



ユーザーガイド

AWS OpsWorks



API バージョン 2013-02-18

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS OpsWorks: ユーザーガイド

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標とトレードドレスは、Amazon 以外の製品またはサービスとの関連において、顧客に混乱を招いたり、Amazon の名誉または信用を毀損するような方法で使用することはできません。Amazon が所有していない他のすべての商標は、それぞれの所有者の所有物であり、Amazon と提携、接続、または後援されている場合とされていない場合があります。

Table of Contents

AWS OpsWorksとは	1
AWS OpsWorks サービス	1
AWS OpsWorks for Puppet Enterprise	4
for Puppet Enterprise OpsWorks のリージョンサポート	5
サポート終了のよくある質問	6
このサポート終了により、既存のお客様にはどのような影響がありますか?	6
何もアクションを実行しない場合サーバーはどうなりますか?	7
新規顧客 AWS OpsWorks for Puppet Enterprise を受け入れていませんか?	7
サポート終了はすべての AWS リージョン に同時に影響しますか?	7
では、どのようなレベルのテクニカルサポートを利用できますか AWS OpsWorks for Puppet Enterprise?	7
私は Puppet Enterprise OpsWorks の現在の顧客であり、以前サービスを使用していなかつ たアカウントでサーバーを起動する必要があります。これはできますか?	7
の新機能リリースはありますか AWS OpsWorks for Puppet Enterprise?	7
概要	8
前提条件	8
Puppet マスターの作成	13
設定を完了する	26
管理するノードを追加する	30
Puppet Enterprise コンソールへのサインイン	33
オプション: を使用する CodeCommit	38
で Puppet Master を作成する CloudFormation	45
前提条件	45
AWS CloudFormationで Puppet Enterprise マスターを作成する	46
カスタムドメインを使用するようにサーバーを更新する	53
前提条件	53
制限事項	54
カスタムドメインを使用するようにサーバーを更新する	54
以下の資料も参照してください。	58
タグを使用する	58
でのタグの仕組み AWS OpsWorks for Puppet Enterprise	59
OpsWorks for Puppet Enterprise (コンソール) でのタグの追加と管理	61
OpsWorks for Puppet Enterprise (CLI) でのタグの追加と管理	64
以下の資料も参照してください。	69

サーバーのバックアップと復元	69
OpsWorks for Puppet Enterprise Server のバックアップ	70
OpsWorks for Puppet Enterprise Server の復元	73
システムメンテナンス	74
システムメンテナンスの設定	76
オンデマンドでのシステムメンテナンスの開始	78
メンテナンス後のカスタム設定およびカスタムファイルの復旧	79
ノードの自動的な追加	79
ステップ 1: インスタンスプロファイルとして使用する IAM ロールを作成する	80
ステップ 2: 自動関連付けスクリプトを使用してインスタンスを作成する	81
ノードの削除	82
以下の資料も参照してください。	83
Puppet マスターの削除	84
ステップ 1: 管理されているノードの関連付けを解除する	84
ステップ 2: サーバーを削除する	85
以下の資料も参照してください。	85
Puppet サーバーを Amazon EC2 に移行する	85
ステップ 1: Puppet に連絡してライセンスを購入してください	86
ステップ 2: OpsWorks for Puppet Enterprise サーバーの詳細を取得する	86
ステップ 3: OpsWorks for Puppet Enterprise サーバーのバックアップを作成する	87
ステップ 4: 新規の EC2 インスタンスを起動する	87
ステップ 5: 新規 EC2 インスタンスに Puppet Enterprise をインストールしてください	89
ステップ 6: 新しい EC2 インスタンスにバックアップを復元してください	89
ステップ 7: Puppet ライセンスを設定してください	90
ステップ 8: ノードを移行してください	90
ステップ 9: OpsWorks for Puppet Enterprise サーバーを削除する	93
の使用 AWS CloudTrail	94
OpsWorks の Puppet エンタープライズ情報用 CloudTrail	94
Puppet Enterprise ログファイルエントリ OpsWorks の理解	95
トラブルシューティング	97
一般的なトラブルシューティングのヒント	98
特定のエラーのトラブルシューティング	98
その他のヘルプとサポート	103
Chef Automate OpsWorks の AWS	104
のリージョンサポート AWS OpsWorks for Chef Automate	107
サポート終了のよくある質問	108

このサポート終了により、既存のユーザーはどのような影響を受けますか？	109
何も操作を実行しない場合サーバーはどうなりますか？	109
どのような代替案に移行できますか？	109
このサービスはまだ新規顧客を受け入れていますか？	109
サポート終了はすべての AWS リージョン に同時に影響しますか？	109
どのレベルのテクニカルサポートが受けられますか？	110
Chef Automate OpsWorks の現在の顧客であり、以前サービスを使用していなかったアカウントでサーバーを起動する必要があります。これはできますか？	110
来年に主要な特徴量のリリースは行われますか？	110
Chef Automate 2 にアップグレードする	110
Chef Automate 2 にアップグレードするための前提条件	110
アップグレードプロセスについて	111
Chef Automate 2 にアップグレードする (コンソール)	111
Chef Automate 2 にアップグレードする (CLI)	112
AWS OpsWorks for Chef Automate サーバーを Chef Automate 1 にロールバックする (CLI)	113
以下の資料も参照してください。	114
概要	115
前提条件	115
Chef Automate サーバーの作成	118
設定を完了してクックブックをアップロードする	131
管理するノードを追加する	141
Chef Automate ダッシュボードにサインインする	148
で Chef Automate サーバーを作成する CloudFormation	151
前提条件	152
AWS CloudFormationで Chef Automate サーバーを作成する	153
カスタムドメインを使用するようにサーバーを更新する	160
前提条件	161
制限事項	54
カスタムドメインを使用するようにサーバーを更新する	54
以下の資料も参照してください。	58
スターターキットの再生成	166
を使用して AWS OpsWorks for Chef Automate スターターキットを再生成する AWS CLI	167
タグを使用する	168
でのタグの仕組み AWS OpsWorks for Chef Automate	169
でのタグの追加と管理 AWS OpsWorks for Chef Automate (コンソール)	170

AWS OpsWorks for Chef Automate (CLI) でのタグの追加と管理	173
以下の資料も参照してください。	178
サーバーのバックアップと復元	178
AWS OpsWorks for Chef Automate サーバーのバックアップ	179
AWS OpsWorks for Chef Automate サーバーの復元	181
システムメンテナンス	183
ノードが AWS OpsWorks 認証機関を信頼していることを確認する	184
システムメンテナンスの設定	185
オンデマンドでのシステムメンテナンスの開始	187
メンテナンス後のカスタム設定およびカスタムファイルの復旧	187
コンプライアンススキャン	188
Chef Automate 2.0 の Compliance	189
Chef Automate 1.x の Compliance	197
Compliance の更新	203
コミュニティとカスタム Compliance プロファイル	203
以下の資料も参照してください。	204
ノードの削除	204
関連トピック	205
Chef Automate サーバーの削除	205
ステップ 1: 管理されているノードの関連付けを解除する	206
ステップ 2: サーバーを削除する	206
Chef 認証情報のリセット	207
の使用 AWS CloudTrail	208
AWS OpsWorks for Chef Automate の情報 CloudTrail	209
AWS OpsWorks for Chef Automate ログファイルエントリについて	210
トラブルシューティング	212
一般的なトラブルシューティングのヒント	212
特定のエラーのトラブルシューティング	213
その他のヘルプとサポート	220
セキュリティ AWS OpsWorks 設定管理 (CM)	221
データ保護	222
との統合 AWS Secrets Manager	223
データ暗号化	224
保管時の暗号化	224
転送時の暗号化	224
キーの管理	224

Identity and Access Management	224
対象者	225
アイデンティティを使用した認証	226
ポリシーを使用したアクセスの管理	229
AWS OpsWorks CM の仕組み IAM	232
アイデンティティベースのポリシーの例	237
トラブルシューティング	241
AWS マネージドポリシー	243
AWS OpsWorks CMにおけるサービス間の混乱した代理の防止	251
インターネットトラフィックのプライバシー	255
ログ記録とモニタリング	255
コンプライアンス検証	255
耐障害性	256
インフラストラクチャセキュリティ	257
設定と脆弱性の分析	258
セキュリティのベストプラクティス	258
AWS OpsWorks スタック	260
スタック	263
レイヤー	264
レシピと LifeCycle イベント	264
インスタンス	265
アプリケーション	266
スタックのカスタマイズ	267
リソース管理	268
セキュリティと権限	268
モニタリングとロギング	268
CLI、SDK、および AWS CloudFormation テンプレート	269
サポート終了のよくある質問	269
このサポート終了により、既存のお客様にはどのような影響がありますか？	270
新規顧客 AWS OpsWorks Stacks を受け入れていますか？	270
既存のスタックはどこに移行すればよいですか？	270
サポート終了後に既存の Amazon EC2 インスタンスを維持するにはどうすればよいですか？	271
サポート終了はすべての AWS リージョン に同時に影響しますか？	271
で利用できるテクニカルサポートのレベル AWS OpsWorks Stacksは？	271
の新機能リリースはありますか AWS OpsWorks Stacks？	271

Systems Manager Application Manager へのアプリケーションの移行	272
スクリプトの仕組み	273
前提条件	273
制限事項	274
開始	274
よくある質問	291
トラブルシューティング	302
AWS OpsWorks Stacks Detach in Place ツールの使用	303
プロセスの仕組み	304
制限事項	306
開始	307
概要	316
リージョンのサポート	316
使用開始: サンプル	317
入門ガイド: Linux	338
入門ガイド: Windows	370
使用開始: クックブック	405
ベストプラクティス	440
ルートデバイスストレージ	441
サーバー数の最適化	443
許可の管理	446
アプリケーションとクックブックの管理とデプロイ	449
ローカルでのクックブックの依存関係のパッケージ化	459
スタック	463
EC2-Classic からスタックを移行する	465
新しいスタックを作成する	468
でのスタックの実行 VPC	477
スタックの更新	489
スタックのクローン化	490
スタックコマンドの実行	492
カスタム の使用 JSON	495
スタックの削除	498
レイヤー	502
OpsWorks レイヤーの基本	503
Elastic ロードバランシングレイヤー	519
Amazon RDS Service Layer	524

ECS クラスターレイヤー	530
カスタムレイヤー	537
レイヤーごとのパッケージのインストール	538
インスタンス	540
AWS OpsWorks スタックインスタンスの使用	541
AWS OpsWorks Stacks の外部で作成されたコンピューティングリソースの使用	603
インスタンス設定の編集	652
AWS OpsWorks スタックインスタンスの削除	654
SSH でのログイン	656
RDP でのログイン	660
アプリケーション	664
アプリケーションの追加	665
アプリケーションのデプロイ	673
アプリケーションの編集	677
データベースへの接続	678
環境変数の使用	680
アプリケーションへのデータの引き渡し	682
Git リポジトリの SSH キーの使用	686
カスタムドメインの使用	687
SSL の使用	689
クックブックとレシピ	697
クックブックリポジトリ	698
Chef のバージョン	702
Ruby のバージョン	721
カスタムクックブックのインストール	723
カスタムクックブックの更新	727
レシピの実行	729
リソース管理	737
スタックにリソースを登録する	739
リソースのアタッチと移動を行う	745
リソースをデタッチする	751
リソースの登録を解除する	754
タグ	756
スタックレベルでタグを設定	757
レイヤーレベルでのタグの設定	759
を使用したタグの管理 AWS CLI	761

タグの制限	762
モニタリング	763
Amazon の使用 CloudWatch	763
の使用 AWS CloudTrail	775
Amazon CloudWatch Logs の使用	778
Amazon CloudWatch Events の使用	783
セキュリティと権限	784
ユーザー許可の管理	786
AWS OpsWorks スタックがユーザーに代わって動作することを許可する	811
混乱した代理の防止	816
EC2 インスタンスで実行されるアプリケーションのアクセス許可の指定	820
SSH アクセスの管理	824
セキュリティ更新の管理	832
セキュリティグループの使用	834
Chef 12 Linux	837
概要	838
Chef 12 への移行	839
サポートされるオペレーティングシステム	840
サポートされるインスタンスタイプ	840
詳細情報	840
データバッグへの移行	841
以前の Chef のバージョン	843
Linux 用 Chef 11.10 以前のバージョン	843
他の AWS サービスでの AWS OpsWorks スタックの使用	1281
バックエンドデータストアの使用	1283
ElastiCache Redis	1291
Amazon S3 バケットの使用	1305
AWS OpsWorks スタック AWS CodePipeline での の使用	1320
AWS OpsWorks スタック CLI の使用	1384
インスタンスの作成	1387
アプリケーションのデプロイ	1390
アプリケーションのリストを取得する	1391
リストコマンド	1392
デプロイメントのリストを取得する	1394
Elastic IP アドレスのリストの取得	1395
インスタンスのリストを取得する	1396

スタックのリストを取得する	1397
レイヤーのリストを取得する	1399
レシピの実行	1403
Install Dependencies	1404
スタック設定の更新	1405
デバッグとトラブルシューティングのガイド	1406
レシピのデバッグ	1407
一般的なデバッグとトラブルシューティングの問題	1425
AWS OpsWorks スタックエージェント CLI	1435
agent_report	1438
get_json	1438
instance_report	1443
list_commands	1444
run_command	1445
show_log	1446
stack_state	1447
AWS OpsWorks スタックデータバグリファレンス	1450
アプリケーションデータバグ (aws_opsworks_app)	1454
コマンドデータバグ (aws_opsworks_command)	1458
Amazon ECS クラスターデータバグ (aws_opsworks_ecs_cluster)	1460
Elastic Load Balancing データバグ (aws_opsworks_elastic_load_balancer)	1461
インスタンスデータバグ (aws_opsworks_instance)	1462
レイヤーデータバグ (aws_opsworks_layer)	1467
Amazon RDS データバグ (aws_opsworks_rds_db_instance)	1469
スタックデータバグ (aws_opsworks_stack)	1471
ユーザーデータバグ (aws_opsworks_user)	1473
OpsWorks エージェントの変更	1475
Chef 12 エージェントリリース	1475
Chef 11.10 エージェントリリース	1479
リソース	1485
リファレンスガイド、ツール、およびサポートリソース	1485
AWS ソフトウェア開発キット	1486
ソースソフトウェアを開く	1487
AWS OpsWorks ドキュメント履歴	1488
以前の更新	1496
.....	md

AWS OpsWorksとは

Important

AWS OpsWorks サービスはサポートが終了し、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks は、Puppet または Chef. AWS OpsWorks Stacks を使用してクラウドエンタープライズでアプリケーションを設定および運用するのに役立つ設定管理サービスです。また、[Chef](#) クックブックとソリューションを使用して設定を管理 AWS OpsWorks for Chef Automate できます。一方、OpsWorks for Puppet Enterprise では、[Puppet Enterprise](#) マスターサーバーを設定できます AWS。Puppet は、インフラストラクチャの目的の状態の強化、およびオンデマンドタスクの自動化のための、一連のツールを提供します。

AWS OpsWorks サービス

[AWS OpsWorks for Puppet Enterprise](#)

OpsWorks for Puppet Enterprise では、AWS管理の Puppet マスターサーバーを作成できます。Puppet マスターは、インフラストラクチャ内のノードを管理し、それらのノードに関する情報を保存し、Puppet モジュールの中央リポジトリとして機能します。モジュールは、インフラストラクチャの設定方法に関する手順が保存された Puppet コードの再利用および共有が可能なユニットです。[Puppet Forge](#) からコミュニティモジュールをダウンロードすることも、Puppet 開発キットを使用して独自のカスタムモジュールを作成することもできます。その後、Puppet Code Manager を使用してデプロイを管理できます。

OpsWorks for Puppet Enterprise は、フルマネージド型の Puppet マスター、アプリケーションの検査、配信、運用、および将来の保護を可能にする自動化ツールのスイート、およびノードと Puppet アクティビティに関する情報を表示できるユーザーインターフェイスへのアクセスを提供します。OpsWorks for Puppet Enterprise では、Puppet を使用して、ノードが Amazon EC2 インスタンスであるかオンプレミスデバイスであるかにかかわらず、ノードの設定、デプロイ、管理方法を自動化できます。OpsWorks for Puppet Enterprise マスターは、ソフトウェアとオペレーティングシステムの設定、パッケージのインストール、データベースのセットアップ、変更

管理、ポリシーの適用、モニタリング、品質保証などのタスクを処理するフルスタックのオートメーションを提供します。

OpsWorks for Puppet Enterprise は Puppet Enterprise ソフトウェアを管理するため、選択した時点でサーバーを自動的にバックアップでき、は常に最新の AWS 互換バージョンの Puppet を実行し、常に最新のセキュリティ更新プログラムが適用されます。新しい Amazon EC2 Auto Scaling グループを使用して、新しい Amazon EC2 ノードを自動的にサーバーに関連付けることができます。

[Chef Automate OpsWorks の AWS](#)

AWS OpsWorks for Chef Automate では、Chef [Automate](#) のプレミアム機能を含む AWS 管理の Chef サーバーを作成し、Chef DK やその他の Chef ツールを使用してそれらを管理できます。Chef サーバーは、環境内のノードを管理し、それらのノードに関する情報を保存し、Chef クックブックの中央リポジトリとして機能します。クックブックには、Chef を使用して管理する各ノードに対して Chef Infra クライアント (chef-client) エージェントによって実行されるレシピが含まれています。[knife](#) や [Test Kitchen](#) などの Chef ツールを使用して、AWS OpsWorks for Chef Automate サービスの Chef サーバー上のノードとクックブックを管理できます。

Chef Automate は、継続的なデプロイとコンプライアンスチェックのための自動ワークフローを提供する、含まれているサーバーソフトウェアパッケージです。AWS OpsWorks for Chef Automate は、単一の Amazon Elastic Compute Cloud インスタンスを使用して Chef Automate、Chef Infra、Chef をインストールおよび管理 InSpec します。を使用すると AWS OpsWorks for Chef Automate、AWS OpsWorks 固有の変更を加えることなく、コミュニティで作成された Chef クックブックまたはカスタム Chef クックブックを使用できます。

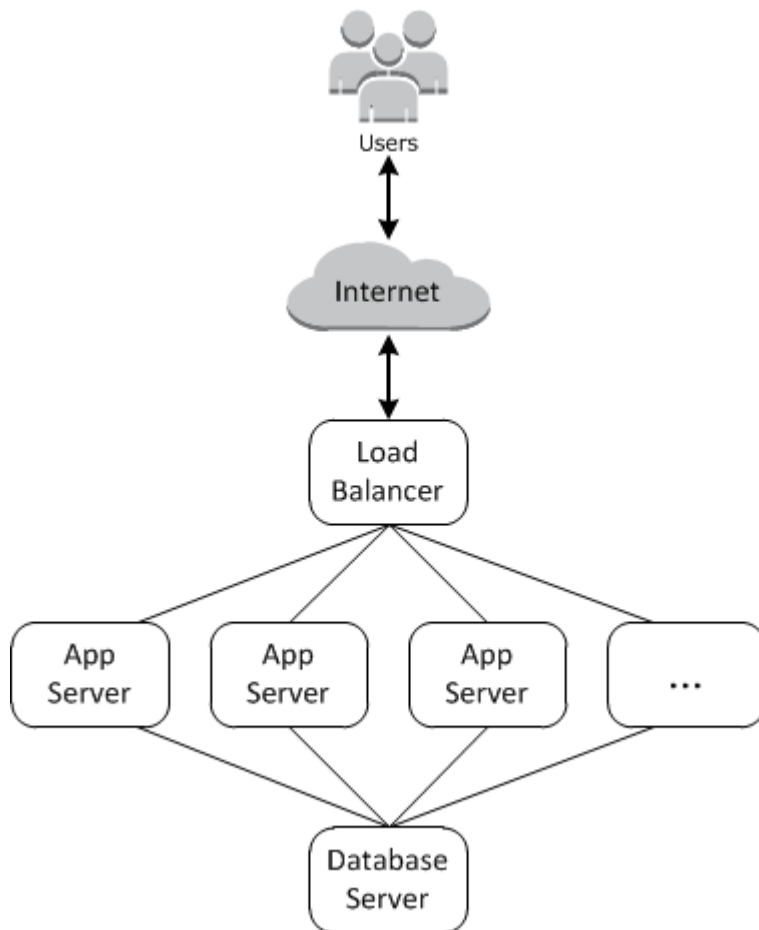
は 1 つのインスタンスで Chef Automate コンポーネント AWS OpsWorks for Chef Automate を管理するため、サーバーは選択した時点で自動的にバックアップでき、は常に最新のマイナーバージョンの Chef を実行し、常に最新のセキュリティ更新が適用されます。新しい Amazon EC2 Auto Scaling グループを使用して、新しい Amazon EC2 ノードを自動的にサーバーに関連付けることができます。

[AWS OpsWorks スタック](#)

クラウドベースのコンピューティングには通常、EC2 インスタンスや Amazon Relational Database Service (RDS) インスタンスなどの AWS リソースのグループが関与します。たとえば、ウェブアプリケーションでは通常、アプリケーションサーバー、データベースサーバー、ロードバランサー、その他のリソースが必要です。このインスタンスのグループは通常、スタックと呼ばれます。

AWS OpsWorks 元のサービスである スタックは、スタックとアプリケーションを作成および管理するためのシンプルで柔軟な方法を提供します。AWS OpsWorks スタックを使用すると、スタックにアプリケーションをデプロイしてモニタリングできます。レイヤーと呼ばれる、特別なグループ内のクラウドリソースの管理に役立つスタックを作成できます。レイヤーは、アプリケーションへのサービス提供やデータベースサーバーのホスティングなどの特定の目的を果たす一連の EC2 インスタンスを表します。レイヤーでは、[Chef レシピ](#)に基づいて、インスタンスへのパッケージのインストール、アプリケーションのデプロイ、スクリプトの実行などのタスクが処理されます。

とは異なり AWS OpsWorks for Chef Automate、AWS OpsWorks スタックは Chef サーバーを要求または作成しません。AWS OpsWorks スタックは Chef サーバーの作業の一部を実行します。AWS OpsWorks スタックは、インスタンスのヘルスチェックをモニタリングし、自動ヒーリングおよび Auto Scaling を使用して、必要な場合に新しいインスタンスのプロビジョニングを実行します。シンプルなアプリケーションサーバースタックの例を次の図に示します。



AWS OpsWorks for Puppet Enterprise

⚠ Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

OpsWorks for Puppet Enterprise では、[Puppet Enterprise](#) マスターを数分で起動でき、はオペレーション、バックアップ、復元、ソフトウェアアップグレードを AWS OpsWorks 処理できます。

OpsWorks for Puppet Enterprise では、Puppet マスターを管理する代わりに、コア設定管理タスクに集中できます。OpsWorks for Puppet Enterprise を使用すると、同じ設定を使用してオンプレミスインフラストラクチャとクラウドインフラストラクチャの両方を管理できるため、ハイブリッド環境でオペレーションを効率的にスケーリングできます。Puppet マスターサーバーの管理は、Puppet Enterprise コンソール、AWS Management Console、および AWS CLI ツールを使用して簡素化されます。

Puppet マスターは、特定のノードの設定カタログを [puppet-agent](#) ソフトウェアに提供することで、ご使用の環境のノードの設定を管理します。また、Puppet モジュールの中央リポジトリとして機能します。OpsWorks for Puppet Enterprise の Puppet マスターはマネージドノード puppet-agent にデプロイされ、Puppet Enterprise のプレミアム機能を提供します。

OpsWorks for Puppet Enterprise マスターは Amazon Elastic Compute Cloud インスタンスで実行されます。OpsWorks for Puppet Enterprise サーバーは、最新バージョンの Amazon Linux (Amazon Linux 2) と最新バージョンの Puppet Enterprise Master バージョン 2019.8.5 を実行するように設定されています。Puppet Enterprise 2019.8.5 の変更の詳細については、「[Puppet Enterprise リリースノート](#)」を参照してください。

Puppet ソフトウェアの新しいバージョンが利用可能になると、システムメンテナンスは AWS テストが完了次第、サーバーにある Puppet Enterprise サーバーのバージョンを自動更新するように設計されています。AWS は Puppet のアップグレードが本番稼働で利用できるものが確認するため広範囲に渡るテストを行い、ユーザーの既存環境を中断しません。

サポートされているオペレーティングシステムを実行しており、OpsWorks for Puppet Enterprise マスターへのネットワークアクセス権を持つオンプレミスのコンピュータまたは EC2 インスタンスに

接続できます。[puppet](#) エージェントソフトウェアは、Puppet マスターによって管理するノードにインストールされます。

トピック

- [for Puppet Enterprise OpsWorks のリージョンサポート](#)
- [AWS OpsWorks for Puppet Enterprise サポート終了FAQs](#)
- [OpsWorks for Puppet Enterprise の開始方法](#)
- [を使用して AWS OpsWorks for Puppet Enterprise マスターを作成する AWS CloudFormation](#)
- [カスタムドメインを使用するように OpsWorks for Puppet Enterprise Server を更新する](#)
- [AWS OpsWorks for Puppet Enterprise リソースでのタグの使用](#)
- [OpsWorks for Puppet Enterprise Server のバックアップと復元](#)
- [Puppet Enterprise OpsWorks のでのシステムメンテナンス](#)
- [OpsWorks for Puppet Enterprise でノードを自動的に追加する](#)
- [OpsWorks for Puppet Enterprise Server からノードの関連付けを解除する](#)
- [OpsWorks for Puppet Enterprise Server を削除する](#)
- [OpsWorks for Puppet Enterprise サーバーを Amazon Elastic Compute Cloud \(Amazon EC2\) に移行する方法](#)
- [OpsWorks を使用した Puppet Enterprise API コールのログ記録 AWS CloudTrail](#)
- [Puppet Enterprise OpsWorks のトラブルシューティング](#)

for Puppet Enterprise OpsWorks のリージョンサポート

以下のリージョンエンドポイント OpsWorks は、Puppet Enterprise masters をサポートしています。OpsWorks for Puppet Enterprise は、インスタンスプロファイル、ユーザー、サービスロールなど、Puppet マスターに関連付けられたリソースを Puppet マスターと同じリージョンエンドポイントに作成します。Puppet マスターは VPC 内にある必要があります。作成するかすでに作成してある VPC を使用することも、デフォルトの VPC を使用することもできます。

- 米国東部 (オハイオ) リージョン
- 米国東部(バージニア州北部) リージョン
- US West (N. California) Region
- 米国西部 (オレゴン) リージョン

- Asia Pacific (Tokyo) Region
- アジアパシフィック (シンガポール) リージョン
- アジアパシフィック (シドニー) リージョン
- 欧州 (フランクフルト) リージョン
- 欧州 (アイルランド) リージョン

AWS OpsWorks for Puppet Enterprise サポート終了FAQs

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

トピック

- [このサポート終了により、既存のお客様にはどのような影響がありますか？](#)
- [何もアクションを実行しない場合サーバーはどうなりますか？](#)
- [新規顧客 AWS OpsWorks for Puppet Enterprise を受け入れていますか？](#)
- [サポート終了はすべての AWS リージョン に同時に影響しますか？](#)
- [では、どのようなレベルのテクニカルサポートを利用できますか AWS OpsWorks for Puppet Enterprise？](#)
- [私は Puppet Enterprise OpsWorks の現在の顧客であり、以前サービスを使用していなかったアカウントでサーバーを起動する必要があります。これはできますか？](#)
- [の新機能リリースはありますか AWS OpsWorks for Puppet Enterprise？](#)

このサポート終了により、既存のお客様にはどのような影響がありますか？

既存のお客様は、2024 年 3 月 31 日、OpsWorks for Puppet Enterprise のサポート終了日まで影響を受けません。サポート終了日を過ぎると、お客様は OpsWorks コンソールまたは API を使用してサーバーを管理できなくなります。

何もアクションを実行しない場合サーバーはどうなりますか？

2024年3月31日以降、OpsWorks コンソールまたは API を使用してサーバーを管理することはできなくなります。その時点で、バックアップやメンテナンスなど、サーバーの継続的な管理機能の実行は停止されます。お客様への影響を制限するために、Puppet Enterprise サーバーをバックアップする EC2 インスタンスは実行したままになりますが、使用が OpsWorks for Puppet Enterprise サービス契約の対象 (または請求) ではなくなったため、ライセンスは無効になります。Puppet Enterprise でインフラストラクチャを管理し続ける場合は、[「for Puppet Enterprise サーバーを Amazon Elastic Compute Cloud \(Amazon EC2\) に移行する方法 OpsWorks」](#) を参照してください。

新規顧客 AWS OpsWorks for Puppet Enterprise を受け入れていませんか？

いいえ。AWS OpsWorks for Puppet Enterprise は新規顧客を受け入れなくなりました。

サポート終了はすべての AWS リージョン に同時に影響しますか？

はい。API とコンソールは 2024年3月31日をもってサポート終了となり、すべてのリージョンで使用できなくなります。AWS OpsWorks for Puppet Enterprise が利用可能な のリストについては、AWS リージョン [AWS 「リージョンサービスリスト」](#) を参照してください。

では、どのようなレベルのテクニカルサポートを利用できますか AWS OpsWorks for Puppet Enterprise ？

AWS は、サポート終了日まで、お客様が現在持ってい AWS OpsWorks for Puppet Enterprise のと同じレベルのサポートを引き続き提供します。ご質問やご不明点がございましたら、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

私は Puppet Enterprise OpsWorks の現在の顧客であり、以前サービスを使用していなかったアカウントでサーバーを起動する必要があります。これはできますか？

例外的な事情がない限り、一般的にはできません。特別な状況が発生した場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームに連絡し、その詳細と根拠を伝えて、リクエストを確認します。

の新機能リリースはありますか AWS OpsWorks for Puppet Enterprise ？

いいえ。サービスのサポート終了が近づいているため、新特徴量のリリースは行いません。しかし、サポート終了日までは、引き続きセキュリティの改善とサーバーの管理を予定どおりに行います。

OpsWorks for Puppet Enterprise の開始方法

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

OpsWorks for Puppet Enterprise では、[Puppet Enterprise](#) サーバーを実行できます AWS。約 15 分で Puppet Enterprise マスターサーバーをプロビジョニングできます。

2021 年 5 月 3 日以降、OpsWorks for Puppet Enterprise は一部の Puppet Enterprise サーバー属性を保存します AWS Secrets Manager。詳細については、「[との統合 AWS Secrets Manager](#)」を参照してください。

以下のチュートリアルは、for Puppet Enterprise で最初の Puppet OpsWorks マスターを作成するのに役立ちます。

前提条件

開始する前に、以下の前提条件を満たしている必要があります

トピック

- [Puppet 開発キットをインストールします。](#)
- [Puppet Enterprise クライアントツールをインストールします。](#)
- [Git コントロールリポジトリのセットアップ](#)
- [VPC のセットアップ](#)
- [EC2 のキーペアをセットアップする \(オプション\)](#)
- [カスタムドメインを使用するための前提条件 \(オプション\)](#)

Puppet 開発キットをインストールします。

1. Puppet ウェブサイトから、ローカルコンピュータのオペレーティングシステムに一致する [Puppet 開発キットをダウンロード](#) します。

2. Puppet 開発キットをインストールします。
3. Puppet 開発キットをローカルコンピュータの PATH 変数に追加します。
 - Linux または macOS オペレーティングシステムでは、Bash シェルで次のコマンドを実行して、Puppet 開発キットを PATH 変数に追加できます。

```
echo 'export PATH=/opt/puppetlabs/pdk/bin/pdk:$PATH' >> ~/.bash_profile && source
~/.bash_profile
```

- Windows ベースのオペレーティングシステムでは、PowerShell セッションで次の .NET Framework コマンドを使用するか、システムプロパティ からアクセス可能な環境変数ダイアログボックスを使用して、Puppet Development Kit を PATH 変数に追加できます。次のコマンドを実行するには、管理者として PowerShell セッションを実行する必要がある場合があります。

```
[Environment]::SetEnvironmentVariable("Path","new path value","Machine")
```

Puppet Enterprise クライアントツールをインストールします。

Puppet Enterprise (PE) クライアントツールは、ワークステーションから Puppet Enterprise サービスにアクセスできるようにする一連のコマンドラインツールです。ツールはさまざまなオペレーティングシステムにインストールできます。また、Puppet を使用して管理しているノードにもインストールできます。このツールをサポートするオペレーティングシステムに関する情報およびインストール方法については、Puppet Enterprise ドキュメントの「[Installing PE client tools](#)」を参照してください。

Git コントロールリポジトリのセットアップ

Puppet マスターを起動する前に、Git 内に Puppet モジュールおよびクラスを保存および変更管理するコントロールリポジトリを設定する必要があります。Puppet Enterprise マスターサーバーを起動するステップでは、Git リポジトリの URL とリポジトリにアクセスするための HTTPS または SSH アカウント情報を求められます。Puppet Enterprise マスターで使用するコントロールリポジトリをセットアップする方法の詳細については、「[Setting up a control repository](#)」を参照してください。コントロールリポジトリの設定手順については、の Puppet [control-repoのサンプルリポジトリの readme](#) を参照してください [GitHub](#)。コントロールリポジトリの構造は次のような内容です。

```
### LICENSE
### Puppetfile
```

```
### README.md
### environment.conf
### hieradata
#   ### common.yaml
#   ### nodes
#       ### example-node.yaml
### manifests
#   ### site.pp
### scripts
#   ### code_manager_config_version.rb
#   ### config_version.rb
#   ### config_version.sh
### site
### profile
#   ### manifests
#       ### base.pp
#       ### example.pp
### role
### manifests
    ### database_server.pp
    ### example.pp
    ### webserver.pp
```

を使用したリポジトリのセットアップ CodeCommit

を使用して新しいリポジトリを作成できます CodeCommit。CodeCommit を使用してコントロールリポジトリを作成する方法の詳細については、このガイド [the section called “オプション: 使用する CodeCommit”](#) の「」を参照してください。で Git の使用を開始する方法の詳細については CodeCommit、[「AWS の開始方法 CodeCommit」](#) を参照してください。リポジトリ OpsWorks の for Puppet Enterprise サーバーを承認するには、AWSCodeCommitReadOnlyポリシーを IAM インスタンスプロファイルロールにアタッチします。

VPC のセットアップ

OpsWorks for Puppet Enterprise マスターは、Amazon Virtual Private Cloud で動作する必要があります。既存の VPC にそのサーバーを追加するか、デフォルトの VPC を使用するか、またはそのサーバーを含めて新しい VPC を作成します。Amazon VPC の情報および新しい VPC の作成方法については、[「Amazon VPC 入門ガイド」](#) を参照してください。

独自の VPC を作成するか、既存の VPC を使用する場合、VPC には次の設定またはプロパティが必要です。

- VPC には少なくとも 1 つのサブネットが必要です。

OpsWorks for Puppet Enterprise マスターがパブリックアクセス可能になる場合は、サブネットをパブリックにして、パブリック IP の自動割り当てを有効にします。

- [DNS resolution] は有効である必要があります。
- サブネットで、[Auto-assign public IP] を有効にします。

VPCs の作成やインスタンスの実行に慣れていない場合は、次の AWS CLI コマンドを実行して、AWS OpsWorks が提供する AWS CloudFormation テンプレートを使用して、単一のパブリックサブネットを持つ VPC を作成できます。を使用する場合は AWS Management Console、[テンプレート](#)を AWS CloudFormation コンソールにアップロードすることもできます。

```
aws cloudformation create-stack --stack-name OpsWorksVPC --template-url https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-vpc.yaml
```

EC2 のキーペアをセットアップする (オプション)

SSH 接続は、Puppet サーバーの一般的な管理に必要なまたは推奨されません。AWS Management Console および AWS CLI コマンドを使用して、Puppet サーバーで多くの管理タスクを実行できます。

Puppet Enterprise ウェブベースコンソールのサインインパスワードがわからなくなった場合や変更する場合には、SSH を使用してサーバーに接続する際に EC2 のキーペアが必要です。その場合は、既存のキーペアを使用するか、または新しいキーペアを作成できます。新しい EC2 キーペア作成方法の詳細については、[「Amazon EC2 Key Pairs」](#) (Amazon EC2 のキーペア) を参照してください。

EC2 のキーペアが必要なければ、Puppet Enterprise マスターを作成する準備ができています。

カスタムドメインを使用するための前提条件 (オプション)

独自のドメインで Puppet Enterprise マスターをセットアップし、サーバーのエンドポイントとして使用するカスタムドメインのパブリックエンドポイントを指定できます。カスタムドメインを使用する場合は、このセクションで詳しく説明するように、次のすべてが必要です。

トピック

- [カスタムドメインをセットアップする](#)
- [証明書を取得する](#)
- [プライベートキーを取得する](#)

カスタムドメインをセットアップする

独自のカスタムドメインで Puppet Enterprise マスターを実行するには、`https://aws.my-company.com` などサーバーのパブリックエンドポイントが必要です。カスタムドメインを指定する場合は、前のセクションで説明したように、証明書とプライベートキーも指定する必要があります。

作成後にサーバーにアクセスするには、優先 DNS サービスに CNAME DNS レコードを追加します。このレコードは、Puppet マスターの作成プロセスで生成されるエンドポイント (サーバーの Endpoint 属性の値) にカスタムドメインをポイントしている必要があります。サーバーがカスタムドメインを使用している場合は、生成された Endpoint 値を使用してサーバーにアクセスできません。

証明書を取得する

独自のカスタムドメインで Puppet マスターをセットアップするには、PEM 形式の HTTPS 証明書が必要です。これは、単一の自己署名証明書、または証明書チェーンです。[Create Puppet Enterprise Master (Puppet Enterprise マスター の作成)] ワークフローを完了するときに、この証明書を指定する場合は、カスタムドメインとプライベートキーも指定する必要があります。

証明書の値の要件は次のとおりです。

- 自己署名証明書、カスタム証明書、または完全な証明書チェーンを指定できます。
- 証明書は、有効な X509 証明書、または PEM 形式の証明書チェーンである必要があります。
- 証明書はアップロード時に有効である必要があります。有効期間の開始 (証明書の NotBefore 日付) 前、または有効期間の終了 (証明書の NotAfter 日) 後に証明書を使用することはできません。
- 証明書の共通名またはサブジェクトの代替名 (SAN) が存在する場合は、カスタムドメインの値と一致する必要があります。
- 証明書は、[Custom private key (カスタムプライベートキー)] フィールドの値と一致する必要があります。

プライベートキーを取得する

独自のカスタムドメインで Puppet マスターをセットアップするには、HTTPS を使用してサーバーに接続するための PEM 形式のプライベートキーが必要です。プライベートキーは暗号化しないでください。パスワードやパスフレーズで保護することはできません。カスタムプライベートキーを指定する場合は、カスタムドメインと証明書も指定する必要があります。

Puppet Enterprise マスターの作成

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。


Puppet Enterprise コンソール OpsWorks の または を使用して、Puppet マスターを作成できます AWS CLI。

トピック

- [を使用して Puppet Enterprise Master を作成する AWS Management Console](#)
- [を使用して Puppet Enterprise Master を作成する AWS CLI](#)

を使用して Puppet Enterprise Master を作成する AWS Management Console


1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/opsworks/> で AWS OpsWorks コンソールを開きます。
2. AWS OpsWorks ホームページで、「Puppet Enterprise の に移動 OpsWorks 」を選択します。



AWS OpsWorks

AWS OpsWorks is a configuration management service that helps you build and operate highly dynamic applications, and propagate changes instantly.

AWS OpsWorks provides three solutions to configure your infrastructure:




OpsWorks Stacks

Define, group, provision, deploy, and operate your applications in AWS by using Chef in local mode.

[Go to OpsWorks Stacks](#)

[Learn more about OpsWorks Stacks](#)




OpsWorks for Chef Automate

Create Chef servers that include Chef Automate premium features, and use the Chef DK or any Chef tooling to manage them.

[Go to OpsWorks for Chef Automate](#)

[Learn more about OpsWorks for Chef Automate](#)



OpsWorks for Puppet Enterprise

Create Puppet servers that include Puppet Enterprise features. Inspect, deliver, update, monitor, and secure your infrastructure.

[Go to OpsWorks for Puppet Enterprise](#)

[Learn more about OpsWorks for Puppet Enterprise](#)

- for Puppet Enterprise ホームページで、「Puppet Enterprise サーバーの作成 OpsWorks 」を選択します。

Welcome to OpsWorks for Puppet Enterprise

OpsWorks for Puppet Enterprise helps you automate, provision, and configure your environment.

Puppet automatically keeps everything in its desired state, enforcing consistency and keeping you compliant, while giving you complete control to make changes as your business needs evolve. [Learn more.](#)

[Create Puppet Enterprise server](#)

- [Set name, region, and type] ページで、サーバーの名前を指定します。Puppet マスター名は 40 文字以内で、文字で始まる必要があります。英数字とハイフンのみを使用できます。サポートされているリージョンを選択し、管理するノード数をサポートしているインスタンスタイプを選択します。サーバーを作成した後も、必要に応じてインスタンスタイプを変更できます。このチュートリアルでは、米国西部 (オレゴン) リージョンに [m5.xlarge] のインスタンスタイプを作成します。[次へ] をクリックします。

Set name, region, and type

Type a name for the Puppet Enterprise server, select the region in which you want to locate the server, and select the Amazon EC2 instance type that best fits your needs.

Puppet Enterprise server name ⓘ
 Maximum 40 characters. Has to start with a letter, and can only contain letters, numbers, and hyphens.

Puppet Enterprise server region ⓘ

EC2 instance type

m5.xlarge 16 GiB Memory Supports up to 450 nodes	c5.2xlarge 16 GiB Memory Supports up to 800 nodes	c5.4xlarge 32 GiB Memory Supports 1600+ nodes
---	--	--

- [Configure server (サーバーの構成)] ページで、キーペア名を指定しない場合は、[SSH key (SSH キー)] ドロップダウンリストでデフォルトの選択のままにします。[Configure Puppet Code Manager] 領域の [r10k remote] フィールドで、Git リモートの有効な SSH URL 値を指定します。r10k プライベートキーフィールドで、AWS OpsWorks を使用して r10k リモートリポジトリにアクセスできる SSH プライベートキーを貼り付けます。これはプライベートリポジトリを作成するときに Git によって提供されますが、HTTPS 認証を使用してコントロールリポジトリにアクセスしている場合は必要ありません。[次へ] をクリックします。

Configure server

Configure EC2, Puppet credentials and server endpoint.

Select an SSH key

Select the EC2 key pair. You will need this key to connect to the Puppet Enterprise server.

SSH key ⓘ

We recommend to use the Puppet Enterprise client tools, which is a set of command line tools that let you access Puppet Enterprise services from a workstation without SSH access.

Configure Puppet Code Manager

Select the Puppet control repository that you want to use to deploy modules.

R10K Remote ⓘ

r10k remote URL - the URL of your control repository (e.g. ssh://git@your.git-repo.com:user/control-repo.git)

R10K Private Key ⓘ

If you are using a private Git repository, specify an SSH URL and a PEM-encoded private SSH key.

6. サーバーを独自のカスタムドメインで使用しない場合は、[Specify server endpoint (サーバーエンドポイントを指定)] をデフォルトの [Use an automatically-generated endpoint (自動的に生成されたエンドポイントを指定する)] のままにして、[Next (次へ)] を選択します。カスタムドメインを設定するには、次のステップに進みます。
7. カスタムドメインを使用するには、[Specify server endpoint (サーバーエンドポイントを指定)] で、ドロップダウンリストから [Use a custom domain (カスタムドメインを使用)] を選択します。
 - a. [Fully qualified domain name (FQDN) (完全修飾ドメイン名 (FQDN))] で、FQDN を指定します。使用するドメイン名を所有している必要があります。
 - b. [SSL certificate] (SSL 証明書) に、PEM 形式の証明書全体を貼り付けます。この証明書は -----BEGIN CERTIFICATE----- で始まり、-----END CERTIFICATE----- で終わります。SSL 証明書のサブジェクトは、前のステップで入力した FQDN と一致する必要があります。証明書の前後の余分な行を削除します。
 - c. [SSL private key] (SSL プライベートキー) には、RSA プライベートキー全体を貼り付けます。このプライベートキーは -----BEGIN RSA PRIVATE KEY----- で始まり、-----END RSA PRIVATE KEY----- で終わります。SSL プライベートキーは、前のステップで入力した SSL 証明書のパブリックキーと一致する必要があります。プライベートキーの前後の余分な行を削除します。[次へ] をクリックします。
8. 詳細設定の設定ページのネットワークとセキュリティエリアで、VPC、サブネット、および 1 つ以上のセキュリティグループを選択します。使用するセキュリティグループ、サービスロール、およびインスタンスプロファイルがまだない場合は、 で生成 AWS OpsWorks できます。サーバーは複数のセキュリティグループのメンバーにできます。このページから次に進んだ後に、Puppet マスターのネットワーク設定およびセキュリティ設定を変更することはできません。

Network and security

You cannot change network and security settings after you launch your Puppet Enterprise server.

VPC vpc-27cdf143 - LinuxAMIVPC ⓘ

You have selected a non-default VPC. Be sure the selected VPC has outbound network access. [Learn more.](#)

Subnet 10.0.0.0/24 - us-west-2a - Public subnet ⓘ

Associate Public IP Address Yes No

Choose Yes if the selected subnet is public.

Security groups *Select a security group to add* ⓘ

sg-0 ✕

sg-1 ✕

Please ensure the following ports are open: 443 (https), 4433 (PE API Endpoint), 8140 (PE Master API), 8142/8143 (PE Orchestrator), 8170 (Code Manager)

Service role aws-opsworks-cm-service-role ⓘ

Instance profile aws-opsworks-cm-ec2-role ⓘ

9. [System maintenance] セクションで、システムメンテナンスを開始する日付と時刻を設定します。システムメンテナンス中はサーバーがオフラインになることを想定する必要があるため、通常の営業時間内でサーバー需要が低い時刻を選択します。

メンテナンス時間は必須です。、、または APIs を使用して AWS Management Console AWS CLI、開始日時を後で変更できます。

System maintenance

AWS OpsWorks installs updates for Puppet Enterprise minor versions or security packages in the time range and on the weekday that you specify here. **Your Puppet Enterprise server will be offline during system maintenance.**

Start day Monday ⓘ

Start time (UTC) 6 pm - 7 pm ⓘ

10. バックアップを設定します。デフォルトでは、自動バックアップが有効になっています。自動バックアップを開始する頻度と時刻を設定し、Amazon Simple Storage Service に保存するバックアップの世代数を設定します。最大 30 個のバックアップを保持できます。最大値に達すると、OpsWorks Puppet Enterprise は最も古いバックアップを削除して新しいバックアップ用のスペースを確保します。

Automated backup

AWS OpsWorks supports two ways to back up your Puppet Enterprise server: manual or automated. Backups are uploaded to your Amazon S3 bucket. If you ever need to restore your Puppet Enterprise server, you can restore it by applying a backup that you choose.

Enable automated backup Yes No

Frequency

Start time (UTC)

Number of generations to keep

Specify how many automated backups to keep. Minimum: 1, maximum: 30.

[Cancel](#) [Previous](#) [Next](#)

11. (オプション) [Tags] (タグ) で、サーバーおよび関連リソース (EC2 インスタンス、Elastic IP アドレス、セキュリティグループ、S3 バケット、バックアップなど) にタグを追加します。OpsWorks for Puppet Enterprise サーバーのタグ付けの詳細については、「」を参照してください [AWS OpsWorks for Puppet Enterprise リソースでのタグの使用](#)。
12. 詳細設定が完了したら、[Next] を選択します。
13. [確認] ページで選択内容を確認します。サーバーを作成する準備ができたなら、[Launch] を選択します。

が Puppet マスターを作成する AWS OpsWorks のを待っている間に、に進み、Starter Kit [スターターキット](#)を使用して [Puppet マスターを設定する](#)と Puppet Enterprise コンソールの認証情報をダウンロードします。これらをダウンロードするために、サーバーがオンラインになるまで待つ必要はありません。

サーバーの作成が完了すると、Puppet マスターは OpsWorks for Puppet Enterprise のホームページでオンライン のステータスになります。サーバーがオンラインになると、Puppet Enterprise コンソールがサーバーのドメインで `https://your_server_name-randomID.region.opsworks-cm.io` の形式の URL で利用できます。

を使用して Puppet Enterprise Master を作成する AWS CLI

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合

は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS CLI コマンドを実行して OpsWorks for Puppet Enterprise マスターサーバーを作成することは、コンソールでのサーバーの作成とは異なります。使用する既存のロールを指定しない場合、コンソールでサービスロールとセキュリティグループ AWS OpsWorks が作成されます。では AWS CLI、AWS OpsWorks はセキュリティグループを指定しない場合に作成できますが、サービスロールは自動的に作成されません。create-server コマンドの一部としてサービスロール ARN を指定する必要があります。コンソールで、AWS OpsWorks が Puppet マスターを作成している間、Puppet Enterprise コンソールのスターターキットとサインイン認証情報をダウンロードします。を使用して OpsWorks for Puppet Enterprise マスターを作成する場合、これを行うことはできません。そのため、JSON 処理ユーティリティを使用して AWS CLI、新しい OpsWorks for Puppet Enterprise マスターがオンラインになった後、create-server コマンドの結果からサインイン認証情報とスターターキットを取得します。

ローカルコンピュータでまだ を実行していない場合は AWS CLI、AWS コマンドラインインターフェイスユーザーガイド AWS CLI [のインストール手順に従って](#) をダウンロードしてインストールします。このセクションでは、create-server コマンドで使用できるパラメータのすべては説明しません。create-server パラメータの詳細については、「[create-server リファレンス](#)」の「AWS CLI」を参照してください。

1. 「[前提条件](#)」を必ず完了してください。Puppet マスターを作成するには、サブネット ID が必要になるため、VPC が必要です。
2. サービスロールとインスタンスプロファイルを作成します。は、両方を作成するために使用できる AWS CloudFormation テンプレート AWS OpsWorks を提供します。次の AWS CLI コマンドを実行して、サービスロールとインスタンスプロファイルを作成する AWS CloudFormation スタックを作成します。

```
aws cloudformation create-stack --stack-name OpsWorksCMRoles --template-url
https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-
cm-roles.yaml --capabilities CAPABILITY_NAMED_IAM
```

3. がスタックの作成 AWS CloudFormation を完了したら、アカウント内のサービスロールARNs を検索してコピーします。

```
aws iam list-roles --path-prefix "/service-role/" --no-paginate
```

`list-roles` コマンドの結果内で、次のようなサービスロール ARN のエントリを探します。サービスロール ARN を書き留めます。Puppet Enterprise マスターを作成するにはこれらの値が必要です。

```
{
  "AssumeRolePolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "ec2.amazonaws.com"
        }
      }
    ]
  },
  "RoleId": "AR0ZZZZZZZZZZQ6R22HC",
  "CreateDate": "2018-01-05T20:42:20Z",
  "RoleName": "aws-opsworks-cm-ec2-role",
  "Path": "/service-role/",
  "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
},
{
  "AssumeRolePolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "opsworks-cm.amazonaws.com"
        }
      }
    ]
  },
  "RoleId": "AR0ZZZZZZZZZZZZZZ6QE",
  "CreateDate": "2018-01-05T20:42:20Z",
  "RoleName": "aws-opsworks-cm-service-role",
  "Path": "/service-role/",
  "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-role"
}
```

```
}
```

4. アカウントでインスタンスプロファイルの ARN を検索してコピーします。

```
aws iam list-instance-profiles --no-paginate
```

`list-instance-profiles` コマンドの結果内で、次のようなインスタンスプロファイル ARN のエントリを探します。インスタンスプロファイルの ARN を書き留めます。Puppet Enterprise マスターを作成するにはこれらの値が必要です。

```
{
  "Path": "/",
  "InstanceProfileName": "aws-opsworks-cm-ec2-role",
  "InstanceProfileId": "EXAMPLEDC6UR3LTUW7VHK",
  "Arn": "arn:aws:iam::123456789012:instance-profile/aws-opsworks-cm-ec2-role",
  "CreateDate": "2017-01-05T20:42:20Z",
  "Roles": [
    {
      "Path": "/service-role/",
      "RoleName": "aws-opsworks-cm-ec2-role",
      "RoleId": "EXAMPLEE4STNUQG6R22HC",
      "Arn": "arn:aws:iam::123456789012:role/service-role/aws-opsworks-cm-ec2-role",
      "CreateDate": "2017-01-05T20:42:20Z",
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      }
    }
  ]
},
```

5. `create-server` コマンドを実行して OpsWorks、for Puppet Enterprise マスターを作成します。

- `--engine` 値は Puppet で、`--engine-model` は Monolithic です。また、`--engine-version` は 2019 または 2017 になり得ます。
- サーバー名は、AWS アカウント内で各リージョン内で一意である必要があります。サーバー名は文字で始める必要があります。その後は文字、数字、またはハイフン (-) を最大 40 文字まで使用できます。
- ステップ 3 と 4 でコピーしたインスタンスプロファイル ARN とサービスロール ARN を使用します。
- 有効なインスタンスタイプは `m5.xlarge`、`c5.2xlarge`、または `c5.4xlarge` です。これらのインスタンスタイプの仕様についての詳細は、「Amazon EC2 User Guide」(Amazon EC2 ユーザーガイド) の「[Instance Types](#)」(インスタンスタイプ) を参照してください。
- `--engine-attributes` パラメータはオプションです。Puppet 管理者パスワードを指定しない場合、サーバー作成プロセスで生成されます。`--engine-attributes` を追加する場合は、`PUPPET_ADMIN_PASSWORD` を指定します。これは、Puppet Enterprise コンソールウェブページにサインインする管理者パスワードです。パスワードに使用できる文字数は 8~32 文字 (ASCII) です。
- SSH キーペアはオプションですが、コンソール管理者パスワードをリセットする必要がある場合に Puppet マスターに接続することができます。SSH キーペアの作成の詳細については、「Amazon EC2 User Guide」(Amazon EC2 ユーザーガイド) の「[Amazon EC2 Key Pairs](#)」(Amazon EC2 キーペア) を参照してください。
- カスタムドメインを使用するには、コマンドに以下のパラメータを追加します。それ以外の場合は、Puppet マスター作成プロセスによって自動的にエンドポイントが生成されます。カスタムドメインを構成するには、3 つのパラメータすべてが必要です。これらのパラメータを使用するためのその他の要件については、AWS OpsWorks CM API リファレンス [CreateServer](#) の「」を参照してください。
 - `--custom-domain` - サーバーのオプションのパブリックエンドポイント (`https://aws.my-company.com` など)。
 - `--custom-certificate` - PEM 形式の HTTPS 証明書。値には、単一の自己署名証明書、または証明書チェーンを指定できます。
 - `--custom-private-key` - HTTPS を使用してサーバーに接続するための PEM 形式のプライベートキー。プライベートキーは暗号化しないでください。パスワードやパスフレーズで保護することはできません。
- 週 1 回のシステムメンテナンスが必要です。次の形式で有効な値を指定する必要があります: `DDD:HH:MM`。指定時刻は協定世界時 (UTC) です。`--preferred-maintenance-window` の

値を指定しない場合、火曜日、水曜日、または金曜日の 1 時間がランダムでデフォルト値になります。

- `--preferred-backup-window` の有効な値は次の形式のいずれかで指定する必要があります: 日次バックアップの場合は HH:MM、週次バックアップの場合は DDD:HH:MM。指定時刻は UTC です。デフォルト値は日次で開始時間はランダムです。自動バックアップを無効にするには、代わりにパラメータ `--disable-automated-backup` を追加します。
- `--security-group-ids` に、1 つ以上のセキュリティグループ ID をスペースで区切って入力します。
- `--subnet-ids` には、サブネット ID を入力します。

```
aws opsworks-cm create-server --engine "Puppet" --engine-model "Monolithic"
--engine-version "2019" --server-name "server_name" --instance-profile-arn
"instance_profile_ARN" --instance-type "instance_type" --engine-attributes
'{"PUPPET_ADMIN_PASSWORD":"ASCII_password"}' --key-pair "key_pair_name" --
preferred-maintenance-window "ddd:hh:mm" --preferred-backup-window "ddd:hh:mm"
--security-group-ids security_group_id1 security_group_id2 --service-role-arn
"service_role_ARN" --subnet-ids subnet_ID
```

次に例を示します。

```
aws opsworks-cm create-server --engine "Puppet" --engine-model
"Monolithic" --engine-version "2019" --server-name "puppet-02" --
instance-profile-arn "arn:aws:iam::111122223333:instance-profile/aws-
opsworks-cm-ec2-role" --instance-type "m5.xlarge" --engine-attributes
'{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' --key-pair "amazon-test"
--preferred-maintenance-window "Mon:08:00" --preferred-backup-window
"Sun:02:00" --security-group-ids sg-b00000001 sg-b00000008 --service-role-arn
"arn:aws:iam::111122223333:role/service-role/aws-opsworks-cm-service-role" --
subnet-ids subnet-383daa71
```

次の例では、カスタムドメインを使用する Puppet マスターを作成します。

```
aws opsworks-cm create-server \
  --engine "Puppet" \
  --engine-model "Monolithic" \
  --engine-version "2019" \
  --server-name "puppet-02" \
  --instance-profile-arn "arn:aws:iam::111122223333:instance-profile/aws-
opsworks-cm-ec2-role" \
```

```
--instance-type "m5.xlarge" \  
--engine-attributes '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' \  
--custom-domain "my-puppet-master.my-corp.com" \  
--custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END  
CERTIFICATE-----" \  
--custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END  
RSA PRIVATE KEY-----" \  
--key-pair "amazon-test" \  
--preferred-maintenance-window "Mon:08:00" \  
--preferred-backup-window "Sun:02:00" \  
--security-group-ids sg-b00000001 sg-b00000008 \  
--service-role-arn "arn:aws:iam::111122223333:role/service-role/aws-opsworks-  
cm-service-role" \  
--subnet-ids subnet-383daa71
```

以下の例では、2つのタグ Stage: Production および Department: Marketing を追加する Puppet マスターを作成します。Puppet Enterprise サーバーのでのタグの追加と管理の詳細については、このガイド[AWS OpsWorks for Puppet Enterprise リソースでのタグの使用](#)の OpsWorks 「」を参照してください。

```
aws opsworks-cm create-server \  
--engine "Puppet" \  
--engine-model "Monolithic" \  
--engine-version "2019" \  
--server-name "puppet-02" \  
--instance-profile-arn "arn:aws:iam::111122223333:instance-profile/aws-  
opsworks-cm-ec2-role" \  
--instance-type "m5.xlarge" \  
--engine-attributes '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' \  
--key-pair "amazon-test" \  
--preferred-maintenance-window "Mon:08:00" \  
--preferred-backup-window "Sun:02:00" \  
--security-group-ids sg-b00000001 sg-b00000008 \  
--service-role-arn "arn:aws:iam::111122223333:role/service-role/aws-opsworks-  
cm-service-role" \  
--subnet-ids subnet-383daa71 \  
--tags [{"Key":"Stage","Value":"Production"}, {"Key":"Department",  
"Value":"Marketing"}]
```

6. OpsWorks for Puppet Enterprise は、新しいサーバーの作成に約 15 分かかります。create-server コマンドの出力を閉じたり、シェルセッションを閉じたりしないでください。今後表示

されない重要な情報が出力に含まれている場合があります。create-server の結果からパスワードとスターターキットを取得するには、次のステップに進みます。

サーバーでカスタムドメインを使用している場合は、create-server コマンドの出力で Endpoint 属性の値をコピーします。次に例を示します。

```
"Endpoint": "puppet-07-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
```

- OpsWorks for Puppet Enterprise でパスワードを生成することを選択した場合は、jq などの JSON プロセッサを使用して、create-server 結果から使用可能な形式で抽出できます。jq をインストールした後、以下のコマンドを実行して Puppet 管理者パスワードとスターターキットを抽出できます。ステップ 3 で独自のパスワードを指定しなかった場合は、抽出した管理者パスワードを使いやすく安全な場所に保存してください。

```
#Get the Puppet password:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
  "PUPPET_ADMIN_PASSWORD") | .Value'

#Get the Puppet Starter Kit:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
  "PUPPET_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
```

Note

AWS Management Console で新しい Puppet マスタースターターキットを再生成することはできません。を使用して Puppet マスターを作成する場合 AWS CLI、前述の jq コマンドを実行して base64 でエンコードされたスターターキットを ZIP ファイルとして create-server 結果に保存します。

- カスタムドメインを使用していない場合は、次のステップに進みます。サーバーでカスタムドメインを使用している場合は、エンタープライズの DNS 管理ツールで CNAME エントリを作成して、ステップ 6 でコピーした OpsWorks for Puppet Enterprise エンドポイントにカスタムドメインをポイントします。このステップを完了するまで、カスタムドメインを使用するサーバーにアクセスしたりサインインしたりすることはできません。
- 次のセクション「[the section called “設定を完了する”](#)」に進みます。

スターターキットを使用して Puppet マスターを設定する

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

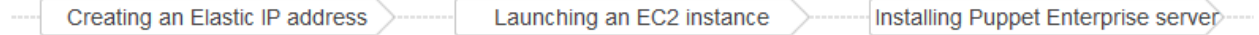
Puppet マスターの作成がまだ進行中の間、サーバーのプロパティページが for OpsWorks Puppet Enterprise コンソールで開きます。新しい Puppet マスターの初回操作時に、[Properties] ページで 2 つの必須項目をダウンロードするように求められます。Puppet サーバーがオンラインになる前に、これらの項目をダウンロードします。新しいサーバーがオンラインになった後は、ダウンロードボタンは使用できません。

test-puppet-server

[Puppet Enterprise dashboard](#) (not yet available)

Actions ▾

AWS OpsWorks is creating your Puppet Enterprise server. This takes about 20 minutes.



Make sure you download the following before your server is online.

- 1 Sign-in credentials for your Puppet Enterprise dashboard
- 2 Starter Kit for your Puppet Enterprise server

i Download the sign-in credentials for your [Puppet Enterprise dashboard](#).

▶ Show sign-in credentials

Download credentials

AWS OpsWorks does not save these credentials, so it is the last time they are available for viewing and downloading. After your server is online, you can change the password by signing in to its [Puppet Enterprise dashboard](#).

i Download the Starter Kit, and follow the [documentation](#) to finish the setup when your server is online.

Download Starter Kit

The Starter Kit contains a Readme with examples, and instructions how to install Puppet Enterprise client tools, as well as userdata templates for Windows and Linux.

Server information

[More settings](#)

Status	Version	Region	System maintenance	Automated backup
creating	2017.3.0	US West (Oregon)	5 pm - 6 pm UTC, every Tuesday	10 pm - 11 pm UTC, daily

Puppet Enterprise Console

<https://test-puppet-server-nxdx8g13l0wi6ug9.us-west-2.opsworks-cm.io>



- Puppet マスター用のサインイン認証情報。これらの認証情報を使用して、ほとんどのノード管理を実行する Puppet Enterprise コンソールにサインインします。これらの認証情報は保存 AWS OpsWorks されません。これは、表示とダウンロードが最後に利用可能になるときです。必要であれば、これらの認証情報で提供されたパスワードをサインイン後に変更できます。
- Starter Kit (スターターキット)。スターターキットには、セットアップを完了する方法を説明する情報と例、および Puppet Enterprise コンソールの管理認証情報が記載された README ファイルが含まれています。スターターキットをダウンロードするごとに、新しい認証情報が生成され、以前の認証情報は無効化されます。

前提条件

1. サーバー作成の進行中に、Puppet マスターのサインイン認証情報をダウンロードし、それを安全かつ便利な場所に保存します。
2. スターターキットをダウンロードし、スターターキットの .zip ファイルをワークスペースディレクトリに解凍します。サインイン認証情報を他者と共有しないでください。他のユーザーが Puppet マスターを管理する場合は、後で Puppet Enterprise コンソールでそのユーザーを管理者として追加します。Puppet マスターにユーザーを追加する詳細情報については、Puppet Enterprise ドキュメントの「[Creating and managing users and user roles](#)」を参照してください。

Puppet マスターの証明書をインストールする

Puppet マスターを使用し、ノードの管理を追加するには、その証明書をインストールする必要があります。次の AWS CLI コマンドを実行してインストールします。このタスクは、では実行できません AWS Management Console。

```
aws --region region opsworks-cm describe-servers --server-name server_name --query "Servers[0].EngineAttributes[?Name=='PUPPET_API_CA_CERT'].Value" --output text > .config/ssl/cert/ca.pem
```

短期トークンを生成する

Puppet API を使用するには、自分の短期トークンを作成する必要があります。このステップは、Puppet Enterprise コンソールを使用するためには必須ではありません。以下のコマンドを実行して、トークンを生成します。

デフォルトのトークンの有効期間は 5 分ですが、このデフォルトは変更できます。

```
puppet-access login --config-file .config/puppetlabs/client-tools/puppet-access.conf --lifetime 8h
```

Note

デフォルトのトークンの有効期間は 5 分であるため、前述のコマンド例では、`--lifetime` パラメータを追加し、トークンの有効期間を延長しています。トークンの有効期間は最長 10 年間 (10y) まで設定できます。デフォルトのトークンの有効期間を変更する方法の詳細については、Puppet Enterprise ドキュメントの「[Change the token's default lifetime](#)」を参照してください。

スターターキットの Apache の例をセットアップする

スターターキットをダウンロードして解凍した後、含まれているサンプル `control-repo-example` フォルダのブランチの例を使用して管理対象ノードで Apache ウェブサーバーを設定できます。

スターターキットには `control-repo` と `control-repo-example` という 2 つの `control-repo` フォルダがあります。`control-repo` フォルダには、[Puppet GitHub リポジトリ](#) に表示されるブランチから変更されていない `production` ブランチが含まれています。`control-repo-example` フォルダにもテストウェブサイトでの Apache サーバーをセットアップするサンプルコードが含まれている `production` ブランチがあります。

1. `control-repo-example production` ブランチを Git リモート (Puppet マスターの `r10k_remote` URL) にプッシュします。Starter Kit ルートディレクトリで、`r10kRemoteUrl` を `r10k_remote` URL に置き換えて、以下を実行します。

```
cd control-repo-example
git remote add origin r10kRemoteUrl
git push origin production
```

Puppet の Code Manager では、環境として Git ブランチが使用されます。デフォルトでは、すべてのノードが本稼働環境にあります。

⚠ Important

master ブランチにプッシュしないでください。master ブランチは Puppet マスター用に予約されています。

- control-repo-example ブランチのコードを Puppet マスターにデプロイします。これにより、Puppet マスターによって Git リポジトリから Puppet コード (r10k_remote) がダウンロードされます。スターターキットのルートディレクトリで、次のコマンドを実行します。

```
puppet-code deploy --all --wait --config-file .config/puppet-code.conf
```

Amazon EC2 に作成したマネージドノードにサンプルの Apache 設定を適用する方法の詳細については、本ガイドの「[ステップ 2: 自動関連付けスクリプトを使用してインスタンスを作成する](#)」を参照してください。

Puppet マスターで管理するノードを追加する

⚠ Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

トピック

- [associateNode\(\) API コールの実行](#)
- [オンプレミスでのノード追加の考慮事項](#)
- [詳細情報](#)

ノードを追加する推奨方法は、API `AWS OpsWorks associateNode()` を使用することです。Puppet Enterprise マスターサーバーは、管理するノードに Puppet エージェントソフトウェアをインストールするために使用するリポジトリをホストします。ノードがオンプレミスの物理コンピュータか仮想マシンかは問いません。一部のオペレーティングシステム用の Puppet エージェント

ソフトウェアは、起動プロセスの一環として OpsWorks、for Puppet Enterprise サーバーにインストールされます。次の表は、起動時に OpsWorks for Puppet Enterprise サーバーで使用できるオペレーティングシステムエージェントを示しています。

プレインストールされたオペレーティングシステムエージェント

サポートされるオペレーティングシステム	バージョン
Ubuntu	16.04, 18.04, 20.04
Red Hat Enterprise Linux (RHEL)	6, 7, 8
Windows	Puppet をサポートする すべての Windows リリースの 64 ビットエディション

他のオペレーティングシステムではサーバーに puppet-agent を追加できます。システムメンテナンスによって、起動後にサーバーに追加されたエージェントは削除されることに注意してください。削除されたエージェントをすでに実行している既存のアタッチ済みノードは引き続き稼働しますが、Debian オペレーティングシステムを実行するノードはレポートを停止する場合があります。エージェントソフトウェアが OpsWorks for Puppet Enterprise サーバー puppet-agent にプリインストールされていないオペレーティングシステムを実行しているノードに、を手動でインストールすることをお勧めします。他のオペレーティングシステムを使用するノードで、puppet-agent をサーバーで使用できるようにする方法の詳細については、Puppet Enterprise ドキュメントの「[Installing agents](#)」を参照してください。

EC2 インスタンスユーザーデータを設定してノードを Puppet マスターに自動的に関連付ける方法の詳細については、「[OpsWorks for Puppet Enterprise でノードを自動的に追加する](#)」を参照してください。

associateNode() API コールの実行

をインストールしてノードを追加すると puppet-agent、ノードは証明書署名リクエスト (CSRs) を for Puppet Enterprise サーバーに送信 OpsWorks します。Puppet コンソールで CSR を表示することができます。ノードの CSR の詳細については、Puppet Enterprise ドキュメントの「[Managing certificate signing requests](#)」を参照してください。OpsWorks for Puppet Enterprise associateNode() API コールを実行すると、ノード CSRs が処理され、ノードがサーバーに関連付けられます。以下は、でこの API コールを使用して 1 つのノード AWS CLI を関連付ける方法の例です。ノードが送信する PEM 形式の CSR が必要です。これは Puppet コンソールから入手できます。

```
aws opsworks-cm associate-node --server-name "test-puppet-
server" --node-name "node or instance ID" --engine-attributes
"Name=PUPPET_NODE_CSR,Value='PEM_formatted_CSR_from_the_node'"
```

associateNode() を使用してノードを自動的に追加する方法の詳細については、「[OpsWorks for Puppet Enterprise でノードを自動的に追加する](#)」を参照してください。

オンプレミスでのノード追加の考慮事項

オンプレミスのコンピュータまたは仮想マシン puppet-agent をインストールしたら、2つの方法のいずれかを使用してオンプレミスノードを OpsWorks for Puppet Enterprise マスターに関連付けることができます。

- ノードが [AWS SDK](#)、[AWS CLI](#)、または [AWS Tools for PowerShell](#) のインストールをサポートしている場合は、ノードの関連付けの推奨方法である associateNode() API コールの実行を使用できます。OpsWorks for Puppet Enterprise マスターを初めて作成するときにダウンロードするスターターキットは、タグを使用してノードにロールを割り当てる方法を示しています。CSR の信頼されたファクトを指定することで、ノードを Puppet マスターに関連付ける複数のタグを同時に適用できます。たとえば、スターターキットに含まれるデモ管理リポジトリは、タグ pp_role を使用して Amazon EC2 インスタンスにロールを割り当てるように設定されています。タグを信頼されたファクトとして CSR に追加する方法の詳細については、Puppet プラットフォームドキュメントの「[Extension requests \(permanent certificate data\)](#)」を参照してください。
- ノードが AWS 管理ツールまたは開発ツールを実行できない場合は、管理されていない Puppet Enterprise マスターに登録するのと同じ方法で OpsWorks for Puppet Enterprise マスターに登録できます。このトピックで説明したように、 をインストールすると、CSR が OpsWorks for Puppet Enterprise マスター puppet-agent に送信されます。承認された Puppet ユーザーが手動で CSR に署名するか、Puppet マスターに保存されている autosign.conf ファイルを編集して CSR の自動署名を設定できます。自動署名の設定と autosign.conf の編集の詳細については、Puppet プラットフォームのドキュメントの「[SSL configuration: autosigning certificate requests](#)」を参照してください。

オンプレミスノードを Puppet マスターに関連付けて Puppet マスターがすべての CSR を受け入れるようにするには、Puppet Enterprise コンソールで次を実行します。この動作を制御するパラメータは puppet_enterprise::profile::master::allow_unauthenticated_ca です。

⚠ Important

自己署名の CSR を受け入れる Puppet マスターを有効にしない限り、すべての CSR はセキュリティ上の理由から推奨されません。デフォルトでは、認証されていない CSR を許可すると、Puppet マスターが誰からでもアクセスできることになります。証明書リクエストのアップロード設定をデフォルトで有効にすることは、Puppet マスターをサービス妨害 (DoS) 攻撃に対して脆弱にします。

1. Puppet Enterprise コンソールへのサインイン
2. [Configure] (設定)、[Classification] (分類)、[PE Master] (PE マスター)の順に選択して、[Configuration] (設定) タブを選択します。
3. [分類] タブで [puppet_enterprise::profile::master] クラスを見つけます。
4. [allow_unauthenticated_ca] パラメータの値を [true] に設定します。
5. 変更を保存します。この変更は次の Puppet 実行時に適用されます。変更が効果を発揮するまで (そして、オンプレミスノードが追加されるまで) に 30 分待つか、あるいは PE コンソールの [Run] (実行) セクションから手動で Puppet を開始します。

詳細情報

[Puppet Enterprise サーバーと Puppet Enterprise コンソール機能での の使用の詳細については、「Puppet の学習」チュートリアルサイトを参照してください。](#) OpsWorks

Puppet Enterprise コンソールへのサインイン

⚠ Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Puppet マスターの [Properties] ページでサインイン認証情報をダウンロードし、サーバーがオンラインになったら、Puppet Enterprise コンソールにサインインします。このウォークスルーでは、モ

ジュールを含むコントロールリポジトリを指定し、管理するノードを1つ以上追加します。そうすることによって、エージェントおよびノードに関する情報をコンソールで表示できるようになります。

Puppet Enterprise コンソールのウェブページに接続しようとする、Puppet サーバーの管理に使用しているクライアントコンピュータに AWS OpsWorks 固有の CA 署名付き SSL 証明書をインストールするまで、証明書の警告がブラウザに表示されます。ダッシュボードのウェブページに進む前に警告が表示されないようにするには、サインインする前に SSL 証明書をインストールします。

AWS OpsWorks SSL 証明書をインストールするには

- システムに適合する証明書を選択します。
- Linux または MacOS ベースのシステムの場合は、次の Amazon S3 の場所から PEM ファイル名拡張子を持つファイルをダウンロードします: <https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-ca-2016-root.pem>。

Note

さらに、<https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-ca-2020-root.pem> から新しい PEM ファイルをダウンロードします。OpsWorks for Puppet Enterprise は現在ルート証明書を更新しているため、古い証明書と新しい証明書の両方を信頼する必要があります。

MacOS で SSL 証明書を管理する方法の詳細については、アップルの Support ウェブサイトの「[Macのキーチェーンアクセスで証明書についての情報を取得する](#)」を参照してください。

- Windows ベースのシステムの場合は、PP7Bファイル名拡張子のファイルを次の Amazon S3 の場所からダウンロードします: <https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-ca-2016-root.p7b>。

Windows に SSL 証明書をインストールする方法の詳細については、「Microsoft で[信頼できるルート証明書を管理する](#)」を参照してください TechNet。

Note

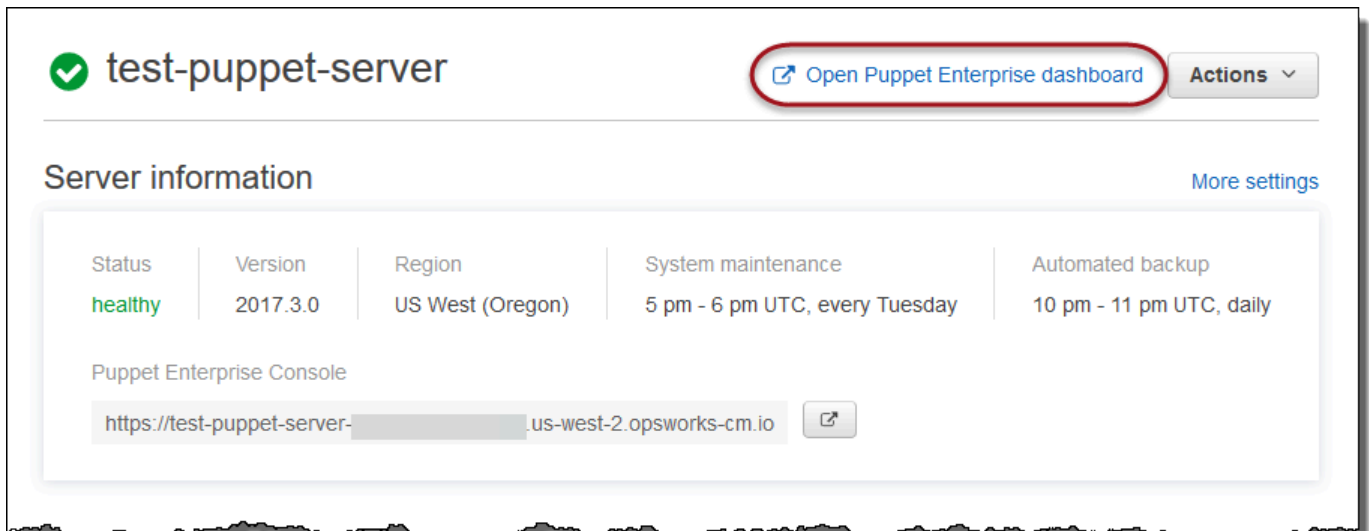
さらに、<https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-ca-2020-root.p7b> から新しい P7B ファイルをダウンロードします。

OpsWorks for Puppet Enterprise は現在ルート証明書を更新しているため、古い証明書と新しい証明書の両方を信頼する必要があります。

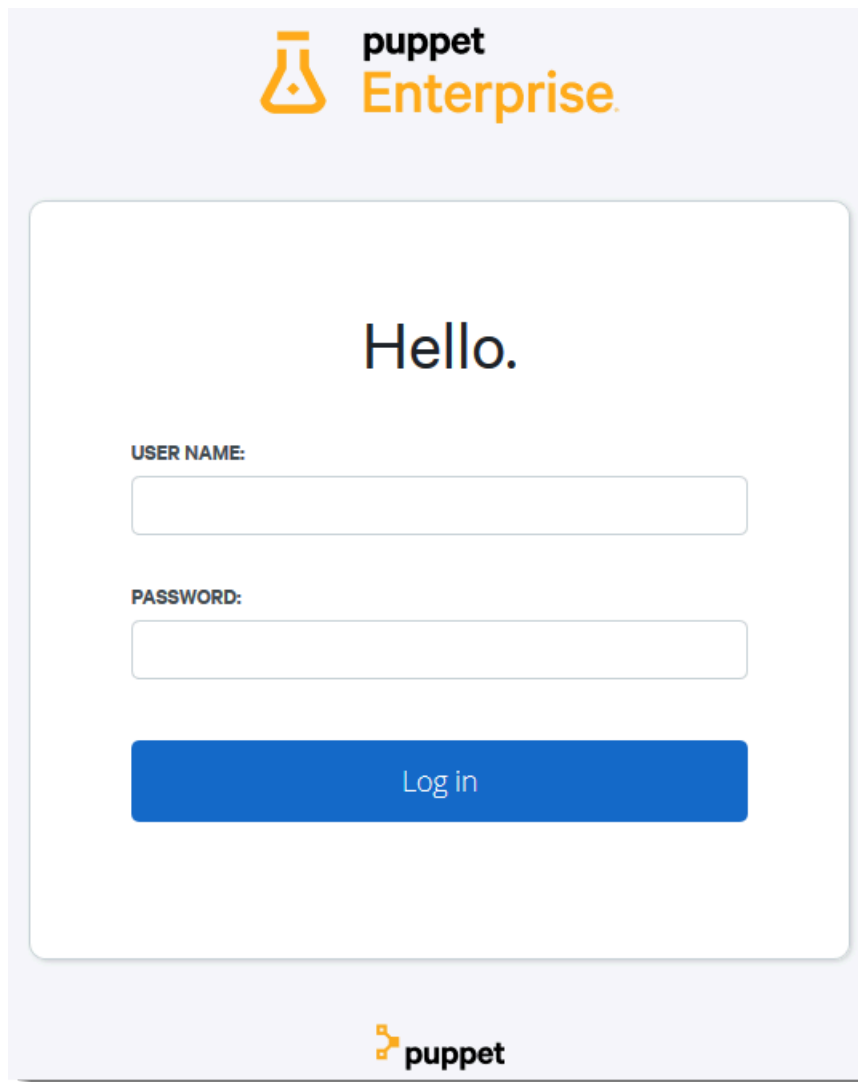
クライアント側の SSL 証明書をインストールした後は、警告メッセージを表示せずに Puppet Enterprise コンソールへのサインインすることができます。

Puppet Enterprise コンソールにサインインするには

1. 「[前提条件](#)」でダウンロードした Puppet Enterprise 認証情報を解凍して開きます。サインインするには、これらの認証情報が必要です。
2. で AWS Management Console、Puppet サーバーのプロパティページを開きます。
3. [プロパティ] ページの右上にある [Open Puppet Enterprise console] (Puppet Enterprise コンソールを開く) を選択します。



4. ステップ 1 で認証情報を使用してサインインします。



puppet
Enterprise.

Hello.

USER NAME:

PASSWORD:

Log in

puppet

5. Puppet Enterprise コンソールでは、管理しているノード、モジュールの実行の進行状況とイベント、ノードのコンプライアンスレベルなどに関する詳細情報を表示できます。Puppet Enterprise コンソールの機能および使用方法の詳細については、Puppet Enterprise ドキュメントの「[Managing nodes](#)」(管理ノード)を参照してください。

The screenshot shows the Puppet Enterprise Status page. The left sidebar contains navigation options like ENFORCEMENT, ORCHESTRATION, INVENTORY, PATCH MANAGEMENT, and ADMIN. The main content area shows the 'Status' section with a 'Run puppet' button and a 'Filter by fact value' dropdown. Three summary cards are visible: '1 Nodes run in enforcement', '0 Nodes run in no-op', and '0 Nodes not reporting'. Each card lists various states with counts and links. Below these is a table with columns for Run status, No-op mode, Job ID, Last report, and Node name. The table shows one entry for 'us-west-1.opsworks-cm.io' with a green status icon and a last report time of '2021-04-28 21:25 Z'.

ノードのグループ化と分類

クラスをノードに適用して希望する設定を指定する前に、ノードを企業内のロールまたは一般的な特長に応じてグループ化します。ノードのグループ化および分類には以下の高レベルのタスクが含まれます。PE コンソールを使用してこれらのタスクを完了することができます。ノードのグループかおよび分類方法の詳細については、Puppet Enterprise ドキュメントの「[Grouping and classifying nodes](#)」を参照してください。

1. ノードグループを作成します。
2. 手動、または作成したルールを適用して自動で、ノードをグループに追加します。
3. クラスをノードグループに割り当てます。

管理者パスワードとユーザーパスワードをリセットする

Puppet Enterprise コンソールにサインインするために使用するパスワードの変更方法については、Puppet Enterprise ドキュメントの「[Reset the console administrator password](#)」(コンソールの管理パスワードをリセット)を参照してください。

デフォルトでは、10 回サインインが試行されると、ユーザーは Puppet コンソールからロックアウトされます。ロックアウトが発生した場合にユーザーパスワードをリセットする方法の詳細については、Puppet Enterprise ドキュメントの「[Password endpoints](#)」を参照してください。

オプション: Puppet r10k リモートコントロールリポジトリ AWS CodeCommit として を使用する

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

を使用して新しいリポジトリを作成し AWS CodeCommit、r10k リモートコントロールリポジトリとして使用できます。このセクションのステップを完了し、CodeCommit リポジトリを使用するには、AWSCodeCommitReadOnly管理ポリシーによって提供されるアクセス許可を持つユーザーが必要です。

トピック

- [ステップ 1: を HTTPS 接続タイプのリポジトリ CodeCommit として使用する](#)
- [ステップ 2: \(オプション\) SSH 接続タイプのリポジトリ CodeCommit として を使用する](#)

ステップ 1: を HTTPS 接続タイプのリポジトリ CodeCommit として使用する

1. CodeCommit コンソールで、新しいリポジトリを作成します。

Create repository ?

Create a secure repository to store and share your code. Begin by typing a repository name and a description for your repository. Repository names are included in the URLs for that repository.

i Access to the repository

Users connecting to an AWS CodeCommit repository for the first time must complete setup steps before they can use it. [Learn more](#)

Repository name*

Description

*Required

Cancel

Create repository

2. [Skip] (スキップ) を選択して Amazon SNS トピックの設定をスキップします。
3. [Code] (コード) ページで、[Connect to your repository] を選択します。
4. [Connect to your repository] ページで、[HTTPS] を [Connection type] として選択し、オペレーティングシステムを選択します。

Connect to your repository

You are signed in using [federated access](#) or temporary credentials. The only supported connection method for these sign-in types is to use the credential manager included with the AWS CLI, as documented below. To configure a connection using SSH or Git credentials over HTTPS, sign in as an [IAM user](#).

Follow the steps below to connect to your repository from your local computer.

Connection type HTTPS
 SSH

Operating system Linux, MacOS, or Unix
 Windows

Prerequisites

1. Install Git (1.7.9 or later supported). If you don't have Git installed, [install it now](#).
2. Install the [AWS CLI](#).
3. At the terminal, type `aws configure` and [configure the AWS CLI](#) with your IAM user access key and secret key.
4. Attach an appropriate [AWS CodeCommit managed policy](#) to the IAM user. [Learn more](#)

Steps to clone your repository

1. At the terminal, paste the following commands:


```
git config --global credential.helper '!aws codecommit credential-helper $@'
git config --global credential.UseHttpPath true
```
2. Clone your repository to your local computer and start working on code:


```
git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/control-repo
```
3. If using MacOS, [Disable the Keychain Access utility](#) for connections to AWS CodeCommit.

[I want more detailed instructions](#)

[Steps to clone your repository] 領域で、`git clone` URL は次のようになります: `https://git-codecommit.region.amazonaws.com/v1/repos/control-repo` この URL を、Puppet サーバーのセットアップに使用しやすい場所にコピーします。

5. リポジトリへの接続ページを閉じ、OpsWorks for Puppet Enterprise サーバーのセットアップに戻ります。
6. ステップ 4 でコピーした URL を、Puppet マスターセットアップウィザードの [Configure credentials] (認証情報を設定) ページにある [r10k remote] 文字列ボックスに貼り付けます。[r10k private key] ボックスは空のままにしておきます。Puppet マスターの作成と起動を完了します。
7. IAM コンソールで、`AWSCodeCommitReadOnly` ポリシーを Puppet マスターのインスタンスプロファイルロールにアタッチします。ポリシーを IAM ロールにアタッチする方法の詳細につ

いては、IAM ユーザーガイドの「[IAM ID アクセス許可の追加 \(コンソール\)](#)」を参照してください。

- 「AWS CodeCommit ユーザーガイド」の「[Git 認証情報を使用した HTTPS ユーザー用セットアップ](#)」の手順に従って、既存のcontrol-repoコンテンツを新しい CodeCommit リポジトリにプッシュします。
- これで、引き続き「[the section called “設定を完了する”](#)」の手順に従い、スターターキットを使用してコードを Puppet マスターにデプロイできます。コマンドの例を次に示します。

```
puppet-code deploy --all --wait --config-file .config/puppet-code.conf
```

ステップ 2: (オプション) SSH 接続タイプのリポジトリ CodeCommit として を使用する

SSH キーペア認証を使用するように AWS CodeCommit r10k リモートコントロールリポジトリを設定できます。この手順を開始する前に、次の前提条件を満たす必要があります。

- 前のセクション OpsWorks 「」で説明したように、HTTPS コントロールリポジトリを使用して for Puppet Enterprise サーバーを起動している必要があります[the section called “ステップ 1: を HTTPS 接続タイプのリポジトリ CodeCommit として使用する”](#)。必要な設定を Puppet マスターにアップロードできるように、この手順は最初に行う必要があります。
- AWSCodeCommitReadOnly 管理ポリシーがアタッチされたユーザーがあることを確認してください。ユーザーの作成方法の詳細については、「IAM ユーザーガイド」の AWS 「[アカウントでの IAM ユーザーの作成](#)」を参照してください。
- SSH キーを作成して、ユーザーを関連付けます。AWS CodeCommit 「ユーザーガイド」の「[ステップ 3: Linux、macOS、または Unix で認証情報を設定する](#)」の「ssh-keygen」に記載されている公開鍵/秘密鍵ペアの作成方法に従ってください。

- AWS CLI セッションで次のコマンドを実行して、プライベートキーファイルの内容を AWS Systems Manager Parameter Store にアップロードします。OpsWorks for Puppet Enterprise サーバーは、このパラメータをクエリして、必要な証明書ファイルを取得します。*private_key_file* を SSH プライベートキーファイルへのパスに置き換えます。

```
aws ssm put-parameter --name puppet_user_pk --type String --value  
"cat private_key_file"
```

- Systems Manager パラメータストアのアクセス権を Puppet マスターに追加します。

- a. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
 - b. 左のナビゲーションペインで、[ロール] を選択します。
 - c. aws-opsworks-cm-ec2 ロールを選択します。
 - d. [Permissions (アクセス許可)] タブで、[Attach policy (ポリシーの添付)] を選択します。
 - e. [Search] バーに「**AmazonSSMManagedInstanceCore**」と入力します。
 - f. 検索結果で、AmazonSSMManagedInstanceCore を選択します。
 - g. Attach policy] (ポリシーのアタッチ) を選択します。
3. 設定ファイルマニフェストを作成します。スターターキットの control-repo-example を使用する場合は、サンプルリポジトリに示されている場所に次のファイルを作成します。それ以外の場合は、独自のコントローラーリポジトリ構造に従って作成します。**IAM_USER_SSH_KEY** を、この手順の前提条件で作成した SSH キー ID と置き換えます。

```
control-repo-example/site/profile/manifests/codecommit.pp
```

```
class profile::codecommit {
  $configfile = @(CONFIGFILE)
  Host git-codecommit.*.amazonaws.com
  User IAM_USER_SSH_KEY
  IdentityFile /etc/puppetlabs/puppetserver/ssh/codecommit.rsa
  StrictHostKeyChecking=no
  | CONFIGFILE

  # Replace REGION with the correct region for your server.
  $command = @(COMMAND)
  aws ssm get-parameters \
    --region REGION \
    --names puppet_user_pk \
    --query "Parameters[0].Value" \
    --output text >| /etc/puppetlabs/puppetserver/ssh/codecommit.rsa
  | COMMAND

  $dirs = [
    '/opt/puppetlabs/server/data/puppetserver/.ssh',
    '/etc/puppetlabs/puppetserver/ssh',
  ]

  file { $dirs:
    ensure => 'directory',
  }
}
```

```
group => 'pe-puppet',
owner  => 'pe-puppet',
mode   => '0750',
}

file { 'ssh-config':
  path      => '/opt/puppetlabs/server/data/puppetserver/.ssh/config',
  require => File[$dirs],
  content => $configfile,
  group    => 'pe-puppet',
  owner    => 'pe-puppet',
  mode     => '0600',
}

exec { 'download-codecommit-certificate':
  command => $command,
  require => File[$dirs],
  creates => '/etc/puppetlabs/puppetserver/ssh/codecommit.rsa',
  path    => '/bin',
  cwd     => '/etc/puppetlabs',
}

file { 'private-key-permissions':
  subscribe => Exec['download-codecommit-certificate'],
  path      => '/etc/puppetlabs/puppetserver/ssh/codecommit.rsa',
  group     => 'pe-puppet',
  owner     => 'pe-puppet',
  mode     => '0600',
}
}
```

4. コントロールリポジトリを にプッシュします CodeCommit。以下のコマンドを実行して、新しいマニフェストファイルをリポジトリにプッシュします。

```
git add ./site/profile/manifests/codecommit.pp
git commit -m 'Configuring for SSH connection to CodeCommit'
git push origin production
```

5. マニフェストファイルを展開します。次のコマンドを実行して、更新された設定を OpsWorks for Puppet Enterprise サーバーにデプロイします。 *STARTER_KIT_DIRECTORY* を Puppet 設定ファイルへのパスに置き換えます。

```
cd STARTER_KIT_DIRECTORY
```

```
puppet-access login --config-file .config/puppetlabs/client-tools/puppet-  
access.conf
```

```
puppet-code deploy --all --wait \  
--config-file .config/puppet-code.conf \  
--token-file .config/puppetlabs/token
```

6. Puppet Enterprise サーバーの分類 OpsWorks の を更新します。デフォルトでは、Puppet エージェントはノード (マスターを含む) 上で 30 分ごとに実行されます。すぐに実行するには、Puppet マスター上で手動でエージェントを実行します。エージェントを実行すると、新しいマニフェストファイルが取得されます。

- a. Puppet Enterprise コンソールへのサインイン
- b. [分類] を選択します。
- c. [PE インフラストラクチャ] を展開します。
- d. [PE マスター] を選択します。
- e. 設定 タブで、[Add new class] (新しいクラスを追加する) に **profile::codecommit** と入力します。

puppet-code deploy を実行してから、新しいクラス profile::codecommit が表示されるまでに少し時間がかかる場合があります。表示されない場合は、このページの [Refresh] (更新) を選択します。

- f. [Add class (クラスの追加)], [Commit 1 change] の順に選択します。
 - g. OpsWorks for Puppet Enterprise サーバーで Puppet エージェントを手動で実行します。[ノード] を選択後、リスト内の該当するサーバーを選択し、[Run Puppet]、[実行] の順に選択します。
7. Puppet Enterprise コンソールで、HTTPS ではなく SSH を使用するようにリポジトリ URL を変更します。これらのステップで実行する設定は、Puppet Enterprise OpsWorks のバックアップおよび復元プロセス中に保存されるため、メンテナンスアクティビティ後にリポジトリ設定を手動で変更する必要はありません。
- a. [分類] を選択します。
 - b. [PE インフラストラクチャ] を展開します。
 - c. [PE マスター] を選択します。
 - d. [設定] タブで、puppet_enterprise::profile::master クラスを見つけます。
 - e. r10k_remote パラメータの横にある [Edit] (編集) を選択します。

- f. リポジトリの HTTPS URL を SSH URL に置き換え、[Commit 1 change] を選択します。
- g. OpsWorks for Puppet Enterprise サーバーで Puppet エージェントを手動で実行します。
[ノード] を選択後、リスト内の該当するサーバーを選択し、[Run Puppet]、[実行] の順に選択します。

を使用して AWS OpsWorks for Puppet Enterprise マスターを作成する AWS CloudFormation

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks for Puppet Enterprise では、[Puppet Enterprise](#) サーバーを実行できます AWS。約 15 分で Puppet Enterprise マスターサーバーをプロビジョニングできます。

2021 年 5 月 3 日以降、OpsWorks for Puppet Enterprise は一部の Puppet Enterprise サーバー属性を保存します AWS Secrets Manager。詳細については、「[との統合 AWS Secrets Manager](#)」を参照してください。

以下のチュートリアルでは、スタックを作成することで、for Puppet Enterprise で OpsWorks Puppet マスターを作成する方法について説明します AWS CloudFormation。

トピック

- [前提条件](#)
- [AWS CloudFormationで Puppet Enterprise マスターを作成する](#)

前提条件

新しい Puppet マスターを作成する前に、Puppet マスターにアクセスして管理する必要がある OpsWorks for Puppet Enterprise の外部にリソースを作成します。詳細については、このガイドの「はじめに」セクションにある「[前提条件](#)」を参照してください。

カスタムドメインを使用するサーバーを作成する場合は、カスタムドメイン、証明書、およびプライベートキーが必要です。AWS CloudFormation テンプレートでは、これら 3 つのパラメータすべてに値を指定する必要があります。CustomDomain、および CustomPrivateKey パラメータの要件の詳細については CustomCertificate、AWS OpsWorks CM API リファレンスの [CreateServer](#) 「」を参照してください。

サーバーの作成に使用するテンプレートでサポートされる値と必要な値については、AWS CloudFormation 「ユーザーガイドテンプレートリファレンス」の [OpsWorks 「-CM」 セクション](#) を参照してください。AWS CloudFormation

AWS CloudFormation で Puppet Enterprise マスターを作成する

このセクションでは、AWS CloudFormation テンプレートを使用して、OpsWorks for Puppet Enterprise マスターサーバーを作成するスタックを構築する方法について説明します。これを行うには、AWS CloudFormation コンソールまたはを使用します AWS CLI。テンプレート [の例 AWS CloudFormation](#) を使用して、OpsWorks for Puppet Enterprise サーバースタックを構築できます。サンプルテンプレートを更新するには、必ず、独自のサーバー名、IAM ロール、インスタンスプロファイル、サーバーの説明、バックアップ保持数、メンテナンスオプション、およびオプションのタグを使用してください。サーバーでカスタムドメインを使用する場合は、CustomDomain テンプレートで、CustomCertificate、CustomPrivateKey および AWS CloudFormation パラメータの値を指定する必要があります。これらのオプションの詳細については、このガイドの「はじめに」セクションの「[the section called “を使用して Puppet Enterprise Master を作成する AWS Management Console”](#)」を参照してください。

トピック

- [を使用して Puppet Enterprise Master を作成する AWS CloudFormation \(コンソール \)](#)
- [AWS CloudFormation \(CLI\) を使用して Puppet Enterprise Master を作成する](#)

を使用して Puppet Enterprise Master を作成する AWS CloudFormation (コンソール)

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/cloudformation> で AWS CloudFormation コンソールを開きます。
2. AWS CloudFormation ホームページで、スタックの作成 を選択します。
3. 前提条件 - テンプレートの準備では、[example AWS CloudFormation template](#) (テンプレートのサンプル) を使用している場合、テンプレートの準備完了 を選択します。

4. [Specify template] 内で、テンプレートのソースを選択します。このチュートリアルでは、テンプレートファイルをアップロードを選択し、Puppet Enterprise サーバーを作成する AWS CloudFormation テンプレートをアップロードします。テンプレートファイルを参照し、[Next] を選択します。

AWS CloudFormation テンプレートは YAML 形式または JSON 形式のいずれかです。[サンプル AWS CloudFormation テンプレート](#)を使用できます。サンプル値は必ず独自の値に置き換えてください。AWS CloudFormation テンプレートデザイナーを使用して、新しいテンプレートを構築したり、既存のテンプレートを検証したりできます。これを行う方法については、AWS CloudFormation ユーザーガイドの[AWS CloudFormation 「Designer Interface Overview」](#) (Designer インターフェイスの概要) を参照してください。

Create stack

Prerequisite - Prepare template

Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

Template is ready Use a sample template Create template in Designer

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source
Selecting a template generates an Amazon S3 URL where it will be stored.

Amazon S3 URL Upload a template file

Upload a template file
 `opsworkscm-server.json`
JSON or YAML formatted file

S3 URL: `https://s3-external-1.amazonaws.com/cf-templates-` `-opsworkscm-server.json`

5. [Specify stack details] ページでスタックの名前を入力します。これは、サーバーの名前と同じ名前にしないでください。スタック名にすぎません。[Parameters (パラメータ)] 領域で、Puppet Enterprise コンソールウェブページにサインインする管理者パスワードを入力します。パスワードに使用できる文字数は 8~32 文字 (ASCII) です。[次へ] をクリックします。

Specify stack details

Stack name

Stack name

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

AdminPassword

Cancel Previous **Next**

- [Options] (オプション) ページで、スタックを使用して作成するサーバーにタグを追加し、テンプレートで使用する IAM ロールをまだ指定していない場合は、リソースを作成するための IAM ロールを選択できます。オプションの指定を終了したら、[Next (次へ)] を選択します。ロールバックトリガーなどの高度なオプションの詳細については、「AWS CloudFormation ユーザーガイド」の [AWS CloudFormation 「スタックオプションの設定」](#) を参照してください。
- [確認] ページで選択内容を確認します。サーバースタックを作成する準備ができたなら、[Create (作成)] を選択します。

がスタックを作成する AWS CloudFormation のを待っている間に、スタックの作成ステータスを表示します。スタックの作成に失敗した場合、コンソールに表示されるエラーメッセージを確認し、問題を解決します。AWS CloudFormation スタックのエラーのトラブルシューティングについての詳細は、AWS CloudFormation ユーザーガイドの [「Troubleshooting Errors」](#) (エラーのトラブルシューティング) を参照してください。

サーバーの作成が完了すると、Puppet マスターは OpsWorks for Puppet Enterprise のホームページでオンライン のステータスになります。サーバーがオンラインになると、Puppet Enterprise コンソールがサーバーのドメインで `https://your_server_name-randomID.region.opsworks-cm.io` の形式の URL で利用できます。

Note

サーバーのカスタムドメイン、証明書、プライベートキーを指定した場合は、エンタープライズの DNS 管理ツールで CNAME エントリを作成します。このエントリは、カスタムドメインを Puppet Enterprise OpsWorks がサーバーに自動的に生成したエンドポイントにマッピングします。生成されたエンドポイントをカスタムドメイン値にマッピングするまで、サーバーを管理したり、サーバーの Puppet Enterprise 管理 Web サイトに接続したりすることはできません。

生成されたエンドポイント値を取得するには、サーバーがオンラインになった後に次の AWS CLI コマンドを実行します。

```
aws opsworks describe-servers --server-name server_name
```

AWS CloudFormation (CLI) を使用して Puppet Enterprise Master を作成する

ローカルコンピュータでまだ を実行していない場合は AWS CLI、AWS コマンドラインインターフェイスユーザーガイド [AWS CLI のインストール手順に従って](#) をダウンロードしてインストールします。このセクションでは、create-stack コマンドで使用できるパラメータのすべては説明しません。create-stack パラメータの詳細については、「[create-stack リファレンス](#)」の「AWS CLI」を参照してください。

1. OpsWorks for Puppet Enterprise マスターを作成 [前提条件](#) するための を必ず完了してください。
2. サービスロールとインスタンスプロファイルを作成します。 は、両方を作成するために使用できる AWS CloudFormation テンプレート AWS OpsWorks を提供します。次の AWS CLI コマンドを実行して、サービスロールとインスタンスプロファイルを作成する AWS CloudFormation スタックを作成します。

```
aws cloudformation create-stack --stack-name OpsWorksCMRoles --template-url  
https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-  
cm-roles.yaml --capabilities CAPABILITY_NAMED_IAM
```

がスタックの作成 AWS CloudFormation を完了したら、アカウント内のサービスロールARNs を検索してコピーします。

```
aws iam list-roles --path-prefix "/service-role/" --no-paginate
```

`list-roles` コマンドの結果内で、次のようなサービスロールとインスタンスプロファイルのエントリを探します。サービスロールとインスタンスプロファイルの ARNs を書き留めて、Puppet マスターサーバスタックの作成に使用している AWS CloudFormation テンプレートに追加します。

```
{
  "AssumeRolePolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "ec2.amazonaws.com"
        }
      }
    ]
  },
  "RoleId": "AROZZZZZZZZZZQ6R22HC",
  "CreateDate": "2018-01-05T20:42:20Z",
  "RoleName": "aws-opsworks-cm-ec2-role",
  "Path": "/service-role/",
  "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
},
{
  "AssumeRolePolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "opsworks-cm.amazonaws.com"
        }
      }
    ]
  },
  "RoleId": "AROZZZZZZZZZZZZZZ6QE",
  "CreateDate": "2018-01-05T20:42:20Z",
  "RoleName": "aws-opsworks-cm-service-role",
  "Path": "/service-role/",
```

```
"Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-  
role"  
}
```

3. create-stack コマンドを再度実行して OpsWorks、for Puppet Enterprise マスターを作成します。
 - `stack_name` をスタックの名前に置き換えます。これは、Puppet マスターではなく、AWS CloudFormation スタックの名前です。Puppet マスター名は、ServerName テンプレートにおける AWS CloudFormation の値です。
 - `template` をテンプレートファイルへのパスに置き換え、`yaml ### json` を必要に応じて `.yaml` または `.json` に置き換えます。
 - の値は、[CreateServer API EngineAttributes](#) から `--parameters` に対応します。Puppet の場合、サーバーを作成するためにユーザーが指定するエンジン属性は以下のとおりです。r10k エンジン属性は、Puppet マスターをコードリポジトリに接続してサーバーの環境設定を管理します。r10k エンジン属性の詳細については、Puppet Enterprise ドキュメントの「[r10k でのコードの管理](#)」を参照してください。
 - PUPPET_ADMIN_PASSWORD: Puppet Enterprise コンソールウェブページにサインインするための管理者パスワードです。パスワードには、8~32 文字の ASCII 文字を使用し、少なくとも 1 つの大文字、1 つの小文字、1 つの数字、1 つの特殊文字を含める必要があります。
 - PUPPET_R10K_REMOTE: コントロールリポジトリの URL です (例: `ssh://git@your.git-repo.com:user/control-repo.git`)。r10k リモートを指定すると、TCP ポート 8170 が開きます。
 - PUPPET_R10K_PRIVATE_KEY。プライベート Git リポジトリを使用している場合、PUPPET_R10K_PRIVATE_KEY を追加して SSH URL と PEM でエンコードされたプライベート SSH キーを指定します。

```
aws cloudformation create-stack --stack-name stack_name  
--template-body file://template.yaml or json --parameters  
ParameterKey=AdminPassword,ParameterValue="password"
```

次に例を示します。

```
aws cloudformation create-stack --stack-name "OpsWorksCMPuppetServerStack"
--template-body file://opsworkscm-puppet-server.json --parameters
ParameterKey=AdminPassword,ParameterValue="09876543210Ab#"
```

次の例では、r10k エンジン属性が AWS CloudFormation テンプレートで指定されていない場合、パラメータとして指定します。r10k エンジン属性 puppet-server-param-attributes.yaml が含まれるサンプルテンプレートは、[サンプル AWS CloudFormation テンプレート](#)に含まれています。

```
aws cloudformation create-stack --stack-name MyPuppetStack --
template-body file://puppet-server-param-attributes.yaml --parameters
ParameterKey=AdminPassword,ParameterValue="superSecret1%3"
ParameterKey=R10KRemote,ParameterValue="https://www.yourRemote.com"
ParameterKey=R10KKey,ParameterValue="$(cat puppet-r10k.pem)"
```

次の例では、AWS CloudFormation テンプレートで r10k エンジン属性とその値を指定します。コマンドは、テンプレートファイルのみポイントする必要があります。--template-body の値として指定されるテンプレート puppet-server-in-file-attributes.yaml は、[サンプル AWS CloudFormation テンプレート](#)に含まれています。

```
aws cloudformation create-stack --stack-name MyPuppetStack --template-body file://
puppet-server-in-file-attributes.yaml
```

4. (オプション) スタックの作成状況を取得するには、次のコマンドを実行します。

```
aws cloudformation describe-stacks --stack-name stack_name
```

5. スタックの作成が終了したら、次のセクション「[the section called “設定を完了する”](#)」に移動します。スタックの作成に失敗した場合、コンソールに表示されるエラーメッセージを確認し、問題を解決します。AWS CloudFormation スタックのエラーのトラブルシューティングの詳細については、「AWS CloudFormation ユーザーガイド」の「[エラーのトラブルシューティング](#)」を参照してください。

カスタムドメインを使用するように OpsWorks for Puppet Enterprise Server を更新する

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、サーバーのバックアップを使用して新しいサーバーを作成することで、既存の OpsWorks for Puppet Enterprise サーバーを更新してカスタムドメインと証明書を使用する方法について説明します。基本的に、バックアップから新しいサーバーを作成し、カスタムドメイン、証明書、プライベートキーを使用するように新しいサーバーを設定することで、Puppet Enterprise 2.0 OpsWorks サーバー用の既存のをコピーします。

トピック

- [前提条件](#)
- [制限事項](#)
- [カスタムドメインを使用するようにサーバーを更新する](#)
- [以下の資料も参照してください。](#)

前提条件

以下は、カスタムドメインと証明書を使用するように既存の OpsWorks for Puppet Enterprise サーバーを更新するための要件です。

- 更新 (またはコピー) するサーバーは、Puppet Enterprise 2019.8.5 を実行している必要があります。
- 新しいサーバーの作成に使用するバックアップを決定します。更新するサーバーのバックアップが少なくとも 1 つ必要です。OpsWorks for Puppet Enterprise のバックアップの詳細については、「」を参照してください [OpsWorks for Puppet Enterprise Server のバックアップ](#)。

- バックアップのソースである既存のサーバーを作成するために使用したサービスロールとインスタンスプロファイルの ARN を準備します。
- 最新リリースの AWS CLI を実行していることを確認してください。AWS CLI ツールの更新の詳細については、AWS コマンドラインインターフェイスユーザーガイドの「[のインストール AWS CLI](#)」を参照してください。

制限事項

バックアップから新しいサーバーを作成して既存のサーバーを更新する場合、新しいサーバーを Puppet Enterprise OpsWorks サーバーの既存のサーバーとまったく同じにすることはできません。

- この手順は、AWS CLI または [AWS SDKs のいずれかを使用してのみ実行できます](#)。AWS Management Console を使用してバックアップから新しいサーバーを作成することはできません。
- 新しいサーバーに、アカウント内および AWS リージョン内の既存のサーバーと同じ名前は使用できません。名前は、バックアップのソースとして使用した既存のサーバーとは異なる必要があります。
- 既存のサーバーに接続されたノードは、新しいサーバーによって管理されません。次のいずれかを行う必要があります。
 - 複数の Puppet マスターでノードを管理することはできないため、異なるノードをアタッチします。
 - 既存のサーバー (バックアップのソース) から新しいサーバーと新しいカスタムドメインエンドポイントにノードを移行します。ノードを移行する方法の詳細については、[Puppet Enterprise のドキュメント](#)を参照してください。

カスタムドメインを使用するようにサーバーを更新する

既存の Puppet マスターを更新するには、バックアップ、カスタムドメイン、カスタム証明書、カスタムプライベートキーを指定するパラメータを追加して create-server コマンドを実行し、コピーを作成します。

1. create-server コマンドで指定できるサービスロールまたはインスタンスプロファイルの ARN がない場合は、「[を使用して Chef Automate サーバーを作成する AWS CLI](#)」のステップ 1 ~ 5 に従って、使用できるサービスロールとインスタンスプロファイルを作成します。
2. まだ作成していない場合は、カスタムドメインで新しいサーバーのベースにする既存の Puppet マスターのバックアップを探します。次のコマンドを実行して、アカウントとリージョンのすべ

ての OpsWorks for Puppet Enterprise バックアップに関する情報を表示します。使用するバックアップの ID を書き留めておきます。

```
aws opsworks-cm --region region name describe-backups
```

3. `create-server` コマンドを実行して OpsWorks、for Puppet Enterprise サーバーを作成します。
 - `--engine` 値は Puppet、`--engine-model` は Monolithic および `--engine-version` は 2019 または 2017 です。
 - サーバー名は、AWS アカウント内で各リージョン内で一意である必要があります。サーバー名は文字で始める必要があります。その後は文字、数字、またはハイフン (-) を最大 40 文字まで使用できます。
 - ステップ 3 と 4 でコピーしたインスタンスプロファイル ARN とサービスロール ARN を使用します。
 - 有効なインスタンスタイプは `c4.large`、`c4.xlarge`、または `c4.2xlarge` です。これらのインスタンスタイプの仕様についての詳細は、「Amazon EC2 User Guide」(Amazon EC2 ユーザーガイド) の「[Instance Types](#)」(インスタンスタイプ) を参照してください。
 - `--engine-attributes` パラメータはオプションです。Puppet 管理者パスワードを指定しない場合、サーバー作成プロセスで生成されます。`--engine-attributes` を追加する場合は、`PUPPET_ADMIN_PASSWORD` を指定します。これは、Puppet Enterprise コンソールウェブページにサインインする管理者パスワードです。パスワードに使用できる文字数は 8~32 文字 (ASCII) です。
 - SSH キーペアはオプションですが、コンソール管理者パスワードをリセットする必要がある場合に Puppet マスターに接続することができます。SSH キーペアの作成の詳細については、「Amazon EC2 User Guide」(Amazon EC2 ユーザーガイド) の「[Amazon EC2 Key Pairs](#)」(Amazon EC2 キーペア) を参照してください。
 - カスタムドメインを使用するには、コマンドに以下のパラメータを追加します。それ以外の場合は、Puppet マスター作成プロセスによって自動的にエンドポイントが生成されます。カスタムドメインを構成するには、3 つのパラメータすべてが必要です。これらのパラメータを使用するためのその他の要件については、AWS OpsWorks CM API リファレンス [CreateServer](#) の「」を参照してください。
 - `--custom-domain` - サーバーのオプションのパブリックエンドポイント (`https://aws.my-company.com` など)。
 - `--custom-certificate` - PEM 形式の HTTPS 証明書。値には、単一の自己署名証明書、または証明書チェーンを指定できます。

- `--custom-private-key` - HTTPS を使用してサーバーに接続するための PEM 形式のプライベートキー。プライベートキーは暗号化しないでください。パスワードやパスフレーズで保護することはできません。
- 週 1 回のシステムメンテナンスが必要です。次の形式で有効な値を指定する必要があります: `DDD:HH:MM`。指定時刻は協定世界時 (UTC) です。 `--preferred-maintenance-window` の値を指定しない場合、火曜日、水曜日、または金曜日の 1 時間がランダムでデフォルト値になります。
- `--preferred-backup-window` の有効な値は次の形式のいずれかで指定する必要があります: 日次バックアップの場合は `HH:MM`、週次バックアップの場合は `DDD:HH:MM`。指定時刻は UTC です。デフォルト値は日次で開始時間はランダムです。自動バックアップを無効にするには、代わりにパラメータ `--disable-automated-backup` を追加します。
- `--security-group-ids` に、1 つ以上のセキュリティグループ ID をスペースで区切って入力します。
- `--subnet-ids` には、サブネット ID を入力します。

```
aws opsworks-cm create-server --engine "Puppet" --engine-model "Monolithic"
--engine-version "2019" --server-name "server_name" --instance-profile-arn
"instance_profile_ARN" --instance-type "instance_type" --engine-attributes
'{"PUPPET_ADMIN_PASSWORD":"ASCII_password"}' --key-pair "key_pair_name" --
preferred-maintenance-window "ddd:hh:mm" --preferred-backup-window "ddd:hh:mm"
--security-group-ids security_group_id1 security_group_id2 --service-role-arn
"service_role_ARN" --subnet-ids subnet_ID
```

次の例では、カスタムドメインを使用する Puppet マスターを作成します。

```
aws opsworks-cm create-server \
--engine "Puppet" \
--engine-model "Monolithic" \
--engine-version "2019" \
--server-name "puppet-02" \
--instance-profile-arn "arn:aws:iam::1019881987024:instance-profile/aws-
opsworks-cm-ec2-role" \
--instance-type "c4.large" \
--engine-attributes '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSFrv1d"}' \
--custom-domain "my-puppet-master.my-corp.com" \
--custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
CERTIFICATE-----" \
```

```
--custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
RSA PRIVATE KEY-----" \
--key-pair "amazon-test"
--preferred-maintenance-window "Mon:08:00" \
--preferred-backup-window "Sun:02:00" \
--security-group-ids sg-b00000001 sg-b00000008 \
--service-role-arn "arn:aws:iam::044726508045:role/service-role/aws-opsworks-
cm-service-role" \
--subnet-ids subnet-383daa71
```

4. OpsWorks for Puppet Enterprise は、新しいサーバーの作成に約 15 分かかります。create-server コマンドの出力で、Endpoint 属性の値をコピーします。次に例を示します。

```
"Endpoint": "puppet-2019-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
```

create-server コマンドの出力を閉じたり、シェルセッションを閉じたりしないでください。今後表示されない重要な情報が出力に含まれている場合があります。create-server の結果からパスワードとスターターキットを取得するには、次のステップに進みます。

5. OpsWorks for Puppet Enterprise でパスワードを生成することを選択した場合は、jq などの JSON プロセッサを使用して、create-server 結果から使用可能な形式で抽出できます。jq をインストールした後、以下のコマンドを実行して Puppet 管理者パスワードとスターターキットを抽出できます。ステップ 3 で独自のパスワードを指定しなかった場合は、抽出した管理者パスワードを使いやすく安全な場所に保存してください。

```
#Get the Puppet password:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
"PUPPET_ADMIN_PASSWORD") | .Value'

#Get the Puppet Starter Kit:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
"PUPPET_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
```

Note

AWS Management Console で新しい Puppet マスタースターターキットを再生成することはできません。を使用して Puppet マスターを作成する場合 AWS CLI、前述の jq コマンドを実行して base64 でエンコードされたスターターキットを ZIP ファイルとして create-server 結果に保存します。

6. オプションで、`create-server` コマンド結果からスターターキットを抽出しなかった場合は、OpsWorks for Puppet Enterprise コンソールのサーバーのプロパティページから新しいスターターキットをダウンロードできます。
7. カスタムドメインを使用していない場合は、次のステップに進みます。サーバーでカスタムドメインを使用している場合は、エンタープライズの DNS 管理ツールで CNAME エントリを作成して、ステップ 4 でコピーした OpsWorks for Puppet Enterprise エンドポイントにカスタムドメインをポイントします。このステップを完了するまで、カスタムドメインを使用するサーバーにアクセスしたりサインインしたりすることはできません。
8. サーバーの作成プロセスが完了したら、「[スターターキットを使用して Puppet マスターを設定する](#)」に進みます。

以下の資料も参照してください。

- [を使用して Puppet Enterprise Master を作成する AWS CLI](#)
- [OpsWorks for Puppet Enterprise Server のバックアップと復元](#)
- [CreateServer](#) AWS OpsWorks CM API リファレンスの
- [create-server](#) 「[コマンドリファレンス](#)」の「AWS CLI」を参照してください。

AWS OpsWorks for Puppet Enterprise リソースでのタグの使用

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

タグとは、AWS リソースを識別および整理するためのメタデータとして使用される単語やフレーズのことです。OpsWorks for Puppet Enterprise では、1 つのリソースに最大 50 個のユーザー適用タグを含めることができます。各タグはキーと 1 つのオプションの値で構成されます。OpsWorks for Puppet Enterprise では、次のリソースにタグを適用できます。

- OpsWorks Puppet Enterprise サーバー用

• OpsWorks for Puppet Enterprise サーバーのバックアップ

AWS リソースのタグは、コストの追跡、リソースへのアクセスの制御、タスクを自動化するためのリソースのグループ化、目的またはライフサイクルステージによるリソースの整理に役立ちます。タグのメリットの詳細については、AWS Billing and Cost Management ユーザーガイドの[AWS タグ付け戦略](#) AWS Answers と [コスト配分タグの使用](#) を参照してください。

タグを使用して OpsWorks for Puppet Enterprise サーバーまたはバックアップへのアクセスを制御するには、AWS Identity and Access Management (IAM) でポリシーステートメントを作成または編集します。詳細については、AWS Identity and Access Management ユーザーガイドの「[Controlling Access to AWS Resources Using Resource Tags](#)」(リソースタグを使用した リソースへのアクセスの制御) を参照してください。

OpsWorks for Puppet Enterprise マスターにタグを適用すると、タグはマスターのバックアップ、バックアップを保存する Amazon S3 バケット、マスターの Amazon EC2 インスタンス、に保存されているマスターのシークレット AWS Secrets Manager、およびマスターで使用される Elastic IP アドレスにも適用されます。タグは、Puppet マスターの作成 AWS OpsWorks に使用する AWS CloudFormation スタックには伝達されません。

トピック

- [でのタグの仕組み AWS OpsWorks for Puppet Enterprise](#)
- [OpsWorks for Puppet Enterprise \(コンソール\) でのタグの追加と管理](#)
- [OpsWorks for Puppet Enterprise \(CLI\) でのタグの追加と管理](#)
- [以下の資料も参照してください。](#)

でのタグの仕組み AWS OpsWorks for Puppet Enterprise

このリリースでは、[AWS OpsWorks CM API](#) または AWS Management Console を使用してタグを追加および管理できます。AWS OpsWorks CM は、EC2 インスタンス、Secrets Manager のシークレット、Elastic IP アドレス、セキュリティグループ、S3 バケット、バックアップなど、サーバーに関連付けられている AWS リソースにサーバーに追加するタグも追加しようとします。

次の表は、OpsWorks for Puppet Enterprise でタグを追加および管理する方法の概要を示しています。

アクション	使用するもの
<p>OpsWorks Puppet Enterprise サーバーの新しいまたは手動で作成するバックアップにタグを追加します。</p>	<ul style="list-style-type: none"> • [Create Puppet Enterprise server (Puppet Enterprise サーバーを作成)] を選択し、[Configure advanced settings (詳細設定の設定)] ページでタグを追加します。 • 既存のサーバーの [Create backup (バックアップの作成)] ページで [Backups (バックアップ)] を選択し、[Create a backup of your Puppet Enterprise server (Puppet Enterprise サーバーのバックアップの作成)] ページでタグを追加します。 • CreateServer または CreateBackup コマンドに Tags パラメータを追加します。
<p>リソースのタグを表示します。</p>	<ul style="list-style-type: none"> • サーバーの詳細ページで、ナビゲーションペインで [Tags (タグ)] を選択します。 • サーバーの [Backups (バックアップ)] ページで、バックアップを選択し、[Edit backup (バックアップの編集)] を選択します。 • ListTagsForResource コマンドを実行します。
<p>バックアップが手動または自動のどちらで作成されたかにかかわらず、既存の OpsWorks Puppet Enterprise サーバーまたはバックアップにタグを追加します。</p>	<ul style="list-style-type: none"> • サーバーの詳細ページで、ナビゲーションペインで [Tags (タグ)] を選択し、[Edit (編集)] を選択します。 • サーバーの [Backups (バックアップ)] ページで、バックアップを選択し、[Edit backup (バックアップの編集)] を選択します。 • TagResource コマンドを実行します。
<p>リソースからタグを削除する</p>	<ul style="list-style-type: none"> • サーバーの詳細ページで、ナビゲーションペインで [Tags (タグ)] を選択し、[Edit (編集)] を選択します。削除するタグの横にある [X] を選択します。

アクション	使用するもの
	<ul style="list-style-type: none"> • サーバーの [Backups (バックアップ)] ページで、バックアップを選択し、[Edit backup (バックアップの編集)] を選択します。削除するタグの横にある [X] を選択します。 • UntagResource コマンドを実行します。

DescribeServers および DescribeBackups のレスポンスにタグ情報は含まれません。タグを表示するには、ListTagsForResource API を使用します。

OpsWorks for Puppet Enterprise (コンソール) でのタグの追加と管理

このセクションの手順は AWS Management Console で実行されます。

タグを追加する場合、タグのキーを空にすることはできません。キーは最大 127 文字で、Unicode 文字、数字、区切り文字、または特殊文字 (+ - = . _ : / @) のみを含めることができます。タグの値はオプションです。キーはあるが値はないタグであれば、追加できます。値は最大 255 文字とし、Unicode 文字、数字、区切り文字、または特殊文字 (+ - = . _ : / @) のみを含めることができます。

トピック

- [新しい OpsWorks for Puppet Enterprise Server にタグを追加する \(コンソール\)](#)
- [新しいバックアップにタグを追加する \(コンソール\)](#)
- [既存のサーバーにタグを追加または表示する \(コンソール\)](#)
- [既存のバックアップにタグを追加または表示する \(コンソール\)](#)
- [サーバからタグを削除する \(コンソール\)](#)
- [バックアップからタグを削除する \(コンソール\)](#)

新しい OpsWorks for Puppet Enterprise Server にタグを追加する (コンソール)

1. OpsWorks for Puppet Enterprise マスターを作成するための[前提条件](#)をすべて満たしてください。
2. [を使用して Puppet Enterprise Master を作成する AWS Management Console](#) の手順 1~8 に従います

3. 自動バックアップ設定を指定した後、[Configure advanced settings] (詳細設定の設定) ページの [Tags] (タグ) の領域にタグを追加します。最大 50 個のタグを追加できます。タグを追加したら、[次へ] を選択します。
4. [を使用して Puppet Enterprise Master を作成する AWS Management Console](#) のステップ 11 に進み、新しいサーバーに対して選択した設定を確認します。

新しいバックアップにタグを追加する (コンソール)

1. OpsWorks for Puppet Enterprise のホームページで、既存の Puppet マスターを選択します。
2. サーバーの詳細ページで、ナビゲーションペインで [Backups (バックアップ)] を選択します。
3. [Backups (バックアップ)] ページで、[Create backup (バックアップの作成)] を選択します。
4. タグを追加する。タグの追加が完了したら、[Create (作成)] を選択します。

既存のサーバーにタグを追加または表示する (コンソール)

1. OpsWorks for Puppet Enterprise のホームページで、既存の Puppet マスターを選択して詳細ページを開きます。
2. ナビゲーションペインで [Tags (タグ)] を選択するか、詳細ページの下部にある [View all tags (すべてのタグを表示)] を選択します。
3. [Tags (タグ)] ページで、[Edit (編集)] を選択します。
4. サーバー上のタグを追加または編集します。完了したら、[保存] を選択します。

Note

Puppet マスターのタグを変更すると、EC2 インスタンス、Elastic IP アドレス、セキュリティグループ、S3 バケット、バックアップなど、サーバーに関連付けられたリソースのタグも変更されることに注意してください。

既存のバックアップにタグを追加または表示する (コンソール)

1. OpsWorks for Puppet Enterprise のホームページで、既存の Puppet マスターを選択して詳細ページを開きます。

2. ナビゲーションペインで [Backups (バックアップ)] を選択するか、詳細ページの [Recent backups (最近のバックアップ)] 領域で [View all backups (すべてのバックアップを表示)] を選択します。
3. [Backups (バックアップ)] ページで、管理するバックアップを選択し、[Edit backup (バックアップの編集)] を選択します。
4. バックアップにタグを追加または編集します。完了したら、[Update (更新)] を選択します。

サーバからタグを削除する (コンソール)

1. OpsWorks for Puppet Enterprise のホームページで、既存の Puppet マスターを選択して詳細ページを開きます。
2. ナビゲーションペインで [Tags (タグ)] を選択するか、詳細ページの下部にある [View all tags (すべてのタグを表示)] を選択します。
3. [Tags (タグ)] ページで、[Edit (編集)] を選択します。
4. タグを削除するには、タグの横にある [X] を選択します。完了したら、[保存] を選択します。

Note

Puppet マスターのタグを変更すると、EC2 インスタンス、Elastic IP アドレス、セキュリティグループ、S3 バケット、バックアップなど、サーバーに関連付けられたリソースのタグも変更されることに注意してください。

バックアップからタグを削除する (コンソール)

1. OpsWorks for Puppet Enterprise のホームページで、既存の Puppet マスターを選択して詳細ページを開きます。
2. ナビゲーションペインで [Backups (バックアップ)] を選択するか、詳細ページの [Recent backups (最近のバックアップ)] 領域で [View all backups (すべてのバックアップを表示)] を選択します。
3. [Backups (バックアップ)] ページで、管理するバックアップを選択し、[Edit backup (バックアップの編集)] を選択します。
4. タグを削除するには、タグの横にある [X] を選択します。完了したら、[Update (更新)] を選択します。

OpsWorks for Puppet Enterprise (CLI) でのタグの追加と管理

このセクションの手順は AWS CLI で実行されます。タグの使用 AWS CLI を開始する前に、の最新リリースを実行していることを確認してください。のインストールまたは更新の詳細については AWS CLI、「AWS Command Line Interface ユーザーガイド」の「[AWS CLI のインストール](#)」を参照してください。

タグを追加する場合、タグのキーを空にすることはできません。キーは最大 127 文字で、Unicode 文字、数字、区切り文字、または特殊文字 (+ - = . _ : / @) のみを含めることができます。タグの値はオプションです。キーはあるが値はないタグであれば、追加できます。値は最大 255 文字とし、Unicode 文字、数字、区切り文字、または特殊文字 (+ - = . _ : / @) のみを含めることができます。

トピック

- [新しい OpsWorks for Puppet Enterprise Server \(CLI\) にタグを追加する](#)
- [新しいバックアップにタグを追加する \(CLI\)](#)
- [既存のサーバーまたはバックアップにタグを追加する \(CLI\)](#)
- [リソースタグを一覧表示する \(CLI\)](#)
- [リソースからタグを削除する \(CLI\)](#)

新しい OpsWorks for Puppet Enterprise Server (CLI) にタグを追加する

OpsWorks for Puppet Enterprise サーバーを作成するときに、を使用してタグ AWS CLI を追加できます。この手順では、サーバーの作成方法について詳しく説明していません。を使用して for Puppet Enterprise サーバーを作成する方法の詳細については AWS CLI、[を使用して Puppet Enterprise Master を作成する AWS CLI](#) このガイドの OpsWorks 「」を参照してください。サーバーには最大 50 個のタグを追加できます。

1. OpsWorks for Puppet Enterprise サーバーを作成するための[前提条件](#)をすべて満たしてください。
2. 「[を使用して Puppet Enterprise Master を作成する AWS CLI](#)」のステップ 1 ~ 4 を完了します。
3. ステップ 5 では、create-server コマンドを実行するときに、以下の例に示すように、コマンドに --tags パラメータを追加します。

```
aws opsworks-cm create-server ... --tags Key=Key1,Value=Value1  
Key=Key2,Value=Value2
```

以下の例では、create-server コマンドのタグ部分のみを示しています。

```
aws opsworks-cm create-server ... --tags Key=Stage,Value=Production  
Key=Department,Value=Marketing
```

4. 「[を使用して Puppet Enterprise Master を作成する AWS CLI](#)」の残りのステップを行います。タグが新しいサーバーに追加されたことを確認するには、このトピックの「[リソースタグを一覧表示する \(CLI\)](#)」の手順に従います。

新しいバックアップにタグを追加する (CLI)

OpsWorks for Puppet Enterprise サーバーの新しい手動バックアップを作成するときに、を使用してタグ AWS CLI を追加できます。この手順では、手動バックアップの作成方法について詳しく説明していません。手動バックアップの作成方法の詳細については、「」の「手動バックアップを実行するには AWS CLI」を参照してください [OpsWorks for Puppet Enterprise Server のバックアップ](#)。バックアップには最大 50 個のタグを追加できます。サーバにタグがある場合、新しいバックアップにはサーバのタグが自動的にタグ付けされます。

デフォルトでは、Puppet Enterprise サーバー OpsWorks 用に新しい を作成すると、自動バックアップが有効になります。このトピックの「[既存のサーバーまたはバックアップにタグを追加する \(CLI\)](#)」で説明されている tag-resource コマンドを実行して、自動バックアップにタグを追加できます。

- 手動バックアップの作成中にタグをバックアップに追加するには、以下のコマンドを実行します。コマンドのタグ部分のみを示しています。フル create-backup コマンドの例に関しては、[OpsWorks for Puppet Enterprise Server のバックアップ](#) の「AWS CLIで手動バックアップを実行するには」を参照してください。

```
aws opsworks-cm create-backup ... --tags Key=Key1,Value=Value1  
Key=Key2,Value=Value2
```

以下の例では、create-backup コマンドのタグ部分のみを示しています。

```
aws opsworks-cm create-backup ... --tags Key=Stage,Value=Production
Key=Department,Value=Marketing
```

既存のサーバーまたはバックアップにタグを追加する (CLI)

tag-resource コマンドを実行して、既存の Puppet Enterprise サーバーまたはバックアップ (バックアップが自動作成されたか手動で作成されたかにかかわらず) OpsWorks にタグを追加できます。ターゲットリソースの Amazon リソースナンバー (ARN) を指定して、タグを追加します。

1. タグを適用するリソースの ARN を取得するには:

- サーバーの場合は、describe-servers --server-name *server_name* を実行します。コマンドの結果には、サーバーの ARN が表示されます。
- バックアップの場合は、describe-backups --backup-id *backup_ID* を実行します。コマンドの結果には、バックアップの ARN が表示されます。を実行してdescribe-backups --server-name *server_name*、特定の OpsWorks for Puppet Enterprise サーバーのすべてのバックアップに関する情報を表示することもできます。

以下の例では、ServerArn コマンドの結果の describe-servers --server-name opsworks-cm-test のみを示しています。tag-resource コマンドに ServerArn 値を指定して、サーバーにタグを追加します。

```
{
  "Servers": [
    {
      ...
      "ServerArn": "arn:aws:opsworks-cm:us-west-2:123456789012:server/
opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
    }
  ]
}
```

2. ステップ 1 で返された ARN を使用して、tag-resource コマンドを実行します。

```
aws opsworks-cm tag-resource --resource-arn "server_or_backup_ARN" --tags
Key=Key1,Value=Value1 Key=Key2,Value=Value2
```

次に例を示します。

```
aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE" --tags Key=Stage,Value=Production Key=Department,Value=Marketing
```

3. タグが正常に追加されたことを確認するには、次の手順「[リソースタグを一覧表示する \(CLI\)](#)」に進みます。

リソースタグを一覧表示する (CLI)

`list-tags-for-resource` コマンドを実行して、for Puppet Enterprise サーバーまたはバックアップ OpsWorks にアタッチされているタグを表示できます。ターゲットリソースの ARN を指定して、そのタグを表示します。

1. タグを一覧表示するリソースの ARN を取得するには:
 - サーバーの場合は、`describe-servers --server-name server_name` を実行します。コマンドの結果には、サーバーの ARN が表示されます。
 - バックアップの場合は、`describe-backups --backup-id backup_ID` を実行します。コマンドの結果には、バックアップの ARN が表示されます。を実行して `describe-backups --server-name server_name`、特定の OpsWorks for Puppet Enterprise サーバーのすべてのバックアップに関する情報を表示することもできます。
2. ステップ 1 で返された ARN を使用して、`list-tags-for-resource` コマンドを実行します。

```
aws opsworks-cm list-tags-for-resource --resource-arn "server_or_backup_ARN"
```

次に例を示します。

```
aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
```

リソースにタグがある場合、コマンドは以下のような結果を返します。

```
{
  "Tags": [
    {
      "Key": "Stage",
```

```
        "Value": "Production"
      },
      {
        "Key": "Department",
        "Value": "Marketing"
      }
    ]
  }
}
```

リソースからタグを削除する (CLI)

`untag-resource` コマンドを実行して、Puppet Enterprise サーバーまたはバックアップ OpsWorks のからタグを削除できます。リソースが削除されると、リソースのタグも削除されます。ターゲットリソースの Amazon リソース番号 (ARN) を指定して、タグを削除します。

1. タグを削除するリソースの ARN を取得するには:
 - サーバーの場合は、`describe-servers --server-name server_name` を実行します。コマンドの結果には、サーバーの ARN が表示されます。
 - バックアップの場合は、`describe-backups --backup-id backup_ID` を実行します。コマンドの結果には、バックアップの ARN が表示されます。を実行して `describe-backups --server-name server_name`、特定の OpsWorks for Puppet Enterprise サーバーのすべてのバックアップに関する情報を表示することもできます。
2. ステップ 1 で返された ARN を使用して、`untag-resource` コマンドを実行します。削除するタグのみを指定します。

```
aws opsworks-cm untag-resource --resource-arn "server_or_backup_ARN" --tags
Key=Key1,Value=Value1 Key=Key2,Value=Value2
```

この例では、`untag-resource` コマンドは、キーが Stage で値が Production のタグのみを削除します。

```
aws opsworks-cm untag-resource --resource-arn "arn:aws:opsworks-cm:us-
west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
--tags Key=Stage,Value=Production
```

3. タグが正常に削除されたことを確認するには、このトピックの「[リソースタグを一覧表示する \(CLI\)](#)」の手順に従います。

以下の資料も参照してください。

- [を使用して Puppet Enterprise Master を作成する AWS CLI](#)
- [OpsWorks for Puppet Enterprise Server のバックアップ](#)
- [AWS タグ付け戦略](#)
- AWS Identity and Access Management ユーザーガイドの[AWS リソースタグを使用したリソースへのアクセスの制御](#)
- 「AWS Billing and Cost Management ユーザーガイド」の「[コスト配分タグの使用](#)」
- 「[CreateBackup](#)」 (AWS OpsWorks CM API リファレンス)
- 「[CreateServer](#)」 (AWS OpsWorks CM API リファレンス)
- 「[TagResource](#)」 (AWS OpsWorks CM API リファレンス)
- 「[ListTagsForResource](#)」 (AWS OpsWorks CM API リファレンス)
- 「[UntagResource](#)」 (AWS OpsWorks CM API リファレンス)

OpsWorks for Puppet Enterprise Server のバックアップと復元

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、OpsWorks for Puppet Enterprise サーバーをバックアップおよび復元する方法について説明します。

トピック

- [OpsWorks for Puppet Enterprise Server のバックアップ](#)
- [OpsWorks for Puppet Enterprise Server をバックアップから復元する](#)

OpsWorks for Puppet Enterprise Server のバックアップ

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Puppet Enterprise サーバーバックアップ OpsWorks の日次または週次定期を定義し、ユーザーに代わって Amazon Simple Storage Service (Amazon S3) にバックアップを保存させることができます。または、必要に応じて手動でバックアップを実行することもできます。

バックアップが Amazon S3 に保存されるため、追加料金を負担します。30 世代を上限としてバックアップ保持期間を定義できます。AWS サポートチャンネルを使用してサービスリクエストを送信し、その制限を変更することができます。Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、「[S3 バケットを空にする方法](#)」または「[S3 バケットを削除する方法](#)」を参照してください。

OpsWorks for Puppet Enterprise マスターのバックアップにタグを追加できます。OpsWorks for Puppet Enterprise マスターにタグを追加した場合、Puppet マスターの自動バックアップはそれらのタグを継承します。バックアップでタグを追加および管理する方法の詳細については、このガイドの「[AWS OpsWorks for Puppet Enterprise リソースでのタグの使用](#)」を参照してください。

トピック

- [自動バックアップ](#)
- [手動バックアップ](#)
- [バックアップの削除](#)

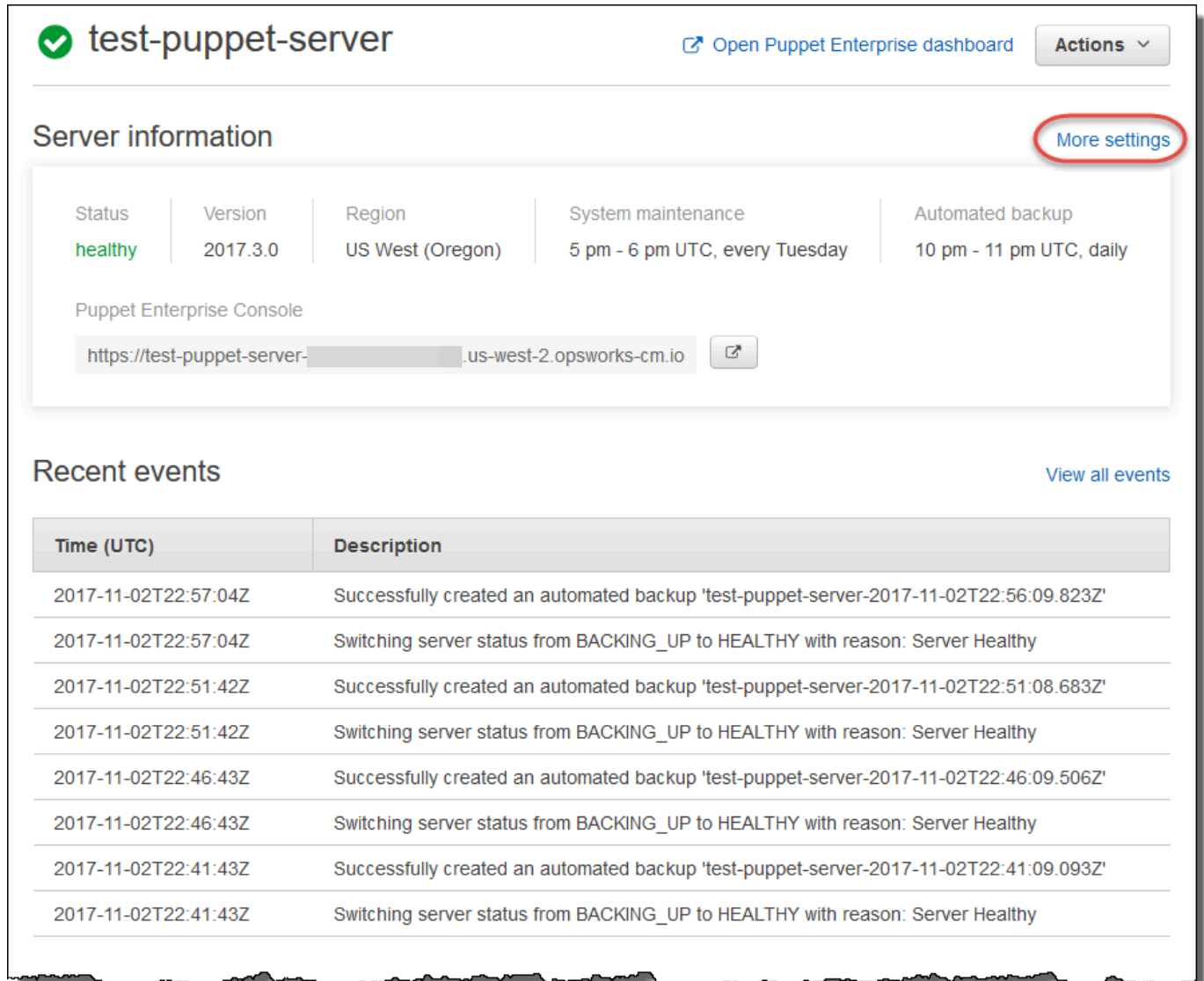
自動バックアップ

OpsWorks for Puppet Enterprise サーバーを設定するときは、自動バックアップまたは手動バックアップを選択します。OpsWorks for Puppet Enterprise は、セットアップウィザードの「詳細設定の構成」ページの「自動バックアップ」セクションで選択した時間および日に自動バックアップを開始

します。サーバーがオンラインになったら、サーバーのプロパティページで次のステップを実行してバックアップ設定を変更できます。

自動バックアップ設定を変更するには

1. サーバーのプロパティページで、[More settings] を選択します。



The screenshot shows the AWS OpsWorks console for a server named 'test-puppet-server'. The server status is 'healthy'. The 'More settings' link is circled in red. Below the server information, there is a table of recent events.

Time (UTC)	Description
2017-11-02T22:57:04Z	Successfully created an automated backup 'test-puppet-server-2017-11-02T22:56:09.823Z'
2017-11-02T22:57:04Z	Switching server status from BACKING_UP to HEALTHY with reason: Server Healthy
2017-11-02T22:51:42Z	Successfully created an automated backup 'test-puppet-server-2017-11-02T22:51:08.683Z'
2017-11-02T22:51:42Z	Switching server status from BACKING_UP to HEALTHY with reason: Server Healthy
2017-11-02T22:46:43Z	Successfully created an automated backup 'test-puppet-server-2017-11-02T22:46:09.506Z'
2017-11-02T22:46:43Z	Switching server status from BACKING_UP to HEALTHY with reason: Server Healthy
2017-11-02T22:41:43Z	Successfully created an automated backup 'test-puppet-server-2017-11-02T22:41:09.093Z'
2017-11-02T22:41:43Z	Switching server status from BACKING_UP to HEALTHY with reason: Server Healthy

2. 自動バックアップを無効にするには、[Enable automated backups] (自動バックアップの有効化) オプションの [No] (いいえ) を選択します。変更を保存します。次のステップに進む必要はありません。
3. [Automated Backup] セクションで、頻度、開始時刻、または保持する世代数を変更します。変更を保存します。

手動バックアップ

手動バックアップは、いつでも開始することも AWS Management Console、[create-backup](#) コマンドを実行して開始 AWS CLI することもできます。手動バックアップは、保存される自動バックアップの最大 30 世代には含まれません。最大 10 件の手動バックアップが保存され、Amazon S3 から手動で削除する必要があります。

で手動バックアップを実行するには AWS Management Console

1. [Puppet Enterprise servers] ページで、バックアップするサーバーを選択します。
2. サーバーのプロパティページの左のナビゲーションペインで、[Backups] を選択します。
3. [Create backup] (バックアップの作成) を選択します。
4. ページでバックアップの [Status] 欄に緑色のチェックマークが表示されると、手動バックアップは完了です。

で手動バックアップを実行するには AWS CLI

OpsWorks for Puppet Enterprise サーバーの新しい手動バックアップを作成するときに、タグを追加できます。手動バックアップを作成するときにタグを追加する方法の詳細については、「[新しいバックアップにタグを追加する \(CLI\)](#)」を参照してください

- 手動バックアップを開始するには、次の AWS CLI コマンドを実行します。

```
aws opsworks-cm --region region name create-backup --server-name "Puppet server name" --description "optional descriptive string"
```

バックアップの削除

バックアップを削除すると、バックアップが保存されている S3 バケットから完全に削除されます。

でバックアップを削除するには AWS Management Console

1. [Puppet Enterprise servers] ページで、バックアップするサーバーを選択します。
2. サーバーのプロパティページの左のナビゲーションペインで、[Backups] を選択します。
3. 削除するバックアップを選択してから、[Delete backup] を選択します。一度に 1 つのバックアップのみを選択できます。

- 削除の確認を指示されたら、[Delete the backup, which is stored in an S3 bucket] チェックボックスにチェックを入れ、[Yes, Delete] を選択します。

でバックアップを削除するには AWS CLI

- バックアップを削除するには、次の AWS CLI コマンドを実行し、 の値を、削除するバックアップの `--backup-id` ID に置き換えます。バックアップ IDs形式は `ServerName-yyyyMMddHHmssSSS` です。例えば `puppet-server-20171218132604388` です。

```
aws opsworks-cm --region region name delete-backup --backup-id ServerName-yyyyMMddHHmssSSS
```

OpsWorks for Puppet Enterprise Server をバックアップから復元する

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

使用可能なバックアップを参照した後、OpsWorks for Puppet Enterprise サーバーを復元するポイントインタイムを簡単に選択できます。サーバーのバックアップには、設定管理ソフトウェアの永続的なデータが含まれています。たとえば、モジュール、クラス、ノード関連付け、データベース情報 (レポートやファクトなど) などです。サーバーのインプレース復元を実行する (つまり、既存の OpsWorks for Puppet Enterprise サーバーを新しい EC2 インスタンスに復元する) と、サーバーの復元に使用するバックアップ時に登録されたノードが再登録され、復元が成功し、Puppet Enterprise サーバーの状態 OpsWorks が に復元されると、トラフィックが新しいインスタンスに切り替わりますHealthy。新しく作成された OpsWorks for Puppet Enterprise サーバーに復元しても、ノード接続は維持されません。サーバーを復元すると、Puppet ソフトウェアのバージョンが更新されません。選択したバックアップにあるものと同じ Puppet バージョンと設定管理データが適用されます。

通常、サーバの復元には新しいサーバを作成するよりも時間がかかり、この時間は選択するバックアップのサイズによります。復元が完了すると、古い EC2 インスタンスは Running または Stopped 状態のままになりますが、一時的にのみです。最終的には終了します。

このリリースでは、を使用して、for Puppet Enterprise で Puppet マスター OpsWorks を AWS CLI 復元できます。

Note

`restore-server` コマンドは、現在のインスタンスタイプを変更する場合や、紛失または漏洩が発生した SSH キーの復元または設定を行う場合にも実行できます。

サーバーをバックアップから復元するには

1. で AWS CLI、次のコマンドを実行して、使用可能なバックアップとその IDs のリストを返します。使用するバックアップの ID を書き留めておきます。バックアップ IDs 形式は `myServerName-yyyyMMddHHmmssSSS` です。

```
aws opsworks-cm --region region name describe-backups
```

2. 以下のコマンドを実行します。

```
aws opsworks-cm --region region name restore-server --backup-id "myServerName-yyyyMMddHHmmssSSS" --instance-type "Type of instance" --key-pair "name of your EC2 key pair" --server-name "name of Puppet master"
```

次に例を示します。

```
aws opsworks-cm --region us-west-2 restore-server --backup-id "MyPuppetServer-20161120122143125" --server-name "MyPuppetServer"
```

3. 復元が完了するまで待ちます。

Puppet Enterprise OpsWorks の システムメンテナンス

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合

は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

必須のシステムメンテナンスにより、セキュリティ更新プログラムを含むで AWS テストされた最新バージョンの Puppet Server が、常に for Puppet Enterprise サーバーで実行されていることが保証され OpsWorks ます。システムメンテナンスは週に 1 回以上実行する必要があります。を使用すると AWS CLI、必要に応じて毎日の自動メンテナンスを設定できます。を使用して AWS CLI、スケジュールされたシステムメンテナンスに加えて、オンデマンドでシステムメンテナンスを実行することもできます。

Puppet ソフトウェアの新しいバージョンが利用可能になると、システムメンテナンスは AWS テストが完了次第、サーバーにある Puppet サーバーのバージョンを自動更新するように設計されています。AWS は広範なテストを実行して、Puppet のアップグレードが本番環境に対応しており、既存の顧客環境を中断しないことを確認します。そのため、Puppet ソフトウェアリリースと、Puppet Enterprise サーバー用の既存の OpsWorks へのアプリケーションの可用性との間に遅延が生じる可能性があります。Puppet ソフトウェアの使用可能なバージョンをオンデマンドで更新するには、このトピックの「[オンデマンドでのシステムメンテナンスの開始](#)」を参照してください。

システムメンテナンスでは、メンテナンスプロセスの一環として実行されるバックアップから新しいインスタンスを起動します。これにより、定期的なメンテナンスを受ける Amazon EC2 インスタンスの劣化や障害によるリスクを軽減することができます。

Important

システムメンテナンスでは、OpsWorks for Puppet Enterprise サーバーに追加したファイルまたはカスタム設定がすべて削除されます。失われた設定やファイルを回復する詳しい方法については、このトピックの「[メンテナンス後のカスタム設定およびカスタムファイルの復旧](#)」を参照してください。

トピック

- [システムメンテナンスの設定](#)
- [オンデマンドでのシステムメンテナンスの開始](#)
- [メンテナンス後のカスタム設定およびカスタムファイルの復旧](#)

システムメンテナンスの設定

Puppet Enterprise サーバー OpsWorks 用に新しい を作成する場合、システムメンテナンスを開始するために、[協定世界時](#) (UTC) で曜日と時刻を設定できます。メンテナンスは、指定した時間中に開始します。システムメンテナンス中はサーバーがオフラインになることを想定する必要があるため、通常の営業時間内でサーバー需要が低い時刻を選択します。メンテナンスの進行中は、サーバーのステータスが UNDER_MAINTENANCE になります。

次のスクリーンショットに示すように、サーバー OpsWorks の設定ページのシステムメンテナンスエリアで設定を変更することで、既存の Puppet Enterprise サーバーのシステムメンテナンス設定を変更することもできます。

Server Information

Name, region and type

Puppet Enterprise server name test-puppet-server

Puppet Enterprise server region US West (Oregon)

EC2 instance type c4.large

Resources

CloudFormation stack [aws-opsworks-cm-instance-test-puppet-server](#)

Network and security

Service role [aws-opsworks-cm-service-role](#)

Instance profile [aws-opsworks-cm-ec2-role](#)

System maintenance

AWS OpsWorks installs updates for Puppet Enterprise minor versions or security packages in the time range and on the weekday that you specify here. **Your Puppet Enterprise server will be offline during system maintenance.**

Start day ⓘ

Start time (UTC) ⓘ

[System maintenance] セクションで、システムメンテナンスを開始する日付と時刻を設定します。

を使用したシステムメンテナンスの設定 AWS CLI

システムメンテナンスの自動開始時刻を AWS CLI で設定することもできます。AWS CLI では、曜日の 3 文字のプレフィックスを省略することで、必要に応じて毎日の自動メンテナンスを設定できます。

`create-server` コマンドで、サーバーインスタンスを作成する要件 (インスタンスタイプ、インスタンスプロファイル ARN、サービスロール ARN など) を指定した後で、`--preferred-maintenance-window` パラメータをコマンドに追加します。次の `create-server` の例では、`--preferred-maintenance-window` を `Mon:08:00` に設定しています。これで、毎月曜の午前 8 時 0 分 (UTC) にメンテナンスが開始されます。

```
aws opsworks-cm create-server --engine "Puppet" --engine-model "Monolithic"
--engine-version "2017" --server-name "puppet-06" --instance-profile-arn
"arn:aws:iam::1119001987000:instance-profile/aws-opsworks-cm-ec2-role"
--instance-type "c4.large" --key-pair "amazon-test" --service-role-arn
"arn:aws:iam::044726508045:role/aws-opsworks-cm-service-role" --preferred-maintenance-
window "Mon:08:00"
```

`update-server` コマンドでは、必要に応じて、`--preferred-maintenance-window` の値のみを更新できます。次の例では、メンテナンス時刻を金曜の午後 6 時 15 分 (UTC) に設定しています。

```
aws opsworks-cm update-server --server-name "puppet-06" --preferred-maintenance-window
"Fri:18:15"
```

メンテナンス時間の開始時刻を毎日午後 6 時 15 分 (UTC) に変更するには、次の例に示すように、曜日を表す 3 文字のプレフィックスを省略します。

```
aws opsworks-cm update-server --server-name "puppet-06" --preferred-maintenance-window
"18:15"
```

を使用して優先システムメンテナンスウィンドウを設定する方法の詳細については AWS CLI、[「create-server」](#) および [「update-server」](#) を参照してください。

オンデマンドでのシステムメンテナンスの開始

システムメンテナンスをオンデマンドで開始するには、毎週または毎日の自動メンテナンスが設定されている場合以外は、次の AWS CLI コマンドを実行します。AWS Management Console でオンデマンドメンテナンスを開始することはできません。

```
aws opsworks-cm start-maintenance --server-name server_name
```

このコマンドの詳細については、[「start-maintenance」](#) を参照してください。

メンテナンス後のカスタム設定およびカスタムファイルの復旧

システムメンテナンスでは、OpsWorks for Puppet Enterprise サーバーに追加したカスタムファイルまたは設定を削除または変更できます。

メンテナンスの実行後、RunCommand または SSH を使用して追加したファイルや設定が Puppet マスターに見つからない場合は、Amazon Machine Image (AMI) を使用して、新しい Amazon EC2 インスタンスを起動できます。サーバーのメンテナンス前の設定に基づいて構築された AMI を利用できます。

新しいインスタンスは、メンテナンス前の Puppet マスターと同じ状態であり、見つからないファイルや設定が含まれています。

Important

新しいインスタンスを使用してサーバーを復元することはできません。インスタンスを Puppet マスターとして実行することはできません。インスタンスは、ファイルや設定の復旧にのみ使用できます。

AMI から EC2 インスタンスを起動するには、Amazon EC2 コンソールで [Launch] (起動) ウィザードを開き、[My AMIs] (マイAMI) を選択して、サーバー名と同じ名前の AMI を選択します。他の任意のインスタンスを起動する手順と同様に、Amazon EC2 ウィザードの手順に従います。

OpsWorks for Puppet Enterprise でノードを自動的に追加する

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このトピックでは、Amazon Elastic Compute Cloud (Amazon EC2) ノードを OpsWorks for Puppet Enterprise サーバーに自動的に追加する方法について説明します。「[Puppet マスターで管理する](#)

[ノードを追加する](#)」では、`associate-node` コマンドを使用して一度に 1 つのノードを Puppet Enterprise サーバーに追加する方法について学びました。このトピックのコードは、ユーザーが介入しない方法を使用して複数のノードを自動的に追加する方法を示しています。ユーザーが介入しない (または自動での) 新しいノードの関連付けの推奨手段は、Amazon EC2 ユーザーデータを設定することです。デフォルトでは、OpsWorks for Puppet Enterprise サーバーは Ubuntu、Amazon Linux、および RHEL ノードオペレーティングシステムで既に [puppet-agent](#) 使用できます。

ノードの関連付けを解除する方法については、このガイド [OpsWorks for Puppet Enterprise Server からノードの関連付けを解除する](#) の「」および for Puppet Enterprise API ドキュメント [disassociate-node](#) の OpsWorks 「」を参照してください。

ステップ 1: インスタンスプロファイルとして使用する IAM ロールを作成する

EC2 インスタンスプロファイルとして使用する AWS Identity and Access Management (IAM) ロールを作成し、次のポリシーを IAM ロールにアタッチします。このポリシーでは、ノード登録時に、`opsworks-cm` API を使用した EC2 インスタンスとの通信が許可されています。インスタンスプロファイルの詳細については、Amazon EC2 のドキュメントの [「Using Instance Profiles」](#) (インスタンスプロファイルの使用) を参照してください。IAM ロールを作成する方法については、Amazon EC2 のドキュメントの [「Creating an IAM Role in the Console」](#) (コンソールでの IAM ロールの作成) を参照してください。。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "opsworks-cm:AssociateNode",
        "opsworks-cm:DescribeNodeAssociationStatus",
        "opsworks-cm:DescribeServers",
        "ec2:DescribeTags"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

AWS OpsWorks には、前述のポリシーステートメントで IAM ロールを作成するために使用できる AWS CloudFormation テンプレートが用意されています。次の AWS CLI コマンドは、このテンプレ

レートを使用してインスタンスプロファイルロールを作成します。デフォルトのリージョンで新しい AWS CloudFormation スタックを作成する場合は、`--region`パラメータを省略できます。

```
aws cloudformation --region region ID create-stack --stack-name myPuppetinstanceprofile
--template-url https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/
misc/owpe/opsworks-cm-nodes-roles.yaml --capabilities CAPABILITY_IAM
```

ステップ 2: 自動関連付けスクリプトを使用してインスタンスを作成する

EC2 インスタンスを作成するには、[Starter Kit](#) に含まれているユーザーデータスクリプトを EC2 インスタンスの手順、Amazon EC2 Auto Scaling グループの起動設定、または AWS CloudFormation テンプレートの `userdata` セクションにコピーします。このスクリプトは、Ubuntu および Amazon Linux オペレーティングシステムを実行している EC2 インスタンスでのみサポートされています。ユーザーデータへのスクリプトの追加の詳細については、Amazon EC2 documentation ドキュメントの「[Running Commands on Your Linux Instance at Launch](#)」(Linux インスタンスでの起動時のコマンドの実行)を参照してください。新規ノードを作成する最も簡単な方法は、[Amazon EC2 instance launch wizard](#) (Amazon EC2 インスタンス起動ウィザード)を使用することです。このチュートリアルでは、「[OpsWorks for Puppet Enterprise の開始方法](#)」で説明された Apache ウェブサーバーのサンプルモジュールのセットアップを使用します。

1. スターターキットのユーザーデータスクリプトは、opsworks-cm API の [associate-node](#) コマンドを実行して、新しいノードを Puppet マスターに関連付けます。このリリースでは、ほとんどのバージョンをまだ実行していない場合に備えて、ノード AWS CLI に の最新バージョンもインストールされます `up-to-date`。このスクリプトを便利な場所に `userdata.sh` として保存します。

デフォルトでは、新しく登録されたノードの名前はインスタンス ID です。

2. EC2 ドキュメントの[インスタンスの作成](#)の手順に従い、ここで説明する変更を加えます。EC2 インスタンス起動ウィザードで、Amazon Linux AMI を選択します。
3. [Configure Instance Details] ページで、[`myPuppetinstanceprofile`] を選択します。これは、「[ステップ 1: インスタンスプロファイルとして使用する IAM ロールを作成する](#)」で IAM ロールとして作成したロールです。
4. [Advanced Details] 領域で、ステップ 1 で作成した `userdata.sh` スクリプトをアップロードします。
5. [Add Storage] ページで必要な変更はありません。[Add Tags] に進みます。

EC2 インスタンスにタグを適用することで、`userdata.sh` の動作をカスタマイズできます。この例では、`apache_webserver` ロールをノードに適用します。これを行うには、値が `pp_role` である `apache_webserver` タグを追加します。

ノードで `pp_role` の値をノードのエージェント証明書に永続的に保存されているデータ値に設定することで、ノードを信頼されたものとして分類できます。詳細については、Puppet プラットフォームドキュメントの「[Extension requests \(permanent certificate data\)](#)」を参照してください。

6. [Configure Security Group] (セキュリティグループの設定) ページで、[Add Rule] (ルールの追加) を選択し、[HTTP] タイプを選択してこの例では Apache ウェブサーバーでポート番号 8080 を開きます。
7. Review and Launch (確認と作成) を選択してから、Launch (起動) を選択します。新しいノードが開始されると、「[スターターキットの Apache の例をセットアップする](#)」でセットアップしたサンプルモジュールの Apache 設定が適用されます。
8. 新しいノードのパブリック DNS にリンクされたウェブページを開くと、Puppet に管理された Apache ウェブサーバーがホストするウェブサイトが表示されるはずですが。

OpsWorks for Puppet Enterprise Server からノードの関連付けを解除する

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、OpsWorks for Puppet Enterprise サーバーによる管理からマネージドノードの関連付けを解除または削除する方法について説明します。このオペレーションはコマンドラインまたは Puppet Enterprise コンソールで実行されます。OpsWorks for Puppet Enterprise マネジメントコンソールでノードの関連付けを解除することはできません。現在、OpsWorks for Puppet Enterprise API では、複数のノードをバッチ削除することはできません。このセクションのコマンドでは、一度に 1 つのノードの関連付けを切り離します。

Puppet マスターを削除する場合は、ノードがサーバーへの再接続を試行することなく動作を続けるように、サーバーとノードとの関連付けを切り離すことをお勧めします。これを行うには、[disassociate-node](#) AWS CLI コマンドを実行します。PE からノードを完全に削除するには、ノードの関連付けを切り離し、その証明書を取り消す必要があります。これにより、ノードが Puppet マスターに対して継続的にチェックインを試行しなくなります。また、Puppet マスターを使用して管理が不要になった [puppet-agent](#) をノードからアンインストールする必要があります。

ノードの関連付けを切り離すには

1. で AWS CLI、次のコマンドを実行してノードの関連付けを解除します。[*Node_name*] (ノード名) は関連付けを切り離すノードの名前です。Amazon EC2 インスタンスでは、これはインスタンス ID です。[*Server_name*] (サーバー名) は、ノードの関連付けを切り離す Puppet マスターの名前です。両方のパラメーターとも必須です。デフォルトのリージョンにない Puppet マスターからノードの関連付けを切り離さない限り、--region パラメーターは必須ではありません。

```
aws opsworks-cm --region Region_name disassociate-node --node-name Node_name --server-name Server_name
```

コマンドの例を次に示します。

```
aws opsworks-cm --region us-west-2 disassociate-node --node-name i-0010zzz00d66zzz90 --server-name opsworkstest
```

2. 関連付けの解除が完了したことを示す応答メッセージが表示されるまで待ちます。

OpsWorks for Puppet Enterprise サーバーを削除する方法の詳細については、「」を参照してください [OpsWorks for Puppet Enterprise Server を削除する](#)。

以下の資料も参照してください。

- Puppet Enterprise ドキュメントの [Remove nodes](#)

OpsWorks for Puppet Enterprise Server を削除する

⚠ Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、OpsWorks for Puppet Enterprise サーバーを削除する方法について説明します。サーバーを削除すると、サーバーに保存されていたイベント、ログ、およびモジュールも削除されます。サポートリソース (Amazon Elastic Compute Cloud インスタンス、Amazon Elastic Block Store ポリユームなど) やすべての自動バックアップも削除されます。

サーバーを削除してもノードは削除されませんが、これらは削除されたサーバーによって管理されなくなり、継続的に再接続が試行されます。このため、Puppet マスターを削除する場合は、管理されているノードの関連付けを事前に解除することをお勧めします。このリリースでは、AWS CLI コマンドを実行してノードの関連付けを解除できます。

ステップ 1: 管理されているノードの関連付けを解除する

Puppet マスターを削除する場合は、ノードがサーバーへの再接続を試行することなく動作を続行できるように、サーバーとノードとの関連付けを解除します。これを行うには、[disassociate-node](#) AWS CLI コマンドを実行します。

ノードの関連付けを切り離すには

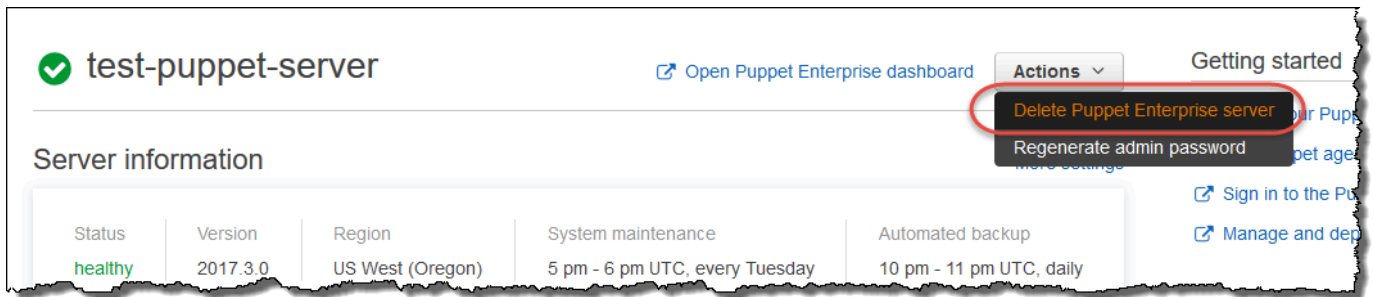
1. で AWS CLI、次のコマンドを実行してノードの関連付けを解除します。*Server_name* は、ノードが関連付けられている Puppet マスターの名前です。--node-name の値はインスタンス ID を指定できます。

```
aws opsworks-cm --region Region_name disassociate-node --node-name Node_name --server-name Server_name
```

2. 関連付けの解除が完了したことを示す応答メッセージが表示されるまで待ちます。

ステップ 2: サーバーを削除する

1. ダッシュボードのサーバータイトルで、[Actions] メニューを展開します。



2. [Delete Puppet Enterprise server] を選択します。
3. 削除の確認を指示されたら、関連ロールおよびリソースを削除する場合はチェックボックスにチェックを入れ、[Yes, Delete] を選択します。

以下の資料も参照してください。

- [OpsWorks for Puppet Enterprise Server からノードの関連付けを解除する](#)

OpsWorks for Puppet Enterprise サーバーを Amazon Elastic Compute Cloud (Amazon EC2) に移行する方法

⚠ Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

以下の手順では、 の外部で設定管理のニーズに対して Puppet Enterprise を引き続き使用する場合に備えて、既存の Puppet Enterprise サーバーを Amazon EC2 に移行する方法について説明します OpsWorks。

トピック

- [ステップ 1: Puppet に連絡してライセンスを購入してください](#)
- [ステップ 2: OpsWorks for Puppet Enterprise サーバーの詳細を取得する](#)
- [ステップ 3: OpsWorks for Puppet Enterprise サーバーのバックアップを作成する](#)
- [ステップ 4: 新規の EC2 インスタンスを起動する](#)
- [ステップ 5: 新規 EC2 インスタンスに Puppet Enterprise をインストールしてください](#)
- [ステップ 6: 新しい EC2 インスタンスにバックアップを復元してください](#)
- [ステップ 7: Puppet ライセンスを設定してください](#)
- [ステップ 8: ノードを移行してください](#)
- [ステップ 9: OpsWorks for Puppet Enterprise サーバーを削除する](#)

ステップ 1: Puppet に連絡してライセンスを購入してください

サーバーを EC2 に移行しても、新しいインスタンスには Puppet ライセンスは付属していません。ライセンスキーを購入するには、[Puppet ウェブサイト](#)の指示に従ってください。

ステップ 2: OpsWorks for Puppet Enterprise サーバーの詳細を取得する

OpsWorks for Puppet Enterprise サーバーの値を検索して保存します。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/s3/> で Amazon S3 コンソールを開きます。

OpsWorks for Puppet Enterprise サーバーの既存の Amazon S3 バケットの名前をコピーします。バケット名は `aws-opsworks-cm-server-name-random-string` という形式です。

2. `aws opsworks-cm describe-servers` コマンドを実行して、OpsWorks for Puppet Enterprise サーバーの設定を取得します。

```
aws opsworks-cm describe-servers \  
  --server-name server-name \  
  --region region
```

レスポンスから

`InstanceType`、`KeyPair`、`SubnetIds`、`SecurityGroupIds`、`InstanceProfileArn`、`Endpoint` の値を保存します。

3. SSH を使用して、既存の OpsWorks for Puppet Enterprise サーバーに接続します。EC2 コンソールでは SSH の代わりにセッションマネージャーを使用できます。

以下のコマンドを実行します。

```
rpm -qa | grep opsworks-cm-puppet-enterprise | cut -d '-' -f 5
```

レスポンスには Puppet エンタープライズバージョン (例えば 2019.8.10) が表示されます。この値を保存してください。

次のステップでは、SSH または Systems Manager を使用します。

ステップ 3: OpsWorks for Puppet Enterprise サーバーのバックアップを作成する

1. 以下のコマンドを実行してローカルバックアップを作成します。

```
mkdir /tmp/puppet-backup/  
sudo /opt/puppetlabs/bin/puppet-backup create --dir=/tmp/puppet-backup/
```

2. 次のコマンドを実行して、バックアップの名前を保存します。

```
ls /tmp/puppet-backup/  
PUPPET_BACKUP=$(ls /tmp/puppet-backup/)
```

3. 次のコマンドを実行して、S3 バケットにパッケージをアップロードします。**S3- ####** を [ステップ 2: OpsWorks for Puppet Enterprise サーバーの詳細を取得する](#) のステップ 1 の値に置き換えます。

```
aws s3 cp /tmp/puppet-backup/PUPPET_BACKUP s3://S3_Bucket/tmp/puppet-backup/
```

PUPPET_BACKUP と S3_BUCKET の値に保存してください。これらの値を新しい EC2 インスタンスにインポートします。

SSH または Systems Manager セッションを終了できます。

ステップ 4: 新規の EC2 インスタンスを起動する

OpsWorks for Puppet Enterprise サーバーと同じ設定を使用して、<https://console.aws.amazon.com/ec2/> の EC2 コンソールから [新しい EC2 EC2 インスタンスを起動](#) します。

パラメータ名	値
OS	Amazon Linux 2
インスタンスタイプ	ステップ 2: OpsWorks for Puppet Enterprise サーバーの詳細を取得する のステップ 2 からの InstanceType の値です。
キーペア名	ステップ 2: OpsWorks for Puppet Enterprise サーバーの詳細を取得する のステップ 2 からの KeyPair の値です。
VPC	ステップ 2: OpsWorks for Puppet Enterprise サーバーの詳細を取得する のステップ 2 からの SubnetIds での VPC です。
サブネット	ステップ 2: OpsWorks for Puppet Enterprise サーバーの詳細を取得する のステップ 2 からの SubnetIds です。
既存のセキュリティグループを選択します -> 共通のセキュリティグループ	ステップ 2: OpsWorks for Puppet Enterprise サーバーの詳細を取得する のステップ 2 からの SecurityGroupIds です。
[Storage (ストレージ)]	少なくとも 120 GB。
IAM インスタンスプロファイル	ステップ 2: OpsWorks for Puppet Enterprise サーバーの詳細を取得する のステップ 2 からの InstanceProfileArn です。

Elastic IP を作成して新しいインスタンスにアタッチする場合は、新しいインスタンスのインスタンス ID をコピーして、[\(オプション\) ステップ 4.1: Elastic IP を作成してアタッチしてください](#) のステップを完了します。

(オプション) ステップ 4.1: Elastic IP を作成してアタッチしてください

Elastic IP アドレスを使用すると、アドレスをアカウント内の別のインスタンスに迅速に再マッピングすることで、インスタンスやソフトウェアの障害をマスクできます。

Elastic IP アドレスを作成して関連付けるには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/ec2/> で Amazon EC2 コンソールを開きます。

2. 伸縮性 IPs を選択してください。
3. [Elastic IP アドレスを割り当てる] をクリックします。
4. [Elastic IP アドレスの割り当て] ページから、割り当てる を選択します。パブリック IPv4 アドレスが作成されます。
5. 割り当てられた IPv4 アドレス をコピーします。
6. アクション で、Elastic IP アドレスの関連付け を選択します。
7. たとえば、新規インスタンスのインスタンス ID を入力します。
8. [関連付ける] を選択します。

ステップ 5: 新規 EC2 インスタンスに Puppet Enterprise をインストールしてください

SSH を使用して 新規 EC2 インスタンスに接続します。EC2 コンソールでは SSH の代わりにセッションマネージャーを使用できます。

```
# switch to sudo user
sudo -i

# Setup environment variables
PUPPET_ENTERPRISE_VERSION=Puppet Enterprise version from step 2.3
hostname Public IPv4 DNS or Custom Domain if available

# Install Puppet Enterprise
curl -JLO https://pm.puppetlabs.com/puppet-enterprise/$PUPPET_ENTERPRISE_VERSION/
puppet-enterprise-$PUPPET_ENTERPRISE_VERSION-el-7-x86_64.tar.gz
tar -xf puppet-enterprise-$PUPPET_ENTERPRISE_VERSION-el-7-x86_64.tar.gz

./puppet-enterprise-$PUPPET_ENTERPRISE_VERSION-el-7-x86_64/puppet-enterprise-installer
```

SSH または Systems Manager セッションは、次のステップのために開いたままにしておくことができます。

ステップ 6: 新しい EC2 インスタンスにバックアップを復元してください

```
# Setup environment variables
S3_BUCKET=S3 bucket name from step 2.1
PUPPET_BACKUP=Puppet backup file name from step 3.2
```

```
# download backup
aws s3 cp s3://$S3_BUCKET/tmp/puppet-backup/$PUPPET_BACKUP

# Prepare Puppet Enterprise backup to remove OpsWorks metadata
mkdir output
tar -xf $PUPPET_BACKUP -C output/
cd output/
rm -f opt/puppetlabs/facter/facts.d/opsworks.json
tar -cf ../$PUPPET_BACKUP *
cd ..
rm -rf output/

# Restore from backup
PATH=$PATH:/opt/puppetlabs/puppet/bin/
puppet-backup restore $PUPPET_BACKUP
puppet agent -t
```

復元された EC2 インスタンスの Puppet コンソールには、インスタンスの `https://##### IPv4` でアクセスできます。パブリック IPv4 DNS は、EC2 コンソールのインスタンスの詳細ページで確認できます。ログイン認証情報は、OpsWorks for Puppet Enterprise サーバーへのアクセスに使用する認証情報と同じです。

SSH または Systems Manager セッションは、次のステップのために開いたままにしておくことができます。

ステップ 7: Puppet ライセンスを設定してください

[Puppet ウェブサイト](#) の手順に従ってライセンスを設定します。

SSH または Systems Manager セッションは、次のステップのために開いたままにしておくことができます。

ステップ 8: ノードを移行してください

OpsWorks for Puppet Enterprise サーバーでサポートされているドメインには、次の 2 種類があります。

- BYODC (Bring Your Own Domain and Certificate)
- OpsWorks エンドポイント

ステップ 8.1: BYODC 用 (独自のドメインと証明書の使用)

これらのノードでは、DNS プロバイダーのカスタムドメインを新しい EC2 インスタンスのパブリック IPv4 DNS またはパブリック IPv4 アドレスにポイントするだけで済みます。

ステップ 8.2: OpsWorks エンドポイントの場合

OpsWorks エンドポイントの場合、Puppet ドキュメントでは、ノードで Puppet エージェントを [アンインストール](#) し、新しく復元された Puppet Enterprise サーバーを使用して Puppet エージェントを [インストールする](#) ことを推奨しています。

Note

Puppet にはエージェントノードを移動する自動手順はありませんが、自動ノード移行を実現するために Puppet コミュニティメンバーが [Puppet Forge ウェブサイト](#) に公開しているモジュールがいくつかあります。これらのモジュールには、[pe_migrate](#) モジュールと、別の作者による二つ目の [移行モジュール](#) が含まれます。Puppet Forge ウェブサイトのモジュールは、Puppet または ではサポートされていません。ただし、Forge モジュール内で明示的に記載されている OpsWorks 場合を除きます。これらのモジュールは注意して使用し、広く使用する前にテストすることをおすすめします。

以下のセクションでは、Linux インスタンスで Puppet エージェントをアンインストールして再インストールする手順を説明します。

トピック

- [ステップ 8.2.1 : Puppet サーバーからアンインストーラーをコピーしてください](#)
- [ステップ 8.2.2: アンインストーラーをダウンロードしてノードで実行してください](#)
- [ステップ 8.2.3 : Puppet エージェントをノードに再インストールしてください](#)

ステップ 8.2.1 : Puppet サーバーからアンインストーラーをコピーしてください

エージェントをアンインストールする前に、ノードの IAM インスタンスプロファイルに S3 ReadOnly 権限があることを確認してください。

次のコマンドを実行して、アンインストーラーを Puppet サーバーから S3 バケットにコピーします。

```
aws s3 cp \  
  /opt/puppetlabs/bin/puppet-enterprise-uninstaller \  
  s3://$S3_BUCKET/tmp/puppet-enterprise-uninstaller
```

コマンドを実行したら、Puppet サーバーの SSH または Session Manager セッションからログアウトできます。

ステップ 8.2.2: アンインストーラーをダウンロードしてノードで実行してください

SSH を使用してノードに接続します。ノードが EC2 インスタンスの場合は、SSH の代わりに EC2 コンソールの Session Manager を使用できます。

```
sudo -i  
  
S3_BUCKET=aws-opsworks-cm-abcdefg-uuhtyn6messn  
aws s3 cp s3://$S3_BUCKET/tmp/puppet-enterprise-uninstaller /opt/puppetlabs/bin/  
chmod 700 /opt/puppetlabs/bin/puppet-enterprise-uninstaller  
/opt/puppetlabs/bin/puppet-enterprise-uninstaller
```

SSH または Systems Manager セッションは、次のステップのために開いたままにしておくことができます。

ステップ 8.2.3 : Puppet エージェントをノードに再インストールしてください

以下の手順を実行して、Puppet エージェントをノードに再インストールします。

トピック

- [ステップ 8.2.3.1 : Puppet エージェントを正しい設定でインストールしてください](#)
- [ステップ 8.2.3.2 : Puppet コンソールで証明書を承認してください](#)
- [ステップ 8.2.3.3: ノードを Puppet Enterprise サーバーにチェックインしてください](#)

ステップ 8.2.3.1 : Puppet エージェントを正しい設定でインストールしてください

以下のコマンドを実行して、Puppet エージェントをインストールします。

```
curl -k https://Public_IPv4_DNS:8140/packages/current/install.bash | bash
```

ステップ 8.2.2.3 までは SSH または Session Manager セッションを開いたままにしておくことができます。

ステップ 8.2.3.2 : Puppet コンソールで証明書を承認してください

1. https://Public_IPv4_DNS の Puppet サーバーのコンソールに移動します。
2. 「証明書」を選択し、「署名されていない証明書」を選択します。
3. 「承認」を選択して Puppet エージェントの証明書に署名します。

ステップ 8.2.3.3: ノードを Puppet Enterprise サーバーにチェックインしてください

ノード上で以下のコマンドを実行して、サーバーにチェックインします。

```
puppet agent -t
```

これで、ノードが Puppet サーバーのコンソールに表示されるはずです。

ステップ 9: OpsWorks for Puppet Enterprise サーバーを削除する

OpsWorks for Puppet Enterprise サーバーを削除するには AWS CLI、OpsWorks コンソールまたはを使用できます。

OpsWorks コンソールを使用してサーバーを削除するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/opsworks/> で AWS OpsWorks コンソールを開きます。
2. ナビゲーションペインで、[Puppet Enterprise サーバー] を選択します。
3. [Puppet Enterprise サーバー] ページで、削除するサーバーを選択します。
4. [アクション]から[Puppet Enterprise サーバーの削除] を選択します。

を使用してサーバーを削除するには AWS CLI

以下のコマンドを実行します。

```
aws opsworks-cm delete-server \  
  --server-name server-name \  
  --region region
```


OpsWorks を使用した Puppet Enterprise API コールのログ記録 AWS CloudTrail

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

OpsWorks for Puppet Enterprise は、と統合されています。これは AWS CloudTrail、ユーザー、ロール、または OpsWorks for Puppet Enterprise の AWS のサービスによって実行されたアクションを記録するサービスです。は、OpsWorks for Puppet Enterprise コンソールからの呼び出しと OpsWorks for Puppet Enterprise API OpsWorks へのコード呼び出しを含む、for Puppet Enterprise のすべての API 呼び出しをイベントとして CloudTrail キャプチャします。APIs 証跡を作成する場合は、OpsWorks for Puppet Enterprise の CloudTrail イベントなど、Amazon S3 バケットへのイベントの継続的な配信を有効にすることができます。Amazon S3 証跡を設定しない場合でも、CloudTrail コンソールのイベント履歴 で最新のイベントを表示できます。で収集された情報を使用して CloudTrail、for Puppet Enterprise OpsWorks に対して行われたリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

の詳細については CloudTrail、「[AWS CloudTrail ユーザーガイド](#)」を参照してください。

OpsWorks の Puppet エンタープライズ情報用 CloudTrail

CloudTrail AWS アカウントを作成すると、がアカウントで有効になります。OpsWorks for Puppet Enterprise でアクティビティが発生すると、そのアクティビティは CloudTrail イベント履歴 の他の AWS サービスイベントとともにイベントに記録されます。AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「[イベント履歴を使用した CloudTrail イベントの表示](#)」を参照してください。

for Puppet Enterprise のイベントなど、AWS アカウント内のイベントの継続的な記録 OpsWorks については、証跡を作成します。証跡により、はログファイル CloudTrail を Amazon S3 バケットに配信できます。デフォルトでは、コンソールで追跡を作成するときに、追跡がすべてのリージョンに適用されます。証跡は、AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集さ

れたイベントデータをより詳細に分析し、それに基づいて行動するように、他の AWS サービスを設定できます。詳細については、以下をご覧ください。

- [証跡を作成するための概要](#)
- [CloudTrail サポートされているサービスと統合](#)
- [の Amazon SNS 通知の設定 CloudTrail](#)
- [複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからの CloudTrail ログファイルの受信](#)

すべての OpsWorks for Puppet Enterprise アクションは、[によってログに記録 CloudTrail](#) され、[OpsWorks for Puppet Enterprise API リファレンス](#) に記載されています。例えば、[CreateServer](#)、および [DescribeServers](#) アクションを呼び出すと [CreateBackup](#)、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。同一性情報は次の判断に役立ちます。

- リクエストが、ルートと IAM ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーションユーザーの一時的なセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、[CloudTrail userIdentity Element](#) を参照してください。

Puppet Enterprise ログファイルエントリ OpsWorks の理解

証跡は、指定した Amazon S3 バケットにイベントをログファイルとして配信できるようにする設定です。CloudTrail ログファイルには 1 つ以上のログエントリが含まれます。イベントは任意のソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。CloudTrail ログファイルはパブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次の例は、for Puppet Enterprise CreateServer アクション OpsWorks の CloudTrail ログエントリを示しています。

```
{"eventVersion": "1.05",
 "userIdentity": {
   "type": "AssumedRole",
```

```
"principalId":"ID number:OpsWorksCMUser",
"arn":"arn:aws:sts::831000000000:assumed-role/Admin/OpsWorksCMUser",
"accountId":"831000000000","accessKeyId":"ID number",
"sessionContext":{
  "attributes":{
    "mfaAuthenticated":"false",
    "creationDate":"2017-01-05T22:03:47Z"
  },
  "sessionIssuer":{
    "type":"Role",
    "principalId":"ID number",
    "arn":"arn:aws:iam::831000000000:role/Admin",
    "accountId":"831000000000",
    "userName":"Admin"
  }
},
"eventTime":"2017-01-05T22:18:23Z",
"eventSource":"opsworks-cm.amazonaws.com",
"eventName":"CreateServer",
"awsRegion":"us-west-2",
"sourceIPAddress":"101.25.190.51",
"userAgent":"console.amazonaws.com",
"requestParameters":{
  "serverName":"test-puppet-server",
  "engineModel":"Single",
  "engine":"Puppet",
  "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-cm-ec2-role",
  "backupRetentionCount":3,"serviceRoleArn":"arn:aws:iam::831000000000:role/service-role/aws-opsworks-cm-service-role",
  "engineVersion":"12",
  "preferredMaintenanceWindow":"Fri:21:00",
  "instanceType":"t2.medium",
  "subnetIds":["subnet-1e111f11"],
  "preferredBackupWindow":"Wed:08:00"
},
"responseElements":{
  "server":{
    "endpoint":"test-puppet-server-xxxx8u4390xo6pd9.us-west-2.opsworks-cm.io",
    "createdAt":"Jan 5, 2017 10:18:22 PM",
    "serviceRoleArn":"arn:aws:iam::831000000000:role/service-role/aws-opsworks-cm-service-role",
    "preferredBackupWindow":"Wed:08:00",
```

```
    "status": "CREATING",
    "subnetIds": ["subnet-1e111f11"],
    "engine": "Puppet",
    "instanceType": "t2.medium",
    "serverName": "test-puppet-server",
    "serverArn": "arn:aws:opsworks-cm:us-west-2:831000000000:server/test-puppet-server/8ezz7f6z-e91f-4z10-89z5-8c6219zzz09f",
    "engineModel": "Single",
    "backupRetentionCount": 3,
    "engineAttributes": [
      { "name": "PUPPET_ADMIN_PASSWORD", "value": "**** Redacted ****" },
      { "name": "PUPPET_API_CA_CERT", "value": "**** Redacted ****" },
    ],
    "engineVersion": "12.11.1",
    "instanceProfileArn": "arn:aws:iam::831000000000:instance-profile/aws-opsworks-cm-ec2-role",
    "preferredMaintenanceWindow": "Fri:21:00"
  }
},
"requestID": "de7z64z9-d394-12ug-8081-7zz0386fbc6",
"eventID": "8z7z18dz-6z90-47bz-87cf-e8346428zzz3",
"eventType": "AwsApiCall",
"recipientAccountId": "831000000000"
}
```

Puppet Enterprise OpsWorks のトラブルシューティング

Important

この AWS OpsWorks for Puppet Enterprise サービスは 2024 年 3 月 31 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このトピックには、Puppet Enterprise OpsWorks の問題に関する一般的ないくつかの解決策と、それらの問題に対する推奨される解決策が含まれています。

トピック

- [一般的なトラブルシューティングのヒント](#)
- [特定のエラーのトラブルシューティング](#)
- [その他のヘルプとサポート](#)

一般的なトラブルシューティングのヒント

Puppet マスターを作成または操作できない場合は、エラーメッセージやログを表示すると、問題のトラブルシューティングに役立ちます。以下のタスクでは、Puppet マスターに関する問題のトラブルシューティングを行うときの一般的な開始点を示します。特定のエラーと解決方法については、このトピックの「[特定のエラーのトラブルシューティング](#)」セクションを参照してください。

- OpsWorks Puppet マスターの起動に失敗した場合のエラーメッセージを表示するには、for Puppet Enterprise コンソールを使用します。Puppet マスターのプロパティページで、サーバーの起動と実行に関連するエラーメッセージが、ページの一番上に表示されます。エラーは、Puppet Enterprise OpsWorks の場合は AWS CloudFormation、または Puppet マスターの作成に使用される Amazon EC2 のサービスから発生する可能性があります。プロパティページでは、実行中のサーバーで発生するイベントを表示することもできます。これには、障害イベントメッセージが含まれる場合があります。
- EC2 に関する問題を解決するため、SSH を使用してサーバーのインスタンスに接続してログを表示します。EC2 インスタンスのログは /var/log/aws/opsworks-cm ディレクトリに保存されています。これらのログは OpsWorks、for Puppet Enterprise が Puppet マスターを起動している間にコマンド出力をキャプチャします。

特定のエラーのトラブルシューティング

トピック

- [サーバーの状態は \[接続が失われました\]](#)
- [サーバーの作成が「リクエストされた設定は現在サポートされていません」というメッセージで失敗する](#)
- [サーバーの Amazon EC2 インスタンスが作成できない](#)
- [サービスロールのエラーによりサーバーを作成できない](#)
- [Elastic IP アドレスの制限を超えた](#)
- [ノードの自動関連付けに失敗する](#)
- [システムメンテナンスの失敗](#)

サーバーの状態は [接続が失われました]

問題: サーバーのステータスに[接続が失われました]と表示される。

原因: これは、以外のエンティティ AWS OpsWorks が OpsWorks for Puppet Enterprise サーバーまたはそのサポートリソースに変更を加えた場合に最もよく発生します。は、接続が失われた状態の Puppet Enterprise サーバーに接続して、バックアップの作成、オペレーティングシステムパッチの適用、Puppet の更新などのメンテナンスタスクを処理する AWS OpsWorks ことはできません。その結果、サーバーが重要なアップデートを見逃したり、セキュリティ上の問題の影響を受けやすくなったり、またはその他の理由で期待どおりに動作しない可能性があります。

解決策: 次の手順を試して、サーバーの接続を復元してください。

1. サービスロールに必要な権限がすべてあることを確認してください。
 - a. サーバーの「設定」ページの「ネットワークとセキュリティ」で、サーバーが使用しているサービスロールへのリンクを選択します。これにより、IAM コンソールに表示されるサービスロールが開きます。
 - b. 「アクセス許可」のタブで、AWSOpsWorksCMServiceRole が「アクセス許可ポリシー」リストに含まれていることを確認します。リストにない場合は、AWSOpsWorksCMServiceRole 管理ポリシーをロールに手動で追加してください。
 - c. 「信頼関係」のタブで、ユーザーに代わって役割を引き受ける opsworks-cm.amazonaws.com サービスを信頼する信頼ポリシーがサービスロールに含まれていることを確認します。ロールで信頼ポリシーを使用する方法の詳細については、[「ロールの変更 \(コンソール\)」](#) または AWS 「セキュリティブログ記事」の [「IAM ロールで信頼ポリシーを使用する方法」](#) を参照してください。
2. インスタンスプロファイルに必要な権限がすべてあることを確認してください。
 - a. サーバーの「設定」ページの「ネットワークとセキュリティ」で、サーバーが使用しているインスタンスプロファイルへのリンクを選択します。これにより、IAM コンソールに表示されるインスタンスプロファイルが開きます。
 - b. 「権限」のタブで、AmazonEC2RoleforSSM と AWSOpsWorksCMInstanceProfileRole が両方とも「アクセス権限ポリシー」リストに含まれていることを確認します。一方または両方がリストにない場合は、これらの管理ポリシーを手動でロールに追加してください。
 - c. 「信頼関係」のタブで、ユーザーに代わって役割を引き受ける ec2.amazonaws.com サービスを信頼する信頼ポリシーがサービスロールに含まれていることを確認します。ロールで信頼ポリシーを使用する方法の詳細については、[「ロールの変更 \(コンソール\)」](#) または

AWS 「セキュリティブログ記事」の「[IAM ロールで信頼ポリシーを使用する方法](#)」を参照してください。

3. Amazon EC2 コンソールで、Puppet Enterprise OpsWorks サーバーのリージョンと同じリージョンにいることを確認し、サーバーが使用している EC2 インスタンスを再起動します。
 - a. `aws-opsworks-cm-instance-server-name` という名前の EC2 インスタンスを選択します。
 - b. [インスタンスの状態]メニューで、[インスタンスの再起動] を選択します。
 - c. サーバーが再起動して完全にオンラインになるまで最大 15 分かかります。
4. OpsWorks for Puppet Enterprise コンソールのサーバーの詳細ページで、サーバーのステータスが正常になったことを確認します。

上記の手順を実行してもサーバーステータスが [接続が失われました] のままの場合は、次のいずれかを試してください。

- [新しいサーバーを作成し、元のサーバーを削除して](#)、サーバーを交換してください。現在のサーバー上のデータが重要な場合は、[最新のバックアップからサーバーを復元し、元の応答しないサーバーを削除する前に](#)データが最新であることを確認してください。
- [AWS Support](#)にお問い合わせください。

サーバーの作成が「リクエストされた設定は現在サポートされていません」というメッセージで失敗する

問題: Puppet Enterprise サーバーの作成を試みているが、「リクエストされた設定は現在サポートされていません。サポートされている設定についてドキュメントをご確認ください」のようなエラーメッセージでサーバーの作成が失敗する。

原因: Puppet マスターに対してサポートされていないインスタンスタイプが指定された可能性があります。[ハードウェア専用インスタンス](#)用など、デフォルトでないテナンシーを持つ VPC で Puppet サーバーの作成を選択した場合、指定された VPC 内のすべてのインスタンスも、専用テナンシーまたはホストテナンシーのインスタンスである必要があります。t2 など一部のインスタンスタイプはデフォルトテナンシーでしか使用できないため、Puppet マスターインスタンスタイプは指定された VPC でサポート可能でない場合があります、そのためにサーバーの作成が失敗します。

解決策: デフォルト以外のテナンシーを持つ VPC を選択した場合は、専用テナンシーをサポートできる m4 インスタンスタイプを使用します。

サーバーの Amazon EC2 インスタンスが作成できない

問題: サーバーの作成が、次のようなエラーメッセージにより失敗する。「The following resource(s) failed to create: [EC2Instance]. Failed to receive 1 resource signal(s) within the specified duration」

原因: この問題のほとんどが、EC2 インスタンスでネットワークアクセスが可能でないことが原因です。

解決策: インスタンスにアウトバウンドインターネットアクセスがあり、AWS サービスエージェントがコマンドを発行できることを確認します。VPC (単一のパブリックサブネットを持つ VPC) で DNS 解決が有効になっていて、サブネットで [パブリック IP の自動割り当て] 設定が有効になっていることを確認します。

サービスロールのエラーによりサーバーを作成できない

問題: サーバーの作成が失敗し、「sts: を実行する権限がありません」というエラーメッセージが表示される AssumeRole。

原因: お使いのサービスロールに新しいサーバーを作成するための適切なアクセス権限がない場合に、この問題が発生する可能性があります。

解決策: OpsWorks for Puppet Enterprise コンソールを開きます。コンソールを使用して、新しいサービスロールとインスタンスプロファイルロールを生成します。独自のサービスロールを使用する場合は、AWSOpsWorksCMServiceRole ポリシーをロールにアタッチします。opsworks-cm.amazonaws.com がロールの信頼関係のサービス間でリストされていることを確認します。Puppet マスターに関連付けられているサービスロールに AWSOpsWorksCMServiceRole 管理ポリシーがアタッチされていることを確認します。

Elastic IP アドレスの制限を超えた

問題: 「The following resource(s) failed to create: [EIP, EC2Instance]. Resource creation cancelled, the maximum number of addresses has been reached.」という状態を示すエラーメッセージにより、サーバーを作成できない。

原因: アカウントで Elastic IP (EIP) アドレスの最大数を使用した場合に、この問題が発生します。EIP アドレスのデフォルトの制限は 5 です。

解決策: 既存の EIP アドレスを解放するか、アカウントがアクティブに使用していないアドレスを削除できます。または、AWS カスタマーサポートに連絡して、アカウントに関連付けられている EIP アドレスの制限を増やすことができます。

ノードの自動関連付けに失敗する

問題: 新しい Amazon EC2 ノードの自動 (無人) 関連付けに失敗する。Puppet マスターに追加されたノードが Puppet Enterprise ダッシュボードに表示されない。

原因: opsworks-cm API コールで新しい EC2 インスタンスとの通信を許可するインスタンスプロファイルとして IAM ロールをセットアップしていない場合に、この問題が発生する可能性があります。

解決策: [OpsWorks for Puppet Enterprise でノードを自動的に追加する](#) で説明したように、EC2 インスタンスのプロファイルにポリシーをアタッチして、AssociateNode と DescribeNodeAssociationStatus の API コールを EC2 と連携できるようにします。

システムメンテナンスの失敗

AWS OpsWorks CM は毎週システムメンテナンスを実行し、セキュリティ更新プログラムを含むで AWS テストされた最新バージョンの Puppet Server が、常に OpsWorks for Puppet Enterprise サーバーで実行されていることを確認します。何らかの理由でシステムメンテナンスが失敗した場合、は失敗 AWS OpsWorks CM を通知します。システムメンテナンスの詳細については、「[Puppet Enterprise OpsWorks のでのシステムメンテナンス](#)」を参照してください。

このセクションでは、考えられる障害の原因を説明し、解決策を提案します。

トピック

- [サービスロールまたはインスタンスプロファイルのエラーによって、システムのメンテナンスが妨げられる](#)

サービスロールまたはインスタンスプロファイルのエラーによって、システムのメンテナンスが妨げられる

問題: システムメンテナンスが失敗し、「sts: を実行する権限がありません」というエラーメッセージ AssumeRole、またはアクセス許可に関する同様のエラーメッセージが表示される。

原因: これは、使用しているサービスロールまたはインスタンスプロファイルのいずれかに、サーバー上でシステムメンテナンスを実行するための適切な権限がない場合に発生する可能性があります。

解決策: サービスロールとインスタンスプロファイルに必要なすべての権限があることを確認してください。

1. サービスロールに必要な権限がすべてあることを確認してください。
 - a. サーバーの「設定」ページの「ネットワークとセキュリティ」で、サーバーが使用しているサービスロールへのリンクを選択します。これにより、IAM コンソールに表示されるサービスロールが開きます。
 - b. 「アクセス許可」のタブで、AWSOpsWorksCMServiceRole がサービスロールにアタッチされていることを確認します。AWSOpsWorksCMServiceRole がリストにない場合は、このポリシーをロールに追加します。
 - c. opsworks-cm.amazonaws.com がロールの信頼関係のサービス間でリストされていることを確認します。ロールで信頼ポリシーを使用する方法の詳細については、[「ロールの変更 \(コンソール\)」](#) または AWS 「セキュリティブログ記事」の [「IAM ロールで信頼ポリシーを使用する方法」](#) を参照してください。
2. インスタンスプロファイルに必要な権限がすべてあることを確認してください。
 - a. サーバーの「設定」ページの「ネットワークとセキュリティ」で、サーバーが使用しているインスタンスプロファイルへのリンクを選択します。これにより、IAM コンソールに表示されるインスタンスプロファイルが開きます。
 - b. 「権限」のタブで、AmazonEC2RoleforSSM と AWSOpsWorksCMInstanceProfileRole が両方とも「アクセス権限ポリシー」リストに含まれていることを確認します。一方または両方がリストにない場合は、これらの管理ポリシーを手動でロールに追加してください。
 - c. 「信頼関係」のタブで、ユーザーに代わって役割を引き受ける ec2.amazonaws.com サービスを信頼する信頼ポリシーがサービスロールに含まれていることを確認します。ロールで信頼ポリシーを使用する方法の詳細については、[「ロールの変更 \(コンソール\)」](#) または AWS 「セキュリティブログ記事」の [「IAM ロールで信頼ポリシーを使用する方法」](#) を参照してください。

その他のヘルプとサポート

発生している特定の問題がこのトピックで説明されていないか、このトピックの提案を試しても問題が解決しない場合は、[AWS OpsWorks フォーラム](#)を参照してください。

また、[AWS Support Center](#) を参照することもできます。AWS サポートセンターは、AWS サポートケースを作成および管理するためのハブです。AWS サポートセンターには、フォーラム、技術上のFAQs、サービスのヘルスステータス、など、その他の役立つリソースへのリンクも含まれています。AWS Trusted Advisor。

Chef Automate OpsWorks の AWS

⚠ Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks for Chef Automate では、AWS で [Chef Automate](#) サーバーを実行できます。Chef サーバーは数分でプロビジョニングでき、[ガオペレーション](#)、バックアップ、復元、ソフトウェアアップグレード AWS OpsWorks for Chef Automate を処理できます。を使用すると、Chef サーバーを管理する代わりに、コア設定管理タスクに集中 AWS OpsWorks for Chef Automate できます。

Chef Automate サーバーは、ノードで実行する [chef-client](#) Chef レシピを指示し、ノードに関する情報を保存し、Chef クックブックの中央リポジトリとして機能することで、環境内のノードの設定を管理します。AWS OpsWorks for Chef Automate は、Chef Automate のプレミアム機能を含む Chef サーバーを提供します: Chef Infra と Chef InSpec。

AWS OpsWorks for Chef Automate サーバーは Amazon Elastic Compute Cloud インスタンスで実行されます。AWS OpsWorks for Chef Automate サーバーは、最新バージョンの Amazon Linux (Amazon Linux 2) を実行するように設定されています。Chef Automate のこのバージョンの変更については、「[Chef Automate リリースノート](#)」を参照してください。次の表は、AWS OpsWorks for Chef Automate サーバーにインストールされる Chef コンポーネントを示しています。

コンポーネント名	説明	AWS OpsWorks for Chef Automate サーバーにインストールされているバージョン
Chef Automate	Chef Automate は、継続的デプロイのための自動化されたワークフローを提供し、管理対象ノードに関する洞察をウェブベースのマネジメントコンソールに表示する、エンタープライズサーバーソフト	2.0

コンポーネント名	説明	AWS OpsWorks for Chef Automate サーバーにインストールされているバージョン
	<p>ウェアパッケージです。Chef Automate は、Chef Infra を含むインフラストラクチャの自動化、Chef を含むセキュリティおよびコンプライアンス情報と強制 InSpec、Chef Habitat を含む自動デプロイを提供します。</p> <p>Chef Automate の詳細については、Chef のウェブサイトの「Chef Automate」を参照してください。</p>	
Chef Infra	<p>Chef Infra Server (旧称 Chef Server) は、Chef Infra クライアント (chef-client) エージェントを使用して、管理対象ノードに設定を継続的に適用することで、目的の状態を維持します。</p> <p>Infra の詳細については、Chef のウェブサイトの「Chef Infra」を参照してください。</p>	12.x

コンポーネント名	説明	AWS OpsWorks for Chef Automate サーバーにインストールされているバージョン
シェフ InSpec	<p>Chef は、ソフトウェアエンジニア、オペレーション、セキュリティエンジニア間で共有できるセキュリティとコンプライアンスのルール InSpec について説明します。コンプライアンス、セキュリティ、その他のポリシー要件により、chef-client エージェントが管理対象ノードに対して実行できる自動テストのフレームワークが形成され、標準の一貫した実施が確実になります。</p> <p>の詳細については InSpec、Chef ウェブサイトの Chef InSpec を参照してください。</p>	3.9.0

AWS OpsWorks for Chef Automate サーバーに関連付けられているノードでサポートされている chef-client の最小バージョンは 13.x です。少なくとも 14.10.9、または最新の安定 chef-client バージョンを実行することをお勧めします。

Chef ソフトウェアの新しいマイナーバージョンが使用可能になった場合、システムメンテナンスでは、それらが AWS のテストをパスし次第、サーバー上の Chef Automate および Chef Server のマイナーバージョンを自動的に更新するように設計されています。AWS は広範なテストを実行して、Chef のアップグレードが本番環境に対応しており、既存の顧客環境を中断しないことを確認します。そのため、Chef ソフトウェアリリースから Chef Automate サーバー用の既存の OpsWorks へのアプリケーションへの適用に遅延が生じる可能性があります。また、システムメンテナンスでは、ご使用のサーバーを Amazon Linux の最新バージョンにアップグレードします。

サポートされているオペレーティングシステムを実行しており、AWS OpsWorks for Chef Automate サーバーへのネットワークアクセスを持つオンプレミスのコンピュータまたは EC2 インスタンスに接続できます。お客様が管理するノードに対応するオペレーティングシステムの一覧は、[Chef のウェブサイト](#)を参照してください。[chef-client](#) エージェントソフトウェアは Chef サーバーで管理したいノードにインストールされます。

トピック

- [のリージョンサポート AWS OpsWorks for Chef Automate](#)
- [AWS OpsWorks Chef Automate のサポート終了に関するFAQs](#)
- [AWS OpsWorks for Chef Automate サーバーを Chef Automate 2 にアップグレードする](#)
- [の開始方法 AWS OpsWorks for Chef Automate](#)
- [を使用して AWS OpsWorks for Chef Automate サーバーを作成する AWS CloudFormation](#)
- [カスタムドメインを使用するように AWS OpsWorks for Chef Automate サーバーを更新する](#)
- [AWS OpsWorks for Chef Automate サーバーのスターターキットを再生成する](#)
- [AWS OpsWorks for Chef Automate リソースでのタグの使用](#)
- [AWS OpsWorks for Chef Automate サーバーのバックアップと復元](#)
- [でのシステムメンテナンス AWS OpsWorks for Chef Automate](#)
- [でのコンプライアンススキャン AWS OpsWorks for Chef Automate](#)
- [AWS OpsWorks for Chef Automate サーバーからノードの関連付けを解除する](#)
- [AWS OpsWorks for Chef Automate サーバーを削除する](#)
- [Chef Automate ダッシュボードの認証情報のリセット](#)
- [を使用した AWS OpsWorks for Chef Automate API コールのログ記録 AWS CloudTrail](#)
- [トラブルシューティング AWS OpsWorks for Chef Automate](#)

のリージョンサポート AWS OpsWorks for Chef Automate

次のリージョンエンドポイントは AWS OpsWorks for Chef Automate、server をサポートしています。は、インスタンスプロファイル、ユーザー、サービスロールなど、Chef サーバーに関連付けられたリソースを、Chef サーバーと同じリージョンエンドポイントに AWS OpsWorks for Chef Automate 作成します。Chef サーバーは VPC 内にある必要があります。作成するかすでに作成してある VPC を使用することも、デフォルトの VPC を使用することもできます。

- 米国東部 (オハイオ) リージョン

- 米国東部(バージニア州北部) リージョン
- US West (N. California) Region
- 米国西部 (オレゴン) リージョン
- Asia Pacific (Tokyo) Region
- アジアパシフィック (シンガポール) リージョン
- アジアパシフィック (シドニー) リージョン
- 欧州 (フランクフルト) リージョン
- 欧州 (アイルランド) リージョン

AWS OpsWorks Chef Automate のサポート終了に関するFAQs

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。

トピック

- [このサポート終了により、既存のユーザーはどのような影響を受けますか？](#)
- [何も操作を実行しない場合サーバーはどうなりますか？](#)
- [どのような代替案に移行できますか？](#)
- [このサービスはまだ新規顧客を受け入れていますか？](#)
- [サポート終了はすべての AWS リージョン に同時に影響しますか？](#)
- [どのレベルのテクニカルサポートが受けられますか？](#)
- [Chef Automate OpsWorks の現在の顧客であり、以前サービスを使用していなかったアカウントでサーバーを起動する必要があります。これはできますか？](#)
- [来年に主要な特徴量のリリースは行われますか？](#)

このサポート終了により、既存のユーザーはどのような影響を受けますか？

既存のお客様は、Chef Automate OpsWorks のサポート終了日である 2024 年 5 月 5 日まで影響を受けません。サポート終了日を過ぎると、お客様は OpsWorks コンソールまたは API を使用してサーバーを管理できなくなります。

何も操作を実行しない場合サーバーはどうなりますか？

2024 年 5 月 5 日以降、OpsWorks コンソールまたは API を使用してサーバーを管理することはできなくなります。その時点で、バックアップやメンテナンスなど、サーバーの継続的な管理機能の実行は停止されます。お客様への影響を抑えるため、Chef Automate サーバーをバックアップしている EC2 インスタンスは実行したままにしておきますが、Chef Automate と OpsWorks Chef のサービス契約で使用がカバー (または請求) されなくなるため、ライセンスは無効になります。[Chef](#) に連絡して新しいライセンスを取得する必要があります。Chef に連絡するときは、Chef Automate のお客様の既存の OpsWorks であり、から移行していることを必ず伝えてください OpsWorks。

どのような代替案に移行できますか？

AWS と Progress Chef では、フルマネージド型の Chef Automate サービスのメリットを引き続き享受できるように、新しい Chef SaaS サービスに移行することをお勧めします。Chef SaaS を使い始めるには、[Chef](#) に連絡して Chef SaaS アカウントの設定方法やデータとノードの移行方法に関するドキュメントを入手してください。

管理する AWS アカウントの EC2 インスタンスで Chef Automate を実行するために Chef SaaS がニーズを満たさない場合、Chef は、Bring [AWS Marketplace Your Own License \(BYOL\) モデル](#) や EC2 でのセルフホストなど、複数のオプションを提供します。このような移行を実行する方法の詳細については、[Progress Chef](#) にお問い合わせください。

このサービスはまだ新規顧客を受け入れていますか？

いいえ。AWS OpsWorks Chef Automate は新規顧客を受け入れなくなりました。

サポート終了はすべての AWS リージョンに同時に影響しますか？

はい。API とコンソールは 2024 年 5 月 5 日をもってサポート終了となり、すべての AWS リージョンを使用できなくなります。Chef Automate AWS リージョンが利用可能なについては、[AWS 「リージョンサービスリスト AWS OpsWorks」](#) を参照してください。

どのレベルのテクニカルサポートが受けられますか？

AWS は、お客様がサポート終了日まで現在持っているのと同じレベルのサポートを OpsWorks Chef Automate に提供し続けます。ご質問やご不明点がございましたら、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。移行サポートについては、お客様が [Progress Chef](#) に連絡することをお勧めします。

Chef Automate OpsWorks の現在の顧客であり、以前サービスを使用していなかったアカウントでサーバーを起動する必要があります。これはできますか？

例外的な事情がない限り、一般的にはできません。特別な状況が発生した場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームに連絡し、その詳細と根拠を伝えて、リクエストを確認します。

来年に主要な特徴量のリリースは行われますか？

いいえ。サービスのサポート終了が近づいているため、新特徴量のリリースは行いません。しかし、サポート終了日までは、引き続きセキュリティの改善とサーバーの管理を予定どおりに行います。

AWS OpsWorks for Chef Automate サーバーを Chef Automate 2 にアップグレードする

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Chef Automate 2 にアップグレードするための前提条件

開始する前に、Chef Automate 2 で新しく追加される機能と、Chef Automate 2 でサポートされない機能を確認してください。Chef Automate 2 の新しい機能およびサポートされない機能については、Chef ウェブサイトにある [Chef Automate 2 のドキュメント](#) を参照してください。

Chef Automate 1 を実行しているサーバーがアップグレード対象となるには、2019 年 11 月 1 日以降に少なくとも 1 つのメンテナンスが正常に実行されている必要があります。

AWS OpsWorks for Chef Automate サーバーのメンテナンスオペレーションと同様に、サーバーはアップグレード中にオフラインになります。アップグレードプロセス中に最大 3 時間のダウンタイムを予定する必要があります。

Chef Automate ダッシュボードウェブサイト用として、このサーバーのサインイン認証情報が必要です。アップグレードが完了したら、Chef Automate ダッシュボードにサインインし、ノードおよび設定情報に変更されていないことを確認します。

Important

AWS OpsWorks for Chef Automate サーバーを Chef Automate 2 にアップグレードする準備ができたなら、ここでの手順のみを使用してアップグレードします。はバックアップの作成など、アップグレードプロセスの多く AWS OpsWorks for Chef Automate を自動化するため、Chef ウェブサイトのアップグレード指示に従わないでください。

アップグレードプロセスについて

アップグレードプロセスでは、アップグレードの開始前と終了後にサーバーがバックアップされます。次のバックアップが作成されます。

- Chef Automate 1 (バージョン 12.17.33) を実行しているサーバーのバックアップ。
- アップグレードの終了後に Chef Automate 2 (バージョン 2019-08) を実行しているサーバーのバックアップ。

アップグレードプロセスでは、サーバーで Chef Automate 1 を実行したときに使用していた Amazon EC2 インスタンスが終了します。Chef Automate 2 サーバーを実行するための新しいインスタンスが作成されます。

Chef Automate 2 にアップグレードする (コンソール)

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/opsworks/> で AWS OpsWorks コンソールを開きます。
2. 左側のナビゲーションペインで、[AWS OpsWorks for Chef Automate] を選択します。

3. サーバーを選択して、そのプロパティページを表示します。ページの上部にある青いバナーは、サーバーが Chef Automate 2 へのアップグレード対象であるかどうかを示します。

Note

Chef Automate 1 を実行しているサーバーがアップグレード対象となるには、2019 年 11 月 1 日以降に少なくとも 1 つのメンテナンスが正常に実行されている必要があります。

4. サーバーがアップグレード対象である場合は、[Start upgrade] を選択します。
5. アップグレードには最大 3 時間かかります。アップグレードプロセス中は、プロパティページにサーバーのステータスとして [Under maintenance] と表示されます。
6. アップグレードが完了すると、プロパティページにメッセージとして、[Successfully upgraded to Automate 2] と [Maintenance completed successfully] の 2 つが表示されます。サーバーのステータスは [HEALTHY] となります。
7. 既存の認証情報を使用して Chef Automate ダッシュボードにサインインし、ノードが正しくレポートすることを確認します。

Chef Automate 2 にアップグレードする (CLI)

1. (オプション) アップグレードの対象となる AWS OpsWorks for Chef Automate サーバーがない場合は、次のコマンドを実行します。デフォルトの AWS リージョンとは異なる AWS リージョンの AWS OpsWorks for Chef Automate サーバーを一覧表示する場合は、必ず `--region` パラメータを追加してください。

```
aws opsworks-cm describe-servers
```

表示された結果で、`true` 属性の値が `CHEF_MAJOR_UPGRADE_AVAILABLE` であるものを探します。これは、サーバーが Chef Automate 2 へのアップグレード対象であることを示します。アップグレードの対象となる AWS OpsWorks for Chef Automate サーバーの名前を書き留めます。

2. `server_name` をサーバーの名前に置き換えて、次のコマンドを実行します AWS OpsWorks for Chef Automate 。定期的なシステムメンテナンスを実行せずに Chef Automate 2 にアップグレードする場合は、コマンドに示すように `CHEF_MAJOR_UPGRADE` エンジン属性を追加します。ターゲットサーバーがデフォルトの AWS リージョンにない場合は、`--region` パラメータを追加します。1 つのコマンドでアップグレードできるサーバーは 1 つだけです。

```
aws opsworks-cm start-maintenance --server-name server_name --engine-attributes
Name=CHEF_MAJOR_UPGRADE,Value=true --region region
```

が何らかの理由でサーバーをアップグレード AWS OpsWorks for Chef Automate できない場合、このコマンドにより検証例外が発生します。

- アップグレードには最大 3 時間かかります。次のコマンドを実行すると、定期的にアップグレードのステータスを確認できます。

```
aws opsworks-cm describe-servers --server-name server_name
```

表示された結果で、Status 値を探します。Status が UNDER_MAINTENANCE である場合は、アップグレードがまだ進行中であることを示します。アップグレードに成功すると、次のようなメッセージが返されます。

```
2019/10/24 00:27:56 UTC      Successfully upgraded to Automate 2.
2019/10/23 23:50:38 UTC      Upgrading Chef server from Automate 1 to Automate
2
```

アップグレードが失敗した場合、はサーバー AWS OpsWorks for Chef Automate を自動的に Chef Automate 1 にロールバックします。

アップグレードには成功したが、サーバーがアップグレード前と同じように動作しない場合 (たとえば、マネージドノードがレポートしない場合) は、サーバーを手動でロールバックできます。手動のロールバックについては、「[AWS OpsWorks for Chef Automate サーバーを Chef Automate 1 にロールバックする \(CLI\)](#)」を参照してください。

AWS OpsWorks for Chef Automate サーバーを Chef Automate 1 にロールバックする (CLI)

アップグレードプロセスが失敗した場合、はサーバー AWS OpsWorks for Chef Automate を自動的に Chef Automate 1 にロールバックします。アップグレードは成功したが、サーバーがアップグレード前と同じように機能していない場合は、を使用して AWS OpsWorks for Chef Automate 手動で Chef Automate 1 にサーバーをロールバックできます AWS CLI。

1. 次のコマンドを実行して、アップグレードの試行前にサーバーで実行された最後のバックアップの BackupId を表示します。サーバーがデフォルトの AWS リージョンとは異なる AWS リージョンにある場合は、`--region` パラメータを追加します。

```
aws opsworks-cm describe-backups server_name
```

バックアップ IDs形式は `ServerName-yyyyMMddHHmmssSSS` です。表示された結果で、次の Chef Automate 1 のプロパティを探します。

```
"Engine": "Chef"  
"EngineVersion": "12.17.33"
```

2. ステップ 1 で返されたバックアップ ID を `--backup-id` の値として使用し、次のコマンドを実行します。

```
aws opsworks-cm restore-server --server-name server_name --backup-id ServerName-  
yyyyMMddHHmmssSSS
```

サーバーに保存したデータの量に応じて、サーバーの復元には 20 分～3 時間ほどかかります。復元中のサーバーのステータスは RESTORING になります。このステータスは、のサーバーのプロパティページに表示され AWS Management Console、`describe-servers` コマンドの結果に返されます。

3. 復元が完了すると、コンソールに [Restore completed successfully] というメッセージが表示されます。AWS OpsWorks for Chef Automate サーバーはオンラインであり、アップグレードプロセスを開始する前と同じです。

以下の資料も参照してください。

- [でのシステムメンテナンス AWS OpsWorks for Chef Automate](#)
- [バックアップから AWS OpsWorks for Chef Automate サーバーを復元する](#)
- 「[DescribeServers](#) API リファレンス」の「AWS OpsWorks」
- 「[StartMaintenance](#) API リファレンス」の「AWS OpsWorks」

の開始方法 AWS OpsWorks for Chef Automate

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks for Chef Automate では、[Chef Automate](#) サーバーを実行できます AWS。約 15 分で Chef サーバーをプロビジョニングできます。

2021 年 5 月 3 日以降、一部の Chef Automate サーバー属性を [AWS OpsWorks for Chef Automate](#) 保存します AWS Secrets Manager。詳細については、「[との統合 AWS Secrets Manager](#)」を参照してください。

以下のチュートリアルは、[Chef Automate](#) で最初の Chef サーバーを作成するのに役立ちます AWS OpsWorks for Chef Automate。

前提条件

開始する前に、以下の前提条件を満たしている必要があります

トピック

- [VPC のセットアップ](#)
- [カスタムドメインを使用するための前提条件 \(オプション\)](#)
- [EC2 のキーペアをセットアップする \(オプション\)](#)

VPC のセットアップ

AWS OpsWorks for Chef Automate サーバーは Amazon Virtual Private Cloud で動作する必要があります。既存の VPC にそのサーバーを追加するか、デフォルトの VPC を使用するか、またはそのサーバーを含めて新しい VPC を作成します。Amazon VPC の情報および新しい VPC の作成方法については、「[Amazon VPC 入門ガイド](#)」を参照してください。

独自の VPC を作成するか、既存の VPC を使用する場合、VPC には次の設定またはプロパティが必要です。

- VPC には少なくとも 1 つのサブネットが必要です。

AWS OpsWorks for Chef Automate サーバーがパブリックアクセス可能になる場合は、サブネットをパブリックにし、パブリック IP の自動割り当てを有効にします。

- [DNS resolution] は有効である必要があります。
- サブネットで、[Auto-assign public IP] を有効にします。

VPCs の作成やインスタンスの実行に慣れていない場合は、次の AWS CLI コマンドを実行して、AWS OpsWorks が提供する AWS CloudFormation テンプレートを使用して、単一のパブリックサブネットを持つ VPC を作成できます。を使用する場合は AWS Management Console、[テンプレート](#)を AWS CloudFormation コンソールにアップロードすることもできます。

```
aws cloudformation create-stack --stack-name OpsWorksVPC --template-url https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-vpc.yaml
```

カスタムドメインを使用するための前提条件 (オプション)

独自のドメインで Chef Automate サーバーをセットアップし、サーバーのエンドポイントとして使用するカスタムドメインのパブリックエンドポイントを指定できます。カスタムドメインを使用する場合は、このセクションで詳しく説明するように、次のすべてが必要です。

トピック

- [カスタムドメインをセットアップする](#)
- [証明書を取得する](#)
- [プライベートキーを取得する](#)

カスタムドメインをセットアップする

独自のカスタムドメインで Chef Automate サーバーを実行するには、<https://aws.my-company.com> などサーバーのパブリックエンドポイントが必要です。カスタムドメインを指定する場合は、前のセクションで説明したように、証明書とプライベートキーも指定する必要があります。

作成後にサーバーにアクセスするには、優先 DNS サービスに CNAME DNS レコードを追加します。このレコードは、Chef Automate サーバーの作成プロセスで生成されるエンドポイント (サーバーの Endpoint 属性の値) にカスタムドメインをポイントしている必要があります。サーバーがカスタムドメインを使用している場合は、生成された Endpoint 値を使用してサーバーにアクセスできません。

証明書を取得する

独自のカスタムドメインで Chef Automate サーバーをセットアップするには、PEM 形式の HTTPS 証明書が必要です。これは、単一の自己署名証明書、または証明書チェーンです。[Create Chef Automate server (Chef Automate サーバーの作成)] ワークフローを完了するときに、この証明書を指定する場合は、カスタムドメインとプライベートキーも指定する必要があります。

証明書の値の要件は次のとおりです。

- 自己署名証明書、カスタム証明書、または完全な証明書チェーンを指定できます。
- 証明書は、有効な X509 証明書、または PEM 形式の証明書チェーンである必要があります。
- 証明書はアップロード時に有効である必要があります。有効期間の開始 (証明書の NotBefore 日付) 前、または有効期間の終了 (証明書の NotAfter 日) 後に証明書を使用することはできません。
- 証明書の共通名またはサブジェクトの代替名 (SAN) が存在する場合は、カスタムドメインの値と一致する必要があります。
- 証明書は、[Custom private key (カスタムプライベートキー)] フィールドの値と一致する必要があります。

プライベートキーを取得する

独自のカスタムドメインで Chef Automate サーバーをセットアップするには、HTTPS を使用してサーバーに接続するための PEM 形式のプライベートキーが必要です。プライベートキーは暗号化しないでください。パスワードやパスフレーズで保護することはできません。カスタムプライベートキーを指定する場合は、カスタムドメインと証明書も指定する必要があります。

EC2 のキーペアをセットアップする (オプション)

Chef サーバーの通常の管理では、SSH 接続は不要であり、推奨されていません。[knife](#) コマンドを使用して、Chef サーバーに対するほとんどの管理タスクを実行できます。

Chef Automate ダッシュボードのサインインパスワードがわからなくなった場合や変更する場合には、SSH を使用してサーバーに接続する際に EC2 のキーペアが必要です。その場合は、既存のキーペアを使用するか、または新しいキーペアを作成できます。EC2 の新しい EC2 のキーペアの作成方法の詳細については、[「Amazon EC2 Key Pairs」](#) (Amazon EC2 のキーペア) を参照してください。

EC2 のキーペアが必要なければ、Chef サーバーを作成する準備ができています。

Chef Automate サーバーの作成

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。


Chef サーバーは、AWS OpsWorks for Chef Automate コンソールまたは [AWS CLI](#) を使用して作成できます。

トピック

- [で Chef Automate サーバーを作成する AWS Management Console](#)
- [を使用して Chef Automate サーバーを作成する AWS CLI](#)

で Chef Automate サーバーを作成する AWS Management Console


1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/opsworks/> で AWS OpsWorks コンソールを開きます。
2. AWS OpsWorks ホームページで、Chef Automate OpsWorks の に移動 を選択します。



AWS OpsWorks

AWS OpsWorks is a configuration management service that helps you build and operate highly dynamic applications, and propagate changes instantly.

AWS OpsWorks provides three solutions to configure your infrastructure:




OpsWorks Stacks

Define, group, provision, deploy, and operate your applications in AWS by using Chef in local mode.

[Go to OpsWorks Stacks](#)

[Learn more about OpsWorks Stacks](#)




OpsWorks for Chef Automate

Create Chef servers that include Chef Automate premium features, and use the Chef DK or any Chef tooling to manage them.

[Go to OpsWorks for Chef Automate](#)

[Learn more about OpsWorks for Chef Automate](#)



OpsWorks for Puppet Enterprise

Create Puppet servers that include Puppet Enterprise features. Inspect, deliver, update, monitor, and secure your infrastructure.

[Go to OpsWorks for Puppet Enterprise](#)

[Learn more about OpsWorks for Puppet Enterprise](#)

3. AWS OpsWorks for Chef Automate ホームページで、Chef Automate サーバーの作成 を選択します。

Welcome to OpsWorks for Chef Automate

OpsWorks for Chef Automate helps you automate, provision, and configure your environment. The Chef Automate platform delivers DevOps workflow, automated compliance, and end-to-end pipeline visibility.

A Chef Automate server manages nodes in your environment, stores information about those nodes, and serves as a central repository for your Chef cookbooks.

[Create Chef Automate server](#)

4. [Set name, region, and type] ページで、サーバーの名前を指定します。Chef サーバー名は 40 文字以内で、英数字とハイフンのみを使用できます。サポートされているリージョンを選択し、管理するノード数をサポートしているインスタンスタイプを選択します。サーバーを作成した後も、必要に応じてインスタンスタイプを変更できます。このウォークスルーでは、米国西部 (オレゴン) で [m5.large] インスタンスタイプを作成します。[次へ] をクリックします。

Set name, region, and type

Type a name for the Chef Automate server, select the region in which you want to locate the server, and select the Amazon EC2 instance type that best fits your needs.

Chef Automate server name ⓘ
Maximum 40 characters. Has to start with a letter, and can only contain letters, numbers, and hyphens.

Chef Automate server region ⓘ

EC2 instance type

m5.large 8 GiB Memory Supports up to 200 nodes	r5.xlarge 30 GiB Memory Supports up to 500 nodes	r5.2xlarge 61 GiB Memory Supports 500+ nodes
---	---	---

[See our pricing plan.](#)

Cancel **Next**

5. [Configure server (サーバーの構成)] ページで、キーペア名を指定しない場合は、[SSH key (SSH キー)] ドロップダウンリストでデフォルトの選択のままにします。

Configure server

Configure the server's EC2 instance credentials and server endpoint.

Select an SSH key

Select the EC2 key pair. You need this key to connect to the Chef Automate server EC2 instance by using SSH.

SSH key ⓘ
You can still use Knife commands to communicate with the Chef Automate server.

6. サーバーを独自のカスタムドメインで使用しない場合は、[Specify server endpoint (サーバーエンドポイントを指定)] をデフォルトの [Use an automatically-generated endpoint (自動的に生成されたエンドポイントを指定する)] のままにして、[Next (次へ)] を選択します。カスタムドメインを設定するには、次のステップに進みます。

Specify server endpoint

Specify a public endpoint that you can use to access the Chef Automate server. It can be either a custom domain that you provide, or an automatically-generated endpoint that uses the opsworks-cm.io domain.

Endpoint ⓘ
This is an automatically-generated endpoint that uses the opsworks-cm.io domain name.

7. カスタムドメインを使用するには、[Specify server endpoint (サーバーエンドポイントを指定)] で、ドロップダウンリストから [Use a custom domain (カスタムドメインを使用)] を選択します。

Specify server endpoint

Specify a public endpoint that you can use to access the Chef Automate server. It can be either a custom domain that you provide, or an automatically-generated endpoint that uses the opsworks-cm.io domain.

Endpoint ⓘ

Provide your own custom domain to be used as the server endpoint.

Fully qualified domain name (FQDN) ⓘ

The fully qualified domain name you want to use for your Chef Automate server. Example: myserver.mycompany.com

SSL certificate ⓘ

A PEM encoded SSL certificate issued for your FQDN. If the certificate is not self-signed, you must also provide the whole SSL certificate chain.

SSL private key ⓘ

The PEM encoded SSL private key for your SSL certificate.

- a. [Fully qualified domain name (FQDN) (完全修飾ドメイン名 (FQDN))] で、FQDN を指定します。使用するドメイン名を所有している必要があります。
 - b. [SSL certificate] (SSL 証明書) に、PEM 形式の証明書全体を貼り付けます。この証明書は -----BEGIN CERTIFICATE----- で始まり、-----END CERTIFICATE----- で終わります。SSL 証明書のサブジェクトは、前のステップで入力した FQDN と一致する必要があります。
 - c. [SSL private key] (SSL プライベートキー) には、RSA プライベートキー全体を貼り付けます。このプライベートキーは -----BEGIN RSA PRIVATE KEY----- で始まり、-----END RSA PRIVATE KEY----- で終わります。SSL プライベートキーは、前のステップで入力した SSL 証明書のパブリックキーと一致する必要があります。[次へ] をクリックします。
8. [Configure Advanced Settings] (詳細設定の設定) ページの [Network and Security] (ネットワークとセキュリティ) 領域で、VPC、サブネット、および 1 つ以上のセキュリティグループを選択します。VPC の要件を以下に示します。
- VPC には、少なくとも 1 つのパブリックサブネットが必要です。
 - DNS 解決を有効にする必要があります。
 - [Auto-assign public IP (パブリック IP の自動割り当て)] を、パブリックサブネット上で有効にする必要があります。

AWS OpsWorks 使用するセキュリティグループ、サービスロール、インスタンスプロファイルがまだない場合は、で生成できます。サーバーは複数のセキュリティグループのメンバーにできます。このページから次に進んだ後に、Chef サーバーのネットワーク設定およびセキュリティ設定を変更することはできません。

Network and security

You cannot change network and security settings after you launch your Chef Automate server.

VPC ⓘ

You have selected a non-default VPC. Be sure the selected VPC has outbound network access. [Learn more.](#)

Subnet ⓘ

Associate Public IP Address Yes No

Choose Yes if the selected subnet is public.

Security groups ⓘ

✕ ✕

Please ensure the following ports are open: 443 (https)

Service role ⓘ

Instance profile ⓘ

- [System maintenance] セクションで、システムメンテナンスを開始する日付と時刻を設定します。システムメンテナンス中はサーバーがオフラインになることを想定する必要があるため、通常の営業時間内でサーバー需要が低い時刻を選択します。接続されたノードは、メンテナンスが完了するまで `pending-server` 状態になっています。

メンテナンス時間は必須です。、、または APIs を使用して AWS Management Console AWS CLI、開始日時を後で変更できます。

System maintenance

AWS OpsWorks installs updates for Chef Automate minor versions or security packages in the time range and on the weekday that you specify here. **Your Chef Automate server will be offline during system maintenance.**

Start day ⓘ

Start time (UTC) ⓘ

- バックアップを設定します。デフォルトでは、自動バックアップが有効になっています。自動バックアップを開始する頻度と時刻を設定し、Amazon Simple Storage Service に保存するバツ

クアップの世代数を設定します。最大 30 個のバックアップが保持されます。最大値に達すると、最も古いバックアップ AWS OpsWorks for Chef Automate を削除して、新しいバックアップ用のスペースを確保します。

Automated backup

AWS OpsWorks supports two ways to back up your Chef Automate server: manual or automated. Backups are uploaded to your Amazon S3 bucket. If you ever need to restore your Chef Automate server, you can restore it by applying a backup that you choose.

Enable automated backup Yes No

Frequency ⓘ

Start time (UTC) ⓘ

Number of generations to keep

Specify how many automated backups to keep. Minimum: 1, maximum: 30.

- (オプション) [Tags] (タグ) で、サーバーおよび関連リソース (EC2 インスタンス、Elastic IP アドレス、セキュリティグループ、S3 バケット、バックアップなど) にタグを追加します。AWS OpsWorks for Chef Automate サーバーのタグ付けの詳細については、「」を参照してください [AWS OpsWorks for Chef Automate リソースでのタグの使用](#)。
- 詳細設定が完了したら、[Next] を選択します。
- [確認] ページで選択内容を確認します。サーバーを作成する準備ができたなら、[Launch] を選択します。

Chef サーバーの作成 AWS OpsWorks を待っている間に、に進み、Starter Kit [スターターキット](#)を使用して [Chef サーバーを設定する](#)と Chef Automate ダッシュボードの認証情報をダウンロードします。これらをダウンロードするために、サーバーがオンラインになるまで待つ必要はありません。

サーバーの作成が完了すると、AWS OpsWorks for Chef Automate のホームページで、Chef サーバーが [online] のステータスになり、利用可能になります。サーバーがオンラインになると、Chef Automate ダッシュボードがサーバーのドメインで `https://your_server_name-random.region.opsworks-cm.io` の形式の URL で利用できます。

を使用して Chef Automate サーバーを作成する AWS CLI

AWS CLI コマンド `AWS OpsWorks for Chef Automate` を実行してサーバーを作成することは、コンソールでサーバーを作成することとは異なります。使用する既存のロールを指定しない場合、コ

コンソールでサービスロールとセキュリティグループ `AWS OpsWorks` が作成されます。では `AWS CLI`、セキュリティグループを指定しない場合、セキュリティグループ `AWS OpsWorks` を作成できますが、サービスロールは自動的に作成されません。`create-server` コマンドの一部としてサービスロール ARN を指定する必要があります。コンソールで、`AWS OpsWorks` が `Chef Automate` サーバーを作成している間、`Chef Automate` スターターキットと `Chef Automate` ダッシュボードのサインイン認証情報をダウンロードします。を使用してサーバーを作成する `AWS OpsWorks for Chef Automate` 場合、これを行うことができないため `AWS CLI`、`JSON 処理ユーティリティ` を使用して、新しい `AWS OpsWorks for Chef Automate` サーバーがオンラインになった後、`create-server` コマンドの結果からサインイン認証情報とスターターキットを取得します。または、新しい `AWS OpsWorks for Chef Automate` サーバーがオンラインになった後、コンソールでサインイン認証情報の新しいセットと新しいスターターキットを生成できます。

ローカルコンピュータでまだ `create-server` を実行していない場合は `AWS CLI`、`AWS コマンドラインインターフェイス` ユーザーガイド `AWS CLI` の [インストール手順に従って](#) をダウンロードしてインストールします。このセクションでは、`create-server` コマンドで使用できるパラメータのすべては説明しません。`create-server` パラメータの詳細については、「[create-server リファレンス](#)」の「`AWS CLI`」を参照してください。

1. 必ず前提条件、特に「[VPC のセットアップ](#)」を満たしてください。または、使用する既存の `VPC` があることを確認してください。`Chef Automate` サーバーを作成するには、サブネット ID が必要です。
2. 必要に応じて、`Chef` の中核キーを [OpenSSL](#) を使用して作成し、そのキーをローカルコンピュータ上の安全で便利なファイルに保存してください。中核キーは、`create-server` コマンドで指定しない場合は、サーバー作成プロセスの一部として自動的に生成されます。このステップをスキップする場合は、代わりに、`create-server` コマンドにより生成される `Chef Automate` 主要キーを取得できます。`Chef Automate` 主要キーの値は、`RSA` キーペアの公開鍵に相当するため、下記のコマンドを使用して主要キーを生成する場合は、必ず `-pubout` パラメータを含めてください。詳細については、ステップ 6 を参照してください。

```
umask 077
openssl genrsa -out "pivotal" 2048
openssl rsa -in "pivotal" -pubout
```

3. サービスロールとインスタンスプロファイルを作成します。は、両方を作成するために使用できる `AWS CloudFormation` テンプレート `AWS OpsWorks` を提供します。次の `AWS CLI` コマンドを実行して、サービスロールとインスタンスプロファイルを作成する `AWS CloudFormation` スタックを作成します。

```
aws cloudformation create-stack --stack-name OpsWorksCMRoles --template-url
https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-
cm-roles.yaml --capabilities CAPABILITY_NAMED_IAM
```

4. ガスタックの作成 AWS CloudFormation を完了したら、アカウント内のサービスロールARNsを検索してコピーします。

```
aws iam list-roles --path-prefix "/service-role/" --no-paginate
```

`list-roles` コマンドの結果内で、次のようなサービスロール ARN のエントリを探します。サービスロール ARN を書き留めます。Chef Automate サーバーを作成するにはこれらの値が必要です。

```
{
  "AssumeRolePolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "ec2.amazonaws.com"
        }
      }
    ]
  },
  "RoleId": "AROZZZZZZZZZZQ6R22HC",
  "CreateDate": "2018-01-05T20:42:20Z",
  "RoleName": "aws-opsworks-cm-ec2-role",
  "Path": "/service-role/",
  "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
},
{
  "AssumeRolePolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "opsworks-cm.amazonaws.com"
        }
      }
    ]
  },
  "RoleId": "AROZZZZZZZZZZQ6R22HC",
  "CreateDate": "2018-01-05T20:42:20Z",
  "RoleName": "aws-opsworks-cm-ec2-role",
  "Path": "/service-role/",
  "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
}
```



```

    }
  }
]
},
"RoleId": "AR0ZZZZZZZZZZZZZZZZ6QE",
"CreateDate": "2018-01-05T20:42:20Z",
"RoleName": "aws-opsworks-cm-service-role",
"Path": "/service-role/",
"Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-
role"
}

```

5. アカウントでインスタンスプロファイルの ARN を検索してコピーします。

```
aws iam list-instance-profiles --no-paginate
```

`list-instance-profiles` コマンドの結果内で、次のようなインスタンスプロファイル ARN のエントリを探します。インスタンスプロファイルの ARN を書き留めます。Chef Automate サーバーを作成するにはこれらの値が必要です。

```

{
  "Path": "/",
  "InstanceProfileName": "aws-opsworks-cm-ec2-role",
  "InstanceProfileId": "EXAMPLEDC6UR3LTUW7VHK",
  "Arn": "arn:aws:iam::123456789012:instance-profile/aws-opsworks-cm-ec2-role",
  "CreateDate": "2017-01-05T20:42:20Z",
  "Roles": [
    {
      "Path": "/service-role/",
      "RoleName": "aws-opsworks-cm-ec2-role",
      "RoleId": "EXAMPLEEE4STNUQG6R22HC",
      "Arn": "arn:aws:iam::123456789012:role/service-role/aws-opsworks-cm-
ec2-role",
      "CreateDate": "2017-01-05T20:42:20Z",
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": "ec2.amazonaws.com"
            }
          }
        ]
      }
    }
  ]
}

```

```
        "Action": "sts:AssumeRole"
      }
    ]
  }
},
```

6. `create-server` コマンドを実行して AWS OpsWorks for Chef Automate サーバーを作成します。
 - `--engine` の値は、`ChefAutomate`、`--engine-model` は、`Single`、`--engine-version` は 12 です。
 - サーバー名は、AWS アカウント内で各リージョン内で一意である必要があります。サーバー名は文字で始める必要があります。その後は文字、数字、またはハイフン (-) を最大 40 文字まで使用できます。
 - ステップ 4 と 5 でコピーしたインスタンスプロファイル ARN とサービスロール ARN を使用します。
 - 有効なインスタンスタイプは `m5.large`、`r5.xlarge`、または `r5.2xlarge` です。これらのインスタンスタイプの仕様の詳細については、「Amazon EC2 ユーザーガイド」の「[インスタンスタイプ](#)」を参照してください。
 - `--engine-attributes` パラメータはオプションです。いずれかまたは両方の値を指定しない場合、サーバー作成プロセスで値が生成されます。`--engine-attributes` を追加する場合は、ステップ 2 で生成した `CHEF_AUTOMATE_PIVOTAL_KEY` 値、`CHEF_AUTOMATE_ADMIN_PASSWORD`、またはその両方を指定します。

`CHEF_AUTOMATE_ADMIN_PASSWORD` の値を設定しない場合、`create-server` レスポンスの一部としてパスワードが生成され返されます。コンソールでスターターキットをもう一度ダウンロードして、このパスワードを再生成することもできます。パスワードの最小の長さは 8 文字、最大は 32 文字です。パスワードには、文字、数字、および特殊文字 (!/@#\$\$%^+=_) を使用できます。パスワードは、少なくとも 1 つの小文字、1 つの大文字、1 つの数字、および 1 つの特殊文字を含む必要があります。

- SSH キーペアはオプションですが、Chef Automate ダッシュボードの管理者パスワードをリセットする必要がある場合に Chef Automate サーバーに接続することができます。SSH キーペアの作成の詳細については、「Amazon EC2 User Guide」(Amazon EC2 ユーザーガイド) の「[Amazon EC2 Key Pairs](#)」(Amazon EC2 のキーペア) を参照してください。

- カスタムドメインを使用するには、コマンドに以下のパラメータを追加します。それ以外の場合は、Chef Automate サーバー作成プロセスによって自動的にエンドポイントが生成されます。カスタムドメインを構成するには、3つのパラメータすべてが必要です。これらのパラメータを使用するためのその他の要件については、AWS OpsWorks CM API リファレンス [CreateServer](#) の「」を参照してください。
- `--custom-domain` - サーバーのオプションのパブリックエンドポイント (`https://aws.my-company.com` など)。
- `--custom-certificate` - PEM 形式の HTTPS 証明書。値には、単一の自己署名証明書、または証明書チェーンを指定できます。
- `--custom-private-key` - HTTPS を使用してサーバーに接続するための PEM 形式のプライベートキー。プライベートキーは暗号化しないでください。パスワードやパスフレーズで保護することはできません。
- 週 1 回のシステムメンテナンスが必要です。次の形式で有効な値を指定する必要があります: `DDD:HH:MM`。指定時刻は協定世界時 (UTC) です。 `--preferred-maintenance-window` の値を指定しない場合、火曜日、水曜日、または金曜日の 1 時間がランダムでデフォルト値になります。
- `--preferred-backup-window` の有効な値は次の形式のいずれかで指定する必要があります: 日次バックアップの場合は `HH:MM`、週次バックアップの場合は `DDD:HH:MM`。指定時刻は UTC です。デフォルト値は日次で開始時間はランダムです。自動バックアップを無効にするには、代わりにパラメータ `--disable-automated-backup` を追加します。
- `--security-group-ids` に、1 つ以上のセキュリティグループ ID をスペースで区切って入力します。
- `--subnet-ids` には、サブネット ID を入力します。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single"
--engine-version "12" --server-name "server_name" --instance-profile-arn
"instance_profile_ARN" --instance-type "instance_type" --engine-attributes
'{"CHEF_AUTOMATE_PIVOTAL_KEY":"pivotal_key","CHEF_AUTOMATE_ADMIN_PASSWORD":"password"}'
--key-pair "key_pair_name" --preferred-maintenance-window
"ddd:hh:mm" --preferred-backup-window "ddd:hh:mm" --security-group-
ids security_group_id1 security_group_id2 --service-role-arn "service_role_ARN" --
subnet-ids subnet_ID
```

次に例を示します。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-
model "Single" --engine-version "12" --server-name "automate-06" --
instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-
opsworks-cm-ec2-role" --instance-type "m5.large" --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
--key-pair "amazon-test" --preferred-maintenance-window "Mon:08:00" --preferred-
backup-window "Sun:02:00" --security-group-ids sg-b00000001 sg-b00000008 --service-
role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-service-role"
--subnet-ids subnet-300aaa00
```

次の例では、カスタムドメインを使用する Chef Automate サーバーを作成します。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single" --
engine-version "12" \
  --server-name "my-custom-domain-server" \
  --instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-opsworks-
cm-ec2-role" \
  --instance-type "m5.large" \
  --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
\
  --custom-domain "my-chef-automate-server.my-corp.com" \
  --custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
CERTIFICATE-----" \
  --custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
RSA PRIVATE KEY-----" \
  --key-pair "amazon-test" \
  --preferred-maintenance-window "Mon:08:00" \
  --preferred-backup-window "Sun:02:00" \
  --security-group-ids sg-b00000001 sg-b00000008 \
  --service-role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-
service-role" \
  --subnet-ids subnet-300aaa00
```

以下の例では、2つのタグ Stage: Production および Department: Marketing を追加する Chef Automate サーバーを作成します。AWS OpsWorks for Chef Automate サーバーでのタグの追加と管理の詳細については、このガイド[AWS OpsWorks for Chef Automate リソースでのタグの使用](#)の「」を参照してください。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single" --
engine-version "12" \
```

```

--server-name "my-test-chef-server" \
--instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-opsworks-
cm-ec2-role" \
--instance-type "m5.large" \
--engine-attributes
'{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
\
--key-pair "amazon-test" \
--preferred-maintenance-window "Mon:08:00" \
--preferred-backup-window "Sun:02:00" \
--security-group-ids sg-b00000001 sg-b00000008 \
--service-role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-
service-role" \
--subnet-ids subnet-300aaa00 \
--tags [{"Key\":"Stage\"}, {"Value\":"Production\"}], [{"Key\":"Department\"},
 {"Value\":"Marketing\"}]}

```

7. AWS OpsWorks for Chef Automate 新しいサーバーの作成には約 15 分かかります。create-server コマンドの出力を閉じたり、シェルセッションを閉じたりしないでください。今後表示されない重要な情報が出力に含まれている場合があります。create-server の結果からパスワードとスターターキットを取得するには、次のステップに進みます。

サーバーでカスタムドメインを使用している場合は、create-server コマンドの出力で Endpoint 属性の値をコピーします。次に例を示します。

```
"Endpoint": "automate-07-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
```

8. がキーとパスワード AWS OpsWorks for Chef Automate を生成することを選択した場合は、jq などの JSON プロセッサを使用して、create-server 結果から使用可能な形式で抽出できます。jq をインストールした後、以下のコマンドを実行して中枢キー、Chef Automate ダッシュボード管理者パスワード、およびスターターキットを抽出できます。ステップ 4 で独自の中枢キーとパスワードを指定しなかった場合は、抽出した中枢キーと管理者パスワードを使いやすく安全な場所に保存してください。

```

#Get the Chef password:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
"CHEF_AUTOMATE_ADMIN_PASSWORD") | .Value'

#Get the Chef Pivotal Key:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
"CHEF_AUTOMATE_PIVOTAL_KEY") | .Value'

```

```
#Get the Chef Starter Kit:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
"CHEF_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
```

9. オプションで、create-serverコマンド結果からスターターキットを抽出しなかった場合は、AWS OpsWorks for Chef Automate コンソールのサーバーのプロパティページから新しいスターターキットをダウンロードできます。新しいスターターキットをダウンロードすると、Chef Automate ダッシュボードの管理者パスワードがリセットされます。
10. カスタムドメインを使用していない場合は、次のステップに進みます。サーバーでカスタムドメインを使用している場合は、企業の DNS 管理ツールで CNAME エントリを作成して、ステップ 7 でコピーした AWS OpsWorks for Chef Automate エンドポイントにカスタムドメインをポイントします。このステップを完了するまで、カスタムドメインを使用するサーバーにアクセスしたりサインインしたりすることはできません。
11. サーバーの作成プロセスが完了したら、「[the section called “設定を完了してクックブックをアップロードする”](#)」に進みます。

スターターキットを使用して Chef サーバーを設定する

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Chef サーバー作成の進行中に、AWS OpsWorks for Chef Automate コンソールでプロパティページを開きます。新しい Chef サーバーの初回操作時に、[Properties] ページで 2 つの必須項目をダウンロードするように求められます。Chef サーバーがオンラインになる前に、これらの項目をダウンロードします。新しいサーバーがオンラインになった後は、ダウンロードボタンは使用できません。

my-chef-server 🔗 [Chef Automate dashboard](#) (not yet available) Actions ▾

AWS OpsWorks is creating your Chef Automate server. This takes about 20 minutes.

Creating an Elastic IP address → Launching an EC2 instance → Installing Chef Automate server

Make sure you download the following before your server is online.

- 1 Sign-in credentials for your Chef Automate dashboard
- 2 Starter Kit for your Chef Automate server

i Download the sign-in credentials for your [Chef Automate dashboard](#).

▶ Show sign-in credentials

[Download credentials](#)

AWS OpsWorks does not save these credentials, so it is the last time they are available for viewing and downloading. After your server is online, you can change the password by signing in to its [Chef Automate dashboard](#).

i Download the Starter Kit, and follow the [documentation](#) to finish the setup when your server is online.

[Download Starter Kit](#)

The Starter Kit contains a Readme with examples, a knife.rb configuration file, and a private key. A new key pair is generated and reset each time you download the Starter Kit.

- [Sign-in credentials for the Chef server (Chef サーバー用のサインイン認証情報)]. これらの認証情報を使用して Chef Automate ダッシュボードにサインインし、ワークフローやコンプライアンススキャンなどの Chef Automate のプレミアム機能を使用します。AWS OpsWorks はこれらの認証

情報を保存しません。これは、表示とダウンロードが最後に利用可能になるときです。必要であれば、これらの認証情報で提供されたパスワードをサインイン後に変更できます。

- Starter Kit (スターターキット)。スターターキットには、サンプル付きの README ファイル、knife.rb の設定ファイル、およびプライマリユーザーまたは主要ユーザー用のプライベートキーが含まれています。スターターキットをダウンロードするごとに、新しいキーペアが生成され、以前のキーはリセットされます。

Starter Kit .zip ファイルには、新しいサーバーでのみ機能する認証情報に加えて、任意の AWS OpsWorks for Chef Automate サーバーで動作する Chef リポジトリの簡単な例が含まれています。Chef リポジトリには、Chef で管理するノード用のクックブック、ロール、設定ファイル、および他のアーティファクトが保存されます。このリポジトリは、バージョン管理システム (Git など) に保存し、ソースコードとして扱うことをお勧めします。Git で追跡されるように Chef リポジトリをセットアップする方法に関する情報およびサンプルについては、Chef ドキュメントの「[About the chef-repo](#)」(chef-repo について) を参照してください。

前提条件

1. サーバー作成の進行中に、Chef サーバーのサインイン認証情報をダウンロードし、セキュアだが便利な場所に保存します。
2. スターターキットをダウンロードし、スターターキットの .zip ファイルをワークスペースディレクトリに解凍します。スターターキットのプライベートキーは共有しないでください。他のユーザーが Chef サーバーを管理する場合は、後で Chef Automate ダッシュボードでそのユーザーを管理者として追加します。
3. [\[Chef Workstation\]](#) (以前は Chef Development Kit または Chef DK として知られていた) をダウンロードして、Chef サーバおよびノードの管理に使用するコンピュータにインストールします。[knife](#) ユーティリティは Chef Workstation の一部です。その手順については、Chef のウェブサイトにある「[Install Chef Workstation](#)」(Chef Workstation のインストール) を参照してください。

スターターキットの内容を見る

スターターキットの内容は以下のとおりです。

- cookbooks/ - 作成するクックブックのディレクトリ。cookbooks/ フォルダには、[Chef スーパーマーケット](#) ウェブサイトの nginx クックブックに依存する opsworks-webserver クックブック、ラッパークックブックが含まれています。クックブックの依存関係が cookbooks/ の

ディレクトリ内で利用できない場合、`Policyfile.rb` のデフォルトでは `Chef supermarket` を二次ソースとして使用します。

- `Policyfile.rb` - ノードのポリシーになるクックブック、依存関係、属性を定義する Ruby ベースのポリシーファイル。
- `userdata.sh` および `userdata.ps1` - Chef Automate サーバーの起動後に、ユーザーデータファイルを使用してノードを自動的に関連付けることができます。`userdata.sh` は Linux ベースのノードのブートストラップ用、`userdata.ps1` は Windows ベースのノード用です。
- `Berkshelf` - `Berkshelf` と `berks` のコマンドを使用してクックブックとその依存関係をアップロードする場合は、このファイルを使用できます。このチュートリアルでは、`Policyfile.rb` と `Chef` コマンドを使用して、クックブック、依存関係、属性をアップロードします。
- `README.md` - スターターキットを使用して Chef Automate サーバーを初めて設定する方法について説明する Markdown ベースのファイル。
- `.chef` - `knife` 設定ファイル (`knife.rb`) とシークレット認証キーファイル (`.pem`) を含む隠しディレクトリ。
 - `.chef/knife.rb` - `knife` 設定ファイル (`knife.rb`)。この [knife.rb](#) ファイルは、Chef の [knife](#) ツールオペレーションが AWS OpsWorks for Chef Automate サーバーに対して実行されるように設定されています。
 - `.chef/ca_certs/opsworks-cm-ca-2020-root.pem` - AWS OpsWorksに付属している、認証機関 (CA) による署名付きの SSL プライベートキー。このキーによって、サーバーで管理されるノード上の Chef Infra クライアントエージェントに対して、サーバーが自身を識別できるようになります。

Chef リポジトリを設定する

Chef リポジトリには複数のディレクトリが置かれています。スターターキットの各ディレクトリには、そのディレクトリの目的、および Chef でシステムを管理するための使用方法が記述されている README ファイルがあります。Chef サーバーにクックブックをインストールするには、2つの方法があります。`knife` コマンドを実行する方法と、Chef コマンドを実行する方法です。後者のコマンドは、ポリシーファイル (`Policyfile.rb`) をサーバーにアップロードし、指定されたクックブックをダウンロードしてインストールします。このチュートリアルでは、Chef コマンドと `Policyfile.rb` を使用してクックブックをサーバーにインストールします。

1. クックブックを保存するためのディレクトリ (`chef-repo` など) をローカルコンピュータ上に作成します。このリポジトリにクックブック、ロール、その他のファイルを追加したら、

CodeCommitGit、Amazon S3 などの安全なバージョン管理されたシステムにアップロードまたは保存することをお勧めします。

2. chef-repo ディレクトリに、以下のディレクトリを作成します。

- cookbooks/ - クックブックを保存する。
- roles/ - ロールを .rb 形式または .json 形式で保存します。
- environments/ - 環境を .rb 形式または .json 形式で保存します。

Policyfile.rb を使用してリモートソースからクックブックを取得する

このセクションでは、クックブックを指定するように Policyfile.rb を編集してから、そのファイルをサーバーにアップロードしてクックブックをインストールする Chef コマンドを実行します。

1. スターターキットの Policyfile.rb を表示します。デフォルトでは Policyfile.rb には、opsworks-webserver ラッパークックブックが含まれています。これは、Chef Supermarket のウェブサイトで入手可能な [nginx](#) クックブックの依存関係となります。nginx は、管理対象ノードにウェブサーバーをインストールして設定するクックブックです。また、管理対象ノードに Chef Infra クライアントエージェントをインストールする、必須の chef-client クックブックも指定されています。

Policyfile.rb は、ノードのコンプライアンススキャンの設定に使用できるオプションの Chef Audit クックブックも参照しています。コンプライアンススキャンの設定と管理対象ノードのコンプライアンス結果の取得の詳細については、「[でのコンプライアンススキャン AWS OpsWorks for Chef Automate](#)」を参照してください。今すぐコンプライアンススキャンと監査を設定しない場合は、'audit' セクションから run_list を削除します。ファイルの末尾に audit クックブックの属性を指定しないでください。

```
# Policyfile.rb - Describe how you want Chef to build your system.
#
# For more information about the Policyfile feature, visit
# https://docs.chef.io/policyfile.html
#
# A name that describes what the system you're building with Chef does.
name 'opsworks-demo-webserver'
```

```
# The cookbooks directory is the preferred source for external cookbooks

default_source :chef_repo, "cookbooks/" do |s|

  s.preferred_for "nginx", "windows", "chef-client", "yum-epel", "seven_zip",
                 "build-essential", "mingw", "ohai", "audit", "logrotate", "cron"

end
# Alternative source
default_source :supermarket

# run_list: chef-client runs these recipes in the order specified.

run_list 'chef-client',
         'opsworks-webserver',
         'audit'
# add 'ssh-hardening' to your runlist to fix compliance issues detected by the ssh-
baseline profile

# Specify a custom source for a single cookbook:

cookbook 'opsworks-webserver', path: 'cookbooks/opsworks-webserver'

# Policyfile defined attributes

# Define audit cookbook attributes
default["opsworks-demo"]["audit"]["reporter"] = "chef-server-automate"
default["opsworks-demo"]["audit"]["profiles"] = [
  {
    "name": "DevSec SSH Baseline",
    "compliance": "admin/ssh-baseline"
  }
]
```

以下に示しているのは、現時点で Policyfile.rb ウェブサーバーのみを設定する場合に、audit クックブックとその属性を削除した nginx の例です。

```
# Policyfile.rb - Describe how you want Chef to build your system.
#
# For more information on the Policyfile feature, visit
# https://docs.chef.io/policyfile.html
```

```
# A name that describes what the system you're building with Chef does.
name 'opsworks-demo-webserver'

# Where to find external cookbooks:
default_source :supermarket

# run_list: chef-client will run these recipes in the order specified.
run_list 'chef-client',
         'opsworks-webserver'

# Specify a custom source for a single cookbook:
cookbook 'opsworks-webserver', path: 'cookbooks/opsworks-webserver'
```

Policyfile.rb に変更を加えた場合は、必ずこのファイルを保存してください。

2. Policyfile.rb に定義されているクックブックをダウンロードしてインストールします。

```
chef install
```

すべてのクックブックはクックブックの metadata.rb ファイルでバージョンングされています。クックブックを変更するたびに metadata.rb にあるクックブックのバージョンを上げる必要があります。

3. コンプライアンススキャンを設定し、audit クックブックの情報をポリシーファイルに保存した場合は、ポリシー opsworks-demo をサーバーにプッシュします。

```
chef push opsworks-demo
```

4. ステップ 3 が完了したら、ポリシーのインストールを確認します。以下のコマンドを実行します。

```
chef show-policy
```

結果は以下のようになります。

```
opsworks-demo-webserver
=====
* opsworks-demo: ec0fe46314
```

- これで、Chef Automate サーバーにノードを追加 (ブートストラップ) する準備ができました。ノードの関連付けを自動化するには、[でノードを自動的に追加する AWS OpsWorks for Chef Automate](#) のステップに従うか、または [ノードを個別に追加します](#) のステップに従ってノードを 1 つずつ追加します。

(代替) Berkshelf を使用してリモートソースからクックブックを取得する

Berkshelf は、クックブックとその依存関係を管理するためのツールです。ローカルストレージにクックブックをインストールするために `Policyfile.rb` の代わりに Berkshelf を使用する場合は、前のセクションの代わりにこのセクションの手順を使用します。Chef サーバーで使用するクックブックとそのバージョンを指定し、クックブックをアップロードできます。スターターキットには、クックブックの一覧表示に使用できる `Berksfile` というファイルが含まれています。

- 使用を開始するには、含まれている `Berksfile` に `chef-client` クックブックを追加します。`chef-client` クックブックは、Chef Automate サーバーに接続する各ノードで Chef Infra クライアントエージェントソフトウェアを設定します。このクックブックの詳細については、Chef Supermarket にある「[Chef Client Cookbook](#)」(Chef クライアントクックブック) を参照してください。
- テキストエディタを使用して、ウェブサーバーアプリケーションをインストールする `Berksfile` に別のクックブックを追加します。たとえば、Apache ウェブサーバーをインストールする `apache2` クックブックです。`Berksfile` は以下ようになります。

```
source 'https://supermarket.chef.io'  
cookbook 'chef-client'  
cookbook 'apache2'
```

- そのクックブックをローカルコンピュータにダウンロードしてインストールします。

```
berks install
```

- そのクックブックを Chef サーバーにアップロードします。

Linux では、次のコマンドを実行します。

```
SSL_CERT_FILE='.chef/ca_certs/opsworks-cm-ca-2020-root.pem' berks upload
```

Windows では、PowerShell セッションで次の Chef Workstation コマンドを実行します。コマンドを実行する前に、`RemoteSigned` の実行ポリシーを PowerShell に設定してください。

追加して `chef shell-init`、Chef Workstation ユーティリティコマンドを で使用できるようにします PowerShell。

```
$env:SSL_CERT_FILE="ca_certs\opsworks-cm-ca-2020-root.pem"
chef shell-init berks upload
Remove-Item Env:\SSL_CERT_FILE
```

5. Chef Automate サーバーで現在使用可能なクックブックのリストを表示して、クックブックがインストールされていることを確認します。そのためには、以下の `knife` コマンドを実行します。

AWS OpsWorks for Chef Automate サーバーで管理するノードを追加する準備ができました。

```
knife cookbook list
```

(オプション) カスタムドメインを使用するように **knife** を設定します。

Chef Automate サーバーがカスタムドメインを使用している場合は、サーバーの証明書チェーンに署名したルート CA の PEM 証明書を追加するか、証明書が自己署名の場合はサーバー PEM 証明書を追加する必要があります。ca_certs は、Chef knife ユーティリティによって信頼される認証機関 (CA) を含む chef/ のサブディレクトリです。

カスタムドメインを使用していない場合、またはカスタム証明書がオペレーティングシステムで信頼されているルート CA によって署名されている場合は、このセクションを省略できます。それ以外の場合は、次のステップで説明するように、Chef Automate サーバーの SSL 証明書を信頼するように knife を設定します。

1. 以下のコマンドを実行します。

```
knife ssl check
```

結果が次のような場合は、この手順の残りの部分をスキップして、「[Chef サーバーで管理するノードを追加する](#)」に進みます。

```
Connecting to host my-chef-automate-server.my-corp.com:443
      Successfully verified certificates from 'my-chef-automate-server.my-corp.com'
```

次のようなエラーメッセージが表示された場合は、次のステップに進みます。

```
Connecting to host my-chef-automate-server.my-corp.com:443
      ERROR: The SSL certificate of my-chef-automate-server.my-corp.com could
not be verified.
      ...
```

2. `knife ssl fetch` を実行して、AWS OpsWorks for Chef Automate サーバーの証明書を信頼します。または、サーバーのルート CA 証明書 (PEM 形式) を、`trusted_certs_dir` の出力の `knife ssl check` の値であるディレクトリに手動でコピーすることもできます。デフォルトでは、このディレクトリはスターターキットの `.chef/ca_certs/` にあります。出力は次のようになります。

```
WARNING: Certificates from my-chef-automate-server.my-corp.com will be fetched and
placed in your trusted_cert
directory (/Users/username/starterkit/.chef/../../chef/ca_certs).

Knife has no means to verify these are the correct certificates. You
should
verify the authenticity of these certificates after downloading.

Adding certificate for my-chef-automate-server in /Users/users/
starterkit/.chef/../../chef/ca_certs/servv-aqtswxu20swzkjgz.crt
Adding certificate for MyCorp_Root_CA in /Users/users/
starterkit/.chef/../../chef/ca_certs/MyCorp_Root_CA.crt
```

3. `knife ssl check` をもう一度実行します。出力は次のようになります。

```
Connecting to host my-chef-automate-server.my-corp.com:443
      Successfully verified certificates from 'my-chef-automate-server.my-
corp.com'
```

これで、Chef Automate サーバーで `knife` を使用する準備ができました。

Chef サーバーで管理するノードを追加する

⚠ Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[chef-client](#) エージェントは、サーバーに関連付けられている物理コンピュータまたは仮想コンピュータ ([nodes] (ノード) と呼ばれる) で Chef レシピを実行します。オンプレミスのコンピュータまたはインスタンスを Chef サーバーに接続して管理できます (サポートされているオペレーティングシステムがそのノードで実行されている場合)。Chef サーバーにノードを登録すると、そのノードに chef-client エージェントソフトウェアがインストールされます。

次の方法を使用してノードを追加できます。

- このチュートリアルでは、Chef サーバが管理できるように、EC2 インスタンスを追加またはブートストラップする knife コマンドを個別に実行する方法を説明します。詳細については、[ノードを個別に追加します](#)を参照してください。
- スクリプトを使って自動的にノードを追加し、Chefサーバとノードの自動関連付けを行います。[スターターキット](#)のコードでは、ユーザーの介入なしに複数のノードを自動的に追加する方法を示しています。詳細については、「[でノードを自動的に追加する AWS OpsWorks for Chef Automate](#)」を参照してください。

トピック

- [ノードを個別に追加します](#)
- [でノードを自動的に追加する AWS OpsWorks for Chef Automate](#)

ノードを個別に追加します

⚠ Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリュー

シオンに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

この節では、ChefサーバがEC2インスタンスを管理できるようにEC2インスタンスを追加、またはブートストラップするknife コマンドの実行方法を説明します。

AWS OpsWorks for Chef Automate サーバーに関連付けられているノードでサポートされている chef-client の最小バージョンは 13.x です。最新の安定した chef-client バージョンを実行することをお勧めします。

トピック

- [\(オプション\) Chef Automate サーバールート CA の URL を指定します。](#)
- [サポートされるオペレーティングシステム](#)
- [knife を使用してノードを追加する](#)

(オプション) Chef Automate サーバールート CA の URL を指定します。

サーバがカスタムドメインと証明書を使用している場合は、ユーザーデータスクリプトの ROOT_CA_URL 変数を編集して、サーバのルート CA 証明書 (PEM 形式) を取得するために使用できるパブリック URL にする必要があります。次の AWS CLI コマンドは、ルート CA を Amazon S3 バケットにアップロードし、1 時間使用できる署名付き URL を生成します。

1. ルート CA 証明書 (PEM 形式) を S3 にアップロードします。

```
aws s3 cp ROOT_CA_PEM_FILE_PATH s3://bucket_name/
```

2. ルート CA をダウンロードするために 1 時間 (この例では 3600 秒) 使用できる署名付き URL を生成します。

```
aws s3 presign s3://bucket_name/ROOT_CA_PEM_FILE_NAME --expires-in 3600
```

3. 署名付き URL の値を使用してユーザーデータスクリプトの ROOT_CA_URL 変数を編集します。

サポートされるオペレーティングシステム

ノードでサポートされているオペレーティングシステムの最新の一覧については、[\[Chef website\]](#) (Chef のウェブサイト) を参照してください。

knife を使用してノードを追加する

[knife-ec2](#) プラグインは Chef Workstation に含まれています。knife-ec2 に慣れていない場合は、knife bootstrap ではなくそれを使用して、新しい EC2 インスタンスをプロビジョニングおよびブートストラップできます。慣れていない場合は、新しい EC2 インスタンスを起動した後に、このセクションの手順に従います。

管理するノードを追加するには

1. 次の knife bootstrap コマンドを実行します。このコマンドは、Chef サーバーで管理されるノードに EC2 インスタンスをブートストラップします。「[the section called “Policyfile.rb を使用してリモートソースからクックブックを取得する”](#)」でインストールした nginx クックブックからレシピを実行するように、Chef サーバーに指示していることに注意してください。knife bootstrap コマンドを使用したノードの追加に関する詳細については、Chef ドキュメントの「[Bootstrap a Node](#)」(ノードのブートストラップ)を参照してください。

このステップの knife コマンドでノードのオペレーティングシステムに対して有効なユーザー名を次の表に示します。root と ec2-user のどちらでも動作しない場合は、AMI プロバイダーに確認してください。Linux ベースのインスタンスへの接続の詳細については、AWS ドキュメントの「[Connecting to Your Linux Instance Using SSH](#)」(SSH を使用した Linux インスタンスへの接続)を参照してください。

ノードのオペレーティングシステムで有効なユーザー名の値

オペレーティングシステム	有効なユーザー名
Amazon Linux	ec2-user
Red Hat Enterprise Linux 5	root または ec2-user
Ubuntu	ubuntu
Fedora	fedora または ec2-user
SUSE Linux	root または ec2-user

```
knife bootstrap INSTANCE_IP_ADDRESS -N INSTANCE_NAME -x USER_NAME --sudo --run-list "recipe[nginx]"
```

2. 次のコマンド (`[INSTANCE_NAME]` (インスタンス名) は追加したインスタンスの名前に置き換えます) を実行して、新しいノードが追加されていることを確認します。

```
knife client show INSTANCE_NAME
knife node show INSTANCE_NAME
```

でノードを自動的に追加する AWS OpsWorks for Chef Automate

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このトピックでは、Amazon Elastic Compute Cloud (Amazon EC2) ノードを Chef サーバーに自動的に追加する方法を説明します。[スターターキット](#)のコードでは、ユーザーの介入なしに複数のノードを自動的に追加する方法を示しています。ユーザーの介入なしに (自動で) 新しいノードを関連付ける方法としては、[Chef クライアントクックブック](#)を設定することをお勧めします。スターターキットの `userdata` スクリプトを使用し、`run_list` スクリプトの `userdata` セクションを変更することも、または、ノードに適用するクックブックの `Policyfile.rb` を変更することもできます。chef-client エージェントを実行する前に、Chef Client クックブックを Chef サーバーにアップロードし、以下のサンプルコマンドで示しているように (例えば HTTPD ロールを使用して)、chef-client エージェントをサービスモードでインストールします。

```
chef-client -r "chef-client,role[httpd]"
```

Chef サーバーと通信するには、chef-client エージェントソフトウェアは、クライアントノードのパブリックキーへのアクセスを許可されている必要があります。Amazon EC2 でパブリックキーとプライベートキーのペアを生成し、そのパブリックキーを AWS OpsWorks `associate-node` ノード名で API コールに渡すことができます。スターターキットに含まれているスクリプトは、お客様の組織名、サーバー名、サーバーエンドポイントを収集します。これにより、そのノードが Chef サーバーに関連付けられ、プライベートキーを照会した後に、そのノードで実行される chef-client エージェントソフトウェアがサーバーと通信できるようになります。

AWS OpsWorks for Chef Automate サーバーに関連付けられているノードでサポートされている chef-client の最小バージョンは 13.x です。最新の安定した chef-client バージョンを実行することをお勧めします。

ノードの関連付けを解除する方法については、このガイド [AWS OpsWorks for Chef Automate サーバーからノードの関連付けを解除する](#) の「 」および AWS OpsWorks for Chef Automate API [disassociate-node](#) ドキュメントの「 」を参照してください。

トピック

- [サポートされるオペレーティングシステム](#)
- [ステップ 1: インスタンスプロファイルとして使用する IAM ロールを作成します](#)
- [ステップ 2: Chef クライアントクックブックをインストールする](#)
- [ステップ 3: 自動関連付けスクリプトを使用してインスタンスを作成する](#)
- [chef-client の繰り返し実行を自動化する他の方法](#)
- [関連トピック](#)

サポートされるオペレーティングシステム

ノードでサポートされているオペレーティングシステムの最新の一覧については、[\[Chef website\]](#) (Chef のウェブサイト) を参照してください。

ステップ 1: インスタンスプロファイルとして使用する IAM ロールを作成します

EC2 インスタンスプロファイルとして使用する AWS Identity and Access Management (IAM) ロールを作成し、次のポリシーを IAM ロールにアタッチします。このポリシーでは、ノード登録時に AWS OpsWorks for Chef Automate (opsworks-cm) API が EC2 インスタンスと通信することが許可されています。インスタンスプロファイルの詳細については、Amazon EC2 のドキュメントの「[Using Instance Profiles](#) (インスタンスプロファイルの使用)」を参照してください。IAM ロールを作成する方法については、Amazon EC2 のドキュメントの「[Creating an IAM Role in the Console](#)」(コンソールでの IAM ロールの作成) を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "opsworks-cm:AssociateNode",
        "opsworks-cm:DescribeNodeAssociationStatus",
```

```
    ],
    "Resource": "*",
    "Effect": "Allow"
  }
]
}
```

AWS OpsWorks には、前述のポリシーステートメントで IAM ロールを作成するために使用できる AWS CloudFormation テンプレートが用意されています。次の AWS CLI コマンドは、このテンプレートを使用してインスタンスプロファイルロールを作成します。デフォルトのリージョンで新しい AWS CloudFormation スタックを作成する場合は、`--region`パラメータを省略できます。

```
aws cloudformation --region region ID create-stack --stack-name myChefAutomateinstanceprofile --template-url https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-nodes-roles.yaml --capabilities CAPABILITY_IAM
```

ステップ 2: Chef クライアントクックブックをインストールする

まだ実行していない場合は、「[\(代替\) Berkshelf を使用してリモートソースからクックブックを取得する](#)」の手順に従って、Berksfile または Policyfile.rb ファイルレファレンスによって Chef クライアントクックブックが参照され、クックブックがインストールされることを確認します。

ステップ 3: 自動関連付けスクリプトを使用してインスタンスを作成する

1. EC2 インスタンスを作成するには、[Starter Kit から EC2](#) インスタンスの手順、Amazon EC2 Auto Scaling グループの起動設定、または AWS CloudFormation テンプレートの `userdata` セクションに `userdata` スクリプトをコピーします。EC2 ユーザーデータへのスクリプトの追加の詳細については、Amazon EC2 ドキュメントの「[Running Commands on Your Linux Instance at Launch](#)」(Linux インスタンスでの起動時のコマンドの実行) を参照してください。

このスクリプトは、`opsworks-cm` API の [associate-node](#) コマンドを実行して、新しいノードを Chef サーバーに関連付けます。

デフォルトでは、登録された新しいノードの名前はインスタンス ID ですが、`NODE_NAME` スクリプトの `userdata` 変数の値を変更することで、その名前を変更できます。現在、Chef コンソール UI の組織名を変更することはできないため、`CHEF_AUTOMATE_ORGANIZATION` の設定は `default` のままにします。

2. EC2 ドキュメントの[インスタンスの作成](#)の手順に従い、ここで説明する変更を加えます。EC2 インスタンス起動ウィザードで、Amazon Linux AMI を選択します。

3. [Configure Instance Details] ページで IAM ロールとして作成した「[ステップ 1: インスタンスプロファイルとして使用する IAM ロールを作成します](#)」を選択します。
4. [Advanced Details] で、先の手順で作成した `userdata.sh` スクリプトをアップロードします。
5. [Add Storage] ページで必要な変更はありません。[Add Tags] に進みます。
6. [セキュリティグループの設定] ページで、[ルールの追加] を選択し、タイプに [HTTP] を選択してこの例の Apache ウェブサーバーでポート番号 443 と 80 を開きます。
7. Review and Launch (確認と作成) を選択してから、Launch (起動) を選択します。新しいノードを開始すると、そのノードは `RUN_LIST` パラメータで渡されたレシピによって指定された設定を適用します。
8. オプション: 実行リストに `nginx` クックブックを追加した場合、新しいノードのパブリック DNS にリンクしたウェブページを開くと、`nginx` ウェブサーバーがホストするウェブサイトが表示されます。

chef-client の繰り返し実行を自動化する他の方法

達成はより難しく、推奨されませんが、このトピックのスクリプトはスタンドアロンインスタンスのユーザーデータの一部としてのみ実行したり、AWS CloudFormation テンプレートを使用して新しいインスタンスのユーザーデータに追加したり、スクリプトを定期的に実行するように cron ジョブを設定したり、サービス `chef-client` 内で実行したりできます。ただし、他の自動化の方法にはいくつかの欠点があるため、Chef クライアントクックブックの方法をお勧めします。

`chef-client` に指定できるパラメータの詳細なリストについては、[Chef のドキュメント](#) を参照してください。

関連トピック

次の AWS ブログ記事では、Auto Scaling グループを使用するか、複数のアカウント内でノードを Chef Automate サーバーに自動的に関連付ける方法について詳しく説明します。

- [Chef Automate OpsWorks での AWS を使用した Auto Scaling による EC2 インスタンスの管理](#)
- [OpsWorks Chef Automate の - 異なるアカウントでノードを自動的にブートストラップする](#)

Chef Automate ダッシュボードにサインインする

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Chef サーバーの [Properties] ページでサインイン認証情報をダウンロードし、サーバーがオンラインになったら、Chef Automate ダッシュボードにサインインします。このワークスルーでは、最初にクックブックをアップロードし、管理するノードを 1 つ以上追加します。そうすることによって、クックブックおよびノードに関する情報をダッシュボードで表示できるようになります。

ダッシュボードのウェブページに接続しようとする、Chef サーバーの管理に使用しているクライアントコンピュータに AWS OpsWorks 固有の CA 署名付き SSL 証明書をインストールするまで、証明書の警告がブラウザに表示されます。ダッシュボードのウェブページに進む前に警告が表示されないようにするには、サインインする前に SSL 証明書をインストールします。

AWS OpsWorks SSL 証明書をインストールするには


- システムに適合する証明書を選択します。
- Linux または MacOS ベースのシステムの場合は、次の Amazon S3 の場所から PEM ファイル名拡張子を持つファイルをダウンロードします: <https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-ca-2016-root.pem>。

Note

さらに、<https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-ca-2020-root.pem> から新しい PEM ファイルをダウンロードします。AWS OpsWorks for Chef Automate は現在ルート証明書を更新しているため、古い証明書と新しい証明書の両方を信頼する必要があります。

MacOS で SSL 証明書を管理する方法の詳細については、アップルの Support ウェブサイトの「[Macのキーチェーンアクセスで証明書についての情報を取得する](#)」を参照してください。


- Windows ベースのシステムの場合は、PP7Bファイル名拡張子の ファイルを Amazon S3 の場所 <https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-ca-2016-root.p7b> からダウンロードします。

 Note

さらに、<https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-ca-2020-root.p7b> から新しい P7B ファイルをダウンロードします。AWS OpsWorks for Chef Automate は現在ルート証明書を更新しているため、古い証明書と新しい証明書の両方を信頼する必要があります。

Windows に SSL 証明書をインストールする方法の詳細については、「Microsoft で [信頼できるルート証明書を管理する](#)」を参照してください TechNet。

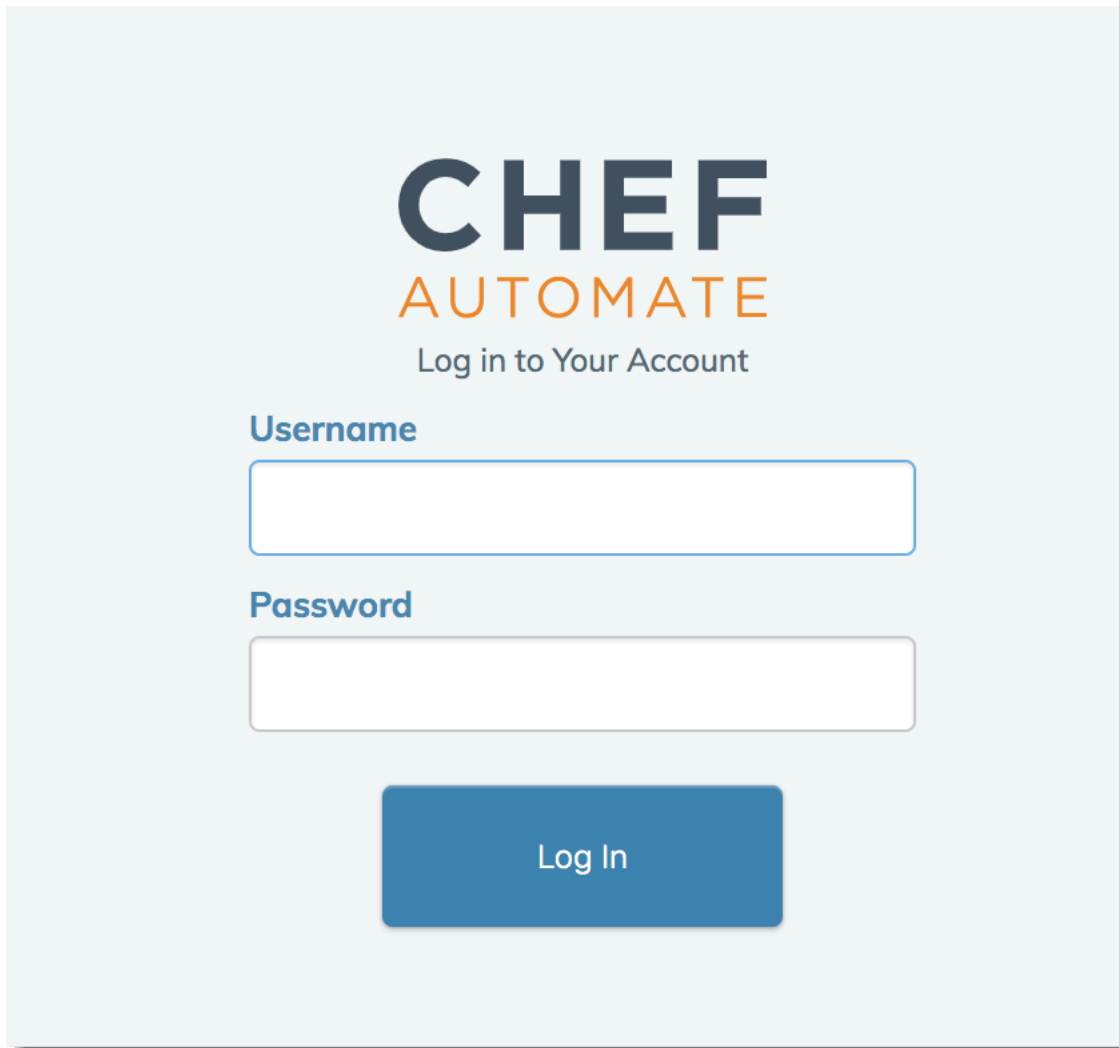
クライアント側 SSL 証明書をインストールした後は、Chef Automate ダッシュボードへのサインイン時に警告メッセージが表示されなくなります。

 Note

Ubuntu および Linux オペレーティングシステムで Google Chrome を使用しているユーザーは、サインインで問題が発生する場合があります。Mozilla Firefox または他のブラウザを使用してサインインし、これらのオペレーティングシステムで Chef Automate ダッシュボードを使用することをお勧めします。Windows または MacOS では Google Chrome の使用で問題は出ていません。

Chef Automate ダッシュボードにサインインするには

1. 「[前提条件](#)」でダウンロードした Chef Automate 認証情報を解凍して開きます。サインインするには、これらの認証情報が必要です。
2. Chef サーバーの [Properties] (プロパティ) ページを開きます。
3. [Properties] (プロパティ) ページの右上にある [Open Chef Automate dashboard] を選択します。
4. ステップ 1 で認証情報を使用してサインインします。



CHEF
AUTOMATE

Log in to Your Account

Username

Password

Log In

5. Chef Automate ダッシュボードでは、ブートストラップしたノード、クックブックの実行の進行状況とイベント、ノードのコンプライアンスレベル、などに関する詳細情報を表示できます。Chef Automate ダッシュボードの機能およびその使用方法の詳細については、[\[Chef Automate Documentation\]](#) (Chef Automate ドキュメント) を参照してください。

Note

Chef Automate ダッシュボードへのサインインに使用するパスワードを変更する方法については、「[Chef Automate ダッシュボードの認証情報のリセット](#)」を参照してください。

を使用して AWS OpsWorks for Chef Automate サーバーを作成する AWS CloudFormation

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks for Chef Automate では、で [Chef Automate](#) サーバーを実行できます AWS。約 15 分で Chef Automate サーバーをプロビジョニングできます。

2021 年 5 月 3 日以降、は一部の Chef Automate サーバー属性を に AWS OpsWorks for Chef Automate 保存します AWS Secrets Manager。詳細については、「[との統合 AWS Secrets Manager](#)」を参照してください。

以下のチュートリアルは、でスタックを作成 AWS OpsWorks for Chef Automate することで、でサーバーを作成するのに役立ちます AWS CloudFormation。

トピック

- [前提条件](#)
- [AWS CloudFormationで Chef Automate サーバーを作成する](#)

前提条件

新しい Chef Automate サーバーを作成する前に、Chef サーバーにアクセスして管理するために必要なリソースを AWS OpsWorks for Chef Automate の外部で作成してください。詳細については、このガイドの「はじめる」セクションにある「[前提条件](#)」を参照してください。

サーバーの作成に使用するテンプレートでサポートされる値と必要な値については、AWS CloudFormation 「[ユーザーガイドテンプレートリファレンス](#)」の [OpsWorks 「-CM」セクション](#) を参照してください。AWS CloudFormation

カスタムドメインを使用するサーバーを作成する場合は、カスタムドメイン、証明書、およびプライベートキーが必要です。AWS CloudFormation テンプレートでは、これら 3 つのパラメータすべてに値を指定する必要があります。CustomDomain、および CustomPrivateKey パラメータの要件の詳細については CustomCertificate、AWS OpsWorks CM API リファレンスの [CreateServer](#) 「」を参照してください。

CHEF_AUTOMATE_ADMIN_PASSWORD エンジン属性のパスワード値を作成します。パスワードの最小の長さは 8 文字、最大は 32 文字です。パスワードには、文字、数字、および特殊文字 ((!/@#\$%^+=_)) を使用できます。パスワードは、少なくとも 1 つの小文字、1 つの大文字、1 つの数字、および 1 つの特殊文字を含む必要があります。このパスワードは、AWS CloudFormation テンプレートで指定するか、スタックの作成時に CHEF_AUTOMATE_ADMIN_PASSWORD パラメータの値として指定します。

で Chef Automate サーバーの作成を開始する前に、base64 でエンコードされた RSA キーペアを生成します AWS CloudFormation。ペアのパブリックキーは CHEF_AUTOMATE_PIVOTAL_KEY、

[CreateServer](#) API [EngineAttributes](#)から Chef 固有の の値です。このキーは、AWS CloudFormation コンソールの Parameters の値、または の create-stack コマンドの値として提供されます AWS CLI。このキーを生成するには、次の方法をお勧めします。

- Linux ベースのコンピュータで、次の [OpenSSL](#) コマンドを実行してこのキーを生成できます。

```
openssl genrsa -out pivotal_key_file_name.pem 2048
```

生成したら、鍵ペアの RSA 公開鍵部分をファイルにエクスポートします。公開鍵は、CHEF_AUTOMATE_PIVOTAL_KEY の値となります。

```
openssl rsa -in pivotal_key_file_name.pem -pubout -out public.pem -outform PEM
```

- Windows ベースのコンピュータでは、PuTTYgen ユーティリティを使用して base64 でエンコードされた RSA キーペアを生成できます。詳細については、SSH.com で「[PuTTYgen - Key Generator for PuTTY on Windows](#)」を参照してください。

AWS CloudFormationで Chef Automate サーバーを作成する

このセクションでは、AWS CloudFormation テンプレートを使用して AWS OpsWorks for Chef Automate サーバーを作成するスタックを構築する方法について説明します。これを行うには、AWS CloudFormation コンソールまたは を使用します AWS CLI。 [サンプル AWS CloudFormation テンプレート](#)を使用して、AWS OpsWorks for Chef Automate サーバースタックを構築できます。サンプルテンプレートを更新するには、必ず、独自のサーバー名、IAM ロール、インスタンスプロファイル、サーバーの説明、バックアップ保持数、メンテナンスオプション、およびオプションのタグを使用してください。サーバーがカスタムドメインを使用する場合は、AWS CloudFormation テンプレートで CustomDomain、CustomCertificate、および CustomPrivateKeyパラメータの値を指定する必要があります。AWS CloudFormation テンプレートで CHEF_AUTOMATE_ADMIN_PASSWORDおよび CHEF_AUTOMATE_PIVOTAL_KEY エンジン属性とその値を指定するか、属性のみを指定し、AWS CloudFormation スタックの作成ウィザードまたはcreate-stackコマンドで属性の値を指定できます。これらの属性の詳細については、このガイドの「はじめに」セクションの「[the section called “で Chef Automate サーバーを作成する AWS Management Console”](#)」を参照してください。

トピック

- [AWS CloudFormation \(コンソール\) を使用して Chef Automate サーバーを作成する](#)
- [AWS CloudFormation \(CLI\) を使用して Chef Automate サーバーを作成する](#)

AWS CloudFormation (コンソール) を使用して Chef Automate サーバーを作成する

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/cloudformation> で AWS CloudFormation コンソールを開きます。
2. AWS CloudFormation ホームページで、スタックの作成 を選択します。
3. 前提条件 - テンプレートを準備する で、[サンプル AWS CloudFormation テンプレート](#) を使用している場合は、テンプレートの準備ができました を選択します。
4. [Specify template] 内で、テンプレートのソースを選択します。このチュートリアルでは、テンプレートファイル をアップロードを選択し、Chef Automate サーバーを作成する AWS CloudFormation テンプレートをアップロードします。テンプレートファイルを参照し、[Next] を選択します。

AWS CloudFormation テンプレートは YAML 形式または JSON 形式のいずれかです。[サンプル AWS CloudFormation テンプレート](#)を使用できます。サンプル値は必ず独自の値に置き換えてください。AWS CloudFormation テンプレートデザイナーを使用して、新しいテンプレートを構築したり、既存のテンプレートを検証したりできます。これを行う方法については、AWS CloudFormation ユーザーガイドの[AWS CloudFormation 「Designer Interface Overview」](#) (Designer インターフェイスの概要) を参照してください。

Create stack

Prerequisite - Prepare template

Prepare template

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

 Template is ready

 Use a sample template

 Create template in Designer

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source

Selecting a template generates an Amazon S3 URL where it will be stored.

 Amazon S3 URL

 Upload a template file

Upload a template file

opsworkscm-server.json

JSON or YAML formatted file

S3 URL: [https://s3-external-1.amazonaws.com/cf-templates-
-opsworkscm-server.json](https://s3-external-1.amazonaws.com/cf-templates-
-opsworkscm-server.json)

5. [Specify stack details] ページでスタックの名前を入力します。これは、サーバーの名前と同じ名前にしないでください。スタック名にすぎません。[Parameters (パラメータ)] 領域に、「[the section called “前提条件”](#)」で作成した値を貼り付けます。[Password] にパスワードを入力します。

RSA キーファイルの内容を に貼り付けますPivotalKey。次のスクリーンショットに示すように、AWS CloudFormation コンソールでは、ピボットキー値の各行の末尾に改行 (\n) 文字を追加する必要があります。[次へ] をクリックします。

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Password

PivotalKey

-----BEGIN PUBLIC KEY-----\n EXAMPLENBgqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAt8riKI+M/USa8EXAMPLE\n EXAMPLEERk3H+QM6+7s6IYRC

6. [Configure stack options] (スタックオプションの設定) ページでは、スタックを使用して作成するサーバーにタグを追加することができ、テンプレートで使用する IAM ロールをまだ指定していない場合には、リソースを作成するための IAM ロールを選択できます。オプションの指定を終了したら、[Next (次へ)] を選択します。ロールバックトリガーなどの高度なオプションの詳細については、「[AWS CloudFormation ユーザーガイド](#)」の [AWS CloudFormation 「スタックオプションの設定」](#) を参照してください。
7. [確認] ページで選択内容を確認します。サーバースタックを作成する準備ができたなら、[Create stack] を選択します。

がスタックを作成する AWS CloudFormation のを待っている間に、スタックの作成ステータスを表示します。スタックの作成に失敗した場合、コンソールに表示されるエラーメッセージを確認し、問題を解決します。AWS CloudFormation スタックのエラーのトラブルシューティングについての詳細は、AWS CloudFormation ユーザーガイドの「[Troubleshooting Errors](#)」(エラーのトラブルシューティング) を参照してください。

サーバーの作成が完了すると、AWS OpsWorks for Chef Automate のホームページで、Chef Automate サーバーが [online] のステータスになり、利用可能になります。サーバーの [Properties (プロパティ)] ページで、新しいスターターキットと Chef Automate ダッシュボードの認証情報を生成します。サーバーがオンラインになると、Chef Automate ダッシュボードがサーバーのドメインで `https://your_server_name-randomID.region.opsworks-cm.io` の形式の URL で利用できます。

Note

サーバーのカスタムドメイン、証明書、プライベートキーを指定した場合は、エンタープライズの DNS 管理ツールで CNAME エントリを作成し、カスタムドメインをサーバー用に AWS OpsWorks for Chef Automate が自動的に生成したエンドポイントにマッピングします。生成されたエンドポイントのカスタムドメイン値にマッピングするまで、サーバーを管理したり、サーバーの Chef Automate ダッシュボードに接続したりできません。

生成されたエンドポイント値を取得するには、サーバーがオンラインになった後に次の AWS CLI コマンドを実行します。

```
aws opsworks describe-servers --server-name server_name
```

AWS CloudFormation (CLI) を使用して Chef Automate サーバーを作成する

ローカルコンピュータでまだ `aws cloudformation create-stack` を実行していない場合は AWS CLI、AWS コマンドラインインターフェイスユーザーガイド [AWS CLI のインストール手順に従って](#) をダウンロードしてインストールします。このセクションでは、`create-stack` コマンドで使用できるパラメータのすべては説明しません。`create-stack` パラメータの詳細については、「[create-stack リファレンス](#)」の「AWS CLI」を参照してください。

1. 必ず [\[前提条件\]](#) を実行して AWS OpsWorks for Chef Automate サーバーを作成してください。
2. サービスロールとインスタンスプロファイルを作成します。は、両方を作成するために使用できる AWS CloudFormation テンプレート `OpsWorksCMRoles` を提供します。次の AWS CLI コマンドを実行して、サービスロールとインスタンスプロファイルを作成する AWS CloudFormation スタックを作成します。

```
aws cloudformation create-stack --stack-name OpsWorksCMRoles --template-url
https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-
cm-roles.yaml --capabilities CAPABILITY_NAMED_IAM
```

がスタックの作成 AWS CloudFormation を完了したら、アカウント内のサービスロールARNs を検索してコピーします。

```
aws iam list-roles --path-prefix "/service-role/" --no-paginate
```

`list-roles` コマンドの結果内で、次のようなサービスロールとインスタンスプロファイルのエントリを探します。サービスロールとインスタンスプロファイルの ARNs を書き留めて、サーバースタックの作成に使用している AWS CloudFormation テンプレートに追加します。

```
{
  "AssumeRolePolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "ec2.amazonaws.com"
        }
      }
    ]
  }
}
```



```

    },
    "RoleId": "AROZZZZZZZZZZQ6R22HC",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-ec2-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
  },
  {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Action": "sts:AssumeRole",
          "Effect": "Allow",
          "Principal": {
            "Service": "opsworks-cm.amazonaws.com"
          }
        }
      ]
    },
    "RoleId": "AROZZZZZZZZZZZZZZ6QE",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-service-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-
role"
  }
}

```

3. create-stack コマンドを再度実行して AWS OpsWorks for Chef Automate サーバーを作成します。

- *stack_name* をスタックの名前に置き換えます。これは、Chef Automate サーバーではなく、AWS CloudFormation スタックの名前です。Chef Automate サーバー名は、AWS CloudFormation テンプレート `ServerName` 内の `値` です。
- *template* をテンプレートファイルへのパスに置き換え、*yaml* `###` *json* を必要に応じて `.yaml` または `.json` に置き換えます。
- `値` は [CreateServer](#) API [EngineAttributes](#) から `--parameters` に対応します。Chef の場合、サーバーを作成するためにユーザーが指定するエンジン属性は、`CHEF_AUTOMATE_PIVOTAL_KEY` (「[the section called “前提条件”](#)」で説明されているユーティリティを使用して生成する base64 でエンコードされた RSA パブリックキー) と `CHEF_AUTOMATE_ADMIN_PASSWORD` (ユーザーが作成する 8~32 文字のパスワード) で

す。CHEF_AUTOMATE_ADMIN_PASSWORD の詳細については、「[を使用して Chef Automate サーバーを作成する AWS CLI](#)」を参照してください。例に示すように、PivotalKey パラメータの値として重要なキーが含まれている PEM ファイルへのポインターを指定できます。CHEF_AUTOMATE_ADMIN_PASSWORD との値がテンプレートで指定されていない場合 CHEF_AUTOMATE_PIVOTAL_KEY は、AWS CLI コマンドで値を指定する必要があります。

```
aws cloudformation create-stack --stack-name stack_name
--template-body file://template.yaml or json --parameters
ParameterKey=PivotalKey,ParameterValue="base64_encoded_RSA_public_key_value"
```

CHEF_AUTOMATE_ADMIN_PASSWORD 属性と CHEF_AUTOMATE_PIVOTAL_KEY 属性のサンプル値が含まれている例を以下に示します。AWS CloudFormation テンプレートでこれらの属性の値を指定しなかった場合は、同様のコマンドを実行します。

```
aws cloudformation create-stack --stack-name "OpsWorksCMChefServerStack"
--template-body file://opsworkscm-server.yaml --parameters
ParameterKey=PivotalKey,ParameterValue="$(openssl rsa -in "pivotalKey.pem" -
pubout)" ParameterKey=Password,ParameterValue="SuPer\secret890"
```

- スタックの作成が完了したら、AWS OpsWorks for Chef Automate コンソールで新しいサーバーのプロパティページを開き、スターターキットをダウンロードします。新しいスターターキットをダウンロードすると、Chef Automate ダッシュボードの管理者パスワードがリセットされます。
- サーバーでカスタムドメイン、証明書、およびプライベートキーを使用する場合は、[\(オプション\) カスタムドメインを使用するように knife を設定します](#) に knife.rb を設定のステップに従い、ステップ 7 に進みます。

カスタムドメインを使用していない場合は、次の Amazon S3 バケットの場所からルート認証局 (CA) 証明書をダウンロードします: <https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-ca-2020-root.pem>。証明書ファイルを安全かつ便利な場所に保存します。この証明書は、次のステップで knife.rb を設定するために必要です。

- 新しいサーバーで knife コマンドを使用するには、Chef knife.rb 設定ファイルの設定を更新します。サンプル knife.rb ファイルは、スターターキットに含まれています。次の例は、カスタムドメインを使用しないサーバーで knife.rb を設定する方法を示しています。カスタムドメインを使用している場合は、knife 設定手順の [\(オプション\) カスタムドメインを使用するように knife を設定します](#) を参照してください。

- **ENDPOINT** をサーバーのエンドポイント値で置き換えます。これは、スタック作成オペレーションの出力の一部です。エンドポイントを取得するには、次のコマンドを実行します。

```
aws cloudformation describe-stacks --stack-name stack_name
```

- `client_key` の設定の `key_pair_file.pem` を、サーバーの作成に使用した `CHEF_AUTOMATE_PIVOTAL_KEY` を含む PEM ファイルの名前に置き換えます。

```
base_dir = File.join(File.dirname(File.expand_path(__FILE__)), '..')

log_level           :info
log_location       STDOUT
node_name          'pivotal'
client_key          File.join(base_dir, '.chef', 'key_pair_file.pem')
syntax_check_cache_path File.join(base_dir, '.chef', 'syntax_check_cache')
cookbook_path       [File.join(base_dir, 'cookbooks')]

chef_server_url     'ENDPOINT/organizations/default'
ssl_ca_file         File.join(base_dir, '.chef', 'ca_certs', 'opsworks-cm-
ca-2020-root.pem')
trusted_certs_dir   File.join(base_dir, '.chef', 'ca_certs')
```

7. サーバーの作成プロセスが完了したら、「[the section called “設定を完了してクックブックをアップロードする”](#)」に進みます。スタックの作成に失敗した場合、コンソールに表示されるエラーメッセージを確認し、問題を解決します。AWS CloudFormation スタックのエラーのトラブルシューティングの詳細については、「AWS CloudFormation ユーザーガイド」の「[エラーのトラブルシューティング](#)」を参照してください。

カスタムドメインを使用するように AWS OpsWorks for Chef Automate サーバーを更新する

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、サーバーのバックアップを使用して新しい AWS OpsWorks for Chef Automate サーバーを作成することで、カスタムドメインと証明書を使用するように既存のサーバーを更新する方法について説明します。基本的に、バックアップから新しいサーバーを作成し、カスタムドメイン、証明書、プライベートキーを使用するように新しいサーバーを設定することで、既存の AWS OpsWorks for Chef Automate 2.0 サーバーをコピーします。

トピック

- [前提条件](#)
- [制限事項](#)
- [カスタムドメインを使用するようにサーバーを更新する](#)
- [以下の資料も参照してください。](#)

前提条件

以下は、カスタムドメインと証明書を使用するように既存の AWS OpsWorks for Chef Automate サーバーを更新するための要件です。

- 更新 (またはコピー) するサーバーは、Chef Automate 2.0 を実行している必要があります。
- 新しいサーバーの作成に使用するバックアップを決定します。更新するサーバーのバックアップが少なくとも 1 つ必要です。でのバックアップの詳細については、AWS OpsWorks for Chef Automate 「」を参照してください [AWS OpsWorks for Chef Automate サーバーのバックアップ](#)。
- バックアップのソースである既存のサーバーを作成するために使用したサービスロールとインスタンスプロファイルの ARN を準備します。
- 最新リリースの AWS CLI を実行していることを確認してください。AWS CLI ツールの更新の詳細については、AWS コマンドラインインターフェイスユーザーガイドの「[のインストール AWS CLI](#)」を参照してください。

制限事項

バックアップから新しいサーバーを作成して既存のサーバーを更新する場合、新しいサーバーを既存の AWS OpsWorks for Chef Automate サーバーとまったく同じにすることはできません。

- この手順は、AWS CLI または [AWS SDKs のいずれかを使用してのみ実行できます](#)。AWS Management Console を使用してバックアップから新しいサーバーを作成することはできません。

- 新しいサーバーに、アカウント内および AWS リージョン内の既存のサーバーと同じ名前は使用できません。名前は、バックアップのソースとして使用した既存のサーバーとは異なる必要があります。
- 既存のサーバーに接続されたノードは、新しいサーバーによって管理されません。次のいずれかを行う必要があります。
 - 複数の Chef Automate サーバーでノードを管理することはできないため、異なるノードをアタッチします。
 - 既存のサーバー (バックアップのソース) から新しいサーバーと新しいカスタムドメインエンドポイントにノードを移行します。ノードの移行方法の詳細については、Chef のドキュメントの「」を参照してください。

カスタムドメインを使用するようにサーバーを更新する

既存の Chef Automate 2.0 サーバーを更新するには、バックアップ、カスタムドメイン、カスタム証明書、カスタムプライベートキーを指定するパラメータを追加して `create-server` コマンドを実行し、コピーを作成します。

1. `create-server` コマンドで指定できるサービスロールまたはインスタンスプロファイルの ARN がない場合は、「[を使用して Chef Automate サーバーを作成する AWS CLI](#)」のステップ 1 ~ 5 に従って、使用できるサービスロールとインスタンスプロファイルを作成します。
2. まだ作成していない場合は、カスタムドメインで新しいサーバーのベースにする既存の Chef Automate 2.0 サーバーのバックアップを探します。次のコマンドを実行して、アカウントとリージョン内のすべての AWS OpsWorks for Chef Automate バックアップに関する情報を表示します。使用するバックアップの ID を書き留めておきます。

```
aws opsworks-cm --region region name describe-backups
```

3. `create-server` コマンドを実行して AWS OpsWorks for Chef Automate サーバーを作成します。
 - `--engine` の値は、`ChefAutomate`、`--engine-model` は、`Single`、`--engine-version` は 12 です。
 - サーバー名は、AWS アカウント内で各リージョン内で一意である必要があります。サーバー名は文字で始める必要があります。その後は文字、数字、またはハイフン (-) を最大 40 文字まで使用できます。
 - ステップ 1 のインスタンスプロファイル ARN とサービスロール ARN を使用します。

- 有効なインスタンスタイプは `m5.large`、`r5.xlarge`、または `r5.2xlarge` です。これらのインスタンスタイプの仕様の詳細については、「Amazon EC2 ユーザーガイド」の「[インスタンスタイプ](#)」を参照してください。
- `--engine-attributes` パラメータはオプションです。いずれかまたは両方の値を指定しない場合、サーバー作成プロセスで値が生成されます。`--engine-attributes` を追加する場合は、ステップ 2 で生成した `CHEF_AUTOMATE_PIVOTAL_KEY` 値、`CHEF_AUTOMATE_ADMIN_PASSWORD`、またはその両方を指定します。

`CHEF_AUTOMATE_ADMIN_PASSWORD` の値を設定しない場合、`create-server` レスポンスの一部としてパスワードが生成され返されます。コンソールでスターターキットをもう一度ダウンロードして、このパスワードを再生成することもできます。パスワードの最小の長さは 8 文字、最大は 32 文字です。パスワードには、文字、数字、および特殊文字 (!/@#\$\$%^+=_) を使用できます。パスワードは、少なくとも 1 つの小文字、1 つの大文字、1 つの数字、および 1 つの特殊文字を含む必要があります。

- SSH キーペアはオプションですが、Chef Automate ダッシュボードの管理者パスワードをリセットする必要がある場合に Chef Automate サーバーに接続することができます。SSH キーペアの作成の詳細については、「Amazon EC2 User Guide」(Amazon EC2 ユーザーガイド)の「[Amazon EC2 Key Pairs](#)」(Amazon EC2のキーペア)を参照してください。
- カスタムドメインを使用するには、コマンドに以下のパラメータを追加します。それ以外の場合は、Chef Automate サーバー作成プロセスによって自動的にエンドポイントが生成されます。カスタムドメインを構成するには、3 つのパラメータすべてが必要です。これらのパラメータを使用するためのその他の要件については、AWS OpsWorks CM API リファレンス [CreateServer](#) の「」を参照してください。
 - `--custom-domain` - サーバーのオプションのパブリックエンドポイント (`https://aws.my-company.com` など)。
 - `--custom-certificate` - PEM 形式の HTTPS 証明書。値には、単一の自己署名証明書、または証明書チェーンを指定できます。
 - `--custom-private-key` - HTTPS を使用してサーバーに接続するための PEM 形式のプライベートキー。プライベートキーは暗号化しないでください。パスワードやパスフレーズで保護することはできません。
- 週 1 回のシステムメンテナンスが必要です。次の形式で有効な値を指定する必要があります: `DDD:HH:MM`。指定時刻は協定世界時 (UTC) です。`--preferred-maintenance-window` の値を指定しない場合、火曜日、水曜日、または金曜日の 1 時間がランダムでデフォルト値になります。

- `--preferred-backup-window` の有効な値は次の形式のいずれかで指定する必要があります: 日次バックアップの場合は HH:MM、週次バックアップの場合は DDD:HH:MM。指定時刻は UTC です。デフォルト値は日次で開始時間はランダムです。自動バックアップを無効にするには、代わりにパラメータ `--disable-automated-backup` を追加します。
- `--security-group-ids` に、1 つ以上のセキュリティグループ ID をスペースで区切って入力します。
- `--subnet-ids` には、サブネット ID を入力します。
- `--backup-id` には、ステップ 2 でコピーしたバックアップの ID を入力します。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single"
--engine-version "12" --server-name "server_name" --instance-profile-arn
"instance_profile_ARN" --instance-type "instance_type" --engine-attributes
'{"CHEF_AUTOMATE_PIVOTAL_KEY":"pivotal_key","CHEF_AUTOMATE_ADMIN_PASSWORD":"password"}'
--key-pair "key_pair_name" --preferred-maintenance-window
"ddd:hh:mm" --preferred-backup-window "ddd:hh:mm" --security-group-
ids security_group_id1 security_group_id2 --service-role-arn "service_role_ARN" --
subnet-ids subnet_ID --backup-id backup_ID
```

次の例では、カスタムドメインを使用する Chef Automate サーバーを作成します。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single" --
engine-version "12" \
  --server-name "my-custom-domain-server" \
  --instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-opsworks-
cm-ec2-role" \
  --instance-type "m5.large" \
  --engine-attributes
'{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLyXSzF
\
  --custom-domain "my-chef-automate-server.my-corp.com" \
  --custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
CERTIFICATE-----" \
  --custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
RSA PRIVATE KEY-----" \
  --key-pair "amazon-test" \
  --preferred-maintenance-window "Mon:08:00" \
  --preferred-backup-window "Sun:02:00" \
  --security-group-ids sg-b00000001 sg-b00000008 \
```

