ユーザーガイド

AWS CodeCommit

API バージョン 2015-04-13

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS CodeCommit: ユーザーガイド

Copyright © 2025 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 が所有しない商標はすべてそれぞれの所有者に所属します。所有者は必ずしも Amazon との提携や関連があるわけではありません。また、Amazon 支援を受けているとはかぎりません。

Table of Contents

..... xv
CodeCommit とは 1
CodeCommit の概要 1
CodeCommit、Git、ニーズに合った AWS サービスの選択 2
CodeCommit の仕組み 6
Amazon S3 でのファイルのバージョニングと CodeCommit の違い 8
CodeCommit の使用を開始するにはどうしたらいいですか? 8
Git の詳細情報 8
設定 9
認証情報の表示と管理 9
Git 認証情報を使用した設定 10
他の方法を使用した設定 10
CodeCommit、Git、および他のコンポーネントの互換性 12
Git 認証情報を使用した HTTPS ユーザーのセットアップ 13
ステップ 1: CodeCommit の初期設定 13
ステップ 2: Git をインストールする 15
ステップ 3: CodeCommit への HTTPS 接続用の Git 認証情報を作成する 15
ステップ 4: CodeCommit コンソールに接続し、リポジトリのクローンを作成する 17
次のステップ 18
git-remote-codecommit を使用した HTTPS 接続のセットアップ手順 19
ステップ 0: git-remote-codecommit の前提条件をインストールする 20
ステップ 1: CodeCommit の初期設定 21
ステップ 2: git-remote-codecommit をインストールする 24
ステップ 3: CodeCommit コンソールに接続し、リポジトリのクローンを作成する 25
次のステップ 27
開発ツールから接続する場合 27
AWS Cloud9 との統合 AWS CodeCommit 30
Visual Studio を と統合する AWS CodeCommit 34
を使用していない SSH ユーザーの場合 AWS CLI 35
ステップ 1: パブリックキーを IAM ユーザーに関連付ける 36
ステップ 2: SSH 設定に CodeCommit を追加する 37
次のステップ 37
Linux、macOS、または Unix での SSH 接続の場合 38
ステップ 1: CodeCommit の初期設定 38

ステップ 2: Git をインストールする ... 39
ステップ 3: Linux、macOS、または Unix で認証情報を設定する ... 40
ステップ 4: CodeCommit コンソールに接続し、リポジトリのクローンを作成する ... 44
次のステップ ... 46
Windows で SSH 接続をセットアップする手順 ... 46
ステップ 1: CodeCommit の初期設定 ... 46
ステップ 2: Git をインストールする ... 47
ステップ 3: Git および CodeCommit 用のパブリックキーとプライベートキーをセットアップする ... 48
ステップ 4: CodeCommit コンソールに接続し、リポジトリのクローンを作成する ... 52
次のステップ ... 54
AWS CLI 認証情報ヘルパーを使用した Linux、macOS、または Unix での HTTPS 接続の場合 ... 54
ステップ 1: CodeCommit の初期設定 ... 55
ステップ 2: Git をインストールする ... 58
ステップ 3: 認証情報ヘルパーを設定する ... 59
ステップ 4: CodeCommit コンソールに接続し、リポジトリのクローンを作成する ... 61
次のステップ ... 62
AWS CLI 認証情報ヘルパーを使用した Windows での HTTPS 接続の場合 ... 62
ステップ 1: CodeCommit の初期設定 ... 63
ステップ 2: Git をインストールする ... 67
ステップ 3: 認証情報ヘルパーを設定する ... 68
ステップ 4: CodeCommit コンソールに接続し、リポジトリのクローンを作成する ... 70
次のステップ ... 71
入門 ... 72
CodeCommit の開始方法 ... 72
前提条件 ... 73
ステップ 1: CodeCommit リポジトリを作成する ... 74
ステップ 2: リポジトリにファイルを追加する ... 77
ステップ 3: リポジトリの内容を参照する ... 79
ステップ 4: プルリクエストを作成して共同作業を行う ... 84
ステップ 5: クリーンアップ ... 90
ステップ 6: 次のステップ ... 91
Git と CodeCommit の開始方法 ... 91
ステップ 1: CodeCommit リポジトリを作成する ... 92
ステップ 2: ローカルリポジトリを作成する ... 94

ステップ 3: 最初のコミットを作成する ... 96
ステップ 4: 最初のコミットをプッシュする ... 97
ステップ 5: CodeCommit リポジトリを共有し、別のコミットをプッシュしてプルする ... 98
ステップ 6: ブランチを作成して共有する ... 100
ステップ 7: タグを作成して共有する ... 102
ステップ 8: アクセス許可を設定する ... 103
ステップ 9: クリーンアップする ... 107
製品およびサービスの統合 ... 109
他の AWS サービスとの統合 ... 109
コミュニティから統合の例 ... 116
ブログ記事 ... 116
コードサンプル ... 120
リポジトリを操作する ... 121
リポジトリの作成 ... 122
リポジトリを作成する (コンソール) ... 123
リポジトリを作成する (AWS CLI) ... 125
リポジトリへの接続 ... 127
CodeCommit リポジトリに接続するための前提条件 ... 128
CodeCommit リポジトリのクローンを作成してリポジトリに接続する ... 129
ローカルリポジトリを CodeCommit リポジトリに接続する ... 130
リポジトリの共有 ... 132
接続プロトコルを選択してユーザーと共有する ... 133
リポジトリの IAM ポリシーを作成する ... 134
リポジトリユーザーの IAM グループを作成する ... 136
接続情報をユーザーと共有する ... 137
リポジトリイベントの通知を設定する ... 138
リポジトリ通知ルールの使用 ... 140
通知ルールの作成 ... 140
通知の変更または無効化 ... 144
通知の削除 ... 145
リポジトリのタグ付け ... 146
リポジトリにタグを追加する ... 147
リポジトリのタグの表示 ... 149
リポジトリのタグを編集する ... 150
リポジトリからタグを削除する ... 152
リポジトリのトリガーの管理 ... 153

リソースの作成と CodeCommit のアクセス許可の追加 154
Amazon SNS トピック用のトリガーを作成する 155
Lambda 関数のトリガーを作成する 162
既存の Lambda 関数のトリガーを作成する 167
リポジトリのトリガーを編集する 175
リポジトリのトリガーのテスト 177
リポジトリからトリガーを削除する 179
リポジトリを Amazon CodeGuru Reviewer に関連付ける、または関連付けを解除する 182
リポジトリを CodeGuru Reviewer に関連付ける 184
CodeGuru Reviewer からリポジトリの関連付けを解除する 185
リポジトリの詳細の表示 185
リポジトリの詳細を表示する (コンソール) 186
CodeCommit リポジトリの詳細を表示する (Git) 186
CodeCommit リポジトリの詳細を表示する (AWS CLI) 188
リポジトリ設定の変更 192
リポジトリ設定を変更する (コンソール) 192
AWS CodeCommit リポジトリ設定の変更 (AWS CLI) 194
リポジトリ間での変更の同期 196
2 つのリポジトリにコミットをプッシュする 198
ロールを使用してリポジトリへのクロスアカウントアクセスを設定する 202
リポジトリへのクロスアカウントアクセス: AccountA の管理者のアクション 204
リポジトリへのクロスアカウントアクセス: AccountB の管理者のアクション 207
リポジトリへのクロスアカウントアクセス: AccountB のリポジトリユーザーのアクション 209
リポジトリの削除 215
CodeCommit リポジトリを削除するには (コンソール) 215
ローカルリポジトリを削除する 216
CodeCommit リポジトリを削除するには (AWS CLI) 216
ファイルの操作 218
リポジトリでのファイルの参照 219
CodeCommit リポジトリを参照する 220
ファイルの作成または追加 221
ファイルを作成またはアップロードする (コンソール) 221
ファイルを追加する (AWS CLI) 223
ファイルを追加する (Git) 224
ファイルの内容を編集する 224

ファイルを編集する (コンソール) ... 225
ファイルの編集または削除する (AWS CLI) ... 226
ファイルを編集する (Git) ... 228
プルリクエストの操作 ... 229
プルリクエストの作成 ... 233
プルリクエストを作成する (コンソール) ... 233
プルリクエストを作成する (AWS CLI) ... 235
承認ルールの作成 ... 237
プルリクエストの承認ルールを作成する (コンソール) ... 238
プルリクエストの承認ルールを作成する (AWS CLI) ... 240
プルリクエストの表示 ... 242
プルリクエストを表示する (コンソール) ... 242
プルリクエストを表示する (AWS CLI) ... 243
プルリクエストのレビュー ... 247
プルリクエストを確認する (コンソール) ... 248
プルリクエストを確認する (AWS CLI) ... 253
プルリクエストの更新 ... 258
プルリクエストを更新する (コンソール) ... 259
プルリクエストを更新する (AWS CLI) ... 259
承認ルールの編集または削除 ... 262
プルリクエストの承認ルールを編集または削をする (コンソール) ... 262
プルリクエストの承認ルールを編集または削除する (AWS CLI) ... 264
プルリクエストの承認ルールの上書き ... 266
承認ルールを上書きする (コンソール) ... 267
承認ルールを上書きする (AWS CLI) ... 267
プルリクエストのマージ ... 269
プルリクエストをマージする (コンソール) ... 270
プルリクエストをマージする (AWS CLI) ... 273
プルリクエスト内の競合を解決 ... 279
プルリクエストの競合を解決する (コンソール) ... 280
プルリクエスト内の競合を解決する (AWS CLI) ... 282
プルリクエストを閉じる ... 290
プルリクエストをクローズする (コンソール) ... 290
プルリクエストをクローズする (AWS CLI) ... 291
承認ルールテンプレートの操作 ... 293
承認ルールテンプレートを作成する ... 295

承認ルールテンプレートを作成する (コンソール) ... 295
承認ルールテンプレートを作成する (AWS CLI) ... 299
承認ルールテンプレートをリポジトリに関連付ける ... 300
承認ルールテンプレートを関連付ける (コンソール) ... 301
承認ルールテンプレートを関連付ける (AWS CLI) ... 301
承認ルールテンプレートの管理 ... 303
承認ルールテンプレートを管理する (コンソール) ... 303
承認ルールテンプレートを管理する (AWS CLI) ... 303
承認ルールテンプレートの関連付けを解除する ... 308
承認ルールテンプレートの関連付けを解除する (コンソール) ... 308
承認ルールテンプレートの関連付けを解除する (AWS CLI) ... 309
承認ルールテンプレートを削除する ... 310
承認ルールテンプレートを削除する (コンソール) ... 310
承認ルールテンプレートを削除する (AWS CLI) ... 311
コミットの操作 ... 312
コミットを作成する ... 313
を使用してリポジトリの最初のコミットを作成する AWS CLI ... 314
Git クライアントを使用してコミットを作成する ... 315
を使用してコミットを作成する AWS CLI ... 319
コミットの詳細の表示 ... 322
リポジトリのコミットの参照 ... 322
コミットの詳細を表示する (AWS CLI) ... 326
コミットの詳細を表示する (Git) ... 332
コミットの比較 ... 334
コミットをその親と比較する ... 335
2 つのコミット指定子を比較 ... 337
コミットについてコメントする ... 339
リポジトリのコミットに対するコメントを表示する ... 340
リポジトリのコミットに対するコメントの追加またはコメントへの応答 ... 340
コメントの表示、追加、更新、返信 (AWS CLI) ... 345
Git タグを作成する ... 354
Git を使用してタグを作成する ... 354
タグの詳細の表示 ... 355
タグの詳細を表示する (コンソール) ... 356
Git タグの詳細を表示する (Git) ... 357
タグの削除 ... 359

Git を使用して Git タグを削除する ... 359
ブランチの操作 ... 361
ブランチを作成する ... 363
ブランチを作成する (コンソール) ... 363
ブランチを作成する (Git) ... 364
ブランチを作成する (AWS CLI) ... 365
ブランチに対するプッシュとマージの制限 ... 367
ブランチへのプッシュとマージを制限するための IAM ポリシーの設定 ... 368
IAM グループまたはロールに IAM ポリシーを適用する ... 370
ポリシーのテスト ... 370
ブランチの詳細の表示 ... 371
ブランチの詳細を表示する (コンソール) ... 371
ブランチの詳細を表示する (Git) ... 372
ブランチの詳細を表示する (AWS CLI) ... 373
ブランチの比較とマージ ... 375
ブランチをデフォルトブランチと比較する ... 375
2 つの特定のブランチを比較する ... 375
2 つのブランチをマージする (AWS CLI) ... 376
ブランチ設定を変更する ... 379
デフォルトのブランチを変更する (コンソール) ... 379
デフォルトのブランチを変更する (AWS CLI) ... 380
ブランチを削除する ... 381
ブランチを削除する (コンソール) ... 382
ブランチを削除する (AWS CLI) ... 382
ブランチを削除する (Git) ... 383
ユーザー設定の操作 ... 385
CodeCommit に移行する ... 386
Git リポジトリを に移行する AWS CodeCommit ... 386
ステップ 0: CodeCommit へアクセスに必要なセットアップを行う ... 387
ステップ 1: CodeCommit リポジトリを作成する ... 393
ステップ 2: リポジトリのクローンを作成して CodeCommit リポジトリにプッシュする ... 396
ステップ 3: CodeCommit でファイルを表示する ... 397
ステップ 4: CodeCommit リポジトリを共有する ... 398
コンテンツを CodeCommit に移行する ... 401
ステップ 0: CodeCommit へアクセスに必要なセットアップを行う ... 402
ステップ 1: CodeCommit リポジトリを作成する ... 407

ステップ 2: ローカルコンテンツを CodeCommit リポジトリに移行する 409
ステップ 3: CodeCommit でファイルを表示する 410
ステップ 4: CodeCommit リポジトリを共有する 411
リポジトリの段階的移行 413
ステップ 0: 段階的に移行するかどうかを決める 414
ステップ 1: 前提条件をインストールし、CodeCommit リポジトリをリモートとして追加する 415
ステップ 2: 段階的移行に使用するスクリプトを作成する 417
ステップ 3: スクリプトを実行し、CodeCommit に段階的に移行する 417
付録: サンプルスクリプト `incremental-repo-migration.py` 419
セキュリティ 427
データ保護 427
AWS KMS および 暗号化 428
認証情報のローテーションの使用 431
Identity and Access Management 436
対象者 436
アイデンティティを使用した認証 437
ポリシーを使用したアクセスの管理 440
認証とアクセスコントロール 443
が IAM と AWS CodeCommit 連携する方法 513
CodeCommit のリソースベースのポリシー 515
CodeCommit タグに基づく認可 515
CodeCommit IAM ロール 518
アイデンティティベースのポリシーの例 519
トラブルシューティング 522
耐障害性 525
インフラストラクチャセキュリティ 525
CodeCommit のモニタリング 527
CodeCommit イベントのモニタリング 528
referenceCreated イベント 529
referenceUpdated イベント 530
referenceDeleted イベント 531
unreferencedMergeCommitCreated イベント 531
commentOnCommitCreated イベント 532
commentOnCommitUpdated イベント 533
commentOnPullRequestCreated イベント 534

commentOnPullRequestUpdated イベント 535
pullRequestCreated イベント 536
pullRequestSourceBranchUpdated イベント 537
pullRequestStatusChanged イベント 538
pullRequestMergeStatusUpdated イベント 539
approvalRuleTemplateCreated イベント 540
approvalRuleTemplateUpdated イベント 540
approvalRuleTemplateDeleted イベント 541
approvalRuleTemplateAssociatedWithRepository イベント 542
approvalRuleTemplateDisassociatedWithRepository イベント 543
approvalRuleTemplateBatchAssociatedWithRepositories イベント 544
approvalRuleTemplateBatchDisassociatedFromRepositories イベント 545
pullRequestApprovalRuleCreated イベント 545
pullRequestApprovalRuleDeleted イベント 547
pullRequestApprovalRuleOverridden イベント 548
pullRequestApprovalStateChanged イベント 550
pullRequestApprovalRuleUpdated イベント 552
reactionCreated イベント 553
reactionUpdated イベント 554
を使用した AWS CodeCommit API コールのログ記録 AWS CloudTrail 555
CloudTrail の CodeCommit 情報 555
CodeCommit ログファイルエントリの理解 556
AWS CloudFormation リソース 564
CodeCommit と AWS CloudFormation テンプレート 564
テンプレートの例 565
AWS CloudFormation、CodeCommit、および AWS Cloud Development Kit (AWS CDK) 566
の詳細 AWS CloudFormation 567
トラブルシューティング 568
Git 認証情報 (HTTPS) のトラブルシューティング 568
Git 認証情報 AWS CodeCommit: ターミナルまたはコマンドラインで CodeCommit リポジトリに接続すると、認証情報の入力プロンプトが引き続き表示されます。 569
Git 認証情報 AWS CodeCommit: Git 認証情報をセットアップしたが、システムが使用していない 569
git-remote-codecommit のトラブルシューティング 570
次のエラーが表示される: git: 'remote-codecommit' is not a git command 570
次のエラーが表示される: fatal: Unable to find remote helper for 'codecommit' 571

クローンエラー: IDE から CodeCommit リポジトリのクローンを作成できない 571
プッシュまたはプルエラー: IDE から CodeCommit リポジトリにコミットをプッシュまたはプルできない 571
インストールエラー: git-remote-codecommitexternally-managed-environmentエラーが表示されます 572
SSH 接続のトラブルシューティング 572
アクセスエラー: パブリックキーが IAM に正常にアップロードされたが、Linux、macOS、または Unix システムでは接続が失敗する。 573
アクセスエラー: パブリックキーは正常に IAM および SSH にアップロードされたが、Windows システム上で接続できない。 574
認証の問題: CodeCommit リポジトリへの接続時にホストの信頼性が確立できない 575
IAM エラー: 「無効な形式」(IAM へのパブリックキーの追加時) 582
SSH 認証情報を使用して複数の Amazon Web Services アカウントの CodeCommit リポジトリにアクセスする必要がある 582
Windows 上の Git: SSH を使用して接続を試みると、Bash エミュレーターまたはコマンドラインはフリーズします。 583
パブリックキー形式は、Linux の一部のディストリビューションにおいて指定する必要があります。 584
アクセスエラー: CodeCommit リポジトリへの接続時に SSH パブリックキーが拒否される 584
認証情報ヘルパー (HTTPS) のトラブルシューティング 584
`git config` コマンドを実行して認証情報ヘルパーを設定しようとするとエラーが発生する 585
リポジトリのクローンを作成しようとすると、ユーザー名の読み取りエラーが表示される 586
認証情報ヘルパーを使用すると Windows でコマンドが見つかりませんというエラーが表示される 586
CodeCommit リポジトリへの接続時にユーザー名の入力を求められる 587
Git for macOS: 認証情報ヘルパーは正常に設定できましたが、リポジトリへのアクセスが拒否されます (403) 587
Git for Windows: Git for Windows をインストールしましたが、リポジトリへのアクセスが拒否されます (403) 591
Git クライアントのトラブルシューティング 593
Git エラー: error: RPC failed; result=56, HTTP code = 200 fatal: リモートエンドが予期せず切断されました 593
Git エラー: リファレンスアップデートコマンドが多すぎる 594

Git エラー: Git の一部のバージョンで HTTPS 経由のプッシュが切断される 594
Git エラー: 'gnutls_handshake() failed' 594
Git エラー: Git で CodeCommit リポジトリを見つけることができない、またはリポジトリへのアクセス許可がない 594
Windows 上の Git: 利用できる認証メソッド (パブリックキー) がサポートされていない 595
アクセスエラーのトラブルシューティング 595
アクセスエラー: Windows から CodeCommit リポジトリへ接続する際、ユーザー名とパスワードの入力を求められる 596
アクセスエラー: CodeCommit リポジトリへの接続時にパブリックキーが拒否される 596
アクセスエラー: CodeCommit リポジトリへの接続時の「レートが超過しました」または「429」メッセージ 597
設定エラーのトラブルシューティング 598
設定エラー: macOS で AWS CLI 認証情報を設定できません 598
コンソールエラーのトラブルシューティング 598
アクセスエラー: コンソールまたは から CodeCommit リポジトリの暗号化キーアクセスが拒否されました AWS CLI 597
暗号化エラー:リポジトリを復号できない 599
コンソールエラー: コンソールから CodeCommit リポジトリのコードを参照できない 600
表示エラー: ファイルまたはファイル間の比較を表示できません 600
トリガーのトラブルシューティング 600
トリガーエラー: リポジトリのトリガーが、想定されているとおりに実行されない 600
デバッグを有効にする 601
CodeCommit リファレンス 603
リージョンと Git 接続エンドポイント 603
CodeCommit AWS リージョン でサポート 604
Git 接続エンドポイント 605
CodeCommit のサーバーフィンガープリント 612
インターフェイス VPC エンドポイント AWS CodeCommit での の使用 619
可用性 620
CodeCommit 用の VPC エンドポイントを作成する 621
CodeCommit 用の VPC エンドポイントポリシーを作成する 622
クォータ 623
コマンドラインリファレンス 631
基本的な Git コマンド 637
設定変数 637
リモートリポジトリ 638

コミット 640
ブランチ 641
[タグ] 642
ドキュメント履歴 644
以前の更新 655
AWS 用語集 662

AWS CodeCommit は新規顧客には利用できなくなりました。 AWS CodeCommit の既存のお客様は、通常どおりサービスを引き続き使用できます。詳細はこちら

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。

とは AWS CodeCommit

AWS CodeCommit は、Amazon Web Services がホストするバージョン管理サービスで、アセット (ドキュメント、ソースコード、バイナリファイルなど) をクラウドにプライベートに保存および管理するために使用できます。CodeCommit の料金については、料金を参照してください。

> **Note**
>
> CodeCommit は多くのコンプライアンスプログラムの範囲内です。 AWS とコンプライアンスの取り組みの詳細については、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」を参照してください。
> これは HIPAA 対象サービスです。1996 AWS年米国医療保険の相互運用性と説明責任に関する法律 (HIPAA)、および AWS サービスを使用して保護医療情報 (PHI) を処理、保存、および送信する方法の詳細については、「HIPAA 概要」を参照してください。
> このサービスについて、また、セキュリティ管理のベストプラクティスを規定したセキュリティ管理規格である ISO 27001 については、ISO 27001 の概要を参照してください。
> このサービスの詳細および Payment Card Industry Data Security Standard (PCI DSS) については、PCI DSS の概要を参照してください。
> このサービスの詳細、および機密情報を保護する暗号化モジュールのセキュリティ要件を指定する連邦情報処理規格 (FIPS) 出版物 140-2 の米国政府規格の詳細については、連邦情報処理規格 (FIPS) 140-2 の概要および Git 接続エンドポイント を参照してください。

トピック

- CodeCommit の概要
- CodeCommit、Git、ニーズに合った AWS サービスの選択
- CodeCommit の仕組み
- Amazon S3 でのファイルのバージョニングと CodeCommit の違い
- CodeCommit の使用を開始するにはどうしたらいいですか?
- Git の詳細情報

CodeCommit の概要

CodeCommit は、プライベート Git リポジトリをホストする、安全で高度にスケーラブルなマネージド型のソース管理サービスです。CodeCommit によって、インフラストラクチャのスケーリン

グに関する不安要素は排除され、お客様が独自のソース管理システムを管理する必要がなくなります。CodeCommit を使用して、コードからバイナリまで何でも保存できます。Git の標準機能がサポートされているため、既存の Git ベースのツールをシームレスに使用できます。

CodeCommit を使用すると、次のことが可能です。

- によってホストされる完全マネージド型サービスのメリット。 AWSCodeCommit では、サービスの高可用性および高耐久性を実現しているため、独自のハードウェアおよびソフトウェアの管理オーバーヘッドは必要ありません。ハードウェアのプロビジョニングおよびスケーリングはもちろん、サーバーソフトウェアのインストール、設定、アップデートも不要です。
- コードを安全に保存します。CodeCommit リポジトリは、不使用時および転送中に暗号化されます。
- 協力してコード作業を行います。CodeCommit リポジトリは、ブランチへマージする前に互いのコード変更を確認してそれにコメントできるプルリクエスト、プルリクエストやコメントについてユーザーに自動的に E メールを送る通知、その他多くをサポートしています。
- バージョン管理プロジェクトを簡単にスケール CodeCommit リポジトリは、スケールアップして開発のニーズを満たすことができます。このサービスでは、多数のファイルやブランチ、大容量のファイル、長期間のバージョン履歴に対応してリポジトリを管理できます。
- あらゆるものをいつでも保存できます。CodeCommit には、保存できるリポジトリのサイズやファイルのタイプに関する制限がありません。
- を他の AWS およびサードパーティーサービスと統合します。CodeCommit は、リポジトリを AWS クラウド内の他の本番稼働用リソースの近くに保持するため、開発ライフサイクルの速度と頻度が向上します。IAM と統合されており、他の AWS サービスや他のリポジトリと並行して使用できます。詳細については、「との製品とサービスの統合 AWS CodeCommit」を参照してください。
- 他のリモートリポジトリからファイルを簡単に移行可能 任意の Git ベースのリポジトリから CodeCommit に移行できます。
- なじみのある Git ツールを使用できます。CodeCommit は Git コマンドと独自の AWS CLI コマンドと APIsをサポートしています。

CodeCommit、Git、ニーズに合った AWS サービスの選択

Git ベースのサービスとして、CodeCommit はほとんどのバージョン管理ニーズに非常に適しています。ファイルサイズ、ファイルタイプ、およびリポジトリサイズに任意の制限はありません。ただし、Git には固有の制限があり、特に時間の経過とともにある種のオペレーションのパフォーマンス

に悪影響を及ぼす可能性があります。CodeCommit リポジトリのパフォーマンスが低下する可能性を回避するには、他の AWS サービスがタスクに適しているユースケースでの使用を回避します。複雑なリポジトリの Git のパフォーマンスを最適化することもできます。ここでは、Git、したがって CodeCommit が最善の解決策ではない場合や、Git を最適化するために追加のステップを実行する必要がある場合があるユースケースをいくつか紹介します。

ユースケース	説明	検討対象のその他のサービス
頻繁に変更される大容量ファイル	Git はデルタエンコードを使用してファイルのバージョン間の違いを保存します。たとえば、ドキュメント内のいくつかの単語を変更した場合、Git はそれらの変更された単語のみを保存します。5 MB を超えるファイルやオブジェクトがたくさん変更されている場合、Git は大きな一連のデルタ差分を再構築する必要がある場合があります。これらのファイルが時間の経過とともに大きくなるにつれて、これはローカルコンピュータ上および CodeCommit 内の両方でますます多くの計算リソースを消費する可能性があります。	大きなファイルをバージョニングするには、Amazon Simple Storage Service (Amazon S3) を検討してください。詳細については、Amazon Simple Storage Service ユーザーガイドのバージョニングの使用を参照してください。
データベース	Git リポジトリは時間とともに大きくなります。バージョニングはすべての変更を追跡するため、変更を加えるとリポジトリのサイズが大きくなります。つまり、データをコミットすると、コミットでデータを削除してもデータはリポジトリに追加されます。処理して送信するデータが増	サイズに関係なく一貫したパフォーマンスでデータベースを作成して使用するには、Amazon DynamoDB を検討してください。詳細については、Amazon DynamoDB の開始方法のガイドを参照してください。

ユースケース	説明	検討対象のその他のサービス
	えるにつれて、Git は遅くなります。これはデータベースのユースケースにとって特に悪影響を与えます。Git はデータベースとして設計されていません。	
監査証跡	通常、監査証跡は長期間保存され、非常に頻繁にシステムプロセスによって継続的に生成されます。Git は開発サイクルで開発者のグループによって生成されたソースコードを安全に保存するように設計されました。プログラムによって生成されたシステムの変更を継続的に保存する急速に変化するリポジトリは、時間の経過とともにパフォーマンスが低下します。	監査証跡を保存するには、Amazon Simple Storage Service (Amazon S3) を検討してください。 AWS アクティビティを監査するには、ユースケースに応じて、、AWS Config、または Amazon CloudWatch AWS CloudTrailの使用を検討してください。

ユースケース	説明	検討対象のその他のサービス
バックアップ	Git は開発者によって作成されたソースコードをバージョン管理するように設計されました。バックアップ戦略として、CodeCommit リポジトリを含む 2 つのリモートリポジトリにコミットをプッシュできます。ただし、Git は、コンピュータのファイルシステムのバックアップ、データベースのダンプ、または同様のバックアップコンテンツを処理するようには設計されていません。これを行うと、システムの速度が低下し、リポジトリのクローン作成とプッシュに必要な時間が長くなる可能性があります。	AWS クラウドへのバックアップの詳細については、「バックアップと復元」を参照してください。

ユースケース	説明	検討対象のその他のサービス

大量のブランチまたはリファレンス	Git クライアントがリポジトリデータをプッシュまたはプルするとき、たとえ単一のブランチが必要な場合も、リモートサーバーは、タグなどの、すべてのブランチとリファレンスを送る必要があります。何千ものブランチやリファレンスがある場合、処理と送信に時間がかかり (パックネゴシエーション)、リポジトリの応答が遅くなる可能性があります。ブランチやタグが多いほど、この処理にかかる時間が長くなります。CodeCommit の使用をお勧めしますが、不要になったブランチやタグは削除してください。	必要ではないものを判別するために CodeCommit リポジトリのリファレンスの数を分析するためには、次のいずれかのコマンドを使用できます。
		• Linux、macOS、Unix、または Windows 上の Bash エミュレーター:
		`git ls-remote | wc -l`
		• Powershell:
		`git ls-remote | Measure-Object -line`

CodeCommit の仕組み

Git ベースのリポジトリのユーザーは CodeCommit を使い慣れていますが、それ以外のユーザーでも CodeCommit への移行は比較的簡単です。CodeCommit のコンソールを使用すると、リポジトリの作成に加え、既存のリポジトリやブランチの一覧表示を簡単に行うことができます。ユーザーは、簡単な数ステップのみで、リポジトリに関する情報を表示し、そのクローンをコンピュータに作成できます。これにより、変更を加えることができるローカルリポジトリを作成して、CodeCommit リポジトリにプッシュできるようになります。ユーザーは、ローカルマシンのコマンドラインから操作するか、GUI ベースのエディタを使用してこの作業を行うことができます。

次の図は、開発マシン、 AWS CLI または CodeCommit コンソール、および CodeCommit サービスを使用してリポジトリを作成および管理する方法を示しています。

1. AWS CLI または CodeCommit コンソールを使用して CodeCommit リポジトリを作成します。
2. 開発マシンから Git を使用して、git clone を使用し、CodeCommit リポジトリの名前を指定します。これにより、CodeCommit リポジトリに接続するローカルリポジトリが作成されます。
3. 開発マシン上でローカルリポジトリを使用してファイルを変更 (追加、編集、削除) し、続いて、git add を実行し、変更したファイルをローカルでステージングします。git commit を実行してファイルをローカルにコミットしてから、git push を実行して、そのファイルを CodeCommit リポジトリに送信します。

4. 他のユーザーの変更をダウンロードします。git pull を実行して、CodeCommit リポジトリ内のファイルをローカルリポジトリと同期させます。これにより、最新バージョンのファイルを操作できます。

AWS CLI または CodeCommit コンソールを使用して、リポジトリを追跡および管理できます。

Amazon S3 でのファイルのバージョニングと CodeCommit の違い

CodeCommit は、チームによるソフトウェア開発のために最適化されています。また、複数のファイルの変更のバッチを管理できるため、他の開発者が操作していても、同時に変更することができます。Amazon S3 のバージョニングでは、過去のバージョンのファイルを復元できますが、ソフトウェア開発チームが必要とする提携ファイルの追跡機能は搭載されていません。

CodeCommit の使用を開始するにはどうしたらいいですか?

CodeCommit の使用を開始するには:

1. 「設定 」の手順に従って、開発マシンを準備します。
2. の 1 つ以上のチュートリアルの手順に従います入門
3. CodeCommit でバージョン管理プロジェクトを作成するか、バージョン管理プロジェクトを CodeCommit に移行します。

Git の詳細情報

詳細をお知りになりたい場合は、「Git の詳細」を参照してください。次に便利なリソースをいくつか紹介します。

- Pro Git: Pro Git の本のオンラインバージョン。執筆者は Scott Chacon です。発行元は Apress です。
- Git Immersion: Git の基礎を紹介したガイド付きツアー。発行元は Neo Innovation, Inc. です。
- Git リファレンス: オンラインのクイックリファレンス。詳細な Git チュートリアルとして使用することもできます。発行元は GitHub です。
- Git チートシート: Git コマンドの基本構文。発行元は GitHub です。
- Git ポケットガイド。執筆者 Richard E. Silverman です。発行元は O'Reilly Media, Inc です。

のセットアップ AWS CodeCommit

にサインイン AWS Management Console し、コンソールから直接リポジトリにファイルをアップロード、追加、編集できます。 AWS CodeCommit これは、素早く変更を加える方法です。ただし、複数のファイルの操作、ブランチ間のファイルなどを行う場合には、ローカルコンピューターを動作してリポジトリで作業することを考慮してください。CodeCommit を設定する最も簡単な方法は、HTTPS Git 認証情報を設定することです AWS CodeCommit。この HTTPS 認証方法は以下のとおりです。

- 静的なユーザー名とパスワードを使用します。
- CodeCommit でサポートされているすべてのオペレーティングシステムで機能します。
- Git 認証情報をサポートする統合開発環境 (IDE) やその他の開発ツールとも互換性があります。

運用上の理由から Git 認証情報を使用したくない、または使用できない場合は、他の方法を使用できます。例えば、フェデレーティッドアクセス、一時的な認証情報、またはウェブ ID プロバイダーを使用して CodeCommit リポジトリにアクセスする場合、Git 認証情報は使用できません。`git-remote-codecommit` コマンドを使用してローカルコンピュータを設定することをお勧めします。以下の選択肢を確認し、お客様に最適な代替方法を決定してください。

- Git 認証情報を使用した設定
- 他の方法を使用した設定
- CodeCommit、Git、および他のコンポーネントの互換性

CodeCommit と Amazon Virtual Private Cloud の使用については、インターフェイス VPC エンドポイント AWS CodeCommit での の使用 を参照してください。

認証情報の表示と管理

CodeCommit 認証情報は、 AWS コンソールから My Security Credentials を使用して表示および管理できます。

> **Note**
>
> フェデレーティッドアクセス、一時的な認証情報、またはウェブ ID プロバイダーを使用するユーザーは、このオプションを使用できません。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。
2. 右上のナビゲーションバーでユーザー名を選択し、続いて [認証情報] を選択します。
3. [AWS CodeCommit 認証情報] タブを選択します。

Git 認証情報を使用した設定

HTTPS 接続と Git 認証情報を使用すると、IAM で静的ユーザー名とパスワードが生成されます。これらの認証情報は、Git だけでなく、Git のユーザー名およびパスワード認証をサポートするサードパーティーツールでも使用します。この方法は、ほとんどの IDE と開発ツールでサポートされています。また、CodeCommit で使用する最もシンプルで簡単な接続方法です。

- Git 認証情報を使用した HTTPS ユーザーのセットアップ: Git 認証情報を使用して、ローカルコンピュータと CodeCommit リポジトリとの間の接続を設定するには、これらの指示に従ってください。
- 開発ツールから接続する場合: Git 認証情報を使用して、IDE または他の開発ツールと CodeCommit リポジトリとの間の接続を設定するには、これらのガイドラインに従ってください。Git 認証情報をサポートする IDEs には、Visual Studio、Xcode、IntelliJ が含まれます (ただし、これらに限定されません) 。

他の方法を使用した設定

HTTPS ではなく SSH プロトコルを使用して、CodeCommit リポジトリに接続できます。SSH 接続では、SSH 認証用に Git および CodeCommit が使用するパブリックキーとプライベートキーのファイルをローカルマシンで作成します。パブリックキーを IAM ユーザーに関連付けます。プライベートキーをローカルマシンに保存します。SSH では、パブリックキーとプライベートキーのファイルを手動で作成し管理する必要があるため、CodeCommit で使用するには、Git 認証情報がよりシンプルで使いやすい場合があります。

Git 認証情報とは異なり、SSH 接続設定は、ローカルコンピュータのオペレーティングシステムによって異なります。

- を使用していない SSH ユーザーの場合 AWS CLI: すでにパブリック/プライベートキーペアがあり、ローカルコンピュータの SSH 接続に精通している場合は、これらの指示 (要約) に従ってください。

- Linux、macOS、または Unix での SSH 接続の場合: パブリック/プライベートキーペアを作成し、Linux、macOS、または Unix オペレーティングシステムで接続を設定する手順については、これらの指示に従ってください。
- Windows で SSH 接続をセットアップする手順: パブリック/プライベートキーペアを作成し、Windows オペレーティングシステムで接続を設定する手順については、これらの指示に従ってください。

CodeCommit に接続し、フェデレーティッドアクセス、ID プロバイダー、または一時的な認証情報 AWS を使用している場合、または IAM ユーザーまたは IAM ユーザーの Git 認証情報を設定しない場合は、次のいずれかの方法で CodeCommit リポジトリへの接続を設定できます。

- git-remote-codecommit をインストールして使用します (推奨)。
- AWS CLIに含まれる認証情報ヘルパーをインストールして使用します。

どちらの方法でも、IAM ユーザーを必要とすることなく CodeCommit リポジトリへのアクセスがサポートされます。つまり、フェデレーティッドアクセスと一時的な認証情報を使用してリポジトリに接続できます。git-remote-codecommit ユーティリティが、推奨のアプローチです。このユーティリティは Git を拡張し、さまざまな Git バージョンや認証情報ヘルパーと互換性があります。ただし、すべての IDE が `git-remote-codecommit` で使用されるクローン URL 形式をサポートしているわけではありません。IDE でリポジトリを操作するには、リポジトリのクローンをローカルコンピュータに対して手動で作成する必要があります。

- git-remote-codecommit を使用して AWS CodeCommit リポジトリへの HTTPS 接続のセットアップステップの手順に従って、Windows、Linux、macOS、または Unix git-remote-codecommitにインストールしてセットアップします。

に含まれる認証情報ヘルパー AWS CLI を使用すると、Git は CodeCommit リポジトリとやり取り AWS するために で認証する必要があるたびに、HTTPS と暗号化で署名されたバージョンの IAM ユーザー認証情報または Amazon EC2 インスタンスロールを使用できます。一部のオペレーティングシステムと Git バージョンには、独自の認証情報ヘルパーがあり、 AWS CLIに含まれる認証情報ヘルパーと競合します。そのため、CodeCommit の接続に問題が発生する可能性があります。

- AWS CLI 認証情報ヘルパーを使用した Linux、macOS、または Unix での HTTPS 接続の場合: Linux、macOS、または Unix システムに認証情報ヘルパーをインストールして設定するには、これらの指示に従ってください。

- AWS CLI 認証情報ヘルパーを使用した Windows での HTTPS 接続の場合: Windows システムに認証情報ヘルパーをインストールして設定するには、これらの指示に従ってください。

他の Amazon Web Services アカウントにホストされている CodeCommit リポジトリに接続する場合には、 AWS CLIに含まれるロール、ポリシーおよび認証情報ヘルパーを使用してアクセスを設定し、接続をセットアップできます。

- ロールを使用して AWS CodeCommit リポジトリへのクロスアカウントアクセスを設定する: 以下のチュートリアルの手順に従って、1 つの Amazon Web Services アカウントから他の Amazon Web Services アカウントの IAM グループのユーザーにクロスアカウントアクセスを設定します。

CodeCommit、Git、および他のコンポーネントの互換性

CodeCommit を操作するときには、Git を使用します。他のプログラムを使用することもできます。以下の表では、バージョン互換性に関する最新のガイダンスを示します。ベストプラクティスとして、最新バージョンの Git やその他のソフトウェアを使用することをお勧めします。

のバージョン互換性情報 AWS CodeCommit

コンポーネント	バージョン
Git	CodeCommit は Git バージョン 1.7.9 以降をサポートしています。Git バージョン 2.28 は、初期コミットのブランチ名の設定をサポートしています。最新バージョンの Git を使用することをお勧めします。
Curl	CodeCommit には curl 7.33 以降が必要です。ただし、HTTPS と curl update 7.41.0 には既知の問題があります。詳細については、「トラブルシューティング」を参照してください。
Python (git-remote-codecommit のみ)	git-remote-codecommit にはバージョン 3 以降が必要です。
Pip (git-remote-codecommit のみ)	git-remote-codecommit には、バージョン 9.0.3 以降が必要です。

コンポーネント	バージョン
AWS CLI （ git-remote-codecommit のみ ）	すべての CodeCommit ユーザーには、最新バージョンの AWS CLI バージョン 2 をお勧めします。 では、フェデレーティッドユーザーなどの一時的な認証情報を必要とする SSO と接続をサポートするために AWS CLI AWS バージョン 2 git-remote-codecommitが必要です。

Git 認証情報を使用した HTTPS ユーザーのセットアップ

AWS CodeCommit リポジトリへの接続を設定する最も簡単な方法は、IAM コンソールで CodeCommit の Git 認証情報を設定し、それらの認証情報を HTTPS 接続に使用することです。静的なユーザー名とパスワードを使用して HTTPS 認証をサポートするサードパーティーのツールまたは統合開発環境 (IDE) でも、同じ認証情報を使用できます。例については、「開発ツールから接続する場合」を参照してください。

> **Note**
>
> 以前に CodeCommit 用の信任状ヘルパーを使用するようにローカルコンピュータを構成していた場合、Git 認証情報を使用する前に .gitconfig ファイルを編集してファイルから認証ヘルパー情報を削除する必要があります。ローカルコンピュータが macOS を実行している場合は、Keychain Access からキャッシュされた認証情報をクリアする必要があります。

ステップ 1: CodeCommit の初期設定

アマゾン ウェブ サービスアカウントを設定し、IAM ユーザーを作成して、CodeCommit へのアクセスを設定するには、以下の手順に従います。

IAM ユーザーを作成および設定して CodeCommit にアクセスするには

1. アマゾン ウェブ サービスアカウントを作成するには、http://aws.amazon.com にアクセスし、[Sign Up] (サインアップ) を選択します。
2. IAM ユーザーを作成するか、アマゾン ウェブ サービスアカウントに関連付けられた既存のユーザーを使用します。アクセスキー ID およびシークレットアクセスキーがその IAM ユーザーに関

連付けられていることを確認します。詳細については、アマゾン ウェブ サービスアカウントの IAM ユーザーの作成を参照してください。

> **Note**
>
> CodeCommit には が必要です AWS Key Management Service。既存の IAM ユーザーを使用している場合は、CodeCommit に必要な AWS KMS アクションを明示的に拒否するポリシーがユーザーにアタッチされていないことを確認してください。詳細については、「AWS KMS および 暗号化」を参照してください。

3. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。
4. IAM コンソールのナビゲーションペインで、[Users] (ユーザー) を選択し、続いて、CodeCommit へアクセスするために設定する IAM ユーザーを選択します。
5. [Permissions (アクセス許可)] タブで、[Add Permissions (アクセス許可の追加)] を選択します。
6. [Grant permissions (アクセス許可の付与)] で、[Attach existing policies directly (既存のポリシーを直接アタッチする)] を選択します。
7. ポリシーの一覧から、[AWSCodeCommitPowerUser] または CodeCommit アクセスの別の管理ポリシーを選択します。詳細については、「AWS CodeCommit の マネージドポリシー」を参照してください。

   アタッチするポリシーを選択したら、[Next: Review] (次へ: 確認) を選択して、IAM ユーザーにアタッチするポリシーのリストを表示します。リストが正しい場合は、[Add permissions (アクセス許可の追加)] を選択します。

   CodeCommit 管理ポリシーや、その他のグループおよびユーザーを含むリポジトリへのアクセス共有の詳細については、リポジトリの共有 および AWS CodeCommitの認証とアクセスコントロール を参照してください。

CodeCommit で AWS CLI コマンドを使用する場合は、 をインストールします AWS CLI。CodeCommit AWS CLI で を使用するためのプロファイルを作成することをお勧めします。詳細については、コマンドラインリファレンス「」および「名前付きプロファイルの使用」を参照してください。

ステップ 2: Git をインストールする

CodeCommit リポジトリのファイル、コミット、およびその他の情報を使用するには、ローカルマシンに Git をインストールする必要があります。CodeCommit は Git バージョン 1.7.9 以降をサポートしています。Git バージョン 2.28 は、初期コミットのブランチ名の設定をサポートしています。最新バージョンの Git を使用することをお勧めします。

Git をインストールするには、Git のダウンロードなどのウェブサイトをお勧めします。

> **Note**
>
> Git は、定期的に更新されている、発展中のプラットフォームです。機能の変更により、CodeCommit での動作が影響を受ける場合があります。特定のバージョンの Git と CodeCommit で問題が発生した場合は、 の情報を確認してくださいトラブルシューティング

ステップ 3: CodeCommit への HTTPS 接続用の Git 認証情報を作成する

Git をインストールした後、IAM の IAM ユーザー用の Git 認証情報を作成します。

CodeCommit の HTTPS Git の認証情報を設定するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。

   CodeCommit への接続に Git 認証情報を作成して使用する IAM ユーザーとしてサインインするようにしてください。

2. IAM コンソールのナビゲーションペインで [Users] (ユーザー) を選択し、ユーザーのリストから自分の IAM ユーザーを選択します。

   > **Note**
   >
   > CodeCommit の認証情報を [My Security Credentials] (セキュリティ認証情報) で直接表示および管理できます。詳細については、「認証情報の表示と管理」を参照してください。

3. ユーザーの詳細ページでセキュリティ認証情報タブを選択し、HTTPS Git 認証情報 AWS CodeCommitで生成を選択します。

> **Note**
>
> Git 認証情報用のユーザー名やパスワードは選択できません。詳細については、CodeCommit で Git 認証情報と HTTPS を使用するを参照してください。

4. IAM が生成したユーザー名とパスワードをコピーするには、この情報を表示して、ローカルコンピュータ上の安全なファイルにコピーアンドペーストするか、[Download credentials] (認証情報のダウンロード) を選択して .CSV ファイルとしてこの情報をダウンロードします。CodeCommit に接続するには、この情報が必要です。

認証情報を保存したら、[Close] を選択します。

> **Important**
>
> これは、ユーザー名とパスワードを保存する唯一の機会です。パスワードを保存しないと、IAM コンソールからユーザー名をコピーすることはできますが、パスワードを参照することはできません。パスワードをリセットして保存する必要があります。

ステップ 4: CodeCommit コンソールに接続し、リポジトリのクローンを作成する

管理者から CodeCommit リポジトリの名前と接続の詳細をすでに受け取っている場合は、このステップをスキップしてリポジトリを直接複製できます。

CodeCommit リポジトリに接続するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタで、リポジトリが作成された AWS リージョン を選択します。リポジトリは に固有です AWS リージョン。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。
3. 接続するリポジトリをリストから見つけて選択します。[クローン URL] を選択してから、リポジトリのクローン作成やリポジトリへの接続時に使用するプロトコルを選択します。これにより、クローン URL が複製されます。
   - IAM ユーザー、または AWS CLIに含まれている認証情報ヘルパーで Git 認証情報を使用している場合は、HTTPS URL をコピーします。
   - ローカルコンピュータで git-remote-codecommit コマンドを使用している場合は、HTTPS (GRC) URL をコピーします。
   - IAM ユーザーで SSH パブリック/プライベートキーペアを使用している場合は、SSH URL をコピーします。

> **Note**
>
> リポジトリのリストの代わりにようこそページが表示された場合は、サインイン AWS リージョン している に AWS アカウントに関連付けられたリポジトリはありません。

リポジトリを作成するには、「the section called “リポジトリの作成”」を参照するか、「Git と CodeCommit の開始方法」チュートリアルのステップに従います。

4. ターミナル、コマンドライン、または Git シェルを開きます。リポジトリのクローンを作成するためにコピーした HTTPS クローン URL を指定して、git clone コマンドを実行します。For example, to clone a repository named *MyDemoRepo* to a local repo named *my-demo-repo* in the US East (Ohio) Region:

```
git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-
demo-repo
```

初めて接続すると、リポジトリのユーザー名とパスワードの入力を求められます。ローカルコンピュータの設定に応じて、このプロンプトは、オペレーティングシステム の認証情報管理システム、Git のバージョン用の認証情報管理ユーティリティ (例えば、Git for Windows に含まれる Git Credential Manager)、IDE、または Git 自体から生成されます。IAM の Git 認証情報用に生成されたユーザー名とパスワードを入力します (ステップ 3: CodeCommit への HTTPS 接続用の Git 認証情報を作成する で作成したもの)。ご使用のオペレーティングシステムおよびその他のソフトウェアによっては、この情報が認証情報ストアまたは認証情報管理ユーティリティに保存される場合があります。その場合は、パスワードを変更したり、Git 認証情報を無効にしたり、IAM の Git 認証情報を削除したりしない限り、再度入力する必要はありません。

ローカルコンピュータに認証情報ストアまたは認証情報管理ユーティリティが構成されていない場合は、それらをインストールできます。Git とその資格証明を管理する方法の詳細については、Git ドキュメントの「認証情報ストレージ」を参照してください。

詳細については、「CodeCommit リポジトリのクローンを作成してリポジトリに接続する」および「コミットを作成する」を参照してください。

次のステップ

前提条件を完了しました。CodeCommit の開始方法 のステップに従って、CodeCommit の使用を開始してください。

最初のコミットを作成してプッシュする方法については、「でコミットを作成する AWS CodeCommit」を参照してください。Git を初めて利用する場合は、Git の詳細情報 および Git と の開始方法 AWS CodeCommit でも情報を確認できます。

git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順

ルートアカウント、フェデレーティッドアクセス、または一時的な認証情報を使用して CodeCommit に接続する場合は、git-remote-codecommit を使用してアクセスを設定する必要があります。このユーティリティは、Git を拡張することにより、CodeCommit リポジトリからコードをプッシュおよびプルするための簡単な方法を提供します。これは、フェデレーティッドアクセス、ID プロバイダー、および一時的な認証情報を使用した接続をサポートするために推奨される方法です。フェデレーテッドアイデンティティ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) 用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、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。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。

IAM ユーザーで git-remote-codecommit を使用することもできます。他の HTTPS 接続方法とは異なり、git-remote-codecommit では、ユーザーの Git 認証情報を設定する必要はありません。

> **Note**
>
> 一部の IDE は、`git-remote-codecommit` で使用されるクローン URL 形式をサポートしていません。任意の IDE でリポジトリを操作する前に、ローカルコンピュータにリポジトリのクローンを手動で作成する必要が生じる場合があります。詳細については、「git-remote-codecommit と のトラブルシューティング AWS CodeCommit」を参照してください。

これらの手順は、アマゾン ウェブ サービスアカウントがあり、CodeCommit で少なくとも 1 つのリポジトリを作成しており、CodeCommit リポジトリに接続するときに管理ポリシーを持つ IAM ユーザーを使用することを前提としています。フェデレーティッドユーザーおよびその他の認証情報の更新タイプのアクセスを設定する方法については、「認証情報をローテーションして AWS CodeCommit リポジトリに接続する」を参照してください。

トピック

- ステップ 0: git-remote-codecommit の前提条件をインストールする
- ステップ 1: CodeCommit の初期設定

- ステップ 2: git-remote-codecommit をインストールする
- ステップ 3: CodeCommit コンソールに接続し、リポジトリのクローンを作成する
- 次のステップ

ステップ 0: git-remote-codecommit の前提条件をインストールする

git-remote-codecommit を使用する前に、ローカルコンピュータにいくつかの前提条件をインストールする必要があります。具体的には次のとおりです。

- サポートされているバージョンの Python。サポートされている Python のバージョンの詳細については、git-remote-codecommit」を参照してください。

  git-remote-codecmit の設定と使用の詳細については、「git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」を参照してください。
- Git

> **Note**
>
> Windows に Python をインストールする場合は、必ず Python をパスに追加するオプションを選択してください。

git-remote-codecommit には pip バージョン 9.0.3 以降が必要です。pip のバージョンを確認するには、ターミナルまたはコマンドラインを開き、次のコマンドを実行します。

```
pip --version
```

次の 2 つのコマンドを実行して、pip のバージョンを最新バージョンに更新できます。

```
curl -O https://bootstrap.pypa.io/get-pip.py
python3 get-pip.py --user
```

CodeCommit リポジトリのファイル、コミット、およびその他の情報を使用するには、ローカルマシンに Git をインストールする必要があります。CodeCommit は Git バージョン 1.7.9 以降をサポートしています。Git バージョン 2.28 は、初期コミットのブランチ名の設定をサポートしています。最新バージョンの Git を使用することをお勧めします。

Git をインストールするには、Git のダウンロードなどのウェブサイトをお勧めします。

> **Note**
>
> Git は、定期的に更新されている、発展中のプラットフォームです。機能の変更により、CodeCommit での動作が影響を受ける場合があります。特定のバージョンの Git と CodeCommit で問題が発生した場合は、 の情報を確認してくださいトラブルシューティング

ステップ 1: CodeCommit の初期設定

次の手順に従って IAM ユーザーを作成し、適切なポリシーを使用してユーザーを設定します。次に、アクセスキーとシークレットキーを取得し、 AWS CLIをインストールして設定します。

IAM ユーザーを作成および設定して CodeCommit にアクセスするには

1. アマゾン ウェブ サービスアカウントを作成するには、http://aws.amazon.com にアクセスし、[Sign Up] (サインアップ) を選択します。
2. IAM ユーザーを作成するか、アマゾン ウェブ サービスアカウントに関連付けられた既存のユーザーを使用します。アクセスキー ID およびシークレットアクセスキーがその IAM ユーザーに関連付けられていることを確認します。詳細については、アマゾン ウェブ サービスアカウントの IAM ユーザーの作成を参照してください。

   > **Note**
   >
   > CodeCommit には が必要です AWS Key Management Service。既存の IAM ユーザーを使用している場合は、CodeCommit に必要な AWS KMS アクションを明示的に拒否するポリシーがユーザーにアタッチされていないことを確認してください。詳細については、「AWS KMS および 暗号化」を参照してください。

3. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。
4. IAM コンソールのナビゲーションペインで、[Users] (ユーザー) を選択し、続いて、CodeCommit へアクセスするために設定する IAM ユーザーを選択します。
5. [Permissions (アクセス許可)] タブで、[Add Permissions (アクセス許可の追加)] を選択します。
6. [Grant permissions (アクセス許可の付与)] で、[Attach existing policies directly (既存のポリシーを直接アタッチする)] を選択します。

7. ポリシーの一覧から、[AWSCodeCommitPowerUser] または CodeCommit アクセスの別の管理ポリシーを選択します。詳細については、「AWS CodeCommit の マネージドポリシー」を参照してください。

   アタッチするポリシーを選択したら、[Next: Review] (次へ: 確認) を選択して、IAM ユーザーにアタッチするポリシーのリストを表示します。リストが正しい場合は、[Add permissions (アクセス許可の追加)] を選択します。

   CodeCommit 管理ポリシーや、その他のグループおよびユーザーを含むリポジトリへのアクセス共有の詳細については、リポジトリの共有 および AWS CodeCommitの認証とアクセスコントロール を参照してください。

をインストールして設定するには AWS CLI

1. ローカルマシンで、 をダウンロードしてインストールします AWS CLI。これは、コマンドラインから CodeCommit とやり取りするための前提条件です。 AWS CLI バージョン 2 のインストールが推奨されます。これは の最新バージョン AWS CLI であり、すべての最新機能をサポートしています。これは、 でルートアカウント、フェデレーティッドアクセス、または一時的な認証情報の使用 AWS CLI をサポートする の唯一のバージョンですgit-remote-codecommit。

   詳細については、AWS 「 コマンドラインインターフェイスのセットアップ」を参照してください。

   > **Note**
   >
   > CodeCommit は、 AWS CLI バージョン 1.7.38 以降でのみ動作します。ベストプラクティスとして、 をインストールまたは最新バージョンにアップグレード AWS CLI します。インストール AWS CLI した のバージョンを確認するには、 aws --version コマンドを実行します。
   > の古いバージョン AWS CLI を最新バージョンにアップグレードするには、「 のインストール AWS Command Line Interface」を参照してください。

2. このコマンドを実行して、 の CodeCommit コマンド AWS CLI がインストールされていることを確認します。

   ```
   aws codecommit help
   ```

   このコマンドは、CodeCommit コマンドのリストを返します。

3. 次のように、 configure コマンドを使用して プロファイル AWS CLI で を設定します。

```
aws configure
```

プロンプトが表示されたら、CodeCommit で使用する IAM ユーザーの AWS アクセスキーと AWS シークレットアクセスキーを指定します。また、 など、リポジトリが存在する AWS リージョン を必ず指定してください`us-east-2`。デフォルトの出力形式の入力を求められたら、`json` を指定します。例えば、IAM ユーザーのプロファイルを設定する場合は、次のようにします。

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

で使用するプロファイルの作成と設定の詳細については AWS CLI、以下を参照してください。

- 名前付きプロファイル
- での IAM ロールの使用 AWS CLI
- Set コマンド
- 認証情報をローテーションして AWS CodeCommit リポジトリに接続する

別のリポジトリまたはリソースに接続するには AWS リージョン、デフォルトのリージョン名 AWS CLI で を再設定する必要があります。CodeCommit でサポートされるデフォルトのリージョン名は以下のとおりです。

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2

- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

CodeCommit と の詳細については AWS リージョン、「」を参照してくださいリージョンと Git 接続エンドポイント。IAM、アクセスキー、シークレットキーに関する詳細については、認証情報を取得する方法および IAM ユーザーのアクセスキーの管理を参照してください。 AWS CLI および プロファイルの詳細については、「名前付きプロファイル」を参照してください。

ステップ 2: git-remote-codecommit をインストールする

git-remote-codecommit をインストールするには、次のステップに従ってください。

git-remote-codecommit をインストールするには

1. ターミナルまたはコマンドラインで、次のコマンドを実行します。

```
pip install git-remote-codecommit
```

> **Note**
>
> オペレーティングシステムと設定によっては、このコマンドを実行するために sudo などの昇格したアクセス許可を使用するか、--user パラメータを使用して、現在のユーザーアカウントなどの特別な特権を必要としないディレクトリにインストールすることが必要になる場合があります。例えば、Linux、macOS、または Unix を実行しているコンピューターの場合は、次のとおりです。
>
> ```
> sudo pip install git-remote-codecommit
> ```
>
> Windows を実行しているコンピュータの場合は、次のとおりです。
>
> ```
> pip install --user git-remote-codecommit
> ```

2. 成功メッセージが表示されるまで、インストールプロセスをモニタリングします。

ステップ 3: CodeCommit コンソールに接続し、リポジトリのクローンを作成する

管理者が CodeCommit リポジトリに対して git-remote-codecommit で使用するクローン URL を既に送信している場合は、コンソールへの接続をスキップして、リポジトリのクローンを直接作成できます。

CodeCommit リポジトリに接続するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタで、リポジトリ AWS リージョン が作成された を選択します。リポジトリは に固有です AWS リージョン。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。

3. 接続するリポジトリをリストから見つけて選択します。[クローン URL] を選択してから、リポジトリのクローン作成やリポジトリへの接続時に使用するプロトコルを選択します。これにより、クローン URL が複製されます。

   - IAM ユーザー、または AWS CLIに含まれている認証情報ヘルパーで Git 認証情報を使用している場合は、HTTPS URL をコピーします。
   - ローカルコンピュータで git-remote-codecommit コマンドを使用している場合は、HTTPS (GRC) URL をコピーします。
   - IAM ユーザーで SSH パブリック/プライベートキーペアを使用している場合は、SSH URL をコピーします。

   > **Note**
   >
   > リポジトリのリストではなくようこそページが表示された場合は、サインイン AWS リージョン している に AWS アカウントに関連付けられたリポジトリはありません。リポジトリを作成するには、「the section called “リポジトリの作成”」を参照するか、「Git と CodeCommit の開始方法」チュートリアルのステップに従います。

4. ターミナルまたはコマンドプロンプトで、git clone コマンドを使用してリポジトリのクローンを作成します。名前付き AWS CLI プロファイルを作成した場合は、コピーした HTTPS git-remote-codecommit URL とプロファイルの名前を使用します。プロファイルを指定しない場合は、デフォルトのプロファイルが使用されます。ローカルリポジトリは、そのコマンドを実行したディレクトリのサブディレクトリに作成されます。例えば、*MyDemoRepo* という名前のリポジトリのクローンを *my-demo-repo* という名前のローカルリポジトリに作成するには、次のようにします。

```
git clone codecommit://MyDemoRepo my-demo-repo
```

   CodeCommitProfile という名前のプロファイルを使用して同じリポジトリのクローンを作成するには、次の手順を実行します。

```
git clone codecommit://CodeCommitProfile@MyDemoRepo my-demo-repo
```

   プロファイルで設定されたリポジトリ AWS リージョン とは異なる でリポジトリのクローンを作成するには、 AWS リージョン 名を含めます。以下に例を示します。

```
git clone codecommit::ap-northeast-1://MyDemoRepo my-demo-repo
```

次のステップ

前提条件を完了しました。CodeCommit の開始方法 のステップに従って、CodeCommit の使用を開始してください。

最初のコミットを作成してプッシュする方法については、「でコミットを作成する AWS CodeCommit」を参照してください。Git を初めて利用する場合は、Git の詳細情報 および Git と の開始方法 AWS CodeCommit でも情報を確認できます。

Git 認証情報を使用して開発ツールからの接続を設定する

IAM コンソール AWS CodeCommit で の Git 認証情報を設定したら、Git 認証情報をサポートする任意の開発ツールでそれらの認証情報を使用できます。たとえば、、Visual Studio AWS Cloud9、Xcode、IntelliJ、または Git 認証情報を統合する統合開発環境 (IDE) で CodeCommit リポジトリへのアクセスを設定できます。アクセスを設定した後は、コードを編集し、変更をコミットし、IDE や他の開発ツールから直接プッシュすることができます。

> **Note**
>
> フェデレーションアクセス、一時的な認証情報、またはウェブ ID プロバイダーを使用して CodeCommit リポジトリにアクセスする場合は、Git 認証情報を使用できません。`git-remote-codecommit` コマンドを使用してローカルコンピュータを設定することをお勧めします。ただし、すべての IDE が git-remote-codecommit などの Git リモートヘルパーと完全に互換性があるわけではありません。問題が発生した場合は、git-remote-codecommit と のトラブルシューティング AWS CodeCommit を参照してください。

トピック

- AWS Cloud9 との統合 AWS CodeCommit
- Visual Studio を と統合する AWS CodeCommit

CodeCommit リポジトリに接続するために使用するユーザー名とパスワードを IDE または開発ツールから求められたら、IAM で作成した [User name] (ユーザー名) と [Password] (パスワード) の Git 認証情報を入力します。

CodeCommit の AWS リージョン および エンドポイントの詳細については、「」を参照してください[リージョンと Git 接続エンドポイント](#)。

ユーザー名とパスワードを保存するためのオペレーティングシステムからのプロンプトが表示される場合もあります。例えば、Windows では、次のように Git 認証情報を入力します。

特定のソフトウェアプログラムまたは開発ツール用の Git 認証情報の設定の詳細については、製品ドキュメントを参照してください。

以下は、IDE の包括的なリストではありません。リンクは、これらのツールの詳細を理解するためにのみ提供されています。 AWS は、これらのトピックのコンテンツについて責任を負いません。

- AWS Cloud9

- Visual Studio

  または、 をインストールします AWS Toolkit for Visual Studio。詳細については、「Visual Studio を と統合する AWS CodeCommit」を参照してください。

- XCode

AWS Cloud9 との統合 AWS CodeCommit

AWS Cloud9 を使用して CodeCommit リポジトリでコード変更を行うことができます。 には、コードの作成、ソフトウェアのビルド、実行、テスト、デバッグ、リリースに使用できるツールのコレクション AWS Cloud9 が含まれています。既存のリポジトリのクローン作成、リポジトリの作成、リポジトリへのコード変更のコミットとプッシュなどはすべて、 AWS Cloud9 EC2 開発環境から行うことができます。 AWS Cloud9 EC2 開発環境は通常 AWS CLI、、Amazon EC2 ロール、Git で事前設定されているため、ほとんどの場合、いくつかの簡単なコマンドを実行してリポジトリとのやり取りを開始できます。

CodeCommit AWS Cloud9 で を使用するには、以下が必要です。

- Amazon Linux で実行されている AWS Cloud9 EC2 開発環境。
- IDE AWS Cloud9 がウェブブラウザで開きます。
- CodeCommit 管理ポリシーの 1 つと、それに適用される AWS Cloud9 管理ポリシーの 1 つを持つ IAM ユーザー。

  詳細については、AWS CodeCommit の マネージドポリシー および セキュリティ認証情報の理解と取得を参照してください。

> **Note**
>
> このトピックでは、CodeCommit およびインターネットからの一般的なアクセス AWS Cloud9 との統合の設定について説明します。分離 AWS Cloud9 された環境で CodeCommit と へのアクセスを設定できますが、これには追加のステップが必要です。詳細については、以下を参照してください。
>
> - インターフェイス VPC エンドポイント AWS CodeCommit での の使用
> - を使用した no-ingress Amazon EC2 インスタンスへのアクセス AWS Systems Manager
> - 共有環境を使用する
> - VPC を他のアカウントと共有する
> - ブログ記事: AWS Cloud9 環境へのネットワークアクセスの分離

トピック

- ステップ 1: AWS Cloud9 開発環境を作成する
- ステップ 2: AWS Cloud9 EC2 開発環境で AWS CLI 認証情報ヘルパーを設定する

- ステップ 3: CodeCommit リポジトリを AWS Cloud9 EC2 開発環境にクローンする
- 次のステップ

ステップ 1: AWS Cloud9 開発環境を作成する

AWS Cloud9 は、Amazon EC2 インスタンスで開発環境をホストします。これは、インスタンスの AWS マネージドー時認証情報を使用して CodeCommit リポジトリに接続できるため、統合する最も簡単な方法です。代わりに独自のサーバーを使用する場合は、AWS Cloud9 ユーザーガイドを参照してください。

AWS Cloud9 環境を作成するには

1. 設定した IAM ユーザー AWS として にサインインし、 AWS Cloud9 コンソールを開きます。
2. AWS Cloud9 コンソールで、環境の作成を選択します。
3. [Step 1: Name environment] (ステップ 1: 環境に名前を付ける) で、環境の名前と説明 (オプション) を入力し、[Next step] (次のステップ) を選択します。
4. ステップ 2: 構成を設定するで、環境を次のように設定します。

   - [Environment type] (環境タイプ) で、[Create a new instance for environment (EC2)] (環境 (EC2) 用に新しいインスタンスを作成) を選択します。
   - [Instance type] (インスタンスタイプ) で、ご利用の開発環境に適したインスタンスタイプを選択します。例えば、サービスを探索するだけの場合は、デフォルトの t2.micro を選択できます。この環境を開発作業に使用する場合は、より大きなインスタンスタイプを選択します。
   - デフォルト以外の設定を使用する理由がある場合 (組織で特定の VPC を使用している場合、またはアマゾン ウェブ サービスアカウントで VPC が設定されていない場合など) を除き、他のデフォルト設定を受け入れて [Next step] (次のステップ) を選択します。
5. [Step 3: Review] (ステップ 3: 確認) で、設定を確認します。変更を加える場合は [Previous step] (前のステップ) を選択します。変更を加えない場合は [Create environment] (環境の作成) を選択します。

   環境を作成して初めて接続するまでには数分かかります。長くかかる場合は、AWS Cloud9 ユーザーガイドの「トラブルシューティング」を参照してください。
6. 環境に接続した後、ターミナルウィンドウで git --version コマンドを実行して、サポートされているバージョンの Git が既にインストールされているかどうかを確認します。

   Git がインストールされていないか、サポートされているバージョンでない場合、サポートされているバージョンをインストールします。CodeCommit は Git バージョン 1.7.9 以降をサポー

トしています。Git バージョン 2.28 は、初期コミットのブランチ名の設定をサポートしています。最新バージョンの Git を使用することをお勧めします。Git をインストールするには、Git のダウンロードなどのウェブサイトをお勧めします。

> **Tip**
>
> ご利用の環境のオペレーティングシステムによっては、yum オプションを指定して sudo コマンドを使用すると、Git を含む更新プログラムをインストールできる場合があります。例えば、管理コマンドのシーケンスは次の 3 つのコマンドのようになります。
>
> ```
> sudo yum -y update
> sudo yum -y install git
> git --version
> ```

7. git config コマンドを実行して、Git コミットと関連付けるユーザー名と E メールアドレスを設定します。以下に例を示します。

```
git config --global user.name "Mary Major"
git config --global user.email mary.major@example.com
```

ステップ 2: AWS Cloud9 EC2 開発環境で AWS CLI 認証情報ヘルパーを設定する

AWS Cloud9 環境を作成したら、認証情報 AWS CLI ヘルパーを設定して CodeCommit リポジトリへの接続の認証情報を管理できます。 AWS Cloud9 開発環境には、IAM ユーザーに関連付けられた AWS マネージドー時認証情報が付属しています。これらの認証情報は、 AWS CLI 認証情報ヘルパーで使用します。