```
--service-role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-  
service-role" \  
--subnet-ids subnet-300aaa00 \  
--backup-id MyChefServer-20191004122143125
```

4. AWS OpsWorks for Chef Automate 新しいサーバーの作成には約 15 分かかります。create-server コマンドの出力で、Endpoint 属性の値をコピーします。次に例を示します。

```
"Endpoint": "automate-07-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
```

create-server コマンドの出力を閉じたり、シェルセッションを閉じたりしないでください。今後表示されない重要な情報が出力に含まれている場合があります。create-server の結果からパスワードとスターターキットを取得するには、次のステップに進みます。

5. デキーとパスワード AWS OpsWorks for Chef Automate を生成することを選択した場合は、jq などの JSON プロセッサを使用して、create-server 結果から使用可能な形式で抽出できます。jq をインストールした後、以下のコマンドを実行して中枢キー、Chef Automate ダッシュボード管理者パスワード、およびスターターキットを抽出できます。ステップ 3 で独自の中枢キーとパスワードを指定しなかった場合は、抽出した中枢キーと管理者パスワードを使いやすく安全な場所に保存してください。

```
#Get the Chef password:  
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==  
"CHEF_AUTOMATE_ADMIN_PASSWORD") | .Value'  
  
#Get the Chef Pivotal Key:  
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==  
"CHEF_AUTOMATE_PIVOTAL_KEY") | .Value'  
  
#Get the Chef Starter Kit:  
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==  
"CHEF_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
```

6. オプションで、create-server コマンド結果からスターターキットを抽出しなかった場合は、AWS OpsWorks for Chef Automate コンソールのサーバーのプロパティページから新しいスターターキットをダウンロードできます。新しいスターターキットをダウンロードすると、Chef Automate ダッシュボードの管理者パスワードがリセットされます。
7. エンタープライズの DNS 管理ツールで CNAME エントリを作成して、ステップ 4 でコピーした AWS OpsWorks for Chef Automate エンドポイントにカスタムドメインをポイントします。このステップを完了するまで、サーバーにアクセスしたりサインインしたりすることはできません。

8. サーバーの作成プロセスが完了したら、「[the section called “設定を完了してクックブックをアップロードする”](#)」に進みます。

以下の資料も参照してください。

- [を使用して Chef Automate サーバーを作成する AWS CLI](#)
- [バックアップから AWS OpsWorks for Chef Automate サーバーを復元する](#)
- [CreateServer](#) AWS OpsWorks CM API リファレンスの
- [create-server](#) 「[コマンドリファレンス](#)」の「AWS CLI」を参照してください。

AWS OpsWorks for Chef Automate サーバーのスターターキットを再生成する

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

のスターターキットには、例を含む README ファイル、knife.rb 設定ファイル、プライマリユーザーまたはピボットユーザーのプライベートキー AWS OpsWorks for Chef Automate が含まれています。スターターキットをダウンロードするごとに、新しいキーペアが生成され、古いキーはリセットされます。AWS OpsWorks for Chef Automate サーバーのスターターキットは、次の 2 つの方法のいずれかで再生成できます。

- AWS OpsWorks コンソールで、AWS OpsWorks for Chef Automate サーバーの詳細ページにあるアクションメニュー。古いピボットキーを再生成してリセットするかどうかを確認するよう求められます。
- `aws opsworks cm create-server` コマンドを実行する AWS CLI。

スターターキットの使用の詳細については、「[スターターキットを使用して Chef サーバーを設定する](#)」を参照してください。

を使用して AWS OpsWorks for Chef Automate スターターキットを再生成する AWS CLI

Note

スターターキットを再生成するときに、Chef Automate サーバーの認証キーペアも再生成してリセットし、現在のキーペアを削除します。

[update-server-engine-attributes](#) コマンドを実行して、スターターキットを再生成します。AWS CLI セッションで、次のコマンドを実行します。サーバ名を `--server-name` の値として指定します。CHEF_AUTOMATE_PIVOTAL_KEY の値として自分の公開鍵を設定するには、`--attribute-value` でパブリックキーの値を指定します。それ以外の場合は、null に `--attribute-value` が設定されます。

```
aws opsworks-cm update-server-engine-attributes \  
  --server-name server_name \  
  --attribute-name "CHEF_AUTOMATE_PIVOTAL_KEY" \  
  --attribute-value your_public_key
```

次のコマンドは、サーバーの管理者が使用する公開キーの値を指定する例です。

```
aws opsworks-cm update-server-engine-attributes \  
  --server-name your-test-server \  
  --attribute-name "CHEF_AUTOMATE_PIVOTAL_KEY" \  
  --attribute-value "-----BEGIN PUBLIC KEY-----ExamplePublicKey-----END PUBLIC  
KEY-----"
```

次のコマンドは、`パブリックキー` AWS OpsWorks for Chef Automate を再生成する例です。

```
aws opsworks-cm update-server-engine-attributes \  
  --server-name your-test-server \  
  --attribute-name "CHEF_AUTOMATE_PIVOTAL_KEY" \  
  --attribute-value null
```

このコマンドの出力は、サーバに関する情報と base64 でエンコードされた ZIP ファイルです。ZIP ファイルには、README、設定ファイル、および必要な RSA プライベートキーを含む Chef スターターキットが含まれています。このファイルを保存して解凍し、ファイルの内容を解凍したディレクトリに移動します。このディレクトリから、knife コマンドを実行することができます。

AWS OpsWorks for Chef Automate リソースでのタグの使用

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

タグとは、AWS リソースを識別および整理するためのメタデータとして使用される単語やフレーズのことです。では AWS OpsWorks for Chef Automate、リソースには最大 50 個のユーザー適用タグを含めることができます。各タグはキーと 1 つのオプションの値で構成されます。AWS OpsWorks for Chef Automate で以下のリソースにタグを適用できます。

- AWS OpsWorks for Chef Automate サーバー
- AWS OpsWorks for Chef Automate サーバーのバックアップ

AWS リソースのタグは、コストの追跡、リソースへのアクセスの制御、タスクを自動化するためのリソースのグループ化、目的またはライフサイクルステージによるリソースの整理に役立ちます。タグのメリットの詳細については、AWS Billing and Cost Management ユーザーガイドの[AWS タグ付け戦略](#) AWS Answers と [コスト配分タグの使用](#) を参照してください。

タグを使用して AWS OpsWorks for Chef Automate サーバーまたはバックアップへのアクセスを制御するには、AWS Identity and Access Management (IAM) でポリシーステートメントを作成または編集します。詳細については、AWS Identity and Access Management ユーザーガイドの「[Controlling Access to AWS Resources Using Resource Tags](#)」(リソースタグを使用した リソースへのアクセスの制御) を参照してください。

AWS OpsWorks for Chef Automate サーバーにタグを適用すると、タグはサーバーのバックアップ、バックアップを保存する Amazon S3 バケット、サーバーの Amazon EC2 インスタンス、に保存されているサーバーのシークレット AWS Secrets Manager、およびサーバーで使用される Elastic IP アドレスにも適用されます。タグは、がサーバーの作成 AWS OpsWorks に使用する AWS CloudFormation スタックには伝達されません。

トピック

- [でのタグの仕組み AWS OpsWorks for Chef Automate](#)

- [でのタグの追加と管理 AWS OpsWorks for Chef Automate \(コンソール \)](#)
- [AWS OpsWorks for Chef Automate \(CLI\) でのタグの追加と管理](#)
- [以下の資料も参照してください。](#)

でのタグの仕組み AWS OpsWorks for Chef Automate

このリリースでは、[AWS OpsWorks CM API](#) または AWS Management Console を使用してタグを追加および管理できます。AWS OpsWorks また、CM は、EC2 インスタンス、Secrets Manager のシークレット、Elastic IP アドレス、セキュリティグループ、S3 バケット、バックアップなど、サーバーに関連付けられている AWS リソースにサーバーに追加するタグを追加しようとします。以下の表では、AWS OpsWorks for Chef Automate でタグを追加および管理する方法の概要を示しています。

アクション	使用するもの
<p>手動で作成する新しい AWS OpsWorks for Chef Automate サーバーまたはバックアップにタグを追加します。</p>	<ul style="list-style-type: none"> • [Create Chef Automate server (Chef Automate サーバーの作成)] を選択し、[Configure advanced settings (詳細設定の設定)] ページでタグを追加します。 • 既存のサーバーの [Create backup] (バックアップの作成) ページで [Backups] (バックアップ) を選択し、[Create a backup of your Chef Automate 2 server] (Chef Automate 2 サーバーのバックアップの作成) ページでタグを追加します。 • CreateServer または CreateBackup コマンドに Tags パラメータを追加します。
<p>リソースのタグを表示します。</p>	<ul style="list-style-type: none"> • サーバーの詳細ページで、ナビゲーションペインで [Tags (タグ)] を選択します。 • サーバーの [Backups (バックアップ)] ページで、バックアップを選択し、[Edit backup (バックアップの編集)] を選択します。 • ListTagsForResource コマンドを実行します。

アクション	使用するもの
<p>バックアップが手動または自動で作成されたかどうかにかかわらず、既存の AWS OpsWorks for Chef Automate サーバーまたはバックアップにタグを追加します。</p>	<ul style="list-style-type: none"> サーバーの詳細ページで、ナビゲーションペインで [Tags (タグ)] を選択し、[Edit (編集)] を選択します。 サーバーの [Backups (バックアップ)] ページで、バックアップを選択し、[Edit backup (バックアップの編集)] を選択します。 TagResource コマンドを実行します。
<p>リソースからタグを削除する</p>	<ul style="list-style-type: none"> サーバーの詳細ページで、ナビゲーションペインで [Tags (タグ)] を選択し、[Edit (編集)] を選択します。削除するタグの横にある [X] を選択します。 サーバーの [Backups (バックアップ)] ページで、バックアップを選択し、[Edit backup (バックアップの編集)] を選択します。削除するタグの横にある [X] を選択します。 UntagResource コマンドを実行します。

DescribeServers および DescribeBackups のレスポンスにタグ情報は含まれません。タグを表示するには、ListTagsForResource API を使用します。

でのタグの追加と管理 AWS OpsWorks for Chef Automate (コンソール)

このセクションの手順は AWS Management Console で実行されます。

タグを追加する場合、タグのキーを空にすることはできません。キーは最大 127 文字で、Unicode 文字、数字、区切り文字、または特殊文字 (+ - = . _ : / @) のみを含めることができます。タグの値はオプションです。キーはあるが値はないタグであれば、追加できます。値は最大 255 文字とし、Unicode 文字、数字、区切り文字、または特殊文字 (+ - = . _ : / @) のみを含めることができます。

トピック

- [新しい AWS OpsWorks for Chef Automate サーバーにタグを追加する \(コンソール\)](#)
- [新しいバックアップにタグを追加する \(コンソール\)](#)

- [既存のサーバーにタグを追加または表示する \(コンソール\)](#)
- [既存のバックアップにタグを追加または表示する \(コンソール\)](#)
- [サーバからタグを削除する \(コンソール\)](#)
- [バックアップからタグを削除する \(コンソール\)](#)

新しい AWS OpsWorks for Chef Automate サーバーにタグを追加する (コンソール)

1. AWS OpsWorks for Chef Automate サーバーを作成するための[前提条件](#)をすべて満たしてください。
2. [Chef Automate サーバーの作成](#) の手順 1～10 に従います。
3. 自動バックアップ設定を指定した後、[Configure advanced settings] (詳細設定の設定) ページの [Tags] (タグ) の領域でタグを追加します。最大 50 個のタグを追加できます。タグを追加したら、[次へ] を選択します。
4. [Chef Automate サーバーの作成](#) のステップ 13 に進み、新しいサーバーに対して選択した設定を確認します。

新しいバックアップにタグを追加する (コンソール)

1. AWS OpsWorks for Chef Automate ホームページで、既存の Chef Automate サーバーを選択します。
2. サーバーの詳細ページで、ナビゲーションペインで [Backups (バックアップ)] を選択します。
3. [Backups (バックアップ)] ページで、[Create backup (バックアップの作成)] を選択します。
4. タグを追加する。タグの追加が完了したら、[Create (作成)] を選択します。

既存のサーバーにタグを追加または表示する (コンソール)

1. AWS OpsWorks for Chef Automate ホームページで、既存の Chef Automate サーバーを選択して詳細ページを開きます。
2. ナビゲーションペインで [Tags (タグ)] を選択するか、詳細ページの下部にある [View all tags (すべてのタグを表示)] を選択します。
3. [Tags (タグ)] ページで、[Edit (編集)] を選択します。
4. サーバー上のタグを追加または編集します。完了したら、[保存] を選択します。

Note

Chef Automate サーバーでタグを変更すると、EC2 インスタンス、Elastic IP アドレス、セキュリティグループ、S3 バケット、バックアップなど、サーバーに関連付けられているリソースのタグも変更されることに注意してください。

既存のバックアップにタグを追加または表示する (コンソール)

1. AWS OpsWorks for Chef Automate ホームページで、既存の Chef Automate サーバーを選択して詳細ページを開きます。
2. ナビゲーションペインで [Backups (バックアップ)] を選択するか、詳細ページの [Recent backups (最近のバックアップ)] 領域で [View all backups (すべてのバックアップを表示)] を選択します。
3. [Backups (バックアップ)] ページで、管理するバックアップを選択し、[Edit backup (バックアップの編集)] を選択します。
4. バックアップにタグを追加または編集します。完了したら、[Update (更新)] を選択します。

サーバからタグを削除する (コンソール)

1. AWS OpsWorks for Chef Automate ホームページで、既存の Chef Automate サーバーを選択して詳細ページを開きます。
2. ナビゲーションペインで [Tags (タグ)] を選択するか、詳細ページの下部にある [View all tags (すべてのタグを表示)] を選択します。
3. [Tags (タグ)] ページで、[Edit (編集)] を選択します。
4. タグを削除するには、タグの横にある [X] を選択します。完了したら、[保存] を選択します。

Note

Chef Automate サーバーでタグを変更すると、EC2 インスタンス、Elastic IP アドレス、セキュリティグループ、S3 バケット、バックアップなど、サーバーに関連付けられているリソースのタグも変更されることに注意してください。

バックアップからタグを削除する (コンソール)

1. AWS OpsWorks for Chef Automate ホームページで、既存の Chef Automate サーバーを選択して詳細ページを開きます。
2. ナビゲーションペインで [Backups (バックアップ)] を選択するか、詳細ページの [Recent backups (最近のバックアップ)] 領域で [View all backups (すべてのバックアップを表示)] を選択します。
3. [Backups (バックアップ)] ページで、管理するバックアップを選択し、[Edit backup (バックアップの編集)] を選択します。
4. タグを削除するには、タグの横にある [X] を選択します。完了したら、[Update (更新)] を選択します。

AWS OpsWorks for Chef Automate (CLI) でのタグの追加と管理

このセクションの手順は AWS CLI で実行されます。タグの使用 AWS CLI を開始する前に、の最新リリースを実行していることを確認してください。のインストールまたは更新の詳細については AWS CLI、「AWS Command Line Interface ユーザーガイド」の「[AWS CLI のインストール](#)」を参照してください。

タグを追加する場合、タグのキーを空にすることはできません。キーは最大 127 文字で、Unicode 文字、数字、区切り文字、または特殊文字 (+ - = . _ : / @) のみを含めることができます。タグの値はオプションです。キーはあるが値はないタグであれば、追加できます。値は最大 255 文字とし、Unicode 文字、数字、区切り文字、または特殊文字 (+ - = . _ : / @) のみを含めることができます。

トピック

- [新しい AWS OpsWorks for Chef Automate サーバーにタグを追加する \(CLI\)](#)
- [新しいバックアップにタグを追加する \(CLI\)](#)
- [既存のサーバーまたはバックアップにタグを追加する \(CLI\)](#)
- [リソースタグのリストを取得する](#)
- [リソースからタグを削除する](#)

新しい AWS OpsWorks for Chef Automate サーバーにタグを追加する (CLI)

AWS OpsWorks for Chef Automate サーバーを作成するときに、を使用してタグ AWS CLI を追加できます。この手順では、サーバーの作成方法について詳しく説明していません。このガイドの「」を

参照して [AWS OpsWorks for Chef Automate サーバーを作成する方法の詳細については を使用して Chef Automate サーバーを作成する AWS CLI](#)、AWS CLI 「」を参照してください。サーバーには最大 50 個のタグを追加できます。

1. AWS OpsWorks for Chef Automate サーバーを作成するための [前提条件](#) をすべて満たしてください。
2. 「[を使用して Chef Automate サーバーを作成する AWS CLI](#)」のステップ 1～5 を完了します。
3. ステップ 6 では、create-server コマンドを実行するときに、以下の例に示すように、コマンドに --tags パラメータを追加します。

```
aws opsworks-cm create-server ... --tags Key=Key1,Value=Value1  
Key=Key2,Value=Value2
```

以下の例では、create-server コマンドのタグ部分のみを示しています。

```
aws opsworks-cm create-server ... --tags Key=Stage,Value=Production  
Key=Department,Value=Marketing
```

4. 「[を使用して Chef Automate サーバーを作成する AWS CLI](#)」の残りのステップを行います。タグが新しいサーバーに追加されたことを確認するには、このトピックの「[リソースタグのリストを取得する](#)」の手順に従います。

新しいバックアップにタグを追加する (CLI)

サーバーの新しい手動バックアップを作成するときに、AWS OpsWorks for Chef Automate を使用してタグ AWS CLI を追加できます。この手順では、手動バックアップの作成方法について詳しく説明していません。手動バックアップの作成方法の詳細については、「」の「手動バックアップを実行するには AWS CLI」を参照してください。[AWS OpsWorks for Chef Automate サーバーのバックアップ](#)。バックアップには最大 50 個のタグを追加できます。サーバにタグがある場合、新しいバックアップにはサーバのタグが自動的にタグ付けされます。

デフォルトでは、新しい AWS OpsWorks for Chef Automate サーバーを作成すると、自動バックアップが有効になります。このトピックの「[既存のサーバーまたはバックアップにタグを追加する \(CLI\)](#)」で説明されている tag-resource コマンドを実行して、自動バックアップにタグを追加できます。

- 手動バックアップの作成中にタグをバックアップに追加するには、以下のコマンドを実行します。コマンドのタグ部分のみを示しています。フル create-backup コマンドの例に関して

は、[AWS OpsWorks for Chef Automate サーバーのバックアップ](#) の「AWS CLIで手動バックアップを実行するには」を参照してください。

```
aws opsworks-cm create-backup ... --tags Key=Key1,Value=Value1
Key=Key2,Value=Value2
```

以下の例では、create-backup コマンドのタグ部分のみを示しています。

```
aws opsworks-cm create-backup ... --tags Key=Stage,Value=Production
Key=Department,Value=Marketing
```

既存のサーバーまたはバックアップにタグを追加する (CLI)

tag-resource コマンドを実行して、タグを既存の AWS OpsWorks for Chef Automate サーバーまたはバックアップに追加できます (バックアップが手動で作成されたか自動的に作成されたかに関係なく)。ターゲットリソースの Amazon リソースナンバー (ARN) を指定して、タグを追加します。

1. タグを適用するリソースの ARN を取得するには:

- サーバーの場合は、describe-servers --server-name *server_name* を実行します。コマンドの結果には、サーバーの ARN が表示されます。
- バックアップの場合は、describe-backups --backup-id *backup_ID* を実行します。コマンドの結果には、バックアップの ARN が表示されます。を実行してdescribe-backups --server-name *server_name*、特定の AWS OpsWorks for Chef Automate サーバーのすべてのバックアップに関する情報を表示することもできます。

以下の例では、ServerArn コマンドの結果の describe-servers --server-name opsworks-cm-test のみを示しています。tag-resource コマンドに ServerArn 値を指定して、サーバーにタグを追加します。

```
{
  "Servers": [
    {
      ...
      "ServerArn": "arn:aws:opsworks-cm:us-west-2:123456789012:server/
opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
    }
  ]
}
```

```
}
```

2. ステップ 1 で返された ARN を使用して、`tag-resource` コマンドを実行します。

```
aws opsworks-cm tag-resource --resource-arn "server_or_backup_ARN" --tags  
Key=Key1,Value=Value1 Key=Key2,Value=Value2
```

次に例を示します。

```
aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-  
west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"  
--tags Key=Stage,Value=Production Key=Department,Value=Marketing
```

3. タグが正常に追加されたことを確認するには、次の手順「[リソースタグのリストを取得する](#)」に進みます。

リソースタグのリストを取得する

`list-tags-for-resource` コマンドを実行して、AWS OpsWorks for Chef Automate サーバーまたはバックアップにアタッチされているタグを表示できます。ターゲットリソースの ARN を指定して、そのタグを表示します。

1. タグを一覧表示するリソースの ARN を取得するには:
 - サーバーの場合は、`describe-servers --server-name server_name` を実行します。コマンドの結果には、サーバーの ARN が表示されます。
 - バックアップの場合は、`describe-backups --backup-id backup_ID` を実行します。コマンドの結果には、バックアップの ARN が表示されます。を実行して `describe-backups --server-name server_name`、特定の AWS OpsWorks for Chef Automate サーバーのすべてのバックアップに関する情報を表示することもできます。
2. ステップ 1 で返された ARN を使用して、`list-tags-for-resource` コマンドを実行します。

```
aws opsworks-cm list-tags-for-resource --resource-arn "server_or_backup_ARN"
```

次に例を示します。

```
aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-  
west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
```

リソースにタグがある場合、コマンドは以下のような結果を返します。

```
{
  "Tags": [
    {
      "Key": "Stage",
      "Value": "Production"
    },
    {
      "Key": "Department",
      "Value": "Marketing"
    }
  ]
}
```

リソースからタグを削除する

`untag-resource` コマンドを実行して、AWS OpsWorks for Chef Automate サーバーまたはバックアップからタグを削除できます。リソースが削除されると、リソースのタグも削除されます。ターゲットリソースの Amazon リソースナンバー (ARN) を指定して、タグを削除します。

1. タグを削除するリソースの ARN を取得するには:
 - サーバーの場合は、`describe-servers --server-name server_name` を実行します。コマンドの結果には、サーバーの ARN が表示されます。
 - バックアップの場合は、`describe-backups --backup-id backup_ID` を実行します。コマンドの結果には、バックアップの ARN が表示されます。を実行して `describe-backups --server-name server_name`、特定の AWS OpsWorks for Chef Automate サーバーのすべてのバックアップに関する情報を表示することもできます。
2. ステップ 1 で返された ARN を使用して、`untag-resource` コマンドを実行します。削除するタグのみを指定します。

```
aws opsworks-cm untag-resource --resource-arn "server_or_backup_ARN" --tags
Key=Key1,Value=Value1 Key=Key2,Value=Value2
```

この例では、`untag-resource` コマンドは、キーが Stage で値が Production のタグのみを削除します。

```
aws opsworks-cm untag-resource --resource-arn "arn:aws:opsworks-cm:us-west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE" --tags Key=Stage,Value=Production
```

3. タグが正常に削除されたことを確認するには、このトピックの「[リソースタグのリストを取得する](#)」の手順に従います。

以下の資料も参照してください。

- [を使用して Chef Automate サーバーを作成する AWS CLI](#)
- [AWS OpsWorks for Chef Automate サーバーのバックアップ](#)
- [AWS タグ付け戦略](#)
- AWS Identity and Access Management ユーザーガイドの[AWS リソースタグを使用したリソースへのアクセスの制御](#)
- 「AWS Billing and Cost Management ユーザーガイド」の「[コスト配分タグの使用](#)」
- 「[CreateBackup](#)」 (AWS OpsWorks CM API リファレンス)
- 「[CreateServer](#)」 (AWS OpsWorks CM API リファレンス)
- 「[TagResource](#)」 (AWS OpsWorks CM API リファレンス)
- 「[ListTagsForResource](#)」 (AWS OpsWorks CM API リファレンス)
- 「[UntagResource](#)」 (AWS OpsWorks CM API リファレンス)

AWS OpsWorks for Chef Automate サーバーのバックアップと復元

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、AWS OpsWorks for Chef Automate サーバーをバックアップおよび復元する方法と、バックアップを削除する方法について説明します。

トピック

- [AWS OpsWorks for Chef Automate サーバーのバックアップ](#)
- [バックアップから AWS OpsWorks for Chef Automate サーバーを復元する](#)

AWS OpsWorks for Chef Automate サーバーのバックアップ

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

毎日または毎週の定期的な AWS OpsWorks for Chef Automate サーバーバックアップを定義し、ユーザーに代わって Amazon Simple Storage Service (Amazon S3) にバックアップを保存させることができます。または、必要に応じて手動でバックアップを実行することもできます。

バックアップが Amazon S3 に保存されるため、追加料金を負担します。30 世代を上限としてバックアップ保持期間を定義できます。AWS サポートチャンネルを使用してサービスリクエストを送信し、その制限を変更することができます。Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、「[S3 バケットを空にする方法](#)」または「[S3 バケットを削除する方法](#)」を参照してください。

AWS OpsWorks for Chef Automate サーバーのバックアップにタグを追加できます。AWS OpsWorks for Chef Automate サーバーにタグを追加した場合、サーバーの自動バックアップはこれらのタグを継承します。バックアップでタグを追加および管理する方法についての詳細は、このガイドの「[AWS OpsWorks for Chef Automate リソースでのタグの使用](#)」を参照してください。

トピック

- [自動バックアップ](#)
- [手動バックアップ](#)
- [バックアップの削除](#)

自動バックアップ

AWS OpsWorks for Chef Automate サーバーを設定するときは、自動バックアップまたは手動バックアップのいずれかを選択します。は、セットアップの高度な設定を構成するページの「自動バックアップ」セクションで選択した時間および日に自動バックアップ AWS OpsWorks for Chef Automate を開始します。サーバーがオンライン状態になった後で、バックアップ設定を変更することもできます。そのためには、Chef Automate サーバーのホームページのサーバータイルまたはサーバーの [Properties] ページで、以下のステップを実行します。

自動バックアップ設定を変更するには

1. [Chef servers] (Chef サーバー) ホームページのサーバのタイルの [Actions] (アクション) メニューで、[Change settings] (設定を変更する) を選択します。
2. 自動バックアップを無効にするには、[Enable automated backups] (自動バックアップの有効化) オプションの [No] (いいえ) を選択します。変更を保存します。次のステップに進む必要はありません。
3. [Automated Backup] セクションで、頻度、開始時刻、または保持する世代数を変更します。変更を保存します。

手動バックアップ

手動バックアップは、いつでも開始することも AWS Management Console、[create-backup](#) コマンドを実行して開始 AWS CLI することもできます。手動バックアップは、保存される最大 30 世代の自動バックアップには含まれず、最大 10 個の手動バックアップが保存され、Amazon S3 から手動で削除する必要があります。

AWS OpsWorks for Chef Automate サーバーの新しい手動バックアップを作成するときに、タグを追加できます。手動バックアップを作成するときにタグを追加する方法の詳細については、「[新しいバックアップにタグを追加する \(CLI\)](#)」を参照してください

で手動バックアップを実行するには AWS Management Console

1. [Chef Automate servers] ページで、バックアップするサーバーを選択します。
2. サーバーのプロパティページの左のナビゲーションペインで、[Backups] を選択します。
3. [Create backup] (バックアップの作成) を選択します。
4. ページでバックアップの [Status] 欄に緑色のチェックマークが表示されると、手動バックアップは完了です。

で手動バックアップを実行するには AWS CLI

- 手動バックアップを開始するには、次の AWS CLI コマンドを実行します。

```
aws opsworks-cm --region region name create-backup --server-name "Chef server name"  
--description "optional descriptive string"
```

バックアップの削除

バックアップを削除すると、バックアップが保存されている S3 バケットから完全に削除されます。

でバックアップを削除するには AWS Management Console

1. [Chef Automate servers] ページで、バックアップするサーバーを選択します。
2. サーバーのプロパティページの左のナビゲーションペインで、[Backups] を選択します。
3. 削除するバックアップを選択してから、[Delete backup] を選択します。一度に 1 つのバックアップのみを選択できます。
4. 削除の確認を指示されたら、[Delete the backup, which is stored in an S3 bucket] チェックボックスにチェックを入れ、[Yes, Delete] を選択します。

でバックアップを削除するには AWS CLI

- バックアップを削除するには、次の AWS CLI コマンドを実行し、 を削除するバックアップの ID `--backup-id` に置き換えます。バックアップ IDs 形式は `ServerName-yyyyMMddHHmmssSSS` です。例えば `test-chef-server-20171218132604388` です。

```
aws opsworks-cm --region region name delete-backup --backup-id ServerName-  
yyyyMMddHHmmssSSS
```

バックアップから AWS OpsWorks for Chef Automate サーバーを復元する

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューション

シオンに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

使用可能なバックアップを参照した後、AWS OpsWorks for Chef Automate サーバーを復元するポイントインタイムを選択できます。サーバーのバックアップには、設定管理ソフトウェアの永続データ (クックブック、登録されたノードなど) のみが含まれています。サーバーのインプレース復元を実行する (つまり、既存の AWS OpsWorks for Chef Automate サーバーを新しい EC2 インスタンスに復元する) と、サーバーの復元に使用するバックアップ時に登録されたノードが再登録され、復元が成功し、復元された AWS OpsWorks for Chef Automate サーバーの状態が `Healthy` になると、トラフィックが新しいインスタンスに切り替わります。新しく作成された AWS OpsWorks for Chef Automate サーバーに復元すると、ノード接続は維持されません。サーバーを復元すると、Chef ソフトウェアのマイナーバージョンが更新されません。選択したバックアップにあるものと同じ Chef バージョンと設定管理データが適用されます。

通常、サーバの復元には新しいサーバを作成するよりも時間がかかります。時間は選択するバックアップのサイズによります。復元が完了すると、古い EC2 インスタンスは `Running` または `Stopped` 状態のままになりますが、一時的にのみです。最終的には終了します。

このリリースでは、を使用して AWS CLI で Chef サーバーを復元できます AWS OpsWorks for Chef Automate。

Note

[restore-server](#) コマンドは、現在のインスタンスタイプを変更する場合や、紛失または漏洩が発生した SSH キーの復元または設定を行う場合にも実行できます。

サーバーをバックアップから復元するには

1. で AWS CLI、次のコマンドを実行して、使用可能なバックアップとその IDs のリストを返します。使用するバックアップの ID を書き留めておきます。バックアップ IDs 形式は `myServerName-yyyyMMddHHmmssSSS` です。

```
aws opsworks-cm --region region name describe-backups
```

2. 以下のコマンドを実行します。

```
aws opsworks-cm --region region name restore-server --backup-id "myServerName-  
yyyyMMddHHmmssSSS" --instance-type "Type of instance" --key-pair "name of your EC2  
key pair" --server-name "name of Chef server"
```

次に例を示します。

```
aws opsworks-cm --region us-west-2 restore-server --backup-id  
"MyChefServer-20161120122143125" --server-name "MyChefServer"
```

3. 復元が完了するまで待ちます。

でのシステムメンテナンス AWS OpsWorks for Chef Automate

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

必須のシステムメンテナンスにより、セキュリティ更新プログラムを含む最新のマイナーバージョンの Chef Server と Chef Automate Server が常に AWS OpsWorks for Chef Automate サーバーで実行されます。システムメンテナンスは週に 1 回以上実行する必要があります。を使用すると AWS CLI、必要に応じて毎日の自動メンテナンスを設定できます。を使用して AWS CLI、スケジュールされたシステムメンテナンスに加えて、オンデマンドでシステムメンテナンスを実行することもできます。

Chef ソフトウェアの新しいマイナーバージョンが使用可能になった場合、システムメンテナンスでは、それらが AWS のテストをパスし次第、サーバー上の Chef Automate および Chef Server のマイナーバージョンを自動的に更新するように設計されています。AWS は広範なテストを実施して、Chef のアップグレードが本番環境に対応しており、既存の顧客環境を中断しないことを確認します。そのため、Chef ソフトウェアリリースと Chef OpsWorks Automate サーバー用の既存のへのアプリケーションの可用性との間に遅延が生じる可能性があります。Chef ソフトウェアの使用可能なマイナーバージョンをオンデマンドで更新するには、このトピックの「[オンデマンドでのシステムメンテナンスの開始](#)」を参照してください。

システムメンテナンスでは、メンテナンスプロセスの一環として実行されるバックアップから新しいインスタンスを起動します。これにより、定期的なメンテナンスを受ける Amazon EC2 インスタンスの低下や障害によるリスクを軽減することができます。

Important

システムメンテナンスでは、AWS OpsWorks for Chef Automate サーバーに追加したすべてのファイルやカスタム設定が削除されます。失われた設定やファイルを回復する詳しい方法については、このトピックの「[メンテナンス後のカスタム設定およびカスタムファイルの復旧](#)」を参照してください。

トピック

- [ノードが AWS OpsWorks 認証機関を信頼していることを確認する](#)
- [システムメンテナンスの設定](#)
- [オンデマンドでのシステムメンテナンスの開始](#)
- [メンテナンス後のカスタム設定およびカスタムファイルの復旧](#)

ノードが AWS OpsWorks 認証機関を信頼していることを確認する

Note

AWS OpsWorks for Chef Automate サーバーでカスタムドメインと証明書を使用している場合、このセクションのステップは必要ありません。

AWS OpsWorks for Chef Automate サーバーで管理しているノードは、証明書を使用してサーバーで認証する必要があります。システムメンテナンス中、はサーバーインスタンスを AWS OpsWorks 置き換え、認証局 (CA) を通じて新しい AWS OpsWorks 証明書を再生成します。メンテナンスの完了後にマネージドノードとの通信を自動的に復元するには、ノードはスターターキットに付属し、サポートされているリージョンでホストされている AWS OpsWorks CA を信頼する必要があります AWS OpsWorks for Chef Automate。AWS OpsWorks CA を使用してノードとサーバー間の信頼を確立すると、ノードはメンテナンス後に新しいサーバーインスタンスに再接続します。で説明されている EC2 userdata スクリプトを使用して EC2 ノードを追加する場合で [ノードを自動的に追加する AWS OpsWorks for Chef Automate](#)、ノードは AWS OpsWorks CA を信頼するように既に設定されています。

- Linux ベースのノードの場合、CA の S3 バケットの場所は `https://opsworks-cm-{REGION}-prod-default-assets.s3.amazonaws.com/misc/opsworks-cm-ca-2020-root.pem` です。AWS OpsWorks 信頼された CA はパスに保存する必要があります/`etc/chef/opsworks-cm-ca-2020-root.pem`。
- Windows ベースのノードの場合、CA の S3 バケットの場所は `https://opsworks-cm-$env:AWS_REGION-prod-default-assets.s3.amazonaws.com/misc/opsworks-cm-ca-2020-root.pem` です。AWS OpsWorks CA はルート Chef フォルダに保存する必要があります。例えば、`C:\chef\opsworks-cm-ca-2020-root.pem`

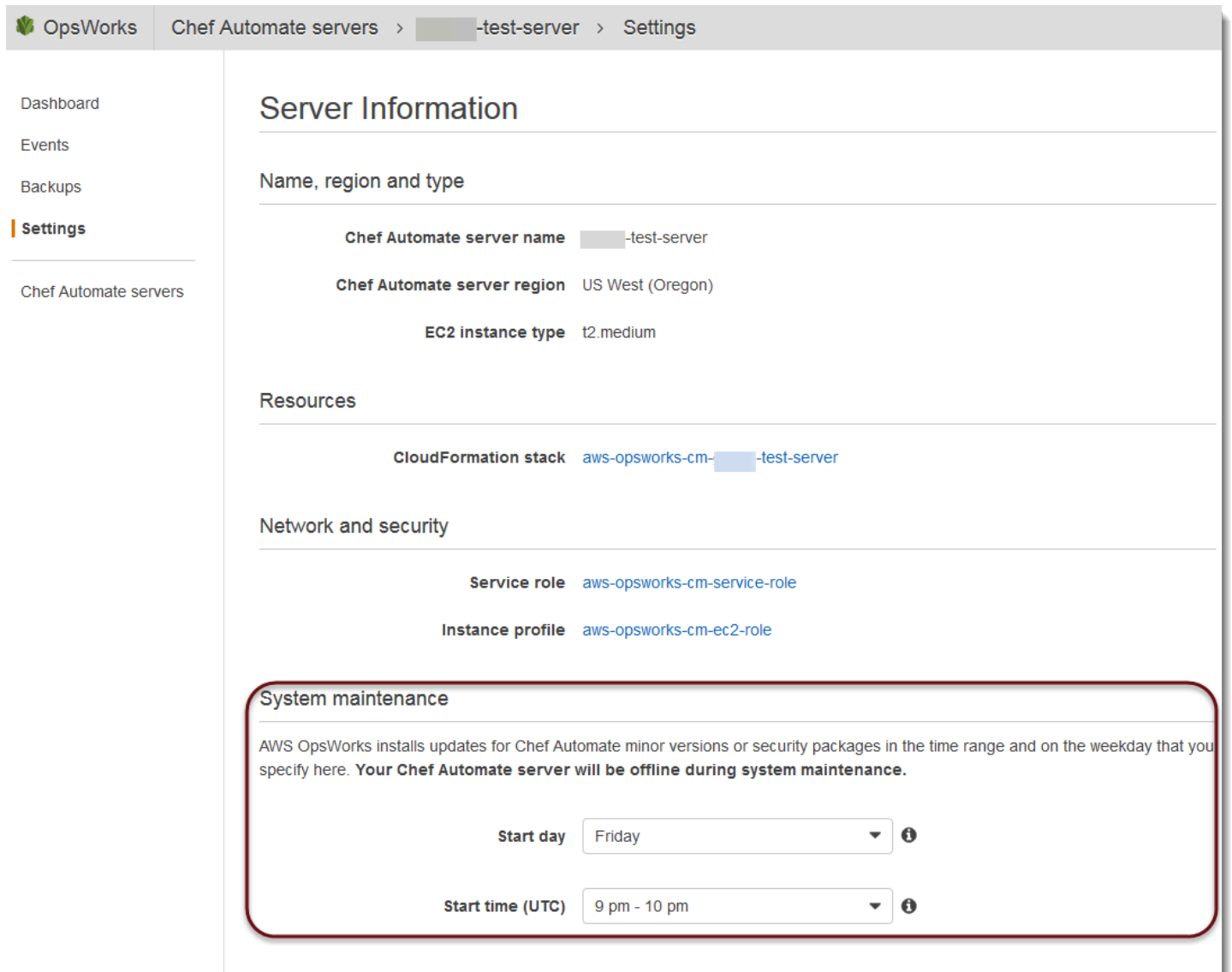
この 2 つのパスで region 変数は次のいずれかに解決されます。

- us-east-2
- us-east-1
- us-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- eu-central-1
- eu-west-1

システムメンテナンスの設定

新しい AWS OpsWorks for Chef Automate サーバーを作成するときは、システムメンテナンスを開始するために、[協定世界時](#) (UTC) で曜日と時刻を設定できます。メンテナンスは、指定した時間中に開始します。システムメンテナンス中はサーバーがオフラインになることを想定する必要があるため、通常の営業時間内でサーバー需要が低い時刻を選択します。メンテナンスの進行中は、サーバーのステータスが `UNDER_MAINTENANCE` になります。

次のスクリーンショットに示すように、AWS OpsWorks for Chef Automate サーバーの設定ページのシステムメンテナンスエリアで設定を変更することで、既存のサーバーのシステムメンテナンス設定を変更することもできます。



The screenshot displays the AWS OpsWorks console interface for configuring a Chef Automate server. The breadcrumb navigation at the top reads: OpsWorks > Chef Automate servers > [redacted]-test-server > Settings. The left-hand navigation pane includes links for Dashboard, Events, Backups, Settings (highlighted), and Chef Automate servers. The main content area is titled 'Server Information' and is divided into several sections:

- Name, region and type:** Displays 'Chef Automate server name' as [redacted]-test-server, 'Chef Automate server region' as US West (Oregon), and 'EC2 instance type' as t2.medium.
- Resources:** Shows 'CloudFormation stack' as aws-opsworks-cm-[redacted]-test-server.
- Network and security:** Shows 'Service role' as aws-opsworks-cm-service-role and 'Instance profile' as aws-opsworks-cm-ec2-role.
- System maintenance:** This section is highlighted with a red border. It contains the following text: 'AWS OpsWorks installs updates for Chef Automate minor versions or security packages in the time range and on the weekday that you specify here. **Your Chef Automate server will be offline during system maintenance.**' Below this text are two dropdown menus: 'Start day' is set to 'Friday' and 'Start time (UTC)' is set to '9 pm - 10 pm'. Each dropdown menu has an information icon (i) to its right.

[System maintenance] セクションで、システムメンテナンスを開始する日付と時刻を設定します。

を使用したシステムメンテナンスの設定 AWS CLI

システムメンテナンスの自動開始時刻を AWS CLI で設定することもできます。AWS CLI では、曜日の 3 文字のプレフィックスを省略することで、必要に応じて毎日の自動メンテナンスを設定できます。

create-server コマンドで、サーバーインスタンスを作成する要件 (インスタンスタイプ、インスタンスプロファイル ARN、サービスロール ARN など) を指定した後で、--preferred-maintenance-window パラメータをコマンドに追加します。次の create-server の例では、--preferred-maintenance-window を Mon:08:00 に設定しています。これで、毎月曜の午前 8 時 0 分 (UTC) にメンテナンスが開始されます。

```
aws opsworks-cm create-server --engine "Chef" --engine-model "Single" --  
engine-version "12" --server-name "automate-06" --instance-profile-arn  
"arn:aws:iam::1019881987024:instance-profile/aws-opsworks-cm-ec2-role"  
--instance-type "t2.medium" --key-pair "amazon-test" --service-role-arn  
"arn:aws:iam::044726508045:role/aws-opsworks-cm-service-role" --preferred-maintenance-  
window "Mon:08:00"
```

update-server コマンドでは、必要に応じて、--preferred-maintenance-window の値のみを更新できます。次の例では、メンテナンス時刻を金曜の午後 6 時 15 分 (UTC) に設定しています。

```
aws opsworks-cm update-server --server-name "shiny-kitchen" --preferred-maintenance-  
window "Fri:18:15"
```

メンテナンス時間の開始時刻を毎日午後 6 時 15 分 (UTC) に変更するには、次の例に示すように、曜日を表す 3 文字のプレフィックスを省略します。

```
aws opsworks-cm update-server --server-name "shiny-kitchen" --preferred-maintenance-  
window "18:15"
```

を使用して優先システムメンテナンスウィンドウを設定する方法の詳細については AWS CLI、[「create-server」](#) および [「update-server」](#) を参照してください。

オンデマンドでのシステムメンテナンスの開始

システムメンテナンスをオンデマンドで開始するには、毎週または毎日の自動メンテナンスが設定されたの外部で、次の AWS CLI コマンドを実行します。AWS Management Console でオンデマンドメンテナンスを開始することはできません。

```
aws opsworks-cm start-maintenance --server-name server_name
```

このコマンドの詳細については、[「start-maintenance」](#) を参照してください。

メンテナンス後のカスタム設定およびカスタムファイルの復旧

システムメンテナンスでは、AWS OpsWorks for Chef Automate サーバーに追加したカスタムファイルまたは設定を削除または変更できます。

RunCommand または SSH を使用して追加したファイルや設定がメンテナンスの実行後に Chef サーバーに見つからない場合は、Amazon マシンイメージ (AMI) を使用して、新しい Amazon EC2 イン

スタンスを起動できます。サーバーのメンテナンス前の設定に基づいて構築された AMI を利用できます。

新しいインスタンスは、メンテナンス前の Chef サーバーと同じ状態であり、見つからないファイルや設定が含まれています。

Important

新しいインスタンスを使用してサーバーを復元することはできません。インスタンスを Chef サーバーとして実行することはできません。インスタンスは、ファイルや設定の復旧にのみ使用できます。

AMI から EC2 インスタンスを起動するには、Amazon EC2 コンソールで [Launch] (起動) ウィザードを開き、[My AMIs] (マイAMI) を選択して、サーバー名と同じ名前の AMI を選択します。他の任意のインスタンスを起動する手順と同様に、Amazon EC2 ウィザードの手順に従います。

でのコンプライアンススキャン AWS OpsWorks for Chef Automate

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

コンプライアンススキャンを使用すると、事前定義されたポリシー (ルールとも呼ばれる) に基づいてインフラストラクチャ内の管理対象ノードのコンプライアンスを追跡できます。Compliance のビューを使用すると、アプリケーションの脆弱性と非準拠設定を定期的に監査できます。Chef には、コンプライアンススキャンに使用できる事前定義された 100 個を超える Compliance プロファイル (特定のノード設定に適用するルールの集合) が用意されています。[Chef InSpec 言語](#)を使用して、独自のカスタムプロファイルを作成することもできます。

サーバーでまだ Chef Automate 2.0 が実行されていない場合は、Audit クックブックをインストールすることで手動で [Chef Compliance](#) を設定できます。

Note

AWS OpsWorks for Chef Automate サーバーに関連付けられたノードでサポートされている Chef Infra クライアントエージェントソフトウェア (chef-client) の最小バージョンは 13.x です。最新の安定 chef-client バージョン、または少なくとも 14.10.9 を実行することをお勧めします。

トピック

- [Chef Automate 2.0 の Compliance](#)
- [Chef Automate 1.x の Compliance](#)
- [Compliance の更新](#)
- [コミュニティとカスタム Compliance プロファイル](#)
- [以下の資料も参照してください。](#)

Chef Automate 2.0 の Compliance

AWS OpsWorks for Chef Automate サーバーが Chef Automate 2.0 を実行している場合は、このセクションの手順を使用して Chef Compliance を設定します。

Chef Automate 2.0 を使用したコンプライアンススキャンジョブの実行

Chef Automate 2.0 には、以前は手動セットアップとクックブック設定が必要だった Chef InSpec コンプライアンススキャン機能が含まれています。Chef Automate 2.0 を実行している AWS OpsWorks for Chef Automate サーバーでスキャンジョブを実行できます。ジョブは、すぐに (1 回) 実行するか、後で実行するようにスケジュールするか、指定した間隔 (毎日または 2 時間ごとなど) で実行するようにスケジュールすることができます。スキャンジョブの結果はコンプライアンスレポートに送信されます。Chef Automate ダッシュボードで、コンプライアンススキャンの結果を表示して対処できます。[Compliance] タブを開いてレポートを表示するには、Chef Automate ダッシュボードの [Scan Jobs] タブで、管理対象ノード行の右側にある [Report] を選択します。

管理対象ノードでスキャンジョブを実行するには、以下のものがが必要です。

- 名前空間にインストールされた 1 つ以上の Compliance プロファイル。
- 手動で追加された 1 つの以上のターゲットノード、または [自動的に追加された](#) EC2 インスタンス。

では AWS OpsWorks for Chef Automate、次のターゲットでスキャンジョブがサポートされています。

- 手動で追加されたノード
- aws-ec2 インスタンス
- AWS リージョン

スキャンジョブの実行方法の詳細については、Chef のドキュメントの「[Chef Automate Scan Jobs](#)」を参照してください。

(オプション、Chef Automate 2.0) Audit クックブックによるコンプライアンスの設定

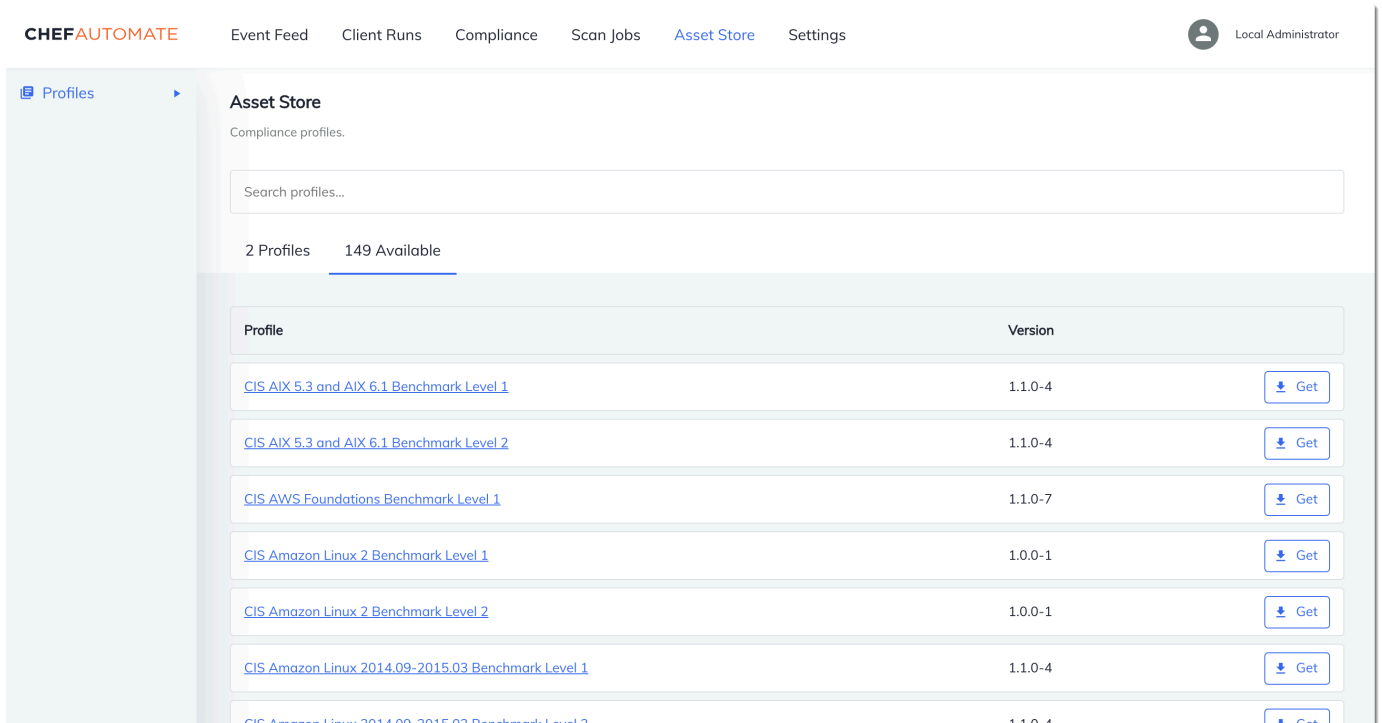
コンプライアンスはどの AWS OpsWorks for Chef Automate サーバーでも設定できます。AWS OpsWorks for Chef Automate サーバーを起動した後、Chef Automate ダッシュボードからプロファイルをインストールしたり、Policyfile.rb ポリシーファイルで Audit クックブックの属性に目的のプロファイルを追加したりできます。スターターキットには、事前入力済みの Policyfile.rb ファイルが含まれています。

監査クックブックの属性として Policyfile.rb プロファイルを編集した後、`chef push` コマンドを実行して [\[Audit cookbook\]](#) (監査クックブック) と Policyfile.rb で指定された他のクックブックを Chef Automate サーバーにアップロードします。Audit クックブックをインストールすると、[Chef InSpec](#) が生成するオープンソースのテストおよび監査フレームワークである Chef の gem もインストールされます。Chef Automate [2.0](#) の場合は、Audit クックブックのバージョン 7.1.0 以降を選択します。InSpec Gem はバージョン 2.2.102 以降である必要があります。

このセクションの手順は `opsworks-audit` クックブックを実装する方法を示しています。Audit クックブックは、Chef Automate サーバーから指定されたプロファイルをダウンロードし、DevSec SSH ベースラインプロファイルに対してノードを評価し、`chef-client` 実行ごとにコンプライアンススキャンの結果を報告します。

Compliance プロファイルをインストールするには

1. まだ行っていない場合は [Chef Automate ウェブベースダッシュボードにサインイン](#) します。AWS OpsWorks for Chef Automate サーバーの作成時にスターターキットをダウンロードした際に受信した認証情報を使用します。
2. Chef Automate ダッシュボードで、[Asset Store] タブを選択します。



The screenshot shows the 'Asset Store' section of the Chef Automate interface. It features a search bar, a navigation bar with '2 Profiles' and '149 Available' tabs, and a table of compliance profiles. The table has columns for 'Profile' and 'Version', and a 'Get' button for each row.

Profile	Version	Get
CIS AIX 5.3 and AIX 6.1 Benchmark Level 1	1.1.0-4	Get
CIS AIX 5.3 and AIX 6.1 Benchmark Level 2	1.1.0-4	Get
CIS AWS Foundations Benchmark Level 1	1.1.0-7	Get
CIS Amazon Linux 2 Benchmark Level 1	1.0.0-1	Get
CIS Amazon Linux 2 Benchmark Level 2	1.0.0-1	Get
CIS Amazon Linux 2014.09-2015.03 Benchmark Level 1	1.1.0-4	Get
CIS Amazon Linux 2014.09-2015.03 Benchmark Level 2	1.1.0-4	Get

3. 定義済みのプロファイルを表示するには、[Available] タブを選択します。
4. プロファイルのリストを参照します。オペレーティングシステムおよび少なくとも1つの管理対象ノードの設定に一致するプロファイルを選択します。プロファイルの対象となる違反の説明および基盤となるルールコードを含むプロファイルの詳細を表示するには、プロファイル項目の右にある [>] を選択します。複数のプロファイルを選択できます。Starter Kit で例を設定する場合は、DevSec SSH ベースライン を選択します。

The screenshot shows the 'Asset Store' section of the Chef Automate interface. At the top, there are navigation tabs: Event Feed, Client Runs, Compliance, Scan Jobs, Asset Store (selected), and Settings. A user profile icon for 'Local Administrator' is in the top right. On the left, a sidebar shows 'Profiles' with a dropdown arrow. The main content area is titled 'Asset Store' and 'Compliance profiles.' It features a search input field labeled 'Search profiles...'. Below the search field, it indicates '1 Profiles' and '149 Available'. A blue 'Upload Profile' button is present. A table lists the available profiles:

Profile	Version	Identifier
DevSec SSH Baseline	2.3.2	admin/ssh-baseline

5. 選択されたプロファイルを Chef Automate サーバーにインストールするには、[Get] を選択します。
6. プロファイルをインストールすると、それらは Chef Automate ダッシュボードの [Profiles] タブに表示されます。

Policyfile.rb を使用してクックブックをインストールするには

1. スターターキットの Policyfile.rb を表示して、Audit クックブックの属性で ['profiles'] の ssh-baseline プロファイルが指定されていることを確認します。

```
# Define audit cookbook attributes
default["opsworks-demo"]["audit"]["reporter"] = "chef-server-automate"
default["opsworks-demo"]["audit"]["profiles"] = [
  {
    "name": "DevSec SSH Baseline",
    "compliance": "admin/ssh-baseline"
  }
]
```

2. Policyfile.rb に定義されているクックブックをダウンロードしてインストールします。

```
chef install
```

すべてのクックブックはクックブックの `metadata.rb` ファイルでバージョンニングされています。クックブックを変更するたびに `metadata.rb` にあるクックブックのバージョンを上げる必要があります。

3. `Policyfile.rb` に定義されているポリシー `opsworks-demo` をサーバーにプッシュします。

```
chef push opsworks-demo
```

4. ポリシーのインストールを確認します。以下のコマンドを実行します。

```
chef show-policy
```

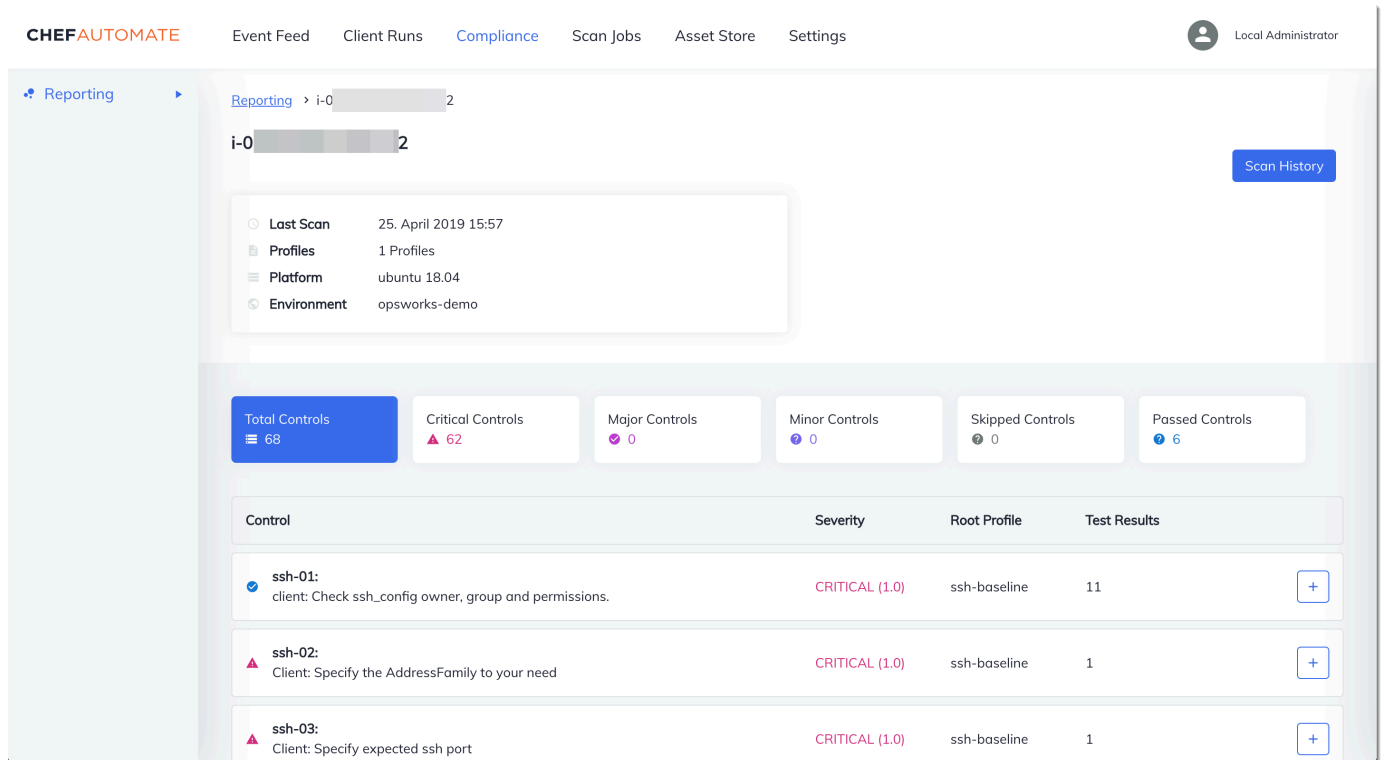
結果は以下のようになります。

```
opsworks-demo-webserver
=====
* opsworks-demo: ec0fe46314
```

5. まだ行っていない場合は、サーバーにノードを追加して管理できるようにします。最初のノードを AWS OpsWorks for Chef Automate サーバーに接続するには、このスターターキットに含まれている `userdata.sh` スクリプトを使用します。AssociateNode API を使用してノードを AWS OpsWorks サーバーに接続します。

[「でノードを自動的に追加する AWS OpsWorks for Chef Automate」](#) のステップに従ってノードの関連付けを自動化できます。または [「ノードを個別に追加します」](#) のステップに従ってノードを 1 度に 1 つずつ追加できます。

6. ノードの実行リストを更新すると、`chef-client` エージェントが次回の実行時に、指定したレシピを実行します。この処理はデフォルトで 1,800 秒 (30 分) ごとに行われます。その実行後、Chef Automate ダッシュボードの [Compliance] タブからコンプライアンス結果を表示し、アクションを実行できます。



The screenshot displays the Chef Automate Compliance interface. At the top, the navigation bar includes 'Event Feed', 'Client Runs', 'Compliance', 'Scan Jobs', 'Asset Store', and 'Settings'. The user is logged in as 'Local Administrator'. The main content area shows the 'Reporting' section for a specific node (i-0...). A summary box indicates the last scan was on 25 April 2019 at 15:57, with 1 profile, ubuntu 18.04 platform, and opsworks-demo environment. Below this, a summary row shows: Total Controls (68), Critical Controls (62), Major Controls (0), Minor Controls (0), Skipped Controls (0), and Passed Controls (6). A table lists the control details:

Control	Severity	Root Profile	Test Results
ssh-01: client: Check ssh_config owner, group and permissions.	CRITICAL (1.0)	ssh-baseline	11
ssh-02: Client: Specify the AddressFamily to your need	CRITICAL (1.0)	ssh-baseline	1
ssh-03: Client: Specify expected ssh port	CRITICAL (1.0)	ssh-baseline	1

コンプライアンススキャンの実行

ノードの実行リストを設定した後、エージェントが初めて実行されるとすぐに、Chef Automate ダッシュボードにコンプライアンススキャンの結果が表示されます。

CHEFAUTOMATE Event Feed Client Runs **Compliance** Scan Jobs Asset Store Settings Local Administrator

Reporting

Compliance Reporting

Compliance reports describe the status of scanned infrastructure. Filtering by a profile, or a profile and one associated control, will enable deep filtering, which will also reflect on the status of the node.

Filter reports by... 4/25/19

▲ Your System is Not Compliant Report Metadata +

Overview 1 Nodes 1 Profiles

Node Status Profile Status

The dashboard shows a summary of compliance status. A donut chart indicates 1 Total Nodes. A legend shows 1 Failed Node, 0 Passed Nodes, and 0 Skipped Nodes. A horizontal bar chart shows 1 Critical Failure, 0 Major Failures, and 0 Minor Failures.

Category	Count
Failed Nodes	1
Passed Nodes	0
Skipped Nodes	0
Critical Failures	1
Major Failures	0
Minor Failures	0

Chef Automate ダッシュボードで、[Compliance] タブを選択します。左のナビゲーションペインの [Reporting] を選択します。[Profiles] タブを選択して [Scan Results] を選択した後、スキャンによって非準拠と見なされたノードを選択して、そのノードがどのルールに対して非準拠になったかを調べます。

CHEFAUTOMATE Event Feed Client Runs **Compliance** Scan Jobs Asset Store Settings Local Administrator

Reporting

Compliance Reporting

Compliance reports describe the status of scanned infrastructure. Filtering by a profile, or a profile and one associated control, will enable deep filtering, which will also reflect on the status of the node.

Filter reports by... 4/25/19

▲ Your System is Not Compliant Report Metadata +

Overview 1 Nodes 1 Profiles

Nodes	Platform	Environment	Last Scan	Control Failures
▲ i-0...f2	ubuntu 18.04	opsworks-demo	vor 26 Minuten	62 FAILED

Scan Results

通常、新しいノードは DevSec SSH ベースラインプロファイルのすべてのルールをまだ満たしていないため、非準拠のスキャン結果が表示されます。コミュニティベースのプロジェクトである

[DevSec Hardening Framework](#) は、DevSec SSH ベースラインプロファイルのルールに違反する問題を修正するためのクックブックを提供しています。

(オプション) 非準拠結果の解決

スターターキットにはオープンソースのクックブックが含まれており `ssh-hardening`、DevSec SSH ベースラインプロファイルに対する実行の非準拠の結果を修正するために実行できます。

Note

`ssh-hardening` クックブックは、DevSec SSH ベースラインルールに準拠するようにノードを変更します。このクックブックを本番稼働用ノードで実行する前に、Chef Automate コンソールの DevSec SSH ベースラインプロファイルの詳細を確認して、クックブックがターゲットとするルール違反を理解してください。本稼働ノードでこれを実行する前に、オープンソースの [ssh-hardening](#) クックブックに関する情報を確認します。

`ssh-hardening` クックブックを実行するには

1. テキストエディタで、`ssh-hardening` の実行リストに `Policyfile.rb` クックブックを追加します。実行リスト `Policyfile.rb` は以下ようになります。

```
run_list 'chef-client', 'opsworks-webserver', 'audit', 'ssh-hardening'
```

2. `Policyfile.rb` を更新し、AWS OpsWorks for Chef Automate サーバーにプッシュします。

```
chef update Policyfile.rb
chef push opsworks-demo
```

3. `opsworks-demo` ポリシーに関連付けられているノードは実行リストを自動的に更新し、次の `chef-client` の実行時に `ssh-hardening` クックブックを適用します。

`chef-client` クックブックを使用しているため、ノードが定期的にチェックインします (デフォルトでは 30 分ごと)。次のチェックインでは、`ssh-hardening` クックブックが実行され、DevSec SSH ベースラインプロファイルのルールを満たすためにノードセキュリティが向上します。

4. `ssh-hardening` クックブックの初回実行が完了したら 30 分待機し再びコンプライアンススキャンを実行します。Chef Automate ダッシュボードで結果を表示します。DevSec SSH ベースラインスキャンの初回実行時に発生した非準拠の結果は解決する必要があります。

Chef Automate 1.x の Compliance

AWS OpsWorks for Chef Automate サーバーが Chef Automate 1.x を実行している場合は、このセクションの手順を使用して Chef Compliance を設定します。

(オプション、Chef Automate 1.x) Chef Compliance の設定

Chef Compliance はどの AWS OpsWorks for Chef Automate サーバーでも設定できます。AWS OpsWorks for Chef Automate サーバーを起動した後、Chef Automate ダッシュボードのプロファイルから実行するプロファイルを選択します。プロファイルをインストールした後、berks コマンドを実行して [Audit クックブック](#) を Chef Automate サーバーにアップロードします。Audit クックブックをインストールすると、用の gem もインストールされます。これは [InSpec](#)、Chef によって生成されたオープンソースのテストフレームワークで、自動テストをデプロイパイプラインの任意のステージに統合できます。Chef Automate 1.x の場合は、監査クックブックのバージョン 5.0.1 以降を選択します。InSpec Gem はバージョン 1.24.0 以降である必要があります。

AWS OpsWorks for Chef Automate スターターキットには、Chef の Audit クックブックの適切なバージョンをダウンロードしてインストールする opsworks-audit ラッパークックブックが含まれています。この opsworks-audit クックブックでは、このトピックの後半で Chef の Compliance コンソールからインストールする DevSec SSH ベースラインプロファイルに対してノードを評価するように chef-client エージェントに指示します。独自の設定に適したように、いずれかのクックブックを使用してコンプライアンスをセットアップできます。このセクションの手順は opsworks-audit クックブックを実装する方法を示しています。

Compliance プロファイルをインストールするには

1. まだ行っていない場合は [Chef Automate ウェブベースダッシュボードにサインイン](#) します。
AWS OpsWorks for Chef Automate サーバーの作成時に Starter Kit をダウンロードしたときに受け取った認証情報を使用します。
2. Chef Automate ダッシュボードで、[Compliance] タブを選択します。

CHEF AUTOMATE

Nodes Compliance Workflow Admin

opsworks Ops default

Reporting Profile Store

Search profiles...

1 Profiles 88 Available

Select a profile and click "Get" to install.

Get

Profile Title	Version
<input checked="" type="radio"/> DevSec Apache Baseline	2.0.2
<input type="radio"/> CIS AIX 5.3 and AIX 6.1 Benchmark Level 1	1.1.0-3
<input type="radio"/> CIS AIX 5.3 and AIX 6.1 Benchmark Level 2	1.1.0-3
<input type="radio"/> CIS IBM AIX 7.1 Benchmark Level 1	1.1.0-2
<input type="radio"/> CIS IBM AIX 7.1 Benchmark Level 2	1.1.0-2
<input type="radio"/> CIS Amazon Linux 2014.09-2015.03 Benchmark Level 1	1.1.0-3
<input type="radio"/> CIS Amazon Linux 2014.09-2015.03 Benchmark Level 2	1.1.0-3

- 左ナビゲーションバーで、[Profile Store] を選択してから [Available] タブを選択して、事前定義されたプロファイルを表示します。
- プロファイルのリストを参照します。オペレーティングシステムおよび少なくとも1つの管理対象ノードの設定に一致するプロファイルを選択します。プロファイルの対象となる違反の説明および基盤となるルールコードを含むプロファイルの詳細を表示するには、プロファイル項目の右にある [>] を選択します。複数のプロファイルを選択できます。

CHEF AUTOMATE

Nodes Compliance Workflow Admin

opsworks Ops default

Reporting Profile Store

Profiles

Download

DevSec SSH Baseline

Test-suite for best-practice SSH hardening

Getting Started

You will need to install this profile to get started scanning. Get

Status	Available
Version	2.2.0
Author	DevSec Hardening Framework Team
License	Apache 2 license
Platform	unix

68 Controls Total Tests Severity

Control	Total Tests	Severity
ssh-01: client: Check ssh_config owner, group and permissions.	1	CRITICAL (1)

client: Check ssh_config owner, group and permissions.

The ssh_config should owned by root, only be writable by owner and readable to all.

View Code

```
control 'ssh-01' do
  impact 1.0
  title 'client: Check ssh_config owner, group and permissions.'
  desc 'The ssh_config should owned by root, only be writable by owner and readable to all.'
  describe file('/etc/ssh/ssh_config') do
    it { should exist }
  end
end
```

5. 選択されたプロファイルを Chef Automate サーバーにインストールするには、[Get] を選択します。
6. ダウンロードが完了したら、次の手順に進んでください。

opsworks-audit クックブックをインストールしてセットアップする

1. このステップはオプションですが、ノードの実行リストにレシピを追加する場合のステップ 6 で時間の節約になります。AWS OpsWorks for Chef Automate サーバーの作成時にダウンロードしたスターターキットに含まれている `roles/opsworks-example-role.rb` ファイルを編集します。次の行を追加します。コンプライアンススキャンを実行してから非標準ノードを解決するために `ssh-hardening` クックブックとレシピを追加することはオプションなので、最後の行はコメントアウトされています。

```
run_list(
  "recipe[chef-client]",
  "recipe[apache2]",
  "recipe[opsworks-audit]"
  # "recipe[ssh-hardening]"
)
```

2. テキストエディタを使用して Berksfile で希望のクックブックを指定します。サンプル Berksfile はスターターキットに含まれています。この例では、Chef Infra クライアント (`[chef-client]`) クックブック、`apache2`クックブック、`opsworks-audit` クックブックをインストールします。Berksfile は以下ようになります。

```
source 'https://supermarket.chef.io'
cookbook 'chef-client'
cookbook 'apache2', '~> 5.0.1'
cookbook 'opsworks-audit', path: 'cookbooks/opsworks-audit', '~> 1.0.0'
```

すべてのクックブックはクックブックの `metadata.rb` ファイルでバージョンニングされています。クックブックを変更するたびに `metadata.rb` にあるクックブックのバージョンを上げる必要があります。

3. 次のコマンドを実行してローカルまたは使用中のコンピュータにある `cookbooks` フォルダにクックブックをダウンロードしインストールします。

```
berks vendor cookbooks
```

4. 次のコマンドを実行して、AWS OpsWorks for Chef Automate サーバーにベンダーのクックブックをアップロードします。

```
knife upload .
```

5. `opsworks-audit` クックブックがインストールされていることを確認するには、次のコマンドを実行してサーバーで現在使用可能なクックブックのリストを表示します。

```
knife cookbook list
```

6. まだ行っていない場合は、サーバーにノードを追加して管理できるようにします。「[でノードを自動的に追加する AWS OpsWorks for Chef Automate](#)」のステップに従ってノードの関連付けを自動化できます。または「[ノードを個別に追加します](#)」のステップに従ってノードを 1 度に 1 つずつ追加できます。ノードの実行リストを編集して、ステップ 1 の `opsworks-example-role` で指定したロールを追加します。この例では `RUN_LIST` の属性を `userdata` スクリプトで編集します。これはノードの関連性を自動化するために使用します。

```
RUN_LIST="role[opsworks-example-role]"
```

ステップ 1 をスキップしロールを設定しなかった場合は、実行リストに各レシピの名前を追加します。変更を保存して [ステップ 3: 自動関連付けスクリプトを使用してインスタンスを作成する](#) のステップに従って、ユーザーデータスクリプトを Amazon EC2 インスタンスに適用します。

```
RUN_LIST="recipe[chef-client],recipe[apache2],recipe[opsworks-audit]"
```

7. ノードの実行リストを更新すると、`chef-client` エージェントが次回の実行時に、指定したレシピを実行します。この処理はデフォルトで 1,800 秒 (30 分) ごとに行われます。実行後、Chef Automate ダッシュボードでコンプライアンス結果を見ることができます。

コンプライアンススキャンの実行

ノードの実行リストを設定した後初めてエージェントデーモンが実行された後短時間で、Chef Automate ダッシュボードにコンプライアンススキャンの結果が表示されます。

The screenshot shows the Chef Automate interface with the 'Compliance' tab selected. A prominent orange banner at the top reads 'Your System is Not Compliant'. Below this, the 'Overview' section shows '1 Nodes' and '1 Profiles'. A 'Node Status' chart displays a large orange circle with the number '1' inside, labeled 'Total Nodes'. To the right, a legend indicates: 1 Failed Nodes (orange dot), 0 Passed Nodes (blue dot), and 0 Skipped Nodes (grey dot). Further right, a 'Severity of Node Failures' chart shows a horizontal bar for 'CRITICAL' at level 1, with 'MAJOR' and 'MINOR' levels at 0. The left sidebar shows 'Reporting' and 'Profile Store' options.

Chef Automate ダッシュボードで、[Compliance] タブを選択します。左のナビゲーションペインの [Reporting] を選択します。[Profiles] タブを選択して [Scan Results] を選択した後、スキャンによって非準拠と見なされたノードを選択して、そのノードがどのルールに対して非準拠になったかを調べます。

This screenshot shows the 'Scan Results' view in the Chef Automate Compliance dashboard. It features a table with columns for 'Nodes', 'Platform', 'Environment', 'Last Scan', and 'Control Failures'. The first row shows a node with ID 'i-009' on the 'amazon' platform in the '_default' environment, scanned '15 minutes ago', with a status of '61 FAILED'. A pagination bar at the bottom indicates '1' of 1 items.

通常、新しいノードは DevSec SSH ベースラインプロファイルのすべてのルールをまだ満たしていないため、非準拠のスキャン結果が表示されます。コミュニティベースのプロジェクトである

[DevSec Hardening Framework](#) は、DevSec SSH ベースラインプロファイルのルールに違反する問題を修正するためのクックブックを提供しています。

(オプション) 非準拠結果の解決

スターターキットにはオープンソースのクックブックが含まれており ssh-hardening、DevSec SSH ベースラインプロファイルに対する実行の非準拠の結果を修正するために実行できます。

Note

ssh-hardening クックブックは、DevSec SSH ベースラインルールに準拠するようにノードを変更します。このクックブックを本番稼働用ノードで実行する前に、Chef Automate コンソールで DevSec SSH ベースラインプロファイルの詳細を確認して、クックブックがターゲットとするルール違反を理解してください。本稼働ノードでこれを実行する前に、オープンソースの [ssh-hardening](#) クックブックに関する情報を確認します。

ssh-hardening クックブックを実行するには

1. テキストエディタで、ssh-hardening クックブックを Berksfile に追加します。Berksfile は以下のようになります。

```
source 'https://supermarket.chef.io'
  cookbook 'chef-client'
  cookbook 'apache2', '~> 5.0.1'
  cookbook 'opsworks-audit', path: 'cookbooks/opsworks-audit', '~> 1.0.0' #
optional
  cookbook 'ssh-hardening'
```

2. 次のコマンドを実行して ssh-hardening クックブックをローカルのクックブックフォルダにダウンロードし、AWS OpsWorks for Chef Automate サーバーにアップロードします。

```
berks vendor cookbooks
knife upload .
```

3. ノードの実行リストに ssh-hardening レシピを追加します。この操作はステップ 1 とステップ 6 ([opsworks-audit クックブックをインストールしてセットアップする](#)) で説明されています。

opsworks-example-role.rb ファイルを更新するには、次のコマンドを実行してサーバーに変更をアップロードします。

```
knife upload .
```

実行リストを直接更新するには、次のコマンドを実行して変更をアップロードします。通常、ノード名はインスタンス ID になります。

```
knife node run_list add <node name> 'recipe[ssh-hardening]'
```

4. chef-client クックブックを使用しているため、ノードが定期的にチェックインします (デフォルトでは 30 分ごと)。次のチェックインでは、ssh-hardening クックブックが実行され、DevSec SSH ベースラインプロファイルのルールを満たすためにノードセキュリティが向上します。
5. ssh-hardening クックブックの初回実行が完了したら 30 分待機し再びコンプライアンススキャンを実行します。Chef Automate ダッシュボードで結果を表示します。DevSec SSH ベースラインスキャンの初回実行時に発生した非準拠の結果は解決する必要があります。

Compliance の更新

AWS OpsWorks for Chef Automate サーバーでは、スケジュールされた[システムメンテナンス](#)によってコンプライアンス機能が自動的に更新されます。Chef Automate、Chef Infra Server、Chef の更新リリースがサーバーで AWS OpsWorks for Chef Automate 利用可能 InSpec になると、サーバーで実行されている Audit クックブックと Chef InSpec gem のサポートされているバージョンを確認して更新する必要がある場合があります。AWS OpsWorks for Chef Automate サーバーに既にインストールされているプロファイルは、メンテナンスの一環として更新されません。

コミュニティとカスタム Compliance プロファイル

Chef には現在 100 種類を超えるコンプライアンススキャンプロファイルが含まれています。コミュニティとカスタムプロファイルをリストに追加して、含まれているプロファイルと同様にそれらのプロファイルに基づいてコンプライアンススキャンをダウンロードして実行できます。コミュニティベースの Compliance プロファイルは [Chef Supermarket](#) から入手できます。カスタムプロファイルは Ruby ベースのプログラムであり、スキャンルールを指定するコントロールのフォルダが含まれています。

以下の資料も参照してください。

- [Chef Compliance 発表ブログ投稿](#)
- [Chef Automate Compliance オンライントレーニング](#)
- [Chef InSpec ウェブサイト](#)
- [Chef InSpec チュートリアル](#)

AWS OpsWorks for Chef Automate サーバーからノードの関連付けを解除する

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、AWS OpsWorks for Chef Automate サーバーによる管理からマネージドノードの関連付けを解除または削除する方法について説明します。このオペレーションはコマンドラインで実行されます。AWS OpsWorks for Chef Automate マネジメントコンソールでノードの関連付けを解除することはできません。現在、AWS OpsWorks for Chef Automate API では複数のノードをバッチ削除することはできません。このセクションのコマンドでは、一度に 1 つのノードの関連付けを切り離します。

Chef サーバーを削除する場合は、ノードがサーバーへの再接続を試行することなく動作を続行できるように、サーバーとノードとの関連付けを解除することをお勧めします。これを行うには、[disassociate-node](#) AWS CLI コマンドを実行します。

ノードの関連付けを切り離すには

1. で AWS CLI、次のコマンドを実行してノードの関連付けを解除します。[*Node_name*] (ノードネーム) は関連付けを解除するノードの名前です。Amazon EC2 インスタンスの場合、これはインスタンス ID になります。*Server_name* は、ノードとの関連付けを解除する Chef サーバーの名前です。--engine-attributes にはデフォルトの CHEF_AUTOMATE_ORGANIZATION 名を指定します。これら 3 つのパラメータはすべて必須です。

デフォルトのリージョンにない Chef サーバーからノードの関連付けを解除しない限り、`--region` パラメータは必須ではありません。

```
aws opsworks-cm --region Region_name disassociate-node --node-name Node_name --server-name Server_name --engine-attributes "Name=CHEF_AUTOMATE_ORGANIZATION,Value='default'"
```

コマンドの例を次に示します。

```
aws opsworks-cm --region us-west-2 disassociate-node --node-name i-0010zzzz00d66zzzz90 --server-name opsworkstest --engine-attributes "Name=CHEF_AUTOMATE_ORGANIZATION,Value='default'"
```

2. 関連付けの解除が完了したことを示す応答メッセージが表示されるまで待ちます。

AWS OpsWorks for Chef Automate サーバーからノードの関連付けを正常に解除した後も、Chef Automate ダッシュボードに表示されることがあります。Chef はデフォルトでノード状態情報の保持期間を適用し、数日後にはノードを自動消去します。

AWS OpsWorks for Chef Automate サーバーを削除する方法の詳細については、「」を参照してください [AWS OpsWorks for Chef Automate サーバーを削除する](#)。

関連トピック

次の AWS ブログ記事では、Auto Scaling グループを使用するか、複数のアカウント内でノードを Chef Automate サーバーに自動的に関連付ける方法について詳しく説明します。

- [Chef Automate OpsWorks での AWS を使用した Auto Scaling による EC2 インスタンスの管理](#)
- [OpsWorks Chef Automate の - 異なるアカウントでノードを自動的にブートストラップする](#)

AWS OpsWorks for Chef Automate サーバーを削除する

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリュー

シヨンに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、AWS OpsWorks for Chef Automate サーバーを削除する方法について説明します。サーバーを削除すると、サーバーに保存されていたイベント、ログ、およびクックブックも削除されます。サポートするリソース (Amazon Elastic Compute Cloud インスタンス、Amazon Elastic Block Store ボリュームなど) やすべての自動バックアップも削除されます。

サーバーを削除してもノードは削除されませんが、これらは削除されたサーバーによって管理されなくなり、継続的に再接続が試行されます。このため、Chef サーバーを削除する場合は、管理されているノードの関連付けを事前に解除することをお勧めします。このリリースでは、AWS CLI コマンドを実行してノードの関連付けを解除できます。

ステップ 1: 管理されているノードの関連付けを解除する

Chef サーバーを削除する場合は、ノードがサーバーへの再接続を試行することなく動作を続行できるように、サーバーとノードとの関連付けを解除します。これを行うには、[disassociate-node](#) AWS CLI コマンドを実行します。

ノードの関連付けを切り離すには

1. `aws cli`、次のコマンドを実行してノードの関連付けを解除します。*Server_name* は、ノードが関連付けられている Chef サーバーの名前です。

```
aws opsworks-cm --region Region_name disassociate-node --node-name Node_name --server-name Server_name
```

2. 関連付けの解除が完了したことを示す応答メッセージが表示されるまで待ちます。

ステップ 2: サーバーを削除する

1. ダッシュボードのサーバータイトルで、[Actions] メニューを展開します。
2. [Delete server] を選択します。
3. 削除の確認を求められたら、[Yes] を選択します。

Chef Automate ダッシュボードの認証情報のリセット

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Chef Automate ダッシュボードへのサインインに使用するパスワードを定期的に変更したい場合があります。また、このセクションに示す Amazon EC2 Systems Manager AWS CLI コマンドを使用して、Chef Automate ダッシュボードのパスワードを紛失した場合に変更することもできます。使用するコマンドは、Chef Automate サーバーが Chef Automate のバージョン 1 とバージョン 2 のどちらを実行しているかによって異なります。

1. Chef サーバーのインスタンス ID を返すには、AWS Management Console を開いて次のページを開きます。

```
https://console.aws.amazon.com/ec2/v2/home?region=region_of_your_server
#Instances:search=aws-opsworks-cm-server_name
```

例えば、米国西部 (オレゴン) リージョン MyChefServer の という名前の Chef サーバーの場合、コンソール URL は次のようになります。

```
https://console.aws.amazon.com/ec2/v2/home?region=us-west-2#Instances:search=aws-opsworks-cm-MyChefServer
```

コンソールに表示されるインスタンス ID を書き留めておきます。パスワードを変更する際にこの値が必要になります。

2. Chef Automate ダッシュボードのサインインパスワードをリセットするには、サーバーが Chef Automate 1 を実行しているか、Chef Automate 2 を実行しているかに応じて、次の AWS CLI コマンドのいずれかを実行します。[*enterprise_name*] を自分のエンタープライズ名または組織名に、[*user_name*] をサーバーの管理者の IAM ユーザー名に、[*new_password*] を使用するパスワードに、[*region_name*] をサーバーが配置されているリージョンに、それぞれ置き換えます。エンタープライズの名前を指定していない場合、エンタープライズ名は default になります。デフォルトでは、*enterprise_name* は、default (常にプロビジョニングされる組織の名前) です。*user_name ####* は という名前のユーザー AWS OpsWorks for Chef

Automate のみを作成します admin。新しいパスワードを書き留めて、安全かつ便利な場所に保存します。

Chef Automate 1 の場合:

```
aws ssm send-command --document-name "AWS-RunShellScript" --comment "reset admin password" --instance-ids "instance_id" --parameters commands="sudo delivery-ctl reset-password enterprise_name user_name new_password" --region region_name --output text
```

Chef Automate 2 の場合:

```
aws ssm send-command --document-name "AWS-RunShellScript" --comment "reset admin password" --instance-ids "instance_id" --parameters commands="sudo chef-automate iam admin-access restore new_password" --region region_name --output text
```

3. パスワード変更が完了したことを示す出力テキスト (この場合はコマンド ID) が表示されるのを待ちます。

を使用した AWS OpsWorks for Chef Automate API コールのログ記録 AWS CloudTrail

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks for Chef Automate は AWS CloudTrail、IAM アイデンティティによって実行されたアクションを記録するサービスであると統合されています AWS OpsWorks for Chef Automate。は、AWS OpsWorks for Chef Automate コンソールからの呼び出しや API AWS へのコード呼び出しを含む、のすべての API 呼び出しをイベント AWS OpsWorks for Chef Automate として CloudTrail キャプチャします。AWS OpsWorks for Chef Automate APIs 証跡を作成する場合は、の CloudTrail イベントなど、Amazon S3 バケットへのイベントの継続的な配信を有効にすることができます AWS OpsWorks for Chef Automate。証跡を設定しない場合でも、CloudTrail コンソールのイベント履歴

で最新のイベントを表示できます。によって収集された情報を使用して CloudTrail、 に対して行われたリクエスト AWS OpsWorks for Chef Automate、 リクエスト元の IP アドレス、 リクエスト者、 リクエスト日時などの詳細を確認できます。

の詳細については CloudTrail、 「 [AWS CloudTrail ユーザーガイド](#)」を参照してください。

AWS OpsWorks for Chef Automate の情報 CloudTrail

CloudTrail AWS アカウントを作成すると、 がアカウントで有効になります。でアクティビティが発生すると AWS OpsWorks for Chef Automate、 そのアクティビティは CloudTrail イベント履歴 の他の AWS サービスイベントとともにイベントに記録されます。AWS アカウントで最近のイベントを表示、 検索、 ダウンロードできます。詳細については、 「[イベント履歴を使用した CloudTrail イベントの表示](#)」を参照してください。

のイベントなど、 AWS アカウント内のイベントの継続的な記録については AWS OpsWorks for Chef Automate、 証跡を作成します。証跡により CloudTrail、 はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、 コンソールで追跡を作成するときに、 追跡がすべてのリージョンに適用されます。証跡は、 AWS パーティション内のすべてのリージョンからのイベントをログに記録し、 指定した Amazon S3 バケットにログファイルを配信します。さらに、 CloudTrail ログで収集されたイベントデータをより詳細に分析し、 それに基づいて行動するように、 他の AWS サービスを設定できます。詳細については、 以下をご覧ください。

- [証跡を作成するための概要](#)
- [CloudTrail サポートされているサービスと統合](#)
- [の Amazon SNS 通知の設定 CloudTrail](#)
- [複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからの CloudTrail ログファイルの受信](#)

すべての AWS OpsWorks for Chef Automate アクションは によってログに記録 CloudTrail され、 [AWS OpsWorks for Chef Automate API リファレンス](#) に記載されています。例えば、 [CreateServer](#)、 および [DescribeServers](#) アクションを呼び出すと [CreateBackup](#)、 CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、 誰がリクエストを生成したかという情報が含まれます。同一性情報は次の判断に役立ちます。

- リクエストが、 ルートと IAM ユーザー認証情報のどちらを使用して送信されたか。

- リクエストがロールまたはフェデレーションユーザーの一時的なセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、[CloudTrail userIdentity Element](#) を参照してください。

AWS OpsWorks for Chef Automate ログファイルエントリについて

証跡は、指定した Amazon S3 バケットにイベントをログファイルとして配信できるようにする設定です。CloudTrail ログファイルには 1 つ以上のログエントリが含まれます。イベントは任意のソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。CloudTrail ログファイルはパブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次の例は、アクションの CloudTrail AWS OpsWorks for Chef Automate CreateServer ログエントリを示しています。

```
{"eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "ID number:OpsWorksCMUser",
    "arn": "arn:aws:sts::831000000000:assumed-role/Admin/OpsWorksCMUser",
    "accountId": "831000000000", "accessKeyId": "ID number",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2017-01-05T22:03:47Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "ID number",
        "arn": "arn:aws:iam::831000000000:role/Admin",
        "accountId": "831000000000",
        "userName": "Admin"
      }
    }
  },
  "eventTime": "2017-01-05T22:18:23Z",
  "eventSource": "opsworks-cm.amazonaws.com",
  "eventName": "CreateServer",
  "awsRegion": "us-west-2",
```

```
"sourceIPAddress":"101.25.190.51",
"userAgent":"console.amazonaws.com",
"requestParameters":{
  "serverName":"OpsChef-test-server",
  "engineModel":"Single",
  "engine":"Chef",
  "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-cm-ec2-role",
  "backupRetentionCount":3,"serviceRoleArn":"arn:aws:iam::831000000000:role/service-role/aws-opsworks-cm-service-role",
  "engineVersion":"12",
  "preferredMaintenanceWindow":"Fri:21:00",
  "instanceType":"t2.medium",
  "subnetIds":["subnet-1e111f11"],
  "preferredBackupWindow":"Wed:08:00"
},
"responseElements":{
  "server":{
    "endpoint":"OpsChef-test-server-thohsgreckcnwgz3.us-west-2.opsworks-cm.io",
    "createdAt":"Jan 5, 2017 10:18:22 PM",
    "serviceRoleArn":"arn:aws:iam::831000000000:role/service-role/aws-opsworks-cm-service-role",
    "preferredBackupWindow":"Wed:08:00",
    "status":"CREATING",
    "subnetIds":["subnet-1e111f11"],
    "engine":"Chef",
    "instanceType":"t2.medium",
    "serverName":"OpsChef-test-server",
    "serverArn":"arn:aws:opsworks-cm:us-west-2:831000000000:server/OpsChef-test-server/8epp7f6z-e91f-4z10-89z5-8c6219cdb09f",
    "engineModel":"Single",
    "backupRetentionCount":3,
    "engineAttributes":[
      {"name":"CHEF_STARTER_KIT","value":"*** Redacted ***"},
      {"name":"CHEF_PIVOTAL_KEY","value":"*** Redacted ***"},
      {"name":"CHEF_DELIVERY_ADMIN_PASSWORD","value":"*** Redacted ***"}],
    "engineVersion":"12.11.1",
    "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-cm-ec2-role",
    "preferredMaintenanceWindow":"Fri:21:00"
  }
},
"requestID":"de7f64f9-d394-12ug-8081-7bb0386fbc6",
"eventID":"8r7b18df-6c90-47be-87cf-e8346428cfc3",
```

```
"eventType": "AwsApiCall",
"recipientAccountId": "831000000000"
}
```

トラブルシューティング AWS OpsWorks for Chef Automate

Important

AWS OpsWorks for Chef Automate は 2024 年 5 月 5 日にサポートが終了し、新規および既存のお客様の両方で無効になっています。既存のお客様は、Chef SaaS または代替ソリューションに移行することをお勧めします。ご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このトピックには、いくつかの一般的な AWS OpsWorks for Chef Automate 問題と、それらの問題に対する推奨される解決策が含まれています。

トピック

- [一般的なトラブルシューティングのヒント](#)
- [特定のエラーのトラブルシューティング](#)
- [その他のヘルプとサポート](#)

一般的なトラブルシューティングのヒント

Chef サーバーを作成または操作できない場合は、エラーメッセージやログを表示すると、問題のトラブルシューティングに役立ちます。以下のタスクでは、Chef サーバーに関する問題のトラブルシューティングを行うときの一般的な開始点を示します。特定のエラーと解決方法については、このトピックの「[特定のエラーのトラブルシューティング](#)」セクションを参照してください。

- Chef サーバーの起動に失敗した場合のエラーメッセージを表示するには、AWS OpsWorks for Chef Automate コンソールを使用します。Chef サーバーの詳細ページで、サーバーの起動と実行に関連するエラーメッセージが、ページの一番上に表示されます。エラーは AWS OpsWorks for Chef Automate、Chef サーバーの作成に使用される AWS CloudFormation、または Amazon EC2 のサービスから発生する可能性があります。詳細ページでは、実行中のサーバーで発生するイベントを表示することもできます。これには、障害イベントメッセージが含まれる場合があります。

- EC2 に関する問題を解決するため、SSH を使用してサーバーのインスタンスに接続してログを表示します。EC2 インスタンスのログは /var/log/aws/opsworks-cm ディレクトリに保存されています。これらのログは、[Chef サーバー AWS OpsWorks for Chef Automate](#) を起動している間のコマンド出力をキャプチャします。

特定のエラーのトラブルシューティング

トピック

- [サーバーの状態は \[接続が失われました\]](#)
- [フルマネージドログ管理ノードが、欠落している列の Chef Automate ダッシュボードに表示される](#)
- [Chef ボールトを作成できず、knife vault コマンドがエラーで失敗する](#)
- [サーバーの作成が「リクエストされた設定は現在サポートされていません」というメッセージで失敗する](#)
- [Chef サーバーが、Chef Automate ダッシュボードで追加された組織名を認識できない](#)
- [サーバーの Amazon EC2 インスタンスを作成できない](#)
- [サービスロールのエラーによりサーバーを作成できない](#)
- [Elastic IP アドレスの制限を超えた](#)
- [Chef Automate ダッシュボードにサインインできない](#)
- [ノードの自動関連付けに失敗する](#)
- [システムメンテナンスの失敗](#)

サーバーの状態は [接続が失われました]

問題: サーバーのステータスに[接続が失われました]と表示される。

原因: これは、[以外](#)のエンティティ AWS OpsWorks が AWS OpsWorks for Chef Automate サーバーまたはそのサポートリソースに変更を加えた場合に最もよく発生します。AWS OpsWorks は、接続が失われた状態で Chef Automate サーバーに接続して、バックアップの作成、オペレーティングシステムパッチの適用、Chef Automate の更新などのメンテナンスタスクを処理することはできません。その結果、サーバーが重要なアップデートを見逃したり、セキュリティ上の問題の影響を受けやすくなったり、またはその他の理由で期待どおりに動作しない可能性があります。

解決策: 次の手順を試して、サーバーの接続を復元してください。

1. サービスロールに必要な権限がすべてあることを確認してください。
 - a. サーバーの「設定」ページの「ネットワークとセキュリティ」で、サーバーが使用しているサービスロールへのリンクを選択します。これにより、IAM コンソールに表示されるサービスロールが開きます。
 - b. 「アクセス許可」のタブで、AWSOpsWorksCMServiceRole が「アクセス許可ポリシー」リストに含まれていることを確認します。リストにない場合は、AWSOpsWorksCMServiceRole 管理ポリシーをロールに手動で追加してください。
 - c. 「信頼関係」のタブで、ユーザーに代わって役割を引き受ける opsworks-cm.amazonaws.com サービスを信頼する信頼ポリシーがサービスロールに含まれていることを確認します。ロールで信頼ポリシーを使用する方法の詳細については、[「ロールの変更 \(コンソール\)」](#) または AWS 「セキュリティブログ記事」の [「IAM ロールで信頼ポリシーを使用する方法」](#) を参照してください。
2. インスタンスプロファイルに必要な権限がすべてあることを確認してください。
 - a. サーバーの「設定」ページの「ネットワークとセキュリティ」で、サーバーが使用しているインスタンスプロファイルへのリンクを選択します。これにより、IAM コンソールに表示されるインスタンスプロファイルが開きます。
 - b. 「権限」のタブで、AmazonEC2RoleforSSM と AWSOpsWorksCMInstanceProfileRole が両方とも「アクセス権限ポリシー」リストに含まれていることを確認します。一方または両方がリストにない場合は、これらの管理ポリシーを手動でロールに追加してください。
 - c. 「信頼関係」のタブで、ユーザーに代わって役割を引き受ける ec2.amazonaws.com サービスを信頼する信頼ポリシーがサービスロールに含まれていることを確認します。ロールで信頼ポリシーを使用する方法の詳細については、[「ロールの変更 \(コンソール\)」](#) または AWS 「セキュリティブログ記事」の [「IAM ロールで信頼ポリシーを使用する方法」](#) を参照してください。
3. Amazon EC2 コンソールで、AWS OpsWorks for Chef Automate サーバーのリージョンと同じリージョンにいることを確認し、サーバーが使用している EC2 インスタンスを再起動します。
 - a. aws-opsworks-cm-instance-*server-name* という名前の EC2 インスタンスを選択します。
 - b. [インスタンスの状態]メニューで、[インスタンスの再起動] を選択します。
 - c. サーバーが再起動して完全にオンラインになるまで最大 15 分かかります。
4. AWS OpsWorks for Chef Automate コンソールのサーバーの詳細ページで、サーバーのステータスが正常になったことを確認します。

上記の手順を実行してもサーバーステータスが [接続が失われました] のままの場合は、次のいずれかを試してください。

- 新しいサーバーを作成し、元のサーバーを削除して、サーバーを交換してください。現在のサーバー上のデータが重要な場合は、最新のバックアップからサーバーを復元し、元の応答しないサーバーを削除する前にデータが最新であることを確認してください。
- [AWS Support](#)にお問い合わせください。

フルマネージドログ管理ノードが、欠落している列の Chef Automate ダッシュボードに表示される

問題: フルマネージドログ管理ノードが、Chef Automate ダッシュボードの欠落している列に表示される。

原因: ノードが、12 時間以上 Chef Automate サーバーに接続されず、chef-client かつノードで実行できない場合、ノードは、12 時間前の状態から変わり、Chef Automate ダッシュボードの欠落している列に移動します。

解決策: ノードがオンラインになっていることを確認してください。knife node show *node_name* --run-list を実行してノードで chef-client を実行できるか、または knife node show -l *node_name* がノードに関するすべての情報を表示するかを確認します。ノードがオフライン、あるいはネットワーク接続が切断している可能性があります。

Chef ボールトを作成できず、knife vault コマンドがエラーで失敗する

問題: knife vault コマンドを実行して、Chef Automate サーバーでボールトを作成しようとしている (ドメイン結合 Windows ベースのノードの認証情報を保存するボールトなど)。このコマンドは、次のようなエラーメッセージを返す。

```
WARN: Auto inflation of JSON data is deprecated. Please pass in the class to inflate or use #edit_hash (CHEF-1)
at /opt/chefdk/embedded/lib/ruby/2.3.0/forwardable.rb:189:in `edit_data'.Please see https://docs.chef.io/deprecations_json_auto_inflate.html for further details and information on how to correct this problem.
WARNING: pivotal not found in users, trying clients.
ERROR: ChefVault::Exceptions::AdminNotFound: FATAL: Could not find pivotal in users or clients!
```

`knife user list` をリモートに実行すると主要ユーザーは返されないが、Chef Automate サーバーで `chef-server-ctl user-show` コマンドをローカルに実行すると、結果で主要ユーザーが表示される。つまり、`knife vault` コマンドでは主要ユーザーを見つけることはできないが、ユーザーが存在していることは確認できる。

原因: 主要ユーザーは、Chef ではスーパーユーザーと見なされており、完全な権限を持っていますが、AWS OpsWorks for Chef Automate で使用されている `default` 組織を含めて、いずれの組織のメンバーでもありません。コマンド `knife user list` は、Chef 設定の現在の組織に属するすべてのユーザーを返します。コマンド `chef-server-ctl user-show` は、主要ユーザーを含めて、組織にかかわらずすべてのユーザーを返します。

解決策: この問題を解決するには、`knife opc` を実行して、デフォルトの組織に主要ユーザーを追加します。

最初に、[knife-opc](#) プラグインをインストールする必要があります。

```
chef gem install knife-opc
```

このプラグインをインストールしたら、次のコマンドを実行して主要ユーザーをデフォルトの組織に追加します。

```
knife opc org user add default pivotal
```

もう一度 `knife user list` を実行して、主要ユーザーがデフォルトの組織に属していることを確認できます。結果には `pivotal` が表示されている必要があります。次に、もう一度 `knife vault` を実行してみます。

サーバーの作成が「リクエストされた設定は現在サポートされていません」というメッセージで失敗する

問題: Chef Automate サーバーの作成を試みているが、「リクエストされた設定は現在サポートされていません。サポートされている設定についてドキュメントをご確認ください」のようなエラーメッセージでサーバーの作成が失敗する。

原因: Chef Automate サーバーに対してサポートされていないインスタンスタイプが指定された可能性があります。[ハードウェア専有インスタンス](#)用など、デフォルトでないテナンシーを持つ VPC で Chef Automate サーバーの作成を選択した場合、指定された VPC 内のすべてのインスタンスも、専有テナンシーまたはホストテナンシーのインスタンスである必要があります。t2 など一部のインスタンスタイプはデフォルトテナンシーでしか使用できないため、Chef Automate サーバーインスタン

スタイプは指定された VPC でサポート可能でない場合があります、そのためにサーバーの作成が失敗します。

解決策: デフォルト以外のテナンシーを持つ VPC を選択した場合は、専用テナンシーをサポートできる m4 インスタンスタイプを使用します。

Chef サーバーが、Chef Automate ダッシュボードで追加された組織名を認識できない

問題: Chef Automate ダッシュボードで新しいワークフロー組織名を追加したり、[無人ノード関連付けスクリプト](#)で "default" 以外の CHEF_AUTOMATE_ORGANIZATION 値を指定したりしましたが、ノード関連付けに失敗します。AWS OpsWorks for Chef Automate サーバーが新しい組織名を認識しない。

原因: Workflow 組織名および Chef サーバー組織名が同じではありません。ウェブベースの Chef Automate ダッシュボードで新しい Workflow 組織を作成できますが、Chef サーバー組織名は作成できません。Chef Automate ダッシュボードのみを使用して、既存の Chef サーバー組織を表示できます。Chef Automate ダッシュボードで作成する新しい組織は Workflow 組織であり、Chef サーバーによって認識されません。ノードの関連付けスクリプトでそれらを指定して、新しい組織名を作成することはできません。組織が最初に Chef サーバーに追加されていないときにノードの関連付けスクリプトで組織名を参照すると、ノードの関連付けに失敗します。

解決策: Chef Server で認識される新しい組織を作成するには、[knife opc org create](#) コマンドを使用するか、[chef-server-ctl org-create](#) を実行します。

サーバーの Amazon EC2 インスタンスを作成できない

問題: サーバーの作成が、次のようなエラーメッセージにより失敗する。「The following resource(s) failed to create: [EC2Instance]. Failed to receive 1 resource signal(s) within the specified duration」

原因: この問題のほとんどが、EC2 インスタンスでネットワークアクセスが可能でないことが原因です。

解決策: インスタンスにアウトバウンドインターネットアクセスがあり、AWS サービスエージェントがコマンドを発行できることを確認します。VPC (単一のパブリックサブネットを持つ VPC) で DNS 解決が有効になっていて、サブネットで [パブリック IP の自動割り当て] 設定が有効になっていることを確認します。

サービスロールのエラーによりサーバーを作成できない

問題: サーバーの作成が失敗し、「sts: を実行する権限がありません」というエラーメッセージが表示される AssumeRole。

原因: お使いのサービスロールに新しいサーバーを作成するための適切なアクセス権限がない場合に、この問題が発生する可能性があります。

解決策: AWS OpsWorks for Chef Automate コンソールを開き、コンソールを使用して新しいサービスロールとインスタンスプロファイルロールを生成します。独自のサービスロールを使用する場合は、AWSOpsWorksCMServiceRoleポリシーをロールにアタッチします。opsworks-cm.amazonaws.com がロールの信頼関係のサービス間でリストされていることを確認します。Chef サーバーに関連付けられているサービスロールに AWSOpsWorksCMServiceRole管理ポリシーがアタッチされていることを確認します。

Elastic IP アドレスの制限を超えた

問題: 「The following resource(s) failed to create: [EIP, EC2Instance]. Resource creation cancelled, the maximum number of addresses has been reached.」という状態を示すエラーメッセージにより、サーバーを作成できない。

原因: アカウントで Elastic IP (EIP) アドレスの最大数を使用した場合に、この問題が発生します。EIP アドレスのデフォルトの制限は 5 です。

解決策: 既存の EIP アドレスを解放するか、アカウントがアクティブに使用していないアドレスを削除するか、AWS カスタマーサポートに連絡して、アカウントに関連付けられている EIP アドレスの制限を増やすことができます。

Chef Automate ダッシュボードにサインインできない

問題: Chef Automate ダッシュボードに、「Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at https://myserver-name.region.opsworks-cm.io/api/v0/e/default/verify-token. (Reason: CORS header 'Access-Control-Allow-Origin' missing)」のようなエラーが表示される。「The User Id / Password combination entered is incorrect.」のようなエラーの場合もある。

原因: Chef Automate ダッシュボードが明示的に FQDN を設定していて、相対 URL を受け付けていません。現時点では、Chef サーバーの IP アドレスを使用してサインインすることはできません。サーバーの DNS 名を使用してサインインできるのみです。

解決策: Chef サーバーの IP アドレスではなく、DNS 名エントリのみを使用して Chef Automate ダッシュボードにサインインします。また、[Chef Automate ダッシュボードの認証情報のリセット](#)で説明したように、AWS CLI コマンドを実行して Chef Automate ダッシュボードの認証情報をリセットしてみることもできます。

ノードの自動関連付けに失敗する

問題: 新しい Amazon EC2 ノードの自動 (無人) 関連付けに失敗する。Chef サーバーに追加されたはずのノードが Chef Automate ダッシュボードに表示されず、`knife client show` または `knife node show` コマンドの結果に含まれない。

原因: `opsworks-cm` API コールで新しい EC2 インスタンスとの通信を許可するインスタンスプロファイルとして IAM ロールをセットアップしていない場合に、この問題が発生する可能性があります。

解決策: [でノードを自動的に追加する AWS OpsWorks for Chef Automate](#) で説明したように、EC2 インスタンスのプロファイルにポリシーをアタッチして、`AssociateNode` と `DescribeNodeAssociationStatus` の API コールを EC2 と連携できるようにします。

システムメンテナンスの失敗

AWS OpsWorks CM は毎週システムメンテナンスを実行し、セキュリティ更新プログラムを含む最新のマイナーバージョンの Chef Server と Chef Automate Server が、Chef Automate サーバー OpsWorks 用の AWS で常に実行されるようにします。何らかの理由でシステムメンテナンスが失敗した場合、は失敗 AWS OpsWorks CM を通知します。システムメンテナンスの詳細については、「[でのシステムメンテナンス AWS OpsWorks for Chef Automate](#)」を参照してください。

このセクションでは、考えられる障害の原因を説明し、解決策を提案します。

トピック

- [サービスロールまたはインスタンスプロファイルのエラーによって、システムのメンテナンスが妨げられる](#)

サービスロールまたはインスタンスプロファイルのエラーによって、システムのメンテナンスが妨げられる

問題: システムメンテナンスが失敗し、「sts: を実行する権限がありません」というエラーメッセージ `AssumeRole`、またはアクセス許可に関する同様のエラーメッセージが表示される。

原因: これは、使用しているサービスロールまたはインスタンスプロファイルのいずれかに、サーバー上でシステムメンテナンスを実行するための適切な権限がない場合に発生する可能性があります。

解決策: サービスロールとインスタンスプロファイルに必要なすべての権限があることを確認してください。

1. サービスロールに必要な権限がすべてあることを確認してください。
 - a. サーバーの「設定」ページの「ネットワークとセキュリティ」で、サーバーが使用しているサービスロールへのリンクを選択します。これにより、IAM コンソールに表示されるサービスロールが開きます。
 - b. 「アクセス許可」のタブで、AWSOpsWorksCMServiceRole がサービスロールにアタッチされていることを確認します。AWSOpsWorksCMServiceRole がリストにない場合は、このポリシーをロールに追加します。
 - c. opsworks-cm.amazonaws.com がロールの信頼関係のサービス間でリストされていることを確認します。ロールで信頼ポリシーを使用する方法の詳細については、[「ロールの変更 \(コンソール\)」](#) または AWS 「セキュリティブログ記事」の [「IAM ロールで信頼ポリシーを使用する方法」](#) を参照してください。
2. インスタンスプロファイルに必要な権限がすべてあることを確認してください。
 - a. サーバーの「設定」ページの「ネットワークとセキュリティ」で、サーバーが使用しているインスタンスプロファイルへのリンクを選択します。これにより、IAM コンソールに表示されるインスタンスプロファイルが開きます。
 - b. 「権限」のタブで、AmazonEC2RoleforSSM と AWSOpsWorksCMInstanceProfileRole が両方とも「アクセス権限ポリシー」リストに含まれていることを確認します。一方または両方がリストにない場合は、これらの管理ポリシーを手動でロールに追加してください。
 - c. 「信頼関係」のタブで、ユーザーに代わって役割を引き受ける ec2.amazonaws.com サービスを信頼する信頼ポリシーがサービスロールに含まれていることを確認します。ロールで信頼ポリシーを使用する方法の詳細については、[「ロールの変更 \(コンソール\)」](#) または AWS 「セキュリティブログ記事」の [「IAM ロールで信頼ポリシーを使用する方法」](#) を参照してください。

その他のヘルプとサポート

発生している特定の問題がこのトピックで説明されていないか、このトピックの提案を試しても問題が解決しない場合は、[AWS OpsWorks フォーラム](#)を参照してください。

また、[AWS Support Center](#) を参照することもできます。AWS サポートセンターは、サポートケースを作成および管理 AWS するためのハブです。AWS サポートセンターには、フォーラム、技術上のFAQs、サービスヘルスステータス、など、その他の役立つリソースへのリンクも含まれています AWS Trusted Advisor。

セキュリティ AWS OpsWorks 設定管理 (CM)

でのクラウドセキュリティ AWS が最優先事項です。として AWS のお客様は、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャからメリットを得られます。

セキュリティは、間で共有される責任です。AWS とユーザー。[責任共有モデル](#)では、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ – AWS は、が実行するインフラストラクチャを保護する責任があります。AWS の サービス AWS クラウド。AWS は、安全に使用できる サービスも提供します。サードパーティーの監査者は、の一環として、当社のセキュリティの有効性を定期的にテストおよび検証します。[AWS コンプライアンスプログラム](#)。に適用されるコンプライアンスプログラムについて学ぶには AWS OpsWorks CM、「」を参照してください。[AWS コンプライアンスプログラムによる対象範囲内のサービス](#)。
- クラウド内のセキュリティ — お客様の責任は によって決まります。AWS 使用する サービス。また、お客様は、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、の使用時に責任共有モデルを適用する方法を理解するのに役立ちます。AWS OpsWorks CM。以下のトピックでは、を設定する方法を示します。AWS OpsWorks セキュリティとコンプライアンスの目標を達成する CM。また、のモニタリングや保護に役立つ他の AWS のサービスの使用方法についても説明します。AWS OpsWorks CM リソース。

トピック

- [でのデータ保護 AWS OpsWorks CM](#)
- [データ暗号化](#)
- [AWS OpsWorks CM の ID とアクセスの管理](#)
- [インターネットトラフィックのプライバシー](#)
- [でのログ記録とモニタリング AWS OpsWorks CM](#)
- [のコンプライアンス検証 AWS OpsWorks CM](#)
- [の耐障害性 AWS OpsWorks CM](#)
- [のインフラストラクチャセキュリティ AWS OpsWorks CM](#)
- [での設定と脆弱性の分析 AWS OpsWorks CM](#)

• [のセキュリティのベストプラクティス AWS OpsWorks CM](#)

でのデータ保護 AWS OpsWorks CM

- AWS [責任共有モデル](#)、AWS OpsWorks 設定管理のデータ保護に適用されます。このモデルで説明されているように、AWS は、すべての を実行するグローバルインフラストラクチャを保護する責任があります。AWS クラウド。お客様は、このインフラストラクチャでホストされているコンテンツの制御を維持する責任があります。また、 のセキュリティ設定と管理タスクについても責任を負います。AWS のサービス 使用する。データプライバシーの詳細については、「[データプライバシーFAQ](#)」を参照してください。欧州でのデータ保護の詳細については、「」を参照してください。[AWS の責任共有モデルとGDPR](#) ブログ記事 AWS セキュリティブログ。

データ保護の目的で、 を保護することをお勧めします。AWS アカウント 認証情報と を使用して個々のユーザーをセットアップする AWS IAM Identity Center または AWS Identity and Access Management (IAM)。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して と通信する AWS リソースの使用料金を見積もることができます。1TLS.2 が 必要で、1.3 TLS をお勧めします。
- で APIとユーザーアクティビティのログ記録を設定する AWS CloudTrail。CloudTrail 証跡を使用してキャプチャする方法については、「」を参照してください。AWS アクティビティ、「」の [「証 CloudTrail 跡の使用」](#) を参照してください。AWS CloudTrail ユーザーガイド。
- 使用アイテム AWS 暗号化ソリューションと 内のすべてのデフォルトのセキュリティコントロール AWS のサービス。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- アクセス時に FIPS 140-3 検証済みの暗号化モジュールが必要な場合 AWS コマンドラインインターフェイスまたは を介してAPI、FIPSエンドポイントを使用します。利用可能なFIPSエンドポイントの詳細については、「[連邦情報処理規格 \(FIPS\) 140-3](#)」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、 OpsWorks CM または他の を使用する場合も同様です。AWS のサービス コンソール、API、AWS CLI、または AWS SDKs。名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログ

に使用される場合があります。URL を外部サーバーに提供する場合は、そのサーバーへのリクエストを検証URLするために認証情報を に含めないことを強くお勧めします。

OpsWorks CM サーバーの名前は暗号化されません。

OpsWorks CM は、 の作成と保守の過程で以下の顧客データを収集します。 AWS OpsWorks for Chef Automate また、 AWS OpsWorks for Puppet Enterprise サーバー。

- OpsWorks for Puppet Enterprise では、 Puppet Enterprise が Puppet マスターノードとマネージドノード間の通信を有効にするために使用するプライベートキーを収集します。
- [AWS OpsWorks for Chef Automateカスタムドメインを使用している場合、サービスにアタッチした証明書のプライベートキーが収集されます。カスタムドメインを持つ Chef Automate サーバーを作成するときに指定したプライベートキーは、サーバーに渡されます。

OpsWorks CM サーバーは、 Chef クックブックや Puppet Enterprise モジュールなどの設定コードを保存します。このコードはサーバーのバックアップに保存されますが、 AWS にはアクセスできません。このコンテンツは暗号化され、 の管理者のみ AWS アカウントはそれにアクセスできます。ソースリポジトリ用の推奨プロトコルを使用して Chef または Puppet 設定コードを保護することをお勧めします。例えば、 のリポジトリへのアクセス許可を [制限できません。AWS CodeCommit](#)、または [GitHub ウェブサイトの GitHub 「リポジトリのセキュリティ保護に関するガイドライン」](#) に従ってください。

OpsWorks CM は、お客様が提供したコンテンツを使用してサービスを維持したり、顧客ログを保持したりしません。OpsWorks CM サーバーに関するログは、Amazon S3 バケットのアカウントに保存されます。OpsWorks CM サーバーに接続するユーザーの IP アドレスは、によってログに記録されます。AWS.

との統合 AWS Secrets Manager

2021 年 5 月 3 日以降、 OpsWorks CM で新しいサーバーを作成すると、 OpsWorks CM はサーバーのシークレットを に保存します。AWS Secrets Manager。新しいサーバーの場合、次の属性は Secrets Manager にシークレットとして保存されます。

- Chef Automate server (Chef Automateサーバー)
 - HTTPS プライベートキー (カスタムドメインを使用しないサーバーのみ)
 - Chef Automate 管理パスワード (CHEF_AUTOMATEADMIN_PASSWORD)
- Puppet Enterprise master (Puppet Enterprise マスター)
 - HTTPS プライベートキー (カスタムドメインを使用しないサーバーのみ)

- Puppet 管理パスワード (PUPPET_ADMIN_PASSWORD)
- Puppet r10k リモート (PUPPET_R10K_REMOTE)

カスタムドメインを使用しない既存のサーバーの場合、Chef Automate サーバーと Puppet Enterprise サーバーの両方について、Secrets Manager に保存されている唯一のシークレットが HTTPS プライベートキーです。これは、毎週自動システムメンテナンス中に生成されるためです。

OpsWorks CM は Secrets Manager にシークレットを自動的に保存しますが、この動作はユーザーが設定することはできません。

データ暗号化

AWS OpsWorks CM は、サーバーのバックアップと承認された 間の通信を暗号化します。AWS ユーザーとその AWS OpsWorks CM サーバー。ただし、 のルート Amazon EBS ボリュームは AWS OpsWorks CM サーバーは暗号化されません。

保管時の暗号化

AWS OpsWorks CM サーバーのバックアップは暗号化されます。ただし、 のルート Amazon EBS ボリュームは AWS OpsWorks CM サーバーは暗号化されません。これはユーザーが設定できません。

転送時の暗号化

AWS OpsWorks CM は TLS 暗号化 HTTP を使用します。AWS OpsWorks CM は、ユーザーが署名付き証明書を提供しない場合、サーバーをプロビジョニングおよび管理するためにデフォルトで自己署名証明書を使用します。証明機関 (CA) によって署名された証明書を使用することをお勧めします。

キーの管理

AWS Key Management Service カスタマーマネージドキーと AWS マネージドキーは、現在 ではサポートされていません。AWS OpsWorks CM。

AWS OpsWorks CM の ID とアクセスの管理

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御するのに役立つ AWS サービスです。IAM 管理者は、 OpsWorks CM リソースを使用するための認

証 (サインイン) および認可 (アクセス許可を持つ) できるユーザーを制御します。IAM は、追加料金なしで使用できる AWS サービスです。

トピック

- [対象者](#)
- [アイデンティティを使用した認証](#)
- [ポリシーを使用したアクセスの管理](#)
- [AWS OpsWorks CM の仕組み IAM](#)
- [AWS OpsWorks CM アイデンティティベースのポリシーの例](#)
- [AWS OpsWorks CM Identity and Access のトラブルシューティング](#)
- [AWS の マネージドポリシー AWS OpsWorks 設定管理](#)
- [AWS OpsWorks CMにおけるサービス間の混乱した代理の防止](#)

対象者

AWS Identity and Access Management (IAM) の使用方法は、 OpsWorks CM で行う作業によって異なります。

サービスユーザー – OpsWorks CM サービスを使用してジョブを実行する場合、管理者は必要な認証情報とアクセス許可を提供します。より多くの OpsWorks CM 機能を使用して作業を行うと、追加のアクセス許可が必要になる場合があります。アクセスの管理方法を理解しておく、管理者に適切な許可をリクエストするうえで役立ちます。OpsWorks CM の機能にアクセスできない場合は、「」を参照してください[AWS OpsWorks CM Identity and Access のトラブルシューティング](#)。

サービス管理者 – 社内の OpsWorks CM リソースを担当している場合は、おそらく OpsWorks CM へのフルアクセスが可能です。サービスユーザーがどの OpsWorks CM 機能やリソースにアクセスできるかを判断するのは、お客様の仕事です。その後、サービスユーザーのアクセス許可を変更するリクエストをIAM管理者に送信する必要があります。このページの情報を確認して、 の基本概念を理解しますIAM。会社が OpsWorks CM IAMで を使用する方法の詳細については、「」を参照してください[AWS OpsWorks CM の仕組み IAM](#)。

IAM 管理者 – IAM管理者の場合は、 OpsWorks CM へのアクセスを管理するためのポリシーの作成方法の詳細を知りたい場合があります。で使用できる OpsWorks CM ID ベースのポリシーの例を表示するにはIAM、「」を参照してください[AWS OpsWorks CM アイデンティティベースのポリシーの例](#)。

アイデンティティを使用した認証

認証は、アイデンティティ認証情報 AWS を使用して にサインインする方法です。として、IAMユーザーとして AWS アカウントのルートユーザー、またはIAMロールを引き受けて認証 (にサインイン AWS) する必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーテッド ID AWS として にサインインできます。AWS IAM Identity Center (IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーテッド ID の例です。フェデレーテッド ID としてサインインすると、管理者は以前にIAMロールを使用して ID フェデレーションをセットアップしていました。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けます。

ユーザーのタイプに応じて、AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、AWS サインイン ユーザーガイドの「[へのサインイン方法 AWS アカウント](#)」を参照してください。

AWS プログラムで にアクセスする場合、 はソフトウェア開発キット (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号化で署名します。AWS ツールを使用しない場合は、自分でリクエストに署名する必要があります。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、IAM ユーザーガイドの[AWS API 「リクエストの署名バージョン 4」](#)を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、では、アカウントのセキュリティを高めるために多要素認証 (MFA) を使用する AWS ことをお勧めします。詳細については、AWS IAM Identity Center 「ユーザーガイド」の「[多要素認証](#)」とIAM 「ユーザーガイド」の[AWS 「多要素認証IAM」](#)を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての および リソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインイン ID から始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインしてアクセスします。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、IAM ユーザーガイドの「[ルートユーザーの認証情報を必要とするタスク](#)」を参照してください。

IAM ユーザーとグループ

[IAM ユーザー](#)とは、1人のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウントを持つ内の ID です。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を持つIAMユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAMユーザーとの長期的な認証情報を必要とする特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、「ユーザーガイド」の「[長期的な認証情報を必要とするユースケースのアクセスキーを定期的にローテーションする](#)」を参照してください。IAM

[IAM グループ](#)は、IAMユーザーのコレクションを指定する ID です。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、という名前のグループがありIAMAdmins、そのグループにIAMリソースを管理するアクセス許可を付与できます。

ユーザーは、ロールとは異なります。ユーザーは1人の人または1つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時認証情報が提供されます。詳細については、「[ユーザーガイド](#)」のIAM「[ユーザーのユースケース](#)」を参照してください。IAM

Warning

IAM ユーザーには長期的な認証情報があり、セキュリティリスクがあります。このリスクを軽減するために、これらのユーザーにはタスクの実行に必要な権限のみを付与し、不要になったユーザーを削除することをお勧めします。

IAM ロール

[IAM ロール](#)は、特定のアクセス許可 AWS アカウントを持つ内の ID です。ユーザーと似ていますがIAM、特定の人物には関連付けられていません。IAMロールを一時的に引き受けるには AWS Management Console、[ユーザーからIAMロール \(コンソール\) に切り替える](#)ことができます。または AWS API オペレーションを AWS CLI 呼び出すか、カスタムを使用してロールを引き受けることができますURL。ロールを使用する方法の詳細については、IAM ユーザーガイドの「[ロールを引き受ける方法](#)」を参照してください。

IAM 一時的な認証情報を持つロールは、以下の状況で役立ちます。

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロールの詳細については、IAM ユーザーガイドの「[サードパーティー ID プロバイダー \(フェデレーション\) のロールを作成する](#)」を参照してください。IAM Identity Center を使用する場合は、アクセス許可セットを設定します。ID が認証された後にアクセスできる内容を制御するために、IAM Identity Center はアクセス許可セットをのロールに関連付けます IAM。アクセス許可セットの詳細については、「AWS IAM Identity Center User Guide」の「[Permission sets](#)」を参照してください。
- 一時的な IAM ユーザーアクセス許可 - IAM ユーザーまたはロールは、特定のタスクに対してさまざまなアクセス許可を一時的に引き受ける IAM ロールを引き受けることができます。
- クロスアカウントアクセス — IAM ロールを使用して、別のアカウントの誰か (信頼できるプリンシパル) が自分のアカウントのリソースにアクセスすることを許可できます。クロスアカウントアクセスを許可する主な方法は、ロールを使用することです。ただし、一部の AWS のサービス、(プロキシとしてロールを使用する代わりに) リソースに直接ポリシーをアタッチできます。クロスアカウントアクセスのロールとリソースベースのポリシーの違いについては、「[ユーザーガイド](#)」の「[クロスアカウントリソースアクセス IAM](#)」を参照してください。 IAM
- クロスサービスアクセス — 他の の機能 AWS のサービス を使用するものもあります AWS のサービス。例えば、サービスで呼び出しを行うと、そのサービスが Amazon でアプリケーションを実行 EC2 したり、Amazon S3 にオブジェクトを保存したりするのが一般的です。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
- 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して でアクションを実行する場合 AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス または リソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するための権限が必要です。FAS リクエストを行う際のポリシーの詳細については、「[アクセスセッションの転送](#)」を参照してください。
- サービスロール – サービスロールは、ユーザーに代わってアクションを実行するためにサービスが引き受ける [IAM ロール](#) です。IAM 管理者は、内からサービスロールを作成、変更、削除できます IAM。詳細については、「[ユーザーガイド](#)」の「[アクセス許可を に委任するロールを作成する AWS のサービス](#)」を参照してください。 IAM

- サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント、 サービスによって所有されます。IAM 管理者はサービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。
- Amazon で実行されているアプリケーション EC2 – IAMロールを使用して、EC2インスタンスで実行され、AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2インスタンス内にアクセスキーを保存するよりも望ましいです。EC2インスタンスに AWS ロールを割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれており、EC2インスタンスで実行されているプログラムが一時的な認証情報を取得できるようにします。詳細については、「[ユーザーガイド](#)」のIAM「[ロールを使用して Amazon EC2インスタンスで実行されているアプリケーションにアクセス許可を付与する](#)」を参照してください。IAM

ポリシーを使用したアクセスの管理

でアクセスを制御するには、ポリシー AWS を作成し、AWS アイデンティティまたはリソースにアタッチします。ポリシーは AWS、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義するオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーはJSONドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、IAM「[ユーザーガイド](#)」のJSON「[ポリシーの概要](#)」を参照してください。

管理者はポリシーを使用して AWS JSON、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。必要なリソースに対してアクションを実行するアクセス許可をユーザーに付与するには、IAM管理者はIAMポリシーを作成できます。その後、管理者はIAMポリシーをロールに追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行に使用する方法に関係なく、アクションのアクセス許可を定義します。例えば、iam:GetRole アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、AWS Management Console、AWS CLI または AWS からロール情報を取得できますAPI。

アイデンティティベースのポリシー

ID ベースのポリシーは、IAMユーザー、ユーザーのグループ、ロールなどの ID にアタッチできる JSONアクセス許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。ID ベースのポリシーを作成する方法については、IAM「ユーザーガイド」の[「カスタマーマネージドポリシーを使用したカスタムIAMアクセス許可の定義」](#)を参照してください。

アイデンティティベースポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。マネージドポリシーは、内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。マネージドポリシーには、AWS マネージドポリシーとカスタマーマネージドポリシーが含まれます。マネージドポリシーまたはインラインポリシーを選択する方法については、IAM ユーザーガイドの[「マネージドポリシーとインラインポリシーの選択」](#)を参照してください。

OpsWorks CM は、で作成IAMし、ユーザー、ロール、またはグループにアタッチするカスタムポリシーをサポートしています。

リソースベースのポリシー

リソースベースのポリシーは、リソースにアタッチするJSONポリシードキュメントです。リソースベースのポリシーの例としては、IAMロール信頼ポリシーと Amazon S3 バケットポリシー があります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、[プリンシパルを指定する](#)必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーテッドユーザー、を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーIAMでは、から AWS 管理ポリシーを使用することはできません。

OpsWorks CM はリソースベースのポリシーをサポートしていません。

アクセスコントロールリスト (ACLs)

アクセスコントロールリスト (ACLs) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするアクセス許可を持っているかを制御します。ACLs はリソースベースのポリシーに似ていますが、JSONポリシードキュメント形式は使用しません。

Amazon S3、および Amazon VPCは AWS WAF、 をサポートするサービスの例ですACLs。の詳細についてはACLs、「Amazon Simple Storage Service デベロッパーガイド」の[「アクセスコントロールリスト \(ACL\) 概要」](#)を参照してください。

OpsWorks CM は を使用しませんACLs。

その他のポリシータイプ

OpsWorks CM は、以下の他のポリシータイプをサポートしていません。

AWS は、追加であり一般的ではないポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- **アクセス許可の境界** – アクセス許可の境界は、アイデンティティベースのポリシーがIAMエンティティ (ユーザーまたはロール) に付与できる最大アクセス許可を設定する高度な機能です。エンティティに許可の境界を設定できます。結果として許可される範囲は、エンティティのアイデンティティベースポリシーとその許可の境界の共通部分になります。Principal フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、IAM「ユーザーガイド」の[IAM「エンティティのアクセス許可の境界」](#)を参照してください。
- **サービスコントロールポリシー (SCPs)** – SCPs は、 の組織または組織単位 (OU) の最大アクセス許可を指定するJSONポリシーです AWS Organizations。AWS Organizations は、ビジネスが所有する複数の AWS アカウントをグループ化して一元管理するためのサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCPs) を任意のアカウントまたはすべてのアカウントに適用できます。は、各 を含むメンバーアカウントのエンティティのアクセス許可SCPを制限します AWS アカウントのルートユーザー。Organizations との詳細については SCPs、AWS Organizations「ユーザーガイド」の[SCPs「仕組み」](#)を参照してください。
- **セッションポリシー** – セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合があります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「ユーザーガイド」の[「セッションポリシー」](#)を参照してください。IAM

複数のポリシータイプ

1つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「ユーザーガイド」の[「ポリシー評価ロジック」](#)を参照してください。IAM

AWS OpsWorks CM の仕組み IAM

IAM を使用して AWS OpsWorks CM へのアクセスを管理する前に、AWS OpsWorks CM で使用できるIAM機能を理解しておく必要があります。AWS OpsWorks CM やその他の AWS サービスがどのように連携するかの概要についてはIAM、IAM ユーザーガイドの[AWS 「と連携するのサービスIAM」](#)を参照してください。

トピック

- [AWS OpsWorks CM アイデンティティベースのポリシー](#)
- [AWS OpsWorks CM およびリソースベースのポリシー](#)
- [AWS OpsWorks CM タグに基づく認可](#)
- [AWS OpsWorks CM IAMロール](#)

AWS OpsWorks CM アイデンティティベースのポリシー

IAM ID ベースのポリシーでは、許可または拒否されたアクションとリソース、およびアクションが許可または拒否される条件を指定できます。AWS OpsWorks CM は、特定のアクション、リソース、および条件キーをサポートしています。JSON ポリシーで使用するすべての要素については、ユーザーガイドの[IAMJSON 「ポリシー要素リファレンス」](#)を参照してください。IAM

AWS OpsWorks CM では、カスタムポリシーステートメントをユーザー、ロール、またはグループにアタッチできます。

アクション

IAM ID ベースのポリシーの Action 要素は、ポリシーによって許可または拒否される特定のアクションを記述します。ポリシーアクションは通常、関連付けられた AWS APIオペレーションと同じ名前を持ちます。このアクションは、関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

AWS OpsWorks CM のポリシーアクションでは、アクションの前に次のプレフィックスを使用します。opsworks-cm:例えば、APIオペレーションを使用して AWS OpsWorks CM サーバーを作成す

るアクセス許可をユーザーに付与するには、`opsworks-cm:CreateServer` アクションをポリシーに含めます。ポリシーステートメントには、`Action` または `NotAction` 要素を含める必要があります。AWS OpsWorks CM は、このサービスで実行できるタスクを説明する独自の一連のアクションを定義します。

単一ステートメントに複数アクションを指定するには、次のようにカンマで区切ります:

```
"Action": [
  "opsworks-cm:action1",
  "opsworks-cm:action2"
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、`Describe` という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "opsworks-cm:Describe*"
```

ワイルドカードを使用して複数のアクションをポリシーステートメントで許可する場合は、これらのアクションを許可する先が承認済みのサービスまたは承認済みのユーザーであることを確認してください。

AWS OpsWorks CM アクションのリストを確認するには、IAM ユーザーガイドの [「のアクション、リソース、および条件キーAWS OpsWorks」](#) を参照してください。

リソース

`Resource` エレメントは、アクションが適用されるオブジェクトを指定します。ステートメントには、`Resource` または `NotResource` エレメントを含める必要があります。ARN またはワイルドカード (*) を使用してリソースを指定し、ステートメントがすべてのリソースに適用されることを示します。

AWS OpsWorks CM サーバーまたはバックアップの Amazon リソース番号 (ARN) を取得するには、[DescribeServers](#) または [DescribeBackups](#) API オペレーションを実行し、それらのリソースのリソースレベルのポリシーをベースとします。

AWS OpsWorks CM サーバーリソースには、ARN 次の形式の `arn:aws:opsworks-cm:Region:Account:server:ServerName:UniqueId` があります。

```
arn:aws:opsworks-cm:Region:Account:server:ServerName:UniqueId
```

AWS OpsWorks CM バックアップリソースには、ARN 次の形式の `arn:aws:opsworks-cm:Region:Account:backup:ServerName:UniqueId` があります。

```
arn:aws:opsworks-cm:{Region}:${Account}:backup/${ServerName}-{Date-and-Time-Stamp-of-Backup}
```

の形式の詳細についてはARNs、[「Amazon リソースネーム \(ARNs\)」](#) および AWS [「サービス名前空間」](#) を参照してください。

例えば、ステートメントで test-chef-automate Chef Automate サーバーを指定するには、次のを使用しますARN。

```
"Resource": "arn:aws:opsworks-cm:us-west-2:123456789012:server/test-chef-automate/EXAMPLE-d1a2bEXAMPLE"
```

特定のアカウントに属するすべての AWS OpsWorks CM サーバーを指定するには、ワイルドカード (*) を使用します。

```
"Resource": "arn:aws:opsworks-cm:us-west-2:123456789012:server/*"
```

次の例では、リソースとして AWS OpsWorks CM サーバーバックアップを指定します。

```
"Resource": "arn:aws:opsworks-cm:us-west-2:123456789012:backup/test-chef-automate-server-2018-05-20T19:06:12.399Z"
```

リソースを作成するためのアクションなど、一部の AWS OpsWorks CM アクションは、特定のリソースに対して実行できません。このような場合は、ワイルドカード * を使用する必要があります。

```
"Resource": "*" 
```

多くのAPIアクションには複数のリソースが含まれます。1つのステートメントで複数のリソースを指定するには、 をカンマARNsで区切ります。

```
"Resource": [
  "resource1",
  "resource2"
]
```

AWS OpsWorks CM リソースタイプとその のリストを確認するにはARNs、IAM ユーザーガイドの [「CM AWS OpsWorks のアクション、リソース、および条件キー」](#) を参照してください。各リソースARNの を指定できるアクションについては、IAM ユーザーガイドの [「CM AWS OpsWorks のアクション、リソース、および条件キー」](#) を参照してください。

条件キー

AWS OpsWorks CM には、ポリシーステートメントの Condition 要素で利用できるサービス固有のコンテキストキーはありません。すべての サービスで利用できるグローバルコンテキストキーのリストについては、IAM ポリシーリファレンスの [AWS 「グローバル条件コンテキストキー」](#) を参照してください。すべての AWS グローバル条件キーを表示するには、ユーザーガイドの [AWS 「グローバル条件コンテキストキー」](#) を参照してください。IAM

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition エlement はオプションです。イコールや以下などの [条件演算子](#) を使用する条件表現を構築して、リクエスト内に値のあるポリシーの条件に一致させることができます。

1 つのステートメントに複数の Condition エlement を指定する場合、または 1 つの Condition エlement に複数のキーを指定する場合、AWS が論理 AND 演算を使用してそれら进行评估します。1 つの条件キーに複数の値を指定すると、は論理 OR オペレーションを使用して条件 AWS を评估します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。たとえば、ユーザー名でタグ付けされている場合のみ、リソースにアクセスする ユーザーアクセス許可を付与できます。詳細については、IAM 「ユーザーガイド」の [IAM 「ポリシー要素: 変数とタグ」](#) を参照してください。

例

AWS OpsWorks CM ID ベースのポリシーの例を表示するには、「」を参照してください [AWS OpsWorks CM アイデンティティベースのポリシーの例](#)。

AWS OpsWorks CM およびリソースベースのポリシー

AWS OpsWorks CM はリソースベースのポリシーをサポートしていません。

リソースベースのポリシーは、指定されたプリンシパルがリソースに対して実行できるアクションと条件を指定する JSON ポリシードキュメントです。

AWS OpsWorks CM タグに基づく認可

タグを AWS OpsWorks CM リソースにアタッチすることも、リクエストのタグを AWS OpsWorks CM に渡すこともできます。タグに基づいてアクセスを制御するには、aws:RequestTag/*key-name*、または aws:TagKeys の条件キーを使用して、ポリシーの [条件要素](#) でタグ情報を指定します。AWS OpsWorks CM リソースのタグ付けの詳細については、このガイド [AWS OpsWorks for Puppet Enterprise リソースでのタグの使用](#) の [AWS OpsWorks for Chef Automate リソースでのタグの使用](#) または を参照してください。

AWS OpsWorks CM IAMロール

[IAM ロール](#)は、特定のアクセス許可を持つ AWS アカウント内のエンティティです。

AWS OpsWorks CM は 2 つのロールを使用します。

- ユーザーの AWS アカウント内で動作するアクセス許可を AWS OpsWorks CM サービスに付与するサービスロール。OpsWorks CM が提供するデフォルトのサービスロールを使用する場合、このロールの名前は `aws-opsworks-cm-service-role` です。
- AWS OpsWorks CM サービスが OpsWorks CM を呼び出すことを許可するインスタンスプロファイルロールAPI。このロールはAmazon S3およびへのアクセスを許可 AWS CloudFormation し、バックアップ用のサーバーと S3 バケットを作成します。OpsWorks CM が提供するデフォルトのインスタンスプロファイルを使用する場合、このインスタンスプロファイルロールの名前は `aws-opsworks-cm-ec2-role` です。

AWS OpsWorks CM はサービスにリンクされたロールを使用しません。

AWS OpsWorks CM による一時的な認証情報の使用

AWS OpsWorks CM は一時的な認証情報の使用をサポートし、その機能を から継承します AWS Security Token Service。

一時的な認証情報を使用して、フェデレーションでサインインしたり、IAMロールを引き受けたり、クロスアカウントロールを引き受けたりすることができます。[AssumeRole](#) や などのオペレーションを呼び出す AWS STS API ことで、一時的なセキュリティ認証情報を取得します [GetFederationToken](#)。

サービスリンクロール

AWS OpsWorks CM はサービスにリンクされたロールを使用しません。

[サービスにリンクされたロール](#)を使用すると、AWS サービスが他のサービスのリソースにアクセスして、ユーザーに代わってアクションを実行できます。サービスにリンクされたロールはIAMアカウントに表示され、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

サービスロール

この機能により、ユーザーに代わってサービスが [サービスロール](#) を引き受けることが許可されます。このロールにより、サービスがお客様に代わって他のサービスのリソースにアクセスし、アクション

を完了することが許可されます。サービスロールはIAMアカウントに表示され、アカウントによって所有されます。つまり、IAM管理者はこのロールのアクセス許可を変更できます。ただし、それにより、サービスの機能が損なわれる場合があります。

AWS OpsWorks CM は 2 つのロールを使用します。

- ユーザーの AWS アカウント内で動作するアクセス許可を AWS OpsWorks CM サービスに付与するサービスロール。OpsWorks CM が提供するデフォルトのサービスロールを使用する場合、このロールの名前は `aws-opsworks-cm-service-role` です。
- AWS OpsWorks CM サービスが OpsWorks CM を呼び出すことを許可するインスタンスプロファイルロールAPI。このロールはAmazon S3およびへのアクセスを許可 AWS CloudFormation し、バックアップ用のサーバーと S3 バケットを作成します。OpsWorks CM が提供するデフォルトのインスタンスプロファイルを使用する場合、このインスタンスプロファイルロールの名前は `aws-opsworks-cm-ec2-role` です。

AWS OpsWorks CM での IAM ロールの選択

AWS OpsWorks CM でサーバーを作成するときは、AWS OpsWorks CM がEC2ユーザーに代わって Amazon にアクセスすることを許可するロールを選択する必要があります。サービスロールを既に作成している場合 AWS OpsWorks、CM は選択するロールのリストを提供します。OpsWorks CM は、ロールを指定しない場合に作成できます。Amazon EC2インスタンスを開始および停止するためのアクセスを許可するロールを選択することが重要です。詳細については、[Chef Automate サーバーの作成](#) または [Puppet Enterprise マスターの作成](#) を参照してください。

AWS OpsWorks CM アイデンティティベースのポリシーの例

デフォルトでは、ユーザーまたはロールには AWS OpsWorks CM リソースを作成または変更するアクセス許可がありません。また、AWS Management Console、AWS CLI、またはを使用してタスクを実行することはできません AWS API。IAM 管理者は、ID IAM に必要な特定のリソースに対して特定のAPIオペレーションを実行するアクセス許可を付与するIAMポリシーを作成する必要があります。続いて、管理者はそれらのアクセス許可が必要なユーザーまたはグループにそのポリシーをアタッチします。

これらのポリシードキュメント例を使用して IAM ID ベースのJSONポリシーを作成する方法については、IAM「ユーザーガイド」の[IAM「ポリシーの作成」](#)を参照してください。

AWS OpsWorks CM では、`AWSOpsWorksCMServiceRole`ポリシーをユーザーに割り当てて、AWS Management Console またはを使用して Chef Automate または Puppet Enterprise サーバーを作成および管理できます AWS CLI。

トピック

- [ポリシーのベストプラクティス](#)
- [ユーザーが自分のアクセス許可を表示できるようにする](#)
- [タグに基づく AWS OpsWorks CM サーバーの表示](#)

ポリシーのベストプラクティス

ID ベースのポリシーは、アカウント内の OpsWorks CM リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションを実行すると、AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS マネージドポリシーを開始し、最小権限のアクセス許可に移行 – ユーザーとワークロードにアクセス許可を付与するには、多くの一般的なユースケースにアクセス許可を付与するAWS マネージドポリシーを使用します。これらはで利用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM 「ユーザーガイド」の「[AWS 管理ポリシーAWS](#)」または [ジョブ機能の 管理ポリシー](#)を参照してください。
- 最小権限のアクセス許可を適用する - IAMポリシーでアクセス許可を設定する場合、タスクの実行に必要なアクセス許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用してアクセス許可を適用する方法の詳細については、ユーザーガイドの「」の「[ポリシーとアクセス許可IAM](#)」を参照してください。 IAM
- IAM ポリシーの条件を使用してアクセスをさらに制限する – ポリシーに条件を追加して、アクションとリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを を使用して送信する必要があることを指定できますSSL。また、 などの特定の を通じてサービスアクションが使用されている場合 AWS のサービス、条件を使用してサービスアクションへのアクセスを許可することもできます AWS CloudFormation。詳細については、「ユーザーガイド」の [IAMJSON 「ポリシー要素: 条件」](#) を参照してください。 IAM
- IAM Access Analyzer を使用してIAMポリシーを検証し、安全で機能的なアクセス許可を確保します – IAM Access Analyzer は、ポリシーがポリシー言語 (JSON) とIAMベストプラクティスに準拠するように、新規および既存のIAMポリシーを検証します。IAM Access Analyzer には、安全で機能的なポリシーの作成に役立つ 100 を超えるポリシーチェックと実用的な推奨事項が用意されています。詳細については、IAM 「ユーザーガイド」の [IAM 「Access Analyzer を使用したポリシーの検証」](#) を参照してください。

- 多要素認証が必要 (MFA) – でIAMユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、 をオンにMFAしてセキュリティを強化します。API オペレーションが呼び出されるMFAタイミングを要求するには、ポリシーにMFA条件を追加します。詳細については、「ユーザーガイド」の [「によるセキュアAPIアクセスMFA」](#) を参照してください。IAM

のベストプラクティスの詳細についてはIAM、「ユーザーガイド」の [「セキュリティのベストプラクティスIAM」](#) を参照してください。IAM

ユーザーが自分のアクセス許可を表示できるようにする

この例では、ユーザー ID にアタッチされたインラインおよび管理ポリシーの表示をユーザーに許可するポリシーを作成する方法を示します。このポリシーには、コンソールで、または AWS CLI または を使用してプログラムでこのアクションを実行するアクセス許可が含まれています AWS API。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": [
        "arn:aws:iam::*:user/${aws:username}"
      ]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",

```

```

        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}

```

タグに基づく AWS OpsWorks CM サーバーの表示

ID ベースのポリシーの条件を使用して、タグに基づいて AWS OpsWorks CM サーバーとバックアップへのアクセスを制御できます。この例では、AWS OpsWorks CM サーバーの表示を許可するポリシーを作成する方法を示しています。ただし、アクセス許可は、AWS OpsWorks CM サーバータグがそのユーザーのユーザー名の値 `Owner` を持つ場合にのみ付与されます。このポリシーでは、このアクションをコンソールで実行するために必要なアクセス許可も付与します。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListServersInConsole",
      "Effect": "Allow",
      "Action": "opsworks-cm:DescribeServers",
      "Resource": "*"
    },
    {
      "Sid": "ViewServerIfOwner",
      "Effect": "Allow",
      "Action": "opsworks-cm:DescribeServers",
      "Resource": "arn:aws:opsworks-cm:region:master-account-ID:server/server-name",
      "Condition": {
        "StringEquals": {"opsworks-cm:ResourceTag/Owner": "${aws:username}"}
      }
    }
  ]
}

```

このポリシーをアカウントのユーザーにアタッチできます。という名前のユーザーが AWS OpsWorks CM サーバーを表示しようとする場合は、サーバーに `Owner=richard-roe` または `owner=richard-roe` のタグを付ける必要があります。それ以外の場合、アクセスは拒否されます。条件キー名では大文字と小文字が区別されないため、条件タグキー `Owner` は `owner`

と owner の両方に一致します。詳細については、IAM「ユーザーガイド」の [IAMJSON「ポリシー要素: 条件」](#) を参照してください。

AWS OpsWorks CM Identity and Access のトラブルシューティング

次の情報を使用して、の使用時に発生する可能性のある一般的な問題を診断して修正しますIAM。AWS OpsWorks CM 固有のトラブルシューティング情報については、[トラブルシューティング AWS OpsWorks for Chef Automate](#)「」および「」を参照してください[Puppet Enterprise OpsWorks のトラブルシューティング](#)。

トピック

- [AWS OpsWorks CM でアクションを実行する権限がありません](#)
- [iam を実行する権限がありません。PassRole](#)
- [AWS アカウント外のユーザーに AWS OpsWorks CM リソースへのアクセスを許可したい](#)

AWS OpsWorks CM でアクションを実行する権限がありません

がアクションを実行する権限がないこと AWS Management Console を指示する場合は、管理者に連絡してサポートを求めてください。管理者とは、サインイン認証情報を提供した担当者です。

次のエラー例は、ユーザーが コンソールを使用して AWS OpsWorks CM mateojackson サーバーの詳細を表示しようとしたときに、opsworks-cm:DescribeServersアクセス許可がない場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
opsworks-cm:DescribeServers on resource: test-chef-automate-server
```

この場合、Mateo は、test-chef-automate-server アクションを使用して opsworks-cm:DescribeServers リソースにアクセスすることを許可するように、ポリシーの更新を管理者に依頼します。

iam を実行する権限がありません。PassRole

iam:PassRole アクションを実行することが認可されていないというエラーが表示された場合、管理者に問い合わせ、サポートを依頼する必要があります。管理者とは、サインイン認証情報を提供した担当者です。OpsWorks CM にロールを渡すことができるようにポリシーを更新するようにその人に依頼します。

一部の AWS サービスでは、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、既存のロールをそのサービスに渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

次のエラー例は、 という名前のユーザーがコンソールを使用して OpsWorks CM marymajor でアクションを実行しようとするると発生します。ただし、アクションには、サービスロールによってサービスに許可が付与されている必要があります。Mary には、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

この場合、Mary は iam:PassRole アクションの実行が許可されるように、担当の管理者にポリシーの更新を依頼します。

AWS アカウント外のユーザーに AWS OpsWorks CM リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACLs) をサポートするサービスでは、これらのポリシーを使用して、リソースへのアクセスをユーザーに許可できます。

詳細については、以下を参照してください。

- AWS OpsWorks CM は、AWS OpsWorks CM サーバーを管理するためのアクセスを複数のアカウントからユーザーに付与することをサポートしています。
- 所有している AWS アカウント間でリソースへのアクセスを提供する方法については、IAM ユーザーガイドの [「所有している別の AWS アカウントの IAM ユーザーへのアクセスを提供する」](#) を参照してください。
- サードパーティー AWS アカウントにリソースへのアクセスを提供する方法については、IAM ユーザーガイドの [「サードパーティーが所有する AWS アカウントへのアクセスを提供する」](#) を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイドの [「外部認証されたユーザーへのアクセスを提供する \(ID フェデレーション\)」](#) を参照してください。
- クロスアカウントアクセスにロールとリソースベースのポリシーを使用する方法の違いについては、IAM 「ユーザーガイド」の [IAM 「ロールとリソースベースのポリシーの違い」](#) を参照してください。

AWS の マネージドポリシー AWS OpsWorks 設定管理

ユーザー、グループ、ロールにアクセス許可を追加するには、を使用する方が簡単です。AWS 自分でポリシーを記述するよりも マネージドポリシー。必要なアクセス許可のみをチームに提供する [IAMカスタマー管理ポリシーを作成するには](#)、時間と専門知識が必要です。すぐに使用を開始するには、AWS マネージドポリシー。これらのポリシーは一般的なユースケースを対象としており、で利用できます。AWS アカウント。の詳細については、「」を参照してください。AWS 管理ポリシー、「」を参照してください。 [AWSIAM ユーザーガイドの マネージドポリシー](#)。

AWS サービスによるメンテナンスと更新 AWS マネージドポリシー。のアクセス許可は変更できません AWS マネージドポリシー。サービスが にアクセス許可を追加することがある AWS 新機能をサポートする マネージドポリシー。この種の更新は、ポリシーがアタッチされているすべてのアイデンティティ (ユーザー、グループ、ロール) に影響を与えます。サービスは を更新する可能性が最も高い AWS 新しい機能が起動されたとき、または新しいオペレーションが利用可能になったときの マネージドポリシー。サービスは からアクセス許可を削除しません AWS ポリシーの更新によって既存のアクセス許可が損なわれないように、管理ポリシー。

さらに、AWS は、複数の サービスにまたがる職務機能の マネージドポリシーをサポートします。例えば、ReadOnlyAccess AWS 管理ポリシーは、すべての への読み取り専用アクセスを提供します。AWS サービスとリソース。サービスが新機能を起動すると、AWS は、新しいオペレーションとリソースの読み取り専用アクセス許可を追加します。職務機能ポリシーのリストと説明については、「」を参照してください。 [AWS ユーザーガイドの ジョブ機能の IAM マネージドポリシー](#)。

AWS 管理ポリシー: `AWSOpsWorksCMServiceRole`

IAM をエンティティ `AWSOpsWorksCMServiceRole` にアタッチできます。OpsWorks CM は、OpsWorks CM がユーザーに代わってアクションを実行できるようにするサービスロールにこのポリシーをアタッチします。

このポリシーでは、*administrative* OpsWorks CM 管理者が OpsWorks CM サーバーとバックアップを作成、管理、削除できるようにする アクセス許可。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `opsworks-cm` — プリンシパルが既存のサーバを削除して、メンテナンス実行を開始できるようにします。
- `acm` — プリンシパルが から証明書を削除またはインポートすることを許可する AWS Certificate Manager ユーザーが OpsWorks CM サーバーに接続できるようにする。

- `cloudformation` – OpsWorks CM が作成および管理できるようにします AWS CloudFormation プリンシパルが OpsWorks CM サーバーを作成、更新、または削除するときのスタック。
- `ec2` – プリンシパルが OpsWorks CM サーバーを作成、更新、または削除するときに、OpsWorks CM が Amazon Elastic Compute Cloud インスタンスを起動、プロビジョニング、更新、および終了できるようにします。
- `iam` – OpsWorks CM が CM OpsWorks サーバーの作成と管理に必要なサービスロールを作成できるようにします。
- `tag` – プリンシパルがサーバーやバックアップなどの OpsWorks CM リソースからタグを適用および削除できるようにします。
- `s3` – OpsWorks CM がサーバーバックアップを保存するための Amazon S3 バケットの作成、プリンシパルリクエスト時の S3 バケット内のオブジェクトの管理 (バックアップの削除など)、およびバケットの削除を許可。
- `secretsmanager` – OpsWorks CM が Secrets Manager シークレットを作成および管理し、シークレットにタグを適用または削除できるようにします。
- `ssm` – OpsWorks CM サーバーであるインスタンスで Systems Manager Run Command の使用を OpsWorks CM に許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket*"
      ],
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteObject",
        "s3:DeleteBucket",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutBucketPolicy",
        "s3:PutObject",
        "s3:GetBucketTagging",
        "s3:PutBucketTagging"
      ]
    }
  ],
}
```

```

    {
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Action": [
        "tag:UntagResources",
        "tag:TagResources"
      ]
    },
    {
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Action": [
        "ssm:DescribeInstanceInformation",
        "ssm:GetCommandInvocation",
        "ssm:ListCommandInvocations",
        "ssm:ListCommands"
      ]
    },
    {
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "ssm:resourceTag/aws:cloudformation:stack-name": "aws-opsworks-cm-
**
        }
      },
      "Action": [
        "ssm:SendCommand"
      ]
    },
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:ssm:*::document/*",
        "arn:aws:s3:::amzn-s3-demo-bucket*"
      ],
      "Action": [

```



```

        "ssm:SendCommand"
    ]
},
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "ec2:AllocateAddress",
        "ec2:AssociateAddress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateImage",
        "ec2:CreateSecurityGroup",
        "ec2:CreateSnapshot",
        "ec2:CreateTags",
        "ec2>DeleteSecurityGroup",
        "ec2>DeleteSnapshot",
        "ec2:DeregisterImage",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAddresses",
        "ec2:DescribeImages",
        "ec2:DescribeInstanceStatus",
        "ec2:DescribeInstances",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSnapshots",
        "ec2:DescribeSubnets",
        "ec2:DisassociateAddress",
        "ec2:ReleaseAddress",
        "ec2:RunInstances",
        "ec2:StopInstances"
    ]
},
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Condition": {
        "StringLike": {
            "ec2:ResourceTag/aws:cloudformation:stack-name": "aws-opsworks-cm-
*"
        }
    }
},

```

```
    "Action": [
      "ec2:TerminateInstances",
      "ec2:RebootInstances"
    ],
  },
  {
    "Effect": "Allow",
    "Resource": [
      "arn:aws:opsworks-cm:*:*:server/*"
    ],
    "Action": [
      "opsworks-cm:DeleteServer",
      "opsworks-cm:StartMaintenance"
    ]
  },
  {
    "Effect": "Allow",
    "Resource": [
      "arn:aws:cloudformation:*:*:stack/aws-opsworks-cm-*"
    ],
    "Action": [
      "cloudformation:CreateStack",
      "cloudformation>DeleteStack",
      "cloudformation:DescribeStackEvents",
      "cloudformation:DescribeStackResources",
      "cloudformation:DescribeStacks",
      "cloudformation:UpdateStack"
    ]
  },
  {
    "Effect": "Allow",
    "Resource": [
      "arn:aws:iam:*:*:role/aws-opsworks-cm-*",
      "arn:aws:iam:*:*:role/service-role/aws-opsworks-cm-*"
    ],
    "Action": [
      "iam:PassRole"
    ]
  },
  {
    "Effect": "Allow",
    "Resource": "*",
    "Action": [
      "acm:DeleteCertificate",
```

```
        "acm:ImportCertificate"
    ]
},
{
    "Effect": "Allow",
    "Resource": "arn:aws:secretsmanager:*:*:opsworks-cm!aws-opsworks-cm-
secrets-*",
    "Action": [
        "secretsmanager:CreateSecret",
        "secretsmanager:GetSecretValue",
        "secretsmanager:UpdateSecret",
        "secretsmanager>DeleteSecret",
        "secretsmanager:TagResource",
        "secretsmanager:UntagResource"
    ]
},
{
    "Effect": "Allow",
    "Action": "ec2:DeleteTags",
    "Resource": [
        "arn:aws:ec2:*:*:instance/*",
        "arn:aws:ec2:*:*:elastic-ip/*",
        "arn:aws:ec2:*:*:security-group/*"
    ]
}
]
```

AWS 管理ポリシー: `AWSOpsWorksCMInstanceProfileRole`

IAM をエンティティ `AWSOpsWorksCMInstanceProfileRole` にアタッチできます。OpsWorks CM は、OpsWorks CM がユーザーに代わってアクションを実行できるようにするサービスロールにこのポリシーをアタッチします。

このポリシーでは、*administrative* OpsWorks CM サーバーとして使用される Amazon EC2 インスタンスが から情報を取得できるようにする アクセス許可 AWS CloudFormation また、AWS Secrets Manager、および は、サーバーバックアップを Amazon S3 バケットに保存します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `acm` – OpsWorks CM サーバーEC2インスタンスが から証明書を取得できるようにします AWS Certificate Manager ユーザーが OpsWorks CM サーバーに接続できるようにする。
- `cloudformation` — OpsWorks CM サーバーEC2インスタンスが に関する情報を取得できるようにします。 AWS CloudFormation インスタンスの作成または更新プロセス中の スタック、および へのシグナルの送信 AWS CloudFormation ステータスについて。
- `s3` — OpsWorks CM サーバーEC2インスタンスがサーバーバックアップをアップロードして S3 バケットに保存し、必要に応じてアップロードを停止またはロールバックし、S3 バケットからバックアップを削除できるようにします。
- `secretsmanager` – OpsWorks CM サーバーEC2インスタンスが OpsWorks CM 関連の Secrets Manager シークレットの値を取得できるようにします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudformation:DescribeStackResource",
        "cloudformation:SignalResource"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    },
    {
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:DeleteObject",
        "s3:GetObject",
        "s3:ListAllMyBuckets",
        "s3:ListBucket",
        "s3:ListMultipartUploadParts",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket*",
      "Effect": "Allow"
    },
    {
      "Action": "acm:GetCertificate",
      "Resource": "*",
    }
  ]
}
```

```

        "Effect": "Allow"
    },
    {
        "Action": "secretsmanager:GetSecretValue",
        "Resource": "arn:aws:secretsmanager:*:*:opsworks-cm!aws-opsworks-cm-
secrets-*",
        "Effect": "Allow"
    }
]
}

```

OpsWorks への CM 更新 AWS 管理ポリシー

の更新に関する詳細を表示する AWS このサービスがこれらの変更の追跡を開始してからの OpsWorks CM の マネージドポリシー。このページの変更に関する自動アラートについては、[OpsWorks CM ドキュメント履歴](#)ページのRSSフィードにサブスクライブしてください。

変更	説明	日付
AWSOpsWorksCMInstanceProfileRole - 管理ポリシーの更新	OpsWorks CM は、OpsWorks CM サーバーとして使用されるEC2インスタンスが および Secrets Manager と CloudFormation情報を共有し、バックアップを管理できるようにする マネージドポリシーを更新しました。この変更opsworks-cm! により、Secrets Manager シークレットのリソース名に追加され、OpsWorks CM がシークレットを所有できるようになります。	2021 年 4 月 23 日
AWSOpsWorksCMServiceRole - 管理ポリシーの更新	OpsWorks CM は、CM 管理者が OpsWorks CM OpsWorks サーバーとバックアップを作成、管理、削除できるようにする マネージドポリ	2021 年 4 月 23 日

変更	説明	日付
	シーを更新しました。この変更 <code>opsworks-cm!</code> により、Secrets Manager シークレットのリソース名に追加され、OpsWorks CM がシークレットを所有できるようになります。	
OpsWorks CM が変更の追跡を開始しました	OpsWorks CM が の変更の追跡を開始しました AWS マネージドポリシー。	2021 年 4 月 23 日

AWS OpsWorks CMにおけるサービス間の混乱した代理の防止

混乱した代理問題は、アクションを実行する許可を持たないエンティティが、より特権のあるエンティティにアクションを実行するように強制できるセキュリティの問題です。AWS では、サービス間でのなりすましが、混乱した代理問題を生じさせることがあります。サービス間でのなりすまは、1つのサービス(呼び出し元サービス)が、別のサービス(呼び出し対象サービス)を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別の顧客のリソースに対する処理を実行するように操作される場合があります。これを防ぐため、AWS では、アカウント内のリソースへのアクセス権が付与されたサービスプリンシパルですべてのサービスのデータを保護するために役立つツールを提供しています。

AWS OpsWorks CM がリソースに別のサービスを提供する権限を制限するために、リソースポリシーで [aws:SourceArn](#) および [aws:SourceAccount](#) グローバル条件コンテキストキーを使用することをお勧めします。aws:SourceArn 値にアカウント ID (Amazon S3 バケット ARN など) が含まれていない場合は、両方のグローバル条件コンテキストキーを使用してアクセス許可を制限する必要があります。同じポリシーステートメントでこれらのグローバル条件コンテキストキーの両方を使用し、アカウント ID にaws:SourceArn の値が含まれていない場合、aws:SourceAccount 値と aws:SourceArn 値の中のアカウントには、同じアカウント ID を使用する必要があります。クロスサービスのアクセスにリソースを1つだけ関連付けたい場合は、aws:SourceArn を使用します。クロスサービスが使用できるように、アカウント内の任意のリソースを関連づけたい場合は、aws:SourceAccount を使用します。

`aws:SourceArn` の値は、OpsWorks CM シェフサーバーまたはパペットサーバーの ARN である必要があります。

混乱した代理問題から保護するための最も効果的な方法は、AWS OpsWorks CMサーバーの完全な ARN を指定しながら、`aws:SourceArn` グローバル条件コンテキストキーを使用することです。完全な ARN が不明な場合や、複数のサーバーARNを指定する場合には、ARN の不明な部分にワイルドカード (*) を含む `aws:SourceArn` グローバルコンテキスト条件キーを使用します。例えば、`arn:aws:service:*:123456789012:*` です。

次のセクションでは、AWS OpsWorks CM で `aws:SourceArn` および `aws:SourceAccount` グローバル条件コンテキストキーを使用して、混乱した代理問題を回避する方法を示します。

AWS OpsWorks CM 混乱した代理攻撃の防止

このセクションでは、AWS OpsWorks CM での混乱を招く副攻撃を防ぐ方法について説明し、AWS OpsWorks CM アクセスに使用している IAM ロールにアタッチできるアクセス権限ポリシーの例を紹介します。セキュリティのベストプラクティスとして、`aws:SourceArn` と `aws:SourceAccount` の条件キーを IAM ロールとほかのサービスとの信頼関係に追加することをお勧めします。信頼関係により、AWS OpsWorks CM は AWS OpsWorks CM サーバーの作成や管理に必要なアクションを他のサービスで実行する役割を担うことができます。

信頼関係を編集するために、`aws:SourceArn` と `aws:SourceAccount` 条件キーを追加するには

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. 左のナビゲーションペインで、[Roles] (ロール) を選択します。
3. 検索ボックスで、AWS OpsWorks CM へのアクセスに使用するロールを検索します。AWS 管理対象ロールは `aws-opsworks-cm-service-role` です。
4. そのロールの 概要 ページで、信頼関係 タブを選択します。
5. [信頼関係] タブで、[信頼関係の編集] を選択します。
6. ポリシードキュメントで、`aws:SourceArn` または `aws:SourceAccount` 条件キーのうち少なくとも一つをポリシーに追加します。サービス間 (AWS Certificate Manager や Amazon EC2 など) と AWS OpsWorks CM の間の信頼関係を、より制限の厳しい特定の AWS OpsWorks CM サーバーに制限するために `aws:SourceArn` を使用します。さらに、`aws:SourceAccount` を追加すると、サービス間と AWS OpsWorks CM の間の信頼関係を、制限が緩和された特定のアカウントのサーバーに制限することもできます。次に例を示します。両方の条件キーを使用する場合、アカウント ID は同じでなければならないことに注意してください。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "Service": "opsworks-cm.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:opsworks-cm:us-east-2:123456789012:server/my-opsworks-server/EXAMPLEabcd-1234-efghEXAMPLE-ID"
      }
    }
  }
]
```

7. 条件キーを追加し終わったら、[信頼ポリシーの更新] を選択します。

`aws:SourceArn` と `aws:SourceAccount` を使用して AWS OpsWorks CM サーバーへのアクセスを制限するロールのその他の例を以下に示します。

トピック

- [例:特定の地域の AWS OpsWorks CM サーバーへのアクセス](#)
- [例: aws:SourceArn に複数のサーバー ARN の追加](#)

例:特定の地域の AWS OpsWorks CM サーバーへのアクセス

次のロール信頼関係ステートメントは米国東部 (オハイオ) リージョン (us-east-2) 内の AWS OpsWorks CM サーバーにアクセスします。リージョンは `aws:SourceArn` の ARN 値で指定されていますが、サーバー ID の値はワイルドカード (*) であることに注意してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```



```
"Principal": {
  "Service": "opsworks-cm.amazonaws.com"
},
"Action": "sts:AssumeRole",
"Condition": {
  "StringEquals": {
    "aws:SourceAccount": "123456789012"
  },
  "ArnEquals": {
    "aws:SourceArn": "arn:aws:opsworks-cm:us-east-2:123456789012:server/*"
  }
}
}
```

例: **aws:SourceArn** に複数のサーバー ARN の追加

次の例では、アカウント ID 123456789012 の二つの AWS OpsWorks CM サーバーからなる配列へのアクセスを制限しています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks-cm.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:opsworks-cm:us-east-2:123456789012:server/my-chef-
server/unique_ID",
            "arn:aws:opsworks-cm:us-east-2:123456789012:server/my-puppet-
server/unique_ID"
          ]
        }
      }
    }
  ]
}
```

```
}  
]  
}
```

インターネットトラフィックのプライバシー

AWS OpsWorks CM は、一般的に使用しているのと同じ送信セキュリティプロトコルを使用します。AWS: HTTPS、または TLS 暗号化 HTTP あり。

でのログ記録とモニタリング AWS OpsWorks CM

AWS OpsWorks CM はすべての API アクションを に記録します CloudTrail。詳細については、以下の各トピックを参照してください。

- [OpsWorks を使用した Puppet Enterprise API コールのログ記録 AWS CloudTrail](#)
- [を使用した AWS OpsWorks for Chef Automate API コールのログ記録 AWS CloudTrail](#)

のコンプライアンス検証 AWS OpsWorks CM

AWS OpsWorks CM は、以下のコンプライアンスプログラムと規制をサポートしています。

- Payment Card Industry (PCI)
- 1996 年健康保険の相互運用性と説明責任に関する法律 (HIPAA)
- AWS システムおよび組織のコントロール (SOC) 1、2、3
- 一般データ保護規則 (GDPR)

サードパーティーの監査者は、 のセキュリティとコンプライアンスを評価します。AWS OpsWorks CM を複数の の一部として AWS コンプライアンスプログラム。これには、SOC、PCI、Fed RAMP、HIPAA などが含まれます。

のリストの場合 AWS 特定のコンプライアンスプログラムの対象となる のサービスについては、「[コンプライアンスAWSプログラムによる対象範囲内のサービス](#)」を参照してください。一般的な情報については、「」を参照してください。 [AWS コンプライアンスプログラム](#)。

サードパーティーの監査レポートは、 を使用してダウンロードできます。AWS Artifact。詳細については、[AWS「アーティファクト」でのレポートのダウンロード](#)」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS OpsWorks CM は、データの機密性、企業のコンプライアンス目的、適用可能な法律および規制によって決まります。AWS では、コンプライアンスに役立つ以下のリソースを提供しています。

- [セキュリティとコンプライアンスのクイックスタートガイド](#) – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境を にデプロイする手順について説明します。AWS.
- [HIPAA セキュリティとコンプライアンスのアーキテクチャに関するホワイトペーパー](#) – このホワイトペーパーでは、企業が を使用する方法について説明します。AWS は、HIPAA 準拠のアプリケーションを作成します。
- [AWS コンプライアンスリソース](#) – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- [AWS Config](#) – これは AWS サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。
- [AWS Security Hub](#) – これは AWS サービスでは、内のセキュリティ状態を包括的に把握できます。AWS は、セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立ちます。

の耐障害性 AWS OpsWorks CM

AWS OpsWorks CM は、サーバーの作成時に、デフォルトでサーバーの日次バックアップを有効にします。バックアップは暗号化され、Amazon S3 バケットに保存されます。デフォルトでは、このバケットにはサーバーの作成元のアカウントからのみアクセスできます。他のユーザーに対してバケットへのアクセスを追加したり、必要に応じて Amazon S3 のクロスリージョンバックアップを設定したりすることができます。Chef と Puppet は、どちらの製品も 間のトラフィックを暗号化するため、クロスリージョン暗号化をサポートしています。AWS OpsWorks CM サーバーとマネージドノード。

AWS OpsWorks CM は高可用性 (HA) 設定をサポートしていません。

- AWS グローバルインフラストラクチャは AWS リージョンとアベイラビリティゾーン。AWS リージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立し隔離されたアベイラビリティゾーンがあります。アベイラビリティゾーンでは、アベイラビリティゾーン間で中断せずに、自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、およびスケーラビリティが優れています。

でサーバーをバックアップおよび復元する方法の詳細については、「」を参照してください。AWS OpsWorks CM、以下を参照してください。

- [OpsWorks for Puppet Enterprise Server のバックアップと復元](#)
- [AWS OpsWorks for Chef Automate サーバーのバックアップと復元](#)

の詳細については、「」を参照してください。AWS リージョンとアベイラビリティゾーンについては、「」を参照してください。 [AWS グローバルインフラストラクチャ](#)。

のインフラストラクチャセキュリティ AWS OpsWorks CM

マネージドサービスとして、AWS OpsWorks 設定管理は によって保護されます。AWS グローバルネットワークセキュリティ。参考情報 AWS セキュリティサービスとその方法 AWS インフラストラクチャを保護するには、「」を参照してください。 [AWS クラウドセキュリティ](#)。を設計するには AWS インフラストラクチャセキュリティのベストプラクティスを使用する 環境については、「セキュリティの柱」の「[インフラストラクチャの保護](#)」を参照してください。AWS Well-Architected フレームワーク。

を使用する AWS は、ネットワーク経由で OpsWorks CM にアクセスするためのAPI呼び出しを発行しました。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。1TLS.2 が必要で、1.3 TLS をお勧めします。
- (Ephemeral Diffie-HellmanPFS) や DHE (Elliptic Curve Ephemeral Diffie-Hellman) などの完全前方秘匿性 ECDHE () を備えた暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

さらに、リクエストは、アクセスキー ID とプリンIAMシパルに関連付けられたシークレットアクセスキーを使用して署名する必要があります。または、 [AWS Security Token Service](#) (AWS STS) リクエストに署名するための一時的なセキュリティ認証情報を生成します。

AWS OpsWorks CM は、プライベートリンクまたはVPCプライベートエンドポイントをサポートしていません。

AWS OpsWorks CM はリソーススペースのポリシーをサポートしていません。詳細については、「[」を参照してくださいAWS IAM](#) でと連携するサービス AWS Identity and Access Management ユーザーガイド。

での設定と脆弱性の分析 AWS OpsWorks CM

AWS OpsWorks CM は、で実行されているオペレーティングシステムに対してカーネルとセキュリティの定期的な更新を実行します。AWS OpsWorks CM サーバー。ユーザーは、現在の日付から最大 2 週間まで、自動更新を実行する時間帯を設定できます。AWS OpsWorks CM は、Chef および Puppet Enterprise のマイナーバージョンの自動更新をプッシュします。の更新の設定の詳細については、「」を参照してください。AWS OpsWorks for Chef Automate、このガイドの「[システムメンテナンス \(Chef\)](#)」を参照してください。Puppet Enterprise OpsWorks のの更新の設定の詳細については、このガイドの「[システムメンテナンス \(Puppet\)](#)」を参照してください。

のセキュリティのベストプラクティス AWS OpsWorks CM

AWS OpsWorks CM、すべてのと同様 AWS サービスでは、独自のセキュリティポリシーを開発および実装する際に考慮すべきセキュリティ機能を提供しています。以下のベストプラクティスは一般的なガイドラインであり、完全なセキュリティソリューションを説明するものではありません。これらのベストプラクティスはおお客様の環境に必ずしも適切または十分でない可能性があるため、処方箋ではなく、あくまで有用な考慮事項とお考えください。

- スターターキットとダウンロードしたログイン認証情報を保護します。新しいを作成する場合 AWS OpsWorks CM サーバーを使用するか、新しいスターターキットと認証情報を からダウンロードします。AWS OpsWorks CM コンソールでは、少なくとも 1 つの認証要素を必要とする安全な場所にこれらの項目を保存します。認証情報は、サーバーへの管理者レベルのアクセスを提供します。
- 設定コードを保護します。ソースリポジトリ用の推奨プロトコルを使用して Chef または Puppet 設定コード (クックブックとモジュール) を保護します。例えば、 のリポジトリへのアクセス許可を制限できます。AWS CodeCommit、または [GitHub ウェブサイトの「リポジトリのセキュリティ保護 GitHubに関するガイドライン」](#)に従ってください。
- CA 署名付き証明書を使用してノードに接続します。ただし、 でノードを登録またはブートストラップするときは、自己署名証明書を使用できます。AWS OpsWorks CM サーバーは、ベストプラクティスとして、CA 署名証明書を使用します。証明機関 (CA) によって署名された証明書を使用することをお勧めします。
- Chef または Puppet 管理コンソールのサインイン認証情報は他のユーザーと共有しません。管理者は、Chef または Puppet コンソールウェブサイトのユーザーごとに別々のユーザーを作成する必要があります。
 - [Chef Automate でのユーザーの管理](#)
 - [Puppet Enterprise でのユーザーの管理](#)

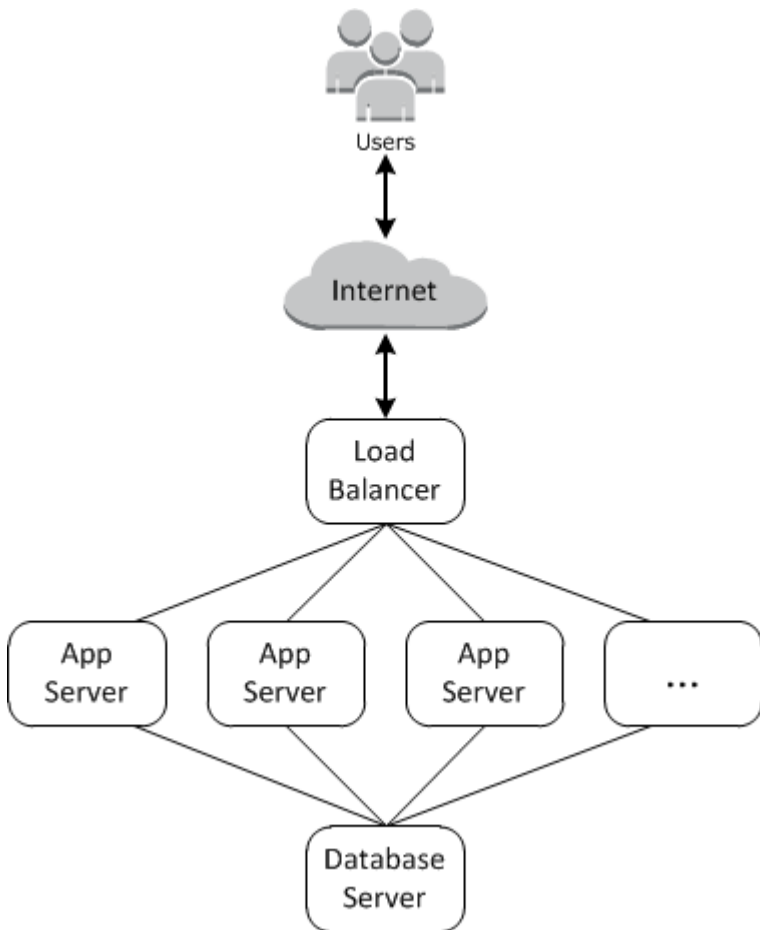
- バックアップとシステムメンテナンスの自動更新を設定します。での自動メンテナンス更新の設定
AWS OpsWorks CM サーバーは、サーバーが最新のセキュリティ関連のオペレーティングシステムの更新を実行していることを保証します。自動バックアップを設定すると、災害対策が容易になり、インシデントや障害発生時の復元時間が短縮されます。を保存する Amazon S3 バケットへのアクセスを制限する AWS OpsWorks CM サーバーのバックアップ。すべての にアクセスを許可しないでください。必要に応じて他のユーザーに個別に読み取りまたは書き込みアクセス権を付与するか、それらのユーザーに IAM でセキュリティグループを作成し、セキュリティグループへのアクセス権を割り当てます。
 - [システムメンテナンス \(Chef\)](#)
 - [システムメンテナンス \(Puppet\)](#)
 - [AWS OpsWorks for Chef Automate サーバーのバックアップと復元](#)
 - [OpsWorks for Puppet Enterprise Server のバックアップと復元](#)
 - [で最初にIAM委任されたユーザーとグループを作成する](#) AWS Identity and Access Management ユーザーガイド
 - [「Amazon Simple Storage Service Developer Guide \(Amazon Simple Storage Service 開発者ガイド\)」の「Security Best Practices for Amazon S3」](#) (Amazon S3 のセキュリティベストプラクティス)]

AWS OpsWorks スタック

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

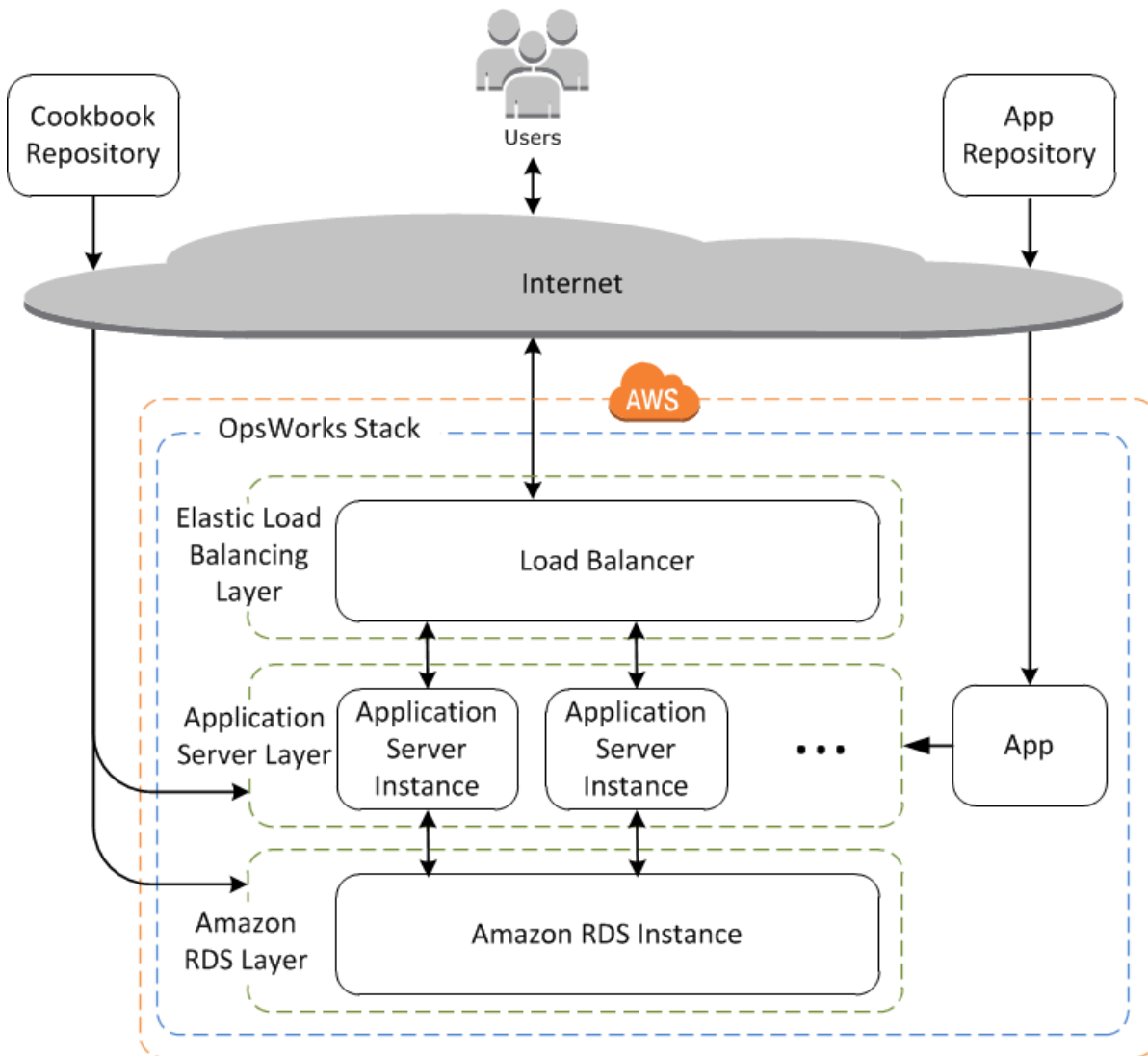
クラウドベースのコンピューティングには、通常、Amazon EC2 インスタンスや Amazon Relational Database Service (RDS) インスタンスなどの AWS リソースのグループが含まれます。これらはまとめて作成および管理する必要があります。たとえば、ウェブアプリケーションには、通常、アプリケーションサーバー、データベースサーバー、ロードバランサーなどが必要です。このインスタンスのグループは、一般にスタックと呼ばれています。例として、シンプルなアプリケーションサーバースタックを図に表すと以下のようになります。



インスタンスを作成したり、必要なパッケージをインストールしたりする他に、通常、アプリケーションをアプリケーションサーバーに分散したり、スタックのパフォーマンスを監視したり、セキュリティおよびアクセス許可を管理したりするなどの操作を行う方法が必要です。

AWS OpsWorks スタックは、スタックとアプリケーションを作成および管理するためのシンプルで柔軟な方法を提供します。

基本的なアプリケーションサーバースタックが AWS OpsWorks スタックでどのように表示されるかは次のとおりです。これは、Elastic Load Balancing ロードバランサーの背後で実行されているアプリケーションサーバーのグループと、バックエンドの Amazon RDS データベースサーバーで構成されます。



比較的シンプルですが、このスタックには AWS OpsWorks スタックの主要な機能がすべて表示されます。構成を以下に示します。

トピック

- [スタック](#)
- [レイヤー](#)
- [レシピと LifeCycle イベント](#)
- [インスタンス](#)
- [アプリケーション](#)
- [スタックのカスタマイズ](#)
- [リソース管理](#)

- [セキュリティと権限](#)
- [モニタリングとロギング](#)
- [CLI、SDK、および AWS CloudFormation テンプレート](#)
- [AWS OpsWorks Stacks サポート終了FAQs](#)
- [AWS OpsWorks Stacks アプリケーションを Application Manager AWS Systems Manager に移行する](#)
- [AWS OpsWorks Stacks Detach in Place ツールの使用](#)
- [AWS OpsWorks スタックの開始方法](#)
- [AWS OpsWorks スタックのベストプラクティス](#)
- [スタック](#)
- [レイヤー](#)
- [インスタンス](#)
- [アプリケーション](#)
- [クックブックとレシピ](#)
- [リソース管理](#)
- [タグ](#)
- [モニタリング](#)
- [セキュリティと権限](#)
- [AWS OpsWorks Chef 12 Linux のスタックサポート](#)
- [スタックでの以前の Chef AWS OpsWorks バージョンのサポート](#)
- [他の AWS サービスでの AWS OpsWorks スタックの使用](#)
- [AWS OpsWorks スタック CLI の使用](#)
- [デバッグとトラブルシューティングのガイド](#)
- [AWS OpsWorks スタックエージェント CLI](#)
- [AWS OpsWorks スタックデータバグリファレンス](#)
- [OpsWorks エージェントの変更](#)

スタック

スタックは、スタック AWS OpsWorks のコアコンポーネントです。基本的には、Amazon EC2 インスタンス、Amazon RDS データベースインスタンスなどのAWSリソースのコンテナであり、共通の

目的を持ち、論理的に一緒に管理する必要があります。スタックは、これらのリソースをグループとして管理し、インスタンスのオペレーティングシステムやAWSリージョンなどのデフォルト設定を定義するのに役立ちます。一部のスタックコンポーネントをユーザーの直接操作から分離する場合は、でスタックを実行できませんVPC。

レイヤー

1つ以上のレイヤーを追加することにより、スタックのコンポーネントを定義します。レイヤーは、アプリケーションの提供やデータベースサーバーのホスティングなど、特定の目的を果たすAmazon EC2インスタンスのセットを表します。

パッケージのデフォルト設定を変更してレイヤーをカスタマイズまたは展開したり、Chef レシピを追加して追加のパッケージのインストールなどのタスクを実行したりできます。

すべてのスタックについて、AWS OpsWorks スタックには以下のサービスを表すAWSサービスレイヤーが含まれています。

- Amazon Relational Database Service
- Elastic Load Balancing
- Amazon Elastic Container Service

レイヤーにより、インストールするパッケージ、インストールするパッケージの設定内容、アプリケーションをデプロイする方法などを完全に制御することができます。

レシピと LifeCycle イベント

レイヤーによって、[Chef レシピ](#)に基づいてインスタンスへのパッケージのインストール、アプリケーションのデプロイ、スクリプトの実行などのタスクが処理されます。AWS OpsWorks スタックの主な機能の1つは、セットアップ、設定、デプロイ、デプロイ解除、シャットダウンという一連のライフサイクルイベントです。これにより、各インスタンスで指定された一連のレシピが適切なタイミングで自動的に実行されます。

各レイヤーでは、これらの各ライフサイクルイベントに一連のレシピが割り当てられており、そのイベントとレイヤーのさまざまなタスクの処理に使用されます。例えば、ウェブサーバーレイヤーに属するインスタンスの起動が終了すると、AWS OpsWorks スタックは以下を実行します。

1. レイヤーの Setup レシピを実行します。これにより、ウェブサーバーのインストールや設定などのタスクが実行できます。

2. レイヤーの Deploy レシピを実行します。これにより、リポジトリからインスタンスにレイヤーのアプリケーションがデプロイされ、サービスの再開などの関連タスクが実行されます。
3. スタックのすべてのインスタンスで Configure レシピを実行すると、各インスタンスは新しいインスタンスに適合するように必要に応じて設定を調整できます。

たとえば、インスタンスでロードバランサーを実行しながら、Configure レシピで新しいインスタンスに含めるロードバランサーの設定を変更できます。

インスタンスが複数のレイヤーに属している場合、AWS OpsWorks スタックは各レイヤーのレシピを実行するため、たとえば、PHPアプリケーションサーバーと MySQL データベースサーバーをサポートするインスタンスを持つことができます。

レシピを実装している場合は、各レシピを適切なレイヤーとイベントに割り当てると、AWS OpsWorks スタックによって適切なタイミングで自動的に実行されます。また、レシピはいつでも手動で実行することができます。

インスタンス

インスタンスは、Amazon EC2インスタンスなどの単一のコンピューティングリソースを表します。インスタンスは、オペレーティングシステムやサイズなど基本的な設定を定義します。Elastic IP アドレスや Amazon EBSボリュームなどのその他の構成設定は、インスタンスのレイヤーによって定義されます。レイヤーのレシピは、パッケージをインストールして設定したりアプリケーションをデプロイしたりするタスクを実行することで設定を完了します。

AWS OpsWorks スタックを使用してインスタンスを作成し、レイヤーに追加できます。インスタンスを起動すると、AWS OpsWorks スタックはEC2インスタンスとそのレイヤーで指定された構成設定を使用して Amazon インスタンスを起動します。Amazon EC2インスタンスの起動が完了すると、AWS OpsWorks スタックはインスタンスとサービス間の通信を処理するエージェントをインストールし、ライフサイクルイベントに応じて適切なレシピを実行します。

AWS OpsWorks スタックでは、次のインスタンスタイプがサポートされています。インスタンスタイプは、起動と停止の方法によって特徴付けられます。

- 24/7 インスタンスは、ユーザーが手動で起動し、ユーザーが停止するまで実行されます。
- 時間ベースのインスタンスは、指定された日次および週次のスケジュールで AWS OpsWorks スタックによって実行されます。

このインスタンスにより、予想される使用パターンに対応するようにスタックのインスタンス数を自動的に調整することが可能となります。

- 負荷ベースのインスタンスは、CPU使用率などの指定された負荷メトリクスに基づいて、AWS OpsWorks スタックによって自動的に起動および停止します。

このインスタンスにより、スタックのインスタンス数を着信トラフィックの変動に対応するように自動的に調整することが可能となります。負荷ベースのインスタンスは Linux ベースのスタックにのみ使用できます。

AWS OpsWorks スタックはインスタンスの自動ヒーリングをサポートしています。エージェントがサービスとの通信を停止すると、AWS OpsWorks スタックは自動的にインスタンスを停止して再起動します。

スタックの外部で作成されたスタックに Linux ベースのコンピューティングリソースを組み込むこともできます AWS OpsWorks。

- Amazon EC2コンソール、CLIまたは を使用して直接作成した Amazon EC2インスタンスAPI。
- 仮想マシンで実行しているインスタンスを含め、独自のハードウェアで実行している On-premises (オンプレミス) インスタンス。

これらのインスタンスのいずれかを登録すると、そのインスタンスは AWS OpsWorks スタックインスタンスになり、スタックで AWS OpsWorks 作成するインスタンスとほぼ同じ方法で管理できます。

アプリケーション

Amazon S3 バケットなどのリポジトリに、アプリケーションおよび関連ファイルを保存します。各アプリケーションは、アプリケーションタイプを指定し、リポジトリからインスタンスにアプリケーションをデプロイするために必要なリポジトリURLやパスワードなどの情報を含むアプリケーションによって表されます。アプリケーションをデプロイすると、AWS OpsWorks スタックは Deploy イベントをトリガーし、スタックのインスタンスで Deploy レシピを実行します。

以下の方法でアプリケーションをデプロイすることができます。

- 自動 — インスタンスを起動すると、AWS OpsWorks スタックによってインスタンスの Deploy レシピが自動的に実行されます。

- 手動 - 新しいアプリケーションを準備する場合または既存のアプリケーションを更新する必要がある場合、ユーザーは手動でオンラインインスタンスの Deploy レシピを実行できます。

通常 AWS OpsWorks、スタックはスタック全体で Deploy レシピを実行するため、他のレイヤーのインスタンスは設定を適切に変更できます。ただし、すべてのアプリケーションサーバーのインスタンスに新しいアプリケーションをデプロイする前に、それをテストする必要がある場合などには、デプロイをインスタンスのサブセットに制限することができます。

スタックのカスタマイズ

AWS OpsWorks スタックでは、特定の要件に合わせてレイヤーをカスタマイズするためのさまざまな方法が提供されています。

- AWS OpsWorks スタックがパッケージを設定する方法を変更するには、さまざまな設定を表す属性を上書きするか、設定ファイルの作成に使用されるテンプレートを上書きします。
- スクリプトの実行や標準外のパッケージのインストールおよび設定などのタスクを実行する独自のレシピを指定することによって、既存のレイヤーを拡張することができます。

すべてのスタックには、レシピの最小限のセットで始める 1 つ以上のレイヤーを含めることができます。パッケージのインストール、アプリケーションのデプロイなどのタスクを処理するレシピを実装することで、レイヤーに機能を追加します。カスタムレシピおよび関連するファイルを 1 つ以上の cookbooks (クックブック) にまとめ、そのクックブックを Amazon S3 や Git などのリポジトリに保管します。

レシピは手動で実行できますが、スタックでは、5 AWS OpsWorks つのライフサイクルイベントのセットをサポートすることでプロセスを自動化することもできます。

- Setup イベントは、新しいインスタンスが正常にブートした後に発生します。
- Configure イベントは、インスタンスがオンライン状態に移行したときとオンライン状態から移行したときに、スタックのすべてのインスタンスで発生します。
- Deploy イベントは、ユーザーがアプリケーションをデプロイするときに発生します。
- Undeploy イベントは、ユーザーがアプリケーションを削除するときに発生します。
- Shutdown イベントは、ユーザーがインスタンスを停止するときに発生します。

各レイヤーで、各イベントに任意の数のレシピを割り当てることができます。レイヤーのインスタンスでライフサイクルイベントが発生すると、AWS OpsWorks Stacks は関連するレシピを実行し

ます。例えば、アプリサーバーインスタンスで Deploy イベントが発生すると、AWS OpsWorks Stacks はレイヤーの Deploy レシピを実行してアプリをダウンロードしたり、関連するタスクを実行したりできます。

リソース管理

[Elastic IP アドレス](#)などの他のAWSリソースをスタックに組み込むことができます。AWS OpsWorks スタックコンソールまたは を使用してAPI、リソースをスタックに登録したり、登録されたリソースをインスタンスにアタッチまたはデタッチしたり、リソースをあるインスタンスから別のインスタンスに移動したりできます。

セキュリティと権限

AWS OpsWorks スタックは AWS Identity and Access Management (IAM) AWS OpsWorks と統合され、ユーザーがスタックにアクセスする方法を制御する堅牢な方法を提供します。

- レイヤーやインスタンスなどのスタックリソースを作成できるかどうか、スタックの Amazon EC2インスタンスに接続RDPするために SSHまたは を使用できるかどうかなど、個々のユーザーが各スタックとやり取りする方法。
- スタックがユーザーに代わって Amazon AWS OpsWorks EC2インスタンスなどのAWSリソースとやり取りする方法。
- スタックインスタンスで実行されるアプリケーションが Amazon AWS OpsWorks S3 バケットなどのAWSリソースにアクセスする方法。 Amazon S3
- ユーザーのパブリックSSHキーとRDPパスワードを管理し、インスタンスに接続する方法。

モニタリングとロギング

AWS OpsWorks スタックには、スタックのモニタリングや、スタックやレシピに関する問題のトラブルシューティングに役立ついくつかの機能があります。すべてのスタック用:

- AWS OpsWorks スタックには Linux スタックのカスタム CloudWatch メトリクスのセットが用意されています。これらのメトリクスは、モニタリングページでわかりやすく要約されています。

AWS OpsWorks スタックは、Windows スタックの標準 CloudWatch メトリクスをサポートします。 CloudWatch コンソールでモニタリングできます。

- CloudTrail ログ。AWSアカウント内の AWS OpsWorks スタックによって、または スタックに代わって行われたAPI呼び出しを記録します。

- イベントログ。スタックのすべてのイベントをリスト表示します。
- Chef ログ。実行されたレシピや発生したエラーなど、各インスタンスで各ライフサイクルイベントに対して起こったことの詳細を表示します。

Linux ベースのスタックには、スタックのインスタンスに関する詳細モニタリングデータの収集と表示に使用できる Ganglia マスターレイヤーもあります。

CLI、SDK、および AWS CloudFormation テンプレート

コンソールに加えて、AWS OpsWorks スタックでは、コマンドラインインターフェイス (CLI) とを、任意のオペレーションの実行に使用できる複数の言語SDKsでもサポートしています。以下の機能を考慮してください。

- AWS OpsWorks スタックCLIは [AWS CLI](#)の一部であり、コマンドラインから任意のオペレーションを実行するために使用できます。

は複数のAWSサービスAWSCLIをサポートしており、Windows、Linux、または OS X システムにインストールできます。

- AWS OpsWorks スタックは [AWS Tools for Windows PowerShell](#) に含まれており、Windows PowerShell コマンドラインから任意のオペレーションを実行するために使用できます。
- AWS OpsWorks スタックSDKは に含まれておりAWSSDKs、[Java](#)、[JavaScript](#) (ブラウザベースおよび Node.js)、[.NET](#)、[Python \(boto\)](#)、または [Ruby PHP](#) で実装されているアプリケーションで使用できます。

AWS CloudFormation テンプレートを使用してスタックをプロビジョニングすることもできます。いくつかの例については、[AWS OpsWorks 「スニペット」](#)を参照してください。

AWS OpsWorks Stacks サポート終了FAQs

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。

トピック

- [このサポート終了により、既存のお客様にはどのような影響がありますか？](#)
- [新規顧客 AWS OpsWorks Stacks を受け入れていますか？](#)
- [既存のスタックはどこに移行すればよいですか？](#)
- [サポート終了後に既存の Amazon EC2 インスタンスを維持するにはどうすればよいですか？](#)
- [サポート終了はすべての AWS リージョン に同時に影響しますか？](#)
- [で利用できるテクニカルサポートのレベル AWS OpsWorks Stacks は？](#)
- [の新機能リリースはありますか AWS OpsWorks Stacks？](#)

このサポート終了により、既存のお客様にはどのような影響がありますか？

既存のお客様は、AWS OpsWorks Stacks のサポート終了日である 2024 年 5 月 26 日まで影響を受けません。2024 年 5 月 26 日以降、お客様は OpsWorks コンソール、API、CLI、および CloudFormation リソースを使用できません。

新規顧客 AWS OpsWorks Stacks を受け入れていますか？

いいえ。AWS OpsWorks Stacks は新規顧客を受け入れなくなり、現時点では既存の顧客のみが新しいスタックを作成できるようになりました。

既存のスタックはどこに移行すればよいですか？

AWS OpsWorks Stacks 以下の機能を活用できる AWS Systems Manager にワークロードを移行することをお勧めします。

- Modern Chef のバージョン
- SSM Agent
- アプリケーション ロード バランサー
- Auto Scaling グループによるスケーリング機能の強化
- EC2 起動テンプレートを使用して希望するホスト特性を定義可能
- より新しいインスタスタイプ
- 新しい EBS ボリュームタイプ

Systems Manager の詳細情報については、[AWS Systems Manager ユーザーガイド](#)を参照してください。への移行については AWS Systems Manager、「」を参照してください。[AWS OpsWorks Stacks アプリケーションを Application Manager AWS Systems Manager に移行する](#)

サポート終了後に既存の Amazon EC2 インスタンスを維持するにはどうすればよいですか？

サポート終了日に達すると、Amazon EC2 インスタンスはアカウントに残りますが、OpsWorks スタックサービスを使用してインスタンスを制御および管理できなくなります。

AWS OpsWorks Stacks Detach in Place ツールを使用して、OpsWorks スタックサービスから OpsWorks インスタンスをデタッチできます。デタッチ後、Amazon EC2 AWS Systems Manager、または任意の EC2 互換アプローチを使用して、インスタンスを設定および管理できます。詳細については、「[AWS OpsWorks Stacks Detach in Place ツールの使用](#)」を参照してください。

サポート終了はすべての AWS リージョン に同時に影響しますか？

はい。OpsWorks コンソール、API、CLI、CloudFormation およびリソースは、2024 年 5 月 26 日にすべての AWS リージョン 同時に廃止されます。AWS OpsWorks Stacks が利用可能な のリストについては、AWS リージョン [AWS 「リージョンサービスリスト」](#)を参照してください。

で利用できるテクニカルサポートのレベル AWS OpsWorks Stacksは？

AWS は、サポート終了日まで、お客様が現在持っている AWS OpsWorks Stacks のと同じレベルのサポートを引き続き提供します。ご質問やご不明点がございましたら、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

の新機能リリースはありますか AWS OpsWorks Stacks？

いいえ。サービスのサポート終了が近づいているため、新特徴量のリリースは行いません。しかし、サポート終了日までは引き続きセキュリティの改善を行い、Amazon EC2 インスタンスを予定どおりに管理します。

AWS OpsWorks Stacks アプリケーションを Application Manager AWS Systems Manager に移行する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。

移行スクリプト AWS Systems Manager を使用して、アプリケーションを の一機能である AWS OpsWorks Stacks [Application Manager](#) に移行できるようになりました。スタックアプリケーションを Systems Manager Application Manager に移行すると AWS OpsWorks Stacks、Graviton などの新しい Amazon EC2 インスタンスタイプ、gp3 などの新しい Amazon Elastic Block Store (EBS) ボリューム、新しいオペレーティングシステム、Auto Scaling グループとの統合、アプリケーションロードバランサーなど、 で利用できない AWS 機能を使用できます。

このリリースでは、Systems Manager Application Manager から利用できる新しいインスタンスタブを使用して、移行したインスタンスの操作を監視および実行できるようになりました。インスタンスタブを使用して、複数の AWS インスタンスを 1 か所で表示できます。このタブでは、インスタンスの状態に関する情報を表示したり、問題をトラブルシューティングしたりできます。「インスタンス」タブの操作について詳しくは、「AWS Systems Manager ユーザーガイド」の「[アプリケーションインスタンスの操作](#)」を参照してください。

トピック

- [スクリプトの仕組み](#)
- [前提条件](#)
- [制限事項](#)
- [開始](#)
- [よくある質問](#)
- [トラブルシューティング](#)

スクリプトの仕組み

AWS OpsWorks には、CloudFormation テンプレートを使用して AWS OpsWorks Stacks アプリケーションを Systems Manager Application Manager に移行するために実行できるスクリプトが用意されています。スクリプトは、既存の OpsWorks レイヤーに関する情報を取得し、スクリプトの `--provision-application` パラメータの値に応じて、アプリケーションのクローンをプロビジョニングするか、を使用して変更できるスターター CloudFormation テンプレートを提供します AWS CloudFormation。

前提条件

- AWS CLI がインストールされ、設定されていることを確認します。のインストールの詳細については AWS CLI、「AWS Command Line Interface ユーザーガイド」の「[の最新バージョンのインストールまたは更新 AWS CLI](#)」を参照してください。

Note

を設定しない場合は AWS CLI、を使用してコマンドを実行することもできます AWS CloudShell。の操作の詳細については CloudShell、「[ユーザーガイド](#)」の [AWS CloudShell](#)「の操作AWS CloudShell」を参照してください。

- Python バージョニング 3.6 以降がインストールされている、または Amazon マシンイメージ (AMI) が付属していることを確認してください。
- ご使用のオペレーティングシステムがサポートされていることを確認してください。移行スクリプトは、次のオペレーティングシステムでダウンロードして実行できます。
 - Amazon Linux および Amazon Linux 2
 - Ubuntu 18.04 LTS, 20.04 LTS, 22.04 LTS
 - Red Hat Enterprise Linux 8
 - Windows Server 2019、Windows 10 Enterprise

Note

Windows Server 2022 はサポートされていません。

制限事項

新しい OpsWorks アーキテクチャは、のアーキテクチャとは異なります AWS OpsWorks Stacks。このセクションでは、このアーキテクチャの既知の制限について説明します。

新しい OpsWorks アーキテクチャでは、以下はサポートされていません。

- Windows と CentOS インスタンスでの Chef レシピの実行
- 組み込み Chef 11 レイヤーと Berkshelf
- Chef の属性とデータバッグ
- オンプレミスインスタンス
- EC2 からインポートされたインスタンス
- ユーザー指定のオペレーティングシステムパッケージリストのインストールはサポートされていません。
- アプリはサポートも移行もされていない

以下は制限付きでサポートされています。

- 移行スクリプトは EBS ボリューム情報を複製しますが、ボリュームに含まれるマウントポイントと実際のデータは除外します。
- 時間ベースおよび負荷ベースのスケールされたインスタンスは移行されますが、これらのインスタンスに関連するスケールリングールは移行されません。Auto Scaling グループを変更して同様の結果を得ることができます。
- OpsWorks コンソールのスタックのアクセス許可ページで定義されている IAM エンティティは作成も生成もされません。
- 移行スクリプトは、Systems Manager でシングルレイヤーアプリケーションをプロビジョニングすることしかできません。たとえば、同じスタック内の二つのレイヤーに対してスクリプトを 2 回実行すると、Systems Manager に二つの異なるアプリケーションが表示されます。

開始

移行スクリプト `stack_exporter.py` は、ローカルまたは EC2 インスタンスで実行できる Python スクリプトです。スクリプトを実行する前に、すべての前提条件が満たされていることを確認してください。前提条件の詳細については、「[前提条件](#)」を参照してください。

以下のセクションのステップでは、OpsWorks スタックを Systems Manager Application Manager に移行する方法を示します。

トピック

- [ステップ 1: スクリプトを実行する環境の準備](#)
- [ステップ 2: 移行スクリプトのダウンロード](#)
- [ステップ 3: スクリプトを実行する環境セットアップ](#)
- [ステップ 4: スクリプトを実行する](#)
- [ステップ 5: CloudFormation スタックをプロビジョニングする](#)
- [ステップ 6: プロビジョニングされたリソースの確認](#)
- [ステップ 7: インスタンスを起動する](#)
- [ステップ 8: インスタンスを確認する](#)
- [ステップ 9: Systems Manager アプリケーションマネージャーを使用してインスタンスのオペレーションを監視および実行する](#)

ステップ 1: スクリプトを実行する環境の準備

オペレーティングシステムに適したコマンドを実行して、環境を準備します。

トピック

- [Amazon Linux 2](#)
- [Amazon Linux](#)
- [Ubuntu 18.04, 20.04, 22.04](#)
- [Red Hat Enterprise Linux 8](#)
- [Windows Server 2019、Windows 10 Enterprise](#)

Amazon Linux 2

```
sudo su
python3 -m pip install pipenv
PATH="$PATH:/usr/local/bin"
yum update
yum install git
```

Amazon Linux

```
sudo su
PATH="$PATH:/usr/local/bin"
export LC_ALL=en_US.utf-8
export LANG=en_US.utf-8
yum update
yum list | grep python3
yum install python36 // Any python version
yum install git
```

Python バージョン 3.6 の場合も実行 :

```
python3 -m pip install pipenv==2022.4.8
```

Python バージョン 3.7 以降の場合も実行 :

```
python3 -m pip install pipenv
```


Ubuntu 18.04, 20.04, 22.04

```
sudo su
export PATH="${HOME}/.local/bin:$PATH"
apt-get update
apt install python3-pip
apt-get install git // if git is not installed
python3 -m pip install --user pipenv==2022.4.8
```

Red Hat Enterprise Linux 8

```
sudo su
sudo dnf install python3
PATH="$PATH:/usr/local/bin"
yum update
yum install git
python3 -m pip install pipenv==2022.4.8
```

Windows Server 2019、Windows 10 Enterprise

 Note

Windows Server 2019 の場合は、Python バージョン 3.6.1 以降をインストールします。

```
pip install pipenv
```

Git がまだインストールされていない場合は、[Git](#)をダウンロードしてインストールします。

Git をクックブックソースとして使用する場合は、Windows でスクリプトを実行する前に Git サーバーを known_hosts ファイルに追加してください。PowerShell を使用して、次の関数を作成できます。

```
function add_to_known_hosts($server){
    $new_host=$(ssh-keyscan $server 2> $null)
    $existing_hosts=''
    if (!(test-path "$env:userprofile\.ssh")) {
        md "$env:userprofile\.ssh"
    }
    if ((test-path "$env:userprofile\.ssh\known_hosts")) {
        $existing_hosts=Get-Content "$env:userprofile\.ssh\known_hosts"
    }
    $host_added=0
    foreach ($line in $new_host) {
        if (!(($existing_hosts -contains $line)) {
            Add-Content -Path "$env:userprofile\.ssh\known_hosts" -Value $line
            $host_added=1
        }
    }
    if ($host_added) {
        echo "$server has been added to known_hosts."
    } else {
        echo "$server already exists in known_hosts."
    }
}
```

その後、この関数を実行するときに、Git サーバー (github.com、git-codecommit など *repository_region*.amazonaws.com) を指定できます。


```
add_to_known_hosts "myGitServer"
```

ステップ 2: 移行スクリプトのダウンロード

次のコマンドを実行して、移行スクリプトとすべての関連ファイルを含む ZIP ファイルをダウンロードします。

```
aws s3api get-object \  
  --bucket export-opsworks-stacks-bucket-prod-us-east-1 \  
  --key export_opsworks_stacks_script.zip export_opsworks_stacks_script.zip
```

Linux を使用している場合は、以下のコマンドを使用して unzip ユーティリティをインストールします。

```
sudo apt-get install unzip  
sudo yum install unzip
```

オペレーティング システムに適したコマンドを使用してファイルを解凍します。

Linux の場合は、次のコマンドを使用します。

```
unzip export_opsworks_stacks_script.zip
```

Windows の場合は、 の Expand-Archive コマンドを使用します PowerShell。

```
Expand-Archive -LiteralPath PathToZipFile -DestinationPath PathToDestination
```

ファイルを解凍すると、次のディレクトリとファイルが使用可能になります。

- README.md
- LICENSE
- NOTICE
- requirements.txt
- テンプレート/
 - OpsWorksCFNTemplate.yaml
 - MountEBSVolumes.yaml
- opsworks/

- cloudformation/
- インスタンス _tab/
- cfn_stack_deployer.py
- s3.py
- stack_exporter_context.py
- stack_exporter.py

ステップ 3: スクリプトを実行する環境セットアップ

以下のコマンドを使用して、スクリプトを実行する環境を設定します。

```
pipenv install -r requirements.txt
pipenv shell
```

Note

現在、このスクリプトは Application Manager で単一レイヤーのアプリケーションのみをプロビジョニングできます。たとえば、同じスタック内の二つのレイヤーに対してスクリプトを 2 回実行すると、そのスクリプトによって Application Manager に二つの異なるアプリケーションが作成されます。

環境を設定したら、スクリプトパラメータを確認します。python3 stack_exporter.py --help コマンドを実行すると、移行スクリプトで使用可能なオプションを確認できます。

パラメータ	説明	必須	タイプ	デフォルト値
--layer-id	この OpsWorks レイヤー ID の CloudFormation テンプレートをエクスポートします。	あり	string	
--region	OpsWorks スタックの AWS リージョン。OpsWorks スタック リージョンと API エンドポイントリージョンが異なる場合は、スタックリージョンを使用します。	なし	string	us-east-1

パラメータ	説明	必須	タイプ	デフォルト値
	これは、スタックの OpsWorks 他のリソース部分 (EC2 インスタンスやサブネットなど) と同じリージョンです。			
-- provision- application	デフォルトでは、スクリプトは CloudFormation テンプレートによってエクスポートされたアプリケーションをプロビジョニングします。このパラメータを FALSE の値でスクリプトに渡して、テンプレートのプロビジョニングをスキップします CloudFormation。	なし	ブール値	TRUE

パラメータ	説明	必須	タイプ	デフォルト値
-- launch- template	<p>このパラメータは、既存の起動テンプレートを使用するか、新しい起動テンプレートを作成するかを定義します。推奨インスタンスプロパティを使用するか、オンラインインスタンスと一致するインスタンスプロパティを使用する新しい起動テンプレートを作成できます。</p> <p>有効な値を次に示します。</p> <ul style="list-style-type: none">• RECOMMENDED - スタックの OS と c5.large インスタンスサイズに OpsWorks最新の AMI のインスタンス特性を使用します。• MATCH_LAST_INSTANCE - 利用可能な最新のオンラインインスタンス特性を使用します。• <i>LaunchTemplateID</i> / [<i>LaunchTemplateVersion</i>] - 既存の起動テンプレートを使用するには 必要に応じて、テンプレートバージョンを指定できます。テンプレートバージョンを指定しない場合、スクリプトはデフォルトバージョンを使用します。	なし	string	RECOMMENDED

パラメータ	説明	必須	タイプ	デフォルト値
<code>--system-updates</code>	<p>カーネルとパッケージの更新をインスタンスの起動時に実行するかどうかを定義します。</p> <p>有効な値を次に示します。</p> <ul style="list-style-type: none"> • <code>ALL_UPDATES</code> - インスタンスの起動時にカーネルとパッケージのシステム更新を実行します。 • <code>NO_UPDATES</code> - インスタンス起動時にシステム更新を実行しません。 • <code>MATCH_LAYER_SETTINGS</code> - OpsWorksレイヤーの または インスタンスの <code>InstallUpdatesOnBoot</code> プロパティを使用して、システム更新をインストールするかどうかを決定します。 	なし	string	<code>ALL_UPDATES</code>
<code>--http-username</code>	カスタムクックブックを含む HTTP アーカイブへの認証に使用されるユーザー名を格納する <code>Systems Manager SecureString</code> パラメーターの名前です。	なし	string	
<code>--http-password</code>	カスタムクックブックを含む HTTP アーカイブへの認証に使用されるパスワードを格納する <code>Systems Manager SecureString</code> パラメーターの名前です。	なし	string	

パラメータ	説明	必須	タイプ	デフォルト値
<code>--repo-private-key</code>	カスタムクックブックを含まリポジトリへの認証に使用される SSH キーを格納する Systems Manager SecureString パラメータの名前です。リポジトリがある場合は GitHub、新しい SSH Ed25519 キーを生成する必要があります。新しい SSH Ed25519 キーを生成しない場合、GitHub リポジトリへの接続は失敗します。	なし	string	
<code>--lb-type</code>	既存のロードバランサーを移行する際に作成するロードバランサーのタイプです。 有効な値を次に示します。 <ul style="list-style-type: none">ALB (Application load balancer)Classis (Classic load balancer)None (ロードバランサーを作成したくない場合)	なし	string	ALB

パラメータ	説明	必須	タイプ	デフォルト値
<code>--lb-access-logs-path</code>	既存の S3 バケットへのパスと、ロードバランサーのアクセスログを保存するためのプレフィックスです。S3 バケットとロードバランサーは同じリージョンに存在する必要があります。値を指定せず、 <code>--lb-type</code> パラメータ値が <code>None</code> に設定されている場合、スクリプトは新しい S3 バケットとプレフィックスを作成します。このプレフィックスには適切なバケットポリシーがあることを確認してください。	なし	string	
<code>--enable-instance-protection</code>	TRUE に設定すると、スクリプトは Auto Scaling グループのカスタム終了ポリシー (Lambda 関数) を作成します。protected_instance タグ付きの EC2 インスタンスはスケールインイベントから保護されます。スケールインイベントから保護したい各 EC2 インスタンスに protected_instance タグを追加します。	なし	ブール値	FALSE
<code>--command-logs-bucket</code>	AWS ApplyChefRecipe と MountEBSVolumes ログを保存する既存の S3 バケットの名前です。値を指定しない場合、スクリプトによって新しい S3 バケットが作成されます。	なし	string	aws-opsworks-application-manager-logs- <i>account-id</i>

パラメータ	説明	必須	タイプ	デフォルト値
-- custom- json- bucket	カスタム JSON を保存する既存の S3 バケットの名称です。値を指定しない場合、スクリプトによって新しい S3 バケットが作成されます。	なし	string	aws-apply- chef-app- lication- manager-t ransition- data- <i>account- id</i>

注意:

- プライベート GitHub リポジトリを使用する場合は、SSH 用の新しい Ed25519 ホストキーを作成する必要があります。これは、SSH でサポートされているキー GitHub が変更され、暗号化されていない Git プロトコルが削除されたためです。Ed25519 ホストキーの詳細については、ブログ記事「[での GitHub Git プロトコルセキュリティの改善](#)」を参照してください。 [GitHub](#) 新しい Ed25519 ホストキーを生成したら、SSH キー用の Systems Manager SecureString パラメータを作成し、その SecureString パラメータ名を --repo-private-key パラメータの値として使用します。Systems Manager SecureString パラメータの作成方法の詳細については、AWS Systems Manager ユーザーガイドの [SecureString 「パラメータの作成 \(AWS CLI\)」](#) または [「Systems Manager パラメータの作成 \(コンソール\)」](#) を参照してください。
- --http-username、--http-password および --repo-private-key パラメータは、Systems Manager SecureString パラメータの名前を指します。移行スクリプトは、AWS-ApplyChefRecipes ドキュメントを実行するときにこれらのパラメータを使用します。
- --http-username パラメータでは、--http-password パラメータの値も指定する必要があります。
- --http-password パラメータでは、--http-username パラメータの値も指定する必要があります。
- --http-password と --repo-private-key の両方に値を設定しないでください。Systems Manager SecureString パラメータ名または SSH キー (--repo-private-key)、またはリポジトリユーザー名 (--http-username) とパスワード (--http-password) のいずれかを指定します。

ステップ 4: スクリプトを実行する

`python3 stack_exporter.py` を実行するときは、アプリケーションをプロビジョニングするか、`--provision-application` パラメータの値を `FALSE` に設定してスターターテンプレートを作成できます。

例 1: Systems Manager アプリケーションマネージャアプリケーションのプロビジョニング

次のコマンドは、既存の OpsWorks レイヤーに関する情報を取得し、新しい OpsWorks アーキテクチャを使用してアプリケーションをプロビジョニングします。これにより、スタックに設定された Chef バージョンと同様の結果が得られます。このスクリプトは、を使用して Auto Scaling グループなど、必要なすべてのリソースをプロビジョニングし CloudFormation、そのアプリケーションを Systems Manager Application Manager に登録します。

`stack-region` と `layer-id` を OpsWorks スタックとレイヤーの値に置き換えます。

```
python3 stack_exporter.py \  
  --layer-id layer-id \  
  --region stack-region
```

例 2: テンプレートの生成

次のコマンドは、既存の OpsWorks レイヤーに関する情報を取得し、CloudFormation テンプレートを生成します。テンプレートをプロビジョニングすると、Chef 14 を使用した場合と同様の結果が得られます。この例では、`--provision-application` パラメータが `FALSE` に設定されているため、リソースはプロビジョニングされません。

`stack-region` と `layer-id` を OpsWorks スタックとレイヤーの値に置き換えます。

```
python3 stack_exporter.py \  
  --layer-id layer-id \  
  --region stack-region \  
  --provision-application FALSE
```

コマンドを実行したら、Systems Manager の Application Manager テンプレートライブラリ内のテンプレートを確認したり、テンプレートをプロビジョニングしたりできます。テンプレートライブラリの表示について詳しくは、「AWS Systems Manager ユーザーガイド」の「[テンプレートライブラリの操作](#)」を参照してください。

ステップ 5: CloudFormation スタックをプロビジョニングする

Note

このステップを完了する必要があるのは、スクリプトの `--provision-application` パラメータを `FALSE` に設定する場合だけです。

パラメータを の値 `--provision-application` で指定すると `FALSE`、スクリプト出力は CloudFormation テンプレートの名前と URL を提供します。このテンプレートは、既存の OpsWorks スタックとレイヤーの代替案を表します。

テンプレートは、Application Manager テンプレートライブラリ (推奨) または を使用してプロビジョニングできます CloudFormation。テンプレートライブラリの操作については、「AWS Systems Manager ユーザーガイド」の「[テンプレートライブラリの操作](#)」を参照してください。

ステップ 6: プロビジョニングされたリソースの確認

これで、プロビジョニングされたリソースを確認する準備ができました。

1. AWS CloudFormation コンソールを使用して、プロビジョニングされたスタックのリソースを確認します。
 - a. <https://console.aws.amazon.com/cloudformation> で AWS CloudFormation コンソールを開き、スタック を選択します。
 - b. 「Stacks」 ページでスタックを選択し、次に「Resources」 タブを選択します。
 - c. 「Resources」 タブで、スタックにリストされているリソースを確認します。リソースのリストには EC2 Auto Scaling グループが含まれており、Auto Scaling コンソール、またはで確認できます AWS CLI。
2. Systems Manager Application Manager を使用して、アプリケーションのリソースを確認します。
 - a. Systems Manager コンソール (<https://console.aws.amazon.com/systems-manager/>) を開きます。
 - b. ナビゲーションペインで、[Application Manager] を選択します。
 - c. [アプリケーション] セクションで、[カスタムアプリケーション] を選択します。Application Manager の [概要] タブが開きます。

- d. [Resources] タブを選択します。リソースタブには、OpsWorks スタックとレイヤーに移行されたすべてのリソースが表示されます。アプリケーション名には OpsWorks スタックの名前が含まれ、`app -stack-name -suffix` の形式になります。`#####`はスタック ID の最初の 6 文字を表します。Application Manager でのリソースの表示について詳しくは、「AWS Systems Manager ユーザーガイド」の「[アプリケーションリソースの表示](#)」を参照してください。

ステップ 7: インスタンスを起動する

インスタンスをプロビジョニングしたら、インスタンスをテストする準備が整います。この時点では、実行中のインスタンスはありません。

インスタンスをオンラインにするには、Auto Scaling グループの Min、Max、Desired capacity の値を、アプリケーションに適した数値に調整します。最初は、これらの値を 1 に設定して 1 つのインスタンスをオンラインにし、そのインスタンスがカスタム Chef レシピの実行を含め、期待されるすべてのアクションを実行することを確認するとよいでしょう。

ステップ 8: インスタンスを確認する

インスタンスを起動したら、期待どおりに動作することを確認します。

1. `--command-logs-bucket` スクリプトのパラメータで指定された S3 バケットにある Chef startup と terminate ログを確認します。デフォルトでは、ログはという名前のバケットに保存されます `aws-opsworks-application-manager-logs-account-id`.
 - a. にサインイン AWS Management Console し、<https://console.aws.amazon.com/s3/> で Amazon S3 コンソールを開きます。
 - b. ログを含むバケットを選択します。
 - c. ApplyChefRecipes プレフィックスに移動してログを表示します。
2. Application Load Balancer の接続と状態を確認します。

ロードバランサーのアクセスログを表示するには、次のステップに従います。ロードバランサーのアクセスログを保存する S3 バケットは、`--lb-access-logs-path` スクリプトのパラメータを使用して指定できます。

- a. にサインイン AWS Management Console し、<https://console.aws.amazon.com/s3/> で Amazon S3 コンソールを開きます。
- b. S3 バケットを選択し、ログを含むプレフィックスに移動します。

3. インスタンスが Auto Scaling と Application Load Balancer のすべてのヘルスチェック (設定している場合) に合格することを確認します。

Auto Scaling のヘルスに関する情報は、新しい [インスタンス] タブで確認できます。

- a. Systems Manager コンソール (<https://console.aws.amazon.com/systems-manager/>) を開きます。
- b. ナビゲーションペインで、[Application Manager] を選択します。
- c. [アプリケーション] セクションで、[カスタムアプリケーション] を選択します。
- d. リストからアプリケーションを選択します。Application Manager の [概要] タブが開きます。
- e. [インスタンス] タブを選択すると、Auto Scaling のヘルスに関する情報が表示されます。

Chef レシピが正常に実行されたことを確認したら、Auto Scaling グループの容量を減らしてインスタンスを終了できます。カスタム終了レシピがある場合は、そのレシピが期待どおりに動作することを確認してください。

ステップ 9: Systems Manager アプリケーションマネージャーを使用してインスタンスのオペレーションを監視および実行する

Application Manager ページの新しい [インスタンス] タブを使用して、インスタンスのオペレーションを監視および実行できるようになりました。「インスタンス」タブの操作については、「AWS Systems Manager ユーザーガイド」の「[アプリケーションインスタンスの操作](#)」を参照してください。

「インスタンス」タブを使用すると、複数の AWS インスタンスを 1 か所に表示できます。このタブでは、インスタンスの状態に関する情報を表示したり、問題をトラブルシューティングしたりできます。

My-Sample-Stack--Linux--Node-js-App-Server-b4340f

Application information

Application type: CustomGroup
Name: My-Sample-Stack--Linux--Node-js-App-Server-b4340f
Application monitoring: Not enabled
Application tags: 1

Overview | Resources | **Instances** | Compliance | Monitoring | OpsItems | Logs | Runbooks

Instances

Instance State: Instance lifecycle state. Running 1 / 100%

Auto Scaling health checks: Amazon EC2, Elastic Load Balancing, and custom health checks (aggregated). Healthy 1 / 100%

Instance status: Amazon EC2 instance system and status checks. OK 1 / 100%

All instances (1)

Instance ID	State	SSM Ping	Last execution	Alarms	Parent ASG	ASG Health
i-Oca3fba229a52a924	Running	Online	AWS-ApplyChefRecipes	0	My-Sample-Stack-Linux-N...	Healthy

Instances タブを表示するには、次の手順を実行します。

1. Systems Manager コンソール (<https://console.aws.amazon.com/systems-manager/>) を開きます。
2. ナビゲーションペインで、[Application Manager] を選択します。
3. [アプリケーション] セクションで、[カスタムアプリケーション] を選択します。
4. リストからアプリケーションを選択します。Application Manager の [概要] タブが開きます。
5. 「Instances」タブを選択すると、インスタンスのステータスと EC2 の状態に関する情報が表示されます。

よくある質問

以下の FAQ では、よく寄せられる質問に対する回答を示します。

トピック

- [どの AWS OpsWorks Stacks バージョンを移行できますか？](#)
- [移行したインスタンスではどの Chef バージョンを使用できますか？](#)
- [どのリポジトリタイプを移行できますか？](#)
- [プライベート Git リポジトリを引き続き使用できますか？](#)
- [インスタンスへのアクセスにはどの SSH キーを使用できますか？](#)
- [インスタンスが自動的にスケールイン/スケールアウトされるのはなぜですか？](#)
- [自動スケーリングをオフにすることはできますか？](#)
- [起動した EC2 インスタンスでカーネルとパッケージの更新を実行できますか？](#)
- [インスタンスの EBS ボリュームにデータが含まれていないのはなぜですか？](#)
- [起動テンプレートに記載されている EBS ボリュームがマウントされないのはなぜですか？](#)
- [Chef レシピとマウント EBS ボリュームのログはどこにありますか？](#)
- [移行スクリプトのデバッグログはどこから入手できますか？](#)
- [移行スクリプトは CloudFormation テンプレートのバージョンングをサポートしていますか？](#)
- [複数の Layer を移行できますか？](#)
- [SecureString パラメータを作成するにはどうしたらよいですか？](#)
- [新しい Auto Scaling グループのインスタンスを終了イベントから保護する方法を教えてください。](#)
- [移行スクリプトではどのようなロードバランサーを利用できますか？](#)
- [カスタムクックブック設定レシピは移行されますか？](#)
- [新しく作成したインスタンスで、デプロイ、アンデプロイのレシピを実行できますか？](#)
- [Auto Scaling グループが対象とするサブネットを変更できますか？](#)

どの AWS OpsWorks Stacks バージョンを移行できますか？

Chef 11.10 と Chef 12、Amazon Linux、Amazon Linux 2、Ubuntu、および Red Hat Enterprise Linux 7 のスタックのみを移行できます。

移行したインスタンスではどの Chef バージョンを使用できますか？

移行したインスタンスは Chef バージョン 11 ~ 14 を使用できます。

Note

Windows のスタックの移行はサポートされていません。

どのリポジトリタイプを移行できますか？

S3、Git、および HTTP のリポジトリタイプを移行できます。

プライベート Git リポジトリを引き続き使用できますか？

はい、プライベート Git リポジトリを引き続き使用できます。

プライベート GitHub リポジトリを使用する場合は、SSH 用の新しい Ed25519 ホストキーを作成する必要があります。これは、SSH でサポートされているキー GitHub が変更され、暗号化されていない Git プロトコルが削除されたためです。Ed25519 ホストキーの詳細については、GitHub ブログ記事「[での Git プロトコルセキュリティの改善 GitHub](#)」を参照してください。新しい Ed25519 ホストキーを生成したら、この SSH キー用の Systems Manager SecureString パラメータを作成し、パラメータ名を `--repo-private-key` パラメータの値として使用します。Systems Manager SecureString パラメータの作成方法の詳細については、「[ユーザーガイド](#)」の [SecureString「パラメータの作成 \(AWS CLI\) AWS Systems Manager](#)」を参照してください。

その他の Git リポジトリタイプでは、この SSH キー用の Systems Manager SecureString パラメータを作成し、そのパラメータ名をスクリプトの `--repo-private-key` パラメータの値として使用します。

インスタンスへのアクセスにはどの SSH キーを使用できますか？

スクリプトを実行すると、スクリプトはスタックに設定されている SSH キーとインスタンスを移行します。SSH キーを使用してインスタンスにアクセスできます。スタックとインスタンスに SSH キーが提供されている場合、スクリプトはスタックのキーを使用します。どの SSH キーを使用すべきかわからない場合は、EC2 コンソール (<https://console.aws.amazon.com/ec2/>) でインスタンスを確認してください。EC2 コンソールの 詳細 ページには、インスタンスの SSH キーが表示されます。

インスタンスが自動的にスケールイン/スケールアウトされるのはなぜですか？

自動スケーリングは、Auto Scaling グループのスケールしているルールに基づいてインスタンスをスケールします。グループの [最小]、[最大]、[必要な容量] の値を設定できます。Auto Scaling グループは、これらの値を更新すると、それに応じて容量を自動的にスケールします。

自動スケーリングをオフにすることはできますか？

Auto Scaling グループの [最小]、[最大]、[必要な容量] の値を同じ数値に設定することで、自動スケーリングを無効にできます。たとえば、常に 10 個のインスタンスを用意したい場合は、[最小]、[最大]、[必要な容量] の値を 10 に設定します。

起動した EC2 インスタンスでカーネルとパッケージの更新を実行できますか？

デフォルトでは、EC2 インスタンスの起動時にカーネルとパッケージの更新が行われます。次の手順を使用して、起動した EC2 インスタンスでカーネルまたはパッケージの更新を実行します。例えば、デプロイまたは設定のレシピを実行した後に、更新を適用できます。

1. EC2 インスタンスに接続します。
2. 次の `perform_upgrade` 関数を作成し、インスタンスで実行します。

```
perform_upgrade() {
    #!/bin/bash
    if [ -e '/etc/system-release' ] || [ -e '/etc/redhat-release' ]; then
        sudo yum -y update
    elif [ -e '/etc/debian_version' ]; then
        sudo apt-get update
        sudo apt-get dist-upgrade -y
    fi
}
perform_upgrade
```

3. カーネルとパッケージの更新後、EC2 インスタンスの再起動が必要な場合があります。再起動が必要かどうかを確認するには、次の `reboot_if_required` 関数を作成して EC2 インスタンスで実行します。

```
reboot_if_required () {
    #!/bin/bash
    if [ -e '/etc/debian_version' ]; then
        if [ -f /var/run/reboot-required ]; then
```



```
    echo "reboot is required"
else
    echo "reboot is not required"
fi
elif [ -e '/etc/system-release' ] || [ -e '/etc/redhat-release' ]; then
    export LC_CTYPE=en_US.UTF-8
    export LC_ALL=en_US.UTF-8
    LATEST_INSTALLED_KERNEL=`rpm -q --last kernel | perl -X -pe 's/^kernel-(\S+).*/$1/' | head -1`
    CURRENTLY_USED_KERNEL=`uname -r`
    if [ "${LATEST_INSTALLED_KERNEL}" != "${CURRENTLY_USED_KERNEL}" ];then
        echo "reboot is required"
    else
        echo "reboot is not required"
    fi
fi
}
reboot_if_required
```

4. `reboot_if_required` 実行結果が `reboot is required` メッセージに表示された場合は、EC2 インスタンスを再起動します。`reboot is not required` メッセージを受け取った場合、EC2 インスタンスを再起動する必要はありません。

インスタンスの EBS ボリュームにデータが含まれていないのはなぜですか？

スクリプトを実行すると、スクリプトは EBS ボリュームの設定を移行し、OpsWorks スタックとレイヤーの代替アーキテクチャを作成します。このスクリプトでは、実際のインスタンスやインスタンスに含まれるデータは移行されません。このスクリプトは、EBS ボリュームの設定を Layer レベルで移行し、空の EBS ボリュームを起動した EC2 インスタンスにアタッチするだけです。

次の手順を実行して、以前のインスタンスの EBS ボリュームからデータを引き出します。

1. 以前のインスタンスの EBS ボリュームのスナップショットを取得します。スナップショットの作成の詳細については、Amazon EC2 ユーザーガイドの [「Amazon EBS スナップショットの作成」](#) を参照してください。
2. スナップショットからボリュームを作成する スナップショットからボリュームを作成する方法の詳細については、Amazon EC2 ユーザーガイドの [「スナップショットからボリュームを作成する」](#) を参照してください。
3. 作成したボリュームをインスタンスにアタッチします。詳細については、Amazon EC2 ユーザーガイドの [「インスタンスへの Amazon EBS ボリュームのアタッチ」](#) を参照してください。

起動テンプレートに記載されている EBS ボリュームがマウントされないのはなぜですか？

EBS ボリュームの `--launch-template` パラメータに起動テンプレート ID を指定すると、スクリプトは EBS ボリュームをアタッチしますが、ボリュームはマウントしません。起動した EC2 インスタンス用にスクリプトが作成した `MountEBSVolumes RunCommand` ドキュメントを実行することで、アタッチされた EBS ボリュームをマウントできます。

`--launch-template` パラメータを設定しない場合、スクリプトはテンプレートを作成し、Auto Scaling グループが新しい EC2 インスタンスを起動すると、Auto Scaling グループは自動的に EBS ボリュームをアタッチし、アタッチされたボリュームを Layer 設定で設定されたマウントポイントにマウントする `SetupAutomation` コマンドを実行します。

Chef レシピとマウント EBS ボリュームのログはどこにありますか？

OpsWorks は、`--command-logs-bucket` パラメータの値を指定して指定できる S3 バケットにログを配信します。デフォルトの S3 バケット名の形式は、`aws-opsworks-stacks-application-manager-logs-account-id` です。Chef レシピログは `ApplyChefRecipes` プレフィックスに保存されます。マウント EBS ボリュームログは `MountEBSVolumes` プレフィックスに保存されます。スタックから移行されるすべてのレイヤーは、同じ S3 バケットにログを配信します。

Note

- S3 バケットのライフサイクル設定には、30 日後にログを削除するルールが含まれています。30 日以上ログを保持する場合は、S3 バケットのライフサイクル設定のルールを更新する必要があります。
- 現在、OpsWorks のみが Chef setup と terminate recipe を記録します。

移行スクリプトのデバッグログはどこから入手できますか？

このスクリプトは、`aws-opsworks-stacks-transition-logs-account-id` という名前のバケットにデバッグログを格納します。デバッグログは、S3 バケットの `migration_script` フォルダで、移行した Layer の名前と一致するフォルダにあります。

移行スクリプトは CloudFormation テンプレートのバージョニングをサポートしていますか？

このスクリプトは、移行 CloudFormation するレイヤーまたはスタックの置き換えを作成するタイプの Systems Manager ドキュメントを生成します。同じパラメーターを使用してスクリプトを再実行すると、以前にエクスポートされたレイヤーテンプレートの新しいバージョンがエクスポートされます。テンプレートバージョンは、スクリプトログと同じ S3 バケットに保存されます。

複数の Layer を移行できますか？

スクリプトの `--layer-id` パラメーターは一つのレイヤーに渡されます。複数のレイヤーを移行するには、異なる `--layer-id` でスクリプトを再実行して渡します。

同じ OpsWorks スタックの一部であるレイヤーは、Application Manager の同じアプリケーションの下に一覧表示されます。

1. Systems Manager コンソール (<https://console.aws.amazon.com/systems-manager/>) を開きます。
2. ナビゲーションペインで、[Application Manager] を選択します。
3. [アプリケーション] セクションで、[カスタムアプリケーション] を選択します。
4. アプリケーションを選択します。アプリケーション名は `app-stack-name-first-six-characters-stack-id` で始まります。
5. アプリで始まる最上位の要素には、OpsWorks スタックに対応するすべてのコンポーネントが表示されます。これには、OpsWorks レイヤーに対応するコンポーネントが含まれます。
6. レイヤーに対応するコンポーネントを選択すると、そのレイヤーのリソースが表示されます。OpsWorks レイヤーを表すコンポーネントは、カスタムアプリケーションセクションからも個々のアプリケーションとして表示されます。

SecureString パラメータを作成するにはどうしたらよいですか？

Systems Manager を使用して SecureString パラメータを作成できます。Systems Manager SecureString パラメータの作成方法の詳細については、「[AWS Systems Manager ユーザーガイド](#)」の [SecureString 「パラメータの作成 \(AWS CLI\)」](#) または [「Systems Manager パラメータの作成 \(コンソール\)」](#) を参照してください。

SecureString、`--http-username`、`--http-password` または `--repo-private-key` パラメータの値としてパラメータを指定する必要があります。

新しい Auto Scaling グループのインスタンスを終了イベントから保護する方法を教えてください。

TRUE に `--enable-instance-protection` パラメータを設定し、`protected_instance` タグキーを終了イベントから保護したい各 EC2 インスタンスに追加することでインスタンスを保護できます。`--enable-instance-protection` パラメータを TRUE に設定して `protected_instance` タグキーを追加すると、スクリプトは新しい Auto Scaling グループにカスタム終了ポリシーを追加し、`ReplaceUnhealthy` プロセスを中断します。`protected_instance` タグキーが付いたインスタンスは、以下の終了イベントから保護されます。

- スケールインイベント
- インスタンスの更新
- リバランシング
- インスタンスの最大存続期間
- インスタンスの終了の一覧表示
- 異常のあるインスタンスの終了と交換

Note

保護するインスタンスには `protected_instance` タグキーを設定する必要があります。タグキーでは大文字と小文字が区別されます。そのタグキーを持つインスタンスは、タグ値に関係なく保護されます。

カスタム終了ポリシーの実行時間を短縮するには、`default_sample_size` 関数コード変数の値を更新することで、Lambda 関数が保護対象インスタンスのフィルタリングに使用するデフォルトのインスタンス数を増やすことができます。デフォルト値は 15 です。`default_sample_size` を増やすと、Lambda 関数に割り当てられるメモリを増やす必要が生じる場合があります。これにより、Lambda 関数のコストが増加します。AWS Lambda の料金については、「[AWS Lambda の料金](#)」を参照してください。

移行スクリプトではどのようなロードバランサーを利用できますか？

このスクリプトには三つのロードバランサーオプションがあります。

- (推奨) 新しい Application Load Balancer を作成します。デフォルトでは、スクリプトは新しい Application Load Balancer を作成します。`--lb-type` パラメータを ALB にも設定できま

す。Application Load Balancers 詳細については、「Elastic Load Balancing ユーザーガイド」の「[Application Load Balancer とは？](#)」を参照してください。

- Application Load Balancer がオプションでない場合は、`--lb-type` パラメータを Classic に設定して Classic Load Balancer を作成します。このオプションを選択すると、レイヤーに OpsWorks アタッチされている既存の Classic Load Balancer はアプリケーションとは別に保持されます。Application Load Balancer の詳細については、Elastic Load Balancing : Classic Load Balancer ユーザーガイドの「[Classic Load Balancer とは？](#)」を参照してください。
- `--lb-type` パラメータを None に設定することで、既存のロードバランサーをアタッチできます。

Important

AWS OpsWorks スタックレイヤー用に新しい Elastic Load Balancing ロードバランサーを作成することをお勧めします。既存の Elastic Load Balancing ロードバランサーを使用する場合は、まず、他の目的で使用されておらず、インスタンスがアタッチされていないことを確認する必要があります。ロードバランサーがレイヤーにアタッチされると、は既存のインスタンスをすべて OpsWorks 削除し、レイヤーのインスタンスのみを処理するようにロードバランサーを設定します。レイヤーにアタッチした後でロードバランサーの設定を Elastic Load Balancing コンソールまたは API を使用して変更することは技術的には可能ですが、そうしないでください。この変更は永続的ではないためです。

既存の OpsWorks レイヤーロードバランサーを Auto Scaling グループにアタッチするには

1. `--lb-type` パラメータを None に設定して移行スクリプトを実行します。値を None に設定すると、スクリプトはロードバランサーを複製したり作成したりしません。
2. スクリプトが CloudFormation スタックをデプロイしたら、Auto Scaling グループ MinMax と Desired capacity 値を更新し、アプリケーションをテストします。
3. スクリプトの出力に表示されている Link to the template を選択します。ターミナルを閉じた場合は、次の手順を実行してテンプレートにアクセスしてください。
 - a. Systems Manager コンソール (<https://console.aws.amazon.com/systems-manager/>) を開きます。
 - b. ナビゲーションペインで、[Application Manager] を選択します。
 - c. CloudFormation スタックを選択し、テンプレートライブラリ を選択します。
 - d. 「Owned by me」を選択し、テンプレートを探します。

- CloudFormation テンプレートから、アクションメニューから編集を選択します。
- テンプレートのApplicationAsgリソースセクション内の LabelBalancerNames CloudFormationプロパティを更新します。

```
ApplicationAsg:
  DependsOn: CustomTerminationLambdaPermission
  Properties:
    #(other properties in ApplicationAsg to remain unchanged)
    LoadBalancerNames:
      - load-balancer-name
    HealthCheckType: ELB
```

- Auto Scaling グループインスタンスのヘルスチェックにロードバランサーのヘルスチェックも使用させたい場合は、HealthCheckType 以下のセクションを削除してELBを入力してください。EC2 ヘルスチェックのみが必要な場合は、テンプレートを変更する必要はありません。
- 変更を保存します。保存すると、新しいデフォルトバージョンのテンプレートが作成されます。レイヤーのスクリプトを初めて実行し、コンソールに変更を初めて保存した場合、新しいバージョンは 2 です。
- [アクション] から [プロビジョニングスタック] を選択します。
- テンプレートのデフォルトバージョンを使用することを確認します。既存のスタックの選択が選択され、更新する CloudFormation スタックが選択されていることを確認してください。
- 「レビューとプロビジョニング」ページが表示されるまで、以降の各ページで [次へ] を選択します。レビューとプロビジョニングページで、がカスタム名で IAM リソースを作成する AWS CloudFormation 可能性があることと、選択したテンプレートの変更によって既存の AWS リソースが AWS CloudFormation 更新または削除される可能性があることの両方を選択します。
- [Provision stack] (スタックのプロビジョニング) を選択します。

更新をロールバックする必要がある場合は、次の手順を実行します。

- [アクション] を選択してから、スタックのプロビジョニング を選択します。
- [既存のバージョンの一つを選択] を選択し、次に以前のテンプレートバージョンを選択します。
- 既存のスタックを選択 を選択し、更新する CloudFormation スタックを選択します。

カスタムクックブック設定レシピは移行されますか？

Configure カスタムクックブックをセットアップイベント中に実行することはサポートされていません。このスクリプトは、カスタムクックブックの設定レシピを移行し、お客様に代わって Systems Manager 自動化ランブックを作成します。しかし、レシピを手動で実行する必要があります。

設定レシピを実行するには、次のステップに従います。

1. Systems Manager コンソール (<https://console.aws.amazon.com/systems-manager/>) を開きます。
2. ナビゲーションペインで、[Application Manager] を選択します。
3. [アプリケーション] セクションで、[カスタムアプリケーション] を選択します。
4. アプリケーションを選択します。アプリケーション名は `app-stack-name` で始まります。
5. 「リソース」を選択し、次に「設定ランブック」を選択します。
6. [オートメーションを実行] を選択します。
7. 設定レシピを実行するインスタンス ID を選択し、実行 を選択します。

新しく作成したインスタンスで、デプロイ、アンデプロイのレシピを実行できますか？

このスクリプトでは、Layer の設定に応じて 3 つの自動化ランブックを作成できます。

- セットアップ
- 構成する
- 終了

このスクリプトでは、AWS-ApplyChefRecipes Run Command ドキュメントの入力値を含む次の Systems Manager パラメータを作成することもできます。

- セットアップ
- デプロイ
- 構成する
- Undeploy
- 終了

スケールアウトイベントが発生すると、セットアップオートメーションランブックが自動的に実行されます。これには、元の OpsWorks レイヤーからのカスタムクックブックレシピのセットアップとデプロイが含まれます。スケールインイベントが発生すると、ターミナルのオートメーションランブックが自動的に実行されます。終了オートメーションランブックには、元の OpsWorks レイヤーからのシャットダウンレシピが含まれています。

レシピを手動で実行、アンデプロイ、または設定する場合は、次の手順を実行します。

1. Systems Manager コンソール (<https://console.aws.amazon.com/systems-manager/>) を開きます。
2. ナビゲーションペインで、[Application Manager] を選択します。
3. [アプリケーション] セクションで、[カスタムアプリケーション] を選択します。
4. アプリケーションを選択します。アプリケーション名は `app-stack-name-first-six-characters-stack-id` で始まります。Application Manager の [概要] タブが開きます。
5. [リソース] を選択し、次に自動化ランブックの設定を選択します。
6. [オートメーションを実行] を選択します。
7. `applyChefRecipesPropertiesParameter` オートメーションランブックの入力パラメータについては、正しい Systems Manager パラメータを参照してください。Systems Manager パラメータ名は `event` の値が `Configure` という `/ApplyChefRecipes-Preset/OpsWorks-stack-name-OpsWorks-layer-name-first-six-characters-stack-id/event`、または、`Deploy`、実行したいレシピに応じて `Undeploy` という形式に従います。
8. レシピを実行するインスタンス ID を選択し、実行 を選択します。

Auto Scaling グループが対象とするサブネットを変更できますか？

デフォルトでは、Auto Scaling グループは OpsWorks スタック VPC 内のすべてのサブネットにまたがります。スパンのサブネットを更新するには、次の手順に従います。

1. スクリプトの出力に表示されている `Link to the template` を選択します。ターミナルを閉じた場合は、次の手順を実行してテンプレートにアクセスしてください。
 - a. Systems Manager コンソール (<https://console.aws.amazon.com/systems-manager/>) を開きます。
 - b. ナビゲーションペインで、[Application Manager] を選択します。
 - c. CloudFormation スタックを選択し、テンプレートライブラリ を選択します。
 - d. 「Owned by me」を選択し、テンプレートを探します。

2. 「アクション」から「プロビジョニングスタック」を選択します。
3. デフォルトのテンプレートを使用することを確認します。既存のスタックを選択し、更新する CloudFormation スタックを選択します。

Note

--provision-application パラメータを `true` に設定してスクリプトを実行した場合は `FALSE`、新しい CloudFormation スタックを作成する必要があります。

4. SubnetIDs パラメータには、Auto Scaling グループが対象とするサブネット ID をカンマで区切ったリストを指定します。
5. 「レビューとプロビジョニング」ページが表示されるまで「次へ」を選択します。
6. レビューとプロビジョニングページで、カスタム名で IAM リソースを作成する AWS CloudFormation 可能性があることを承認し、選択したテンプレートの変更によって既存の AWS リソースが AWS CloudFormation 更新または削除される可能性があることを理解します。
7. [Provision stack] (スタックのプロビジョニング) を選択します。

トラブルシューティング

このセクションでは、一般的な問題と、それらの問題について推奨される解決方法を示します。

トピック

- [入力されたプリンシパルは無効ではない](#)
- [Auto Scaling グループで保護されたインスタンスが有効になっている場合、CloudFormation スタックを削除できません](#)
- [既存の S3 バケットとプレフィックスを指定するとアクセス拒否エラーが発生する](#)

入力されたプリンシパルは無効ではない

問題: 指定したプリンシパルが有効ではないというエラーメッセージが表示されます。

原因: これは、Auto Scaling グループにサービスロールがないために発生します。

解決策: エラーが発生したリージョンに Auto Scaling グループを作成します。Auto Scaling グループを作成すると、カスタム終了ポリシーに必要なサービスリンクロールが作成されます。

Auto Scaling グループで保護されたインスタンスが有効になっている場合、CloudFormation スタックを削除できません

問題: `--enable-instance-protection` パラメータが TRUE に設定されており、Auto Scaling グループの EC2 インスタンスの一部が `protected_instance` タグキーで保護されているため、AWS CloudFormation スタックが完全に削除されません。

原因: EC2 インスタンスには、終了イベントから保護する `protected_instance` タグキーがあります。

解決策: EC2 インスタンスから `protected_instance` タグキーを削除します。これにより、Auto Scaling グループはスケールダウンできます。Auto Scaling グループがスケールダウンしたら、AWS CloudFormation スタックを削除できます。

既存の S3 バケットとプレフィックスを指定するとアクセス拒否エラーが発生する

問題: 既存の S3 バケットとプレフィックスを指定すると `AccessDenied` エラーになる。

原因: S3 バケットポリシーには、ロードバランサーのログをバケットに配信するのに必要な権限が付与されていません。

解決策: S3 バケットポリシーを更新して、スクリプトがロードバランサーのアクセスログをバケットに配信できるようにします。バケットポリシーを更新する方法の詳細については、「[伸縮性ロードバランサー：Application Load Balancer ユーザーガイド](#)」の「[Application Load Balancerのアクセスログを有効にする](#)」を参照してください。

AWS OpsWorks Stacks Detach in Place ツールの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。

このセクションでは、AWS OpsWorks Stacks Detach in Place ツールを使用して OpsWorks スタックサービスから OpsWorks インスタンスをデタッチする方法について説明します。

デタッチしたインスタンスはに残りますが AWS アカウント、 を使用して管理することはできなくなります OpsWorks。代わりに、Amazon EC2 AWS Systems Manager、または EC2 互換のアプローチを使用して、インスタンスを設定および管理します。

大まかに言うと、デタッチプロセスには次のステップが含まれます。


1. このツールは、検証チェックを実行して、リソースをデタッチする準備ができていることを確認します。
2. このツールは、OpsWorks スタックからカスタム JSON をエクスポートし、オブジェクトとして Amazon S3 に保存します。
3. このツールは、各 OpsWorks スタックのライフサイクルイベントを表す Systems Manager Automation ドキュメントを作成します。
4. このツールは、デタッチされているすべてのインスタンスの AWS Service Catalog AppRegistry カタログを作成し、レイヤーから Elastic Load Balancing (ELB) ロードバランサーをデタッチします OpsWorks。
5. 最後に、このツールは Amazon Relational Database Service (Amazon RDS) インスタンスなどの他のリソースをデタッチおよび登録解除します。

プロセスの仕組み

Detach In Place ツールは、レイヤーのデタッチに進む前にインスタンスをチェックして設定するための一連のステップをガイドする、以下の 3 つのコマンドとウィザードのようなエクスペリエンスを提供します。

コマンド	説明
<code>handle-prerequisites</code>	<p>このコマンドは、レイヤー内のすべてのインスタンスがデタッチの対象であるかどうかを分析し、前提条件を解決します。インスタンスは正常な状態である必要があり OpsWorks、時間や負荷ベースのオートスケーラーを持つことはできません。また、最新の OpsWorks エージェントバージョンがインストールされている必要があります。</p> <p>さらに、コマンドは、すべてのインスタンスに SSM エージェントをサポートするために必</p>

コマンド	説明
	<p>要なアクセス許可があるかどうか、および最新の SSM エージェントバージョンがインストールされているかどうかを確認します。コマンドは、SSM エージェントが存在しない場合はインストールし、最新バージョンを使用していない場合は SSM エージェントを更新します。コマンドは、必要なアクセス許可も追加します。</p>
detach	<p>このコマンドは、指定されたレイヤーのすべての OpsWorks インスタンスをデタッチします。</p> <p>まず、コマンドは前提条件チェックを実行して、レイヤーがデタッチの対象となることを確認します。前提条件を解決しない場合は、強制的にデタッチするオプションが与えられます。</p> <p>次に、コマンドは、OpsWorks タグ付け APIs またはレイヤーとスタックからのタグの伝達を通じてインスタンスに追加されたすべてのタグが保持されることを示します。デタッチが完了したら、関連する EC2 APIs を使用してこれらのタグのいずれかを削除できます。</p> <p>次に、コマンドは Chef 関連の設定を SSM パラメータにエクスポートするかどうかをチェックします。</p> <p>Classic Load Balancer がレイヤーにアタッチされている場合、コマンドはダウンタイムを防ぐためにロードバランサーをデタッチできるかどうかを尋ねます。</p>

コマンド	説明
cleanup	<p>このコマンドは、アカウント OpsWorks からのすべてのエンティティを削除します。これにより、インスタンスが終了し、すべてのスタックが削除されます。これは、アカウントをクリーンアップする最後のステップとして不要になったリソースに使用する必要があります。</p> <div data-bbox="829 541 1508 905" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px;"><p> Note</p><p>cleanup コマンドを実行する前に、新しいセットアップを数日間実行することをお勧めします。これにより、スタックから必要な設定を必要に応じて簡単に利用できるようになります。</p></div>

制限事項

Detach In Place ツールの主な目的は、OpsWorks スタックインスタンスを安全にデタッチすることです。このセクションでは、ツールの制限事項を要約します。

- Windows SSM Agent – SSM Agent がインスタンスにインストールされていない場合は、手動でインストールする必要があります。エージェントを最新バージョンに更新しない場合も同様です。
- Time/Load Auto Scaling インスタンス – デタッチツールは Auto Scaling が有効になっているインスタンスをサポートしていません。デタッチするインスタンスで Auto Scaling を無効にする必要があります。
- アクセス許可 – デタッチツールは OpsWorks、コンソールのアクセス許可ページで指定された IAM エンティティを作成または生成しません。
- アプリケーション – デタッチツールは、の外部でアプリケーションを作成または生成しません OpsWorks。

開始

ステップ 1: 前提条件が満たされていることを確認する

Detach In Place ツールの 3 つのコマンドはすべて Python スクリプトで、ローカル、EC2 インスタンス、または [AWS CloudShell](#) を使用して実行できます。

AWS CloudShell はブラウザベースのシェルで、一般的なツール (AWS CLI や Python など) がプリインストールされている選択した AWS リージョン。AWS CloudShell comes のリソースに AWS コマンドラインでアクセスできます。を使用する場合 AWS CloudShell、コンソールへのサインインに使用するのと同じ認証情報を使用します。

このチュートリアルでは、を使用していることを前提としています AWS CloudShell。

ステップ 2: スクリプトをダウンロードする

1. 次のコマンドを実行して、移行スクリプトとすべての関連ファイルを含む zip ファイルをダウンロードします。

```
aws s3api get-object \  
--bucket detach-in-place-bucket-prod-us-east-1 \  
--key detach_in_place_script.zip detach_in_place_script.zip
```

2. 次のコマンドを実行して、ファイルを解凍します。

```
unzip detach_in_place_script.zip
```

ファイルを解凍すると、次のファイルを使用できます。

- README.md
- LICENSE
- NOTICE
- requirements.txt
- TODO.py

3. 必要に応じて、次のコマンド `pipenv` を実行して をインストールします。

```
pip install pipenv
```

ステップ 3: スクリプトを実行する

まず、次のコマンドを実行してスクリプトを実行できるように環境を設定します。

```
pipenv install -r requirements.txt
pipenv shell
```

次に、スクリプトパラメータを確認します。

Command	パラメータ	説明	タイプ	必須	デフォルト
handle-prepare-requisites	--layer-id	デタッチするレイヤーの ID。	文字列	あり	-
	--region	OpsWorks スタックのリージョン。OpsWorks スタックリージョンと API エンドポイントリージョンが異なる場合は、スタックリージョンを使用します。これは、OpsWorks スタックの他のリソース部分 (EC2 インスタンスやサブネットなど) と同じリージョンです。	文字列	なし	us-east-1
detach	--layer-id	デタッチするレイヤーの ID。	文字列	あり	-
	--batch-size	レイヤーからデタッチするインスタンスの数 (例: 5)。	文字列	なし	-
	--region	OpsWorks スタックのリージョン。OpsWorks スタックリージョンと API エンドポイントリージョンが異なる場合は、スタック	文字列	なし	us-east-1

Command	パラメータ	説明	タイプ	必須	デフォルト
		クリージョンを使用します。これは、OpsWorks スタックの他のリソース部分 (EC2 インスタンスやサブネットなど) と同じリージョンです。			
cleanup	--stack-id	削除するスタックの ID。	文字列	なし	相互に排他的に、レイヤー ID またはスタック ID のいずれかを指定する必要があります
	--layer-id	削除するレイヤーの ID	文字列	なし	
	--region	OpsWorks スタックのリージョン。OpsWorks スタックリージョンと API エンドポイントリージョンが異なる場合は、スタッククリージョンを使用します。これは、OpsWorks スタックの他のリソース部分 (EC2 インスタンスやサブネットなど) と同じリージョンです。	文字列	なし	us-east-1

、 handle-prerequisites コマンドで使用可能なオプションを確認するには detach、次のように --help オプションを指定 cleanup して コマンドを実行します。

```
python3 layer_detacher.py detach --help
```



```
python3 layer_detacher.py handle-prerequisites --help
python3 layer_detacher.py cleanup --help
```

これで、開始する準備が整いました。次の例は、さまざまなユースケースでコマンドを実行する方法を示しています。

例:

- [例 1: レイヤーがすべての前提条件を満たし、デタッチの対象となるかどうかを確認します。](#)
- [例 2: レイヤーのすべてのインスタスをデタッチする](#)
- [例 3: レイヤーのすべてのインスタスをバッチでデタッチする](#)
- [例 4: レイヤーのすべてのリソースをクリーンアップし、レイヤーを削除する](#)
- [例 5: スタックのすべてのリソースをクリーンアップしてスタックを削除する](#)

例 1: レイヤーがすべての前提条件を満たし、デタッチの対象となるかどうかを確認します。

次のコマンドは、レイヤー OpsWorks (およびそれに含まれるインスタス) に関する情報を読み取り、以下の前提条件が満たされているかどうかを確認します。

- すべてのインスタスはオンラインです。
- Load/Time Auto Scaling インスタスはありません。
- すべてのインスタスに最新の OpsWorks エージェントがあります。
- すべてのインスタスには、最新の SSM エージェントがインストールされ、設定されています。
- すべてのインスタスには SSH キーペアがあります。
- すべてのインスタスは 1 つのレイヤーに属します。

```
python3 layer_detacher.py handle-prerequisites \  
--layer-id opsworks-layer-id \  
--region opsworks-stack-region
```

例 2: レイヤーのすべてのインスタスをデタッチする

次のコマンドは、レイヤーのすべてのインスタスを反復処理し、インスタスが前提条件を満たしているかどうかを確認し、前提条件を満たすすべてのインスタスを並行してデタッチしようとしています。1 つ以上の前提条件が満たされない場合、コマンドは残りの非準拠インスタスに強制デタッチオプションを提供します。

インスタンスをデタッチする前に、コマンドは次の操作を行います。

1. カスタム JSON を保存し、S3 にアップロードします。
2. レイヤーの OpsWorks ライフサイクルイベントごとに SSM Automation ドキュメントを作成し、オートメーションドキュメントの実行ログを S3 にアップロードします。
3. デタッチされるすべてのインスタンスの AppRegistry アプリケーションを作成します。アプリケーションには、デタッチされたすべてのインスタンスとリソースを保持するリソースグループが関連付けられています。リソースには、ライフサイクルイベントとカスタム Chef レシピに関する情報を保持する SSM Automation ドキュメントと SSM パラメータが含まれます。
4. Classic Load Balancer が存在する場合は、レイヤーからデタッチします。

このコマンドは OpsWorks リソースのみを変更します。EC2 インスタンスのステータスは変わりません。

```
python3 layer_detacher.py detach \  
--layer-id opsworks-layer-id \  
--region opsworks-stack-region
```

例 3: レイヤーのすべてのインスタンスをバッチでデタッチする

次のコマンドは、[前の例](#)と同じ操作を行います。唯一の違いは、インスタンスをバッチでデタッチすることです。

このコマンドは OpsWorks リソースのみを変更します。EC2 インスタンスのステータスは変わりません。

```
python3 layer_detacher.py detach \  
--layer-id opsworks-layer-id \  
--region opsworks-stack-region \  
--batch-size 5
```

例 4: レイヤーのすべてのリソースをクリーンアップし、レイヤーを削除する

次のコマンドは、レイヤーのすべてのリソースを反復処理して削除します。より詳細には、および EC2 内のすべてのインスタンスを停止および削除 OpsWorks し、ロードバランサーをデタッチして、Amazon RDS インスタンス、Elastic IPs、ボリュームの登録を解除します。リソースをクリーンアップすると、レイヤーが削除されます。

このコマンドは、OpsWorks リソースと EC2 インスタンスを削除します。EC2 インスタンスをそのまま使用する場合は、detach コマンドを使用する前に cleanup コマンドを使用します。これにより、cleanup コマンドは残りのリソースをすべて削除します。

```
python3 layer_detacher.py cleanup \  
--layer-id opsworks-layer-id \  
--region opsworks-stack-region
```

例 5: スタックのすべてのリソースをクリーンアップしてスタックを削除する

次のコマンドは、すべてのレイヤーを反復処理し、各レイヤーのリソースを反復処理します。レイヤーごとに、コマンドは および EC2 内のすべてのインスタンスを停止および削除 OpsWorks し、ロードバランサーをデタッチし、Amazon RDS インスタンス、Elastic IPs ポリユームを登録解除します。次に、コマンドはレイヤーを削除します。このスタックに属するすべてのレイヤーで同じプロセスが実行されます。最後に、すべてのレイヤーが削除されると、スタックは削除されます。

このコマンドは、OpsWorks リソースと EC2 インスタンスを削除します。EC2 インスタンスをそのまま使用する場合は、detach コマンドを使用する前に cleanup コマンドを使用します。これにより、cleanup コマンドは残りのリソースをすべて削除します。

```
python3 layer_detacher.py cleanup \  
--stack-id opsworks-stack-id \  
--region opsworks-stack-region
```

ステップ 4: からデタッチした後もリソースを運用し続ける OpsWorks

detach コマンドを実行すると、ツールはデタッチされたレイヤーに対応する新しい AWS Service Catalog AppRegistry アプリケーションを作成します。アプリケーション名は の形式に従います *layer-name---layer-id*。また、OpsWorksLayerId タグを追加して、デタッチされたレイヤーに一致するアプリケーションを一意に識別します。

このアプリケーションに新しい AWS リソース (新しい EC2 インスタンスなど) を追加するには、次のいずれかを実行します。

1. リソースにアプリケーションの一意のアプリケーションタグを AppRegistry タグ付けします。

タグキー: awsApplication

値: arn:aws:resource-groups:*region*:*account-id*:group/*application-name/application-id*>

2. [associate-resource](#) コマンドを実行します。

さらに、AppRegistry アプリケーションごとにリソースグループが作成されます。リソースグループには、次のタグが含まれています。

タグキー	値
EnableAWSServiceCatalogAppRegistry	TRUE
aws:servicecatalog:applicationName	<i>application-name</i>
aws:servicecatalog:applicationId	<i>application-id</i>
aws:servicecatalog:applicationArn	arn:aws:servicecatalog: <i>region</i> : <i>account-id</i> :/applications/ <i>application-id</i>

デタッチ後のタスクの実行

次の表に、デタッチ後にタスクを実行する方法を示します。

タスク	説明
ライフサイクルイベントの実行	<p>detach コマンドを実行し、オプションを選択した場合、スクリプトは 5 OpsWorks ライフサイクルイベントに一致する 5 つのオートメーションドキュメントを作成します。</p> <p>各オートメーションドキュメントの名前は、次の形式に従います: <i>layer-id_lifecycle-event_automation_document</i> 。</p> <p>Systems Manager OpsWorks の動作をシミュレートするには、プロビジョニング、EC2 インスタンスの終了、またはレシピのデプロイ/</p>

タスク	説明
	削除時に、オートメーション実行を手動でトリガーする必要があります。
カスタム JSON の更新	<p>スタックとレイヤーのカスタム JSON は、デタッチ時に指定された S3 バケットに保存されるか、新しい S3 バケットに作成されます。</p> <p>JSON ファイルに保存されるファイル名は次のとおりです。</p> <ul style="list-style-type: none">• layercustomjson.json• stackcustomjson.json
ライフサイクルイベントの実行リストの変更	<p>各ライフサイクルイベントの実行リストは、対応するオートメーションドキュメントで定義されます。実行リストを変更するには、AppRegistry アプリケーションでオートメーションドキュメントを検索し、RunList パラメータを変更します。</p> <p>自動化ドキュメントがトリガーする は 同じソースをサポートしているため AWS-Apply ChefRecipes 、レシピとクックブックを更新するプロセスは変更されません OpsWorks。</p>
自動ヒーリング/自動スケーリングの管理	<p>インスタンスをデタッチすると、OpsWorks エージェントはアンインストールします。エージェントがないと、OpsWorks 異常なインスタンスを自動的に修復または置き換えることも、フリートを自動スケーリングすることもできません。自動スケーリングを続行し、失敗したインスタンスを置き換えるには、Amazon EC2 Auto Scaling グループを作成します。Amazon EC2 が置き換えが必要な異常なインスタンスを検出すると、グループは新しいインスタンスを起動して希望する容量を維持します。</p>

タスク	説明
Load Balancer の管理	レイヤーが Classic Load Balancer を使用している場合、detach コマンドはインスタンスの登録を解除する前にそのレイヤーをデタッチします。これは、デタッチの過程ですべての ELB インスタンスの関連付けが Amazon EC2 に保持され、ダウンタイムがゼロになるように行われます。プロセスが完了すると、EC2 で ELB を管理できるようになります。
インスタンスへの接続	<p>handle-prerequisites または detach コマンドを実行すると、次の 2 つのチェックが行われます。</p> <ul style="list-style-type: none">• SSM エージェントのバージョンとアクセス許可• SSH キー <p>コマンドでは、SSM エージェントを更新し、必要なアクセス許可を追加して、Session Manager を使用してインスタンスに接続することもできます。SSH キーが存在する場合は、インスタンスに SSH 接続することもできます。</p>

Systems Manager Application Manager インスタンス タブの使用

デタッチ後、Application Manager インスタンス [タブでインスタンス](#) を表示および管理できます。

インスタンスタブには、ステータス、ヘルスステータス、最後のコマンドステータスなど、アプリケーションの EC2 インスタンスに関する集計情報が表示されます。このタブを使用すると、コマンド履歴、アラーム状態、Systems Manager エージェントの状態など、個々のインスタンスに関する詳細情報を表示できます。インスタンスタブには、Chef レシピの適用、インスタンスの開始または停止、Auto Scaling グループへのインスタンスの追加や削除など、さまざまなアクションも用意されています。

AWS OpsWorks スタックの開始方法

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックには、さまざまなカスタマイズ可能なコンポーネントが用意されており、組み合わせて特定の目的を満たすスタックを作成できます。新規ユーザーにとっての課題は、これらのコンポーネントをどのように実際のスタックに組み込み、効果的に管理するかを理解することです。ここで最初からご説明します。

目的が	このウォークスルーの完了:
サンプルスタックをできるだけ迅速に作成する	使用開始: サンプル
Linux ベースのスタックを試す	入門ガイド: Linux
Windows ベースのスタックを試す	入門ガイド: Windows
独自の Chef クックブックを作成する方法を学習する	使用開始: クックブック

Amazon EC2 インスタンス、または独自のハードウェアで実行されているオンプレミスインスタンスなどの既存のコンピューティングリソースがある場合は、スタックで作成したインスタンスとともにスタック [に組み込む](#) ことができます AWS OpsWorks。その後、AWS OpsWorks スタックを使用して、作成方法に関係なく、関連するすべてのインスタンスをグループとして管理できます。

リージョンのサポート

AWS OpsWorks スタックにはグローバルにアクセスできます。また、インスタンスをグローバルに作成および管理することもできます。ユーザーは、(米国西部) および中国 AWS GovCloud (北京) AWS リージョンを除く任意のリージョンで起動するように AWS OpsWorks スタックインスタンス

を設定できます。AWS OpsWorks スタックを使用するには、インスタスが次のいずれかの AWS OpsWorks スタックインスタンスサービス API エンドポイントに接続できる必要があります。

リソースは、そのリソースを作成したリージョンでのみ管理できます。あるリージョンのエンドポイントで作成されたリソースは、他のリージョンのエンドポイントからは使用できず、他のリージョンのエンドポイントにクローニングすることもできません。インスタンスを起動できるリージョンは以下の通りです。

- 米国東部 (オハイオ) リージョン
- 米国東部(バージニア州北部) リージョン
- 米国西部 (オレゴン) リージョン
- US West (N. California) リージョン
- カナダ (中部) リージョン (API のみ、AWS Management Consoleで作成されたスタックは使用できません。)
- アジアパシフィック (ムンバイ) リージョン
- アジアパシフィック (シンガポール) リージョン
- アジアパシフィック (シドニー) リージョン
- アジアパシフィック (東京) リージョン
- アジアパシフィック (ソウル) リージョン
- 欧州 (フランクフルト) リージョン
- 欧州 (アイルランド) リージョン
- 欧州 (ロンドン) リージョン
- 欧州 (パリ) リージョン
- 南米 (サンパウロ) リージョン

サンプルスタックの使用開始

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

このチュートリアルでは、 の使用方法を示します。AWS OpsWorks マウスを数回クリックするだけで、コードを記述せずにサンプル Node.js アプリケーション環境をすばやく作成するためのスタック。完了したら、Chef 12 を実行している Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、Node.js HTTPサーバー、および Twitter を操作してウェブページにコメントを残すために使用できるウェブアプリがあります。

Note

このチュートリアルを完了すると、c3.large タイプのインスタンスが自動的に作成されるため、このチュートリアルや のサンプルスタック作成ツールを で使用することはできません。AWS OpsWorks のスタック [AWS 無料利用枠](#)。でサンプルスタック作成ツールを使用すると t2.medium インスタンスVPCが作成されますが、VPCsは現在 では使用できません。[AWS 無料利用枠](#)。

ステップ 1: 前提条件を完了する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、にお問い合わせてください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

ウォークスルーを開始する前に、次のセットアップ手順を完了する必要があります。これらのセットアップ手順には、へのサインアップが含まれます。AWS アカウント、管理ユーザーの作成、へのアクセス許可の割り当て AWS OpsWorks スタック。

トピック

- [にサインアップする AWS アカウント](#)
- [管理アクセスを持つユーザーを作成する](#)
- [サービスアクセス権限を割り当てます。](#)

にサインアップする AWS アカウント

をお持ちでない場合 AWS アカウントで、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. <https://portal.aws.amazon.com/billing/サインアップ> を開きます。
2. オンラインの手順に従います。

サインアップ手順の一環として、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

にサインアップするとき AWS アカウント、AWS アカウントのルートユーザー が作成されます。ルートユーザーはすべての にアクセスできます AWS のサービス アカウントの および リソース。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用して [ルートユーザーアクセスが必要なタスク](#) を実行してください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。 <https://aws.amazon.com/> に移動し、マイアカウント を選択すると、いつでも現在のアカウントアクティビティを表示し、アカウントを管理できます。

管理アクセスを持つユーザーを作成する

にサインアップした後 AWS アカウント、 をセキュリティで保護する AWS アカウントのルートユーザー、有効化 AWS IAM Identity Center、および 管理ユーザーを作成して、日常的なタスクにルートユーザーを使用しないようにします。

のセキュリティ保護 AWS アカウントのルートユーザー

1. [にサインインします。AWS Management Console](#) ルートユーザーを選択し、AWS アカウント E メールアドレス。次のページでパスワードを入力します。

ルートユーザーを使用してサインインする方法については、「」の「[ルートユーザーとしてサインイン](#)する」を参照してください。AWS サインイン ユーザーガイド。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

手順については、「」の仮想MFAデバイスの[有効化](#)を参照してください。 [AWS アカウントIAM ユーザーガイドのルートユーザー \(コンソール\)](#)。

管理アクセスを持つユーザーを作成する

1. IAM Identity Center を有効にします。

手順については、[「の有効化」を参照してください。AWS IAM Identity Center \(\)AWS IAM Identity Center ユーザーガイド](#)。

2. IAM Identity Center で、ユーザーに管理アクセス権を付与します。

の使用に関するチュートリアル IAM アイデンティティセンターディレクトリ ID ソースとして、[「デフォルトを使用してユーザーアクセスを設定する」を参照してください。IAM アイデンティティセンターディレクトリ \(\)AWS IAM Identity Center ユーザーガイド](#)。

管理アクセス権を持つユーザーとしてサインインする

- IAM Identity Center ユーザーでサインインするには、IAM Identity Center ユーザーの作成時に E メールアドレスに URL 送信されたサインインを使用します。

IAM Identity Center ユーザーを使用してサインインする方法については、[「へのサインイン」を参照してください。AWS のアクセスポータル AWS サインイン ユーザーガイド](#)。

追加のユーザーにアクセス権を割り当てる

1. IAM Identity Center で、最小特権のアクセス許可を適用するベストプラクティスに従うアクセス許可セットを作成します。

手順については、[「」の「アクセス許可セットの作成」を参照してください。AWS IAM Identity Center ユーザーガイド](#)。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

手順については、[「」の「グループの追加」を参照してください。AWS IAM Identity Center ユーザーガイド](#)。

サービスアクセス権限を割り当てます。

へのアクセスを有効にする AWS OpsWorks スタックサービス (および AWS OpsWorks スタックは、ロールまたはユーザーに `AWSOpsWorks_FullAccess` および `アクセス AmazonS3FullAccess` 許可を追加することで) に依存します。

アクセス許可の追加の詳細については、[IAM「ID アクセス許可の追加 \(コンソール\)」](#)を参照してください。

これですべてのセットアップステップが完了したので、[このウォークスルーを開始](#)できます。

ステップ 2: スタックを作成する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

このステップでは、AWS OpsWorks スタックを作成するためのスタックコンソール。スタックは、インスタンス (Amazon EC2 インスタンスなど) のコレクションであり、関連する AWS 共通の目的を持ち、一緒に管理する リソース。(詳細については、[スタック](#) を参照してください)。このウォークスルーのインスタンスは 1 つのみです。

このステップを開始する前に、[前提条件](#)を完了してください。

スタックを作成するには

1. <https://console.aws.amazon.com/opsworks/> にサインインする AWS Management Console を開き、AWS OpsWorks の コンソール
2. 該当する場合、次のいずれかを実行します。
 - へようこそ AWS OpsWorks スタックページが表示され、最初のスタックを追加するまたは最初のスタックを追加するを選択します。AWS OpsWorks スタックスタック (両方の選択肢が同じことを行います)。[Add stack] ページが表示されます。
 - OpsWorks ダッシュボードページが表示されたら、スタックの追加 を選択します。[Add stack] ページが表示されます。
3. [Add stack] ページが表示されたら、すでに選択されていない場合は [Sample stack] を選択します。
4. [Operating system type] (オペレーションシステムタイプ) で [Linux] が既に選択されている状態で、[Create stack] (スタックの作成) を選択します：

Add stack

Which type of stack do you want to create?

Sample stack
Explore AWS OpsWorks with a sample Node.js app

Chef 12 stack
Bring your own cookbooks and use community cookbooks

Chef 11 stack
Use built-in cookbooks for applications and deployments

Create a Chef 12 sample stack with a Node.js app
A Node.js app will be set up to help you explore the features and configuration options of AWS OpsWorks, for example: layers and lifecycle events. [Learn more.](#)

Operating system type Linux Windows

Cancel **Create stack**

5. AWS OpsWorks スタックは、My Sample Stack (Linux) という名前のスタックを作成します。AWS OpsWorks スタックは、アプリケーションをスタックにデプロイするために必要なすべてのコンポーネントも追加します。
 - レイヤー。これはインスタンスの設計図です。インスタンス設定、リソース、インストールされているパッケージ、セキュリティグループなどを指定します。(詳しくは、[レイヤー](#)を参照してください)。レイヤーの名前は Node.js App Server となります。
 - この場合、Amazon Linux 2 EC2インスタンスであるインスタンス。(インスタンスの詳細については、「[インスタンス](#)」を参照してください)。インスタンスのホスト名は nodejs-server1 です。
 - アプリケーション。これはインスタンスで実行するコードです (アプリケーションの詳細については、「[アプリケーション](#)」を参照してください)。アプリケーションの名前は Node.js Sample App です。
6. 後 AWS OpsWorks スタックはスタックを作成し、サンプルスタックを探索を選択して、My Sample Stack (Linux) ページを表示します (このチュートリアルを複数回完了すると、My Sample Stack (Linux) の後に 2 や 3 などのシーケンシャル番号が表示される場合があります)。

Setting up a sample stack

- ✓ 1. Creating a stack named "My Sample Stack (Linux)"
- ✓ 2. Setting the Chef cookbook repository of the stack
- ✓ 3. Creating a layer named "Node.js App Server" in the stack
- ✓ 4. Assigning a recipe to the deploy lifecycle event in the layer
- ✓ 5. Adding an instance to the layer

Cancel

Explore the sample stack

[次のステップ](#)では、インスタンスを開始し、インスタンスにアプリケーションをデプロイします。

ステップ 3: インスタンスを起動し、アプリケーションをデプロイする

⚠ Important

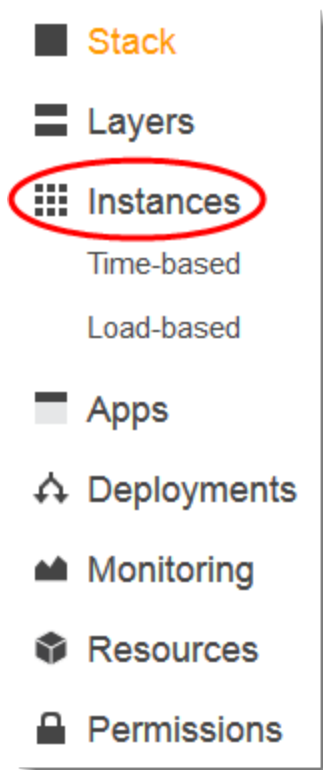
- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

これでインスタンスとアプリケーションが用意されたので、インスタンスを開始し、アプリケーションをインスタンスにデプロイします。

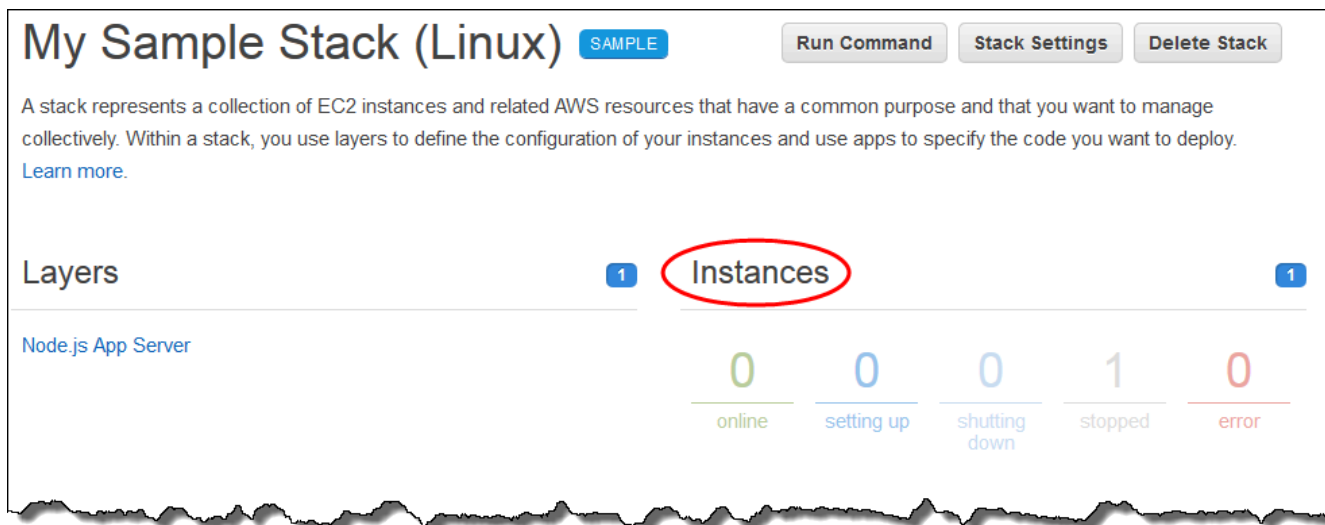
インスタンスを起動し、アプリケーションをデプロイするには

1. 次のいずれかを行います。

- サービスのナビゲーションペインで、[Instances] を選択します。



- [My Sample Stack (Linux)] ページで、[Instances] を選択します。



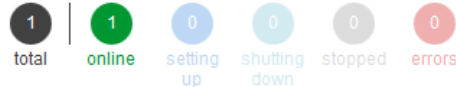
- [Instances] ページで、[Node.js App Server]、[nodejs-server1] に [start] を選択します。

Node.js App Server

Search for instances in this layer by name, status, size, type, AZ or IP						
Hostname	Status	Size	Type	AZ	Public IP	Actions
nodejs-server1	stopped	c3.large	24/7	us-east-1a	-	▶ start delete

- [online] の円が明るい緑色に成るまで先の手順に進まないでください。(エラーメッセージが表示される場合は、[デバッグとトラブルシューティングのガイド](#) を参照してください)。
- インスタンスの設定中、AWS OpsWorks スタックは、アプリケーションをインスタンスにデプロイします。
- 結果は、続行する前に次のスクリーンショットに類似する必要があります (エラーメッセージが表示される場合は、「[デバッグとトラブルシューティングのガイド](#)」を参照してください)。

Instances ⓘ


[Stop All Instances](#)

An instance represents a server. It can belong to one or more layers, that define the instance's settings, resources, installed packages, profiles and security groups. When you start the instance, OpsWorks uses the associated layer's blueprint to create and configure a corresponding EC2 instance. [Learn more.](#)

Node.js App Server

Search for instances in this layer by name, status, size, type, AZ or IP						
Hostname	Status	Size	Type	AZ	Public IP	Actions
nodejs-server1	online	t2.medium	24/7	us-west-2a		■ stop ssh

[+ Instance](#)

これで、インスタンスと、インスタンスにデプロイされたアプリケーションが用意されました。

[次のステップ](#)では、インスタンスでアプリケーションをテストします。

ステップ 4: インスタンスにデプロイされたアプリケーションをテストする

⚠ Important

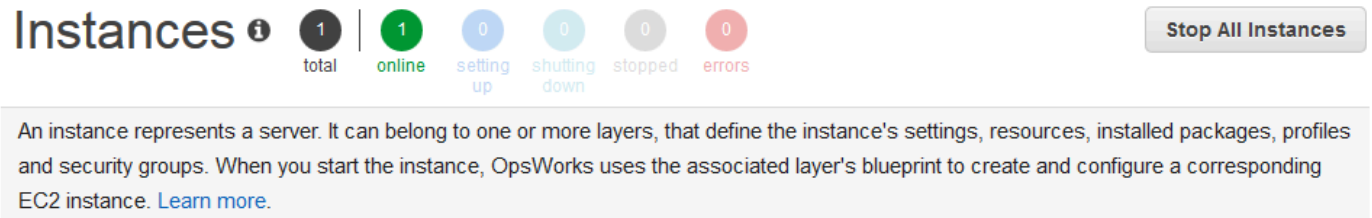
- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューション

シオンに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

インスタンスでのアプリケーションのデプロイ結果をテストします。

インスタンス上のデプロイをテストするには

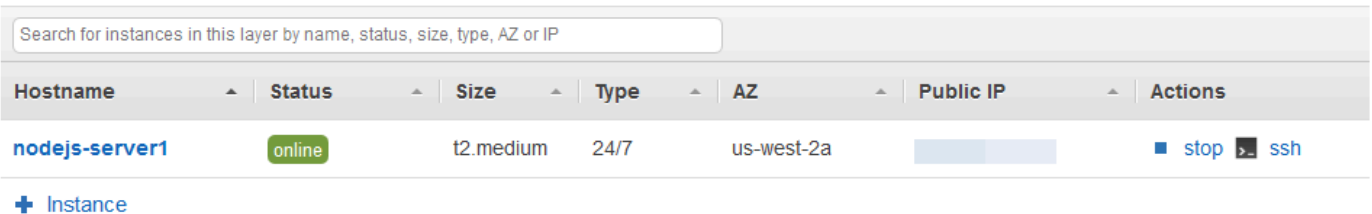
1. 前のステップで [Instances] (インスタンス) ページを表示した状態で、[Node.js App Server] (Node.js アプリケーションサーバー)、[nodejs-server1] (nodejs サーバー)、[Public IP] (パブリック ID) で IP アドレスを選択します。



Instances ⓘ | 1 total | 1 online | 0 setting up | 0 shutting down | 0 stopped | 0 errors | [Stop All Instances](#)

An instance represents a server. It can belong to one or more layers, that define the instance's settings, resources, installed packages, profiles and security groups. When you start the instance, OpsWorks uses the associated layer's blueprint to create and configure a corresponding EC2 instance. [Learn more](#).

Node.js App Server



Search for instances in this layer by name, status, size, type, AZ or IP

Hostname	Status	Size	Type	AZ	Public IP	Actions
nodejs-server1	online	t2.medium	24/7	us-west-2a		stop ssh

[+ Instance](#)

2. おめでとうウェブページの [Leave a comment] テキストボックスにコメントを入力し、[Send] を選択してアプリケーションをテストします。コメントがウェブページに追加されます。コメントの追加を継続し、何度でも [Send] を選択します。



Congratulations!

You just deployed your first app with AWS OpsWorks.

[Tweet](#) [Follow @AWSOpsWorks](#)

 **OpsWorks**
Made in Berlin

This app runs on nodejs-app-1 (Linux). Your request came from [redacted]
[redacted] The system time is 11/18/2015, 9:19:10 PM. Page rendered using Node.js version v4.1.1.

Leave a comment

Send

Hello, World!
11/18/2015, 9:19:10 PM

3. Twitter アカウントをお持ちの場合は、ツイートまたはフォロー @AWSOpsWorks を選択し、画面の指示に従ってアプリについてツイートするか、@ に従いますAWSOpsWorks。

これで、インスタンス上で正常にアプリケーションをテストおよびデプロイしました。

残りのステップでは、AWS OpsWorks スタックとそのコンポーネントの設定を調べるためのスタックコンソール。[次のステップ](#)では、スタックの設定を調べることで確認を開始できます。

ステップ 5: スタックの設定を確認する

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)をください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

方法を確認する AWS OpsWorks スタックはスタックをセットアップします。

スタックの設定を表示するには

1. サービスのナビゲーションバーで [Stack] を選択します。[My Sample Stack (Linux)] ページが表示されます。
2. [Stack Settings] を選択します。[Settings My Sample Stack (Linux)] ページに次のように表示されます。



Settings My Sample Stack (Linux) Edit	
Settings	
Stack name	My Sample Stack (Linux)
Region	US East (N. Virginia)
VPC	No VPC
Default Availability Zone	us-east-1a
Default operating system	Amazon Linux 2017.03
Default SSH key	No default key
Chef version	12
Use custom Chef cookbooks	yes
Repository type	HTTP Archive
Repository URL	https://s3.amazonaws.com/opsworks-demo-assets/opsworks-linux-demo-cookbooks-nodejs.tar.gz
User name	-

設定の多くの詳細については、[Edit] を選択し、各設定の上にマウスを動かします (すべての設定に画面上の説明があるわけではありません)。これらの設定の詳細については、「[新しいスタックを作成する](#)」をご参照ください。

このチュートリアルで使用した Chef クックブックを確認するには、で [opsworks-linux-demo-cookbooks-nodejs](#) リポジトリを開きます GitHub。

[次のステップ](#)では、レイヤーの設定を確認できます。

ステップ 6:レイヤーの設定を確認する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

方法を確認する AWS OpsWorks スタックはレイヤーを設定します。

レイヤーの設定を表示するには

1. サービスナビゲーションペインで、[Layers] (レイヤー) を選択します。[Layers] (レイヤー) ページが表示されます。
2. [Node.js App Server] を選択します。[Layer Node.js App Server] ページが表示されます。レイヤーの設定を表示するには、一般設定、レシピ、ネットワーク、ボリューム、セキュリティを選択します。EBS

Layer Node.js App Server

Edit

Delete

Instances

Monitoring

General Settings

Recipes

Network

EBS Volumes

Security

CloudWatch Logs

Settings

Name	Node.js App Server
Short name	nodejs-server
OpsWorks ID	
Instance shutdown timeout	120 seconds
Auto healing enabled	yes

設定の多くの詳細については、[Edit] を選択し、各設定の上にマウスを動かします (すべての設定に画面上の説明があるわけではありません)。これらの設定の詳細については、「[OpsWorks レイヤーの設定の編集](#)」をご参照ください。

[次のステップ](#)では、インスタンスの設定とログを確認できます。

ステップ 7: インスタンスの設定とログを確認する

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

の設定を確認する AWS OpsWorks インスタンスの起動に使用されるスタック。以下のインスタンスログを調べることもできます。AWS OpsWorks スタックが作成されました。

インスタンスの設定とログを表示するには

1. サービスのナビゲーションペインで、[Instances] を選択します。[Instances] ページが表示されます。

2. [Node.js App Server] で、[nodejs-server1] を選択します。インスタンスのプロパティページが表示されます。

The screenshot shows the details page for an instance named 'nodejs-server1'. At the top right, there are buttons for 'Start', 'Edit', and 'Delete'. The page is divided into two main sections: 'Details' on the left and 'Monitoring', 'Volumes', 'Elastic Load Balancing', and 'Elastic IP' on the right.

Details

Hostname	nodejs-server1
Status	stopped
Layers	Node.js App Server
EC2 instance ID	i-██████████5
OpsWorks ID	██████████
Instance type	24/7
Size	c3.large
Availability Zone	us-east-1a
Operating system	Amazon Linux 2017.03
OW Agent version	Inherited from stack
Tenancy	default
Architecture	64bit
Virtualization type	paravirtual
EBS Optimized	no
Root device type	EBS backed
Root device ID	vol-██████████1d

Monitoring
OpsWorks uses CloudWatch metrics to provide detailed monitoring for your instance.

Volumes
No volumes. [Manage in resources.](#)

Elastic Load Balancing
This instance does not belong to any layers with an ELB attached. [Change layer settings.](#)

Elastic IP
No Elastic IP. [Manage in resources.](#)

3. インスタンスログを確認するには、[Logs] セクションの [Log] で [show] を選択します。

The screenshot shows the 'Logs' section of the instance details page. It contains a table with columns for 'Created at', 'Command', 'Comment', 'Duration', and 'Log'. The 'Log' column contains a 'show' link for each log entry, which is circled in red in the image.

Created at	Command	Comment	Duration	Log
✓ 2015-11-18 21:14:11 UTC	configure		00:01:09	show
✓ 2015-11-18 21:10:09 UTC	setup		00:04:02	show

4. AWS OpsWorks スタックは、別のウェブブラウザタブにログを表示します。

```
✓ Instance: nodejs-app-1 | Stack: My Sample Stack (Linux) | Layer: Node.js App Server | Type: configure

1 [2015-11-18T21:15:11+00:00] INFO: AWS OpsWorks instance , Agent version 4002-20151110164726
2 [2015-11-18T21:15:12+00:00] INFO: Started chef-zero at chefzero://localhost:8889 with repository at /opt/aws/opsworks/curre
3 One version per cookbook
4 data_bags at /var/lib/aws/opsworks/data.internal/data_bags
5 nodes at /var/lib/aws/opsworks/data.internal/nodes
6
7 [2015-11-18T21:15:12+00:00] INFO: Forking chef instance to converge...
8 [2015-11-18T21:15:12+00:00] INFO: *** Chef 12.4.1 ***
9 [2015-11-18T21:15:12+00:00] INFO: Chef-client pid: 586
10 [2015-11-18T21:15:14+00:00] WARN: Run List override has been provided.
11 [2015-11-18T21:15:14+00:00] WARN: Original Run List: []
12 [2015-11-18T21:15:14+00:00] WARN: Overridden Run List: [recipe[aws_opsworks_agent]]
13 [2015-11-18T21:15:14+00:00] INFO: Run List is [recipe[aws_opsworks_agent]]
14 [2015-11-18T21:15:14+00:00] INFO: Run List expands to [aws_opsworks_agent]
15 [2015-11-18T21:15:14+00:00] INFO: Starting Chef Run for nodejs-app-1.localdomain
```

一部のインスタンス設定が表示する内容を確認するには、[nodejs-server1] ページに戻り、[Stop] を選択して、確認のメッセージが表示されたら、[Stop] を選択します。[Status] (ステータス) が [stopping] (停止中) から [stopped] (停止) に変わった後に [Edit] (編集) を選択し、続いて各設定の上にカーソルを合わせます。(すべての設定に画面上の説明があるわけではありません)。これらの設定の詳細については、「[レイヤーへのインスタンスの追加](#)」をご参照ください。

設定の確認を終了したら、[Start] を選択してインスタンスを再起動し、[Status] が [online] に変わるまで待ちます。それ以外の場合、インスタンスは停止されたままであるため、後でアプリケーションをテストすることはできません。

Note

インスタンスにログインしてさらに詳しく調べる場合は、`まず` を指定する必要があります。AWS OpsWorks パブリックSSHキーに関する情報を含むスタック (ssh-keygen や P などのツールを使用して作成できます uTTYgen) と、ユーザーがインスタンスにログインできるようにするには、My Sample Stack (Linux) スタックに対するアクセス許可を設定する必要があります。手順については、「[ユーザーのパブリック SSH キーの登録](#)」および「[SSH でのログイン](#)」を参照してください。

[次のステップ](#)では、アプリケーションの設定を確認します。

ステップ 8: アプリケーションの設定を確認する

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

の設定を確認する AWS OpsWorks アプリケーションに使用されるスタック。

アプリケーションの設定を表示するには

1. サービスナビゲーションペインで、[Apps] を選択します。[Apps] ページが表示されます。
2. [Node.js Sample App] を選択します。[App Node.js Sample App] ページが表示されます。

The screenshot displays the AWS OpsWorks console interface for an application named "Node.js Sample App". At the top right, there are three buttons: "Deploy App" (in blue), "Edit", and "Delete". Below the title, the "Settings" section is visible, listing the following details:

Name	Node.js Sample App
Short name	nodejs_sample_app
OpsWorks ID	[REDACTED]
Type	Other

Below the settings, there are two columns: "Application Source" and "Data Sources".

Application Source		Data Sources	
App source type	Git	Data source type	OpsWorks
Repository URL	https://github.com/awslabs/opsworks-windows-demo-nodejs.git	Database instance	(automatic selection)
		Database name	nodejs_sample_app

At the bottom of the screenshot, the "Environment Variables" section is partially visible.

一部の設定が表す内容の詳細については、[Edit] を選択し、各設定の上にマウスを動かします。(すべての設定に画面上の説明があるわけではありません)。これらの設定の詳細については、「[アプリケーションの追加](#)」を参照してください。

[次のステップ](#)では、レイヤーのモニタリングレポートを確認します。

ステップ 9:レイヤーのモニタリングレポートを確認する

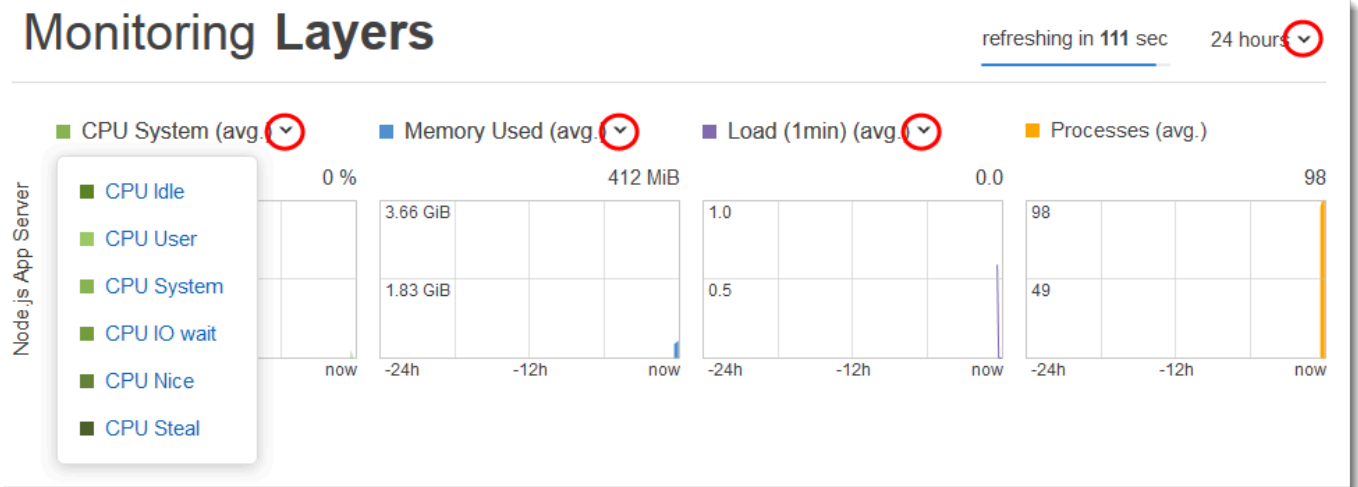
⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

次のレポートを調べる AWS OpsWorks スタックは、レイヤーのコンピューティングパフォーマンスについて を生成します。

レイヤーのモニタリングのレポートを表示するには

1. サービスのナビゲーションペインで、[Monitoring] を選択します。[Monitoring Layers] (レイヤーのモニタリング) ページが表示されます。
2. 追加のビューを確認するには、メモリCPU、ロード、時間の横にある矢印を選択します。



これらのレポートやその他のレポートの詳細については、「[Amazon の使用 CloudWatch](#)」および「[モニタリング](#)」を参照してください。

[次のステップ](#)では、追加のスタック設定を確認できます。

ステップ 10: 追加のスタック設定を確認する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)をください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

このステップでは、追加のスタック設定を確認できます。

AWS OpsWorks スタックは個別のデプロイを実行せず、追加のリソースをプロビジョニングせず、このスタックの一部として追加のアクセス許可を調整しなかったため、デプロイとコマンド、リソース、およびアクセス許可ページにはあまり関心がありません。これらの設定を表示する場合は、サービスナビゲーションペインで、それぞれ [Deployments]、[Resources]、および [Permissions] を選択します。これらのページが表す内容の詳細については、「[アプリケーションのデプロイ](#)」、「[リソース管理](#)」、および「[ユーザー許可の管理](#)」を参照してください。

[次のステップ](#)では、[をクリーンアップ](#)できます。AWS このチュートリアルで使用した リソース。このステップは任意です。これらを引き続き使用したい場合があります。AWS の詳細を学習し続けるための リソース AWS OpsWorks スタック。ただし、これらを AWS のリソースは、[に継続的に課金](#)される場合があります。AWS アカウント。これらを保持する場合 AWS 後の使用のために [リソースを取り巻く](#)ため、このチュートリアルはこれで完了です。 [にスキップ](#)できます [次のステップ](#)。

ステップ 11 (オプション): クリーンアップする

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)

わしてください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

に追加料金が発生しないようにするには AWS アカウントでは、アプリと を削除できます。AWS インスタンスや など、このチュートリアルで使用した リソース AWS OpsWorks スタックスタック。(詳細については、「」を参照してください。 [AWS OpsWorks 料金表](#)) ただし、これらの を引き続き使用することをお勧めします。AWS の詳細を学習し続けるための リソース AWS OpsWorks スタック。これらを保持する場合 AWS リソースが利用可能になりました。このチュートリアルはこれで完了です。 にスキップできます [次のステップ](#)。

このチュートリアルのために作成したリソースに保存されているコンテンツには、個人識別情報が含まれている可能性があります。この情報の保存が不要になった場合 AWS、このトピックのステップに従います。

アプリケーションをスタックから削除するには

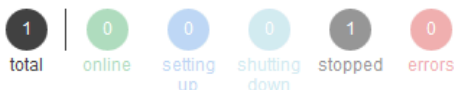
1. サービスナビゲーションペインで、[Apps] を選択します。[Apps] ページが表示されます。
2. [Node.js Sample App] で、[Actions] の [delete] を選択します。確認メッセージが表示されたら、[Delete] を選択します。アプリケーションを削除すると、[No apps] メッセージが表示されます。

スタックのインスタンスを削除するには

1. サービスのナビゲーションペインで、[Instances] を選択します。[Instances] ページが表示されます。
2. [Node.js App Server] で、[nodejs-server1]、[Actions] の [stop] を選択します。確認メッセージが表示されたら、[Stop] を選択します。

このプロセスには数分かかることがあります。メトリック AWS OpsWorks スタックが終了すると、次の結果が表示されます。

Instances

[Start All Instances](#)

An instance represents a server. It can belong to one or more layers, that define the instance's settings, resources, installed packages, profiles and security groups. When you start the instance, OpsWorks uses the associated layer's blueprint to create and configure a corresponding EC2 instance. [Learn more.](#)

Node.js App Server

Hostname	Status	Size	Type	AZ	Public IP	Actions
nodejs-server1	stopped	t2.medium	24/7	us-west-2a	-	▶ start 🗑 delete

[+ Instance](#)

3. [Actions] (アクション) で、[delete] (削除) を選択します。確認メッセージが表示されたら、[Delete] を選択します。インスタンスは削除され、[No instances] メッセージが表示されます。

スタックを削除するには

1. サービスナビゲーションペインで、[Stack] を選択します。[My Sample Stack (Linux)] ページが表示されます。
2. [Delete Stack] を選択します。確認メッセージが表示されたら、[Delete] を選択します。スタックが削除され、OpsWorksダッシュボードページが表示されます。

必要に応じて、このチュートリアルで利用したユーザーと Amazon EC2キーペアを他のへのアクセスに再利用したくない場合は、削除できます。AWS サービスとEC2インスタンス。手順については、[IAM「ユーザーと Amazon のキーペアと Linux インスタンスの削除」](#)を参照してください。

[EC2](#)

これで、このウォークスルーが完了しました。詳細については、「[次のステップ](#)」を参照してください。

次のステップ

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューション

シオンに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

このチュートリアルを完了したので、 の使用についてさらに詳しく知ることができます。AWS OpsWorks スタック :

- を使用して、このスタックを手動で再作成する方法 AWS OpsWorks スタック。「[入門ガイド: Linux](#)」を参照してください。
- クックブックと、AWS OpsWorks このウォークスルーに使用されるスタック。付属の [詳細: このウォークスルーで使用されているクックブックの学習](#) ウォークスルーの「[詳細: このウォークスルーで使用されているアプリケーションの学習](#)」および「[入門ガイド: Linux](#)」を参照してください。
- の使用を練習する AWS OpsWorks Windows インスタンスを含む スタック。「[入門ガイド: Windows](#)」を参照してください。
- [新しいスタックを作成する](#) の使用方法を参照してスタックの詳細について学びます。
- 「[OpsWorks レイヤーの設定の編集](#)」でレイヤーの詳細について学びます。
- 「[レイヤーへのインスタンスの追加](#)」でインスタンスの詳細について学びます。
- 「[アプリケーションのデプロイ](#)」でアプリケーションの詳細について学びます。
- [クックブックとレシピ](#) の詳細を確認してください。
- 独自のクックブックを作成します。「[使用開始: クックブック](#)」を参照してください。
- 「[セキュリティと権限](#)」でスタックへのアクセスをコントロールする方法について説明します。

Linux スタックの使用開始

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

このチュートリアルでは、 の使用方法を説明します。AWS OpsWorks Node.js アプリケーション環境を作成するスタック。完了すると、Chef 12 を実行する Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、Node.js HTTPサーバー、および Twitter を操作してウェブページにコメントを残すために使用できるウェブアプリが作成されます。

Chef は、EC2インスタンスなどのサーバーを設定および維持し、それらのサーバーにアプリをデプロイおよび維持するためのサードパーティフレームワークです。Chef に慣れていない場合は、このチュートリアルを完了した後、Chef の詳細を学習して、これらをすべて活用することをお勧めします。AWS OpsWorks スタックは を提供する必要があります。(詳細については、[Learn Chef](#) のウェブサイトを参照してください。)

AWS OpsWorks スタックは、Amazon Linux、Ubuntu Server、CentOS、Red Hat Enterprise Linux の 4 つの Linux ディストリビューションをサポートしています。このウォークスルーでは、Ubuntu Server を使用します。AWS OpsWorks スタックでは、Windows Server も使用できます。Windows Server スタックには同等のチュートリアルがありますが、 に関する基本的な概念を学習するには、まずこのチュートリアルを完了することをお勧めします。AWS OpsWorks そこで繰り返されないスタックと Chef。このウォークスルーを完了した後で、[入門ガイド: Windows](#) のウォークスルーを参照してください。

トピック

- [ステップ 1: 前提条件を完了する](#)
- [ステップ 2: スタックを作成する](#)
- [ステップ 3: スタックにレイヤーを追加する](#)
- [ステップ 4: インスタンスにデプロイするアプリケーションを指定する](#)
- [ステップ 5: インスタンスを起動する](#)
- [ステップ 6: インスタンスにアプリケーションをデプロイする](#)
- [ステップ 7: インスタンスにデプロイされたアプリケーションをテストする](#)
- [ステップ 8 \(オプション\): クリーンアップする](#)
- [次のステップ](#)
- [詳細: このウォークスルーで使用されているクックブックの学習](#)
- [詳細: このウォークスルーで使用されているアプリケーションの学習](#)

ステップ 1: 前提条件を完了する

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

ウォークスルーを開始する前に、次のセットアップ手順を完了します。これらのセットアップ手順には、へのサインアップが含まれます。AWS アカウント、管理ユーザーの作成、へのアクセス許可の割り当て AWS OpsWorks スタック。

すでに「[使用開始: サンプル](#)」ウォークスルーを完了している場合はこのウォークスルーの前提条件を満たしているため、省略して「[ステップ 2: スタックを作成する](#)」にお進みください。

トピック

- [にサインアップする AWS アカウント](#)
- [管理アクセスを持つユーザーを作成する](#)
- [サービスアクセス権限を割り当てます。](#)

にサインアップする AWS アカウント

をお持ちでない場合 AWS アカウントで、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. <https://portal.aws.amazon.com/billing/サインアップ> を開きます。
2. オンラインの手順に従います。

サインアップ手順の一環として、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

にサインアップするとき AWS アカウント、AWS アカウントのルートユーザー が作成されます。ルートユーザーはすべてのにアクセスできます AWS のサービス アカウントの および リ

ソース。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用して[ルートユーザーアクセスが必要なタスク](#)を実行してください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。<https://aws.amazon.com/> に移動し、マイアカウント を選択すると、いつでも現在のアカウントアクティビティを表示し、アカウントを管理できます。

管理アクセスを持つユーザーを作成する

にサインアップした後 AWS アカウント、 をセキュリティで保護する AWS アカウントのルートユーザー、有効化 AWS IAM Identity Center、および 管理ユーザーを作成して、日常的なタスクにルートユーザーを使用しないようにします。

のセキュリティ保護 AWS アカウントのルートユーザー

1. [にサインインします。AWS Management Console](#) ルートユーザーを選択し、AWS アカウント E メールアドレス。次のページでパスワードを入力します。

ルートユーザーを使用してサインインする方法については、「」の「[ルートユーザーとしてサインイン](#)する」を参照してください。AWS サインイン ユーザーガイド。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

手順については、「 の仮想MFAデバイスの[有効化](#)」を参照してください。[AWS アカウントIAM ユーザーガイドのルートユーザー \(コンソール\)](#)。

管理アクセスを持つユーザーを作成する

1. IAM Identity Center を有効にします。

手順については、「 [の有効化](#)」を参照してください。[AWS IAM Identity Center](#) ()AWS IAM Identity Center ユーザーガイド。

2. IAM Identity Center で、ユーザーに管理アクセス権を付与します。

の使用に関するチュートリアル IAM アイデンティティセンターディレクトリ ID ソースとして、「[デフォルトを使用してユーザーアクセスを設定する](#)」を参照してください。[IAM アイデンティティセンターディレクトリ](#) ()AWS IAM Identity Center ユーザーガイド。

管理アクセス権を持つユーザーとしてサインインする

- IAM Identity Center ユーザーでサインインするには、IAM Identity Center ユーザーの作成時に E メールアドレスに URL 送信されたサインインを使用します。

IAM Identity Center ユーザーを使用してサインインする方法については、「への[サインイン](#)」を参照してください。[AWS の アクセスポータル](#) AWS サインイン ユーザーガイド。

追加のユーザーにアクセス権を割り当てる

1. IAM Identity Center で、最小特権のアクセス許可を適用するベストプラクティスに従うアクセス許可セットを作成します。

手順については、「」の「[アクセス許可セットの作成](#)」を参照してください。AWS IAM Identity Center ユーザーガイド。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

手順については、「」の「[グループの追加](#)」を参照してください。AWS IAM Identity Center ユーザーガイド。

サービスアクセス権限を割り当てます。

へのアクセスを有効にする AWS OpsWorks スタックサービス (および AWS OpsWorks スタックは、ロールまたはユーザーに AWSOpsWorks_FullAccess および アクセス AmazonS3FullAccess 許可を追加することで) に依存します。

アクセス許可の追加の詳細については、[IAM 「ID アクセス許可の追加 \(コンソール\)」](#) を参照してください。

これですべてのセットアップステップが完了したので、[このウォークスルーを開始](#)できます。

ステップ 2: スタックを作成する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)

わしてください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。


を使用します。AWS OpsWorks スタックを作成するためのスタックコンソール。スタックは、インスタンスおよび関連する のコレクションです。AWS 共通の目的を持ち、一緒に管理する リソース。(詳細については、[スタック](#) を参照してください)。このウォークスルーでは、インスタンスは 1 つのみです。

開始する前に、まだ行っていない場合は[前提条件](#)を満たします。

スタックを作成するには

1. にサインインする AWS Management Console を開き、AWS OpsWorks の コンソール<https://console.aws.amazon.com/opsworks/>。
2. 該当する場合、次のいずれかを実行します。
 - へようこそ AWS OpsWorks スタックページが表示され、最初のスタックを追加するまたは最初のスタックを追加を選択します。AWS OpsWorks スタックスタック (両方の選択肢が同じことを行います) 。[Add stack] ページが表示されます。
 - OpsWorks ダッシュボードページが表示されたら、スタックの追加 を選択します。[Add stack] ページが表示されます。
3. [Add stack] ページが表示されたら、すでに選択されていない場合は [Chef 12 stack] を選択します。
4. [Stack name] ボックスに、**MyLinuxDemoStack** のようなスタックの名前を入力します (別の名前を入力することはできますが、必ずこのウォークスルー全体でこれを MyLinuxDemoStack に置き換えてください)。
5. [リージョン] で [米国西部(オレゴン)] を選択します。
6. の場合はVPC、次のいずれかを実行します。
 - VPC が利用可能な場合は、それを選択します。(詳細については、[でのスタックの実行 VPC](#) を参照してください)。
 - それ以外の場合は [いいえVPC] を選択します。
7. デフォルトのオペレーティングシステム で、Linux と Ubuntu 18.04 を選択します。 LTS
8. [Use custom Chef cookbooks] で、[Yes] を選択します。
9. [Repository type] で、[Http Archive] を選択します。

10. リポジトリ の場合URL、 と入力します。 <https://s3.amazonaws.com/opsworks-demo-assets/opsworks-linux-demo-cookbooks-nodejs.tar.gz>
11. 以下はデフォルト値のままにします。
 - [Default Availability Zone] (us-west-2a)
 - デフォルトSSHキー (デフォルトSSHキー は使用しないでください)
 - [User name] (空白)
 - [Password] (空白)
 - [Stack color] (dark blue)
12. [Advanced] (アドバンスド) を選択します。
13. IAM ロール については、次のいずれかを実行します (詳細については、「」を参照してください [AWS OpsWorks スタックがユーザーに代わって動作することを許可する](#))。
 - aws-opsworks-service-role が利用可能な場合は、それを選択します。
 - aws-opsworks-service-role が使用できない場合は、新しいIAMロール を選択します。
14. デフォルトのIAMインスタンスプロファイル については、次のいずれかを実行します (詳細については、「」を参照してください [EC2 インスタンスで実行されるアプリケーションのアクセス許可の指定](#))。
 - aws-opsworks-ec2 ロールが使用可能な場合は、それを選択します。
 - aws-opsworks-ec2 ロールが使用できない場合は、新しいIAMインスタンスプロファイル を選択します。
15. API エンドポイントリージョン で、スタックを関連付けるリージョンAPIエンドポイントを選択します。スタックを米国東部 (バージニア北部) リージョンのエンドポイント内の米国西部 (オレゴン) リージョンに配置する場合は、[us-east-1] を選択します。スタックを米国西部 (オレゴン) リージョンに配置するとともに米国西部 (オレゴン) リージョンのエンドポイントに関連付ける場合は、[us-west-2] を選択します。

 Note

米国東部 (バージニア北部) リージョンのエンドポイントには、古い が含まれます。AWS リージョン 下位互換性のために、管理している場所に最も近いリージョンエンドポイントを選択することがベストプラクティスです。AWS。詳細については、「」を参照してください [リージョンのサポート](#)。


16. 以下はデフォルト値のままにします。


- デフォルトのルートデバイスタイプ (EBS がサポート)
- [Hostname theme] (Layer Dependent)
- OpsWorks エージェントバージョン (最新バージョン)
- カスタム JSON (空白)
- OpsWorks セキュリティグループを使用する (はい)


17. 結果は、VPC IAM ロール、およびデフォルト IAM インスタンスプロファイルを除く次のスクリーンショットと一致する必要があります。

Add stack

Which type of stack do you want to create?