1. ターミナルウィンドウを開き、次のコマンドを実行して AWS CLI がインストールされていることを確認します。

```
aws --version
```

成功した場合、このコマンドは現在インストールされている AWS CLIのバージョンを返します。の古いバージョン AWS CLI を最新バージョンにアップグレードするには、「 のインストール AWS Command Line Interface」を参照してください。

2. ターミナルで、次のコマンドを実行して、HTTPS 接続用の AWS CLI 認証情報ヘルパーを設定します。

```
git config --global credential.helper '!aws codecommit credential-helper $@'
git config --global credential.UseHttpPath true
```

> **Tip**
>
> 認証情報ヘルパーは、開発環境のデフォルトの Amazon EC2 インスタンスロールを使用します。開発環境を使用して CodeCommit でホストされていないリポジトリに接続する予定の場合、それらのリポジトリへの SSH 接続を設定するか、それらの他のリポジトリに接続するときに代替認証情報管理システムが使用されるようにローカル `.gitconfig` ファイルを設定します。詳細については、Git ウェブサイトの Git Tools - Credential Storage を参照してください。

ステップ 3: CodeCommit リポジトリを AWS Cloud9 EC2 開発環境にクローンする

AWS CLI 認証情報ヘルパーを設定したら、CodeCommit リポジトリのクローンを作成できます。その後、コードで作業を開始できます。

1. ターミナルで、クローンを作成するリポジトリの HTTPS クローン URL を指定して git clone コマンドを実行します。例えば、MyDemoRepo という名前のリポジトリのクローンを米国東部 (オハイオ) リージョンに作成する場合は、次のように入力します。

```
git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
```

> **Tip**
>
> CodeCommit コンソールでリポジトリのクローン URL を確認するには、[Clone URL] (URL のクローン) を選択します。

2. クローンの作成が完了したら、リポジトリのフォルダをサイドナビゲーションで展開し、開いて編集するファイルを選択します。あるいは、[File] (ファイル) を選択し、[New File] (新しいファイル) を選択してファイルを作成します。

3. ファイルの編集または作成が終了したら、ターミナルウィンドウで、クローン作成されたリポジトリにディレクトリを変更し、変更をコミットしてプッシュします。例えば、*MyFile.py* という名前の新しいファイルを追加したとします。

```
cd MyDemoRepo
git commit -a MyFile.py
git commit -m "Added a new file with some code improvements"
git push
```

次のステップ

詳細については、AWS Cloud9 ユーザーガイドおよび「AWS Cloud9の CodeCommit ECR サンプル」を参照してください。CodeCommit で Git を使用する方法の詳細については、Git と の開始方法 AWS CodeCommit を参照してください。

Visual Studio を と統合する AWS CodeCommit

Visual Studio を使用して CodeCommit リポジトリでコードを変更できます。には、Visual Studio での CodeCommit の操作を簡単かつ便利にする機能 AWS Toolkit for Visual Studio が追加されました。Toolkit for Visual Studio の統合は、Git 認証情報および IAM ユーザーと連携するように設計されています。既存のリポジトリのクローン作成、リポジトリの作成、リポジトリへのコード変更のコミットとプッシュなどを行うことができます。

> **Important**
>
> Toolkit for Visual Studio は、Windows オペレーティングシステムでのみインストールできます。Visual Studio Code での作業に関する情報については、「AWS Toolkit for Visual Studio Code」を参照してください。

以前 Toolkit for Visual Studio を使用した経験がある場合は、おそらく既にアクセスキーとシークレットキーを含む AWS 認証情報プロファイルの設定についてはご存知でしょう。認証情報プロファイルは、 AWS Toolkit for Visual Studio でサービス APIs への呼び出しを有効にするために使用されます (Amazon S3 への呼び出しでバケットを一覧表示したり、CodeCommit への呼び出しでリポジトリを一覧表示したりするなど) 。コードをプルして CodeCommit リポジトリにプッシュするには、Git 認証情報も必要です。Git 認証情報がない場合は、Toolkit for Visual Studio でこれらの認証情報を生成して適用できます。これで大幅に時間を節約できます。

CodeCommit で Visual Studio を使用するには、次が必要です。

- 有効な認証情報 (アクセスキーとシークレットキー) を持つ IAM ユーザー。この IAM ユーザーには以下も必要です。

  CodeCommit 管理ポリシーと IAMSelfManageServiceSpecificCredentials 管理ポリシーの 1 つが適用されていること。

  または

  IAM ユーザーに既に Git 認証情報が設定されている場合、CodeCommit 管理ポリシーのいずれか、または同等のアクセス許可のいずれかです。

  詳細については、AWS CodeCommit の マネージドポリシー および セキュリティ認証情報の理解と取得を参照してください。
- Visual Studio AWS Toolkit for Visual Studio をインストールしたコンピュータにインストールされている 。詳細については、「AWS Toolkit for Visual Studioをセットアップする」を参照してください。

CodeCommit AWS Toolkit for Visual Studio で を使用する方法の詳細については、「 Toolkit for Visual Studio ユーザーガイド」の「Using AWS CodeCommit with Visual Studio Team Explorer」を参照してください。

を使用していない SSH ユーザーのセットアップ AWS CLI

リポジトリに SSH 接続を使用する場合は、 をインストール AWS CodeCommit せずに に接続できます AWS CLI。 AWS CLI には、CodeCommit リポジトリを使用および管理する際に便利なコマンドが含まれていますが、初期設定には必要ありません。

このトピックでは以下のことを前提としています。

- CodeCommit に必要なポリシーまたはアクセス許可、さらにキーのアップロードに必要な [IAMUserSSHKeys] 管理ポリシーまたは同等のアクセス許可を持つ IAM ユーザーが設定済みです。詳細については、「CodeCommit でのアイデンティティベースのポリシー (IAM ポリシー) の使用」を参照してください。
- パブリックキーとプライベートキーペアをすでに所有しているか、作成方法を知っています。SSH キーには安全なパスフレーズを使用することを強くお勧めします。
- SSH、Git クライアント、およびその設定ファイルに精通しています。

- Windows を使用している場合は、bash シェルをエミュレートするコマンド行ユーティリティ (Git Bash など) をインストールしています。

詳しいガイダンスが必要な場合は、Linux、macOS、または Unix での SSH 接続の場合 または Windows で SSH 接続をセットアップする手順 の手順に従ってください。

トピック

- ステップ 1: パブリックキーを IAM ユーザーに関連付ける
- ステップ 2: SSH 設定に CodeCommit を追加する
- 次のステップ

ステップ 1: パブリックキーを IAM ユーザーに関連付ける

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。
2. IAM コンソールのナビゲーションペインで [Users] (ユーザー) を選択し、ユーザーのリストから自分の IAM ユーザーを選択します。
3. [Security Credentials] タブで、[Upload SSH public key] を選択します。
4. SSH パブリックキーの内容をフィールドに貼り付け、[Upload SSH Key] を選択します。

> **Tip**
>
> パブリックとプライベートのキーペアは、OpenSSH 形式の SSH-2 RSA でなければならず、2048 ビットを含む必要があります。キーはこれに似たものになります。
>
> ```
> ssh-rsa EXAMPLE-
> AfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJB
> gNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb2
> 5zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhc
> NMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAw
> DgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAS=EXAMPLE user-
> name@ip-192-0-2-137
> ```
>
> IAM では、OpenSSH 形式のパブリックキーのみを使用できます。別の形式のパブリックキーを指定した場合は、キーの形式が無効であることを示すエラーメッセージが表示されます。

5. SSH キー ID (たとえば、*APKAEIBAERJR2EXAMPLE*) をコピーし、コンソールを閉じます。

ステップ 2: SSH 設定に CodeCommit を追加する

1. ターミナル (Linux、macOS、または Unix) または bash エミュレーター (Windows) で、SSH 設定ファイルを編集するには、cat>> ~/.ssh/config と入力してください。

```
Host git-codecommit.*.amazonaws.com
User Your-SSH-Key-ID, such as APKAEIBAERJR2EXAMPLE
IdentityFile Your-Private-Key-File, such as ~/.ssh/codecommit_rsa or ~/.ssh/id_rsa
```

> **Tip**
>
> 複数の SSH 設定がある場合は、コンテンツの前後に空白行を含めてください。`Ctrl` と `d` キーを同時に押してファイルを保存します。

2. 以下のコマンドを実行して、SSH 設定をテストします。

```
ssh git-codecommit.us-east-2.amazonaws.com
```

プロンプトが表示されたら、SSH キーファイルのパスフレーズを入力します。すべてが正しく設定されている場合は、次の成功メッセージが表示されます。

```
You have successfully authenticated over SSH. You can use Git to interact with
 CodeCommit.
```

次のステップ

前提条件を完了しました。CodeCommit の開始方法 のステップに従って、CodeCommit の使用を開始してください。

既存のリポジトリに接続するには、リポジトリへの接続 の手順に従ってください。このリポジトリを作成するには、リポジトリの作成 の手順に従ってください。

Linux、macOS、または Unix 上の AWS CodeCommit リポジトリへの SSH 接続のセットアップ手順

CodeCommit に初めて接続する前に、最初にいくつかの設定手順を完了する必要があります。コンピュータと AWS プロファイルを設定したら、CodeCommit リポジトリに接続し、そのリポジトリをコンピュータにクローンできます (ローカルリポジトリの作成とも呼ばれます) 。Git を初めて利用する場合は、 でも情報を確認できますGit の詳細情報

トピック

- ステップ 1: CodeCommit の初期設定
- ステップ 2: Git をインストールする
- ステップ 3: Linux、macOS、または Unix で認証情報を設定する
- ステップ 4: CodeCommit コンソールに接続し、リポジトリのクローンを作成する
- 次のステップ

ステップ 1: CodeCommit の初期設定

アマゾン ウェブ サービスアカウントを設定し、IAM ユーザーを作成して、CodeCommit へのアクセスを設定するには、以下の手順に従います。

IAM ユーザーを作成および設定して CodeCommit にアクセスするには

1. アマゾン ウェブ サービスアカウントを作成するには、http://aws.amazon.com にアクセスし、[Sign Up] (サインアップ) を選択します。
2. IAM ユーザーを作成するか、アマゾン ウェブ サービスアカウントに関連付けられた既存のユーザーを使用します。アクセスキー ID およびシークレットアクセスキーがその IAM ユーザーに関連付けられていることを確認します。詳細については、アマゾン ウェブ サービスアカウントの IAM ユーザーの作成を参照してください。

> **Note**
>
> CodeCommit には が必要です AWS Key Management Service。既存の IAM ユーザーを使用している場合は、CodeCommit に必要な AWS KMS アクションを明示的に拒否す

るポリシーがユーザーにアタッチされていないことを確認してください。詳細については、「AWS KMS および 暗号化」を参照してください。

3. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。
4. IAM コンソールのナビゲーションペインで、[Users] (ユーザー) を選択し、続いて、CodeCommit へアクセスするために設定する IAM ユーザーを選択します。
5. [Permissions (アクセス許可)] タブで、[Add Permissions (アクセス許可の追加)] を選択します。
6. [Grant permissions (アクセス許可の付与)] で、[Attach existing policies directly (既存のポリシーを直接アタッチする)] を選択します。
7. ポリシーの一覧から、[AWSCodeCommitPowerUser] または CodeCommit アクセスの別の管理ポリシーを選択します。詳細については、「AWS CodeCommit の マネージドポリシー」を参照してください。

   アタッチするポリシーを選択したら、[Next: Review] (次へ: 確認) を選択して、IAM ユーザーにアタッチするポリシーのリストを表示します。リストが正しい場合は、[Add permissions (アクセス許可の追加)] を選択します。

   CodeCommit 管理ポリシーや、その他のグループおよびユーザーを含むリポジトリへのアクセス共有の詳細については、リポジトリの共有 および AWS CodeCommitの認証とアクセスコントロール を参照してください。

> **Note**
>
> CodeCommit で AWS CLI コマンドを使用する場合は、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

ステップ 2: Git をインストールする

CodeCommit リポジトリのファイル、コミット、およびその他の情報を使用するには、ローカルマシンに Git をインストールする必要があります。CodeCommit は Git バージョン 1.7.9 以降をサポートしています。Git バージョン 2.28 は、初期コミットのブランチ名の設定をサポートしています。最新バージョンの Git を使用することをお勧めします。

Git をインストールするには、Git のダウンロードなどのウェブサイトをお勧めします。

> **Note**
>
> Git は、定期的に更新されている、発展中のプラットフォームです。機能の変更により、CodeCommit での動作が影響を受ける場合があります。特定のバージョンの Git と CodeCommit で問題が発生した場合は、 の情報を確認してください[トラブルシューティング]

ステップ 3: Linux、macOS、または Unix で認証情報を設定する

SSH および Linux、macOS、または Unix: Git と CodeCommit 用にパブリックキーとプライベートキーをセットアップする

Git および CodeCommit 用のパブリックキーとプライベートキーをセットアップするには

1. ローカルマシンのターミナルから ssh-keygen コマンドを実行し、手順に従って、プロフィールの .ssh ディレクトリにファイルを保存します。

   > **Note**
   >
   > 必ず、キーファイルの保存場所および使用するファイルの命名パターンをシステム管理者に確認してください。

   例:

```
$ ssh-keygen

Generating public/private rsa key pair.
Enter file in which to save the key (/home/user-name/.ssh/id_rsa): Type /home/
your-user-name/.ssh/ and a file name here, for example /home/your-user-name/.ssh/
codecommit_rsa

Enter passphrase (empty for no passphrase): <Type a passphrase, and then press
 Enter>
Enter same passphrase again: <Type the passphrase again, and then press Enter>

Your identification has been saved in /home/user-name/.ssh/codecommit_rsa.
Your public key has been saved in /home/user-name/.ssh/codecommit_rsa.pub.
The key fingerprint is:
45:63:d5:99:0e:99:73:50:5e:d4:b3:2d:86:4a:2c:14 user-name@client-name
```

```
The key's randomart image is:
+--[ RSA 2048]----+
|        E.+.o*.++|
|        .o .=.=o.|
|       . ..  *. +|
|        ..o . +..|
|        So . . . |
|          .      |
|                 |
|                 |
|                 |
+-----------------+
```

これにより、以下が生成されます。

- *codecommit_rsa* ファイル (プライベートキーファイル)
- *codecommit_rsa*.pub ファイル (パブリックキーファイル)

> **Tip**
>
> デフォルトで、ssh-keygen は 2048 ビットキーを生成します。-t および-b パラメータを使用して、キーのタイプと長さを指定することができます。RSA 形式の 4096 ビットキーが必要な場合は、次のパラメータを持つコマンドを実行することで、これを指定します。
>
> ```
> ssh-keygen -t rsa -b 4096
> ```
>
> SSH キーに必要な形式と長さの詳細については、CodeCommit での IAM の使用を参照してください。

2. 次のコマンドを実行して、パブリックキーファイル (*codecommit_rsa*.pub) の値を表示します。

```
cat ~/.ssh/codecommit_rsa.pub
```

この値をコピーします。それは次のようになります。

```
ssh-rsa EXAMPLE-AfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJB
gNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb2
```

```
5zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhc
NMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAw
DgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAS=EXAMPLE user-name@ip-192-0-2-137
```

3. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。

> **Note**
>
> CodeCommit の認証情報を [My Security Credentials] (セキュリティ認証情報) で直接表示および管理できます。詳細については、「認証情報の表示と管理」を参照してください。

4. IAM コンソールのナビゲーションペインで [Users] (ユーザー) を選択し、ユーザーのリストから自分の IAM ユーザーを選択します。
5. ユーザーの詳細ページで、[Security Credentials] タブを選択し、次に Upload SSH public key を選択します。
6. SSH パブリックキーの内容をフィールドに貼り付け、[Upload SSH public key] を選択します。
7. SSH Key ID の情報を (*APKAEIBAERJR2EXAMPLE* など) をコピーまたは保存します。

> **Note**
>
> 複数の SSH キー ID をアップロードしている場合、キーは、アップロードの日付ではなく、キー ID のアルファベット順にリストされます。正しいアップロードの日付に関連付けられたキー ID をコピーしていることを確認します。

8. ローカルマシンで、テキストエディタを使用して ~/.ssh ディレクトリに設定ファイルを作成し、そのファイルに次の行を追加します。ここで、*User* は、前の手順で作成した SSH キー ID を表します。

```
Host git-codecommit.*.amazonaws.com
  User APKAEIBAERJR2EXAMPLE
  IdentityFile ~/.ssh/codecommit_rsa
```

> **Note**
>
> プライベートキーファイルに *codecommit_rsa* 以外の名前を付けた場合は、ここでその名前を必ず使用してください。
> 複数のアマゾン ウェブ サービスアカウントでリポジトリへの SSH アクセスを設定できます。詳細については、 への SSH 接続のトラブルシューティング AWS CodeCommit を参照してください。

このファイル名を `config` として保存します。

9. ターミナルから、次のコマンドを実行して設定ファイルのアクセス権限を変更します。

```
chmod 600 config
```

10. 以下のコマンドを実行して、SSH 設定をテストします。

```
ssh git-codecommit.us-east-2.amazonaws.com
```

`git-codecommit.us-east-2.amazonaws.com` が既知の Hosts ファイルにまだ含まれていないため、接続の確認を求められます。この確認の一部として、CodeCommit サーバーのフィンガープリントが表示されます (MD5 の場合は `a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e`、SHA256 の場合は `3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ`)。

> **Note**
>
> CodeCommit サーバーのフィンガープリントは、すべての に対して一意です AWS リージョン。のサーバーフィンガープリントを表示するには AWS リージョン、「」を参照してくださいCodeCommit のサーバーフィンガープリント。

接続の確認後、そのサーバーが既知の Hosts ファイルに追加されたことの確認と接続の成功メッセージが表示されます。成功メッセージが表示されない場合は、CodeCommit にアクセスするために設定した IAM ユーザーの ~/.ssh ディレクトリに `config` ファイルが保存されていること、適切なプライベートキーファイルが指定されていることを確認します。

問題のトラブルシューティングに役立つ情報については、ssh パラメータを指定して -v コマンドを実行してください。例:

```
ssh -v git-codecommit.us-east-2.amazonaws.com
```

接続の問題のトラブルシューティングに役立つ情報については、への SSH 接続のトラブルシューティング AWS CodeCommit を参照してください。

ステップ 4: CodeCommit コンソールに接続し、リポジトリのクローンを作成する

管理者から CodeCommit リポジトリの名前と接続の詳細をすでに受け取っている場合は、このステップをスキップしてリポジトリを直接複製できます。

CodeCommit リポジトリに接続するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタで、リポジトリが作成された AWS リージョン を選択します。リポジトリは に固有です AWS リージョン。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。
3. 接続するリポジトリをリストから見つけて選択します。[クローン URL] を選択してから、リポジトリのクローン作成やリポジトリへの接続時に使用するプロトコルを選択します。これにより、クローン URL が複製されます。
   - IAM ユーザー、または AWS CLIに含まれている認証情報ヘルパーで Git 認証情報を使用している場合は、HTTPS URL をコピーします。
   - ローカルコンピュータで git-remote-codecommit コマンドを使用している場合は、HTTPS (GRC) URL をコピーします。
   - IAM ユーザーで SSH パブリック/プライベートキーペアを使用している場合は、SSH URL をコピーします。

> **Note**
>
> リポジトリのリストの代わりにようこそページが表示された場合は、サインイン AWS リージョン している に AWS アカウントに関連付けられたリポジトリはありません。リポジトリを作成するには、「the section called “リポジトリの作成”」を参照するか、「Git と CodeCommit の開始方法」チュートリアルのステップに従います。

4. ターミナルを開きます。/tmp ディレクトリから、レポジトリのクローンを作成するためにコピーした SSH URL を使用して git clone コマンドを実行します。For example, to clone a repository named *MyDemoRepo* to a local repo named *my-demo-repo* in the US East (Ohio) Region:

```
git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
```

> **Note**
>
> 正常に接続されたことを確認できましたが、コマンドを実行してもクローンを作成できない場合には、設定ファイルへのアクセスに必要な権限が付与されていないか、他の設定が設定ファイルと競合している可能性があります。もう一度接続をお試しください。今度は、コマンドに SSH キー ID を含めます。例:
>
> ```
> git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
> ```
>
> 詳細については、「アクセスエラー: パブリックキーが IAM に正常にアップロードされたが、Linux、macOS、または Unix システムでは接続が失敗する。」を参照してください。

リポジトリへの接続方法の詳細については、 を参照してくださいCodeCommit リポジトリのクローンを作成してリポジトリに接続する

次のステップ

前提条件を完了しました。CodeCommit の開始方法 のステップに従って、CodeCommit の使用を開始してください。

Windows 上の AWS CodeCommit リポジトリへの SSH 接続のセットアップ手順

AWS CodeCommit に初めて接続する前に、いくつかの初期設定手順を完了する必要があります。コンピュータと AWS プロファイルを設定したら、CodeCommit リポジトリに接続し、そのリポジトリをコンピュータにクローンできます (ローカルリポジトリの作成とも呼ばれます) 。Git を初めて利用する場合は、 でも情報を確認できますGit の詳細情報

トピック

- ステップ 1: CodeCommit の初期設定
- ステップ 2: Git をインストールする
- ステップ 3: Git および CodeCommit 用のパブリックキーとプライベートキーをセットアップする
- ステップ 4: CodeCommit コンソールに接続し、リポジトリのクローンを作成する
- 次のステップ

ステップ 1: CodeCommit の初期設定

アマゾン ウェブ サービスアカウントを設定し、IAM ユーザーを作成して、CodeCommit へのアクセスを設定するには、以下の手順に従います。

IAM ユーザーを作成および設定して CodeCommit にアクセスするには

1. アマゾン ウェブ サービスアカウントを作成するには、http://aws.amazon.com にアクセスし、[Sign Up] (サインアップ) を選択します。
2. IAM ユーザーを作成するか、アマゾン ウェブ サービスアカウントに関連付けられた既存のユーザーを使用します。アクセスキー ID およびシークレットアクセスキーがその IAM ユーザーに関連付けられていることを確認します。詳細については、アマゾン ウェブ サービスアカウントの IAM ユーザーの作成を参照してください。

> **Note**
>
> CodeCommit には が必要です AWS Key Management Service。既存の IAM ユーザーを使用している場合は、CodeCommit に必要な AWS KMS アクションを明示的に拒否するポリシーがユーザーにアタッチされていないことを確認してください。詳細については、「AWS KMS および 暗号化」を参照してください。

3. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。
4. IAM コンソールのナビゲーションペインで、[Users] (ユーザー) を選択し、続いて、CodeCommit へアクセスするために設定する IAM ユーザーを選択します。
5. [Permissions (アクセス許可)] タブで、[Add Permissions (アクセス許可の追加)] を選択します。
6. [Grant permissions (アクセス許可の付与)] で、[Attach existing policies directly (既存のポリシーを直接アタッチする)] を選択します。
7. ポリシーの一覧から、[AWSCodeCommitPowerUser] または CodeCommit アクセスの別の管理ポリシーを選択します。詳細については、「AWS CodeCommit の マネージドポリシー」を参照してください。

   アタッチするポリシーを選択したら、[Next: Review] (次へ: 確認) を選択して、IAM ユーザーにアタッチするポリシーのリストを表示します。リストが正しい場合は、[Add permissions (アクセス許可の追加)] を選択します。

   CodeCommit 管理ポリシーや、その他のグループおよびユーザーを含むリポジトリへのアクセス共有の詳細については、リポジトリの共有 および AWS CodeCommitの認証とアクセスコントロール を参照してください。

> **Note**
>
> CodeCommit で AWS CLI コマンドを使用する場合は、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

ステップ 2: Git をインストールする

CodeCommit リポジトリのファイル、コミット、およびその他の情報を使用するには、ローカルマシンに Git をインストールする必要があります。CodeCommit は Git バージョン 1.7.9 以降をサポー

トしています。Git バージョン 2.28 は、初期コミットのブランチ名の設定をサポートしています。最新バージョンの Git を使用することをお勧めします。

Git をインストールするには、Git のダウンロードなどのウェブサイトをお勧めします。

> **Note**
>
> Git は、定期的に更新されている、発展中のプラットフォームです。機能の変更により、CodeCommit での動作が影響を受ける場合があります。特定のバージョンの Git と CodeCommit で問題が発生した場合は、 の情報を確認してくださいトラブルシューティング

インストールした Git のバージョンに Bash エミュレーター (Git Bash など) が含まれていない場合はインストールします。SSH 接続を設定するときは、Windows コマンドラインの代わりにこのエミュレーターを使用します。

ステップ 3: Git および CodeCommit 用のパブリックキーとプライベートキーをセットアップする

Windows で Git および CodeCommit 用のパブリックキーとプライベートキーをセットアップするには

1. Bash エミュレーターを開きます。

   > **Note**
   >
   > 管理者権限でエミュレーターを実行する必要があります。

   エミュレーターから ssh-keygen コマンドを実行し、指示に従ってファイルをプロファイルの .ssh ディレクトリに保存します。

   例:

   ```
   $ ssh-keygen

   Generating public/private rsa key pair.
   Enter file in which to save the key (/drive/Users/user-name/.ssh/id_rsa): Type a
    file name here, for example /c/Users/user-name/.ssh/codecommit_rsa
   ```

```
Enter passphrase (empty for no passphrase): <Type a passphrase, and then press
 Enter>
Enter same passphrase again: <Type the passphrase again, and then press Enter>

Your identification has been saved in drive/Users/user-name/.ssh/codecommit_rsa.
Your public key has been saved in drive/Users/user-name/.ssh/codecommit_rsa.pub.
The key fingerprint is:
45:63:d5:99:0e:99:73:50:5e:d4:b3:2d:86:4a:2c:14 user-name@client-name
The key's randomart image is:
+--[ RSA 2048]----+
|        E.+.o*.++|
|        .o .=.=o.|
|       . ..  *. +|
|        ..o . +..|
|        So . . . |
|          .      |
|                 |
|                 |
|                 |
+-----------------+
```

これにより、以下が生成されます。

- *codecommit_rsa* ファイル (プライベートキーファイル)
- *codecommit_rsa*.pub ファイル (パブリックキーファイル)

> **Tip**
>
> デフォルトで、ssh-keygen は 2048 ビットキーを生成します。-t および-b パラメータを使用して、キーのタイプと長さを指定することができます。RSA 形式の 4096 ビットキーが必要な場合は、次のパラメータを持つコマンドを実行することで、これを指定します。
>
> ```
> ssh-keygen -t rsa -b 4096
> ```
>
> SSH キーに必要な形式と長さの詳細については、CodeCommit での IAM の使用を参照してください。

2. 以下のコマンドを実行して、パブリックキーファイル (*codecommit_rsa*.pub) の値を表示します。

```
cd .ssh
notepad codecommit_rsa.pub
```

ファイルの内容をコピーし、保存せずにメモ帳を閉じます。ファイルの内容は以下のようになります。

```
ssh-rsa EXAMPLE-AfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJB
gNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb2
5zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhc
NMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAw
DgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAS=EXAMPLE user-name@computer-name
```

3. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。

> **Note**
>
> CodeCommit の認証情報を [My Security Credentials] (セキュリティ認証情報) で直接表示および管理できます。詳細については、「認証情報の表示と管理」を参照してください。

4. IAM コンソールのナビゲーションペインで [Users] (ユーザー) を選択し、ユーザーのリストから自分の IAM ユーザーを選択します。
5. ユーザーの詳細ページで、[Security Credentials] タブを選択し、次に Upload SSH public key を選択します。
6. SSH パブリックキーの内容をフィールドに貼り付け、[Upload SSH public key] を選択します。
7. SSH Key ID の情報を (*APKAEIBAERJR2EXAMPLE* など) をコピーまたは保存します。

> **Note**
> 複数の SSH キー ID をアップロードしている場合、キーは、アップロードの日付ではなく、キー ID のアルファベット順にリストされます。正しいアップロードの日付に関連付けられたキー ID をコピーしていることを確認します。

8. Bash エミュレーターで、以下のコマンドを実行して ~/.ssh ディレクトリに設定ファイルを作成するか、このファイルがすでにあれば編集します。

```
notepad ~/.ssh/config
```

9. ファイルに以下の行を追加します。ここで、*User* の値は先ほどコピーした SSH キー ID であり、*IdentityFile* の値はプライベートキーファイルのパスと名前です。

```
Host git-codecommit.*.amazonaws.com
  User APKAEIBAERJR2EXAMPLE
  IdentityFile ~/.ssh/codecommit_rsa
```

> **Note**
> プライベートキーファイルに *codecommit_rsa* 以外の名前を付けた場合は、ここでその名前を必ず使用してください。
> 複数のアマゾン ウェブ サービスアカウントでリポジトリへの SSH アクセスを設定できます。詳細については、への SSH 接続のトラブルシューティング AWS CodeCommit を参照してください。

ファイルを config.txt ではなく config として保存し、メモ帳を閉じます。

> **Important**
> このファイル名はファイル拡張子なしの `config` であることが必要です。そうでない場合、SSH 接続が失敗します。

10. 以下のコマンドを実行して、SSH 設定をテストします。

```
ssh git-codecommit.us-east-2.amazonaws.com
```

`git-codecommit.us-east-2.amazonaws.com` が既知の Hosts ファイルにまだ含まれていないため、接続の確認を求められます。この確認の一部として、CodeCommit サーバーのフィンガープリントが表示されます (MD5 の場合は `a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e`、SHA256 の場合は `3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ`)。

> **Note**
>
> CodeCommit サーバーのフィンガープリントは、それぞれに一意です AWS リージョン。のサーバーフィンガープリントを表示するには AWS リージョン、「」を参照してください[CodeCommit のサーバーフィンガープリント](#)。

接続の確認後、そのサーバーが既知の Hosts ファイルに追加されたことの確認と接続の成功メッセージが表示されます。成功メッセージが表示されない場合は、CodeCommit へのアクセス用に設定した IAM ユーザーの ~/.ssh ディレクトリに`config`ファイルを保存したこと、`config`ファイルにファイル拡張子がないこと (たとえば、config.txt という名前が付けられていないこと) 、および正しいプライベートキーファイル (*`codecommit_rsa`*.pub ではなく *`codecommit_rsa`*) を指定したことを再度確認します。

問題のトラブルシューティングを行うには、`-v` パラメータを指定して `ssh` コマンドを実行してください。例:

```
ssh -v git-codecommit.us-east-2.amazonaws.com
```

接続の問題のトラブルシューティングに役立つ情報については、[への SSH 接続のトラブルシューティング AWS CodeCommit](#) を参照してください。

ステップ 4: CodeCommit コンソールに接続し、リポジトリのクローンを作成する

管理者から CodeCommit リポジトリの名前と接続の詳細をすでに受け取っている場合は、このステップをスキップしてリポジトリを直接複製できます。

CodeCommit リポジトリに接続するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタで、リポジトリが作成された AWS リージョン を選択します。リポジトリは に固有です AWS リージョン。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。
3. 接続するリポジトリをリストから見つけて選択します。[クローン URL] を選択してから、リポジトリのクローン作成やリポジトリへの接続時に使用するプロトコルを選択します。これにより、クローン URL が複製されます。

   - IAM ユーザー、または AWS CLIに含まれている認証情報ヘルパーで Git 認証情報を使用している場合は、HTTPS URL をコピーします。
   - ローカルコンピュータで git-remote-codecommit コマンドを使用している場合は、HTTPS (GRC) URL をコピーします。
   - IAM ユーザーで SSH パブリック/プライベートキーペアを使用している場合は、SSH URL をコピーします。

   > **Note**
   > リポジトリのリストの代わりにようこそページが表示された場合は、サインイン AWS リージョン している に AWS アカウントに関連付けられたリポジトリはありません。リポジトリを作成するには、「the section called “リポジトリの作成”」を参照するか、「Git と CodeCommit の開始方法」チュートリアルのステップに従います。

4. Bash エミュレーターで、リポジトリのクローンを作成するためにコピーした SSH URL を使用して git clone コマンドを実行します。このコマンドは、コマンドを実行したディレクトリのサブディレクトリにローカルリポジトリを作成します。For example, to clone a repository named *MyDemoRepo* to a local repo named *my-demo-repo* in the US East (Ohio) Region:

   ```
   git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
   ```

   あるいは、コマンドプロンプトを開き、IAM にアップロードしたパブリックキーの URL と SSH キー ID を使用して、git clone コマンドを実行します。ローカルリポジトリは、そのコマンドを実行したディレクトリのサブディレクトリに作成されます。例えば、*MyDemoRepo* という名前

のリポジトリのクローンを *my-demo-repo* という名前のローカルリポジトリに作成するには、次のようにします。

```
git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo my-demo-repo
```

詳細については、「CodeCommit リポジトリのクローンを作成してリポジトリに接続する」および「コミットを作成する」を参照してください。

次のステップ

前提条件を完了しました。CodeCommit の開始方法 のステップに従って、CodeCommit の使用を開始してください。

AWS CLI 認証情報ヘルパーを使用して Linux、macOS、または Unix 上の AWS CodeCommit リポジトリへの HTTPS 接続を設定する手順

AWS CodeCommit に初めて接続する前に、初期設定手順を完了する必要があります。ほとんどのユーザーにとっては、これは Git 認証情報を使用した HTTPS ユーザーのセットアップ の手順に従って簡単に行うことができます。ただし、ルートアカウント、フェデレーテッドアクセス、または一時的な認証情報を使用して CodeCommit に接続する場合は、 AWS CLIに含まれている認証情報ヘルパーを使用できます。

> **Note**
>
> 認証情報ヘルパーは、フェデレーティッドアクセス、ID プロバイダー、または一時的な認証情報を使用して CodeCommit に接続するためにサポートされている方法ですが、git-remote-codecommit ユーティリティをインストールして使用する方法が推奨されます。詳細については、「git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」を参照してください。

トピック

- ステップ 1: CodeCommit の初期設定

- ステップ 2: Git をインストールする
- ステップ 3: 認証情報ヘルパーを設定する
- ステップ 4: CodeCommit コンソールに接続し、リポジトリのクローンを作成する
- 次のステップ

ステップ 1: CodeCommit の初期設定

アマゾン ウェブ サービスアカウントをセットアップし、IAM ユーザーを作成および設定し、 AWS CLIをインストールするには、次の手順を実行します。

IAM ユーザーを作成および設定して CodeCommit にアクセスするには

1. アマゾン ウェブ サービスアカウントを作成するには、http://aws.amazon.com にアクセスし、[Sign Up] (サインアップ) を選択します。
2. IAM ユーザーを作成するか、アマゾン ウェブ サービスアカウントに関連付けられた既存のユーザーを使用します。アクセスキー ID およびシークレットアクセスキーがその IAM ユーザーに関連付けられていることを確認します。詳細については、アマゾン ウェブ サービスアカウントの IAM ユーザーの作成を参照してください。

   > **Note**
   >
   > CodeCommit には が必要です AWS Key Management Service。既存の IAM ユーザーを使用している場合は、CodeCommit に必要な AWS KMS アクションを明示的に拒否するポリシーがユーザーにアタッチされていないことを確認してください。詳細については、「AWS KMS および 暗号化」を参照してください。

3. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。
4. IAM コンソールのナビゲーションペインで、[Users] (ユーザー) を選択し、続いて、CodeCommit へアクセスするために設定する IAM ユーザーを選択します。
5. [Permissions (アクセス許可)] タブで、[Add Permissions (アクセス許可の追加)] を選択します。
6. [Grant permissions (アクセス許可の付与)] で、[Attach existing policies directly (既存のポリシーを直接アタッチする)] を選択します。
7. ポリシーの一覧から、[AWSCodeCommitPowerUser] または CodeCommit アクセスの別の管理ポリシーを選択します。詳細については、「AWS CodeCommit の マネージドポリシー」を参照してください。

アタッチするポリシーを選択したら、[Next: Review] (次へ: 確認) を選択して、IAM ユーザーにアタッチするポリシーのリストを表示します。リストが正しい場合は、[Add permissions (アクセス許可の追加)] を選択します。

CodeCommit 管理ポリシーや、その他のグループおよびユーザーを含むリポジトリへのアクセス共有の詳細については、リポジトリの共有 および AWS CodeCommitの認証とアクセスコントロール を参照してください。

をインストールして設定するには AWS CLI

1. ローカルマシンで、 をダウンロードしてインストールします AWS CLI。これは、コマンドラインから CodeCommit とやり取りするための前提条件です。 AWS CLI バージョン 2 のインストールが推奨されます。これは の最新バージョン AWS CLI であり、すべての最新機能をサポートしています。これは、 でルートアカウント、フェデレーティッドアクセス、または一時的な認証情報の使用 AWS CLI をサポートする の唯一のバージョンですgit-remote-codecommit。

   詳細については、AWS 「 コマンドラインインターフェイスのセットアップ」を参照してください。

   > **Note**
   >
   > CodeCommit は、 AWS CLI バージョン 1.7.38 以降でのみ動作します。ベストプラクティスとして、 をインストールまたは最新バージョンにアップグレード AWS CLI します。インストール AWS CLI した のバージョンを確認するには、 aws --version コマンドを実行します。
   > の古いバージョン AWS CLI を最新バージョンにアップグレードするには、「 のインストール AWS Command Line Interface」を参照してください。

2. このコマンドを実行して、 の CodeCommit コマンド AWS CLI がインストールされていることを確認します。

   ```
   aws codecommit help
   ```

   このコマンドは、CodeCommit コマンドのリストを返します。

3. 次のように、 configure コマンドを使用して プロファイル AWS CLI で を設定します。

```
aws configure
```

プロンプトが表示されたら、CodeCommit で使用する IAM ユーザーの AWS アクセスキーと AWS シークレットアクセスキーを指定します。また、 など、リポジトリが存在する AWS リージョン を必ず指定してください`us-east-2`。デフォルトの出力形式の入力を求められたら、`json` を指定します。例えば、IAM ユーザーのプロファイルを設定する場合は、次のようにします。

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

で使用するプロファイルの作成と設定の詳細については AWS CLI、以下を参照してください。

- 名前付きプロファイル
- での IAM ロールの使用 AWS CLI
- Set コマンド
- 認証情報をローテーションして AWS CodeCommit リポジトリに接続する

リポジトリまたは別のリソースに接続するには AWS リージョン、デフォルトのリージョン名 AWS CLI で を再設定する必要があります。CodeCommit でサポートされるデフォルトのリージョン名は以下のとおりです。

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3

- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

CodeCommit と の詳細については AWS リージョン、「」を参照してくださいリージョンと Git 接続エンドポイント。IAM、アクセスキー、シークレットキーに関する詳細については、認証情報を取得する方法および IAM ユーザーのアクセスキーの管理を参照してください。 AWS CLI および プロファイルの詳細については、「名前付きプロファイル」を参照してください。

ステップ 2: Git をインストールする

CodeCommit リポジトリのファイル、コミット、およびその他の情報を使用するには、ローカルマシンに Git をインストールする必要があります。CodeCommit は Git バージョン 1.7.9 以降をサポートしています。Git バージョン 2.28 は、初期コミットのブランチ名の設定をサポートしています。最新バージョンの Git を使用することをお勧めします。

Git をインストールするには、Git のダウンロードなどのウェブサイトをお勧めします。

> **Note**
>
> Git は、定期的に更新されている、発展中のプラットフォームです。機能の変更により、CodeCommit での動作が影響を受ける場合があります。特定のバージョンの Git と CodeCommit で問題が発生した場合は、 の情報を確認してくださいトラブルシューティング

ステップ 3: 認証情報ヘルパーを設定する

1. ターミナルから、Git を使用して を実行しgit config、 AWS 認証情報プロファイルで Git 認証情報ヘルパーの使用を指定し、Git 認証情報ヘルパーがリポジトリにパスを送信できるようにします。

```
git config --global credential.helper '!aws codecommit credential-helper $@'
git config --global credential.UseHttpPath true
```

> **Tip**
>
> 認証情報ヘルパーは、デフォルトの AWS 認証情報プロファイルまたは Amazon EC2 インスタンスロールを使用します。CodeCommit で使用する AWS 認証情報プロファイルを作成した場合は、`CodeCommitProfile` など、使用するプロファイルを指定できます。
>
> ```
> git config --global credential.helper '!aws --profile CodeCommitProfile
> codecommit credential-helper $@'
> ```
>
> プロファイル名にスペースが含まれる場合は、名前を二重引用符 (") で囲みます。`--global` ではなく `--local` を使用して、グローバルではなくリポジトリごとにプロファイルを設定できます。

Git 認証情報ヘルパーは、次の値を `~/.gitconfig` に書き込みます。

```
[credential]
    helper = !aws --profile CodeCommitProfile codecommit credential-helper $@
    UseHttpPath = true
```

> **Important**
>
> CodeCommit の同じローカルマシンで別の IAM ユーザーを使用する場合は、 git config コマンドを再度実行し、別の AWS 認証情報プロファイルを指定する必要があります。

2. git config --global --edit を実行して、前の値が `~/.gitconfig` に書き込まれたことを確認します。成功した場合は (Git グローバルコンフィグレーションファイルにすでに存在する値に加えて) 以前の値が表示されます。終了するには、通常、**:q** と入力して [Enter] を押します。

   認証情報ヘルパーを設定した後で問題が発生する場合は、「トラブルシューティング」を参照してください。

> **Important**
>
> macOS を使用している場合は、以下の手順を実行して、認証情報ヘルパーが正しく設定されていることを確認してください。

3. macOS を使用している場合は、HTTPS を使用して CodeCommit リポジトリに接続します。HTTPS を使用して CodeCommit リポジトリに初めて接続すると、約 15 分後に後続のアクセスが失敗します。macOS のデフォルトの Git バージョンは、Keychain Access ユーティリティを使用して認証情報を保存します。セキュリティ対策のために、CodeCommit リポジトリへのアクセス用に生成されるパスワードは一時的なものであり、約 15 分後にキーチェーンに保存されている認証情報は機能しなくなります。期限切れの認証情報が使用されないようにするには、これらのいずれかが必要になります。

   - デフォルトでキーチェーンを使用しない Git バージョンをインストールします。
   - CodeCommit リポジトリの認証情報を提供しないように Keychain Access ユーティリティを設定します。

   1. Keychain Access ユーティリティを開きます。(Finder を使用して位置を指定できます)
   2. `git-codecommit.us-east-2.amazonaws.com` を検索します。行をハイライト表示し、コンテキストメニューを開くか、右クリックして、[Get Info] を選択します。
   3. [Access Control] タブを選択します。
   4. [Confirm before allowing access] で、[`git-credential-osxkeychain`] を選択し、マイナス記号を選択してリストから削除します。

> **Note**
>
> リストから `git-credential-osxkeychain` を削除した後、Git コマンドを実行するたびにポップアップメッセージが表示されます。[Deny] を選択して続行します。ポップアップが不適正な場合は、次のようないくつかのその他のオプションがあります。
>
> - HTTPS ではなく SSH を使用して CodeCommit に接続します。詳細については、「Linux、macOS、または Unix での SSH 接続の場合」を参照してください。
> - Keychain Access ユーティリティで、`git-codecommit.us-east-2.amazonaws.com` の [Access Control] (アクセスコントロール) タブから、[Allow all applications to access this item (access to this item is not restricted)] (すべてのアプリケーションがこのアイテムにアクセスすることを許可 (このアイテムへのアクセスは制限されない)) オプションを選択します。これにより、ポップアップは表示されませんが、認証情報はやがて無効になり (平均で約 15 分後)、403 エラーメッセージが表示されます。この場合、キーチェーンアイテムを削除して機能を復元する必要があります。
> - 詳細については、「Git for macOS: 認証情報ヘルパーは正常に設定できましたが、リポジトリへのアクセスが拒否されます (403)」を参照してください。

ステップ 4: CodeCommit コンソールに接続し、リポジトリのクローンを作成する

管理者から CodeCommit リポジトリの名前と接続の詳細をすでに受け取っている場合は、このステップをスキップしてリポジトリを直接複製できます。

CodeCommit リポジトリに接続するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタで、リポジトリが作成された AWS リージョン を選択します。リポジトリは に固有です AWS リージョン。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。

3. 接続するリポジトリをリストから見つけて選択します。[クローン URL] を選択してから、リポジトリのクローン作成やリポジトリへの接続時に使用するプロトコルを選択します。これにより、クローン URL が複製されます。

   - IAM ユーザー、または AWS CLIに含まれている認証情報ヘルパーで Git 認証情報を使用している場合は、HTTPS URL をコピーします。
   - ローカルコンピュータで git-remote-codecommit コマンドを使用している場合は、HTTPS (GRC) URL をコピーします。
   - IAM ユーザーで SSH パブリック/プライベートキーペアを使用している場合は、SSH URL をコピーします。

   > **Note**
   >
   > リポジトリのリストの代わりにようこそページが表示された場合は、サインイン AWS リージョン している に AWS アカウントに関連付けられたリポジトリはありません。リポジトリを作成するには、「the section called “リポジトリの作成”」を参照するか、「Git と CodeCommit の開始方法」チュートリアルのステップに従います。

4. ターミナルを開き、コピーした HTTPS URL で git clone コマンドを実行します。For example, to clone a repository named *MyDemoRepo* to a local repo named *my-demo-repo* in the US East (Ohio) Region:

```
git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-
demo-repo
```

次のステップ

前提条件を完了しました。CodeCommit の開始方法 のステップに従って、CodeCommit の使用を開始してください。

AWS CLI 認証情報ヘルパーを使用した Windows 上の AWS CodeCommit リポジトリへの HTTPS 接続のセットアップ手順

AWS CodeCommit に初めて接続する前に、初期設定手順を完了する必要があります。ほとんどのユーザーにとっては、これは Git 認証情報を使用した HTTPS ユーザーのセットアップ の手順に従っ

て簡単に行うことができます。ただし、ルートアカウント、フェデレーテッドアクセス、または一時的な認証情報を使用して CodeCommit に接続する場合は、 AWS CLIに含まれている認証情報ヘルパーを使用できます。

> **Note**
>
> 認証情報ヘルパーは、フェデレーティッドアクセス、ID プロバイダー、または一時的な認証情報を使用して CodeCommit に接続するためにサポートされている方法ですが、git-remote-codecommit ユーティリティをインストールして使用する方法が推奨されます。詳細については、「git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」を参照してください。

このトピックでは、 のインストール AWS CLI、コンピュータと AWS プロファイルの設定、CodeCommit リポジトリへの接続、ローカルリポジトリの作成とも呼ばれるコンピュータへのリポジトリのクローン作成の手順について説明します。Git を初めて利用する場合は、 でも情報を確認できますGit の詳細情報

トピック

- ステップ 1: CodeCommit の初期設定
- ステップ 2: Git をインストールする
- ステップ 3: 認証情報ヘルパーを設定する
- ステップ 4: CodeCommit コンソールに接続し、リポジトリのクローンを作成する
- 次のステップ

ステップ 1: CodeCommit の初期設定

アマゾン ウェブ サービスアカウントをセットアップし、IAM ユーザーを作成および設定し、 AWS CLIをインストールするには、次の手順を実行します。 AWS CLI には、CodeCommit リポジトリへの HTTPS 接続を設定する認証情報ヘルパーが組み込まれています。

IAM ユーザーを作成および設定して CodeCommit にアクセスするには

1. アマゾン ウェブ サービスアカウントを作成するには、http://aws.amazon.com にアクセスし、[Sign Up] (サインアップ) を選択します。
2. IAM ユーザーを作成するか、アマゾン ウェブ サービスアカウントに関連付けられた既存のユーザーを使用します。アクセスキー ID およびシークレットアクセスキーがその IAM ユーザーに関

連付けられていることを確認します。詳細については、アマゾン ウェブ サービスアカウントの IAM ユーザーの作成を参照してください。

> **Note**
>
> CodeCommit には が必要です AWS Key Management Service。既存の IAM ユーザーを使用している場合は、CodeCommit に必要な AWS KMS アクションを明示的に拒否するポリシーがユーザーにアタッチされていないことを確認してください。詳細については、「AWS KMS および 暗号化」を参照してください。

3. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。
4. IAM コンソールのナビゲーションペインで、[Users] (ユーザー) を選択し、続いて、CodeCommit へアクセスするために設定する IAM ユーザーを選択します。
5. [Permissions (アクセス許可)] タブで、[Add Permissions (アクセス許可の追加)] を選択します。
6. [Grant permissions (アクセス許可の付与)] で、[Attach existing policies directly (既存のポリシーを直接アタッチする)] を選択します。
7. ポリシーの一覧から、[AWSCodeCommitPowerUser] または CodeCommit アクセスの別の管理ポリシーを選択します。詳細については、「AWS CodeCommit の マネージドポリシー」を参照してください。

   アタッチするポリシーを選択したら、[Next: Review] (次へ: 確認) を選択して、IAM ユーザーにアタッチするポリシーのリストを表示します。リストが正しい場合は、[Add permissions (アクセス許可の追加)] を選択します。

   CodeCommit 管理ポリシーや、その他のグループおよびユーザーを含むリポジトリへのアクセス共有の詳細については、リポジトリの共有 および AWS CodeCommitの認証とアクセスコントロール を参照してください。

をインストールして設定するには AWS CLI

1. ローカルマシンで、 をダウンロードしてインストールします AWS CLI。これは、コマンドラインから CodeCommit とやり取りするための前提条件です。 AWS CLI バージョン 2 のインストールが推奨されます。これは の最新バージョン AWS CLI であり、すべての最新機能をサポートしています。これは、 でルートアカウント、フェデレーティッドアクセス、または一時的な認証情報の使用 AWS CLI をサポートする の唯一のバージョンですgit-remote-codecommit。

詳細については、AWS 「コマンドラインインターフェイスのセットアップ」を参照してください。

> **Note**
>
> CodeCommit は、 AWS CLI バージョン 1.7.38 以降でのみ動作します。ベストプラクティスとして、 をインストールまたは最新バージョンにアップグレード AWS CLI します。インストール AWS CLI した のバージョンを確認するには、 aws --version コマンドを実行します。
> の古いバージョン AWS CLI を最新バージョンにアップグレードするには、「 のインストール AWS Command Line Interface」を参照してください。

2. このコマンドを実行して、 の CodeCommit コマンド AWS CLI がインストールされていることを確認します。

```
aws codecommit help
```

このコマンドは、CodeCommit コマンドのリストを返します。

3. 次のように、 configure コマンドを使用して プロファイル AWS CLI で を設定します。

```
aws configure
```

プロンプトが表示されたら、CodeCommit で使用する IAM ユーザーの AWS アクセスキーと AWS シークレットアクセスキーを指定します。また、 など、リポジトリが存在する AWS リージョン を必ず指定してくださいus-east-2。デフォルトの出力形式の入力を求められたら、json を指定します。例えば、IAM ユーザーのプロファイルを設定する場合は、次のようにします。

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

で使用するプロファイルの作成と設定の詳細については AWS CLI、以下を参照してください。

- 名前付きプロファイル
- での IAM ロールの使用 AWS CLI
- Set コマンド
- 認証情報をローテーションして AWS CodeCommit リポジトリに接続する

リポジトリまたは別のリソースに接続するには AWS リージョン、デフォルトのリージョン名 AWS CLI で を再設定する必要があります。CodeCommit でサポートされるデフォルトのリージョン名は以下のとおりです。

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1

- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

CodeCommit と の詳細については AWS リージョン、「」を参照してくださいリージョンと Git 接続エンドポイント。IAM、アクセスキー、シークレットキーに関する詳細については、認証情報を取得する方法および IAM ユーザーのアクセスキーの管理を参照してください。 AWS CLI および プロファイルの詳細については、「名前付きプロファイル」を参照してください。

ステップ 2: Git をインストールする

CodeCommit リポジトリのファイル、コミット、およびその他の情報を使用するには、ローカルマシンに Git をインストールする必要があります。CodeCommit は Git バージョン 1.7.9 以降をサポートしています。Git バージョン 2.28 は、初期コミットのブランチ名の設定をサポートしています。最新バージョンの Git を使用することをお勧めします。

Git をインストールするには、Git for Windows などのウェブサイトをお勧めします。このリンクを使用して Git をインストールする場合、以下を除くすべてのインストールのデフォルト設定を使用できます。

- [Adjusting your PATH environment (PATH 環境の調整)] ステップでプロンプトが表示されたら、[Use Git from the Windows Command Prompt (Windows コマンドプロンプトから Git を使用する)] オプションを選択します。
- （オプション) CodeCommit の Git 認証情報を設定する AWS CLI 代わりに、 に含まれる認証情報ヘルパーで HTTPS を使用する場合は、「追加のオプションの設定」ページで、Git 認証情報マネージャーを有効にするオプションがオフになっていることを確認してください。Git 認証情報マネージャーは、IAM ユーザーが Git 認証情報を設定する場合のみ、CodeCommit と互換性があります。詳細については、「Git 認証情報を使用した HTTPS ユーザーのセットアップ」および「Git for Windows: Git for Windows をインストールしましたが、リポジトリへのアクセスが拒否されます (403)」を参照してください。

> **Note**
>
> Git は、定期的に更新されている、発展中のプラットフォームです。機能の変更により、CodeCommit での動作が影響を受ける場合があります。特定のバージョンの Git と CodeCommit で問題が発生した場合は、 の情報を確認してください[トラブルシューティング]

ステップ 3: 認証情報ヘルパーを設定する

には、CodeCommit で使用できる Git 認証情報ヘルパー AWS CLI が含まれています。Git 認証情報ヘルパーには AWS 、IAM ユーザーの AWS アクセスキー ID と AWS シークレットアクセスキー (デフォルト AWS リージョン 名とデフォルト出力形式) のコピーを保存する認証情報プロファイルが必要です。Git 認証情報ヘルパーはこの情報を使用して CodeCommit で自動的に認証するため、Git を使用して CodeCommit とやり取りするたびにこの情報を入力する必要はありません。

1. コマンドプロンプトを開き、Git を使用して を実行しgit config、 AWS 認証情報プロファイルで Git 認証情報ヘルパーの使用を指定します。これにより、Git 認証情報ヘルパーはリポジトリにパスを送信できます。

   ```
   git config --global credential.helper "!aws codecommit credential-helper $@"
   git config --global credential.UseHttpPath true
   ```

   Git 認証情報ヘルパーは、.gitconfig ファイルに次の情報を書き込みます。

   ```
   [credential]
       helper = !aws codecommit credential-helper $@
       UseHttpPath = true
   ```

   > **Important**
   >
   > - Windows のコマンドラインではなく Bash エミュレーターを使用している場合は、二重引用符ではなく単一引用符を使用する必要があります。
   > - 認証情報ヘルパーは、デフォルトの AWS プロファイルまたは Amazon EC2 インスタンスロールを使用します。*CodeCommitProfile* など、使用する AWS 認証情報プロファイルを作成した場合は、次のようにコマンドを変更して代わりに使用できます。

```
git config --global credential.helper "!aws codecommit credential-helper
 --profile CodeCommitProfile $@"
```

これにより、.gitconfig ファイルに次のように書き込まれます。

```
[credential]
    helper = !aws codecommit credential-helper --profile=CodeCommitProfile
 $@
    UseHttpPath = true
```

- プロファイル名にスペースが含まれる場合は、このコマンドを実行した後、.gitconfig ファイルを編集して単一引用符 (') で囲みます。そうしないと、認証情報ヘルパーは動作しません。
- Git for Windows のインストールに Git 認証情報マネージャーユーティリティが含まれている場合、最初の数回の接続試行後に 認証情報マネージャーユーティリティに認証情報を提供するかどうかを確認する 403 エラーが表示されます。CodeCommit と互換性がないため、この問題を解決する最も確実な方法は、Git 認証情報マネージャーユーティリティのオプションを使用せずに Git for Windows をアンインストールしてから再インストールすることです。Git Credential Manager ユーティリティを維持する場合は、追加の設定ステップを実行して CodeCommit も使用する必要があります。これには、.gitconfig ファイルを手動で変更して CodeCommit に接続する AWS CodeCommit ときに の認証情報ヘルパーの使用を指定することが含まれます。認証情報マネージャーユーティリティから格納されている資格情報を削除します (このユーティリティはコントロールパネルにあります)。格納されている認証情報を削除した後、以下を .gitconfig ファイルに追加して保存し、新しいコマンドプロンプトウィンドウから再度接続してみてください。

```
[credential "https://git-codecommit.us-east-2.amazonaws.com"]
    helper = !aws codecommit credential-helper $@
    UseHttpPath = true

[credential "https://git-codecommit.us-east-1.amazonaws.com"]
    helper = !aws codecommit credential-helper $@
    UseHttpPath = true
```

git config または --system の代わりに --global を指定して、--local 設定を再構成する必要がある場合もあります。

- CodeCommit で同じローカルマシンで異なる IAM ユーザーを使用する場合は、 の代わりに を指定 git config --localしgit config --global、各 AWS 認証情報プロファイルの設定を実行する必要があります。

2. git config --global --edit を実行して、前の値がユーザープロファイルの .gitconfig file (デフォルトでは `%HOME%\.gitconfig` または `drive:\Users\UserName\.gitconfig`) に書き込まれていることを検証します。成功した場合は (Git グローバルコンフィグレーションファイルにすでに存在する値に加えて) 以前の値が表示されます。終了するには、通常、**:q** と入力して [Enter] を押します。

ステップ 4: CodeCommit コンソールに接続し、リポジトリのクローンを作成する

管理者から CodeCommit リポジトリの名前と接続の詳細をすでに受け取っている場合は、このステップをスキップしてリポジトリを直接複製できます。

CodeCommit リポジトリに接続するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタで、リポジトリが作成された AWS リージョン を選択します。リポジトリは に固有です AWS リージョン。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。
3. 接続するリポジトリをリストから見つけて選択します。[クローン URL] を選択してから、リポジトリのクローン作成やリポジトリへの接続時に使用するプロトコルを選択します。これにより、クローン URL が複製されます。

   - IAM ユーザー、または AWS CLIに含まれている認証情報ヘルパーで Git 認証情報を使用している場合は、HTTPS URL をコピーします。
   - ローカルコンピュータで git-remote-codecommit コマンドを使用している場合は、HTTPS (GRC) URL をコピーします。
   - IAM ユーザーで SSH パブリック/プライベートキーペアを使用している場合は、SSH URL をコピーします。

> **Note**
>
> リポジトリのリストの代わりにようこそページが表示された場合は、サインイン AWS リージョン している に AWS アカウントに関連付けられたリポジトリはありません。リポジトリを作成するには、「the section called “リポジトリの作成”」を参照するか、「Git と CodeCommit の開始方法」チュートリアルのステップに従います。

4. コマンドプロンプトを開き、コピーした HTTPS URL で git clone コマンドを実行します。ローカルリポジトリは、そのコマンドを実行したディレクトリのサブディレクトリに作成されます。For example, to clone a repository named *MyDemoRepo* to a local repo named *my-demo-repo* in the US East (Ohio) Region:

```
git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-
demo-repo
```

   Windows の一部のバージョンでは、ユーザー名とパスワードの入力を求めるポップアップメッセージが表示されることがあります。これは Windows 用の組み込みの認証情報管理システムですが、 の認証情報ヘルパーと互換性がありません AWS CodeCommit[Cancel] を選択します。

次のステップ

前提条件を完了しました。CodeCommit の開始方法 のステップに従って、CodeCommit の使用を開始してください。

入門

CodeCommit の使用を開始する最も簡単な方法は、CodeCommit の開始方法 のステップを実行することです。Git および CodeCommit を初めて利用する場合は、次の Git と CodeCommit の開始方法 のステップに従うことも検討してください。これにより、CodeCommit の操作に慣れるだけでなく、CodeCommit リポジトリとの通信時における Git の基本的な使用方法を理解できます。

また、CodePipeline および CodeCommit を使用したシンプルなパイプラインの演習のチュートリアルでは、継続的デリバリーパイプラインの一環として CodeCommit リポジトリを使用する方法を理解できます。

このセクションのチュートリアルは、次の前提条件とセットアップを完了したことを前提として書かれています。

- IAM ユーザーにアクセス許可を割り当てます。
- このチュートリアルで使用するローカルマシンの HTTPS および SSH 接続に必要な認証マネージメントのセットアップ
- リポジトリの作成を含むすべてのオペレーションでコマンドラインまたはターミナル AWS CLI を使用する場合は、 を設定します。

トピック

- の開始方法 AWS CodeCommit
- Git と の開始方法 AWS CodeCommit

の開始方法 AWS CodeCommit

このチュートリアルでは、いくつかの主要な CodeCommit 機能の使用方法を説明します。まず、リポジトリを作成し、リポジトリにいくつかの変更をコミットします。次に、ファイルを参照し、変更を表示します。他のユーザーのプルリクエストを作成して、自分のコードの変更を確認したり、コメントを追加したりすることもできます。

既存のコードを CodeCommit に移行する方法については、「への移行 AWS CodeCommit」を参照してください。

Git に慣れていない場合は、Git と CodeCommit の開始方法 を完了することも検討してください。これらのチュートリアルを完了後、自身のプロジェクトやチーム環境で CodeCommit を使用するには、十分な練習が必要です。

CodeCommit コンソールの折りたたみ可能なパネルを使用して、有益な情報を表示することができます。これは、情報アイコン

( )、

またはそのページの [Info] (情報) リンクから開くことができます。このパネルは、いつでも閉じることができます。

また、CodeCommit コンソールでは、リポジトリ、ビルドプロジェクト、デプロイアプリケーション、パイプラインなどのリソースをすばやく検索することもできます。[Go to resource (リソースに移動)] または / キーを押して、リソースの名前を入力します。一致するものはすべてリストに表示されます。検索では大文字と小文字が区別されません。リソースを表示する権限がある場合のみ表示されます。詳細については、「コンソールでのリソースの表示」を参照してください。

前提条件

開始する前に、以下の前提条件と設定手順を完了する必要があります。

- IAM ユーザーにアクセス許可を割り当てます。

- このチュートリアルでローカルマシンでの HTTP および SSH 接続に使用する認証情報管理の設定
- リポジトリの作成を含むすべてのオペレーションでコマンドラインまたはターミナル AWS CLI を使用する場合は、 を設定します。

トピック

- ステップ 1: CodeCommit リポジトリを作成する
- ステップ 2: リポジトリにファイルを追加する
- ステップ 3: リポジトリの内容を参照する
- ステップ 4: プルリクエストを作成して共同作業を行う
- ステップ 5: クリーンアップ
- ステップ 6: 次のステップ

ステップ 1: CodeCommit リポジトリを作成する

CodeCommit コンソールを使用して CodeCommit リポジトリを作成できます。このチュートリアルで使用するリポジトリがすでにある場合は、このステップをスキップできます。

> **Note**
> 使用状況によっては、リポジトリの作成またはアクセスに対して課金される場合があります。詳細については、CodeCommit 製品情報ページの料金を参照してください。

CodeCommit リポジトリを作成するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタを使用して、リポジトリ AWS リージョン を作成する を選択します。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。
3. [Repositories (リポジトリ)] ページで、[Create repository (リポジトリの作成)] を選択します。
4. [リポジトリの作成] ページの [リポジトリ名] に、新しいリポジトリの名前を入力します (例: **`MyDemoRepo`**)。

> **Note**
>
> リポジトリ名は 100 文字以内 (大文字と小文字は区別する) で入力してください。詳細については、「制限」を参照してください。

5. (オプション) [Description (説明)] に、説明 (例: **`My demonstration repository`**) を入力します。この説明は、お客様と他のユーザーがリポジトリの用途を識別するのに役立ちます。
6. (オプション) タグの追加 を選択して、1 つ以上のリポジトリタグ (AWS リソースの整理と管理に役立つカスタム属性ラベル) をリポジトリに追加します。詳細については、「でのリポジトリのタグ付け AWS CodeCommit」を参照してください。
7. (オプション) 追加設定を展開して、このリポジトリ内のデータを暗号化および復号するためにデフォルト AWS マネージドキー または独自のカスタマーマネージドキーを使用するかどうかを指定します。独自のカスタマーマネージドキーを使用する場合は、リポジトリを作成する AWS リージョン でそのキーが使用可能であること、およびキーがアクティブであることを確認する必要があります。詳細については、「AWS Key Management ServiceAWS CodeCommit リポジトリの および 暗号化」を参照してください。
8. (オプション) このリポジトリに Java または Python コードが含まれ、CodeGuru Reviewer でコードを分析する場合は、[Enable Amazon CodeGuru Reviewer for Java and Python] (Java および Python 用に Amazon CodeGuru Reviewer を有効化) を選択します。CodeGuru Reviewer は複数の機械学習モデルを使用して、コードの欠陥を検出し、プルリクエストの改善と修正を自動的に提案します。詳細については、Amazon CodeGuru Reviewer ユーザーガイドを参照してください。
9. [Create] を選択します。

> **Note**
>
> リポジトリに MyDemoRepo 以外の名前を使用する場合は、その名前を残りのステップで必ず使用してください。

リポジトリを開いたら、CodeCommit コンソールから直接ファイルを追加する方法が表示されます。

ステップ 2: リポジトリにファイルを追加する

リポジトリにファイルを追加するには以下の方法で行います。

- CodeCommit コンソールでファイルを作成します。コンソールでリポジトリ用の最初のファイルを作成すると、main という名前のブランチが作成されます。このブランチは、リポジトリのデフォルトのブランチです。
- CodeCommit コンソールを使用して、ローカルコンピュータからファイルをアップロードします。コンソールからリポジトリの最初のファイルをアップロードすると、main という名前のブランチが作成されます。このブランチは、リポジトリのデフォルトのブランチです。
- Git クライアントを使用してこのリポジトリをローカルコンピュータにクローンし、CodeCommit リポジトリにファイルの追加、コミット、およびプッシュを行います。ブランチは Git からの最初のコミットの一部として作成され、リポジトリのデフォルトのブランチとして設定されます。ブランチの名前は、Git クライアントがデフォルトで選択するものです。最初のブランチの名前として main を使用するように Git クライアントを設定することをご検討ください。

> **Note**
>
> ブランチを作成し、リポジトリのデフォルトのブランチをいつでも変更できます。詳細については、「AWS CodeCommit リポジトリでのブランチの使用」を参照してください。

もっとも簡単な開始方法は、CodeCommit コンソールを開いてファイルを追加することです。これを実行して、main という名前のリポジトリのデフォルトのブランチも作成します。を使用してファイルを追加し、リポジトリへの最初のコミットを作成する方法については AWS CLI、「 を使用してリポジトリの最初のコミットを作成する AWS CLI」を参照してください。

リポジトリにファイルを追加するには

1. リポジトリのナビゲーションバーで [コード] を選択します。
2. [ファイルの追加] を選択し、ファイルの作成、あるいはコンピュータからファイルのアップロードを選択します。このチュートリアルでは、両方を実行する方法を示します。
3. ファイルを追加するには、次の操作を行います。

   a. ブランチのドロップダウンリストで、ファイルを追加するブランチを選択します。デフォルトのブランチが自動的に選択されます。この例では、デフォルトブランチの名前は `main` です。別のブランチにファイルを追加する場合は、別のブランチを選択します。

b. [File name (ファイル名)] に、ファイルの名前を入力します。コードエディタで、ファイルのコードを入力します。

c. [Author name (作者名)] に、他のリポジトリーユーザーに表示する名前を入力します。

d. [E メールアドレス] に E メールアドレスを入力します。

e. (オプション) [メッセージのコミット] に短いメッセージを入力します。これはオプションですが、このファイルを追加した理由をチームメンバーが把握しやすいように、コミットメッセージを追加することをお勧めします。コミットメッセージを入力しない場合は、デフォルトメッセージが使用されます。

f. [変更のコミット] を選択します。

ファイルをアップロードするには、次の操作を行います。

- ファイルをアップロードしている場合は、アップロードするファイルを選択します。

- [Author name (作者名)] に、他のリポジトリーユーザーに表示する名前を入力します。
- [E メールアドレス] に E メールアドレスを入力します。
- (オプション) [メッセージのコミット] に短いメッセージを入力します。これはオプションですが、このファイルを追加した理由をチームメンバーが把握しやすいように、コミットメッセージを追加することをお勧めします。コミットメッセージを入力しない場合は、デフォルトメッセージが使用されます。
- [変更のコミット] を選択します。

詳細については、「AWS CodeCommit リポジトリ内のファイルの使用」を参照してください。

Git クライアントを使用してリポジトリをクローンするには、Git をローカルコンピュータにインストールし、CodeCommit リポジトリをクローンします。いくつかのファイルをローカルリポジトリに追加し、CodeCommit リポジトリにプッシュします。詳細な説明については、「Git と CodeCommit の開始方法」を参照してください。Git は使い慣れているが、CodeCommit リポジトリでこれを実行する方法が不明な場合は、コミットを作成する、ステップ 2: ローカルリポジトリを作成する、または リポジトリへの接続 で例や説明を確認することができます。

CodeCommit リポジトリにファイルを追加したら、それらのファイルをコンソールで表示できます。

ステップ 3: リポジトリの内容を参照する

CodeCommit コンソールを使用して、リポジトリ内のファイルを確認したり、ファイルの内容をすばやく読み取ったりすることができます。これは、どのブランチをチェックアウトするか、リポジトリのローカルコピーを作成するかどうかを判断するのに役立ちます。

リポジトリを参照するには

1. [Repositories] (リポジトリ) から [MyDemoRepo] を選択します。
2. このページには、リポジトリのデフォルトブランチのコンテンツが表示されます。別のブランチを表示したり、特定のタグのコードを表示するには、リストから表示するブランチまたはタグを選択します。次のスクリーンショットで、ビューは main ブランチに設定されています。

3. リポジトリ内のファイルの内容を表示するには、リストからファイルを選択します。表示されるコードの色を変更するには、設定アイコンを選択します。

詳細については、「リポジトリでのファイルの参照」を参照してください。

4. リポジトリのコミット履歴を参照するには、[Commits (コミット)] を選択します。コンソールには、デフォルトブランチのコミット履歴が新しい日付順に表示されます。作成者、日付などによってコミットの詳細を確認します。

5. ブランチ 別または Git タグ 別にコミット履歴を表示するには、リストから表示するブランチまたはタグを選択します。

6. コミットとその親コミットとの違いを表示するには、省略形式のコミット ID を選択します。空白の変更の表示/非表示、変更のインライン表示 (統合ビュー)、または並べて表示 (分割ビュー) など、変更の表示方法を選択できます。

> **Note**
>
> コードやその他のコンソール設定を表示するための設定は、変更するたびにブラウザの Cookie として保存されます。詳細については、「ユーザー設定の操作」を参照してください。

7. コミットに関するすべてのコメントを表示するには、そのコミットを選択し、変更をスクロールしてインラインで表示します。自分のコメントを追加して、他のユーザーによって行われたコメントに返信することもできます。

   詳細については、「コミットについてコメントする」を参照してください。

8. 2 つのコミット指定子 (タグ、ブランチ、コミット ID など) の違いを表示するには、ナビゲーションペインで [コミット]、[コミットの比較] の順に選択します。

詳細については、「リポジトリのコミット履歴を参照する」および「コミットの比較」を参照してください。

9. [コミット] で、[Commit Visualizer] タブを選択します。

コミットグラフが表示され、グラフでは各コミットの件名が該当するコミットポイントの横に示されます。件名の表示は 80 文字に制限されています。

10. コミットの詳細を確認するには、省略コミット ID を選択します。特定のコミットからグラフをレンダリングするには、グラフ内のポイントを選択します。詳細については、「リポジトリのコミット履歴のグラフを表示する」を参照してください。

ステップ 4: プルリクエストを作成して共同作業を行う

リポジトリで他のユーザーと作業する場合、コードを共同作業して変更を確認することができます。プルリクエストを作成して、他のユーザーがブランチのコード変更を確認してコメントできるようにすることができます。プルリクエストの承認ルールを 1 つ以上作成することもできます。たとえば、プルリクエストをマージする前に、少なくとも 2 人の他のユーザーがプルリクエストを承認する必要がある承認ルールを作成できます。プルリクエストが承認されたら、それらの変更を送信先ブランチにマージできます。リポジトリの通知を設定した場合、リポジトリーユーザーは、リポジトリーイベント (例: プルリクエストの場合、コードにコメントがついた時) に関する通知を E メールで受け取ることができます。詳細については、「AWS CodeCommit リポジトリイベントの通知を設定する」を参照してください。

> **Important**
> プルリクエストを作成する前に、レビューするコードの変更が含まれるブランチを作成する必要があります。詳細については、「ブランチを作成する」を参照してください。

プルリクエストを作成し、共同作業するには

1. ナビゲーションペインで、[プルリクエスト] を選択します。
2. [プルリクエスト] で、[プルリクエストの作成] を選択します。

   > **Tip**
   > また、プルリクエストは、[Branches] や [Code] から作成することもできます。

   [Create pull request] の [Source] で、レビューする変更が含まれるブランチを選択します。[送信先] で、プルリクエストがクローズされた際に確認済みのコードをマージするブランチを確認します。[Compare] を選択します。
3. マージの詳細と変更を確認し、プルリクエストにレビューする変更とコミットが含まれていることを確認します。その場合、[タイトル] にこのレビューのタイトルを入力します。これは、リポジトリのプルリクエストのリストに表示されるタイトルです。[Description (説明)] に、このレビューに関する詳細やレビューアにとって有益な情報を入力します。[Create] を選択します。

4. プルリクエストは、リポジトリのプルリクエストのリストに表示されます。開いているリクエスト、閉じたリクエスト、自分が作成したリクエストなどを表示するようにビューをフィルタリングできます。

5. プルリクエストに承認ルールを追加して、特定の条件が満たされていることを確認してからマージできます。プルリクエストに承認ルールを追加するには、リストからプルリクエストを選択します。[Approvals (承認)] タブで、[Create approval rule (承認ルールの作成)] を選択します。
6. [Rule name (ルール名)] で、ルールにわかりやすい名前を付けます。たとえば、プルリクエストをマージする前に 2 人のユーザーにプルリクエストの承認をリクエストする場合は、ルール

に **Require two approvals before merge** という名前を付けます。[Number of approvals needed (必要な承認の数)] に必要な番号、「**2**」を入力します。デフォルトは 1 です。[Submit] を選択します。承認ルールと承認プールメンバーの詳細については、「プルリクエストの承認ルールを作成する」を参照してください。

7. リポジトリの通知を設定し、プルリクエストイベントをユーザーに通知することを選択した場合、ユーザーは新しいプルリクエストに関する E メールを受信します。ユーザーは、特定の行のコードへの変更とコメント、ファイル、プルリクエスト自体を表示できます。テキストや絵文字でコメントに返信することもできます。必要に応じて、プルリクエストブランチに変更をプッシュでき、これによりプルリクエストが更新されます。

8. リクエストに加えられた変更に問題がなければ、[Approve (承認)] を選択します。プルリクエストに対して承認ルールが設定されていない場合でも、プルリクエストを承認することを選択できます。これにより、プルリクエストを確認したことと、変更の承認を明確に記録できます。気が変わった場合は、承認を取り消すこともできます。

> **Note**
>
> プルリクエストを作成した場合は、プルリクエストは承認できません。

9. すべてのコード変更が確認され、合意に達したことに満足したら、次のいずれかを実行します。

- ブランチのマージなしでプルリクエストを終了するには、[Close pull request (プルリクエストの終了)] を選択します。
- ブランチをマージしてプルリクエストを閉じる場合は、[Merge (マージ)] を選択します。送信元と送信先の差異に応じてコードに使用できるマージ戦略を選び、マージが完了した後に送信元のブランチを自動的に削除するかどうかを選択できます。選択が決まったら、[Merge pull request (プルリクエストのマージ)] を選択してマージを完了します。

- 自動的に解決できないマージ競合がブランチにある場合、CodeCommit コンソールで解決するか、ローカルの Git クライアントを使用してブランチをマージし、マージをプッシュできま

す。詳細については、「AWS CodeCommit リポジトリのプルリクエストの競合を解決する」を参照してください。

> **Note**
> ローカルリポジトリの git merge コマンドを使用して変更をプッシュすることで、いつでも手動でプルリクエストブランチを含めたブランチをマージできます。

詳細については、「プルリクエストの操作」および「承認ルールテンプレートの操作」を参照してください。

ステップ 5: クリーンアップ

CodeCommit リポジトリを必要としなくなった場合には、CodeCommit リポジトリおよびこの演習で使用した他のリソースを削除して、ストレージ領域に対して課金され続けないようにすることが必要です。

> **Important**
> このアクションは元に戻すことができません。削除したリポジトリは ローカルリポジトリや共有リポジトリに複製できなくなります。また、ローカルリポジトリまたは共有リポジトリから、そのリポジトリに対してデータをプルまたはプッシュしたり、Git オペレーションを実行できなくなります。
> リポジトリの通知を設定した場合、リポジトリを削除するとリポジトリ用に作成された Amazon CloudWatch Events ルールも削除されます。そのルールのターゲットとして使用された Amazon SNS トピックは削除されません。
> リポジトリにトリガーを設定した場合、リポジトリを削除しても、それらのトリガーのターゲットとして設定した Amazon SNS トピックまたは Lambda 関数は削除されません。不要なリソースは必ず削除してください。詳細については、「リポジトリからトリガーを削除する」を参照してください。

CodeCommit リポジトリを削除するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。

2. [リポジトリ] で、削除するリポジトリを選択します。このトピックの命名規則に従っていれば、MyDemoRepo という名前になります。
3. ナビゲーションペインで [Settings] (設定) をクリックします。
4. [設定] ページの [リポジトリの削除] で、[リポジトリの削除] を選択します。
5. 「**delete**」と入力後、[削除] を選択します。リポジトリは完全に削除されます。

ステップ 6: 次のステップ

CodeCommit とその機能のいくつかを理解できたところで、以下のことを検討します。

- Git と CodeCommit を初めて使用する場合や、CodeCommit で Git を使用する例を確認する場合は、Git と CodeCommit の開始方法 チュートリアルに進んでください。
- CodeCommit リポジトリの他の部分を操作する場合は、リポジトリの共有 を参照してください。(他のアマゾン ウェブ サービスアカウントのユーザーとリポジトリを共有する場合は、ロールを使用して AWS CodeCommit リポジトリへのクロスアカウントアクセスを設定する を参照してください。)
- リポジトリを CodeCommit に移行する場合は、CodeCommit に移行する のステップを実行してください。
- リポジトリを継続的な配信パイプラインに追加する場合は、シンプルなパイプラインのチュートリアルの手順に従ってください。
- コミュニティの例など、CodeCommit と統合する製品やサービスの詳細については、製品およびサービスの統合 を参照してください。

Git と の開始方法 AWS CodeCommit

Git と CodeCommit を初めて使用するお客様向けに、このチュートリアルでは、シンプルなコマンドの使用方法について説明します。Git にすでに精通している場合は、このチュートリアルをスキップし、「CodeCommit の開始方法 」に進むことができます。

このチュートリアルでは、CodeCommit リポジトリのローカルコピーを表すリポジトリを作成します。このリポジトリは、ローカルリポジトリと呼びます。

ローカルリポジトリの作成後、いくつかの変更を加えます。その後、変更を CodeCommit リポジトリに送信 (プッシュ) します。

また、2 人のユーザーが個別にローカルリポジトリへの変更をコミットし、その変更を CodeCommit リポジトリにプッシュする、チーム環境をシミュレートします。その後、ユーザーは CodeCommit リポジトリから変更を自身のローカルリポジトリにプルして、他のユーザーが加えた変更を確認します。

また、ブランチとタグを作成し、CodeCommit リポジトリでのアクセス許可を管理します。

このチュートリアルを完了後、自身のプロジェクトで Git と CodeCommit の主要概念を使用するには、十分な練習が必要です。

以下の前提条件と設定を完了します。

- IAM ユーザーにアクセス許可を割り当てます。
- HTTPS、SSH、または git-remote-codecommit を使用してリポジトリに接続するように CodeCommit を設定します。これらの選択肢の詳細については、「のセットアップ AWS CodeCommit」を参照してください。
- リポジトリの作成を含むすべてのオペレーションでコマンドラインまたはターミナル AWS CLI を使用する場合は、 を設定します。

トピック

- ステップ 1: CodeCommit リポジトリを作成する
- ステップ 2: ローカルリポジトリを作成する
- ステップ 3: 最初のコミットを作成する
- ステップ 4: 最初のコミットをプッシュする
- ステップ 5: CodeCommit リポジトリを共有し、別のコミットをプッシュしてプルする
- ステップ 6: ブランチを作成して共有する
- ステップ 7: タグを作成して共有する
- ステップ 8: アクセス許可を設定する
- ステップ 9: クリーンアップする

ステップ 1: CodeCommit リポジトリを作成する

このステップでは、CodeCommit コンソールを使用してリポジトリを作成します。

使用する CodeCommit リポジトリが既にある場合は、このステップをスキップできます。

> **Note**
> 使用状況によっては、リポジトリの作成またはアクセスに対して課金される場合があります。詳細については、CodeCommit 製品情報ページの料金を参照してください。

CodeCommit リポジトリを作成するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタを使用して、リポジトリ AWS リージョン を作成する を選択します。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。
3. [Repositories (リポジトリ)] ページで、[Create repository (リポジトリの作成)] を選択します。
4. [リポジトリの作成] ページの [リポジトリ名] に、新しいリポジトリの名前を入力します (例: **`MyDemoRepo`**)。

   > **Note**
   > リポジトリ名は 100 文字以内 (大文字と小文字は区別する) で入力してください。詳細については、「制限」を参照してください。

5. (オプション) [Description (説明)] に、説明 (例: **`My demonstration repository`**) を入力します。この説明は、お客様と他のユーザーがリポジトリの用途を識別するのに役立ちます。
6. (オプション) タグの追加 を選択して、1 つ以上のリポジトリタグ (AWS リソースの整理と管理に役立つカスタム属性ラベル) をリポジトリに追加します。詳細については、「でのリポジトリのタグ付け AWS CodeCommit」を参照してください。
7. (オプション) 追加設定を展開して、このリポジトリ内のデータを暗号化および復号するためにデフォルト AWS マネージドキー または独自のカスタマーマネージドキーを使用するかどうかを指定します。独自のカスタマーマネージドキーを使用する場合は、リポジトリを作成する AWS リージョン でそのキーが使用可能であること、およびキーがアクティブであることを確認する必要があります。詳細については、「AWS Key Management ServiceAWS CodeCommit リポジトリの および 暗号化」を参照してください。
8. (オプション) このリポジトリに Java または Python コードが含まれ、CodeGuru Reviewer でコードを分析する場合は、[Enable Amazon CodeGuru Reviewer for Java and Python] (Java および Python 用に Amazon CodeGuru Reviewer を有効化) を選択します。CodeGuru Reviewer は複数の機械学習モデルを使用して、コードの欠陥を検出し、プルリクエストの改善と修正を自

動的に提案します。詳細については、Amazon CodeGuru Reviewer ユーザーガイドを参照してください。

9. [Create] を選択します。

> **Note**
>
> このチュートリアルの残りのステップでは、CodeCommit リポジトリの名前として `MyDemoRepo` を使用します。別の名前を選択した場合は、このチュートリアル全体でそれを使用してください。

ターミナルやコマンドラインからリポジトリを作成する方法を含め、リポジトリの作成の詳細については、「リポジトリの作成」を参照してください。

ステップ 2: ローカルリポジトリを作成する

このステップでは、ローカルマシン上のローカルリポジトリをお客様のレポジトリに接続するように設定します。そのためには、ローカルマシン上のローカルリポジトリを表すディレクトリを選択します。Git を使用して、空の CodeCommit リポジトリのコピーをそのディレクトリ内に複製し、初期化します。次に、コミットに注釈を付けるために使用する Git ユーザー名と E メールアドレスを指定します。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタで、リポジトリ AWS リージョン が作成された を選択します。リポジトリは に固有です AWS リージョン。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。
3. 接続するリポジトリをリストから見つけて選択します。[クローン URL] を選択してから、リポジトリのクローン作成やリポジトリへの接続時に使用するプロトコルを選択します。これにより、クローン URL が複製されます。

   - IAM ユーザー、または AWS CLIに含まれている認証情報ヘルパーで Git 認証情報を使用している場合は、HTTPS URL をコピーします。
   - ローカルコンピュータで git-remote-codecommit コマンドを使用している場合は、HTTPS (GRC) URL をコピーします。
   - IAM ユーザーで SSH パブリック/プライベートキーペアを使用している場合は、SSH URL をコピーします。

> **Note**
>
> リポジトリのリストの代わりにようこそページが表示された場合は、サインイン AWS リージョン している に AWS アカウントに関連付けられたリポジトリはありません。リポジトリを作成するには、「the section called “リポジトリの作成”」を参照するか、「Git と CodeCommit の開始方法」チュートリアルのステップに従います。

4. (オプション) リポジトリのデフォルトのブランチの名前として **main** を使用するようにローカル Git クライアントを設定することをお勧めします。これは、このガイドのすべての例で、デフォルトのブランチに使用される名前です。また、コンソールで最初のコミットを行う場合に CodeCommit が使用するデフォルトのブランチ名とも同じです。次のコマンドを実行して、デフォルトのブランチ名をシステムにおいてグローバルに設定します。

```
git config --global init.defaultBranch main
```

すべてのリポジトリで異なるデフォルトのブランチ名を使用する場合は、**main** を任意の名前に置き換えてください。このチュートリアルでは、デフォルトのブランチの名前が main であることを前提としています。

リポジトリごとに異なるデフォルトのブランチ名を使用する場合は、この属性をグローバル (--local) ではなくローカル (--global) に設定できます。

5. ターミナルまたはコマンドプロンプトで、git clone コマンドを使用してリポジトリのクローンを作成し、ステップ 3 でコピーしたクローン URL を指定します。クローン URL は、使用するプロトコルと設定によって異なります。例えば、HTTPS と Git 認証情報を使用して、米国東部 (オハイオ) リージョンで *MyDemoRepo* という名前のリポジトリのクローンを作成する場合:

```
git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-
demo-repo
```

HTTPS を git-remote-codecommit で使用している場合:

```
git clone codecommit://MyDemoRepo my-demo-repo
```

SSH を使用している場合:

```
git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-
repo
```

> **Note**
>
> リポジトリのクローン作成時にエラーが表示される場合は、ローカルコンピュータに必要なセットアップが完了していない可能性があります。詳細については、「のセットアップ AWS CodeCommit」を参照してください。

ステップ 3: 最初のコミットを作成する

このステップでは、ローカルリポジトリで最初のコミットを作成します。そのためには、ローカルリポジトリに 2 つのサンプルファイルを作成します。Git を使用して変更をステージングし、ローカルリポジトリにコミットします。

1. テキストエディタを使用して、ディレクトリに以下の 2 つのサンプルテキストファイルを作成します。ファイルに `cat.txt` および `dog.txt` という名前を付けます。

```
cat.txt
-------
The domestic cat (Felis catus or Felis silvestris catus) is a small, usually furry,
 domesticated, and carnivorous mammal.
```

```
dog.txt
-------
The domestic dog (Canis lupus familiaris) is a canid that is known as man's best
 friend.
```

2. git config を実行して、プレースホルダー *your-user-name* と *your-email-address* で表されている、ユーザー名と E メールアドレスをローカルレポジトリに追加します。これにより、コミットを識別しやすくなります。

```
git config --local user.name "your-user-name"
git config --local user.email your-email-address
```

3. ローカルリポジトリの作成時にデフォルトのブランチ名をグローバルに設定しなかった場合は、次のコマンドを実行して、デフォルトのブランチ名を **main** に設定します。

```
git config --local init.defaultBranch main
```

4. git add を実行して、変更をステージングします。

```
git add cat.txt dog.txt
```

5. git commit を実行して、以下の変更をコミットします。

```
git commit -m "Added cat.txt and dog.txt"
```

> **Tip**
> 先ほど行ったコミットの詳細を表示するには、git log を実行します。

ステップ 4: 最初のコミットをプッシュする

このステップでは、ローカルリポジトリから CodeCommit リポジトリにコミットをプッシュします。

ローカルリポジトリ (git push) のデフォルトブランチから、Git が CodeCommit リポジトリ (`origin`) に使用するデフォルトのリモート名でコミットをプッシュするために `main` を実行します。

```
git push -u origin main
```

> **Tip**
> ファイルを CodeCommit リポジトリにプッシュしたら、CodeCommit コンソールを使用してコンテンツを表示できます。詳細については、「リポジトリでのファイルの参照」を参照してください。

ステップ 5: CodeCommit リポジトリを共有し、別のコミットをプッシュしてプルする

このステップでは、CodeCommit リポジトリに関する情報を他のチームメンバーと共有します。チームメンバーは、この情報を使用してローカルコピーを取得し、変更を加えた後、そのローカルコピーを CodeCommit リポジトリにプッシュします。その後、CodeCommit リポジトリからローカルリポジトリに変更をプルします。

このチュートリアルでは、ステップ 2 で作成したディレクトリとは別のディレクトリが Git によって作成されるようにすることで、同じチームのユーザーをシミュレートします（通常、このディレクトリは別のマシンにあります）。この新しいディレクトリは、CodeCommit リポジトリのコピーです。既存のディレクトリまたはこの新しいディレクトリには、変更が個別に加えられます。これらのディレクトリへの変更を識別する唯一の方法は、CodeCommit リポジトリからプルすることです。

同じローカルマシン上にあっても、既存のディレクトリをローカルリポジトリ、新しいディレクトリを共有リポジトリと呼びます。

新しいディレクトリから、CodeCommit リポジトリの別のコピーを取得します。次に、新しいサンプルファイルを追加し、共有リポジトリに変更をコミットした後、共有リポジトリから CodeCommit リポジトリにコミットをプッシュします。

最後に、変更をリポジトリからローカルリポジトリにプルした後に参照して、他のユーザーがコミットした変更を確認します。

1. `/tmp` ディレクトリか `c:\temp` ディレクトリに切り替えます。
2. git clone を実行して、リポジトリのコピーを共有リポジトリにプルダウンします。

   HTTPS の場合:

   ```
   git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
    shared-demo-repo
   ```

   git-remote-codecommit で HTTPS を使用する場合:

   ```
   git clone codecommit://MyDemoRepo shared-demo-repo
   ```

   SSH の場合:

```
git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo shared-
demo-repo
```

> **Note**
>
> Windows オペレーティングシステムで SSH を使用してリポジトリのクローンを作成する場合は、以下のように SSH キー ID の接続文字列への追加が必要になる場合があります。
>
> ```
> git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/
> repos/MyDemoRepo my-demo-repo
> ```
>
> 詳細については、「Windows で SSH 接続をセットアップする手順」を参照してください。

このコマンドでは、MyDemoRepo は CodeCommit リポジトリの名前です。shared-demo-repo は /tmp ディレクトリか c:\temp ディレクトリで Git が作成するディレクトリの名前です。ディレクトリの作成後、Git はリポジトリのコピーを shared-demo-repo ディレクトリにプルダウンします。

3. shared-demo-repo ディレクトリに切り替えます。

```
(For Linux, macOS, or Unix) cd /tmp/shared-demo-repo
(For Windows) cd c:\temp\shared-demo-repo
```

4. git config を実行して、プレースホルダー *other-user-name* と *other-email-address* で表されている、別のユーザー名と E メールアドレスを追加します。これにより、他のユーザーが作成するコミットをより簡単に識別できます。

```
git config --local user.name "other-user-name"
git config --local user.email other-email-address
```

5. テキストエディタを使用して、shared-demo-repo ディレクトリに以下のサンプルテキストファイルを作成します。ファイルに horse.txt という名前を付けます。

```
horse.txt
-------
```

```
The horse (Equus ferus caballus) is one of two extant subspecies of Equus ferus.
```

6. git add を実行して、変更を共有リポジトリにステージングします。

```
git add horse.txt
```

7. git commit を実行して、変更を共有リポジトリにコミットします。

```
git commit -m "Added horse.txt"
```

8. ローカルリポジトリ (git push) のデフォルトブランチから、Git が CodeCommit リポジトリ (`origin`) に使用するデフォルトのリモート名で初期コミットをプッシュするために `main` を実行します。

```
git push -u origin main
```

9. ローカルリポジトリに切り替えて、git pull を実行し、共有リポジトリが CodeCommit リポジトリに対して作成したコミットをローカルリポジトリにプルします。その後、git log を実行して、共有リポジトリから開始されたコミットを確認します。

ステップ 6: ブランチを作成して共有する

このステップでは、ローカルリポジトリにブランチを作成し、いくつかの変更を加えた後、そのブランチを CodeCommit リポジトリにプッシュします。その後、CodeCommit リポジトリから共有リポジトリにブランチをプルします。

ブランチを使用すると、異なるバージョンのリポジトリの内容を個別に開発できます (たとえば、チームメンバーの作業に影響を与えずに、新しいソフトウェア機能の開発に取り組むことができます)。その機能が安定したら、ブランチをソフトウェアのより安定したブランチにマージします。

Git を使用してブランチを作成し、そのブランチが最初のコミットを参照するようにします。Git を使用して CodeCommit リポジトリにブランチをプッシュします。その後、共有リポジトリに切り替え、Git を使用して新しいブランチを共有 ローカルリポジトリにプルし、そのブランチを確認します。

1. ローカルリポジトリから、ブランチの名前 (git checkout など) と、ローカルリポジトリで作成した最初のコミットの ID を指定して、`MyNewBranch` を実行します。

   コミットの ID がわからない場合は、git log を実行して取得します。コミットのユーザー名と E メールアドレスが、他のユーザーのものではなく、お客様のものであることを確認し

てください。これにより、main が、CodeCommit リポジトリの安定したバージョンであり、MyNewBranch が、比較的不安定な新しい機能に使用されるブランチであることがシミュレートされます。

```
git checkout -b MyNewBranch commit-ID
```

2. git push を実行して、ローカルリポジトリから CodeCommit リポジトリに新しいブランチを送信します。

```
git push origin MyNewBranch
```

3. 続いて、そのブランチを共有リポジトリにプルし、結果を確認します。

   1. 共有リポジトリディレクトリ (shared-demo-repo) に切り替えます。
   2. 新しいブランチをプルします (git fetch origin)。
   3. ブランチがプルされたことを確認します (git branch --all はリポジトリ内のすべてのブランチのリストを表示します)。
   4. 新しいブランチに切り替えます (git checkout MyNewBranch)。
   5. MyNewBranch ブランチに切り替えたことを確認するには、git status または git branch を実行します。出力に現在のブランチが表示されます。この場合は MyNewBranch です。
   6. ブランチ内のコミットのリストを表示します (git log)。

   呼び出す Git コマンドのリストは以下のとおりです。

```
git fetch origin
git branch --all
git checkout MyNewBranch
git branch or git status
git log
```

4. main ブランチに戻り、そのコミットのリストを表示します。Git のコマンドは以下のようになります。

```
git checkout main
git log
```

5. ローカルリポジトリ内の main ブランチに切り替えます。git status または git branch を実行できます。出力に現在のブランチが表示されます。この場合は main です。Git のコマンドは以下のようになります。

```
git checkout main
git branch or git status
```

ステップ 7: タグを作成して共有する

このステップでは、2 つのタグをローカルリポジトリに作成し、コミットに関連付けてから、CodeCommit リポジトリにプッシュします。その後、CodeCommit リポジトリから共有リポジトリに変更をプルします。

タグは、コミット (またはブランチや別のタグ) に人間が読める名前を付けるために使用されます。たとえば、これを行うのは、コミットに v2.1 というタグを付ける場合です。コミット、ブランチ、またはタグには、任意の数のタグを関連付けることができますが、個々のタグは 1 つのコミット、ブランチ、またはタグにのみ関連付けることができます。このチュートリアルでは、1 つのコミットに release タグを、もう 1 つのコミットに beta タグを付けます。

Git を使用してタグを作成し、最初のコミットに release タグを付け、他のユーザーが作成したコミットに beta タグを付けます。その後、Git を使用して CodeCommit リポジトリにタグをプッシュします。その後、共有リポジトリに切り替え、Git を使用して新しいタグを共有 ローカルリポジトリにプルし、そのタグを確認します。

1. ローカルリポジトリから、新しいタグの名前 (git tag) と、ローカルリポジトリで作成した最初のコミットの ID を指定して、release を実行します。

   コミットの ID がわからない場合は、git log を実行して取得します。コミットのユーザー名と E メールアドレスが、他のユーザーのものではなく、お客様のものであることを確認してください。これにより、コミットが CodeCommit リポジトリの安定したバージョンであることがシミュレートされます。

   ```
   git tag release commit-ID
   ```

   git tag をもう一度実行して、他のユーザーからのコミットに beta タグを付けます。これにより、それが比較的不安定な新しい機能に使用されるコミットであることがシミュレートされます。

```
git tag beta commit-ID
```

2. git push --tags を実行して、CodeCommit リポジトリにタグを送信します。
3. ここで、タグを共有リポジトリにプルし、結果を確認します。

   1. 共有リポジトリディレクトリ (shared-demo-repo) に切り替えます。
   2. 新しいタグをプルします (git fetch origin)。
   3. タグがプルされたことを確認します (git tag はリポジトリ内のタグのリストを表示します)。
   4. 各ログ (git log release および git log beta) に関する情報を表示します。

   呼び出す Git コマンドのリストは以下のとおりです。

```
git fetch origin
git tag
git log release
git log beta
```

4. これをローカルリポジトリでも試してみます。

```
git log release
git log beta
```

ステップ 8: アクセス許可を設定する

このステップでは、共有リポジトリを CodeCommit リポジトリに同期させるアクセス許可をユーザーに付与します。これは任意の手順です。ユーザーが Git 認証情報を使用する場合、または CodeCommit レポジトリにアクセスする IAM ユーザーに SSH キーペアが使用される場合に、CodeCommit レポジトリへのアクセスコントロール方法を学習したいユーザーに推奨されます。

> **Note**
>
> フェデレーティッドアクセス、一時的な認証情報、または IAM Identity Center などのウェブ ID プロバイダーを使用している場合は、ID プロバイダーのユーザー、アクセス、アクセス許可を設定してから、git-remote-codecommit を使用します。詳細については、git-remote-

codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順および認証情報をローテーションして AWS CodeCommit リポジトリに接続するを参照してください。

このステップでは、IAM コンソールを使用してユーザーを作成します。このユーザーにはデフォルトで、共有リポジトリを CodeCommit リポジトリに同期させるためのアクセス許可がありません。このアクセス許可の有無を確認するには、git pull を実行します。新しいユーザーに同期させるアクセス権限がないと、このコマンドは機能しません。次に、IAM コンソールに戻り、ユーザーによる git pull の使用を許可するポリシーを適用します。もう一度、git pull を実行して、このアクセス許可の有無を確認します。

このステップは、アマゾン ウェブ サービスアカウントに IAM ユーザーを作成するアクセス許可があると仮定して書かれています。これらのアクセス許可がない場合は、このステップを実行することはできません。チュートリアルで使用したリソースをクリーンアップするには、「ステップ 9: クリーンアップする」に進みます。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。

   で使用したものと同じユーザー名とパスワードでサインインします設定

2. ナビゲーションペインで [ユーザー] を選択し、続いて [Create New Users (新しいユーザーの作成)] を選択します。
3. 最初の [Enter User Names] (ユーザー名を入力) ボックスに、サンプルユーザー名 (**`JaneDoe-CodeCommit`** など) を入力します。[Generate an access key for each user (ユーザーごとにアクセスキーを生成)] ボックスを選択し、[作成] を選択します。
4. [Show User Security Credentials] を選択します。アクセスキー ID とシークレットアクセスキーをメモするか、[Download Credentials (認証情報のダウンロード)] を選択します。
5. Git 認証情報を使用した HTTPS ユーザーのセットアップ の手順に従って、IAM ユーザーの認証情報を生成して入力します。

   SSH を使用する場合は、「SSH および Linux、macOS、または Unix: Git と CodeCommit 用にパブリックキーとプライベートキーをセットアップする」または「ステップ 3: Git および CodeCommit 用のパブリックキーとプライベートキーをセットアップする」の指示に従って、パブリックキーとプライベートキーでユーザーを設定します。

6. git pull を実行します。以下のエラーが表示されます。

   HTTPS の場合:

```
fatal: unable to access 'https://git-codecommit.us-
east-2.amazonaws.com/v1/repos/repository-name/': The requested URL
returned error: 403.
```

SSH の場合:

```
fatal: unable to access 'ssh://git-codecommit.us-east-2.amazonaws.com/
v1/repos/repository-name/': The requested URL returned error: 403.
```

このエラーが表示されるのは、新しいユーザーに共有リポジトリを CodeCommit リポジトリに同期させるアクセス許可がないためです。

7. IAM コンソールに戻ります。ナビゲーションペインで、[Policies] を選択し、[Create Policy] を選択します。([Get Started] ボタンが表示された場合は、そのボタンを選択してから、[Create Policy] を選択します)。
8. [独自のポリシーを作成] の横で、[選択] を選択します。
9. [Policy Name] (ポリシー名) ボックスに、名前 (例: **CodeCommitAccess-GettingStarted**) を入力します。
10. [Policy Document] (ポリシードキュメント) ボックスに以下のように入力して、IAM ユーザーがその IAM ユーザー自身に関連付けられたリポジトリからプルできるようにします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Tip**
>
> IAM ユーザーがその IAM ユーザー自身に関連付けられたリポジトリにコミットをプッシュできるようにするには、代わりに以下のように入力します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull",
        "codecommit:GitPush"
      ],
      "Resource": "*"
    }
  ]
}
```

他の CodeCommit アクションや、リソースに対してユーザーに付与できるその他のアクセス許可については、AWS CodeCommitの認証とアクセスコントロール を参照してください。

11. ナビゲーションペインで [Users] (ユーザー) を選択します。
12. ポリシーをアタッチするサンプルユーザー名 (**JaneDoe-CodeCommit** など) を選択します。
13. [Permissions] タブを選択します。
14. [管理ポリシー] で、[Attach Policy (ポリシーのアタッチ)] を選択します。
15. 先ほど作成した **CodeCommitAccess-GettingStarted** ポリシーを選択してから、[Attach Policy] (ポリシーのアタッチ) を選択します。
16. git pull を実行します。今回は、コマンドが機能し、"`Already up-to-date`" メッセージが表示されます。
17. HTTPS を使用する場合は、元の Git 認証情報、または git-remote-codecommit を使用している場合は通常のプロファイルに切り替えます。詳細については、「Git 認証情報を使用した HTTPS ユーザーのセットアップ」または「git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」の指示を参照してください。

    SSH を使用している場合は、元のキーに切り替えます。詳細については、「SSH および Linux、macOS、または Unix: Git と CodeCommit 用にパブリックキーとプライベートキーをセットアップする」または「ステップ 3: Git および CodeCommit 用のパブリックキーとプライベートキーをセットアップする」を参照してください。

次は、このチュートリアルの最後のステップです。

ステップ 9: クリーンアップする

このステップでは、このチュートリアルで使用した CodeCommit リポジトリを削除します。これにより、ストレージ領域に対して使用料金は発生しなくなります。

CodeCommit リポジトリの削除後は不要になるため、ローカルマシン上のローカルリポジトリと共有リポジトリも削除します。

> **Important**
>
> このリポジトリを削除すると、そのリポジトリを ローカルリポジトリまたは共有リポジトリに複製できなくなります。また、ローカルリポジトリまたは共有リポジトリに対してデータをプルすることもプッシュすることもできなくなります。このアクションは元に戻すことができません。

CodeCommit リポジトリを削除するには (コンソール)

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Dashboard] (ダッシュボード) ページのリポジトリのリストから、[MyDemoRepo] を選択します。
3. ナビゲーションペインで [Settings] (設定) をクリックします。
4. [設定] ページの [リポジトリの削除] で、[リポジトリの削除] を選択します。
5. [Type the name of the repository to confirm deletion] (削除の確認のため、リポジトリ名を入力) の横にあるボックスに **MyDemoRepo** と入力し、[Delete] (削除) を選択します。

CodeCommit リポジトリを削除するには (AWS CLI)

delete-repository コマンドを実行します。

```
aws codecommit delete-repository --repository-name MyDemoRepo
```

ローカルリポジトリと共有リポジトリを削除するには

Linux、macOS、Unix の場合:

```
cd /tmp
rm -rf /tmp/my-demo-repo
rm -rf /tmp/shared-demo-repo
```

Windows の場合:

```
cd c:\temp
rd /s /q c:\temp\my-demo-repo
rd /s /q c:\temp\shared-demo-repo
```

との製品とサービスの統合 AWS CodeCommit

デフォルトでは、CodeCommit は多数の AWS サービスと統合されています。CodeCommit は、 以外の製品やサービスでも使用できます AWS。以下の情報は、使用する製品やサービスを統合するための CodeCommit の設定に役立ちます。

トピック

- 他の AWS サービスとの統合
- コミュニティから統合の例

他の AWS サービスとの統合

CodeCommit は、次の AWS サービスと統合されています。

AWS Amplify	AWS Amplify は、 AWSを使用したスケーラブルなモバイルアプリケーションの作成、設定、実装を容易にします。Amplify はモバイルバックエンドをシームレスにプロビジョニングして管理し、バックエンドを iOS、Android、ウェブ、React Native のフロントエンドと簡単に統合するためのシンプルなフレームワークを提供します。また、Amplify は、フロントエンドとバックエンドの両方のアプリケーションリリースプロセスを自動化し、機能をより迅速に提供することができます。 Amplify コンソールで CodeCommit リポジトリに接続できます。Amplify コンソールを承認すると、Amplify はリポジトリプロバイダーからアクセストークンを取得しますが、 AWS サーバーにはトークンを保存しません。Amplify は、特定のリポジトリにのみインストールされているデプロイキーを使用してリポジトリにアクセスします。

	詳細はこちら: • AWS Amplify ユーザーガイド • ご利用開始にあたって
AWS Cloud9	AWS Cloud9 には、クラウド上でソフトウェアのコード作成、ビルド、実行、テスト、デバッグ、リリースに使用するツールのコレクションが含まれています。このツールのコレクションは、 AWS Cloud9 統合開発環境、または IDE と呼ばれます。 ウェブブラウザから AWS Cloud9 IDE にアクセスします。この IDE では、リッチなコード編集エクスペリエンスを実現しており、複数のプログラミング言語、ランタイムデバッガ、および組み込みターミナルがサポートされています。 詳細はこちら: • AWS Cloud9 ユーザーガイド • AWS CodeCommit のサンプル AWS Cloud9 • AWS Cloud9 との統合 AWS CodeCommit

AWS CloudFormation	AWS CloudFormation は、 AWS リソースをモデル化してセットアップするのに役立つサービスです。これにより、リソースの管理に費やす時間を減らし、アプリケーションに集中する時間を増やすことができます。CodeCommit リポジトリを含むリソースを記述するテンプレートを作成し、 AWS CloudFormation がそれらのリソースのプロビジョニングと設定を行います。 詳細はこちら: • AWS CodeCommit リポジトリリソースページ
AWS CloudTrail	CloudTrail は、Amazon Web Services アカウントによって、または Amazon Web Services アカウントに代わって行われた AWS API コールおよび関連イベントをキャプチャし、指定した Amazon S3 バケットにログファイルを配信します。 AWS CodeCommit コンソールからの API コール、 からの CodeCommit コマンド、ローカル Git クライアント AWS CLI、および CodeCommit API から API コールをキャプチャするように CloudTrail を設定できます。 詳細はこちら: • を使用した AWS CodeCommit API コールのログ記録 AWS CloudTrail

Amazon CloudWatch Events	CloudWatch Events は、 AWS リソースの変更を記述するシステムイベントのほぼリアルタイムのストリームを提供します。すぐに設定できる簡単なルールを使用して、ルールに一致したイベントを 1 つ以上のターゲット関数またはストリームに振り分けることができます。CloudWatch Events が発生すると、運用上の変更が認識されます。CloudWatch Events は、オペレーションの変更に応答し、必要に応じて、応答メッセージを環境に送り、機能をアクティブ化し、変更を行い、状態情報を収集することによって、アクションを実行します。 Amazon Simple Queue Service、Amazon Kinesis など、他の AWS サービスのストリーム、関数、タスク、またはその他のプロセスをターゲットにすることで、CodeCommit リポジトリをモニタリングし AWS Lambda、リポジトリイベントに応答するように CloudWatch Events を設定できます。 Amazon Kinesis 詳細はこちら: • CloudWatch Events ユーザーガイド • AWS CodeCommit Events • ブログ記事: Amazon CloudWatch Events と JGit を使用してサーバーレス AWS CodeCommit ワークフローを構築する

AWS CodeBuild	CodeBuild は、ソースコードをコンパイルして、単体テストを実行し、デプロイする準備が整ったアーティファクトを生成する完全に管理されたクラウド構築サービスです。ビルドするソースコードとビルド仕様を CodeCommit リポジトリに格納することができます。CodeBuild は CodeCommit で直接使用することも、CodePipeline を使用して継続的配信パイプラインに CodeBuild と CodeCommit の両方を組み込むこともできます。 詳細はこちら: • ビルドを計画する • ビルドプロジェクトを作成する • CodePipeline を AWS CodeBuild と共に使用してビルドを実行する
Amazon CodeGuru Reviewer	Amazon CodeGuru Reviewer は、プログラム分析と機械学習を使用して一般的な問題を検出し、Java または Python コードにおける修正点を提案する自動化されたコードレビューサービスです。アマゾン ウェブ サービスアカウントのリポジトリを CodeGuru Reviewer に関連付けることができます。実行すると、CodeGuru Reviewer はサービスにリンクされたロールを作成します。これにより、CodeGuru Reviewer は、関連付けが行われた後に作成されたすべてのプルリクエストのコードを分析できます。 詳細はこちら: • AWS CodeCommit リポジトリを Amazon CodeGuru Reviewer に関連付ける、または関連付けを解除する • Amazon CodeGuru Reviewer ユーザーガイド

AWS CodePipeline	CodePipeline は、ソフトウェアをリリースするために必要な手順のモデル化、視覚化、および自動化に使用できる継続的な配信サービスです。CodePipeline を設定して CodeCommit リポジトリをパイプラインのソースアクションとして使用し、変更のビルド、テスト、およびデプロイを自動化できます。 詳細はこちら: • CodePipeline と を使用したシンプルなパイプラインのチュートリアル AWS CodeCommit • CodeCommit リポジトリを使用して、パイプライン用に Amazon CloudWatch Events 変更検出に移行する • パイプラインの自動開始に使用される変更検出方法
AWS Elastic Beanstalk	Elastic Beanstalk は、アプリケーションを実行するインフラストラクチャを気にすることなく、 AWS クラウドへのアプリケーションのデプロイと管理を容易にするマネージドサービスです。Elastic Beanstalk コマンドラインインターフェイス (EB CLI) を使用して、新規または既存の CodeCommit リポジトリから直接アプリケーションをデプロイできます。 詳細はこちら: • AWS CodeCommit で EB CLI を使用する • 既存の AWS CodeCommit リポジトリの使用 • eb codesource (EB CLI コマンド)

AWS Key Management Service	AWS KMS は、データの暗号化に使用される暗号化キーの作成と管理を容易にするマネージド型サービスです。デフォルトでは、CodeCommit は を使用してリポジトリ AWS KMS を暗号化します。 詳細はこちら: • AWS KMS および 暗号化
AWS Lambda	Lambda を使用することで、サーバーのプロビジョニングや管理をすることなく、コードを実行できます。リポジトリイベントに応答して Lambda 関数を呼び出す CodeCommit リポジトリのトリガーを設定できます。 詳細はこちら: • Lambda 関数のトリガーを作成する • AWS Lambda デベロッパーガイド
Amazon Simple Notification Service	Amazon SNS は、アプリケーション、エンドユーザー、およびデバイスでクラウドからすぐに通知を送受信できるようにするウェブサービスです。リポジトリイベントに応答して Amazon SNS 通知を送信する CodeCommit リポジトリのトリガーを設定できます。Amazon SNS 通知を使用して、他の AWS サービスと統合することもできます。例えば、Amazon SNS 通知を使用して、Amazon Simple Queue Service キューにメッセージを送信できます。 詳細はこちら: • Amazon SNS トピック用のトリガーを作成する • Amazon Simple Notification Service デベロッパーガイド

コミュニティから統合の例

以下のセクションは、ブログの投稿や記事、およびコミュニティで提供されている例へのリンクです。

> **Note**
>
> これらのリンクは情報提供のみを目的としており、包括的なリストや例の内容の支持とはみなされません。 AWS は、外部コンテンツの内容や正確性について責任を負いません。

トピック

- ブログ記事
- コードサンプル

ブログ記事

- AWS CodeCommitでプルリクエスト承認者として SonarQube を統合する

  プルリクエストをマージする前に、SonarQube 品質分析を成功させる必要がある CodeCommit リポジトリを作成する方法を説明します。

  発行日: 2019 年 12 月 12 日
- GitLab AWS CodeCommitAWS CodePipelineAWS CodeBuild との間の移行

  複数のリポジトリを GitLab AWS CodeCommit から に移行し、 AWS CodePipeline と を使用して CI/CD パイプラインを設定する方法について説明します AWS CodeBuild。

  発行日: 2019 年 11 月 22 日
- AWS CodePipeline、、 を使用した GitFlow AWS CodeCommitAWS CodeBuildの実装 AWS CodeDeploy

  AWS CodePipeline、 AWS CodeCommit、 AWS CodeBuildおよび を使用して GitFlow を実装する方法について説明します AWS CodeDeploy。

  発行日: 2019 年 2 月 22 日
- 複数の AWS アカウントで Git AWS CodeCommit を使用する

複数のアマゾン ウェブ サービスアカウントで Git の設定を管理する方法を説明します。

発行日: 2019 年 2 月 12 日

- AWS CodeBuildおよび を使用した AWS CodeCommit プルリクエストの検証 AWS Lambda

  AWS CodeCommit、 AWS CodeBuild、および を使用してプルリクエストを検証する方法について説明します AWS Lambda。提案された変更をデフォルトのブランチにマージする前にテストを実行することで、プルリクエストで高レベルの品質を確保し、潜在的な問題を捉え、変更に関連してデベロッパーの信頼を高めることができます。

  発行日: 2019 年 2 月 11 日

- でのフェデレーティッドアイデンティティの使用 AWS CodeCommit

  ビジネスで使用される ID AWS CodeCommit を使用して のリポジトリにアクセスする方法について説明します。

  公開日: 2018 年 10 月 5 日

- でのブランチへのアクセスの絞り込み AWS CodeCommit

  コンテキストキーを使用する IAM ポリシーを作成および適用して、リポジトリブランチへのコミットを制限する方法について説明します。

  2018 年 5 月 16 日公開

- Fargate を使用して AWS リージョン間で AWS CodeCommit リポジトリをレプリケートする

  サーバーレスアーキテクチャを使用して、1 つの AWS リージョンから別のリージョンへの CodeCommit リポジトリの継続的なレプリケーションを設定する方法について説明します。

  2018 年 4 月 11 日公開

- AWS OpsWorks for Chef Automate インフラストラクチャの配布

  CodePipeline、CodeCommit、CodeBuild、および を使用して、クックブックやその他の設定が 1 つ以上の にある 2 つ以上の Chef Server に一貫してデプロイ AWS Lambda されるようにする方法について説明します AWS リージョン。

  2018 年 3 月 9 日公開

- ピーナツバターとチョコレート: AWS CodeCommit を使用した Azure 関数の CI/CD パイプライン

コードが CodeCommit リポジトリに保存される、PowerShell ベースの Azure 関数の CI/CD パイプラインを作成する方法について説明します。

2018 年 2 月 19 日公開

- Amazon ECR AWS CodePipeline、 AWS CodeCommit、 AWS CodeBuild、 を使用した Kubernetes への継続的なデプロイ AWS Lambda

  Kubernetes と を AWS 一緒に使用して、コンテナベースのアプリケーション用のフルマネージドの継続的デプロイパイプラインを作成する方法について説明します。

2018 年 1 月 11 日公開

- AWS CodeCommit プルリクエストを使用してコードレビューをリクエストし、コードについて話し合う

  プルリクエストを使用して、CodeCommit リポジトリでのコード変更を確認、コメント、インタラクティブに反復する方法について説明します。

2017 年 11 月 20 日公開

- Amazon CloudWatch Events と JGit を使用してサーバーレス AWS CodeCommit ワークフローを構築する

  CodeCommit リポジトリイベントと他の AWS サービスのターゲットアクションを使用して、リポジトリの変更を処理する CloudWatch Events ルールを作成する方法について説明します。コミットに Git コミットメッセージポリシーを適用し、CodeCommit リポジトリをレプリケートし、CodeCommit リポジトリを Amazon S3 にバックアップする AWS Lambda 関数はその一例です。

2017 年 8 月 3 日公開

- への移行 AWS CodeCommit

  SourceTree を使用するときに別の Git リポジトリから CodeCommit への移行の一環として、2 つのリポジトリにコードをプッシュする方法について説明します。

2016 年 9 月 6 日公開

- Appium、、 AWS CodeCommit Jenkins、および を使用した継続的なテストの設定 AWS Device Farm

Appium、CodeCommit、Jenkins、Device Farm を使用してモバイルデバイスの連続テストプロセスを作成する方法について説明します。

2016 年 2 月 2 日公開

- 複数の Amazon Web Services アカウントの Git リポジトリでの AWS CodeCommit の使用

  CodeCommit リポジトリをクローンする方法について説明し、1 つのコマンドで、そのリポジトリへの接続に特定の IAM ロールを使用するように認証情報ヘルパーを設定します。

2015 年 11 月公開

- OpsWorks と の統合 AWS CodeCommit

  OpsWorks が CodeCommit からアプリと Chef クックブックを自動的に取得する方法について説明します。

2015 年 8 月 25 日公開

- AWS CodeCommit および GitHub の認証情報ヘルパーを使用する

  CodeCommit と GitHub 認証情報ヘルパーの両方で動作するように gitconfig ファイルを設定する方法について説明します。

2015 年 9 月公開

- Eclipse AWS CodeCommit からの の使用

  Eclipse で EGit ツールを使用して CodeCommit で作業する方法について説明します。

2015 年 8 月公開

- AWS CodeCommit と Amazon EC2 ロール認証情報

  CodeCommit リポジトリへの自動エージェントアクセスを設定するときに Amazon EC2 のインスタンスプロファイルを使用する方法について説明します。

2015 年 7 月発行

- Jenkins AWS CodeCommit との統合

  CodeCommit と Jenkins を使用して、2 つの単純な連続統合 (CI) シナリオをサポートする方法について説明します。

2015 年 7 月発行

- レビューボード AWS CodeCommit との統合

  Review Board コードレビューシステムを使用して CodeCommit を開発ワークフローに統合する方法について説明します。

  2015 年 7 月発行

コードサンプル

以下は、CodeCommit ユーザーにとって興味深いコードサンプルです。

- Mac OS X Script to Periodically Delete Cached Credentials in the OS X Certificate Store

  Mac OS X で認証情報ヘルパーを CodeCommit に使用すると、キャッシュされた認証情報の問題に精通することができます。このスクリプトは、1 つのソリューションを示しています。

  筆者: Nico Coetzee

  2016 年 2 月投稿

でのリポジトリの使用 AWS CodeCommit

リポジトリは、CodeCommit の基本的なバージョン管理オブジェクトです。プロジェクトのコードとファイルを安全に保存する場所です。また、最初のコミットから最新の変更までのプロジェクト履歴も保存されます。リポジトリを他のユーザーと共有することができるので、プロジェクトで一緒に作業することができます。リポジトリに AWS タグを追加すると、リポジトリユーザーがイベントに関する E メールを受信するように通知を設定できます (たとえば、別のユーザーがコードにコメントするなど) 。リポジトリのデフォルト設定を変更したり、その内容を参照したりすることもできます。リポジトリ用のトリガーを作成して、コードプッシュやその他のイベントによって E メールやコード関数などのアクションがトリガーされるようにすることができます。ローカルコンピュータのリポジトリ (ローカルリポジトリ) を設定して、変更を複数のリポジトリにプッシュすることもできます。

CodeCommit リポジトリに変更をプッシュする前に、Amazon Web Services アカウントで IAM ユーザーを設定するか、フェデレーションアクセスまたは一時認証情報のアクセスを設定する必要があります。詳細については、「ステップ 1: CodeCommit の初期設定」および「git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」を参照してください。

CodeCommit でリポジトリの他の部分を操作する方法については、ファイルの操作、プルリクエストの操作、コミットの操作、ブランチの操作、および ユーザー設定の操作 を参照してください。CodeCommit に移行する方法については、「CodeCommit に移行する」を参照してください。

トピック

- AWS CodeCommit リポジトリを作成する
- AWS CodeCommit リポジトリに接続する
- AWS CodeCommit リポジトリを共有する
- AWS CodeCommit リポジトリイベントの通知を設定する
- でのリポジトリのタグ付け AWS CodeCommit
- AWS CodeCommit リポジトリのトリガーを管理する
- AWS CodeCommit リポジトリを Amazon CodeGuru Reviewer に関連付ける、または関連付けを解除する
- CodeCommit リポジトリの詳細を表示する
- AWS CodeCommit リポジトリ設定を変更する
- ローカルリポジトリと AWS CodeCommit リポジトリとの間で変更を同期させる
- 追加の Git リポジトリにコミットをプッシュする
- ロールを使用して AWS CodeCommit リポジトリへのクロスアカウントアクセスを設定する
- AWS CodeCommit リポジトリを削除する

AWS CodeCommit リポジトリを作成する

AWS CodeCommit コンソールまたは AWS Command Line Interface（AWS CLI) を使用して、空の CodeCommit リポジトリを作成します。リポジトリを作成した後にタグを追加するには、「リポジトリにタグを追加する」を参照してください。

これらの手順では、設定 のステップを完了していることを前提としています。

> **Note**
>
> 使用状況によっては、リポジトリの作成またはアクセスに対して課金される場合があります。詳細については、CodeCommit 製品情報ページの料金を参照してください。

トピック

- リポジトリを作成する (コンソール)
- リポジトリを作成する (AWS CLI)

リポジトリを作成する (コンソール)

CodeCommit リポジトリを作成するには

> **Important**
>
> でリポジトリを作成する AWS CodeCommit ことは、新規のお客様は利用できなくなりました。の既存のお客様は、通常どおり API を AWS CodeCommit 引き続き使用できます。詳細はこちら

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタで、リポジトリ AWS リージョン を作成する を選択します。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。
3. [Repositories (リポジトリ)] ページで、[Create repository (リポジトリの作成)] を選択します。
4. [Create repository (リポジトリの作成)] ページの [Repository name (リポジトリ名)] に、リポジトリの名前を入力します。

   > **Note**
   >
   > リポジトリ名では大文字と小文字が区別されます。Amazon Web Services アカウントの名前は、 AWS リージョン 内で一意である必要があります。

5. (オプション) [Description (説明)] にリポジトリの説明を入力します。この説明は、お客様と他のユーザーがリポジトリの用途を識別するのに役立ちます。

   > **Note**
   >
   > コンソールの説明フィールドに [Markdown] と表示され、すべての HTML 文字とすべての有効な Unicode 文字を使用できます。アプリケーションデベロッパーとして `GetRepository` または `BatchGetRepositories` API を使用していて、ウェブブラ

ウザにレポジトリの説明フィールドを表示させる場合は、CodeCommit API リファレンスを参照してください。

6. (オプション) タグの追加 を選択して、1 つ以上のリポジトリタグ (AWS リソースの整理と管理に役立つカスタム属性ラベル) をリポジトリに追加します。詳細については、「でのリポジトリのタグ付け AWS CodeCommit」を参照してください。
7. (オプション) 追加設定を展開して、このリポジトリ内のデータの暗号化と復号にデフォルト AWS マネージドキー または独自のカスタマーマネージドキーを使用するかどうかを指定します。独自のカスタマーマネージドキーを使用する場合は、リポジトリを作成する AWS リージョン で使用可能であること、およびキーがアクティブであることを確認する必要があります。詳細については、「AWS Key Management ServiceAWS CodeCommit リポジトリの および 暗号化」を参照してください。
8. (オプション) このリポジトリに Java または Python コードが含まれており、CodeGuru Reviewer で分析する場合は、[Enable Amazon CodeGuru Reviewer for Java and Python] (Java および Python 用に Amazon CodeGuru Reviewer を有効化) を選択します。CodeGuru Reviewer は、複数の機械学習モデルを使用して、コードの欠陥を検出し、プルリクエストの改善と修正を提案します。詳細については、Amazon CodeGuru Reviewer ユーザーガイドを参照してください。
9. [Create] を選択します。

リポジトリを作成したら、CodeCommit コンソール、またはローカルの Git クライアントを使用するか、CodeCommit リポジトリを任意の IDE に統合して、リポジトリに接続し、コードを追加することができます。詳細については、「のセットアップ AWS CodeCommit」を参照してください。リポジトリを継続的な配信パイプラインに追加することもできます。詳細については、「シンプルなパイプラインの演習」を参照してください。

新しい CodeCommit リポジトリに関する情報 (例: リポジトリのクローン作成時に使用する URL など) を取得するには、リストからリポジトリ名を選択するか、リポジトリ名の横の使用する接続プロトコルを使用します。

このリポジトリを他のユーザーと共有するには、リポジトリのクローンを作成するために必要な HTTPS リンクまたは SSH リンクを送信する必要があります。リポジトリにアクセスするために必要なアクセス権限があることを確認してください。詳細については、「リポジトリの共有」および「AWS CodeCommitの認証とアクセスコントロール」を参照してください。

リポジトリを作成する (AWS CLI)

を使用して CodeCommit リポジトリ AWS CLI を作成できます。コンソールとは異なり、 AWS CLI を使用した作成ではリポジトリにタグを追加できます。

> **Important**
>
> でリポジトリを作成する AWS CodeCommit ことは、新規のお客様は利用できなくなりました。の既存のお客様は、通常どおり API を AWS CodeCommit 引き続き使用できます。詳細はこちら

1. AWS リージョン リポジトリが存在する AWS CLI で が設定されていることを確認します。リージョンを確認するには、コマンドラインまたはターミナルで次のコマンドを実行し、デフォルトのリージョン名の情報を確認します。

   ```
   aws configure
   ```

   デフォルトのリージョン名は、CodeCommit のリポジトリ AWS リージョン の と一致する必要があります。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。

2. 次のように指定して create-repository コマンドを実行します。

   - CodeCommit を一意に識別する名前 (`--repository-name` オプションを指定)。

     > **Note**
     >
     > この名前は、Amazon Web Services アカウント全体で一意である必要があります。

   - CodeCommit に関するオプションのコメント (`--repository-description` オプションを指定)。
   - CodeCommit リポジトリへのタグとして (`--tags` オプションで) 使用するオプションのキー値ペア (複数可)。
   - このリポジトリを暗号化および復号するときに使用するオプションのカスタマーマネージドキー。すべてのリポジトリは、転送中と保管時のいずれも AWS KMSのキーを使用して暗号化されます。キーが指定されていない場合は、デフォルトの AWS マネージドキー`aws/codecommit`が使用されます。

例えば、MyDemoRepo という名前の CodeCommit リポジトリを、"My demonstration repository" という説明と、*Team* という名前のキー名で値が *Saanvi* のタグを使用して作成するには、このコマンドを使用します。

```
aws codecommit create-repository --repository-name MyDemoRepo --repository-
description "My demonstration repository" --tags Team=Saanvi
```

> **Note**
>
> コンソールの説明フィールドに [Markdown] と表示され、すべての HTML 文字とすべての有効な Unicode 文字を使用できます。アプリケーションデベロッパーとして GetRepository または BatchGetRepositories API を使用していて、ウェブブラウザにレポジトリの説明フィールドを表示させる場合は、CodeCommit API リファレンスを参照してください。

3. 成功すると、このコマンドは次の情報を持つ repositoryMetadata オブジェクトを出力します。

   - 説明 (repositoryDescription)。
   - 一意のシステム生成 ID (repositoryId)。
   - 名前 (repositoryName)。
   - CodeCommit リポジトリに関連付けられた Amazon Web Services アカウントの ID (accountId)。

   以下は、前述のコマンド例に基づく出力例です。

```
{
    "repositoryMetadata": {
        "repositoryName": "MyDemoRepo",
        "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo",
        "lastModifiedDate": 1446071622.494,
        "repositoryDescription": "My demonstration repository",
        "cloneUrlHttp": "https://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo",
        "defaultBranch": main,
        "kmsKeyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
```

```
        "creationDate": 1446071622.494,
        "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
        "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
        "accountId": "111111111111"
    }
}
```

> **Note**
> リポジトリの作成時に追加されたタグは、出力で返されません。リポジトリに関連付けられたタグのリストを表示するには、list-tags-for-resource コマンドを実行します。

4. CodeCommit リポジトリの名前と ID を書き留めます。特に AWS CLIを使用する場合は、CodeCommit に関する情報をモニタリングし、変更する必要があります。

   名前または ID を忘れた場合は、「」の指示に従ってくださいCodeCommit リポジトリの詳細を表示する (AWS CLI)

リポジトリを作成すると、リポジトリに接続してコードを追加することができます。詳細については、「リポジトリへの接続」を参照してください。リポジトリを継続的な配信パイプラインに追加することもできます。詳細については、「シンプルなパイプラインの演習」を参照してください。

AWS CodeCommit リポジトリに接続する

初めて CodeCommit リポジトリに接続するときは、通常、ローカルマシンにその内容のクローンを作成します。CodeCommit コンソールから直接、リポジトリのファイルの追加や、ファイルの編集を行うことができます。ローカルリポジトリが既にある場合は、CodeCommit リポジトリをリモートとして追加することもできます。このトピックでは、CodeCommit リポジトリに接続する手順について説明します。既存のリポジトリを CodeCommit に移行する方法については、「CodeCommit に移行する」を参照してください。

> **Note**
> 使用状況によっては、リポジトリの作成またはアクセスに対して課金される場合があります。詳細については、CodeCommit 製品情報ページの料金を参照してください。

トピック

- CodeCommit リポジトリに接続するための前提条件
- CodeCommit リポジトリのクローンを作成してリポジトリに接続する
- ローカルリポジトリを CodeCommit リポジトリに接続する

CodeCommit リポジトリに接続するための前提条件

CodeCommit リポジトリのクローンを作成するか、ローカルリポジトリを CodeCommit リポジトリに接続するには、以下の操作を行います。

- ローカルコンピュータで、CodeCommit への接続に必要なソフトウェアなどの設定を行う必要があります。これには、Git のインストールと設定が含まれます。詳細については、「設定 」および「Git と の開始方法 AWS CodeCommit」を参照してください。
- 接続先の CodeCommit リポジトリのクローン URL を指定します。詳細については、「 リポジトリの詳細の表示」を参照してください。

  CodeCommit リポジトリが作成されていない場合は、「リポジトリの作成」の手順に従って、CodeCommit リポジトリのクローン URL をコピーし、このページに戻ります。

  CodeCommit リポジトリは作成されているが、リポジトリ名が不明な場合は、「 リポジトリの詳細の表示」の手順を行います。
- 接続する CodeCommit リポジトリのローカルコピーを保存するには、ローカルマシン上の場所を指定する必要があります。(この CodeCommit リポジトリのローカルコピーは、ローカルリポジトリと呼ばれます。) 次に、この場所から Git コマンドを指定して実行します。例えば、テスト目的で一時的にクローンを作成する場合は、`/tmp` (Linux、macOS、UNIX の場合) または `c:\temp` (Windows の場合) を使用します。これは、これらの例で使用されるディレクトリパスです。

> **Note**
>
> 任意のディレクトリを使用することができます。長期間使用するためにリポジトリのクローンを作成する場合は、作業ディレクトリからクローンを作成することを検討し、一時ファイルには使用しないでください。`/tmp` または `c:\temp` 以外のディレクトリを使用している場合は、これらの手順を行うときにそのディレクトリを置き換えてください。

CodeCommit リポジトリのクローンを作成してリポジトリに接続する

ローカルリポジトリが作成されていない場合は、この手順のステップを実行して、CodeCommit リポジトリのクローンをローカルマシンに作成します。

1. 前提条件 (例: 設定) を完了します。

   > **Important**
   >
   > セットアップが完了していない場合は、リポジトリへの接続や、クローンの作成を行うことはできません。

2. `/tmp` ディレクトリまたは `c:\temp` ディレクトリから、Git を使用して clone コマンドを実行します。*`MyDemoRepo`* という名前のリポジトリのクローンを米国東部 (オハイオ) リージョンに作成する方法を、以下の例に示します。

   Git 認証情報を使用した HTTPS の場合、または AWS CLIに含まれている認証情報ヘルパーの場合:

   ```
   git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
   ```

   git-remote-codecommit を使用した HTTPS の場合 (AWS CLIでデフォルトのプロファイルと AWS リージョン が設定されていると仮定):

   ```
   git clone codecommit://MyDemoRepo my-demo-repo
   ```

   SSH の場合:

   ```
   git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
   ```

   この例では、`git-codecommit.us-east-2.amazonaws.com` は、リポジトリが存在する米国東部 (オハイオ) リージョン リージョンの Git の接続ポイントです。`MyDemoRepo` は CodeCommit リポジトリの名前、`my-demo-repo` は Git によって `/tmp` ディレクトリまたは `c:\temp` ディレクトリに作成されるディレクトリ名を表します。CodeCommit AWS リージョン をサポートする とその Git 接続の詳細については AWS リージョン、「」を参照してくださいリージョンと Git 接続エンドポイント。

> **Note**
>
> Windows オペレーティングシステムで SSH を使用してリポジトリのクローンを作成する場合は、次のように SSH キー ID を接続文字列に追加する必要があります。
>
> ```
> git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
> ```
>
> 詳細については、「Windows で SSH 接続をセットアップする手順」および「トラブルシューティング」を参照してください。

Git でディレクトリが作成されたら、その新しく作成されたディレクトリに CodeCommit のコピーをダウンロードします。

CodeCommit リポジトリが新しく作成された場合、または空の場合は、空のリポジトリのクローンを作成中である旨のメッセージが表示されます。これは通常の動作です。

> **Note**
>
> Git で CodeCommit が見つからない、または CodeCommit に接続するためのアクセス許可が付与されていないことを表すエラーが表示された場合は、前提条件を満たしていることを確認してください。この前提条件には、IAM ユーザーへのアクセス許可の割り当てや、ローカルマシンでの Git や CodeCommit の IAM ユーザー認証情報のセットアップなどが含まれます。また、指定したリポジトリ名が正しいことを確認します。

ローカルリポジトリから CodeCommit リポジトリに正常に接続されたら、ローカルリポジトリから Git コマンドを実行して、コミットやブランチ、タグを作成できるだけでなく、CodeCommit リポジトリからプッシュおよびプルを行うことができます。

ローカルリポジトリを CodeCommit リポジトリに接続する

ローカルリポジトリが既に作成されており、リモートリポジトリとして CodeCommit リポジトリを追加するには、次のステップを完了します。リモートリポジトリが既に作成されており、CodeCommit リポジトリやその他のリモートリポジトリにコミットをプッシュするには、「2 つのリポジトリにコミットをプッシュする」のステップを行います。

1. 「前提条件」を完了します。
2. コマンドプロンプトまたはターミナルより、ローカルリポジトリディレクトリに切り替えて git remote add コマンドを実行し、ローカルリポジトリのリモートリポジトリとして CodeCommit リポジトリを追加します。

   例えば、以下のコマンドは、リモート (**origin**) を https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo に追加します。

   HTTPS の場合:

   ```
   git remote add origin https://git-codecommit.us-east-2.amazonaws.com/v1/repos/
   MyDemoRepo
   ```

   SSH の場合:

   ```
   git remote add origin ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
   MyDemoRepo
   ```

   このコマンドは何も返しません。
3. ローカルリポジトリのリモートとして CodeCommit リポジトリが追加されたことを確認するには、git remote -v コマンドを実行します。その結果、次のように出力されます。

   HTTPS の場合:

   ```
   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)