 **Sample stack**
Explore AWS OpsWorks Stacks with a sample Node.js app

 **Chef 12 stack**
Bring your own cookbooks and use community cookbooks

 **Chef 11 stack**
Use built-in cookbooks for applications and deployments

Create a stack with Linux or Windows instances that run Chef 12

The more advanced experience. Bring your own cookbooks and use community cookbooks. AWS OpsWorks Stacks does separate Chef runs to isolate its internal cookbooks from yours. [Learn more.](#)

Stack name	<input type="text" value="MyLinuxDemoStack"/>
Region	<input type="text" value="US West (Oregon)"/>
VPC	<input type="text" value="No VPC"/>
Default Availability Zone	<input type="text" value="us-west-2a"/>
Default operating system	<input checked="" type="radio"/> Linux <input type="radio"/> Windows <input type="text" value="Ubuntu 16.04 LTS"/> <i>Need a different OS? Let us know.</i>
Default SSH key	<input type="text" value="Do not use a default SSH key"/>
Chef version	12
Use custom Chef cookbooks	<input checked="" type="checkbox"/> Yes <i>Define the source of your Chef cookbooks</i>
Repository type	<input type="text" value="Git"/>
Repository URL	<input type="text" value="https://github.com/opsworks-cookbooks/opsworks-recipes"/>

Repository type: Git

Repository URL: https://github.com/user/cookbooks.git

Repository SSH key: Optional

Branch/Revision: Optional

Stack color: [Color selection icons]

Advanced options

Default root device type: EBS backed, Instance store

IAM role: aws-opsworks-service-role

Default IAM instance profile: aws-opsworks-ec2-role

API endpoint region **NEW**: us-west-2 **REGIONAL**, us-east-1 **CLASSIC**

Hostname theme: Layer Dependent

OpsWorks Agent version: 4021 (Dec 16th 2016)

Custom JSON: Optional

Enter custom JSON that is passed to your Chef recipes for all instances in your stack. You can use this to override and customize built-in recipes or pass variables to your own recipes. [Learn more.](#)

Security

Use OpsWorks security groups: Yes

Cancel Add stack

18. [Add Stack] を選択します。AWS OpsWorks スタックはスタックを作成し、MyLinuxDemoStackページを表示します。

このウォークスルー用の正しい設定のスタックが作成されました。

[次のステップ](#)では、スタックにレイヤーを追加します。

ステップ 3: スタックにレイヤーを追加する

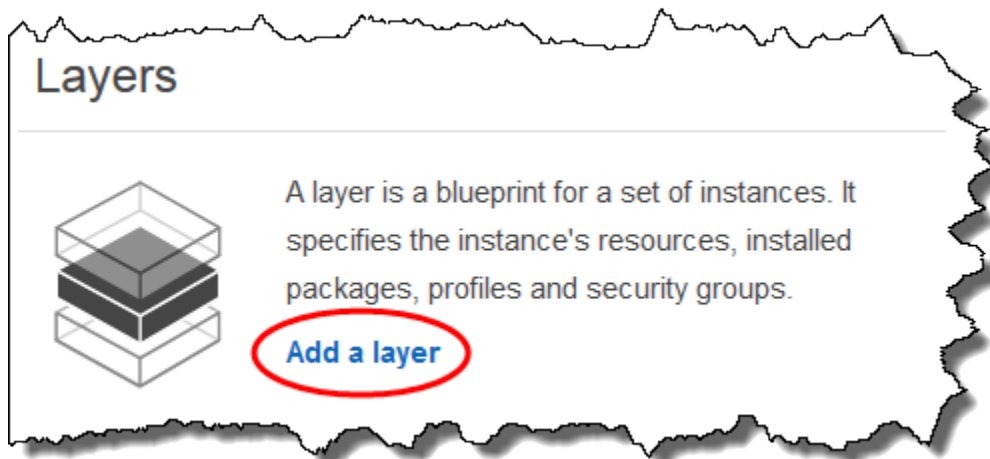
⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

レイヤーは、Amazon インスタンスなどの一連の EC2 インスタンスの設計図です。インスタンス設定、リソース、インストールされているパッケージ、セキュリティグループなどの情報を指定します。次に、スタックにレイヤーを追加します (レイヤーの詳細については、「[レイヤー](#)」を参照してください)。

レイヤーをスタックに追加するには

1. 前のステップで表示された MyLinuxDemoStack ページで、レイヤー でレイヤーの追加 を選択します。



2. [Add Layer] (レイヤーを追加) ページが表示されます。OpsWorks タブの名前に入力します **MyLinuxDemoLayer**。(別の名前を入力することはできますが、必ずこのワークスルー全体でこれを MyLinuxDemoLayer に置き換えてください)。
3. [Short name] に「**demo**」と入力します (別の名前を入力することはできますが、必ずこのワークスルー全体でこれを demo に置き換えてください)。

Add layer

OpsWorks ECS RDS

A layer is a blueprint and container for your instances. You can add Chef recipes to lifecycle events of your instances, for example to install and configure any required software. [Learn more](#).

Name

Short name

Need further support? [Let us know](#).

Cancel Add layer

- [Add layer] を選択します。AWS OpsWorks スタックによりレイヤーが作成され、[レイヤー] ページが表示されます。
- レイヤーページで MyLinuxDemoLayer、ネットワーク を選択します。
- [Network] タブの、[Automatically Assign IP Addresses] で、[Public IP addresses] が [yes] に設定されていることを確認します。変更した場合、[Save] を選択します。

Automatically Assign IP Addresses ⓘ

Public IP addresses

yes

Elastic IP addresses

No

- [Layers] (レイヤー) ページで、[Security] (セキュリティ) を選択します。

Layers

A layer is a blueprint for a set of Amazon EC2 instances. It specifies the instance's settings, associated resources, installed packages, profiles, and security groups. You can also add recipes to lifecycle events of your instances, for example: to set up, deploy, configure your instances, or discover your resources. [Learn more](#).

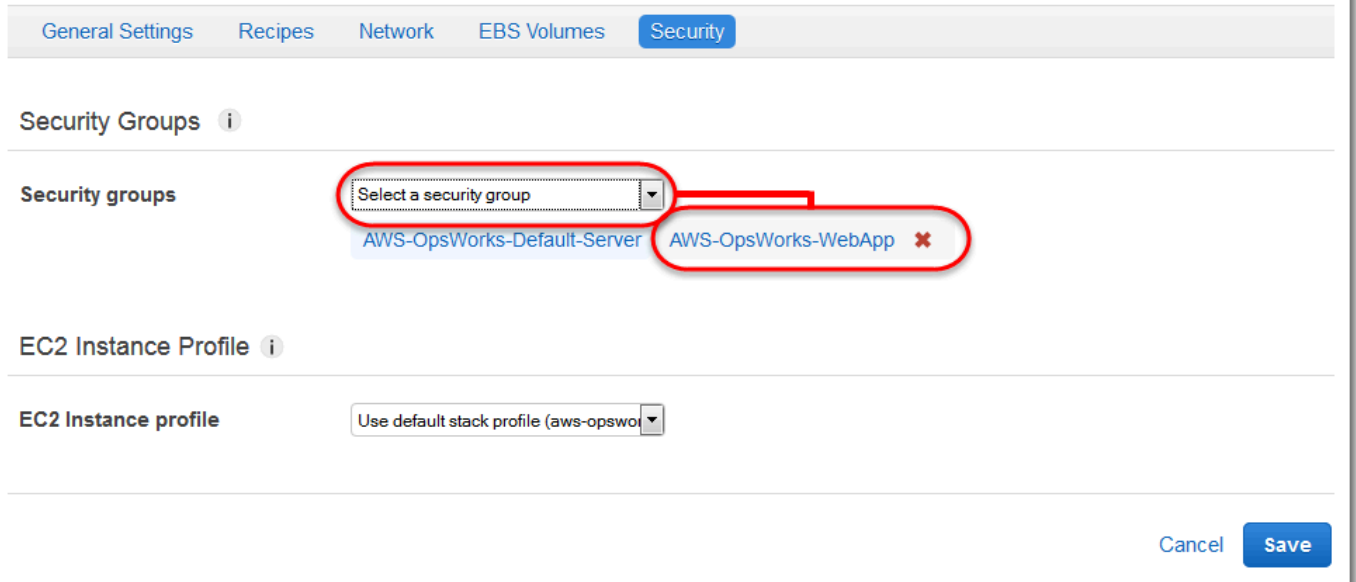
 **MyLinuxDemoLayer**
[Settings](#) [Recipes](#) [Network](#) [EBS Volumes](#) [Security](#) 🗑️ Delete

No instances
[Add instance](#)

[+ Layer](#)

- レイヤー MyLinuxDemoLayerページが表示され、セキュリティタブが開きます。セキュリティグループで AWS-OpsWorks-WebApp を選択し、保存 を選択します。

Layer MyLinuxDemoLayer



The screenshot shows the 'Layer MyLinuxDemoLayer' configuration page in the AWS OpsWorks console. The 'Security' tab is active. Under 'Security Groups', a dropdown menu is open, showing 'AWS-OpsWorks-Default-Server' and 'AWS-OpsWorks-WebApp' (selected). Under 'EC2 Instance Profile', the dropdown shows 'Use default stack profile (aws-opswo)'. 'Cancel' and 'Save' buttons are at the bottom right.

- AWS-OpsWorks-WebApp セキュリティグループがレイヤーに追加されます (このセキュリティグループがあると、このウォークスルーで後述するインスタンス上のアプリにユーザーが接続できるようになります。このセキュリティグループがないと、ユーザーはウェブブラウザでインスタンスに接続できないというメッセージを受け取ります)。

このウォークスルー用の正しい設定のレイヤーが作成されました。

[次のステップ](#)では、インスタンスにデプロイするアプリケーションを指定します。

ステップ 4: インスタンスにデプロイするアプリケーションを指定する

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)

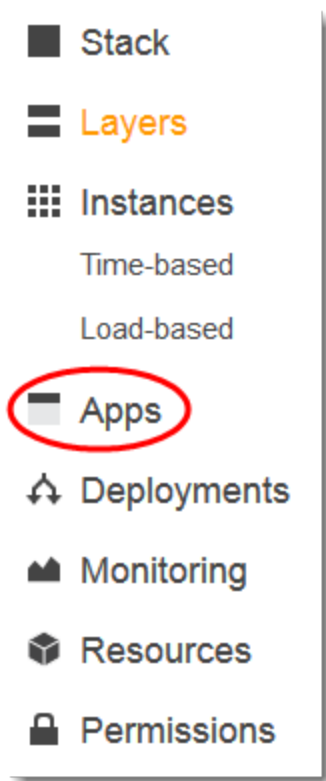
わしてください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

指示する AWS OpsWorks このチュートリアルの後半でインスタンスにデプロイするアプリケーションに関するスタック。このコンテキストでは、AWS OpsWorks スタックは、インスタンスで実行するコードとしてアプリケーションを定義します。(詳細については、[アプリケーション](#) を参照してください)。

このセクションの手順は、Chef 12 以降のスタックに適用されます。Chef 11 スタックのレイヤーにアプリを追加する方法については、「[ステップ 2.4: アプリケーション - Chef 11 を作成してデプロイする](#)」を参照してください。

デプロイするアプリケーションを指定するには

1. サービスナビゲーションペインで、[Apps] を選択します。



2. [Apps] ページが表示されます。[Add an app] を選択します。[Add App] ページが表示されます。
3. [Settings] で、[Name] に「**MyLinuxDemoApp**」と入力します (別の名前を入力することはできませんが、必ずこのワークスルー全体でこれを MyLinuxDemoApp に置き換えてください)。

- アプリケーションソースの場合、リポジトリの場合URL、と入力します。 **https://github.com/awslabs/opsworks-windows-demo-nodejs.git**
- 以下はデフォルト値のままにします。
 - [Settings]、[Document root] (空白)
 - [Data Sources]、[Data source type] (None)
 - [Repository type] (Git)
 - リポジトリSSHキー (空白)
 - [Branch/Revision] (空白)
 - 環境変数 (空白 KEY、空白 VALUE、未チェックの保護値)
 - [Add Domains]、[Domain Name] (空白)
 - SSL 設定、有効化 SSL (なし)

Add App

Settings

Name

Document root

Data Sources

Data source type RDS None

Application Source

Repository type

Repository URL

Repository SSH key

Branch/Revision

Environment Variables

KEY	VALUE	<input type="checkbox"/> Protected value
-----	-------	--

Add Domains

Domain name +

SSL Settings

Enable SSL No

Cancel **Add App**

- [Add App] を選択します。AWS OpsWorks スタックによりアプリケーションが追加され、[Apps] (アプリケーション) ページが表示されます。

このウォークスルー用の正しい設定のアプリケーションが作成されました。

[次のステップ](#)では、インスタンスを起動します。

ステップ 5: インスタンスを起動する

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

使用アイテム AWS OpsWorks Ubuntu Server Amazon EC2 インスタンスを起動するスタック。このインスタンスでは、このウォークスルーで前に作成したレイヤーで定義した設定が使用されます (詳しくは、[インスタンス](#) を参照してください)。

インスタンスを起動するには

1. サービスのナビゲーションペインで、[Instances] を選択します。[Instances] ページが表示されます。
2. でMyLinuxDemoLayer、インスタンスの追加 を選択します。
3. [New] タブで、次のデフォルト値をそのままにします。
 - [Hostname] (demo1)
 - [Size] (c3.large)
 - サブネット (*IP address* us-west-2a)
4. [Advanced] (アドバンスド) を選択します。
5. 以下はデフォルト値のままにします。
 - Scaling type (24/7)
 - SSH キー (デフォルトSSHキー は使用しないでください)
 - オペレーティングシステム (Ubuntu 18.04 LTS)
 - OpsWorks エージェントバージョン (スタック から継承)
 - テナンシー (デフォルト - VPC設定に依存します)
 - ルートデバイスタイプ (EBS backed)
 - ボリュームタイプ (汎用 (SSD))
 - [Volume size] (8)
6. 結果は以下のスクリーンショットのようになります。

New
 Existing OpsWorks
 EC2 instances and own servers

Hostname:

Size:

Subnet:

Scaling type:

 24/7

 Time-based

 Load-based

SSH key:

Operating system:

OpsWorks Agent version:

Tenancy:

Root device type:

 EBS backed

 Instance store

Volume type:

Volume size:

Min: 8 GiB, Max: 16384 GiB

[Cancel](#)
[Add Instance](#)

7. [Add Instance] を選択します。AWS OpsWorks スタックによりインスタンスがレイヤーに追加され、[Instances] (インスタンス) ページが表示されます。
8. の場合MyLinuxDemoLayer、demo1 の場合、アクション の場合、開始 を選択します。

MyLinuxDemoLayer

Hostname	Status	Size	Type	AZ	Public IP	Actions
demo1	stopped	c3.large	24/7	us-west-2a	-	▶ start 🗑 delete

+ Instance

9. 数分以内に、以下が発生します。
 - [setting up] の円が [0] から [1] に変わります。

- [Status] が [stopped] から [requested]、[pending]、[booting]、[running_setup] に変わり、最終的に [online] になります。この処理には数分かかることもありますのでご注意ください。
 - [(ステータス] が [オンライン] に変わった後、[設定] 円形インジケータが [1] から [0] に変わり、[オンライン (オンライン)] の円が [0] から [1] に変わって明るい緑色になります。[online] の円が明るい緑色になり、[1] つのインスタンスがオンラインであることが表示されるまで、先の手順に進まないでください。
10. 結果は、続行する前に次のスクリーンショットに一致する必要があります (エラーメッセージが表示される場合は、「[デバッグとトラブルシューティングのガイド](#)」を参照してください)。

Instances 1 total | 1 online | 0 setting up | 0 shutting down | 0 stopped | 0 errors Stop All Instances

MyLinuxDemoLayer

Search for instances in this layer by name, status, size, type, AZ or IP

Hostname	Status	Size	Type	AZ	Public IP	Actions
demo1	online	c3.large	24/7	us-west-2a		stop ssh

+ Instance

これで、アプリケーションをデプロイできるインスタンスが作成されました。

Note

インスタンスにログインしてさらに詳しく調べる場合は、まず `ssh-keygen` を指定する必要があります。AWS OpsWorks パブリックSSHキーに関する情報を含むスタック (`ssh-keygen` や `P` などのツールを使用して作成できます `uTTYgen`)。次に `MyLinuxDemoStack`、スタックに対するアクセス許可を設定して、ユーザーがインスタンスにログインできるようにする必要があります。手順については、「[ユーザーのパブリック SSH キーの登録](#)」および「[SSH でのログイン](#)」を参照してください。SSH を使用して Pu 経由でインスタンスに接続する場合は TTY、の「[Pu を使用して Windows から Linux インスタンスに接続する TTY](#)」を参照してください。AWS ドキュメント内) を参照してください。

[次のステップ](#)では、アプリケーションをインスタンスにデプロイします。

ステップ 6: インスタンスにアプリケーションをデプロイする

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

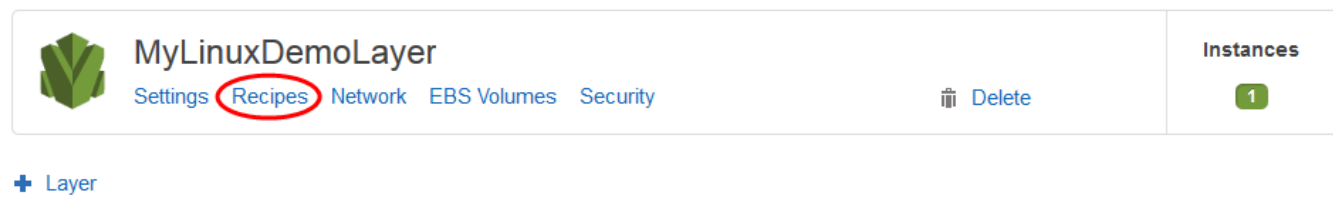
このステップでは、 から実行中のインスタンス GitHub にアプリケーションをデプロイします。(詳細については、[アプリケーションのデプロイ](#) を参照してください)。アプリケーションをデプロイする前に、デプロイを調整するために使用するレシピを指定する必要があります。レシピは Chef の概念です。レシピは、Ruby 言語の構文で書かれた手順であり、使用するリソースと、それらのリソースを適用する順序を指定します (詳細については、[「Learn Chef」](#) (Chef の説明) ウェブサイトで [「About Recipes」](#) (レシピについて) をご覧ください。)

アプリケーションをインスタンスにデプロイするために使用するレシピを指定するには

1. サービスナビゲーションペインで、[Layers] (レイヤー) を選択します。[Layers] (レイヤー) ページが表示されます。
2. で MyLinuxDemoLayer、レシピ を選択します。

Layers

A layer is a blueprint for a set of Amazon EC2 instances. It specifies the instance's settings, associated resources, installed packages, profiles, and security groups. You can also add recipes to lifecycle events of your instances, for example: to set up, deploy, configure your instances, or discover your resources. [Learn more](#).



The screenshot shows the AWS OpsWorks console interface for a layer named 'MyLinuxDemoLayer'. The layer is represented by a green leaf icon. Below the name, there are several tabs: 'Settings', 'Recipes', 'Network', 'EBS Volumes', and 'Security'. The 'Recipes' tab is currently selected and highlighted with a red circle. To the right of the tabs is a 'Delete' button with a trash icon. On the far right, there is a box labeled 'Instances' with a green circle containing the number '1'. Below the layer card, there is a '+ Layer' button.

レイヤー MyLinuxDemoLayer ページが表示され、レシピタブが開きます。

3. [Custom Chef Recipes]、[Deploy] で、タイプ「**nodejs_demo::default**」と入力し、Enter キーを押します。nodejs_demo はクックブックの名前で、default はクックブック内の対象レシピの名前です (レシピのコードについて調べるには、「[詳細: このウォークスルーで使用されているクックブックの学習](#)」を参照してください)。結果は、次のスクリーンショットに一致する必要があります。

Layer MyLinuxDemoLayer

General Settings **Recipes** Network EBS Volumes Security

Custom Chef Recipes ⓘ

Repository URL `https://s3.amazonaws.com/opsworks-demo-assets/opsworks-linux-demo-cookbooks-nodejs.tar.gz`
(change)

0 Setup	<code>mycookbook::myrecipe, mycookt</code> +
0 Configure	<code>mycookbook::myrecipe, mycookt</code> +
1 Deploy	<code>mycookbook::myrecipe, mycookt</code> + <code>nodejs_demo::default</code> ✖
0 Undeploy	<code>mycookbook::myrecipe, mycookt</code> +
0 Shutdown	<code>mycookbook::myrecipe, mycookt</code> +

Cancel Save

4. [Save] を選択します。AWS OpsWorks スタックはレシピをレイヤーのデプロイライフサイクルイベントに追加します。

アプリケーションをインスタンスにデプロイするには

1. サービスナビゲーションペインで、[Apps] を選択します。[Apps] ページが表示されます。
2. ではMyLinuxDemoApp、アクションで、次のスクリーンショットに示すように のデプロイを選択します。

Apps

An app represents code stored in a repository that you want to install on application server instances. [Learn more.](#)

Name	Type	Data Source	Last Deployment	Actions
MyLinuxDemoApp	Other			 deploy  edit  delete
+ App				

3. [Deploy App] ページで、次のデフォルト値をそのままにします。

- [Command] (Deploy)
- [Comment] (空白)
- 設定、アドバンスト、カスタム Chef JSON (空白)
- インスタンス、アドバンスト (チェック すべてのを 選択、チェック MyLinuxDemoLayer、チェック demo1)

4. 結果は、次のスクリーンショットに一致する必要があります。

Deploy App

Settings

App MyLinuxDemoApp

Command

Deploy an app.

Comment

Custom Chef JSON

Enter custom JSON that is passed to your Chef recipes for all instances in your stack. You can use this to override and customize built-in recipes or pass variables to your own. [Learn more.](#)

Instances ⓘ

OpsWorks will run this command on **1 of 1** instances. The assigned recipes are run on all selected instances.

Select all

MyLinuxDemoLayer demo1 ●

Click to select instances in this layer

Cancel

- [デプロイ] を選択します。デプロイ MyLinuxDemoApp – デプロイページが表示されます。
[Status] が [running] から [successful] に変わります。[demo1] の横に回転する円が表示され、その後緑のチェックマークに変わります。この処理には数分かかることもありますのでご注意ください。[Status] が [successful] となるとともに、緑のチェックマークアイコンが表示されるまで続行しないでください。
- 結果は、当然ながら [Created at]、[Completed at]、[Duration]、および [User] を除いて次のスクリーンショットと一致する必要があります。[ステータス] が [失敗] の場合、トラブルシューティングするには、[ログ] で [表示] を選択してエラーの詳細を確認します。

Deployment MyLinuxDemoApp - deploy

[Repeat](#)

Status **successful** User OpsWorksDemoUser

Created at 2015-11-12 17:12:49 UTC

Completed at 2015-11-12 17:14:02 UTC

Duration 00:01:13

Hostname	SSH	Layers	Duration	Log
✓ demo1	ssh	MyLinuxDemoLayer	00:01:13	show

これで、インスタンスに正常にアプリケーションをデプロイしました。

[次のステップ](#)では、インスタンスでデプロイ済みのアプリケーションをテストします。

ステップ 7: インスタンスにデプロイされたアプリケーションをテストする

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

ここで、インスタンス上のアプリケーションのデプロイをテストします。

インスタンス上のデプロイをテストするには

1. サービスのナビゲーションペインで、[Instances] を選択します。[Instances] ページが表示されます。
2. では MyLinuxDemoLayerdemo1、パブリック IP では IP アドレスを選択します。

Instances

1 total | 1 online | 0 setting up | 0 shutting down | 0 stopped | 0 errors

[Stop All Instances](#)

MyLinuxDemoLayer

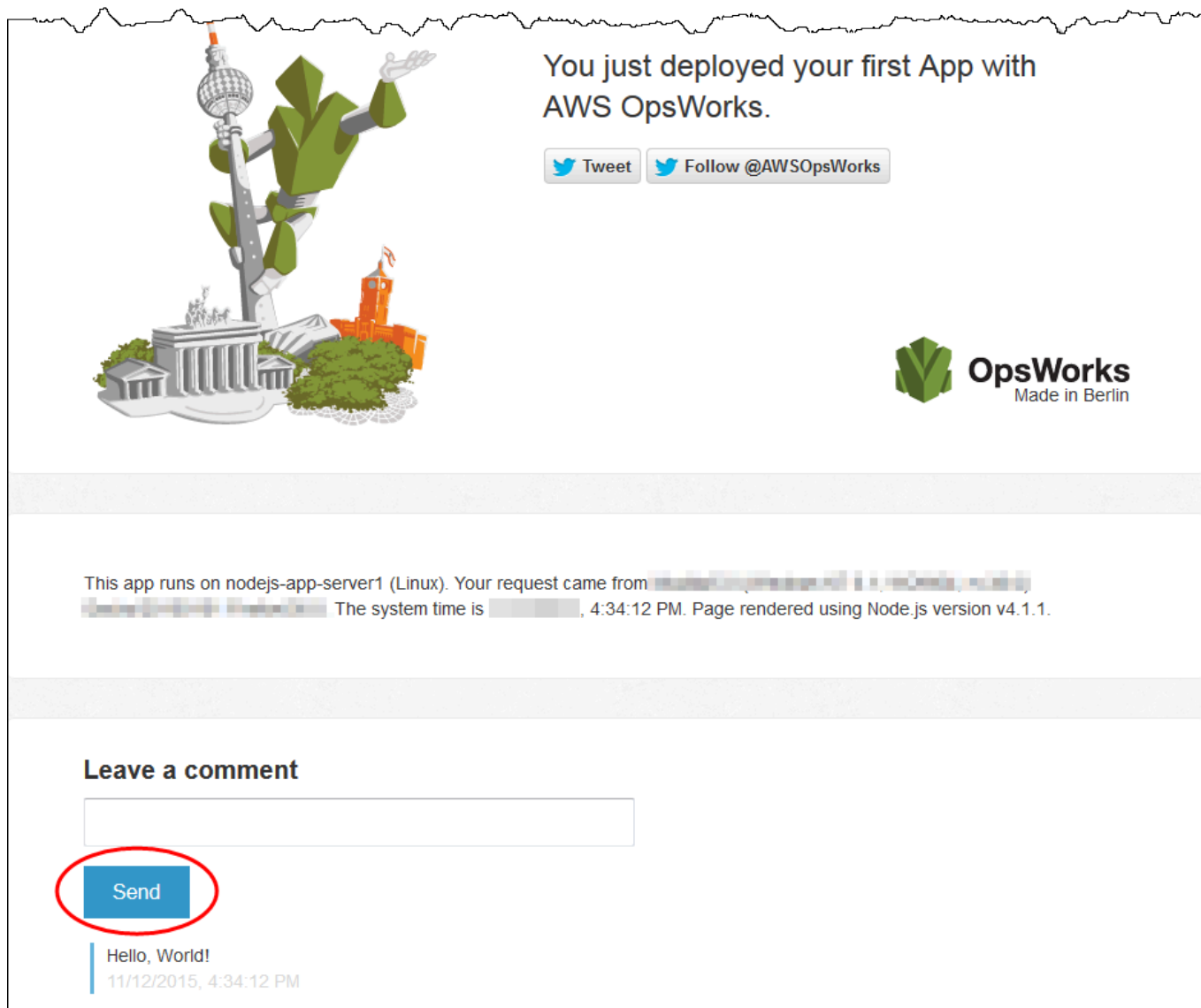
Search for instances in this layer by name, status, size, type, AZ or IP

Hostname	Status	Size	Type	AZ	Public IP	Actions
demo1	online	c3.large	24/7	us-west-2a		stop ssh

[+ Instance](#)


新しいウェブブラウザタブがアプリケーションに表示されます。

- おめでとうウェブページの [Leave a comment] テキストボックスにコメントを入力し、[Send] を選択してアプリケーションをテストします。コメントがウェブページに追加されます。コメントの追加を継続し、何度でも [Send] を選択します。



You just deployed your first App with AWS OpsWorks.

[Tweet](#) [Follow @AWSOpsWorks](#)

 **OpsWorks**
Made in Berlin

This app runs on nodejs-app-server1 (Linux). Your request came from [redacted]. The system time is [redacted], 4:34:12 PM. Page rendered using Node.js version v4.1.1.

Leave a comment

[Send](#)

Hello, World!
11/12/2015, 4:34:12 PM

4. Twitter アカウントをお持ちの場合は、ツイートまたはフォロー @AWSOpsWorks を選択し、画面の指示に従ってアプリをツイートするか、@ をフォローします AWSOpsWorks。

これで、インスタンス上で正常にアプリケーションをテストおよびデプロイしました。

[次のステップ](#) では、AWS このチュートリアルで使用した リソース。このステップは任意です。これらを引き続き使用したい場合があります。AWS の詳細を学習し続けるための リソース AWS OpsWorks スタック。ただし、これらを AWS のリソースは、に継続的に課金される場合があります。AWS アカウント。これらを保持する場合 AWS 後で使用するために リソースを取り巻くため、このチュートリアルはこれで完了です。にスキップできます [次のステップ](#)。

ステップ 8 (オプション): クリーンアップする

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

に追加料金が発生しないようにするには AWS アカウント、 を削除できます。AWS このチュートリアルで使用した リソース。これら AWS リソースには、 が含まれます。AWS OpsWorks スタックスタックとスタックのコンポーネント。(詳細については、「」を参照してください。[AWS OpsWorks 料金表](#))ただし、これらの を引き続き使用することをお勧めします。AWS の詳細を学習し続けるための リソース AWS OpsWorks スタック。これらを保持する場合 AWS これで、 リソースが利用可能になりました。このチュートリアルは完了です。 にスキップできます [次のステップ](#)。

このチュートリアルのために作成したリソースに保存されているコンテンツには、個人識別情報が含まれている可能性があります。によってこの情報を保存しなくなった場合 AWS、このトピックのステップに従います。

アプリケーションをスタックから削除するには

1. 左 AWS OpsWorks スタックコンソールのサービスナビゲーションペインで、アプリ を選択します。[Apps] ページが表示されます。
2. ではMyLinuxDemoApp、アクション では、 の削除を選択します。確認メッセージが表示された場合は、[Delete] を選択します。AWS OpsWorks スタックによりアプリケーションが削除されます。

スタックのインスタンスを削除するには

1. サービスのナビゲーションペインで、[Instances] を選択します。[Instances] ページが表示されます。
2. の場合MyLinuxDemoLayer、demo1 の場合、アクション の場合、停止 を選択します。確認メッセージが表示されたら、[Stop] を選択します。以下のことが起こります。
 - [Status] が [online] から [stopping] に変わり、最終的に [stopped] になります。

- [online] は [1] から [0] に変わります。
- [shutting down] は [0] から [1] に変わり、最終的に [0] になります。
- [stopped] は最終的に [0] から [1] に変わります。

このプロセスには数分かかることがあります。メトリック AWS OpsWorks スタックが終了すると、次の結果が表示されます。

Instances 1 total | 0 online | 0 setting up | 0 shutting down | 1 stopped | 0 errors Start All Instances

MyLinuxDemoLayer

Search for instances in this layer by name, status, size, type, AZ or IP

Hostname	Status	Size	Type	AZ	Public IP	Actions
demo1	stopped	c3.large	24/7	us-west-2a		▶ start 🗑️ delete

+ Instance

3. [Actions] (アクション) で、[delete] (削除) を選択します。確認メッセージが表示されたら、[Delete] を選択します。AWS OpsWorks スタックによってインスタンスが削除され、[No instances] (インスタンスがありません) と表示されます。

スタックを削除するには

1. サービスナビゲーションペインで、[Stack] を選択します。[MyLinuxDemoStack] ページが表示されます。
2. [Delete Stack] を選択します。確認メッセージが表示されたら、[Delete] を選択します。AWS OpsWorks スタックはスタックを削除し、OpsWorks ダッシュボードページを表示します。

オプションで、他のへのアクセスに再利用したくない場合は、このチュートリアルに使用したユーザーと Amazon EC2 キーペアを削除できます。AWS サービスと EC2 インスタンス。手順については、[IAM 「ユーザーと Amazon キーペアと Linux インスタンスの削除」](#) を参照してください。 [EC2](#)

これで、このウォークスルーが完了しました。詳細については、「[次のステップ](#)」を参照してください。

次のステップ

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

このチュートリアルを完了したので、[このチュートリアル](#) の使用についてさらに詳しく知ることができます。AWS OpsWorks スタック :

- このウォークスルーに使用したクックブックとアプリケーションについて調べます。「[詳細: このウォークスルーで使用されているクックブックの学習](#)」および「[詳細: このウォークスルーで使用されているアプリケーションの学習](#)」を参照してください。
- の使用を練習する AWS OpsWorks Windows インスタンスを含む スタック。「[入門ガイド: Windows](#)」を参照してください。
- [新しいスタックを作成する](#) の使用方法を参照してスタックの詳細について学びます。
- 「[OpsWorks レイヤーの設定の編集](#)」でレイヤーの詳細について学びます。
- 「[レイヤーへのインスタンスの追加](#)」でインスタンスの詳細について学びます。
- 「[アプリケーションのデプロイ](#)」でアプリケーションの詳細について学びます。
- [クックブックとレシピ](#) の詳細を確認してください。
- 独自のクックブックを作成します。「[使用開始: クックブック](#)」を参照してください。
- 「[セキュリティと権限](#)」でスタックへのアクセスをコントロールする方法について説明します。

詳細: このウォークスルーで使用されているクックブックの学習

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。

わしてください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

このトピックでは、クックブックについて説明します。AWS OpsWorks ウォークスルーに使用されるスタック。

クックブックは Chef の概念です。クックブックは、レシピ、属性値、ファイル、テンプレート、ライブラリ、定義、カスタムリソースなどの設定情報を含むアーカイブファイルです。レシピは、Chef の概念でもあります。レシピは、Ruby 言語の構文で書かれた手順であり、使用するリソースと、それらのリソースを適用する順序を指定します 詳細については、[「Learn Chef」](#) (Chef の説明) ウェブサイトで [「About Cookbooks」](#) (クックブックについて) および [「About Recipes」](#) (レシピについて) をご覧ください。

このチュートリアルで使ったクックブックの内容を確認するには、[opsworks-linux-demo-cookbooks-nodejs.tar.gz](#) ファイルの内容をローカルワークステーションの空のディレクトリに抽出します。(クックブックをデプロイしたインスタンスにログインし、`/var/chef/cookbooks` ディレクトリの内容を確認することもできます。)

`cookbooks/nodejs_demo/recipes` ディレクトリの `default.rb` ファイルは、クックブックがそのコードを実行する場所です。

```
app = search(:aws_opsworks_app).first
app_path = "/srv/#{app['shortname']}"

package "git" do
  options "--force-yes" if node["platform"] == "ubuntu" && node["platform_version"] == "18.04"
end

application app_path do
  javascript "4"
  environment.update("PORT" => "80")

  git app_path do
    repository app["app_source"]["url"]
    revision app["app_source"]["revision"]
  end

  link "#{app_path}/server.js" do
    to "#{app_path}/index.js"
  end
end
```

```
end

npm_install
npm_start
end
```

ファイルで実行される操作は次のとおりです。

- `search(:aws_opsworks_app).first` は Chef 検索を使用して、最終的にインスタンスにデプロイされるアプリケーションについての情報を検索します。この情報には、アプリケーションの短縮名、ソースリポジトリの詳細などの設定が含まれます。このウォークスルーでデプロイされたのは 1 つのアプリケーションのみであるため、Chef 検索ではインスタンスの `aws_opsworks_app` 検索インデックス内の情報の最初の項目から、これらの設定を取得します。インスタンスが起動されるたびに、AWS OpsWorks スタックは、この情報およびその他の関連情報をインスタンス自体に一連のデータバッグとして保存し、Chef 検索を通じてデータバッグの内容を取得します。このレシピにこれらの設定をハードコードすることができますが、データバッグと Chef 検索を使用するのが、より堅牢な方法です。データバッグの詳細については、Chef の詳細ウェブサイトでの「[AWS OpsWorks スタックデータバッグリファレンス](#)」を参照してください。[\[Learn Chef\]](#) (Chef の説明) ウェブサイトで [\[About Data Bags\]](#) (データバッグについて) も参照してください。Chef の検索の詳細については、[\[Learn Chef\]](#) (Chef の説明) ウェブサイトで [\[About Search\]](#) (検索について) を参照してください。
- `package` リソースはインスタンスに Git をインストールします。
- `application` リソースはウェブアプリケーションを記述し、デプロイします。
 - `javascript` は、インストールする JavaScript ランタイムのバージョンです。
 - `environment` は環境変数を設定します。
 - `git` は指定されたリポジトリとブランチからソースコードを取得します。
 - `app_path` はリポジトリをクローン化するパスです。パスがインスタンスに存在しない場合は、AWS OpsWorks スタックによって作成されます。
 - `link` はシンボリックリンクを作成します。
 - `npm_install` は、Node.js 用のデフォルトのパッケージマネージャーであるノードパッケージマネージャをインストールします。
 - `npm_start` は Node.js を実行します。

ただし、AWS OpsWorks スタックは、このチュートリアルで使用したクックブックを作成し、独自のクックブックを作成できます。この方法の詳細は、[使用開始: クックブック](#)を参照してください

い。また、[「Learn Chef」](#) (Chef の説明) ウェブサイトで [「About Cookbooks」](#) (クックブックについて)、[「About Recipes」](#) (レシピについて)、および [「Learn the Chef Basics on Ubuntu」](#) (Ubuntu でシェフの基礎を学ぶ) を、[「Getting started with Chef」](#) (Chef の使用開始) ウェブサイトでは [「First steps with Chef」](#) (Chef の最初のステップ) にある「最初のシェフクックブック」セクションもご覧ください。

詳細: このウォークスルーで使用されているアプリケーションの学習

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

このトピックでは、AWS OpsWorks スタックは、このチュートリアルのためにインスタンスにデプロイされます。

アプリケーションのソースコードを表示するには、[opsworks-windows-demo-nodejs](#) GitHub リポジトリの内容をローカルワークステーションの空のディレクトリに抽出します。クックブックをデプロイしたインスタンスにログインし、`/srv/mylinuxdemoapp` ディレクトリの内容を確認することもできます。

`index.js` ファイルには、アプリケーションで最も重要なコードが含まれています。

```
var express = require('express');
var app = express();
var path = require('path');
var os = require('os');
var bodyParser = require('body-parser');
var fs = require('fs');

var add_comment = function(comment) {
  var comments = get_comments();
  comments.push({"date": new Date(), "text": comment});
  fs.writeFileSync('./comments.json', JSON.stringify(comments));
};

var get_comments = function() {
```

```
var comments;
if (fs.existsSync('./comments.json')) {
  comments = fs.readFileSync('./comments.json');
  comments = JSON.parse(comments);
} else {
  comments = [];
}
return comments;
};

app.use(function log (req, res, next) {
  console.log([req.method, req.url].join(' '));
  next();
});
app.use(express.static('public'));
app.use(bodyParser.urlencoded({ extended: false }));

app.set('view engine', 'jade');
app.get('/', function(req, res) {
  var comments = get_comments();
  res.render("index",
    { agent: req.headers['user-agent'],
      hostname: os.hostname(),
      nodeversion: process.version,
      time: new Date(),
      admin: (process.env.APP_ADMIN_EMAIL || "admin@unconfigured-value.com" ),
      comments: get_comments()
    });
});

app.post('/', function(req, res) {
  var comment = req.body.comment;
  if (comment) {
    add_comment(comment);
    console.log("Got comment: " + comment);
  }
  res.redirect("/#form-section");
});

var server = app.listen(process.env.PORT || 3000, function() {
  console.log('Listening on %s', process.env.PORT);
});
```

ファイルで実行される操作は次のとおりです。

- `require` は、このウェブアプリケーションを予期どおりに実行するために必要な一部の依存コードを含むモジュールをロードします。
- `add_comment` 関数と `get_comments` 関数は、`comments.json` ファイルに対して情報の書き込みと読み取りを行います。
- `app.get`、`app.listen`、`app.set` および `app.use` の詳細については `Express` `app.post` リファレンス を参照してください。 [API](#)

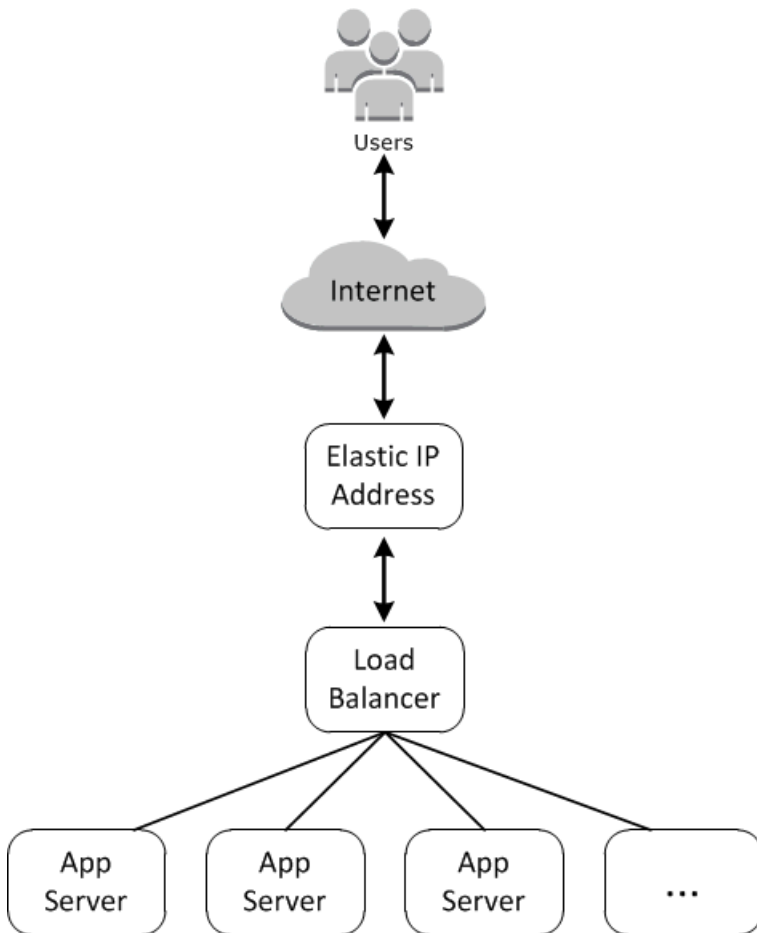
デプロイ用のアプリケーションの作成とパッケージ化の方法については、「[Application Source](#)」を参照してください。

Windows スタックの使用開始

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS サポート](#) でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

通常、クラウドベースのアプリケーションでは、アプリケーションサーバー、データベースサーバーなどの関連リソースのグループが必要で、一括して作成および管理する必要があります。この一連のインスタンスはスタックと呼ばれます。シンプルなアプリケーションスタックの例を以下に示します。



基本的なアーキテクチャは以下の要素で構成されます。

- ユーザーリクエストを受信する Elastic IP アドレス。
- リクエストをアプリケーションサーバーに均等に分散させるロードバランサー。
- トラフィックを処理するために必要な数の一連のアプリケーションサーバーインスタンス。

また、通常、アプリケーションサーバーにアプリケーションを配布する方法や、ユーザー アクセス権限を管理する方法も必要です。

AWS OpsWorks スタックは、スタックおよび関連するアプリケーションとリソースを作成および管理するためのシンプルで簡単な方法を提供します。この章では、の基本について説明します。AWS OpsWorks スタックは、そのより高度な機能の一部とともに、図のアプリケーションサーバースタックを作成するプロセスを順を追って説明します。次のような増分開発モデルを使用しています。AWS OpsWorks スタックは簡単にフォローできます。基本的なスタックを設定し、正しく動作したら、フル機能の実装に到達するまでコンポーネントを追加します。

- 「[ステップ 1: 前提条件を完了する](#)」に、ウォークスルーを開始するためのセットアップ方法を示します。
- [ステップ 2: 基本的なアプリケーションサーバースタックを作成する](#) は、インターネットインフォメーションサービス (IIS) をサポートする基本的なスタックを作成し、アプリケーションをサーバーにデプロイする方法を示しています。
- 「[ステップ 3: スケールアウトする IISExample](#)」では、さらに多くのアプリケーションサーバー、着信トラフィックを分散させるためのロードバランサー、着信リクエストを受信する Elastic IP アドレスを追加することによって、増大した負荷を処理するためにスタックをスケールアウトする方法を示します。

トピック

- [ステップ 1: 前提条件を完了する](#)
- [ステップ 2: 基本的なアプリケーションサーバースタックを作成する](#)
- [ステップ 3: スケールアウトする IISExample](#)
- [次のステップ](#)

ステップ 1: 前提条件を完了する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

ウォークスルーを開始する前に、次のセットアップ手順を完了します。これらのセットアップ手順には、へのサインアップが含まれます。AWS アカウント、管理ユーザーの作成、へのアクセス許可の割り当て AWS OpsWorks スタック。

すでに「[使用開始: サンプル](#)」または「[入門ガイド: Linux](#)」ウォークスルーを完了している場合、このウォークスルーの前提条件を満たしており、省略して「[ステップ 2: 基本的なアプリケーションサーバースタックを作成する](#)」に進むことができます。

トピック

- [にサインアップする AWS アカウント](#)
- [管理アクセスを持つユーザーを作成する](#)
- [サービスアクセス権限を割り当てます。](#)
- [以下を確認してください。 AWS OpsWorks ユーザーがドメインに追加されるスタック](#)

にサインアップする AWS アカウント

をお持ちでない場合 AWS アカウントで、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. <https://portal.aws.amazon.com/billing/サインアップ> を開きます。
2. オンラインの手順に従います。

サインアップ手順の一環として、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

にサインアップするとき AWS アカウント、AWS アカウントのルートユーザー が作成されます。ルートユーザーはすべての にアクセスできます AWS のサービス アカウントの および リソース。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用して [ルートユーザーアクセスが必要なタスク](#) を実行してください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。 <https://aws.amazon.com/> に移動し、マイアカウント を選択すると、いつでも現在のアカウントアクティビティを表示し、アカウントを管理できます。

管理アクセスを持つユーザーを作成する

にサインアップした後 AWS アカウント、 をセキュリティで保護する AWS アカウントのルートユーザー、有効化 AWS IAM Identity Center、および日常的なタスクにルートユーザーを使用しないように管理ユーザーを作成します。

のセキュリティ保護 AWS アカウントのルートユーザー

1. [にサインインします。AWS Management Console](#) ルートユーザーを選択し、AWS アカウント E メールアドレス。次のページでパスワードを入力します。

ルートユーザーを使用してサインインする方法については、[「」の「ルートユーザーとしてサインインする」](#)を参照してください。AWS サインイン ユーザーガイド。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

手順については、「[の仮想MFAデバイスの有効化](#)」を参照してください。AWS アカウントIAM ユーザーガイドのルートユーザー (コンソール)。

管理アクセスを持つユーザーを作成する

1. IAM Identity Center を有効にします。

手順については、「[の有効化](#)」を参照してください。AWS IAM Identity Center ()AWS IAM Identity Center ユーザーガイド。

2. IAM Identity Center で、ユーザーに管理アクセス権を付与します。

の使用に関するチュートリアル IAM アイデンティティセンターディレクトリ ID ソースとして、「[デフォルトを使用してユーザーアクセスを設定する](#)」を参照してください。IAM アイデンティティセンターディレクトリ ()AWS IAM Identity Center ユーザーガイド。

管理アクセス権を持つユーザーとしてサインインする

- IAM Identity Center ユーザーでサインインするには、IAM Identity Center ユーザーの作成時に E メールアドレスに URL 送信されたサインインを使用します。

IAM Identity Center ユーザーを使用してサインインする方法については、「[へのサインイン](#)」を参照してください。AWS の アクセスポータル AWS サインイン ユーザーガイド。

追加のユーザーにアクセス権を割り当てる

1. IAM Identity Center で、最小特権のアクセス許可を適用するベストプラクティスに従うアクセス許可セットを作成します。

手順については、「[」の「アクセス許可セットの作成](#)」を参照してください。AWS IAM Identity Center ユーザーガイド。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

手順については、「[」の「グループの追加](#)」を参照してください。AWS IAM Identity Center ユーザーガイド。

サービスアクセス権限を割り当てます。

へのアクセスを有効にする AWS OpsWorks スタックサービス (および AWS OpsWorks スタックは、ロールまたはユーザーに `AWSOpsWorks_FullAccess` および `アクセスAmazonS3FullAccess` 許可を追加することで) に依存します。

アクセス許可の追加の詳細については、[IAM「ID アクセス許可の追加 \(コンソール\)」](#) を参照してください。

以下を確認してください。AWS OpsWorks ユーザーがドメインに追加されるスタック

Chef 12.2 スタックでは、付属の `aws_opsworks_users` クックブックによって、Windows ベースのインスタンスへの SSH およびリモートデスクトッププロトコル (RDP) アクセスを持つユーザーが作成されます。スタック内の Windows インスタンスを Active Directory ドメインに結合すると、AWS OpsWorks スタックユーザーは Active Directory に存在しません。ドメインとの結合後に再起動すると、Active Directory でユーザーが認識されない場合はインスタンスの状態が `setup failed` になります。ドメインに参加している Windows インスタンスでは、を付与するだけでは不十分です。AWS OpsWorks ユーザーアクセス許可ページでユーザー SSH/RDP アクセスをスタックします。

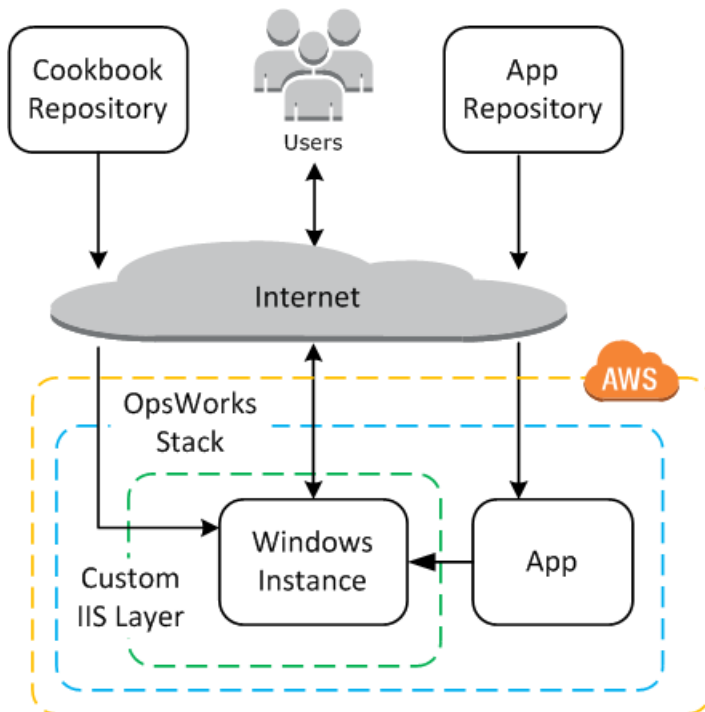
Chef 12.2 スタックの Windows インスタンスを Active Directory ドメインに結合する前に、すべての AWS OpsWorks Windows ベースのスタックのスタックユーザーは、ドメインのメンバーです。これを行う最善の方法は、Windows ベースのスタックを作成する IAM 前に `フェデレーテッド ID` を設定し、`フェデレーテッドユーザー` を `に` インポートすることです。AWS OpsWorks スタック内のインスタンスをドメインに結合する前のスタック。これを行う方法の詳細については、AWS セキュリティブログの「[Windows Active Directory、ADFS および 2.0 SAML AWS の使用へのフェデレーションの有効化](#)」およびユーザーガイドの「[既存ユーザーのフェデレーション](#)」を参照してください IAM。

ステップ 2: 基本的なアプリケーションサーバースタックを作成する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[にお問い合わせください](#)。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

基本的なアプリケーションサーバスタックは、ユーザーのリクエストを受信するためのパブリック IP アドレスを持つ、単一のアプリケーションサーバインスタンスで構成されます。アプリケーションコードと関連ファイルは別々のリポジトリに保存されており、そこからサーバーにデプロイされます。そのようなスタックを次の図に示します。



スタックには次のようなコンポーネントがあります。

- レイヤー。インスタンスのグループを表し、それらの設定方法を指定します。

この例のレイヤーは、IISインスタンスのグループを表します。

- Amazon インスタンスを表すEC2インスタンス。

この場合、レイヤーは を実行するように 1 つのインスタンスを設定しますがIIS、レイヤーには任意の数のインスタンスを含めることができます。

- アプリケーション: インスタンスにアプリケーションをインストールするために必要な情報が含まれます。
- カスタムIISレイヤーをサポートするカスタム Chef レシピを含むクックブック。クックブックとアプリケーションコードは、Amazon S3 バケットまたは Git リポジトリ内のアーカイブファイルなど、リモートリポジトリに保管されます。

以下のセクションでは、AWS OpsWorks スタックコンソールを使用してスタックを作成し、アプリケーションをデプロイします。

トピック

- [ステップ 2.1: スタックの作成](#)
- [ステップ 2.2: RDP アクセスを許可する](#)
- [ステップ 2.3: カスタムクックブックの実装](#)
- [ステップ 2.4: IISレイヤーを追加する](#)
- [ステップ 2.5: アプリケーションをデプロイする](#)

ステップ 2.1: スタックの作成

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

を起動する AWS OpsWorks スタックは、インスタンスやその他のリソースのコンテナとして機能するスタックを作成することによって射影します。スタック設定では、AWSリージョンやデフォルトのオペレーティングシステムなど、スタックのすべてのインスタンスで共有されるいくつかの基本設定を指定します。

新しいスタックを作成するには

1. スタックの追加

まだサインインしていない場合は、[こちら](#) にサインインします。

- アカウントに既存のスタックがない場合は、「ようこそAWS OpsWorks」ページが表示され、「最初のスタックを追加」を選択します。
- それ以外の場合は、[こちら](#) が表示されます。AWS OpsWorks アカウントのスタックを一覧表示する [スタックダッシュボード](#)。スタックを追加 [こちら](#) を選択します。

2. スタックの設定

[Add Stack] ページで、[Chef 12 stack] を選択し、次の設定を指定します。

スタックの名前

スタックの名前を入力します。英数字 (a~z、A~Z および 0~9) とハイフン (-) を含めることができます。このウォークスルーで使用する例のスタック名は、**IISWalkthrough** です。

リージョン

スタックのリージョンとして米国西部 (オレゴン) を選択します。

スタックはどのリージョンでもスタックを作成できますが、チュートリアルでは 米国西部 (オレゴン) をお勧めします。

Default operating system

[Windows] を選択し、デフォルト設定である [Microsoft Windows Server 2022 Base] を指定します。

Use custom Chef cookbooks

このウォークスルーでは、このオプションに [No] を指定します。

3. **Advanced** を選択して、IAMロールとデフォルトのIAMインスタンスプロファイルが選択されていることを確認します。

IAM ロール

スタックの IAM (AWS Identity and Access Management) ロール。AWS OpsWorks スタックは、Amazon EC2インスタンスの作成や管理などのタスクを実行するために、他の AWS サービスにアクセスする必要があります。IAM role は、AWS OpsWorks スタックは、ユーザーに代わって他の AWSサービスと連携することを前提としています。詳細については、「[AWS OpsWorks スタックがユーザーに代わって動作することを許可する](#)」を参照してください。

- アカウントに既存のがある場合 AWS OpsWorks スタックIAMロールは、リストから選択できます。

ロールが によって作成された場合 AWS OpsWorks スタックには という名前が付けられずaws-opsworks-service-role。

- それ以外の場合は、新しいIAMロールを選択して指示します。AWS OpsWorks 適切なアクセス許可を持つ新しいロールを作成するスタック。

注：AWS OpsWorks スタックのフルアクセス許可、新しいロールの作成には、いくつかの追加のIAM許可が必要です。詳細については、「[ポリシーの例](#)」を参照してください。

4. その他の設定についてはデフォルト値を受け入れて、[Add Stack] を選択します。さまざまなスタック設定の詳細については、「[新しいスタックを作成する](#)」を参照してください。

ステップ 2.2: RDP アクセスを許可する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

これでスタックを作成したので、レイヤーを作成し、Windows インスタンスをレイヤーに追加します。ただし、これを行う前に、[AWS OpsWorks RDP](#) を使用してカスタムレイヤーのインスタンスRDPに接続できるようにスタックを設定する必要があります。そのためには、以下を実行する必要があります。

- RDP アクセスを制御するインバウンドルールをセキュリティグループに追加します。
- の設定 AWS OpsWorks アクセスを許可するこのスタックのRDPアクセス許可をスタックします。

リージョンで最初のスタックを作成すると、AWS OpsWorks スタックは、一連のセキュリティグループを作成します。これにはAWS-OpsWorks-RDP-Server、のような名前が含まれます。AWS OpsWorks スタックはすべての Windows インスタンスにアタッチされ、RDPアクセスを許可します。ただし、デフォルトでは、このセキュリティグループにはルールがないため、インスタンスRDPへのアクセスを許可するインバウンドルールを追加する必要があります。

RDP アクセスを許可するには

1. [Amazon EC2コンソール](#) を開き、スタックのリージョンに設定し、ナビゲーションペインからセキュリティグループを選択します。
2. AWS-OpsWorksRDP-Server を選択し、Inbound タブを選択し、Edit を選択します。
3. [Add Rule] を選択し、次の設定を指定します。

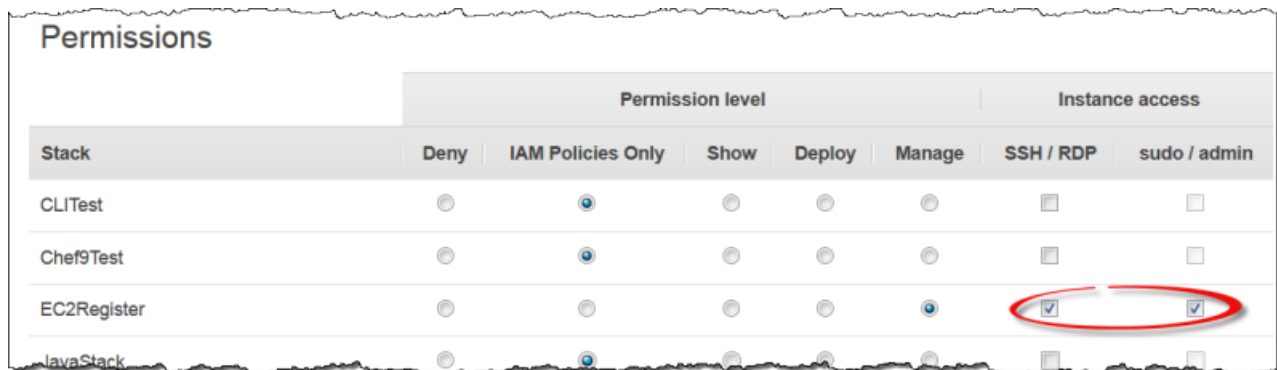
- タイプ – RDP。
- Source (ソース) – 許可される送信元 IP アドレス。

通常、IP アドレスまたは指定された IP アドレス範囲 (通常は会社の IP アドレス範囲) からのインバウンドRDPリクエストを許可します。学習目的では、多くの場合、任意の IP アドレスからのRDPアクセスを許可する 0.0.0.0/0 を指定するだけで十分です。

セキュリティグループは、インスタンスがRDP接続リクエストを受信することを許可しますが、これはストーリーの半分にすぎません。通常、ユーザーは、 が提供するパスワードを使用してインスタンスにログインします。AWS OpsWorks スタックを使用するには AWS OpsWorks スタックはそのパスワードを生成します。ユーザーのRDPアクセスを明示的に許可する必要があります。

ユーザーRDPに対して を許可するには

1. 左 AWS OpsWorks スタックダッシュボードで、IISWalkthroughスタックを選択します。
2. スタックのナビゲーションペインで、[Permissions] を選択します。
3. [Permissions] ページの [Edit] を選択します。
4. ユーザーのリストで、必要なアクセス許可を付与するユーザーの SSH/RDP のチェックボックスをオンにします。ユーザーに管理者アクセス権限を設定する場合、[sudo/admin] も選択します。



Stack	Permission level					Instance access	
	Deny	IAM Policies Only	Show	Deploy	Manage	SSH / RDP	sudo / admin
CLITest	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>
Chef9Test	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>
EC2Register	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
javaStack	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>

5. [Save] を選択します。

その後、ユーザーはパスワードを取得し、後で説明するようにインスタンスへのログインに使用します。

Note

管理者としてログインすることもできます。詳細については、「[管理者としてログイン](#)」を参照してください。

ステップ 2.3: カスタムクックブックの実装

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

スタックは、基本的にはインスタンスのコンテナですが、インスタンスを直接スタックに追加することはありません。それぞれ関連するインスタンスのグループを表すレイヤーを 1 つ以上追加し、そのレイヤーにインスタンスを追加します。

レイヤーは基本的に、AWS OpsWorks スタックは を使用して、同じ設定の Amazon EC2 インスタンスのセットを作成します。インスタンスは、オペレーティングシステムの基本バージョンから開始し、インスタンスのレイヤーはさまざまなタスクをインスタンスで実行して、次のような設計図を実装します。

- ディレクトリとファイルの作成
- ユーザーの管理
- ソフトウェアのインストールと設定
- サーバーの起動または停止
- アプリケーションコードおよび関連ファイルのデプロイ

レイヤーは、[\[Chef recipes\]](#) (シェフのレシピ) (略してレシピ) を実行してインスタンスに対してタスクを実行します。recipe は、Chef のドメイン固有の言語 (DSL) を使用してインスタンスの最終状態を記述する Ruby アプリケーションです。で AWS OpsWorks スタックでは、各レシピは通常、レイヤーの [ライフサイクルイベント](#) の 1 つに割り当てられます: セットアップ、設定、デプロイ、デプロイ

イ解除、シャットダウン。インスタンスでライフサイクルイベントが発生すると、AWS OpsWorks スタックはイベントのレシピを実行して、適切なタスクを実行します。たとえば、Setup イベントは、インスタンスの起動の完了後に発生します。AWS OpsWorks スタックはその後、Setup レシピを実行します。それは通常、サーバーソフトウェアのインストールや設定、関連サービスの起動などのタスクを実行します。

AWS OpsWorks スタックは、標準タスクを実行する一連の組み込みレシピを各レイヤーに提供します。カスタムレシピを実装して追加タスクを実行し、レイヤーのライフサイクルイベントに割り当てることで、レイヤーの機能を拡張できます。Windows スタックは、いくつかの基本タスクのみ実行する最小限のレシピセットを持つ[カスタムレイヤー](#)をサポートします。Windows インスタンスに機能を追加するには、ソフトウェアのインストール、アプリケーションのデプロイなどを行うカスタムレシピを実装する必要があります。このトピックでは、IISインスタンスをサポートするシンプルなカスタムレイヤーを作成する方法について説明します。

トピック

- [クックブックとレシピの簡単な説明](#)
- [インストールして開始するレシピの実装 IIS](#)
- [カスタムクックブックの有効化](#)

クックブックとレシピの簡単な説明

レシピは、インスタンスの予想される状態の 1 つ以上の側面を定義します (存在するディレクトリ、インストールが必要なソフトウェアパッケージ、デプロイが必要なアプリケーションなど)。レシピは、クックブックにパッケージ化されており、通常は 1 つ以上の関連レシピと、設定ファイルを作成するためのテンプレートなどの関連するファイルが含まれています。

このトピックはレシピの非常に基本的な概要であり、シンプルなカスタムIISレイヤーをサポートするクックブックを実装する方法を示すのに十分です。クックブックのより全般的な説明については、「[クックブックとレシピ](#)」を参照してください。Windows 固有のトピックを含む、クックブックの実装方法の詳細なチュートリアルについては、「[クックブック 101](#)」を参照してください。

Chef レシピは技術的には Ruby アプリケーションですが、すべてではないにしても、ほとんどのコードは Chef にあります DSL。は、主に一連のリソース DSL で構成されており、インスタンスの状態の側面を宣言的に指定するために使用できます。例えば、[directory リソース](#)は、システムに追加するディレクトリを定義します。次の例は、指定されたユーザーに属する、フルコントロール権限を持つ C:\data ディレクトリを定義しており、親ディレクトリの権限を継承しません。

```
directory 'C:\data' do
```

```
rights :full_control, 'WORKGROUP\username'
inherits false
action :create
end
```

Chef は、レシピを実行するとき、関連付けられたプロバイダ (インスタンスの状態の変更の詳細を処理する Ruby オブジェクト) にデータを渡すことにより各リソースを実行します。この場合、プロバイダは指定された設定を使用して新しいディレクトリを作成します。

カスタム IIS レイヤーのカスタムクックブックは、次のタスクを実行する必要があります。

- IIS 機能をインストールし、サービスを開始します。

通常は、インスタンスの起動が終了した直後のセットアップ中にこのタスクを実行します。

- この例の簡単な HTML ページであるアプリケーションをインスタンスにデプロイします。

通常は、セットアップ時にこのタスクを実行します。ただし、アプリケーションは通常定期的に更新する必要があるため、インスタンスがオンラインのときに更新をデプロイする必要もあります。

1 つのレシピでこれらのすべてのタスクを実行する必要があります。ただし、セットアップタスクとデプロイタスクに別個のレシピを使用することをお勧めします。このようにして、セットアップコードを実行しなくても、いつでもアプリケーションの更新をデプロイすることができます。次に、カスタム IIS レイヤーをサポートするクックブックを設定する方法について説明します。以降のトピックでは、レシピを実装する方法を示します。

開始するには、以下の手順を実行します。

1. ワークステーション上の適切な場所に、`iis-cookbook` というディレクトリを作成します。
2. `iis-cookbook` に、次のコンテンツを含む `metadata.rb` ファイルを追加します。

```
name "iis-cookbook"
version "0.1.0"
```

この例では、最小限の `metadata.rb` を使用します。このファイルを使用する方法の詳細については、「[metadata.rb](#)」を参照してください。

3. `iis-cookbook` に `recipes` ディレクトリを追加します。

このディレクトリ (名前を `recipes` にする必要があります) には、クックブックのレシピが含まれています。

通常、クックブックには他のさまざまなディレクトリを含めることができます。たとえば、レシピがテンプレートを使用して設定ファイルを作成する場合、テンプレートは通常 `templates/default` ディレクトリに配置されます。この例のクックブックは完全にレシピで構成されているため、他のディレクトリは必要ありません。また、この例では、1つのクックブックを使用しますが、必要な数のクックブックを使用できます。複雑なプロジェクトでは、多くの場合、複数のクックブックを使用することが推奨されます。たとえば、セットアップタスクとデプロイタスクに別個のクックブックを使用することができます。クックブックのその他の例については、「[クックブックとレシピ](#)」を参照してください。

インストールして開始するレシピの実装 IIS

IIS は Windows 機能であり、Windows Server にインストールできるオプションのシステムコンポーネントのセットの 1 つです。recipe は、次のいずれか IIS の方法でインストールできます。

- [powershell_script](#) リソースお使用して [Install-WindowsFeature](#) コマンドレットを実行する。
- Chef [Windows クックブック](#) の `windows_feature` リソースを使用する。

`windows` クックブックには、プロバイダーが [Deployment Image Servicing and Management \(\)](#) を使用して、機能のインストールを含む Windows インスタンスでさまざまなタスクを実行する一連のリソースが含まれています。DISM

Note

`powershell_script` は、特に Windows レシピに役立ちます。PowerShell スクリプトまたはコマンドレットを実行することで、インスタンスでさまざまなタスクを実行できます。特に、Chef リソースによってサポートされていないタスクに役立ちます。

この例では、ウェブサーバー () をインストールして起動する PowerShell スクリプトを実行します IIS。 `windows` クックブックについては後で説明します。 `windows_feature` を使用して をインストールする方法の例については、IIS 「」を参照してください [Windows 機能のインストール: IIS](#)。

以下の内容で `install.rb` という名前のレシピをクックブックの `recipes` ディレクトリに追加します。

```
powershell_script 'Install IIS' do
  code 'Install-WindowsFeature Web-Server'
  not_if "(Get-WindowsFeature -Name Web-Server).Installed"
```

```
end

service 'w3svc' do
  action [:start, :enable]
end
```

レシピには、2つのリソースが含まれています。

powershell_script

`powershell_script` は、指定された PowerShell スクリプトまたはコマンドレットを実行します。例では、以下の属性設定を使用します。

- `code` – 実行する PowerShell コマンドレット。

この例では、ウェブサーバー () `Install-WindowsFeature` をインストールする コマンドレットを実行します IIS。通常、`code` 属性の行数に制限はないため、必要な数のコマンドレットを実行できます。

- `not-if` – recipe [がまだインストールされていない場合にのみインストールされるようにするガード属性](#)。IIS

通常は、レシピを `[idempotent]` (べき等) にするため、同じタスクを複数回実行して時間を無駄にすることがありません。

各リソースには、プロバイダが実行するアクションを指定するアクションがあります。この例には明示的なアクションがないため、プロバイダーは指定された PowerShell スクリプトを実行するデフォルトの `:run` アクションを実行します。詳細については、「[Windows PowerShell スクリプトの実行](#)」を参照してください。

service

この場合、はサービスであるウェブサーバー IIS サービス (W3SVC) `service` を管理します。この例では、デフォルトの属性を使用し、を起動して有効に `:enable` する `:start` との 2 つのアクションを指定します IIS。

Note

などのパッケージインストーラーを使用するソフトウェアをインストールする場合は MSI、`windows_package` リソースを使用できます。詳細については、「[パッケージのインストール](#)」を参照してください。

カスタムクックブックの有効化

AWS OpsWorks スタックは、各インスタンスのローカルキャッシュからレシピを実行します。カスタムレシピを実行するには、以下を実行する必要があります。

- リモートリポジトリにクックブックを保存します。

AWS OpsWorks スタックは、このリポジトリから各インスタンスのローカルキャッシュにクックブックをダウンロードします。

- カスタムクックブックが有効になるようにスタックを編集します。

カスタムクックブックはデフォルトで無効になっているため、スタックのカスタムクックブックを有効にし、リポジトリURLと関連情報を提供する必要があります。

AWS OpsWorks スタックはカスタムクックブックの S3 アーカイブと Git リポジトリをサポートします。この例では S3 アーカイブを使用します。詳細については、「[クックブックリポジトリ](#)」を参照してください。

S3 アーカイブを使用するには

1. .zip ディレクトリの iis-cookbook アーカイブを作成します。

AWS OpsWorks スタックは、Windows スタックの .tgz (gzip 圧縮 tar) アーカイブもサポートしています。

2. アーカイブを米国西部 (カリフォルニア北部) リージョンの S3 バケットにアップロードし、ファイルを公開します。プライベート S3 アーカイブを使用することもできますが、この例ではパブリックアーカイブで十分であり、作業がいくらかシンプルになります。
 - a. にサインインする AWS Management Console で Amazon S3 コンソールを開きます <https://console.aws.amazon.com/s3/>。
 - b. us-west-1 内にバケットがまだ存在しない場合は、[Create Bucket] (バケットを作成する) を選択して、バケットを米国西部 (カリフォルニア北部) リージョン内に作成します。
 - c. バケットリストの中からファイルをアップロードしたいバケットの名前を選択し、[Upload] を選択します。
 - d. [Add Files] を選択します。
 - e. アップロードするアーカイブを選択し、[Open] を選択します。
 - f. [Upload - Select Files and Folders] ダイアログの下部で、[Set Details] を選択します。

- g. [Set Details] ダイアログの下部で、[Set Permissions] を選択します。
- h. [Set Permissions] ダイアログで、[Make everything public] を選択します。
- i. [Set Permissions] ダイアログの下部で、[Start Upload] を選択します。アップロードが終了すると、iis-cookbook.zip ファイルがバケット内に表示されます。
- j. バケットを選択し、選択したバケットの [Properties] タブを選択します。リンクの横にあるアーカイブファイルのを記録URLして後で使用します。

Amazon S3 バケットへのファイルのアップロードの詳細については、「Amazon S3 Console User Guide」(Amazon S3 コンソールユーザーガイド)で「[How Do I Upload Files and Folders to an S3 Bucket?](#) (S3 バケットにファイルとフォルダをアップロードする方法) を参照してください。

Important

ここまでは、ウォークスルーには少し時間しかかかりません。AWS OpsWorks スタックサービス自体は無料です。ただし、Amazon S3 ストレージなど、使用するすべてのAWSリソースに対して料金を支払う必要があります。アーカイブをアップロードするとすぐに料金が発生し始めます。詳細については、「[のAWS料金](#)」を参照してください。

スタックのカスタムクックブックを有効にするには

1. 左 AWS OpsWorks スタックコンソールで、ナビゲーションペインでスタックを選択し、右上のスタック設定を選択します。
2. [Settings] ページの右上にある [Edit] を選択します。
3. [Settings] ページで、[Use custom Chef cookbooks] を [Yes] に設定し、次の情報を入力します。
 - [リポジトリタイプ] - [S3 Archive] (S3 アーカイブ)
 - リポジトリ URL - 前に記録したURLクックブックアーカイブファイルの S3。
4. [Save] を選択してスタック設定を更新します。

AWS OpsWorks スタックは、すべての新しいインスタンスにカスタムクックブックをインストールします。次の点に注意してください。AWS OpsWorks スタックは、オンラインインスタンスにカスタムクックブックを自動的にインストールまたは更新しません。これは、後で説明するように、手動で行うことができます。

ステップ 2.4: IISレイヤーを追加する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

クックブックには、[IIS をインストールして起動するレシピ](#)が 1 つあります。これは、レイヤーを作成し、動作している IIS インスタンスがあることを確認するのに十分です。後で、レイヤーにアプリケーションデプロイ機能を追加します。

レイヤーの作成

まず、スタックにレイヤーを追加します。次に、適切なライフサイクルイベントにカスタムレシピを割り当てることによって、そのレイヤーに機能を追加します。

スタックに IIS レイヤーを追加するには

1. ナビゲーションペインで [Layers] (レイヤー) を選択し、[Add a layer] (レイヤーの追加) を選択します。
2. レイヤーを次のように設定します。
 - [Name] (名前) - **IISExample**
 - [Short name] (短縮名) - **iisexample**

AWS OpsWorks スタックは短縮名を使用して内部的にレイヤーを識別します。短縮名を使用してレシピ内のレイヤーを識別することもできますが、この例はそうしていません。短縮名を指定できますが、小文字の英数字と少数の句読点のみ使用できます。詳細については、「[カスタムレイヤー](#)」を参照してください。

3. [Add Layer] を選択します。

IIS Walkthrough この時点でインスタンスを [追加して起動する](#)場合は、AWS OpsWorks スタックは自動的にクックブックをインストールしますが、`install.rb` は実行されません。インスタンスがオンラインになった後、[Execute Recipes スタックコマンド](#)を使用してレシピを手動で実行でき

ます。ただし、Layer の [ライフサイクルイベント](#) のいずれかにレシピを割り当てる方が適切です。AWS OpsWorks スタックは次に、インスタンスのライフサイクルの適切なポイントでレシピを自動的に実行します。

インスタンスの起動が完了すると IIS すぐにインストールして起動します。これを行うには、`install.rb` をレイヤーの Setup イベントに割り当てます。

ライフサイクルイベントにレシピを割り当てるには

1. ナビゲーションペインで [Layers] (レイヤー) を選択します。
2. IISExample レイヤーのボックスで、レシピ を選択します。
3. 右上の [Edit] を選択します。
4. [Custom Chef Recipes] の [Setup] レシピボックスに「**iis-cookbook::install**」と入力します。

Note

`cookbook-name::recipe-name` を使用してレシピを識別します。このとき、レシピ名の `.rb` サフィックスは省略します。

5. [+] を選択してレシピをレイヤーに追加します。後で簡単に削除できるように、赤色の x がレシピの横に表示されます。
6. [Save] を選択して新しい設定を保存します。カスタム Setup レシピに `iis-cookbook::install` が含まれるようになります。

インスタンスをレイヤーに追加して起動

インスタンスを Layer に追加して起動することにより、レシピを試すことができます。AWS OpsWorks インスタンスが起動を完了するとすぐに、スタックがクックブックを自動的にインストールし、セットアップ中に `install.rb` を実行します。

インスタンスをレイヤーに追加して起動するには

1. 左 AWS OpsWorks スタックナビゲーションペインで、インスタンス を選択します。
2. IISExample レイヤー で、インスタンスの追加 を選択します。
3. 適切なサイズを選択します。この例では、`t2.micro` (または利用可能な最小サイズ) で十分です。

4. [Add Instance] を選択します。デフォルトでは、AWS OpsWorks スタックはレイヤーの短縮名に整数を追加することでインスタンス名を生成するため、インスタンスの名前は iisexample1 にする必要があります。
5. インスタンスのアクション列で開始を選択して、インスタンスを起動します。AWS OpsWorks スタックはインスタンスを起動EC2し、Setup レシピを実行して設定します。この時点でレイヤーに Deploy レシピがある場合、AWS OpsWorks スタックは、セットアップレシピが完了した後にそれらを実行します。

プロセスには数分かかる場合があります、その間 [Status] 列には一連のステータスが表示されます。ステータスが [online] になると、設定プロセスが完了し、インスタンスが使用可能になります。

IIS がインストール済みで実行中であることを確認します。

RDP を使用してインスタンスに接続し、Setup レシピが正しく動作したことを確認できます。

IIS がインストールされ、実行されていることを確認するには

1. ナビゲーションペインでインスタンスを選択し、iisexample1 インスタンスのアクション列で rdp を選択します。AWS OpsWorks スタックは、指定した期間後に期限切れになるRDPパスワードを自動的に生成します。
2. [Session valid for] を 2 時間に設定し、[Generate Password] を選択します。
3. AWS OpsWorks スタックにはパスワードと、インスタンスのパブリックDNS名とユーザー名が表示されます。3 つすべてをコピーして、[Acknowledge and close] をクリックします。
4. RDP クライアントを開き、ステップ 3 のデータを使用してインスタンスに接続します。
5. インスタンスで、Windows エクスプローラを開いて C: ドライブを調べます。IIS インストールによって作成された C:\inetpub ディレクトリが必要です。
6. [コントロール パネル] の [管理ツール] アプリケーションを開き、[サービス] を開きます。リストの下部近くに IIS サービスが表示されます。「World Wide Web Publishing サービス」という名前前で、ステータスは [実行中] になっています。
7. に戻る AWS OpsWorks スタックコンソールで、iisexample1 インスタンスのパブリック IP アドレスを選択します。必ず でこれを行ってください。AWS OpsWorks Amazon EC2コンソールではなく スタック。これにより、アドレスにHTTPリクエストが自動的に送信され、デフォルトのIISようこそページが開きます。

次のトピックでは、この例のシンプルな静的HTMLページであるインスタンスにアプリケーションをデプロイする方法について説明します。ただし、休憩したい場合は、の [iisexample1] インスタンス

の [Actions] (アクション) 列で [stop] (停止) を選択すると、インスタンスが停止し、不要な料金が発生するのを防ぐことができます。続行する準備が整ったら、インスタンスを再起動します。

ステップ 2.5: アプリケーションをデプロイする

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

IIS インストールにより、アプリケーションのコードと関連ファイルの C:\inetpub\wwwroot ディレクトリが作成されます。次のステップとして、そのディレクトリにアプリケーションをインストールします。この例では、静的 HTML ホームページ を default.html にインストールします C:\inetpub\wwwroot。一般的なアプローチを簡単に拡張して、アプリケーションなど、より複雑なシナリオに対応できます ASP.NET

アプリケーションのファイルをクックブックに含めて、install.rb がそれらのファイルを C:\inetpub\wwwroot にコピーするようにすることができます。これを行う方法の例については、「[例 6: ファイルの作成](#)」を参照してください。ただし、このアプローチは柔軟性や効率性があまり高くないため、通常はクックブック開発をアプリケーション開発と分割することをお勧めします。

推奨される解決策は、クックブックリポジトリだけでなく、任意のリポジトリであるリポジトリからアプリケーションのコードと関連ファイルを取得し、各 IIS サーバーインスタンスにインストールする個別のデプロイレシピを実装することです。このアプローチにより、アプリケーション開発からクックブック開発が分割されるため、アプリケーションを更新する必要があるときは、クックブックを更新しなくてもデプロイレシピだけを再度実行することができます。

このトピックでは、default.html IIS サーバーにデプロイするシンプルなデプロイレシピを実装する方法を示します。この例は、より複雑なアプリケーションに簡単に拡張できます。

トピック

- [アプリケーションの作成とリポジトリへの保存](#)
- [レシピの実装によるアプリケーションのデプロイ](#)
- [インスタンスのクックブックの更新](#)

- [レシピをカスタムIISレイヤーに追加する](#)
- [アプリケーションの追加](#)
- [アプリケーションのデプロイとアプリケーションの実行](#)

アプリケーションの作成とリポジトリへの保存

アプリケーションにはどのリポジトリでも使用できます。わかりやすいように、この例では default.htm をパブリック S3 バケットに保存します。

アプリケーションを作成するには

1. ワークステーション上の適切な場所に、iis-application というディレクトリを作成します。
2. iis-application に、次のコンテンツを含む default.htm ファイルを追加します。

```
<!DOCTYPE html>
<html>
  <head>
    <title>IIS Example</title>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

3. [S3 バケットを作成し、バケット default.htm にアップロード](#)して、URL 後で使用するために記録します。わかりやすいよう、[ファイルを公開](#)します。

Note

これは、とてもシンプルなアプリケーションですが、基本的な原則を拡張して本稼働レベルのアプリケーションを扱うことができます。

- 複数のファイルを含むより複雑なアプリケーションの場合、通常は iis-application の .zip アーカイブを作成し、S3 バケットにアップロードした方がシンプルです。

アップロード後には .zip ファイルをダウンロードし、内容を適切なディレクトリに展開できます。複数のファイルをダウンロードしたり、ディレクトリ構造を作成したりする必要はありません。

- 本稼働アプリケーションの場合、ファイルを秘密にすることができます。レシピがプライベート S3 バケットからファイルをダウンロードできるようにする方法の例は、「[スタックの Windows インスタンスでの Ruby AWS OpsWorks SDK用の使用](#)」を参照してください。
- アプリケーションは、任意の適切なリポジトリに保存できます。

通常、リポジトリのパブリックを使用してアプリケーションをダウンロードします API。この例では、Amazon S3 を使用します API。例えば、アプリケーションを に保存する場合は GitHub、 を使用できます [GitHub API](#)。

レシピの実装によるアプリケーションのデプロイ

以下のコンテンツを含む `deploy.rb` という名前のレシピを `iis-cookbook recipes` ディレクトリに追加します。

```
chef_gem "aws-sdk-s3" do
  compile_time false
  action :install
end

ruby_block "download-object" do
  block do
    require 'aws-sdk-s3'

    #1
    # Aws.config[:ssl_ca_bundle] = 'C:\ProgramData\Git\bin\curl-ca-bundle.crt'
    Aws.use_bundled_cert!

    #2
    query = Chef::Search::Query.new
    app = query.search(:aws_opsworks_app, "type:other").first
    s3region = app[0][:environment][:S3REGION]
    s3bucket = app[0][:environment][:BUCKET]
    s3filename = app[0][:environment][:FILENAME]

    #3
    s3_client = Aws::S3::Client.new(region: s3region)
    s3_client.get_object(bucket: s3bucket,
                        key: s3filename,
                        response_target: 'C:\inetpub\wwwroot\default.htm')
```

```
end
  action :run
end
```

この例では、[SDKRuby v2 の](#) を使用して ファイルをダウンロードします。ただし、AWS OpsWorks スタックはこれを Windows インスタンスSDKにインストールしないため、レシピはそのタスクを処理する [chef_gem](#) リソースで始まります。

Note

`chef_gem` リソースは、`gem` を Chef の専用 Ruby バージョン (そのレシピが使用するバージョン) にインストールします。システム全体の Ruby バージョンに `gem` をインストールする場合、[gem_package](#) リソースを使用します。

recipe の大部分は[ruby_block](#)リソースであり、Ruby の を使用して をダウンロードする Ruby SDK コードのブロックを実行しますdefault.htm。ruby_block 内のコードは以下のセクションに分割できます。それぞれ、コード例の番号付きコメントに対応します。

1: 証明書バンドルの指定

Amazon S3 は を使用するためSSL、S3 バケットからオブジェクトをダウンロードするには適切な証明書が必要です。SDK for Ruby v2 には証明書バンドルが含まれていないため、証明書バンドルを指定し、Ruby SDKが証明書バンドルを使用するには を設定する必要があります。AWS OpsWorks スタックは、証明書バンドルを直接インストールしませんが、証明書バンドル (`curl-ca-bundle.crt`) が含まれている Git はインストールします。便宜上、この例では、SDKの Git 証明書バンドルを使用するように Ruby の を設定しますSSL。独自のバンドルをインストールし、SDKそれに応じて を設定することもできます。

2: リポジトリデータの取得

Amazon S3 からオブジェクトをダウンロードするには、AWSリージョン、バケット名、およびキー名が必要です。後で説明するように、この例では、一連の環境変数をアプリと関連付けることでこの情報を提供します。アプリケーションをデプロイすると、AWS OpsWorks スタックは、インスタンスのノードオブジェクトに一連の属性を追加します。これらの属性は本質的に、アプリケーションの設定 (環境変数など) を含むハッシュテーブルです。このアプリケーションのアプリケーション属性は、JSON 形式の次のようになります。

```
{
```

```
"app_id": "8f71a9b5-de7f-451c-8505-3f35086e5bb3",
"app_source": {
  "password": null,
  "revision": null,
  "ssh_key": null,
  "type": "other",
  "url": null,
  "user": null
},
"attributes": {
  "auto_bundle_on_deploy": true,
  "aws_flow_ruby_settings": {},
  "document_root": null,
  "rails_env": null
},
"data_sources": [{"type": "None"}],
"domains": ["iis_example_app"],
"enable_ssl": false,
"environment": {
  "S3REGION": "us-west-2",
  "BUCKET": "windows-example-app",
  "FILENAME": "default.htm"
},
"name": "IIS-Example-App",
"shortname": "iis_example_app",
"ssl_configuration": {
  "certificate": null,
  "private_key": null,
  "chain": null
},
"type": "other",
"deploy": true
}
```

アプリケーションの環境変数は、[:environment] 属性に保存されます。環境変数を取得するには、Chef 検索クエリを使用してアプリケーションのハッシュテーブルを取得します。このテーブルは、aws_opsworks_app ノード配下にあります。このアプリは、other タイプとして定義されるため、クエリはそのタイプのアプリケーションを検索します。レシピは、このインスタンスにはアプリケーションが 1 つしかないという事実を利用しているため、対象となるハッシュテーブルは app[0] だけです。便宜上の理由で、レシピはその後リージョン、バケット、ファイル名を変数に割り当てます。

Chef 検索の使用方法的詳細については、「[Chef の検索での属性値の取得](#)」を参照してください。

3: ファイルのダウンロード

レシピの 3 番目の部分では、[S3 クライアントオブジェクト](#)が作成され、その `get_object` メソッドを使用して `default.htm` がインスタンスの `C:\inetpub\wwwroot` ディレクトリにダウンロードされます。

Note

レシピは Ruby アプリケーションであるため、Ruby コードは必ずしも `ruby_block` に存在する必要はありません。ただし、レシピの本文のコードが最初に実行され、その後リソースが順番に実行されます。この例では、レシピ本文にダウンロードコードを配置すると、`chef_gem` リソースが Ruby SDK用の をまだインストールしていないため失敗します。`ruby_block` リソース内のコードは、リソースが Ruby SDKの をインストールした後、`chef_gem` リソースの実行時に実行されます。

インスタンスのクックブックの更新

AWS OpsWorks スタックは、新しいインスタンスにカスタムクックブックを自動的にインストールします。ただし、既存のインスタンスを使用しているため、クックブックを手動で更新する必要があります。

インスタンスのクックブックを更新するには

1. `iis-cookbook` の `.zip` アーカイブを作成し、S3 バケットにアップロードします。

これにより既存のクックブックが上書きされますが、同じURLままだため、スタック設定を更新する必要はありません。

2. インスタンスがオンラインでない場合は再起動してください。
3. インスタンスがオンラインになったら、ナビゲーションペインで [Stack] を選択し、[Run Command] を選択します。
4. [Command] で、[\[Update Custom Cookbooks\]](#) を選択します。このコマンドを実行すると、更新されたクックブックがインスタンスにインストールされます。
5. [\[Update Custom Cookbooks\]](#) を選択します。コマンドの実行完了までには数分かかることがあります。

レシピをカスタムIISレイヤーに追加する

`install.rb` と同様、デプロイを処理するには、`deploy.rb` を適切なライフサイクルイベントに割り当てることをお勧めします。通常は、Deploy イベントにデプロイレシピを割り当てます。それらは、まとめて Deploy レシピと呼ばれます。レシピを Deploy イベントに割り当ててもイベントはトリガーされません。代わりに、次のようになります。

- 新しいインスタンスの場合、AWS OpsWorks スタックは Setup レシピの完了後、Deploy レシピを自動的に実行するため、新しいインスタンスは自動的に現在のアプリケーションバージョンになります。
- オンラインインスタンスでは、[デプロイコマンド](#)を使用して新しいアプリケーションまたは更新されたアプリケーションを手動でインストールします。

このコマンドはスタックのインスタンスで Deploy イベントをトリガーし、これによって Deploy レシピが実行されます。

レイヤーの Deploy イベントに `deploy.rb` を割り当てるには

1. ナビゲーションペインで「レイヤー」を選択し、「レイヤーIISExample」の下にある「レシピ」を選択します。
2. [Custom Chef Recipes] で、[`iis-cookbook::deploy`] を [デプロイ] レシピボックスに追加し、[+] を選択してレシピをレイヤーに追加します。
3. [Save] を選択して新しい設定を保存します。カスタム Deploy レシピに `iis-cookbook::deploy` が含まれるようになります。

アプリケーションの追加

最後のタスクは、スタックにアプリケーションを追加して、内のアプリケーションを表すことです。AWS OpsWorks スタック環境。アプリケーションには、アプリケーションの表示名などのメタデータと、リポジトリからアプリケーションをダウンロードするのに必要なデータが含まれます。

アプリケーションをスタックに追加するには

1. ナビゲーションペインで [Apps] を選択し、[Add an app] を選択します。
2. 以下の設定を使用してアプリケーションを設定します。
 - [Name] (名前) - **IIS-Example-App**
 - [Repository Type] (リポジトリタイプ) - [Other] (その他)

- [Environment Variables] (環境変数) 以下の 3 つの環境変数を追加します。
 - **S3REGION** - バケットのリージョン (この場合は us-west-1)。
 - **BUCKET** - バケット名 (windows-example-app など)。
 - **FILENAME** - ファイル名 (**default.htm**)。
3. 残りの設定はデフォルト値をそのまま使い、[Add App] を選択してアプリケーションをスタックに追加します。

Note

この例では、ダウンロードデータを提供するために環境変数を使用します。別の方法は、S3 Archive リポジトリタイプを使用し、ファイルの を提供することですURL。AWS OpsWorks スタックは、AWS認証情報などのオプションデータとともに情報をアプリケーションの `app_source` 属性に追加します。デプロイレシピは、アプリケーション属性URLから を取得し、それを解析してリージョン、バケット名、ファイル名を抽出する必要があります。

アプリケーションのデプロイとアプリケーションの実行

AWS OpsWorks スタックは、アプリケーションを自動的に新しいインスタンスにデプロイしますが、オンラインインスタンスにはデプロイしません。インスタンスは既に実行されているため、アプリケーションを手動でデプロイする必要があります。

アプリケーションをデプロイするには

1. ナビゲーションペインで「アプリケーション」を選択し、続いてアプリの「アクション」列で「デプロイ」を選択します。
2. [Command] は [Deploy] に設されていない必要があります。[Deploy App] (デプロイアプリケーション) ページの右下で [Deploy] (デプロイ) を選択します。コマンドの実行完了までには数分かかることがあります。

デプロイメントが完了したら、[Apps] ページに戻ります。[Status] インジケータには緑で [successful] と表示され、アプリケーション名の横には、デプロイメントの正常終了を示す緑のチェックマークが表示されます。

Note

Windows アプリケーションは常に [Other] アプリケーションタイプのため、アプリケーションをデプロイすると次の処理が実行されます。

- 前述のとおり、アプリケーションのデータが、[スタック設定とデプロイ属性](#)に追加されます。
- カスタム Deploy レシピを実行する Deploy イベントがスタックのインスタンスでトリガーされます。

Note

失敗したデプロイまたはアプリケーションをトラブルシューティングする方法についての詳細は、[レシピのデバッグ](#) を参照してください。

アプリケーションがインストールされます。[Navigation] (ナビゲーション) ペインで [Instances] (インスタンス) を選択し、続いてインスタンスのパブリック IP アドレスを選択することで開くことができます。これにより、インスタンスに HTTP リクエストが送信され、ブラウザに次のような内容が表示されます。

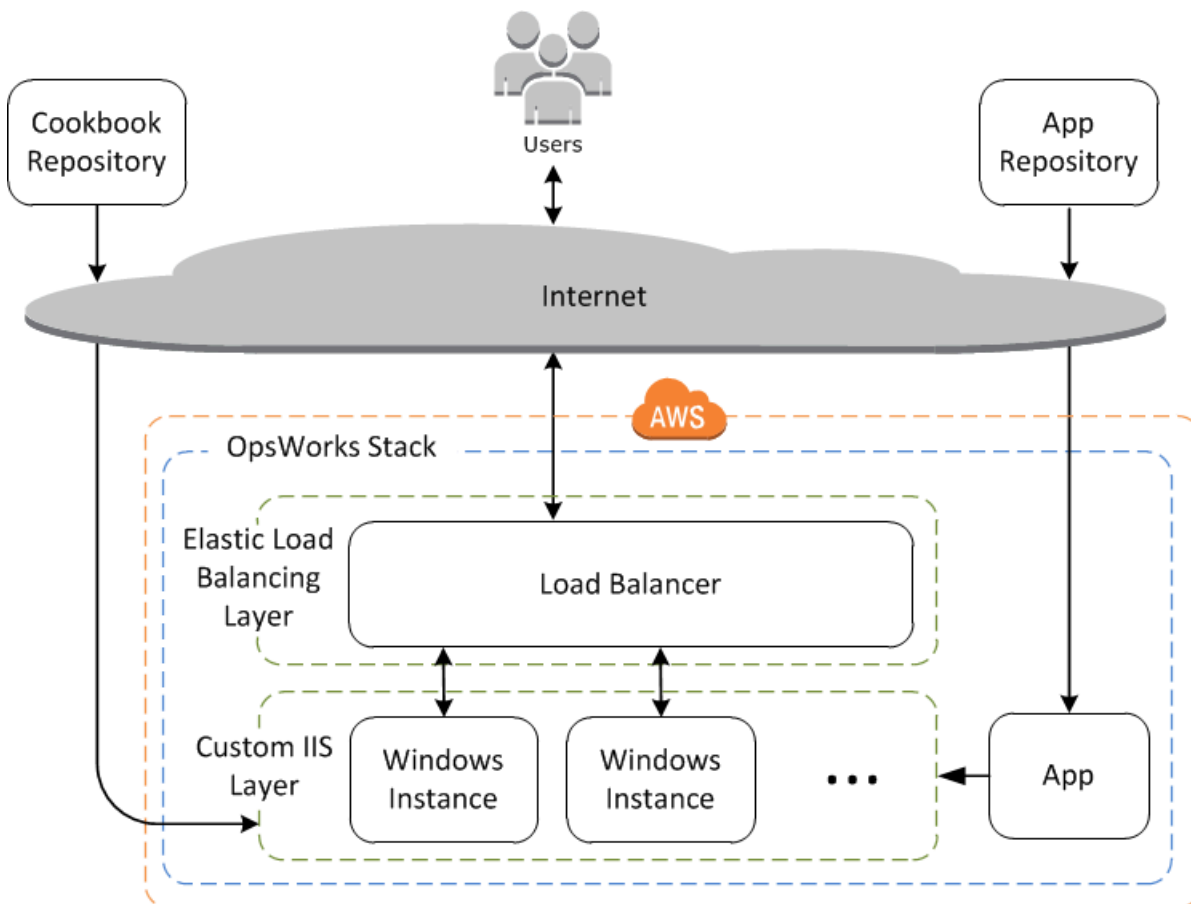
Hello World!

ステップ 3: スケールアウトする IISExample

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

受信ユーザーリクエストが単一の t2.micro インスタンスで処理可能な制限に近づいている場合、サーバー容量を増やす必要があります。大きいインスタンスに移行できますが、制限があります。インスタンスをスタックに追加し、ロードバランサーの背後に配置すると、柔軟性が向上します。基本的なアーキテクチャは次のようになります。



とりわけ、このアプローチには、単一の大きいインスタンスより堅牢性がかなり高いというメリットがあります。

- いずれかのインスタンスが失敗した場合、ロードバランサーは受信リクエストを残りのインスタンスに分散し、アプリケーションは動作し続けます。
- インスタンスを異なるアベイラビリティーゾーンに置いた場合 (推奨される方法)、アベイラビリティーゾーンで問題が発生してもアプリケーションが動作し続けます。

AWS OpsWorks スタックを使用すると、スタックを簡単にスケールアウトできます。このセクションでは、2 つ目の 24/7 PHP App Server インスタンスを に追加IISExampleし、両方のインスタンスを Elastic Load Balancing ロードバランサーの背後に配置することで、スタックをスケールアウトする方法の基本について説明します。プロシージャを簡単に拡張して任意の数の 24/7 インスタ

ンスを追加することも、時間ベースのインスタンスを使用して追加することもできます。AWS OpsWorks スタックはスタックを自動的にスケールリングします。詳細については、「[時間ベースおよび負荷ベースのインスタンスによる負荷の管理](#)」を参照してください。

ロードバランサーの追加

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

Elastic Load Balancing は、受信アプリケーショントラフィックを複数の Amazon EC2 インスタンスに自動的に分散する AWS サービスです。ロードバランサーで 2 つの目的を達成できます。明らかな目的は、アプリケーションサーバーで負荷を均等化することです。多くのサイトでは、ユーザーがアプリケーションサーバーおよびデータベースに直接アクセスできないようにすることが求められます。Elastic Load Balancing は、トラフィックの分散の他に、以下も行実行します。

- 異常な Amazon EC2 インスタンスを検出します。

問題のあるインスタンスが復旧するまで、トラフィックのルートを残りの正常なインスタンスに変更します。

- 受信トラフィックに応じて、自動的にそのリクエスト処理能力を拡張します。

Note

AWS OpsWorks スタックは Application Load Balancer をサポートしていません。Classic Load Balancer は でのみ使用できます。AWS OpsWorks スタック。

多くの場合、Elastic Load Balancing はレイヤーと呼ばれますが、他の組み込みレイヤーとは動作が多少異なります。レイヤーを作成してインスタンスを追加する代わりに、Amazon EC2 コンソールを使用して Elastic Load Balancing ロードバランサーを作成し、既存のレイヤーの 1 つ、通常はアプリケーションサーバーレイヤーにアタッチします。AWS OpsWorks スタックは次に、レイヤーの既存

のインスタンスをサービスに登録し、すべての新しいインスタンスを自動的に追加します。次の手順では、ロードバランサーを追加する方法について説明します。

ロードバランサーをカスタムIISレイヤーにアタッチするには

1. Amazon EC2コンソールを使用して、の新しいロードバランサーを作成しますIISExample。詳細については、「[Elastic Load Balancing の開始方法](#)」を参照してください。[Create Load Balancer] ウィザードを実行するときに、ロードバランサーを次のように設定します。

1: ロードバランサーの定義

ロードバランサーに IIS-LB などのわかりやすい名前を割り当てて、での検索を容易にします。AWS OpsWorks スタックコンソール。残りの設定のデフォルト値を受け入れ、[Next: Assign Security Groups] を選択します。

2: セキュリティグループの割り当て

アカウントがデフォルトのをサポートしている場合VPC、ウィザードはこのページを表示してロードバランサーのセキュリティグループを決定します。EC2 Classic ではこのページは表示されません。

このチュートリアルでは、デフォルトのVPCセキュリティグループを指定し、次へ: セキュリティ設定 を選択します。

3: セキュリティ設定の構成

このチュートリアルでは、ロードバランサーが安全なリスナー (つまり、HTTPSフロントエンド接続SSLで) を使用する必要があるため、次へ: ヘルスチェックを設定を選択して続行します。

4: ヘルスチェックの設定

ping のパスを / に設定します。残りの設定のデフォルトを受け入れ、次へ: EC2 インスタンスの追加 を選択します。

5: EC2 インスタンスを追加する

AWS OpsWorks スタックは、ロードバランサーへのインスタンスの登録を自動的に処理します。[Next Add Tags] を選択して続行します。

6: タグの追加

この例ではタグを使用しません。[Review and Create] を選択します。

7: 確認

選択を確認して [Create] を選択し、[Close] を選択すると、ロードバランサーが起動します。

2. アカウントがデフォルトのをサポートしている場合VPC、ロードバランサーを起動した後、セキュリティグループに適切なインバウンドルールがあることを確認する必要があります。デフォルトルールでは、着信トラフィックは受け入れられません。
 1. Amazon EC2ナビゲーションペインでセキュリティグループを選択します。
 2. デフォルトのVPCセキュリティグループを選択する
 3. [インバウンド] タブで、[編集] を選択します。
 4. このウォークスルーでは、[Source] を [Anywhere] に設定します。これにより、ロードバランサーでは任意の IP アドレスからの着信トラフィックを受け入れます。
 5. [Save] (保存) をクリックします。
3. に戻る AWS OpsWorks スタックコンソール。[Layers] (レイヤー) ページで、[Network] (ネットワーク) を選択します。
4. Elastic Load Balancingで、ステップ 1 で作成した IIS-LB ロードバランサーを選択し、保存 をクリックします。

ロードバランサーをレイヤーにアタッチしたら、AWS OpsWorks スタックは、レイヤーの現在のインスタンスを自動的に登録し、オンラインになると新しいインスタンスを追加します。

5. [Layers] (レイヤー) ページで、ロードバランサーの名前をクリックして詳細ページを開きます。ロードバランサーページのインスタンスの横のチェックマークは、インスタンスがヘルスチェックに合格したことを示します。

ロードバランサーにリクエストを送信することで、IIS-Example-App を実行できるようになりました。

ロードバランサーを介して IIS-Example-App を実行するには

1. [Layers] (レイヤー) を選択します。IIS-ELB ロードバランサーはレイヤーとしてリストされ、ヘルス列には正常なインスタンスを示す緑色のインスタンスが 1 つある必要があります。
2. 実行するロードバランサーDNSの名前 IIS-Example-App を選択します。これは、ロードバランサーの名前の下に一覧表示され、IIS-LB-1802910859.us-west-2.elb.amazonaws.com のように表示されます。ロードバランサーがインスタンスにリクエストを転送すると、レスポンスが返されます。これは、インスタンスのパブリック IP アドレスをクリックした場合と同じレスポンスになります。

この時点ではインスタンスが 1 個しかいないため、ロードバランサーがあっても実際にはそれほど変わりません。ただし、レイヤーにインスタンスを追加できます。

インスタンスをレイヤーに追加するには

1. [Instances] を選択し、[+ instance] を選択して別のインスタンスをレイヤーに追加します。
2. インスタンスを起動します。

新しいインスタンスであるため、AWS OpsWorks スタックは、現在のカスタムクックブックを自動的にインストールし、セットアップ中に現在のアプリケーションバージョンをデプロイします。インスタンスがオンラインになると、AWS OpsWorks スタックによってロードバランサーに自動的に追加されるため、インスタンスはすぐにリクエストの処理を開始します。アプリケーションがまだ動作していることを確認するには、ロードバランサーDNSの名前を再度選択します。

次のステップ

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

このウォークスルーでは、シンプルな Windows アプリケーションサーバースタックのセットアップの基本について説明しました。ここでは、次のステップに関する推奨事項をいくつか示します。

- 詳細を知りたい場合は、クックブックの実装に関するチュートリアル [使用開始: クックブック](#) を紹介、いくつかの AWS OpsWorks スタック固有の例。
- [Amazon Relational Database Service \(Amazon RDS\) レイヤー](#) をスタックに追加して、バックエンドデータベースサーバーとして使用できます。アプリケーションをデータベースに接続する方法については、「[カスタムレシピの使用](#)」を参照してください。

でのクックブックの開始方法 AWS OpsWorks スタック

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

本番稼働用レベル AWS OpsWorks スタックスタックには通常、ある程度のカスタマイズが必要で、多くの場合、カスタム Chef クックブックを実装することを意味します。クックブックは、レシピと呼ばれる手順を含む設定情報が含まれているパッケージファイルです。レシピは、Ruby 言語構文で書かれ、使用するリソースとそれらのリソースを適用する順序を指定する、1 つ以上の指示のセットです。Chef で使用されるリソースは、設定ポリシーのステートメントです。このチュートリアルでは、用の Chef クックブックの実装に関する基本的な概要を説明します。AWS OpsWorks スタック。Chef、クックブック、レシピ、およびリソースの詳細については、「[次のステップ](#)」のリンクを参照してください。

このウォークスルーでは、主にユーザー独自のクックブックを作成する方法について説明します。[Chef Supermarket](#) のようなウェブサイトで購入できるコミュニティ提供のクックブックを使用することもできます。コミュニティクックブックの使用を開始できるようにするため、このウォークスルーでは Chef Supermarket からのコミュニティクックブックを使用する手順が後述されています。

このウォークスルーを開始する前にいくつかの設定ステップを完了します。すでにこの章の他のウォークスルー（「[使用開始: サンプル](#)」など）を完了している場合、このウォークスルーの前提条件を満たしており、省略して「[このウォークスルーの開始](#)」に進むことができます。それ以外の場合、必ず[前提条件](#)を満たしてからこのウォークスルーに戻ってください。

トピック

- [ステップ 1: クックブックを作成する](#)
- [ステップ 2: スタックとそのコンポーネントを作成する](#)
- [ステップ 3: レシピを実行し、テストする](#)
- [ステップ 4: クックブックを更新してパッケージをインストールする](#)
- [ステップ 5: インスタンスのクックブックを更新し、レシピを実行する](#)

- [ステップ 6: クックブックを更新してユーザーを追加する](#)
- [ステップ 7: クックブックを更新してディレクトリを作成する](#)
- [ステップ 8: クックブックを更新してファイルを作成し、コピーする](#)
- [ステップ 9: クックブックを更新してコマンドを実行する](#)
- [ステップ 10: クックブックを更新してスクリプトを実行する](#)
- [ステップ 11: クックブックを更新してサービスを管理する](#)
- [ステップ 12: カスタムを使用するようにクックブックを更新する JSON](#)
- [ステップ 13: クックブックを更新してデータバッグを使用する](#)
- [ステップ 14: クックブックを更新して繰り返しを使用する](#)
- [ステップ 15: クックブックを更新して条件付きロジックを使用する](#)
- [ステップ 16: クックブックを更新してコミュニティクックブックを使用する](#)
- [ステップ 17: \(オプション\) クリーンアップする](#)
- [次のステップ](#)

ステップ 1: クックブックを作成する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

クックブックの作成から開始します。ここで作成するクックブックは、実務で使い始めるのに十分なものではなく、このウォークスルーにおける残りの作業の基礎となるものです。

Note

このステップでは、クックブックを手動で作成する方法を示します。ローカルのワークステーションでコマンド [chef generate cookbook](#) を実行することにより、Chef 開発キット ([Chef DK](#)) を使用して、より短時間でクックブックを作成することができます。ただし、こ

のコマンドでは、このウォークスルーで必要ではないフォルダとファイルがいくつか作成されます。

クックブックを作成するには

1. ローカルワークステーションで、`opsworks_cookbook_demo` という名前のディレクトリを作成します (別の名前を使用することはできますが、必ずこのウォークスルー全体でこれを `opsworks_cookbook_demo` に置き換えてください)。
2. `opsworks_cookbook_demo` ディレクトリで、テキストエディタを使い `metadata.rb` という名前のファイルを作成します。クックブックの名前を指定するには下記のコードを追加します。`metadata.rb` の詳細については、Chef ウェブサイトの [metadata.rb](#) を参照してください。

```
name "opsworks_cookbook_demo"
```

3. `opsworks_cookbook_demo` ディレクトリに `recipes` という名前のファイルを作成します。このサブディレクトリには、このウォークスルーのクックブック用に作成するすべてのレシピが含まれます。
4. `recipes` ディレクトリに `default.rb` という名前のファイルを作成します。このファイルにはファイルと同じ名前のレシピが含まれますが、ファイル拡張子がなく、`default` となります。次の 1 行のコードを、`default.rb` ファイルに追加します。このコードは、レシピの実行時にログにシンプルなメッセージを表示する 1 行のレシピです。

```
Chef::Log.info("***** Hello, World! *****")
```

5. ターミナルまたはコマンドプロンプトで、`tar` コマンドを使用して `opsworks_cookbook_demo.tar.gz` というファイルを作成します。これには、`opsworks_cookbook_demo` ディレクトリとそのコンテンツが含まれます。例えば:

```
tar -czvf opsworks_cookbook_demo.tar.gz opsworks_cookbook_demo/
```

別のファイル名を使用することはできますが、必ずこのウォークスルー全体でこれを `opsworks_cookbook_demo.tar.gz` に置き換えてください。

Note

Windows で tar ファイルを作成する場合、最上位ディレクトリはクックブックの親ディレクトリでなければなりません。このウォークスルーは、tar パッケージによって提供される tar コマンドを使用して Linux で、[\[Git Bash\]](#) で提供される tar コマンドを使って Windows でそれぞれテスト済みです。他のコマンドやプログラムを使用して圧縮TARされた (.tar.gz) ファイルを作成すると、期待どおりに動作しない場合があります。

6. S3 バケットを作成するか、既存のバケットを使用します。詳細については、「[バケットの作成](#)」を参照してください。
7. `opsworks_cookbook_demo.tar.gz` ファイルを S3 バケットにアップロードします。詳細については、「[バケットへのオブジェクトの追加](#)」を参照してください。

これで、このウォークスルー全体で使用するクックブックが作成されました。

[次のステップ](#)では、`opsworks_cookbook_demo` を作成します。AWS OpsWorks 後でクックブックをアップロードし、クックブックのレシピを実行するために使用するスタック。

ステップ 2: スタックとそのコンポーネントを作成する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

を作成する AWS OpsWorks スタックは、レイヤーとインスタンスを含むスタックとそのコンポーネントです。後のステップでは、インスタンスにクックブックをアップロードし、そのインスタンスでクックブックのレシピを実行します。

スタックを作成するには

1. にサインインする AWS OpsWorks <https://console.aws.amazon.com/opsworks> のスタックコンソール。
2. 該当する場合、次のいずれかを実行します。
 - へようこそ AWS OpsWorks スタックページが表示されたら、最初のスタックの追加または最初のスタックの追加を選択します。AWS OpsWorks スタックスタック (両方の選択肢が同じことを行います)。[Add stack] ページが表示されます。
 - OpsWorks ダッシュボードページが表示されている場合は、スタックの追加 を選択します。[Add Stack] ページが表示されます。
3. [Chef 12 stack] を選択します。
4. [Stack name] ボックスに、スタック名 (たとえば **MyCookbooksDemoStack**) を入力します。別の名前を入力することはできますが、必ずこのウォークスルー全体でこれを MyCookbooksDemoStack に置き換えてください。
5. [リージョン] で [米国西部 (オレゴン)] を選択します。
6. の場合はVPC、次のいずれかを実行します。
 - VPC が利用可能な場合は、それを選択します。詳細については、「[でのスタックの実行 VPC](#)」を参照してください。
 - それ以外の場合は [いいえVPC] を選択します。
7. [Use custom Chef cookbooks] で、[Yes] を選択します。
8. [Repository type] で、[S3 Archive] を選択します。

Note

[入門ガイド: Linux](#) のウォークスルーでは、[Http Archive] を選択しました。ここでは、必ず代わりに [S3 Archive] を選択します。

9. リポジトリにURL、S3 の `opsworks_cookbook_demo.tar.gz` ファイルへのパスを入力します。パスを取得するには、S3 コンソールで、`opsworks_cookbook_demo.tar.gz` ファイルを選択します。[Properties] ペインで、[Link] フィールドの値をコピーします (これに似たものになるはずですが、`https://s3.amazonaws.com/amzn-s3-demo-bucket/opsworks_cookbook_demo.tar.gz`)
10. S3 バケットがデフォルトであるプライベートの場合は、アクセスキー ID とシークレットアクセスキーに、このチュートリアルで使用しているIAMユーザーのアクセスキー ID とシークレ

トアクセスキーを入力します。詳細については、「[オブジェクトに対するアクセス許可を編集する](#)」および「[他ユーザーとのオブジェクトの共有](#)」を参照してください。

11. 以下はデフォルト値のままにします。

- [Default Availability Zone] (us-west-2a)
- Default operating system (デフォルトオペレーティングシステム) (Linux および Amazon Linux 2016.09)
- デフォルトSSHキー (デフォルトSSHキー は使用しないでください)
- [Stack color] (dark blue)

12. [Advanced] (アドバンスド) を選択します。

13. IAM ロール の場合、次のいずれかを実行します。

- aws-opsworks-service-role が利用可能な場合は、それを選択します。
- aws-opsworks-service-role が使用できない場合は、新しいIAMロール を選択します。

14. デフォルトのIAMインスタンスプロファイル の場合、次のいずれかを実行します。

- aws-opsworks-ec2 ロールが使用可能な場合は、それを選択します。
- aws-opsworks-ec2 ロールが使用できない場合は、新しいIAMインスタンスプロファイル を選択します。

15. 以下はデフォルト値のままにします。

- デフォルトのルートデバイスタイプ (EBS がサポート)
- [Hostname theme] (Layer Dependent)
- OpsWorks エージェントバージョン (最新バージョン)
- カスタム Chef JSON (空白)
- セキュリティ、 OpsWorks セキュリティグループを使用する (はい)

16. [Add stack] を選択します。AWS OpsWorks スタックはスタックを作成し、MyCookbooksDemoStackページを表示します。

レイヤーを作成するには

1. サービスナビゲーションペインで、[Layers] (レイヤー) を選択します。[Layers] (レイヤー) ページが表示されます。
2. [Add a layer] (レイヤーを追加) を選択します。

3. OpsWorks タブの名前に と入力します**MyCookbooksDemoLayer**。別の名前を入力することはできませんが、必ずこのウォークスルー全体でこれを MyCookbooksDemoLayer に置き換えてください。
4. [Short name] に、「**cookbooks-demo**」と入力します。別の名前を入力することはできませんが、必ずこのウォークスルー全体でこれを cookbooks-demo に置き換えてください。
5. [Add layer] を選択します。AWS OpsWorks スタックによりレイヤーが追加され、[レイヤー] ページが表示されます。

インスタンスを作成して起動するには

1. サービスのナビゲーションペインで、[Instances] を選択します。[Instances] ページが表示されます。
2. [Add an instance] を選択します。
3. [New] タブの [Advanced] を選択します。
4. 以下はデフォルト値のままにします。
 - [Hostname] (cookbooks-demo1)
 - [Size] (c3.large)
 - サブネット (*IP address* us-west-2a)
 - Scaling type (24/7)
 - SSH キー (デフォルトSSHキー は使用しないでください)
 - オペレーティングシステム: (Amazon Linux 2016.09)
 - OpsWorks エージェントバージョン (スタック から継承)
 - テナンシー (デフォルト - VPC設定に依存します)
 - ルートデバイスタイプ (EBS backed)
 - ボリュームタイプ (汎用 (SSD))
 - [Volume size] (8)
5. [Add instance] を選択します。
6. ではMyCookbooksDemoLayer、クックブック-demo1 では、アクション では、開始 を選択します。[Status] が [online] に変わるまで進まないでください。このプロセスは完了までに数分かかることがあるため、しばらくお待ちください。

これで、スタック、レイヤー、およびクックブックが S3 バケットから自動的にコピーされたインスタンスが用意できました。[次のステップ](#)では、インスタンスのクックブック内からデフォルトのレシピを実行し、テストします。

ステップ 3: レシピを実行し、テストする

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

クックブック内から default レシピを実行してテストします。AWS OpsWorks インスタンスにコピーされたスタック。すでに説明したように、これはレシピの実行時にログにシンプルなメッセージを表示する 1 行のレシピです。

レシピを実行するには

1. サービスナビゲーションペインで、[Stack] を選択します。[MyCookbooksDemoStack] ページが表示されます。
2. [Run Command] を選択します。[Run Command] ページが表示されます。
3. [Command] の [Execute Recipes] を選択します。
4. [Recipes to execute] に「**opsworks_cookbook_demo::default**」と入力します。

opsworks_cookbook_demo は、`metadata.rb` ファイルで宣言されているクックブックの名前です。**default** は実行するレシピの名前です。つまり、クックブックの `default.rb` サブディレクトリの `recipes` ファイルの、ファイル拡張子がない名前です。

5. 次のデフォルト設定はそのままにしておきます。
 - [Comment] (空白)
 - アドバンスド、カスタム Chef JSON (空白)
 - インスタンス (チェック、MyCookbooksDemoLayer チェック、クックブック-demo1 チェックをすべて選択)

6. [Execute Recipes] を選択します。[Running command execute_recipes] ページが表示されます。[Status] が [successful] に変わるまで進まないでください。このプロセスは数分かかることがあるため、しばらくお待ちください。

レシピの結果を確認するには

1. [Running command execute_recipes] ページを表示し、[cookbooks-demo1] と [Log] の [show] を選択します。[execute_recipes] ログページが表示されます。
2. ログを下へスクロールして、次のようなエントリを見つけます。

```
[2015-11-13T19:14:39+00:00] INFO: ***** Hello, World! *****
```

最初のレシピを正常に実行しました。[次のステップ](#)では、インスタンスにパッケージをインストールするレシピを追加してクックブックを更新します。

ステップ 4: クックブックを更新してパッケージをインストールする

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

人気のあるテキストエディタ GNU Emacs を含むパッケージをインスタンスにインストールするレシピを追加して、クックブックを更新します。

インスタンスに簡単にログインしてパッケージを 1 回インストールできますが、レシピを記述すると、[から](#) レシピを実行できます。AWS OpsWorks スタックを 1 回スタックすると、スタック内の複数のインスタンスに複数のパッケージを同時にインストールできます。

クックブックを更新してパッケージをインストールするには

1. ローカルワークステーションの opsworks_cookbook_demo ディレクトリにある recipes サブディレクトリで、以下のコードを含む install_package.rb という名前のファイルを作成します。


```
package "Install Emacs" do
  package_name "emacs"
end
```

このレシピは emacs パッケージをインスタンスにインストールします (詳細については、「[package](#)」を参照してください)。

Note

レシピには任意のファイル名を付けることができます。必要なときに常に正しいレシピ名を指定してください。AWS OpsWorks recipe を実行するスタック。

2. ターミナルまたはコマンドプロンプトで、tar コマンドを使用して `opsworks_cookbook_demo.tar.gz` というファイルの新しいバージョンを作成します。これには、`opsworks_cookbook_demo` ディレクトリとその更新されたコンテンツが含まれます。
3. 更新済みの `opsworks_cookbook_demo.tar.gz` ファイルを S3 バケットにアップロードします。

この新しいレシピは、インスタンスのクックブックを更新し、更新されたクックブック内から新しいレシピを実行するときに実行されます。次の手順は、これを行う方法を示しています。

[次のステップ](#) を完了すると、インスタンスにログインし、コマンドプロンプト `emacs` から `emacs` を入力して Emacs GNU を起動できます。(詳細については、「[Linux インスタンスへの接続](#)」を参照してください)。GNU Emacs を終了するには、`Ctrl+X` を押し、`Ctrl+C` を押します。

Important

インスタンスにログインするには、`user` を指定する必要があります。AWS OpsWorks パブリックSSHキーに関する情報を含むスタック (`ssh-keygen` や `P` などのツールを使用して作成できます `uTTYgen`) と、ユーザーがインスタンスにログインできるようにするには `MyCookbooksDemoStack`、スタックに対するアクセス許可を設定する必要があります。手順については、「[ユーザーのパブリック SSH キーの登録](#)」および「[SSH でのログイン](#)」を参照してください。

ステップ 5: インスタンスのクックブックを更新し、レシピを実行する

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

インスタンスのクックブックを変更し、インスタンスの更新されたクックブック内からレシピを実行します。このウォークスルーの残りの部分では、新しいレシピを追加してクックブックを更新するたびに、この手順を繰り返します。

インスタンスのクックブックを更新するには

1. サービスナビゲーションペインで、[Stack] を選択します。[MyCookbooksDemoStack] ページが表示されます。
2. [Run Command] を選択します。[Run Command] ページが表示されます。
3. [Command] で、[Update Custom Cookbooks] を選択します。
4. 次のデフォルト設定はそのままにしておきます。
 - [Comment] (空白)
 - アドバンスト、カスタム Chef JSON (空白)
 - アドバンスト、インスタンス (チェック済み、MyCookbooksDemoLayer チェック済み、クックブック-demo1 チェック済み) をすべて選択
5. [Update Custom Cookbooks] を選択します。[Running command update_custom_cookbooks] ページが表示されます。[Status] が [successful] に変わるまで進まないでください。このプロセスは完了までに数分かかることがあるため、しばらくお待ちください。

レシピを実行するには

1. サービスナビゲーションペインで、[Stack] を選択します。[MyCookbooksDemoStack] ページが表示されます。
2. [Run Command] を選択します。[Run Command] ページが表示されます。

3. [Command] の [Execute Recipes] を選択します。
4. [Recipes to execute] で、実行するレシピの名前を入力します。これを最初に行うときに、レシピは `opsworks_cookbook_demo::install_package` という名前になります。

Note

後でこの手順を繰り返すときに、クックブックの名前 (`opsworks_cookbook_demo`) を入力し、その後に 2 つのコロン (`::`)、レシピの名前 (`.rb` ファイル拡張子を除くレシピのファイル名) を付けます。

5. 次のデフォルト設定はそのままにしておきます。
 - [Comment] (空白)
 - アドバンスド、カスタム Chef JSON (空白)
 - インスタンス チェック済み、チェック済み、クックブック-demo1 チェック済みをすべて選択) MyCookbooksDemoLayer
6. [Execute Recipes] を選択します。[Running command execute_recipes] ページが表示されます。[Status] が [successful] に変わるまで進まないでください。このプロセスは数分かかることがあるため、しばらくお待ちください。

Note

レシピを手動で実行する必要はありません。セットアップイベントや設定イベントなど、レイヤーのライフサイクルイベントにレシピを割り当てることができます。AWS OpsWorks スタックは、イベントが発生すると、これらのレシピを自動的に実行します。詳細については、「[AWS OpsWorks スタックライフサイクルイベント](#)」を参照してください。

次のステップでは、クックブックを更新してユーザーをインスタンスに追加します。

ステップ 6: クックブックを更新してユーザーを追加する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)

わしてください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

インスタンスにローカルユーザーを追加するレシピを追加してクックブックを更新し、ユーザーのホームディレクトリとシェルを設定します。これは、Linux の `adduser` または `useradd` コマンドまたは Windows の `net user` コマンドの実行に似ています。インスタンスのファイルとディレクトリへのアクセスを制御する場合などに、インスタンスにローカルユーザーを追加します。

クックブックを使用せずにユーザーを管理することもできます。詳細については、「[ユーザーの管理](#)」を参照してください。

インスタンスのクックブックを更新し、新しいレシピを実行するには

1. ローカルワークステーションの `opsworks_cookbook_demo` ディレクトリにある `recipes` サブディレクトリで、以下のコードを含む `add_user.rb` という名前のファイルを作成します (詳細については、[user](#) を参照してください)。

```
user "Add a user" do
  home "/home/jdoe"
  shell "/bin/bash"
  username "jdoe"
end
```

2. ターミナルまたはコマンドプロンプトで、`tar` コマンドを使用して `opsworks_cookbook_demo.tar.gz` というファイルの新しいバージョンを作成します。これには、`opsworks_cookbook_demo` ディレクトリとその更新されたコンテンツが含まれます。
3. 更新済みの `opsworks_cookbook_demo.tar.gz` ファイルを S3 バケットにアップロードします。
4. 「[ステップ 5: インスタンスのクックブックを更新し、レシピを実行する](#)」の手順に従って、インスタンスのクックブックを更新し、レシピを実行します。「レシピを実行するには」の手順の [Recipes to execute] に、「`opsworks_cookbook_demo::add_user`」と入力します。

レシピをテストするには

1. すでに行っていない場合は、インスタンスにログインします。
2. コマンドプロンプトで次のコマンドを実行して、新しいユーザーが追加されたことを確認します。

```
grep jdoe /etc/passwd
```

ユーザー名、ID 番号、グループ ID 番号、ホームディレクトリ、シェルなど詳細を含めて、ユーザーに関する次のような情報が表示されます。

```
jdoe:x:501:502:~/home/jdoe:/bin/bash
```

[次のステップ](#)では、クックブックを更新してインスタンスにディレクトリを作成します。

ステップ 7: クックブックを更新してディレクトリを作成する

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

インスタンスにディレクトリを追加するレシピを追加してクックブックを更新します。これは、Linux の mkdir コマンドまたは Windows の md または mkdir コマンドの実行に似ています。

インスタンスのクックブックを更新し、新しいレシピを実行するには

1. ローカルワークステーションの opsworks_cookbook_demo ディレクトリにある recipes サブディレクトリで、以下のコードを含む create_directory.rb という名前のファイルを作成します。詳細については、「[directory](#)」を参照してください。

```
directory "Create a directory" do
  group "root"
  mode "0755"
  owner "ec2-user"
  path "/tmp/create-directory-demo"
end
```

2. ターミナルまたはコマンドプロンプトで、tar コマンドを使用して `opsworks_cookbook_demo.tar.gz` というファイルの新しいバージョンを作成します。これには、`opsworks_cookbook_demo` ディレクトリとその更新されたコンテンツが含まれます。
3. 更新済みの `opsworks_cookbook_demo.tar.gz` ファイルを S3 バケットにアップロードします。
4. 「[ステップ 5: インスタンスのクックブックを更新し、レシピを実行する](#)」の手順に従って、インスタンスのクックブックを更新し、レシピを実行します。「レシピを実行するには」の手順の [Recipes to execute] に、「`opsworks_cookbook_demo::create_directory`」と入力します。

レシピをテストするには

1. すでに行っていない場合は、インスタンスにログインします。
2. コマンドプロンプトで次のコマンドを実行して、新しいディレクトリが追加されたことを確認します。

```
ls -la /tmp/create-directory-demo
```

新しく追加されたディレクトリに関する情報が、アクセス権限、所有者名、グループ名などの情報を含めて表示されます。

```
drwxr-xr-x 2 ec2-user root 4096 Nov 18 00:35 .  
drwxrwxrwt 6 root      root 4096 Nov 24 18:17 ..
```

[次のステップ](#)では、クックブックを更新してインスタンスにファイルを作成します。

ステップ 8: クックブックを更新してファイルを作成し、コピーする

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

インスタンスに 2 つのファイルを追加するレシピを追加してクックブックを更新します。レシピの最初のリソースは、レシピコードを使用して完全にファイルを作成します。これは、Linux の `cat`、`echo`、または `touch` コマンド、または Windows の `echo` または `fsutil` コマンドに似ています。この手法は、少数のファイル、小さいファイル、またはシンプルなファイルに対して有効です。レシピの 2 番目のリソースはクックブックのファイルをインスタンスの別のディレクトリにコピーします。これは、Linux の `cp` コマンドまたは Windows の `copy` コマンドの実行と似ています。この手法は、多数のファイル、大きいファイル、または複雑なファイルに対して有効です。

このステップを開始する前に、「[ステップ 7: クックブックを更新してディレクトリを作成する](#)」を完了して、ファイルの親ディレクトリがすでに存在することを確認します。

インスタンスのクックブックを更新し、新しいレシピを実行するには

1. ローカルワークステーションの `opsworks_cookbook_demo` ディレクトリで、`files` という名前のサブディレクトリを作成します
2. `files` サブディレクトリで、`hello.txt` というテキストを含む **Hello, World!** という名前のファイルを作成します。
3. `opsworks_cookbook_demo` ディレクトリにある `recipes` サブディレクトリで、以下のコードを含む `create_files.rb` という名前のファイルを作成します。詳細については、[file](#) と [cookbook_file](#) を参照してください。

```
file "Create a file" do
  content "<html>This is a placeholder for the home page.</html>"
  group "root"
  mode "0755"
  owner "ec2-user"
  path "/tmp/create-directory-demo/index.html"
end

cookbook_file "Copy a file" do
  group "root"
  mode "0755"
  owner "ec2-user"
  path "/tmp/create-directory-demo/hello.txt"
  source "hello.txt"
end
```

`file` リソースは、指定されたパスでファイルを作成します。`cookbook_file` リソースは、クックブックで作成した `files` ディレクトリから、インスタンスの別のディレクトリにファイ

ルをコピーします (Chef では、ファイルのコピー元として `files` という名前のサブディレクトリがあることが想定されます)。

- ターミナルまたはコマンドプロンプトで、`tar` コマンドを使用して `opsworks_cookbook_demo.tar.gz` というファイルの新しいバージョンを作成します。これには、`opsworks_cookbook_demo` ディレクトリとその更新されたコンテンツが含まれます。
- 更新済みの `opsworks_cookbook_demo.tar.gz` ファイルを S3 バケットにアップロードします。
- [「ステップ 5: インスタンスのクックブックを更新し、レシピを実行する」](#) の手順に従って、インスタンスのクックブックを更新し、レシピを実行します。「レシピを実行するには」の手順の [Recipes to execute] に、「`opsworks_cookbook_demo::create_files`」と入力します。

レシピをテストするには

- すでに行っていない場合は、インスタンスにログインします。
- コマンドプロンプトで、新しいファイルが追加されたことを確認するため、次のコマンドを 1 つずつ実行します。

```
sudo cat /tmp/create-directory-demo/index.html  
  
sudo cat /tmp/create-directory-demo/hello.txt
```

ファイルの内容が表示されます。

```
<html>This is a placeholder for the home page.</html>  
  
Hello, World!
```

[次のステップ](#)では、クックブックを更新してインスタンスでコマンドを実行します。

ステップ 9: クックブックを更新してコマンドを実行する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)

わしてください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

インスタンスに SSHキーを作成するコマンドを実行するレシピを追加して、クックブックを更新します。

インスタンスのクックブックを更新し、新しいレシピを実行するには

1. ローカルワークステーションの `opsworks_cookbook_demo` ディレクトリにある `recipes` サブディレクトリで、以下のコードを含む `run_command.rb` という名前のファイルを作成します。詳細については、「[execute](#)」を参照してください。

```
execute "Create an SSH key" do
  command "ssh-keygen -f /tmp/my-key -N fLyC3jbY"
end
```

2. ターミナルまたはコマンドプロンプトで、`tar` コマンドを使用して `opsworks_cookbook_demo.tar.gz` というファイルの新しいバージョンを作成します。これには、`opsworks_cookbook_demo` ディレクトリとその更新されたコンテンツが含まれます。
3. 更新済みの `opsworks_cookbook_demo.tar.gz` ファイルを S3 バケットにアップロードします。
4. 「[ステップ 5: インスタンスのクックブックを更新し、レシピを実行する](#)」の手順に従って、インスタンスのクックブックを更新し、レシピを実行します。「レシピを実行するには」の手順の [Recipes to execute] に、「`opsworks_cookbook_demo::run_command`」と入力します。

レシピをテストするには

1. すでに行っていない場合は、インスタンスにログインします。
2. コマンドプロンプトから、次のコマンドを一度に 1 つずつ実行して、SSHキーが作成されたことを確認します。

```
sudo cat /tmp/my-key

sudo cat /tmp/my-key.pub
```

SSH プライベートキーとパブリックキーの内容が表示されます。

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: AES-128-CBC, DEF7A09C...541583FA
A5p9dCuo...wp0YYH1c
-----END RSA PRIVATE KEY-----

ssh-rsa AAAAB3N...KaNogZkT root@cookbooks-demo1
```

[次のステップ](#)では、クックブックを更新してインスタンスでスクリプトを実行します。

ステップ 10: クックブックを更新してスクリプトを実行する

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

インスタンスでスクリプトを実行するレシピを追加してクックブックを更新します。このレシピではディレクトリを作成し、そのディレクトリでファイルを作成します。複数のコマンドを含むスクリプトを実行するレシピの作成は、それらのコマンドを 1 つずつ実行するよりも簡単です。

インスタンスのクックブックを更新し、新しいレシピを実行するには

1. ローカルワークステーションの `opsworks_cookbook_demo` ディレクトリにある `recipes` サブディレクトリで、以下のコードを含む `run_script.rb` という名前のファイルを作成します。詳細については、「[script](#)」を参照してください。

```
script "Run a script" do
  interpreter "bash"
  code <<-EOH
    mkdir -m 777 /tmp/run-script-demo
    touch /tmp/run-script-demo/helloworld.txt
    echo "Hello, World!" > /tmp/run-script-demo/helloworld.txt
  EOH
```

```
end
```

2. ターミナルまたはコマンドプロンプトで、tar コマンドを使用して `opsworks_cookbook_demo.tar.gz` というファイルの新しいバージョンを作成します。これには、`opsworks_cookbook_demo` ディレクトリとその更新されたコンテンツが含まれます。
3. 更新済みの `opsworks_cookbook_demo.tar.gz` ファイルを S3 バケットにアップロードします。
4. 「[ステップ 5: インスタンスのクックブックを更新し、レシピを実行する](#)」の手順に従って、インスタンスのクックブックを更新し、レシピを実行します。「レシピを実行するには」の手順の [Recipes to execute] に、「`opsworks_cookbook_demo::run_script`」と入力します。

レシピをテストするには

1. すでに行っていない場合は、インスタンスにログインします。
2. コマンドプロンプトで次のコマンドを実行して、新しいファイルが追加されたことを確認します。

```
sudo cat /tmp/run-script-demo/helloworld.txt
```

ファイルの内容が表示されます。

```
Hello, World!
```

[次のステップ](#)では、クックブックを更新してインスタンスでサービスを管理します。

ステップ 11: クックブックを更新してサービスを管理する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

インスタンスのサービスを管理するレシピを追加してクックブックを更新します。これは、Linux の `service` コマンドまたは Windows の、`net stop`、`net start` などのコマンドの実行に似ています。このレシピはインスタンスの `crond` サービスを停止します。

インスタンスのクックブックを更新し、新しいレシピを実行するには

1. ローカルワークステーションの `opsworks_cookbook_demo` ディレクトリにある `recipes` サブディレクトリで、以下のコードを含む `manage_service.rb` という名前のファイルを作成します。詳細については、「[service](#)」を参照してください。

```
service "Manage a service" do
  action :stop
  service_name "crond"
end
```

2. ターミナルまたはコマンドプロンプトで、`tar` コマンドを使用して `opsworks_cookbook_demo.tar.gz` というファイルの新しいバージョンを作成します。これには、`opsworks_cookbook_demo` ディレクトリとその更新されたコンテンツが含まれます。
3. 更新済みの `opsworks_cookbook_demo.tar.gz` ファイルを S3 バケットにアップロードします。
4. 「[ステップ 5: インスタンスのクックブックを更新し、レシピを実行する](#)」の手順に従って、インスタンスのクックブックを更新し、レシピを実行します。「レシピを実行するには」の手順の [Recipes to execute] に、「`opsworks_cookbook_demo::manage_service`」と入力します。

レシピをテストするには

1. すでに行っていない場合は、インスタンスにログインします。
2. コマンドプロンプトで次のコマンドを実行し、`crond` サービスが停止したことを確認します。

```
service crond status
```

以下が表示されます。

```
crond is stopped
```

3. `crond` サービスを再起動するには、次のコマンドを実行します。

```
sudo service crond start
```

以下が表示されます。

```
Starting crond: [ OK ]
```

4. crond サービスが開始されたことを確認するには、次のコマンドを再度実行します。

```
service crond status
```

以下のような情報が表示されます。

```
crond (pid 3917) is running...
```

[次のステップ](#)では、クックブックを更新して、インスタンスJSONにカスタムとして保存されている情報を参照します。

ステップ 12: カスタムを使用するようにクックブックを更新する JSON

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

インスタンスに保存されているカスタム を参照するレシピを追加してJSON、クックブックを更新します。

スタックを作成、更新、またはクローンするたびに、またはデプロイまたはスタックコマンドを実行するときに、カスタムJSON形式で情報を指定できます。これは、データベースからこのデータを取得する代わりに、インスタンスのレシピでデータのごく一部を固定するときに役立ちます。詳細については、「[カスタムの使用 JSON](#)」を参照してください。

このチュートリアルでは、カスタム を使用してJSON、顧客請求書に関する架空の情報を提供します。カスタムについてはJSON、このステップの後半で説明します。

インスタンスのクックブックを更新し、新しいレシピを実行するには

1. ローカルワークステーションの `recipes` ディレクトリにある `opsworks_cookbook_demo` サブディレクトリで、以下のレシピコードを含む `custom_json.rb` という名前のファイルを作成します。

```
Chef::Log.info("***** For customer '#{node['customer-id']}' invoice
 '#{node['invoice-number']}' *****")
Chef::Log.info("***** Invoice line number 1 is a '#{node['line-items']
 ['line-1']}' *****")
Chef::Log.info("***** Invoice line number 2 is a '#{node['line-items']
 ['line-2']}' *****")
Chef::Log.info("***** Invoice line number 3 is a '#{node['line-items']
 ['line-3']}' *****")
```

このレシピは、カスタム の値に関するメッセージをログに表示しますJSON。

2. ターミナルまたはコマンドプロンプトで、`tar` コマンドを使用して `opsworks_cookbook_demo.tar.gz` というファイルの新しいバージョンを作成します。これには、`opsworks_cookbook_demo` ディレクトリとその更新されたコンテンツが含まれます。
3. 更新済みの `opsworks_cookbook_demo.tar.gz` ファイルを S3 バケットにアップロードします。
4. 「[ステップ 5: インスタンスのクックブックを更新し、レシピを実行する](#)」の手順に従って、インスタンスのクックブックを更新し、レシピを実行します。「レシピを実行するには」の手順の [Recipes to execute] に、「`opsworks_cookbook_demo::custom_json`」と入力します。アドバンスト、カスタム Chef JSONには、次のカスタム を入力しますJSON。

```
{
  "customer-id": "0123",
  "invoice-number": "9876",
  "line-items": {
    "line-1": "tractor",
    "line-2": "passenger car",
    "line-3": "trailer"
  }
}
```

レシピをテストするには

1. 前の手順の [Running command execute_recipes] ページを表示した状態で、[cookbooks-demo1]、[Log] の [show] を選択します。[execute_recipes] ログページが表示されます。
2. ログを下にスクロールして、次のようなエントリを見つけます。

```
[2015-11-14T14:18:30+00:00] INFO: ***** For customer '0123' invoice '9876'
*****
[2015-11-14T14:18:30+00:00] INFO: ***** Invoice line number 1 is a 'tractor'
*****
[2015-11-14T14:18:30+00:00] INFO: ***** Invoice line number 2 is a 'passenger
car' *****
[2015-11-14T14:18:30+00:00] INFO: ***** Invoice line number 3 is a 'trailer'
*****
```

これらのエントリには、アドバンスド JSON、カスタム Chef ボックスに入力されたカスタム JSON からの情報が表示されます。

[次のステップ](#) では、クックブックを更新してデータバッグから情報を取得します。データバッグは、AWS OpsWorks スタックは各インスタンスに を保存します。

ステップ 13: クックブックを更新してデータバッグを使用する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

のスタック設定を参照するレシピを追加してクックブックを更新する AWS OpsWorks スタックは、一連のデータバッグにインスタンスに を保存します。このレシピは、インスタンスに保存された特定のスタック設定についてログにメッセージを表示します。詳細については、「[AWS OpsWorks スタックデータバッグリファレンス](#)」を参照してください。

インスタンスのクックブックを更新し、新しいレシピを実行するには

1. ローカルワークステーションの `recipes` ディレクトリにある `opsworks_cookbook_demo` サブディレクトリで、以下のコードを含む `data_bags.rb` という名前のファイルを作成します。

```
instance = search("aws_opsworks_instance").first
layer = search("aws_opsworks_layer").first
stack = search("aws_opsworks_stack").first

Chef::Log.info("***** This instance's instance ID is
 '#{instance['instance_id']}' *****")
Chef::Log.info("***** This instance's public IP address is
 '#{instance['public_ip']}' *****")
Chef::Log.info("***** This instance belongs to the layer '#{layer['name']}'
 *****")
Chef::Log.info("***** This instance belongs to the stack '#{stack['name']}'
 *****")
Chef::Log.info("***** This stack gets its cookbooks from
 '#{stack['custom_cookbooks_source']['url']}' *****")
```

このレシピは、インスタンスに保存された特定のスタック設定についてログにメッセージを表示します。

2. ターミナルまたはコマンドプロンプトで、`tar` コマンドを使用して `opsworks_cookbook_demo.tar.gz` というファイルの新しいバージョンを作成します。これには、`opsworks_cookbook_demo` ディレクトリとその更新されたコンテンツが含まれます。
3. 更新済みの `opsworks_cookbook_demo.tar.gz` ファイルを S3 バケットにアップロードします。
4. [「ステップ 5: インスタンスのクックブックを更新し、レシピを実行する」](#) の手順に従って、インスタンスのクックブックを更新し、レシピを実行します。「レシピを実行するには」の手順の [Recipes to execute] に、「**opsworks_cookbook_demo::data_bags**」と入力します。

レシピをテストするには

1. 前の手順の [Running command execute_recipes] ページを表示した状態で、[cookbooks-demo1]、[Log] の [show] を選択します。[execute_recipes] ログページが表示されます。
2. ログを下にスクロールして、次のようなエントリを見つけます。


```
[2015-11-14T14:39:06+00:00] INFO: ***** This instance's instance ID is
'f80fa119-81ab-4c3c-883d-6028e52c89EX' *****
[2015-11-14T14:39:06+00:00] INFO: ***** This instance's public IP address is
'192.0.2.0' *****
[2015-11-14T14:39:06+00:00] INFO: ***** This instance belongs to the layer
'MyCookbooksDemoLayer' *****
[2015-11-14T14:39:06+00:00] INFO: ***** This instance belongs to the stack
'MyCookbooksDemoStack' *****
[2015-11-14T14:39:06+00:00] INFO: ***** This stack gets its cookbooks from
'https://s3.amazonaws.com/amzn-s3-demo-bucket/opsworks_cookbook_demo.tar.gz'
*****
```

このレシピは、インスタンスに保存された特定のスタック設定についてメッセージを表示します。

[次のステップ](#)では、クックブックを更新してレシピコードを複数回実行します。

ステップ 14: クックブックを更新して繰り返しを使用する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[にお問い合わせください](#)。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

レシピコードを複数回繰り返す、繰り返しの手法を使用するレシピを追加してクックブックを更新します。このレシピは、複数のコンテンツを含むデータバッグ項目についてログにメッセージを表示します。

インスタンスのクックブックを更新し、新しいレシピを実行するには

1. ローカルワークステーションの recipes ディレクトリにある opsworks_cookbook_demo サブディレクトリで、以下のコードを含む iteration_demo.rb という名前のファイルを作成します。

```
stack = search("aws_opsworks_stack").first
```

```
Chef::Log.info("***** Content of 'custom_cookbooks_source' *****")

stack["custom_cookbooks_source"].each do |content|
  Chef::Log.info("***** '#{content}' *****")
end
```

Note

前述のレシピコードの作成は、繰り返しを使わない次のレシピコードを記述するよりも短く、より柔軟になり、エラーが起きにくくなります。

```
stack = search("aws_opsworks_stack").first
Chef::Log.info("***** Content of 'custom_cookbooks_source' *****")

Chef::Log::info("***** '['type'", \"#{@stack['custom_cookbooks_source']
['type']}\"]' *****")
Chef::Log::info("***** '['url'", \"#{@stack['custom_cookbooks_source']
['url']}\"]' *****")
Chef::Log::info("***** '['username'",
 \"#{@stack['custom_cookbooks_source']['username']}\"]' *****")
Chef::Log::info("***** '['password'",
 \"#{@stack['custom_cookbooks_source']['password']}\"]' *****")
Chef::Log::info("***** '['ssh_key'",
 \"#{@stack['custom_cookbooks_source']['ssh_key']}\"]' *****")
Chef::Log::info("***** '['revision'",
 \"#{@stack['custom_cookbooks_source']['revision']}\"]' *****")
```

2. ターミナルまたはコマンドプロンプトで、tar コマンドを使用して `opsworks_cookbook_demo.tar.gz` というファイルの新しいバージョンを作成します。これには、`opsworks_cookbook_demo` ディレクトリとその更新されたコンテンツが含まれます。
3. 更新済みの `opsworks_cookbook_demo.tar.gz` ファイルを S3 バケットにアップロードします。
4. 「[ステップ 5: インスタンスのクックブックを更新し、レシピを実行する](#)」の手順に従って、インスタンスのクックブックを更新し、レシピを実行します。「レシピを実行するには」の手順の [Recipes to execute] に、「`opsworks_cookbook_demo::iteration_demo`」と入力します。

レシピをテストするには

1. 前の手順の [Running command execute_recipes] ページを表示した状態で、[cookbooks-demo1]、[Log] の [show] を選択します。[execute_recipes] ログページが表示されます。
2. ログを下にスクロールして、次のようなエントリを見つけます。

```
[2015-11-16T19:56:56+00:00] INFO: ***** Content of 'custom_cookbooks_source'
*****
[2015-11-16T19:56:56+00:00] INFO: ***** '['type', 's3']' *****
[2015-11-16T19:56:56+00:00] INFO: ***** '['url', 'https://s3.amazonaws.com/
amzn-s3-demo-bucket/opsworks_cookbook_demo.tar.gz']' *****
[2015-11-16T19:56:56+00:00] INFO: ***** '['username', 'secret-key-value']'
*****
[2015-11-16T19:56:56+00:00] INFO: ***** '['password', 'secret-access-key-
value']' *****
[2015-11-16T19:56:56+00:00] INFO: ***** '['ssh_key', nil]' *****
[2015-11-16T19:56:56+00:00] INFO: ***** '['revision', nil]' *****
```

このレシピは、複数のコンテンツを含むデータバッグ項目についてログにメッセージを表示します。データバッグ項目は `aws_opsworks_stack` データバッグにあります。データバッグ項目には、`custom_cookbooks_source` という名前のコンテンツがあります。このコンテンツの内部には、`type`、`url`、`username`、`password`、`ssh_key`、および `revision` という名前の6つのコンテンツがあり、それらの値も表示されます。

[次のステップ](#)では、クックブックを更新して、特定の条件が満たされた場合のみレシピコードを実行します。

ステップ 15: クックブックを更新して条件付きロジックを使用する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

ここで、条件付きロジックを使用するレシピを追加してクックブックを作成します。これは、特定の条件が満たされた場合のみコードを実行する手法です。詳細については、「[if Statements](#)」および「[case Statements](#)」を参照してください。

このレシピでは、データバッグのコンテンツに基づいて2つの処理を実行します。インスタンスが実行されているオペレーティングシステムを識別するメッセージをログに表示し、オペレーティングシステムがLinuxである場合のみ、特定のLinuxディストリビューションに対して適切なパッケージマネージャーを使用してパッケージをインストールします。このパッケージはtreeという名前であり、ディレクトリリストを視覚化するためのシンプルなアプリケーションです。

インスタンスのクックブックを更新し、新しいレシピを実行するには

1. ローカルワークステーションの `opsworks_cookbook_demo` directory にある `recipes` サブディレクトリで、以下のコードを含む `conditional_logic.rb` という名前のファイルを作成します。

```
instance = search("aws_opsworks_instance").first
os = instance["os"]

if os == "Red Hat Enterprise Linux 7"
  Chef::Log.info("***** Operating system is Red Hat Enterprise Linux.
*****")
elsif os == "Ubuntu 14.04 LTS" || os == "Ubuntu 16.04 LTS" || os == "Ubuntu 18.04
LTS"
  Chef::Log.info("***** Operating system is Ubuntu. *****")
elsif os == "Microsoft Windows Server 2012 R2 Base"
  Chef::Log.info("***** Operating system is Windows. *****")
elsif os == "Amazon Linux 2015.03" || os == "Amazon Linux 2015.09" || os == "Amazon
Linux 2016.03" || os == "Amazon Linux 2016.09" || os == "Amazon Linux 2017.03"
|| os == "Amazon Linux 2017.09" || os == "Amazon Linux 2018.03" || os == "Amazon
Linux 2"
  Chef::Log.info("***** Operating system is Amazon Linux. *****")
elsif os == "CentOS Linux 7"
  Chef::Log.info("***** Operating system is CentOS 7. *****")
else
  Chef::Log.info("***** Cannot determine operating system. *****")
end

case os
when "Ubuntu 14.04 LTS", "Ubuntu 16.04 LTS", "Ubuntu 18.04 LTS"
  apt_package "Install a package with apt-get" do
    package_name "tree"
  end
end
```

```
end
when "Amazon Linux 2015.03", "Amazon Linux 2015.09", "Amazon Linux 2016.03",
    "Amazon Linux 2016.09", "Amazon Linux 2017.03", "Amazon Linux 2017.09", "Amazon
    Linux 2018.03", "Amazon Linux 2", "Red Hat Enterprise Linux 7", "CentOS Linux 7"
    yum_package "Install a package with yum" do
    package_name "tree"
    end
else
    Chef::Log.info("***** Cannot determine operating system type, or operating
    system is not Linux. Package not installed. *****")
end
```

2. ターミナルまたはコマンドプロンプトで、tar コマンドを使用して `opsworks_cookbook_demo.tar.gz` というファイルの新しいバージョンを作成します。これには、`opsworks_cookbook_demo` ディレクトリとその更新されたコンテンツが含まれます。
3. 更新済みの `opsworks_cookbook_demo.tar.gz` ファイルを S3 バケットにアップロードします。
4. 「[ステップ 5: インスタンスのクックブックを更新し、レシピを実行する](#)」の手順に従って、インスタンスのクックブックを更新し、レシピを実行します。「レシピを実行するには」の手順の [Recipes to execute] に、「`opsworks_cookbook_demo::conditional_logic`」と入力します。

レシピをテストするには

1. 前の手順の [Running command execute_recipes] ページを表示した状態で、[cookbooks-demo1]、[Log] の [show] を選択します。[execute_recipes] ログページが表示されます。
2. ログを下へスクロールして、次のようなエントリを見つけます。

```
[2015-11-16T19:59:05+00:00] INFO: ***** Operating system is Amazon Linux.
*****
```

インスタンスのオペレーティングシステムは Amazon Linux 2016.09 であるため、(レシピのコードの 5 つの可能なエントリのうち)先行するエントリのみがログに表示されます。

3. オペレーティングシステムが Linux の場合、レシピは `tree` パッケージをインストールします。ディレクトリのコンテンツを視覚的に表示するには、希望するディレクトリまたはそのディレクトリのパス (`tree /var/chef/runs` など) からコマンドプロンプトで「`tree`」と入力します。

[次のステップ](#)では、クックブックを更新して、Chef コミュニティで提供される外部クックブックの機能を使用します。

ステップ 16: クックブックを更新してコミュニティクックブックを使用する

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

最後に、クックブックを更新して、Chef コミュニティで提供された外部のクックブックの機能を使用します。このウォークスルーに使用する外部のクックブックは、外部の Chef クックブックにアクセスするための一般的な場所である [Chef Supermarket](#) を通じて利用できます。この外部クックブックでは、「[ステップ 4: クックブックを更新してパッケージをインストールする](#)」で行ったようにアプリケーションをダウンロードしてインストールできるカスタムリソースが提供されます。ただし、このリソースはパッケージに加えてウェブアプリケーションや他の種類のアプリケーションをインストールできます。

クックブックが別のクックブックに依存している場合、他のクックブックへの依存関係を指定する必要があります。クックブックの依存関係を宣言し、管理するには、Berkshelf と呼ばれるツールを使用することをお勧めします。Berkshelf をお客様のローカルワークステーションにインストールする方法の詳細については、Chef ウェブサイトの [About Berkshelf](#) を参照してください。

Berkshelf をインストールしたら、以下の手順に従ってクックブックの依存関係を宣言し、外部のクックブックでリソースを呼び出すレシピを作成します。

クックブックの依存関係を宣言するには

1. ローカルワークステーションの `opsworks_cookbook_demo` ディレクトリで、次の行を `metadata.rb` ファイルの末尾に追加します。

```
depends "application", "5.0.0"
```

これにより、`application version 5.0.0` という名前のクックブックで依存関係が宣言されます。

2. `opsworks_cookbook_demo` ディレクトリのルートから次のコマンドを実行します。コマンドの最後のピリオドは意図的なものです。

```
berks init .
```

Berkshelf により、より高度なシナリオで後から使用できる多くのフォルダとファイルが作成されます。このウォークスルーに必要なファイルは、`Berksfile` という名前のファイルのみです。

3. `Berksfile` ファイルの末尾に次の行を追加します。

```
cookbook "application", "5.0.0"
```

これにより、ユーザーが [application cookbook version 5.0.0](#) の使用を希望している事実が Berkshelf に伝えられ、Berkshelf は、Chef Supermarket から該当バージョンをダウンロードします。

4. 端末またはコマンドプロンプトで、`opsworks_cookbook_demo` ディレクトリのルートから次のコマンドを実行します。

```
berks install
```

Berkshelf が、クックブックとアプリケーションクックブック両方について、依存関係のリストを作成します。Berkshelf は、次の手順でこの依存関係リストを使用します。

インスタンスのクックブックを更新し、新しいレシピを実行するには

1. `recipes` ディレクトリにある `opsworks_cookbook_demo` サブディレクトリで、以下のコードを含む `dependencies_demo.rb` という名前のファイルを作成します。

```
application "Install NetHack" do
  package "nethack.x86_64"
end
```

このレシピは、一般的なテキストベースのアドベンチャーゲームをインスタンス NetHack にインストールするために、アプリケーションクックブックのアプリケーションリソースによって異なります。(もちろん、パッケージがインスタンスでパッケージマネージャーに対して容易に利用できる限り、他のパッケージ名に置き換えることができます)。

2. `opsworks_cookbook_demo` ディレクトリのルートから次のコマンドを実行します。

```
berks package
```

Berkshelf は前の手順の依存関係のリストを使用して、`cookbooks-timestamp.tar.gz` という名前のファイルを作成します。これには、クックブックの依存関係クックブックを含めて、`opsworks_cookbook_demo` ディレクトリとその更新されたコンテンツが含まれます。このファイル `opsworks_cookbook_demo.tar.gz` の名前を変更します。

3. 更新済みで名前を変更した `opsworks_cookbook_demo.tar.gz` ファイルを S3 バケットにアップロードします。
4. 「[ステップ 5: インスタンスのクックブックを更新し、レシピを実行する](#)」の手順に従って、インスタンスのクックブックを更新し、レシピを実行します。「レシピを実行するには」の手順の [Recipes to execute] に、「`opsworks_cookbook_demo::dependencies_demo`」と入力します。
5. レシピを実行した後は、インスタンスにログインし、コマンドプロンプトで「`nethack`」と入力してプレイを開始できます（ゲームの詳細については、[NetHack 「](#)」および[NetHack 「ガイドブック](#)」を参照してください。）

[次のステップ](#) では、をクリーンアップできます。AWS このチュートリアルに使用した リソース。このステップは任意です。これらを引き続き使用したい場合があります。AWS の詳細を学習し続けるための リソース AWS OpsWorks スタック。ただし、これらを AWS のリソースでは、に継続的に課金される場合があります。AWS アカウント。これらを保持する場合 AWS 後で使用するためにリソースをめぐって、このチュートリアルはこれで完了です。「」にスキップできます[次のステップ](#)。

ステップ 17: (オプション) クリーンアップする

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

に追加料金が発生しないようにするには AWS アカウント、 を削除できます。AWS このチュートリアルで使用した リソース。これら AWS リソースには、S3 バケット、AWS OpsWorks スタックスタック、およびスタックのコンポーネント。(詳細については、「 の [AWS OpsWorks 料金](#)」を参照してください。) ただし、これらの を引き続き使用することをお勧めします。AWS の詳細を学習し続けるための リソース AWS OpsWorks スタック。これらを保持する場合 AWS リソースが利用可能になりました。このチュートリアルはこれで完了です。「」にスキップできます [次のステップ](#)。

このチュートリアルのために作成したリソースに保存されているコンテンツには、個人識別情報が含まれている可能性があります。この情報の保存が不要になった場合 AWS、このトピックのステップに従います。

S3 バケットを削除するには

- 「[Amazon S3 バケットの削除](#)」を参照してください。

スタックのインスタンスを削除するには

1. 左 AWS OpsWorks スタックコンソールのサービスナビゲーションペインで、インスタンス を選択します。[Instances] ページが表示されます。
2. では MyCookbooksDemoLayer、クックブック-demo1 では、アクション では、停止 を選択します。確認メッセージが表示されたら、[Stop] を選択します。
3. 以下の変更が行われるには数分間かかるため、しばらくお待ちください。すべてが終了するまで先に進まないでください。
 - [Status] が [online] から [stopping] に変わり、最終的に [stopped] になります。
 - [online] は [1] から [0] に変わります。
 - [shutting down] は [0] から [1] に変わり、最終的に [0] になります。
 - [stopped] は最終的に [0] から [1] に変わります。
4. [Actions] (アクション) で、[delete] (削除) を選択します。確認メッセージが表示されたら、[Delete] を選択します。AWS OpsWorks スタックによってインスタンスが削除され、[No instances] (インスタンスがありません) と表示されます。

スタックを削除するには

1. サービスナビゲーションペインで、[Stack] を選択します。[MyCookbooksDemoStack] ページが表示されます。

2. [Delete Stack] を選択します。確認メッセージが表示されたら、[Delete] を選択します。AWS OpsWorks スタックによってスタックが削除され、[Dashboard] (ダッシュボード) ページが表示されます。

オプションで、このチュートリアルで使用したIAMユーザーと Amazon EC2キーペアを他のへのアクセスに再利用しない場合は、削除できます。AWS サービスとEC2インスタンス。手順については、[IAM「ユーザーと Amazon のキーペアと Linux インスタンスの削除」](#)を参照してください。

[EC2](#)

これで、このウォークスルーが完了しました。詳細については、「[次のステップ](#)」を参照してください。

次のステップ

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

このチュートリアルを完了したので、[このチュートリアル](#) の詳細を確認できます。AWS OpsWorks スタックは、以下のリソースを確認して Chef クックブックをサポートします。

- [クックブックとレシピ](#) – Chef と Ruby のバージョンについて説明します。AWS OpsWorks スタックは現在 をサポートしています。また、インスタンスで、カスタムクックブックをインストールおよび更新する方法と、インスタンスでレシピを実行する方法を示しています。
- [Learn Chef](#) – Chef のチュートリアル、Chef のスキルライブラリ、Chef の完全なドキュメント、および Chef のトレーニングクラスへのリンクを提供しています。
- [\[All about Chef\]](#) (Chefのすべて) - Chef に関する完全なドキュメントを提供しています。特に興味深いトピックは次のとおりです。
 - [\[About Cookbooks\]](#)(クックブックについて) - 属性、レシピ、ファイル、メタデータ、テンプレートなど、クックブックの主要コンポーネントについて説明しています。
 - [\[About Recipes\]](#) (レシピについて) - データバッグの使用法、他のレシピの追加方法、レシピで Ruby コードの使用法など、レシピの基本事項について説明しています。

- [\[Resources\]](#) (リソース) - apt_package、cookbook_file、directory、execute、file および package など、全組み込みの Chef リソースの使用方法について説明しています。
- [レシピについて DSL](#) — Chef レシピのコードを、if、`case`、などのステートメントで記述する方法について説明します data_bag data_bag_item search。
- [テンプレートについて](#) — Embedded Ruby (ERB) テンプレートを使用して、設定ファイルなどの静的テキストファイルを動的に生成する方法について説明します。
- [\[Learning Tracks\]](#) (ラーニングトラック) - インスタンスや基本的なウェブアプリケーションの管理、インフラストラクチャコードの開発およびテスト、Chef 分析の使用など、Chef を使用方法を説明しています。
- [\[Learning Chef\]](#) (ラーニングシェフ) - Chef の概要です。発行元は O'Reilly Media です。
- [\[Learning Chef code examples\]](#) (Chefコードの例を学ぶ) - O'Reilly Media によって出版された書籍 [Learning Chef] (ラーニングシェフ) に付随するコードの例を提供しています。

AWS OpsWorks スタックのベストプラクティス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションの戦略、手法、提案は、AWS OpsWorks スタックから最大限のメリットと最適な結果を得るのに役立ちます。

トピック

- [ベストプラクティス: インスタンスのルートデバイスストレージ](#)
- [ベストプラクティス: アプリケーションサーバー数の最適化](#)
- [ベストプラクティス: アクセス権限の管理](#)
- [ベストプラクティス: アプリケーションとクックブックの管理とデプロイ](#)
- [ローカルでのクックブックの依存関係のパッケージ化](#)

ベストプラクティス: インスタンスのルートデバイスストレージ

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

ℹ Note

このトピックは、Amazon Elastic Block Store-backed である必要がある Windows インスタンスには適用されません。

Amazon Elastic Compute Cloud (Amazon EC2) Linux インスタンスには、次のルートデバイスストレージオプションがあります。

- [インスタンスストアでバックアップされたインスタンス] – ルートデバイスは 一時的 なものです。

インスタンスを停止すると、ルートデバイス上のデータは消失し、復元できなくなります。詳細については、「[Amazon EC2 インスタンスストア](#)」を参照してください。

- [Amazon EBS-backed instances (Amazon EBS-backed インスタンス)] – ルートデバイスは、Amazon EBS ボリュームです。

インスタンスを停止しても、Amazon EBS ボリュームは保持されます。インスタンスを起動すると、ボリュームは自動的に再マウントされ、インスタンスの状態と保存されたすべてのデータが復元されます。別のインスタンスにボリュームをマウントすることもできます。詳細については、[Amazon Elastic Block Store \(Amazon EBS\)](#) を参照してください。

使用するルートデバイスストレージオプションを決める際は、以下を考慮します。

起動時間

最初の起動後、Amazon EBS インスタンスは通常より速く再起動します。

最初の起動にかかる時間は、どちらのストレージタイプでもほぼ同じです。いずれのタイプでも完全な設定を行う必要があり、これには、比較的時間のかかるタスク (リモートリポジトリからのパッケージのインストールなど) も含まれます。ただし、その後にインスタンスを再起動するときに、次の違いに注意してください。

- Instance store-Backed インスタンスでは、パッケージのインストールを含む初回の起動時に実行されたのと同じ設定タスクが実行されます。

再起動には、最初の起動と同じくらいの時間がかかります。

- Amazon EBS でバックアップされたインスタンスではルートボリュームが再マウントされ、Setup レシピ が実行されます。

通常、再起動は最初の起動より大幅に速くなります。これは、ルートボリュームにすでにインストールされているパッケージの再インストールなどのタスクを Setup recipes で実行する必要がないためです。

コスト

Amazon EBS-backed インスタンスは、さらにコストがかかります。

- Instance-store Backed インスタンスでは、インスタンスを実行している間だけ課金されます。
- Amazon EBS-backed インスタンスでは、インスタンスが実行しているかどうかに関わらず、Amazon EBS ボリュームの料金を支払います。

詳細については、[「Amazon EBS 料金表」](#)を参照してください。

ログ記録

Amazon EBS-backed インスタンスでは、ログが自動的に保持されます。

- Instance store-Backed インスタンスでは、インスタンスが停止するとログは非表示になります。

インスタンスを停止する前にログを取得するか、[CloudWatch Logs](#) などのサービスを使用して選択したログをリモートに保存する必要があります。

- Amazon EBS-backed インスタンスでは、ログは Amazon EBS ボリュームに保存されます。

ログを表示するには、インスタンスを再起動するか、別のインスタンスにボリュームをマウントします。

依存関係

2 種類のストレージタイプでは、依存関係が異なります。

- Instance-store backed インスタンスは、Amazon S3 に依存しています。

インスタンスを起動するときは、Amazon S3 から AMI をダウンロードする必要があります。

- Amazon EBS-backed インスタンスは、Amazon EBS に依存しています。

インスタンスを起動すると、Amazon EBS ルートボリュームをマウントする必要があります。

[Recommendation: (推奨事項:)] お客様の要件に最適なストレージタイプがわからない場合は、Amazon EBS インスタンスから開始することをお勧めします。Amazon EBS ボリュームには少額の費用がかかりますが、意図しないデータ損失のリスクは軽減されます。

ベストプラクティス: アプリケーションサーバー数の最適化

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

一般的に、本稼働用スタックには、複数のアベイラビリティーゾーン全体に分散されるアプリケーションサーバーが複数含まれます。ただし、入って来るリクエストの数は、時間または曜日によって大幅に異なる場合があります。予測される最大負荷を処理するのに最小限必要なサーバーを実行しようとしても、結局、必要とする容量より多くのサーバー容量に料金を支払うはめになることが少なくありません。サイトを効率的に実行するために推奨される方法は、サーバー数を現在求められているボリュームと一致させることです。

AWS OpsWorks スタックには、サーバーインスタンスの数を管理する 3 つの方法があります。

- [24/7 インスタンス](#) はユーザーが手動で起動し、ユーザーが手動で停止するまで実行されます。
- [時間ベースのインスタンス](#) は、ユーザーが指定したスケジュールで AWS OpsWorks スタックによって自動的に開始および停止します。
- [負荷ベースのインスタンス](#) は、CPU やメモリの使用率など、ユーザーが指定した負荷メトリクスのしきい値を超えると、AWS OpsWorks スタックによって自動的に開始および停止します。

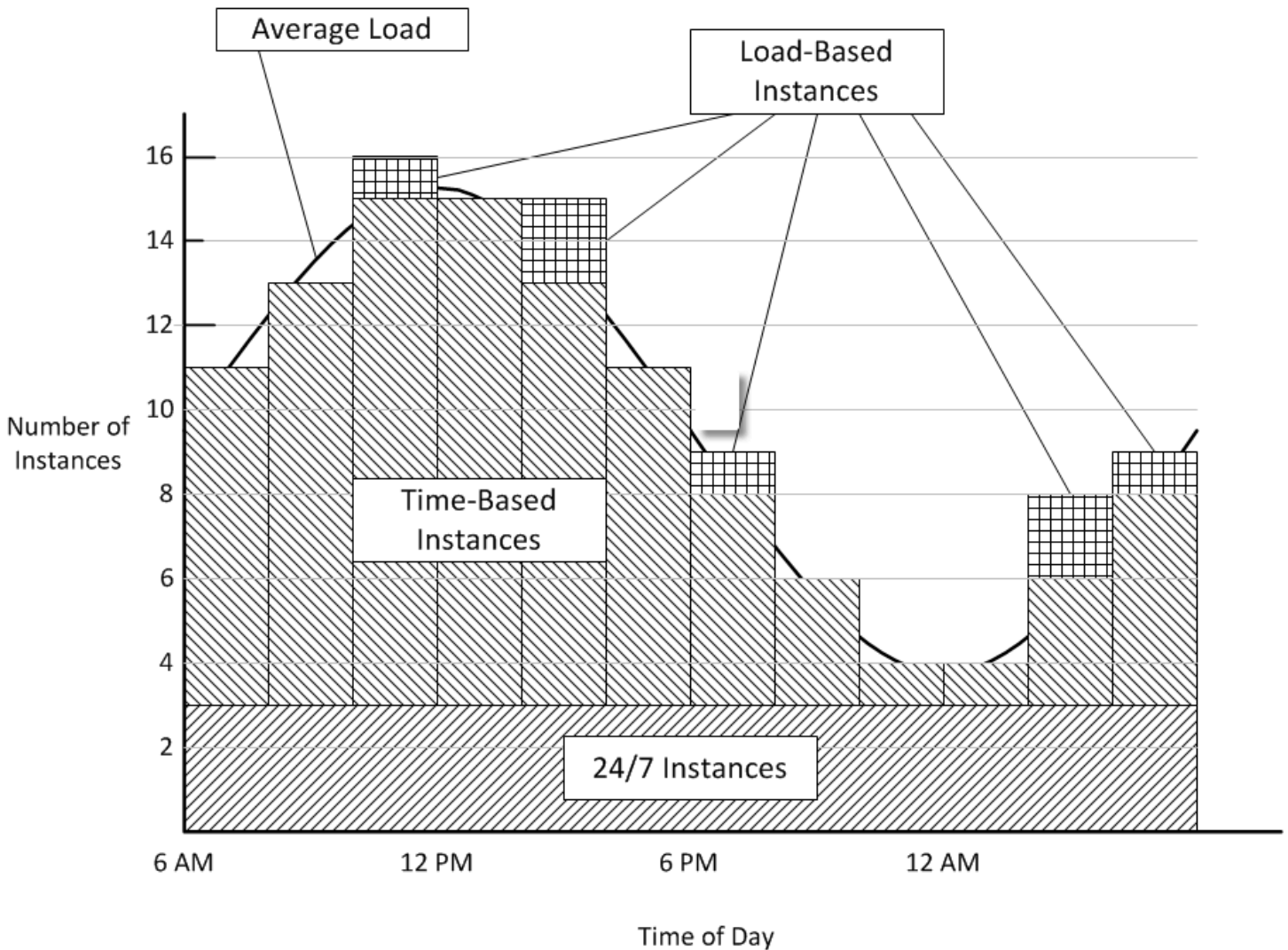
Note

スタックの時間ベースおよび負荷ベースのインスタンスを作成して設定すると、指定した設定に基づいて AWS OpsWorks スタックによって自動的に起動および停止されます。インスタンスの設定または数を変更することを決定しない限り、特別な操作をする必要はありません。

推奨事項: アプリケーションサーバーインスタンスが数個以上あるスタックを管理する場合は、これらの 3 つのインスタンスタイプを組み合わせることをお勧めします。次の例では、スタックのサーバー容量を管理して、さまざまな特徴を持つ変動する日単位のリクエスト量を処理する方法を示します。

- リクエストの平均量は、1 日を通じて正弦的に変化します。
- リクエストの平均最小量では、5 個のアプリケーションサーバーインスタンスが必要です。
- リクエストの平均最大量では、16 個のアプリケーションサーバーインスタンスが必要です。
- リクエストボリュームの急激な増大には、通常 1~2 個のアプリケーションサーバーインスタンスで対処できます。

これは、説明の目的で便宜上用意されていますが、リクエストのボリュームあらゆる変化に簡単に応用でき、また週単位の変化に対応するように拡張することもできます。次の図に、3 つのインスタンスタイプを使用して、このリクエストの量を管理する方法を示します。



この例には次の特徴があります。

- スタックに、常に稼働し、基本負荷を処理する 3 つの 24/7 インスタンスがあります。
- スタックに 12 個の時間ベースのインスタンスがあります。これらは、平均の日単位変動を処理するように設定されています。

1 つは午後 10 時から午前 2 時まで、別の 2 つは午後 8 時から午後 10 時までと午前 2 時から午前 4 時までのように実行されます。わかりやすいように、この図では 2 時間ごとの時間ベースのインスタンスの数に変更していますが、より詳細に管理するために 1 時間当たりの数に変更することもできます。

- スタックには、24/7 および時間ベースのインスタンスで処理できる量を超えたトラフィックの急増を処理するのに十分な負荷ベースのインスタンスがあります。

AWS OpsWorks スタックは、現在実行中のすべてのサーバー全体の負荷が指定されたメトリクスを超えた場合にのみ、ロードベースのインスタンスを起動します。実行されていないインスタンスにかかるコストは最小限 (Amazon EBS でバックアップされたインスタンス) またはゼロ (instance store-backed インスタンス) であるため、推奨される方法としては、予測される最大リクエストボリュームを無理なく処理できるだけのインスタンスを作成します。この例では、スタックに少なくとも 3 個の負荷ベースのインスタンスが必要です。

Note

サービス中断の影響を軽減するために、3 つすべてのインスタンスタイプを複数のアベイラビリティゾーン全体に分散してください。

ベストプラクティス: アクセス権限の管理

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

アカウントのリソースにアクセスするには、AWS 認証情報のフォームが必要です。従業員へのアクセスを提供するための一般的なガイドラインいくつか次に示します。

- 第一に、AWS リソースへのアクセスにアカウントのルート認証情報を使用しないことをお勧めします。

代わりに、従業員に [IAMユーザー](#) を作成し、適切なアクセス権限を付与するポリシーをアタッチします。各従業員は、自分のユーザー認証情報を使用してリソースにアクセスできるようになります。
- 従業員には、ジョブを実行するのに必要なリソースのみにアクセスするためのアクセス権限が必要です。

たとえば、アプリケーション開発者は、アプリケーションを実行するスタックにのみアクセスする必要があります。

- 従業員には、ジョブを実行するのに必要なアクションのみを使用するためのアクセス権限が必要です。

アプリケーション開発者は、開発スタックに対する完全なアクセス権限が必要な場合もあれば、アプリを対応する本稼働用スタックにデプロイするためのアクセス権限が必要な場合もあります。または、本稼働用スタックでインスタンスを起動または停止したり、レイヤーを作成または削除したりするためのアクセス権限は必要ではない場合もあります。

アクセス許可の管理の概要については、「[AWS セキュリティの認証情報](#)」を参照してください。

AWS OpsWorks スタックまたは IAM を使用して、ユーザーのアクセス許可を管理できます。この 2 つのオプションは相互排他的ではありません。両方を使用することが望ましい場合もあります。

AWS OpsWorks スタックのアクセス許可の管理

各スタックには [Permissions] ページがあり、これを使用してスタックにアクセスする権限をユーザーに付与したり、ユーザーが実行できるアクションを指定したりできます。ユーザーのアクセス権限を指定するには、次のいずれかのアクセス権限レベルを設定します。各レベルは、標準のアクションセットにアクセス許可を付与する IAM ポリシーを表します。

- [Deny] は、どのような方法であってもスタックを操作するためのアクセス権限を拒否します。
- [Show] では、任意の方法で、スタック設定を表示するためのアクセス権限が付与されますが、スタック状態を変更するアクセス権限は付与されません。
- [Deploy] では [Show] アクセス権限に加えて、アプリケーションをデプロイすることがユーザーに許可されます。
- [Manage] には [Deploy] アクセス権限が含まれ、また、さまざまなスタック管理アクション (インスタンスやレイヤーの作成または削除など) を実行することがユーザーに許可されます。

Note

アクセス許可の管理レベルでは、スタックの作成やクローン作成など、少数の高レベルの AWS OpsWorks スタックアクションに対するアクセス許可は付与されません。これらのアクセス許可を付与するには、IAM ポリシーを使用する必要があります。

アクセス権限レベルの設定に加え、スタックの [Permissions] ページを使用して、ユーザーに SSH/RDP および sudo/admin 権限を付与できます。AWS OpsWorks スタックのアクセス権限の管理の詳細については、「[スタックごとの権限の付与](#)」を参照してください。SSH アクセスを管理する方法の詳細については、「[SSH アクセスの管理](#)」を参照してください。

IAM アクセス許可の管理

IAM アクセス許可の管理では、IAM コンソール、API、または CLI を使用して、明示的にアクセス許可を指定する JSON 形式のポリシーをユーザーにアタッチします。IAM アクセス許可を管理する方法の詳細については、「[IAM とは何ですか?](#)」を参照してください。

推奨事項： AWS OpsWorks スタックのアクセス許可管理から始めます。ユーザーのアクセス権限の微調整、または Manage アクセス権限レベルに含まれていないアクセス権限をユーザーに付与する場合は、これらの 2 つの方法を組み合わせで使用できます。AWS OpsWorks 次に、スタックは両方のポリシーを評価して、ユーザーのアクセス許可を決定します。

Important

ユーザーに競合するアクセス権限を持つ複数のポリシーがある場合、拒否の権限が常に優先されます。例えば、IAM ポリシーをユーザーに添付して特定のスタックへのアクセスを許可する一方で、[Permissions (アクセス権限)] ページを使用して [Deny (拒否)] アクセス権限レベルをユーザーに割り当てるとします。この場合、Deny アクセス権限レベルが優先され、ユーザーはスタックにアクセスすることができません。詳細については、「[IAM ポリシーの評価論理](#)」を参照してください。

たとえば、レイヤーの追加または削除以外のスタックでのほぼすべての操作をユーザーが実行できるようにする必要があります。

- Manage アクセス権限レベルを指定します。これにより、ユーザーはレイヤーの作成や削除を含むほとんどのスタック管理アクションを実行できるようになります。
- 次の[カスタマー管理ポリシー](#)をユーザーにアタッチします。これにより、そのスタックで [CreateLayer](#) および [DeleteLayer](#) アクションを使用するアクセス許可が拒否されます。スタックは、スタックの [Settings (設定)] ページにある [Amazon Resource Name \(ARN\)](#) (Amazon リソースネーム (ARN)) を使用して識別します。

```
{  
  "Version": "2012-10-17",
```

```
"Statement": [  
  {  
    "Effect": "Deny",  
    "Action": [  
      "opsworks:CreateLayer",  
      "opsworks>DeleteLayer"  
    ],  
    "Resource": "arn:aws:opsworks:*:*:stack/2f18b4cb-4de5-4429-a149-ff7da9f0d8ee/"  
  }  
]
```

詳細とポリシーの例については、「[IAM ポリシーをアタッチして AWS OpsWorks スタックのアクセス許可を管理する](#)」を参照してください。

Note

IAM ポリシーを使用する別の方法としては、指定された IP アドレスまたはアドレス範囲を持つ従業員への、スタックアクセスを制限する条件を設定することです。たとえば、従業員が会社のファイアウォールの内側からのみスタックにアクセスできるようにするには、社内用 IP アドレス範囲へのアクセスを制限する条件を設定します。詳細については、「[条件](#)」を参照してください。

ベストプラクティス: アプリケーションとクックブックの管理とデプロイ

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは、リモートリポジトリから新しいインスタンスにアプリケーションとクックブックをデプロイします。インスタンスのライフタイム中、機能の追加やバグの修正などのために、スタックのオンラインインスタンスでアプリケーションとクックブックを頻繁に更新する必要

があります。スタックのアプリケーションとクックブックの管理にはさまざまな方法がありますが、使用するアプローチは以下の一般的な要件を満たす必要があります。

- すべての本稼働スタックインスタンスには、A/B テストに使用する場合など限られた例外を除いて、同じアプリケーションとカスタムクックブックのコードが必要です。
- 更新のデプロイによって、何か問題が発生しても、サイトのサービスが中断されないことが必要です。

このセクションでは、アプリケーションとカスタムクックブックを管理およびデプロイするためのベストプラクティスについて説明します。

トピック

- [整合性の維持](#)
- [オンラインインスタンスへのコードのデプロイ](#)

整合性の維持

一般的に、本稼働スタックで実行されるアプリケーションとクックブックのコードは厳密に管理する必要があります。また、すべてのインスタンスでは、コードの現在承認されているバージョンを実行する必要があります。ただし例外があり、アプリケーションやクックブックを更新するときや、後述するように、A/B テストの実施など特殊な場合に対応するときは除きます。

アプリケーションとクックブックのコードは、以下の2つの方法で、指定したソースリポジトリからスタックのインスタンスにデプロイされます。

- インスタンスを起動すると、AWS OpsWorks スタックは現在のアプリケーションとクックブックのコードをインスタンスに自動的にデプロイします。
- オンラインインスタンスの場合は、[Deploy コマンド](#) (アプリケーション用) または [Update Custom Cookbooks コマンド](#) (クックブック用) を実行して、現在のアプリケーションまたはクックブックのコードを手動でデプロイする必要があります。

2つのデプロイメカニズムがあるため、重要なのは、違ったコードを別のインスタンスで意図せず実行しないように、ソースコードを慎重に管理することです。例えば、Git マスターブランチからアプリケーションまたはクックブックをデプロイする場合、AWS OpsWorks Stacks はその時点でそのブランチにあるものをデプロイします。マスターブランチにあるコードを更新し、新しいインスタンスを起動した場合、そのインスタンスには、以前のインスタンスにあるコードよりも新しいバージョ

ンのコードがデプロイされます。より新しいバージョンであっても本稼働用として承認されない場合があります。

レコメンデーション: Amazon S3 アーカイブ

すべてのインスタンスに承認済みのコードバージョンがあることを確認するには、Amazon Simple Storage Service (Amazon S3) アーカイブからアプリケーションとクックブックをデプロイすることをお勧めします。これにより、コードが静的なアーティファクト (.zip またはその他のアーカイブファイル) であり、明示的に更新する必要があることが保証されます。さらに、Amazon S3 は信頼性が高いため、アーカイブにアクセスできなくなることはほとんどありません。一貫性をさらに確実に維持するには、命名規則を使用するか、[\[Amazon S3 versioning \(Amazon S3 バージョニング\)\]](#) を使用して、各アーカイブファイルのバージョンを明示的に管理します。これにより監査証跡が提供され、以前のバージョンに簡単に戻せるようになります。

例えば、[\[Jenkins\]](#) などのツールを使用してデプロイパイプラインを作成できます。デプロイするコードがコミットされてテストされた後、アーカイブファイルを作成し、Amazon S3 にアップロードします。すべてのアプリケーションのデプロイやクックブックの更新では、そのアーカイブファイル内のコードがインストールされるため、すべてのインスタンスのコードは同じになります。

推奨事項: Git または Subversion リポジトリ

Git または Subversion リポジトリを使用する場合は、マスターブランチからデプロイしないでください。その代わりに、承認されたバージョンにタグを付け、[アプリケーション](#) または [クックブック](#) ソースとしてそのバージョンを指定します。

オンラインインスタンスへのコードのデプロイ

AWS OpsWorks スタックは、更新されたコードをオンラインインスタンスに自動的にデプロイしません。手動でデプロイする必要がありますが、以下の課題があります。

- デプロイプロセス中、サイトでのお客様のリクエストの処理を中断することなく、更新を効率的にデプロイする。
- デプロイされるアプリケーションまたはクックブックに関する問題やデプロイプロセス自体に関する問題のため失敗したデプロイを処理する。

最も簡単なアプローチは、デフォルトの [Deploy コマンド](#) (アプリケーション用) または [Update Custom Cookbooks コマンド](#) (クックブック用) を実行して、更新をすべてのインスタンスに同時にデプロイすることです。このアプローチは、シンプルで高速ですが、エラーに対する許容範囲があ

りません。デプロイが失敗した場合や、更新されたコードに問題がある場合、本稼働スタックのすべてのインスタンスが影響を受けることがあります。問題が解決されるか、以前のバージョンにロールバックされるまで、サイトのサービスが中断されるか無効になる可能性があります。

推奨事項: 堅牢なデプロイ戦略を使用することをお勧めします。デプロイが成功して、新しいバージョンのコードを実行しているインスタンスにすべての受信トラフィックが転送できるようになるまで、以前のバージョンのコードを実行しているインスタンスでリクエストの処理が継続されるようにします。

以下のセクションでは、堅牢なデプロイ戦略の 2 例を示してから、デプロイ中にバックエンドデータベースを管理する方法について説明します。わかりやすいように、更新されたアプリケーションについて説明していますが、クックブックについても方法は同じです。

トピック

- [ローリングデプロイの使用](#)
- [個別のスタックの使用](#)
- [バックエンドデータベースの管理](#)

ローリングデプロイの使用

ローリングデプロイでは、スタックのオンラインアプリケーションサーバーインスタンスでアプリケーションが複数のフェーズで更新されます。各フェーズでオンラインインスタンスのサブセットを更新し、次のフェーズの開始前に更新が成功していることを確認します。問題が発生した場合、その問題を解決するまで、以前のバージョンのアプリケーションを実行しているインスタンスで受信トラフィックの処理が継続されます。

以下の例では、複数のアベイラビリティーゾーンにまたがってスタックのアプリケーションサーバーインスタンスを配布するというベストプラクティスの使用を前提としています。

ローリングデプロイを実行するには

1. [\[Deploy App\] ページ](#)で、[Advanced] を選択し、1 つのアプリケーションサーバーインスタンスを選択して、そのインスタンスにアプリケーションをデプロイします。

慎重を期す場合は、アプリケーションのデプロイ前に、ロードバランサーからインスタンスを削除できます。これにより、更新されたアプリケーションは正常に動作していることが確認されるまで、ユーザーからアクセスされなくなります。Elastic Load Balancing を使用する場合は、[\[remove the instance \(インスタンスを削除する\)\]](#) Elastic Load Balancing コンソール、CLI、または SDK を使用してロードバランサーから取得します。

2. 更新されたアプリケーションが正常に動作していること、インスタンスのパフォーマンスメトリックスが許容可能であることを確認します。

Elastic ロードバランサーからインスタンスを削除した場合は、Elastic Load Balancing コンソール、CLI、または API を使用してそのインスタンスを復元します。これで、更新されたアプリケーションのバージョンで、ユーザーのリクエストが処理されるようになりました。

3. アベイラビリティゾーン内の残りのインスタンスに更新をデプロイし、それらのインスタンスが正常に動作していること、メトリックスが許容可能であることを確認します。
4. スタックの他のアベイラビリティゾーンに対して、一度に 1 ゾーンずつ、手順 3 を繰り返します。特に注意が必要な場合は、手順 1 から 3 を繰り返します。

Note

Elastic Load Balancing ロードバランサーを使用している場合は、そのヘルスチェックを使用して、デプロイメントが成功したことを確認できます。ただし、[ping パス](#) を設定するときは、依存関係をチェックしてすべてが正常に動作していることを確認するアプリケーションを指定してください。アプリケーションサーバーが動作していることを確認するだけの静的ファイルを指定しないでください。

個別のスタックの使用

アプリケーションを管理するための別のアプローチは、アプリケーションのライフサイクルの各フェーズに別個のスタックを使用することです。この個別のスタックは環境と呼ばれることがあります。このアプローチでは、パブリックアクセス可能でないスタックで開発とテストを行うことができます。更新をデプロイする準備ができたなら、現在のバージョンのアプリケーションのホストするスタックから、更新されたバージョンをホストするスタックに、ユーザートラフィックを切り替えます。

トピック

- [開発、ステージング、本稼働スタックの使用](#)
- [Blue-Green Deployment 戦略の使用](#)

開発、ステージング、本稼働スタックの使用

最も一般的なアプローチでは、以下のスタックを使用します。

開発スタック

開発スタックは、新しい機能の実装やバグの修正などのタスクに使用します。開発スタックは基本的に、本稼働スタックに含まれる同じレイヤー、アプリケーション、リソースなどが属するプロトタイプスタックです。開発スタックでは通常、本稼働スタックと同じ負荷を処理する必要がないため、インスタンスの数やサイズはより小さくなります。

開発スタックは公開されません。以下のようにアクセスを制御します。

- 指定した IP アドレスまたはアドレス範囲からのみ受信リクエストを許可するように、アプリケーションサーバーまたはロードバランサーの[セキュリティグループのインバウンドルール](#)を設定することで、ネットワークアクセスを制限します。

たとえば、HTTP、HTTPS、および SSH アクセスを自社のアドレス範囲内のアドレスに制限します。

- AWS OpsWorks スタックのアクセス[許可ページ](#)を使用して、スタックのスタック管理機能へのアクセスを制御します。

たとえば、Manage アクセス権限レベルを開発チームに、Show アクセス権限を他のすべての従業員に付与します。

ステージングスタック

ステージングスタックは、更新された本稼働スタックの候補のテストと最終処理に使用します。開発が完了したら、[開発スタックを複製](#)することで、ステージングスタックを作成します。その後、ステージングスタックでテストスイートを実行し、そのスタックに更新をデプロイして、発生する問題を修正します。

ステージングスタックも公開されません。開発スタックに対する同じ方法でスタックおよびネットワークアクセスを制御します。開発スタックのクローンを作成してステージングスタックを作成する場合、AWS OpsWorks スタックのアクセス許可管理によって付与されたアクセス許可のクローンを作成できます。ただし複製は、ユーザーの IAM ポリシーによって付与されるアクセス権限には影響を与えません。これらのアクセス権限を変更するには、IAM コンソール、CLI、または SDK を使用する必要があります。詳細については、「[ユーザー許可の管理](#)」を参照してください。

本稼働スタック

本稼働スタックは、現在のアプリケーションをサポートする公開スタックです。ステージングスタックがテストに合格したら、本稼働スタックに昇格させ、古い本稼働スタックを削除します。これを行う方法の例については、「[Blue-Green Deployment 戦略の使用](#)」を参照してください。

Note

AWS OpsWorks スタックコンソールを使用してスタックを手動で作成する代わりに、スタックごとに AWS CloudFormation テンプレートを作成します。このアプローチには以下の利点があります。

- スピードと利便性 – テンプレートを起動すると、AWS CloudFormation によって必要なすべてのインスタンスを含むスタックが自動的に作成されます。
- 一貫性 – 各スタックのテンプレートをソースリポジトリに保存することで、デベロッパーが同じ目的に同じスタックを使用するようになります。

Blue-Green Deployment 戦略の使用

Blue-Green Deployment 戦略は、個別のスタックを効率的に使用して、更新されたアプリケーションを本稼働にデプロイする一般的な方法の 1 つです。

- Blue 環境は、現在のアプリケーションをホストする本稼働スタックです。
- Green 環境は、更新されたアプリケーションをホストするステージングスタックです。

更新されたアプリケーションを本稼働にデプロイする準備ができたなら、Blue スタックから新しい本稼働スタックとなる Green スタックにユーザートラフィックを切り替えます。その後、古い Blue スタックを削除します。

次の例では、AWS OpsWorks Stacks スタックを使用して、[\[Route 53 \(ルート 53\)\]](#) および [\[Elastic Load Balancing load balancers \(Elastic Load Balancing ロードバランサー\)\]](#) のプールを併用して、ブルーグリーンのデプロイを実行する方法を説明します。切り替え前に以下の点を確認してください。

- Green スタックにある更新されたアプリケーションはテストに合格し、本稼働用に準備ができている。
- Green スタックは、更新されたアプリケーションが含まれていること、公開されないことを除いて、Blue スタックと同一である。

両方のスタックでは、アクセス権限、各レイヤーのインスタンスの数とタイプ、[時間ベースと負荷ベース](#)の設定などは同一です。

- Green スタックの 24/7 インスタンスおよびスケジュール済みの時間ベースのインスタンスのすべてがオンラインになっている。

- いずれかのスタックのレイヤーに動的にアタッチでき、[\[pre-warmed](#) (事前にウォーミングアップ)] することができるElastic Load Balancing ロードバランサーのプールがあり、予想されるトラフィックボリュームを処理します。
- Route 53の [\[weighted routing feature](#) (加重ルーティング機能)] を使用して、プールされたロードバランサーを含むホストゾーンにレコードセットを作成しま。
- Blue スタックのアプリケーションサーバーレイヤーにアタッチされているロードバランサーにゼロ以外の重みを割り当て、未使用のロードバランサーにゼロの重みに割り当てている。これにより、Blue スタックのロードバランサーによってすべての受信トラフィックが処理されるようになります。

Green スタックにユーザーを切り替えるには

1. Green スタックのアプリケーションサーバーレイヤーに[プールの未使用のロードバランサーのいずれかをアタッチ](#)します。フラッシュトラフィックを想定できる場合や、トラフィックを徐々に増加させるための負荷テストを設定できない場合など、シナリオによっては、想定したトラフィックを処理できるようにロードバランサーの[事前ウォーミング](#)を行います。
2. Green スタックのすべてのインスタンスが Elastic Load Balancing ヘルスチェックに合格した後、Route 53 レコードセットの重みを変更し、Green スタックのロードバランサーの重みがゼロ以外になり、それに応じて Blue スタックのロードバランサーの重みを減らすようにします。まずは、Green スタックで受信リクエストのごく一部 (多くの場合 5%) が処理され、Blue スタックで残りが処理されるようにすることをお勧めします。これで、2つの本稼働スタックが用意できました。Green スタックで受信リクエストの一部が処理され、Blue スタックで残りが処理されます。
3. Green スタックのパフォーマンスメトリクスを監視します。それらのメトリクスが許容可能であれば、多くの場合、Green スタックの重みを増やして、受信トラフィックの 10% が処理されるようにします。
4. Green スタックで受信トラフィックの約半分が処理されるようになるまで、手順 3 を繰り返します。すべての問題はこの時点までに確認できるため、Green スタックのパフォーマンスメトリクスが許容可能であれば、Blue スタックの重みをゼロまで減らすことで、プロセスを完了できます。これで、Green スタックは新しい Blue スタックになり、このスタックですべての受信トラフィックが処理されるようになります。
5. 古い Blue スタックのアプリケーションサーバーレイヤーから[ロードバランサーをデタッチ](#)し、プールに戻します。
6. 古い Blue スタックは、ユーザーのリクエストの処理に使用されなくなりますが、新しい Blue スタックに問題が発生した場合に備えて、しばらく保持することをお勧めします。その場合は、

古い Blue スタックに受信トラフィックを割り振り直すことで、更新をロールバックできます。新しい Blue スタックのパフォーマンスメトリクスが許容可能になったことを確認できたら、[古い Blue スタックをシャットダウン](#)します。

バックエンドデータベースの管理

アプリケーションがバックエンドデータベースに依存している場合は、古いアプリケーションから新しいアプリケーションに移行する必要があります。AWS OpsWorks スタックでは、次のデータベースオプションがサポートされています。

Amazon RDS Layer

[\[Amazon Relational Database Service \(Amazon RDS\) layer Amazon Relational Database Service \(Amazon RDS\) レイヤー\]](#) では、RDS DB インスタンスを個別に作成し、スタックに登録します。RDS DB インスタンスに登録できるのは一度に 1 つのスタックのみですが、スタック間で RDS DB インスタンスを切り替えることができます。

AWS OpsWorks スタックは、アプリケーションサーバーに接続データを含むファイルを、アプリケーションで簡単に使用できる形式でインストールします。AWS OpsWorks スタックは、データベース接続情報をスタック設定とデプロイ属性に追加し、レシピからアクセスできます。さらに、JSON を使用して、アプリケーションに接続データを渡すこともできます。詳細については、「[データベースへの接続](#)」を参照してください。

データベースに依存するアプリケーションを更新するには、2 つの基本的な課題があります。

- 移行中にすべてのトランザクションを適切に記録すると同時に、アプリケーションの新旧バージョン間の競合状態を回避する。
- サイトのパフォーマンスへの影響を制限し、かつダウンタイムを最小限に抑えることなくす方法で、移行を行う。

このトピックで説明するデプロイの戦略を使用するときは、データベースを古いアプリケーションからデタッチし、新しいアプリケーションに再アタッチするだけでは済みません。アプリケーションの両方のバージョンが移行中に並行して実行され、同じデータにアクセスする必要があります。以下に説明しているのは、移行を管理するための 2 つのアプローチです。いずれのアプローチにも利点と課題があります。

アプローチ 1: 両方のアプリケーションが同じデータベースに接続するようにする

利点

- 移行中のダウンタイムはありません。

データベースへのアクセスを一方のアプリケーションが徐々に停止しながら、他方のアプリケーションが徐々に引き継ぎます。

- 2つのデータベース間でデータを同期する必要はありません。

チャレンジ

- 両方のアプリケーションが同じデータベースにアクセスするため、アクセスを管理してデータの損失や破損を防ぐ必要があります。
- 新しいデータベーススキーマに移行する必要がある場合、古いバージョンのアプリケーションで新しいスキーマを使用できる必要があります。

個別のスタックを使用している場合、インスタンスが特定のスタックに永続的に関連付けられず、異なるスタックで動作するアプリケーションからアクセスできるため、このアプローチはおそらく Amazon RDS に最適です。ただし、一度に複数のスタックに RDS DB インスタンスを登録できないため、JSON などを使用して両方のアプリケーションに接続データを渡す必要があります。詳細については、「[カスタムレシピの使用](#)」を参照してください。

ローリングアップグレードを使用する場合、新旧のアプリケーションバージョンは同じスタックでホストされるため、Amazon RDS または MySQL のいずれかのレイヤーを使用できます。

アプローチ 2: アプリケーションの各バージョンに専用のデータベースを提供する

利点

- 各バージョンに専用のデータベースがあるため、スキーマ間の互換性は不要です。

チャレンジ

- 移行中に 2つのデータベース間でデータを同期して、データの損失や破損がないようにする必要があります。
- 同期手順によりサイトで重大なダウンタイムやパフォーマンスの大幅な低下が生じないようにする必要があります。

個別のスタックを使用する場合、各スタックに専用のデータベースを用意できます。ローリングデプロイを使用する場合、2つのデータベースをスタックにアタッチし、各アプリケーションに1つのデータベースを用意できます。古いアプリケーションと更新されたアプリケーションとの間でデータベーススキーマの互換性がない場合、このアプローチの方が適しています。

[Recommendation: (レコメンデーション:)] 一般的に、アプリケーションのバックエンドデータベースとしては、移行シナリオを問わず、柔軟に適用できる Amazon RDS レイヤーを使用することをお勧めします。移行の処理方法の詳細については、「[Amazon RDS User Guide](#) (Amazon RDS ユーザーガイド)」を参照してください。

ローカルでのクックブックの依存関係のパッケージ化

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Berkshelf を使用してローカルでクックブックの依存関係をパッケージ化し、Amazon S3 にパッケージをアップロードし、スタックを変更して、Amazon S3 パッケージをクックブックのソースとして使用できます。Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、「[S3 バケットを空にする方法](#)」または「[S3 バケットを削除する方法](#)」を参照してください。

以下のチュートリアルでは、クックブックとその依存関係を .zip ファイルに事前にパッケージ化し、AWS OpsWorks スタックの Linux インスタンスのクックブックソースとして .zip ファイルを使用する方法について説明します。最初のチュートリアルでは、1 つのクックブックをパッケージ化する方法を説明します。2 番目のチュートリアルでは、複数のクックブックをパッケージ化する方法について説明します。

始める前に、[Chef Development Kit](#) (Chef DK と呼ばれる) をインストールします。これは Chef コミュニティによって開発されたツールのセットです。ChefDK は chef コマンドラインツールを使用するために必要です。

Chef 12 での依存関係のローカルパッケージ化

Chef 12 Linux では、Berkshelf がスタックインスタンスにデフォルトでインストールされなくなりました。ローカルの開発用コンピュータに Berkshelf をインストールし、これを使用してクックブックの依存関係をローカルにパッケージ化することをお勧めします。依存関係を含めたパッケージを Amazon S3 にアップロードします。最後に、クックブックソースとしてアップロードされたパッ

ページを使用するように Chef Linux 12 のスタックを変更します。Chef 12 でクックブックをパッケージ化する場合は、以下の相違点に注意してください。

1. ローカルコンピュータで、chef コマンドラインツールを使用してクックブックを作成します。

```
chef generate cookbook "server-app"
```

このコマンドによってクックブック、Berksfile、metadata.rb ファイル、およびレシピディレクトリが作成され、クックブックと同じ名前のフォルダに配置されます。次の例は、作成される項目の構造を示しています。

```
server-app <-- the cookbook you've just created
  ### Berksfile
  ### metadata.rb
  ### recipes
```

2. テキストエディタで Berksfile を編集し、server-app クックブックの依存先のクックブックを指定します。この例では server-app が Chef Supermarket の [java](#) クックブックに依存するように指定します。ここでは、バージョン 1.50.0 または以降の新しいマイナーバージョンを指定しますが、任意の発行済みバージョンを単一引用符で囲んで入力できます。変更内容を保存し、ファイルを閉じます。

```
source 'https://supermarket.chef.io'
cookbook 'java', '~> 1.50.0'
```

3. metadata.rb ファイルを編集して、依存関係を追加します。変更内容を保存し、ファイルを閉じます。

```
depends 'java' , '~> 1.50.0'
```

4. Chef で作成された server-app クックブックのディレクトリに変更し、package コマンドを実行して、クックブックの tar ファイルを作成します。複数のクックブックをパッケージ化する場合は、このコマンドを、すべてのクックブックが保存されているルートディレクトリで実行します。1つのクックブックをパッケージ化するには、このコマンドをクックブックディレクトリレベルで実行します。次の例では、このコマンドを server-app ディレクトリで実行します。

```
berks package cookbooks.tar.gz
```

出力は以下のようになります。tar.gz ファイルがローカルディレクトリに作成されます。

```
Cookbook(s) packaged to /Users/username/tmp/berks/cookbooks.tar.gz
```

5. で AWS CLI、先ほど作成したパッケージを Amazon S3 にアップロードします。S3; にアップロードした後でクックブックパッケージの新しい URL を書き留めます。この URL は、スタックの設定で必要になります。

```
aws s3 cp cookbooks.tar.gz s3://bucket-name/
```

出力は以下のようになります。

```
upload: ./cookbooks.tar.gz to s3://bucket-name/cookbooks.tar.gz
```

6. AWS OpsWorks スタックで、クックブックソースとしてアップロードしたパッケージを使用するように [スタックを変更します](#)。
 - a. [Use custom Chef cookbooks] を [Yes] に設定します。
 - b. [Repository type] を [S3 Archive] に設定します。
 - c. [Repository URL] に、ステップ 5 でアップロードしたクックブックパッケージの URL を貼り付けます。

スタックの変更を保存します。

ローカルで 1 つのクックブックの依存関係をパッケージ化する

1. ローカルコンピュータで、以下の chef コマンドラインツールを使用してクックブックを作成します。

```
chef generate cookbook "server-app"
```

このコマンドによってクックブックと Berksfile が作成され、クックブックと同じ名前のフォルダに配置されます。

2. Chef によって作成されたクックブックのディレクトリに変更し、以下のコマンドを実行して、その内容をすべてパッケージ化します。


```
berks package cookbooks.tar.gz
```

出力は次のようになります。

```
Cookbook(s) packaged to /Users/username/tmp/berks/cookbooks.tar.gz
```

3. で AWS CLI、先ほど作成したパッケージを Amazon S3 にアップロードします。

```
aws s3 cp cookbooks.tar.gz s3://bucket-name/
```

出力は次のようになります。

```
upload: ./cookbooks.tar.gz to s3://bucket-name/cookbooks.tar.gz
```

4. AWS OpsWorks スタックで、クックブックソースとしてアップロードしたパッケージを使用するように [スタックを変更します](#)。

ローカルで複数のクックブックの依存関係をパッケージ化する

この例では、2つのクックブックを作成し、それらの依存関係をパッケージ化します。

1. ローカルコンピュータで、以下の chef コマンドを実行して、2つのクックブックを生成します。

```
chef generate cookbook "server-app"  
chef generate cookbook "server-utils"
```

この例では、server-app クックブックによって Java が設定されるため、Java への依存関係を追加する必要があります。

2. コミュニティ Java クックブックへの依存関係を追加するように、server-app/metadata.rb を以下のように編集します。

```
maintainer "The Authors"  
maintainer_email "you@example.com"  
license "all_rights"  
description "Installs/Configures server-app"  
long_description "Installs/Configures server-app"  
version "0.1.0"
```

```
depends "java"
```

3. Berkshelf にパッケージ化対象を指定するように、クックブックのルートディレクトリ内の Berkshelf ファイルを以下のように編集します。

```
source "https://supermarket.chef.io"  
cookbook "server-app", path: "./server-app"  
cookbook "server-utils", path: "./server-utils"
```

ファイル構造は以下のようになります。

```
..  
  ### Berkshelf  
  ### server-app  
  ### server-utils
```

4. 最後に、zip パッケージを作成して Amazon S3 にアップロードし、新しいクックブックソースを使用するように AWS OpsWorks スタックスタックを変更します。そのためには、「[ローカルで1つのクックブックの依存関係をパッケージ化する](#)」の手順 2 ~ 4 に従います。

追加リソース

クックブックの依存関係のパッケージ化の詳細については、以下を参照してください。

- [AWS ブログの「Berkshelf を使用してクックブックの依存関係をローカルにパッケージ化する DevOps」](#)
- AWS OpsWorks フォーラムでの [Linux Chef 12 と Berkshelf](#)
- AWS OpsWorks フォーラムの [Chef 12 の Berkshelf](#)
- このガイドの「[カスタムクックブックのインストール](#)」
- このガイドの「[クックブックリポジトリ](#)」

スタック

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソ

リユースに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

スタックは最上位の AWS OpsWorks スタックエンティティです。これは、アプリケーションを配信するなどの共通の目的があるため、まとめて管理する一連のインスタンスを表します。スタックは、コンテナとして機能すること以外に、アプリケーションとクックブックの管理など、インスタンスのグループ全体に適用されるタスクを処理します。

たとえば、ウェブアプリケーションを提供するスタックは、次のように表示される場合があります。

- 一連のアプリケーションサーバーのインスタンス。各インスタンスが、着信トラフィックのそれぞれの部分を処理します。
- ロードバランサーインスタンス。着信トラフィックを処理し、アプリケーションサーバーに配信します。
- データベースインスタンス。アプリケーションサーバーのバックエンド データストアとなります。

一般的な方法は、異なる環境の複数のスタックを持つことです。一連のスタックは一般的に次のように構成されます。

- 開発スタック。開発者が、機能の追加、バグの修正、他の開発の実行、タスクの管理に使用します。
- ステージングスタック。更新または修正を公開する前に検証します。
- プロダクションスタック。ユーザーからのリクエストを処理するように一般公開されています。

このセクションでは、スタックの取り扱いの基本情報について説明します。

トピック

- [Amazon EC2-Classic からへのスタックの移行 VPC](#)
- [新しいスタックを作成する](#)
- [でのスタックの実行 VPC](#)
- [スタックの更新](#)
- [スタックのクローン化](#)

- [AWS OpsWorks スタックスタックコマンドを実行する](#)
- [カスタム の使用 JSON](#)
- [スタックの削除](#)

Amazon EC2-Classic から へのスタックの移行 VPC

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このトピックでは、Amazon EC2 Classic ネットワークプラットフォームから Amazon [Amazon Virtual Private Cloud](#) (VPC) ネットワークに AWS OpsWorks Stacks スタックを移行する方法について説明します。

2013-12-04 より前に AWS アカウントを作成した場合は、一部の AWS リージョンで EC2-Classic がサポートされている場合があります。拡張ネットワーキングや新しいインスタンスタイプなど、一部の Amazon EC2 リソースと機能には、仮想プライベートクラウド (VPC) が必要です。一部のリソースは EC2-Classic との間で共有できますが VPC、では共有できません。サービスの中断を避けるため、AWS OpsWorks Stacks スタックを VPC に移行することをお勧めします。

トピック

- [前提条件](#)
- [AWS OpsWorks Stacks スタックを VPC に移行する VPC](#)
- [関連情報](#)

前提条件

開始する前に、AWS OpsWorks Stacks 設定 VPC 要件を満たす必要があります。のでプライベートサブネットを設定するには AWS OpsWorks Stacks、このガイドでの [スタックの実行 VPC](#) の VPC 「」を参照してください。Amazon VPC マネジメントコンソール VPC を使用してカスタム を作成できます。詳細については、[「Amazon Virtual Private Cloud ユーザーガイド」](#) の [「Amazon VPC コンソール](#)

[ルウィザードの設定](#)」と「」および「サブネット」を参照してください。 [VPCs](#) Amazon Virtual Private Cloud

移行を続行するには、使用する VPC ID とサブネット ID が必要です。

AWS OpsWorks Stacks スタックを に移行する VPC

まず、AWS OpsWorks Stacks コンソールまたは を使用して、既存の EC2-Classic スタックのクローンを作成しますAPI。続いて、既存のスタックのリソースを新しいスタックに移動します。クローン化されたスタックで新しいインスタンスを起動し、アプリケーションをデプロイします。新しいインスタンスが機能していることを確認します。最後に、EC2-Classic スタックから EC2-Classic リソースを削除してから、古いスタックを削除します。

1. 既存の EC2-Classic スタックを にクローンしますVPC。スタックをクローン化することで、スタック設定、レイヤー、アプリケーション、ユーザー、およびユーザー権限が新しいスタックにコピーされます。スタックをクローン化する方法の詳細については、「[スタックのクローン化](#)」を参照してください。

を使用してスタックのクローンを作成することもできます AWS OpsWorks Stacks API。AWS CLI または を使用してスタックのクローンを作成する場合は AWS SDKs、VpcId/パラメータの値を でVPC作成した の ID に設定します [前提条件](#)。詳細については、AWS OpsWorks Stacks API リファレンスの「[CloneStack](#)」を参照してください。

2. クローン化されたスタックのレイヤーに新しいインスタンスを作成します。 [前提条件](#) で作成したサブネットの ID を必ず指定してください。スタック内にインスタンスを作成する方法の詳細については、このガイドの「[レイヤーへのインスタンスの追加](#)」を参照してください。
3. EC2 セキュリティグループ、Elastic Load Balancing ロードバランサー、Elastic IP アドレスなどのクラシックリソースを に移行しVPC、クローンされたスタックに関連付けます。詳細については、「[Amazon EC2ユーザーガイド](#)」の「[リソースを に移行するVPC](#)」を参照してください。
4. クローンされたスタックに Amazon EBSボリュームと Amazon RDSインスタンスを登録します。スタックにリソースを登録することについての詳細は、このガイドの「[スタックにリソースを登録する](#)」を参照してください。

Amazon EBSボリュームは に関連付けられておらずVPC、 の EC2-Classic スタックと スタックの両方のインスタンスで使用できますVPC。EC2-Classic の Amazon RDSインスタンスは、 の EC2-Classic スタックと スタックの両方に登録できますVPC。

5. クローン化されたスタックでインスタンスを起動し、ワークロードのごく一部をクローンされたスタックに移動します。例えば、トラフィックのごく一部をクローン化されたスタック内の

Elastic Load Balancing ロードバランサーに移動します。Amazon Route 53 を使用している場合は、「Amazon Route 53 デベロッパーガイド」の[ELB「ロードバランサーへのトラフィックのルーティング」](#)を参照してください。

新しいスタックが機能し、アプリケーションをサポートしていることを確認するまで、ごく一部のトラフィックだけをルーティングします。一週間の試用期間などに、新しいスタックをごく一部のトラフィックで動作させます。新しいスタックが動作していることを確認後、残りのトラフィックをスタックにルーティングします。

- クローン化されたスタックが動作していることを確認後、残りの本番トラフィックまたはワークロードをクローンされたスタックに移動します。EC2-Classic スタックでインスタンスを停止できるようになりました。移行後数週間に新しいスタックで問題が発生した場合、ワークロードを古いスタックに戻すことができるように、古いスタックを数週間利用可能な状態にしておくことをお勧めします。
- 新しいスタックが数週間動作している場合は、EC2-Classic スタックのインスタンスを削除します。インスタンスの削除方法の詳細については、このガイドの「[AWS OpsWorks スタックインスタンスの削除](#)」を参照してください。

Important

Amazon EC2コンソールまたは [AWS CLI](#) を使用してAPI、インスタンスを停止または削除 AWS OpsWorks しないでください。

- EC2-Classic スタックでアプリケーションを削除します。アプリケーションの削除方法の詳細については、このガイドの「[アプリケーションをスタックから削除するには](#)」を参照してください。
- EC2-Classic スタックを削除します。スタックを削除する方法の詳細については、このガイドの「[スタックの削除](#)」を参照してください。

関連情報

- [EC2-Classic からへの移行 VPC](#)
- [デバッグとトラブルシューティングのガイド](#)
- [でのスタックの実行 VPC](#)

新しいスタックを作成する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

新しいスタックを作成するには、AWS OpsWorks スタックダッシュボードでスタックの追加をクリックします。[Add Stack] ページで、スタックを設定します。完了したら、[Add Stack] をクリックします。

トピック

- [作成するスタックのタイプを選択します。](#)
- [基本オプション](#)
- [詳細オプション](#)

作成するスタックのタイプを選択します。

スタックを作成する前に、作成するスタックのタイプを決定する必要があります。ヘルプについては次の表を参照してください。

作成するもの	次の場合に作成するスタックのタイプ	方法については、以下の手順に従います
サンプルスタック	Linux ベースの Chef 12 スタックとサンプル Node.js アプリAWS OpsWorks を使用して、の基本について説明します。	使用開始: サンプル
Linux ベースの Chef 12 スタック	がAWS OpsWorks サポートする最新バージョンの Chef	入門ガイド: Linux

作成するもの	次の場合に作成するスタックのタイプ	方法については、以下の手順に従います
	<p>を使用する Linux ベースのスタックを作成します。豊富なコミュニティ cookbook を活用するか、独自のカスタム cookbook を作成することを希望する高度な Chef ユーザーは、このオプションを選択します。詳細については、「Chef 12 Linux」を参照してください。</p>	
Windows ベースの Chef 12.2 スタック	Windows ベースのスタックを作成します。	入門ガイド: Windows
Linux ベースの Chef 11.10 スタック	組織で後方互換性のため Chef 11.10 と Linux の使用が必要な場合は、このスタックを作成します。	Chef 11 Linux スタックの使用開始

基本オプション

[Add Stack] ページには、以下の基本オプションがあります。

スタックの名前

(必須) AWS OpsWorks スタックコンソールでスタックを識別するために使用される名前。名前は一意である必要はありません。AWS OpsWorks スタックはスタック ID も生成します。これはスタックを一意的に識別する GUID です。例えば、[update-stack](#) などの [AWSCLI](#) コマンドでは、スタック ID を使用して特定のスタックを識別します。スタックの作成後、ナビゲーションペインで [Stack] を選択してから、[Stack Settings] を選択することで、その ID を見つけることができます。ID には OpsWorks ID というラベルが付けられます。

リージョン

(必須) インスタンスを起動する AWS リージョン。

VPC

(オプション) スタックを起動VPCする先の の ID。すべてのインスタンスはこの で起動され VPC、後で ID を変更することはできません。

- アカウントが EC2 Classic をサポートしている場合は、 を使用しない場合は No VPC (デフォルト値) を指定できますVPC。

EC2 Classic の詳細については、[「サポートされているプラットフォーム」](#)を参照してください。

- アカウントが EC2 Classic をサポートしていない場合は、 を指定する必要がありますVPC。

デフォルト設定はデフォルトです。これは VPCClassic の使いやすさEC2とVPCネットワーク機能の利点を組み合わせたものです。通常の でスタックを実行する場合はVPC、[VPCコンソール](#)、[API](#)または を使用してスタックを作成する必要があります[CLI](#)。AWS OpsWorks スタックVPCスタックの を作成する方法の詳細については、「」を参照してください[でのスタックの実行 VPC](#)。一般的な情報については、[「Amazon Virtual Private Cloud」](#)を参照してください。

Default Availability Zone/Default subnet

(オプション) この設定は、 でスタックを作成するかどうかによって異なりますVPC。

- アカウントが EC2 Classic をサポートしており、 VPCを No VPCに設定した場合、この設定には、インスタンスを起動するデフォルトのアベイラビリティゾーンを指定するデフォルトのAWSアベイラビリティゾーンというラベルが付けられます。
- アカウントが EC2 Classic をサポートしていない場合、または を指定することを選択した場合 VPC、このフィールドには、インスタンスを起動するデフォルトサブネットを指定するデフォルトサブネットというラベルが付けられます。インスタンスの作成時にこの値を上書きすることによって、他のサブネットのインスタンスを起動できます。各サブネットは1つのアベイラビリティゾーンに関連付けられます。

インスタンスの作成時にこの設定を上書きすることで、AWS OpsWorks スタックで別のアベイラビリティゾーンまたはサブネットでインスタンスを起動させることができます。[レイヤーへのインスタンスの追加](#)

でスタックを実行する方法の詳細についてはVPC、「」を参照してください[でのスタックの実行 VPC](#)。

Default operating system

(オプション) デフォルトで各インスタンスにインストールされるオペレーティングシステム。次のオプションがあります。

- 組み込みの Linux オペレーティングシステムのいずれか。
- Microsoft Windows Server 2012 R2.
- サポートされているオペレーティングシステムのいずれかAMIに基づくカスタム。

カスタム の使用 AMIを選択した場合、オペレーティングシステムはインスタンスの作成時に AMI指定したカスタム によって決まります。詳細については、「[カスタム AMI の使用](#)」を参照してください。

使用可能なオペレーティングシステムの詳細については、「[AWS OpsWorks スタックオペレーティングシステム](#)」を参照してください。

Note

インスタンスの作成時、デフォルトのオペレーティングシステムを上書きできます。ただし、Linux オペレーティングシステムを上書きして Windows を指定したり、Windows オペレーティングシステムを上書きして Linux を指定したりはできません。

デフォルトSSHキー

(オプション) スタックのリージョンからの Amazon EC2キーペア。デフォルト値は none です。キーペアを指定すると、AWS OpsWorks スタックはインスタンスにパブリックキーをインストールします。

- Linux インスタンスでは、SSHクライアントでプライベートキーを使用してスタックのインスタンスにログインできます。

詳細については、「[SSHでのログイン](#)」を参照してください。

- Windows インスタンスでは、Amazon EC2コンソールまたは でプライベートキーを使用して、インスタンスの管理者パスワードCLIを取得できます。

その後、そのパスワードを RDPクライアントで使用して、管理者としてインスタンスにログインできます。詳細については、「[RDPでのログイン](#)」を参照してください。

SSH キーの管理方法の詳細については、「」を参照してください[SSH アクセスの管理](#)。

Note

この設定を上書きするには、[インスタンスの作成するときに別のキーペアを指定するか、キーペアを指定しません。](#)

Chef version

これは、選択した Chef バージョンを示します。

Chef のバージョンの詳細については、「[Chef のバージョン](#)」を参照してください。

Use custom Chef cookbooks

スタックのインスタンスにカスタム Chef クックブックをインストールするかどうか。

Chef 12 の場合、デフォルト設定は [Yes] です。Chef 11 の場合、デフォルト設定は No です。Yes オプションには、AWS OpsWorks リポジトリなど、カスタムクックブックをリポジトリからスタックのインスタンスにデプロイするために必要な情報をスタックに提供するいくつかの追加設定が表示されますURL。詳細は、クックブックに使用しているリポジトリによって異なります。詳細については、「[カスタムクックブックのインストール](#)」を参照してください。

Stack color

(オプション) AWS OpsWorks スタックコンソールでスタックを表すために使用される色相。異なるスタックに異なる色を使用することによって、開発、ステージング、本稼動用の各スタックなど、スタックを区別しやすくすることができます。

スタックタグ

スタックおよびレイヤーレベルでタグを適用することができます。タグを作成する場合、タグ付けされた構造内すべてのリソースにタグを適用することになります。例えば、スタックにタグを適用する場合、タグをすべてのレイヤーに適用し、各レイヤー内で、レイヤー内のすべてのインスタンス、Amazon EBSボリューム、または Elastic Load Balancing ロードバランサーに適用します。タグをアクティブ化し、それを使用して AWS OpsWorks スタックリソースのコストを追跡および管理する方法の詳細については、「[Billing and Cost Management ユーザーガイド](#)」の「[コスト配分タグの使用](#)」および「[ユーザー定義のコスト配分タグのアクティブ化](#)」を参照してください。AWS OpsWorks スタックでのタグ付けの詳細については、「」を参照してください [タグ](#)。

詳細オプション

詳細設定の場合、[Advanced >>] をクリックして、[Advanced options] および [Security] セクションを表示します。

[Advanced options] セクションには、以下のオプションがあります。

デフォルトのルートデバイスのタイプ。

インスタンスのルートボリュームに使用されるストレージのタイプ。詳細については、[「ストレージ」](#)を参照してください。

- Linux スタックはデフォルトで Amazon EBS-backed ルートボリュームを使用しますが、instance store-backed ルートボリュームを指定することもできます。
- Windows スタックでは、Amazon EBS-backed ルートボリュームを使用する必要があります。

IAM ロール

(オプション) スタックがAWSユーザーに代わってとやり取りするために使用する、スタックのAWS Identity and Access Management (IAM) AWS OpsWorks ロール。

デフォルトのIAMインスタンスプロファイル

(オプション) スタックの Amazon EC2 インスタンスに関連付けるデフォルトの[IAMロール](#)。このロールは、スタックのインスタンスで実行されているアプリケーションに、S3 バケットなどのAWSリソースにアクセスするためのアクセス許可を付与します。

- アプリケーションに特定のアクセス権限を付与するには、該当するポリシーが割り当てられた既存のインスタンスプロファイル (ロール) を選択します。
- 初期状態では、プロファイルのロールはアクセス許可を付与しませんが、IAMコンソール、API、または [を使用](#)して適切なポリシーCLIをアタッチできます。詳細については、「[EC2 インスタンスで実行されるアプリケーションのアクセス許可の指定](#)」を参照してください。

API エンドポイントリージョン

この設定は、スタックの基本設定で選択されたリージョンの値を使用します。次のリージョンのエンドポイントから選択できます。

- 米国東部 (バージニア北部) リージョン
- 米国東部 (オハイオ) リージョン
- 米国西部 (オレゴン) リージョン
- US West (N. California) リージョン

- カナダ (中部) リージョン (API のみ。で作成されたスタックでは使用できません AWS Management Console
- アジアパシフィック (ムンバイ) リージョン
- アジアパシフィック (シンガポール) リージョン
- アジアパシフィック (シドニー) リージョン
- アジアパシフィック (東京) リージョン
- アジアパシフィック (ソウル) リージョン
- 欧州 (フランクフルト) リージョン
- 欧州 (アイルランド) リージョン
- 欧州 (ロンドン) リージョン
- 欧州 (パリ) リージョン
- 南米 (サンパウロ) リージョン

あるAPIエンドポイントで作成されたスタックは、別のAPIエンドポイントでは使用できません。AWS OpsWorks スタックユーザーもリージョン固有であるため、これらのエンドポイントリージョンの AWS OpsWorks スタックユーザーに別のエンドポイントリージョンのスタックを管理させる場合は、スタックが関連付けられているエンドポイントにユーザーをインポートする必要があります。ユーザーのインポートの詳細については、「[AWS OpsWorks スタックへのユーザーのインポート](#)」を参照してください。

Hostname theme

(オプション) 各インスタンスのデフォルトのホスト名を生成するために使用される文字列。デフォルト値は [Layer Dependent] で、インスタンスのレイヤーの短い名前を使用し、各インスタンスに一意の数値を追加します。たとえば、ロールに依存する Load Balancer のテーマのルートが「lb」であるとし、レイヤーに追加する最初のインスタンスには「lb1」、2 番目のインスタンスには「lb2」のように名前が付けられます。

OpsWorks エージェントバージョン

(オプション) 新しいバージョンが利用可能になったときに AWS OpsWorks スタックエージェントを自動的に更新するか、指定されたエージェントバージョンを使用して手動で更新するか。この機能は Chef 11.10 スタックおよび Chef 12 スタックでのみ使用できます。デフォルト設定は [Manual update] であり、最新のエージェントバージョンに設定されます。

AWS OpsWorks スタックは、サービスと通信する各インスタンスにエージェントをインストールし、[ライフサイクルイベント](#)に応じて Chef の実行を開始するなどのタスクを処理します。この

エージェントは定期的に更新されます。スタックのエージェントバージョンを指定するときは、次の 2 つのオプションがあります。

- 自動更新 – AWS OpsWorks スタックは、更新が利用可能になるとすぐに、スタックのインスタンスに新しい各エージェントバージョンを自動的にインストールします。
- 手動更新 – AWS OpsWorks スタックは、指定されたエージェントバージョンをスタックのインスタンスにインストールします。

AWS OpsWorks 新しいエージェントバージョンが利用可能になると、スタックはスタックページにメッセージを投稿しますが、スタックのインスタンスは更新されません。エージェントを更新するには、[スタック設定を手動で更新](#)して新しいエージェントバージョンを指定する必要があります。そうすると AWS OpsWorks、スタックはスタックのインスタンスを更新します。

特定のインスタンスのデフォルトのOpsWorks エージェントバージョン設定は、[設定を更新すること](#)で上書きできます。この場合、そのインスタンスの設定が優先されます。たとえば、デフォルト設定は [Auto-update] ですが、特定のインスタンスに対して [Manual update] を指定しているとします。AWS OpsWorks スタックが新しいエージェントバージョンをリリースすると、手動更新に設定されているものを除き、スタックのすべてのインスタンスが自動的に更新されます。特定のインスタンスに新しいエージェントバージョンをインストールするには、手動で[そのインスタンスの設定を更新](#)し、新しいバージョンを指定する必要があります。

Note

コンソールに、短縮されたエージェントバージョン番号が表示されます。完全なバージョン番号を確認するには、AWS CLI [describe-agent-versions](#) コマンド、または同等の API メソッドまたは SDKメソッドを呼び出します。それにより、使用可能なエージェントバージョンの完全なバージョン番号が返されます。

カスタム JSON

(オプション) JSON構造としてフォーマットされた 1 つ以上のカスタム属性。これらの属性は、すべてのインスタンスにインストールされる[スタック設定とデプロイ属性](#)にマージされ、レシピで使用できます。例えばJSON、カスタム を使用して、デフォルト設定を指定する組み込み属性を上書きすることで、設定をカスタマイズできます。詳細については、「[カスタムの使用 JSON](#)」を参照してください。

セキュリティには、OpsWorks セキュリティグループを使用するという 1 つのオプションがあります。これにより、AWS OpsWorks スタックの組み込みセキュリティグループをスタックのレイヤーに関連付けるかどうかを指定できます。

AWS OpsWorks スタックには、デフォルトでレイヤーに関連付けられている、レイヤーごとに 1 つずつの組み込みセキュリティグループの標準セットが用意されています。セキュリティ OpsWorks グループを使用すると、代わりに独自のカスタムセキュリティグループを指定できます。詳細については、「[セキュリティグループの使用](#)」を参照してください。

OpsWorks セキュリティグループの使用には、次の設定があります。

- はい - AWS OpsWorks スタックは、適切な組み込みセキュリティグループを各レイヤーに自動的に関連付けます (デフォルト設定)。

追加のセキュリティグループを作成した後、レイヤーに関連付けることができますが、組み込みセキュリティグループを削除することはできません。

- いいえ - AWS OpsWorks スタックは組み込みセキュリティグループをレイヤーに関連付けません。

適切な EC2 セキュリティグループを作成し、作成する各レイヤーにセキュリティグループを関連付ける必要があります。ただし、作成時に組み込みセキュリティグループと層を手動で関連付けることもできます。カスタムセキュリティグループは、カスタム設定を必要とする層にのみ必要です。

次の点に注意してください:

- OpsWorks セキュリティグループの使用が「はい」に設定されている場合、より制限の厳しいセキュリティグループをレイヤーに追加して、デフォルトのセキュリティグループのポートアクセス設定を制限することはできません。複数のセキュリティグループでは、Amazon は最も許容度の高い設定 EC2 を使用します。さらに、組み込みのセキュリティグループの設定を変更して、より制限の厳しい設定を作成することはできません。スタックを作成すると、AWS OpsWorks スタックによって組み込みセキュリティグループの設定が標準設定で上書きされるため、次回スタックを作成するときに行った変更は失われます。レイヤーで組み込みのセキュリティグループよりも制限の厳しいセキュリティグループ設定が必要な場合は、OpsWorks 「セキュリティグループを使用」を「いいえ」に設定し、任意の設定でカスタムセキュリティグループを作成し、作成時にレイヤーに割り当てます。
- AWS OpsWorks スタックのセキュリティグループを誤って削除して再作成する場合は、グループ名の大文字と小文字を含め、元のセキュリティグループと完全に重複している必要があります。グループを手動で再作成する代わりに、AWS OpsWorks スタックによってこのタスクが実行される

ようにすることをお勧めします。同じAWSリージョン、および存在する場合は新しいスタックを作成するだけでVPC、削除したセキュリティグループを含むすべての組み込みセキュリティグループが AWS OpsWorks スタックによって自動的に再作成されます。その後、今後使用しないスタックを削除します。セキュリティグループは残ります。

でのスタックの実行 VPC

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

スタックのインスタンスへのユーザーアクセスは、仮想プライベートクラウド (VPC) で作成することで制御できます。たとえば、スタックのアプリケーションサーバーまたはデータベースにユーザーが直接アクセスできるようにしたくない場合、代わりに、すべてのパブリックトラフィックが Elastic Load Balancing を経由するようにします。

でスタックを実行する基本的な手順VPCは次のとおりです。

1. Amazon VPCコンソール、VPC、または AWS CloudFormation テンプレートを使用してAPI、適切に設定された VPC を作成します。
2. スタックの作成時に VPC ID を指定します。
3. 適切なサブネットでスタックのインスタンスを起動します。

以下に、AWS OpsWorks スタックでの VPCsの動作について簡単に説明します。

Important

VPC エンドポイント機能を使用する場合は、スタック内の各インスタンスが Amazon Simple Storage Service (Amazon S3) から次のアクションを実行できる必要があることに注意してください。

- インスタンスエージェントをインストールする。

- Ruby などの資産をインストールする。
- Chef の実行ログをアップロードする。
- スタックコマンドを取得する。

これらのアクションを有効にするには、スタックのインスタスが、スタックのリージョンと一致する、以下のバケットにアクセスできる必要があります。それ以外の場合、前述のオペレーションは失敗します。

Chef 12 Linux および Chef 12.2 Windows の場合、バケットは以下のとおりです。

エージェントバケッ ト	資産バケッ ト	ログバケッ ト	DNA バケッ ト
• opsworks- instance-agent- sa-east-1	• opsworks- instance-assets- us-east-2	• opsworks-us- east-2-log	• opsworks-us- east-2-dna
• opsworks- instance-agent- ap-south-1	• opsworks- instance-assets- us-east-1	• opsworks-us- east-1-log	• opsworks-us- east-1-dna
• opsworks- instance-agent- ap-northeast-1	• opsworks- instance-assets- ap-south-1	• opsworks-ap- south-1-log	• opsworks-ap- south-1-dna
• opsworks- instance-agent- ap-northeast-2	• opsworks- instance-assets- ap-northeast-1	• opsworks-ap- northeast-1-log	• opsworks-ap- northeast-1-dna
• opsworks- instance-agent- ap-southeast-1	• opsworks- instance-assets- ap-northeast-2	• opsworks-ap- northeast-2-log	• opsworks-ap- northeast-2-dna
• opsworks- instance-agent- ap-southeast-2	• opsworks- instance-assets- ap-southeast-1	• opsworks-ap- southeast-1-log	• opsworks-ap- southeast-1-dna
• opsworks- instance-agent- ca-central-1	• opsworks- instance-assets- ap-southeast-2	• opsworks-ap- southeast-2-log	• opsworks-ap- southeast-2-dna
• opsworks- instance-agent- eu-central-1	• opsworks- instance-assets- ca-central-1	• opsworks-ca- central-1-log	• opsworks-ca- central-1-dna
• opsworks- instance-agent- eu-west-1	• opsworks- instance-assets- eu-central-1	• opsworks-eu- central-1-log	• opsworks-eu- central-1-dna
• opsworks- instance-agent- eu-west-2	• opsworks- instance-assets- eu-west-1	• opsworks-eu- west-1-log	• opsworks-eu- west-1-dna
		• opsworks-eu- west-2-log	• opsworks-eu- west-2-dna
		• opsworks-eu- west-3-log	• opsworks-eu- west-3-dna
		• opsworks-sa- east-1-log	• opsworks-sa- east-1-dna
		• opsworks-us- west-1-log	• opsworks-us- west-1-dna
		• opsworks-us- west-2-log	• opsworks-us- west-2-dna

エージェントバケッ ト	資産バケッ ト	ログバケッ ト	DNA バケッ ト
<ul style="list-style-type: none"> • opsworks- instance-agent- eu-west-3 	<ul style="list-style-type: none"> • opsworks- instance-assets- eu-west-2 		
<ul style="list-style-type: none"> • opsworks- instance-agent- us-east-1 	<ul style="list-style-type: none"> • opsworks- instance-assets- eu-west-3 		
<ul style="list-style-type: none"> • opsworks- instance-agent- us-east-2 	<ul style="list-style-type: none"> • opsworks- instance-assets- sa-east-1 		
<ul style="list-style-type: none"> • opsworks- instance-agent- us-west-1 	<ul style="list-style-type: none"> • opsworks- instance-assets- us-west-1 		
<ul style="list-style-type: none"> • opsworks- instance-agent- us-west-2 	<ul style="list-style-type: none"> • opsworks- instance-assets- us-west-2 		

Linux 用 Chef 11.10 以前のバージョンの場合、バケッ
トは次のとおりです。Chef 11.4 ス
タックは、米国東部 (バージニア北部) リージョ
ン 以外のリージョンエンドポイントではサ
ポートされていません。

エージェントバケッ ト	資産バケッ ト	ログバケッ ト	DNA バケッ ト
<ul style="list-style-type: none"> • opsworks-instance-agent-us-east-2 • opsworks-instance-agent-us-east-1 • opsworks-instance-agent-ap-south-1 • opsworks-instance-agent-ap-northeast-1 • opsworks-instance-agent-ap-northeast-2 • opsworks-instance-agent-ap-southeast-1 • opsworks-instance-agent-ap-southeast-2 • opsworks-instance-agent-ca-central-1 • opsworks-instance-agent-eu-central-1 • opsworks-instance-agent-eu-west-1 	<ul style="list-style-type: none"> • opsworks-instance-assets-us-east-2 • opsworks-instance-assets-us-east-1 • opsworks-instance-assets-ap-south-1 • opsworks-instance-assets-ap-northeast-1 • opsworks-instance-assets-ap-northeast-2 • opsworks-instance-assets-ap-southeast-1 • opsworks-instance-assets-ap-southeast-2 • opsworks-instance-assets-ca-central-1 • opsworks-instance-assets-eu-central-1 • opsworks-instance-assets-eu-west-1 	<ul style="list-style-type: none"> • prod_stage-log 	<ul style="list-style-type: none"> • prod_stage-dna

エージェントバケッ ト	資産バケッ ト	ログバケッ ト	DNA バケッ ト
<ul style="list-style-type: none">• opsworks- instance-agent- eu-west-2	<ul style="list-style-type: none">• opsworks- instance-assets- eu-west-2		
<ul style="list-style-type: none">• opsworks- instance-agent- eu-west-3	<ul style="list-style-type: none">• opsworks- instance-assets- eu-west-3		
<ul style="list-style-type: none">• opsworks- instance-agent- us-east-1	<ul style="list-style-type: none">• opsworks- instance-assets- sa-east-1		
<ul style="list-style-type: none">• opsworks- instance-agent- us-west-1	<ul style="list-style-type: none">• opsworks- instance-assets- us-west-1		
<ul style="list-style-type: none">• opsworks- instance-agent- us-west-2	<ul style="list-style-type: none">• opsworks- instance-assets- us-west-2		

詳細については、「[VPC エンドポイント](#)」を参照してください。

Note

AWS OpsWorks スタックを有効にしたVPCエンドポイントに接続するには、NATまたはパブリック IP AWS OpsWorks のルーティングも設定する必要があります。スタックエージェントは引き続きパブリックエンドポイントにアクセスする必要があります。

トピック

- [VPC の基礎](#)
- [AWS OpsWorks スタックVPCスタック用の を作成する](#)

VPC の基礎

の詳細についてはVPCs、[Amazon Virtual Private Cloud](#)」を参照してください。簡単に言うと、VPCは1つ以上のサブネット構成され、各サブネットには1つ以上のインスタンスが含まれます。また、各サブネットには、送信先IPアドレスに基づいてアウトバウンドトラフィックを送信する、関連付けられたルーティングテーブルがあります。

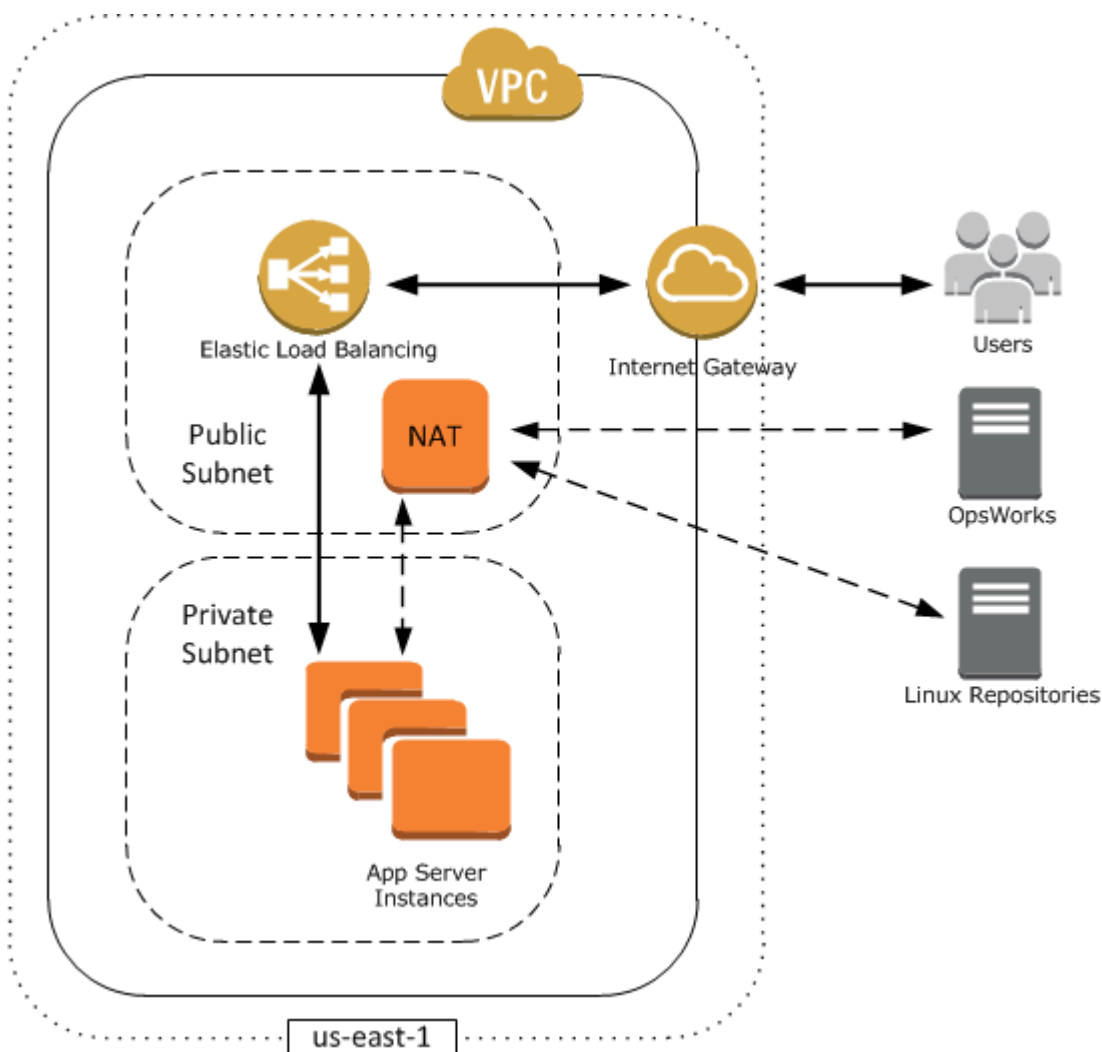
- 内のインスタンスVPCは、サブネットに関係なく、デフォルトで相互に通信できます。ただし、ネットワークアクセスコントロールリスト (ACLs)、セキュリティグループポリシー、または静的IPアドレスの使用を変更すると、この通信が中断される可能性があります。
- サブネットのインスタンスがインターネットと通信できる場合、このサブネットはパブリックサブネットと呼ばれます。
- インスタンスが内の他のインスタンスとのみ通信できVPC、インターネットと直接通信できないサブネットは、プライベートサブネットと呼ばれます。

AWS OpsWorks スタックではVPC、プライベートサブネットのインスタンスを含むスタック内のすべてのインスタンスが次のエンドポイントにアクセスできるようにを設定する必要があります。

- 「 」の「リージョンサポート」セクションに記載されている AWS OpsWorks スタックサービスエンドポイントの1つ[AWS OpsWorks スタックの開始方法](#)。
- AWS OpsWorks スタックエージェントで使用される、次のいずれかのインスタンスサービスエンドポイント。エージェントは、マネージドカスタマーインスタンスで実行され、サービスとデータを交換します。
 - opsworks-instance-service.us-east-2.amazonaws.com
 - opsworks-instance-service.us-east-1.amazonaws.com
 - opsworks-instance-service.us-west-1.amazonaws.com
 - opsworks-instance-service.us-west-2.amazonaws.com
 - opsworks-instance-service.ap-south-1.amazonaws.com
 - opsworks-instance-service.ap-northeast-1.amazonaws.com
 - opsworks-instance-service.ap-northeast-2.amazonaws.com
 - opsworks-instance-service.ap-southeast-1.amazonaws.com
 - opsworks-instance-service.ap-southeast-2.amazonaws.com
 - opsworks-instance-service.ca-central-1.amazonaws.com
 - opsworks-instance-service.eu-central-1.amazonaws.com

- opsworks-instance-service.eu-west-1.amazonaws.com
- opsworks-instance-service.eu-west-2.amazonaws.com
- opsworks-instance-service.eu-west-3.amazonaws.com
- Amazon S3
- オペレーティングシステムに応じたパッケージリポジトリ (Amazon Linux や Ubuntu Linux のリポジトリなど)。
- 使用するアプリケーションとカスタムクックブックのリポジトリ

この接続を提供するVPCようにを設定するには、さまざまな方法があります。以下は、スタックアプリケーションサーバースタックVPC用に AWS OpsWorks を設定する方法の簡単な例です。



これにはいくつかのコンポーネントVPCがあります。

サブネット

VPC には、パブリックとプライベートの 2 つのサブネットがあります。

- パブリックサブネットには、ロードバランサーとネットワークアドレス変換 (NAT) デバイスが含まれており、外部アドレスやプライベートサブネットのインスタンスと通信できます。
- プライベートサブネットには、パブリックサブネットの NAT およびロードバランサーと通信できるが、外部アドレスと直接通信できないアプリケーションサーバーが含まれています。

インターネットゲートウェイ

インターネットゲートウェイを使用すると、ロードバランサーなどのパブリック IP アドレスを持つインスタンスは、外部のアドレスと通信できますVPC。

ロードバランサー

Elastic Load Balancing ロードバランサーは、ユーザーからの着信トラフィックを処理し、プライベートサブネットのアプリケーションサーバーに配信して、ユーザーにレスポンスを返します。

NAT

(NAT) デバイスは、アプリケーションサーバーにインターネットアクセスを制限します。これは通常、外部リポジトリからソフトウェア更新をダウンロードするなどの目的で使用されます。すべての AWS OpsWorks スタックインスタンスは、AWS OpsWorks スタックおよび適切な Linux リポジトリと通信する必要があります。この問題を処理する 1 つの方法は、関連付けられた Elastic IP アドレスを持つ NAT デバイスをパブリックサブネットに配置することです。その後、プライベートサブネット内のインスタンスからのアウトバウンドトラフィックを 経由でルーティングできます NAT。

Note

1 つの NAT インスタンスで、プライベートサブネットのアウトバウンドトラフィックに単一障害点が作成されます。障害が発生した場合に相互に引き継ぐ NAT インスタンスのペア VPC で を設定することで、信頼性を向上させることができます。詳細については、[「Amazon VPC NAT インスタンスの高可用性」](#)を参照してください。NAT ゲートウェイを使用することもできます。詳細については、[「Amazon VPC ユーザーガイド NAT」](#)の「」を参照してください。

最適な VPC 設定は AWS OpsWorks スタックスタックによって異なります。以下は、特定の VPC 設定を使用する場合の例です。その他の VPC シナリオの例については、[「Amazon を使用するシナリオ VPC」](#)を参照してください。

パブリックサブネットで1つのインスタンスを使用する

パブリックにアクセスできない Amazon インスタンスなど、プライベートリソースが関連付けられていない単一RDSインスタンススタックがある場合は、1つのパブリックサブネットVPCを持つを作成し、そのサブネットにインスタンスを配置できます。デフォルトのを使用していない場合はVPC、インスタンスのレイヤーに Elastic IP アドレスを割り当てる必要があります。詳細については、「[OpsWorks レイヤーの基本](#)」を参照してください。

プライベートリソースを使用する

パブリックにアクセスできないリソースがある場合は、1つのパブリックサブネットと1つのプライベートサブネットVPCを持つを作成できます。例えば、負荷分散された自動スケーリング環境では、すべての Amazon EC2 インスタンスをプライベートサブネットに配置し、ロードバランサーをパブリックサブネットに配置することができます。これにより、Amazon EC2 インスタンスにインターネットから直接アクセスすることはできません。すべての受信トラフィックはロードバランサーを介してルーティングする必要があります。

プライベートサブネットはインスタンスを Amazon EC2 の直接ユーザーアクセスから分離しますが、アウトバウンドリクエストを AWS および適切な Linux パッケージリポジトリに送信する必要があります。このようなリクエストを許可するには、例えば、独自の Elastic IP アドレスを持つネットワークアドレス変換 (NAT) デバイスを使用し、インスタンスのアウトバウンドトラフィックを経由でルーティングします NAT。前の例に示すように、ロードバランサーと同じパブリックサブネット NAT にを配置できます。

- Amazon RDS インスタンスなどのバックエンドデータベースを使用している場合は、それらのインスタンスをプライベートサブネットに配置できます。Amazon RDS インスタンスの場合、異なるアベイラビリティゾーンで少なくとも2つの異なるサブネットを指定する必要があります。
- プライベートサブネット内のインスタンスへの直接アクセスが必要な場合、例えば、SSHを使用してインスタンスにログインする場合、インターネットからのリクエストをプロキシする踏み台ホストをパブリックサブネットに配置することができます。

独自のネットワークの拡張 AWS

独自のネットワークをクラウドに拡張し、 からインターネットに直接アクセスする場合は VPC、VPNゲートウェイを作成できます。詳細については、「[シナリオ 3: パブリックサブネットとプライベートサブネットとハードウェアVPNアクセスVPCを使用する](#)」を参照してください。

AWS OpsWorks スタックVPCスタック用の を作成する

このセクションでは、サンプル[AWS CloudFormation](#)テンプレートを使用して AWS OpsWorks スタックVPCスタックの を作成する方法を示します。テンプレートは [OpsWorksVPCtemplates.zip](#) [ファイル](#)でダウンロードできます。このトピックで説明しているVPCような を手動で作成する方法については、「[シナリオ 2: パブリックサブネットとプライベートサブネットVPCを使用する](#)」を参照してください。ルーティングテーブル、セキュリティグループなどを設定する方法の詳細については、サンプルテンプレートを参照してください。

Note

デフォルトでは、AWS OpsWorks スタックはCIDR範囲と などのアベイラビリティゾーンを連結してサブネット名を表示します10.0.0.1/24 - us-east-1b。名前をより読みやすくするには、キーを に設定Nameし、値をサブネット名に設定して、サブネットごとにタグを作成します。AWS OpsWorks スタックは、サブネット名をデフォルト名に追加します。例えば、次の例のプライベートサブネットには、名前が に設定されたタグがありPrivate、として OpsWorks 表示されます10.0.0.1/24 us-east - 1b - Private。

AWS CloudFormation コンソールを使用して、わずか数ステップでVPCテンプレートを起動できます。次の手順では、サンプルテンプレートを使用して、米国東部 (バージニア北部) リージョンVPCに を作成します。テンプレートを使用して他のリージョンVPCで を作成する方法については、「」の手順に従う [メモ](#)を参照してください。

VPC を作成するには

1. [AWS CloudFormation コンソール](#)を開き、[米国東部 (バージニア北部)] リージョンを選択し、[スタックの作成] を選びます。
2. [Select Template] ページで、[Upload a template] を選択します。 [OpsWorksVPCtemplates.zip](#) OpsWorksinVPC.template ファイルでダウンロードしたファイルを参照します。続行を選択します。

Select Template

Select the template that describes the stack that you want to create. A stack is a group of related resources that you manage as a single unit.

Design a template Use AWS CloudFormation Designer to create or modify an existing template. [Learn more.](#)

Design template

Choose a template A template is a JSON-formatted text file that describes your stack's resources and their properties. [Learn more.](#)

Select a sample template

Upload a template to Amazon S3

Browse... No file selected.

Specify an Amazon S3 template URL

[View/Edit template in Designer](#)

また、[AWS CloudFormation サンプルテンプレート](#)を開き、スタック AWS OpsWorks VPCテンプレートを見つけて、起動スタックを選択することで、このスタックを起動することもできます。

3. [Specify Parameters (パラメータを指定)] ページで、デフォルトの値をそのまま使用して、[Continue (続行)] を選択します。
4. タグの追加ページで、キーを `OpsWorks` に設定し、値をVPC名前に設定してタグを作成します。このタグを使用すると、スタックを作成するVPCときに AWS OpsWorks を識別しやすくなります。
5. [Continue (続行)] を選択し、[Close (閉じる)] を選択して、スタックを起動します。

注: 次の方法のいずれかを使用して、他のリージョンVPCで を作成できます。

- [「異なるリージョンでテンプレートを使用する」](#)に移動し、適切なリージョンを選択し、AWS OpsWorks スタックVPCテンプレートを見つけて、スタックを起動するを選択します。
- テンプレートファイルをシステムにコピーし、[AWS CloudFormation コンソール](#)で適切なリージョンを選択し、[スタックの作成] ウィザードの [テンプレートを Amazon S3 へのアップロード] オプションを使用して、システムからテンプレートをアップロードします。

サンプルテンプレートにはVPC、AWS OpsWorks スタックの作成に必要な、サブネット、ロードバランサーIDsを提供する出力が含まれています。コンソール AWS CloudFormation ウィンドウの下部にある出力タブを選択すると、それらを表示できます。

Stack: OpsWorks-in-VPC						
Description	Outputs	Resources	Events	Template	Parameters	Tags
Stack Outputs						
Output values may have been specified by the template author and will be available when stack creation is complete.						
Key	Value	Description				
VPC	vpc-d1ed9ebf	VPC				
PublicSubnets	subnet-66ec9f08	Public Subnet				
PrivateSubnets	subnet-6dec9f03	Private Subnet				
LoadBalancer	OpsWorks-ElasticL-K0FQ6TD5S14A	Load Balancer				

スタックの更新

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

スタックの作成後、設定をいつでも更新できます。[Stack] (スタック) ページで、[Stack Settings] (スタックの設定) をクリックしてから [Edit] (編集) をクリックすると、[Settings] (設定) ページが表示されます。必要な変更を加えて [Save] をクリックします。

各設定は、「[新しいスタックを作成する](#)」に記載されているものと同じです。詳細については、このトピックを参照してください。ただし、以下の点に注意してください。

- リージョンまたは VPC ID を変更することはできません。
- スタックが で実行されている場合VPC、設定には のサブネットを一覧表示するデフォルトのVPC サブネット設定が含まれます。スタックが で実行されていない場合VPC、設定にはデフォルトの アベイラビリティゾーンというラベルが付けられ、リージョンの アベイラビリティゾーンが一覧表示されます。
- デフォルトのオペレーティングシステムを変更することはできますが、Windows スタックに Linux オペレーティングシステムを指定したり、Linux スタックに Windows オペレーティングシステムを指定したりすることはできません。

- ホスト名テーマやデフォルトSSHキーなどのデフォルトのインスタンス設定を変更した場合、新しい値は、既存のインスタンスではなく、作成した新しいインスタンスにのみ適用されます。
- 名前を変更することで、コンソールに表示される名前が変更されます。AWS OpsWorks スタックがスタックを識別するために使用する基盤となる短縮名は変更されません。
- OpsWorks セキュリティグループの使用を「はい」から「いいえ」に変更する前に、各レイヤーには、レイヤーの組み込みセキュリティグループに加えて、少なくとも1つのセキュリティグループが必要です。詳細については、「[OpsWorks レイヤーの設定の編集](#)」を参照してください。

AWS OpsWorks その後、スタックはすべてのレイヤーから組み込みセキュリティグループを削除します。

- OpsWorks セキュリティグループの使用を「いいえ」から「はい」に変更した場合、AWS OpsWorks スタックは各レイヤーに適切な組み込みセキュリティグループを追加しますが、既存のセキュリティグループは削除しません。

スタックのクローン化

Important




この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

スタックの複数のコピーを作成すると便利な場合があります。たとえば、災害対策または予防措置として冗長性を追加したり、または新しいスタックの開始点として既存のスタックを使用したりした方がよいでしょう。最も簡単な方法は、ソーススタックのクローン化することです。AWS OpsWorks スタックダッシュボードで、クローンを作成するスタックの行の Actions 列でクローンを選択し、スタックのクローンページを開きます。

OpsWorks Dashboard

Add stack

Register instances

Stack name	Region	Layers	Instances	Apps	Actions
 [Redacted]	us-east-1	1	1	0	edit clone delete
 [Redacted]	us-west-2	2	1	0	edit clone delete
 MyLinuxDemoStack	us-west-2	1	1	1	edit clone delete

+ Stack

初期状態では、クローンスタックの設定は、スタック名に「copy」という語が付加される点を除き、ソーススタックの設定とまったく同じです。これらの設定については、「[新しいスタックを作成する](#)」を参照してください。次の2つの追加オプション設定もあります。

アクセス許可

[all permissions] がオンの場合 (デフォルト)、ソーススタックのアクセス権限がクローンスタックに適用されます。

アプリケーション

ソーススタックにデプロイされるアプリケーションが表示されます。表示されている各アプリケーションの対応するチェックボックスがオンの場合 (デフォルト)、そのアプリケーションがクローンスタックにデプロイされます。

Note

一つのリージョンのエンドポイントから別のリージョンのエンドポイントにスタックのクローンを作成できません。例えば、米国西部 (オレゴン) リージョン (us-west-2) から アジアパシフィックアジアパシフィック (ムンバイ) (ap-south-1) にスタックをクローン化できません。

設定を確定したら、スタックのクローンを作成します。AWS OpsWorks スタックは、ソーススタックのレイヤーと、オプションでそのアプリケーションとアクセス許可で構成される新しいスタックを作成します。レイヤーの設定は、ユーザーが行った変更が適用された元の設定と同じです。ただし、クローン化ではインスタンスは作成されません。クローンスタックの各レイヤーにインスタンスの適切なセットを追加し、インスタンスを起動する必要があります。スタックと同様に、クローンスタック

クで通常の管理タスク (レイヤーの追加、削除、または変更、アプリケーションの追加およびデプロイなど) を実行できます。

クローンされたスタックを機能させるには、インスタンスを起動します。AWS OpsWorks スタックは、レイヤーメンバーシップに従って各インスタンスをセットアップおよび設定します。また、新しいスタックの場合と同様にアプリケーションをデプロイします。

AWS OpsWorks スタックスタックコマンドを実行する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックには、スタックのインスタンスでさまざまなオペレーションを実行するために使用できる一連のスタックコマンドが用意されています。スタックコマンドを実行するには、[Stack] (スタック) ページで [Run Command] (コマンドの実行) をクリックします。該当するコマンドを選択し、任意のオプションを指定して、右下にあるコマンド名のラベルの付いたボタンをクリックします。

Note

AWS OpsWorks スタックは、アプリケーションのデプロイを管理するために使用する一連のデプロイコマンドもサポートしています。詳細については、「[アプリケーションのデプロイ](#)」を参照してください。

任意のスタックに対して以下のスタックコマンドを実行できます。

カスタムクックブックの更新

インスタンスのカスタムクックブックを、リポジトリに保存されている最新バージョンに更新します。このコマンドではレシピを実行しません。更新されたレシピを実行するには、Execute Recipes、Setup、Configure のいずれかのスタックコマンドを使用するか、[アプリケーション](#)

[ンを再デプロイして](#)、Deploy レシピを実行します。カスタムクックブックの詳細については、「[クックブックとレシピ](#)」を参照してください。

Execute Recipes

インスタンスに対して指定された一連のレシピを実行します。詳細については、「[レシピを手動で実行する](#)」を参照してください。

セットアップ

インスタンスの Setup レシピを実行します。

構成する

インスタンスの Configure レシピを実行します。

Note

[Setup] または [Configure] を使用してインスタンスでレシピを実行するには、インスタンスのレイヤーに対応するライフサイクルイベントに割り当てられている必要があります。詳細については、「[レシピの実行](#)」を参照してください。

Linux ベースのスタックでのみ、任意のスタックに対して以下のスタックコマンドを実行できます。

Install Dependencies

インスタンスのパッケージをインストールします。Chef 12 以降、このコマンドは使用できなくなります。

Update Dependencies

(Linux のみ。Chef 12 以降、このコマンドは使用できません。) 定期的なオペレーティングシステム更新とパッケージ更新をインストールします。詳細はインスタンスのオペレーティングシステムによって異なります。詳細については、「[セキュリティ更新の管理](#)」を参照してください。

[Upgrade Operating System] コマンドを使用して、インスタンスを新しい Amazon Linux バージョンにアップグレードします。

オペレーティングシステムのアップグレード

(Linux のみ) インスタンスの Amazon Linux オペレーティングシステムを最新バージョンにアップグレードします。詳細については、「[AWS OpsWorks スタックオペレーティングシステム](#)」を参照してください。

⚠ Important

[Upgrade Operating System] を実行した後で、Setup も実行することをお勧めします。これにより、サービスが正しく再開されます。

スタックコマンドには次のオプションがあります (一部は特定のコマンドにのみ表示されます)。

コメント

(オプション) 説明を付け加えたい場合のコメントを入力します。

Recipes to execute

(必須) この設定は、[Execute Recipes] コマンドを選択した場合にのみ表示されます。標準の `cookbook_name::recipe_name` 形式を使用して実行するレシピをカンマで区切って入力します。複数のレシピを指定すると、AWS OpsWorks スタックはリストされた順序でそれらを実行します。

Allow reboot

(オプション) この設定は Upgrade Operating System コマンドを選択した場合にのみ表示されます。デフォルト値は Yes で、アップグレードのインストール後にインスタンスを再起動するように AWS OpsWorks スタックに指示します。

カスタム Chef JSON

(オプション) Advanced を選択してこのオプションを表示します。これにより、スタック設定およびデプロイJSON属性に組み込むカスタム属性を指定できます。 [スタック設定およびデプロイメント属性](#)

インスタンス

(オプション) コマンドを実行するインスタンスを指定します。オンラインインスタンスはすべてデフォルトで選択されています。インスタンスのサブセットに対してコマンドを実行するには、適切な レイヤーまたはインスタンスを選択します。

i Note

[Deployment and Commands] ページに、実行しなかった execute_recipes 実行のリストが表示される場合があります。これは通常、ユーザーに対するアクセス許可の付与や削除な

ど、SSHアクセス許可の変更の結果です。このような変更を行うと、AWS OpsWorks スタックは `execute_recipes` を使用してインスタンスのアクセス許可を更新します。

カスタム の使用 JSON

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

いくつかの AWS OpsWorks スタックアクションでは、AWS OpsWorks スタックがインスタンスにインストールしJSON、レシピで使用できるカスタム を指定できます。

カスタム は、以下の状況JSONで指定できます。

- スタックを作成、更新、またはクローン化する

AWS OpsWorks スタックは、後続のすべての [ライフサイクルイベント](#) のカスタム をすべてのインスタンスJSONにインストールします。

- デプロイメントコマンドまたはスタックコマンドを実行する

AWS OpsWorks スタックは、そのイベントのインスタンスにのみカスタムJSONを渡します。

カスタム JSON は、有効な JSON オブジェクトで表され、その形式である必要があります。以下に例を示します。

```
{
  "att1": "value1",
  "att2": "value2"
  ...
}
```

AWS OpsWorks スタックは次のJSON場所にカスタム を保存します。

Linux インスタンスの場合:

- `/var/chef/runs/run-ID/attribs.json`
- `/var/chef/runs/run-ID/nodes/hostname.json`

Windows インスタンスの場合:

- `drive:\chef\runs\run-ID\attribs.json`
- `drive:\chef\runs\run-ID\nodes\hostname.json`

Note

Linux 用の Chef 11.10 以前のバージョンでは、カスタム JSON は Linux インスタンスの次のパスにあり、Windows インスタンスは使用できず、`attribs.json` ファイルもありません。ログは、同じフォルダまたはディレクトリに保存されます。Linux 用の JSON Chef 11.10 以前のバージョンのカスタムの詳細については、「[カスタムログと Chef ログによる属性の上書き JSON](https://docs.aws.amazon.com/opsworks/latest/userguide/troubleshoot-debug-log.html#troubleshoot-debug-log-instance)」を参照してください。 <https://docs.aws.amazon.com/opsworks/latest/userguide/troubleshoot-debug-log.html#troubleshoot-debug-log-instance>
`/var/lib/aws/opsworks/chef/hostname.json`

上記のパスでは、`run-ID` は AWS OpsWorks スタックがインスタンスで実行される各 Chef に割り当てられる一意の ID であり、`hostname` はインスタンスのホスト名です。

Chef レシピ JSON からカスタムにアクセスするには、標準の Chef node 構文を使用します。

たとえば、デプロイするアプリケーション用のシンプルな設定 (アプリケーションが最初に表示されるかどうか、アプリケーションの初期の前景色と背景色など) を定義するとします。次のように JSON、オブジェクトを使用してこれらのアプリ設定を定義するとします。

```
{
  "state": "visible",
  "colors": {
    "foreground": "light-blue",
    "background": "dark-gray"
  }
}
```

スタックのカスタムJSONを宣言するには：

1. スタックのページで、[Stack Settings]、[Edit] の順番に選択します。
2. カスタム Chef でJSON、JSON オブジェクトを入力し、保存を選択します。

Note

カスタム JSONは、デプロイ、レイヤー、スタックレベルで宣言できます。カスタムを個々のデプロイまたはレイヤーにのみ表示する場合はJSON、これを行うことができます。または、例えば、スタックレベルでJSON宣言されたカスタムをレイヤーレベルでJSON宣言されたカスタムで一時的に上書きできます。JSON 複数のレベルでカスタムを宣言する場合、デプロイレベルでJSON宣言されたカスタムは、レイヤーレベルとスタックレベルの両方でJSON宣言されたカスタムよりも優先されます。レイヤーレベルでJSON宣言されたカスタムは、スタックレベルでのみJSON宣言されたカスタムを上書きします。

AWS OpsWorks スタックコンソールを使用してデプロイJSONのカスタムを指定するには、アプリケーションのデプロイページでアドバンストを選択します。カスタム Chef ボックスにJSONカスタムJSONを入力し、保存を選択します。

AWS OpsWorks スタックコンソールを使用してJSONレイヤーのカスタムを指定するには、レイヤーページで、目的のレイヤーの設定を選択します。カスタム ボックスにJSONカスタム JSON を入力し、保存 を選択します。

詳細については、[OpsWorks レイヤーの設定の編集](#)および[アプリケーションのデプロイ](#)を参照してください。

デプロイまたはスタックコマンドを実行すると、レシピは、カスタムJSONオブジェクトの階層に直接マッピングされる標準の Chef node構文を使用して、これらのカスタム値を取得できます。たとえば、次のレシピコードは、前述のカスタムJSON値に関するメッセージを Chef ログに書き込みます。

```
Chef::Log.info("***** The app's initial state is '#{node['state']}' *****")
Chef::Log.info("***** The app's initial foreground color is '#{node['colors']
['foreground']}' *****")
Chef::Log.info("***** The app's initial background color is '#{node['colors']
['background']}' *****")
```

このアプローチは、レシピにデータを渡すのに役立ちます。AWS OpsWorks スタックはそのデータをインスタンスに追加し、レシピは標準の Chef node構文を使用してデータを取得できます。

Note

カスタムJSONは 120 KB に制限されています。さらに容量が必要な場合は、Amazon Simple Storage Service (Amazon S3) でデータの一部を保存することをお勧めします。その後、カスタムレシピは [AWSCLI](#) または [AWS SDK for Ruby](#) を使用して [AWS SDK for Ruby](#)、Amazon S3 バケットからインスタンスにデータをダウンロードできます。

スタックの削除

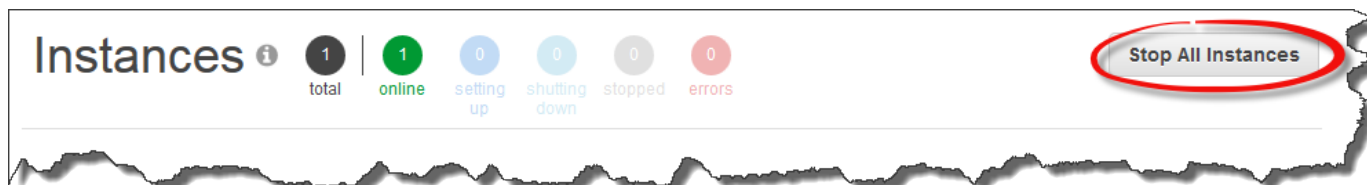
Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

スタックが不要になった場合には、それを削除することができます。空のスタックのみ削除できます。最初にスタック内のすべてのインスタンス、アプリ、レイヤーを削除する必要があります。

スタックを削除するには

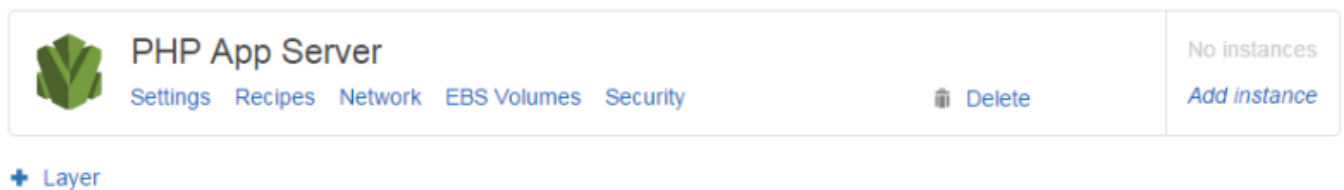
1. AWS OpsWorks スタックダッシュボードで、シャットダウンして削除するスタックを選択します。
2. ナビゲーションペインで、[インスタンス] を選択します。
3. [Instances] ページで [Stop all Instances] を選択します。



4. インスタンスが停止したら、レイヤーの各インスタンスで [Actions (アクション) 列の[delete] (削除) を選択します。確認を求められたら、[Yes, Delete] を選択します。



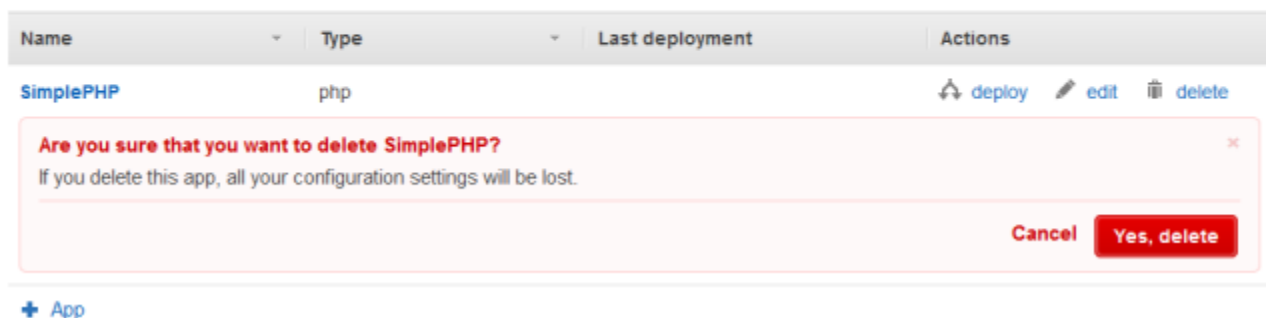
5. すべてのインスタンスが削除されたら、ナビゲーションペインで [Layers] (レイヤー) を選択します。
6. [Layers] (レイヤー) ページでスタックの各レイヤーに対して [delete] (削除) を選択します。確認プロンプトで、[Yes, Delete] を選択します。



7. すべてのレイヤーが削除されたら、ナビゲーションペインで [Apps] を選択します。
8. [Apps] (アプリケーション) ページでスタックの各アプリに対して [Actions] (アクション) 列の [delete] (削除) を選択します。確認プロンプトで、[Yes, Delete] を選択します。

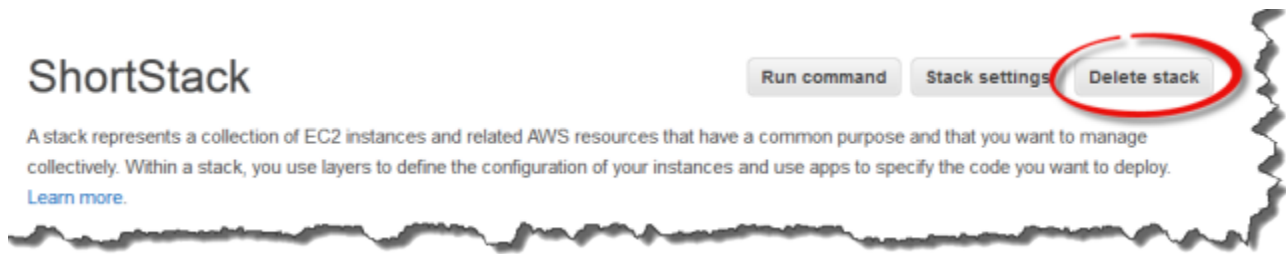
Apps

An app represents code stored in a repository that you want to install on application server instances. When you deploy the app, OpsWorks downloads the code from the repository to the specified server instances. [Learn more.](#)



9. すべてのアプリが削除されたら、ナビゲーションペインで [Stack] を選択します。

10. スタックページで、[Delete stack] を選択します。確認プロンプトで、[Yes, Delete] を選択します。



スタックで使用される他の AWS リソースの削除

AWS OpsWorks スタックで他の AWS リソースを使用して、スタックを作成および管理します。スタックを削除するときは、別のスタックで使用されておらず、スタック外のリソースでも使用されていない場合 AWS OpsWorks は、スタックで機能するリソースも削除することを検討してください。スタックで使用した外部 AWS リソースをクリーンアップする推奨理由は次のとおりです。

- 外部 AWS リソースでは、引き続きアカウントに AWS 料金が発生する可能性があります。
- Amazon S3 バケットなどのリソースには、個人識別情報、機密情報あるいは秘密情報が含まれる場合があります。

⚠ Important

これらのリソースが他のスタックで使用されている場合は削除しないでください。IAM ロールとセキュリティグループはグローバルであるため、他のリージョンのスタックはこれらの同じリソースを使用している可能性があります。

スタックが使用するその他の AWS リソースと、それらを削除する方法に関する情報へのリンクを次に示します。

サービスロールとインスタンスプロファイル

スタックを作成するときは、AWS OpsWorks IAMロールとスタックがユーザーに代わって許可されたリソースを作成するために使用するインスタンスプロファイルを指定します。AWS OpsWorks 既存のリソースを選択しない場合は、によってロールとインスタンスプロファイルが作成されます。が AWS OpsWorks 作成するロールとインスタンスプロファイルにはaws-opsworks-ec2-role、それぞれ aws-opsworks-service-roleと という名前が付けられま

す。アカウントの他のスタックが IAM ロールとインスタンスプロファイルを使用していない場合、これらのリソースを安全に削除できます。IAM ロールとインスタンスプロファイルを削除する方法については、「IAMユーザーガイド」の[「ロールまたはインスタンスプロファイルの削除」](#)を参照してください。

セキュリティグループ

AWS OpsWorks スタックでは、レイヤーレベルでユーザー定義のセキュリティグループを指定できます。Amazon EC2コンソールまたは を使用してセキュリティグループを作成します API。セキュリティグループがグローバルであるため、他のリージョンのスタックとレイヤーで同じセキュリティグループを使用できます。セキュリティグループは、他の AWS リソースで使用されていない場合は削除できます。セキュリティグループを削除する方法の詳細については、「Amazon EC2ユーザーガイド」の[「セキュリティグループの削除」](#)を参照してください。

Amazon EBSボリューム

AWS OpsWorks スタックでは、EBSボリュームをレイヤーレベルで追加し、レイヤー内のインスタンスにアタッチします。Amazon EC2サービスコンソールまたは を使用してEBSボリュームを作成しAPI、レイヤーレベルで AWS OpsWorks スタックインスタンスにアタッチします。EBSボリュームは[アベイラビリティーゾーン](#)に固有です。特定のリージョンとアベイラビリティーゾーンのスタックでEBSボリュームを使用しなくなった場合は、ボリュームを削除できます。Amazon EBSボリュームを削除する方法の詳細については、「[Amazon ユーザーガイド](#)」の[「Amazon EBSボリュームの削除」](#)を参照してください。 EC2

Amazon Simple Storage Service (Amazon S3) の バケット

AWS OpsWorks スタックでは、以下の目的で Amazon S3 バケットを使用できます。Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、「[S3 バケットを空にする方法](#)」または「[S3 バケットを削除する方法](#)」を参照してください。

- アプリケーションコードの保管
- クックブックとレシピの保管
- CloudTrail スタックで CloudTrail ログ記録を有効にしている場合の AWS OpsWorks ログ
- スタックで有効にした場合の Amazon CloudWatch Logs AWS OpsWorks ストリーム

Elastic IP アドレス

AWS OpsWorks スタックに [Elastic IP アドレスを登録](#)し、Elastic IP アドレスが不要になった場合は、[Elastic IP アドレスを解放](#)できます。

Elastic Load Balancing ロードバランサー

スタック内のレイヤーで使用していた Elastic Load Balancing クラシックロードバランサーが不要になった場合は、削除することができます。詳細については、『[Classic Load Balancer のユーザーガイド](#)』の「ロードバランサーの削除」を参照してください。

Amazon Relational Database Service (Amazon RDS) インスタンス

Amazon RDS データベース (DB) インスタンスを AWS OpsWorks スタックに[登録](#)し、不要になった場合は、DB インスタンスを削除できます。DB インスタンスを削除する方法の詳細については、「[Amazon RDSユーザーガイド](#)」の「[DB インスタンスの削除](#)」を参照してください。

Amazon Elastic Container Service (Amazon ECS) クラスタ

スタックに ECS クラスタレイヤーが含まれていて、レイヤーに登録された ECS クラスタを使用しなくなった場合は、ECS クラスタを削除できます。ECS クラスタを削除する方法の詳細については、「[Amazon ECSデベロッパーガイド](#)」の「[クラスタの削除](#)」を参照してください。

レイヤー

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

各スタックには、ロードバランサーや一連のアプリケーションサーバーなどのスタックコンポーネントを表すレイヤーが少なくとも 1 つ含まれます。

AWS OpsWorks スタックレイヤーを使用する場合は、次の点に注意してください。

- スタックの各レイヤーにはインスタンスが少なくとも 1 つ必要です。オプションで複数のインスタンスを指定することもできます。
- スタックの各インスタンスは、[登録されたインスタンス](#)を除いて、少なくとも 1 つのレイヤーのメンバーとなっている必要があります。

SSH キーやホスト名などの基本的な設定を除き、インスタンスを直接設定することはできません。適切なレイヤーを作成および設定し、そのレイヤーにインスタンスを追加する必要があります。

Amazon EC2インスタンスは、オプションで複数のレイヤーのメンバーにすることができます。この場合、AWS OpsWorks スタックはレシピを実行して、インスタンスの各レイヤーのパッケージのインストールと設定、アプリケーションのデプロイなどを行います。

複数のレイヤーにインスタンスを割り当てることには次のような利点があります。

- 単一のインスタンスでデータベースサーバーやロードバランサーをホストして費用を削減できる。
- 管理用アプリケーションサーバーの 1 つを使用できる。

カスタム管理用レイヤーを作成し、アプリケーションサーバーインスタンスの 1 つを追加します。管理用レイヤーのレシピにより、追加したアプリケーションサーバーインスタンスが管理タスクを実行するように設定され、必要に応じて追加のソフトウェアがインストールされます。その他のアプリケーションサーバーインスタンスは単なるアプリケーションサーバーです。

このセクションでは、レイヤーを操作する方法について説明します。

トピック

- [OpsWorks レイヤーの基本](#)
- [Elastic ロードバランシングレイヤー](#)
- [Amazon RDS Service Layer](#)
- [ECS クラスターレイヤー](#)
- [カスタム AWS OpsWorks スタックレイヤー](#)
- [レイヤーごとのオペレーティングシステムパッケージのインストール](#)

OpsWorks レイヤーの基本

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、すべての AWS OpsWorks スタックレイヤーに共通のオペレーションを実行する方法について説明します。

トピック

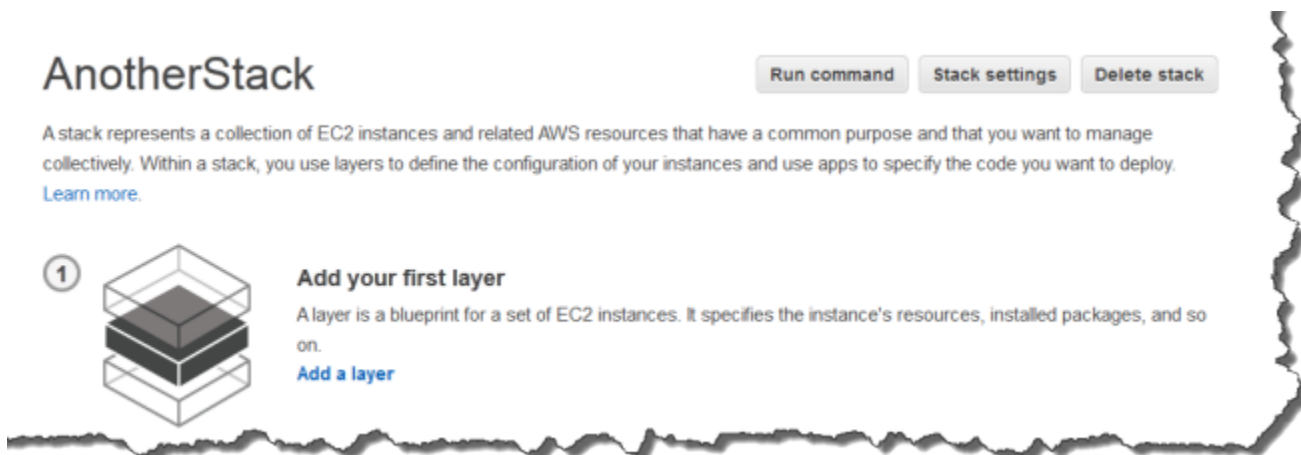
- [OpsWorks レイヤーの作成](#)
- [OpsWorks レイヤーの設定の編集](#)
- [自動ヒーリングを使用した、失敗したインスタンスの置き換え](#)
- [OpsWorks レイヤーの削除](#)

OpsWorks レイヤーの作成

⚠ Important


この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

新しいスタックを作成すると、次のページが表示されます。




最初の OpsWorks レイヤーを追加するには

1. [Add a Layer] (レイヤーの追加) をクリックします。
2. [Add Layer] (レイヤーの追加) ページで Layer の設定オプションを確認し、適切なレイヤー選択します。
3. レイヤーを適切に設定し、[Add Layer] (レイヤーの追加) をクリックしてスタックに追加します。次のセクションでは、さまざまなレイヤーの設定方法について説明します。

 Note

[Add Layer] (レイヤーの追加) ページには、レイヤーごとによく使用される設定のみが表示されます。[レイヤーを編集](#)することで、その他の設定を指定できます。

4. インスタンスをレイヤーに追加して起動します。

 Note

インスタンスが複数のレイヤーのメンバーである場合は、インスタンスを起動する前にすべてのレイヤーに追加する必要があります。オンラインインスタンスをレイヤーに追加することはできません。

レイヤーを追加するには、[Layers] (レイヤー) ページを開いて [+ Layer] をクリックし、[Add Layer] (レイヤーの追加) ページを開きます。

インスタンスを起動すると、AWS OpsWorks スタックはインスタンスの各レイヤーの Setup レシピと Deploy レシピを自動的に実行して、適切なパッケージをインストールおよび設定し、適切なアプリケーションをデプロイします。[レイヤーのセットアップと設定プロセスは、適切なライフサイクルイベントにカスタムレシピを割り当てるなど、さまざまな方法でカスタマイズ](#)できます。AWS OpsWorks スタックは、各イベントの標準レシピの後にカスタムレシピを実行します。詳細については、「[クックブックとレシピ](#)」を参照してください。

以下のレイヤー固有のセクションでは、さまざまな AWS OpsWorks スタックレイヤーのステップ 2 と 3 を処理する方法について説明します。インスタンスの追加方法の詳細については、「[レイヤーへのインスタンスの追加](#)」を参照してください。

OpsWorks レイヤーの設定の編集

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

レイヤーを作成した後、一部のプロパティ (AWSリージョンなど) はイミュータブルですが、ほとんどのレイヤー設定はいつでも変更できます。レイヤーを編集すると、[Add Layer] (レイヤーの追加) ページにない設定にもアクセスできます。この設定は、新しい設定を保存するとすぐに有効になります。

OpsWorks レイヤーを編集するには

1. ナビゲーションペインで、[Layers] (レイヤー) をクリックします。
2. [Layers] (レイヤー) ページで、レイヤー名を選択して詳細ページを開きます。詳細ページには現在の設定が表示されます。

ℹ Note

レイヤーの名前の下にあるいずれかの名前を選択すると、詳細ページの関連タブに直接移動します。

3. 編集をクリックして、全般設定、レシピ、ネットワーク、EBSボリューム、セキュリティのタブを選択します。

次のセクションでは、すべてのレイヤーで使用できるさまざまなタブの設定について説明します。一部のレイヤーには、レイヤー固有の追加設定があり、ページの上部に表示されます。さらに、注記のとおり、一部の設定は Linux ベースのスタックでのみ使用できます。

トピック

- [全般設定](#)
- [recipe](#)

- [ネットワーク](#)
- [EBS ボリューム](#)
- [セキュリティ](#)
- [CloudWatch ログ](#)
- [\[タグ\]](#)

全般設定

すべてのレイヤーは次のように設定されます。

Auto healing enabled

レイヤーのインスタンスに対して[自動ヒール](#)が有効かどうかを示します。デフォルトの設定は、[Yes] です。

カスタム JSON

このレイヤーのすべてのインスタンスの Chef レシピに渡される JSON 形式のデータ。たとえば、これを使用してデータを独自のレシピに渡すことができます。詳細については、「[カスタムの使用 JSON](#)」を参照してください。

Note

カスタム JSON は、デプロイ、レイヤー、スタックレベルで宣言できます。これは、スタック全体または個々のデプロイにのみカスタム JSON を表示する場合に実行できます。または、例えば、レイヤーレベルで JSON 宣言されたカスタム をデプロイレベルで JSON 宣言されたカスタム で一時的に上書きできます。JSON 複数のレベルでカスタムを宣言する場合、デプロイレベルで JSON 宣言されたカスタムは、レイヤーレベルとスタックレベルの両方で JSON 宣言されたカスタムよりも優先されます。レイヤーレベルで JSON 宣言されたカスタムは、スタックレベルでのみ JSON 宣言されたカスタムを上書きします。AWS OpsWorks スタックコンソールを使用してデプロイ JSON のカスタムを指定するには、アプリケーションのデプロイページでアドバンストを選択します。カスタム Chef ボックスに JSON カスタム JSON を入力し、保存を選択します。

AWS OpsWorks スタックコンソールを使用して JSON スタックのカスタムを指定するには、スタック設定ページでカスタム JSON ボックスにカスタムを入力し、保存を選択します。

詳細については、[カスタムの使用 JSON](#) および [アプリケーションのデプロイ](#) を参照してください。

Instance shutdown timeout

Amazon EC2インスタンスを停止または終了するまでに、[Shutdown ライフサイクルイベント](#)をトリガーした後に AWS OpsWorks スタックが待機する時間 (秒単位) を指定します。デフォルトの設定は 120 秒です。設定の目的は、インスタンスを終了する前に、インスタンスの Shutdown レシピに対して、タスクを完了するための十分な時間を与えることです。カスタム Shutdown レシピでさらに時間が必要な場合は、それに応じて設定を変更します。インスタンスのシャットダウンの詳細については、「[インスタンスの停止](#)」を参照してください。

このタブのその他の設定は、レイヤーの種類によって異なり、レイヤーの [Add Layer] (レイヤーの追加) ページの設定と同じです。

recipe

[Recipes] タブには次の設定が含まれます。

[Custom Chef recipes]

レイヤーのライフサイクルイベントにカスタム Chef レシピを割り当てることができます。詳細については、「[レシピの実行](#)」を参照してください。

ネットワーク

[Network] タブには次の設定が含まれます。

Elastic Load Balancing

あらゆる Layer に Elastic Load Balancing のロードバランサーをアタッチできます。AWS OpsWorks その後、スタックはレイヤーのオンラインインスタンスをロードバランサーに自動的に登録し、オフラインになると登録を解除します。ロードバランサーの Connection Draining 機能を有効にしている場合は、AWS OpsWorks スタックがサポートするかどうかを指定できます。詳細については、「[Elastic ロードバランシングレイヤー](#)」を参照してください。

[Automatically Assign IP Addresses]

AWS OpsWorks スタックがレイヤーのインスタンスにパブリック IP アドレスまたは Elastic IP アドレスを自動的に割り当てるかどうかを制御できます。このオプションを有効にすると、次のようになります。

- instance store-backed インスタンスの場合、AWS OpsWorks スタックはインスタンスが起動されるたびにアドレスを自動的に割り当てます。

- Amazon EBS-backed インスタンスの場合、インスタンスが初めて起動されると、AWS OpsWorks スタックによってアドレスが自動的に割り当てられます。
- インスタンスが複数のレイヤーに属している場合、少なくとも1つのレイヤーに対して自動割り当てを有効にしている場合、AWS OpsWorks スタックは自動的にアドレスを割り当てます。

Note

パブリック IP アドレスの自動割り当てを有効にすると、新しいインスタンスにのみ適用されます。AWS OpsWorks スタックは既存のインスタンスのパブリック IP アドレスを更新できません。

スタックがで実行されている場合VPC、パブリック IP アドレスと Elastic IP アドレスの設定は別々になります。次の表はパブリック IP アドレスと Elastic IP アドレスが対話する方法を説明しています。

Public IP addresses

		Yes	No
Elastic IP addresses	Yes	Instances receive an Elastic IP address when they are started for the first time, or a public IP address if an Elastic IP cannot be assigned.	Instances receive an Elastic IP address when they are started for the first time.
	No	Instances receive a public IP address each time they are started.	Instances receive only a private IP address, which is not accessible from outside the VPC.

Note

インスタンスには、AWS OpsWorks スタックサービス、Linux パッケージリポジトリ、クックブックリポジトリと通信する方法が必要です。パブリック IP アドレスまたは Elastic IP アドレスを指定しない場合、レイヤーのインスタンスが外部サイトと通信NATできるようにするなどのコンポーネントを に含めるVPC必要があります。詳細については、「[でのスタックの実行 VPC](#)」を参照してください。

スタックがで実行されていない場合はVPC、Elastic IP アドレスが唯一の設定です。

- Yes: インスタンスの初回起動時に、インスタンスに Elastic IP アドレスが割り当てられます。Elastic IP アドレスを割り当てることができない場合は、パブリック IP アドレスが割り当てられます。

- No: インスタンスを起動するたびに、パブリック IP アドレスが割り当てられます。

EBS ボリューム

EBS ボリュームタブには、次の設定が含まれます。

EBS 最適化インスタンス

レイヤーのインスタンスを Amazon Elastic Block Store (Amazon EBS) 用に最適化するかどうか。詳細については、[「Amazon EBS最適化インスタンス」](#)を参照してください。

追加EBSボリューム

(Linux のみ) レイヤーのインスタンスに [Amazon EBSボリューム](#) を追加または削除できます。インスタンスを起動すると、AWS OpsWorks スタックによってボリュームが自動的に作成され、インスタンスにアタッチされます。リソースページを使用して、スタックのEBSボリュームを管理できます。詳細については、「[リソース管理](#)」を参照してください。

- マウントポイント – (必須) EBSボリュームをマウントするマウントポイントまたはディレクトリを指定します。
- # Disks – (オプション) RAID配列を指定した場合、配列内のディスク数。

各RAIDレベルにはデフォルトのディスク数がありますが、リストからさらに多くのディスクを選択できます。

- [Size total (GiB)] (合計サイズ (GiB)) - (必須) ボリュームのサイズ (GiB 単位) を指定します。

RAID 配列の場合、この設定は各ディスクのサイズではなく、配列の合計サイズを指定します。

次の表は、各ボリュームタイプで許容される最小および最大ボリュームサイズをまとめたものです。

ボリュームタイプ	最小サイズ (GiB)	最大サイズ (GiB)
マグネティック	1	1024
プロビジョニング済み IOPS (SSD)	4	16384
汎用 (SSD)	1	16384

ボリュームタイプ	最小サイズ (GiB)	最大サイズ (GiB)
スループット最適化 (HDD)	500	16384
コールド HDD	500	16384

- ボリュームタイプ – (オプション) マグネティック、汎用、スループット最適化 SSD、HDD、コールド HDD、または PIOPS ボリュームを作成するかどうかを指定します。

デフォルト値は Magnetic です。

- Encrypted – (オプション) EBS ボリュームのコンテンツを暗号化するかどうかを指定します。
- IOPS ディスクあたり – (プロビジョンド SSD ボリューム IOPSSSD と汎用 ボリュームで必須) プロビジョンド ボリューム IOPSSSD または 汎用 SSD ボリュームを指定する場合は、IOPS ディスクごとにも指定する必要があります。

プロビジョニングされた IOPS ボリュームの場合、ボリュームの作成時に IOPS レートを指定できます。IOPS プロビジョニングされたボリュームとリクエストされたボリュームサイズの比率は最大 30 です (つまり、3000 のボリュームは少なくとも 100 GB IOPS である必要があります)。汎用 (SSD) ボリュームタイプには、ボリュームサイズ \times 3 IOPS のベースラインがあり、最大 10,000 IOPS 個で、30 分間最大 3,000 IOPS 個までバーストできます。

ボリュームをレイヤーに追加するか、レイヤーから削除する場合は、次の点に注意してください。

- ボリュームを追加すると、すべての新しいインスタンスに新しいボリュームが追加されますが、AWS OpsWorks スタックは既存のインスタンスを更新しません。
- ボリュームを削除する場合、新しいインスタンスにのみ適用されます。既存のインスタンスのボリュームは削除されません。

マウントポイントの指定

任意のマウントポイントを指定することができます。ただし、一部のマウントポイントは スタック または Amazon AWS OpsWorks が使用できるように予約 EC2 されており、Amazon EBS ボリュームには使用しないでください。/home や /etc などの一般的な Linux システムフォルダは使用しないでください。

以下のマウントポイントは、AWS OpsWorks スタックで使用するために予約されています。

- /srv/www

- /var/log/apache2 (Ubuntu)
- /var/log/httpd (Amazon Linux)
- /var/log/mysql
- /var/www

autofs (自動マウントデーモン) は、インスタンスの起動または再起動時に /media/ephemeral0 のようなエフェメラルデバイスマウントポイントを使用してマウントをバインドします。このオペレーションは、Amazon EBSボリュームがマウントされる前に実行されます。Amazon EBSボリュームのマウントポイントが autofs と競合しないようにするには、エフェメラルデバイスマウントポイントを指定しないでください。可能なエフェメラルデバイスマウントポイントは、特定のインスタンスタイプ、およびインスタンスストア-backed か Amazon-backed EBSかによって異なります。autofs との競合を避けるため、次を実行します。

- 特定のインスタンスタイプのエフェメラルデバイスマウントポイントと、使用するバックिंगストアを確認します。
- Instance Store-Backed インスタンスで動作するマウントポイントは、AmazonEBS-Backed インスタンスに切り替えると autofs と競合する可能性があり、その逆も同様であることに注意してください。

Note

インスタンスストアブロックデバイスマッピングを変更する場合は、カスタムを作成できますAMI。詳細については、「[Amazon EC2 Instance Store](#)」を参照してください。AWS OpsWorks スタックAMIのカスタムを作成する方法の詳細については、「」を参照してください。[カスタムAMIの使用](#)。

次の例は、ボリュームのマウントポイントが autofs と競合しないよう、どのようにカスタムレシピを使用できるかを示しています。特定のユースケースに応じて適応してください。

マウントポイントの競合を回避するには

1. Amazon EBSボリュームを目的のレイヤーに割り当てますが、autofs と競合しない /mnt/workspace などのマウントポイントを使用します。
2. 次のカスタムレシピを実装します。これにより、Amazon EBSボリュームにアプリケーションディレクトリが作成され、からリンクされます/srv/www/。カスタムレシピの実装方法の詳細

については、「[クックブックとレシピ](#)」と「[AWS OpsWorks スタックのカスタマイズ](#)」を参照してください。

```
mount_point = node['ebs']['raids']['/dev/md0']['mount_point'] rescue nil

if mount_point
  node[:deploy].each do |application, deploy|
    directory "#{mount_point}/#{application}" do
      owner deploy[:user]
      group deploy[:group]
      mode 0770
      recursive true
    end

    link "/srv/www/#{application}" do
      to "#{mount_point}/#{application}"
    end
  end
end
```

3. 「depends 'deploy'」行をカスタムクックブックの metadata.rb ファイルに追加します。
4. [このレシピをレイヤーの Setup イベントに割り当てます。](#)

セキュリティ

[Security] タブには次の設定が含まれます。

セキュリティグループ

レイヤーには、セキュリティグループを少なくとも1つ関連付ける必要があります。スタックを[作成](#)または[更新](#)するときにセキュリティグループを関連付ける方法を指定します。AWS OpsWorks スタックには、標準の組み込みセキュリティグループのセットが用意されています。

- デフォルトのオプションでは、AWS OpsWorks スタックによって適切な組み込みセキュリティグループが各レイヤーに自動的に関連付けられます。
- 組み込みセキュリティグループが自動的に関連付けされないように設定し、レイヤーの作成時にカスタムセキュリティグループを各レイヤーに関連付けることも可能です。

セキュリティグループの詳細については、「[セキュリティグループの使用](#)」を参照してください。

レイヤーの作成後、Security Groups を使い、[Custom security groups] リストからセキュリティグループを選択してレイヤーに追加することができます。レイヤーにセキュリティグループを追加すると、AWS OpsWorks スタックによってすべての新しいインスタンスに追加されます。(再起動されたインスタンスも新しいインスタンスとして起動されるため、新しいセキュリティグループも作成されます)。AWS OpsWorks スタックは、オンラインインスタンスにセキュリティグループを追加しません。

次のように、[x] をクリックして既存のセキュリティグループを削除することができます。

- AWS OpsWorks スタックに組み込みセキュリティグループを自動的に関連付けることを選択した場合、x をクリックして以前に追加したカスタムセキュリティグループを削除できますが、組み込みグループを削除することはできません。
- 自動的に組み込みセキュリティグループが関連付けられないように設定した場合は、元のセキュリティグループも含め、任意の既存のセキュリティグループを削除することができます。ただし、レイヤーにはセキュリティグループを少なくとも 1 つ関連付けておく必要があります。

レイヤーからセキュリティグループを削除した後、AWS OpsWorks スタックはそれを新しいインスタンスまたは再起動されたインスタンスに追加しません。AWS OpsWorks スタックはオンラインインスタンスからセキュリティグループを削除しません。

Note

スタックがで実行されている場合はVPC、Amazon EC2コンソール、またはを使用してAPI、オンラインインスタンスのセキュリティグループを追加または削除できますCLI。ただし、このセキュリティグループはAWS OpsWorks スタックコンソールに表示されません。セキュリティグループを削除する場合は、Amazon も使用する必要がありますEC2。詳細については、「[セキュリティグループ](#)」を参照してください。

次の点に注意してください:

- より制限の厳しいセキュリティグループを追加しても、組み込みセキュリティグループのポートアクセスの設定は制限されません。複数のセキュリティグループがある場合、Amazon は最も許容度の高い設定EC2を使用します。
- 組み込みセキュリティグループの設定は変更しないでください。スタックを作成すると、AWS OpsWorks スタックによって組み込みのセキュリティグループの設定が上書きされるため、次回スタックを作成するときに行った変更は失われます。

より制限が厳しい設定のセキュリティグループをレイヤーに使用する必要がある場合は、次のステップに従います。

1. 適切な設定のカスタムセキュリティグループを作成し、適切なレイヤーに追加します。

カスタム設定が必要なレイヤーが 1 つだけの場合も、スタックの各レイヤーに組み込みセキュリティグループの他にセキュリティグループを少なくとも 1 つ追加する必要があります。

2. [スタック設定を編集](#)し、セキュリティグループの使用 OpsWorks設定を「いいえ」に切り替えます。

AWS OpsWorks スタックは、すべてのレイヤーから組み込みセキュリティグループを自動的に削除します。

セキュリティグループの詳細については、[「Amazon EC2 セキュリティグループ」](#)を参照してください。

EC2 インスタンスプロファイル

レイヤーのインスタンスのEC2プロファイルを変更できます。詳細については、「[EC2 インスタンスで実行されるアプリケーションのアクセス許可の指定](#)」を参照してください。

CloudWatch ログ

CloudWatch ログタブでは、Amazon CloudWatch Logs を有効または無効にできます。CloudWatch ログ統合は、Chef 11.10 および Chef 12 Linux ベースのスタックで機能します。Logs 統合を有効にし、CloudWatch Logs コンソールで管理する CloudWatch ログを指定する方法の詳細については、「」を参照してください[AWS OpsWorks スタックでの Amazon CloudWatch Logs の使用](#)。

[タグ]

[Tags] タブでは、レイヤーにコスト配分タグを適用することができます。タグを追加したら、AWS Billing and Cost Management コンソールでアクティブ化できます。タグを作成する場合、タグ付けされた構造内すべてのリソースにタグを適用することになります。たとえば、レイヤーにタグを適用すると、レイヤー内のすべてのインスタンス、Amazon EBSボリューム、または Elastic Load Balancing ロードバランサーにタグが適用されます。タグをアクティブ化し、それらを使用して AWS OpsWorks スタックリソースのコストを追跡および管理する方法の詳細については、「Billing and Cost Management ユーザーガイド」の「[コスト配分タグの使用](#)」および「[ユーザー定義のコスト配分タグのアクティブ化](#)」を参照してください。AWS OpsWorks スタックへのタグ付けの詳細については、[タグ](#)を参照してください。

自動ヒーリングを使用した、失敗したインスタンスの置き換え

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

すべてのインスタンスには、サービスと定期的に通信する AWS OpsWorks スタックエージェントがあります。AWS OpsWorks スタックは、その通信を使用してインスタンスの状態をモニタリングします。エージェントが約 5 分以上サービスと通信しない場合、AWS OpsWorks スタックはインスタンスが失敗したと見なします。

自動ヒーリングはレイヤーレベルで設定されます。次のスクリーンショットにあるように、レイヤー設定を編集することで自動ヒーリングの設定を変更できます。

Layer windowscompute

The screenshot shows the 'Settings' page for a layer named 'windowscompute'. The 'General Settings' tab is selected. The settings are as follows:

Setting	Value
Name	windowscompute
Short name	compute
Instance shutdown timeout	120
Auto healing enabled	Yes

Note

インスタンスは複数のレイヤーのメンバーである場合があります。これらのレイヤーのいずれかで自動ヒーリングが無効になっている場合、AWS OpsWorks スタックは失敗してもインスタンスをヒーリングしません。

レイヤーで自動ヒーリングが有効になっている場合、デフォルト設定では、AWS OpsWorks スタックはレイヤーの障害が発生したインスタンスを次のように自動的に置き換えます。

Instance store-backed インスタンス

1. Amazon EC2インスタンスを停止し、シャットダウンしたことを確認します。
2. ルートボリューム上のデータは削除されます。
3. 同じホスト名、設定、レイヤーメンバーシップを持つ新しい Amazon EC2インスタンスを作成します。
4. 古いインスタンスが最初に起動された後にアタッチされたEBSボリュームを含め、Amazon ボリュームを再アタッチします。
5. 新しいパブリックアドレスおよびプライベート IP アドレスが割り当てられます。
6. 古いインスタンスが Elastic IP アドレスに関連付けられている場合は、同じ IP アドレスが新しいインスタンスに関連付けられます。

Amazon EBS-backed インスタンス

1. Amazon EC2インスタンスを停止し、停止したことを確認します。
2. EC2 インスタンスを起動します。

自動修復されたインスタンスがオンラインに戻ると、AWS OpsWorks スタックはスタックのすべてのインスタンスで [Configure ライフサイクルイベント](#) をトリガーします。関連する [スタック設定およびデプロイ属性](#) には、インスタンスのパブリック IP アドレスおよびプライベート IP アドレスが含まれます。カスタム Configure レシピは、ノードオブジェクトから新しい IP アドレスを取得できます。

レイヤーのインスタンスに [Amazon EBSボリュームを指定すると](#)、AWS OpsWorks スタックによって新しいボリュームが作成され、インスタンスの起動時に各インスタンスにアタッチされます。後でインスタンスからボリュームをデタッチする場合は、[\[Resources\]](#) ページを使用します。

AWS OpsWorks スタックがレイヤーのインスタンスの 1 つを自動ヒーリングすると、ボリュームは次の方法で処理されます。

- インスタンスが失敗したときにボリュームがインスタンスにアタッチされた場合、ボリュームとそのデータは保存され、AWS OpsWorks スタックによって新しいインスタンスにアタッチされます。
- ボリュームが添付されていないインスタンスが失敗した場合は、レイヤーで指定されている設定を使用して AWS OpsWorks スタックが新しい空のボリュームを作成し、新しいインスタンスにそのボリュームを添付します。

自動ヒーリングはすべてのレイヤーで自動的に有効になっていますが、[レイヤーの全般設定を編集して無効にすることができます](#)。

Important

自動ヒーリングを可能にしている場合は、必ず次の手順に従います。

- インスタンスを停止APIするには、AWS OpsWorks スタックコンソールCLI、または [AWS CLI](#) のみを使用します。

Amazon EC2コンソールを使用するなど、他の方法でインスタンスを停止すると、AWS OpsWorks スタックはインスタンスを失敗として扱い、自動修復します。

- Amazon EBSボリュームを使用して、インスタンスが自動修復された場合に失いたくないデータを保存します。

自動ヒーリングは古い Amazon EC2インスタンスを停止し、Amazon EBSボリュームに保存されていないデータは破棄されます。Amazon EBSボリュームは新しいインスタンスに再アタッチされ、保存されたデータは保持されます。

OpsWorks レイヤーの削除

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックレイヤーが不要になった場合は、スタックから削除できます。

OpsWorks レイヤーを削除するには

1. ナビゲーションペインの [Instances] をクリックします。
2. [Instances] (インスタンス) ページの削除したいレイヤーの名前にある、各インスタンスの [Actions] (アクション) 列内の [stop] (停止) をクリックします。

PHP App Server

Host Name	Status	Size	Type	AZ	Public IP	Actions
php-app1	online	c1.medium	24/7	us-east-1a	54.242.127.207	stop


Are you sure you want to stop php-app1?

All data not stored on EBS volumes will be lost.

Cancel Stop

+ Instance

3. 各インスタンスの停止後、[delete] (削除) をクリックしてレイヤーから削除します。
4. ナビゲーションペインで、[Layers] (レイヤー) をクリックします。
5. [Layers] (レイヤー) ページで、[Delete] (削除) を選択します。

 PHP App Server Settings Recipes Network EBS Volumes Security Delete	No instances Add instance
--	--

+ Layer

Elastic ロードバランシングレイヤー

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Elastic Load Balancing AWS OpsWorks は、スタックレイヤーとは動作が多少異なります。レイヤーを作成してインスタンスを追加する代わりに、Elastic Load Balancing コンソールまたは API を使用してロードバランサーを作成し、既存のレイヤーにアタッチします。レイヤーのインスタンスにトラフィックを分散することに加えて、Elastic Load Balancing は以下を行います。

- 異常な Amazon EC2 インスタンスを検出し、異常なインスタンスが復元されるまで、残りの正常なインスタンスにトラフィックを再ルーティングします。
- 受信トラフィックに応じて、自動的にそのリクエスト処理能力を拡張します。
- [Connection Draining](#) を有効にすると、異常があるまたは登録解除されるインスタンスへの新しいリクエストの送信がロードバランサーで停止されるものの、指定された最大タイムアウト値まで接続が維持されるため、インスタンスはすべての処理中のリクエストを完了できます。

ロードバランサーをレイヤーにアタッチすると、AWS OpsWorks Stacks は以下を実行します。

- 現在登録されているインスタンスの登録を解除します。
- レイヤーのインスタンスがオンラインになるとこれを自動的に登録し、オフラインになるとインスタンスの登録を解除します。これには、負荷ベースのインスタンスも、時間ベースのインスタンスも含まれます。
- アベイラビリティゾーンの登録されたインスタンスへのリクエストのルーティングを自動的に開始します。

ロードバランサーの [Connection Draining](#) 機能を有効にしている場合は、AWS OpsWorks スタックがサポートするかどうかを指定できます。Connection Draining サポート (デフォルト設定) を有効にすると、インスタンスがシャットダウンされた後、AWS OpsWorks Stacks は以下を実行します。

- ロードバランサーからインスタンスを登録解除します。

ロードバランサーは新しいリクエストの送信を停止し、Connection Drainingを開始します。

- ロードバランサーが Connection Draining を完了するまで、[Shutdown ライフサイクルイベント](#)のトリガーを遅らせます。

Connection Draining サポートを有効にしない場合、インスタンスがまだロードバランサーに接続されていても、インスタンスがシャットダウンするとすぐに AWS OpsWorks、スタックによって Shutdown イベントがトリガーされます。

スタックで Elastic Load Balancing を使用するには、まず Elastic Load Balancing コンソール、または を使用して、同じリージョンに 1 つ以上のロードバランサーを作成する必要がありますAPI。CLI以下に注意する必要があります。

- レイヤーにアタッチできるロードバランサーの数は 1 つのみです。
- 各ロードバランサーで処理できるレイヤーは 1 つのみです。
- AWS OpsWorks スタックは Application Load Balancer をサポートしていません。AWS OpsWorks スタックは Classic Load Balancer でのみ使用できます。

つまり、負荷分散する各スタックのレイヤーごとに個別の Elastic Load Balancing ロードバランサーを作成し、その目的のためだけに使用する必要があります。推奨される方法は、MyStack1-RailsLayer- など、AWS OpsWorks スタックで使用する予定の各 Elastic Load Balancing ロードバランサーに固有の名前を割り当てELBで、複数の目的でロードバランサーを使用しないことです。

Important

スタックレイヤー用に新しい Elastic Load Balancing AWS OpsWorks ロードバランサーを作成することをお勧めします。既存の Elastic Load Balancing ロードバランサーを使用する場合は、まず、他の目的で使用されておらず、インスタンスがアタッチされていないことを確認する必要があります。ロードバランサーがレイヤーにアタッチされると、は既存のインスタンスをすべて OpsWorks 削除し、レイヤーのインスタンスのみを処理するようにロードバランサーを設定します。技術的には、Elastic Load Balancing コンソールを使用したり、レイヤーAPIにアタッチした後にロードバランサーの設定を変更したりすることは可能ですが、そうしないでください。変更は永続的ではありません。

Elastic Load Balancing ロードバランサーをレイヤーにアタッチするには

1. まだ作成していない場合は、[Elastic Load Balancing コンソール](#)、API、または を使用してCLI、スタックのリージョンにロードバランサーを作成します。ロードバランサーを作成する場合、以下の作業を行います。
 - アプリケーションに適したヘルスチェック ping パスを指定してください。

デフォルトの ping パスは /index.html であるため、アプリケーションのルートに index.html が含まれていない場合、適切な ping パスを指定する必要があります。そうしないと、ヘルスチェックは失敗します。

- [Connection Draining](#) を使用する場合は、この機能が有効になっていて、適切なタイムアウト値が設定されていることを確認します。

詳細については、「[Elastic Load Balancing](#)」を参照してください。

2. バランスを取りたい [レイヤーを作成](#) するか、[既存のレイヤーのネットワーク設定を編集](#) します。

Note

カスタムレイヤーの作成時にロードバランサーをアタッチすることはできません。レイヤーの設定を編集する必要があります。

3. Elastic Load Balancing で、レイヤーにアタッチするロードバランサーを選択し、スタックで Connection Draining AWS OpsWorks をサポートするかどうかを指定します。

ロードバランサーをレイヤーにアタッチすると、AWS OpsWorks スタックはスタックのインスタンスで [Configure ライフサイクルイベント](#) をトリガーして変更を通知します。また、ロードバランサーをデタッチすると AWS OpsWorks、スタックは Configure イベントもトリガーします。

Note

インスタンスが起動すると、AWS OpsWorks スタックは [Setup レシピ](#) と [Deploy レシピ](#) を [実行](#) し、パッケージをインストールしてアプリケーションをデプロイします。これらのレシピが完了すると、インスタンスはオンライン状態になり、AWS OpsWorks スタックはインスタンスを Elastic Load Balancing に登録します。AWS OpsWorks スタックは、インスタンスがオンラインになった後に Configure イベントもトリガーします。これは、Elastic Load Balancing 登録と Configure レシピが同時に実行される可能性があり、Configure レシピの終了前にインスタンスが登録される可能性があることを意味します。インスタンスが Elastic Load Balancing に登録される前にレシピを終了させるには、レイヤーの Setup または Deploy ライフサイクルイベントにレシピを追加する必要があります。詳細については、「[レシピの実行](#)」を参照してください。

ロードバランサーからインスタンスを削除した方がよい場合があります。たとえば、アプリケーションを更新するときは、アプリケーションを 1 つのインスタンスにデプロイし、各インスタンスにデプロイする前にアプリケーションが正しく動作していることを確認することをお勧めします。通常、

ロードバランサーからそのインスタンスを削除するため、更新を確認するまでユーザーリクエストを受信しません。

ロードバランサーからオンラインインスタンスAPIを一時的に削除するには、Elastic Load Balancing コンソールまたは を使用する必要があります。コンソールを使用する方法を次に説明します。

ロードバランサーから一時的にインスタンスを削除するには

1. [Amazon EC2コンソール](#)を開き、ロードバランサーを選択します。
2. 適切なロードバランサーを選択し、[Instances] タブを開きます。
3. インスタンスの [Actions] (アクション) 列で [Remove from Load Balancer] (ロードバランサーから移動) を選択します。
4. 完了したら、[Edit Instances] を選択し、インスタンスをロードバランサーに戻します。

Important

Elastic Load Balancing コンソールまたは を使用してロードバランサーからインスタンスAPIを削除する場合は、Elastic Load Balancing を使用してインスタンスに戻す必要もあります。AWS OpsWorks スタックは、他のサービスコンソールまたは で実行するオペレーションを認識しないためAPIs、インスタンスがロードバランサーに返されることはありません。AWS OpsWorks スタックはインスタンスを に追加し直す場合がありますがELB、これは保証された動作ではなく、すべてのケースで発生するわけではありません。

次のように、インスタンスの特定のセットに複数のロードバランサーをアタッチできます。

複数のロードバランサーをアタッチするには

1. [Elastic Load Balancing コンソール](#)、API、または CLIを使用して、ロードバランサーのセットを作成します。
2. ロードバランサーごとに[カスタムレイヤーを作成](#)し、該当するロードバランサーをアタッチします。これらのレイヤーにカスタムレシピを実装する必要はありません。デフォルトのカスタムレイヤーで十分です。
3. 各カスタムレイヤーに[インスタンスのセットを追加](#)します。

[Instances] ページに移動して、適切なロードバランサーの名前をクリックすると、ロードバランサーのプロパティを確認できます。

Host Name	Status	Size	Type	AZ	Public IP	Actions
php-app2	online	c1.medium	24/7	us-west-2b	54.214.153.69	stop
php-app1	online	c1.medium	24/7	us-west-2a	50.112.48.186	stop

Using ELB: My-Stack-PHP | visit URL

ELB このページには、関連するインスタンスDNSの名前とヘルスステータスを含むロードバランサーの基本プロパティが表示されます。スタックがで実行されている場合VPC、ページにはアベイラビリティゾーンではなくサブネットが表示されます。緑のチェックマークは、正常なインスタンスを示します。名前をクリックすると、ロードバランサーを通じてサーバーに接続できます。

ELB My-Stack-PHP

[Disconnect ELB](#)

Elastic Load Balancing associates your load balancer with your EC2 instances using IP addresses. [Learn more.](#)

Settings

DNS Name	My-Stack-PHP-1556928710.us-west-2.elb.amazonaws.com
Layer	PHP App Server
Region	us-west-2

us-west-2a	1	us-west-2b	1
php-app1 ●	✓	php-app2 ●	✓

Amazon RDS Service Layer

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Amazon RDSサービスレイヤーは Amazon RDSインスタンスを表します。レイヤーは既存の Amazon RDSインスタンスのみを表すことができます。このインスタンスは、[Amazon RDSコンソール](#)または [を使用して個別に作成する必要がありますAPI](#)。

Amazon RDSサービスレイヤーをスタックに組み込む基本的な手順は次のとおりです。

1. Amazon RDSコンソール、API、または CLIを使用してインスタンスを作成します。

インスタンスの ID、マスターユーザー名、マスターパスワード、データベース名は必ず記録しておいてください。

2. Amazon RDSレイヤーをスタックに追加するには、Amazon RDSインスタンスをスタックに登録します。
3. レイヤーをアプリケーションにアタッチします。これにより、Amazon RDSインスタンスの接続情報がアプリケーションの[deploy属性](#)に追加されます。
4. 言語固有のファイルまたはdeploy属性内の情報を使用して、アプリケーションを Amazon RDS インスタンスに接続します。

アプリケーションをデータベースサーバーに接続する方法の詳細については、「[the section called “データベースへの接続”](#)」を参照してください。

Warning

インスタンスのマスターパスワードおよびユーザー名に使用した文字に、アプリケーションサーバーが対応していることを確認します。たとえば、Java App Server レイヤーでは、いずれかの文字列に XML を含めると解析エラーが発生し、Tomcat サーバーが起動できなくなります。

トピック

- [セキュリティグループの指定](#)
- [スタックへの Amazon RDS インスタンスの登録](#)
- [Amazon RDS Service Layers とアプリの関連付け](#)
- [スタックからの Amazon RDS Service Layer の削除](#)

セキュリティグループの指定

AWS OpsWorks スタックで Amazon RDS インスタンスを使用するには、データベースまたは VPC セキュリティグループが適切な IP アドレスからのアクセスを許可する必要があります。本稼働環境用のセキュリティグループでは、データベースにアクセスする必要のある IP アドレスにそのアクセス権を限定するのが一般的です。通常、データベースの管理に使用するシステムのアドレスと、データベースにアクセスする必要のある AWS OpsWorks スタックインスタンスが含まれます。AWS OpsWorks スタックは、リージョンで最初のスタックを作成するときに、レイヤーのタイプごとに Amazon EC2 セキュリティグループを自動的に作成します。スタックインスタンスへのアクセスを提供する簡単な方法は、適切な AWS OpsWorks スタックセキュリティグループを Amazon AWS OpsWorks RDS インスタンスまたは VPC に割り当てることです。

既存の Amazon RDS インスタンスのセキュリティグループを指定するには

1. <https://console.aws.amazon.com/rds/> で Amazon RDS コンソールを開きます。
2. ナビゲーションペインのインスタンスをクリックし、適切な Amazon RDS インスタンスを選択します。[Instance Actions]、[Modify] の順にクリックします。
3. [Security Group] の一覧から次のセキュリティグループを選択し、[Continue]、[Modify DB Instance] の順にクリックしてインスタンスを更新します。
 - AWS-OpsWorks-DB-Master-Server (*security_group_id*) セキュリティグループ。
 - インスタンスがデータベースに接続する、アプリケーションサーバーレイヤーのセキュリティグループ。グループ名には、レイヤー名を含めます。例えば、PHPApp Server インスタンスへのデータベースアクセスを提供するには、AWS-OpsWorksPHP-App-Server グループを指定します。

新しい Amazon RDS インスタンスを作成する場合は、DB インスタンス起動ウィザードの「詳細設定の構成」ページで適切な AWS OpsWorks スタックセキュリティグループを指定できます。このウィザードの使用方法については、[「DBSQL インスタンスの作成」](#)および[「DB インスタンスでの SQL データベースへの接続」](#)を参照してください。

VPC セキュリティグループを指定する方法については、[「VPC のセキュリティグループ」](#)を参照してください。

スタックへの Amazon RDS インスタンスの登録

スタックに Amazon RDS サービスレイヤーを追加するには、スタックにインスタンスを登録する必要があります。

Amazon RDSインスタンスをスタックに登録するには

1. AWS OpsWorks スタックコンソールで、ナビゲーションペインの Layer をクリックし、+ Layer または Add a Layer をクリックして Add Layer ページを開き、RDSタブをクリックします。
2. 必要に応じて、スタックのサービスロールを更新します (「[スタックのサービスロールの更新](#)」を参照)。
3. RDS タブをクリックして、使用可能な Amazon RDSインスタンスを一覧表示します。

Note

アカウントに Amazon RDSインスタンスがない場合は、RDSタブでRDSインスタンスの追加をクリックして作成できます。これにより、Amazon RDSコンソールに移動し、DB インスタンスの起動ウィザードを起動できます。[Amazon RDSコンソール](#)に直接移動してDB インスタンスの起動をクリックするか、Amazon RDSAPIまたはを使用することもできますCLI。Amazon RDSインスタンスの作成方法の詳細については、「[Amazon の開始方法RDS](#)」を参照してください。

4. 目的のインスタンスを選択し、[User] と [Password] に、それぞれ適切なユーザーとパスワードの値を設定して、[Register to Stack] をクリックします。

Important

Amazon RDSインスタンスの登録に使用するユーザーとパスワードが、有効なユーザーとパスワードに対応していることを確認する必要があります。対応していない場合、アプリケーションからインスタンスに接続できません。ただし、[レイヤーを編集](#)して、ユーザーとパスワードに有効な値を指定したうえで、再度アプリケーションをデプロイすることはできます。

Add Layer

OpsWorks RDS

Instance Identifier	Engine	Storage (GB)	Type	Status	Multi-AZ	Availability Zone
<input checked="" type="radio"/> opsinstance2	mysql	5	t1.micro	available	No	us-east-1a

Connection Details for opsinstance2

User:

Password: [SHOW](#)

Please verify that OpsWorks can connect to your RDS Instance by setting [Security Groups](#) on that instance. [Learn more.](#)

[Cancel](#) [Register with Stack](#)

Amazon RDSサービスレイヤーをスタックに追加すると、AWS OpsWorks スタックは ID を割り当て、関連する Amazon RDS設定を [スタック設定およびデプロイ属性の属性に追加します](#)。
`[:opsworks][:stack]`

Note

登録された Amazon RDSインスタンスのパスワードを変更する場合は、AWS OpsWorks スタックのパスワードを手動で更新してから、アプリケーションを再デプロイして、スタックのインスタンスのスタック設定とデプロイ属性を更新する必要があります。

トピック

- [スタックのサービスロールの更新](#)

スタックのサービスロールの更新

すべてのスタックには、AWS OpsWorks [IAMサービスロール](#)があり、スタックがユーザーに代わって他の AWS サービスで実行できるアクションを指定します。Amazon RDSインスタンスをスタックに登録するには、そのサービスロールがスタックに Amazon AWS OpsWorks へのアクセス許可を付与する必要がありますRDS。

Amazon RDSサービスレイヤーをスタックの1つに初めて追加する場合、サービスロールに必要なアクセス許可が不足している可能性があります。その場合は、レイヤーの追加ページのRDSタブをクリックすると、以下が表示されます。

Add Layer



To use RDS instances, your OpsWorks IAM role needs to have an RDS instances access policy.

Update

更新をクリックして、AWS OpsWorks スタックがサービスロールのポリシーを次のように更新します。

```
{
  "Statement": [
    {
      "Action": [
        "ec2:*",
        "iam:PassRole",
        "cloudwatch:GetMetricStatistics",
        "elasticloadbalancing:*",
        "rds:*"
      ],
      "Effect": "Allow",
      "Resource": ["*"]
    }
  ]
}
```

Note

更新を実行する必要があるのは1回だけです。更新済みのロールがその後すべてのスタックで自動的に使用されます。

Amazon RDS Service Layers とアプリの関連付け

Amazon RDSサービスレイヤーを追加したら、アプリに関連付けることができます。

- Amazon RDSレイヤーをアプリに関連付けるには、[アプリの作成時](#)、または後で[アプリの設定を編集します](#)。
- アプリから Amazon RDSレイヤーの関連付けを解除するには、アプリの設定を編集して別のデータベースサーバーを指定するか、サーバーを指定しません。

Amazon RDSレイヤーはスタックの一部であり、別のアプリケーションに関連付けることができません。

Amazon RDSインスタンスをアプリケーションに関連付けると、AWS OpsWorks スタックはデータベース接続情報をアプリケーションのサーバーに配置します。各サーバーインスタンスのアプリケーションは、この情報を使用して、データベースに接続できます。Amazon RDSインスタンスに接続する方法の詳細については、「」を参照してください[the section called “データベースへの接続”](#)。

スタックからの Amazon RDS Service Layer の削除

スタックから Amazon RDSサービスレイヤーを削除するには、登録を解除します。

Amazon RDSサービスレイヤーの登録を解除するには

1. ナビゲーションペインの Layers をクリックし、Amazon RDSサービスレイヤーの名前をクリックします。
2. [Deregister] をクリックし、レイヤーの登録を解除することを確認します。

この手順では、スタックからレイヤーを削除しますが、基盤となる Amazon RDSインスタンスは削除されません。インスタンスとデータベースはアカウント内に維持され、他のスタックに登録することができます。インスタンスCLIを削除するには、Amazon RDSコンソール、API、または を使用する必要があります。詳細については、「[DB インスタンスの削除](#)」を参照してください。

ECS クラスターレイヤー

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

[Amazon Elastic Container Service サービス](#) (Amazon ECS) は、コンテナインスタンスと呼ばれる Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのクラスター上の Docker コンテナを管

理します。クラスターレイヤーは Amazon ECS ECSクラスターを表し、次のような機能を提供することでクラスター管理を簡素化します。

- 効率化されたインスタンスコンテナのプロビジョニングと管理
- コンテナのインスタンスオペレーティングシステムおよびパッケージの更新
- ユーザー許可の管理
- コンテナインスタンスのパフォーマンスのモニタリング
- Amazon Elastic Block Store (Amazon EBS) ボリューム管理
- パブリック IP アドレスと Elastic IP アドレスの管理
- セキュリティグループの管理

ECS クラスターレイヤーには、次の制限と要件があります。

- レイヤーは、[デフォルトの VPC](#)を含むVPC、 で実行されている Chef 11.10 または Chef 12 Linux スタックでのみ使用できます。
- レイヤーのインスタンスは、次のオペレーティングシステムのいずれかを実行している必要があります。
 - Amazon Linux 2
 - Amazon Linux 2018.03
 - Amazon Linux 2017.09
 - Amazon Linux 2017.03
 - Amazon Linux 2016.09
 - Amazon Linux 2016.03
 - Amazon Linux 2015.09
 - Amazon Linux 2015.03
 - Ubuntu 18.04 LTS
 - Ubuntu 16.04 LTS
 - Ubuntu 14.04 LTS
 - カスタム
- このレイヤーのインスタンスの [AWS OpsWorks スタックエージェントバージョン](#)は 3425-20150727112318 以降のバージョンである必要があります。

トピック

- [ECS クラスターレイヤーをスタックに追加する](#)
- [ECS クラスターの管理](#)
- [スタックからECSクラスターレイヤーを削除する](#)

ECS クラスターレイヤーをスタックに追加する

AWS OpsWorks スタックは、既存の Amazon ECS クラスターのコンテナインスタンスを起動および維持するプロセスを簡素化します。クラスターやタスクなどの他の Amazon ECS エンティティを作成または起動するには、Amazon ECS コンソール、コマンドラインインターフェイス (CLI)、または `aws` を使用します API。(詳細については、「[Amazon Elastic Container Service Developer Guide](#)」(Amazon Elastic Container Service デベロッパーガイド) を参照してください。) その後、クラスターレイヤーを作成して ECS クラスターをスタックに関連付けることができます。このレイヤーを使用して、AWS OpsWorks スタックでクラスターを管理できます。

以下のとおり、スタックとクラスターを関連付けることができます。

- 各スタックには、単一のクラスターを表す 1 ECS つのクラスターレイヤーを含めることができます。
- クラスターは 1 つのスタックのみに関連付けることができます。

スタックに ECS クラスターレイヤーを追加する前に、通常は AWS OpsWorks という名前の AWS Identity and Access Management (IAM) スタック (IAM) サービスロールを更新して `aws-opsworks-service-role`、AWS OpsWorks スタックが ECS ユーザーに代わって Amazon とやり取りできるようにする必要があります。サービスロールの詳細については、「[AWS OpsWorks スタックがユーザーに代わって動作することを許可する](#)」を参照してください。

ECS クラスターレイヤーを初めて作成すると、コンソールに更新ボタンが表示されます。このボタンを選択すると AWS OpsWorks、スタックにロールの更新を指示できます。その後、AWS OpsWorks スタックにレイヤーの追加ページが表示され、スタックにレイヤーを追加できます。サービスロールの更新は 1 回だけでかまいません。その後、更新されたロールを使用して、任意のスタックに ECS クラスターレイヤーを追加できます。

Note

必要に応じて、以下のように `ecs:*` 許可を既存のポリシーに許可を追加することで、サービスロールのポリシーを手動で更新できます。

```
{
  "Statement": [
    {
      "Action": [
        "ec2:*",
        "iam:PassRole",
        "cloudwatch:GetMetricStatistics",
        "elasticloadbalancing:*",
        "rds:*",
        "ecs:*"
      ],
      "Effect": "Allow",
      "Resource": ["*"]
    }
  ]
}
```

スタックとクラスターを関連付けるには、スタックへのクラスターの登録と関連付けられたレイヤーの作成という2つのアクションが必要になります。AWS OpsWorks スタックコンソールは、これらのステップを組み合わせます。レイヤーの作成により、指定されたクラスターが自動的に登録されます。AWS OpsWorks スタック API、またはを使用する場合はSDK、クラスターを登録し、関連するレイヤーを作成するにはCLI、別のオペレーションを使用する必要があります。コンソールを使用してスタックにECSクラスターレイヤーを追加するには、レイヤーを選択し、+レイヤーまたはレイヤーの追加を選択してから、ECSクラスターレイヤータイプを選択します。

Add Layer

OpsWorks RDS

Layer type Looking for a different Layer type? [Let us know.](#)

The ECS Cluster layer registers a cluster with Amazon EC2 Container Service and acts as a blueprint for ECS instances managed by OpsWorks. [Learn More.](#)

ECS Cluster

EC2 Instance profile

This profile has access to ECS.

Cancel Add Layer

[Add Layer] (追加のレイヤー) ページには、以下の設定オプションがあります。

ECS クラスター

スタックに登録する Amazon ECS クラスター。

EC2 インスタンスプロファイル

クラスターの Amazon Elastic Compute Cloud (Amazon EC2) インスタンスプロファイル。このプロファイルは、クラスターのコンテナインスタンスで実行されているアプリケーションに、Amazon を含む他の AWS サービスにアクセスするためのアクセス許可を付与します ECS。最初の ECS クラスターレイヤーを作成するときは、AWS OpsWorks スタック ECS にアクセス可能な新しいプロファイルを選択して、という名前の必要なプロファイルを作成します `aws-opsworks-ec2-role-with-ecs`。その後、そのプロファイルを後続のすべての ECS クラスターレイヤーに使用できます。インスタンスプロファイルの詳細については、「[EC2 インスタンスで実行されるアプリケーションのアクセス許可の指定](#)」を参照してください。

[レイヤー設定を編集](#)して、以下のその他の設定を指定できます。

- レイヤーに [Elastic Load Balancing のロードバランサーをアタッチ](#)します。

このアプローチは一部のユースケースに適しているかもしれませんが、Amazon ECS ではより高度なオプションを提供しています。詳細については、「[サービスロードバランシング](#)」を参照してください。

- [パブリック IP アドレスまたは Elastic IP アドレス](#)のコンテナインスタンスへの自動割り当てを許可するかどうかを指定します。

両方のアドレスタイプの自動割り当てを無効にすると、サブネットに が正しく設定されていない限り、インスタンスはオンラインになりません NAT。詳細については、「[でのスタックの実行 VPC](#)」を参照してください。

ECS クラスターの管理

ECS クラスターレイヤーを作成したら、次のように AWS OpsWorks スタックを使用してクラスターを管理できます。

コンテナインスタンスのをプロビジョニングと管理

初期状態では、元のクラスターであっても、ECSクラスターレイヤーにはコンテナインスタンスは含まれません。1つのオプションは、以下のような適切な組み合わせを使用して、レイヤーのインスタンスを管理することです。

- 手動で [24/7 インスタンス](#) をレイヤーに追加して、不要となったら [それらを削除](#) します。
- [時間ベースのインスタンス](#) を追加することで、スケジュールにあるインスタンスの追加または削除を行います。
- スタックホスト AWS OpsWorks メトリクスまたは CloudWatch アラームに基づいてインスタンスを追加または削除するには、[負荷ベースのインスタンス](#) をレイヤーに追加します。

Note

Amazon ECSがスタックのデフォルトのオペレーティングシステムでサポートされていない場合は、コンテナインスタンスを作成するときに、サポートされているオペレーティングシステムとして Amazon Linux 2、Amazon Linux 2018.03、Amazon Linux 2017.09、Amazon Linux 2017.03、Amazon Linux 2016.09、Amazon Linux 2016.03、Amazon Linux 2015.09、Amazon Linux 2015.03、Ubuntu 18.04LTS、Ubuntu 16.04LTS、Ubuntu 14.04LTS、または Custom を明示的に指定する必要があります。ECS 最適化された を使用して ECSレイヤーにインスタンスAMIを作成しないでください。これにはAMI、既に ECS エージェントが含まれています。AWS OpsWorks スタックはインスタンスのセットアッププロセス中にECSエージェントのインストールも試みるため、競合によりセットアップが失敗する可能性があります。

詳細については、「」を参照してください[サーバー数の最適化](#)。AWS OpsWorks スタックは AWS-OpsWorksECS-Cluster セキュリティグループを各インスタンスに割り当てます。新しいインスタンスの起動が完了すると、AWS OpsWorks スタックは Docker と Amazon ECS エージェントをインストールし、インスタンスをクラスターに登録することで、そのインスタンスをコンテナインスタンスに変換します。

既存のコンテナインスタンスを使用する場合は、[それらをスタックに登録し](#)、[ECSクラスターレイヤーに割り当てる](#) ことができます。インスタンスは、サポートされているオペレーティングシステム、Amazon Linux 2015.03 以降、または Ubuntu 14.04 LTS以降を実行している必要があることに注意してください。

Note

コンテナインスタンスはECS、クラスターレイヤーと別の組み込みレイヤーの両方に属することはできません。ただし、コンテナインスタンスは [クラスターレイヤー](#) と 1 ECS つ以上の [カスタムレイヤー](#) に属することができます。

オペレーティングシステムとパッケージの更新を実行する

新しいインスタンスの起動が完了すると、AWS OpsWorks スタックによって最新の更新がインストールされます。その後、AWS OpsWorks スタックを使用してコンテナインスタンスを最新の状態に保つことができます。詳細については、「[セキュリティ更新の管理](#)」を参照してください。

ユーザーアクセス許可の管理

AWS OpsWorks スタックは、ユーザーのSSHキーの管理など、コンテナインスタンスに対するアクセス許可を管理する簡単な方法を提供します。詳細については、[ユーザー許可の管理](#)および[SSH アクセスの管理](#)を参照してください。

パフォーマンスメトリクスのモニタリング

AWS OpsWorks スタックには、スタック、レイヤー、または個々のインスタンスのパフォーマンスメトリクスをモニタリングするさまざまな方法が用意されています。詳細については、「[モニタリング](#)」を参照してください。

Amazon を通じて、タスクやサービスの作成など、他の管理タスクを処理しますECS。詳細については、[Amazon Elastic Container Service デベロッパーガイド](#)を参照してください。

Note

Amazon ECSコンソールでクラスターのページに直接移動するには、インスタンスを選択し、ECSクラスターレイヤーのセクションの右上隅近くにあるECSクラスターを選択します。

スタックからECSクラスターレイヤーを削除する

クラスターが不要になったら、ECSクラスターレイヤーを削除し、関連付けられたクラスターの登録を解除します。スタックからクラスターを削除するには、クラスターの登録解除と関連付けられた

レイヤーの削除という、2つのアクションが必要になります。AWS OpsWorks スタックコンソールはこれらのステップを組み合わせます。レイヤーを削除すると、指定したクラスターが自動的に登録解除されます。AWS OpsWorks スタック API、またはを使用する場合はSDK、別のオペレーションを使用してクラスターの登録を解除しCLI、関連するレイヤーを削除する必要があります。

コンソールを使用してECSクラスターレイヤーを削除するには

1. タスクのシャットダウン方法を制御したい場合は、Amazon ECSコンソール、API、またはCLIを使用してクラスターのサービスをスケールダウンおよび削除します。詳細については、「[Amazon ECSリソースのクリーンアップ](#)」を参照してください。
2. [レイヤーのインスタンスを停止](#)してから、[それらを削除](#)します。コンテナインスタンスを停止すると、AWS OpsWorks スタックは実行中のタスクを自動的に停止し、クラスターからインスタンスを登録解除して、インスタンスを終了します。

Note

スタックに既存のコンテナインスタンスを登録している場合は、[レイヤーからインスタンスの割り当てを解除](#)してから、[登録を解除](#)できます。これにより、ECS制御するインスタンスが返されます。

3. [レイヤーを削除](#)します。AWS OpsWorks スタックは関連付けられたクラスターを登録解除しますが、削除はしません。クラスターは Amazon に残りますECS。

カスタム AWS OpsWorks スタックレイヤー

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または[AWS プレミアムサポート](#)を通じて AWS サポート チームにお問い合わせください。

カスタムレイヤーには、最小限のレシピのみ含まれています。その後、[カスタムレシピ](#)を実装してレイヤーの[ライフサイクルイベント](#)に割り当てることにより、適切な機能を Layer に追加します。

カスタムレイヤーには次のような設定があります。

Note

AWS OpsWorks スタックは、レイヤーのインスタンスに Ruby を自動的にインストールします。インスタンスで Ruby コードを実行するが、デフォルトの Ruby バージョンを使用しない場合は、カスタムJSONまたはカスタム属性ファイルを使用して、希望するバージョンを指定できます。詳細については、「[Ruby のバージョン](#)」を参照してください。

カスタムレイヤーを作成するための基本的な手順は次のとおりです。

1. パッケージのインストールと設定、設定変更の処理、アプリケーションのデプロイなどに必要な、レシピと関連ファイルが含まれた[クックブック](#)を実装します。

要件によっては、デプロイ解除タスクやシャットダウンタスクを処理するためのレシピが必要になることもあります。詳細については、「[クックブックとレシピ](#)」を参照してください。

2. カスタムレイヤーを作成します。
3. 適切な[ライフサイクルイベント](#)にレシピを割り当てます。

次に、レイヤーにインスタンスを追加して起動し、アプリケーションをそれらのインスタンスにデプロイします。

Important

カスタムレイヤーのインスタンスにアプリケーションをデプロイするには、デプロイ操作を処理して、アプリケーションをレイヤーの Deploy イベントに割り当てるためのレシピを実装する必要があります。

レイヤーごとのオペレーティングシステムパッケージのインストール

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Chef 12 からは、カスタムレシピを使用して、さまざまなオペレーティングシステムを実行しているレイヤーでパッケージをインストールする必要があります。この方法により、パッケージのインストールについての最大の柔軟性とコントロールが得られます。

例えば、Linux オペレーティングシステムの実行中のレイヤー RedHat、Ubuntu、Amazon バージョンに Apache をインストールするとします。RedHat と Amazon Linux の Apache パッケージはと呼ばれますが httpd、Ubuntu ではと呼ばれます apache2。

パッケージの命名の違いに対応するために、次の例のレシピで、次のような構文を使用できます。レシピは各オペレーティングシステムに対して適切な Apache パッケージをインストールします。この例は、[Chef のドキュメント](#)に基づいています。

```
package "Install Apache" do
  case node[:platform]
    when "redhat", "amazon"
      package_name "httpd"
    when "ubuntu"
      package_name "apache2"
  end
end
```

package リソースを使用してパッケージを管理する方法の詳細については、Chef のドキュメントの[パッケージ](#)のページを参照してください。

または、Chef Recipe DSL (ドメイン固有の言語) の `value_for_platform` ヘルパーメソッドを使用することもできます。これにより、より簡潔に同じことを実現できます。

```
package "Install Apache" do
  package_name value_for_platform(
    ["redhat", "amazon"] => { "default" => "httpd" },
    ["ubuntu"] => { "default" => "apache2" }
  )
end
```

`value_for_platform` ヘルパーメソッドの使用については、[「レシピについてDSL」](#)を参照してください。

インスタンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

インスタンスは、Amazon EC2 インスタンスなどのコンピューティングリソースを表し、アプリケーションの提供やトラフィックのバランシングなどを処理します。インスタンスのオペレーティングシステムには、複数の Linux ディストリビューション、または Windows Server 2012 R2 のうち任意のものを使用できます。

次のいずれかの方法で、インスタンスをスタックに追加できます。

- AWS OpsWorks スタックを使用して、スタックにインスタンスを追加します。追加するインスタンスは、Amazon EC2 インスタンスを表します。
- Linux ベースのスタックの場合、Amazon EC2 で作成したインスタンスや、自社のハードウェアで動作するオンプレミスのインスタンスなど、別の場所で作成したインスタンスを登録することができます。

その後、AWS OpsWorks スタックを使用して、AWS OpsWorks スタックで作成されたインスタンスとほぼ同じ方法でこれらのインスタンスを管理できます。

このセクションでは、AWS OpsWorks スタックを使用してインスタンスを作成および管理する方法を説明します。

トピック

- [AWS OpsWorks スタックインスタンスの使用](#)
- [AWS OpsWorks Stacks の外部で作成されたコンピューティングリソースの使用](#)
- [インスタンス設定の編集](#)
- [AWS OpsWorks スタックインスタンスの削除](#)
- [SSH を使用した Linux インスタンスへのログイン](#)

- [RDP を使用した Windows インスタンスへのログイン](#)

AWS OpsWorks スタックインスタンスの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックを使用してインスタンスを作成し、スタックに追加できます。

トピック

- [AWS OpsWorks スタックオペレーティングシステム](#)
- [レイヤーへのインスタンスの追加](#)
- [カスタム AMI の使用](#)
- [24/7 インスタンスの手動による起動、停止、再起動](#)
- [時間ベースおよび負荷ベースのインスタンスによる負荷の管理](#)

AWS OpsWorks スタックオペレーティングシステム

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは、Amazon および Ubuntu Linux ディストリビューション、Microsoft Windows Server など、いくつかの組み込みオペレーティングシステムの 64 ビットバージョンをサポートしています。一般的な注意事項をいくつか示します。

- スタックのインスタンスは、Linux または Windows を実行できます。

スタックは、インスタンスごとに異なる Linux バージョンまたはディストリビューションを実行できますが、Linux インスタンスと Windows インスタンスを混在させることはできません。

- [カスタム AMIs](#) (Amazon マシンイメージ) を使用できますが、このセクションのトピックで説明されている AWS OpsWorks スタックがサポートする AMIs のいずれかに基づいている必要があります。カスタム AMI またはコミュニティで作成された AMI から作成された他のオペレーティングシステム (CentOS 6.x など) を使用してインスタンスを作成または登録できる場合もありますが、そのような方法は公式にはサポートされていません。

- [Linux オペレーティングシステム:](#)

- [Microsoft Windows Server](#)

- [インスタンスを手動で開始および停止する](#) ことも、AWS OpsWorks スタックでインスタンス数を [自動的にスケールする](#) こともできます。

どのスタックでも、時間ベースの自動スケーリングを使用できます。Linux スタックでは、負荷ベースのスケーリングも使用できます。

- AWS OpsWorks スタックを使用して Amazon EC2 インスタンスを作成するだけでなく、AWS OpsWorks スタックの外部で作成された [Linux スタックにインスタンスを登録](#) することもできます。

これには、Amazon EC2 インスタンスと、独自のハードウェアを実行するインスタンスが含まれます。ただし、そのインスタンスがサポートされている Linux ディストリビューションの 1 つを実行している必要があります。Amazon EC2 インスタンスまたはオンプレミスの Windows インスタンスを登録することはできません。

AWS OpsWorks Stacks [DescribeOperatingSystems](#) API を実行して、サポートされているオペレーティングシステムと Chef のサポートされているバージョンのリストを返すことができます。以下は AWS CLI を使用したコマンドの例です。

```
aws opsworks describe-operating-systems
```

以下に、応答の例を示します。

```
{
  "OperatingSystems": [
    {
      "Name": "Amazon Linux",
```

```
"Id": "Amazon Linux",
>Type": "Linux",
>ConfigurationManagers": [
>  {
>    "Name": "Chef",
>    "Version": "11.10"
>  },
>  {
>    "Name": "Chef",
>    "Version": "11.4"
>  },
>  {
>    "Name": "Chef",
>    "Version": "0.9"
>  }
>],
>ReportedName": "amazon",
>ReportedVersion": "2014.03",
>Supported": false
},
{
>Name": "Amazon Linux 2",
>Id": "Amazon Linux 2",
>Type": "Linux",
>ConfigurationManagers": [
>  {
>    "Name": "Chef",
>    "Version": "12"
>  }
>],
>ReportedName": "amazon",
>ReportedVersion": "2"
},
{
>Name": "Amazon Linux 2014.09",
>Id": "Amazon Linux 2014.09",
>Type": "Linux",
>ConfigurationManagers": [
>  {
>    "Name": "Chef",
>    "Version": "11.10"
>  },
>  {
>    "Name": "Chef",
```

```
        "Version": "11.4"
      },
      {
        "Name": "Chef",
        "Version": "0.9"
      }
    ],
    "ReportedName": "amazon",
    "ReportedVersion": "2014.09",
    "Supported": false
  },
  {
    "Name": "Amazon Linux 2015.03",
    "Id": "Amazon Linux 2015.03",
    "Type": "Linux",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12"
      },
      {
        "Name": "Chef",
        "Version": "11.10"
      },
      {
        "Name": "Chef",
        "Version": "11.4"
      },
      {
        "Name": "Chef",
        "Version": "0.9"
      }
    ],
    "ReportedName": "amazon",
    "ReportedVersion": "2015.03",
    "Supported": false
  },
  {
    "Name": "Amazon Linux 2015.09",
    "Id": "Amazon Linux 2015.09",
    "Type": "Linux",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
```

```
        "Version": "12"
      },
      {
        "Name": "Chef",
        "Version": "11.10"
      },
      {
        "Name": "Chef",
        "Version": "11.4"
      },
      {
        "Name": "Chef",
        "Version": "0.9"
      }
    ],
    "ReportedName": "amazon",
    "ReportedVersion": "2015.09",
    "Supported": false
  },
  {
    "Name": "Amazon Linux 2016.03",
    "Id": "Amazon Linux 2016.03",
    "Type": "Linux",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12"
      },
      {
        "Name": "Chef",
        "Version": "11.10"
      },
      {
        "Name": "Chef",
        "Version": "11.4"
      },
      {
        "Name": "Chef",
        "Version": "0.9"
      }
    ],
    "ReportedName": "amazon",
    "ReportedVersion": "2016.03"
  },
}
```

```
{
  "Name": "Amazon Linux 2016.09",
  "Id": "Amazon Linux 2016.09",
  "Type": "Linux",
  "ConfigurationManagers": [
    {
      "Name": "Chef",
      "Version": "12"
    },
    {
      "Name": "Chef",
      "Version": "11.10"
    },
    {
      "Name": "Chef",
      "Version": "11.4"
    },
    {
      "Name": "Chef",
      "Version": "0.9"
    }
  ],
  "ReportedName": "amazon",
  "ReportedVersion": "2016.09"
},
{
  "Name": "Amazon Linux 2017.03",
  "Id": "Amazon Linux 2017.03",
  "Type": "Linux",
  "ConfigurationManagers": [
    {
      "Name": "Chef",
      "Version": "12"
    },
    {
      "Name": "Chef",
      "Version": "11.10"
    },
    {
      "Name": "Chef",
      "Version": "11.4"
    },
    {
      "Name": "Chef",

```

```
        "Version": "0.9"
      }
    ],
    "ReportedName": "amazon",
    "ReportedVersion": "2017.03"
  },
  {
    "Name": "Amazon Linux 2017.09",
    "Id": "Amazon Linux 2017.09",
    "Type": "Linux",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12"
      },
      {
        "Name": "Chef",
        "Version": "11.10"
      },
      {
        "Name": "Chef",
        "Version": "11.4"
      },
      {
        "Name": "Chef",
        "Version": "0.9"
      }
    ],
    "ReportedName": "amazon",
    "ReportedVersion": "2017.09"
  },
  {
    "Name": "Amazon Linux 2018.03",
    "Id": "Amazon Linux 2018.03",
    "Type": "Linux",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12"
      },
      {
        "Name": "Chef",
        "Version": "11.10"
      }
    ]
  }
}
```

```
    ],
    "ReportedName": "amazon",
    "ReportedVersion": "2018.03"
  },
  {
    "Name": "CentOS Linux 7",
    "Id": "CentOS Linux 7",
    "Type": "Linux",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12"
      }
    ],
    "ReportedName": "CentOS Linux",
    "ReportedVersion": "7"
  },
  {
    "Name": "Microsoft Windows Server 2012 R2 Base",
    "Id": "Microsoft Windows Server 2012 R2 Base",
    "Type": "Windows",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12.2"
      }
    ],
    "ReportedName": "microsoft windows server",
    "ReportedVersion": "2012 r2 standard",
    "Supported": false
  },
  {
    "Name": "Microsoft Windows Server 2012 R2 with SQL Server Express",
    "Id": "Microsoft Windows Server 2012 R2 with SQL Server Express",
    "Type": "Windows",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12.2"
      }
    ],
    "ReportedName": "microsoft windows server",
    "ReportedVersion": "2012 r2 standard",
    "Supported": false
  }
]
```

```
    },
    {
      "Name": "Microsoft Windows Server 2012 R2 with SQL Server Standard",
      "Id": "Microsoft Windows Server 2012 R2 with SQL Server Standard",
      "Type": "Windows",
      "ConfigurationManagers": [
        {
          "Name": "Chef",
          "Version": "12.2"
        }
      ],
      "ReportedName": "microsoft windows server",
      "ReportedVersion": "2012 r2 standard",
      "Supported": false
    },
    {
      "Name": "Microsoft Windows Server 2012 R2 with SQL Server Web",
      "Id": "Microsoft Windows Server 2012 R2 with SQL Server Web",
      "Type": "Windows",
      "ConfigurationManagers": [
        {
          "Name": "Chef",
          "Version": "12.2"
        }
      ],
      "ReportedName": "microsoft windows server",
      "ReportedVersion": "2012 r2 standard",
      "Supported": false
    },
    {
      "Name": "Microsoft Windows Server 2019 Base",
      "Id": "Microsoft Windows Server 2019 Base",
      "Type": "Windows",
      "ConfigurationManagers": [
        {
          "Name": "Chef",
          "Version": "12.2"
        }
      ],
      "ReportedName": "microsoft windows server",
      "ReportedVersion": "2019 datacenter"
    },
    {
      "Name": "Microsoft Windows Server 2019 with SQL Server Express",
```



```
"Id": "Microsoft Windows Server 2019 with SQL Server Express",
>Type": "Windows",
>ConfigurationManagers": [
>  {
>    "Name": "Chef",
>    "Version": "12.2"
>  }
>],
>ReportedName": "microsoft windows server",
>ReportedVersion": "2019 datacenter"
>},
>{
>  "Name": "Microsoft Windows Server 2019 with SQL Server Standard",
>  "Id": "Microsoft Windows Server 2019 with SQL Server Standard",
>  "Type": "Windows",
>  "ConfigurationManagers": [
>    {
>      "Name": "Chef",
>      "Version": "12.2"
>    }
>  ],
>  "ReportedName": "microsoft windows server",
>  "ReportedVersion": "2019 datacenter"
>},
>{
>  "Name": "Microsoft Windows Server 2019 with SQL Server Web",
>  "Id": "Microsoft Windows Server 2019 with SQL Server Web",
>  "Type": "Windows",
>  "ConfigurationManagers": [
>    {
>      "Name": "Chef",
>      "Version": "12.2"
>    }
>  ],
>  "ReportedName": "microsoft windows server",
>  "ReportedVersion": "2019 datacenter"
>},
>{
>  "Name": "Microsoft Windows Server 2022 Base",
>  "Id": "Microsoft Windows Server 2022 Base",
>  "Type": "Windows",
>  "ConfigurationManagers": [
>    {
>      "Name": "Chef",
```

```
        "Version": "12.2"
      }
    ],
    "ReportedName": "microsoft windows server",
    "ReportedVersion": "2022 datacenter"
  },
  {
    "Name": "Microsoft Windows Server 2022 with SQL Server Express",
    "Id": "Microsoft Windows Server 2022 with SQL Server Express",
    "Type": "Windows",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12.2"
      }
    ],
    "ReportedName": "microsoft windows server",
    "ReportedVersion": "2022 datacenter"
  },
  {
    "Name": "Microsoft Windows Server 2022 with SQL Server Standard",
    "Id": "Microsoft Windows Server 2022 with SQL Server Standard",
    "Type": "Windows",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12.2"
      }
    ],
    "ReportedName": "microsoft windows server",
    "ReportedVersion": "2022 datacenter"
  },
  {
    "Name": "Microsoft Windows Server 2022 with SQL Server Web",
    "Id": "Microsoft Windows Server 2022 with SQL Server Web",
    "Type": "Windows",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12.2"
      }
    ],
    "ReportedName": "microsoft windows server",
    "ReportedVersion": "2022 datacenter"
  }
]
```

```
  },
  {
    "Name": "Red Hat Enterprise Linux 7",
    "Id": "Red Hat Enterprise Linux 7",
    "Type": "Linux",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12"
      },
      {
        "Name": "Chef",
        "Version": "11.10"
      }
    ],
    "ReportedName": "Red Hat Enterprise Linux",
    "ReportedVersion": "7"
  },
  {
    "Name": "Ubuntu 12.04 LTS",
    "Id": "Ubuntu 12.04 LTS",
    "Type": "Linux",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12"
      },
      {
        "Name": "Chef",
        "Version": "11.10"
      },
      {
        "Name": "Chef",
        "Version": "11.4"
      },
      {
        "Name": "Chef",
        "Version": "0.9"
      }
    ],
    "ReportedName": "ubuntu",
    "ReportedVersion": "12.04",
    "Supported": false
  },
}
```

```
{
  "Name": "Ubuntu 14.04 LTS",
  "Id": "Ubuntu 14.04 LTS",
  "Type": "Linux",
  "ConfigurationManagers": [
    {
      "Name": "Chef",
      "Version": "12"
    },
    {
      "Name": "Chef",
      "Version": "11.10"
    }
  ],
  "ReportedName": "ubuntu",
  "ReportedVersion": "14.04"
},
{
  "Name": "Ubuntu 16.04 LTS",
  "Id": "Ubuntu 16.04 LTS",
  "Type": "Linux",
  "ConfigurationManagers": [
    {
      "Name": "Chef",
      "Version": "12"
    }
  ],
  "ReportedName": "ubuntu",
  "ReportedVersion": "16.04"
},
{
  "Name": "Ubuntu 18.04 LTS",
  "Id": "Ubuntu 18.04 LTS",
  "Type": "Linux",
  "ConfigurationManagers": [
    {
      "Name": "Chef",
      "Version": "12"
    }
  ],
  "ReportedName": "ubuntu",
  "ReportedVersion": "18.04"
},
{
```

```
    "Name": "Ubuntu 20.04 LTS",
    "Id": "Ubuntu 20.04 LTS",
    "Type": "Linux",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12"
      }
    ],
    "ReportedName": "ubuntu",
    "ReportedVersion": "20.04"
  },
  {
    "Name": "Custom",
    "Id": "Custom",
    "Type": "Linux",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12"
      },
      {
        "Name": "Chef",
        "Version": "11.10"
      },
      {
        "Name": "Chef",
        "Version": "11.4"
      },
      {
        "Name": "Chef",
        "Version": "0.9"
      }
    ]
  },
  {
    "Name": "CustomWindows",
    "Id": "CustomWindows",
    "Type": "Windows",
    "ConfigurationManagers": [
      {
        "Name": "Chef",
        "Version": "12.2"
      }
    ]
  }
}
```

```
    ]
  }
]
}
```

トピック

- [Linux オペレーティングシステム:](#)
- [Microsoft Windows Server](#)

Linux オペレーティングシステム:

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは、次の Linux オペレーティングシステムの 64 ビットバージョンをサポートしています。

- [Amazon Linux](#) と [Amazon Linux 2](#) (現在サポートされているバージョンについては、「[AWS OpsWorks スタックコンソール](#)」を参照)
- [Ubuntu 20.04 LTS](#)
- [CentOS 7](#)
- [Red Hat Enterprise Linux 7](#)

これらのオペレーティングシステムに基づく [カスタム AMI](#) を使用することもできます。

Linux インスタンスの一般的な注意事項をいくつか示します。

サポートされているパッケージのバージョン

Ruby などのパッケージでサポートされているバージョンとパッチレベルは、以下のセクションで説明するようにオペレーティングシステムとバージョンによって異なります。

更新

デフォルトでは、AWS OpsWorks スタックは、インスタンスの起動 `yum update` `apt-get update` 後に または を自動的に呼び出すことで、Linux インスタンスに最新のセキュリティパッチを適用します。自動更新を無効にするには [CreateInstance](#)、[UpdateInstance](#)、[CreateLayer](#)、または [UpdateLayer](#) アクション、または同等の [AWS SDK](#) メソッドまたは [AWS CLI](#) コマンドを使用して、`InstallUpdatesOnBoot` パラメータを に設定します `false`。

サービスの中断を避けるため、AWS OpsWorks スタックはインスタンスがオンラインになった後に更新を自動的にインストールしません。オンラインインスタンスのオペレーティングシステムは、[Upgrade Operating System スタックコマンド](#) を実行することでいつでも手動で更新できます。セキュリティの更新を管理する方法の詳細については、「[セキュリティ更新の管理](#)」を参照してください。

AWS OpsWorks スタックがインスタンスを更新する方法をより詳細に制御するには、サポートされているオペレーティングシステムのいずれかに基づいてカスタム AMI を作成します。たとえば、カスタム AMI を使用して、インスタンスにインストールするパッケージのバージョンを指定できます。Linux ディストリビューションによってサポートタイムラインとパッケージマージポリシーが異なるため、要件に最適な方法を検討する必要があります。詳細については、「[カスタム AMI の使用](#)」を参照してください。

ホストファイル

各オンラインインスタンスには、IP アドレスをホスト名にマッピングする `/etc/hosts` ファイルがあります。AWS OpsWorks スタックには、各インスタンスの `hosts` ファイル内のすべてのスタックのオンラインインスタンスのパブリックアドレスとプライベートアドレスが含まれます。例えば、2 つの Node.js アプリケーションサーバーインスタンス (`nodejs-app1` と `nodejs-app2`) と、1 つの MySQL インスタンス (`db-master1`) があるとします。`nodejs-app1` インスタンスの `hosts` ファイルは以下の例のようなものであり、他のインスタンスにも同様の `hosts` ファイルがあります。

```
...
# OpsWorks Layer State
192.0.2.0 nodejs-app1.localdomain nodejs-app1
10.145.160.232 db-master1
198.51.100.0 db-master1-ext
10.243.77.78 nodejs-app2
203.0.113.0 nodejs-app2-ext
10.84.66.6 nodejs-app1
```

```
192.0.2.0 nodejs-app1-ext
```

AWS OpsWorks スタックエージェントプロキシのサポート

Chef 11.10 以降の AWS OpsWorks スタック用の スタックエージェントには、プロキシサーバーの基本サポートが含まれています。プロキシサーバーは通常、分離された VPCs で使用されます。プロキシサーバーのサポートを有効にするには、HTTP と HTTPS のトラフィックに適した設定を定義した `/etc/environment` ファイルがインスタンスに必要です。このファイルは以下のようになります (強調表示されたテキストはプロキシサーバーの URL とポートに置き換えます)。

```
http_proxy="http://myproxy.example.com:8080/"
https_proxy="http://myproxy.example.com:8080/"
no_proxy="169.254.169.254"
```

プロキシのサポートを有効にするには、該当する [ファイルを含むカスタム AMI](#) を作成/`etc/environment` し、その AMI を使用してインスタンスを作成することをお勧めします。

Note

カスタムレシピを使用してインスタンスに `/etc/environment` ファイルを作成することはお勧めしません。AWS OpsWorks スタックでは、カスタムレシピが実行される前に、セットアッププロセスの早い段階でプロキシサーバーデータが必要です。

トピック

- [Amazon Linux](#)
- [Ubuntu LTS](#)
- [CentOS](#)
- [Red Hat Enterprise Linux](#)

Amazon Linux

AWS OpsWorks スタックは、Amazon Linux および Amazon Linux 2 の 64 ビットバージョンをサポートしています。Amazon Linux では定期的な更新やパッチに加えて、新しいバージョンを約 6 か月ごとにリリースしており、大きな変更が実施される場合もあります。スタックまたは新しいインス

タンスを作成する際に、使用する Amazon Linux のバージョンを指定する必要があります。AWS から新しいバージョンがリリースされたとき、ユーザーが明示的にバージョンを変更するまで、インスタンスでは指定されたバージョンが引き続き実行されます。新しい Amazon Linux バージョンのリリース後 4 週間は移行期間となっており、その間は古いバージョン向けの定期的な更新が引き続き AWS で提供されます。移行期間の終了後も、インスタンスで古いバージョンを引き続き実行できますが、AWS では更新が提供されなくなります。詳細については、「[Amazon Linux AMI に関するよくある質問](#)」を参照してください。

Amazon Linux の新しいバージョンがリリースされたら、インスタンスがセキュリティの更新を引き続き受け取ることができるように、移行期間の間に新しいバージョンに更新することをお勧めします。本稼働用スタックのインスタンスを更新する前に、新しいインスタンスを起動し、新しいバージョンでアプリケーションが正常に実行されるかどうかを確認することをお勧めします。その後で、本稼働用スタックインスタンスを更新できます。

Note

デフォルトで、Amazon Linux に基づくカスタム AMI は、新しいバージョンがリリースされると自動的に更新されます。カスタム AMI は特定の Amazon Linux バージョンにロックしておき、新しいバージョンをテストするまで更新を延期できるようにすることをお勧めします。詳細については、「[AMI を特定のバージョンに固定するにはどうすればよいですか?](#)」を参照してください。

テンプレートを使用して Amazon Linux を実行しているインスタンスでスタック AWS CloudFormation を作成する場合、テンプレートは Amazon Linux バージョンを明示的に指定する必要があります。特に、テンプレートで Amazon Linux を指定している場合、インスタンスでは引き続きバージョン 2016.09 が実行されます。詳細については、[AWS::OpsWorks::Stack](#)「」および「」を参照してください [AWS::OpsWorks::Instance](#)。

インスタンスの Amazon Linux のバージョンを更新するには、次のいずれかを実行します。

- オンラインインスタンスの場合、[Upgrade Operating System スタックコマンド](#)を実行します。

新しいバージョンの Amazon Linux が利用可能になると、[Instances] ページと [Stack] ページに通知が表示されます。この通知内のリンクをクリックすると、[Run Command] ページが表示されます。ここで [Upgrade Operating System] を実行して、インスタンスをアップグレードできます。

- オフラインの Amazon Elastic Block Store-backed (EBS-backed) インスタンスの場合、インスタンスを起動して、前の項目で説明した方法でオペレーティングシステムのアップグレードを実行します。

- 時間ベースのインスタンスと負荷ベースのインスタンスを含め、オフラインの Instance Store-Backed インスタンスの場合は、[インスタンスの \[Operating system\] 設定を編集して新しいバージョンを指定してください](#)。

AWS OpsWorks スタックは、インスタンスを再起動すると、インスタンスを新しいバージョンに自動的に更新します。

Amazon Linux: サポートされている Node.js バージョン

Amazon Linux バージョン	Node.js バージョン
2	(Not applicable to operating systems that are available for Chef 12 and higher stacks only)
2018.03	0.12.18
2017.09	0.12.18
2017.03	0.12.18
2016.09	0.12.18 0.12.17 0.12.16 0.12.15
2016.03	0.12.18 0.12.17 0.12.16 0.12.15 0.12.14 0.12.13 0.12.12 0.12.10

Amazon Linux: サポートされている Chef のバージョン

Chef のバージョン	サポートされている Amazon Linux のバージョン
12	Amazon Linux 2 Amazon Linux 2018.03 Amazon Linux 2017.09 Amazon Linux 2017.03 Amazon Linux 2016.09 Amazon Linux 2016.03
11.10	Amazon Linux 2018.03 Amazon Linux 2017.09 Amazon Linux 2017.03 Amazon Linux 2016.09 Amazon Linux 2016.03
11.4 (deprecated)	Amazon Linux 2016.09 Amazon Linux 2016.03

⚠ Important

t1.micro インスタンスを更新する前に、各インスタンスに一時スワップファイル `/var/swapfile` があることを確認してください。Chef 0.9 スタックの t1.micro インスタンスにはスワップファイルがありません。Chef 11.4 および Chef 11.10 スタックでは、最近のバージョンのインスタンスエージェントによって、t1.micro インスタンスのスワップファイルが自動的に作成されます。ただし、この変更が導入されたのは数週間の期間であったため、2014 年 3 月 24 日頃より前に作成されたインスタンスに `/var/swapfile` が存在するかどうかを確認する必要があります。

スワップファイルがない t1.micro インスタンスでは、次の方法でスワップファイルを作成できます。

- Chef 11.10 以降のスタックの場合、新しい t1.micro インスタンスを作成すると、スワップファイルが自動的に作成されます。
- Chef 0.9 スタックの場合、各インスタンスで root ユーザーとして次のコマンドを実行します。

```
dd if=/dev/zero of=/var/swapfile bs=1M count=256
mkswap /var/swapfile
chown root:root /var/swapfile
chmod 0600 /var/swapfile
swapon /var/swapfile
```

新しいインスタンスを作成しない場合、これらのコマンドは Chef 11.10 以降のスタックでも使用できます。

Ubuntu LTS

Ubuntu では、約 2 年ごとに新しい Ubuntu LTS バージョンがリリースされ、各リリースは約 5 年間サポートされます。オペレーティングシステムのサポート期間中は、セキュリティパッチと更新が提供されます。詳細については、[Ubuntu Wiki の LTS に関するページ](#)を参照してください。

- 既存の Ubuntu インスタンスを以降の Ubuntu に更新することはできません。

[新しい Ubuntu インスタンスを作成し](#)、[古いインスタンスを削除](#)する必要があります。

- Ubuntu 20.04 LTS は、Chef 12 以降のスタックでのみサポートされます。

CentOS

AWS OpsWorks スタックは [CentOS 7](#) の 64 ビットバージョンをサポートしています。最初にサポートされているバージョンは CentOS 7 で、CentOS は約 2 年ごとに新しいバージョンをリリースします。

CentOS スタックで新しいインスタンスを起動すると、AWS OpsWorks スタックは最新の CentOS バージョンを自動的にインストールします。AWS OpsWorks スタックは、新しい CentOS マイナーバージョンがリリースされたときに既存のインスタンスのオペレーティングシステムを自動的に更新しないため、新しく作成されたインスタンスは、スタックの既存のインスタンスよりも新しいバージョンを受け取る可能性があります。以下の方法で既存のインスタンスを現在の CentOS バージョンに更新することで、スタック間のバージョンの整合性を維持することができます。

- オンラインインスタンスの場合は、[\[Upgrade Operating System\] スタックコマンド](#)を yum update 指定したインスタンスで実行して、オペレーティングシステムを現在のバージョンに更新します。

新しい CentOS 7 のマイナーバージョンが利用可能になると、[Instances] ページと [Stack] ページに通知が表示されます。この通知内のリンクをクリックすると、[Run Command] ページが表示されます。ここで [Upgrade Operating System] を実行して、インスタンスをアップグレードできます。

- オフラインの更新された Amazon EBS インスタンスの場合は、インスタンスを起動し、前のリスト項目で説明したようにオペレーティングシステムのアップグレードを実行します。
- オフラインの instance store-backed インスタンスの場合、インスタンスが再起動されると、AWS OpsWorks スタックは自動的に新しいバージョンをインストールします。

CentOS: サポートされている Chef のバージョン

Chef のバージョン	サポートされている CentOS のバージョン
12	CentOS 7
11.10	(None supported)
11.4 (deprecated)	(None supported)

Note

AWS OpsWorks スタックは CentOS インスタンスの Apache 2.4 をサポートしています。

Red Hat Enterprise Linux

AWS OpsWorks スタックは、[Red Hat Enterprise Linux 7](#) (RHEL 7) の 64 ビットバージョンをサポートしています。最初にサポートされているバージョンは RHEL 7.1 で、Red Hat は約 9 か月ごとにマイナーバージョンをリリースします。マイナーバージョンは RHEL 7.0 と互換性があります。詳細については、「[ライフサイクルと更新ポリシー](#)」を参照してください。

新しいインスタンスを起動すると、AWS OpsWorks スタックは現在の RHEL 7 バージョンを自動的にインストールします。AWS OpsWorks スタックは、新しい RHEL 7 マイナーバージョンがリリースされたときに既存のインスタンスのオペレーティングシステムを自動的に更新しないため、新しく

作成されたインスタンスは、スタックの既存のインスタンスよりも新しいバージョンを受け取る可能性があります。以下の方法で既存のインスタンスを現在の RHEL 7 バージョンに更新することで、スタック間のバージョンの整合性を維持することができます。

- オンラインインスタンスの場合は、[\[Upgrade Operating System\] スタックコマンド](#)をyum update指定したインスタンスで実行して、オペレーティングシステムを現在のバージョンに更新します。

新しいバージョンの RHEL 7 が利用可能になると、[Instances] ページと [Stack] ページに通知が表示されます。この通知内のリンクをクリックすると、[Run Command] ページが表示されます。ここで [Upgrade Operating System] を実行して、インスタンスをアップグレードできます。

- オフラインの更新された Amazon EBS インスタンスの場合は、インスタンスを起動し、前のリスト項目で説明したようにオペレーティングシステムのアップグレードを実行します。
- オフラインの instance store-backed インスタンスの場合、インスタンスが再起動されると、AWS OpsWorks スタックは自動的に新しいバージョンをインストールします。

Red Hat Enterprise Linux: サポートされている Node.js のバージョン

RHEL バージョン	Node.js バージョン
7	(Node.js versions only apply to Chef 11.10 stacks) 0.8.19 0.8.26 0.10.11 0.10.21 0.10.24 0.10.25 0.10.27 0.10.29 0.10.40 0.12.10 0.12.12 0.12.13 0.12.15

Red Hat Enterprise Linux: サポートされている Chef のバージョン

Chef のバージョン	サポートされている RHEL バージョン
12	Red Hat Enterprise Linux 7
11.10	Red Hat Enterprise Linux 7
11.4 (deprecated)	(None supported)

0.10.40 よりも古いバージョンの Node.js はすべて非推奨です。また、0.12.7 と 0.12.9 も非推奨となっています。

Note

AWS OpsWorks スタックは、RHEL 7 インスタンスの Apache 2.4 をサポートしています。

Microsoft Windows Server

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行に関するご質問は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

以下の注意事項では、Windows インスタンスの AWS OpsWorks スタックサポートについて説明します。Windows インスタンスは、Chef 12.2 スタックでのみ使用できます。Windows スタックの正確な Chef バージョン番号は 12.22 です。

現在、AWS OpsWorks スタックエージェントは、英語 - 米国 (en-US) 以外のシステム UI 言語を使用する Windows ベースのインスタンスにはインストールできず、AWS OpsWorks スタックは管理できません。

バージョン

AWS OpsWorks スタックは、次の Windows 64 ビットバージョンをサポートしています。

- Microsoft Windows Server 2022 Base
- Microsoft Windows Server 2022 および SQL Server Express
- Microsoft Windows Server 2022 および SQL Server Standard
- Microsoft Windows Server 2022 および SQL Server Web
- Microsoft Windows Server 2019 Base
- Microsoft Windows Server 2019 および SQL Server Express
- Microsoft Windows Server 2019 および SQL Server Standard
- Microsoft Windows Server 2019 および SQL Server Web

インスタンスの作成

AWS OpsWorks スタックコンソール、API、または CLI を使用して Windows インスタンスを作成します。Windows インスタンスは Amazon EBS でバックアップされていますが、余分な Amazon EBS ボリュームをマウントすることはできません。

Windows スタックでは、手動での起動と停止が可能な [24/7](#) インスタンスを使用できます。また、[時間ベースの自動スケーリング](#)を使用することもできます。これは、ユーザーが指定したスケジュールでインスタンスを自動的に起動および停止する機能です。Windows ベースのスタックでは、[負荷ベースの自動スケーリング](#)を使用することはできません。

AWS OpsWorks スタックの外部で作成された [Windows インスタンスをスタックに登録](#)することはできません。

更新

AWS は、パッチセットごとに Windows AMI を更新するため、インスタンスを作成すると最新の更新が適用されます。ただし、AWS OpsWorks スタックはオンライン Windows インスタンスに更新を適用する方法を提供しません。Windows を最新の状態に保つには、最新の AMI が常に実行されるようにインスタンスを定期的に置き換えるのが最も簡単です。

レイヤー

ソフトウェアのインストールと設定やアプリケーションのデプロイなどのタスクを処理するには、カスタムレシピを使用して 1 つ以上の [カスタムレイヤー](#) を実装する必要があります。

Chef

Windows インスタンスは Chef 12.22 を使用して [chef-client をローカルモード](#) で実行します。これにより、[chef-zero](#) と呼ばれるローカルインメモリの Chef サーバーが起動されます。このサーバーが存在する場合、カスタムレシピで Chef 検索およびデータバッグを使用できます。

リモートログイン

AWS OpsWorks スタックは、Windows インスタンスへのログインに使用できるパスワードを認可された IAM ユーザーに提供します。このパスワードは、指定した時間が経過すると期限切れになります。管理者は、SSH キーペアを使用してインスタンスの管理者パスワードを取得し、[RDP に無制限にアクセス](#) することができます。詳細については、「[RDP でのログイン](#)」を参照してください。

AWS SDK

AWS OpsWorks スタックは、各インスタンス [AWS SDK for .NET](#) に を自動的にインストールします。このパッケージには、AWS Tools for を含む AWS .NET ライブラリと [AWS Tools for Windows PowerShell](#) が含まれています。Ruby SDK を使用する場合、カスタムレシピによって適切な gem をインストールすることができます。

モニタリングおよびメトリクス

Windows インスタンスは、CloudWatch コンソールで表示できる標準の [Amazon CloudWatch \(CloudWatch\) メトリクス](#) をサポートしています。

Ruby

AWS OpsWorks スタックが Windows インスタンスにインストールする Chef 12.22 クライアントには、Ruby 2.3.6 が付属しています。ただし、AWS OpsWorks スタックは実行可能ファイルのディレクトリを PATH 環境変数に追加しません。通常であれば C:\opscode\chef\embedded\bin\ で検索し、アプリケーションがこの Ruby バージョンを使用できるようにします。

AWS OpsWorks スタックエージェント CLI

Windows インスタンスの AWS OpsWorks スタックエージェントは、[コマンドラインインターフェイス](#) を公開しません。

プロキシのサポート

Windows インスタンス用のプロキシサポートをセットアップするには、次の作業を行います。

1. を変更machine.configして以下を追加します。これにより、Windows PowerShell (初期ブートストラップ) および .NET (AWS OpsWorks スタックエージェント) アプリケーションにプロキシサポートが追加されます。

```
<system.net>
  <defaultProxy>
    <proxy autoDetect="false" bypassonlocal="true"
    proxyaddress="http://10.100.1.91:3128" usesystemdefault="false" />
    <bypasslist>
      <add address="localhost" />
      <add address="169.254.169.254" />
    </bypasslist>
  </defaultProxy>
</system.net>
```

2. Chef と Git で後で使用するために、次のコマンドを実行して環境変数を設定します。

```
setx /m no_proxy "localhost,169.254.169.254"
setx /m http_proxy "http://10.100.1.91:3128"
setx /m https_proxy "http://10.100.1.91:3128"
```

Note

AWS OpsWorks スタックがインスタンスを更新する方法をより詳細に制御するには、Microsoft Windows Server 2022 Base に基づいてカスタム AMI を作成します。たとえば、カスタム AMI を使用してインスタンスにインストールするソフトウェア (Web Server (IIS) など) を指定できます。詳細については、「[カスタム AMI の使用](#)」を参照してください。

レイヤーへのインスタンスの追加

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

レイヤーを作成した後、通常は最低 1 個のインスタンスを追加します。現在の設定が負荷を処理できない場合、より多くのインスタンスを後で追加できます。自動的にインスタンス数を拡張するために、[ロードベースまたは時間ベースのインスタンス](#)を使用することもできます。

レイヤーに新規または既存のインスタンスを追加できます。

- 新規 - 仕様に合わせて設定された新しいインスタンスOpsWorks を作成し、レイヤーのメンバーにします。
- 既存 - 互換性のある任意のレイヤーの既存のインスタンスを追加できます。ただし、オフライン (停止)状態である必要があります。

インスタンスが複数のレイヤーに属している場合、ライフサイクルイベントが発生したときや、[スタック](#)や[デプロイメント](#)コマンドを実行したときに、AWS OpsWorks Stacks はインスタンスの各レイヤーのレシピを実行します。

また、インスタンスの設定を編集することによって、インスタンスを複数のレイヤーのメンバーにすることができます。詳細については、「[インスタンス設定の編集](#)」を参照してください。

レイヤーに新しいインスタンスを追加するには

1. [Instances] ページで適切なレイヤーの [+Instance] を選択し、必要であれば [New] タブを選択します。[ホスト名]、[サイズ]、[サブネット]、[アベイラビリティアベイラビリティゾーン] 以外の設定も行う場合は、[Advanced >>] を選択してさらにオプションを表示します。以下にすべてのオプションを示します。

The screenshot shows the 'Add Instance' configuration page in the AWS OpsWorks console. It features three tabs: 'New', 'Existing OpsWorks', and 'EC2 instances and own servers'. The 'New' tab is active. The configuration fields are as follows:

- Hostname: rails-app1
- Size: c3.large
- Subnet: - us-west-2c
- Scaling type: 24/7, Time-based, Load-based
- SSH key: Do not set an SSH key
- Operating system: Amazon Linux 2015.09
- OpsWorks Agent version: Inherit from stack
- Tenancy: Default - Rely on VPC settings
- Root device type: EBS backed, Instance store
- Volume type: Magnetic
- Volume size: 8 (Min: 8 GiB, Max: 1024 GiB)

Buttons for 'Cancel' and 'Add Instance' are located at the bottom right of the form.

- 必要に応じて、多くはスタックの作成時に指定したデフォルト設定をオーバーライドできます。詳細については、「[新しいスタックを作成する](#)」を参照してください。

ホスト名


ネットワークのインスタンスを識別します。デフォルトでは、AWS OpsWorks スタックは、スタックの作成時に指定したホスト名テーマを使用して各インスタンスのホスト名を生成します。この値をオーバーライドして、任意のホスト名を指定できます。

サイズ

メモリの量や仮想コアの数など、インスタンスのリソースを指定する Amazon EC2 インスタンスタイプ。AWS OpsWorks スタックは、各インスタンスのデフォルトサイズを指定します。このサイズは、任意のインスタンスタイプで上書きできます。

AWS OpsWorks スタックでサポートされているインスタンスタイプは、スタックが VPC 内にあるかどうかによって異なります。また、AWS 無料利用枠を使用しているアカウントでは、インスタンスタイプは制限されています。[Size (サイズ)] ドロップダウンリストには、

使用しているスタックでサポートされている Chef バージョンに対してサポートされている インスタンスタイプが表示されます。t1.micro などのマイクロインスタンスには、いくつかのレイヤーをサポートするだけの十分なリソースがないことがあることに注意してください。詳細については、「[のインスタンスタイプ](#)」を参照してください。

 Note

[負荷分散されたインスタンス](#)を使用している場合、[Configure ライフサイクルイベント](#)によって、1分以上続くことがある大きな CPU 負荷のスパイクが発生する可能性があることに注意してください。より小規模のインスタンスでは、この負荷のスパイクにより上限がトリガーされる可能性があります (特に、頻繁な Configure イベントがある大規模な負荷分散スタックの場合)。以下に、Configure イベントが不要な上限を発生させる可能性を減らすことができるいくつかの方法を示します。

- より大きなインスタンスを使用し、Configure イベントからの追加の負荷が上限をトリガーしないようにします。
- CPU リソースを共有する T2 などのインスタンスタイプを使用しないでください。

これにより、Configure イベントが発生した場合は、すべてのインスタンスの CPU リソースすべてが直ちに使用できるようになります。

- exceeded threshold 時間を、Configure イベントの処理に必要な時間よりもかなり長くします (5 分など)。

詳細については、「[自動負荷ベースのスケールリングの使用](#)」を参照してください。

アベイラビリティゾーン/サブネット

スタックが VPC 内でない場合、この設定は [Availability Zone] と表示され、リージョンのゾーンがリストされます。スタックを作成したときに指定したデフォルトのアベイラビリティゾーンを上書きするには、この設定を使用できます。

スタックが VPC で実行されている場合、この設定は、[Subnet] と表示され、VPC のサブネットがリストされます。スタックを作成したときに指定したデフォルトのサブネットを上書きするには、この設定を使用できます。

Note

デフォルトでは、AWS OpsWorks スタックはサブネットの CIDR 範囲を一覧表示します。リストをより読みやすくするには、VPC コンソールまたは API を使用して、キーを `cidr` に設定し、値をサブネットの名前に設定します。AWS OpsWorks スタックはその名前を CIDR 範囲に追加します。前の例では、サブネットの Name タグは `Private` に設定されています。

スケーリングタイプ

インスタンスの起動方法と停止方法を決定します。

- デフォルト値は 24/7 インスタンスで、手動で開始および停止します。
- AWS OpsWorks スタックは、指定されたスケジュールに基づいて時間ベースのインスタンスを起動および停止します。
- (Linux のみ) AWS OpsWorks スタックは、指定されたロードメトリクスに基づいてロードベースのインスタンスを起動および停止します。

Note

負荷ベースまたは時間ベースのインスタンスを手動で開始または停止することはできません。代わりに、インスタンスを設定すると、AWS OpsWorks スタックは設定に基づいてインスタンスを起動および停止します。詳細については、「[時間ベースおよび負荷ベースのインスタンスによる負荷の管理](#)」を参照してください。

SSH キー

Amazon EC2 キーペア。AWS OpsWorks スタックは、パブリックキーをインスタンスにインストールします。

- Linux インスタンスの場合、SSH クライアントで対応するプライベートキーを使用して [インスタンスにログイン](#) できます。
- Windows インスタンスの場合、対応するプライベートキーを使用して [インスタンスの管理者パスワードを取得](#) できます。その後、RDP でそのパスワードを使用してインスタンスに管理者としてログインできます。

この設定は、スタックの作成時に指定した [Default SSH key] 値に初期設定されます。

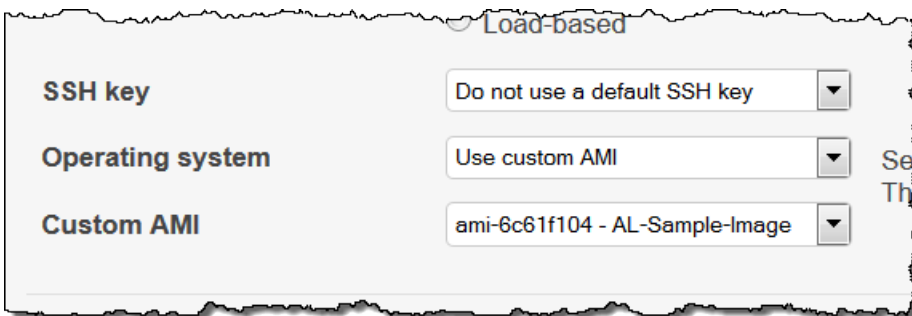
- デフォルト値が [デフォルトのSSHキーを使用しない] に設定されている場合は、自分のアカウントの Amazon EC2 キーのいずれかを指定できます。
- デフォルト値が Amazon EC2 キー に設定されている場合は、別のキーを指定することも、キーを指定しないこともできます。

オペレーティングシステム

オペレーティングシステムは、インスタンスが実行されているオペレーティングシステムを指定します。AWS OpsWorks スタックは 64 ビットオペレーティングシステムのみをサポートします。

この設定は、スタックの作成時に指定した [デフォルトのオペレーティングシステム] 値に初期設定されます。デフォルト値を上書きして、別の Linux オペレーティングシステムまたはカスタム Amazon マシンイメージ (AMI) を指定できます。ただし、Linux から Windows にまたは Windows から Linux に切り替えることはできません。

[Use custom AMI] を選択すると、ページには [Architecture] および [Root device type] ではなくカスタム AMI のリストが表示されます。



Load-based

SSH key Do not use a default SSH key

Operating system Use custom AMI

Custom AMI ami-6c61f104 - AL-Sample-Image

Settings

詳細については、「[カスタム AMI の使用](#)」を参照してください。

OpsWorks エージェントバージョン

OpsWorks エージェントバージョンは、インスタンスで実行する AWS OpsWorks Stacks エージェントのバージョンを指定します。AWS OpsWorks スタックがエージェントを自動的に更新するようにするには、[Inherit from stack (スタックから継承する)] をオンにします。エージェントの特定のバージョンをインストールし、インスタンスでエージェントを手動更新するには、ドロップダウンリストからバージョンを選択します。

Note

エージェントのバージョンとオペレーティング システムのリリースの組み合わせによってはうまく機能しない場合があります。インスタンスがエージェントを実行している場合、またはインスタンスオペレーティングシステムで完全にサポートされていないエージェントをインスタンスにインストールしている場合、AWS OpsWorks スタックコンソールには、互換性のあるエージェントをインストールするように指示するエラーメッセージが表示されます。

テナンシー

インスタンスのテナンシーオプションを選択します。お客様専用の物理サーバー上でインスタンスを実行することも可能です。

- Default - Rely on VPC settings: テナンシーなし、または VPC からテナンシー設定を継承します。
- Dedicated - Run a dedicated instance: シングルテナントハードウェアで実行されるインスタンスに対して、時間単位でお支払いいただきます。詳細については、Amazon VPC ユーザーガイドの「[ハードウェア専有インスタンス](#)」、および「[Amazon EC2 ハードウェア専有インスタンス](#)」を参照してください。
- Dedicated host - Run this instance on a dedicated host: 完全にインスタンスの実行専用の物理ホストに対してお支払いいただき、既存のソケット単位、コア単位、または VM 単位のソフトウェアライセンスを持ち込んでコストを削減できます。詳細については、Amazon EC2 ドキュメントの[専有ホストの概要](#)および[Amazon EC2 専有ホスト](#)を参照してください。

ルートデバイスタイプ

インスタンスのルートデバイスストレージを指定します。

- Linux インスタンスは、Amazon EBS-backed または Instance store-backed のどちらかです。
- Windows インスタンスは、Amazon EBS-Backed である必要があります。

詳細については、「[ストレージ](#)」を参照してください。

Note

AWS OpsWorks スタックはインスタンスのソフトウェアをゼロから再インストールする必要がないため、最初の起動後、Amazon EBS-backed インスタンスは instance store-backed インスタンスよりも速く起動します。詳細については、「[ルートデバイスストレージ](#)」を参照してください。

ボリュームタイプ

ルートデバイスボリュームタイプ ([Magnetic]、[Provisioned IOPS (SSD)] または [General Purpose (SSD)]) を指定します。詳細については、「[Amazon EBS ボリュームのタイプ](#)」を参照してください。

ボリュームサイズ

指定したボリュームタイプのルートデバイスボリュームのサイズを指定します。詳細については、「[Amazon EBS ボリュームのタイプ](#)」を参照してください。

- 汎用 (SSD)。許容される最小サイズは 8 GiB で、最大サイズは 16384 GiB です。
- プロビジョンド IOPS (SSD)。許容される最小サイズは 8 GiB で、最大サイズは 16384 GiB です。設定できる 1 秒あたりの入力/出力操作数は、最小 100 IOPS、最大 240 IOPS です。
- マグネティック。許容される最小サイズは 8 GiB で、最大サイズは 1024 GiB です。

3. [Add Instance] を選択して、新しいインスタンスを作成します。

Note

インスタンスを作成するとき、[スタックのデフォルトエージェントバージョン](#)の設定を上書きすることはできません。カスタムエージェントバージョンの設定を指定するには、インスタンスを作成し、[その設定を編集](#)する必要があります。

既存のインスタンスをレイヤーに追加するには

1. [Instances] ページで適切なレイヤーの [+Instance] を選択し、[Existing] タブを開きます。

Note

既存のインスタンスを使用しないことにした場合は、前述の手順に従って、[New] を選択して、新しいインスタンスを作成します。

2. [Existing] タブで、リストからインスタンスを選択します。
3. [Add Instance] を選択して、新しいインスタンスを作成します。

インスタンスは Amazon EC2 インスタンスを表しますが、基本的には AWS OpsWorks スタックのデータ構造にすぎません。次のセクションで説明するように、実行中の Amazon EC2 インスタンスを作成するには、インスタンスが開始されている必要があります。

Important

デフォルト VPC 内でインスタンスを起動する場合は、VPC 設定の変更について注意する必要があります。インスタンスは、AWS OpsWorks スタックサービス、Amazon S3、およびパッケージリポジトリと常に通信する必要があります。例えば、デフォルトゲートウェイを削除すると、インスタンスは AWS OpsWorks スタックサービスへの接続を失い、インスタンスは失敗として扱われ、[自動修復](#)されます。ただし、AWS OpsWorks スタックは修復されたインスタンスにインスタンスエージェントをインストールできません。エージェントがない場合、インスタンスはサービスと通信できず、起動プロセスは booting ステータスの先に進みません。デフォルト VPC の詳細については、「[サポートされているプラットフォーム](#)」を参照してください。

AWS OpsWorks スタックの外部で作成されたスタックに Linux コンピューティングリソースを組み込むこともできます。

- コンソール、CLI、または API を使用して直接作成した Amazon EC2 インスタンス。
- 仮想マシンで実行しているインスタンスを含め、独自のハードウェアで実行している On-premises (オンプレミス) インスタンス。

詳細については、「[AWS OpsWorks Stacks の外部で作成されたコンピューティングリソースの使用](#)」を参照してください。

カスタム AMI の使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは、カスタム [Amazon マシンイメージ \(AMIs\)](#) の方法でインスタンスをカスタマイズできます。どちらのアプローチでも、インストールするパッケージおよびパッケージバージョンの種類や設定方法などをコントロールできます。ただし、それぞれの利点は異なるため、どちらが最適であるかは要件によって変わります。

カスタム AMI の使用を検討する必要がある主な状況は、次のとおりです。

- インスタンスの起動後に特定のパッケージをインストールするのではなく、特定のパッケージを事前にバンドルする場合。
- レイヤーに一貫したベースイメージを提供するようにパッケージ更新のタイミングを管理する場合。
- インスタンス (特に [負荷ベース](#) のインスタンス) を迅速に起動する場合。

Chef レシピの使用を検討すべき主な場合は、次のとおりです。

- カスタム AMI より柔軟性が高い場合。
- 更新が容易な場合。
- 実行中のインスタンスで更新を実行できる場合。

現実的には、両方の手法を組み合わせることが最適なソリューションと考えることもできます。recipe の詳細については、「[クックブックとレシピ](#)」を参照してください。

トピック

- [カスタム AMIs と AWS OpsWorks スタックの連携方法](#)
- [AWS OpsWorks スタック用のカスタム AMI の作成](#)

カスタム AMIs と AWS OpsWorks スタックの連携方法

インスタンスにカスタム AMI を指定するには、新しいインスタンスを作成するときにインスタンスのオペレーティングシステムとしてカスタム AMI を使用するを選択します。次に、AWS OpsWorks スタックはスタックのリージョンにあるカスタム AMIs のリストを表示し、リストから適切なものを選択します。詳細については、「[レイヤーへのインスタンスの追加](#)」を参照してください。

Note

スタックのデフォルトオペレーティングシステムとして特定のカスタム AMI を指定することはできません。スタックのデフォルトオペレーティングシステムとして Use custom AMI を設定できますが、特定の AMI を指定できるのは、新しいインスタンスをレイヤーに追加するときのみです。詳細については、「[レイヤーへのインスタンスの追加](#)」および「[新しいスタックを作成する](#)」を参照してください。カスタム AMI またはコミュニティで作成された AMI から作成された他のオペレーティングシステム (CentOS 6.x など) を使用してインスタンスを作成できる場合もありますが、そのような方法は公式にはサポートされていません。

このトピックでは、カスタム AMI を作成または使用する前に考慮する必要がある一般的な問題について説明します。

トピック

- [開始時の動作](#)
- [レイヤーの選択](#)
- [アプリケーションの処理](#)

開始時の動作

インスタンスを起動すると、AWS OpsWorks スタックは指定されたカスタム AMI を使用して新しい Amazon EC2 インスタンスを起動します。AWS OpsWorks スタックは [cloud-init](#) を使用してインスタンスに AWS OpsWorks スタックエージェントをインストールし、エージェントはインスタンスの Setup レシピと Deploy レシピを実行します。インスタンスがオンラインになると、エージェントは新しく追加されたインスタンスを含め、スタックのすべてのインスタンスに対して Configure レシピを実行します。

レイヤーの選択

AWS OpsWorks スタックエージェントは通常、インストールされているパッケージと競合しません。ただし、インスタンスは少なくとも1つのレイヤーのメンバーである必要があります。AWS OpsWorks スタックは常にそのレイヤーのレシピを実行するため、問題が発生する可能性があります。カスタム AMI を持つインスタンスをそのレイヤーに追加する前に、レイヤーのレシピによるインスタンスへの操作について正確に理解する必要があります。

インスタンスでレイヤータイプ別に実行されるレシピを確認するには、そのレイヤーを含むスタックを開きます。次に、ナビゲーションペインの [Layers] (レイヤー) をクリックし、目的のレイヤーの [Recipes] (レシピ) をクリックします。実際のコードを表示するには、レシピの名前をクリックします。

Note

Linux AMIs、競合の可能性を減らす方法の1つは、AWS OpsWorks スタックを使用して、カスタム AMI の基礎となるインスタンスをプロビジョニングおよび設定することです。詳細については、「[AWS OpsWorks スタックインスタンスからカスタム Linux AMI を作成する](#)」を参照してください。

アプリケーションの処理

パッケージに加えて、AMI にアプリケーションを含めたい場合もあります。大規模で複雑なアプリケーションがある場合、AMI に含めると、インスタンスの起動時間を短縮できます。AMI に小さなアプリケーションを含めることはできますが、通常、AWS OpsWorks スタックによるアプリケーションのデプロイに比べて、時間上の利点はほとんどまたはまったくありません。

1つのオプションは、アプリケーションを AMI に含めると共に、リポジトリからインスタンスにアプリケーションをデプロイする[アプリケーションを作成](#)することです。このアプローチでは、起動時間が短縮されるだけでなく、インスタンスの実行後にアプリケーションを更新する際に便利な方法を使用できます。Chef レシピはべき等であるため、リポジトリのバージョンがインスタンスのバージョンと同じである限り、デプロイレシピはアプリケーションを変更しません。

AWS OpsWorks スタック用のカスタム AMI の作成

AWS OpsWorks スタックでカスタム AMI を使用するには、まずカスタマイズされたインスタンスから AMI を作成する必要があります。2つのオプションから選択できます。

- Amazon EC2 コンソールまたは API を使用して、[AWS OpsWorks スタック対応 AMI](#) のいずれかの 64 ビット版に基づくインスタンスを作成およびカスタマイズします。
- Linux AMIs OpsWorks を使用して、関連するレイヤーの設定に基づいて Amazon EC2 インスタンスを作成します。

カスタム Linux AMI を作成する前に、`/tmp`パーティション`noexec`で を無効にして、AWS OpsWorks スタックがカスタム Linux インスタンスにエージェントをインストールできるようにします。

Note

AMI はすべてのインスタンスタイプに対応している訳ではないことにご注意ください。開始する AMI が、使用するインスタンスタイプと互換性があることを確認してください。特に、[R3](#) インスタンスタイプには、ハードウェアアシストによる仮想化 (HVM) AMI が必要です。

次に Amazon EC2 コンソールまたは API を使用して、カスタマイズしたインスタンスからカスタム AMI を作成します。インスタンスをレイヤーに追加し、カスタム AMI を指定することで、同じリージョンのすべてのスタックでカスタム AMI を使用できます。カスタム AMI を使用するインスタンスの作成方法の詳細については、「[レイヤーへのインスタンスの追加](#)」を参照してください。

Note

デフォルトでは、AWS OpsWorks スタックは起動時にすべての Amazon Linux 更新プログラムをインストールし、最新のリリースを提供します。また Amazon Linux は新しいバージョンを約 6 か月ごとにリリースしており、大きな変更が実施される場合もあります。デフォルトで、Amazon Linux に基づくカスタム AMI は、新しいバージョンがリリースされると自動的に更新されます。カスタム AMI は特定の Amazon Linux バージョンにロックしておき、新しいバージョンをテストするまで更新を延期できるようにすることをお勧めします。詳細については、「[AMI を特定のバージョンに固定するにはどうすればよいですか?](#)」を参照してください。

トピック

- [Amazon EC2 を使用したカスタム AMI の作成](#)
- [AWS OpsWorks スタックインスタンスからカスタム Linux AMI を作成する](#)

• [カスタム Windows AMI の作成](#)

Amazon EC2 を使用したカスタム AMI の作成

カスタム AMI を作成する最も簡単な方法 (Windows AMI の唯一のオプション) は、Amazon EC2 コンソールまたは API を使用してタスク全体を実行することです。次のステップの詳細については、[\[独自の AMI の作成\]](#) を参照してください。

Amazon EC2 コンソールまたは API を使用してカスタム AMI を作成するには

1. [いずれかの AWS OpsWorks スタック対応 AMI](#) の 64 ビット版を使用してインスタンスを作成します。
2. インスタンスを設定し、パッケージをインストールするなどして、ステップ 1 からインスタンスをカスタマイズします。インストールしたもののすべてが、AMI に基づいてすべてのインスタンス上で再現されるため、特定のインスタンスに固有の項目は含めないでください。
3. インスタンスを停止し、カスタム AMI を作成します。

AWS OpsWorks スタックインスタンスからカスタム Linux AMI を作成する


カスタマイズされた AWS OpsWorks Stacks Linux インスタンスを使用して AMI を作成するには、によって作成されたすべての Amazon EC2 インスタンスに一意的 ID OpsWorks が含まれていることに注意してください。このようなインスタンスからカスタム AMI を作成した場合、その ID が含まれ、AMI に基づくすべてのインスタンスが同じ ID を持つことになります。カスタム AMI に基づくインスタンスが確実に固有の ID を得るには、AMI を作成する前に、カスタマイズしたインスタンスから ID を削除する必要があります。

AWS OpsWorks スタックインスタンスからカスタム AMI を作成するには

1. [Linux スタックを作成し、1 つ以上のレイヤーを追加](#)して、カスタマイズしたインスタンスの設定を定義します。組み込みレイヤー、完全なカスタムレイヤーに加え、必要に応じてカスタマイズしたレイヤーを使用できます。詳細については、「[AWS OpsWorks スタックのカスタマイズ](#)」を参照してください。
2. [レイヤーを編集し、を無効にします](#) AutoHealing。
3. [任意の Linux ディストリビューションを使用してインスタンスをレイヤーに追加し、起動します](#)。Amazon EBS-backed インスタンスの使用をお勧めします。インスタンスの詳細ページを開き、後で使用できるよう Amazon EC2 ID を記録します。

4. インスタンスがオンラインである場合は、[SSH でログイン](#)し、インスタンスのオペレーティングシステムに応じて次の 4 つのステップのいずれかを実行します。
5. Chef 11 または Chef 12 スタックの Amazon Linux インスタンス、または Chef 11 スタックの Red Hat Enterprise Linux 7 インスタンスの場合、次を実行します。

- a. `sudo /etc/init.d/monit stop`
- b. `sudo /etc/init.d/opsworks-agent stop`
- c. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/cloud/ /etc/chef`

 Note

Chef 12 スタックのインスタンスについて、このコマンドに次の 2 つのフォルダを追加します。

- `/var/chef`
- `/opt/chef`

- d. `sudo rpm -e opsworks-agent-ruby`
- e. `sudo rpm -e chef`
6. Chef 12 スタックの Ubuntu 16.04 LTS または 18.04 LTS インスタンスでは、次を実行します。
 - a. `sudo systemctl stop opsworks-agent`
 - b. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/cloud/ /var/chef /opt/chef /etc/chef`
 - c. `sudo apt-get -y remove chef`
 - d. `sudo dpkg -r opsworks-agent-ruby`
 - e. `systemctl stop apt-daily.timer`
 - f. `systemctl stop apt-daily-upgrade.timer`
 - g. `rm /var/lib/systemd/timers/stamp-apt-daily.timer`
 - h. `rm /var/lib/systemd/timers/stamp-apt-daily-upgrade.timer`

7. Chef 12 スタックの他のサポートされている Ubuntu バージョンでは、次を実行します。

- a. `sudo /etc/init.d/monit stop`
- b. `sudo /etc/init.d/opsworks-agent stop`
- c. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/
aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-
agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/
cloud/ /var/chef /opt/chef /etc/chef`
- d. `sudo apt-get -y remove chef`
- e. `sudo dpkg -r opsworks-agent-ruby`

8. Chef 12 スタックの Red Hat Enterprise Linux 7 インスタンスの場合は、次を実行します。

- a. `sudo systemctl stop opsworks-agent`
- b. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/
aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-
agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/
cloud/ /etc/chef /var/chef`
- c. `sudo rpm -e opsworks-agent-ruby`
- d. `sudo rpm -e chef`

9. このステップは、インスタンスタイプによって異なります。

- Amazon EBS-Backed インスタンスの場合は、「Amazon EBS-Backed Linux AMI の作成」の説明に従って、AWS OpsWorks スタックコンソールを使用して [インスタンスを停止](#)し、AMI を作成します。 <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-ebs.html>
- instance store-backed インスタンスの場合は、「[Instance Store-Backed Linux AMI の作成](#)」の説明に従って AMI を作成し、AWS OpsWorks スタックコンソールを使用してインスタンスを停止します。

AMI を作成する際は、必ず証明書ファイルを含めます。例えば、`-i` 引数を `-i $(find /etc /usr /opt -name '*.pem' -o -name '*.crt' -o -name '*.gpg' | tr '\n' ',')` に設定して [ec2-bundle-vol](#) コマンドを呼び出すことができます。バンドルするとき、apt パブリックキーを削除しないでください。デフォルトの `ec2-bundle-vol` コマンドがこのタスクを処理します。

10. スタックをクリーンアップするには、AWS OpsWorks スタックコンソールに戻り、スタックから [インスタンスを削除します](#)。

カスタム Windows AMI の作成

次の手順では、Windows Server 2022 Base 用のカスタム AMI を作成します。Amazon EC2 管理コンソールで、他の Windows Server オペレーティングシステムを選択できます。

Important

現在、AWS OpsWorks スタックエージェントは、英語 - 米国 (en-US) 以外のシステム UI 言語を使用する Windows ベースのインスタンスにはインストールできず、AWS OpsWorks スタックは管理できません。

トピック

- [Sysprep を使用したカスタム Windows AMI の作成](#)
- [Sysprep を使用しないカスタム Windows AMI の作成](#)
- [カスタム Windows AMI を使用した新しいインスタンスの追加](#)

Sysprep を使用したカスタム Windows AMI の作成

通常、Sysprep を使用してカスタム Windows AMI を作成するとインスタンスの起動が遅くなりますが、よりクリーンなプロセスとなります。で作成されたイメージから作成されたインスタンスの初回起動には、セットアップや設定など、Sysprepアクティビティ、再起動、AWS OpsWorks スタックのプロビジョニング、および最初の AWS OpsWorks スタックの実行により、さらに時間が Sysprep かかります。Amazon EC2 コンソールで、カスタム Windows AMI を作成するためのステップを完了します。

Sysprep でカスタム Windows AMI を作成するには

1. Amazon EC2 コンソールで、[Launch Instance (インスタンスを起動する)] を選択します。
2. [Microsoft Windows Server 2022 Base] を見つけて、[選択] を選択します。
3. 目的のインスタンスタイプを選択し、[Configure Instance Details] を選択します。AMI で、マシン名、ストレージ、セキュリティグループ設定などの設定変更を行います。[Launch] (起動する) を選択します。

4. インスタンスのブートプロセスが終了したら、パスワードを使用して、Windows の [リモートデスクトップ接続] ウィンドウでインスタンスに接続します。
5. Windows のスタート画面で の開始 を選択し、EC2 コンソールが表示される `ec2configservice` まで入力を開始します。EC2ConfigServiceSettings コンソールを開きます。
6. 全般 タブで、UserData 実行を有効にする チェックボックスがオンになっていることを確認します (このオプションは 必須ではありませんが Sysprep、AWS OpsWorks スタックがエージェントをインストールするために必要です)。[Set the computer name of the instance...] (インスタンスのコンピュータ名の設定...) オプションのチェックボックスをオフにします。このオプションをオンにすると、AWS OpsWorks スタックで再起動が繰り返されることがあるためです。
7. [Image] (イメージ) タブで、[Administrator Password] (管理者パスワード) を、Amazon EC2 が SSH キーで取得できるパスワードを自動的に生成することを許可する [Random] (ランダム)、または自分のパスワードを指定する [Specify] (指定) のいずれかに設定します。Sysprep はこの設定を保存します。独自のパスワードを指定した場合は、都合の良い場所にパスワードを保存します。[Keep Existing] は選択しないことをお勧めします。
8. [Apply] を選択し、[Shutdown with Sysprep] を選択します。確認を求められたら、[Yes] を選択します。
9. インスタンスが停止したら、Amazon EC2 コンソールで [インスタンス] リストのインスタンスを右クリックし、[イメージ] を選択して、[イメージの作成] を選択します。
10. [Create Image] ページで、イメージの名前と説明を指定し、ボリュームの設定を指定します。終了したら、[Create Image] を選択します。
11. [Images] ページを開き、イメージが [pending] 段階から [available] に変わるのを待ちます。新しい AMI を使用する準備ができました。

Sysprep を使用しないカスタム Windows AMI の作成

Amazon EC2 コンソールで、カスタム Windows AMI を作成するためのステップを完了します。

Sysprep なしでカスタム Windows AMI を作成するには

1. Amazon EC2 コンソールで、[Launch Instance] (インスタンスを起動する) を選択します。
2. [Microsoft Windows Server 2022 Base] を見つけて、[選択] を選択します。

3. 目的のインスタンスタイプを選択し、[Configure Instance Details] を選択します。AMI で、マシン名、ストレージ、セキュリティグループ設定などの設定変更を行います。[Launch] (起動する) を選択します。
4. インスタンスのブートプロセスが終了したら、パスワードを使用して、Windows の [リモートデスクトップ接続] ウィンドウでインスタンスに接続します。
5. インスタンスで、C:\Program Files\Amazon\Ec2ConfigService\Settings\config.xml を開き、次の 2 つの設定を変更してから、ファイルを保存して閉じます。
 - Ec2SetPassword ~ Enabled
 - Ec2HandleUserData ~ Enabled
6. リモートデスクトップセッションからの接続を切り、Amazon EC2 コンソールに戻ります。
7. [Instances] リストで、インスタンスを停止します。
8. インスタンスが停止したら、コンソールで [Instances] (インスタンス) リストのインスタンスを右クリックし、[Image] (イメージ) を選択して、[Create Image] (イメージの作成) を選択します。
9. [Create Image] ページで、イメージの名前と説明を指定し、ボリュームの設定を指定します。終了したら、[Create Image] を選択します。
10. [Images] ページを開き、イメージが [pending] 段階から [available] に変わるのを待ちます。新しい AMI を使用する準備ができました。

カスタム Windows AMI を使用した新しいインスタンスの追加

イメージが [available] 状態に変わったら、カスタム Windows AMI に基づいて新しいインスタンスを作成できます。[Operating system] (オペレーティングシステム) のリストから [Use custom Windows AMI] (カスタム Window AMI を使用する) を選択すると、AWS OpsWorks Stacks はカスタム AMI のリストを表示します。

カスタム Windows AMI に基づいて新しいインスタンスを追加するには

1. 新しい AMI が使用可能になったら、AWS OpsWorks スタックコンソールに移動し、Windows スタックのインスタンスページを開き、ページの下部にある + インスタンスを選択して新しいインスタンスを追加します。
2. [New] タブの [Advanced] を選択します。
3. [Operating system] ドロップダウンリストで、[Use custom Windows AMI] を選択します。

4. [Custom AMI] ドロップダウンリストで、作成した AMI を選択し、[Add Instance] を選択します。

これで、インスタンスを起動して実行できるようになりました。

24/7 インスタンスの手動による起動、停止、再起動

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

Linux スタックと Windows スタックの両方で 24/7 インスタンスを使用できます。

24/7 インスタンスをレイヤーに追加したら、インスタンスを手動で起動して対応する Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを起動し、手動で停止して Amazon EC2 インスタンスを終了する必要があります。正しく機能していないインスタンスを手動で再起動することもできます。AWS OpsWorks スタックは、時間ベースおよび負荷ベースのインスタンスを自動的に開始および停止します。詳細については、「[時間ベースおよび負荷ベースのインスタンスによる負荷の管理](#)」を参照してください。

Important

AWS OpsWorks スタックインスタンスは、コンソールでのみ AWS OpsWorks 起動、停止、再起動する必要があります。Amazon EC2 AWS OpsWorks コンソールで実行された起動、停止、再起動オペレーションは認識されません。

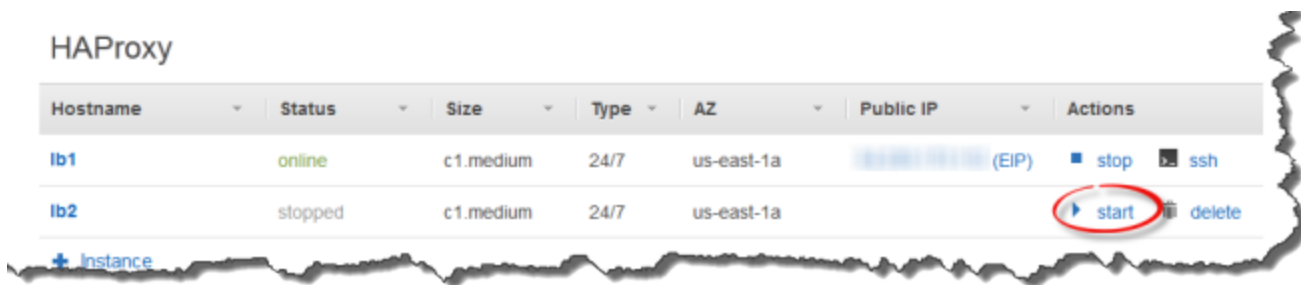
トピック

- [インスタンスの起動または再起動](#)

- [インスタンスの停止](#)
- [インスタンスの再起動](#)

インスタンスの起動または再起動

新しいインスタンスを起動するには、[Instances] (インスタンス) ページでインスタンスの [Actions] (アクション) 列の [start] (開始) をクリックします。



複数のインスタンスを作成し、[Start all Instances] をクリックしてこれらのすべてを同時に起動することもできます。

インスタンスを起動すると、AWS OpsWorks スタックは Amazon EC2 インスタンスを起動し、オペレーティングシステムを起動します。起動プロセスには通常数分かかり、Windows インスタンスの方が Linux インスタンスより少し時間がかかります。起動処理が進行するに従って、インスタンスの [Status] フィールドに次のような一連の値が表示されます。

1. リクエスト - AWS OpsWorks スタックは Amazon EC2 サービスを呼び出して Amazon EC2 インスタンスを作成しました。
2. 保留中 - AWS OpsWorks スタックは Amazon EC2 インスタンスの起動を待っています。
3. [booting] (起動中) - Amazon EC2 インスタンスの起動中です。
4. running_setup - AWS OpsWorks スタックは Setup イベントをトリガーし、レイヤーの Setup レシピを実行し、その後に Deploy レシピを実行します。詳細については、「[レシピの実行](#)」を参照してください。[カスタムクックブック](#)をスタックに追加した場合、AWS OpsWorks スタックは Setup および Deploy レシピを実行する前にリポジトリから最新バージョンをインストールします。
5. online - インスタンスは利用可能です。

[Status] が [online] に変わると、インスタンスは完全に操作可能になります。

- レイヤーにロードバランサーがアタッチされている場合、AWS OpsWorks スタックはインスタンスを追加します。
- AWS OpsWorks スタックは、各インスタンスのConfigureレシピを実行するConfigureイベントをトリガーします。

必要に応じて、新しいインスタンスに対応するためにこれらのレシピによりインスタンスが更新されます。

- AWS OpsWorks スタックは、インスタンスの開始アクションを停止に置き換えます。停止は、インスタンスを停止するために使用できます。

インスタンスが正常に起動しなかった場合、または Setup レシピが失敗した場合、ステータスはそれぞれ `start_failed` または `setup_failed` に設定されます。ログを確認して原因を特定できます。詳細については、「[デバッグとトラブルシューティングのガイド](#)」を参照してください。

停止されたインスタンスはスタックの一部として残り、すべてのリソースを保持します。例えば、Amazon EBS ボリュームや Elastic IP アドレスは、停止したインスタンスに関連付けられたままです。停止したインスタンスを再開するには、インスタンスの [Actions] (アクション) 列で [start] (開始) を選択します。停止したインスタンスを再開すると、次の処理が実行されます。

- Instance store-backed インスタンス – AWS OpsWorks スタックは、同じ設定で新しい Amazon EC2 インスタンスを起動します。
- Amazon EBS-backed インスタンス – AWS OpsWorks スタックは Amazon EC2 インスタンスを再起動し、ルートボリュームを再アタッチします。

インスタンスの起動が完了すると、AWS OpsWorks スタックはオペレーティングシステムの更新をインストールし、最初の起動と同様に Setup および Deploy レシピを実行します。AWS OpsWorks スタックは、再起動されたインスタンスに対して必要に応じて以下も実行します。

- Elastic IP アドレスを再度関連付けます。
- Amazon Elastic Block Store (Amazon EBS) ボリュームを再接続します。
- Instance store-backed インスタンスの場合、最新のクックブックバージョンをインストールします。

Amazon EBS-backed インスタンスは、ルートボリュームに保存されたカスタムクックブックを使用し続けます。インスタンスを停止してからカスタムクックブックが変化した場合、インスタンスがオンラインになったら手動で更新する必要があります。詳細については、「[カスタムクックブックの更新](#)」を参照してください。

Note

Elastic IP アドレスが再開されたインスタンスに再度関連付けられるには数分かかる場合があります。インスタンスの Elastic IP 設定はメタデータを表しており、アドレスがインスタンスと関連付けられる必要があることを示しているにすぎない点に注意してください。[Public IP] 設定はインスタンスの状態を反映しており、最初は空の可能性もあります。Elastic IP アドレスがインスタンスに関連付けられると、そのアドレスは [Public IP] 設定に割り当てられ、その後に「(EIP)」が付きます。

インスタンスの停止

「インスタンス」ページで、インスタンスの「アクション」列の停止をクリックします。これにより、シャットダウンレシピを実行して EC2 インスタンスを終了するように AWS OpsWorks スタックに通知します。

PHP App Server

Host Name	Status	Size	Type	AZ	Public IP	Actions
php-app1	online	c1.medium	24/7	us-east-1a	54.242.127.207	stop

Are you sure you want to stop php-app1?

All data not stored on EBS volumes will be lost.

Cancel Stop

+ Instance

また、[Stop All Instances] をクリックして、すべてのインスタンスをシャットダウンすることもできます。

インスタンスを停止すると、AWS OpsWorks スタックはいくつかのタスクを実行します。

1. インスタンスのレイヤーに Elastic Load Balancing ロードバランサーがアタッチされている場合、AWS OpsWorks Stacks はインスタンスの登録を解除します。

レイヤーでロードバランサーの Connection Draining 機能がサポートされている場合、AWS OpsWorks スタックは Connection Draining が完了するまで Shutdown イベントのトリガーを遅らせます。詳細については、「[Elastic ロードバランシングレイヤー](#)」を参照してください。

2. AWS OpsWorks スタックは、インスタンスの Shutdown レシピを実行する Shutdown イベントをトリガーします。

3. Shutdown イベントをトリガーした後、AWS OpsWorks スタックは指定された時間待機して Shutdown レシピが終了するのを待ってから、以下を実行します。
 - Instance store-Backed インスタンスを終了し、これによりすべてのデータが削除されます。
 - Amazon EBS-Backed インスタンスを停止します。これによりルートボリュームのデータが保持されます。

インスタンスストレージについては、「[ストレージ](#)」を参照してください。

Note

デフォルトのシャットダウンタイムアウト設定は 120 秒です。Shutdown レシピでさらに時間が必要な場合は、[レイヤー設定を編集](#)して設定を変更できます。

シャットダウンプロセスは、インスタンスの Status 列を見ることでモニタリングできます。シャットダウン処理が進行するに従って、次のような一連の値が表示されます。

1. 終了中 - AWS OpsWorks スタックは Amazon EC2 インスタンスを終了しています。
2. shutting_down - AWS OpsWorks スタックはレイヤーのShutdownレシピを実行しています。
3. [terminated終了] (終了) - Amazon EC2 インスタンスを終了しました。
4. stopped - インスタンスは停止しました。

インスタンスの再起動

[Instances] ページで、機能していないインスタンスの名前をクリックし、詳細ページを開いた後、[Reboot] をクリックします。



このコマンドは、関連付けられた Amazon EC2 インスタンスのソフト再起動を実行します。Instance store-backed インスタンスの場合でもインスタンスのデータを削除せず、[ライフサイクルイベント](#)をトリガーしません。

Note

障害が発生したインスタンスを AWS OpsWorks スタックで自動的に置き換えるには、自動ヒーリングを有効にします。詳細については、「[自動ヒーリングの使用](#)」を参照してください。

時間ベースおよび負荷ベースのインスタンスによる負荷の管理

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

着信トラフィックが一定でない場合、負荷を処理するにはスタックのインスタンスが少なすぎたり、必要以上に多すぎたりします。時間ベースまたは負荷ベースのインスタンスを使用することで、レイヤーのインスタンス数が自動的に調整され、不必要な容量に余分な支払いをすることなく着信トラフィックを処理するのに十分なインスタンスを常に確保できるので、時間と費用の両方で節約できます。サーバーの負荷を監視したり、手動でインスタンスを起動または停止したりする必要はありません。また、時間ベースおよび負荷ベースのインスタンスは自動的に、リージョン内の複数のアベイラビリティゾーンでアプリケーションを分散、スケーリング、および均等にし、地理的冗長性と拡張性を提供します。

自動スケーリングは、2 つのインスタンスタイプに基づいており、異なる条件でレイヤーのオンラインインスタンスを調整します。

- 時間ベースのインスタンス

これらのインスタンスは、特定の時間または特定の日にのみ実行するインスタンスを含めることで、スタックが予測パターンに従って負荷を処理できるようにします。たとえば、午後 6 時から一部のインスタンスを起動して夜間のバックアップタスクを実行したり、トラフィックが少ない週末に一部のインスタンスを停止したりできます。

- 負荷ベースのインスタンス

これらのインスタンスにより、任意の負荷メトリクスに基づいて、トラフィックが多い時には追加のインスタンスを起動し、トラフィックが少ない時にはトラフィック停止することで、スタックが変化する負荷を処理できるようにします。例えば、平均 CPU 使用率が 80% を超えたときに AWS OpsWorks スタックでインスタンスを起動し、平均 CPU 負荷が 60% を下回ったときにインスタンスを停止させることができます。

Linux スタックでは時間ベースと負荷ベース両方のインスタンスがサポートされますが、Windows スタックでは時間ベースのインスタンスのみがサポートされています。

手動で起動および停止する必要がある 24/7 インスタンスとは異なり、時間ベースまたは負荷ベースのインスタンスは手動で起動または停止する必要がありません。代わりに、インスタンスを設定すると、AWS OpsWorks スタックは設定に基づいてインスタンスを起動または停止します。例えば、指定したスケジュールで開始および停止するように時間ベースのインスタンスを設定します。AWS OpsWorks スタックはその設定に従ってインスタンスを開始および停止します。

一般的な方法は、次のように 3 つのインスタンスタイプを共に使用することです。

- 基本負荷を処理する一連の 24/7 インスタンス。通常、これらのインスタンスを起動し、常に実行するようにします。
- 予測可能なトラフィックの変動を処理するために AWS OpsWorks スタックが開始および停止する、時間ベースのインスタンスのセット。たとえば、就業時間にトラフィックが最も多い場合、朝に起動して夜にシャットダウンするように時間ベースのインスタンスを設定できます。
- 予測不可能なトラフィックの変動を処理するために AWS OpsWorks スタックが開始および停止する一連のロードベースのインスタンス。AWS OpsWorks スタックは、ロードがスタックの 24/7 および時間ベースのインスタンスの容量に近づいたときに起動し、トラフィックが通常に戻るときに停止します。

これらのスケーリング時間を処理する方法の詳細については、「[サーバー数の最適化](#)」を参照してください。

Note

インスタンスのレイヤー用のアプリケーションを作成した場合、またはカスタムクックブックを作成した場合、AWS OpsWorks スタックは、最初に起動されたときに、タイムベースおよびロードベースのインスタンスに最新バージョンを自動的にデプロイします。ただし、AWS OpsWorks スタックは、再起動されたオフラインインスタンスに最新のクックブックを

必ずしもデプロイするわけではありません。詳細については、「[アプリケーションの編集](#)」および「[カスタムクックブックの更新](#)」を参照してください。

トピック

- [自動時間ベースのスケーリングの使用](#)
- [自動負荷ベースのスケーリングの使用](#)
- [負荷ベースのスケーリングと自動修復の違い](#)

自動時間ベースのスケーリングの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

時間ベースのスケーリングでは、指定したスケジュールでインスタンスを開始または停止することで、レイヤーが特定の時間帯または曜日にオンラインにするインスタンスの数を制御できます。AWS OpsWorks スタックは数分ごとにチェックし、必要に応じてインスタンスを開始または停止します。次のように、インスタンスごとに個別の予定を指定します。

- 時刻。たとえば、夜間よりも日中により多くのインスタンスを実行することができます。
- 曜日。たとえば、週末より平日により多くのインスタンスを実行することができます。

Note

特定の日付を指定することはできません。

トピック

- [レイヤーへの時間ベースのインスタンスを追加する](#)

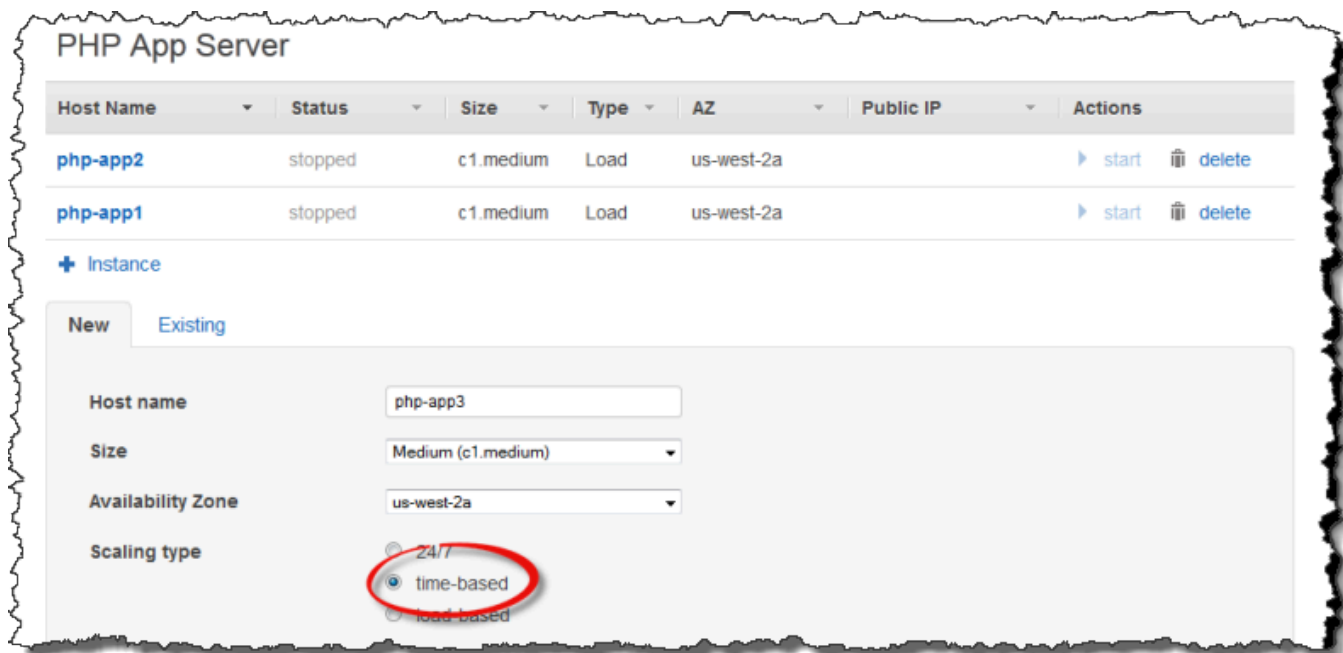
• 時間ベースのインスタンスの構成

レイヤーへの時間ベースのインスタンスを追加する

レイヤーに新しい時間ベースのインスタンスを追加するか、既存のインスタンスを使用します。

新しい時間ベースのインスタンスを追加するには

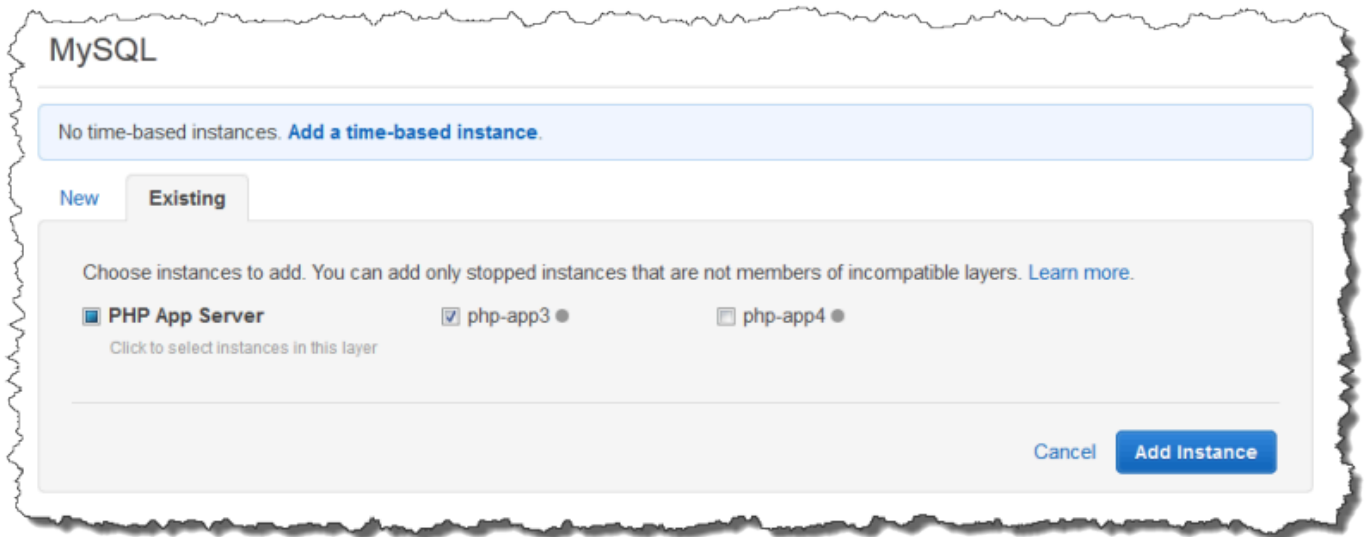
1. [インスタンス] ページで [+ インスタンス] を選択して、インスタンスを追加します。[新規] タブで、[アドバンスド] を選択してから、[時間ベース] を選択します。



2. インスタンスを設定します。次に、[インスタンスの追加] を選択して、インスタンスをレイヤーに追加します。

既存の時間ベースのインスタンスをレイヤーに追加するには

1. [時間ベースのインスタンス] ページで、すでに時間ベースのインスタンスがある場合、[+ インスタンス] を選択します。それ以外の場合は、[時間ベースのインスタンスを追加] をクリックします。次に、[既存] タブを選択します。



2. [既存] タブで、リストからインスタンスを選択します。リストには時間ベースのインスタンスのみが表示されます。

Note

既存のインスタンスを使用しなくなった場合は、前述の手順に従って、[新規] タブで新しいインスタンスを作成します。

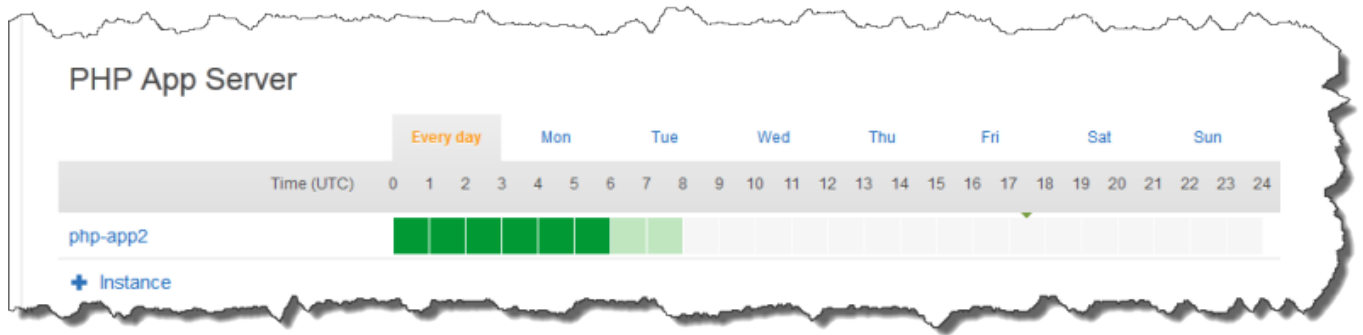
3. [インスタンスの追加] を選択して、インスタンスをレイヤーに追加します。

時間ベースのインスタンスの構成

レイヤーに時間ベースのインスタンスを追加したら、次のようにスケジュールを設定します。

時間ベースのインスタンスを設定するには

1. ナビゲーションペインの [インスタンス] で、[時間ベース] を選択します。
2. 希望する時間の下にある適切なボックスを入力して、各時間ベースのインスタンスにオンライン期間を指定します。
 - 毎日同じスケジュールを使用するには、[毎日] タブを選択し、次にオンライン期間を指定します。
 - 異なる日に異なるスケジュールを使用するには、毎日を選択し、次に適切な期間を選択します。



Note

インスタンスの起動にかかる時間を確保し、AWS OpsWorks スタックがインスタンスを起動または停止する必要があるかどうかを確認するのに数分ごとにのみチェックするようにします。たとえば、インスタンスを 1 時 (UTC) までに実行する必要がある場合、0 時 (UTC) にそのインスタンスを起動します。そうしないと、AWS OpsWorks スタックは UTC の 1:00 を数分経過するまでインスタンスを起動せず、インスタンスがオンラインになるまでにさらに数分かかることがあります。

上記の手順を実行することで、インスタンスのオンライン期間をいつでも変更できます。次回 AWS OpsWorks スタックがチェックするときは、新しいスケジュールを使用して、インスタンスを起動するか停止するかを決定します。

Note

レイヤーに新しい時間ベースのインスタンスを追加するには、[時間ベース] ページを開き、[時間ベースのインスタンスを追加] (レイヤーに時間ベースのインスタンスをまだ追加していない場合) を選択、または [+インスタンス] (レイヤーにすでに1つ以上の時間ベースのインスタンスがある場合) を選択します。次に、前述の手順で説明したようにインスタンスを構成します。

自動負荷ベースのスケーリングの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

ロードベースのインスタンスを使用すると、受信トラフィックの変化に応じてインスタンスをすばやく開始または停止できます。AWS OpsWorks スタックは [Amazon CloudWatch](#) データを使用して、レイヤーごとに次のメトリクスを計算します。これは、レイヤーのすべてのインスタンスの平均値を表します。

- CPU: CPU の平均使用率 (例、80%)
- メモリ: メモリの平均使用率 (例、60%)
- 負荷: システムが 1 分で実行する平均計算作業。

これらのメトリクスのいずれか、またはすべてに、上限しきい値と下限しきい値を定義します。カスタム CloudWatch アラームをしきい値として使用することもできます。

しきい値を超えると、スケールイベントがトリガーされます。以下の内容を指定することで、AWS OpsWorks スタックがスケールイベントに対応する方法を決定します。

- 起動または停止するインスタンス数。
- インスタンスを起動または削除する前に、AWS OpsWorks スタックがしきい値を超えた後に待機する時間。たとえば、CPU 使用率は少なくとも 15 分間はしきい値を超過する必要があります。この値により、トラフィックの短い変動を無視することができます。
- メトリクスを再度モニタリングする前に、インスタンスの起動または停止後に AWS OpsWorks スタックが待機する時間。レイヤーが引き続きしきい値を超えているかどうかを評価する前に、起動されたインスタンスがオンラインになる、または停止されたインスタンスがシャットダウンされるまで十分な時間が経過する必要があります。

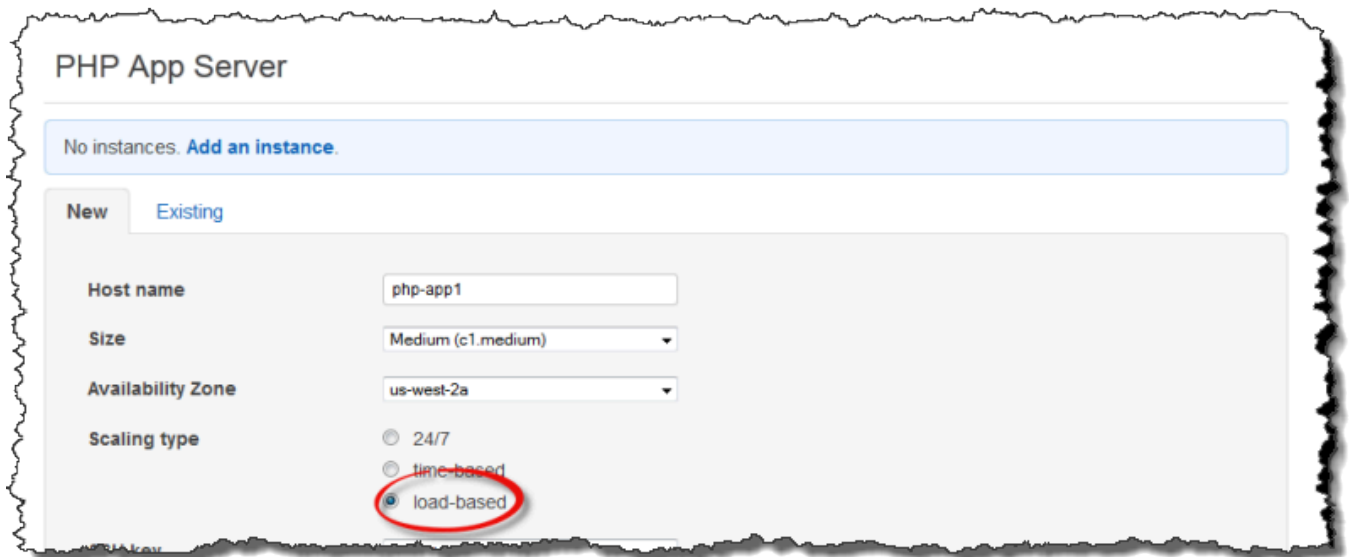
スケーリングイベントが発生すると、AWS OpsWorks スタックは負荷ベースのインスタンスのみを開始または停止します。24/7 インスタンスまたは時間ベースのインスタンスは起動または停止されません。

Note

負荷ベースの自動スケーリングでは、新しいインスタンスは作成されません。ユーザーが作成したインスタンスのみが起動および停止されます。したがって、予測される最大負荷を処理するのに十分な負荷ベースのインスタンスを事前に備える必要があります。

負荷ベースのインスタンスを作成するには

1. [インスタンス] ページの [+ インスタンス] を選択して、インスタンスを追加します。[アドバンスト] を選択してから、[負荷ベース] を選択します。



2. インスタンスを設定し、[+ インスタンスを追加] を選択して、インスタンスをレイヤーに追加します。

必要な数のインスタンスが作成されるまで、この手順を繰り返します。必要に応じて、後からインスタンスを追加または削除することができます。

負荷ベースのインスタンスをレイヤーに追加したら、負荷ベースのスケーリングを可能にし、設定を指定する必要があります。負荷ベースのスケーリング設定はインスタンスのプロパティではなく、レイヤーのプロパティであるため、レイヤーが負荷ベースのインスタンスを起動または停止する時を指定します。負荷ベースのインスタンスを使用するレイヤーごとに個別に指定する必要があります。

負荷ベースの自動スケーリングを可能にし設定するには

1. ナビゲーションペインの [インスタンス] で [負荷ベース] を選択し、続いて適切なレイヤーの [編集] を選択します。

The screenshot shows the AWS OpsWorks console interface. On the left is a navigation sidebar with options: Stack, Layers, Instances (Time-based, Load-based), Apps, Deployments, Monitoring, and Permissions. The main content area is titled 'Load-based instances' and shows details for a 'PHP App Server' layer. A yellow banner indicates 'Load-based auto scaling is disabled - edit'. Below this, it states '0 of 1 instances are running show >' and includes a '+ Instance' button.

2. [Load-based auto scaling enabled] (負荷ベースの自動スケーリングの有効化) を [On] に設定します。次に、しきい値とスケーリングパラメータを設定し、インスタンスを追加する方法とタイミングを定義します。

Load-based Rails App Server Configuration

Scaling configuration On

Based on Layer averages

Metric	UP	DOWN
Average CPU	80 %	30 %
Average memory		
Average load		

Scaling parameters

	UP	DOWN
Start servers in batches of	1	Stop servers in batches of 1
If thresholds are exceeded	5 min	If thresholds are undershot 10 min
After scaling, ignore metrics	5 min	After scaling, ignore metrics 10 min

Based on Amazon CloudWatch alarms

UP

DOWN

Cancel

レイヤー平均しきい値

スケーリングしきい値は、レイヤーのすべてのインスタンス間で平均された以下の値に基づいて設定できます。

- [Average CPU] (平均CPU) – レイヤーの平均 CPU 使用率 (合計に対する割合)。
- [Average memory] (平均メモリ) – レイヤーの平均メモリ使用率 (合計に対する割合)。
- [Average load] (平均負荷) – レイヤーの平均負荷。

負荷を計算する方法の詳細については、ウィキペディアで [\[Load \(computing\)\]](#) (負荷(コンピューティング)) を参照してください。

しきい値を超えると、スケーリングイベントが発生し、より多くのインスタンスが必要な場合はアップスケーリングされ、より少ないインスタンスが必要な場合はダウンスケーリングされます。AWS OpsWorks その後、スタックはスケーリングパラメータに基づいてインスタンスを追加または削除します。

カスタム CloudWatch アラーム

最大 5 つのカスタム CloudWatch アラームをアップスケーリングまたはダウンスケーリングのしきい値として使用できます。これらは、スタックと同じリージョンにある必要があります。カスタムアラームの作成方法の詳細については、[「Amazon CloudWatch アラームの作成」](#) を参照してください。

Note

カスタムアラームを使用するには、サービスロールを更新して `cloudwatch:DescribeAlarms` を許可する必要があります。この機能を初めて使用する場合は、AWS OpsWorks スタックでロールを更新するか、手動でロールを編集できます。詳細については、[「AWS OpsWorks スタックがユーザーに代わって動作することを許可する」](#) を参照してください。

負荷ベース設定に複数のアラームが設定されている場合、あるアラームが `INSUFFICIENT_DATA` メトリックアラーム状態になっていると、他のアラームが `ALARM` 状態になっていてもロードベースのインスタンススケーリングができません。オートスケーリングは、すべてのアラームが `OK` or `ALARM` の状態の場合にのみ実行できます。Amazon CloudWatch アラームの使用の詳細については、[「Amazon ユーザーガイド」](#) の [「Amazon CloudWatch アラームの使用」](#) を参照してください。
CloudWatch

スケーリングパラメータ

以下のパラメータは、AWS OpsWorks スタックがスケーリングイベントを管理する方法を制御します。

- [Start servers in batches of] (サーバをバッチで起動) - スケーリングイベントが発生したときに追加または削除するインスタンスの数。
- しきい値を超えた場合 - AWS OpsWorks スタックがスケーリングイベントをトリガーするまでに、負荷がアップスケーリングしきい値またはダウンスケーリングしきい値を下回っている必要がある時間 (分単位)。
- スケーリング後、メトリクスを無視する - スケーリングイベントが発生してから、AWS OpsWorks スタックがメトリクスを無視して追加のスケーリングイベントを抑制する時間 (分単位)。

例えば、AWS OpsWorks スタックはアップスケーリングイベント後に新しいインスタンスを追加しますが、インスタンスは起動して設定されるまで負荷の軽減を開始しません。新しいインスタンスがオンラインになってリクエストを処理するまで (通常、これには数分かかります)、追加のスケーリングイベントが発生するポイントはありません。この設定を使用すると、新しいインスタンスがオンラインになるまでの間 AWS OpsWorks スタックにスケーリングイベントを抑制させることができます。

この設定を増やすことで、[Average CPU] (平均CPU)、[Average memory] (平均メモリ)、[Average load] (平均負荷) などのレイヤーの平均値が一時的に不一致になった場合に、スケーリングの急激な変動を防ぐことができます。

例えば、CPU 使用率が制限を超え、メモリ使用量がダウンスケールに近い場合、インスタンスのアップスケールイベントの直後に、メモリのダウンスケールイベントが続く可能性があります。これを回避するには、[After scaling, ignore metrics] (スケーリング後、メトリクスを無視する) 設定で分単位で時間を増やすことができます。この例では、CPU のスケーリングが発生しますが、メモリのダウンスケールイベントは発生しません。

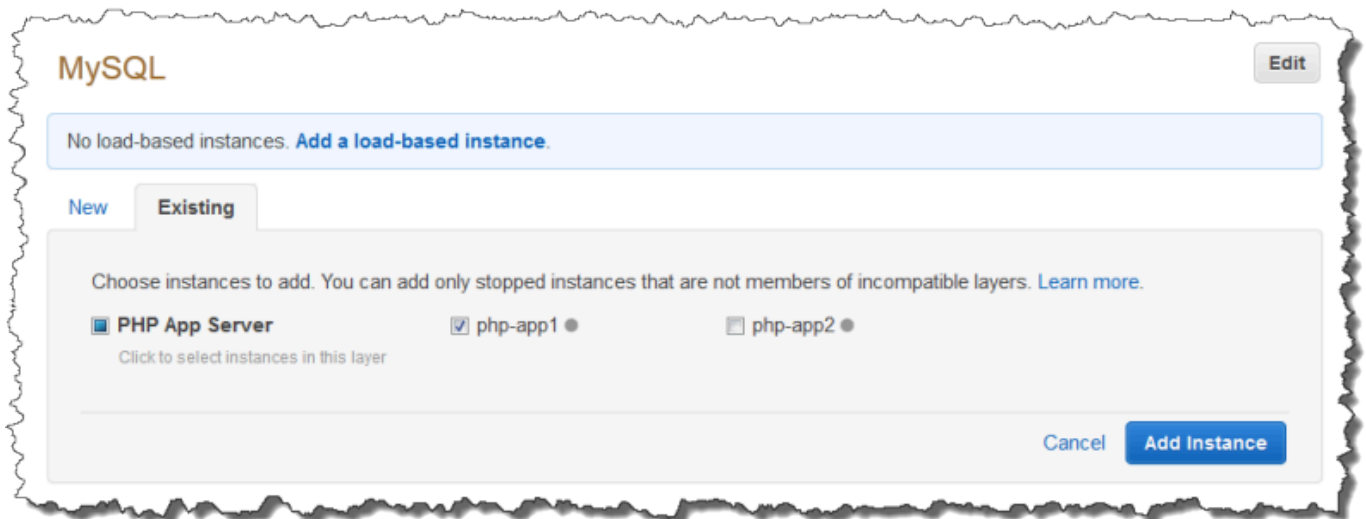
3. 負荷ベースのインスタンスを追加するには、[+ Instance] (+ インスタンス) を選択し、設定を行い、[Add Instance] (インスタンスを追加) を選択します。予測される最大負荷を処理するのに十分な負荷ベースのインスタンス数になるまで繰り返します。次に、[Save] (保存) を選択します。

Note

レイヤーに新しい負荷ベースのインスタンスを追加するには、[Load-based] (負荷ベース) ページを開き、[Add a load-based instance] (ロードベースインスタンスを追加) (負荷ベースのインスタンスをレイヤーにまだ追加していない場合) を選択するか、[+ Instance] (+ インスタンス) (1 つ以上の負荷ベースのインスタンスがすでにレイヤーにある場合) を選択します。次に、このセクションで前述したように、インスタンスを設定します。

既存の負荷ベースのインスタンスをレイヤーに追加するには

1. ナビゲーションペインの [Instances] (インスタンス) で、[Load-based] (負荷ベース) を選択します。
2. レイヤーの負荷ベースの自動スケーリングをすでに有効にしている場合は、[+ Instance] (+ インスタンス) を選択します。それ以外の場合は、[Add a load-based instance] (負荷ベースのインスタンスを追加) を選択します。[EXisting] (既存) タブ。



3. [EXisting] (既存) タブで、インスタンスを選択します。リストには、負荷ベースのインスタンスのみが表示されます。

Note

既存のインスタンスを使用しなくなった場合は、[New] (新規) タブで、前述の手順に従って新しいインスタンスを作成します。

4. [Add Instance] (インスタンスを追加) を選択して、インスタンスをレイヤーに追加します。

いつでも、負荷ベースの自動スケーリングの設定を変更したり、負荷ベースの自動スケーリングを無効にしたりすることができます。

負荷ベースのスケーリングを無効にするには

1. ナビゲーションペインの [Instances] (インスタンス) で、[負荷ベース] (Load-based) を選択してから、適切なレイヤーの [edit] (編集) を選択します。
2. [Load-based auto scaling enabled] (負荷ベースの自動スケーリングの有効化) を [No] (いいえ) に切り替えます。

負荷ベースのスケーリングと自動修復の違い

負荷ベースの自動スケーリングは、実行中のすべてのインスタンスで平均される負荷メトリクスを使用します。メトリクスが指定されたしきい値の間にとどまる場合、AWS OpsWorks スタックはインスタンスを開始または停止しません。一方、自動ヒーリングでは、インスタンスの応答が停止すると、AWS OpsWorks スタックは同じ設定で新しいインスタンスを自動的に起動します。ネットワークの問題やインスタンスに障害が発生した場合に、インスタンスが応答できない場合があります。

例えば、CPU のアップスケーリングのしきい値が 80% で、1 つのインスタンスが応答を停止するとします。

- 自動ヒーリングが無効になっており、残りの実行中のインスタンスが平均 CPU 使用率を 80% 未満に維持できる場合、AWS OpsWorks スタックは新しいインスタンスを起動しません。残りのインスタンス全体で CPU の平均使用率が 80% を超えた場合、代替インスタンスを起動します。
- 自動ヒーリングが有効になっている場合、AWS OpsWorks スタックは負荷しきい値に関係なく代替インスタンスを起動します。

AWS OpsWorks Stacks の外部で作成されたコンピューティングリソースの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

この機能は Linux スタックでのみサポートされています。

[インスタンス](#) では、AWS OpsWorks スタックを使用して Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのグループを作成および管理する方法を説明します。スタックの外部で作成されたスタックに Linux AWS OpsWorks コンピューティングリソースを組み込むこともできます。

- Amazon EC2コンソール、CLIまたは を使用して直接作成した Amazon EC2インスタンスAPI。
- 仮想マシンで実行しているインスタンスを含め、独自のハードウェアで実行している On-premises (オンプレミス) インスタンス。

これらのコンピューティングリソースは AWS OpsWorks スタックマネージドインスタンスになり、通常の AWS OpsWorks スタックインスタンスと同様に管理できます。

- ユーザーアクセス許可の管理 – [AWS OpsWorks スタックのユーザー管理](#)を使用して、スタックへのアクセスを許可するユーザー、スタックのインスタンスで実行できるアクション、SSHアクセス権限と sudo 権限があるかどうかを指定できます。
- タスクの自動化 – AWS OpsWorks スタックでカスタム Chef レシピを実行して、スタックのインスタンスの一部またはすべてで1つのコマンドでスクリプトを実行するなどのタスクを実行できます。

インスタンスを[レイヤー](#)に割り当てると、AWS OpsWorks Stacks は、カスタムレシピを含む[ライフサイクル](#)の主要なポイントで、指定された一連の Chef レシピをインスタンスで自動的に実行します。登録された Amazon EC2インスタンスは、[カスタムレイヤー](#)にのみ割り当てることができます。

- リソースの管理 – スタックを使用すると、内のリソースをグループ化して管理でき AWS リージョン、OpsWorks ダッシュボードにはすべてのリージョンのスタックのステータスが表示されます。
- [Install packages] (パッケージのインストール) - Chef レシピを使用して、スタックの任意のインスタンスにパッケージをインストールできます。

- オペレーティングシステムの更新 – AWS OpsWorks スタックは、オペレーティングシステムのセキュリティパッチと更新をスタックのインスタンスに簡単にインストールする方法を提供します。
- アプリケーションのデプロイ – AWS OpsWorks スタックは、スタックのすべてのアプリケーションサーバーインスタンスにアプリケーションを一貫してデプロイします。
- モニタリング – AWS OpsWorks スタックは、スタックのすべてのインスタンスをモニタリングするためのカスタム [CloudWatch](#) メトリクスを作成します。

料金に関する情報については、[\[AWS OpsWorks の料金\]](#)を参照してください。

以下に、登録されたインスタンスを使用する際の基本的な手順を示します。

1. スタックにインスタンスを登録します。

これで、インスタンスはスタックの一部になり、AWS OpsWorks スタックによって管理されません。

2. オプションで、インスタンスをレイヤーに割り当てます。

このステップでは、AWS OpsWorks スタックの管理機能を最大限に活用できます。登録されたオンプレミスインスタンスは任意のレイヤーに割り当てることができます。登録された Amazon EC2 インスタンスはカスタムレイヤーにのみ割り当てることができます。

3. AWS OpsWorks スタックを使用してインスタンスを管理します。

4. スタックでインスタンスが不要になった場合は、登録を解除します。これにより、インスタンスは AWS OpsWorks スタックから削除されます。

以下のセクションではこのプロセスを詳しく説明します。

トピック

- [AWS OpsWorks スタックスタックへのインスタンスの登録](#)
- [登録されたインスタンスの管理](#)
- [登録されたインスタンスをレイヤーに割り当てる](#)
- [登録されたインスタンスの割り当てを解除する](#)
- [登録されたインスタンスの登録を解除する](#)
- [登録されたインスタンスのライフサイクル](#)

AWS OpsWorks スタックスタックへのインスタンスの登録

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

この機能は Linux スタックでのみサポートされています。

AWS OpsWorks スタックの外部にあるインスタンスを登録するには、コマンドを実行します AWS CLI `aws opsworks register`。このコマンドは、登録したいインスタンスから実行することも、別のコンピュータから実行することもできます。AWSOpsWorksRegisterCLI_EC2 または AWSOpsWorksRegisterCLI_OnPremises ポリシーをユーザーまたはグループに適用して、が EC2 または オンプレミスインスタンスをそれぞれ登録 AWS CLI するために必要なアクセス許可を付与します。これらのポリシーには、バージョン 1.16.180 AWS CLI 以降が必要です。

Note

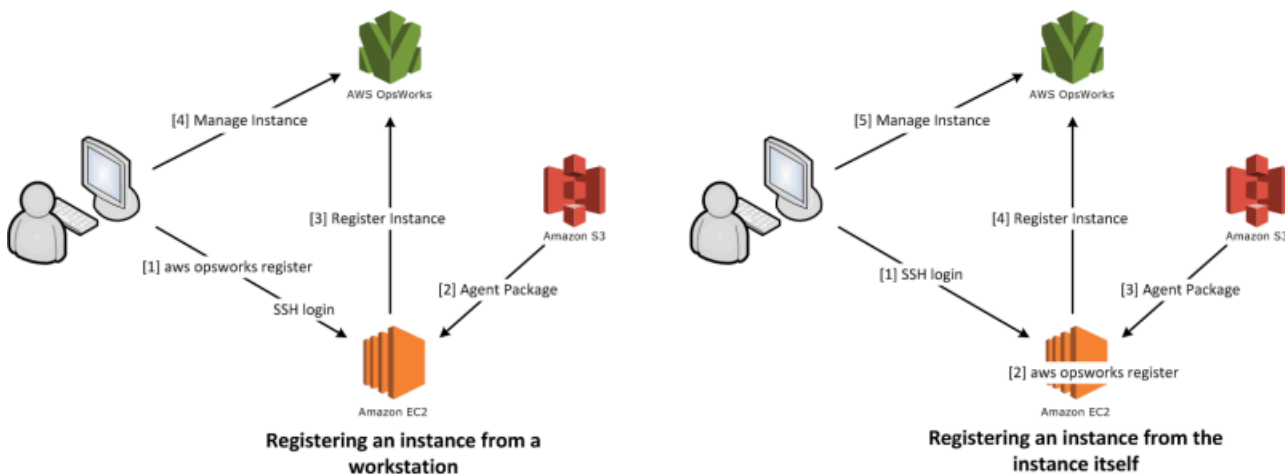
ユーザーまたはロールがインスタンスを登録できないようにするには、インスタンスプロファイルを更新して `register` コマンドへのアクセスを拒否します。

登録プロセスでは、スタックを使用して管理するインスタンスにエージェントをインストールし、指定した AWS OpsWorks AWS OpsWorks スタックにインスタンスを登録します。インスタンスを登録すると、インスタンスはスタックの一部となり、AWS OpsWorks Stacks により管理されます。詳細については、「[登録されたインスタンスの管理](#)」を参照してください。

Note

[AWS Tools for PowerShell](#) には、[Register-OpsInstance](#) registerAPIアクションを呼び出す コマンドレットが含まれていますが、代わりに AWS CLI を使用して register コマンドを実行することをお勧めします。

次の図は、Amazon EC2インスタンスを登録する両方のアプローチを示しています。オンプレミスインスタンスを登録する場合も同じ方法を使用できます。

**Note**

[AWS OpsWorks スタックコンソール](#)を使用して登録されたインスタンスを管理できますが、AWS CLI `register` コマンドを実行してインスタンスを登録する必要があります。これは、登録処理はインスタンスから実行しなければならず、コンソールでは実行できないためです。

以下のセクションではこの手続きを詳しく説明します。

トピック

- [チュートリアル: ワークステーションからインスタンスを登録する](#)
- [Amazon インスタンスEC2とオンプレミスインスタンスの登録](#)

チュートリアル: ワークステーションからインスタンスを登録する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

この機能は Linux スタックでのみサポートされています。

登録処理は、いくつかのシナリオをサポートしています。このセクションでは、end-to-endワークステーションを使用して Amazon EC2 インスタンスを登録する方法という 1 つのシナリオの例を示します。他の登録シナリオでも手順は似ています。詳細については、「[Amazon インスタンス EC2 と オンプレミスインスタンスの登録](#)」を参照してください。

Note

通常、既存の Amazon EC2 インスタンスを登録します。ただし、チュートリアル専用新しいインスタンスと新しいスタックを作成し、チュートリアルの完了後に削除できます。

トピック

- [ステップ 1: スタックとインスタンスを作成する](#)
- [ステップ 2: AWS CLI をインストールおよび設定する](#)
- [ステップ 3: EC2Register スタックにインスタンスを登録する](#)

ステップ 1: スタックとインスタンスを作成する

開始するには、スタックと Amazon EC2 インスタンスをそのスタックに登録する必要があります。

スタックとインスタンスを作成するには

1. **EC2Register** という名前の [新しいスタックの作成](#) ために [AWS OpsWorks スタックコンソール](#) を使用します。その他のスタック設定については、デフォルト値を使用できます。
2. [Amazon EC2コンソール](#) から新しいインスタンスを起動します。次の点に注意してください。

- インスタンスは、スタックVPCと同じリージョンおよび [ゾーン](#) にある必要があります。

を使用している場合はVPC、このワークスルーのパブリックサブネットを選択します。

- SSH キーを作成する必要がある場合は、プライベートキーファイルをワークステーションに保存し、名前とファイルの場所を記録します。

既存のキーを使用する場合は、名前とプライベートキーファイルの場所を記録します。これらの値は後で必要になります。

- インスタンスは、「[サポートされている Linux オペレーティングシステム](#)」のいずれかに基づいている必要があります。例えば、スタックが米国西部 (オレゴン) にある場合、ami-35501205を使用してそのリージョンで Ubuntu 14.04 LTSインスタンスを起動できません。

そうしない場合は、デフォルト値をそのまま使用します。

インスタンスの起動中、次のセクションに進むことができます。

ステップ 2: AWS CLIをインストールおよび設定する

登録は、AWS CLI `aws opsworks register` コマンドを使用して実行されます。最初のインスタンスを登録する前に、バージョン 1.16.180 AWS CLI 以降を実行している必要があります。インストールの詳細は使用しているワークステーションのオペレーティングシステムによって異なります。のインストールの詳細については AWS CLI、[AWS 「コマンドラインインターフェイスのインストール」](#) を参照してください。起動している AWS CLI のバージョンを確認するには、シェルセッション内に `aws --version` を入力します。

Note

ユーザーまたはロールがインスタンスを登録できないようにするには、インスタンスプロファイルを更新して `register` コマンドへのアクセスを拒否します。

ワークステーション AWS CLI で を既に実行している場合でも、このステップはスキップしないことを強くお勧めします。AWS CLI の最新リリースを使用することがセキュリティのベストプラクティスです。

適切なアクセス許可を持つAWS認証情報のセットを `register` に提供する必要があります。インスタンスに直接認証情報をインストールしないようにするには、インスタンスプロファイルで起動したインスタンスを登録し、`register` コマンドに `--use-instance-profile` スイッチを追加する方法が推奨されます。インスタンスプロファイルから認証情報を取得する場合は、この手順をスキップして、このトピックの「[ステップ 3: EC2Register スタックにインスタンスを登録する](#)」に進んでください。ただし、インスタンスがインスタンスプロファイルで起動されていない場合は、IAM ユーザーを作成できます。以下の手順では、適切なアクセス権限がある新しいユーザーを作成し、ユーザーの認証情報をワークステーションにインストールして、その認証情報を `register` に渡しています。

Warning

IAM ユーザーは長期的な認証情報を持っているため、セキュリティ上のリスクがあります。このリスクを軽減するために、これらのユーザーにはタスクの実行に必要な権限のみを付与し、不要になったユーザーを削除することをお勧めします。

ユーザーを作成するには

1. [IAM コンソール](#)で、ナビゲーションペインでユーザーを選択し、ユーザーの追加を選択します。
2. **EC2Register** という名前のユーザーを追加します。
3. [Next (次へ)] を選択します。
4. 許可を設定 ページで、ポリシーを直接アタッチする を選択します。
5. **OpsWorks** アクセス許可ポリシーフィルターボックスにと入力して AWS OpsWorks ポリシーを表示し、次のいずれかのポリシーを選択してから、次へ: 確認を選択します。このポリシーは、`register` の実行に必要なアクセス権限をお客様のユーザーに付与します。
 - インスタンスプロファイルを使用するEC2インスタンスを登録するアクセス許可をユーザーに付与AWSOpsWorksRegisterCLI_EC2するには、 を選択します。
 - オンプレミスインスタンスを登録するためのユーザー権限を付与するには、 [AWSOpsWorksRegisterCLI_OnPremises] を選択します。
6. [Next (次へ)] を選択します。

7. [確認] ページで、[ユーザーの作成] を選択します。
8. ユーザーのアクセスキーを作成します。ナビゲーションペインから、ユーザー を選択し、アクセス キーを作成するユーザーを選択します。
9. セキュリティ認証情報 タブを選択してから アクセスキーの作成 を選択します。
10. タスクに最も適した アクセスキーのベストプラクティスと代替案 を選択してください。
11. [Next (次へ)] を選択します。
12. (オプション) アクセスキーを識別するタグを入力します。
13. [Next (次へ)] を選択します。
14. [Download .csvファイル] を選択し、認証情報ファイルをシステムの都合のよい場所に保存してから、完了 を選択します。

IAM ユーザーの認証情報を `register` に提供する必要があります。このチュートリアルでは、ワークステーションの `credentials` ファイルに EC2Register 認証情報をインストールしてタスクを処理します。の認証情報を管理するその他の方法については AWS CLI、[「設定ファイルと認証情報ファイル」](#) を参照してください。

ユーザーの認証情報をインストールするには

1. ワークステーションの `credentials` ファイルを作成するか開きます。このファイルは、`~/.aws/credentials` (Linux、Unix、OS X) または `C:\Users\User_Name\.aws\credentials` (Windows システム) にあります。
2. 次の形式を使用して、EC2Register ユーザーのプロファイルを `credentials` ファイルに追加します。

```
[ec2register]
aws_access_key_id = access_key_id
aws_secret_access_key = secret_access_key
```

access_key_id と *secret_access_key* を、前にダウンロードした の EC2Register キー *secret_access_key* に置き換えます。

ステップ 3: EC2Register スタックにインスタンスを登録する

これでインスタンスを登録する準備が整いました。

インスタンスを登録するには

1. AWS OpsWorks スタックで、EC2Registerスタックに戻り、ナビゲーションペインでインスタンスを選択し、インスタンスの登録を選択します。
2. EC2 インスタンスを選択し、次へ: インスタンスを選択し、リストからインスタンスを選択します。
3. 次へ: のインストールAWSCLI、次へ: インスタンスの登録 AWS OpsWorks スタックは、スタック ID やインスタンス ID などの利用可能な情報を自動的に使用してregisterコマンドテンプレートを作成します。これはインスタンスの登録ページに表示されます。この例では、registerを使用して SSHキーでインスタンスにログインし、キーファイルを明示的に指定するため、キーを使用してインスタンスを はい SSH に接続します。このコマンドテンプレートは次のようになっています。

```
aws opsworks register --infrastructure-class ec2 --region region endpoint ID
--stack-id 247be7ea-3551-4177-9524-1ff804f453e3 --ssh-username [username]
--ssh-private-key [key-file] i-f1245d10
```

Note

AWS OpsWorks スタックがリージョンエンドポイントに関連付けられたクラシックリージョン内にある場合は、us-east-1リージョンをスタックのリージョンではなく、スタックサービスのエンドポイントリージョンに設定する必要があります。AWS OpsWorks スタックは、スタック ID からスタックのリージョンを決定します。

4. コマンド テンプレートには複数のユーザー固有の引数値も含まれます。これらの引数値はブラケット付きで示され、適切な値に置き換える必要があります。コマンドテンプレートをテキストエディタにコピーして、次のように編集します。

Important

登録プロセス中に作成された IAM ユーザーは、登録されたインスタンスの存続中に必要です。ユーザーを削除すると、AWS OpsWorks スタックエージェントはサービスと通信できなくなります。--use-instance-profile ユーザーが誤って削除された場合に登録済みインスタンスの管理に関する問題が発生するのを回避するために、registerコマンドに パラメータを追加して、代わりにインスタンスの組み込みインスタンスプロファイルを使用します。また、--use-instance-profileパラメータを追加すると、AWS アカウントアクセスキーを 90 日ごとにローテーションするとき

エラーが発生することも防止されます AWS OpsWorks (推奨されるベストプラクティス)。これは、エージェントと必要なIAMユーザーが利用できるアクセスキーが一致しなくなるためです。

- を、インスタンスの作成時に保存した Amazon キーペアのプライベートEC2キーファイルの完全修飾パス *key file* に置き換えます。

必要に応じて相対パスを使用できます。

- をインスタンスのユーザー名 *username* に置き換えます。

この例では、ユーザー名は `ubuntu`、Ubuntu インスタンスの場合は、Red Hat Enterprise Linux (RHEL) または Amazon Linux インスタンス `ec2-user` の場合は です。

- インスタンスプロファイルを使用して `register` を実行するように `--use-instance-profile` を追加して、キーローテーション時や、プリンシパル IAM ユーザーが誤って削除された場合のエラーを回避します。

コマンドは次のようになります。

```
aws opsworks register --use-instance-profile --infrastructure-class ec2 \  
  --region us-west-2 --stack-id 247be7ea-3551-4177-9524-1ff804f453e3 --ssh-  
  username ubuntu \  
  --ssh-private-key "./keys/mykeys.pem" i-f1245d10
```

5. ワークステーションでターミナルウィンドウを開き、エディタから `register` コマンドを貼り付けてコマンドを実行します。

通常、登録には約 5 分かかります。完了したら、AWS OpsWorks スタックコンソールに戻り、完了を選択します。次に、ナビゲーションペインで [Instances (インスタンス)] を選択します。[Unassigned Instances] にインスタンスが表示されます。次に、インスタンスをどのように管理するかに応じて、[インスタンスをレイヤーに割り当てる](#)か、そのままにします。

6. 完了したら、[インスタンスを停止](#)し、AWS OpsWorks スタックコンソールまたは コマンドを使用して [インスタンスを削除](#)します。これにより Amazon EC2 インスタンスが終了するため、それ以上の料金は発生しません。

Amazon インスタンスEC2とオンプレミスインスタンスの登録

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

この機能は Linux スタックでのみサポートされています。

このセクションでは、Amazon EC2 または オンプレミスインスタンスを スタックスタックに登録する AWS OpsWorks 方法について説明します。

トピック

- [インスタンスの準備](#)
- [AWS CLI のインストールおよび設定](#)
- [インスタンスの登録](#)
- [register コマンドの使用](#)
- [register コマンドの例](#)
- [インスタンス登録ポリシー](#)

インスタンスの準備

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

この機能は Linux スタックでのみサポートされています。

インスタンスを登録する前に、そのインスタンスが AWS OpsWorks スタックと互換性があることを確認する必要があります。詳細は、オンプレミスインスタンスと Amazon EC2 インスタンスのどちらを登録するかによって異なります。

オンプレミスインスタンス

オンプレミスインスタンスは以下の基準を満たしている必要があります。

- インスタンスは、「[サポートされている Linux オペレーティングシステム](#)」のいずれかを実行する必要があります。カスタムまたはコミュニティで生成されたから作成された他のオペレーティングシステム (CentOS 6.x など) でインスタンスを作成または登録することは可能ですが AMIs、これらは公式にはサポートされていません。

インスタンスで `libyaml` パッケージをインストールする必要があります。Ubuntu インスタンスの場合、パッケージの名前は `libyaml-0-2` です。CentOS および Red Hat Enterprise Linux インスタンスの場合、パッケージの名前は `libyaml` です。

- インスタンスのインスタンスタイプ (インスタンスサイズと呼ばれることもあります) がサポート対象である必要があります。サポートされているインスタンスタイプはオペレーティングシステムによって異なり、スタックがあるかどうかによって異なります。VPC。サポートされているインスタンスタイプのリストについては、ターゲットスタックに新しいインスタンスを作成しようとすると AWS OpsWorks、スタックコンソールに表示されるサイズドロップダウンリストの値を表示します。インスタンスタイプが灰色で表示されていて、ターゲットスタックに作成できない場合、そのタイプのインスタンスは登録できません。
- インスタンスには、AWS OpsWorks スタックサービスエンドポイントとの通信を許可するインターネットアクセスが必要です `opsworks.us-east-1.amazonaws.com` (HTTPS)。インスタンスは、Amazon S3 などの AWS リソースへのアウトバウンド接続もサポートする必要があります。
- 別のワークステーションからインスタンスを登録する場合は、登録されたインスタンスがワークステーションからの SSH ログインをサポートしている必要があります。

SSH インスタンスから登録コマンドを実行する場合、ログインは必要ありません。

- AWS アクセスキーは、エージェントから AWS OpsWorks AWS OpsWorks スタックサービスへの認証に使用されます。アクセスキーを推奨どおりに 90 日ごとにローテーションする場合は、新

しいキーを使用するように AWS OpsWorks エージェントを手動で更新します。オンプレミスのコンピュータまたはインスタンスで、新しいアクセスキーとシークレットキーを使用して `/etc/aws/opsworks/instance-agent.yml` ファイルを編集します。次のコマンドを実行すると、このファイル内のアクセスキーとシークレットキーが表示されます。以前のキーを使用しているエージェントでは、エラーが発生することがあります。

```
cat /etc/aws/opsworks/instance-agent.yml | egrep "access_key|secret_key"
:access_key_id: AKIAIOSFODNN7EXAMPLE
:secret_access_key: wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
```

Amazon EC2 インスタンス

Amazon EC2インスタンスは、次の条件を満たす必要があります。

- は、サポートされている Linux オペレーティングシステムのいずれかに基づいてAMIいる必要があります。最新のリストについては、「[AWS OpsWorks スタックオペレーティングシステム](#)」を参照してください。

詳細については、「[カスタム AMI の使用](#)」を参照してください。

インスタンスがサポートされている標準 から派生AMIするカスタム に基づいている場合AMI、またはインスタンスに非常に最小限のセットアップが含まれている場合は、インスタンスに `libyaml` パッケージをインストールする必要があります。Ubuntu インスタンスの場合、パッケージの名前は `libyaml-0-2` です。Amazon Linux および Red Hat Enterprise Linux インスタンスの場合、パッケージの名前は `libyaml` です。

- インスタンスのインスタンスタイプ (インスタンスサイズと呼ばれることもあります) がサポート対象である必要があります。サポートされているインスタンスタイプはオペレーティングシステムによって異なり、スタックが にあるかどうかによって異なりますVPC。サポートされているインスタンスタイプのリストについては、ターゲットスタックに新しいインスタンスを作成しようとすると AWS OpsWorks、スタックコンソールに表示されるサイズドロップダウンリストの値を表示します。インスタンスタイプが灰色で表示され、ターゲットスタック内に作成できない場合は、そのタイプのインスタンスを登録することもできません。
- インスタンスは `running` の状態である必要があります。
- インスタンスは [Auto Scaling グループ](#) の一部であってはなりません。

詳細については、「[Auto Scaling グループからEC2インスタンスをデタッチする](#)」を参照してください。

- インスタンスはの一部にすることができますが[VPC](#)、スタックVPCと同じに存在し、が AWS OpsWorks スタックで正しく動作するように設定VPCする必要があります。
- [スポットインスタンス](#)はサポートされていません。これは、スポットインスタンスが[自動ヒーリング](#)で機能しないためです。

Amazon EC2インスタンスを登録しても、AWS OpsWorks スタックはインスタンスの[セキュリティグループ](#)やルールを変更しません。インスタンスのセキュリティグループルールが、次の AWS OpsWorks スタックの要件と一致していることを確認してください。

受信ルール

受信ルールでは以下が許可されている必要があります。

- SSH ログイン。
- 適切なレイヤーからのトラフィック。

たとえば、データベースサーバーは通常、スタックのアプリケーションサーバー層からのインバウンドトラフィックを許可します。

- 適切なポートへのトラフィック。

例えば、アプリケーションサーバーインスタンスは通常、ポート 80 (HTTP) および 443 () へのすべてのインバウンドトラフィックを許可しますHTTPS。

送信ルール

送信ルールでは以下が許可されている必要があります。

- インスタンスで実行されているアプリケーションから AWS OpsWorks スタックサービスへのトラフィック。
- を使用してアプリケーションから Amazon S3 AWS などのAWSリソースにアクセスするためのトラフィックAPI。

一般的なアプローチとして、送信ルールを指定せず、アウトバウンドトラフィックに制限を設定しない方法があります。

AWS CLIのインストールおよび設定

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

最初のインスタンスを登録する前に、`register` を実行するコンピュータでのバージョン 1.16.180 AWS CLI 以降を実行している必要があります。インストールの詳細は使用しているワークステーションのオペレーティングシステムによって異なります。のインストールの詳細については AWS CLI、[AWS「コマンドラインインターフェイスのインストール」](#) および [AWS「コマンドラインインターフェイスの設定」](#) を参照してください。起動している AWS CLI のバージョンを確認するには、シェルセッション内に `aws --version` を入力します。

i Note

[AWS Tools for PowerShell](#) には、[Register-OpsInstance](#) `registerAPI` アクションを呼び出す コマンドレットが含まれていますが、代わりに AWS CLI を使用して `register` コマンドを実行することをお勧めします。

適切なアクセス権限で `register` を実行する必要があります。アクセス権限は、IAM ロールを使用することで取得できますが、最善ではないものの、登録するワークステーションまたはインスタンスに、適切なアクセス権限を持つユーザー認証情報をインストール方法でも取得できます。取得後は、追って説明する方法により、取得した認証情報を用いて `register` を実行できます。ユーザーまたはロールに IAM ポリシーをアタッチしてアクセス許可を指定します。では `register`、`AWSOpsWorksRegisterCLI_EC2` ポリシーまたは `AWSOpsWorksRegisterCLI_OnPremises` ポリシーのいずれかを使用します。このポリシーは、Amazon EC2 インスタンスまたはオンプレミスインスタンスをそれぞれ登録するアクセス許可を付与します。

i Note

Amazon EC2 インスタンス `register` で `register` を実行する場合は、理想的には IAM ロールを使用して認証情報を提供する必要があります。既存のインスタンスに IAM ロールをアタッチする方

法の詳細については、「Amazon EC2ユーザーガイド」の「[インスタンスに IAMロールをアタッチする](#)」または「[IAMロールを置き換える](#)」を参照してください。

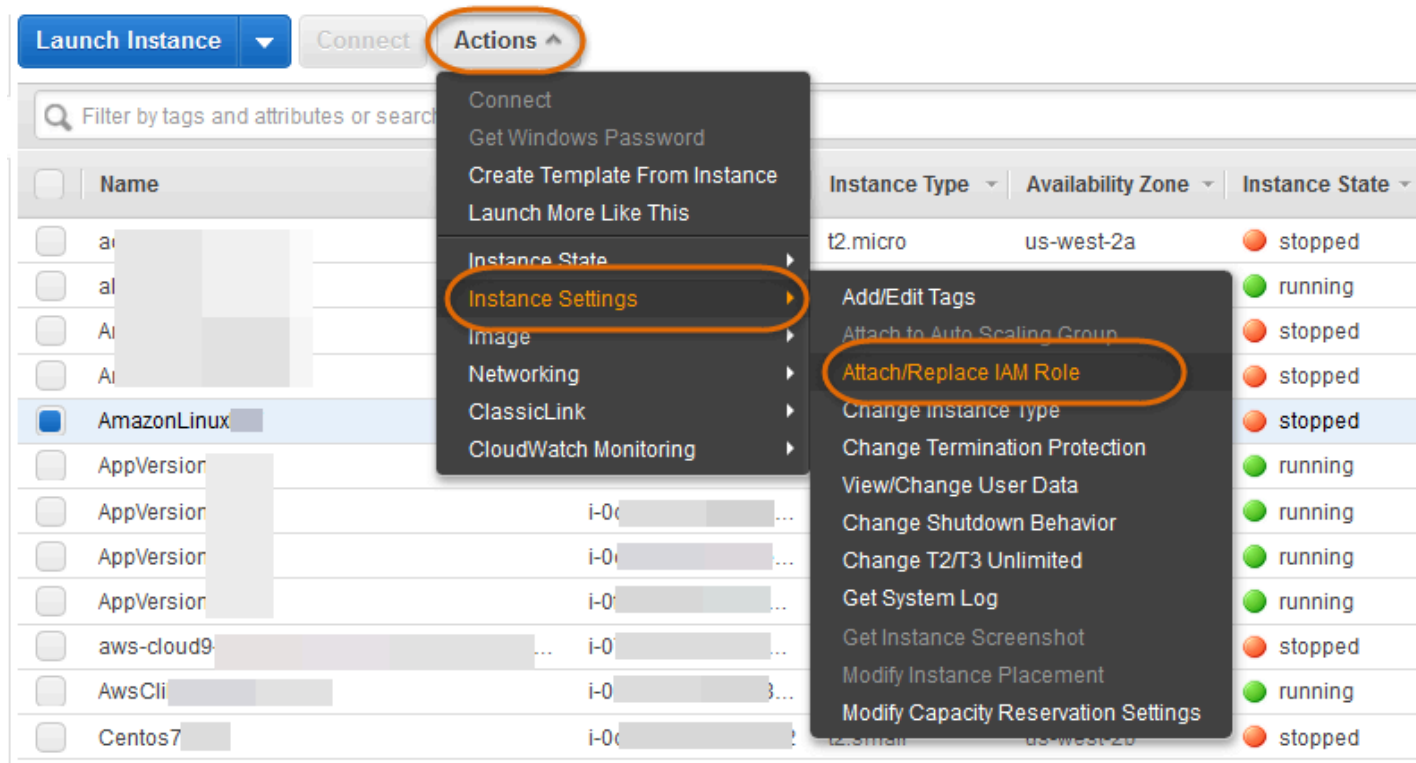
例示のために部分的に抜粋した AWSOpsWorksRegisterCLI_EC2 および AWSOpsWorksRegisterCLI_OnPremises ポリシーについては、[インスタンス登録ポリシー](#)を参照してください。AWS 認証情報の作成と管理の詳細については、[AWS「セキュリティ認証情報」](#)を参照してください。

トピック

- [IAM ロールの使用](#)
- [インストールした認証情報の使用](#)

IAM ロールの使用

登録する Amazon EC2インスタンスから コマンドを実行する場合、に認証情報を提供する際に推奨される方法は、AWSOpsWorksRegisterCLI_EC2ポリシーまたは同等のアクセス許可がアタッチされた IAMロールを使用することregisterです。このアプローチでは、インスタンスに認証情報をインストールせずに済みます。これを行う 1つの方法は、次の図に示すように、EC2コンソールで Attach/Replace IAM Role コマンドを使用することです。



既存のインスタンスに IAM ロールをアタッチする方法の詳細については、「Amazon EC2 ユーザーガイド」の「[インスタンスに IAM ロールをアタッチする](#)」または「[IAM ロールを置き換える](#)」を参照してください。インスタンスプロファイルを使用して起動されたインスタンス (推奨方法) の場合は、`--use-instance-profile` コマンドに `register` スイッチを追加して認証情報を提供します。`--profile` パラメータは使用しないでください。

インスタンスが実行中でありロールが設定されている場合は、そのロールに `AWSOpsWorksRegisterCLI_EC2` ポリシーをアタッチすることによって、必要なアクセス権限を付与できます。ロールは、インスタンスのデフォルトの認証情報を提供します。インスタンスにどの認証情報もインストールしていない場合に限り、`register` は自動的にロールを引き継ぎ、ロールのアクセス権限を使用して実行されます。

Important

インスタンスに認証情報をインストールしないことをお勧めします。セキュリティリスクが発生するだけでなく、インスタンスのロールは、デフォルトの認証情報を見つけるために AWS CLI 使用するデフォルトのプロバイダーチェーンの最後にあります。インストールした認証情報はロールより優先されることがあるため、`register` に必要なアクセス権限が与えられない場合があります。詳細については、「[Getting started with the AWS CLI](#)」を参照してください。

実行中のインスタンスにロールが設定されていない場合は、必要なアクセス権限を持つ認証情報をインスタンスにインストールする必要があります。「[インストールした認証情報の使用](#)」を参照してください。インスタンスプロファイルを使用して起動されたインスタンスを使用することが推奨方法であり、簡単かつエラーが起きにくい方法です。

インストールした認証情報の使用

システムにユーザー認証情報をインストールし、コマンドに提供する方法はいくつかあります AWS CLI。以下に、推奨されなくなったが、EC2 インスタンスプロファイルなしで起動されたインスタンスを登録する場合に使用できるアプローチを示します。アタッチしたポリシーによって必要なアクセス権限が付与されるのであれば、既存のユーザーの認証情報を使用することもできます。認証情報をインストールする他の方法を含む追加情報については、「[設定ファイルと認証情報ファイル](#)」を参照してください。

インストールした認証情報を使用するには

1. [IAM ユーザーを作成し](#)、アクセスキー ID とシークレットアクセスキーを安全な場所に保存します。

⚠ Warning

IAM ユーザーは長期的な認証情報を持っているため、セキュリティ上のリスクがあります。このリスクを軽減するために、これらのユーザーにはタスクの実行に必要な権限のみを付与し、不要になったユーザーを削除することをお勧めします。

2. [AWSOpsWorksRegisterCLI_OnPremises ポリシーをユーザーにアタッチ](#)します。必要であれば、より広範なアクセス権限を付与するポリシーをアタッチできますが、そのポリシーには、AWSOpsWorksRegisterCLI_OnPremises アクセス権限が含まれていなければなりません。
3. システムの credentials ファイル内にユーザーのプロファイルを作成します。このファイルは、`~/.aws/credentials`(Linux、Unix、OS X) または `C:\Users\User_Name\.aws\credentials` (Windows システム) にあります。ファイルには 1 つ以上のプロファイルが次の形式で含まれており、各プロファイルにユーザーのアクセスキー ID とシークレットアクセスキーが含まれています。

```
[profile_name]  
aws_access_key_id = access_key_id  
aws_secret_access_key = secret_access_key
```

前に保存したIAM認証情報を *access_key_id* および *secret_access_key* 値に置き換えます。プロファイル名には任意の名前を指定できますが、名前は一意でなければならないという制約と、デフォルトのプロファイルは default という名前にする必要があるという制約があります。既存のプロファイルに必要なアクセス権限が設定されていれば、既存のプロファイルを使用できます。

4. register コマンドの `--profile` パラメータを使用してプロファイル名を指定します。register コマンドは、関連付けられている認証情報に付与されているアクセス権限を使用して実行されます。

`--profile` を省略することもできます。その場合、register はデフォルトの認証情報を使用して実行されます。これらは必ずしもデフォルトプロファイルの認証情報ではないため、デフォルトの認証情報に必要なアクセス権限があることを確認してください。がデフォルトの認証情報

AWS CLI を決定する方法の詳細については、[AWS「コマンドラインインターフェイスの設定」](#)を参照してください。

インスタンスの登録

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

この機能は Linux スタックでのみサポートされています。

インスタンスを登録するには、ワークステーションまたはインスタンスから AWS CLI `register` コマンドを実行します。このオペレーションを処理する最も簡単な方法は、[AWS OpsWorks スタックコンソール](#) の登録ウィザードを使用することです。このウィザードを使用すると、コマンド文字列を構築する処理が簡略化されます。登録手続きに慣れた後、必要に応じてウィザードを省略し、`register` コマンドを実行することができます。

登録ウィザードを使用して既存のスタックにインスタンスを登録する方法について、次に説明します。

Note

新しいスタックにインスタンスを登録するには、AWS OpsWorks スタックダッシュボードでインスタンスの登録を選択します。これで、新しいスタックを設定する追加のページ以外は既存スタック用のウィザードと同じウィザードが起動します。

登録ウィザードを使用してインスタンスを登録するには

1. [AWS OpsWorks スタックコンソール](#)で、スタックを作成するか、または既存のスタックを開きます。
2. ナビゲーションペインで [Instances] を選択し、[register an instance] を選択します。
3. 「インスタンスタイプの選択」ページで、Amazon EC2インスタンスを登録するか、オンプレミスインスタンスを登録するかを指定します。
 - Amazon EC2インスタンスを登録する場合は、次へ: インスタンスを選択します。
 - オンプレミスインスタンスを登録する場合は、次へ: AWS をインストールするCLIを選択し、ステップ 5 に進みます。
4. Amazon EC2インスタンスを登録する場合は、インスタンスの選択ページを開いて、登録するインスタンスを選択します。AWS OpsWorks スタックは、コマンドの構築に必要な情報を収集します。完了したら、次へ: インストール AWS CLIを選択します。
5. を実行する予定のインスタンスは、 のバージョン 1.16.180 AWS CLI 以降を実行registerしている必要があります。AWS CLIをインストールまたは更新するには、インストールウィザードページに表示されるインストールおよび設定の手順へのリンクを使用します。AWS CLI のインストールを検証したら、登録するインスタンスからコマンドを実行するか、別のワークステーションから実行するかを選択してから、[Next: Register Instances] を選択します。
6. [Register Instances] ページに、選択したオプションが組み込まれた register コマンド文字列のテンプレートが表示されます。例えば、別のワークステーションから Amazon EC2インスタンスを登録する場合、デフォルトのテンプレートは次のようになります。

```
aws opsworks register --infrastructure-class ec2 --region us-west-2
  --stack-id 247be7ea-3551-4177-9524-1ff804f453e3 --ssh-username [username] i-
f1245d10
```

Important

登録プロセス中に作成された IAM ユーザーは、登録されたインスタンスの存続中に必要です。ユーザーを削除すると、AWS OpsWorks スタックエージェントはサービスと通信できなくなります。--use-instance-profile ユーザーが誤って削除された場合に登録済みインスタンスの管理に関する問題が発生するのを回避するために、registerコマンドにパラメータを追加して、代わりにインスタンスの組み込みインスタンスプロファイルを使用します。また、--use-instance-profileパラメータを追加すると、AWS アカウントアクセスキーを 90 日ごとにローテーションするときに

エラーが発生することも防止されます AWS OpsWorks (推奨されるベストプラクティス)。これは、エージェントと必要なIAMユーザーが利用できるアクセスキーが一致しなくなるためです。

SSH キーの使用を Yes に設定すると、AWS OpsWorks スタックは `--ssh-private-key` 数を文字列に追加します。この引数を使用してプライベートSSHキーファイルを指定できます。

Note

パスワードでregisterログオンする場合は、SSHキーを使用する `はい` に設定します。を実行するとregister、パスワードの入力を求められます。

この文字列をテキストエディタにコピーし、必要に応じて編集します。次の点に注意してください。

- 角括弧で囲まれたテキストは、SSHキーファイルの場所など、指定する必要がある情報を表します。
- テンプレートは、デフォルトのAWS認証情報registerで実行していることを前提としています。そうでない場合、コマンド文字列に `--profile` 引数を追加し、使用する認証情報プロフィール名を指定します。

その他のシナリオでは、コマンドをさらに変更することが必要な場合があります。使用できるregisterの引数とコマンド文字列を構築する別の方法については、「[register コマンドの使用](#)」を参照してください。コマンドラインから `aws opsworks help register` を実行してコマンドの説明を表示することもできます。コマンド文字列の例については、「[register コマンドの例](#)」を参照してください。

7. コマンド文字列の編集が完了したら、ワークステーションでターミナルウィンドウを開くか、SSHを使用してインスタンスにログオンし、コマンドを実行します。通常、操作の最初から最後まで5分ほどかかります。その間、インスタンスは [Registering] 状態になります。
8. 操作が終了したら、[Done] を選択します。これでインスタンスは [Registered] 状態になり、スタックの [Instances] ページに未割り当てインスタンスとして表示されます。

register コマンドは次のことを行います。

1. `register` がワークステーションで実行されている場合、コマンドはまず SSHを使用して、登録するインスタンスにログインします。

インスタンスに対して残りの処理が実行されます。コマンドを実行した場所にかかわらず、処理は同じです。

2. Amazon AWS OpsWorks S3 から スタックエージェントパッケージをダウンロードします。
Amazon S3

3. [AWS SDK for Ruby](#) など、エージェントとその依存関係を解凍してインストールします。

4. 以下の項目を作成します。

- 安全な通信を提供するために AWS OpsWorks スタックサービスでエージェントをブートストラップする IAM ユーザー。

ユーザーのアクセス権限により、`opsworks:RegisterInstance` アクションのみが許可され、15 分後に有効期限が切れます。

- スタックの IAMグループ。登録されたインスタンスのユーザーが含まれます。

5. RSA キーペアを作成し、パブリックキーを AWS OpsWorks スタックに送信します。

このキーペアは、エージェントと AWS OpsWorks スタックの間の通信を暗号化するために使用されます。

6. インスタンスを AWS OpsWorks スタックに登録します。スタックは、以下の項目を含む一連の初期 Setup レシピを実行してインスタンスを設定します。

- インスタンスのホストファイルの上書き。

インスタンスに登録することで、AWS OpsWorks スタックにユーザー管理を渡しました。スタックには、SSHログイン許可を制御するための独自のホストファイルが必要です。

- Amazon EC2インスタンスの場合、初期設定には、アタッチされた Amazon EBSボリュームまたは Elastic IP アドレスをスタックに登録することも含まれます。

Amazon EBSボリュームが、`/var/www`やインスタンスのレイヤーによって予約されているマウントポイントなど、予約されたマウントポイントにマウントされていないことを確認する必要があります。スタックリソースの管理の詳細については、「[リソース管理](#)」を参照してください。レイヤーのマウントポイントの詳細については、「[AWS OpsWorks スタックレイヤーリファレンス](#)」を参照してください。

初期セットアップ設定の変更についての詳しい説明は、「[初期セットアップ設定の変更](#)」を参照してください。

Note

初期セットアップでは登録されたインスタスのオペレーティングシステムは更新されません。そのタスクはユーザー自身が処理する必要があります。詳細については、「[セキュリティ更新の管理](#)」を参照してください。

register コマンドの使用**Important**

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

この機能は Linux スタックでのみサポートされています。

インスタスを登録するには、AWS CLI のバージョンが少なくとも 1.16.180 でなければなりません。register コマンドの一般的な構文を次に示します。

```
aws opsworks register \
  [--profile profile_name] \
  [--region region_name] \
  --infrastructure-class instance_type \
  --stack-id stack ID \
  [--local] | [--ssh-private-key key_file --ssh-username username] | [--override-ssh command_string] \
  [--override-hostname hostname] \
  [--debug] \
  [--override-public-ip public IP] \
  [--override-private-ip private IP] \
  ..[--use-instance-profile] \
```

```
[ [IP address] | [hostname] | [instance ID]
```

以下の引数は、すべての AWS CLI コマンドに共通です。

--profile

(オプション) 認証情報のプロファイル名。この引数を省略すると、コマンドはデフォルトの認証情報を使用して実行されます。がデフォルトの認証情報 AWS CLI を決定する方法の詳細については、[AWS「コマンドラインインターフェイスの設定」](#)を参照してください。

--region

(オプション) AWS OpsWorks スタックサービスエンドポイントのリージョン。をスタックのリージョン--regionに設定しないでください。AWS OpsWorks スタックは、スタック ID からスタックのリージョンを自動的に決定します。

Note

デフォルトのリージョンがすでに設定されている場合は、この引数を省略できます。デフォルトのリージョンを指定する方法の詳細については、[AWS「コマンドラインインターフェイスの設定」](#)を参照してください。

Amazon インスタンスEC2とオンプレミスインスタンスの両方に次の引数を使用します。

--infrastructure-class

(必須) このパラメータは、Amazon ec2インスタンスを登録するかオンプレミスインスタンスを登録するかを示すon-premisesために、EC2またはに設定する必要があります。

--stack-id

(必須) インスタンスを登録するスタックの ID。

Note

スタック ID を見つけるには、[Stack (スタック)] ページで [Settings (設定)] を選択します。スタック ID には OpsWorks ID というラベルGUIDが付けられ、のようなですad21bce6-7623-47f1-bf9d-af2affad8907。

SSH ログイン引数

`register` でインスタンスにログインする方法を指定するには、次の引数を使用します。

--local

(オプション) この引数は、コマンドを実行する対象のインスタンスを登録するために使用します。

この場合、`register` でインスタンスにログインする必要はありません。

--ssh-private-key および **--ssh-username**

(オプション) 別のワークステーションからインスタンスを登録する場合に、ユーザー名またはプライベートキーファイルを明示的に指定するには、これらの引数を使用します。

- `--ssh-username` - この引数を使用して SSH ユーザー名を指定します。

`--ssh-username` を省略すると、`ssh` はデフォルトのユーザー名を使用します。

- `--ssh-private-key` - この引数は、プライベートキーファイルを明示的に指定する場合に使用します。

`--ssh-private-key` を省略すると、`ssh` は、デフォルトのプライベートキーの使用などの、パスワード不要の認証方法によるログインを試みます。それらの方法のいずれもサポートされていない場合、`ssh` はパスワードを問い合わせます。が認証 `ssh` を処理する方法の詳細については、[「Secure Shell \(SSH\) Authentication Protocol」](#) を参照してください。

--override-ssh

(オプション) この引数は、別のワークステーションから登録する場合に、カスタムの `ssh` コマンド文字列を指定するときに使用します。`register` コマンドは、このコマンド文字列を使用して、登録されたインスタンスにログインします。

`ssh` の詳細については、「[SSH](#)」を参照してください。

--override-hostname

(オプション) AWS OpsWorks スタックでのみ使用されるインスタンスのホスト名を指定します。デフォルト値はインスタンスのホスト名です。

--debug

(オプション) 登録処理が失敗した場合にデバッグ情報を提供します。トラブルシューティング情報については、「[インスタンス登録の問題のトラブルシューティング](#)」を参照してください。

--use-instance-profile

(オプションですが、Amazon EC2インスタンスには強く推奨されます) `register` コマンドが IAM ユーザーを作成する代わりに、アタッチされたインスタンスプロファイルを使用できるようにします。このパラメータを追加すると、IAM ユーザーが誤って削除された場合に登録済みインスタンスを管理しようとしたときに発生するエラーを回避するうえで役立ちます。

Important

登録プロセス中に作成された IAM ユーザーは、登録されたインスタンスの存続中に必要です。ユーザーを削除すると、AWS OpsWorks スタックエージェントはサービスと通信できなくなります。--use-instance-profile ユーザーが誤って削除された場合に登録済みインスタンスの管理に関する問題が発生するのを回避するために、`register` コマンドにパラメータを追加して、代わりにインスタンスの組み込みインスタンスプロファイルを使用します。また、--use-instance-profile パラメータを追加する AWS OpsWorks と、AWS アカウントアクセスキーを 90 日ごとにローテーションするときエラーが発生することも防止されます (推奨されるベストプラクティス)。エージェントと必要なユーザーが利用できるアクセスキーの不一致を防ぐためです。

Target

(条件付き) ワークステーションからこのコマンドを実行する場合、コマンド文字列の最終値で、次のいずれかの方法で登録ターゲットを指定します。

- インスタンスのパブリック IP アドレス。
- インスタンスのホスト名。
- Amazon EC2 インスタンスの場合は、インスタンス ID。

AWS OpsWorks スタックはインスタンス ID を使用して、インスタンスのパブリック IP アドレスを含むインスタンス設定を取得します。デフォルトでは、AWS OpsWorks スタックはこのアドレスを使用して、インスタンスへのログインに使用する `ssh` コマンド文字列を作成します。プライベート IP アドレスに接続する必要がある場合は、--override-ssh を使用してカスタムコマンド文字列を提供する必要があります。例については、[ワークステーションからのオンプレミスインスタンスの登録](#)を参照してください。

Note

ホスト名を指定した場合、sshはDNSサーバーに依存して名前を特定のインスタンスに解決します。ホスト名が一意かどうか不確かな場合は、sshを使用して、ホスト名が正しいインスタンスに解決されることを確認します。

登録するインスタンスからこのコマンドを実行する場合は、インスタンス識別子を省略し、代わりに `--local` 引数を使用します。

以下の引数は オンプレミス インスタンスの場合のみ使用できます。

--override-public-ip

(オプション) AWS OpsWorks スタックは、指定されたアドレスをインスタンスのパブリック IP アドレスとして表示します。インスタンスのパブリック IP アドレスは変更されません。ただし、インスタンスページでアドレスを選択するなど、ユーザーがコンソールを使用してインスタンスに接続する場合、AWS OpsWorks スタックは指定されたアドレスを使用します。AWS OpsWorks スタックは引数のデフォルト値を自動的に決定します。

--override-private-ip

(オプション) AWS OpsWorks スタックは、指定されたアドレスをインスタンスのプライベート IP アドレスとして表示します。インスタンスのプライベート IP アドレスは変更されません。AWS OpsWorks スタックは引数のデフォルト値を自動的に決定します。

register コマンドの例

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

この機能は Linux スタックでのみサポートされています。

このセクションでは、register コマンド文字列の例を示します。

ワークステーションから Amazon EC2 インスタンスを登録する

次の例では、ワークステーションから Amazon EC2 インスタンスを登録します。コマンド文字列はデフォルトの認証情報を使用し、Amazon インスタンス ID で EC2 インスタンスを識別します。この例は、ec2 を on-premises に変更すれば オンプレミス インスタンスに使用できません。

```
aws opsworks register \  
  --region us-west-2 \  
  --use-instance-profile \  
  --infrastructure-class ec2 \  
  --stack-id ad21bce6-7623-47f1-bf9d-af2affad8907 \  
  --ssh-user-name my-sshusername \  
  --ssh-private-key "./keys/mykeys.pem" \  
  i-2422b9c5
```

ワークステーションからのオンプレミスインスタンスの登録

次の例では、別のワークステーションからオンプレミスインスタンスを登録します。このコマンド文字列はデフォルトの認証情報を使用し、指定した ssh コマンド文字列を使用してインスタンスにログインします。インスタンスにパスワードが必要な場合は、register でパスワードの入力を求められます。を on-premises に変更することで、Amazon EC2 インスタンスの例を使用できます ec2。

```
aws opsworks register \  
  --region us-west-2 \  
  --infrastructure-class on-premises \  
  --stack-id ad21bce6-7623-47f1-bf9d-af2affad8907 \  
  --override-ssh "ssh your-user@192.0.2.0"
```

Note

--override-ssh を使用して任意のカスタムSSHコマンド文字列を指定できます。AWS OpsWorks スタックは、コマンド文字列を構築する代わりに、指定された文字列を使用してインスタンスにログインします。別の例については、「[カスタムSSHコマンド文字列を使用してインスタンスを登録する](#)」を参照してください。

カスタムSSHコマンド文字列を使用してインスタンスを登録する

次の例では、ワークステーションからオンプレミスインスタンスを登録し、--override-ssh引数を使用して、ログインregisterに使用するカスタムSSHコマンドを指定します。この例では、sshpass を使用し、ユーザー名とパスワードを指定してログインしていますが、任意の有効な ssh コマンド文字列を指定できます。

```
aws opsworks register \  
  --region us-west-2 \  
  --infrastructure-class on-premises \  
  --stack-id 2f92ff9d-04f2-4728-879b-f4283b40783c \  
  --override-ssh "sshpass -p 'mypassword' ssh your-user@192.0.2.0"
```

インスタンスからの register 実行によるインスタンスの登録

次の例は、EC2インスタンス自体registerから を実行して Amazon インスタンスを登録する方法を示しています。このコマンド文字列はデフォルトの認証情報からアクセス権限を取得します。この例をオンプレミスインスタンスで使用する場合は、--infrastructure-class に on-premises に変更します。

```
aws opsworks register \  
  --region us-west-2 \  
  --infrastructure-class ec2 \  
  --stack-id ad21bce6-7623-47f1-bf9d-af2affad8907 \  
  --local
```

プライベート IP アドレスを使用するインスタンスの登録

デフォルトでは、register はインスタンスのパブリック IP アドレスを使用してインスタンスにログインします。のプライベートサブネット内のインスタンスなど、VPCプライベート IP アドレスでインスタンスを登録するには、--override-ssh を使用してカスタムsshコマンド文字列を指定する必要があります。

```
aws opsworks register \  
  --region us-west-2 \  
  --infrastructure-class ec2 \  
  --stack-id 2f92ff9d-04f2-4728-879b-f4283b40783c \  
  --override-ssh "ssh -i mykey.pem ec2-user@10.183.201.93" \  
  i-2422b9c5
```

インスタンス登録ポリシー

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWSOpsWorksRegisterCLI_EC2 および AWSOpsWorksRegisterCLI_OnPremises ポリシーは、EC2 および オンプレミス インスタンスをそれぞれ登録するための正しいアクセス許可を提供します。EC2 インスタンスを登録するには AWSOpsWorksRegisterCLI_EC2 を IAM ユーザーに追加しますが、オンプレミス インスタンスを登録する AWSOpsWorksRegisterCLI_OnPremises には をユーザーに追加します。これらのポリシーを使用するには、 のバージョン 1.16.180 AWS CLI 以降を実行している必要があります。

AWSOpsWorksRegisterCLI_EC2 ポリシー

EC2 インスタンスを登録するには、AWSOpsWorksRegisterCLI_EC2 をユーザーに追加します。EC2 インスタンスのみを登録する場合は、このプロファイルを使用する必要があります。このポリシーを使用すると、EC2 インスタンスのインスタンスプロファイルによってアクセス許可が提供されます。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "opsworks:AssignInstance",
```

```

        "opsworks:CreateLayer",
        "opsworks:DeregisterInstance",
        "opsworks:DescribeInstances",
        "opsworks:DescribeStackProvisioningParameters",
        "opsworks:DescribeStacks",
        "opsworks:UnassignInstance"
    ],
    "Resource": [
        "*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:DescribeInstances"
    ],
    "Resource": [
        "*"
    ]
}
]
}

```

AWSOpsWorksRegisterCLI_OnPremises ポリシー

オンプレミス インスタンスを登録する場合には、AWSOpsWorksRegisterCLI_OnPremises をユーザーに追加します。このポリシーには、AttachUserPolicy といった IAM アクセス権限が含まれていますが、この権限でアクセスできるリソースには限りがあります。

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "opsworks:AssignInstance",
                "opsworks:CreateLayer",
                "opsworks:DeregisterInstance",
                "opsworks:DescribeInstances",
                "opsworks:DescribeStackProvisioningParameters",
                "opsworks:DescribeStacks",
                "opsworks:UnassignInstance"
            ],

```

```
    "Resource": [
      "*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeInstances"
    ],
    "Resource": [
      "*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:CreateGroup",
      "iam:AddUserToGroup"
    ],
    "Resource": [
      "arn:aws:iam::*:group/AWS/OpsWorks/OpsWorks-*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:CreateUser",
      "iam:CreateAccessKey"
    ],
    "Resource": [
      "arn:aws:iam::*:user/AWS/OpsWorks/OpsWorks-*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:AttachUserPolicy"
    ],
    "Resource": [
      "arn:aws:iam::*:user/AWS/OpsWorks/OpsWorks-*"
    ],
    "Condition": {
      "ArnEquals": {

```

```
        "iam:PolicyARN": "arn:aws:iam::aws:policy/
AWSOpsWorksInstanceRegistration"
    }
}
}
]
```

(廃止済み) AWSOpsWorksRegisterCLI ポリシー

Important

AWSOpsWorksRegisterCLI ポリシーは廃止されているため、新しいインスタンスの登録には使用できません。すでに登録されているインスタンスを対象とする下位互換作業にのみ使用できます。AWSOpsWorksRegisterCLI ポリシーには、CreateUser、PutUserPolicy、および AddUserToGroup を含む多くの IAM アクセス権限が含まれています。これらは管理者レベルのアクセス権限であるため、AWSOpsWorksRegisterCLI ポリシーは信頼できる管理ユーザーにのみ割り当てる必要があります。

登録されたインスタンスの管理

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

この機能は Linux スタックでのみサポートされています。

インスタンスを登録すると、そのインスタンスは AWS OpsWorks スタックインスタンスになり、AWS OpsWorks スタックで作成されたインスタンスとほぼ同じ方法で管理できます。2 つの主な違いがあります。

- 登録されたインスタンスをレイヤーに関連付ける必要はありません。
- 登録されたインスタンスを登録解除して直接制御に戻すことができます。

インスタンスを登録すると、そのインスタンスは登録済み状態になります。AWS OpsWorks スタックは、登録されたすべてのインスタンスに次の管理機能を提供します。

- **ヘルスチェック** – AWS OpsWorks スタックはエージェントをモニタリングして、インスタンスが引き続き機能するかどうかを評価します。

インスタンスがヘルスチェックに失敗した場合、AWS OpsWorks スタックは登録された [Amazon インスタンスを自動ヒーリング](#) し、登録されたオンプレミスインスタンスのステータスを `connection lost`。EC2

- **CloudWatch モニタリング** – 登録されたインスタンスで CloudWatch モニタリングが有効になっています。

CPU 使用率や使用可能なメモリなどのメトリクスをモニタリングし、メトリクスが指定されたしきい値を超えた場合にオプションで通知を受け取ることができます。

- **ユーザー管理** – AWS OpsWorks スタックでは、インスタンスにアクセスできるユーザーと、実行できるオペレーションを簡単に指定できます。詳細については、「[ユーザー許可の管理](#)」を参照してください。
- **レシピの実行** - [Execute Recipes スタックコマンド](#) を使用して、インスタンスに対して Chef レシピを実行できます。
- **オペレーティングシステム更新** - [Update Dependencies スタックコマンド](#) を使用して、インスタンスのオペレーティングシステムを更新できます。

AWS OpsWorks スタック管理機能を最大限に活用するには、インスタンスをレイヤーに割り当てることができます。詳細については、「[登録されたインスタンスをレイヤーに割り当てる](#)」を参照してください。

スタックが Amazon AWS OpsWorks インスタンスを管理する方法 EC2 と オンプレミスインスタンスを管理する方法には違いがあります。

Amazon EC2 インスタンス

- 登録された Amazon EC2 インスタンスを停止すると、AWS OpsWorks Stacks は instance store-backed インスタンスを終了し、Amazon EBS-backed インスタンスを停止します。

インスタンスはスタックに登録されておりレイヤーに割り当てられているため、必要に応じて再開できます。登録されたインスタンスをスタックから削除するには、[明示的に登録解除する](#)、または[インスタンスを削除する](#) (自動的に登録が解除されます) 必要があります。

- 登録された Amazon EC2 インスタンスを再起動した場合、またはインスタンスが失敗して自動修復された場合、結果は Amazon を使用してインスタンスを停止および再起動した場合と同じです EC2。次の違いがあります:

- Instance store-backed インスタンス – AWS OpsWorks スタックは同じを使用して新しいインスタンスを起動します AMI。

AWS OpsWorks スタックには、ソフトウェアパッケージのインストールなど、登録前にインスタンスで実行したオペレーションの知識がないことに注意してください。AWS OpsWorks スタックで起動時にパッケージをインストールしたり、他の設定タスクを実行したりする場合は、必要なタスクを実行するカスタム Chef レシピを指定し、適切なレイヤーの Setup イベントに割り当てる必要があります。

- Amazon EBS-backed インスタンス – AWS OpsWorks スタックは同じを使用して新しいインスタンスを起動 AMI し、ルートボリュームを再アタッチして、インスタンスを以前の設定に復元します。
- 登録済みの Amazon EC2 インスタンスを登録解除すると、通常の Amazon EC2 インスタンスに戻ります。

オンプレミスインスタンス

- AWS OpsWorks スタックは、登録されたオンプレミスインスタンスを停止または開始することはできません。

登録された オンプレミス インスタンスの割り当てを解除すると、Shutdown イベントがトリガーされます。ただし、そのイベントは割り当てられたレイヤーの Shutdown レシピのみを実行します。サービスのシャットダウンなどのタスクを実行しますが、インスタンスは停止しません。

- AWS OpsWorks スタックは、登録されたオンプレミスインスタンスが失敗しても自動修復できませんが、インスタンスは接続が失われたとマークされます。
- オンプレミスインスタンスは、Elastic Load Balancing、Amazon EBS、または Elastic IP アドレスサービスを使用できません。

登録されたインスタンスをレイヤーに割り当てる

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

この機能は Linux スタックでのみサポートされています。

登録したインスタンスは、1 つ以上のレイヤーに割り当てることができます。インスタンスを未割り当てのままにせずにレイヤーに割り当てる利点は、レイヤーの [ライフサイクルイベント](#) にカスタムレシピを割り当てることができることです。AWS OpsWorks スタックは、そのイベントのレイヤーのレシピの後に、適切な時間にインスタンスを自動的に実行します。

- 登録された任意のインスタンスを [カスタムレイヤー](#) に割り当てることができます。カスタムレイヤーにはいずれのパッケージもインストールしない最小セットのレシピがあるため、インスタンスの既存の設定と競合することはありません。
- オンプレミスインスタンスを AWS OpsWorks スタックの [組み込みレイヤー](#) に割り当てることができます。

すべての組み込みレイヤーには、1 つ以上のパッケージを自動的にインストールするレシピが含まれています。例えば、Java App Server Setup レシピは Apache と Tomcat をインストールします。レイヤーのレシピは、サービスの再起動やアプリケーションのデプロイなど、他の操作を実行する場合があります。オンプレミス インスタンスを組み込みレイヤーに割り当てる前に、現在インスタンスにあるものとは異なるバージョンのアプリケーションサーバーのインストールを試みるなど、レイヤーのレシピが競合しないことを確認する必要があります。詳細については、[レイヤー](#) および [AWS OpsWorks スタックレイヤーリファレンス](#) を参照してください。

登録されたインスタンスをレイヤーに割り当てるには

- まだ追加していない場合は、使用するレイヤーをスタックに追加します。

2. ナビゲーションペインで [インスタンス] をクリックし、インスタンスの [Actions] 列で [assign] をクリックします。
3. 適切なレイヤーを選択して [Save (保存)] を選択します。

インスタンスをレイヤー AWS OpsWorks スタックに割り当てると、以下の処理が実行されます。

- レイヤーの Setup レシピを実行します。
- スタックのリソースにアタッチされた Elastic IP アドレスまたは Amazon EBS ボリュームを追加します。

その後、AWS OpsWorks スタックを使用してこれらのリソースを管理できます。詳細については、「[リソース管理](#)」を参照してください。

完了すると、インスタンスはオンラインステータスになり、スタックに完全に組み込まれます。AWS OpsWorks スタックは、ライフサイクルイベントが発生するたびにレイヤーに割り当てられたレシピを実行します。

登録されたインスタンスの割り当てを解除する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

この機能は Linux スタックでのみサポートされています。

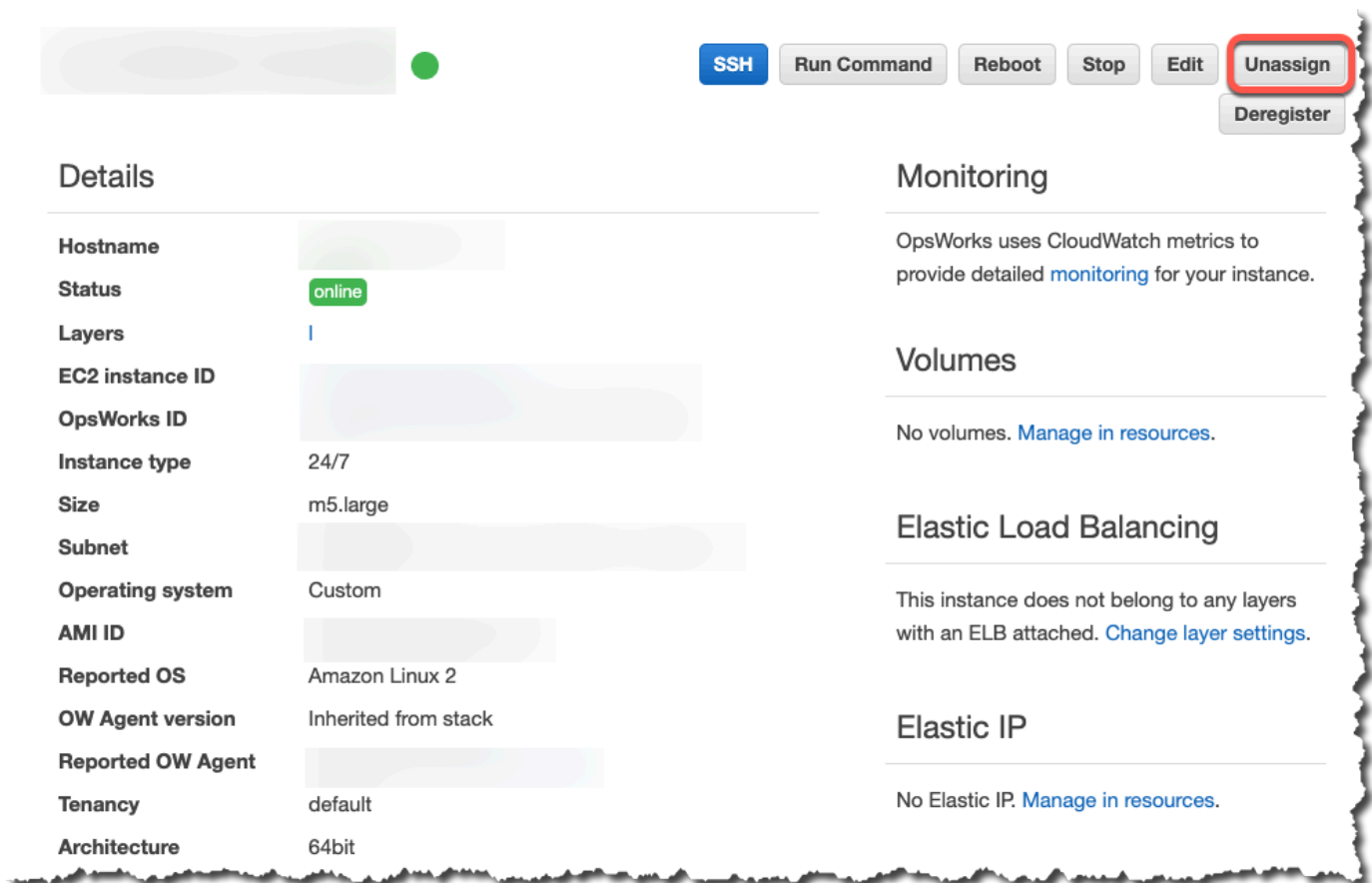
AWS OpsWorks コンソール、または SDK オペレーションを使用して AWS CLI、登録されたインスタンスをレイヤーから割り当て解除できます。

インスタンスの割り当てを解除すると、AWS OpsWorks スタックはインスタンスでレイヤーの Shutdown レシピを実行します。これらのレシピはサービスのシャットダウンなどのタスクを実行し

ますが、インスタンスは停止しません。インスタンスが複数のレイヤーに割り当てられている場合、割り当ての解除はすべてのレイヤーに適用されるため、一部のレイヤーからのみインスタンスの割り当てを解除することはできません。ただし、インスタンスの登録はスタックに残るため、必要に応じて他のレイヤーに割り当てることができます。

コンソールを使用して登録済みインスタンスの割り当てを解除するには

1. ナビゲーションペインで、[インスタンス] を選択します。
2. 割り当てを解除するインスタンスを選択します。
3. インスタンスの「詳細」 ページで、「割り当て解除」 を選択します。



を使用して登録されたインスタンスの割り当てを解除するには AWS CLI

[aws opsworks unassign-instance](#) コマンドを実行して、登録されたインスタンスを、そのインスタンスを使用しているすべての Layer から割り当て解除します。

```
aws opsworks unassign-instance --region region --instance-id instance-id
```

登録されたインスタンスの登録を解除する

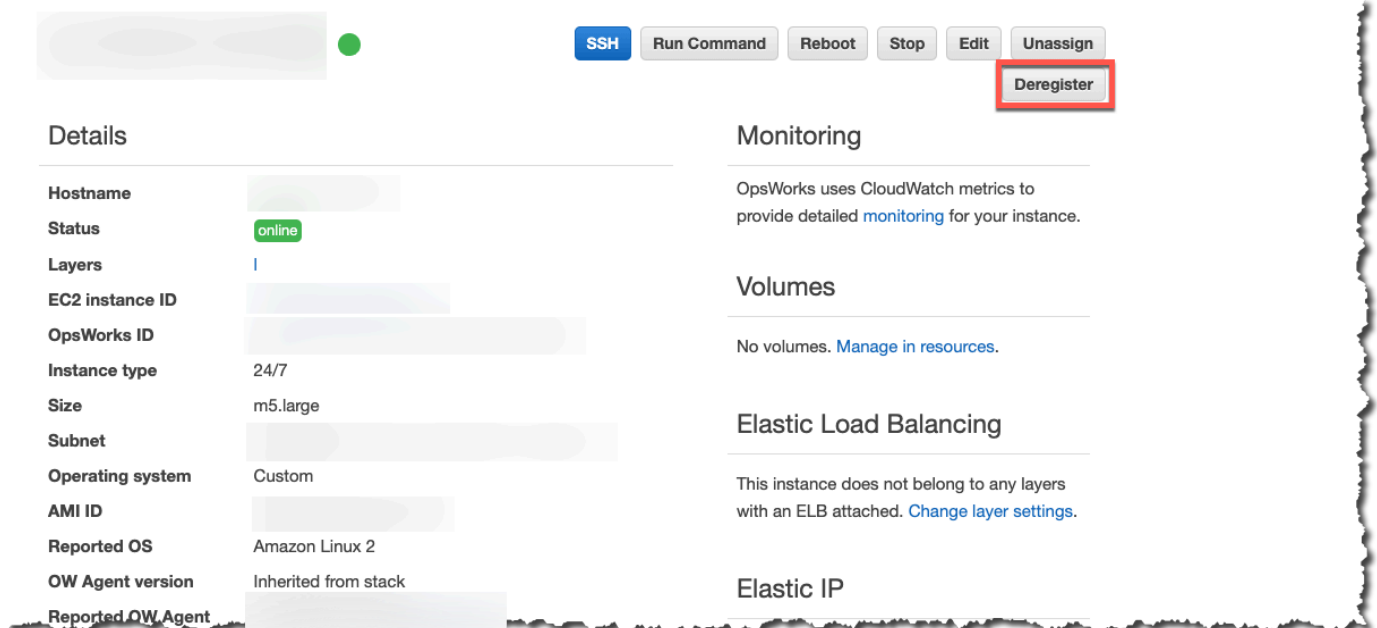
⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks コンソール、または SDK オペレーションを使用して AWS CLI、インスタンスの登録を解除できます。

コンソールを使用してインスタンスを登録解除するには

1. ナビゲーションペインで、[インスタンス] を選択します。
2. 登録解除するインスタンスを選択します。
3. インスタンスの「詳細」ページで「登録解除」を選択します。



を使用してインスタンスの登録を解除するには AWS CLI

[aws opsworks deregister-instance](#) コマンドを実行してスタックからインスタンスを登録解除します。

```
aws opsworks deregister-instance --region region --instance-id instance-id
```

インスタンスの登録を解除すると、AWS OpsWorks Stacks は以下を実行します。

- スタックからインスタンスを削除します。
- 割り当てられたすべてのレイヤーからインスタンスの割り当てを解除します。
- エージェントをシャットダウンし、アンインストールします。
- アタッチされたリソース (Elastic IP アドレスと Amazon EBSボリューム) の登録を解除します。

この手順には、登録前にインスタンスにアタッチされたリソースと、スタックの一部である間に AWS OpsWorks スタックを使用してインスタンスにアタッチしたリソースが含まれます。登録を解除すると、そのリソースはスタックの一部ではなくなりますが、インスタンスにはアタッチされたままになります。

- オンプレミス インスタンスの場合は、料金が発生しなくなります。
- がインスタンス OpsWorks に追加したすべてのタグを削除します。

インスタンスは実行中の状態のままですが、直接制御され、AWS OpsWorks スタックによって管理されなくなります。

Note

コンピュータまたはインスタンスの登録と登録解除は、Linux スタック内でのみ完全にサポートされます。Windows スタックの場合、インスタンスの登録解除は許可されますが、インスタンスから OpsWorks エージェントをアンインストールすることはできません。登録解除では、変更されたすべてのファイルは削除されず、特定のファイルのバックアップされたコピーに完全に戻るわけではありません。このリストは、Chef 11.10 および Chef 12 スタックの両方に適用されます。2つのバージョンの違いを以下に示します。

- `/etc/hosts` は `/var/lib/aws/opsworks/local-mode-cache/backup/etc/` にバックアップされますが、復元はされません。
- `aws` および `opsworks` について、`passwd`、`group`、`shadow files` などのエントリは残ります。
- `/etc/sudoers` には、AWS OpsWorks スタックディレクトリへの参照が含まれていません。

- 以下のファイルはそのままにしても安全です。長期的には、`/var/lib/aws/opsworks` を削除してください。
- `/var/log/aws/opsworks` は、Chef 11.10 のスタックでインスタンスに残ります。
- `/var/lib/aws/opsworks` は、Chef 11.10 と Chef 12 両方のスタックに残ります。
- `/var/chef` は、Chef 12 のスタックでインスタンスに残ります。
- そのまま残されるその他のファイル:
 - `/etc/logrotate.d/opsworks-agent`
 - `/etc/cron.d/opsworks-agent-updater`
 - `/etc/ld.so.conf.d/opsworks-user-space.conf`
 - `/etc/motd.opsworks-static`
 - `/etc/aws/opsworks`
 - `/etc/sudoers.d/opsworks`
 - `/etc/sudoers.d/opsworks-agent`

登録されたインスタンスのライフサイクル

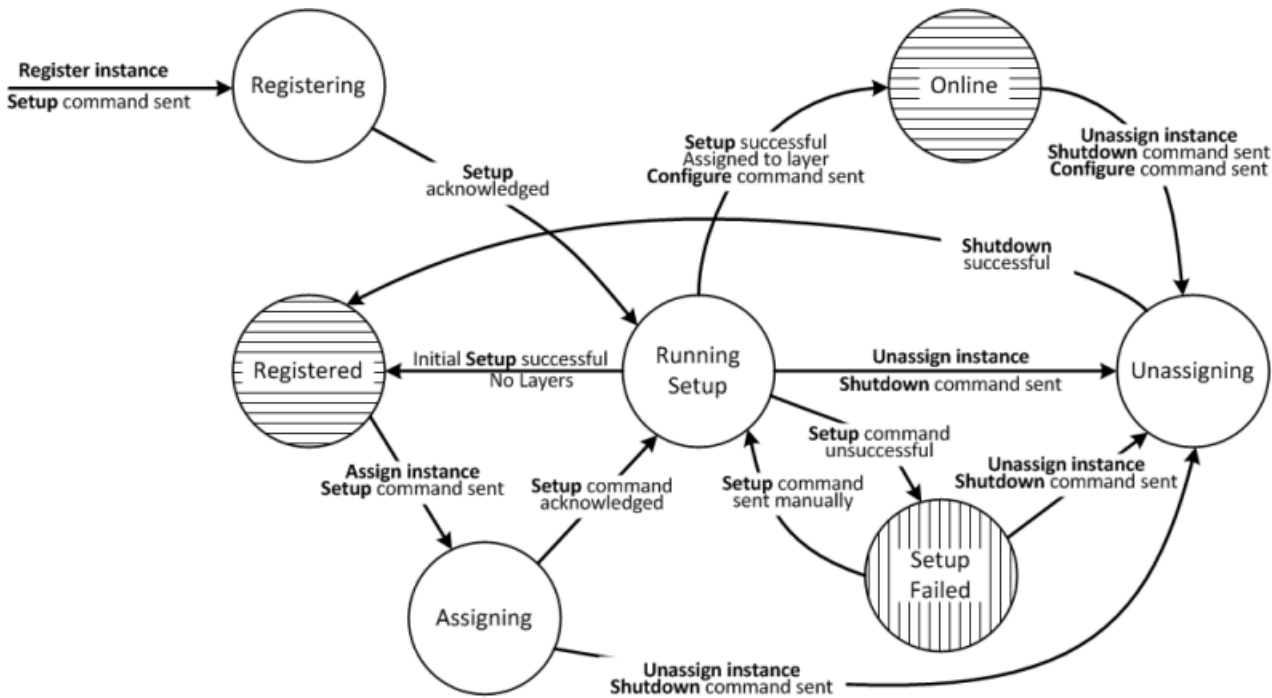
Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

この機能は Linux スタックでのみサポートされています。

登録されたインスタンスのライフサイクルは、エージェントがインストールおよび実行されると開始します。その時点で、AWS OpsWorks スタックにインスタンスを登録するようにスタックに指示します。次の図は、状態とライフサイクルの主要な要素をまとめたものです。



各状態はインスタンスのステータスに対応します。エッジは、次の AWS OpsWorks スタックコマンドのいずれかを表します。詳細については、以降のセクションで説明します。

- 設定 - このコマンドは Setup [ライフサイクル](#) イベントに対応し、インスタンスの Setup レシピを実行します。
- 設定 - このコマンドは Configure ライフサイクルイベントに対応します。

AWS OpsWorks スタックは、インスタンスがオンライン状態に出入りすると、スタック内のすべてのインスタンスでこのイベントをトリガーします。インスタンスでは Configure レシピが実行され、これにより新しいインスタンスに合わせて必要な変更が加えられます。

- シャットダウン - このコマンドは Shutdown ライフサイクルイベントに対応し、インスタンスの Shutdown レシピを実行します。

これらのレシピはサービスのシャットダウンなどのタスクを実行しますが、インスタンスは停止しません。

- 登録解除 - このコマンドはインスタンスの登録を解除しますが、ライフサイクルイベントには対応しません。

Note

この図は簡略化されており、登録解除中状態と削除済み状態は示されていません。インスタンスの登録解除は、図に示したすべての状態で実行できます。登録を解除すると、Deregister コマンドがインスタンスに送信され、インスタンスが登録解除中状態に移行します。

- オンラインインスタンスの登録を解除すると、AWS OpsWorks スタックはスタック内の残りのインスタンスに Configure コマンドを送信して、インスタンスがオフラインになることを通知します。
- Deregister コマンドが認識されると、インスタンスは実行中のまま削除済み状態に移行し、スタックの一部ではなくなります。インスタンスを再度スタックに組み込むには、再登録する必要があります。

トピック

- [を登録する](#)
- [セットアップ実行中](#)
- [登録済み](#)
- [割り当て中](#)
- [オンライン](#)
- [セットアップ失敗](#)
- [割り当て解除中](#)
- [初期セットアップ設定の変更](#)

を登録する

エージェントが登録リクエストを送信すると、AWS OpsWorks スタックは Setup コマンドをインスタンスに送信してインスタンスのライフサイクルを開始し、登録状態にします。インスタンスで Setup コマンドが認識されると、インスタンスは[セットアップ実行中](#)状態に移行します。

セットアップ実行中

セットアップ実行中状態では、インスタンスの Setup レシピが実行されます。セットアップの挙動は、セットアップ前の状態に応じて変化します。

Note

セットアップの実行状態のインスタンスの割り当てを解除すると、AWS OpsWorks スタックは Shutdown コマンドを送信します。このコマンドはインスタンスの Shutdown レシピを実行しますが、インスタンスは停止しません。インスタンスは [割り当て解除中](#) 状態に移行します。

トピック

- [を登録する](#)
- [割り当て中](#)
- [セットアップ失敗](#)

を登録する

登録プロセス中に、セットアップは AWS OpsWorks スタック内の登録されたインスタンスを表すスタックインスタンスを作成し、インスタンスで一連のコア Setup レシピを実行します。

初期セットアップによって実施される主な変更の 1 つは、インスタンスのホストファイルの上書きです。インスタンスを登録することで、ユーザー管理を AWS OpsWorks スタックに引き渡しました。スタックには、SSHログイン許可を制御するための独自のホストファイルが必要です。また、初期セットアップでは複数のファイルが作成および変更されるほか、Ubuntu システムではパッケージのソースが変更されて一連のパッケージがインストールされます。詳細については、「[初期セットアップ設定の変更](#)」を参照してください。

登録中には、前提として作成した IAM ユーザーにアタッチされているアクセス権限の一部である、IAM AttachUserPolicy が呼び出されます。AttachUserPolicy が存在しない場合 (おそらく古いリリースを実行しているため CLI) AWS、プロセスは の呼び出しにフォールバックしませんPutUserPolicy。

Note

一貫性を保つために、AWS OpsWorks スタックはすべてのコア Setup レシピを実行します。ただし、一部のレシピでは、インスタンスが 1 つ以上のレイヤーに割り当てられている場合にのみタスクの一部またはすべてを実行するため、初期セットアップに影響するとは限りません。

- セットアップが成功すると、インスタンスは[登録済み](#)状態に移行します。
- セットアップが失敗すると、インスタンスは[セットアップ失敗](#)状態に移行します。

割り当て中

インスタンスには、少なくとも 1 つの割り当てられたレイヤーがあります。AWS OpsWorks スタックは、レイヤーの Setup [イベントに割り当てたカスタムレシピを含め、各レイヤーの Setup レシピ](#) を実行します。

- セットアップが成功すると、インスタンスがオンライン状態に移行し、AWS OpsWorks スタックによってスタック内のすべてのインスタンスで Configure ライフサイクルイベントがトリガーされ、新しいインスタンスに通知されます。
- セットアップが失敗すると、インスタンスは[セットアップ失敗](#)状態に移行します。

Note

このセットアッププロセスは 2 回目のコア レシピを実行します。ただし、Chef のレシピはべき等であるため、すでに実行されているタスクは繰り返されません。

セットアップ失敗

[割り当て中](#)状態にあるインスタンスのセットアッププロセスが失敗した場合は、インスタンスの Setup レシピを [Setup スタックコマンド](#) を使用して手動で再実行できます。

- セットアップが成功すると、割り当てられているインスタンスは [オンライン](#) 状態に移行し、AWS OpsWorks スタックはそのスタック内のすべてのインスタンスに対して Configure ライフサイクルイベントをトリガーして、新しいインスタンスについて通知します。
- セットアップの試行が失敗すると、インスタンスは[セットアップ失敗](#)状態に戻ります。

登録済み

登録状態のインスタンスはスタックの一部であり、AWS OpsWorks スタックによって管理されますが、レイヤーには割り当てられません。これらのインスタンスは永続的にこの状態を維持できます。

インスタンスを 1 つ以上のレイヤーに割り当てると、AWS OpsWorks スタックは Setup コマンドをインスタンスに送信し、[割り当て中](#)状態に移行します。

割り当て中

インスタンスで Setup コマンドが認識されると、インスタンスは[セットアップ実行中](#)状態に移行します。

割り当て中状態のインスタンスの割り当てを解除すると、AWS OpsWorks スタックはセットアッププロセスを終了し、Shutdown コマンドを送信します。インスタンスは[割り当て解除中](#)状態に移行します。

オンライン

インスタンスは 1 つ以上の レイヤーのメンバーであり、通常の AWS OpsWorks スタックインスタンスと同様に処理されます。このインスタンスは永続的にこの状態を維持できます。

オンライン状態のインスタンスの割り当てを解除すると、AWS OpsWorks スタックはインスタンスに Shutdown コマンドを送信し、残りのスタックのインスタンスに Configure コマンドを送信します。インスタンスは[割り当て解除中](#)状態に移行します。

セットアップ失敗

Setup コマンドが失敗しました。

- [Setup スタックコマンド](#)を再度実行して再試行できます。

インスタンスは[セットアップ実行中](#)状態に戻ります。

- インスタンスの割り当てを解除すると、AWS OpsWorks スタックはインスタンスに Shutdown コマンドを送信します。

インスタンスは[割り当て解除中](#)状態に移行します。

割り当て解除中

Shutdown コマンドを実行すると、インスタンスはすべてのレイヤーから割り当てが解除され、[登録済み](#)状態に戻ります。

Note

インスタンスが複数のレイヤーに割り当てられている場合、割り当ての解除はすべてのレイヤーに適用されます。割り当てられたレイヤーの一部のみの割り当てを解除することはでき

ません。レイヤーの割り当てを変更するには、一度インスタンスの割り当てを解除してから希望のレイヤーに再度割り当てる必要があります。

初期セットアップ設定の変更

初期セットアップでは、登録されたすべてのインスタンスで次のファイルとディレクトリが作成または変更されます。

作成されたファイル

```
/etc/apt/apt.conf.d/99-no-pipelining
/etc/aws/
/etc/init.d/opsworks-agent
/etc/motd
/etc/motd.opsworks-static
/etc/sudoers.d/opsworks
/etc/sudoers.d/opsworks-agent
/etc/sysctl.d/70-opsworks-defaults.conf
/opt/aws/opsworks/
/usr/sbin/opsworks-agent-cli
/var/lib/aws/
/var/log/aws/
/vol/
```

変更されたファイル

```
/etc/apt/apt.conf.d/99-no-pipelining
/etc/crontab
/etc/default/monit
/etc/group
/etc/gshadow
/etc/monit/monitrc
/etc/passwd
/etc/security/limits.conf (removing limits only for EC2 micro instances)
/etc/shadow
/etc/sudoers
```

初期設定では、Amazon EC2マイクロインスタンスにスワップファイルも作成されます。

初期セットアップにより、Ubuntu システムに対して次の変更が加えられます。

パッケージソース

初期セットアップでは、パッケージソースが次のように変更されます。

- `deb http://archive.ubuntu.com/ubuntu/ ${code_name} main universe`

```
To: deb-src http://archive.ubuntu.com/ubuntu/ ${code_name} main
universe
```

- `deb http://archive.ubuntu.com/ubuntu/ ${code_name}-updates main universe`

```
To: deb-src http://archive.ubuntu.com/ubuntu/ ${code_name}-updates main
universe
```

- `deb http://archive.ubuntu.com/ubuntu ${code_name}-security main universe`

```
To: deb-src http://archive.ubuntu.com/ubuntu ${code_name}-security main
universe
```

- `deb http://archive.ubuntu.com/ubuntu/ ${code_name}-updates multiverse`

```
To: deb-src http://archive.ubuntu.com/ubuntu/ ${code_name}-updates
multiverse
```

- `deb http://archive.ubuntu.com/ubuntu ${code_name}-security multiverse`

```
To: deb-src http://archive.ubuntu.com/ubuntu ${code_name}-security
multiverse
```

- `deb http://archive.ubuntu.com/ubuntu/ ${code_name} multiverse`

```
To: deb-src http://archive.ubuntu.com/ubuntu/ ${code_name} multiverse
```

- `deb http://security.ubuntu.com/ubuntu ${code_name}-security multiverse`

```
To: deb-src http://security.ubuntu.com/ubuntu ${code_name}-security
multiverse
```

パッケージ

初期セットアップにより `landscape` がアンインストールされ、次のパッケージがインストールされます。

autofs	libicu-dev	libopenssl-ruby
libssl-dev	libxml2-dev	libxslt-dev
libyaml-dev	monit	ntpd
procps	ruby	ruby-dev
rubygems	screen	sqlite
vim	xfst	

インスタンス設定の編集

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[登録された Amazon Elastic Compute Cloud \(Amazon EC2\) インスタンス](#)などのインスタンス設定を編集できますが、以下の制限があります。

- インスタンスは停止状態である必要があります。

オンラインインスタンスのプロパティは変更できませんが、インスタンスのレイヤーを編集することで、その設定の一部の側面を変更できます。詳細については、「[OpsWorks レイヤーの設定の編集](#)」を参照してください。

- [Availability Zone] や [Scaling Type] などの一部の設定はインスタンスの作成時に決定され、後で変更することはできません。
- Instance store-Backed インスタンスでは変更できても、Amazon Elastic Block Store-Backed インスタンスでは変更できない設定があります。

例えば、Instance store-backed インスタンスのオペレーティングシステムは変更できますが、Amazon EBS-backed インスタンスでは、インスタンスの作成時に指定したオペレーティング

システムを使用する必要があります。インスタンスストレージについては、「[ストレージ](#)」を参照してください。

- デフォルトでは、インスタンスは[スタックのエージェントバージョン](#)の設定を継承します。

OpsWorks エージェントバージョンを使用して、スタックのエージェントバージョン設定を上書きし、インスタンスに特定のエージェントバージョンを指定できます。インスタンスのエージェントバージョンを指定した場合、AWS OpsWorks スタックのエージェントバージョン設定が自動更新であっても、新しいバージョンが使用可能になったときに、スタックはエージェントを自動的に更新しません。インスタンス設定を編集して、インスタンスのエージェントバージョンを手動で更新する必要があります。AWS OpsWorks スタックは、指定されたエージェントバージョンをインスタンスにインストールします。

Note

登録したオンプレミスインスタンスの設定を編集することはできません。

インスタンスの設定を編集するには

1. インスタンスをまだ停止していない場合は、停止します。
2. [Instances] ページで、インスタンス名をクリックして、[Details] ページを表示します。
3. [Edit] をクリックして、編集ページを表示します。
4. 必要に応じてインスタンスの設定を編集します。

[Host name]、[Size]、[SSH key] および [Operating system] 設定の説明については、「[レイヤーへのインスタンスの追加](#)」を参照してください。[Layer] 設定により、レイヤーを追加または削除することができます。インスタンスの現在のレイヤーがレイヤーのリストに続いて表示されます。

- 別のレイヤーを追加するには、リストから選択します。
- レイヤーの 1 つからインスタンスを削除するには、適切なレイヤーの横の [x] をクリックします。

インスタンスは、少なくとも 1 つのレイヤーのメンバーである必要があるため、最後のレイヤーを削除することはできません。

インスタンスを再起動すると、AWS OpsWorks スタックは更新された設定で新しい Amazon EC2 インスタンスを起動します。

AWS OpsWorks スタックインスタンスの削除

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックを使用して、[登録された Amazon EC2 インスタンスを含むインスタンス](#)を停止できます。これにより、EC2 インスタンスは停止しますが、インスタンス自体はスタックに残ります。インスタンスのアクション列で開始をクリックして、を再起動できます。インスタンスが不要になり、スタックから削除するには、それを削除することで、インスタンスがスタックから削除され、関連付けられた Amazon EC2 インスタンスが終了します。インスタンスを削除すると、そのインスタンスに関連するログまたはデータ、Amazon Elastic Block Store (EBS) ボリュームも削除されます。

Important

このトピックは、AWS OpsWorks スタックによって管理される Amazon EC2 インスタンスにのみ適用されます。Amazon EC2 コンソールまたは API を使用して管理するインスタンスの削除方法の詳細については、「[インスタンスの終了](#)」を参照してください。

Note

AWS OpsWorks スタックを使用して、登録されたオンプレミスインスタンスを削除することはできません。

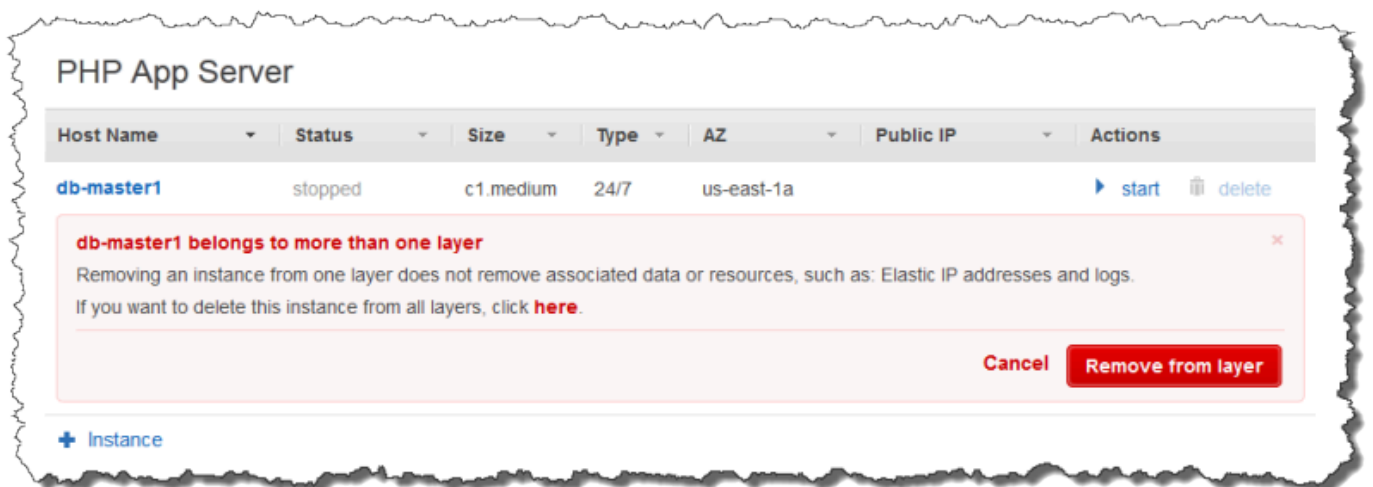
インスタンスが複数のレイヤーに属している場合は、スタックからインスタンスを削除するか、単に特定のレイヤーを削除します。また、「[インスタンス設定の編集](#)」で説明しているように、インスタンスの設定を編集することで、インスタンスからレイヤーを削除することもできます。

⚠ Important

AWS OpsWorks スタックインスタンスは、AWS OpsWorks スタックコンソールまたは API を使用してのみ削除する必要があります。特に、Amazon EC2 アクションは AWS OpsWorks スタックと自動的に同期されないため、Amazon EC2 コンソールまたは API を使用して AWS OpsWorks スタックインスタンスを削除しないでください。例えば、自動ヒーリングを有効にしている場合に Amazon EC2 コンソールを使用してインスタンスを終了すると、AWS OpsWorks スタックはその終了したインスタンスを失敗したインスタンスとして扱い、別の Amazon EC2 インスタンスを起動して置き換えます。詳細については、「[自動ヒーリングの使用](#)」を参照してください。

インスタンスを削除するには

1. [Instances] ページで、該当するレイヤーのインスタンスを探します。インスタンスが実行されている場合は、[アクション] 列で [停止] をクリックします。
2. ステータスが [stopped] に変わったら、[delete] をクリックします。インスタンスが複数のレイヤーのメンバーである場合、レイヤー AWS OpsWorks スタックには次のセクションが表示されます。



- 選択したレイヤーからのみインスタンスを削除するには、[Remove from layer] をクリックします。

インスタンスは、その他のレイヤーではメンバーとして残り、再起動することができます。

- インスタンスをすべてのレイヤーから削除し、スタックから削除するには、[here] をクリックします。

3. スタックからインスタンスを完全に削除することを選択した場合、またはインスタンスが1つのレイヤーのみのメンバーである場合、AWS OpsWorks Stacks は削除の確認を求めます。

[Delete] を選択して確定します。インスタンスをスタックから削除することに加えて、このアクションは、インスタンスに関連付けられているログやデータ、インスタンスにアタッチされているルートボリュームも削除します。すべてのインスタンスボリュームを削除するには、[削除] を選択する前に、インスタンスの EBS ボリュームを削除 (スナップショットは削除されません) します。

SSH を使用した Linux インスタンスへのログイン

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

SSH を使用してオンライン Linux インスタンスにログインするには、組み込み MindTerm クライアント、または PuTTY などのサードパーティークライアントを使用します。SSH は、通常 RSA キーペアを使用して認証を行います。インスタンスにパブリックキーをインストールし、対応するプライベートキーを SSH クライアントに提供します。AWS OpsWorks スタックは、スタックのインスタンスへのパブリックキーのインストールを次のように処理します。

- Amazon Elastic Compute Cloud (Amazon EC2) キーペア – スタックのリージョンに 1 つ以上の Amazon EC2 キーペアがある場合、[スタックのデフォルトの SSH キーペアを指定できます](#)。

オプションで、デフォルトのキーペアを上書きし、インスタンスの作成時に異なるペアを指定できます。いずれの場合も、AWS OpsWorks Stacks は指定されたキーペアのパブリックキーをインスタンスにインストールします。Amazon EC2 のキーペアの作成方法の詳細については、[Amazon EC2 のキーペア](#) を参照してください。

- 個人キーペア – 各ユーザーは AWS OpsWorks スタックで [個人キーペアを登録](#) できます。

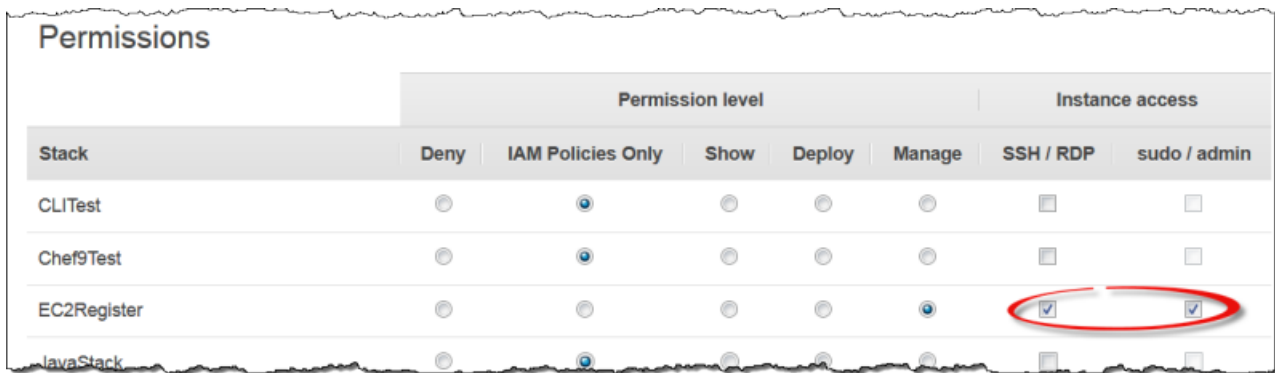
ユーザーまたは管理者はパブリックキーを AWS OpsWorks スタックに登録し、ユーザーはプライベートキーをローカルに保存します。スタックのアクセス権限を設定するとき、管理者はどのユーザーにスタックのインスタンスに対する SSH アクセスを付与するかを指定します。AWS

OpsWorks スタックは、承認された各ユーザーのシステムユーザーをスタックのインスタンスに自動的に作成し、ユーザーの個人用パブリックキーをインストールします。

ユーザーは、SSH クライアントを使用するか、スタックのインスタンスにログインするために個人キーペアを使用するための MindTerm SSH 認証を持っている必要があります。

ユーザーに SSH を許可する

1. AWS OpsWorks スタックナビゲーションペインで、アクセス許可 をクリックします。
2. 必要な権限を付与するために、目的のIAM) ユーザーに SSH/RDP を選択します。例えば、[エージェントCLI](#) コマンドを実行するなど、ユーザーが sudo を使用して権限を昇格させることを許可する場合は、sudo/admin も選択します。



Stack	Permission level					Instance access	
	Deny	IAM Policies Only	Show	Deploy	Manage	SSH / RDP	sudo / admin
CLITest	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>
Chef9Test	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>
EC2Register	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
javaStack	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>

AWS OpsWorks スタックを使用して SSH アクセスを管理する方法の詳細については、「」を参照してください[SSH アクセスの管理](#)。

トピック

- [組み込み MindTerm SSH クライアントの使用](#)
- [サードパーティ SSH クライアントの使用](#)

組み込み MindTerm SSH クライアントの使用

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

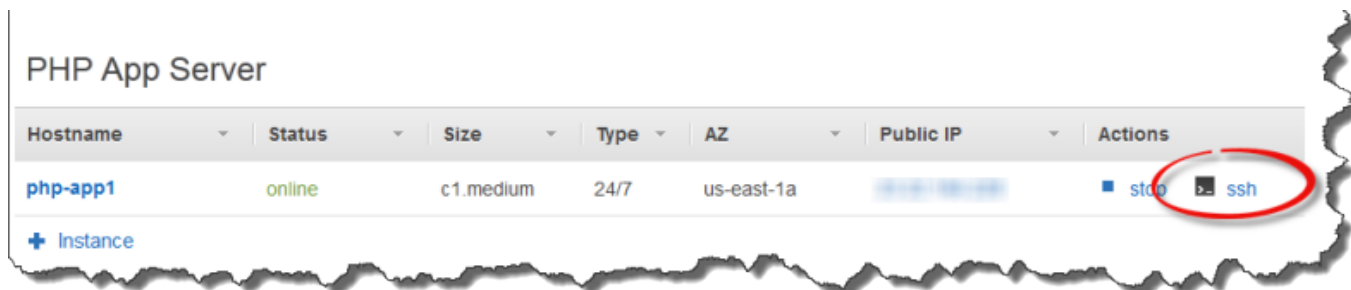
Linux インスタンスにログインする最も簡単な方法は、組み込みの MindTerm SSH クライアントを使用することです。各オンラインインスタンスには、MindTerm クライアントの起動に使用できる ssh アクションが含まれています。

Note

MindTerm クライアントを使用するには、ブラウザで Java が有効になっている必要があります。

MindTerm クライアントでログインするには

1. まだ行っていない場合は、前述のセクションで説明したように、インスタンスに接続する IAM ユーザーに SSH アクセスを許可します。
2. IAM ユーザーとしてログインします。
3. インスタンスページで、該当するインスタンスの [Actions] 列で [ssh] を選択します。



4. プライベートキー で、インスタンスにインストールしたパブリックキーに応じて、IAM ユーザーの個人プライベートキーまたは Amazon EC2 プライベートキーへのパスを入力します。
5. [Launch Mindterm] を選択し、ターミナルウィンドウを使用してインスタンスでコマンドを実行します。

サードパーティ SSH クライアントの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

サードパーティーの SSH クライアント (PuTTY など) を使用して Linux インスタンスに接続することもできます。

サードパーティ SSH クライアントを使用するには

1. 前述のように、AWS OpsWorks スタックが Amazon EC2 パブリックキーまたは IAM ユーザーの個人パブリックキーをインスタンスにインストールしていることを確認します。
2. 詳細ページからインスタンスのパブリック DNS 名またはパブリック IP アドレスを取得します。
3. クライアントにインスタンスのホスト名を指定します。これは、次のようにオペレーティングシステムによって異なります。

- Amazon Linux および Red Hat Enterprise Linux (RHEL)- `ec2-user@DNSName/Address` .
- Ubuntu -。 `ubuntu@DNSName/Address`

`[DNSName/Address]` は、前のステップのパブリック DNS 名または IP アドレスで置き換えます。

4. インストールされたパブリックキーに対応するプライベートキーをクライアントに指定します。インスタンスにインストールされたパブリックキーに応じて、Amazon EC2 プライベートキーまたは IAM ユーザーの個人プライベートキーを使用することができます。

RDP を使用した Windows インスタンスへのログイン

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Windows リモートデスクトッププロトコル (RDP) を使用して、次のようにオンライン Windows インスタンスにログインできます。

- インスタンスには、RDP アクセスを許可するインバウンドルールを持つセキュリティグループが必要です。

セキュリティグループの使用の詳細については、「[セキュリティグループの使用](#)」を参照してください。

- 通常のコピー - AWS OpsWorks スタックは、許可された通常のコピーに、30 分から 12 時間の範囲の期間限定の RDP パスワードを提供します。

承認されるだけでなく、ユーザーは少なくとも [Show アクセス許可レベル](#) を持っているか、アタッチされた AWS Identity and Access Management (IAM) ポリシーで `opsworks:GrantAccess` アクションを許可する必要があります。

- 管理者 - 管理者パスワードを使用してログインできます。時間に制限はありません。

後で説明するように、インスタンスに Amazon Elastic Compute Cloud (Amazon EC2) キーペアを指定している場合、それを使用して管理者パスワードを取得できます。

Note

このトピックでは、Windows リモートデスクトップ接続クライアントを使用して Windows ワークステーションからログインする方法について説明します。さらに、Linux または OS X の使用可能な RDP クライアントのいずれかを使用することもできますが、手順は少し異なる可能性があります。Microsoft Windows Server 2012 R2 と互換性のある RDP クライア

ントの詳細については、「[Microsoft リモートデスクトップクライアント](#)」を参照してください。

トピック

- [RDP アクセスを許可するセキュリティグループの提供](#)
- [通常のユーザーとしてログイン](#)
- [管理者としてログイン](#)

RDP アクセスを許可するセキュリティグループの提供

RDP を使用して Windows インスタンスにログインには、インスタンスのセキュリティグループのインバウンドルールが RDP 接続を許可している必要があります。リージョン内で最初のスタックを作成すると、AWS OpsWorks スタックによって一連のセキュリティグループが作成されます。これには、のような名前の 1 つが含まれます。これは AWS-OpsWorks-RDP-Server、RDP アクセスを許可するために AWS OpsWorks スタックがすべての Windows インスタンスにアタッチします。ただし、デフォルトでは、このセキュリティグループにはルールが存在しないため、インスタンスへの RDP アクセスを許可するインバウンドルールを追加する必要があります。

RDP アクセスを許可するには

1. [Amazon EC2 console](#) (Amazon EC2 コンソール) を開き、スタックのリージョンに設定して、ナビゲーションペインから [Security Groups] (セキュリティグループ) を選択します。
2. AWS-OpsWorks-RDP-Server を選択し、インバウンドタブを選択し、編集 を選択します。
3. [Add Rule] を選択し、次の設定を指定します。
 - タイプ - RDP
 - Source (ソース) – 許可される送信元 IP アドレス。

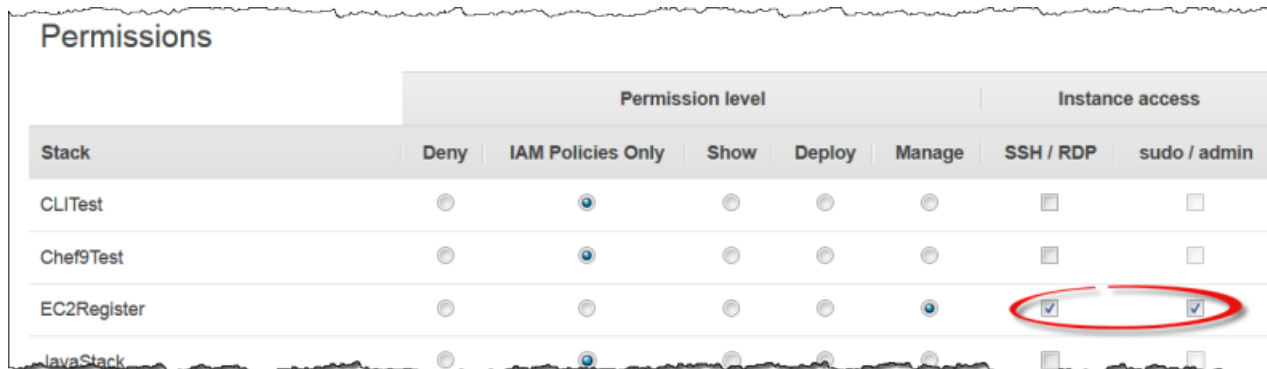
通常は、自身の IP アドレスまたは指定した IP アドレス範囲 (社内の IP アドレス範囲が一般的) へのインバウンド RDP リクエストを許可します。

通常のユーザーとしてログイン

許可されたユーザーは、AWS OpsWorks スタックにより提供される一時パスワードを使用してインスタンスにログインできます。

ユーザーに RDP を許可する

1. AWS OpsWorks スタックナビゲーションペインで、アクセス許可 をクリックします。
2. 必要なアクセス権限を付与するための目的のユーザーに対する [SSH/RDP] チェックボックスを選択します。ユーザーに管理者アクセス権限を付与する場合、[sudo/admin] も選択してください。



Stack	Permission level					Instance access	
	Deny	IAM Policies Only	Show	Deploy	Manage	SSH / RDP	sudo / admin
CLITest	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>
Chef9Test	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>
EC2Register	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
javaStack	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>

許可されたユーザーは、次のようにスタックのどのオンラインインスタンスにもログインできます。

通常のIAM ユーザーとしてログインするには

1. IAM ユーザーとしてログインします。
2. [インスタンス] ページで、適切なインスタンスの [アクション] 列の [rdp] を選択します。
3. セッションの長さ (30 分から 12 時間) を指定し、[Generate Password] を選択します。パスワードは、指定したセッション期間中のみ有効です。
4. [public DNS name]、[username]、[password] の値を記録し、[Acknowledge and close] を選択します。
5. Windows リモートデスクトップ接続クライアントを開いて、[Show Options] を選択し、ステップ 4 で記録して情報から次の内容を入力します。
 - Computer - インスタンスのパブリック DNS 名。

必要に応じて、パブリック IP アドレスを使用することもできます。[Instances] を選択し、インスタンスの [Public IP] 列からアドレスをコピーします。
 - User name - ユーザー名。
6. クライアントにより認証情報の入力を求められた場合、ステップ 4 で保存したパスワードを入力します。

Note

AWS OpsWorks スタックは、オンラインインスタンスに対してのみユーザーパスワードを生成します。インスタンスを起動した後、たとえばカスタム Setup レシピの 1 つが失敗した場合、インスタンスは `setup_failed` 状態になります。インスタンスは AWS OpsWorks スタックに関してオンラインではありませんが、EC2 インスタンスは実行中であり、問題のトラブルシューティングのためにログインすると便利です。この場合、AWS OpsWorks スタックはパスワードを生成しませんが、SSH キーペアをインスタンスに割り当てている場合は、EC2 コンソールまたは CLI を使用してインスタンスの管理者パスワードを取得し、管理者としてログインできます。詳細については、以下のセクションを参照してください。

管理者としてログイン

適切なパスワードを使用して管理者としてインスタンスにログインできます。EC2 キーペアをインスタンスに割り当てた場合、Amazon EC2 はインスタンスの起動時にそのキーペアを使用して管理者パスワードを自動的に作成および暗号化します。その後、EC2 コンソール、API、または CLI でキーペアのプライベートキーを使用して、パスワードを取得および復号できます。

Note

個人 SSH キーペアを使用して管理者パスワードを取得することはできません。EC2 キーペアを使用する必要があります。

EC2 コンソールを使用して管理者パスワードを取得し、インスタンスにログインする方法について次に説明します。コマンドラインツールを希望する場合、AWS CLI [get-password-data](#) コマンドを使用してパスワードを取得することもできます。

管理者としてログインするには

1. インスタンスに EC2 キーペアを指定したことを確認します。スタックを作成するときに [スタックのすべてのインスタンスにデフォルトのキーペアを指定する](#)か、インスタンスを作成するときに [特定のインスタンスのキーペアを指定する](#)ことができます。
2. [EC2 コンソール](#)を開いてスタックのリージョンに設定し、ナビゲーションペインから [Instances] を選択します。
3. インスタンスを選択し、[Connect] を選択して [Get Password] を選択します。

- ワークステーションにおける EC2 キーペアのプライベートキーへのパスを指定し、[Decrypt Password] を選択します。後で使用できるように復号されたパスワードをコピーします。
- Windows リモートデスクトップ接続クライアントを開き、[Show Options] を選択して次の情報を入力します。
 - コンピュータ - インスタンスの詳細ページから取得可能なインスタンスのパブリック DNS 名またはパブリック IP アドレス。
 - User name - Administrator。
- クライアントにより認証情報の入力を求められたら、ステップ 4 の復号されたパスワードを入力してください。

アプリケーション

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックアプリケーションは、アプリケーションサーバーで実行するコードを表します。コード自体は Amazon S3 アーカイブなどのリポジトリにあります。アプリケーションには、適切なアプリケーションサーバーのインスタンスにコードをデプロイするために必要な情報が含まれます。

アプリケーションをデプロイすると、AWS OpsWorks スタックは Deploy イベントをトリガーし、各レイヤーの Deploy レシピを実行します。AWS OpsWorks スタックは、アプリケーションのリポジトリやデータベース接続データなど、アプリケーションのデプロイに必要なすべての情報を含む [スタック設定とデプロイ属性](#) もインストールします。

スタック設定およびデプロイ属性からアプリケーションのデプロイデータを取得してデプロイタスクを処理するカスタムレシピを実装する必要があります。

トピック

- [アプリケーションの追加](#)

- [アプリケーションのデプロイ](#)
- [アプリケーションの編集](#)
- [アプリケーションのデータベースサーバーへの接続](#)
- [環境変数の使用](#)
- [アプリケーションへのデータの引き渡し](#)
- [Git リポジトリの SSH キーの使用](#)
- [カスタムドメインの使用](#)
- [SSL の使用](#)

アプリケーションの追加

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

アプリケーションサーバーにアプリケーションをデプロイする際、最初に行うことは、そのアプリケーションをスタックに追加することです。app はアプリケーションを表し、さまざまなメタデータ (アプリケーションの名前や種類など) と、アプリケーションをサーバーインスタンスにデプロイするために必要な情報 (リポジトリ URL など) を保持します。アプリケーションをスタックに追加するには、Manage 権限が必要です。詳細については、「[ユーザー許可の管理](#)」を参照してください。

Note

このセクションの手順は、Chef 12 以降のスタックに適用されます。Chef 11 スタックのレイヤーにアプリを追加する方法については、「[ステップ 2.4: アプリケーション - Chef 11 を作成してデプロイする](#)」を参照してください。

スタックにアプリケーションを追加するには

1. コードは、Amazon S3 アーカイブ、Gitリポジトリ、Subversion リポジトリ、HTTP アーカイブなど、望ましいリポジトリに加えてください。詳細については、「[Application Source](#)」を参照してください。
2. ナビゲーションペインで [Apps] をクリックします。最初のアプリケーションの場合、[Apps] ページで [Add an app] をクリックします。それ以降のアプリケーションについては、[+App] をクリックします。
3. 次のセクションの説明に従い、[App New] ページを使用してアプリケーションを設定します。

アプリケーションの設定

[Add App] ページは、[Settings]、[Application source]、[Data Sources]、[Environment Variables]、[Add Domains]、[SSL Settings] の各セクションで構成されています。

トピック

- [設定](#)
- [Application Source](#)
- [データソース](#)
- [環境可変](#)
- [ドメインと SSL の設定](#)

設定

名前

UI でアプリを表すために使用されるアプリ名。AWS OpsWorks スタックは、この名前を使用して、内部的に使用されるアプリの短縮名を生成し、[スタック設定およびデプロイ属性](#) でアプリを識別します。アプリケーションをスタックに追加した後、ナビゲーションペインの [Apps] (アプリケーション) をクリックし、アプリケーションの名前をクリックして詳細ページを開くと短縮名を確認できます。

[Document root]

AWS OpsWorks スタックは、アプリケーションの `[:document_root]` 属性の `deploy` 属性にドキュメントルート設定を割り当てます。デフォルト値は、`null`です。デプロイレシピで標準の Chef ノード構文を使い、その値を `deploy` 属性から取得して、特定のコードをサーバー上の適

切な場所にデプロイすることができます。アプリケーションのデプロイ方法の詳細については、「[Deploy レシピ](#)」を参照してください。

Application Source

以下の Git、Amazon S3 bundle、HTTP bundle などのリポジトリタイプからアプリケーションをデプロイできます。いずれのリポジトリタイプも、リポジトリのタイプとリポジトリの URL を指定する必要があります。リポジトリタイプにはそれぞれ固有の要件があります。以下、それらの要件について説明します。

Note

AWS OpsWorks スタックは、標準リポジトリから組み込みサーバーレイヤーにアプリケーションを自動的にデプロイします。Windows スタックの唯一のオプションである Other リポジトリタイプを使用する場合、AWS OpsWorks Stacks はリポジトリ情報をアプリケーションの `deploy` 属性に配置しますが、デプロイタスクを処理するにはカスタムレシピを実装する必要があります。

トピック

- [HTTP アーカイブ](#)
- [Amazon S3 アーカイブ](#)
- [Git リポジトリ](#)
- [その他のリポジトリ](#)

HTTP アーカイブ

パブリックにアクセス可能な HTTP サーバーをリポジトリとして使用するには、次の手順を使用します。

1. アプリケーションのコードとあらゆる関連ファイルを含むフォルダの圧縮アーカイブ (zip、gzip、bzip2、Java WAR、tarball など) を作成します。

Note

AWS OpsWorks スタックは非圧縮 tarball をサポートしていません。

2. アーカイブファイルをサーバーにアップロードします。
3. コンソールでリポジトリを指定するには、リポジトリタイプとして [HTTP Archive] を選択し、URL を入力します。

アーカイブがパスワードで保護されている場合、アプリケーションソース でサインイン認証を指定します。

Amazon S3 アーカイブ

Amazon シンプルストレージサービスバケットをリポジトリとして使用するには:

1. 公開またはプライベートの Amazon S3 バケットを作成します。詳細については、[「Amazon S3 Documentation」](#) (Amazon S3のドキュメント) を参照してください。
2. AWS OpsWorks スタックがプライベートバケットにアクセスするには、少なくとも Amazon S3 バケットに対する読み取り専用権限を持つユーザーである必要があります。アクセスキー ID とシークレットアクセスキーが必要です。詳細については、[AWS Identity and Access Management ドキュメント](#) を参照してください。
3. コードとすべての関連ファイルを単一フォルダに配置し、そのフォルダを圧縮アーカイブ (zip、gzip、bzip2、Java WAR、tarball のいずれか) に保存します。

Note

AWS OpsWorks スタックは非圧縮 tarball をサポートしていません。

4. アーカイブファイルを Amazon S3 バケットにアップロードし、URL を記録します。
5. AWS OpsWorks スタックコンソールでリポジトリを指定するには、リポジトリタイプを S3 Archive に設定し、アーカイブの URL を入力します。プライベートアーカイブの場合、ポリシーがバケットへのアクセス権限を付与する AWS アクセスキー ID とシークレットアクセスキーを提供する必要があります。パブリックアーカイブでは、これらの設定を空白のままにします。

Git リポジトリ

[Git](#) リポジトリはソースコントロールとバージョンングを提供します。AWS OpsWorks スタックは、[GitHub](#)や [Bitbucket](#) などのパブリックにホストされているリポジトリサイトと、プライベートにホストされている Git サーバーをサポートします。アプリケーションも Git サブモジュールも、

[Application Source] (アプリケーションソース) にリポジトリの URL を指定する際の形式は、リポジトリがパブリックであるかプライベートであるかによって異なります。

[Public repository (パブリックリポジトリ)] - HTTPS または Git の読み取り専用プロトコルを使用します。例えば、は、次のいずれかの URL 形式でアクセスできるパブリック GitHub リポジトリ [Chef 11 Linux スタックの使用開始](#) を使用します。

- Git 読み取り専用: **git://github.com/amazonwebservices/opsworks-demo-php-simple-app.git**
- HTTPS: **https://github.com/amazonwebservices/opsworks-demo-php-simple-app.git**

[Private repository (プライベートリポジトリ)] - 次の例に示す SSH の読み取り/書き込み形式を使用します。

- Github リポジトリ: **git@github.com:project/repository**。
- Git サーバー上のリポジトリ: **user@server:project/repository**

[Source Control (ソースコントロール)] で [Git (Git)] を選択すると、次の 2 つの追加オプション設定が表示されます。

[Repository SSH key]

プライベート Git リポジトリにアクセスするには、デプロイ SSH キーを指定する必要があります。このフィールドにはプライベートキーが必要となり、パブリックキーは Git リポジトリに割り当てられます。Git サブモジュールの場合、指定するキーには、それらのサブモジュールへのアクセス権が必要です。詳細については、「[Git リポジトリの SSH キーの使用](#)」を参照してください。

Important

デプロイ SSH キーはパスワードを要求できません。AWS OpsWorks スタックにはパスワードを渡す方法がありません。

[Branch/Revision]

リポジトリに複数のブランチがある場合、AWS OpsWorks スタックはデフォルトでマスターブランチをダウンロードします。特定のブランチを指定するには、ブランチ名、SHA1 ハッシュ、タグ名のいずれかを入力します。特定のコミットを指定するには、40 桁の 16 進数コミット ID を入力します。

その他のリポジトリ

標準のリポジトリでは要件を満たすことができない場合、他のリポジトリ ([Bazaar](#) など) を使用することができます。ただし、AWS OpsWorks スタックはそのようなリポジトリからアプリケーションを自動的にデプロイしません。デプロイプロセスを処理するカスタムレシピを独自に実装し、それらを適切なレイヤーの Deploy イベントに割り当てる必要があります。Deploy レシピの実装方法の例については、「[Deploy レシピ](#)」を参照してください。

データソース

このセクションはアプリケーションにデータベースをアタッチします。次のオプションがあります。

- RDS - スタックの [Amazon RDS サービスレイヤー](#) の 1 つをアタッチします。
- [None (なし)] - データベースサーバーをアタッチしません。

[RDS] を選択する場合、以下の情報を指定する必要があります。

Database instance

このリストには、すべての Amazon RDS サービスレイヤーが含まれています。加えて、次のいずれかを選択することができます。

(必須) アプリケーションにアタッチするデータベースサーバーを指定します。リストの内容は、データソースによって異なります。

- [RDS (RDS)] - スタックの Amazon RDS サービスレイヤーのリスト。

データベース名

(オプション) データベース名を指定します。

- Amazon RDS レイヤー - Amazon RDS インスタンスに対して指定したデータベース名を入力します。

データベース名は、[\[Amazon RDS console \(Amazon RDSコンソール\)\]](#) から取得できます。

データベースがアタッチされたアプリケーションをデプロイすると、AWS OpsWorks スタックはデータベースインスタンスの接続をアプリケーションの [deploy属性](#) に追加します。

カスタムレシピを書き込んで deploy 属性から情報を取得し、アプリケーションからアクセスすることができるファイルに配置できます。これは、Other アプリケーションタイプにデータベース接続情報を提供する唯一のオプションです。

データベース接続を処理する方法の詳細については、「[データベースへの接続](#)」を参照してください。

データベースサーバーをアプリケーションからデタッチするには、[アプリケーションの設定を編集](#)して、異なるデータベースサーバーを指定するか、サーバーの指定を削除します。

環境可変

各アプリケーションに対し、アプリケーションに固有の一連の環境変数を指定できます。たとえば、2つのアプリケーションがある場合、最初のアプリケーションに定義した環境変数を、もう1つのアプリケーションに定義することはできません。その逆も同様です。また、複数のアプリケーションに同じ環境変数を定義し、アプリケーションごとに異なる値を割り当てることもできます。

Note

環境変数の数に、特定の制限はありません。ただし、変数の名前、値、および保護されたフラグ値を含む、関連するデータ構造のサイズは、20 KB を超えることはできません。この制限は、すべてではないものの、ほとんどのユースケースに適合します。これを超えると、サービスエラー (コンソール) または例外 (API) が発生し、"Environment: is too large (maximum is 20KB)" というメッセージが表示されます。

AWS OpsWorks スタックは、変数を属性としてアプリケーションの [deploy属性](#) に保存します。標準の Chef ノード構文を使用して、カスタムレシピにこれらの値を取得することができます。アプリケーションの環境変数にアクセスする方法の例については [環境変数の使用](#) を参照してください。

キー

変数名。最大 64 個の大文字および小文字の文字、数字、下線 (_) を含めることができますが、冒頭には文字または下線を使用する必要があります。

値

変数値。最大 256 文字を含めることができますが、すべて表示可能な文字である必要があります。

Protected value

値を保護するかどうかを指定します。この設定では、パスワードなどの機密情報を非表示にすることができます。変数に [Protected value] を設定した場合、アプリケーションの作成後は次のようになります。

- アプリケーションの詳細ページには値が表示されず、変数名だけが表示されます。
- アプリケーションを編集するアクセス許可がある場合、[Update value] をクリックして新しい値を指定することはできますが、古い値を確認したり、編集したりすることはできません。

Note

Chef のデプロイログには、環境変数が含まれることがあります。これは、保護された変数がコンソールに表示される場合があることを意味します。保護された変数がコンソールに表示されないようにするには、コンソールに表示したくない保護された変数のストレージとして Amazon S3 バケットを使用すること推奨します。この目的で S3 バケットを使用する例は、このガイドの「[Amazon S3 バケットの使用](#)」を参照してください。

ドメインと SSL の設定

その他のアプリタイプでは、AWS OpsWorks スタックはアプリのdeploy属性に設定を追加します。レシピは、それらの属性からデータを取得し、必要に応じてサーバーを設定できます。

Domain Settings

このセクションには、ドメインを指定するためのオプションの [Add Domains] フィールドがあります。詳細については、「[カスタムドメインの使用](#)」を参照してください。

SSL Settings

このセクションの [SSL Support] トグルを使用して、SSL の有効と無効を切り替えることができます。[Yes] をクリックした場合は、SSL 証明書の情報を指定する必要があります。詳細については、「[SSL の使用](#)」を参照してください。

アプリケーションのデプロイ

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

デプロイの主な目的は、アプリケーションサーバーインスタンスにアプリケーションコードと関連ファイルをデプロイすることです。デプロイ操作は、インスタンスのレイヤーによって決まる各インスタンスのデプロイレシピによって処理されます。




インスタンスを起動すると、Setup レシピが完了すると、AWS OpsWorks Stacks はインスタンスの Deploy レシピを自動的に実行します。ただし、アプリケーションを追加または変更するときには、オンラインインスタンスすべてに手動でデプロイする必要があります。アプリケーションをデプロイするには、Manage 権限または Deploy 権限が必要です。詳細については、「[ユーザー許可の管理](#)」を参照してください。

アプリケーションをデプロイするには

1. [Apps] ページで、アプリケーションの [deploy] アクションをクリックします。

Apps

An app represents code stored in a repository that you want to install on application server instances. When you deploy the app, OpsWorks downloads the code from the repository to the specified server instances. [Learn more.](#)

Name	Type	Last deployment	Actions
SimplePHP	PHP		 deploy  edit  delete
+ App			

📘 Note

ナビゲーションペインの [Deployments] をクリックしてアプリケーションをデプロイすることもできます。[Deployments & Commands] ページで、[Deploy an app] をクリック

します。この操作を行う場合、デプロイするアプリケーションを選択することもできます。

2. 次を指定します:

- (必須) [Command:] を [deploy] に設定します (まだ選択されていない場合)。
- (オプション) コメントを入力します。

3. Advanced >> をクリックしてカスタム JSON を指定します。AWS OpsWorks スタックは、一連の[スタック設定とデプロイ属性](#)をノードオブジェクトに追加します。deploy 属性にはデプロイの詳細が含まれており、Deploy レシピでこれらの属性を使用して、インストールと設定を処理できます。Linux スタックでは、カスタム JSON フィールドを使用して[デフォルトの AWS OpsWorks スタック設定を上書き](#)したり、カスタム設定をカスタムレシピに渡すことができます。カスタム JSON の使用方法の詳細については、「[カスタムの使用 JSON](#)」を参照してください。

Note

カスタム JSON をここで指定する場合は、このデプロイのみのスタック設定およびデプロイ属性に追加されます。カスタム JSON を永続的に追加する場合は、[スタックに追加](#)する必要があります。カスタム JSON は 120 KB に制限されています。さらに容量が必要な場合は、Amazon S3 でデータの一部を保存することをお勧めします。カスタムレシピで AWS CLI または [AWS SDK for Ruby](#) を使用して、バケットからインスタンスにデータをダウンロードできます。例については、[SDK for Ruby の使用](#)を参照してください。

4. [Instances] で、[Advanced >>] をクリックし、デプロイコマンドを実行するインスタスを指定します。

デプロイコマンドは Deploy イベントをトリガーし、これによって選択したインスタンスでデプロイレシピが実行されます。関連付けられたアプリケーションサーバーのデプロイレシピは、リポジトリからコードと関連ファイルをダウンロードし、インスタンスにインストールします。したがって、ユーザーは通常、関連付けられたアプリケーションサーバーのインスタンスをすべて選択します。ただし、他のインスタスタイプでは、新しいアプリケーションに対応するように設定の変更を要求する場合があります。そのため、通常、それらのインスタンスでもデプロイレシピを実行することをお勧めします。それらのレシピは、必要に応じて設定を更新しますが、アプリケーションのファイルはインストールしません。recipe の詳細については、「[クックブックとレシピ](#)」を参照してください。

- [Deploy] をクリックして、指定したインスタンスでデプロイレシピを実行します。これにより、[Deployment] ページが表示されます。プロセスが完了すると、AWS OpsWorks Stacks はデプロイが成功したことを示す緑色のチェックでアプリをマークします。デプロイが失敗した場合、AWS OpsWorks スタックはアプリに赤い X をマークします。その場合は、デプロイページに移動し、デプロイログを調べて詳細を確認できます。

Deployment **PHPTestApp - deploy**

[Repeat](#)

Status **successful** User

Created at 2017-04-11 18:59:10 UTC

Completed at 2017-04-11 18:59:59 UTC

Duration 00:00:49

Hostname	SSH	Layers	Duration	Log
✓ app1	ssh	MyLayer	00:00:49	show

Note

JPS アプリケーションの更新プログラムをデプロイする際に、Tomcat は更新プログラムを認識せず、アプリケーションの既存のバージョンを継続して実行する可能性があります。これが発生するのは、JSP ページのみが含まれている .zip ファイルとしてアプリケーションをデプロイする場合などです。Tomcat が、デプロイされた最新バージョンを確実に実行するようにするには、プロジェクトのルートディレクトリに、web.xml ファイルを含む WEB-INF ディレクトリを含める必要があります。web.xml ファイルはさまざまなコンテンツを含むことができますが、Tomcat が確実に更新プログラムを認識し、現在デプロイされているアプリケーションのバージョンを実行するようにするには、次のコンテンツで十分です。各更新プログラムのバージョンを変更する必要はありません。Tomcat は、バージョンが変更されていない場合でも更新プログラムを認識します。

```
<context-param>
  <param-name>appVersion</param-name>
  <param-value>0.1</param-value>
</context-param>
```

他のデプロイコマンド

[Deploy app] ページには、アプリケーションと関連サーバーを管理するためのその他のコマンドがいくつか含まれています。次のコマンドのうち、Chef 12 スタックのアプリで利用できるのは Undeploy のみです。

Undeploy

Undeploy [ライフサイクルイベント](#) をトリガーし、指定されたインスタンスからアプリケーションのすべてのバージョンを削除するデプロイ解除レシピを実行します。

ロールバック

以前にデプロイされたアプリケーションのバージョンを復元します。たとえば、アプリケーションを 3 回デプロイした後、[Rollback] を実行した場合、サーバーは 2 回目のデプロイからアプリケーションを提供します。再度 [Rollback] を実行すると、サーバーは最初のデプロイからアプリケーションを提供します。デフォルトでは、AWS OpsWorks スタックには最新の 5 つのデプロイが保存されるため、最大 4 つのバージョンをロールバックできます。保存されているバージョンの数を越えた場合、このコマンドは失敗し、最も古いバージョンのままになります。このコマンドは、Chef 12 スタックでは使用できません。

Start Web Server

指定されたインスタンスでアプリケーションサーバーを起動するレシピを実行します。このコマンドは、Chef 12 スタックでは使用できません。

Stop Web Server

指定されたインスタンスでアプリケーションサーバーを停止するレシピを実行します。このコマンドは、Chef 12 スタックでは使用できません。

Restart Web Server

指定されたインスタンスでアプリケーションサーバーを再起動するレシピを実行します。このコマンドは、Chef 12 スタックでは使用できません。

Important

[Start Web Server]、[Stop Web Server]、[Restart Web Server]、および [Rollback] は、基本的には [\[Execute Recipes\] スタックコマンド](#) のカスタマイズされたバージョンです。これらのコマンドは、指定されたインスタンスでタスクを実行する一連のレシピを実行します。

- これらのコマンドはライフサイクルイベントをトリガーしないため、カスタムコードを実行するためにフックすることはできません。
- これらのコマンドは、組み込みの [\[application server layers\]](#) (アプリケーションサーバーレイヤー) に対してのみ機能します。

特に、これらのコマンドはカスタムレイヤーに影響を及ぼしません。アプリケーションサーバーがサポートされている場合でも同様です。カスタムレイヤーでサーバーを起動、停止、または再起動するには、これらのタスクを実行するカスタムレシピを実装し、[\[Execute Recipes stack command\]](#) (レシピスタックコマンドの実行) を使用してカスタムレシピを実行する必要があります。カスタムレシピを実装およびインストールする方法の詳細については、「[クックブックとレシピ](#)」を参照してください。

アプリケーションの編集

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

アプリケーションを編集することで、設定を更新できます。例えば、新しいバージョンをデプロイする準備ができたなら、アプリケーションの AWS OpsWorks スタック設定を編集して新しいリポジトリブランチを使用できます。アプリケーションの設定を編集するには、Manage または Deploy 権限が必要です。詳細については、「[ユーザー許可の管理](#)」を参照してください。

アプリケーションを編集するには

1. [Apps] (アプリケーション) ページで、詳細ページを開くアプリケーションの名前をクリックします。
2. [Edit] (編集) をクリックして、アプリケーションの設定を変更します。
 - アプリの名前を変更すると、AWS OpsWorks Stacks は新しい名前を使用してコンソールでアプリを識別します。

名前を変更しても、関連する短縮名は変更されません。短縮名はアプリケーションがスタックに追加された時に設定され、その後に修正はできません。

- 保護された環境変数を指定した場合は、値を表示または変更することはできません。ただし、[Update value] (値を更新) をクリックして新しい値を指定できます。
3. [Save] (保存) をクリックして新規設定を保存し、[Deploy App] (デプロイアプリケーション) をクリックしてアプリケーションをデプロイします。

アプリケーションを編集すると、AWS OpsWorks スタックの設定は変更されますが、スタックのインスタンスには影響しません。初めて [\[deploy an app\]](#) (アプリケーションをデプロイ) すると、デプロイレシピによりアプリケーションサーバーのインスタンスにコードと関連ファイルがダウンロードされます。それにより、ローカルコピーが実行されます。リポジトリでアプリを変更したり、その他の設定を変更したりする場合は、次のようにアプリをデプロイして、アプリサーバーインスタンスに更新をインストールする必要があります。AWS OpsWorks スタックは、起動時に現在のアプリバージョンを新しいインスタンスに自動的にデプロイします。ただし、既存のインスタンスは事情が異なります。

- AWS OpsWorks スタックは、起動時に現在のアプリケーションバージョンを新しいインスタンスに自動的にデプロイします。
- AWS OpsWorks スタックは、再起動時に、[ロードベースおよび時間ベースのインスタンスを含むオフラインインスタンス](#)に最新のアプリケーションバージョンを自動的にデプロイします。
- 更新されたアプリケーションをオンラインインスタンスに手動でデプロイする必要があります。

アプリケーションのデプロイ方法の詳細については、「[アプリケーションのデプロイ](#)」を参照してください。

アプリケーションのデータベースサーバーへの接続

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[\[create the app \(アプリケーションを作成する\)\]](#) とき、または後で [\[editing the app \(アプリケーションを編集する\)\]](#) ことで、Amazon RDS データベースサーバーをアプリケーションと関連付けることができます。その後、アプリケーションはデータベース接続情報 (ユーザー名、パスワードなど) を使用してデータベースサーバーに接続することができます。[アプリケーションをデプロイすると、Stacks](#) は 2 つの方法でこの情報をアプリケーションに提供します。AWS OpsWorks

- Linux スタックでは、AWS OpsWorks スタックにより、各組み込みアプリケーションサーバーインスタンスに、アプリケーションがデータベースサーバーへの接続に使用できる接続データを含むファイルが作成されます。
- AWS OpsWorks スタックには、各インスタンスにインストールされている [スタック設定とデプロイ属性](#) の接続情報が含まれます。

カスタムレシピを実装してこれらの属性から接続情報を取得し、任意の形式でファイルに配置することができます。詳細については、「[アプリケーションへのデータの引き渡し](#)」を参照してください。

Important

Linux スタックでは、Amazon RDS サービス レイヤーをアプリケーションと関連付ける場合は、次の手順に従って、関連付けたアプリケーションサーバーレイヤーに適切なドライバパッケージを追加する必要があります。

1. ナビゲーションペインで [Layers] (レイヤー) をクリックし、アプリケーションサーバーの [Recipes] (レシピ) タブを開きます。
2. [Edit] をクリックして適切なドライバパッケージを [OS Packages] に追加します。たとえば、レイヤーに Amazon Linux のインスタンスが含まれている場合は [mysql] を、レイヤーに Ubuntu インスタンスが含まれている場合は [mysql-client] を指定する必要があります。
3. 変更を保存してアプリケーションを再デプロイします。

カスタムレシピの使用

アプリケーションの [deploy 属性](#) から接続データを取得するカスタムレシピを実装し、YAML ファイルなど、アプリケーションが読み取ることができる形式で保存できます。

[アプリケーションを作成する](#)際、または後で[アプリケーションを編集する](#)際に、データベースサーバーをアプリケーションにアタッチします。アプリケーションをデプロイすると、AWS OpsWorks スタックは、データベース接続情報を含む[スタック設定とデプロイ属性](#)を各インスタンスにインストールします。その後、アプリケーションは適切な属性を取得できます。詳細は、Linux スタックと Windows スタックのどちらを使用しているかによって異なります。

Linux スタックのデータベースサーバーへの接続

Linux スタックでは、[スタック設定およびデプロイ属性](#)の `deploy` 名前空間に、デプロイされた各アプリケーションの属性が含まれており、アプリケーションの短縮名が付いています。データベースサーバーをアプリケーションにアタッチすると、AWS OpsWorks Stacks はアプリケーションの `[:database]` 属性に接続情報を入力し、後続のデプロイごとにスタックのインスタンスにインストールします。属性値はユーザーが指定する値または AWS OpsWorks スタックにより生成される値です。

Note

AWS OpsWorks スタックを使用すると、データベースサーバーを複数のアプリケーションにアタッチできますが、各アプリケーションにはアタッチできるデータベースサーバーが 1 つだけあります。アプリケーションを複数のデータベースサーバーに接続する場合は、サーバーの 1 つをアプリケーションにアタッチし、アプリケーションの `deploy` 属性の情報を使用してそのサーバーに接続します。カスタム JSON を使用して、他のデータベースサーバーの接続情報をアプリケーションに渡します。詳細については、「[アプリケーションへのデータの引き渡し](#)」を参照してください。

アプリケーションはインスタンスの `deploy` 属性の接続情報を使用してデータベースに接続します。ただし、アプリケーションはその情報に直接アクセスすることはできません。レシピだけが `deploy` 属性にアクセスできます。カスタムレシピを実装し、`deploy` 属性から接続情報を取得してアプリケーションで読み取り可能なファイルに保存すれば、この問題は解決します。

環境変数の使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

このトピックの推奨事項は、Chef 11.10 およびそれ以前のバージョンの Chef に適用されます。Chef 12 以降で環境変数を取得するには、アプリケーションデータバッグを使用する必要があります。詳細については、「[AWS OpsWorks データバッグリファレンス](#)」および「[アプリケーションデータバッグ \(aws_opsworks_app\)](#)」を参照してください。

[アプリケーションの環境変数を指定すると、Stacks は変数定義をアプリケーションの `deploy` 属性に追加します。](#) AWS OpsWorks

カスタムレイヤーでは、レシピを使用して標準ノード構文によって変数の値を取得し、その値をレイヤーのアプリがアクセスできる形式で保存することができます。

インスタンスの `deploy` 属性から環境変数の値を取得するカスタムレシピを実装する必要があります。これにより、レシピではアプリケーションからアクセスできる形式のデータ (YAML ファイルなど) をインスタンスに保存できます。アプリケーションの環境変数の定義は、アプリケーションの `deploy` 内の `environment_variables` 属性に保存されます。以下の例は、属性構造を表す JSON を使用して、`simplephpapp` という名前のアプリケーションのそれらの属性の位置を示しています。

```
{
  ...
  "ssh_users": {
  },
  "deploy": {
    "simplephpapp": {
      "application": "simplephpapp",
      "application_type": "php",
      "environment_variables": {
        "USER_ID": "168424",
        "USER_KEY": "somepassword"
      },
    },
    ...
  }
}
```

```
}
```

レシピでは、標準ノード構文を使用して変数の値を取得できます。以下の例では、前述の JSON から USER_ID 値を取得してそれを Chef ログに記録する方法を示します。

```
Chef::Log.info("USER_ID: #{node[:deploy]['simplephpapp'][:environment_variables]
[:USER_ID]}")
```

スタック設定およびデプロイメント JSON から情報を取得し、それをインスタンスに保存する方法の詳細については、「[アプリケーションへのデータの引き渡し](#)」を参照してください。

アプリケーションへのデータの引き渡し

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

多くの場合、サーバーのアプリケーションにキーと値のペアなどのデータを渡すと便利です。そのためには、[カスタム JSON](#) を使用してデータをスタックに追加します。AWS OpsWorks スタックは、ライフサイクルイベントごとに各インスタンスのノードオブジェクトにデータを追加します。

ただし、Chef 属性を使用したノードオブジェクトからのカスタム JSON データの取得は、レシピからはできますが、アプリケーションからはできないことに注意してください。カスタム JSON データを 1 つ以上のアプリケーションに渡すためには、node オブジェクトのデータを抽出して、それをアプリケーションが読み取ることができるファイルに書き込むカスタムレシピを実装する方法があります。このトピックの例では YAML ファイルにデータを書き込む方法を示しますが、同じ基本手順を JSON や XML などの他の形式にも使用できます。

スタックのインスタンスにキーと値のデータを渡すには、スタックに次のようなカスタム JSON を追加します。カスタム JSON をスタックに追加する方法の詳細については、「[カスタムの使用 JSON](#)」を参照してください。

```
{
  "my_app_data": {
    "app1": {
      "key1": "value1",
      "key2": "value2",
      "key3": "value3"
    },
    "app2": {
      "key1": "value1",
      "key2": "value2",
      "key3": "value3"
    }
  }
}
```

この例では、app1 と app2 という短縮名の 2 つのアプリケーションそれぞれに 3 つのデータ値があることを前提としています。添付のレシピでは、関連付けられたデータを識別するためにアプリケーションの短縮名を使用しており、他の名前は任意であることを前提としています。アプリケーションの短縮名の詳細については、「[設定](#)」を参照してください。

次の例のレシピでは、deploy 属性から各アプリケーション用のデータを抽出し、.yaml ファイルに保存する方法を示しています。レシピは、カスタム JSON に各アプリケーション用のデータが含まれていることを前提としています。

```
node[:deploy].each do |app, deploy|
  file File.join(deploy[:deploy_to], 'shared', 'config', 'app_data.yaml') do
    content YAML.dump(node[:my_app_data][app].to_hash)
  end
end
```

deploy 属性には、各アプリケーションの属性が含まれています (アプリケーションの短縮前が付いています)。各アプリケーション属性には、アプリケーションに関するさまざまな情報を表す一連の属性が含まれます。この例では、[:deploy][:app_short_name][:deploy_to] 属性で表されるアプリケーションのデプロイメントディレクトリを使用しています。[:deploy] の詳細については、「[deploy 属性](#)」をご参照ください。

deploy の各アプリケーションに対して、レシピは以下のことを行います。

1. app_data.yaml という名前のファイルを、アプリケーションの [:deploy_to] ディレクトリの shared/config サブディレクトリに作成します。

AWS OpsWorks スタックによるアプリケーションのインストール方法の詳細については、「」を参照してください[Deploy レシピ](#)。

2. アプリケーションのカスタム JSON 値を YAML に変換し、YAML 形式のデータを `app_data.yml` に書き込みます。

アプリケーションにデータを渡すには

1. スタックにアプリケーションを追加し、その短縮名を書き留めます。詳細については、「[アプリケーションの追加](#)」を参照してください。
2. 前述のように、カスタム JSON とアプリケーションのデータを `deploy` 属性に追加します。スタックにカスタム JSON を追加する方法の詳細については、「[カスタムの使用 JSON](#)」を参照してください。
3. クックブックを作成し、レシピを追加します。レシピのコードは、前述の例をベースにして、必要に応じてカスタム JSON に使用した属性名に変更してください。クックブックとレシピの作成方法の詳細については、「[クックブックとレシピ](#)」を参照してください。このスタック用のカスタムクックブックがすでにある場合は、既存のクックブックにレシピを追加することも、既存の Deploy レシピにコードを追加することもできます。
4. スタックにクックブックをインストールします。詳細については、「[カスタムクックブックのインストール](#)」を参照してください。
5. アプリケーションサーバーレイヤーのデプロイライフサイクルイベントに `recipe` を割り当てます。AWS OpsWorks スタックは、起動後に新しいインスタンスごとに `recipe` を実行します。詳細については、「[レシピの実行](#)」を参照してください。
6. アプリケーションをデプロイすると、データが含まれたスタック設定とデプロイメント属性もインストールされます。

Note

アプリケーションのデプロイ前にデータファイルが用意できている場合は、レイヤーの Setup ライフサイクルイベントにレシピを割り当てることもできます。このイベントは、インスタンスの起動直後に 1 回のみ発生します。ただし、AWS OpsWorks スタックはデプロイディレクトリをまだ作成していないため、データファイルを作成する前にレシピで必要なディレクトリを明示的に作成する必要があります。次の例では、アプリケーションの `/shared/config` ディレクトリを明示的に作成し、次にそのディレクトリにデータファイルを作成しています。

```
node[:deploy].each do |app, deploy|

  directory "#{deploy[:deploy_to]}/shared/config" do
    owner "deploy"
    group "www-data"
    mode 0774
    recursive true
    action :create
  end

  file File.join(deploy[:deploy_to], 'shared', 'config', 'app_data.yml') do
    content YAML.dump(node[:my_app_data][app].to_hash)
  end
end
```

データをロードするには、次に示すような [Sinatra](#) コードを使用できます。

```
#!/usr/bin/env ruby
# encoding: UTF-8
require 'sinatra'
require 'yaml'

get '/' do
  YAML.load(File.read(File.join('..', '..', 'shared', 'config', 'app_data.yml')))
End
```

次のようにカスタム JSON を更新することで、いつでもアプリケーションのデータ値を更新することができます。

アプリケーションのデータを更新するには

1. データ値を更新するには、カスタム JSON を編集します。
2. アプリケーションを再度デプロイします。これにより、AWS OpsWorks スタックはスタックのインスタンスで Deploy レシピを実行するように指示されます。レシピでは更新されたスタック設定とデプロイメント属性の属性が使用されるため、カスタムレシピによってデータファイルが現在の値で更新されます。

Git リポジトリの SSH キーの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Git リポジトリの SSH キー (デプロイ SSH キーと呼ばれることもある) は、プライベート Git リポジトリへのアクセスを提供するパスワードのない SSH キーです。この SSH キーは、特定の開発者に固有のものではないことが理想的です。その目的は、AWS OpsWorks スタックが Git リポジトリからアプリケーションやクックブックを非同期的にデプロイできるようにすることです。ユーザーからの追加の入力は必要ありません。

次に、リポジトリ SSH キーを作成するための基本的な手順について説明します。詳細については、使用しているリポジトリのドキュメントを参照してください。例えば、[デプロイキーの管理](#)ではリポジトリのリポジトリ SSH キーを作成する方法について説明し GitHub、[Bitbucket のデプロイキー](#)では Bitbucket リポジトリのリポジトリ SSH キーを作成する方法について説明します。一部のドキュメントでは、サーバーにキーを作成する方法が記載されていることに注意してください。AWS OpsWorks スタックの場合は、手順の「サーバー」を「ワークステーション」に置き換えるだけです。

リポジトリ SSH キーを作成するには

1. ssh-keygen などのプログラムを使用して、ワークステーションに Git リポジトリ用のデプロイ SSH キーペアを作成します。

Important

AWS OpsWorks スタックは SSH キーパスフレーズをサポートしていません。

2. リポジトリにパブリックキーを割り当てて、プライベートキーをワークステーションに保存します。

3. アプリケーションを追加する場合や、クックブックリポジトリを指定する場合は、[Repository SSH Key] ボックスにプライベートキーを入力します。詳細については、「[アプリケーションの追加](#)」を参照してください。

AWS OpsWorks スタックはリポジトリ SSH キーを各インスタンスに渡し、組み込みレシピはそのキーを使用してリポジトリに接続し、コードをダウンロードします。キーは `deploy` 属性に `node[:deploy]['appshortname'][:scm][:ssh_key]` として保存され、ルートユーザーに対してのみアクセス可能になります。

カスタムドメインの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

サードパーティでドメイン名をホストしている場合、そのドメイン名をアプリケーションにマッピングできます。基本的な手順は次のとおりです。

1. サブドメインを DNS レジストラで作成し、ロードバランサーの Elastic IP アドレスまたはアプリケーションサーバーのパブリック IP アドレスにマッピングします。
2. アプリケーションの設定を更新して、サブドメインを指定し、アプリケーションを再デプロイします。

Note

非修飾ドメイン名 (myapp1.example.com など) が、修飾ドメイン名 (www.myapp1.example.com など) に転送されることを確認します。これにより、どちらもアプリケーションにマッピングされます。

アプリケーション用にドメインを設定すると、サーバー設定ファイルのサーバーエイリアスとして一覧表示されます。ロードバランサーを使用する場合、ロードバランサーはリクエストを受け取ったときに URL のドメイン名を確認し、そのドメインに基づいてトラフィックをリダイレクトします。

サブドメインを IP アドレスにマッピングするには

1. ロードバランサーを使用する場合、[Instances] (インスタンス) ページでロードバランサーインスタンスをクリックして詳細ページを開き、インスタンスの [Elastic IP] アドレスを確認します。それ以外の場合、アプリケーションサーバーインスタンスの詳細ページでパブリック IP アドレスを確認します。
2. DNS レジストラが指定している手順に従って、ステップ 1 から、サブドメインを作成して IP アドレスにマッピングします。

Note

ある時点でロードバランサーインスタンスが終了した場合、新しい Elastic IP アドレスが割り当てられています。新しい Elastic IP アドレスにマッピングするために、DNS レジストラ設定を更新する必要があります。

AWS OpsWorks スタックは、単にドメイン設定をアプリケーションの [deploy 属性](#) に追加します。カスタムレシピを実装して、ノードオブジェクトから情報を取得し、適切にサーバーを設定する必要があります。詳細については、「[クックブックとレシピ](#)」を参照してください。

同じアプリケーションサーバー上での複数のアプリケーションの実行

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

このトピックの情報は、Node.js アプリケーションには適用されません。

同じタイプの複数のアプリケーションがある場合、同じアプリケーションサーバーインスタンスで実行すると、コストパフォーマンスが向上する可能性があります。

複数のアプリケーションを同じサーバーで実行するには

1. 各アプリケーション用スタックにアプリケーションを追加します。
2. アプリケーションごとに別々のサブドメインを取得し、アプリケーションサーバーまたはロードバランサーの IP アドレスにマッピングします。
3. 各アプリケーションの設定を編集し、適切なサブドメインを指定します。

この作業を実行する方法の詳細については、「[カスタムドメインの使用](#)」を参照してください。

Note

アプリケーションサーバーで複数の HTTP アプリケーションを実行している場合、Elastic Load Balancing を使用して負荷を分散できます。複数の HTTPS アプリケーションがある場合、ロードバランサーで SSL 接続を終了するか、アプリケーションごとに別々のスタックを作成する必要があります。HTTPS リクエストが暗号化されていると、SSL 接続をサーバーで終了した場合、ロードバランサーはドメイン名を確認できず、リクエストを処理するアプリケーションを判断できません。

SSL の使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

アプリケーションで SSL を使用するには、最初に認証機関 (CA) からデジタルサーバー証明書を取得する必要があります。簡単になるように、このチュートリアルでは、証明書を作成し、その証明書を自己署名します。自己署名した証明書は学習やテストに便利ですが、本稼働用スタックには、CA によって署名された証明書を常に使用する必要があります。

このウォークスルーでは、次の作業を行います。

1. OpenSSL をインストールおよび設定します。
2. プライベートキーを作成します。
3. 証明書署名リクエストを作成します。
4. 自己署名証明書を生成します。
5. 証明書情報を使用してアプリケーションを編集します。

Important

アプリケーションで SSL を使用する場合、[CVE-2014-3566](#) で説明されている脆弱性に対応するため、可能であればアプリケーションサーバーレイヤーで SSLv3 を無効にすることをお勧めします。スタックに Ganglia レイヤーが含まれる場合は、そのレイヤーでも SSL v3 を無効にする必要があります。その詳細は特定のレイヤーによって異なります。詳細については、以下を参照してください。

- [Java アプリケーションサーバー AWS OpsWorks スタックレイヤー](#)
- [Node.js アプリケーションサーバー AWS OpsWorks スタックレイヤー](#)
- [PHP アプリケーションサーバー AWS OpsWorks スタックレイヤー](#)
- [Rails App Server AWS OpsWorks スタックレイヤー](#)
- [静的ウェブサーバー AWS OpsWorks スタックレイヤー](#)
- [Ganglia レイヤー](#)

トピック

- [ステップ 1: OpenSSL をインストールおよび設定する](#)
- [ステップ 2: プライベートキーを作成する](#)
- [ステップ 3: 証明書署名リクエストを作成する](#)
- [ステップ 4: CSR を認証機関に送信する](#)

- [ステップ 5: アプリケーションを編集する](#)

ステップ 1: OpenSSL をインストールおよび設定する

サーバー証明書を作成およびアップロードするには、SSL プロトコルと TLS プロトコルをサポートするツールが必要です。OpenSSL はオープンソースのツールで、RSA トークンの作成やそのトークンにプライベートキーを署名するための基本的な暗号関数を提供します。

次の手順の説明は、コンピュータにまだ OpenSSL がインストールされていないことを前提としています。

Linux や Unix で OpenSSL をインストールするには

1. [OpenSSL の \[Source\] の \[Tarballs\]](#) に移動します。
2. 最新のソースをダウンロードします。
3. パッケージをビルドします。

Windows へ OpenSSL をインストールするには

1. Microsoft Visual C++ 2008 再頒布可能パッケージがシステムにインストールされていない場合は、[こちらからダウンロード](#)してください。
2. インストーラを実行し、Microsoft Visual C++ 2008 Redistributable のセットアップウィザードの指示に従って、再頒布可能コードをインストールします。
3. [\[OpenSSL: Binary Distributions\]](#) に移動し、ご利用の環境に応じたバージョンの OpenSSL バイナリをクリックして、インストーラをローカルに保存します。
4. インストーラを実行し、OpenSSL セットアップウィザードの指示に従ってバイナリをインストールします。

ターミナルウィンドウまたはコマンドウィンドウを開き、次のコマンドラインを使用して、OpenSSL のインストールポイントを指す環境変数を作成します。

- Linux および UNIX の場合

```
export OpenSSL_HOME=path_to_your_OpenSSL_installation
```

- Windows の場合

```
set OpenSSL_HOME=path_to_your_OpenSSL_installation
```

ターミナルウィンドウまたはコマンドウィンドウを開き、次のコマンドラインを使用して、コンピュータのパス変数に OpenSSL バイナリのパスを追加します。

- Linux および UNIX の場合

```
export PATH=$PATH:$OpenSSL_HOME/bin
```

- Windows の場合

```
set Path=OpenSSL_HOME\bin;%Path%
```

Note

これらのコマンドラインを使用して環境変数に加えた変更は、現在のコマンドラインセッションでのみ有効です。

ステップ 2: プライベートキーを作成する

証明書署名リクエスト (CSR) を作成するには一意のプライベートキーが必要です。次のコマンドラインを使用してキーを作成します。

```
openssl genrsa 2048 > privatekey.pem
```

ステップ 3: 証明書署名リクエストを作成する

証明書署名リクエスト (CSR) は、デジタルサーバー証明書を申請するために認証機関 (CA) に送信されるファイルです。次のコマンドラインを使用して CSR を作成します。

```
openssl req -new -key privatekey.pem -out csr.pem
```

コマンドの出力は以下のようになります。

You are about to be asked to enter information that will be incorporated into your certificate request.
 What you are about to enter is what is called a Distinguished Name or a DN.
 There are quite a few fields but you can leave some blank
 For some fields there will be a default value,
 If you enter '.', the field will be left blank.

次の表は、証明書リクエストを作成する際に役立ちます。

証明書リクエストデータ

名前	説明	例
国名	2文字の ISO 略称 (国名コード)。	例 :US = アメリカ
州または県	あなたが所属する組織の所在地の州または県。省略不可です。	ワシントン
市区町村	あなたが所属する組織の所在地の市区町村。	Seattle
組織名	組織の正式名称。組織名は、省略不可です。	CorporationX
部門名	(オプション) 追加の組織情報。	マーケティング
共通名	CNAME に対する完全に適合するドメイン名。完全に一致しない場合は、証明書名チェックの警告が通知されません。	www.example.com
E メールアドレス	サーバー管理者の E メールアドレス	someone@example.com

Note

共通名フィールドは誤解されることが多く、間違って入力されることがよくあります。共有名とは通常の場合、ホスト名にドメイン名を付け加えたものです。"www.example.com" または "example.com" のようになります。正しい共有名を使用して CSR を作成しなければなりません。

ステップ 4: CSR を認証機関に送信する

本稼働環境で使用する場合は、認証機関 (CA) に CSR を送信してサーバー証明書を取得します。この場合、他の認証情報や識別の根拠が必要になることがあります。申請が正常に処理された場合、CA はデジタル署名されたアイデンティティ証明書と、場合によっては証明書チェーンファイルを返します。AWS が特定の CA を推奨することはありません。利用可能な CA の一覧 (一部のみ) については、Wikipedia の [Certificate Authority - Providers](#) を参照してください

また、テスト目的でのみ使用できる自己署名証明書を生成することもできます。この例では、次のコマンドラインを使用して自己署名証明書を生成します。

```
openssl x509 -req -days 365 -in csr.pem -signkey privatekey.pem -out server.crt
```

出力は以下のようになります。

```
Loading 'screen' into random state - done
Signature ok
subject=/C=us/ST=Washington/L=Seattle/O=CorporationX/OU=Marketing/CN=example.com/
emailAddress=someone@example.com
Getting Private key
```

ステップ 5: アプリケーションを編集する

証明書を生成して署名したら、SSL を有効化して証明書情報を指定するために、アプリケーションを更新します。[Apps (アプリ)] ページで、アプリを選択して詳細ページを開き、[Edit App (アプリの編集)] をクリックします。SSL サポートを有効化するには、[Enable SSL (SSL の有効化)] を [Yes (はい)] に設定します。次の設定オプションが表示されます。

SSL 証明書

ボックスにパブリックキー証明書 (.crt) ファイルの内容を貼り付けます。証明書の例を次に示します。

```
-----BEGIN CERTIFICATE-----  
MIICuTCCAiICCCQCtqFKItVQJpzANBgkqhkiG9w0BAQUFADCBoDELMakGA1UEBhMC  
dXMxEzARBgNVBAgMCndhc2hpbmd0b24xEDA0BgNVBACMB3N1YXR0bGUxDzANBgNV  
BAoMBmFtYXpvbjEWMBQGA1UECwwNRGV2IGFuZCBUb29sczEdMBsGA1UEAwwUc3Rl  
cGhhbm1lYXBpZXJjZS5jb20xIjAgBgkqhkiG9w0BCQEW3NhcG11cmNlQGftYXpv  
...  
-----END CERTIFICATE-----
```

Note

Nginx を使用していて、証明書チェーンファイルがある場合は、その内容をパブリックキー証明書ファイルに追加する必要があります。

既存の証明書を更新する場合は、次のようにします。

- 証明書を更新するには [Update SSL certificate (SSL 証明書の更新)] を選択します。
- 新しい証明書が既存のプライベートキーと一致しない場合は、[Update SSL certificate key (SSL 証明書キーの更新)] を選択します。
- 新しい証明書が既存の証明書チェーンと一致しない場合は、[Update SSL certificates (SSL 証明書の更新)] を選択します。

[SSL Certificate Key]

ボックスにプライベートキーファイル (.pem ファイル) の内容を貼り付けます。次のようになります。

```
-----BEGIN RSA PRIVATE KEY-----  
MIICXQIBAAKBgQC0CYk1JY5r4vV2NHQYEpwtsLuMMBhy1MrgBShKq+HHVLYQQCL6  
+wGIiRq5qXqZ1RXje3GM5Jvc6q0R71MfRI11FuzKyqDtneZaAIEYniZibHiUnm0  
/UNqPFdosw/6hY30Nk0fSB1U4ivD0Gjpf6J80jL3DJ4R23Ed0sdL4pRT3QIDAQAB  
AoGBAKmMfW1rNRqYVtGKgnWB6Tji9QrKQLMXjmHeGg95mppdJELiXHhpMvrHtpIyK  
...  
-----END RSA PRIVATE KEY-----
```

[SSL certificates of Certification Authorities]

証明書チェーンファイルがある場合は、ボックスにその内容を貼り付けます。

Note

Nginx を使用している場合は、このボックスを空にしておく必要があります。証明書チェーンファイルがある場合は、これを [SSL Certificate (SSL 証明書)] のパブリックキー証明書ファイル に追加します。

The screenshot shows the 'App railsapp' configuration page in the AWS OpsWorks console. Under the 'Deploy Settings' section, the 'SSL Settings' are visible. The 'SSL Support' toggle is turned 'On'. Below it are three text input fields: 'SSL Certificate', 'SSL Certificate Key', and 'SSL Certificates of Certification Authorities'. At the bottom right of the settings area, there are 'Cancel' and 'Save' buttons.