   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
   ```

   SSH の場合:

   ```
   origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)

   origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
   ```

ローカルリポジトリから CodeCommit リポジトリに正常に接続されたら、ローカルリポジトリから Git コマンドを実行して、コミットやブランチ、タグを作成できるだけでなく、CodeCommit リポジトリからプッシュおよびプルを行うことができます。

AWS CodeCommit リポジトリを共有する

CodeCommit リポジトリを作成したら、他のユーザーと共有できます。まず、CodeCommit にアクセスするときに、フェデレーティッドアクセス、一時認証情報、または IAM Identity Center などのウェブ ID プロバイダーのどれを使用するか、または IAM ユーザーに Git 認証情報または SSH キーペアを使用するかを決定します。前者を使用する場合は、ID プロバイダーのユーザー、アクセス、アクセス許可を設定してから、ユーザーが git-remote-codecommit を使用するための手順を提供する必要があります。詳細については、「git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」および「認証情報をローテーションして AWS CodeCommit リポジトリに接続する」を参照してください。

フェデレーティッドアクセスまたは ID プロバイダーで Git 認証情報または SSH キーペアを使用することはできませんが、多くの IDE はこれらの認証情報に対して最適に機能します。この場合は、クローンを作成するときと Git クライアントや IDE を使用してリポジトリに接続するときにユーザーに推奨するプロトコル (HTTPS または SSH) を決定します。次に、リポジトリを共有するユーザーに URL と接続情報を送信します。セキュリティ要件によっては、リポジトリを共有する際、アクセスを詳細に設定するため、または IAM ロールを作成して使用するために、IAM グループの作成、このグループへの管理ポリシーの適用、IAM ポリシーの編集も合わせて必要になることがあります。

> **Note**
>
> コンソールからリポジトリへのアクセスをユーザーに許可したら、リポジトリユーザーは Git クライアントや他の接続を設定することなく、コンソールで直接ファイルを追加または編集できます。詳細については、「AWS CodeCommit リポジトリへのファイルの作成または追加」および「AWS CodeCommit リポジトリ内のファイルの内容を編集する」を参照してください。

これらの手順は、「設定 」および「リポジトリの作成」のステップを既に完了していることを前提としています。

> **Note**
>
> 使用状況によっては、リポジトリの作成またはアクセスに対して課金される場合があります。詳細については、CodeCommit 製品情報ページの料金を参照してください。

トピック

- 接続プロトコルを選択してユーザーと共有する
- リポジトリの IAM ポリシーを作成する
- リポジトリユーザーの IAM グループを作成する
- 接続情報をユーザーと共有する

接続プロトコルを選択してユーザーと共有する

CodeCommit でリポジトリを作成すると、HTTPS 接続用と SSH 接続用の 2 つのエンドポイントが生成されます。どちらもネットワーク経由で安全な接続を提供します。ユーザーは、いずれかのプロトコルを使用できます。ユーザーに推奨するプロトコルに関係なく、エンドポイントはいずれもアクティブの状態を維持します。

HTTPS 接続を行うには、以下のいずれかが必要です。

- Git 認証情報。IAM ユーザーが IAM で生成できます。Git 認証情報は、リポジトリのユーザーがセットアップおよび使用する上で最も簡単な方法です。
- 引き受ける AWS アクセスキーまたはロール。リポジトリユーザーが認証情報プロファイルで設定する必要があります。git-remote-codecommit を設定するか (推奨)、 AWS CLIに含まれる認証情報ヘルパーを設定できます。これらはルートアカウントやフェデレーティッドユーザーが使用できる唯一の方法です。

SSH 接続を行うには、以下の作業が必要です。

- パブリックキーとプライベートキーのペアを生成する。
- パブリックキーを保存する。
- パブリックキーを IAM ユーザーに関連付ける。
- ローカルコンピュータで既知のホストファイルを設定する。
- ローカルコンピュータで設定ファイルを作成して管理する。

SSH 接続の設定プロセスは複雑であるため、CodeCommit に接続するには HTTPS および Git 認証情報を使用することをお勧めします。

HTTPS、SSH、Git、git-remote-codecommit、リモートリポジトリに関する詳細については、「設定」、「認証情報をローテーションして AWS CodeCommit リポジトリに接続する」、または Git のドキュメントを参照してください。

> **Note**
>
> Git ではさまざまな接続プロトコルがサポートされていますが、CodeCommit は、ローカルプロトコルまたは一般的な HTTP のようなセキュリティが確保されていないプロトコルとの接続はサポートしていません。

リポジトリの IAM ポリシーを作成する

AWS はCodeCommit の IAM で 3 つの管理ポリシーを提供します。これらのポリシーを編集したり、Amazon Web Services アカウントに関連付けられているすべてのリポジトリに適用したりすることはできません。ただし、これらのポリシーをテンプレートとして使用し、共有するリポジトリにのみ適用する独自のカスタム管理ポリシーを作成することができます。カスタマー管理ポリシーは、共有するリポジトリに特別に適用することができます。詳細については、「管理ポリシー」と「IAM ユーザーとグループ」を参照してください。

> **Tip**
>
> リポジトリへのアクセスを詳細に制御するには、複数のカスタマー管理ポリシーを作成して、別の IAM ユーザーおよびグループに適用します。

管理ポリシーの内容の確認、およびポリシーを使用してアクセス許可を作成および適用する方法については、「AWS CodeCommitの認証とアクセスコントロール」を参照してください。

リポジトリのカスタマー管理ポリシーを作成します。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. [ダッシュボード] ナビゲーションエリアで、[ポリシー] を選択し、次に [ポリシーの作成] を選択します。

3. [ポリシーの作成] ページで、[Import managed policy (マネージドポリシーのインポート)] を選択します。

4. [Import managed policies (マネージドポリシーのインポート)] ページの [フィルタポリシー] に「**`AWSCodeCommitPowerUser`**」と入力します。ポリシー名の横にあるボタンを選択し、[インポート] を選択します。

5. [Create policy] (ポリシーの作成) ページで [JSON] を選択します。次に示すように、CodeCommit アクションの `Resource` 行の「`*`」部分を、CodeCommit リポジトリの Amazon リソースネーム (ARN) に置き換えます。

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
 ]
```

> **Tip**
>
> CodeCommit リポジトリの ARN を確認するには、CodeCommit コンソールに移動し、リストからリポジトリ名を選択して [Settings] (設定) を選択します。詳細については、「リポジトリの詳細の表示」を参照してください。

このポリシーに複数のリポジトリを適用するには、リソースに ARN を指定して各リポジトリを追加します。次に示すように、各 Resource ステートメントはカンマで区切ります。

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
 "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo"
 ]
```

編集を完了したら、[ポリシーの確認] を選択します。

6. [Review Policy] (ポリシーの確認) ページの [Name] (名前) に、ポリシーの新しい名前 (例えば、*`AWSCodeCommitPowerUser-MyDemoRepo`*) を入力します。必要に応じて、このポリシーの説明を入力します。

7. [ポリシーの作成] を選択します。

リポジトリユーザーの IAM グループを作成する

リポジトリへのアクセスを管理するには、リポジトリユーザーの IAM グループを作成し、そのグループに IAM ユーザーを追加します。その後、前のステップで作成したカスタマー管理ポリシーをアタッチします。または、カスタマー管理ポリシーがアタッチされたロールを作成し、ユーザーにそのロールを引き継がせることもできます。

SSH を使用する場合は、IAMUserSSHKeys グループに別の管理ポリシーをアタッチする必要があります。この IAM 管理ポリシーにより、ユーザーは SSH 公開鍵をアップロードし、IAM ユーザーと関連付けて CodeCommit に接続できるようになります。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. [ダッシュボード] ナビゲーションエリアで、[グループ] を選択し、次に [Create New Group (新しいグループの作成)] を選択します。
3. [Set Group Name] (グループ名の設定) ページの [Group Name] (グループ名) に、グループの名前 (例: *MyDemoRepoGroup*) を入力し、[Next Step] (次のステップ) を選択します。ここで、グループ名の一部として、リポジトリ名を含めることを検討してください。

   > **Note**
   > この名前は、Amazon Web Services アカウント全体で一意である必要があります。

4. 前のセクションで作成したカスタマー管理ポリシー (例: AWSCodeCommitPowerUser-MyDemoRepo) の横にあるボックスをオンにします。
5. [Review] ページで、[Create Group] を選択します。IAM は、指定されたポリシーが既にアタッチされた状態でこのグループを作成します。このグループは、アマゾン ウェブ サービスアカウントに関連付けられたグループのリストに表示されます。
6. リストからグループを選択します。
7. グループの概要ページで、[ユーザー] タブを選択し、次に [Add Users to Group (グループにユーザーを追加)] を選択します。アマゾン ウェブ サービスアカウントに関連付けられているすべてのユーザーを示すリスト上で、CodeCommit リポジトリへのアクセスを許可するユーザーの横にあるボックスをオンにして、[Add Users] (ユーザーを追加) を選択します。

   > **Tip**
   > 検索ボックスに名前を入力して、ユーザーをすばやく見つけることができます。

8. ユーザーの追加が完了したら、IAM コンソールを閉じます。

接続情報をユーザーと共有する

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタで、リポジトリ AWS リージョン が作成された を選択します。リポジトリは に固有です AWS リージョン。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。
3. [リポジトリ] ページで、共有するリポジトリを選択します。
4. [Clone URL] で、ユーザーが使用するプロトコルを選択します。接続プロトコルのクローン URL がコピーされます。
5. のインストール、プロファイルの設定、Git のインストールなど AWS CLI、他の手順とともにクローン URL をユーザーに送信します。接続プロトコルの設定情報を含めるようにしてください (HTTPS など)。

次の E メール例では、ユーザーが 米国東部 (オハイオ) (us-east-2) リージョンで HTTPS 接続プロトコルと Git 認証情報を使用して MyDemoRepo リポジトリに接続するための情報を示します。この E メールでは、ユーザーが既に Git をインストールしており、操作に慣れていることを前提としています。

```
I've created a CodeCommit repository for us to use while working on our project.
The name of the repository is MyDemoRepo, and
it is in the  US East (Ohio) (us-east-2) region.
Here's what you need to do in order to get started using it:

1. Make sure that your version of Git on your local computer is 1.7.9 or later.
2. Generate Git credentials for your IAM user by signing into the IAM console
 here: https://console.aws.amazon.com/iam/  .
Switch to the Security credentials tab for your IAM user and choose the Generate button
 in HTTPS Git credentials for CodeCommit.
Make sure to save your credentials in a secure location!
3. Switch to a directory of your choice and clone the CodeCommit repository to your
 local machine by running the following command:
    git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-
demo-repo
4. When prompted for user name and password, use the Git credentials you just saved.
```

```
That's it!  If you'd like to learn more about using CodeCommit, you can start with the
 tutorial here.
```

全体的なセットアップ手順は、「」で確認できます設定

AWS CodeCommit リポジトリイベントの通知を設定する

指定したリポジトリイベントタイプについてリポジトリユーザーが E メールを受け取れるように、リポジトリの通知ルールを設定できます。イベントが通知ルールの設定と一致すると通知が送信されます。通知用の Amazon SNS トピックを作成するか、Amazon Web Services アカウントの既存のトピックを使用できます。CodeCommit コンソールと を使用して AWS CLI 、通知ルールを設定できます。

トピック

- リポジトリ通知ルールの使用
- 通知ルールの作成

- 通知の変更または無効化
- 通知の削除

リポジトリ通知ルールの使用

通知ルールを設定すると、他のユーザーに影響を及ぼすアクションがあったときにリポジトリユーザーに E メールを送信できます。たとえば、コミット時にコメントが作成されたときに通知を送信するよう通知ルールを設定できます。この設定では、リポジトリユーザーがコミット中のコード行についてコメントすると、他のリポジトリユーザーに E メールが届きます。サインインしてコメントを表示できます。コメントに応答しても E メールが生成されるため、リポジトリユーザーは常に把握することができます。

通知ルールはリポジトリトリガーとは異なり、2019 年 11 月 5 日以前に CodeCommit コンソールで設定できた通知とも異なります。

- 一部のリポジトリイベントに関する E メールを送信するために Amazon SNS を使用するようにトリガーを設定できますが、それらのイベントはブランチの作成やブランチへのコードのプッシュなどの操作イベントに制限されています。トリガーは、リポジトリイベントを評価するために CloudWatch Events ルールを使用しません。これらは範囲がさらに制限されています。トリガーの使用に関する詳細については、「リポジトリのトリガーの管理」を参照してください。
- 2019 年 11 月 5 日以前に設定された通知では、使用可能なイベントタイプが少なく、Amazon Chime チャットルームおよび Slack チャネルとの統合用に設定できませんでした。2019 年 11 月 5 日以前に設定された通知を引き続き使用できますが、このタイプの通知を作成することはできません。代わりに、通知ルールを作成して使用します。通知ルールを使用し、2019 年 11 月 5 日以前に作成された通知は無効化または削除することをお勧めします。詳細については、「通知ルールの作成」および「通知の削除」を参照してください。

通知ルールの作成

通知ルールを使用して、リポジトリでプルリクエストが作成されたときなど、重要な変更をユーザーに通知できます。通知ルールは、イベントと、通知の送信に使用される Amazon SNS トピックの両方を指定します。詳細については、「通知とは」を参照してください。

> **Note**
>
> この機能は、ヨーロッパ (ミラノ) リージョンでは使用できません。そのリージョンで使用可能なエクスペリエンスでの通知の設定方法については、「リポジトリ通知の設定」を参照してください。

コンソールまたは を使用して AWS CLI 、通知ルールを作成できます AWS CodeCommit。

通知ルールを作成するには (コンソール)

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/codecommit/ で CodeCommit コンソールを開きます。
2. [リポジトリ] を選択し、通知ルールを追加するリポジトリを選択します。
3. レポジトリページで、[Notify (通知)]、[Create notification rule (通知ルールの作成)] の順に選択します。レポジトリの [Settings (設定)] ページに移動し、[Create notification rule (通知ルールの作成)] を選択することもできます。
4. [通知名] に、ルールの名前を入力します。
5. Amazon EventBridge に提供された情報のみを通知に含める場合は、[Detail type (詳細タイプ)] で [Basic (基本)] を選択します。Amazon EventBridge に提供される情報に加えて、CodeCommit または通知マネージャから提供される場合がある情報も含める場合は、[Full (完全)] を選択します。

   詳細については、「通知の内容とセキュリティについて」を参照してください。
6. [Events that trigger notifications (通知をトリガーするイベント)] で、通知を送信するイベントを選択します。詳細については、「 リポジトリでの通知ルールのイベント」を参照してください。
7. [Targets (ターゲット)] で、次のいずれかの操作を行います。

   - 通知で使用するリソースをすでに設定している場合は、「ターゲットタイプを選択」で、チャットアプリケーション (Slack) で Amazon Q Developer または SNS トピックを選択します。ターゲットの選択で、クライアントの名前 (チャットアプリケーションの Amazon Q Developer で設定された Slack クライアントの場合) または Amazon SNS トピックの Amazon リソースネーム (ARN) (通知に必要なポリシーで既に設定された Amazon SNS トピックの場合) を選択します。
   - 通知で使用するリソースを設定していない場合は、[Create target]、[SNS topic] の順に選択します。codestar-notifications- の後にトピックの名前を指定し、[Create] を選択します。

> **Note**
>
> - 通知ルールの作成の一環として Amazon SNS トピックを作成すると、トピックへのイベント発行を通知機能に許可するポリシーが適用されます。通知ルール用に作成したトピックを使用すると、このリソースに関する通知を受信するユーザーのみをサブスクライブできます。
> - 通知ルールの作成の一環として、チャットアプリケーションクライアントで Amazon Q Developer を作成することはできません。チャットアプリケーション (Slack) で Amazon Q Developer を選択すると、チャットアプリケーションで Amazon Q Developer でクライアントを設定するように指示するボタンが表示されます。このオプションを選択すると、チャットアプリケーションコンソールで Amazon Q Developer が開きます。詳細については、「Configure Integrations Between Notifications and Amazon Q Developer in chat applications」を参照してください。
> - 既存の Amazon SNS トピックをターゲットとして使用する場合は、このトピック用の他のすべてのポリシーに加えて、AWS CodeStar Notifications に必要なポリシーを追加する必要があります。詳細については、「通知用の Amazon SNS トピックを設定する」および「通知の内容とセキュリティについて」を参照してください。

8. ルールの作成を終了するには、[Submit (送信)] を選択します。
9. 通知を受け取るには、そのルールの Amazon SNS トピックにユーザーをサブスクライブする必要があります。詳細については、「ターゲットである Amazon SNS トピックへのユーザーのサブスクライブ」を参照してください。チャットアプリケーションで通知と Amazon Q Developer の統合を設定して、Amazon Chime チャットルームに通知を送信することもできます。詳細については、「Configure Integration Between Notifications and Amazon Q Developer in chat applications」を参照してください。

通知ルールを作成するには (AWS CLI)

1. ターミナルまたはコマンドプロンプトで、create-notification rule コマンドを実行して、JSON スケルトンを生成します。

```
aws codestar-notifications create-notification-rule --generate-cli-skeleton
 > rule.json
```

ファイルには任意の名前を付けることができます。この例では、ファイルの名前を *rule.json* とします。

2. プレーンテキストエディタで JSON ファイルを開き、これを編集してルールに必要なリソース、イベントタイプ、ターゲットを含めます。次の例は、ID *123456789012* の AWS アカウントの *MyDemoRepo* という名前のリポジトリ**MyNotificationRule**に という名前の通知ルールを示しています。ブランチとタグが作成されると、完全な詳細タイプの通知は、*MyNotificationTopic* という Amazon SNS トピックに送信されます。

```
{
    "Name": "MyNotificationRule",
    "EventTypeIds": [
        "codecommit-repository-branches-and-tags-created"
    ],
    "Resource": "arn:aws:codecommit:us-east-1:123456789012:MyDemoRepo",
    "Targets": [
        {
            "TargetType": "SNS",
            "TargetAddress": "arn:aws:sns:us-
east-1:123456789012:MyNotificationTopic"
        }
    ],
    "Status": "ENABLED",
    "DetailType": "FULL"
}
```

ファイルを保存します。

3. 先ほど編集したファイルを使用して、ターミナルまたはコマンドラインで、create-notification-rule コマンドを再度実行し、通知ルールを作成します。

```
aws codestar-notifications create-notification-rule --cli-input-json
 file://rule.json
```

4. 成功すると、コマンドは次のような通知ルールの ARN を返します。

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

通知の変更または無効化

AWS CodeCommit コンソールを使用して、ユーザーに E メールを送信するイベントタイプや、リポジトリに関する E メールを送信するために使用される Amazon SNS トピックなど、2019 年 11 月 5 日より前に作成された通知の設定方法を変更できます。また、CodeCommit コンソールを使用して、トピックにサブスクライブされている E メールアドレスおよびエンドポイントのリストの管理や、通知の無効化を行うこともできます。

通知の設定を変更するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Repositories (リポジトリ)] で、2019 年 11 月 5 日以前に作成された通知を設定するリポジトリの名前を選択します。
3. ナビゲーションペインで、[Settings] を選択し、[Notifications] を選択します。通知ルールではなく通知があることを知らせるバナーが表示された場合は、[Manage existing notifications (既存の通知の管理)] を選択します。
4. [Edit] を選択します。
5. 変更を行ってから、[Save] を選択します。

通知を無効にすると、簡単にユーザーが一時的にリポジトリイベントに関する E メールを受信しないようにできます。

2019 年 11 月 5 日以前に作成された通知を完全に削除するには、「通知の削除」の手順に従います。

通知を無効化するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Repositories (リポジトリ)] で、通知を無効にするリポジトリの名前を選択します。
3. ナビゲーションペインで、[Settings] を選択し、[Notifications] を選択します。[Manage existing notifications (既存の通知の管理)] を選択します。
4. [編集] を選択し、[Event status (イベントのステータス)] で、スライダーを使用して [Enable notifications (通知の有効化)] をオフにします。[保存] を選択します。

5. イベントステータスが [Disabled (無効)] に変わります。イベントに関する E メールは送信されません。通知を無効にすると、リポジトリの CloudWatch Events ルールは自動的に無効になります。CloudWatch Events コンソールで、そのステータスを手動で変更しないでください。

通知の削除

2019 年 11 月 5 日以前にリポジトリ用に作成された通知が不要になった場合は、通知に関連付けられた Amazon CloudWatch Events ルールを削除できます。これにより、通知が自動的に削除されます。通知に使用されるサブスクリプションまたは Amazon SNS トピックは削除されません。

> **Note**
>
> コンソールからリポジトリの名前を変更すると、2019 年 11 月 5 日以前に作成された通知は変更されず引き続き機能します。ただし、コマンドラインまたは API を使用してリポジトリの名前を変更した場合は、通知は機能しなくなります。通知設定を削除して通知を再設定するのが、復元する最も簡単な方法です。

通知設定を削除するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Repositories (リポジトリ)] で、2019 年 11 月 5 日以前に作成された通知を削除するリポジトリの名前を選択します。
3. ナビゲーションペインで、[Settings] を選択し、[Notifications] を選択します。通知ルールではなく通知があることを知らせるバナーが表示された場合は、[Manage existing notifications (既存の通知の管理)] を選択します。
4. [CloudWatch event rule (CloudWatch イベントルール)] で、通知用に作成されたルールの名前をコピーします。
5. にサインイン AWS Management Console し、https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
6. [イベント] の [ルール] を選択します。[名前] に、通知用に作成されたルールの名前を貼り付けます。ルールを選択し、[アクション] の [削除] を選択します。
7. (オプション) 通知設定削除後に通知に使用する Amazon SNS トピックを変更または削除するには、Amazon SNS コンソール (https://console.aws.amazon.com/sns/v3/home) に移動します。詳

細については、『Amazon Simple Notification Service デベロッパーガイド』の「クリーンアップ」を参照してください。

でのリポジトリのタグ付け AWS CodeCommit

タグは、リソース AWS に割り当てるカスタム属性ラベルです AWS 。 AWS タグは、コミットに適用できる Git タグとは異なります。各 AWS タグには 2 つの部分があります。

- タグキー (`CostCenter`、`Environment`、`Project`、`Secret` など)。タグキーでは、大文字と小文字が区別されます。
- タグ値と呼ばれるオプションのフィールド (`111122223333`、`Production`、チーム名など)。タグ値を省略すると、空の文字列を使用した場合と同じになります。タグキーと同様に、タグ値では大文字と小文字が区別されます。

これらは共にキーと値のペアと呼ばれます。リポジトリに保持できるタグの数の制限およびタグキーとタグ値に適用される制限について詳細は、「制限」を参照してください。

タグは、 AWS リソースの識別と整理に役立ちます。多くの AWS サービスはタグ付けをサポートしているため、異なるサービスのリソースに同じタグを割り当てて、リソースが関連していることを示すことができます。例えば、Amazon S3 バケットに割り当てたものと同じタグを CodeCommit リポジトリに割り当てることができます。タグ付け戦略の詳細については、AWS 「リソースのタグ付け」を参照してください。

CodeCommit では、プライマリリソースがリポジトリとなります。CodeCommit コンソール、 AWS CLI、CodeCommit APIs、または AWS SDKs を使用して、リポジトリのタグを追加、管理、削除できます。タグを使用してリポジトリを識別、整理、および追跡するだけでなく、IAM ポリシーのタグを使って、リポジトリを表示および操作できるユーザーを制御することもできます。タグベースのアクセスポリシーの例については、「例 5: タグを使用してリポジトリに対するアクションを許可または拒否する」を参照してください。

トピック

- リポジトリにタグを追加する
- リポジトリのタグの表示
- リポジトリのタグを編集する
- リポジトリからタグを削除する

リポジトリにタグを追加する

リポジトリにタグを追加すると、 AWS リソースを識別して整理し、リソースへのアクセスを管理するのに役立ちます。まず、リポジトリに 1 つ以上のタグ (キーと値のペア) を追加します。リポジトリに保持できるタグの数には制限があります。キーフィールドおよび値フィールドに使用できる文字には制限があります。詳細については、「制限」を参照してください。タグを作成した後、IAM ポリシーを作成して、タグに基づいてリポジトリへのアクセスを管理できます。CodeCommit コンソールまたは を使用して AWS CLI 、リポジトリにタグを追加できます。

> **Important**
>
> リポジトリにタグを追加すると、追加したリポジトリに影響が生じる場合があります。リポジトリにタグを追加する前に、タグを使用してリポジトリなどのリソースへのアクセスを管理する可能性のある IAM ポリシーを必ず確認してください。タグベースのアクセスポリシーの例については、「例 5: タグを使用してリポジトリに対するアクションを許可または拒否する」を参照してください。

リポジトリの作成時にタグを追加する方法について詳細は、「リポジトリを作成する (コンソール)」を参照してください。

トピック

- リポジトリにタグを追加する (コンソール)
- リポジトリにタグを追加する (AWS CLI)

リポジトリにタグを追加する (コンソール)

CodeCommit コンソールを使用して、CodeCommit リポジトリに 1 つ以上のタグを追加できます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Repositories (リポジトリ)] で、タグを追加するリポジトリの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) をクリックします。[Repository tags (レポジトリタグ)] を選択します。
4. リポジトリにタグが追加されていない場合、[Add tag] を選択します。それ以外の場合は、[Edit]、[Add tag] の順に選択します。
5. [Key] に、タグの名前を入力します。[Value] では、任意でタグに値を追加できます。

6. (オプション) 別のタグを追加するには、[Add tag] を再度選択します。
7. タグの追加を完了したら、[Submit] を選択します。

リポジトリにタグを追加する (AWS CLI)

を使用して CodeCommit リポジトリにタグ AWS CLI を追加するには、次の手順に従います。リポジトリを作成するときにタグを追加するには、「リポジトリを作成する (AWS CLI)」を参照してください。

以下のステップでは、 AWS CLI の最新版をすでにインストールしているか、最新版に更新しているものとします。詳細については、「AWS Command Line Interfaceのインストール」を参照してください。

ターミナルまたはコマンドラインで、タグを追加するリポジトリの Amazon リソースネーム (ARN) および追加するタグのキーと値を指定して、tag-resource コマンドを実行します。リポジトリには複数のタグを追加できます。例えば、*MyDemoRepo* という名前のリポジトリに 2 つのタグを付けます。*Secret* のタグ値がある *Status* という名前のタグと、*Saanvi* のタグ値がある *Team* という名前のタグです。

```
aws codecommit tag-resource --resource-arn arn:aws:codecommit:us-
west-2:111111111111:MyDemoRepo --tags Status=Secret,Team=Saanvi
```

成功すると、このコマンドは何も返しません。

リポジトリのタグの表示

タグは、 AWS リソースを識別して整理し、リソースへのアクセスを管理するのに役立ちます。タグ付け戦略の詳細については、AWS 「リソースのタグ付け」を参照してください。タグベースのアクセスポリシーの例については、「例 5: タグを使用してリポジトリに対するアクションを許可または拒否する」を参照してください。

リポジトリのタグを表示する (コンソール)

CodeCommit コンソールを使用して、CodeCommit リポジトリに関連付けられたタグを表示できます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Repositories (リポジトリ)] で、タグを表示するリポジトリの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) をクリックします。[Repository tags (レポジトリタグ)] を選択します。

リポジトリのタグを表示する (AWS CLI)

を使用して CodeCommit リポジトリの AWS タグ AWS CLI を表示するには、次の手順に従います。タグが追加されていない場合、返されるリストは空になります。

ターミナルまたはコマンドラインで、list-tags-for-resource コマンドを実行します。例えば、ARN が *arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo* である *MyDemoRepo* という名前のリポジトリのタグキーとタグ値のリストを表示するには、次のコマンドを実行します。

```
aws codecommit list-tags-for-resource --resource-arn arn:aws:codecommit:us-
west-2:111111111111:MyDemoRepo
```

成功した場合、このコマンドは次のような情報を返します。

```
{
    "tags": {
        "Status": "Secret",
        "Team": "Saanvi"
    }
}
```

リポジトリのタグを編集する

リポジトリに関連付けられたタグの値を変更できます。キーの名前を変更することもできます。これは、現在のタグを削除して、新しい名前と他のタグと同じ値を持つ、別のタグを追加することになります。キーフィールドと値フィールドに使用できる文字には制限があることにご注意ください。詳細については、「制限」を参照してください。

> **Important**
>
> リポジトリのタグを編集すると、追加したリポジトリに影響が生じる場合があります。リポジトリのタグの名前 (キー) または値を編集する前に、タグのキーや値を使用してリポジトリなどのリソースへのアクセスを管理する可能性のある IAM ポリシーを必ず確認してください。タグベースのアクセスポリシーの例については、「例 5: タグを使用してリポジトリに対するアクションを許可または拒否する」を参照してください。

リポジトリのタグを編集する (コンソール)

CodeCommit コンソールを使用して、CodeCommit リポジトリに関連付けられたタグを編集できます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。

2. [Repositories (リポジトリ)] で、タグを編集するリポジトリの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) をクリックします。[Repository tags (レポジトリタグ)] を選択します。
4. [Edit] を選択します。
5.

   次のいずれかを行ってください。

   - タグを変更するには、[Key] に新しい名前を入力します。タグの名前を変更することは、タグを削除して、新しいキー名を持つタグを追加することになります。
   - タグの値を変更するには、新しい値を入力します。値を空にする場合は、現在の値を削除してフィールドを空のままにします。
6. タグの編集を完了したら、[Submit] を選択します。

リポジトリのタグを編集する (AWS CLI)

を使用して CodeCommit リポジトリのタグ AWS CLI を更新するには、次の手順に従います。既存のキーの値を変更したり、別のキーを追加できます。

端末またはコマンドラインで、tag-resource コマンドを実行して、タグを更新するリポジトリの Amazon リソースネーム (ARN) を指定し、タグキーとタグ値を指定します。

```
aws codecommit tag-resource --resource-arn arn:aws:codecommit:us-
west-2:111111111111:MyDemoRepo --tags Team=Li
```

リポジトリからタグを削除する

リポジトリに関連付けられた 1 つ以上のタグを削除できます。タグを削除しても、そのタグに関連付けられている他の AWS リソースからタグは削除されません。

> **Important**
> リポジトリのタグを削除すると、そのリポジトリへのアクセスに影響が生じる場合があります。リポジトリからタグを削除する前に、タグのキーや値を使用してリポジトリなどのリソースへのアクセスを管理する可能性のある IAM ポリシーを必ず確認してください。タグベースのアクセスポリシーの例については、「例 5: タグを使用してリポジトリに対するアクションを許可または拒否する」を参照してください。

リポジトリからタグを削除する (コンソール)

CodeCommit コンソールを使用して、タグと CodeCommit リポジトリとの関連付けを解除できます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Repositories (リポジトリ)] で、タグを削除するリポジトリの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) をクリックします。[Repository tags (レポジトリタグ)] を選択します。
4. [Edit] を選択します。
5. 削除するタグを見つけ、[Remove tag] を選択します。
6. タグの削除を完了したら、[Submit] を選択します。

リポジトリからタグを削除する (AWS CLI)

を使用して CodeCommit リポジトリからタグ AWS CLI を削除するには、次の手順に従います。タグを解除してもこれが削除されるわけではありません。タグとリポジトリ間の関連付けを解除するのみです。

> **Note**
>
> CodeCommit リポジトリを削除すると、関連付けられたすべてのタグが、削除されたリポジトリから解除されます。リポジトリを削除する前にタグを削除する必要はありません。

ターミナルまたはコマンド行で、タグを解除するリポジトリの ARN と解除するタグのタグキーを指定して、untag-resource コマンドを実行します。例えば、*Status* のタグキーがある *MyDemoRepo* という名前のタグを解除するには、次を行います。

```
aws codecommit untag-resource --resource-arn arn:aws:codecommit:us-
west-2:111111111111:MyDemoRepo --tag-keys Status
```

成功すると、このコマンドは何も返しません。リポジトリに関連付けられているタグを確認するには、list-tags-for-resource コマンドを実行します。

AWS CodeCommit リポジトリのトリガーを管理する

コードプッシュや他のイベントが Amazon Simple Notification Service (Amazon SNS) からの通知の送信や AWS Lambdaの関数の呼び出しなどのアクションをトリガーするように、CodeCommit リポジトリを設定できます。CodeCommit リポジトリごとに、最大 10 のトリガーを作成することができます。

トリガーは一般的に次のように設定されます。

- リポジトリにプッシュする度に登録ユーザーに E メールを送信する。
- リポジトリのメインブランチにプッシュされたらビルドを開始するように外部のビルドシステムに通知する。

外部のビルドシステムのようなシナリオの場合は、他のアプリケーションと相互作用するように Lambda 関数を記述する必要があります。この E メールのシナリオで必要なことは、Amazon SNS トピックを作成することだけです。

このトピックでは、CodeCommit で Amazon SNS および Lambda のアクションをトリガーできるようにするアクセス許可の設定方法について説明します。また、トリガーの作成、編集、テスト、削除の例を示したリンクも紹介します。

トピック

- リソースの作成と CodeCommit のアクセス許可の追加
- 例: Amazon SNS トピックの AWS CodeCommit トリガーを作成する
- 例: AWS Lambda 関数の AWS CodeCommit トリガーを作成する
- 例: 既存の AWS Lambda 関数 AWS CodeCommit のトリガーを で作成する
- AWS CodeCommit リポジトリのトリガーを編集する
- AWS CodeCommit リポジトリのテストトリガー
- AWS CodeCommit リポジトリからトリガーを削除する

リソースの作成と CodeCommit のアクセス許可の追加

Amazon SNS トピックと Lambda 関数は、CodeCommit のトリガーと統合できますが、まず、リソースを作成し、CodeCommit にアクセス許可を付与してこのリソースと通信できるようにするポリシーを設定する必要があります。CodeCommit リポジトリ AWS リージョン と同じ にリソースを作成する必要があります。例えば、リポジトリが米国東部 (オハイオ) (us-east-2) にある場合、Amazon SNS トピックまたは Lambda 関数は、米国東部 (オハイオ) にあることが必要です。

- Amazon SNS トピックの場合、Amazon SNS トピックが CodeCommit リポジトリと同じアカウントを使用して作成されている場合は、IAM ポリシーおよびアクセス許可を追加設定する必要はありません。CodeCommit トリガーは、Amazon SNS トピックを作成し、サブスクライブした時点で作成できます。
  - Amazon SNS のトピック作成の詳細については、「Amazon SNS のドキュメント」を参照してください。
  - Amazon SNS を使用して Amazon SQS キューにメッセージを送信する方法の詳細については、『Amazon SNS デベロッパーガイド』の「Amazon SQS キューへのメッセージの送信」を参照してください。
  - Amazon SNS を使用して Lambda 関数を呼び出す方法の詳細については、『Amazon SNS デベロッパーガイド』の「Lambda 関数の呼び出し」を参照してください。
- ただし、トリガーを設定して、別の AWS アカウントで Amazon SNS トピックを使用する場合は、まず CodeCommit でそのトピックにパブリッシュできるようにするポリシーを使用してトピックを設定する必要があります。詳細については、「例 1: Amazon SNS トピックへのクロスアカウントアクセスを有効にするポリシーを作成する」を参照してください。
- Lambda 関数を設定するには、この関数の一部として Lambda コンソールでトリガーを作成します。Lambda コンソールで作成されるトリガーには、CodeCommit で Lambda 関数を呼び出すために必要なアクセス許可が元々割り当てられているため、この方法が最も簡単です。CodeCommit

にトリガーを作成する場合は、CodeCommit でこの関数を呼び出せるようにするポリシーを含む必要があります。詳細については、「既存の Lambda 関数のトリガーを作成する」および「例 3: CodeCommit トリガーと AWS Lambda 統合するためのポリシーを作成する」を参照してください。

例: Amazon SNS トピックの AWS CodeCommit トリガーを作成する

CodeCommit リポジトリのトリガーを作成すると、そのリポジトリ内のイベントに応じて Amazon Simple Notification Service (Amazon SNS) トピックから通知をトリガーできます。リポジトリイベント (ブランチの削除など) に関する通知をユーザーが受信できるようにするには、Amazon SNS トピックへのトリガーを作成することをお勧めします。Amazon SNS トピックと Amazon Simple Queue Service (Amazon SQS) や などの他の サービスとの統合を活用することもできます AWS Lambda。

> **Note**
>
> - このトリガーは、リポジトリイベントに応答して実行されるアクションとなる既存の Amazon SNS トピックにポイントする必要があります。Amazon SNS トピックの作成とサブスクライブの詳細については、Amazon Simple Notification Service 入門ガイドを参照してください。
> - Amazon SNS FIFO (先入れ先出し) トピックは、CodeCommit トリガー向けにサポートされていません。

トピック

- CodeCommit リポジトリで Amazon SNS トピックへのトリガーを作成する (コンソール)
- CodeCommit リポジトリで Amazon SNS トピックへのトリガーを作成する (AWS CLI)

CodeCommit リポジトリで Amazon SNS トピックへのトリガーを作成する (コンソール)

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリで、リポジトリイベントのトリガーを作成するリポジトリを選択します。
3. リポジトリのナビゲーションペインで、[設定] を選択し、[トリガー] を選択します。

4. [トリガーの作成] を選択してから、次の操作を行います。

   - [Trigger name (トリガー名)] に、トリガーの名前 (例: *MyFirstTrigger*) を入力します。
   - [イベント] で、Amazon SNS トピックをトリガーして通知を送信するリポジトリイベントを選択します。

     [すべてのリポジトリイベント] を選択すると、他のイベントは選択できません。イベントのサブセットを選択するには、[すべてのリポジトリイベント] の選択を解除し、リストから 1 つ以上のイベントを選択します。例えば、ユーザーが CodeCommit リポジトリにタグやブランチを作成した場合にのみトリガーを実行するには、[すべてのリポジトリイベント] の選択を解除し、次に [ブランチまたはタグを作成する] を選択します。
   - トリガーをリポジトリのすべてのブランチに適用する場合は、[ブランチ] で選択を空白のままにします。このデフォルトのオプションでは、トリガーがすべてのブランチに自動的に適用されます。このトリガーを特定のブランチにのみ適用する場合は、リポジトリブランチのリストから最大 10 個のブランチ名を選択します。
   - [Choose the service to use (使用するサービスを選択する)] で、[Amazon SNS] を選択します。
   - [Amazon SNS] で、一覧からトピック名を選択するか、トピックの ARN を入力します。

     > **Note**
     >
     > Amazon SNS FIFO (先入れ先出し) トピックは、CodeCommit トリガー向けにサポートされていません。タイプが Standard (標準) に設定されている Amazon SNS トピックを選択する必要があります。
   - [カスタムデータ] では、Amazon SNS トピックから送信される通知に含めるオプションの情報 (開発者がこのリポジトリで開発について検討するときに使用する IRC チャネル名など) を指定します。このフィールドは文字列です。これは動的パラメータを渡すために使用することはできません。

5. (オプション) [トリガーのテスト] を選択します。このステップは、CodeCommit と Amazon SNS トピック間のアクセスが正しく設定されていることを確認するのに役立ちます。このステップでは、Amazon SNS トピックでリポジトリのデータを使用してテスト通知を送信します。実際のデータがない場合は、テスト通知サンプルデータが使用されます。
6. [Create trigger (トリガーを作成する)] を選択して、トリガーの作成を完了します。

CodeCommit リポジトリで Amazon SNS トピックへのトリガーを作成する (AWS CLI)

コマンドラインを使用して、CodeCommit リポジトリイベント (リポジトリにコミットをプッシュする場合など) に応答する Amazon SNS トピックのトリガーを作成することもできます。

Amazon SNS トピック用のトリガーを作成するには

1. プレーンテキストエディタを開き、次を指定する JSON ファイルを作成します。

   - Amazon SNS トピック名。

     > **Note**
     >
     > Amazon SNS FIFO (先入れ先出し) トピックは、CodeCommit トリガー向けにサポートされていません。タイプが Standard (標準) に設定されている Amazon SNS トピックを選択する必要があります。

   - このトリガーで監視するリポジトリとブランチ。(ブランチを指定しない場合、トリガーはリポジトリのすべてのブランチに適用されます。)
   - このトリガーをアクティブ化するイベントです。

   ファイルを保存します。

   例えば、*main* と *preprod* という 2 つのブランチについての *MySNSTopic* という Amazon SNS トピックに対してすべてのリポジトリイベントを発行する *MyDemoRepo* という名前のリポジトリのトリガーを作成するには、次のようにします。

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "name": "MyFirstTrigger",
            "destinationArn": "arn:aws:sns:us-east-2:111122223333:MySNSTopic",
            "customData": "",
            "branches": [
                "main", "preprod"
            ],
            "events": [
                "all"
```

```
            ]
        }
    ]
}
```

リポジトリのトリガーごとに、JSON にトリガーブロックが必要です。リポジトリに対して複数のトリガーを作成するには、JSON に複数のトリガーブロックを含めます。このファイルに作成されたすべてのトリガーは指定されたリポジトリであることに注意してください。1 つの JSON ファイルに複数のリポジトリのトリガーを作成することはできません。たとえば、リポジトリに対して 2 つのトリガーを作成する場合は、2 つのトリガーブロックを持つ JSON ファイルを作成できます。次の例では、2 番目のトリガーにブランチが指定されていないため、トリガーはすべてのブランチに適用されます。

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "name": "MyFirstTrigger",
            "destinationArn": "arn:aws:sns:us-east-2:111122223333:MySNSTopic",
            "customData": "",
            "branches": [
                "main", "preprod"
            ],
            "events": [
                "all"
            ]
        },
        {
            "name": "MySecondTrigger",
            "destinationArn": "arn:aws:sns:us-east-2:111122223333:MySNSTopic2",
            "customData": "",
            "branches": [],
            "events": [
                "updateReference", "deleteReference"
            ]
        }
    ]
}
```

たとえば、コミットがリポジトリにプッシュされたときは、指定したイベントのトリガーを作成できます。イベントタイプは次のとおりです。

- `all`: 指定されたリポジトリとブランチのすべてのイベント。
- `updateReference` : 指定されたリポジトリやブランチにコミットをプッシュされた場合。
- `createReference`: 指定されたリポジトリに新しいブランチまたはタグが作成された場合。
- `deleteReference`: 指定されたリポジトリ内のブランチまたはタグが削除された場合。

> **Note**
> トリガー内で複数のイベントタイプを使用できます。ただし、`all` を指定すると、他のイベントを指定することはできません。

有効なイベントタイプのリストを一覧表示するには、ターミナルまたはコマンドプロンプトで、「aws codecommit put-repository-triggers help」と入力します。

また、`customData` に文字列を含めることもできます (たとえば、開発者がこのリポジトリで開発について検討するときに使用する IRC チャネル名)。このフィールドは文字列です。これは動的パラメータを渡すために使用することはできません。この文字列は、トリガーに応答して戻される CodeCommit JSON に属性として追加されます。

2. (オプション) ターミナルまたはコマンドラインプロンプトで、test-repository-triggers コマンドを実行することもできます。このテストでは、リポジトリからのサンプルデータを使用して (データがない場合はサンプルデータを生成して) Amazon SNS トピックの受信者に通知を送信します。例えば、次のコードでは、トリガーファイル (*`trigger.json`*) 内の JSON が有効であり、CodeCommit が Amazon SNS トピックに発行できることをテストします。

```
aws codecommit test-repository-triggers --cli-input-json file://trigger.json
```

成功した場合、このコマンドは次のような情報を返します。

```
{
    "successfulExecutions": [
        "MyFirstTrigger"
    ],
    "failedExecutions": []
```

```
}
```

3. ターミナルまたはコマンドプロンプトで、put-repository-triggers コマンドを実行して CodeCommit にトリガーを作成します。たとえば、*trigger.json* という JSON ファイルを使用してトリガーを作成するには、次のようにします。

   ```
   aws codecommit put-repository-triggers --cli-input-json
   file://trigger.json
   ```

   このコマンドでは、次のような設定 ID が返されます。

   ```
   {
       "configurationId": "0123456-I-AM-AN-EXAMPLE"
   }
   ```

4. トリガーの設定を表示するには、get-repository-triggers コマンドを実行して、リポジトリの名前を指定します。

   ```
   aws codecommit get-repository-triggers --repository-name MyDemoRepo
   ```

   このコマンドは、リポジトリ用に構成されたすべてのトリガーの構造を、次のように戻します。

   ```
   {
       "configurationId": "0123456-I-AM-AN-EXAMPLE",
       "triggers": [
           {
               "events": [
                   "all"
               ],
               "destinationArn": "arn:aws:sns:us-east-2:111122223333:MySNSTopic",
               "branches": [
                   "main",
                   "preprod"
               ],
               "name": "MyFirstTrigger",
               "customData": "Project ID 12345"
           }
       ]
   }
   ```

5. トリガー自体の機能をテストするには、トリガーを設定したリポジトリにコミットを作成してプッシュします。Amazon SNS トピックからの応答が表示されます。例えば、E メールを送信

するように Amazon SNS トピックを設定すると、このトピックに登録されている E メールアカウントに Amazon SNS からの E メールが表示されます。

CodeCommit リポジトリへのプッシュに応答して Amazon SNS から送信された E メールの出力例を次に示します。

```
{
  "Records":[
     {
        "awsRegion":"us-east-2",
        "codecommit":{
           "references" : [
              {
                    "commit":"317f8570EXAMPLE",
                    "created":true,
                    "ref":"refs/heads/NewBranch"
              },
              {
                    "commit":"4c925148EXAMPLE",
                    "ref":"refs/heads/preprod",
              }
            ]
          },
        "eventId":"11111-EXAMPLE-ID",
        "eventName":"ReferenceChange",
        "eventPartNumber":1,
        "eventSource":"aws:codecommit",
        "eventSourceARN":"arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
        "eventTime":"2016-02-09T00:08:11.743+0000",
        "eventTotalParts":1,
        "eventTriggerConfigId":"0123456-I-AM-AN-EXAMPLE",
        "eventTriggerName":"MyFirstTrigger",
        "eventVersion":"1.0",
        "customData":"Project ID 12345",
        "userIdentityARN":"arn:aws:iam::111122223333:user/JaneDoe-CodeCommit",
     }
  ]
}
```

例: AWS Lambda 関数の AWS CodeCommit トリガーを作成する

CodeCommit リポジトリのトリガーを作成して、リポジトリのイベントから Lambda 関数を呼び出すことができます。この例では、リポジトリのクローンを作成するために使用する URL を Amazon CloudWatch Logs に返す Lambda 関数を作成します。

トピック

- Lambda 関数を作成する
- AWS CodeCommit リポジトリの Lambda 関数用のトリガーを表示する

Lambda 関数を作成する

Lambda コンソールを使用してこの関数を作成する場合は、Lambda 関数の CodeCommit トリガーを作成することもできます。以下の手順には、サンプル Lambda 関数が含まれます。このサンプルは、JavaScript と Python の 2 つの言語で使用できます。この関数は、リポジトリのクローンを CloudWatch Logs に作成するために使用する URL を返します。

Lambda の設計図を使用して Lambda 関数を作成します。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/lambda/ で AWS Lambda コンソールを開きます。
2. [Lambda Functions] (Lambda 関数) ページで、[Create function] (関数の作成) を選択します。(これまでに Lambda を使用したことがない場合は、[今すぐ始める] を選択します。)
3. [関数の作成] ページで、[一から作成] を選択します。[関数名] に、関数の名前 (*MyLambdaFunctionforCodeCommit* など) を入力します。[ランタイム] で、関数の記述に使用する言語を選択し、[関数の作成] を選択します。
4. [設定] タブで、[トリガーの追加] を選択します。
5. [トリガー設定] で、サービスのドロップダウンリストから [CodeCommit] を選択します。

- [リポジトリ名] で、リポジトリイベントに応答して Lambda 関数を使用するトリガーを設定するリポジトリの名前を選択します。

- [Trigger name (トリガー名)] に、トリガーの名前 (例: *MyLambdaFunctionTrigger*) を入力します。
- [Events (イベント)] で、Lambda 関数をトリガーするリポジトリイベントを選択します。[すべてのリポジトリイベント] を選択すると、他のイベントは選択できません。イベントのサブセットを選択する場合は、[すべてのリポジトリイベント] の選択を解除してから、使用するイベントをリストから選択します。たとえば、ユーザーが AWS CodeCommit リポジトリにタグまたはブランチを作成したときにのみトリガーを実行する場合は、すべてのリポジトリイベントを削除し、ブランチまたはタグの作成を選択します。
- リポジトリのすべてのブランチにトリガーを適用するには、[ブランチ] で、[すべてのブランチ] を選択します。それ以外の場合は、[特定のブランチ] を選択します。リポジトリのデフォルトブランチがデフォルトで追加されます。リストからこのブランチを保持または削除することができます。リポジトリブランチのリストから最大 10 のブランチ名を選択します。
- (オプション) [Custom data (カスタムデータ)] に、Lambda 関数に含める情報 (例: 開発者がリポジトリでの開発に関するやり取りに使用する IRC チャネルの名前) を入力します。このフィールドは文字列です。これは動的パラメータを渡すために使用することはできません。

[Add] (追加) を選択します。

6. [設定] ページの [関数コード] で、コードエントリタイプとして [コードをインラインで編集] を選択します。[ランタイム] で、[Node.js] を選択します。サンプルの Python 関数を作成する場合は、[Python] を選択します。
7. [コードエントリタイプ] で、[コードをインラインで編集] を選択した後、hello world コードを以下の 2 つのサンプルのいずれかに置き換えます。

   Node.js の場合:

```
import {
  CodeCommitClient,
  GetRepositoryCommand,
} from "@aws-sdk/client-codecommit";

const codecommit = new CodeCommitClient({ region: "your-region" });

/**
 * @param {{ Records: { codecommit: { references: { ref: string }[] },
 eventSourceARN: string  }[]} event
 */
export const handler = async (event) => {
```

```
  // Log the updated references from the event
  const references = event.Records[0].codecommit.references.map(
    (reference) => reference.ref,
  );
  console.log("References:", references);

  // Get the repository from the event and show its git clone URL
  const repository = event.Records[0].eventSourceARN.split(":")[5];
  const params = {
    repositoryName: repository,
  };

  try {
    const data = await codecommit.send(new GetRepositoryCommand(params));
    console.log("Clone URL:", data.repositoryMetadata.cloneUrlHttp);
    return data.repositoryMetadata.cloneUrlHttp;
  } catch (error) {
    console.error("Error:", error);
    throw new Error(
      `Error getting repository metadata for repository ${repository}`,
    );
  }
};
```

Python の場合:

```
import json
import boto3

codecommit = boto3.client("codecommit")


def lambda_handler(event, context):
    # Log the updated references from the event
    references = {
        reference["ref"]
        for reference in event["Records"][0]["codecommit"]["references"]
    }
    print("References: " + str(references))

    # Get the repository from the event and show its git clone URL
    repository = event["Records"][0]["eventSourceARN"].split(":")[5]
```

```
    try:
        response = codecommit.get_repository(repositoryName=repository)
        print("Clone URL: " + response["repositoryMetadata"]["cloneUrlHttp"])
        return response["repositoryMetadata"]["cloneUrlHttp"]
    except Exception as e:
        print(e)
        print(
            "Error getting repository {}. Make sure it exists and that your
repository is in the same region as this function.".format(
                repository
            )
        )
        raise e
```

8. [Permissions] (アクセス許可) タブの [Execution role] (実行ロール) で、ロールを選択して IAM コンソールで開きます。アタッチされたポリシーを編集して、トリガーを使用するリポジトリの `GetRepository` アクセス許可を追加します。

AWS CodeCommit リポジトリの Lambda 関数用のトリガーを表示する

Lambda 関数を作成したら、 AWS CodeCommitでトリガーを表示してテストできます。トリガーをテストすると、指定したリポジトリイベントに応答して関数が実行されます。

Lambda 関数のトリガーを表示およびテストするには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] で、トリガーを表示するリポジトリを選択します。
3. リポジトリのナビゲーションペインで、[設定] を選択し、[トリガー] を選択します。
4. リポジトリのトリガーのリストを確認します。Lambda コンソールで作成したトリガーが表示されています。リストからそのトリガーを選択し、[Test trigger (トリガーのテスト)] を選択します。このオプションでは、リポジトリの最新コミット ID など、リポジトリに関するサンプルデータを使用して、関数を呼び出します。(コミット履歴が存在しない場合は、ゼロで構成されるサンプル値が生成されます。) これにより、 AWS CodeCommit と Lambda 関数間のアクセスが正しく設定されていることを確認できます。
5. トリガーの機能をさらに検証するには、コミットを作成し、トリガーを設定したリポジトリにプッシュします。Lambda コンソールで、Lambda 関数からの応答がその関数の [モニタリ

ング] タブに表示されます。[モニタリング] タブから、[CloudWatch のログを表示] を選択します。CloudWatch コンソールが新しいタブで開き、関数のイベントが表示されます。リストから、コミットをプッシュした時間に対応するログストリームを選択します。以下のようなイベントデータが表示されます。

```
START RequestId: 70afdc9a-EXAMPLE Version: $LATEST
2015-11-10T18:18:28.689Z 70afdc9a-EXAMPLE References: [ 'refs/heads/main' ]
2015-11-10T18:18:29.814Z 70afdc9a-EXAMPLE Clone URL: https://git-codecommit.us-
east-2.amazonaws.com/v1/repos/MyDemoRepo
END RequestId: 70afdc9a-EXAMPLE
REPORT RequestId: 70afdc9a-EXAMPLE Duration: 1126.87 ms Billed Duration: 1200 ms
 Memory Size: 128 MB Max Memory Used: 14 MB
```

例: 既存の AWS Lambda 関数 AWS CodeCommit のトリガーを で作成する

Lambda 関数を呼び出すトリガーを作成する最も簡単な方法は、Lambda コンソールでトリガーを作成することです。この組み込みの統合を使用すると、この関数を実行するために必要なアクセス許可を CodeCommit に付与することができます。既存の Lambda 関数のトリガーを追加するには、Lambda コンソールに移動し、この関数を選択します。関数の [トリガー] タブで、[Add trigger (トリガーの追加)] のステップに従います。これらのステップは、「Lambda 関数を作成する」に示されている手順と似ています。

また、Lambda 関数のトリガーを CodeCommit リポジトリに作成することもできます。そのためには、既存の Lambda 関数を選択して呼び出す必要があります。また、関数を実行するために CodeCommit に必要なアクセス許可を手動設定する必要があります。

トピック

- Lambda 関数の実行を CodeCommit に許可するためのアクセス許可を手動で設定する
- Lambda 関数のトリガーを CodeCommit リポジトリに作成する (コンソール)
- Lambda 関数のトリガーを CodeCommit リポジトリに作成する (AWS CLI)

Lambda 関数の実行を CodeCommit に許可するためのアクセス許可を手動で設定する

CodeCommit に Lambda 関数を呼び出すトリガーを作成する場合は、CodeCommit で Lambda 関数を実行できるようにするアクセス許可を手動で設定する必要があります。この手動設定を避けるには、代わりに Lambda コンソールで関数のトリガーを作成することを検討してください。

CodeCommit でLambda 関数を実行できるようにするには

1. プレーンテキストエディタを開いて、Lambda 関数名、CodeCommit リポジトリの詳細、Lambda で実行するアクションを指定した JSON ファイルを作成します。次のようになります。

```
{
    "FunctionName": "MyCodeCommitFunction",
    "StatementId": "1",
    "Action": "lambda:InvokeFunction",
    "Principal": "codecommit.amazonaws.com",
    "SourceArn": "arn:aws:codecommit:us-east-1:111122223333:MyDemoRepo",
    "SourceAccount": "111122223333"
}
```

2. 覚えやすい名前 (例: *AllowAccessfromMyDemoRepo*.json) を付けて、JSON ファイルを保存します。

3. 前の手順で作成した JSON ファイルを使用して、ターミナル (Linux、macOS、Unix) またはコマンドライン (Windows) で、aws lambda add-permissions コマンドを実行して、Lambda 関数に関連付けられたリソースポリシーに対するアクセス許可を追加します。

```
aws lambda add-permission --cli-input-json file://AllowAccessfromMyDemoRepo.json
```

このコマンドでは、先ほど追加したポリシーステートメントの JSON が返ります。次のようになります。

```
{
    "Statement": "{\"Condition\":{\"StringEquals\":{\"AWS:SourceAccount
\":\"111122223333\"},\"ArnLike\":{\"AWS:SourceArn\":
\"arn:aws:codecommit:us-east-1:111122223333:MyDemoRepo\"}},\"Action
\":[\"lambda:InvokeFunction\"],\"Resource\":\"arn:aws:lambda:us-
east-1:111122223333:function:MyCodeCommitFunction\",\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"codecommit.amazonaws.com\"},\"Sid\":\"1\"}"
}
```

Lambda 関数のリソースポリシーの詳細については、『AWS Lambda ユーザーガイド』の「AddPermission」および「プル/プッシュイベントモデル」を参照してください。

4. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。

5. [ダッシュボード] ナビゲーションペインで、[ロール] を選択し、ロールのリストで、[*lambda_basic_execution*] を選択します。
6. ロールの [Summary] ページで、[Permissions (アクセス許可)] タブを選択し、[Inline Policies (インラインポリシー)] セクションで、[Create Role Policy (ロールポリシーの作成)] を選択します。
7. [Set Permissions (アクセス許可の設定)] ページで、[Custom Generator (カスタムジェネレーター)] を選択し、次に [選択] をクリックします。
8. [Edit Permissions] ページで、以下を実行します。

   - [Effect] で、[Allow] を選択します。
   - [AWS のサービス] で、[AWS CodeCommit] を選択します。
   - [アクション] で、GetRepository を選択します。
   - [Amazon リソースネーム (ARN)] に、リポジトリの ARN (例: `arn:aws:codecommit:us-east-1:`*`111122223333`*`:`*`MyDemoRepo`*) を入力します。

   [Add Statement]、[Next Step] の順に選択します。
9. [Review Policy] ページで、[Apply Policy] を選択します。

   ポリシーステートメントは、次の例のようになります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt11111111",
            "Effect": "Allow",
            "Action": [
                "codecommit:GetRepository"
            ],
            "Resource": [
                "arn:aws:codecommit:us-east-1:111122223333:MyDemoRepo"
            ]
        }
    ]
}
```

Lambda 関数のトリガーを CodeCommit リポジトリに作成する (コンソール)

Lambda 関数を作成したら、指定したリポジトリのイベントに対してこの関数を実行できるようにするトリガーを CodeCommit に作成することができます。

> **Note**
>
> トリガーをテストまたは実行する前に、CodeCommit で Lambda 関数を呼び出して、この関数でリポジトリに関する情報を取得できるようにするポリシーを設定する必要があります。詳細については、「CodeCommit でLambda 関数を実行できるようにするには」を参照してください。

Lambda 関数のトリガーを作成する

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリで、リポジトリイベントのトリガーを作成するリポジトリを選択します。
3. リポジトリのナビゲーションペインで、[設定] を選択し、[トリガー] を選択します。
4. [Create trigger (トリガーの作成)] を選択します。
5. [Create trigger (トリガーの作成)] で、以下の操作を行います。

   - [Trigger name (トリガー名)] に、トリガーの名前 (例: `MyLambdaFunctionTrigger`) を入力します。
   - [Events (イベント)] で、Lambda 関数をトリガーするリポジトリイベントを選択します。

     [すべてのリポジトリイベント] を選択すると、他のイベントは選択できません。イベントのサブセットを選択する場合は、[すべてのリポジトリイベント] の選択を解除してから、使用するイベントをリストから選択します。例えば、ユーザーが CodeCommit リポジトリにタグやブランチを作成した場合にのみトリガーを実行するには、[すべてのリポジトリイベント] の選択を解除し、次に [ブランチまたはタグを作成する] を選択します。
   - トリガーをリポジトリのすべてのブランチに適用する場合は、[ブランチ] で選択を空白のままにします。このデフォルトのオプションでは、トリガーがすべてのブランチに自動的に適用されます。このトリガーを特定のブランチにのみ適用する場合は、リポジトリブランチのリストから最大 10 個のブランチ名を選択します。

- [Choose the service to use (使用するサービスを選択する)] で、[AWS Lambda] を選択します。
- [Lambda 関数] で、リストから関数名を選択するか、関数の ARN を入力します。
- (オプション) [Custom data (カスタムデータ)] に、Lambda 関数に含める情報 (例: 開発者がリポジトリでの開発に関するやり取りに使用する IRC チャネルの名前) を入力します。このフィールドは文字列です。これは動的パラメータを渡すために使用することはできません。

6. (オプション) [トリガーのテスト] を選択します。このオプションでは、リポジトリの最新コミット ID など、リポジトリに関するサンプルデータを使用して、関数を呼び出します。(コミット履歴が存在しない場合は、ゼロで構成されるサンプル値が生成されます。) これにより、CodeCommit と Lambda 関数の間のアクセスが正しく設定されていることを確認しやすくなります。
7. [Create trigger (トリガーを作成する)] を選択して、トリガーの作成を完了します。
8. トリガーの機能を検証するには、コミットを作成し、トリガーを設定したリポジトリに対してプッシュします。Lambda コンソールで、Lambda 関数からの応答がその関数の [モニタリング] タブに表示されます。

Lambda 関数のトリガーを CodeCommit リポジトリに作成する (AWS CLI)

コマンドラインを使用して、CodeCommit リポジトリイベントに応答する Lambda 関数のトリガーを作成することもできます (リポジトリにコミットをプッシュする場合など)。

Lambda 関数のトリガーを作成するには

1. プレーンテキストエディタを開き、次を指定する JSON ファイルを作成します。

   - Lambda 関数の名前。
   - このトリガーで監視するリポジトリとブランチ。(ブランチを指定しない場合、トリガーはリポジトリのすべてのブランチに適用されます。)
   - このトリガーをアクティブ化するイベントです。

   ファイルを保存します。

   例えば、*MyDemoRepo* という名前のリポジトリのトリガーを作成して、*main* および *preprod* という 2 つのブランチについての *MyCodeCommitFunction* という名前の Lambda 関数にすべてのリポジトリイベントを発行するには、次のようにします。

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "name": "MyLambdaFunctionTrigger",
            "destinationArn": "arn:aws:lambda:us-
east-1:111122223333:function:MyCodeCommitFunction",
            "customData": "",
            "branches": [
                "main", "preprod"
            ],
            "events": [
                "all"
            ]
        }
    ]
}
```

リポジトリのトリガーごとに、JSON にトリガーブロックが必要です。リポジトリのトリガーを複数作成するには、JSON にブロックを追加します。このファイルに作成されたすべてのトリガーは指定されたリポジトリであることに注意してください。1 つの JSON ファイルに複数のリポジトリのトリガーを作成することはできません。たとえば、リポジトリに対して 2 つのトリガーを作成する場合は、2 つのトリガーブロックを持つ JSON ファイルを作成できます。次の例では、2 番目のトリガーブロックに指定されているブランチはないため、トリガーはすべてのブランチに適用されます。

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "name": "MyLambdaFunctionTrigger",
            "destinationArn": "arn:aws:lambda:us-
east-1:111122223333:function:MyCodeCommitFunction",
            "customData": "",
            "branches": [
                "main", "preprod"
            ],
            "events": [
                "all"
            ]
```

```
        },
        {
            "name": "MyOtherLambdaFunctionTrigger",
            "destinationArn": "arn:aws:lambda:us-
east-1:111122223333:function:MyOtherCodeCommitFunction",
            "customData": "",
            "branches": [],
            "events": [
                "updateReference", "deleteReference"
            ]
        }
    ]
}
```

たとえば、コミットがリポジトリにプッシュされたときは、指定したイベントのトリガーを作成できます。イベントタイプは次のとおりです。

- `all`: 指定されたリポジトリとブランチのすべてのイベント。
- `updateReference` : 指定されたリポジトリやブランチにコミットをプッシュされた場合。
- `createReference`: 指定されたリポジトリに新しいブランチまたはタグが作成された場合。
- `deleteReference`: 指定されたリポジトリ内のブランチまたはタグが削除された場合。

> **Note**
>
> トリガー内で複数のイベントタイプを使用できます。ただし、`all` を指定すると、他のイベントを指定することはできません。

有効なイベントタイプのリストを一覧表示するには、ターミナルまたはコマンドプロンプトで、「aws codecommit put-repository-triggers help」と入力します。

また、`customData` に文字列を含めることもできます (たとえば、開発者がこのリポジトリで開発について検討するときに使用する IRC チャネル名)。このフィールドは文字列です。これは動的パラメータを渡すために使用することはできません。この文字列は、トリガーに応答して戻される CodeCommit JSON に属性として追加されます。

2. (オプション) ターミナルまたはコマンドラインプロンプトで、test-repository-triggers コマンドを実行することもできます。例えば、以下は、JSON ファイル (*`trigger.json`*) が有効であることと、CodeCommit で Lambda 関数をトリガーできることを検証するために使用されます。

利用できる実際のデータがない場合、このテストでは、サンプルデータを使用して関数をトリガーします。

```
aws codecommit test-repository-triggers --cli-input-json file://trigger.json
```

成功した場合、このコマンドは次のような情報を返します。

```
{
    "successfulExecutions": [
        "MyLambdaFunctionTrigger"
    ],
    "failedExecutions": []
}
```

3. ターミナルまたはコマンドプロンプトで、put-repository-triggers コマンドを実行して CodeCommit にトリガーを作成します。たとえば、*trigger.json* という JSON ファイルを使用してトリガーを作成するには、次のようにします。

   ```
   aws codecommit put-repository-triggers --cli-input-json
   file://trigger.json
   ```

   このコマンドでは、次のような設定 ID が返されます。

   ```
   {
       "configurationId": "0123456-I-AM-AN-EXAMPLE"
   }
   ```

4. トリガーの設定を表示するには、get-repository-triggers コマンドを実行して、リポジトリの名前を指定します。

   ```
   aws codecommit get-repository-triggers --repository-name MyDemoRepo
   ```

   このコマンドは、リポジトリ用に構成されたすべてのトリガーの構造を、次のように戻します。

   ```
   {
       "configurationId": "0123456-I-AM-AN-EXAMPLE",
       "triggers": [
           {
               "events": [
                   "all"
               ],
   ```

```
            "destinationArn": "arn:aws:lambda:us-
east-1:111122223333:MyCodeCommitFunction",
            "branches": [
                "main",
                "preprod"
            ],
            "name": "MyLambdaFunctionTrigger",
            "customData": "Project ID 12345"
        }
    ]
}
```

5. トリガーの機能を確認するには、コミットを作成し、トリガーを設定したリポジトリに対してプッシュします。Lambda コンソールで、Lambda 関数からの応答がその関数の [モニタリング] タブに表示されます。

AWS CodeCommit リポジトリのトリガーを編集する

CodeCommit リポジトリ用に作成したトリガーを編集できます。トリガーのイベントとブランチ、イベントに応答したアクション、その他の設定を変更できます。

トピック

- リポジトリのトリガーを編集する (コンソール)
- リポジトリのトリガーを編集する (AWS CLI)

リポジトリのトリガーを編集する (コンソール)

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリで、リポジトリイベントのトリガーを編集するリポジトリを選択します。
3. リポジトリのナビゲーションペインで、[設定] を選択し、[トリガー] を選択します。
4. リポジトリのトリガーのリストから、編集するトリガーを選択し、[編集] を選択します。
5. トリガーに変更を加え、[保存] を選択します。

リポジトリのトリガーを編集する (AWS CLI)

1. ターミナル (Linux、macOS、Unix) またはコマンドプロンプト (Windows) で、get-repository-triggers コマンドを実行して、リポジトリ用に設定されたすべてのトリガーの構造を持つ JSON ファイルを作成します。例えば、*MyDemoRepo* という名前のリポジトリ用に設定したすべてのトリガー構造を持つ *MyTriggers.json* という名前の JSON ファイルを作成するには、次のようにします。

```
aws codecommit get-repository-triggers --repository-name MyDemoRepo
 >MyTriggers.json
```

   このコマンドは何も返しませんが、コマンドを実行したディレクトリに *MyTriggers.json* という名前のファイルが作成されます。

2. プレーンテキストエディタでこの JSON ファイルを開き、編集するトリガーのトリガーブロックに変更を加えます。`configurationId` ペアを `repositoryName` ペアに置き換えます。ファイルを保存します。

   例えば、*MyDemoRepo* という名前のリポジトリの *MyFirstTrigger* という名前のトリガーをすべてのブランチに適用されるように編集する場合は、`configurationId` を `repositoryName` に置き換えて、*######*で示している `main` および `preprod` ブランチを削除します。デフォルトでは、ブランチを指定しない場合、リポジトリ内のすべてのブランチにトリガーが適用されます。

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "destinationArn": "arn:aws:sns:us-
east-2:111122223333:MyCodeCommitTopic",
            "branches": [
                "main",
                "preprod"
            ],
            "name": "MyFirstTrigger",
            "customData": "",
            "events": [
                "all"
            ]
        }
    ]
```

```
}
```

3. ターミナルまたはコマンドラインで、put-repository-triggers コマンドを実行します。このコマンドでは、*MyFirstTrigger* トリガーへの変更を含め、リポジトリのすべてのトリガーが更新されます。

```
aws codecommit put-repository-triggers --repository-name MyDemoRepo
 file://MyTriggers.json
```

このコマンドでは、次のような設定 ID が返されます。

```
{
    "configurationId": "0123456-I-AM-AN-EXAMPLE"
}
```

AWS CodeCommit リポジトリのテストトリガー

CodeCommit リポジトリ用に作成したトリガーをテストできます。テストするには、リポジトリのサンプルデータ (最新のコミット ID など) を使用してトリガーを実行する必要があります。リポジトリのコミット履歴が存在しない場合、ゼロから構成されるサンプル値が生成されます。トリガーをテストすると、 AWS Lambda 関数であるか Amazon Simple Notification Service 通知であるかにかかわらず、CodeCommit とトリガーのターゲット間のアクセスが正しく設定されていることを確認できます。

トピック

- リポジトリのトリガーをテストする (コンソール)
- リポジトリのトリガーをテストする (AWS CLI)

リポジトリのトリガーをテストする (コンソール)

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] で、リポジトリイベントのトリガーをテストするリポジトリを選択します。
3. リポジトリのナビゲーションペインで、[設定] を選択し、[トリガー] を選択します。

4. テストするトリガーを選択して [トリガーのテスト] を選択します。成功または失敗のメッセージが表示されます。成功した場合は、対応するアクションのレスポンスも Lambda 関数または Amazon SNS トピックによって表示されます。

リポジトリのトリガーをテストする (AWS CLI)

1. ターミナル (Linux、macOS、Unix) またはコマンドプロンプト (Windows) で、get-repository-triggers コマンドを実行して、リポジトリ用に設定されたすべてのトリガーの構造を持つ JSON ファイルを作成します。例えば、MyDemoRepo という名前のリポジトリ用に設定されたすべてのトリガーの構造を持つ *TestTrigger.json* という名前の JSON ファイルを作成するには、次のようにします。

```
aws codecommit get-repository-triggers --repository-name MyDemoRepo
 >TestTrigger.json
```

   このコマンドは、コマンドを実行したディレクトリに *TestTriggers.json* という名前のファイルを作成します。

2. プレーンテキストエディタで JSON ファイルを編集し、トリガーステートメントを変更します。`configurationId` ペアを `repositoryName` ペアに置き換えます。ファイルを保存します。

   例えば、*MyDemoRepo* という名前のリポジトリで *MyFirstTrigger* という名前のトリガーをテストしてすべてのブランチに適用する場合は、`configurationId` を `repositoryName` に置き換えて、次のようなファイルを *TestTrigger.json* として保存します。

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "destinationArn": "arn:aws:sns:us-
east-2:111122223333:MyCodeCommitTopic",
            "branches": [
                "main",
                "preprod"
            ],
            "name": "MyFirstTrigger",
            "customData": "",
            "events": [
                "all"
```

```
            ]
        }
    ]
}
```

3. ターミナルまたはコマンドラインで、test-repository-triggers コマンドを実行します。このコマンドでは、*MyFirstTrigger* トリガーへの変更を含め、リポジトリのすべてのトリガーが更新されます。

```
aws codecommit test-repository-triggers --cli-input-json file://TestTrigger.json
```

このコマンドでは次のようなレスポンスが返されます。

```
{
    "successfulExecutions": [
        "MyFirstTrigger"
    ],
    "failedExecutions": []
}
```

AWS CodeCommit リポジトリからトリガーを削除する

使用されなくなったトリガーは、削除することをお勧めします。トリガーの削除は元に戻すことはできませんが、1 つを再作成することはできます。

> **Note**
> リポジトリに 1 つ以上のトリガーを設定した場合、リポジトリを削除しても、それらのトリガーのターゲットとして設定した Amazon SNS トピックおよび Lambda 関数は削除されません。不要になったリソースも削除してください。

トピック

- リポジトリからトリガーを削除する (コンソール)
- リポジトリからトリガーを削除する (AWS CLI)

リポジトリからトリガーを削除する (コンソール)

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリのリストから、リポジトリイベントのトリガーを削除するリポジトリを選択します。
3. リポジトリのナビゲーションペインで、[Settings] を選択します。[設定] で、[トリガー] を選択します。
4. トリガーのリストから削除するトリガーを選択し、[削除] を選択します。
5. ダイアログボックスで、確認して [削除] と入力します。

リポジトリからトリガーを削除する (AWS CLI)

1. ターミナル (Linux、macOS、Unix) またはコマンドプロンプト (Windows) で、get-repository-triggers コマンドを実行して、リポジトリ用に設定されたすべてのトリガーの構造を持つ JSON ファイルを作成します。例えば、MyDemoRepo という名前のリポジトリ用に設定したすべてのトリガー構造を持つ *MyTriggers.json* という名前の JSON ファイルを作成するには、次のようにします。

   ```
   aws codecommit get-repository-triggers --repository-name MyDemoRepo
    >MyTriggers.json
   ```

   このコマンドは、コマンドを実行したディレクトリに *MyTriggers.json* という名前のファイルを作成します。

2. プレーンテキストエディタで JSON ファイルを編集し、削除するトリガーのトリガーブロックを削除します。`configurationId` ペアを `repositoryName` ペアに置き換えます。ファイルを保存します。

   例えば、*MyDemoRepo* という名前のリポジトリから *MyFirstTrigger* という名前のトリガーを削除する場合は、`configurationId` を `repositoryName` に置き換えて、*######*のステートメントを削除します。

   ```
   {
       "repositoryName": "MyDemoRepo",
       "triggers": [
           {
               "destinationArn": "arn:aws:sns:us-
   east-2:111122223333:MyCodeCommitTopic",
   ```

```
            "branches": [
                "main",
                "preprod"
            ],
            "name": "MyFirstTrigger",
            "customData": "",
            "events": [
                "all"
            ]
        },
        {
            "destinationArn": "arn:aws:lambda:us-
east-2:111122223333:function:MyCodeCommitJSFunction",
            "branches": [],
            "name": "MyLambdaTrigger",
            "events": [
                "all"
            ]
        }
    ]
}
```

3. ターミナルまたはコマンドラインで、put-repository-triggers コマンドを実行します。その結果、リポジトリのトリガーは更新され、*MyFirstTrigger* トリガーは削除されます。

```
aws codecommit put-repository-triggers --repository-name MyDemoRepo
 file://MyTriggers.json
```

このコマンドでは、次のような設定 ID が返されます。

```
{
    "configurationId": "0123456-I-AM-AN-EXAMPLE"
}
```

> **Note**
>
> *MyDemoRepo* という名前のリポジトリのすべてのトリガーを削除するには、JSON ファイルが次のようになります。
>
> ```
> {
>  "repositoryName": "MyDemoRepo",
> ```

```
    "triggers": []
}
```

AWS CodeCommit リポジトリを Amazon CodeGuru Reviewer に関連付ける、または関連付けを解除する

Amazon CodeGuru Reviewer は、プログラム分析と機械学習を使用して一般的な問題を検出し、Java または Python コードにおける修正点を提案する自動化されたコードレビューサービスです。アマゾン ウェブ サービスアカウントのリポジトリを CodeGuru Reviewer に関連付けることができます。実行すると、CodeGuru Reviewer はサービスにリンクされたロールを作成します。これにより、CodeGuru Reviewer は、関連付けが行われた後に作成されたすべてのプルリクエストのコードを分析できます。

リポジトリを関連付けると、CodeGuru Reviewer は、プルリクエストの作成時に見つかった問題を分析し、コメントします。各コメントには Amazon CodeGuru Reviewer の名称が示され、CodeGuru Reviewer からのものとして明確にマークされています。プルリクエストの他のコメントと同様に、これらのコメントに返信できます。また、提案の品質に関するフィードバックを提供することもできます。このフィードバックは CodeGuru Reviewer と共有され、サービスとその提案を改善するのに役立ちます。

> **Note**
>
> リポジトリーが関連付けられる前に作成されたプルリクエストでは、CodeGuru Reviewer からのコメントは表示されません。次の理由により、関連付け後に作成されたプルリクエストにコメントが表示されない場合があります。
>
> - プルリクエストに Java または Python コードが含まれていない。
> - CodeGuru Reviewer には、プルリクエストのコードを実行して確認するのに十分な時間がありませんでした。このプロセスには最長 30 分かかることがあります。コメントはレビューの進行中に表示されますが、ジョブのステータスが [完了済み] と表示されるまでコメントは完了しません。
> - CodeGuru Reviewer は、プルリクエストの Java または Python コードに問題を発見しませんでした。
> - コードレビュージョブの実行に失敗しました。プルリクエストのレビューのステータスを確認するには、プルリクエストの [アクティビティ] タブを参照してください。

- プルリクエストに対する変更を変更タブで表示しています。プルリクエストが更新され、Amazon CodeGuru Reviewer は変更の中に問題を発見しませんでした。Amazon CodeGuru Reviewer コメントが [Changes] (変更) タブに表示されるのは、最新リビジョンのプルリクエストに対してコメントを作成した場合のみです。[アクティビティ] タブには、コメントが常に表示されます。

詳細については、AWS CodeCommit リポジトリのプルリクエストを操作する、プルリクエストのレビュー、およびAmazon CodeGuru Reviewer ユーザーガイドを参照してください。

> **Note**
>
> リポジトリを CodeGuru Reviewer に関連付けたり関連付けを解除したりするのに十分なアクセス許可を持つ IAM ユーザーまたはロールでサインインする必要があります。これらのアクセス許可を含む CodeCommit のマネージドポリシーの詳細については、「AWS CodeCommit の マネージドポリシー」および「AWS CodeCommit マネージドポリシーと Amazon CodeGuru Reviewer」を参照してください。CodeGuru Reviewer のアクセス許可とセキュリティの詳細については、Amazon CodeGuru Reviewer ユーザーガイドを参照してください。

トピック

- リポジトリを CodeGuru Reviewer に関連付ける
- CodeGuru Reviewer からリポジトリの関連付けを解除する

リポジトリを CodeGuru Reviewer に関連付ける

AWS CodeCommit コンソールを使用して、リポジトリを CodeGuru Reviewer にすばやく関連付けます。その他の方法については、Amazon CodeGuru Reviewer ユーザーガイドを参照してください。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Repositories (リポジトリ)] で、CodeGuru Reviewer に関連付けるリポジトリーの名前を選択します。
3. [設定] を選択してから、[Amazon CodeGuru Reviewer] を選択します。
4. [Associate repository (リポジトリーの関連付け)] を選択します。

> **Note**
>
> リポジトリを CodeGuru Reviewer に完全に関連付けるには、最大で 10 分かかることがあります。ステータスは自動的に更新されません。現在のステータスを表示するには、更新ボタンを選択します。

CodeGuru Reviewer からリポジトリの関連付けを解除する

AWS CodeCommit コンソールを使用して、CodeGuru Reviewer からリポジトリの関連付けをすばやく解除します。その他の方法については、Amazon CodeGuru Reviewer ユーザーガイドを参照してください。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Repositories (リポジトリ)] で、CodeGuru Reviewer から関連付けを解除するリポジトリの名前を選択します。
3. [設定] を選択してから、[Amazon CodeGuru Reviewer] を選択します。
4. [Disassociate repository (リポジトリーの関連付け解除)] を選択します。

CodeCommit リポジトリの詳細を表示する

AWS CodeCommit コンソール AWS CLI、または CodeCommit リポジトリに接続されたローカルリポジトリの Git を使用して、使用可能なリポジトリに関する情報を表示できます。

以下の手順を実行する前に、「」のステップを完了してください設定

トピック

- リポジトリの詳細を表示する (コンソール)
- CodeCommit リポジトリの詳細を表示する (Git)
- CodeCommit リポジトリの詳細を表示する (AWS CLI)

リポジトリの詳細を表示する (コンソール)

AWS CodeCommit コンソールを使用して、Amazon Web Services アカウントで作成されたすべてのリポジトリをすばやく表示します。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] には、サインインしている AWS リージョン のリポジトリに関する詳細が表示されます。リージョンセレクタを使用して、別の AWS リージョンを選択すると、そのリージョン内のリポジトリが表示されます。
3. 詳細を表示するレポジトリの名前を選択してから、以下のいずれかの操作を行います。
   - リポジトリをクローンするための URL を表示するには、[Clone URL (クローン URL)] を選択して、リポジトリのクローンに使用するプロトコルを選択します。これにより、クローン URL が複製されます。確認するには、それをプレーンテキストエディタに貼り付けます。
   - リポジトリの設定可能なオプションと、リポジトリ ARN やリポジトリ ID などの詳細を表示するには、ナビゲーションペインで [設定] を選択します。

> **Note**
>
> IAM ユーザーとしてサインインしている場合、コードやその他のコンソール設定の表示用設定を設定して保存できます。詳細については、「ユーザー設定の操作」を参照してください。

CodeCommit リポジトリの詳細を表示する (Git)

ローカルリポジトリから Git を使用して CodeCommit リポジトリの詳細を表示するには、git remote show コマンドを実行します。

以下のステップを実行する前に、ローカルリポジトリを CodeCommit リポジトリに接続します。手順については、「リポジトリへの接続」を参照してください。

1. git remote show ***remote-name*** コマンドを実行します。ここで、*remote-name* は CodeCommit のエイリアスです (デフォルトでは `origin`)。

   > **Tip**
   >
   > CodeCommit リポジトリの名前と URL のリストを取得するには、git remote -v コマンドを実行します。

   例えば、エイリアスが `origin` である CodeCommit の詳細を表示するには、次のようにします。

   ```
   git remote show origin
   ```

2. HTTPS の場合:

   ```
   * remote origin
     Fetch URL: https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
     Push  URL: https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
     HEAD branch: (unknown)
     Remote branches:
       MyNewBranch tracked
       main tracked
     Local ref configured for 'git pull':
       MyNewBranch merges with remote MyNewBranch (up to date)
     Local refs configured for 'git push':
       MyNewBranch pushes to MyNewBranch (up to date)
       main pushes to main (up to date)
   ```

   SSH の場合:

   ```
   * remote origin
     Fetch URL: ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
     Push  URL: ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
     HEAD branch: (unknown)
     Remote branches:
       MyNewBranch tracked
       main tracked
   ```

```
  Local ref configured for 'git pull':
    MyNewBranch merges with remote MyNewBranch (up to date)
  Local refs configured for 'git push':
    MyNewBranch pushes to MyNewBranch (up to date)
    main pushes to main (up to date)
```

> **Tip**
>
> IAM ユーザーの SSH キー ID を検索するには、IAM コンソールを開き、IAM ユーザーの詳細ページで [Security Credentials] を展開します。SSH キー ID は の SSH キー AWS CodeCommitにあります。

他のオプションについては、Git のドキュメントを参照してください。

CodeCommit リポジトリの詳細を表示する (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

を使用してリポジトリの詳細 AWS CLI を表示するには、次のコマンドを実行します。

- CodeCommit リポジトリ名および対応する ID を一覧表示するには、list-repositories を実行します。
- 単一の CodeCommit リポジトリに関する情報を表示するには、get-repository を実行します。
- 複数の CodeCommit リポジトリに関する情報を表示するには、batch-get-repositories を実行します。

CodeCommit リポジトリのリストを表示するには

1. list-repositories コマンドを実行します。

   ```
   aws codecommit list-repositories
   ```

   オプションの `--sort-by` または `--order` のオプションを使用して、返される情報の順序を変更することができます。

2. 成功した場合、このコマンドは、Amazon Web Services アカウントに関連付けられた CodeCommit 内のすべてのリポジトリの名前と ID を含む `repositories` オブジェクトを出力します。

   前述のコマンド例に基づいて、出力例をいくつか示します。

```
{
    "repositories": [
        {
           "repositoryName": "MyDemoRepo",
           "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE"
        },
        {
           "repositoryName": "MyOtherDemoRepo",
           "repositoryId": "cfc29ac4-b0cb-44dc-9990-f6f51EXAMPLE"
        }
    ]
}
```

単一の CodeCommit リポジトリの詳細を表示するには

1. CodeCommit リポジトリの名前、および `--repository-name` オプションを指定して、get-repository コマンドを実行します。

   > **Tip**
   > CodeCommit リポジトリの名前を取得するには、list-repositories コマンドを実行します。

   例えば、`MyDemoRepo` という名前の CodeCommit リポジトリの詳細を表示するには、次のようにします。

```
aws codecommit get-repository --repository-name MyDemoRepo
```

2. 成功すると、このコマンドは次の情報を持つ `repositoryMetadata` オブジェクトを出力します。

   - リポジトリの名前 (`repositoryName`)。

- リポジトリの説明 (`repositoryDescription`)。
- リポジトリ固有のシステム生成 ID (`repositoryId`)。
- リポジトリに関連付けられたAmazon Web Services アカウントの ID (`accountId`)。

前述のコマンド例に基づいて、出力例をいくつか示します。

```
{
        "repositoryMetadata": {
            "creationDate": 1429203623.625,
            "defaultBranch": "main",
            "repositoryName": "MyDemoRepo",
            "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
            "lastModifiedDate": 1430783812.0869999,
            "repositoryDescription": "My demonstration repository",
            "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
            "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
            "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
            "accountId": "111111111111"
        }
}
```

複数の CodeCommit リポジトリの詳細情報を表示するには

1. batch-get-repositories オプションを使用して `--repository-names` コマンドを実行します。各 CodeCommit リポジトリ名の間にスペースを追加します。

> **Tip**
>
> CodeCommit リポジトリの名前を取得するには、list-repositories コマンドを実行します。

例えば、`MyDemoRepo` および `MyOtherDemoRepo` という 2 つの CodeCommit リポジトリの詳細を表示するには、次のようにします。

```
aws codecommit batch-get-repositories --repository-names MyDemoRepo MyOtherDemoRepo
```

2. 成功すると、次の情報を含むオブジェクトが出力されます。

   - 見つからなかった CodeCommit リポジトリのリスト (`repositoriesNotFound`)。
   - CodeCommit リポジトリのリスト (`repositories`)。各 CodeCommit リポジトリ名の後に、以下が続きます。
     - リポジトリの説明 (`repositoryDescription`)。
     - リポジトリ固有のシステム生成 ID (`repositoryId`)。
     - リポジトリに関連付けられたAmazon Web Services アカウントの ID (`accountId`)。

   前述のコマンド例に基づいて、出力例をいくつか示します。

```
{
        "repositoriesNotFound": [],
        "repositories": [
             {
                "creationDate": 1429203623.625,
                "defaultBranch": "main",
                "repositoryName": "MyDemoRepo",
                "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo",
                "lastModifiedDate": 1430783812.0869999,
                "repositoryDescription": "My demonstration repository",
                "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo",
                "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
                "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
                "accountId": "111111111111"
            },
            {
                "creationDate": 1429203623.627,
                "defaultBranch": "main",
                "repositoryName": "MyOtherDemoRepo",
                "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyOtherDemoRepo",
                "lastModifiedDate": 1430783812.0889999,
                "repositoryDescription": "My other demonstration repository",
                "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/
repos/MyOtherDemoRepo",
                "repositoryId": "cfc29ac4-b0cb-44dc-9990-f6f51EXAMPLE",
                "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo",
                "accountId": "111111111111"
```

```
            }
        ],
        "repositoriesNotFound": []
    }
```

AWS CodeCommit リポジトリ設定を変更する

AWS CLI および AWS CodeCommit コンソールを使用して、CodeCommit リポジトリの説明や名前などの設定を変更できます。

> **Important**
> リポジトリの名前を変更すると、リモート URL に古い名前を使用するローカルレポジトリが壊れることがあります。git remote set-url コマンドを実行して、新しいリポジトリの名前を使用するようにリモート URL を更新します。

トピック

- リポジトリ設定を変更する (コンソール)
- AWS CodeCommit リポジトリ設定の変更 (AWS CLI)

リポジトリ設定を変更する (コンソール)

AWS CodeCommit コンソールを使用して で CodeCommit リポジトリの設定を変更するには AWS CodeCommit、次の手順に従います。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリのリストで、設定を変更するリポジトリの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) をクリックします。
4. リポジトリの名前を変更するには、[Repository name (リポジトリ名)] の [Name (名前)] テキストボックスに新しい名前を入力し、[Save (保存)] を選択します。プロンプトが表示されたら、選択を確認します。

> **Important**
>
> AWS CodeCommit リポジトリの名前を変更すると、ユーザーがリポジトリに接続する必要がある SSH および HTTPS URLs が変更されます。ユーザーは、接続設定を更新するまでこのリポジトリに接続できなくなります。また、リポジトリの ARN が変更されるため、リポジトリ名を変更すると、このリポジトリの ARN に依存するすべての IAM ユーザーポリシーが無効になります。
> リポジトリ名の変更後、そのリポジトリに接続する各ユーザーは、git remote set-url コマンドで、使用する新しい URL を指定する必要があります。例えば、リポジトリの名前を MyDemoRepo から MyRenamedDemoRepo に変更した場合、HTTPS を使用してリポジトリに接続するユーザーは、以下の Git コマンドを実行します。
>
> ```
> git remote set-url origin https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyRenamedDemoRepo
> ```
>
> SSH を使用してリポジトリに接続するユーザーは、以下の Git コマンドを実行します。
>
> ```
> git remote set-url origin ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyRenamedDemoRepo
> ```
>
> 他のオプションについては、Git のドキュメントを参照してください。

5. リポジトリの説明を変更するには、[Description (説明)] テキストボックスでテキストを変更し、[Save (保存)] を選択します。

> **Note**
>
> コンソールの説明フィールドに [Markdown] と表示され、すべての HTML 文字とすべての有効な Unicode 文字を使用できます。アプリケーションデベロッパーとして `GetRepository` または `BatchGetRepositories` API を使用していて、ウェブブラウザにレポジトリの説明フィールドを表示させる場合は、CodeCommit API リファレンスを参照してください。

6. デフォルトのブランチを変更するには、[デフォルトブランチ] のブランチドロップダウンリストから別のブランチを選択します。[保存] を選択します。
7. リポジトリ内のデータの暗号化と復号に使用する AWS KMS 暗号化キーを変更するには、リポジトリ暗号化キーで、使用するキーのタイプを指定する AWS マネージドキーまたはカスタマー

マネージドキーを選択します。カスタマーマネージドキーを選択する場合は、キーの ARN を入力します。[保存] を選択します。

8. リポジトリを削除するには、[Delete repository] を選択します。[Type the name of the repository to confirm deletion] (削除の確認のため、リポジトリ名を入力) の横にあるボックスに **`delete`** と入力し、[Delete] (削除) を選択します。

> **Important**
>
> でこのリポジトリを削除すると AWS CodeCommit、ローカルリポジトリまたは共有リポジトリに複製できなくなります。また、ローカルリポジトリおよび共有リポジトリから、そのリポジトリに対して、データをプルしたりプッシュしたりできなくなります。このアクションは元に戻すことができません。

AWS CodeCommit リポジトリ設定の変更 (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

AWS CLI を使用して で CodeCommit リポジトリの設定を変更するには AWS CodeCommit、次のコマンドを 1 つ以上実行します。

- update-repository-description。CodeCommit リポジトリの説明を変更します。
- update-repository-name。CodeCommit リポジトリの名前を変更します。

CodeCommit リポジトリの説明を変更するには

1. 次のように指定して update-repository-description コマンドを実行します。

   - CodeCommit リポジトリの名前 (`--repository-name` オプション)。

     > **Tip**
     >
     > CodeCommit リポジトリの名前を取得するには、list-repositories コマンドを実行します。

   - 新しいリポジトリの説明 (`--repository-description` オプションで指定)。

> **Note**
>
> コンソールの説明フィールドに [Markdown] と表示され、すべての HTML 文字とすべての有効な Unicode 文字を使用できます。アプリケーションデベロッパーとして `GetRepository` または `BatchGetRepositories` API を使用していて、ウェブブラウザにレポジトリの説明フィールドを表示させる場合は、CodeCommit API リファレンスを参照してください。

例えば、`MyDemoRepo` という名前の CodeCommit リポジトリの説明を `This description was changed` に変更するには、以下のようにします。

```
aws codecommit update-repository-description --repository-name MyDemoRepo --
repository-description "This description was changed"
```

このコマンドは、エラーがある場合にのみ出力を生成します。

2. 変更した説明を確認するには、get-repository コマンドを実行します。このとき、`--repository-name` オプションで、説明を変更した CodeCommit リポジトリの名前を指定します。

   コマンドの出力には、`repositoryDescription` の変更後のテキストが表示されます。

CodeCommit リポジトリの名前を変更するには

1. 次のように指定して update-repository-name コマンドを実行します。

   - CodeCommit リポジトリの現在の名前 (`--old-name` オプション)。

     > **Tip**
     >
     > CodeCommit リポジトリの名前を取得するには、list-repositories コマンドを実行します。

   - CodeCommit リポジトリの新しい名前 (`--new-name` オプション)。

たとえば、MyDemoRepo という名前のリポジトリを MyRenamedDemoRepo に変更するには、以下のようにします。

```
aws codecommit update-repository-name --old-name MyDemoRepo --new-name
 MyRenamedDemoRepo
```

このコマンドは、エラーがある場合にのみ出力を生成します。

> **Important**
>
> AWS CodeCommit リポジトリの名前を変更すると、ユーザーがリポジトリに接続する必要がある SSH および HTTPS URLs が変更されます。ユーザーは、接続設定を更新するまで、このリポジトリに接続できなくなります。また、リポジトリの ARN が変更されるため、リポジトリ名を変更すると、このリポジトリの ARN に依存する IAM ユーザーポリシーはすべて無効になります。

2. 変更した名前を確認するには、list-repositories コマンドを実行し、リポジトリ名のリストを表示します。

ローカルリポジトリと AWS CodeCommit リポジトリとの間で変更を同期させる

Git を使用して、ローカルリポジトリと、ローカルリポジトリに接続されている CodeCommit リポジトリとの間で変更を同期させます。

git push ***remote-name branch-name*** を実行して、ローカルリポジトリから CodeCommit リポジトリに変更をプッシュします。

git pull ***remote-name branch-name*** を実行して、CodeCommit リポジトリからローカルリポジトリに変更をプルします。

プッシュとプルの両方で、*remote-name* は、ローカルリポジトリが CodeCommit リポジトリに使用するニックネームです。*branch-name* は、CodeCommit 内のプッシュ先またはプル元のブランチの名前です。

> **Tip**
>
> ローカルリポジトリが CodeCommit リポジトリに使用するニックネームを取得するには、git remote を実行します。ブランチ名のリストを取得するには、git branch を実行します。現在のブランチ名の横にはアスタリスク (*) が表示されます (git status を実行して、ブランチ名を表示することもできます)

> **Note**
>
> リポジトリのクローンを作成した場合、ローカルリポジトリから見ると、*remote-name* は CodeCommit リポジトリの名前ではありません。リポジトリを複製すると、*remote-name* は自動的に `origin` に設定されます。

例えば、ローカルリポジトリから、`origin` というニックネームの CodeCommit リポジトリにある `main` ブランチに変更をプッシュするには、次のようにします。

```
git push origin main
```

例えば、`origin` というニックネームの CodeCommit リポジトリにある `main` ブランチから、ローカルリポジトリに変更をプルするには、次のようにします。

```
git pull origin main
```

> **Tip**
>
> `-u` に git push オプションを追加すると、アップストリーム追跡情報が設定されます。たとえば、git push -u origin main を実行すると、以降、*remote-name branch-name* なしで git push と git pull を実行できます。アップストリーム追跡情報を取得するには、git remote show ***remote-name*** (例: git remote show origin) を実行します。

他のオプションについては、Git のドキュメントを参照してください。

追加の Git リポジトリにコミットをプッシュする

2 つのリモートリポジトリに変更をプッシュするようにローカルリポジトリを設定できます。たとえば、 AWS CodeCommitを試している間、既存の Git リポジトリソリューションを引き続き使用できるようにする場合があります。以下の基本的なステップに従って、ローカルリポジトリの変更を CodeCommit および別の Git リポジトリにプッシュします。

> **Tip**
>
> Git リポジトリがない場合は、CodeCommit 以外のサービスで空のリポジトリを作成し、CodeCommit リポジトリをそのリポジトリに移行できます。「」に示している同様の手順に従ってくださいCodeCommit に移行する

1. コマンドプロンプトまたはターミナルから、ローカルの repo ディレクトリに切り替えて、git remote -v コマンドを実行します。次のような出力が表示されます。

   HTTPS の場合:

   ```
   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
   ```

   SSH の場合:

   ```
   origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
   origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
   ```

2. git remote set-url --add --push origin ***git-repository-name*** コマンドを実行します。ここで、*git-repository-name* はコードをホストする Git リポジトリの URL と名前です。このコマンドは `origin` のプッシュ先をその Git リポジトリに変更します。

   > **Note**
   >
   > git remote set-url --add --push は、プッシュのデフォルト URL よりも優先されるため、このコマンドを 2 回実行する必要があります。詳しくは後で説明します。

例えば、以下のコマンドはオリジンのプッシュ先を *some-URL*/MyDestinationRepo に変更します。

```
git remote set-url --add --push origin some-URL/MyDestinationRepo
```

このコマンドは何も返しません。

> **Tip**
> 認証情報を必要とする Git リポジトリにプッシュする場合は、認証情報ヘルパーまたは *some-URL* 文字列の設定で、それらの認証情報を設定してください。設定しないと、そのリポジトリへのプッシュは失敗します。

3. git remote -v コマンドをもう一度実行すると、以下のような出力が生成されます。

   HTTPS の場合:

   ```
   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
   origin  some-URL/MyDestinationRepo (push)
   ```

   SSH の場合:

   ```
   origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
   origin  some-URL/MyDestinationRepo (push)
   ```

4. CodeCommit リポジトリを追加します。git remote set-url --add --push origin をもう一度実行します。今度は、CodeCommit リポジトリの URL とリポジトリ名を指定します。

   例えば、以下のコマンドは、オリジンのプッシュ を https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo に追加します。

   HTTPS の場合:

   ```
   git remote set-url --add --push origin https://git-codecommit.us-
   east-2.amazonaws.com/v1/repos/MyDemoRepo
   ```

   SSH の場合:

```
git remote set-url --add --push origin ssh://git-codecommit.us-
east-2.amazonaws.com/v1/repos/MyDemoRepo
```

このコマンドは何も返しません。

5. git remote -v コマンドをもう一度実行すると、以下のような出力が生成されます。

   HTTPS の場合:

```
origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
origin  some-URL/MyDestinationRepo (push)
origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
```

   SSH の場合:

```
origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
origin  some-URL/MyDestinationRepo (push)
origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
```

   ここではプッシュ先として 2 つの Git リポジトリがありますが、最初に *some-URL*/MyDestinationRepo にプッシュされます。そのリポジトリへのプッシュが失敗すると、コミットはいずれのリポジトリにもプッシュされません。

   > **Tip**
   > 他のリポジトリに手動で入力する認証情報が必要な場合は、プッシュ順を変更して、最初に CodeCommit にプッシュするようにしてください。git remote set-url --delete を実行して、最初にプッシュされるリポジトリを削除してから、git remote set-url --add を実行して、そのリポジトリをもう一度追加し、リストで 2 番目のプッシュ先になるようにします。
   > 他のオプションについては、Git のドキュメントを参照してください。

6. この時点で、両方のリモートリポジトリにプッシュしていることを確認するには、テキストエディタを使用して、ローカルリポジトリに以下のテキストファイルを作成します。

```
bees.txt
-------
```

```
Bees are flying insects closely related to wasps and ants, and are known for their
 role in pollination and for producing honey and beeswax.
```

7. git add を実行して、変更をローカルリポジトリにステージングします。

```
git add bees.txt
```

8. git commit を実行して、変更をローカルリポジトリにコミットします。

```
git commit -m "Added bees.txt"
```

9. コミットをローカルリポジトリからリモートリポジトリにプッシュするには、git push -u ***remote-name branch-name*** を実行します。ここで、*remote-name* は、ローカルリポジトリがリモートリポジトリに使用するニックネームであり、*branch -name* は、リポジトリにプッシュするブランチの名前です。

> **Tip**
>
> -u オプションを使用する必要があるのは、初めてプッシュするときのみです。その後、アップストリーム追跡情報が設定されます。

たとえば、git push -u origin main を実行すると、想定されるブランチの両方のリモートリポジトリにプッシュが送られ、以下のような出力が表示されます。

HTTPS の場合:

```
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 5.61 KiB | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To some-URL/MyDestinationRepo
   a5ba4ed..250f6c3  main -> main
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 5.61 KiB | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote:
To https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
```

```
   a5ba4ed..250f6c3  main -> main
```

SSH の場合:

```
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 5.61 KiB | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To some-URL/MyDestinationRepo
   a5ba4ed..250f6c3  main -> main
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 5.61 KiB | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote:
To ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
   a5ba4ed..250f6c3  main -> main
```

他のオプションについては、Git のドキュメントを参照してください。

ロールを使用して AWS CodeCommit リポジトリへのクロスアカウントアクセスを設定する

別の AWS アカウントの IAM ユーザーおよびグループに、CodeCommit リポジトリへのアクセスを設定できます。このプロセスはよく、クロスアカウントアクセスと呼ばれます。このセクションでは、 AWS アカウント (AccountA と表示) の米国東部 (オハイオ) リージョンの *MySharedDemoRepo* という名のリポジトリに対する、別の AWS アカウント (AccountB と表示) の *DevelopersWithCrossAccountRepositoryAccess* という名の IAM グループに属する IAM ユーザーのクロスアカウントアクセスを設定する例と手順について説明します。

このセクションは、3 つのパートに分かれています。

- AccountA の管理者用アクション。
- AccountB の管理者用アクション。
- AccountB のユーザーリポジトリ用アクション。

クロスアカウントアクセスを設定するには

- AccountA の管理者は、CodeCommit でリポジトリを作成および管理し、IAM でロールを作成するためのアクセス許可がある IAM ユーザーとしてサインインします。管理ポリシーを使用している場合、この IAM ユーザーに IAMFullAccess および AWSCodeCommitFullAccess を適用します。

  AccountA におけるアカウント ID の例は、*111122223333* とします。

- AccountB の管理者は、IAM ユーザーとグループを作成および管理し、ユーザーとグループにポリシーを設定するためのアクセス許可がある IAM ユーザーとしてサインインします。管理ポリシーを使用している場合、この IAM ユーザーに IAMFullAccess を適用します。

  AccountB におけるアカウント ID の例は、*888888888888* とします。

- AccountB のリポジトリユーザーは、デベロッパーのアクティビティをエミュレートするために、AccountA で CodeCommit リポジトリにアクセスできるように作成された IAM グループのメンバーの IAM ユーザーとしてサインインします。このアカウントは以下のように設定する必要があります。
  - AWS マネジメントコンソールへのアクセス。
  - AWS リソースに接続するときに使用するアクセスキーとシークレットキー、および AccountA のリポジトリにアクセスするときに引き受けるロールの ARN。
  - リポジトリのクローンが作成されるローカルコンピュータ上の git-remote-codecommit ユーティリティ。このユーティリティには、Python とそのインストーラ pip が必要です。このユーティリティは、Python パッケージインデックスウェブサイトの git-remote-codecommit からダウンロードできます。

  詳細については、git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順 および IAM ユーザーを参照してください。

トピック

- リポジトリへのクロスアカウントアクセス: AccountA の管理者のアクション
- リポジトリへのクロスアカウントアクセス: AccountB の管理者のアクション
- リポジトリへのクロスアカウントアクセス: AccountB のリポジトリユーザーのアクション

リポジトリへのクロスアカウントアクセス: AccountA の管理者のアクション

AccountB のユーザーまたはグループが AccountA のリポジトリにアクセスすることを許可するには、AccountA の管理者は以下を行う必要があります。

- AccountA でリポジトリにアクセスを付与するポリシーを作成します。
- AccountB の IAM ユーザーおよびグループが引き受けることができるロールを AccountA で作成します。
- ロールへのポリシーの付与

次のセクションでは、手順と例を示します。

トピック

- ステップ 1: AccountA でリポジトリアクセス用のポリシーを作成する
- ステップ 2: AccountA でリポジトリアクセス用のロールを作成する

ステップ 1: AccountA でリポジトリアクセス用のポリシーを作成する

AccountA で、AccountB のユーザーに AccountA のリポジトリへのアクセスを許可するポリシーを作成できます AccountB 。許可するアクセスのレベルに応じて、次のいずれかのリンクを実行します。

- AccountB ユーザーに特定のリポジトリへのアクセスを許可しますが、AccountA のすべてのリポジトリの一覧を閲覧することを許可しないポリシーを設定します。
- AccountB ユーザーが AccountA のすべてのリポジトリの一覧からリポジトリを選択することを許可する追加のアクセスを設定します。

リポジトリアクセス用のポリシーを作成するには

1. AccountA でポリシーを作成するアクセス許可を持つ IAM ユーザーとして AWS マネジメントコンソールにサインインします。
2. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
3. ナビゲーションペインで、ポリシー を選択してください。
4. [ポリシーの作成] を選択します。

5. [JSON] タブを選択して、次の JSON ポリシードキュメントを JSON テキストボックスに貼り付けます。*us-east-2* をレポジトリの AWS リージョン に、*111122223333* を AccountA のアカウント ID に、*MySharedDemoRepo* を AccountA の CodeCommit レポジトリの名前に、それぞれ置き換えます。

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "codecommit:BatchGet*",
            "codecommit:Create*",
            "codecommit:DeleteBranch",
            "codecommit:Get*",
            "codecommit:List*",
            "codecommit:Describe*",
            "codecommit:Put*",
            "codecommit:Post*",
            "codecommit:Merge*",
            "codecommit:Test*",
            "codecommit:Update*",
            "codecommit:GitPull",
            "codecommit:GitPush"
        ],
        "Resource": [
            "arn:aws:codecommit:us-east-2:111122223333:MySharedDemoRepo"
        ]
    }
]
}
```

このロールを引き受けるユーザーが CodeCommit コンソールのリポジトリの一覧を表示できるようにするには、次に示すように、追加のステートメントをポリシーに追加します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codecommit:BatchGet*",
```

```
                "codecommit:Create*",
                "codecommit:DeleteBranch",
                "codecommit:Get*",
                "codecommit:List*",
                "codecommit:Describe*",
                "codecommit:Put*",
                "codecommit:Post*",
                "codecommit:Merge*",
                "codecommit:Test*",
                "codecommit:Update*",
                "codecommit:GitPull",
                "codecommit:GitPush"
            ],
            "Resource": [
                "arn:aws:codecommit:us-east-2:111122223333:MySharedDemoRepo"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "codecommit:ListRepositories",
            "Resource": "*"
        }
    ]
}
```

このアクセスは、このポリシーでこのロールを引き受けるユーザーにアクセス許可があるリポジトリの検索を容易にします。これで、一覧からリポジトリの名前を選択し、共有するリポジトリのホームページに移動することができます (Code)。ユーザーは一覧に表示されるその他のリポジトリにはアクセスすることはできませんが、[ダッシュボード] ページで AccountA のリポジトリを表示することはできます。

このロールを引き受けるユーザーに AccountA のすべてのリポジトリの一覧を表示したくない場合には、最初のポリシー例を使用しながら、CodeCommit コンソールの共有するリポジトリのホームページへの直接リンクをこれらのユーザーに送信するようにします。

6. [ポリシーの確認] を選択します。ポリシー検証で構文エラーが報告されます (サンプルの Amazon Web Services アカウント ID およびリポジトリ名を、自身の Amazon Web Services アカウント ID およびリポジトリ名に置き換えることを忘れた場合など)。
7. [ポリシーの確認] ページで、ポリシー名を入力します (例: *CrossAccountAccessForMySharedDemoRepo*)。このポリシーにオプションの説明を提供することもできます。[ポリシーの作成] を選択します。

ステップ 2: AccountA でリポジトリアクセス用のロールを作成する

ポリシーを設定したら、AccountB の IAM ユーザーおよびグループが引き受けるロールを作成し、ポリシーをこのロールにアタッチします。

リポジトリアクセス用のロールを作成するには

1. IAM コンソールで、[ロール] を選択します。
2. [ロールの作成] を選択します。
3. 別のAmazon Web Services アカウントを選択します。
4. [アカウント ID]、で AccountB の Amazon Web Services アカウント ID を入力します (例: *888888888888*)。[Next: Permissions (次へ: アクセス許可)] を選択します。
5. [アクセス権限ポリシーをアタッチする] で、前の手順で作成したポリシーを選択します (*CrossAccountAccessForMySharedDemoRepo*)。[Next: Review] を選択します。
6. [ロール名] に、ロールの名前を入力します (例: *MyCrossAccountRepositoryContributorRole*)。他のユーザーがこのロールの役割を理解するように、オプションの説明を入力することもできます。
7. [ロールの作成] を選択します。
8. 作成したロールを開き、ロールの ARN をコピーします (たとえば、`arn:aws:iam::`*111122223333*`:role/`*MyCrossAccountRepositoryContributorRole*)。A の管理者にこの ARN を提供する必要があります。

リポジトリへのクロスアカウントアクセス: AccountB の管理者のアクション

AccountB のユーザーまたはグループが AccountA のリポジトリにアクセスすることを許可するには、AccountB の管理者は AccountB でグループを作成する必要があります。このグループには、AccountA の管理者が作成したロールをグループのメンバーが引き受けることを許可するポリシーを設定する必要があります。

次のセクションでは、手順と例を示します。

トピック

- ステップ 1: AccountB ユーザーにリポジトリアクセスがある IAM グループを作成する
- ステップ 2: ポリシーを作成し、IAM グループにユーザーを追加する

ステップ 1: AccountB ユーザーにリポジトリアクセスがある IAM グループを作成する

AccountB のどの IAM ユーザーが AccountA のリポジトリにアクセスできるかを管理する最も簡単な方法は、AccountA でロールを引き受けるアクセス権限がある IAM グループを AccountB で作成し、IAM ユーザーをこのグループに追加することです。

クロスアカウントリポジトリアクセス用のグループを作成するには

1. AccountB で IAM グループとポリシーを作成し、IAM ユーザーを管理するために必要なアクセス許可を持つ IAM ユーザーとして AWS マネジメントコンソールにサインインします。
2. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
3. IAM コンソールで [グループ] を選択します。
4. [Create New Group (新しいグループの作成)] を選択します。
5. [グループ名] でこのグループの名前を入力します (たとえば、*DevelopersWithCrossAccountRepositoryAccess*)。[Next Step] (次のステップ) をクリックします。
6. [Attach Policy] で、[Next Step] を選択します。次の手順でクロスアカウントポリシーを作成します。このグループの作成を完了します。

ステップ 2: ポリシーを作成し、IAM グループにユーザーを追加する

グループを作成したら、このグループのメンバーが AccountA のリポジトリにアクセスできるロールを引き受けることができるポリシーを作成します。続いて、AccountA へのアクセスを許可する AccountB の IAM ユーザーをグループに追加します。

グループにポリシーを作成し、それにユーザーを追加する

1. IAM コンソールで [グループ] を選択し、作成したグループの名前を選択します (例: *DevelopersWithCrossAccountRepositoryAccess*)。
2. [Permissions] タブを選択します。[インラインポリシー] を展開し、インラインポリシーを作成するリンクを選択します。(インラインポリシーがすでにあるグループを設定する場合には、[グループポリシーの作成] を選択します。)
3. [Custom Policy] を選択し、[Select] を選択します。
4. [ポリシー名] にこのポリシーの名前を入力します (*AccessPolicyForSharedRepository* など)。

5. [ポリシードキュメント] で次のポリシーを貼り付けます。`Resource` で、ARN を AccountA の管理者が作成したポリシーの ARN に置き換え (例: arn:aws:iam::*111122223333*:role/*MyCrossAccountRepositoryContributorRole*)、[ポリシーの適用] を選択します。AccountA の管理者が作成したポリシーについての詳細は、「ステップ 1: AccountA でリポジトリアクセス用のポリシーを作成する」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource":
 "arn:aws:iam::111122223333:role/MyCrossAccountRepositoryContributorRole"
  }
}
```

6. [Users] (ユーザー) タブを選択します。[グループにユーザーを追加] タブを選択し、AccountB の IAM ユーザーを追加します。例えば、ユーザー名が *Saanvi_Sarkar* の IAM ユーザーをこのグループに追加できます。

> **Note**
>
> AccountB のユーザーは、共有 CodeCommit リポジトリにアクセスするためにローカルコンピューターを設定するためのアクセスキーおよびシークレットキーを含む、プログラムへのアクセス権限を有している必要があります。IAM ユーザーを作成する場合は、アクセスキーとシークレットキーを必ず保存してください。 AWS アカウントのセキュリティを確保するために、シークレットアクセスキーには、アクセスキーの作成時にのみアクセスできます。

リポジトリへのクロスアカウントアクセス: AccountB のリポジトリユーザーのアクション

AccountA でリポジトリにアクセスするには、AccountB グループのユーザーがリポジトリアクセスのためにローカルコンピューターを設定する必要があります。次のセクションでは、手順と例を示します。

トピック

- ステップ 1: AccountA のリポジトリにアクセスするように AccountB ユーザーの AWS CLI と Git AccountA を設定する
- ステップ 2: AccountA の CodeCommit リポジトリのクローンを作成してアクセスする

ステップ 1: AccountA のリポジトリにアクセスするように AccountB ユーザーの AWS CLI と Git AccountA を設定する

SSH キーまたは Git 認証情報を使用して別の Amazon Web Services アカウントのリポジトリにアクセスすることはできません。AccountB のユーザーは、git-remote-codecommit を使用するか (推奨)、AccountA で共有 CodeCommit リポジトリにアクセスするための認証情報ヘルパーを使用して、自身のコンピューターを設定する必要があります。ただし、AccountB のリポジトリにアクセスするときには、SSH キーまたは Git 認証情報を続けて使用できます。

git-remote-codecommit を使用してアクセスを設定する手順は、次のとおりです。git-remote-codecommit をまだインストールしていない場合は、Python パッケージインデックスウェブサイトの git-remote-codecommit からダウンロードします。

クロスアカウントアクセス用に AWS CLI と Git を設定するには

1. ローカルコンピュータ AWS CLI に をインストールします。詳細については、「AWS CLIのインストール」で使用しているオペレーティングシステムの説明を参照してください。
2. ローカルコンピュータに Git をインストールします。Git をインストールするには、Git ダウンロード や Git for Windows などのウェブサイトが推奨されます。

> **Note**
>
> CodeCommit は Git バージョン 1.7.9 以降をサポートしています。Git バージョン 2.28 は、初期コミットのブランチ名の設定をサポートしています。最新バージョンの Git を使用することをお勧めします。Git は、定期的に更新されている、発展中のプラットフォームです。機能の変更により、CodeCommit での動作が影響を受ける場合があります。特定のバージョンの Git と CodeCommit で問題が発生した場合は、 の情報を確認してください トラブルシューティング

3. ターミナルまたはコマンドラインから、リポジトリをクローンするディレクトリの場所で、git config --local user.name コマンドおよび git config --local user.email コマンドを実行して、リポジトリに作成するコミットのユーザー名および E メールを設定します。例:

```
git config --local user.name "Saanvi Sarkar"
git config --local user.email saanvi_sarkar@example.com
```

これらのコマンドは何も返しませんが、指定した E メールおよびユーザー名は AccountA のリポジトリで作成したコミットに関連付けられます。

4. AccountB でリソースに接続する際に使用するデフォルトのプロファイルを設定するには、aws configure --profile コマンドを実行します。プロンプトが表示されたら、IAM ユーザーのアクセスキーとシークレットキーを提供します。

> **Note**
>
> を既にインストール AWS CLI してプロファイルを設定している場合は、このステップをスキップできます。

たとえば、次のコマンドを実行して、米国東部 (オハイオ) (us-east-2) の AccountB の AWS リソースにアクセスするために使用するデフォルト AWS CLI プロファイルを作成します。

```
aws configure
```

プロンプトが表示されたら、次の情報を入力します。

```
AWS Access Key ID [None]: Your-IAM-User-Access-Key
AWS Secret Access Key ID [None]: Your-IAM-User-Secret-Access-Key
Default region name ID [None]: us-east-2
Default output format [None]: json
```

5. aws configure --profile コマンドを実行して、AccountA でリポジトリに接続するときに使用するプロファイルを設定します。プロンプトが表示されたら、IAM ユーザーのアクセスキーとシークレットキーを提供します。例えば、次のコマンドを実行して、米国東部 (オハイオ) (us-east-2) の AccountA のリポジトリにアクセスするために使用する *MyCrossAccountAccessProfile* という名前の AWS CLI プロファイルを作成します。

```
aws configure --profile MyCrossAccountAccessProfile
```

プロンプトが表示されたら、次の情報を入力します。

```
AWS Access Key ID [None]: Your-IAM-User-Access-Key
AWS Secret Access Key ID [None]: Your-IAM-User-Secret-Access-Key
Default region name ID [None]: us-east-2
Default output format [None]: json
```

6. プレーンテキストエディタで `config` ファイル (AWS CLI 設定ファイルとも呼ばれる) を開きます。オペレーティングシステムによっては、このファイルは Linux、macOS、Unix の `~/.aws/config`、あるいは Windows の *drive*:\Users*USERNAME*\.aws\config にある場合があります。

7. ファイルで、AccountB のリポジトリへのアクセス用に設定したデフォルトプロファイルに対応するエントリを見つけます。これは次のように表示されます。

```
[default]
region = us-east-2
output = json
```

プロファイル設定に `account` に追加します。AccountB の AWS アカウント ID を指定します。例:

```
[default]
account = 888888888888
region = us-east-2
output = json
```

8. 作成した *MyCrossAccountAccessProfile* に対応するエントリをこのファイルで検索します。これは次のように表示されます。

```
[profile MyCrossAccountAccessProfile]
region = us-east-2
output = json
```

プロファイル設定に `account`、`role_arn`、および `source_profile` を追加します。AccountA の Amazon Web Services アカウント ID (他のアカウントのリポジトリにアクセスするために引き受ける AccountA のロールの ARN)、および AccountB のデフォルトの AWS CLI プロファイルの名前を入力します。例:

```
[profile MyCrossAccountAccessProfile]
region = us-east-2
```

```
account = 111122223333
role_arn = arn:aws:iam::111122223333:role/MyCrossAccountRepositoryContributorRole
source_profile = default
output = json
```

変更を保存し、プレーンテキストエディタを閉じます。

ステップ 2: AccountA の CodeCommit リポジトリのクローンを作成してアクセスする

git clone、git push、および git pull を実行して、CodeCommit リポジトリのクロスアカウントをクローン、プッシュ、プルします。 AWS マネジメントコンソールにサインインし、ロールを切り替え、CodeCommit コンソールを使用して他のアカウントのリポジトリとやり取りすることもできます。

> **Note**
>
> IAM ロールの設定に応じて、CodeCommit のデフォルトページでリポジトリを表示できる場合もあります。リポジトリを表示できない場合は、CodeCommit コンソールの共有リポジトリの [コード] ページへの URL リンクを E メールで送信するようにリポジトリ管理者に申請してください。この URL は次のようになります。
>
> ```
> https://console.aws.amazon.com/codecommit/home?region=us-east-2#/
> repository/MySharedDemoRepo/browse/HEAD/--/
> ```

クロスアカウントリポジトリをローカルコンピューターにクローンするには

1. コマンドラインまたはターミナルで、リポジトリをクローンするディレクトリから、HTTPS (GRC) クローン URL で git clone コマンドを実行します。例:

   ```
   git clone codecommit://MyCrossAccountAccessProfile@MySharedDemoRepo
   ```

   他の指定がない限り、リポジトリはこのリポジトリと同じ名前のサブディレクトリにクローンされます。

2. クローンしたリポジトリのディレクトリを変更し、ファイルに変更を加えるか、またはファイルを追加します。たとえば、*NewFile.txt* という名前のファイルを追加できます。

3. ローカルリポジトリの変更履歴にファイルを追加して変更をコミットし、そのファイルを CodeCommit リポジトリにプッシュします。例:

```
git add NewFile.txt
git commit -m "Added a file to test cross-account access to this repository"
git push
```

詳細については、「Git と の開始方法 AWS CodeCommit」を参照してください。

ファイルを追加したら、CodeCommit コンソールに移動してコミットの表示、他のユーザーによるリポジトリの変更の確認、プルリクエストへの参加などを行います。

CodeCommit コンソールでクロスアカウントリポジトリにアクセスするには

1. AccountA AWS Management Console のAccountB (*888888888888*) の にサインインします。AccountA
2. ナビゲーションバーでユーザー名を選択し、ドロップダウンリストから [ロールの切り替え] を選択します。

> **Note**
> このオプションを初めて選択する場合には、このページの情報を確認し、[ロールの切り替え] を再度選択します。

3. [ロールの切り替え] ページで次を実行します。
   - [アカウント] で AccountA のアカウント ID を入力します (例: *111122223333*)。
   - [ロール] で AccountA のリポジトリにアクセスするために引き受けるロール名を入力します (例: *MyCrossAccountRepositoryContributorRole*)。
   - [表示名] にこのロールのフレンドリ名を入力します。この名前は、ロールを引き受けたときにコンソールに表示されます。また、このコンソールで次回にロールを切り替えるときに引き受けるロールの一覧にも表示されます。
   - (オプション) [色] で表示名のカラーラベルを選択します。
   - [Switch Role] を選択します。

   詳細については、「ロールの切り替え (AWS Management Console)」を参照してください。

4. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。

   引き受けたロールに AccountA のリポジトリの名前を閲覧するアクセス権限がある場合には、リポジトリの一覧、およびそのステータスを閲覧するアクセス権限がないことを通知するエラーメッセージが表示されます。これは想定される動作です。リストから共有リポジトリの名前を選択します。

   引き受けたロールに AccountA のリポジトリの名前を閲覧するアクセス権限がない場合には、エラーメッセージおよびリポジトリがない空のリストが表示されます。リポジトリに URL リンクを貼り付けるか、コンソールリンクを変更し、`/list` を共有リポジトリの名前に変更します (`/MySharedDemoRepo` など)。
5. [コード] で、ローカルコンピュータから追加したファイルの名前を検索します。その名前を選択してファイルでコードを表示し、残りのリポジトリを表示してこの機能の使用を開始します。

   詳細については、「の開始方法 AWS CodeCommit」を参照してください。

AWS CodeCommit リポジトリを削除する

CodeCommit コンソールまたは AWS CLI を使用して、CodeCommit リポジトリを削除できます。

> **Note**
> リポジトリを削除しても、そのリポジトリのローカルコピー (ローカルレポジトリ) は削除されません。ローカルリポジトリを削除するには、ローカルマシンのディレクトリとファイル管理ツールを使用します。

トピック

- CodeCommit リポジトリを削除するには (コンソール)
- ローカルリポジトリを削除する
- CodeCommit リポジトリを削除するには (AWS CLI)

CodeCommit リポジトリを削除するには (コンソール)

CodeCommit コンソールを使用して CodeCommit リポジトリを削除するには、次のステップに従ってください。

> **Important**
>
> 削除した CodeCommit リポジトリは ローカルリポジトリや共有リポジトリに複製できなくなります。また、ローカルリポジトリおよび共有リポジトリから、そのリポジトリに対して、データをプルしたりプッシュしたりできなくなります。このアクションは元に戻すことができません。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリで、削除するリポジトリの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) をクリックします。
4. [全般] タブの [リポジトリの削除] で、[リポジトリの削除]を選択します。「**`delete`**」と入力し、[Delete (削除)] を選択します。リポジトリは完全に削除されます。

> **Note**
>
> CodeCommit のリポジトリを削除しても、ローカルリポジトリは削除されません。

ローカルリポジトリを削除する

ローカルリポジトリを含むディレクトリを削除するには、ローカルマシンのディレクトリとファイル管理ツールを使用します。

ローカルリポジトリを削除しても、それに接続されている CodeCommit レポジトリは削除されません。

CodeCommit リポジトリを削除するには (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

を使用して CodeCommit リポジトリ AWS CLI を削除するには、 delete-repository コマンドを実行し、削除する CodeCommit リポジトリの名前を指定します (`--repository-name`オプションを使用) 。

> **Important**
>
> 削除した CodeCommit リポジトリは ローカルリポジトリや共有リポジトリに複製できなくなります。また、ローカルリポジトリおよび共有リポジトリから、そのリポジトリに対して、データをプルしたりプッシュしたりできなくなります。このアクションは元に戻すことができません。

> **Tip**
>
> CodeCommit リポジトリの名前を取得するには、list-repositories コマンドを実行します。

たとえば、MyDemoRepo という名前のリポジトリを削除するには、次のようにします。

```
aws codecommit delete-repository --repository-name MyDemoRepo
```

成功すると、永久的に削除された CodeCommit リポジトリの ID が出力に表示されます。

```
{
    "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE"
}
```

CodeCommit リポジトリを削除しても、それに接続されているローカルレポジトリは削除されません。

AWS CodeCommit リポジトリ内のファイルの使用

CodeCommit では、ファイルとはバージョン管理で自己完結型の情報であり、ユーザーおよびこのファイルが保存されているリポジトリとブランチの他のユーザーに利用可能です。リポジトリのファイルは、コンピューター上と同じく、ディレクトリ構造で整理できます。コンピュータとは異なり、CodeCommit はファイルへのすべての変更を自動的に追跡します。ファイルのバージョンを比較し、別々のリポジトリブランチに異なるバージョンのファイルを保存できます。

リポジトリのファイルを追加または編集するには、Git クライアントを使用できます。CodeCommit コンソール、 AWS CLI、または CodeCommit API を使用することもできます。

CodeCommit でリポジトリの他の部分を操作する方法については、 リポジトリを操作する、プルリクエストの操作 、ブランチの操作、コミットの操作、および ユーザー設定の操作 を参照してください。

トピック

- AWS CodeCommit リポジトリ内のファイルを参照する

- AWS CodeCommit リポジトリへのファイルの作成または追加
- AWS CodeCommit リポジトリ内のファイルの内容を編集する

AWS CodeCommit リポジトリ内のファイルを参照する

CodeCommit リポジトリに接続すると、ローカルのリポジトリにクローンを作成するか、CodeCommit コンソールを使用してその内容を参照することができます。このトピックでは、CodeCommit コンソールを使用して CodeCommit リポジトリの内容を参照する方法について説明します。

> **Note**
>
> アクティブな CodeCommit ユーザーの場合、CodeCommit コンソールからコードを参照することは無料です。料金の適用時期についての情報は、料金を参照してください。

CodeCommit リポジトリを参照する

CodeCommit コンソールを使用して、リポジトリに含まれるファイルを確認したり、ファイルの内容をすばやく読み取ったりすることができます。

リポジトリの内容を参照するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] ページで、リポジトリのリストから参照するリポジトリを選択します。
3. [コード] 表示で、リポジトリのブランチのデフォルトのコンテンツを参照します。

   表示を別のブランチまたはタグに変更するには、表示セレクタボタンを選択します。ドロップダウンリストからブランチまたはタグ名を選択するか、フィルタボックスにブランチまたはタグの名前を入力し、リストから選択します。
4. 次のいずれかを行います：

   - ディレクトリの内容を表示するには、リストから選択します。ナビゲーションリスト内の任意のディレクトリを選択して、そのディレクトリ表示に戻ることができます。また、ディレクトリリストの上部にある上矢印を使用することもできます。
   - ファイルの内容を表示するには、リストから選択します。ファイルがコミットオブジェクトの制限よりも大きい場合は、コンソールに表示できず、代わりにローカルのリポジトリで表示する必要があります。詳細については、「クォータ」を参照してください。ファイルビューを終了するには、コードナビゲーションバーから、表示するディレクトリを選択します。

> **Note**
>
> すべてのバイナリファイルがコンソールで表示できるわけではありません。バイナリファイルを選択し、そのファイルが表示可能になる可能性がある場合、内容の表示を確認する警告メッセージが表示されます。ファイルの内容を表示するには、[Show file contents] を選択します。ファイルを表示しない場合は、コードナビゲーションバーから、表示するディレクトリを選択します。
> マークダウンファイル (.md) を選択した場合は、[Rendered Markdown] および [Markdown Source] ボタンを使用して、レンダリングビューと構文ビューを切り替えます。詳細については、「コンソールでの Markdown の使用」を参照してください。

AWS CodeCommit リポジトリへのファイルの作成または追加

CodeCommit コンソール AWS CLI、または Git クライアントを使用して、リポジトリにファイルを追加できます。ローカルコンピュータからファイルをアップロード、またはコンソールのコードエディタを使用してファイルを作成できます。エディタは、readme.md file などのシンプルなファイルをリポジトリのブランチに追加する素早く簡単な方法です。

トピック

- ファイルを作成またはアップロードする (コンソール)
- ファイルを追加する (AWS CLI)
- ファイルを追加する (Git)

ファイルを作成またはアップロードする (コンソール)

CodeCommit コンソールを使用してファイルを作成し、CodeCommit リポジトリ内のブランチに追加できます。ファイルの作成の一環として、ユーザー名と E メールアドレスを提供できます。コミットメッセージを追加できるため、他のユーザーがファイルの作成者と作成理由を理解できます。また、ローカルコンピューターからリポジトリのブランチに直接ファイルをアップロードすることもできます。

リポジトリにファイルを追加するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] で、ファイルを追加するレポジトリを選択します。
3. [コード] ビューで、ファイルを追加するブランチを選択します。デフォルトでは、[コード] ビューを開くとデフォルトのブランチの内容が表示されます。

   表示を別のブランチに変更するには、表示セレクタボタンを選択します。ドロップダウンリストからブランチ名を選択するか、またはフィルタボックスにブランチの名前を入力し、リストから選択します。
4. [ファイルの追加] を選択し、次のいずれかのオプションを選択します。
   - コードエディタを使用してファイルの内容を作成してリポジトリに追加するには、[ファイルの作成] を選択します。
   - ローカルコンピュータからリポジトリにファイルをアップロードするには、[ファイルのアップロード] を選択します。
5. 他のユーザーにリポジトリにこのファイルを追加した者と追加の理由についての情報を提供します。
   - [作成者名] に名前を入力します。この名前は、コミット情報で作成者名およびコミット名の両方として使用できます。CodeCommit は、デフォルトで IAM ユーザー名あるいはコンソールログインの派生を作成者名として使用します。
   - [Email address] (E メールアドレス) に E メールアドレスを入力し、他のリポジトリユーザーがこの変更について連絡できるようにします。
   - [メッセージのコミット] に短い説明を入力します。これはオプションですが、入力することを強くお勧めします。入力しない場合は、デフォルトのコミットメッセージが使用されます。
6. 次のいずれかを行ってください。
   - ファイルをアップロードする場合、ローカルコンピュータからファイルを選択します。
   - ファイルを作成している場合は、コードエディタで追加する内容を入力し、ファイルの名前を指定します。
7. [変更のコミット] を選択します。

ファイルを追加する (AWS CLI)

AWS CLI および put-file コマンドを使用して、CodeCommit リポジトリにファイルを追加できます。また、put-file コマンドを使用して、ファイルにディレクトリまたはパス構造を追加することもできます。

> **Note**
>
> CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

リポジトリにファイルを追加するには

1. ローカルコンピュータで、CodeCommit リポジトリに追加するファイルを作成します。
2. ターミナルまたはコマンドラインで、put-file コマンドを実行し、次を指定します。
   - ファイルを追加するリポジトリ。
   - ファイルを追加するブランチ。
   - ブランチに作成された最新のコミットの完全なコミット ID (ヒントあるいはヘッドコミットとも呼ばれます)。
   - ファイルのローカルの場所。この場所に使用される構文は、ローカルのオペレーティングシステムによって異なります。
   - 追加するファイル名 (ある場合には、リポジトリにアップデートしたファイルが保存される場所を含む)。
   - このファイルに関連付けるユーザー名および E メール。
   - このファイルの追加理由を説明するコミットメッセージ。

   ユーザー名、E メールアドレス、コミットメッセージはオプションですが、他のユーザーに変更者と変更の理由について理解してもらうために便利です。ユーザー名を指定しない場合、CodeCommit はデフォルトで IAM ユーザー名あるいはコンソールログインの派生を作成者名として使用します。

   たとえば、*ExampleSolution.py* という名前のファイルを *MyDemoRepo* という名前のリポジトリの *feature-randomizationfeature* という名前のブランチに加えます。最新のコミットの ID は *4c925148EXAMPLE* です。

```
aws codecommit put-file --repository-name MyDemoRepo --branch-name feature-
randomizationfeature --file-content file://MyDirectory/ExampleSolution.py --file-
path /solutions/ExampleSolution.py --parent-commit-id 4c925148EXAMPLE --name "María
 García" --email "maría_garcía@example.com" --commit-message "I added a third
 randomization routine."
```

> **Note**
>
> バイナリファイルを追加する場合、`fileb://` を使用してファイルのローカルな場所を指定することを確認します。

成功すると、このコマンドは以下のような出力を返します。

```
{
   "blobId": "2eb4af3bEXAMPLE",
   "commitId": "317f8570EXAMPLE",
   "treeId": "347a3408EXAMPLE"
}
```

ファイルを追加する (Git)

ローカルリポジトリでファイルを追加し、CodeCommit リポジトリに変更をプッシュできます。詳細については、「Git と の開始方法 AWS CodeCommit」を参照してください。

AWS CodeCommit リポジトリ内のファイルの内容を編集する

CodeCommit コンソール AWS CLI、または Git クライアントを使用して、CodeCommit リポジトリ内のファイルの内容を編集できます。

トピック

- ファイルを編集する (コンソール)
- ファイルの編集または削除する (AWS CLI)
- ファイルを編集する (Git)

ファイルを編集する (コンソール)

CodeCommit コンソールを使用して、CodeCommit リポジトリ内のブランチに追加されたファイルを編集できます。ファイルの編集の一環として、ユーザー名と E メールアドレスを提供できます。コミットメッセージを追加できるため、他のユーザーが変更者と変更理由を理解できます。

リポジトリ内のファイルを編集するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] で、ファイルを編集するレポジトリを選択します。

3. [コード] ビューで、ファイルを編集するブランチを選択します。デフォルトでは、[コード] ビューを開くとデフォルトのブランチの内容が表示されます。

   表示を別のブランチに変更するには、表示セレクタボタンを選択します。ドロップダウンリストからブランチ名を選択するか、またはフィルタボックスにブランチの名前を入力し、リストから選択します。

4. ブランチ内を移動して、編集するファイルを選択します。ファイルビューで [編集] を選択します。

   > **Note**
   >
   > バイナリファイルを選択すると、内容の表示を確認する警告メッセージが表示されます。バイナリファイルの編集には、CodeCommit コンソールを使用しないでください。

5. ファイルを編集し、他のユーザーに変更者と変更の理由についての情報を提供します。

   - [作成者名] に名前を入力します。この名前は、コミット情報で作成者名およびコミット名の両方として使用できます。CodeCommit は、デフォルトで IAM ユーザー名あるいはコンソールログインの派生を作成者名として使用します。
   - [Email address] (E メールアドレス) に E メールアドレスを入力し、他のリポジトリユーザーがこの変更について連絡できるようにします。
   - [コミットメッセージ] に、変更に関する短い説明を入力します。

6. [変更のコミット] を選択して変更をファイルに保存し、リポジトリにこの変更をコミットします。

ファイルの編集または削除する (AWS CLI)

AWS CLI および put-file コマンドを使用して、CodeCommit リポジトリ内のファイルに変更を加えることができます。元の場所と異なる場所に変更したファイルを保存する場合には、put-file コマンドを使用して変更したファイルにディレクトリまたはパス構文を追加することもできます。完全にファイルを削除するには、delete-file コマンドを使用できます。

> **Note**
>
> CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

リポジトリ内のファイルを編集するには

1. ファイルのローカルコピーを使用して、CodeCommit リポジトリに追加する変更を加えます。
2. ターミナルまたはコマンドラインで、put-file コマンドを実行し、次を指定します。
   - 編集したファイルを追加するリポジトリ。
   - 編集したファイルを追加するブランチ。
   - ブランチに作成された最新のコミットの完全なコミット ID (ヒントあるいはヘッドコミットとも呼ばれます)。
   - ファイルのローカルの場所。
   - ある場合には、更新したファイルがリポジトリで保存されるパスを含む、追加する更新ファイル名。
   - このファイル変更に関連付けるユーザー名および E メール。
   - 加えた変更についての説明をするコミットメッセージ。

   ユーザー名、E メールアドレス、コミットメッセージはオプションですが、他のユーザーに変更者と変更の理由について理解してもらうために便利です。ユーザー名を指定しない場合、CodeCommit はデフォルトで IAM ユーザー名あるいはコンソールログインの派生を使用します。

   たとえば、*ExampleSolution.py* という名前のファイルの編集を *MyDemoRepo* という名前のリポジトリの *feature-randomizationfeature* という名前のブランチに加えます。最新のコミットの ID は *4c925148EXAMPLE* です。

```
aws codecommit put-file --repository-name MyDemoRepo --branch-name feature-
randomizationfeature --file-content file://MyDirectory/ExampleSolution.py --file-
path /solutions/ExampleSolution.py --parent-commit-id 4c925148EXAMPLE --name "María
 García" --email "maría_garcía@example.com" --commit-message "I fixed the bug Mary
 found."
```

> **Note**
>
> 変更したバイナリファイルを追加するには、`--file-content` を表記した **`fileb`**`://MyDirectory/MyFile.raw` を必ず使用してください。

成功すると、このコマンドは以下のような出力を返します。

```
{
   "blobId": "2eb4af3bEXAMPLE",
   "commitId": "317f8570EXAMPLE",
   "treeId": "347a3408EXAMPLE"
}
```

ファイルを削除するには、delete-file コマンドを使用します。例えば、*MyDemoRepo* というリポジトリ名の最新のコミット ID (*c5709475EXAMPLE*) で、*main* というブランチの *README.md* という名前のファイルを削除するには、次のようにします。

```
aws codecommit delete-file --repository-name MyDemoRepo --branch-name main --file-
path README.md --parent-commit-id c5709475EXAMPLE
```

成功すると、このコマンドは以下のような出力を返します。