[Save] をクリックしたら、[アプリケーションを再デプロイ](#)して、オンラインインスタンスを更新します。

[組み込みアプリケーションサーバーレイヤー](#) の場合、AWS OpsWorks スタックはサーバー設定を自動的に更新します。デプロイが完了したら、以下のようにして OpenSSL のインストールが成功したことを確認できます。

OpenSSL のインストールを検証するには

1. [Instances] ページに移動します。
2. アプリケーションサーバーのインスタンスの IP アドレス (または、ロードバランサーを使用している場合はロードバランサーの IP アドレス) をクリックして、アプリケーションを実行します。
3. IP アドレスのプレフィックスを **http://** から **https://** に変更し、ブラウザを更新して、SSL でページが正しく読み込まれることを検証します。

アプリが Mozilla Firefox で作動するように設定されていると、「SEC_ERROR_UNKNOWN_ISSUER」という証明書エラーが表示されることがあります。このエラーは、お客様の組織のウイルス対策およびマルウェア対策プログラムに搭載されている証明書交換機能、一部の種類のネットワークトラフィックモニタリングおよびフィルタリングソフトウェア、またはマルウェアのいずれかが原因で発生することがあります。このエラーのトラブルシューティング方法の詳細については、Mozilla Firefox Support ウェブサイトに掲載されている、「[安全なウェブサイトでの「安全な接続ではありません」エラーコードをトラブルシューティングするには](#)」を参照してください。

カスタムレイヤーも含めてすべてのレイヤーに対して、AWS OpsWorks スタックによって SSL 設定がアプリケーションの [deploy 属性](#) に追加されるだけです。カスタムレシピを実装して、ノードオブジェクトから情報を取得し、適切にサーバーを設定する必要があります。

クックブックとレシピ

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは Chef クックブックを使用して、パッケージのインストールと設定、アプリケーションのデプロイなどのタスクを処理します。このセクションでは、AWS OpsWorks スタックでクックブックを使用する方法について説明します。詳細については、「[Chef](#)」を参照してください。

Note

AWS OpsWorks スタックは現在、Chef バージョン 12、11.10.4、11.4.4、および 0.9.15.5 をサポートしています。ただし、Chef 0.9.15.5 は廃止されており、新しいスタック用に使用することはお勧めしません。便宜上の理由で、これらは通常、メジャーバージョン番号とマイナーバージョン番号でのみ示されます。Chef 0.9 または 11.4 を実行中のスタックは [Chef Solo](#) を使用し、Chef 12 または 11.10 を実行中のスタックは [Chef Client](#) をローカルモードで使用します。Linux スタックの場合、使用する Chef のバージョンは、[スタックの作成時](#)に設定マネージャを使用して指定できます。Windows スタックは Chef 12.2 を使用する必要があります。より新しい Chef のバージョンにスタックを移行するためのガイドラインなど、詳細については、「[Chef のバージョン](#)」を参照してください。

トピック

- [クックブックリポジトリ](#)
- [Chef のバージョン](#)
- [Ruby のバージョン](#)
- [カスタムクックブックのインストール](#)
- [カスタムクックブックの更新](#)
- [レシピの実行](#)

クックブックリポジトリ

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

カスタムクックブックはオンラインリポジトリ (.zip ファイルのようなアーカイブまたは Git のようなソース管理マネージャー) に保存する必要があります。スタックに割り当てることができるカスタムクックブックは 1 つのみですが、リポジトリには任意の数のクックブックを保存できます。クックブックをインストールまたは更新すると、AWS OpsWorks スタックは各スタックのインスタンスのローカルキャッシュにリポジトリ全体をインストールします。たとえば、インスタンスで 1 つ以上のレシピを実行する必要がある場合、ローカルキャッシュのコードが使用されます。

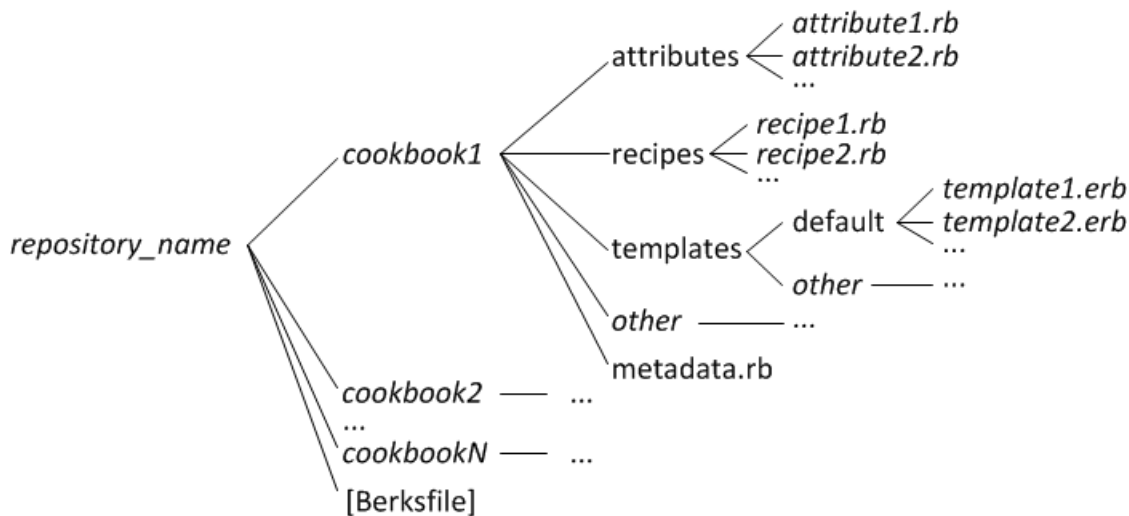
以下では、タイプ別にクックブックリポジトリを構築する方法について説明します。図で斜体で示されているテキストは、リポジトリ名またはアーカイブ名を含むユーザー定義ディレクトリとファイル名を表します。

ソース管理マネージャー

AWS OpsWorks スタックでは、次のソースコントロールマネージャーがサポートされています。

- Linux スタック – Git と Subversion
- Windows スタック – Git

以下に示しているのは、必要なディレクトリとファイル構造です。



- クックブックのディレクトリはすべてトップレベルにあることが必要です。

アーカイブ

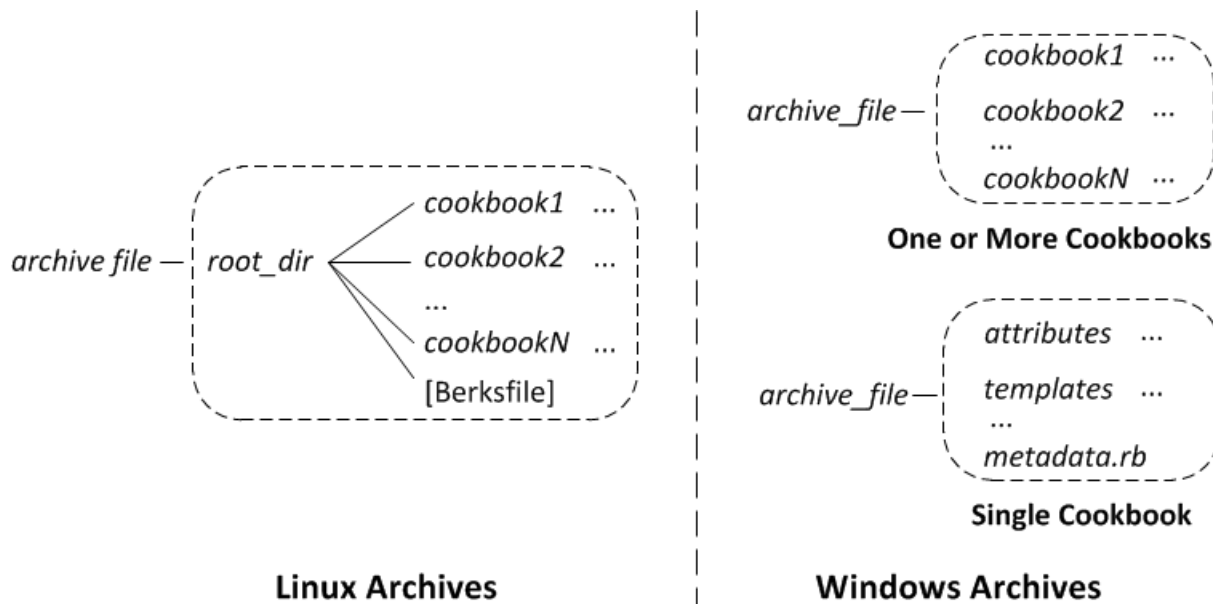
AWS OpsWorks スタックでは、次のアーカイブがサポートされています。

- Linux スタック – Amazon S3 またはウェブサイト (アーカイブ) に保存されている zip、gzip、bzip2、または tarball ファイル。HTTP

AWS OpsWorks スタックは非圧縮 tarball をサポートしていません。

- Windows スタック – Amazon S3 に保存されている zip と tgz (gzip で圧縮された tar) ファイル。

以下では、実行しているのが Linux スタックであるか Windows スタックであるかに応じて、必要なディレクトリとファイルの構造を示しています。クックブック構造はSCMリポジトリの場合と同じであるため、省略記号 (...) で表されます。



- Linux スタック – クックブックディレクトリはルートディレクトリにあることが必要です。
- Windows スタック – クックブックはアーカイブのトップレベルであることが必要です。

クックブックが1つしかない場合は、必要に応じてクックブックディレクトリを省略し、クックブックファイルをトップレベルに配置できます。その場合、AWS OpsWorks スタックによって metadata.rb からクックブック名が取得されます。

各クックブックディレクトリには、次に示す標準ディレクトリとファイルが少なくとも1つ (通常はすべて)、標準の名前で含まれています。

- `attributes` – クックブックの属性ファイルです。
- `recipes` – クックブックのレシピファイルです。
- `templates` – クックブックのテンプレートファイルです。
- `other` – 定義や仕様など、他のファイルタイプを含むオプションのユーザー定義ディレクトリ。
- `metadata.rb` – クックブックのメタデータです。

Chef 11.10 以降では、レシピが他のクックブックに依存している場合、クックブックの `depends` ファイルに、対応する `metadata.rb` ステートメントを含める必要があります。たとえば、クックブックに `include_recipe anothercookbook::somerecipe` のようなステートメントを持つレシピが含まれている場合は、クックブックの `metadata.rb` ファイルに `depends "anothercookbook"` という行が含まれている必要があります。詳細については、「[クックブックのメタデータについて](#)」を参照してください。

テンプレートは、`templates` ディレクトリのサブディレクトリに保存されている必要があります。`templates` ディレクトリには、少なくとも 1 つの、またはオプションで複数のサブディレクトリが含まれています。それらのサブディレクトリにも同様に、サブディレクトリが含まれている場合があります。

- テンプレートには、通常、`default` というサブディレクトリがあります。ここには、Chef がデフォルトで使用するテンプレートファイルが含まれています。
- `other` は、オペレーティングシステムに固有のテンプレート用に使用できる、オプションのサブディレクトリを表します。
- Chef は、「[File Specificity](#)」で説明する命名規則に従って、適切なサブディレクトリのテンプレートを自動的に使用します。例えば、Linux および オペレーティングシステムの場合、`amazon` `amazon` または `ubuntu` `ubuntu` という名前のサブディレクトリに、それぞれのオペレーティングシステム固有のテンプレートを保存できます。

カスタムクックブックの処理方法の詳細は、使用するリポジトリのタイプによって異なります。

アーカイブを使用するには

1. 前のセクションで示したフォルダ構造を使用して、クックブックを実装します。
2. 圧縮されたアーカイブを作成し、Amazon S3 バケットやウェブサイトにアップロードします。

クックブックを更新した場合、新しいアーカイブファイルを作成してアップロードする必要があります。Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、「[S3 バケットを空にする方法](#)」または「[S3 バケットを削除する方法](#)」を参照してください。

を使用するには SCM

1. 先ほど示した構造を使用して、Git または Subversion リポジトリを設定します。

- 必要に応じて、リポジトリのバージョン管理機能を使用して複数のブランチまたはバージョンを実装します。

クックブックを更新する場合は、新しいブランチで更新し、新しいバージョンを使用する OpsWorks ように に指示できます。特定のタグ付きバージョンを指定することもできます。詳細については、「[カスタムクックブックリポジトリの指定](#)」を参照してください。

[カスタムクックブックのインストール](#) では、AWS OpsWorks スタックにクックブックリポジトリをスタックのインスタンスにインストールさせる方法について説明します。

Important

リポジトリ内の既存のクックブックを更新したら、`update_cookbooks`スタックコマンドを実行して、各オンラインインスタンスのローカルキャッシュを更新するように AWS OpsWorks スタックに指示する必要があります。詳細については、「[スタックコマンドの実行](#)」を参照してください。

Chef のバージョン

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは複数のバージョンの Chef をサポートしています。[スタックの作成時](#)にバージョンを選択します。その後、AWS OpsWorks スタックは、そのバージョンと互換性のある組み込みレシピのセットとともに、スタックのすべてのインスタンスにそのバージョンの Chef をインストールします。カスタムレシピをインストールする場合、スタックの Chef バージョンと互換性がある必要があります。

AWS OpsWorks スタックは現在、Linux スタックでは Chef バージョン 12、11.10、11.4、および 0.9、Windows スタックでは Chef 12.2 (現在の Chef 12.22) をサポートしています。便宜上の理由

で、これらは通常、メジャーバージョン番号とマイナーバージョン番号でのみ示されます。Linux スタックの場合、使用する Chef のバージョンは、[スタックの作成](#)時に設定マネージャを使用して指定できます。Windows スタックは Chef 12.2 を使用する必要があります。より新しい Chef のバージョンにスタックを移行するためのガイドラインなど、詳細については、「[Chef のバージョン](#)」を参照してください。バージョンの詳細については、「[AWS OpsWorks スタックオペレーティングシステム](#)」を参照してください。

Chef 12.2

Chef 12.2 のサポートは、2015 年 5 月に導入されました。Windows スタックでのみ使用されます。Windows スタックの Chef の現行バージョンは Chef 12.22 です。Ruby 2.3.6 を使用して実行され、[chef-client をローカルモード](#)で使用します。これにより、[chef-zero](#) と呼ばれるローカルインメモリ Chef サーバーが起動されます。このサーバーが存在する場合、レシピで Chef 検索およびデータバッグを使用できます。このサポートにはいくつかの制限事項がありますが（「[レシピを実装する: Chef 12.2](#)」を参照）、多くのコミュニティクックブックを変更せずに実行できます。

Chef 12

Chef 12 のサポートは、2015 年 12 月に導入されました。Linux スタックでのみ使用されます。Ruby 2.1.6 または 2.2.3 を使用して実行され、[chef-client をローカルモード](#)で使用します。これにより、レシピが Chef 検索とデータバッグを使用できるようになります。詳細については、「[AWS OpsWorks スタックオペレーティングシステム](#)」を参照してください。

Chef 11.10

Chef 11.10 のサポートは、2014 年 3 月に導入されました。Linux スタックでのみ使用されます。Ruby 2.0.0 を使用して実行され、[chef-client をローカルモード](#)で使用します。これにより、レシピが Chef 検索とデータバッグを使用できるようになります。このサポートにはいくつかの制限事項がありますが（「[レシピの実装: Chef 11.10](#)」を参照）、多くのコミュニティクックブックを変更せずに実行できます。[Berkshelf](#) を使用してクックブックの依存関係を管理することもできます。サポートされている Berkshelf バージョンは、オペレーティングシステムによって異なります。詳細については、「[AWS OpsWorks スタックオペレーティングシステム](#)」を参照してください。Chef 11.10 を使用する CentOS スタックを作成することはできません。

Chef 11.4

Chef 11.4 のサポートは、2013 年 7 月に導入されました。Linux スタックでのみ使用されます。Ruby 1.8.7 を使用して実行され、[chef-solo](#) を使用します。Chef 検索やデータバッグはサポートされません。多くの場合、AWS OpsWorks スタックでこれらの機能に依存するコミュニティクックブックを使用できますが、「」の説明に従って変更する必要があります。[新しい Chef バージョンへの移行](#)。Chef 11.4 を使用する CentOS スタックを作成することはできません

ん。Chef 11.4 スタックは、米国東部 (バージニア北部) リージョン以外のリージョンエンドポイントでのサポートはされていません。

Chef 0.9

Chef 0.9 は Linux スタックでのみ使用されますが、サポートが中止されました。注意:

- コンソールを使用して新しい Chef 0.9 スタックを作成することはできません。

CLI または を使用するか API、別の Chef バージョンでスタックを作成し、スタック設定を編集する必要があります。

- 新しい AWS OpsWorks スタック機能は、Chef 0.9 スタックでは使用できません。
- オペレーティングシステムの新しいバージョンでは、Chef 0.9 スタックに対して一部のサポートのみが提供される予定です。

特に、Amazon Linux 2014.09 およびそれ以降のバージョンでは、Ruby 1.8.7 が必要な Rails アプリケーションサーバーレイヤーを使用する Chef 0.9 スタックはサポートしていません。

- 欧州 (フランクフルト) を含む新しい AWS リージョンでは、Chef 0.9 スタックはサポートされていません。

Note

新しいスタックへの Chef 0.9 の使用はお勧めしません。既存のスタックを新しいバージョンの Chef にできるだけ早く移行してください。

AWS OpsWorks スタックでコミュニティ cookbook を使用する場合は、新しい Linux スタックに [Chef 12 を指定](#)し、既存の Linux スタックを Chef 12 に移行することをお勧めします。AWS OpsWorks スタックコンソール、API、または を使用して、既存のスタックを新しい Chef バージョン CLI に移行できます。詳細については、「[新しい Chef バージョンへの移行](#)」を参照してください。

トピック

- [Chef 12.2 スタック用のレシピを実装する](#)
- [Chef 12 スタック用のレシピの実装](#)
- [Chef 11.10 スタック用のレシピの実装](#)
- [Chef 11.4 スタック用のレシピの実装](#)
- [新しいバージョンの Chef への既存の Linux スタックの移行](#)

Chef 12.2 スタック用のレシピを実装する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Chef 12.2 (現バージョン Chef 12.22) は、この Chef バージョンの実行が必須とされている Windows スタックでのみ使用可能となっています。

- 目的によっては、レシピは Windows 固有の属性とリソースを使用する必要があります。

詳細については、「[Microsoft Windows 用の Chef](#)」を参照してください。

- Chef の実行に Ruby 2.3.6 が使用されるため、レシピで新しい Ruby の構文を使用できます。
- レシピで Chef の検索およびデータバッグを使用できます。

Chef 12.2 スタックでは、多くのコミュニティクックブックを変更しないで使用できます。詳細については、[Chef の検索の使用](#) および [データバッグの使用](#) を参照してください。

- 「[AWS OpsWorks スタックデータバッグリファレンス](#)」および「[組み込みクックブックの属性](#)」で説明されているスタック設定およびデプロイ属性のほとんどは、Windows レシピに使用できます。

これらの属性値は、Chef 検索を使用して取得できます。例については、[Chef の検索での属性値の取得](#) を参照してください。属性のリストについては、「[AWS OpsWorks スタックデータバッグリファレンス](#)」を参照してください。

Chef 12 スタック用のレシピの実装

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Chef 12 スタックは、Chef 11.10 スタックと比べて、次のような利点があります。

- Chef の実行に Ruby 2.1.6 が使用されるため、レシピで新しい Ruby の構文を使用できます。
- Chef 12 スタックでは、さらに多くのコミュニティクックブックを変更せずに使用できます。組み込みのクックブックがなければ、組み込みのクックブックとカスタムクックブック間の名前の競合は発生しなくなります。
- スタックが構築済みのパッケージを提供している Berkshelf AWS OpsWorks バージョンに制限されなくなりました。Berkshelf は Chef 12 の AWS OpsWorks スタックインスタンスにインストールされなくなりました。代わりに、ローカルワークステーション上の任意の Berkshelf バージョンを使用できます。
- スタックが Chef 12 (Elastic Load Balancing、Amazon、Amazon ECS) AWS OpsWorks で提供する組み込みクックブック RDS とカスタムクックブックが明確に分離されるようになりました。これにより、エラーが発生した Chef のトラブルシューティングが容易になります。

Chef 11.10 スタック用のレシピの実装

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Chef 11.10 スタックは、Chef 11.4 スタックと比べて、次のような利点があります。

- Chef の実行に Ruby 2.0.0 が使用されるため、レシピで新しい Ruby の構文を使用できます。
- レシピで Chef の検索およびデータバッグを使用できます。

Chef 11.10 スタックでは、多くのコミュニティクックブックを変更せずに使用できます。

- クックブックの管理に Berkshelf を使用できます。

Berkshelf は、カスタムクックブックを管理し、スタックでコミュニティクックブックを使用するための非常に柔軟な方法を提供します。

- クックブックは、`metadata.rb` で依存関係を宣言する必要があります。

クックブックが別のクックブックに依存する場合、その依存関係をカスタムクックブックの `metadata.rb` ファイルに含める必要があります。たとえば、クックブックに `include_recipe anothercookbook::somerecipe` のようなステートメントを持つレシピが含まれている場合は、クックブックの `metadata.rb` ファイルに `depends "anothercookbook"` という行が含まれている必要があります。

- AWS OpsWorks スタックは、スタックに MySQL レイヤーが含まれている場合にのみ、スタックのインスタンスに My SQL クライアントをインストールします。
- AWS OpsWorks スタックは、スタックに Ganglia レイヤーが含まれている場合にのみ、スタックのインスタンスに Ganglia クライアントをインストールします。
- デプロイで `bundle install` を実行し、インストールが失敗した場合、デプロイも失敗します。

Important

カスタムクックブックまたはコミュニティクックブックに組み込みクックブックの名前を再使用しないでください。組込クックブックと同じ名前のカスタムクックブックは失敗する可能性があります。Chef 11.10、11.4、および 0.9 スタックで使用できる組み込みクックブックの完全なリストについては、の [opsworks-cookbooks リポジトリ GitHub](#) を参照してください。

Chef 0.9 および 11.4 スタックで正常に実行される ASCII 以外の文字を持つクックブックは、Chef 11.10 スタックで失敗することがあります。これは Chef 11.10 スタックが、Ruby 1.8.7 よりもエンコードが厳密である Ruby 2.0.0 を使用して Chef を実行するためです。このようなクックブックを Chef 11.10 スタックで正常に実行するには、ASCII 以外の文字を使用する各ファイルに、エンコードに関するヒントを提供するコメントが上部にある必要があります。たとえば、UTF-8 エンコーディングの場合、コメントは `# encoding: UTF-8` になります。Ruby 2.0.0 のエンコードの詳細については、「[エンコード](#)」を参照してください。

トピック

- [クックブックのインストールと優先順位](#)
- [Chef の検索の使用](#)

- [データバッグの使用](#)
- [Berkshelf の使用](#)

クックブックのインストールと優先順位

AWS OpsWorks スタッククックブックをインストールする手順は、Chef 11.10 スタックでは以前の Chef バージョンと若干異なります。Chef 11.10 スタックの場合、AWS OpsWorks スタックが組み込みクックブック、カスタムクックブック、および Berkshelf クックブックをインストールすると、次の順序で共通のディレクトリにマージされます。

1. 組み込みクックブック。
2. Berkshelf クックブック (ある場合)。
3. カスタムクックブック (ある場合)。

AWS OpsWorks スタックがこのマージを実行すると、レシピを含むディレクトリのコンテンツ全体がコピーされます。重複がある場合、次のルールが適用されます。

- Berkshelf クックブックの内容は組み込みクックブックより優先されます。
- カスタムクックブックの内容は Berkshelf クックブックより優先されます。

このプロセスがどのように機能するかを示すために、3つのすべてのクックブックのディレクトリに `mycookbook` という名前のクックブックが含まれている、次のようなシナリオを考えてみます。

- 組み込みクックブック — `mycookbook` は、`someattributes.rb` という名前の属性ファイル、`sometemplate.erb` という名前のテンプレートファイル、`somerecipe.rb` という名前のレシピを含みます。
- Berkshelf クックブック — `mycookbook` は、`sometemplate.erb` および `somerecipe.rb` を含みます。
- カスタムクックブック — `mycookbook` は、`somerecipe.rb` を含みます。

マージされたクックブックには次のものが含まれます。

- 組み込みクックブックの `someattributes.rb`。
- Berkshelf クックブックの `sometemplate.erb`。
- カスタムクックブックの `somerecipe.rb`。

⚠ Important

組み込みクックブック全体をリポジトリにコピーし、クックブックの一部を変更することによって、Chef 11.10 スタックをカスタマイズしないでください。そうすることで、レシピを含む組み込みクックブック全体が上書きされます。AWS OpsWorks スタックがそのクックブックを更新した場合、プライベートコピーを手動で更新しない限り、スタックはこれらの更新のメリットを享受できません。スタックをカスタマイズする方法の詳細については、「[AWS OpsWorks スタックのカスタマイズ](#)」を参照してください。

Chef の検索の使用

レシピで Chef の [search メソッド](#) を使用して、スタックデータのクエリを実行できます。Chef サーバーの場合と同じ構文を使用しますが、スタックは Chef AWS OpsWorks サーバーをクエリするのではなく、ローカルノードオブジェクトからデータを取得します。これには、以下のデータが含まれます。

- インスタンスの [スタック設定およびデプロイ属性](#)。
- インスタンスの組み込みクックブックとカスタムクックブックの属性ファイルにある属性。
- Ohai によって収集されるシステムデータ。

スタック設定属性とデプロイ属性には、スタック内の各オンラインインスタンスのホスト名や IP アドレスなどのデータなど、レシピが検索を通じて通常取得する情報のほとんどが含まれます。AWS OpsWorks スタックは、[ライフサイクルイベント](#)ごとにこれらの属性を更新し、現在のスタック状態を正確に反映します。つまり、スタック内で検索に依存するコミュニティレシピを、変更せずに、頻繁に使用できます。search メソッドでは適切なデータが返されますが、データはサーバーではなく、スタック設定およびデプロイ属性から取得されます。

AWS OpsWorks スタック検索の主な制限は、ローカルノードオブジェクト、特にスタック設定およびデプロイ属性のデータのみを処理することです。そのため、次のような種類のデータは検索で使用できない可能性があります。

- 他のインスタンスにあるローカルに定義された属性。

レシピで属性がローカルに定義されている場合、その情報は AWS OpsWorks スタックサービスに報告されないため、検索を使用して他のインスタンスからそのデータにアクセスすることはできません。

- カスタム deploy 属性。

[アプリケーションをデプロイ](#)し、対応する属性がそのデプロイのスタックのインスタンスにインストールされるJSONときに、`カスタム` を指定できます。ただし、選択したインスタンスにのみデプロイする場合、属性はそのインスタンスにのみインストールされます。これらのカスタムJSON属性のクエリは、他のすべてのインスタンスで失敗します。さらに、カスタム属性は、その特定のデプロイに対してJSONのみスタック設定とデプロイに含まれます。カスタム属性にアクセスできるのは、次のライフサイクルイベントで、新しい一連のスタック設定およびデプロイ属性がインストールされるまでの間だけです。[スタックJSONにカスタムを指定](#)した場合、属性はライフサイクルイベントごとにすべてのインスタンスにインストールされ、検索を通じて常にアクセス可能であることに注意してください。

- 他のインスタンスの Ohai データ。

Chef の [Ohai ツール](#) は、インスタンスでさまざまなシステムデータを取得し、ノードオブジェクトに追加します。このデータはローカルに保存され、AWS OpsWorks スタックサービスに報告されないため、検索で他のインスタンスから Ohai データにアクセスすることはできません。ただし、このデータの一部はスタック設定およびデプロイ属性に含まれる場合があります。

- オフラインインスタンス。

スタック設定およびデプロイ属性には、オンラインインスタンスのデータのみが含まれます。

次のレシピの抜粋は、検索を使用してPHPレイヤーのインスタンスのプライベート IP アドレスを取得する方法を示しています。

```
appserver = search(:node, "role:php-app").first
Chef::Log.info("The private IP is '#{appserver[:private_ip]}')
```

Note

AWS OpsWorks スタックがスタック設定属性とデプロイ属性をノードオブジェクトに追加すると、実際には 2 セットのレイヤー属性が作成されます。各属性は同じデータを持ちます。1 つのセットは `layers` 名前空間にあります。これは、AWS OpsWorks スタックがデータを保存する方法です。もう 1 つのセットは `role` 名前空間にあります。Chef サーバーでは同等のデータはこのように保存されます。`role` 名前空間の目的は、Chef サーバーが AWS OpsWorks スタックインスタンスで実行するために実装された検索コードを許可することです。スタック専用のコードを記述する場合は、前の例 `role:php-app` で `layers:php-app` または AWS OpsWorks のいずれかを使用すると、同じ結果が返され `search` ます。

データバッグの使用

レシピ内で Chef の [data_bag_item メソッド](#) を使用して、データバッグ内の情報に対するクエリを実行できます。Chef サーバーを対象とする場合と同じ構文を使用しますが、AWS OpsWorks スタックではインスタンスのスタック設定およびデプロイ属性からデータを取得します。ただし、AWS OpsWorks スタックは現在 Chef 環境をサポートしていないため、`node.chef_environment` は常に `_default` を返します。

`カスタム` を使用して `[::opsworks][::data_bags]` 属性JSONに 1 つ以上の属性を追加することで、データバッグを作成します。次の例は、`カスタム` でデータバッグを作成するための一般的な形式を示していますJSON。

Note

クックブックリポジトリに追加することで、データバッグを作成することはできません。`カスタム` を使用する必要がありますJSON。

```
{
  "opsworks": {
    "data_bags": {
      "bag_name1": {
        "item_name1": {
          "key1" : "value1",
          "key2" : "value2",
          ...
        }
      },
      "bag_name2": {
        "item_name1": {
          "key1" : "value1",
          "key2" : "value2",
          ...
        }
      },
      ...
    }
  }
}
```

通常、[スタックJSONにカスタムを指定します](#)。これにより、後続のライフサイクルイベントごとにすべてのインスタンスにカスタム属性がインストールされます。アプリケーションをデプロイJSONするときにカスタムを指定することもできますが、これらの属性はそのデプロイにのみインストールされ、選択したインスタンスのセットにのみインストールされる場合があります。詳細については、「[アプリケーションのデプロイ](#)」を参照してください。

次のカスタムJSON例では、 という名前のデータバッグを作成しますmyapp. mysql という項目が 1 個と、キーと値のペアが 2 個含まれます。

```
{ "opsworks": {
  "data_bags": {
    "myapp": {
      "mysql": {
        "username": "default-user",
        "password": "default-pass"
      }
    }
  }
}
```

レシピでこのデータを使用するために、次の例に示すように、data_bag_item を呼び出して、データバッグと値の名前を渡すことができます。

```
mything = data_bag_item("myapp", "mysql")
Chef::Log.info("The username is '#{mything['username']}' ")
```

データバッグ内のデータを変更するには、カスタムを変更するだけでJSON、次のライフサイクルイベントのためにスタックのインスタンスにインストールされます。

Berkshelf の使用

Chef 0.9 および 11.4 スタックでは、カスタムクックブックリポジトリを 1 つだけインストールできます。Chef 11.10 スタックでは、[Berkshelf](#) を使用してクックブックとその依存関係を管理でき、これにより複数のリポジトリからクックブックをインストールできます。(詳細については、[ローカルでのクックブックの依存関係のパッケージ化](#)を参照してください)。特に、Berkshelf を使用すると、カスタムクックブックリポジトリにコピーする代わりに、AWS OpsWorks スタック互換のコミュニティクックブックをリポジトリから直接インストールできます。サポートされている

Berkshelf バージョンは、オペレーティングシステムによって異なります。詳細については、「[AWS OpsWorks スタックオペレーティングシステム](#)」を参照してください。

Berkshelf を使用するには、「[カスタムクックブックのインストール](#)」に説明されているように、明示的に有効にする必要があります。次に、インストールするクックブックを指定する Berkshelf ファイルを、クックブックリポジトリのルートディレクトリに含めます。

Berkshelf で外部クックブックソースを指定するには、デフォルトのリポジトリを指定するソース属性をファイルの上部に含めますURL。リポジトリを明示的に指定URLsしない限り、Berkshelf はソース内のクックブックを検索します。次に、インストールする各クックブックに対応する行を次の形式で追加します。

```
cookbook 'cookbook_name', ['>= cookbook_version'], [cookbook_options]
```

cookbook に続くフィールドが特定のクックブックを指定します。

- ***cookbook_name*** – (必須) クックブックの名前を指定します。

他のフィールドを含めない場合、Berkshelf は指定されたソースからクックブックをインストールしますURLs。

- ***cookbook_version*** – (オプション) クックブックのバージョンを指定します。

= や >= などのプレフィックスを使用して、特定のバージョンまたは使用可能なバージョンの範囲を指定します。バージョンを指定しない場合、Berkshelf は最新のバージョンをインストールします。

- ***cookbook_options*** – (オプション) 最後のフィールドは、リポジトリの場所などのオプションを指定する 1 つ以上のキーと値のペアを含むハッシュです。

例えば、git キーを含めて特定の Git リポジトリを指定したり、tag キーを含めて特定のリポジトリブランチを指定したりすることができます。リポジトリブランチを指定することは、通常、目的のクックブックを確実にインストールするための最適な方法です。

Important

Berkshelf に metadata 行を追加し、metadata.rb でクックブックの依存関係を宣言することによって、クックブックを宣言することは避けてください。この処理が正常に実行されるためには、両方のファイルが同じディレクトリにある必要があります。AWS

OpsWorks スタックでは、Berksfile はリポジトリのルートディレクトリにある必要がありますが、`metadata.rb` ファイルはそれぞれのクックブックディレクトリにある必要があります。代わりに、Berksfile 内で外部クックブックを明示的に宣言してください。

次に、クックブックを指定するさまざまな方法を示す Berksfile の例を示します。Berksfile を作成する方法の詳細については、「[Berkshelf](#)」を参照してください。

```
source "https://supermarket.chef.io"

cookbook 'apt'
cookbook 'bluepill', '>= 2.3.1'
cookbook 'ark', git: 'git://github.com/opscode-cookbooks/ark.git'
cookbook 'build-essential', '>= 1.4.2', git: 'git://github.com/opscode-cookbooks/build-essential.git', tag: 'v1.4.2'
```

このファイルは、次のクックブックをインストールします。

- コミュニティクックブックリポジトリにある `apt` の最新バージョン。
- バージョン 2.3.1 以降である場合に限り、コミュニティクックブックにある `bluepill` の最新バージョン。
- 指定されたりポジトリにある `ark` の最新バージョン。

この例 URL のは、 のパブリックコミュニティクックブックリポジトリ用ですが GitHub、プライベートリポジトリを含む他のリポジトリからクックブックをインストールできます。詳細については「[Berkshelf](#)」を参照してください。

- 指定されたりポジトリの `v1.4.2` ブランチにある `build-essential` クックブック。

カスタムクックブックリポジトリには、Berksfile に加えて、カスタムクックブックを含めることができます。この場合、AWS OpsWorks スタックは両方のクックブックをインストールします。つまり、インスタンスには最大 3 つのクックブックリポジトリを含めることができます。

- 組み込みクックブックは、`/opt/aws/opsworks/current/cookbooks` にインストールされます。
- カスタムクックブックリポジトリにクックブックが含まれる場合、そのクックブックは `/opt/aws/opsworks/current/site-cookbooks` にインストールされます。

- Berkshelf を有効にし、カスタムクックブックリポジトリに Berkshelf が含まれる場合、指定されたクックブックは `/opt/aws/opsworks/current/berkshelf-cookbooks` にインストールされます。

組み込みクックブックとカスタムクックブックは、セットアップ時に各インスタンスにインストールされ、カスタムクックブックの更新スタックコマンドを手動で実行しない限り、その後更新されません。AWS OpsWorks スタックは Chef の実行 `berks install` ごとに実行されるため、Berkshelf クックブックはライフサイクルイベントごとに次のルールに従って更新されます。

- リポジトリに新しいバージョンのクックブックがある場合、このオペレーションは、リポジトリからのクックブックを更新します。
- それ以外の場合、このオペレーションは、ローカルキャッシュから Berkshelf クックブックを更新します。

Note

このオペレーションは Berkshelf クックブックを上書きするため、クックブックのローカルコピーを変更している場合、その変更内容が上書きされます。詳細については「[Berkshelf](#)」を参照してください。

また、Berkshelf クックブックとカスタムクックブックの両方を更新する `Update Custom Cookbooks` スタックコマンドを実行することによって、Berkshelf クックブックを更新することもできます。

Chef 11.4 スタック用のレシピの実装

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

⚠ Important

カスタムクックブックまたはコミュニティクックブックに組み込みクックブックの名前を再使用しないでください。組込クックブックと同じ名前のカスタムクックブックは失敗する可能性があります。Chef 11.10、11.4、および 0.9 スタックで使用できる組み込みクックブックの完全なリストについては、の [opsworks-cookbooks リポジトリ GitHub](#) を参照してください。

Chef 11.4 スタックの大きな制限は、レシピで Chef の検索やデータバッグを使用できないことです。ただし、AWS OpsWorks スタックは、検索で取得する次のような情報の多くを含む [スタック設定およびデプロイ属性](#) を各インスタンスにインストールします。

- ホストまたはアプリケーション名など、コンソールからのユーザー定義のデータ。
- スタックのレイヤー、アプリケーション、インスタンスなど、AWS OpsWorks スタックサービスによって生成されたスタック設定データ、および IP アドレスなどの各インスタンスに関する詳細。
- ユーザーによって提供され、データバッグとほぼ同じ目的を果たすことができるデータを含むカスタムJSON属性。

AWS OpsWorks スタックは、イベントの Chef 実行を開始する前に、各ライフサイクルイベントの各インスタンスにスタック設定およびデプロイ属性の最新バージョンをインストールします。データは、標準の `node[:attribute][:child_attribute][...]` 構文を通じてレシピに使用できます。たとえば、スタック設定およびデプロイ属性にはスタック名 `node[:opsworks][:stack][:name]` が含まれています。

組み込みのレシピの 1 つに含まれる次のコードは、スタック名を取得し、その名前を使用して設定ファイルを作成します。

```
template '/etc/ganglia/gmetad.conf' do
  source 'gmetad.conf.erb'
  mode '0644'
  variables :stack_name => node[:opsworks][:stack][:name]
  notifies :restart, "service[gmetad]"
end
```

スタック設定およびデプロイ属性の値の多くには、複数の属性が含まれています。必要な情報を取得するために、これらの属性を反復処理する必要があります。以下の例は、便宜上、JSON オブジェクトとして表されるスタック設定およびデプロイ属性からの抜粋を示しています。これには、最上位属性 `deploy` が含まれています。この最上位属性には、スタックの各アプリケーションの属性が含まれており、アプリケーションの短縮名が付いています。

```
{
  ...
  "deploy": {
    "app1_shortcode": {
      "document_root": "app1_root",
      "deploy_to": "deploy_directory",
      "application_type": "php",
      ...
    },
    "app2_shortcode": {
      "document_root": "app2_root",
      ...
    }
  },
  ...
}
```

各アプリケーション属性には、アプリケーションの特性を示す一連の属性が含まれています。例えば、`deploy_to` 属性は、アプリケーションのデプロイディレクトリを表します。次のコードは、アプリケーションの各デプロイディレクトリのユーザー、グループ、およびパスを設定します。

```
node[:deploy].each do |application, deploy|
  opsworks_deploy_dir do
    user deploy[:user]
    group deploy[:group]
    path deploy[:deploy_to]
  end
  ...
end
```

スタック設定およびデプロイ属性の詳細については、「[AWS OpsWorks スタックのカスタマイズ](#)」を参照してください。デプロイディレクトリの詳細については、「[Deploy レシピ](#)」を参照してください。

Chef 11.4 スタックはデータバッグをサポートしていませんが、[カスタム JSON](#)を指定することで、スタック設定およびデプロイ属性に任意のデータを追加できます。レシピは、標準的な Chef のノード構文を使用して、データにアクセスできます。詳細については、「[カスタムの使用 JSON](#)」を参照してください。

暗号化されたデータバッグの機能が必要な場合、選択肢の 1 つとして、プライベート Amazon S3 バケットなどの安全な場所に機密性の高い属性を保存する方法があります。レシピは、すべてのスタックインスタンスにインストールされている [AWS Ruby SDK](#) AWS OpsWorks を使用して、バケットからデータをダウンロードすることができます。

Note

各 AWS OpsWorks スタックインスタンスにはインスタンスプロファイルがあります。関連付けられた [IAM ロール](#) は、インスタンスで実行されているアプリケーションでアクセスできる AWS リソースを指定します。レシピで Amazon S3 バケットにアクセスするためには、ロールのポリシーに次のようなステートメントを含めて、指定されたバケットからファイルを取得するアクセス許可を付与する必要があります。

```
"Action": ["s3:GetObject"],
"Effect": "Allow",
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
```

インスタンスプロファイルの詳細については、「[EC2 インスタンスで実行されるアプリケーションのアクセス許可の指定](#)」を参照してください。

新しいバージョンの Chef への既存の Linux スタックの移行

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

スタックコンソール、API、または `awscli` を使用して、Linux AWS OpsWorks スタックを新しい Chef バージョン CLI に移行できます。ただし、新しいバージョンに対応するために、レシピに変更を加えることが必要になる場合があります。スタックを移行するための準備をするときに、次の点を考慮します。

- スタックを編集またはクローンすることで AWS OpsWorks、スタックのスタックバージョンを Chef 11 から Chef 12 に変更することはできません。Chef のメジャーバージョンアップグレードは、このセクションの手順を使用して行うことはできません。Chef 11.10 から Chef 12 への移行の詳細については、「[レシピの実装: Chef 12](#)」を参照してください。
- Chef のあるバージョンから別のバージョンへの移行には多くの変更が伴い、その一部は互換性を破る変更です。

Chef 0.9 から Chef 11.4 への移行の詳細については、「[新しい Chef バージョンへの移行](#)」を参照してください。Chef 11.4 から Chef 11.10 への移行の詳細については、「[レシピの実装: Chef 11.10](#)」を参照してください。Chef 11.10 から Chef 12 への移行の詳細については、「[レシピの実装: Chef 12](#)」を参照してください。

- Chef の実行で使用される Ruby のバージョンは、Chef 0.9 および Chef 11.4 のスタックの場合 (Ruby 1.8.7)、Chef 11.10 のスタックの場合 (Ruby 2.0.0)、および Chef 12 のスタックの場合 (Ruby 2.1.6) とで異なります。

詳細については、「[Ruby のバージョン](#)」を参照してください。

- Chef 11.10 のスタックでは、クックブックのインストールの処理が、Chef 0.9 または Chef 11.4 のスタックの場合とは異なります。

この違いによって、カスタムクックブックを使用するスタックを Chef 11.10 に移行するときに問題が発生する可能性があります。詳細については、「[クックブックのインストールと優先順位](#)」を参照してください。

Chef のスタックを新しいバージョンの Chef に移行する際の推奨ガイドラインを以下に示します。

スタックを新しいバージョンの Chef に移行するには

1. [本稼働用スタックをクローン化します](#)。[Clone Stack] (クローンスタック) ページで、[Advanced>>] (アドバンスト>>) をクリックして [Configuration Management] (構成管理) セクションを表示し、[Chef version] (Chef のバージョン) を上位のバージョンに変更します。

Note

Chef 0.9 のスタックで作業を始める場合、直接 Chef 11.10 にアップグレードすることはできません。最初に Chef 11.4 にアップグレードする必要があります。レシピをテストする前に、Chef 11.10 にスタックを移行する場合は、更新が実行されるまで 20 分間待機してから、スタックを 11.4 から 11.10 にアップグレードします。

2. インスタンスをレイヤーに追加し、テストまたはステージング用システムにあるクローン化したスタックのアプリケーションとクックブックをテストします。詳細については、「[All about Chef ...](#)」を参照してください。
3. テスト結果が適切である場合、次のいずれかを実行します。
 - これが必要な Chef バージョンである場合、クローン化したスタックを本稼働用スタックとして使用するか、本稼働用スタックの Chef バージョンをリセットできます。
 - Chef 0.9 スタックから Chef 11.10 に 2 段階で移行している場合は、Chef 11.4 から Chef 11.10 にスタックを移行する処理を繰り返します。

Note

レシピをテストするときは、[SSHを使用してインスタンスに接続し、インスタンスエージェントの CLI `run_command` コマンド](#)を使用して、さまざまなライフサイクルイベントに関連付けられたレシピを実行できます。エージェント CLI は Setup レシピのテストに特に役立ちます。これは、Setup が失敗し、インスタンスがオンライン状態にならない場合でも使用できるためです。[Setup スタックコマンド](#)を使用して Setup レシピを再実行することもできますが、このコマンドは Setup が成功し、インスタンスがオンラインになっている場合にのみ利用できます。

実行中のスタックを新しいバージョンの Chef に更新することができます。

実行中のスタックを新しいバージョンの Chef に更新するには

1. [スタックを編集](#)し、スタックの設定の [Chef version] を変更します。
2. 新しい設定を保存し、AWS OpsWorks スタックがインスタンスを更新するのを待ちます。通常、更新には 15~20 分かかります。

⚠ Important

AWS OpsWorks スタックは Chef バージョンの更新をライフサイクルイベントと同期しません。本稼働用スタックで Chef のバージョンを更新する場合は、次の[ライフサイクルイベント](#)が発生する前に、更新が完了するように注意する必要があります。イベントが発生した場合 (主に Deploy または Configure イベント)、インスタンスエージェントは、バージョンの更新が完了しているかどうかにかかわらず、カスタムクックブックを更新し、イベントに割り当てられたレシピを実行します。バージョンの更新が完了したことを直接確認する方法はありませんが、デプロイのログに Chef のバージョンが記録されています。

Ruby のバージョン

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Linux スタック内のすべてのインスタンスに Ruby がインストールされています。AWS OpsWorks スタックは各インスタンスに Ruby パッケージをインストールします。このパッケージは Chef レシピとインスタンスエージェントの実行に使用されます。AWS OpsWorks スタックは、スタックが実行されている Chef バージョンに基づいて Ruby バージョンを決定します。このバージョンを変更しないでください。このバージョンを変更すると、インスタンスエージェントが無効になります。

AWS OpsWorks スタックは、Windows スタックにアプリケーション Ruby 実行可能ファイルをインストールしません。Chef 12.2 クライアントには Ruby 2.0.0 p451 が付属していますが、Ruby 実行可能ファイルはインスタンスの PATH 環境変数に追加されません。この実行可能ファイルを使用して Ruby コードを実行する場合、実行可能ファイルは Windows ドライブの `\opscod\chef\embedded\bin\ruby.exe` にあります。

次の表は AWS OpsWorks、Stacks Ruby のバージョンをまとめたものです。また、使用可能なアプリケーションの Ruby バージョンは、インスタンスのオペレーティングシステムによって異なります。詳細と使用可能なパッチバージョンの詳細については、「[AWS OpsWorks スタックオペレーティングシステム](#)」を参照してください。

Chef のバージョン	Chef の Ruby のバージョン	使用できるアプリケーションの Ruby のバージョン
0.9 (c)	1.8.7	1.8.7(a)、1.9.3(e)、2.0.0
11.4 (c)	1.8.7	1.8.7(a)、1.9.3(e)、2.0.0、2.1、2.2.0、2.3
11.10	2.0.0-p481	1.9.3(c、 e)、2.0.0、2.1、2.2.0、2.3、2.6.1
12 (b)	2.1.6、2.2.3	なし
12.22 (d)	2.3.6	[なし]

(a) Amazon Linux 2014.09 以降、Red Hat Enterprise Linux (RHEL)、または Ubuntu 14.04 では使用できませんLTS。

(b) Linux スタックにのみ使用できます。

(c) では使用できませんRHEL。

(d) Windows スタックにのみ使用できます。メジャーバージョンは 12.2 です。現在のマイナーバージョンは 12.22 です。

(e) 廃止が完了しました。サポートは終了しています。

インストール先の場所は、Chef のバージョンによって決まります。

- アプリケーションは、Chef のすべてのバージョンについて、`/usr/local/bin/ruby` の実行可能ファイルを使用します。
- Chef 0.9 と 11.4 の場合、インスタンスエージェントと Chef のレシピは `/usr/bin/ruby` の実行可能ファイルを使用します。
- Chef 11.10 の場合、インスタンスエージェントと Chef のレシピは `/opt/aws/opsworks/local/bin/ruby` の実行可能ファイルを使用します。

カスタムクックブックのインストール

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

スタックがカスタムクックブックをインストールおよび使用するようになるには、すでに設定されていない場合はカスタムクックブックが有効になるようにスタックを設定する必要があります。次に、リポジトリURLとパスワードなどの関連情報を指定する必要があります。

Important

カスタムクックブックをサポートするようにスタックを設定すると、AWS OpsWorks スタックは起動時にすべての新しいインスタンスにクックブックを自動的にインストールします。ただし、カスタムクックブックの更新 AWS OpsWorks スタックコマンドを実行して、既存のインスタンスに新規または更新されたクックブックをインストールするようにスタックに明示的に指示する必要があります。詳細については、「[カスタムクックブックの更新](#)」を参照してください。スタックの [Use custom Chef cookbooks (カスタム Chef クックブックを使用する)] を有効化する前に、実行するカスタムクックブックとコミュニティクックブックが、スタックで使用する Chef のバージョンをサポートしていることを確認してください。

カスタムクックブックのスタックを設定するには

1. スタックのページで、[Stack Settings] をクリックして [Settings] ページを表示し、[Edit] をクリックして設定を編集します。
2. [Use custom Chef cookbooks] を [Yes] に切り替えます。

Use custom Chef cookbooks Yes

Repository type

Repository URL

Repository SSH key

Branch/Revision

Stack color

3. カスタムクックブックを設定します。

設定が完了したら、[Save] をクリックして更新されたスタックを保存します。

カスタムクックブックリポジトリの指定

Linux スタックは、次のリポジトリタイプからカスタムクックブックをインストールできます。

- HTTP または Amazon S3 アーカイブ。

これらはパブリックでもプライベートでもかまいませんが、通常プライベートアーカイブに推奨されるオプションは Amazon S3 です。

- Git リポジトリと Subversion リポジトリは、ソース管理と複数のバージョンを設定する機能を提供します。

Windows スタックは、Amazon S3 アーカイブと Git リポジトリからカスタムクックブックをインストールできます。

すべてのリポジトリタイプに、以下の必須フィールドがあります。

- [Repository type] (リポジトリタイプ) – リポジトリのタイプ
- リポジトリ URL – リポジトリ URL

AWS OpsWorks スタックは、[GitHub](#)や [Bitbucket](#) などのパブリックにホストされた Git リポジトリサイトと、プライベートにホストされた Git サーバーをサポートします。Git リポジトリの場合、リ

ポジトリがパブリックかプライベートかに応じて、次のいずれかのURL形式を使用する必要があります。Git サブモジュールについても同じURLガイドラインに従ってください。

パブリック Git リポジトリの場合は、HTTPSまたは Git 読み取り専用プロトコルを使用します。

- Git 読み取り専用 – `git://github.com/amazonwebservices/opsworks-example-cookbooks.git`。
- HTTPS – `https://github.com/amazonwebservices/opsworks-example-cookbooks.git`。

プライベート Git リポジトリの場合は、次の例に示すように、SSH読み取り/書き込み形式を使用する必要があります。

- Github リポジトリ – `git@github.com:project/repository`。
- Git サーバー上のリポジトリ – `user@server:project/repository`

残りの設定はリポジトリのタイプによって異なり、これらについて以下のセクションで説明します。

HTTP アーカイブ

[Repository type] (リポジトリタイプ) で [Http Archive] (Http アーカイブ) を選択すると、2つの追加設定が表示されますが、アーカイブがパスワードで保護されている場合には設定を完了する必要があります。

- User name - ユーザーネーム。
- [Password] — パスワード

Amazon S3 のアーカイブ

リポジトリタイプに S3 アーカイブを選択すると、次の追加オプション設定が表示されます。AWS OpsWorks スタックは、スタックAPIとコンソールのどちらを使用するかにかかわらず、Amazon EC2ロール (ホストオペレーティングシステムマネージャー認証) AWS OpsWorks を使用してリポジトリにアクセスできます。

- アクセスキー ID – などのAWSアクセスキー ID AKIAIOSFODNN7EXAMPLE.
- シークレットアクセスキー – などの対応するAWSシークレットアクセスキー wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY.

Git リポジトリ

[Source Control] (ソースコントロール) で [Git] を選択すると、以下の追加オプション設定が表示されます。

リポジトリSSHキー

プライベート Git リポジトリにアクセスするには、デプロイSSHキーを指定する必要があります。Git サブモジュールの場合、指定するキーには、それらのサブモジュールへのアクセス権が必要です。詳細については、「[Git リポジトリの SSH キーの使用](#)」を参照してください。

Important

デプロイSSHキーはパスワードを要求できません。AWS OpsWorks スタックにはパスワードを渡す方法がありません。

[Branch/Revision]

リポジトリに複数のブランチがある場合、AWS OpsWorks スタックはデフォルトでマスターブランチをダウンロードします。特定のブランチを指定するには、ブランチ名、SHA1ハッシュ名、またはタグ名を入力します。特定のコミットを指定するには、40桁の16進数コミットIDを入力します。

Subversion リポジトリ

[Source Control] (ソースコントロール) で [Subversion] を選択すると、以下の追加設定が表示されます。

- [User name] (ユーザー名) - プライベートリポジトリのユーザー名。
- [Password] (パスワード) - プライベートリポジトリのパスワード。
- [Revision] (リビジョン) - (オプション) 複数のリビジョンがある場合のリビジョン名。

ブランチまたはタグを指定するには、リポジトリを変更する必要があります。URL例: **http://repository_domain/repos/myapp/branches/my-apps-branch**または **http://repository_domain_name/repos/calc/myapp/my-apps-tag**。

カスタムクックブックの更新

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックにカスタムクックブックを提供すると、組み込みの Setup レシピは新しく起動された各インスタンスにローカルキャッシュを作成し、クックブックをキャッシュにダウンロードします。AWS OpsWorks スタックは、リポジトリではなくキャッシュからレシピを実行します。リポジトリ内のカスタムクックブックを変更する場合は、更新されたクックブックがインスタンスのローカルキャッシュにインストールされていることを確認する必要があります。AWS OpsWorks スタックは、起動時に最新のクックブックを新しいインスタンスに自動的にデプロイします。ただし、既存のインスタンスは事情が異なります。

- オンラインのインスタンスには、更新したカスタムクックブックを手動でデプロイする必要があります。
- 負荷ベースのインスタンスと時間ベースのインスタンスを含め、オフラインの Instance Store-Backed インスタンスには、更新したカスタムクックブックをデプロイする必要はありません。

AWS OpsWorks スタックは、インスタンスが再起動されると、現在のクックブックを自動的にデプロイします。

- 負荷ベースまたは時間ベースではないオフラインの EBS-backed 24/7 インスタンスを起動する必要があります。
- オフラインの EBS-backed 負荷ベースおよび時間ベースのインスタンスは開始できないため、最も簡単な方法は、オフラインインスタンスを削除し、新しいインスタンスを追加して置き換えることです。

これらは新しいインスタンスになったため、AWS OpsWorks スタックはインスタンスの起動時に現在のカスタムクックブックを自動的にデプロイします。

カスタムクックブックを手動で更新するには

1. 変更されたクックブックでリポジトリを更新します。AWS OpsWorks スタックは、クックブックを最初にインストールしたときにURL指定したキャッシュを使用するため、クックブックのルートファイル名、リポジトリの場所、およびアクセス権は変更しないでください。
 - Amazon S3 または HTTP リポジトリの場合は、元の .zip ファイルを同じ名前の新しい .zip ファイルに置き換えます。
 - Git または Subversion のリポジトリの場合、[スタックの設定を編集](#)して、[Branch/Revision] フィールドを新しいバージョンに変更します。
2. スタックのページで、[Run Command] をクリックし、[Update Custom Cookbooks] コマンドを選択します。

Run Command

Settings

Command

Update Custom Cookbooks ▾

Deploys an updated set of custom Chef cookbooks from the repository to each instance's cookbooks cache.

Comment

Optional

Advanced »

Instances ⓘ

OpsWorks will run this command on **1 of 2** instances. The assigned recipes are run on all selected instances.

Select all

Rails App Server

Click to select instances in this layer

rails-app1 ●

MySQL

Click to select instances in this layer

db-master1 ●

Cancel

Update Custom Cookbooks

3. 必要に応じてコメントを追加します。
4. 必要に応じて、コマンドのカスタムJSONオブジェクトを指定して、スタックがインスタンスにAWS OpsWorks インストールするスタック設定およびデプロイ属性にカスタム属性を追加します。詳細については、[カスタムの使用 JSON](#)および[属性の上書き](#)を参照してください。

5. デフォルトでは、AWS OpsWorks スタックはすべてのインスタンスのクックブックを更新します。更新するインスタンスを指定するには、ページの最後にある一覧から適切なインスタンスを選択します。レイヤー内のすべてのインスタンスを選択するには、左側の列で目的のレイヤーのチェックボックスをオンにします。
6. カスタムクックブックの更新をクリックして、更新されたクックブックをインストールします。AWS OpsWorks スタックは、指定されたインスタンスでキャッシュされたカスタムクックブックを削除し、リポジトリから新しいクックブックをインストールします。

Note

この手順は、既存のインスタンスで、古いバージョンのクックブックがキャッシュに存在する場合のみ必要となります。その後、レイヤーにインスタンスを追加すると、AWS OpsWorks スタックは現在リポジトリにあるクックブックをデプロイし、最新バージョンを自動的に取得します。

レシピの実行

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

レシピは、次の 2 とおりの方法で実行できます。

- 適切なレイヤーのライフサイクルイベントにレシピを割り当てることによって自動的に実行します。
- 手動で、[Execute Recipes スタックコマンド](#) を実行するか、エージェント を使用します CLI。

トピック

- [AWS OpsWorks スタックライフサイクルイベント](#)
- [レシピを自動的に実行する](#)

- [レシピを手動で実行する](#)

AWS OpsWorks スタックライフサイクルイベント

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

各レイヤーには、5 つの一連のライフサイクルイベントがあり、それぞれのイベントに、そのレイヤーに固有の一連のレシピが関連付けられます。レイヤーのインスタンスでイベントが発生すると、一連の適切なレシピが AWS OpsWorks スタックによって自動的に実行されます。これらのイベントにカスタムレスポンスを提供するには、カスタムレシピを実装し、各レイヤーの [適切なイベントに割り当てます](#)。AWS OpsWorks スタックは、イベントの組み込みレシピの後にそれらのレシピを実行します。

Setup

このイベントは、開始されたインスタンスの起動が終了した後で発生します。また、Setup [Setup スタックコマンド](#) を使用して イベントを発生させます。AWS OpsWorks スタックは、レイヤーに従ってインスタンスを設定するレシピを実行します。例えば、インスタンスが Rails App Server レイヤーのメンバーである場合、Setup レシピによって、Apache、Ruby Enterprise Edition、Passenger、Ruby on Rails がインストールされます。

Note

Setup イベントは、インスタンスを停止中の状態にします。Setup ライフサイクルイベントの実行時はインスタンスが **オンライン** 状態でないため、Setup イベントを実行するインスタンスはロードバランサーから削除されます。

Configure

このイベントは、以下のいずれかが発生したときに、スタックのすべてのインスタンスで発生します。

- インスタンスがオンライン状態に移行したか、オンライン状態から移行した。
- [Elastic IP アドレスをインスタンスに関連付けたか](#)、[インスタンスから Elastic IP アドレスの関連付けを解除した](#)。
- [Elastic Load Balancing ロードバランサー](#)をレイヤーにアタッチしたり、レイヤーからデタッチしたりします。

たとえば、スタックにインスタンス A、B、C があるとき、新しいインスタンス D を開始するとします。D のセットアップレシピの実行が完了した後、AWS OpsWorks スタックによって Configure イベントが A、B、C、D でトリガーされます。さらに、その後 A を停止した場合、AWS OpsWorks スタックによって Configure B、C、および D。AWS OpsWorks Stacks のイベントが に応答します。Configure イベントへの応答として、各レイヤーの Configure recipes。現在のオンラインインスタンスのセットを反映するようにインスタンスの設定を更新します。- Configure したがって、 イベントは設定ファイルを再生成するのに適しています。たとえば、HAProxyConfigure レシピは、ロードバランサーを再構成して、一連のオンラインアプリケーションサーバーインスタンスにおける変更をすべて反映します。

[Configure スタックコマンド](#)を使用して、Configure イベントを手動でトリガーすることもできます。

Deploy

このイベントは、Deploy コマンドを実行したときに発生します。通常、一連のアプリケーションサーバーインスタンスにアプリケーションをデプロイする目的でこのコマンドが実行されます。アプリケーションとその関連ファイルをリポジトリからレイヤーのインスタンスにデプロイするレシピがインスタンスによって実行されます。たとえば、Rails Application Server インスタンスの場合、Deploy レシピは、指定された Ruby アプリケーションをチェックアウトし、[Phusion Passenger](#) に再ロードするように指示します。他のインスタンスで Deploy を実行することもできます。たとえば、新しくデプロイされたアプリケーションに応じてそのインスタンスの設定を更新できます。

Note

Setup には Deploy が含まれます。セットアップの完了後に Deploy レシピが実行されます。

Undeploy

このイベントは、アプリケーションを削除したとき、つまり、Undeploy 一連のアプリケーションサーバーインスタンスからアプリケーションを削除する コマンド。指定したインスタンスによってレシピが実行され、すべてのアプリケーションバージョンが削除されて、必要なクリーンアップ処理が実行されます。

Shutdown

このイベントは、インスタンスをシャットダウンするように AWS OpsWorks スタックに指示した後、関連付けられた Amazon EC2 インスタンスが実際に終了する前に発生します。AWS OpsWorks スタックによってサービスのシャットダウンなど、各種クリーンアップタスクを行うレシピが実行されます。

Elastic Load Balancing ロードバランサーをレイヤーにアタッチし、[Connection Draining のサポートを有効](#)している場合、AWS OpsWorks スタックは Connection Draining が完了するまで待ってから、Shutdown event。

をトリガーした後 Shutdown event、AWS OpsWorks Stacks は以下を許可します。Shutdown は、タスクを実行するために指定された時間をレシピし、Amazon EC2 インスタンスを停止または終了します。デフォルトの Shutdown タイムアウト値は 120 秒です。の場合 Shutdown レシピにはさらに時間がかかる場合があります。[レイヤー設定を編集](#)してタイムアウト値を変更できます。インスタンスの Shutdown 「[インスタンスの停止](#)」を参照してください。

Note

[インスタンスを再起動](#)しても、ライフサイクルイベントはトリガーされません。

アプリケーションコマンド Deploy and Undeploy の詳細については、「[アプリケーションのデプロイ](#)」を参照してください。

起動されたインスタンスの起動が終了した後の、残りの起動シーケンスは次のとおりです。

1. AWS OpsWorks スタックはインスタンスの組み込み Setup レシピが実行された後で、カスタム Setup レシピからアクセス可能になっています。
2. AWS OpsWorks スタックはインスタンスの組み込み Deploy レシピが実行された後で、カスタム Deploy レシピからアクセス可能になっています。

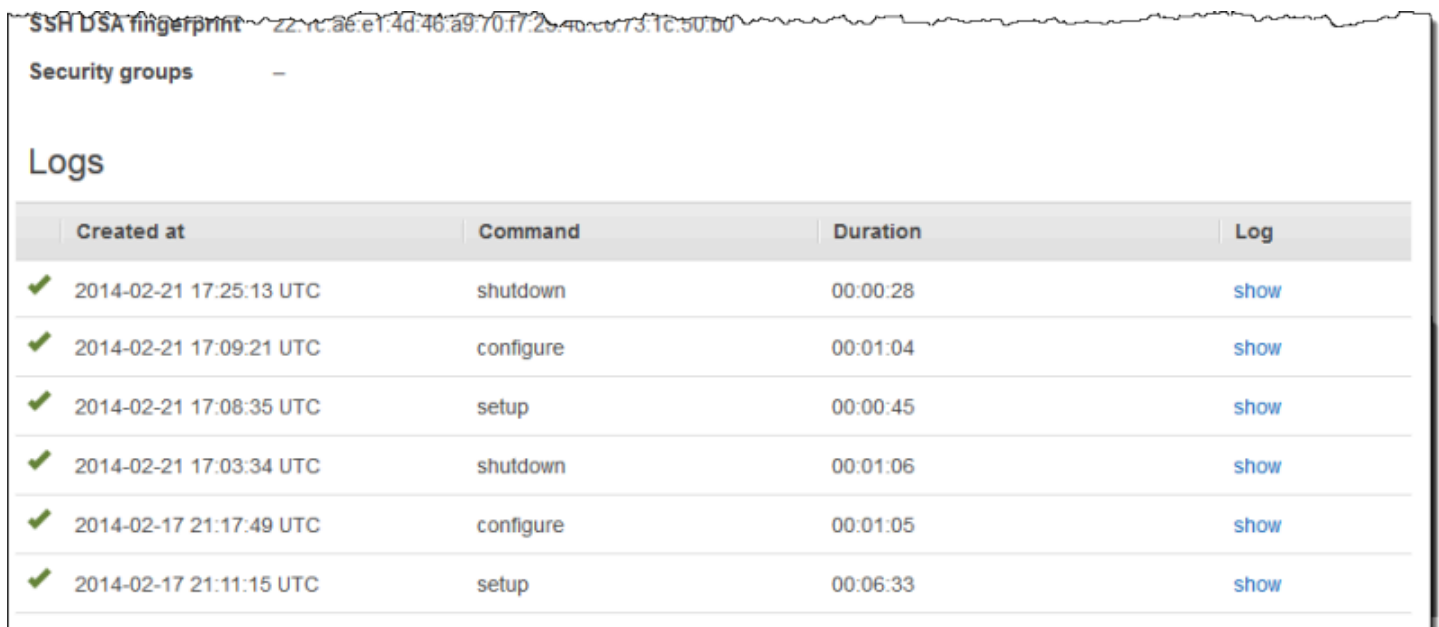
インスタンスがオンライン状態になります。

3. AWS OpsWorks スタックは をトリガーします。Configure イベントがトリガーされます。

AWS OpsWorks スタックはインスタンスの組み込み Configure レシピが実行された後で、カスタム Configure レシピからアクセス可能になっています。

Note

特定のインスタンスで発生したライフサイクルイベントを確認するには、[Instances] ページにアクセスし、そのインスタンスの名前をクリックして対応する詳細ページを開きます。ページ最下部の [Logs] セクションにイベントが一覧表示されます。[Log] (ログ) 列の [show] (表示) をクリックすると、イベントの Chef ログを確認できます。イベントの処理方法についての詳細な情報 (実行されたレシピを含む) が表示されます。Chef ログの解釈の詳細については、「[Chef ログ](#)」を参照してください。



Created at	Command	Duration	Log
✓ 2014-02-21 17:25:13 UTC	shutdown	00:00:28	show
✓ 2014-02-21 17:09:21 UTC	configure	00:01:04	show
✓ 2014-02-21 17:08:35 UTC	setup	00:00:45	show
✓ 2014-02-21 17:03:34 UTC	shutdown	00:01:06	show
✓ 2014-02-17 21:17:49 UTC	configure	00:01:05	show
✓ 2014-02-17 21:11:15 UTC	setup	00:06:33	show

各ライフサイクルイベントについて、AWS OpsWorks スタックは、現在の[スタック状態を含む各インスタンスに一連のスタック設定およびデプロイ属性](#)をインストールし、の場合は Deploy イベント、デプロイに関する情報。属性には、利用可能なインスタンスとその IP アドレスなどの情報が含まれています。詳細については、「[スタック設定およびデプロイメント属性](#)」を参照してください。

Note

多数のインスタスを同時に開始または停止すると、短時間のうちに大量の Configure イベント。不要な処理を避けるため、AWS OpsWorks スタックは最後のイベントにのみ応答します。変更範囲全体についてスタックのインスタスを更新するために必要な情報はすべて、そのイベントのスタック設定とデプロイメント属性に保存されています。したがって、先行する Configure events. AWS OpsWorks Stacks は未処理の にラベルを付けます。Configure イベントは、&OPS; スタックによって superseded と分類されます。

レシピを自動的に実行する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

それぞれのレイヤーには、ライフサイクルイベントごとに割り当てられた一連の組み込みレシピがあります (ただし、Undeploy レシピがないレイヤーも一部存在します)。インスタンスでライフサイクルイベントが発生すると、AWS OpsWorks Stacks は関連するレイヤーの適切なレシピのセットを実行します。

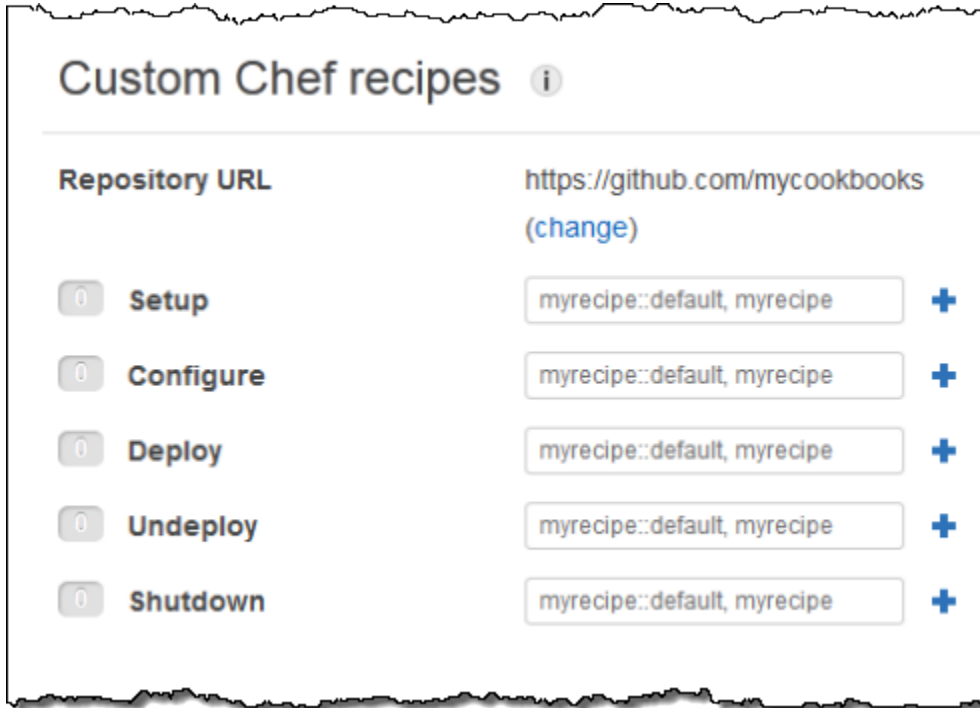
カスタムクックブックをインストールしている場合は、各レシピをレイヤーのライフサイクルイベントに割り当てることで、AWS OpsWorks スタックでレシピの一部またはすべてを自動的に実行させることができます。イベントが発生すると、AWS OpsWorks スタックはレイヤーの組み込みレシピの後に指定されたカスタムレシピを実行します。

カスタムレシピをレイヤーのイベントに割り当てるには

1. [Layers] (レイヤー) ページで、目的のレイヤーの [Recipes] (レシピ) をクリックし、[Edit] (編集) をクリックします。まだカスタムクックブックを有効にしていない場合は、[configure cookbooks] をクリックして、スタックの [Settings] ページを開きます。[Use custom Chef Cookbooks] を [Yes] に切り替え、クックブックのリポジトリ情報を指定します。その後、

[Save] をクリックし、[Recipes] タブの編集ページに戻ります。詳細については、「[カスタムクックブックのインストール](#)」を参照してください。

- [Recipes] タブで、目的のイベントフィールドに個々のカスタムレシピを入力し、[+] をクリックしてリストに追加します。レシピを次のように指定します：`cookbook:somerecipe` (`.rb` 拡張子は省略します)。



新しいインスタンスを起動すると、AWS OpsWorks スタックは標準レシピを実行した後、各イベントのカスタムレシピを自動的に実行します。

Note

カスタムレシピは、コンソールに入力した順序で実行されます。また、適切な順序でレシピを実行するメタレシピを実装することにより実行順序を制御することもできます。

レシピを手動で実行する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソ

リユーシオンに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

通常レシピは、ライフサイクルイベントの発生時に自動的に実行されますが、スタックの特定のまたはすべてのインスタンスに対し、いつでも手動でレシピを実行することができます。この機能は通常、インスタンスのバックアップなど、ライフサイクルイベントとはあまり関係のないタスクに使用します。カスタムレシピを手動で実行するには、そのカスタムレシピがいずれかのカスタムクックブックに含まれている必要があります。ただし、カスタムレシピがライフサイクルイベントに割り当てられている必要はありません。レシピを手動で実行すると、AWS OpsWorks スタックは Deploy イベントと同じdeploy属性をインストールします。

スタックのインスタンスに対してレシピを手動で実行するには

1. [Stack] ページで、[Run command] をクリックします。[Command] の [Execute Recipes] を選択します。

Run Command

Settings

Command

Recipes to execute

Comment

Custom Chef JSON

Enter custom JSON that is passed to your Chef recipes for all instances in your stack. You can use this to override and customize built-in recipes or pass variables to your own. [Learn more.](#)

Instances ⓘ

No running instances with the OpsWorks status online or setup_failed. Start [instances](#) now.

[Cancel](#) [Execute Recipes](#)

2. 標準の `cookbookname::recipe` 形式を使用して、実行するレシピを実行するレシピボックスに入力します。複数のレシピはカンマで区切って指定します。ここに指定した順序でレシピが実行されます。
3. 必要に応じて、カスタム Chef JSON ボックスを使用して、インスタンスにインストールされているスタック設定およびデプロイ属性にマージされるカスタム属性を定義するカスタムJSONオブジェクトを追加します。カスタムJSONオブジェクトの使用の詳細については、[カスタムの使用 JSON](#)「」および「」を参照してください[属性の上書き](#)。
4. インスタンスで、AWS OpsWorks スタックがレシピを実行するインスタンスを選択します。

ライフサイクルイベントが発生すると、AWS OpsWorks スタックエージェントは関連するレシピを実行するコマンドを受け取ります。これらのコマンドは、適切な[スタックコマンドを使用する](#)か、エージェントのコマンドを使用して、特定のインスタンスで手動で実行できますCLI `run_command`。エージェントの使用方法の詳細についてはCLI、「」を参照してください[AWS OpsWorks スタックエージェント CLI](#)。

リソース管理

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

リソースページでは、AWS OpsWorks スタックスタックでアカウントの [Elastic IP アドレス](#)、[Amazon EBS ボリューム](#)、または [Amazon RDS](#) インスタンスリソースを使用できます。`[Resources]` を使用して、以下を行うことができます。

- [リソースの登録](#)。リソースをスタックに登録することにより、リソースをスタックのインスタンスの 1 つにアタッチできます。
- [リソースのアタッチ](#)。リソースをスタックのインスタンスの 1 つにアタッチします。
- [リソースの移動](#)。あるインスタンスから別のインスタンスにリソースを移動します。
- [リソースのデタッチ](#)。リソースをインスタンスからデタッチします。リソースは登録された状態で残っているため、別のインスタンスにアタッチすることができます。

- [リソースの登録解除](#)。未登録のリソースは AWS OpsWorks スタックでは使用できませんが、削除しない限りアカウントに残り、別のスタックに登録できます。

以下の制約があることに注意してください。

- 登録済みの Amazon EBS ボリュームを Windows インスタンスにアタッチすることはできません。
- リソースページでは、スタンダード、PIOPS、スループット最適化 HDD、Cold HDD、または汎用 (SSD) Amazon EBS ボリュームを管理できますが、RAID アレイは管理できません。
- Amazon EBS ボリュームは、XFS フォーマットである必要があります。

AWS OpsWorks スタックは、ext4 などの他のファイル形式をサポートしていません。Amazon EBS ボリュームの準備に関する詳細については、[「Making an Amazon EBS Volume Available for Use」](#) (Amazon EBS ボリュームを使用可能にする) を参照してください。

- 実行中のインスタンスに対して、Amazon EBS ボリュームをアタッチすることもデタッチすることもできません。

オフラインインスタンスでのみ操作できます。たとえば、使用中のボリュームをスタックに登録してオフラインのインスタンスにアタッチできます。ただし、新しいインスタンスを開始する前に、元のインスタンスを停止して、そのボリュームをデタッチする必要があります。そうしないと、起動プロセスが失敗します。

- 登録されたすべてのリソースは、でのみ管理されます AWS OpsWorks。これにより、EC2 ボリュームの DeleteOnTermination などのリソースライフサイクルプロパティを上書きできます。
- Elastic IP アドレスは、実行中のインスタンスに対してアタッチすることもデタッチすることもできます。

オンラインのインスタンスでもオフラインのインスタンスでも操作できます。例えば、使用中のアドレスを登録して実行中のインスタンスに割り当てると、AWS OpsWorks スタックは自動的にアドレスを再割り当てします。

- リソースの登録または登録の解除を行うには、IAM ポリシーで以下のアクションに対するアクセス権限を付与する必要があります。

Amazon EBS ボリューム	Elastic IP アドレス	Amazon RDS インスタンス
RegisterVolume	RegisterElasticIp	RegisterRdsDbInstance

Amazon EBS ボリューム	Elastic IP アドレス	Amazon RDS インスタンス
UpdateVolume	UpdateElasticIp	UpdateRdsDbInstance
DeregisterVolume	DeregisterElasticIp	DeregisterRdsDbInstance

[Manage 権限レベル](#)は、これらのアクションのすべてに対してアクセス許可を付与します。Manage ユーザーが特定リソースの登録や登録解除をできなくするには、該当するアクションへのアクセス許可を拒否するように IAM ポリシーを編集します。詳細については、「[セキュリティと権限](#)」を参照してください。

トピック

- [スタックにリソースを登録する](#)
- [リソースのアタッチと移動を行う](#)
- [リソースをデタッチする](#)
- [リソースの登録を解除する](#)

スタックにリソースを登録する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Amazon EBS ボリュームまたは Elastic IP アドレスは、インスタンスにアタッチする前に、スタックに登録する必要があります。AWS OpsWorks スタックがスタックのリソースを作成すると、スタックに自動的に登録されます。外部で作成されたリソースを使用する場合は、明示的にリソースを登録する必要があります。次の点に注意してください。

- リソースは、一度に 1 つのスタックにしか登録できません。
- スタックを削除すると、AWS OpsWorks スタックはすべてのリソースの登録を解除します。

トピック

- [スタックを使用して Amazon EBS ボリュームを登録する](#)
- [スタックに Elastic IP アドレスを登録する](#)
- [スタックに Amazon RDS インスタンスを登録する](#)

スタックを使用して Amazon EBS ボリュームを登録する

Note

このリソースは Linux スタックにのみ使用できます。Windows スタックを使用して Amazon EBS ボリュームを登録できますが、インスタンスにアタッチすることはできません。

[Resources (リソース)] ページを使用して、Amazon EBS ボリュームをスタックに登録できます。このとき、次のような制限があります。

- アタッチされた非ルート Amazon EBS ボリュームは、スタンダード、スループット最適化 HDD、Cold HDD、PIOPS、または汎用 (SSD) である必要がありますが、RAID アレイである必要はありません。最大および最小ボリュームサイズの詳細については、このガイドの「[EBS ボリューム](#)」を参照してください。
- ボリュームは、XFS フォーマットである必要があります。
- AWS OpsWorks スタックは、非ルート Amazon EBS ボリュームに対して、ext4 (fourth extended file system) などその他のファイル形式をサポートしません。Amazon EBS ボリュームの準備に関する詳細については、「[Making an Amazon EBS Volume Available for Use](#)」(Amazon EBS ボリュームを使用可能にする) を参照してください。そのトピックの例では、ext4 ベースのボリュームを作成する方法について説明していますが、XFS ベースのボリュームと同じ手順に従うことができます。

Amazon EBS ボリュームを登録するには

1. 目的のスタックを開き、ナビゲーションペインで、[Resources] をクリックします。
2. [Volumes] (ボリューム) をクリックして、使用可能な Amazon EBS ボリュームを表示します。次の図に示すように、最初は、スタックに登録されているボリュームはありません。

Resources

[Show Unregistered Volumes](#)
[Volumes](#)
[Elastic IPs](#)
[RDS](#)

No volumes have been registered yet. [Show unregistered volumes.](#)

- [Show Unregistered Volumes] (未登録のボリュームの表示) をクリックして、スタックのリージョンおよびスタックの VPC (該当する場合) にあるアカウントの Amazon EBS ボリュームを表示します。[Status] 列には、ボリュームが使用できるかが示されます。[ボリュームタイプ] には、ボリュームがスタンダード (standard)、汎用 SSD (gp2)、PIOPS (io1、後ろの括弧内にディスクごとの IOPS 値を表示)、スループット最適化 HDD (st1)、または Cold HDD (sc1) のいずれであるかを示します。

Resources Unregistered Volumes

[Volumes](#)
[Elastic IPs](#)
[RDS](#)

The list contains only volumes created in **us-east-1**. Add a Volume on **EC2**.

<input type="checkbox"/>	Name	EC2 ID	EC2 Instance ID	Size (GiB)	Device	Volume Type	AZ	Status
<input type="checkbox"/>	Disk 1 of 2	vol-3753f475		50		standard	us-east-1a	available
<input type="checkbox"/>	Disk 2 of 2	vol-eb54f3a9		50		standard	us-east-1a	available
<input type="checkbox"/>	PHP-LB-Standard	vol-6a4bec28		100		standard	us-east-1a	available
<input type="checkbox"/>	no name	vol-68702625	i-9a5328ba	8	/dev/sda1	standard	us-east-1c	in-use

[Cancel](#)
[Register with Stack](#)

- 登録するボリュームを選択して、[Register to Stack] をクリックします。[Resources] ページには、新規に登録したボリュームのリストが表示されます。

Resources

[Show Unregistered Volumes](#)
[Volumes](#)
[Elastic IPs](#)
[RDS](#)

Name	EC2 ID	Instance	Size (GiB)	Volume Type	AZ	Actions
PHP-LB-Standard	vol-6a4bec28	assign to instance	100	standard	us-east-1a	edit

[+ Unregistered Volumes](#)

ボリュームを追加登録する場合は、[Show Unregistered Volumes] または [+ Unregistered Volumes] をクリックし、同じ手順を繰り返します。

スタックに Elastic IP アドレスを登録する

Elastic IP アドレスを登録するには、次の手順を使用します。

Elastic IP アドレスを登録するには

1. スタックの [Resources] ページを開き、[Elastic IPs] をクリックして、使用可能な Elastic IP アドレスを表示します。次の図に示すように、最初は、スタックに登録されているアドレスはありません。

Resources

[Show Unregistered Elastic IPs](#)

Volumes

Elastic IPs

RDS

Search

No Elastic IPs have been registered yet. [Show unregistered Elastic IPs.](#)

2. [Show Unregistered Elastic IPs] をクリックして、スタックのリージョンにあるアカウントで使用できる Elastic IP アドレスを表示します。

Resources Unregistered Elastic IPs

Volumes

Elastic IPs

RDS

Search

The list contains only Elastic IPs created in **us-east-1** in **standard** domain. Add an Elastic IP on **EC2**.
You can register an Elastic IP that is currently associated with an instance, OpsWorks will not change the association until you disassociate the IP or swap it.

<input type="checkbox"/>	Address	Instance	Domain
<input type="checkbox"/>	192.0.2.0		standard
<input checked="" type="checkbox"/>	192.0.2.10		standard
<input type="checkbox"/>	192.0.2.20		standard

Cancel

[Register with Stack](#)

3. 登録するアドレスを選択して、[Register to Stack] をクリックします。これによって、[Resources] ページに戻り、新規に登録されたアドレスのリストが表示されます。

Resources

[Show Unregistered Elastic IPs](#)

Volumes	Elastic IPs	RDS	Search		
Address	Name	Instance	Public DNS	Actions	
192.0.2.0	-	associate with instance	-	edit	
+ Unregistered Elastic IPs					

アドレスを追加登録する場合は、[Show Unregistered Elastic IPs] または [+ Unregistered Elastic IPs] をクリックし、同じ手順を繰り返します。

スタックに Amazon RDS インスタンスを登録する

Amazon RDS インスタンスを登録するには、以下の手順を使用します。

Amazon RDS インスタンスを登録するには

1. スタックの [Resources] (リソース) ページを開き、[RDS] をクリックして、使用できる Amazon RDS インスタンスを表示します。次の図に示すように、最初は、スタックに登録されたインスタンスはありません。

Resources

[Show Unregistered RDS DB instances](#)

Volumes	Elastic IPs	RDS	Search		
No RDS DB instances have been registered yet. Show unregistered RDS DB instances.					

2. [Show Unregistered RDS DB instances] (登録されていないRDS DBインスタンスの表示) をクリックして、スタックのリージョンにあるアカウントで使用できる Amazon RDS インスタンスを表示します。

Resources Unregistered RDS DB instances

Volumes Elastic IPs **RDS**

The list contains only RDS DB instances created in **us-east-1**. Add an instance on **RDS**.

Instance Identifier	Engine	Storage (GB)	Type	Status	Multi-AZ	Availability Zone
<input checked="" type="radio"/> opsinstance1	mysql	5	t1.micro	available	No	us-east-1d
<input type="radio"/> opsinstance2	mysql	5	t1.micro	available	No	us-east-1d

Connection Details for opsinstance1

User:

Password: [SHOW](#)

Your **RDS DB instance** must accept connections from your OpsWorks instances. [Learn more.](#)

[Cancel](#) [Register with Stack](#)

- 登録するインスタンスを選択し、[User] および [Password] にそのインスタンスのマスターユーザー値およびマスターパスワード値を入力します。次に、[Register to Stack] をクリックします。これによって、[Resources] ページに戻り、新規に登録されたインスタンスのリストが表示されます。

Resources

[Show Unregistered RDS DB instances](#)

Volumes Elastic IPs **RDS**

Instance Identifier	Engine	Apps	Type	Multi-AZ	AZ	Actions
opsinstance1	mysql	Add app	t1.micro	No	us-east-1d	edit

[+ Unregistered RDS DB instances](#)

⚠ Important

Amazon RDS インスタンスを登録するときに使用するユーザーとパスワードが、有効なユーザーとパスワードに確実に対応していることを確実にする必要があります。対応していない場合、アプリケーションからインスタンスに接続できません。

インスタンスを追加登録する場合は、[Show Unregistered RDS DB instances] または [+ Unregistered RDS DB instances] をクリックして、同じ手順を繰り返します。AWS OpsWorks スタックで Amazon RDS インスタンスを使用する方法の詳細については、「」を参照してください [Amazon RDS Service Layer](#)。

Note

Amazon RDS インスタンスは、[Layers] (レイヤー) ページを通じて登録することもできます。詳細については、「[Amazon RDS Service Layer](#)」を参照してください。

リソースのアタッチと移動を行う

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

リソースは、スタックに登録されると、スタックのインスタンスの 1 つにアタッチできるようになります。アタッチされたリソースは、あるインスタンスから別のインスタンスに移動することもできます。次の点に注意してください。

- Amazon EBS ボリュームをアタッチまたは移動する場合、オペレーションに関わるインスタンスはオフラインになっている必要があります。該当するインスタンスが [Resources] ページにない場合には [Instances] ページに移動して、[インスタンスを停止](#)します。インスタンスを停止すると、[Resources] ページに戻って、リソースのアタッチまたは移動ができます。
- Elastic IP アドレスをアタッチまたは移動すると、インスタンスをオンラインやオフラインにすることができます。
- インスタンスを削除しても、アタッチされているリソースはスタックに登録されたまま残っています。このリソースは別のインスタンスにアタッチすることができます。また、このリソースがもう必要ない場合には、リソースの登録を解除できます。

トピック

- [Amazon EBS ボリュームをインスタンスに割り当てる](#)
- [Elastic IP アドレスをインスタンスに関連付ける](#)
- [Amazon RDS インスタンスをアプリケーションにアタッチする](#)

Amazon EBS ボリュームをインスタンスに割り当てる

Note

Windows インスタンスに Amazon EBS ボリュームを割り当てることはできません。

登録されている Amazon EBS ボリュームをインスタンスに割り当てて、あるインスタンスから別のインスタンスに移動することができますが、両方のインスタンスがいずれもオフラインである必要があります。

インスタンスに Amazon EBS ボリュームを割り当てるには

1. [Resources] ページで、該当するボリュームの [Instance] 列で [assign to instance] をクリックします。

Resources

[Show Unregistered Volumes](#)[Volumes](#)[Elastic IPs](#)

Name	EC2 ID	Instance	Size (GiB)	Volume Type	AZ	Actions
Created for db-master1	vol-24ac9267	db-master1 ●	10	standard	us-east-1a	
PHP-LB-PIOPs	vol-0faf914c	assign to instance	100	io1 (2000)	us-east-1a	edit
PHP-LB-Standard	vol-53af9110	assign to instance	100	standard	us-east-1a	edit

[+ Unregistered Volumes](#)

2. ボリュームの詳細ページで、適切なインスタンスを選択し、ボリューム名とマウントポイントを指定します。次に、[Save] をクリックしてボリュームをインスタンスにアタッチします。

Volume PHP-LB-PIOPs

Name	PHP-LB-PIOPs
EC2 Volume ID	vol-0faf914c
Mount point	/vol/mountpoint
Availability Zone	us-east-1a
Instance	-
Status	<i>PHP App Server</i> php-app1 <i>Unassigned</i>
Size	100 GiB
Device	-
Volume Type	io1
IOPS	2000
Snapshot ID	-
OpsWorks ID	a402f9f9-6814-403d-8b2d-dfee98950e9c

Cancel

Save

Important

使用中の外部ボリュームをユーザーのインスタンスに割り当てた場合には、Amazon EC2 コンソール、API、または CLI を使用して、元のインスタンスに対するボリュームの割り当てを解除する必要があります。そうしないと、起動プロセスが失敗します。

詳細ページを使用して、割り当てられた Amazon EBS ボリュームをスタックの別のインスタンスに移動することもできます。

Amazon EBS ボリュームを別のインスタンスに移動するには

1. 両方のインスタンスがオフライン状態になっていることを確認します。
2. [Resources] (リソース) ページで、[Volumes] (ボリューム) をクリックして、ボリュームの [Actions] (アクション) 列で [edit] (編集) をクリックします。
3. 次のいずれかを行います。

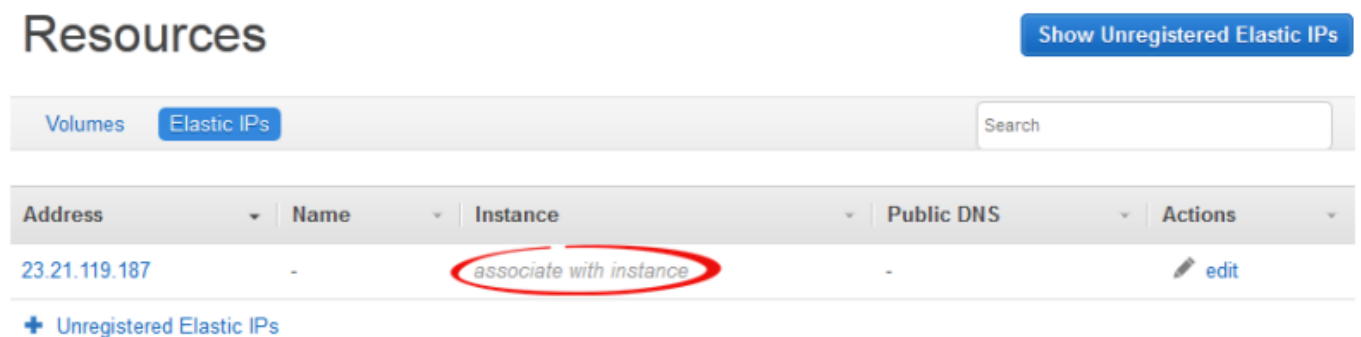
- ボリュームを同じスタックの別のインスタンスに移動するには、[Instance] リストから移動するインスタンスを選択し、[Save] をクリックします。
- ボリュームを別のスタックのインスタンスに移動するには、[ボリュームの登録を解除して、新しいスタックにボリュームを登録し](#)、さらに新しいインスタンスに[ボリュームをアタッチ](#)します。

Elastic IP アドレスをインスタンスに関連付ける

登録されている Elastic IP アドレスをインスタンスに関連付けて、あるインスタンスから別のインスタンス (他のスタックのインスタンスを含む) に移動することができます。この場合のインスタンスは、オンラインでもオフラインでもかまいません。

Elastic IP アドレスをインスタンスに関連付けるには

1. [Resources] ページで、該当するアドレスの [Instance] 列で [associate with instance] をクリックします。



The screenshot shows the 'Resources' page in AWS OpsWorks. At the top right, there is a button labeled 'Show Unregistered Elastic IPs'. Below this, there are tabs for 'Volumes' and 'Elastic IPs', with 'Elastic IPs' being the active tab. A search bar is located to the right of the tabs. The main content is a table with the following columns: 'Address', 'Name', 'Instance', 'Public DNS', and 'Actions'. The first row of data has the address '23.21.119.187', a hyphen in the 'Name' column, and 'associate with instance' in the 'Instance' column. The 'associate with instance' text is circled in red. In the 'Actions' column, there is an 'edit' link with a pencil icon. Below the table, there is a link '+ Unregistered Elastic IPs'.

2. アドレスの詳細ページで、目的のインスタンスを選択し、アドレスの名前を指定します。次に、[Save] をクリックして、アドレスをインスタンスに関連付けます。

Elastic IP 23.21.119.187

IP	23.21.119.187
Name	<input type="text" value="PHP-EIP"/>
Region	us-east-1
Domain	standard
Stack	MyStack change..
Instance	<div style="border: 1px solid #ccc; padding: 2px;"><div style="border-bottom: 1px solid #ccc; padding: 2px;">-</div><div style="padding: 2px;"><i>PHP App Server</i></div><div style="padding: 2px;">php-app1</div><div style="padding: 2px;">php-app2</div><div style="padding: 2px;">php-app3</div><div style="padding: 2px;"><i>Not associated</i></div><div style="border-bottom: 1px solid #ccc; padding: 2px;">-</div></div> Select the instance the Elastic IP should be associated with.

Note

Elastic IP アドレスが現在別のオンラインインスタンスに関連付けられている場合、AWS OpsWorks Stacks はそのアドレスを新しいインスタンスに自動的に再割り当てします。

詳細ページを使用して、関連付けられた Elastic IP アドレスを別のインスタンスに移動することもできます。

Elastic IP アドレスを別のインスタンスに移動するには

1. [Resources] ページで、[Elastic IPs] をクリックし、アドレスの [アクション] 列で [編集] をクリックします。
2. 次のいずれかを行います。
 - アドレスを同じスタックの別のインスタンスに移動する場合は、[Instance] リストから目的のインスタンスを選択し、[Save] をクリックします。
 - アドレスを別のスタックのインスタンスに移動するには、スタックの設定の変更をクリックすると、利用可能なスタックのリストが表示されます。[Stack] リストからスタックを選択し、[Instance] リストからインスタンスを選択します。次に、[Save] をクリックします。

Elastic IP PHP-EIP1

IP	54.221.232.99
Name	<input type="text" value="PHP-EIP1"/>
Region	us-east-1
Domain	standard
Stack	MyStack change.
Instance	<input type="text" value="php-app1 [current]"/>

アドレスをアタッチまたは移動すると、AWS OpsWorks スタックは [Configure ライフサイクルイベント](#) をトリガーして、スタックのインスタンスに変更を通知します。

Amazon RDS インスタンスをアプリケーションにアタッチする

Amazon RDS インスタンスを 1 つ以上のアプリケーションにアタッチすることができます。

Amazon RDS インスタンスをアプリケーションにアタッチするには

1. [Resources] ページで、目的のインスタンスの [Apps] 列で [Add app] をクリックします。

Resources

[Show Unregistered RDS DB instances](#)

Volumes	Elastic IPs	RDS	<input type="text" value="Search"/>			
Instance Identifier	Engine	Apps	Type	Multi-AZ	AZ	Actions
opsinstance1	mysql	Add app	t1.micro	No	us-east-1d	edit
+ Unregistered RDS DB instances						

2. [Add App] (アプリケーションの追加) ページを使用して、Amazon RDS インスタンスをアタッチします。詳細については、「[アプリケーションの追加](#)」を参照してください。

Amazon RDS は、複数のアプリケーションにアタッチできるため、あるアプリケーションから別のアプリケーションにインスタンスを移動するための特別な手順はありません。移動元のアプリケー

ションを編集して RDS インスタンスを削除し、移動先のアプリケーションを編集して RDS インスタンスを追加するだけです。詳細については、「[アプリケーションの編集](#)」を参照してください。

リソースをデタッチする

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

アタッチされているリソースがもう必要ない場合には、デタッチすることができます。このリソースはスタックに登録されたまま残るため、どこにでもアタッチすることができます。

トピック

- [Amazon EBS ボリュームの割り当て解除](#)
- [Elastic IP アドレスの関連付けを解除する](#)
- [Amazon RDS インスタンスのデタッチする](#)

Amazon EBS ボリュームの割り当て解除

Amazon EBS ボリュームのインスタンスからの割り当てを解除するには、次の手順を使用します。

Amazon EBS ボリュームの割り当てを解除するには

1. インスタンスがオフライン状態になっていることを確認します。
2. [Resources] ページで、[Volumes] をクリックし、ボリューム名をクリックします。
3. ボリュームの詳細ページで、[Unassign] をクリックします。

Volume PHP-LB-PIOPs

[Edit](#)[Unassign](#)

Volumes are the block level storage associated with your instance. [Learn more.](#)

Settings

Name	PHP-LB-PIOPs
EC2 Volume ID	vol-0faf914c
Mount point	/vol/mountpoint
Availability Zone	us-east-1a
Instance	php-app1 ●
Status	available
Size	100 GiB
Device	/dev/sdi
Volume Type	io1
IOPS	2000
Snapshot ID	–
OpsWorks ID	a402f9f9-6814-403d-8b2d-dfee98950e9c

Elastic IP アドレスの関連付けを解除する

インスタンスへの Elastic IP アドレスの関連付けを解除するには、次の手順を使用します。

Elastic IP アドレスの関連付けを解除するには

1. [Resources] ページで、[Elastic IPs] をクリックし、アドレスの[アクション] 列で [編集] をクリックします。
2. アドレスの詳細ページで [Disassociate] をクリックします。

Elastic IP PHP-Vol2

[Edit](#)[Disassociate](#)

Elastic IPs are static IP addresses for your instance. [Learn more.](#)

Settings

IP	23.21.119.187
Name	PHP-Vol2
Region	us-east-1
Domain	standard
Instance	php-app1 ●

アドレスの関連付けを解除すると、AWS OpsWorks スタックは [Configure ライフサイクルイベント](#) をトリガーして、スタックのインスタンスに変更を通知します。

Amazon RDS インスタンスのデタッチする

アプリケーションから Amazon RDS をデタッチするには、以下の手順を使用します。

Amazon RDS インスタンスをデタッチするには

1. [Resources] ページで、[RDS] をクリックし、[Apps] 列にある目的のアプリケーションをクリックします。
2. [Edit] をクリックし、アプリケーション設定を編集してインスタンスをデタッチします。詳細については、「[アプリケーションの編集](#)」を参照してください。

Note

この手順で、1つのアプリケーションから Amazon RDS をデタッチします。インスタンスが複数のアプリケーションにアタッチされている場合には、各アプリケーションに対してこの手順を繰り返します。

リソースの登録を解除する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

スタックにリソースを登録しておく必要がない場合には、リソースの登録を解除することができます。登録解除しても、アカウントからリソースは削除されません。リソースはそのまま残り、別のスタックに登録することも、AWS OpsWorks スタックの外部で使用することもできます。リソースを完全に削除するには、2つの方法があります。

- Elastic IP または Amazon EBS のリソースがインスタンスにアタッチされている場合、インスタンスを削除する際にリソースを削除することができます。

[Instances] (インスタンス) ページに移動し、インスタンスの [Actions] (アクション) 列で [delete] (削除) をクリックし、その後 [Delete instance's EBS volumes] (インスタンスのEBSボリュームを削除) または [Delete the instance's Elastic IP] (インスタンスのElastic IPを削除) を選択します。

- リソースの登録を解除し、さらに、Amazon EC2 が Amazon RDS のコンソール、API、または CLI を使用してリソースを削除します。

トピック

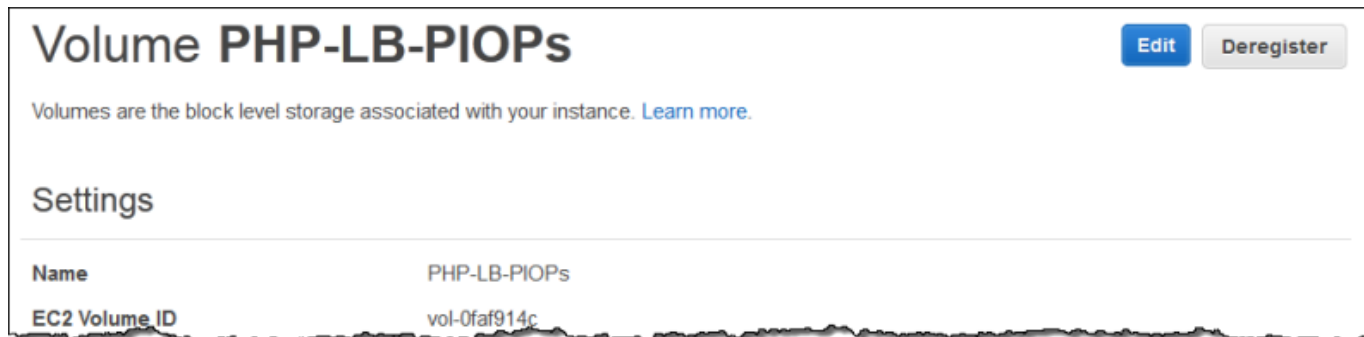
- [Amazon EBS ボリュームの登録を解除する](#)
- [Elastic IP アドレスの登録を解除する](#)
- [Amazon RDS インスタンスの登録を解除する](#)

Amazon EBS ボリュームの登録を解除する

Amazon EBS ボリュームの登録を解除するには、以下の手順を使用します。

Amazon EBS ボリュームの登録を解除するには

1. ボリュームがインスタンスにアタッチされている場合には、「[Amazon EBS ボリュームの割り当て解除](#)」の記載に従って割り当てを解除します。
2. [Resources] ページで、[Name] 列のボリューム名をクリックします。
3. ボリュームの詳細ページで、[Deregister] をクリックします。



Elastic IP アドレスの登録を解除する

Elastic IP アドレスの登録を解除するには、次の手順を使用します。

Elastic IP アドレスの登録を解除するには

1. アドレスがインスタンスに関連付けられている場合は、「[Elastic IP アドレスの関連付けを解除する](#)」の記載に従って、関連付けを解除します。
2. [Resources] ページで、[Elastic IPs] をクリックし、[Address] 列で IP アドレスをクリックします。
3. アドレスの詳細ページで、[Deregister] をクリックします。

Elastic IP PHP-Vol2

Edit Deregister

Elastic IPs are static IP addresses for your instance. [Learn more.](#)

Settings

IP	23.21.119.187
Name	PHP-Vol2
Region	us-east-1
Domain	standard
Instance	associate with instance

Note

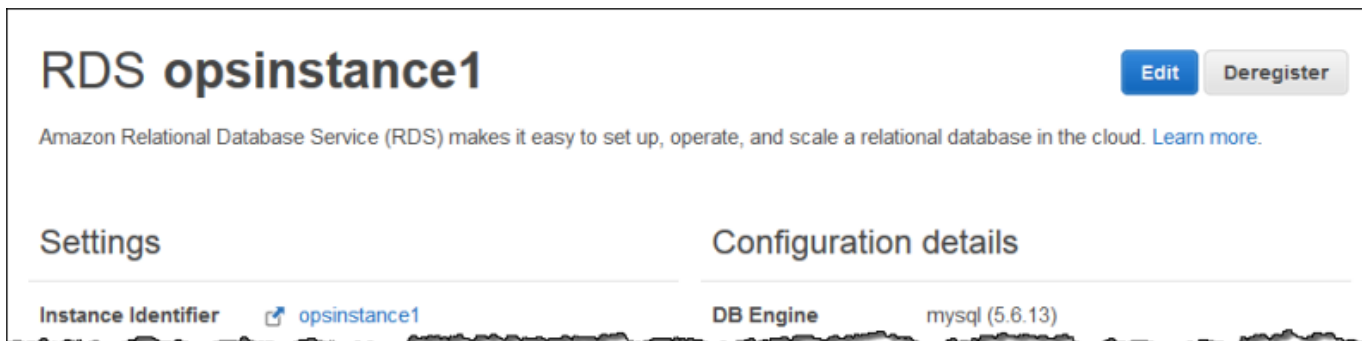
単に、別のスタックに Elastic IP アドレスを登録する場合でも、現在のスタックへの登録を解除してから新しいスタックに登録する必要があります。ただし、アタッチされている Elastic IP アドレスは、別のスタックのインスタンスに直接移動することができます。詳細については、「[リソースのアタッチと移動を行う](#)」を参照してください。

Amazon RDS インスタンスの登録を解除する

Amazon RDS インスタンスの登録を解除するには、以下の手順を使用します。

Amazon RDS インスタンスの登録を解除するには

1. インスタンスがアプリケーションに関連付けられている場合は、「[リソースをデタッチする](#)」の記載に従って、インスタンスをデタッチします。
2. [Resources] ページで、[RDS] をクリックし、さらにインスタンス名をクリックします。
3. インスタンスの詳細ページで、[Deregister] をクリックします。



タグ

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

タグによって、Chef 11.10、Chef 12、Chef 12.2 のスタックでのリソースのグループ化と、[AWS Billing and Cost Management](#)でこれらのリソースを使用するコストの追跡が容易になります。

スタックおよびレイヤーレベルでタグを適用することができます。タグを作成する場合、タグ付けされた構造内すべてのリソースにタグを適用することになります。例えば、あるレイヤーにタグを適用すると、そのレイヤーのすべてのインスタンス、Amazon EBS ボリューム (ルートを除く)、または Elastic Load Balancing ロードバランサーにタグが適用されます。タグは現在、インスタンスのルートまたはデフォルトの EBS ボリュームには適用できません。

タグは、スタックのスタックまたはレイヤーに割り当てるキーと値のペア AWS OpsWorks です。タグを作成した後、Management コンソールを開いてユーザー定義のタグを有効化します。タグをアクティブ化して AWS OpsWorks スタックリソースのコストを追跡および管理する方法の詳細については、請求情報とコスト管理ユーザーガイドの「[コスト配分タグの使用](#)」と「[ユーザー定義のコスト配分タグのアクティブ化](#)」を参照してください。

タグは、AWS OpsWorks スタックのカスタム属性と同様の方法で機能します。スタックに適用するタグは、スタックの各レイヤーによって継承されます。レイヤーレベルでは、継承されたタグの値 (キー名ではなく) を上書きし、新しいレイヤー固有のタグを追加できます。は、結果のタグセットをレイヤー内のすべてのリソース AWS OpsWorks に適用します。リソースを新規作成または既存のリソースをレイヤーに割り当てるとき、そのレイヤーの新しいリソースは、同じタグでタグ付けされます。

トピック

- [スタックレベルでタグを設定](#)
- [レイヤーレベルでのタグの設定](#)
- [を使用したタグの管理 AWS CLI](#)
- [タグの制限](#)

スタックレベルでタグを設定

スタックレベルでは、スタックのホームページで Tags を選択してタグを追加および管理できます。

MyStack

[Run Command](#)
[Stack Settings](#)
[Delete Stack](#)

A stack represents a collection of EC2 instances and related AWS resources that have a common purpose and that you want to manage collectively. Within a stack, you use layers to define the configuration of your instances and use apps to specify the code you want to deploy. [Learn more.](#)

Layers

1

MyLayer

Instances

1

1

online

0

setting up

0

shutting
down

0

stopped

0

error

Apps

1

PHPTestApp

deploy

Deployments and Commands

5

- ✓ 2 months ago [C](#)
- ✓ 9 months ago AWS-CodePipeline-Service/14... [C](#)
- ✓ 9 months ago AWS-CodePipeline-Service/14... [C](#)
- ✓ A year ago AWS-CodePipeline-Service/1484... [C](#)

Resources



The Resources page enables you to use any of your account's Elastic IP addresses, volumes, or RDS instances in your stack.

[Register resources](#)


AWS OpsWorks uses Amazon CloudWatch to provide thirteen custom metrics with detailed monitoring for each instance in the stack.

[Show monitoring](#)

Permissions



Permissions specify how imported IAM users can access this stack. To import users, go to the [Users](#) page.

[Manage permissions](#)

Tags NEW



You can specify tags to apply to resources in the stack. Tags can help you identify resources in cost allocation reports.

[Manage stack tags](#)

Tags ページで、タグをキーと値のペアとして追加します。次のスクリーンショットでは、タグの例をいくつか示しています。タグを削除するには、キーと値のペアの右側にある赤色の X を選択します。

Tags

Tags specified here will be applied to all resources in the stack. To apply tags only to resources in specific layers, visit the Tags section of the [Layers](#) page.

You must activate tags in the [Billing and Cost Management console](#) before they will appear in cost allocation reports. [Learn more](#).

Key (127 characters maximum)	Value (255 characters maximum)	
<input type="text" value="Organization"/>	<input type="text" value="Mobile"/>	✖
<input type="text" value="Staging"/>	<input type="text" value="Demo"/>	✖
<input type="text" value="Add key"/>	<input type="text" value="Add value (optional)"/>	








[Cancel](#) [Save](#)

レイヤーレベルでのタグの設定

レイヤーレベルで、[Tags] (タグ) タブを選択してタグを設定します。このタブは、レイヤーのホームページ、および各レイヤーのホームページにあります。

Layers ③

Add layer

 ELB: dd dd-1207428707.us-west-2.elb.amazonaws.com	Health 6
 HAProxy Settings Recipes Network EBS Volumes Security CloudWatch Logs Tags Delete	Instances 6
 Rails App Server Settings Recipes Network EBS Volumes Security CloudWatch Logs Tags Delete	Instances 18
 ELB: PHP-LB PHP-LB-1945746225.us-west-2.elb.amazonaws.com	Health 68
 PHP App Server Settings Recipes Network EBS Volumes Security CloudWatch Logs Tags Delete	Instances 68
 Node.js App Server Settings Recipes Network EBS Volumes Security CloudWatch Logs Tags Delete	Instances 1
 MySQL Settings Recipes Network EBS Volumes Security CloudWatch Logs Tags Delete	Instances 6

レイヤーレベルでタグを変更または追加するときに、親スタックレベルで追加されたタグはレイヤーとそのリソースによって継承されることに注意してください。継承されたタグの値を変更することはできませんが、キー名の変更や継承されたタグを削除することはできません。スタック設定で、キー名を変更または親スタックから継承されたタグを削除します。次のスクリーンショットでは、スタックレベルから継承されたタグの例を示しています。継承されたタグは灰色で表示されます。

Layer MyLayer

General Settings Recipes Network EBS Volumes Security CloudWatch Logs **Tags**

Tags ⓘ

Key (127 characters maximum)	Value (255 characters maximum)	
Organization	Mobile	✖
Staging	Demo	✖
Add key	Add value (optional)	

You cannot remove a tag that is inherited from the parent stack.

スタックへのタグの追加については、「[新しいスタックを作成する](#)」を参照してください。レイヤーへのタグの追加については、「[OpsWorks レイヤーの設定の編集](#)」を参照してください。

を使用したタグの管理 AWS CLI

AWS CLI コマンドを使用して、スタックおよびレイヤーレベルでタグを追加および削除することもできます。のダウンロードとインストールの詳細については AWS CLI、「[コマンドラインインターフェイスのインストール AWS](#)」を参照してください。タグ付けしたいスタックがデフォルトのリージョンにない場合は、コマンドに `--region` パラメータを忘れずに追加してください。レイヤー ARNs は現在、に表示されません AWS Management Console。レイヤーの ARN を取得するには、[describe-layers](#) コマンドを実行します。

を使用してタグを追加するには AWS CLI

- AWS CLI コマンドプロンプトで、次のコマンドを入力して `stack_or_layer_ARN` を置き換え、キーと値のペアタグを指定し、Enter キーを押します。二重引用符はバックスラッシュでエスケープされます。

```
aws opsworks tag-resource --resource-arn stack_or_layer_ARN --tags "{\"key\": \"value\", \"key\": \"value\"}"
```

次に例を示します。

```
aws opsworks tag-resource --resource-arn arn:aws:opsworks:us-east-2:800000000003:stack/500b99c0-ec00-4cgg-8a0d-1000000jjd1b --tags "{\"Stage\": \"Production\", \"Organization\": \"Mobile\"}"
```

を使用してタグを削除するには AWS CLI

- AWS CLI コマンドプロンプトで、次のように入力し、Enter キーを押します。

```
aws opsworks untag-resource --resource-arn stack_or_layer_ARN --tag-keys "[\"key\", \"key\"]"
```

タグを削除するには、削除したいタグのキーのみを指定します。次に例を示します。

```
aws opsworks untag-resource --resource-arn arn:aws:opsworks:us-east-2:800000000003:stack/500b99c0-ec00-4cgg-8a0d-1000000jjd1b --tag-keys "[\"Stage\", \"Organization\"]"
```

Note

レイヤーから継承されたタグ (親スタックレベルで追加されたタグ) を削除することはできません。代わりに、スタックから継承されたタグを削除します。

タグの制限

タグを作成するときは、次の制限事項に留意してください。

- AWS OpsWorks スタックは、親レベルから継承されたユーザー定義タグを含め、スタックおよびレイヤーレベルでのユーザー定義タグの数を 40 に制限します。これにより、が付加されたデフォルトタグ `opsworks:`、および他の AWS プロセスによって設定されたタグに 10 個のロットが残ります。リソースでは、によって作成されたユーザー定義タグとデフォルトタグの両方を含め、最大 50 個のタグを使用できます AWS。
- タグキーの先頭を、`aws:`、`opsworks:` または `rds:` にすることはできません。Name は AWS OpsWorks スタックによって予約されているため、`name` または `name` をタグキーNameとして使用しないでください。
- キーは最大 127 文字で、Unicode 文字、区切り文字、数字、または `+ - = . _ : /` の特殊文字のみを含めることができます。
- 値は最大 255 文字で、Unicode 文字、数字、区切り文字、または `+ - = . _ : /` の特殊文字のみを含めることができます。

モニタリング

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

スタックは次の方法で監視できます。

- AWS OpsWorks スタックは Amazon を使用して CloudWatch、スタック内の各インスタンスの詳細なモニタリングを 13 のカスタムメトリクスに提供します。
- AWS OpsWorks スタックはと統合され AWS CloudTrail、すべての AWS OpsWorks スタック API コールをログに記録し、そのデータを Amazon S3 バケットに保存します。
- Amazon CloudWatch Logs を使用して、スタックのシステム、アプリケーション、カスタムログをモニタリングできます。

トピック

- [Amazon を使用した スタックのモニタリング CloudWatch](#)
- [を使用した AWS OpsWorks スタックAPI呼び出しのログ記録 AWS CloudTrail](#)
- [AWS OpsWorks スタックでの Amazon CloudWatch Logs の使用](#)
- [Amazon CloudWatch Events を使用したスタックのモニタリング](#)

Amazon を使用した スタックのモニタリング CloudWatch

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは Amazon CloudWatch (CloudWatch) を使用してスタックをモニタリングします。

- Linux スタックの場合、AWS OpsWorks Stacks は 13 個のカスタムメトリクスをサポートしており、スタック内の各インスタンスの詳細モニタリングを提供し、モニタリングページでデータを要約します。
- Windows スタックの場合、コンソール を使用してインスタンスの標準 Amazon EC2 [CloudWatch](#) メトリクスをモニタリングできます。

[Monitoring] ページには、Windows メトリクスは表示されません。

モニタリングページには、スタック全体、レイヤー、またはインスタンスのメトリクスが表示されます。AWS OpsWorks スタックメトリクスは Amazon EC2 メトリクスとは異なります。CloudWatch コンソールから追加のメトリクスを有効にすることもできますが、通常は追加料金が必要です。次のように、基盤となるデータを CloudWatch コンソールで表示することもできます。

で OpsWorks カスタムメトリクスを表示するには CloudWatch

1. <https://console.aws.amazon.com/cloudwatch/> で CloudWatch コンソールを開きます。
2. ナビゲーションバーで、スタックのリージョンを選択します。
3. ナビゲーションペインで [メトリクス]を選択します。
4. OpsWorks メトリクス で、インスタンスメトリクス、レイヤーメトリクス、または スタックメトリクス を選択します。

CloudWatch Metrics by Category

Your CloudWatch metric summary has loaded. Total metrics: **362**

EBS Metrics: 16

Per-Volume Metrics: 16

EC2 Metrics: 61

Per-Instance Metrics: 61

ElastiCache Metrics: 51

: 17

CacheClusterId: 17

Cache Node Metrics: 17

OpsWorks Metrics: 225

Instance Metrics: 105

Layer Metrics: 75

Stack Metrics: 45

Note

AWS OpsWorks スタックは、各インスタンス (インスタンスエージェント) でプロセスを実行してメトリクスを収集します。はハイパーバイザーを使用してメトリクスを CloudWatch 収集する方法が異なるため、CloudWatch コンソールの値は、スタックコンソールのモニタリングページの AWS OpsWorks 対応する値と若干異なる場合があります。

CloudWatch コンソールを使用してアラームを設定することもできます。アラームの作成方法の詳細については、[「Amazon CloudWatch アラームの作成」](#)を参照してください。CloudWatch カスタムメトリクスのリストについては、[「AWS OpsWorks メトリクスとディメンション」](#)を参照してください。詳細については、[「Amazon CloudWatch」](#)を参照してください。

トピック

- [AWS OpsWorks スタックメトリクス](#)
- [AWS OpsWorks スタックメトリクスのディメンション](#)
- [スタックメトリクス](#)
- [レイヤーメトリクス](#)
- [インスタンスメトリクス](#)

AWS OpsWorks スタックメトリクス

AWS OpsWorks スタックは、次のメトリクスを 5 分 CloudWatch ごとに送信します。

CPU メトリクス

メトリクス	説明
cpu_idle	<p>CPU がアイドル状態の時間の割合。</p> <p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>
cpu_nice	<p>CPU が正の値でプロセスを処理している時間の割合 nice 値。スケジューリングの優先順位が低くなります。この測定値の詳細については、[nice (Unix)] (い いな (Unix)) を参照してください。</p> <p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>
cpu_steal	<p>AWS は、増加するインスタンスにハイパーバイザー CPU リソースを割り当てると、仮想化の負荷が上昇し、ハイパーバイザーがインスタンス上で求められた作業を実行できる頻度に影響を与える可能性があります。cpu_steal ハイパーバイザーが物理 CPU リソースを割り当ててのをインスタンスが待機している時間の割合を測定します。</p>

メトリクス	説明
	<p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>
cpu_system	<p>CPU がシステムオペレーションを処理している時間の割合。</p> <p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>
cpu_user	<p>CPU がユーザー操作を処理している時間の割合。</p> <p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>

メトリクス	説明
cpu_waitio	<p>CPU が入力/出力のオペレーションを待機している時間の割合。</p> <p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>

メモリメトリクス

メトリクス	説明
memory_buffers	<p>バッファリングされたメモリの量。</p> <p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>
memory_cached	<p>キャッシュされたメモリの量。</p> <p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>

メトリクス	説明
memory_free	<p>空きメモリの量。</p> <p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>
memory_swap	<p>スワップ領域の量。</p> <p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>
memory_total	<p>メモリの合計量。</p> <p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>

メトリクス	説明
memory_used	<p>使用中のメモリの量。</p> <p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>

負荷メトリクス

メトリクス	説明
load_1	<p>負荷は 1 分間のウィンドウで平均化されます。</p> <p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>
load_5	<p>負荷は 5 分間のウィンドウで平均化されます。</p> <p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>
load_15	<p>負荷は 15 分間のウィンドウで平均化されます。</p>

メトリクス	説明
	<p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>

プロセスメトリクス

メトリクス	説明
procs	<p>アクティブなプロセスの数。</p> <p>有効なディメンション: メトリクスを表示している個々のリソースの IDs: StackId、LayerId、または InstanceId。</p> <p>有効な統計: Average、Minimum、Maximum、Sum または Data Samples。</p> <p>単位: なし</p>

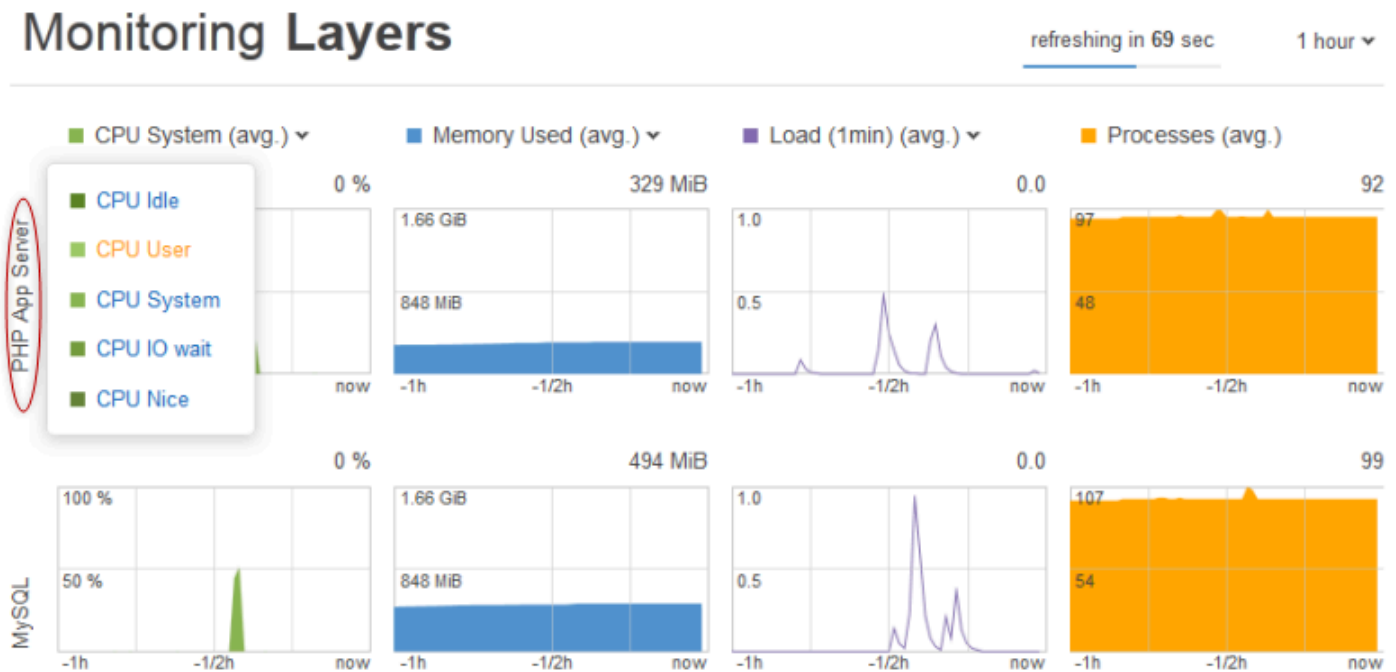
AWS OpsWorks スタックメトリクスのディメンション

AWS OpsWorks スタックメトリクスは AWS OpsWorks スタック名前空間を使用し、次のディメンションのメトリクスを提供します。

ディメンション	説明
StackId	スタックの平均値。
LayerId	レイヤーの平均値。
InstanceId	インスタンスの平均値。

スタックメトリクス

スタック全体のメトリクスの概要を表示するには、AWS OpsWorks スタックダッシュボードでスタックを選択し、ナビゲーションペインのモニタリングをクリックします。以下に示したのは、PHP および DB レイヤーを含んだスタックの例です。



スタックビューには、指定した期間 (1 時間、8 時間、24 時間、1 週間、2 週間) を対象とする 4 種類のメトリクスのグラフがレイヤーごとに表示されます。次の点に注意してください。

- AWS OpsWorks スタックは定期的にグラフを更新します。右上のカウントダウンタイマーは、次の更新までの残り時間を示します。
- レイヤーに複数のインスタンスが存在する場合、グラフには、そのレイヤーの平均値が表示されます。
- 右上のリストをクリックし、目的の値を選択することで期間を指定できます。

表示するメトリクスは、各メトリクスタイプのグラフの上にあるリストで選択できます。

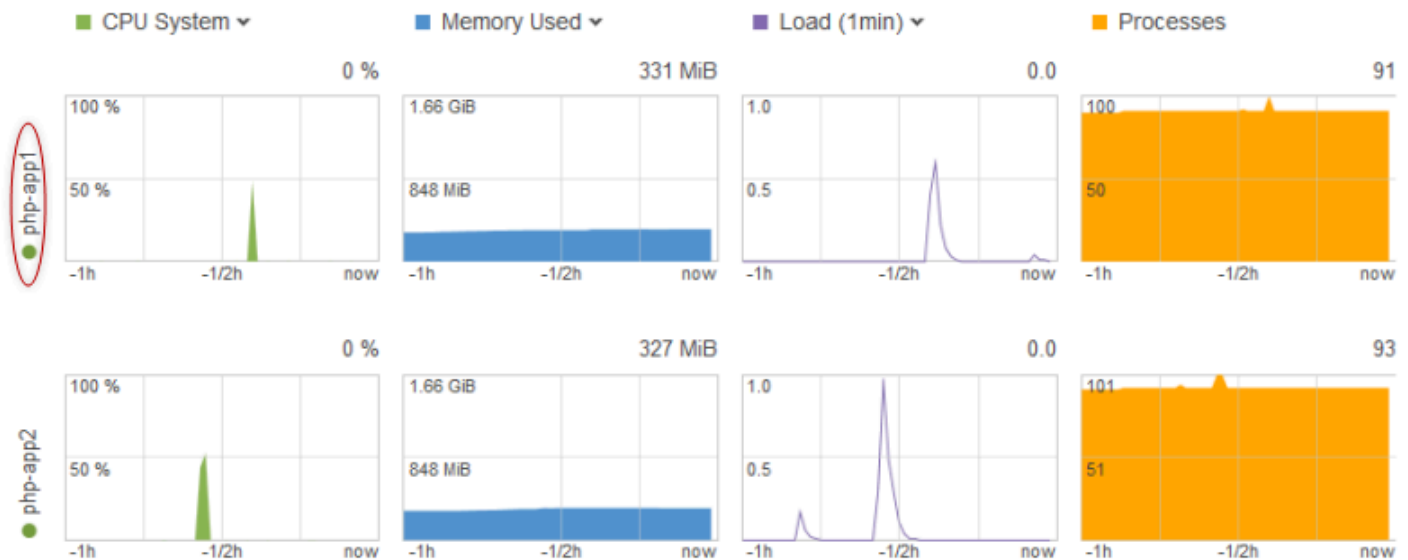
レイヤーメトリクス

特定のレイヤーのメトリクスを表示するには、[Monitoring Layers] (レイヤーのモニタリング) ビューでレイヤー名をクリックします。次に示したのは、2 つのインスタンスが存在する PHP レイヤーのメトリクスの例です。

Layer PHP App Server

refreshing in 111 sec

1 hour ▾



メトリクスのタイプは、スタックのメトリクスと同じです。各メトリクスタイプのグラフの上にあるリストを使用して、表示するメトリクスを選択できます。

Note

レイヤーの詳細ページに移動し、右上の [Monitoring] をクリックして、レイヤーのメトリクスを表示することもできます。

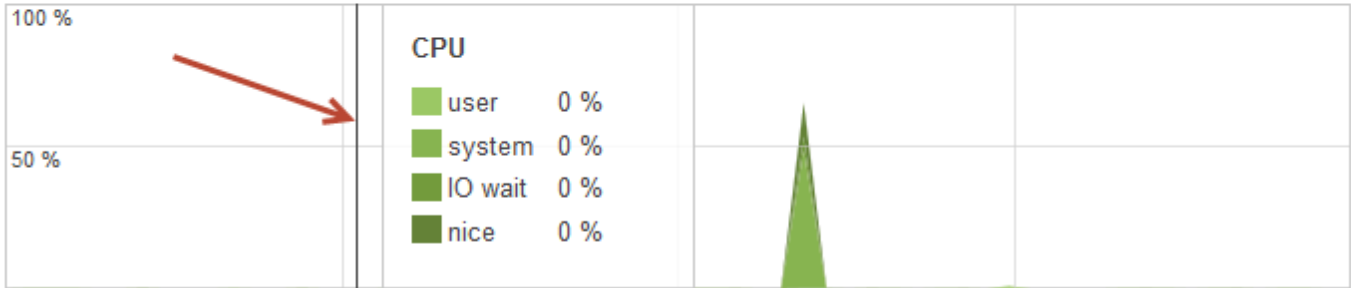
インスタンスメトリクス

特定のインスタンスのメトリクスを表示するには、レイヤーのモニタリングビューでインスタンス名をクリックします。次の例は、PHP レイヤーの php-app1 インスタンスに関するメトリクスを示しています。

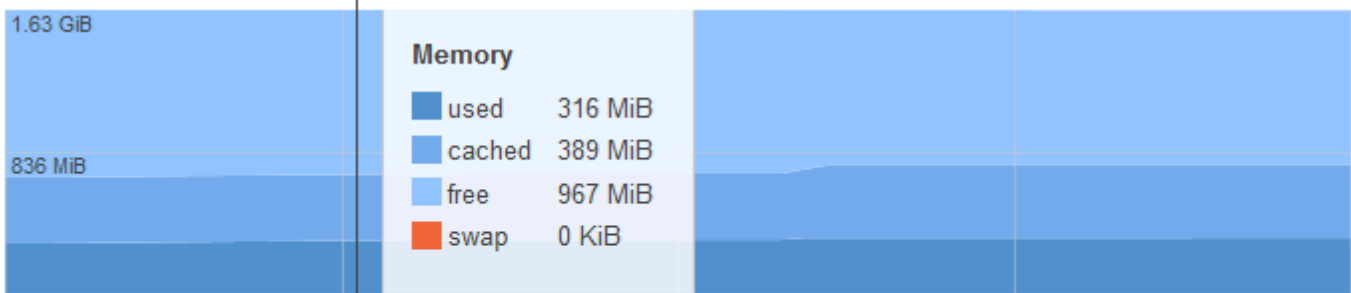
Instance php-app1 ●

refreshing in

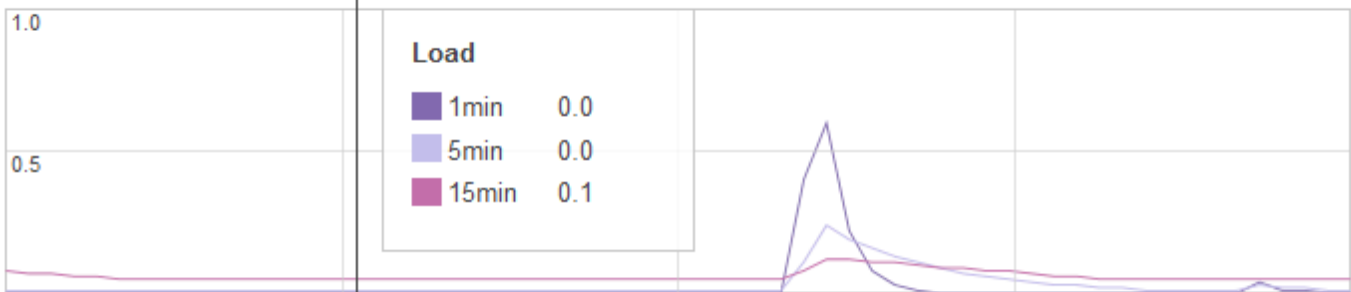
2013-07-16 18:09 UTC



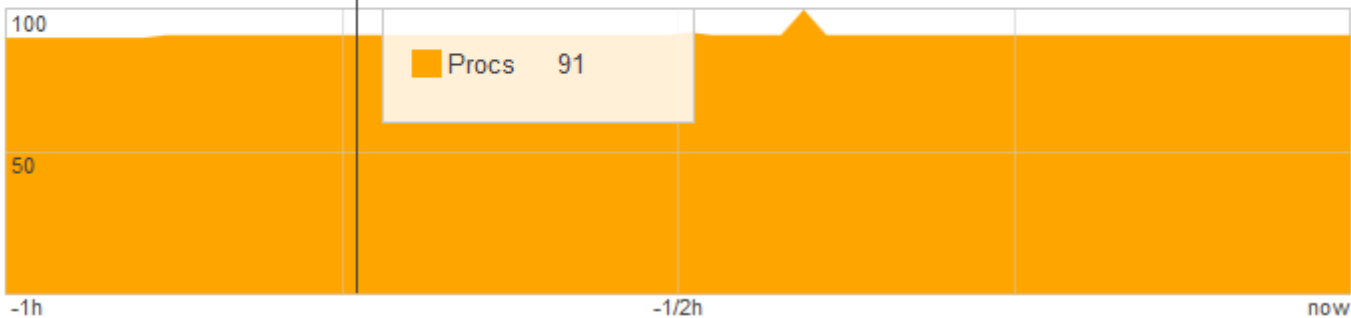
2013-07-16 18:09 UTC



2013-07-16 18:09 UTC



2013-07-16 18:09 UTC



これらのグラフは、利用できるすべてのメトリクスをメトリクスタイプごとにまとめたものです。特定時点の正確な値を表示するには、マウスを使用してスライダ (前の図に赤色の矢印で示した箇所) を適切な位置に移動します。

Note

インスタンスの詳細ページに移動し、右上の [Monitoring] を選択して、インスタンスのメトリクスを表示することもできます。

を使用した AWS OpsWorks スタックAPI呼び出しのログ記録 AWS CloudTrail

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは AWS CloudTrail、IAMアイデンティティによって実行されたアクションを記録するサービス、または AWS OpsWorks Stacks の AWS サービスと統合されています。は、スタックコンソールからのAPI呼び出しや AWS OpsWorks スタックへのコード呼び出しを含む、AWS OpsWorks スタックのすべての呼び出しをイベントとして CloudTrail キャプチャ AWS OpsWorks します APIs。証跡を作成する場合は、スタックの CloudTrail イベントなど、Amazon S3 AWS OpsWorks バケットへのイベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールのイベント履歴で最新のイベントを表示できます。によって収集された情報を使用して CloudTrail、AWS OpsWorks スタックに対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

詳細については CloudTrail、[「AWS CloudTrail ユーザーガイド」](#) を参照してください。

AWS OpsWorks のスタック情報 CloudTrail

CloudTrail AWS アカウントを作成すると、がアカウントで有効になります。AWS OpsWorks スタックでアクティビティが発生すると、そのアクティビティは CloudTrail イベント履歴の他の AWS

サービスイベントとともにイベントに記録されます。AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、[「イベント履歴を使用した CloudTrail イベントの表示」](#)を参照してください。

AWS OpsWorks スタックのイベントなど、AWS アカウント内のイベントの継続的な記録については、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで追跡を作成するときに、追跡がすべてのリージョンに適用されます。証跡は、AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析してそれに基づいて行動するように、他の AWS サービスを設定できます。詳細については、以下を参照してください。

- [証跡を作成するための概要](#)
- [CloudTrail サポートされているサービスと統合](#)
- [の Amazon SNS 通知の設定 CloudTrail](#)
- [複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからの CloudTrail ログファイルの受信](#)

すべての AWS OpsWorks スタックアクションは、[AWS OpsWorks スタックAPIリファレンス](#)によってログに記録され、[「AWS OpsWorks スタックAPIリファレンス」](#)に記載されています。例えば、[CreateLayer](#)、および [StartInstance](#) アクションを呼び出すと [DescribeInstances](#)、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。ID 情報は次の判断に役立ちます。

- リクエストが、ルートと IAM ユーザー認証情報のどちらを使用して送信されたか。
- リクエストが、ロールとフェデレーションユーザーの一時的なセキュリティ認証情報のどちらを使用して送信されたか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、[CloudTrail userIdentity 「要素」](#)を参照してください。

AWS OpsWorks スタックログファイルエントリについて

証跡は、指定した Amazon S3 バケットにイベントをログファイルとして配信できるようにする設定です。CloudTrail ログファイルには 1 つ以上のログエントリが含まれます。イベントは任意のソー

スからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。CloudTrail ログファイルはパブリックAPIコールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次の例は、CreateLayerアクションを示す CloudTrail ログエントリを示しています。

```
{
  "Records": [
    {
      "awsRegion": "us-west-2",
      "eventID": "342cd1ec-8214-4a0f-a68f-8e6352feb5af",
      "eventName": "CreateLayer",
      "eventSource": "opsworks.amazonaws.com",
      "eventTime": "2014-05-28T16:05:29Z",
      "eventVersion": "1.01",
      "requestID": "e3952a2b-e681-11e3-aa71-81092480ee2e",
      "requestParameters": {
        "attributes": {},
        "customRecipes": {},
        "name": "2014-05-28 16:05:29 +0000 a073",
        "shortname": "customcf4571d5c0d6",
        "stackId": "a263312e-f937-4949-a91f-f32b6b641b2c",
        "type": "custom"
      },
      "responseElements": null,
      "sourceIPAddress": "198.51.100.0",
      "userAgent": "aws-sdk-ruby/2.0.0 ruby/2.1 x86_64-linux",
      "userIdentity": {
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "accountId": "111122223333",
        "arn": "arn:aws:iam::111122223333:user/A-User-Name",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "type": "IAMUser",
        "userName": "A-User-Name"
      }
    },
    {
      "awsRegion": "us-west-2",
      "eventID": "a860d8f8-c1eb-449b-8f55-eafc373b49a4",
      "eventName": "DescribeInstances",
      "eventSource": "opsworks.amazonaws.com",
      "eventTime": "2014-05-28T16:05:31Z",
```

```
"eventVersion": "1.01",
"requestID": "e4691bfd-e681-11e3-aa71-81092480ee2e",
"requestParameters": {
  "instanceIds": [
    "218289c4-0492-473d-a990-3fbe1efa25f6"
  ]
},
"responseElements": null,
"sourceIPAddress": "198.51.100.0",
"userAgent": "aws-sdk-ruby/2.0.0 ruby/2.1x86_64-linux",
"userIdentity": {
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "accountId": "111122223333",
  "arn": "arn:aws:iam::111122223333:user/A-User-Name",
  "principalId": "AKIAI44QH8DHBEXAMPLE",
  "type": "IAMUser",
  "userName": "A-User-Name"
}
}
]
```

AWS OpsWorks スタックでの Amazon CloudWatch Logs の使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

複数のインスタンスのログをモニタリングするプロセスを簡素化するために、AWS OpsWorks スタックは Amazon CloudWatch Logs をサポートします。AWS OpsWorks Stacks のレイヤーレベルで CloudWatch Logs を有効にします。CloudWatch Logs 統合は Chef 11.10 および Chef 12 Linux ベースのスタックと連携します。Logs を有効にすると追加料金が発生するため、開始する前に Amazon CloudWatch の料金を確認してください。 [CloudWatch](#)

CloudWatch Logs は、選択したログをモニタリングして、ユーザーが指定したパターンの発生を確認します。例えば、NullReferenceException のようなリテラルタームが発生したログをモニタ

リングしたり、発生数をカウントしたりできます。AWS OpsWorks スタックで CloudWatch ログを有効にすると、AWS OpsWorks スタックエージェントはログを CloudWatch ログに送信します。CloudWatch ログの詳細については、[CloudWatch 「ログの開始方法」](#)を参照してください。

前提条件

CloudWatch Logs を有効にする前に、インスタンスは Chef 11.10 スタックの Stacks エージェントのバージョン 3444 以降 AWS OpsWorks、Chef 12 スタックの 4023 以降を実行している必要があります。CloudWatch ログを使用してモニタリングするインスタンスには、互換性のあるインスタンスプロファイルも使用する必要があります。

カスタムインスタンスプロファイル (スタックの作成時に AWS OpsWorks スタックが提供しなかったプロファイル) を使用している場合、AWS OpsWorks スタックはインスタンスプロファイルを自動的にアップグレードできません。IAM を使用してAWSOpsWorksCloudWatchLogs、ポリシーをプロファイルに手動でアタッチする必要があります。IAM ポリシーの詳細については IAM ユーザーガイドの「[IAMポリシーの管理](#)」を参照してください。

エージェントバージョンまたはインスタンスプロファイルをアップグレードする必要がある場合、レイヤーページの CloudWatch ログタブを開くと、AWS OpsWorks スタックに次のスクリーンショットのようなリマインダーが表示されます。

CloudWatch Logs integration ⓘ

Upgrade Required

This feature requires instances in this layer to have a compatible instance profile and OpsWorks agent version. In order to enable this feature please ensure that:

- All instances in this stack are upgraded to OpsWorks agent version [4023](#).
- The [AWSOpsWorksCloudWatchLogs](#) managed policy is attached to [aws-opsworks-ec2-role](#) instance profile.

Cancel **Save**

レイヤーのすべてのインスタンスでエージェントを更新するには時間がかかります。エージェントのアップグレードが完了する前にレイヤーで CloudWatch ログを有効にしようとすると、次のようなメッセージが表示されます。

OpsWorks Agent Upgrade in Progress

1 instances in this layer are upgrading their OpsWorks agent to a version compatible with CloudWatch Logs. If this upgrade has not completed within 15 minutes, visit [this page](#) for details on how to resolve the issue.

CloudWatch ログの有効化

1. 必要なエージェントとインスタンスプロファイルのアップグレードが完了したら、CloudWatch Logs タブのスライダーコントロールを On に設定することで CloudWatch、Logs を有効にできます。

Layer PHP App Server

General Settings

Recipes

Network

EBS Volumes

Security

CloudWatch Logs

CloudWatch Logs integration ⓘ

On

2. コマンドログをストリーミングするには、[Stream command logs] を [On] に設定します。これにより、レイヤーのインスタンスで Chef アクティビティとユーザーが開始したコマンドのログが CloudWatch Logs に送信されます。

これらのログに含まれるデータは、ログ URL のターゲットを開く

と、[DescribeCommands](#) オペレーションの結果に表示されるデータと密接に一致しま

す。setup、configure、deploy、undeploy、start、stop、およびレシピの実行コマンドに関するデータが含まれています。

3. レイヤーのインスタンスのカスタムロケーションに保存されているアクティビティのログ (/var/log/apache/myapp/mylog* など) をストリーミングするには、[Stream custom logs] 文字列ボックスにカスタムロケーションを入力し、[Add] (+) を選択します。
4. [保存] を選択します。数分以内に、AWS OpsWorks スタックログストリームが CloudWatch ログコンソールに表示されるはずです。

Layer PHP App Server

Edit

Delete

Instances

Monitoring

General Settings

Recipes

Network

EBS Volumes

Security

CloudWatch Logs

CloudWatch Logs integration ⓘ

Opsworks Chef Logs yes

Custom Log Streams

CloudWatch ログをオフにする

CloudWatch ログを無効にするには、レイヤー設定を編集します。

1. レイヤーのプロパティページで、[編集] を選択します。

Layer PHP App Server

Edit

Delete

Instances

Monitoring

General Settings

Recipes

Network

EBS Volumes

Security

CloudWatch Logs

CloudWatch Logs integration ⓘ

Opsworks Chef Logs yes

Custom Log Streams

2. 編集ページで、CloudWatch ログタブを選択します。
3. CloudWatch ログ エリアで、ストリームコマンドログ をオフにします。カスタムログで X を選択して、ログストリームから削除します (該当する場合)。
4. [Save] を選択します。

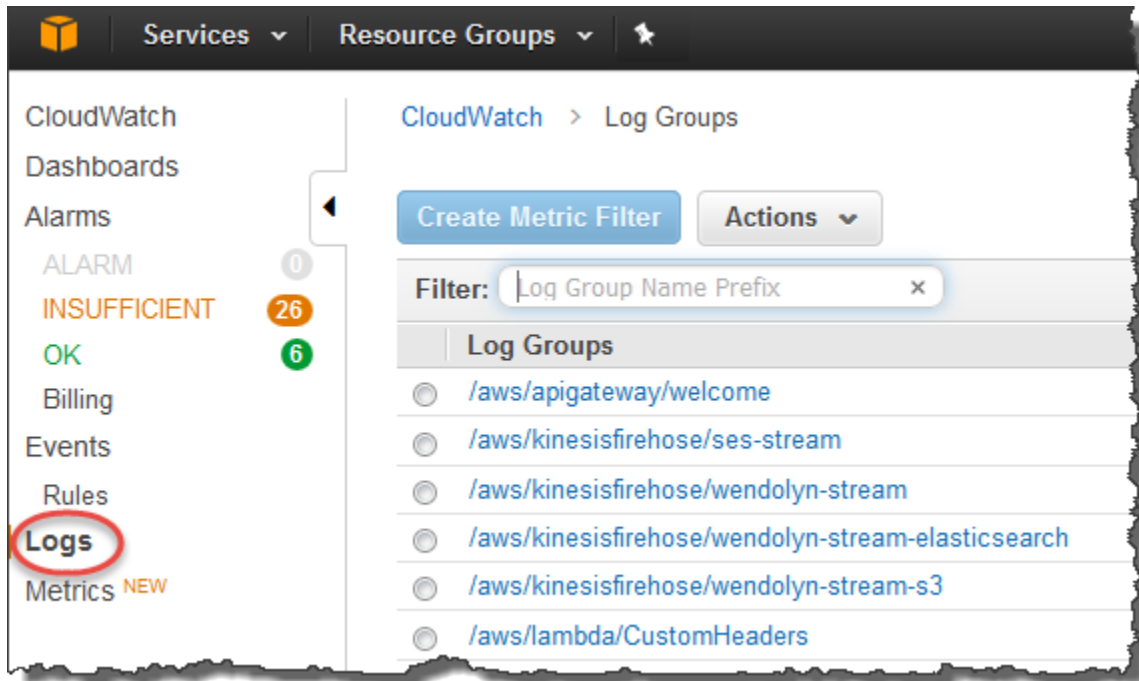
ログからのストリーミング CloudWatch ログの削除

AWS OpsWorks スタックからの CloudWatch ログストリーミングをオフにしても、既存のログは CloudWatch 引き続きログ管理コンソールで使用できます。ログを Amazon S3 にエクスポートまたは削除しない限り、保存されたログに対しての料金が引き続き発生します。S3 へのログのエクスポートの詳細については [\[Exporting Log Data to Amazon S3\]](#) (Amazon S3 へのログデータのエクスポート) を参照してください。

ログストリームとロググループは、CloudWatch ログ管理コンソールで、または [delete-log-stream](#) および [delete-log-group](#) AWS CLI コマンドを実行して削除できます。ログ保持期間の変更の詳細については、「ログの[ログデータ保持期間の変更 CloudWatch](#)」を参照してください。

ログでの CloudWatch ログの管理

ストリーミングしているログは、CloudWatch ログコンソールで管理されます。



AWS OpsWorks は、デフォルトのロググループとログストリームを自動的に作成します。AWS OpsWorks スタックデータのロググループには、以下のパターンに一致する名前があります。

stack_name/layer_name/chef_log_name

カスタムログには次のパターンに一致する名前があります。

/stack_name/layer_short_name/file_path_name パス名は人が読んで理解しやすいように、アスタリスク (*) などの特殊文字が削除されています。

CloudWatch Logs でログを見つけたら、[ログをグループに整理し、メトリクスフィルターを作成してログを検索およびフィルタリングし、カスタムアラームを作成できます。](#)

CloudWatch ログを使用するように Chef 12.2 Windows レイヤーを設定する

CloudWatch ログの自動統合は、Windows ベースのインスタンスではサポートされていません。CloudWatch ログタブは、Chef 12.2 スタックのレイヤーでは使用できません。Windows ベース

のインスタンスの CloudWatch ログへのストリーミングを手動で有効にするには、次の手順を実行します。

- Logs エージェントが適切なアクセス許可を持つように、Windows CloudWatch ベースのインスタンスのインスタンスプロファイルを更新します。AWSOpsWorksCloudWatchLogs ポリシーステートメントには、必要なアクセス許可が表示されます。

通常、このタスクを行うのは 1 回のみです。これで、更新したインスタンスプロファイルはレイヤー内のすべての Windows インスタンスで使用することができます。

- 各インスタンスで以下の JSON 設定ファイルを編集します。このファイルには、どのログをモニタリングするかなどログストリームの設定が含まれています。

```
%PROGRAMFILES%\Amazon\Ec2ConfigService\Settings  
\AWS.EC2.Windows.CloudWatch.json
```

必要なタスクを処理するカスタムレシピを作成し、Chef 12.2 レイヤーの Setup イベントに割り当てることで、前述の 2 つのタスクを自動化できます。これらのレイヤーで新しいインスタンスを起動するたびに、インスタンスの起動が完了すると、AWS OpsWorks スタックによってレシピが自動的に実行され、CloudWatch ログが有効になります。

Windows ベースのインスタンスで CloudWatch ログを無効にするには、プロセスを逆にします。EC2 サービスプロパティダイアログボックスの CloudWatch ログ統合の有効化チェックボックスをオフにし、AWS.EC2.Windows.CloudWatch.json ファイルからログストリーム設定を削除し、Chef 12.2 レイヤーの新しいインスタンスに CloudWatch ログ許可を自動的に割り当てている Chef レシピの実行を停止します。

Amazon CloudWatch Events を使用したスタックのモニタリング

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックリソースの変更を警告するように Amazon CloudWatch Events のルールを設定し、CloudWatch イベントの内容に基づいてアクションを実行するように Events に指示できます。CloudWatch イベントの開始方法とルールの設定方法の詳細については、「[CloudWatch イベントユーザーガイド](#)」の CloudWatch 「イベントの開始方法」を参照してください。

イベントでは、次の AWS OpsWorks スタックイベントタイプがサポートされています
CloudWatch。

インスタンスの状態変更

AWS OpsWorks スタックインスタンスの状態の変化を示します。

コマンドの状態変更

AWS OpsWorks Stacks コマンドの状態に変更が発生したことを示します。

デプロイの状態変更

AWS OpsWorks スタックデプロイの状態に変更が発生したことを示します。

アラート

AWS OpsWorks スタックサービスエラーが発生したことを示します。

イベントでサポートされている AWS OpsWorks スタックイベントタイプの詳細については、CloudWatch 「イベントユーザーガイド」の [AWS OpsWorks 「スタックイベント」](#) を参照してください。CloudWatch

セキュリティと権限

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

各ユーザーは、アカウントの AWS リソースにアクセスするための適切な AWS 認証情報を持っている必要があります。ユーザーに認証情報を提供する推奨方法は、[AWS Identity and Access](#)

[Management](#) (IAM) を使用することです。AWS OpsWorks スタックは IAM と統合され、以下を制御できます。

- 個々のユーザーが AWS OpsWorks スタックを操作する方法。

たとえば、あるユーザーに対してはスタックへのアプリケーションのデプロイはできるが、スタック自体の変更はできないようにし、他のユーザーに対しては特定のスタックに限定してフルアクセスができるようにします。

- AWS OpsWorks スタックがユーザーに代わって Amazon EC2 インスタンスや Amazon S3 バケットなどのスタックリソースにアクセスする方法。

AWS OpsWorks スタックは、これらのタスクのアクセス許可を付与するサービスロールを提供します。

- AWS OpsWorks スタックによって制御される Amazon EC2 インスタンスで実行されるアプリケーションが、Amazon S3 バケットに保存されているデータなどの他の AWS リソースにアクセスする方法。

インスタンスプロファイルをレイヤーのインスタンスに割り当てて、それらのインスタンスで実行されているアプリケーションに他の AWS リソースへのアクセス許可を付与できます。

- ユーザーベースの SSH キーを管理し、SSH または RDP によりインスタンスに接続する方法

スタックごとに、管理ユーザーは、各ユーザーに個人 SSH キーを割り当てたり、ユーザーに自身のキーの指定を許可したりできます。また、各ユーザーのスタックのインスタンスに対する SSH または RDP アクセスと、sudo または管理者特権を許可できます。

セキュリティのその他の側面には、次のようなものがあります。

- 最新のセキュリティパッチによるインスタンスのオペレーティングシステムの更新を管理する方法。

詳細については、「[セキュリティ更新の管理](#)」を参照してください。

- インスタンスとの間で送受信されるネットワークトラフィックを [Amazon EC2 security groups](#) (Amazon EC2 セキュリティグループ) で制御するように設定する方法。

AWS OpsWorks スタックのデフォルトのセキュリティグループの代わりにカスタムセキュリティグループを指定する方法。詳細については、「[セキュリティグループの使用](#)」を参照してください。

トピック

- [AWS OpsWorks スタックユーザーアクセス許可の管理](#)
- [AWS OpsWorks スタックがユーザーに代わって動作することを許可する](#)
- [AWS OpsWorks スタックでのサービス間の混乱した代理の防止](#)
- [EC2 インスタンスで実行されるアプリケーションのアクセス許可の指定](#)
- [SSH アクセスの管理](#)
- [Linux セキュリティ更新の管理](#)
- [セキュリティグループの使用](#)

AWS OpsWorks スタックユーザーアクセス許可の管理

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

ベストプラクティスとして、AWS OpsWorks スタックのユーザーを、指定された一連のアクションまたは一連のスタックリソースに制限します。AWS OpsWorks スタックユーザーアクセス許可は、AWS OpsWorks スタックアクセス許可ページを使用する方法と、適切な IAM ポリシーを適用する方法の 2 つの方法で制御できます。

アクセス OpsWorks 許可ページ、または同等の CLI または API アクションでは、複数のアクセス許可レベルのいずれかを各ユーザーに割り当てて、マルチユーザー環境でユーザーアクセス許可をスタックごとに制御できます。それぞれのレベルで付与される権限は、特定のスタックリソースに対する標準的なアクション一式に対応します。[Permissions] ページを使用して制御できる権限は次のとおりです。

- 各スタックにアクセスできるユーザー。
- 個々のユーザーが各スタックに対して実行できるアクション。

たとえば、一部のユーザーにはスタックの閲覧のみを許可する一方で、他のユーザーにはアプリケーションのデプロイ、インスタンスの追加などを許可することができます。

- 各スタックを管理できるユーザー。

指定したユーザー (複数可) に各スタックの管理を委任できます。

- 各スタックの Amazon EC2 インスタンスにおけるユーザーレベルの SSH アクセスと sudo 特権 (Linux) または RDP アクセスと管理者特権 (Windows) を持つ人。

個々のユーザーについて、これらの権限をいつでも許可したり取り消したりすることができます。

Important

SSH/RDP アクセスを拒否しても、必ずしもユーザーがインスタンスにログインできなくなるわけではありません。インスタンスに Amazon EC2 キーペアを指定した場合、対応するプライベートキーを持つユーザーは誰でもログインするかキーを使用して Windows 管理者パスワードを取得できます。詳細については、「[SSH アクセスの管理](#)」を参照してください。

[IAM コンソール](#)、CLI または API から、AWS OpsWorks スタックの各種リソースやアクションに対するアクセス許可を明示的に付与するポリシーをユーザーに追加することもできます。

- 権限は、IAM ポリシーを使用した方が、権限レベルで行うよりも柔軟に指定できます。
- [IAM ID \(ユーザー、ユーザーグループ、ロール\)](#) を設定して、ユーザーやユーザーグループなどの IAM ID にアクセス権限を付与したり、フェデレーションユーザーに関連付けられる [ロール](#) を定義したりできます。
- IAM ポリシーは、特定のキー AWS OpsWorks スタックアクションのアクセス許可を付与する唯一の方法です。

例えば、`opsworks:CreateStack` と `opsworks:CloneStack` (それぞれスタックの作成とクローン化に使用) へのアクセス権限を付与するには、IAM を使用する必要があります。

コンソールでフェデレーティッドユーザーをインポートすることは明示的には不可能ですが、フェデレーティッドユーザーは、AWS OpsWorks スタックコンソールの右上にある「マイ設定」を選択し、右上にある「ユーザー」を選択することで、暗黙的にユーザープロフィールを作成できます。ユーザー ページで、API または CLI を使用して作成された、またはコンソールを通じて暗黙的に作成されたアカウントのフェデレーティッドユーザーは、非フェデレーティッド IAM ユーザーと同じ様にアカウントを管理できます。

この2つのアプローチは相互排他的なものではなく、両者を組み合わせることで利便性が高まることもあります。両者を組み合わせて使用した場合、両方の権限の集合が AWS OpsWorks スタックによって評価されます。たとえば、インスタンスの追加と削除は許可するが、レイヤーの追加と削除をユーザーに許可するのは望ましくないとします。AWS OpsWorks スタックのアクセス許可レベルは、その特定のアクセス許可セットを付与しません。ただし、許可ページを使用して、管理権限レベルをユーザーに付与し、ほとんどのスタック操作を許可したうえで、レイヤーの追加と削除の権限を拒否する IAM ポリシーを適用することは可能です。詳細については、[\[のポリシーを使用して AWS リソースへのアクセスを制御する\]](#)を参照してください。

ユーザーの権限を管理するための標準的なモデルを以下に示します。いずれのケースも、ここではお客様が管理ユーザーであるという前提で記述しています。

1. [IAM コンソール](#)を使用して、1人以上の管理ユーザーに AWSOpsWorks_FullAccess ポリシーを適用します。
2. 非管理ユーザーそれぞれに、AWS OpsWorks スタックの権限が付与されていないポリシーでユーザーを作成します。

ユーザーが AWS OpsWorks スタックへのアクセスのみを必要とする場合は、ポリシーをまったく適用する必要がなくなる場合があります。代わりに、AWS OpsWorks スタックのアクセス許可ページでアクセス許可を管理できます。

3. AWS OpsWorks スタックユーザーページを使用して、管理者以外のユーザーを AWS OpsWorks スタックにインポートします。
4. スタックごとにその [Permissions] ページを使用し、それぞれのユーザーに権限レベルを割り当てます。
5. 必要に応じて、適切に構成した IAM ポリシーを適用し、ユーザーの権限レベルをカスタマイズします。

ユーザー管理に関する推奨事項については、「[ベストプラクティス: アクセス権限の管理](#)」を参照してください。

IAM でのベストプラクティスの詳細については、IAM ユーザーガイドの「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

トピック

- [AWS OpsWorks スタックユーザーの管理](#)
- [AWS OpsWorks スタックごとのスタックユーザーアクセス許可の付与](#)
- [IAM ポリシーをアタッチして AWS OpsWorks スタックのアクセス許可を管理する](#)

- [ポリシーの例](#)
- [AWS OpsWorks スタックのアクセス許可レベル](#)

AWS OpsWorks スタックユーザーの管理

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

ユーザーを AWS OpsWorks スタックにインポートしてアクセス許可を付与する前に、まず個々のユーザーを作成しておく必要があります。IAM ユーザーを作成するには、まず、IAMFullAccess ポリシーで定義されているアクセス許可が付与されたユーザー AWS としてサインインします。次に、IAM コンソールを使用して、[スタックにアクセスする必要があるすべてのユーザーの IAM ユーザーを作成します](#)。AWS OpsWorks その後、これらのユーザーを AWS OpsWorks スタックにインポートし、次のようにユーザーアクセス許可を付与できます。

通常の AWS OpsWorks スタックユーザー

標準ユーザーには、アタッチされたポリシーは不要です。スタックがある場合は、通常、AWS OpsWorks スタックのアクセス許可は含まれません。代わりに、AWS OpsWorks スタックのアクセス許可 ページを使用して、以下のアクセス許可レベルのいずれかを通常のユーザーに stack-by-stack 割り当てます。

- Show 権限: ユーザーは、スタックを表示することはできますが、その他のオペレーションは一切実行できません。
- Deploy 権限: Show 権限に加えて、アプリケーションのデプロイと更新が許可されます。
- Manage 権限: Deploy 権限に加え、スタック管理操作を実行できます。たとえば、レイヤーやインスタンスを追加したり、[Permissions] ページを使ってユーザーの権限を設定したり、自分の SSH/RDP と sudo/admin 特権を有効にしたりすることができます。
- Deny 権限: スタックへのアクセスは拒否されます。

これらのパーミッションレベルが特定のユーザーにとって必要なものでない場合、IAMポリシーを適用することでユーザーのパーミッションをカスタマイズすることができます。例えば、

AWS OpsWorks スタックのアクセス許可 ページを使用して、アクセス許可の管理レベルをユーザーに割り当てます。これにより、すべてのスタック管理オペレーションを実行するアクセス許可がユーザーに付与されますが、スタックを作成またはクローンすることはできません。そこで、レイヤーの追加や削除を許可しないことでそれらの権限を制限したり、スタックの作成やクローンを許可することでそれらの権限を増強するポリシーを適用することができます。詳細については、「[IAM ポリシーをアタッチして AWS OpsWorks スタックのアクセス許可を管理する](#)」を参照してください。

AWS OpsWorks スタック管理ユーザー

管理ユーザーは、[AWSOpsWorks_FullAccess ポリシー](#) で定義されたアクセス許可を持つアカウント所有者または IAM ユーザーです。このポリシーには、Manage ユーザーに付与される権限に加え、[Permissions] ページでは付与することのできない以下のようなアクションの権限が含まれます。

- AWS OpsWorks スタックへのユーザーのインポート
- スタックの作成とクローン化

ポリシー全体については、「[ポリシーの例](#)」を参照してください。IAMポリシーを適用することによってのみユーザーに付与できる権限の詳細なリストについては、「[AWS OpsWorks スタックのアクセス許可レベル](#)」を参照してください。

トピック

- [ユーザーとリージョン](#)
- [AWS OpsWorks スタック管理ユーザーの作成](#)
- [AWS OpsWorks スタックの IAM ユーザーの作成](#)
- [AWS OpsWorks スタックへのユーザーのインポート](#)
- [AWS OpsWorks スタックのユーザー設定の編集](#)

ユーザーとリージョン

AWS OpsWorks スタックユーザーは、作成されたリージョンエンドポイント内で使用できます。以下のどのリージョンにもユーザーを作成できます。

- 米国東部 (オハイオ) リージョン
- 米国東部(バージニア州北部) リージョン
- 米国西部 (オレゴン) リージョン

- US West (N. California) リージョン
- カナダ (中部) リージョン (API のみ。では利用できません AWS Management Console)
- アジアパシフィック (ムンバイ) リージョン
- アジアパシフィック (シンガポール) リージョン
- アジアパシフィック (シドニー) リージョン
- アジアパシフィック (東京) リージョン
- アジアパシフィック (ソウル) リージョン
- 欧州 (フランクフルト) リージョン
- 欧州 (アイルランド) リージョン
- 欧州 (ロンドン) リージョン
- 欧州 (パリ) リージョン
- 南米 (サンパウロ) リージョン

ユーザーを AWS OpsWorks スタックにインポートするときは、リージョンエンドポイントの 1 つにインポートします。ユーザーが複数のリージョンで使用できるようにするには、そのユーザーをそのリージョンにインポートする必要があります。AWS OpsWorks スタックユーザーをあるリージョンから別のリージョンにインポートすることもできます。同じ名前のユーザーが既にあるリージョンにユーザーをインポートすると、インポートされたユーザーは既存のユーザーを置き換えます。ユーザーのインポートの詳細については、[ユーザーのインポート](#)を参照してください。

AWS OpsWorks スタック管理ユーザーの作成

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタック管理ユーザーを作成するには、AWSOpsWorks_FullAccessポリシーをユーザーに追加します。これにより、そのユーザーに AWS OpsWorks スタックのフルアクセス許可が付与されます。管理ユーザーの作成について詳しくは、「[管理ユーザーの作成](#)」を参照してください。

Note

この AWSOpsWorks_FullAccess ポリシーでは、ユーザーは AWS OpsWorks スタックスタックを作成および管理できますが、スタックの IAM サービスロールを作成することはできません。既存のロールを使用する必要があります。スタックを最初に作成するユーザーには別途、「[管理権限](#)」で説明されている IAM 権限が必要です。このユーザーが最初のスタックを作成すると、AWS OpsWorks スタックは必要なアクセス許可を持つ IAM サービスロールを作成します。以後、opsworks:CreateStack 権限を持つユーザーは、そのロールを使って別のスタックを作成することができます。詳細については、「[AWS OpsWorks スタックがユーザーに代わって動作することを許可する](#)」を参照してください。

ユーザーを作成すると、必要に応じてカスタマー管理ポリシーを追加してユーザーの権限を微調整できます。たとえば、スタックの作成と削除を管理ユーザーに許可する一方で、新しいユーザーのインポートは禁止することができます。詳細については、「[IAM ポリシーをアタッチして AWS OpsWorks スタックのアクセス許可を管理する](#)」を参照してください。

複数の管理ユーザーがある場合は、ユーザーごとにアクセス許可を個別に設定する代わりに、AWSOpsWorks_FullAccess ポリシーを IAM グループに追加し、そのグループにユーザーを追加できます。

グループの作成の詳細については、「[IAM ユーザー](#)」を参照してください。グループを作成するときは、AWSOpsWorks_FullAccessポリシーを追加します。アクセスAWSOpsWorks_FullAccess許可を含むAdministratorAccessポリシーを追加することもできます。

既存のグループにアクセス権限を追加する方法については、「[IAM ユーザーグループへのポリシーのアタッチ](#)」を参照してください。

AWS OpsWorks スタックの IAM ユーザーの作成

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

IAM ユーザーを AWS OpsWorks スタックにインポートする前に、ユーザーを作成する必要があります。これは、[IAM console](#) (IAM コンソール)、コマンドライン、または API のいずれかを使用して実行できます。詳細な手順については、「[アカウントでの IAM ユーザーの作成 AWS](#)」を参照してください。

[管理ユーザー](#)とは異なり、権限を定義するためにポリシーをアタッチする必要はありません。権限は、[ユーザーを AWS OpsWorks スタックにインポート](#)した後で設定できます（「[ユーザー許可の管理](#)」を参照）。

IAMユーザーとグループの作成の詳細については、「[IAM の使用開始](#)」を参照してください。

AWS OpsWorks スタックへのユーザーのインポート

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

管理ユーザーは、ユーザーを AWS OpsWorks スタックにインポートできます。また、リージョンのエンドポイント間で AWS OpsWorks スタックユーザーをインポートすることもできます。ユーザーを AWS OpsWorks スタックにインポートするときは、スタックリージョンエンドポイントのいずれかにインポートします AWS OpsWorks 。ユーザーを複数のリージョンで使用できるようにするには、該当するリージョンにユーザーをインポートする必要があります。

コンソールでフェデレーティッドユーザーをインポートすることは明示的には不可能ですが、フェデレーティッドユーザーは、AWS OpsWorks スタックコンソールの右上にあるマイ設定を選択し、右上にあるユーザーを選択することで、暗黙的にユーザープロファイルを作成できます。ユーザーページで、API または CLI を使用して作成された、またはコンソールを通じて暗黙的に作成されたアカウントのフェデレーティッドユーザーは、非フェデレーティッド IAM ユーザーと同じ様にアカウントを管理できます。

ユーザーを AWS OpsWorks スタックにインポートするには

1. 管理ユーザーまたはアカウント所有者として AWS OpsWorks スタックにサインインします。
2. 右上の [Users] を選択して [Users] ページを開きます。

Users

The Users page lets you import IAM (Identity and Access Management) users into AWS OpsWorks, as well as OpsWorks users from other regions. After importing users, use the Permissions page to change their permissions and grant them access to stacks. Only an AWS account owner or a user with appropriate IAM permissions can change user settings on the Permissions page. To create users, open the IAM console.

Name	SSH Username	Self Management	Actions
Demc	demc	-	edit delete
Emms	emms	-	edit delete
Oggs	oggs	-	edit delete
elgg-test	elgg-test	<input checked="" type="checkbox"/>	edit delete
Robot	robot	-	edit delete
root	root	-	edit delete

[Import IAM Users to US East \(N. Virginia\)](#)
[Import OpsWorks users from another region to US East \(N. Virginia\)](#)

- [Import IAM Users to <#####>] を選択し、使用可能なユーザーアカウントのうちまだインポートされていないものを表示します。



- [Select all] チェックボックスをオンにするか、個別のユーザーを 1 つ以上選択します。完了したら、[Import](#) を選択します OpsWorks。

Note

ユーザーを AWS OpsWorks スタックにインポートした後、IAM コンソールまたは API を使用してアカウントからユーザーを削除しても、ユーザーは AWS OpsWorks スタックを通じて付与した SSH アクセスを自動的に失いません。また、ユーザーページ AWS OpsWorks を開き、ユーザーのアクション列で削除を選択して、スタックからユーザーを削除する必要があります。

AWS OpsWorks スタックユーザーをあるリージョンから別のリージョンにインポートするには

AWS OpsWorks スタックユーザーは、作成されたリージョンエンドポイント内で使用できます。[ユーザーとリージョン](#) に表示されているリージョンでユーザーを作成できます。

AWS OpsWorks スタックユーザーを 1 つのリージョンから、ユーザーリストが現在フィルタリングされているリージョンにインポートできます。インポート先のリージョンにすでに同じ名前のユーザーが設定されている場合、既存のユーザーはインポートされたユーザーに置き換わります。

- 管理ユーザーまたはアカウント所有者として AWS OpsWorks スタックにサインインします。
- 右上の [Users] を選択して [Users] ページを開きます。複数のリージョンに AWS OpsWorks スタックユーザーがある場合は、フィルターコントロールを使用して、ユーザーをインポートするリージョンをフィルタリングします。

Users

The Users page lets you import IAM (Identity and Access Management) users into AWS OpsWorks, as well as OpsWorks users from other regions. After importing users, use the Permissions page to change their permissions and grant them access to stacks. Only an AWS account owner or a user with appropriate IAM permissions can change user settings on the Permissions page. To create users, open the IAM console.

Name	SSH Username	Self Management	Actions
Demo	demo	-	edit delete
Emma	emma	-	edit delete
Oiga	oiga	-	edit delete
oiga-test	oiga-test	✓	edit delete
Robot	robot	-	edit delete
root	root	-	edit delete

[+ Import IAM Users to US East \(N. Virginia\)](#)
[+ Import OpsWorks users from another region to US East \(N. Virginia\)](#)

- 別のリージョンから <###リージョン> に AWS OpsWorks スタックユーザーをインポートを選択します。

OpsWorks Users

Filter: US West (Oregon)

Name	SSH User Name	Self Management	Actions
	techwriters- -i	-	edit
tw-	tw-	-	edit delete
tw-	tw-	-	edit delete
tw-	tw-	-	edit delete

[+ Import IAM users to US West \(Oregon\)](#)
[+ Import OpsWorks users from another region to US West \(Oregon\)](#)

OpsWorks users are created and stored regionally. You can import users from another region to this region, US West (Oregon). Duplicate users are replaced by users that you import. [Learn more.](#)

Step 1.

Select the region from which you want to import users. Asia Pacific (Mumbai)

Step 2.

Select the user(s) that you want to import to this region, and then choose **Import to this region.**

Select all users
 TechWritersAdminAccess,

Cancel [Import to this region](#)

- AWS OpsWorks スタックユーザーをインポートするリージョンを選択します。
- インポートするユーザーを選択するか (複数可)、すべてのユーザーを選択して、[Import to this region] を選択します。AWS OpsWorks スタックがインポートされたユーザーをユーザーリストに表示するまで待ちます。

AWS OpsWorks スタックの外部で作成された Unix IDs とユーザー

AWS OpsWorks は、AWS OpsWorks スタックインスタンスの Unix ID (UID) 値を 2000 から 4000 の間で割り当てます。AWS OpsWorks は 2000-4000 範囲の UID を予約するため、の外部で作成したユーザー AWS OpsWorks (クックブックレシピを使用するか、IAM AWS OpsWorks からユーザーをインポートするなど) は、別のユーザーのスタックによって AWS OpsWorks 上書きされる UID を持つことができます。これにより、AWS OpsWorks スタックの外部で作成したユーザーがデータバグの検索結果に表示されなかったり、AWS OpsWorks スタックの組み込み `sync_remote_users` オペレーションから除外されたりする可能性があります。

外部プロセスでは、AWS OpsWorks スタックが上書きできる UID を持つユーザーを作成することもできます。たとえば一部のオペレーティングシステムパッケージはインストール後のプロセスの一環としてユーザーを作成することができます。ユーザーまたはソフトウェアプロセスが、デフォルトである UID を明示的に指定せずに Linux ベースのオペレーティングシステムにユーザーを作成する場合、AWS OpsWorks スタックによって割り当てられた UID は `#### AWS OpsWorks UID #### + 1` です。

ベストプラクティスとして、AWS OpsWorks スタックユーザーを作成し、AWS OpsWorks スタックコンソールで AWS CLI、または AWS SDK を使用してアクセスを管理します。の外部にある AWS OpsWorks スタックインスタンスでユーザーを作成する場合は AWS OpsWorks、4000 を超える `UnixID` 値を使用します。

AWS OpsWorks スタックのユーザー設定の編集

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

ユーザーをインポートした後、次の手順に従って、それらのユーザーの設定を編集することができます。

ユーザーの設定を編集するには

1. ユーザーページのユーザーのアクション列で **編集** を選択します。

2. 次の設定を指定できます。

Self Management

はい を選択して、ユーザーが MySettings ページを使用して個人 SSH キーを指定できるようにします。

Note

また、[DescribeMyUserProfile](#)および [UpdateMyUserProfile](#) アクションのアクセス許可を付与する IAM ポリシーを IAM アイデンティティに追加することで、自己管理を有効にすることもできます。

Public SSH key

(オプション) ユーザーのパブリック SSH キーを入力します。このキーは、ユーザーの [My Settings] ページに表示されます。自己管理が有効にされている場合、ユーザーは [My Settings] を編集し、独自のキーを指定できます。詳細については、「[ユーザーのパブリック SSH キーの登録](#)」を参照してください。

AWS OpsWorks スタックはすべての Linux インスタンスにこのキーをインストールします。ユーザーは関連するプライベートキーを使用してログインできます。詳細については、「[SSH でのログイン](#)」を参照してください。Windows のスタックでこのキーを使用することはできません。

アクセス許可

(オプション) 各スタックに対してユーザーが持つ権限レベルを、各スタックの [Permissions] ページで個別に設定するのではなく、一箇所で設定します。権限レベルの詳細については、「[スタックごとの権限の付与](#)」を参照してください。

User windows-test-user

Name windows-test-user

ARN arn:aws:iam::645732743964:user/windows-test-user

Self Management No

SSH Username windows-test-user

Public SSH key

The user will be created on **linux-based instances** if they have a **Public SSH Key**.
Clearing the public key will cause all SSH logins of the user to be deleted on **linux-based instances**.
Running processes will be terminated.

Permissions

Stack	Permission level					Instance access	
	Deny	IAM Policies Only	Show	Deploy	Manage	SSH / RDP	sudo / admin
CLITest	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>
Chef9Test	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>
EC2Register	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
JavaStack	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>

AWS OpsWorks スタックごとのスタックユーザーアクセス許可の付与

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックのユーザーアクセス許可を管理する最も簡単な方法は、スタックのアクセス許可ページを使用することです。各スタックには、その権限を付与するページがそれぞれに存在します。

何らかの権限設定に変更を加える場合は、管理ユーザーまたは Manage ユーザーとしてサインインする必要があります。リストには、AWS OpsWorks スタックにインポートされたユーザーのみが表示されます。ユーザーの作成とインポートの方法については、「[ユーザーの管理](#)」を参照してください。

デフォルトの権限レベルはIAMポリシーのみで、IAMポリシーにある権限のみがユーザーに付与されます。

- IAM または別のリージョンからユーザーをインポートする際、ユーザーは既存のすべてのスタックのリストに IAM Policies Only (IAMポリシーのみ) 権限レベルで追加されます。
- デフォルトでは、別のリージョンからインポートされたユーザーは、インポート先のリージョンのスタックにアクセスできません。別のリージョンからインポートしたユーザーがインポート先のリージョンを管理できるようにするには、インポートしたユーザーにこれらのスタックへのアクセス権限を割り当てる必要があります。
- 新しいスタックを作成すると、現在の全ユーザーが、IAM Policies Only の権限レベルでリストに追加されます。

トピック

- [ユーザーの権限の設定](#)
- [権限の表示](#)
- [IAM 条件キーを使用した一時認証情報の確認](#)

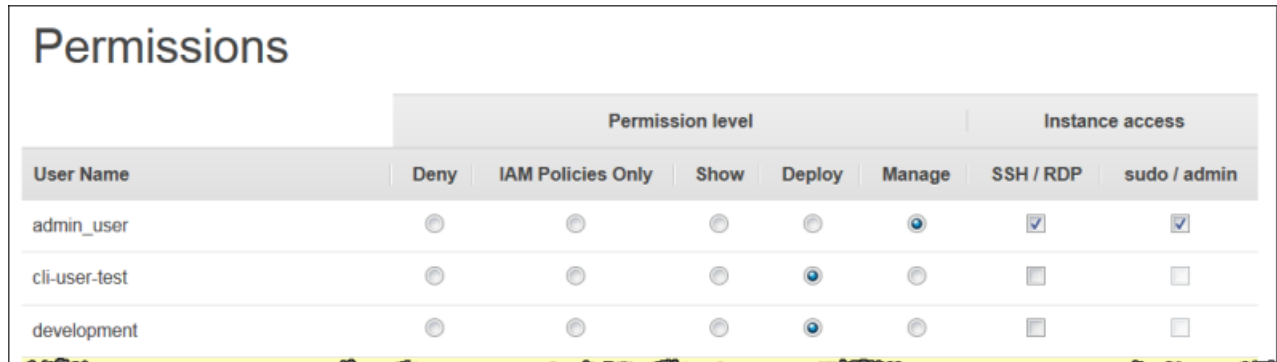
ユーザーの権限の設定

ユーザーの権限を設定するには

1. ナビゲーションペインの [アクセス権限] を選択します。
2. [Permissions] ページの [Edit] を選択します。
3. [Permission level] と [Instance access] の設定を変更します。
 - [Permissions level] の設定を使用して、いずれかの標準権限レベルをユーザーごとに割り当てます。そのスタックにユーザーがアクセスできるかどうかや、どのようなアクションを実行できるかが、この設定によって決まります。ユーザーが IAM ポリシーを持っている場合、AWS OpsWorks スタックは両方のアクセス許可セットを評価します。例については、「[ポリシーの例](#)」を参照してください。

- [Instance access SSH/RDP] 設定は、ユーザーがスタックのインスタンスに SSH (Linux) または RDP (Windows) アクセスできるかどうかを指定します。

SSH/RDP アクセスを許可した場合、オプションで sudo/admin を選択できます。これにより、ユーザーにスタックのインスタンスでの sudo (Linux) または管理者 (Windows) 特権が付与されます。



User Name	Permission level					Instance access	
	Deny	IAM Policies Only	Show	Deploy	Manage	SSH / RDP	sudo / admin
admin_user	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
cli-user-test	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>
development	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	<input type="checkbox"/>

ユーザーにはそれぞれ、次のいずれかの権限レベルを割り当てることができます。各レベルによって許可される一連のアクションについては、「[AWS OpsWorks スタックのアクセス許可レベル](#)」を参照してください。

拒否

AWS OpsWorks スタックにフルアクセス許可を付与する IAM ポリシーがある場合でも、ユーザーは AWS OpsWorks スタックに対してスタックアクションを実行できません。未リリース製品のスタックに対する一部のユーザーのアクセスを拒否する場合などに使用します。

IAM Policies Only

新しくインポートされたすべてのユーザー、および新しく作成されたスタックのすべてのユーザーに割り当てられるデフォルトのレベルです。アタッチされている IAM ポリシーによって、ユーザーの権限が決まります。ユーザーに IAM ポリシーがない場合、またはポリシーに明示的な AWS OpsWorks スタックアクセス許可がない場合、スタックにアクセスすることはできません。管理ユーザーは、IAM ポリシーによってすでにフルアクセス権限が付与されているため、通常このレベルが割り当てられます。

表示

ユーザーはスタックを表示できますが、一切オペレーションは実行できません。たとえば、マネージャは通常、アカウントのスタックを必要に応じて監視しますが、アプリケーションをデプロイしたりスタックに変更を加えたりする必要はありません。

デプロイ

Show 権限に加えて、アプリケーションをデプロイすることがユーザーに許可されます。たとえば、アプリケーション開発者は、必要に応じてスタックのインスタンスに対する更新をデプロイしますが、スタックにレイヤーやインスタンスを追加する必要はありません。

管理

Deploy 権限に加え、さまざまなスタック管理オペレーションがユーザーに許可されます。その例を次に示します。

- レイヤーやインスタンスを追加または削除する。
- スタックの [Permissions] ページを使用して権限レベルをユーザーに割り当てる。
- リソースを登録または登録を解除する。

たとえば、スタック内のインスタンスの数とタイプを適切に保ち、パッケージやオペレーティングシステムの更新処理を担うマネージャをスタックごとに指定できます。

Note

Manage レベルのユーザーがスタックを作成したり、スタックをクローン化したりすることはできません。これらの権限は、IAMポリシーによって付与されなければならない必要があります。例については、[アクセス許可の管理](#)を参照してください。

ユーザーが IAM ポリシーも持っている場合、AWS OpsWorks Stacks は両方のアクセス許可セットを評価します。これにより、ユーザーに権限レベルを割り当て、ポリシーを適用して、そのレベルで許可されるアクションを制限または強化することができます。たとえば、スタックの作成とクローン化を許可したり、リソースの登録と登録解除を拒否したりするポリシーを管理ユーザーにアタッチすることができます。そうしたポリシーの例については、「[ポリシーの例](#)」を参照してください。

Note

ユーザーのポリシーによって追加のアクションが許可された場合、[Permissions] ページの設定が見かけ上オーバーライドされます。例えば、ユーザーに [CreateLayer](#) アクションを

許可するポリシーがあっても、アクセス許可ページを使用してデプロイアクセス許可を指定すると、ユーザーは引き続きレイヤーの作成が許可されます。このルールの例外は、AWSOpsWorks_FullAccess ポリシーを持つユーザーに対してもスタックへのアクセスを拒否する拒否オプションです。詳細については、[「ポリシーを使用した AWS リソースへのアクセスの制御」](#)を参照してください。

権限の表示

[自己管理](#)が有効になっている場合、ユーザーは、右上の [My Settings] を選択することで、すべてのスタックに関して自分に割り当てられている権限レベルの概要を表示できます。ポリシーが [DescribeMyUserProfile](#) および [UpdateMyUserProfile](#) アクションのアクセス許可を付与している場合、ユーザーはマイ設定にアクセスすることもできます。

IAM 条件キーを使用した一時認証情報の確認

AWS OpsWorks スタックには、追加の認証ケースをサポートする認証レイヤーが組み込まれています (個々のユーザーのスタックへの読み取り専用アクセスまたは読み取り/書き込みアクセスの簡素化管理など)。この認証レイヤーは、一時認証情報の使用に依存しています。このため、IAM ドキュメントの [「条件キーの有無をチェックする条件演算子」](#) で説明されているように、ユーザーが長期的な認証情報を使用していることを確認したり、一時認証情報を使用しているユーザーのアクションをブロックしたりするために、aws:TokenIssueTime 条件を使用することはできません。

IAM ポリシーをアタッチして AWS OpsWorks スタックのアクセス許可を管理する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

IAM ポリシーをアタッチすることで、ユーザーの AWS OpsWorks スタックのアクセス許可を指定できます。次に示す一部の権限については、ポリシーのアタッチが必須となります。

- 管理ユーザー権限 (ユーザーのインポートなど)。
- 特定のアクションの権限 (スタックの作成、スタックのクローン化など)。

アタッチされたポリシーが必要となる全アクションの一覧については、「[AWS OpsWorks スタックのアクセス許可レベル](#)」を参照してください。

ポリシーを使用することによって、許可 ページで付与された権限レベルをカスタマイズすることもできます。このセクションでは、IAM ポリシーをユーザーに適用して AWS OpsWorks スタックのアクセス許可を指定する方法の概要を説明します。詳細については、「[AWS リソースのアクセス管理](#)」を参照してください。

IAM ポリシーは、1 つ以上の statements (ステートメント) を含んだ JSON オブジェクトです。各ステートメント要素には、一連の権限が記述され、それぞれ 3 つの基本要素が存在します。

[アクション]

権限の対象となるアクション。AWS OpsWorks スタックアクションは `として指定` します `opsworks:action`。Action には、特定のアクションを指定できます。例えば、`opsworks:CreateStack` は、[CreateStack](#) の呼び出しをユーザーに許可するかどうかを指定します。また、ワイルドカードを使用して複数のアクションをまとめて指定することもできます。例えば、すべての作成アクションを指定するには、`opsworks:Create*` とします。AWS OpsWorks スタックアクションの完全なリストについては、「[AWS OpsWorks スタック API リファレンス](#)」を参照してください。

[Effect] (効果)

指定したアクションを許可するか拒否するかを示します。

リソース

アクセス許可が影響する AWS リソース。AWS OpsWorks スタックには、スタックという 1 つのリソースタイプがあります。特定のスタックリソースの権限を指定するには、Resource 形式で、スタックの ARN を `arn:aws:opsworks:region:account_id:stack/stack_id/` に指定します。

また、ワイルドカードを使用することもできます。例えば、Resource を `*` に設定すると、すべてのリソースに対する権限が付与されます。

たとえば、次のポリシーでは、ID が `2860-2f18b4cb-4de5-4429-a149-ff7da9f0d8ee` であるスタックのインスタンスをユーザーが停止できないようにしています。

```
{
  "Version": "2012-10-17",
```

```
"Statement": [  
  {  
    "Action": "opsworks:StopInstance",  
    "Effect": "Deny",  
    "Resource": "arn:aws:opsworks:*:*:stack/2f18b4cb-4de5-4429-a149-ff7da9f0d8ee/"  
  }  
]  
}
```

IAM ユーザーに権限を付与する方法の詳細については、「https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_change-permissions.html#users_change_permissions-add-console」を参照してください。

IAM ポリシーの作成や変更を行う方法の詳細については、[許可とポリシー](#) を参照してください。AWS OpsWorks スタックポリシーの例については、「」を参照してください [ポリシーの例](#)。

ポリシーの例

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、AWS OpsWorks スタックユーザーに適用できる IAM ポリシーの例について説明します。

- [管理権限](#) では、管理ユーザーに権限を付与するために使用されるポリシーを説明します。
- [アクセス許可の管理](#) と [Deploy 権限](#) では、管理とデプロイの権限レベルを補足したり制限したりする目的でユーザーに適用できるポリシーの例を紹介します。

AWS OpsWorks スタックは、IAM ポリシーによって付与されたアクセス許可と、アクセス許可ページによって付与されたアクセス許可を評価することで、ユーザーのアクセス許可を決定します。詳細については、「[ポリシーを使用した AWS リソースへのアクセスの制御](#)」を参照してください。[Permissions] ページの権限の詳細については、「[AWS OpsWorks スタックのアクセス許可レベル](#)」を参照してください。

管理権限

IAM コンソール <https://console.aws.amazon.com/iam/> を使用して AWSOpsWorks_FullAccess ポリシーにアクセスし、このポリシーをユーザーにアタッチして、すべての AWS OpsWorks スタックアクションを実行するアクセス許可を付与します。ユーザーのインポートを管理ユーザーに許可するなど、特定の操作については、IAM の権限が必須となります。

AWS OpsWorks スタックがユーザーに代わって Amazon EC2 インスタンスなどの他の AWS リソースにアクセスできるようにする [IAM ロール](#) を作成する必要があります。通常、このタスクは、管理ユーザーに最初のスタックを作成し、AWS OpsWorks スタックにロールを作成させることで処理します。以後そのロールは、後続のすべてのスタックで使用することができます。詳細については、「[AWS OpsWorks スタックがユーザーに代わって動作することを許可する](#)」を参照してください。

最初のスタックを作成する管理ユーザーは、AWSOpsWorks_FullAccess ポリシーに含まれていない一部の IAM アクションに対するアクセス許可を持っている必要があります。ポリシーの Actions セクションに、次のアクセス許可を追加します。適切な JSON 構文のために、アクション間にカンマを追加し、アクションのリストの末尾にあるコンマを削除してください。

```
"iam:PutRolePolicy",  
"iam:AddRoleToInstanceProfile",  
"iam:CreateInstanceProfile",  
"iam:CreateRole"
```

アクセス許可の管理

Manage 権限レベルのユーザーは、レイヤーの追加と削除を含め、さまざまなスタック管理アクションを実行できます。このトピックでは、管理ユーザーにアタッチすることで、標準の権限を制限したり補足したりできるいくつかのポリシーについて説明します。

Manage ユーザーによるレイヤーの追加と削除を拒否する

管理 権限レベルを制限し、レイヤーの追加と削除を除くすべての 管理 アクションの実行を許可するには、次の IAM ポリシーをアタッチします。#####、*account_id*、および *stack_id* を設定に適した値に置き換えます。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",
```



```
"Action": [
  "opsworks:CreateLayer",
  "opsworks>DeleteLayer"
],
"Resource": "arn:aws:opsworks:region:account_id:stack/stack_id/"
}
]
}
```

スタックの作成とクローン化を Manage ユーザーに許可する

Manage (管理) 権限レベルでは、スタックの作成もクローン化も許可されません。以下の IAM ポリシーをアタッチすることで、ユーザーがスタックの作成やクローン化を作成できるように 管理を変更することができます。##### および `account_id` を設定に適した値に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRolePolicy",
        "iam:ListRoles",
        "iam:ListInstanceProfiles",
        "iam:ListUsers",
        "opsworks:DescribeUserProfiles",
        "opsworks:CreateUserProfile",
        "opsworks>DeleteUserProfile"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:opsworks::account_id:stack/*/",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "opsworks.amazonaws.com"
        }
      }
    }
  ]
}
```

```
]
}
```

Manage ユーザーによるリソースの登録と登録解除を拒否する

管理 権限レベルのユーザーは、[スタックへの Amazon EBS と Elastic IP アドレスリソースの登録および登録解除](#)を行うことができます。リソースの登録を除くすべての 管理 アクションを許可するように 管理 権限を制限するには、次のポリシーを適用します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "opsworks:RegisterVolume",
        "opsworks:RegisterElasticIp"
      ],
      "Resource": "*"
    }
  ]
}
```

ユーザーのインポートを Manage ユーザーに許可する

アクセス許可の管理レベルでは、ユーザーは AWS OpsWorks スタックにユーザーをインポートできません。ユーザーをインポートしたり削除したりできるように 管理 権限を補足するには、次の IAM ポリシーを適用します。#### および `account_id` を設定に適した値に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRolePolicy",
        "iam:ListRoles",
        "iam:ListInstanceProfiles",
        "iam:ListUsers",
        "iam:PassRole",
        "opsworks:DescribeUserProfiles",

```

```
    "opsworks:CreateUserProfile",
    "opsworks>DeleteUserProfile"
  ],
  "Resource": "arn:aws:iam:region:account_id:user/*",
  "Condition": {
    "StringEquals": {
      "iam:PassedToService": "opsworks.amazonaws.com"
    }
  }
}
```

Deploy 権限

Deploy 権限レベルのユーザーは、アプリケーションを作成することも削除することもできません。アプリケーションを作成したり削除したりできるように デプロイ 権限を補足するには、次の IAM ポリシーをアタッチします。####、*account_id*、および *stack_id* を設定に適した値に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "opsworks:CreateApp",
        "opsworks>DeleteApp"
      ],
      "Resource": "arn:aws:opsworks:region:account_id:stack/stack_id/"
    }
  ]
}
```

AWS OpsWorks スタックのアクセス許可レベル

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、AWS OpsWorks スタックのアクセス許可ページでの表示、デプロイ、アクセス許可レベルの管理で許可されるアクションを一覧表示します。また、ユーザーにIAMポリシーを適用することによってのみパーミッションを与えることができるアクションのリストも含まれています。

表示

表示 レベルでは、DescribeXYZ コマンドが許可されます。ただし次のコマンドは例外です。

```
DescribePermissions
DescribeUserProfiles
DescribeMyUserProfile
DescribeStackProvisioningParameters
```

ユーザーの自己管理を管理ユーザーが有効にした場合、表示 権限レベルのユーザーは、DescribeMyUserProfile と UpdateMyUserProfile も使用できます。自己管理の詳細については、「[ユーザー設定の編集](#)」を参照してください。

デプロイ

デプロイ レベルでは、表示 レベルで許可されるアクションに加えて次のアクションが許可されます。

```
CreateDeployment
UpdateApp
```

管理

管理 レベルでは、デプロイ レベルと 表示 レベルで許可されるアクションに加えて次のアクションが許可されます。

```
AssignInstance
AssignVolume
AssociateElasticIp
AttachElasticLoadBalancer
CreateApp
CreateInstance
```

```
CreateLayer
DeleteApp
DeleteInstance
DeleteLayer
DeleteStack
DeregisterElasticIp
DeregisterInstance
DeregisterRdsDbInstance
DeregisterVolume
DescribePermissions
DetachElasticLoadBalancer
DisassociateElasticIp
GrantAccess
GetHostnameSuggestion
RebootInstance
RegisterElasticIp
RegisterInstance
RegisterRdsDbInstance
RegisterVolume
SetLoadBasedAutoScaling
SetPermission
SetTimeBasedAutoScaling
StartInstance
StartStack
StopInstance
StopStack
UnassignVolume
UpdateElasticIp
UpdateInstance
UpdateLayer
UpdateRdsDbInstance
UpdateStack
UpdateVolume
```

Permissions That Require an IAM Policy (IAM ポリシーを必要とする権限)

適切な IAM ポリシーをユーザーに適用して、以下のアクションの権限を付与する必要があります。例については、「[ポリシーの例](#)」を参照してください。

```
CloneStack
CreateStack
CreateUserProfile
DeleteUserProfile
```

```
DescribeUserProfiles  
UpdateUserProfile
```

AWS OpsWorks スタックがユーザーに代わって動作することを許可する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは、ユーザーに代わってさまざまな AWS サービスとやり取りする必要があります。例えば、AWS OpsWorks スタックは Amazon EC2 とやり取りしてインスタンスを作成し、Amazon とやり取り CloudWatch してモニタリング統計を取得します。スタックを作成するときは、通常はサービスロールと呼ばれる IAM ロールを指定します。このロールは、AWS OpsWorks スタックに適切なアクセス許可を付与します。

The screenshot shows the configuration interface for an AWS OpsWorks stack. The fields are as follows:

- Stack name:** ShortStack
- Region:** US West (Oregon)
- VPC:** No VPC
- Default Availability Zone:** us-west-2a
- Default operating system:** Linux (selected), Windows
- Default SSH key:** Amazon Linux 2
- Default SSH key:** Do not use a default SSH key
- Chef version:** 12
- Use custom Chef cookbooks:** No
- Stack color:** A row of color swatches including purple, blue, teal, green, yellow, orange, and red.
- Advanced options:**
 - Default root device type:** Instance store (selected), EBS backed
 - IAM role:** aws-opsworks-service-role (highlighted with a red circle)

新しいスタックのサービスロールを指定するときは、次のいずれかの操作を実行します。

- 以前に作成した標準サービスロールを指定します。

通常は、最初のスタックの作成時に標準サービスロールを作成し、そのロールを以降のすべてのスタックに使用できます。

- IAM コンソールまたは API を使用して作成したカスタムサービスロールを指定します。

このアプローチは、標準サービスロールよりも制限されたアクセス許可を AWS OpsWorks スタックに付与する場合に便利です。

Note

最初のスタックを作成するには、IAM AdministratorAccessポリシーテンプレートで定義されたアクセス許可が必要です。これらの権限により、AWS OpsWorks スタックは新しいIAM サービスロールを作成し、[前述のように](#)ユーザーをインポートすることができます。以降のすべてのスタックでは、ユーザーは最初のスタック用に作成されたサービスロールを選択できます。スタックを作成するための完全な管理権限は必要ではありません。

標準サービスロールにより、次のアクセス許可が付与されます。

- すべての Amazon EC2 アクションを実行する (ec2:*)。
- CloudWatch 統計 () を取得します cloudwatch:GetMetricStatistics。
- Elastic Load Balancing を使用してサーバーにトラフィックを分散させる (elasticloadbalancing:*)。
- Amazon RDS インスタンスをデータベースサーバーとして使用する (rds:*)。
- IAM ロール (iam:PassRole) を使用して、AWS OpsWorks スタックと Amazon EC2 インスタンス間の安全な通信を提供します。

カスタムサービスロールを作成する場合は、スタックの管理に必要なすべてのアクセス許可が AWS OpsWorks スタックに付与されていることを確認する必要があります。次の JSON 例は、標準サービスロールのポリシーステートメントです。カスタムサービスロールのポリシーステートメントには、少なくとも以下のアクセス権限が含まれていなければなりません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:*",
        "iam:PassRole",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:DescribeAlarms",
        "ecs:*",
        "elasticloadbalancing:*",
        "rds:*"
      ],
      "Effect": "Allow",
    }
  ]
}
```



```
    "Resource": [
      "*"
    ],
    "Condition": {
      "StringEquals": {
        "iam:PassedToService": "ec2.amazonaws.com"
      }
    }
  }
]
```

また、サービスロールには信頼関係があります。AWS OpsWorks スタックによって作成されたサービスロールには、次の信頼関係があります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "StsAssumeRole",
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS OpsWorks スタックがユーザーに代わって動作するには、サービスロールにこの信頼関係が必要です。デフォルトのサービスロールを使用する場合は、信頼関係を変更しないでください。カスタムサービスロールを作成する場合は、以下のいずれかを実行することで信頼関係を指定します。

- [IAM console](#) (IAMコンソール) で [Create role] (ロールの作成) ウィザードを使用している場合は、[Choose a use case] (ユースケースの選択) で [Opsworks] を選択します。このロールには適切な信頼関係がありますが、暗黙のうちにアタッチされているポリシーはありません。AWS OpsWorks スタックにユーザーに代わって行動する権限を付与するために、以下を含むカスタマー管理ポリシーを作成し、新しいロールにアタッチします。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:DescribeAlarms",
      "cloudwatch:GetMetricStatistics",
      "ec2:*",
      "ecs:*",
      "elasticloadbalancing:*",
      "iam:GetRolePolicy",
      "iam:ListInstanceProfiles",
      "iam:ListRoles",
      "iam:ListUsers",
      "rds:*"
    ],
    "Resource": [
      "*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:PassRole"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "iam:PassedToService": "ec2.amazonaws.com"
      }
    }
  }
]
```

- AWS CloudFormation テンプレートを使用している場合は、テンプレートのリソースセクションに次のような内容を追加できます。

```
"Resources": {
  "OpsWorksServiceRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
```

```
    "Statement": [ {
      "Effect": "Allow",
      "Principal": {
        "Service": [ "opsworks.amazonaws.com" ]
      },
      "Action": [ "sts:AssumeRole" ]
    } ]
  },
  "Path": "/",
  "Policies": [ {
    "PolicyName": "opsworks-service",
    "PolicyDocument": {
      ...
    }
  } ]
}
```

AWS OpsWorks スタックでのサービス間の混乱した代理の防止

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

混乱した代理問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐため、AWS では、アカウント内のリソースへのアクセス権が付与されたサービスプリンシパルですべてのサービスのデータを保護するために役立つツールを提供しています。

スタックアクセスポリシーで [aws:SourceArn](#) および [aws:SourceAccount](#) グローバル条件コンテキストキーを使用して、AWS OpsWorks スタックが別のサービスに付与するアクセス許可をスタックに制限することをお勧めします。aws:SourceArn の値に Amazon S3 バケット ARN などのアカウント ID が含まれていない場合は、両方のグローバル条件コンテキストキーを使用して、アクセス許可を制限する必要があります。同じポリシーステートメントでこれらのグローバル条件コンテキストキーの両方を使用し、アカウント ID にaws:SourceArn の値が含まれていない場合、aws:SourceAccount 値と aws:SourceArn 値の中のアカウントには、同じアカウント ID を使用する必要があります。クロスサービスのアクセスにスタックを 1 つだけ関連付けたい場合は、aws:SourceArn を使用します。クロスサービスが使用できるように、アカウント内の任意のスタックを関連づけたい場合は、aws:SourceAccount を使用します。

の値は、AWS OpsWorks スタックの ARN `aws:SourceArn` である必要があります。

混乱した代理問題から保護するための最も効果的な方法は、AWS OpsWorks スタックの完全な ARN を指定しながら、aws:SourceArn グローバル条件コンテキストキーを使用することです。完全な ARN が不明な場合や、複数のサーバー ARN 場合には、ARN の不明な部分にワイルドカード (*) を含む aws:SourceArn グローバルコンテキスト条件キーを使用します。例えば `arn:aws:service:*:123456789012:*` です。

次のセクションでは、AWS OpsWorks スタックで aws:SourceArn および aws:SourceAccount グローバル条件コンテキストキーを使用して、混乱した代理問題を回避する方法を示します。

AWS OpsWorks スタックでの混乱した代理攻撃の防止

このセクションでは、スタックで AWS OpsWorks 混乱した代理攻撃を防ぐ方法と、スタックへのアクセスに使用している IAM ロールにアタッチできるアクセス許可ポリシーの例について説明します AWS OpsWorks。セキュリティのベストプラクティスとして、aws:SourceArn と aws:SourceAccount の条件キーを IAM ロールとほかのサービスとの信頼関係に追加することをお勧めします。信頼関係により、AWS OpsWorks スタックは、スタックスタックの作成または管理に必要な他の サービスでアクションを実行するロール AWS OpsWorks を引き受けることができます。

信頼関係を編集するために、aws:SourceArn と aws:SourceAccount 条件キーを追加するには

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. 左のナビゲーションペインで、[ロール] を選択します。
3. 検索ボックスで、AWS OpsWorks スタックへのアクセスに使用するロールを検索します。AWS マネージドロールは `aws-opsworks-service-role` です。

4. そのロールの **概要** ページで、**信頼関係 タブ**を選択します。
5. **信頼関係 タブ**で、**信頼ポリシーの編集** を選択します。
6. [**信頼ポリシーの編集**] ページで、少なくとも `aws:SourceArn` や `aws:SourceAccount` 条件キーの一つをポリシーに追加します。を使用して`aws:SourceArn`、クロスサービス (Amazon EC2 など) と AWS OpsWorks スタック間の信頼関係を、より制限の厳しい特定の AWS OpsWorks スタックスタックに制限します。`aws:SourceAccount` を追加して、クロスサービスと AWS OpsWorks スタック間の信頼関係を、制限の少ない特定のアカウントのスタックに制限します。次に例を示します。両方の条件キーを使用する場合、アカウント ID は同じでなければならないことに注意してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "arn:aws:opsworks:us-east-2:123456789012:stack/
EXAMPLEd-5699-40a3-80c3-22c32EXAMPLE/"
        }
      }
    }
  ]
}
```

7. 条件キーの追加が終了したら、[**更新ポリシー**] を選択します。

`aws:SourceArn` と `aws:SourceAccount` を使用してスタックへのアクセスを制限するロールのその他の例を以下に示します。

トピック

- [例: 特定のリージョンのスタックへのアクセス](#)

- [例: aws:SourceArn に複数のスタック ARN の追加](#)

例: 特定のリージョンのスタックへのアクセス

次のロール信頼関係ステートメントは、米国東部 (オハイオ) リージョン () のすべての AWS OpsWorks スタックスタックにアクセスしますus-east-2。リージョンは aws:SourceArn の ARN 値で指定されていますが、スタック ID の値はワイルドカード (*) であることに注意してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:opsworks:us-east-2:123456789012:stack/*"
        }
      }
    }
  ]
}
```

例: **aws:SourceArn** に複数のスタック ARN の追加

次の例では、アカウント ID 123456789012 の 2 つの AWS OpsWorks スタックスタックの配列へのアクセスを制限します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

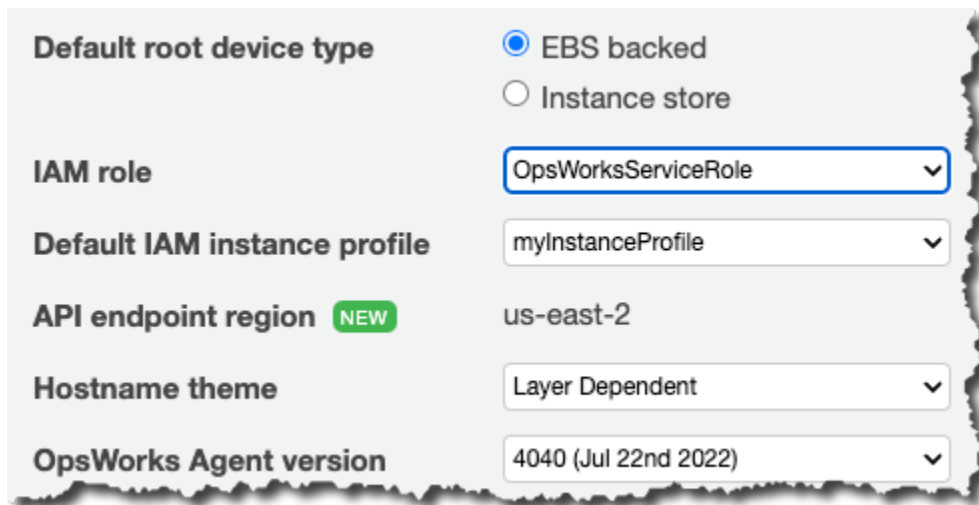
```
    "Service": "opsworks.amazonaws.com"
  },
  "Action": "sts:AssumeRole",
  "Condition": {
    "StringEquals": {
      "aws:SourceAccount": "123456789012"
    },
    "ArnEquals": {
      "aws:SourceArn": [
        "arn:aws:opsworks:us-east-2:123456789012:stack/unique_ID1",
        "arn:aws:opsworks:us-east-2:123456789012:stack/unique_ID2"
      ]
    }
  }
}
```

EC2 インスタンスで実行されるアプリケーションのアクセス許可の指定

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

スタックの Amazon EC2 インスタンスで実行されているアプリケーションが Amazon S3 バケットなどの他の AWS リソースにアクセスする必要がある場合は、適切なアクセス許可が必要です。アクセス許可を付与するには、インスタンスプロファイルを使用します。[AWS OpsWorks スタックスタックを作成するときに](#)、インスタンスごとにインスタンスプロファイルを指定できます。



Default root device type	<input checked="" type="radio"/> EBS backed <input type="radio"/> Instance store
IAM role	OpsWorksServiceRole
Default IAM instance profile	myInstanceProfile
API endpoint region NEW	us-east-2
Hostname theme	Layer Dependent
OpsWorks Agent version	4040 (Jul 22nd 2022)

[レイヤー設定を編集](#)して、レイヤーのインスタンスのプロファイルを指定することも可能です。

インスタンスプロファイルは IAM ロールを指定します。インスタンスで実行されているアプリケーションは、ロールのポリシーによって付与されるアクセス許可に従って、AWS リソースにアクセスするためにそのロールを引き受けることができます。アプリケーションがロールを引き受ける方法の詳細については、[「API 呼び出しを使用してロールを引き受ける」](#)を参照してください。

次のいずれかの方法でインスタンスプロファイルを作成することができます。

- IAM コンソールまたは API を使用してプロファイルを作成します。

詳細については、[「ロール \(委任とフェデレーション\)」](#)を参照してください。

- AWS CloudFormation テンプレートを使用してプロファイルを作成します。

IAM テンプレートにリソースを含める方法の例については、[「Identity and Access Management \(IAM\) Template Snippets」](#)を参照してください。

インスタンスプロファイルには、信頼関係と、AWS リソースへのアクセス許可を付与するアタッチされたポリシーが必要です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
```



```
    "Service": "ec2.amazonaws.com"
  },
  "Action": "sts:AssumeRole"
}
]
```

インスタンスプロファイルには、AWS OpsWorks スタックがユーザーに代わって動作するために、この信頼関係が必要です。デフォルトのサービスロールを使用する場合は、信頼関係を変更しないでください。カスタムサービスロールを作成する場合は、次のように信頼関係を指定します。

- コンソールでロールの作成ウィザードを使用している場合は、ウィザードの2ページ目のサービスロールで Amazon EC2 ロールタイプを指定します。 [IAM](#) AWS
- AWS CloudFormation テンプレートを使用している場合は、テンプレートのリソースセクションに次のような内容を追加できます。

```
"Resources": {
  "OpsWorksEC2Role": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Statement": [ {
          "Effect": "Allow",
          "Principal": {
            "Service": [ "ec2.amazonaws.com" ]
          },
          "Action": [ "sts:AssumeRole" ]
        } ]
      },
      "Path": "/"
    }
  },
  "RootInstanceProfile": {
    "Type": "AWS::IAM::InstanceProfile",
    "Properties": {
      "Path": "/",
      "Roles": [ {
        "Ref": "OpsWorksEC2Role"
      } ]
    }
  }
}
```

```
}
```

インスタンスプロファイルを作成すると、その時点でプロファイルのロールに適切なポリシーをアタッチできます。スタックを作成したら、[IAMコンソール](#)または `aws iam attach-policy --profile-name <プロファイル名> --policy-arn <ポリシーARN>` を使用して、プロファイルのロールAPIに適切なポリシーをアタッチする必要があります。例えば、次のポリシーは、`amzn-s3-demo-bucket` という名前の Amazon S3 バケット内のすべてのオブジェクトへのフルアクセスを許可します。置換 `region` および `amzn-s3-demo-bucket` と、設定に適した値。

```
{
  "Version": "2012-10-17",
  "Statement": [ {
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:region::amzn-s3-demo-bucket/*"
  }
]
```

インスタンスプロファイルの作成および使用方法の例については、「[Amazon S3 バケットの使用](#)」を参照してください。

アプリケーションがインスタンスプロファイルを使用してEC2インスタンスAPIから AWS OpsWorks スタックを呼び出す場合、ポリシーでは、AWS OpsWorks スタックやその他の AWS サービスに対する適切な `iam:PassRole` アクションに加えて、アクションを許可する必要があります。iam:PassRole アクセス許可により、AWS OpsWorks スタックがユーザーに代わってサービスロールを引き受けることができるようになります。AWS OpsWorks スタックの詳細については、[AWS OpsWorks API 「リファレンスAPI」](#) を参照してください。

以下は、EC2インスタンスから AWS OpsWorks スタックアクションと Amazon EC2または Amazon S3 アクションを呼び出すことを許可する IAMポリシーの例です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:*",
        "s3:*",
        "opsworks:*"
      ]
    }
  ]
}
```

```
    "iam:PassRole"
  ],
  "Resource": "arn:aws:ec2:region:account_id:instance/*",
  "Condition": {
    "StringEquals": {
      "iam:PassedToService": "opsworks.amazonaws.com"
    }
  }
}
]
```

Note

を許可しない場合 iam:PassRole、AWS OpsWorks スタックアクションを呼び出す試行は失敗し、次のようなエラーが発生します。

```
User: arn:aws:sts::123456789012:federated-user/Bob is not authorized
to perform: iam:PassRole on resource:
arn:aws:sts::123456789012:role/OpsWorksStackIamRole
```

EC2 インスタンスでロールを使用してアクセス許可を取得する方法の詳細については、[「ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションにAWSリソースへのアクセスを許可する」](#)を参照してください。AWS Identity and Access Management

SSH アクセスの管理

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは、Linux スタックと Windows スタックの両方の SSH キーをサポートします。

- Linux インスタンスでは、[エージェント CLI](#) コマンドを実行するためなど、インスタンスにログインするときに SSH を使用できます。

詳細については、「[SSH でのログイン](#)」を参照してください。

- Windows インスタンスの場合、SSH キーを使用してインスタンスの管理者パスワードを取得してから、そのパスワードを使用して RDP でログインできます。

詳細については、「[RDP でのログイン](#)」を参照してください。

認証は、パブリックキーとプライベートキーで構成される SSH キーペアに基づきます。

- インスタンスにパブリックキーをインストールします。

場所は特定のオペレーティングシステムによって異なりますが、AWS OpsWorks スタックが詳細を処理します。

- プライベートキーをローカルに保存し、SSH クライアント (ssh.exe など) で指定して、インスタンスにアクセスします。

SSH クライアントはプライベートキーを使用して、インスタンスに接続します。

スタックのユーザーに SSH アクセス権限を与えるには、SSH キーペアを作成し、パブリックキーをスタックのインスタンスにインストールして、プライベートキーを安全に管理する方法が必要です。

Amazon EC2は、インスタンスにパブリック SSH キーをインストールする簡単な方法を提供します。Amazon EC2 コンソールまたは API を使用して、使用する予定の AWS リージョンごとに 1 つまたは複数のキーペアを作成できます。Amazon EC2 は公開キーを AWS に保存し、ユーザーはプライベートキーをローカルに保存します。インスタンスの起動時に、リージョンのいずれかのキーペアを指定すると、そのペアは Amazon EC2 でインスタンスに自動的にインストールされます。次に、対応するプライベートキーを使用して、インスタンスにログインします。詳細については、「[Amazon EC2 のキーペア](#)」を参照してください。

AWS OpsWorks スタックでは、スタックの作成時にリージョンの Amazon EC2 キーペアの 1 つを指定し、オプションで各インスタンスの作成時に別のキーペアで上書きできます。AWS OpsWorks スタックが対応する Amazon EC2 インスタンスを起動するときにキーペアが指定され、Amazon EC2 でインスタンスにパブリックキーがインストールされます。その後、標準の Amazon EC2 インスタンスと同様に、プライベートキーを使用してログインしたり、管理者パスワードを取得したりできます。詳細については、「[Amazon EC2 キーをインストールする](#)」を参照してください。

Amazon EC2 のキーペアの使用は便利ですが、2 つの重要な制限があります。

- Amazon EC2 キーペアは、特定の AWS リージョンに関連付けられています。

複数のリージョンで作業する場合、複数のキーペアを管理しなければなりません。

- 1 つのインスタンスにインストールできる Amazon EC2 キーペアは 1 つだけです。

複数のユーザーにログインを許可する場合、そのすべてがプライベートキーのコピーを所有する必要がありますが、セキュリティ上、これは推奨されません。

Linux スタックの場合、AWS OpsWorks スタックは SSH キーペアを管理するためのよりシンプルで柔軟な方法を提供します。

- ユーザーは各自、個人キーペアを登録します。

で説明されているように、プライベートキーをローカルに保存し、パブリックキーを AWS OpsWorks スタックに登録します [ユーザーのパブリック SSH キーの登録](#)。

- スタックのユーザーアクセス権限を設定した場合、どのユーザーにスタックのインスタンスに対する SSH アクセスを付与するかを指定します。

AWS OpsWorks スタックは、承認された各ユーザーのシステムユーザーをスタックのインスタンスに自動的に作成し、パブリックキーをインストールします。これでユーザーは、「[SSH でのログイン](#)」で説明しているように、対応するプライベートキーを使用してログインできます。

個人 SSH キーを使用すると、以下のような利点があります。

- インスタンスでキーを手動で設定する必要はありません。AWS OpsWorks スタックはすべてのインスタンスに適切なパブリックキーを自動的にインストールします。
- AWS OpsWorks スタックは、承認されたユーザーの個人パブリックキーのみをインストールします。

承認されていないユーザーは個人プライベートキーを使用してインスタンスにアクセスすることはできません。Amazon EC2 キーペアを使用すると、対応するプライベートキーを持つすべてのユーザーは、SSH アクセス権限の有無にかかわらず、ログインできます。

- SSH アクセスが不要になったユーザーは、[\[Permissions\] ページ](#)を使用して、ユーザーの SSH/RDP アクセス権限を無効にすることができます。

AWS OpsWorks スタックは、スタックのインスタンスからパブリックキーをすぐにアンインストールします。

- どの AWS リージョンにも同じキーを使用できます。

ユーザーは、1 個のプライベートキーを管理するだけで済みます。

- プライベートキーを共有する必要がありません。

各ユーザーが、独自のプライベートキーを使用できます。

- キーを簡単に更新できます。

管理者またはユーザーが [My Settings] でパブリックキーを更新すると、AWS OpsWorks スタックによってインスタンスが自動的に更新されます。

Amazon EC2 キーをインストールする

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

スタックを作成する際に、スタック内のすべてのインスタンスにデフォルトでインストールされる Amazon EC2 の SSH キーを指定することができます。

Add Stack

Name	<input type="text"/>
Region	Asia Pacific (Singapore) ▼
VPC NEW	No VPC ▼
Default Availability Zone	ap-southeast-1a ▼
Default operating system	Amazon Linux ▼
Default root device type	<input checked="" type="radio"/> Instance store <input type="radio"/> EBS backed
IAM role	aws-opsworks-service-role-alpha ▼
Default SSH key	somekey ▼ <i>Existing keys</i> somekey <i>None</i> Do not use a default SSH key
Default IAM instance profile	Layer Dependent ▼
Host name theme	
Stack color	

[Advanced](#) **NEW** »

[Default SSH key (デフォルトのSSHキー)] リストに、AWS アカウントの Amazon EC2 キーが表示されます。次のいずれかを試すことができます。

- リストから適切なキーを選択します。
- キーを指定しない場合は、[Do not use a default SSH key] を選択します。

[Do not use a default SSH key] を選択した場合、またはスタックのデフォルトのキーを上書きする場合は、インスタンスの作成時にキーを指定できます。

PHP App Server

No instances. [Add an instance.](#)

New Existing

Host name

Size

Availability Zone

Scaling type
 24/7
 time-based
 load-based

SSH key
Existing Keys
somekey
None
Do not use a default SSH key

Operating system

Architecture
 64bit
 32bit

インスタンスを起動すると、AWS OpsWorks スタックはパブリックキーを `authorized_keys` ファイルにインストールします。

ユーザーのパブリック SSH キーの登録

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

ユーザーのパブリック SSH キーを登録するには、次の 2 つの方法があります。

- 管理ユーザーが、1人または複数のユーザーにパブリック SSH キーを割り当て、そのユーザーに対応するプライベートキーを提供します。
- 管理ユーザーが、1人または複数のユーザーの自己管理機能を有効にします。

その後、これらのユーザーは自身のパブリック SSH キーを指定できるようになります。

自己管理機能を有効にする方法や、管理ユーザーが他のユーザーにパブリックキーを割り当てる方法の詳細については、「[ユーザー設定の編集](#)」を参照してください。

PuTTY 端末で SSH を使用して Linux ベースのインスタンスに接続するには、追加ステップが必要です。詳細については、AWS ドキュメントの「[PuTTY を使用した Windows から Linux インスタンスへの接続](#)」および「[インスタンスへの接続に関するトラブルシューティング](#)」を参照してください。

以下に説明しているのは、自己管理機能が有効化されたユーザーがパブリックキーを指定する方法です。

SSH パブリックキーを指定するには

1. SSH キーペア を作成します。

最も簡単なアプローチは、キーペアをローカルに生成する方法です。詳細については、「[独自のキーを作成して Amazon EC2 にインポートする方法](#)」を参照してください。

Note

PuTTYgen を使用してキーペアを生成する場合は、パブリックキーからパブリックキーをコピーして OpenSSH `authorized_keys` ファイルボックスに貼り付けます。パブリックキーの保存をクリックすると、サポートされていない形式でパブリックキーが保存されます MindTerm。

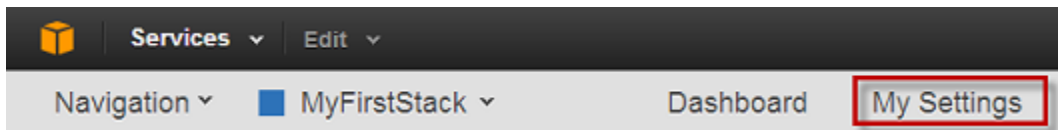
2. 自己管理が有効になっている IAM ユーザーとして AWS OpsWorks スタックコンソールにサインインします。

Important

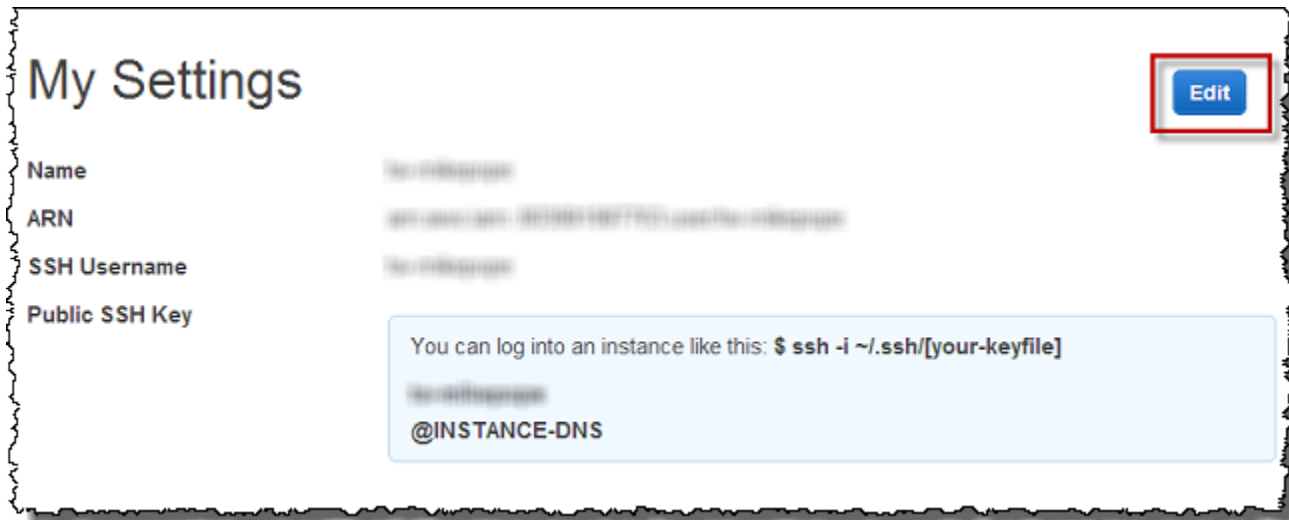
アカウント所有者として、または自己管理が有効になっていない IAM ユーザーとしてサインインした場合、AWS OpsWorks スタックにはマイ設定が表示されません。管

理ユーザーまたはアカウント所有者である場合は、代わりに [Users] ページに移動して [ユーザー設定を編集する](#) ことで、SSH キーを指定できます。

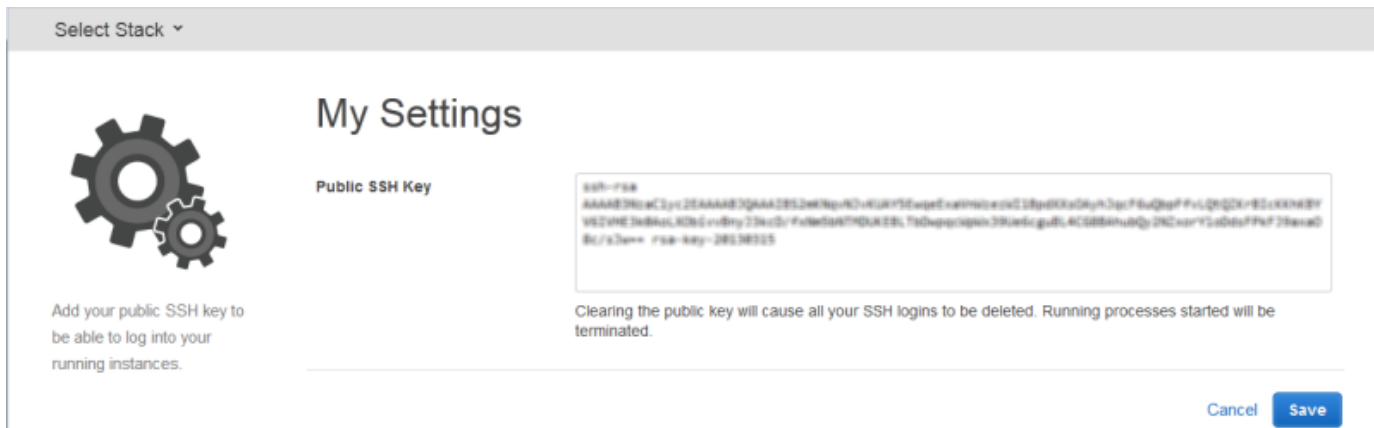
3. 自分の設定 を選択すると、サインインしている ユーザーの設定が表示されます。



4. [My Settings] ページで、[Edit] をクリックします。



5. [Public SSH Key] ボックスで、SSH パブリックキーを入力し、[Save] をクリックします。



⚠ Important

組み込みの MindTerm SSH クライアントを使用して Amazon EC2 インスタンスに接続するには、ユーザーが IAM ユーザーとしてサインインし、AWS OpsWorks スタックにパブ

リック SSH キーが登録されている必要があります。詳細については、「[組み込み MindTerm SSH クライアントの使用](#)」を参照してください。

Linux セキュリティ更新の管理

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

セキュリティの更新

Linux オペレーティングシステムのプロバイダーは定期的な更新を提供し、その多くがオペレーティングシステムのセキュリティパッチですが、中にはインストールされているパッケージに対する更新も含まれます。インスタンスのオペレーティングシステムは、最新のセキュリティパッチで最新の状態にする必要があります。

デフォルトでは、AWS OpsWorks スタックは、インスタンスの起動が完了した後、セットアップ中に最新の更新を自動的にインストールします。AWS OpsWorks スタックは、アプリケーションサーバーの再起動などの中断を避けるため、インスタンスがオンラインになった後に更新を自動的にインストールしません。代わりに、中断の可能性を最小限に抑えるため、オンラインインスタンスへの更新はユーザーが自身で管理します。

次のいずれかの方法を使用して、オンラインインスタンスを更新することをお勧めします。

- 新しいインスタンスを作成して、現在のオンラインインスタンスを置き換えます。次に、現在のインスタンスを削除します。

新しいインスタンスに、セットアップ時にインストールされたセキュリティパッチの最新のセットが適用されます。

- Chef 11.10 以前のスタックの Linux ベースのインスタンスでは、[Update Dependencies スタック コマンド](#)を実行します。このコマンドは指定したインスタンスに、セキュリティパッチの現在のセットと他の更新をインストールします。

これらのアプローチの両方で、AWS OpsWorks スタックは `yum update` Amazon Linux および `Red Hat Enterprise Linux (RHEL)` または `Ubuntu apt-get update` に対して を実行して更新を実行します。それぞれのディストリビューションで更新の処理方法は若干異なるため、更新がインスタンスにどのような影響を及ぼすか正確に把握するため、関連リンクの情報を確認してください。

- [Amazon Linux (Amazon Linux)] – Amazon Linux は、セキュリティパッチ更新のほか、パッケージ更新など、機能の更新をインストールします。

詳細については、「[Amazon Linux AMI に関するよくある質問](#)」を参照してください。

- [Ubuntu (Ubuntu)] – Ubuntu では主にセキュリティパッチのインストールに限定されますが、制限された数の重要な修正でパッケージ更新をインストールします。

詳細については、[Ubuntu Wiki の LTS に関するページ](#)を参照してください。

- CentOS – CentOS の更新では、一般的に前のバージョンとのバイナリ互換性が維持されます。
- RHEL – RHEL の更新では、一般的に前のバージョンとのバイナリ互換性が維持されます。

詳細については、「[Red Hat Enterprise Linux ライフサイクル](#)」を参照してください。

特定のパッケージバージョンを指定するなど、更新をより詳細に制御したい場合は、[CreateInstance](#)、または [UpdateLayer](#) アクション [CreateLayer](#)、または同等の [AWS SDK UpdateInstance](#) メソッドまたは [AWS CLI](#) コマンドを使用して、`InstallUpdatesOnBoot` パラメータを に設定することで、自動更新を無効にすることができます `false`。次の例に、AWS CLI を使用して、既存のレイヤーのデフォルト設定として `InstallUpdatesOnBoot` を無効にする方法を示します。

```
aws opsworks update-layer --layer-id layer ID --no-install-updates-on-boot
```

その後、更新を自身で管理する必要があります。たとえば、次のいずれかの方法を使用できます。

- [適切なシェルコマンドを実行](#)するカスタムレシピを実装して、希望の更新をインストールします。

システムの更新は[ライフサイクルイベント](#)とは本質的に無関係であるため、カスタムクックブックにレシピを含めて、それを[手動で実行](#)します。パッケージ更新の場合、シェルコマンドの代わりに、[yum_package](#) (Amazon Linux) または [apt_package](#) (Ubuntu) リソースを使用することもできます。

- [SSH で各インスタンスにログイン](#)し、適切なコマンドを手動で実行してください。

セキュリティグループの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

セキュリティグループ

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

各 Amazon EC2 インスタンスには、むしろファイアウォールに近い、インスタンスのネットワークトラフィックを管理する 1 つ以上の関連付けられたセキュリティグループがあります。セキュリティグループには 1 つ以上のルールがあり、各ルールでは許可するトラフィックの特定のカテゴリが指定されます。ルールでは、以下を指定します。

- 許可するトラフィックのタイプ (SSH、HTTP など)
- トラフィックのプロトコル (TCP、UDP など)
- トラフィックの発信元の IP アドレスの範囲
- トラフィックの許可するポート範囲

セキュリティグループには、次の 2 種類のルールがあります。

- 着信ネットワークトラフィックを規定するインバウンドルール。

たとえば一般的に、アプリケーションサーバーのインスタンスには、IP アドレスからポート 80 へのインバウンド HTTP トラフィックを許可するインバウンドルールと、特定の一連の IP アドレスからのポート 22 へのインバウンド SSH トラフィックを許可する別のインバウンドルールがあります。

- アウトバウンドルールは、アウトバウンドネットワークトラフィックを規定します。

一般的な方法としては、アウトバウンドトラフィックを許可するデフォルト設定を使用します。

セキュリティグループの詳細については、「[Amazon EC2 Security Groups](#) (Amazon EC2 セキュリティグループ)」を参照してください。

リージョンで初めてスタックを作成すると、AWS OpsWorks Stacks は適切なルールセットを持つ各レイヤーの組み込みセキュリティグループを作成します。すべてのグループには、すべてのアウトバウンドトラフィックを許可するデフォルトのアウトバウンドルールが設定されます。一般的に、インバウンドルールは以下を許可します。

- 適切な AWS OpsWorks スタックレイヤーからのインバウンド TCP、UDP、ICMP トラフィック
- ポート 22 のインバウンド TCP トラフィック (SSH ログイン)

Warning

デフォルトのセキュリティグループの設定では、任意のネットワークの場所 (0.0.0.0/0) に対して SSH (ポート 22) が開かれます。これにより、すべての IP アドレスからの SSH によるインスタンスへのアクセスが許可されます。実稼働環境では、特定の IP アドレスまたはアドレス範囲からの SSH によるアクセスのみを許可する設定を使用する必要があります。デフォルトのセキュリティグループを作成直後に更新するか、カスタムセキュリティグループを使用します。

- ウェブサーバーレイヤーの場合、ポート 80 (HTTP) および 443 (HTTPS) へのすべてのインバウンド TCP および UDP トラフィック

Note

組み込みの AWS-OpsWorks-RDP-Server セキュリティグループは、RDP アクセスを許可するため、すべての Windows インスタンスに割り当てられます。ただし、デフォルトではルールが何もありません。Windows スタックを実行しており、RDP を使用してインスタン

スにアクセスする場合、RDP アクセスを許可するインバウンドルールを追加する必要があります。詳細については、「[RDP でのログイン](#)」を参照してください。

各グループの詳細については、[\[Amazon EC2 console \(Amazon EC2 コンソール\)\]](#) に移動し、ナビゲーションペインで [\[Security Groups \(セキュリティグループ\)\]](#) を選択して、適切なレイヤーのセキュリティグループを選択します。例えば、AWS-OpsWorks-Default-Server はすべてのスタックのデフォルトの組み込みセキュリティグループであり、AWS-OpsWorks-WebApp は Chef 12 サンプルスタックのデフォルトの組み込みセキュリティグループです。

Note

AWS OpsWorks スタックセキュリティグループを誤って削除した場合、再作成するには、AWS OpsWorks スタックにタスクを実行させることをお勧めします。同じ AWS リージョンに新しいスタックを作成するだけで、AWS OpsWorks VPC が存在する場合は、削除したセキュリティグループを含むすべての組み込みセキュリティグループが自動的に再作成されます。その後、今後使用しないスタックを削除します。セキュリティグループは残ります。セキュリティグループを手動で再作成する場合は、グループ名の大文字と小文字の設定を含め、元のセキュリティグループとまったく同じである必要があります。

さらに、以下のいずれかが発生した場合、AWS OpsWorks スタックはすべての組み込みセキュリティグループの再作成を試みます。

- スタックコンソールの AWS OpsWorks スタックの設定ページに変更を加えます。
- スタックのインスタンスの 1 つを起動する。
- 新しいスタックを作成する。

セキュリティグループを指定するには、次のいずれかの方法を使用できます。スタックの作成時に OpsWorks、セキュリティグループの使用設定を使用して設定を指定します。

- はい (デフォルト設定) — AWS OpsWorks スタックは、適切な組み込みセキュリティグループを各レイヤーに自動的に関連付けます。

レイヤーの組み込みセキュリティグループを微調整するには、必要な設定を指定してカスタムセキュリティグループを追加します。ただし、Amazon EC2 で複数のセキュリティグループを評価するときに最も制限が緩いルールが使用されるため、この方法を使用して、組み込みのグループより厳しいルールを指定することはできません。

- いいえ — AWS OpsWorks スタックは組み込みセキュリティグループをレイヤーに関連付けません。

適切なセキュリティグループを作成し、1つ以上のそのグループを作成した各レイヤーと関連付ける必要があります。この方法は、組み込みのグループより厳しいルールを指定する場合に使用します。必要に応じて、組み込みセキュリティグループとレイヤーを手動で関連付けることもできます。カスタムセキュリティグループは、カスタム設定を必要とするレイヤーにのみ必要です。

Important

組み込みのセキュリティグループを使用する場合、グループの設定を手動で変更して、より厳しいルールを作成することはできません。スタックを作成するたびに、AWS OpsWorks スタックは組み込みのセキュリティグループの設定を上書きするため、次回スタックを作成するときに行った変更は失われます。レイヤーで組み込みのセキュリティグループよりも制限の厳しいセキュリティグループ設定が必要な場合は、セキュリティグループを使用 OpsWorks をなしに設定し、任意の設定でカスタムセキュリティグループを作成し、作成時にレイヤーに割り当てます。

AWS OpsWorks Chef 12 Linux のスタックサポート

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、Chef 12 Linux の AWS OpsWorks スタックの概要を説明します。Windows での Chef 12 の詳細については、「[入門ガイド: Windows](#)」を参照してください。Linux での以前の Chef バージョンの詳細については、「[Linux 用 Chef 11.10 以前のバージョン](#)」を参照してください。

概要

AWS OpsWorks スタックは、Linux スタックの Chef の最新バージョンである Chef 12 をサポートしています。詳細については、「[Learn Chef](#)」を参照してください。

AWS OpsWorks スタックは Linux スタックの Chef 11.10 を引き続きサポートします。ただし、さまざまなコミュニティクックブックを活用するか、独自のカスタムクックブックを作成することを希望する高度な Chef ユーザーの場合は、Chef 12 を使用することをお勧めします。Chef 12 スタックには、Linux 用 Chef 11.10 以前のバージョンに対して次のような利点があります。

- 2つの別々の Chef 実行 - インスタンスでコマンドが実行されると、AWS OpsWorks スタック エージェントは2つの独立した Chef 実行を実行するようになりました。1つはインスタンスを AWS Identity and Access Management (IAM) などの他の AWS サービスと統合するタスク用、もう1つはカスタムクックブック用です。最初の Chef 実行では、インスタンスに AWS OpsWorks スタックエージェントをインストールし、ユーザーのセットアップと管理、ボリュームのセットアップと設定、CloudWatch メトリクスの設定などのシステムタスクを実行します。2回目の実行は、[AWS OpsWorks スタックライフサイクルイベント](#) 用のカスタムレシピの実行専用です。この2回目の実行により、独自の Chef クックブックまたはコミュニティクックブックを使用することができます。
- 名前空間の競合の解決 - Chef 12 以前では、AWS OpsWorks スタックがシステムタスクを実行し、共有環境で組み込みのレシピとカスタムレシピを実行していました。これにより、名前空間の競合が発生し、AWS OpsWorks スタックが実行したレシピが明確ではありません。不要なデフォルト設定は、手動で上書きする必要がありました。これは時間がかかり、エラー発生の原因となるタスクです。Linux 用 Chef 12 では、AWS OpsWorks スタックは PHP、Node.js、Rails などのアプリケーションサーバー環境用の組み込み Chef クックブックをサポートしなくなりました。組み込みレシピを排除することで、AWS OpsWorks スタックは組み込みレシピとカスタムレシピ間の命名衝突の問題を排除します。
- Chef コミュニティクックブックの強力なサポート - AWS OpsWorks スタック Chef 12 Linux は、Chef スーパーマーケットのコミュニティクックブックの互換性とサポートを強化します。スタックが AWS OpsWorks 以前に提供した組み込みクックブック、つまり最新のアプリケーションサーバー環境やフレームワークで使用するよう設計されたクックブックよりも優れたコミュニティクックブックを使用できるようになりました。これらのクックブックのほとんどは、Linux 用 Chef 12 を変更することなく実行できます。詳細については、Learn [Chef ウェブサイトの Chef Supermarket](#)、[Chef Supermarket](#) ウェブサイト、およびの [Chef Cookbooks](#) リポジトリを参照してください [GitHub](#)。 <https://supermarket.chef.io/>
- タイムリーな Chef 12 の更新 - AWS OpsWorks スタックは、各 Chef リリースの直後に Chef 環境を最新の Chef 12 バージョンに更新します。Chef 12 では、Chef のマイナーな更新と新しい AWS

OpsWorks Stacks エージェントリリースが一致します。これにより、新しい Chef リリースを直接テストすることができ、Chef レシピおよびアプリケーションで Chef の最新機能を利用できます。

Chef 12 以前にサポートされていた Chef のバージョンの詳細については、「[Linux 用 Chef 11.10 以前のバージョン](#)」を参照してください。

Chef 12 への移行

Chef 12 Linux のキー AWS OpsWorks スタックの変更点を、以前の Chef バージョン 11.10、11.4、および 0.9 のサポートと比較した場合は次のとおりです。

- Linux スタック用 Chef 12 に対して組み込みレイヤーは提供されず、サポートもされません。カスタムレシピのみが実行されるため、このサポートの削除により、インスタンスのセットアップ方法に透明性が与えられ、カスタムクックブックの作成と維持がかなり容易になります。例えば、組み込みの AWS OpsWorks Stacks レシピの属性を上書きする必要はなくなりました。組み込みレイヤーを削除すると、AWS OpsWorks スタックは Chef コミュニティによって開発および保守されているクックブックをより適切にサポートできるため、それらをフルに活用できます。Linux 用 Chef 12 で使用できなくなった組み込みのレイヤータイプは、[AWS Flow \(Ruby\)](#)、[Ganglia](#)、[HAProxy](#)、[Java App Server](#)、[Memcached](#)、[MySQL](#)、[Node.js App Server](#)、[PHP App Server](#)、[Rails App Server](#)、および [Static Web Server](#) です。
- AWS OpsWorks スタックは指定したレシピを実行しているため、カスタムクックブックを実行して組み込みの AWS OpsWorks スタック属性を上書きする必要はなくなりました。独自のレシピまたはコミュニティレシピで属性を無効にするには、Chef 12 ドキュメントの「[属性について](#)」の手順と例に従ってください。
- AWS OpsWorks スタックは、Chef 12 Linux スタックの以下のレイヤーのサポートを引き続き提供します。
 - [カスタムレイヤー](#)
 - [Amazon RDS Service Layer](#)
 - [ECS クラスターレイヤー](#)
- Chef 12 Linux 用のスタック設定とデータバッグは変更され、Chef 12.2 Windows 用と非常によく類似したものになりました。これにより、これらのデータバッグのクエリ、分析、およびトラブルシューティングが簡単になりました (特に、さまざまなオペレーティングシステムタイプでスタックを操作する場合)。AWS OpsWorks スタックは暗号化されたデータバッグをサポートしていないことに注意してください。パスワードや証明書などの機密データを暗号化された形式で保存するには、プライベート S3 バケットに保存することをお勧めします。これで、[Amazon SDK for Ruby](#)

を使用するカスタムレシピを作成できます。例については、[SDK for Ruby の使用](#) を参照してください。詳細については、[AWS OpsWorks スタックデータバググリファレンス](#) を参照してください。

- Chef 12 Linux で、Berkshelf はスタックインスタンスでインストールされなくなりました。代わりに、ローカルの開発用マシンで Berkshelf を使用して、クックブックの依存関係をローカルにパッケージ化することをお勧めします。次に、依存関係を含めて、パッケージを Amazon Simple Storage サービスにアップロードします。最後に、クックブックソースとしてアップロードされたパッケージを使用するように Chef Linux 12 のスタックを変更します。詳細については、「[ローカルでのクックブックの依存関係のパッケージ化](#)」を参照してください。
- EBS ボリュームに対する RAID 設定のサポートは終了しました。パフォーマンスを向上させるには、[\[provisioned IOPS for Amazon Elastic Block Store \(Amazon EBS\)\]](#) (Amazon Elastic Block Store (Amazon EBS) 用にプロビジョニングされた IOPS) を使用することができます。
- autofs のサポートは終了しました。
- サブバージョンリポジトリのサポートは終了しました。
- レイヤーごとの OS パッケージのインストールは、カスタムレシピで完了する必要があります。詳細については、「[レイヤーごとのパッケージのインストール](#)」を参照してください。

サポートされるオペレーティングシステム

Chef 12 は、以前のバージョンの Chef と同じ Linux オペレーティングシステムをサポートします。Chef 12 Linux スタックで使用できる Linux オペレーティングシステムタイプとバージョンのリストについては、「[Linux オペレーティングシステム:](#)」を参照してください。

サポートされるインスタンスタイプ

AWS OpsWorks スタックは、ハイパフォーマンスコンピューティング (HPC) クラスターコンピューティング、クラスター GPU、ハイメモリクラスターインスタンスタイプなどの特殊なインスタンスタイプを除く、Chef 12 Linux スタックのすべてのインスタンスタイプをサポートします。

詳細情報

Linux スタック用 Chef 12 の操作方法の詳細については、以下を参照してください。

- [使用開始: サンプル](#)

Node.js アプリケーション環境を作成するための AWS OpsWorks スタックコンソールでの簡単な実践的な演習を通じて、AWS OpsWorks スタックを紹介します。

- [入門ガイド: Linux](#)

AWS OpsWorks Stacks コンソールでの実践的な演習を通じて、AWS OpsWorks スタックと Chef 12 Linux を紹介し、トラフィックを処理する Node.js アプリケーションを備えたシンプルなレイヤーを含む基本的な Chef 12 Linux スタックを作成します。

- [カスタムレイヤー](#)

Chef 12 Linux スタックのクックブックとレシピを含むレイヤーの追加に関するガイダンスを提供します。Chef コミュニティが提供している、すぐに使用できるクックブックとレシピを使用するか、または独自のクックブックとレシピを作成できます。

- [データバッグへの移行](#)

Chef 11 以前のバージョンを実行している Linux スタックによって使用されているインスタンス JSON を Chef 12 と比較します。Chef 12 のインスタンスの JSON 形式に関する参照資料へのリンクも示します。

属性からデータバッグへのスタック設定の移行

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは、さまざまなスタック設定を Chef レシピに公開します。これらのスタック設定には、次のような値が含まれます。

- スタッククックブックのソース URL
- レイヤーボリューム設定
- インスタンスのホスト名
- Elastic Load Balancing DNS 名
- アプリケーションソースの URL
- ユーザー名

レシピからスタックの設定を参照すると、スタック設定を直接レシピでハードコードするよりも堅牢になり、エラーが起きにくくなります。このトピックでは、これらのスタック設定にアクセスする方法、および Linux 用の Chef 11.10 以前のバージョンの属性から Chef 12 Linux のデータバッグに移行する方法について説明します。

Linux 用の Chef 11.10 以前のバージョンでは、スタック設定は [Chef 属性](#) として利用でき、Chef node オブジェクトまたは Chef 検索によってアクセスします。これらの属性は、`/var/lib/aws/opsworks/chef` ディレクトリ内の一連の JSON ファイルの AWS OpsWorks スタックインスタンスに保存されます。詳細については、「[スタック設定およびデプロイ属性: Linux](#)」を参照してください。

Chef 12 Linux で、スタック設定は、[Chef データバッグ](#) として使用でき、Chef 検索を通じてのみアクセスされます。データバッグは `/var/chef/runs/run-ID/data_bags`、ディレクトリ内の一連の JSON ファイル内の AWS OpsWorks スタックインスタンスに保存されます。`### run-ID` は、インスタンスで実行される各 Chef に AWS OpsWorks スタックが割り当ててる一意の ID です。スタック設定は、現在では Chef 属性として利用できないので、Chef node オブジェクトを通じてスタック設定にアクセスすることはできません。詳細については、[を参照してください AWS OpsWorks スタックデータバッグリファレンス](#)

たとえば、Linux 用の Chef 11.10 以前のバージョンでは、次のレシピコードで Chef node オブジェクトを使用して、アプリケーションの短縮名およびソース URL を表す属性を取得します。次に、Chef ログを使用してこれら 2 つの属性値を書き込みます。

```
Chef::Log.info("***** The app's short name is '#{node['opsworks']
['applications'].first['slug_name']}' *****")
Chef::Log.info("***** The app's URL is '#{node['deploy']['simplephpapp']['scm']
['repository']}' *****")
```

Chef 12 Linux では、次のレシピコードで `aws_opsworks_app` 検索インデックスを使用して、`aws_opsworks_app` データバッグの最初のデータバッグ項目のコンテンツを取得します。次に、コードは、Chef ログに 2 つのメッセージを書き込みます。1 つはアプリケーションの短縮名のデータバッグコンテンツで、もう 1 つはアプリケーションのソース URL データバッグのコンテンツです。

```
app = search("aws_opsworks_app").first

Chef::Log.info("***** The app's short name is '#{app['shortname']}' *****")
Chef::Log.info("***** The app's URL is '#{app['app_source']['url']}' *****")
```

Linux 用 Chef 11.10 以前のバージョンからスタック設定にアクセスするレシピコードを Chef 12 Linux に移行するには、次のようにコードを変更する必要があります。

- Chef 属性の代わりに Chef データバッグにアクセスします。
- Chef node オブジェクトの代わりに Chef 検索を使用します。
- opsworks や などの AWS OpsWorks スタック属性名を使用する代わりにaws_opsworks_app、などの AWS OpsWorks スタックデータバッグ名を使用しますdeploy。

詳細については、「[AWS OpsWorks スタックデータバッグリファレンス](#)」を参照してください。

スタックでの以前の Chef AWS OpsWorks バージョンのサポート

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、以前の Chef AWS OpsWorks バージョンの スタックドキュメントの概要を説明します。

[Linux 用 Chef 11.10 以前のバージョン](#)

Linux AWS OpsWorks スタックの Chef 11.10、11.4、および 0.9 の スタックサポートに関するドキュメントを提供します。

Linux 用 Chef 11.10 以前のバージョン

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、Linux AWS OpsWorks 用 Chef 11.10、11.4、および 0.9 のスタックドキュメントの概要を説明します。

[Chef 11 Linux スタックの使用開始](#)

シンプルで機能的なPHPアプリケーションサーバースタックを作成する方法を示すチュートリアルを提供します。

[Node.js スタックの初回作成](#)

Node.js アプリケーションサーバーをサポートする Linux スタックの作成方法と、シンプルなアプリケーションのデプロイ方法について説明します。

[AWS OpsWorks スタックのカスタマイズ](#)

特定の要件を満たすように AWS OpsWorks スタックをカスタマイズする方法について説明します。

[クックブック 101](#)

AWS OpsWorks スタックインスタンスのレシピを実装する方法について説明します。

[レイヤーの負荷分散](#)

使用可能な AWS OpsWorks スタックの負荷分散オプションを使用する方法について説明します。

[でのスタックの実行 VPC](#)

仮想プライベートクラウドでスタックを作成および実行する方法について説明します。

[Chef Server からの移行](#)

Chef Server から AWS OpsWorks スタックへの移行に関するガイドラインを提供します。

[AWS OpsWorks スタックレイヤーリファレンス](#)

使用可能な AWS OpsWorks スタックの組み込みレイヤーについて説明します。

[クックブックのコンポーネント](#)

クックブックの 3 つの標準コンポーネント (属性、テンプレート、およびレシピ) について説明します。

[スタック設定およびデプロイ属性: Linux](#)

Linux のスタック設定およびデプロイ属性について説明します。

[組み込みクックブックの属性](#)

組み込みのレシピの属性を使用して、インストールしたソフトウェアの設定を制御する方法について説明します。

[Linux 用 Chef 11.10 以前のバージョンのトラブルシューティング](#)

AWS OpsWorks スタックのさまざまな問題をトラブルシューティングする方法について説明します。

Chef 11 Linux スタックの使用開始

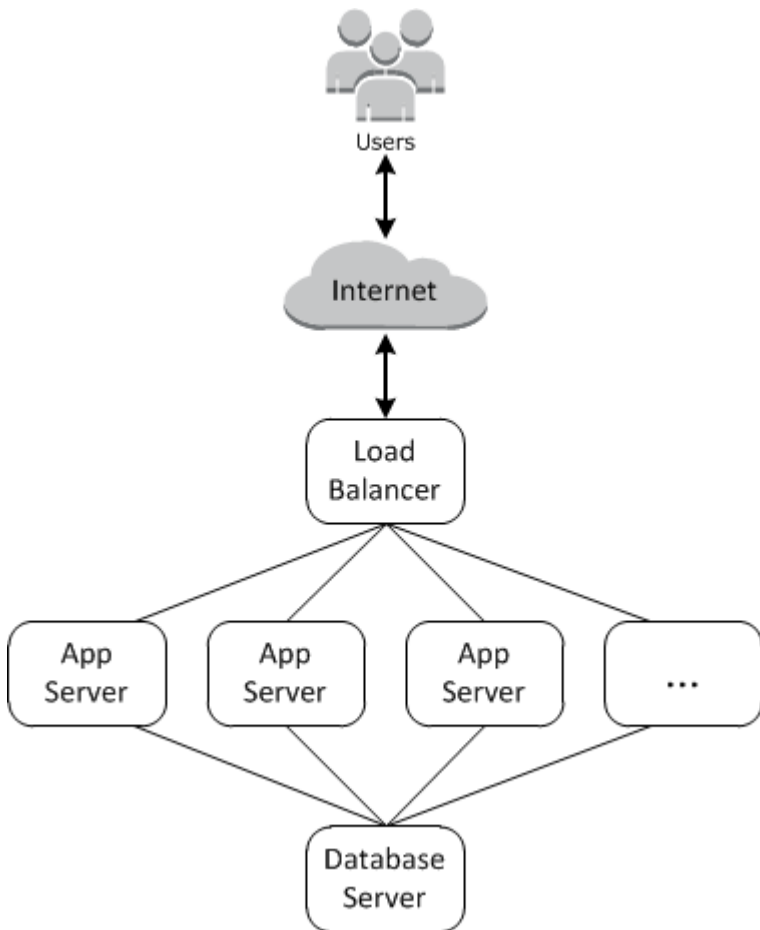
Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このセクションでは、Chef 11 を使用して Linux スタックの使用を開始する方法について説明します。Chef 12 Linux スタックの使用開始の詳細については、「[入門ガイド: Linux](#)」を参照してください。Chef 12 Windows スタックの使用開始の詳細については、「[入門ガイド: Windows](#)」を参照してください。

通常、クラウドベースのアプリケーションでは、アプリケーションサーバー、データベースサーバーなどの関連リソースのグループが必要で、一括して作成および管理する必要があります。この一連のインスタンスはスタックと呼ばれます。シンプルなアプリケーションスタックの例を以下に示します。



基本的なアーキテクチャは以下の要素で構成されます。

- ユーザーからの着信トラフィックをアプリケーションサーバーに均等に分散するロードバランサー。
- トラフィックを処理するために必要な数の一連のアプリケーションサーバーインスタンス。
- アプリケーションサーバーにバックエンドデータストアを提供するデータベースサーバー。

また、通常、アプリケーションサーバーにアプリケーションを配布する方法や、スタックを監視する方法も必要です。

AWS OpsWorks スタックは、スタックおよび関連するアプリケーションとリソースを作成および管理するためのシンプルで簡単な方法を提供します。この章では、AWS OpsWorks スタックの基本とさらに高度な機能を紹介し、図表でアプリケーションサーバースタックを作成するプロセスを順を追って説明します。AWS OpsWorks スタックが簡単に従える増分開発モデルを使用します。基本スタックを設定し、正しく動作したら、フル機能の実装に到達するまでコンポーネントを追加します。

- 「[ステップ 1: 前提条件を完了する](#)」に、ウォークスルーを開始するためのセットアップ方法を示します。
- 「[ステップ 2: 簡単なアプリケーションサーバースタック - Chef 11 を作成する](#)」では、単一のアプリケーションサーバーから成る最小限のスタックを作成する方法を示します。
- 「[ステップ 3: バックエンドデータストアを追加する](#)」では、データベースサーバーを追加し、データベースサーバーをアプリケーションサーバーに接続する方法を示します。
- 「[ステップ 4: スケールアウトする MyStack](#)」では、さらに多くのアプリケーションサーバーと、着信トラフィックを分散させるためのロードバランサーを追加することによって、増大した負荷を処理するためにスタックをスケールアウトする方法を示します。

トピック

- [ステップ 1: 前提条件を完了する](#)
- [ステップ 2: 簡単なアプリケーションサーバースタック - Chef 11 を作成する](#)
- [ステップ 3: バックエンドデータストアを追加する](#)
- [ステップ 4: スケールアウトする MyStack](#)
- [ステップ 5: 削除 MyStack](#)

ステップ 1: 前提条件を完了する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

ウォークスルーを開始する前に、次のセットアップ手順を完了します。これらのセットアップ手順には、AWS アカウントへのサインアップ、管理ユーザーの作成、AWS OpsWorks スタックへのアクセス許可の割り当てが含まれます。

すでにいずれかの「[AWS OpsWorks スタックの開始方法](#)」ウォークスルーを完了している場合、このウォークスルーの前提条件を満たしており、省略して「[ステップ 2: 簡単なアプリケーションサーバースタック - Chef 11 を作成する](#)」に進むことができます。

トピック

- [にサインアップする AWS アカウント](#)
- [管理アクセスを持つユーザーを作成する](#)
- [ユーザーにサービスアクセス権限を割り当てる](#)

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. <https://portal.aws.amazon.com/billing/サインアップ>を開きます。
2. オンラインの手順に従います。

サインアップ手順の一環として、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

にサインアップすると AWS アカウント、AWS アカウントのルートユーザー が作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用して[ルートユーザーアクセスが必要なタスク](#)を実行してください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。 / に移動し、マイアカウントを選択すると、いつでも現在のアカウントアクティビティを表示<https://aws.amazon.com>し、アカウントを管理できます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、日常的なタスクにルートユーザーを使用しないように、のセキュリティを確保し AWS IAM Identity Center、AWS アカウントのルートユーザーを有効にして、管理ユーザーを作成します。

を保護する AWS アカウントのルートユーザー

1. ルートユーザーを選択し、AWS アカウント E メールアドレスを入力して、アカウント所有者[AWS Management Console](#)として にサインインします。次のページでパスワードを入力します。

ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの[ルートユーザーとしてサインインする](#)を参照してください。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

手順については、「[ユーザーガイド](#)」の AWS アカウント「[ルートユーザーの仮想MFAデバイスを有効にする \(コンソール\)](#)」を参照してください。IAM

管理アクセスを持つユーザーを作成する

1. IAM Identity Center を有効にします。

手順については、「AWS IAM Identity Center ユーザーガイド」の「[AWS IAM Identity Centerの有効化](#)」を参照してください。

2. IAM Identity Center で、ユーザーに管理アクセス権を付与します。

を ID ソース IAM アイデンティティセンターディレクトリとして使用する方法的チュートリアルについては、「AWS IAM Identity Center ユーザーガイド」の「[デフォルトを使用してユーザーアクセスを設定する IAM アイデンティティセンターディレクトリ](#)」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM Identity Center ユーザーでサインインするには、IAM Identity Center ユーザーの作成時に E メールアドレスに URL 送信されたサインインを使用します。

IAM Identity Center ユーザーを使用してサインインする方法については、「AWS サインイン ユーザーガイド」の [AWS 「アクセスポータルにサインインする」](#) を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM Identity Center で、最小特権のアクセス許可を適用するベストプラクティスに従うアクセス許可セットを作成します。

手順については、「AWS IAM Identity Center ユーザーガイド」の「[権限設定を作成する](#)」を参照してください。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

手順については、「AWS IAM Identity Center ユーザーガイド」の「[グループの結合](#)」を参照してください。

ユーザーにサービスアクセス権限を割り当てる

ロールまたはユーザーに AWS OpsWorks AWSOpsWorks_FullAccess および アクセス AmazonS3FullAccess 許可を追加することで、スタックサービス (および AWS OpsWorks スタックが依存する関連サービス) へのアクセスを有効にします。

アクセス許可の追加の詳細については、[IAM「ID アクセス許可の追加 \(コンソール\)」](#)を参照してください。

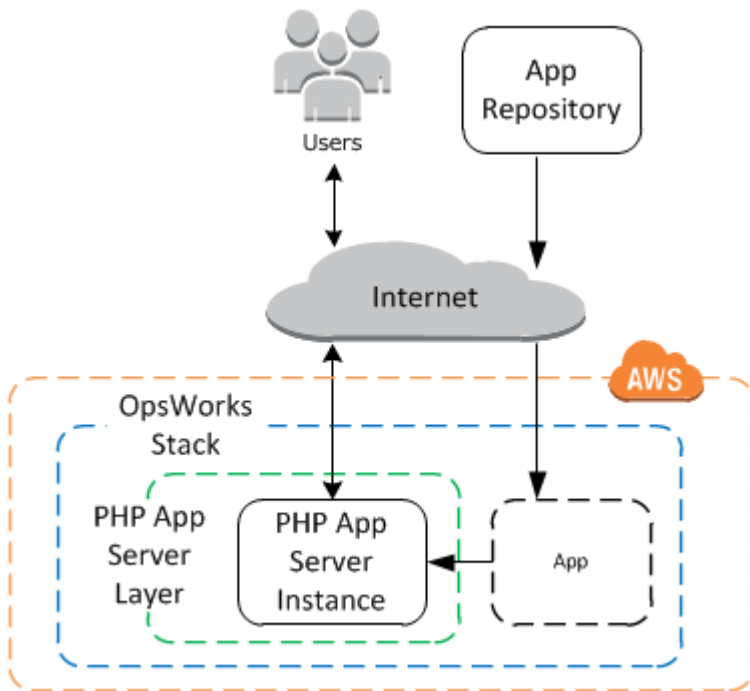
これですべてのセットアップステップが完了したので、[このウォークスルーを開始](#)できます。

ステップ 2: 簡単なアプリケーションサーバースタック - Chef 11 を作成する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

基本的なアプリケーションサーバースタックは、ユーザーのリクエストを受信するためのパブリック IP アドレスを持つ、単一のアプリケーションサーバーインスタンスで構成されます。アプリケーションコードと関連ファイルは別々のリポジトリに保存されており、そこからサーバーにデプロイされます。そのようなスタックを次の図に示します。



スタックには次のようなコンポーネントがあります。

- レイヤー。インスタンスのグループを表し、それらの設定方法を指定します。

この例のレイヤーは、PHPApp Server インスタンスのグループを表します。

- インスタンス。Amazon EC2インスタンスを表します。

この場合、インスタンスはPHPアプリケーションサーバーを実行するように設定されます。レイヤーには、任意の数のインスタンスを含めることができます。AWS OpsWorks スタックは、他のいくつかのアプリケーションサーバーもサポートしています。詳細については、「[アプリケーションサーバーレイヤー](#)」を参照してください。

- アプリケーションサーバーにアプリケーションをインストールするために必要な情報を含む、アプリケーション。

コードは、Git リポジトリや Amazon S3 バケットなどのリモートリポジトリに保存されます。

以下のセクションでは、AWS OpsWorks スタックコンソールを使用してスタックを作成し、アプリケーションをデプロイする方法について説明します。テンプレートを使用してスタックを AWS CloudFormation プロビジョニングすることもできます。このトピックで説明するスタックをプロビジョニングするテンプレートの例については、[AWS OpsWorks 「スニペット」](#)を参照してください。

トピック

- [ステップ 2.1: スタック - Chef 11 を作成する](#)
- [ステップ 2.2: PHPアプリケーションサーバーレイヤーを追加する - Chef 11](#)
- [ステップ 2.3: PHP App Server Layer にインスタンスを追加する - Chef 11](#)
- [ステップ 2.4: アプリケーション - Chef 11 を作成してデプロイする](#)

ステップ 2.1: スタック - Chef 11 を作成する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックプロジェクトを開始するには、インスタンスやその他のリソースのコンテナとして機能するスタックを作成します。スタック設定では、AWSリージョンやデフォルトのオペレーティングシステムなど、スタックのすべてのインスタンスで共有される基本的な設定を指定します。

Note

このページは、Chef 11スタックの作成に役立ちます。Chef 12 スタックを作成する方法については、「[スタックを作成する](#)」を参照してください。

このページは、Chef 11 でのスタックの作成に役立ちます。

新しいスタックを作成するには

1. スタックの追加

[AWS OpsWorks スタックコンソール](#) にサインインします。アカウントに既存のスタックがない場合は、「ようこそAWS OpsWorks」ページが表示され、最初のスタックを追加をクリックします。それ以外の場合は、AWS OpsWorks スタックダッシュボードが表示され、アカウントのスタックが一覧表示されます。スタックの追加をクリックします。



2. スタックの設定

[Add Stack] ページで、[Chef 11 stack] を選択し、次の設定を指定します。

スタックの名前

スタックの名前を入力します。英数字 (a~z、A~Z および 0~9) とハイフン (-) を含めることができます。このワークスルーで使用する例のスタック名は、**MyStack** です。

リージョン

スタックのリージョンとして米国西部 (オレゴン) を選択します。

その他の設定についてはデフォルト値を受け入れて、[Add Stack] をクリックします。さまざまなスタック設定の詳細については、「[新しいスタックを作成する](#)」を参照してください。

ステップ 2.2: PHPアプリケーションサーバーレイヤーを追加する - Chef 11

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

スタックは、基本的にはインスタンスのコンテナですが、インスタンスを直接スタックに追加することはありません。関連するインスタンスのグループを表すレイヤーを追加し、そのレイヤーにインスタンスを追加します。

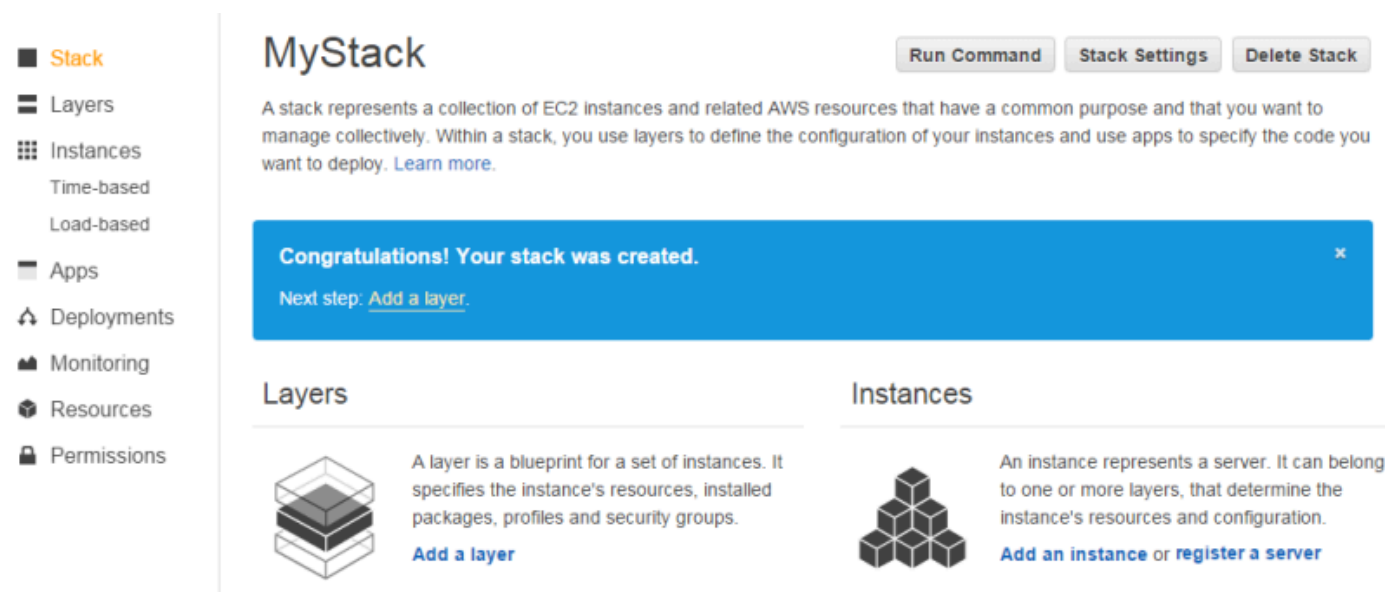
レイヤーは基本的に、同じ設定の Amazon EC2 インスタンスのセットを作成するために AWS OpsWorks スタックが使用する設計図です。関連するインスタンスのグループごとに 1 つの Layer をスタックに追加します。AWS OpsWorks スタックには、MySQL データベースサーバーや PHP アプリケーションサーバーなどの標準ソフトウェアパッケージを実行しているインスタンスのグループを表す一連の組み込みレイヤーが含まれています。さらに、特定の要件に合わせて一部または全体をカスタマイズしたレイヤーを作成することもできます。詳細については、「[AWS OpsWorks スタックのカスタマイズ](#)」を参照してください。

MyStack には、アプリケーションサーバーとして機能するインスタンスのグループを表す組み込み PHP の App PHP Server レイヤーである 1 つのレイヤーがあります。組み込みレイヤーなどの詳細については、「[レイヤー](#)」を参照してください。

App PHP Server レイヤーを に追加するには MyStack

1. [Add Layer] (レイヤーの追加) ページを開く

スタックの作成が完了すると、AWS OpsWorks スタックにスタックページが表示されます。[\[Add a layer\]](#) (レイヤーの追加) をクリックして、最初のレイヤーを追加します。



The screenshot shows the AWS OpsWorks console interface for a stack named "MyStack". On the left is a navigation sidebar with options like Stack, Layers, Instances, Apps, Deployments, Monitoring, Resources, and Permissions. The main content area has a header with "MyStack" and buttons for "Run Command", "Stack Settings", and "Delete Stack". Below the header is a blue notification box stating "Congratulations! Your stack was created." and "Next step: [Add a layer.](#)". The main content is divided into two sections: "Layers" and "Instances". The "Layers" section includes a 3D cube icon and text explaining that a layer is a blueprint for a set of instances, with a link to "Add a layer". The "Instances" section includes a 3D cube icon and text explaining that an instance represents a server, with a link to "Add an instance or register a server".

2. レイヤータイプの指定とレイヤーの設定

Layer type ボックスで、PHPApp Server を選択し、デフォルトの Elastic Load Balancer 設定を受け入れ、Add Layer をクリックします。レイヤーを作成したら、[レイヤーを編集](#)してEBSボリューム設定などの他の属性を指定できます。

Add layer

OpsWorks ECS RDS

Layer type

The PHP Application Server layer is a blueprint for instances that function as PHP application servers. The supported versions depend on the operating system. [Learn more.](#)

Elastic Load Balancer

Need further support? [Let us know.](#)

Cancel **Add layer**

ステップ 2.3: PHP App Server Layer にインスタンスを追加する - Chef 11

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

スタックインスタンスは、特定の Amazon AWS OpsWorks EC2 インスタンスを表します。

- インスタンスの設定では、Amazon EC2 operating のシステムやサイズなどのいくつかの基本が指定されています。インスタンスは実行されますが、あまり実行されません。
- インスタンスのレイヤーでは、インストールするパッケージを決定したり、インスタンスに Elastic IP アドレスがあるかどうかを指定したりすることで、インスタンスに機能を追加します。

AWS OpsWorks スタックは、サービスとやり取りする各インスタンスにエージェントをインストールします。レイヤーの機能をインスタンスに追加するために、AWS OpsWorks スタックは [Chef レシピ](#) と呼ばれる小さなアプリケーションを実行するようにエージェントに指示します。これにより、アプリケーションやパッケージのインストール、設定ファイルの作成などが可能になります。AWS OpsWorks スタックは、インスタンスの [ライフサイクル](#) の重要なポイントでレシピを実行します。例えば、 はインスタンスの起動後に Setup レシピ OpsWorks を実行してソフトウェアのインストールなどのタスクを処理し、 はアプリケーションをデプロイしてコードと関連ファイルをインストールするときに Deploy レシピを実行します。

Note

レシピの動作に関心がある場合は、すべての AWS OpsWorks スタックの組み込みレシピがパブリック GitHub リポジトリの [OpsWorks クックブック](#) にあります。また、後述するように、独自のカスタムレシピを作成して AWS OpsWorks スタックで実行することもできます。

PHP アプリケーションサーバーを に追加するには MyStack、前のステップで作成した PHP App Server レイヤーにインスタンスを追加します。

インスタンスを PHP App Server レイヤーに追加するには

1. インスタンスの追加

レイヤーの追加が完了すると、AWS OpsWorks スタックにレイヤーページが表示されます。ナビゲーションペインでインスタンスをクリックし、PHPApp Server でインスタンスの追加をクリックします。

2. インスタンスの設定

各インスタンスには、AWS OpsWorks スタックによって生成されるデフォルトのホスト名があります。この例では、AWS OpsWorks スタックはレイヤーの短縮名に数値を追加するだけです。アベイラビリティゾーンやオペレーティングシステムといった、スタックの作成時に指定したデフォルト設定の一部を上書きするなどして、各インスタンスを個別に設定できます。このウォークスルーでは、デフォルト設定を受け入れて [Add Instance] をクリックし、レイヤーにインスタンスを追加します。詳細については、「[インスタンス](#)」を参照してください。

PHP App Server

No instances. Add an instance.

New Existing OpsWorks EC2 instances and own servers

Hostname

Size

Subnet

[Advanced »](#)

3. インスタンスの作成

ここまでで、インスタンスの設定を指定しました。実行中の Amazon インスタンスを作成するには、EC2インスタンスを起動する必要があります。その後、AWS OpsWorks スタックは設定を使用して、指定されたアベイラビリティゾーンで Amazon EC2インスタンスを起動します。インスタンスの起動方法の詳細は、インスタンスのスケールリングタイプによって異なります。前のステップでは、デフォルトのスケールリングタイプである 24/7 (手動で起動する必要があり、手動で停止するまで実行される) を使用してインスタンスを作成しました。時間ベースおよび負荷ベースのスケールリングタイプを作成することもできます。このスケールリングタイプは、スケジュールまたは現在の負荷に基づいて AWS OpsWorks スタックが自動的に開始および停止します。詳細については、「[時間ベースおよび負荷ベースのインスタンスによる負荷の管理](#)」を参照してください。

App PHP Server の php-app1 に移動し、行の Actions 列で start をクリックしてインスタンスを起動します。

PHP App Server

Hostname	Status	Size	Type	AZ	Public IP	Actions
php-app1	stopped	c3.large	24/7	us-west-2a	-	start delete

[+ Instance](#)

4. 起動中のインスタンスのステータスの監視

通常、Amazon EC2インスタンスの起動とパッケージのインストールには数分かかります。起動処理が進行するに従って、インスタンスの [Status] フィールドに次のような一連の値が表示されます。

1. リクエスト済み - AWS OpsWorks スタックが Amazon EC2サービスを呼び出して Amazon EC2インスタンスを作成しました。
2. 保留中 - AWS OpsWorks スタックは Amazon EC2インスタンスの起動を待っています。
3. booting - Amazon EC2インスタンスが起動しています。
4. running_setup - AWS OpsWorks スタックエージェントは、パッケージの設定やインストールなどのタスクを処理するレイヤーの Setup レシピと、インスタンスにアプリケーションをデプロイする Deploy レシピを実行しています。
5. online - インスタンスは利用可能です。

php-app1 がオンライン状態になったら、[Instances] ページには次のように表示されます。

PHP App Server

Hostname	Status	Size	Type	AZ	Public IP	Actions
php-app1	online	c3.large	24/7	us-west-2a	192.0.2.1	stop ssh

+ Instance

ページの先頭には、スタックのすべてのインスタンスの簡単な要約が表示されます。ここでは、オンライン状態のインスタンスが 1 つ表示されています。php-app1 の [Actions] (アクション) 列で、[start] (開始) と [delete] (削除) が置き換えられ、インスタンスを停止する [stop] (停止) であることに注目してください。

ステップ 2.4: アプリケーション - Chef 11 を作成してデプロイする

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

MyStack より役立つように、App Server インスタンスにPHPアプリケーションをデプロイする必要があります。アプリケーションのコードや関連ファイルは、Git などのリポジトリに保存します。これらのファイルをアプリケーションサーバーに取得するには、いくつかの手順を実行する必要があります。

Note

このセクションの手順は、Chef 11 スタックに適用されます。Chef 12 スタックのレイヤーにアプリを追加する方法については、「[アプリケーションの追加](#)」を参照してください。

1. アプリケーションを作成します。

アプリケーションには、リポジトリからコードと関連ファイルをダウンロードするために AWS OpsWorks スタックが必要とする情報が含まれています。アプリケーションのドメインなどの追加情報を指定することもできます。

2. アプリケーションサーバーにアプリケーションをデプロイします。

アプリケーションをデプロイすると、AWS OpsWorks スタックはデプロイライフサイクルイベントをトリガーします。次に、エージェントによってインスタンスの Deploy レシピが実行されます。このレシピによって、サーバーの設定やサービスの再起動などの関連タスクとともにファイルが適切なディレクトリにダウンロードされます。

Note

新しいインスタンスを作成すると、AWS OpsWorks スタックによって既存のアプリケーションが自動的にインスタンスにデプロイされます。ただし、新しいアプリケーションを作成したり、既存のアプリケーションを更新したりする場合は、既存のすべてのインスタンスに対して、アプリケーションまたは更新を手動でデプロイする必要があります。

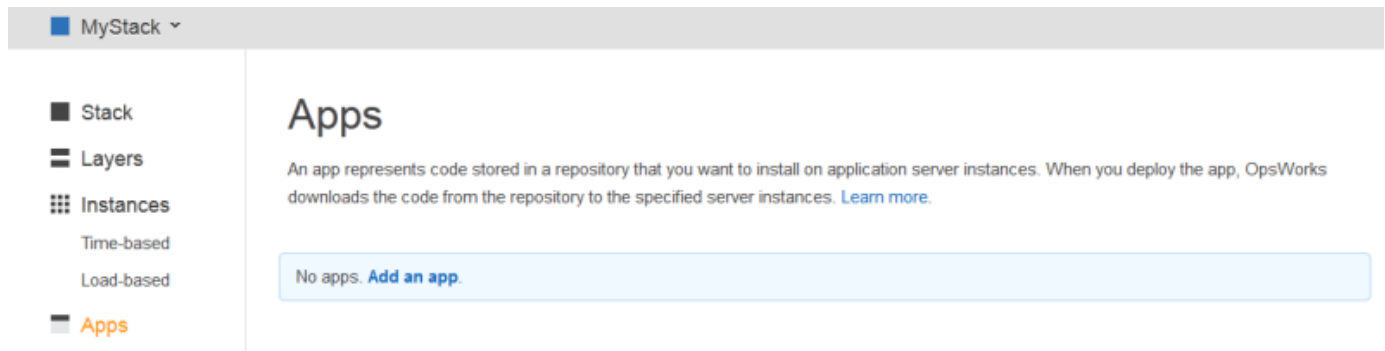
このステップでは、Git のパブリックリポジトリからアプリケーションサーバーに、サンプルアプリケーションを手動でデプロイする方法を示します。アプリケーションを調べる場合は、<https://>

github.com/amazonwebservices/opsworks-demo-php-simple 「-app」を参照してください。この例で使用されているアプリケーションは version1 ブランチにあります。AWS OpsWorks スタックは、他のいくつかのリポジトリタイプもサポートしています。詳細については、「[Application Source](#)」を参照してください。

アプリケーションを作成してデプロイするには

1. [Apps] ページを開く

ナビゲーションペインで [Apps] をクリックし、[Apps] ページで [Add an app] をクリックします。



2. アプリケーションの設定

[App] ページで、次の値を指定します。

名前

AWS OpsWorks スタックが表示目的で使用するアプリケーションの名前。サンプルアプリケーションの名前は **SimplePHPApp** です。後で説明するように、AWS OpsWorks スタックは、内部的および Deploy レシピによって使用される短縮名 `simplephpapp` も生成します。

Type

アプリケーションのデプロイ先を決定する、アプリケーションのタイプです。この例では PHP、を使用して、App Server インスタンスに PHP アプリケーションをデプロイします。

データソースタイプ

関連付けられるデータベースサーバーです。今回は [None] を選択します。データベースサーバーについては、「[ステップ 3: バックエンドデータストアを追加する](#)」で説明します。

リポジトリタイプ

アプリケーションのリポジトリタイプです。この例では、アプリケーションは [Git] リポジトリに保存されています。

リポジトリ URL

アプリケーションのリポジトリ URL。例URLは次のとおりです。 **git://github.com/awslabs/opsworks-demo-php-simple-app.git**

[Branch/Revision]

アプリケーションのブランチまたはバージョンです。このワークスルーでは、**version1** ブランチを使用します。

残りの設定ではデフォルト値を受け入れ、[Add App] をクリックします。詳細については、「[アプリケーションの追加](#)」を参照してください。

Add App

Settings

Name	<input type="text" value="SimplePHPApp"/>
Type	<input type="text" value="PHP"/>
Document root	<input type="text" value="Optional"/>

Data Sources

Data source type RDS OpsWorks None

Application Source

Repository type	<input type="text" value="Git"/>
Repository URL	<input type="text" value="git://github.com/amazonwebservices/oj"/>
Repository SSH key	<input type="text" value="Optional"/>
Branch/Revision	<input type="text" value="version1"/>

3. [Deployment] ページを開く

サーバーにコードをインストールするには、アプリケーションをデプロイする必要があります。これを行うには、SimplePHPApp Actions 列のデプロイをクリックします。

Apps

An app represents code stored in a repository that you want to install on application server instances. When you deploy the app, OpsWorks downloads the code from the repository to the specified server instances. [Learn more.](#)

Name	Type	Data Source	Last Deployment	Actions
SimplePHPApp	PHP			deploy edit delete
+ App				

4. アプリケーションのデプロイ

アプリケーションをデプロイすると、エージェントはPHPアプリケーションサーバーインスタンスで Deploy レシピを実行し、アプリケーションをダウンロードして設定します。

あらかじめ [Command] が [deploy] に設定されている必要があります。その他の設定ではデフォルト値を受け入れ、[Deploy] をクリックし、アプリケーションをデプロイします。

Deploy app

Settings

App	SimplePHPApp
Command	deploy
Comment	Optional

[Advanced »](#)

Instances ⓘ

OpsWorks will run this command on **1 of 1** instances. The assigned recipes are run on all selected instances.

- PHP App Server** php-app1 (online)
Click to select all instances in this layer

Cancel **Deploy**

デプロイメントが完了すると、[Deployment] (デプロイ) ページの [Status] (ステータス) に [Successful] (成功) と表示され、[php-app1] (php-app1) の横に緑色のチェックマークが付きま

す。

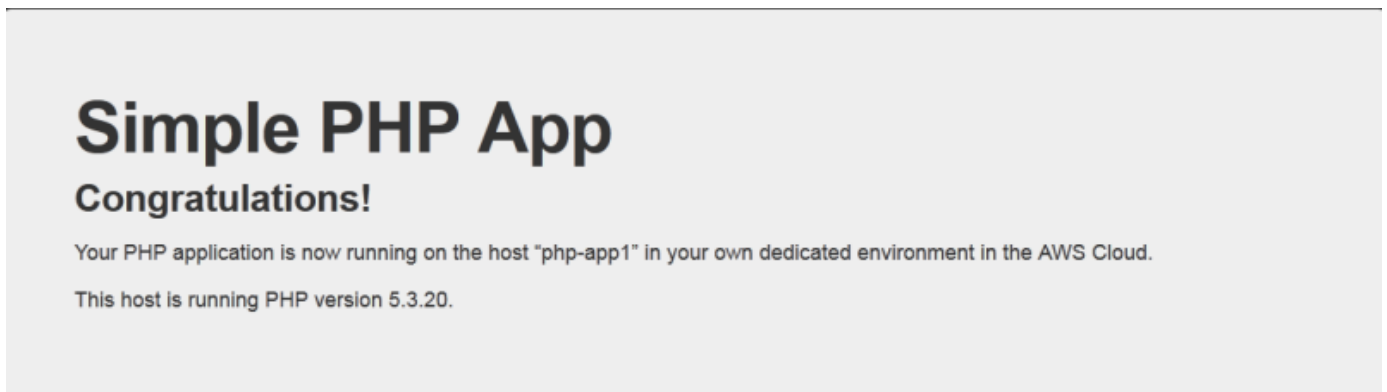
5. S を実行するimplePHPApp

これで S がインストールされ、準備implePHPApp が整いました。実行するには、ナビゲーションペインで [Instances] をクリックし、[Instances] ページに移動します。次に、php-app1 インスタンスのパブリック IP アドレスをクリックします。



Hostname	Status	Size	Type	AZ	Public IP	Actions
php-app1	online	c3.large	24/7	us-west-2a	192.0.2.1	stop ssh

ブラウザに次のようなページが表示されます。



Simple PHP App

Congratulations!

Your PHP application is now running on the host "php-app1" in your own dedicated environment in the AWS Cloud.

This host is running PHP version 5.3.20.

Note

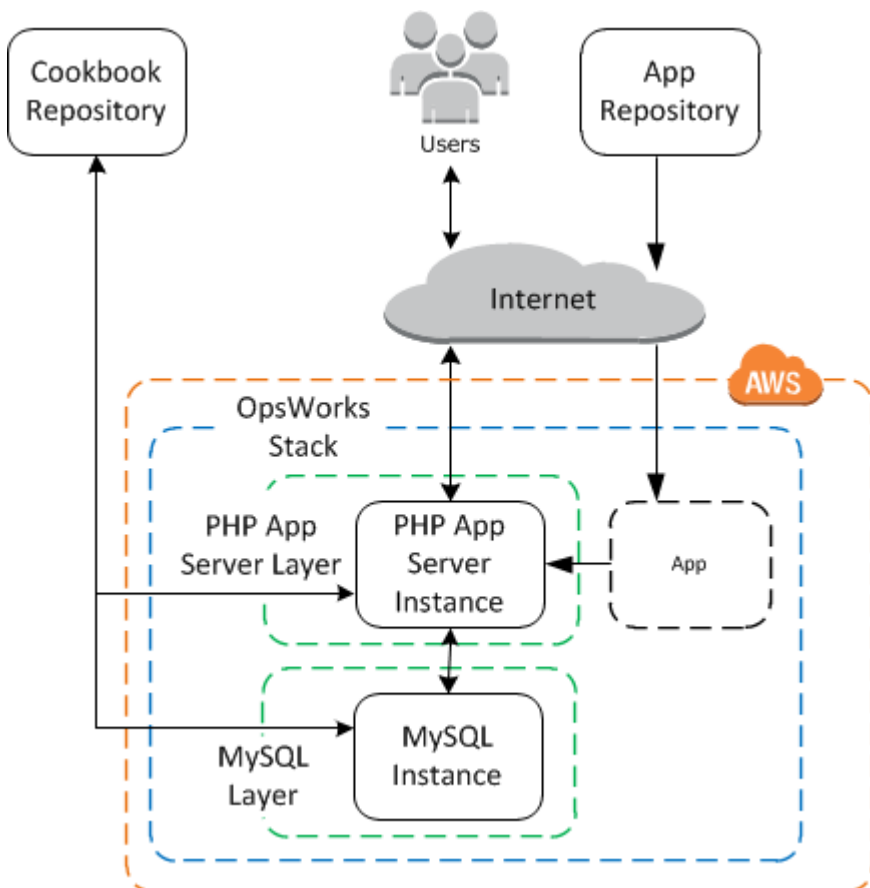
このウォークスルーでは、次のセクションに進んで1回のセッションでウォークスルーをすべて完了することを想定しています。必要に応じて、いつでも停止し、後で AWS OpsWorks スタックにサインインしてスタックを開くことで続行できます。ただし、オンラインインスタンスなど、使用したAWSリソースに対しては課金されます。不要な料金が発生しないように、インスタンスを停止すると、対応するEC2インスタンスが終了します。続行する準備が整ったら、インスタンスを再起動します。

ステップ 3: バックエンドデータストアを追加する

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

[ステップ 2.1: スタック - Chef 11 を作成する](#) では、PHPアプリケーションを提供するスタックを作成する方法を示しました。ただし、これは、静的なテキストを表示するだけの非常にシンプルなアプリケーションでした。本稼動アプリケーションでは、一般的にバックエンドデータストアを使用して、次の図のようなスタック設定を実現します。



このセクションでは、を拡張 MyStack してバックエンドの MySQL データベースサーバーを含める方法を示します。ただし、スタックに MySQL サーバーを追加する以上のことを行う必要があります。また、データベースサーバーと正しく通信するようにアプリを設定する必要があります。AWS

OpsWorks スタックではこれを行うことはありません。このタスクを処理するには、いくつかのカスタムレシピを実装する必要があります。

トピック

- [ステップ 3.1: バックエンドデータベースを追加する](#)
- [ステップ 3.2: S を更新するimplePHPApp](#)
- [短いディグレッション: クックブック、レシピ、AWS OpsWorks スタック属性](#)
- [ステップ 3.3: カスタムクックブックを に追加する MyStack](#)
- [ステップ 3.4: レシピを実行する](#)
- [ステップ 3.5: SimplePHPApp、バージョン 2 をデプロイする](#)
- [ステップ 3.6: S を実行するimplePHPApp](#)

ステップ 3.1: バックエンドデータベースを追加する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

新しいバージョンの S は、バックエンドデータベースにデータimplePHPApp を保存します。AWS OpsWorks スタックは、次の 2 種類のデータベースサーバーをサポートしています。

- [My SQL AWS OpsWorks スタックレイヤー](#) は、MySQL データベースマスターをホストする Amazon EC2 インスタンスを作成するための設計図です。
- Amazon RDS サービスレイヤーは、[Amazon RDS インスタンス](#) をスタックに組み込む方法を提供します。

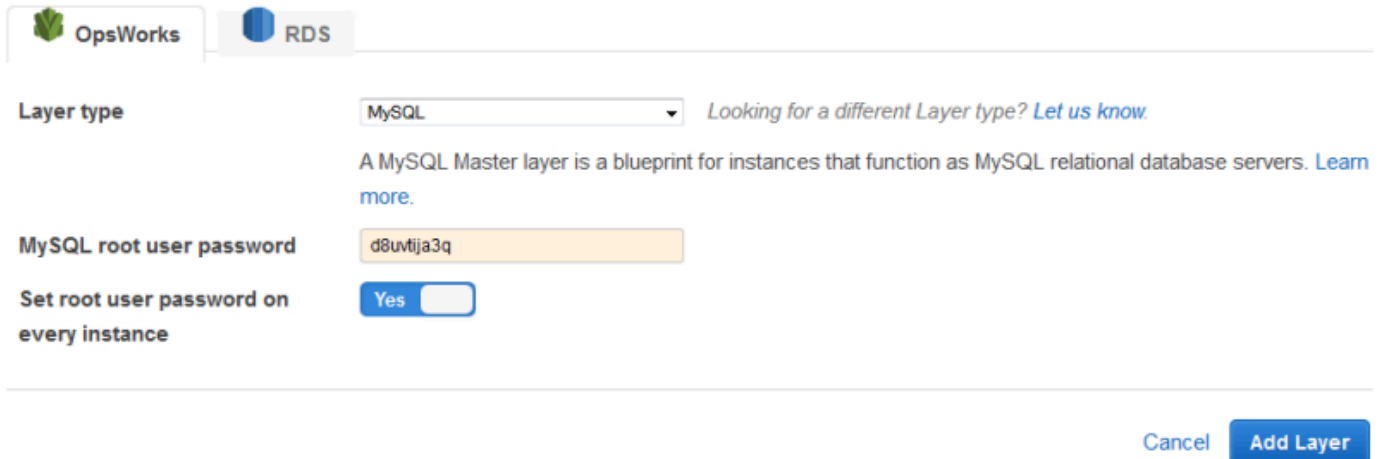
Amazon DynamoDB などの他のデータベースを使用することも、[\[MongoDB\]](#) などのデータベースをサポートするカスタムレイヤーを作成することもできます。詳細については、「[the section called “バックエンドデータストアの使用”](#)」を参照してください。

この例では、MySQL レイヤーを使用します。

My レイヤーをSQL に追加するには MyStack

1. [Layers] (レイヤー) ページで、[+ Layer] (+ レイヤー) をクリックします。
2. レイヤーの追加ページのレイヤータイプで、MySQLを選択し、デフォルト設定を受け入れて、レイヤーの追加をクリックします。

Add Layer



OpsWorks RDS

Layer type [Looking for a different Layer type? Let us know.](#)

A MySQL Master layer is a blueprint for instances that function as MySQL relational database servers. [Learn more.](#)

MySQL root user password

Set root user password on every instance Yes

[Cancel](#) [Add Layer](#)

インスタンスをMySQLレイヤーに追加するには

1. レイヤーページのMySQL行で、インスタンスの追加をクリックします。
2. 「インスタンス」ページの「MySQL」で、「インスタンスの追加」をクリックします。
3. デフォルト値を変更せずに [Add instance] をクリックしますが、まだ起動しないでください。

Note

AWS OpsWorks スタックは、この例のアプリケーションの短縮名 simplephpapp を使用して、という名前のデータベースを自動的に作成します。[Chef レシピ](#)を使用してデータベースと対話する場合、この名前が必要になります。

ステップ 3.2: S を更新するimplePHPApp

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

開始するには、バックエンドデータストアimplePHPApp を使用する新しいバージョンの S が必要です。AWS OpsWorks スタックでは、アプリケーションを更新することは簡単です。Git または Subversion リポジトリを使用している場合は、アプリケーションのバージョンごとに個別のリポジトリブランチを設定できます。サンプルアプリケーションでは、バックエンドデータベースを使用するアプリケーションのバージョンを、Git リポジトリの version2 ブランチに保存します。新しいブランチを指定するようにアプリケーションの設定を更新し、アプリケーションを再デプロイする必要があります。

S を更新するにはimplePHPApp

1. アプリケーションの編集ページを開く

ナビゲーションペインで、アプリをクリックし、SimplePHPApp 行のアクション列の編集をクリックします。

2. アプリケーションの設定を更新する

以下を設定を変更します。

[Branch/Revision]

この設定では、アプリケーションのリポジトリブランチを指定します。S の最初のバージョンがデータベースに接続implePHPApp されませんでした。アプリケーションのデータベースに対応したバージョンを使用するには、この値を **version2** に設定します。

[Document root]

この設定はアプリケーションのルートフォルダを指定します。S の最初のバージョンでは、サーバーの標準ルートフォルダ (/srv/www PHP アプリの場合は) index.php に をインストールするデフォルト設定implePHPApp を使用しました。ここでサブフォルダを指定した場

合、名前だけを指定しても、先頭に「/」が付いたAWS OpsWorks スタックでは標準のフォルダパスに追加されません。バージョン 2 の SimplePHPApp は になる必要があるため /srv/www/web、ドキュメントルートを に設定します**web**。

データソースタイプ

この設定は、データベースサーバーをアプリケーションに関連付けます。この例では、前のステップで作成した MySQL インスタンスを使用しているため、データソースタイプを に設定 OpsWorks し、データベースインスタンスを前のステップで作成したインスタンス db-master1 (mysql) に設定します。データベース名を空のままにします。AWS OpsWorks スタックは、アプリケーションの短縮名 simplephpapp を使用して という名前のデータベースをサーバー上に作成します。

[Save] をクリックして新しい設定を保存します。

Add App

Settings

Name	<input type="text" value="SimplePHP"/>
Type	<input type="text" value="PHP"/>
Document root	<input type="text" value="web"/>

Data Sources

Data source type	<input type="radio"/> RDS <input checked="" type="radio"/> OpsWorks <input type="radio"/> None
Database instance	<input type="text" value="db-master1 (mysql)"/>
Database name	<input type="text"/>

Application Source

Repository type	<input type="text" value="Git"/>
Repository URL	<input type="text" value="git://github.com/amazonwebservices/oj"/>
Repository SSH key	<input type="text" value="Optional"/>
Branch/Revision	<input type="text" value="version2"/>

Add Domains

3. インスタンスSQLを起動します。

アプリケーションを更新すると、AWS OpsWorks スタックは、起動時に新しいアプリケーションサーバーインスタンスに自動的に新しいアプリケーションバージョンをデプロイします。ただし、AWS OpsWorks スタックは新しいアプリケーションバージョンを既存のサーバーインスタンスに自動的にデプロイしません。「」で説明されているように、手動でデプロイする必要があります [ステップ 2.4: アプリケーション - Chef 11 を作成してデプロイする](#)。ここで、更新された S をデプロイできますがimplePHPApp、この例では少し待つことをお勧めします。

短いディグレッション: クックブック、レシピ、AWS OpsWorks スタック属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

これでアプリケーションとデータベースサーバーを設定しましたが、まだ使用する準備はできていません。データベースを設定し、アプリの接続設定を構成する必要があります。AWS OpsWorks スタックはこれらのタスクを自動的に処理しませんが、Chef クックブック、レシピ、動的属性はサポートしています。レシピのペアを実装できます。1 つはデータベースをセットアップするため、もう AWS OpsWorks 1 つはアプリケーションの接続設定を構成するため、スタックで実行させることができます。

必須のレシピを含む phpapp クックブックはすでに実装されており、使用できます。必要に応じて、「[ステップ 3.3: カスタムクックブックを追加する MyStack](#)」に進むこともできます。詳細が必要な場合は、このセクションで、クックブックとレシピの背景を紹介し、レシピの仕組みについて説明します。クックブック自体を確認するには、[phpapp クックブック](#)を参照してください。

トピック

- [レシピと属性](#)
- [データベースのセットアップ](#)
- [アプリケーションをデータベースに接続する](#)

レシピと属性

Chef レシピは基本的に、パッケージのインストール、設定ファイルの作成、シェルコマンドの実行などのタスクをインスタンスで実行する特殊な Ruby アプリケーションです。関連のレシピのグループはクックブックとしてまとめられます。クックブックには、設定ファイルを作成するためのテンプレートなどのサポートファイルも含まれます。

AWS OpsWorks スタックには、組み込みレイヤーをサポートするクックブックのセットがあります。独自のレシピでカスタムクックブックを作成して、インスタンスでカスタムタスクを実行するこ

ともできます。このトピックでは、レシピについて簡単に説明し、レシピを使用してデータベースのセットアップやアプリケーションの接続の設定を行う方法を示します。クックブックとレシピの詳細については、「[クックブックとレシピ](#)」または「[AWS OpsWorks スタックのカスタマイズ](#)」を参照してください。

レシピは通常、入力データの Chef 属性によって決まります。

- これらの属性の一部は Chef によって定義され、オペレーティングシステムなど、インスタンスに関する基本情報を提供します。
- AWS OpsWorks スタックは、レイヤー設定などのスタックに関する情報と、アプリケーションリポジトリなどのデプロイされたアプリケーションに関する情報を含む属性のセットを定義します。

スタックまたはデプロイにカスタムを割り当てることで、このセットに[カスタムJSON](#)属性を追加できます。

- クックブックでも、クックブックに固有の属性を定義できます。

phpapp クックブック属性は、すべて `attributes/default.rb` で定義されます。

スタック属性の完全なリストについては、AWS OpsWorks [スタック設定およびデプロイ属性: Linux](#) 「」および「」を参照してください。[組み込みクックブックの属性](#)。詳細については、「[属性の上書き](#)」を参照してください。

属性は階層構造に編成され、JSONオブジェクトとして表すことができます。

このデータを、次のような Chef ノード構文を使用することによって、アプリケーションに組み込むことができます。

```
[ :deploy ][ :simplephpapp ][ :database ][ :username ]
```

deploy ノードには、アプリケーションのデータベース、Git リポジトリなどに関する情報を含む 1 つのアプリケーションノード simplephpapp ノードがあります。この例は、データベースユーザー名の値を表し、この値が root に解決されます。

データベースのセットアップ

MySQL Layer の組み込み Setup レシピは、アプリの短縮名で という名前のアプリのデータベースを自動的に作成するため、この例では simplephpapp という名前のデータベースが既に作成されています。ただし、アプリケーションでデータを保存するためのテーブルを作成してセットアップを終了する必要があります。テーブルは手動で作成できますが、タスクを処理するカスタムレシピを

実装し、AWS OpsWorks スタックで実行することをお勧めします。このセクションでは、レシピ `dbsetup.rb` を実装する方法を説明します。AWS OpsWorks スタックでレシピを実行する手順については、後で説明します。

リポジトリ内のレシピを表示するには、[dbsetup.rb](#) に移動します。次の例に `dbsetup.rb` のコードを示します。

`execute` は、指定されたコマンドを実行する Chef リソースです。この場合、テーブルを作成するのは MySQL コマンドです。not_if デイレクティブによって、指定されたテーブルがすでに存在する場合、コマンドは実行されません。Chef リソースの詳細については、「[リソースおよびプロバイダについて](#)」を参照してください。

このレシピは、前に説明したノードの構文を使用して、コマンド文字列に属性値を挿入します。たとえば、次のようにしてデータベースのユーザー名を挿入します。

```
#{deploy[:database][:username]}
```

この暗号のようなコードを展開してみましょう。

- 反復のたびに `deploy` は現在のアプリケーションノードに設定されるため、`[:deploy]` `[:app_name]` に解決されます。この例では、`[:deploy][:simplephpapp]` に解決されます。
- 前に示したデプロイ属性値を使用して、ノード全体は `root` に解決されます。
- ノードを `#{}` にラップして文字列に挿入します。

その他のノードの多くも同様に解決されます。`#{node[:phpapp][:dbtable]}` は例外で、カスタムクックブックの属性ファイルで定義され、テーブル名 `urler` に解決されます。したがって、インスタンス SQL で実行される実際のコマンドは次のとおりです。