```
{
  "blobId":"559b44fEXAMPLE",
  "commitId":"353cf655EXAMPLE",
  "filePath":"README.md",
  "treeId":"6bc824cEXAMPLE"
}
```

ファイルを編集する (Git)

ローカルリポジトリでファイルを編集し、CodeCommit リポジトリに変更をプッシュできます。詳細については、「Git と の開始方法 AWS CodeCommit」を参照してください。

AWS CodeCommit リポジトリのプルリクエストを操作する

プルリクエストは、お客様と他のリポジトリーユーザーが、ブランチから別のブランチへのコード変更を確認、コメント、およびマージすることができる主な方法です。プルリクエストを使用すると、わずかな変更や修正、主要な機能の追加、リリースされたソフトウェアの新しいバージョンのコード変更を共同で確認することができます。プルリクエストの考えられるワークフローを以下に示します。

MyDemoRepo というリポジトリで作業するデベロッパーの Li Juan は、製品の今後のバージョンの新機能を開発したいと考えています。作業を本稼働環境に使用できるコードから分離するために、彼女はブランチをデフォルトブランチから作成し、それを `feature-randomizationfeature` と命名します。コードを書き込み、コミットし、このブランチに新機能コードをプッシュします。変更をデフォルトブランチにマージする前に、他のリポジトリーユーザーに、品質のためにコードを確認してほしいと考えます。これを行うために、プルリクエストを作成します。プルリクエストには、作業ブランチと、変更をマージする予定のコードのブランチ (この場合はデフォルトブランチ) との比較が含まれています。また、プルリクエストを承認するために指定した数のユーザーをリクエストする承認ルールを作成することもできます。また、ユーザーの承認プールを指定することもできます。他のユーザーは、自分のコードと変更内容を確認し、コメントや提案を追加します。コメントに応じてコードの変更を加えて作業ブランチを何度も更新するかもしれません。彼女の変更は、CodeCommit のブランチにプッシュされるたびにプルリクエストに組み込まれます。プルリクエストが開かれている間に、意図した送信先ブランチに加えられた変更を組み込むこともできるため、ユーザーは提案されたすべての変更をコンテキストで確実に確認できます。レビュー担当者が満足し、承認ルールの条件 (存在する場合) が満たされると、レビュー担当者またはレビュー担当者の 1 人がコードをマージして、プルリクエストをクローズします。

プルリクエストには 2 つのブランチが必要です。レビューするコードを含む送信元ブランチと、レビュー済みのコードをマージする送信先ブランチです。送信元ブランチには、AFTER コミットが含まれています。これは、送信先ブランチにマージする変更が含まれるコミットです。送信先ブランチには、BEFORE コミットが含まれています。これは、コードの「前」の状態を表しています (プルリクエストブランチが送信先ブランチにマージされる前)。マージ戦略の選択は、CodeCommit コンソール内の送信元ブランチと送信先ブランチ間でどのようにコミットがマージされるかについての詳細に影響します。CodeCommit 内でのマージ戦略の詳細については、プルリクエストをマージする (コンソール) を参照してください。

プルリクエストには、プルリクエストの作成時に送信元ブランチの先端と送信先ブランチの最新コミット間の相違点が表示されるため、ユーザーは変更を表示してコメントを追加できます。送信元ブランチへの変更をコミットしてプッシュすることによって、コメントに応じてプルリクエストに更新できます。

コードが確認され、承認ルールの要件（ある場合）が満たされたら、次のいずれかの方法でプルリクエストをクローズすることができます。

- ブランチをローカルでマージし、変更をプッシュします。これにより、早送りマージ戦略が使用され、マージの競合がない場合、リクエストは自動的に閉じられます。
- AWS CodeCommit コンソールを使用して、マージせずにプルリクエストを閉じたり、マージの競合を解決したり、競合がない場合は、利用可能なマージ戦略のいずれかを使用してブランチを閉じてマージしたりできます。
- AWS CLIを使用します。

プルリクエストを作成する前に、次の操作を実行します。

- レビュー対象であるコードの変更をブランチ (送信元ブランチ) にコミットしてプッシュしたことを確認する。
- 他のユーザーにプルリクエストとその変更について通知できるようにリポジトリの通知を設定する (このステップはオプションですが推奨されます)。
- 承認ルールテンプレートを作成してリポジトリに関連付けます。これにより、コードの品質を確保するためにプルリクエストに対して承認ルールが自動的に作成されます。詳細については、「承認ルールテンプレートの操作」を参照してください。

プルリクエストは、アマゾン ウェブ サービスアカウントのリポジトリユーザー用に IAM ユーザーを設定するとさらに効果的です。どのユーザーがどのコメントを行ったかを簡単に識別できます。もう 1 つの利点は、IAM ユーザーがリポジトリへのアクセスに Git 認証情報を使用できることです。詳細については、「ステップ 1: CodeCommit の初期設定」を参照してください。フェデレーティッドアクセスユーザーを含む他の種類のユーザーとのプルリクエストを使用できます。

CodeCommit でリポジトリの他の部分を操作する方法については、 リポジトリを操作する、承認ルールテンプレートの操作 、ファイルの操作、コミットの操作、ブランチの操作、および ユーザー設定の操作 を参照してください。

トピック

- プルリクエストの作成
- プルリクエストの承認ルールを作成する
- AWS CodeCommit リポジトリのプルリクエストを表示する
- プルリクエストのレビュー
- プルリクエストの更新
- プルリクエストの承認ルールを編集または削除する

- プルリクエストの承認ルールの上書き
- AWS CodeCommit リポジトリにプルリクエストをマージする
- AWS CodeCommit リポジトリのプルリクエストの競合を解決する
- AWS CodeCommit リポジトリのプルリクエストをクローズする

プルリクエストの作成

プルリクエストを作成すると、他のユーザーがコード変更を他のブランチにマージする前にそのコードの変更を確認するのに役立ちます。まず、コード変更のためのブランチを作成します。これは、プルリクエストのソースブランチとして参照されます。変更をコミットしてリポジトリにプッシュすると、そのブランチ (送信元ブランチ) の内容と、プルリクエストがクローズされた後の変更をマージするブランチ (送信先ブランチ) とを比較するプルリクエストを作成できます。

AWS CodeCommit コンソールまたは を使用して AWS CLI 、リポジトリのプルリクエストを作成できます。

トピック

- プルリクエストを作成する (コンソール)
- プルリクエストを作成する (AWS CLI)

プルリクエストを作成する (コンソール)

CodeCommit コンソールを使用して、CodeCommit リポジトリにプルリクエストを作成できます。リポジトリに通知が設定されている場合は、プルリクエストを作成すると、登録ユーザーに E メールが送信されます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリで、プルリクエストを作成するリポジトリの名前を選択します。
3. ナビゲーションペインで、[Pull Requests (プルリクエスト)] を選択します。

> **Tip**
> また、プルリクエストは、[Branches] や [Code] から作成することもできます。

4. [Create pull request] を選択します。

5. [Create pull request] の [Source] で、レビューする変更が含まれるブランチを選択します。
6. [送信先] で、プルリクエストがクローズされた際にコード変更をマージするブランチを確認します。
7. [Compare] を選択します。比較は 2 つのブランチで実行され、それらの違いが表示されます。また、プルリクエストがクローズされたときに 2 つのブランチが自動的にマージできるかどうかを判断するための分析も実行されます。
8. 比較の詳細と変更を確認し、プルリクエストにレビューする変更とコミットが含まれていることを確かめます。含まれていない場合、送信元と送信先のブランチを選択してから、再度 [Compare] を選択します。
9. 比較結果に問題がなければ、[Title] に、このレビューの説明を示すタイトルを入力します。これは、リポジトリのプルリクエストのリストに表示されるタイトルです。
10. (オプション) [Description (説明)] に、このレビューに関する詳細やレビューアにとって有益な情報を入力します。
11. [Create] を選択します。

プルリクエストは、リポジトリのプルリクエストのリストに表示されます。通知を設定した場合は、Amazon SNS トピックの受信者に新しく作成されたプルリクエストに関する情報が E メールで送信されます。

プルリクエストを作成する (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

を使用して CodeCommit リポジトリにプルリクエスト AWS CLI を作成するには

1. 次のように指定して create-pull-request コマンドを実行します。
   - プルリクエストの名前 (--title オプションを指定)。
   - プルリクエストの説明 (--description オプションを指定)。
   - create-pull-request コマンドのターゲットのリストは次の通りです。
     - プルリクエストが作成される CodeCommit リポジトリの名前 (repositoryName 属性付き)。

- ソースブランチとも呼ばれるレビューするコードの変更を含むブランチの名前 (sourceReference 属性を使用)。
- (オプション) デフォルトのブランチにマージしたくない場合は送信先ブランチとも呼ばれるコード変更をマージする予定のブランチの名前 (destinationReference 属性を使用)。

- 冪等性の一意、クライアントで生成されたトークン (--client-request-token オプションを指定)。

この例では、*###########*という名前のプルリクエストを作成し、*jane-branch* ソースブランチを対象とする*####################*という説明を追加します。プルリクエストは、MyDemoRepo という名前の CodeCommit リポジトリのデフォルトのブランチ *main* にマージされます。

```
aws codecommit create-pull-request --title "Pronunciation difficulty analyzer"
 --description "Please review these changes by Tuesday" --client-request-token
 123Example --targets repositoryName=MyDemoRepo,sourceReference=jane-branch
```

2. このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approver-rule-for-main",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "originApprovalRuleTemplate": {
                    "approvalRuleTemplateId": "dd3d22fe-EXAMPLE",
                    "approvalRuleTemplateName": "2-approver-rule-for-main"
                },
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::111111111111:user/Jane_Doe",
        "description": "Please review these changes by Tuesday",
```

```
        "title": "Pronunciation difficulty analyzer",
        "pullRequestTargets": [
            {
                "destinationCommit": "5d036259EXAMPLE",
                "destinationReference": "refs/heads/main",
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "317f8570EXAMPLE",
                "sourceReference": "refs/heads/jane-branch",
                "mergeMetadata": {
                    "isMerged": false
                }
            }
        ],
        "lastActivityDate": 1508962823.285,
        "pullRequestId": "42",
        "clientRequestToken": "123Example",
        "pullRequestStatus": "OPEN",
        "creationDate": 1508962823.285
    }
}
```

プルリクエストの承認ルールを作成する

プルリクエストの承認ルールを作成すると、コードを送信先ブランチにマージする前にプルリクエストの承認をユーザーにリクエストできるため、コードの品質が保証されます。プルリクエストを承認する必要があるユーザーの数を指定できます。ルールのユーザーの承認プールを指定することもできます。その場合、それらのユーザーからの承認のみがルールに必要な承認数にカウントされます。

> **Note**
>
> また、承認ルールテンプレートを作成することもできます。これにより、リポジトリ間であらゆるプルリクエストに適用される承認ルールの作成を自動化できます。詳細については、「承認ルールテンプレートの操作」を参照してください。

AWS CodeCommit コンソールまたは を使用して AWS CLI 、リポジトリの承認ルールを作成できます。

トピック

- プルリクエストの承認ルールを作成する (コンソール)

- プルリクエストの承認ルールを作成する (AWS CLI)

プルリクエストの承認ルールを作成する (コンソール)

CodeCommit コンソールを使用して、CodeCommit リポジトリ内のプルリクエストの承認ルールを作成できます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Repositories (リポジトリ)] で、プルリクエストの承認ルールを作成するリポジトリの名前を選択します。
3. ナビゲーションペインで、[Pull Requests (プルリクエスト)] を選択します。
4. 承認ルールを作成するプルリクエストをリストから選択します。オープンプルリクエストの承認ルールのみを作成できます。

5. プルリクエストで、[Approvals (承認)] を選択し、[Create approval rule (承認ルールの作成)] を選択します。
6. [Rule name (ルール名)] で、ルールにわかりやすい名前を付けます。たとえば、プルリクエストをマージする前に 2 人のユーザーにプルリクエストの承認をリクエストする場合は、ルールに **`Require two approvals before merge`** という名前を付けます。

> **Note**
>
> 承認ルールの作成後に名前を変更することはできません。

[Number of approvals needed (必要な承認の数)] に、必要な数値を入力します。デフォルトは 1 です。

7. （オプション）プルリクエストの承認を特定のユーザーグループからリクエストする場合は、[Approval rule members (承認ルールのメンバー)] で [Add (追加)] を選択します。[Approver type (承認者のタイプ)] で、次のいずれかを選択します。

   - [IAM user name or assumed role] (IAM ユーザー名または引き受けたロール): このオプションでは、サインインに使用したアカウントが AWS アカウント ID に事前入力され、名前のみが必要です。指定された名前と名前が一致する IAM ユーザーおよびフェデレーティッドアクセスユーザーの両方に使用できます。これは非常に強力なオプションで、柔軟性が大きく高まります。例えば、Amazon Web Services アカウント 123456789012「」でサインインし、このオプションを選択し、 を指定した場合**Mary_Major**、以下のすべてがそのユーザーからの承認としてカウントされます。

     - アカウントの IAM ユーザー (`arn:aws:iam::123456789012:user/Mary_Major`)
     - Mary_Major として IAM で識別されるフェデレーティッドユーザー (`arn:aws:sts::123456789012:federated-user/Mary_Major`)

     このオプションは、ワイルドカード (**CodeCommitReview**) を指定しない限り、ロールセッション名が Mary_Major (`arn:aws:sts::123456789012:assumed-role/CodeCommitReview/Mary_Major`) の `*Mary_Major` のロールを引き受けるユーザーのアクティブなセッションを認識しません。ロール名を明示的に指定することもできます (`CodeCommitReview/Mary_Major`)。

- [Fully qualified ARN] (完全修飾 ARN): このオプションでは、IAM ユーザーまたはロールの完全修飾 Amazon リソースネーム (ARN) を指定できます。このオプションは、 AWS Lambda や などの他の AWS のサービスで使用される引き受けたロールもサポートします AWS CodeBuild。委任ロールの ARN 形式は、ロールの場合は `arn:aws:sts::AccountID:assumed-role/RoleName`、関数の場合は `arn:aws:sts::AccountID:assumed-role/FunctionName` です。

承認者のタイプとして [IAM user name or assumed role] (IAM ユーザー名または引き受けたロール) を選択した場合は、[Value] (値) に、IAM ユーザーまたはロールの名前またはユーザーかロールの完全修飾 ARN を入力します。承認が必要な承認の数にカウントされるすべてのユーザーまたはロールを追加するまで、[Add (追加)] を再度選択してユーザーまたはロールを追加します。

どちらの承認者タイプでも、値にワイルドカード (*) を使用できます。例えば、[IAM user name or assumed role] (IAM ユーザー名または引き受けたロール) オプションを選択し、**`CodeCommitReview/*`** を指定した場合、**`CodeCommitReview`** のロールを引き受けるすべてのユーザーが承認プールにカウントされます。個々のロールセッション名は、必要な承認者数にカウントされます。このようにして、Mary_Major と Li_Juan は、サインインして `CodeCommitReview` のロールを引き受けるときに承認としてカウントされます。IAM ARN、ワイルドカード、および形式の詳細については、IAM 識別子を参照してください。

> **Note**
> 承認ルールは、クロスアカウント承認をサポートしていません。

8. 承認ルールの設定が完了したら、[Submit (送信)] を選択します。

プルリクエストの承認ルールを作成する (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

CodeCommit リポジトリのプルリクエストの承認ルールを作成するには

1. 次のように指定して create-pull-request-approval-rule コマンドを実行します。
   - プルリクエストの ID (--id オプションを指定)。
   - 承認ルールの名前 (--approval-rule-name オプション付き) 。

- 承認ルールの内容 (--approval-rule-content オプション付き) 。

承認ルールを作成するときに、次のいずれかの方法で承認プールの承認者を指定できます。

- CodeCommitApprovers: このオプションでは、アマゾン ウェブ サービスアカウントとリソースのみが必要です。指定されたリソース名と一致する名前を持つ IAM ユーザーとフェデレーティッドアクセスユーザーの両方に使用できます。これは非常に強力なオプションで、柔軟性が大きく高まります。例えば、アマゾン ウェブ サービスアカウント 123456789012 と **Mary_Major** を指定した場合、次のすべてがそのユーザーからの承認としてカウントされます。
  - アカウントの IAM ユーザー (`arn:aws:iam::123456789012:user/Mary_Major`)
  - Mary_Major として IAM で識別されるフェデレーティッドユーザー (`arn:aws:sts::123456789012:federated-user/Mary_Major`)

  このオプションは、ワイルドカード (**CodeCommitReview**) を指定しない限り、ロールセッション名が Mary_Major (`arn:aws:sts::123456789012:assumed-role/CodeCommitReview/Mary_Major`) の `*Mary_Major` のロールを引き受けるユーザーのアクティブなセッションを認識しません。
- [Fully qualified ARN] (完全修飾 ARN): このオプションでは、IAM ユーザーまたはロールの完全修飾 Amazon リソースネーム (ARN) を指定できます。

IAM ARN、ワイルドカード、および形式の詳細については、IAM 識別子を参照してください。

次の例では、ID が 27 のプルリクエストの `Require two approved approvers` という名前の承認ルールを作成します。ルールは、承認プールから 2 つの承認が必要であることを指定します。プールには、CodeCommit にアクセスし、123456789012 アマゾン ウェブ サービスアカウントで **CodeCommitReview** のロールを引き受けるすべてのユーザーが含まれます。また、同じ アマゾン ウェブ サービスアカウントに IAM ユーザーまたは `Nikhil_Jayashankar` という名前のフェデレーティッドユーザーも含まれます。

```
aws codecommit create-pull-request-approval-rule --pull-request-id 27
 --approval-rule-name "Require two approved approvers" --approval-
rule-content "{\"Version\": \"2018-11-08\",\"Statements\": [{\"Type\":
 \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers
\": [\"CodeCommitApprovers:123456789012:Nikhil_Jayashankar\",
 \"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}"
```

2. このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "approvalRule": {
        "approvalRuleName": "Require two approved approvers",
        "lastModifiedDate": 1570752871.932,
        "ruleContentSha256": "7c44e6ebEXAMPLE",
        "creationDate": 1570752871.932,
        "approvalRuleId": "aac33506-EXAMPLE",
        "approvalRuleContent": "{\"Version\": \"2018-11-08\",\"Statements\":
 [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers
\": [\"CodeCommitApprovers:123456789012:Nikhil_Jayashankar\",
 \"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major"
    }
}
```

AWS CodeCommit リポジトリのプルリクエストを表示する

AWS CodeCommit コンソールまたは を使用して AWS CLI 、リポジトリのプルリクエストを表示できます。デフォルトでは、開いているリクエストのみ表示されますが、フィルターを変更して、すべてのプルリクエスト、閉じているリクエストのみ、作成したプルリクエストのみなどを表示できます。

トピック

- プルリクエストを表示する (コンソール)
- プルリクエストを表示する (AWS CLI)

プルリクエストを表示する (コンソール)

AWS CodeCommit コンソールを使用して、CodeCommit リポジトリのプルリクエストのリストを表示できます。フィルターを変更することで、特定のプルリクエストのセットのみをリスト表示するよう変更できます。たとえば、作成したプルリクエストでステータスが [Open] のものを一覧表示したり、別のフィルターを選択して、作成したプルリクエストでステータスが [Closed] のものを表示したりできます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。

2. [Repositories (リポジトリー)] で、プルリクエストを表示するリポジトリーの名前を選択します。
3. ナビゲーションペインで、[Pull Requests (プルリクエスト)] を選択します。
4. デフォルトでは、すべてのオープンプルリクエストが一覧表示されます。

5. 表示フィルターを変更するには、使用可能なフィルターのリストから選択します。
   - [Open pull requests] (デフォルト): ステータスが [Open] のすべてのプルリクエストが表示されます。
   - [All pull requests]: すべてのプルリクエストが表示されます。
   - [Closed pull requests]: ステータスが [Closed] のすべてのプルリクエストが表示されます。
   - [My pull requests (自分用のプルリクエスト)]: 作成したプルリクエストがステータスに関係なくすべて表示されます。自分がコメントした、または参加した評価は表示されません。
   - [My open pull requests (自分用の開いているプルリクエスト)]: 作成したプルリクエストでステータスが [Open] のものがすべて表示されます。
   - [My closed pull requests (自分用の閉じているプルリクエスト)]: 作成したプルリクエストでステータスが [Closed] のものがすべて表示されます。
6. 表示されたリストに確認したいプルリクエストを見つけたら、それを選択します。

プルリクエストを表示する (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

を使用して CodeCommit リポジトリでプルリクエスト AWS CLI を表示するには、次の手順に従います。

1. リポジトリーのプルリクエストを一覧表示するには、list-pull-requests コマンドを実行し、以下を指定します。

   - プルリクエストを表示する CodeCommit リポジトリの名前 (--repository-name オプション)。
   - (オプション) プルリクエストのステータス (--pull-request-status オプションで指定)。
   - (オプション) プルリクエストを作成した IAM ユーザーの Amazon リソースネーム (ARN) (--author-arn オプション)。
   - (オプション) 結果のバッチを返すために使用できる列挙トークン (--next-token オプションで指定)。
   - (オプション) リクエストごとに返される結果の上限数 (--max-results オプションで指定)。

   例えば、MyDemoRepo という名前の CodeCommit リポジトリにあり、ARN が *arn:aws:iam::111111111111:user/Li_Juan* の IAM ユーザーが作成したプルリクエストでステータスが *CLOSED* のものをリスト表示するには、以下を実行します。

   ```
   aws codecommit list-pull-requests --author-arn arn:aws:iam::111111111111:user/
   Li_Juan --pull-request-status CLOSED --repository-name MyDemoRepo
   ```

   このコマンドが正常に実行されると、次のような出力が生成されます。

   ```
   {
      "nextToken": "",
      "pullRequestIds": ["2","12","16","22","23","35","30","39","47"]
   }
   ```

   プルリクエスト ID は、新しいアクティビティの順序で表示されます。

2. プルリクエストの詳細を表示するには、get-pull-request コマンドを実行します。この場合、--pull-request-id オプションでプルリクエストの ID を指定します。たとえば、ID *27* のプルリクエストの詳細を確認するには、以下を実行します。

   ```
   aws codecommit get-pull-request --pull-request-id 27
   ```

   このコマンドが正常に実行されると、次のような出力が生成されます。

   ```
   {
       "pullRequest": {
           "approvalRules": [
   ```

```
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approver-rule-for-main",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "lastActivityDate": 1562619583.565,
        "pullRequestTargets": [
            {
                "sourceCommit": "ca45e279EXAMPLE",
                "sourceReference": "refs/heads/bugfix-1234",
                "mergeBase": "a99f5ddbEXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": false
                },
                "destinationCommit": "2abfc6beEXAMPLE",
                "repositoryName": "MyDemoRepo"
            }
        ],
        "revisionId": "e47def21EXAMPLE",
        "title": "Quick fix for bug 1234",
        "authorArn": "arn:aws:iam::123456789012:user/Nikhil_Jayashankar",
        "clientRequestToken": "d8d7612e-EXAMPLE",
        "creationDate": 1562619583.565,
        "pullRequestId": "27",
        "pullRequestStatus": "OPEN"
    }
}
```

3. プルリクエストの承認を表示するには、次を指定して、get-pull-request-approval-state コマンドを実行します。

   - プルリクエストの ID（ --pull-request-id オプションを指定 ）。
   - プルリクエストのリビジョン ID（ --revision-id option) を使用。プルリクエストの現在のリビジョン ID を取得するには、get-pull-request コマンドを使用します。

たとえば、ID が *8*、リビジョン ID が *9f29d167EXAMPLE* のプルリクエストの承認を表示するには、次のようにします。

```
aws codecommit get-pull-request-approval-state --pull-request-id 8 --revision-
id 9f29d167EXAMPLE
```

このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "approvals": [
        {
            "userArn": "arn:aws:iam::123456789012:user/Mary_Major",
            "approvalState": "APPROVE"
        }
    ]
}
```

4. プルリクエストのイベントを表示するには、describe-pull-request-events コマンドを実行します。この場合、--pull-request-id オプションでプルリクエストの ID を指定します。たとえば、ID *8* のプルリクエストのイベントを確認するには、以下を実行します。

```
aws codecommit describe-pull-request-events --pull-request-id 8
```

このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "pullRequestEvents": [
        {
            "pullRequestId": "8",
            "pullRequestEventType": "PULL_REQUEST_CREATED",
            "eventDate": 1510341779.53,
            "actor": "arn:aws:iam::111111111111:user/Zhang_Wei"
        },
        {
            "pullRequestStatusChangedEventMetadata": {
                "pullRequestStatus": "CLOSED"
            },
            "pullRequestId": "8",
            "pullRequestEventType": "PULL_REQUEST_STATUS_CHANGED",
```

```
            "eventDate": 1510341930.72,
            "actor": "arn:aws:iam::111111111111:user/Jane_Doe"
        }
    ]
}
```

5. プルリクエストにマージの競合があるかどうかを確認するには、get-merge-conflicts コマンドを実行します。この場合、以下を指定します。

   - CodeCommit リポジトリの名前 (--repository-name オプション)。
   - マージの評価で使用する変更の送信元のブランチ、タグ、HEAD、他の完全修飾参照 (--source-commit-specifier オプションで指定)。
   - マージの評価で使用する変更の送信先のブランチ、タグ、HEAD、他の完全修飾参照 (--destination-commit-specifier オプションで指定)。
   - 使用するマージオプション (--merge-option オプションで指定)

   例えば、*my-feature-branch* という名前のソースブランチの先端と、MyDemoRepo という名前のリポジトリ内の *main* という名前の送信先ブランチの間にマージの競合があるかどうかを確認します。

```
aws codecommit get-merge-conflicts --repository-name MyDemoRepo --source-commit-
specifier my-feature-branch --destination-commit-specifier main --merge-option
 FAST_FORWARD_MERGE
```

   成功すると、このコマンドは以下のような出力を返します。

```
{
    "destinationCommitId": "fac04518EXAMPLE",
    "mergeable": false,
    "sourceCommitId": "16d097f03EXAMPLE"
}
```

プルリクエストのレビュー

AWS CodeCommit コンソールを使用して、プルリクエストに含まれる変更を確認できます。コメントは、リクエスト、ファイル、個別のコード行に追加できます。他のユーザーのコメントに返信することもできます。リポジトリが configured with notifications と設定されている場合、ユーザーがコメ

ントに返信するとき、またはプルリクエストにユーザーがコメントするときに E メールで通知が送信されます。

を使用してプルリクエストに AWS CLI コメントし、コメントに返信できます。変更を確認するには、CodeCommit コンソール、git diff コマンドまたは diff ツールを使用する必要があります。

トピック

- プルリクエストを確認する (コンソール)
- プルリクエストを確認する (AWS CLI)

プルリクエストを確認する (コンソール)

CodeCommit コンソールを使用して、CodeCommit リポジトリ内のプルリクエストを確認できます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリで、リポジトリの名前を選択します。
3. ナビゲーションペインで、[プルリクエスト] を選択します。
4. デフォルトでは、すべてのオープンプルリクエストが一覧表示されます。確認したいオープンプルリクエストを選択します。

> **Note**
>
> クローズまたはマージされたプルリクエストにコメントすることはできますが、マージまたは再度開くことはできません。

5. プルリクエストで、[Changes] を選択します。
6. 次のいずれかを行ってください。

   - プルリクエスト全体に全般的なコメントを追加するには、[Comments on changes (変更に関するコメント)] や [New comment (新規コメント)] にコメントを入力して、[Save (保存)] を選択します。マークダウンを使用するか、プレーンテキスト形式でコメントを入力します。

   - コミット内のファイルにコメントを追加するには、[Changes] でファイルの名前を見つけます。ファイル名の横に表示されるコメント用アイコン ( ) を選択し、コメントを入力して [保存] を選択します。

   - プルリクエスト内の変更された行にコメントを追加するには、[Changes] でコメントする行に移動します。その行の横に表示されるコメント用アイコン

( )

を選択し、コメントを入力して [保存] を選択します。

7. コミットのコメントに返信するには、[Changes] または [Activity] で、[Reply] を選択します。テキストと絵文字を使用して返信できます。

特定の絵文字リアクションを使用して応答したユーザーの名前を、その絵文字リアクションを選択して表示できます。すべての絵文字リアクションと、特定の絵文字を使用して応答したユーザーに関する情報を表示するには、[View all reactions (すべてのリアクションを表示)」を選択し

ます。コメントに絵文字で応答した場合、その応答は絵文字リアクションボタンのアイコンに表示されます。

> **Note**
>
> コンソールに表示されるリアクション数は、ページが読み込まれた時点で正確です。絵文字のリアクション数に関する最新情報を表示するには、ページを更新するか、[View all reactions (すべてのリアクションを表示)] を選択します。

8. (オプション) Amazon CodeGuru Reviewer によって作成された推奨事項に返信し、推奨事項の品質に関するフィードバックを提供する場合は、[Reply] (返信) を選択します。リアクションボタンを使用して、推奨事項を承認するかどうかに関する一般的な情報を提供します。コメントフィールドを使用して、リアクションの詳細を入力します。

> **Note**
>
> Amazon CodeGuru Reviewer は、プログラム分析と機械学習を使用して一般的な問題を検出し、Java または Python コードにおける修正点を提案する自動化されたコードレビューサービスです。
>
> - Amazon CodeGuru Reviewer コメントが表示されるのは、リポジトリを Amazon CodeGuru Reviewer に関連付けている場合、分析が完了している場合、およびプルリクエストのコードが Java または Python コードである場合のみです。詳細について

は、「AWS CodeCommit リポジトリを Amazon CodeGuru Reviewer に関連付ける、または関連付けを解除する」を参照してください。

- Amazon CodeGuru Reviewer コメントが [Changes] (変更) タブに表示されるのは、最新リビジョンのプルリクエストに対してコメントを作成した場合のみです。[アクティビティ] タブには、コメントが常に表示されます。
- Amazon CodeGuru Reviewer の推奨事項に対する応答には使用可能な絵文字リアクションのいずれでも使用できますが、推奨事項の有用性を評価する場合は、親指を上げた手と親指を下げた手の絵文字アクションのみを使用します。

9. プルリクエストで行われた変更を承認するには、[Approve (承認)] を選択します。

> **Note**
>
> 自分で作成したプルリクエストを自分で承認することはできません。

承認、プルリクエストの承認ルール、および承認ルールテンプレートによって作成された承認ルールを [Approvals (承認)] で表示できます。プルリクエストを承認しない場合は、[Revoke approval (承認の取り消し)] を選択できます。

> **Note**
>
> 承認または取り消すことができるのは、オープンプルリクエストの承認のみです。ステータスが「マージ済」または「クローズ」のプルリクエストでは、承認を承認または取り消すことはできません。

プルリクエストを確認する (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

プルリクエストは、次の AWS CLI コマンドを使用して確認できます。

- post-comment-for-pull-request (プルリクエストにコメントを追加する)
- get-comments-for-pull-request (プルリクエストに残されたコメントを表示する)
- update-pull-request-approval-state (プルリクエストを承認するか承認を取り消す)
- post-comment-reply (プルリクエスト内のコメントに返信する)

次のコマンドを使用して、プルリクエストでコメントに絵文字を付けることもできます。

- 絵文字を使用してコメントに返信するには、put-comment-reaction を実行します。
- コメントへの絵文字リアクションを表示するには、get-comment-reactions を実行します。

を使用して CodeCommit リポジトリのプルリクエスト AWS CLI を確認するには

1. リポジトリのプルリクエストにコメントを追加するには、post-comment-for-pull-request コマンドを実行し、以下を指定します。

   - プルリクエストの ID (--pull-request-id オプションを指定)。
   - プルリクエストが含まれているリポジトリの名前 (--repository-name オプションで指定)。
   - プルリクエストのマージ先である送信先ブランチのコミットの完全なコミット ID (--before-commit-id オプションで指定)。
   - コメントの投稿時にプルリクエストでブランチの現在の先端となっている送信元ブランチのコミットの完全なコミット ID (--after-commit-id オプションで指定)。
   - 冪等性の一意、クライアントで生成されたトークン (--client-request-token オプションを指定)。
   - コメントのコンテンツ (--content オプションを指定)。
   - コメントの配置場所に関する場所情報のリストには、以下を含みます。
     - 拡張子やサブディレクトリなど、比較されているファイルの名前 (ある場合) (filePath 属性を指定)。
     - 比較ファイル内の変更の行番号 (filePosition 属性を指定)。
     - 送信元ブランチと送信先ブランチの比較で、変更のコメントが「前」か「後」か (relativeFileVersion 属性を指定)。

   たとえば、このコマンドを使用して、*################################* というコメントを、リポジトリ名が *MyDemoRepo* で、プルリクエスト ID が *47*、ファイルが *ahs_count.py* の変更に追加する場合。

```
aws codecommit post-comment-for-pull-request --pull-request-id "47" --
repository-name MyDemoRepo --before-commit-id 317f8570EXAMPLE --after-
commit-id 5d036259EXAMPLE --client-request-token 123Example --content
 "These don't appear to be used anywhere. Can we remove them?" --location
 filePath=ahs_count.py,filePosition=367,relativeFileVersion=AFTER
```

   このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
         "afterBlobId": "1f330709EXAMPLE",
         "afterCommitId": "5d036259EXAMPLE",
         "beforeBlobId": "80906a4cEXAMPLE",
         "beforeCommitId": "317f8570EXAMPLE",
         "comment": {
               "authorArn": "arn:aws:iam::111111111111:user/Saanvi_Sarkar",
               "clientRequestToken": "123Example",
               "commentId": "abcd1234EXAMPLEb5678efgh",
               "content": "These don't appear to be used anywhere. Can we remove
 them?",
               "creationDate": 1508369622.123,
               "deleted": false,
               "lastModifiedDate": 1508369622.123,
               "callerReactions": [],
               "reactionCounts": []
            }
             "location": {
               "filePath": "ahs_count.py",
               "filePosition": 367,
               "relativeFileVersion": "AFTER"
             },
         "repositoryName": "MyDemoRepo",
         "pullRequestId": "47"
 }
```

2. プルリクエストのコメントを表示するには、get-comments-for-pull-request コマンドを実行し、以下を指定します。

   - CodeCommit リポジトリの名前 (`--repository-name` オプション)。
   - プルリクエストのシステム生成の ID （ `--pull-request-id` オプション付き ）。
   - (オプション) 結果の次のバッチを返す列挙トークン (`--next-token` オプションを使用)。
   - (オプション) 返される結果の数を制限する負でない整数 (`--max-results` オプションを使用)。

   たとえば、ID が 42 のプルリクエストのコメントを表示するには、このコマンドを使用します。

```
aws codecommit get-comments-for-pull-request --pull-request-id 42
```

このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
   "commentsForPullRequestData": [
      {
         "afterBlobId": "1f330709EXAMPLE",
         "afterCommitId": "5d036259EXAMPLE",
         "beforeBlobId": "80906a4cEXAMPLE",
         "beforeCommitId": "317f8570EXAMPLE",
         "comments": [
            {
               "authorArn": "arn:aws:iam::111111111111:user/Saanvi_Sarkar",
               "clientRequestToken": "",
               "commentId": "abcd1234EXAMPLEb5678efgh",
               "content": "These don't appear to be used anywhere. Can we remove
 them?",
               "creationDate": 1508369622.123,
               "deleted": false,
               "lastModifiedDate": 1508369622.123,
               "callerReactions": [],
               "reactionCounts":
                {
                  "THUMBSUP" : 6,
                  "CONFUSED" : 1
                }
            },
            {
               "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
               "clientRequestToken": "",
               "commentId": "442b498bEXAMPLE5756813",
               "content": "Good catch. I'll remove them.",
               "creationDate": 1508369829.104,
               "deleted": false,
               "lastModifiedDate": 150836912.273,
               "callerReactions": ["THUMBSUP"]
               "reactionCounts":
                {
                  "THUMBSUP" : 14
                }
             }
         ],
         "location": {
            "filePath": "ahs_count.py",
```

```
            "filePosition": 367,
            "relativeFileVersion": "AFTER"
        },
        "repositoryName": "MyDemoRepo",
        "pullRequestId": "42"
    }
  ],
  "nextToken": "exampleToken"
}
```

3. プルリクエストを承認するか取り消すには、次を指定して、update-pull-request-approval-state コマンドを実行します。

   - プルリクエストの ID （ --pull-request-id オプションを指定 ）。
   - プルリクエストのリビジョン ID （ --revision-id option) を使用。プルリクエストの現在のリビジョン ID を取得するには、get-pull-request コマンドを使用します。
   - 適用する承認状態 (--approval-state を使用) 。有効な承認状態には、APPROVE および REVOKE があります。

   たとえば、ID が *27* で、リビジョン ID が *9f29d167EXAMPLE* のプルリクエストを承認するには、このコマンドを使用します。

```
aws codecommit update-pull-request-approval-state --pull-request-id 27 --revision-
id 9f29d167EXAMPLE --approval-state "APPROVE"
```

   成功すると、このコマンドは何も返しません。

4. プルリクエストのコメントに返答するには、post-comment-reply コマンドを実行して、次のように指定します。

   - 返信したいコメントのシステム生成 ID (--in-reply-to オプションを指定)。
   - 冪等性の一意、クライアントで生成されたトークン (--client-request-token オプションを指定)。
   - 返答のコンテンツ (--content オプションを指定)。

   たとえば、このコマンドを使用して、*##################* という返答を、システムによって生成された ID が *abcd1234EXAMPLEb5678efgh* のコメントに追加する場合。

```
aws codecommit post-comment-reply --in-reply-to abcd1234EXAMPLEb5678efgh --
content "Good catch. I'll remove them." --client-request-token 123Example
```

このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "comment": {
        "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
        "clientRequestToken": "123Example",
        "commentId": "442b498bEXAMPLE5756813",
        "content": "Good catch. I'll remove them.",
        "creationDate": 1508369829.136,
        "deleted": false,
        "lastModifiedDate": 150836912.221,
        "callerReactions": [],
        "reactionCounts": []
    }
 }
```

プルリクエストの更新

オープンプルリクエストのソースブランチにコミットをプッシュすることによって、追加のコード変更でプルリクエストを更新することができます。詳細については、「でコミットを作成する AWS CodeCommit」を参照してください。

AWS CodeCommit コンソールまたは を使用して AWS CLI 、プルリクエストのタイトルまたは説明を更新できます。プルリクエストのタイトルまたは説明の更新は、以下の状況の場合に実行するとよいでしょう。

- 他のユーザーが説明を理解できないか、元のタイトルが誤解を招く可能性がある。
- オープンプルリクエストの送信元ブランチに加えた変更が反映されたタイトルまたは説明にしたい。

プルリクエストを更新する (コンソール)

CodeCommit コンソールを使用して、CodeCommit リポジトリ内のプルリクエストのタイトルと説明を更新できます。プルリクエストのコードを更新するには、オープンプルリクエストのソースブランチにコミットをプッシュします。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] で、プルリクエストを更新するリポジトリの名前を選択します。
3. ナビゲーションペインで、[プルリクエスト] を選択します。
4. デフォルトでは、すべてのオープンプルリクエストが一覧表示されます。更新したいオープンプルリクエストを選択します。
5. プルリクエストで、[詳細]、[詳細の編集] の順に選択し、タイトルまたは説明を編集します。

> **Note**
>
> 閉じているプルリクエストやマージされたプルリクエストのタイトルまたは説明を更新することはできません。

プルリクエストを更新する (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

また、次のコマンドにも興味があるかもしれません。

- update-pull-request-approval-state で、プルリクエストを承認するか承認を取り消します。
- create-pull-request-approval-rule で、プルリクエストの承認ルールを作成します。
- delete-pull-request-approval-rule で、プルリクエストの承認ルールを削除します。
- を使用してコミットを作成する AWS CLI または Git クライアントを使用してコミットを作成する で、追加のコード変更を作成し、オープンプルリクエストのソースブランチにプッシュします。

を使用して CodeCommit リポジトリのプルリクエスト AWS CLI を更新するには

1. リポジトリのプルリクエストのタイトルを更新するには、update-pull-request-title コマンドを実行し、以下を指定します。

- プルリクエストの ID (--pull-request-id オプションを指定)。
- プルリクエストのタイトル (--title オプションで指定)。

たとえば、ID *47* のプルリクエストのタイトルを更新するには、以下を実行します。

```
aws codecommit update-pull-request-title --pull-request-id 47 --title
 "Consolidation of global variables - updated review"
```

このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approver-rule-for-main",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "originApprovalRuleTemplate": {
                    "approvalRuleTemplateId": "dd8b26gr-EXAMPLE",
                    "approvalRuleTemplateName": "2-approver-rule-for-main"
                },
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.12,
        "description": "Review the latest changes and updates to the global
 variables. I have updated this request with some changes, including removing some
 unused variables.",
        "lastActivityDate": 1508372657.188,
        "pullRequestId": "47",
        "pullRequestStatus": "OPEN",
        "pullRequestTargets": [
            {
```

```
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": false,
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
            }
        ],
        "title": "Consolidation of global variables - updated review"
    }
}
```

2. プルリクエストの説明を更新するには、update-pull-request-description コマンドを実行し、以下を指定します。

   - プルリクエストの ID (--pull-request-id オプションを指定)。
   - 説明 (--description オプションで指定)。

   たとえば、ID *47* のプルリクエストの説明を更新するには、以下を実行します。

```
aws codecommit update-pull-request-description --pull-request-id 47 --description
 "Updated the pull request to remove unused global variable."
```

   このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
   "pullRequest": {
      "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
      "clientRequestToken": "",
      "creationDate": 1508530823.155,
      "description": "Updated the pull request to remove unused global variable.",
      "lastActivityDate": 1508372423.204,
      "pullRequestId": "47",
      "pullRequestStatus": "OPEN",
      "pullRequestTargets": [
         {
            "destinationCommit": "9f31c968EXAMPLE",
            "destinationReference": "refs/heads/main",
            "mergeMetadata": {
```

```
                "isMerged": false,
            },
            "repositoryName": "MyDemoRepo",
            "sourceCommit": "99132ab0EXAMPLE",
            "sourceReference": "refs/heads/variables-branch"
        }
    ],
    "title": "Consolidation of global variables"
  }
}
```

プルリクエストの承認ルールを編集または削除する

プルリクエストに承認ルールがある場合、その条件が満たされるまでプルリクエストをマージできません。プルリクエストの承認ルールを変更して、その条件を容易に満たしたり、レビューの厳しさを高めることができます。プルリクエストを承認する必要があるユーザーの数を変更できます。また、ルールのユーザーの承認プールのメンバーシップを追加、削除、または変更することもできます。最後に、プルリクエストに承認ルールを使用しなくなった場合は、それを削除できます。

> **Note**
>
> プルリクエストの承認ルールを上書きすることもできます。詳細については、「プルリクエストの承認ルールの上書き」を参照してください。

AWS CodeCommit コンソールまたは を使用して AWS CLI 、リポジトリの承認ルールを編集および削除できます。

トピック

- プルリクエストの承認ルールを編集または削をする (コンソール)
- プルリクエストの承認ルールを編集または削除する (AWS CLI)

プルリクエストの承認ルールを編集または削をする (コンソール)

CodeCommit コンソールを使用して、CodeCommit リポジトリ内のプルリクエストの承認ルールを編集または削除できます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] で、プルリクエストの承認ルールを編集または削除するリポジトリの名前を選択します。
3. ナビゲーションペインで、[Pull Requests (プルリクエスト)] を選択します。
4. 承認ルールを編集または削除するプルリクエストを選択します。オープンプルリクエストの承認ルールのみを編集および削除できます。

5. プルリクエストで [Approvals (承認)] を選択し、編集または削除するルールをリストから選択します。次のいずれかを行ってください。
   - ルールを編集するには、[Edit (編集)] を選択します。
   - ルールを削除する場合は、[Delete (削除)] を選択し、指示に従ってルールの削除を確認します。
6. [Edit approval rule (承認ルールの編集)] で、ルールに対して必要な変更を行い、[Submit (送信)] を選択します。

7. 承認ルールの設定が完了したら、[Submit (送信)] を選択します。

プルリクエストの承認ルールを編集または削除する (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

を使用して AWS CLI 、承認ルールの内容を編集し、承認ルールを削除できます。

> **Note**
>
> また、次のコマンドにも興味があるかもしれません。

- update-pull-request-approval-state で、プルリクエストを承認するか承認を取り消します。
- get-pull-request-approval-states で、プルリクエストの承認を表示します。
- evaluate-pull-request-approval-rules で、プルリクエストの承認ルールが条件を満たしているかどうかを判断します。

を使用して CodeCommit リポジトリのプルリクエストの承認ルール AWS CLI を編集または削除するには

1. 承認ルールを編集するには、update-pull-request-approval-rule-content コマンドを実行して、次を指定します。

   - プルリクエストの ID (--id オプションを指定)。
   - 承認ルールの名前 (--approval-rule-name オプション付き) 。
   - 承認ルールの内容 (--approval-rule-content オプション付き) 。

   次の使用例は、ID が *27* のプルリクエストに対して *2 ##########*という名前の承認ルールを更新します。このルールでは、*123456789012* アマゾン ウェブ サービスアカウントのすべての IAM ユーザーを含む承認プールから 1 人のユーザー承認が必要です。

```
aws codecommit update-pull-request-approval-rule-content --pull-request-id 27
 --approval-rule-name "Require two approved approvers" --approval-rule-content
 "{Version: 2018-11-08, Statements: [{Type: \"Approvers\", NumberOfApprovalsNeeded:
 1, ApprovalPoolMembers:[\"CodeCommitApprovers:123456789012:user/*\"]}]}}"
```

2. このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "approvalRule": {
        "approvalRuleContent": "{Version: 2018-11-08, Statements:
 [{Type: \"Approvers\", NumberOfApprovalsNeeded: 1, ApprovalPoolMembers:
[\"CodeCommitApprovers:123456789012:user/*\"]}]}}",
        "approvalRuleId": "aac33506-EXAMPLE",
        "originApprovalRuleTemplate": {},
        "creationDate": 1570752871.932,
        "lastModifiedDate": 1570754058.333,
```

```
        "approvalRuleName": Require two approved approvers",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
        "ruleContentSha256": "cd93921cEXAMPLE",
    }
}
```

3. 承認ルールを削除するには、次を指定して、delete-pull-request-approval-rule コマンドを実行します。

   - プルリクエストの ID (--id オプションを指定)。
   - 承認ルールの名前 (--approval-rule-name オプション付き) 。

   たとえば、ID が *15* のプルリクエストの *#######*という名前の承認ルールを削除するには、次のようにします。

```
aws codecommit delete-pull-request-approval-rule --pull-request-id 15 --approval-
rule-name "My Approval Rule"
```

   成功すると、このコマンドは以下のような出力を返します。

```
{
    "approvalRuleId": "077d8e8a8-EXAMPLE"
}
```

プルリクエストの承認ルールの上書き

通常の開発過程では、プルリクエストをマージする前に、ユーザーが承認ルールの条件を満たすようにします。ただし、プルリクエストのマージを迅速化する必要がある場合があります。たとえば、バグ修正を本番稼働環境に配置したいが、承認プールの誰もプルリクエストを承認できないとします。このような場合は、プルリクエストの承認ルールを上書きすることを選択できます。プルリクエスト用に特別に作成され、承認ルールテンプレートから生成されたものを含む、プルリクエストのすべての承認ルールを上書きできます。特定の承認ルールを選択的に上書きすることはできません。すべてのルールだけです。ルールを上書きして承認ルールの要件を確保したら、プルリクエストを送信先ブランチにマージできます。

プルリクエストの承認ルールを上書きすると、ルールを上書きしたユーザーに関する情報が、プルリクエストのアクティビティに記録されます。このようにして、プルリクエストの履歴に戻り、誰が

ルールを上書きしたかを確認できます。プルリクエストがまだ開いている場合は、上書きを取り消すこともできます。プルリクエストがマージされると、上書きを取り消すことができなくなります。

トピック

- 承認ルールを上書きする (コンソール)
- 承認ルールを上書きする (AWS CLI)

承認ルールを上書きする (コンソール)

プルリクエストの確認の一環として、コンソールでプルリクエストの承認ルールの要件を上書きできます。気が変わったら、上書きを取り消すことができ、承認ルールの要件が再適用されます。プルリクエストが開いている場合にのみ、承認ルールを上書きするか、上書きを取り消すことができます。マージまたはクローズしている場合は、上書き状態を変更できません。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリで、リポジトリの名前を選択します。
3. ナビゲーションペインで、[プルリクエスト] を選択します。承認ルールの要件を上書きするプルリクエストを選択するか、上書きを取り消します。
4. [Approvals (承認)] タブで、[Override approval rules (承認ルールの上書き)] を選択します。要件は確保され、ボタンのテキストは [Revoke override (上書きを取り消し)] に変わります。承認ルールの要件を再適用するには、[Revoke override (上書きの取り消し)] を選択します。

承認ルールを上書きする (AWS CLI)

を使用して、承認ルールの要件 AWS CLI を上書きできます。プルリクエストの上書きステータスを表示する場合にも使用できます。

プルリクエストの承認ルール要件を上書きするには

1. ターミナルまたはコマンドラインで、次を指定して、override-pull-request-approval-rules コマンドを実行します。
   - プルリクエストのシステム生成 ID。
   - プルリクエストの最新リビジョン ID。この情報を表示するには、get-pull-request を使用します。

- 上書きするステータス、OVERRIDE または REVOKE。REVOKE ステータスによって OVERRIDE ステータスが削除されますが、保存されません。

たとえば、ID が、**34** で、リビジョン ID が **927df8d8EXAMPLE** であるプルリクエストの承認ルールを上書きするには、次のようにします。

```
aws codecommit override-pull-request-approval-rules --pull-request-id 34 --
revision-id 927df8d8dEXAMPLE --override-status OVERRIDE
```

2. 成功した場合、このコマンドは何も返しません。
3. ID が、**34** で、リビジョン ID が **927df8d8EXAMPLE** であるプルリクエストの上書きを取り消すには、次のようにします。

```
aws codecommit override-pull-request-approval-rules --pull-request-id 34 --
revision-id 927df8d8dEXAMPLE --override-status REVOKE
```

プルリクエストの上書きステータスに関する情報を取得するには

1. ターミナルまたはコマンドラインで、次を指定して、get-pull-request-override-state コマンドを実行します。

   - プルリクエストのシステム生成 ID。
   - プルリクエストの最新リビジョン ID。この情報を表示するには、get-pull-request を使用します。

   たとえば、ID が、**34** で、リビジョン ID が **927df8d8EXAMPLE** であるプルリクエストの上書き状態を表示するには、次のようにします。

```
aws codecommit get-pull-request-override-state --pull-request-id 34 --revision-
id 927df8d8dEXAMPLE
```

2. このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "overridden": true,
    "overrider": "arn:aws:iam::123456789012:user/Mary_Major"
}
```

AWS CodeCommit リポジトリにプルリクエストをマージする

コードがレビューされ、プルリクエストのすべての承認ルール (存在する場合) が満たされたら、次のいずれかの方法でプルリクエストをマージできます。

- コンソールを使用して、利用できるいずれかのマージ戦略を使用して送信元ブランチを送信先ブランチにマージすることができます。また、これはプルリクエストを終了できます。また、コンソールでマージの競合を解決することもできます。プルリクエストがマージ可能かどうか、または競合を解決する必要があるかどうかを示すメッセージがコンソールに表示されます。すべての競合が解決され、[Merge (マージ)] を選択すると、選択したマージ戦略を使用してマージが実行されます。早送りは、Git のデフォルトのオプションであるデフォルトのマージ戦略です。送信元ブランチと送信先ブランチ内のコードの状態によってはこの戦略を利用できないことがありますが、スカッシュや 3 方向などの他のオプションを使用できる可能性があります。
- を使用して AWS CLI 、早送り、スカッシュ、または 3 方向マージ戦略を使用してプルリクエストをマージおよびクローズできます。
- ローカルコンピュータで、git merge コマンドを使用して、ソースブランチを送信先ブランチにマージしてから、マージしたコードを送信先ブランチにプッシュすることができます。このアプローチには、慎重に検討する必要がある欠点があります。プルリクエストの承認ルールの要件が満たされているかどうかにかかわらず、プルリクエストをマージし、これらのコントロールを回避します。送信先ブランチをマージしてプッシュすると、早送りマージ戦略を利用してプルリクエストをマージする場合に、プルリクエストを自動的にクローズします。この方法の利点の 1 つは、git merge コマンドで、CodeCommit コンソールで使用できないマージオプションまたは戦略を選択できることです。git merge とマージオプションの詳細については、「git-merge」または Git ドキュメントを参照してください。

CodeCommit は、プルリクエストの送信元ブランチまたは送信先ブランチのいずれかが削除された場合、プルリクエストを自動的にクローズします。

トピック

- プルリクエストをマージする (コンソール)
- プルリクエストをマージする (AWS CLI)

プルリクエストをマージする (コンソール)

CodeCommit コンソールを使用して、CodeCommit リポジトリにプルリクエストをマージできます。プルリクエストの状態が [Merged (マージ済み)] に変更されると、これはオープンプルリクエストのリストに表示されなくなります。マージされたプルリクエストはクローズ済みに分類されます。これを [Open (オープン)] に戻すことはできませんが、ユーザーはこの変更に対して引き続きコメントでき、コメントに対して返信できます。プルリクエストがマージまたはクローズされると、そのプルリクエストを承認、承認取り消し、プルリクエストに適用された承認ルールを上書きすることはできません。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリで、リポジトリの名前を選択します。
3. ナビゲーションペインで、[プルリクエスト] を選択します。
4. デフォルトでは、すべてのオープンプルリクエストが一覧表示されます。マージするオープンプルリクエストを選択します。
5. プルリクエストで、[Approvals (承認)] を選択します。承認者のリストを確認し、すべての承認ルール (存在する場合) が条件を満たしていることを確認します。1 つ以上の承認ルールのステータスが [Rule not satisfied (ルールが満たされていません)] の場合、プルリクエストをマージできません。プルリクエストを誰も承認していない場合は、マージするか、承認を待機するかを検討します。

> **Note**
>
> プルリクエストの承認ルールが作成されている場合は、編集または削除してマージのブロックを解除できます。承認ルールテンプレートを使用して作成された承認ルールは、編集または削除できません。要件を上書きすることのみを選択できます。詳細については、「プルリクエストの承認ルールの上書き」を参照してください。

6. [Merge (マージ)] を選択します。
7. プルリクエストで、使用可能なマージ方法のいずれかを選択します。適用できないマージ戦略は、グレー表示されます。利用可能なマージ戦略がない場合、CodeCommit コンソールで競合を手動で解決するか、または Git クライアントをローカルで使用して解決することができます。詳細については、「AWS CodeCommit リポジトリのプルリクエストの競合を解決する」を参照してください。

- 早送りマージは、送信元ブランチの最新コミットに送信先のブランチへのリファレンスを転送します。これは、利用可能な場合の Git のデフォルトの動作です。マージコミットは作成されませんが、送信元ブランチからのすべてのコミット履歴は送信先ブランチで発生していたかのように保持されます。早送りマージは、マージコミットが作成されないため、送信先ブランチの履歴のコミットビジュアライザービューにブランチマージとして表示されません。送信元ブランチのヒントが送信先ブランチのヒントに早送りされます。
- スカッシュマージによって送信元ブランチ内の変更を含む 1 つのコミットが作成され、この単一のスカッシュ済みコミットが送信先ブランチに適用されます。デフォルトでは、このス

カッシュコミットへのコミットメッセージには送信元ブランチ内の変更におけるすべてのコミットメッセージが含まれています。ブランチの変更における個別のコミット履歴は一切保持されません。これによってリポジトリ履歴の簡素化に役立ち、また、送信先ブランチの履歴のコミットビューではマージのグラフ表示が保持されます。

- 3 方向マージは送信先ブランチのマージにマージコミットを作成しますが、送信元ブランチで行われた個別のコミットも送信先ブランチの履歴の一部として保持します。これは、リポジトリへの変更の完全な履歴を保持することに役立ちます。

8. スカッシュあるいは 3 方向のマージ戦略を選択する場合、自動生成されるコミットメッセージを確認して、情報を変更する必要に応じてこれを編集します。コミット履歴に名前および E メールアドレスを追加します。
9. （オプション）必要に応じて、送信元ブランチをマージの一部として削除するオプションの選択を解除します。デフォルトでは、プルリクエストがマージされたときにソースブランチを削除します。
10. [Merge pull request (プルリクエストのマージ)] を選択してマージを完了します。

プルリクエストをマージする (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

を使用して CodeCommit リポジトリのプルリクエストを AWS CLI マージするには

1. プルリクエストのすべての承認ルールが満たされ、マージする準備ができているかどうかを評価するには、以下を指定して、evaluate-pull-request-approval-rules コマンドを実行します。

   - プルリクエストの ID（--pull-request-id オプションを指定）。
   - プルリクエストのリビジョン ID（--revision-id option) を使用。プルリクエストの現在のリビジョン ID を取得するには、get-pull-request コマンドを使用します。

   たとえば、ID が *27* で、リビジョン ID が *9f29d167EXAMPLE* を持つプルリクエストの承認ルールの状態を評価するには、次のようにします。

   ```
   aws codecommit evaluate-pull-request-approval-rules --pull-request-id 27 --
   revision-id 9f29d167EXAMPLE
   ```

   このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "evaluation": {
        "approved": false,
        "approvalRulesNotSatisfied": [
            "Require two approved approvers"
        ],
        "overridden": false,
        "approvalRulesSatisfied": []
    }
}
```

> **Note**
>
> この出力は、承認ルールの要件が満たされていないため、プルリクエストがマージ可能でないことを示します。このプルリクエストをマージするには、ルールの条件を満たすためにレビュー担当者に承認してもらいます。アクセス許可とルールの作成方法によっては、ルールを編集、上書き、または削除できる場合もあります。詳細については、「プルリクエストのレビュー」、「プルリクエストの承認ルールの上書き」、および「プルリクエストの承認ルールを編集または削除する」を参照してください。

2. 早送りマージ戦略を使用してプルリクエストをマージしてクローズするには、次を指定して merge-pull-request-by-fast-forward コマンドを実行します。

   - プルリクエストの ID (--pull-request-id オプションを指定)。
   - ソースブランチのチップの完全なコミット ID (--source-commit-id オプションを指定)。
   - レポジトリの名前 (--repository-name オプションを指定)。

   例えば、*MyDemoRepo* という名前のリポジトリで、ID が *47*、ソースコミット ID が *99132ab0EXAMPLE* のプルリクエストをマージしてクローズするには、以下の操作を行います。

```
aws codecommit merge-pull-request-by-fast-forward --pull-request-id 47 --source-
commit-id 99132ab0EXAMPLE --repository-name MyDemoRepo
```

   このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
```

```
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "I want one approver for this pull request",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.142,
        "description": "Review the latest changes and updates to the global
 variables",
        "lastActivityDate": 1508887223.155,
        "pullRequestId": "47",
        "pullRequestStatus": "CLOSED",
        "pullRequestTargets": [
            {
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": true,
                    "mergedBy": "arn:aws:iam::123456789012:user/Mary_Major"
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
            }
        ],
        "title": "Consolidation of global variables"
    }
}
```

3. スカッシュマージ戦略を使用してプルリクエストをマージしてクローズするには、次を指定して merge-pull-request-by-squash コマンドを実行します。

   - プルリクエストの ID (--pull-request-id オプションを指定)。

- ソースブランチのチップの完全なコミット ID (--source-commit-id オプションを指定)。
- レポジトリの名前 (--repository-name オプションを指定)。
- 使用する競合の詳細のレベル (--conflict-detail-level オプションを指定)。指定しない場合、デフォルトの **FILE_LEVEL** が使用されます。
- 使用する競合の解決戦略 (--conflict-resolution-strategy オプションを指定)。指定しない場合、このデフォルトは NONE になり、競合を手動で解決する必要があります。
- 含めるコミットメッセージ (--commit-message オプションを指定)。
- コミットに使用する名前 (--author-name オプションを指定)。
- コミットに使用する E メールアドレス (--email オプションを指定)。
- 空のフォルダを保持するかどうか (--keep-empty-folders オプションを指定)。

次の例では、*MyDemoRepo* という名前のリポジトリで、ID が *47*、ソースコミット ID が *99132ab0EXAMPLE* のプルリクエストをマージしてクローズします。これは、LINE_LEVEL の競合の詳細と ACCEPT_SOURCE の競合解決戦略を使用します

```
aws codecommit merge-pull-request-by-squash --pull-request-id 47 --source-commit-
id 99132ab0EXAMPLE --repository-name MyDemoRepo --conflict-detail-level LINE_LEVEL
 --conflict-resolution-strategy ACCEPT_SOURCE --author-name "Jorge Souza" --email
 "jorge_souza@example.com" --commit-message "Merging pull request 47 by squash and
 accepting source in merge conflicts"
```

正常に実行されると、このコマンドは早送りによるマージと同じ種類の出力を行います。出力は次のようになります。

```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approver-rule-for-main",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "originApprovalRuleTemplate": {
```

```
                    "approvalRuleTemplateId": "dd8b17fe-EXAMPLE",
                    "approvalRuleTemplateName": "2-approver-rule-for-main"
                },
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.142,
        "description": "Review the latest changes and updates to the global
 variables",
        "lastActivityDate": 1508887223.155,
        "pullRequestId": "47",
        "pullRequestStatus": "CLOSED",
        "pullRequestTargets": [
            {
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": true,
                    "mergedBy": "arn:aws:iam::123456789012:user/Mary_Major"
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
            }
        ],
        "title": "Consolidation of global variables"
    }
}
```

4. 3 方向のマージ戦略を使用してプルリクエストをマージしてクローズするには、次を指定して merge-pull-request-by-three-way コマンドを実行します。

   - プルリクエストの ID (--pull-request-id オプションを指定)。
   - ソースブランチのチップの完全なコミット ID (--source-commit-id オプションを指定)。
   - レポジトリの名前 (--repository-name オプションを指定)。
   - 使用する競合の詳細のレベル (--conflict-detail-level オプションを指定)。指定しない場合、デフォルトの **FILE_LEVEL** が使用されます。
   - 使用する競合の解決戦略 (--conflict-resolution-strategy オプションを指定)。指定しない場合、このデフォルトは NONE になり、競合を手動で解決する必要があります。

- 含めるコミットメッセージ (--commit-message オプションを指定)。
- コミットに使用する名前 (--author-name オプションを指定)。
- コミットに使用する E メールアドレス (--email オプションを指定)。
- 空のフォルダを保持するかどうか (--keep-empty-folders オプションを指定)。

次の例では、*MyDemoRepo* という名前のリポジトリで、ID が *47*、ソースコミット ID が *99132ab0EXAMPLE* のプルリクエストをマージしてクローズします。これは、競合の詳細と競合解決戦略のデフォルトのオプションを使用します。

```
aws codecommit merge-pull-request-by-three-way --pull-request-id 47 --source-
commit-id 99132ab0EXAMPLE --repository-name MyDemoRepo --author-name "Maria Garcia"
 --email "maria_garcia@example.com" --commit-message "Merging pull request 47 by
 three-way with default options"
```

正常に実行されると、このコマンドは早送りによるマージと同じ種類の次のような出力を行います。

```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approver-rule-for-main",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "originApprovalRuleTemplate": {
                    "approvalRuleTemplateId": "dd8b17fe-EXAMPLE",
                    "approvalRuleTemplateName": "2-approver-rule-for-main"
                },
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.142,
```

```
        "description": "Review the latest changes and updates to the global
 variables",
        "lastActivityDate": 1508887223.155,
        "pullRequestId": "47",
        "pullRequestStatus": "CLOSED",
        "pullRequestTargets": [
            {
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": true,
                    "mergedBy": "arn:aws:iam::123456789012:user/Mary_Major"
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
            }
        ],
        "title": "Consolidation of global variables"
    }
}
```

AWS CodeCommit リポジトリのプルリクエストの競合を解決する

プルリクエストに競合があるためにマージできない場合、つぎのいずれかの方法でこの競合の解決を試みることができます。

- ローカルコンピューターで、git diff コマンドを使用して 2 つのブランチ間の競合を見つけ、解決するために変更を加えます。また、別のツールや他のソフトウェアを使用して差異を見つけて解決を試みることもできます。満足できる解決方法を実行したら、解決した競合が含まれる変更がある送信元のブランチをプッシュして、プルリクエストを更新できます。git diff および git difftool についての詳細は、Git ドキュメントを参照してください。
- コンソールでは、[Resolve conflicts (競合の解決)] を選択できます。これにより、プレーンテキストエディタが開き、git diff コマンドに類似する方法で競合が表示されます。競合が含まれるファイルごとに手動で競合を確認し、変更を加えたら、変更されたものでプルリクエストを更新します。
- では AWS CLI、 を使用してマージの競合に関する情報 AWS CLI を取得し、参照されていないマージコミットを作成してマージをテストできます。

トピック

- プルリクエストの競合を解決する (コンソール)
- プルリクエスト内の競合を解決する (AWS CLI)

プルリクエストの競合を解決する (コンソール)

CodeCommit コンソールを使用して、CodeCommit リポジトリ内のプルリクエストにおける競合を解決できます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリで、リポジトリの名前を選択します。
3. ナビゲーションペインで、[プルリクエスト] を選択します。
4. デフォルトでは、すべてのオープンプルリクエストが一覧表示されます。マージしたいけれど、競合があるオープンプルリクエストを選択します。
5. プルリクエストで [Resolve conflicts] を選択します。このオプションは、プルリクエストがマージできるようになる前に競合を解決する必要がある場合のみ表示されます。

6. 解決する必要がある競合がある各ファイルをリスト化した競合解決ウィンドウが開きます。リストの各ファイルを選択して競合を確認し、すべての競合が解決されるまで必要な変更を加えます。

- 送信元のファイルコンテンツあるいは送信先のファイルコンテンツを使用するかを選択でき、また、ファイルがバイナリファイルではない場合には手動でファイルのコンテンツを編集して必要な変更のみが含まれるようにできます。スタンダード git diff マーカーは、ファイルの送信先ブランチ (HEAD) と送信元ブランチ間の競合を示すために使用されます。
- ファイルがバイナリファイル、Git サブモジュールの場合、またはファイルあるいはフォルダの名前の競合がある場合、競合を解決するために送信元のファイルあるいは送信先のファイルを使用するかを選択する必要があります。CodeCommit コンソールでバイナリファイルを表示あるいは編集することはできません。
- ファイルモードの競合がある場合、送信元ファイルのファイルモードと送信先ファイルのファイルモードのどちらかを選択することで、解決するためのオプションが表示されます。

- ファイルの変更を破棄して競合状態に戻すには、[Reset file (ファイルのリセット)] を選択します。これにより、異なる方法で競合を解決できます。

7. 変更に満足したら、[Update pull request (プルリクエストの更新)] を選択します。

> **Note**
>
> 変更を使用してプルリクエストを正常に更新する前に、すべてのファイルですべての競合を解決する必要があります。

8. プルリクエストは、変更を使用して更新され、マージ可能になります。マージページが表示されます。この時点でプルリクエストをマージするか、またはプルリクエストのリストに戻るかを選択できます。

プルリクエスト内の競合を解決する (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

プルリクエストの競合を解決し、そのリクエストをマージできる AWS CLI コマンドはありません。ただし、個別のコマンドを使用すると、競合の検出、これらの解決の試行、およびプルリクエストがマージ可能かどうかのテストを行うことができます。次を使用できます。

- get-merge-options を使用して、どのマージオプションが 2 つのコミット識別子間のマージに使用できるかを検出します。
- get-merge-conflicts を使用して、2 つのコミット識別子間のマージにおけるマージ競合があるファイルのリストを返します。
- batch-describe-merge-conflicts を使用して、指定するマージ戦略を使用した 2 つのコミット間のマージにおけるファイル内のすべてのマージ競合についての情報を取得します。
- describe-merge-conflicts を使用して、指定するマージ戦略を使用した 2 つのコミット間の特定のファイルにおけるマージ競合に関する詳細を取得します。
- create-unreferenced-merge-commit を使用して、指定するマージ戦略を使用した 2 つのコミット識別子をマージする結果をテストします。

1. 2 つのコミット識別子間でのマージに使用できるマージオプションを判断するためには、次を指定して get-merge-options コマンドを実行します。

- マージの送信元のコミット識別子 (--source-commit-specifier オプションを指定)。
- マージの送信先のコミット識別子 (--destination-commit-specifier オプションを指定)。
- レポジトリの名前 (--repository-name オプションを指定)。
- (オプション) 使用する競合解決戦略 (--conflict-resolution-strategy オプションを指定)。
- (オプション) すべての競合についての詳細度レベル (--conflict-detail-level オプションを指定)。

例えば、*bugfix-1234* という名前の送信元ブランチを *MyDemoRepo* という名前のリポジトリ内で *main* という名前の送信先ブランチにマージするために使用できるマージ戦略を判断するためには、次のようにします。

```
aws codecommit get-merge-options --source-commit-specifier bugfix-1234 --
destination-commit-specifier main --repository-name MyDemoRepo
```

このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "mergeOptions": [
        "FAST_FORWARD_MERGE",
        "SQUASH_MERGE",
        "THREE_WAY_MERGE"
    ],
    "sourceCommitId": "d49940adEXAMPLE",
    "destinationCommitId": "86958e0aEXAMPLE",
    "baseCommitId": "86958e0aEXAMPLE"
}
```

2. 2 つのコミット識別子間のマージのマージ競合が含まれるファイルのリストを取得するには、次を指定して get-merge-conflicts コマンドを実行します。

- マージの送信元のコミット識別子 (--source-commit-specifier オプションを指定)。
- マージの送信先のコミット識別子 (--destination-commit-specifier オプションを指定)。
- レポジトリの名前 (--repository-name オプションを指定)。
- 使用するマージオプション (--merge-option オプションを指定)。
- (オプション) すべての競合についての詳細度レベル (--conflict-detail-level オプションを指定)。

- (オプション) 使用する競合解決戦略 (--conflict-resolution-strategy オプションを指定)。
- (オプション) 返す競合があるファイルの最大数 (--max-conflict-files オプションを指定)。

例えば、MyDemoRepo という名前のリポジトリで 3 方向マージ戦略を使用した、feature-randomizationfeature という名前の送信元ブランチと main という名前の送信先ブランチ間のマージにおける競合があるファイルのリストを取得するには、次のようにします。

```
aws codecommit get-merge-conflicts --source-commit-specifier feature-
randomizationfeature --destination-commit-specifier main --merge-option
 THREE_WAY_MERGE --repository-name MyDemoRepo
```

このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "mergeable": false,
    "destinationCommitId": "86958e0aEXAMPLE",
    "sourceCommitId": "6ccd57fdEXAMPLE",
    "baseCommitId": "767b6958EXAMPLE",
    "conflictMetadataList": [
        {
            "filePath": "readme.md",
            "fileSizes": {
                "source": 139,
                "destination": 230,
                "base": 85
            },
            "fileModes": {
                "source": "NORMAL",
                "destination": "NORMAL",
                "base": "NORMAL"
            },
            "objectTypes": {
                "source": "FILE",
                "destination": "FILE",
                "base": "FILE"
            },
            "numberOfConflicts": 1,
            "isBinaryFile": {
                "source": false,
                "destination": false,
                "base": false
```

```
            },
            "contentConflict": true,
            "fileModeConflict": false,
            "objectTypeConflict": false,
            "mergeOperations": {
                "source": "M",
                "destination": "M"
            }
        }
    ]
}
```

3. 2 つのコミット識別子間のマージにおいて、すべてのファイルあるいはファイルのサブセットないのマージ競合についての情報を取得するには、batch-describe-merge-conflicts コマンドを実行します。

- マージの送信元のコミット識別子 (--source-commit-specifier オプションを指定)。
- マージの送信先のコミット識別子 (--destination-commit-specifier オプションを指定)。
- 使用するマージオプション (--merge-option オプションを指定)。
- レポジトリの名前 (--repository-name オプションを指定)。
- (オプション) 使用する競合解決戦略 (--conflict-resolution-strategy オプションを指定)。
- (オプション) すべての競合についての詳細度レベル (--conflict-detail-level オプションを指定)。
- (オプション) 返されるマージハンクの最大数 (--max-merge-hunks オプションを指定)。
- (オプション) 返す競合があるファイルの最大数 (--max-conflict-files オプションを指定)。
- (オプション) 競合を説明するために使用されるターゲットファイルのパス (--file-paths オプションを指定)。

例えば、*feature-randomizationfeature* という名前の送信元のブランチと *main* という名前の送信先ブランチを *MyDemoRepo* という名前のリポジトリで *THREE_WAY_MERGE* 戦略を使用してマージするときのマージ競合を判断するためには、次のようにします。

```
aws codecommit batch-describe-merge-conflicts --source-commit-specifier feature-
randomizationfeature --destination-commit-specifier main --merge-
option THREE_WAY_MERGE --repository-name MyDemoRepo
```

このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "conflicts": [
        {
            "conflictMetadata": {
                "filePath": "readme.md",
                "fileSizes": {
                    "source": 139,
                    "destination": 230,
                    "base": 85
                },
                "fileModes": {
                    "source": "NORMAL",
                    "destination": "NORMAL",
                    "base": "NORMAL"
                },
                "objectTypes": {
                    "source": "FILE",
                    "destination": "FILE",
                    "base": "FILE"
                },
                "numberOfConflicts": 1,
                "isBinaryFile": {
                    "source": false,
                    "destination": false,
                    "base": false
                },
                "contentConflict": true,
                "fileModeConflict": false,
                "objectTypeConflict": false,
                "mergeOperations": {
                    "source": "M",
                    "destination": "M"
                }
            },
            "mergeHunks": [
                {
                    "isConflict": true,
                    "source": {
                        "startLine": 0,
                        "endLine": 3,
                        "hunkContent": "VGhpcyBpEXAMPLE=="
                    },
                    "destination": {
```

```
                        "startLine": 0,
                        "endLine": 1,
                        "hunkContent": "VXNlIHRoEXAMPLE="
                    }
                }
            ]
        }
    ],
    "errors": [],
    "destinationCommitId": "86958e0aEXAMPLE",
    "sourceCommitId": "6ccd57fdEXAMPLE",
    "baseCommitId": "767b6958EXAMPLE"
}
```

4. 2 つのコミット識別子間のマージの特定のファイルにおけるマージ競合についての詳細を取得するには、次を指定して describe-merge-conflicts コマンドを実行します。

- マージの送信元のコミット識別子 (--source-commit-specifier オプションを指定)。
- マージの送信先のコミット識別子 (--destination-commit-specifier オプションを指定)。
- 使用するマージオプション (--merge-option オプションを指定)。
- 競合を説明するために使用するターゲットファイルのパス (--file-path オプションを指定)。
- レポジトリの名前 (--repository-name オプションを指定)。
- (オプション) 使用する競合解決戦略 (--conflict-resolution-strategy オプションを指定)。
- (オプション) すべての競合についての詳細度レベル (--conflict-detail-level オプションを指定)。
- (オプション) 返されるマージハンクの最大数 (--max-merge-hunks オプションを指定)。
- (オプション) 返す競合があるファイルの最大数 (--max-conflict-files オプションを指定)。

例えば、*readme.md* という名前のファイルで *feature-randomizationfeature* という名前の送信元ブランチと *main* という名前の送信先ブランチでの *MyDemoRepo* という名前のリポジトリ内の *THREE_WAY_MERGE* 戦略を使用したマージ競合を判断するには、次のようにします。

```
aws codecommit describe-merge-conflicts --source-commit-specifier feature-
randomizationfeature --destination-commit-specifier main --merge-
option THREE_WAY_MERGE --file-path readme.md --repository-name MyDemoRepo
```

このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "conflictMetadata": {
        "filePath": "readme.md",
        "fileSizes": {
            "source": 139,
            "destination": 230,
            "base": 85
        },
        "fileModes": {
            "source": "NORMAL",
            "destination": "NORMAL",
            "base": "NORMAL"
        },
        "objectTypes": {
            "source": "FILE",
            "destination": "FILE",
            "base": "FILE"
        },
        "numberOfConflicts": 1,
        "isBinaryFile": {
            "source": false,
            "destination": false,
            "base": false
        },
        "contentConflict": true,
        "fileModeConflict": false,
        "objectTypeConflict": false,
        "mergeOperations": {
            "source": "M",
            "destination": "M"
        }
    },
    "mergeHunks": [
        {
            "isConflict": true,
            "source": {
                "startLine": 0,
                "endLine": 3,
                "hunkContent": "VGhpcyBpEXAMPLE=="
            },
            "destination": {
```

```
                "startLine": 0,
                "endLine": 1,
                "hunkContent": "VXNlIHRoEXAMPLE="
            }
        }
    ],
    "destinationCommitId": "86958e0aEXAMPLE",
    "sourceCommitId": "6ccd57fdEXAMPLE",
    "baseCommitId": "767b69580EXAMPLE"
}
```

5. 2 つのコミット識別子をマージした結果を表す非参照のコミットを作成するには、次を指定して create-unreferenced-merge-commit コマンドを実行します。

- マージの送信元のコミット識別子 (--source-commit-specifier オプションを指定)。
- マージの送信先のコミット識別子 (--destination-commit-specifier オプションを指定)。
- 使用するマージオプション (--merge-option オプションを指定)。
- レポジトリの名前 (--repository-name オプションを指定)。
- (オプション) 使用する競合解決戦略 (--conflict-resolution-strategy オプションを指定)。
- (オプション) すべての競合についての詳細度レベル (--conflict-detail-level オプションを指定)。
- (オプション) 含めるコミットメッセージ (--commit-message オプションを指定)。
- (オプション) コミットに使用する名前 (--name オプションを指定)。
- (オプション) コミットに使用する E メールアドレス (--email オプションを指定)。
- (オプション) 空のフォルダを保持するかどうか (--keep-empty-folders オプションを指定)。

例えば、*bugfix-1234* という名前の送信元のブランチと *main* という名前の送信先ブランチを *MyDemoRepo* という名前のリポジトリで ACCEPT_SOURCE 戦略を使用してマージするときのマージ競合を判断するためには、次のようにします。

```
aws codecommit create-unreferenced-merge-commit --source-commit-
specifier bugfix-1234 --destination-commit-specifier main --merge-option
 THREE_WAY_MERGE --repository-name MyDemoRepo --name "Maria Garcia" --email
 "maria_garcia@example.com" --commit-message "Testing the results of this merge."
```

このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "commitId": "4f178133EXAMPLE",
    "treeId": "389765daEXAMPLE"
}
```

AWS CodeCommit リポジトリのプルリクエストをクローズする

コードをマージせずにプルリクエストを閉じる場合、次のいずれかの方法を使用できます。

- コンソールでは、コードをマージせずにプルリクエストをクローズできます。これを行うのは、上記のように、git merge コマンドを使用してブランチを手動でマージする場合や、プルリクエストソースブランチのコードが送信先ブランチにマージするコードではない場合です。
- プルリクエストで指定されたソースブランチを削除できます。CodeCommit は、プルリクエストの送信元ブランチまたは送信先ブランチのいずれかが削除された場合、プルリクエストを自動的にクローズします。
- では AWS CLI、プルリクエストのステータスを から `OPEN`に更新できます`CLOSED`。これによって、コードをマージせずにプルリクエストがクローズされます。

トピック

- プルリクエストをクローズする (コンソール)
- プルリクエストをクローズする (AWS CLI)

プルリクエストをクローズする (コンソール)

CodeCommit コンソールを使用して、CodeCommit リポジトリ内のプルリクエストを閉じることができます。プルリクエストのステータスが Closed に変更されると、それを Open に戻すことはできませんが、ユーザーは変更についてコメントしたり、コメントに返信することができます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリで、リポジトリの名前を選択します。
3. ナビゲーションペインで、[プルリクエスト] を選択します。
4. デフォルトでは、すべてのオープンプルリクエストが一覧表示されます。解決済みにする未解決のプルリクエストを選択します。

5. プルリクエストで、[Close pull request (プルリクエストのクローズ)] を選択します。このオプションは、ソースブランチを送信先ブランチにマージすることなくプルリクエストをクローズします。このオプションでは、プルリクエストをクローズする際にソースブランチを削除する方法は提供していませんが、リクエストがクローズされた後にソースブランチを自分で削除することができます。

プルリクエストをクローズする (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

を使用して CodeCommit リポジトリのプルリクエスト AWS CLI を閉じるには

- リポジトリ内のプルリクエストのステータスを `OPEN` から `CLOSED` に更新するには、次のように指定して update-pull-request-status コマンドを実行します。
  - プルリクエストの ID (--pull-request-id オプションを指定)。
  - プルリクエストのステータス (--pull-request-status オプションを指定)。

  例えば、ID *42* のプルリクエストのステータスを `MyDemoRepo` という名前の CodeCommit リポジトリで *`CLOSED`* に更新するには、以下の操作を行います。

  ```
  aws codecommit update-pull-request-status --pull-request-id 42 --pull-request-
  status CLOSED
  ```

  このコマンドが正常に実行されると、次のような出力が生成されます。

  ```
  {
  ```

```
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approvers-needed-for-this-change",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.165,
        "description": "Updated the pull request to remove unused global
 variable.",
        "lastActivityDate": 1508372423.12,
        "pullRequestId": "47",
        "pullRequestStatus": "CLOSED",
        "pullRequestTargets": [
            {
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": false,
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
            }
        ],
        "title": "Consolidation of global variables"
    }
}
```

承認ルールテンプレートの操作

プルリクエストの承認ルールを作成できます。リポジトリで作成されたプルリクエストの一部またはすべてに承認ルールを自動的に適用するには、承認ルールテンプレートを使用します。承認ルールテンプレートを使用すると、異なるブランチが適切なレベルの承認とコントロールを持つように、リポジトリ間で開発ワークフローをカスタマイズできます。本番稼働用ブランチと開発ブランチに異なるルールを定義できます。これらのルールは、ルール条件に一致するプルリクエストが作成されるたびに適用されます。承認ルールテンプレートのマネージドポリシーと許可の詳細については、「承認ルールテンプレートに対するアクションのアクセス許可 および AWS CodeCommit の マネージドポリシー」を参照してください。

承認ルールテンプレートは、 AWS リージョン 作成された の 1 つ以上のリポジトリに関連付けることができます。テンプレートがリポジトリに関連付けられている場合、プルリクエストの作成の一環として、そのリポジトリのプルリクエストの承認ルールが自動的に作成されます。単一の承認ルールと同様に、承認ルールテンプレートは承認ルールの構造を定義します。これには、必要な承認の数や、承認が必要なオプションのユーザープールが含まれます。承認ルールとは異なり、送信先リファレンス (ブランチ) を定義することもできます。ブランチフィルターとも呼ばれます。送信先リファレンスを定義する場合、テンプレートの指定されたブランチ名 (送信先リファレンス) と一致する送信先ブランチ名を持つプルリクエストにのみルールが作成されます。例えば、送信先リファレンスとして **`refs/heads/main`** を指定した場合、テンプレートで定義された承認ルールは、送信先ブランチが `main` である場合にのみプルリクエストに適用されます。

Approval rule template
Approval rule template name
Require 1 approver from a senior developer for main branch
Description - optional
Before a pull request can be merged to the main branch, at least one senior developer must give an approval to the changes.
Number of approvals needed
1
Approval pool members - optional
If approval pool members are specified, only approvals from these members will count toward satisfying this rule. You can use wildcards to match multiple approvers with one value.
Approver type Info
Value
IAM user name or assumed role
SeniorDevelopers/*
Remove
Fully qualified ARN
s:iam::123456789012:user/Mary_Major
Remove
Add
Branch filters - optional
Use branch filters to only apply this template to a pull request if the destination branch name matches a name in the filter list.
Branch name
main
Remove
Add
Associated repositories
Repositories - optional
MyDemoRepo
MyTestRepo

トピック

- 承認ルールテンプレートを作成する
- 承認ルールテンプレートをリポジトリに関連付ける
- 承認ルールテンプレートの管理
- 承認ルールテンプレートの関連付けを解除する
- 承認ルールテンプレートを削除する

承認ルールテンプレートを作成する

1 つ以上の承認ルールテンプレートを作成して、リポジトリ間で開発ワークフローをカスタマイズできます。複数のテンプレートを作成することで、異なるブランチが適切なレベルの承認とコントロールを持つように、承認ルールの自動作成を設定できます。たとえば、本番稼働用ブランチと開発ブランチ用に異なるテンプレートを作成し、これらのテンプレートを 1 つ以上のリポジトリに適用できます。ユーザーがこれらのリポジトリでプルリクエストを作成すると、そのリクエストはそれらのテンプレートに対して評価されます。リクエストが適用されたテンプレートの条件と一致する場合、プルリクエストの承認ルールが作成されます。

コンソールまたは を使用して AWS CLI 、承認ルールテンプレートを作成できます。承認ルールテンプレートのマネージドポリシーと許可の詳細については、「承認ルールテンプレートに対するアクションのアクセス許可 および AWS CodeCommit の マネージドポリシー」を参照してください。

トピック

- 承認ルールテンプレートを作成する (コンソール)
- 承認ルールテンプレートを作成する (AWS CLI)

承認ルールテンプレートを作成する (コンソール)

承認ルールテンプレートは、デフォルトではどのリポジトリにも関連付けられません。テンプレートを作成するときに、テンプレートと 1 つ以上のリポジトリ間の関連付けを作成することも、後で関連付けを追加することもできます。

承認ルールテンプレートを作成するには (コンソール)

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Approval rule templates (承認ルールテンプレート)] を選択し、[Create template (テンプレートの作成)] を選択します。

3. [Approval rule template name (承認ルールのテンプレート名)] で、テンプレートにわかりやすい名前を付けます。例えば、プルリクエストをマージする前に、上級デベロッパーのうちの 1 人に対して、プルリクエストを承認するように求める場合は、ルール **Require 1 approver from a senior developer** に名前を付けることができます。
4. （オプション）[Description (説明)] に、このテンプレートの目的の説明を入力します。これは、このテンプレートがリポジトリに適しているかどうかを判断するのに役立ちます。
5. [Number of approvals needed (必要な承認の数)] に、必要な数値を入力します。デフォルトは 1 です。
6. （オプション）プルリクエストの承認を特定のユーザーグループからリクエストする場合は、[Approval rule members (承認ルールのメンバー)] で [Add (追加)] を選択します。[Approver type (承認者のタイプ)] で、次のいずれかを選択します。

   - [IAM user name or assumed role] (IAM ユーザー名または引き受けたロール): このオプションでは、サインインに使用したアカウントのアマゾン ウェブ サービスアカウント ID が事前に入力され、名前のみが必要です。指定された名前と名前が一致する IAM ユーザーおよびフェデレーティッドアクセスユーザーの両方に使用できます。これは非常に強力なオプションで、柔軟性が大きく高まります。例えば、このオプションを選択し、アマゾン ウェブ サービスアカウント 123456789012 でサインインして **Mary_Major** を指定した場合、次のすべてがそのユーザーからの承認としてカウントされます。

     - アカウントの IAM ユーザー (`arn:aws:iam::123456789012:user/Mary_Major`)
     - Mary_Major として IAM で識別されるフェデレーティッドユーザー (`arn:aws:sts::123456789012:federated-user/Mary_Major`)

     このオプションでは、ロールセッション名が Mary_Major (**CodeCommitReview**) の `arn:aws:sts::123456789012:assumed-role/CodeCommitReview/Mary_Major` ロールを引き受けるユーザーのアクティブなセッションは、ワイルドカード (`*Mary_Major`) を含めないかぎり認識されません。ロール名を明示的に指定することもできます (`CodeCommitReview/Mary_Major`)。

   - [Fully qualified ARN] (完全修飾 ARN): このオプションでは、IAM ユーザーまたはロールの完全修飾 Amazon リソースネーム (ARN) を指定できます。このオプションは、 AWS Lambda や などの他の AWS のサービスで使用される引き受けたロールもサポートしています AWS CodeBuild。委任ロールの ARN 形式は、ロールの場合は `arn:aws:sts::`*`AccountID`*`:assumed-role/`*`RoleName`*、関数の場合は `arn:aws:sts::`*`AccountID`*`:assumed-role/`*`FunctionName` です。

承認者のタイプとして [IAM user name or assumed role] (IAM ユーザー名または引き受けたロール) を選択した場合は、[Value] (値) に、IAM ユーザーまたはロールの名前またはユーザーかロールの完全修飾 ARN を入力します。承認が必要な承認の数にカウントされるすべてのユーザーまたはロールを追加するまで、[Add (追加)] を再度選択してユーザーまたはロールを追加します。

どちらの承認者タイプでも、値にワイルドカード (*) を使用できます。例えば、[IAM user name or assumed role] (IAM ユーザー名または引き受けたロール) オプションを選択し、**`CodeCommitReview/*`** を指定した場合、**`CodeCommitReview`** のロールを引き受けるすべてのユーザーが承認プールにカウントされます。個々のロールセッション名は、必要な承認者数にカウントされます。このようにして、Mary_Major と Li_Juan は、サインインして `CodeCommitReview` のロールを引き受けるときに承認としてカウントされます。IAM ARN、ワイルドカード、および形式の詳細については、IAM 識別子を参照してください。

> **Note**
>
> 承認ルールは、クロスアカウント承認をサポートしていません。

7. （オプション） [Branch filters (ブランチフィルター)] で、承認ルールの作成をフィルタリングするために使用する送信先ブランチ名を入力します。例えば、*`main`* を指定した場合、プルリクエストの送信先ブランチが *`main`* という名前のブランチである場合にのみ、関連付けられたリポジトリのプルリクエストに対して承認ルールが作成されます。ブランチ名にワイルドカード (*) を使用すると、ワイルドカードケースに一致するすべてのブランチ名に承認ルールを適用できます。ただし、ブランチ名の先頭にワイルドカードを使用することはできません。最大 100 のブランチ名を指定できます。フィルターを指定しない場合、テンプレートは関連付けられたリポジトリのすべてのブランチに適用されます。
8. (オプション) [Associated repositories (関連付けられたリポジトリ)] の [Repositories (リポジトリ)] リストで、この承認ルールに関連付けるこの AWS リージョン のリポジトリを選択します。

> **Note**
>
> テンプレートを作成した後で、リポジトリを関連付けることができます。詳細については、「承認ルールテンプレートをリポジトリに関連付ける」を参照してください。

9. [Create] を選択します。

Approval rule template

Approval rule template name

Require 1 approver from a senior developer for main branch

Description - *optional*

Before a pull request can be merged to the main branch, at least one senior developer must give an approval to the changes.

Number of approvals needed

1

Approval pool members - *optional*

If approval pool members are specified, only approvals from these members will count toward satisfying this rule. You can use wildcards to match multiple approvers with one value.

Approver type Info	Value	
IAM user name or assumed role	SeniorDevelopers/*	Remove
Fully qualified ARN	s:iam::123456789012:user/Mary_Major	Remove

Add

Branch filters - *optional*

Use branch filters to only apply this template to a pull request if the destination branch name matches a name in the filter list.

Branch name

main Remove

Add

Associated repositories

Repositories - *optional*

MyDemoRepo MyTestRepo

承認ルールテンプレートを作成する (AWS CLI)

を使用して AWS CLI 、承認ルールテンプレートを作成できます。を使用する場合 AWS CLI、テンプレートの送信先リファレンスを指定して、送信先ブランチがテンプレート内のブランチと一致するプルリクエストにのみ適用されます。

承認ルールテンプレートを作成するには (AWS CLI)

1. ターミナルまたはコマンドラインで、次を指定して、create-approval-rule-template コマンドを実行します。

   - 承認ルールテンプレートの名前。目的を説明する名前を使用することを検討してください。
   - 承認ルールテンプレートの説明。名前と同様に、詳細な説明を提供することを検討してください。
   - 承認ルールテンプレートの JSON 構造。この構造には、承認ルールが適用されるプルリクエストの送信先ブランチである送信先リファレンスの要件と、必要な承認の数に承認がカウントされるユーザーである承認プールメンバーの要件を含めることができます。

   承認ルールのコンテンツを作成するときに、次のいずれかの方法で、承認プールの承認者を指定できます。

   - CodeCommitApprovers: このオプションでは、アマゾン ウェブ サービスアカウントとリソースのみが必要です。指定されたリソース名と一致する名前を持つ IAM ユーザーとフェデレーティッドアクセスユーザーの両方に使用できます。これは非常に強力なオプションで、柔軟性が大きく高まります。例えば、 AWS アカウント 123456789012 と **Mary_Major** を指定した場合、以下のすべてがそのユーザーからの承認としてカウントされます。

     - アカウントの IAM ユーザー (`arn:aws:iam::123456789012:user/Mary_Major`)
     - Mary_Major として IAM で識別されるフェデレーティッドユーザー (`arn:aws:sts::123456789012:federated-user/Mary_Major`)

     このオプションでは、ロールセッション名が *Mary_Major* (`arn:aws:sts::123456789012:assumed-role/`*SeniorDevelopers*`/`*Mary_Major*) である *SeniorDevelopers* のロールを引き受けるユーザーのアクティブなセッションは、ワイルドカード (`*Mary_Major`) を含めない限り認識されません。

   - [Fully qualified ARN] (完全修飾 ARN): このオプションでは、IAM ユーザーまたはロールの完全修飾 Amazon リソースネーム (ARN) を指定できます。

IAM ARN、ワイルドカード、および形式の詳細については、IAM 識別子を参照してください。

次の例では、**2-approver-rule-for-main** という名前の承認ルールテンプレートと **Requires two developers from the team to approve the pull request if the destination branch is main** の説明を作成します。このテンプレートでは、**CodeCommitReview** のロールを引き受ける 2 人のユーザーがプルリクエストを **main** ブランチにマージする前に承認する必要があります。

```
aws codecommit create-approval-rule-template --approval-rule-template-name 2-
approver-rule-for-main --approval-rule-template-description "Requires two
 developers from the team to approve the pull request if the destination branch
 is main" --approval-rule-template-content "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}"
```

2. 成功すると、このコマンドは以下のような出力を返します。

```
{
    "approvalRuleTemplate": {
        "approvalRuleTemplateName": "2-approver-rule-for-main",
        "creationDate": 1571356106.936,
        "approvalRuleTemplateId": "dd8b17fe-EXAMPLE",
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
        "approvalRuleTemplateDescription": "Requires two developers from the team
 to approve the pull request if the destination branch is main",
        "lastModifiedDate": 1571356106.936,
        "ruleContentSha256": "4711b576EXAMPLE"
    }
}
```

承認ルールテンプレートをリポジトリに関連付ける

承認ルールテンプレートは特定の に作成されますが AWS リージョン、関連付けられる AWS リージョン までその のリポジトリには影響しません。1 つまたは複数のリポジトリにテンプレートを適

用するには、そのテンプレートをリポジトリに関連付ける必要があります。1 つのテンプレートを 1 つの AWS リージョンの複数のリポジトリに適用できます。これにより、プルリクエストの承認とマージの一貫した条件を作成することで、リポジトリの開発ワークフローを自動化および標準化できます。

承認ルールテンプレートは、承認ルールテンプレートが作成された AWS リージョン のリポジトリにのみ関連付けることができます。

承認ルールテンプレートのマネージドポリシーと許可の詳細については、「承認ルールテンプレートに対するアクションのアクセス許可 および AWS CodeCommit の マネージドポリシー」を参照してください。

トピック

- 承認ルールテンプレートを関連付ける (コンソール)
- 承認ルールテンプレートを関連付ける (AWS CLI)

承認ルールテンプレートを関連付ける (コンソール)

リポジトリの作成時に、承認ルールテンプレートとリポジトリが関連付けられている場合があります。（このステップは省略可能です）テンプレートを編集して、関連付けを追加または削除できます。

承認ルールテンプレートをリポジトリに関連付けるには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Approval rule templates (承認ルールテンプレート)] を選択します。テンプレートを選択し、[Edit (編集)] を選択します。
3. [Associated Repositories (関連付けられたリポジトリ)] で、[Repositories (リポジトリ)] リストからリポジトリを選択します。関連する各リポジトリがリストボックスの下に表示されます。
4. [保存] を選択します。承認ルールは、関連付けられたリポジトリで作成されたすべてのプルリクエストに適用されるようになりました。

承認ルールテンプレートを関連付ける (AWS CLI)

を使用して AWS CLI 、承認ルールテンプレートを 1 つ以上のリポジトリに関連付けることができます。

テンプレートを単一のリポジトリに関連付けるには

1. ターミナルまたはコマンドラインで、associate-approval-rule-template-with-repository コマンドを実行し、次を指定します。

   - リポジトリに関連付ける承認ルールテンプレートの名前。
   - 承認ルールテンプレートに関連付けるリポジトリの名前。

   例えば、*2-approver-rule-for-main* という名前の承認ルールテンプレートを、*MyDemoRepo* というリポジトリに関連付けるには、次のようにします。

   ```
   aws codecommit associate-approval-rule-template-with-repository --repository-name MyDemoRepo --approval-rule-template-name 2-approver-rule-for-main
   ```

2. 成功すると、このコマンドは何も返しません。

テンプレートを複数のリポジトリに関連付けるには

1. ターミナルまたはコマンドラインで、batch-associate-approval-rule-template-with-repositories コマンドを実行し、次を指定します。

   - リポジトリに関連付ける承認ルールテンプレートの名前。
   - 承認ルールテンプレートに関連付けるリポジトリの名前。

   例えば、**2-approver-rule-for-main** という名前の承認ルールテンプレートを **MyDemoRepo** および **MyOtherDemoRepo** という名前のリポジトリに関連付けるには、次のようにします。

   ```
   aws codecommit batch-associate-approval-rule-template-with-repositories --repository-names "MyDemoRepo", "MyOtherDemoRepo" --approval-rule-template-name 2-approver-rule-for-main
   ```

2. 成功すると、このコマンドは以下のような出力を返します。

   ```
   {
       "associatedRepositoryNames": [
           "MyDemoRepo",
           "MyOtherDemoRepo"
       ],
       "errors": []
   ```

```
}
```

承認ルールテンプレートの管理

の承認ルールテンプレートは、 で管理 AWS リージョン することで、その使用方法と目的を理解できます。たとえば、承認ルールテンプレートの名前と説明を編集して、他のユーザーがその目的を理解しやすくすることができます。内のすべての承認ルールテンプレートを一覧表示し AWS リージョン、テンプレートの内容と構造に関する情報を取得できます。どのテンプレートがリポジトリに関連付けられているか、およびどのリポジトリがテンプレートに関連付けられているかを確認できます。

承認ルールテンプレートのマネージドポリシーと許可の詳細については、「承認ルールテンプレートに対するアクションのアクセス許可 および AWS CodeCommit の マネージドポリシー」を参照してください。

承認ルールテンプレートを管理する (コンソール)

承認ルールテンプレートは、CodeCommit コンソールで表示および管理できます。

承認ルールテンプレートを管理するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. 承認ルールテンプレートを選択して、サインイン AWS リージョン している の承認ルールテンプレートのリストを表示します。

   > **Note**
   > 承認ルールテンプレートは、作成された AWS リージョン でのみ使用できます。

3. テンプレートを変更する場合は、リストからテンプレートを選択し、[Edit (編集)] を選択します。
4. 変更を行ってから、[Save] を選択します。

承認ルールテンプレートを管理する (AWS CLI)

承認ルールテンプレートは、次の AWS CLI コマンドで管理できます。

- list-approval-rule-templates、 内のすべての承認ルールテンプレートのリストを表示するには AWS リージョン
- get-approval-rule-template、承認ルールテンプレートのコンテンツを表示します。
- update-approval-rule-template-content、承認ルールテンプレートのコンテンツを変更します。
- update-approval-rule-template-name、承認ルールテンプレートの名前を変更します。
- update-approval-rule-template-description、承認ルールテンプレートの説明を変更します。
- list-repositories-for-approval-rule-template、承認ルールテンプレートに関連付けられているすべてのリポジトリを表示します。
- list-associated-approval-rule-templates-for-repository、リポジトリに関連付けられたすべての承認ルールテンプレートが表示されます。

AWS リージョンのすべての承認ルールテンプレートを一覧表示するには

1. ターミナルまたはコマンドラインで、list-approval-rule-templates コマンドを実行します。例えば、米国東部 (オハイオ) リージョンのすべての承認ルールテンプレートを一覧表示するには、次のようにします。

```
aws codecommit list-approval-rule-templates --region us-east-2
```

2. 成功すると、このコマンドは以下のような出力を返します。

```
{
    "approvalRuleTemplateNames": [
        "2-approver-rule-for-main",
        "1-approver-rule-for-all-pull-requests"
    ]
}
```

承認ルールテンプレートのコンテンツを取得するには

1. ターミナルまたはコマンドラインで、承認ルールテンプレートの名前を指定して、get-approval-rule-template コマンドを実行します。

```
aws codecommit get-approval-rule-template --approval-rule-template-name 1-approver-
rule-for-all-pull-requests
```

2. 成功すると、このコマンドは以下のような出力を返します。

```
{
    "approvalRuleTemplate": {
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "ruleContentSha256": "621181bbEXAMPLE",
        "lastModifiedDate": 1571356106.936,
        "creationDate": 1571356106.936,
        "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Li_Juan",
        "approvalRuleTemplateId": "a29abb15-EXAMPLE",
        "approvalRuleTemplateDescription": "All pull requests must be approved by
 one developer on the team."
    }
}
```

承認ルールテンプレートのコンテンツを更新するには

1. ターミナルまたはコマンドプロンプトで、テンプレートの名前と変更されたコンテンツを指定して、update-approval-rule-template-content コマンドを実行します。例えば、**1-approver-rule** という名前の承認ルールテンプレートの内容を変更して、**CodeCommitReview** のロールを引き受けるユーザーに承認プールを再定義するには、次のようにします。

```
aws codecommit update-approval-rule-template-content --approval-rule-template-
name 1-approver-rule --new-rule-content "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}"
```

2. 成功すると、このコマンドは以下のような出力を返します。

```
{
    "approvalRuleTemplate": {
        "creationDate": 1571352720.773,
        "approvalRuleTemplateDescription": "Requires 1 approval for all pull
 requests from the CodeCommitReview pool",
        "lastModifiedDate": 1571358728.41,
        "approvalRuleTemplateId": "41de97b7-EXAMPLE",
```

```
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",
        "ruleContentSha256": "2f6c21a5EXAMPLE",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Li_Juan"
    }
}
```

承認ルールテンプレートの名前を更新するには

1. ターミナルまたはコマンドプロンプトで、現在の名前と変更先の名前を指定して、update-approval-rule-template-name コマンドを実行します。たとえば、承認ルールテンプレートの名前を **1-approver-rule** から **1-approver-rule-for-all-pull-requests** に変更するには、次のようにします。

```
aws codecommit update-approval-rule-template-name --old-approval-rule-template-name
 "1-approver-rule" --new-approval-rule-template-name "1-approver-rule-for-all-pull-
requests"
```

2. 成功すると、このコマンドは以下のような出力を返します。

```
{
    "approvalRuleTemplate": {
        "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",
        "lastModifiedDate": 1571358241.619,
        "approvalRuleTemplateId": "41de97b7-EXAMPLE",
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "creationDate": 1571352720.773,
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
        "approvalRuleTemplateDescription": "All pull requests must be approved by
 one developer on the team.",
        "ruleContentSha256": "2f6c21a5cEXAMPLE"
    }
}
```

承認ルールテンプレートの説明を更新するには

1. ターミナルまたはコマンドラインで、承認ルールテンプレートの名前と新しい説明を指定して、update-approval-rule-template-description コマンドを実行します。

```
aws codecommit update-approval-rule-template-description --approval-rule-template-
name "1-approver-rule-for-all-pull-requests" --approval-rule-template-description
 "Requires 1 approval for all pull requests from the CodeCommitReview pool"
```

2. このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "approvalRuleTemplate": {
        "creationDate": 1571352720.773,
        "approvalRuleTemplateDescription": "Requires 1 approval for all pull
 requests from the CodeCommitReview pool",
        "lastModifiedDate": 1571358728.41,
        "approvalRuleTemplateId": "41de97b7-EXAMPLE",
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",
        "ruleContentSha256": "2f6c21a5EXAMPLE",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Li_Juan"
    }
}
```

テンプレートに関連付けられているすべてのリポジトリを一覧表示するには

1. コマンドラインまたはターミナルで、テンプレートの名前を指定して、list-repositories-for-approval-rule-template コマンドを実行します。

```
aws codecommit list-repositories-for-approval-rule-template --approval-rule-
template-name 2-approver-rule-for-main
```

2. 成功すると、このコマンドは以下のような出力を返します。

```
{
    "repositoryNames": [
        "MyDemoRepo",
        "MyClonedRepo"
```

```
    ]
}
```

リポジトリに関連付けられているすべてのテンプレートを一覧表示するには

1. コマンドラインまたはターミナルで、リポジトリの名前を指定して、list-associated-approval-rule-templates-for-repository コマンドを実行します。

```
aws codecommit list-associated-approval-rule-templates-for-repository --repository-name MyDemoRepo
```

2. 成功すると、このコマンドは以下のような出力を返します。

```
{
    "approvalRuleTemplateNames": [
        "2-approver-rule-for-main",
        "1-approver-rule-for-all-pull-requests"
    ]
}
```

承認ルールテンプレートの関連付けを解除する

承認ルールテンプレートによって生成された承認ルールが、リポジトリのチームのワークフローに意味をなさない場合には、そのリポジトリからテンプレートの関連付けを解除できます。テンプレートの関連付けを解除しても、テンプレートがリポジトリに関連付けられている間に作成された承認ルールは削除されません。

承認ルールテンプレートのマネージドポリシーと許可の詳細については、「承認ルールテンプレートに対するアクションのアクセス許可 および AWS CodeCommit の マネージドポリシー」を参照してください。

承認ルールテンプレートの関連付けを解除する (コンソール)

コンソールを使用して、リポジトリと承認ルールテンプレート間の関連付けを削除できます。

承認ルールテンプレートとリポジトリの関連付けを解除するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。

2. [Approval rule templates (承認ルールテンプレート)] を選択します。リポジトリから関連付けを解除するテンプレートを選択し、[Edit (編集)] を選択します。
3. [Associated repositories (関連付けられたリポジトリ)] で、関連付けを解除するリポジトリの横にある [X] を選択します。リポジトリ名が表示されなくなります。
4. [保存] を選択します。承認ルールは、それらのリポジトリで作成されたプルリクエストには適用されません。ルールは、関連付けが行われている間に行われたプルリクエストに適用されます。

承認ルールテンプレートの関連付けを解除する (AWS CLI)

を使用して AWS CLI 、承認ルールテンプレートから 1 つ以上のリポジトリの関連付けを解除できます。

承認ルールテンプレートをリポジトリから関連付け解除するには

1. ターミナルまたはコマンドラインで、disassociate-approval-rule-template-from-repository コマンドを実行し、次を指定します。

   - 承認ルールテンプレートの名前。
   - リポジトリの名前。

   たとえば、**1-approver-rule-for-all-pull-requests** という名前の承認ルールテンプレートを **MyDemoRepo** という名前のリポジトリから関連付け解除するには、次のようにします。

   ```
   aws codecommit disassociate-approval-rule-template-from-repository --repository-
   name MyDemoRepo --approval-rule-template-name 1-approver-rule-for-all-pull-requests
   ```

2. 成功すると、このコマンドは何も返しません。

複数のリポジトリから承認ルールテンプレートの関連付けを解除するには

1. ターミナルまたはコマンドラインで、batch-disassociate-approval-rule-template-from-repositories コマンドを実行し、次を指定します。

   - 承認ルールテンプレートの名前。
   - リポジトリの名前。

たとえば、**1-approver-rule-for-all-pull-requests** という名前の承認ルールテンプレートを **MyDemoRepo** と **MyOtherDemoRepo** という名前のリポジトリから関連付け解除するには、次のようにします。

```
aws codecommit batch-disassociate-approval-rule-template-from-repositories --
repository-names "MyDemoRepo", "MyOtherDemoRepo" --approval-rule-template-name 1-
approver-rule-for-all-pull-requests
```

2. 成功すると、このコマンドは以下のような出力を返します。

```
{
    "disassociatedRepositoryNames": [
        "MyDemoRepo",
        "MyOtherDemoRepo"
    ],
    "errors": []
}
```

承認ルールテンプレートを削除する

リポジトリで使用していない場合は、承認ルールテンプレートを削除できます。使用されていない承認ルールテンプレートを削除すると、テンプレートを整理しやすくなり、ワークフローに適したテンプレートを簡単に見つけることができます。

承認ルールテンプレートのマネージドポリシーと許可の詳細については、「承認ルールテンプレートに対するアクションのアクセス許可 および AWS CodeCommit の マネージドポリシー」を参照してください。

トピック

- 承認ルールテンプレートを削除する (コンソール)
- 承認ルールテンプレートを削除する (AWS CLI)

承認ルールテンプレートを削除する (コンソール)

開発作業に関連しなくなった承認ルールテンプレートを削除できます。コンソールを使用して承認ルールテンプレートを削除すると、削除プロセス中にリポジトリとの関連付けが解除されます。

承認ルールテンプレートを削除するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Approval rule templates (承認ルールテンプレート)] を選択します。削除するテンプレートを選択し、[Delete (削除)] を選択します。

承認ルールテンプレートを削除する (AWS CLI)

すべてのリポジトリから関連付けが解除されている場合は、 を使用して承認ルール AWS CLI を削除できます。詳細については、「承認ルールテンプレートの関連付けを解除する (AWS CLI)」を参照してください。

承認ルールテンプレートを削除するには

1. ターミナルまたはコマンドラインで、削除する承認ルールテンプレートの名前を指定して、delete-approval-rule-template コマンドを実行します。

```
aws codecommit delete-approval-rule-template --approval-rule-template-name 1-approver-for-all-pull-requests
```

2. 成功すると、このコマンドは以下のような出力を返します。承認ルールテンプレートが既に削除されている場合、このコマンドは何も返されません。

```
{
    "approvalRuleTemplateId": "41de97b7-EXAMPLE"
}
```

AWS CodeCommit リポジトリでのコミットの使用

コミットは、内容のスナップショットとリポジトリの内容への変更です。ユーザーが変更をコミットし、プッシュする度に、その情報は保存し、保管されます。そのため、情報には、変更や、コミットの日付、時刻、およびコミットの一部として行われる変更をコミットしたユーザーも含まれます。コミットにタグを追加して、特定のコミットを簡単に識別することもできます。CodeCommit では、次のことが可能です。

- コミットを確認する。
- コミットの履歴をグラフで表示する。
- コミットをその親または別の指定子と比較する。
- コミットにコメントを追加し、他の人のコメントに返信する。

コミットを CodeCommit リポジトリにプッシュする前に、ローカルコンピュータをリポジトリに接続する必要があります。最も簡単な方法については、「Git 認証情報を使用した HTTPS ユーザーのセットアップ」を参照してください。

CodeCommit でリポジトリの他の部分を操作する方法については、リポジトリを操作する、ファイルの操作、プルリクエストの操作、ブランチの操作、および ユーザー設定の操作 を参照してください。

トピック

- でコミットを作成する AWS CodeCommit
- でコミットの詳細を表示する AWS CodeCommit
- でコミットを比較する AWS CodeCommit
- でのコミットに関するコメント AWS CodeCommit
- で Git タグを作成する AWS CodeCommit
- で Git タグの詳細を表示する AWS CodeCommit
- で Git タグを削除する AWS CodeCommit

でコミットを作成する AWS CodeCommit

新しいリポジトリの最初のコミットを作成するときは、 AWS CLI および put-file コマンドを使用します。これにより、最初のコミットが作成され、新しいリポジトリのデフォルトのブランチを作成および指定することが可能になります。Git または を使用して、CodeCommit リポジトリにコミット AWS CLI を作成できます。ローカルリポジトリが CodeCommit リポジトリに接続されている場合は、Git を使用して、ローカルリポジトリから CodeCommit リポジトリにコミットをプッシュします。CodeCommit コンソールで直接コミットを作成するには、「AWS CodeCommit リポジトリへのファイルの作成または追加」および「AWS CodeCommit リポジトリ内のファイルの内容を編集する」を参照してください。

> **Note**
>
> ベストプラクティスとして、サポートされている最新バージョンの AWS CLI、Git、およびその他のソフトウェアを使用することをお勧めします。を使用する場合は AWS CLI、`create-commit` コマンドを含むバージョンを使用していることを確認するために、最新バージョンがインストールされていることを確認してください。

トピック

- を使用してリポジトリの最初のコミットを作成する AWS CLI
- Git クライアントを使用してコミットを作成する
- を使用してコミットを作成する AWS CLI

を使用してリポジトリの最初のコミットを作成する AWS CLI

AWS CLI および `put-file` コマンドを使用して、リポジトリの最初のコミットを作成できます。put-file を使用すると、空のリポジトリにファイルを追加する最初のコミットが作成され、指定した名前のブランチが作成されます。この新しいブランチは、リポジトリのデフォルトブランチとして指定されます。

> **Note**
>
> CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

を使用してリポジトリの最初のコミットを作成するには AWS CLI

1. ローカルコンピュータで、CodeCommit リポジトリへの最初のファイルとして追加するファイルを作成します。一般的なプラクティスとして、このリポジトリの用途を他のリポジトリユーザーに説明する `README.md` マークダウンファイルを作成します。`README.md` ファイルを含めると、このファイルの内容が CodeCommit コンソールでリポジトリの [Code] (コード) ページの下部に自動的に表示されます。
2. ターミナルまたはコマンドラインで、put-file コマンドを実行し、次を指定します。
   - 最初のファイルを追加するリポジトリの名前。
   - デフォルトのブランチとして作成するブランチの名前。
   - ファイルのローカルの場所。この場所に使用される構文は、ローカルのオペレーティングシステムによって異なります。
   - 追加するファイルの名前 (リポジトリ内の更新ファイルの保存先を示すパスを含む)。
   - このファイルに関連付けるユーザー名および E メールアドレス。
   - このファイルの追加理由を説明するコミットメッセージ。

   ユーザー名、E メールアドレス、コミットメッセージは、オプションですが、他のユーザーに変更者や変更の理由を示すために役立ちます。ユーザー名を指定しない場合、CodeCommit はデフォルトで IAM ユーザー名あるいはコンソールログインの派生を作成者名として使用します。

たとえば、ベース 6 でエンコードされたファイルコンテンツの例を含む *README.md* という名前のファイルを、*##*という名前のブランチの *MyDemoRepo* という名前のリポジトリに追加するには：

```
aws codecommit put-file --repository-name MyDemoRepo --branch-name development --
file-path README.md --file-content "EXAMPLE0123456789example0123456789EXAMPLE1"
 --name "Mary Major" --email "mary_major@example.com" --commit-message "I added a
 quick readme for our new team repository."
```

成功すると、このコマンドは以下のような出力を返します。

```
{
    "commitId": "724caa36EXAMPLE",
    "blobId": "a8a94062EXAMPLE",
    "treeId": "08b2fc73EXAMPLE"
}
```

Git クライアントを使用してコミットを作成する

ローカルコンピュータにインストールされた Git クライアントを使用してコミットを作成してから、それらのコミットを CodeCommit リポジトリにプッシュできます。

1. 前提条件 (例: 設定) を完了します。

   > **Important**
   > 設定が完了していない場合は、Git を使用してリポジトリに対して接続やコミットを行うことはできません。

2. 正しいブランチでコミットを作成していることを確認します。使用可能なブランチのリストを表示し、現在使用するように設定されているブランチを見つけるには、git branch を実行します。すべてのブランチが表示されます。現在のブランチの横にはアスタリスク (*) が表示されます。別のブランチに切り替えるには、git checkout ***branch-name*** を実行します。これが最初のコミットである場合は、git config コマンドを実行して、そのブランチに使用する名前を持つ最初のブランチを作成するように Git クライアントを設定します。例えば、デフォルトのブランチに *development* という名前を付ける場合は、次のようにします。

```
git config --local init.defaultBranch development
```

> **Tip**
>
> このコマンドは、Git v.2.28 以降でのみ使用できます。
> このコマンドを実行して、新しく作成されたすべてのリポジトリについて、デフォルトのブランチ名を **development** に設定することもできます。
>
> ```
> git config --global init.defaultBranch development
> ```

3. ブランチに変更を加える (ファイルの追加、変更、削除など)。

   例えば、ローカルリポジトリで、以下のテキストを含む `bird.txt` という名前のファイルを作成します。

   ```
   bird.txt
   --------
   Birds (class Aves or clade Avialae) are feathered, winged, two-legged, warm-
   blooded, egg-laying vertebrates.
   ```

4. git status を実行します。これにより、保留中のコミットに `bird.txt` がまだ含まれていないことが示されます。

   ```
   ...
   Untracked files:
     (use "git add <file>..." to include in what will be committed)

           bird.txt
   ```

5. git add bird.txt を実行して、保留中のコミットに新しいファイルを含めます。
6. git status をもう一度実行すると、以下のような出力が表示されます。`bird.txt` が保留中のコミットに含まれるか、コミットのためにステージングされたことがわかります。

   ```
   ...
   Changes to be committed:
     (use "git reset HEAD <file>..." to unstage)

           new file:   bird.txt
   ```

7. コミットを確定するには、git commit オプションを指定して -m を実行します (例: git commit -m "***Adding bird.txt to the repository.***")。-m オプションを指定すると、コミットメッセージが作成されます。

8. git status をもう一度実行すると、以下のような出力が表示されます。コミットがローカルリポジトリから CodeCommit リポジトリにプッシュされる準備ができていることがわかります。

```
...
nothing to commit, working directory clean
```

9. ローカルリポジトリから確定したコミットを CodeCommit リポジトリにプッシュする前に、git diff --stat ***remote-name***/***branch-name*** を実行することで、プッシュしている内容を表示できます。ここで、*remote-name* はローカルリポジトリが CodeCommit リポジトリに使用するニックネームであり、*branch-name* は比較するブランチの名前です。

> **Tip**
>
> ニックネームを取得するには、git remote を実行します。ブランチ名のリストを取得するには、git branch を実行します。現在のブランチの横にはアスタリスク (*) が表示されます。git status を実行して、ブランチ名を取得することもできます。

> **Note**
>
> リポジトリのクローンを作成した場合、ローカルリポジトリから見ると、*remote-name* は CodeCommit リポジトリの名前ではありません。リポジトリを複製すると、*remote-name* は自動的に origin に設定されます。

たとえば、git diff --stat origin/main では、以下のような出力が表示されます。

```
bird.txt | 1 +
1 file changed, 1 insertion(+)
```

この出力では、ローカルリポジトリを CodeCommit リポジトリに接続済みであることを前提としています。(手順については、「」を参照してください[リポジトリへの接続]

10. コミットをローカルリポジトリから CodeCommit リポジトリにプッシュする準備ができたら、git push ***remote-name branch-name*** を実行します。ここで、*remote-name* は、ロー

カルリポジトリが CodeCommit リポジトリに使用するニックネームであり、*branch -name* は、CodeCommit リポジトリにプッシュするブランチの名前です。

たとえば、 git push origin main を実行すると以下のような出力が表示されます。

HTTPS の場合:

```
Counting objects: 7, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 516 bytes | 0 bytes/s, done.
Total 5 (delta 2), reused 0 (delta 0)
remote:
To https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
    b9e7aa6..3dbf4dd main -> main
```

SSH の場合:

```
Counting objects: 7, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 516 bytes | 0 bytes/s, done.
Total 5 (delta 2), reused 0 (delta 0)
remote:
To ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
    b9e7aa6..3dbf4dd main -> main
```

> **Tip**
>
> -u オプションを git push に追加した場合 (例: git push -u origin main)、アップストリーム追跡情報が設定されているため、これ以降は git push を実行するだけで済みます。アップストリーム追跡情報を取得するには、git remote show ***remote-name*** (例: git remote show origin) を実行します。

他のオプションについては、Git のドキュメントを参照してください。

を使用してコミットを作成する AWS CLI

AWS CLI および `create-commit` コマンドを使用して、指定されたブランチの先端にリポジトリのコミットを作成できます。非参照のマージコミットを作成して、2 つのコミット識別子のマージ結果を表すこともできます。詳細については、「非参照コミットを作成する」を参照してください。

> **Note**
>
> CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

コミットを作成するには

1. ローカルコンピュータで、必要な変更を CodeCommit リポジトリに加えます。
2. ターミナルまたはコマンドラインで、create-commit コマンドを実行し、次を指定します。
   - 変更を加えるリポジトリ。
   - 変更を加えるブランチ。
   - ブランチに作成された最新のコミットの完全なコミット ID (ヒント、ヘッドコミット、または親コミット ID とも呼ばれます)。
   - 行った変更によってそれらのフォルダの内容が削除された場合に、空のフォルダを保持するかどうか。デフォルトでは、この値は false に設定されます。
   - 追加、変更、または削除するファイルに関する情報。
   - これらの変更に関連付けるユーザー名および E メール。
   - これらの変更を加えた理由についての説明をするコミットメッセージ。

   ユーザー名、E メールアドレス、コミットメッセージはオプションですが、他のユーザーに変更者と変更の理由について理解してもらうために便利です。ユーザー名を指定しない場合、CodeCommit はデフォルトで IAM ユーザー名あるいはコンソールログインの派生を作成者名として使用します。

   例えば、*`main`* ブランチの *`MyDemoRepo`* というリポジトリに `README.md` ファイルを追加するコミットをリポジトリに作成するには、次のようにします。ファイルの内容は Base64 にあり、「チームリポジトリへようこそ!」とあります。

```
aws codecommit create-commit --repository-name MyDemoRepo --
branch-name main --parent-commit-id 4c925148EXAMPLE --put-files
 "filePath=README.md,fileContent=V2VsY29tZSB0byBvdXIgdGVhbSByZXBvc2l0b3J5IQo="
```

> **Tip**
>
> 親コミット ID を取得するには、get-par コマンドを実行します。

成功すると、このコマンドは以下のような出力を返します。

```
{
    "commitId": "4df8b524-EXAMPLE",
    "treeId": "55b57003-EXAMPLE",
    "filesAdded": [
        {
            "blobId": "5e1c309dEXAMPLE",
            "absolutePath": "meeting.md",
            "fileMode": "NORMAL"
        }
    ],
    "filesDeleted": [],
    "filesUpdated": []
}
```

file1.py および *file2.py* という名前のファイルに変更を行うコミットを作成するには、ファイルの名前を *picture.png* から *image1.png* に変更し、それを *pictures* という名前のディレクトリから *images* という名前のディレクトリに移動します。次に、最新コミットの ID が *4c925148EXAMPLE* である *MyFeatureBranch* という名前のブランチの *MyDemoRepo* という名前のリポジトリにある *ExampleSolution.py* という名前のファイルを削除します。

```
aws codecommit create-commit --repository-name MyDemoRepo --branch-
name MyFeatureBranch --parent-commit-id 4c925148EXAMPLE --author-name "Saanvi
 Sarkar"
 --email "saanvi_sarkar@example.com" --commit-message "I'm creating this commit to
 update a variable name in a number of files."
 --keep-empty-folders false  --put-files '{"filePath": "file1.py", "fileMode":
 "EXECUTABLE", "fileContent": "bucket_name = sys.argv[1] region = sys.argv[2]"}'
```

```
'{"filePath": "file2.txt", "fileMode": "NORMAL", "fileContent": "//Adding a comment
 to explain the variable changes in file1.py"}' '{"filePath": "images/image1.png",
"fileMode": "NORMAL", "sourceFile": {"filePath": "pictures/picture.png", "isMove":
 true}}' --delete-files filePath="ExampleSolution.py"
```

> **Note**
>
> --put-files セグメントの構文は、オペレーティングシステムによって異なります。上記の例は、Linux、macOS、Unix ユーザー、および Bash エミュレータを持つ Windows ユーザー向けに最適化されています。コマンドラインあるいは Powersell の Windows ユーザーは、このシステムに適した構文を使用する必要があります。

成功すると、このコマンドは以下のような出力を返します。

```
{
   "commitId": "317f8570EXAMPLE",
   "treeId": "347a3408EXAMPLE",
   "filesAdded": [
        {
        "absolutePath": "images/image1.png",
        "blobId": "d68ba6ccEXAMPLE",
        "fileMode": "NORMAL"
        }
    ],
    "filesUpdated": [
        {
        "absolutePath": "file1.py",
        "blobId": "0a4d55a8EXAMPLE",
        "fileMode": "EXECUTABLE"
        },
        {
        "absolutePath": "file2.txt",
        "blobId": "915766bbEXAMPLE",
        "fileMode": "NORMAL"
        }
    ],
    "filesDeleted": [
        {
        "absolutePath": "ExampleSolution.py",
        "blobId": "4f9cebe6aEXAMPLE",
```

```
        "fileMode": "EXECUTABLE"
        },
        {
        "absolutePath": "pictures/picture.png",
        "blobId": "fb12a539EXAMPLE",
        "fileMode": "NORMAL"
        }
    ]
}
```

でコミットの詳細を表示する AWS CodeCommit

AWS CodeCommit コンソールを使用して、リポジトリ内のコミットの履歴を参照できます。これにより、リポジトリに加えられた変更を特定しやすくなります。変更内容には、以下を含みます。

- 変更日時および変更者。
- 特定のコミットがブランチにマージされた日時。

ブランチのコミット履歴を表示すると、ブランチ間の相違点を把握しやすくなります。タグ付けを使用する場合は、タグでラベル付けられたコミットや、そのタグが付いたコミットの親をすばやく表示できます。コマンドラインで Git を使用し、ローカルリポジトリや CodeCommit リポジトリのコミットに関する詳細を表示できます。

リポジトリのコミットの参照

AWS CodeCommit コンソールを使用して、リポジトリへのコミットの履歴を参照できます。また、リポジトリ内のコミットとそのブランチのグラフを時系列表示することもできます。これにより、変更日時など、リポジトリの履歴を理解しやすくなります。

> **Note**
>
> git rebase コマンドを使用してリポジトリをリベースすると、リポジトリの履歴が変更されます。これにより、コミットの順番が入れ替わることがあります。詳細については、「Gitでのブランチ作成からリベースまで」または Git のドキュメントを参照してください。

トピック

- リポジトリのコミット履歴を参照する

- リポジトリのコミット履歴のグラフを表示する

リポジトリのコミット履歴を参照する

コミッターやコミットメッセージに関する情報を含め、リポジトリの特定のブランチまたはタグのコミット履歴を閲覧できます。コミットのコードを表示することもできます。

コミットの履歴を参照するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Repositories (リポジトリ)] で、コミット履歴を確認するリポジトリを選択します。
3. ナビゲーションペインで、[Commits] を選択します。コミット履歴の表示では、リポジトリのコミット履歴がデフォルトのブランチに、コミット日の新しい順で表示されます。日付と時刻は協定世界時 (UTC) 表示されます。別のブランチのコミット履歴を表示するには、ビューセレクタボタンを選択後、リストからブランチを選択します。リポジトリでタグを使用している場合は、ビューセレクタボタンのタグを選択して、特定のタグが付けられたコミット、およびその親を表示できます。

4. コミットとその親の違いを表示し、変更に関するコメントを確認するには、省略されたコミット ID を選択します。詳細については、「コミットをその親と比較する」および「コミットについてコメントする」を参照してください。コミットと、それ以外のコミット指定子 (例: ブラン

チ、タグ、コミット ID) の違いを表示するには、「2 つのコミット指定子を比較」を参照してください。

5. 次の 1 つ以上の操作を行います。

   - 変更の日時を表示するには、コミットの日付の上にマウスカーソルを置きます。
   - フルコミット ID を表示するには、コピーして、テキストエディタまたは他の場所に貼り付けます。コピーするには、[Copy ID] を選択します。
   - コミット時のコードをそのまま表示するには、[Browse (参照)] を選択します。コミット時のリポジトリの内容が [Code] ビューに表示されます。ビューセレクタボタンを選択すると、ブランチまたはタグではなく、省略されたコミット ID が表示されます。

リポジトリのコミット履歴のグラフを表示する

リポジトリのコミット履歴のグラフを表示できます。[Commit Visualizer] ビューは、リポジトリのブランチに対するすべてのコミットの Directed Acyclic Graph (DAG) を表したものです。この図は、コミットや関連機能が追加またはマージされたタイミングを理解するのに役立ちます。また、他の変更との前後関係を指定しやすくなります。

> **Note**
>
> 早送りメソッドを使用してマージされたコミットは、コミットのグラフに別々の行として表示されません。

コミットのグラフを表示するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Repositories (リポジトリ)] で、コミットのグラフを表示するリポジトリを選択します。
3. ナビゲーションペインで [Commits (コミット)] を選択し、[Commit visualizer] を選択します。

コミットのグラフにおいて、各コミットメッセージの省略されたコミット ID と件名は、グラフの該当場所の横に表示されます。

> **Note**
>
> グラフは、1 ページに 35 ブランチまで表示できます。ブランチが 36 以上存在する場合は、グラフが複雑になるため表示できません。次の 2 つの方法でシンプルに表示することができます。
>
> - ビューセレクタボタンを使用して、特定のブランチのグラフを表示します。
> - フルコミット ID を検索ボックスに貼り付けて、グラフをコミットからレンダリングします。

4. コミットから新しいグラフをレンダリングするには、そのコミットに対応するグラフ内のポイントを選択します。ビューセレクタボタンは、省略されたコミット ID に変更されています。

コミットの詳細を表示する (AWS CLI)

Git を使用して、コミットに関する詳細を表示できます。を使用して AWS CLI 、次のコマンドを実行して、ローカルリポジトリまたは CodeCommit リポジトリのコミットに関する詳細を表示することもできます。

- コミットに関する情報を表示するには、aws codecommit get-commit を実行します。
- 複数のコミットに関する情報を表示するには、aws codecommit batch-get-commits を実行します。
- マージコミットに関する情報を表示するには、aws codecommit get-merge-commit を実行します。
- コミット指定子の変更に関する情報 (ブランチ、タグ、HEAD または コミット ID などのその他の完全に修飾されたリファレンス) を表示するには、aws codecommit get-differences を実行します。
- base64 でエンコードされたリポジトリの Git blob オブジェクトのコンテンツを表示するには、aws codecommit get-blob を実行します。

コミットに関する情報を表示するには

1. 次のように指定して aws codecommit get-commit コマンドを実行します。

- CodeCommit リポジトリの名前 (`--repository-name` オプション)。
- フルコミット ID。

例えば、`MyDemoRepo` という名前の CodeCommit リポジトリで ID が `317f8570EXAMPLE` のコミットに関する詳細を表示するには、次のようにします。

```
aws codecommit get-commit  --repository-name MyDemoRepo  --commit-id
 317f8570EXAMPLE
```

2. 正常に実行された場合、このコマンドの出力には以下が含まれます。

- (Git に設定されている) コミットの作者に関する情報 (タイムスタンプ形式の日付および協定世界時 (UTC) オフセットなど)
- (Git に設定されている) コミッターに関する情報 (タイムスタンプ形式の日付および協定世界時 (UTC) オフセットなど)
- コミットが存在する Git ツリーの ID。
- 親コミットのコミット ID。
- コミットメッセージ。

前述のコマンド例に基づいて、出力例をいくつか示します。

```
{
    "commit": {
        "additionalData": "",
        "committer": {
            "date": "1484167798 -0800",
            "name": "Mary Major",
            "email": "mary_major@example.com"
        },
        "author": {
            "date": "1484167798 -0800",
            "name": "Mary Major",
            "email": "mary_major@example.com"
        },
        "treeId": "347a3408EXAMPLE",
        "parents": [
            "4c925148EXAMPLE"
        ],
```

```
        "message": "Fix incorrect variable name"
    }
}
```

マージコミットに関する情報を表示するには

1. 次のように指定して get-merge-commit コマンドを実行します。

   - マージの送信元のコミット識別子 (--source-commit-specifier オプションを指定)。
   - マージの送信先のコミット識別子 (--destination-commit-specifier オプションを指定)。
   - 使用するマージオプション (--merge-option オプションを指定)。
   - レポジトリの名前 (--repository-name オプションを指定)。

   例えば、*MyDemoRepo* というリポジトリで *THREE_WAY_MERGE* 戦略を使用して、*main* という名前の送信先ブランチと *bugfix-bug1234* という名前の送信元ブランチのマージコミットに関する情報を表示するには、次のようにします。

```
aws codecommit get-merge-commit --source-commit-specifier bugfix-bug1234 --
destination-commit-specifier main --merge-option THREE_WAY_MERGE --repository-
name MyDemoRepo
```

2. 成功すると、このコマンドは次のような情報を返します。

```
{
    "sourceCommitId": "c5709475EXAMPLE",
    "destinationCommitId": "317f8570EXAMPLE",
    "baseCommitId": "fb12a539EXAMPLE",
    "mergeCommitId": "ffc4d608eEXAMPLE"
}
```

複数のコミットに関する情報を表示するには

1. 次のように指定して batch-get-commits コマンドを実行します。

   - CodeCommit リポジトリの名前 (`--repository-name` オプション)。
   - 情報を表示する各コミットの完全なコミット ID のリスト。

例えば、MyDemoRepo という名前の CodeCommit リポジトリで ID が 317f8570EXAMPLE および 4c925148EXAMPLE のコミットに関する詳細を表示するには、次のようにします。

```
aws codecommit batch-get-commits  --repository-name MyDemoRepo  --commit-ids
 317f8570EXAMPLE 4c925148EXAMPLE
```

2. 正常に実行された場合、このコマンドの出力には以下が含まれます。

   - (Git に設定されている) コミットの作者に関する情報 (タイムスタンプ形式の日付および協定世界時 (UTC) オフセットなど)
   - (Git に設定されている) コミッターに関する情報 (タイムスタンプ形式の日付および協定世界時 (UTC) オフセットなど)
   - コミットが存在する Git ツリーの ID。
   - 親コミットのコミット ID。
   - コミットメッセージ。

   前述のコマンド例に基づいて、出力例をいくつか示します。

```
{
    "commits": [
      {
        "additionalData": "",
        "committer": {
            "date": "1508280564 -0800",
            "name": "Mary Major",
            "email": "mary_major@example.com"
        },
        "author": {
            "date": "1508280564 -0800",
            "name": "Mary Major",
            "email": "mary_major@example.com"
        },
        "commitId": "317f8570EXAMPLE",
        "treeId": "1f330709EXAMPLE",
        "parents": [
            "6e147360EXAMPLE"
        ],
        "message": "Change variable name and add new response element"
```

```
    },
    {
        "additionalData": "",
        "committer": {
            "date": "1508280542 -0800",
            "name": "Li Juan",
            "email": "li_juan@example.com"
        },
        "author": {
            "date": "1508280542 -0800",
            "name": "Li Juan",
            "email": "li_juan@example.com"
        },
        "commitId": "4c925148EXAMPLE",
        "treeId": "1f330709EXAMPLE",
        "parents": [
            "317f8570EXAMPLE"
        ],
        "message": "Added new class"
    }
}
```

コミット指定子の変更に関する情報を表示するには

1. 次のように指定して aws codecommit get-differences コマンドを実行します。

   - CodeCommit リポジトリの名前 (`--repository-name` オプション)。
   - 取得する情報のコミット指定子。`--after-commit-specifier` のみ必須です。`--before-commit-specifier` を指定しない場合は、`--after-commit-specifier` 時点で最新のファイルがすべて表示されます。

   例えば、`MyDemoRepo` という名前の CodeCommit リポジトリで ID が `317f8570EXAMPLE` と `4c925148EXAMPLE` のコミット間の違いに関する情報を表示するには、次のようにします。

   ```
   aws codecommit get-differences  --repository-name MyDemoRepo  --before-commit-
   specifier 317f8570EXAMPLE --after-commit-specifier 4c925148EXAMPLE
   ```

2. 正常に実行された場合、このコマンドの出力には以下が含まれます。

   - 変更タイプなど、変更点を示したリスト (追加は A、削除は D、変更は M)。

- ファイル変更タイプのモード。
- 変更を含む Git blob オブジェクトの ID。

前述のコマンド例に基づいて、出力例をいくつか示します。

```
{
    "differences": [
        {
            "afterBlob": {
                "path": "blob.txt",
                "blobId": "2eb4af3bEXAMPLE",
                "mode": "100644"
            },
            "changeType": "M",
            "beforeBlob": {
                "path": "blob.txt",
                "blobId": "bf7fcf28fEXAMPLE",
                "mode": "100644"
            }
        }
    ]
}
```

Git blob オブジェクトに関する情報を表示するには

1. 次のように指定して aws codecommit get-blob コマンドを実行します。

   - CodeCommit リポジトリの名前 (`--repository-name` オプション)。
   - Git blob の ID (`--blob-id` オプションを指定)。

   例えば、`MyDemoRepo` という名前の CodeCommit リポジトリで ID が `2eb4af3bEXAMPLE` の Git blob に関する詳細を表示するには、次のようにします。

```
aws codecommit get-blob  --repository-name MyDemoRepo  --blob-id 2eb4af3bEXAMPLE
```

2. 正常に実行された場合、このコマンドの出力には以下が含まれます。

   - base64 でエンコードされた blob の内容 (通常はファイル)。

たとえば、前のコマンドの出力は、以下のようになります。

```
{
    "content": "QSBCaW5hcnkgTGFyToEXAMPLE="
}
```

コミットの詳細を表示する (Git)

以下のステップを実行する前に、ローカルリポジトリを CodeCommit リポジトリに接続して変更をコミット済みであるものとします。手順については、「リポジトリへの接続」を参照してください。

リポジトリに対する最新のコミットの変更を表示するには、git show コマンドを実行します。

```
git show
```

このコマンドでは、以下のような出力が生成されます。

```
commit 4f8c6f9d
Author: Mary Major <mary.major@example.com>
Date:   Mon May 23 15:56:48 2016 -0700

    Added bumblebee.txt

diff --git a/bumblebee.txt b/bumblebee.txt
new file mode 100644
index 0000000..443b974
--- /dev/null
+++ b/bumblebee.txt
@@ -0,0 +1 @@
+A bumblebee, also written bumble bee, is a member of the bee genus Bombus, in the
 family Apidae.
\ No newline at end of file
```