```
"/usr/bin/mysql
-u root
-p vjud1hw5v8
simplephpapp
-e 'CREATE TABLE urler(
  id INT UNSIGNED NOT NULL AUTO_INCREMENT,
  author VARCHAR(63) NOT NULL,
  message TEXT,
  PRIMARY KEY (id))'
```

このコマンドは、デプロイ属性の認証情報とデータベース名を使用して、ID、作成者、およびメッセージフィールドを持つ、`urler` という名前のテーブルを作成します。

アプリケーションをデータベースに接続する

2 番目の要素はアプリケーションです。アプリケーションはテーブルにアクセスする際にデータベースパスワードなどの接続情報が必要になります。S には `implePHPApp`、実質的に 1 つの作業ファイルしか `index.php` ありません `app.php`。すべての作業ファイルは `ロード` です `app.php`。

`app.php` にはデータベース接続を処理する `db-connect.php` が含まれていますが、このファイルはリポジトリにはありません。は特定のインスタンスに基づいてデータベースを定義するため、`db-connect.php` `db-connect.php` を事前に作成することはできません。代わりに、`appsetup.rb` レシピは、デプロイ属性の接続データを使用して `db-connect.php` を生成します。

リポジトリ内のレシピを表示するには、[appsetup.rb](#) に移動します。次の例に `appsetup.rb` のコードを示します。

`dbsetup.rb` と同様に、`appsetup.rb` は、`deploy` ノードでアプリケーション (ここでも `simplephpapp`) を繰り返し処理します。これは、`script` リソースと `template` リソースを含むコードブロックを実行します。

`script` リソースは、PHPアプリケーションの依存関係マネージャーである [Composer](#) をインストールします。次に、Composer の `install` コマンドを実行して、サンプルアプリケーションの依存ファイルをアプリケーションのルートディレクトリにインストールします。

`template` リソースは、`db-connect.php` を生成して、これを `/srv/www/simplephpapp/current` に配置します。次の点に注意してください:

- レシピでは、条件ステートメントを使用して、インスタンスのオペレーティングシステムによって異なるファイル所有者を指定します。
- `only_if` ディレクティブは、指定されたディレクトリが存在する場合にのみテンプレートを生成するように Chef に指示します。

`template` リソースは、基本的に関連ファイルと同じコンテンツと構造を持っているが、さまざまなデータ値のプレースホルダが含まれているテンプレートを操作します。`source` パラメータは、テンプレート `db-connect.php.erb` を指定します。このテンプレートは、`phpapp` のクックブックの `templates/default` ディレクトリにあり、次の内容が含まれます。

Chef はテンプレートを処理するときに、プレースホルダ `<%= =>` を template リソース内の対応する変数の値に置き換えます。これらの変数はデプロイ属性から取得されます。したがって、生成されるファイルは次のようになります。

ステップ 3.3: カスタムクックブックを に追加する MyStack

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

カスタムクックブックは、アプリケーションと同様に、リポジトリに保存します。各スタックには、1 組のカスタムクックブックを含むリポジトリを設定できます。次に、AWS OpsWorks スタックのインスタンスにカスタムクックブックをインストールするように スタックに指示します。

1. ナビゲーションペインで [Stack] をクリックして、現在のスタックのページを表示します。
2. [Stack Settings] をクリックし、[Edit] をクリックします。
3. スタック設定を次のように変更します。
 - Use custom Chef Cookbooks (カスタムChefクックブックを使用する) - Yes (はい)
 - Repository type (リポジトリタイプ) - Git
 - リポジトリ URL – **git://github.com/amazonwebservices/opsworks-example-cookbooks.git**
4. [Save] をクリックしてスタック設定を更新します。



Use custom Chef cookbooks	Yes <input type="checkbox"/>
Repository type	Git <input type="text"/> Select the repository type
Repository URL	git://github.com/amazonwebservices/opsworks-example-cookbooks.git
Repository SSH key	Optional

AWS OpsWorks その後、スタックは、クックブックリポジトリの内容をスタックのすべてのインスタンスにインストールします。新しいインスタンスを作成すると、AWS OpsWorks スタックによってクックブックリポジトリが自動的にインストールされます。

Note

いずれかのクックブックを更新したり、リポジトリに新しいクックブックを追加したりする必要がある場合は、スタック設定を変更せずに行うことができます。AWS OpsWorks スタックは、更新されたクックブックをすべての新しいインスタンスに自動的にインストールします。ただし、AWS OpsWorks スタックは、スタックのオンラインインスタンスに更新されたクックブックを自動的にインストールしません。スタックコマンドを実行してクックブックを更新するように AWS OpsWorks Update Cookbooksスタックに明示的に指示する必要があります。詳細については、「[スタックコマンドの実行](#)」を参照してください。

ステップ 3.4: レシピを実行する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

カスタムクックブックの準備ができたなら、適切なインスタンスでレシピを実行する必要があります。レシピは[手動で実行](#)できます。ただし、レシピは通常、インスタンスが起動した後、またはアプリケーションをデプロイするときなど、インスタンスのライフサイクルでの予測可能なポイントで実行される必要があります。このセクションでは、AWS OpsWorks スタックが適切なタイミングで自動的に実行するという、よりシンプルなアプローチについて説明します。

AWS OpsWorks スタックは、レシピの実行を簡素化する一連の[ライフサイクルイベント](#)をサポートします。たとえば、Setup イベントはインスタンスが起動した後で発生し、Deploy イベントはアプリケーションをデプロイするときに発生します。各レイヤーには、各ライフサイクルに関連付けられた一連の組み込みレシピがあります。インスタンスでライフサイクルイベントが発生したときに、エージェントはインスタンスの各レイヤーに関連付けられたレシピを実行します。AWS OpsWorks

スタックでカスタムレシピを自動的に実行するには、それを適切なレイヤーの適切なライフサイクルイベントに追加すると、組み込みレシピが完了した後にエージェントがレシピを実行します。

この例では、M dbsetup.rbと PHP App Server インスタンスMySQLInstance appsetup.rbで2つのレシピを実行する必要があります。

Note

コンソールでレシピを指定するには、`cookbook_name::recipe_name` 形式を使用します。`recipe_name`には .rb 拡張子は含まれません。例えば、dbsetup.rb は `phpapp::dbsetup` と指定します。

カスタムレシピをライフサイクルイベントに割り当てるには

1. Layers ページの My でSQL、レシピをクリックしてから Edit をクリックします。
2. [Custom Chef recipes] (カスタムシェフレシピ) セクションで、[Deploy] (デプロイ) に対して [phpapp::dbsetup](#) と入力します。



3. [+] アイコンをクリックしてレシピをイベントに割り当て、[Save] をクリックして新しいレイヤー設定を保存します。
4. Layers ページに戻り、PHPApp Server Layer の Deploy イベント `phpapp::appsetup` に割り当てる手順を繰り返します。

ステップ 3.5: SimplePHPApp、バージョン 2 をデプロイする

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。




最後のステップは、新しいバージョンの S をデプロイすることですimplePHPApp。

S をデプロイするにはimplePHPApp

1. アプリページで、SimplePHPApp アプリのアクションでデプロイをクリックします。

Apps

An app represents code stored in a repository that you want to install on application server instances. When you deploy the app, OpsWorks downloads the code from the repository to the specified server instances. [Learn more.](#)

Name	Type	Last deployment	Actions
SimplePHPApp	php	2013-02-19 21:34:43 UTC	 deploy  edit  delete
+ App			

2. デフォルト値を変更せずに、[Deploy] をクリックします。

Deploy App

Settings

App	SimplePHPApp
Command	Deploy
Comment	Optional

Advanced »

Instances ⓘ

OpsWorks will run this command on **2 of 2** instances. The assigned recipes are run on all selected instances.

- | | |
|--|--|
| <input checked="" type="checkbox"/> PHP App Server
Click to select instances in this layer | <input checked="" type="checkbox"/> php-app1 ● |
| <input checked="" type="checkbox"/> MySQL
Click to select instances in this layer | <input checked="" type="checkbox"/> db-master1 ● |

Cancel **Deploy**

[Deploy App (デプロイアプリケーション)] ページで [Deploy (デプロイ)] をクリックすると、デプロイのライフサイクルイベントをトリガーし、エージェントにデプロイレシピを実行するように通知します。デフォルトでは、スタックのすべてのインスタンスでイベントをトリガーします。組み込みの Deploy レシピは、この場合はアプリタイプに適したインスタンスである PHP App Server インスタンスにのみアプリをデプロイします。ただし、通常、他のインスタンスで Deploy イベントをトリガーして、アプリケーションのデプロイに応答できるようにすると便利です。この場合、データベースをセットアップするために、インスタンスSQLで Deploy をトリガーすることもできます。

次の点に注意してください:

- PHP App Server インスタンスのエージェントは、レイヤーの組み込みレシピを実行し、その後を `appsetup.rb` を実行します。これにより `appsetup.rb`、アプリケーションのデータベース接続が設定されます。
- My インスタンスのSQL エージェントは何もインストールしませんが、 `dbsetup.rb` を実行して URL テーブル `dbsetup.rb` を作成します。

デプロイが完了すると、[Deployment] ページで [Status] が `successful` に変わります。

ステップ 3.6: S を実行するimplePHPApp

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

デプロイステータスが成功に変わったら、次のように新しい S implePHPApp バージョンを実行できます。

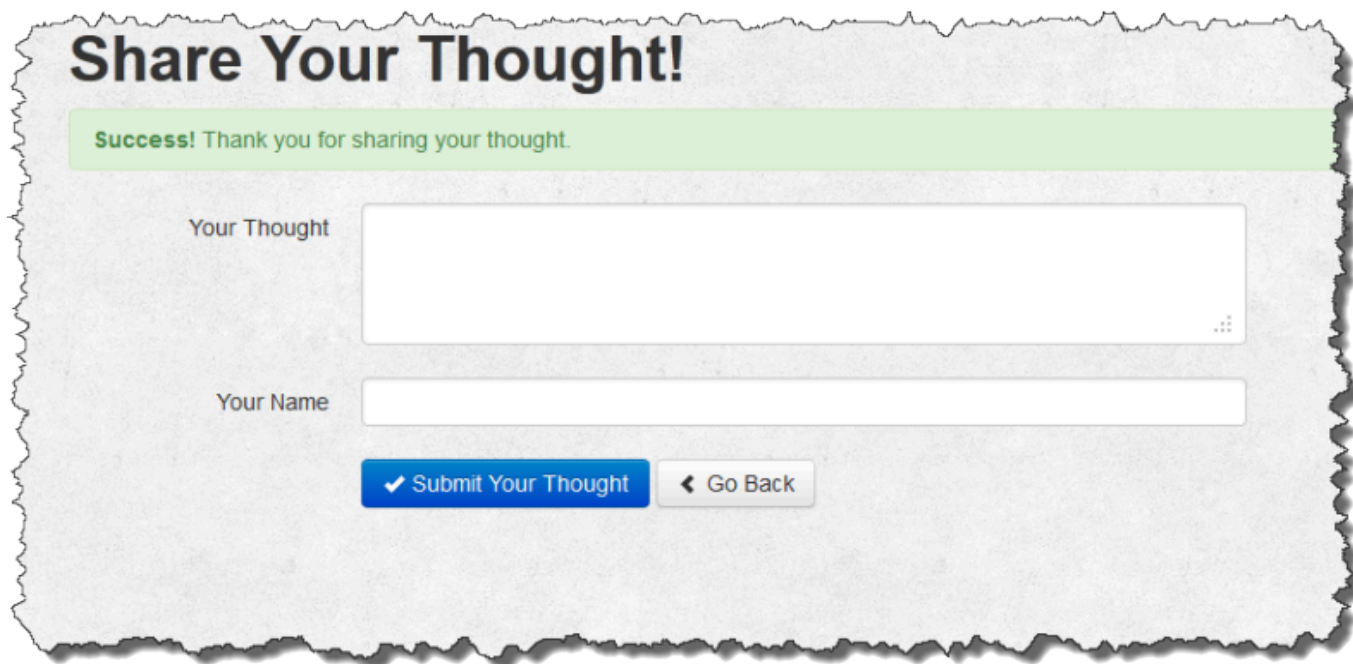
S を実行するにはimplePHPApp

1. [Instances] ページで、[php-app1] 行のパブリック IP アドレスをクリックします。

ブラウザに次のページが表示されます。



2. [Share Your Thought] (考えを共有) をクリックし、そして、[Your Thought] (あなたの考え) には **Hello world!** のようなもの、そして [Your Name] (あなたの名前) にはあなたの名前を入力します。次に、[Submit Your Thought] をクリックして、データベースにメッセージを追加します。



The screenshot shows a web form titled "Share Your Thought!". At the top, a green banner displays the message "Success! Thank you for sharing your thought." Below this, there are two input fields: "Your Thought" and "Your Name". At the bottom of the form, there are two buttons: a blue button labeled "Submit Your Thought" with a checkmark icon, and a grey button labeled "Go Back" with a left-pointing arrow icon.

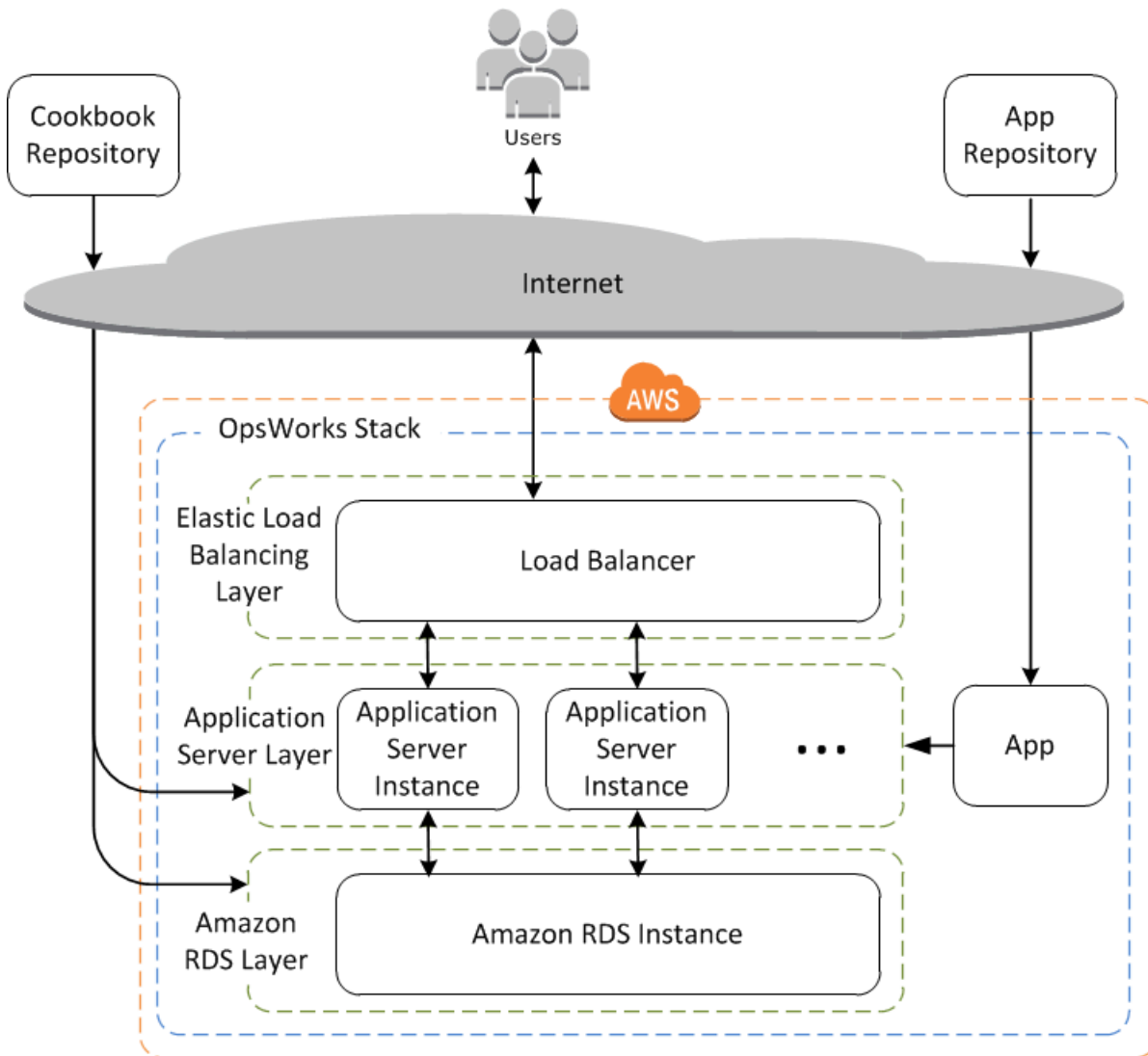
3. [Go Back] をクリックすると、データベース内のすべてのメッセージが表示されます。

ステップ 4: スケールアウトする MyStack

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

MyStack 現在、にはアプリケーションサーバーが 1 つしかありません。本稼働用のスタックでは、おそらく、複数のアプリケーションサーバーで着信トラフィックを処理し、ロードバランサーでアプリケーションサーバー全体の着信トラフィックを均等に分散させる必要があります。アーキテクチャは次のようになります。



AWS OpsWorks スタックを使用すると、スタックを簡単にスケールアウトできます。このセクションでは、2 つ目の 24/7 PHP App Server インスタンスを に追加 MyStack し、両方のインスタンスを Elastic Load Balancing ロードバランサーの背後に配置することで、スタックをスケールアウトする方法の基本について説明します。この手順を簡単に拡張して任意の数の 24/7 インスタンスを追加したり、時間ベースまたは負荷ベースのインスタンスを使用して AWS OpsWorks スタックでスタックを自動的にスケーリングしたりできます。詳細については、「[時間ベースおよび負荷ベースのインスタンスによる負荷の管理](#)」を参照してください。

ステップ 4.1: Load Balancer の追加

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソ

リユーシオンに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Elastic Load Balancing は、受信アプリケーショントラフィックを複数の Amazon EC2 インスタンスに自動的に分散する AWS サービスです。Elastic Load Balancing は、トラフィックの分散の他に、以下のことも行います。

- 異常な Amazon EC2 インスタンスを検出します。

問題のあるインスタンスが復旧するまで、トラフィックのルートを残りの正常なインスタンスに変更します。

- 着信トラフィックに応じて、自動的にそのリクエスト処理能力を拡張します。

Note

ロードバランサーで 2 つの目的を達成できます。明らかな目的は、アプリケーションサーバーで負荷を均等化することです。さらに、多くのサイトでは、ユーザーがアプリケーションサーバーおよびデータベースに直接アクセスできないようにすることが求められます。AWS OpsWorks スタックでは、次のようにパブリックサブネットとプライベートサブネットを持つ仮想プライベートクラウド (VPC) でスタックを実行することでこれを行うことができます。

- アプリケーションサーバーとデータベースをプライベートサブネットに配置します。このサブネットには、他のインスタンスからアクセスできます VPC が、ユーザーがアクセスすることはできません。
- パブリックサブネット内のロードバランサーにユーザートラフィックを送ると、このトラフィックはプライベートサブネット内のアプリケーションサーバーに転送され、ユーザーにレスポンスが返されます。

詳細については、「[でのスタックの実行 VPC](#)」を参照してください。このワークスルーの例を拡張してで実行する AWS CloudFormation テンプレートについては VPC、[OpsWorksVPCtemplates.zip ファイル](#) をダウンロードします。

多くの場合、Elastic Load Balancing はレイヤーと呼ばれますが、他の組み込みレイヤーとは動作が多少異なります。レイヤーを作成してインスタンスを追加する代わりに、Amazon EC2コンソールを使用して Elastic Load Balancing ロードバランサーを作成し、既存のレイヤーの 1 つ、通常はアプリケーションサーバーレイヤーにアタッチします。AWS OpsWorks スタックは、レイヤーの既存のインスタンスをサービスに登録し、新しいインスタンスを自動的に追加します。次の手順では、ロードバランサーを MyStack の PHP App Server レイヤーに追加する方法について説明します。

Note

AWS OpsWorks スタックは Application Load Balancer をサポートしていません。Classic Load Balancer は AWS OpsWorks スタックでのみ使用できます。

ロードバランサーを PHP App Server レイヤーにアタッチするには

1. Amazon EC2コンソールを使用して、新しいロードバランサーを作成します MyStack。詳細は、アカウントが EC2 Classic をサポートしているかどうかによって異なります。詳細については、「[Elastic Load Balancing の開始方法](#)」を参照してください。[Create Load Balancer] ウィザードを実行するときに、ロードバランサーを次のように設定します。

Load Balancer の定義

ロードバランサーに PHP-LB のようなわかりやすい名前を割り当てて、AWS OpsWorks スタックコンソールで見つけやすくします。次に [Continue] を選択し、残りの設定のデフォルト値を受け入れます。

LB 内部の作成メニューから 1 つ以上のサブネットVPCを持つを選択する場合は、ロードバランサーによってトラフィックがルーティングされるアベイラビリティゾーンごとにサブネットを選択する必要があります。

セキュリティグループの割り当て

アカウントがデフォルトのをサポートしている場合VPC、ウィザードはこのページを表示してロードバランサーのセキュリティグループを決定します。EC2 Classic ではこのページは表示されません。

このチュートリアルでは、デフォルトのVPCセキュリティグループを選択します。

セキュリティ設定の構成

Load Balancerの定義ページでロードバランサープロトコルHTTPSとして を選択した場合は、このページで証明書、暗号、SSLプロトコルの設定を行います。 Load Balancer このウォークスルーでは、デフォルト値を受け入れ、[Configure Health Check] を選択します。

ヘルスチェックの設定

ping パスに / を設定し、残りの設定にデフォルト値を受け入れます。

EC2インスタンスの追加

Continue を選択します。 AWS OpsWorks スタックはロードバランサーにインスタンスを自動的に登録します。

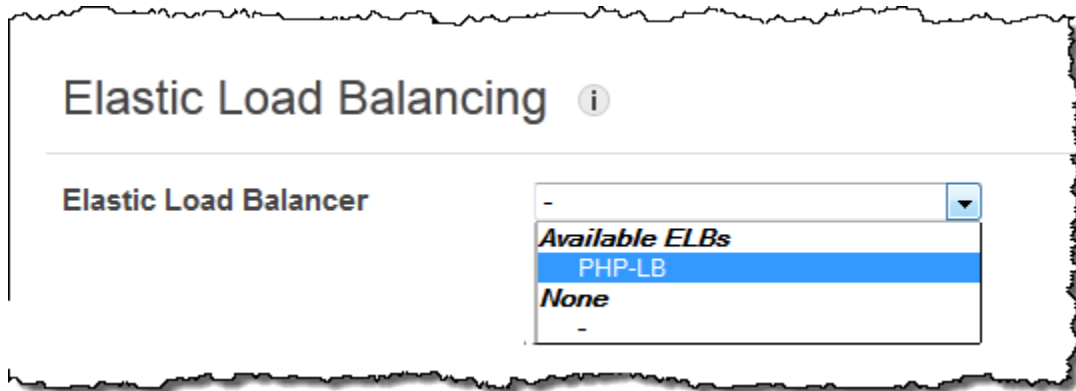
タグの追加

タグを追加すると検索しやすくなります。各タグはキーと値のペアです。たとえば、このウォークスルーでは、キーとして「**Description**」、値として「**Test LB**」を指定できます。

確認

選択内容を確認して [Create] を選択し、次に [Close] を選択すると、ロードバランサーが起動します。

2. アカウントがデフォルトのをサポートしている場合はVPC、ロードバランサーを起動した後、セキュリティグループに適切な進入ルールがあることを確認する必要があります。デフォルトルールでは、着信トラフィックは受け入れられません。
 1. Amazon EC2ナビゲーションペインでセキュリティグループを選択します。
 2. デフォルトのVPCセキュリティグループを選択する
 3. [Inbound] (インバウンド) タブで、[Edit] (編集) を選択します。
 4. このウォークスルーでは、[Source] を [Anywhere] に設定します。これにより、ロードバランサーでは任意の IP アドレスからの着信トラフィックを受け入れます。
3. AWS OpsWorks スタックコンソールに戻ります。[Layers] (レイヤー) ページで、レイヤーの [Network] (ネットワーク) リンクを選択し、[Edit] (編集) を選択します。
4. [Elastic Load Balancing] で、ステップ 1 で作成した ロードバランサーを選択し、[Save] を選択します。



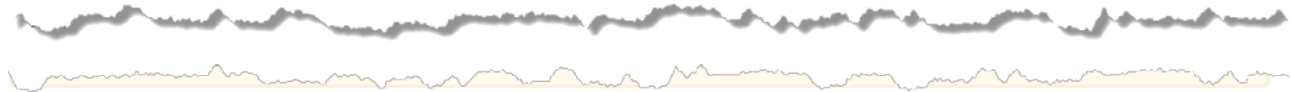
ロードバランサーをレイヤーにアタッチすると、AWS OpsWorks スタックはレイヤーの現在のインスタンスを自動的に登録し、オンラインになると新しいインスタンスを追加します。

5. [Layers] (レイヤー) ページで、ロードバランサーの名前をクリックして詳細ページを開きます。登録が完了し、インスタンスがヘルスチェックに合格すると、ロードバランサーページのインスタンスの横に緑色のチェックマークが AWS OpsWorks スタックに表示されます。

ELB PHP-LB

[Detach](#)

Elastic Load Balancing associates your load balancer with your EC2 instances using IP addresses. [Learn more.](#)



192.0.2.1/20 -
us-west-2a

1

php-app1 ● ✓ InService

ロードバランサーに リクエストを送信implePHPApp して S を実行できるようになりました。

ロードバランサーimplePHPApp を介して S を実行するには

1. ロードバランサーの詳細ページを再度開きます (開いていない場合)。
2. プロパティページで、インスタンスのヘルスチェックステータスを確認し、ロードバランサー DNSの名前をクリックして S を実行しますimplePHPApp。ロードバランサーは、リクエストを PHP App Server インスタンスに転送し、レスポンスを返します。これは、PHPApp Server インスタンスのパブリック IP アドレスをクリックしたときのレスポンスとまったく同じように見えます。

ELB PHP-LB

Elastic Load Balancing associates your load balancer with your EC2 instances using IP addresses. [Learn more.](#)

Settings

Layer	PHP App Server
DNS Name	PHP-LB-862966592.us-west-2.elb.amazonaws.com
Region	US West (Oregon)
Attached availability zones	us-west-2a

Note

AWS OpsWorks スタックはHAProxyロードバランサーもサポートしているため、一部のアプリケーションには利点がある場合があります。詳細については、「[HAProxy AWS OpsWorks スタックレイヤー](#)」を参照してください。

ステップ 4.2: PHPアプリケーションサーバーインスタンスを追加する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

これでロードバランサーが配置され、PHP App Server レイヤーにインスタンスを追加することでスタックをスケールアウトできます。ユーザーから見た場合、このオペレーションはシームレスです。新しい PHP App Server インスタンスがオンラインになるたびに、AWS OpsWorks スタックによってロードバランサーに自動的に登録され、S がデプロイされるためimplePHPApp、サーバーは受信トラフィックの処理をすぐに開始できます。簡潔にするために、このトピックでは 1 つの PHP App Server インスタンスを追加する方法を示しますが、同じ方法を使用して必要な数だけ追加できます。

PHP App Server レイヤーに別のインスタンスを追加するには

1. インスタンス ページで、PHPアプリサーバー の + インスタンス をクリックします。
2. デフォルト設定を受け入れて、[Add Instance] をクリックします。
3. [start] をクリックしてインスタンスを開始します。

ステップ 4.3: モニタリング MyStack

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは Amazon CloudWatch を使用してスタックのメトリクスを提供し、モニタリングページで便宜上要約します。スタック全体、特定のレイヤー、特定のインスタンスに対するメトリクスを閲覧することができます。

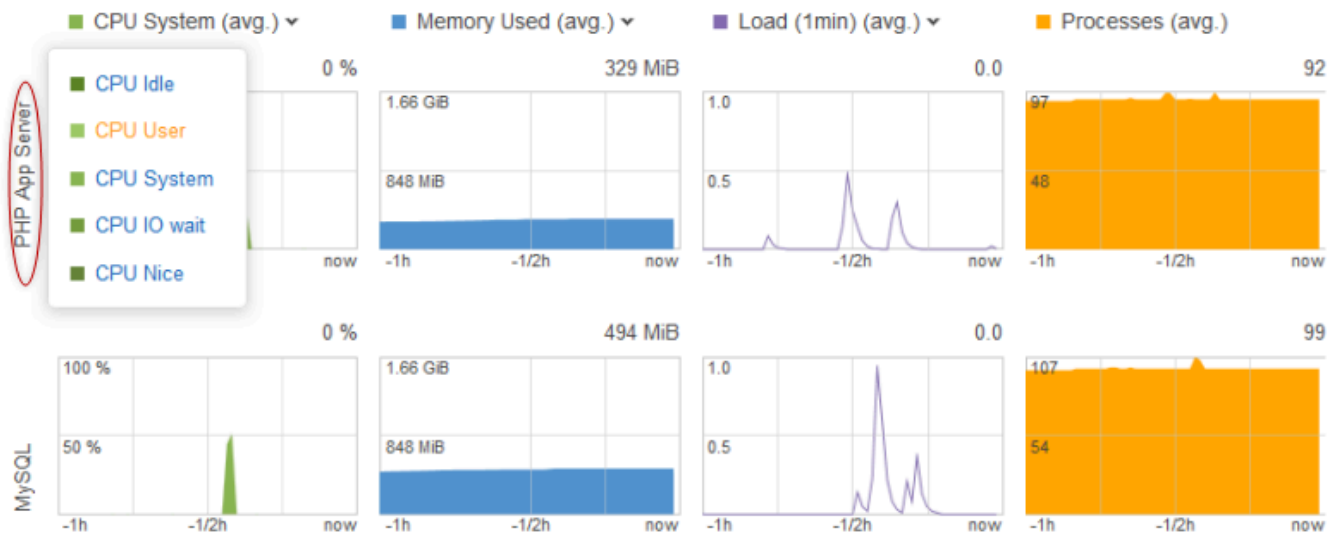
をモニタリングするには MyStack

1. ナビゲーションペインで [Monitoring] をクリックすると、各レイヤーの平均メトリクスを含む一連のグラフが表示されます。CPU システム、使用されたメモリ、ロードのメニューを使用して、さまざまな関連メトリクスを表示できます。

Monitoring Layers

refreshing in 69 sec

1 hour ▾



2. PHP App Server をクリックして、レイヤーの各インスタンスのメトリクスを表示します。

Layer PHP App Server

refreshing in 111 sec

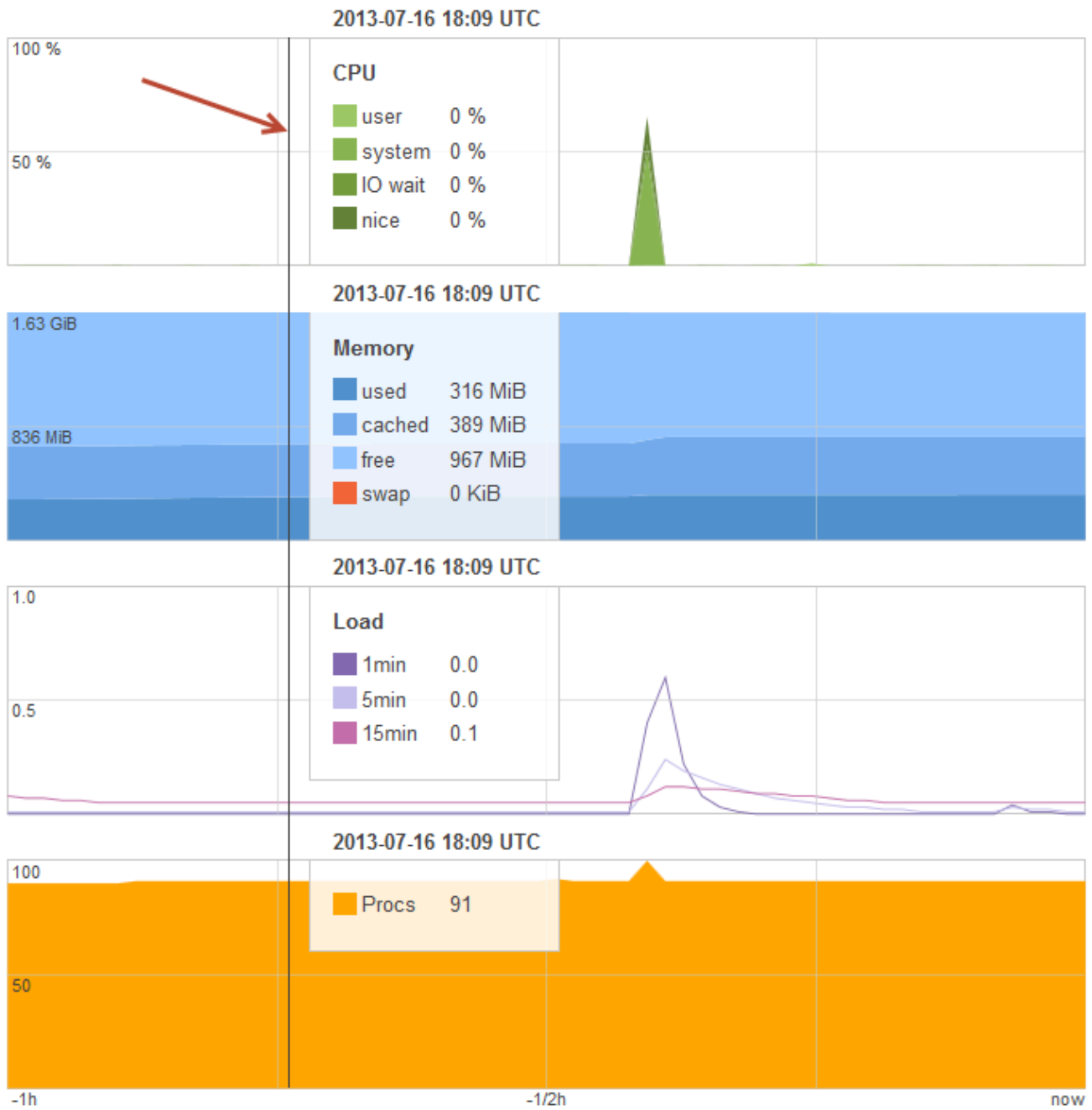
1 hour ▾



3. [php-app1] をクリックすると、そのインスタンスのメトリクスが表示されます。スライダーを動かすことで、特定の時点のメトリクスを確認することができます。

Instance php-app1 ●

refreshing in



Note

AWS OpsWorks スタックは Ganglia モニタリングサーバーもサポートしており、一部のアプリケーションには利点がある場合があります。詳細については、「[Ganglia レイヤー](#)」を参照してください。

ステップ 5: 削除 MyStack

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Amazon EC2 インスタンスなどの AWS リソースの使用を開始すると、使用量に基づいて課金されます。ひとまず終了した場合は、インスタンスを停止して不要な料金が発生しないようにする必要があります。不要なスタックは、削除することができます。

削除するには MyStack

1. すべてのインスタンスの停止

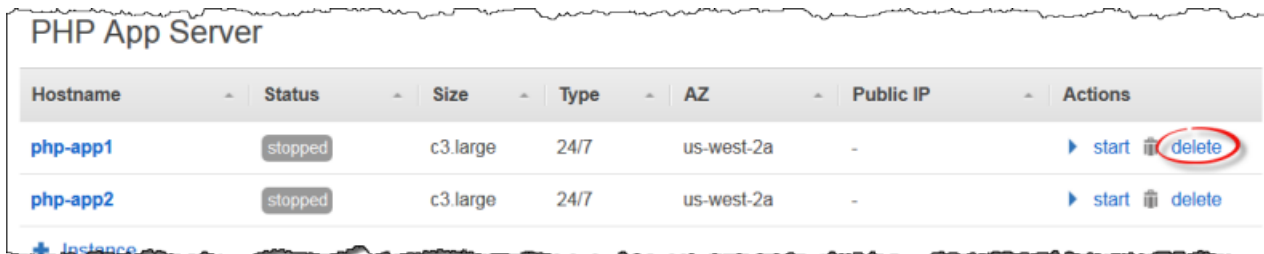
[Instances] ページで、[Stop All Instances] をクリックして、操作の確認を求められたら [Stop] をクリックします。

The screenshot shows the 'Instances' page with a summary bar: 'Instances' (1 total, 1 online, 0 setting up, 0 shutting down, 0 stopped, 0 errors). A 'Stop All Instances' button is visible. Below is a confirmation dialog with the text: 'Are you sure you want to stop this stack?' and 'All data not stored on EBS volumes will be lost.' The dialog has 'Cancel' and 'Stop' buttons.

Stop をクリックすると、AWS OpsWorks スタックは関連付けられた Amazon EC2 インスタンスを終了しますが、Elastic IP アドレスや Amazon EBS ボリュームなどの関連付けられたリソースは終了しません。

2. すべてのインスタンスの削除

インスタンスを停止すると、関連付けられた Amazon EC2 インスタンスが終了するだけです。インスタンスのステータスが停止になってから、各インスタンスを削除する必要があります。PHP アプリケーションサーバーレイヤーで、php-app1 インスタンスの Actions 列の delete をクリックします。



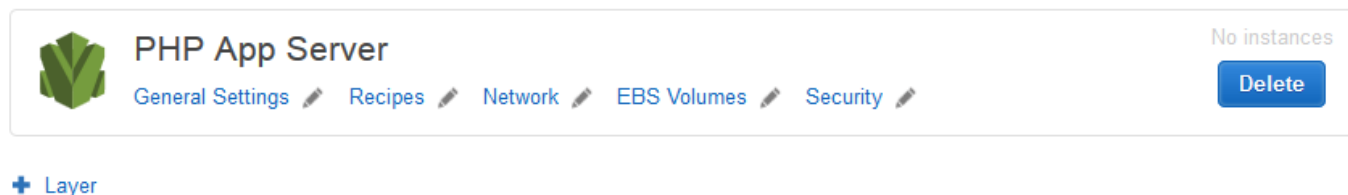
Hostname	Status	Size	Type	AZ	Public IP	Actions
php-app1	stopped	c3.large	24/7	us-west-2a	-	▶ start delete
php-app2	stopped	c3.large	24/7	us-west-2a	-	▶ start delete

AWS OpsWorks その後、スタックは削除の確認をユーザーに求め、依存するリソースを表示します。これらのリソースの一部またはすべて保持することを選択することができます。この例では、依存リソースがないので、[Delete] をクリックします。

php-app2 と MySQL インスタンス db-master1 のプロセスを繰り返します。db-master1 には、Amazon Elastic Block Store の関連ボリュームが含まれていることに注意してください。それは、デフォルトで選択されています。選択した状態にしておくと、インスタンスと共にボリュームも削除されます。

3. レイヤーを削除します。

[Layers] (レイヤー) ページで、[Delete] (削除) をクリックし、再び [Delete] (削除) をクリックして削除を確認します。



MySQL レイヤーのプロセスを繰り返します。

4. アプリケーションの削除

アプリページで、SimplePHPApp アプリのアクション列の削除をクリックし、削除をクリックして確認します。

Name	Type	Last Deployment	Actions
SimplePHPApp	PHP	2013-09-13 14:54:15 UTC	deploy edit delete

Are you sure that you want to delete SimplePHPApp?

If you delete this app, all your configuration settings will be lost.

Cancel Delete

+ App

5. 削除 MyStack

[Stack] ページで、[Delete Stack] をクリックし、[Delete] をクリックして削除を確定します。

MyStack


Stack Settings Delete Stack

Are you sure that you want to delete MyStack?

If you delete this stack, all your settings will be lost.

Cancel Delete

A stack represents a collection of EC2 instances and related AWS resources that have a common purpose and that you want to manage collectively. Within a stack, you use layers to define the configuration of your instances and use apps to specify the code you want to deploy. [Learn more.](#)

1  Add your first layer

この演習はこれで終了です。

Node.js スタックの初回作成

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

この例では、Node.js アプリケーションサーバーをサポートする Linux スタックの作成方法と、シンプルなアプリケーションのデプロイ方法について説明します。スタックは以下のコンポーネントから構成されます。

- 2 つのインスタンスを伴う [\[Node.js App Server Layer\]](#) (Node.js アプリケーションサーバーレイヤー)
- アプリケーションサーバーインスタンスにトラフィックを分散させる [Elastic Load Balancing](#) (Elastic ロードバランシング)
- バックエンドデータベースを提供する [Amazon Relational Database Service \(Amazon RDS\) サービレイヤー](#)

トピック

- [前提条件](#)
- [アプリケーションの実装](#)
- [データベースサーバーとロードバランサーの作成](#)
- [スタックの作成](#)
- [アプリケーションのデプロイ](#)
- [次のステップ](#)

前提条件

このウォークスルーでは、以下のことを前提としています。

- AWS アカウントがあり、AWS OpsWorks スタックの使用方法に関する基本的な理解がある。

AWS OpsWorks スタックまたは を初めて使用する場合はAWS、「」の入門チュートリアルを完了して基本を学習します[Chef 11 Linux スタックの使用開始](#)。

- Node.js アプリケーションの基本的な実装方法を理解している。

Node.js に慣れていない場合は、「[Node: Up and Running](#)」などの入門チュートリアルを完了して、基本事項を学習してください。

- この例で使用する予定のAWSリージョンに、少なくとも 1 つのスタックを作成済みです。

リージョンで最初のスタックを作成すると、AWS OpsWorks スタックはレイヤータイプごとに Amazon Elastic Compute Cloud (Amazon EC2) セキュリティグループを作成します。Amazon

RDS データベース (DB) インスタンスを作成するには、これらのセキュリティグループが必要です。AWS OpsWorks スタックを初めて使用する場合は、「」のチュートリアルに従ったときと同じリージョンをこの例に使用することをお勧めします[Chef 11 Linux スタックの使用開始](#)。新しいリージョンを使用する場合は、リージョンに新しいスタックを作成します。このスタックにレイヤーやインスタンスは必要ありません。スタックを作成するとすぐに、AWS OpsWorks スタックはリージョンに一連のセキュリティグループを自動的に追加します。

- スタックは[デフォルトの VPC](#)で作成します。

このチュートリアルでは EC2-Classic を使用できますが、一部の詳細は若干異なります。たとえば、EC2-Classic では、サブネットの代わりにインスタンスのアベイラビリティゾーン (AZ) を指定します。

- ユーザーIAMには AWS OpsWorks、スタックに対するフルアクセス許可があります。

セキュリティ上の理由から、ウォークスルー用にアカウントのルート認証情報を指定しないことを強くお勧めします。代わりに、AWS OpsWorks スタックのフルアクセス許可を持つユーザーを作成し、それらの認証情報を AWS OpsWorks スタックで使用します。詳細については、「[管理者ユーザーの作成](#)」を参照してください。

アプリケーションの実装

このチュートリアルでは、Amazon RDS DB インスタンスに接続し、インスタンスのデータベースを一覧表示するシンプルな [Express](#) アプリケーションを使用します。

アプリケーションを実装するには、ワークステーション上の使いやすい場所に nodedb という名前のディレクトリを作成し、以下に示す 3 つのファイルを追加します。

トピック

- [パッケージディスクリプター](#)
- [レイアウトファイル](#)
- [コードファイル](#)

パッケージディスクリプター

アプリケーションのパッケージディスクリプターを作成するには、以下に示す内容の package.json というファイルを nodedb ディレクトリに追加します。package.json は、Express アプリケーションに必要であり、アプリケーションのルートディレクトリに配置されている必要があります。

```
{
  "name": "Nodejs-DB",
  "description": "Node.js example application",
  "version": "0.0.1",
  "dependencies": {
    "express": "*",
    "ejs": "*",
    "mysql": "*"
  }
}
```

この例の `package.json` は、ほぼ最小限の内容です。必要な `name` 属性と `version` 属性が定義され、従属パッケージがリストされています。

- `express` は、[Express](#) パッケージを参照しています。
- `ejs` は、アプリケーションがHTMLレイアウトファイルにテキストを挿入するために使用する [EJS](#) パッケージを参照します。
- `mysql` は、アプリケーションがRDSインスタンスへの接続に使用する [node-mysql](#) パッケージを参照します。

パッケージディスクリプターファイルの詳細については、「[package.json](#)」を参照してください。

レイアウトファイル

アプリケーションのレイアウトファイルを作成するには、`views` ディレクトリに `nodedb` ディレクトリを追加し、以下に示す内容の `views` というファイルを `index.html` に追加します。

```
<!DOCTYPE html>
<html>
<head>
  <title>AWS Opsworks Node.js Example</title>
</head>
<body>
  <h1>AWS OpsWorks Node.js Example</h1>
  <p>Amazon RDS Endpoint: <i><%= hostname %></i></p>
  <p>User: <i><%= username %></i></p>
  <p>Password: <i><%= password %></i></p>
  <p>Port: <i><%= port %></i></p>
```

```
<p>Database: <i><%= database %></i></p>

<p>Connection: <%= connectionerror %></p>
<p>Databases: <%= databases %></p>
</body>
</html>
```

この例では、レイアウトファイルは Amazon からのデータを表示するシンプルなHTMLドキュメントですRDS。各 `<%= ... =>` 要素は、次に作成するコードファイル内で定義されている変数の値を示します。

コードファイル

アプリケーションのコードファイルを作成するには、以下に示す内容の `server.js` ファイルを `nodedb` ディレクトリに追加します。

Important

AWS OpsWorks スタックでは、Node.js アプリケーションのメインコードファイルに という名前 `server.js` を付け、アプリケーションのルートフォルダに配置する必要があります。

```
var express = require('express');
var mysql = require('mysql');
var dbconfig = require('opsworks'); //[1] Include database connection data
var app = express();
var outputString = "";

app.engine('html', require('ejs').renderFile);

//[2] Get database connection data
app.locals.hostname = dbconfig.db['host'];
app.locals.username = dbconfig.db['username'];
app.locals.password = dbconfig.db['password'];
app.locals.port = dbconfig.db['port'];
app.locals.database = dbconfig.db['database'];
app.locals.connectionerror = 'successful';
app.locals.databases = '';

//[3] Connect to the Amazon RDS instance
var connection = mysql.createConnection({
```

```
    host: dbconfig.db['host'],
    user: dbconfig.db['username'],
    password: dbconfig.db['password'],
    port: dbconfig.db['port'],
    database: dbconfig.db['database']
  });

connection.connect(function(err)
{
  if (err) {
    app.locals.connectionerror = err.stack;
    return;
  }
});

// [4] Query the database
connection.query('SHOW DATABASES', function (err, results) {
  if (err) {
    app.locals.databases = err.stack;
  }

  if (results) {
    for (var i in results) {
      outputString = outputString + results[i].Database + ', ';
    }
    app.locals.databases = outputString.slice(0, outputString.length-2);
  }
});

connection.end();

app.get('/', function(req, res) {
  res.render('./index.html');
});

app.use(express.static('public'));

//[5] Listen for incoming requests
app.listen(process.env.PORT);
```

この例では、データベース接続情報を表示し、データベースサーバーに照会して、サーバーのデータベースを表示しています。これは、必要に応じてデータベースとやり取りするために簡単に一般化することができます。コード内の番号付きのコメントに対応する注釈を以下に示します。

[1] Include database connection data (データベース接続データを含める)

この `require` ステートメントでは、データベース接続データを含めています。後で説明するように、データベースインスタンスをアプリケーションにアタッチすると、AWS OpsWorks スタックは接続データを `opsworks.js` という名前のファイルに配置します。これは次のようになります。

```
exports.db = {
  "host": "nodeexample.cd1qlk5uwd0k.us-west-2.rds.amazonaws.com",
  "database": "nodeexampledb",
  "port": 3306,
  "username": "opsworkuser",
  "password": "your_pwd",
  "reconnect": true,
  "data_source_provider": "rds",
  "type": "mysql"}
```

`opsworks.js` はアプリケーションの `shared/config` ディレクトリ `/srv/www/app_shortcode/shared/config` にあります。ただし、AWS OpsWorks スタックはアプリケーションのルートディレクトリ `opsworks.js` にシンボリックリンクを配置するため、のみを使用してオブジェクトを含めることができます `require 'opsworks'`。

[2] Get database connection data (データベース接続データを取得する)

これら一連のステートメントは、`opsworks.js` に記述されている接続データを表示します。そのため、`db` オブジェクトの値を一連の `app.locals` プロパティに割り当てています。これらのプロパティはそれぞれ、`index.html` ファイル内の `<%= ... %>` 要素のいずれかに対応しています。レンダリング後のドキュメントでは、`<%= ... %>` 要素が、対応するプロパティ値に置き換わります。

[3] Amazon RDSインスタンスに接続する

この例では、`node-mysql` を使用してデータベースにアクセスしています。データベースに接続するために、接続データを `connection` に渡して `createConnection` オブジェクトを作成し、`connection.connect` を呼び出して接続を確立しています。

[4] Query the database (データベースを照会する)

接続を確立した後、データベースを照会するために `connection.query` を呼び出します。この例では単純に、サーバーのデータベース名を照会しています。 `query` は、データベース名を `Database` プロパティに割り当てて、各データベースに対応する `results` オブジェクトの配列

を返します。この例では、名前を連結し、レンダリング `app.locals.databases`, されたHTML ページにリストを表示する に名前を割り当てます。

この例では、RDSインスタンスの作成時に指定したデータベースと、Amazon によって自動的に作成される `nodeexampleldb` データベースの 5 つがありますRDS。

[5] Listen for incoming requests (受信されるリクエストをリッスンする)

最後のステートメントは、指定されたポートで受信されるリクエストをリッスンします。明示的なポート値を指定する必要はありません。アプリケーションをスタックに追加するときは、アプリケーションが HTTP または HTTPS リクエストをサポートするかどうかを指定します。AWS OpsWorks スタックは `PORT` 環境変数を 80 (HTTP) または 443 (HTTPS) に設定し、その変数をアプリケーションで使用できます。

他のポートでリッスンすることは可能ですが、Node.js アプリケーションサーバーレイヤーの組み込みセキュリティグループ `AWS-OpsWorks-nodejs-App-Server` では、ポート 80、443、22 () へのインバウンドユーザートラフィックのみが許可されますSSH。他のポートへの着信ユーザートラフィックを許可するには、適切なインバウンドルールを使用して [セキュリティグループを作成し、Node.js アプリケーションサーバーレイヤーに割り当てます](#)。組み込みのセキュリティグループを編集することでインバウンドルールを変更しないでください。スタックを作成するたびに、AWS OpsWorks スタックによって組み込みセキュリティグループが標準設定で上書きされるため、変更を加えると変更内容は失われます。

Note

関連するアプリケーションを [作成](#) または [更新](#) するときに、カスタムの環境変数をアプリケーションに関連付けることができます。カスタムレシピJSONとカスタムレシピを使用して、アプリケーションにデータを渡すこともできます。詳細については、「[アプリケーションへのデータの引き渡し](#)」を参照してください。

データベースサーバーとロードバランサーの作成

この例では、Amazon RDS データベースサーバーと Elastic Load Balancing ロードバランサーインスタンスを使用します。各インスタンスは別々に作成し、スタックに組み込む必要があります。このセクションでは、新しいデータベースとロードバランサーのインスタンスを作成する方法を説明します。既存のインスタンスを使用することもできますが、この手順を読み、そのインスタンスが正しく設定されているか確認することをお勧めします。

以下に、この例で十分である最小限の設定の RDS DB インスタンスを作成する方法について説明します。詳細については、「[Amazon RDSユーザーガイド](#)」を参照してください。

RDS DB インスタンスを作成するには

1. コンソールを開きます。

[Amazon RDSコンソールを開き](#)、リージョンを米国西部 (オレゴン) に設定します。ナビゲーションペインで、RDSダッシュボードを選択し、DB インスタンスを起動を選択します。

2. データベースエンジンを指定します。

データベースエンジンとして MySQL Community Edition を選択します。

3. マルチ AZ 配置を拒否します。

[No, this instance...] を選択し、[Next] を選択します。この例にマルチ AZ 配置は必要ありません。

4. 基本的な設定を指定します。

[DB Instance Details] ページで、次の設定を指定します。

- DB Instance Class: db.t2.micro
- Multi-AZ Deployment: No
- ストレージ割り当て: 5 GB
- DB インスタンス識別子: **nodeexample**
- マスターユーザーの名前: **opsworksuser**
- [Master Password]: 任意のパスワード

後で使用できるようにインスタンス識別子、ユーザー名、およびパスワードを書き留め、その他のオプションについてはデフォルト設定をそのままにして、[Next] を選択します。

5. 詳細設定を指定します。

[Configure Advanced Settings] ページで、以下の設定を指定します。

- データベース名: **nodeexampledb**
- DB セキュリティグループ (複数可) : AWS-OpsWorks-DB-Master-Server

Note

AWS-OpsWorks-DB-Master-Server セキュリティグループでは、スタックのインスタンスのみがデータベースにアクセスできます。データベースに直接アクセスする場合は、適切なインバウンドルールを使用して DB RDS インスタンスに追加のセキュリティグループをアタッチします。詳細については、[「Amazon RDS セキュリティグループ」](#)を参照してください。インスタンスをに配置することでアクセスを制御することもできます VPC。詳細については、[「でのスタックの実行 VPC」](#)を参照してください。

後で使用できるようにデータベース名を書き留め、その他の設定についてはデフォルト値をそのままにして、[Launch DB Instance] を選択します。

以下の手順では、この例の Elastic Load Balancing ロードバランサーを作成する方法を説明します。詳細については、[Elastic Load Balancing ユーザーガイド](#)を参照してください。

ロードバランサーを作成する方法

1. Amazon EC2コンソールを開きます。

[Amazon EC2コンソール](#)を開き、リージョンが米国西部 (オレゴン) に設定されていることを確認します。ナビゲーションペインで、[Load Balancers] を選択し、[Create Load Balancer] を選択します。

2. ロードバランサーを定義します。

[Define Load Balancer] ページで、以下の設定を指定します。

- 名前 – **Node-LB**
- 内部で LB を作成する – デフォルト VPC

その他のオプションについてはデフォルト設定をそのままにして、[Next] を選択します。

3. セキュリティグループを割り当てます。

[Assign Security Groups] ページで、以下のグループを指定します。

- デフォルトのVPCセキュリティグループ

- AWS-OpsWorks-nodejs-App-Server

[Next (次へ)] を選択します。[Configure Security Settings] ページで、[Next] を選択します。この例にセキュアリスナーは必要ありません。

4. ヘルスチェックを設定します。

[Configure Health Check (ヘルスチェックの設定)] ページで、[Ping Path (ping パス)] を / に設定し、その他の設定についてはデフォルト値をそのままにします。[Next (次へ)] を選択します。EC2 インスタンスの追加ページで、次へを選択します。タグの追加ページで、レビューと作成を選択します。AWS OpsWorks スタックはロードバランサーにEC2インスタンスを追加するタスクを処理するため、この例のタグは必要ありません。

5. ロードバランサーを作成します。

[Review] ページで、[Create] を選択してロードバランサーを作成します。

スタックの作成

必要なコンポーネントがすべて揃ったので、スタックを作成できます。

スタックを作成するには

1. AWS OpsWorks スタックコンソールにサインインします。

[AWS OpsWorks スタックコンソール](#)にサインインし、[Add Stack (スタックを追加)] を選択します。

2. スタックを作成します。

新しいスタックを作成するには、[Chef 11 stack] を選択して、以下の設定を指定します。

- – **NodeStack**
- [リージョン] – 米国西部 (オレゴン)

スタックは任意のAWSリージョンで作成できますが、チュートリアルでは米国西部 (オレゴン) をお勧めします。

[Add Stack] を選択します。スタック設定の詳細については、「[新しいスタックを作成する](#)」を参照してください。

3. ロードバランサーがアタッチされた Node.js App Server layer を追加します。

NodeStack ページで、レイヤーの追加を選択し、次の設定を指定します。

- Layer type (レイヤータイプ) – Node.js App Server (Node.js アプリケーションサーバー)
- Elastic Load Balancer (Elastic ロードバランサー) – Node-LB (ノード-LB)

その他の設定についてはデフォルト値をそのままにして、[Add Layer] を選択します。

4. インスタンスをレイヤーに追加して起動します。

ナビゲーションペインで、[Instances] を選択し、次の手順に従って 2 つのインスタンスを Rails アプリケーションサーバーレイヤーに追加します。

1. Node.js App Server (Node.js アプリケーションサーバー) で、Add instance (インスタンスの追加) を選択します。

[Size] を [t2.micro] に設定し、その他の設定についてはデフォルト値をそのままにして、[Add Instance] を選択します。

2. [+Instance] を選択し、2 番目の t2.micro インスタンスを別のサブネット内のレイヤーに追加します。

これにより、インスタンスが別のアベイラビリティゾーン (AZ) に配置されます。

3. [Add instance] を選択します。

4. 両方のインスタンスを起動するには、[Start All Instances] を選択します。

Elastic Load Balancing ロードバランサーをこのレイヤーに割り当てました。インスタンスがオンライン状態に出入りすると、AWS OpsWorks スタックはロードバランサーにインスタンスを自動的に登録または登録解除します。

Note

本番稼働用スタックでは、アプリケーションサーバーインスタンスを複数の AZs に分散することをお勧めします。分散しておくことで、ユーザーが 1 つの AZ に接続できない場合、着信トラフィックがロードバランサーによって別のゾーンにルーティングされるため、サイトの機能が続行します。

5. RDS DB インスタンスをスタックに登録します。

ナビゲーションペインで、リソースを選択し、次のように RDS DB インスタンスをスタックに登録します。

1. RDS タブを選択し、未登録RDSの DB インスタンスを表示を選択します。
2. nodeexampleddb インスタンスを選択し、以下の設定を指定します。
 - User (ユーザー) – インスタンスの作成時に指定したマスターユーザー名 (この例では、) **opsworksuser**。
 - [Password] (パスワード) – インスタンスの作成時に指定したマスターパスワード。
3. スタックに登録を選択して、RDSDB インスタンスを [Amazon RDSサービスレイヤー](#) としてスタックに追加します。

Warning

AWS OpsWorks スタックはユーザーまたはパスワードの値を検証せず、単にアプリケーションに渡します。これらを誤って入力すると、アプリケーションがデータベースに接続できません。

RDS DB インスタンスを [Amazon RDSサービスレイヤー](#) としてスタックに追加するには、スタックに登録を選択します。

アプリケーションのデプロイ

アプリケーションはリモートリポジトリに保存する必要があります。デプロイすると、AWS OpsWorks スタックはコードと関連ファイルをリポジトリからアプリケーションサーバーインスタンスにデプロイします。便宜上、この例では、パブリック Amazon Simple Storage Service (Amazon S3) アーカイブをリポジトリとして使用していますが、Git や Subversion など、いくつか他の種類のリポジトリも使用できます。詳細については、「[Application Source](#)」を参照してください。

アプリケーションをデプロイするには

1. アプリケーションをアーカイブファイルにパッケージ化します。

.zip ディレクトリおよびサブディレクトリの nodedb アーカイブを nodedb.zip という名前で作成します。gzip、bzip2、tarball など、他の種類のアーカイブファイルを使用することもでき

ます。AWS OpsWorks スタックは非圧縮 tarball をサポートしていないことに注意してください。詳細については、「[Application Source](#)」を参照してください。

2. アーカイブファイルを Amazon S3 にアップロードします。

Amazon S3 バケット `nodedb.zip` にアップロードし、ファイルを公開して、URL 後で使用するためにファイルの をコピーします。バケットの作成方法とファイルのアップロード方法の詳細については、「[Amazon Simple Storage Service の使用開始](#)」を参照してください。

Note

AWS OpsWorks スタックは Amazon S3 バケットからプライベートファイルをデプロイすることもできますが、わかりやすくするために、この例ではパブリックファイルを使用します。詳細については、「[Application Source](#)」を参照してください。

3. AWS OpsWorks スタックアプリケーションを作成します。

AWS OpsWorks スタックコンソールに戻り、ナビゲーションペインでアプリを選択し、アプリの追加を選択します。以下の設定を指定します。

- [Name] (名前) – NodeDB。

この文字列は、アプリケーションの表示名です。ほとんどの場合、アプリケーションの短縮名が必要です。短縮名は、すべての文字を小文字に変換し、句読点を削除することで AWS OpsWorks、スタックが表示名から生成します。この例の場合、短縮名は `nodedb` になります。アプリケーションの短縮名を確認するには、アプリケーションの作成後、[Apps] ページでアプリケーションを選択して詳細ページを表示します。

- Type (タイプ) – Node.js。
- Data source type (データソースタイプ) – RDS。
- データベースインスタンス – 以前に登録した Amazon RDS DB インスタンスを選択します。
- Database name (データベース名) - 作成しておいたデータベースの名前を指定します (この例の場合は `nodeexampledb`)。
- Repository type (リポジトリタイプ) - Http Archive。

パブリック Amazon S3 ファイルには、このリポジトリタイプを使用する必要があります。S3 Archive は、プライベートアーカイブにのみ使用する種類です。

- リポジトリ URL – アーカイブファイルの Amazon S3 URL。

残りの設定にはデフォルト値を使用し、[Add App] をクリックしてアプリケーションを作成します。

4. アプリケーションをデプロイします。

[Apps] ページに移動し、NodeDB アプリケーションの [Actions] 列で、[deploy] を選択します。次に、デプロイを選択してアプリケーションをサーバーインスタンスにデプロイします。AWS OpsWorks スタックは各インスタンスでデプロイレシピを実行します。これにより、リポジトリからアプリケーションがダウンロードされ、サーバーが再起動されます。各インスタンスに緑色のチェックマークが付き、[Status] が [successful] であれば、デプロイが完了しており、アプリケーションでリクエストの処理を開始できる状態になっています。

Note

デプロイが失敗した場合は、[Log] (ログ) 列で [show] (表示) を選択すると、デプロイメントの Chef ログが表示されます。エラー情報は下の方にあります。

5. アプリケーションを開きます。

アプリケーションを開くには、レイヤーを選択し、ロードバランサーを選択してから、ロードバランサーに HTTP リクエストを送信するロードバランサー DNS の名前を選択します。次のような内容が表示されます。

AWS OpsWorks Node.js Example

Amazon RDS Endpoint: *nodeexample.cdlq1k5uwd0k.us-west-2.rds.amazonaws.com*

User: *opsworksuser*

Password: *Your-Pwd*

Port: *3306*

Database: *nodeexampledb*

Connection: *successful*

Databases: *information_schema, innodb, mysql, nodeexampledb, performance_schema*

Note

AWS OpsWorks スタックは、セットアップ中に新しいインスタンスにアプリケーションを自動的にデプロイします。オンラインインスタンスに対してのみ、手動のデプロイが必要になります。詳細については、「[アプリケーションのデプロイ](#)」を参照してください。高度なデプロイ方法など、デプロイ全般については、「[アプリケーションとクックブックの管理とデプロイ](#)」を参照してください。

次のステップ

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

このウォークスルーでは、シンプルな Node.js アプリケーションサーバースタックのセットアップの基本について説明しました。ここでは、次のステップに関する推奨事項をいくつか示します。

Node.js の組み込みクックブックを調べる

インスタンスの設定方法について詳しく知りたい場合は、レイヤーの組み込みクックブックである [opsworks_nodejs](#) を参照してください。このクックブックには、AWS OpsWorks スタックがソフトウェアのインストールと設定に使用するレシピおよび関連ファイル、および AWS OpsWorks スタックがアプリケーションのデプロイに使用するレシピを含む組み込みデプロイ [クックブック](#) が含まれています。

サーバー設定をカスタマイズする

例では、基本的なスタックを使用しています。本稼働用に使用する場合は、スタックをカスタマイズすることをお勧めします。詳細については、「[AWS OpsWorks スタックのカスタマイズ](#)」を参照してください。

SSL サポートの追加

アプリの作成時に、アプリSSLのサポートを有効にし、適切な証明書を AWS OpsWorks スタックに提供できます。AWS OpsWorks スタックは、適切なディレクトリに証明書をインストールします。詳細については、「[SSL の使用](#)」を参照してください。

メモリ内データのキャッシュを追加する

本稼働レベルのサイトでは、多くの場合、Redis や Memcache などのメモリ内キー値ストアにデータをキャッシュすることでパフォーマンスが向上します。AWS OpsWorks スタックスタックでは、のいずれかを使用できます。詳細については、[ElastiCache Redis](#)および[Memcached](#)を参照してください。

高度なデプロイ方法を使用する

ここに示した例では、すべてのインスタンスに更新を同時にデプロイするというシンプルな方法でアプリをデプロイしました。このアプローチは、シンプルで高速ですが、エラーに対する許容範囲がありません。デプロイが失敗した場合や、更新に問題がある場合、本稼働スタックのすべてのインスタンスが影響を受けることがあります。問題が解決されるまで、サイトのサービスが中断されるか無効になる可能性があります。デプロイ方法の詳細については、「[アプリケーションとクックブックの管理とデプロイ](#)」を参照してください。

Node.js アプリケーションサーバーレイヤーを拡張する

レイヤーは、さまざまな方法で拡張できます。たとえば、インスタンスに対してスクリプトを実行するためのレシピや、アプリのデプロイをカスタマイズするための Chef デプロイフックを実装することができます。詳細については、「[レイヤーの拡張](#)」を参照してください。

環境変数を定義する

関連付けられたアプリケーションの環境変数を定義することによって、アプリケーションにデータを渡すことができます。アプリケーションをデプロイすると、AWS OpsWorks スタックはこれらの変数をエクスポートして、アプリケーションからアクセスできるようにします。詳細については、「[環境変数の使用](#)」を参照してください。

AWS OpsWorks スタックのカスタマイズ

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックの組み込みレイヤーは、多くの目的に十分な標準機能を提供します。ただし、以下の状況が考えられます。

- 組み込みのレイヤーの標準設定では最適とは言えない場合、特定の要件に合わせて最適化することができます。

例えば、ワーカプロセスの最大数または `keepalivetimeout` 値などの設定に独自の値を指定することで、Static ウェブサーバーレイヤーの Nginx サーバー設定を調整することができます。

- 組み込みのレイヤー機能でも問題ありませんが、追加パッケージをインストールまたはカスタムのインストールスクリプトを実行することで拡張することができます。

例えば、Redis サーバーをインストールして PHP App Server レイヤーを拡張できます。

- 組み込みのレイヤーでは扱うことのできない要件があります。

例えば、AWS OpsWorks スタックには、一般的なデータベースサーバー用の組み込みレイヤーは含まれていません。レイヤーのインスタンスに、それらのサーバーをインストールするカスタムレイヤーを作成することができます。

- カスタムレイヤーのみをサポートする Windows スタックを実行しています。

AWS OpsWorks スタックには、特定の要件を満たすようにレイヤーをカスタマイズするさまざまな方法が用意されています。次の例は、複雑さと能力の低い方から高い方の順に表示されています。

Note

これらのアプローチのいくつかは Linux スタックにのみ当てはまります。詳細については、以下のトピックを参照してください。

- カスタム JSON を使用して、デフォルトの AWS OpsWorks スタック設定を上書きします。
- デフォルトのスタック設定を上書きする属性ファイルを使用して、カスタム Chef AWS OpsWorks クックブックを実装します。
- デフォルトのスタックテンプレートを上書きまたは拡張するテンプレートを使用して、カスタム Chef AWS OpsWorks クックブックを実装します。

- シェルスクリプトを実行するシンプルなレシピが含まれるカスタムの Chef クックブックを実行します。
- ディレクトリの作成と設定、パッケージのインストール、設定ファイルの作成、アプリケーションのデプロイなどのタスクを実行するレシピを含むカスタムの Chef クックブックを実装します。

スタックの Chef バージョンおよびオペレーティングシステムに応じて、レシピを上書きすることもできます。

- Chef 0.9 および 11.4 スタックでは、クックブックやレシピが同じ名前のカスタムレシピを実装して組み込みのレシピを上書きすることはできません。

ライフサイクルイベントごとに、AWS OpsWorks スタックは常に組み込みレシピを最初に実行し、その後にカスタムレシピを実行します。これらの Chef バージョンでは、クックブックやレシピが同じ名前のレシピが 2 度実行されないため、組み込みレシピが優先され、カスタムレシピは実行されません。

- Chef 11.10 スタックの組み込みレシピは上書きできます。

詳細については、「[クックブックのインストールと優先順位](#)」を参照してください。

- Windows スタックの組み込みレシピは上書きできません。

AWS OpsWorks スタックが Windows スタックの Chef 実行を処理する方法では、組み込みレシピを上書きすることはできません。

Note

多くの手法はカスタムクックブックを使用するため、クックブックの実装にまだ慣れず [クックブックとレシピ](#) がない場合は、まず「」をお読みください。 [クックブックの基本](#) では、カスタムクックブックの実装に関する詳細なチュートリアルを紹介し、AWS OpsWorks スタックインスタンスのクックブックの実装方法の詳細をいくつか [AWS OpsWorks スタックのクックブックの実装](#) 説明します。

トピック

- [属性を上書きして AWS OpsWorks スタック設定をカスタマイズする](#)
- [カスタムテンプレートを使用した AWS OpsWorks スタック設定ファイルの拡張](#)
- [レイヤーの拡張](#)

- [カスタム Tomcat サーバーレイヤーの作成](#)
- [スタック設定およびデプロイメント属性](#)

属性を上書きして AWS OpsWorks スタック設定をカスタマイズする

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

Windows スタックと Chef 12 Linux スタックの場合、AWS OpsWorks スタックは組み込みレシピとカスタムレシピに別々の Chef 実行を使用します。つまり、このセクションで説明している手法を使用して、Windows スタックおよび Chef 12 Linux スタック用の組み込み属性を上書きすることはできません。

レシピとテンプレートは、レイヤーの設定やアプリケーションサーバーの設定など、インスタスやスタックに固有の情報に対応するさまざまな Chef 属性に依存します。これらの属性には複数のソースがあります。

- カスタム JSON- オプションで、スタックの作成、更新、クローン作成、またはアプリケーションのデプロイ時にカスタムJSON属性を指定できます。
- スタック設定属性 -AWS OpsWorks スタックは、コンソール設定で指定した情報など、スタック設定情報を保持するためにこれらの属性を定義します。
- デプロイ属性 - デプロイイベントのデプロイ関連の属性AWS OpsWorks を定義します。
- Cookbook attributes (クックブックの属性) - 組み込みのクックブックやカスタムクックブックには、通常、1 つ以上の [attribute files](#) (属性ファイル) が含まれており、属性ファイルにはアプリケーションサーバー設定などのクックブック固有の値を表す属性が含まれます。
- Chef-Chef の [Ohai ツール](#)は、CPUタイプやインストールされたメモリなど、さまざまなシステム構成設定を表す属性を定義します。

スタック設定属性、デプロイ属性、組み込みクックブック属性のリストについては、「[スタック設定およびデプロイ属性: Linux](#)」および「[組み込みクックブックの属性](#)」を参照してください。Ohai の属性の詳細については、「[Ohai](#)」を参照してください。

Deploy や Configure などの [lifecycle event](#) (ライフサイクルイベント) が発生したり、execute_recipes または update_packages などの [stack command](#) (スタックコマンド) を実行すると、AWS OpsWorks スタックは次のように動作します。

- 対応するコマンドを、影響を受ける各インスタンスのエージェントに送信します。

エージェントは適切なレシピを実行します。たとえば、Deploy イベントの場合、エージェントは組み込みの Deploy レシピを実行した後、カスタム Deploy レシピを実行します。

- カスタム属性JSONとデプロイ属性をスタック設定属性とマージし、インスタンスにインストールします。

カスタムの属性JSON、スタック設定およびデプロイ属性、クックブック属性、および Ohai 属性はノードオブジェクトにマージされ、レシピに属性値を提供します。インスタンスは、カスタムを含むスタック設定属性に関しては基本的にステートレスですJSON。デプロイまたはスタックコマンドを実行すると、関連付けられたレシピでは、そのコマンドでダウンロードされたスタック設定属性が使用されます。

トピック

- [属性の優先順位](#)
- [カスタムによる属性の上書き JSON](#)
- [カスタムクックブック属性を使用した AWS OpsWorks スタック属性の上書き](#)

属性の優先順位

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

属性を一意に定義した場合、Chef によってその属性はノードオブジェクトに組み込まれるだけです。ただし、任意の属性ソースが任意の属性を定義できるため、同じ属性に複数の定義が存在し、値が異なる可能性があります。例えば、組み込みクックブック `apache2` は `node[:apache][:keepalive]` を定義しますが、その属性をカスタムクックブック JSON またはカスタムクックブックで定義することもできます。属性に複数の定義がある場合、後で説明する順序に従って定義が評価され、最も優先順位の高い定義がノードオブジェクトに組み込まれます。

属性は次のように定義されます。

```
node.type[:attribute][:sub_attribute][:...] = value
```

属性に複数の定義がある場合、タイプによってどの定義が優先されるかが決まり、その定義がノードオブジェクトに組み込まれます。AWS OpsWorks スタックは次の属性タイプを使用します。

- **default (デフォルト)** - これは最も一般的なタイプであり、基本的に「この属性がまだ定義されていない場合はこの値を使用する」ことを意味します。属性のすべての定義が default タイプである場合、評価順序の最初の定義が優先され、それ以降の値は無視されます。AWS OpsWorks スタックは、すべてのスタック設定およびデプロイ属性定義を default 型に設定することに注意してください。
- **normal (通常)** - このタイプの属性は、評価順序で以前に定義されているすべての default または normal 属性を上書きします。たとえば、組み込みクックブックの最初の属性が default タイプであり、2 番目が normal タイプのユーザー定義属性である場合、2 番目の定義が優先されます。
- **set (セット)** - これは、古いクックブックに表示される可能性のある非推奨のタイプです。このタイプは、優先順位が同じである normal に置き換えられました。

Chef は、他のすべての属性定義よりも優先される **automatic** タイプなど、いくつかの追加属性タイプをサポートしています。Chef の Ohai ツールによって生成された属性定義は、すべて automatic タイプであるため、事実上読み取り専用です。これは通常問題ではありません。上書きする理由がなく、AWS OpsWorks スタックの属性とは異なるためです。ただし、カスタムクックブック属性の名前を指定するときには、Ohai 属性とは異なる名前になるように注意する必要があります。詳細については、「[属性について](#)」を参照してください。

Note

Ohai ツールは、コマンドラインから実行できる実行可能ファイルです。インスタンスの Ohai 属性のリストを取得するには、ターミナルウィンドウでインスタンスにログイン

し、`ohai` を実行します。それによって非常に長い出力が生成することに注意してください。

ノードオブジェクトにさまざまな属性定義を組み込む手順を以下に示します。

1. カスタムスタック設定属性をスタック設定属性とデプロイ属性にマージします。

カスタムJSON属性は、スタックまたは特定のデプロイに対して設定できます。これらの属性は最初に評価されるため、事実上 `normal` タイプです。1 つ以上のスタック設定属性がカスタムでも定義されている場合JSON、カスタムJSON値が優先されます。それ以外の場合、AWS OpsWorks スタックはカスタムJSON属性をスタック設定に単純に組み込むだけです。

2. デプロイカスタムJSON属性をスタック設定およびデプロイ属性にマージします。

デプロイカスタムJSON属性も効果的に `normal` タイプされるため、組み込みおよびカスタムスタック設定JSONや組み込みデプロイよりも優先されますJSON。

3. インスタンスのノードオブジェクトにスタック設定およびデプロイ属性をマージします。

4. インスタンスの組み込みクックブック属性をノードオブジェクトにマージします。

組み込みクックブック属性はすべて `default` タイプです。1 つ以上の組み込みクックブック属性がスタック設定およびデプロイ属性でも定義されている場合、通常はカスタムで定義されているためJSON、スタック設定定義は組み込みクックブック定義よりも優先されます。他の組み込みクックブック属性はすべて、単にノードオブジェクトに組み込まれます。

5. インスタンスのカスタムクックブック属性をノードオブジェクトにマージします。

カスタムクックブック属性は、通常 `normal` タイプまたは `default` タイプです。一意の属性はノードオブジェクトに組み込まれます。ステップ 1~3 でカスタムクックブック属性も定義されている場合 (通常はカスタム で定義した場合JSON)、優先順位はカスタムクックブック属性のタイプによって異なります。

- ステップ 1~3 で定義された属性は、カスタムクックブック `default` 属性よりも優先されません。
- カスタムクックブック `normal` 属性は、ステップ 1~3 の定義よりも優先されます。

⚠ Important

スタック設定属性または組み込みクックブック属性を上書きするために、カスタムクックブックの default 属性を使用しないでください。カスタムクックブックの属性は最後に評価されるため、default 属性の優先順位は最も低く、何も上書きすることができません。

カスタムによる属性の上書き JSON

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

i Note

スタックは、Windows AWS OpsWorks スタックと Linux スタックでは Chef の実行が異なるため、このセクションで説明する手法を Windows スタックで使用することはできません。

AWS OpsWorks スタック属性を上書きする最も簡単な方法は、カスタムで定義することです。これは JSON、スタック設定およびデプロイ属性だけでなく、組み込みおよびカスタムクックブック default 属性よりも優先されます。詳細については、「[属性の優先順位](#)」を参照してください。

⚠ Important

スタック設定およびデプロイ属性を上書きする場合は注意が必要です。例えば、opsworks 名前空間の属性を上書きすると、組み込みのレシピに問題が生じる可能性があります。詳細については、「[スタック設定およびデプロイメント属性](#)」を参照してください。

カスタム を使用して一意の属性 JSON を定義することもできます。通常はカスタムレシピにデータを渡すためです。属性は単にノードオブジェクトに組み込まれ、レシピは標準の Chef ノード構文を使用してオブジェクトを参照できます。

カスタム を指定する方法 JSON

カスタム を使用して属性値をJSON上書きするには、まず属性の完全修飾属性名を決定する必要があります。次に、上書きする属性を含む JSON オブジェクトを作成し、を任意の値に設定します。わかりやすくするために、「[スタック設定およびデプロイ属性: Linux](#)」および「[組み込みクックブックの属性](#)」に、一般的に使用されるスタック設定、デプロイ、および組み込みのクックブックの属性、およびそれらの完全修飾名がまとめられています。

オブジェクトの親子関係は、適切な完全修飾 Chef ノードに対応している必要があります。たとえば、Apache の次の属性を変更するとします。

- ノードが `node[:apache][:keepalivetimeout]` であり、デフォルト値が 3 である [keepalivetimeout](#) 属性。
- ノードが `node[:apache][:logrotate][:schedule]` であり、デフォルト値が "daily" である `logrotate` [schedule](#) 属性。

属性を上書きし "weekly"、値をそれぞれ 5 と に設定するには、次のカスタム を使用しますJSON。

```
{
  "apache" : {
    "keepalivetimeout" : 5,
    "logrotate" : {
      "schedule" : "weekly"
    }
  }
}
```

カスタム を指定するタイミング JSON

次のタスクのカスタムJSON構造を指定できます。

- [新しいスタックを作成する](#)
- [スタックを更新する](#)
- [スタックコマンドを実行する](#)
- [スタックをクローン化する](#)
- [アプリケーションをデプロイする](#)

AWS OpsWorks スタックは、タスクごとにカスタムJSON属性をスタック設定およびデプロイ属性とマージし、インスタンスに送信してノードオブジェクトにマージします。ただし、以下の点に注意してください。

- スタックの作成、クローン作成、または更新JSON時にカスタム を指定した場合、属性は後続のすべてのライフサイクルイベントとスタックコマンドのスタック設定およびデプロイ属性にマージされます。
- デプロイJSONにカスタム を指定すると、属性は対応するイベントに対してのみスタック設定とデプロイ属性にマージされます。

これらのカスタム属性を後続のデプロイに使用する場合は、カスタムを再度明示的に指定する必要がありますJSON。

レシピによって使用される場合、属性はインスタンスにのみ影響することを覚えておく必要があります。属性値を上書きしても、それ以降のレシピでその属性が参照されない場合、その変更は有効ではありません。関連するレシピを実行する前にカスタムJSONが送信されていることを確認するか、適切なレシピが再実行されることを確認する必要があります。

カスタムJSONベストプラクティス

カスタム を使用して任意の AWS OpsWorks スタック属性JSONを上書きできますが、手動での情報の入力は多少面倒であり、ソース管理の対象ではありません。Custom JSONは以下の目的に最適です。

- 少数の属性のみを上書きする必要があり、その他の目的でカスタムクックブックを使用する必要がない場合。

カスタム を使用するとJSON、クックブックリポジトリをセットアップして維持するオーバーヘッドを回避して、いくつかの属性を上書きできます。

- パスワードや認証キーなどの機密情報を含む値。

クックブック属性はリポジトリに保存されるため、どの機密情報にも何らかの漏洩のリスクがあります。代わりに、ダミー値で属性を定義しJSON、カスタム を使用して実際の値を設定します。

- 異なることが予想される値。

たとえば、本稼働用スタックを別の開発およびステージングスタックでサポートする方法が推奨されています。これらのスタックが、支払いを受け付けるアプリケーションをサポートしているとします。カスタム を使用して支払いエンドポイントJSONを指定する場合は、ステージングスタック

クURLのテストを指定できます。更新されたスタックを本番稼働用スタックに移行する準備ができたなら、同じクックブックを使用し、カスタム を使用して支払いエンドポイントを本番稼働用にJSON設定できますURL。

- 特定のスタックまたはデプロイコマンドに固有の値。

カスタムクックブック属性を使用した AWS OpsWorks スタック属性の上書き

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

Windows スタックの場合、AWS OpsWorks スタックは組み込みレシピとカスタムレシピに別々の Chef 実行を使用します。つまり、このセクションで説明している手法を使用して、Windows スタック用の組み込み属性を上書きすることはできません。

カスタム JSON は、AWS OpsWorks スタックスタック設定と組み込みクックブック属性を上書きする便利な方法ですが、いくつかの制限があります。特に、定義を管理する堅牢な方法がないため、使用ごとにカスタムJSONを手動で入力する必要があります。よりよい方法として、カスタムクックブック属性ファイルを使用して組み込みの属性を上書きする方法があります。これによって、定義をソース管理の下に置くことができます。

カスタム属性ファイルを使用して AWS OpsWorks スタック定義を上書きする手順は簡単です。

AWS OpsWorks スタック属性定義を上書きするには

1. 「[クックブックとレシピ](#)」の説明に従ってクックブックリポジトリをセットアップします。
2. 上書きする属性を含む組み込みクックブックと同じ名前で作成します。たとえば、Apache の属性を上書きするには、カスタムクックブックの名前を apache2 にする必要があります。

3. クックブックに `attributes` フォルダを追加し、そのフォルダに `customize.rb` という名前のファイルを追加します。
4. 上書きする組み込みクックブックの属性ごとに、優先する値に設定された属性定義をファイルに追加します。属性は `normal`、`タイプ` 以上で、対応する AWS OpsWorks スタック属性とノード名が完全に同じである必要があります。ノード名を含む AWS OpsWorks スタック属性の詳細なリストについては、[スタック設定およびデプロイ属性: Linux](#) 「」および「」を参照してください。[組み込みクックブックの属性](#)。属性および属性ファイルの詳細については、「[属性ファイルについて](#)」を参照してください。

Important

スタック属性を上書きするには、属性が AWS OpsWorks `normal` 型である必要があります。 `default` 型が優先されません。例えば、 `customize.rb` ファイルに `default[:apache][:keepalivetimeout] = 5` という属性定義がある場合、組み込みの `apache.rb` 属性ファイル内の対応する属性が最初に評価され、優先されます。詳細については、「[属性の上書き](#)」を参照してください。

5. 上書きする属性を持つ組み込みクックブックごとに、ステップ 2~4 を繰り返します。
6. スタックのカスタムクックブックを有効にし、AWS OpsWorks スタックがクックブックをスタックのインスタンスにダウンロードするために必要な情報を提供します。詳細については、「[カスタムクックブックのインストール](#)」を参照してください。

Note

この手順の詳細なウォークスルーについては、「[組み込み属性の上書き](#)」を参照してください。

後続のライフサイクルイベント、デプロイコマンド、スタックコマンドで使用されるノードオブジェクトに、AWS OpsWorks スタック値ではなく属性定義が含まれるようになりました。

たとえば、「`keepalivetimeout`」で説明した、組み込みの `logrotate schedule` および [カスタムを指定する方法 JSON](#) の設定を上書きするには、`apache2apache` クックブックをリポジトリに追加し、次のような内容の `customize.rb` ファイルをクックブックの `attributes` フォルダに追加します。

```
normal[:apache][:keepalivetimeout] = 5
normal[:apache][:logrotate][:schedule] = 'weekly'
```

⚠ Important

関連する組み込み属性ファイルのコピーを変更して、AWS OpsWorks スタック属性を上書きしないでください。例えば、`apache.rb` を `apache2/attributes` フォルダにコピーし、その設定の一部を変更すると、基本的に組み込みファイル内のすべての属性が上書きされます。レシピでは、コピーの属性定義が使用され、組み込みファイルは無視されます。AWS OpsWorks スタックによって後で組み込みの属性ファイルが変更された場合、手動でコピーを更新しない限り、レシピでは変更内容にアクセスできません。

このような状況を回避するために、組み込みのすべてのクックブックには空の `customize.rb` 属性ファイルが含まれています。このファイルは、`include_attribute` デイレクティブを介してすべてのモジュールが必要です。`customize.rb` のコピーで属性を上書きすることによって、その特定の属性にのみ変更を反映できます。レシピは組み込みの属性ファイルから他の属性値を取得し、上書きしなかった属性の現在の値を自動的に取得します。

このアプローチでは、クックブックリポジトリ内の属性の数を少数に抑えることができます。これにより、メンテナンスのオーバーヘッドが削減され、将来のアップグレードの管理が容易になります。

カスタムテンプレートを使用した AWS OpsWorks スタック設定ファイルの拡張

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

ℹ Note

スタックは、Windows AWS OpsWorks スタックと Linux スタックでは Chef の実行が異なるため、このセクションで説明する手法を Windows スタックで使用することはできません。

AWS OpsWorks スタックはテンプレートを使用して設定ファイルなどのファイルを作成します。通常、このファイルは多くの設定の属性に依存します。カスタムJSONまたはカスタムクックブック属性を使用して AWS OpsWorks スタック定義を上書きする場合、優先設定は AWS OpsWorks スタック設定の代わりに設定ファイルに組み込まれます。ただし、AWS OpsWorks スタックでは、可能なすべての設定に属性を指定するとは限りません。一部の設定のデフォルトを受け入れ、テンプレート内で直接他の設定をハードコードします。対応する AWS OpsWorks スタック属性がない場合、カスタムJSONまたはカスタムクックブック属性を使用して優先設定を指定することはできません。

カスタムテンプレートを作成することによって、追加の設定を含めるように設定ファイルを拡張できます。次に、必要な任意の設定またはその他のコンテンツをファイルに追加し、ハードコーディングされた設定を変更できます。テンプレートの詳細については「[テンプレート](#)」を参照してください。

Note

opsworks-agent.monitrc.erb を除く組み込みのテンプレートを変更できます。

カスタムテンプレートを作成するには

1. 組み込みクックブックと同じ構造とディレクトリ名のクックブックを作成します。次に、カスタマイズする組み込みのテンプレートと同じ名前と、適切なディレクトリにテンプレートファイルを作成します。たとえば、カスタムテンプレートを使用して Apache の httpd.conf 設定ファイルを拡張するには、リポジトリに apache2 クックブックを実装し、テンプレートファイルが apache2/templates/default/apache.conf.erb である必要があります。まったく同じ名前を使用すると、AWS OpsWorks スタックはカスタムテンプレートを認識し、組み込みテンプレートの代わりにそれを使用できます。

最も簡単な方法は、組み込みテンプレートファイルを[組み込みクックブック GitHubのリポジトリ](#)からクックブックにコピーし、必要に応じて変更することです。

Important

カスタマイズするテンプレートファイル以外のファイルを、組み込みクックブックからコピーしないでください。レシピなど、他の種類のクックブックファイルをコピーすると、重複する Chef リソースが作成され、エラーの原因となる可能性があります。

クックブックにもカスタム属性、レシピ、関連ファイルを含めることができますが、そのファイル名は組み込みのファイル名と重複しないようにする必要があります。

2. テンプレートファイルをカスタマイズして、要件を満たす設定ファイルを作成します。設定の追加、既存の設定の削除、ハードコーディングされた属性の置換などを行うことができます。
3. カスタムクックブックをまだ有効にしてない場合は、スタック設定を編集してカスタムクックブックを有効にして、クックブックリポジトリを指定してください。詳細については、「[カスタムクックブックのインストール](#)」を参照してください。

Note

この手順の詳細なウォークスルーについては、「[組み込みテンプレートの上書き](#)」を参照してください。

テンプレートを上書きするために、レシピを実装したり、[レイヤー設定にレシピを追加](#)したりする必要はありません。AWS OpsWorks スタックは常に組み込みレシピを実行します。設定ファイルを作成するレシピが実行される際に、自動的に組み込みのテンプレートの代わりにカスタムテンプレートが使用されます。

Note

AWS OpsWorks スタックが組み込みテンプレートに変更を加えると、カスタムテンプレートが同期しなくなり、正しく動作しなくなる可能性があります。たとえば、テンプレートが依存ファイルを参照し、ファイル名が変更されるとします。AWS OpsWorks スタックはそのような変更を頻繁に行うことはありません。テンプレートが変更されると、変更が一覧表示され、新しいバージョンにアップグレードするオプションが提供されます。AWS OpsWorks スタックリポジトリの変更をモニタリングし、必要に応じてテンプレートを手動で更新する必要があります。

レイヤーの拡張

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックの属性を変更するか、テンプレートをカスタマイズすることで処理できる範囲を拡大し、組み込みレイヤーをカスタマイズすることが必要な場合があります。たとえば、シンボリックリンクの作成、ファイルまたはフォルダのモードの設定、追加パッケージのインストールなどを行う必要があるとします。最小限の機能に加えて他の機能も提供するには、カスタムレイヤーを拡張する必要があります。その場合、カスタマイズタスクを処理する 1 つ以上のカスタムクックブックを実装する必要があります。このトピックでは、レシピを使用してレイヤーを拡張する方法を例で示します。

Chef を初めて使う場合は、最初に「[クックブック 101](#)」をお読みください。これは、クックブックを実装してさまざまな一般的なタスクを実行する方法の基本事項を紹介するチュートリアルです。カスタムレイヤーの実装方法の詳細な例については、「[カスタム Tomcat サーバーレイヤーの作成](#)」を参照してください。

トピック

- [レシピを使用したスクリプトの実行](#)
- [Chef デプロイブックの使用](#)
- [Linux インスタンスでの cron ジョブの実行](#)
- [Linux インスタンスでパッケージをインストールおよび設定](#)

レシピを使用したスクリプトの実行

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

必要なカスタマイズタスクを実行するスクリプトがすでにある場合、通常、レイヤーを拡張する最も簡単な方法は、スクリプトを実行する簡単なレシピを実装することです。その後、適切なライフサイ

クックブック (通常は Setup または Deploy) にレシピを割り当てることも、`execute_recipes` スタックコマンドを使用してレシピを手動で実行することもできます。

次の例では、Linux インスタンスでシェルスクリプトを実行しますが、Windows スクリプトを含む他のタイプの PowerShell スクリプトにも同じアプローチを使用できます。

```
cookbook_file "/tmp/lib-installer.sh" do
  source "lib-installer.sh"
  mode 0755
end

execute "install my lib" do
  command "sh /tmp/lib-installer.sh"
end
```

`cookbook_file` リソースは、クックブックの `files` ディレクトリのサブディレクトリに保存されたファイルを表します。このファイルはインスタンスの指定した場所に転送されます。この例では、シェルスクリプト `lib-installer.sh` をインスタンスの `/tmp` ディレクトリに転送し、ファイルのモードを `0755` に設定します。詳細については、「[cookbook_file](#)」を参照してください。

`execute` リソースは、シェルコマンドなどのコマンドを表します。この例では `lib-installer.sh` を実行します。詳細については、「[execute](#)」を参照してください。

スクリプトをレシピに組み込んで実行することもできます。次の例では Bash スクリプトを実行しますが、Chef では Csh、Perl、Python、Ruby もサポートしています。

```
script "install_something" do
  interpreter "bash"
  user "root"
  cwd "/tmp"
  code <<-EOH
  #insert bash script
  EOH
end
```

`script` リソースはスクリプトを表します。この例では Bash インタープリタを指定し、ユーザーを `"root"` に、作業ディレクトリを `/tmp` にそれぞれ設定します。次に、`code` ブロックで Bash スクリプトを実行します。このブロックには、行を必要な数だけ含めることができます。詳細については、「[script](#)」を参照してください。

レシピを使用してスクリプトを実行する方法の詳細については、「[例 7: コマンドとスクリプトの実行](#)」を参照してください。Windows インスタンスで PowerShell スクリプトを実行する方法の例については、「[Windows PowerShell スクリプトの実行](#)」を参照してください。

Chef デプロイフックの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

必要なタスクを実行するカスタムレシピを実装し、適切なレイヤーの Deploy イベントに割り当てることで、デプロイをカスタマイズできます。特に他の目的でクックブックを実装する必要がない場合は、Chef デプロイメントフックを使用してカスタマイズコードを実行する、別のシンプルなアプローチがあります。さらに、組み込みレシピによってデプロイメントが実行された後に、カスタム Deploy レシピを実行します。デプロイフックを使用すると、デプロイメントの途中で (たとえば、アプリケーションのコードがリポジトリからチェックアウトされた後、Apache が再起動される前に) 操作することが可能になります。

Chef では、次の 4 つのステージでアプリケーションをデプロイします。

- Checkout (チェックアウト) - リポジトリからファイルをダウンロードします。
- Migrate (移行) - 必要に応じて移行を実行します。
- Symlink (シンボリックリンク) - シンボリックリンクを作成します。
- Restart (再起動) — アプリケーションを再起動します。

Chef デプロイフックを使用すると、各ステージの完了後にユーザーが提供する Ruby アプリケーションを必要に応じて実行することで、デプロイメントを簡単にカスタマイズできます。デプロイフックを使用するには、1 つ以上の Ruby アプリケーションを実装し、アプリケーションの /deploy ディレクトリに配置します。(アプリケーションに /deploy ディレクトリがない場合は、APP_ROOT レベルに 1 つ作成します)。アプリケーションには、いつ実行するかを示す次のいずれかの名前を指定する必要があります。

- before_migrate.rb - チェックアウトステージの完了後、移行ステージの前に実行します。

- `before_symlink.rb` – 移行ステージの完了後、シンボリックリンクステージの前に実行します。
- `before_restart.rb` – シンボリックリンクステージの完了後、再起動ステージの前に実行します。
- `after_restart.rb` – 再起動ステージの完了後に実行します。

Chef デプロイフックは標準のノード構文を使用して、レシピのようなノードオブジェクトにアクセスできます。また、デプロイフックは、指定した任意の[アプリケーション環境変数](#)の値にアクセスできます。ただし、`new_resource.environment["VARIABLE_NAME"]` を使用して、`ENV["VARIABLE_NAME"]` ではなく変数の値にアクセスする必要があります。

Linux インスタンスでの cron ジョブの実行

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Linux の cron ジョブは、指定したスケジュールで 1 つ以上のコマンドを実行するよう cron デーモンに指示します。例えば、スタックが e PHP コマースアプリケーションをサポートしているとします。毎週指定した時刻にサーバーから販売レポートが送信されるように cron ジョブをセットアップできます。cron の詳細については、Wikipedia の「[Cron](#)」を参照してください。Linux ベースのコンピュータまたはインスタンスで直接クローンジョブを実行する方法の詳細については、Indiana University Knowledge Base Web サイトの「[What are cron and crontab, and how do I use them?](#)」を参照してください。

個々の Linux ベースのインスタンスにジョブを手動で設定するには、を使用して cron ジョブに接続し SSH、その crontab エントリを編集しますが、AWS OpsWorks スタックの主な利点は、インスタンスのレイヤー全体でタスクを実行するように指示できることです。次の手順では、PHP App Server レイヤーのインスタンスで cron ジョブを設定する方法について説明しますが、どのレイヤーでも同じアプローチを使用できます。

レイヤーのインスタンスで **cron** ジョブをセットアップするには

1. ジョブをセットアップする cron リソースを使用するレシピを含むクックブックを実装します。この例では、レシピ名を `cronjob.rb` としています。実装の詳細については後述します。クックブックとレシピの詳細については、「[クックブックとレシピ](#)」を参照してください。
2. スタックにクックブックをインストールします。詳細については、「[カスタムクックブックのインストール](#)」を参照してください。
3. 次のライフサイクルイベントに割り当てることで、AWS OpsWorks スタックでレイヤーのインスタンスでレシピを自動的に実行させます。詳細については、「[レシピを自動的に実行する](#)」を参照してください。
 - セットアップ – このイベント `cronjob.rb` に割り当てると、すべての新しいインスタンスでレシピを実行するように AWS OpsWorks スタックに指示します。
 - デプロイ – このイベント `cronjob.rb` に割り当てると、アプリケーションをレイヤーにデプロイまたは再デプロイするときに、すべてのオンラインインスタンスでレシピを実行するように AWS OpsWorks スタックに指示します。

Execute Recipes スタックコマンドを使用して、オンラインインスタンスでレシピを手動で実行することもできます。詳細については、「[スタックコマンドの実行](#)」を参照してください。

以下は、サーバーから売上データを収集し、レポートをメールする、ユーザーが実装したPHPアプリケーションを週に 1 回実行するように cron ジョブを設定する `cronjob.rb` 例です。cron リソースの使用法のその他の例については、「[cron](#)」を参照してください。

```
cron "job_name" do
  hour "1"
  minute "10"
  weekday "6"
  command "cd /srv/www/myapp/current && php .lib/mailing.php"
end
```

`cron` は、cron ジョブを表す Chef リソースです。AWS OpsWorks スタックがインスタンスでレシピを実行すると、関連するプロバイダーがジョブの設定の詳細を処理します。

- `job_name` は、cron ジョブのユーザー定義名 (`weekly report` など) です。

- `hour/minute/weekday` では、コマンドをいつ実行するかを指定します。この例では、毎週土曜日の午前 1 時 10 分にコマンドを実行します。
- `command` では、実行するコマンドを指定します。

この例では 2 つのコマンドを実行します。まず `/srv/www/myapp/current` ディレクトリに移動します。2 番目のコマンドでは、販売データを収集してレポートを送信する、ユーザーが実装した `mailing.php` アプリケーションを実行します。

Note

デフォルトで、`bundle` コマンドでは `cron` ジョブを操作できません。これは、AWS OpsWorks スタックが `/usr/local/bin` ディレクトリにバンドルをインストールするためです。`bundle` ジョブで `cron` を使用するには、`cron` ジョブに明示的にパス `/usr/local/bin` を追加する必要があります。また、`$PATH` 環境変数は `cron` ジョブで拡張されない可能性があるため、ベストプラクティスは、`$PATH` 変数の拡張に依存することなく、ジョブに必要なパス情報を明示的に追加することです。以下の例では、`bundle` ジョブで `cron` を使用する方法を 2 種類示しています。

```
cron "my first task" do
  path "/usr/local/bin"
  minute "*/10"
  command "cd /srv/www/myapp/current && bundle exec my_command"
end
```

```
cron_env = {"PATH" => "/usr/local/bin"}
cron "my second task" do
  environment cron_env
  minute "*/10"
  command "cd /srv/www/myapp/current && /usr/local/bin/bundle exec my_command"
end
```

スタックに複数のアプリケーションサーバーがある場合、PHPApp Server レイヤーのライフサイクルイベント `cronjob.rb` に割り当てることは理想的なアプローチではない可能性があります。たとえば、レイヤーのすべてのインスタンスでレシピが実行されるので、ユーザーには複数のレポートが送

信されることとなります。この場合、適切な方法は、カスタムレイヤーを使用して1つのサーバーだけがレポートを送信できるようにすることです。

レイヤーの1つのインスタンスでのみレシピを実行するには

1. 例えば、 という名前のカスタムレイヤーPHPAdminを作成し、そのセットアップイベントとデプロイイベントcronjob.rbに を割り当てます。カスタムレイヤーでは、必ずしも多くのことを実行する必要はありません。この場合、 はインスタンスで1つのカスタムレシピPHPAdminのみを実行します。
2. App Server インスタンスの1 PHP つを に割り当てます AdminLayer。インスタンスが複数のレイヤーに属している場合、AWS OpsWorks スタックは各レイヤーの組み込みレシピとカスタムレシピを実行します。

App PHP Server とPHPAdminレイヤーに属するインスタンスは1つだけであるため、 はそのインスタンスでのみcronjob.rb実行され、レポートは1つだけ受け取ります。

Linux インスタンスでパッケージをインストールおよび設定

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

組み込みレイヤーでは、特定のパッケージのみがサポートされます。詳細については、「[レイヤー](#)」を参照してください。Redis サーバーなど、他のパッケージをインストールするには、関連するセットアップ、設定、デプロイメントの各タスクを処理するカスタムレシピを実装します。組み込みレイヤーを拡張して、レイヤーの標準パッケージと共に目的のパッケージをインスタンスにインストールするのが最良の方法である場合もあります。例えば、PHPアプリケーションをサポートするスタックがあり、Redis サーバーを含める場合は、PHPアプリケーションサーバーを拡張して、PHPアプリケーションサーバーに加えて、レイヤーのインスタンスに Redis サーバーをインストールして設定できます。

通常、パッケージのインストールレシピでは、次のようなタスクを実行する必要があります。

- 1 つ以上のディレクトリを作成し、各ディレクトリのモードを設定する。
- テンプレートから設定ファイルを作成する。
- インストーラを実行してインスタンスにパッケージをインストールする。
- 1 つ以上のサービスを開始する。

Tomcat サーバーのインストール方法の例については、「[カスタム Tomcat サーバーレイヤーの作成](#)」を参照してください。このトピックでは、カスタム Redis レイヤーをセットアップする方法を説明していますが、ほぼ同じコードを使用して組み込みレイヤーに Redis をインストールし、設定することができます。他のパッケージをインストールする方法の例については、<https://github.com/aws/opsworks-cookbooks> の [組み込みクックブック](#) を参照してください。

カスタム Tomcat サーバーレイヤーの作成

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このトピックでは、Linux スタックのカスタムレイヤーを実装する方法について説明します。ただし、一部の基本的な原則とコードは、特に、アプリケーションのデプロイのセクションで、Windows スタックのカスタムレイヤーの実装にも当てはまります。

AWS OpsWorks スタックインスタンスで非標準パッケージを使用する最も簡単な方法は、[既存のレイヤーを拡張](#)することです。ただし、この方法は、レイヤーのインスタンスに標準と標準外の両方のパッケージをインストールして実行するため、常に望ましいわけではありません。要求が厳しくなりますがより強力な方法は、カスタムレイヤーを実装することです。そうすれば、レイヤーのインスタンスに対して、次の事項をほぼ完全にコントロールできるようになります。

- どのパッケージをインストールするか
- 各パッケージをどのように設定するか

- リポジトリからインスタンスにどのようにアプリケーションをデプロイするか

コンソールと のどちらを使用するかにかかわらずAPI、「」で説明されているように、カスタムレイヤーを作成および管理します [カスタムレイヤー](#)。ただし、カスタムレイヤーの組み込みのレシピは、Ganglia クライアントをインストールして Ganglia マスターにメトリクスをレポートするなど、一部の最も基本的なタスクだけを実行します。カスタムレイヤーのインスタンスが最小限の機能にとどまらないようにするために、パッケージのインストールと設定、アプリケーションのデプロイなどのタスクを処理するための Chef レシピと関連ファイルを備えた 1 つ以上のカスタムクックブックを実装する必要があります。ただし、必ずしもすべてを最初から新しく実装する必要はありません。たとえば、標準のリポジトリの 1 つにアプリケーションを保存する場合は、組み込みのレシピを使用することで、アプリケーションをレイヤーのインスタンスにインストールする作業の多くを処理できます。

Note

Chef を初めて使う場合は、最初に「[クックブック 101](#)」をお読みください。これは、クックブックを実装してさまざまな一般的なタスクを実行する方法の基本事項を紹介するチュートリアルです。

次のウォークスルーでは、Tomcat アプリケーションサーバーをサポートするカスタムレイヤーの実装方法を説明します。このレイヤーは、パッケージのインストール、デプロイなどを処理するためのレシピを含む Tomcat というカスタムクックブックに基づいています。このウォークスルーは、Tomcat クックブックからの抜粋を含んでいます。完全なクックブックはリポジトリからダウンロードできます [GitHub](#)。 [Opscode Chef](#) に慣れていない場合は、最初に「[クックブックとレシピ](#)」をお読みください。

Note

AWS OpsWorks スタックには、本番稼働用のフル機能の [Java App Server レイヤー](#) が含まれています。Tomcat クックブックの目的は、カスタムレイヤーの実装方法を示すことです。そのため、 などの機能を含まない限定バージョンの Tomcat のみがサポートされます SSL。フル機能の実装の例については、組み込みの [opsworks_java](#) クックブックを参照してください。

Tomcat クックブックは、インスタンスが次の特徴を持つカスタムレイヤーをサポートします。

- インスタンスは、Apache のフロントエンドで Tomcat Java アプリケーションサーバーをサポートします。
- Tomcat は、アプリケーションが JDBC DataSource オブジェクトを使用して、バックエンド データストアとして機能する別の MySQL インスタンスに接続できるように設定されています。

このプロジェクトのクックブックには、複数の主要コンポーネントがあります。

- [属性ファイル](#)には、さまざまなレシピで使用される設定が含まれます。
- [Setup レシピ](#)は、[レイヤーの Setup ライフサイクルイベント](#)に割り当てられます。このレシピは、インスタンスの起動後に実行され、パッケージのインストールや設定ファイルの作成などのタスクを行います。
- [Configure レシピ](#)は、レイヤーの [Configure ライフサイクルイベント](#)に割り当てられます。スタックの設定変更後、主にインスタンスがオンラインになったときやオフラインになったときに実行され、必要な構成変更を処理します。
- Deploy レシピは、レイヤーの [Deploy ライフサイクル](#) イベントに割り当てられます。このレシピは、Setup レシピの後、アプリケーションを手動でデプロイし、レイヤーのインスタンスでコードおよび関連ファイルをインストールするときに実行されます。また、サービスの再起動などの関連タスクを処理するためにも実行されます。

最後のセクション「」では[スタックの作成とアプリケーションの実行](#)、Tomcat クックブックに基づくカスタムレイヤーを含むスタックを作成する方法と、別の MySQL レイヤーに属するインスタンスで実行されている MySQL データベースのデータを表示するシンプルな JSP アプリケーションをデプロイして実行する方法について説明します。

Note

Tomcat クックブックレシピは、一部の AWS OpsWorks スタックの組み込みレシピに依存します。各レシピのオリジンを明確にするために、このトピックでは Chef `cookbookname::recipe` 規則を使用してレシピを識別します。

トピック

- [属性ファイル](#)
- [Setup レシピ](#)
- [Configure レシピ](#)

- [Deploy レシピ](#)
- [スタックの作成とアプリケーションの実行](#)

属性ファイル

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

レシピを確認する前に、最初に Tomcat クックブックの属性ファイルを調べておくと便利です。属性ファイルには、そのレシピで使用するさまざまな設定ファイルが含まれています。属性は必須ではなく、レシピやテンプレートで値を単にハードコーディングすることもできます。ただし、属性を使用して構成設定を定義する場合は、スタックコンソールまたは AWS OpsWorks を使用して、カスタム属性を定義することで値を API 変更できます。カスタム JSON 属性は、設定を変更するたびにレシピまたはテンプレートコードを書き換えるよりもシンプルで柔軟性があります。この方法では、たとえば、複数のスタックに同じクックブックを使用しながら、スタックごとに Tomcat サーバーを異なる設定にすることができます。属性および属性をオーバーライドする方法の詳細については、「[属性の上書き](#)」を参照してください。

次の例は、完全な属性ファイル `default.rb` を示しています。このファイルは、Tomcat クックブックの `attributes` ディレクトリにあります。

```
default['tomcat']['base_version'] = 6
default['tomcat']['port'] = 8080
default['tomcat']['secure_port'] = 8443
default['tomcat']['ajp_port'] = 8009
default['tomcat']['shutdown_port'] = 8005
default['tomcat']['uri_encoding'] = 'UTF-8'
default['tomcat']['unpack_wars'] = true
default['tomcat']['auto_deploy'] = true
case node[:platform]
when 'centos', 'redhat', 'fedora', 'amazon'
  default['tomcat']['java_opts'] = ''
```

```
when 'debian', 'ubuntu'
  default['tomcat']['java_opts'] = '-Djava.awt.headless=true -Xmx128m -XX:
+UseConcMarkSweepGC'
end
default['tomcat']['catalina_base_dir'] = "/etc/tomcat#{node['tomcat']['base_version']}"
default['tomcat']['webapps_base_dir'] = "/var/lib/tomcat#{node['tomcat']
['base_version']}/webapps"
default['tomcat']['lib_dir'] = "/usr/share/tomcat#{node['tomcat']['base_version']}/lib"
default['tomcat']['java_dir'] = '/usr/share/java'
default['tomcat']['mysql_connector_jar'] = 'mysql-connector-java.jar'
default['tomcat']['apache_tomcat_bind_mod'] = 'proxy_http' # or: 'proxy_ajp'
default['tomcat']['apache_tomcat_bind_config'] = 'tomcat_bind.conf'
default['tomcat']['apache_tomcat_bind_path'] = '/tc/'
default['tomcat']['webapps_dir_entries_to_delete'] = %w(config log public tmp)
case node[:platform]
when 'centos', 'redhat', 'fedora', 'amazon'
  default['tomcat']['user'] = 'tomcat'
  default['tomcat']['group'] = 'tomcat'
  default['tomcat']['system_env_dir'] = '/etc/sysconfig'
when 'debian', 'ubuntu'
  default['tomcat']['user'] = "tomcat#{node['tomcat']['base_version']}"
  default['tomcat']['group'] = "tomcat#{node['tomcat']['base_version']}"
  default['tomcat']['system_env_dir'] = '/etc/default'
end
```

設定自体については、関連するセクションで後ほど説明します。次の注意事項が一般的に適用されま

- すべてのノード定義は default タイプであるため、[カスタムJSON属性](#)で上書きできます。
- このファイルは、case ステートメントを使用して、インスタンスのオペレーティングシステムに応じて条件付きで一部の属性値を設定します。

platform ノードが Chef の Ohai ツールによって生成され、インスタンスのオペレーティングシステムを表します。

Setup レシピ

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソ

リユーシオンに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Setup レシピは、レイヤーの Setup [ライフサイクル](#) イベントに割り当てられ、インスタンスの起動後に実行されます。このレシピは、パッケージのインストール、設定ファイルの作成、サービスの開始などのタスクを実行します。Setup レシピの実行が完了すると、AWS OpsWorks Stacks は [Deploy レシピ](#) を実行して、新しいインスタンスにアプリケーションをデプロイします。

トピック

- [tomcat::setup](#)
- [tomcat::install](#)
- [tomcat::service](#)
- [tomcat::container_config](#)
- [tomcat::apache_tomcat_bind](#)

tomcat::setup

tomcat::setup レシピは、レイヤーの Setup ライフサイクルイベントに割り当てられます。

```
include_recipe 'tomcat::install'
include_recipe 'tomcat::service'

service 'tomcat' do
  action :enable
end

# for EBS-backed instances we rely on autofs
bash '(re-)start autofs earlier' do
  user 'root'
  code <<-EOC
  service autofs restart
EOC
  notifies :restart, resources(:service => 'tomcat')
end

include_recipe 'tomcat::container_config'
```

```
include_recipe 'apache2'  
include_recipe 'tomcat::apache_tomcat_bind'
```

`tomcat::setup` レシピは、主としてメタレシピです。Tomcat および関連パッケージのインストールと設定の詳細のほとんどを処理する一連の依存レシピが含まれています。`tomcat::setup` の最初の部分は、後で説明する次のレシピを実行します。

- [tomcat::install](#) レシピは、Tomcat サーバーパッケージをインストールします。
- [tomcat::service](#) レシピは、Tomcat サービスをセットアップします。

`tomcat::setup` の中央部分は、Tomcat サービスを有効にして起動します。

- Chef [service リソース](#) は、Tomcat サービスを起動時に有効にします。
- Chef [bash リソース](#) は Bash スクリプトを実行して `autofs` デーモンを起動します。これは Amazon EBS-backed インスタンスに必要です。次に、このリソースは、Tomcat サービスの再起動を `service` リソースに通知します。

詳細については、「[autofs](#)」(Amazon Linux の場合) または「[Autofs](#)」(Ubuntu の場合) を参照してください。

`tomcat::setup` の最後の部分は、設定ファイルを作成し、フロントエンド Apache サーバーをインストールおよび設定します。

- [tomcat::container_config](#) レシピは、設定ファイルを作成します。
- `apache2` レシピ (の省略形 `apache2::default`) は、Apache AWS OpsWorks サーバーをインストールして設定する スタックの組み込みレシピです。
- [tomcat::apache_tomcat_bind](#) レシピは、Tomcat サーバーのフロントエンドとして機能するように Apache サーバーを設定します。

Note

組み込みのレシピを使用して必要なタスクの一部を実行することにより、多くの場合、時間と労力を削減できます。このレシピは、組み込みの `apache2::default` レシピを使用することで、Apache を最初から実装する必要なくインストールします。組み込みのレシピを使用するその他の例については、「[Deploy レシピ](#)」を参照してください。

以下のセクションでは、Tomcat クックブックの Setup レシピについてさらに詳しく説明します。apache2 レシピの詳細については、「[opsworks-cookbooks/apache2](#)」を参照してください。

tomcat::install

tomcat::install レシピは、Tomcat サーバー、OpenJDK、My サーバーへの接続を処理する Java コネクタライブラリをインストールしますSQL。

```
tomcat_pkgs = value_for_platform(
  ['debian', 'ubuntu'] => {
    'default' => ["tomcat#{node['tomcat']['base_version']}", 'libtcnative-1',
  'libmysql-java']
  },
  ['centos', 'redhat', 'fedora', 'amazon'] => {
    'default' => ["tomcat#{node['tomcat']['base_version']}", 'tomcat-native', 'mysql-
connector-java']
  },
  'default' => ["tomcat#{node['tomcat']['base_version']}"]
)

tomcat_pkgs.each do |pkg|
  package pkg do
    action :install
  end
end

link ::File.join(node['tomcat']['lib_dir'], node['tomcat']['mysql_connector_jar']) do
  to ::File.join(node['tomcat']['java_dir'], node['tomcat']['mysql_connector_jar'])
  action :create
end

# remove the ROOT webapp, if it got installed by default
include_recipe 'tomcat::remove_root_webapp'
```

このレシピは、次のタスクを実行します。

1. インスタンスのオペレーティングシステムに応じて、インストールするパッケージのリストを作成します。
2. リストに各パッケージをインストールします。

Chef [package resource](#) (パッケージリソース) は適切なプロバイダを使用します。yum Amazon Linux および apt-get Ubuntu の場合— インストールを処理します。パッケージプロバイダーは Tomcat の依存関係として OpenJDK をインストールしますが、MySQL コネクタライブラリは明示的にインストールする必要があります。

3. Chef [リンクリソース](#)を使用して、Tomcat サーバーの lib ディレクトリにシンボリックリンクを作成し、のMySQLコネクタライブラリを作成しますJDK。

デフォルトの属性値を使用すると、Tomcat lib ディレクトリは `/usr/share/tomcat6/lib`で、MySQL コネクタライブラリ (`mysql-connector-java.jar`) は にあります`/usr/share/java/`。

`tomcat::remove_root_webapp` レシピは、一部のセキュリティ問題を回避するために、ROOT ウェブアプリケーション (`/var/lib/tomcat6/webapps/ROOT`デフォルトでは) を削除します。

```
ruby_block 'remove the ROOT webapp' do
  block do
    ::FileUtils.rm_rf(::File.join(node['tomcat']['webapps_base_dir'], 'ROOT'), :secure
=> true)
  end
  only_if { ::File.exists?(::File.join(node['tomcat']['webapps_base_dir'], 'ROOT'))
&& !::File.symlink?(::File.join(node['tomcat']['webapps_base_dir'], 'ROOT')) }
end
```

`only_if` ステートメントにより、このレシピは、ファイルが存在する場合にのみファイルを削除します。

Note

Tomcat のバージョンは、`['tomcat']['base_version']` 属性で指定されます。この属性ファイルでは 6 に設定されています。Tomcat 7 をインストールするには、カスタムJSON 属性を使用して属性を上書きできます。[スタック設定を編集](#)してカスタム Chef JSON JSON ボックスに次のように入力するか、既存のカスタム に追加しますJSON。

```
{
  'tomcat' : {
    'base_version' : 7
  }
}
```

```
}
```

カスタムJSON属性はデフォルトの属性を上書きし、Tomcat バージョンを 7 に設定します。属性の上書きの詳細については、「[属性の上書き](#)」を参照してください。

```
tomcat::service
```

`tomcat::service` レシピは、Tomcat サービスの定義を作成します。

```
service 'tomcat' do
  service_name "tomcat#{node['tomcat']['base_version']}"

  case node[:platform]
  when 'centos', 'redhat', 'fedora', 'amazon'
    supports :restart => true, :reload => true, :status => true
  when 'debian', 'ubuntu'
    supports :restart => true, :reload => false, :status => true
  end

  action :nothing
end
```

このレシピは、Chef [service リソース](#)を使用して Tomcat サービス名 (デフォルトでは tomcat6) を指定し、`supports` 属性を設定して、さまざまなオペレーティングシステムでサービスの `restart` コマンド、`reload` コマンド、および `status` コマンドを Chef によって管理する方法を定義します。

- `true` は、Chef が `init` またはその他のサービスプロバイダーを使用してコマンドを実行できることを示します。
- `false` は、Chef がその他の方法でコマンドの実行を試みる必要があることを示します。

`action` が `:nothing` に設定されていることに注意してください。ライフサイクルイベントごとに、AWS OpsWorks スタックは [Chef 実行](#)を開始して適切なレシピのセットを実行します。Tomcat クックブックは、レシピでサービスの定義を作成するがサービスの再起動はしないという一般的なパターンに従います。Chef 実行のその他のレシピは、通常、設定ファイルの作成に使用する `notifies` リソースに `template` コマンドを含めることで、再起動を処理します。通知を使用すると、設定が変更された場合にのみサービスが再起動されるため、通知はサービスの再起動のための便利な方法です。また、Chef 実行で 1 つのサービスに対して複数の再起動通知がある場合、Chef は最

大で 1 回サービスを再起動します。これにより、Tomcat のエラーの一般的な原因である、完全には機能していないサービスを再起動しようとする際に発生する可能性がある問題が回避されます。

Tomcat サービスは、再起動通知を使用する任意の Chef 実行に対して定義する必要があります。したがって、`tomcat::service` は複数のレシピに含まれ、それぞれの Chef 実行に対してサービスが定義されることとなります。Chef 実行に `tomcat::service` の複数のインスタンスが含まれる場合のペナルティはありません。これは、Chef ではレシピの実行回数が Chef 実行ごとに 1 回のみであるためです。

`tomcat::container_config`

`tomcat::container_config` レシピは、クックブックテンプレートファイルから設定ファイルを作成します。

```
include_recipe 'tomcat::service'

template 'tomcat environment configuration' do
  path ::File.join(node['tomcat']['system_env_dir'], "tomcat#{node['tomcat']
['base_version']}")
  source 'tomcat_env_config.erb'
  owner 'root'
  group 'root'
  mode 0644
  backup false
  notifies :restart, resources(:service => 'tomcat')
end

template 'tomcat server configuration' do
  path ::File.join(node['tomcat']['catalina_base_dir'], 'server.xml')
  source 'server.xml.erb'
  owner 'root'
  group 'root'
  mode 0644
  backup false
  notifies :restart, resources(:service => 'tomcat')
end
```

このレシピは、必要に応じてサービスを定義する `tomcat::service` を最初に呼び出します。このレシピの大部分は、2 つの [template resources](#) で構成され、クックブックのいずれかのテンプレートファイルから設定ファイルを作成し、ファイルのプロパティを設定し、Chef にサービスを再起動するよう通知します。

Tomcat 環境設定

最初の template リソースは、tomcat_env_config.erb テンプレートファイルを使用して Tomcat 環境設定ファイルを作成します。このファイルは、JAVA_HOME などの環境変数の設定に使用します。デフォルトのファイル名は、template リソースの引数です。tomcat::container_config は、path 属性を使用し、デフォルト値を上書きして設定ファイルに /etc/sysconfig/tomcat6 (Amazon Linux の場合) または /etc/default/tomcat6 (Ubuntu の場合) の名前を付けます。また、template リソースは、ファイルの所有者、グループ、およびモードの設定を指定し、バックアップファイルを作成しないように Chef に指示します。

ソースコードを確認すると、実際には 3 つのバージョンの tomcat_env_config.erb が存在し、それぞれ templates ディレクトリの異なるサブディレクトリ内にあります。ubuntu ディレクトリおよび amazon ディレクトリには、それぞれのオペレーティングシステムに対応するテンプレートが含まれます。default フォルダには、1 行のコメント行があるダミーのテンプレートが含まれています。このテンプレートは、サポートされていないオペレーティングシステムを使用するインスタンスでこのレシピを実行しようとした場合にのみ使用されます。tomcat::container_config レシピでは、使用する tomcat_env_config.erb を指定する必要はありません。Chef は、「[File Specificity](#)」で説明されているルールに基づいて、インスタンスのオペレーティングシステム用の適切なディレクトリを自動的に選択します。

この例の tomcat_env_config.erb ファイルは、大部分がコメントで構成されています。追加の環境変数を設定するには、適切な行をコメント解除し、新しい値を指定します。

Note

変更する可能性がある設定は、テンプレートにハードコーディングするのではなく、属性として定義する必要があります。そうすれば、設定を変更するためにテンプレートを書き直す必要はなく、属性を上書きするだけで済みます。

次の抜粋に示すように、Amazon Linux テンプレートは 1 つの環境変数のみを設定します。

```
...
# Use JAVA_OPTS to set java.library.path for libtcnative.so
#JAVA_OPTS="-Djava.library.path=/usr/lib"

JAVA_OPTS="${JAVA_OPTS} <%= node['tomcat']['java_opts'] %>"

# What user should run tomcat
```

```
#TOMCAT_USER="tomcat"
...
```

JAVA_OPTS を使用して、ライブラリパスなどの Java オプションを指定できます。デフォルトの属性値を使用すると、このテンプレートは Amazon Linux の Java オプションを設定しません。カスタム['tomcat']['java_opts']属性を使用するなど、JSON 属性を上書きすることで、独自の Java オプションを設定できます。例については、[スタックの作成](#)を参照してください。

次のテンプレートの抜粋に示すように、Ubuntu テンプレートは複数の環境変数を設定します。

```
# Run Tomcat as this user ID. Not setting this or leaving it blank will use the
# default of tomcat<%= node['tomcat']['base_version'] %>.
TOMCAT<%= node['tomcat']['base_version'] %>_USER=tomcat<%= node['tomcat']
['base_version'] %>
...
# Run Tomcat as this group ID. Not setting this or leaving it blank will use
# the default of tomcat<%= node['tomcat']['base_version'] %>.
TOMCAT<%= node['tomcat']['base_version'] %>_GROUP=tomcat<%= node['tomcat']
['base_version'] %>
...
JAVA_OPTS="<%= node['tomcat']['java_opts'] %>"

<% if node['tomcat']['base_version'].to_i < 7 -%>
# Unset LC_ALL to prevent user environment executing the init script from
# influencing servlet behavior. See Debian bug #645221
unset LC_ALL
<% end -%>
```

デフォルトの属性値を使用すると、このテンプレートは次のように Ubuntu 環境変数を設定します。

- Tomcat のユーザーとグループを表す TOMCAT6_USER および TOMCAT6_GROUP は、いずれも tomcat6 に設定されます。

['tomcat']['base_version'] を tomcat7 に設定する場合、変数名は TOMCAT7_USER および TOMCAT7_GROUP に解決され、いずれも tomcat7 に設定されます。
- JAVA_OPTS は、-Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC に設定されます。
 - Djava.awt.headless を true に設定すると、インスタンスがヘッドレスでコンソールがないことがグラフィックエンジンに通知されます。これにより、特定のグラフィカルアプリケーションの不良動作に対処します。

- -Xmx128m は、この例で に十分なメモリリソース 128MB JVMがあることを確認します。
 - -XX:+UseConcMarkSweepGC は、コンカレントマークスイープガベージコレクションを指定します。これは、ガベージコレクションの一時停止を限定するのに役立ちます。
- 詳細については、「[コンカレントマークスイープコレクタの拡張機能](#)」を参照してください。
- Tomcat のバージョンが 7 より前の場合、このテンプレートは、Ubuntu のバグに対処する LC_ALL を取り消します。

Note

デフォルトの属性を使用すると、これらの環境変数の一部が単にデフォルト値に設定されます。ただし、環境変数を属性に明示的に設定すると、カスタムJSON属性を定義してデフォルトの属性を上書きし、カスタム値を指定できます。属性の上書きの詳細については、「[属性の上書き](#)」を参照してください。

完全なテンプレートファイルについては、[ソースコード](#)を参照してください。

Server.xml 設定ファイル

2 番目のtemplateリソースは server.xml.erbを使用して[system.xml設定ファイル](#)を作成します。このファイルは servlet/JSP コンテナを設定します。server.xml.erbにはオペレーティングシステム固有の設定がないため、templateディレクトリの defaultサブディレクトリにあります。

このテンプレートは、標準設定を使用しますが、Tomcat 6 または Tomcat 7 のいずれかに対して system.xml ファイルを作成できます。たとえば、このテンプレートのサーバーセクションの次のコードは、指定されたバージョンに合わせてリスナーを適切に設定します。

```
<% if node['tomcat']['base_version'].to_i > 6 -%>
  <!-- Security listener. Documentation at /docs/config/listeners.html
  <Listener className="org.apache.catalina.security.SecurityListener" />
  -->
<% end -%>
  <!--APR library loader. Documentation at /docs/apr.html -->
  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
  <!--Initialize Jasper prior to webapps are loaded. Documentation at /docs/jasper-howto.html -->
```

```

<Listener className="org.apache.catalina.core.JasperListener" />
<!-- Prevent memory leaks due to use of particular java/javax APIs-->
<Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
<% if node['tomcat']['base_version'].to_i < 7 -%>
  <!-- JMX Support for the Tomcat server. Documentation at /docs/non-existent.html -->
  <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener" />
<% end -%>
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
<% if node['tomcat']['base_version'].to_i > 6 -%>
  <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />
<% end -%>

```

テンプレートはハードコードされた設定の代わりに属性を使用するため、カスタムJSON属性を定義することで設定を簡単に変更できます。以下に例を示します。

```

<Connector port="<%= node['tomcat']['port'] %>" protocol="HTTP/1.1"
  connectionTimeout="20000"
  URIEncoding="<%= node['tomcat']['uri_encoding'] %>"
  redirectPort="<%= node['tomcat']['secure_port'] %>" />

```

詳細については、[ソースコード](#)を参照してください。

tomcat::apache_tomcat_bind

tomcat::apache_tomcat_bind レシピは、Apache サーバーが Tomcat のフロントエンドとして動作すること、つまり、受信するリクエストを受け取って Tomcat に転送し、応答をクライアントに返すことを可能にします。この例では、Apache のプロキシ/ゲートウェイとして [mod_proxy](#) を使用します。

```

execute 'enable mod_proxy for apache-tomcat binding' do
  command '/usr/sbin/a2enmod proxy'
  not_if do
    ::File.symlink?(::File.join(node['apache']['dir'], 'mods-enabled', 'proxy.load'))
  || node['tomcat']['apache_tomcat_bind_mod'] !~ /\Aproxy/
  end
end

execute 'enable module for apache-tomcat binding' do
  command "/usr/sbin/a2enmod #{node['tomcat']['apache_tomcat_bind_mod']}"
end

```

```
not_if {::File.symlink?(::File.join(node['apache']['dir'], 'mods-enabled',
"#{node['tomcat']['apache_tomcat_bind_mod'].load"))}
end

include_recipe 'apache2::service'

template 'tomcat thru apache binding' do
  path ::File.join(node['apache']['dir'], 'conf.d', node['tomcat']
['apache_tomcat_bind_config'])
  source 'apache_tomcat_bind.conf.erb'
  owner 'root'
  group 'root'
  mode 0644
  backup false
  notifies :restart, resources(:service => 'apache2')
end
```

`mod_proxy` を有効にするには、`proxy` モジュールとプロトコルベースのモジュールを有効にする必要があります。プロトコルモジュールには、次の 2 つのオプションがあります。

- HTTP: `proxy_http`
- [Apache JServプロトコル \(AJP\)](#): `proxy_ajp`

AJP は内部 Tomcat プロトコルです。

このレシピの両方の [execute リソース](#)が `a2enmod` コマンドを実行します。このコマンドは、必要なシンボリックリンクを作成することで、指定されたモジュールを有効にします。

- 最初の `execute` リソースは、`proxy` モジュールを有効にします。
- 2 番目の `execute` リソースは、デフォルトで `proxy_http` に設定されているプロトコルモジュールを有効にします。

を使用する場合は AJP、`apache_tomcat_bind_mod` 属性を上書き JSON して に設定するカスタム を定義できません `proxy_ajp`。

`apache2::service` レシピは、Apache サービスを定義する AWS OpsWorks スタックの組み込みレシピです。詳細については、スタック GitHub リポジトリの [recipe](#) AWS OpsWorks を参照してください。

template リソースでは、`apache_tomcat_bind.conf.erb` を使用して、デフォルトで `tomcat_bind.conf` という名前のファイルを作成します。ファイルは、`['apache']` `['dir']/.conf.d` ディレクトリに配置されます。`['apache']``['dir']` 属性は、組み込みの `apache2` 属性ファイルで定義され、デフォルトでは `/etc/httpd` (Amazon Linux の場合) または `/etc/apache2` (Ubuntu の場合) に設定されます。この template リソースが設定ファイルを作成または変更する場合は、`notifies` コマンドが Apache サービスの再起動をスケジュール設定します。

```
<% if node['tomcat']['apache_tomcat_bind_mod'] == 'proxy_ajp' -%>
ProxyPass <%= node['tomcat']['apache_tomcat_bind_path'] %> ajp://localhost:<%=
node['tomcat']['ajp_port'] %>/
ProxyPassReverse <%= node['tomcat']['apache_tomcat_bind_path'] %> ajp://localhost:<%=
node['tomcat']['ajp_port'] %>/
<% else %>
ProxyPass <%= node['tomcat']['apache_tomcat_bind_path'] %> http://localhost:<%=
node['tomcat']['port'] %>/
ProxyPassReverse <%= node['tomcat']['apache_tomcat_bind_path'] %> http://localhost:<%=
node['tomcat']['port'] %>/
<% end -%>
```

テンプレートは、[ProxyPass](#)および[ProxyPassReverse](#)ディレクティブを使用して、Apache と Tomcat 間のトラフィックを渡すために使用されるポートを設定します。両方のサーバーは同じインスタンス上にあるため、`localhost` を使用できURL、どちらもデフォルトで `http://localhost:8080` に設定されます。

Configure レシピ

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Configure レシピは、レイヤーの Configure [ライフサイクル](#) イベントに割り当てられます。このイベントは、スタックのすべてのインスタンスで、インスタンスがオンライン状態になったとき、またはオンライン状態から別の状態になったときに発生します。Configure レシピを使用することで、必要

に応じ、変更に対応してインスタンスの設定を調整します。Configure レシピを実装するときは、スタックの設定の変更がこのレイヤーとは無関係なインスタンスに参与している場合があることに注意してください。このレシピは、適切に応答できる必要があります。それは、場合によっては何も処理を実行しないことがあることを意味します。

tomcat::configure

tomcat::configure レシピは、レイヤーの Configure ライフサイクルイベントに対応します。

```
include_recipe 'tomcat::context'  
# Optional: Trigger a Tomcat restart in case of a configure event, if relevant  
# settings in custom JSON have changed (e.g. java_opts/JAVA_OPTS):  
#include_recipe 'tomcat::container_config'
```

tomcat::configure レシピは、2 つの依存レシピを実行するメタレシピです。

1. tomcat::context レシピは、ウェブアプリケーションのコンテキスト設定ファイルを作成します。

このファイルは、次のセクションで説明するように、アプリケーションがインスタンスSQLとの通信に使用するJDBCリソースを設定します。設定イベントに応じてこのレシピを実行すると、データベースレイヤーが変更された場合に、レイヤーでウェブアプリケーションのコンテキスト設定ファイルを更新することができます。

2. tomcat::container_config Setup レシピは、コンテナ設定のあらゆる変更をキャプチャするために再度実行されます。

include の tomcat::container_config は、この例のようにコメントアウトされます。カスタムを使用して JSON Tomcat 設定を変更する場合は、コメントを削除できます。その後、Configure ライフサイクルイベントでは、tomcat::container_config を実行することにより「[tomcat::container_config](#)」で説明するように Tomcat 関連の設定ファイルを更新し、Tomcat サービスを再起動します。

tomcat::context

Tomcat クックブックを使用すると、アプリケーションは [J2EE DataSource](#) オブジェクトを使用して、別のインスタンスで実行できる MySQL データベースサーバーにアクセスできます。Tomcat では、各アプリケーションに対してウェブアプリケーションのコンテキスト設定ファイルを作成してインストールすることにより、接続を有効にできます。このファイルは、アプリケーションと、アプリ

セッションがデータベースとの通信に使用するJDBCリソースとの関係を定義します。詳細については、「[コンテキスト コンテナ](#)」を参照してください。

tomcat::context レシピの主な目的は、この設定ファイルを作成することです。

```
include_recipe 'tomcat::service'

node[:deploy].each do |application, deploy|
  context_name = deploy[:document_root].blank? ? application : deploy[:document_root]

  template "context file for #{application} (context name: #{context_name})" do
    path ::File.join(node['tomcat']['catalina_base_dir'], 'Catalina', 'localhost',
"#{context_name}.xml")
    source 'webapp_context.xml.erb'
    owner node['tomcat']['user']
    group node['tomcat']['group']
    mode 0640
    backup false
    only_if { node['datasources'][context_name] }
    variables(:resource_name => node['datasources'][context_name], :webapp_name =>
application)
    notifies :restart, resources(:service => 'tomcat')
  end
end
```

Tomcat クックブック属性に加えて、このレシピは、[スタックが Configure イベントでインストールするスタック設定およびデプロイ属性](#)を使用します。AWS OpsWorks AWS OpsWorks スタックサービスは、データバグまたは検索を使用してレシピが通常取得する情報を含む属性を各インスタンスのノードオブジェクトに追加し、各インスタンスにその属性をインストールします。それらの属性に含まれるのは、スタック設定、デプロイされるアプリケーション、ユーザーに必要なカスタムデータに関する詳細情報です。レシピは、標準の Chef ノード構文を使用することで、スタック設定およびデプロイ属性からデータを取得できます。詳細については、「[スタック設定およびデプロイメント属性](#)」を参照してください。Chef 11.10 スタックの場合、Chef の検索を使用してスタック設定およびデプロイデータを取得することもできます。詳細については、「[Chef の検索の使用](#)」を参照してください。

deploy 属性とは、[:deploy] 名前空間を指します。この名前空間には、コンソールまたはを通じて定義された、または AWS OpsWorks スタックサービスによって生成されたAPIデプロイ関連の属性が含まれます。deploy 属性には、デプロイされる各アプリケーションの属性が含まれています。この属性の名前はアプリケーションの短縮名です。各アプリケーション属性には、ドキュメント

のルート (`[:deploy][:appname][:document_root]`) など、アプリケーションの特性を示す一連の属性が含まれています。

context レシピは、最初に、[tomcat::service](#) を呼び出すことにより、この Chef 実行に対してサービスが定義されるようにします。次に、設定ファイルの名前を表す `context_name` 変数を定義します。拡張子の `.xml` は除外されます。デフォルトのドキュメントのルートを使用する場合、`context_name` はアプリケーションの短縮名に設定されます。それ以外の場合は、指定されたドキュメントのルートに設定されます。で説明されている例では[スタックの作成とアプリケーションの実行](#)、ドキュメントのルートを に設定しているため "ROOT"、コンテキストは ROOT で、設定ファイルの名前は `ROOT.xml` です。

レシピの大部分は、デプロイされているアプリケーションのリストを対象として機能し、各アプリケーションに対して `webapp_context.xml.erb` テンプレートを使用してコンテキスト設定ファイルを作成します。この例では、1つのアプリケーションのみをデプロイしますが、`deploy` 属性でアプリケーションのリストとして定義する必要があります。

`webapp_context.xml.erb` テンプレートは、オペレーティングシステム固有でないため、`templates` ディレクトリの `default` サブディレクトリに配置されています。

このレシピは、次のように設定ファイルを作成します。

- デフォルトの属性値を使用すると、設定ファイル名は `context_name.xml` に設定され、`/etc/tomcat6/Catalina/localhost/` ディレクトリにインストールされます。

スタック設定属性の `['datasources']` ノードには 1 つ以上の属性が含まれており、それぞれが、関連付けられたアプリケーションがデータベースとの通信に使用する JDBC データリソースにコンテキスト名をマッピングします。「」で後述するように、スタックの作成 JSON 時にノードとその内容は `custom` で定義されます [スタックの作成とアプリケーションの実行](#)。この例では、ROOT コンテキスト名を `jdbc/mydb` という名前の JDBC リソースに関連付ける単一の属性があります。

- デフォルトの属性値を使用すると、ファイルのユーザーとグループはどちらも、Tomcat パッケージの `tomcat` (Amazon Linux の場合) または `tomcat6` (Ubuntu の場合) によって定義された値に設定されます。
- `template` リソースは、`['datasources']` ノードが存在し、`context_name` 属性を含む場合にのみ、設定ファイルを作成します。
- `template` リソースは、2 つの変数 `resource_name` と `webapp_name` を定義します。

`resource_name` は `context_name` に関連付けられているリソース名に設定され、`webapp_name` はアプリケーションの短縮名に設定されます。

- `template` リソースは、Tomcat サービスを再起動し、変更を読み込んでアクティブにします。

`webapp_context.xml.erb` テンプレートは、属性の固有のセットを使用する `Context` 要素を含む `Resource` 要素で構成されています。

`Resource` 属性は、コンテキスト設定の特性を示します。

- `name` - JDBCリソース名。で定義された `resource_name` 値に設定されま
す `tomcat::context`。

たとえば、リソース名は `jdbc/mydb` に設定されます。

- `auth` と `type` - これらは `JDBCDataSource` 接続の標準設定です。
- `maxActive`、`maxIdle`、`maxWait`- アクティブ接続とアイドル接続の最大数、および接続が返されるまでの最大待機時間。
- `username` (ユーザーネーム) および `password` (パスワードパスワード) - データベースのユーザー名およびルートパスワード。 `deploy` 属性から取得されます。
- `driverClassName`- JDBCドライバーのクラス名。MySQLドライバーに設定されます。
- `url` - 接続 URL。

プレフィックスはデータベースによって異なります。My の場合は、Postgres `jdbc:mysqlSQLjdbc:postgresql` の場合は、SQLServer `jdbc:sqlserver` の場合はに設定する必要があります。この例では、を URL に設定します。ここで `jdbc:mysql://host_IP_Address:3306:simplejsp`、`simplejsp` はアプリケーションの短縮名です。

- `factory` – データベースに必要な `DataSourceSQL` ファクトリ。

この設定ファイルの詳細については、Tomcat Wiki の [「Using DataSources」](#) トピックを参照してください。

Deploy レシピ

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Deploy レシピは、レイヤーの Deploy [ライフサイクル](#) イベントに割り当てられます。これは通常、アプリケーションをデプロイするたびにスタックのすべてのインスタンスで発生しますが、オプションでイベントを指定されたインスタンスのみに制限できます。AWS OpsWorks スタックは、Setup レシピの完了後に、新しいインスタンスで Deploy レシピも実行します。Deploy レシピの主な目的は、コードと関連ファイルをリポジトリからアプリケーションサーバーレイヤーのインスタンスにデプロイすることです。ただし、多くの場合、その他のレイヤーでも Deploy レシピを実行します。これにより、これらのレイヤーのインスタンスで、たとえば、新しくデプロイされたアプリケーションに応じて設定を更新できるようになります。Deploy レシピを実装するときは、Deploy イベントが必ずしもアプリケーションがインスタンスにデプロイされることを意味するわけではないことに注意してください。アプリケーションがスタック内の別のインスタンスにデプロイされていることを単に通知することで、そのインスタンスで必要な更新ができるようにしているだけの場合もあります。このレシピは、適切に応答できる必要があります。それは、何も処理を実行しない場合があることを意味します。

AWS OpsWorks スタックは、標準アプリケーションタイプのアプリケーションを、対応する組み込みアプリケーションサーバーレイヤーに自動的にデプロイします。アプリケーションをカスタムレイヤーにデプロイするには、アプリケーションのファイルをリポジトリからインスタンスの適切な場所にダウンロードするカスタム Deploy レシピを実装する必要があります。ただし、多くの場合、組み込みの [デプロイクックブック](#) を使用してデプロイの一部の側面を処理することで、記述する必要のあるコードの量を制限することができます。たとえば、サポートされているリポジトリの 1 つにファイルを保存する場合、リポジトリからレイヤーのインスタンスへのファイルダウンロードの詳細を組み込みのクックブックで処理できます。

tomcat::deploy レシピは、Deploy ライフサイクルイベントに割り当てられます。

```
include_recipe 'deploy'

node[:deploy].each do |application, deploy|
  opsworks_deploy_dir do
    user deploy[:user]
    group deploy[:group]
    path deploy[:deploy_to]
  end
end
```

```
opsworks_deploy do
  deploy_data deploy
  app application
end
...
```

tomcat::deploy レシピは、アプリケーション固有ではないデプロイの側面に組み込みのデプロイクックブックを使用します。deploy レシピ (組み込み deploy::default レシピの省略表現) は、deploy 属性のデータに基づいてユーザー、グループなどのセットアップの詳細を処理する組み込みレシピです。

このレシピは、2 つの組み込みの Chef 定義 opsworks_deploy_dir と opsworks_deploy を使用して、アプリケーションをインストールします。

opsworks_deploy_dir 定義は、アプリケーションのデプロイからのデータに基づいてディレクトリ構造を設定します。JSON。定義は、基本的にはリソース定義をパッケージするための便利な方法であり、クックブックの definitions ディレクトリに配置されています。レシピでは、リソースと同様に定義を使用できますが、この定義自体には、定義に含まれた単なるリソースである関連付けられたプロバイダーはありません。基盤となるリソース定義に渡されるレシピ内の変数は定義できます。tomcat::deploy レシピは user、デプロイからのデータに基づいて group、および path 変数を設定します。これらの変数は、定義の [directory リソース](#) に渡されます。このリソースはディレクトリを管理します。

Note

`[:opsworks][:deploy_user][:user]` 属性と `[:opsworks][:deploy_user][:group]` 属性により、デプロイされているアプリケーションのユーザーとグループが定義されます。これらの属性は、[組み込みのデプロイクックブックの `deploy.rb` 属性ファイル](#) で定義されています。`[:opsworks][:deploy_user][:user]` の初期値は `deploy` です。`[:opsworks][:deploy_user][:group]` のデフォルト値は、インスタンスのオペレーティングシステムによって異なります。

- Ubuntu インスタンスでは、デフォルトのグループは `www-data` です。
- Nginx と Unicorn を使用する Rails アプリケーションアプリケーションサーバーレイヤーのメンバーである Amazon Linux インスタンスの場合、デフォルトのグループは `nginx` です。
- その他のすべての Amazon Linux インスタンスの場合、デフォルトのグループは `apache` です。

カスタムJSONまたはカスタム属性ファイルを使用して適切な属性を上書きすることで、設定を変更できます。詳細については、「[属性の上書き](#)」を参照してください。

その他の定義 `opsworks_deploy` は、アプリケーションのコードとリポジトリの関連ファイルの確認、およびそれらのインスタンスへのデプロイの詳細を、`deploy` 属性のデータに基づいて処理します。この定義は、任意のアプリケーションタイプに使用できます。ディレクトリ名などのデプロイの詳細は、コンソールまたは `awsopsworkscli` を通じて指定APIされ、`deploy` 属性を入力します。ただし、`opsworks_deploy` は、Git、Subversion、S3、の4つの[サポートされているリポジトリタイプ](#)でのみ機能します。異なるリポジトリのタイプを使用する場合は、このコードを自分で実装する必要があります。

アプリケーションのファイルを、Tomcat の `webapps` ディレクトリにインストールします。一般的な方法は、ファイルを `webapps` に直接コピーすることです。ただし、AWS OpsWorks スタックのデプロイは、インスタンスに最大5つのバージョンのアプリケーションを保持するように設計されているため、必要に応じて以前のバージョンにロールバックできます。したがって、AWS OpsWorks スタックは以下を実行します。

1. `/srv/www/my_1st_jsp/releases/20130731141527` など、名前にタイムスタンプが含まれた明確に区別できるディレクトリにアプリケーションをデプロイします。
2. この一意のディレクトリに、`current` という名前が付いた `/srv/www/my_1st_jsp/current` などのシンボリックリンクを作成します。
3. `webapps` ディレクトリからステップ2で作成した `current` へのシンボリックリンクがまだない場合は、これを作成します。

以前のバージョンにロールバックする必要がある場合は、該当するタイムスタンプが含まれた明確に区別できるディレクトリをポイントするように `current` シンボリックリンクを変更します。例えば、`/srv/www/my_1st_jsp/current` のリンクターゲットを変更します。

`tomcat::deploy` の中間のセクションは、シンボリックリンクをセットアップします。

```
...
current_dir = ::File.join(deploy[:deploy_to], 'current')
webapp_dir = ::File.join(node['tomcat']['webapps_base_dir'],
deploy[:document_root].blank? ? application : deploy[:document_root])
```

```
# opsworks_deploy creates some stub dirs, which are not needed for typical webapps
ruby_block "remove unnecessary directory entries in #{current_dir}" do
  block do
    node['tomcat']['webapps_dir_entries_to_delete'].each do |dir_entry|
      ::FileUtils.rm_rf(::File.join(current_dir, dir_entry), :secure => true)
    end
  end
end

link webapp_dir do
  to current_dir
  action :create
end
...
```

このレシピは、まず2つの変数 `current_dir` と `webapp_dir` を作成して、`current` ディレクトリと `webapp` ディレクトリをそれぞれ表します。次に、`link` リソースを使用して、`webapp_dir` を `current_dir` にリンクします。AWS OpsWorks スタック `deploy::default` レシピは、この例に必要なではないスタブディレクトリをいくつか作成するため、抜粋の中央部分では削除します。

`tomcat::deploy` の最後の部分は、必要に応じて Tomcat サービスを再起動します。

```
...
include_recipe 'tomcat::service'

execute 'trigger tomcat service restart' do
  command '/bin/true'
  not_if { node['tomcat']['auto_deploy'].to_s == 'true' }
  notifies :restart, resources(:service => 'tomcat')
end
end

include_recipe 'tomcat::context'
```

このレシピは、最初に `tomcat::service` を実行して、この Chef 実行に対してサービスが定義されるようにします。次に、[execute リソース](#) を使用して、再起動するようにサービスに通知しますが、それは `['tomcat']['auto_deploy']` が `'true'` に設定されている場合のみです。その他の場合、Tomcat は `webapps` ディレクトリ内の変更をリッスンします。これにより、明示的な Tomcat サービスの再起動が不必要になります。

Note

execute リソースは実際には実質的なことを何も実行せず、/bin/true は単に成功コードを返すダミーのシェルスクリプトです。単に再起動通知を生成するための便利な方法として使用されています。前に説明したように、通知を使用することで、サービスが必要以上に頻繁に再起動されなくなります。

最後に、tomcat::deploy は tomcat::context を実行し、バックエンドデータベースが変更されている場合は、ウェブアプリケーションのコンテキスト設定ファイルを更新します。

スタックの作成とアプリケーションの実行

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、Tomcat クックブックを使用して、Simple という名前のシンプルな Java サーバーページ (JSP) アプリケーションを実行する基本的なスタック設定を実装する方法を示します。スタックは、という名前の Tomcat ベースのカスタムレイヤー TomCustom と MySQL レイヤーで構成されます。SimpleJSP は にデプロイ TomCustom され、MySQL データベースからのいくつかの情報が表示されます。AWS OpsWorks スタックの使用法の基本にまだ慣れていない場合は、まず「」をお読みください [Chef 11 Linux スタックの使用開始](#)。

シンプルなJSPアプリケーション

SimpleJSP アプリケーションは、データベース接続を設定し、スタックの MySQL データベースからデータを取得する方法の基本を示します。

```
<html>
  <head>
    <title>DB Access</title>
  </head>
  <body>
```

```
<%@ page language="java" import="java.sql.*,javax.naming.*,javax.sql.*" %>
<%
    StringBuffer output = new StringBuffer();
    DataSource ds = null;
    Connection con = null;
    Statement stmt = null;
    ResultSet rs = null;
    try {
        Context initCtx = new InitialContext();
        ds = (DataSource) initCtx.lookup("java:comp/env/jdbc/mydb");
        con = ds.getConnection();
        output.append("Databases found:<br>");
        stmt = con.createStatement();
        rs = stmt.executeQuery("show databases");
        while (rs.next()) {
            output.append(rs.getString(1));
            output.append("<br>");
        }
    }
    catch (Exception e) {
        output.append("Exception: ");
        output.append(e.getMessage());
        output.append("<br>");
    }
    finally {
        try {
            if (rs != null) {
                rs.close();
            }
            if (stmt != null) {
                stmt.close();
            }
            if (con != null) {
                con.close();
            }
        }
        catch (Exception e) {
            output.append("Exception (during close of connection): ");
            output.append(e.getMessage());
            output.append("<br>");
        }
    }
}
%>
<%= output.toString() %>
```



```
</body>
</html>
```

SimpleJSP は、DataSource オブジェクトを使用して MySQL データベースと通信します。Tomcat は、[ウェブアプリケーションのコンテキスト設定ファイル](#)内のデータを使用し、DataSource オブジェクトを作成して初期化し、論理名に結合します。次に、Java 命名およびディレクトリインターフェイス (JNDI) 命名サービスに論理名を登録します。適切な DataSource オブジェクトのインスタンスを取得するには、InitialContext オブジェクトを作成し、リソースの論理名をオブジェクトの lookup メソッドに渡します。これにより、適切なオブジェクトを取得します。シンプルなJSP例の論理名にはjava:comp/env/jdbc/mydb、次のコンポーネントがあります。

- ルート名前空間の java。名前の残りの部分とはコロン (:) で区切られます。
- 任意の追加の名前空間。スラッシュ (/) で区切られます。

Tomcat は、comp/env 名前空間にリソースを自動的に追加します。

- リソース名。ウェブアプリケーションのコンテキスト設定ファイルで定義され、名前空間とはスラッシュで区切られます。

この例では、リソース名は jdbc/mydb です。

データベースへの接続を確立するために、SimpleJSP は以下を実行します。

1. DataSource オブジェクトの getConnection メソッドを呼び出し、Connection オブジェクトを返します。
2. Connection オブジェクトの createStatement メソッドを呼び出し、データベースとの通信に使用する Statement オブジェクトを作成します。
3. 適切な Statement メソッドを呼び出すことで、データベースと通信します。

サーバーのデータベースを一覧表示するSHOWDATABASESクエリexecuteQueryを実行するシンプルなJSP呼び出し。

executeQuery メソッドは、クエリの結果を含む ResultSet オブジェクトを返します。SimpleJSPは、返されたResultSetオブジェクトからデータベース名を取得し、それらを連結して出力文字列を作成します。最後に、この例では ResultSet、Statement、および Connection の各オブジェクトを閉じます。JSP および の詳細についてはJDBC、[JavaServer 「ページテクノロジ」とJDBC基本](#)」をそれぞれ参照してください。

スタックで SimpleJSP を使用するには、リポジトリに配置する必要があります。サポートされている任意のリポジトリを使用できますが、次のセクションで説明するサンプルスタックで SimpleJSP を使用するには、パブリック S3 アーカイブに配置する必要があります。その他の標準のリポジトリの使用方法の詳細については、「[クックブックリポジトリ](#)」を参照してください。

Simple JSPを S3 アーカイブリポジトリに配置するには

1. コード例を `simplejsp.jsp` という名前のファイルにコピーし、そのファイルを `simplejsp` という名前のディレクトリに配置します。
2. `.zip` ディレクトリの `simplejsp` アーカイブを作成します。
3. パブリックの Amazon S3 バケットを作成し、`simplejsp.zip` をそのバケットにアップロードして、ファイルをパブリックにします。

このタスクを実行する方法については、「[Amazon Simple Storage Service の使用開始](#)」を参照してください。

スタックの作成

SimpleJSP を実行するには、次のレイヤーを持つスタックが必要です。

- バックエンド MySQL サーバーをサポートする MySQL レイヤー。
- Tomcat サーバー インスタンスをサポートするために Tomcat クックブックを使用するカスタムレイヤー。

スタックを作成するには

1. AWS OpsWorks スタックダッシュボードで、スタックの追加をクリックして新しいスタックを作成し、詳細 >> をクリックしてすべてのオプションを表示します。スタックを次のように設定します。
 - 名前 - ユーザー定義のスタック名。この例では `TomStack` を使用します。
 - [Use custom Chef cookbooks] (カスタムChefのクックブックを使用) - トグルを [Yes] (はい) に設定します。いくつかの追加オプションが表示されます。
 - Repository type (リポジトリタイプ) - Git
 - リポジトリ URL - `git://github.com/amazonwebservices/opsworks-example-cookbooks.git`。
 - カスタム Chef JSON - 次の JSON を追加します。

```
{
  "tomcat": {
    "base_version": 7,
    "java_opts": "-Djava.awt.headless=true -Xmx256m"
  },
  "datasources": {
    "ROOT": "jdbc/mydb"
  }
}
```

残りのオプションについては、デフォルトをそのまま使用できます。

カスタムは以下JSONを実行します。

- Tomcat クックブックの ['base_version'] 属性を上書きして Tomcat のバージョンを 7 に設定します。デフォルト値は 6 です。
- Tomcat クックブックの ['java_opts'] 属性を上書きして、インスタンスがヘッドレスであることを指定し、JVM最大ヒープサイズを 256MB に設定します。デフォルト値では、Amazon Linux を実行しているインスタンスのオプションは設定されません。
- で説明されているように、JDBCリソース名 (jdbc/mydb) をウェブアプリケーションコンテキスト名 (ROOT) に割り当てる ['datasources'] 属性値を指定します [tomcat::context](#)。

この最後の属性にはデフォルト値がないため、カスタム で設定する必要があります JSON。



The screenshot shows the 'Configuration Management' section of the AWS OpsWorks console. It includes three main settings:

- Chef version:** Three radio buttons are shown: '11.10' (selected, labeled 'NEW DEFAULT'), '11.4', and '0.9' (labeled 'DEPRECATED'). A link 'Need a different configuration management option? Let us know.' is present.
- Use custom Chef cookbooks:** A toggle switch is set to 'No'.
- Custom JSON:** A text area contains the following JSON:

```
{
  "tomcat": {
    "base_version": 7,
    "java_opts": "-Djava.awt.headless=true -Xmx256m"
  },
  "datasources": {
    "ROOT": "jdbc/mydb"
  }
}
```

Below the text area, there is a note: 'Enter custom JSON that is passed to your Chef recipes for all instances in your stack. You can use this to override and customize built-in recipes or pass variables to your own recipes. [Learn more.](#)'

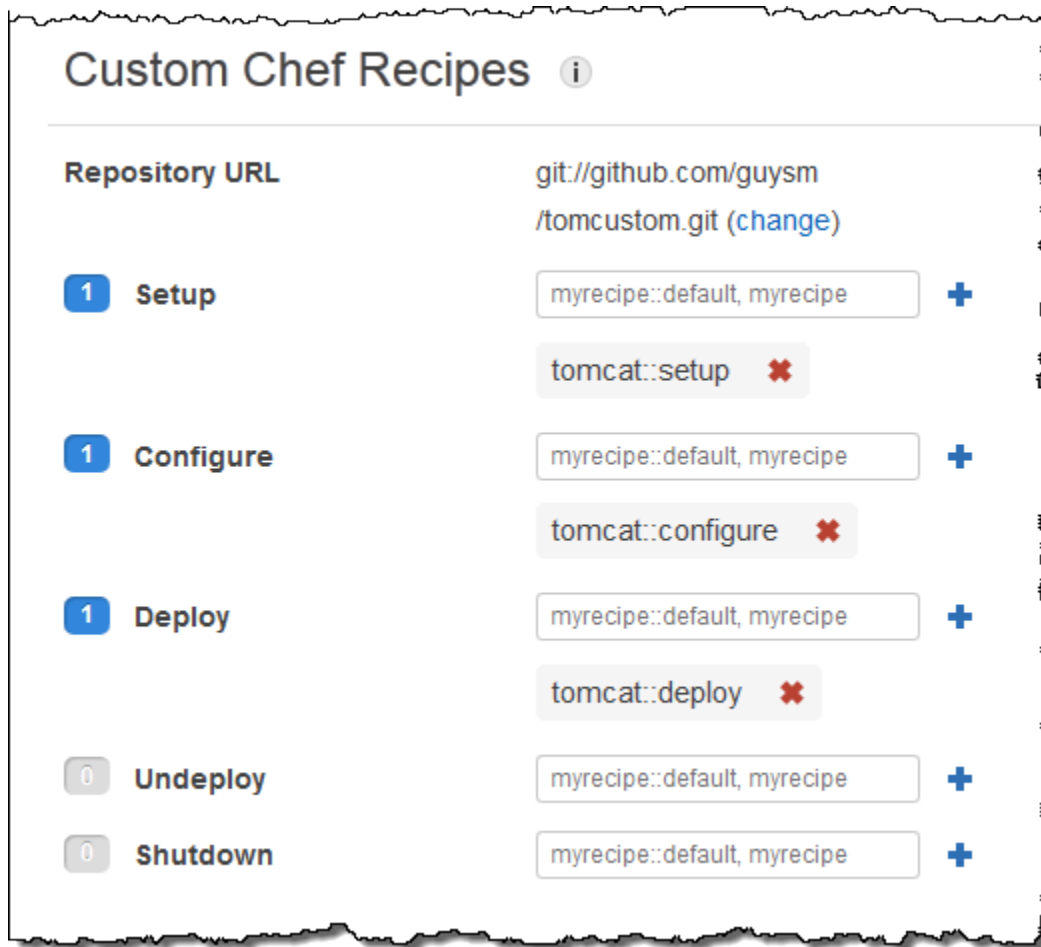
2. [Add a layer] をクリックします。レイヤータイプで、MySQL を選択します。次に、[Add Layer] をクリックします。
3. ナビゲーションペインで [Instances] をクリックし、[Add an instance] をクリックします。[Add Instance] をクリックして、デフォルトの値をそのまま使用します。インスタンスに関する行で、[start] をクリックします。
4. [Layers] (レイヤー) ページに戻り、[+ Layer] (+ レイヤー) をクリックしてレイヤーを追加します。[Layer type] では [Custom] をクリックします。このテンプレートでは、**TomCustom** と **tomcustom** を、それぞれレイヤーの名前と短縮名として使用します。

Add Layer

Layer type	<input type="text" value="Custom"/>
<p>The Custom layer allows you to create a fully customized layer. Standard recipes handle basic setup and configuration for the layer instances, and you implement custom Chef recipes to install and configure any required software. You can create as many custom layers as you require. Learn more.</p>	
Name	<input type="text" value="TomCustom"/>
Short name	<input type="text" value="tomcustom"/>

[Cancel](#)

5. [Layers] (レイヤー) ページで、カスタムレイヤーに対して、[Recipes] (レシピ) をクリックし、[Edit] (編集) をクリックします。[Custom Chef Recipes] で、次のように、Tomcat クックブックのレシピをレイヤーのライフサイクルイベントに割り当てます。
 - [Setup] には **tomcat::setup** を入力し、[+] をクリックします。
 - [Configure] には **tomcat::configure** を入力し、[+] をクリックします。
 - [Deploy] には **tomcat::deploy** を入力し、[+] をクリックします。次に、[Save] をクリックします。



6. ナビゲーションペインで [Apps] をクリックし、[Add an app] をクリックします。次のオプションを指定し、[Add App] をクリックします。

- Name – アプリケーションの名前。この例では SimpleJSP を使用し、スタックによって生成される短縮名は simplejsp AWS OpsWorks になります。
- App type (アプリケーションタイプ) - このオプションでは Other (その他) を設定します。

AWS OpsWorks スタックは、関連するサーバーインスタンスに標準アプリケーションタイプを自動的にデプロイします。[App type] でこれ以外を選択すると、AWS OpsWorks スタックは単純に Deploy レシピを実行し、このレシピでデプロイを処理することになります。

- Document root (ドキュメントルート) - このオプションを **ROOT** に設定します。

Document root (ドキュメントルート) の値は、コンテキスト名を指定します。

- Repository type (リポジトリタイプ) - このオプションを [S3 Archive (S3 アーカイブ)] に設定します。

- リポジトリ URL- 先ほどURL作成したアプリの Amazon S3 に設定します。

その他のオプションについては、デフォルトの設定を使用します。

App New

Settings

Name

App type

Document root

Application Source

Repository type

Repository URL

User name

Password

Add Domains

7. インスタンスページを使用して、インスタンスを TomCustom レイヤーに追加して起動します。AWS OpsWorks スタックは Setup レシピの完了後に新しいインスタンスで Deploy レシピを自動的に実行するため、インスタンスを起動すると Simple もデプロイされます JSP。
8. TomCustom インスタンスがオンラインになったら、インスタンスページでインスタンス名をクリックして詳細を表示します。パブリック IP アドレスをコピーします。次に、`http://publicIP/tc/` URL のように を構築します `appname.jsp`。この例では、これは の URL ようになります `http://50.218.191.172/tc/simplejsp.jsp`。

Note

Tomcat にリクエストURLを転送する Apache は、デフォルトの ['tomcat'] ['apache_tomcat_bind_path'] 属性 に設定されます /tc/。SimpleJSP ドキュメントのルートは、ROOT に解決される特殊な値である に設定されます /。したがって、URL は 「.../tc/simplejsp.jsp」 です。

9. 前のステップURLの をブラウザに貼り付けます。次のように表示されます。

```
Databases found:
information_schema
simplejsp
test
```

Note

スタックに MySQL インスタンスがある場合、AWS OpsWorks Stacks はアプリごとに、アプリの短縮名で という名前のデータベースを自動的に作成します。

スタック設定およびデプロイメント属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックがインスタンスでコマンドを実行する場合、例えば Deploy ライフサイクルイベントにตอบสนองして deploy コマンドを実行すると、スタックの現在の設定を記述する属性のセットがインスタンスのノードオブジェクトに追加されます。デプロイイベントと [Execute Recipes スタックコマンド](#) の場合、AWS OpsWorks スタックはデプロイ属性をインストールします。これにより、追加のデプロイ情報が提供されます。ノードオブジェクトの詳細については、「[属性の上書き](#)」

を参照してください。完全修飾ノード名などの、一般的に使用されるスタック設定とデプロイメント属性の一覧については、「[スタック設定およびデプロイ属性: Linux](#)」および「[組み込みクックブックの属性](#)」を参照してください。

Note

Linux スタックでは、エージェントの CLI [get_json コマンド](#)を使用して、JSONオブジェクトとしてフォーマットされたこれらの属性の完全なリストを取得できます。

以降のセクションでは、以下のような構成の単純なスタックの Configure イベントと Deploy イベントに関連する属性を示します。

- 2つのインスタンスを持つPHPアプリケーションサーバーレイヤー
- 1つのインスタンスを持つ1つの HAProxy Layer

例は、PHPApp Server インスタンスの1つである php-app1 からのものです。便宜上、属性はJSON オブジェクトとしてフォーマットされます。オブジェクトの構造は属性の完全修飾名に対応付けられます。例えば、`node[:opsworks][:ruby_version]` 属性はJSON表現で次のように表示されます。

```
{
  "opsworks": {
    ...
    "ruby_version": "1.8.7",
    ...
  }
}
```

トピック

- [属性の設定](#)
- [デプロイ属性](#)

属性の設定

次のJSONオブジェクトは、Configure イベントの属性を示しています。これは、インスタンスがオンラインになったとき、またはオフラインになったときにスタック内のすべてのインスタンスで発

生じます。属性には、組み込みスタック設定属性と、イベントの前にスタックに定義された[カスタムJSON属性](#) (この例ではなし) が含まれます。長さの関係で編集されています。さまざまな属性の詳細な説明については、「[スタック設定およびデプロイ属性: Linux](#)」および「[組み込みクックブックの属性](#)」を参照してください。

```
{
  "opsworks": {
    "layers": {
      "php-app": {
        "id": "4a2a56c8-f909-4b39-81f8-556536d20648",
        "instances": {
          "php-app2": {
            "elastic_ip": null,
            "region": "us-west-2",
            "booted_at": "2013-02-26T20:41:10+00:00",
            "ip": "192.0.2.0",
            "aws_instance_id": "i-34037f06",
            "availability_zone": "us-west-2a",
            "instance_type": "c1.medium",
            "private_dns_name": "ip-10-252-0-203.us-west-2.compute.internal",
            "private_ip": "10.252.0.203",
            "created_at": "2013-02-26T20:39:39+00:00",
            "status": "online",
            "backends": 8,
            "public_dns_name": "ec2-192-0-2-0.us-west-2.compute.amazonaws.com"
          },
          "php-app1": {
            ...
          }
        },
        "name": "PHP Application Server"
      },
      "lb": {
        "id": "15c86142-d836-4191-860f-f4d310440f14",
        "instances": {
          "lb1": {
            ...
          }
        },
        "name": "Load Balancer"
      }
    }
  },
}
```

```
"agent_version": "104",
"applications": [

],
"stack": {
  "name": "MyStack"
},
"ruby_version": "1.8.7",
"sent_at": 1361911623,
"ruby_stack": "ruby_enterprise",
"instance": {
  "layers": [
    "php-app"
  ],
  "region": "us-west-2",
  "ip": "192.0.2.0",
  "id": "45ef378d-b87c-42be-a1b9-b67c48edafd4",
  "aws_instance_id": "i-32037f00",
  "availability_zone": "us-west-2a",
  "private_dns_name": "ip-10-252-84-253.us-west-2.compute.internal",
  "instance_type": "c1.medium",
  "hostname": "php-app1",
  "private_ip": "10.252.84.253",
  "backends": 8,
  "architecture": "i386",
  "public_dns_name": "ec2-192-0-2-0.us-west-2.compute.amazonaws.com"
},
"activity": "configure",
"rails_stack": {
  "name": null
},
"deployment": null,
"valid_client_activities": [
  "reboot",
  "stop",
  "setup",
  "configure",
  "update_dependencies",
  "install_dependencies",
  "update_custom_cookbooks",
  "execute_recipes"
]
},
"opsworks_custom_cookbooks": {
```

```
"recipes": [
],
"enabled": false
},
"recipes": [
  "opsworks_custom_cookbooks::load",
  "opsworks_ganglia::configure-client",
  "ssh_users",
  "agent_version",
  "mod_php5_apache2::php",
  "php::configure",
  "opsworks_stack_state_sync",
  "opsworks_custom_cookbooks::execute",
  "test_suite",
  "opsworks_cleanup"
],
"opsworks_rubygems": {
  "version": "1.8.24"
},
"ssh_users": {
},
"opsworks_bundler": {
  "manage_package": null,
  "version": "1.0.10"
},
"deploy": {
}
}
```

情報の大部分は、名前空間とも呼ばれる `opsworks` 属性に含まれています。主な属性を次に示します。

- `layers` 属性 - それぞれの属性がスタックの1つのレイヤーの設定を説明している、属性のセットです。

レイヤーは短縮名 (この例では `php-app` と `lb`) で識別されます。その他の短縮名の詳細については、「[AWS OpsWorks スタックレイヤーリファレンス](#)」を参照してください。

- `instances` 属性 - すべてのレイヤーにはインスタンスの短縮名が付けられた `instances` 要素があり、レイヤーのオンラインインスタンスごとに1つの属性が含まれています。

PHP App Server レイヤーには、php-app1と の 2 つのインスタンスがありますphp-app2。HAProxy Layer には 1 つのインスタンス (lb1) があります。

Note

instances 要素には、特定のスタック設定およびデプロイ属性が作成された時点でオンライン状態にあったインスタンスのみが含まれています。

- インスタンス属性 – 各インスタンス属性には、インスタンスのプライベート IP アドレスやプライベートDNS名など、インスタンスを特徴付ける属性のセットが含まれています。わかりやすいように、例には php-app2 属性の詳細のみを示します。他のインスタンスにも同様の情報が含まれています。
- applications - デプロイされるアプリケーションのリスト (この例では使用していません)。
- stack - スタック名 (この例では) MyStack
- instance - これらの属性がインストールされるインスタンス (この例では) php-app1 レシピは、この属性を使用して、インスタンスのパブリック IP アドレスなどの、レシピが実行されているインスタンスに関する情報を取得できます。
- activity - 属性を生成したアクティビティ (この例では Configure イベント)。
- rails_stack - Rails アプリケーションサーバーレイヤーが含まれるスタックの Rails スタックです。
- deployment - これらの属性がデプロイに関連付けられているかどうか。この例では、それらの属性が Configure イベントに関連付けられているため、null に設定されています。
- valid_client_activities - 有効なクライアントアクティビティのリストです。

opsworks 属性の後には、以下のような他の最上位属性がいくつか続きます。

- opsworks_custom_cookbooks - カスタムクックブックが有効化されているかどうかを示します。有効化されている場合、属性にはカスタムレシピのリストが含まれます。
- recipes - このアクティビティによって実行されたレシピです。
- opsworks_rubygems – インスタンス RubyGemsのバージョン。
- ssh_users – SSH ユーザーのリスト。この例ではなし。
- opsworks_bundler - Bundler のバージョンとそれが有効化されているかどうかを示します。
- deploy - デプロイメントアクティビティに関する情報です (この例にはありません)。

デプロイ属性

Deploy イベントまたは [Execute Recipes スタックコマンド](#) の属性は、組み込みのスタック設定およびデプロイ属性と、カスタムのスタック設定またはデプロイ属性で構成されます (この例にはありません)。次のJSONオブジェクトは、SimplePHP アプリケーションをスタックのPHPインスタンスにデプロイした Deploy イベントに関連付けられている php-app1 の属性を示しています。オブジェクトの多くは、前のセクションで説明した Configure イベントの属性に類似したスタック設定属性で構成されます。そのため、例では主にデプロイ固有の属性に焦点を当てています。さまざまな属性の詳細な説明については、「[スタック設定およびデプロイ属性: Linux](#)」および「[組み込みクックブックの属性](#)」を参照してください。

```
{
  ...
  "opsworks": {
    ...
    "activity": "deploy",
    "applications": [
      {
        "slug_name": "simplephp",
        "name": "SimplePHP",
        "application_type": "php"
      }
    ],
    "deployment": "5e6242d7-8111-40ee-bddb-00de064ab18f",
    ...
  },
  ...
  {
    "ssh_users": {
    },
    "deploy": {
      "simplephpapp": {
        "application": "simplephpapp",
        "application_type": "php",
        "environment_variables": {
          "USER_ID": "168424",
          "USER_KEY": "somepassword"
        },
      },
      "auto_bundle_on_deploy": true,
      "deploy_to": "/srv/www/simplephpapp",
      "deploying_user": "arn:aws:iam::123456789012:user/guysm",
```

```
"document_root": null,
"domains": [
  "simplephpapp"
],
"migrate": false,
"mounted_at": null,
"rails_env": null,
"restart_command": "echo 'restarting app'",
"sleep_before_restart": 0,
"ssl_support": false,
"ssl_certificate": null,
"ssl_certificate_key": null,
"ssl_certificate_ca": null,
"scm": {
  "scm_type": "git",
  "repository": "git://github.com/amazonwebservices/opsworks-demo-php-simple-
app.git",
  "revision": "version1",
  "ssh_key": null,
  "user": null,
  "password": null
},
"symlink_before_migrate": {
  "config/opsworks.php": "opsworks.php"
},
"symlinks": {
},
"database": {
},
"memcached": {
  "host": null,
  "port": 11211
},
"stack": {
  "needs_reload": false
}
},
}
```

opsworks 属性は、前のセクションで説明した例のものと同様です。以降のセクションでは、デプロイに特に関連する属性を取り上げています。

- activity - これらの属性に関連付けられているイベント (この例では Deploy イベント)。
- applications - 各アプリケーションの属性 (アプリケーションの名前、スラッグ名、タイプ) のリスト。

スラッグ名はアプリケーションの短縮名で、AWS OpsWorks スタックはアプリケーション名から生成します。SimplePHP のスラッグ名は simplephp です。

- deployment - デプロイを一意に識別するデプロイ ID。

deploy 属性には、デプロイされるアプリケーションに関する情報が含まれています。例えば、組み込み Deploy レシピによって、deploy 属性のデータが使用されて、適切なディレクトリ内のファイルがインストールされ、データベース接続ファイルが作成されます。deploy 属性には、デプロイされる各アプリケーションの属性が含まれています。この属性の名前はアプリケーションの短縮名です。各アプリケーション属性には、以下の属性が含まれています。

- environment_variables - アプリケーションに対して定義したすべての環境変数が含まれています。詳細については、「[環境可変](#)」を参照してください。
- domains - デフォルトではアプリケーションの短縮名 (この例では simplephpapp)。カスタムドメインを割り当てている場合は、同様にそれらがここに表示されます。詳細については、「[カスタムドメインの使用](#)」を参照してください。
- application - アプリケーションの短縮名。
- scm - この要素には、リポジトリ (この例では Git リポジトリ) からアプリケーションのファイルをダウンロードするために必要な情報が含まれています。
- database - スタックにデータベースレイヤーが含まれている場合は、データベースの情報。
- document_root - ドキュメントのルートです。この例では null に設定されており、ルートがパブリックであることを示しています。
- ssl_certificate_ca、ssl_support、ssl_certificate_key - アプリがSSLサポートしているかどうかを示します。サポートしている場合、ssl_certificate_key および ssl_certificate_ca 属性は対応する証明書に設定されます。
- deploy_to - アプリケーションのルートディレクトリ。

クックブック 101

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

通常、本番稼働レベルの AWS OpsWorks スタックスタックにはある程度の [カスタマイズ](#) が必要です。つまり、多くの場合、1 つ以上のレシピ、属性ファイル、またはテンプレートファイルを使用してカスタム Chef クックブックを実装します。このトピックでは、AWS OpsWorks スタック用のクックブックの実装に関するチュートリアルを紹介します。

AWS OpsWorks スタックがクックブックを使用する方法の詳細については、「」を参照してください。クックブックの簡単な一般的な概要が含まれています [クックブックとレシピ](#)。Chef レシピを実装してテストする方法の詳細については、「[Chef を利用したテスト中心のインフラストラクチャ \(第 2 版\)](#)」を参照してください。

チュートリアルの例は、2 つのセクションに分割されます。

- 「[クックブックの基本](#)」では、Chef に慣れていないユーザー向けの例となるウォークスルーがまとめられています。Chef に慣れたユーザーは、このセクションを省略できます。

この例では、パッケージのインストールやディレクトリの作成など、一般的なタスクを実行するクックブックを実装するための基本方法を説明します。プロセスを簡素化するために、仮想マシンのローカルに置かれた例の多くは [Vagrant](#) と [Test Kitchen](#) という 2 つの便利なツールを使用して実行します。「[クックブックの基本](#)」を開始する前に、「[Vagrant と Test Kitchen](#)」を読んで、この 2 つのツールのインストール方法と使用方法を学習してください。Test Kitchen ではまだ Windows がサポートされていないため、例はすべて Linux に当てはまるものです。それらの例を Windows に当てはめる方法は注意に示しています。

- [AWS OpsWorks スタックのクックブックの実装](#) では、Windows AWS OpsWorks スタック用のなど、スタック用のレシピを実装する方法について説明します。

また、Berkshelfを使って外部のクックブックを管理する方法など、より高度な内容も含まれています。「[クックブックの基本](#)」で示している例などは、Chef の使用経験がないユーザー向けに作

成されています。ただし AWS OpsWorks、スタックは Chef サーバーと少し異なるため、経験のある Chef ユーザーは少なくともこのセクションを読むことをお勧めします。

Vagrant と Test Kitchen

Linux インスタンス用のレシピを使用する場合、Vagrant と Test Kitchen は、学習や初期の開発およびテストに非常に便利なツールです。ここでは、Vagrant と Test Kitchen について簡単に説明し、インストール方法を示します。さらに、このツールの基本的な使用方法を理解し習得するためのウォークスルーも含まれています。Vagrant では Windows がサポートされていますが、Test Kitchen ではサポートされていません。そのため、Linux の例のみをこれらのツール用に示しています。

Vagrant

[Vagrant](#) は、仮想マシンでコードを実行しテストするための一貫した環境を実現します。Vagrant ボックスと呼ばれるさまざまな環境をサポートしており、それぞれが構成されたオペレーティングシステムを表します。AWS OpsWorks スタックの場合、対象の環境は Ubuntu、Amazon、または Red Hat Enterprise Linux (RHEL) ディストリビューションに基づいているため、例では主に という名前の Vagrant ボックスを使用します `opscode-ubuntu-12.04`。

Vagrant は Linux、Windows、および Macintosh の各システムに対応しているため、好みのワークステーションを使用して、任意のサポート対象オペレーティングシステムにレシピを実装し、テストすることができます。この章の例は Ubuntu Linux システムで作成されていますが、その手順を Windows システムや Macintosh システム用に読み替えるのは簡単です。

基本的に、Vagrant は仮想化プロバイダー用のラッパーです。ほとんどの例では、[VirtualBox](#) プロバイダーを使用しています。VirtualBox は Linux、Windows、および Macintosh システムで無料で利用できます。Vagrant チュートリアルでは、システムに がまだない場合はインストール手順 VirtualBox について説明します。Ubuntu ベースの環境は で実行できますが VirtualBox、Amazon Linux は Amazon EC2 インスタンスでのみ使用できます。ただし、CentOS などの同様のオペレーティングシステムを で実行することはできません。これは VirtualBox 初期の開発とテストに役立ちます。

その他のプロバイダーの詳細については、[Vagrant](#) のドキュメントを参照してください。特に、`vagrant-aws` プラグインプロバイダーでは、Amazon EC2 インスタンスで Vagrant を使用できます。このプロバイダーは、Amazon EC2 インスタンスでのみ利用可能な Amazon Linux でのレシピのテストに特に役立ちます。`vagrant-aws` プロバイダーは無料ですが、アカウントがあり AWS、使用する AWS リソースに対して料金を支払う必要があります。

次に、Vagrant の「[使用開始ウォークスルー](#)」を参照してください。このウォークスルーでは、ワークステーションに Vagrant をインストールする方法と、Vagrant の基本的な使用方法を見ることができます。この章の例では Git リポジトリを使用しないため、必要に応じて、ウォークスルーの関連する部分を省略してもかまいません。

Test Kitchen

[Test Kitchen](#) は、Vagrant におけるクックブックの実行とテストのプロセスを簡略化します。実際のところ、Vagrant を直接使用する必要はほとんどありません。Test Kitchen は、以下のような最も一般的なタスクを実行します。

- Vagrant でのインスタンスの起動。
- インスタンスへのクックブックの転送。
- インスタンスでのクックブックのレシピの実行。
- インスタンスでのクックブックのレシピのテスト。
- SSH を使用してインスタンスにログインします。

Test Kitchen gem を直接インストールする代わりに、[Chef DK](#) をインストールすることをお勧めします。Chef 自体に加えて、このパッケージには Test Kitchen、[Berkshelf](#)、[ChefSpec](#)、およびその他の便利なツールが含まれています。

次に、Test Kitchen の「[使用開始ウォークスルー](#)」を参照してください。ここでは、Test Kitchen を使用してレシピを実行し、テストするための基本的な方法が紹介されています。

Note

この章の例では、レシピを実行するための便利な方法として Test Kitchen を使用しています。例の実行に必要な情報は「手動確認」セクションにすべて記載されているため、このセクションを終了した後で「使用開始ウォークスルー」を中止してもかまいません。ただし、Test Kitchen は主に [bash 自動テストシステム \(BATS\) などのテストフレームワークをサポートするテストプラットフォーム](#)です。Test Kitchen を使用してレシピをテストする方法を学ぶには、どこかの時点でウォークスルーの残りを実行します。

クックブックの基本

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

クックブックを使用して、さまざまなタスクを実現できます。以下のトピックは Chef を初めて使用する方を対象に、クックブックを使用した一般的なタスクの実行方法について説明します。Test Kitchen ではまだ Windows がサポートされていないため、例はすべて Linux に当てはまるものです。それらの例を Windows に当てはめる方法は注意に示しています。Chef を初めて使用する場合は、Windows で作業するとしても、これらの例を実行することをお勧めします。このトピックの例のほとんどは、注意に示しているように、若干の変更で Windows インスタンスに使用できます。例のすべては仮想マシンで実行されるため、Linux コンピュータは必要ありません。日常使用しているワークステーションに Vagrant と Test Kitchen をインストールするだけです。

Note

Windows インスタンスでこれらのレシピを実行する場合、最も簡単な方法は、Windows スタックを作成して、スタックのインスタンスのいずれかでレシピを実行することです。AWS OpsWorks スタックの Windows インスタンスでレシピを実行する方法の詳細については、「」を参照してください [Windows インスタンスでのレシピの実行](#)。

先に進む前に、Vagrant および Test Kitchen がインストールされていることを確認し、使用開始のウォークスルーを完了させてください。詳細については、「[Vagrant と Test Kitchen](#)」を参照してください。

トピック

- [レシピの構造](#)
- [例 1: パッケージのインストール](#)
- [例 2: ユーザー管理](#)
- [例 3: ディレクトリの作成](#)

- [例 4: フロー制御の追加](#)
- [例 5: 属性の使用](#)
- [例 6: ファイルの作成](#)
- [例 7: コマンドとスクリプトの実行](#)
- [例 8: 管理サービス](#)
- [例 9: Amazon EC2 インスタンスの使用](#)
- [次のステップ](#)

レシピの構造

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

クックブックは主に、インスタンス上のさまざまなタスクを実行できる一連のレシピです。レシピを実行する方法をわかりやすくするために、シンプルな例を見てみましょう。以下にあげるのは組み込み [HAProxy Layer](#) の設定レシピです。現時点では全体的な構造に注目して詳細にはあまり気を取られないでください。詳細は後述の例で説明します。

```
package 'haproxy' do
  action :install
end

if platform?('debian','ubuntu')
  template '/etc/default/haproxy' do
    source 'haproxy-default.erb'
    owner 'root'
    group 'root'
    mode 0644
  end
end
```

```
include_recipe 'haproxy::service'

service 'haproxy' do
  action [:enable, :start]
end

template '/etc/haproxy/haproxy.cfg' do
  source 'haproxy.cfg.erb'
  owner 'root'
  group 'root'
  mode 0644
  notifies :restart, "service[haproxy]"
end
```

Note

作業レシピおよび関連ファイルのここに挙げた例やその他の例については、「[AWS OpsWorks スタック組み込みレシピ](#)」を参照してください。

例では、以下のセクションで説明するレシピの主要な要素に焦点を当てています。

トピック

- [リソース](#)
- [フロー制御](#)
- [内包レシピ](#)

リソース

レシピは、Chef リソースのセットの大部分を構成します。それぞれのレシピは、インストールするパッケージや開始するサービスなど、インスタンスの最終状態の特定の部分を指定します。例には 4 つのリソースがあります。

- インストールされたパッケージを表す package リソース、この例の [HAProxy サーバー](#)。
- この HAProxy 例のサービスであるサービスを表す service リソース。
- 指定されたテンプレートから作成されるファイルを表す 2 つの template リソース、この例の 2 つの HAProxy 設定ファイル。

リソースはインスタンスの状態を指定する宣言方法を提供します。バックグラウンドで各リソースにはプロバイダーが関連付けられ、パッケージのインストール、ディレクトリの作成と設定、サービスの開始など必要なタスクを実行します。タスクの詳細が特定のオペレーティングシステムに依存する場合、リソースは複数のプロバイダーを持ってシステムに応じて使い分けます。たとえば、Red Hat Linux システムでは package プロバイダーは yum を使用してパッケージをインストールします。Ubuntu Linux システムでは、package プロバイダーは apt-get を使用します。

以下の一般的な形式で Ruby コードブロックとしてリソースを実行します。

```
resource_type "resource_name" do
  attribute1 'value1'
  attribute2 'value2'
  ...
  action :action_name
  notifies : action 'resource'
end
```

要素は以下のとおりです。

リソースタイプ

(必須) この例では 3 個のリソースタイプを含んでいます。package、service、および template です。

リソース名

(必須) リソース名は特定のリソースを識別し、場合によっては属性のデフォルト値の 1 つとして使用されます。例では、package は haproxy という名前のパッケージリソースを示し、最初の template リソースは /etc/default/haproxy という名前の設定ファイルを表します。

属性

(オプション) 属性はリソース設定を指定するもので、リソースのタイプと設定方法によって異なります。

- 例にある template リソースは、作成されたファイルのソース、所有者、グループおよびモードを指定する一連の属性を明示的に定義します。
- 例にある package および service リソースは、どのような属性も明示的に定義しません。

通常、リソース名は必須属性のデフォルト値で、それだけで十分な場合もあります。たとえば、リソース名は package リソースの package_name 属性のデフォルト値であり、唯一の必須属性です。

また、リソースプロバイダーを実行する時に指定される、ガード属性と呼ばれる特殊な属性がいくつかあります。例えば、`only_if` 属性は指定された条件が満たされる場合だけリソースプロバイダーにアクション実行を指示します。HAProxy レシピはガード属性を使用しませんが、以下のいくつかの例で使用されます。

アクションおよび通知

(オプション) アクションおよび通知は、プロバイダーがどのタスクを実行するかを指定します。

- `action` はインストールや作成など指定したアクションを実行するようにプロバイダーに指示します。

各リソースには特定のリソースに依存する一連のアクションがあり、そのうちの1つはデフォルトのアクションです。例では、`package` リソースのアクションは `install` であり、プロバイダーにパッケージをインストールするように指示します。最初の `template` リソースには `action` 要素はないので、プロバイダーはデフォルトの `create` アクションを実行します。

- `notifies` はリソースの状態が変化した場合のみ、別のリソースのプロバイダーにアクションを実行するように指示します。

`notifies` は通常、`template` や `file` のようなリソースとともに使用され、設定ファイルを変更した後にサービスを再起動するなどのタスクを実行します。リソースにはデフォルトの通知はありません。通知が必要な場合は、リソースに明示的な `notifies` 要素が必要です。HAProxy レシピでは、2番目の `template` リソースは、関連付けられた設定ファイルが変更された場合にHAProxyサービスを再起動するように `haproxy service` リソースに通知します。

リソースはオペレーティングシステムによって異なる場合があります。

- 一部のリソースは Linux または Windows システムにのみ使用できます。

たとえば、[package](#) によって、Linux システムにパッケージがインストールされ、[windows_package](#) によって Windows システムにパッケージがインストールされます。

- 一部のリソースはいずれのオペレーティングシステムでも使用できますが、特定のシステムに固有の属性があります。

たとえば、[file](#) リソースは、Linux または Windows システムのいずれかに使用できますが、アクセス権を設定するための属性の個別のセットはありません。

標準的なリソースおよびそれぞれの使用可能な属性、アクション、通知についての説明は、「[リソースおよびプロバイダーについて](#)」を参照してください。

フロー制御

レシピは Ruby アプリケーションであるため、レシピにフロー制御を組み込むために Ruby の制御構造を使用できます。たとえば、レシピが異なるシステムで異なる動作をするように Ruby の条件付きロジックを使用できます。HAProxy レシピには、レシピが Debian または Ubuntu システムで実行されている場合にのみ、template リソースを使用して設定ファイルを作成する if ブロックが含まれています。

もう 1 つの一般的なシナリオとしては、ループを使用して異なる属性設定でリソースを複数回実行します。たとえば、異なるディレクトリで directory リソースを複数回実行するループを使用して、一連のディレクトリを作成できます。

Note

Ruby になじみがない場合は、「[Chef のための Ruby 基礎](#)」を参照してください。ほとんどのレシピに必要な知識を説明しています。

内包レシピ

include_recipe はコードの中に他のレシピを含め、レシピをモジュール化して複数のレシピで同じコードを再利用できます。ホストのレシピを実行すると、Chef はそれぞれの include_recipe の要素を指定されたレシピのコードに置き換えてからホストのレシピを実行します。内包レシピは、標準の Chef 構文である cookbook_name::recipe_name を使用して識別されます。この recipe_name では .rb 拡張子が省略されます。この例には、HAProxy サービス haproxy::service を表すレシピが 1 つ含まれています。

Note

Chef 11.10 以降で実行されているレシピに include_recipe を使用して他のクックブックからのレシピを含める場合、depends ステートメントを使用してクックブックの metadata.rb ファイル内の依存関係を宣言する必要があります。詳細については、「[レシピの実装: Chef 11.10](#)」を参照してください。

例 1: パッケージのインストール

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

パッケージのインストールはレシピのもっとも一般的な用途の 1 つで、パッケージによっては非常にシンプルにできます。たとえば、以下のレシピは Linux システムに Git をインストールします。

```
package 'git' do
  action :install
end
```

[package リソース](#) はパッケージのインストールを処理します。この例では、属性を指定する必要はありません。リソース名が `package_name` 属性のデフォルト値であり、これでパッケージを識別します。install アクションがプロバイダーにパッケージをインストールするように指示します。install を省略することでコードをよりシンプルにできます。これは package リソースのデフォルトアクションです。レシピを実行すると、Chef は適切なプロバイダーを使用してパッケージをインストールします。例として使用する Ubuntu システムでは、プロバイダーは apt-get を呼び出して Git をインストールします。

Note

Windows システムにソフトウェアをインストールするには、多少異なる手順が必要です。詳細については、「[Windows ソフトウェアのインストール](#)」を参照してください。

Test Kitchen を使用して Vagrant でこのレシピを実行するには、まずクックブックを設定し Test Kitchen を初期化して設定する必要があります。以下にあげるのは Linux システム用ですが、Windows および Macintosh システムでも基本的に手順は同様です。ターミナルウィンドウを開くことから始めます。この章の例ではすべてコマンドラインツールを使用します。

クックブックを準備するには

1. ホームディレクトリで、`opsworks_cookbooks` という名前のサブディレクトリを作成します。この章でのクックブックはすべてここに置きます。次にこのクックブック用のサブディレクトリ `installpkg` を作成し、そのディレクトリに移動します。
2. `installpkg` で、以下のコードを含む `metadata.rb` というファイルを作成します。

```
name "installpkg"  
version "0.1.0"
```

この章の例ではわかりやすいようにクックブックの名前とバージョンのみを指定しますが、`metadata.rb`には幅広いクックブックのメタデータを含めることができます。詳細については、「[クックブックのメタデータについて](#)」を参照してください。

Note

必ず Test Kitchen を初期化する前に `metadata.rb` を作成してください。デフォルトの設定ファイルを作成するためにこのデータを使用します。

3. `installpkg` で `kitchen init` を実行し、Test Kitchen を初期化してデフォルトの Vagrant ドライバをインストールします。
4. `kitchen init` コマンドは、`installpkg` という名前YAMLの に設定ファイルを作成します `.kitchen.yml`。お好きなテキストエディタで ファイルを開きます。`.kitchen.yml` には、レシピを実行するシステムを指定する `platforms` セクションが含まれています。Test Kitchen はインスタンスを作成して、各プラットフォームで指定されたレシピを実行します。

Note

デフォルトでは、Test Kitchen は一度に 1 つのプラットフォームでレシピを実行します。インスタンスを作成するいずれかのコマンドに `-p` 引数を追加した場合、Test Kitchen はそれぞれのプラットフォームで平行してレシピを実行します。

この例では単一のプラットフォームが有効です。したがって `.kitchen.yml` を編集して `centos-6.4` プラットフォームを削除します。`.kitchen.yml` ファイルは以下のようになります。

```
---
driver:
  name: vagrant

provisioner:
  name: chef_solo

platforms:
  - name: ubuntu-12.04

suites:
  - name: default
    run_list:
      - recipe[installpkg::default]
    attributes:
```

Test Kitchen は `.kitchen.yml` の実行リストにあるレシピのみを実行します。レシピを識別するには、`[cookbook_name::recipe_name]`形式を使用します。`recipe_name` は `.rb` 拡張子を省略します。初期状態で、`.kitchen.yml` 実行リストにはクックブックのデフォルトのレシピである `installpkg::default` が含まれています。これはこれから実行するレシピです。実行リストを変更する必要はありません。

5. `installpkg` という名前の `recipes` のサブディレクトリを作成します。

クックブックにレシピが含まれる場合 (たいていは含まれますが)、`recipes` サブディレクトリに置かれている必要があります。

これで、クックブックにレシピを追加し、Test Kitchen を使用してインスタンスで実行できるようになりました。

レシピを実行するには

1. セクションの冒頭にある Git インストールコード例を含む `default.rb` という名前のファイルを作成し、サブディレクトリ `recipes` に保存します。
2. `installpkg` ディレクトリで `kitchen converge` を実行します。このコマンドは Vagrant で新しい Ubuntu インスタンスを起動し、クックブックをインスタンスにコピーし、`.kitchen.yml` の実行リストにあるレシピを実行するために Chef の実行を開始します。

3. レシピが成功したことを確認するには、`chef-client` を実行して `kitchen login` インスタンス SSH への接続を開きます。次に `git --version` を実行して、Git が正しくインストールされたことを確認します。ワークステーションに戻るには、`exit` を実行します。
4. 完了したら、`kitchen destroy` を実行してインスタンスをシャットダウンします。以下の例では、別のクックブックを使用します。

この例は手始めには適していますが、非常にシンプルです。インストールがより複雑なパッケージもあります。以下の一部またはすべてを実行しなければならない場合もあります。

- ユーザーを作成して設定します。
- データやログなどのために 1 つ以上のディレクトリを作成します。
- 1 つ以上の設定ファイルをインストールします。
- さまざまなオペレーティングシステムに対して異なるパッケージ名や属性値を指定します。
- サービスを開始し、必要に応じて再起動します。

以下の例でこれらの問題を処理する方法を説明するとともに、役に立つ操作をいくつかご紹介します。

例 2: ユーザー管理

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

シンプルなタスクとして他にインスタンスでのユーザー管理があります。以下のレシピは Linux インスタンスに新しいユーザーを追加します。

```
user "myuser" do
  home "/home/newuser"
  shell "/bin/bash"
end
```

[user](#) を使用して、Linux と Windows の両方のシステムでユーザーを管理しますが、一部の属性は一方のシステムにのみ適用されます。例では `myuser` という名前のユーザーを作成し、ホームディレクトリとシェルを指定します。指定されたアクションはないので、リソースはデフォルトの `create` アクションを使用します。user に属性を追加して、パスワードやグループ ID などのさまざまな設定を指定できます。ユーザー設定の変更やユーザーの削除などの関連するユーザー管理タスクにも、`user` を使用できます。詳細については、「[user](#)」を参照してください。

レシピを実行するには

1. `opsworks_cookbooks` 内に `newuser` という名前のディレクトリを作成し、そこに移動します。
2. 以下のコードを含む `metadata.rb` ファイルを作成し、それを `newuser` に保存します。

```
name "newuser"
version "0.1.0"
```

3. 「[例 1: パッケージのインストール](#)」の説明に従って Test Kitchen を初期化して設定し、`recipes` ディレクトリ内に `newuser` ディレクトリを追加します。
4. 例のレシピの `default.rb` をクックブックの `recipes` ディレクトリに追加します。
5. `kitchen converge` を実行してレシピを実行します。
6. `kitchen login` を使用してインスタンスにログインし、`cat /etc/passwd` を実行して新しいユーザーが存在することを確認します。`myuser` ユーザーはファイルの一番下に置く必要があります。

例 3: ディレクトリの作成

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

インスタンスにパッケージをインストールするとき、多くの場合は設定ファイルを作成して適切なディレクトリに配置する必要があります。しかし、ディレクトリがまだない場合があります。また、

データやログファイル用のディレクトリを作成する必要がある場合もあります。例えば、ほとんどの例で使用する Ubuntu システムを最初に起動しますが、`/srv` ディレクトリにはサブディレクトリがありません。アプリケーションサーバーをインストールする際に、`/srv/www/` ディレクトリや場合によってはデータファイルやログなどのためにサブディレクトリが必要になるかもしれません。以下のレシピはインスタンスで `/srv/www/` を作成します。

```
directory "/srv/www/" do
  mode 0755
  owner 'root'
  group 'root'
  action :create
end
```

[directory resource](#) を使用して、Linux と Windows の両方のシステムでディレクトリを作成して設定しますが、一部の属性の使用方法はシステムによって異なります。リソース名はリソースの `path` 属性のデフォルト値であるため、例では `/srv/www/` を作成し、`mode`、`owner`、および `group` プロパティを指定します。

レシピを実行するには

1. `opsworks_cookbooks`内に `createdir` という名前のディレクトリを作成し、そのディレクトリに移動します。
2. 「[例 1: パッケージのインストール](#)」の説明に従って Test Kitchen を初期化、設定し、`recipes` 内に `createdir` ディレクトリを追加します。
3. レシピコードの `default.rb` ファイルをクックブックの `recipes` サブディレクトリに追加します。
4. `kitchen converge` を実行してレシピを実行します。
5. `kitchen login` を実行して `/srv` に移動し、`www` サブディレクトリがあることを確認します。
6. `exit` を実行して、インスタンスを実行したままワークステーションに戻ります。

Note

インスタンスのホームディレクトリに関連するディレクトリを作成するには、ホームディレクトリを示すために `#{ENV['HOME']}` を使用します。たとえば、以下は `~/shared` ディレクトリを作成します。

```
directory "#{ENV['HOME']}/shared" do
  ...
end
```

`/srv/www/shared` のようにさらに多層のディレクトリを作成したい時があるかもしれません。前述のレシピを以下のように変更できます。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  action :create
end
```

レシピを実行するには

1. `default.rb` のコードを前述のレシピで置き換えます。
2. `kitchen converge` ディレクトリから `createdir` を実行します。
3. ディレクトリが実際に作成されたことを確認するには、`kitchen login` を実行して `/srv/www` に移動し、そこに `shared` サブディレクトリが含まれていることを確認します。
4. `kitchen destroy` を実行してインスタンスをシャットダウンします。

`kitchen converge` コマンドがはるかに高速に実行されたことがわかります。これはインスタンスがすでに実行中であり、インスタンスの起動や Chef のインストールなどが必要ないためです。Test Kitchen は更新されたクックブックをインスタンスにコピーし、Chef の実行を開始するだけでいいのです。

ここで再び `kitchen converge` を実行して、新しいインスタンスでレシピを実行します。以下のような結果になるはずですが、

```
Chef Client failed. 0 resources updated in 1.908125788 seconds
[2014-06-20T20:54:26+00:00] ERROR: directory[/srv/www/shared] (createdir::default line
 1) had an error: Chef::Exceptions::EnclosingDirectoryDoesNotExist: Parent directory /
srv/www does not exist, cannot create /srv/www/shared
```

```
[2014-06-20T20:54:26+00:00] FATAL: Chef::Exceptions::ChildConvergeError: Chef run process exited unsuccessfully (exit code 1)
>>>>> Converge failed on instance <default-ubuntu-1204>.
>>>>> Please see .kitchen/logs/default-ubuntu-1204.log for more details
>>>>> -----Exception-----
>>>>> Class: Kitchen::ActionFailed
>>>>> Message: SSH exited (1) for command: [sudo -E chef-solo --config /tmp/kitchen/solo.rb --json-attributes /tmp/kitchen/dna.json --log_level info]
>>>>> -----
```

何が起きたのか。問題は、デフォルトでは `directory` リソースは一度に 1 個のディレクトリしか作成できないことです。ディレクトリのチェーンは作成できません。前のレシピがうまくいった理由は、インスタンスで最初に実行したレシピですでに `/srv/www` が作成されていたため、`/srv/www/shared` の作成ではサブディレクトリが 1 個つだけ作成されたためです。

Note

`kitchen converge` を実行する場合は、レシピを実行するインスタンスが新規なのか既存のものなのかを確認してください。結果が異なる場合があります。

サブディレクトリのチェーンを作成するには、`recursive` 属性を `directory` に追加して、それを `true` に設定します。以下のレシピは、新しいインスタンスに `/srv/www/shared` を直接作成します。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

例 4: フロー制御の追加

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソ

リユーシオンに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

単なる Chef リソースの連なりであるレシピもあります。この場合、レシピを実行すると単純にリソースプロバイダーを 1 つずつ順番に実行します。ただし、より洗練された実行パスを持つ方が役に立つ場合が多くあります。以下に一般的なシナリオを 2 つ示します。

- 同じリソースを異なる属性設定で複数回実行するレシピがほしい。
- さまざまなオペレーティングシステムで異なる属性設定を使用したい。

レシピに Ruby の制御構造を組み込むことで、このようなシナリオに対処できます。このセクションでは、「[例 3: ディレクトリの作成](#)」のレシピを変更して両方のシナリオに対処する方法を説明します。

トピック

- [イテレーション](#)
- [条件付きロジック](#)

イテレーション

「[例 3: ディレクトリの作成](#)」では、`directory` リソースを使用してディレクトリまたはディレクトリチェーンを作成する方法を説明しました。ただし別々の 2 つのディレクトリ、`/srv/www/config` と `/srv/www/shared` を作成したい場合があるかもしれません。各ディレクトリに別々のディレクトリリソースを実装することもできますが、非常に多数のディレクトリを作成する場合はこの方法は面倒です。以下のレシピは、このタスクを処理するよりシンプルな方法を示します。

```
[ "/srv/www/config", "/srv/www/shared" ].each do |path|
  directory path do
    mode 0755
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
end
```

各サブディレクトリに対して別のディレクトリリソースを使用する代わりに、レシピではサブディレクトリのパスを含む文字列コレクションを使用します。Rubyの `each` メソッドはコレクションの各要素を最初のものから始めてそれぞれ 1 回ずつ実行します。要素の値はリソース内で `path` 変数で表されます。この場合はディレクトリパスを表します。この例を簡単に応用して、サブディレクトリをいくつでも作成できます。

レシピを実行するには

1. `createdir` ディレクトリにとどまります。以下のいくつかの例で、このクックブックを使用します。
2. `kitchen destroy` をまだ実行していない場合は実行して、新しいインスタンスで始められるようにします。
3. `default.rb` のコードを例で置き換えて、`kitchen converge` を実行します。
4. インスタンスにログインします。`/srv` の下に新しく作成されたディレクトリがあります。

ハッシュテーブルを使用して各イテレーションに 2 個の値を指定できます。以下のレシピは `/srv/www/config` および `/srv/www/shared` を、別モードでそれぞれ作成します。

```
{ "/srv/www/config" => 0644, "/srv/www/shared" => 0755 }.each do |path, mode_value|
  directory path do
    mode mode_value
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
end
```

レシピを実行するには

1. `kitchen destroy` をまだ実行していない場合は実行して、新しいインスタンスで始められるようにします。
2. `default.rb` のコードを例で置き換えて、`kitchen converge` を実行します。
3. インスタンスにログインします。`/srv` の下に指定したモードで新しく作成されたディレクトリがあります。

Note

AWS OpsWorks スタックレシピは通常、このアプローチを使用して[スタック設定とデプロイ JSON](#)から値を抽出します。これは基本的に大きなハッシュテーブルです。そして、リソースに挿入します。例については、[Deploy レシピ](#)を参照してください。

条件付きロジック

複数の実行ブランチを作成するために Ruby の条件付きロジックを使用することもできます。以下のレシピでは、if-elsif-else ロジックを使用して 1 つ前の例を拡張しています。/srv/www/shared という名前のサブディレクトリを作成しますが、Debian および Ubuntu システムに対してのみです。その他のすべてのシステムの場合は、Test Kitchen の出力に表示するエラーメッセージを記録します。

```
if platform?("debian", "ubuntu")
  directory "/srv/www/shared" do
    mode 0755
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
else
  log "Unsupported system"
end
```

レシピの例を実行するには

1. インスタンスが実行中の場合は、`kitchen destroy` を実行してシャットダウンします。
2. `default.rb` のコードを例のコードで置き換えます。
3. `.kitchen.yml` を編集してプラットフォームの一覧に CentOS 6.4 システムを追加します。ファイルの `platforms` セクションはこのようになります。

```
...
platforms:
  - name: ubuntu-12.04
  - name: centos-6.4
```

...

4. `kitchen converge` を実行すると、インスタンスを作成し `.kitchen.yml` のプラットフォームごとのレシピを順次実行します。

Note

1 個のインスタンスのみをコンバースする場合は、インスタンス名をパラメータとして追加します。例えば、Ubuntu プラットフォームのレシピのみをコンバースするには、`kitchen converge default-ubuntu-1204` を実行します。プラットフォーム名を忘れた場合は、`kitchen list` を実行します。

Test Kitchen の出力の CentOS 部分にログメッセージがあります。以下のようなものになります。

```
...
Converging 1 resources
Recipe: createdir::default
* log[Unsupported system] action write[2014-06-23T19:10:30+00:00] INFO: Processing
  log[Unsupported system] action write (createdir::default line 12)
[2014-06-23T19:10:30+00:00] INFO: Unsupported system

[2014-06-23T19:10:30+00:00] INFO: Chef Run complete in 0.004972162 seconds
```

これでインスタンスにログインし、ディレクトリが作成されている (または作成されていない) ことを確認できます。ただし、単純に `kitchen login` を実行することはできません。たとえば `kitchen login default-ubuntu-1204` のようにプラットフォーム名を追加してインスタンスを指定する必要があります。

Note

Test Kitchen コマンドがインスタンス名をとる場合、完全な名前を入力する必要はありません。Test Kitchen は Ruby の正規表現としてインスタンス名を処理しますので、ユニークな一致ができる文字数があれば十分です。例えば `kitchen converge ub` を実行して Ubuntu のインスタンスのみをコンバースすることもできますし、`kitchen login 64` を実行して CentOS のインスタンスにログインすることもできます。

ここまでで、レシピがどのように実行中のプラットフォームを見分けているか疑問に思うことでしょう。Chef は実行のたびに [Ohai](#) というツールを実行してプラットフォームを含むシステムデータを収集し、ノードオブジェクトという構造の中で一連の属性として示します。Chef `platform?` メソッドはカッコ内のシステムを Ohai のプラットフォームの値と比較して、一致すれば `true` を返します。

`node['attribute_name']` を使用してコード内で直接ノード属性の値を参照できます。プラットフォームの値は、例えば、`node['platform']` で表されます。たとえば、前述の例を以下のように記述することもできます。

```
if node[:platform] == 'debian' or node[:platform] == 'ubuntu'
  directory "/srv/www/shared" do
    mode 0755
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
else
  log "Unsupported system"
end
```

レシピに条件付きロジックを含める一般的な理由は、異なる Linux ファミリーが時としてパッケージやディレクトリに異なる名前を使用していることに対応するためです。たとえば Apache のパッケージ名は CentOS システムでは `httpd` であり、Ubuntu システムでは `apache2` です。

異なるシステム用に異なる文字列が必要な場合、Chef [value_for_platform](#) メソッドは、`if-elsif-else` よりも簡単なソリューションです。以下のレシピは CentOS システムでは `/srv/www/shared` ディレクトリ、Ubuntu システムでは `/srv/www/data` ディレクトリ、その他の場合は `/srv/www/config` を作成します。

```
data_dir = value_for_platform(
  "centos" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
)
directory data_dir do
  mode 0755
  owner 'root'
```

```
group 'root'
  recursive true
  action :create
end
```

`value_for_platform` は `data_dir` に適切なパスを割り当て、`directory` リソースはその値を使用してディレクトリを作成します。

レシピの例を実行するには

1. インスタンスが実行中の場合は、`kitchen destroy` を実行してシャットダウンします。
2. `default.rb` のコードを例のコードで置き換えます。
3. `kitchen converge` を実行し、その後各インスタンスにログインして適切なディレクトリがあることを確認します。

例 5: 属性の使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

前述のセクションのレシピでは、プラットフォーム以外のすべてにハードコーディングされた値を使用しました。この方法は、たとえば複数のレシピで同じ値を使用したい場合は不便なことがあります。クックブックに属性ファイルを含めることにより、値をレシピとは別に定義できます。

属性ファイルは、1 つ以上の属性に値を割り当てる Ruby アプリケーションです。クックブックの `attributes` フォルダにある必要があります。Chef はノードオブジェクトに属性を組み込みます。属性を参照することでどのレシピでも属性値を使用できます。このトピックでは、「[イテレーション](#)」のレシピを変更して属性を使用する方法を説明します。参考までに、元のレシピを以下に示します。

```
[ "/srv/www/config", "/srv/www/shared" ].each do |path|
  directory path do
```

```
mode 0755
owner 'root'
group 'root'
recursive true
action :create
end
end
```

以下はサブディレクトリ名、モード、所有者およびグループの値の属性を定義します。

```
default['createdir']['shared_dir'] = 'shared'
default['createdir']['config_dir'] = 'config'
default['createdir']['mode'] = 0755
default['createdir']['owner'] = 'root'
default['createdir']['group'] = 'root'
```

次の点に注意してください:

- 各定義は属性タイプから始まります。

属性が複数回定義されている場合 (おそらく異なる属性ファイルで)、属性タイプは属性の優先順位を指定し、ノードオブジェクトに組み込まれるかが決定されます。詳細については、「[属性の優先順位](#)」を参照してください。この例の定義はすべて、この目的のための通常のタイプである default 属性タイプです。

- 属性はネストされた名前を持ちます。

ノードオブジェクトは基本的に任意に深くネストされたハッシュテーブルであるため、属性名もネストされたものになることがよくあります。この属性ファイルは、クックブック名でネストされた名前を使用する場合の標準的な手法に従い、createdir が最初の要素になります。

属性の最初の要素として createdir を使用する理由は、Chef を実行する時に Chef がクックブックからノードオブジェクトに属性を取り入れるためです。AWS OpsWorks スタックでは、ノードオブジェクトには、定義した属性に加えて、[組み込みクックブック](#)の多数の属性が含まれます。クックブック名を属性名に含めることで、特に属性が port や user のような名前である場合、別のクックブックからの属性名と競合する危険性を大幅に減少します。その属性値を上書きする場合を除き、属性に "[\[:apache2\]\[:user\]](#)" のような名前をつけないでください。詳細については、「[カスタムクックブック属性の使用](#)」を参照してください。

以下の例で、ハードコードされた値の代わりに属性を使用した元のレシピを示します。

```
[ "/srv/www/#{node['createdir']['shared_dir']}", "/srv/www/#{node['createdir']
['config_dir']}" ].each do |path|
  directory path do
    mode node['createdir']['mode']
    owner node['createdir']['owner']
    group node['createdir']['group']
    recursive true
    action :create
  end
end
end
```

Note

属性値を文字列に組み込みたい場合は、#{ } でパッケージ化します。前述の例では、#{node['createdir']['shared_dir']} は "/srv/www/" に "shared" を追加しています。

レシピを実行するには

1. `kitchen destroy` を実行して新しいインスタンスで始められるようにします。
2. `recipes/default.rb` のコードを前述のレシピ例で置き換えます。
3. `createdir` に `attributes` という名前のサブディレクトリを作成して、`default.rb` という名前の属性定義を含むファイルを追加します。
4. `.kitchen.yml` を編集してプラットフォームの一覧から CentOS を削除します。
5. `kitchen converge` を実行し、その後インスタンスにログインして `/srv/www/shared` と `/srv/www/config` がそこにあることを確認します。

Note

AWS OpsWorks スタックでは、値を属性として定義すると、さらに利点があります。[カスタム JSON](#) を使用して、スタックごと、またはデプロイごとにこれらの値を上書きできます。以下を含むさまざまな目的に便利です。

- クックブックを変更せずに、設定やユーザー名などのレシピの動作をカスタマイズできます。

たとえば、異なるスタックに同じクックブックを使用し、`カスタム` を使用して特定のスタックのキー設定JSONを指定できます。これにより、クックブックを変更したりそれぞれのスタックに異なるクックブックを使用する手間を省くことができます。

- クックブックリポジトリにデータベースのパスワードなどの機密と考えられる情報を置く必要はありません。

代わりに、属性を使用してデフォルト値を定義し、`カスタム` を使用してその値を実際の値でJSON上書きできます。

`カスタム` を使用して属性JSONを上書きする方法の詳細については、「」を参照してください。[属性の上書き](#)。

属性ファイルは Ruby アプリケーションであるため、もっともシンプルなものは `default.rb` という名前になります。これにより、たとえば条件付きロジックを使用してオペレーティングシステムに基づいて属性値を指定できます。「[条件付きロジック](#)」で、レシピで異なる Linux ファミリーに対して異なるサブディレクトリ名を指定しました。属性ファイルを使用すると、属性ファイルに条件付きロジックを配置できます。

以下の属性ファイルは、`value_for_platform` を使用してオペレーティングシステムによって異なる `['shared_dir']` 属性値を指定します。その他の条件では、Ruby の `if-elsif-else` ロジックまたは `case` ステートメントを使用できます。

```
data_dir = value_for_platform(
  "centos" => { "default" => "shared" },
  "ubuntu" => { "default" => "data" },
  "default" => "user_data"
)
default['createdir']['shared_dir'] = data_dir
default['createdir']['config_dir'] = "config"
default['createdir']['mode'] = 0755
default['createdir']['owner'] = 'root'
default['createdir']['group'] = 'root'
```

レシピを実行するには

- `kitchen destroy` を実行して新しいインスタンスで開始できるようにします。

2. `attributes/default.rb` のコードを前述の例で置き換えます。
3. `.kitchen.yml` を編集して、「[条件付きロジック](#)」の説明に従い CentOS プラットフォームをプラットフォームセクションに追加します。
4. `kitchen converge` を実行し、その後インスタンスにログインしてディレクトリがそこにあることを確認します。

完了したら、`kitchen destroy` を実行してインスタンスを終了します。以下の例では、新しいクックブックを使用します。

例 6: ファイルの作成

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

ディレクトリを作成したら、多くの場合設定ファイルやデータファイルなどを配置する必要があります。このトピックでは、インスタンスにファイルをインストールする 2 つの方法を説明します。

トピック

- [クックブックからのファイルのインストール](#)
- [テンプレートからのファイルの作成](#)

クックブックからのファイルのインストール

インスタンスにファイルをインストールする最も簡単な方法は、[cookbook_file](#) リソースを使用してクックブックから、Linux と Windows の両方のインスタンスの指定した場所にファイルをコピーすることです。この例では、「[例 3: ディレクトリの作成](#)」のレシピを拡張して、ディレクトリの作成後に `/srv/www/shared` にファイルを追加します。参考までに、元のレシピを以下に示します。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

クックブックをセットアップするには

1. `opsworks_cookbooks` ディレクトリ内に `createfile` という名前のディレクトリを作成し、そのディレクトリに移動します。
2. `createfile` に、次のコンテンツを含む `metadata.rb` ファイルを追加します。

```
name "createfile"
version "0.1.0"
```

3. 「[例 1: パッケージのインストール](#)」の説明に従って Test Kitchen を初期化し設定して、`platforms` リストから CentOS を削除します。
4. `createfile` に `recipes` サブディレクトリを追加します。

インストールするファイルには、次のJSONデータが含まれています。

```
{
  "my_name" : "myname",
  "your_name" : "yourname",
  "a_number" : 42,
  "a_boolean" : true
}
```

データファイルをセットアップするには

1. `files` サブディレクトリを `createfile` に、`default` サブディレクトリを `files` に追加します。cookbook_file と一緒にインストールするファイルは `files` のサブディレクトリに置く必要があります。例えば、`files/default` などです。

Note

異なるシステムに異なるファイルを指定する場合は、システム固有のファイルを `files/ubuntu` のようにシステム用に名前をつけたサブフォルダに置きます。cookbook_file リソースは、適切なシステム固有のファイルが存在する場合はそれをコピーし、ない場合は default ファイルを使用します。詳細については、「[cookbook_file](#)」を参照してください。

2. 前の例JSONの `example_data.json` を使用して という名前のファイルを作成し、 に追加します `files/default`。

以下のレシピは `example_data.json` を指定した場所にコピーします。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end

cookbook_file "/srv/www/shared/example_data.json" do
  source "example_data.json"
  mode 0644
  action :create_if_missing
end
```

ディレクトリリソースが `/srv/www/shared` を作成した後、cookbook_file リソースは `example_data.json` をそのディレクトリにコピーし、またファイルのユーザー、グループ、モードを設定します。

Note

cookbook_file リソースでは `create_if_missing` という新しいアクションが導入されています。create アクションも使用できますが、これは既存のファイルを上書きします。何も上書きしたくない場合は、`create_if_missing` を使用すると、`example_data.json` が存在しない場合のみこれをインストールします。

レシピを実行するには

1. `kitchen destroy` を実行して新しいインスタンスで開始できるようにします。
2. 前述のレシピを含む `default.rb` ファイルを作成し、`recipes` に保存します。
3. `kitchen converge` を実行し、その後インスタンスにログインして `/srv/www/shared` に `example_data.json` があることを確認します。

テンプレートからのファイルの作成

`cookbook_file` リソースはさまざまな目的に役立ちますが、クックブックにあるファイルしかインストールしません。[template](#) リソースを使えば、テンプレートから動的にファイルを作成することで、より柔軟にファイルを Windows または Linux インスタンスにインストールできます。その後で実行時にファイルの内容の詳細を検討し、必要に応じて変更できます。たとえば、インスタンスを開始する際に特定の設定をするために必要な設定ファイルがあり、あとでスタックにインスタンスを追加する際にその設定を変更する必要があるかもしれません。

この例では `createfile` クックブックを変更し、`template` リソースを使用して `example_data.json` をほんの少し変更したバージョンをインストールします。

インストールされたファイルはこのようになります。

```
{
  "my_name" : "myname",
  "your_name" : "yourname",
  "a_number" : 42,
  "a_boolean" : true,
  "a_string" : "some string",
  "platform" : "ubuntu"
}
```

通常、テンプレートリソースは属性ファイルとともに使用します。例では以下の値を定義するために1つ使用しています。

```
default['createfile']['my_name'] = 'myname'
default['createfile']['your_name'] = 'yourname'
default['createfile']['install_file'] = true
```

クックブックをセットアップするには

1. createfile クックブックの files ディレクトリおよびそのコンテンツを削除します。
2. attributes に createfile サブディレクトリを追加し、default.rb ファイルを前述の属性定義を含む attributes に追加します。

テンプレートは .erb ファイルであり、基本的に内容の一部がプレースホルダーで表される最終ファイルのコピーです。template リソースがファイルを作成する場合、テンプレートの内容を指定されたファイルにコピーし、プレースホルダーを割り当てられた値で上書きします。example_data.json のテンプレートを次に示します。

```
{
  "my_name" : "<%= node['createfile']['my_name'] %>",
  "your_name" : "<%= node['createfile']['your_name'] %>",
  "a_number" : 42,
  "a_boolean" : <%= @a_boolean_var %>,
  "a_string" : "<%= @a_string_var %>",
  "platform" : "<%= node['platform'] %>"
}
```

<%=...%> の値はプレースホルダーです。

- <%=node[...]%> はノード属性値を示します。

この例では、"your_name" の値がクックブックの属性ファイルからの属性値の 1 つを表すプレースホルダーです。

- <%=@...%> はすでに説明したようにテンプレートリソースで定義されている変数の値を示します。

テンプレートファイルを作成するには

1. templates サブディレクトリを createfile クックブックに、default サブディレクトリを templates に追加します。

Note

templates ディレクトリは files ディレクトリと似たような機能を果たします。システム固有のテンプレートを ubuntu などシステム名をつけたサブディレクトリに配置で

きます。template リソースは適切なシステム固有テンプレートがある場合はそれを使用し、ない場合は default テンプレートを使用します。

2. `example_data.json.erb` ディレクトリに `templates/default` という名前のファイルを作成します。テンプレート名は任意ですが、通常は、拡張子を含むファイル名に `.erb` を付けて作成します。

以下のレシピでは、template リソースを使用して `/srv/www/shared/example_data.json` を作成します。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end

template "/srv/www/shared/example_data.json" do
  source "example_data.json.erb"
  mode 0644
  variables(
    :a_boolean_var => true,
    :a_string_var => "some string"
  )
  only_if {node['createfile']['install_file']}
end
```

template リソースはテンプレートから `example_data.json` を作成して `/srv/www/shared` にインストールします。

- テンプレートの名前 `/srv/www/shared/example_data.json` は、インストールされているファイルのパスと名前を指定します。
- `source` 属性は、ファイルの作成に使用したテンプレートを指定します。
- `mode` 属性は、インストールされているファイルのモードを指定します。
- リソースは、2 つの変数 `a_boolean_var` と `a_string_var` を定義します。

リソースが `example_data.json` を作成する際、テンプレートのプレースホルダーをリソースの対応する値で上書きします。

- `only_if` ガード属性は `['createfile']['install_file']` が `true` に設定されている時のみファイルを作成するようにリソースに指示します。

レシピを実行するには

1. `kitchen destroy` を実行して新しいインスタンスで開始できるようにします。
2. `recipes/default.rb` のコードを前述の例で置き換えます。
3. `kitchen converge` を実行し、その後インスタンスにログインしてファイルが `/srv/www/shared` に存在し内容が正しいことを確認します。

完了したら、`kitchen destroy` を実行してインスタンスをシャットダウンします。以下のセクションでは新しいクックブックを使用します。

例 7: コマンドとスクリプトの実行

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Chef リソースはインスタンスのさまざまなタスクを処理できますが、シェルコマンドまたはスクリプトを使用した方がいい場合もあります。たとえば、特定のタスクを実行するスクリプトをすでに使用していて、新しいコードを実装するよりもそれを使い続ける方が簡単な場合もあります。このセクションでは、インスタンスでコマンドまたはスクリプトを実行する方法を説明します。

トピック

- [コマンドの実行](#)
- [スクリプトの実行](#)

コマンドの実行

[script](#) リソースは、1 つ以上のコマンドを実行します。また、`cs`、`bash`、`Perl`、`Python`、`Ruby` のコマンドインタプリタがサポートされているため、適切なインタプリタがインストールされていれば、Linux または Windows のいずれかのシステムで使用できます。このトピックでは、Linux インスタンスで単純な Bash コマンドを実行する方法について説明します。Chef では、[powershell_script](#) と [batch](#) もサポートされており、Windows でのスクリプトの実行に使用できます。詳細については、「[Windows PowerShell スクリプトの実行](#)」を参照してください。

開始するには、以下の手順を実行します。

1. `opsworks_cookbooks` ディレクトリ内に `script` という名前のディレクトリを作成し、そのディレクトリに移動します。
2. `script` に、次のコンテンツを含む `metadata.rb` ファイルを追加します。

```
name "script"
version "0.1.0"
```

3. 「[例 1: パッケージのインストール](#)」の説明に従って Test Kitchen を初期化し設定して、`platforms` リストから CentOS を削除します。
4. `script` 内に、`recipes` という名前のディレクトリを作成します。

`script` リソース自体を使用してコマンドを実行することもできますが、Chef はまた、インタプリタ名をつけた一連のリソースのコマンドインタプリタ固有バージョンをサポートします。以下のレシピは [bash](#) リソースを使用して、シンプルな Bash スクリプトを実行します。

```
bash "install_something" do
  user "root"
  cwd "/tmp"
  code <<-EOH
    touch somefile
  EOH
  not_if do
    File.exists?("/tmp/somefile")
  end
end
```

`bash` リソースは以下のように設定されます。

- デフォルトのアクション `run` を使用して `code` ブロックでコマンドを実行します。

この例では 1 個のコマンド `touch somefile` がありますが、`code` ブロックには複数のコマンドを含めることができます。

- `user` 属性は、コマンドを実行するユーザーを指定します。
- `cwd` 属性は、作業ディレクトリを指定します。

この例では、`touch` は `/tmp` ディレクトリにファイルを作成します。

- `not_if` ガード属性はファイルがすでに存在する場合はリソースにアクションを取らないよう指示します。

レシピを実行するには

1. 前述の例のコードを含む `default.rb` ファイルを作成し、`recipes` に保存します。
2. `kitchen converge` を実行し、その後インスタンスにログインして `/tmp` にファイルがあることを確認します。

スクリプトの実行

`script` リソースは、特に 1 つか 2 つのコマンドのみを実行する必要がある場合に便利ですが、ファイルにスクリプトを保存してそのファイルを実行する方が多い場合も多くあります。[execute](#) リソースによって、スクリプトファイルなど指定した実行可能ファイルが Linux または Windows で実行されます。このトピックでは、前述の例から `script` クックブックを変更し、`execute` を使用して単純なシェルスクリプトを実行します。簡単に例をより複雑なスクリプトや別のタイプの実行ファイルに拡張できます。

スクリプトファイルを設定するには

1. `files` サブディレクトリを `script` に、`default` サブディレクトリを `files` に追加します。
2. 以下を含む `touchfile` という名前のファイルを作成し、そのファイルを `files/default` へ追加します。この例では、一般的な Bash インタプリタラインが使用されています。必要に応じてシェル環境に対応するインタプリタを代入してください。

```
#!/usr/bin/env bash
touch somefile
```

スクリプトファイルには、任意の数のコマンドを含めることができます。わかりやすいように、このスクリプトの例には 1 つの touch コマンドのみを使用しています。

以下のレシピはスクリプトを実行します。

```
cookbook_file "/tmp/touchfile" do
  source "touchfile"
  mode 0755
end

execute "touchfile" do
  user "root"
  cwd "/tmp"
  command "./touchfile"
end
```

cookbook_file リソースは /tmp にスクリプトファイルをコピーし、ファイルを実行可能にするモードを設定します。execute リソースはその後以下のようにファイルを実行します。

- user 属性はコマンドのユーザーを指定します (この例では root)。
- cwd 属性は作業ディレクトリ (この例では /tmp) を指定します。
- command 属性は作業ディレクトリにある実行すべきスクリプト (この例では touchfile) を指定します。

レシピを実行するには

1. recipes/default.rb のコードを前述の例で置き換えます。
2. kitchen converge を実行し、その後インスタンスにログインして、/tmp にスクリプトファイルが含まれており、モードが 0755 に設定されていて、かつ somefile があることを確認します。

完了したら、kitchen destroy を実行してインスタンスをシャットダウンします。以下のセクションでは新しいクックブックを使用します。

例 8: 管理サービス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

通常、アプリケーションサーバーなどのパッケージには、開始、停止、再起動などを行う必要がある関連サービスがあります。たとえばパッケージをインストールしたりインスタンスの起動が完了した後は Tomcat サービスを開始する必要があり、また設定ファイルを変更するたびにこのサービスを再起動する必要があります。このトピックでは、例として Tomcat アプリケーションサーバーを使用し、Linux インスタンスでのサービスの管理方法の基礎を説明します。サービスリソースは、詳細に多少の違いはあるものの、多くの Windows インスタンスで同じように動作します。詳細については、「[service](#)」を参照してください。

Note

例では、Tomcat のインストールを必要最小限にして service リソースの使用方法の基礎を提示します。より機能的な Tomcat サーバーのレシピを実行する方法の例は、「[カスタム Tomcat サーバーレイヤーの作成](#)」を参照してください。

トピック

- [サービスの定義と起動](#)
- [通知を使用したサービスの開始または再起動](#)

サービスの定義と起動

このセクションでは、サービスの定義と起動方法を説明します。

開始するには、以下の手順を実行します。

1. `opsworks_cookbooks` ディレクトリ内に `tomcat` という名前のディレクトリを作成し、そのディレクトリに移動します。

2. tomcat に、次のコンテンツを含む metadata.rb ファイルを追加します。

```
name "tomcat"  
version "0.1.0"
```

3. 「[例 1: パッケージのインストール](#)」の説明に従って Test Kitchen を初期化し設定して、platforms リストから CentOS を削除します。
4. tomcat に recipes サブディレクトリを追加します。

[service](#) リソースを使ってサービスを管理します。以下のデフォルトのレシピは Tomcat をインストールしサービスを起動します。

```
execute "install_updates" do  
  command "apt-get update"  
end  
  
package "tomcat7" do  
  action :install  
end  
  
include_recipe 'tomcat::service'  
  
service 'tomcat' do  
  action :start  
end
```

このレシピでは、以下のような処理を実行します。

- execute リソースは apt-get update を実行して現在のシステム更新をインストールします。
この例で使用されている Ubuntu のインスタンスでは、Tomcat をインストールする前に更新をインストールする必要があります。その他のシステムでは異なる要件がある場合があります。
- package リソースは Tomcat 7 をインストールします。
- 中に含まれる tomcat::service レシピはサービスを定義します。説明は後述します。
- service リソースは Tomcat サービスを開始します。

このリソースでサービスの停止や再起動などの他のコマンドを発行することもできます。

以下の例で `tomcat::service` レシピを説明します。

```
service 'tomcat' do
  service_name "tomcat7"
  supports :restart => true, :reload => false, :status => true
  action :nothing
end
```

レシピは、Tomcat サービスの定義を以下のように作成します。

- リソース名 `tomcat` は他のレシピがサービスを参照するために使用されます。

例えば、`default.rb` は `tomcat` を参照してサービスを開始します。

- `service_name` リソースは サービス名を指定します。

インスタンスでサービスをリストする場合は、Tomcat サービスは `tomcat7` という名前になります。

- `supports` は Chef がサービスの `restart`、`reload`、および `status` コマンドを管理する方法を指定します。
 - `true` は、Chef が `init` またはその他のサービスプロバイダーを使用してコマンドを実行できることを示します。
 - `false` は、Chef がその他の方法でコマンドの実行を試みる必要があることを示します。

`action` が `:nothing` に設定され、リソースにアクションを取らないよう指示していることに注意してください。サービスリソースはもちろん `start` や `restart` などのアクションをサポートしています。ただし、このクックブックではサービス定義を使用する標準的な手法をとっており、どこにあるサービスであってもアクションをとらずに開始または再起動を行います。サービスを開始または再起動するレシピはまずそれを定義します。したがって最もシンプルな方法は、サービス定義を別のレシピに配置して必要に応じてそれを他のレシピにも含めることです。

Note

わかりやすいように、この例のデフォルトのレシピでは `service` リソースを使用してサービス定義を実行した後にサービスを開始します。本番の実装では通常、後述のように `notifies` を使用してサービスを開始または再開します。

レシピを実行するには

1. デフォルトのレシピ例を含む `default.rb` ファイルを作成して、`recipes` に保存します。
2. サービス定義の例を含む `service.rb` ファイルを作成し、`recipes` に保存します。
3. `kitchen converge` を実行し、その後インスタンスにログインします。以下のコマンドを実行し、サービスが実行中であることを確認します。

```
sudo service tomcat7 status
```

Note

`service.rb` を `default.rb` とは別に実行した場合、`.kitchen.yml` を編集して実行リストに `tomcat::service` を追加する必要がある場合があります。ただしレシピを含む場合は、コードはレシピが実行される前に親レシピに組み込まれます。したがって `service.rb` は基本的に `default.rb` の一部であり、あらためて実行リストに追加する必要ではありません。

通知を使用したサービスの開始または再起動

本番の実装では、サービスの開始や再起動に通常は `service` は使用しません。代わりに、`notifies` を任意のリソースに追加します。例えば、設定ファイルを変更した後サービスを再起動する場合、`notifies` を関連する `template` リソースに含めます。`notifies` を使用すると、`service` リソースを使用した明示的なサービスの再起動に対して以下の利点があります。

- `notifies` 要素は、関連する設定ファイルが変更された場合にのみサービスを再起動するので、必要のない時にサービスが再起動されるリスクを避けられます。
- Chef は、実行されるサービスに含まれる `notifies` の数に関係なく、各実行の最後に必要に応じて 1 度だけサービスを再起動します。

たとえば、Chef の実行に含まれる複数のテンプレートリソースがそれぞれ異なる設定ファイルを変更し、ファイルが変更された場合にはサービスの再起動を要求する可能性があります。しかし、サービスの再起動は Chef の実行の最後に 1 度だけにしたいものでしょう。そうしないと、先の再起動からまだ完全に機能が立ち上がらないうちにまたサービスを再起動することになり、エラーが発生する可能性があります。

この例では `tomcat::default` を変更して、`template` を使用してサービスを再起動する `notifies` リソースを含めます。実際の例では、テンプレートリソースを使用して Tomcat 設定ファイルの 1 つをカスタマイズしたバージョンを作成しますが、それは非常に長く複雑なものになります。わかりやすいように、例では「[テンプレートからのファイルの作成](#)」のテンプレートリソースを使用します。Tomcat とは関係ありませんが、`notifies` の使用方法をわかりやすく説明できます。テンプレートを使用して Tomcat の設定ファイルを作成する方法の例は、「[Setup レシピ](#)」を参照してください。

クックブックをセットアップするには

1. `templates` サブディレクトリを `tomcat` に、`default` サブディレクトリを `templates` に追加します。
2. `example_data.json.erb` クックブックから `createfile` テンプレートを `templates/default` ディレクトリへコピーします。
3. `attributes` に `tomcat` サブディレクトリを追加します。
4. `default.rb` クックブックから `createfile` 属性ファイルを `attributes` ディレクトリへコピーします。

以下のレシピは `notifies` を使用して Tomcat サービスを再起動します。

```
execute "install_updates" do
  command "apt-get update"
end

package "tomcat7" do
  action :install
end

include_recipe 'tomcat::service'

service 'tomcat' do
  action :enable
end

directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
end
```



```
  action :create
end

template "/srv/www/shared/example_data.json" do
  source "example_data.json.erb"
  mode 0644
  variables(
    :a_boolean_var => true,
    :a_string_var => "some string"
  )
  only_if {node['createfile']['install_file']}
  notifies :restart, resources(:service => 'tomcat')
end
```

この例は、「[テンプレートからのファイルの作成](#)」のレシピを前述のセクションのレシピにマージします。2つの大幅な変更があります。

- `service` リソースは依然として同じ場所にありますが、別の用途で使用されます。
`:enable` アクションで、起動時に Tomcat サービスが有効になります。
- これでテンプレートリソースに `notifies` が含まれ、`example_data.json` が変更された場合には Tomcat サービスを再起動します。

これで、Tomcat が最初にインストールされた時および設定が変更されるたびに再起動した時に、確実にサービスが開始されます。

レシピを実行するには

1. `kitchen destroy` を実行して新しいインスタンスで始められるようにします。
2. `default.rb` のコードを前述の例で置き換えます。
3. `kitchen converge` を実行し、その後インスタンスにログインしてサービスが実行中であることを確認します。

Note

サービスを再起動したいがレシピに `template` をサポートする `notifies` などのリソースが含まれていない場合、代わりにダミーの `execute` リソースを使用できます。例

```
execute 'trigger tomcat service restart' do
  command 'bin/true'
  notifies :restart, resources(:service => 'tomcat')
end
```

execute リソースには、command を実行する手段としてのみリソースを使用している場合でも、notifies 属性が必要です。この例では、/bin/true を実行することでこの要件を回避しています。これは単純に成功コードを返すだけのシェルコマンドです。

例 9: Amazon EC2 インスタンスの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

この時点で、でインスタンスをローカルで実行しています VirtualBox。これは迅速かつ簡単ですが、最終的には Amazon EC2 インスタンスでレシピをテストする必要があります。特に、Amazon Linux でレシピを実行する場合は、Amazon でのみ使用できます EC2。CentOS などの同様のシステムを予備的な実装やテストに使用できますが、Amazon Linux でレシピを完全にテストする唯一の方法は Amazon EC2 インスタンスを使用することです。

このトピックでは、Amazon EC2 インスタンスでレシピを実行する方法について説明します。前述のセクションとほとんど同じ方法で Test Kitchen や Vagrant を使用しますが、2 つ違いがあります。

- ドライバは Vagrant ではなく [kitchen-ec2](#) です。
- クックブックの .kitchen.yml ファイルは、Amazon EC2 インスタンスを起動するために必要な情報で設定する必要があります。

Note

代替手段として、`vagrant-aws` Vagrant プラグインを使用する方法があります。詳細については、[「Vagrant AWS Provider」](#)を参照してください。

Amazon EC2インスタンスを作成するにはAWS認証情報が必要です。AWS アカウントがない場合は、次のように取得できます。

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. <https://portal.aws.amazon.com/billing/サインアップ>を開きます。
2. オンラインの手順に従います。

サインアップ手順の一環として、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

にサインアップすると AWS アカウント、AWS アカウントのルートユーザー が作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用して[ルートユーザーアクセスが必要なタスク](#)を実行してください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。 / に移動し、マイアカウントを選択すると、いつでも現在のアカウントアクティビティを表示<https://aws.amazon.com>し、アカウントを管理できます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、日常的なタスクにルートユーザーを使用しないように AWS アカウントのルートユーザー、 を保護し AWS IAM Identity Center、 を有効にして、管理ユーザーを作成します。

を保護する AWS アカウントのルートユーザー

1. ルートユーザーを選択し、AWS アカウント E メールアドレスを入力して、アカウント所有者[AWS Management Console](#)としてサインインします。次のページでパスワードを入力します。

ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの[ルートユーザーとしてサインインする](#)を参照してください。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

手順については、[「ユーザーガイド」の AWS アカウント「ルートユーザーの仮想MFAデバイスを有効にする \(コンソール\)」](#)を参照してください。IAM

管理アクセスを持つユーザーを作成する

1. IAM Identity Center を有効にします。

手順については、「AWS IAM Identity Center ユーザーガイド」の[「AWS IAM Identity Centerの有効化」](#)を参照してください。

2. IAM Identity Center で、ユーザーに管理アクセス権を付与します。

を ID ソース IAM アイデンティティセンターディレクトリとして使用する方法的チュートリアルについては、「AWS IAM Identity Center ユーザーガイド」の[「デフォルトを使用してユーザーアクセスを設定する IAM アイデンティティセンターディレクトリ」](#)を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM Identity Center ユーザーでサインインするには、IAM Identity Center ユーザーの作成時に E メールアドレスに URL 送信されたサインインを使用します。

IAM Identity Center ユーザーを使用してサインインする方法については、「AWS サインイン ユーザーガイド」の[AWS「アクセスポータルにサインインする」](#)を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM Identity Center で、最小特権のアクセス許可を適用するベストプラクティスに従うアクセス許可セットを作成します。

手順については、「AWS IAM Identity Center ユーザーガイド」の「[権限設定を作成する](#)」を参照してください。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

手順については、「AWS IAM Identity Center ユーザーガイド」の「[グループの結合](#)」を参照してください。

Amazon にアクセスするためのアクセス許可を持つ [IAM ユーザーを作成し](#) EC2、ユーザーのアクセスキーとシークレットキーをワークステーション上の安全な場所に保存する必要があります。Test Kitchen はインスタンスを作成するためにこれらの認証情報を使用します。Test Kitchen に認証情報を提供するには、ワークステーションの以下の環境変数にキーを割り当てることをお勧めします。

Warning

IAM ユーザーは長期的な認証情報を持っているため、セキュリティ上のリスクがあります。このリスクを軽減するために、これらのユーザーにはタスクの実行に必要な権限のみを付与し、不要になったユーザーを削除することをお勧めします。

- AWS_ACCESS_KEY – ユーザーのアクセスキー。AKIAIOSFODNN7EXAMPLE.
- AWS_SECRET_KEY – ユーザーのシークレットキー。wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY.

この手法によって、たとえば認証情報を含むプロジェクトを公開リポジトリにアップロードしてしまうなど不慮の事故でアカウントが侵害される機会が減ります。

クックブックをセットアップするには

1. kitchen-ec2 ドライバを使用するには、ruby-dev パッケージがシステムにインストールされている必要があります。以下の例のコマンドで aptitude を使用して Ubuntu システムでパッケージをインストールする方法を説明します。

```
sudo aptitude install ruby1.9.1-dev
```

2. kitchen-ec2 ドライバは gem です。以下のようにインストールします。

```
gem install kitchen-ec2
```

ワークステーションによっては、このコマンドに `sudo`、などの Ruby 環境マネージャーを使用することもできます [RVM](#)。この手順は、`kitchen-ec2` ドライバーのバージョン 0.8.0 でテスト済みですが、より新しいバージョンがあります。[特定のバージョン](#)をインストールするには、`gem install kitchen-ec2 -v <version number>` を実行します。

3. Test Kitchen がインスタンスへの接続に使用できる Amazon EC2SSHキーペアを指定する必要があります。Amazon EC2キーペアがない場合は、Amazon [EC2 キーペア](#)の作成方法を参照してください。キーペアは、インスタンスと同じAWSリージョンに属している必要があることに注意してください。この例では、米国西部 (北カリフォルニア) を使用します。

キーペアを選択したら、`opsworks_cookbooks` のサブディレクトリ `ec2_keys` を作成し、そのサブディレクトリにキーペアのプライベートキー (`.pem`) ファイルをコピーします。`ec2_keys` にプライベートキーを配置するのはコードをほんの少し簡素化する便宜上のもので、システムのどこに配置してもかまいません。

4. `createdir-ec2` という名前の `opsworks_cookbooks` のサブディレクトリを作成し、そのディレクトリに移動します。
5. `createdir-ec2` に、次のコンテンツを含む `metadata.rb` ファイルを追加します。

```
name "createdir-ec2"
version "0.1.0"
```

6. 「[例 1: パッケージのインストール](#)」に説明したように Test Kitchen を初期化します。次のセクションでは `.kitchen.yml`、Amazon EC2インスタンスで非常に複雑な を設定する方法について説明します。
7. `createdir-ec2` に `recipes` サブディレクトリを追加します。

Amazon の `.kitchen.yml` の設定 EC2

`kitchen-ec2` ドライバーが適切に設定された Amazon EC2インスタンスを起動するために必要な情報 `.kitchen.yml` を使用して を設定します。以下に示しているのは、米国西部 (北カリフォルニア) リージョンにある Amazon Linux インスタンス用の `.kitchen.yml` ファイルの例です。

```
driver:
  name: ec2
```

```
aws_ssh_key_id: US-East1
region: us-west-1
availability_zone: us-west-1c
require_chef_omnibus: true
security_group_ids: sg.....
subnet_id: subnet-.....
associate_public_ip: true
interface: dns

provisioner:
  name: chef_solo

platforms:
  -name: amazon
  driver:
    image_id: ami-xxxxxxx
  transport:
    username: ec2-user
    ssh_key: ../ec2_keys/US-East1.pem

suites:
  - name: default
    run_list:
      - recipe[createdir-ec2::default]
    attributes:
```

provisioner および suites セクションにデフォルト設定を使用できますが、デフォルトの driver と platforms の設定は変更する必要があります。この例では、最小限の設定リストを使用して残りはデフォルト値を受け入れます。kitchen-ec2 設定の完全なリストについては、「[Kitchen::Ec2: A Test Kitchen Driver for Amazon EC2](#)」を参照してください。

例では以下の driver 属性を設定します。前に説明したように、ユーザーのアクセスキーおよびシークレットキーは標準的な環境変数に割り当てられていることを前提とします。ドライバはこれらのキーをデフォルトで使用します。それ以外の場合、適切なキー値に設定した aws_access_key_id および aws_secret_access_key を driver 属性に追加して、明示的にキーを指定する必要があります。

名前

(必須) この属性は ec2 に設定する必要があります。

aws_ssh_key_id

(必須) この例US-East1で名前が付けられた Amazon EC2SSHキーペア名。

transport.ssh_key

(必須) aws_ssh_key_id で指定したキーのプライベートキー (.pem) ファイルです。US-East1.pem 属性は作業ディレクトリ (この例では ../opsworks/ec2_keys) を指定します。

region

(必須) インスタンスのAWSリージョン。この例では、us-west-1 で表される米国西部 (北カリフォルニア) を使用しています。

availability_zone

(オプション) インスタンスの利用可能ゾーンです。この設定を省略した場合、指定されたリージョン、米国西部 (北カリフォルニア)のデフォルトのアベイラビリティゾーンの us-west-1b が Test Kitchen で使用されます。ただし、デフォルトのゾーンがお使いのアカウントでは使用できない場合があります。その場合は、明示的に利用可能ゾーンを指定する必要があります。ちょうど、この例を用意するために使用したアカウントは us-west-1b をサポートしていないため、例では明示的に us-west-1c を指定しています。

require_chef_omnibus

true に設定すると、この設定は、オムニバス形式のインストーラが chef-client をすべてのプラットフォームインスタンスにインストールするときに必ず使用されるようにします。

security_group_ids

(オプション) インスタンスIDsに適用するセキュリティグループのリスト。この設定は、default セキュリティグループをそのインスタンスグループに適用します。セキュリティグループの進入ルールでインバウンドSSH接続が許可されていることを確認してください。許可されていない場合、Test Kitchen はインスタンスと通信できません。default セキュリティグループを使用する場合、適宜編集する必要がある可能性があります。詳細については、[「Amazon EC2 セキュリティグループ」](#)を参照してください。

subnet_id

インスタンスのターゲットサブネットの ID (該当する場合)。

associate_public_ip

インターネットからインスタンスにアクセスできるようにする場合は、Amazon にパブリック IP アドレスをインスタンスEC2に関連付けることができます。

インターフェイスからリクエスト

インスタンスにアクセスするために使用したホスト名設定タイプ。有効な値は `dns`、`public`、`private`、または `private_dns` です。この属性の値を指定しない場合、`kitchen-ec2` は次の順序でホスト名設定を指定します。この属性を省略すると、設定タイプが設定されていません。

1. DNS 名
2. パブリック IP アドレス
3. プライベート IP アドレス
4. プライベートDNS名

Important

アクセスキーとシークレットキーにアカウントの認証情報を使用するよりも、ユーザーを作成してその認証情報を Test Kitchen に提供することをお勧めします。詳細については、[AWS「アクセスキーを管理するためのベストプラクティス」](#)を参照してください。パブリックリポジトリ GitHub や Bitbucket リポジトリにアップロードするなど、パブリックにアクセス可能な場所に `.kitchen.yml` 置かないように注意してください。ご自分の認証情報が公開され、アカウントのセキュリティが侵害される恐れがあります。

`kitchen-ec2` ドライバは以下のプラットフォームをデフォルトでサポートします。

- Ubuntu-10.04
- Ubuntu-12.04
- Ubuntu-12.10
- Ubuntu-13.04
- Ubuntu-13.10
- Ubuntu-14.04
- CentOS 6.4
- Debian-7.1.0
- windows-2012r2
- windows-2008r2

これらのプラットフォームを1つ以上使用する場合は、適切なプラットフォーム名を `platforms` に追加します。 `kitchen-ec2` ドライバーは自動的に適切な を選択しAMI、SSH ユーザー名を生成します。他のプラットフォーム (この例では Amazon Linux を使用) を使用できませんが、以下の `platforms` 属性を明確に指定する必要があります。

名前

プラットフォーム名。この例では Amazon Linux を使用しているので、`name` は `amazon` に設定します。

ドライバー

`driver` 属性です。以下のものが含まれます。

- `image_id` – プラットフォームの AMI。指定されたリージョンに属している必要があります。この例では `ami-ed8e9284`、米国西部 (北カリフォルニア) リージョンAMIの Amazon Linux である を使用しています。
- `transport.username` – Test Kitchen がインスタンスとの通信に使用するSSHユーザー名。

Amazon Linux には `ec2-user` を使用します。ユーザー名が異なるAMIs場合もあります。

`.kitchen.yml` のコードを例で置き換え、`aws_access_key_id` などのアカウント固有の属性に適切な値を割り当てます。

レシピの実行

この例では、「[イテレーション](#)」のレシピを使用します。

レシピを実行するには

1. 以下のコードとともに `default.rb` という名前のファイルを作成してクックブックの `recipes` フォルダに保存します。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

2. `kitchen converge` を実行してレシピを実行します。Amazon EC2インスタンスの起動と初期化に必要な時間のため、このコマンドの完了には前の例よりも時間がかかることに注意してください。
3. [Amazon EC2コンソール](#)に移動し、米国西部 (北カリフォルニア) リージョンを選択し、ナビゲーションペインのインスタンスをクリックします。リストに新しく作成されたインスタンスが表示されます。
4. を実行してインスタンス`kitchen login`にログインします。これは、 で実行されているインスタンスの場合と同様に行います VirtualBox。 `/srv` の下に新しく作成したディレクトリがあるはずですが、任意のSSHクライアントを使用してインスタンスに接続することもできます。

次のステップ

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

この章では Chef クックブックを実行する方法の基本を見てきましたが、まだ他にもたくさんあります。

- 例では、よく使用されるリソースの使用方法を説明してきましたが、まだ他にもたくさんあります。

説明したリソースや使用した例は、使用できる属性やアクションのほんの一部でしかありません。詳細については、「[リソースおよびプロバイダについて](#)」を参照してください。

- 例ではクックブックの主要な要素、`recipes`、`attributes`、`files`、および `templates` を使用したのみです。

クックブックには、`libraries`、`definitions`、および `specs` などのさまざまな要素を含めることができます。詳細については、「[Chef のドキュメント](#)」を参照してください。

- 例では、Test Kitchen をインスタンスの開始、レシピの実行、インスタンスへのログインを行うための便利な方法としてのみ使用しました。

テスト台所は基本的にはレシピでさまざまなテストを実行するために使用するテストプラットフォームです。もしまだであれば「[Test Kitchen ウォクスルー](#)」の残りの部分をご一読ください。テスト機能について説明しています。

- [AWS OpsWorks スタックのクックブックの実装](#) では、より高度な例をいくつか示し、AWS OpsWorks スタック用のクックブックを実装する方法を示します。

AWS OpsWorks スタックのクックブックの実装

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

「[クックブックの基本](#)」では、クックブックとレシピを紹介しました。そのセクションの例では、設計がシンプルで、スタックインスタンスを含む Chef AWS OpsWorks をサポートするすべてのインスタンスで動作します。AWS OpsWorks スタック用により高度なクックブックを実装するには、通常、AWS OpsWorks スタック環境を最大限に活用する必要があります。これは、さまざまな点で標準の Chef とは異なります。

このトピックでは、AWS OpsWorks スタックインスタンスのレシピを実装するための基本について説明します。

Note

クックブックの実装に慣れていない場合は、先に「[クックブックの基本](#)」をご覧ください。

トピック

- [スタックの Linux AWS OpsWorks インスタンスでのレシピの実行](#)
- [Windows インスタンスでのレシピの実行](#)
- [Windows PowerShell スクリプトの実行](#)
- [Vagrant でのスタック設定およびデプロイ属性の模倣](#)

- [スタック設定およびデプロイ属性値の使用](#)
- [Linux インスタンスでの外部クックブック Berkshelf の使用](#)
- [SDK for Ruby の使用: Amazon S3 からファイルをダウンロードする](#)
- [Windows ソフトウェアのインストール](#)
- [組み込み属性の上書き](#)
- [組み込みテンプレートの上書き](#)

スタックの Linux AWS OpsWorks インスタンスでのレシピの実行

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Test Kitchen と Vagrant は、クックブックを実装するためのシンプルで効率的な方法を提供しますが、クックブックのレシピが本番環境で正しく実行されることを確認するには、AWS OpsWorks スタックインスタンスで実行する必要があります。このトピックでは、カスタムクックブックを AWS OpsWorks スタックの Linux インスタンスにインストールし、単純なレシピを実行する方法について説明します。また、レシピのバグを効率的に修正するためのヒントも紹介します。

Windows インスタンスでレシピを実行する方法の詳細については、「[Windows インスタンスでのレシピの実行](#)」を参照してください。

トピック

- [レシピの作成と実行](#)
- [レシピの自動実行](#)
- [レシピのトラブルシューティングと修正](#)

レシピの作成と実行

最初に、スタックを作成する必要があります。以下に、この例のスタックを作成する方法を簡単に示します。詳細については、「[新しいスタックを作成する](#)」を参照してください。

スタックを作成するには

1. [AWS OpsWorks スタックコンソール](#)を開いて [Add Stack] をクリックします。
2. 次の設定を指定し、その他の設定はデフォルト値を受け入れて、[Add Stack] をクリックします。

- 名前 – OpsTest
- デフォルトSSHキー – Amazon EC2キーペア

Amazon EC2キーペアを作成する必要がある場合は、「[Amazon EC2 キーペア](#)」を参照してください。キーペアは、インスタンスと同じAWSリージョンに属している必要があることに注意してください。この例では、デフォルトに米国西部(オレゴン) リージョンを使用します。

3. [Add a layer] をクリックし、次の設定を使用してスタックに[カスタムレイヤーを追加](#)します。
 - 名前 – OpsTest
 - Short name (短縮名) - opstest

Linux スタックの場合、どのレイヤータイプでも実際には動作しますが、この例は他のレイヤータイプによってインストールされるパッケージを必要としないので、カスタムレイヤーが最も簡単なアプローチです。

4. デフォルト設定でレイヤーに [24/7 インスタンスを追加](#)し、[起動](#)します。

インスタンスの起動中に (通常は数分かかります)、クックブックを作成できます。この例では、「[条件付きロジック](#)」のレシピをわずかに変更したバージョンを使用して、データディレクトリを作成します。ディレクトリの名前は、プラットフォームに依存します。

クックブックをセットアップするには

1. opsworks_cookbooks 内に opstest という名前のディレクトリを作成し、そこに移動します。
2. 以下の内容で metadata.rb ファイルを作成し、opstest に保存します。

```
name "opstest"  
version "0.1.0"
```

3. recipes 内に opstest ディレクトリを作成します。

4. 次のレシピで `default.rb` ファイルを作成し、`recipes` ディレクトリに保存します。

```
Chef::Log.info("*****Creating a data directory.*****")

data_dir = value_for_platform(
  "centos" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
)

directory data_dir do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

レシピによってメッセージが記録されますが、そのために `Chef::Log.info` が呼び出されることに注意してください。この例では Test Kitchen を使用していないため、`log` メソッドはあまり有用ではありません。 `Chef::Log.info` はメッセージを Chef ログに配置します。これは Chef の実行が完了した後に読み取ることができます。AWS OpsWorks スタックは、後で説明するように、これらのログを簡単に表示する方法を提供します。

Note

Chef ログには、通常、多くの日常的で比較的重要ではない情報が含まれています。メッセージテキストを「*」文字で囲んでおくと、見分けるのが容易になります。

5. `opsworks_cookbooks` の `.zip` アーカイブを作成します。AWS OpsWorks スタックインスタンスにクックブックをインストールするには、そのクックブックをリポジトリに保存し、インスタンスにクックブックをダウンロードするために必要な情報を AWS OpsWorks スタックに提供する必要があります。サポートされている任意のリポジトリのタイプにクックブックを保存できます。この例では、クックブックを含むアーカイブファイルを Amazon S3 バケットに保存します。クックブックリポジトリの詳細については、「[クックブックリポジトリ](#)」を参照してください。

Note

わかりやすいように、この例では `opsworks_cookbooks` ディレクトリ全体をアーカイブします。ただし、1 AWS OpsWorks つのクックブックしか使用しない場合でも、スタックは内のすべてのクックブックをインスタンス `opsworks_cookbooks` にダウンロードします。例のクックブックのみをインストールするには、別の親ディレクトリを作成し、そのディレクトリに `opstest` を移動します。次に、親ディレクトリの `.zip` アーカイブを作成し、`opsworks_cookbooks.zip` の代わりに使用します。Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、[「S3 バケットを空にする方法」](#) または [「S3 バケットを削除する方法」](#) を参照してください。

6. [アーカイブを Amazon S3 バケットにアップロードし、アーカイブを公開](#)して、アーカイブのを記録しますURL。

これで、クックブックをインストールし、レシピを実行できるようになりました。

レシピを実行するには

1. [カスタムクックブックを有効にするようにスタックを編集](#)し、以下の設定を指定します。

- Repository type (リポジトリタイプ) - S3 Archive (S3 アーカイブ)
- リポジトリ URL – 前に記録URLしたクックブックアーカイブ

その他の設定にはデフォルト値を使用し、[Save] をクリックしてスタック設定を更新します。

2. [\[Update Custom Cookbooks\] スタックコマンドを実行](#)します。スタックのインスタンスにあるカスタムクックブックの最新バージョンがインストールされます。以前のバージョンのクックブックがある場合は、このコマンドによって上書きされます。
3. 実行するレシピを `opstest::default` に設定してレシピの実行スタックコマンドを実行することで、レシピを実行します。このコマンドは、`opstest::default` から成る実行リストで Chef 実行を開始します。

レシピが正常に実行された後で、それを検証できます。

opstest を検証するには

1. 最初のステップは、[Chef ログ](#)を調べることです。opstest1 インスタンスの [Log] (ログ) 列の [show] (表示) をクリックして、ログを表示します。下へスクロールすると、下部にログメッセージが表示されます。

```
...
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
attributes/customize.rb in the cache.
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
metadata.rb in the cache.
[2014-07-31T17:01:46+00:00] INFO: *****Creating a data directory.*****
[2014-07-31T17:01:46+00:00] INFO: Processing template[/etc/hosts] action create
(opsworks_stack_state_sync::hosts line 3)
...
```

2. [SSH を使用してインスタンスにログイン](#)し、 の内容を一覧表示します/srv/www/。

すべてのステップを実行した場合は、予期していた /srv/www/config ディレクトリではなく、/srv/www/shared ディレクトリが表示されます。次のセクションでは、このようなバグをすばやく修正するためのガイドラインについて説明します。

レシピの自動実行

[Execute Recipes] コマンドは、カスタムレシピをテストするための便利な方法であるため、ここでの例のほとんどで使用しています。ただし、実際には、インスタンスの起動後やアプリケーションのデプロイ時など、インスタンスのライフサイクルの標準ポイントでレシピを実行します。AWS OpsWorks スタックは、各レイヤーの一連の[ライフサイクルイベント](#)をサポートすることで、インスタンスでの実行レシピを簡素化します。Setup、Configure、Deploy、Undeploy、および Shutdown。適切なライフサイクルイベントにレシピを割り当てることで、AWS OpsWorks スタックでレイヤーのインスタンスでレシピを自動的に実行させることができます。

通常は、インスタンスの起動が完了したら (Setup イベント)、すぐにディレクトリを作成します。以下に示しているのは、前の例で作成した同じスタックを使用して、セットアップ時にサンプルレシピを実行する方法です。他のイベントにも同じ手順を使用できます。

セットアップ時にレシピを自動的に実行するには

1. ナビゲーションペインでレイヤーを選択し、OpsTest レイヤーのレシピリンクの横にある鉛筆アイコンを選択します。

2. `opstest::default` をレイヤーの セットアップ レシピに追加します。次に、[+] をクリックしてレイヤーに追加し、保存を選択して設定を保存します。
3. [Instances] を選択し、レイヤーに別のインスタンスを追加して、そのインスタンスを起動します。

インスタンスの名前を `opstest2` に変更する必要があります。起動が完了すると、AWS OpsWorks スタックは `opstest::default` を実行します。

4. `opstest2` インスタンスがオンラインになった後、`/srv/www/shared` があることを確認します。

Note

Setup、Configure、または Deploy イベントにレシピを割り当てた場合は、[スタックコマンド](#) (Setup および Configure) または [デプロイコマンド](#) (Deploy) を使用してイベントをトリガーすることで、それらのレシピを手動で実行することもできます。イベントに複数のレシピが割り当てられている場合、これらのコマンドによってそれらのレシピがすべて実行されます。

レシピのトラブルシューティングと修正

予期した結果が得られなかったり、レシピが正常に実行されなかったりする場合、通常は Chef ログの調査からトラブルシューティングを始めます。これには、実行の詳細な説明と、レシピからのインラインログメッセージが含まれます。ログは、レシピが単純に失敗した場合に、特に便利です。そのような場合には、エラーが記録されます (スタックトレースなど)。

この例のようにレシピが成功した場合、Chef ログはあまり有用ではありません。この場合は、次のように特にレシピの最初の数行を詳しく調べることで、問題の原因を把握できます。

```
Chef::Log.info("*****Creating a data directory.*****")

data_dir = value_for_platform(
  "centos" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
)
...
```

Vagrant でレシピをテストしている場合は CentOS が Amazon Linux の適切な代役ですが、ここでは実際の Amazon Linux インスタンスで実行しています。Amazon Linux のプラットフォーム値は `amazon` です。この値は `value_for_platform` 呼び出しには含まれていないため、レシピはデフォルトで `/srv/www/config` を作成します。トラブルシューティングの詳細については、[デバッグとトラブルシューティングのガイド](#) を参照してください。

これで問題を特定できたので、レシピを更新し、修正を検証する必要があります。元のソースファイルに戻り、`default.rb` を更新し、新しいアーカイブを Amazon S3 にアップロードするなどの操作を行うことができます。ただし、そのプロセスには手間と時間がかかる場合があります。以下に、例のバグのような単純なレシピのバグで特に便利な、よりすばやいアプローチを示します。インスタンスのレシピを編集する方法です。

インスタンスのレシピを編集するには

1. SSH を使用してインスタンスにログインし、`sudo su` を実行して権限を昇格します。クックブックディレクトリにアクセスするには、ルート権限が必要です。
2. AWS OpsWorks スタックはクックブックを `/opt/aws/opsworks/current/site-cookbooks/` に保存するため `/opt/aws/opsworks/current/site-cookbooks/` に移動します `/opt/aws/opsworks/current/site-cookbooks/opstest/recipes`。

Note

AWS OpsWorks スタックでは、クックブックのコピーも `/opt/aws/opsworks/current/merged-cookbooks` に保存されます。そのクックブックは編集しないでください。レシピを実行すると、AWS OpsWorks スタックはクックブックを `.../site-cookbooks` から `.../merged-cookbooks` にコピーするため `.../merged-cookbooks`、で行った変更は上書き `.../merged-cookbooks` されます。

3. インスタンスでテキストエディタを使用して `default.rb` を編集し、`centos` を `amazon` に置き換えます。これで、レシピは次のようになります。

```
Chef::Log.info("*****Creating a data directory.*****")

data_dir = value_for_platform(
  "amazon" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
)
```

...

修正を検証するには、もう一度 Execute Recipe スタックコマンドを実行して、レシピを実行します。現在、インスタンスに `/srv/www/shared` ディレクトリがあるはずですが、レシピをさらに変更する必要がある場合は、Execute Recipe を何度でも実行できます。コマンドを実行するたびにインスタンスを停止および再開する必要はありません。レシピが正常に動作していることを確認できたら、ソースブックのコードも必ず更新してください。

Note

レシピをライフサイクルイベントに割り当てて AWS OpsWorks スタックが自動的に実行するようにした場合は、いつでもレシピの実行を使用してレシピを再実行できます。AWS OpsWorks スタックコンソールを使用して適切なイベントを手動でトリガーすることで、インスタンスを再起動せずにレシピを何度でも再実行することもできます。ただしこのアプローチでは、イベントのすべてのレシピが実行されます。次の点に注意してください。

- Setup または Configure イベントをトリガーするには、[スタックコマンド](#)を使用します。
- Deploy または Undeploy イベントをトリガーするには、[デプロイコマンド](#)を使用します。

Windows インスタンスでのレシピの実行

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

このトピックでは、基本的に「[Linux インスタンスでのレシピの実行](#)」の簡略版であり、Windows スタックでレシピを実行する方法を示しています。最初に「[Linux インスタンスでのレシピの実行](#)」を参照することをお勧めします。より詳細に説明されており、内容の大部分がいずれかのタイプのオペレーティングシステムに関連しているためです。

AWS OpsWorks Stacks Linux インスタンスでレシピを実行する方法については、「」を参照してください。[Linux インスタンスでのレシピの実行](#)。

トピック

- [RDP アクセスの有効化](#)
- [レシピの作成と実行](#)
- [レシピの自動実行](#)

RDP アクセスの有効化

開始する前に、まだ設定していない場合は、インスタンス RDP へのアクセスを許可するインバウンドルールを使用してセキュリティグループを設定する必要があります。スタックを作成するとき、そのグループが必要になります。

リージョンに最初のスタックを作成すると、AWS OpsWorks スタックによって一連のセキュリティグループが作成されます。これにはAWS-OpsWorks-RDP-Server、スタックがRDPアクセスを許可するためにすべての Windows AWS OpsWorks インスタンスにアタッチする のような名前のもが含まれます。ただし、デフォルトでは、このセキュリティグループにはルールがないため、インスタンスRDPへのアクセスを許可するインバウンドルールを追加する必要があります。

RDP アクセスを許可するには

1. [Amazon EC2コンソール](#)を開き、スタックのリージョンに設定し、ナビゲーションペインからセキュリティグループを選択します。
2. AWS-OpsWorksRDP--Server を選択し、インバウンドタブを選択し、編集を選択します。
3. 以下の設定でルールを追加します。
 - 型 – RDP
 - Source (ソース) – 許可される送信元 IP アドレス。

通常、IP アドレスまたは指定された IP アドレス範囲 (通常は企業 IP アドレス範囲) からのインバウンドRDPリクエストを許可します。

Note

後で説明するように、通常のユーザーのRDPアクセスを許可するには、ユーザーのアクセス許可も編集する必要があります。

詳細については、「[RDP でのログイン](#)」を参照してください。

レシピの作成と実行

以下に、この例のスタックを作成する方法を簡単に示します。詳細については、「[新しいスタックを作成する](#)」を参照してください。

Stack を作成する

1. [AWS OpsWorks スタックコンソール](#)を開いて [Add Stack] を選択します。以下の設定を指定し、その他の設定はデフォルト値を受け入れて、[Add Stack] を選択します。

- 名前 – WindowsRecipeTest
- リージョン – 米国西部 (オレゴン)

この例はいずれのリージョンでも動作しますが、チュートリアルでは米国西部 (オレゴン) を使用することをお勧めします。

- Default operating system (デフォルトのオペレーティングシステム) – Microsoft Windows Server 2012 R2
2. [Add a layer] を選択し、以下の設定を使用してスタックに[カスタムレイヤーを追加](#)します。
 - 名前 – RecipeTest
 - Short name (短縮名) – recipetest
 3. デフォルト設定の [24/7 インスタンス](#)を RecipeTestレイヤーに追加して[起動](#)します。

AWS OpsWorks スタックは、このインスタンスに自動的に AWS-OpsWorks-RDP-Server を割り当てます。これにより、承認されたユーザーはインスタンスにログインできます。

4. アクセス許可を選択し、編集して、SSH/RDP と sudo/admin を選択します。通常のユーザーは、インスタンスにログインするために、AWS-OpsWorks-RDP-Server セキュリティグループに加えて、この権限が必要です。

Note

管理者としてログインすることもできますが、それには別の手順が必要です。詳細については、「[RDP でのログイン](#)」を参照してください。

インスタンスの起動中に (通常は数分かかります)、クックブックを作成できます。この例のレシピによってデータディレクトリが作成されます。これは基本的に「[例 3: ディレクトリの作成](#)」からのレシピであり、Windows 用に修正しています。

Note

AWS OpsWorks スタックの Windows インスタンスにクックブックを実装するときは、スタックの Linux AWS OpsWorks インスタンスにクックブックを実装するときとは異なるディレクトリ構造を使用します。詳細については、「[クックブックリポジトリ](#)」を参照してください。

クックブックをセットアップするには

1. `windowstest` という名前のディレクトリを作成し、そのディレクトリに移動します。
2. 以下の内容で `metadata.rb` ファイルを作成し、`windowstest` に保存します。

```
name "windowstest"
version "0.1.0"
```

3. `recipes` 内に `windowstest` ディレクトリを作成します。
4. 次のレシピで `default.rb` ファイルを作成し、`recipes` ディレクトリに保存します。

```
Chef::Log.info("*****Creating a data directory.*****")

directory 'C:\data' do
  rights :full_control, 'instance_name\username'
  inherits false
  action :create
end
```

username をユーザー名に置き換えます。

5. リポジトリへのクックブックの保存

AWS OpsWorks スタックインスタンスにクックブックをインストールするには、そのクックブックをリポジトリに保存し、インスタンスにクックブックをダウンロードするために必要な情報を AWS OpsWorks スタックに提供する必要があります。S3 バケットまたは Git リポジトリにアーカイブファイルとして Windows のクックブックを保存できます。この例では、S3 バケットを使用しているため、`windowstest` ディレクトリの `.zip` アーカイブを作成する必要があります。クックブックリポジトリの詳細については、「[クックブックリポジトリ](#)」を参照してください。

6. [アーカイブを S3 バケットにアップロード](#)し、[アーカイブを公開](#)して、アーカイブの `URL` を記録します。プライベートアーカイブを使用することもできますが、この例ではパブリックアーカイブで十分であり、作業がいくらか簡単になります。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、[「S3 バケットを空にする方法」](#)または [「S3 バケットを削除する方法」](#)を参照してください。

これで、クックブックをインストールし、レシピを実行できるようになりました。

レシピを実行するには

1. [カスタムクックブックを有効にするようにスタックを編集](#)し、以下の設定を指定します。

- Repository type (リポジトリタイプ) – S3 Archive (アーカイブ)
- リポジトリ URL – 前に記録URLしたクックブックアーカイブ

その他の設定ではデフォルト値を受け入れ、[Save] を選択してスタック設定を更新します。

2. [\[Update Custom Cookbooks\] スタックコマンドを実行](#)します。カスタムクックブックの最新バージョンがスタックのインスタンス (オンラインインスタンスを含む) にインストールされます。以前のバージョンのクックブックがある場合は、このコマンドによって上書きされます。
3. カスタムクックブックの更新が終了したら、実行するレシピを `windowstest::default` に設定して[レシピの実行スタックコマンド](#)を実行することで、レシピを実行します。このコマンドによって Chef 実行が開始され、レシピで構成される実行リストが渡されます。

レシピが正常に実行された後で、それを検証できます。

`windowstest` を検証するには

1. [Chef ログ](#)を調べます。opstest1 インスタンスの [Log] (ログ) 列で [show] (表示) を選択すると、ログが表示されます。下へスクロールすると、下部にログメッセージが表示されます。

```
...
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
attributes/customize.rb in the cache.
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
metadata.rb in the cache.
```



```
[2014-07-31T17:01:46+00:00] INFO: *****Creating a data directory.*****  
[2014-07-31T17:01:46+00:00] INFO: Processing template[/etc/hosts] action create  
(opsworks_stack_state_sync::hosts line 3)  
...
```

2. インスタンスを選択し、インスタンスのアクション列で rdp を選択し、適切な有効期限のRDP パスワードをリクエストします。DNS 名前、ユーザー名、パスワードをコピーします。その後、その情報を Windows リモートデスクトップ接続RDPクライアントなどのクライアントで使用して、インスタンスにログインし、c:\dataが存在することを確認できます。詳細については、「[RDP でのログイン](#)」を参照してください。

Note

レシピが正しく実行されない場合、トラブルシューティングのヒントについては「[レシピのトラブルシューティングと修正](#)」を参照してください。それらのヒントのほとんどは Windows インスタンスにも当てはまります。インスタンスでレシピを編集して修正をテストする場合は、AWS OpsWorks スタックがカスタムクックブックをインストールする C:\chef\cookbooks ディレクトリでクックブックを探します。

レシピの自動実行

[Execute Recipes] コマンドは、カスタムレシピをテストするための便利な方法であるため、ここでの例のほとんどで使用しています。ただし、実際には、インスタンスの起動後やアプリケーションのデプロイ時など、インスタンスのライフサイクルの標準ポイントでレシピを実行します。AWS OpsWorks スタックは、各レイヤーの一連の[ライフサイクルイベント](#)をサポートすることで、インスタンスでの実行レシピを簡素化します。設定、設定、デプロイ、デプロイ解除、シャットダウン。適切なライフサイクルイベントにレシピを割り当てることで、AWS OpsWorks スタックでレイヤーのインスタンスでレシピを自動的に実行させることができます。

通常は、インスタンスの起動が完了したら (Setup イベント)、すぐにディレクトリを作成します。以下に示しているのは、前の例で作成した同じスタックを使用して、セットアップ時にサンプルレシピを実行する方法です。他のイベントにも同じ手順を使用できます。

セットアップ時にレシピを自動的に実行するには

1. ナビゲーションペインでレイヤーを選択し、RecipeTest レイヤーのレシピリンクの横にある鉛筆アイコンを選択します。

2. `windowstest::default` をレイヤーの Setup (セットアップ) レシピに追加します。次に、[+] を選択してレイヤーに追加し、[Save] (保存) を選択して設定を保存します。
3. [Instances] を選択し、レイヤーに別のインスタンスを追加して、そのインスタンスを起動します。

インスタンスの名前を `recipetest2` に変更する必要があります。起動が完了すると、AWS OpsWorks スタックは `windowstest::default` を実行します。

4. `recipetest2` インスタンスがオンラインになった後、`c:\data` があることを確認します。

Note

Setup、Configure、または Deploy イベントにレシピを割り当てた場合は、[スタックコマンド](#) (Setup および Configure) または [デプロイコマンド](#) (Deploy) を使用してイベントをトリガーすることで、それらのレシピを手動で実行することもできます。イベントに複数のレシピが割り当てられている場合、これらのコマンドによってそれらのレシピがすべて実行されます。

Windows PowerShell スクリプトの実行

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

以下の例では、「[Windows インスタンスでのレシピの実行](#)」の例を実行済みであることを前提としています。実行済みでない場合は、最初にその例を実行する必要があります。特に、インスタンス [RDPへのアクセスを有効にする](#) 方法について説明します。

レシピが Windows インスタンスでタスクを実行する方法の 1 つ、特に対応する Chef リソースを持たないタスクは、レシピに Windows PowerShell スクリプトを実行することです。このセクションでは、Windows PowerShell スクリプトを使用して Windows 機能をインストールする方法について説明し、基本について説明します。

[powershell_script](#) リソースは、インスタンスで Windows PowerShell コマンドレットを実行します。次の例では、[Install-WindowsFeature cmdlet](#) を使用してインスタンスに XPS ビューワーをインストールします。

以下に、この例のスタックを作成する方法を簡単に示します。詳細については、「[新しいスタックを作成する](#)」を参照してください。

Stack を作成する

1. [AWS OpsWorks スタックコンソール](#)を開いて [Add Stack] を選択します。次の設定を指定し、その他の設定はデフォルト値を受け入れて、[Add Stack] をクリックします。

- 名前 – PowerShellTest
- リージョン – 米国西部 (オレゴン)

この例はいずれのリージョンでも動作しますが、チュートリアルでは米国西部 (オレゴン) を使用することをお勧めします。

- Default operating system (デフォルトのオペレーティングシステム) – Microsoft Windows Server 2012 R2

2. [Add a layer] を選択し、以下の設定を使用してスタックに[カスタムレイヤーを追加](#)します。

- 名前 – PowerShell
- Short name (短縮名) – powershell

3. デフォルト設定の [24/7 インスタンス](#)を PowerShell レイヤーに追加して[起動](#)します。

4. アクセス許可を選択し、編集して、SSH/RDP と sudo/admin を選択します。通常のコマンドとしてインスタンスにログインするには、AWS-OpsWorks-RDP-Server セキュリティグループに加えて、この権限が必要です。

インスタンスの起動中に (通常は数分かかります)、クックブックを作成できます。この例のレシピによってデータディレクトリが作成されます。これは基本的に「[例 3: ディレクトリの作成](#)」からのレシピであり、Windows 用に修正しています。

クックブックをセットアップするには

1. powershell という名前のディレクトリを作成し、そのディレクトリに移動します。
2. 以下の内容で metadata.rb ファイルを作成し、windowstest に保存します。

```
name "powershell"  
version "0.1.0"
```

3. recipes 内に powershell ディレクトリを作成します。
4. 次のレシピで default.rb ファイルを作成し、recipes ディレクトリに保存します。

```
Chef::Log.info("*****Installing XPS.*****")  
  
powershell_script "Install XPS Viewer" do  
  code <<-EOH  
    Install-WindowsFeature XPS-Viewer  
  EOH  
  guard_interpreter :powershell_script  
  not_if "(Get-WindowsFeature -Name XPS-Viewer).installed"  
end
```

- powershell_script リソースは コマンドレットを実行してビューXPSワーをインストールします。

この例では、1つのコマンドレットのみを実行していますが、code ブロックには任意の数のコマンドラインを含めることができます。

- guard_interpreter 属性は、Chef に Windows の 64 ビットバージョンを使用するように指示します PowerShell。
 - not_if ガード属性を指定すると、Chef によってインストール済みの機能はインストールされません。
5. .zip ディレクトリの powershell アーカイブを作成します。
 6. [アーカイブを Amazon S3 バケットにアップロード](#)し、[アーカイブを公開](#)して、アーカイブのURLを記録します。プライベートアーカイブを使用することもできますが、この例ではパブリックアーカイブで十分であり、作業がいくらか簡単になります。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、[「S3 バケットを空にする方法」](#)または[「S3 バケットを削除する方法」](#)を参照してください。

これで、クックブックをインストールし、レシピを実行できるようになりました。

レシピを実行するには

1. [カスタムクックブックを有効にするようにスタックを編集](#)し、以下の設定を指定します。

- Repository type (リポジトリタイプ) – S3 Archive (アーカイブ)
- リポジトリ URL – 前に記録URLしたクックブックアーカイブ

その他の設定ではデフォルト値を受け入れ、[Save] を選択してスタック設定を更新します。

2. [\[Update Custom Cookbooks\] スタックコマンドを実行](#)して、カスタムクックブックの最新バージョンをインスタンスにインストールします。
3. カスタムクックブックの更新が終了したら、実行するレシピを **powershell::default** に設定して[レシピの実行スタックコマンド](#)を実行することで、レシピを実行します。

Note

この例では、便宜上 Execute Recipes を使用していますが、通常 AWS OpsWorks、適切なライフサイクルイベントに割り当てることで、スタックで[レシピを自動的に実行](#)します。このようなレシピは、イベントを手動でトリガーすることによって実行できます。Setup および Configure イベントをトリガーするにはスタックコマンドを使用し、Deploy および Undeploy イベントをトリガーするには[デプロイコマンド](#)を使用できます。

レシピが正常に実行された後で、それを検証できます。

PowerShell レシピを確認するには

1. [Chef ログ](#)を調べます。powershell1 インスタンスの [Log] (ログ) 列の [show] (表示) をクリックすると、ログが表示されます。下へスクロールすると、下部にログメッセージが表示されます。

```
...
[2015-04-27T18:12:09+00:00] INFO: Storing updated cookbooks/powershell/metadata.rb
in the cache.
[2015-04-27T18:12:09+00:00] INFO: *****Installing XPS.*****
[2015-04-27T18:12:09+00:00] INFO: Processing powershell_script[Install XPS Viewer]
action run (powershell::default line 3)
[2015-04-27T18:12:09+00:00] INFO: Processing powershell_script[Guard resource]
action run (dynamically defined)
[2015-04-27T18:12:42+00:00] INFO: powershell_script[Install XPS Viewer] ran
successfully
...
```

2. [RDP を使用してインスタンスにログイン](#)し、スタートメニューを開きます。XPSビューワーには Windows のアクセサリが表示されます。

Vagrant でのスタック設定およびデプロイ属性の模倣

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

このトピックは Linux インスタンスにのみ当てはまります。Test Kitchen はまだ Windows をサポートしていないため、すべての Windows の例を AWS OpsWorks スタックインスタンスで実行します。

AWS OpsWorks スタックは、すべてのライフサイクルイベントについて、スタック内の各インスタンスのノードオブジェクトに [スタック設定およびデプロイ属性](#) を追加します。これらの属性から、各レイヤーの設定とそのオンラインインスタンス、デプロイされた各アプリケーションの設定など、スタック設定のスナップショットが渡されます。これらの属性はノードオブジェクトにあるため、どのレシピでもアクセスできます。スタックインスタンスのほとんどのレシピでは、これらの属性の 1 AWS OpsWorks つ以上を使用します。

Vagrant ボックスで実行されているインスタンスは AWS OpsWorks スタックによって管理されないため、そのノードオブジェクトにはデフォルトでスタック設定およびデプロイ属性は含まれません。ただし、適切な属性のセットを Test Kitchen 環境に追加できます。次に、Test Kitchen はインスタンスのノードオブジェクトに属性を追加し、レシピは AWS OpsWorks スタックインスタンスと同様に属性にアクセスできます。

このトピックでは、適切なスタック設定およびデプロイ属性のコピーを取得し、インスタンスにインストールして、属性にアクセスする方法について説明します。

Note

Test Kitchen を使用してレシピでテストを実行する場合、[Baliquehai](#) はスタック設定とデプロイをモックする代替方法を提供しますJSON。

クックブックをセットアップするには

1. `printjson` という名前の `opsworks_cookbooks` のサブディレクトリを作成し、そのディレクトリに移動します。
2. 「[例 1: パッケージのインストール](#)」で説明されているとおりに Test Kitchen を初期化および設定します。
3. `printjson` と `recipes` にサブディレクトリを 2 つ追加します `environments`。

適切な定義で属性ファイルをクックブックに追加することで、スタック設定およびデプロイ属性を模倣することができますが、より良いアプローチは Test Kitchen 環境を使用することです。通常は、次のいずれかの方法でログインします。

- `.kitchen.yml` に属性定義を追加します。

このアプローチは、属性の数がほんの数個の場合に最も便利です。詳細については、「[kitchen.yml](#)」を参照してください。

- 環境ファイルで属性を定義し、そのファイルを `.kitchen.yml` で参照します。

このアプローチは、環境ファイルがすでに JSON 形式であるため、通常、スタック設定およびデプロイ属性に適しています。属性のコピーは、適切な AWS OpsWorks スタックインスタンスから JSON 形式で取得し、貼り付けるだけです。すべての例で環境ファイルを使用します。

クックブックのスタック設定およびデプロイ属性を作成する最も簡単な方法は、適切に設定されたスタックを作成し、インスタンスからとして結果の属性をコピーすることですJSON。Test Kitchen 環境ファイルを管理しやすいようにするために、レシピに必要な属性のみを持つJSONように編集できます。この章の例では、ロードバランサー[Chef 11 Linux スタックの使用開始](#)、PHPアプリケーションサーバー、My データベースサーバーを備えたシンプルなPHPアプリケーションサーバースタックであるのSQLスタックに基づいています。

スタック設定とデプロイを作成するには JSON

1. 「」の説明 MyStack に従って を作成します。これには [Chef 11 Linux スタックの使用開始](#) のデプロイも含まれますimplePHPApp。必要に応じて、 で を呼び出す 2 番目の PHP App Server インスタンスを省略できます[ステップ 4: スケールアウトする MyStack](#)。この例では、これらの属性を使用しません。
2. まだ起動していない場合は、php-app1インスタンスを起動し、 [でログインSSH](#)します。
3. ターミナルウィンドウで、以下の[エージェント CLI](#) コマンドを実行します。

```
sudo opsworks-agent-cli get_json
```

このコマンドは、インスタンスの最新のスタック設定およびデプロイ属性を JSON 形式でターミナルウィンドウに出力します。

4. JSON を .json ファイルにコピーし、ワークステーションの便利な場所に保存します。詳細は SSHクライアントによって異なります。例えば、Windows で PuTTY を使用している場合は、ターミナルウィンドウのすべてのテキストを Windows クリップボードにコピーする Copy All to Clipboard コマンドを実行できます。その後、.json ファイルに内容を貼り付け、ファイルを編集して余分なテキストをすべて削除できます。
5. 必要に応じて を編集 MyStack JSONします。スタック設定およびデプロイ属性は多数あるため、クックブックでは通常、その一部しか使用しません。環境ファイルを管理しやすいJSONように、 を編集して元の構造を保持し、クックブックが実際に使用する属性のみを含めることができます。

この例では、 ['id]と MyStack JSONの 2 つの ['opsworks'] ['stack']属性のみを含む、の高度に編集されたバージョンを使用しています ['name']。編集したバージョンの MyStack JSON を次のように作成します。

```
{  
  "opsworks": {
```



```
"stack": {
  "name": "MyStack",
  "id": "42dfd151-6766-4f1c-9940-ba79e5220b58",
},
},
}
```

これをインスタスのノードオブジェクトJSONに入れるには、Test Kitchen 環境に追加する必要があります。

スタック設定およびデプロイ属性を Test Kitchen 環境に追加するには

1. 以下の内容で `test.json` という名前の環境ファイルを作成し、クックブックの `environments` フォルダに保存します。

```
{
  "default_attributes": {
    "opsworks" : {
      "stack" : {
        "name" : "MyStack",
        "id" : "42dfd151-6766-4f1c-9940-ba79e5220b58"
      }
    }
  },
  "chef_type" : "environment",
  "json_class" : "Chef::Environment"
}
```

環境ファイルには以下の要素があります。

- `default_attributes` – JSON形式のデフォルトの属性。

これらの属性は、すべてのスタック設定およびデプロイ `default` 属性で使用されるタイプである JSON 属性タイプを使用してノードオブジェクトに追加されます。この例では、前JSON に示したスタック設定とデプロイの編集済みバージョンを使用します。

- `chef_type` – この要素を `environment` に設定します。
- `json_class` – この要素を `Chef::Environment` に設定します。

2. `.kitchen.yml` を編集して、Test Kitchen 環境を次のように定義します。

```
---
driver:
  name: vagrant

provisioner:
  name: chef_solo
  environments_path: ./environments

platforms:
  - name: ubuntu-12.04

suites:
  - name: printjson
    provisioner:
      solo_rb:
        environment: test
    run_list:
      - recipe[printjson::default]
    attributes:
```

`kitchen init` によって作成されたデフォルトの `.kitchen.yml` に以下の要素を追加して、環境を定義します。

provisioner

以下の要素を追加します。

- `name` – この要素を `chef_solo` に設定します。

AWS OpsWorks スタック環境をより詳細にレプリケートするには、[Chef solo の代わりに Chef クライアントのローカルモード](#)を使用できます。ローカルモードは、リモートサーバーではなくインスタンスでローカルに実行される Chef サーバーの軽量バージョン (Chef Zero) を使用する Chef Client オプションです。これによって、レシピはリモートサーバーに接続しなくても、検索やデータバッグなどの Chef サーバー機能を使用できます。

- `environments_path` – クックブックのサブディレクトリです。この例では、環境ファイル `./environments` が含まれています。

suites:provisioner

`solo_rb` 要素を環境ファイル名 (.json 拡張子は含めない) に設定し、`environment` 要素を追加します。この例では、`environment` を `test` に設定します。

3. 以下の内容で `default.rb` という名前のレシピファイルを作成し、クックブックの `recipes` ディレクトリに保存します。

```
log "Stack name: #{node['opsworks']['stack']['name']}"
log "Stack id: #{node['opsworks']['stack']['id']}"
```

このレシピは、環境に追加した 2 個のスタック設定およびデプロイメント 値を単純に記録します。レシピは仮想ボックスでローカルに実行されていますが、レシピが AWS OpsWorks スタックインスタンスで実行されている場合と同じノード構文を使用してこれらの属性を参照します。

4. `kitchen converge` を実行します。次のようなログ出力が表示されます。

```
...
Converging 2 resources
Recipe: printjson::default
  * log[Stack name: MyStack] action write[2014-07-01T23:14:09+00:00] INFO:
    Processing log[Stack name: MyStack] action write (printjson::default line 1)

[2014-07-01T23:14:09+00:00] INFO: Stack name: MyStack

  * log[Stack id: 42dfd151-6766-4f1c-9940-ba79e5220b58] action
    write[2014-07-01T23:14:09+00:00] INFO: Processing log[Stack id:
    42dfd151-6766-4f1c-9940-ba79e5220b58] action write (printjson::default line 2)

[2014-07-01T23:14:09+00:00] INFO: Stack id: 42dfd151-6766-4f1c-9940-ba79e5220b58
...
```

スタック設定およびデプロイ属性値の使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソ

リユースに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

レシピでは、スタック設定やデプロイされたアプリケーションに関する情報がしばしば必要になります。たとえば、設定ファイルを作成するためにスタックの IP アドレスのリストが必要になったり、ログディレクトリを作成するためにアプリケーションのデプロイメントディレクトリが必要になることがあります。このデータを中央サーバーに保存する代わりに、AWS OpsWorks スタックは、各ライフサイクルイベントについて、各インスタンスのノードオブジェクトに一連のスタック設定およびデプロイ属性をインストールします。これらの属性は、デプロイされるアプリケーションを含め、現在のスタックの状態を表します。その後、レシピは必要なデータをノードオブジェクトから取得できます。

Note

アプリケーションで、スタック設定およびデプロイ属性値のようなノードオブジェクトからの情報が必要になることもあります。しかし、アプリケーションはノードオブジェクトにアクセスできません。ノードオブジェクトのデータをアプリケーションに渡すために、必要な情報をノードオブジェクトから抽出し、利用しやすい形式でファイルに追加するレシピを実装できます。そうすれば、アプリケーションはこのファイルからデータを読み取れるようになります。詳細と例については、「[アプリケーションへのデータの引き渡し](#)」を参照してください。

以下のように、レシピによってノードオブジェクトからスタック設定およびデプロイ属性値が取得されるようにできます。

- 属性の完全修飾名を使用して直接。

このアプローチは Linux スタックには使用できますが、Windows スタックには使用できません。

- ノードオブジェクトに対して属性値のクエリを実行できる Chef の検索で。

このアプローチは Windows スタックと Chef 11.10 Linux スタックに使用できます。

Note

Linux スタックでは、エージェントを使用して、インスタンスのスタック設定およびデプロイ属性のコピーを JSON形式で CLI 取得できます。詳細については、「[Vagrant でのスタック設定およびデプロイ属性の模倣](#)」を参照してください。

トピック

- [属性値の直接取得](#)
- [Chef の検索での属性値の取得](#)

属性値の直接取得

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このアプローチは Linux スタックにのみ使用できます。

「[Vagrant でのスタック設定およびデプロイ属性の模倣](#)」では、特定の属性を直接参照するためのノード構文を使用して、スタック設定およびデプロイ属性を取得する方法について説明しています。これが最良のアプローチである場合もあります。しかし、多くの属性は、内容や名前がスタックごとに違ったり、特定のスタックでも時間と共に変化したりするコレクションまたはリストで定義されています。例えば、`deploy` 属性には、アプリケーションの属性のリストが含まれており、アプリケーションの短縮名が付けられています。アプリケーションの属性名を含むこのリストは通常、スタックによっても、デプロイによっても異なります。

リストまたはコレクション内の属性を列挙して必要なデータを取得する方法が便利なが多く、この方法でなければならないこともあります。たとえば、スタックのインスタンスのパブリック IP ア

ドレスを知りたいとします。その情報は ['opsworks']['layers'] 属性にありますが、この属性はスタックのレイヤーごとにレイヤーの短い名前を持つ 1 個の要素を含んでいるハッシュテーブルに設定されています。各レイヤー要素が、レイヤーの属性を含むハッシュテーブルに設定されます。属性の 1 つが ['instances'] です。次に、その要素が、レイヤーのインスタンスごとにインスタンスの短い名前を持っている属性を含んでいる別のハッシュテーブルに設定されます。各インスタンス属性は、パブリック IP アドレスを表す ['ip'] などのインスタンス属性を含む、さらに別のハッシュテーブルに設定されます。これを視覚化できない場合は、次の手順に JSON 形式の例が含まれています。

この例では、スタックの JSON レイヤーのスタック設定とデプロイからデータを取得する方法を示します。

クックブックをセットアップするには

1. opsworks_cookbooks 内に listip という名前のディレクトリを作成し、そこに移動します。
2. 「[例 1: パッケージのインストール](#)」で説明されているとおりに Test Kitchen を初期化および設定します。
3. listip: recipes と environments にディレクトリを 2 つ追加します。
4. 関連する属性を含む MyStack 設定属性とデプロイ属性の編集 JSON 済みバージョンを作成します。次のようになります。

```
{
  "opsworks": {
    "layers": {
      "php-app": {
        "name": "PHP App Server",
        "id": "efd36017-ec42-4423-b655-53e4d3710652",
        "instances": {
          "php-app1": {
            "ip": "192.0.2.0"
          }
        }
      },
      "db-master": {
        "name": "MySQL",
        "id": "2d8e0b9a-0d29-43b7-8476-a9b2591a7251",
        "instances": {
          "db-master1": {
```

```
        "ip": "192.0.2.5"
      }
    }
  },
  "lb": {
    "name": "HAProxy",
    "id": "d5c4dda9-2888-4b22-b1ea-6d44c7841193",
    "instances": {
      "lb1": {
        "ip": "192.0.2.10"
      }
    }
  }
}
```

5. という名前の環境ファイルを作成し `test.json`、その例を JSON に貼り付けて `default_attributes`、クックブックの `environments` フォルダにファイルを保存します。ファイルは次のようになります (簡潔にするために、例のほとんど JSON は省略記号で表されます)。

```
{
  "default_attributes" : {
    "opsworks": {
      "layers": {
        ...
      }
    }
  },
  "chef_type" : "environment",
  "json_class" : "Chef::Environment"
}
```

6. `.kitchen.yml` のテキストを以下に置き換えます。

```
---
driver:
  name: vagrant

provisioner:
```

```
name: chef_zero
environments_path: ./environment

platforms:
  - name: ubuntu-12.04

suites:
  - name: listip
    provisioner:
      client_rb:
        environment: test
    run_list:
      - recipe[listip::default]
    attributes:
```

クックブックを設定したら、次のレシピを使用してレイヤー をログに記録できますIDs。

```
node['opsworks']['layers'].each do |layer, layerdata|
  log "#{layerdata['name']} : #{layerdata['id']}"
end
```

レシピは ['opsworks']['layers'] 内のレイヤーを列挙し、各レイヤーの名前と ID を記録します。

レシピを記録しながらレイヤーID を実行するには

1. 例のレシピで default.rb という名前のファイルを作成し、recipes ディレクトリに保存します。
2. kitchen converge を実行します。

出力の関連する部分は次のようになります。

```
Recipe: listip::default
 * log[PHP App Server : efd36017-ec42-4423-b655-53e4d3710652] action
write[2014-07-17T22:56:19+00:00] INFO: Processing log[PHP App Server : efd36017-ec42-4423-b655-53e4d3710652] action write (listip::default line 4)
```



```
[2014-07-17T22:56:19+00:00] INFO: PHP App Server : efd36017-ec42-4423-b655-53e4d3710652
```

```
* log[MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251] action
write[2014-07-17T22:56:19+00:00] INFO: Processing log[MySQL : 2d8e0b9a-0d29-43b7-8476-
a9b2591a7251] action write (listip::default line 4)
[2014-07-17T22:56:19+00:00] INFO: MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251
```

```
* log[HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193] action
write[2014-07-17T22:56:19+00:00] INFO: Processing log[HAProxy : d5c4dda9-2888-4b22-
b1ea-6d44c7841193] action write (listip::default line 4)
[2014-07-17T22:56:19+00:00] INFO: HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193
```

インスタンスの IP アドレスを一覧表示するには、次のような入れ子のループが必要です。

```
node['opsworks']['layers'].each do |layer, layerdata|
  log "#{layerdata['name']} : #{layerdata['id']}"
  layerdata['instances'].each do |instance, instancedata|
    log "Public IP: #{instancedata['ip']}"
  end
end
```

内側のループで各レイヤーのインスタンスを反復処理して、IP アドレスを記録します。

レシピを記録しながらインスタンス IP を実行するには

1. default.rb 内のコードを例のレシピに置き換えます。
2. kitchen converge を実行してレシピを実行します。

出力の関連する部分は次のようになります。

```
* log[PHP App Server : efd36017-ec42-4423-b655-53e4d3710652] action
write[2014-07-17T23:09:34+00:00] INFO: Processing log[PHP App Server : efd36017-
ec42-4423-b655-53e4d3710652] action write (listip::default line 2)
[2014-07-17T23:09:34+00:00] INFO: PHP App Server : efd36017-ec42-4423-b655-53e4d3710652
```

```
* log[Public IP: 192.0.2.0] action write[2014-07-17T23:09:34+00:00] INFO: Processing
log[Public IP: 192.0.2.0] action write (listip::default line 4)
[2014-07-17T23:09:34+00:00] INFO: Public IP: 192.0.2.0

* log[MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251] action
write[2014-07-17T23:09:34+00:00] INFO: Processing log[MySQL : 2d8e0b9a-0d29-43b7-8476-
a9b2591a7251] action write (listip::default line 2)
[2014-07-17T23:09:34+00:00] INFO: MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251

* log[Public IP: 192.0.2.5] action write[2014-07-17T23:09:34+00:00] INFO: Processing
log[Public IP: 192.0.2.5] action write (listip::default line 4)
[2014-07-17T23:09:34+00:00] INFO: Public IP: 192.0.2.5

* log[HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193] action
write[2014-07-17T23:09:34+00:00] INFO: Processing log[HAProxy : d5c4dda9-2888-4b22-
b1ea-6d44c7841193] action write (listip::default line 2)
[2014-07-17T23:09:34+00:00] INFO: HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193

* log[Public IP: 192.0.2.10] action write[2014-07-17T23:09:34+00:00] INFO: Processing
log[Public IP: 192.0.2.10] action write (listip::default line 4)
[2014-07-17T23:09:34+00:00] INFO: Public IP: 192.0.2.10
```

完了したら、`kitchen destroy` を実行します。次のトピックでは、新しいクックブックを使用します。

Note

スタック設定とデプロイJSONコレクションを列挙する最も一般的な理由の1つは、デプロイディレクトリなどの特定のデプロイされたアプリケーションのデータを取得することです。例については、[Deploy レシピ](#)を参照してください。

Chef の検索での属性値の取得

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソ

リューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このアプローチは Windows スタックと Chef 11.10 Linux スタックに使用できます。

スタック設定およびデプロイ属性値をノードオブジェクトから直接取得する操作は複雑になる場合があります。Windows スタックでは使用できません。別のアプローチは、[Chef の検索](#) を使用して目的の属性のクエリを実行することです。Chef サーバーに精通している場合は、Chef 検索が AWS OpsWorks スタックと少し異なることがわかります。AWS OpsWorks スタックはローカルモードで chef-client を使用するため、Chef 検索は chef-zero と呼ばれる Chef サーバーのローカルバージョンに依存するため、検索はリモートサーバーではなくインスタンスのノードオブジェクトにローカルに保存されているデータで動作します。

実際には、検索をローカルに保存されているデータに制限することは、AWS OpsWorks スタックインスタンスのノードオブジェクトに [スタック設定とデプロイ属性](#) が含まれているため、通常は重要ではありません。これには、レシピが通常 Chef サーバーから取得して同じ名前を使用するデータの大部分が含まれています。そのため、通常、スタックインスタンスで Chef AWS OpsWorks サーバー用に記述された検索コードを変更せずに使用できます。詳細については、「[Chef の検索の使用](#)」を参照してください。

以下に示しているのは、検索クエリの基本構造です。

```
result = search(:search_index, "key:pattern")
```

- 検索インデックスでは、クエリが適用される属性と、返されるオブジェクトのタイプを指定します。
- キーでは、属性名を指定します。
- パターンでは、取得する属性の値を指定します。

特定の属性値のクエリを実行したり、ワイルドカードを使用して属性値の範囲のクエリを実行したりできます。

- 結果として、クエリに一致するオブジェクトのリストが返されます。各オブジェクトは、複数の関連属性が保存されているハッシュテーブルです。

例えば、node の検索インデックスを使用した場合、クエリによってインスタンスオブジェクトのリストが返され、各オブジェクトはクエリに一致した各インスタンスに対応しています。各オブジェクトは、インスタンスの設定 (ホスト名や IP アドレスなど) を定義する属性のセットが保存されているハッシュテーブルです。

たとえば、以下のクエリでは node 検索インデックスを使用しています。これは、スタックのインスタンス (Chef の用語ではノード) に適用される標準の Chef インデックスです。そのクエリでは myhost のホスト名とインスタンスが検索されます。

```
result = search(:node, "hostname:myhost")
```

検索によって、ホスト名が myhost であるインスタンスオブジェクトのリストが返されます。たとえば、最初のインスタンスのオペレーティングシステムが必要な場合、クエリは `result[0][:os]` で表すことになります。クエリで複数のオブジェクトが返される場合は、それらのオブジェクトのリストを取得して、必要な情報を得ることができます。

レシピで検索を使用する方法の詳細は、使用しているのが Linux スタックか Windows スタックかによって異なります。以下のトピックでは、両方のスタックタイプの例を示しています。

トピック

- [Linux スタックでの検索の使用](#)
- [Windows スタックでの検索の使用](#)

Linux スタックでの検索の使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

この例は、単一のPHPアプリケーションサーバーを持つ Linux スタックに基づいています。Chef の検索を使用してサーバーのパブリック IP アドレスを取得し、アドレスを /tmp ディレクトリのファイルに保存します。基本的には「[属性値の直接取得](#)」と同じ情報をノードオブジェクトから取得しますが、コードは大幅に簡素になり、スタック設定およびデプロイ属性の構造の詳細に依存しません。

以下に、この例のスタックを作成する方法を簡単に示します。詳細については、「[新しいスタックを作成する](#)」を参照してください。

Note

AWS OpsWorks スタックインスタンスでカスタムレシピを実行したことがない場合は、まず[Linux インスタンスでのレシピの実行例](#)を実行する必要があります。

Stack を作成する

1. [AWS OpsWorks スタックコンソール](#)を開いて [Add Stack] をクリックします。
2. 次の設定を指定し、その他の設定はデフォルト値を受け入れて、[Add Stack] をクリックします。

- 名前 – 検索JSON
- デフォルトSSHキー – Amazon EC2キーペア

Amazon EC2キーペアを作成する必要がある場合は、「[Amazon EC2 キーペア](#)」を参照してください。キーペアは、インスタンスと同じAWSリージョンに属している必要があることに注意してください。この例では、米国西部 (オレゴン) リージョンを使用します。

3. レイヤーを追加をクリックし、デフォルト設定でスタックに[PHPアプリケーションサーバーレイヤーを追加します](#)。
4. デフォルト設定でレイヤーに [24/7 インスタンスを追加](#)し、[起動](#)します。

クックブックをセットアップするには

1. opsworks_cookbooks 内に searchjson という名前のディレクトリを作成し、そこに移動します。
2. 以下の内容で metadata.rb ファイルを作成し、opstest に保存します。

```
name "searchjson"
version "0.1.0"
```

3. recipes 内に searchjson ディレクトリを作成します。
4. 次のレシピで default.rb ファイルを作成し、recipes ディレクトリに保存します。

```
phpserver = search(:node, "layers:php-app").first
Chef::Log.info("*****The public IP address is: '#{phpserver[:ip]}'*")

file "/tmp/ip_addresses" do
  content "#{phpserver[:ip]}"
  mode 0644
  action :create
end
```

Linux スタックの場合、node 検索インデックスがサポートされています。レシピによって、このインデックスが使用されて、php-app レイヤーのインスタンスのリストが取得されます。レイヤーのインスタンスは 1 つのみであることがわかっているため、レシピによって最初のインスタンスが phpserver に渡されます。レイヤーに複数のインスタンスがある場合は、それらのインスタンスのリストを取得して、必要な情報を得ることができます。リストの各オブジェクトは、インスタンスの属性のセットが保存されているハッシュテーブルです。ip 属性をインスタンスのパブリック IP アドレスに設定しているため、以降のレシピコードではそのアドレスを phpserver[:ip] として参照できます。

Chef ログへのメッセージの追加後、レシピによって [file](#) リソースが使用されて、ip_addresses という名前のファイルが作成されます。content 属性は phpserver[:ip] の文字列表現で設定します。Chef によって ip_addresses が作成されるとき、その文字列がファイルに追加されます。

5. の .zip アーカイブを作成し opsworks_cookbooks、[アーカイブを Amazon S3 バケットにアップロードし、アーカイブを公開して、アーカイブの を記録します URL](#)。クックブックリポジトリの詳細については、「[クックブックリポジトリ](#)」を参照してください。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、「[S3 バケットを空にする方法](#)」または「[S3 バケットを削除する方法](#)」を参照してください。

これで、クックブックをインストールし、レシピを実行できるようになりました。

レシピを実行するには

1. [カスタムクックブックを有効にするようにスタックを編集](#)し、以下の設定を指定します。

- Repository type (リポジトリタイプ) – Http Archive (Http アーカイブ)
- リポジトリ URL – 前に記録URLしたクックブックアーカイブ

その他の設定にはデフォルト値を使用し、[Save] をクリックしてスタック設定を更新します。

2. カスタムレイヤー設定を編集し、レイヤーの Setup イベントに [を割り当て searchjson::default](#) ます。AWS OpsWorks スタックは、インスタンスの起動後、または Setup イベントを明示的にトリガーした場合にレシピを実行します。
3. [\[Update Custom Cookbooks\] スタックコマンドを実行](#) します。スタックのインスタンスにあるカスタムクックブックリポジトリの最新バージョンがインストールされます。以前のバージョンのリポジトリがある場合は、このコマンドによって上書きされます。
4. Setup スタックコマンドを実行してレシピを実行します。このコマンドによってインスタンスの Setup イベントがトリガーされ、searchjson::default が実行されます。[Running command setup page] を開いたままにしておきます。

レシピが正常に実行された後で、それを検証できます。

searchjson を検証するには

1. 最初のステップは、最新の Setup イベントの [Chef ログ](#) を調べることです。[Running command setup page] (コマンドセットアップを実行ページ) で、php-app1 インスタンスの [Log] (ログ) 列の [show] (表示) をクリックすると、ログが表示されます。下にスクロールすると、中央近くに次のようなログメッセージが表示されます。

```
...
[2014-09-05T17:08:41+00:00] WARN: Previous
  bash[logdir_existence_and_restart_apache2]: ...
[2014-09-05T17:08:41+00:00] WARN: Current
  bash[logdir_existence_and_restart_apache2]: ...
[2014-09-05T17:08:41+00:00] INFO: *****The public IP address is:
  '192.0.2.0'*****
```

```
[2014-09-05T17:08:41+00:00] INFO: Processing directory[/etc/sysctl.d] action create
(opsworks_initial_setup::sysctl line 1)
...
```

2. [SSH を使用してインスタンスにログイン](#)し、の内容を一覧表示します。これには/tmp、IP アドレスip_addressesを含む という名前のファイルが含まれている必要があります。

Windows スタックでの検索の使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックには、Windows スタックで検索を使用するための 2 つのオプションがあります。

- 標準の Chef の属性のセットのクエリを実行するために使用できる node 検索インデックス。
を使用する検索コードを持つ既存のレシピがある場合node、通常は変更なしで AWS OpsWorks スタックスタックで動作します。
- AWS OpsWorks スタックに固有の属性と一部の標準属性のセットのクエリに使用できる検索インデックスの追加セット。

これらのインデックスについては、「[Windows AWS OpsWorks スタックでのスタック固有の検索インデックスの使用](#)」で説明しています。

ホスト名や IP アドレスなど標準の情報を取得するには、node を使用することをお勧めします。そのアプローチにより、レシピは標準の Chef 手法に従うようになります。AWS OpsWorks スタック検索インデックスを使用して、AWS OpsWorks スタックに固有の情報を取得します。

トピック

- [Windows スタックでのノード検索インデックスの使用](#)
- [Windows AWS OpsWorks スタックでのスタック固有の検索インデックスの使用](#)

Windows スタックでのノード検索インデックスの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

この例では、「[Windows インスタンスでのレシピの実行](#)」の例を実行済みであることを前提としています。実行済みでない場合は、最初にその例を実行する必要があります。特に、インスタンスRDPへのアクセスを有効にする方法について説明します。

1 つのインスタンスが属するカスタムレイヤーから成る Windows スタックに基づいています。Chef の検索で node 検索インデックスを使用してサーバーのパブリック IP アドレスを取得し、アドレスを C:\tmp ディレクトリのファイルに保存します。以下に、この例のスタックを作成する方法を簡単に示します。詳細については、「[新しいスタックを作成する](#)」を参照してください。

Stack を作成する

1. [AWS OpsWorks スタックコンソール](#)を開いて [Add Stack] を選択します。
2. 以下の設定を指定し、その他の設定はデフォルト値を受け入れて、[Add Stack] を選択します。

- 名前 – NodeSearch
- リージョン – 米国西部 (オレゴン)

この例はいずれのリージョンでも動作しますが、チュートリアルでは米国西部 (オレゴン) を使用することをお勧めします。

- Default operating system (デフォルトのオペレーティングシステム) – Microsoft Windows Server 2012 R2
3. [Add a layer] を選択し、以下の設定を使用してスタックに[カスタムレイヤーを追加](#)します。
 - 名前 – IPTest

- Short name (短縮名) – iptest
4. デフォルト設定の [24/7 t2.micro インスタンス](#) をIPTestレイヤーに追加して [起動します](#)。そのインスタンスの名前は iptest1 になります。

AWS OpsWorks スタックは、このインスタンスに自動的に AWS-OpsWorks-RDP-Server を割り当てます。これにより、承認されたユーザーはインスタンスにログインできます。

5. アクセス許可を選択し、編集して、SSH/RDP と sudo/admin を選択します。通常のユーザーは、インスタンスにログインするために、AWS-OpsWorks-RDP-Server セキュリティグループに加えて、この権限が必要です。

Note

管理者としてログインすることもできますが、それには別の手順が必要です。詳細については、「[RDPでのログイン](#)」を参照してください。

クックブックをセットアップするには

1. nodesearch という名前のディレクトリを作成し、そのディレクトリに移動します。
2. 以下の内容で metadata.rb ファイルを作成し、opstest に保存します。

```
name "nodesearch"  
version "0.1.0"
```

3. recipes 内に nodesearch ディレクトリを作成します。
4. 次のレシピで default.rb ファイルを作成し、recipes ディレクトリに保存します。

```
directory 'C:\tmp' do  
  rights :full_control, 'Everyone'  
  recursive true  
  action :create  
end  
  
windowsserver = search(:node, "hostname:iptest*").first  
Chef::Log.info("*****The public IP address is:  
'#{windowsserver[:ipaddress]}'*****")
```

```
file 'C:\tmp\addresses.txt' do
  content "#{windowsserver[:ipaddress]}"
  rights :full_control, 'Everyone'
  action :create
end
```

このレシピでは、以下のような処理を実行します。

1. ディレクトリリソースを使用して、ファイルの C:\tmp ディレクトリを作成します。

このリソースの詳細については、「[例 3: ディレクトリの作成](#)」を参照してください。

2. Chef の検索で node 検索インデックスを使用して、ノード (インスタンス) と iptest で始まるホスト名のリストを取得します。

レイヤーの短縮名に整数を追加してホスト名を作成するデフォルトのテーマを使用すると、このクエリはIPTestレイヤー内のすべてのインスタンスを返します。この例では、レイヤーのインスタンスは 1 つのみであることがわかっているため、レシピによって最初のインスタンスが windowsserver に渡されます。複数のインスタンスがある場合は、それらのインスタンスのリストを取得できます。

3. IP アドレスを含むメッセージをこの実行のために Chef のログに追加します。

windowsserver オブジェクトはハッシュテーブルであるため、その ipaddress 属性がインスタンスのパブリック IP アドレスに設定されているため、以降のレシピコードでそのアドレスを windowsserver[:ipaddress] として参照できます。レシピによって、対応する文字列がメッセージに挿入され、Chef ログに追加されます。

4. file のリソースを使用して、その IP アドレスを含むファイルを C:\tmp\addresses.txt という名前で作成します。

リソースの content 属性には、ファイルに追加する内容 (この場合はパブリック IP アドレス) を指定します。

5. の .zip アーカイブを作成し nodesearch、[アーカイブを S3 バケットにアップロード](#)し、[アーカイブを公開](#)して、アーカイブの `URL` を記録します。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、「[S3 バケットを空にする方法](#)」または「[S3 バケットを削除する方法](#)」を参照してください。

これで、クックブックをインストールし、レシピを実行できるようになりました。

クックブックをインストールし、レシピを実行するには

1. [カスタムクックブックを有効にするようにスタックを編集](#)し、以下の設定を指定します。

- Repository type (リポジトリタイプ) – S3 Archive (アーカイブ)
- リポジトリ URL – 前に記録URLしたクックブックアーカイブ

その他の設定ではデフォルト値を受け入れ、[Save] を選択してスタック設定を更新します。

2. [\[Update Custom Cookbooks\] スタックコマンドを実行](#)します。カスタムクックブックの最新バージョンがスタックのインスタンス (オンラインインスタンスを含む) にインストールされます。以前のバージョンのクックブックがある場合は、このコマンドによって上書きされます。
3. カスタムクックブックの更新が終了したら、実行するレシピを **nodesearch::default** に設定して[レシピの実行スタックコマンド](#)を実行することで、レシピを実行します。このコマンドによって Chef 実行が開始され、レシピで構成される実行リストが渡されます。execute_recipes ページを開いたままにします。

レシピが正常に実行された後で、それを検証できます。

nodesearch を検証するには

1. 最新の execute_recipes イベントについて [Chef ログ](#)を調べます。[Running command execute_recipes page] (コマンドexecute_recipes実行ページ) で、iptest1 インスタンスの [Log] (ログ) 列の [show] (表示) を選択すると、ログが表示されます。下にスクロールすると、末尾近くに以下のようなログメッセージが見つかります。

```
...
[2015-05-13T18:55:47+00:00] INFO: Storing updated cookbooks/nodesearch/recipes/default.rb in the cache.
[2015-05-13T18:55:47+00:00] INFO: Storing updated cookbooks/nodesearch/metadata.rb in the cache.
[2015-05-13T18:55:47+00:00] INFO: *****The public IP address is:
'192.0.0.1'*****
[2015-05-13T18:55:47+00:00] INFO: Processing directory[C:\tmp] action create (nodesearch::default line 1)
[2015-05-13T18:55:47+00:00] INFO: Processing file[C:\tmp\addresses.txt] action create (nodesearch::default line 10)
...
```

2. [RDP を使用してインスタンスにログイン](#)し、の内容を確認しますC:\tmp\addresses.txt。

Windows AWS OpsWorks スタックでのスタック固有の検索インデックスの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

この例では、「[Windows インスタンスでのレシピの実行](#)」の例を実行済みであることを前提としています。実行済みでない場合は、最初にその例を実行する必要があります。特に、インスタンスRDPへのアクセスを有効にする方法について説明します。

AWS OpsWorks スタックには、に加えて次の検索インデックスが用意されていますnode。

- aws_opsworks_stack – スタック設定。
- aws_opsworks_layer – スタックのレイヤー設定。
- aws_opsworks_instance – スタックのインスタンス設定。
- aws_opsworks_app – スタックのアプリケーション設定。
- aws_opsworks_user – スタックのユーザー設定。
- aws_opsworks_rds_db_instance – 登録されたRDSインスタンスの接続情報。

これらのインデックスには標準の Chef 属性が含まれていますが、主に AWS OpsWorks スタック固有の属性を取得することを目的としています。例えば、aws_opsworks_instance には、status など、インスタンスのステータスを渡す online 属性が含まれます。

Note

可能であれば `node` を使用して、レシピが標準の Chef 手法に従うようにすることをお勧めします。例については、[Windows スタックでのノード検索インデックスの使用](#)を参照してください。

この例では、スタックインデックスを使用して AWS OpsWorks AWS OpsWorks スタック固有の属性の値を取得する方法を示します。この例は、1つのインスタンスが属するカスタムレイヤーから成る Windows スタックに基づいています。Chef 検索を使用してインスタンスの AWS OpsWorks スタック ID を取得し、その結果を Chef ログに配置します。

以下に、この例のスタックを作成する方法を簡単に示します。詳細については、「[新しいスタックを作成する](#)」を参照してください。

Stack を作成する

1. [AWS OpsWorks スタックコンソール](#)を開いて [+ Stack] を選択します。以下の設定を指定し、その他の設定はデフォルト値を受け入れて、[Add Stack] を選択します。

- 名前 – IDSearch
- リージョン – 米国西部 (オレゴン)

この例はいずれのリージョンでも動作しますが、チュートリアルでは米国西部 (オレゴン) を使用することをお勧めします。

- Default operating system (デフォルトのオペレーティングシステム) – Microsoft Windows Server 2012 R2
2. [Add a layer] を選択し、以下の設定を使用してスタックに[カスタムレイヤーを追加](#)します。
 - 名前 – IDCheck
 - Short name (短縮名) – idcheck
 3. デフォルト設定の [24/7 t2.micro インスタンス](#)をIDCheckレイヤーに追加して[起動](#)します。そのインスタンスの名前は `iptest1` になります。

AWS OpsWorks スタックは、このインスタンスに自動的に `AWS-OpsWorks-RDP-Server` を割り当てます。[RDP アクセスの有効化](#) は、許可されたユーザーがインスタンスにログインできるようにするインバウンドルールをこのセキュリティグループに追加する方法について説明します。

4. アクセス許可を選択し、編集して、SSH/RDP と sudo/admin を選択します。通常のユーザーは、インスタンスにログインするために、AWS-OpsWorks-RDP-Server セキュリティグループに加えて、この権限が必要です。

Note

管理者としてログインすることもできますが、それには別の手順が必要です。詳細については、「[RDP でのログイン](#)」を参照してください。

クックブックをセットアップするには

1. idcheck という名前のディレクトリを作成し、そのディレクトリに移動します。
2. 以下の内容で metadata.rb ファイルを作成し、opstest に保存します。

```
name "idcheck"
version "0.1.0"
```

3. recipes 内に idcheck ディレクトリを作成し、以下のレシピを保存した default.rb ファイルをそのディレクトリに追加します。

```
windowserver = search(:aws_opsworks_instance, "hostname:idcheck*").first
Chef::Log.info("*****The public IP address is:
 '#{windowserver[:instance_id]}*****")
```

このレシピでは、aws_opsworks_instance 検索インデックスを持つChef検索を使用して、スタック内のホスト名が idcheck で始まる各インスタンスの [\[instance attributes](#) (インスタンス属性)] を取得します。レイヤーの短縮名に整数を追加してホスト名を作成するデフォルトのテーマを使用すると、このクエリはIDCheckレイヤー内のすべてのインスタンスを返します。この例では、レイヤーのインスタンスは 1 つのみであることがわかっているため、レシピによって最初のインスタンスが windowserver に渡されます。複数のインスタンスがある場合は、それらのインスタンスのリストを取得できます。

レシピでは、スタックにこのホスト名のインスタンスが 1 つしかないことがわかっているため、最初の結果が正しくなります。スタックに複数のインスタンスがある場合は、他の属性で検索すると、複数の結果が返されることがあります。インスタンスの属性のリストについては、「[インスタンスデータバグ \(aws_opsworks_instance\)](#)」を参照してください。

インスタンス属性は基本的にハッシュテーブルであり、インスタンスの AWS OpsWorks スタック ID は `instance_id` 属性に割り当てられるため、ID をと参照できます `windowserver[:instance_id]`。レシピによって、対応する文字列がメッセージに挿入され、Chef ログに追加されます。

4. `ipaddress` クックブックの `.zip` アーカイブを作成し、[アーカイブを Amazon S3 バケットにアップロード](#)して、アーカイブの `URL` を記録します。クックブックリポジトリの詳細については、「[クックブックリポジトリ](#)」を参照してください。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、「[S3 バケットを空にする方法](#)」または「[S3 バケットを削除する方法](#)」を参照してください。

これで、クックブックをインストールし、レシピを実行できるようになりました。

クックブックをインストールし、レシピを実行するには

1. [カスタムクックブックを有効にするようにスタックを編集](#)し、以下の設定を指定します。
 - Repository type (リポジトリタイプ) – S3 Archive (アーカイブ)
 - リポジトリ URL – 前に記録URLしたクックブックアーカイブ

その他の設定ではデフォルト値を受け入れ、[Save] を選択してスタック設定を更新します。

2. [\[Update Custom Cookbooks\] スタックコマンドを実行](#)します。カスタムクックブックの最新バージョンがスタックのインスタンス (オンラインインスタンスを含む) にインストールされます。以前のバージョンのクックブックがある場合は、このコマンドによって上書きされます。
3. カスタムクックブックの更新が終了したら、実行するレシピを `idcheck::default` に設定して [レシピの実行スタックコマンド](#) を実行することで、レシピを実行します。このコマンドによって Chef 実行が開始され、レシピで構成される実行リストが渡されます。execute_recipes ページを開いたままにします。

レシピが正常に実行された後、最新の execute_recipes イベントの [Chef ログ](#) を調べることで、その結果を確認できます。[Running command execute_recipes page] (コマンド execute_recipes 実行ページ) で、iptest1 インスタンスの [Log] (ログ) 列の [show] (表示) を選択すると、ログが表示されます。下にスクロールすると、末尾近くに以下のようなログメッセージが見つかります。


```
...
[2015-05-13T20:03:47+00:00] INFO: Storing updated cookbooks/nodesearch/recipes/
default.rb in the cache.
[2015-05-13T20:03:47+00:00] INFO: Storing updated cookbooks/nodesearch/metadata.rb in
the cache.
[2015-05-13T20:03:47+00:00] INFO: *****The instance ID is: 'i-8703b570'*****
[2015-05-13T20:03:47+00:00] INFO: Chef Run complete in 0.312518 seconds
...
```

Linux インスタンスでの外部クックブック Berkshelf の使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

Berkshelf は Chef 11.10 Linux スタックにのみ使用できます。

クックブックを実装する前に、[Chef Community Cookbooks](#) を確認してください。ここでは、Chef コミュニティのメンバーによってさまざまな目的で作成されたクックブックがあります。これらのクックブックの多くは、変更せずに AWS OpsWorks スタックで使用できるため、すべてのコードを自分で実装するのではなく、一部のタスクで活用できる場合があります。

インスタンスで外部クックブックを使用するには、それをインストールし、依存関係を管理する方法が必要です。最適なアプローチは、依存関係マネージャをサポートする、Berkshelf という名前のクックブックを実装することです。Berkshelf は スタック EC2 インスタンスを含む Amazon AWS OpsWorks インスタンスで動作しますが、Test Kitchen や Vagrant でも動作するように設計されています。ただし、Vagrant での使用は AWS OpsWorks スタックの使用と少し異なるため、このトピックには両方のプラットフォームの例が含まれています。Berkshelf の使用方法の詳細については、「[Berkshelf](#)」を参照してください。

トピック

- [Test Kitchen および Vagrant での Berkshelf の使用](#)
- [スタックでの Berkshelf AWS OpsWorks の使用](#)

Test Kitchen および Vagrant での Berkshelf の使用

この例では、Berkshelf を使用して getting-started コミュニティクックブックをインストールし、そのレシピを実行する方法を示します。このレシピは、短いテキストファイルをインスタンスのホームディレクトリにインストールします。

Berkshelf をインストールし、クックブックを初期化するには

1. ワークステーションで、Berkshelf gem を次のようにインストールします。

```
gem install berkshelf
```

ワークステーションによっては、このコマンドに `sudo`、などの Ruby 環境マネージャーを使用することもできます[RVM](#)。Berkshelf が正常にインストールされたことを確認するには、`berks --version` を実行します。

2. このトピックのクックブックは、`external_cookbook` という名前です。以前のトピックで採用した手動のアプローチの代わりに、Berkshelf を使用して、初期化されたクックブックを作成できます。そのためには、`opsworks_cookbooks` ディレクトリに移動して、次のコマンドを実行します。

```
berks cookbook external_cookbook
```

このコマンドによって、`external_cookbook` ディレクトリといくつかの標準的な Chef および Test Kitchen サブディレクトリ (`recipes`、`test` など) が作成されます。さらに、次のような多数の標準ファイルのデフォルトバージョンも作成されます。

- `metadata.rb`
- Vagrant、Test Kitchen、および Berkshelf の設定ファイル
- `default.rb` ディレクトリにある空の `recipes` レシピ

Note

`kitchen init` を実行する必要はありません。それらのタスクは `berks cookbook` コマンドによって処理されます。

3. `kitchen converge` を実行します。新しく作成したクックブックは、この時点では興味深いことを何もしませんが、集中を行います。

Note

また、`berks init` を使用して、`Berkshelf` を使用するように既存のクックブックを初期化できます。

`Berkshelf` を使用してクックブックの外部依存関係を管理するには、クックブックのルートディレクトリに `Berksfile` が含まれている必要があります。これは、`Berkshelf` がどのように依存関係を管理するかを指定する設定ファイルです。`berks cookbook` クックブックを作成するために `external_cookbook` を使用した場合、以下の内容の `Berksfile` が作成されます。

```
source "https://supermarket.chef.io"  
metadata
```

このファイルには、以下の宣言があります。

- `source` – クックブックソースURLの。

`Berksfile` は、任意の数の `source` 宣言を持つことができます。各宣言は、依存クックブックのデフォルトのソースを指定します。クックブックのソースを明示的に指定しない場合、`Berkshelf` はデフォルトのリポジトリで同じ名前のクックブックを探します。デフォルト `Berksfile` には、コミュニティクックブックリポジトリを指定する単一の `source` 属性が含まれています。そのリポジトリには `getting-started` クックブックが含まれているので、行を変更する必要はありません。

- `metadata` – クックブックの `metadata.rb` ファイルで宣言されているクックブックの依存関係を取り込むように `Berkshelf` に指示します。

また、後で説明するように、cookbook 属性を取り込むことによって、Berksfile で依存クックブックを宣言できます。

クックブックの依存関係を宣言するには、次の 2 つの方法があります。

- cookbook 宣言を Berksfile に含めます。

これは、AWS OpsWorks スタックで使用されるアプローチです。たとえば、この例で使用されている getting-started クックブックを指定するには、Berksfile に cookbook "getting-started" を含めます。そうすると、Berkshelf はデフォルトリポジトリでその名前のクックブックを探します。また、cookbook を使用して、明示的にクックブックソースと特定のバージョンを指定することもできます。詳細については「[Berkshelf](#)」を参照してください。

- metadata 宣言を Berksfile に含め、依存関係を metadata.rb で宣言します。

この宣言は、metadata.rb で宣言されているクックブックの依存関係を取り込むように Berkshelf に指示します。たとえば、getting-started の依存関係を宣言するには、クックブックの depends 'getting-started' ファイルに metadata.rb 宣言を追加します。

この例では、AWS OpsWorks スタックとの一貫性を保つために、最初のアプローチを使用します。

getting-started クックブックをインストールするには

1. デフォルト Berksfile を編集して、metadata 宣言を cookbook の getting-started 宣言に置き換えます。内容は次のようになります。

```
source "https://supermarket.chef.io"

cookbook 'getting-started'
```

2. `berks install` を実行すると、コミュニティクックブックリポジトリからワークステーションの Berkshelf ディレクトリ (通常は `~/.berkshelf`) に getting-started クックブックがダウンロードされます。このディレクトリは、多くの場合、単に Berkshelf と呼ばれます。Berkshelf の cookbooks ディレクトリを見ると、クックブックのディレクトリ (getting-started-0.4.0getting-started- のような名前のディレクトリ) が表示されるはずです。
3. `external_cookbook::default` 実行リスト内の `.kitchen.yml` を `getting-started::default` に置き換えます。この例は external_cookbook のどのレシピも実

行しません。基本的には、単に getting-started クックブックを使用するための手段です。 .kitchen.yml ファイルは次のようになっているはずですが。

```
---
driver:
  name: vagrant

provisioner:
  name: chef_solo

platforms:
  - name: ubuntu-12.04

suites:
  - name: default
    run_list:
      - recipe[getting-started::default]
    attributes:
```

4. `kitchen converge` を実行してから `kitchen login` を使用してインスタンスにログインします。ログインディレクトリには、次のような内容の `chef-getting-started.txt` という名前のファイルが含まれているはずですが。

```
Welcome to Chef!

This is Chef version 11.12.8.
Running on ubuntu.
Version 12.04.
```

Test Kitchen によって、インスタンスの `/tmp/kitchen/cookbooks` ディレクトリ内のクックブックがインストールされます。このディレクトリの内容を一覧表示すると、`external_cookbook` と `getting-started` という2つのクックブックが表示されます。

5. `kitchen destroy` を実行して、インスタンスをシャットダウンします。次の例では、AWS OpsWorks スタックインスタンスを使用します。

スタックでの Berkshelf AWS OpsWorks の使用

AWS OpsWorks スタックは、オプションで Chef 11.10 スタックの Berkshelf をサポートします。スタックで Berkshelf を使用するには、以下の操作を実行する必要があります。

- スタックの Berkshelf を有効にします。

AWS OpsWorks その後、スタックはスタックのインスタンスに Berkshelf をインストールする詳細を処理します。

- クックブックリポジトリのルートディレクトリに Berkfile を追加します。

Berkfile には、すべての依存クックブックの source および cookbook 宣言が含まれている必要があります。

AWS OpsWorks スタックがインスタンスにカスタムクックブックリポジトリをインストールすると、Berkshelf を使用して、リポジトリの Berkfile で宣言されている依存クックブックをインストールします。詳細については、「[Berkshelf の使用](#)」を参照してください。

この例では、Berkshelf を使用してスタックインスタンスに getting-started AWS OpsWorks コミュニティクックブックをインストールする方法を示します。また、指定したディレクトリ内にファイルを作成する、createfile カスタムクックブックのバージョンもインストールします。createfile の動作の詳細については、「[クックブックからのファイルのインストール](#)」を参照してください。

Note

AWS OpsWorks スタックスタックにカスタムクックブックを初めてインストールする場合は、まず[Linux インスタンスでのレシピの実行例](#)を実行する必要があります。

以下にまとめたように、スタックの作成から始めます。詳細については、「[新しいスタックを作成する](#)」を参照してください。

Stack を作成する

1. [AWS OpsWorks スタックコンソール](#)を開いて [Add Stack] をクリックします。
2. 次の設定を指定し、その他の設定はデフォルト値を受け入れて、[Add Stack] をクリックします。

- 名前 – BerksTest

- デフォルトSSHキー – Amazon EC2キーペア

Amazon EC2キーペアを作成する必要がある場合は、[「Amazon EC2 キーペア」](#)を参照してください。キーペアは、インスタンスと同じAWSリージョンに属している必要があることに注意してください。この例では、デフォルトに米国西部(オレゴン) リージョンを使用します。

3. [Add a layer] をクリックし、次の設定を使用してスタックに[カスタムレイヤーを追加](#)します。

- 名前 – BerksTest
- Short name (短縮名) – berkstest

この例では、実際には任意のレイヤータイプを使用できます。しかし、この例では他のレイヤーによってインストールされるパッケージは必要ないので、カスタムレイヤーが最もシンプルなアプローチです。

4. デフォルト設定で [24/7 インスタンス](#) を BerksTest レイヤーに追加しますが、まだ起動しないでください。

AWS OpsWorks スタックでは、クックブックは標準のディレクトリ構造を持つリモートリポジトリにある必要があります。次に、スタックにダウンロード情報を提供します AWS OpsWorks。これにより、起動時にスタックの各インスタンスにリポジトリが自動的にダウンロードされます。わかりやすくするために、この例のリポジトリはパブリック Amazon S3 アーカイブですが、AWS OpsWorks スタックではHTTPアーカイブ、Git リポジトリ、および Subversion リポジトリもサポートされています。詳細については、[「クックブックリポジトリ」](#)を参照してください。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、[「S3 バケットを空にする方法」](#)または [「S3 バケットを削除する方法」](#)を参照してください。

クックブックリポジトリを作成するには

1. opsworks_cookbooks ディレクトリに berkstest_cookbooks という名前のディレクトリを作成します。このディレクトリは、リポジトリにアップロードするため、都合のよい任意の場所に作成することもできます。
2. 以下の内容で、Berksfile という名前のファイルを berkstest_cookbooks に追加します。

```
source "https://supermarket.chef.io"
```

```
cookbook 'getting-started'
```

このファイルは、getting-started クックブックの依存関係を宣言し、それをコミュニティクックブックサイトからダウンロードするように Berkshelf に指示します。

3. 以下を含む createfile に berkstest_cookbooks ディレクトリを追加します。
 - 以下の内容の metadata.rb ファイル。

```
name "createfile"
version "0.1.0"
```

- 以下のコンテンツを持つ files/default ファイルが含まれている example_data.json ディレクトリ。

```
{
  "my_name" : "myname",
  "your_name" : "yourname",
  "a_number" : 42,
  "a_boolean" : true
}
```

ファイルの名前および内容は任意です。レシピは単に、指定された場所にファイルをコピーします。

- 以下のレシピコードを持つ recipes ファイルが含まれている default.rb ディレクトリ。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end

cookbook_file "/srv/www/shared/example_data.json" do
  source "example_data.json"
  mode 0644
  action :create_if_missing
end
```



```
end
```

このレシピは、`/srv/www/shared` を作成し、クックブックの `example_data.json` ディレクトリからそのディレクトリに `files` をコピーします。

4. の `.zip` アーカイブを作成し `berkstest_cookbooks`、[アーカイブを Amazon S3 バケットにアップロード](#)し、[アーカイブを公開](#)して、アーカイブの `URL` を記録します。

これで、クックブックをインストールし、レシピを実行できるようになりました。

クックブックをインストールし、レシピを実行するには

1. [カスタムクックブックを有効にするようにスタックを編集](#)し、以下の設定を指定します。

- Repository type (リポジトリタイプ) – Http Archive (Http アーカイブ)
- リポジトリ URL – 前に記録URLしたクックブックアーカイブ
- Manage Berkshelf (Berkshelf の管理) – Yes (はい)

最初の 2 つの設定は AWS OpsWorks、インスタンスにクックブックリポジトリをダウンロードするために必要な情報をスタックに提供します。最後の設定は、Berkshelf サポートを有効にします。これによって、`getting-started` クックブックがインスタンスにダウンロードされます。その他の設定ではデフォルト値を受け入れ、[Save] をクリックしてスタック設定を更新します。

2. BerksTest レイヤーを編集して、[レイヤーの Setup ライフサイクルイベントに次のレシピを追加](#)します。

- `getting-started::default`
- `createfile::default`

3. インスタンスを[起動](#)します。Setup イベントは、インスタンスの起動が完了した後に発生します。AWS OpsWorks スタックはクックブックリポジトリをインストールし、Berkshelf を使用して `getting-started` クックブックをダウンロードし、`getting-started::default` やなどのレイヤーのセットアップレシピとデプロイレシピを実行します `createfile::default`。
4. インスタンスがオンラインになったら、[SSH を使用してログイン](#)します。次のように表示されます。

- `/srv/www/shared` には `example_data.json` が含まれている必要があります。
- `/root` には `chef-getting-started.txt` が含まれている必要があります。

AWS OpsWorks スタックはレシピをルートとして実行するため、getting-started はホーム/ rootディレクトリではなく ディレクトリに ファイルをインストールします。

SDK for Ruby の使用: Amazon S3 からファイルをダウンロードする

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Chef リソースでは処理できない AWSサービスとのやり取りなどのタスクがいくつかあります。たとえば、ファイルをリモートで保存し、レシピによってインスタンスにダウンロードすることが望ましい場合があります。その場合は、[remote_file](#) リソースを使用して、リモートサーバーからファイルをダウンロードできます。ただし、ファイルを [Amazon S3 バケット](#) に保存する場合、オペレーションが [ACL](#) 許可されている場合にのみ、remote_file はそれらのファイルをダウンロードできません。

レシピは [AWS SDK for Ruby](#) を使用してほとんどの AWSサービスにアクセスできます。このトピックでは、SDK for Ruby を使用して S3 バケットからファイルをダウンロードする方法について説明します。

Note

を使用して暗号化と復号 [AWS SDK for Ruby](#) を処理する方法の詳細については、[AWS 「::S3::S3Object」](#) を参照してください。Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、[「S3 バケットを空にする方法」](#) または [「S3 バケットを削除する方法」](#) を参照してください。

トピック

- [Vagrant インスタンスでの SDK Ruby 用の使用](#)

- [スタックの Linux インスタンスでの Ruby AWS OpsWorks SDK用 の使用](#)
- [スタックの Windows インスタンスでの Ruby AWS OpsWorks SDK用 の使用](#)

Vagrant インスタンスでの SDK Ruby 用 の使用

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

このトピックでは、Vagrant インスタンスで実行されているレシピで [AWS SDK for Ruby](#) を使用して Amazon S3 からファイルをダウンロードする方法を説明します。開始する前に、レシピが Amazon S3 にアクセスできるようにするアクセスキーとシークレットアクセスキーという AWS 認証情報のセットが必要です。

⚠ Important

この目的でルートアカウントの認証情報を使わないことを強くお勧めします。代わりに、適切なポリシーでユーザーを作成し、レシピにこれらの認証情報を提供します。

認証情報を含むファイルを IAMパブリックリポジトリ GitHub や Bitbucket リポジトリにアップロードするなど、パブリックにアクセス可能な場所に認証情報を含めないように注意してください。ご自分の認証情報が公開され、アカウントのセキュリティが侵害される恐れがあります。

EC2Amazon EC2 インスタンスで実行されているレシピは、「」で説明されているように、さらに優れたアプローチである IAMロールを使用できます [スタックの Linux インスタンスでの Ruby AWS OpsWorks SDK用 の使用](#)。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、「[S3 バケットを空にする方法](#)」または「[S3 バケットを削除する方法](#)」を参照してください。

まだ適切なユーザーが存在しない場合は、次のようにして作成できます。詳細については、「[IAM とは](#)」を参照してください。

⚠ Warning

IAM ユーザーは長期的な認証情報を持っているため、セキュリティ上のリスクがあります。このリスクを軽減するために、これらのユーザーにはタスクの実行に必要な権限のみを付与し、不要になったユーザーを削除することをお勧めします。

IAM ユーザーを作成するには

1. にサインイン AWS Management Console し、 で IAMコンソールを開きます <https://console.aws.amazon.com/iam/>。
2. ナビゲーションペインで **ユーザー** を選択し、必要に応じて **ユーザーの追加** を選択して、新しい管理ユーザーを作成します。
3. 許可を設定 ページで、**ポリシーを直接アタッチする** を選択します。
4. 許可ポリシー の検索ボックスに **S3** と入力すると、Amazon S3 のポリシーが表示されます。

AmazonS3ReadOnlyAccess を選択します。必要に応じて、AmazonS3FullAccess などのより広範なアクセス許可を付与するポリシーを指定できますが、標準的な方法は、必要なアクセス許可のみを付与することです。この場合、レシピはファイルをダウンロードするだけなので、読み取り専用アクセスで十分です。

5. [Next (次へ)] を選択します。
6. **ユーザーの作成** を選択します。
7. 次に、ユーザーのアクセスキーを作成します。アクセスキーの作成の詳細については、「IAM ユーザーガイド」の [IAM「ユーザーのアクセスキーの管理」](#) を参照してください。

次に、ダウンロードされるファイルを用意する必要があります。この例では、新たに作成した `myfile.txt` という名前の S3 バケットに `cookbook_bucket` という名前のファイルが保存してあることを前提としています。

ダウンロード用ファイルを提供するには

1. 以下のテキストを持つ `myfile.txt` という名前のファイルを作成し、ワークステーションの便利な場所に保存します。

```
This is the file that you just downloaded from Amazon S3.
```

2. [Amazon S3 console](#) (Amazon S3 コンソール) で、[Standard] (標準) リージョンに `cookbook_bucket` という名前のバケットを作成し、バケットに `myfile.txt` をアップロードします。

クックブックを以下のようにセットアップします。

クックブックをセットアップするには

1. `opsworks_cookbooks` 内に `s3bucket` という名前のディレクトリを作成し、そこに移動します。
2. 「[例 1: パッケージのインストール](#)」で説明されているとおりに Test Kitchen を初期化および設定します。
3. `.kitchen.yml` のテキストを以下に置き換えます。

```
---
driver:
  name: vagrant

provisioner:
  name: chef_solo
  environments_path: ./environments

platforms:
  - name: ubuntu-14.04

suites:
  - name: s3bucket
    provisioner:
      solo_rb:
        environment: test
    run_list:
      - recipe[s3bucket::default]
    attributes:
```

4. `s3bucket: recipes` と `environments` にディレクトリを 2 つ追加します。
5. 以下の `default_attributes` セクションを持つ `test.json` という名前の環境ファイルを作成し、`access_key` および `secret_key` の値をユーザーの対応するキーに置き換えます。クックブックの `environments` フォルダにファイルを保存します。

```
{
  "default_attributes" : {
    "cookbooks_101" : {
      "access_key": "AKIAIOSFODNN7EXAMPLE",
      "secret_key" : "wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY"
    }
  },
  "chef_type" : "environment",
  "json_class" : "Chef::Environment"
}
```

インスタンスで実行されているレシピに認証情報を提供するには、さまざまな方法があります。重要な注意点は、誤ってキーを公開し、アカウントのセキュリティが侵害される可能性を減らす必要があることです。そのため、コードで明示的なキー値を使用することはお勧めできません。例では、代わりにキー値をノードオブジェクトに保存します。その場合、レシピはリテラル値を公開せずに、ノード構文を使用してキー値を参照できます。ノードオブジェクトにアクセスするには、ルート権限が必要です。そのため、キーが公開される可能性を制限できます。詳細については、[AWS「アクセスキーを管理するためのベストプラクティス」](#)を参照してください。

Note

例では入れ子になった属性 (最初の要素は cookbooks_101) が使用されていることに注意してください。これにより、ノードオブジェクト内に他の access_key または secret_key 属性がある場合に、名前の競合の可能性が制限されます。

次のレシピは、myfile.text バケットから cookbook_bucket をダウンロードします。

```
gem_package "aws-sdk ~> 3" do
  action :install
end

ruby_block "download-object" do
  block do
    require 'aws-sdk'

    s3 = Aws::S3::Client.new(
```

```
      :access_key_id => "#{node['cookbooks_101']['access_key']}",
      :secret_access_key => "#{node['cookbooks_101']['secret_key']}")

myfile = s3.bucket['cookbook_bucket'].objects['myfile.txt']
Dir.chdir("/tmp")
File.open("myfile.txt", "w") do |f|
  f.write(myfile.read)
  f.close
end
end
action :run
end
```

レシピの最初の部分では、Gem パッケージである SDK for Ruby をインストールします。[gem_package](#) リソースは、レシピや他のアプリケーションによって使用される gem をインストールします。

Note

インスタンスには 2 つの Ruby インスタンスがあり、このインスタンスは通常バージョンが異なります。1 つは Chef Client によって使用される専用インスタンスです。もう 1 つは、インスタンスで実行されているアプリケーションとレシピによって使用されます。gem パッケージをインストールする際に、この相違点を理解しておくことが重要です。gem をインストールするリソースには、[gem_package](#) と [chef_gem](#) という 2 つの種類があるためです。アプリケーションまたはレシピが gem パッケージを使用する場合には、gem_package でインストールします。chef_gem は、Chef クライアントが使用する gem パッケージに限り使用します。

レシピの残りの部分は、[ruby_block](#) リソースです。これには、ファイルをダウンロードする Ruby コードが含まれています。レシピは Ruby アプリケーションであるため、コードをレシピに直接記述できると考えられがちです。しかし、Chef 実行はリソースを実行する前にすべてのコードをコンパイルします。例のコードをレシピに直接記述すると、Ruby は require 'aws-sdk' リソースを実行する前に gem_package ステートメントを解決しようとし、SDK for Ruby はまだインストールされていないため、コンパイルは失敗します。

ruby_block リソースのコードは、そのリソースが実行されるまでコンパイルされません。この例では、ruby_block リソースが Ruby SDK用のインストールを完了した後に gem_package リソースが実行されるため、コードは正常に実行されます。

ruby_block 内のコードは、次のように動作します。

1. サービスインターフェイスを提供する、新しい [Aws::S3](#) オブジェクトを作成します。

アクセスキーとシークレットキーは、ノードオブジェクトに保存されている値を参照することによって指定されます。

2. S3 オブジェクトの `bucket.objects` アソシエーションを呼び出します。このアソシエーションは、`myfile.txt` を表す `myfile` という名前の [Aws::S3::Object](#) オブジェクトを返します。
3. `Dir.chdir` を使用して、`/tmp` に作業ディレクトリを設定します。
4. `myfile.txt` という名前のファイルを開き、`myfile` の内容を書き込んで、ファイルを閉じます。

レシピを実行するには

1. 例のレシピで `default.rb` という名前のファイルを作成し、`recipes` ディレクトリに保存します。
2. `kitchen converge` を実行します。
3. `kitchen login` を実行してインスタンスにログインし、`ls /tmp` を実行します。いくつかの Test Kitchen ファイルおよびディレクトリと共に、`myfile.txt` が表示されるはずですが、

```
vagrant@s3bucket-ubuntu-1204:~$ ls /tmp
install.sh  kitchen  myfile.txt  stderr
```

ファイルの内容が正しいことを検証するために、`cat /tmp/myfile.txt` を実行することもできます。

完了したら、`kitchen destroy` を実行してインスタンスを終了します。

スタックの Linux インスタンスでの Ruby AWS OpsWorks SDK用 の使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

このトピックでは、スタックの Linux AWS OpsWorks インスタンスで SDK for Ruby を使用して Amazon S3 バケットからファイルをダウンロードする方法について説明します。AWS OpsWorks スタックは、すべての Linux インスタンスに SDK for Ruby を自動的にインストールします。ただし、サービスのクライアントオブジェクトを作成するときは、適切なAWS認証情報のセット、AWS::S3.new または他のサービスに同等のものを指定する必要があります。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、[「S3 バケットを空にする方法」](#) または [「S3 バケットを削除する方法」](#) を参照してください。

[Vagrant インスタンスでの SDK Ruby 用の使用](#)「」では、認証情報をノードオブジェクトに保存し、レシピコードで属性を参照することによって、認証情報が公開されるリスクを軽減する方法について説明しています。Amazon EC2 インスタンスでレシピを実行する場合は、[IAM ロール](#) というさらに優れたオプションがあります。

IAM ロールは、IAM ユーザーとほぼ同様に動作します。さまざまな AWS サービスを使用するためのアクセス許可を付与するポリシーがアタッチされています。ただし、ロールは個人ではなく Amazon EC2 インスタンスに割り当てます。そのようにすると、そのインスタンスで実行されているアプリケーションは、アタッチされたポリシーによって付与されたアクセス許可を取得できます。ロールを使用することで、認証情報はコード内に出現しなくなります。間接的に出現することもあります。このトピックでは、IAM ロールを使用して Amazon EC2 インスタンス [Vagrant インスタンスでの SDK Ruby 用の使用](#) からレシピを実行する方法について説明します。

このレシピは、「[例 9: Amazon EC2 インスタンスの使用](#)」で説明したように、kitchen-ec2 ドライバーを使用して Test Kitchen で実行することができます。ただし、Amazon EC2 インスタンスに SDK for Ruby をインストールするのは少し複雑であり、AWS OpsWorks スタックで心配する必要はありません。すべての AWS OpsWorks スタック Linux インスタンスには、デフォルトで SDK for Ruby がインストールされています。したがって、わかりやすくするために、この例では AWS OpsWorks スタックインスタンスを使用します。

最初のステップは、IAM ロールをセットアップすることです。この例では、最もシンプルなアプローチを採用します。つまり、最初のスタックの作成時に AWS OpsWorks スタックが作成する Amazon EC2 ロールを使用します。その名前は aws-opsworks-ec2-role です。ただし、AWS OpsWorks スタックはそのロールにポリシーをアタッチしないため、デフォルトではアクセス許可を付与しません。

AmazonS3ReadOnlyAccess ポリシーを aws-opsworks-ec2-role ロールにアタッチして、適切な権限を付与する必要があります。ロールにポリシーをアタッチする方法の詳細については、「IAMユーザーガイド」の [IAM「ID アクセス許可の追加 \(コンソール\)」](#) を参照してください。

スタックを作成または更新する際に、ロールを指定します。「[Linux インスタンスでのレシピの実行](#)」で説明したように、カスタムレイヤーを持つスタックをセットアップします。ただし、1つだけ追加の操作があります。スタックの追加ページで、デフォルトのIAMインスタンスプロファイルが aws-opsworks-ec2-role に設定されていることを確認します。その後、AWS OpsWorks スタックはそのロールをスタックのすべてのインスタンスに割り当てます。

クックブックをセットアップする手順は、「[Linux インスタンスでのレシピの実行](#)」で使用した手順に似ています。以下に簡単なまとめを示します。詳細については、例を参照してください。

クックブックをセットアップするには

1. s3bucket_ops という名前のディレクトリを作成し、そのディレクトリに移動します。
2. 以下の内容で metadata.rb ファイルを作成し、s3bucket_ops に保存します。

```
name "s3bucket_ops"
version "0.1.0"
```

3. recipes 内に s3bucket_ops ディレクトリを作成します。
4. 次のレシピで default.rb ファイルを作成し、recipes ディレクトリに保存します。

```
Chef::Log.info("*****Downloading a file from Amazon S3.*****")

ruby_block "download-object" do
  block do
    require 'aws-sdk'

    s3 = AWS::S3.new

    myfile = s3.buckets['cookbook_bucket'].objects['myfile.txt']
    Dir.chdir("/tmp")
    File.open("myfile.txt", "w") do |f|
      f.syswrite(myfile.read)
      f.close
    end
  end
end
```

```
action :run
end
```

- s3bucket_ops の .zip アーカイブを作成し、アーカイブを Amazon S3 バケットにアップロードします。分かりやすいように、[アーカイブを公開](#)してから、URL後で使用するためにアーカイブのを記録します。また、クックブックをプライベート Amazon S3 アーカイブやその他のタイプのリポジトリに保存することもできます。詳細については、「[クックブックリポジトリ](#)」を参照してください。

このレシピは、前の例で使用したレシピに似ていますが、以下の点が異なります。

- スタックは Ruby AWS OpsWorks SDK用の を既にインストールしているため、chef_gemリソースは削除されました。
- レシピが AWS::S3.new に認証情報を渡しません。

認証情報はインスタンスのロールに基づいてアプリケーションに自動的に割り当てられます。

- レシピは Chef::Log.info を使用して Chef ログにメッセージを追加します。

以下のようにこの例のスタックを作成します。既存の Windows スタックを使用することもできます。後で説明するように、クックブックを更新するだけです。

スタックを作成するには

- [AWS OpsWorks スタックコンソール](#)を開いて [Add Stack] をクリックします。
- 次の設定を指定し、その他の設定はデフォルト値を受け入れて、[Add Stack] をクリックします。
 - 名前 – RubySDK
 - デフォルトSSHキー – Amazon EC2キーペア

Amazon EC2キーペアを作成する必要がある場合は、「[Amazon EC2 キーペア](#)」を参照してください。キーペアは、インスタンスと同じAWSリージョンに属している必要があることに注意してください。この例では、デフォルトに米国西部(オレゴン) リージョンを使用します。

- [Add a layer] をクリックし、次の設定を使用してスタックに[カスタムレイヤーを追加](#)します。
 - Name (名前) – S3Download
 - Short name (短縮名) – s3download

Linux スタックの場合、どのレイヤータイプでも実際には動作しますが、この例は他のレイヤータイプによってインストールされるパッケージを必要としないので、カスタムレイヤーが最も簡単なアプローチです。

4. デフォルト設定でレイヤーに [24/7 インスタンスを追加](#)し、[起動](#)します。

これで、レシピをインストールして実行できるようになりました

レシピを実行するには

1. [カスタムクックブックを有効にするようにスタックを編集](#)し、以下の設定を指定します。

- Repository type (リポジトリタイプ) – Http Archive (Http アーカイブ)
- リポジトリ URL – 前に記録URLしたクックブックのアーカイブ。

その他の設定にはデフォルト値を使用し、[Save] をクリックしてスタック設定を更新します。

2. [\[Update Custom Cookbooks\] スタックコマンドを実行](#)します。スタックのインスタンスにあるカスタムクックブックの最新バージョンがインストールされます。以前のバージョンのクックブックがある場合は、このコマンドによって上書きされます。
3. 実行するレシピを `s3bucket_ops::default` に設定してレシピの実行スタックコマンドを実行することで、レシピを実行します。このコマンドは、`s3bucket_ops::default` から成る実行リストで Chef 実行を開始します。

Note

通常 AWS OpsWorks、スタックは[レシピを適切なライフサイクルイベントに割り当てることで自動的に実行](#)します。このようなレシピは、イベントを手動でトリガーすることによって実行できます。Setup および Configure イベントをトリガーするにはスタックコマンドを使用し、Deploy および Undeploy イベントをトリガーするには[デプロイコマンド](#)を使用できます。

レシピが正常に実行された後で、それを検証できます。

s3bucket_ops を検証するには

1. 最初のステップは、Chef ログを調べることです。スタックに、opstest1 という名前の 1 つのインスタンスがあるはずですが、[Instances] (インスタンス) ページで、インスタンスの [Log] (ログ) 列の [show] (表示) をクリックして、Chef ログを表示します。下へスクロールすると、下部にログメッセージが表示されます。

```
...
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
attributes/customize.rb in the cache.
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
metadata.rb in the cache.
[2014-07-31T17:01:46+00:00] INFO: *****Downloading a file from Amazon S3.*****
[2014-07-31T17:01:46+00:00] INFO: Processing template[/etc/hosts] action create
(opsworks_stack_state_sync::hosts line 3)
...
```

2. [SSH を使用してインスタンスにログイン](#)し、`/tmp` の内容を一覧表示します。

スタックの Windows インスタンスでの Ruby AWS OpsWorks SDK 用の使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

この例では、「[Windows インスタンスでのレシピの実行](#)」の例を実行済みであることを前提としています。実行済みでない場合は、最初にその例を実行する必要があります。特に、インスタンスRDPへのアクセスを有効にする方法について説明します。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、[「S3 バケットを空にする方法」](#)または[「S3 バケットを削除する方法」](#)を参照してください。

このトピックでは、スタックの Windows AWS OpsWorks インスタンス [AWS SDK for Ruby](#) を使用して S3 バケットからファイルをダウンロードする方法について説明します。

Ruby アプリケーションが AWS リソースにアクセスする必要がある場合は、適切なアクセス許可を持つ AWS 認証情報のセットを提供する必要があります。レシピの場合、AWS 認証情報を提供する最適なオプションは、AWS Identity and Access Management ([IAM](#)) [ロール](#) を使用することです。IAM ロールは、さまざまな AWS サービスを使用するアクセス許可を付与するポリシーがアタッチされている IAM ユーザーとよく似ています。ただし、ロールは、個人ではなく Amazon Elastic Compute Cloud (Amazon EC2) インスタンスに割り当てます。そのようにすると、そのインスタンスで実行されているアプリケーションは、アタッチされたポリシーによって付与されたアクセス許可を取得できます。ロールを使用することで、認証情報はコード内に出現しなくなります。間接的に出現することはありません。

最初のステップは、IAM ロールをセットアップすることです。この例では、最もシンプルなアプローチを採用します。つまり、最初のスタックの作成時に AWS OpsWorks スタックが作成する Amazon EC2 ロールを使用します。その名前は `aws-opsworks-ec2-role` です。ただし、AWS OpsWorks スタックはそのロールにポリシーをアタッチしないため、デフォルトではアクセス許可を付与しません。

`AmazonS3ReadOnlyAccess` ポリシーを `aws-opsworks-ec2-role` ロールにアタッチして、適切な権限を付与する必要があります。ロールにポリシーをアタッチする方法の詳細については、「IAM ユーザーガイド」の [IAM 「ID アクセス許可の追加 \(コンソール\)」](#) を参照してください。

スタックを作成または更新する際に、ロールを指定します。「[Windows インスタンスでのレシピの実行](#)」で説明したように、カスタムレイヤーを持つスタックをセットアップします。ただし、1 つだけ追加の操作があります。スタックの追加ページで、デフォルトの IAM インスタンスプロファイルが `aws-opsworks-ec2-role` に設定されていることを確認します。その後、AWS OpsWorks スタックはそのロールをスタックのすべてのインスタンスに割り当てます。

クックブックをセットアップする手順は、「[Linux インスタンスでのレシピの実行](#)」で使用した手順に似ています。以下に簡単なまとめを示します。詳細については、例を参照してください。

クックブックをセットアップするには

1. `s3bucket_ops` という名前のディレクトリを作成し、そのディレクトリに移動します。

2. 以下の内容で `metadata.rb` ファイルを作成し、`s3bucket_ops` に保存します。

```
name "s3download"
version "0.1.0"
```

3. `recipes` 内に `s3download` ディレクトリを作成します。
4. 以下のレシピで `default.rb` ファイルを作成し、`recipes` ディレクトリに保存します。を、ダウンロードするファイルの保存に使用する S3 バケットの名前 `windows-cookbooks` に置き換えます。

```
Chef::Log.info("*****Downloading an object from S3*****")

chef_gem "aws-sdk-s3" do
  compile_time false
  action :install
end

ruby_block "download-object" do
  block do
    require 'aws-sdk-s3'

    Aws.use_bundled_cert!

    s3_client = Aws::S3::Client.new(region:'us-west-2')

    s3_client.get_object(bucket: 'windows-cookbooks',
                        key: 'myfile.txt',
                        response_target: '/chef/myfile.txt')

  end
  action :run
end
```

5. `s3download` の `.zip` アーカイブを作成し、ファイルを S3 バケットにアップロードします。ファイルを公開し、後で使用するために URL を記録します。
6. `myfile.txt` という名前のテキストファイルを作成し、S3 バケットにアップロードします。これは、レシピによってダウンロードされるファイルであるため、便利なバケットを使用できません。

このレシピによって以下のタスクが実行されます。

1: SDK for Ruby v2 をインストールします。

この例では、SDK for Ruby を使用してオブジェクトをダウンロードします。ただし、AWS OpsWorks スタックは Windows インスタンスSDKにこれをインストールしないため、レシピの最初の部分では [chef_gem](#) リソースを使用してそのタスクを処理します。このリソースを使用して、レシピを含む gem が Chef 用にインストールされるようにします。

2: ファイルをダウンロードする。

レシピの 3 番目の部分では、 [ruby_block](#) リソースを使用して Ruby v2 コードSDKを実行し、という名前の S3 バケット myfile.txt からインスタンスの /chef ディレクトリ *windows-cookbooks* にダウンロードします。 *windows-cookbooks* を myfile.txt を含むバケットの名前に変更します。

Note

レシピは Ruby アプリケーションであるため、Ruby コードをレシピの本文内に記述できません。 `ruby_block` リソース内に記述する必要はありません。しかし、Chef によってレシピの本文内の Ruby のコードが最初に実行され、その後、各リソースが順番に実行されます。この例では、レシピの本文にダウンロードコードを配置すると、Ruby SDKの に依存し、 をインストールする `chef_gem` リソースがまだ実行SDKされていないため、失敗します。 `ruby_block` リソース内のコードは、リソースの実行時に実行され、 `chef_gem` リソースが SDK for Ruby をインストールした後に実行されます。

以下のようにこの例のスタックを作成します。既存の Windows スタックを使用することもできます。後で説明するように、クックブックを更新するだけです。

Stack を作成する

1. [AWS OpsWorks スタックコンソール](#)を開いて [Add Stack] を選択します。以下の設定を指定し、その他の設定はデフォルト値を受け入れて、[Add Stack] を選択します。

- Name (名前) – S3Download
- リージョン – 米国西部 (オレゴン)

この例はいずれのリージョンでも動作しますが、チュートリアルでは米国西部 (オレゴン) を使用することをお勧めします。

- Default operating system (デフォルトのオペレーティングシステム) – Microsoft Windows Server 2012 R2
2. [Add a layer] を選択し、以下の設定を使用してスタックに[カスタムレイヤーを追加](#)します。
 - Name (名前) – S3Download
 - Short name (短縮名) – s3download
 3. デフォルト設定で S3Downloadレイヤーに [24/7 インスタンスを追加](#)し、[起動](#)します。

これで、レシピをインストールして実行できるようになりました

レシピを実行するには

1. [カスタムクックブックを有効にするようにスタックを編集](#)し、以下の設定を指定します。
 - Repository type (リポジトリタイプ) – S3 Archive (アーカイブ)
 - リポジトリ URL – 前に記録URLしたクックブックのアーカイブ。

その他の設定ではデフォルト値を受け入れ、[Save] を選択してスタック設定を更新します。

2. [\[Update Custom Cookbooks\] スタックコマンドを実行](#)します。スタックのオンラインインスタンスにカスタムクックブックの最新バージョンがインストールされます。以前のバージョンのクックブックがある場合は、このコマンドによって上書きされます。
3. 実行するレシピを `s3download::default` に設定してレシピの実行スタックコマンドを実行することで、レシピを実行します。このコマンドは、`s3download::default` から成る実行リストで Chef 実行を開始します。

Note

通常 AWS OpsWorks、スタックは[レシピを適切なライフサイクルイベントに割り当てることで自動的に実行](#)します。このようなレシピは、イベントを手動でトリガーすることも実行できます。Setup および Configure イベントをトリガーするにはスタックコマンドを使用し、Deploy および Undeploy イベントをトリガーするには[デプロイコマンド](#)を使用できます。

レシピが正常に実行された後で、それを検証できます。

s3download を検証するには

1. 最初のステップは、Chef ログを調べることです。スタックに s3download1 という名前の 1 つのインスタンスがあります。Instances ページで、インスタンスの ログ 列の 表示 を選択すると、Chef のログが表示されます。下にスクロールすると、末尾近くに以下のようなログメッセージが見つかります。

```
...
[2015-05-01T21:11:04+00:00] INFO: Loading cookbooks [s3download@0.0.0]
[2015-05-01T21:11:04+00:00] INFO: Storing updated cookbooks/s3download/recipes/default.rb in the cache.
[2015-05-01T21:11:04+00:00] INFO: *****Downloading an object from S3*****
[2015-05-01T21:11:04+00:00] INFO: Processing chef_gem[aws-sdk] action install
(s3download::default line 3)
[2015-05-01T21:11:05+00:00] INFO: Processing ruby_block[download-object] action run
(s3download::default line 8)
...
```

2. [RDP を使用してインスタンスにログイン](#)し、 の内容を調べます c:\chef。

Windows ソフトウェアのインストール

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

以下の例では、「[Windows インスタンスでのレシピの実行](#)」の例を実行済みであることを前提としています。実行済みでない場合は、最初にその例を実行する必要があります。特に、インスタンスRDPへのアクセスを有効にする方法について説明します。

Windows インスタンスは Windows Server 2012 R2 Standard で起動されるため、通常、いくつかのソフトウェアをインストールする必要があります。詳細はソフトウェアのタイプによって異なります。

- Windows NET機能は、フレームワークや Internet Information Services (IIS) などのオプションのシステムコンポーネントであり、インスタンスにダウンロードできます。
- サードパーティソフトウェアは通常、MSI ファイルなどのインストーラパッケージに付属しており、インスタンスにダウンロードしてから実行する必要があります。

一部の Microsoft ソフトウェアもインストーラパッケージで提供されます。

このセクションでは、Windows の機能とパッケージをインストールするクックブックを実装する方法について説明します。また、Chef Windows クックブックも紹介します。このクックブックには、Windows インスタンスの実装レシピを簡素化するリソースやヘルパー関数が含まれています。

トピック

- [Windows 機能のインストール: IIS](#)
- [Windows Instance でのパッケージのインストール](#)

Windows 機能のインストール: IIS

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Windows 機能は、.NET フレームワークや Internet Information Services (IIS) など、オプションのシステムコンポーネントのセットです。このトピックでは、一般的に使用される機能である Internet Information Services () をインストールするためのクックブックを実装する方法について説明します IIS。

Note

[パッケージのインストール](#) は、ファイルなど、インストーラパッケージに付属するソフトウェアをインストールする方法を示しています。このMSIファイルは、インスタンスにダウンロードして実行する必要があります。ク[IIS cookbook](#)

「[Windows インスタンスでのレシピの実行](#)」では、powershell_script リソースを使用して Windows の機能をインストールする方法を示しています。この例では、別のアプローチとして、Chef の [Windows cookbook](#) の windows_feature リソースを使用しています。この cookbook には、[デプロイイメージサービスおよび管理](#) を使用して Windows でさまざまなタスク (機能のインストールなど) を実行する一連のリソースが含まれています。

Note

Chef には IIS cookbook もあり、の管理に使用できます IIS。詳細については、ク[IIS cookbook](#) を参照してください。

cookbook をセットアップするには

1. [Windows cookbook GitHub リポジトリ](#) に移動し、ク windows cookbook をダウンロードします。

この例では、windows リポジトリを .zip ファイルとしてダウンロードしますが、必要に応じてリポジトリを複製できることも前提としています。

2. [chef_handler cookbook GitHub リポジトリ](#) に移動し、ク chef-handler cookbook をダウンロードします。

windows cookbook は chef_handler に依存しており、直接使用することはありません。この例では、chef_handler リポジトリを .zip ファイルとしてダウンロードしますが、必要に応じてリポジトリを複製できることも前提としています。

3. windows と chef_handler という名前のディレクトリに windows および chef_handler cookbook を抽出します。
4. install-iis という名前の cookbook ディレクトリ内にディレクトリを作成し、そのディレクトリに移動します。
5. install-iis に、次のコンテンツを含む metadata.rb ファイルを追加します。

```
name "install-iis"  
version "0.1.0"  
  
depends "windows"
```

depends ディレクティブにより、レシピで windows クックブックのリソースを使用できません。

6. recipes ディレクトリを install-iis に追加し、以下のレシピコードを保存した default.rb という名前のレシピファイルをそのディレクトリに追加します。

```
%w{ IIS-WebServerRole IIS-WebServer }.each do |feature|  
  windows_feature feature do  
    action :install  
  end  
end  
  
service 'w3svc' do  
  action [:start, :enable]  
end
```

レシピによって windows クックブックの windows_feature リソースが使用されて、以下のものがインストールされます。

1. [IIS ウェブサーバーロール](#)。
2. [IIS ウェブサーバー](#)。

次に、レシピは [service](#) リソースを使用して IIS サービス (W3) を開始および有効にします SVC。

Note

使用可能な Windows 機能の完全なリストについては、[RDPを使用してインスタンスにログイン](#)し、コマンドプロンプトウィンドウを開いて、次のコマンドを実行します。リストは非常に長くなっています。

```
dism /online /Get-Features
```

7. `install-iis`、`chef_handler`、および `windows` クックブックを含む `.zip` アーカイブを作成し、アーカイブを S3 バケットにアップロードします。アーカイブを公開し、後で使用するために URL を記録します。この例では、アーカイブ名を `install-iis.zip` としています。詳細については、「[クックブックリポジトリ](#)」を参照してください。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、「[S3 バケットを空にする方法](#)」または「[S3 バケットを削除する方法](#)」を参照してください。

以下のようにこの例のスタックを作成します。既存の Windows スタックを使用することもできます。後で説明するように、クックブックを更新するだけです。

Stack を作成する

1. [AWS OpsWorks スタックコンソール](#)を開いて [Add Stack] を選択します。以下の設定を指定し、その他の設定はデフォルト値を受け入れて、[Add Stack] を選択します。

- 名前 – インストールIIS
- リージョン – 米国西部 (オレゴン)

この例はいずれのリージョンでも動作しますが、チュートリアルでは米国西部 (オレゴン) を使用することをお勧めします。

- Default operating system (デフォルトのオペレーティングシステム) – Microsoft Windows Server 2012 R2
2. [Add a layer] を選択し、以下の設定を使用してスタックに[カスタムレイヤーを追加](#)します。
 - 名前 – IIS
 - Short name (短縮名) – iis
 3. デフォルト設定の [24/7 インスタンス](#)をIISレイヤーに追加して[起動](#)します。

これで、クックブックをインストールし、レシピを実行できるようになりました。

クックブックをインストールし、レシピを実行するには

1. [カスタムクックブックを有効にするようにスタックを編集](#)し、以下の設定を指定します。

- Repository type (リポジトリタイプ) - S3 Archive (S3 アーカイブ)
- リポジトリ URL – 前に記録URLしたクックブックアーカイブの。

その他の設定ではデフォルト値を受け入れ、[Save] を選択してスタック設定を更新します。

2. [\[Update Custom Cookbooks\] スタックコマンド](#)を実行します。スタックのオンラインインスタンスにカスタムクックブックの最新バージョンがインストールされます。以前のバージョンのクックブックがある場合は、このコマンドによって上書きされます。
3. カスタムクックブックの更新が終了したら、実行するレシピを `install-iis::default` に設定してレシピの実行スタックコマンドを実行することで、レシピを実行します。このコマンドによって Chef 実行が開始され、指定したレシピが実行されます。

Note

この例では、便宜上 Execute Recipes を使用していますが、通常 AWS OpsWorks、適切なライフサイクルイベントに割り当てることによって、スタックで[レシピを自動的に実行](#)します。このようなレシピは、イベントを手動でトリガーすることによって実行できます。Setup および Configure イベントをトリガーするにはスタックコマンドを使用し、Deploy および Undeploy イベントをトリガーするには[デプロイコマンド](#)を使用できます。

4. インストールを確認するには、[を使用してインスタンスRDPに接続](#)し、Windows Explorer を開きます。ファイルシステムに C:\inetpub ディレクトリがあることを確認できます。管理ツールのコントロールパネルアプリケーションでサービスのリストを確認する場合、IISは下部にあります。ただし、ではなく、World Wide Web Publishing Service という名前になりますIIS。

Windows Instance でのパッケージのインストール

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

この例では、「[Windows インスタンスでのレシピの実行](#)」の例を実行済みであることを前提としています。実行済みでない場合は、最初にその例を実行する必要があります。特に、インスタンスRDPへのアクセスを有効にする方法について説明します。

ソフトウェアが などのインストーラパッケージに含まれている場合はMSI、ファイルをインスタンスにダウンロードしてから実行する必要があります。この例では、MSIパッケージ、Python ランタイムをインストールするためのクックブックを実装する方法と、関連する環境変数を定義する方法を示します。などの Windows 機能をインストールする方法の詳細についてはIIS、「」を参照してください[Windows 機能のインストール: IIS](#)。

クックブックをセットアップするには

1. `installpython` という名前のディレクトリを作成し、そのディレクトリに移動します。
2. `installpython` に、次のコンテンツを含む `metadata.rb` ファイルを追加します。

```
name "installpython"
version "0.1.0"
```

3. `installpython` に `recipes` ディレクトリと `files` ディレクトリを追加して、ファイルに `default` ディレクトリを追加します。
4. [Python Releases for Windows](#) からクックブックの `files\default` ディレクトリに Python パッケージをダウンロードします。この例では、`python-3.4.3.amd64.msi` という名前のMSIインストーラを使用する Python 3.5.0a3 の Windows x86-64 バージョンをインストールします。
5. 以下のレシピコードを保存した `default.rb` という名前のファイルを `recipes` ディレクトリに追加します。

```
directory 'C:\tmp' do
  rights :full_control, 'Everyone'
```



```
recursive true
action :create
end

cookbook_file 'C:\tmp\python-3.4.3.amd64.msi' do
  source "python-3.4.3.amd64.msi"
  rights :full_control, 'Everyone'
  action :create
end

windows_package 'python' do
  source 'C:\tmp\python-3.4.3.amd64.msi'
  action :install
end

env "PATH" do
  value 'c:\python34'
  delim ";"
  action :modify
end
```

このレシピでは、以下のような処理を実行します。

1. [ディレクトリ](#) リソースを使用して、C:\tmp ディレクトリを作成します。

このリソースの詳細については、「[例 3: ディレクトリの作成](#)」を参照してください。

2. [cookbook_file](#) リソースを使用して、クックブックの files\default ディレクトリから C:\tmp にインストーラをコピーします。

このリソースの詳細については、「[クックブックからのファイルのインストール](#)」を参照してください。

3. [Windows_package](#) リソースを使用して MSI インストーラを実行し、Python を c:\python34 にインストールします。

インストーラによって必要なディレクトリを作成され、ファイルがインストールされますが、システムの PATH 環境変数は変更されません。

4. [env](#) リソースを使用して、システムパスに c:\python34 を追加します。

env リソースを使用して、環境変数を定義します。この例では、レシピによって c:\python34 がパスに追加されて、コマンドラインから Python スクリプトを簡単に実行できるようになります。

- この例では、リソース名に環境変数の名前 PATH を指定します。
 - この例では、value 属性に変数の値 c:\\python34 を指定します (\ 文字をエスケープする必要があります)。
 - :modify アクションによって、指定した値が変数の現在の値に付加されます。
 - delim 属性には、既存の値と新しい値を区切る記号 (この例では ;) を指定します。
6. .zip の installpython アーカイブを作成し、S3 バケットにアップロードして、公開します。URL 後で使用するためにアーカイブの を記録します。詳細については、「[クックブックリポジトリ](#)」を参照してください。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、「[S3 バケットを空にする方法](#)」または「[S3 バケットを削除する方法](#)」を参照してください。

以下のようにこの例のスタックを作成します。既存の Windows スタックを使用することもできます。後で説明するように、クックブックを更新するだけです。

Stack を作成する

1. [AWS OpsWorks スタックコンソール](#)を開いて [Add Stack] を選択します。以下の設定を指定し、その他の設定はデフォルト値を受け入れて、[Add Stack] を選択します。
 - 名前 – InstallPython
 - リージョン – 米国西部 (オレゴン)

この例はいずれのリージョンでも動作しますが、チュートリアルでは米国西部 (オレゴン) を使用することをお勧めします。

 - Default operating system (デフォルトのオペレーティングシステム) – Microsoft Windows Server 2012 R2
2. [Add a layer] を選択し、以下の設定を使用してスタックに[カスタムレイヤーを追加](#)します。
 - Name (名前) – Python
 - Short name (短縮名) – python
3. デフォルト設定で Pythonレイヤーに [24/7 インスタンスを追加](#)し、[起動](#)します。

インスタンスがオンラインになった後、クックブックをインストールし、レシピを実行できます


クックブックをインストールし、レシピを実行するには

1. [カスタムクックブックを有効にするようにスタックを編集](#)し、以下の設定を指定します。

- Repository type (リポジトリタイプ) – S3 Archive (アーカイブ)
- リポジトリ URL – 前に記録URLしたクックブックのアーカイブ。

その他の設定ではデフォルト値を受け入れ、[Save] を選択してスタック設定を更新します。

2. [\[Update Custom Cookbooks\] スタックコマンド](#)を実行します。スタックのオンラインインスタンスにカスタムクックブックの最新バージョンがインストールされます。以前のバージョンのクックブックがある場合は、このコマンドによって上書きされます。
3. カスタムクックブックの更新が終了したら、実行するレシピを `installpython::default` に設定してレシピの実行スタックコマンドを実行することで、レシピを実行します。このコマンドは、`installpython::default` から成る実行リストで Chef 実行を開始します。

 Note

この例では、便宜上 Execute Recipes を使用していますが、通常 AWS OpsWorks、スタックはレシピを適切なライフサイクルイベントに割り当てることで[自動的に実行](#)します。このようなレシピは、イベントを手動でトリガーすることによって実行できます。Setup および Configure イベントをトリガーするにはスタックコマンドを使用し、Deploy および Undeploy イベントをトリガーするには[デプロイコマンド](#)を使用できます。

4. インストールを確認するには、[を使用してインスタンスRDPに接続](#)し、Windows Explorer を開きます。

- ファイルシステムに `C:\Python34` ディレクトリがあることを確認できます。
- コマンドラインから `path` を実行した場合は、`PATH=c:\python34;C:\Windows\system32;...` のようになります。
- コマンドラインから `python --version` を実行した場合は、Python 3.4.3 が返されます。

組み込み属性の上書き

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このトピックは Linux スタックにのみ当てはまります。Windows スタックの組み込み属性は上書きできません。

AWS OpsWorks スタックは、各インスタンスに一連の組み込みクックブックをインストールします。組み込みのクックブックの多くは組み込みのレイヤーをサポートしており、属性ファイルは Apache サーバー設定のような、さまざまなデフォルト システムとアプリケーション設定を定義します。属性ファイルにこれらの設定を指定することで、以下の方法のいずれかで、対応する組み込み属性を上書きすることにより多くの設定をカスタマイズできます。

- カスタムで属性を定義しますJSON。

このアプローチには、シンプルかつ柔軟であるという利点があります。ただし、カスタムJSONを手動で入力する必要があるため、属性定義を管理する堅牢な方法はありません。

- カスタムクックブックを実装して、`customize.rb` 属性ファイルの属性を定義します。

このアプローチは、カスタムを使用するよりも柔軟性は低くなりますがJSON、カスタムクックブックをソース管理下に置くことができるため、より堅牢です。

このトピックでは、Apache サーバーを例として、カスタムクックブック属性ファイルを使用して組み込み属性を上書きする方法について説明します。カスタムで属性を上書きする方法の詳細についてはJSON、「」を参照してください[カスタムの使用 JSON](#)。属性を上書きする方法の全般的な説明については、「[属性の上書き](#)」を参照してください。

Note

属性を上書きすることは設定をカスタマイズする優れた方法ですが、設定は常に属性で表されるものではありません。その場合は、設定ファイルを作成するために組み込みのレシピが使用するテンプレートを上書きすることで、設定ファイルをカスタマイズしてことができます。例については、[組み込みテンプレートの上書き](#)を参照してください。

組み込みの属性は、通常設定ファイルを作成するためにセットアップレシピが使用するテンプレートファイルの値を示します。例えば、apache2 セットアップレシピの一つである [default.rb](#) では、[apache2.conf.erb](#) テンプレートを使用して Apache サーバーの主要な設定ファイルである httpd.conf (Amazon Linux) または apache2.conf (Ubuntu) を作成します。以下は、テンプレートファイルからの抜粋です。

```
...
#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests <%= node[:apache][:keepaliverequests] %>
#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout <%= node[:apache][:keepalivetimeout] %>
##
## Server-Pool Size Regulation (MPM specific)
##
...
```

この例の KeepAliveTimeout 設定は、[:apache][:keepalivetimeout] 属性の値です。この属性のデフォルト値は、以下の抜粋に示す、apache2 クックブックの [apache.rb](#) 属性ファイルで定義されます。

```
...
# General settings
```

```
default[:apache][:listen_ports] = [ '80','443' ]
default[:apache][:contact] = 'ops@example.com'
default[:apache][:log_level] = 'info'
default[:apache][:timeout] = 120
default[:apache][:keepalive] = 'Off'
default[:apache][:keepaliverequests] = 100
default[:apache][:keepalivetimeout] = 3
...
```

Note

よく使われる組み込みの属性の詳細については、「[組み込みクックブックの属性](#)」を参照してください。

組み込み属性の上書きをサポートするために、すべての組み込みクックブックには `customize.rb` 属性ファイルが含まれています。このファイルは、`include_attribute` デイレクティブを介してすべてのモジュールに組み込まれます。組み込みクックブックの `customize.rb` ファイルには属性定義は含まれておらず、組み込み属性には影響を与えません。組み込み属性を上書きするには、同じ名前のカスタムクックブックを組み込みクックブックとして作成して、`customize.rb` という名前が付けられている属性ファイルにカスタム属性定義として配置します。そのファイルは組み込みバージョンに優先され、関連する各モジュールに含まれます。`customize.rb` に組み込み属性を定義すると、対応する組み込み属性が上書きされます。

この例では、組み込み `[:apache][:keepalivetimeout]` 属性を設定して、その値に 3 の代わりに 5 を設定する方法を示します。すべての組み込み属性に対して同様の方法を使用できます。ただし、上書きする属性には注意が必要です。例えば、`opsworks` 名前空間の属性を上書きすると、組み込みのレシピで問題が発生する場合があります。

Important

組み込み属性ファイル自体のコピーを変更することで、組み込み属性を上書きしないでください。例えば、`apache.rb` のコピーをカスタムクックブックの `apache2/attributes` フォルダに配置して、その設定の一部を変更することができます。ただし、このファイルは組み込みバージョンを上書きしてしまうため、組み込みレシピは新たなバージョンの `apache.rb` を使用するようになります。AWS OpsWorks スタックが後で組み込み `apache.rb` ファイルを変更した場合、手動でバージョンを更新しない限り、レシピは新しい値を取得しません。を使用すると `customize.rb`、指定された属性のみを上書きします。

組み込みレシピは、上書きされていないすべての属性の値を自動的に取得 up-to-dateし続けます。

開始するには、カスタムクックブックを作成します。

クックブックを作成するには

1. `opsworks_cookbooks` ディレクトリ内に、`apache2` というクックブックのディレクトリを作成して、そこに移動します。

組み込み属性を上書きするには、カスタムクックブックは組み込みのクックブックと同じ名前である必要があります。この例では `apache2` です。

2. `apache2` ディレクトリに `attributes` ディレクトリを作成します。
3. `customize.rb` ファイルを `attributes` ディレクトリに追加し、それを使用して上書きする組み込みクックブックの属性を定義します。この例では、ファイルには以下のものが含まれます。

```
normal[:apache][:keepalivetimeout] = 5
```

Important

組み込みの属性を上書きするには、カスタム属性のタイプが `normal` 以上であり、対応する組み込み属性名と正確に同じである必要があります。`normal` タイプでは、すべての `default` タイプの組み込み属性にカスタム属性が優先されます。詳細については、「[属性の優先順位](#)」を参照してください。

4. `opsworks_cookbooks.zip` という名前の `opsworks_cookbooks` の `.zip` アーカイブを作成し、Amazon Simple Storage Service (Amazon S3) バケットにそのアーカイブをアップロードします。わかりやすいよう、[ファイルを公開](#)します。後で使用できるURLようにを記録します。また、クックブックをプライベート Amazon S3 アーカイブやその他のタイプのリポジトリに保存することもできます。詳細については、「[クックブックリポジトリ](#)」を参照してください。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、「[S3 バケットを空にする方法](#)」または「[S3 バケットを削除する方法](#)」を参照してください。

カスタム属性を使用するには、スタックを作成し、カスタムクックブックをインストールします。

カスタム属性を使用する

1. [AWS OpsWorks スタックコンソール](#)を開いて [Add Stack] を選択します。

2. 以下の標準設定を指定できます。

- 名前 – ApacheConfig
- リージョン – 米国西部 (オレゴン)

いずれのリージョンにもスタックを配置できますが、チュートリアルでは米国西部 (オレゴン) をお勧めします。

- デフォルトSSHキー – EC2キーペア

EC2 キーペアを作成する必要がある場合は、[「Amazon EC2 キーペア」](#)を参照してください。キーペアはスタックと同じAWSリージョンに属している必要があることに注意してください。

[Advanced>>] を選択し、[Use custom Chef cookbooks] を [Yes] に設定して、以下の設定を指定します。

- Repository type (リポジトリタイプ) – Http Archive (Http アーカイブ)
- リポジトリ URL – 前に記録URLしたクックブックアーカイブの

そのほかの設定のデフォルト値を受け入れ、[Add Stack] を選択してスタックを作成します。

Note

この例では、デフォルトのオペレーティングシステムである Amazon Linux を使用します。希望に応じて、Ubuntu を使用することもできます。唯一の相違点は Ubuntu システムであるということで、組み込みセットアップレシピが `apache2.conf` という同じ名前の設定ファイルを作成し、それを `/etc/apache2` ディレクトリに配置します。

3. レイヤーの追加 を選択してから、デフォルト設定でスタックに [Java アプリケーションアプリケーションサーバーレイヤーを追加](#)します。

4. デフォルト設定でレイヤーに [24/7 インスタンスを追加](#)し、インスタンスを起動します。

この例では、t2.micro インスタンスで十分です。

5. インスタンスがオンラインになったら、[を使用してインスタンスに接続します SSH](#)。httpd.conf ファイルは、/etc/httpd/conf ディレクトリにあります。ファイルを確認すると、カスタム KeepAliveTimeout 設定が表示されます。残りの設定には、組み込みの apache.rb ファイルからのデフォルト値があります。httpd.conf の関連する部分は以下のようになります。

```
...  
#  
# KeepAliveTimeout: Number of seconds to wait for the next request from the  
# same client on the same connection.  
#  
KeepAliveTimeout 5  
...
```

組み込みテンプレートの上書き

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このトピックは Linux スタックにのみ当てはまります。Windows スタックの組み込みテンプレートは上書きできません。

AWS OpsWorks スタックの組み込みレシピは、テンプレートを使用して、主に Apache などのサーバーの設定ファイルをインスタンスに作成します。例えば、apache2 レシピでは、[apache2.conf.erb](#) テンプレートを使用して Apache サーバーのプライマリ設定ファイルである httpd.conf (Amazon Linux) または apache2.conf (Ubuntu) を作成します。

これらのテンプレートの設定のほとんどは属性によって表されるため、設定ファイルをカスタマイズする主な方法は、適切な組み込み属性を上書きすることです。例については、[組み込み属性の上書き](#)を参照してください。ただし、カスタマイズする設定が組み込み属性で表されない場合、またはテンプレート内にない場合は、テンプレート自体を上書きする必要があります。このトピックでは、組み込みテンプレートを上書きして、カスタム Apache 設定を指定する方法について説明します。

ErrorDocument 設定を httpd.conf ファイルに追加することで、Apache にカスタムラー応答を提供することができます。apache2.conf.erb には、以下のようないくつかのコメントアウトされた例だけがあります。

```
...
#
# Customizable error responses come in three flavors:
# 1) plain text 2) local redirects 3) external redirects
#
# Some examples:
#ErrorDocument 500 "The server made a boo boo."
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
...
```

これらの設定はハードコードコメントであるため、属性を上書きすることでカスタム値を指定することはできません。テンプレート自体を上書きする必要があります。ただし、属性とは異なり、テンプレート ファイルに含まれる特定の部分を上書きする方法はありません。組み込みバージョンと同じ名前で作成したカスタムクックブックを作成し、テンプレートファイルと同じサブディレクトリにコピーして、必要に応じてファイルを変更します。このトピックでは、apache2.conf.erb を上書きしてカスタム応答にエラー 500 を提供する方法について説明します。テンプレートの上書きについての一般的な説明については、「[カスタムテンプレートの使用](#)」を参照してください。

Important

built組み込みテンプレートを上書きすると、組み込みレシピは組み込みバージョンではなく、カスタマイズされたバージョンのテンプレートを使用します。AWS OpsWorks スタックが組み込みテンプレートを更新すると、カスタムテンプレートは同期しなくなり、正しく動作しない可能性があります。AWS OpsWorks スタックはそのような変更を頻繁に行うことはありません。テンプレートが変更されると、AWS OpsWorks スタックは変更を一覧表示し、新しいバージョンにアップグレードするオプションを提供します。[AWS OpsWorks スタックのリポジトリ](#)の変更をモニタリングし、必要に応じてテンプレートを手動で更新する

ことをお勧めします。リポジトリに、サポートされている Chef バージョンにそれぞれ個別のブランチがあり、正しいブランチにいることを確認します。

開始するには、カスタムクックブックを作成します。

クックブックを作成するには

1. opsworks_cookbooks ディレクトリ内に、apache2 というクックブックのディレクトリを作成して、そこに移動します。組み込みテンプレートを上書きするには、カスタムクックブックは組み込みのクックブックと同じ名前である必要があります。この例では apache2 です。

Note

すでに「[組み込み属性の上書き](#)」ウォークスルーを完了している場合は、この例で同じ apache2 クックブックを使用して、ステップ 2 をスキップできます。

2. 以下の内容で metadata.rb ファイルを作成し、apache2 ディレクトリに保存します。

```
name "apache2"
version "0.1.0"
```

3. apache2 ディレクトリに templates/default ディレクトリを作成します。

Note

templates/default ディレクトリは、デフォルトの apache2.conf.erb テンプレートを使用する Amazon Linux インスタンスで動作します。Ubuntu 14.04 インスタンスは、オペレーティングシステム固有の apache2.conf.erb テンプレートを使用します。これは templates/ubuntu-14.04 ディレクトリにあります。Ubuntu 14.04 インスタンスにカスタム設定を適用する場合は、そのテンプレートも上書きする必要があります。

4. templates/default ディレクトリに、[\[built-in apache2.conf.erb template](#) (組み込みテンプレート)] をコピーします。テンプレートファイルを開いて ErrorDocument 500 行をコメントアウトすると、以下の様なカスタムエラーメッセージが表示されます。

```
...
ErrorDocument 500 "A custom error message."
#ErrorDocument 404 /missing.html
...
```

5. `opsworks_cookbooks.zip` という名前の `opsworks_cookbooks` の `.zip` アーカイブを作成し、ファイルを Amazon Simple Storage Service (Amazon S3) バケットにアップロードします。わかりやすいよう、[アーカイブを公開](#)します。URL 後で使用するためにアーカイブの を記録します。また、クックブックをプライベート Amazon S3 アーカイブやその他のタイプのリポジトリに保存することもできます。詳細については、「[クックブックリポジトリ](#)」を参照してください。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、「[S3 バケットを空にする方法](#)」または「[S3 バケットを削除する方法](#)」を参照してください。

Note

わかりやすいように、この例では、テンプレートにハードコード化されたエラーメッセージを追加します。これを変更するには、テンプレートを変更して、[クックブックを再インストール](#)する必要があります。より柔軟な方法として、[デフォルトカスタム属性](#)をカスタムクックブック `customize.rb` 属性のエラー文字列に定義して、その属性の値を `ErrorDocument 500` に割り当てることができます。例えば、属性に `[:apache][:custom][:error500]` という名前を付けると、`apache2.conf.erb` の対応する行は以下ようになります。

```
...
ErrorDocument 500 <%= node[:apache][:custom][:error500] %>
#ErrorDocument 404 /missing.html
...
```

`[:apache][:custom][:error500]` を上書きすることで、カスタムエラーメッセージをいつでも変更できます。[カスタムを使用して属性をJSON上書き](#)する場合、クックブックを操作する必要もありません。

カスタムテンプレートを使用するには、スタックを作成し、カスタムクックブックをインストールします。

カスタムテンプレートを使用する

1. [AWS OpsWorks スタックコンソール](#)を開いて [Add Stack] を選択します。
2. 以下の標準設定を指定できます。

- 名前 – ApacheTemplate
- リージョン – 米国西部 (オレゴン)
- デフォルトSSHキー – Amazon Elastic Compute Cloud (Amazon EC2) キーペア

Amazon EC2キーペアを作成する必要がある場合は、[「Amazon EC2 キーペア」](#)を参照してください。キーペアは、インスタンスと同じAWSリージョンに属している必要があることに注意してください。

[Advanced>>] を選択し、[Use custom Chef cookbooks] を選択して、以下の設定を指定します。

- Repository type (リポジトリタイプ) – Http Archive (Http アーカイブ)
- リポジトリ URL – 前に記録URLしたクックブックアーカイブの

そのほかの設定のデフォルト値を受け入れ、[Add Stack] を選択してスタックを作成します。

3. [Add a layer] (レイヤーの追加) を選択してから、デフォルト設定でスタックに [\[add a Java App Server layer\]](#) (Java アプリケーションサーバーレイヤーを追加) します。
4. デフォルト設定でレイヤーに [24/7 インスタンスを追加](#)し、インスタンスを起動します。

この例では、t2.micro インスタンスで十分です。

5. インスタンスがオンラインになったら、[を使用してインスタンスに接続します SSH](#)。httpd.conf ファイルは、/etc/httpd/conf ディレクトリにあります。ファイルには、以下のようなカスタムの ErrorDocument設定が含まれています。

```
...
# Some examples:
ErrorDocument 500 "A custom error message."
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
```

...

レイヤーの負荷分散

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックには、[Elastic Load Balancing](#) との 2 つの負荷分散オプションが用意されています。これらは、通常 [HAProxy](#)、アプリケーションサーバーレイヤーのインスタンス間で負荷を分散するために使用されます。このトピックでは、レイヤーに負荷分散を追加するときのオプションの選択に役立つように、各オプションの利点と制限事項について説明します。場合によって、両方のオプションの使用をお勧めします。

SSL 終了

組み込み HAProxy レイヤーは SSL 終了を処理しません。サーバー SSL で終了する必要があります。このアプローチの利点は、トラフィックがサーバーに到達するまで暗号化されることです。ただし、サーバーによる復号の処理が必要であり、それによりサーバーの負荷が増えます。さらに、ユーザーがよりアクセスしやすいアプリケーションサーバーに SSL 証明書を配置する必要があります。

Elastic Load Balancing を使用すると、ロードバランサー SSL で終了できます。これにより、アプリケーションサーバーの負荷は低減されますが、ロードバランサーとサーバーとの間のトラフィックは暗号化されません。Elastic Load Balancing では、[サーバー SSL で終了](#) することもできますが、セットアップは少し複雑です。

スケーリング

受信トラフィックが HAProxy ロードバランサーの容量を超える場合は、その容量を手動で増やす必要があります。

Elastic Load Balancing は、着信トラフィックを処理するように自動的にスケールされます。最初にオンラインになったときに想定負荷を処理するための容量を十分に確保するには、Elastic Load Balancing ロードバランサーを[事前ウォーミング](#)します。

ロードバランサーの障害

HAProxy サーバーをホストしているインスタンスに障害が発生した場合、インスタンスを再起動できるようになるまでサイト全体がオフラインになる可能性があります。

Elastic Load Balancing は、よりも耐障害性に優れていますHAProxy。例えば、EC2インスタンスを登録した各アベイラビリティゾーンにロードバランシングノードをプロビジョニングします。1つのゾーンでサービスが中断された場合は、他のノードで受信トラフィックの処理が継続されます。詳細については、「[Elastic Load Balancing Concepts](#)」(Elastic Load Balancing のコンセプト) を参照してください。

アイドルタイムアウト

指定したアイドルタイムアウト値よりも長い時間アイドル状態になった場合、両方のロードバランサーによって接続が終了されます。

- HAProxy – アイドルタイムアウト値には上限がありません。
- Elastic Load Balancing – デフォルトのアイドルタイムアウト値は 60 秒で、最大値は 3600 秒 (60 分) です。

Elastic Load Balancing のアイドル時間の制限は、ほとんどの目的に対して十分たります。より長いアイドルタイムアウトHAProxyが必要な場合は、を使用することをお勧めします。以下に例を示します。

- プッシュ通知に使用される長時間実行されるHTTP接続。
- 管理インターフェイスを使用して、60 分を超える可能性のあるタスクを実行する場合。

URLベースのマッピング

ロードバランサーに、リクエストの に基づいて受信リクエストを特定のサーバーに転送させることもできますURL。たとえば、オンラインコマースアプリケーションをサポートする 10 台のアプリケーションサーバーのグループがあるとします。そのうちの 8 台のサーバーではカタログを処理し、2 台のサーバーでは支払いを処理します。HTTP リクエスト に基づいて、すべての支払い関連のリクエストを支払いサーバーに転送しますURL。この場合、「支払い」または「チェックアウト」を含むすべての URLs を支払いサーバーの 1 つに転送します。

ではHAProxy、 URLベースのマッピングを使用して、指定された文字列を特定のサーバーに URLs含むように に指示できます。AWS OpsWorks スタックで URLベースのマッピングを使用

するには、haproxy組み込みクックブックの haproxy-default.erb テンプレートを上書きしてカスタムHAProxy設定ファイルを作成する必要があります。詳細については、[HAProxy「設定マニュアル」](#)および「」を参照してください[カスタムテンプレートの使用](#)。HTTPS リクエストに URLベースのマッピングを使用することはできません。HTTPS リクエストは暗号化されているため、HAProxyはリクエストを調べる方法がありませんURL。

Elastic Load Balancing では、URLマッピングのサポートが制限されています。詳細については、[「Listener Configurations for Elastic Load Balancing」](#) (Elastic Load Balancing のリスナー設定) を参照してください。

推奨事項：でのみ処理できる要件がない限り、ロードバランシングには Elastic Load Balancing を使用することをお勧めしますHAProxy。この場合、受信トラフィックを一連のHAProxyサーバーに分散するフロントエンドのロードバランサーとして Elastic Load Balancing を使用することで、この2つを組み合わせるのが最善の方法です。これを実行するには：

- 各スタックのアベイラビリティゾーンにHAProxyインスタンスをセットアップして、ゾーンのアプリケーションサーバーにリクエストを分散します。
- HAProxy インスタンスを Elastic Load Balancing ロードバランサーに割り当て、ロードバHAProxyランサーに受信リクエストを分散します。

このアプローチにより、HAProxyの URLベースのマッピングを使用して、さまざまなタイプのリクエストを適切なアプリケーションサーバーに分散できます。ただし、いずれかのHAProxyサーバーがオフラインになっても、Elastic Load Balancing ロードバランサーは受信トラフィックを正常なHAProxyサーバーに自動的に分散するため、サイトは引き続き機能します。Elastic Load Balancing をフロントエンドロードバランサーとして使用する必要があることに注意してください。HAProxyサーバーは他のHAProxyサーバーにリクエストを分散できません。

Chef Server から AWS OpsWorks スタックへの移行

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または[AWS プレミアムサポート](#)を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは Chef に基づいているため、Chef Server から AWS OpsWorks スタックへの移行は比較的簡単です。このトピックでは、AWS OpsWorks スタックで動作するように Chef Server コードを変更するためのガイドラインを示します。

Note

11.10 より前のバージョンの Chef を使用したスタックの移行はお勧めしません。それらのバージョンは chef-solo に基づいており、Chef 検索やデータバッグがサポートされていないためです。

トピック

- [レイヤーへのロールのマッピング](#)
- [データバッグの使用](#)
- [Chef の検索の使用](#)
- [クックブックとレシピの管理](#)
- [Chef 環境の使用](#)

レイヤーへのロールのマッピング

Chef Server では、ロールを使用して、目的と設定が同じインスタンス (Java アプリケーションサーバーをホストするインスタンスのセットなど) を定義および管理します。[AWS OpsWorks スタック レイヤー](#)は基本的に Chef ロールと同じ目的を果たします。レイヤーは、同じ設定、インストールされたパッケージ、アプリケーションのデプロイ手順などを使用して Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのセットを作成するための設計図です。

AWS OpsWorks スタックには、複数のタイプのアプリケーションサーバー、HAProxyロードバランサー、MySQL データベースマスター、Ganglia モニタリングマスター用の[一連の組み込みレイヤー](#)が含まれています。例えば、組み込み [Java App Server](#) (Java アプリケーションサーバー) レイヤーは、Tomcat サーバーをホストするインスタンスを作成するためのブループリントです。

AWS OpsWorks スタックに移行するには、各ロールを同等の機能を提供するレイヤーに関連付ける必要があります。ロールによっては、いずれかの組み込みレイヤーのみ使用できる場合があります。また、さまざまな程度のカスタマイズが必要になる場合があります。まずは、組み込みレイヤーの機能と各レイヤーに関連付けられているレシピを調べることで、最低限必要なロールの機能が用意されているかどうかを確認します。組み込みレイヤーの詳細については、「[レイヤー](#)」と「[AWS](#)

[OpsWorks スタックレイヤーリファレンス](#)」を参照してください。組み込みレシピを確認するには、[AWS OpsWorks スタックパブリック GitHubリポジトリ](#)を参照してください。

この後の作業の進め方は、以下のように、レイヤーが各ロールにどの程度一致するかによって異なります。

組み込みレイヤーがロールのすべての機能をサポートする

必要に応じて若干のカスタマイズにより、組み込みレイヤーを直接使用できます。例えば、ロールが Tomcat サーバーをサポートしている場合、Java アプリケーションサーバーレイヤーのレシピはすでに、若干のカスタマイズにより、そのロールのすべてのタスクを処理するようになっていることがあります。たとえば、レイヤーの組み込みレシピで、カスタム Tomcat または Apache 設定が使用されるように、該当する[属性](#)または[テンプレート](#)を上書きできます。

組み込みレイヤーがロールの一部の機能のみをサポートする

レイヤーの拡張により、[組み込みレイヤー](#)を使用できる場合があります。この拡張には一般的に、不足している機能をサポートするカスタムレシピの実装と、レイヤーのライフサイクルイベントへのそれらのレシピの割り当てが含まれます。たとえば、Tomcat サーバーをホストする同じインスタンスに Redis サーバーをインストールするロールがあるとします。レイヤーのインスタンスに Redis をインストールするカスタムレシピを実装し、そのレシピをレイヤーの Setup イベントに割り当てることで、ロールの機能と一致するように Java アプリケーションサーバーレイヤーを拡張できます。

組み込みレイヤーがロールの機能を適切にサポートしない

カスタムレイヤーを実装します。たとえば、MongoDB データベースサーバーをサポートするロールがあり、この機能がいずれの組み込みレイヤーでもサポートされていないとします。必要なパッケージのインストールやサーバーの設定などを行うレシピを実装し、そのレシピをカスタムレイヤーのライフサイクルイベントに割り当てることで、レイヤーでこの機能がサポートされるようにできます。一般的に、少なくともロールのレシピのいくつかはこの目的に使用できます。カスタムレイヤーを実装する方法の詳細については、「[カスタム Tomcat サーバーレイヤーの作成](#)」を参照してください。

データバッグの使用

Chef Server では、データバッグを使用してユーザー定義のデータをレシピに渡すことができます。

- クックブックにデータを保存すると、Chef によってそのデータは各インスタンスにインストールされます。
- パスワードなどの機密データには、暗号化されたデータバッグを使用できます。

AWS OpsWorks スタックはデータバッグをサポートします。レシピは Chef Server とまったく同じコードを使用してデータを取得できます。ただし、このサポートには以下の制限と違いがあります。

- データバッグは Chef 11.10 Linux 以降のスタックでのみサポートされています。

Chef の以前のバージョンを実行している Windows スタックと Linux スタックでは、データバッグはサポートされていません。

- データバッグはクックブックのリポジトリに保存しません。

代わりに、カスタム を使用してデータバッグのデータJSONを管理します。

- AWS OpsWorks スタックは、暗号化されたデータバッグをサポートしていません。

パスワードや証明書などの機密データを暗号化された形式で保存する必要がある場合は、プライベート S3 バケットに保存することをお勧めします。その後、[Amazon SDK for Ruby](#) を使用してデータを取得するカスタムレシピを作成できます。例については、[SDK for Ruby の使用](#) を参照してください。

詳細については、「[データバッグの使用](#)」を参照してください。

Chef の検索の使用

Chef Server では、IP アドレスやロール設定などのスタック設定情報がサーバーに保存されます。レシピは Chef 検索を使用してこのデータを取得します。AWS OpsWorks スタックでは、多少異なるアプローチを使用します。たとえば、Chef 11.10 Linux スタックは Chef クライアントのローカルモードに基づいています。このモードは Chef クライアントのオプションの 1 つであり、インスタンスでローカルに Chef Server の軽量バージョン (Chef Zero) が実行されます。Chef Zero では、インスタンスのノードオブジェクトに保存されたデータに対する検索がサポートされています。

AWS OpsWorks スタックは、リモートサーバーにスタックデータを保存する代わりに、ライフサイクルイベントごとにスタック [設定とデプロイ属性](#) のセットを各インスタンスのノードオブジェクトに追加します。これらの属性は、スタック設定のスナップショットとなります。Chef Server と同じ構文を使用して、サーバーからのレシピの取得に必要なデータの大部分を定義します。

多くの場合、AWS OpsWorks スタックのレシピの検索依存コードを変更する必要はありません。Chef 検索はスタック設定とデプロイ属性を含むノードオブジェクトで動作するため、スタックの検索クエリは通常 Chef Server AWS OpsWorks とまったく同じように動作します。

主な例外は、スタック設定属性とデプロイ属性に、インスタンスに属性をインストールするときに AWS OpsWorks スタックが認識するデータのみが含まれていることです。特定のインスタンスで属

性をローカルに作成または変更した場合、それらの変更は AWS OpsWorks スタックに伝達されず、他のインスタンスにインストールされているスタック設定およびデプロイ属性に組み込まれません。検索を使用して、そのインスタンス上の属性値のみを取得できます。詳細については、「[Chef の検索の使用](#)」を参照してください。

Chef Server との互換性のために、AWS OpsWorks スタックはノードオブジェクトに一連の `role` 属性を追加します。各属性にはスタックのレイヤー属性の 1 つが含まれます。レシピで検索キーとして `roles` を使用している場合、検索コードを変更する必要はありません。クエリによって、対応するレイヤーのデータが自動的に返されます。たとえば、以下の両方のクエリによってレイヤー `php-app` の属性が返されます。

```
phpserver = search(:node, "layers:php-app").first
```

```
phpserver = search(:node, "roles:php-app").first
```

クックブックとレシピの管理

AWS OpsWorks スタックと Chef Server では、クックブックとレシピの処理が多少異なります。Chef Server の場合:

- すべてのクックブックは独自に実装するかコミュニティクックブックを使用することで用意します。
- クックブックはサーバーに保存します。
- レシピは手動でまたは定期的なスケジュールで実行します。

AWS OpsWorks スタックの場合 :

- AWS OpsWorks スタックには、組み込みレイヤーごとに 1 つ以上のクックブックが用意されています。これらのクックブックによって、組み込みレイヤーのソフトウェアのインストールと設定、アプリケーションのデプロイなど、標準的なタスクが処理されます。

組み込みクックブックによって実行されないタスクを処理するには、カスタムクックブックをスタックに追加するか、またはコミュニティクックブックを使用します。

- AWS OpsWorks スタッククックブックは、S3 バケットや Git リポジトリなどのリモートリポジトリに保存します。

詳細については、「[クックブックの保存](#)」を参照してください。

- [レシピは手動で実行](#)できますが、通常は AWS OpsWorks インスタンスの[ライフサイクル中に重要なポイントで発生する一連のライフサイクルイベント](#)に応じて、スタックでレシピを実行します。

詳細については、「[レシピの実行](#)」を参照してください。

- AWS OpsWorks スタックは、Chef 11.10 スタックでのみ Berkshelf をサポートします。Berkshelf を使用してクックブックの依存関係を管理する場合、Chef 11.4 以前のバージョンを実行しているスタックを使用することはできません。

詳細については、「[Berkshelf の使用](#)」を参照してください。

トピック

- [クックブックの保存](#)
- [レシピの実行](#)

クックブックの保存

Chef Server では、クックブックをサーバーに保存し、サーバーからインスタンスにデプロイします。AWS OpsWorks スタックでは、S3 または HTTP アーカイブ、Git または Subversion リポジトリなどのリポジトリにクックブックを保存します。[クックブックをインストールする](#)ときに AWS OpsWorks、スタックがリポジトリからスタックのインスタンスにコードをダウンロードするために必要な情報を指定します。

Chef Server から移行するには、これらのリポジトリのいずれかにクックブックを配置する必要があります。クックブックのリポジトリを構築する方法については、「[クックブックリポジトリ](#)」を参照してください。

レシピの実行

AWS OpsWorks スタックでは、各レイヤーにセットアップ、設定、デプロイ、デプロイ解除、シャットダウンという一連の[ライフサイクルイベント](#)があり、それぞれがインスタンスのライフサイクル中のキーポイントで発生します。カスタムレシピを実行するには、一般的に、そのレシピを該当するレイヤーのイベントに割り当てます。そのイベントが発生すると、割り当てられたレシピが AWS OpsWorks スタックによって実行されます。たとえば、インスタンスの起動の完了後に Setup イベントが発生するため、一般的にこのイベントに、パッケージのインストールと設定やサービスの開始などのタスクを実行するレシピを割り当てます。

[Execute Recipes スタックコマンド](#)を使用して、レシピを手動で実行できます。このコマンドは開発やテストに便利ですが、このコマンドを使用して、ライフサイクルイベントに割り当てられていないレシピを実行することもできます。また、Execute Recipes コマンドを使用して、Setup および Configure イベントを手動でトリガーすることもできます。

AWS OpsWorks スタックコンソールに加えて、[AWSCLI](#)または [SDKs](#) を使用してレシピを実行できます。これらのツールはすべての [AWS OpsWorks スタックAPIアクション](#)をサポートしますが、よりも簡単に使用できますAPI。 [create-deployment](#) CLI コマンドを使用して、関連するすべてのレシピを実行するライフサイクルイベントをトリガーします。このコマンドを使用して、イベントをトリガーすることなく、1つ以上のレシピを実行することもできます。同等のSDKコードは特定の言語によって異なりますが、一般的に CLI コマンドに似ています。

次の例では、create-deployment CLI コマンドを使用してアプリケーションのデプロイを自動化する2つの方法について説明します。

- 1つのインスタンスが属するカスタムレイヤーをスタックに追加することで、アプリケーションを定期的にデプロイします。

インスタンスで cron ジョブを作成して、指定したスケジュールでそのコマンドを実行する、カスタム Setup レシピをレイヤーに追加します。cron ジョブを作成するレシピの使用法の例については、「[Linux インスタンスでの cron ジョブの実行](#)」を参照してください。

- create-deployment CLI コマンドを使用してアプリケーションをデプロイするタスクを継続的統合パイプラインに追加します。

Chef 環境の使用

AWS OpsWorks スタックは Chef 環境をサポートしていません。node.chef_environmentは常にを返します_default。

AWS OpsWorks スタックレイヤーリファレンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックがデプロイするすべてのインスタンスは、スタック内のインスタンスのロールを定義し、インスタンスのセットアップと設定、パッケージのインストール、アプリケーションのデプロイなどの詳細を制御する、少なくとも1つのレイヤーのメンバーである必要があります。AWS OpsWorks スタックを使用してレイヤーを作成および管理する方法の詳細については、「」を参照してください[レイヤー](#)。

各レイヤーの説明には、レイヤーのライフサイクルイベントごとに AWS OpsWorks スタックが実行する組み込みレシピのリストが含まれます。これらのレシピは <https://github.com/aws/opsworks-cookbooks> に保存されます。リストには、AWS OpsWorks スタックによって直接実行されるレシピのみが含まれていることに注意してください。それらのレシピは、依存レシピを実行する場合があります。それら依存レシピはリストされていません。依存レシピやカスタムレシピを含む特定のイベントのレシピの完全なリストを参照するには、適切な[ライフサイクルイベントの Chef ログ](#)の実行リストを確認してください。

トピック

- [HAProxy Layer リファレンス](#)
- [HAProxy AWS OpsWorks スタックレイヤー](#)
- [MySQLレイヤーリファレンス](#)
- [MySQL OpsWorks レイヤー](#)
- [アプリケーションサーバーレイヤーリファレンス](#)
- [アプリケーションサーバーレイヤー](#)
- [ECS クラスターレイヤーリファレンス](#)
- [カスタムレイヤーリファレンス](#)
- [その他のレイヤーリファレンス](#)
- [その他のレイヤー](#)

HAProxy Layer リファレンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

HAProxy レイヤーは [HAProxy](#)、信頼性の高い高性能 TCP/HTTP ロードバランサーである を使用して、TCP および HTTP ベースのアプリケーションに高可用性のロードバランシングとプロキシサービスを提供します。永続性やレイヤー7 処理を必要とする一方で、非常に高い負荷でクローलする必要があるウェブサイト特に役立ちます。

HAProxy はトラフィックをモニタリングし、関連するインスタンスの統計と正常性をウェブページに表示します。デフォルトでは、URI は `http://DNSName/haproxy?stats` で、*DNSName* は HAProxy インスタンス DNS の名前です。

Short name: lb

Compatibility: HAProxy Layer は、custom、db-master、および memcached の Layer と互換性があります。

オープンポート：ポート 22 (SSH)、80 ()、および 443 (HTTP) へのパブリックアクセス HAProxy を許可します HTTPS。

Autoassign Elastic IP addresses: デフォルトで On

デフォルト EBS ボリューム：なし

デフォルトのセキュリティグループ：AWS-OpsWorks-LB-Server

Configuration: HAProxy Layer を設定するには、次を指定する必要があります。

- ヘルスチェック URI (デフォルト: `http://DNSName/`)。
- 統計 URI (デフォルト: `http://DNSName/haproxy?stats`)。
- 統計パスワード (オプション)
- ヘルスチェック方法 (オプション) デフォルトでは、は `HTTPOPTIONS` メソッド HAProxy を使用します。GET または を指定することもできます HEAD。
- 統計の有効化 (オプション)
- ポート。デフォルトでは、AWS OpsWorks スタックは HTTP と の両方の HTTPS トラフィックを処理する HAProxy ように を設定します。Chef 設定 [テンプレート](#) を上書きすることで、いずれか一方のみを処理する HAProxy ように を設定できます `haproxy.cfg.erb`。

Setup recipes:

- `opsworks_initial_setup`
- `ssh_host_keys`
- `ssh_users`
- `mysql::client`
- `dependencies`
- `ebs`
- `opsworks_ganglia::client`
- `haproxy`

Configure recipes:

- `opsworks_ganglia::configure-client`
- `ssh_users`
- `agent_version`
- `haproxy::configure`

Deploy recipes:

- `deploy::default`
- `haproxy::configure`

Shutdown recipes:

- `opsworks_shutdown::default`
- `haproxy::stop`

インストール:

- AWS OpsWorks スタックは、インスタンスのパッケージインストーラを使用して、デフォルトの場所に をインストールHAProxyします。
- 指定した場所にログファイルを保存するように `syslog` を設定する必要があります。詳細については、「[HAProxy](#)」を参照してください。

HAProxy AWS OpsWorks スタックレイヤー

Note

このレイヤーは、Chef 11 以前の Linux ベースのスタックでのみ使用できます。

AWS OpsWorks スタックHAProxyレイヤーは、AWS OpsWorks [HAProxy](#)サーバーをホストするインスタンスの設計図を提供するスタックレイヤーです。信頼性の高い高性能 TCP/HTTP 負荷分散です。通常は、1つのスモールインスタンスで、すべてのアプリケーションサーバートラフィックを十分に処理できます。

Note

スタックは1つのリージョンに制限されます。アプリケーションを複数のリージョンに分散させるには、リージョンごとに個別のスタックを作成する必要があります。

HAProxy Layer を作成するには

1. ナビゲーションペインで、[Layers] (レイヤー) をクリックします。
2. [Layers] (レイヤー) ページで [Add a Layer] (レイヤーの追加) または [+ Layer] (+ レイヤー) をクリックします。レイヤータイプで、 を選択しますHAProxy。

レイヤーには、次の設定があります。すべてがオプションです。

HAProxy 統計

レイヤーが統計を収集および表示するかどうか。デフォルト値は [Yes] です。

統計 URL

統計ページのURLパス。完全な URLは http:// です。ここで *DNSNameStatisticsPath*、*DNSName*は関連付けられたインスタンスDNSの名前です。 *StatisticsPath* デフォルト値は /haproxy?stats で、http://ec2-54-245-151-7.us-west-2.compute.amazonaws.com/haproxy?stats. などに対応します。

Statistics user name

統計ページのユーザー名。統計ページを表示するために指定する必要があります。デフォルト値は、「opsWorks」です。

Statistics password

統計ページのパスワード。統計ページを表示するには、指定する必要があります。デフォルト値はランダムに生成される文字列です。

ヘルスチェック URL

ヘルスチェックURLのサフィックス。はURLこれHAProxyを使用して、各アプリケーションサーバーインスタンスでHTTPメソッドを定期的に呼び出し、インスタンスが機能しているかどうかを判断します。ヘルスチェックが失敗した場合、は手動で、または[自動ヒーリング](#)によって再起動されるまで、インスタンスへのトラフィックのルーティングHAProxyを停止します。URL サフィックスのデフォルト値は「/」で、サーバーインスタンスのホームページ `http://DNSName/` に対応します。

Health check method

インスタンスが機能しているかどうかを確認するために使用されるHTTP方法。デフォルト値はOPTIONSで、GETまたはを指定することもできますHEAD。詳細については、「[httpchk](#)」を参照してください。

Custom security groups

この設定は、組み込み AWS OpsWorks の スタックセキュリティグループをレイヤーに自動的に関連付けないことを選択した場合に表示されます。レイヤーに関連付けるセキュリティグループを指定する必要があります。グループが正しく設定され、レイヤー間のトラフィックが許可されていることを確認します。詳細については、「[新しいスタックを作成する](#)」を参照してください。

Add layer

 OpsWorks ECS RDS**Layer type**

HAProxy ▾

An HAProxy layer is a blueprint for instances that expose a single IP address to represent a set of application servers. It receives incoming requests, distributes them across the application server instances, and returns responses to the caller. [Learn more.](#)

HAProxy statisticsYes

Statistics URL

/haproxy?stats

Statistics user name

opsworks

Statistics password

dzrfl9y66r

Health check URL

/

Health check method

OPTIONS ▾

Need further support? [Let us know.](#)

Cancel

Add layer**Note**

後で使用するためにパスワードを記録します。AWS OpsWorks スタックでは、レイヤーの作成後にパスワードを表示することはできません。ただし、レイヤーの [Edit] (編集) ページにアクセスし、[General Settings] (一般設定) タブの [Update password] (パスワードパスワードの更新) をクリックすることによって、パスワードを更新することはできます。

Layer HAProxy

General Settings Recipes Network EBS Volumes Security

Settings

HAProxy statistics Yes

Statistics URL

Statistics user name

Statistics password [Update password](#)

Health check URL

Health check method

Instance shutdown timeout

Auto healing enabled Yes

Custom JSON

Enter custom JSON that is passed to your Chef recipes for all instances in this layer. You can use this to override and customize built-in recipes or pass variables to your own recipes. [Learn more.](#)

Cancel

HAProxy Layer のしくみ

デフォルトでは、は以下HAProxyを実行します。

- HTTP および HTTPS ポートでリクエストをリッスンします。

Chef 設定テンプレート を上書きすることで、HTTPまたは HTTPS ポートのみをリッスン HAProxyするように を設定できますhaproxy.cfg.erb。

- 任意のアプリケーションサーバーレイヤーのメンバーであるインスタンスに受信トラフィックをルーティングします。

デフォルトでは、AWS OpsWorks スタックは任意のアプリケーションサーバーレイヤーのメンバーであるインスタンスにトラフィックを分散HAProxyするように を設定します。例えば、Rails App Server レイヤーと PHP App Server レイヤーの両方を持つスタックがあるとします。HAProxyマスターは両方のレイヤーのインスタンスにトラフィックを分散します。カスタムレシピを使用すると、デフォルトルーティングを設定できます。

- 複数のアベイラビリティゾーンにトラフィックをルーティングします。

1つのアベイラビリティゾーンがダウンした場合、ロードバランサーは他のゾーン内のインスタンスに受信トラフィックをルーティングするので、アプリケーションは中断せずに実行できます。そのため、推奨される方法は、複数のアベイラビリティゾーンにアプリケーションサーバーを分散することです。

- 指定されたヘルスチェックメソッドを各アプリケーションサーバーインスタンスに対して定期的に行い、正常性を評価します。

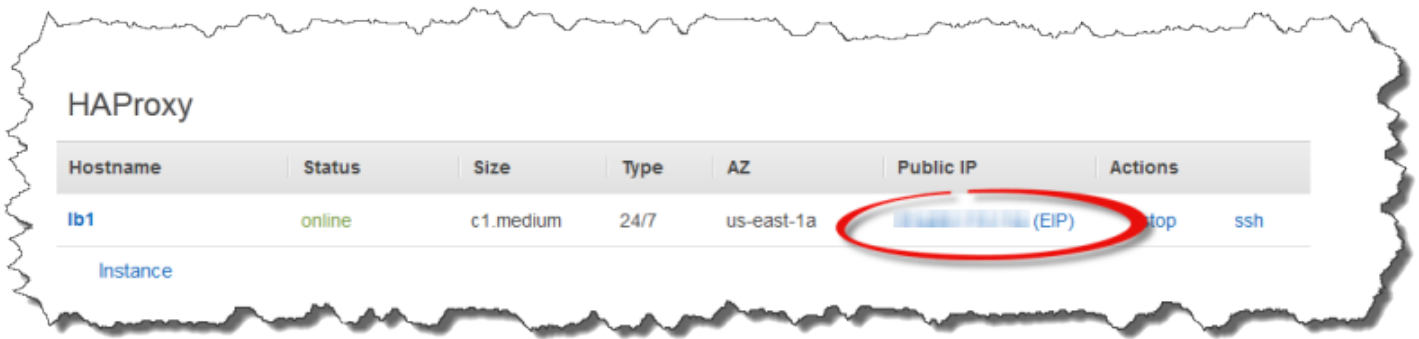
メソッドが指定されたタイムアウト期間内に返らない場合、インスタンスは失敗したと見なされ、インスタンスへのリクエストのルーティングをHAProxy停止します。AWS OpsWorks スタックは、失敗したインスタンスを自動的に置き換える方法も提供します。詳細については、「[自動ヒーリングの使用](#)」を参照してください。レイヤーを作成する際に、ヘルスチェックメソッドを変更できます。

- 統計情報を収集し、オプションでウェブページに表示します。

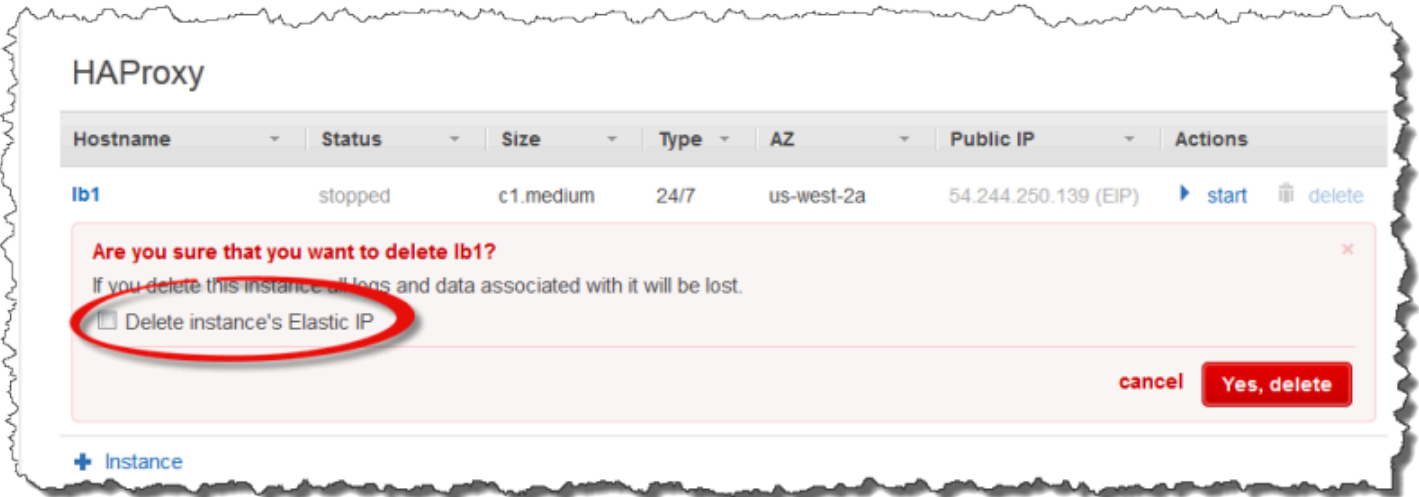
Important

ヘルスチェックをデフォルトの OPTIONSメソッドで正しく動作させるには、アプリが 2xx または 3xx ステータスコードを返す必要があります。

デフォルトでは、HAProxyレイヤーにインスタンスを追加すると、AWS OpsWorks Stacks によってアプリケーションを表す Elastic IP アドレスが割り当てられます。このアドレスは世界に公開されています。HAProxy インスタンスの Elastic IP アドレスはアプリケーションの唯一の公開であるためURL、基盤となるアプリケーションサーバーインスタンスのパブリックドメイン名を作成および管理する必要はありません。次の図に示すように、[Instances] ページに移動し、インスタンスのパブリック IP アドレスを調べることで、アドレスを取得することもできます。(EIP) の後に続くアドレスは、Elastic IP アドレスです。Elastic IP アドレスの詳細については、「[Elastic IP アドレス \(EIP\)](#)」を参照してください。



HAProxy インスタンスを停止すると、AWS OpsWorks スタックは Elastic IP アドレスを保持し、再起動時にインスタンスに再割り当てします。HAProxy インスタンスを削除すると、デフォルトでは、AWS OpsWorks スタックはそのインスタンスの IP アドレスを削除します。アドレスを保持するには、次の図に示されているように、[Delete instance's Elastic IP] オプションをオフにします。



このオプションは、削除されたインスタンスと置き換えるために新しいインスタンスをレイヤーに追加したときに何が起こるかに影響します。

- 削除されたインスタンスの Elastic IP アドレスを保持した場合、AWS OpsWorks スタックはそのアドレスを新しいインスタンスに割り当てます。
- それ以外の場合、AWS OpsWorks スタックは新しい Elastic IP アドレスをインスタンスに割り当てます。新しいアドレスにマッピングするようにDNSレジストラ設定を更新する必要があります。

アプリケーションサーバーインスタンスがオンラインになったり、オフラインになったりした場合 (手動で、または [automatic scaling](#) (自動スケーリング) または [auto healing](#) (自動ヒーリング) の結果として)トラフィックを現在のオンラインインスタンスのセットにルーティングするには、ロードバ

ランサー設定を更新する必要があります。このタスクは、レイヤーの組み込みレシピによって自動的に処理されます。

- 新しいインスタスがオンラインになると、AWS OpsWorks スタックは Configure [ライフサイクルイベント](#) をトリガーします。HAProxy Layer の組み込み Configure レシピは、リクエストを任意の新しいアプリケーションサーバーインスタンスにも分散するようにロードバランサー設定を更新します。
- インスタスがオフラインになるか、インスタスがヘルスチェックに失敗すると、AWS OpsWorks スタックは Configure ライフサイクルイベントもトリガーします。HAProxy Configure レシピは、ロードバランサーの設定を更新して、残りのオンラインインスタンスにのみトラフィックをルーティングします。

最後に、HAProxy Layer でカスタムドメインを使用することもできます。詳細については、「[カスタムドメインの使用](#)」を参照してください。

統計ページ

統計ページを有効にしている場合、は、指定された `de` さまざまなメトリクスを含むページHAProxy を表示しますURL。

HAProxy 統計を表示するには

1. HAProxy インスタスの詳細ページからインスタスのパブリックDNS名を取得し、コピーします。
2. [Layers] ページの [HAProxy] をクリックし、その Layer の詳細ページを開きます。
3. レイヤーの詳細URLから統計を取得し、公開DNS名に追加します。例えば、`http://ec2-54-245-102-172.us-west-2.compute.amazonaws.com/haproxy?stats` とします。
4. 前のステップURLの `de` をブラウザに貼り付け、レイヤーの作成時に指定したユーザー名とパスワードを使用して統計ページを開きます。

HAProxy version 1.4.22, released 2012/08/09

Statistics Report for pid 2468

> General process information

pid = 2468 (process #1, nbproc = 1)
 uptime = 0d 2h48m51s
 system limits: memmax = unlimited; ulimit-n = 160013
 maxsock = 160013; maxconn = 80000; maxpipes = 0
 current conns = 1; current pipes = 0/0
 Running tasks: 1/2

■ active UP ■ backup UP
■ active UP, going down ■ backup UP, going down
■ active DOWN, going up ■ backup DOWN, going up
■ active or backup DOWN ■ not checked
■ active or backup DOWN for maintenance (M)

Note: UP with load-balancing disabled is reported as DOWN

application	Queue			Session rate			Sessions				Bytes		Denied		Errors			
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp
Frontend				1	1	-	1	1	80 000	2		335	262	0	0	0		
localhost	0	0	-	0	0		0	0	5	0	0	0	0	0	0	0		
Backend	0	0		0	0		0	0	80 000	0	0	335	262	0	0			0

MySQLレイヤーリファレンス

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

ℹ Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

MySQL Layer は SQL、広く使用されているリレーショナルデータベース管理システムである MySQL をサポートしています。AWS OpsWorks スタックは、オペレーティングシステムに応じて利用可能な最新バージョンをインストールします。MySQL インスタンスを追加すると、必要なアクセス情報がアプリケーションサーバーレイヤーに提供されます。マスター-マスターまたはマスター-スレーブ構成をセットアップするにはカスタム Chef レシピを書き込む必要があります。

Short name: db-master

互換性 : MySQL レイヤーは、custom、lb、memcached、monitoring-master、nodejs-app、php-app、rails-app、web の各レイヤーと互換性があります。

オープンポート : MySQL レイヤーは、スタックのウェブサーバー、カスタムサーバー、Rails、Node.js アプリケーションサーバーからポート 22 (SSH) PHPとすべてのポートへのパブリックアクセスを許可します。

Autoassign Elastic IP addresses: デフォルトで Off

デフォルトEBSボリューム : はい、で /vol/mysql

デフォルトのセキュリティグループ : AWS-OpsWorks-DB-Master-Server

設定 : MySQL レイヤーを設定するには、以下を指定する必要があります。

- root ユーザーのパスワード
- SQL エンジン

Setup recipes:

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- mysql::server
- dependencies
- deploy::mysql

Configure recipes:

- opsworks_ganglia::configure-client
- ssh_users

- agent_version
- deploy::mysql

Deploy recipes:

- deploy::default
- deploy::mysql

Shutdown recipes:

- opsworks_shutdown::default
- mysql::stop

インストール:

- AWS OpsWorks スタックはインスタンスのパッケージインストーラを使用して、MySQL とそのログファイルをデフォルトの場所にインストールします。詳細については、[「MySQLドキュメント」](#)を参照してください。

MySQL OpsWorks レイヤー

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Chef 11 以前の Linux ベースのスタックでのみ使用できます。

My SQL OpsWorks レイヤーは、[MySQL](#) データベースマスターとして機能する Amazon EC2 インスタンスの設計図を提供します。アプリケーションサーバーレイヤーにデプロイされた各アプリケーションには、組み込みのレシピによってデータベースが作成されます。例えば、PHP アプリケーション「myapp」をデプロイすると、レシピは「myapp」データベースを作成します。

MySQL Layer には次の構成設定があります。

SQL ルートユーザーのパスワード

(必須) root ユーザーのパスワード。

Set root user password on every instance

(オプション) root ユーザーのパスワードが、スタックの各インスタンスにインストールされているスタック設定およびデプロイ属性に含まれているかどうか。デフォルトの設定は、[Yes] です。

この値を No に設定すると、AWS OpsWorks スタックはルートパスワードをアプリケーションサーバーインスタンスにのみ渡します。

Custom security groups

(オプション) レイヤーに関連付けるカスタムセキュリティグループ。詳細については、「[新しいスタックを作成する](#)」を参照してください。

Add layer

OpsWorks ECS RDS

Layer type

A MySQL Master layer is a blueprint for instances that function as MySQL relational database servers. [Learn more.](#)

MySQL root user password

Set root user password on every instance Yes

Need further support? [Let us know.](#)

Cancel Add layer

レイヤーに 1 つ以上のインスタンスを追加できます。各インスタンスは個別の MySQL データベースマスターを表します。その後、[インスタンスをアプリケーションにアタッチ](#)すると、そのアプリケーションサーバーに必要な接続情報がインストールされます。アプリケーションは、この接続情報を使用して、[インスタンスのデータベースサーバーに接続](#)することができます。

アプリケーションサーバーレイヤーリファレンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは、複数の異なるアプリケーションサーバーと静的ウェブページサーバーをサポートしています。

トピック

- [AWS フロー \(Ruby\) レイヤーリファレンス](#)
- [Java アプリケーションサーバーレイヤーリファレンス](#)
- [Node.js アプリケーションサーバーレイヤーリファレンス](#)
- [PHP アプリケーションサーバーレイヤーリファレンス](#)
- [Rails アプリケーションサーバーレイヤーリファレンス](#)
- [静的 ウェブサーバーレイヤーリファレンス](#)

AWS フロー (Ruby) レイヤーリファレンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

AWS フロー (Ruby) レイヤーは、Amazon Simple Workflow Service アクティビティとワークフローワーカーをホストするインスタンスの設計図を提供します。

短縮名 : aws-flow-ruby

互換性 : AWSフロー (Ruby) レイヤーは、PHPApp Server、MySQL、Memcached、Ganglia、およびカスタムレイヤーと互換性があります。

[Open ports:] なし。

IAM role: aws-opsworks-ec2- role-with-swf は、リクエストに応じて AWS OpsWorks スタックが作成する標準のAWSフロー (Ruby) ロールです。

Autoassign Elastic IP addresses: デフォルトで Off

デフォルトEBSボリューム : いいえ

デフォルトのセキュリティグループ : AWS-OpsWorks-AWS-Flow-Ruby-Server

Setup recipes:

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_aws_flow_ruby::setup

Configure recipes:

- opsworks_ganglia::configure-client
- ssh_users

- `mysql::client`
- `agent_version`
- `opsworks_aws_flow_ruby::configure`

Deploy recipes:

- `deploy::default`
- `デプロイ::aws-flow-ruby`

Undeploy recipes:

- `デプロイ::aws-flow-ruby-undeploy`

Shutdown recipes:

- `opsworks_shutdown::default`

Java アプリケーションサーバーレイヤーリファレンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

Java アプリケーションサーバーレイヤーは、[Apache Tomcat 7.0](#) アプリケーションサーバーをサポートしています。

Short name: `java-app`

Compatibility: (互換性) Java アプリケーションサーバーレイヤーは、custom、db-master および memcached のレイヤーと互換性があります。

オープンポート : Java アプリケーションサーバーレイヤーは、ロードバランサーからポート 22 (SSH)、80 (HTTP)、443 (HTTPS)、およびすべてのポートへのパブリックアクセスを許可します。

Autoassign Elastic IP addresses: デフォルトで Off

デフォルトEBSボリューム : なし

デフォルトのセキュリティグループ : AWS-OpsWorks-Java-App-Server

Setup recipes:

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_java::setup

Configure recipes:

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- opsworks_java::configure

Deploy recipes:

- deploy::default
- deploy::java

Undeploy recipes:

- `deploy::java-undeploy`

Shutdown recipes:

- `opsworks_shutdown::default`
- `deploy::java-stop`

インストール:

- Tomcat を `/usr/share/tomcat7` にインストールします。
- ログファイルの生成方法の詳細については、「[Tomcat にログインする](#)」を参照してください。

Node.js アプリケーションサーバーレイヤーリファレンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

Node.js アプリケーションサーバーレイヤーは [\[Node.js\]](#) アプリケーションサーバーをサポートします。これは、拡張性の高いネットワークアプリケーションサーバーを実装するプラットフォームです。プログラムはイベント駆動型の非同期 I/O を使用して書き込まれ JavaScript、オーバーヘッドを最小限に抑え、スケーラビリティを最大化します。

Short name: `nodejs-app`

Compatibility: (互換性) Node.js アプリケーションサーバーレイヤーは、`custom`、`db-master`、`memcached` および `monitoring-master` のレイヤーと互換性があります。

オープンポート : Node.js アプリケーションサーバーレイヤーは、ロードバランサーからポート 22 (SSH)、80 (HTTP)、443 (HTTPS)、およびすべてのポートへのパブリックアクセスを許可します。

Autoassign Elastic IP addresses: デフォルトで Off

デフォルトEBSボリューム : なし

デフォルトのセキュリティグループ : AWS-OpsWorks-nodejs-App-Server

Setup recipes:

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_nodejs
- opsworks_nodejs::npm

Configure recipes:

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- opsworks_nodejs::configure

Deploy recipes:

- deploy::default
- opsworks_nodejs
- opsworks_nodejs::npm

- `deploy::nodejs`

Undeploy recipes:

- `deploy::nodejs-undeploy`

Shutdown recipes:

- `opsworks_shutdown::default`
- `deploy::nodejs-stop`

インストール:

- Node.js を `/usr/local/bin/node` にインストールします。
- ログファイルの生成方法の詳細については、Nodejitsu ウェブサイトの「[How to log in node.js](#)」を参照してください。

Node.js application configuration:

- Node.js によって実行される主ファイルは、`server.js` という名前で、デプロイされたアプリケーションのルートディレクトリにあります。
- Node.js アプリケーションは、ポート 80 (または該当する場合、ポート 443) でリッスンするように設定する必要があります。

Note

Express を実行する Node.js のアプリケーションは、一般的に次のコードを使用してリッスンするポートを設定します。ここで、`process.env.PORT` はデフォルトポートを表し、80 に解決します。

```
app.set('port', process.env.PORT || 3000);
```

AWS OpsWorks スタックでは、次のようにポート 80 を明示的に指定する必要があります。

```
app.set('port', 80);
```

PHP アプリケーションサーバーレイヤーリファレンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

PHP App Server レイヤーは、mod_php で [Apache2](#) を使用して PHP アプリケーションサーバーをサポートします。

Short name: php-app

互換性： PHP App Server レイヤーは、custom、db-master、memcached、monitoring-master、および rails-app のレイヤーと互換性があります。

オープンポート： PHP アプリケーションサーバーレイヤーは、ロードバランサーからポート 22 (SSH)、80 (HTTP)、443 (HTTPS)、およびすべてのポートへのパブリックアクセスを許可します。

Autoassign Elastic IP addresses: デフォルトで Off

デフォルトEBSボリューム： いいえ

デフォルトのセキュリティグループ： AWS-OpsWorks-PHP-App-Server

Setup recipes:

- opsworks_initial_setup
- ssh_host_keys
- ssh_users

- `mysql::client`
- `dependencies`
- `ebs`
- `opsworks_ganglia::client`
- `mysql::client`
- `dependencies`
- `mod_php5_apache2`

Configure recipes:

- `opsworks_ganglia::configure-client`
- `ssh_users`
- `agent_version`
- `mod_php5_apache2::php`
- `php::configure`

Deploy recipes:

- `deploy::default`
- `deploy::php`

Undeploy recipes:

- `deploy::php-undeploy`

Shutdown recipes:

- `opsworks_shutdown::default`
- `apache2::stop`

インストール:

- AWS OpsWorks スタックは、インスタンスのパッケージインストーラを使用し、`Apache2`、`mod_php`、および関連するログファイルをデフォルトの場所にインストールしま

す。インストールの詳細については、「[Apache](#)」を参照してください。ログ記録の詳細については、「[ログファイル](#)」を参照してください。

Rails アプリケーションサーバーレイヤーリファレンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

Rails アプリケーションサーバーレイヤーは [Ruby on Rails](#) アプリケーションサーバーをサポートします。

Short name: rails-app

Compatibility: (互換性) Rails アプリケーションサーバーレイヤーは、custom、db-master、memcached、monitoring-master および php-app のレイヤーと互換性があります。

ポート: Rails App Server レイヤーは、ロードバランサーからポート 22 (SSH)、80 (HTTP)、443 (HTTPS)、およびすべてのポートへのパブリックアクセスを許可します。

Autoassign Elastic IP addresses: デフォルトで Off

デフォルトEBSボリューム: なし

デフォルトのセキュリティグループ: AWS-OpsWorks-Rails-App-Server

Configuration: (構成) Rails アプリケーションサーバーレイヤーを構成するには、以下を指定する必要があります。

- Ruby バージョン
- Rails スタック
- Rubygems バージョン
- [Bundler](#) をインストールおよび管理するかどうか
- Bundler バージョン

Setup recipes:

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- apache2 apache2::mod_deflate
- passenger_apache2
- passenger_apache2::mod_rails
- passenger_apache2::rails

Configure recipes:

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- rails::configure

Deploy recipes:

- deploy::default
- deploy::rails

Undeploy recipes:

- `deploy::rails-undeploy`

Shutdown recipes:

- `opsworks_shutdown::default`
- `apache2::stop`

インストール:

- AWS OpsWorks スタックはインスタンスのパッケージインストーラを使用し、`mod_passenger`、`mod_rails`、および関連するログファイルを含む Apache2 をデフォルトの場所にインストールします。インストールの詳細については、「[Phusion Passenger](#)」を参照してください。ログ記録の詳細については、「[ログファイル](#)」を参照してください。

静的 ウェブサーバーレイヤーリファレンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

静的ウェブサーバーレイヤーは静的HTMLページを提供します。これには、などのクライアント側のコードを含めることができます JavaScript。これは、オープンソースの、リバースプロキシHTTP、メールプロキシサーバーである [Nginx](#) に基づいています。

Short name: `web`

Compatibility: (互換性) 静的ウェブサーバーレイヤーは、custom、db-master、および memcached のレイヤーと互換性があります。

オープンポート：静的ウェブサーバーレイヤーは、ロードバランサーからポート 22 (SSH)、80 (HTTP)、443 (HTTPS)、およびすべてのポートへのパブリックアクセスを許可します。

Autoassign Elastic IP addresses: デフォルトで Off

デフォルトEBSボリューム：いいえ

デフォルトのセキュリティグループ：AWS-OpsWorks-Web-Server

Setup recipes:

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- nginx

Configure recipes:

- opsworks_ganglia::configure-client
- ssh_users
- agent_version

Deploy recipes:

- deploy::default
- deploy::web

Undeploy recipes:

- deploy::web-undeploy

Shutdown recipes:

- `opsworks_shutdown::default`
- `nginx::stop`

インストール:

- Nginx を `/usr/sbin/nginx` にインストールします。
- Nginx ログファイルは `/var/log/nginx` に保存されます。

アプリケーションサーバーレイヤー

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

これらのレイヤーは、Chef 11 以前の Linux ベースのスタックでのみ使用できます。

AWS OpsWorks スタックは、「application」に静的ウェブページが含まれている複数の異なるアプリケーションサーバーをサポートしています。サーバーのタイプごとに個別の AWS OpsWorks スタックレイヤーがあり、アプリケーションサーバーおよび関連するパッケージを各レイヤーのインスタンスにインストールしたり、アプリケーションをデプロイしたりする処理を行う組み込みレシピがあります。たとえば、Java App Server レイヤーは、Apache、Tomcat、Open などの複数のパッケージをインストールし、Java アプリケーションをレイヤーの各インスタンスにデプロイします。

アプリケーションサーバーレイヤーを使用するための基本的な手順は次のとおりです。

1. 使用可能な [アプリケーションサーバーレイヤータイプ](#) のうちの 1 つを作成します。

2. レイヤーに [1つ以上のインスタンスを追加](#)します。
3. アプリケーションを作成し、インスタンスにデプロイします。詳細については、「[アプリケーション](#)」を参照してください。
4. (オプション) レイヤーに複数のインスタンスがある場合は、受信トラフィックを各インスタンスに分散するロードバランサーを追加できます。詳細については、「[HAProxy AWS OpsWorks スタックレイヤー](#)」を参照してください。

トピック

- [AWS フロー \(Ruby\) レイヤー](#)
- [Java アプリケーションサーバー AWS OpsWorks スタックレイヤー](#)
- [Node.js アプリケーションサーバー AWS OpsWorks スタックレイヤー](#)
- [PHP アプリケーションサーバー AWS OpsWorks スタックレイヤー](#)
- [Rails App Server AWS OpsWorks スタックレイヤー](#)
- [静的ウェブサーバー AWS OpsWorks スタックレイヤー](#)

AWS フロー (Ruby) レイヤー

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

AWS フロー (Ruby) レイヤーは、[Amazon SWF](#) AWS OpsWorks アクティビティワーカーとワークフローワーカーをホストするインスタンスの設計図を提供する スタックレイヤーです。ワーカーは、[AWSFlow Framework for Ruby](#) を使用して実装されます。これは、Amazon のすべての利点を提供しながら、分散非同期アプリケーションの実装プロセスを簡素化するプログラミングフレームワー

クですSWF。このフレームワークは、業務プロセス、メディアエンコーディング、長期タスク、バックグラウンド処理など、幅広いシナリオに対応するアプリケーションの実装に適しています。

AWS フロー (Ruby) レイヤーには、次の設定が含まれています。

RubyGems バージョン

フレームワークの Gem のバージョン。

[Bundler version (Bundler のバージョン)]

[Bundler](#) のバージョン。

EC2 インスタンスプロファイル

レイヤーのEC2インスタンスで使用されるユーザー定義の Amazon インスタンスプロファイル。このプロファイルは、レイヤーのインスタンスで実行されているアプリケーションが Amazon にアクセスするためのアクセス許可を付与する必要がありますSWF。

アカウントに適切なプロファイルがない場合は、スタックが AWS OpsWorks のプロファイルを更新するSWFアクセス権を持つ新しいプロファイルを選択するか、[IAMコンソール](#)を使用して自分で更新できます。その後、更新されたプロファイルを後続のすべてのAWSフローレイヤーに使用できます。次に、IAM コンソールを使用してプロファイルを作成する方法を簡単に説明します。詳細については、「[Amazon Simple Workflow ServiceのID とアクセスの管理](#)」を参照してください。

AWS フロー (Ruby) インスタンスのプロファイルの作成

1. <https://console.aws.amazon.com/iam/> で IAM コンソールを開きます。
2. ナビゲーションペインの **ポリシー** を選択し、**ポリシーを作成** ポリシーをクリックして、新しいカスタマー管理ポリシーを作成します。
3. [Service (サービス)] で、[SWF] を選択します。
4. アクション で、すべてのSWFアクション (swf:*) を選択します。
5. Amazon リソースネーム (ARN) には、ワーカーがアクセスできる Amazon SWFドメインARN を指定する を入力します。「**All resources**」を選択して、すべてのドメインへのアクセスを許可します。
6. [Next (次へ)] を選択します。
7. オプションで、ポリシーを識別するタグを入力します。
8. [Next (次へ)] を選択します。
9. 完了したら、**ポリシーの作成** を選択します。

10. ナビゲーションペインで [ロール] を選択した後、[ロールの作成] を選択します。
11. ロール名を指定して、次のステップを選択します。ロールを作成した後に名前を変更することはできません。
12. AWS サービスを選択し、EC2を選択します。
13. [Next (次へ)] を選択します。
14. アクセス許可ポリシーリストから、以前に作成したポリシーを選択します。
15. [Next (次へ)] を選択します。
16. ロール名を入力し、[ロールの作成] を選択します。ロールを作成した後に名前を変更することはできません。
17. スタックでAWSフロー (Ruby) AWS OpsWorks レイヤーを作成するときに、このプロファイルを指定します。

Java アプリケーションサーバー AWS OpsWorks スタックレイヤー

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

Java アプリケーションサーバーレイヤーは、Java AWS OpsWorks アプリケーションサーバーとして機能するインスタンスの設計図を提供する スタックレイヤーです。このレイヤーは [Apache Tomcat 7.0](#) と [Open 7 JDK](#) に基づいています。AWS OpsWorks スタックは Java コネクタライブラリもインストールします。これにより、Java アプリケーションは JDBC DataSource オブジェクトを使用してバックエンドデータストアに接続できます。

インストール: Tomcat は `/usr/share/tomcat7` にインストールされます。

[Add Layer] ページには、以下の設定オプションがあります。

Java VM Options

この設定を使用すると、カスタム Java VM オプションを指定できます。デフォルトオプションはありません。たとえば、一般的なオプションのセットは `-Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC` です。Java VM オプションを使用する場合は、有効なオプションセットを渡すようにしてください。AWS OpsWorks スタックは文字列を検証しません。無効なオプションを渡そうとすると、通常は Tomcat サーバーが開始に失敗し、セットアップが失敗します。その場合は、インスタンスのセットアップ Chef ログで詳細を確認できます。Chef ログを表示および解釈する方法の詳細については、「[Chef ログ](#)」を参照してください。

Custom security groups

この設定は、組み込み AWS OpsWorks のスタックセキュリティグループをレイヤーに自動的に関連付けないことを選択した場合に表示されます。レイヤーに関連付けるセキュリティグループを指定する必要があります。詳細については、「[新しいスタックを作成する](#)」を参照してください。

[Elastic Load Balancer]

レイヤーのインスタンスに、Elastic Load Balancing のロードバランサーをアタッチできます。詳細については、「[Elastic ロードバランシングレイヤー](#)」を参照してください。

カスタムJSONまたはカスタム属性ファイルを使用して、他の設定を指定できます。詳細については、「[カスタム設定](#)」を参照してください。

Important

Java アプリケーションが を使用している場合はSSL、SSLv3可能であれば を無効にして、[CVE-2014-3566](#) で説明されている脆弱性に対処することをお勧めします。詳細については、「[Apache サーバーSSLv3での の無効化](#)」を参照してください。

トピック

- [Apache サーバーSSLv3での の無効化](#)
- [カスタム設定](#)
- [Java アプリケーションのデプロイ](#)

Apache サーバー-SSLv3での の無効化

を無効にするにはSSLv3、Apache サーバーの `ssl.conf` ファイルSSLProtocol設定を変更する必要があります。そのためには、組み込みの [apache2 cookbook's](#) (apache2 クックブック) の `ssl.conf.erb` テンプレートファイルをオーバーライドする必要があります。このテンプレートファイルは、Java アプリケーションアプリケーションサーバーレイヤーの Setup レシピで `ssl.conf` を作成するために使用されます。詳細は、レイヤーのインスタンスに指定するオペレーティングシステムによって異なります。以下に、Amazon Linux および Ubuntu システムに必要な変更をまとめます。SSLv3は Red Hat Enterprise Linux (RHEL) システムでは自動的に無効になります。組み込みテンプレートのオーバーライドの詳細については、「[カスタムテンプレートの使用](#)」を参照してください。

Amazon Linux

これらのオペレーティングシステムの `ssl.conf.erb` ファイルは、`apache2` クックブックの `apache2/templates/default/mods` ディレクトリにあります。組み込みファイルの関連する部分を次に示します。

```
...
#SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

# enable only secure protocols: SSLv3 and TLSv1.2, but not SSLv2
SSLProtocol all -SSLv2
</IfModule>
```

`ssl.conf.erb` をオーバーライドし、SSLProtocol 設定を次のように変更します。

```
...
#SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

# enable only secure protocols: SSLv3 and TLSv1.2, but not SSLv2
SSLProtocol all -SSLv3 -SSLv2
</IfModule>
```

Ubuntu 14.04 LTS

このオペレーティングシステムの `ssl.conf.erb` ファイルは、`apache2` クックブックの `apache2/templates/ubuntu-14.04/mods` ディレクトリにあります。組み込みファイルの関連する部分を次に示します。

```
...
# The protocols to enable.
# Available values: all, SSLv3, TLSv1.2
# SSL v2 is no longer supported
SSLProtocol all
...
```

この設定を次のように変更します。

```
...
# The protocols to enable.
# Available values: all, SSLv3, TLSv1.2
# SSL v2 is no longer supported
SSLProtocol all -SSLv3 -SSLv2
...
```

カスタム設定

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは、追加の設定を組み込み属性として公開します。これらはすべて `opsworks_java` 名前空間にあります。カスタム JSON またはカスタム属性ファイルを使用して、組み込み属性を上書きし、カスタム値を指定できます。例えば、JVM および Tomcat のバージョンは、組み込みの属性 `jvm_version` と `java_app_server_version` 属性で表され、どちらも 7 に設定されます。カスタム JSON またはカスタム属性ファイルを使用して、のいずれかまたは両方を 6 に設定できます。次の例では JSON、カスタム を使用して両方の属性を 6 に設定します。

```
{
  "opsworks_java": {
```



```
"jvm_version": 6,  
"java_app_server_version" : 6  
}  
}
```

詳細については、「[カスタムの使用 JSON](#)」を参照してください。

カスタム設定のもう 1 つの例

は、`use_custom_pkg_location`、`custom_pkg_location_url_debian` および `custom_pkg_location_url_rhel` 属性を上書きしてカスタムをインストールすることで

Note

組み込みクックブックをオーバーライドすると、これらのコンポーネントを自分で更新しなければなりません。

属性および属性をオーバーライドする方法の詳細については、「[属性の上書き](#)」を参照してください。組み込み属性の一覧については、「[opsworks_java 属性](#)」を参照してください。

Java アプリケーションのデプロイ

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

以下のトピックでは、Java アプリケーションサーバーレイヤーのインスタンスにアプリケーションをデプロイする方法について説明します。例は JSP アプリ用ですが、他のタイプの Java アプリをインストールするために基本的に同じ手順を使用できます。

サポートされている任意のリポジトリから JSP ページをデプロイできます。WAR ファイルをデプロイする場合、AWS OpsWorks スタックは Amazon S3 または HTTP アーカイブからデプロイされた WAR ファイルを自動的に抽出しますが、Git または Subversion リポジトリからは抽出しないことに

注意してください。WAR ファイルで Git または Subversion を使用する場合は、次のいずれかを実行できます。

- 抽出されたアーカイブをリポジトリに保存します。
- 次の例で説明するように、WAR ファイルをリポジトリに保存し、Chef デプロイフックを使用してアーカイブを抽出します。

Chef デプロイメントフックを使用すると、4 つのデプロイメントステージのうちの任意のステージのインスタンスでユーザー提供の Ruby アプリケーションを実行できます。アプリケーション名によってステージが決まります。以下は、Git または Subversion リポジトリからデプロイされた WAR ファイルを抽出する `before_migrate.rb`、という名前の Ruby アプリケーションの例です。アプリケーションは、名前によって Checkout デプロイメントフックに関連付けられているので、デプロイメント操作の開始時に、コードのチェック後、移行の前に実行されます。この例を使用する方法の詳細については、「[Chef デプロイフックの使用](#)」を参照してください。

```
::Dir.glob(::File.join(release_path, '*.war')) do |archive_file|
  execute "unzip_#{archive_file}" do
    command "unzip #{archive_file}"
    cwd release_path
  end
end
```

Note

JSP アプリに更新をデプロイすると、Tomcat は更新を認識せず、代わりに既存のアプリバージョンを引き続き実行する可能性があります。これは、例えば、JSP ページのみを含む `.zip` ファイルとしてアプリケーションをデプロイする場合などに発生する可能性があります。Tomcat が最後にデプロイされたバージョンを実行するようにするには、プロジェクトのルートディレクトリに `web.xml` ファイルを含む `WEB-INF` ディレクトリが含まれている必要があります。`web.xml` ファイルはさまざまなコンテンツを含むことができますが、Tomcat が確実に更新プログラムを認識し、現在デプロイされているアプリケーションのバージョンを実行するようにするには、次のコンテンツで十分です。各更新プログラムのバージョンを変更する必要はありません。Tomcat は、バージョンが変更されていない場合でも更新プログラムを認識します。

```
<context-param>
```

```
<param-name>appVersion</param-name>
<param-value>0.1</param-value>
</context-param>
```

トピック

- [JSP アプリケーションのデプロイ](#)
- [バックエンドデータベースを使用したJSPアプリケーションのデプロイ](#)

JSP アプリケーションのデプロイ

JSP アプリケーションをデプロイするには、名前とリポジトリ情報を指定します。オプションでドメインとSSL設定を指定することもできます。アプリケーションの作成方法の詳細については、「[アプリケーションの追加](#)」を参照してください。次の手順は、パブリック Amazon S3 アーカイブからシンプルなJSPページを作成してデプロイする方法を示しています。プライベートの Amazon S3 アーカイブなど、他のリポジトリの種類の使用方法の詳細については、「[Application Source](#)」を参照してください。

次の例はJSP、いくつかのシステム情報を表示するページを示しています。

```
<%@ page import="java.net.InetAddress" %>
<html>
<body>
<%
    java.util.Date date = new java.util.Date();
    InetAddress inetAddress = InetAddress.getLocalHost();
%>
The time is
<%
    out.println( date );
    out.println("<br>Your server's hostname is "+inetAddress.getHostName());
%>
<br>
</body>
</html>
```

Note

次の手順では、スタックの作成、レイヤーへのインスタンスの追加などの基本操作にすでに慣れていることを前提としています。AWS OpsWorks スタックを初めて使用する場合は、まず「」を参照してください[Chef 11 Linux スタックの使用開始](#)。

Amazon S3 アーカイブからJSPページをデプロイするには

1. Java アプリケーションサーバーのレイヤーを持つ[スタックを作成](#)し、レイヤーに [24/7 のインスタンスを追加](#)して[起動](#)します。
2. コードを simplejsp.jsp という名前のファイルにコピーし、ファイルを simplejsp という名前のフォルダに保存して、そのフォルダの .zip アーカイブを作成します。名前は任意です。ファイル名とフォルダ名を自由に指定できます。gzip、bzip2、tarball、Java WAR ファイルなど、他のタイプのアーカイブを使用することもできます。AWS OpsWorks スタックは非圧縮 tarball をサポートしていないことに注意してください。複数のJSPページをデプロイするには、それらを同じアーカイブに含めます。
3. Amazon S3 バケットにアーカイブをアップロードし、ファイルを公開します。URL 後で使用するために、ファイルの をコピーします。バケットの作成方法とファイルのアップロード方法の詳細については、「[Amazon Simple Storage Service の使用開始](#)」を参照してください。
4. スタックに[アプリケーションを追加](#)し、次の設定を指定します。
 - 名前 – SimpleJSP
 - App type (アプリケーションタイプ) - Java
 - Repository type (リポジトリタイプ) – Http Archive
 - リポジトリ URL – URLアーカイブファイルの Amazon S3。

残りの設定にはデフォルト値を使用し、[Add App] をクリックしてアプリケーションを作成します。

5. Java アプリケーションサーバーインスタンスに[アプリケーションをデプロイ](#)します。

これで、アプリの に移動URLしてアプリを表示できるようになりました。ドメインを指定していない場合は、インスタンスのパブリック IP アドレスまたはそのパブリックDNS名URLを使用して構築できます。インスタンスのパブリック IP アドレスまたはパブリックDNS名を取得するには、

AWS OpsWorks スタックコンソールに移動し、インスタンスページでインスタンスの名前をクリックして、その詳細ページを開きます。

残りの部分は、アプリケーションの短縮名URLによって異なります。短縮名は、アプリケーションの作成時に指定したアプリケーション名から AWS OpsWorks スタックが生成する小文字の名前です。たとえば、SimpleJSP の短縮名は simplejsp です。アプリケーションの短縮名は、詳細ページで取得できます。

- 短縮名が root である場合は、`http://public_DNS/appname.jsp` または `http://public_IP/appname.jsp` を使用できます。
- それ以外の場合は、`http://public_DNS/app_shortcode/appname.jsp` または `http://public_IP/app_shortcode/appname.jsp` を使用できます。

アプリのドメインを指定した場合、URLは `http://domain/appname.jsp` です。

この例URLの は のようになります `http://192.0.2.0/simplejsp/simplejsp.jsp`。

同じインスタンスに複数のアプリケーションをデプロイする場合は、短縮名として root を使用しないでください。これにより、URL競合が発生し、アプリが正しく動作しなくなる可能性があります。その代わりに、各アプリケーションに異なるドメイン名を割り当てます。

バックエンドデータベースを使用したJSPアプリケーションのデプロイ

JSP ページでは、JDBC DataSource オブジェクトを使用してバックエンドデータベースに接続できます。このようなアプリケーションを作成およびデプロイするには、前のセクションの手順を使用しますが、接続をセットアップするステップが 1 つ追加されます。

次のJSPページでは、DataSource オブジェクトに接続する方法を示します。

```
<html>
  <head>
    <title>DB Access</title>
  </head>
  <body>
    <%@ page language="java" import="java.sql.*,javax.naming.*,javax.sql.*" %>
    <%
      StringBuffer output = new StringBuffer();
      DataSource ds = null;
      Connection con = null;
      Statement stmt = null;
```

```
ResultSet rs = null;
try {
    Context initCtx = new InitialContext();
    ds = (DataSource) initCtx.lookup("java:comp/env/jdbc/mydb");
    con = ds.getConnection();
    output.append("Databases found:<br>");
    stmt = con.createStatement();
    rs = stmt.executeQuery("show databases");
    while (rs.next()) {
        output.append(rs.getString(1));
        output.append("<br>");
    }
}
catch (Exception e) {
    output.append("Exception: ");
    output.append(e.getMessage());
    output.append("<br>");
}
finally {
    try {
        if (rs != null) {
            rs.close();
        }
        if (stmt != null) {
            stmt.close();
        }
        if (con != null) {
            con.close();
        }
    }
    catch (Exception e) {
        output.append("Exception (during close of connection): ");
        output.append(e.getMessage());
        output.append("<br>");
    }
}
%>
<%= output.toString() %>
</body>
</html>
```

AWS OpsWorks スタックはDataSourceオブジェクトを作成および初期化し、論理名にバインドして、Java 命名およびディレクトリインターフェイス (JNDI) 命名サービスに登録します。完全な論理名は、`java:comp/env/user-assigned-name` です。以下に示すように、カスタムJSON属性をスタック設定およびデプロイ属性に追加して属性を定義し['opsworks_java'] ['datasources']、名前のユーザー割り当て部分を指定する必要があります。

MySQLデータベースに接続するJSPページをデプロイするには

1. Java アプリケーションサーバーのレイヤーを持つ [スタックを作成](#)し、各レイヤーに [24/7 インスタンスを追加](#)して[開始](#)します。
2. スタックにデータベースレイヤーを追加します。詳細は、使用しているデータベースによって異なります。

この例で MySQL インスタンスを使用するには、スタックに [MySQL レイヤーを追加](#)し、[24/7 インスタンスをレイヤーに追加](#)して[起動](#)します。

この例で Amazon RDS (MySQL) インスタンスを使用するには：

- インスタンスの MySQL データベースエンジンを指定します。
- AWS-OpsWorks-DB-Master-Server (*security_group_id*) および AWS-OpsWorks-Java-App-Server (*security_group_id*) セキュリティグループをインスタンスに割り当てます。AWS OpsWorks スタックは、リージョンで最初のスタックを作成するときに、これらのセキュリティグループを作成します。
- simplejspdb という名前のデータベースを作成します。
- マスターユーザー名とパスワードに & または Tomcat エラーの原因になる他の文字が含まれていないことを確認します。

特に起動時に、Tomcat はウェブアプリケーションコンテキストファイルを解析する必要があります。これは、マスターパスワードとユーザー名を含む XML ファイルです。いずれかの文字列に&文字が含まれている場合、XMLパーサーはそれを不正な形式のXMLエンティティとして扱い、解析例外をスローします。これにより、Tomcat が起動できなくなります。ウェブアプリケーションのコンテキストファイルの詳細については、「[tomcat::context](#)」を参照してください。

- [MySQL ドライバー](#)を Java アプリケーションサーバーレイヤーに追加します。
- [RDS インスタンスをスタックに登録](#)します。

スタックで Amazon AWS OpsWorks RDS インスタンスを使用する方法の詳細については、「」を参照してください [Amazon RDS Service Layer](#)。

3. サンプルコードを `simplejspdb.jsp` という名前のファイルにコピーし、ファイルを `simplejspdb` という名前のフォルダに保存して、そのフォルダの `.zip` アーカイブを作成します。名前は任意です。ファイル名とフォルダ名を自由に指定できます。gzip、bzip2、tarball など、他の種類のアーカイブも使用できます。複数の JSP ページをデプロイするには、それらと同じアーカイブに含めます。他のリポジトリタイプのアプリケーションをデプロイする方法の詳細については、「[Application Source](#)」を参照してください。
4. Amazon S3 バケットにアーカイブをアップロードし、ファイルを公開します。URL 後で使用するために、ファイルの をコピーします。バケットの作成方法とファイルのアップロード方法の詳細については、「[Amazon Simple Storage Service の使用開始](#)」を参照してください。
5. スタックに [アプリケーションを追加](#)し、次の設定を指定します。
 - 名前 – SimpleJSPDB
 - App type (アプリケーションタイプ) - Java
 - データソースタイプ – OpsWorks (MySQL インスタンスの場合) または RDS (Amazon RDS インスタンスの場合)。
 - データベースインスタンス – 前に作成した MySQL インスタンス。通常は `db-master1 (mysql)` という名前、または Amazon RDS インスタンス **`DB_instance_name`** (mysql) という名前になります。
 - Database name (データベース名) – `simplejspdb`
 - Repository type (リポジトリタイプ) – Http Archive
 - リポジトリ URL – URL アーカイブファイルの Amazon S3。

残りの設定にはデフォルト値を使用し、[Add App] をクリックしてアプリケーションを作成します。

6. 次のカスタム JSON 属性をスタック設定属性に追加します。ここで、`simplejspdb` はアプリケーションの短縮名です。

```
{
  "opsworks_java": {
    "datasources": {
      "simplejspdb": "jdbc/mydb"
    }
  }
}
```



```
    }  
  }  
}
```

AWS OpsWorks スタックはこのマッピングを使用して、必要なデータベース情報を含むコンテキストファイルを生成します。

スタック設定JSON属性にカスタム属性を追加する方法の詳細については、「」を参照してください [カスタム の使用 JSON](#)。

7. Java アプリケーションサーバーインスタンスに [アプリケーションをデプロイ](#) します。

アプリの を使用してアプリURLを表示できるようになりました。の構築方法の詳細については URL、「」を参照してください [JSP アプリケーションのデプロイ](#)。

この例URLの は のようになります `http://192.0.2.0/simplejspdb/simplejspdb.jsp`。

Note

`datasources` 属性には、複数の属性を含めることができます。各属性はアプリケーションの短縮名によって名付けられ、論理名の適切なユーザー割り当て部分に設定されます。複数のアプリケーションがある場合は、別の論理名を使用できます。そのためには、次のようなカスタムJSONのものがが必要です。

```
{  
  "opsworks_java": {  
    "datasources": {  
      "myjavaapp": "jdbc/myappdb",  
      "simplejsp": "jdbc/myjspdb",  
      ...  
    }  
  }  
}
```

Node.js アプリケーションサーバー AWS OpsWorks スタックレイヤー

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

Node.js アプリケーションサーバーレイヤーは、[Node.js](#) AWS OpsWorks アプリケーションサーバーとして機能するインスタンスの設計図を提供する スタックレイヤーです。AWS OpsWorks スタックは [Express](#) もインストールするため、レイヤーのインスタンスは標準アプリケーションと Express アプリケーションの両方をサポートします。

インストール: Node.js は `/usr/local/bin/node` にインストールされます。

[Add Layer] ページには、以下の設定オプションがあります。

Node.js バージョン

現在サポートされているバージョンのリストについては、「[AWS OpsWorks スタックオペレーティングシステム](#)」を参照してください。

Custom security groups

この設定は、組み込み AWS OpsWorks の スタックセキュリティグループをレイヤーに自動的に関連付けないことを選択した場合に表示されます。レイヤーに関連付けるセキュリティグループを指定する必要があります。詳細については、「[新しいスタックを作成する](#)」を参照してください。

[Elastic Load Balancer]

レイヤーのインスタンスに、Elastic Load Balancing のロードバランサーをアタッチできます。

⚠ Important

Node.js アプリケーションが を使用している場合はSSL、[CVE-2015-8027](#) で説明されている脆弱性に対処するSSLv3ために を無効にすることをお勧めします。そのためには、[Node.js version] を 0.12.9 に設定する必要があります。

Node.js アプリケーションのデプロイ

AWS OpsWorks スタック用にシンプルな Node.js アプリケーションを実装してそれをスタックにデプロイする方法の手順については、「[Node.js スタックの初回作成](#)」を参照してください。一般に、AWS OpsWorks スタック用の Node.js アプリケーションは、以下の条件を満たす必要があります。

- メインファイルは `server.js` という名前にして、デプロイされたアプリケーションのルートディレクトリに置く必要があります。
- [Express](#) アプリケーションには、アプリケーションのルートディレクトリに `package.json` ファイルを含める必要があります。
- デフォルトでは、アプリケーションはポート 80 (HTTP) またはポート 443 () でリッスンする必要がありますHTTPS。

他のポートでリッスンすることは可能ですが、Node.js アプリケーションサーバーレイヤーの組み込みセキュリティグループ `AWS-OpsWorks-nodejs-App-Server` では、ポート 80、443、22 () へのインバウンドユーザートラフィックのみが許可されますSSH。他のポートへの着信ユーザートラフィックを許可するには、適切なインバウンドルールを使用して[セキュリティグループを作成し、Node.js アプリケーション-サーバーレイヤーに割り当てます](#)。組み込みのセキュリティグループを編集することでインバウンドルールを変更しないでください。スタックを作成するたびに、AWS OpsWorks スタックは組み込みセキュリティグループを標準設定で上書きするため、変更を加えると変更内容は失われます。

i Note

AWS OpsWorks スタックはPORT環境変数を 80 (デフォルト) または 443 (を有効にした場合SSL) に設定するため、次のコードを使用してリクエストをリッスンできます。

```
app.listen(process.env.PORT);
```

[をサポートするように Node.js アプリを設定するSSL](#)場合は、キーと証明書を指定する必要があります。AWS OpsWorks スタックは、次のように、各アプリケーションサーバーインスタンスのデータを `/srv/www/app_shortname/shared/config` ディレクトリに個別のファイルとして配置します。

- `ssl.crt` – SSL証明書。
- `ssl.key` – SSLキー。
- `ssl.ca` – チェーン証明書 (指定した場合)。

アプリケーションは、これらのファイルからSSLキーと証明書を取得できます。

PHP アプリケーションサーバー AWS OpsWorks スタックレイヤー

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

PHP アプリケーションサーバーレイヤーは、PHPアプリケーションサーバーとして機能するインスタンスの設計図を提供する AWS OpsWorks スタックレイヤーです。PHP App Server レイヤーは、を使用した [Apache2](#) に基づいて `mod_php` おり、標準設定オプションはありません。PHP および Apache のバージョンは、レイヤーのインスタンスに指定する [オペレーティングシステム](#) によって異なります。

オペレーティングシステム	PHP バージョン	Apache のバージョン
Amazon Linux 2018.03	5.3	2.2

オペレーティングシステム	PHP バージョン	Apache のバージョン
Amazon Linux 2017.09	5.3	2.2
Amazon Linux 2017.03	5.3	2.2
Amazon Linux 2016.09	5.3	2.2
Amazon Linux 2016.03	5.3	2.2
Amazon Linux 2015.09	5.3	2.2
Amazon Linux 2015.03	5.3	2.2
Amazon Linux 2014.09	5.3	2.2
Ubuntu 14.04 LTS	5.5	2.4

インストール: AWS OpsWorks スタックはインスタンスのパッケージインストーラを使用して、デフォルトの場所に Apache2 と `mod_php` をインストールします。インストールの詳細については、「[Apache](#)」を参照してください。

[Add Layer] ページには、以下の設定オプションがあります。

Custom security groups

この設定は、組み込み AWS OpsWorks のスタックセキュリティグループをレイヤーに自動的に関連付けないことを選択した場合に表示されます。レイヤーに関連付けるセキュリティグループを指定する必要があります。詳細については、「[新しいスタックを作成する](#)」を参照してください。

[Elastic Load Balancer]

レイヤーのインスタンスに、Elastic Load Balancing のロードバランサーをアタッチできます。

一部の Apache 設定は、カスタムJSONまたはカスタム属性ファイルを使用して変更できます。詳細については、「[属性の上書き](#)」を参照してください。オーバーライドできる Apache 属性の一覧については、「[apache2 属性](#)」を参照してください。

PHP アプリケーションをバックエンドデータベースに接続する方法など、アプリケーションをデプロイする方法の例については、「[Chef 11 Linux スタックの使用開始](#)」を参照してください。

⚠ Important

PHP アプリケーションで を使用している場合はSSL、SSLv3可能であれば を無効にして、[CVE-2014-3566](#) で説明されている脆弱性に対処することをお勧めします。そのためには、Apache サーバーの SSLProtocol ファイルの ssl.conf 設定を変更する必要があります。この設定を変更する方法の詳細については、「[Apache サーバーSSLv3での の無効化](#)」を参照してください。

Rails App Server AWS OpsWorks スタックレイヤー

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または[AWS プレミアムサポート](#)を通じて AWS サポート チームにお問い合わせください。

i Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

Rails アプリケーションサーバーレイヤーは、Rails AWS OpsWorks アプリケーションサーバーとして機能するインスタンスの設計図を提供する スタックレイヤーです。

インストール: AWS OpsWorks スタックはインスタンスのパッケージインストーラを使用して、サーバーパッケージをデフォルトの場所にインストールします。Apache/Passenger のインストールの詳細については、「[Phusion Passenger](#)」を参照してください。ログ記録の詳細については、「[ログファイル](#)」を参照してください。Nginx/Unicorn のインストールの詳細については、「[Unicorn](#)」を参照してください。

[Add Layer] ページには、以下の設定オプションがあります。いずれもオプションです。

Ruby Version

アプリケーションで使用される Ruby バージョン。デフォルト値は 2.3 です。

[\[:opsworks\]\[:ruby_version\]](#) 属性を上書きすることで、目的の Ruby のバージョンを指定することもできます。

Note

AWS OpsWorks スタックは、レシピとインスタンスエージェントで使用する別の Ruby パッケージをインストールします。詳細については、「[Ruby のバージョン](#)」を参照してください。

Rails Stack

デフォルト Rails スタックは、[Apache2](#) と [Phusion Passenger](#) です。また、[Nginx](#) と [Unicorn](#) を使用することもできます。

Note

Nginx と Unicorn を使用する場合は、以下の例のように、アプリケーションの Gemfile に unicorn gem を追加する必要があります。

```
source 'https://rubygems.org'
gem 'rails', '3.2.15'
...
# Use unicorn as the app server
gem 'unicorn'
...
```

Passenger Version

Apache2/Passenger を指定した場合は、Passenger のバージョンを指定する必要があります。デフォルト値は 5.0.28 です。

[Rubygems Version (Rubygems のバージョン)]

デフォルトの [Rubygems](#) バージョンは 2.5.1 です。

[Install and Manage Bundler (Bundler のインストールと管理)]

[Bundler](#) をインストールおよび管理するかどうかを選択できます。デフォルト値は [Yes] です。

[Bundler version (Bundler のバージョン)]

デフォルトの Bundler バージョンは 1.12.5 です。

Custom security groups

この設定は、組み込み AWS OpsWorks の スタックセキュリティグループをレイヤーに自動的に関連付けないことを選択した場合に表示されます。レイヤーに関連付けるセキュリティグループを指定する必要があります。詳細については、「[新しいスタックを作成する](#)」を参照してください。

[Elastic Load Balancer]

レイヤーのインスタンスに、Elastic Load Balancing のロードバランサーをアタッチできます。

一部の設定は、カスタムJSONまたはカスタム属性ファイルを使用して変更できます。詳細については、「[属性の上書き](#)」を参照してください。オーバーライドできる Apache、Nginx、Phusion Passenger、および Unicorn 属性の一覧については、「[組み込みクックブックの属性](#)」を参照してください。

Important

Ruby on Rails アプリケーションが を使用している場合はSSL、[CVE-2014-3566](#) で説明されている脆弱性に対処するSSLv3ために を無効にすることをお勧めします。詳細については、「[SSLv3 Rails サーバーの の無効化](#)」を参照してください。

トピック

- [SSLv3 Rails サーバーの の無効化](#)
- [データベースへの接続](#)
- [Rails アプリケーションの Ruby のデプロイ](#)

SSLv3 Rails サーバーの の無効化

SSLv3 Rails サーバーの を無効にするには、レイヤーの Ruby バージョン設定を 2.1 以上に更新します。これにより、アプリケーションが使用するバージョンとして Ruby 2.1.4 以降がインストールされます。

- レイヤーの [Ruby Version (Ruby のバージョン)] 設定を 2.1 以上に更新します。

- Rails スタックの設定ファイルを以下のように更新します。

Apache と Phusion Passenger

「SSLProtocol」で説明しているように、Apache Server の `ssl.conf` ファイルの [Apache サーバー-SSLv3での の無効化](#) 設定を更新します。

Nginx と Unicorn

Nginx サーバーの `ssl_protocols` ファイルに明示的な `nginx.conf` ディレクティブを追加します。を無効にするにはSSLv3、Rails App Server レイヤーの Setup レシピが の作成に使用する組み込み [nginx クックブック](#) の `nginx.conf.erb` テンプレートファイルを上書きし `nginx.conf`、次のディレクティブを追加します。

```
ssl_protocols TLSv1.2;
```

の設定方法の詳細については `nginx.conf`、[HTTPS「サーバーの設定」](#) を参照してください。組み込みテンプレートのオーバーライドの詳細については、「[カスタムテンプレートの使用](#)」を参照してください。

データベースへの接続

アプリケーションをデプロイすると、AWS OpsWorks スタックはアプリケーションの [deploy 属性](#) からの情報を使用して新しい `database.yml` ファイルを作成します。[MySQL または Amazon RDS インスタンスをアプリケーションにアタッチ](#) すると、AWS OpsWorks スタックは接続情報を `deploy` 属性に追加し、 に正しい接続データ `database.yml` が自動的に含まれるようにします。

アプリにデータベースがアタッチされていない場合、デフォルトでは、AWS OpsWorks スタックは `deploy` 属性に接続情報を追加せず、 を作成しません `database.yml`。別のデータベースを使用する場合は、カスタム を使用してJSON、接続情報を含むアプリケーションの属性にデータベース `deploy` 属性を追加できます。属性はすべて の下にあり `["deploy"]["appshortname"]` `["database"]`、 *appshortname* はアプリケーションの短縮名で、AWS OpsWorks スタックはアプリケーション名から生成します。カスタムで指定した値は、デフォルト設定をJSON上書きします。詳細については、「[アプリケーションの追加](#)」を参照してください。

AWS OpsWorks スタックは、次の `[:...][:database]` 属性値を に組み込みます `database.yml`。必要な属性は特定のデータベースによって異なりますが、 `host` 属性が必要です。そうしないと AWS OpsWorks、スタックは を作成しません `database.yml`。

- `[:adapter]` (String) – データベースアダプタ (mysql など)。
- `[:database]` (文字列) – データベース名。
- `[:encoding]` (文字列) – 通常は utf8 に設定されるエンコード。
- `[:host]` (文字列) – URLなどのホスト `railsexample.cd1qlk5uwd0k.us-west-2.rds.amazonaws.com`。
- `[:reconnect]` (ブール) – 接続が存在しなくなった場合にアプリケーションを再接続するかどうか。
- `[:password]` (文字列) – データベースのパスワード。
- `[:port]` (数値)。 – データベースのポートナンバー。アダプタによって設定されるデフォルトのポート番号をオーバーライドするには、この属性を使用します。
- `[:username]` (文字列) – データベースユーザー名。

次の例は、短縮名が `myapp` のアプリJSONのカスタムを示しています。

```
{
  "deploy" : {
    "myapp" : {
      "database" : {
        "adapter" : "adapter",
        "database" : "databasename",
        "host" : "host",
        "password" : "password",
        "port" : portnumber
        "reconnect" : true/false,
        "username" : "username"
      }
    }
  }
}
```

カスタムを指定する方法についてはJSON、「」を参照してください[カスタムの使用JSON](#)。 `database.yml` を作成するために使用したテンプレート (`database.yml.erb`) を見るには、[組み込みクックブックリポジトリ](#)にアクセスしてください。

Rails アプリケーションの Ruby のデプロイ

サポートされている任意のリポジトリから Rails アプリケーションの Ruby をデプロイすることができます。以下に、Apache/Passenger Rails スタックを実行しているサーバーにサンプルの Rails アプリケーションの Ruby をデプロイする方法を示します。サンプルコードはパブリック GitHub リポジトリに保存されますが、基本的な手順はサポートされている他のリポジトリでも同じです。アプリケーションの作成およびデプロイの方法の詳細については、「[アプリケーション](#)」を参照してください。広範なコメントを含む例のコードを表示するには、<https://github.com/aws-labs/opsworks-demo-rails-photo-share-app> にアクセスしてください。

GitHub リポジトリから Ruby on Rails アプリケーションをデプロイするには

1. Rails スタックとして Apache/Passenger を使用した Rails アプリケーションサーバーレイヤーで [スタックを作成](#) し、その層に [24/7 インスタンスを追加](#) して、[開始](#) します。
2. インスタンスがオンラインになってから、スタックに [アプリケーションを追加](#) し、以下の設定を指定します。

- Name (名前) – 任意の名前。この例では PhotoPoll を使用しています。

AWS OpsWorks スタックは表示目的でこの名前を使用し、内部使用のために短縮名を生成し、[スタック設定およびデプロイ属性](#) でアプリケーションを識別します。例えば、PhotoPoll 短縮名は photopoll です。

- App type (アプリケーションタイプ) – Ruby on Rails (Ruby on Rails)。
- Rails environment (Rails 環境) – 使用可能な環境はアプリケーションによって決定されます。

この例のアプリケーションには、**development**、**test**、**production** の 3 つの環境があります。この例では、環境を **development** に設定します。各環境の説明については、サンプルコードを参照してください。

- Repository type (リポジトリタイプ) – サポートされている任意のリポジトリタイプ。この例では Git を指定します。
- リポジトリ URL – コードをデプロイするリポジトリ。

この例では、URL を に設定します **git://github.com/aws-labs/opsworks-demo-rails-photo-share-app**。

残りの設定にはデフォルト値を使用し、[Add App] をクリックしてアプリケーションを作成します。

3. Rails アプリケーションサーバーインスタンスに[アプリケーションをデプロイ](#)します。
4. デプロイが完了したら、[Instances] (インスタンス) ページに移動して、Rails アプリケーションサーバー インスタンスのパブリック IP アドレスをクリックします。次のように表示されます。



静的ウェブサーバー AWS OpsWorks スタックレイヤー

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

i Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

静的ウェブサーバーレイヤーは、インスタンスが静的HTMLページを処理するためのテンプレートを提供する AWS OpsWorks スタックレイヤーです。これには、クライアント側のスクリプトが含まれる場合があります。このレイヤーは [Nginx](#) に基づいています。

インストール: Nginx は `/usr/sbin/nginx` にインストールされます。

[Add Layer] ページには、以下の設定オプションがあります。

Custom security groups

この設定は、組み込み AWS OpsWorks の スタックセキュリティグループをレイヤーに自動的に関連付けないことを選択した場合に表示されます。レイヤーに関連付けるセキュリティグループを指定する必要があります。詳細については、「[新しいスタックを作成する](#)」を参照してください。

[Elastic Load Balancer]

レイヤーのインスタンスに、Elastic Load Balancing のロードバランサーをアタッチできます。

一部の Nginx 設定は、カスタムJSONまたはカスタム属性ファイルを使用して変更できます。詳細については、「[属性の上書き](#)」を参照してください。オーバーライドできる Apache 属性の一覧については、「[nginx 属性](#)」を参照してください。

Important

ウェブアプリケーションがを使用している場合はSSL、[CVE-2014-3566](#) で説明されている脆弱性に対処するSSLv3ためにを無効にすることをお勧めします。

を無効にするにはSSLv3、Nginx サーバーの `nginx.conf` ファイルを変更する必要があります。そのためには、組み込みの[nginx クックブック](#)の `nginx.conf.erb` テンプレートファイルをオーバーライドします。これにより、Rails アプリケーションサーバーレイヤーのセットアップレシピは、`nginx.conf` を作成し、次のディレクティブを追加します。

```
ssl_protocols TLSv1.2;
```

の設定方法の詳細については`nginx.conf`、[HTTPS「サーバーの設定」](#)を参照してください。組み込みテンプレートのオーバーライドの詳細については、「[カスタムテンプレートの使用](#)」を参照してください。

ECS クラスターレイヤーリファレンス

Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

ECS クラスターレイヤーは [Amazon Elastic Container Service \(Amazon ECS\) クラスター](#) を表し、クラスター管理を簡素化します。

Short name: ecs クラスター

互換性 : [Amazon ECSサービス](#) レイヤーはカスタムレイヤーとのみ互換性があります

オープンポート : ECS クラスターはポート 22 へのパブリックアクセスを許可します (SSH)

Autoassign Elastic IP addresses: デフォルトで Off

デフォルトEBSボリューム : いいえ

デフォルトのセキュリティグループ : AWS-OpsWorks-ECS-Cluster

設定 : ECSクラスターレイヤーを設定するには、以下を指定する必要があります。

- コンテナインスタンスにパブリック IP アドレスまたは Elastic IP アドレスを許可するかどうか
- コンテナインスタンスのインスタンスプロファイル

Setup recipes:

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_ecs::setup

Configure recipes:

- `opsworks_ganglia::configure-client`
- `ssh_users`
- `mysql::client`
- `agent_version`
- `opsworks_ecs::configure`

Deploy recipes:

- `deploy::default`
- `opsworks_ecs::deploy`

Undeploy recipes:

- `opsworks_ecs::undeploy`

Shutdown recipes:

- `opsworks_shutdown::default`
- `opsworks_ecs::shutdown`

インストール:

- AWS OpsWorks スタックはインスタンスのパッケージインストーラを使用して、Docker をデフォルトの場所にインストールします。
- Setup イベントの Chef ログには、Amazon ECS エージェントが正常にインストールされたかどうか記録されます。それ以外の場合、AWS OpsWorks スタックによって提供されるログには Amazon ECS エラーログ情報は含まれません。エラーの発生 `handleAmazon ECS` 方法の詳細については、[「Amazon の ECS トラブルシューティング」](#) を参照してください。

カスタムレイヤーリファレンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソ

リユースに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

標準のレイヤーが要件を満たさない場合、カスタムレイヤーを作成することができます。スタックには、複数のカスタムレイヤーを含ませることができます。デフォルトでは、カスタムレイヤーによって、基本機能をサポートする一部の標準レシピのみが実行されます。ライフサイクルイベントごとに一連のカスタムの Chef のレシピを実装してレイヤーのソフトウェアの設定などを行うことで、レイヤーの主要機能を実装できます。カスタムレシピは、各イベントの標準 AWS OpsWorks スタックレシピの後に実行されます。

Short name: ユーザー定義。スタック内のカスタムレイヤーごとに別の短縮名が必要です。

オープンポート: デフォルトでは、カスタムサーバーレイヤーは、スタックの Rails および PHP アプリケーションサーバーレイヤーからポート 22 (SSH)、80 (HTTP)、443 (HTTPS)、およびすべてのポートへのパブリックアクセスを開きます。

Autoassign Elastic IP Addresses: デフォルトで Off

デフォルトEBSボリューム: なし

デフォルトのセキュリティグループ: AWS-OpsWorks-Custom-Server

Compatibility: カスタムレイヤーは custom、db-master、lb、memcached、monitoring-master、nodejs-app、php-app、rails-app、および web のレイヤーと互換性があります。

Configuration: カスタムレイヤーを設定するには、次を指定する必要があります。

- レイヤーの名前
- レイヤーの短縮名。Chef のレシピのレイヤーを識別し、a~z と数字のみを使用します。

Linux スタックの場合、カスタムレイヤーでは、以下のレシピを使用します。

Setup recipes:

- opsworks_initial_setup
- ssh_host_keys
- ssh_users

- `mysql::client`
- `dependencies`
- `ebs`
- `opsworks_ganglia::client`

Configure recipes:

- `opsworks_ganglia::configure-client`
- `ssh_users`
- `agent_version`

Deploy recipes:

- `deploy::default`

Shutdown recipes:

- `opsworks_shutdown::default`

その他のレイヤーリファレンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックでは、次のレイヤーもサポートされています。

トピック

- [Ganglia レイヤーリファレンス](#)
- [Memcached レイヤーリファレンス](#)

Ganglia レイヤーリファレンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

Ganglia レイヤーは、インスタンスメトリックスのストレージと可視化を管理する分散管理システムである [Ganglia](#) をサポートします。これは、階層インスタンスポートロジで動作するように設計されており、インスタンスのグループにとって特に便利です。Ganglia には、次の 2 つの基本コンポーネントがあります。

- 低オーバーヘッド顧客。スタックの各インスタンスにインストールされ、マスターにメトリクスを送信します。
- マスター。クライアントからメトリクスを収集し、Amazon EBSボリュームに保存します。また、メトリクスをウェブページに表示します。

AWS OpsWorks スタックには、管理する各インスタンスに Ganglia モニタリングエージェントがあります。スタックに Ganglia レイヤーを追加して起動すると、各インスタンスの Ganglia エージェントはインスタンスにメトリクスをレポートします。Ganglia を使用するには、1 つのインスタンスを持つ Ganglia レイヤーをスタックに追加します。マスターの IP アドレスで Ganglia のバックエンドにログインして、データにアクセスできます。Chef レシピを記述して、追加メトリクス定義を提供することができます。

Short name: monitoring-master

Compatibility: (互換性) Ganglia レイヤーは、custom、db-master、memcached、php-app、rails-app のレイヤーと互換性があります。

オープンポート : Load-Balancer は、ポート 22 (SSH)、80 ()、および 443 (HTTP) へのパブリックアクセスを許可しますHTTPS。

Autoassign Elastic IP addresses: デフォルトで Off

デフォルトEBSボリューム : はい、 で /vol/ganglia

デフォルトのセキュリティグループ : AWS-OpsWorks-Monitoring-Master-Server

Configuration: (構成) Ganglia レイヤーを設定するには、次を指定する必要があります。

- モニタリンググラフへのアクセスURIを提供する。デフォルト値は `http://DNSName/ganglia` で、*DNSName*は Ganglia インスタンスDNSの名前です。
- 監視統計へのアクセスを制御するユーザー名とパスワード

Setup recipes:

- `opsworks_initial_setup`
- `ssh_host_keys`
- `ssh_users`
- `mysql::client`
- `dependencies`
- `ebs`
- `opsworks_ganglia::client`
- `opsworks_ganglia::server`

Configure recipes:

- `opsworks_ganglia::configure-client`
- `ssh_users`
- `agent_version`
- `opsworks_ganglia::configure-server`

Deploy recipes:

- `deploy::default`

- `opsworks_ganglia::configure-server`
- `opsworks_ganglia::deploy`

Shutdown recipes:

- `opsworks_shutdown::default`
- `apache2::stop`

インストール:

- Ganglia クライアントは、`/etc/ganglia` にインストールされています。
- Ganglia ウェブフロントエンドは、`/usr/share/ganglia-webfrontend` にインストールされています。
- Ganglia logtailer は、`/usr/share/ganglia-logtailer` にインストールされています。

Memcached レイヤーリファレンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Linux ベースのスタックでのみ使用できます。

[Memcached](#) は、任意のデータの分散メモリキャッシュシステムです。文字列とオブジェクトをのキーと値としてキャッシュすることでウェブサイトを高速化RAMし、外部データソースを読み取る必要がある回数を減らします。

Memcached をスタックで使用するには、Memcached レイヤーを作成して、1 つ以上のインスタンスを追加します。そのインスタンスは、Memcached サーバーとして機能します。インスタンス

により Memcached は自動的にインストールされ、スタックの他のインスタンスは、Memcached サーバーにアクセスしてそれを使用できます。Rails App Server レイヤーを使用する場合、AWS OpsWorks Stacks はレイヤー内の各インスタンスの設定ディレクトリに設定 `memcached.yml` ファイルを自動的に配置します。このファイルから Memcached サーバーとポート番号を取得することができます。

Short name: memcached

Compatibility: (互換性) Memcached レイヤーは custom、db-master、lb、monitoring-master、nodejs-app、php-app、rails-app およびウェブのレイヤーと互換性があります。

オープンポート： Memcached レイヤーは、スタックのウェブサーバー、カスタムサーバー、Rails、Node.js アプリケーションサーバーからポート 22 (SSH) PHP とすべてのポートへのパブリックアクセスを許可します。

Autoassign Elastic IP addresses: デフォルトで Off

デフォルトEBSボリューム： いいえ

デフォルトのセキュリティグループ： AWS-OpsWorks-Memcached-Server

Memcached レイヤーを設定するには、キャッシュサイズ (MB) を指定する必要があります。

Setup recipes:

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- memcached

Configure recipes:

- opsworks_ganglia::configure-client
- ssh_users

- `agent_version`

Deploy recipes:

- `deploy::default`

Shutdown recipes:

- `opsworks_shutdown::default`
- `memcached::stop`

インストール:

- AWS OpsWorks スタックはインスタンスのパッケージインストーラを使用して、Memcached とそのログファイルをデフォルトの場所にインストールします。

その他のレイヤー

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

これらのレイヤーは、Chef 11 以前の Linux ベースのスタックでのみ使用できます。

AWS OpsWorks スタックは、Ganglia レイヤーと Memcached レイヤーもサポートしています。

トピック

- [Ganglia レイヤー](#)
- [Memcached](#)

Ganglia レイヤー

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Chef 11 以前の Linux ベースのスタックでのみ使用できます。

AWS OpsWorks スタックは、すべてのインスタンスとボリュームのメトリクスを [Amazon CloudWatch](#) に送信します。これにより、グラフの表示とアラームの設定が容易になり、リソースの状態に基づいたトラブルシューティングや自動アクションの実行に役立ちます。Ganglia AWS OpsWorks スタックレイヤーを使用して、選択したメトリクスの保存などの追加のアプリケーションモニタリングオプションを使用することもできます。

Ganglia レイヤーは、[Ganglia](#) 分散モニタリングを使用してスタックを監視するインスタンスのブループリントです。スタックに置かれる Ganglia インスタンスは通常 1 つだけです。Ganglia レイヤーには、次のオプション構成があります。

ガングリア URL

統計の URL パス。完全な URL は `http://` です。ここで *DNSNameURLPath*、*DNSName* は関連付けられたインスタンス DNS の名前です。*URLPath* デフォルト値は「/ganglia」で、`http://ec2-54-245-151-7.us-west-2.compute.amazonaws.com/ganglia` などに対応します。

Ganglia user name

統計情報のウェブページのユーザー名。ページを表示するにはユーザー名を準備する必要があります。デフォルト値は、「opsWorks」です。

Ganglia password

統計情報のウェブページへのアクセスを制御するパスワード。ページを閲覧する際はパスワードを指定する必要があります。デフォルト値はランダムに生成される文字列です。

Note

後で使用するためにパスワードを記録します。AWS OpsWorks スタックでは、レイヤーの作成後にパスワードを表示することはできません。ただし、レイヤーの [Edit] ページにアクセスし、[Update password] をクリックすることによって、パスワードを更新することはできます。

Custom security groups

この設定は、組み込み AWS OpsWorks の スタックセキュリティグループをレイヤーに自動的に関連付けないことを選択した場合に表示されます。レイヤーに関連付けるセキュリティグループを指定する必要があります。詳細については、「[新しいスタックを作成する](#)」を参照してください。

[Elastic Load Balancer]

レイヤーのインスタンスに、Elastic Load Balancing のロードバランサーをアタッチできます。

Important

スタックに Ganglia レイヤーが含まれている場合は、SSLv3可能であればそのレイヤーを無効にして、[CVE-2014-3566](#) で説明されている脆弱性に対処することをお勧めします。そのためには、Apache サーバーの `ssl.conf.erb` テンプレートをオーバーライドして、SSLProtocol 設定を変更する必要があります。詳細については、「[Apache サーバー SSLv3での無効化](#)」を参照してください。

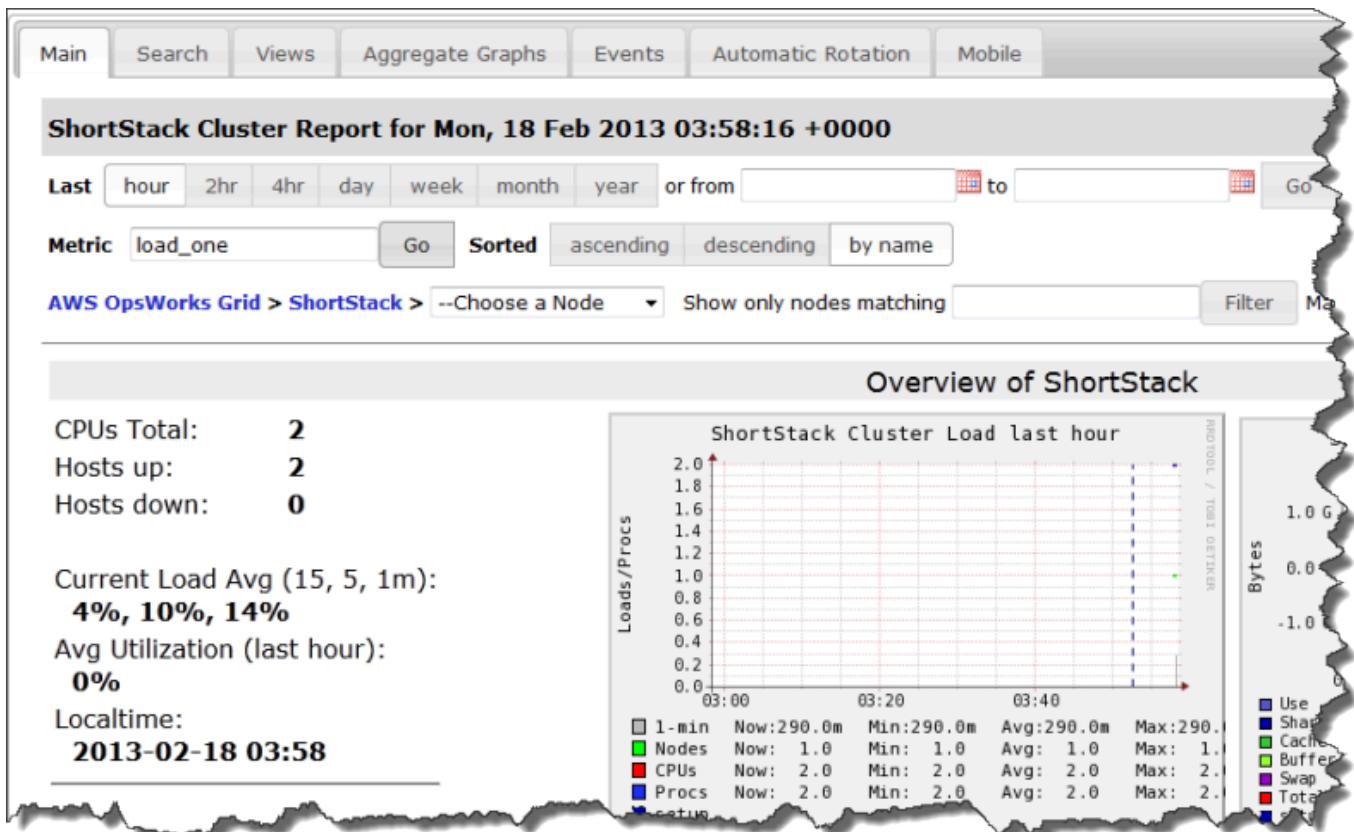
Ganglia の統計情報の表示

AWS OpsWorks スタックレシピは、すべてのインスタンスにオーバーヘッドの低い Ganglia クライアントをインストールします。スタックに Ganglia レイヤーが含まれている場合、インスタンスがオンラインになった時点ですぐに、Ganglia クライアントによって Ganglia へのレポートが自動的に開始されます。Ganglia は、クライアントデータを使用してさまざまな統計を算出し、その結果を統計情報ウェブページでグラフィカルに表示します。

Ganglia の統計情報を閲覧するには

1. [Layers] (レイヤー) ページの [Ganglia] をクリックし、そのレイヤーの詳細ページを開きます。

- ナビゲーションペインの [Instances] をクリックします。[Ganglia] (Ganglia) でインスタンス名をクリックします。
- インスタンスのパブリックDNS名をコピーします。
- DNS 名前を使用して統計を作成します。URLこれは `http://ec2-54-245-151-7.us-west-2.compute.amazonaws.com/ganglia` のようになります。
- 完成した をブラウザURLに貼り付け、ページに移動し、Ganglia ユーザー名とパスワードを入力してページを表示します。以下に例を示します。



Memcached

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

このレイヤーは、Chef 11 以前の Linux ベースのスタックでのみ使用できます。

Memcached レイヤーは、任意のデータ用の分散メモリキャッシュシステムである [Memcached](#) サーバーとして機能するインスタンスの設計図を提供する AWS OpsWorks スタックレイヤーです。Memcached レイヤーには、次の環境設定があります。

Allocated memory (MB)

(オプション)レイヤーのインスタンスごとのキャッシュメモリのサイズ (MB)です。デフォルトでは 512 MB です。

Custom security groups

この設定は、組み込み AWS OpsWorks の スタックセキュリティグループをレイヤーに自動的に関連付けないことを選択した場合に表示されます。レイヤーに関連付けるセキュリティグループを指定する必要があります。詳細については、「[新しいスタックを作成する](#)」を参照してください。

クックブックのコンポーネント**Important**

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

クックブックには通常、基本となる次のコンポーネントが含まれています。

- 属性ファイルには、レシピやテンプレートによって使用される値を表す一連の属性が含まれています。
- テンプレートファイルは、他のファイル (設定ファイルなど) を作成する際のひな型としてレシピで使用されます。

通常、テンプレートファイルでは、設定ファイルを書き換えるのではなく、クックブックに触れることなく属性をオーバーライドして、設定ファイルを変更できます。インスタンスの設定ファイルをわずかでも変更する場合は、テンプレートファイルを使用するのが一般的です。

- レシピファイルは、フォルダの作成と設定、パッケージのインストールと設定、サービスの開始など、システムを設定するうえで必要なあらゆる事柄を定義する Ruby アプリケーションです。

クックブックに 3 つのコンポーネントがすべて含まれているとは限りません。属性ファイルまたはテンプレートファイルだけを使用した、もっと簡単なカスタマイズ方法があります。さらに、クックブックには、必要に応じて他のファイルタイプ (定義、仕様など) を追加することもできます。

このセクションでは、3 つ標準的なクックブックのコンポーネントについて説明します。詳細 (特にレシピの実装方法) については、「[Opscode](#)」を参照してください。

トピック

- [属性](#)
- [テンプレート](#)
- [recipe](#)

属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

レシピとテンプレートは、設定を指定するさまざまな値に依存しています。それらの値をレシピやテンプレートに直接ハードコードするのではなく、各値を表す属性が記載された属性ファイルを作成します。レシピとテンプレートには、明示的な値ではなく、この属性を指定します。属性を使用する利点は、クックブックに変更を加えることなしに、値を上書きできることです。このような理由から、次に示すタイプの値の定義には、常に属性を使用するようにしてください。

- ユーザー名などの、スタックごとに異なる、または時間とともに変化する値。

このような値をハードコードした場合は、値を変更するたびにレシピやテンプレートを変更する必要があります。これらの値の定義に属性を使用することで、すべてのスタックに対して同じクックブックを使用し、該当する属性のみを上書きすることができます。

- パスワードやシークレットキーなどの機密情報を含む値。

クックブックに機密情報を明示的に入力すると、漏洩の危険が増します。代わりに、ダミーの値を指定した属性を定義し、実際の値は上書きで設定するようにします。このような属性を上書きする最善の方法は、カスタムを使用することですJSON。詳細については、「[カスタムの使用JSON](#)」を参照してください。

属性および属性を上書きする方法の詳細については、「[属性の上書き](#)」を参照してください。

次の例は、属性ファイルの例の一部です。

```
...
default["apache"]["listen_ports"] = [ '80','443' ]
default["apache"]["contact"] = 'ops@example.com'
default["apache"]["timeout"] = 120
default["apache"]["keepalive"] = 'Off'
default["apache"]["keepaliverequests"] = 100
default["apache"]["keepalivetimeout"] = 3
default["apache"]["prefork"]["startservers"] = 16
default["apache"]["prefork"]["minspareservers"] = 16
default["apache"]["prefork"]["maxspareservers"] = 32
default["apache"]["prefork"]["serverlimit"] = 400
default["apache"]["prefork"]["maxclients"] = 400
default["apache"]["prefork"]["maxrequestspchild"] = 10000
...
```

AWS OpsWorks スタックは、次の構文を使用して属性を定義します。

```
node.type["attribute"]["subattribute"]["..."]=value
```

次のように、コロン (:) を使用することもできます。

```
node.type[:attribute][:subattribute][:...]=value
```

属性の定義には、次のようなコンポーネントが含まれます。

node.

node. プレフィックスはオプションです。例に示すように、通常は省略されます。

type

タイプは、属性を上書きできるかどうかを制御します。AWS OpsWorks スタック属性は通常、次のいずれかのタイプを使用します。

- `default` は、属性の上書きが許可されるため、最も一般的に使用されるタイプです。
- `normal` は、標準の AWS OpsWorks スタック属性値のいずれかを上書きする属性を定義します。

Note

Chef は追加のタイプをサポートしています。これは AWS OpsWorks スタックには必要ありませんが、プロジェクトに役立つ場合があります。詳細については、「[属性について](#)」を参照してください。

attribute name

属性名には、Chef の標準ノード構文である `[:attribute][:subattribute][...]` を使用します。属性には、希望する任意の名前を使用することができます。ただし、「[属性の上書き](#)」で説明されているように、カスタムクックブック属性は、スタック設定とデプロイ属性、および Chef の [Ohai ツール](#)とともに、インスタンスのノードオブジェクト内にマージされます。port や user などは、一般的に使用される設定名であるため、さまざまなクックブックに出現する可能性があります。

名前の競合を避けるためには、命名規則に従います。例に示すように、少なくとも 2 つの要素を使用して修飾した属性名を作成します。1 つ目は、Apache のように、通常は製品名に基づいた一意の要素を使用します。次に、`[:user]` や `[:port]` などの、特定の値を識別できる 1 つ以上のサブ属性が続きます。プロジェクトの要件に合った数のサブ属性を使用することができます。

value

属性は、次のようなタイプの値に設定することができます。

- `default[:apache][:keepalive] = 'Off'` のような文字列。
- `default[:apache][:timeout] = 120` のような数値 (引用符なし)。
- `true` または `false` のブール値 (引用符なし)。

- `default[:apache][:listen_ports] = ['80', '443']` のような値のリスト。

属性ファイルは Ruby アプリケーションです。そのため、ノード構文と論理演算子を使用して、他の属性に基づいた値を割り当てることもできます。属性の定義方法の詳細については、「[属性について](#)」を参照してください。動作する属性ファイルの例については、opsworks-cookbooks AWS OpsWorks のスタックの組み込みクックブックを参照してください。 <https://github.com/aws/>

テンプレート

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

多くのパッケージは、設定ファイルを作成してそれを適切なディレクトリに配置することによって設定されます。設定ファイルをクックブックに含めて目的のディレクトリにコピーすることもできますが、レシピを使用してテンプレートから設定ファイルを作成する方が、より柔軟性の高い方法です。テンプレートを使用する利点は、属性を使用してテンプレートの値を定義できることです。これにより、例えば、`custom` を使用して適切な属性値を JSON 上書きすることで、クックブックに触れずに設定ファイルを変更することができます。

テンプレートの内容と構造は、基本的に、関連付けられたファイルと同じです。以下に `httpd.conf` ファイルの例を示します。

```
ServerRoot "<%= node[:apache][:dir] %>"
<% if node[:platform] == "debian" || node[:platform] == "ubuntu" -%>
  LockFile /var/lock/apache2/accept.lock
<% else -%>
  LockFile logs/accept.lock
<% end -%>
PidFile <%= node[:apache][:pid_file] %>
Timeout <%= node[:apache][:timeout] %>
KeepAlive <%= node[:apache][:keepalive] %>
MaxKeepAliveRequests <%= node[:apache][:keepaliverequests] %>
```

```
KeepAliveTimeout <%= node[:apache][:keepalivetimeout] %>
<IfModule mpm_prefork_module>
  StartServers      <%= node[:apache][:prefork][:startservers] %>
  MinSpareServers   <%= node[:apache][:prefork][:minspareservers] %>
  MaxSpareServers   <%= node[:apache][:prefork][:maxspareservers] %>
  ServerLimit       <%= node[:apache][:prefork][:serverlimit] %>
  MaxClients        <%= node[:apache][:prefork][:maxclients] %>
  MaxRequestsPerChild <%= node[:apache][:prefork][:maxrequestsperschild] %>
</IfModule>
...

```

次の例は、Ubuntu インスタンス用に生成された httpd.conf ファイルです。

```
ServerRoot "/etc/httpd"
LockFile logs/accept.lock
PidFile /var/run/httpd/httpd.pid
Timeout 120
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 3
<IfModule mpm_prefork_module>
  StartServers      16
  MinSpareServers   16
  MaxSpareServers   32
  ServerLimit       400
  MaxClients        400
  MaxRequestsPerChild 10000
</IfModule>
...

```

テンプレートに含まれるテキストの多くは、テンプレートから httpd.conf ファイルに単にコピーされます。ただし、<%= ... %> の内容については、次のように処理されます。

- Chef によって、<%= node[:attribute][:sub_attribute][:...] %> が属性の値で置き換えられます。

例えば、StartServers <%= node[:apache][:prefork][:startservers] %> は、StartServers 16 では httpd.conf になります。

- 条件付きで値を選択するには、<%if-%>, <%else-%>, and <%end-%> を使用します。

この例では、プラットフォームに応じて、accept.lock の異なるファイルパスを設定します。

Note

使用できる属性は、クックブックの属性ファイルに設定された属性のみに制限されているわけではありません。インスタンスのノードオブジェクトで属性を使用できます。たとえば、[Ohai](#) と呼ばれる Chef ツールによって生成され、ノードオブジェクトに組み込まれます。属性の詳細については、「[属性の上書き](#)」を参照してください。

Ruby コードの組み込み方法など、テンプレートの詳細については、「[テンプレートについて](#)」を参照してください。

recipe

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

レシピは、システムの設定を定義する Ruby アプリケーションです。これを使用して、パッケージのインストール、テンプレートからの設定ファイルの作成、シェルコマンドの実行、ファイルとディレクトリの作成などを行います。通常 AWS OpsWorks、インスタンスで [ライフサイクルイベント](#) が発生したときにスタックでレシピを自動的に実行しますが、[Execute Recipes スタックコマンド](#) を使用していつでも明示的に実行することもできます。詳細については、「[レシピについて](#)」を参照してください。

通常、レシピの大部分は、それぞれがシステムのある側面の目的の状態を表す、一連のリソースで構成されます。各リソースには、目的の状態の定義と実行するアクションの指定を行う、一連の属性が含まれています。Chef によって、アクションを実行する適切なプロバイダーに各リソースが関連付けられます。詳細については、「[リソースおよびプロバイダーリファレンス](#)」を参照してください。

package リソースは、Linux インスタンスのソフトウェアパッケージを管理するのに役立ちます。次の例では、Apache パッケージをインストールします。


```
...
package 'apache2' do
  case node[:platform]
  when 'centos', 'redhat', 'fedora', 'amazon'
    package_name 'httpd'
  when 'debian', 'ubuntu'
    package_name 'apache2'
  end
  action :install
end
...
```

Chef は、プラットフォーム用の適切なパッケージプロバイダーを使用します。通常、リソース属性には、単に値が割り当てられるだけですが、Ruby の論理演算子を使用して、条件付き割り当てを実行することもできます。例では、`case` 演算子を使用しています。これは、`node[:platform]` を使用してインスタンスのオペレーティングシステムを識別し、それに応じて `package_name` 属性を設定します。標準の Chef ノード構文を使用して、レシピに属性を挿入できます。属性は Chef によって、関連付けられている値に置き換えられます。ノードオブジェクトでは、クックブックにある属性だけでなく、任意の属性を使用できます。

適切なパッケージ名を判別したら、コードセグメントはパッケージをインストールする `install` アクションを実行して終了します。このリソースに対するアクションには、他に `upgrade` や `remove` などがあります。詳細については、「[package](#)」を参照してください。

インストールや設定を行う複雑なタスクを 1 つ以上のサブタスクに分割し、それぞれを別個のレシピとして実装して、適切なタイミングでプライマリレシピで実行すると効率的です。次の例は、前述の例に続くコード行を示しています。

```
include_recipe 'apache2::service'
```

レシピで子レシピを実行するには、`include_recipe` キーワードを使用し、その後ろにレシピ名を指定します。レシピは、標準の Chef 構文である `CookbookName::RecipeName` を使用して識別されます。この `RecipeName` では `.rb` 拡張子が省略されます。

Note

`include_recipe` ステートメントは、プライマリレシピにおいて、その時点のレシピを効率的に実行します。ただし、実際には、プライマリレシピの実行前に、Chef によって各 `include_recipe` ステートメントが指定されたレシピのコードで置き換えられます。

`directory` リソースはディレクトリ (パッケージのファイルが含まれるディレクトリなど) を表します。次に示す `default.rb` リソースは、Linux ログディレクトリを作成します。

```
directory node[:apache][:log_dir] do
  mode 0755
  action :create
end
```

ログディレクトリは、クックブックの属性ファイルのいずれかで定義されています。リソースでは、ディレクトリのモードを 0755 に指定し、`create` アクションを使用してディレクトリを作成します。詳細については、「[directory](#)」を参照してください。また、Windows インスタンスでもこのリソースを使用できます。

`execute` リソースは、シェルコマンドやスクリプトなどのコマンドを表します。 `module.load` ファイルを生成する例を次に示します。

```
execute 'generate-module-list' do
  if node[:kernel][:machine] == 'x86_64'
    libdir = 'lib64'
  else
    libdir = 'lib'
  end
  command "/usr/local/bin/apache2_module_conf_generate.pl /usr/#{libdir}/httpd/modules /etc/httpd/mods-available"
  action :run
end
```

リソースが最初に CPU タイプを決定します。 `[:kernel][:machine]` は、Chef がさまざまなシステムプロパティを表すために生成する自動属性の 1 つであり、この場合は CPU タイプです。次に、コマンド (Perl スクリプト) を指定し、`run` アクションを使用してスクリプトを実行します。これにより、`module.load` ファイルが生成されます。詳細については、「[execute](#)」を参照してください。

`template` リソースは、クックブックのテンプレートファイルの 1 つから生成されるファイル (通常は設定ファイル) を意味します。次の例では、`httpd.conf` で説明した `apache2.conf.erb` テンプレートから [テンプレート](#) 設定ファイルが作成されます。

```
template 'apache2.conf' do
```

```
case node[:platform]
when 'centos','redhat','fedora','amazon'
  path "#{node[:apache][:dir]}/conf/httpd.conf"
when 'debian','ubuntu'
  path "#{node[:apache][:dir]}/apache2.conf"
end
source 'apache2.conf.erb'
owner 'root'
group 'root'
mode 0644
notifies :restart, resources(:service => 'apache2')
end
```

リソースは、インスタンスのオペレーティングシステムに基づいて、生成されるファイルの名前と場所を決定します。次に、ファイルの生成に使用するテンプレートとして `apache2.conf.erb` を指定し、ファイルの所有者、グループ、およびモードを設定します。notify アクションを実行して、Apache サーバーを表す `service` リソースに、サーバーを再起動するよう通知します。詳細については、「[template](#)」を参照してください。

スタック設定およびデプロイ属性: Linux

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このトピックでは、最もよく使われるスタック設定およびデプロイ属性と、関連するノード構文を紹介します。これは、Linux スタックにより使用されるスタック設定名前空間構造により整理されています。同じ属性名が異なる目的で使用され、異なる名前空間で発生する可能性があることに注意してください。例えば、`id` はスタック ID、レイヤー ID、アプリケーション ID などを示すことができるため、この属性値を使用するには完全修飾名が必要です。このデータを視覚化する便利な方法は、JSON オブジェクトとして表示することです。例については、「[スタック設定およびデプロイメント属性](#)」を参照してください。

Note

Linux インスタンスでは、AWS OpsWorks スタックはノードJSONオブジェクトにデータを追加することに加えて、各インスタンスにこのオブジェクトをインストールします。[エージェントの CLI `get_json` コマンド](#)を使用して取得できます。

トピック

- [opsworks 属性](#)
- [opsworks_custom_cookbooks 属性](#)
- [dependencies 属性](#)
- [ganglia 属性](#)
- [mysql 属性](#)
- [passenger 属性](#)
- [opsworks_bundler 属性](#)
- [deploy 属性](#)
- [その他の最上位属性](#)

opsworks 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

opsworks 要素 — opsworks 名前空間と呼ばれることもあります。基本的なスタック設定を定義する一連の属性が含まれます。

⚠ Important

opsworks 名前空間内の属性値を上書きすることは推奨されません。上書きすると、組み込みのレシピが失敗する可能性があります。

トピック

- [applications](#)
- [instance 属性](#)
- [レイヤー属性](#)
- [rails_stack 属性](#)
- [stack 属性](#)
- [その他の最上位の opsworks 属性](#)

applications

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

スタックに対して存在するアプリケーションごとに、1 個の埋め込みオブジェクトのリストが含まれます。各埋め込みオブジェクトには、アプリケーション設定を記述する次の属性が含まれます。

i Note

これらの属性の一般的なノード構文を次に示します。ここで、*i* はインスタンスの 0 から始まるリストインデックスを指定します。

```
node["opsworks"]["applications"]["i"]["attribute_name"]
```

application_type

アプリケーションのタイプ (文字列)。可能な値は以下のとおりです。

- php: PHPアプリ
- rails: Ruby on Rails アプリケーション
- java: Java アプリケーション
- nodejs: Node.js アプリケーション
- web: 静的HTMLページ
- other: その他のすべてのアプリケーションタイプ

```
node["opsworks"]["applications"]["i"]["application_type"]
```

名前

ユーザー定義の表示名 ("SimplePHP" など) (文字列)。

```
node["opsworks"]["applications"]["i"]["name"]
```

slug_name

短縮名。これは、アプリの名前 (文字列) OpsWorks から によって生成される "simplephp" などのすべて小文字の名前です。

```
node["opsworks"]["applications"]["i"]["slug_name"]
```

instance 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

instance 属性には、このインスタンスの設定を指定する属性のセットが含まれます。

architecture	availability_zone	backends
aws_instance_id	hostname	id
instance_type	ip	レイヤー
private_dns_name	private_ip	public_dns_name
region		

architecture

インスタンスのアーキテクチャ ("i386" など) (文字列)。

```
node["opsworks"]["instance"]["architecture"]
```

availability_zone

インスタンスのアベイラビリティゾーン ("us-west-2a" など) (文字列)。

```
node["opsworks"]["instance"]["availability_zone"]
```

backends

バックエンドウェブプロセスの数 (文字列)。例えば、HAProxyが Rails バックエンドに転送する同時接続の数を決定します。デフォルト値は、インスタンスのメモリおよびコアの数によって決まります。

```
node["opsworks"]["instance"]["backends"]
```

aws_instance_id

EC2 インスタンス ID (文字列)。

```
node["opsworks"]["instance"]["aws_instance_id"]
```

hostname

ホスト名 ("php-app1" など) (文字列)。

```
node["opsworks"]["instance"]["hostname"]
```

id

インスタンス ID。インスタンスを一意に識別 GUID する AWS OpsWorks スタックによって生成されます (文字列)。

```
node["opsworks"]["instance"]["id"]
```

instance_type

インスタンスのタイプ ("c1.medium" など) (文字列)。

```
node["opsworks"]["instance"]["instance_type"]
```

ip

パブリック IP アドレス (文字列)。

```
node["opsworks"]["instance"]["ip"]
```

レイヤー

短縮名で識別されるインスタンスのレイヤーのリスト ("lb"、"db-master" など) (文字列のリスト)。

```
node["opsworks"]["instance"]["layers"]
```

private_dns_name

プライベート DNS 名 (文字列)。

```
node["opsworks"]["instance"]["private_dns_name"]
```

private_ip

プライベート IP アドレス (文字列)。

```
node["opsworks"]["instance"]["private_ip"]
```


public_dns_name

パブリックDNS名 (文字列)。

```
node["opsworks"]["instance"]["public_dns_name"]
```

region

("us-west-2"文字列) などのAWSリージョン。

```
node["opsworks"]["instance"]["region"]
```

レイヤー属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

layers 属性には、スタックのレイヤーごとに 1 個のレイヤー属性のセットが含まれ、レイヤーの短縮名で名前が付けられます (php-app など)。スタックは組み込みレイヤーごとに最大 1 個を保持することができます。短縮名は次のとおりです。

- db-master: MySQLレイヤー
- java-app: Java アプリケーションサーバーレイヤー
- lb: HAProxy Layer
- monitoring-master: Gangliaレイヤー
- memcached: Memcachedレイヤー
- nodejs-app: Node.js アプリケーションサーバーレイヤー
- php-app: PHP アプリケーションサーバーレイヤー
- rails-app: Rails アプリケーションサーバーレイヤー

- web: 静的ウェブサーバーレイヤー

スタックには、ユーザー定義の短縮名を持つ、任意の数のカスタムレイヤーを含めることができます。

各レイヤー属性には、次の属性が含まれます。

- [id](#)
- [インスタンス](#)
- [\[名前\]](#)

id

レイヤー ID。によって生成 OpsWorks され、レイヤーを一意に識別GUIDする です (文字列)。

```
node["opsworks"]["layers"]["layershortname"]["id"]
```

インスタンス

instances 要素には、レイヤーのオンラインインスタンスごとに 1 個の instance 属性のセットが含まれます。これらは、php-app1 などインスタンスのホスト名で名前が付けられます。

Note

instances 要素には、特定のスタック設定およびデプロイ属性が作成された時点でオンライン状態にあったインスタンスのみが含まれています。

各インスタンス要素には、次の属性が含まれます。

availability_zone	aws_instance_id	backends
booted_at	created_at	elastic_ip
instance_type	ip	private_ip
public_dns_name	private_dns_name	region
ステータス		

availability_zone

アベイラビリティゾーン ("us-west-2a" など) (文字列)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]  
["availability_zone"]
```

aws_instance_id

EC2 インスタンス ID (文字列)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]  
["aws_instance_id"]
```

backends

バックエンドウェブプロセスの数 (数値)。例えば、HAProxyが Rails バックエンドに転送する同時接続の数を決定します。デフォルト値は、インスタンスのメモリおよびコアの数によって決まります。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]  
["backends"]
```

booted_at

yyyy-mm-ddThh:mm:ss+hh:mm UTC 形式 (文字列) を使用して EC2 インスタンスが起動された時刻。例えば、"2013-10-01T08:35:22+00:00" は 2013 年 10 月 10 日 8:35:22 (タイムゾーンオフセットなし) に対応します。詳細については、[ISO 「8601」](#) を参照してください。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]  
["booted_at"]
```

created_at

yyyy-mm-ddThh:mm:ss+hh:mm UTC 形式 (文字列) を使用して EC2 インスタンスが作成された時刻。例えば、"2013-10-01T08:35:22+00:00" は 2013 年 10 月 10 日 8:35:22 (タイムゾーンオフセットなし) に対応します。詳細については、[ISO 「8601」](#) を参照してください。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]  
["created_at"]
```

elastic_ip

Elastic IP アドレス (文字列)。インスタンスにアドレスがない場合は null に設定されます。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]  
["elastic_ip"]
```

instance_type

インスタンスのタイプ ("c1.medium" など) (文字列)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]  
["instance_type"]
```

ip

パブリック IP アドレス (文字列)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]  
["ip"]
```

private_ip

プライベート IP アドレス (文字列)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]  
["private_ip"]
```

public_dns_name

パブリックDNS名 (文字列)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]  
["public_dns_name"]
```

private_dns_name

プライベートDNS名 (文字列)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]  
["private_dns_name"]
```

region

("us-west-2"文字列) などのAWSリージョン。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]  
["region"]
```

ステータス

ステータス (文字列)。可能な値は以下のとおりです。

- "requested"
- "booting"
- "running_setup"
- "online"
- "setup_failed"
- "start_failed"
- "terminating"
- "terminated"
- "stopped"
- "connection_lost"

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]  
["status"]
```

名前

コンソール内のレイヤーを表すために使用されるレイヤーの名前 (文字列)。ユーザー定義も使用でき、一意である必要はありません。

```
node["opsworks"]["layers"]["layershortname"]["name"]
```

rails_stack 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソ

リユーシオンに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

名前

rails スタックを指定し、"apache_passenger" または "nginx_unicorn" に設定されます (文字列)。

```
node["opsworks"]["rails_stack"]["name"]
```

recipe

関連付けられたレシピ (文字列)。Passenger または Unicorn のどちらを使用しているかによって異なります。

- Unicorn: "unicorn::rails"
- Passenger: "passenger_apache2::rails"

```
node["opsworks"]["rails_stack"]["recipe"]
```

restart_command

再起動コマンド (文字列)。Passenger または Unicorn のどちらを使用しているかによって異なります。

- Unicorn: "../../shared/scripts/unicorn clean-restart"
- Passenger: "touch tmp/restart.txt"

service

サービス名 (文字列)。Passenger または Unicorn のどちらを使用しているかによって異なります。

- Unicorn: "unicorn"
- Passenger: "apache2"

```
node["opsworks"]["rails_stack"]["service"]
```

stack 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

stack 属性は、サービスレイヤー設定などスタック設定の一部の側面を指定します。

- [elb-load-balancers](#)
- [id](#)
- [名前](#)
- [rds_instances](#)
- [vpc_id](#)

elb-load-balancers

スタックの Elastic Load Balancing ロードバランサーごとに、1 個の埋め込みオブジェクトのリストが含まれます。各埋め込みオブジェクトには、ロードバランサー設定を記述する次の属性が含まれます。

Note

これらの属性の一般的なノード構文を次に示します。ここで、*i* はインスタンスの 0 から始まるリストインデックスを指定します。

```
node["opsworks"]["stack"]["elb-load-balancers"]["i"]["attribute_name"]
```

dns_name

ロードバランサーDNSの名前 (文字列)。

```
node["opsworks"]["stack"]["elb-load-balancers"]["i"]["dns_name"]
```

名前

ロードバランサーの名前 (文字列)。

```
node["opsworks"]["stack"]["elb-load-balancers"]["i"]["name"]
```

layer_id

ロードバランサーがアタッチされるレイヤーの ID (文字列)。

```
node["opsworks"]["stack"]["elb-load-balancers"]["i"]["layer_id"]
```

id

スタック ID (文字列)。

```
node["opsworks"]["stack"]["id"]
```

名前

スタック名 (文字列)。

```
node["opsworks"]["stack"]["name"]
```

rds_instances

スタックに登録されている Amazon RDS インスタンスごとに 1 つずつ、埋め込みオブジェクトのリストが含まれます。各埋め込みオブジェクトには、インスタンスの設定を定義する属性のセットが含まれます。これらの値は、Amazon RDS コンソールまたは API を使用してインスタンスを作成するときに指定します。Amazon RDS コンソールまたは API を使用して API、インスタンスの作成後に一部の設定を編集することもできます。詳細については、[Amazon の RDS ドキュメント](#)を参照してください。

Note

これらの属性の一般的なノード構文を次に示します。ここで、**i** はインスタンスの 0 から始まるリストインデックスを指定します。


```
node["opsworks"]["stack"]["rds_instances"]["i"]["attribute_name"]
```

スタックに複数の Amazon RDS インスタンスがある場合、レシピで特定のインスタンスを使用する方法の例を次に示します。

```
if my_rds = node["opsworks"]["stack"]["rds_instances"].select{|rds_instance|
  rds_instance["db_instance_identifier"] == 'db_id' }.first
  template "/etc/rds.conf" do
    source "rds.conf.erb"
    variables :address => my_rds["address"]
  end
end
```

アドレス	allocated_storage	arn
auto_minor_version_upgrade	availability_zone	backup_retention_period
db_instance_class	db_instance_identifier	db_instance_status
db_name	db_parameter_groups	db_security_groups
db_user	engine	instance_create_time
license_model	multi_az	option_group_memberships
port	preferred_backup_window	preferred_maintenance_window
publicly_accessible	read_replica_db_instance_identifiers	region
status_infos	vpc_security_groups	

アドレス

(opsinstance.ccdvt3hwog1a.us-west-2.rds.amazonaws.com文字列) URLなどのインスタンス。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["address"]
```

allocated_storage

割り当て済みストレージ (GB 単位) (数値)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["allocated_storage"]
```

arn

インスタンスの ARN (文字列) 。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["arn"]
```

auto_minor_version_upgrade

自動的にマイナーバージョンアップグレードを適用するかどうか (ブール)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["auto_minor_version_upgrade"]
```

availability_zone

インスタンスの Availability Zone (us-west-2a など) (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["availability_zone"]
```

backup_retention_period

バックアップ保持期間 (日数) (数値)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["backup_retention_period"]
```

db_instance_class

DB インスタンスクラス (db.m1.small など) (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_instance_class"]
```

db_instance_identifier

ユーザー定義の DB インスタンス識別子 (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_instance_identifier"]
```

db_instance_status

インスタンスのステータス (文字列)。詳細については、「[DB インスタンス](#)」を参照してください。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_instance_status"]
```

db_name

ユーザー定義の DB 名 (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_name"]
```

db_parameter_groups

パラメータグループごとに 1 個の埋め込みオブジェクトのリストを含む、インスタンスの DB パラメータグループ。詳細については、「[DB パラメータグループの操作](#)」を参照してください。各オブジェクトには、次の属性が含まれます。

db_parameter_group_name

グループ名 (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_parameter_groups"][j]
["db_parameter_group_name"]
```

parameter_apply_status

適用のステータス (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_parameter_groups"][j]
["parameter_apply_status"]
```

db_security_groups

セキュリティグループごとに 1 個の埋め込みオブジェクトのリストを含む、インスタンスのデータベースセキュリティグループ。詳細については、「[DB セキュリティグループの操作](#)」を参照してください。各オブジェクトには、次の属性が含まれます。

db_security_group_name

セキュリティグループ名 (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_security_groups"][j]
["db_security_group_name"]
```

ステータス

ステータス (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_security_groups"][j]
["status"]
```

db_user

ユーザー定義のマスターユーザー名 (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_user"]
```

engine

データベースエンジン (mysql(5.6.13) など) (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["engine"]
```

instance_create_time

インスタンス作成時間 (2014-04-15T16:13:34Z など) (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["instance_create_time"]
```

license_model

インスタンスのライセンスモデル (general-public-license など) (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["license_model"]
```

multi_az

マルチ AZ 配置を有効にするかどうか (ブール)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["multi_az"]
```

option_group_memberships

オプショングループごとに 1 個の埋め込みオブジェクトのリストを含む、インスタンスのオプショングループメンバーシップ。詳細については、「[オプショングループの操作](#)」を参照してください。各オブジェクトには、次の属性が含まれます。

option_group_name

グループの名前 (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["option_group_memberships"]  
[j]["option_group_name"]
```

ステータス

グループのステータス (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["option_group_memberships"]  
[j]["status"]
```

port

データベースサーバーのポート (数値)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["port"]
```

preferred_backup_window

日ごとの優先バックアップ時間 (06:26-06:56 など) (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["preferred_backup_window"]
```

preferred_maintenance_window

週ごとの優先メンテナンス時間 (thu:07:13-thu:07:43 など) (文字列)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["preferred_maintenance_window"]
```

publicly_accessible

データベースをパブリックにアクセス可能にするかどうか (ブール)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["publicly_accessible"]
```

read_replica_db_instance_identifiers

リードレプリカのインスタンス識別子のリスト (文字列のリスト)。詳細については、「[リードレプリカの使用](#)」を参照してください。

```
node["opsworks"]["stack"]["rds_instances"]["i"]  
["read_replica_db_instance_identifiers"]
```

region

(us-west-2文字列) などのAWSリージョン。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["region"]
```

status_infos

ステータス情報のリスト (文字列のリスト)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["status_infos"]
```

vpc_security_groups

VPC セキュリティグループのリスト (文字列のリスト)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["vpc_security_groups"]
```

vpc_id

VPC ID (文字列)。インスタンスnullがない場合、この値は ですVPC。

```
node["opsworks"]["stack"]["vpc_id"]
```

その他の最上位の opsworks 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、子属性のない opsworks 属性を紹介します。

activity

deploy (文字列) など、属性に関連付けられたアクティビティ。

```
node["opsworks"]["activity"]
```

agent_version

インスタンスの OpsWorks エージェントのバージョン (文字列)。

```
node["opsworks"]["agent_version"]
```

deploy_chef_provider

デプロイされたアプリケーションのディレクトリ構造に影響を与える Chef のデプロイプロバイダー (文字列)。この属性は、次のいずれかに設定できます。

- Branch
- Revision
- Timestamped (デフォルト値)

```
node["opsworks"]["deploy_chef_provider"]
```

ruby_stack

Ruby スタック (文字列)。デフォルト設定はエンタープライズバージョンです (ruby_enterprise)。MRI バージョンでは、この属性を に設定します ruby。

```
node["opsworks"]["ruby_stack"]
```

ruby_version

アプリケーションで使用される Ruby バージョン (文字列)。メジャーバージョンおよびマイナーバージョンのみを指定する場合に、この属性を使用できます。パッチバージョンを指定するには、該当する [\["ruby"\]](#) 属性を使用する必要があります。バージョンの指定方法と例については、「[Ruby のバージョン](#)」を参照してください。AWS OpsWorks スタックで Ruby バージョンを特定する方法の詳細については、組み込みの属性ファイル [ruby.rb](#) を参照してください。

```
node["opsworks"]["ruby_version"]
```

run_cookbook_tests

Chef 11.4 クックブックで [minitest-chef-handler](#) テストを実行するかどうか (ブール値)。

```
node["opsworks"]["run_cookbook_tests"]
```

sent_at

このコマンドがインスタンスにいつ送信されたか (数値)。

```
node["opsworks"]["sent_at"]
```

デプロイ

これらの属性がデプロイアクティビティに関連付けられている場合、deployment はデプロイを一意に識別 GUID する AWS OpsWorks スタックによって生成されたデプロイ ID (文字列) に設定されます。それ以外の場合、属性は null に設定されます。

```
node["opsworks"]["deployment"]
```


opsworks_custom_cookbooks 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

スタックのカスタムクックブックを指定する属性が含まれます。

有効

カスタムクックブックが有効になっているかどうか (ブール)。

```
node["opsworks_custom_cookbooks"]["enabled"]
```

recipes

このコマンドで実行されるレシピのリスト (文字列)。 *cookbookname::recipe* 形式を使用したカスタムレシピが含まれます。

```
node["opsworks_custom_cookbooks"]["recipes"]
```

dependencies 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

update_dependencies [スタックコマンド](#)に関連するいくつかの属性が含まれます。

gem_binary

Gems バイナリの場所 (文字列)。

upgrade_debs

Debs パッケージをアップグレードするかどうか (ブール)。

update_debs

Debs パッケージを更新するかどうか (ブール)。

ganglia 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Ganglia 統計ウェブページのアクセス方法を指定するいくつかの属性を含む、web 属性が含まれます。

password

統計ページにアクセスするために必要なパスワード (文字列)。

```
node["ganglia"]["web"]["password"]
```

url

(`"/ganglia"`文字列) などの統計ページの URL パス。完全な URL は `http://DNSNameURLPath`、`DNSName` は関連付けられたインスタンス DNS の名前です。

```
node["ganglia"]["web"]["url"]
```

ユーザー

統計ページにアクセスするために必要なユーザー名 (文字列)。

```
node["ganglia"]["web"]["user"]
```

mysql 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

MySQLデータベースサーバー設定を指定する属性のセットが含まれています。

clients

クライアント IP アドレスのリスト (文字列のリスト)。

```
node["mysql"]["clients"]
```

server_root_password

ルートパスワード (文字列)。

```
node["mysql"]["server_root_password"]
```

passenger 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Phusion Passenger 設定を指定する属性のセットが含まれます。

gem_bin

(`"/usr/local/bin/gem"` 文字列) RubyGems など、バイナリの場所。

```
node["passenger"]["gem_bin"]
```

max_pool_size

最大プールサイズ (数値)。

```
node["passenger"]["max_pool_size"]
```

ruby_bin

Ruby バイナリの場所。"`/usr/local/bin/ruby`" など。

```
node["passenger"]["ruby_bin"]
```

version

Passenger のバージョン (文字列)。

```
node["passenger"]["version"]
```

opsworks_bundler 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

[Bundler](#) サポートを指定する要素が含まれます。

manage_package

Bundler をインストールおよび管理するかどうか (ブール)。


```
node["opsworks_bundler"]["manage_package"]
```

version

Bundler のバージョン (文字列)。

```
node["opsworks_bundler"]["version"]
```

deploy 属性

 Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

属性が [Deploy イベント](#) または [Execute Recipes スタックコマンド](#) に関連付けられている場合、deploy 属性には、デプロイされた各アプリケーションの属性が含まれます (アプリケーションの短縮名から名前が付けられます)。各アプリケーション属性には、次の属性が含まれます。

アプリケーション	application_type	auto_bundle_on_deploy
データベース	deploy_to	domains
document_root	environment_variables	グループ
keep_releases	memcached	migrate
mounted_at	purge_before_symlink	rails_env
restart_command	scm	ssl_certificate

ssl_certificate_ca	ssl_certificate_key	ssl_support
スタック	symlink_before_migrate	symlinks
ユーザー		

アプリケーション

アプリケーションのslug名 ("simplephp" など) (文字列)。

```
node["deploy"]["appshortname"]["application"]
```

application_type

アプリケーションタイプ (文字列)。可能な値は以下のとおりです。

- java: Java アプリケーション
- nodejs: Node.js アプリケーション
- php: PHPアプリ
- rails: Ruby on Rails アプリケーション
- web: 静的HTMLページ
- other: その他のすべてのアプリケーションタイプ

```
node["deploy"]["appshortname"]["application_type"]
```

auto_bundle_on_deploy

Rails アプリケーションで、デプロイメント中に Bundler を実行するかどうか (ブール)。

```
node["deploy"]["appshortname"]["auto_bundle_on_deploy"]
```

データベース

アプリケーションのデータベースを接続するために必要な情報を含みます。アプリにデータベースレイヤーがアタッチされている場合、AWS OpsWorks スタックはこれらの属性に適切な値を自動的に割り当てます。

アダプター

データベースアダプタ (mysql など) (文字列)。

```
node["deploy"]["appshortname"]["database"]["adapter"]
```

データベース

データベース名 (文字列)。通常は "simplephp" のようなアプリケーションのプラグ名です。

```
node["deploy"]["appshortname"]["database"]["database"]
```

data_source_provider

データソース (mysql または rds) (文字列)。

```
node["deploy"]["appshortname"]["database"]["data_source_provider"]
```

ホスト

データベースホストの IP アドレス (文字列)。

```
node["deploy"]["appshortname"]["database"]["host"]
```

password

データベースのパスワード (文字列)。

```
node["deploy"]["appshortname"]["database"]["password"]
```

port

データベースポート (数値)。

```
node["deploy"]["appshortname"]["database"]["port"]
```

reconnect

Rails アプリケーションで、接続が存在しなくなった場合にアプリケーションを再接続するかどうか (ブール)。

```
node["deploy"]["appshortname"]["database"]["reconnect"]
```

username

ユーザー名 (文字列)。

```
node["deploy"]["appshortname"]["database"]["username"]
```

deploy_to

アプリケーションがデプロイされる場所 (文字列)。"/srv/www/simplephp"など。

```
node["deploy"]["appshortname"]["deploy_to"]
```

domains

アプリケーションのドメインのリスト (文字列のリスト)。

```
node["deploy"]["appshortname"]["domains"]
```

document_root

デフォルト以外のルートを指定する場合はドキュメントのルート、デフォルトルートを使用する場合は null (文字列)。

```
node["deploy"]["appshortname"]["document_root"]
```

environment_variables

アプリケーションに対して定義されたユーザー指定の環境変数を表す最大 20 個の属性の集合です。アプリケーション環境変数の定義方法の詳細については、「[アプリケーションの追加](#)」を参照してください。各属性名が環境変数名に設定され、対応する値が変数の値に設定されます。したがって以下の構文を使用して特定の値を参照できます。

```
node["deploy"]["appshortname"]["environment_variables"]["variable_name"]
```

グループ

アプリケーションのグループ (文字列)。

```
node["deploy"]["appshortname"]["group"]
```


keep_releases

AWS OpsWorks スタックが保存するアプリケーションのデプロイの数 (数値)。この属性は、アプリケーションをロールバックできる回数を制御します。デフォルトでは、グローバル値の [deploy_keep_releases](#) に設定されます。このデフォルト値は 5 です。keep_releases を上書きして、特定のアプリケーションの保存したデプロイメントの数を指定することができます。

```
node["deploy"]["appshortname"]["keep_releases"]
```

memcached

memcached 設定を定義する 2 つの属性が含まれます。

ホスト

Memcached サーバーインスタンスの IP アドレス (文字列)。

```
node["deploy"]["appshortname"]["memcached"]["host"]
```

port

memcached サーバーがリッスンするポート (数値)。

```
node["deploy"]["appshortname"]["memcached"]["port"]
```

migrate

Rails アプリケーションで、移行を実行するかどうか (ブール)。

```
node["deploy"]["appshortname"]["migrate"]
```

mounted_at

デフォルト以外のマウントポイントを指定する場合はアプリケーションのマウントポイント、デフォルトのマウントポイントを使用する場合は null です (文字列)。

```
node["deploy"]["appshortname"]["mounted_at"]
```

purge_before_symlink

Rails アプリケーションでは、シンボリックリンク (文字列のリスト) を作成する前に消去されるパスの配列。

```
node["deploy"]["appshortname"]["purge_before_symlink"]
```

rails_env

Rails アプリケーションサーバー インスタンスでは、"production" (文字列)などの rails 環境です。

```
node["deploy"]["appshortname"]["rails_env"]
```

restart_command

アプリケーションが再開されたときに実行されるコマンド ("echo 'restarting app'" など)。

```
node["deploy"]["appshortname"]["restart_command"]
```

scm

がソースコントロールリポジトリからアプリケーションをデプロイするために使用する情報 OpsWorksを指定する属性のセットが含まれています。この属性はリポジトリのタイプによって異なります。

password

プライベートリポジトリの場合はパスワード、パブリックリポジトリの場合は null (文字列)。プライベート Amazon S3 バケットでは、この属性はシークレットキーに設定されます。

```
node["deploy"]["appshortname"]["scm"]["password"]
```

リポジトリ

("git://github.com/amazonwebservicesservices/opsworks-demo-php-simple-app.git"文字列) URLなどのリポジトリ。

```
node["deploy"]["appshortname"]["scm"]["repository"]
```

revision

リポジトリに複数のブランチがある場合、その属性はアプリケーションのブランチまたはバージョンを指定します ("version1" など) (文字列)。それ以外の場合は、null に設定されます。

```
node["deploy"]["appshortname"]["scm"]["revision"]
```

scm_type

リポジトリのタイプ (文字列)。可能な値は以下のとおりです。

- "git": Git リポジトリ
- "svn": Subversion リポジトリ
- "s3": Amazon S3 バケット
- "archive": HTTPアーカイブ
- "other": 別のリポジトリのタイプ

```
node["deploy"]["appshortname"]["scm"]["scm_type"]
```

ssh_key

プライベート Git リポジトリにアクセスするための[デプロイSSHキー](#)、パブリックリポジトリの場合は null (文字列)。

```
node["deploy"]["appshortname"]["scm"]["ssh_key"]
```

ユーザー

プライベートリポジトリの場合はユーザー名、パブリックリポジトリの場合は null (文字列)。プライベート Amazon S3 バケットでは、この属性はアクセスキーに設定されます。

```
node["deploy"]["appshortname"]["scm"]["user"]
```

ssl_certificate

SSL サポートを有効にした場合はアプリケーションのSSL証明書、それ以外の場合は null (文字列)。

```
node["deploy"]["appshortname"]["ssl_certificate"]
```

ssl_certificate_ca

SSL が有効になっている場合、中間認証機関キーまたはクライアント認証 (文字列) を指定するための属性。

```
node["deploy"]["appshortname"]["ssl_certificate_ca"]
```

ssl_certificate_key

SSL サポートを有効にした場合はアプリケーションのSSLプライベートキー、それ以外の場合は null (文字列)。

```
node["deploy"]["appshortname"]["ssl_certificate_key"]
```

ssl_support

SSL がサポートされているかどうか (ブール値)。

```
node["deploy"]["appshortname"]["ssl_support"]
```

スタック

デプロイメント中にアプリケーションサーバーを再読み込みするかどうかを指定するブール属性 `needs_reload` が含まれます。

```
node["deploy"]["appshortname"]["stack"]["needs_reload"]
```

symlink_before_migrate

Rails アプリケーションでは、移行を実行する前に作成されるシンボリックリンクが "`link`": "`target`" ペアとして含まれます。

```
node["deploy"]["appshortname"]["symlink_before_migrate"]
```

symlinks

デプロイメントのシンボリックリンクが "`link`": "`target`" ペアとして含まれます。

```
node["deploy"]["appshortname"]["symlinks"]
```

ユーザー

アプリケーションのユーザー (文字列)。

```
node["deploy"]["appshortname"]["user"]
```

その他の最上位属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、子属性のない最上位スタック設定属性を紹介します。

rails 属性

サーバーの最大プールサイズを指定する `max_pool_size` 属性が含まれます (数値)。属性値は AWS OpsWorks スタックによって設定され、インスタンスタイプによって異なりますが、カスタムJSONまたはカスタム属性ファイルを使用して [上書き](#) できます。

```
node["rails"]["max_pool_size"]
```

recipes 属性

`"cookbookname::recipeName"` 形式を使用して、このアクティビティにより実行された組み込みレシピのリスト (文字列のリスト)。

```
node["recipes"]
```

opsworks_rubygems 属性

バージョンを指定する RubyGems バージョン要素が含まれます (文字列)。

```
node["opsworks_rubygems"]["version"]
```

languages 属性

インストールした各言語に対して、言語に由来する名前が付けられた属性が含まれます (ruby など)。この属性は、インストールフォルダ (など) を指定する属性 (`ruby_bin"/usr/bin/ruby"` など) を含むオブジェクトです。

ssh_users 属性

一連の属性が含まれ、各属性はアクセスSSH許可が付与されたユーザーの1つを記述します。各属性には、ユーザーの Unix ID で名前が付けられます。AWS OpsWorks スタックは、など、2000-4000 の範囲の各ユーザーに一意的 ID を生成し"2001"、その ID を持つユーザーをすべてのインスタンスに作成します。AWS OpsWorks は 2000-4000 の範囲を予約するため、の外部で作成したユーザー AWS OpsWorks (クックブックレシピを使用するか、AWS OpsWorks からユーザーをインポートするなど) はIAM、別のユーザーの AWS OpsWorks スタックによって上書きUIDsされるを持つことができます。ベストプラクティスとして、AWS OpsWorks スタックコンソールでユーザーを作成し、アクセスを管理します。AWS OpsWorks スタックの外部でユーザーを作成する場合は、4000 を超える *UnixID* 値を使用します。

各属性には、次の属性が含まれます。

email

ユーザーの E メールアドレス (文字列)。

```
node["ssh_users"]["UnixID"]["email"]
```

public_key

ユーザーのパブリックSSHキー (文字列) 。

```
node["ssh_users"]["UnixID"]["public_key"]
```

sudoer

ユーザーに sudo アクセス許可があるかどうか (ブール)。

```
node["ssh_users"]["UnixID"]["sudoer"]
```

名前

ユーザー名 (文字列)。

```
node["ssh_users"]["UnixID"]["name"]
```

組み込みクックブックの属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

これらの属性のほとんどは Linux スタックにのみ使用できます。

組み込みのレシピのほとんどには、さまざまな設定を定義する 1 つ以上の[属性ファイル](#)があります。カスタムレシピでこれらの設定にアクセスし、カスタム JSON を使用して上書きできます。通常、AWS OpsWorks スタックでサポートされているさまざまなサーバーテクノロジーの設定を制御する属性にアクセスまたは上書きする必要があります。このセクションでは、これらの属性の概要を示します。完全な属性ファイル、および関連するレシピとテンプレートは、<https://github.com/aws/opsworks-cookbooks.git> で入手できます。

Note

組み込みのレシピの属性はすべて default 型です。

トピック

- [apache2 属性](#)
- [deploy 属性](#)
- [haproxy 属性](#)
- [memcached 属性](#)
- [mysql 属性](#)
- [nginx 属性](#)

- [opsworks_berkshelf 属性](#)
- [opsworks_java 属性](#)
- [passenger_apache2 属性](#)
- [ruby 属性](#)
- [unicorn 属性](#)

apache2 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

これらの属性は Linux スタックにのみ使用できます。

[apache2 属性](#) は [Apache HTTPサーバー](#) 設定を指定します。詳細については、「[Apache コア機能](#)」を参照してください。組み込み属性を上書きしてカスタム値を指定する方法の詳細については、「[属性の上書き](#)」を参照してください。

バイナリ	contact	deflate_types
dir	document_root	グループ
hide_info_headers	icondir	init_script
keepalive	keepaliverequests	keepalivetimeout
lib_dir	libexecdir	listen_ports
log_dir	logrotate 属性	pid_file

prefork 属性	serversignature	servertokens
timeout	traceenable	ユーザー
version	worker 属性	

バイナリ

Apache バイナリの場所 (文字列)。デフォルト値は `'/usr/sbin/httpd'` です。

```
node[:apache][:binary]
```

contact

Eメールの連絡先 (文字列)。デフォルト値はダミーのアドレス `'ops@example.com'` です。

```
node[:apache][:contact]
```

deflate_types

Mime タイプがブラウザでサポートされている場合は、指定した Mime タイプで圧縮を有効にするために `mod_deflate` を指定します (文字列のリスト)。デフォルト値は次のとおりです。

```
['application/javascript',  
'application/json',  
'application/x-javascript',  
'application/xhtml+xml',  
'application/xml',  
'application/xml+rss',  
'text/css',  
'text/html',  
'text/javascript',  
'text/plain',  
'text/xml']
```

Warning

圧縮によってセキュリティリスクが生じる可能性があります。圧縮を完全に無効にするには、この属性を次のように設定します。

```
node[:apache][:deflate_types] = []
```

```
node[:apache][:deflate_types]
```

dir

サーバーのルートディレクトリ (文字列)。デフォルト値は次のとおりです。

- Amazon Linux および Red Hat Enterprise Linux (RHEL): '/etc/httpd'
- Ubuntu: '/etc/apache2'

```
node[:apache][:dir]
```

document_root

ドキュメントのルート (文字列)。デフォルト値は次のとおりです。

- Amazon Linux および RHEL: '/var/www/html'
- Ubuntu: '/var/www'

```
node[:apache][:document_root]
```

グループ

グループ名 (文字列)。デフォルト値は次のとおりです。

- Amazon Linux および RHEL: 'apache'
- Ubuntu: 'www-data'

```
node[:apache][:group]
```

hide_info_headers

HTTP ヘッダー ('true'/'false') からバージョンとモジュール情報を省略するかどうか (文字列)。デフォルト値は 'true' です。

```
node[:apache][:hide_info_headers]
```

icondir

アイコンのディレクトリ (文字列)。デフォルト値は次のとおりです。

- Amazon Linux および RHEL: '/var/www/icons/'
- Ubuntu: '/usr/share/apache2/icons'

```
node[:apache][:icondir]
```

init_script

初期化スクリプト (文字列)。デフォルト値は次のとおりです。

- Amazon Linux および RHEL: '/etc/init.d/httpd'
- Ubuntu: '/etc/init.d/apache2'

```
node[:apache][:init_script]
```

keepalive

キープアライブ接続を有効にするかどうか (文字列)。指定できる値は 'On' および 'Off' です (文字列)。デフォルト値は 'Off' です。

```
node[:apache][:keepalive]
```

keepaliverequests

Apache が同時に処理するキープアライブリクエストの最大数 (数値)。デフォルト値は 100 です。

```
node[:apache][:keepaliverequests]
```

keepalivetimeout

Apache は、接続を閉じる前にリクエストを待機します (数値)。デフォルト値は 3 です。

```
node[:apache][:keepalivetimeout]
```

lib_dir

オブジェクトコードライブラリを含むディレクトリ (文字列)。デフォルト値は次のとおりです。

- Amazon Linux (x86): '/usr/lib/httpd'
- Amazon Linux (x64) および RHEL: '/usr/lib64/httpd'
- Ubuntu: '/usr/lib/apache2'

```
node[:apache][:lib_dir]
```

libexecdir

プログラム実行可能ファイルを含むディレクトリ (文字列)。デフォルト値は次のとおりです。

- Amazon Linux (x86): '/usr/lib/httpd/modules'
- Amazon Linux (x64) および RHEL: '/usr/lib64/httpd/modules'
- Ubuntu: '/usr/lib/apache2/modules'

```
node[:apache][:libexecdir]
```

listen_ports

サーバーがリッスンするポートのリスト (文字列のリスト)。デフォルト値は ['80', '443'] です。

```
node[:apache][:listen_ports]
```

log_dir

ログディレクトリ (文字列)。デフォルト値は次のとおりです。

- Amazon Linux および RHEL: '/var/log/httpd'
- Ubuntu: '/var/log/apache2'

```
node[:apache][:log_dir]
```

logrotate 属性

これらの属性は、ログファイルを更新する方法を指定します。

delaycompress

次のローテーションサイクルの開始まで、閉じられたログファイルの圧縮を遅らせるかどうか ('true'/'false') (文字列)。デフォルト値は 'true' です。

```
node[:apache][:logrotate][:delaycompress]
```

グループ

ログファイルのグループ (文字列)。デフォルト値は 'adm' です。

```
node[:apache][:logrotate][:group]
```

モード

ログファイルのモード (文字列)。デフォルト値は '640' です。

```
node[:apache][:logrotate][:mode]
```

owner (オーナー)

ログファイルの所有者 (文字列)。デフォルト値は 'root' です。

```
node[:apache][:logrotate][:owner]
```

rotate

閉じられたログファイルが削除される前のローテーションサイクルの数 (文字列)。デフォルト値は '30' です。

```
node[:apache][:logrotate][:rotate]
```

schedule

更新スケジュール (文字列)。可能な値は以下のとおりです。

- 'daily'
- 'weekly'
- 'monthly'

デフォルト値は 'daily' です。

```
node[:apache][:logrotate][:schedule]
```

pid_file

デーモンのプロセス ID を含むファイル (文字列)。デフォルト値は次のとおりです。

- Amazon Linux および RHEL: `'/var/run/httpd/httpd.pid'`
- Ubuntu: `'/var/run/apache2.pid'`

```
node[:apache][:pid_file]
```

prefork 属性

これらの属性は事前分岐設定を指定します。

maxclients

供給される同時リクエストの最大数 (数値)。デフォルト値は 400 です。

Note

この属性は、Amazon Linux または を実行しているインスタンスにのみ使用しません RHEL。インスタンスが Ubuntu 14.04 を実行している場合は LTS、 を使用しません [maxrequestworkers](#)。

```
node[:apache][:prefork][:maxclients]
```

maxrequestspchild

子サーバープロセスが処理するリクエストの最大数 (数値)。デフォルト値は 10000 です。

```
node[:apache][:prefork][:maxrequestspchild]
```

maxrequestworkers

供給される同時リクエストの最大数 (数値)。デフォルト値は 400 です。

Note

この属性は、Ubuntu 14.04 を実行しているインスタンスにのみ使用しません LTS。インスタンスで Amazon Linux または を実行している場合は RHEL、 を使用しません [maxclients](#)。

```
node[:apache][:prefork][:maxrequestworkers]
```

maxspareservers

アイドル状態の子サーバープロセスの最大数 (数値)。デフォルト値は 32 です。

```
node[:apache][:prefork][:maxspareservers]
```

minspareservers

アイドル状態の子サーバープロセスの最小数 (数値)。デフォルト値は 16 です。

```
node[:apache][:prefork][:minspareservers]
```

serverlimit

設定できるプロセスの最大数 (数値)。デフォルト値は 400 です。

```
node[:apache][:prefork][:serverlimit]
```

startservers

起動時に作成される子サーバープロセスの数 (数値)。デフォルト値は 16 です。

```
node[:apache][:prefork][:startservers]
```

serversignature

サーバーで生成されたドキュメントの末尾のフッターを設定するかどうか、および設定方法を指定します (文字列)。指定できる値は 'On'、'Off'、および 'Email' です。デフォルト値は 'Off' です。

```
node[:apache][:serversignature]
```

servertokens

レスポンスヘッダーに含まれるサーバーバージョン情報のタイプを指定します (文字列)。

- 'Full': 完全な情報。例えば、サーバー: Apache/2.4.2 (Unix) PHP/4.2.2 MyMod/1.2

- 'Prod': 製品名。たとえば、「Server: Apache」のようになります。
- 'Major': メジャーバージョン。たとえば、「Server: Apache/2」のようになります。
- 'Minor': メジャーバージョンとマイナーバージョン。たとえば、「Server: Apache/2.4」のようになります。
- 'Min': 最小バージョン。たとえば、「Server: Apache/2.4.2」のようになります。
- 'OS': バージョンとオペレーティングシステム。たとえば、「Server: Apache/2.4.2 (Unix)」のようになります。

デフォルト値は 'Prod' です。

```
node[:apache][:servertokens]
```

timeout

Apache が I/O を待機する時間 (数値)。デフォルト値は 120 です。

```
node[:apache][:timeout]
```

traceenable

TRACE リクエストを有効にするかどうか (文字列)。指定できる値は 'On' および 'Off' です。デフォルト値は 'Off' です。

```
node[:apache][:traceenable]
```

ユーザー

ユーザー名 (文字列)。デフォルト値は次のとおりです。

- Amazon Linux および RHEL: 'apache'
- Ubuntu: 'www-data'

```
node[:apache][:user]
```

version

Apache のバージョン (文字列)。デフォルト値は次のとおりです。

- Amazon Linux: 2.2
- Ubuntu 14.04LTS: 2.4
- RHEL: 2.4

```
node[:apache][:version]
```

worker 属性

これらの属性は、ワーカースタンププロセス設定を指定します。

startservers

起動時に作成される子サーバースタンププロセスの数 (数値)。デフォルト値は 4 です。

```
node[:apache][:worker][:startservers]
```

maxclients

供給される同時リクエストの最大数 (数値)。デフォルト値は 1024 です。

```
node[:apache][:worker][:maxclients]
```

maxsparethreads

アイドル状態のスレッドの最大数 (数値)。デフォルト値は 192 です。

```
node[:apache][:worker][:maxsparethreads]
```

minsparethreads

アイドル状態のスレッドの最小数 (数値)。デフォルト値は 64 です。

```
node[:apache][:worker][:minsparethreads]
```

threadsperchild

子プロセスごとのスレッドの数 (数値)。デフォルト値は 64 です。

```
node[:apache][:worker][:threadsperchild]
```

maxrequestperchild

子サーバープロセスが処理するリクエストの最大数 (数値)。デフォルト値は 10000 です。

```
node[:apache][:worker][:maxrequestperchild]
```

deploy 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[組み込みのデプロイクックブックの deploy.rb 属性ファイル](#)は、opsworks 名前空間の以下の属性を定義します。デプロイディレクトリの詳細については、「[Deploy レシピ](#)」を参照してください。組み込み属性を上書きしてカスタム値を指定する方法の詳細については、「[属性の上書き](#)」を参照してください。

deploy_keep_releases

AWS OpsWorks スタックが保存するアプリケーションデプロイの数のグローバル設定 (数値)。デフォルト値は 5 です。この値は、アプリケーションをロールバックできる回数を制御します。

```
node[:opsworks][:deploy_keep_releases]
```

グループ

(Linux のみ) アプリケーションのデプロイディレクトリに対する group 設定 (文字列)。デフォルト値は、インスタンスのオペレーティングシステムによって決まります。

- Ubuntu インスタンスでは、デフォルト値は www-data です。
- Nginx と Unicorn を使用する Rails App Server レイヤーのメンバーである Amazon Linux または RHEL インスタンスの場合、デフォルト値は nginx です。
- 他のすべての Amazon Linux または RHEL インスタンスの場合、デフォルト値は apache です。

```
node[:opsworks][:deploy_user][:group]
```

ユーザー

(Linux のみ) アプリケーションのデプロイディレクトリに対する user 設定 (文字列)。デフォルト値は deploy です。

```
node[:opsworks][:deploy_user][:user]
```

haproxy 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

これらの属性は Linux スタックにのみ使用できます。

[haproxy 属性](#) は [HAProxy サーバー](#) 設定を指定します。詳細については、[HAProxy 「Docs」](#) を参照してください。組み込み属性を上書きしてカスタム値を指定する方法の詳細については、「[属性の上書き](#)」を参照してください。

balance	check_interval	client_timeout
connect_timeout	default_max_connections	global_max_connections
health_check_method	health_check_url	queue_timeout
http_request_timeout	maxcon_factor_nodejs_app	maxcon_factor_nodejs_app_ss !

maxcon_factor_php_app	maxcon_factor_php_app_ssl	maxcon_factor_rails_app
maxcon_factor_rails_app_ssl	maxcon_factor_static	maxcon_factor_static_ssl
retries	server_timeout	stats_url
stats_user		

balance

サーバーを選択するためにロードバランサーが使用するアルゴリズムです (文字列)。デフォルト値は 'roundrobin' です。その他のオプションは次のとおりです。

- 'static-rr'
- 'leastconn'
- 'source'
- 'uri'
- 'url_param'
- 'hdr(name)'
- 'rdp-cookie'
- 'rdp-cookie(name)'

これらの引数の詳細については、「[balance](#)」を参照してください。

```
node[:haproxy][:balance]
```

check_interval

ヘルスチェックの間隔 (文字列)。デフォルト値は '10s' です。

```
node[:haproxy][:check_interval]
```

client_timeout

クライアントを非アクティブにすることができる最大時間 (文字列)。デフォルト値は '60s' です。

```
node[:haproxy][:client_timeout]
```

connect_timeout

HAProxy がサーバー接続の試行が成功するまでの最大待機時間 (文字列)。デフォルト値は '10s' です。

```
node[:haproxy][:connect_timeout]
```

default_max_connections

接続の最大数のデフォルト値 (文字列)。デフォルト値は '80000' です。

```
node[:haproxy][:default_max_connections]
```

global_max_connections

接続の最大数 (文字列)。デフォルト値は '80000' です。

```
node[:haproxy][:global_max_connections]
```

health_check_method

ヘルスチェックの方法 (文字列)。デフォルト値は 'OPTIONS' です。

```
node[:haproxy][:health_check_method]
```

health_check_url

サーバーのヘルスをチェックするために使用されるURLパス (文字列)。デフォルト値は '/' です。

```
node[:haproxy][:health_check_url ]
```

queue_timeout

無料接続を待機する最大時間 (文字列)。デフォルト値は '120s' です。

```
node[:haproxy][:queue_timeout]
```

http_request_timeout

HAProxy が完全なHTTPリクエストを待機する最大時間 (文字列)。デフォルト値は '30s' です。

```
node[:haproxy][:http_request_timeout]
```

retries

サーバー接続が失敗した後の再試行の数 (文字列)。デフォルト値は '3' です。

```
node[:haproxy][:retries]
```

server_timeout

クライアントを非アクティブにすることができる最大時間 (文字列)。デフォルト値は '60s' です。

```
node[:haproxy][:server_timeout]
```

stats_url

統計ページのURLパス (文字列)。デフォルト値は '/haproxy?stats' です。

```
node[:haproxy][:stats_url]
```

stats_user

統計ページのユーザー名 (文字列)。デフォルト値は 'opsworks' です。

```
node[:haproxy][:stats_user]
```

maxcon 属性は、[バックエンド](#) HAProxy に許可する接続の最大数を計算するために使用される負荷係数乗数を表します。例えば、backend 値が 4 の小さなインスタンスに Rails アプリケーションサーバーがあるとします。つまり、スタックはそのインスタンスに 4 AWS OpsWorks つの Rails プロセスを設定します。maxcon_factor_rails_app デフォルト値の 7 を使用すると、HAProxy は Rails サーバーへの 28 (4*7) 接続を処理します。

maxcon_factor_nodejs_app

Node.js アプリケーションサーバーの maxcon 要素 (数値)。デフォルト値は 10 です。

```
node[:haproxy][:maxcon_factor_nodejs_app]
```

maxcon_factor_nodejs_app_ssl

を持つ Node.js アプリケーションサーバーの maxcon 係数 SSL (数値)。デフォルト値は 10 です。

```
node[:haproxy][:maxcon_factor_nodejs_app_ssl]
```

maxcon_factor_php_app

PHP アプリケーションサーバーの maxcon 係数 (数値)。デフォルト値は 10 です。

```
node[:haproxy][:maxcon_factor_php_app]
```

maxcon_factor_php_app_ssl

(SSL数値) を持つPHPアプリケーションサーバーの maxcon 係数。デフォルト値は 10 です。

```
node[:haproxy][:maxcon_factor_php_app_ssl]
```

maxcon_factor_rails_app

Rails アプリケーションサーバーの maxcon 要素 (数値)。デフォルト値は 7 です。

```
node[:haproxy][:maxcon_factor_rails_app]
```

maxcon_factor_rails_app_ssl

Rails アプリサーバーの SSL maxcon 係数 (数値)。デフォルト値は 7 です。

```
node[:haproxy][:maxcon_factor_rails_app_ssl]
```

maxcon_factor_static

静的ウェブサーバーの maxcon 要素 (数値)。デフォルト値は 15 です。

```
node[:haproxy][:maxcon_factor_static]
```

maxcon_factor_static_ssl

(SSL数値) を持つ静的ウェブサーバーの maxcon 係数。デフォルト値は 15 です。

```
node[:haproxy][:maxcon_factor_static_ssl]
```

memcached 属性

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

ℹ Note

これらの属性は Linux スタックにのみ使用できます。

[memcached 属性](#) は [Memcached](#) サーバー設定を指定します。組み込み属性を上書きしてカスタム値を指定する方法の詳細については、「[属性の上書き](#)」を参照してください。

メモリ	max_connections	pid_file
port	start_command	stop_command
ユーザー		

メモリ

使用する最大メモリ (MB 単位) (数値)。デフォルト値は 512 です。


```
node[:memcached][:memory]
```

max_connections

接続の最大数 (文字列)。デフォルト値は '4096' です。

```
node[:memcached][:max_connections]
```

pid_file

デーモンのプロセス ID を含むファイル (文字列)。デフォルト値は 'var/run/memcached.pid' です。

```
node[:memcached][:pid_file]
```

port

リッスンするポート (数値)。デフォルト値は 11211 です。

```
node[:memcached][:port]
```

start_command

開始コマンド (文字列)。デフォルト値は '/etc/init.d/memcached start' です。

```
node[:memcached][:start_command]
```

stop_command

停止コマンド (文字列)。デフォルト値は '/etc/init.d/memcached stop' です。

```
node[:memcached][:stop_command]
```

ユーザー

ユーザー (文字列)。デフォルト値は 'nobody' です。

```
node[:memcached][:user]
```

mysql 属性

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

ℹ Note

これらの属性は Linux スタックにのみ使用できます。

[mysql 属性](#)は、[MySQL](#) マスター設定を指定します。詳細については、「[サーバーシステム変数](#)」を参照してください。組み込み属性を上書きしてカスタム値を指定する方法の詳細については、「[属性の上書き](#)」を参照してください。

basedir	bind_address	clients
conf_dir	confd_dir	datadir
grants_path	mysql_bin	mysqladmin_bin
pid_file	port	root_group
server_root_password	ソケット	tunable 属性

basedir

ベースディレクトリ (文字列)。デフォルト値は '/usr' です。

```
node[:mysql][:basedir]
```

bind_address

リSQLツスするアドレス (文字列)。デフォルト値は '0.0.0.0' です。

```
node[:mysql][:bind_address]
```

clients

クライアントのリスト (文字列のリスト)。

```
node[:mysql][:clients]
```

conf_dir

設定ファイルを含むディレクトリ (文字列)。デフォルト値は次のとおりです。

- Amazon Linux および RHEL: '/etc'
- Ubuntu: '/etc/mysql'

```
node[:mysql][:conf_dir]
```

confd_dir

追加の設定ファイルを含むディレクトリ (文字列)。デフォルト値は '/etc/mysql/conf.d' です。

```
node[:mysql][:confd_dir]
```

datadir

データディレクトリ (文字列)。デフォルト値は '/var/lib/mysql' です。

```
node[:mysql][:datadir]
```

grants_path

許可のテーブルの場所 (文字列)。デフォルト値は '/etc/mysql_grants.sql' です。

```
node[:mysql][:grants_path]
```

mysql_bin

mysql バイナリの場所 (文字列)。デフォルト値は '/usr/bin/mysql' です。

```
node[:mysql][:mysql_bin]
```

mysqladmin_bin

mysqladmin の場所 (文字列)。デフォルト値は '/usr/bin/mysqladmin' です。

```
node[:mysql][:mysqladmin_bin]
```

pid_file

デーモンのプロセス ID を含むファイル (文字列)。デフォルト値は '/var/run/mysqld/mysqld.pid' です。

```
node[:mysql][:pid_file]
```

port

サーバーがリッスンするポート (数値)。デフォルト値は 3306 です。

```
node[:mysql][:port]
```

root_group

ルートグループ (文字列)。デフォルト値は 'root' です。

```
node[:mysql][:root_group]
```

server_root_password

サーバーのルートパスワード (文字列)。デフォルト値はランダムに生成されます。

```
node[:mysql][:server_root_password]
```

ソケット

ソケットファイルの場所 (文字列)。デフォルト値は '/var/lib/mysql/mysql.sock' です。デフォルト値は次のとおりです。

- Amazon Linux および RHEL: '/var/lib/mysql/mysql.sock'
- Ubuntu: '/var/run/mysqld/mysqld.sock'

```
node[:mysql][:socket]
```

tunable 属性

調整可能な属性はパフォーマンス調整に使用されます。

back_log	innodb_additional_mem_pool_size	innodb_buffer_pool_size
innodb_flush_log_at_trx_commit	innodb_lock_wait_timeout	key_buffer
log_slow_queries	long_query_time	max_allowed_packet
max_connections	max_heap_table_size	net_read_timeout
net_write_timeout	query_cache_limit	query_cache_size
query_cache_type	thread_cache_size	thread_stack
wait_timeout	table_cache	

back_log

未処理のリクエストの最大数 (文字列)。デフォルト値は '128' です。

```
node[:mysql][:tunable][:back_log]
```

innodb_additional_mem_pool_size

[Innodb](#) が内部データ構造を保存するために使用するプールのサイズ (文字列)。デフォルト値は '20M' です。

```
node[:mysql][:tunable][:innodb_additional_mem_pool_size]
```

innodb_buffer_pool_size

[Innodb](#) バッファプールサイズ (文字列)。属性値は AWS OpsWorks スタックによって設定され、インスタンスタイプによって異なりますが、カスタムJSONまたはカスタム属性ファイルを使用して[上書き](#)できます。

```
node[:mysql][:tunable][:innodb_buffer_pool_size]
```

innodb_flush_log_at_trx_commit

[Innodb](#) がログバッファをフラッシュする頻度 (文字列)。デフォルト値は '2' です。詳細については、「[innodb_flush_log_at_trx_commit](#)」を参照してください。

```
node[:mysql][:tunable][:innodb_flush_log_at_trx_commit]
```

innodb_lock_wait_timeout

[Innodb](#) トランザクションが行ロックを待機する最大時間 (秒数) (文字列)。デフォルト値は '50' です。

```
node[:mysql][:tunable][:innodb_lock_wait_timeout]
```

key_buffer

インデックスバッファサイズ (文字列)。デフォルト値は '250M' です。

```
node[:mysql][:tunable][:key_buffer]
```

log_slow_queries

スロークエリのログファイルの場所 (文字列)。デフォルト値は '/var/log/mysql/mysql-slow.log' です。

```
node[:mysql][:tunable][:log_slow_queries]
```

long_query_time

クエリを長いクエリとして指定するための時間 (秒) (文字列)。デフォルト値は '1' です。

```
node[:mysql][:tunable][:long_query_time]
```

max_allowed_packet

許容される最大パケットサイズ (文字列)。デフォルト値は '32M' です。

```
node[:mysql][:tunable][:max_allowed_packet]
```

max_connections

同時クライアント接続の最大数 (文字列)。デフォルト値は '2048' です。

```
node[:mysql][:tunable][:max_connections]
```

max_heap_table_size

ユーザーが作成した MEMORY テーブルの最大サイズ (文字列)。デフォルト値は '32M' です。

```
node[:mysql][:tunable][:max_heap_table_size]
```

net_read_timeout

接続からの追加のデータを待機する時間 (秒) (文字列) デフォルト値は '30' です。

```
node[:mysql][:tunable][:net_read_timeout]
```

net_write_timeout

接続に書き込まれるブロックを待機する時間 (秒) (文字列)。デフォルト値は '30' です。

```
node[:mysql][:tunable][:net_write_timeout]
```

query_cache_limit

キャッシュした各クエリの最大サイズ (文字列)。デフォルト値は '2M' です。

```
node[:mysql][:tunable][:query_cache_limit]
```

query_cache_size

クエリのキャッシュのサイズ (文字列)。デフォルト値は '128M' です。

```
node[:mysql][:tunable][:query_cache_size]
```

query_cache_type

クエリのキャッシュのタイプ (文字列)。指定できる値は次のとおりです。

- '0': キャッシュなし、またはキャッシュしたデータの取得なし。
- '1': SELECT SQL_NO_CACHE で開始されないキャッシュのステートメント。
- '2': SELECT SQL_CACHE で開始されるキャッシュのステートメント。

デフォルト値は '1' です。

```
node[:mysql][:tunable][:query_cache_type]
```

thread_cache_size

再利用のためにキャッシュされるクライアントのスレッドの数 (文字列)。デフォルト値は '8' です。

```
node[:mysql][:tunable][:thread_cache_size]
```

thread_stack

各スレッドのスタックのサイズ (文字列)。デフォルト値は '192K' です。

```
node[:mysql][:tunable][:thread_stack]
```

wait_timeout

非対話形式の接続で待機する時間 (秒)。デフォルト値は '180' です (文字列)。

```
node[:mysql][:tunable][:wait_timeout]
```

table_cache

開いたテーブルの数 (文字列)。デフォルト値は '2048' です。

```
node[:mysql][:tunable][:table_cache]
```

nginx 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

これらの属性は Linux スタックにのみ使用できます。

[nginx 属性](#)は [Nginx](#) 設定を指定します。詳細については、「[ディレクティブインデックス](#)」を参照してください。組み込み属性を上書きしてカスタム値を指定する方法の詳細については、「[属性の上書き](#)」を参照してください。

バイナリ	dir	gzip
gzip_comp_level	gzip_disable	gzip_http_version
gzip_proxied	gzip_static	gzip_types
gzip_vary	keepalive	keepalive_timeout
log_dir	ユーザー	server_names_hash_bucket_size
worker_processes	worker_connections	

バイナリ

Nginx バイナリの場所 (文字列)。デフォルト値は '/usr/sbin/nginx' です。

```
node[:nginx][:binary]
```

dir

設定ファイルなどのファイルの場所 (文字列)。デフォルト値は '/etc/nginx' です。

```
node[:nginx][:dir]
```

gzip

gzip 圧縮が有効であるかどうか (文字列)。指定できる値は 'on' および 'off' です。デフォルト値は 'on' です。

⚠ Warning

圧縮によってセキュリティリスクが生じる可能性があります。圧縮を完全に無効にするには、この属性を次のように設定します。

```
node[:nginx][:gzip] = 'off'
```

```
node[:nginx][:gzip]
```

gzip_comp_level

圧縮レベルは、1~9 の値域を定めることができ、1 は最小圧縮 (文字列) に対応します。デフォルト値は '2' です。

```
node[:nginx][:gzip_comp_level]
```

gzip_disable

指定したユーザーエージェントに対して gzip 圧縮を無効にします (文字列)。値は正規表現で、デフォルト値は 'MSIE [1-6].(?:.*SV1)' です。

```
node[:nginx][:gzip_disable]
```

gzip_http_version

指定されたHTTPバージョン (文字列) の gzip 圧縮を有効にします。デフォルト値は '1.0' です。

```
node[:nginx][:gzip_http_version]
```

gzip_proxied

プロキシリクエストへのレスポンスを圧縮するかどうか、およびその方法。次のいずれかの値になります (文字列)。

- 'off': プロキシされたリクエストを圧縮しない
- 'expired': 有効期限切れヘッダーでキャッシングが妨げられる場合は圧縮する

- 'no-cache': キャッシュ制御ヘッダーが「no-cache」に設定されている場合は圧縮する
- 'no-store': キャッシュ制御ヘッダーが「no-store」に設定されている場合は圧縮する
- 'private': のキャッシュ制御ヘッダーが「private」に設定されている場合は圧縮する
- 'no_last_modified': [Last-Modified] が設定されていない場合は圧縮する
- 'no_etag': リクエストに ETagヘッダーがない場合は圧縮する
- 'auth': リクエストに認可ヘッダーが含まれる場合は圧縮する
- 'any': すべてのプロキシされたリクエストを圧縮する

デフォルト値は 'any' です。

```
node[:nginx][:gzip_proxied]
```

gzip_static

gzip 静的モジュールが有効かどうか (文字列)。指定できる値は 'on' および 'off' です。デフォルト値は 'on' です。

```
node[:nginx][:gzip_static]
```

gzip_types

圧縮するMIMEタイプのリスト (文字列のリスト)。デフォルト値は ['text/plain', 'text/html', 'text/css', 'application/x-javascript', 'text/xml', 'application/xml', 'application/xml+rss', 'text/javascript'] です。

```
node[:nginx][:gzip_types]
```

gzip_vary

Vary:Accept-Encoding レスponseヘッダーを有効にするかどうか (文字列)。指定できる値は 'on' および 'off' です。デフォルト値は 'on' です。

```
node[:nginx][:gzip_vary]
```

keepalive

キープアライブ接続を有効にするかどうか (文字列)。指定できる値は 'on' および 'off' です。デフォルト値は 'on' です。

```
node[:nginx][:keepalive]
```

keepalive_timeout

キープアライブ接続を開いたままにする最大時間 (秒) (数値)。デフォルト値は 65 です。

```
node[:nginx][:keepalive_timeout]
```

log_dir

ログファイルの場所 (文字列)。デフォルト値は '/var/log/nginx' です。

```
node[:nginx][:log_dir]
```

ユーザー

ユーザー (文字列)。デフォルト値は次のとおりです。

- Amazon Linux および RHEL: 'www-data'
- Ubuntu: 'nginx'

```
node[:nginx][:user]
```

server_names_hash_bucket_size

32、64、または 128 に設定できるサーバー名のハッシュテーブルのバケットサイズ (数値)。デフォルト値は 64 です。

```
node[:nginx][:server_names_hash_bucket_size]
```

worker_processes

ワーカースレッドの数 (数値)。デフォルト値は 10 です。

```
node[:nginx][:worker_processes]
```

worker_connections

ワーカー接続の最大数 (数値)。デフォルト値は 1024 です。クライアントの最大数は `worker_processes * worker_connections` に設定されます。

```
node[:nginx][:worker_connections]
```

opsworks_berkshelf 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

これらの属性は Linux スタックにのみ使用できます。

[opsworks_berkshelf 属性](#) は Berkshelf 設定を指定します。詳細については「[Berkshelf](#)」を参照してください。組み込み属性を上書きしてカスタム値を指定する方法の詳細については、「[属性の上書き](#)」を参照してください。

デバッグ

Chef ログに Berkshelf デバッグ情報を含めるかどうかを指定します (ブール値)。デフォルト値は `false` です。

```
node['opsworks_berkshelf']['debug']
```

opsworks_java 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

これらの属性は Linux スタックにのみ使用できます。

[opsworks_java](#) 属性は [Tomcat](#) サーバー設定を指定します。詳細については、「[Apache Tomcat 設定リファレンス](#)」を参照してください。組み込み属性を上書きしてカスタム値を指定する方法の詳細については、「[属性の上書き](#)」を参照してください。

datasources	java_app_server_version	java_shared_lib_dir
jvm_pkg 属性	custom_pkg_location_url_debian	java_home_basedir
custom_pkg_location_url_rhel	use_custom_pkg_location	jvm_options
jvm_version	tomcat 属性	

datasources

JNDI リソース名を定義する属性のセット (文字列)。この属性を使用する方法の詳細については、「[バックエンドデータベースを使用したJSPアプリケーションのデプロイ](#)」を参照してください。デフォルト値は空のハッシュで、アプリケーションの短縮名とJNDI名前間のカスタムマッピングで埋めることができます。詳細については、「[バックエンドデータベースを使用したJSPアプリケーションのデプロイ](#)」を参照してください。

```
node['opsworks_java']['datasources']
```

java_app_server_version

Java アプリケーションサーバーバージョン (数値)。デフォルト値は 7 です。この属性を上書きしてバージョン 6 を指定することができます。デフォルト以外の をインストールするとJDK、この属性は無視されます。

```
node['opsworks_java']['java_app_server_version']
```

java_shared_lib_dir

Java 共有ライブラリのディレクトリ (文字列)。デフォルト値は `/usr/share/java` です。

```
node['opsworks_java']['java_shared_lib_dir']
```

jvm_pkg 属性

デフォルト以外の をインストールするためにオーバーライドできる属性のセットJDK。

use_custom_pkg_location

OpenJDK (ブール値) JDKの代わりにカスタム をインストールするかどうか。デフォルト値は `false` です。

```
node['opsworks_java']['jvm_pkg']['use_custom_pkg_location']
```

custom_pkg_location_url_debian

Ubuntu インスタンスにインストールされるJDKパッケージの場所 (文字列)。デフォルト値は `'http://aws.amazon.com/'` です。これは単なる初期値であり、特別な意味はありません。デフォルト以外の をインストールする場合はJDK、この属性を上書きし、適切な に設定する必要がありますURL。

```
node['opsworks_java']['jvm_pkg']['custom_pkg_location_url_debian']
```

custom_pkg_location_url_rhel

Amazon Linux および RHELインスタンスにインストールされるJDKパッケージの場所 (文字列)。デフォルト値は `'http://aws.amazon.com/'` です。これは単なる初期値であり、特別な意味はありません。デフォルト以外の をインストールする場合はJDK、この属性を上書きして適切な に設定する必要がありますURL。

```
node['opsworks_java']['jvm_pkg']['custom_pkg_location_url_rhel']
```

java_home_basedir

JDK パッケージが抽出されるディレクトリ (文字列)。デフォルト値は `/usr/local` です。RPM パッケージにこの設定を指定する必要はありません。完全なディレクトリ構造が含まれています。

```
node['opsworks_java']['jvm_pkg']['java_home_basedir']
```

jvm_options

JVM コマンドラインオプション。ヒープサイズ (文字列) などの設定を指定できます。オプションの一般的な設定は `-Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC` です。デフォルト値にオプションはありません。

```
node['opsworks_java']['jvm_options']
```

jvm_version

OpenJDK バージョン (数値)。デフォルト値は 7 です。この属性を上書きして Open JDK バージョン 6 を指定できます。デフォルト以外の をインストールすると JDK、この属性は無視されます。

```
node['opsworks_java']['jvm_version']
```

tomcat 属性

デフォルトの Tomcat 設定をインストールするために上書きできる属性のセット。

ajp_port	apache_tomcat_bind_mod	apache_tomcat_bind_path
auto_deploy	connection_timeout	mysql_connector_jar
port	secure_port	shutdown_port
threadpool_max_threads	threadpool_min_spare_thread	unpack_wars
	s	
uri_encoding	use_ssl_connector	use_threadpool

userdatabase_pathname

ajp_port

AJP ポート (数値)。デフォルト値は 8009 です。

```
node['opsworks_java']['tomcat']['ajp_port']
```

apache_tomcat_bind_mod

プロキシモジュール (文字列)。デフォルト値は proxy_http です。この属性を上書きして、AJPプロキシモジュール を指定できます proxy_ajp。

```
node['opsworks_java']['tomcat']['apache_tomcat_bind_mod']
```

apache_tomcat_bind_path

Apache Tomcat のバインドパス (文字列)。デフォルト値は / です。この属性を上書きしないでください。バインドパスを変更すると、アプリケーションの動作が停止する場合があります。

```
node['opsworks_java']['tomcat']['apache_tomcat_bind_path']
```

auto_deploy

自動デプロイするかどうか (ブール)。デフォルト値は true です。

```
node['opsworks_java']['tomcat']['auto_deploy']
```

connection_timeout

接続タイムアウト (ミリ秒) (数値)。デフォルト値は 2000020 秒です。

```
node['opsworks_java']['tomcat']['connection_timeout']
```

mysql_connector_jar

MySQLコネクタライブラリの JAR ファイル (文字列)。デフォルト値は mysql-connector-java.jar です。

```
node['opsworaks_java']['tomcat']['mysql_connector_jar']
```

port

標準ポート (数値)。デフォルト値は 8080 です。

```
node['opsworaks_java']['tomcat']['port']
```

secure_port

安全なポート (数値)。デフォルト値は 8443 です。

```
node['opsworaks_java']['tomcat']['secure_port']
```

shutdown_port

シャットダウンポート (数値)。デフォルト値は 8005 です。

```
node['opsworaks_java']['tomcat']['shutdown_port']
```

threadpool_max_threads

スレッドプールのスレッドの最大数 (数値)。デフォルト値は 150 です。

```
node['opsworaks_java']['tomcat']['threadpool_max_threads']
```

threadpool_min_spare_threads

スレッドプールの予備のスレッドの最小数 (数値)。デフォルト値は 4 です。

```
node['opsworaks_java']['tomcat']['threadpool_min_spare_threads']
```

unpack_wars

WAR ファイルを解凍するかどうか (ブール値)。デフォルト値は true です。

```
node['opsworaks_java']['tomcat']['unpack_wars']
```

uri_encoding

URI エンコード (文字列)。デフォルト値は UTF-8 です。

```
node['opsworks_java']['tomcat']['uri_encoding']
```

use_ssl_connector

SSL コネクタを使用するかどうか (ブール値)。デフォルト値は false です。

```
node['opsworks_java']['tomcat']['use_ssl_connector']
```

use_threadpool

スレッドプールを使用するかどうか (ブール値)。デフォルト値は false です。

```
node['opsworks_java']['tomcat']['use_threadpool']
```

userdatabase_pathname

ユーザーデータベースのパス名 (文字列)。デフォルト値は conf/tomcat-users.xml です。

```
node['opsworks_java']['tomcat']['userdatabase_pathname']
```

passenger_apache2 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

これらの属性は Linux スタックにのみ使用できます。

[passenger_apache2](#) 属性は [Phusion Passenger](#) 設定を指定します。詳細については、「[Phusion Passenger ユーザーガイド、Apache バージョン](#)」を参照してください。組み込み属性を上書きしてカスタム値を指定する方法の詳細については、「[属性の上書き](#)」を参照してください。

friendly_error_pages	gem_bin	gems_path
high_performance_mode	root_path	max_instances_per_app
max_pool_size	max_requests	module_path
pool_idle_time	rails_app_spawner_idle_time	rails_framework_spawner_idle_time
rails_spawn_method	ruby_bin	ruby_wrapper_bin
stat_throttle_rate	version	

friendly_error_pages

アプリケーションが起動に失敗した場合に、わかりやすいエラーページを表示するかどうかを指定します (文字列)。この属性は、'on' または 'off' に設定できます。デフォルト値は 'off' です。

```
node[:passenger][:friendly_error_pages]
```

gem_bin

Gem バイナリの場所 (文字列)。デフォルト値は '/usr/local/bin/gem' です。

```
node[:passenger][:gem_bin]
```

gems_path

gems パス (文字列)。デフォルト値は、Ruby のバージョンによって異なります。例えば:

- Ruby バージョン 1.8: '/usr/local/lib/ruby/gems/1.8/gems'
- Ruby バージョン 1.9: '/usr/local/lib/ruby/gems/1.9.1/gems'

```
node[:passenger][:gems_path]
```

high_performance_mode

Passenger の高パフォーマンスモードを使用するかどうか (文字列)。指定できる値は 'on' および 'off' です。デフォルト値は 'off' です。

```
node[:passenger][:high_performance_mode ]
```

root_path

Passenger ルートディレクトリ (文字列)。デフォルト値は、Ruby および Passenger のバージョンによって異なります。Chef 構文では、この値は "`#{node[:passenger][:gems_path]}/passenger-#{passenger[:version]}`" です。

```
node[:passenger][:root_path]
```

max_instances_per_app

アプリケーションごとのアプリケーションプロセスの最大数 (数値)。デフォルト値は 0 です。詳細については、「[PassengerMaxInstancesPerApp](#)」を参照してください。

```
node[:passenger][:max_instances_per_app]
```

max_pool_size

アプリケーションプロセッサの最大数 (数値)。デフォルト値は 8 です。詳細については、「[PassengerMaxPoolSize](#)」を参照してください。

```
node[:passenger][:max_pool_size]
```

max_requests

リクエストの最大数 (数値)。デフォルト値は 0 です。

```
node[:passenger][:max_requests]
```

module_path

モジュールのパス (文字列)。デフォルト値は次のとおりです。

- Amazon Linux および RHEL: "`#{node['apache']['libexecdir']}/mod_passenger.so`"

- Ubuntu: "`#{passenger[:root_path]}/ext/apache2/mod_passenger.so`"

```
node[:passenger][:module_path]
```

pool_idle_time

アプリケーションプロセスをアイドル状態にすることができる最大時間 (秒) (数値)。デフォルト値は 14400 (4 時間) です。詳細については、「[PassengerPoolIdleTime](#)」を参照してください。

```
node[:passenger][:pool_idle_time]
```

rails_app_spawner_idle_time

Rails アプリケーションのスポナーの最大アイドル時間 (数値)。この属性をゼロに設定した場合、アプリケーションのスポナーはタイムアウトになりません。デフォルト値は 0 です。詳細については、「[Spawning Methods Explained](#)」を参照してください。

```
node[:passenger][:rails_app_spawner_idle_time]
```

rails_framework_spawner_idle_time

Rails フレームワークのスポナーの最大アイドル時間 (数値)。この属性がゼロに設定されている場合、フレームワークのスポナーはタイムアウトになりません。デフォルト値は 0 です。詳細については、「[Spawning Methods Explained](#)」を参照してください。

```
node[:passenger][:rails_framework_spawner_idle_time]
```

rails_spawn_method

Rails のスポンの手法 (文字列)。デフォルト値は 'smart-lv2' です。詳細については、「[Spawning Methods Explained](#)」を参照してください。

```
node[:passenger][:rails_spawn_method]
```

ruby_bin

Ruby バイナリの場所 (文字列)。デフォルト値は '/usr/local/bin/ruby' です。

```
node[:passenger][:ruby_bin]
```

ruby_wrapper_bin

Ruby ラッパースクリプトの場所 (文字列)。デフォルト値は '/usr/local/bin/ruby_gc_wrapper.sh' です。

```
node[:passenger][:ruby_wrapper_bin]
```

stat_throttle_rate

Passenger がファイルシステムチェックを実行するレート (数値)。デフォルト値は 5 です。これは、チェックが最大でも 5 秒ごとに実行されることを意味します。詳細については、「[PassengerStatThrottleRate](#)」を参照してください。

```
node[:passenger][:stat_throttle_rate]
```

version

バージョン (文字列)。デフォルト値は '3.0.9' です。

```
node[:passenger][:version]
```

ruby 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

これらの属性は Linux スタックにのみ使用できます。

[ruby 属性](#) は、アプリケーションで使用する Ruby バージョンを指定します。Ruby 2.1 でセマンティクスのバージョンングが導入されたことにより、属性の使用方法が変わったことに注意してください

い。バージョンの指定方法と例については、「[Ruby のバージョン](#)」を参照してください。AWS OpsWorks スタックが Ruby バージョンを決定する方法の詳細については、組み込み属性ファイル [ruby.rb](#) を参照してください。組み込み属性を上書きしてカスタム値を指定する方法の詳細については、「[属性の上書き](#)」を参照してください。

full_version

完全なバージョン番号 (文字列)。この属性を上書きしないでください。代わりに [\[:opsworks\]](#) [\[:ruby_version\]](#) と適切なパッチバージョン属性を使用してバージョンを指定してください。

```
[ :ruby ][ :full_version ]
```

major_version

メジャーバージョン番号 (文字列)。この属性を上書きしないでください。代わりに [\[:opsworks\]](#) [\[:ruby_version\]](#) を使用してメジャーバージョンを指定してください。

```
[ :ruby ][ :major_version ]
```

minor_version

マイナーバージョン番号 (文字列)。この属性を上書きしないでください。代わりに [\[:opsworks\]](#) [\[:ruby_version\]](#) を使用してマイナーバージョンを指定してください。

```
[ :ruby ][ :minor_version ]
```

patch

パッチレベル (文字列)。この属性は Ruby バージョン 2.0.0 以前で有効です。それより後の Ruby バージョンでは、`patch_version` 属性を使用します。

```
[ :ruby ][ :patch ]
```

パッチ番号の前に `p` を置く必要があります。たとえば、次のカスタム を使用してパッチレベル 484 JSON を指定します。

```
{
  "ruby": {"patch": "p484"}
}
```


patch_version

パッチの数 (文字列)。この属性は Ruby バージョン 2.1 以降で有効です。それより前の Ruby バージョンでは、patch 属性を使用します。

```
[ :ruby ][ :patch_version ]
```

pkgrelease

パッケージのリリース番号 (文字列)。

```
[ :ruby ][ :pkgrelease ]
```

unicorn 属性

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

これらの属性は Linux スタックにのみ使用できます。

[unicorn 属性](#) は [ユニコーン](#) 設定を指定します。詳細については、「[Unicorn::Configurator](#)」を参照してください。組み込み属性を上書きしてカスタム値を指定する方法の詳細については、「[属性の上書き](#)」を参照してください。

accept_filter	backlog	delay
tcp_nodelay	tcp_nopush	preload_app
timeout	tries	version

[worker_processes](#)**accept_filter**

承認フィルタの 'httpready' または 'dataready' (文字列)。デフォルト値は 'httpready' です。

```
node[:unicorn][:accept_filter]
```

backlog

キューが保持できるリクエストの最大数 (数値)。デフォルト値は 1024 です。

```
node[:unicorn][:backlog]
```

delay

ソケットのバインドの再試行を待機する時間 (秒) (数値)。デフォルト値は 0.5 です。

```
node[:unicorn][:delay]
```

preload_app

ワーカプロセスを分岐する前に、アプリケーションを事前ロードするかどうか (ブール値)。デフォルト値は true です。

```
node[:unicorn][:preload_app]
```

tcp_nodelay

TCP ソケットの Nagle のアルゴリズムを無効にするかどうか (ブール値)。デフォルト値は true です。

```
node[:unicorn][:tcp_nodelay]
```

tcp_nopush

TCP_CORK (ブール値) を有効にするかどうか。デフォルト値は false です。

```
node[:unicorn][:tcp_nopush]
```

timeout

ワーカーが各リクエストで使用できる最大時間 (秒) (数値)。タイムアウト値を超えたワーカーは終了します。デフォルト値は 60 です。

```
node[:unicorn][:timeout]
```

tries

ソケットへのバインドの再試行の最大数 (数値)。デフォルト値は 5 です。

```
node[:unicorn][:tries]
```

version

ユニコーンバージョン (文字列)。デフォルト値は '4.7.0' です。

```
node[:unicorn][:version]
```

worker_processes

ワーカープロセスの数 (数値)。デフォルト値は max_pool_size (存在する場合)、それ以外の場合は 4 です。

```
node[:unicorn][:worker_processes]
```

Linux 用 Chef 11.10 以前のバージョンのトラブルシューティング

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS プレミアムサポート](#) を通じて AWS サポート チームにお問い合わせください。

Note

トラブルシューティングの詳細については、「[デバッグとトラブルシューティングのガイド](#)」を参照してください。

Linux 用 Chef 11.10 以前のバージョンの Chef ログ

AWS OpsWorks スタックは、各インスタンスの Chef ログを `/var/lib/aws/opsworks/chef` ディレクトリに保存します。このディレクトリにアクセスするには、`sudo` 権限が必要です。各実行のログは、`YYYY-MM-DD-HH-MM-SS-NN.log` という名前のファイルにあります。

詳細については、次を参照してください。

- [コンソールを使用した Chef ログの表示](#)
- [CLI または API を使用した Chef ログの表示](#)
- [Chef ログの解釈](#)
- [Chef ログの一般的なエラー](#)

他の AWS サービスでの AWS OpsWorks スタックの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックスタックで実行されているアプリケーションサーバーに、スタックと直接統合されていないさまざまな AWS サービスを使用させることができます AWS OpsWorks。例えば、アプリケーションサーバがバックエンドのデータベースとして Amazon RDS を使用することができます。このようなサービスにアクセスするための一般的な方法は、次のとおりです。

1. AWS コンソール、API、または CLI を使用して AWS サービスの作成と設定を行い、アプリケーションからサービスにアクセスするために必要となるホスト名やポートなどの設定データを記録します。
2. 1 つ以上のカスタムレシピを作成して、アプリケーションからサービスにアクセスできるように設定します。

レシピは、「[スタック設定とデプロイメント JSON](#)」で示された属性 (レシピを実行する前にカスタム JSON で設定) から設定データを取得します。

3. アプリケーションサーバーレイヤーの Deploy ライフサイクルイベントにカスタムレシピを割り当てます。
4. 設定データ属性に適切な値を割り当てるカスタム JSON オブジェクトを作成し、これをスタック設定とデプロイメント JSON に追加します。
5. アプリケーションをスタックにデプロイします。

デプロイメントによって実行されるカスタムレシピは、カスタム JSON で定義した設定データの値を使用して、アプリケーションからサービスにアクセスできるように設定します。

このセクションでは、AWS OpsWorks スタックアプリケーションサーバーにさまざまな AWS サービスへのアクセスを許可する方法について説明します。ここでは、Chef クックブックや、レシピでスタックと設定 JSON 属性を使用してアプリケーションを設定する方法 (通常は設定ファイルを作成) について、ユーザーがすでに知識を持っていることを前提としています。知識を持っていない場合は、最初に「[クックブックとレシピ](#)」と「[AWS OpsWorks スタックのカスタマイズ](#)」をお読みください。

トピック

- [バックエンドデータストアの使用](#)
- [ElastiCache Redis をインメモリキーバリューストアとして使用する](#)
- [Amazon S3 バケットの使用](#)
- [AWS OpsWorks スタック AWS CodePipeline での使用](#)

バックエンドデータストアの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

アプリケーションサーバースタックには通常、バックエンドデータストアを提供するデータベースサーバーが含まれています。AWS OpsWorks スタックは、MySQL [レイヤーを介した MySQL](#) サーバーと [Amazon Relational Database Service \(Amazon RDS\) レイヤー](#) を介した複数のタイプのデータベースサーバーの統合サポートを提供します。ただし、アプリケーションサーバーが Amazon DynamoDB や MongoDB などの他のデータベースサーバーを使用できるようにスタックを簡単にカスタマイズできます。このトピックでは、アプリケーションサーバーを AWS データベースサーバーに接続するための基本的な手順について説明します。「[Chef 11 Linux スタックの使用開始](#)」のスタックおよびアプリケーションを使用して、PHP のアプリケーションサーバーを RDS データベースに手動で接続する方法を示します。この例は Linux スタックに基づいていますが、基本的な原則は Windows スタックにも適用されます。MongoDB データベースサーバーをスタックに組み込む方法の例については、「[で MongoDB をデプロイする OpsWorks](#)」を参照してください。

Note

このトピックでは、ちょうどいい例として Amazon RDS を使用します。ただし、スタックで Amazon RDS データベースを使用する場合、Amazon RDS レイヤーを使用する方がはるかに簡単です。

トピック

- [データベース接続をセットアップする方法](#)
- [アプリケーションサーバーのインスタンスを Amazon RDS に接続する方法](#)

データベース接続をセットアップする方法

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

カスタムレシピを使用して、アプリケーションサーバーとバックエンドデータベース間の接続をセットアップします。レシピによりアプリケーションサーバーが必要に応じて設定されます。通常、設定ファイルを作成して設定されます。recipe は、[スタック設定の一連の属性と、スタックがすべてのインスタンスにインストールするデプロイ属性](#)から、ホストやデータベース名などの接続データを取得します。AWS OpsWorks

例えば、 のステップ 2 [Chef 11 Linux スタックの使用開始](#)は、PHP App Server と MySQL の 2 つのレイヤー MyStack を持つ という名前のスタックに基づいており、それぞれに 1 つのインスタンスがあります。MySQL SimplePHPApp という名前のアプリケーションを PHP App Server インスタンスにデプロイし、バックエンドデータストアとして MySQL インスタンスでデータベースを使用します。アプリケーションをデプロイすると、AWS OpsWorks スタックによって、データベース接続情報を含むスタック設定およびデプロイ属性がインストールされます。以下の例は、JSON として表されたデータベース接続属性を示しています。

```
{
  ...
  "deploy": {
    "simplephpapp": {
      ...
      "database": {
        "reconnect": true,
        "password": null,
        "username": "root",
        "host": null,
        "database": "simplephpapp"
      }
      ...
    },
    ...
  }
}
```

```
    }  
  }  
}
```

属性値は AWS OpsWorks スタックによって提供され、生成されるか、ユーザーが提供した情報に基づいています。

SimplePHPApp がデータストアにアクセスするのを許可するには、appsetup.rb という名前のカスタムレシピを レイヤーのデプロイ [\[lifecycle event\]](#) (ライフサイクルイベント) に割り当てることで、PHP アプリケーションサーバーと MySQL データベース間の接続を設定する必要があります。SimplePHPApp をデプロイすると、AWS OpsWorks スタックは を実行し appsetup.rb、次の抜粋に示すように、接続db-connect.phpを設定する という名前の設定ファイルを作成します。

```
node[:deploy].each do |app_name, deploy|  
  ...  
  template "#{deploy[:deploy_to]}/current/db-connect.php" do  
    source "db-connect.php.erb"  
    mode 0660  
    group deploy[:group]  
  
    if platform?("ubuntu")  
      owner "www-data"  
    elsif platform?("amazon")  
      owner "apache"  
    end  
  
    variables(  
      :host => (deploy[:database][:host] rescue nil),  
      :user => (deploy[:database][:username] rescue nil),  
      :password => (deploy[:database][:password] rescue nil),  
      :db => (deploy[:database][:database] rescue nil),  
      :table => (node[:phpapp][:dbtable] rescue nil)  
    )  
    ...  
  end  
end
```

接続を特徴づける変数 host、user などには、[deploy JSON's](#) (JSON をデプロイする) [:deploy] [:app_name][:database] 属性から対応する値が設定されます。わかりやすいように、この例で

は、`urler` という名前のテーブルが作成されており、そのテーブル名はクックブックの属性ファイルでは `[:phpapp][:dbtable]` で表されることを前提としています。

このレシピにより、PHP アプリケーションサーバーは MySQL レイヤーのメンバーだけでなく任意の MySQL データベースサーバーに実際に接続されます。別の MySQL サーバーを使用するには、サーバーに適した値に `[:database]` 属性を設定するだけで済みます。カスタム [JSON](#) を使用して属性と値をスタック設定とデプロイ属性に AWS OpsWorks 組み込み、それら `appsetup.rb` を使用して接続を設定するテンプレートを作成します。スタック設定およびデプロイメント JSON の上書きの詳細については、「[属性の上書き](#)」を参照してください。

アプリケーションサーバーのインスタンスを Amazon RDS に接続する方法

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、MyStack から をカスタマイズ [Chef 11 Linux スタックの使用開始](#) して PHP アプリケーションサーバーを RDS インスタンスに接続する方法について説明します。

トピック

- [Amazon RDS MySQL データベースを作成する](#)
- [RDS データベースに接続するためにスタックをカスタマイズする](#)

Amazon RDS MySQL データベースを作成する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Amazon RDS コンソールの起動 DB インスタンスウィザードを使用して見本としての RDS データベースを作成する準備が整いました。次の手順は、必要な詳細についての要約です。データベースを作成する方法の詳細については、『[Amazon RDS の入門ガイド](#)』を参照してください。

Amazon RDS データベースを作成するには

1. RDS データベースを初めて作成する場合は、[Get Started Now] をクリックします。それ以外の場合は、ナビゲーションペインで [RDS Dashboard] をクリックして、[Launch a DB Instance] をクリックします。
2. DB インスタンスとして [MySQL Community Edition] を選択します。
3. [本番稼働用にこのデータベースを使用する予定はありますか?] では、[このインスタンスは本番環境以外...] を選択します。この例ではこれで十分です。本稼働環境で使用する場合は、[Yes, use Multi-AZ Deployment...] を選択する必要がある場合もあります。[Next Step] をクリックします。
4. [Specify DB Details] ページで、次の設定を指定します。
 - DB Instance Class: db.t2.micro
 - Multi-AZ Deployment: No
 - ストレージ割り当て: 5 GB
 - DB インスタンス識別子: **rdsexample**
 - マスターユーザーの名前: **opsworkuser**
 - [Master Password]: 適切なパスワードを指定し、後で使用できるように記録しておきます。

他のオプションのデフォルト設定を受け入れ、[Next Step] をクリックします。

5. [Configure Advanced Settings] ページで、以下の設定を指定します。
 - [Network & Security] で、[VPC Security Group(s)] の [phpsecgroup (VPC)] を選択します
 - [データベースの設定] セクションで、[データベースの名前] に「**rdsexampledb**」と入力します
 - [Backup] セクションで、このウォークスルーでは [Backup Retention Period] を [0] に設定します。

他のオプションのデフォルト設定を受け入れ、[Launch DB Instance] をクリックします。

6. [View Your DB Instances] を選択して DB インスタンスのリストを表示します。

7. リストの rdsexample インスタンスを選択し、矢印をクリックしてインスタンスのエンドポイントおよびその他の詳細を表示します。後で使用できるように、エンドポイントを記録しておきます。rdsexample.c6c8mntzhgv0.us-west-2.rds.amazonaws.com:3306 のようになります。DNS 名を記録します。ポート番号は必要ではありません。
8. MySQL Workbench などのツールを使用して urler データベースに rdsexampledb という名前のテーブルを作成するには、次の SQL コマンドを使用します。

```
CREATE TABLE urler(id INT UNSIGNED NOT NULL AUTO_INCREMENT,author VARCHAR(63) NOT NULL,message TEXT,PRIMARY KEY (id))
```

RDS データベースに接続するためにスタックをカスタマイズする

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

PHP アプリケーションサーバーのバックエンドデータベースとして使用する [RDS インスタンスを作成](#)したら、からカスタマイズ MyStack できます [Chef 11 Linux スタックの使用開始](#)。

PHP アプリケーションサーバーを RDS データベースに接続するには

1. 「」の説明に従って、AWS OpsWorks スタックコンソールを開き、1 つのインスタンスを含む PHP アプリケーションサーバーレイヤーを使用してスタックを作成し、SimplePHPApp をデプロイします [Chef 11 Linux スタックの使用開始](#)。このスタックは SimplePHPApp の version1 を使用します。これは、データベース接続を使用しません。
2. [スタック設定を更新](#)し、appsetup.rb レシピ、関連するテンプレートや属性ファイルを含むカスタムクックブックを使用します。
 1. [Use custom Chef cookbooks] を [Yes] に設定します。
 2. [Repository type] を [Git] に、[Repository URL] を [git://github.com/amazonwebservices/opsworks-example-cookbooks.git] に設定します。

3. 次をスタックの [Custom Chef JSON] ボックスに追加し、[:database] が設定ファイルを作成するのに使用する appsetup.rb 属性にRDS 接続データを割り当てます。

```
{
  "deploy": {
    "simplephpapp": {
      "database": {
        "username": "opsworksuser",
        "password": "your_password",
        "database": "rdsexampled",
        "host": "rds_endpoint",
        "adapter": "mysql"
      }
    }
  }
}
```

次の属性値を使用します。

- username: RDS インスタンスの作成時に指定したマスターユーザー名

この例では opsworksuser を使用します。

- [password]: RDS のインスタンスの作成時に指定したマスターパスワード

指定したパスワードを入力します。

- database: RDS インスタンスの作成時に作成したデータベース

この例では rdsexampled を使用します。

- host: RDS インスタンスのエンドポイント。前のセクションでインスタンスを作成した際に、RDS コンソールから取得しています。ポート番号は必要ありません。

- adapter: アダプタ

この例の RDS インスタンスは MySQL を使用するため、[adapter] は [mysql] に設定されます。その他の属性とは異なり、[adapter] は [appsetup.rb] で使用されません。代わりに、別の構成ファイルを作成するために PHP App Server レイヤーの組み込み構成レシピにより使用されます。

4. [SimplePHPApp 設定を編集](#)して、次のように、バックエンドデータベースを使用する SimplePHPApp のバージョンを指定します。

- Document root: このオプションを web に設定します。
- Branch/Revision: このオプションを version2 に設定します。

残りのオプションは変更しません。

5. [\[Edit the PHP App Server layer\]](#) (PHP アプリケーションサーバーレイヤーを編集)し、`phpapp::appsetup` をレイヤーのデプロイレシピに追加することでデータベース接続を設定します。
6. [SimplePHPApp の新しいバージョンをデプロイします。](#)
7. SimplePHPApp がデプロイされる時、[Instances] ページに進み、`php-app1` インスタンスのパブリック IP アドレスをクリックして、アプリケーションを実行します。ブラウザに次のページが表示されるので、そこでテキストを入力し、それをデータベースに保存します。



Note

スタックに MySQL レイヤーがある場合、AWS OpsWorks Stacks は対応する接続データを `[:database]` 属性に自動的に割り当てます。ただし、スタックに異なる `[:database]` 値を定義するカスタム JSON を割り当てると、デフォルト値は上書きされます。

`[:deploy]` 属性は、すべてのインスタンスにインストールされているため、`[:database]` 属性に依存するレシピは MySQL レイヤーのデータではなくカスタム接続データを使用しま

す。特定のアプリケーションサーバーのレイヤーがカスタム接続データを使用するようにする場合、カスタム JSON をレイヤーのデプロイイベントに割り当て、そのレイヤーにそのデプロイメントを制限します。デプロイメント属性の使用の詳細については、「[アプリケーションのデプロイ](#)」を参照してください。AWS OpsWorks スタックの組み込み属性のオーバーライドについては、「[属性の上書き](#)」を参照してください。

ElastiCache Redis をインメモリキーバリューストアとして使用する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

このトピックは Linux スタックに基づいていますが、Windows スタックは Amazon ElastiCache () を使用することもできますElastiCache。Windows インスタンス ElastiCache でを使用する方法の例については、[ElastiCache ASP.NET セッションストアとして](#) を参照してください。

多くの場合、キャッシュサーバーを使用して文字列などの小さなデータ項目に対してインメモリ key-value ストアを提供することで、アプリケーションサーバーのパフォーマンスを向上させることができます。Amazon ElastiCache は、[Memcached](#) または [Redis キャッシュエンジンのいずれかを使用して、アプリケーションサーバーのキャッシュサポートを簡単に提供できる](#) AWS のサービスです。AWS OpsWorks スタックは、[Memcached の組み込みサポートを提供します](#)。ただし、Redis が要件により適している場合は、アプリケーションサーバーが Redis ElastiCache を使用するようにスタックをカスタマイズできます。

このトピックでは、Rails ElastiCache アプリケーションサーバーを例として使用して、Linux スタックの Redis キャッシュサポートを提供する基本的なプロセスについて説明します。ここでは、

適切な Ruby on Rails アプリケーションがすでにあることを前提としています。の詳細については [ElastiCache](#)、[「Amazon とは ElastiCache」](#) を参照してください。

トピック

- [ステップ 1: ElastiCache Redis クラスターを作成する](#)
- [ステップ 2: Rails スタックをセットアップする](#)
- [ステップ 3: カスタムクックブックを作成してデプロイする](#)
- [ステップ 4: レシピをイベントに割り当てる LifeCycle](#)
- [ステップ 5: スタック設定 JSON にアクセス情報を追加する](#)
- [ステップ 6: アプリをデプロイして実行する](#)

ステップ 1: ElastiCache Redis クラスターを作成する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

まず、ElastiCache コンソール、API、または ElastiCache CLI を使用して Amazon Redis クラスターを作成する必要があります。コンソールを使用してクラスターを作成する方法を以下に示します。

Redis ElastiCache クラスターを作成するには

1. [ElastiCache コンソール](#) に移動し、キャッシュクラスターの起動をクリックして、キャッシュクラスターウィザードを起動します。
2. [Cache Cluster Details] ページで次の操作を実行します。
 - [Name] でキャッシュサーバー名を指定します。
この例では OpsWorks-Redis を使用しています。
 - [Engine] を [redis] に設定します。

- [Topic for SNS Notification] を [Disable Notifications] に設定します。
- その他の設定にはデフォルト値をそのまま使用し、[Continue] をクリックします。

Launch Cache Cluster Wizard Cancel X

CACHE CLUSTER DETAILS ADDITIONAL CONFIGURATION REVIEW

To get started, provide the details for your Cache Cluster below.

Name:*

Engine:

Cache Engine Version:

Node Type:

Number of Nodes:*

Cache Port:* (e.g. 11211)

Cache Subnet Group:

Preferred Zone:

Topic for SNS Notification: **Manual ARN input**

S3 Snapshot Location:

Auto Minor Version Upgrade: Yes No

Note: "Auto Minor Version Upgrade" only applies to the Cache Engine software. Critical System Software patches (e.g. security related) may be applied irrespective of this selection.

* Required

3. [Additional Configuration] ページでデフォルト値をそのまま使用し、[Continue] をクリックします。

Launch Cache Cluster Wizard

Cancel X

CACHE CLUSTER DETAILS **ADDITIONAL CONFIGURATION** REVIEW

Security Group

A **Cache Security Group** acts like a firewall that controls network access to your Cache Clusters. Please select one or more Cache Security Groups for this Cache Cluster.

Cache Security Group(s):

Cache Parameter Group

A **Cache Parameter Group** acts as a "container" for engine configuration values that can be applied to one or more Cache Clusters. If you have created a custom Cache Parameter Group you want to use, select it from below, otherwise proceed with the **default** one we created for you.

Cache Parameter Group:

Maintenance Window

Maintenance Window allows you to specify the time range (UTC) during which any scheduled maintenance activities such as software patching or pending Cache Cluster modifications you requested would occur. Scheduled maintenance activities occur infrequently (generally once every few months) and will be announced on the AWS forum two weeks prior to being scheduled.

Maintenance Window: No Preference Select Window

< Back * Required

Continue ▶

4. [Launch Cache Cluster] をクリックしてクラスターを作成します。

Important

この例ではデフォルトのキャッシュセキュリティグループで十分ですが、本稼働環境で使用する場合は、環境に適したグループを作成する必要があります。詳細については、[「キャッシュセキュリティグループの管理」](#)を参照してください。

5. クラスターが起動したら、名前をクリックして詳細ページを開き、[Nodes] タブをクリックします。後で使用できるように、クラスターの [Port] と [Endpoint] の値を書き留めておきます。

	Node Id	Node Status	Created on	Port	Endpoint	Parameter Group Status
<input type="checkbox"/>	0001	available	Thu Sep 05 16:32:45 GMT-700 2013	6379	opsworks-redis.b47jtf.0001.use1.cache.amazonaws.com	in-sync

ステップ 2: Rails スタックをセットアップする

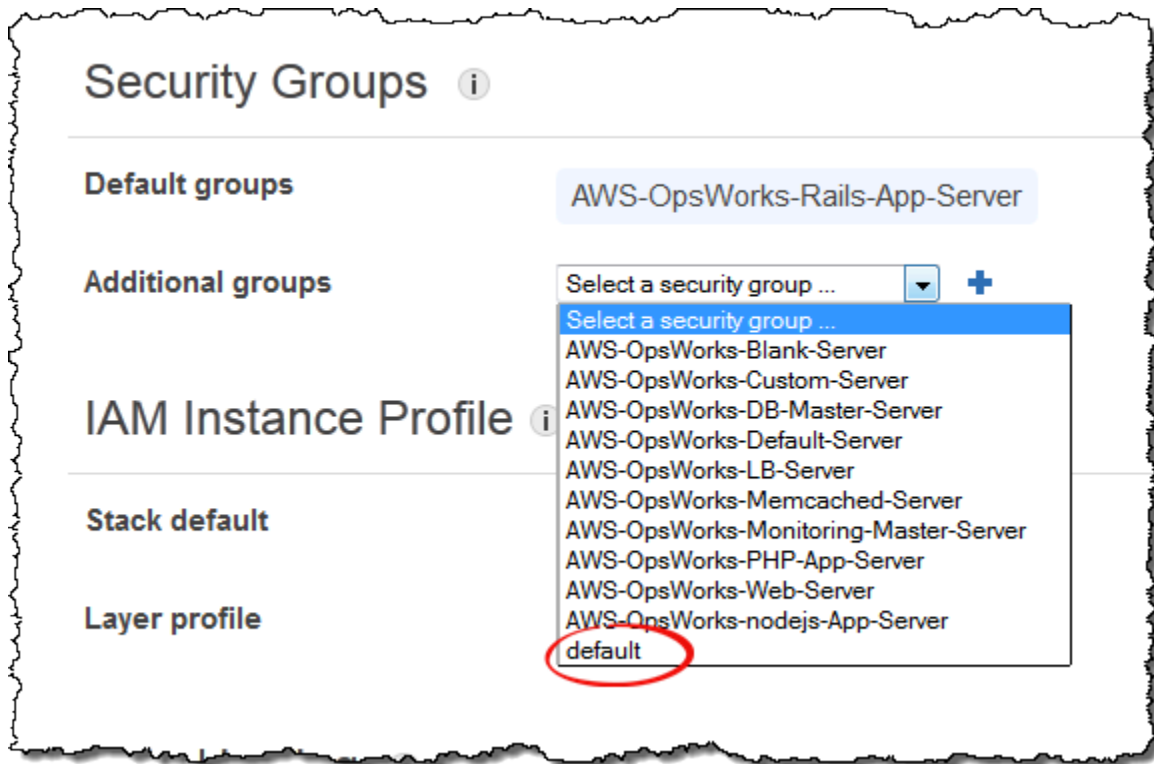
Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Rails アプリケーションサーバーレイヤーをサポートするスタックの作成に加え、Rails サーバーが Redis サーバーと正常に通信できるように、レイヤーのセキュリティグループも設定する必要があります。

スタックをセットアップするには

1. 新しいスタック **RedisStack** (を参照)を作成するには、Rails アプリケーションサーバーレイヤーを追加します。両方にデフォルト設定を使用します。詳細については、「[新しいスタックを作成する](#)」および「[OpsWorks レイヤーの作成](#)」を参照してください。
2. [Layers] (レイヤー) ページで、Rails App Server に対して、[Security] (セキュリティ) をクリックし、続いて [Edit] (編集) をクリックします。
3. セキュリティグループセクションに移動し、クラスターのセキュリティグループを追加グループに追加します ElastiCache。この例では、[default] セキュリティグループを選択し、[+] をクリックしてレイヤーに追加します。次に、[Save] をクリックして新しい設定を保存します。



4. Rails アプリケーションサーバーレイヤーにインスタンスを追加し、起動します。インスタンスを追加して起動する方法の詳細については、「[レイヤーへのインスタンスの追加](#)」を参照してください。

ステップ 3: カスタムクックブックを作成してデプロイする

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

現時点では、スタックはまだ完全には機能していません。アプリケーションが Redis サーバーにアクセスできるようにする必要があります。最も柔軟性のある方法は、アクセス情報が含まれた YAML ファイルをアプリケーションの config サブフォルダに配置することです。そうすれば、アプリケーションはこのファイルから情報を取得できるようになります。この方法を使用すると、アプリケーションを書き直して再デプロイすることなく接続情報を変更できます。この例では、ファイル

に という名前 `redis.yml` を付け、次のように ElastiCache クラスターのホスト名とポートを含める必要があります。

```
host: cache-cluster-hostname
port: cache-cluster-port
```

このファイルをサーバーに手動でコピーすることもできますが、Chef レシピを実装してファイルを生成し、AWS OpsWorks スタックですべてのサーバーでレシピを実行する方が良い方法です。Chef レシピは、AWS OpsWorks スタックがパッケージのインストールや設定ファイルの作成などのタスクをインスタンスで実行するために使用する特殊な Ruby アプリケーションです。レシピはクックブックにパッケージ化されます。クックブックには、複数のレシピと関連ファイル (設定ファイルのテンプレートなど) を含めることができます。クックブックは などのリポジトリに配置され GitHub、標準のディレクトリ構造である必要があります。カスタムクックブックリポジトリがまだない場合、これをセットアップする方法の詳細については、「[クックブックリポジトリ](#)」を参照してください。

この例では、`redis-config` という名前のクックブックを、次の内容でクックブックリポジトリに追加します。

```
my_cookbook_repository
  redis-config
    recipes
      generate.rb
    templates
      default
        redis.yml.erb
```

`recipes` フォルダには、次のように `generate.rb` からアプリケーションの設定ファイルを生成する、`redis.yml.erb` というレシピが保存されています。

```
node[:deploy].each do |app_name, deploy_config|
  # determine root folder of new app deployment
  app_root = "#{deploy_config[:deploy_to]}/current"

  # use template 'redis.yml.erb' to generate 'config/redis.yml'
  template "#{app_root}/config/redis.yml" do
    source "redis.yml.erb"
    cookbook "redis-config"
```

```
# set mode, group and owner of generated file
mode "0660"
group deploy_config[:group]
owner deploy_config[:user]

# define variable "@redis" to be used in the ERB template
variables(
  :redis => deploy_config[:redis] || {}
)

# only generate a file if there is Redis configuration
not_if do
  deploy_config[:redis].blank?
end
end
end
```

recipe は、AWS OpsWorks 各インスタンスにインストールされ、[スタックとデプロイされたアプリケーションに関する詳細情報を含む、スタックスタック設定とデプロイ JSON](#) オブジェクトのデータによって異なります。このオブジェクトの deploy ノードの構造は次のとおりです。

```
{
  ...
  "deploy": {
    "app1": {
      "application" : "short_name",
      ...
    }
    "app2": {
      ...
    }
    ...
  }
}
```

deploy ノードには、デプロイする各アプリケーションに対応し、そのアプリケーションの短縮名が付けられた一連の埋め込み JSON オブジェクトが含まれます。各アプリケーションオブジェクトには、アプリケーションの設定 (ドキュメントのルートやアプリケーションタイプなど) を定義する一連の属性が含まれます。deploy 属性のリストについては、「[deploy 属性](#)」を参照してください。レ

レシピでは、Chef の属性構文を使用して、スタック設定およびデプロイメント JSON の値を表すことができます。例えば、`[:deploy][:app1][:application]` は、app1 アプリケーションの短縮名を表します。

`[:deploy]` の各アプリケーションでは、レシピによって関連するコードブロックが実行されます。コードブロックでは、`deploy_config` がアプリケーション属性を表します。レシピでは、まず `app_root` をアプリケーションのルートディレクトリ `[:deploy][:app_name][:deploy_to]/current` に設定します。次に、Chef の [テンプレートリソース](#) を使用して `redis.yml.erb` から設定ファイルを生成し、`app_root/config` に配置します。

通常、設定ファイルは、Chef 属性によって定義された設定が多数含まれたテンプレートから作成されます。属性を使用すると、テンプレートファイルを書き換えるのではなく、後述するカスタム JSON を使用して設定を変更できます。`redis.yml.erb` テンプレートには、次の行が含まれます。

```
host: <%= @redis[:host] %>
port: <%= @redis[:port] || 6379 %>
```

`<%... %>` 要素は、属性値を表すプレースホルダーです。

- `<%= @redis[:host] %>` は、キャッシュクラスターのホスト名である `redis[:host]` の値を表します。
- `<%= @redis[:port] || 6379 %>` は、`redis[:port]` の値を表します。この属性が定義されていない場合は、デフォルトのポート値である 6379 が使用されます。

template リソースは、次のように機能します。

- `source` と `cookbook` は、それぞれテンプレート名とクックブック名を指定します。
- `mode`、`group`、および `owner` は、設定ファイルにアプリケーションと同じアクセス権を与えます。
- `variables` セクションでは、テンプレートで使用する `@redis` 変数をアプリケーションの `[:redis]` 属性値に設定します。

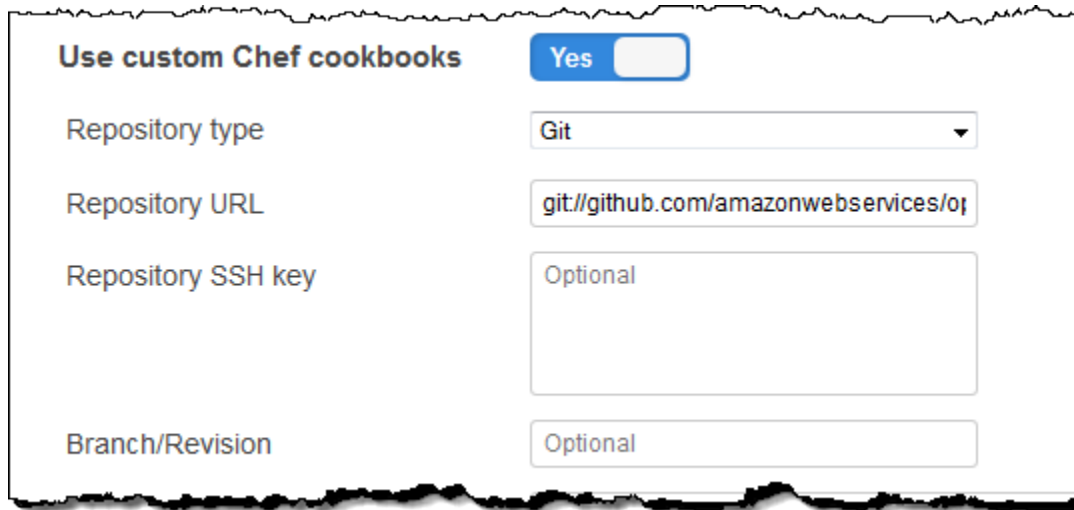
`[:redis]` 属性の値は、後述するカスタム JSON を使用して設定されます。これは、アプリケーションの標準属性ではありません。

- `not_if` ディレクティブは、設定ファイルがすでに存在する場合にレシピで設定ファイルを生成しないことを示します。

クックブックを作成したら、各インスタンスのクックブックキャッシュにデプロイする必要があります。このオペレーションではレシピは実行されません。スタックのインスタンスに新しいクックブックをインストールするだけです。後述するように、通常はレイヤーのライフサイクルイベントにレシピを割り当てることによってレシピを実行します。

カスタムクックブックをデプロイするには

1. AWS OpsWorks スタックスタック ページで、**スタック設定** をクリックし、**を編集** します。
2. [Configuration Management] (構成の管理) セクションで、[Use custom Chef cookbooks] (カスタム Chef のクックブックを使用) を [Yes] (はい) に設定し、クックブックリポジトリの情報を入力します。次に、[Save] (保存) をクリックしてスタック設定を更新します。



The screenshot shows a configuration form for 'Use custom Chef cookbooks'. The form is enclosed in a box with a torn-paper effect. It contains the following fields:

- Use custom Chef cookbooks**: A toggle switch set to 'Yes'.
- Repository type**: A dropdown menu with 'Git' selected.
- Repository URL**: A text input field containing 'git://github.com/amazonwebservices/oj'.
- Repository SSH key**: A text input field with the placeholder text 'Optional'.
- Branch/Revision**: A text input field with the placeholder text 'Optional'.

3. [Stack] (スタック) ページで [Run Command] (コマンドの実行) をクリックし、[Update Custom Cookbooks] (カスタムクックブックの更新) スタックコマンドを選択します。次に、[Update Custom Cookbooks] (カスタムクックブックの更新) をクリックして、インスタンスのクックブックキャッシュに新しいクックブックをインストールします。

Run Command

Settings

Command

Update Custom Cookbooks

Comment

Optional

Deploy comment.

Advanced »

Instances ⓘ

OpsWorks will run this command on 1 of 1 instances. The assigned recipes are run on all selected instances.

Rails App Server

Click to select instances in this layer

rails-app1 ●

Cancel

Update Custom Cookbooks

クックブックを変更した場合は、[Update Custom Cookbooks] をもう一度実行して、最新バージョンをインストールします。この手順の詳細については、「[カスタムクックブックのインストール](#)」を参照してください。

ステップ 4: レシピをイベントに割り当てる LifeCycle

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

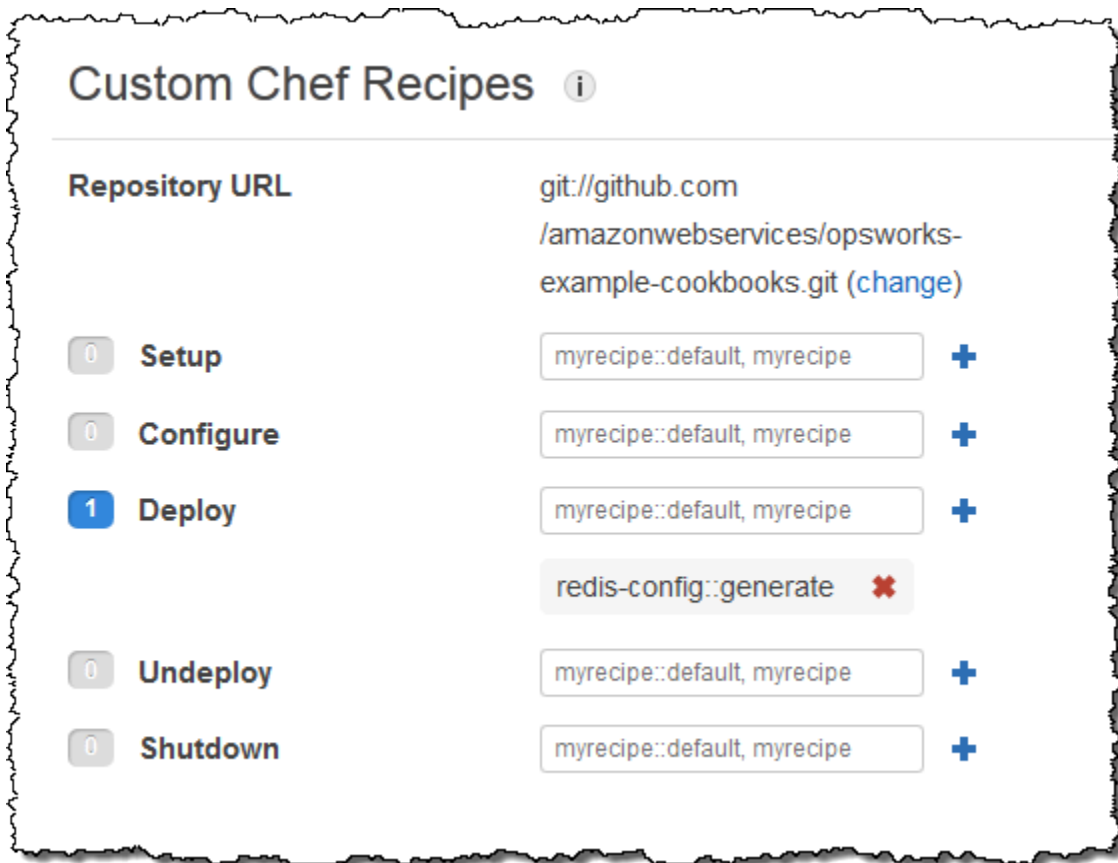
カスタムレシピは[手動](#)で実行できますが、通常は AWS OpsWorks スタックで自動的に実行するのが最善の方法です。すべてのレイヤーには、5 つの [lifecycle events](#) (ライフサイクルイベント) (セットアップ、設定、デプロイ、アンデプロイおよびシャットダウン) のそれぞれに割り当てられたレシピが組み込まれています。インスタンスに対してイベントが発生するたびに、AWS OpsWorks スタックによってインスタンスの各レイヤーの関連付けられたレシピが実行され、対応するタスクが処理されます。例えば、インスタンスの起動が完了すると、AWS OpsWorks スタックは Setup イベントを

トリガーします。このイベントでは、関連付けられたレイヤーの Setup レシピが実行され、通常はパッケージのインストールと設定などのタスクが処理されます。

適切なライフサイクルイベントにレシピを割り当てることで、AWS OpsWorks スタックでレイヤーのインスタンスでカスタムレシピを実行できます。この例では、Rails App Server レイヤーの Deploy イベントに generate.rb レシピを割り当てる必要があります。AWS OpsWorks スタックは、起動時、Setup レシピが完了した後、およびアプリケーションをデプロイするたびに、レイヤーのインスタンスでレシピを実行します。詳細については、「[レシピを自動的に実行する](#)」を参照してください。

Rails アプリケーションサーバーレイヤーのデプロイイベントにレシピを割り当てるには

1. AWS OpsWorks スタックレイヤー ページの Rails App Server で、レシピをクリックし、編集をクリックします。
2. [Custom Chef Recipes] で、完全修飾レシピ名を Deploy イベントに追加し、[+] をクリックします。完全修飾レシピ名では、`cookbookname::recipe` の形式を使用します。`recipe` には `.rb` 拡張子は含まれません。この例では、完全修飾名は `redis-config::generate` です。次に、[Save] をクリックしてレイヤー設定を更新します。



ステップ 5: スタック設定 JSON にアクセス情報を追加する

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

generate.rb レシピは、Redis サーバーのホスト名とポートを表すスタック設定およびデプロイメント JSON の属性ペアによって異なります。これらの属性は標準[:deploy]の名前空間の一部ですが、AWS OpsWorks スタックによって自動的に定義されません。代わりに、カスタム JSON オブジェクトをスタックに追加して属性とその値を定義します。次の例は、この例のカスタム JSON を示しています。

スタック設定およびデプロイメント JSON にアクセス情報を追加するには

1. AWS OpsWorks スタックスタック ページで、スタック設定 をクリックし、 を編集します。
2. [Configuration Management] セクションで、[Custom Chef JSON] ボックスにアクセス情報を追加します。以下の変更を加えると、次の例のようになります。
 - elasticache_redis_example をアプリケーションの短縮名で置き換えます。
 - host と のport値を、 で作成した ElastiCache Redis サーバーインスタンスの値に置き換えます。[ステップ 1: ElastiCache Redis クラスタを作成する](#)。

```
{
  "deploy": {
    "elasticache_redis_example": {
      "redis": {
        "host": "mycluster.XXXXXXXXXX.amazonaws.com",
        "port": "6379"
      }
    }
  }
}
```



The screenshot shows a form in the AWS OpsWorks console. At the top, there is a field for "Branch/Revision" with the value "Optional". Below it is a section titled "Custom Chef JSON" containing a text area with the following JSON code:

```
{
  "deploy": {
    "elasticache_redis_example": {
      "redis": {
        "host": "mycluster.YXXXXXXXXX.amazonaws.com",
        "port": "6379"
      }
    }
  }
}
```

Below the text area, there is a note: "Enter custom JSON that is passed to your Chef recipes for all instances in your stack. You can use this to override and customize built-in recipes or pass variables to your own recipes. [Learn more.](#)" At the bottom right of the form, there are "Cancel" and "Save" buttons.

この方法の利点は、カスタムクックブックにタッチすることなく、ポートまたはホスト値をいつでも変更できることです。AWS OpsWorks スタックは、カスタム JSON を組み込み JSON にマージし、後続のすべてのライフサイクルイベントのスタックのインスタンスにインストールします。これにより、アプリケーションは、「[ステップ 3: カスタムクックブックを作成してデプロイする](#)」で説明した Chef ノード構文を使用して属性値にアクセスできるようになります。次回アプリケーションをデプロイするときには、AWS OpsWorks スタックによって、新しい定義が含まれたスタック設定およびデプロイメント JSON がインストールされ、`generate.rb` によって、ホストとポートの値が更新された設定ファイルが作成されます。

Note

`[:deploy]` にはデプロイするすべてのアプリケーションの属性が自動的に含まれるので、スタック設定 JSON には `[:deploy][elasticache_redis_example]` がすでに含まれています。ただし、`[:deploy][elasticache_redis_example]` には `[:redis]` 属性が含まれず、カスタム JSON で定義することで、AWS OpsWorks スタックはそれらの属性を追加します `[:deploy][elasticache_redis_example]`。カスタム JSON を使用して既存の属性を上書きすることもできます。詳細については、「[属性の上書き](#)」を参照してください。

ステップ 6: アプリをデプロイして実行する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

この例では、Redis を使用する Ruby on Rails アプリケーションがあることを前提としています。設定ファイルにアクセスするには、次のように、redis で config/initializers/redis.rb gem を Gem ファイルに追加し、Rails 初期化子を作成します。

```
REDIS_CONFIG = YAML::load_file(Rails.root.join('config', 'redis.yml'))
$redis = Redis.new(:host => REDIS_CONFIG['host'], :port => REDIS_CONFIG['port'])
```

次に、アプリケーションを示すための [\[create an app\]](#) (アプリケーションを作成) し、Rails アプリケーションサーバーレイヤーのインスタンスに [\[deploy it\]](#) (それをデプロイ) します。これにより、アプリケーションコードが更新され、generate.rb が実行されて設定ファイルが生成されます。アプリケーションを実行すると、Redis ElastiCache インスタンスがインメモリーキーバリューストアとして使用されます。

Amazon S3 バケットの使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

アプリケーションは、イメージやその他のメディアファイルなどの大きなアイテムを保存するために Amazon Simple Storage Service (Amazon S3) バケットをよく使用します。AWS OpsWorks ス

スタックは Amazon S3 の統合サポートを提供していませんが、スタックを簡単にカスタマイズして、アプリケーションが Amazon S3 ストレージを使用できるようにすることができます。このトピックでは、PHP アプリケーションサーバーが属する Linux スタックの使用を例に、アプリケーションに Amazon S3 へのアクセス権限を与える基本的な手順について説明します。基本的な原則は Windows スタックにも当てはまります。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、[「S3 バケットを空にする方法」](#)または[「S3 バケットを削除する方法」](#)を参照してください。

トピック

- [ステップ 1: Amazon S3 バケットを作成する](#)
- [ステップ 2: PHP アプリケーションサーバースタックを作成する](#)
- [ステップ 3: カスタムクックブックを作成してデプロイする](#)
- [ステップ 4: LifeCycle イベントにレシピを割り当てる](#)
- [ステップ 5: スタック設定およびデプロイ属性にアクセス情報を追加する](#)
- [ステップ 6: デプロイと実行 PhotoApp](#)

ステップ 1: Amazon S3 バケットを作成する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

まず Amazon S3 バケットを作成する必要があります。これは、Amazon S3 コンソール、API、または CLI を使用することで直接作成することができますが、AWS CloudFormation テンプレートを使用すると、より簡単にリソースを作成できます。次のテンプレートでは、この例の Amazon S3 バケットを作成し、バケットに対する無制限のアクセスを許可する [IAM ロール](#)を伴う [インスタンスプロファイル](#)をセットアップします。次に、レイヤー設定を使用して、スタックのアプリケーションサーバーのインスタンスにインスタンスプロファイルをアタッチします。これにより、後述のよ

うにアプリケーションがバケットにアクセスできるようになります。インスタンスプロファイルは、Amazon S3 ばかりでなく、さまざまな AWS サービスを統合する場合にも役立ちます。

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "AppServerRootRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [ {
            "Effect": "Allow",
            "Principal": {
              "Service": [ "ec2.amazonaws.com" ]
            },
            "Action": [ "sts:AssumeRole" ]
          } ]
        },
        "Path": "/"
      }
    },
    "AppServerRolePolicies": {
      "Type": "AWS::IAM::Policy",
      "Properties": {
        "PolicyName": "AppServerS3Perms",
        "PolicyDocument": {
          "Statement": [ {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": { "Fn::Join" : [ "", [ "arn:aws:s3:::", { "Ref" :
"AppBucket" } , "/"* ] ]
          } ]
        },
        "Roles": [ { "Ref": "AppServerRootRole" } ]
      }
    },
    "AppServerInstanceProfile": {
      "Type": "AWS::IAM::InstanceProfile",
      "Properties": {
        "Path": "/",
        "Roles": [ { "Ref": "AppServerRootRole" } ]
      }
    }
  }
}
```

```
    }
  },
  "AppBucket" : {
    "Type" : "AWS::S3::Bucket"
  }
},
"Outputs" : {
  "BucketName" : {
    "Value" : { "Ref" : "AppBucket" }
  },
  "InstanceProfileName" : {
    "Value" : { "Ref" : "AppServerInstanceProfile" }
  }
}
}
```

テンプレートを起動すると、以下の項目が実行されます。

- [AWS::S3::Bucket](#) リソースは Amazon S3 バケットを作成します。
- [AWS::IAM::InstanceProfile](#) リソースは、アプリケーションサーバーインスタンスに割り当てられるインスタンスプロファイルを作成します。
- [AWS::IAM::Role](#) リソースによってインスタンスプロファイルのロールが作成されます。
- [AWS::IAM::Policy](#) リソースは Amazon S3 バケットへの無制限アクセスを許可するアクセス許可をロールに設定します。
- テンプレートを起動すると、Outputs コンソールの [AWS CloudFormation] セクションに、バケット名およびインスタンスプロファイル名が表示されます。

これらの値は、スタックとアプリケーションをセットアップするために必要です。

AWS CloudFormation テンプレートの作成方法の詳細については、「[テンプレートの基本を学ぶ](#)」を参照してください。

Amazon S3 バケットを作成するには

1. システムのテキストファイルにサンプルテンプレートをコピーします。

この例では、ファイル名を `appserver.template` としています。

2. [AWS CloudFormation](#) コンソールを開き、[スタックの作成] を選択します。
3. [Stack Name] ボックスに、スタック名を入力します。

この例では、名前を **AppServer** としています。

4. [Upload template file]、[Browse] の順に選択し、ステップ 1 で作成した `appserver.template` ファイルを選択して、[Next Step] を選択します。
5. [Specify Parameters] ページで、[I acknowledge that this template may create IAM resources] を選択し、最後のページに到達するまで、ウィザードの各ページで [Next Step] を選択します。[作成] を選択します。
6. AppServer スタックが `CREATE_COMPLETE` ステータスになったら、それを選択し、出カタブを選択します。

場合によっては、何回か更新してステータスをアップデートする必要があります。

7. 出カタブで、後で使用するために `BucketName` と `InstanceProfileName` の値を記録します。

Note

AWS CloudFormation はスタックという用語を使用して、テンプレートから作成されたりソースのコレクションを参照します。AWS OpsWorks スタックスタックとは異なります。

ステップ 2: PHP アプリケーションサーバースタックを作成する

Important

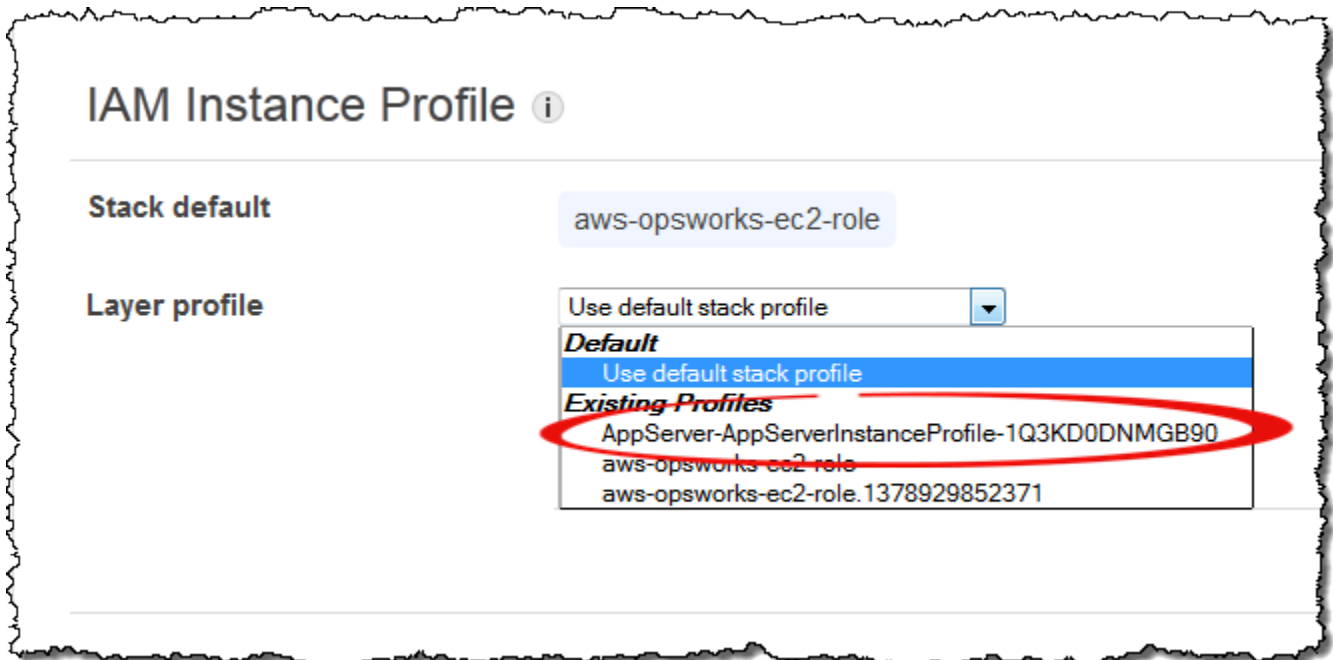
この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

スタックは、PHP アプリケーションサーバーと MySQL の 2 つのレイヤーで構成されており、それぞれに 1 つのインスタンスが存在します。アプリケーションは、Amazon S3 バケットに写真を保存しますが、各写真のメタデータを保持するためのバックエンドデータストアとしては MySQL インスタンスを使用します。

Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、[「S3 バケットを空にする方法」](#)または[「S3 バケットを削除する方法」](#)を参照してください。

スタックを作成するには

1. 新しいスタック **PhotoSite** (を参照) を作成するには、PHP アプリケーションサーバーレイヤーを追加します。両方にデフォルト設定を使用します。詳細については、「[新しいスタックを作成する](#)」および「[OpsWorks レイヤーの作成](#)」を参照してください。
2. [Layers] (レイヤー) ページで、PHP アプリケーションサーバーに対して、[Security] (セキュリティ) を選択し、[Edit] (編集) を選択します。
3. レイヤープロファイルセクションで、AppServer AWS CloudFormation スタックを起動した後、前に記録したインスタンスプロファイル名を選択します。これは のようなものです AppServer-AppServerInstanceProfile-1Q3KD0DNMGB90。AWS OpsWorks スタックは、このプロファイルをレイヤーのすべての Amazon EC2 インスタンスに割り当てます。これにより、レイヤーのインスタンスで実行されているアプリケーションに Amazon S3 バケットにアクセスするアクセス許可が付与されます。



4. PHP アプリケーションサーバーレイヤーにインスタンスを追加し、起動します。インスタンスを追加して起動する方法の詳細については、「[レイヤーへのインスタンスの追加](#)」を参照してください。
5. スタックに MySQL レイヤーを追加し、インスタンスを追加して起動します。レイヤーとインスタンスの両方にデフォルト設定を使用します。特に、MySQL インスタンスは Amazon S3 バ

ケットにアクセスする必要がないため、デフォルトで選択されている標準の AWS OpsWorks スタックインスタンスプロファイルを使用できます。

ステップ 3: カスタムクックブックを作成してデプロイする

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

スタックは、まだ完全には準備できていません。

- アプリケーションが MySQL データベースサーバーと Amazon S3 バケットにアクセスするためには、データベースのホスト名や Amazon S3 バケット名などの情報が必要です。
- MySQL データベースサーバーにデータベースをセットアップし、写真のメタデータを保持するためのテーブルを作成する必要があります。

これらのタスクは手動で処理できますが、Chef recipe を実装し、AWS OpsWorks スタックにレシピを適切なインスタンスで自動的に実行させることがより良い方法です。Chef レシピは、AWS OpsWorks スタックがパッケージのインストールや設定ファイルの作成などのタスクをインスタンスで実行するために使用する特殊な Ruby アプリケーションです。レシピは、クックブックにパッケージ化されます。クックブックには、複数のレシピと関連ファイル (設定ファイルのテンプレートなど) を含めることができます。クックブックは などのリポジトリに配置され GitHub、標準のディレクトリ構造である必要があります。カスタムクックブックリポジトリがまだない場合、これをセットアップする方法の詳細については、「[クックブックリポジトリ](#)」を参照してください。

この例では、クックブックは実装されており、[パブリック GitHub リポジトリ](#) に保存されます。このクックブックには、appsetup.rb と dbsetup.rb という 2 つのレシピと、db-connect.php.erb というテンプレートファイルが含まれています。

appsetup.rb レシピは、アプリケーションがデータベースおよび Amazon S3 バケットにアクセスするために必要となる情報を含む設定ファイルを作成します。これは基本的に [アプリケーションを](#)

[データベースに接続する](#) で説明した `appsetup.rb` レシピを軽微に修正したものです。主な違いはテンプレートに渡される変数であり、これらはアクセス情報を表します。

最初の 4 つの属性はデータベース接続設定を定義し、MySQL インスタンスの作成時に AWS OpsWorks スタックによって自動的に定義されます。

これらの変数には、元のレシピの変数と異なる点が 2 つあります。

- 元のレシピと同じように、`table` 変数は `dbsetup.rb` によって作成されたデータベーステーブルの名前を表し、クックブックの属性ファイルで定義された属性の値に設定されます。

ただし、その属性名は `[:photoapp][:dbtable]` という異なる名前になります。

- `s3bucket` 変数はこの例に固有のものであり、Amazon S3 バケット名 `[:photobucket]` を表す属性の値に設定されます。

`[:photobucket]` は、後述するように、カスタム JSON を使用して定義されます。属性の詳細については、「[属性](#)」を参照してください。

属性の詳細については、「[属性](#)」を参照してください。

`dbsetup.rb` レシピは、各写真のメタデータを保持するためのデータベーステーブルをセットアップします。これは、基本的には `dbsetup.rb` レシピに若干の変更を加えたものです。このレシピの詳細については、「[データベースのセットアップ](#)」を参照してください。

この例と元のレシピの唯一の相異点はデータベーススキーマです。このデータベーススキーマには、Amazon S3 バケットに保存される各写真の ID、URL およびキャプションが含まれる 3 つの列があります。

レシピはすでに実装されているため、必要なのは、PhotoApp クックブックを各インスタンスのクックブックキャッシュにデプロイすることです。AWS OpsWorks スタックは、後述するように、適切なライフサイクルイベントが発生したときにキャッシュされたレシピを実行します。

photoapp クックブックをデプロイするには

- AWS OpsWorks スタックスタック ページで、スタック設定 を選択し、編集 を選択します。
- [Configuration Management] セクションで、次のように設定します。
 - [Use custom Chef cookbooks] を [Yes] に設定します。
 - [Repository type] を Git に設定します。

- Repository URL を `git://github.com/amazonwebservicesservices/opsworks-example-cookbooks.git` に設定します。
- [Stack] (スタックスタック) ページで [Run Command] (コマンドの実行) を選択し、[Update Custom Cookbooks] (カスタムクックブックの更新) スタックコマンドを選択します。次に、[Update Custom Cookbooks] (カスタムクックブックの更新) を選択して、インスタンスクックブックキャッシュに新しいクックブックをインストールします。

Run Command

Settings

Command

Update Custom Cookbooks

Comment

Optional

Deploy comment.

[Advanced »](#)

Instances ⓘ

OpsWorks will run this command on **1 of 1** instances. The assigned recipes are run on all selected instances.

Rails App Server

rails-app1 ●

[Click to select instances in this layer](#)

Cancel

Update Custom Cookbooks

ステップ 4: LifeCycle イベントにレシピを割り当てる

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

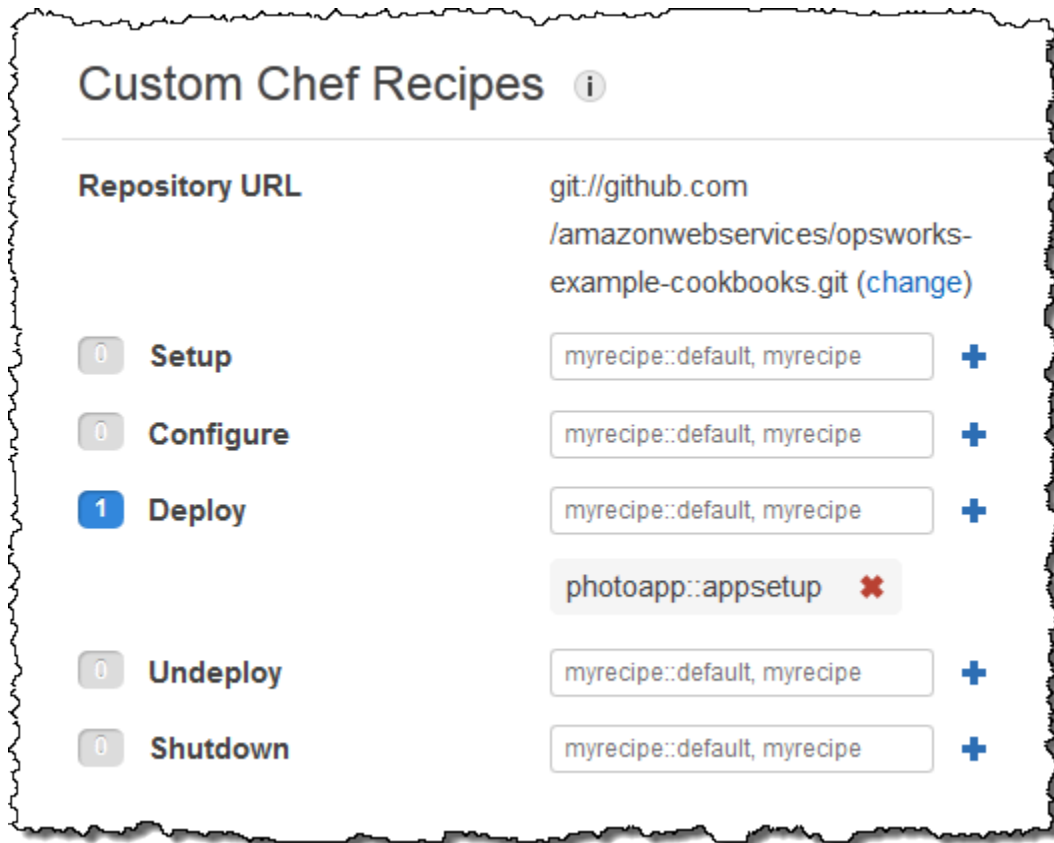
カスタムレシピは [手動](#) で実行できますが、通常は AWS OpsWorks スタックで自動的に実行するのが最善の方法です。すべてのレイヤーには、5 つの [lifecycle events](#) (ライフサイクルイベント) (セットアップ、設定、デプロイ、アンデプロイ、およびシャットダウン) のそれぞれに割り当てられたレシ

ピが組み込まれています。インスタンスでイベントが発生するたびに、AWS OpsWorks Stacks はインスタンスの各レイヤーに関連するレシピを実行し、必要なタスクを処理します。例えば、インスタンスの起動が完了すると、AWS OpsWorks スタックは Setup イベントをトリガーして Setup レシピを実行します。通常、これはパッケージのインストールや設定などのタスクを処理します。

AWS OpsWorks スタックは、各レシピを適切なライフサイクルイベントに割り当てることで、レイヤーのインスタンスでカスタムレシピを実行できます。AWS OpsWorks スタックは、レイヤーの組み込みレシピが完了した後にカスタムレシピを実行します。この例では、`appsetup.rb`を PHP アプリケーションサーバーレイヤーの Deploy イベントと `dbsetup.rb` MySQL レイヤーの Deploy イベントに割り当てます。AWS OpsWorks スタックは、起動時、組み込みセットアップレシピの完了後、およびビルドされた Deploy レシピの終了後にアプリケーションをデプロイするたびに、関連するレイヤーのインスタンスでレシピを実行します。詳細については、「[レシピを自動的に実行する](#)」を参照してください。

レイヤーの Deploy イベントにカスタムレシピを割り当てるには

1. AWS OpsWorks スタックレイヤーページで、PHP アプリケーションサーバー でレシピ を選択し、編集 を選択します。
2. [Custom Chef Recipes] で、レシピ名を Deploy イベントに追加し、[+] を選択します。名前は、Chef の `cookbookname::recipename` 形式の名前である必要があります。`recipename` には `.rb` 拡張子は含まれません。この例では、`photoapp::appsetup` と入力します。次に、[Save] を選択してレイヤー設定を更新します。



3. [Layers] (レイヤー) ページで、MySQL レイヤーの [Actions] (アクション) 列で [edit] (編集) を選択します。
4. レイヤーの Deploy イベントに `photoapp::dbsetup` を追加し、新しい設定を保存します。

ステップ 5: スタック設定およびデプロイ属性にアクセス情報を追加する

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

`appsetup.rb` レシピは、AWS OpsWorks 各インスタンスにインストールされ、[スタックとデプロイされたアプリケーションに関する詳細情報を含むスタックスタックの設定とデプロイ属性](#) のデー

タによって異なります。オブジェクトの `deploy` 属性は以下の構造になっており、わかりやすいように、JSON オブジェクトの形式で表示されます。

```
{
  ...
  "deploy": {
    "app1": {
      "application" : "short_name",
      ...
    }
    "app2": {
      ...
    }
    ...
  }
}
```

`deploy` ノードには、デプロイされる各アプリケーションの属性が含まれます。この属性の名前はアプリケーションの短縮名です。各アプリケーション属性には、アプリケーションの設定 (ドキュメントのルートやアプリケーションタイプなど) を定義する属性のセットが含まれています。`deploy` 属性のリストについては、「[deploy 属性](#)」を参照してください。レシピにスタック設定およびデプロイ属性値を記述するには、Chef 属性構文を使用します。例えば、`[:deploy][:app1][:application]` は、`app1` アプリケーションの短縮名を表します。

カスタムレシピは、データベースおよび Amazon S3 のアクセス情報を表すいくつかのスタック構成およびデプロイメント属性によって異なります。

- などのデータベース接続属性は `[:deploy][:database][:host]`、MySQL レイヤーの作成時に AWS OpsWorks スタックによって定義されます。
- テーブル名属性 `[:photoapp][:dbtable]` は、カスタムクックブックの属性ファイルで定義され、`foto` に設定されます。
- バケット名属性 `[:photobucket]` を定義するには、カスタム JSON を使用して、その属性をスタック設定およびデプロイ属性に追加する必要があります。

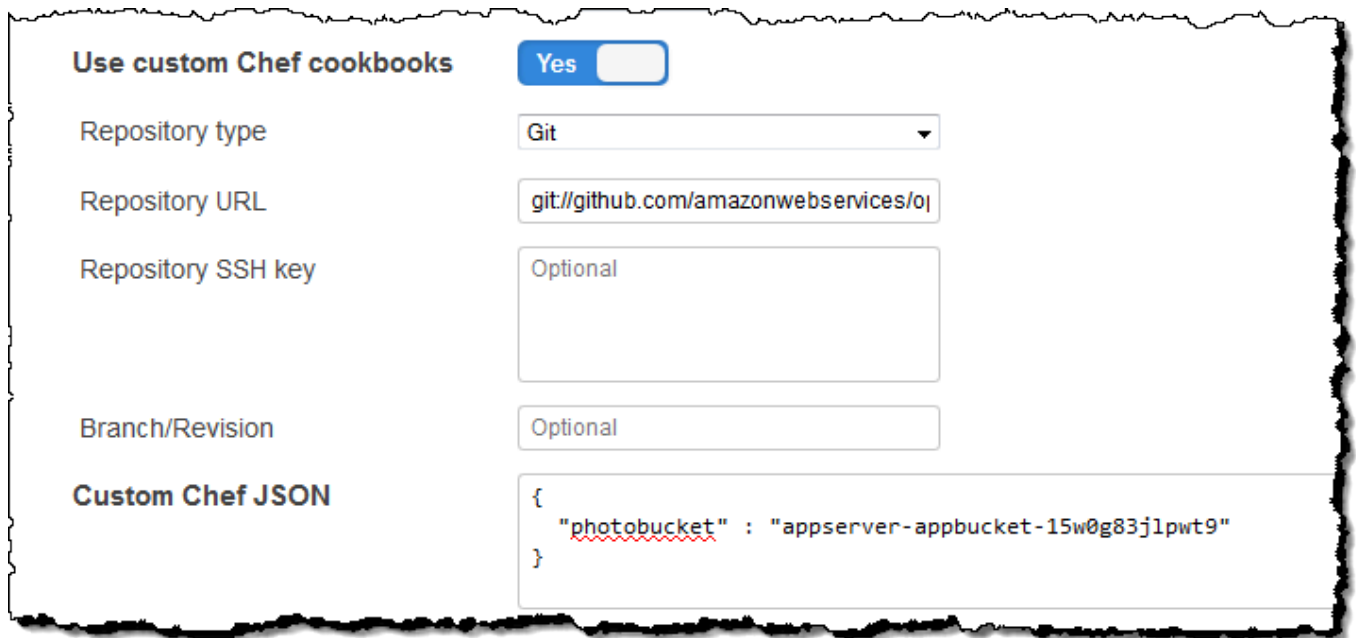
Amazon S3 バケット名属性を定義するには

1. AWS OpsWorks スタックスタックページで、`スタック設定` を選択し、`を編集` します。

2. [Configuration Management] セクションで、[Custom Chef JSON] ボックスにアクセス情報を追加します。次のようになります。

```
{
  "photobucket" : "yourbucketname"
}
```

[*yourbucketname*] を、「[ステップ 1: Amazon S3 バケットを作成する](#)」で書き留めたバケット名で置き換えます。



The screenshot shows the configuration interface for Custom Chef cookbooks. The 'Use custom Chef cookbooks' toggle is set to 'Yes'. The 'Repository type' is 'Git', the 'Repository URL' is 'git://github.com/amazonwebservices/oj', and the 'Repository SSH key' and 'Branch/Revision' fields are 'Optional'. The 'Custom Chef JSON' field contains the following JSON:

```
{
  "photobucket" : "appserver-appbucket-15w0g83jlpwt9"
}
```

AWS OpsWorks スタックは、カスタム JSON をスタックのインスタンスにインストールする前にスタック設定とデプロイ属性にマージします。その後、[:photobucket] 属性からバケット名を取得appsetup.rbできます。バケットを変更する場合は、レシピに変更を加える必要はありません。新しいバケット名を渡すように[属性を上書きする](#)だけです。

ステップ 6: デプロイと実行 PhotoApp

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

この例では、アプリケーションも実装されており、[パブリック GitHub リポジトリ](#) に保存されます。必要な処理は、アプリケーションをスタックに追加し、アプリケーションサーバーにデプロイして実行することだけです。

アプリケーションをスタックに追加してアプリケーションサーバーにデプロイするには

1. [Apps] ページを開き、[Add an app] を選択します。
2. [Add App] ページで、次のように設定します。
 - [Name] を **[PhotoApp]** に設定します。
 - [App type] を [PHP] に設定します。
 - [Document root] を **[web]** に設定します。
 - [Repository type] を [Git] に設定します。
 - Repository URL を **`git://github.com/awslabs/opsworks-demo-php-photo-share-app.git`** に設定します。
 - [Add App] を選択し、その他の設定はデフォルト値を受け入れます。

Add App

Settings

Name

App type

Document root

Application Source

Repository type

Repository URL


Repository SSH key

Branch/Revision

3. アプリページで、PhotoApp アプリのアクション列でデプロイを選択します。

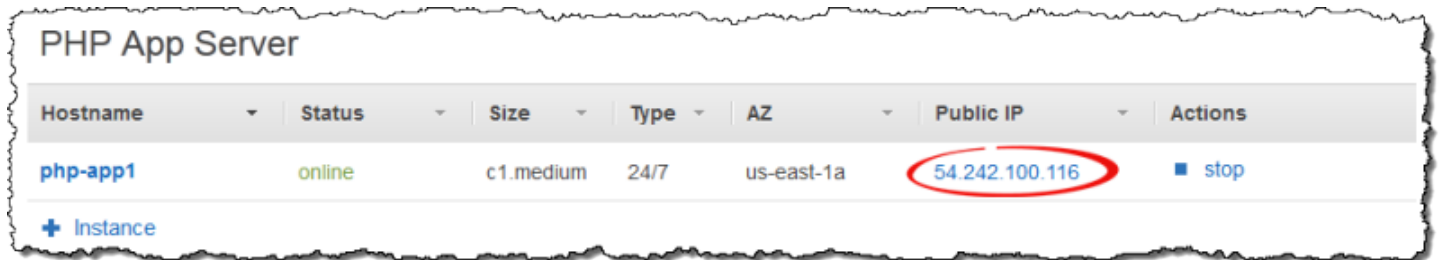
Apps

An app represents code stored in a repository that you want to install on application server instances. When you deploy the app, OpsWorks downloads the code from the repository to the specified server instances. [Learn more](#).

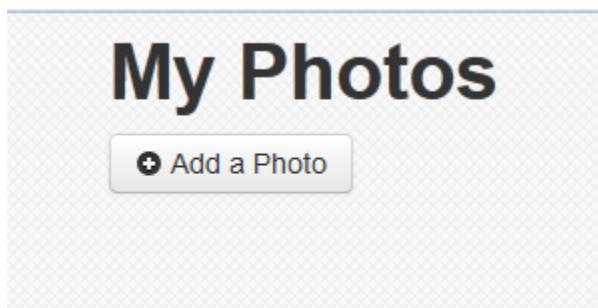
Name	Type	Last Deployment	Actions
PhotoApp	PHP	2013-09-27 17:38:35 UTC	 deploy  edit  delete
+ App			

4. デフォルト値を受け入れ、[Deploy] を選択してアプリケーションをサーバーにデプロイします。

を実行するには PhotoApp、インスタンス ページに移動し、PHP App Server インスタンスのパブリック IP アドレスを選択します。



次のようなユーザーインターフェイスが表示されます。Amazon S3 バケットに写真を保存し、バックエンドデータストアにメタデータを保存するには、[Add a Photo] (写真を追加) を選択します。



AWS OpsWorks スタック AWS CodePipeline での の使用

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[AWS CodePipeline](#) では、Amazon Simple Storage Service (Amazon S3) CodeCommit、などのソースからのコード変更を追跡する継続的デリバリーパイプラインを作成できます [GitHub](#)。を使用して CodePipeline、Chef 11.10、Chef 12、Chef 12.2 スタックの AWS OpsWorks スタックへの Chef クックブックとアプリケーションコードのリリースを自動化できます。このセクションの例では、AWS OpsWorks スタックレイヤーで実行するコードのデプロイツール CodePipeline として、からシンプルなパイプラインを作成して使用方法について説明します。

Note

CodePipeline および AWS OpsWorks スタックの統合は、Chef 11.4 以前のスタックへのデプロイではサポートされていません。

トピック

- [AWS CodePipelineAWS OpsWorks スタックあり - Chef 12 スタック](#)
- [AWS CodePipelineAWS OpsWorks スタックあり - Chef 11 スタック](#)

AWS CodePipelineAWS OpsWorks スタックあり - Chef 12 スタック

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[AWS CodePipeline](#) では、Amazon Simple Storage Service (Amazon S3) CodeCommit、などのソースからのコード変更を追跡する継続的デリバリーパイプラインを作成できます[GitHub](#)。このトピックの例では、AWS OpsWorks スタックレイヤーで実行するコードのデプロイツール CodePipeline として、からシンプルなパイプラインを作成して使用方法について説明します。この例では、シンプルな [Node.js アプリケーションのパイプラインを作成し](#)、Chef 12 AWS OpsWorks スタック (この場合は 1 つのインスタンス) のレイヤー内のすべてのインスタンスでアプリケーションを実行するようにスタックに指示します。

Note

このトピックでは、パイプラインを使用して Chef 12 スタックでアプリケーションを実行および更新する方法について説明します。パイプラインを使用して Chef 11.10 スタックでアプリケーションを実行および更新する方法については、「[AWS CodePipelineAWS OpsWorks スタックあり - Chef 11 スタック](#)」を参照してください。Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの

削除の詳細については、[「S3 バケットを空にする方法」](#)または[「S3 バケットを削除する方法」](#)を参照してください。

トピック

- [前提条件](#)
- [サポートされている他のシナリオ](#)
- [ステップ 1: スタックにスタック、レイヤー、インスタンスを作成する AWS OpsWorks](#)
- [ステップ 2: カスタムクックブックを使用するようにスタックとレイヤーを設定する](#)
- [ステップ 3: アプリケーションコードを Amazon S3 バケットにアップロードする](#)
- [ステップ 4: AWS OpsWorks スタックにアプリケーションを追加する](#)
- [ステップ 5: でパイプラインを作成する CodePipeline](#)
- [ステップ 6: AWS OpsWorks スタックでアプリケーションのデプロイを確認する](#)
- [ステップ 7 \(オプション\): アプリコードを更新して、アプリを自動的に CodePipeline 再デプロイすることを確認する](#)
- [ステップ 8 \(オプション\): リソースをクリーンアップする](#)

前提条件

このウォークスルーを開始する前に、次のすべてのタスクを実行するための管理者権限があることを確認してください。AdministratorAccess ポリシーが適用されたグループのメンバーにすることも、次の表に示すアクセス許可とポリシーを持つグループのメンバーにすることもできます。セキュリティのベストプラクティスとして、必要な権限を個別のユーザーアカウントに割り当てるのではなく、次のタスクを実行する権限を持つグループにユーザーが属している必要があります。

でセキュリティグループを作成しIAM、そのグループにアクセス許可を割り当てる方法の詳細については、IAM [「ユーザーグループの作成」](#)を参照してください。AWS OpsWorks スタックのアクセス許可の管理の詳細については、[「ベストプラクティス: アクセス許可の管理」](#)を参照してください。

アクセス許可	グループにアタッチすることが推奨されるポリシー
スタックでスタック、レイヤー、インスタンスを作成および編集 AWS OpsWorks します。	AWSOpsWorks_FullAccess

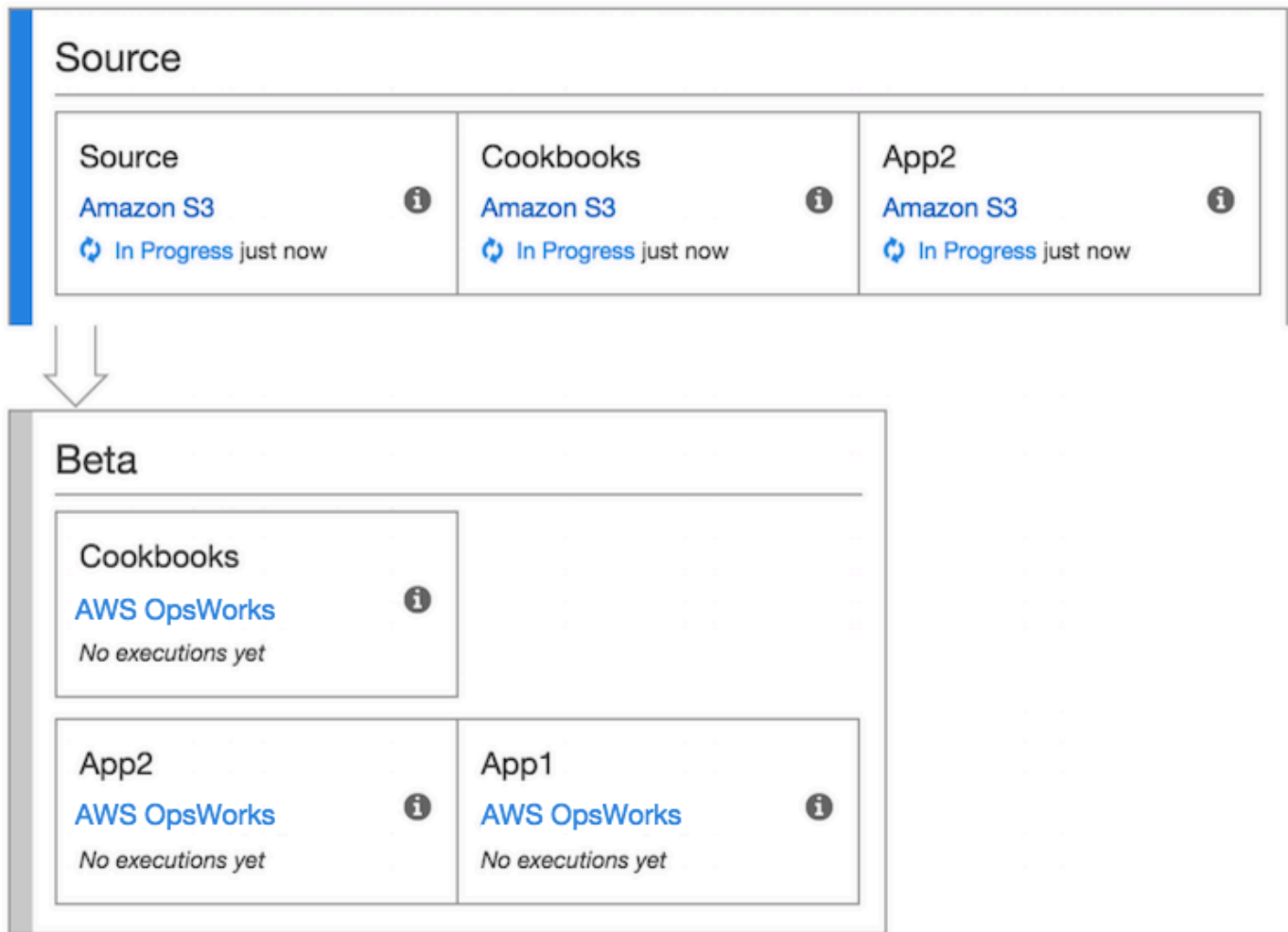
アクセス許可	グループにアタッチすることが推奨されるポリシー
AWS CloudFormationでテンプレートを作成、編集、実行します。	AmazonCloudFormationFullAccess
Amazon S3 バケットの作成、編集、アクセスを行います。	AmazonS3FullAccess
でパイプライン、特に AWS OpsWorks スタックをプロバイダーとして使用するパイプラインを作成 CodePipeline、編集、実行します。	AWSCodePipeline_FullAccess

Amazon EC2キーペアも必要です。このチュートリアルでサンプルスタック、レイヤー、インスタンスを作成する AWS CloudFormation テンプレートを実行するときに、このキーペアの名前を指定するように求められます。Amazon EC2コンソールでキーペアを取得する方法の詳細については、Amazon [ドキュメントの「キーペアの作成」](#)を参照してください。EC2キーペアは、米国東部 (バージニア北部) リージョンにある必要があります。そのリージョンに既存のキーペアがある場合は、そのキーペアを使用できます。

サポートされている他のシナリオ

このウォークスルーでは、[Source] を 1 つ、[Deploy] ステージを 1 つ含むシンプルなパイプラインを作成します。ただし、AWS OpsWorks スタックをプロバイダーとして使用する、より複雑なパイプラインを作成できます。サポートされているパイプラインとシナリオの例を以下に示します。

- パイプラインを編集し、Chef クックブックを [Source] ステージに、更新されたクックブックの関連ターゲットを [Deploy] に追加できます。この場合は、ソースを変更した際にクックブックの更新をトリガーする [デプロイ] アクションを追加します。更新されたクックブックは、アプリの前にデプロイされます。
- カスタムクックブックと複数のアプリケーションを使用して複雑なパイプラインを作成し、AWS OpsWorks スタックスタックにデプロイできます。パイプラインはアプリケーションソースとクックブックソースの両方の変化を追跡し、変更を行った際に再デプロイを行います。同様の複雑なパイプラインの例を以下に示します。



の使用の詳細については CodePipeline、[「CodePipeline ユーザーガイド」](#)を参照してください。

ステップ 1: スタックにスタック、レイヤー、インスタンスを作成する AWS OpsWorks

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックをパイプラインのデプロイプロバイダーとして使用するには、まずスタック、レイヤー、およびレイヤーに少なくとも1つのインスタンスが必要です。「Linux AWS OpsWorks スタックの開始方法」または「[Windows スタックの開始方法](#)」の手順に従ってスタックを作成できますが、時間を節約するために、この例では AWS CloudFormation テンプレートを使用して Linux ベースの Chef 12 スタック、レイヤー、インスタンスを作成します。<https://docs.aws.amazon.com/opsworks/latest/userguide/gettingstarted-linux.html>このテンプレートで作成されたインスタンスは、Amazon Linux 2016.03 を実行します。インスタンスタイプは c3.large です。テンプレートによりカスタムクックブックを使用するようにスタックが設定されませんが、チュートリアルの後半でこのように設定します。

Important

AWS CloudFormation テンプレートは、後でアプリケーションをアップロードする Amazon S3 バケットと同じリージョンと、後でパイプラインを作成するリージョンに保存して実行する必要があります CodePipeline。現時点では、は米国東部 (バージニア北部) リージョン (us-east-1) でのみ AWS OpsWorks スタックプロバイダー CodePipeline をサポートしています。このチュートリアルのすべてのリソースは、米国東部 (バージニア北部) リージョンで作成する必要があります。

スタックの作成に失敗した場合、アカウントで許可されるIAMロールの最大数に近づいている可能性があります。またスタックの作成は、アカウントがインスタンスを c3.large インスタンスタイプで起動できないときに、失敗する場合があります。例えば、AWS 無料利用枠を使用している場合に、Root device type: must be included in EBSなどのエラーが発生することがあります。AWS 無料利用枠によって課される制限など、アカウントで作成が許可されているインスタンスタイプに制限がある場合は、テンプレートのインスタンスブロックの InstanceTypeパラメータの値を、アカウントが使用できるインスタンスタイプに変更してみてください。

を使用してスタック、レイヤー、インスタンスを作成するには AWS CloudFormation

1. 次の AWS CloudFormation テンプレートを新しいプレーンテキストドキュメントにコピーします。ローカルコンピュータ上の便利な場所にファイルを保存し、NewOpsWorksStack.template という名前、または便利な別の名前を付けます。

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Mappings": {
    "Region2Principal": {
```



```
"us-east-1": {
  "EC2Principal": "ec2.amazonaws.com",
  "OpsWorksPrincipal": "opsworks.amazonaws.com"
},
"us-west-2": {
  "EC2Principal": "ec2.amazonaws.com",
  "OpsWorksPrincipal": "opsworks.amazonaws.com"
},
"us-west-1": {
  "EC2Principal": "ec2.amazonaws.com",
  "OpsWorksPrincipal": "opsworks.amazonaws.com"
},
"eu-west-1": {
  "EC2Principal": "ec2.amazonaws.com",
  "OpsWorksPrincipal": "opsworks.amazonaws.com"
},
"ap-southeast-1": {
  "EC2Principal": "ec2.amazonaws.com",
  "OpsWorksPrincipal": "opsworks.amazonaws.com"
},
"ap-northeast-1": {
  "EC2Principal": "ec2.amazonaws.com",
  "OpsWorksPrincipal": "opsworks.amazonaws.com"
},
"ap-northeast-2": {
  "EC2Principal": "ec2.amazonaws.com",
  "OpsWorksPrincipal": "opsworks.amazonaws.com"
},
"ap-southeast-2": {
  "EC2Principal": "ec2.amazonaws.com",
  "OpsWorksPrincipal": "opsworks.amazonaws.com"
},
"sa-east-1": {
  "EC2Principal": "ec2.amazonaws.com",
  "OpsWorksPrincipal": "opsworks.amazonaws.com"
},
"cn-north-1": {
  "EC2Principal": "ec2.amazonaws.com.cn",
  "OpsWorksPrincipal": "opsworks.amazonaws.com.cn"
},
"eu-central-1": {
  "EC2Principal": "ec2.amazonaws.com",
  "OpsWorksPrincipal": "opsworks.amazonaws.com"
}
}
```

```
    }
  },
  "Parameters": {
    "EC2KeyName": {
      "Type": "String",
      "Description": "The name of an existing EC2 key pair that lets you use SSH to
connect to the OpsWorks instance."
    }
  },
  "Resources": {
    "CPOpsDeploySecGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupDescription": "Lets you manage OpsWorks instances to which you deploy
apps with CodePipeline"
      }
    },
    "CPOpsDeploySecGroupIngressHTTP": {
      "Type": "AWS::EC2::SecurityGroupIngress",
      "Properties": {
        "IpProtocol": "tcp",
        "FromPort": "80",
        "ToPort": "80",
        "CidrIp": "0.0.0.0/0",
        "GroupId": {
          "Fn::GetAtt": [
            "CPOpsDeploySecGroup", "GroupId"
          ]
        }
      }
    },
    "CPOpsDeploySecGroupIngressSSH": {
      "Type": "AWS::EC2::SecurityGroupIngress",
      "Properties": {
        "IpProtocol": "tcp",
        "FromPort": "22",
        "ToPort": "22",
        "CidrIp": "0.0.0.0/0",
        "GroupId": {
          "Fn::GetAtt": [
            "CPOpsDeploySecGroup", "GroupId"
          ]
        }
      }
    }
  }
}
```

```
},
"MyStack": {
  "Type": "AWS::OpsWorks::Stack",
  "Properties": {
    "Name": {
      "Ref": "AWS::StackName"
    },
    "ServiceRoleArn": {
      "Fn::GetAtt": [
        "OpsWorksServiceRole",
        "Arn"
      ]
    },
  },
  "ConfigurationManager" : { "Name": "Chef","Version": "12" },
  "DefaultOs": "Amazon Linux 2016.03",
  "DefaultInstanceProfileArn": {
    "Fn::GetAtt": [
      "OpsWorksInstanceProfile",
      "Arn"
    ]
  },
  "UseCustomCookbooks": "false"
},
"MyLayer": {
  "Type": "AWS::OpsWorks::Layer",
  "Properties": {
    "StackId": {
      "Ref": "MyStack"
    },
  },
  "Name": "Node.js App Server",
  "Type": "custom",
  "Shortname": "app1",
  "EnableAutoHealing": "true",
  "AutoAssignElasticIps": "false",
  "AutoAssignPublicIps": "true",
  "CustomSecurityGroupIds": [
    {
      "Fn::GetAtt": [
        "CPOpsDeploySecGroup", "GroupId"
      ]
    }
  ]
},
```

```
"DependsOn": [
  "MyStack",
  "CP0psDeploySecGroup"
],
"OpsWorksServiceRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": [
              {
                "Fn::FindInMap": [
                  "Region2Principal",
                  {
                    "Ref": "AWS::Region"
                  }
                ],
                "OpsWorksPrincipal"
              }
            ]
          }
        }
      ],
      "Action": [
        "sts:AssumeRole"
      ]
    }
  }
},
"Path": "/",
"Policies": [
  {
    "PolicyName": "opsworks-service",
    "PolicyDocument": {
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "ec2:*",
            "iam:PassRole",
            "cloudwatch:GetMetricStatistics",
            "elasticloadbalancing:*"
          ]
        }
      ]
    }
  }
]
```

```
        ],
        "Resource": "*"
    }
]
}
]
},
"OpsWorksInstanceProfile": {
    "Type": "AWS::IAM::InstanceProfile",
    "Properties": {
        "Path": "/",
        "Roles": [
            {
                "Ref": "OpsWorksInstanceRole"
            }
        ]
    }
},
"OpsWorksInstanceRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
        "AssumeRolePolicyDocument": {
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": [
                            {
                                "Fn::FindInMap": [
                                    "Region2Principal",
                                    {
                                        "Ref": "AWS::Region"
                                    },
                                ],
                                "EC2Principal"
                            }
                        ]
                    }
                }
            ]
        },
        "Action": [
            "sts:AssumeRole"
        ]
    }
}
```

```
    ]
  },
  "Path": "/",
  "Policies": [
    {
      "PolicyName": "s3-get",
      "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "s3:GetObject"
            ],
            "Resource": "*"
          }
        ]
      }
    }
  ]
}
],
"myinstance": {
  "Type": "AWS::OpsWorks::Instance",
  "Properties": {
    "LayerIds": [
      {
        "Ref": "MyLayer"
      }
    ],
    "StackId": {
      "Ref": "MyStack"
    },
    "InstanceType": "c3.large",
    "SshKeyName": {
      "Ref": "EC2KeyPairName"
    }
  }
}
},
"Outputs": {
  "StackId": {
    "Description": "Stack ID for the newly created AWS OpsWorks stack",
    "Value": {
```

```
        "Ref": "MyStack"
    }
}
}
```

2. にサインイン AWS Management Console し、<https://console.aws.amazon.com/cloudformation> で AWS CloudFormation コンソールを開きます。
3. AWS CloudFormation ホームページで、スタックの作成 を選択します。
4. [Select Template] ページの [Choose a template] エリアで、[Upload a template to Amazon S3]、[Browse] の順に選択します。
5. ステップ 1 で保存した AWS CloudFormation テンプレートを参照し、「を開く」を選択します。[Select Template] ページで、[Next] を選択します。

Select Template

Select the template that describes the stack that you want to create. A stack is a group of related resources that you manage as a single unit.

Design a template Use AWS CloudFormation Designer to create or modify an existing template. [Learn more.](#)

Design template

Choose a template A template is a JSON-formatted text file that describes your stack's resources and their properties. [Learn more.](#)

Select a sample template

Upload a template to Amazon S3

NewOpsWorksStack.template

Specify an Amazon S3 template URL

Cancel

Next

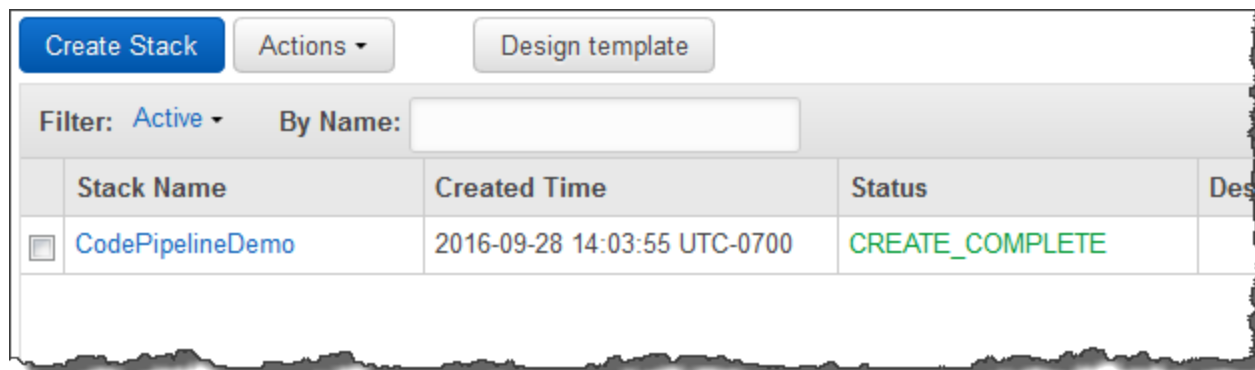
6. 詳細の指定ページで、スタックに という名前を付けるかCodePipelineDemo、アカウントに固有のスタック名を指定します。スタックに別の名前を付ける場合は、このウォークスルー中のスタック名スループットを変更してください。
7. パラメータ領域で、作成後に AWS OpsWorks スタックインスタンスへのアクセスに使用する EC2キーペアの名前を指定します。[Next (次へ)] を選択します。
8. [Options(オプション)] ページで、[Next(次へ)] を選択します。(このページの設定はこのウォークスルーでは必要ありません)
9. このチュートリアルで使用する AWS CloudFormation テンプレートは、IAMロール、インスタンスプロファイル、インスタンスを作成します。

⚠ Important

の作成 を選択する前に、コスト を選択して、このテンプレートでリソースを作成 AWS するために発生する可能性のある料金を見積もります。

IAM リソースの作成が許容できる場合は、このテンプレートによってIAMリソース AWS CloudFormation の作成が発生する可能性があることを確認するチェックボックスを選択し、 の作成を選択します。IAM リソースの作成が受け入れられない場合は、この手順を続行できません。

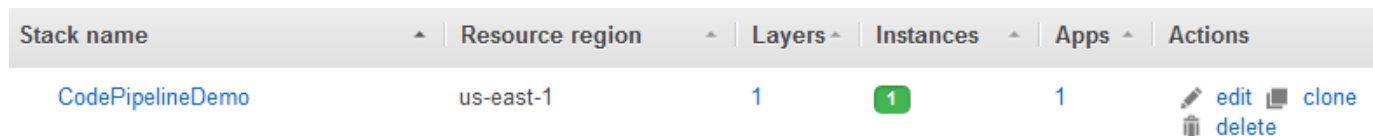
10. AWS CloudFormation ダッシュボードでは、スタックの作成の進行状況を表示できます。次のステップに進む前に、ステータス列に CREATE_COMPLETE が表示されるまで待ちます。



Stack Name	Created Time	Status	Des
<input type="checkbox"/> CodePipelineDemo	2016-09-28 14:03:55 UTC-0700	CREATE_COMPLETE	

スタックでの AWS OpsWorks スタックの作成を確認するには

1. で AWS OpsWorks コンソールを開きます <https://console.aws.amazon.com/opsworks/>。
2. AWS OpsWorks スタックダッシュボードで、作成したスタックを表示します。



Stack name	Resource region	Layers	Instances	Apps	Actions
CodePipelineDemo	us-east-1	1	1	1	edit clone delete

3. スタックを開き、レイヤーとインスタンスを表示します。レイヤーとインスタンスが、AWS CloudFormation テンプレートで指定された名前やその他のメタデータで作成されたことを確認します。カスタム Chef クックブックとレシピを使用するようにスタックとレイヤーを設定する準備ができました。

ステップ 2: カスタムクックブックを使用するようにスタックとレイヤーを設定する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックの Chef 12 スタックでは、カスタムアプリケーションレイヤーを構築するために、独自のクックブックまたはコミュニティで作成されたクックブックが必要です。このチュートリアルでは、[Chef のクックブック](#) とレシピのセットが含まれたリポジトリを参照できます。これらのレシピにより、Node.js パッケージとその依存関係がインスタンスにインストールされます。他の Chef レシピを使用して、「[ステップ 4: AWS OpsWorks スタックにアプリケーションを追加する](#)」で用意することになる Node.js アプリケーションをデプロイします。このステップで指定した Chef レシピは、によってアプリケーションの新しいバージョンがデプロイされるたびに実行されず CodePipeline。

1. AWS OpsWorks スタックコンソールで、で作成したスタックを開きます[ステップ 1: スタックにスタック、レイヤー、インスタンスを作成する AWS OpsWorks](#)。[Stack Settings]、[Edit] の順に選択します。
2. [Use custom Chef cookbooks] を [Yes] に設定します。これにより、関連するカスタムクックブック設定が表示されます。
3. [Repository type] ドロップダウンリストから、[S3 Archive] を選択します。CodePipeline との両方を使用するには AWS OpsWorks、クックブックソースが S3 である必要があります。
4. リポジトリで URL、を指定します `https://s3.amazonaws.com/opsworks-demo-assets/opsworks-linux-demo-cookbooks-nodejs.tar.gz`。設定は以下のようになります。

Use custom Chef cookbooks	<input checked="" type="checkbox"/>
Repository type	S3 Archive
Repository URL	<code><s-linux-demo-cookbooks-nodejs.tar.gz</code>
Access key ID	Optional
Secret access key	Optional

5. [Save] を選択します。
6. ナビゲーションペインで [Layers] (レイヤー) を選択します。
7. [ステップ 1: スタックにスタック、レイヤー、インスタンスを作成する AWS OpsWorks](#) で作成したレイヤーの [設定] を選択します。
8. [General Settings] タブで、レイヤー名が Node.js App Server で、レイヤーの短い名前が app1 であることを確認します。[Recipes] を選択します。
9. [Recipes] (レシピ) タブで、[Deploy] (デプロイ) のライフサイクルイベント中に実行するレシピとして **nodejs_demo** を指定します。[Save] を選択します。
10. セキュリティタブのセキュリティグループのドロップダウンリストから、AWS-OpsWorks-Webapp セキュリティグループを選択します。
11. [Save] を選択します。

ステップ 3: アプリケーションコードを Amazon S3 バケットにアップロードする

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

パイプラインの設定の一部としてコードリポジトリへのリンクを指定する必要があるため、パイプラインを作成する前にコードリポジトリを準備してください。このウォークスルーでは、Node.js アプリケーションを Amazon S3 バケットにアップロードします。

CodePipeline はソースから直接、GitHub またはソース CodeCommit としてコードを使用できませんが、このチュートリアルでは Amazon S3 バケットの使用方法を示します。このチュートリアルでは、サンプル [\[Node.js app\]](#) (Node.js アプリケーション) を自分の Amazon S3 バケットにアップロードして、アプリケーションを変更できるようにします。このステップで作成する Amazon S3 バケットにより CodePipeline はアプリケーションコードの変更を検出し、変更されたアプリケーションを自動的にデプロイできます。必要に応じて、既存のバケットを使用できます。バケットが CodePipeline ドキュメントの [Simple Pipeline チュートリアル \(Amazon S3 バケット\)](#) で説明されている基準を満たしていることを確認します。

Important

Amazon S3 バケットは、後でパイプラインを作成する同じリージョンに存在する必要があります。現時点では、は米国東部 (バージニア北部) リージョン (us-east-1) でのみ AWS OpsWorks スタックプロバイダー CodePipeline をサポートしています。このチュートリアルのすべてのリソースは、米国東部 (バージニア北部) リージョンで作成する必要があります。ではバージョニングされたソース CodePipeline が必要なため、バケットもバージョニングされている必要があります。詳細については、「[バージョニングの使用](#)」を参照してください。

アプリケーションを Amazon S3 バケットにアップロードするには

1. AWS OpsWorks スタックサンプルの ZIP ファイル、[Node.js アプリ](#) をダウンロードし、ローカルコンピュータ上の便利な場所に保存します。
2. で Amazon S3 コンソールを開きます <https://console.aws.amazon.com/s3/>。
3. バケットの作成 を選択します。
4. [Create a Bucket - Select a Bucket Name and Region] ページの [Bucket Name] で、バケットの一意の名前を入力します。バケット名は、自分の AWS アカウントだけでなく、すべてのアカウントで一意である必要があります。このワークスルーでは **my-appbucket** という名前を使用していますが、**my-appbucket-yearmonthday** を使用して一意のバケット名にすることができます。[Region] ドロップダウンリストで、[US Standard]、[Create] の順に選択します。[US Standard] は、us-east-1 と同等です。

Create a Bucket - Select a Bucket Name and Region

Cancel

A bucket is a container for objects stored in Amazon S3. When creating a bucket, you can choose a Region to optimize for latency, minimize costs, or address regulatory requirements. For more information regarding bucket naming conventions, please visit the [Amazon S3 documentation](#).

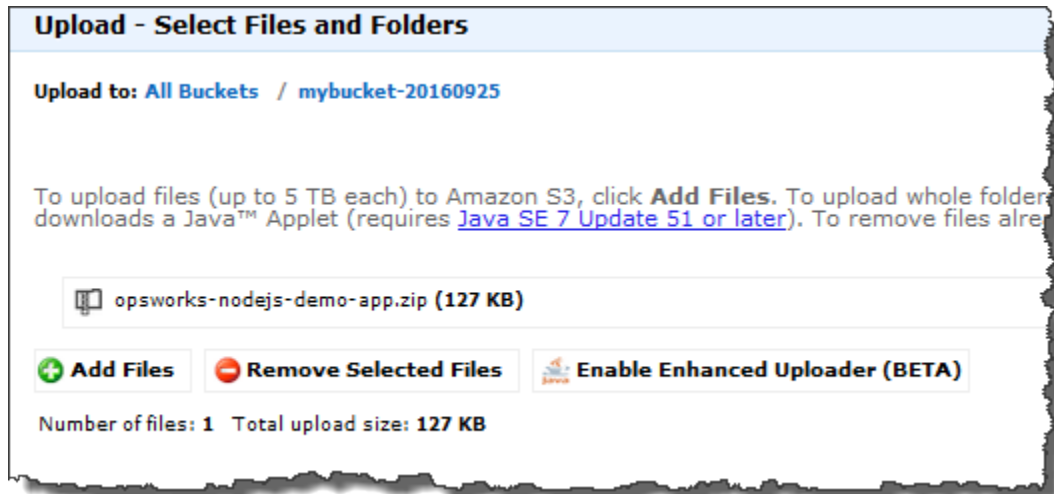
Bucket Name: Region:

Set Up Logging >

Create

Cancel

- 作成したバケットを [All Buckets] リストから選択します。
- バケットのページで [Upload] を選択します。
- [Upload - Select Files and Folders] ページで、[Add files] を選択します。ステップ 1 で保存した ZIP ファイルを参照し、「を開く」を選択し、「アップロードの開始」を選択します。



- アップロードが完了したら、バケット内の ZIP ファイルのリストからファイルを選択し、プロパティを選択します。
- プロパティペインで、ZIP ファイルへのリンクをコピーし、リンクを書き留めます。パイプラインを作成するには、このリンクのバケット名と ZIP ファイル名部分が必要です。

ステップ 4: AWS OpsWorks スタックにアプリケーションを追加する

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

でパイプラインを作成する前に CodePipeline、Node.js テストアプリケーションを AWS OpsWorks スタックに追加します。パイプラインを作成するときは、AWS OpsWorks スタックに追加したアプリを選択する必要があります。

前の手順のステップ 9 の Amazon S3 バケットリンクを準備してください。この手順を完了するには、テストアプリケーションを保存したバケットへのリンクが必要です。

AWS OpsWorks スタックにアプリケーションを追加するには

1. AWS OpsWorks スタックコンソールで を開き CodePipelineDemo、ナビゲーションペインでアプリ を選択します。
2. [Add app] を選択します。
3. [Add App] ページで、以下の情報を入力します。
 - a. アプリの名前を指定します。このウォークスルーでは、名前に Node.js Demo App を使用します。
 - b. [Data source type] で [None] を選択します。このアプリには、外部データベースやデータソースは必要ありません。
 - c. [Repository type] ドロップダウンリストで、[S3 Archive] を選択します。
 - d. リポジトリURL文字列ボックスに、 のステップ 9 でコピーURLした を貼り付けます [ステップ 3: アプリケーションコードを Amazon S3 バケットにアップロードする](#)。フォームは次のようになります。

Add App

All app attributes are stored in Chef data bags. [Learn more.](#)

Settings

Name	<input type="text" value="Node.js Demo App"/>
Document root	<input type="text" value="opsworks-nodejs-demo-app"/>

Data Sources

Data source type RDS None

Application Source

Repository type	<input type="text" value="S3 Archive"/>
Repository URL	<input type="text" value="I60925/opsworks-nodejs-demo-app.zip"/>
Access key ID	<input type="text" value="Optional"/>
Secret access key	<input type="text" value="Optional"/>

Environment Variables

<input type="text" value="KEY"/>	<input type="text" value="VALUE"/>	<input type="checkbox"/> Protected value
----------------------------------	------------------------------------	--

Add Domains

Domain name	<input type="text" value="Optional"/>	<input type="button" value="+"/>
-------------	---------------------------------------	----------------------------------

SSL Settings

Enable SSL	<input type="checkbox"/> No
------------	-----------------------------

[Cancel](#)[Add App](#)

- このフォームでは、他の設定を変更する必要はありません。[Add App] を選択します。
- [Apps] (アプリケーション) ページのリストに [Node.js Demo App] (Node.js デモアプリケーション) が表示されたら、次の手順 [ステップ 5: でパイプラインを作成する CodePipeline](#) に進みます。

ステップ 5: でパイプラインを作成する CodePipeline

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

レイヤーとスタックに少なくとも 1 つのインスタンスが設定された AWS OpsWorks スタックを作成したら、で AWS OpsWorks スタック CodePipeline をプロバイダーとしてパイプラインを作成し、アプリケーションまたは Chef クックブックを AWS OpsWorks スタックリソースにデプロイします。

パイプラインを作成するには

1. で CodePipeline コンソールを開きます <https://console.aws.amazon.com/codepipeline/>。
2. パイプラインの作成 を選択します。
3. 「開始 CodePipeline方法」ページで、**MyOpsWorksPipeline** 「」、またはアカウントに固有のその他のパイプライン名を入力し、次のステップ「」を選択します。
4. [Source Location] (送信元の場所) ページで、[Source provider] (送信元のプロバイダー) ドロップダウンリストからの [Amazon S3] を選択します。
5. [Amazon S3 details] (Amazon S3 詳細) 領域で、使用する Amazon S3 バケットパスを **s3://bucket-name/file name** という形式で入力します。 [ステップ 3: アプリケーションコードを Amazon S3 バケットにアップロードする](#) のステップ 9 で記したリンクを参照してください。このウォークスルーでは、パスは s3://my-appbucket/opsworks-nodejs-demo-app.zip になります。 [Next step] を選択します。

Source location ?

Specify where your source code is stored. Choose the provider, and then provide connection details for that provider.

Source provider*

Amazon S3

Amazon S3 details

Specify your Amazon S3 location, such as `s3://my-bucket/path/to/object.zip`.

Amazon S3 location*

`s3://my-appbucket/opsworks-nodejs-demo-app.zip`

* Required

Cancel

Previous

Next step

6. [Build] ページのドロップダウンリストで、[No Build]、[Next step] の順に選択します。
7. [デプロイ] ページで、デプロイプロバイダーとして [AWS OpsWorks Stacks] を選択します。

Deploy ?

Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

Deployment provider*

AWS OpsWorks Stacks i

Choose one of your existing stacks.

Stack* ↻

Choose the layer that your target instances belong to.

Layer ↻

Choose the app that you want to update and deploy, or [create a new one in AWS OpsWorks Stacks](#).

App* ↻

The application source that you specified for 'PHPTestApp' in AWS OpsWorks Stacks will use a new Amazon S3 archive, and the repository URL will point to the version of the artifact that you are deploying.
[Learn more](#)

* Required

Cancel

Previous

Next step

- [Stack] フィールドに、CodePipelineDemo または [ステップ 1: スタックにスタック、レイヤー、インスタンスを作成する AWS OpsWorks](#) で作成したスタックの名前を入力します。
- [Layer] (レイヤー) フィールドに、「Node.js App Server」と入力するか、「[ステップ 1: スタックにスタック、レイヤー、インスタンスを作成する AWS OpsWorks](#)」で作成したレイヤーの名前を入力します。

10. [App] (アプリケーション) フィールドで、[ステップ 3: アプリケーションコードを Amazon S3 バケットにアップロードする](#) で Amazon S3 にアップロードしたアプリケーションを選択してから、[Next step] (次のステップ) を選択します。
11. AWS サービスロール ページで、ロールの作成 を選択します。

新しいウィンドウが開き、コンソール IAM ページに、作成されるロール が表示されます AWS-CodePipeline-Service。[Policy name] ドロップダウンリストで、[Create new policy] を選択します。ポリシードキュメントに次のコンテンツがあることを確認します。[Edit] を選択し、必要に応じてポリシードキュメントを変更します。

```
{
  "Statement": [
    {
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketVersioning"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": "opsworks:*",
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

ポリシードキュメントへの変更が完了したら、[Allow] を選択します。変更が IAM コンソールに表示されます。

▼ Hide Details

Role Summary 

Role Description Provides read and write access to AWS services and resources.


IAM Role

Policy Name

▼ Hide Policy Document

[Edit](#)

```
{
  "Statement": [
    {
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketVersioning"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ],
}
```

 Note

ロールの作成が失敗した場合、AWS-CodePipeline-Service という名前のIAMロールが既にあることが原因である可能性があります。2016年5月より前に AWS-CodePipeline-Service ロールを使用している場合、そのロールにはデプロイプロバイダーとして AWS OpsWorks スタックを使用するアクセス許可がない可能性があります。この場合、このステップで見たようにポリシーステートメントを更新する必要があります。エラーメッセージが表示された場合は、この手順の最初に戻ってから、[Create role] (ロールの作成) の代わりに [Use existing role] (既存のロールを使用する) を選択します。既存のロールを使用する場合は、このステップに示される権限を含むポリシーがロールにアタッチされている必要があります。サービスロールとそのポリシーステートメントの詳細については、[IAM「サービスロールのポリシーの編集」](#)を参照してください。

12. ロールの作成プロセスが成功すると、IAMページが閉じ、AWS サービスロールページに戻ります。[Next step] を選択します。
13. [Review your pipeline] ページで、ページに表示される選択肢を確認した後、[Create pipeline] を選択します。

14. 準備ができたパイプラインは、自動的にソースコードを見つけ、アプリケーションをスタックにデプロイし始めます。この処理には数分かかることもあります。

ステップ 6: AWS OpsWorks スタックでアプリケーションのデプロイを確認する

Important

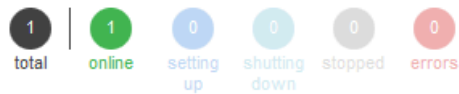
この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

が Node.js アプリをスタックに CodePipeline デプロイしたことを確認するには、 で作成したインスタンスにサインインします [ステップ 1: スタックにスタック、レイヤー、インスタンスを作成する AWS OpsWorks](#)。Node.js ウェブアプリケーションを表示して使用できる必要があります。

AWS OpsWorks スタックインスタンスでのアプリケーションのデプロイを確認するには

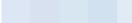
1. で AWS OpsWorks コンソールを開きます <https://console.aws.amazon.com/opsworks/>。
2. AWS OpsWorks スタックダッシュボードで、 を選択し CodePipelineDemo、Node.js アプリケーションサーバー を選択します。
3. ナビゲーションペインで [Instances] を選択した後、作成したインスタンスのパブリック IP アドレスを選択し、ウェブアプリケーションを表示します。

Instances

[Stop All Instances](#)

An instance represents a server. It can belong to one or more layers, that define the instance's settings, resources, installed packages, profiles and security groups. When you start the instance, OpsWorks uses the associated layer's blueprint to create and configure a corresponding EC2 instance. [Learn more.](#)

Node.js App Server

Hostname	Status	Size	Type	AZ	Public IP	Actions
nodejs-server1	online	c3.large	24/7	us-east-1a		stop ssh

[+ Instance](#)

アプリケーションが新しいブラウザタブに表示されます。



Congratulations!

You just deployed your first app with AWS OpsWorks.

!!! Deployed with CodePipeline !!!

 Tweet

 Follow @AWSOpsWorks



This app runs on app11 (Linux). Your request came from Mozilla/5.0
. The system time is 9/28/2016, 6:06:43 PM. Page rendered using Node.js version v4.1.1.

Leave a comment

Send

So cool!

9/28/2016, 12:40:20 AM

ステップ 7 (オプション): アプリコードを更新して、アプリを自動的に CodePipeline 再デプロイすることを確認する

Important

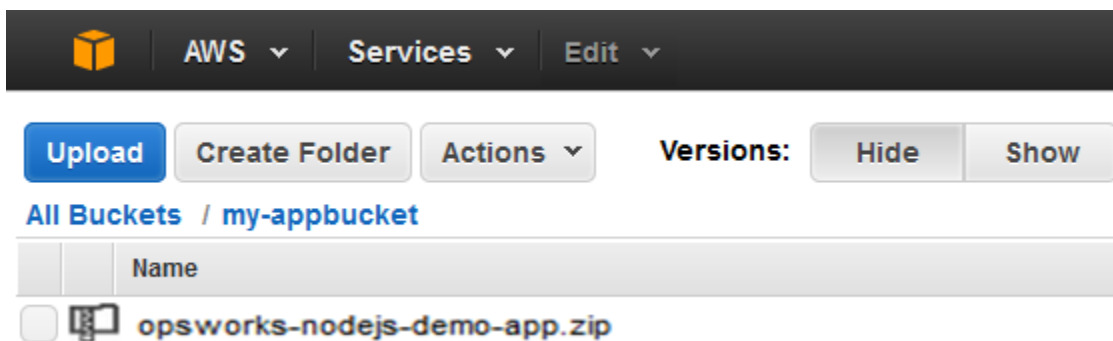
この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

を使用してデプロイしたアプリケーションまたはクックブックのコードを変更すると CodePipeline、更新されたアーティファクトは によって CodePipeline ターゲットインスタンス (この場合はターゲット AWS OpsWorks スタックスタック) に自動的にデプロイされます。このセクションでは、サンプル Node.js アプリケーションでコードを更新した際の自動再デプロイを示します。このウォークスルー用のアプリケーションコードがまだローカルに保存してあり、ウォークスルーを開始してから他の誰もコードを変更していない場合は、この手順のステップ 1~4 をスキップできます。

サンプルアプリケーションでコードを編集するには

1. にサインイン AWS Management Console し、 で Amazon S3 コンソールを開きます <https://console.aws.amazon.com/s3/>。
2. サンプル Node.js アプリケーションを保存しているバケットを開きます。



3. アプリを含むZIPファイルを選択します。[Actions] メニューで、[Download] を選択します。
4. ダイアログボックスでコンテキスト (右クリック) メニューを開き、ダウンロード を選択し、ZIPファイルを便利な場所に保存します。[OK] を選択します。
5. ZIP ファイルの内容を便利な場所に抽出します。展開したフォルダとそのサブフォルダやコンテンツで権限を変更し、編集を許可することが必要な場合があります。[opsworks-nodejs-demo-app\views] フォルダで、編集する [header.html] ファイルを開きます。
6. 「You just deployed your first app with」という語句で検索します。deployed という単語を updated に置き換えます。次の行で AWS OpsWorks. を AWS OpsWorks and AWS CodePipeline. に変更します。テキスト以外の部分を編集しないでください。

```
<div id="main" role="main">LF
...<div class="container">LF
...<div class="hero-unit">LF
...<div class="robot">LF
...<h1>Congratulations!</h1>LF
...<h2>LF
...<h2> You just updated your first app with<br/>LF
...AWS OpsWorks and AWS CodePipeline. LF
...</h2>LF
```

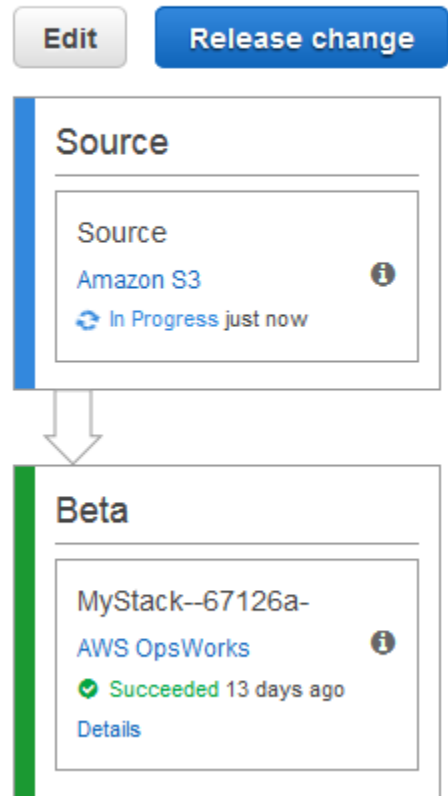
- header.html ファイルを保存して閉じます。
- opsworks-nodejs-demo-app フォルダを圧縮し、ファイルをZIP便利な場所に保存します。ZIP ファイルの名前を変更しないでください。
- 新しいZIPファイルを Amazon S3 バケットにアップロードします。このウォークスルーでは、バケット名は my-appbucket です。
- CodePipeline コンソールを開き、AWS OpsWorks スタックパイプライン () を開きますMyOpsWorksPipeline。[Release Change] を選択します。

(CodePipeline は、Amazon S3 バケット内のアプリの更新バージョンからのコード変更を検出するのを待つことができます。このチュートリアルでは、時間を節約するために、リリース変更を選択するように指示します)。

- パイプラインのステージで CodePipeline 実行されているを確認します。まず、はソースアーティファクトの変更 CodePipeline を検出します。

MyOpsWorksPipeline

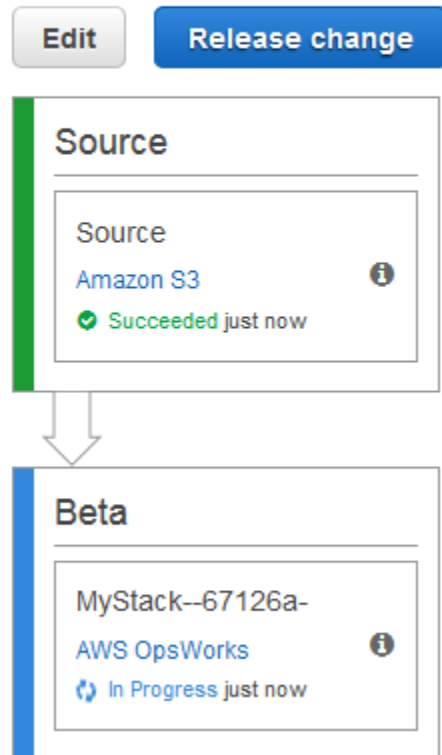
View progress and manage your pipeline.



CodePipeline は、更新されたコードを スタックの AWS OpsWorks スタックにプッシュします。

MyOpsWorksPipeline

View progress and manage your pipeline.



12. パイプラインの両ステージが正常に完了したら、AWS OpsWorks スタックでスタックを開きます。
13. スタックのプロパティページで、[Instances] を選択します。
14. [Public IP] 列で、インスタンスのパブリック IP アドレスを選択し、更新されたアプリケーションのテキストを表示します。



Congratulations!

You just updated your first app with AWS OpsWorks and AWS CodePipeline.

!!! Deployed with CodePipeline !!!

 Tweet

 Follow @AWSOpsWorks



This app runs on app11 (Linux). Your request came from Mozilla/5.0
. The system time is 9/28/2016, 6:06:43 PM. Page rendered using Node.js version v4.1.1.

Leave a comment

Send

So cool!
9/28/2016, 12:40:20 AM

ステップ 8 (オプション): リソースをクリーンアップする

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS アカウントへの不要な請求を防ぐため、このチュートリアルで使用した AWS リソースを削除できます。これらの AWS リソースには、AWS OpsWorks スタックスタック、IAMロールとインスタンスプロファイル、およびで作成したパイプラインが含まれます CodePipeline。ただし、AWS OpsWorks スタックと の詳細については、これらの AWS リソースを引き続き使用することをお勧めします CodePipeline。これらのリソースを保持するには、このウォークスルーを完了する必要があります。

アプリケーションをスタックから削除するには

AWS CloudFormation テンプレートの一部としてアプリケーションを作成または適用しなかったため、スタックを削除する前に Node.js テストアプリケーションを削除してください AWS CloudFormation。

1. AWS OpsWorks スタックコンソールのサービスナビゲーションペインで、アプリ を選択します。
2. [Apps] ページで [Node.js Demo App] を選択した後、[Actions] で [delete] を選択します。確認のプロンプトが表示されたら、「 の削除」を選択します。AWS OpsWorks スタックはアプリケーションを削除します。

スタックを削除するには

AWS CloudFormation テンプレートを実行してスタックを作成したため、テンプレートが作成したレイヤー、インスタンス、インスタンスプロファイル、セキュリティグループなどのスタックを AWS CloudFormation コンソールで削除できます。

1. AWS CloudFormation コンソールを開きます。
2. AWS CloudFormation コンソールダッシュボードで、作成したスタックを選択します。[Actions] メニューで、[Delete Stack] を選択します。確認を求められたら [Yes, Delete] を選択します。
3. スタックのステータス列に DELETE_COMPLETE が表示されるまで待ちます。

パイプラインを削除するには

1. CodePipeline コンソールを開きます。

2. CodePipeline ダッシュボードで、このチュートリアル用に作成したパイプラインを選択します。
3. パイプラインページで、[Edit] を選択します。
4. [Edit] ページで、[Delete] を選択します。確認を求められたら [Delete] (削除) を選択します。

AWS CodePipelineAWS OpsWorks スタックあり - Chef 11 スタック

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行に関するご質問は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[AWS CodePipeline](#) では、Amazon Simple Storage Service (Amazon S3) CodeCommit、などのソースからのコード変更を追跡する継続的デリバリーパイプラインを作成できます[GitHub](#)。このトピックの例では、AWS OpsWorks スタックレイヤーで実行するコードのデプロイツール CodePipeline として、からシンプルなパイプラインを作成して使用方法について説明します。この例では、[シンプルなPHPアプリケーションのパイプラインを作成し](#)、Chef 11.10 AWS OpsWorks スタック (この場合は 1 つのインスタンス) のレイヤー内のすべてのインスタンスでアプリケーションを実行するように スタックに指示します。

Note

このトピックでは、パイプラインを使用して Chef 11.10 スタックでアプリケーションを実行および更新する方法について説明します。パイプラインを使用して Chef 12 スタックでアプリケーションを実行および更新する方法については、「[AWS CodePipelineAWS OpsWorks スタックあり - Chef 12 スタック](#)」を参照してください。Amazon S3 バケットに配信されるコンテンツには、カスタマーコンテンツが含まれている場合があります。機密データの削除の詳細については、「[S3 バケットを空にする方法](#)」または「[S3 バケットを削除する方法](#)」を参照してください。

トピック

- [前提条件](#)

- [サポートされている他のシナリオ](#)
- [ステップ 1: AWS OpsWorks スタックでスタック、レイヤー、インスタンスを作成する](#)
- [ステップ 2: アプリケーションコードを Amazon S3 バケットにアップロードする](#)
- [ステップ 3: AWS OpsWorks スタックにアプリケーションを追加する](#)
- [ステップ 4: でパイプラインを作成する CodePipeline](#)
- [ステップ 5: AWS OpsWorks スタックでのアプリケーションのデプロイを検証する](#)
- [ステップ 6 \(オプション\): アプリコードを更新して、アプリを自動的に再デプロイすることを確認する CodePipeline](#)
- [ステップ 7 \(オプション\): リソースをクリーンアップする](#)

前提条件

このウォークスルーを開始する前に、以下のすべてのタスクを実行するための管理者権限があることを確認してください。AdministratorAccess ポリシーが適用されたグループのメンバーにすることも、次の表に示すアクセス許可とポリシーを持つグループのメンバーにすることもできます。セキュリティのベストプラクティスとして、必要な権限を個別のユーザーアカウントに割り当てるのではなく、次のタスクを実行する権限を持つグループにユーザーが属している必要があります。

でセキュリティグループを作成しIAM、グループにアクセス許可を割り当てる方法の詳細については、[IAM「ユーザーグループの作成」](#)を参照してください。AWS OpsWorks スタックのアクセス許可の管理の詳細については、[「ベストプラクティス: アクセス許可の管理」](#)を参照してください。

アクセス許可	グループにアタッチすることが推奨されるポリシー
スタックでスタック、レイヤー、インスタンスを作成および編集 AWS OpsWorks します。	AWSOpsWorks_FullAccess
AWS CloudFormationでテンプレートを作成、編集、実行します。	AmazonCloudFormationFullAccess
Amazon S3 バケットの作成、編集、アクセスを行います。	AmazonS3FullAccess

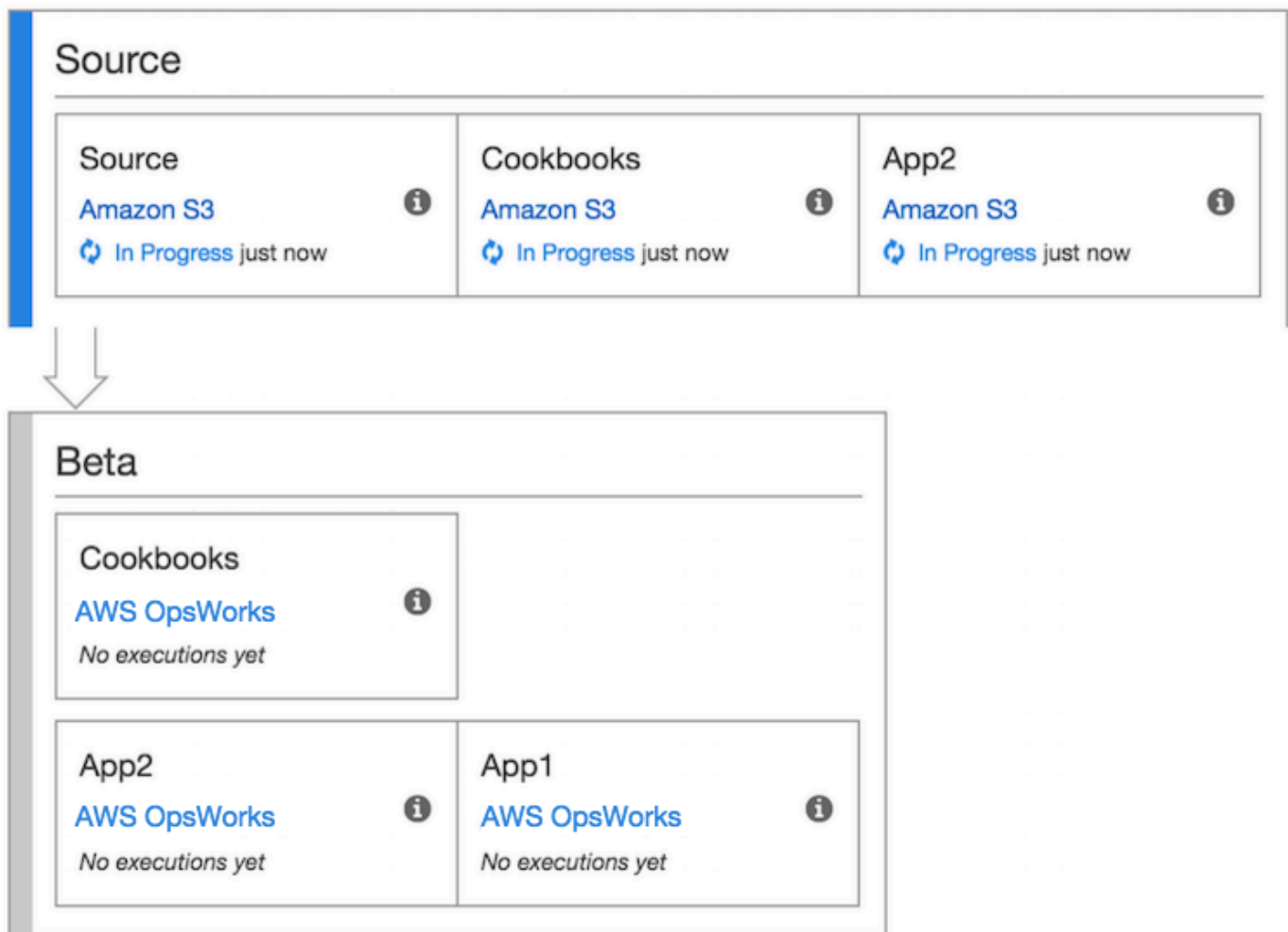
アクセス許可	グループにアタッチすることが推奨されるポリシー
でパイプライン、特に AWS OpsWorks スタックをプロバイダーとして使用するパイプラインを作成 CodePipeline、編集、実行します。	AWSCodePipeline_FullAccess

Amazon EC2キーペアも必要です。このチュートリアルでサンプルスタック、レイヤー、インスタンスを作成する AWS CloudFormation テンプレートを実行するとき、このキーペアの名前を指定するように求められます。Amazon EC2コンソールでキーペアを取得する方法の詳細については、Amazon [ドキュメントの「キーペアの作成」](#)を参照してください。EC2キーペアは、米国東部 (バージニア北部) リージョンにある必要があります。そのリージョンに既存のキーペアがある場合は、そのキーペアを使用できます。

サポートされている他のシナリオ

このウォークスルーでは、[Source] を 1 つ、[Deploy] ステージを 1 つ含むシンプルなパイプラインを作成します。ただし、AWS OpsWorks スタックをプロバイダーとして使用する、より複雑なパイプラインを作成できます。サポートされているパイプラインとシナリオの例を以下に示します。

- パイプラインを編集し、Chef クックブックを [Source] ステージに、更新されたクックブックの関連ターゲットを [Deploy] に追加できます。この場合は、ソースを変更した際にクックブックの更新をトリガーする [デプロイ] アクションを追加します。更新されたクックブックは、アプリの前にデプロイされます。
- カスタムクックブックと複数のアプリケーションを使用して複雑なパイプラインを作成し、AWS OpsWorks スタックスタックにデプロイできます。パイプラインはアプリケーションソースとクックブックソースの両方の変化を追跡し、変更を行った際に再デプロイを行います。同様の複雑なパイプラインの例を以下に示します。



の操作の詳細については CodePipeline、[CodePipelineドキュメント](#)を参照してください。

ステップ 1: AWS OpsWorks スタックでスタック、レイヤー、インスタンスを作成する

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックをパイプラインのデプロイプロバイダーとして使用するには、まずスタック、レイヤー、およびレイヤーに少なくとも1つのインスタンスが必要です。AWS OpsWorks [「Linux スタックの開始方法」](#)または[「Windows スタックの開始方法」](#)の手順に従ってスタックを作成できますが、時間を節約するために、この例では AWS CloudFormation テンプレートを使用して Linux ベースの Chef 11.10 スタック、レイヤー、インスタンスを作成します。このテンプレートで作成されたインスタンスは、Amazon Linux 2016.03 を実行します。インスタンスタイプは c3.large です。

Important

AWS CloudFormation テンプレートは、後でアプリケーションをアップロードする Amazon S3 バケットと同じリージョンと、後でパイプラインを作成するリージョンに保存して実行する必要があります CodePipeline。現時点では、は米国東部 (バージニア北部) リージョン (us-east-1) でのみ AWS OpsWorks スタックプロバイダー CodePipeline をサポートしています。このチュートリアルすべてのリソースは、米国東部 (バージニア北部) リージョンで作成する必要があります。

スタックの作成に失敗した場合、アカウントで許可されるIAMロールの最大数に近づいている可能性があります。またスタックの作成は、アカウントがインスタンスを c3.large インスタンスタイプで起動できないときに、失敗する場合があります。例えば、AWS 無料利用枠を使用している場合に、Root device type: must be included in EBSなどのエラーが発生することがあります。AWS 無料利用枠によって課される制限など、アカウントで作成が許可されているインスタンスタイプに制限がある場合は、テンプレートのインスタンスブロックの InstanceTypeパラメータの値を、アカウントが使用できるインスタンスタイプに変更してみてください。

を使用してスタック、レイヤー、インスタンスを作成するには AWS CloudFormation

1. 次の AWS CloudFormation テンプレートを新しいプレーンテキストドキュメントにコピーします。ローカルコンピュータ上の便利な場所にファイルを保存し、NewOpsWorksStack.template という名前、または便利な別の名前を付けます。

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Mappings": {
    "Region2Principal": {
      "us-east-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      }
    }
  }
}
```

```
    },
    "us-west-2": {
      "EC2Principal": "ec2.amazonaws.com",
      "OpsWorksPrincipal": "opsworks.amazonaws.com"
    },
    "us-west-1": {
      "EC2Principal": "ec2.amazonaws.com",
      "OpsWorksPrincipal": "opsworks.amazonaws.com"
    },
    "eu-west-1": {
      "EC2Principal": "ec2.amazonaws.com",
      "OpsWorksPrincipal": "opsworks.amazonaws.com"
    },
    "ap-southeast-1": {
      "EC2Principal": "ec2.amazonaws.com",
      "OpsWorksPrincipal": "opsworks.amazonaws.com"
    },
    "ap-northeast-1": {
      "EC2Principal": "ec2.amazonaws.com",
      "OpsWorksPrincipal": "opsworks.amazonaws.com"
    },
    "ap-northeast-2": {
      "EC2Principal": "ec2.amazonaws.com",
      "OpsWorksPrincipal": "opsworks.amazonaws.com"
    },
    "ap-southeast-2": {
      "EC2Principal": "ec2.amazonaws.com",
      "OpsWorksPrincipal": "opsworks.amazonaws.com"
    },
    "sa-east-1": {
      "EC2Principal": "ec2.amazonaws.com",
      "OpsWorksPrincipal": "opsworks.amazonaws.com"
    },
    "cn-north-1": {
      "EC2Principal": "ec2.amazonaws.com.cn",
      "OpsWorksPrincipal": "opsworks.amazonaws.com.cn"
    },
    "eu-central-1": {
      "EC2Principal": "ec2.amazonaws.com",
      "OpsWorksPrincipal": "opsworks.amazonaws.com"
    }
  }
},
"Parameters": {
```

```
"EC2KeyName": {
  "Type": "String",
  "Description": "The name of an existing EC2 key pair that allows you to use SSH
to connect to the OpsWorks instance."
},
"Resources": {
  "CP0psDeploySecGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
      "GroupDescription": "Lets you manage OpsWorks instances deployed to by
CodePipeline"
    }
  },
  "CP0psDeploySecGroupIngressHTTP": {
    "Type": "AWS::EC2::SecurityGroupIngress",
    "Properties": {
      "IpProtocol": "tcp",
      "FromPort": "80",
      "ToPort": "80",
      "CidrIp": "0.0.0.0/0",
      "GroupId": {
        "Fn::GetAtt": [
          "CP0psDeploySecGroup", "GroupId"
        ]
      }
    }
  },
  "CP0psDeploySecGroupIngressSSH": {
    "Type": "AWS::EC2::SecurityGroupIngress",
    "Properties": {
      "IpProtocol": "tcp",
      "FromPort": "22",
      "ToPort": "22",
      "CidrIp": "0.0.0.0/0",
      "GroupId": {
        "Fn::GetAtt": [
          "CP0psDeploySecGroup", "GroupId"
        ]
      }
    }
  },
  "MyStack": {
    "Type": "AWS::OpsWorks::Stack",
```

```
"Properties": {
  "Name": {
    "Ref": "AWS::StackName"
  },
  "ServiceRoleArn": {
    "Fn::GetAtt": [
      "OpsWorksServiceRole",
      "Arn"
    ]
  },
  "ConfigurationManager" : { "Name": "Chef", "Version": "11.10" },
  "DefaultOs": "Amazon Linux 2016.03",
  "DefaultInstanceProfileArn": {
    "Fn::GetAtt": [
      "OpsWorksInstanceProfile",
      "Arn"
    ]
  }
},
"Shortname": "mylayer",
"Type": "php-app",
"EnableAutoHealing": "true",
"AutoAssignElasticIps": "false",
"AutoAssignPublicIps": "true",
"CustomSecurityGroupIds": [
  {
    "Fn::GetAtt": [
      "CP0psDeploySecGroup", "GroupId"
    ]
  }
],
"DependsOn": [
  "MyStack",
  "CP0psDeploySecGroup"
]
```

```
},
"OpsWorksServiceRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": [
              {
                "Fn::FindInMap": [
                  "Region2Principal",
                  {
                    "Ref": "AWS::Region"
                  },
                ],
                "OpsWorksPrincipal"
              }
            ]
          },
          "Action": [
            "sts:AssumeRole"
          ]
        }
      ]
    },
    "Path": "/",
    "Policies": [
      {
        "PolicyName": "opsworks-service",
        "PolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Action": [
                "ec2:*",
                "iam:PassRole",
                "cloudwatch:GetMetricStatistics",
                "elasticloadbalancing:*"
              ],
              "Resource": "*"
            }
          ]
        }
      ]
    ]
  }
}
```

```
    }
  }
]
},
"OpsWorksInstanceProfile": {
  "Type": "AWS::IAM::InstanceProfile",
  "Properties": {
    "Path": "/",
    "Roles": [
      {
        "Ref": "OpsWorksInstanceRole"
      }
    ]
  }
},
"OpsWorksInstanceRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": [
              {
                "Fn::FindInMap": [
                  "Region2Principal",
                  {
                    "Ref": "AWS::Region"
                  },
                ],
                "EC2Principal"
              }
            ]
          }
        }
      ]
    },
    "Action": [
      "sts:AssumeRole"
    ]
  }
},
  "Path": "/",
"Policies": [
```

```
{
  "PolicyName": "s3-get",
  "PolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "s3:GetObject"
        ],
        "Resource": "*"
      }
    ]
  }
},
"myinstance": {
  "Type": "AWS::OpsWorks::Instance",
  "Properties": {
    "LayerIds": [
      {
        "Ref": "MyLayer"
      }
    ],
    "StackId": {
      "Ref": "MyStack"
    },
    "InstanceType": "c3.large",
    "SshKeyName": {
      "Ref": "EC2KeyPairName"
    }
  }
},
"Outputs": {
  "StackId": {
    "Description": "Stack ID for the newly created AWS OpsWorks stack",
    "Value": {
      "Ref": "MyStack"
    }
  }
}
```

```
}
```

- にサインイン AWS Management Console し、<https://console.aws.amazon.com/cloudformation> で AWS CloudFormation コンソールを開きます。
- AWS CloudFormation ホームページで、スタックの作成 を選択します。
- [Select Template] ページの [Choose a template] エリアで、[Upload a template to Amazon S3]、[Browse] の順に選択します。
- ステップ 1 で保存した AWS CloudFormation テンプレートを参照し、「 を開く 」を選択します。 [Select Template] ページで、[Next] を選択します。

Select Template

Select the template that describes the stack that you want to create. A stack is a group of related resources that you manage as a single unit.

Design a template Use AWS CloudFormation Designer to create or modify an existing template. [Learn more.](#)

Design template

Choose a template A template is a JSON-formatted text file that describes your stack's resources and their properties. [Learn more.](#)

Select a sample template

Upload a template to Amazon S3

NewOpsWorksStack.template

Specify an Amazon S3 template URL

Cancel

Next

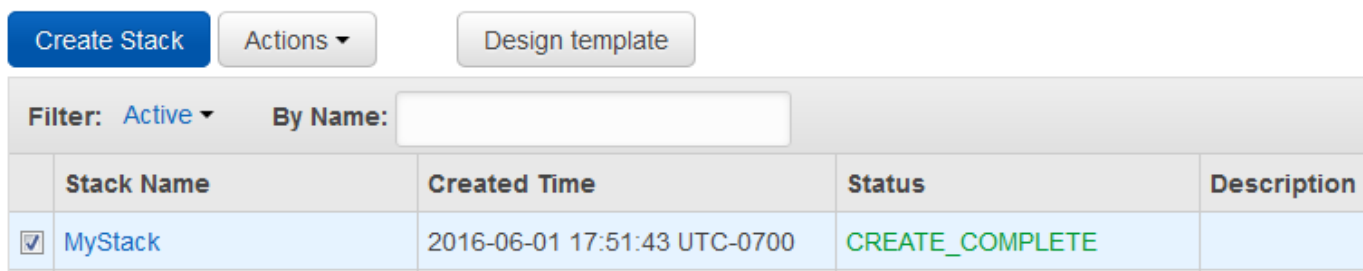
- 詳細の指定ページで、スタックに という名前を付けるかMyStack、アカウントに固有のスタック名を指定します。スタックに別の名前を付ける場合は、このウォークスルー中のスタック名スループットを変更してください。
- パラメータ領域で、作成後に AWS OpsWorks スタックインスタンスへのアクセスに使用する EC2キーペアの名前を指定します。[Next (次へ)] を選択します。
- [Options(オプション)] ページで、[Next(次へ)] を選択します。(このページの設定はこのウォークスルーでは必要ありません)
- このチュートリアルで使用する AWS CloudFormation テンプレートは、IAMロール、インスタンスプロファイル、インスタンスを作成します。

⚠ Important

の作成 を選択する前に、このテンプレートでリソースを作成するために発生する可能性のある料金を見積 AWS もるコスト を選択します。

IAM リソースの作成が許容できる場合は、このテンプレートによってIAMリソースAWS CloudFormation の作成が発生する可能性があることを確認するチェックボックスを選択し、 の作成を選択します。IAM リソースの作成が受け入れられない場合は、この手順を続行できません。

10. AWS CloudFormation ダッシュボードでは、スタックの作成の進行状況を表示できます。次のステップに進む前に、ステータス列に CREATE_COMPLETE が表示されるまで待ちます。

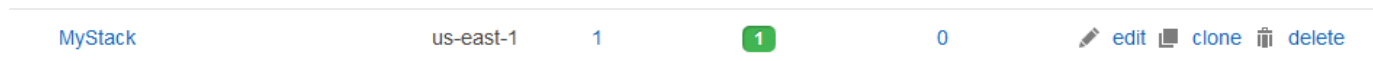


The screenshot shows the AWS CloudFormation console interface. At the top, there are buttons for 'Create Stack', 'Actions', and 'Design template'. Below these is a filter section with 'Filter: Active' and a search box 'By Name:'. The main part of the image is a table with the following columns: 'Stack Name', 'Created Time', 'Status', and 'Description'. There is one row in the table with the following data: 'MyStack', '2016-06-01 17:51:43 UTC-0700', 'CREATE_COMPLETE', and an empty description. A checkbox is checked next to the stack name.




	Stack Name	Created Time	Status	Description
<input checked="" type="checkbox"/>	MyStack	2016-06-01 17:51:43 UTC-0700	CREATE_COMPLETE	

スタックでの AWS OpsWorks スタックの作成を確認するには

1. で AWS OpsWorks コンソールを開きます <https://console.aws.amazon.com/opsworks/>。
2. AWS OpsWorks スタックダッシュボードで、作成したスタックを表示します。



The screenshot shows a row in the AWS OpsWorks console. The row contains the following information: 'MyStack', 'us-east-1', '1', '0', and a set of action icons (edit, clone, delete).

MyStack	us-east-1	1	0	 edit  clone  delete
---------	-----------	---	---	---

3. スタックを開き、レイヤーとインスタンスを表示します。レイヤーとインスタンスが、AWS CloudFormation テンプレートで指定された名前やその他のメタデータで作成されたことを確認します。これでアプリケーションを Amazon S3 バケットにアップロードできます。

ステップ 2: アプリケーションコードを Amazon S3 バケットにアップロードする

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソ

リユーシオンに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

パイプラインの設定の一部としてコードリポジトリへのリンクを指定する必要があるため、パイプラインを作成する前にコードリポジトリを準備してください。このチュートリアルでは、Amazon S3 バケットにPHPアプリケーションをアップロードします。

CodePipeline はソースから直接、GitHub またはソース CodeCommit としてコードを使用できますが、このチュートリアルでは Amazon S3 バケットの使用法を示します。Amazon S3 バケットを使用すると CodePipeline はアプリケーションコードの変更を検出し、変更されたアプリケーションを自動的にデプロイできます。必要に応じて、既存のバケットを使用できます。CodePipeline ドキュメントの Simple Pipeline チュートリアル (Amazon S3 バケット) で説明 CodePipeline されているように、バケットが の基準を満たしていることを確認してください。 [Amazon S3](#)

Important

Amazon S3 バケットは、後でパイプラインを作成するリージョンと同じリージョンに存在する必要があります。現時点では、は米国東部 (バージニア北部) リージョン (us-east-1) でのみ AWS OpsWorks スタックプロバイダー CodePipeline をサポートしています。このチュートリアルのすべてのリソースは、米国東部 (バージニア北部) リージョンで作成する必要があります。ではバージョニングされたソース CodePipeline が必要なため、バケットもバージョニングされている必要があります。詳細については、「[バージョニングの使用](#)」を参照してください。

アプリケーションを Amazon S3 バケットにアップロードするには

1. [GitHub ウェブサイト](#) から、AWS OpsWorks スタックサンプルPHPアプリケーションのZIP ファイルをダウンロードし、ローカルコンピュータ上の便利な場所に保存します。
2. index.php フォルダと ASSETS フォルダがダウンロードしたZIPファイルのルートレベルであることを確認します。そうでない場合は、ファイルを解凍し、これらのZIPファイルをルートレベルで持つ新しいファイルを作成します。
3. で Amazon S3 コンソールを開きます <https://console.aws.amazon.com/s3/>。
4. バケットの作成 を選択します。

5. [Create a Bucket - Select a Bucket Name and Region] ページの [Bucket Name] で、バケットの一意の名前を入力します。バケット名は、自分の AWS アカウントだけでなく、すべてのアカウントで一意である必要があります。このウォークスルーでは **my-appbucket** という名前を使用していますが、**my-appbucket-*yearmonthday*** を使用して一意のバケット名にすることができます。[Region] ドロップダウンリストで、[US Standard]、[Create] の順に選択します。[US Standard] は、us-east-1 と同等です。

Create a Bucket - Select a Bucket Name and Region

Cancel

A bucket is a container for objects stored in Amazon S3. When creating a bucket, you can choose a Region to optimize for latency, minimize costs, or address regulatory requirements. For more information regarding bucket naming conventions, please visit the [Amazon S3 documentation](#).

Bucket Name:

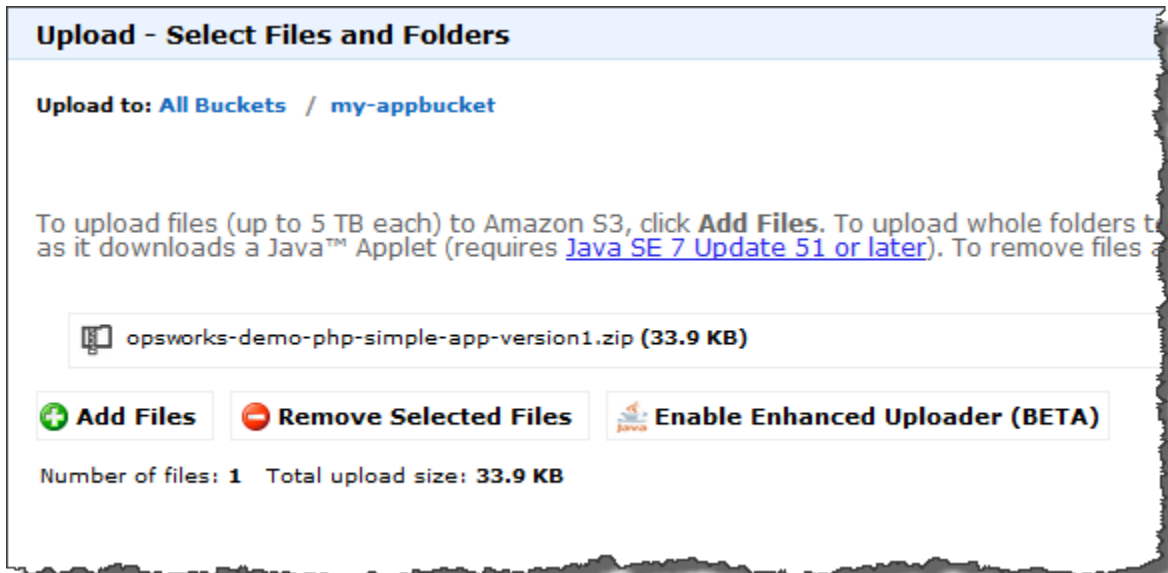
Region:

Set Up Logging >

Create

Cancel

6. 作成したバケットを [All Buckets] リストから選択します。
7. バケットのページで [Upload] を選択します。
8. [Upload - Select Files and Folders] ページで、[Add files] を選択します。ステップ 1 で保存した ZIP ファイルを参照し、「を開く」を選択し、「アップロードの開始」を選択します。



- アップロードが完了したら、バケット内のZIPファイルのリストからファイルを選択し、プロパティを選択します。
- プロパティペインで、ZIPファイルへのリンクをコピーし、リンクを書き留めます。パイプラインを作成するには、このリンクのバケット名とZIPファイル名部分が必要です。

ステップ 3: AWS OpsWorks スタックにアプリケーションを追加する

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

でパイプラインを作成する前に CodePipeline、PHPテストアプリケーションを AWS OpsWorks スタックに追加します。パイプラインを作成するときは、AWS OpsWorks スタックに追加したアプリを選択する必要があります。

前の手順のステップ 10 の Amazon S3 バケットリンクを準備してください。この手順を完了するには、テストアプリケーションを保存したバケットへのリンクが必要です。

AWS OpsWorks スタックにアプリケーションを追加するには

1. AWS OpsWorks スタックコンソールで [を開き](#) MyStack、ナビゲーションペインでアプリ [を選択](#) します。
2. [Add app] [を選択](#) します。
3. [Add App] ページで、以下の情報を入力します。
 - a. アプリの名前を指定します。このウォークスルーでは、名前に PHPTestApp を使用しま
す。
 - b. Type ドロップダウンリストで、[を選択](#) します PHP。
 - c. [Data source type] で [None] [を選択](#) します。このアプリには、外部データベースやデータ
ソースは必要ありません。
 - d. [Repository type] ドロップダウンリストで、[S3 Archive] [を選択](#) します。
 - e. リポジトリ文字列ボックスに、[のステップ URL10 でコピーURLした](#) [を貼り付けます](#) [ス
テップ 2 : アプリケーションコードを Amazon S3 バケットにアップロードする](#)。フォーム
は次のようになります。

Add App

Settings

Name	<input type="text" value="PHPTestApp"/>
Type	<input type="text" value="PHP"/>
Document root	<input type="text" value="Optional"/>

Data Sources

Data source type RDS OpsWorks None

Application Source

Repository type	<input type="text" value="S3 Archive"/>
Repository URL	<input type="text" value="'ks-demo-php-simple-app-version1.zip'"/>
Access key ID	<input type="text" value="Optional"/>
Secret access key	<input type="text" value="Optional"/>

Environment Variables

<input type="text" value="KEY"/>	<input type="text" value="VALUE"/>	<input type="checkbox"/> Protected value
----------------------------------	------------------------------------	--

Add Domains

Domain name	<input type="text" value="Optional"/>	<input type="button" value="+"/>
-------------	---------------------------------------	----------------------------------

SSL Settings

Enable SSL	<input type="checkbox"/> No
------------	-----------------------------

[Cancel](#)[Add App](#)

- このフォームでは、他の設定を変更する必要はありません。[Add App] を選択します。
- PHPTestApp アプリケーションページのリストにアプリケーションが表示されたら、次の手順に進みます [ステップ 4: でパイプラインを作成する CodePipeline](#)。

ステップ 4: でパイプラインを作成する CodePipeline

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

レイヤーとスタックで設定されたインスタンスが少なくとも 1 つある AWS OpsWorks スタックを作成したら、で AWS OpsWorks スタック CodePipeline をプロバイダーとしてパイプラインを作成し、アプリケーションまたは Chef クックブックを AWS OpsWorks スタックリソースにデプロイします。

パイプラインを作成するには

1. で CodePipeline コンソールを開きます <https://console.aws.amazon.com/codepipeline/>。
2. パイプラインの作成 を選択します。
3. 「開始 CodePipeline方法」ページで **MyOpsWorksPipeline**、**「**」、またはアカウントに固有のその他のパイプライン名を入力し、次のステップ **「**」を選択します。
4. [Source Location] (送信元の場所) ページで、[Source provider] (送信元のプロバイダー) ドロップダウンリストからの [Amazon S3] を選択します。
5. [Amazon S3 details] (Amazon S3 詳細) 領域で、使用する Amazon S3 バケットパスを **s3://bucket-name/file name** という形式で入力します。ステップ 10 ([ステップ 2: アプリケーションコードを Amazon S3 バケットにアップロードする](#))。このワークスルーでは、パスは s3://my-appbucket/opsworks-demo-php-simple-app-version1.zip になります。
[Next step] を選択します。

Source location ?

Specify where your source code is stored. Choose the provider, and then provide connection details for that provider.

Source provider*

Amazon S3

Amazon S3 details

Specify your Amazon S3 location, such as `s3://my-bucket/path/to/object.zip`.

Amazon S3 location*

`s3://my-appbucket/opsworks-windows-demo-nodejs-master.zip`

* Required

Cancel

Previous

Next step

- [Build] ページのドロップダウンリストで、[No Build]、[Next step] の順に選択します。
- [デプロイ] ページで、デプロイプロバイダーとして [AWS OpsWorks Stacks] を選択します。

Deploy ?

Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

Deployment provider*

AWS OpsWorks Stacks i

Choose one of your existing stacks.

Stack* ↻

Choose the layer that your target instances belong to.

Layer ↻

Choose the app that you want to update and deploy, or [create a new one in AWS OpsWorks Stacks](#).

App* ↻

The application source that you specified for 'PHPTestApp' in AWS OpsWorks Stacks will use a new Amazon S3 archive, and the repository URL will point to the version of the artifact that you are deploying.
[Learn more](#)

* Required

Cancel

Previous

Next step

- [Stack] フィールドに、MyStack または [ステップ 1: AWS OpsWorks スタックでスタック、レイヤー、インスタンスを作成する](#) で作成したスタックの名前を入力します。
- [Layer] (レイヤー) フィールドに、「MyLayer」と入力するか、「[ステップ 1: AWS OpsWorks スタックでスタック、レイヤー、インスタンスを作成する](#)」で作成したレイヤーの名前を入力します。

10. [App] (アプリケーション) フィールドで、[ステップ 2: アプリケーションコードを Amazon S3 バケットにアップロードする](#) で Amazon S3 にアップロードしたアプリケーションを選択してから、[Next step] (次のステップ) を選択します。
11. AWS サービスロールページで、ロールの作成 を選択します。

新しいウィンドウが開き、コンソールIAMページが開き、作成されるロール が示されますAWS-CodePipeline-Service。[Policy name] ドロップダウンリストで、[Create new policy] を選択します。ポリシードキュメントに次のコンテンツがあることを確認します。[Edit] を選択し、必要に応じてポリシードキュメントを変更します。

```
{
  "Statement": [
    {
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketVersioning"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": "opsworks:*",
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

ポリシードキュメントへの変更が完了したら、[Allow] を選択します。変更は IAMコンソールに表示されます。

▼ Hide Details

Role Summary 

Role Description Provides read and write access to AWS services and resources.


IAM Role

Policy Name

▼ Hide Policy Document

[Edit](#)

```
{
  "Statement": [
    {
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketVersioning"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ],
}
```

 Note

ロールの作成に失敗した場合、AWS-CodePipeline-Service という名前のIAMロールが既にあることが原因である可能性があります。2016年5月より前に AWS-CodePipeline-Service ロールを使用していた場合、そのロールにはデプロイプロバイダーとして AWS OpsWorks スタックを使用するアクセス許可がない可能性があります。この場合、このステップに示すようにポリシーステートメントを更新する必要があります。エラーメッセージが表示された場合は、この手順の最初に戻ってから、[Create role] (ロールの作成) の代わりに [Use existing role] (既存のロールを使用する) を選択します。既存のロールを使用する場合は、このステップに示される権限を含むポリシーがロールにアタッチされている必要があります。サービスロールとそのポリシーステートメントの詳細については、[IAM「サービスロールのポリシーの編集」](#)を参照してください。

12. ロールの作成プロセスが成功すると、IAMページが閉じ、AWSサービスロールページに戻ります。 [Next step] を選択します。
13. [Review your pipeline] ページで、ページに表示される選択肢を確認した後、[Create pipeline] を選択します。

We will create your pipeline with the following resources.

Source Stage

Source provider Amazon S3

Amazon S3 location s3://my-appbucket0/opsworks-demo-php-simple-app-version1.zip

Build Stage

Build provider No Build

Beta Stage

Deployment provider AWS OpsWorks

Stack MyStack

App PHPTestApp

Layer MyLayer

Pipeline settings

Pipeline name MyOpsWorksPipeline

Artifact location s3://codepipeline-us-east-
AWS CodePipeline will use this existing S3 bucket to store artifacts for this pipeline. Depending on the size of your artifacts, you might be charged for storage costs. For more information, see [Amazon S3 storage pricing](#).

Role name AWS-CodePipeline-Service

To save this configuration with these resources, choose Create pipeline.

Would you like to create this pipeline?

[Cancel](#)

[Previous](#)

[Create pipeline](#)

- 準備ができたパイプラインは、自動的にソースコードを見つけ、アプリケーションをスタックにデプロイし始めます。この処理には数分かかることもあります。

ステップ 5: AWS OpsWorks スタックでのアプリケーションのデプロイを検証する

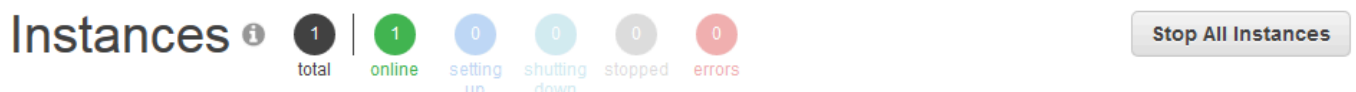
Important








この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

がアプリケーションをスタックPHPに CodePipeline デプロイしたことを確認するには、で作成したインスタンスにサインインします [ステップ 1: AWS OpsWorks スタックでスタック、レイヤー、インスタンスを作成する](#)。PHP ウェブアプリを表示して使用できるはずです。

AWS OpsWorks スタックインスタンスでのアプリケーションのデプロイを確認するには

- で AWS OpsWorks コンソールを開きます <https://console.aws.amazon.com/opsworks/>。
- AWS OpsWorks スタックダッシュボードで、を選択しMyStack、を選択しますMyLayer。
- ナビゲーションペインで [Instances] を選択した後、作成したインスタンスのパブリック IP アドレスを選択し、ウェブアプリケーションを表示します。



Instances   |      Stop All Instances

MyLayer

Search for instances in this layer by name, status, size, type, AZ or IP						
Hostname	Status	Size	Type	AZ	Public IP	Actions
php-app1	online	c3.large	24/7	us-east-1a	54.242.188.34	stop ssh

アプリケーションが新しいブラウザタブに表示されます。

Simple PHP App

Congratulations!

Your PHP application is now running on the host "php-app1" in your own dedicated environment in the AWS Cloud.

This host is running PHP version 5.3.29.

ステップ 6 (オプション): アプリコードを更新して、アプリを自動的に再デプロイすることを確認する CodePipeline

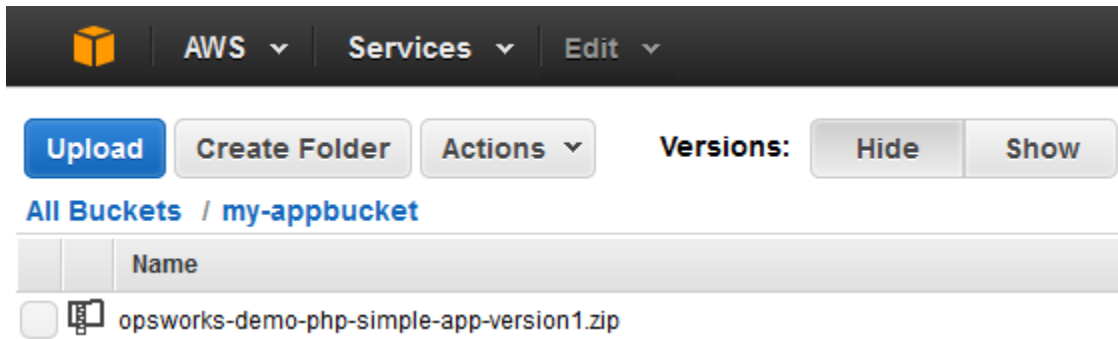
Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

を使用してデプロイしたアプリケーションまたはクックブックのコードを変更すると CodePipeline、更新されたアーティファクトは によって CodePipeline ターゲットインスタンス (この場合はターゲット AWS OpsWorks スタックスタック) に自動的にデプロイされます。このセクションでは、サンプルPHPアプリでコードを更新するときの自動再デプロイについて説明します。

サンプルアプリケーションでコードを編集するには

1. にサインイン AWS Management Console し、 で Amazon S3 コンソールを開きます <https://console.aws.amazon.com/s3/>。
2. サンプルPHPアプリケーションを保存するバケットを開きます。



3. アプリを含むZIPファイルを選択します。[Actions] メニューで、[Download] を選択します。
4. ダイアログボックスで、コンテキスト (右クリック) メニューを開き、ダウンロード を選択し、ZIPファイルを便利な場所に保存します。[OK] を選択します。
5. ZIP ファイルの内容を便利な場所に抽出します。展開したフォルダとそのサブフォルダやコンテンツで権限を変更し、編集を許可することが必要な場合があります。[opsworks-demo-php-simple-app-version1] フォルダで、編集する [index.php] ファイルを開きます。
6. 「Your PHP application is now running」という語句で検索します。「Your PHP application is now running」というテキストを「You've just deployed your first app to AWS OpsWorks with AWS CodePipeline,」に置き換えます。変数は編集しないでください。

```
<body>
<div class="container">
  <div class="hero-unit">
    <h1>Simple PHP App</h1>
    <h2>Congratulations!</h2>
    <p>You've just deployed your first app to AWS OpsWorks with AWS CodePipeline,</p>
    <p>on the host &ldquo;<?php echo gethostname(); ?&rdquo; </p>
    <p>in your own dedicated environment in the AWS&nbsp;Cloud.</p>
    <p>This host is running PHP version <?php echo phpversion(); ?>.</p>
  </div>
</div>

<script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
<script src="assets/js/bootstrap.min.js"></script>
</body>
```

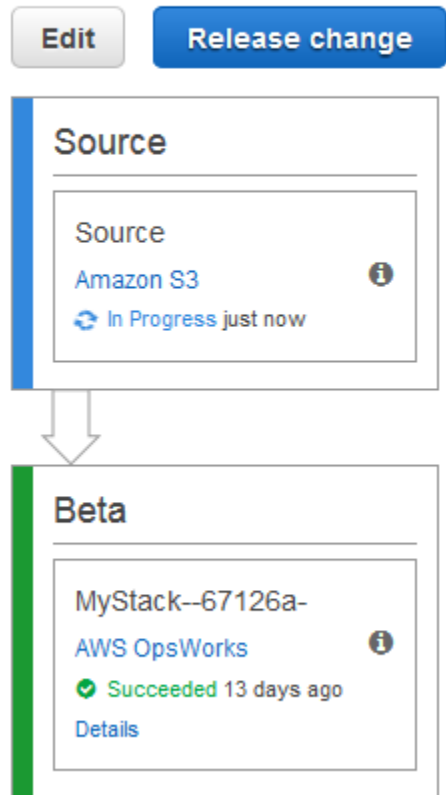
7. index.php ファイルを保存して閉じます。
8. opsworks-demo-php-simple-app-version1 フォルダを圧縮し、ファイルをZIP便利な場所に保存します。ZIP ファイルの名前を変更しないでください。
9. 新しいZIPファイルを Amazon S3 バケットにアップロードします。このウォークスルーでは、バケット名は my-appbucket です。
10. CodePipeline コンソールを開き、AWS OpsWorks スタックパイプライン () を開きます MyOpsWorksPipeline。[Release Change] を選択します。

(CodePipeline は、Amazon S3 バケット内のアプリの更新バージョンからのコード変更を検出するのを待つことができます。このチュートリアルでは、時間を節約するために、リリース変更を選択するように指示します)。

11. パイプラインのステージで CodePipeline 実行されているを確認します。まず、はソースアーティファクトの変更 CodePipeline を検出します。

MyOpsWorksPipeline

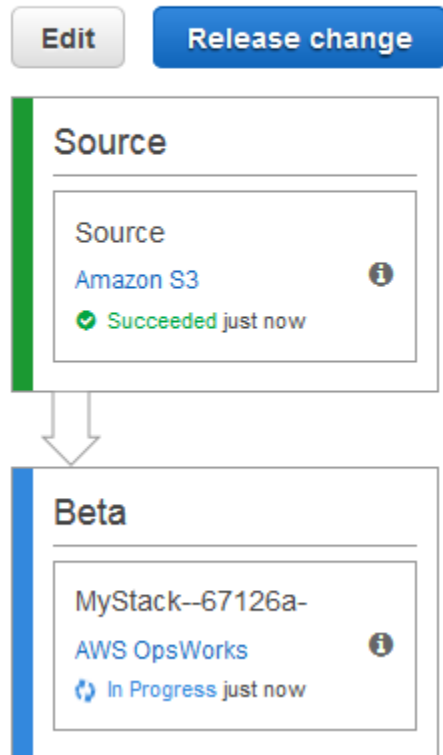
View progress and manage your pipeline.



CodePipeline は、更新されたコードを スタックの AWS OpsWorks スタックにプッシュします。

MyOpsWorksPipeline

View progress and manage your pipeline.



12. パイプラインの両方のステージが正常に完了したら、スタック () で AWS OpsWorks スタックを開きますMyStack。
13. MyStack プロパティページで、インスタンス を選択します。
14. [Public IP] 列で、インスタンスのパブリック IP アドレスを選択し、更新されたアプリケーションのテキストを表示します。

Simple PHP App

Congratulations!

You've just deployed your first app to AWS OpsWorks with AWS CodePipeline, on the host "php-app1", in your own dedicated environment in the AWS Cloud. This host is running PHP version 5.3.29.

ステップ 7 (オプション): リソースをクリーンアップする

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS アカウントへの不要な請求を防ぐため、このチュートリアルで使った AWS リソースを削除できます。これらの AWS リソースには、AWS OpsWorks スタックスタック、IAM ロールとインスタンスプロファイル、およびで作成したパイプラインが含まれます CodePipeline。ただし、AWS OpsWorks スタックと の詳細については、これらの AWS リソースを引き続き使用することをお勧めします CodePipeline。これらのリソースを保持するには、このウォークスルーを完了する必要があります。

アプリケーションをスタックから削除するには

AWS CloudFormation テンプレートの一部としてアプリケーションを作成または適用しなかったため、スタックを削除する前に PHP テストアプリケーションを削除してください AWS CloudFormation。

1. AWS OpsWorks スタックコンソールのサービスナビゲーションペインで、アプリ を選択します。
2. アプリページで を選択し PHPTestApp、アクション での削除を選択します。確認のプロンプトが表示されたら、「 の削除」を選択します。AWS OpsWorks スタックはアプリを削除します。

スタックを削除するには

AWS CloudFormation テンプレートを実行してスタックを作成したため、テンプレートが作成したレイヤー、インスタンス、インスタンスプロファイル、セキュリティグループなどのスタックを AWS CloudFormation コンソールで削除できます。

1. AWS CloudFormation コンソールを開きます。

2. AWS CloudFormation コンソールダッシュボードで、作成したスタック () を選択しますMyStack。[Actions] メニューで、[Delete Stack] を選択します。確認を求められたら [Yes, Delete] を選択します。
3. スタックのステータス列に DELETE_COMPLETE が表示されるまで待ちます。

パイプラインを削除するには

1. CodePipeline コンソールを開きます。
2. CodePipeline ダッシュボードで、このチュートリアル用に作成したパイプラインを選択します。
3. パイプラインページで、[Edit] を選択します。
4. [Edit] ページで、[Delete] を選択します。確認を求められたら [Delete] (削除) を選択します。

AWS OpsWorks スタック CLI の使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックコマンドラインインターフェイス (CLI) は、コンソールと同じ機能を提供し、さまざまなタスクに使用できます。AWS OpsWorks スタック CLI は の一部です AWS CLI。のインストールと設定方法などの詳細については、「[AWS コマンドラインインターフェイスとは AWS CLI](#)」を参照してください。各コマンドの詳細な説明については、「[AWS OpsWorks スタック リファレンス](#)」を参照してください。

Note

Windows ベースのワークステーションを使用している場合は、AWS Tools for Windows を実行して PowerShell、コマンドラインから AWS OpsWorks スタックオペレーションを実行

することもできます。詳細については、[「AWS Tools for Windows PowerShell」](#)を参照してください。

AWS OpsWorks スタックコマンドの一般的な形式は次のとおりです。

```
aws opsworks --region us-west-1 opsworks command-name [--argument1 value] [...]
```

引数値が JSON オブジェクトである場合は、" 文字をエスケープする必要があります。そうしないと、コマンドから JSON が有効でないというエラーが返されます。例えば、JSON オブジェクトが `{"somekey":"somevalue"}` である場合は、`{\"somekey\": \"somevalue\"}` という形式にします。別のアプローチは、JSON オブジェクトをファイルに保存し、`file://` を使用してコマンドラインでそのファイルを指定することです。以下の例では、`appsource.json` に保存されたアプリケーションソースオブジェクトを使用して、アプリケーションを作成しています。

```
aws opsworks --region us-west-1 create-app --stack-id 8c428b08-a1a1-46ce-a5f8-feddc43771b8 --name SimpleJSP --type java --app-source file://appsource.json
```

ほとんどの場合、コマンドから JSON オブジェクトとしてパッケージ化された 1 つ以上の値が返されます。以下のセクションでは、いくつかの例を示します。各コマンドの戻り値の詳細については、[「AWS OpsWorks スタックリファレンス」](#)を参照してください。

Note

AWS CLI コマンドでは、例に示すようにリージョンを指定する必要があります。--リージョンパラメータの有効値を次の表に示します。AWS OpsWorks Stacks コマンド文字列を簡素化するには、--regionパラメータを省略できるように、デフォルトのリージョンを指定するように CLI を設定します。通常、複数のリージョンエンドポイントで作業する場合は、デフォルトのリージョンエンドポイントを使用する AWS CLI ように を設定しないでください。カナダ (中部) リージョンエンドポイントは API および AWS CLI でのみ使用できません。で作成したスタックでは使用できません AWS Management Console。詳細については、[「AWS リージョンの設定」](#)を参照してください。

リージョン名	コマンドコード
米国東部(オハイオ州)リージョン	us-east-2
米国東部 (バージニア州北部) リージョン	us-east-1

リージョン名	コマンドコード
US West (N. California) リージョン	us-west-1
米国西部 (オレゴン州) リージョン	us-west-2
カナダ (中部) リージョン	ca-central-1
欧州 (アイルランド) リージョン	eu-west-1
欧州 (ロンドン) リージョン	eu-west-2
欧州 (パリ) リージョン	eu-west-3
欧州 (フランクフルト) リージョン	eu-central-1
アジアパシフィック (東京) リージョン	ap-northeast-1
アジアパシフィック (ソウル) リージョン	ap-northeast-2
アジアパシフィック (ムンバイ) リージョン	ap-south-1
アジアパシフィック (シンガポール) リージョン	ap-southeast-1
アジアパシフィック (シドニー) リージョン	ap-southeast-2
南米 (サンパウロ) リージョン	sa-east-1

CLI コマンドを使用するには、適切なアクセス許可が必要です。AWS OpsWorks スタックのアクセス権限の詳細については、「[ユーザー許可の管理](#)」を参照してください。特定のコマンドに必要なアクセス許可を調べるには、「[AWS OpsWorks スタックリファレンス](#)」のコマンドリファレンスのページを参照してください。

以下のセクションでは、AWS OpsWorks スタック CLI を使用してさまざまな一般的なタスクを実行する方法について説明します。

インスタンスの作成 (create-instance)

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[create-instance](#) コマンドを使用して、指定したスタックにインスタンスを作成します。

トピック

- [デフォルトのホスト名を使用したインスタンスの作成](#)
- [テーマ付きホスト名のインスタンスの作成](#)
- [カスタム AMI を使用したインスタンスの作成](#)

デフォルトのホスト名を使用したインスタンスの作成

```
C:\>aws opsworks --region us-west-1 create-instance --stack-id 935450cc-61e0-4b03-a3e0-160ac817d2bb
--layer-ids 5c8c272a-f2d5-42e3-8245-5bf3927cb65b --instance-type m1.large --os
"Amazon Linux"
```

引数は次のとおりです。

- `stack-id` – スタック ID は、コンソールのスタックの設定ページ (ID を参照 OpsWorks) から取得するか、[describe-stacks](#) を呼び出すことで取得できます。
- `layer-ids` – レイヤー IDs は、コンソールのレイヤーの詳細ページ (ID を探 OpsWorks してください) から、または [describe-layers](#) を呼び出すことで取得できます。この例では、インスタンスは 1 つのレイヤーにのみ属しています。
- `instance-type` – インスタンス (この例では) `m1.large` のメモリ、CPU、ストレージ容量、および時間あたりのコストの定義を指定します。
- `os` - インスタンスのオペレーティングシステム (この例では Amazon Linux) です。

次に示すように、インスタンス ID を含む JSON オブジェクトがコマンドから返されます。

```
{
  "InstanceId": "5f9adeaa-c94c-42c6-aeef-28a5376002cd"
}
```

この例では、デフォルトのホスト名 (整数のみ) を持つインスタンスが作成されます。以下のセクションでは、テーマから生成されたホスト名を持つインスタンスを作成する方法を説明します。

テーマ付きホスト名のインスタンスの作成

テーマ付きホスト名のインスタンスを作成することもできます。スタックを作成する際にテーマを指定します。詳細については、「」を参照してください[新しいスタックを作成する](#)。インスタンスを作成するには、まず [get-hostname-suggestion](#) を呼び出して名前を生成します。例えば：

```
C:\>aws opsworks get-hostname-suggestion --region us-west-1 --layer-id 5c8c272a-f2d5-42e3-8245-5bf3927cb65b
```

デフォルトの Layer Dependent テーマを指定した場合は、`get-hostname-suggestion` によってレイヤーの短縮名に単に数字が付加されます。詳細については、「[新しいスタックを作成する](#)」を参照してください。

コマンドによって生成されたホスト名が返されます。

```
{
  "Hostname": "php-app2",
  "LayerId": "5c8c272a-f2d5-42e3-8245-5bf3927cb65b"
}
```

生成された名前を `create-instance` に渡すには、次のように `hostname` 引数を使用します。

```
c:\>aws --region us-west-1 opsworks create-instance --stack-id 935450cc-61e0-4b03-a3e0-160ac817d2bb
  --layer-ids 5c8c272a-f2d5-42e3-8245-5bf3927cb65b --instance-type m1.large --os
"Amazon Linux" --hostname "php-app2"
```

カスタム AMI を使用したインスタンスの作成

次の [create-instance](#) コマンドは、カスタム AMI を使用してインスタンスを作成します。これは、スタックのリージョンからのものである必要があります。AWS OpsWorks スタックのカスタム AMI を作成する方法の詳細については、「」を参照してください [カスタム AMI の使用](#)。

```
C:\>aws opsworks create-instance --region us-west-1 --stack-id c5ef46ce-3ccd-472c-
a3de-9bec94c6028e
    --layer-ids 6ff8a2ac-c9cc-49cf-9c67-fc852539ade4 --instance-type c3.large --os
Custom
    --ami-id ami-6c61f104
```

引数は次のとおりです。

- `stack-id` – スタック ID は、コンソールのスタックの設定ページから取得するか (OpsWorks ID を探します)、 [describe-stacks](#) を呼び出すことで取得できます。
- `layer-ids` – レイヤー IDs は、コンソールのレイヤーの詳細ページ (ID を探OpsWorks してください) から、または [describe-layers](#) を呼び出すことで取得できます。この例では、インスタンスは 1 つのレイヤーにのみ属しています。
- `instance-type` - この値はインスタンスのメモリ、CPU、ストレージ容量、および時間あたりのコストを定義します。AMI と互換性がある必要があります (この例では `c3.large` を参照)。
- `os` – インスタンスのオペレーティングシステムです。カスタム AMI に対して Custom に設定する必要があります。
- `ami-id` – AMI ID です。 `ami-6c61f104` のようになります

Note

カスタム AMI を使用すると、ブロックデバイスマッピングはサポートされず、`--block-device-mappings` オプションに指定した値は無視されます。

次に示すように、インスタンス ID を含む JSON オブジェクトがコマンドから返されます。

```
{
  "InstanceId": "5f9adeaa-c94c-42c6-aeeef-28a5376002cd"
```


}

アプリケーションのデプロイ (create-deployment)

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[create-deployment](#) コマンドを使用して、指定したスタックにアプリケーションをデプロイします。

トピック

- [アプリケーションのデプロイ](#)

アプリケーションのデプロイ

```
aws opsworks --region us-west-1 create-deployment --stack-id cfb7e082-ad1d-4599-8e81-  
de1c39ab45bf  
  --app-id 307be5c8-d55d-47b5-bd6e-7bd417c6c7eb --command "{\"Name\":\"deploy\"}"
```

引数は次のとおりです。

- `stack-id` – スタック ID は、コンソールのスタックの設定ページ (ID を参照 OpsWorks) から取得するか、`describe-stacks` を呼び出して取得できます。
- `app-id` – アプリケーション ID は、アプリケーションの詳細ページ (ID を参照 OpsWorks) から、または [describe-apps](#) を呼び出すことで取得できます。
- `command` – 引数は、コマンド名を `deploy` に設定する JSON オブジェクトを取ります。このコマンドは、指定したアプリケーションをスタックにデプロイします。

JSON オブジェクトの " 文字がすべてエスケープされていることに注意してください。そうしないと、JSON が無効であるというエラーがコマンドから返されます。

次に示すように、デプロイメント ID を含む JSON オブジェクトがコマンドから返されます。

```
{
  "DeploymentId": "5746c781-df7f-4c87-84a7-65a119880560"
}
```

Note

前述の例では、スタックのすべてのインスタンスにデプロイします。インスタンスの指定したサブセットにデプロイするには、`instance-ids` 引数を追加し、インスタンス ID をリストします。

スタックのアプリケーションのリストを取得する (describe-apps)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[describe-apps](#) コマンドを使用して、スタックのアプリケーションのリストまたは指定したアプリケーションに関する詳細を取得します。

```
aws opsworks --region us-west-1 describe-apps --stack-id 38ee91e2-abdc-4208-
a107-0b7168b3cc7a
```

前述の例では、各アプリケーションの情報を含む JSON オブジェクトが返されます。この例では、対象となるアプリケーションは 1 つのみです。各パラメータの説明については、「[describe-apps](#)」を参照してください。

```
{
  "Apps": [
```

```
{
  "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
  "AppSource": {
    "Url": "url",
    "Type": "archive"
  },
  "Name": "SimpleJSP",
  "EnableSsl": false,
  "SslConfiguration": {},
  "AppId": "da1decc1-0dff-43ea-ad7c-bb667cd87c8b",
  "Attributes": {
    "RailsEnv": null,
    "AutoBundleOnDeploy": "true",
    "DocumentRoot": "ROOT"
  },
  "Shortname": "simplejsp",
  "Type": "other",
  "CreatedAt": "2013-08-01T21:46:54+00:00"
}
]
```

スタックのコマンドのリストを取得する (describe-commands)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[describe-commands](#) コマンドを使用して、スタックのコマンドのリストまたは指定したコマンドに関する詳細を取得します。次の例では、指定されたインスタンスで実行されたコマンドに関する情報を取得します。

```
aws opsworks --region us-west-1 describe-commands --instance-id
8c2673b9-3fe5-420d-9cfa-78d875ee7687
```

各コマンドの詳細を含む JSON オブジェクトがコマンドから返されます。この例では、コマンド名、デプロイまたはデプロイ解除が Type パラメータによって識別されます。その他のパラメータの説明については、「[describe-commands](#)」を参照してください。

```
{
  "Commands": [
    {
      "Status": "successful",
      "CompletedAt": "2013-07-25T18:57:47+00:00",
      "InstanceId": "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
      "DeploymentId": "6ed0df4c-9ef7-4812-8dac-d54a05be1029",
      "AcknowledgedAt": "2013-07-25T18:57:41+00:00",
      "LogUrl": "https://s3.amazonaws.com/prod_stage-log/logs/008c1a91-ec59-4d51-971d-3adff54b00cc?AWSAccessKeyId=AIDACKCEVSQ6C2EXAMPLE&Expires=1375394373&Signature=HkXil6UuNfxTCC37EPQAa462E1E%3D&response-cache-control=private&response-content-encoding=gzip&response-content-type=text%2Fplain",
      "Type": "undeploy",
      "CommandId": "008c1a91-ec59-4d51-971d-3adff54b00cc",
      "CreatedAt": "2013-07-25T18:57:34+00:00",
      "ExitCode": 0
    },
    {
      "Status": "successful",
      "CompletedAt": "2013-07-25T18:55:40+00:00",
      "InstanceId": "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
      "DeploymentId": "19d3121e-d949-4ff2-9f9d-94eac087862a",
      "AcknowledgedAt": "2013-07-25T18:55:32+00:00",
      "LogUrl": "https://s3.amazonaws.com/prod_stage-log/logs/899d3d64-0384-47b6-a586-33433aad117c?AWSAccessKeyId=AIDACKCEVSQ6C2EXAMPLE&Expires=1375394373&Signature=xMsJvtLuUqWmsr8s%2FAjVru0BtRs%3D&response-cache-control=private&response-content-encoding=gzip&response-content-type=text%2Fplain",
      "Type": "deploy",
      "CommandId": "899d3d64-0384-47b6-a586-33433aad117c",
      "CreatedAt": "2013-07-25T18:55:29+00:00",
      "ExitCode": 0
    }
  ]
}
```

スタックのデプロイメントのリストを取得する (describe-deployments)

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[describe-deployments](#) コマンドを使用して、スタックのデプロイメントのリストを取得するか、指定したデプロイメントに関する詳細を取得します。

```
aws opsworks --region us-west-1 describe-deployments --stack-id 38ee91e2-abdc-4208-a107-0b7168b3cc7a
```

前述のコマンドでは、指定したスタックの各デプロイメントの詳細を含む JSON オブジェクトがコマンドから返されます。各パラメータの説明については、「[describe-deployments](#)」を参照してください。

```
{
  "Deployments": [
    {
      "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
      "Status": "successful",
      "CompletedAt": "2013-07-25T18:57:49+00:00",
      "DeploymentId": "6ed0df4c-9ef7-4812-8dac-d54a05be1029",
      "Command": {
        "Args": {},
        "Name": "undeploy"
      },
      "CreatedAt": "2013-07-25T18:57:34+00:00",
      "Duration": 15,
      "InstanceIds": [
        "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
        "9e588a25-35b2-4804-bd43-488f85ebe5b7"
      ]
    },
    {
      "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
```

```
"Status": "successful",
"CompletedAt": "2013-07-25T18:56:41+00:00",
"IamUserArn": "arn:aws:iam::444455556666:user/example-user",
"DeploymentId": "19d3121e-d949-4ff2-9f9d-94eac087862a",
"Command": {
  "Args": {},
  "Name": "deploy"
},
"InstanceIds": [
  "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
  "9e588a25-35b2-4804-bd43-488f85ebe5b7"
],
"Duration": 72,
"CreatedAt": "2013-07-25T18:55:29+00:00"
}
]
}
```

スタックの Elastic IP アドレスを一覧表示する (describe-elastic-ips)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[describe-elastic-ips](#) コマンドを使用して、スタックに登録されている Elastic IP アドレスを一覧表示するか、指定された Elastic IP アドレスの詳細を取得します。

```
aws opsworks --region us-west-2 describe-elastic-ips --instance-id b62f3e04-e9eb-436c-a91f-d9e9a396b7b0
```

前述のコマンドでは、指定したインスタンスの各 Elastic IP アドレス (この例では 1 つ) の詳細を含む JSON オブジェクトがコマンドから返されます。各パラメータの説明については、「」を参照してください [describe-elastic-ips](#)。

```
{
```

```
"ElasticIps": [  
  {  
    "Ip": "192.0.2.0",  
    "Domain": "standard",  
    "Region": "us-west-2"  
  }  
]  
}
```

スタックのインスタンスのリストを取得する (describe-instances)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[describe-instances](#) コマンドを使用して、スタックのインスタンスのリストを取得するか、指定したインスタンスに関する詳細を取得します。

```
C:\>aws opsworks --region us-west-2 describe-instances --stack-id 38ee91e2-abdc-4208-a107-0b7168b3cc7a
```

前述のコマンドでは、指定したスタックの各インスタンスの詳細を含む JSON オブジェクトがコマンドから返されます。各パラメータの説明については、「[describe-instances](#)」を参照してください。

```
{  
  "Instances": [  
    {  
      "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",  
      "SshHostRsaKeyFingerprint":  
"f4:3b:8e:27:1b:73:98:80:5d:d7:33:e2:b8:c8:8f:de",  
      "Status": "stopped",  
      "AvailabilityZone": "us-west-2a",  
      "SshHostDsaKeyFingerprint":  
"e8:9b:c7:02:18:2a:bd:ab:45:89:21:4e:af:0b:07:ac",
```

```
    "InstanceId": "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
    "Os": "Amazon Linux",
    "Hostname": "db-master1",
    "SecurityGroupIds": [],
    "Architecture": "x86_64",
    "RootDeviceType": "instance-store",
    "LayerIds": [
      "41a20847-d594-4325-8447-171821916b73"
    ],
    "InstanceType": "c1.medium",
    "CreatedAt": "2013-07-25T18:11:27+00:00"
  },
  {
    "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
    "SshHostRsaKeyFingerprint":
"ae:3a:85:54:66:f3:ce:98:d9:83:39:1e:10:a9:38:12",
    "Status": "stopped",
    "AvailabilityZone": "us-west-2a",
    "SshHostDsaKeyFingerprint":
"5b:b9:6f:5b:1c:ec:55:85:f3:45:f1:28:25:1f:de:e4",
    "InstanceId": "9e588a25-35b2-4804-bd43-488f85ebe5b7",
    "Os": "Amazon Linux",
    "Hostname": "tomcustom1",
    "SecurityGroupIds": [],
    "Architecture": "x86_64",
    "RootDeviceType": "instance-store",
    "LayerIds": [
      "e6cbcd29-d223-40fc-8243-2eb213377440"
    ],
    "InstanceType": "c1.medium",
    "CreatedAt": "2013-07-25T18:15:52+00:00"
  }
]
}
```

アカウントのスタックのリストを取得する (describe-stacks)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

`describe-stacks` コマンドを使用して、アカウントのスタックのリストを取得するか、指定したスタックに関する詳細を取得します。

```
aws opsworks --region us-west-2 describe-stacks
```

前述のコマンドでは、アカウントの各スタック (この例では 2 つ) の詳細を含む JSON オブジェクトがコマンドから返されます。各パラメータの説明については、「[describe-stacks](#)」を参照してください。

```
{
  "Stacks": [
    {
      "ServiceRoleArn": "arn:aws:iam::444455556666:role/aws-opsworks-service-
role",
      "StackId": "aeb7523e-7c8b-49d4-b866-03aae9d4fbc",
      "DefaultRootDeviceType": "instance-store",
      "Name": "TomStack-sd",
      "ConfigurationManager": {
        "Version": "11.4",
        "Name": "Chef"
      },
      "UseCustomCookbooks": true,
      "CustomJson": "{\n  \"tomcat\": {\n    \"base_version\": 7,\n  \"java_opts\": \"-Djava.awt.headless=true -Xmx256m\"\n  },\n  \"datasources\": {\n    \"ROOT\": \"jdbc/mydb\"\n  }\n}",
      "Region": "us-west-2",
      "DefaultInstanceProfileArn": "arn:aws:iam::444455556666:instance-profile/
aws-opsworks-ec2-role",
      "CustomCookbooksSource": {
        "Url": "git://github.com/example-repo/tomcustom.git",
        "Type": "git"
      },
      "DefaultAvailabilityZone": "us-west-2a",
      "HostnameTheme": "Layer_Dependent",
      "Attributes": {
        "Color": "rgb(45, 114, 184)"
      },
      "DefaultOs": "Amazon Linux",
    }
  ]
}
```

```
    "CreatedAt": "2013-08-01T22:53:42+00:00"
  },
  {
    "ServiceRoleArn": "arn:aws:iam::444455556666:role/aws-opsworks-service-
role",
    "StackId": "40738975-da59-4c5b-9789-3e422f2cf099",
    "DefaultRootDeviceType": "instance-store",
    "Name": "MyStack",
    "ConfigurationManager": {
      "Version": "11.4",
      "Name": "Chef"
    },
    "UseCustomCookbooks": false,
    "Region": "us-west-2",
    "DefaultInstanceProfileArn": "arn:aws:iam::444455556666:instance-profile/
aws-opsworks-ec2-role",
    "CustomCookbooksSource": {},
    "DefaultAvailabilityZone": "us-west-2a",
    "HostnameTheme": "Layer_Dependent",
    "Attributes": {
      "Color": "rgb(45, 114, 184)"
    },
    "DefaultOs": "Amazon Linux",
    "CreatedAt": "2013-10-25T19:24:30+00:00"
  }
]
}
```

スタックのレイヤーのリストを取得 (describe-layers)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[describe-layers](#) コマンドを使用して、スタックのレイヤーのリストを取得するか、指定したレイヤーの詳細を取得します。

```
aws opsworks --region us-west-2 describe-layers --stack-id 38ee91e2-abdc-4208-
a107-0b7168b3cc7a
```

前述のコマンドでは、指定したスタック (この例では MySQL レイヤーとカスタムレイヤー) の詳細を含む JSON オブジェクトがコマンドから返されます。各パラメータの説明については、「[describe-layers](#)」を参照してください。

```
{
  "Layers": [
    {
      "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
      "Type": "db-master",
      "DefaultSecurityGroupNames": [
        "AWS-OpsWorks-DB-Master-Server"
      ],
      "Name": "MySQL",
      "Packages": [],
      "DefaultRecipes": {
        "Undeploy": [],
        "Setup": [
          "opsworks_initial_setup",
          "ssh_host_keys",
          "ssh_users",
          "mysql::client",
          "dependencies",
          "ebs",
          "opsworks_ganglia::client",
          "mysql::server",
          "dependencies",
          "deploy::mysql"
        ],
        "Configure": [
          "opsworks_ganglia::configure-client",
          "ssh_users",
          "agent_version",
          "deploy::mysql"
        ],
        "Shutdown": [
          "opsworks_shutdown::default",
          "mysql::stop"
        ],
        "Deploy": [
```

```
        "deploy::default",
        "deploy::mysql"
    ]
},
"CustomRecipes": {
    "Undeploy": [],
    "Setup": [],
    "Configure": [],
    "Shutdown": [],
    "Deploy": []
},
"EnableAutoHealing": false,
"LayerId": "41a20847-d594-4325-8447-171821916b73",
"Attributes": {
    "MysqlRootPasswordUbiquitous": "true",
    "RubygemsVersion": null,
    "RailsStack": null,
    "HaproxyHealthCheckMethod": null,
    "RubyVersion": null,
    "BundlerVersion": null,
    "HaproxyStatsPassword": null,
    "PassengerVersion": null,
    "MemcachedMemory": null,
    "EnableHaproxyStats": null,
    "ManageBundler": null,
    "NodejsVersion": null,
    "HaproxyHealthCheckUrl": null,
    "MysqlRootPassword": "*****FILTERED*****",
    "GangliaPassword": null,
    "GangliaUser": null,
    "HaproxyStatsUrl": null,
    "GangliaUrl": null,
    "HaproxyStatsUser": null
},
"Shortname": "db-master",
"AutoAssignElasticIps": false,
"CustomSecurityGroupIds": [],
"CreatedAt": "2013-07-25T18:11:19+00:00",
"VolumeConfigurations": [
    {
        "MountPoint": "/vol/mysql",
        "Size": 10,
        "NumberOfDisks": 1
    }
]
```

```
]
},
{
  "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
  "Type": "custom",
  "DefaultSecurityGroupNames": [
    "AWS-OpsWorks-Custom-Server"
  ],
  "Name": "TomCustom",
  "Packages": [],
  "DefaultRecipes": {
    "Undeploy": [],
    "Setup": [
      "opsworks_initial_setup",
      "ssh_host_keys",
      "ssh_users",
      "mysql::client",
      "dependencies",
      "ebs",
      "opsworks_ganglia::client"
    ],
    "Configure": [
      "opsworks_ganglia::configure-client",
      "ssh_users",
      "agent_version"
    ],
    "Shutdown": [
      "opsworks_shutdown::default"
    ],
    "Deploy": [
      "deploy::default"
    ]
  },
  "CustomRecipes": {
    "Undeploy": [],
    "Setup": [
      "tomcat::setup"
    ],
    "Configure": [
      "tomcat::configure"
    ],
    "Shutdown": [],
    "Deploy": [
      "tomcat::deploy"
    ]
  }
}
```

```
    ],
  },
  "EnableAutoHealing": true,
  "LayerId": "e6cbcd29-d223-40fc-8243-2eb213377440",
  "Attributes": {
    "MysqlRootPasswordUbiquitous": null,
    "RubygemsVersion": null,
    "RailsStack": null,
    "HaproxyHealthCheckMethod": null,
    "RubyVersion": null,
    "BundlerVersion": null,
    "HaproxyStatsPassword": null,
    "PassengerVersion": null,
    "MemcachedMemory": null,
    "EnableHaproxyStats": null,
    "ManageBundler": null,
    "NodejsVersion": null,
    "HaproxyHealthCheckUrl": null,
    "MysqlRootPassword": null,
    "GangliaPassword": null,
    "GangliaUser": null,
    "HaproxyStatsUrl": null,
    "GangliaUrl": null,
    "HaproxyStatsUser": null
  },
  "Shortname": "tomcustom",
  "AutoAssignElasticIps": false,
  "CustomSecurityGroupIds": [],
  "CreatedAt": "2013-07-25T18:12:53+00:00",
  "VolumeConfigurations": []
}
]
```

レシピの実行 (create-deployment)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

`create-deployment` コマンドを使用して、[スタックコマンド](#)と[デプロイメントコマンド](#)を実行します。次の例では、指定されたスタックでカスタムレシピを実行するためのスタックコマンドを実行します。

```
aws opsworks --region us-west-1 create-deployment --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
  --command "{\"Name\": \"execute_recipes\", \"Args\": {\"recipes\": [\"phpapp::appsetup
\"]}}"
```

command 引数は、次のような形式の JSON オブジェクトを取得します。

- Name - コマンド名を指定します。この例で使用した `execute_recipes` コマンドは、指定されたレシピをスタックのインスタンス上で実行します。
- Args - 引数とそれらの値のリストを指定します。この例では、1つの引数 `recipes` を使用しています。これには、実行するレシピ (`phpapp::appsetup`) が指定されています。

JSON オブジェクトの " 文字がすべてエスケープされていることに注意してください。そうしないと、JSON が無効であるというエラーがコマンドから返されます。

コマンドは、`describe-commands` などのその他の CLI コマンドでこのコマンドを識別するために使用できる、デプロイメント ID を返します。

```
{
  "DeploymentId": "5cbaa7b9-4e09-4e53-aa1b-314fbd106038"
}
```

インストールの依存関係 (create-deployment)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

`create-deployment` コマンドを使用して、[スタックコマンド](#)と[デプロイメントコマンド](#)を実行します。次の例では、スタックのインスタンスの依存関係を更新するための `update_dependencies` スタックコマンドを実行します。

```
aws opsworks --region us-west-1 create-deployment --stack-id 935450cc-61e0-4b03-a3e0-160ac817d2bb
--command "{\"Name\":\"install_dependencies\"}"
```

`command` 引数は、`Name` パラメータが指定された JSON オブジェクトを取得します。この例では、パラメータの値にコマンド名 `install_dependencies` が指定されています。JSON オブジェクトの `"` 文字がすべてエスケープされていることに注意してください。そうしないと、JSON が無効であるというエラーがコマンドから返されます。

コマンドは、`describe-commands` などのその他の CLI コマンドでこのコマンドを識別するために使用できる、デプロイメント ID を返します。

```
{
  "DeploymentId": "aef5b255-8604-4928-81b3-9b0187f962ff"
}
```

スタック設定の更新 (update-stack)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

`update-stack` コマンドを使用して、指定したスタックの設定を更新します。以下の例では、スタックを更新してカスタム JSON を [スタック設定属性](#) に追加します。


```
aws opsworks --region us-west-1 update-stack --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
  --custom-json "{\"somekey\":\"somevalue\"}" --service-role-arn
arn:aws:iam::444455556666:role/aws-opsworks-service-role
```

JSON オブジェクトの " 文字がすべてエスケープされていることに注意してください。そうしないと、JSON が無効であるというエラーがコマンドから返されます。

Note

また、この例では、スタック用のサービスロールを指定しています。service-role-arn を有効なサービスロール ARN に設定する必要があります。そうしないと、アクションは失敗します。デフォルト値はありません。必要に応じて、スタックの現在のサービスロール ARN を指定することもできますが、その場合は明示的に指定する必要があります。

update-stack コマンドから返される値はありません。

デバッグとトラブルシューティングのガイド

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

レシピをデバッグしたり、サービスの問題をトラブルシューティングしたりする必要がある場合、通常、以下の手順を順番に実行するのが最も適切な方法です。

1. 現在の問題が「[一般的なデバッグとトラブルシューティングの問題](#)」に掲載されていないかを確認します。
2. [AWS OpsWorks スタックフォーラム](#)を検索して、その問題が議論されていないか確認します。

フォーラムには経験豊富なユーザーが多数含まれており、AWS OpsWorks スタックチームによって監視されています。

3. レシピの問題については、「[レシピのデバッグ](#)」を参照してください。
4. AWS OpsWorks スタックサポートに問い合わせるか、[AWS OpsWorks スタックフォーラム](#) に問題を投稿してください。

以下のセクションでは、レシピをデバッグするためのガイダンスを提供します。最終セクションでは、一般的なデバッグやトラブルシューティングに関する問題とその解決方法について説明します。

Note

各 Chef はログを作成します。このログは、実行の詳細な説明を提供する有用なトラブルシューティングリソースとなります。ログの詳細情報の量を指定するには、目的のログレベルを指定するカスタムレシピに、[Chef::Log.level](#) ステートメントを追加します。デフォルト値は、`:info` です。次の例では、Chef ログレベルを `:debug` に設定する方法を示します。この場合、実行の最も詳細な説明が提供されます。

```
Chef::Log.level = :debug
```

Chef ログを表示および解釈する方法の詳細については、「[Chef ログ](#)」を参照してください。

トピック

- [レシピのデバッグ](#)
- [一般的なデバッグとトラブルシューティングの問題](#)

レシピのデバッグ

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

ライフサイクルイベントが発生したときや、[\[Execute Recipes\]](#) (レシピの実行) スタックコマンドを実行したときに、AWS OpsWorks スタックは [\[agent\]](#) (エージェント) に対してコマンドを発行し、指定されたインスタンスで [\[Chef Solo\]](#) (ソロシェフ) の実行を開始して、適切なレシピ (カスタムレシピも含む) を実行します。このセクションでは、失敗したレシピをデバッグする方法について説明します。

トピック

- [障害が発生したインスタンスへのログイン](#)
- [Chef ログ](#)
- [AWS OpsWorks スタックエージェント CLI の使用](#)

障害が発生したインスタンスへのログイン

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

レシピが失敗すると、インスタンスはオンラインではなく `setup_failed` 状態になります。インスタンスは AWS OpsWorks スタックに関してオンラインではありませんが、EC2 インスタンスは実行中であり、問題のトラブルシューティングのためにログインすると便利です。たとえば、アプリケーションまたはカスタムクックブックが正しくインストールされているかどうかを確認できます。AWS OpsWorks スタックの [SSH](#) および [RDP](#) ログインの組み込みサポートは、オンライン状態のインスタンスでのみ使用できます。ただし、インスタンスに SSH キーペアを割り当てている場合は、以下のようにログインできます。

- Linux インスタンス – SSH キーペアのプライベートキーを使用して、OpenSSH や PuTTY のようなサードパーティ SSH クライアントでログインします。

そのために、EC2 キーペアまたは [個人 SSH キーペア](#) を使用できます。

- Windows インスタンス – EC2 キーペアのプライベートキーを使用して、インスタンスの管理者パスワードを取得します。

そのパスワードを使用して、目的の RDP クライアントでログインします。詳細については、「[管理者としてログイン](#)」を参照してください。[個人 SSH キーペア](#)を使用して、管理者パスワードを取得することはできません。

Chef ログ

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Chef ログは、特にレシピのデバッグに関する主要なトラブルシューティングリソースの 1 つです。AWS OpsWorks スタックは各コマンドの Chef ログをキャプチャし、インスタンスの最新の 30 個のコマンドのログを保持します。実行はデバッグモードであるため、ログには Chef 実行の詳細 (stdout や stderr に送信されたテキストなど) が含まれます。レシピが失敗した場合、ログには Chef スタックトレースが含まれます。

AWS OpsWorks スタックには、Chef ログを表示する方法がいくつかあります。ログ情報を入手したら、その情報を使用して失敗したレシピをデバッグできます。

Note

SSH を使用してインスタンスに接続し、エージェント CLI の `show_log` コマンドを実行することで、指定したログの末尾を表示することもできます。詳細については、「[Chef ログの表示](#)」を参照してください。

トピック

- [コンソールを使用した Chef ログの表示](#)
- [CLI または API を使用した Chef ログの表示](#)
- [インスタンスの Chef ログの表示](#)
- [Chef ログの解釈](#)

• [Chef ログの一般的なエラー](#)

コンソールを使用した Chef ログの表示

Chef ログを表示する最も簡単な方法は、インスタンスの詳細ページを参照することです。[Logs] セクションには、各イベントと [Execute Recipes](#) コマンドのエントリが含まれています。Configure および Setup の各ライフサイクルイベントに対応する configure コマンドと setup コマンドが含まれた、インスタンスの [Logs] セクションを次に示します。



Created at	Command	Duration	Log
✓ 2013-10-02 21:06:56 UTC	configure	00:01:04	show
✓ 2013-10-02 21:01:15 UTC	setup	00:05:40	show

該当するコマンドの [Log] 列で [show] をクリックすると、対応する Chef ログが表示されます。エラーが発生すると、AWS OpsWorks スタックは自動的にログをエラーに開きます。通常はファイルの最後にあります。

CLI または API を使用した Chef ログの表示

AWS OpsWorks Stacks CLI [describe-commands](#) コマンドまたは [DescribeCommands](#) API アクシオンを使用して、Amazon S3 バケットに保存されているログを表示できます。次に、CLI を使用して、指定したインスタンスの現在のログファイルセットを表示する方法を示します。DescribeCommands を使用する場合は基本的にほぼ同じです。

AWS OpsWorks スタックを使用してインスタンスの Chef ログを表示するには

1. インスタンスの詳細ページを開き、その OpsWorks ID 値をコピーします。
2. 次のように、ID 値を使用して describe-commands CLI コマンドを実行します。

```
aws opsworks describe-commands --instance-id 67bf0da2-29ed-4217-990c-d895d51812b9
```

コマンドは、AWS OpsWorks スタックがインスタンスで実行した各コマンドのオブジェクトが埋め込まれている JSON オブジェクトを、最新のものととも返します。Type パラメータには、各組み込みオブジェクトのコマンドタイプ (この例では configure コマンドと setup コマンド) が含まれます。

```
{
  "Commands": [
    {
      "Status": "successful",
      "CompletedAt": "2013-10-25T19:38:36+00:00",
      "InstanceId": "67bf0da2-29ed-4217-990c-d895d51812b9",
      "AcknowledgedAt": "2013-10-25T19:38:24+00:00",
      "LogUrl": "https://s3.amazonaws.com/prod_stage-log/logs/
b6c402df-5c23-45b2-a707-ad20b9c5ae40?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Expires=1382731518&Signature=YkqS5IZN2P4wixjHwoC3aCmbn5s%3D&response-cache-
control=private&response-content-encoding=gzip&response-content-
type=text%2Fplain",
      "Type": "configure",
      "CommandId": "b6c402df-5c23-45b2-a707-ad20b9c5ae40",
      "CreatedAt": "2013-10-25T19:38:11+00:00",
      "ExitCode": 0
    },
    {
      "Status": "successful",
      "CompletedAt": "2013-10-25T19:31:08+00:00",
      "InstanceId": "67bf0da2-29ed-4217-990c-d895d51812b9",
      "AcknowledgedAt": "2013-10-25T19:29:01+00:00",
      "LogUrl": "https://s3.amazonaws.com/prod_stage-log/logs/2a90e862-
f974-42a6-9342-9a4f03468358?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Expires=1382731518&Signature=cxKYH08mCCd4Mv0yFb6ywebeQtA%3D&response-cache-
control=private&response-content-encoding=gzip&response-content-
type=text%2Fplain",
      "Type": "setup",
      "CommandId": "2a90e862-f974-42a6-9342-9a4f03468358",
      "CreatedAt": "2013-10-25T19:26:01+00:00",
      "ExitCode": 0
    }
  ]
}
```

3. LogUrl 値をブラウザにコピーしてログを表示します。

インスタンスに数個以上のコマンドがある場合は、`describe-commands` にパラメータを追加して、応答オブジェクトに含めるコマンドをフィルタできます。詳細については、「[describe-commands](#)」を参照してください。

インスタンスの Chef ログの表示

Note

このセクションのトピックは、Chef 12 に適用されます。Chef 11.10 以前リリースの Chef のログの場所については、「[Linux 用 Chef 11.10 以前のバージョンのトラブルシューティング](#)」を参照してください。

Linux インスタンス

AWS OpsWorks スタックは、各インスタンスの Chef ログを `/var/chef/runs` ディレクトリに保存します。(Linux インスタンスの場合、このディレクトリには、JSON 形式のファイルとして保存される、関連[データバッグ](#)も含まれます)。このディレクトリにアクセスするには、[sudo 権限](#)が必要です。各実行のログは、個別の実行のサブディレクトリ内にある `chef.log` という名前のファイルにあります。

AWS OpsWorks スタックは内部ログをインスタンスの `/var/log/aws/opsworks` フォルダに保存します。通常、この情報はトラブルシューティングにはあまり役立ちません。ただし、これらのログは AWS OpsWorks スタックのサポートに役立ちます。サービスに問題が発生した場合は、ログの提供を求められることがあります。Linux のログも、トラブルシューティングに役立つデータを提供する場合があります。

Windows インスタンス

エージェントログ

Windows インスタンスでは、OpsWorks ログは次のような `ProgramData` パスに保存されます。数値にはタイムスタンプが含まれます。

```
C:\ProgramData\OpsWorksAgent\var\logs\number
```

Note

デフォルトでは、`ProgramData` は非表示フォルダです。非表示を解除するには、[Folder Options] に移動します。[View] で、非表示のファイルを表示するオプションを選択します。

次の例では、エージェントは Windows インスタンスにログオンします。

Mode	LastWriteTime	Length	Name
----	-----	-----	----
-a---	5/24/2015 11:59 PM	127277	command.20150524.txt
-a---	5/25/2015 11:59 PM	546772	command.20150525.txt
-a---	5/26/2015 11:59 PM	551514	command.20150526.txt
-a---	5/27/2015 9:43 PM	495181	command.20150527.txt
-a---	5/24/2015 11:59 PM	24353	keepalive.20150524.txt
-a---	5/25/2015 11:59 PM	106232	keepalive.20150525.txt
-a---	5/26/2015 11:59 PM	106208	keepalive.20150526.txt
-a---	5/27/2015 8:54 PM	92593	keepalive.20150527.txt
-a---	5/24/2015 7:19 PM	3891	service.20150524.txt
-a---	5/27/2015 8:54 PM	1493	service.20150527.txt
-a---	5/24/2015 11:59 PM	112549	wire.20150524.txt
-a---	5/25/2015 11:59 PM	501501	wire.20150525.txt
-a---	5/26/2015 11:59 PM	499640	wire.20150526.txt
-a---	5/27/2015 8:54 PM	436870	wire.20150527.txt

Chef ログ

Windows インスタンスでは、Chef のログは次のような ProgramData のパスに保存されます。数値にはタイムスタンプが含まれます。

```
C:\ProgramData\OpsWorksAgent\var\commands\number
```

Note

このディレクトリには、最初の (OpsWorks 所有の) Chef 実行の出力のみが含まれます。

次の例は、Windows インスタンスの OpsWorks 所有 Chef ログを示しています。

Mode	LastWriteTime	Name
----	-----	----
d----	5/24/2015 7:23 PM	
		configure-7ecb5f47-7626-439b-877f-5e7cb40ab8be
d----	5/26/2015 8:30 PM	configure-8e74223b-d15d-4372-aeaa-
		a87b428ffc2b
d----	5/24/2015 6:34 PM	configure-
		c3980a1c-3d08-46eb-9bae-63514cee194b
d----	5/26/2015 8:32 PM	grant_remote_access-70dbf834-1bfa-4fce-
		b195-e50e85402f4c


```

d----          5/26/2015  10:30 PM          revoke_remote_access-1111fce9-843a-4b27-
b93f-ecc7c5e9e05b
d----          5/24/2015   7:21 PM          setup-754ec063-8b60-4cd4-
b6d7-0e89d7b7aa78
d----          5/26/2015   8:27 PM          setup-af5bed36-5afd-4115-
af35-5766f88bc039
d----          5/24/2015   6:32 PM          setup-d8abeffa-24d4-414b-
bfb1-4ad07319f358
d----          5/24/2015   7:13 PM          shutdown-c7130435-9b5c-4a95-
be17-6b988fc6cf9a
d----          5/26/2015   8:25 PM
sync_remote_users-64c79bdc-1f6f-4517-865b-23d2def4180c
d----          5/26/2015   8:48 PM
update_custom_cookbooks-2cc59a94-315b-414d-85eb-2bdea6d76c6a

```

ユーザーの Chef ログ

Chef の実行ログは、次の図のように、番号が付けられた Chef コマンドに関連する名前のフォルダの、logfile.txt という名前のファイルにあります。

```
C:/chef └─ runs └─ command-12345 └─ attribs.json └─ client.rb └─ logfile.txt
```

Chef ログの解釈

ほとんどの場合、ログの先頭には内部 Chef ログが含まれています。

```

# Logfile created on Thu Oct 17 17:25:12 +0000 2013 by logger.rb/1.2.6
[2013-10-17T17:25:12+00:00] INFO: *** Chef 11.4.4 ***
[2013-10-17T17:25:13+00:00] DEBUG: Building node object for php-app1.localdomain
[2013-10-17T17:25:13+00:00] DEBUG: Extracting run list from JSON attributes provided on
command line
[2013-10-17T17:25:13+00:00] INFO: Setting the run_list to
["opsworks_custom_cookbooks::load", "opsworks_custom_cookbooks::execute"] from JSON
[2013-10-17T17:25:13+00:00] DEBUG: Applying attributes from json file
[2013-10-17T17:25:13+00:00] DEBUG: Platform is amazon version 2013.03
[2013-10-17T17:25:13+00:00] INFO: Run List is [recipe[opsworks_custom_cookbooks::load],
recipe[opsworks_custom_cookbooks::execute]]
[2013-10-17T17:25:13+00:00] INFO: Run List expands to [opsworks_custom_cookbooks::load,
opsworks_custom_cookbooks::execute]
[2013-10-17T17:25:13+00:00] INFO: Starting Chef Run for php-app1.localdomain
[2013-10-17T17:25:13+00:00] INFO: Running start handlers
[2013-10-17T17:25:13+00:00] INFO: Start handlers complete.

```

```
[2013-10-17T17:25:13+00:00] DEBUG: No cheffignore file found at /opt/aws/opsworks/
releases/20131015111601_209/cookbooks/cheffignore no files will be ignored
[2013-10-17T17:25:13+00:00] DEBUG: Cookbooks to compile: ["gem_support", "packages",
"opsworks_bundler", "opsworks_rubygems", "ruby", "ruby_enterprise", "dependencies",
"opsworks_commons", "scm_helper", :opsworks_custom_cookbooks]
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook gem_support's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/gem_support/libraries/
current_gem_version.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook packages's library file: /opt/aws/
opsworks/releases/20131015111601_209/cookbooks/packages/libraries/packages.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook dependencies's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/dependencies/libraries/
current_gem_version.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook opsworks_commons's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/opsworks_commons/libraries/
activesupport_blank.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook opsworks_commons's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/opsworks_commons/libraries/
monkey_patch_chefgem_resource.rb
...
```

ファイルのこの部分は Chef の専門家に非常に役立ちます。ほとんどのコマンドは多数のレシピに関連していますが、実行リストに含まれるレシピは 2 つに限られます。この 2 つのレシピでは、他のすべての組み込みレシピとカスタムレシピをロードして実行するタスクを処理します。

通常、ファイルで最も関心を引く部分は末尾にあります。実行が正常に終了すると、次のような内容が表示されます。

```
...
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: STDERR:
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: ---- End output of /sbin/service mysqld
restart ----
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: Ran /sbin/service mysqld restart returned 0
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: service[mysql]: restarted successfully
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: Chef Run complete in 84.07096 seconds
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: cleaning the checksum cache
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: removing unused checksum cache file /var/chef/
cache/checksums/chef-file--tmp-chef-rendered-template20130611-4899-8wef7e-0
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: removing unused checksum cache file /var/chef/
cache/checksums/chef-file--tmp-chef-rendered-template20130611-4899-1xpwyb6-0
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: removing unused checksum cache file /var/chef/
cache/checksums/chef-file--etc-monit-conf
```

```
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: Running report handlers
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: Report handlers complete
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: Exiting
```

Note

エージェント CLI を使用して、実行中または実行後にログの末尾を表示できます。詳細については、「[Chef ログの表示](#)」を参照してください。

レシピが失敗した場合は、次のように、例外の後に Chef スタックトレースが続く ERROR レベルの出力を探す必要があります。

```
...
Please report any problems with the /usr/scripts/mysqlbug script!

[ OK ]
MySQL Daemon failed to start.
Starting mysqld: [FAILED]STDERR: 130611 15:07:55 [Warning] The syntax '--log-slow-queries' is deprecated and will be removed in a future release. Please use '--slow-query-log'/'--slow-query-log-file' instead.
130611 15:07:56 [Warning] The syntax '--log-slow-queries' is deprecated and will be removed in a future release. Please use '--slow-query-log'/'--slow-query-log-file' instead.
---- End output of /sbin/service mysqld start ----

/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/mixin/command.rb:184:in `handle_command_failures'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/mixin/command.rb:131:in `run_command'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/provider/service/init.rb:37:in `start_service'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/provider/service.rb:60:in `action_start'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/resource.rb:406:in `send'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/resource.rb:406:in `run_action'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/runner.rb:53:in `run_action'
```

```
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/runner.rb:89:in `converge'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/runner.rb:89:in `each'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/runner.rb:89:in `converge'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/resource_collection.rb:94:in `execute_each_resource'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/resource_collection/stepable_iterator.rb:116:in `call'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/resource_collection/stepable_iterator.rb:116:in
`call_iterator_block'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/resource_collection/stepable_iterator.rb:85:in `step'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/resource_collection/stepable_iterator.rb:104:in `iterate'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/resource_collection/stepable_iterator.rb:55:in
`each_with_index'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/resource_collection.rb:92:in `execute_each_resource'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/runner.rb:84:in `converge'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/client.rb:268:in `converge'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/client.rb:158:in `run'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/application/solo.rb:190:in `run_application'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/application/solo.rb:181:in `loop'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/application/solo.rb:181:in `run_application'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/./lib/chef/application.rb:62:in `run'
/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/chef-solo:25
/opt/aws/opsworks/current/bin/chef-solo:16:in `load'
/opt/aws/opsworks/current/bin/chef-solo:16
```

ファイルの末尾が Chef スタックトレースです。また、例外の直前の出力も調べる必要があります。この部分には `package not available` などのシステムエラーが含まれていることが多く、失敗の原因の特定に役立つ場合もあります。この例では、MySQL デーモンの起動に失敗しています。

Chef ログの一般的なエラー

Chef ログの一般的なエラーとその対処方法を以下に示します。

ログは見つかりませんでした

Chef の実行が開始されると、インスタンスは署名付き Amazon S3 URL を受け取ります。この URL により、Chef の実行が終了したときにウェブページでログを表示することができます。この URL は 2 時間で失効するため、Chef の実行に 2 時間以上かかると、Chef の実行中に何も問題が起きていない場合であっても、Amazon S3 サイトにログはアップロードされません。ログを作成するコマンドは成功しますが、ログが表示されるのはインスタンス上だけで、署名済み URL では表示されません。

ログが突然終了する

成功が示されることも、エラー情報が表示されることもなく、Chef ログが突然終了した場合、メモリ不足の状態が発生したために Chef がログを完了できなかったと考えられます。このような場合は、大規模なインスタンスでもう一度試してみてください。

クックブックまたはレシピが見つからない

Chef 実行時にクックブックキャッシュにないクックブックまたはレシピに遭遇した場合、次のような内容が表示されます。

```
DEBUG: Loading Recipe mycookbook::myrecipe via include_recipe
ERROR: Caught exception during execution of custom recipe: mycookbook::myrecipe:
       Cannot find a cookbook named mycookbook; did you forget to add metadata to a
       cookbook?
```

このエントリは、`mycookbook` というクックブックがクックブックキャッシュにないことを示しています。Chef 11.4 では、`metadata.rb` で依存関係を正しく宣言していない場合にも、このエラーが発生する可能性があります。

AWS OpsWorks スタックは、インスタンスのクックブックキャッシュからレシピを実行します。インスタンスの起動時に、リポジトリからこのキャッシュにクックブックがダウンロードされます。ただし、後でリポジトリ内のクックブックを変更しても、AWS OpsWorks スタックはオン

ラインインスタンスのキャッシュを自動的に更新しません。インスタンスの起動以降にクックブックの変更や新しいクックブックの追加を行った場合は、次の手順を実行します。

1. 変更がリポジトリにコミットされていることを確認します。
2. [Update Cookbooks スタックコマンド](#) を実行して、リポジトリの最新バージョンでクックブックキャッシュを更新します。

ローカルコマンドのエラー

Chef の `execute` リソースが指定されたコマンドを実行できない場合、次のような内容が表示されます。

```
DEBUG: ---- End output of ./configure --with-config-file-path=/ returned 2
ERROR: execute[PHP: ./configure] (/root/opsworks-agent/site-cookbooks/php-fpm/
recipes/install.rb line 48) had an error:
    ./configure --with-config-file-path=/
```

ログをスクロールアップすると、コマンドの `stderr` および `stdout` 出力が表示されます。これらの出力は、コマンドが失敗した理由を突き止める際に役立ちます。

パッケージのエラー

パッケージのインストールに失敗した場合、次のような内容が表示されます。

```
ERROR: package[zend-server-ce-php-5.3] (/root/opsworks-agent/site-cookbooks/
zend_server/recipes/install.rb line 20)
    had an error: apt-get -q -y --force-yes install zend-server-ce-php-5.3=5.0.4+b17
returned 100, expected 0
```

ログをスクロールアップすると、コマンドの `STDOUT` および `STDERROR` 出力が表示されます。これらの出力は、パッケージのインストールに失敗した理由を突き止める際に役立ちます。

AWS OpsWorks スタックエージェント CLI の使用

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソ

リューションに移行することを強くお勧めします。移行に関するご質問は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Note

エージェントの CLI は Linux インスタンスにのみ使用できます。

すべてのオンラインインスタンスで、AWS OpsWorks スタックは サービスと通信する エージェントをインストールします。次に、AWS OpsWorks スタックサービスは、ライフサイクルイベントが発生したときにインスタンスで Chef 実行を開始するなどのタスクを実行するコマンドをエージェントに送信します。Linux インスタンス用に、エージェントにはトラブルシューティングに役立つコマンドラインインターフェイス (CLI) が用意されています。エージェント CLI コマンドを実行するには、[SSH を使用してインスタンスに接続](#)します。その後、次のようなさまざまなタスクを行うエージェント CLI コマンドを実行します。

- レシピを実行する。
- Chef ログを表示する。
- [スタック設定およびデプロイメント JSON](#) を表示する。

インスタンスへの SSH 接続をセットアップする方法の詳細については、「[SSH でのログイン](#)」を参照してください。[SSH とスタックに対する sudo 権限](#)も必要です。

このセクションでは、エージェント CLI をトラブルシューティングに使用する方法について説明します。詳細およびコマンドリファレンスについては、「[AWS OpsWorks スタックエージェント CLI](#)」を参照してください。

トピック

- [レシピの実行](#)
- [Chef ログの表示](#)
- [スタック設定およびデプロイメント JSON の表示](#)

レシピの実行

エージェント CLI の [run_command](#) コマンドは、以前に実行したコマンドを再実行するようエージェントに指示します。トラブルシューティングに最も役立つコマンド

(`setup`、`configure`、`deploy`、`undeploy`) は、それぞれライフサイクルイベントに対応しています。これらのコマンドは、Chef 実行を開始して関連付けられたレシピを実行するようエージェントに指示します。

Note

`run_command` コマンドで実行されるのは、指定されたコマンドに関連付けられたレシピ (通常はライフサイクルイベントに関連付けられたレシピ) のグループに限られています。このコマンドを使用して特定のレシピを実行することはできません。指定した 1 つ以上のレシピを実行するには、[Execute Recipes スタックコマンド](#)を使用するか、同等の CLI または API アクション ([create-deployment](#) および [CreateDeployment](#)) を使用します。

`run_command` コマンドは、カスタムレシピ (特に、コンソールから直接トリガーできない、`Setup` および `Configure` の各ライフサイクルイベントに割り当てられたレシピ) をデバッグする際に役立ちます。`run_command` を使用すると、インスタンスを起動または停止しなくても、必要な頻度で特定のイベントのレシピを実行できます。

Note

AWS OpsWorks スタックは、クックブックリポジトリではなく、インスタンスのクックブックキャッシュからレシピを実行します。AWS OpsWorks スタックは、インスタンスの起動時にクックブックをこのキャッシュにダウンロードしますが、後でクックブックを変更してもオンラインインスタンスのキャッシュは自動的に更新されません。インスタンスの起動以降にクックブックを変更した場合は、[Update Cookbooks スタックコマンド](#)を実行して、リポジトリの最新バージョンでクックブックキャッシュを更新する必要があります。

エージェントは最新のコマンドだけをキャッシュに入れます。これらのコマンドを表示するには、キャッシュされたコマンドと各コマンドの実行時刻のリストを返す [list_commands](#) を実行します。

```
sudo opsworks-agent-cli list_commands
2013-02-26T19:08:26      setup
2013-02-26T19:12:01      configure
2013-02-26T19:12:05      configure
2013-02-26T19:22:12      deploy
```


最新のコマンドを再実行するには、次のコマンドを実行します。

```
sudo opsworks-agent-cli run_command
```

指定したコマンドの最新のインスタンスを再実行するには、次のコマンドを実行します。

```
sudo opsworks-agent-cli run_command command
```

たとえば、Setup レシピを再実行するには、次のコマンドを実行します。

```
sudo opsworks-agent-cli run_command setup
```

各コマンドには、そのコマンドの実行時のスタックとデプロイメントの状態を表す[スタック設定およびデプロイメント JSON](#) が関連付けられています。このデータは次のコマンドまでの間に変更される可能性があるため、コマンドの古いインスタンスは最新のデータとは多少異なるデータを使用している場合があります。コマンドの特定のインスタンスを再実行するには、`list_commands` の出力から時刻をコピーし、次のコマンドを実行します。

```
sudo opsworks-agent-cli run_command time
```

前の例はすべて、デフォルトの JSON (そのコマンド用にインストールされた JSON) を使用してコマンドを再実行しています。次のように、任意の JSON ファイルに対してコマンドを再実行できます。

```
sudo opsworks-agent-cli run_command -f /path/to/valid/json.file
```

Chef ログの表示

エージェント CLI の [show_log](#) コマンドは、指定されたログを表示します。このコマンドが完了すると、ユーザーはファイルの末尾を確認すると考えられます。そのため、`show_log` コマンドでは、一般にユーザーがエラー情報を確認する場所であるログの末尾が表示されます。スクロールアップすると、ログの前の部分を表示できます。

現在のコマンドのログを表示するには、次のコマンドを実行します。

```
sudo opsworks-agent-cli show_log
```

特定のコマンドのログを表示することもできますが、エージェントがキャッシュに入れるのは、最新 30 個のコマンドのログだけであることに注意してください。インスタンスのコマンドを一覧表示す

るには、キャッシュされたコマンドと各コマンドの実行時刻のリストを返す [list_commands](#) を実行します。例については、[レシピの実行](#)を参照してください。

直近に実行された特定のコマンドのログを表示するには、次のコマンドを実行します。

```
sudo opsworks-agent-cli show_log command
```

command パラメータは、setup、configure、deploy、undeploy、start、stop、または restart に設定できます。これらのコマンドのほとんどはライフサイクルイベントに対応しており、関連付けられているレシピを実行するようエージェントに指示します。

実行された特定のコマンドのログを表示するには、list_commands の出力から日付をコピーし、次のコマンドを実行します。

```
sudo opsworks-agent-cli show_log date
```

コマンドがまだ実行中の場合、show_log はログの現在の状態を表示します。

Note

show_log を使用してエラーや out-of-memory問題をトラブルシューティングする方法の 1 つは、実行中にログを次のように末尾に残すことです。

1. run_command を使用して、適切なライフサイクルイベントをトリガーします。詳細については、「[レシピの実行](#)」を参照してください。
2. show_log を繰り返し実行して、ログの書き込み中にログの末尾を表示します。

Chef がメモリ不足になったり、予期せず終了した場合は、ログが突然終了します。レシピが失敗した場合、ログは例外とスタックトレースで終了します。

スタック設定およびデプロイメント JSON の表示

レシピで使用するデータの大半は、[スタック設定およびデプロイメント JSON](#) から取得されます。JSON は、スタック設定、デプロイメント、およびユーザーが追加できるオプションのカスタム属性の詳細な記述である一連の Chef 属性を定義したものです。コマンドごとに、AWS OpsWorks Stacks はコマンドの時点でスタックとデプロイ状態を表す JSON をインストールします。詳細については、「[スタック設定およびデプロイメント属性](#)」を参照してください。

カスタムレシピでスタック設定およびデプロイ JSON からデータを取得した場合、JSON を調べることでデータを確認できます。スタック設定およびデプロイ JSON を表示する最も簡単な方法は、JSON オブジェクトの書式設定されたバージョンを表示する、エージェント CLI の [get_json](#) コマンドを実行することです。標準的な出力の最初の数行を次に示します。

```
{
  "opsworks": {
    "layers": {
      "php-app": {
        "id": "4a2a56c8-f909-4b39-81f8-556536d20648",
        "instances": {
          "php-app2": {
            "elastic_ip": null,
            "region": "us-west-2",
            "booted_at": "2013-02-26T20:41:10+00:00",
            "ip": "10.112.235.192",
            "aws_instance_id": "i-34037f06",
            "availability_zone": "us-west-2a",
            "instance_type": "c1.medium",
            "private_dns_name": "ip-10-252-0-203.us-west-2.compute.internal",
            "private_ip": "10.252.0.203",
            "created_at": "2013-02-26T20:39:39+00:00",
            "status": "online",
            "backends": 8,
            "public_dns_name": "ec2-10-112-235-192.us-west-2.compute.amazonaws.com"
          }
        }
      }
    }
  }
  ...
}
```

最新のスタック設定およびデプロイメント JSON を表示するには、次のコマンドを実行します。

```
sudo opsworks-agent-cli get_json
```

指定したコマンドの最新のスタック設定およびデプロイメント JSON を表示するには、次のコマンドを実行します。

```
sudo opsworks-agent-cli get_json command
```

`command` パラメータは、`setup`、`configure`、`deploy`、`undeploy`、`start`、`stop`、または `restart` に設定できます。これらのコマンドのほとんどはライフサイクルイベントに対応しており、関連付けられているレシピを実行するようエージェントに指示します。

特定のコマンド実行のスタック設定およびデプロイメント JSON を表示するには、次のようにコマンドの日付を指定します。

```
sudo opsworks-agent-cli get_json date
```

このコマンドを使用する最も簡単な方法は次のとおりです。

1. インスタンスで実行されたコマンドと各コマンドが実行された日付のリストを返す、`list_commands` を実行します。
2. 該当するコマンドの日付をコピーし、`get_json` の `date` 引数として使用します。

一般的なデバッグとトラブルシューティングの問題

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、一般的なデバッグやトラブルシューティングに関する問題とその解決方法について説明します。

トピック

- [AWS OpsWorks スタックのトラブルシューティング](#)
- [インスタンス登録の問題のトラブルシューティング](#)

AWS OpsWorks スタックのトラブルシューティング

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、

[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このセクションでは、よく発生する AWS OpsWorks スタックの問題とその解決策について説明します。

トピック

- [インスタンスを管理できない](#)
- [Chef の実行後にインスタンスが起動しない](#)
- [Elastic Load Balancing ヘルスチェックでレイヤーのインスタンスがすべて失敗する](#)
- [Elastic Load Balancing のロードバランサーと通信できない](#)
- [インポートしたオンプレミスインスタンスで再起動後にボリュームの設定を完了できない](#)
- [再起動後、EBS ボリュームが再アタッチされない](#)
- [AWS OpsWorks スタックセキュリティグループを削除できない](#)
- [AWS OpsWorks スタックセキュリティグループが誤って削除されました](#)
- [Chef のログが突然終了する](#)
- [クックブックが更新されない](#)
- [インスタンスが起動中のステータスでスタックする](#)
- [インスタンスが予期せずに再起動する](#)
- [opsworks-agent プロセスがインスタンスで実行されている](#)
- [予期しない execute_recipes コマンド](#)

インスタンスを管理できない

問題: 以前に管理可能であったインスタンスを管理できなくなった。場合によっては、次のようなエラーがログに表示される。

```
Aws::CharlieInstanceService::Errors::UnrecognizedClientException - The security token included in the request is invalid.
```

原因: インスタンスが依存している AWS OpsWorks 外部のリソースが編集または削除された場合に、この問題が発生する可能性があります。インスタンスとの通信を遮断する可能性があるリソースの変更例を次に示します。

- インスタンスに関連付けられた IAM ユーザーまたはロールが、AWS OpsWorks スタックの外部で誤って削除されました。これにより、インスタンスにインストールされている AWS OpsWorks エージェントと スタックサービス間の通信が失敗します AWS OpsWorks。インスタンスの存続中は、インスタンスに関連付けられたユーザーが必要です。
- インスタンスがオフラインの間にボリュームまたはストレージの設定を編集すると、インスタンスを管理できなくなる場合があります。
- EC2 インスタンスを ELB に手動で追加します。は、インスタンスがオンライン状態になったり、オンライン状態を離れたりするたびに AWS OpsWorks、割り当てられた Elastic Load Balancing ロードバランサー AWS OpsWorks を再設定します。は、有効なメンバーであることがわかっているインスタンス AWS OpsWorks のみと見なします。の外部、または他のプロセスによって追加されたインスタンスは削除されます。その他のインスタンスはすべて削除されます。

解決策: インスタンスが依存している IAM ユーザーまたはロールを削除しないようにします。可能であれば、依存しているインスタンスが実行中の間のみ、ボリュームまたはストレージの設定を編集します。を使用して AWS OpsWorks、AWS OpsWorks インスタンスのロードバランサーまたは EIP メンバーシップを管理します。--use-instance-profile ユーザーが誤って削除された場合に登録済みインスタンスの管理に関する問題が発生するのを回避するために、register コマンドにパラメータを追加して、代わりにインスタンスの組み込みインスタンスプロファイルを使用します。

Chef の実行後にインスタンスが起動しない

問題: カスタムクックブックを使用するように設定されている Chef 11.10 以前のスタックで、コミュニティクックブックを使用した Chef の実行後、インスタンスが起動しません。ログメッセージが、レシピがコンパイルに失敗したことを示すか (「Recipe Compile Error」)、依存関係を見つけられないことを理由にロードされない可能性があります。

原因: もっとも可能性の高い原因は、カスタムクックブックまたはコミュニティクックブックが、スタックの使用する Chef バージョンをサポートしていないことです。[apt](#) や [build-essential](#) などの一般的なコミュニティクックブックには、Chef 11.10 との互換性に問題があることが知られているものがあります。

解決策: カスタム Chef クックブックの使用設定が有効になっている AWS OpsWorks スタックでは、カスタムクックブックまたはコミュニティクックブックは常にスタックが使用する Chef のバージョンをサポートしている必要があります。コミュニティクックブックを、スタック設定で設定されている Chef のバージョンと互換性があるバージョンにピンしてください (クックブックのバージョンを特定のバージョンに設定します)。サポートされているクックブックバージョンを探すには、コンパイルに失敗したクックブックの変更ログを表示し、スタックでサポートされているクックブッ

クの最新バージョンのみを使用します。クックブックのバージョンをピンするには、カスタムクックブックリポジトリの Berkfile にバージョン番号を正確に指定します。例えば、cookbook 'build-essential', '= 3.2.0'。

Elastic Load Balancing ヘルスチェックでレイヤーのインスタンスがすべて失敗する

問題: アプリケーションサーバーレイヤーに Elastic Load Balancing ロードバランサーをアタッチしましたが、すべてのインスタンスがヘルスチェックで失敗します。

原因: Elastic Load Balancing ロードバランサーを作成するときに、インスタンスが正常であるかどうかを判断するためにロードバランサーが呼び出す ping のパスを指定する必要があります。アプリケーションに適切な ping パスを指定してください。デフォルト値は /index.html です。アプリケーションに index.html が含まれていない場合、適切なパスを指定する必要があります。指定しない場合、ヘルスチェックは失敗します。たとえば、「[Chef 11 Linux スタックの使用開始](#)」で使用されている SimplePHPApp は index.html を使用しません。これらのサーバーに適した ping パスは / です。

解決策: ロードバランサーの ping パスを編集します。詳細については、「[Elastic Load Balancing](#)」を参照してください。

Elastic Load Balancing のロードバランサーと通信できない

問題: Elastic Load Balancing ロードバランサーを作成し、アプリケーションサーバーレイヤーにアタッチしても、ロードバランサーの DNS 名または IP アドレスをクリックしてアプリケーションを実行すると、「リモートサーバーが応答していません」というエラーが表示されます。

原因: スタックがデフォルトの VPC で実行されている場合、そのリージョンに Elastic Load Balancing ロードバランサーを作成するときに、セキュリティグループを指定する必要があります。セキュリティグループには、IP アドレスからのインバウンドトラフィックを許可する進入ルールが必要です。デフォルトの VPC セキュリティグループを指定する場合、デフォルトの進入ルールはインバウンドトラフィックを許可しません。

解決策: 適切な IP アドレスからのインバウンドトラフィックを許可するようにセキュリティグループの進入ルールを編集します。

1. [Amazon EC2 コンソール](#) ナビゲーションペインの [Security Groups] (セキュリティグループ) をクリックします。
2. ロードバランサーのセキュリティグループを選択します。
3. [Inbound] (インバウンド) タブで [Edit] (編集) をクリックします。
4. [Source] を適切な CIDR に設定して進入ルールを追加します。

たとえば、[Anywhere] を指定すると、CIDR が 0.0.0.0/0 に設定され、任意の IP アドレスからの着信トラフィックを許可するようにロードバランサーに指示できます。

インポートしたオンプレミスインスタンスで再起動後にボリュームの設定を完了できない

問題： AWS OpsWorks スタックにインポートした EC2 インスタンスを再起動すると、インスタンスのステータスとして AWS OpsWorks スタックコンソールに失敗と表示されます。これは、Chef 11 または Chef 12 のインスタンスで発生する場合があります。

原因:AWS OpsWorks スタックでの設定プロセス中に、インスタンスにボリュームをアタッチできなかった可能性があります。1つの原因として、AWS OpsWorks コマンドの実行時に、setup スタックによってインスタンスのボリューム設定が上書きされていることが考えられます。

解決策: インスタンスの [Details] ページを開き、[Volumes] エリアでボリューム設定を確認します。ボリューム設定を変更できるのは、インスタンスのステータスが [stopped] の場合のみであることに注意してください。すべてのボリュームにマウントポイントと名前が指定されていることを確認します。インスタンスを再起動する前に、AWS OpsWorks スタックの設定で正しいマウントポイントを指定していることを確認します。

再起動後、EBS ボリュームが再アタッチされない

問題: Amazon EC2 コンソールを使用して Amazon EBS ボリュームをインスタンスにアタッチしても、インスタンスを再起動すると、ボリュームはアタッチされません。

原因： AWS OpsWorks スタックは、認識している Amazon EBS ボリュームのみを再アタッチできます。ただし、以下の制限があります。

- AWS OpsWorks スタックによって作成されたボリューム。
- [Resources] ページを使用して、明示的にスタックに登録したアカウントのボリューム。

解決策： AWS OpsWorks スタックコンソール、API、または CLI を使用してのみ Amazon EBS ボリュームを管理します。自分のアカウントの Amazon EBS ボリュームの 1 つをスタックで使用する場合は、スタックの [Resources] (リソースリソース) ページを使用してボリュームを登録し、インスタンスにアタッチします。詳細については、「[リソース管理](#)」を参照してください。

AWS OpsWorks スタックセキュリティグループを削除できない

問題： スタックを削除すると、削除できない AWS OpsWorks スタックセキュリティグループがいくつか残されます。

原因: セキュリティグループは、特定の順序で削除する必要があります。

解決策: まず、どのインスタンスもセキュリティグループを使用していないことを確認します。その後、以下のセキュリティグループが存在する場合は、次の順序で任意のセキュリティグループを削除します。

1. AWS-OpsWorks-Blank-Server
2. AWS-OpsWorks-モニタリングマスターサーバー
3. AWS-OpsWorks-DB-マスターサーバー
4. AWS-OpsWorks-Memcached-Server
5. AWS-OpsWorks-カスタムサーバー
6. AWS-OpsWorks-nodejs-App-Server
7. AWS-OpsWorks-PHP-App-Server
8. AWS-OpsWorks-Rails-App-Server
9. AWS-OpsWorks-Web-Server
10. AWS-OpsWorks-Default-Server
11. AWS-OpsWorks-LB-Server

AWS OpsWorks スタックセキュリティグループが誤って削除されました

問題: AWS OpsWorks スタックセキュリティグループの 1 つを削除したため、再作成する必要があります。

原因: これらのセキュリティグループは誤って削除されることがよくあります。

解決策: 再作成されたグループは、グループ名の大文字と小文字を含め、元のグループとまったく同じである必要があります。グループを手動で再作成する代わりに、AWS OpsWorks スタックでこのタスクを実行する方法をお勧めします。同じ AWS リージョンに新しいスタックを作成するだけで、AWS OpsWorks VPC が存在する場合は、削除したセキュリティグループを含むすべての組み込みセキュリティグループが自動的に再作成されます。その後、今後使用しないスタックを削除します。セキュリティグループは残ります。

Chef のログが突然終了する

問題: Chef のログが突然終了します。ログの最後には、正常な実行は示されず、例外やスタックトレースも表示されません。

原因: この動作は、通常、メモリが不十分であることによって発生します。

解決策: より大きいインスタンスを作成し、エージェント CLI の `run_command` コマンドを使用してレシピを再度実行します。詳細については、「[レシピの実行](#)」を参照してください。

クックブックが更新されない

問題: クックブックを更新しましたが、スタックのインスタンスはまだ古いレシピを実行しています。

原因: AWS OpsWorks スタックは各インスタンスでクックブックをキャッシュし、リポジトリではなくキャッシュからレシピを実行します。新しいインスタンスを起動すると、AWS OpsWorks スタックはリポジトリからインスタンスのキャッシュにクックブックをダウンロードします。ただし、その後でカスタムクックブックを変更した場合、AWS OpsWorks スタックはオンラインインスタンスのキャッシュを自動的に更新しません。

解決策: [クックブックの更新スタックコマンド](#)を実行して、AWS OpsWorks スタックにオンラインインスタンスのクックブックキャッシュの更新を明示的に指示します。

インスタンスが起動中のステータスでスタックする

問題: インスタンスを再起動したとき、または自動ヒーリングが自動的にインスタンスを再起動したときに、起動オペレーションが booting ステータスで停止します。

原因: この問題の 1 つの原因として考えられるのは、デフォルト VPC を含む VPC 設定です。インスタンスは、AWS OpsWorks スタックサービス、Amazon S3、パッケージ、クックブック、およびアプリケーションリポジトリと常に通信できる必要があります。例えば、デフォルト VPC からデフォルトゲートウェイを削除すると、インスタンスは AWS OpsWorks スタックサービスへの接続を失います。AWS OpsWorks スタックはインスタンス [エージェント](#) と通信できなくなるため、インスタンスは失敗として扱われ、[自動修復されます](#)。ただし、接続がないと、AWS OpsWorks スタックは修復されたインスタンスにインスタンスエージェントをインストールできません。エージェントがないと、AWS OpsWorks スタックはインスタンスで Setup レシピを実行できないため、スタートアップオペレーションが「ブート」ステータスを超えることはできません。

解決策: インスタンスが必要な接続を利用できるように、VPC の設定を変更します。

インスタンスが予期せずに再起動する

問題: 停止されたインスタンスが予期せずに再起動します。

原因 1: インスタンスの [\[auto healing\]](#) (オートヒーリング) を有効にしている場合、AWS OpsWorks スタックは関連付けられた Amazon EC2 インスタンスで定期的にヘルスチェックを実行し、異常な

インスタンスを再起動します。Amazon EC2 コンソール、AWS OpsWorks API、または CLI を使用してスタックマネージドインスタンスを停止または終了した場合、AWS OpsWorks スタックは通知されません。代わりに、停止されたインスタンスを異常と見なし、自動的に再起動します。

解決策: AWS OpsWorks スタックコンソール、API、または CLI を使用することによってのみインスタンスを管理します。AWS OpsWorks スタックを使用してインスタンスを停止または削除する場合、インスタンスは再起動されません。詳細については、「[24/7 インスタンスの手動による起動、停止、再起動](#)」および「[AWS OpsWorks スタックインスタンスの削除](#)」を参照してください。

原因 2: インスタンスは、さまざまな理由で失敗する可能性があります。自動ヒーリングが有効になっている場合、AWS OpsWorks スタックは失敗したインスタンスを自動的に再起動します。

解決策: これは通常の実行中のオペレーションです。障害が発生したインスタンスを AWS OpsWorks スタックで再起動したくない場合を除き、何もする必要はありません。自動的に再起動しない場合は、自動ヒーリングを無効にする必要があります。

opsworks-agent プロセスがインスタンスで実行されている

問題: インスタンスで複数の opsworks-agent プロセスが実行されています。例えば:

```
aws 24543 0.0 1.3 172360 53332 ? S Feb24 0:29 opsworks-agent: master 24543
aws 24545 0.1 2.0 208932 79224 ? S Feb24 22:02 opsworks-agent: keep_alive of master
24543
aws 24557 0.0 2.0 209012 79412 ? S Feb24 8:04 opsworks-agent: statistics of master
24543
aws 24559 0.0 2.2 216604 86992 ? S Feb24 4:14 opsworks-agent: process_command of master
24
```

原因: これらは、エージェントの通常の実行中のオペレーションに必要なとされる正当なプロセスです。これらは、デプロイの処理や、サービスへのキープアライブメッセージの送信などのタスクを実行します。

解決策: これは正常な動作です。これらのプロセスを停止しないでください。停止すると、エージェントの実行中のオペレーションに支障が生じます。

予期しない execute_recipes コマンド

問題: インスタンスの詳細ページの [Logs] (ログ) セクションに、予期しない execute_recipes コマンドが表示されます。予期しない execute_recipes コマンドは、[Stack] (スタック) ページや [Deployments] (デプロイ) ページにも表示される場合があります。

原因: この問題は通常、アクセス許可の変更によって発生します。ユーザーまたはグループの SSH または sudo アクセス許可を変更すると、AWS OpsWorks スタックは `execute_recipes` を実行してスタックのインスタンスを更新し、Configure イベントもトリガーします。`execute_recipes` コマンドのもう 1 つのソースは、インスタンスエージェントを更新している AWS OpsWorks スタックです。

解決策: これは通常のオペレーションです。何もする必要はありません。

`execute_recipes` コマンドが実行したアクションを確認するには、[Deployments] (デプロイ) ページに移動し、コマンドのタイムスタンプをクリックします。これによりコマンドの詳細ページが開き、実行されたキーレシピがリスト表示されます。例えば、次の詳細ページは、`execute_recipes` を実行して SSH アクセス許可を更新する `ssh_users` コマンドのためのページです。

Ran command `execute_recipes`

[Repeat](#)

Status	successful	User	OpsWorks
Created at	2014-02-21 17:15:40 UTC	Recipes	ssh_users
Completed at	2014-02-21 17:16:32 UTC		
Duration	00:00:52		

Hostname	SSH	Layers	Duration	Log
✓ php-app1		PHP App Server	00:00:52	show

すべての詳細を表示するには、コマンドの [Log] (ログ) 列で [show] (表示) をクリックすると、関連付けられた Chef ログが表示されます。ログで を検索します **Run List**。AWS OpsWorks スタックのメンテナンスレシピは の下にあります OpsWorks Custom Run List。たとえば、次は、前のスクリーンショットに示した `execute_recipes` コマンドの実行リストで、コマンドに関連付けられたすべてのレシピが表示されます。

```
[2014-02-21T17:16:30+00:00] INFO: OpsWorks Custom Run List:
["opsworks_stack_state_sync",
 "ssh_users", "test_suite", "opsworks_cleanup"]
```

インスタンス登録の問題のトラブルシューティング

このセクションでは、インスタンスの登録の一般的な問題とその解決方法を示します。

Note

登録の問題が発生した場合、register 引数を使用して --debug を実行します。これにより、デバッグの詳細情報を取得できます。

トピック

- [EC2User Is Not Authorized to Perform: ... \(EC2User に以下を実行する権限がありません: ...\)](#)
- [Credential Should Be Scoped to a Valid Region \(認証情報の範囲を有効なリージョンに限定する必要があります\)](#)

EC2User Is Not Authorized to Perform: ... (EC2User に以下を実行する権限がありません: ...)

問題: register コマンドで、次のような結果が返されます。

```
A client error (AccessDenied) occurred when calling the CreateGroup operation:
User: arn:aws:iam::123456789012:user/ImportEC2User is not authorized to
perform: iam:CreateGroup on resource:
arn:aws:iam::123456789012:group/AWS/OpsWorks/OpsWorks-b583ce55-1d01-4695-b3e5-
ee19257d1911
```

原因: register コマンドは、必要なアクセス権限を付与しないユーザーの認証情報で実行されています。ユーザーのポリシーでは、数あるアクションの中で iam:CreateGroup アクションが許可されている必要があります。

ソリューション: register を必要なアクセス権限が付与された IAM ユーザー認証情報で指定します。詳細については、「[AWS CLIのインストールおよび設定](#)」を参照してください。

Credential Should Be Scoped to a Valid Region (認証情報の範囲を有効なリージョンに限定する必要があります)

問題: register コマンドにより、以下のような出力が返されます。

```
A client error (InvalidSignatureException) occurred when calling the
DescribeStacks operation: Credential should be scoped to a valid region, not 'cn-
north-1'.
```

原因: コマンドのリージョンは、有効な AWS OpsWorks スタックリージョンである必要があります。サポートされているリージョンのリストについては、「[リージョンのサポート](#)」を参照してください。通常、このエラーは、次のいずれかの原因により発生します。

- スタックが別のリージョンにあり、スタックのリージョンをコマンドの `--region` 引数に割り当てた場合。

スタックリージョンを指定する必要はありません。AWS OpsWorks スタックはスタック ID から自動的に決定します。

- `--region` 引数を省略したことにより、暗黙的にデフォルトのリージョンが指定されたものの、デフォルトのリージョンが AWS OpsWorks スタックでサポートされていません。

解決策: サポートされている AWS OpsWorks スタックリージョン `--region` に明示的に設定するか、`config` ファイルを編集してデフォルトのリージョンをサポートされている AWS OpsWorks スタックリージョンに変更します AWS CLI。詳細については、「[AWS コマンドラインインターフェイスの設定](#)」を参照してください。

AWS OpsWorks スタックエージェント CLI

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

Note

この機能は、Linux インスタンスでのみご利用いただけます。

が使用するエージェント AWS OpsWorks スタックは、すべてのインスタンスに をインストールし、コマンドラインインターフェイス () を公開します CLI。 [SSH を使用してインスタンスにログイン](#) する場合は、CLIを使用して以下を実行できます。

- Chef 実行のログファイルにアクセスする。
- アクセス AWS OpsWorks スタックコマンド。
- Chef レシピを手動で実行する。
- インスタンスレポートを表示する。
- エージェントレポートを表示する。
- スタック設定およびデプロイ属性の制限されたセットを表示する。

Important

エージェントCLIコマンドは、ルートとして、または `sudo` を使用してのみ実行できます。

基本的なコマンド構文は次のとおりです。

```
sudo opsworks-agent-cli [--help] [command [activity] [date]]
```

4 つの引数は次のとおりです。

help (ヘルプ)

(オプション) 単独で使用した場合は、使用可能なコマンドの簡単な概要が表示されます。コマンドと共に `help` を使用した場合は、そのコマンドの説明が表示されます。

コマンド

(オプション) エージェントCLIコマンド。次のいずれかに設定する必要があります。

- [agent_report](#)
- [get_json](#)
- [instance_report](#)
- [list_commands](#)
- [run_command](#)
- [show_log](#)
- [stack_state](#)

activity

(オプション) 特定の を指定するためのいくつかのコマンドを含む引数として使用されます AWS OpsWorks スタックアクティビティ: `setup`、`configure`、`deploy`、`undeploy`、`start`、`stop`、または `restart`。

date

(オプション) 特定の を指定するためのいくつかのコマンドを含む引数として使用されます AWS OpsWorks スタックコマンドの実行。でコマンドが実行されたタイムスタンプに日付を設定して、コマンドの実行を指定します。 `yyyy-mm-ddThh:mm:ss` 一重引用符を含む 形式。たとえば、2013 年 2 月 5 日火曜日 10 時 31 分 55 秒の場合、`'2013-02-05T10:31:55'` を使用します。特定の をいつ決定するかを決定するには AWS OpsWorks スタックコマンドが実行されました。 を実行します [list_commands](#)。

Note

エージェントが同じ を実行した場合 AWS OpsWorks アクティビティを複数回スタックする場合は、アクティビティと実行時刻の両方を指定することで、特定の実行を選択できます。アクティビティを指定して時間を省略すると、エージェントCLIコマンドはアクティビティの最新の実行に対して動作します。両方の引数を省略すると、エージェントCLIコマンドは最新のアクティビティで動作します。

以下のセクションでは、各コマンドと関連する引数について説明します。簡潔に示すために、構文セクションでは、どのコマンドでも使用できる省略可能な `--help` オプションを省略しています。

トピック

- [agent_report](#)
- [get_json](#)
- [instance_report](#)
- [list_commands](#)
- [run_command](#)
- [show_log](#)
- [stack_state](#)

agent_report

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

エージェントレポートを返します。

```
sudo opsworks-agent-cli agent_report
```

configure アクティビティを直近に実行したインスタンスの出力例を次に示します。

```
$ sudo opsworks-agent-cli agent_report
```

```
AWS OpsWorks Instance Agent State Report:
```

```
Last activity was a "configure" on 2015-12-01 18:19:23 UTC  
Agent Status: The AWS OpsWorks agent is running as PID 30998  
Agent Version: 4004-20151201152533, up to date
```

get_json

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

Chef 実行に関する情報をJSONオブジェクトとして返します。

```
sudo opsworks-agent-cli get_json [activity] [date] [-i | --internal | --no-i | --no-internal]
```

デフォルトでは、`get_json` は最後に実行された Chef の実行について、ユーザーが提供した情報を表示します。特定の情報セットを指定するには、次のオプションを使用します。

activity

最後に指定されたアクティビティに関連付けられた Chef の実行に関する情報を表示します。有効なアクティビティのリストを取得するには、[list_commands](#) を実行します。

date

指定されたタイムスタンプに対して実行されたアクティビティに関連付けられている Chef の実行についての情報を表示します。有効なタイムスタンプのリストを取得するには、[list_commands](#) を実行します。

-i, --internal

次の情報を表示します。AWS OpsWorks スタックは、Chef 実行のために内部的に を使用しません。

--no-i, --no-internal

Chef を実行するためにユーザーが提供した情報を明示的に表示します。これは、それ以外の方法で指定されない場合のデフォルトです。

Note

Chef 12 Linux インスタンスの場合、このコマンドを実行すると、インスタンスのスタック設定やデプロイ属性など、有効な情報が返されます。ただし、より詳細な情報を取得するには、AWS OpsWorks スタックはインスタンスに を作成します。詳細については、「[AWS OpsWorks スタックデータバグリファレンス](#)」を参照してください。

次の出力例は、最新の `configure` アクティビティに対して最後に実行された Chef の実行についてユーザーが提供した情報を示しています。

```
$ sudo opsworks-agent-cli get_json configure  
  
{
```

```
"run_list": [  
  "recipe[opsworks_cookbook_demo::configure]"  
]  
}
```

次の出力例は、次の情報を示しています。AWS OpsWorks スタックは、指定されたタイムスタンプに対して実行される Chef 実行に内部的に を使用します。

```
$ sudo opsworks-agent-cli get_json 2015-12-01T18:20:24 -i  
  
{  
  "aws_opsworks_agent": {  
    "version": "4004-20151201152533",  
    "valid_client_activities": [  
      "reboot",  
      "stop",  
      "deploy",  
      "grant_remote_access",  
      "revoke_remote_access",  
      "update_agent",  
      "setup",  
      "configure",  
      "update_dependencies",  
      "install_dependencies",  
      "update_custom_cookbooks",  
      "execute_recipes",  
      "sync_remote_users"  
    ],  
    "command": {  
      "type": "configure",  
      "args": {  
        "app_ids": [  
  
        ]  
      },  
      "sent_at": "2015-12-01T18:19:23+00:00",  
      "command_id": "5c2113f3-c6d5-40eb-bcfa-77da2885eeEX",  
      "iam_user_arn": null,  
      "instance_id": "cfdaa716-42fe-4e3b-9762-fef184ddd8EX"  
    },  
    "resources": {  
      "apps": [  

```

```
],
"layers": [
  {
    "layer_id": "93f50d83-1e73-45c4-840a-0d4f07cda1EX",
    "name": "MyCookbooksDemoLayer",
    "packages": [

    ],
    "shortname": "cookbooks-demo",
    "type": "custom",
    "volume_configurations": [

    ]
  }
],
"instances": [
  {
    "ami_id": "ami-d93622EX",
    "architecture": "x86_64",
    "auto_scaling_type": null,
    "availability_zone": "us-west-2a",
    "created_at": "2015-11-18T00:21:05+00:00",
    "ebs_optimized": false,
    "ec2_instance_id": "i-a480e960",
    "elastic_ip": null,
    "hostname": "cookbooks-demo1",
    "instance_id": "cfdaa716-42fe-4e3b-9762-fef184ddd8EX",
    "instance_type": "c3.large",
    "layer_ids": [
      "93f50d83-1e73-45c4-840a-0d4f07cda1EX"
    ],
    "os": "Amazon Linux 2015.09",
    "private_dns": "ip-192-0-2-0.us-west-2.compute.internal",
    "private_ip": "10.122.69.33",
    "public_dns": "ec2-203-0-113-0.us-west-2.compute.amazonaws.com",
    "public_ip": "192.0.2.0",
    "root_device_type": "ebs",
    "root_device_volume_id": "vol-f6f7e8EX",
    "ssh_host_dsa_key_fingerprint": "f2:...:15",
    "ssh_host_dsa_key_public": "ssh-dss AAAAB3Nz...a8vMbqA=",
    "ssh_host_rsa_key_fingerprint": "0a:...:96",
    "ssh_host_rsa_key_public": "ssh-rsa AAAAB3Nz...yhPanvo7",
    "status": "online",
    "subnet_id": null,
```

```
    "virtualization_type": "paravirtual",
    "infrastructure_class": "ec2",
    "ssh_host_dsa_key_private": "-----BEGIN DSA PRIVATE KEY-----
\nMIIDVwIB...g50tgQ==\n-----END DSA PRIVATE KEY-----\n",
    "ssh_host_rsa_key_private": "-----BEGIN RSA PRIVATE KEY-----
\nMIIIEowIB...78kprtIw\n-----END RSA PRIVATE KEY-----\n"
  }
],
"users": [

],
"elastic_load_balancers": [

],
"rds_db_instances": [

],
"stack": {
  "arn": "arn:aws:opsworks:us-west-2:80398EXAMPLE:stack/040c3def-b2b4-4489-bb1b-
e08425886fEX/",
  "custom_cookbooks_source": {
    "type": "s3",
    "url": "https://s3.amazonaws.com/amzn-s3-demo-bucket/opsworks-cookbook-
demo.tar.gz",
    "username": "AKIAJUQN...WG644EXA",
    "password": "05v+4Zz+...rcKbFTJu",
    "ssh_key": null,
    "revision": null
  },
  "name": "MyCookbooksDemoStack",
  "region": "us-west-2",
  "stack_id": "040c3def-b2b4-4489-bb1b-e08425886fEX",
  "use_custom_cookbooks": true,
  "vpc_id": null
},
"ecs_clusters": [

],
"volumes": [

]
},
"chef": {
  "customer_recipes": [
```

```
    "opsworks_cookbook_demo::configure"  
  ],  
  "customer_json": "e30=\n",  
  "customer_data_bags": "e30=\n"  
}  
}  
}
```

instance_report

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

拡張インスタンスレポートを返します。

```
sudo opsworks-agent-cli instance_report
```

インスタンスの出力例を次に示します。

```
$ sudo opsworks-agent-cli instance_report
```

```
AWS OpsWorks Instance Agent State Report:
```

```
Last activity was a "configure" on 2015-12-01 18:19:23 UTC  
Agent Status: The AWS OpsWorks agent is running as PID 30998  
Agent Version: 4004-20151201152533, up to date  
OpsWorks Stack: MyCookbooksDemoStack  
OpsWorks Layers: MyCookbooksDemoLayer  
OpsWorks Instance: cookbooks-demo1  
EC2 Instance ID: i-a480e9EX  
EC2 Instance Type: c3.large  
Architecture: x86_64  
Total Memory: 3.84 Gb  
CPU: 2x Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz
```

Location:

EC2 Region: us-west-2

EC2 Availability Zone: us-west-2a

Networking:

Public IP: 192.0.2.0

Private IP: 198.51.100.0

list_commands

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#)ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

このインスタンスで実行された各アクティビティの時刻を表示します。これらの時間は、他の agent-CLI コマンドで特定の実行を指定するために使用できます。

```
sudo opsworks-agent-cli list_commands [activity] [date]
```

設定、セットアップ、およびカスタムクックブックの更新の各アクティビティを実行したインスタンスの出力例を次に示します。

```
$ sudo opsworks-agent-cli list_commands  
  
2015-11-24T21:00:28      update_custom_cookbooks  
2015-12-01T18:19:09      setup  
2015-12-01T18:20:24      configure
```

run_command

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

を実行します。AWS OpsWorks スタックコマンド。 の実行に必要な情報を含む Chef 実行リストを含むJSONファイルです。AWS OpsWorks スタックアクティビティ (セットアップ、設定、デプロイなど)。run_command コマンドにより、ログエントリが生成されます。ログエントリを表示するには、[show_log](#) を実行します。このオプションは開発のみを目的としているため、AWS OpsWorks スタックは変更を追跡しません。

```
sudo opsworks-agent-cli run_command [activity] [date] [path/to/valid/json.file]
```

デフォルトでは、は最新の run_command を実行します。AWS OpsWorks スタックコマンド。特定のコマンドを指定するには、次のオプションを使用します。

activity

指定された を実行する AWS OpsWorks スタックコマンド:

setup、configure、deploy、undeploy、start、stop、または restart。

date

指定されたタイムスタンプで実行されたAWS OpsWorks コマンドを実行します。有効なタイムスタンプのリストを取得するには、[list_commands](#) を実行します。

file

指定されたコマンドJSONファイルを実行します。コマンドのファイルパスを取得するには、[get_json](#) を実行します。

次のインスタンスの出力例では configure コマンドを実行しています。

```
$ sudo opsworks-agent-cli run_command configure
```



```
[2015-12-02 16:52:53] INFO [opsworks-agent(21970)]: About to re-run 'configure' from
2015-12-01T18:20:24
...
[2015-12-02 16:53:02] INFO [opsworks-agent(21970)]: Finished Chef run with exitcode 0
```

show_log

Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) ください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

コマンドのログファイルを返します。

```
sudo opsworks-agent-cli show_log [activity] [date]
```

デフォルトでは、show_log は最新のログファイルの末尾を表示します。特定のコマンドを指定するには、次のオプションを使用します。

activity

指定されたアクティビティのログファイルを表示します。

date

指定されたタイムスタンプに実行されたアクティビティのログファイルを表示します。有効なタイムスタンプのリストを取得するには、[list_commands](#) を実行します。

次の出力例は、最新のログを示しています。

```
$ sudo opsworks-agent-cli show_log

[2015-12-02T16:52:59+00:00] INFO: Storing updated cookbooks/opsworks_cookbook_demo/
opsworks-cookbook-demo.tar.gz in the cache.
...
```

```
[2015-12-02T16:52:59+00:00] INFO: Report handlers complete
```

stack_state

⚠ Important

- AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[お問い合わせ](#) してください。AWS サポート でのチーム [AWS re:Post](#) または through [AWS プレミアムサポート](#)。

次の情報を表示します。AWS OpsWorks スタックは、最新の Chef 実行のために内部的に `opsworks-agent-cli` を使用します。

```
opsworks-agent-cli stack_state
```

ℹ Note

Chef 12 Linux インスタンスの場合、このコマンドを実行すると、インスタンスのスタック設定やデプロイ属性など、有効な情報が返されます。ただし、より詳細な情報を取得するには、AWS OpsWorks スタックはインスタンスに `opsworks-agent-cli` を作成します。詳細については、「[AWS OpsWorks スタックデータバググリファレンス](#)」を参照してください。

インスタンスの出力例を次に示します。

```
$ sudo opsworks-agent-cli stack_state

{
  "last_command": {
    "sent_at": "2015-12-01T18:19:23+00:00",
    "activity": "configure"
  },
  "instance": {
    "ami_id": "ami-d93622EX",
    "architecture": "x86_64",
    "auto_scaling_type": null,
```

```
"availability_zone": "us-west-2a",
"created_at": "2015-11-18T00:21:05+00:00",
"ebs_optimized": false,
"ec2_instance_id": "i-a480e9EX",
"elastic_ip": null,
"hostname": "cookbooks-demo1",
"instance_id": "cfdaa716-42fe-4e3b-9762-fef184ddd8EX",
"instance_type": "c3.large",
"layer_ids": [
  "93f50d83-1e73-45c4-840a-0d4f07cda1EX"
],
"os": "Amazon Linux 2015.09",
"private_dns": "ip-192-0-2-0.us-west-2.compute.internal",
"private_ip": "10.122.69.33",
"public_dns": "ec2-203-0-113-0.us-west-2.compute.amazonaws.com",
"public_ip": "192.0.2.0",
"root_device_type": "ebs",
"root_device_volume_id": "vol-f6f7e8EX",
"ssh_host_dsa_key_fingerprint": "f2:...:15",
"ssh_host_dsa_key_public": "ssh-dss AAAAB3Nz...a8vMbqA=",
"ssh_host_rsa_key_fingerprint": "0a:...:96",
"ssh_host_rsa_key_public": "ssh-rsa AAAAB3Nz...yhPanvo7",
"status": "online",
"subnet_id": null,
"virtualization_type": "paravirtual",
"infrastructure_class": "ec2",
"ssh_host_dsa_key_private": "-----BEGIN DSA PRIVATE KEY-----\nMIIDVwIB...g50tgQ==
\n-----END DSA PRIVATE KEY-----\n",
"ssh_host_rsa_key_private": "-----BEGIN RSA PRIVATE KEY-----\nMIIEowIB...78kprtIw
\n-----END RSA PRIVATE KEY-----\n"
},
"layers": [
  {
    "layer_id": "93f50d83-1e73-45c4-840a-0d4f07cda1EX",
    "name": "MyCookbooksDemoLayer",
    "packages": [

    ],
    "shortname": "cookbooks-demo",
    "type": "custom",
    "volume_configurations": [

    ]
  }
]
```

```
],
"applications": null,
"stack": {
  "arn": "arn:aws:opsworks:us-west-2:80398EXAMPLE:stack/040c3def-b2b4-4489-bb1b-
e08425886fEX/",
  "custom_cookbooks_source": {
    "type": "s3",
    "url": "https://s3.amazonaws.com/amzn-s3-demo-bucket/opsworks-cookbook-
demo.tar.gz",
    "username": "AKIAJUQN...WG644EXA",
    "password": "05v+4Zz+...rcKbFTJu",
    "ssh_key": null,
    "revision": null
  },
  "name": "MyCookbooksDemoStack",
  "region": "us-west-2",
  "stack_id": "040c3def-b2b4-4489-bb1b-e08425886fEX",
  "use_custom_cookbooks": true,
  "vpc_id": null
},
"agent": {
  "valid_activities": [
    "reboot",
    "stop",
    "deploy",
    "grant_remote_access",
    "revoke_remote_access",
    "update_agent",
    "setup",
    "configure",
    "update_dependencies",
    "install_dependencies",
    "update_custom_cookbooks",
    "execute_recipes",
    "sync_remote_users"
  ]
}
}
```

AWS OpsWorks スタックデータバグリファレンス

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックは、Chef データバグコンテンツとしてレシピにさまざまな設定を公開します。このリファレンスでは、このデータバグコンテンツを一覧で示します。

データバグは Chef の概念です。データバグはインスタンスで JSON データとして保存されるグローバル変数です。JSON データは Chef からアクセス可能です。例えば、データバグは、アプリケーションのソース URL、インスタンスのホスト名、関連するスタックの VPC 識別子などのグローバル変数を保存できます。AWS OpsWorks スタックは、各スタックのインスタンスにデータバグを保存します。Linux インスタンスでは、AWS OpsWorks スタックはデータバグを `/var/chef/runs/run-ID/data_bags` ディレクトリに保存します。Windows インスタンスでは、スタックはデータバグを `drive:\chef\runs\run-id\data_bags` ディレクトリに保存します。いずれの場合も、*run-ID* は、AWS OpsWorks スタックがインスタンスで実行される各 Chef に割り当てられている一意の ID です。これらのディレクトリにはデータバグ (サブディレクトリ) のセットが含まれています。各データバグには、ゼロ個以上のデータバグアイテムが含まれ、これらはデータバグコンテンツのセットを含む JSON 形式のファイルです。

Note

AWS OpsWorks スタックは、暗号化されたデータバグをサポートしていません。パスワードや証明書などの機密データを暗号化された形式で保存するには、プライベート S3 バケットに保存することをお勧めします。これで、[Amazon SDK for Ruby](#) を使用するカスタムレシピを作成できます。例については、[SDK for Ruby の使用](#) を参照してください。

データバグのコンテンツには、次のいずれかを含めることができます。

- 標準の Ruby 構文に従う String コンテンツ。一重引用符または二重引用符を使用できますが、一部の特殊文字を含む文字列は二重引用符が必要です。詳細については、[Ruby](#) のドキュメントのサイトを参照してください。
- [Boolean] (ブール値) コンテンツのどちらか true または false (引用符なし)。
- [Number] (数値) などの整数または10進数のいずれかコンテンツ、4 または 2.5 (引用符なし)。
- [List] (リスト) コンテンツは、角カッコ (引用符なし) で囲まれたカンマ区切り値の形式をとります。['80', '443']
- "my-app": {"elastic_ip": null, ...} などの追加のデータバッグコンテンツを含む [JSON objects] (JSON オブジェクト)。

Chef レシピは、Chef 検索を通じて、または直接データバッグ、データバッグ項目、およびデータバッグコンテンツにアクセスできます。以下では、両方のアクセス方法を使用する方法を説明します (ただし、Chef 検索をお勧めします)。

Chef 検索からデータバッグにアクセスするには、[検索](#)方法を使用して目的の検索インデックスを指定します。AWS OpsWorks スタックには、次の検索インデックスがあります。

- スタック用のデプロイ済みアプリケーションのセットを表す [aws_opsworks_app](#)。
- スタックで実行されたコマンドのセットを表す [aws_opsworks_command](#)。
- スタック用のクラスターインスタンスのセットを表す [\[aws_opsworks_ecs_cluster\]](#) Amazon Elastic Container Service (Amazon ECS)。
- スタック用の一連の Elastic Load Balancing ロードバランサーを表す [\[aws_opsworks_elastic_load_balancer\]](#)。
- スタック用のインスタンスのセットを表す [aws_opsworks_instance](#)。
- スタック用のレイヤーのセットを表す [aws_opsworks_layer](#)。
- スタック用の Amazon Relational Database Service (Amazon RDS) インスタンスのセットを表す [\[aws_opsworks_rds_db_instance\]](#)。
- スタックを表す [aws_opsworks_stack](#)。
- スタック用のユーザーのセットを表す [aws_opsworks_user](#)。

検索インデックス名がわかれば、その検索インデックスのデータバッグのコンテンツにアクセスできます。例えば、次のレシピコードは [aws_opsworks_app](#) 検索インデックスを使用して、[aws_opsworks_app](#) データバッグ ([aws_opsworks_app](#)ディレクトリ) の最初のデータバッグ項目 (最初の JSON ファイル) のコンテンツを取得します。次に、コードは 2 つのメッセージを Chef

ログに書き込みます。1 つはアプリケーションの短縮名のデータバッグコンテンツ (JSON ファイルの文字列) で、もう 1 つはアプリケーションのソース URL データバッグコンテンツ (JSON ファイルの別の文字列) です。

```
app = search("aws_opsworks_app").first
Chef::Log.info("***** The app's short name is '#{app['shortname']}' *****")
Chef::Log.info("***** The app's URL is '#{app['app_source']['url']}' *****")
```

ここで、`['shortname']` と `['app_source']['url']` は、対応する JSON ファイルの次のデータバッグコンテンツを指定します。

```
{
  ...
  "shortname": "mylinuxdemoapp",
  ...
  "app_source": {
    ...
    "url": "https://s3.amazonaws.com/opsworks-demo-assets/opsworks-linux-demo-
nodejs.tar.gz",
  },
  ...
}
```

検索できるデータバッグコンテンツのリストについては、このセクションのリファレンストピックを参照してください。

また、データバッグのデータバッグ項目のセットを反復的に処理することもできます。たとえば、次のレシピコードは前の例と似ています。複数のデータバッグ項目があると、データバッグの各データバッグ項目を反復処理します。

```
search("aws_opsworks_app").each do |app|
  Chef::Log.info("***** The app's short name is '#{app['shortname']}' *****")
  Chef::Log.info("***** The app's URL is '#{app['app_source']['url']}'
*****")
end
```

特定のデータバッグコンテンツが存在することがわかっている場合は、次の構文で対応するデータバッグ項目を検索できます。

```
search("search_index", "key:value").first
```

例えば、次のレシピコードでは、`aws_opsworks_app` 検索インデックスを使用して、`mylinuxdemoapp` のアプリケーションの短縮名を含むデータバッグを検索します。次に、対応するアプリケーションの短縮名とソース URL とともに、データバッグ項目のコンテンツを使用して Chef ログにメッセージを書き込みます。

```
app = search("aws_opsworks_app", "shortname:mylinuxdemoapp").first
Chef::Log.info("***** For the app with the short name '#{app['shortname']}', the
  app's URL is '#{app['app_source']['url']}' *****")
```

`aws_opsworks_instance` 検索インデックスのみについて、`self:true` を指定して、レシピを実行中のインスタンスを表すことができます。次のレシピコードは、対応するデータバッグ項目の内容を使用して、対応するインスタンスの AWS OpsWorks スタック生成 ID とオペレーティングシステムで Chef ログにメッセージを書き込みます。

```
instance = search("aws_opsworks_instance", "self:true").first
Chef::Log.info("***** For instance '#{instance['instance_id']}', the instance's
  operating system is '#{instance['os']}' *****")
```

Chef 検索を使用してデータバッグ、データバッグ項目、データバッグコンテンツにアクセスする代わりに、それらに直接アクセスできます。そのためには、それぞれ [data_bag](#) および [data_bag_item](#) メソッドを使用して、データバッグおよびデータバッグ項目にアクセスします。たとえば、次のレシピコードは前の例と同じことを行いますが、単一のデータバッグ項目にアクセスしてから、複数の項目がある場合は複数のデータ項目にアクセスします。

```
# Syntax: data_bag_item("the data bag name", "the file name in the data bag without the
  file extension")
app = data_bag_item("aws_opsworks_app", "mylinuxdemoapp")
Chef::Log.info("***** The app's short name is '#{app['shortname']}' *****")
Chef::Log.info("***** The app's URL is '#{app['app_source']['url']}' *****")

data_bag("aws_opsworks_app").each do |data_bag_item|
  app = data_bag_item("aws_opsworks_app", data_bag_item)
  Chef::Log.info("***** The app's short name is '#{app['shortname']}' *****")
  Chef::Log.info("***** The app's URL is '#{app['app_source']['url']}'
  *****")
end
```

これらの2つの方法のうち、Chef 検索を使用することをお勧めします。このガイドの関連する例では、すべてこの方法を示します。

トピック

- [アプリケーションデータバッグ \(aws_opsworks_app\)](#)
- [コマンドデータバッグ \(aws_opsworks_command\)](#)
- [Amazon ECS クラスターデータバッグ \(aws_opsworks_ecs_cluster\)](#)
- [Elastic Load Balancing データバッグ \(aws_opsworks_elastic_load_balancer\)](#)
- [インスタンスデータバッグ \(aws_opsworks_instance\)](#)
- [レイヤーデータバッグ \(aws_opsworks_layer\)](#)
- [Amazon RDS データバッグ \(aws_opsworks_rds_db_instance\)](#)
- [スタックデータバッグ \(aws_opsworks_stack\)](#)
- [ユーザーデータバッグ \(aws_opsworks_user\)](#)

アプリケーションデータバッグ (aws_opsworks_app)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

[Deploy イベント](#) または [Execute Recipes スタックコマンド](#) では、アプリケーションの設定を表します。

以下の例は、Chef 検索を使用して単一のデータバッグ項目を通じて検索を実行し、次に複数のデータバッグ項目を使用して、アプリケーションの短縮名とソース URL とともに Chef ログにメッセージを書き込む方法を示しています。

```
app = search("aws_opsworks_app").first
Chef::Log.info("***** The app's short name is '#{app['shortname']}' *****")
Chef::Log.info("***** The app's URL is '#{app['app_source']['url']}' *****")

search("aws_opsworks_app").each do |app|
  Chef::Log.info("***** The app's short name is '#{app['shortname']}' *****")
end
```

```

Chef::Log.info("***** The app's URL is '#{app['app_source']['url']}'
*****")
end

```

app_id	app_source	data_sources
デプロイ	attributes	domains
enable_ssl	環境	name
shortname	ssl_configuration	type

app_id

アプリケーション ID (文字列)。アプリケーションを識別する GUID。

app_source

AWS OpsWorks スタックがソースコントロールリポジトリからアプリケーションをデプロイするために使用する情報を指定する一連のコンテンツ。コンテンツはリポジトリのタイプによって異なります。

password

プライベートリポジトリの場合はパスワード、パブリックリポジトリの場合は "null" (文字列)。プライベート S3 バケットでは、このコンテンツはシークレットキーに設定されます。

revision

リポジトリに複数のブランチがある場合、そのコンテンツはアプリケーションのブランチまたはバージョンを指定します "version1" (など) (文字列)。それ以外の場合は、"null" に設定されます。

ssh_key

プライベート Git リポジトリにアクセスする場合は [デプロイ SSH キー](#)、パブリックリポジトリの場合は (文字列)。"null"

type

アプリケーションのソースの場所 (文字列)。有効な値を次に示します。

- "archive"

- "git"
- "other"
- "s3"

url

アプリケーションが存在する場所 (文字列)。

ユーザー

プライベートリポジトリの場合はユーザー名、パブリックリポジトリの場合は "null" (文字列)。プライベート S3 バケットでは、このコンテンツはアクセスキーに設定されます。

attributes

アプリのディレクトリ構造と内容を説明する一連のコンテンツ。

document_root

ドキュメントツリーのルートディレクトリです。デプロイメントディレクトリに関連するドキュメントルートへのパス (またはアプリケーションのホームページの場所 (home_html など)) を定義します。この属性を指定しない場合、document_root はデフォルトの public になります。document_root の値は、必ず、a-z、A-Z、0-9、_ (アンダースコア) または - (ハイフン) で始まります。

data_sources

アプリケーションのデータベースに接続するために必要な情報。アプリにデータベースレイヤーがアタッチされている場合、AWS OpsWorks スタックはこのコンテンツに適切な値を自動的に割り当てます。

data_sources の値は配列で、配列はキーではなく整数の補正值によってアクセスされます。例えば、アプリの最初のデータソースにアクセスするには、app[:data_sources][0][:type] を使用します。

database_name

データベース名 (文字列)。通常はアプリケーションの短縮名です。

type

データベースインスタンスのタイプ。通常は "RdsDbInstance" (文字列) です。

arn

データベースインスタンスの Amazon リソースネーム (ARN) (文字列)。

デプロイ

アプリケーションを展開する必要があるかどうか (ブール)。Deploy ライフサイクルイベントでは、デプロイする必要があるアプリケーションに対して true です。Setup ライフサイクルイベントで、このコンテンツはすべてのアプリケーションに対して true になります。いずれのアプリケーションをインスタンスにデプロイする必要があるかを調べるには、インスタンスが属するレイヤーを確認します。

domains

アプリケーションのドメインのリスト (文字列のリスト)。

enable_ssl

SSL のサポートが有効かどうか (ブール)。

環境

アプリケーションに対して定義されたユーザー指定の環境変数の集合。アプリケーション環境変数の定義方法の詳細については、「[アプリケーションの追加](#)」を参照してください。各コンテンツ名が環境変数名に設定され、対応する値が変数の値に設定されます。

name

表示の目的で使用されるアプリケーションの名前 (文字列)。

shortname

アプリケーションの短縮名。名前 (文字列) から AWS OpsWorks スタックによって生成されます。内部的に、およびレシピでは短縮名が使用されます。短縮名は、アプリファイルのインストール先ディレクトリ名として使用されます。

ssl_configuration

証明書

SSL サポートを有効にした場合はアプリケーションの SSL 証明書、それ以外の場合は "null" (文字列)。

chain

SSL が有効である場合は、中間認証局キーまたはクライアント認証を指定するためのコンテンツ (文字列)。

private_key

SSL サポートを有効にした場合はアプリケーションの SSL プライベートキー、それ以外の場合は "null" (文字列)。

type

Chef 12 Linux と Chef 12.2 Windows スタックでは常に "other" に設定されるアプリケーションのタイプ (文字列)。

コマンドデータバグ (aws_opsworks_command)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

AWS OpsWorks スタックが 1 つ以上のインスタンスで実行するコマンドの設定を表します。

以下の例は、Chef 検索を使用して単一のデータバグ項目を通じて検索を実行し、次に複数のデータバグ項目を使用して、コマンドのタイプと送信されるタイミングとともに Chef ログにメッセージを書き込む方法を示しています。

```
command = search("aws_opsworks_command").first
Chef::Log.info("***** The command's type is '#{command['type']}' *****")
Chef::Log.info("***** The command was sent at '#{command['sent_at']}' *****")

search("aws_opsworks_command").each do |command|
  Chef::Log.info("***** The command's type is '#{command['type']}' *****")
  Chef::Log.info("***** The command was sent at '#{command['sent_at']}'
  *****")
end
```

[args](#)

[command_id](#)

[iam_user_arn](#)

[instance_id](#)

[sent_at](#)

[type](#)

args

コマンドの引数 (文字列)。

command_id

AWS OpsWorks スタック (文字列) によって割り当てられたコマンドのランダムな一意の識別子。

iam_user_arn

コマンドがユーザーによって作成された場合、コマンドを作成したユーザーの Amazon リソース名 (ARN) (文字列)。

instance_id

コマンドが実行されたインスタンスの識別子 (文字列)。

sent_at

AWS OpsWorks スタックがコマンドを実行したときのタイムスタンプ (文字列)。

type

コマンドのタイプ (文字列)。有効な値を次に示します。

- "configure"
- "deploy"
- "deregister"
- "execute_recipes"
- "grant_remote_access"
- "install_dependencies"
- "restart"
- "revoke_remote_access"
- "rollback"
- "setup"
- "shutdown"
- "start"
- "stop"

- "sync_remote_users"
- "undeploy"
- "update_agent"
- "update_custom_cookbooks"
- "update_dependencies"

Amazon ECS クラスターデータバグ (aws_opsworks_ecs_cluster)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行に関するご質問は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Amazon ECS クラスターの設定を示します。

以下の例は、Chef 検索を使用して単一のデータバグ項目を通じて検索してから、複数のデータバグ項目を検索して、Amazon ECS クラスターの名前と Amazon リソースネーム (ARN) とともにメッセージを Chef ログに書き込む方法を示しています。

```
ecs_cluster = search("aws_opsworks_ecs_cluster").first
Chef::Log.info("***** The ECS cluster's name is
 '#{ecs_cluster['ecs_cluster_name']}' *****")
Chef::Log.info("***** The ECS cluster's ARN is '#{ecs_cluster['ecs_cluster_arn']}'
 *****")

search("aws_opsworks_ecs_cluster").each do |ecs_cluster|
  Chef::Log.info("***** The ECS cluster's name is
 '#{ecs_cluster['ecs_cluster_name']}' *****")
  Chef::Log.info("***** The ECS cluster's ARN is
 '#{ecs_cluster['ecs_cluster_arn']}' *****")
end
```

[ecs_cluster_arn](#)

[ecs_cluster_name](#)

ecs_cluster_arn

クラスターの Amazon リソースネーム (ARN) (文字列)。

ecs_cluster_name

クラスターの名前 (文字列)。

Elastic Load Balancing データバッグ (aws_opsworks_elastic_load_balancer)

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Elastic Load Balancing ロードバランサーの設定を示します。

以下の例は、Chef 検索を使用して単一のデータバッグ項目を通じて検索してから、複数のデータバッグ項目を検索して、Elastic Load Balancing のロードバランサーの名前と DNS 名とともにメッセージを Chef ログに書き込む方法を示しています。

```
elastic_load_balancer = search("aws_opsworks_elastic_load_balancer").first
Chef::Log.info("***** The ELB's name is
 '#{elastic_load_balancer['elastic_load_balancer_name']}' *****")
Chef::Log.info("***** The ELB's DNS name is '#{elastic_load_balancer['dns_name']}'
 *****")

search("aws_opsworks_elastic_load_balancer").each do |elastic_load_balancer|
  Chef::Log.info("***** The ELB's name is
 '#{elastic_load_balancer['elastic_load_balancer_name']}' *****")
  Chef::Log.info("***** The ELB's DNS name is
 '#{elastic_load_balancer['dns_name']}' *****")
end
```


[elastic_load_balancer_name](#)[dns_name](#)[layer_id](#)

elastic_load_balancer_name

ロードバランサーの名前 (文字列)。

dns_name

ロードバランサーの DNS 名 (文字列)。

layer_id

ロードバランサーが割り当てられているレイヤーの AWS OpsWorks スタック ID (文字列)。

インスタンスデータバグ (aws_opsworks_instance)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

インスタンスの設定を表します。

以下の例は、Chef 検索を使用して単一のデータバグ項目を通じて検索を実行し、次に複数のデータバグ項目を使用して、インスタンスのホスト名と ID とともに Chef ログにメッセージを書き込む方法を示しています。

```
instance = search("aws_opsworks_instance").first
Chef::Log.info("***** The instance's hostname is '#{instance['hostname']}'
*****")
Chef::Log.info("***** The instance's ID is '#{instance['instance_id']}'
*****")

search("aws_opsworks_instance").each do |instance|
  Chef::Log.info("***** The instance's hostname is '#{instance['hostname']}'
*****")
```

```

Chef::Log.info("***** The instance's ID is '#{instance['instance_id']}'
*****")
end

```

以下の例は、Chef 検索を使用して複数のデータバッグ項目を検索し、指定された Amazon EC2 インスタンス ID を含むデータバッグ項目を見つけるためのさまざまな方法を示しています。次に、データバッグ項目のコンテンツを使用して、対応するインスタンスのパブリック IP アドレスとともに Chef ログにメッセージを書き込みます。

```

instance = search("aws_opsworks_instance", "ec2_instance_id:i-12345678").first
Chef::Log.info("***** For instance '#{instance['ec2_instance_id']}', the
instance's public IP address is '#{instance['public_ip']}' *****")

search("aws_opsworks_instance").each do |instance|
  if instance['ec2_instance_id'] == 'i-12345678'
    Chef::Log.info("***** For instance '#{instance['ec2_instance_id']}', the
instance's public IP address is '#{instance['public_ip']}' *****")
  end
end
end

```

次の例では、Chef 検索で `self:true` を使用して、レシピを実行しているインスタンスに関連する情報を含むデータバッグ項目を検索する方法を示しています。次に、データバッグ項目の内容を使用して、対応するインスタンスの AWS OpsWorks スタック生成 ID とインスタンスのパブリック IP アドレスを含むメッセージを Chef ログに書き込みます。

```

instance = search("aws_opsworks_instance", "self:true").first
Chef::Log.info("***** For instance '#{instance['instance_id']}', the instance's
public IP address is '#{instance['public_ip']}' *****")

```

ami_id	architecture	auto_scaling_type
availability_zone	created_at	ebs_optimized
ec2_instance_id	elastic_ip	hostname
instance_id	instance_type	layer_ids
os	private_dns	private_ip
public_dns	public_ip	root_device_type

root_device_volume_id	self	ssh_host_dsa_key_fingerprint
ssh_host_dsa_key_private	ssh_host_dsa_key_public	ssh_host_rsa_key_fingerprint
ssh_host_rsa_key_private	ssh_host_rsa_key_public	status
subnet_id	virtualization_type	

ami_id

インスタンスの AMI (Amazon Machine Image) ID (文字列)。

architecture

インスタンスのアーキテクチャ (文字列)。常に "x86_64" に設定されます。

auto_scaling_type

インスタンスのスケールリングタイプ: null、timer、または load (文字列)。

availability_zone

インスタンスのアベイラビリティゾーン (AZ) ("us-west-2a" など) (文字列)。

created_at

UTC の "*yyyy-mm-ddThh:mm:ss+hh:mm*" 形式で示されるインスタンスが作成された時間 (文字列)。例えば、"2013-10-01T08:35:22+00:00" は 2013 年 10 月 10 日 8:35:22 (タイムゾーンオフセットなし) に対応します。詳細については、[ISO 8601](#) を参照してください。

ebs_optimized

インスタンスが EBS 最適化されているかどうか (ブール値)。

ec2_instance_id

EC2 インスタンス ID (文字列)。

elastic_ip

Elastic IP アドレス (文字列)。インスタンスに Elastic IP アドレスがない場合は "null" に設定されます。

hostname

ホスト名 ("demo1" など) (文字列)。

instance_id

インスタンス ID。インスタンス (文字列) を一意に識別する AWS OpsWorks スタック生成の GUID です。

instance_type

インスタンスのタイプ ("c1.medium" など) (文字列)。

layer_ids

一意の ID (307ut64c-c7e4-40cc-52f0-67d5k1f9992c など) で識別されるインスタンスのレイヤーのリスト。

os

インスタンスのオペレーティングシステム (文字列)。有効な値を次に示します。

- "Amazon Linux 2"
- "Amazon Linux 2018.03"
- "Amazon Linux 2017.09"
- "Amazon Linux 2017.03"
- "Amazon Linux 2016.09"
- "Custom"
- "Microsoft Windows Server 2022 Base"
- "Microsoft Windows Server 2022 with SQL Server Express"
- "Microsoft Windows Server 2022 with SQL Server Standard"
- "Microsoft Windows Server 2022 with SQL Server Web"
- "Microsoft Windows Server 2019 Base"
- "Microsoft Windows Server 2019 with SQL Server Express"
- "Microsoft Windows Server 2019 with SQL Server Standard"
- "Microsoft Windows Server 2019 with SQL Server Web"
- "CentOS 7"
- "Red Hat Enterprise Linux 7"
- "Ubuntu 20.04 LTS"
- "Ubuntu 18.04 LTS"
- "Ubuntu 16.04 LTS"

- "Ubuntu 14.04 LTS"

private_dns

プライベート DNS 名 (文字列)。

private_ip

プライベート IP アドレス (文字列)。

public_dns

パブリック DNS 名 (文字列)。

public_ip

パブリック IP アドレス (文字列)。

root_device_type

ルートデバイスのタイプ (文字列)。有効な値を次に示します。

- "ebs"
- "instance-store"

root_device_volume_id

ルートデバイスのボリューム ID (文字列)。

self

このデータバグ項目に、レシピを実行中のインスタンスに関する情報が含まれている場合は true。それ以外の場合は false (ブール型)。この値は、AWS OpsWorks スタック API ではなくレシピでのみ使用できます。

ssh_host_dsa_key_fingerprint

長い DSA パブリックキー (文字列) を特定するバイトの短いシーケンス。

ssh_host_dsa_key_private

インスタンスの SSH 認証用の DSA で生成されたプライベートキー (文字列)。

ssh_host_dsa_key_public

インスタンスの SSH 認証用の DSA で生成されたパブリックキー (文字列)。

ssh_host_rsa_key_fingerprint

長い RSA パブリックキー (文字列) を特定するバイトの短いシーケンス。

ssh_host_rsa_key_private

インスタンスの SSH 認証用の RSA で生成されたプライベートキー (文字列)。

ssh_host_rsa_key_public

インスタンスの SSH 認証用の RSA で生成されたパブリックキー (文字列)。

status

インスタンスのステータス (文字列)。有効な値を次に示します。

- "requested"
- "booting"
- "running_setup"
- "online"
- "setup_failed"
- "start_failed"
- "terminating"
- "terminated"
- "stopped"
- "connection_lost"

subnet_id

インスタンスのサブネット ID (文字列)。

virtualization_type

インスタンスの仮想化タイプ (文字列)。

レイヤーデータバグ (aws_opsworks_layer)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

レイヤーの設定を表します。

以下の例は、Chef 検索を使用して単一のデータバッグ項目を通じて検索を実行し、次に複数のデータバッグ項目を使用して、レイヤーの名前および短縮名とともに Chef ログにメッセージを書き込む方法を示しています。

```
layer = search("aws_opsworks_layer").first
Chef::Log.info("***** The layer's name is '#{layer['name']}' *****")
Chef::Log.info("***** The layer's shortname is '#{layer['shortname']}'
*****")

search("aws_opsworks_layer").each do |layer|
  Chef::Log.info("***** The layer's name is '#{layer['name']}' *****")
  Chef::Log.info("***** The layer's shortname is '#{layer['shortname']}'
*****")
end
```

ecs_cluster_arn	layer_id	name
パッケージ	shortname	type
volume_configurations		

[ecs_cluster_arn](#)

レイヤーに Amazon ECS クラスターが割り当てられている場合は、Amazon ECS クラスターの Amazon リソースネーム (ARN) (文字列)。

[暗号化](#)

EBS ボリュームが暗号化されている場合は true。それ以外の場合は false (ブール値)。

[layer_id](#)

レイヤー ID。AWS OpsWorks スタックによって生成され、レイヤー (文字列) を一意に識別する GUID です。

[name](#)

コンソール内のレイヤーを表すために使用されるレイヤーの名前 (文字列)。ユーザー定義も使用でき、一意である必要はありません。

パッケージ

インストールされるパッケージのリスト (文字列のリスト)。

shortname

ユーザー定義のレイヤーの短縮名 (文字列)。

type

Chef 12 Linux と Chef 12.2 Windows では常に "custom" に設定されるレイヤーのタイプ (文字列)。

volume_configurations

Amazon EBS ボリューム設定のリスト。

iops

ボリュームがサポートできる 1 秒あたりの I/O オペレーションの数。

mount_point

ボリュームのマウントポイントディレクトリ。

number_of_disks

ボリュームのディスク数。

raid_level

ボリュームの RAID 設定レベル。

size

GiB 単位のボリュームサイズ。

volume_type

ボリュームのタイプ: 汎用、マグネティック、プロビジョンド IOPS、スループット最適化 HDD、Cold HDD。

Amazon RDS データバッグ (aws_opsworks_rds_db_instance)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソ

リユーシオンに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

以下のような、Amazon Relational Database Service (Amazon RDS) インスタンスの設定を指定するデータバッグコンテンツのセット。

アドレス	db_instance_identifier	db_password
db_user	engine	rds_db_instance_arn
region		

以下の例は、Chef 検索を使用して単一のデータバッグ項目を通じて検索を実行し、次に複数のデータバッグ項目を使用して、Amazon RDS インスタンスのアドレスおよびデータベースエンジンタイプとともに Chef ログにメッセージを書き込む方法を示しています。

```
rds_db_instance = search("aws_opsworks_rds_db_instance").first
Chef::Log.info("***** The RDS instance's address is
 '#{rds_db_instance['address']}' *****")
Chef::Log.info("***** The RDS instance's database engine type is
 '#{rds_db_instance['engine']}' *****")

search("aws_opsworks_rds_db_instance").each do |rds_db_instance|
  Chef::Log.info("***** The RDS instance's address is
 '#{rds_db_instance['address']}' *****")
  Chef::Log.info("***** The RDS instance's database engine type is
 '#{rds_db_instance['engine']}' *****")
end
```

アドレス

インスタンスの DNS 名。

port

インスタンスのポートです。

db_instance_identifier

インスタンスの ID。

db_password

インスタンスのマスターパスワード。

db_user

インスタンスのマスターユーザー名。

engine

インスタンスのデータベースエンジン (mysql など)。

rds_db_instance_arn

インスタンスの Amazon リソースネーム (ARN)。

region

インスタンスの AWS リージョン (us-west-2 など)。

スタックデータバグ (aws_opsworks_stack)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

スタックの設定を表します。

以下の例では、Chef 検索を使用して、スタックの名前とクックブックのソース URL とともに、メッセージを Chef ログに書き込む方法を示しています。

```
stack = search("aws_opsworks_stack").first
Chef::Log.info("***** The stack's name is '#{stack['name']}' *****")
Chef::Log.info("***** The stack's cookbook URL is
 '#{stack['custom_cookbooks_source']['url']}' *****")
```

arn	custom_cookbooks_source	name
region	stack_id	use_custom_cookbooks
vpc_id		

arn

スタックの Amazon リソースネーム (ARN) (文字列)

custom_cookbooks_source

カスタムクックブックのソースリポジトリを指定するコンテンツのセット。

type

リポジトリのタイプ (文字列)。有効な値を次に示します。

- "archive"
- "git"
- "s3"

url

リポジトリの URL ("git://github.com/amazonwebservices/opsworks-demo-php-simple-app.git" など) (文字列)。

username

プライベートリポジトリの場合はユーザー名、パブリックリポジトリの場合は null (文字列)。プライベート Amazon Simple Storage Service (Amazon S3) バケットの場合、このコンテンツにはアクセスキーに設定されます。

password

プライベートリポジトリの場合はパスワード、パブリックリポジトリの場合は null (文字列)。プライベート S3 バケットでは、このコンテンツはシークレットキーに設定されます。

ssh_key

プライベート Git リポジトリにアクセスする場合は [デプロイ SSH キー](#)、パブリックリポジトリの場合は (文字列)。null

revision

リポジトリに複数のブランチがある場合、そのコンテンツはアプリケーションのブランチまたはバージョンを指定します "version1" (など) (文字列)。それ以外の場合は、null に設定されます。

name

スタックの名前 (文字列)。

region

スタックの AWS リージョン (文字列)。

stack_id

スタックを識別する GUID (文字列)。

use_custom_cookbooks

カスタムクックブックが有効になっているかどうか (ブール)。

vpc_id

スタックが VPC で実行されている場合は VPC ID (文字列)。

ユーザーデータバグ (aws_opsworks_user)

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行に関するご質問は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

ユーザーの設定を表します。

以下の例は、Chef 検索を使用して単一のデータバグ項目を検索してから、複数のデータバグ項目を検索して、ユーザーのユーザー名と Amazon リソースネーム (ARN) とともにメッセージを Chef ログに書き込む方法を示しています。

```
user = search("aws_opsworks_user").first
```

```
Chef::Log.info("***** The user's user name is '#{user['username']}' *****")
Chef::Log.info("***** The user's user ARN is '#{user['iam_user_arn']}'
*****")

# Or...

search("aws_opsworks_user").each do |user|
  Chef::Log.info("***** The user's user name is '#{user['username']}' *****")
  Chef::Log.info("***** The user's user ARN is '#{user['iam_user_arn']}'
*****")
end
```

[administrator_privileges](#)[iam_user_arn](#)[remote_access](#)[ssh_public_key](#)[unix_user_id](#)[username](#)

administrator_privileges

ユーザーが管理者アクセス権限を持っているかどうか (ブール値)。

iam_user_arn

ユーザーの Amazon リソースネーム (ARN) (文字列)。

remote_access

ユーザーが RDP を使用してインスタンスにログインできるかどうか (ブール値)。

ssh_public_key

AWS OpsWorks スタックコンソールまたは API (文字列) を介して提供されるユーザーのパブリックキー。

unix_user_id

ユーザーの UNIX ID (数値)。

username

ユーザー名 (文字列)。

OpsWorks エージェントの変更

⚠ Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

Chef 12 エージェントリリース

次の表は、AWS OpsWorks Stacks が管理するインスタンスにインストールする Chef 12 エージェントの重要な変更点を示しています。

エージェントのバージョン	説明	リリース日
4042	<ul style="list-style-type: none">このエージェントリリースには小さな変更のみが含まれており、新特徴量はない	2023 年 2 月 7 日
4041	<ul style="list-style-type: none">このエージェントリリースには小さな変更のみが含まれており、新特徴量はないAmazon CA 証明書の更新	2023 年 1 月 27 日
4040	<ul style="list-style-type: none">このエージェントリリースには小さな変更のみが含まれており、新特徴量はない	2022 年 7 月 22 日
4039	<ul style="list-style-type: none">Ubuntu AMI の ECS インテグレーションの修正	2020 年 4 月 30 日
4038	<ul style="list-style-type: none">DST 変更中にインスタンス統計を送信するときのバグを修正エージェントのダウンロードおよびインストール中に no_proxy 環境変数を優先	2020 年 3 月 5 日
4037	<ul style="list-style-type: none">SigV4 を使用してリージョンなしで S3 URL へのリクエストに署名するためのサポートを追加	2019 年 6 月 4 日

エージェントのバージョン	説明	リリース日
	<ul style="list-style-type: none">• SigV2 を使用して S3 リクエストに署名するためのサポートを削除	
4035	<ul style="list-style-type: none">• ECS 設定中のバグを修正• インスタンスタイプ変更後の fstab エントリの重複を修正	2019 年 5 月 8 日
4033	<ul style="list-style-type: none">• Ubuntu 18.04 のサポートを追加• Amazon Linux 2 用のエージェントのインストール時のバグを修正	2018 年 11 月 26 日
4032	<ul style="list-style-type: none">• Amazon Linux 2 のサポートを追加	2018 年 10 月 24 日
4031	<ul style="list-style-type: none">• Amazon Linux 2018.03 のサポートを追加する• 別のアカウントでホストされているパブリック S3 アーカイブのサポート	2018 年 8 月 15 日
4030	<ul style="list-style-type: none">• c5d インスタンスのボリューム処理を修正	2018 年 5 月 31 日
4029	<ul style="list-style-type: none">• Ubuntu 14.04 に nvme-cli をインストール• c5 および m5 インスタンスでボリュームのマウントを修正• 再起動時に常にホスト名を保持	2018 年 5 月 2 日
4028	<ul style="list-style-type: none">• CentOS の monit 設定を修正	2018 年 3 月 20 日
4027	<ul style="list-style-type: none">• Ubuntu 14.04 上で NVMe ボリュームをマウントするサポート (nvme-cli が手動でインストールされている必要があります)• ボリュームに name プロパティは必要ではありません。	2018 年 2 月 17 日

エージェントのバージョン	説明	リリース日
4026	<ul style="list-style-type: none">• EBS ボリューム ID を使用した、NVMe ベースの EBS ボリュームのマウント• i3 インスタンスで EBS ボリュームのマウントを修正• c5 および m5 インスタンスで EBS ボリュームをマウントする順序を修正	2018 年 1 月 31 日
4025	<ul style="list-style-type: none">• NVMe デバイスの処理を修正	2017 年 13 月 12 日
4024	<ul style="list-style-type: none">• Amazon Linux 2017.09 のサポートを追加する	2017 年 5 月 12 日
4023	<ul style="list-style-type: none">• CloudWatch Logs 統合のサポートを追加	2017 年 4 月 2 日
4022	<ul style="list-style-type: none">• Chef クライアントバージョンを 12.18.31 に更新する	2017 年 2 月 1 日
4021	<ul style="list-style-type: none">• プロキシ処理の向上	2016 年 12 月 16 日
4020	<ul style="list-style-type: none">• Chef クライアントバージョンを 12.16.42 に更新する	2016 年 12 月 8 日
4019	<ul style="list-style-type: none">• エージェントのインストール中のソースプロキシ変数• Red Hat Enterprise Linux 7 では systemd の代わりに monit を使用する• Red Hat Enterprise Linux 7 で EPEL をセットアップしない• プロセスをロックするために flock ではなく lockrun.c を使用する• ps -p1 を確認するときの systemd の異常な出力を回避する	2016 年 10 月 19 日

エージェントのバージョン	説明	リリース日
4018	<ul style="list-style-type: none">• Chef クライアントバージョンを 12.13.37 に更新する• Amazon Linux 2016.09 のサポートを追加する	2016 年 8 月 25 日
4017	<ul style="list-style-type: none">• Chef クライアントバージョンを 12.12.15 に更新する	2016 年 8 月 10 日
4016	<ul style="list-style-type: none">• monit が使用されていないシステムのエージェントのアンインストールを修正します	2016 年 6 月 23 日
4015	<ul style="list-style-type: none">• Amazon Linux 2016.03 の ECS 設定を固定する	2016 年 6 月 17 日
4011	<ul style="list-style-type: none">• Chef クライアントバージョンを 12.10.24 に更新する• ログのアップロード処理を向上させる	2016 年 5 月 19 日
4008	<ul style="list-style-type: none">• Amazon Linux 2016.03 のサポートを追加する• バンドルインストールにタイムアウトを追加する• xfs が存在する場合は /etc/filesystems に追加する	2016 年 3 月 16 日
4007	<ul style="list-style-type: none">• Chef クライアントバージョンを 12.7.2 に更新する• オンプレミスインスタンス用のエラー処理を向上させる (AWS 外でホストされるサーバー)• 最新 chef-sugar との互換性を向上させる• デプロイのアーカイブのダウンロードを再試行する	2016 年 3 月 4 日
4006	<ul style="list-style-type: none">• Chef クライアントバージョンを 12.6.0 に更新する• エージェントインストールに libxml2-devel/libxml2-dev パッケージと libxslt-devel/libxslt-dev パッケージをインストールしない	2016 年 1 月 21 日
4005	<ul style="list-style-type: none">• ec2 インフラストラクチャ用 Ohai で常に ec2 データを有効化することで ec2 インポートを固定する	2015 年 12 月 17 日

エージェントのバージョン	説明	リリース日
4004	<ul style="list-style-type: none"> AWS OpsWorks Chef 12 Linux - Chef Client 12.5.1 のスタックサポート 	2015 年 12 月 3 日

Chef 11.10 エージェントリリース

次の表は、AWS OpsWorks Stacks が管理するインスタンスにインストールする Chef 11.10 エージェントの重要な変更点を示しています。

エージェントのバージョン	説明	リリース日
3456	<ul style="list-style-type: none"> このエージェントリリースには小さな変更のみが含まれており、新特微量はない Amazon CA 証明書の更新 	2023 年 1 月 27 日
3455	<ul style="list-style-type: none"> このエージェントリリースには小さな変更のみが含まれており、新特微量はない 	2022 年 11 月 1 日
3454	<ul style="list-style-type: none"> Ubuntu AMI の ECS インテグレーションの修正 	2020 年 4 月 28 日
3453	<ul style="list-style-type: none"> DST 変更中にインスタンス統計を送信するときのバグを修正 RHEL7 セットアップでパッケージが不足しているバグを修正 エージェントのダウンロードおよびインストール中に no_proxy 環境変数を優先 	2020 年 3 月 5 日
3452	<ul style="list-style-type: none"> us-east-1 の場合、Amazon S3 仮想パス URL からそのリージョンを除外 内部クックブックを抽出してステージのリージョン固有のバケットにアップロード Chef 11.10 の fstab エントリを修正 	2019 年 8 月 13 日

エージェントのバージョン	説明	リリース日
	<ul style="list-style-type: none"> S3 に対する SigV2 の使用を削除し、リクエストでバケットのリージョンを取得 	
3451	<ul style="list-style-type: none"> Ruby 2.6.1 のサポートを追加 	2019 年 3 月 20 日
3450	<ul style="list-style-type: none"> デフォルトの EBS 属性を修正 Amazon Linux 2 の CloudWatchLogs エージェントインストールを修正 2.6.14 より新しいバージョンの rubygem 用のバンドラーのインストールを修正 パブリック S3 アーカイブのサポートを修正 	2018 年 12 月 3 日
3449	<ul style="list-style-type: none"> c5d インスタンスのボリューム処理を修正 NVMe デバイスインスタンスの RAID アレイのサポートを修正 	2018 年 6 月 5 日
3448	<ul style="list-style-type: none"> Ruby のデフォルト 2.3 バージョンを 2.3.7 にアップグレード Ubuntu 14.04 インスタンスの NVMe ベースのインスタンスにマウントされた EBS ボリュームの修正 別のアカウントでホストされている公開 Amazon S3 アーカイブをサポートします Red Hat Enterprise Linux インスタンスにおける <code>opsworks-agent</code> 起動時の問題を修正 	2018 年 5 月 8 日
3447	<ul style="list-style-type: none"> EBS ボリューム ID を使用した、NVMe ベースの EBS ボリュームのマウント i3 インスタンスで EBS ボリュームのマウントを修正 c5 および m5 で EBS ボリュームをマウントする順序を修正 Ruby のデフォルト 2.3 バージョンを 2.3.6 に更新 	2018 年 1 月 31 日

エージェントのバージョン	説明	リリース日
3446	<ul style="list-style-type: none"> NVMe デバイスの処理を修正 Ruby のデフォルト 2.3 バージョンを 2.3.5 に更新 	2017 年 12 月 14 日
3445	<ul style="list-style-type: none"> Amazon Linux 2017.09 のサポートを追加する Ruby のデフォルト 2.2 バージョンを 2.2.8 に更新 	2017 年 10 月 31 日
3444	<ul style="list-style-type: none"> CloudWatch ログのサポートを追加する 	2017 年 4 月 1 日
3443	<ul style="list-style-type: none"> プロキシ処理の向上 	2016 年 12 月 15 日
3442	<ul style="list-style-type: none"> Ruby のデフォルト 2.3 バージョンを 2.3.3 に更新 Ruby のデフォルト 2.2 バージョンを 2.2.6 に更新 	2016 年 6 月 12 日
3441	<ul style="list-style-type: none"> エージェントのインストール中のソースプロキシ変数 	2016 年 10 月 21 日
3440	<ul style="list-style-type: none"> Amazon Linux 2016.09 のサポートを追加する 	2016 年 9 月 13 日
3439	<ul style="list-style-type: none"> 小さな変更。新機能はありません 	2016 年 7 月 29 日
3438	<ul style="list-style-type: none"> Ruby 2.3.1 のサポートを追加 IAM インスタンスプロファイルの認証情報によるインスタンスの登録の改善 s3curl.pl の残りを削除 Amazon Linux 2016.03 の ECS 設定を固定する 	2016 年 6 月 17 日
3437	<ul style="list-style-type: none"> Ruby のデフォルト 2.2 バージョンを 2.2.5 に更新 	2016 年 5 月 4 日
3436	<ul style="list-style-type: none"> Red Hat Enterprise Linux の EPEL URL のアップデート 重要: この変更なしでは、Red Hat Enterprise Linux インスタンスの起動は失敗します。 	2016 年 4 月 18 日

エージェントのバージョン	説明	リリース日
3435	<ul style="list-style-type: none"> • Ruby のデフォルト 2.1 バージョンを 2.1.9 に更新 • Amazon S3 とアーカイブのデプロイメントの処理を改善します 	2016 年 4 月 6 日
3434	<ul style="list-style-type: none"> • Amazon Linux 2016.03 のサポートを追加する • パッケージのインストールの再試行 	2016 年 3 月 16 日
3433	<ul style="list-style-type: none"> • オンプレミスインスタンス (の外部でホストされるサーバー AWS) のいくつかの改善点 • 最新 との互換性を向上させる chef-sugar • デプロイのアーカイブのダウンロードを再試行する • Ruby gem のインストール URL の修正 	2016 年 2 月 27 日
3432	<ul style="list-style-type: none"> • バケット名で特殊文字を処理する改善 • s3_file をバージョン 2.6.6 に更新 • 指定されたマウントポイントがないボリュームのスキップ • 停止して開始する代わりに、常に unicorn を再起動して、デプロイ中のダウンタイムを防止する • setup コマンドで常にカスタムクックブックを更新する • RAID アレイを作成した後で initramfs を更新して、再起動時のマッピング問題を防止する 	2016 年 1 月 20 日
3431	<ul style="list-style-type: none"> • Rails レイヤーにおける passenger および unicorn gem のインストール問題が修正されました • Ruby の デフォルト 2.0、2.1 および 2.2 バージョンを 2.0.0p648、2.1.8 および 2.2.4 に更新 • postgres パッケージ名がカスタム JSON で設定されることが可能 • Node.js のデフォルトバージョンを 0.12.9 に更新 	2015 年 12 月 22 日

エージェントのバージョン	説明	リリース日
3430	<ul style="list-style-type: none"> 小さな変更。新機能はありません 	2015 年 11 月 25 日
3429	<ul style="list-style-type: none"> OpsWorks エージェントデーモナイズの向上 (stdout/stderr を閉じる) s3_file リソースの堅牢性の向上 (再試行、キャッチされた例外) 	2015 年 18 月 11 日
3428	<ul style="list-style-type: none"> Gemfile に基づく postgres アダプタ検出の追加によって、https://github.com/aws/opsworks-cookbook/s/issues/136 を修正 	2016 年 6 月 17 日
3427	<ul style="list-style-type: none"> エージェントでにおける認証情報の取得に関する問題を修正しました Ruby の デフォルト 2.0、2.1 および 2.2 バージョンを 2.0.0p647、2.1.7 および 2.2.3 に更新 	2015 年 9 月 11 日
3426	<ul style="list-style-type: none"> aws-sdk を 1.65.0 に更新しました s3curl を s3_file cookbook に置き換えることで、Amazon S3 からのダウンロードを改善します Node.js のデフォルトバージョンを 0.12.7 に変更 Node.js アプリにログ記録が追加されました。STDERR および STDOUT が shared/log ディレクトリにログされ、ローテーションされました カスタムクックブックのサブモジュールが明示的に更新をチェックアウトするようになりました deploy ディレクトリが作成される前にバインドのマウントが実行されたことを確認する https://github.com/aws/opsworks-cookbooks/issues/213 の回避策が追加されました 	2015 年 8 月 27 日
3425	<ul style="list-style-type: none"> Amazon Linux および Ubuntu への ECS のサポート 	2015 年 7 月 27 日

エージェントのバージョン	説明	リリース日
3424	<ul style="list-style-type: none">小さな変更。新機能はありません	2015年7月9日
3422	<ul style="list-style-type: none">Red Hat Enterprise Linux 7 の完全サポート/etc/hosts 世代のエラーに対する回復力を高めました	2015年6月29日
3421	<ul style="list-style-type: none">Red Hat Enterprise Linux 7 のデータベースパッケージ名の上書きオプションsystemd が kill 信号を monit によってモニタリングされるプロセスに送信することを回避するために、monit systemd 設定が更新されました。	2015年6月11日

AWS OpsWorks スタックリソース

Important

この AWS OpsWorks Stacks サービスは 2024 年 5 月 26 日にサポート終了となり、新規および既存のお客様の両方で無効になっています。できるだけ早くワークロードを他のソリューションに移行することを強くお勧めします。移行についてご質問がある場合は、[AWS re:Post](#) または [AWS Premium Support](#) を通じて AWS サポート チームにお問い合わせください。

このサービスを利用する際に役立つ関連リソースは次のとおりです。

リファレンスガイド、ツール、およびサポートリソース

複数の有益なガイド、フォーラム、連絡先情報、およびその他のリソースを AWS OpsWorks スタックとアマゾン ウェブ サービスで利用可能です。

- [AWS OpsWorks スタック API リファレンス](#) — 一般的なパラメータやエラーコードなど、AWS OpsWorks スタックのアクションとデータ型に関する説明、構文、使用例。
- [AWS OpsWorks スタックの技術的なよくある質問](#) — 開発者がこの製品について尋ねた上位の質問。
- [AWS OpsWorks Stacks Release Notes](#) (スタックリリースノート) – 最新リリースのハイレベルな概要。このドキュメントでは、新機能、修正内容、および既知の問題について特に説明します。
- [AWS Tools for PowerShell](#) – PowerShell 環境 AWS SDK for .NET 内の の機能を公開する一連の Windows PowerShell コマンドレット。
- [AWS Command Line Interface](#) (AWS コマンドラインインターフェイス) – AWS サービスにアクセスするための統一コマンドライン構文。AWS CLI は、サポートされているすべてのサービスにアクセスできるように、単一セットアッププロセスを使用します。
- [AWS OpsWorks スタックコマンドラインリファレンス](#) — コマンドラインプロンプトで使用する AWS OpsWorks スタック固有のコマンド。
- [クラスとワークショップ](#) – AWS スキルを磨き、実践的な経験を積むのに役立つセルフペースラボに加えて、ロールベースのコースと専門コースへのリンク。

- [AWS デベロッパーセンター](#) – チュートリアルを詳しく調べたり、ツールをダウンロードしたり、デ AWS ベロツパーイベントについて学習したりします。
- [AWS デベロッパーツール](#) – AWS アプリケーションを開発および管理するためのデベロッパー ツール、SDKs、IDE ツールキット、コマンドラインツールへのリンク。
- [入門リソースセンター](#) – のセットアップ方法 AWS アカウント、AWS コミュニティへの参加方法、最初のアプリケーションを起動する方法について説明します。
- [ハンズオンチュートリアル](#) – step-by-step チュートリアルに従って、で最初のアプリケーションを起動します AWS。
- [AWS ホワイトペーパー](#) – ソリューションアーキテクトや他の技術専門家が AWS 作成したアーキテクチャ、セキュリティ、経済などのトピックを網羅した、技術 AWS ホワイトペーパーの包括的なリストへのリンクです。
- [AWS サポート センター](#) – AWS サポート ケースを作成および管理するためのハブ。フォーラム、技術上のFAQs、サービスヘルスステータス、など、その他の役立つリソースへのリンクも含まれています AWS Trusted Advisor。
- [サポート](#) – クラウドでのアプリケーションの構築と実行に役立つ サポート one-on-one、高速応答サポートチャネルに関する情報のプライマリウェブページ。
- [お問い合わせ](#) - AWS の請求、アカウント、イベント、不正使用、その他の問題などに関するお問い合わせの受付窓口です。
- [AWS サイト規約](#) – 当社の著作権と商標、お客様のアカウント、ライセンス、サイトアクセス、およびその他のトピックに関する詳細情報。

AWS ソフトウェア開発キット

Amazon Web Services は、複数の異なるプログラミング言語から AWS OpsWorks スタックにアクセスするためのソフトウェア開発キットを提供しています。SDK ライブラリは、サービスリクエストに対する署名の暗号化、リクエストの再試行、エラーレスポンスの処理など、多数の一般的なタスクを自動化します。

- AWS SDK for Java – [Setup](#) (セットアップ) と [other documentation](#) (その他のドキュメント)
- AWS SDK for .NET – [Setup](#) (セットアップ) と [other documentation](#) (その他のドキュメント)。
- AWS SDK for PHP – [ドキュメント](#)
- AWS SDK for Ruby – [Documentation](#) (ドキュメント)
- [その他の文書](#)

- AWS SDK for Python (Boto) – [Setup](#) (セットアップ) と [other documentation](#) (その他のドキュメント)

ソースソフトウェアを開く

AWS OpsWorks スタックには、それぞれのライセンスによって管理されるさまざまなオープンソースソフトウェアパッケージが含まれています。詳細については、次を参照してください。

- Chef 12 の Linux インスタンスの場合、インスタンスの `/opt/aws/opsworks/current` ディレクトリで `THIRD_PARTY_LICENSES` ファイルを開きます。
- Linux 用の Chef 11.10 以前のバージョンについては、[OpsWorks Linux エージェント属性ドキュメント PDF](#) をダウンロードしてください。

AWS OpsWorks ドキュメント履歴

変更	説明	日付
AWS OpsWorks スタックへの更新	Detach in Place ツールを使用して OpsWorks Stacks OpsWorks サービスからインスタンスをデタッチできるようになりました。本ガイドの「 AWS OpsWorks Stacks Detach in Place ツールの使用 」を参照してください。	2024 年 4 月 11 日
AWS OpsWorks スタックへの更新	これで、移行スクリプトを使用して AWS OpsWorks Stacks を AWS Systems Manager Application Manager に移行できるようになりました。詳細については、本ガイドの「 AWS OpsWorks StacksAWS Systems Manager アプリケーションマネージャーへのアプリケーションの移行 」を参照してください。	2022 年 12 月 22 日
AWS OpsWorks for Chef Automate およびのアップデート AWS OpsWorks for Puppet Enterprise	AWS OpsWorks for Chef Automate ご使用のサーバーまたは OpsWorks Puppet Enterprise サーバーのシステムメンテナンスが失敗した場合にできることについて説明したトラブルシューティング手順が利用可能になりました。詳細については、このガイドの「 Chef Automate	2022 年 9 月 29 日

[サーバーのシステムメンテナンス失敗](#)」または「[Puppet Enterprise サーバーのシステムメンテナンス失敗](#)」を参照してください。

[およびのアップデート AWS OpsWorks for Chef Automate AWS OpsWorks for Puppet Enterprise](#)

AWS OpsWorks for Chef Automate ご使用のサーバーまたは Puppet Enterprise OpsWorks サーバーが何らかの状態になった場合のトラブルシューティングが可能になりました。Connection lost 詳細については、本ガイドの「[Chef Automate サーバーが Connection lost の状態にある](#)」または「[Puppet Enterprise サーバーが Connection lost の状態にある](#)」を参照してください。

2022 年 3 月 23 日

[AWS OpsWorks スタックへの更新](#)

セキュリティのベストプラクティスとして、AWS OpsWorks Stacks が他のサービスのタスクを実行できるようにする信頼関係ポリシーに OR aws:SourceAccount 条件キー (または両方) を追加できるようになりました。aws:SourceArn AWS 詳細については、本ガイドの「[AWS OpsWorks スタック](#)におけるクロスサービスでの混乱した代理処理を防止するを参照してください。

2022 年 3 月 4 日

[AWS OpsWorks for Chef Automate とが更新されました。AWS OpsWorks for Puppet Enterprise](#)

セキュリティのベストプラクティスとして、Puppet Enterprise AWS OpsWorks for Chef Automate AWS に他のサービスのタスクの実行を許可したり許可したりする信頼関係ポリシーに OpsWorks OR aws:SourceAccount 条件キー (または両方) を追加できるようになりました。aws:SourceArn 詳細については、[クロスサービスでの混乱した代理処理を防止する](#)を参照してください。

2022 年 1 月 10 日

[とが更新されました。AWS OpsWorks for Chef Automate AWS OpsWorks for Puppet Enterprise](#)

AWS OpsWorks for Chef Automate と OpsWorks Puppet Enterprise [AWSOpsWorksCMServiceRole](#) では管理ポリシーが更新され [AWSOpsWorksCMInstanceProfileRole](#)、[シークレットがに保存されるよう](#)になりました。AWS Secrets Manager

2021 年 5 月 3 日

[が更新されました。AWS OpsWorks for Puppet Enterprise](#)

コンソールで作成する OpsWorks Puppet Enterprise サーバーのエンジンバージョンが 2019.8.5 になりました。API を使用することで、Puppet Enterprise サーバーを作成する際に、2019 または 2017 のどちらのバージョンも指定することができます DescribeServers API は、その結果に PUPPET_API_CRL という属性を返すようになりました。この属性は、内部使用のための証明書取り消しリストを含んでいます。

2021 年 4 月 28 日

[AWS OpsWorks Stacks は新しい管理ポリシーを使用します。](#)

AWS OpsWorks Stacks は、Stacks AWS OpsWorks 内のすべてのアクションを実行する権限を含む管理ポリシーを変更しました。新しいポリシーはです。AWSOpsWorks_FullAccess このポリシーのアクセス権限についての詳細は、[「Example policies」](#) (ポリシーの例) を参照してください。

2021 年 2 月 19 日

[EC2-Classic から VPC AWS OpsWorks Stacks へのスタックの移行](#)

EC2-Classic から VPC AWS OpsWorks Stacks にスタックを移行する方法を説明するドキュメントが追加されました。

2020 年 9 月 29 日

[およびのスターターキットを再生成してください。AWS OpsWorks for Chef AutomateAWS OpsWorks for Puppet Enterprise](#)

AWS OpsWorks for Chef Automate またはサーバのスターターキットを再生成する方法を説明するドキュメントが追加されました。AWS OpsWorks for Puppet Enterprise

2020 年 7 月 29 日

[AWS OpsWorks for Puppet Enterprise カスタムドメイン、証明書、秘密鍵を使用するサーバーを作成できます。](#)

カスタムドメイン、証明書、プライベートキーを使用する OpsWorks for Puppet Enterprise サーバーを作成できるようになりました。既存のサーバーのバックアップからサーバーを作成することで、カスタムドメインを使用するように既存の Puppet Enterprise サーバーを更新できます。

2020 年 4 月 17 日

[AWS OpsWorks for Chef AutomateAWS OpsWorks for Puppet Enterprise コンソールでのタグ付けもサポートできるようになりました。](#)

またはを使用して、AWS OpsWorks for Chef Automate サーバ、AWS OpsWorks for Puppet Enterprise マスター、またはサーババックアップにタグを追加できるようになりました。AWS Management Console AWS CLI詳細については、「[タグを使用する \(Chef\)](#)」または「[タグを使用する \(Puppet\)](#)」を参照してください。

2020 年 2 月 26 日

[AWS OpsWorks for Chef Automate 既存の Chef Automate 1 サーバーを Chef Automate 2 に簡単にアップグレードできます。](#)

Chef Automate 1 AWS OpsWorks for Chef Automate を実行している対象サーバーを Chef Automate 2 にアップグレードするには、コンソールのサーバーの詳細ページで [アップグレードの開始] を選択するか、StartMaintenance API アクションを実行します。詳細については、「[AWS OpsWorks for Chef Automate サーバーを Chef Automate 2 にアップグレードする](#)」を参照してください。

2020 年 1 月 24 日

[AWS OpsWorks for Chef Automate および AWS OpsWorks for Puppet Enterprise](#)

AWS OpsWorks CM (AWS OpsWorks for Chef Automate と AWS OpsWorks for Puppet Enterprise) のセキュリティに関する新しい章がガイドに追加されました。

2019 年 12 月 23 日

[AWS OpsWorks for Chef Automate また、AWS OpsWorks for Puppet Enterprise タグ付けもサポートしています。](#)

を使用して、サーバ、AWS OpsWorks for Puppet Enterprise マスター、AWS OpsWorks for Chef Automate またはサーババックアップにタグを追加できるようになりました。AWS CLI AWS OpsWorks CM はタグベースの認証をサポートするようになりました。

2019 年 12 月 18 日

[AWS OpsWorks for Chef Automate カスタムドメイン、証明書、秘密鍵を使用するサーバーを作成できます。](#)

カスタムドメイン、証明書、秘密鍵を使用する AWS OpsWorks for Chef Automate 2.0 サーバーを作成できるようになりました。既存のサーバーのバックアップからサーバーを作成することで、既存の Chef Automate 2.0 サーバーをカスタムドメインを使用するように更新できます。

2019 年 10 月 22 日

[AWS OpsWorks スタックは Ruby 2.6.1 をサポートするようになりました。](#)

AWS OpsWorks スタックは Chef 11.10 スタックの Rails アプリケーションサーバーレイヤーで Ruby 2.6.1 をサポートします。

2019 年 5 月 2 日

[AWS OpsWorks for Chef Automate Chef Automate 2.0 をサポートするようになりました。](#)

AWS OpsWorks for Chef Automate 新しいサーバーでは Chef Automate 2.0 が実行されます。これには Chef のアップデート InSpec、コンプライアンススキャンとレポートの新機能、Chef Infraが含まれます。

2019 年 4 月 30 日

[AWS OpsWorks for Chef Automate および AWS OpsWorks for Puppet Enterprise](#)

AWS CloudFormation AWS OpsWorks for Chef Automate AWS OpsWorks for Puppet Enterprise を使用してサーバーまたはマスターサーバーを作成できるようになりました。

2019 年 1 月 24 日

AWS OpsWorks スタック	AWS OpsWorks スタックは Chef 12 スタックで Ubuntu 18.04 LTS を実行するインスタンスをサポートするようになりました。	2018 年 12 月 18 日
OpsWorks パペットエンタープライズ向け AWS	を使用するコントロールリポジトリへの SSH ベースの接続をセットアップする手順を追加しました。CodeCommit	2018 年 12 月 3 日
AWS OpsWorks スタック	AWS OpsWorks スタックは Chef 12 スタックで Amazon Linux 2 を実行するインスタンスをサポートするようになりました。	2018 年 11 月 15 日
AWS OpsWorks スタック	AWS OpsWorks スタックは、Chef 11.10 スタックで Amazon Linux 2018.03 を実行するインスタンスをサポートするようになりました。	2018 年 10 月 23 日
AWS OpsWorks スタック	AWS OpsWorks スタックは Chef 12 スタックで Amazon Linux 2018.03 を実行するインスタンスをサポートするようになりました。	2018 年 8 月 23 日
AWS OpsWorks for Chef Automate として Puppet Enterprise 用もあります。OpsWorks	OpsWorks for Puppet EnterpriseはPE 2018.1.2 にアップグレードされました。AWS OpsWorks for Chef Automate がシェフオートメーション 1.8.68 にアップグレードされました。	2018 年 6 月 29 日

- AWS OpsWorks for Chef Automate パペット・エンタープライズ API OpsWorks バージョンの場合:2016-11-01
- AWS OpsWorks スタック API バージョン:2016-03-08
- ドキュメントの最新更新日:2024-04-11

以前の更新

次の表に、2018年6月以前の「AWS OpsWorks ユーザーガイド」の各リリースにおける重要な変更点を示します。

説明	日付
AWS OpsWorks Windows ベースのスタックのスタック Chef バージョンが 12.22 にアップグレードされました。Ruby バージョンは 2.3.6 になりました。	2018年4月19日
を使用して Puppet Enterprise AWS OpsWorks for Chef Automate 用のサーバーまたはマスターを作成する手順が新しくなりました OpsWorks 。 AWS CLI	2018年3月23日
Chef Automate バージョンを 1.8 に更新しました。opsworks-audit クックブックを追加し Chef Compliance セットアップを簡素化しました。	2018年3月5日
Amazon AWS OpsWorks CloudWatch イベントのスタックイベントのサポートが追加されました。	2018年2月20日
AWS OpsWorks スタックの新しい EBS ポリリュームタイプと新しい API のサポートが追加されました。 DescribeOperatingSystems	2018年1月25日
OpsWorks Puppet Enterprise 向けになり、AWS OpsWorks for Chef Automate サーバーの作成時に複数のセキュリティグループを選択できるようになりました。	2018年1月18日
ヨーロッパ (パリ) AWS OpsWorks リージョンのスタックのサポートが追加されました。	2017年12月19日

説明	日付
さらに 6 つのリージョンで Puppet Enterprise AWS OpsWorks for Chef Automate OpsWorks のサポートとサポートが追加され、内の Puppet Enterprise AWS OpsWorks for Chef Automate OpsWorks サーバーのバックアップとバックアップを作成する手順が追加されました。 AWS Management Console	2017 年 12 月 18 日
Puppet Enterprise OpsWorks 向けの新しいサービスとドキュメントを追加しました。	2017 年 11 月 16 日
Amazon Linux 2017.09 AWS OpsWorks のサポートをスタックに追加しました。	2017 年 11 月 7 日
Chef コンプライアンスのサポートをに追加しました。 AWS OpsWorks for Chef Automate	2017 年 10 月 25 日
Amazon Linux 2017.09 のサポートをに追加しました。 AWS OpsWorks for Chef Automate	2017 年 10 月 9 日
AWS OpsWorks for Chef Automate システムメンテナンスのトピックを章に追加しました。	2017 年 7 月 28 日
AWS OpsWorks Stacks 内のタグのサポートが追加されました。	2017 年 6 月 6 日
CloudWatch Logs との統合が追加されました。	2017 年 4 月 10 日
AWS OpsWorks for Chef Automate 新しいサービスとドキュメントを追加しました。	2016 年 12 月 1 日
米国東部 (オハイオ) リージョンのリージョンエンドポイントへのサポートが追加されました。	2016 年 10 月 12 日
Amazon Linux 2016.09 オペレーティングシステムを実行するスタックとインスタンスのサポートが追加されました。	2016 年 9 月 30 日
アジアパシフィック (ソウル) と 9 つの追加リージョンエンドポイントへのサポートが追加されました。	2016 年 8 月 15 日

説明	日付
組み込みレイヤーで Node.js 0.12.15 と Ruby 2.3 のサポートが追加されました。	2016 年 7 月 6 日
アジアパシフィック (ムンバイ) リージョンへのサポートが追加されました。	2016 年 6 月 28 日
CentOS 7 オペレーティングシステムを実行するスタックとインスタンスのサポートが追加されました。	2016 年 6 月 22 日
CodePipeline ワークスルーの説明と AWS OpsWorks Stacks の統合を追加しました。	2016 年 6 月 2 日
Ubuntu 16.04 LTS オペレーティングシステムを実行するスタックとインスタンスのサポートが追加されました。	2016 年 6 月 1 日
Chef 12 Linux のサポートと関連ドキュメントを追加しました。	2015 年 12 月 3 日
「ご利用開始にあたって」に Node.js walkthrough を追加しました。	2015 年 7 月 14 日
2 つの新しいクックブックの例をクックブック 101 に追加しました。	2015 年 7 月 14 日
エージェントバージョン管理のサポートを追加しました。	2015 年 6 月 23 日
エージェントバージョンを管理するためのサポートを追加しました。	2015 年 6 月 24 日
カスタム Windows AMI のサポートが追加されました。	2015 年 6 月 22 日
ベストプラクティスに関する新しいトピックを 3 つ追加しました。	2015 年 6 月 11 日
Windows スタックのサポートが追加されました。	2015 年 5 月 18 日
ベストプラクティスに関する章を追加しました。	2014 年 12 月 15 日
Elastic Load Balancing の Connection Draining とカスタム Shutdown タイムアウトのサポートが追加されました。	2014 年 12 月 15 日
AWS OpsWorks Stacks の外部で作成されたインスタンスを登録するためのサポートが追加されました。	2014 年 12 月 9 日

説明	日付
Amazon SWF に対するサポートが追加されました。	2014 年 9 月 4 日
アプリケーションへの環境変数の関連付けのサポートと、増補したクックブック 101 を追加しました。	2014 年 7 月 16 日
クックブックを実装するチュートリアルの説明である、クックブック 101 を追加しました。	2014 年 7 月 16 日
のサポートが追加されました。 CloudTrail	2014 年 6 月 4 日
Amazon RDS に対するサポートが追加されました。	2014 年 5 月 14 日
Chef 11.10 と Berkshelf のサポートが追加されました。	2014 年 3 月 27 日
Amazon EBS の PIOPS ボリュームに対するサポートが追加されました。	2013 年 12 月 16 日
リソースベースの権限が追加されました。	2013 年 12 月 5 日
リソース管理が追加されました。	2013 年 10 月 7 日
VPC のサポートが追加されました。	2013 年 8 月 29 日
カスタム AMI と Chef 11.4 のサポートが追加されました。	2013 年 7 月 24 日
インスタンスごとに複数のレイヤーのコンソールサポートが追加されました。	2013 年 7 月 1 日
Amazon EBS ベースのインスタンス、Elastic Load Balancing、および Amazon CloudWatch モニタリングのサポートが追加されました。	2013 年 5 月 14 日
AWS OpsWorks Stacks ユーザーガイドの初回リリース。	2013 年 2 月 18 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。