> **Note**
>
> この例と次の例において、コミット ID は省略されています。フルコミット ID は表示されません。

発生した変更を表示するには、git show コマンドとコミット ID を使用します。

```
git show 94ba1e60

commit 94ba1e60
Author: John Doe <johndoe@example.com>
Date:   Mon May 23 15:39:14 2016 -0700

    Added horse.txt

diff --git a/horse.txt b/horse.txt
new file mode 100644
index 0000000..080f68f
--- /dev/null
+++ b/horse.txt
@@ -0,0 +1 @@
+The horse (Equus ferus caballus) is one of two extant subspecies of Equus ferus.
```

2 つのコミット間の違いを表示するには、2 つのコミット ID を含めて git diff コマンドを実行します。

```
git diff ce22850d 4f8c6f9d
```

この例では、この 2 つのコミット間の違いは、2 つのファイルが追加されたことです。このコマンドでは、以下のような出力が生成されます。

```
diff --git a/bees.txt b/bees.txt
new file mode 100644
index 0000000..cf57550
--- /dev/null
+++ b/bees.txt
@@ -0,0 +1 @@
+Bees are flying insects closely related to wasps and ants, and are known for their
 role in pollination and for producing honey and beeswax.
diff --git a/bumblebee.txt b/bumblebee.txt
new file mode 100644
index 0000000..443b974
--- /dev/null
+++ b/bumblebee.txt
@@ -0,0 +1 @@
+A bumblebee, also written bumble bee, is a member of the bee genus Bombus, in the
 family Apidae.
```

```
\ No newline at end of file
```

Git を使用してローカルリポジトリのコミットに関する詳細を表示するには、git log コマンドを実行します。

```
git log
```

このコマンドが正常に実行されると、次のような出力が生成されます。

```
commit 94ba1e60
Author: John Doe <johndoe@example.com>
Date:   Mon May 23 15:39:14 2016 -0700

    Added horse.txt

commit 4c925148
Author: Jane Doe <janedoe@example.com>
Date:   Mon May 22 14:54:55 2014 -0700

    Added cat.txt and dog.txt
```

コミット ID およびメッセージのみを表示するには、git log --pretty=oneline コマンドを実行します。

```
git log --pretty=oneline
```

このコマンドが正常に実行されると、次のような出力が生成されます。

```
94ba1e60 Added horse.txt
4c925148 Added cat.txt and dog.txt
```

他のオプションについては、Git のドキュメントを参照してください。

でコミットを比較する AWS CodeCommit

CodeCommit コンソールを使用すると、CodeCommit リポジトリ内のコミット指定子の違いを表示できます。コミットとその親の違いをすばやく確認できます。コミット ID を含む 2 つの参照を比較することもできます。

トピック

- コミットをその親と比較する

- 2 つのコミット指定子を比較

コミットをその親と比較する

コミットとその親の違いをすばやく確認して、コミットメッセージ、コミッター、および変更された内容を確認できます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] ページで、コミットとその親の違いを表示するリポジトリを選択します。
3. ナビゲーションペインで、[Commits] を選択します。
4. リスト内の任意のコミットの省略コミット ID を選択します。ビューは、このコミットの詳細 (親コミットとの相違点など) を表示するように変更されます。

変更は、並べて ([Split] ビュー) またはインラインで ([Unified] ビュー) 表示できます。空白の変更を非表示または表示することもできます。コメントを追加することもできます。詳細については、「コミットについてコメントする」を参照してください。

> **Note**
>
> コードやその他のコンソール設定を表示するための設定は、変更するたびにブラウザの Cookie として保存されます。詳細については、「ユーザー設定の操作」を参照してください。

Developer Tools > CodeCommit > Repositories > MyDemoRepo > Commits >
7d09e44c
Commit 7d09e44c
Copy commit ID
Browse
Details
Author
Mary Major
mary_major@example.com
Commit date
48 minutes ago
Parent commit
e6aca768
Commit message
Adding a readme file to the repository.
Page 1 of 1
Go to file
Hide whitespace changes
Unified
Split
readme.md
Browse file contents
Comment on file
1 - This is a readme file that provides a basic description of what's in this repository.
\ No newline at end of file
1 + Use this repository for code changes to the *Demo* project. The default branch is *master*. Cod
\ No newline at end of file

> **Note**
>
> 行末のスタイル、コードエディタ、およびその他の要因によっては、行の特定の変更ではなく、行全体が追加または削除されることがあります。詳細のレベルは、git show または git diff コマンドで返されるものと一致します。

5. [Commit Visualizer] タブからコミットをその親と比較するには、省略コミット ID を選択します。コミットの詳細 (例: コミットとその親の間の変更) が表示されます。

2 つのコミット指定子を比較

CodeCommit コンソールでは、2 つのコミット指定子の違いを表示できます。コミット指定子は、ブランチ、タグ、コミット ID などのリファレンスです。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] ページで、コミット、ブランチ、またはタグ付きコミットを比較するリポジトリを選択します。
3. ナビゲーションペインで、[コミット]、[コミットの比較] の順に選択します。

4. ボックスを使用して、2 つのコミット指定子を比較します。

   - ブランチの先端を比較するには、リストからブランチ名を選択します。これは、比較のためにそのブランチからの最新のコミットを選択します。
   - コミットを関連する特定のタグと比較するには、リストから (ある場合) タグ名を選択します。これは、比較のためにタグ付けされたコミットを選択します。
   - 特定のコミットを比較するには、ボックスにコミット ID を入力するか、貼り付けます。完全なコミット ID を取得するには、ナビゲーションバーの [Commits] を選択し、リストからコミット ID をコピーします。[コミットの比較] ページで、テキストボックスに完全なコミット ID を貼り付け、[Use commit ID (コミット ID の使用)] を選択します。

5. 指定子を選択した後、[Compare] を選択します。

違いは、並べて ([Split] ビュー) またはインラインで ([Unified] ビュー) 表示できます。空白の変更を非表示または表示することもできます。

6. 比較の選択をクリアするには、[Cancel (キャンセル)] を選択します。

でのコミットに関するコメント AWS CodeCommit

CodeCommit コンソールを使用して、リポジトリのコミットについてコメントしたり、コミットについての他のユーザーのコメントを表示してそれに返答したりすることができます。これにより、以下の点を含め、リポジトリに加えられた変更について検討しやすくなります。

- なぜ変更が行われたか。
- 変更がさらに必要かどうか。
- 変更が別のブランチにマージされる必要があるか。

コメントは、コミット全体、コミット内のファイル、またはファイル内の特定の行や変更に対して行うことができます。コードの 1 行へのリンクを作成することもできます。そのためには、1 行を選択して、ブラウザに表示される URL をコピーします。

> **Note**
>
> 最良の結果を得るには、IAM ユーザーとしてサインインしているときにコメントを使用します。このコメント機能は、ルートアカウント認証情報、フェデレーティッドアクセスまたは一時的な認証情報を使用してサインインするユーザーには最適化されていません。

トピック

- リポジトリのコミットに対するコメントを表示する
- リポジトリのコミットに対するコメントの追加またはコメントへの応答
- コメントの表示、追加、更新、返信 (AWS CLI)

リポジトリのコミットに対するコメントを表示する

CodeCommit コンソールを使用して、コミットのコメントを表示します。

コミットについてのコメントを表示するには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] で、コミットのコメントを表示するレポジトリを選択します。
3. ナビゲーションペインで、[Commits] を選択します。コメントを表示するコミットのコミット ID を選択します。

   そのコミットのページがコメントとともに表示されます。

リポジトリのコミットに対するコメントの追加またはコメントへの応答

CodeCommit コンソールを使用して、コミットと親の比較や指定した 2 つのコミットの比較にコメントを追加することができます。また、絵文字、各自のコメント、その両方を使用してコメントに応答することもできます。

コミットにコメントを追加したり、コメントに応答したりする (コンソール)

テキストと絵文字を使用してコミットにコメントを追加したり、コメントに応答したりできます。コメントや絵文字は、コンソールへのサインインに使用した IAM ユーザーやロールに属するものとしてマークされます。

コミットについてのコメントの追加や返答をするには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] で、コミットにコメントするレポジトリを選択します。
3. ナビゲーションペインで、[Commits] を選択します。コメントを追加、またはコメントに返信するコミットのコミット ID を選択します。

   そのコミットのページがコメントとともに表示されます。
4. コメントを追加するには、以下のいずれかを実行します。

   - 全般的なコメントを追加するには、[Comments on changes (変更に関するコメント)] にコメントを入力して、[Save (保存)] を選択します。マークダウンを使用するか、プレーンテキスト形式でコメントを入力します。

   - コミット内のファイルにコメントを追加するには、ファイルの名前を見つけます。[Comment on file (ファイルに関するコメント)] を選択してコメントを入力し、[保存] を選択します。

   - コミット内の変更された行にコメントを追加するには、変更が表示される行に移動します。コメント用吹き出し

( )

を選択してコメントを入力し、[保存] を選択します。

> **Note**
>
> コメントは、保存後に編集できます。その内容を削除することもできます。コメントには、内容が削除されたことを示すメッセージが残ります。保存する前にコメントの [Preview markdown] モードの使用を検討してください。

5. コミットへのコメントに返信するには、[Reply] を選択します。絵文字を使用してコメントに返信するには、リストから目的の絵文字を選択します。1 つのコメントにつき 1 つの絵文字のみ選択できます。絵文字リアクションを変更する場合は、リストから別のリアクションを選択するか、[なし] を選択してリアクションを削除します。

2 つのコミット指定子を比較する際のコメントの追加とコメントへの返答

ブランチ間、タグ間、またはコミット間の比較にコメントを追加できます。

コミット指定子の比較時にコメントの追加または返信を行うには

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] ページで、リポジトリのリストから、コミット、ブランチ、またはタグ付きコミットを比較するリポジトリを選択します。
3. ナビゲーションペインで、[コミット]、[コミットの比較] タブの順に選択します。

4. 2 つのコミット指定子を比較するには、[送信先] フィールドおよび [送信元] フィールドを使用します。ドロップダウンリストを使用するか、コミット ID を貼り付けます。[Compare] を選択します。

5. 次の 1 つ以上の操作を行います。

- ファイルまたは行にコメントを追加するには、コメント用吹き出し

  を選択します。
- 比較された変更について全般的なコメントを追加するには、[Comments on changes] に移動します。

コメントの表示、追加、更新、返信 (AWS CLI)

コマンドの内容を表示、返信、更新、および削除するには、次のコマンドを実行します。

- : 2 つのコミット間の比較に対するコメントを表示するには、get-comments-for-compared-commit を実行します。
- コメントの詳細を表示するには、get-comment を実行します。
- 作成したコメントの内容を削除するには、delete-comment-content を実行します。
- 2 つのコミット間の比較に対するコメントを作成するには、post-comment-for-compared-commit を実行します。
- コメントを更新するには、update-comment を実行します。
- コメントに返信するには、post-comment-reply を実行します。
- 絵文字を使用してコメントに返信するには、put-comment-reaction を実行します。
- コメントへの絵文字リアクションを表示するには、get-comment-reactions を実行します。

コミットについてのコメントを表示するには

1. 次のように指定して get-comments-for-compared-commit コマンドを実行します。
   - CodeCommit リポジトリの名前 (`--repository-name` オプション)。
   - 比較の方向性を確立するためのコミット後のフルコミット ID (`--after-commit-id` option を使用)。
   - 比較の方向性を確立するためのコミット前のフルコミット ID (`--before-commit-id` オプションを使用)。
   - (オプション) 結果の次のバッチを返す列挙トークン (`--next-token` オプションを使用)。
   - (オプション) 返される結果の数を制限する負でない整数 (`--max-results` オプションを使用)。

例えば、*MyDemoRepo* という名前のリポジトリで 2 つのコミット間の比較に対して作成されたコメントを表示するには、次の操作を行います。

```
aws codecommit get-comments-for-compared-commit --repository-name MyDemoRepo --
before-commit-id 6e147360EXAMPLE --after-commit-id 317f8570EXAMPLE
```

2. このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
   "commentsForComparedCommitData": [
      {
         "afterBlobId": "1f330709EXAMPLE",
         "afterCommitId": "317f8570EXAMPLE",
         "beforeBlobId": "80906a4cEXAMPLE",
         "beforeCommitId": "6e147360EXAMPLE",
         "comments": [
            {
               "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
               "clientRequestToken": "123Example",
               "commentId": "ff30b348EXAMPLEb9aa670f",
               "content": "Whoops - I meant to add this comment to the line, not
 the file, but I don't see how to delete it.",
               "creationDate": 1508369768.142,
               "deleted": false,
               "CommentId": "123abc-EXAMPLE",
               "lastModifiedDate": 1508369842.278,
               "callerReactions": [],
               "reactionCounts":
                {
                  "SMILE" : 6,
                  "THUMBSUP" : 1
                }
            },
            {
               "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
               "clientRequestToken": "123Example",
               "commentId": "553b509bEXAMPLE56198325",
               "content": "Can you add a test case for this?",
               "creationDate": 1508369612.240,
               "deleted": false,
               "commentId": "456def-EXAMPLE",
```

```
                "lastModifiedDate": 1508369612.240,
                "callerReactions": [],
                "reactionCounts":
                 {
                   "THUMBSUP" : 2
                 }
             }
         ],
         "location": {
            "filePath": "cl_sample.js",
            "filePosition": 1232,
            "relativeFileVersion": "after"
         },
         "repositoryName": "MyDemoRepo"
      }
   ],
   "nextToken": "exampleToken"
}
```

コミットに対するコメントの詳細を表示するには

1. システムによって生成されたコメント ID を指定して、get-comment コマンドを実行します。例:

```
aws codecommit get-comment --comment-id ff30b348EXAMPLEb9aa670f
```

2. 成功すると、このコマンドは以下のような出力を返します。

```
{
   "comment": {
      "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
      "clientRequestToken": "123Example",
      "commentId": "ff30b348EXAMPLEb9aa670f",
      "content": "Whoops - I meant to add this comment to the line, but I don't see
 how to delete it.",
      "creationDate": 1508369768.142,
      "deleted": false,
      "commentId": "",
      "lastModifiedDate": 1508369842.278,
      "callerReactions": [],
      "reactionCounts":
         {
           "SMILE" : 6,
```

```
            "THUMBSUP" : 1
          }
    }
}
```

コミットに対するコメントの内容を削除するには

1. システムによって生成されたコメント ID を指定して、delete-comment-content コマンドを実行します。例:

```
aws codecommit delete-comment-content --comment-id ff30b348EXAMPLEb9aa670f
```

> **Note**
>
> コメントの内容を削除できるのは、AWSCodeCommitFullAccess ポリシーが適用されているか、`DeleteCommentContent` アクセス許可が [許可] に設定されている場合のみです。

2. このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
   "comment": {
      "creationDate": 1508369768.142,
      "deleted": true,
      "lastModifiedDate": 1508369842.278,
      "clientRequestToken": "123Example",
      "commentId": "ff30b348EXAMPLEb9aa670f",
      "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
      "callerReactions": [],
      "reactionCounts":
         {
           "CLAP" : 1
         }
   }
}
```

コミットに対するコメントを作成するには

1. 次のように指定して post-comment-for-compared-commit コマンドを実行します。

- CodeCommit リポジトリの名前 (`--repository-name` オプション)。
- 比較の方向性を確立するためのコミット後のフルコミット ID (`--after-commit-id` オプションを使用)。
- 比較の方向性を確立するためのコミット前のフルコミット ID (`--before-commit-id` オプションを使用)。
- 冪等性の一意、クライアントで生成されたトークン (--client-request-token オプションを指定)。
- コメントのコンテンツ (--content オプションを指定)。
- コメントの配置場所に関する場所情報のリストには、以下を含みます。
  - 拡張子やサブディレクトリなど、比較されているファイルの名前 (ある場合) (filePath 属性を指定)。
  - 比較ファイル内の変更の行番号 (filePosition 属性を指定)。
  - 送信元ブランチと送信先ブランチの比較で、変更のコメントが前か後か (relativeFileVersion 属性を指定)。

例えば、*##################?#* というコメントを、*MyDemoRepo* という名前のリポジトリ内の 2 つのコミット間の比較における *cl_sample.js* ファイルの変更に対して追加するには、次のようにします。

```
aws codecommit post-comment-for-compared-commit --repository-name MyDemoRepo
 --before-commit-id 317f8570EXAMPLE --after-commit-id 5d036259EXAMPLE --client-
request-token 123Example --content "Can you add a test case for this?" --location
 filePath=cl_sample.js,filePosition=1232,relativeFileVersion=AFTER
```

2. このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
        "afterBlobId": "1f330709EXAMPLE",
        "afterCommitId": "317f8570EXAMPLE",
        "beforeBlobId": "80906a4cEXAMPLE",
        "beforeCommitId": "6e147360EXAMPLE",
        "comment": {
              "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
              "clientRequestToken": "",
              "commentId": "553b509bEXAMPLE56198325",
              "content": "Can you add a test case for this?",
              "creationDate": 1508369612.203,
```

```
                "deleted": false,
                "commentId": "abc123-EXAMPLE",
                "lastModifiedDate": 1508369612.203,
                "callerReactions": [],
                "reactionCounts": []
            },
            "location": {
                "filePath": "cl_sample.js",
                "filePosition": 1232,
                "relativeFileVersion": "AFTER"
            },
        "repositoryName": "MyDemoRepo"
}
```

コミットに対するコメントを更新するには

1. システムによって生成された ID と、既存のコンテンツと置き換えるコンテンツを指定して、update-comment コマンドを実行します。

   例えば、*##############################* というコメントを、ID *442b498bEXAMPLE5756813* を持つコメントに追加するには、次のようにします。

   ```
   aws codecommit update-comment --comment-id 442b498bEXAMPLE5756813 --content "Fixed
    as requested. I'll update the pull request."
   ```

2. このコマンドが正常に実行されると、次のような出力が生成されます。

   ```
   {
       "comment": {
           "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
           "clientRequestToken": "",
           "commentId": "442b498bEXAMPLE5756813",
           "content": "Fixed as requested. I'll update the pull request.",
           "creationDate": 1508369929.783,
           "deleted": false,
           "lastModifiedDate": 1508369929.287,
           "callerReactions": [],
           "reactionCounts":
             {
               "THUMBSUP" : 2
             }
   ```

```
    }
}
```

コミットに対するコメントに返信するには

1. プルリクエストのコメントに応答するには、post-comment-reply コマンドを実行し、以下を指定します。

   - 返信したいコメントのシステム生成 ID (--in-reply-to オプションを指定)。
   - 冪等性の一意、クライアントで生成されたトークン (--client-request-token オプションを指定)。
   - 返答のコンテンツ (--content オプションを指定)。

   例えば、*##################* という返答を、システムによって生成された ID が *abcd1234EXAMPLEb5678efgh* のコメントに追加するには、次のようにします。

```
aws codecommit post-comment-reply --in-reply-to abcd1234EXAMPLEb5678efgh --
content "Good catch. I'll remove them." --client-request-token 123Example
```

2. このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "comment": {
        "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
        "clientRequestToken": "123Example",
        "commentId": "442b498bEXAMPLE5756813",
        "content": "Good catch. I'll remove them.",
        "creationDate": 1508369829.136,
        "deleted": false,
        "CommentId": "abcd1234EXAMPLEb5678efgh",
        "lastModifiedDate": 150836912.221,
        "callerReactions": [],
        "reactionCounts": []
    }
}
```

絵文字を使用してコミットのコメントに返信するには

1. 絵文字を使用してプルリクエストのコメントに返信したり、絵文字リアクションの値を変更したりするには、put-comment-reaction コマンドを実行し、以下を指定します。

   - 絵文字を使用して返信する対象のコメントの、システムによって生成された ID。
   - 追加または更新するリアクションの値。値として、サポートされている絵文字、ショートコード、Unicode 値を使用できます。

   CodeCommit では、絵文字として以下の値がサポートされています。

絵文字	ショートコード	Unicode
#	:thumbsup:	U+1F44D
#	:thumbsdown:	U+1F44E
#	:smile:	U+1F604
❤	:heart:	U+2764
#	:angry:	U+1F620
#	:confused:	U+1F615
#	:scream:	U+1F631
#	:sob:	U+1F62D
#	:clap:	U+1F44F
#	:confetti_ball:	U+1F38A
#	:ship:	U+1F6A2
#	:eyes:	U+1F440
	なし	U+0000

例えば、絵文字として*#######*を、システムによって生成された ID が *abcd1234EXAMPLEb5678efgh* のコメントに追加するには、次のようにします。

```
aws codecommit put-comment-reaction --comment-id abcd1234EXAMPLEb5678efgh --
reaction-value :thumbsup:
```

2. このコマンドが成功した場合、出力は生成されません。

コメントへの絵文字リアクションを表示するには

1. コメントへの絵文字リアクション (これらの絵文字に反応したユーザーを含む) を表示するには、システムによって生成されたコメントの ID を指定して、get-comment-reactions コマンドを実行します。

   例えば、システムによって生成された ID が *abcd1234EXAMPLEb5678efgh* のコメントへの絵文字リアクションを表示するには、次のようにします。

```
aws codecommit get-comment-reactions --comment-id abcd1234EXAMPLEb5678efgh
```

2. このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "reactionsForComment": {
        [
           {
                "reaction": {
                    "emoji:"#",
                    "shortCode": "thumbsup",
                    "unicode": "U+1F44D"
                },
                "users": [
                    "arn:aws:iam::123456789012:user/Li_Juan",
                    "arn:aws:iam::123456789012:user/Mary_Major",
                    "arn:aws:iam::123456789012:user/Jorge_Souza"
                ]
            },
            {
                "reaction": {
                    "emoji": "#",
                    "shortCode": "thumbsdown",
```

```
                    "unicode": "U+1F44E"
                },
                "users": [
                    "arn:aws:iam::123456789012:user/Nikhil_Jayashankar"
                ]
            },
            {
                "reaction": {
                    "emoji": "#",
                    "shortCode": "confused",
                    "unicode": "U+1F615"
                },
                "users": [
                    "arn:aws:iam::123456789012:user/Saanvi_Sarkar"
                ]
            }
        ]
    }
}
```

で Git タグを作成する AWS CodeCommit

Git タグを使用して、他のリポジトリユーザーがその重要性を理解するのに役立つラベル付きのコミットをマークすることができます。CodeCommit リポジトリに Git タグを作成するには、CodeCommit リポジトリに接続されたローカルリポジトリから Git を使用します。ローカルリポジトリに Git タグを作成したら、git push --tags を使用してCodeCommit リポジトリにプッシュします。

詳細については、「タグの詳細の表示」を参照してください。

Git を使用してタグを作成する

以下のステップに従い、ローカルリポジトリから Git を使用して CodeCommit リポジトリに Git タグを作成します。

これらの手順では、ローカルリポジトリが既に CodeCommit リポジトリに接続されていることを前提としています。手順については、「リポジトリへの接続」を参照してください。

1. git tag ***new-tag-name commit-id*** コマンドを実行します。ここで、*new-tag-name* は、新しい Git タグの名前、*commit-id* は、Git タグと関連付けるコミットの ID を表します。

たとえば、次のコマンドでは、beta という名前の Git タグを作成し、コミット ID (dc082f9a...af873b88) と関連付けます。

```
git tag beta dc082f9a...af873b88
```

2. ローカルリポジトリから新しい Git タグを CodeCommit リポジトリにプッシュするには、git push ***remote-name new-tag-name*** コマンドを実行します。ここで、*remote-name* は CodeCommit の名前、*new-tag-name* は新しい Git タグの名前を表します。

   例えば、新しい Git タグ (beta) を CodeCommit リポジトリ (origin) にプッシュするには、以下のように行います。

```
git push origin beta
```

> **Note**
>
> すべての新しい Git タグをローカルリポジトリから CodeCommit リポジトリにプッシュするには、git push --tags を実行します。
> ローカルリポジトリが CodeCommit リポジトリのすべての Git タグで更新されていることを確認するには、git fetch、git fetch --tagsの順に実行します。

他のオプションについては、Git のドキュメントを参照してください。

で Git タグの詳細を表示する AWS CodeCommit

Git では、タグはコミットのようなリファレンスに適用できるラベルで、他のリポジトリユーザーにとって重要な情報でマークするために使用します。たとえば、プロジェクトのベータリリースポイントであったコミットに beta というタグを付けることができます。**beta**詳細については、「Git を使用してタグを作成する」を参照してください。Git タグはリポジトリタグとは異なります。リポジトリタグの使用方法について詳細は、「リポジトリにタグを追加する」を参照してください。

AWS CodeCommit コンソールを使用して、各 Git タグによって参照されるコミットの日付やコミットメッセージなど、リポジトリ内の Git タグに関する情報を表示できます。コンソールから、タグに参照されているコミットをリポジトリのデフォルトブランチの先頭と比較することができます。他のコミットと同様に、その Git タグのポイントのコードを表示することもできます。

また、ターミナルやコマンドラインから Git を使用して、ローカルリポジトリの Git タグに関する詳細を表示することもできます。

トピック

- タグの詳細を表示する (コンソール)
- Git タグの詳細を表示する (Git)

タグの詳細を表示する (コンソール)

AWS CodeCommit コンソールを使用して、リポジトリの Git タグのリストと、Git タグによって参照されるコミットの詳細をすばやく表示します。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [Repositories (リポジトリ)] で、タグを表示するリポジトリの名前を選択します。
3. ナビゲーションペインで、[Git tags] を選択します。

4. 次のいずれかを行ってください。
   - そのコミットにあるコードをそのまま表示するには、Git タグ名を選択します。
   - コミットの詳細 (コミットメッセージ全体、コミッター、作成者など) を表示するには、省略されたコミット ID を選択します。

Git タグの詳細を表示する (Git)

Git を使用してローカルリポジトリの Git タグに関する詳細を表示するには、以下のコマンドのいずれかを実行します。

- git tag: Git タグ名の一覧を表示。
- git show: 特定の Git タグに関する情報を表示。
- git ls-remote: CodeCommit リポジトリの Git タグに関する情報を表示。

> **Note**
> ローカルリポジトリが CodeCommit リポジトリのすべての Git タグで更新されていることを確認するには、git fetch、git fetch --tagsの順に実行します。

以下の手順では、ローカルリポジトリが既に CodeCommit リポジトリに接続されていることを前提としています。手順については、「リポジトリへの接続」を参照してください。

ローカルリポジトリの Git タグの一覧を表示するには

1. git tag コマンドを実行します。

```
git tag
```

2. このコマンドが正常に実行されると、次のような出力が生成されます。

```
beta
release
```

> **Note**
> タグが定義されていない場合、git tag からは何も返されません。

他のオプションについては、Git のドキュメントを参照してください。

ローカルリポジトリの Git タグに関する情報を表示するには

1. git show ***tag-name*** コマンドを実行します。たとえば、beta という名前の Git タグに関する情報を表示するには、次のように実行します。

```
git show beta
```

2. このコマンドが正常に実行されると、次のような出力が生成されます。

```
commit 317f8570...ad9e3c09
Author: John Doe <johndoe@example.com>
Date:   Tue Sep 23 13:49:51 2014 -0700

    Added horse.txt

diff --git a/horse.txt b/horse.txt
new file mode 100644
index 0000000..df42ff1
--- /dev/null
+++ b/horse.txt
@@ -0,0 +1 @@
+The horse (Equus ferus caballus) is one of two extant subspecies of Equus ferus
\ No newline at end of file
```

> **Note**
>
> Git タグ情報の出力を終了するには、「:q」と入力します。

他のオプションについては、Git のドキュメントを参照してください。

CodeCommit リポジトリの Git タグに関する情報を表示するには

1. git ls-remote --tags コマンドを実行します。

```
git ls-remote --tags
```

2. このコマンドが正常に実行されると、CodeCommit リポジトリの Git タグ一覧が次のように出力されます。

```
129ce87a...70fbffba    refs/tags/beta
```

```
785de9bd...59b402d8    refs/tags/release
```

Git タグが定義されていない場合、git ls-remote --tags からは空白の行が返されます。

他のオプションについては、Git のドキュメントを参照してください。

で Git タグを削除する AWS CodeCommit

CodeCommit リポジトリの Git タグを削除するには、CodeCommit リポジトリに接続されたローカルリポジトリから Git を使用します。

Git を使用して Git タグを削除する

以下のステップに従い、ローカルリポジトリから Git を使用して CodeCommit リポジトリの Git タグを削除します。

これらのステップでは、ローカルリポジトリが既に CodeCommit リポジトリに接続されていることを前提としています。手順については、「リポジトリへの接続」を参照してください。

1. ローカルリポジトリから Git タグを削除するには、git tag -d ***tag-name*** コマンドを実行します。ここで、*tag-name* は削除する Git タグの名前です。

   > **Tip**
   > Git タグ名のリストを取得するには、git tag を実行します。

   例えば、beta という名前のローカルリポジトリのタグを削除するには、次のようにします。

   ```
   git tag -d beta
   ```

2. CodeCommit リポジトリから Git タグを削除するには、git push ***remote-name*** --delete ***tag-name*** コマンドを実行します。ここで、*remote-name* はローカルリポジトリが CodeCommit リポジトリに使用するニックネームであり、*tag-name* は、CodeCommit から削除する Git タグの名前です。

> **Tip**
>
> CodeCommit リポジトリの名前と URL のリストを取得するには、git remote -v コマンドを実行します。

例えば、`origin` という名前の CodeCommit リポジトリで `beta` という Git タグを削除するには、次のようにします。

```
git push origin --delete beta
```

AWS CodeCommit リポジトリでのブランチの使用

ブランチとは Git では、ブランチはコミットへのポインターまたは参照です。開発時、ブランチはタスクを整理するのに便利な方法です。ブランチを使用すると、他のブランチの作業に影響を与えることなく、新しいバージョンまたは異なるバージョンのファイルで個別に作業を行うことができます。また、新しい機能を開発したり、特定のコミットからプロジェクトの特定のバージョンを保存したりすることができます。最初のコミットを作成すると、デフォルトのブランチが作成されます。このデフォルトブランチは、ユーザーがリポジトリをクローンするときに、ローカルリポジトリのベースブランチまたはデフォルトブランチとして使用されるブランチです。そのデフォルトのブランチの名前は、最初のコミットの作成方法によって異なります。CodeCommit コンソール、、 AWS CLIまたはいずれかの SDKs を使用してリポジトリに最初のファイルを追加すると、そのデフォルトブランチの名前がメインになります。これは、このガイドの例で使用されているデフォルトのブランチ名です。Git クライアントを使用して最初のコミットをプッシュした場合、デフォルトのブランチの名前は、Git クライアントがデフォルトとして指定するものになります。最初のブランチの名前として main を使用するように Git クライアントを設定することをご検討ください。

CodeCommit では、リポジトリのデフォルトブランチを変更できます。ブランチを作成および削除することも、ブランチに関する詳細を表示することもできます。ブランチとデフォルトブランチ (または任意の 2 つのブランチ) の違いをすばやく比較できます。リポジトリ内のブランチとマージの履歴を表示するには、次の図に示すように、コミットビジュアライザーを使用できます。

CodeCommit でリポジトリの他の部分を操作する方法については、リポジトリを操作する、ファイルの操作 、プルリクエストの操作、コミットの操作、および ユーザー設定の操作 を参照してください。

トピック

- でブランチを作成する AWS CodeCommit
- のブランチへのプッシュとマージを制限する AWS CodeCommit
- でブランチの詳細を表示する AWS CodeCommit
- AWS CodeCommitでブランチを比較およびマージする
- でブランチ設定を変更する AWS CodeCommit
- でブランチを削除する AWS CodeCommit

でブランチを作成する AWS CodeCommit

CodeCommit コンソールまたは を使用して AWS CLI 、リポジトリのブランチを作成できます。これは、デフォルトのブランチの作業に影響を与えることなく、新しいバージョンまたは異なるバージョンのファイルで作業をすばやく個別に行う方法です。CodeCommit コンソールでブランチを作成したら、その変更をローカルリポジトリにプルする必要があります。または、CodeCommit に接続されたローカルリポジトリから Git を使用してブランチをローカルに作成し、その変更をプッシュすることもできます。

トピック

- ブランチを作成する (コンソール)
- ブランチを作成する (Git)
- ブランチを作成する (AWS CLI)

ブランチを作成する (コンソール)

CodeCommit コンソールを使用して CodeCommit リポジトリにブランチを作成できます。ユーザーが次に変更をリポジトリからプルするときに、新しいブランチが表示されます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリで、ブランチを作成するリポジトリの名前を選択します。
3. ナビゲーションペインで、[Branches] を選択します。
4. [Create branch] を選択します。

[Branch name (ブランチ名)] にブランチの名前を入力します。[Branch from (ブランチ元)] で、リストからブランチまたはタグを選択するか、コミット ID を貼り付けます。[Create branch] を選択します。

ブランチを作成する (Git)

以下の手順に従って、ローカルリポジトリから Git を使用してローカルリポジトリにブランチを作成して、そのブランチを CodeCommit リポジトリにプッシュします。

これらのステップでは、ローカルリポジトリが既に CodeCommit リポジトリに接続されていることを前提としています。手順については、「リポジトリへの接続」を参照してください。

1. git checkout -b ***new-branch-name*** コマンドを実行してローカルリポジトリにブランチを作成します。ここで、*new-branch-name* は新しいブランチの名前を表します。

   例えば、次のコマンドでは、MyNewBranch という名前のブランチがローカルリポジトリに作成されます。

   ```
   git checkout -b MyNewBranch
   ```

2. ローカルリポジトリから CodeCommit リポジトリに新しいブランチをプッシュするには、***remote-name*** と ***new-branch-name*** を両方とも指定して git push コマンドを実行します。

   例えば、origin というニックネームの CodeCommit リポジトリに、ローカルリポジトリの MyNewBranch という名前の新しいブランチをプッシュするには、次のようにします。

   ```
   git push origin MyNewBranch
   ```

> **Note**
>
> -u オプションを git push (例: git push -u origin main) に追加すると、今後 *remote-name branch-name* なしで git push を実行できます。アップストリーム追跡情報が設定されます。アップストリーム追跡情報を取得するには、git remote show ***remote-name*** (例: git remote show origin) を実行します。
> ローカルとリモートのすべての追跡ブランチのリストを見るには、git branch --all を実行してください。
> CodeCommit リポジトリ内のブランチに接続されているローカルリポジトリにブランチを設定するには、git checkout ***remote-branch-name*** を実行します。

他のオプションについては、Git のドキュメントを参照してください。

ブランチを作成する (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

を使用して CodeCommit リポジトリにブランチ AWS CLI を作成し、そのブランチを CodeCommit リポジトリにプッシュするには、次の手順に従います。最初のコミットを作成し、空のリポジトリのデフォルトブランチの名前を指定する手順については、「AWS CLIを使用してリポジトリの最初のコミットを作成する」を参照してください。

1. 次のように指定して create-branch コマンドを実行します。

   - ブランチが作成される CodeCommit の名前 (--repository-name オプションを指定)。

     > **Note**
     >
     > CodeCommit リポジトリの名前を取得するには、list-repositories コマンドを実行します。

   - 新しいブランチの名前 (--branch-name オプションを指定)。
   - 新しいブランチが指すコミットの ID (--commit-id オプションを指定)。

   例えば、コミット ID `317f8570EXAMPLE` を指す `MyNewBranch` という名前のブランチを、`MyDemoRepo` という名前の CodeCommit リポジトリに作成するには、次のようにします。

   ```
   aws codecommit create-branch --repository-name MyDemoRepo --branch-name MyNewBranch
    --commit-id 317f8570EXAMPLE
   ```

   このコマンドは、エラーがある場合にのみ出力を生成します。

2. ローカルリポジトリの利用可能な CodeCommit リポジトリブランチのリストを新しいリモートブランチ名で更新するには、git remote update ***`remote-name`*** を実行します。

   例えば、CodeCommit の利用可能なブランチのリストをニックネーム `origin` で更新するには、次のようにします。

   ```
   git remote update origin
   ```

   > **Note**
   >
   > または、git fetch コマンドを使用できます。また、git branch --all を実行するとすべてのリモートブランチを表示できますが、ローカルリポジトリのリストを更新するまで、作成したリモートブランチはリストに表示されません。

他のオプションについては、Git のドキュメントを参照してください。

3. CodeCommit リポジトリ内の新しいブランチに接続されているローカルリポジトリにブランチを設定するには、git checkout *remote-branch-name* を実行します。

> **Note**
>
> CodeCommit リポジトリの名前と URL のリストを取得するには、git remote -v コマンドを実行します。

のブランチへのプッシュとマージを制限する AWS CodeCommit

デフォルトでは、リポジトリにコードをプッシュするための十分な権限を持つ CodeCommit リポジトリユーザーは、そのリポジトリ内のブランチに貢献できます。これは、リポジトリにブランチをコンソール、コマンドライン、または Git を使用して追加した場合でも当てはまります。ただし、一部のリポジトリユーザーだけがそのブランチにコードをプッシュまたはマージできるように、ブランチを設定することもできます。たとえば、シニア開発者のサブセットのみが変更をそのブランチにプッシュまたはマージできるように、実動コードに使用されるブランチを設定することができます。他の開発者は、依然としてブランチから引き出し、独自のブランチを作成して、プルリクエストを作成できますが、そのブランチに変更をプッシュまたはマージすることはできません。IAM の 1 つ以上のブランチのコンテキストキーを使用する条件付きポリシーを作成して、このアクセスを設定できます。

> **Note**
>
> このトピックのいくつかの手順を完了するには、IAM ポリシーを設定して適用するのに十分なアクセス権限を持つ管理者ユーザーでサインインする必要があります。詳細については、「IAM 管理者のユーザーおよびグループの作成」を参照してください。

トピック

- ブランチへのプッシュとマージを制限するための IAM ポリシーの設定
- IAM グループまたはロールに IAM ポリシーを適用する
- ポリシーのテスト

ブランチへのプッシュとマージを制限するための IAM ポリシーの設定

IAM にポリシーを作成すると、ユーザーがブランチにコミットをプッシュしたり、プルリクエストをブランチにマージするなど、ブランチを更新できないようにすることができます。これを行うには、条件が満たされている場合にのみ Deny ステートメントの効果が適用されるように、ポリシーで条件ステートメントを使用します。Deny ステートメントに含める API によって、どのアクションが許可されないかが決まります。このポリシーは、リポジトリ内の 1 つのみのブランチ、リポジトリ内のいくつかのブランチ、または Amazon Web Services アカウント内のすべてのリポジトリで条件に一致するすべてのブランチに適用するように設定できます。

ブランチの条件付きポリシーを作成するには

1. にサインイン AWS Management Console し、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で、ポリシー を選択してください。
3. [ポリシーの作成] を選択します。
4. [JSON] を選択し、以下のポリシー例を貼り付けます。`Resource` の値を、アクセスを制限するブランチを含むリポジトリの ARN に置き換えます。`codecommit:References` の値を、アクセスを制限したいブランチへの参照に置き換えます。たとえば、このポリシーは、コミットのプッシュ、ブランチのマージ、ブランチの削除、ファイルの削除、プルリクエストのマージ、 という名前のブランチ*main*と という名前のリポジトリ*prod*内の という名前のブランチへのファイルの追加を拒否します*MyDemoRepo*。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "codecommit:GitPush",
                "codecommit:DeleteBranch",
                "codecommit:DeleteFile",
                "codecommit:PutFile",
                "codecommit:MergeBranchesByFastForward",
                "codecommit:MergeBranchesBySquash",
                "codecommit:MergeBranchesByThreeWay",
                "codecommit:MergePullRequestByFastForward",
                "codecommit:MergePullRequestBySquash",
                "codecommit:MergePullRequestByThreeWay"
```

```
            ],
            "Resource": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
            "Condition": {
                "StringEqualsIfExists": {
                    "codecommit:References": [
                        "refs/heads/main",
                        "refs/heads/prod"
                     ]
                },
                "Null": {
                    "codecommit:References": "false"
                }
            }
        }
    ]
}
```

Git のブランチは、ヘッドコミットの SHA-1 値へのポインタ (リファレンス) です。そのため、条件では `References` が使用されます。効果が `Null` で、`Deny` がアクションの 1 つであるポリシーでは、`GitPush` ステートメントが必要です。これは、ローカルリポジトリから CodeCommit に変更をプッシュするときに、Git と `git-receive-pack` がどのように動作するかによって必要となります。

> **Tip**
>
> Amazon Web Services アカウントですべてのリポジトリ内の main という名前のすべてのブランチに適用されるポリシーを作成するには、`Resource` の値をリポジトリ ARN からアスタリスク (`*`) に変更します。

5. [ポリシーの確認] を選択します。ポリシーステートメントのエラーを修正し、[ポリシーの作成] に進みます。
6. JSON が検証されると、[ポリシーの作成] ページが表示されます。[概要] セクションに、このポリシーによってアクセス許可が付与されないことを示す警告が表示されます。これは通常の動作です。
   - [Name] (名前) に、このポリシーの名前 (例: **`DenyChangesToMain`**) を入力します。
   - [説明] に、ポリシーの目的の説明を入力します。これはオプションですが推奨されます。
   - [ポリシーの作成] を選択します。

IAM グループまたはロールに IAM ポリシーを適用する

ブランチへのプッシュとマージを制限するポリシーを作成しましたが、そのポリシーは IAM ユーザー、グループ、またはロールに適用するまで効果がありません。ベストプラクティスとして、IAM グループまたはロールにポリシーを適用することを検討してください。個々の IAM ユーザーにポリシーを適用しても、うまくスケールされません。

条件付きポリシーをグループまたはロールに適用するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. ポリシーを IAM グループに適用する場合は、ナビゲーションペインで [グループ] を選択し、ユーザーが引き受けるロールにポリシーを適用する場合は、[ロール] を選択します。グループまたはロールの名前を選択します。
3. [Permissions] タブで [Attach Policy] を選択します。
4. 作成した条件付きポリシーをポリシーのリストから選択し、[ポリシーのアタッチ] を選択します。

詳細については、「IAM ポリシーをアタッチおよびデタッチする」を参照してください。

ポリシーのテスト

グループまたはロールに適用したポリシーの効果をテストして、期待どおりに機能するようにする必要があります。その方法は数多くあります たとえば、上記のようなポリシーをテストするには、以下のようにします。

- CodeCommit コンソールに、ポリシーが適用されている IAM グループのメンバーである IAM ユーザー、またはポリシーが適用されているロールを持つユーザーとしてサインインします。コンソールで、制限が適用されるブランチにファイルを追加します。そのブランチにファイルを保存またはアップロードしようとすると、エラーメッセージが表示されます。ファイルを別のブランチに追加します。これで、オペレーションが成功します。
- CodeCommit コンソールに、ポリシーが適用されている IAM グループのメンバーである IAM ユーザー、またはポリシーが適用されているロールを持つユーザーとしてサインインします。制限が適用されるブランチにマージするプルリクエストを作成します。プルリクエストを作成できるはずですが、マージしようとするとエラーが発生します。

- ターミナルまたはコマンドラインから、制限が適用されるブランチでコミットを作成し、そのコミットを CodeCommit リポジトリにプッシュします。エラーメッセージが表示されます。他のブランチからのコミットとプッシュはいつものように動作するはずです。

でブランチの詳細を表示する AWS CodeCommit

CodeCommit コンソールを使用して、CodeCommit リポジトリのブランチの詳細を表示できます。ブランチへの最後のコミットの日付、コミットメッセージなどを表示できます。CodeCommit リポジトリに接続されたローカルリポジトリから AWS CLI または Git を使用することもできます。CodeCommit

トピック

- ブランチの詳細を表示する (コンソール)
- ブランチの詳細を表示する (Git)
- ブランチの詳細を表示する (AWS CLI)

ブランチの詳細を表示する (コンソール)

CodeCommit コンソールを使用すると、リポジトリのブランチのリストとブランチの詳細をすばやく表示できます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] で、ブランチの詳細を表示するリポジトリの名前を選択します。
3. ナビゲーションペインで、[Branches] を選択します。

4. リポジトリのデフォルトとして使用されるブランチの名前が [Default branch (デフォルトブランチ)] ラベルの横に表示されます。ブランチに対する最新のコミットの詳細を表示するには、ブランチを選択し、[View last commit (最新のコミットを表示)] を選択します。ブランチ内のファイルとコードを表示するには、ブランチ名を選択します。

ブランチの詳細を表示する (Git)

ローカルリポジトリから Git を使って CodeCommit リポジトリのローカルおよびリモート追跡ブランチの両方に関する詳細を表示するには、git branch コマンドを実行します。

次のステップでは、ローカルリポジトリが既に CodeCommit リポジトリに接続されていることを前提としています。手順については、「リポジトリへの接続」を参照してください。

1. git branch オプションを指定して --all コマンドを実行します。

```
git branch --all
```

2. 成功すると、このコマンドは以下のような出力を返します。

```
  MyNewBranch
* main
  remotes/origin/MyNewBranch
```

```
remotes/origin/main
```

現在開いているブランチの横にはアスタリスク (*) が表示されます。それ以降のエントリはリモート追跡の参照です。

> **Tip**
>
> git branch でローカルブランチが表示されます。
> git branch -r でリモートブランチが表示されます。
> git checkout ***existing-branch-name*** は、指定したブランチ名に切り替わります。また、git branch を直後に実行すると、アスタリスク (*) がブランチ名の横に表示されます。
> git remote update ***remote-name*** は、使用可能な CodeCommit リポジトリブランチのリストでローカルリポジトリを更新します。CodeCommit リポジトリの名前と URL のリストを取得するには、git remote -v コマンドを実行します。

他のオプションについては、Git のドキュメントを参照してください。

ブランチの詳細を表示する (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

を使用して CodeCommit リポジトリ内のブランチの詳細 AWS CLI を表示するには、次のコマンドを 1 つ以上実行します。

- ブランチ名のリストを表示するには、list-branches を実行します。
- 特定のブランチに関する情報を表示するには、get-branch を実行します。

ブランチ名のリストを表示するには

1. CodeCommit リポジトリの名前、および `--repository-name` オプションを指定して、list-branches コマンドを実行します。

> **Tip**
>
> CodeCommit リポジトリの名前を取得するには、list-repositories コマンドを実行します。

例えば、MyDemoRepo という名前の CodeCommit 内のブランチに関する詳細を表示するには、次のようにします。

```
aws codecommit list-branches --repository-name MyDemoRepo
```

2. 成功すると、このコマンドは branchNameList オブジェクトを出力します。各エントリは各ブランチに対応しています。

   前述のコマンド例に基づいて、出力例をいくつか示します。

```
{
    "branches": [
        "MyNewBranch",
        "main"
    ]
}
```

ブランチに関する情報を表示するには

1. 次のように指定して get-branch コマンドを実行します。

   - リポジトリ名 (--repository-name オプションで指定)。
   - ブランチ名 (--branch-name オプションで指定)。

   例えば、MyDemoRepo という名前の CodeCommit リポジトリ内の、MyNewBranch という名前のブランチに関する情報を表示するには、次のようにします。

```
aws codecommit get-branch --repository-name MyDemoRepo --branch-name MyNewBranch
```

2. 成功すると、このコマンドはブランチの名前と、そのブランチに作成された最後のコミットの ID を出力します。

前述のコマンド例に基づいて、出力例をいくつか示します。

```
{
    "branch": {
          "branchName": "MyNewBranch",
          "commitID": "317f8570EXAMPLE"
    }
}
```

AWS CodeCommitでブランチを比較およびマージする

CodeCommit コンソールを使用して、CodeCommit リポジトリのブランチを比較できます。ブランチの比較により、すばやくブランチとデフォルトブランチ間の違いを表示したり、任意の 2 つのブランチ間の違いを表示できます。

トピック

- ブランチをデフォルトブランチと比較する
- 2 つの特定のブランチを比較する
- 2 つのブランチをマージする (AWS CLI)

ブランチをデフォルトブランチと比較する

CodeCommit コンソールを使用して、リポジトリのブランチとデフォルトブランチ間の違いをすばやく表示します。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリのリストで、ブランチを比較するリポジトリの名前を選択します。
3. ナビゲーションペインで、[コミット]、[コミットの比較] タブの順に選択します。
4. [送信先] で、デフォルトブランチの名前を選択します。[送信元] で、デフォルトブランチと比較するブランチを選択します。[Compare] を選択します。

2 つの特定のブランチを比較する

CodeCommit コンソールを使用して、比較する 2 つのブランチ間の違いを表示します。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリのリストで、ブランチを比較するリポジトリの名前を選択します。
3. ナビゲーションペインで、[コミット]、[コミットの比較] タブの順に選択します。
4. [送信先] および [送信元] で、比較する 2 つのブランチを選択し、[比較] を選択します。変更されたファイルのリストを表示するには、変更されたファイルのリストを展開します。ファイル内の変更は、並べて ([Split] ビュー) またはインラインで ([Unified] ビュー) 表示できます。

> **Note**
>
> IAM ユーザーとしてサインインしている場合、コードやその他のコンソール設定の表示用設定を設定して保存できます。詳細については、「ユーザー設定の操作」を参照してください。

2 つのブランチをマージする (AWS CLI)

CodeCommit リポジトリ内の 2 つのブランチをマージするには、次のいずれかのコマンドを実行して、使用可能なマージ戦略のいずれか AWS CLI を使用します。

- 早送りマージ戦略を使用して 2 つのブランチをマージするには、merge-branches-by-fast-forward コマンドを実行します。
- スカッシュマージ戦略を使用して 2 つのブランチをマージするには、merge-branches-by-squash コマンドを実行します。
- 3 方向マージ戦略を使用して 2 つのブランチをマージするには、merge-branches-by-three-way コマンドを実行します。

create-unreferenced-merge-commit コマンドを実行して、マージをテストすることもできます。詳細については、「プルリクエスト内の競合の解決」を参照してください。

> **Note**
>
> CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

を使用して CodeCommit リポジトリ内の 2 つのブランチを AWS CLI マージするには

1. 早送りマージ戦略を使用して 2 つのブランチをマージするには、以下を指定して merge-branches-by-fast-forward コマンドを実行します。

   - マージする変更を含む送信元ブランチの名前 (--source-commit-specifier オプションで指定)。
   - 変更をマージする送信先ブランチの名前 (--destination-commit-specifier オプションで指定)。
   - レポジトリの名前 (--repository-name オプションを指定)。

   例えば、*bugfix-1234* という名前の送信元ブランチを、*MyDemoRepo* という名前のリポジトリの *preprod* という名前の送信先ブランチにマージするには、次のようにします。

   ```
   aws codecommit merge-branches-by-fast-forward --source-commit-specifier bugfix-bug1234 --destination-commit-specifier preprod --repository-name MyDemoRepo
   ```

   このコマンドが正常に実行されると、次のような出力が生成されます。

   ```
   {
       "commitId": "4f178133EXAMPLE",
       "treeId": "389765daEXAMPLE"
   ```

```
}
```

2. スカッシュマージ戦略を使用して 2 つのブランチをマージするには、以下を指定して merge-branches-by-squash コマンドを実行します。

   - マージする変更を含む送信元ブランチの名前 (--source-commit-specifier オプションで指定)。
   - 変更をマージする送信先ブランチの名前 (--destination-commit-specifier オプションで指定)。
   - レポジトリの名前 (--repository-name オプションを指定)。
   - 含めるコミットメッセージ (--commit-message オプションを指定)。
   - コミットに使用する名前 (--name オプションを指定)。
   - コミットに使用する E メールアドレス (--email オプションを指定)。

   たとえば、*bugfix-bug1234* という名前の送信元ブランチを *MyDemoRepo* という名前のリポジトリの *bugfix-quarterly* という名前の送信先ブランチにマージするには、次のようにします。

```
aws codecommit merge-branches-by-squash --source-commit-specifier bugfix-bug1234 --
destination-commit-specifier bugfix-quarterly --author-name "Maria Garcia" --email
 "maria_garcia@example.com" --commit-message "Merging in fix branches to prepare
 for a general patch." --repository-name MyDemoRepo
```

   このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "commitId": "4f178133EXAMPLE",
    "treeId": "389765daEXAMPLE"
}
```

3. 3 方向マージ戦略を使用して 2 つのブランチをマージするには、以下を指定して merge-branches-by-three-way コマンドを実行します。

   - マージする変更を含む送信元ブランチの名前 (--source-commit-specifier オプションで指定)。
   - 変更をマージする送信先ブランチの名前 (--destination-commit-specifier オプションで指定)。
   - レポジトリの名前 (--repository-name オプションを指定)。
   - 含めるコミットメッセージ (--commit-message オプションを指定)。
   - コミットに使用する名前 (--name オプションを指定)。

- コミットに使用する E メールアドレス (--email オプションを指定)。

例えば、*main* という名前の送信元ブランチを、*MyDemoRepo* という名前のリポジトリの *bugfix-1234* という名前の送信先ブランチにマージするには、次のようにします。

```
aws codecommit merge-branches-by-three-way --source-commit-specifier main --
destination-commit-specifier bugfix-bug1234 --author-name "Jorge Souza" --email
 "jorge_souza@example.com" --commit-message "Merging changes from main to bugfix
 branch before additional testing."  --repository-name MyDemoRepo
```

このコマンドが正常に実行されると、次のような出力が生成されます。

```
{
    "commitId": "4f178133EXAMPLE",
    "treeId": "389765daEXAMPLE"
}
```

でブランチ設定を変更する AWS CodeCommit

デフォルトブランチとして使用するブランチは、 AWS CodeCommit コンソールまたは で変更できます AWS CLI。例えば、デフォルトのブランチを master に設定する Git クライアントを使用して最初のコミットを作成した場合、main という名前のブランチを作成し、新しいブランチがリポジトリのデフォルトのブランチとして設定されるようにブランチ設定を変更することができます。他のブランチ設定を変更するには、CodeCommit リポジトリに接続されたローカルリポジトリから Git を使用できます。

トピック

- デフォルトのブランチを変更する (コンソール)
- デフォルトのブランチを変更する (AWS CLI)

デフォルトのブランチを変更する (コンソール)

CodeCommit リポジトリのデフォルトブランチは、 AWS CodeCommit コンソールで指定できます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリのリストで、設定を変更するリポジトリの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) をクリックします。
4. [Default branch (デフォルトブランチ)] のドロップダウンリストを選択し、別のブランチを選択します。[保存] を選択します。

> **Tip**
>
> - ドロップダウンリストに別のブランチが表示されない場合は、追加のブランチを作成していません。リポジトリにブランチが 1 つしかない場合、リポジトリのデフォルトのブランチを変更することはできません。詳細については、「でブランチを作成する AWS CodeCommit」を参照してください。
> - [Default branch] (デフォルトブランチ) セクションが表示されず、通知ルールと接続に関する項目が表示される場合は、コンソールの全般設定メニューを表示しています。リポジトリの設定メニューは、[Code] (コード) および[Pull requests] (プルリクエスト) と同じレベルの[Repositories] (リポジトリ) の下に表示されます。

デフォルトのブランチを変更する (AWS CLI)

CodeCommit で AWS CLI コマンドを使用するには、 をインストールします AWS CLI。詳細については、「コマンドラインリファレンス」を参照してください。

を使用して CodeCommit AWS CLI リポジトリ内のリポジトリのブランチ設定を変更するには、次のコマンドを実行します。

- update-default-branch。デフォルトのブランチを変更します。

デフォルトのブランチを変更するには

1. 次のように指定して update-default-branch コマンドを実行します。
   - デフォルトのブランチが更新される CodeCommit リポジトリの名前 (--repository-name オプションを指定)。

> **Tip**
> CodeCommit リポジトリの名前を取得するには、list-repositories コマンドを実行します。

- 新しいデフォルトブランチの名前 (--default-branch-name オプションを指定)。

> **Tip**
> ブランチの名前を取得するには、list-branches コマンドを実行します。

2. 例えば、MyDemoRepo という名前の CodeCommit リポジトリのデフォルトブランチを MyNewBranch に変更するには、次のようにします。

```
aws codecommit update-default-branch --repository-name MyDemoRepo --default-branch-name MyNewBranch
```

このコマンドは、エラーがある場合にのみ出力を生成します。

他のオプションについては、Git のドキュメントを参照してください。

でブランチを削除する AWS CodeCommit

CodeCommit コンソールを使用して、リポジトリのブランチを削除できます。CodeCommit でブランチを削除しても、ローカルリポジトリのブランチは削除されません。したがって、ユーザーは、次回変更をプルするまで引き続きそのブランチのコピーを持つことになります。ローカルでブランチを削除して CodeCommit リポジトリにその変更をプッシュするには、CodeCommit リポジトリに接続されたローカルリポジトリから Git を使用します。

ブランチを削除してもコミットは削除されませんが、そのブランチ内のコミットへのリファレンスをすべて削除します。レポジトリ内の別のブランチにマージされていないコミットが含まれるブランチを削除した場合、それらのコミットの完全なコミット ID がなければ、それらを取得することはできません。

> **Note**
>
> このトピックの手順を使用して、リポジトリのデフォルトブランチを削除することはできません。デフォルトブランチを削除する場合は、ブランチを作成し、そのブランチをデフォルトブランチにしてから、古いブランチを削除する必要があります。詳細については、「ブランチを作成する」および「ブランチ設定を変更する」を参照してください。

トピック

- ブランチを削除する (コンソール)
- ブランチを削除する (AWS CLI)
- ブランチを削除する (Git)

ブランチを削除する (コンソール)

CodeCommit コンソールを使用して、CodeCommit リポジトリのブランチを削除できます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リポジトリで、ブランチを削除するリポジトリの名前を選択します。
3. ナビゲーションペインで、[Branches] を選択します。
4. 削除するブランチの名前を選択してから、[ブランチの削除] を選択し、選択内容を確認します。

ブランチを削除する (AWS CLI)

AWS CLI を使用して、そのブランチがリポジトリのデフォルトブランチでない場合は、CodeCommit リポジトリ内のブランチを削除できます。のインストールと使用の詳細については AWS CLI、「」を参照してくださいコマンドラインリファレンス。

1. ターミナルまたはコマンドラインで、delete-branch コマンドを実行し、次を指定します。
   - ブランチが削除される CodeCommit リポジトリの名前 (--repository-name オプションを指定)。

> **Tip**
>
> CodeCommit リポジトリの名前を取得するには、list-repositories コマンドを実行します。

- 削除するブランチの名前 (branch-name オプションを指定)。

> **Tip**
>
> ブランチの名前を取得するには、list-branches コマンドを実行します。

2. 例えば、MyDemoRepo という名前の CodeCommit リポジトリ内の、MyNewBranch という名前のブランチを削除するには、次のようにします。

```
aws codecommit delete-branch --repository-name MyDemoRepo --branch-name MyNewBranch
```

このコマンドでは、削除されたブランチに関する情報が返され、それには、削除されたブランチの名前、およびブランチのヘッドだったコミットの完全なコミット ID が含まれます。例:

```
"deletedBranch": {
    "branchName": "MyNewBranch",
    "commitId": "317f8570EXAMPLE"
}
```

ブランチを削除する (Git)

以下のステップに従い、ローカルリポジトリから Git を使用して CodeCommit リポジトリのブランチを削除します。

これらのステップでは、ローカルリポジトリが既に CodeCommit リポジトリに接続されていることを前提としています。手順については、「リポジトリへの接続」を参照してください。

1. ローカルリポジトリからブランチを削除するには、git branch -D ***branch-name*** コマンドを実行します。ここで、*branch-name* は削除するブランチの名前です。

> **Tip**
> ブランチ名のリストを取得するには、git branch --all を実行します。

例えば、`MyNewBranch` という名前のローカルリポジトリのブランチを削除するには、次のようにします。

```
git branch -D MyNewBranch
```

2. CodeCommit リポジトリからブランチを削除するには、git push ***remote-name*** --delete ***branch-name*** コマンドを実行します。ここで、*remote-name* はローカルリポジトリが CodeCommit リポジトリに使用するニックネームであり、*branch-name* は CodeCommit リポジトリから削除するブランチの名前です。

> **Tip**
> CodeCommit リポジトリの名前と URL のリストを取得するには、git remote -v コマンドを実行します。

例えば、`origin` という名前の CodeCommit リポジトリ内の、`MyNewBranch` という名前のブランチを削除するには、次のようにします。

```
git push origin --delete MyNewBranch
```

> **Tip**
> このコマンドでは、デフォルトブランチは削除されません。

他のオプションについては、Git のドキュメントを参照してください。

ユーザー設定の操作

AWS CodeCommit コンソールを使用して、いくつかのデフォルト設定を設定できます。たとえば、コードの変更を表示する設定 (インライン表示または分割表示) を変更できます。これらの設定のいずれかを変更すると、 AWS CodeCommit コンソールはブラウザに Cookie を設定し、コンソールを使用するたびに選択を保存して適用します。これらの設定は、そのブラウザを使用して AWS CodeCommit コンソールにアクセスするたびに、すべてのリージョンのすべてのリポジトリに適用されます。ユーザー設定は、リポジトリ固有またはリージョン固有のものではありません。これらは、、 AWS CodeCommit API AWS CLI、または とやり取りする他の サービスとのやり取りには影響しません AWS CodeCommit。

> **Note**
>
> ユーザー設定のクッキーはブラウザ固有です。ブラウザからクッキーを消去すると、ユーザー設定は消去されます。同様に、別のブラウザを使用してリポジトリにアクセスした場合、該当するブラウザのクッキーは使用できません。ユーザー設定は保持されません。

以下はユーザー設定の例です。

- コードの変更を表示する際に、[統合] ビューまたは [分割] ビューを使用するか、および空白の変更を表示するかどうか。
- コードの表示、編集、または作成時に、コードエディタウィンドウで明るい背景を使用するか暗い背景を使用するか。

設定を変更するための特定のページはありません。代わりに、コンソールで設定 (コードの変更を表示する方法など) を変更するたびに、その変更が保存されて該当箇所に適用されます。

への移行 AWS CodeCommit

Git リポジトリを CodeCommit リポジトリに移行するには、クローニング、ミラーリング、ブランチの全部または一部の移行などさまざまな方法があります。また、コンピュータ上のローカルでバージョン管理されていないコンテンツを CodeCommit に移行することもできます。

次のトピックでは、リポジトリを移行する方法をいくつか示します。使用するステップは、リポジトリの種類、スタイル、複雑さ、および移行する内容や方法によって異なる場合があります。非常に大きなリポジトリの場合、 を段階的に移行する場合があります。

> **Note**
>
> Perforce、Subversion、TFS などの他のバージョン管理システムから CodeCommit に移行することもできますが、まず Git に移行する必要があります。
> 他のオプションについては、Git のドキュメントを参照してください。
> あるいは、Scott Chacon と Ben Straub の Pro Git の本で、Git への移行についての情報を参照できます。

トピック

- Git リポジトリを に移行する AWS CodeCommit
- ローカルまたはバージョン管理されていないコンテンツを に移行する AWS CodeCommit
- リポジトリを段階的に移行する

Git リポジトリを に移行する AWS CodeCommit

既存の Git リポジトリを CodeCommit リポジトリに移行できます。このトピックの手順では、別の Git リポジトリにホストされているプロジェクトを CodeCommit に移行する方法について説明します。このプロセスの一環として、次の作業を行います。

- CodeCommit に必要な初期セットアップを完了します。
- CodeCommit リポジトリを作成します。
- リポジトリをクローンし、CodeCommit にプッシュします。
- CodeCommit リポジトリ内のファイルを表示します。
- CodeCommit リポジトリをチームと共有します。

トピック

- ステップ 0: CodeCommit へアクセスに必要なセットアップを行う
- ステップ 1: CodeCommit リポジトリを作成する
- ステップ 2: リポジトリのクローンを作成して CodeCommit リポジトリにプッシュする
- ステップ 3: CodeCommit でファイルを表示する
- ステップ 4: CodeCommit リポジトリを共有する

ステップ 0: CodeCommit へアクセスに必要なセットアップを行う

リポジトリを CodeCommit にマイグレーションする前に、CodeCommit 用の IAM ユーザーを作成して設定し、アクセス用にローカルコンピュータを設定する必要があります。また、 AWS CLI をインストールして CodeCommit を管理する必要があります。ほとんどの CodeCommit タスクは CLI なしで実行できますが、 AWS CLI はコマンドラインや端末で Git を操作するときに柔軟性を提供します。

CodeCommit 用に既に設定されている場合は、ステップ 1: CodeCommit リポジトリを作成する にスキップできます。

IAM ユーザーを作成および設定して CodeCommit にアクセスするには

1. アマゾン ウェブ サービスアカウントを作成するには、http://aws.amazon.com にアクセスし、[Sign Up] (サインアップ) を選択します。
2. IAM ユーザーを作成するか、アマゾン ウェブ サービスアカウントに関連付けられた既存のユーザーを使用します。アクセスキー ID およびシークレットアクセスキーがその IAM ユーザーに関連付けられていることを確認します。詳細については、アマゾン ウェブ サービスアカウントの IAM ユーザーの作成を参照してください。

   > **Note**
   >
   > CodeCommit には が必要です AWS Key Management Service。既存の IAM ユーザーを使用している場合は、CodeCommit に必要な AWS KMS アクションを明示的に拒否するポリシーがユーザーにアタッチされていないことを確認してください。詳細については、「AWS KMS および 暗号化」を参照してください。

3. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。
4. IAM コンソールのナビゲーションペインで、[Users] (ユーザー) を選択し、続いて、CodeCommit へアクセスするために設定する IAM ユーザーを選択します。
5. [Permissions (アクセス許可)] タブで、[Add Permissions (アクセス許可の追加)] を選択します。
6. [Grant permissions (アクセス許可の付与)] で、[Attach existing policies directly (既存のポリシーを直接アタッチする)] を選択します。
7. ポリシーの一覧から、[AWSCodeCommitPowerUser] または CodeCommit アクセスの別の管理ポリシーを選択します。詳細については、「AWS CodeCommit の マネージドポリシー」を参照してください。

   アタッチするポリシーを選択したら、[Next: Review] (次へ: 確認) を選択して、IAM ユーザーにアタッチするポリシーのリストを表示します。リストが正しい場合は、[Add permissions (アクセス許可の追加)] を選択します。

   CodeCommit 管理ポリシーや、その他のグループおよびユーザーを含むリポジトリへのアクセス共有の詳細については、リポジトリの共有 および AWS CodeCommitの認証とアクセスコントロール を参照してください。

をインストールして設定するには AWS CLI

1. ローカルマシンで、 をダウンロードしてインストールします AWS CLI。これは、コマンドラインから CodeCommit とやり取りするための前提条件です。 AWS CLI バージョン 2 のインストールが推奨されます。これは の最新バージョン AWS CLI であり、すべての最新機能をサポートしています。これは、 でルートアカウント、フェデレーティッドアクセス、または一時的な認証情報の使用 AWS CLI をサポートする の唯一のバージョンですgit-remote-codecommit。

   詳細については、AWS 「 コマンドラインインターフェイスのセットアップ」を参照してください。

   > **Note**
   >
   > CodeCommit は、 AWS CLI バージョン 1.7.38 以降でのみ動作します。ベストプラクティスとして、 をインストールまたは最新バージョンにアップグレード AWS CLI します。インストール AWS CLI した のバージョンを確認するには、 aws --version コマンドを実行します。
   > の古いバージョン AWS CLI を最新バージョンにアップグレードするには、「 のインストール AWS Command Line Interface」を参照してください。

2. このコマンドを実行して、 の CodeCommit コマンド AWS CLI がインストールされていることを確認します。

   ```
   aws codecommit help
   ```

   このコマンドは、CodeCommit コマンドのリストを返します。

3. 次のように、 configure コマンドを使用して プロファイル AWS CLI で を設定します。

   ```
   aws configure
   ```

   プロンプトが表示されたら、CodeCommit で使用する IAM ユーザーの AWS アクセスキーと AWS シークレットアクセスキーを指定します。また、 など、リポジトリが存在する AWS リージョン を必ず指定してくださいus-east-2。デフォルトの出力形式の入力を求められたら、json を指定します。例えば、IAM ユーザーのプロファイルを設定する場合は、次のようにします。

   ```
   AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
    Enter
   ```

```
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

で使用するプロファイルの作成と設定の詳細については AWS CLI、以下を参照してください。

- 名前付きプロファイル
- での IAM ロールの使用 AWS CLI
- Set コマンド
- 認証情報をローテーションして AWS CodeCommit リポジトリに接続する

別のリポジトリまたはリソースに接続するには AWS リージョン、デフォルトのリージョン名 AWS CLI で を再設定する必要があります。CodeCommit でサポートされるデフォルトのリージョン名は以下のとおりです。

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1

- ca-central-1

- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

CodeCommit と の詳細については AWS リージョン、「」を参照してくださいリージョンと Git 接続エンドポイント。IAM、アクセスキー、シークレットキーに関する詳細については、認証情報を取得する方法および IAM ユーザーのアクセスキーの管理を参照してください。 AWS CLI および プロファイルの詳細については、「名前付きプロファイル」を参照してください。

次に、Git をインストールする必要があります。

- Linux、macOS、Unix の場合:

  CodeCommit リポジトリのファイル、コミット、およびその他の情報を使用するには、ローカルマシンに Git をインストールする必要があります。CodeCommit は Git バージョン 1.7.9 以降をサポートしています。Git バージョン 2.28 は、初期コミットのブランチ名の設定をサポートしています。最新バージョンの Git を使用することをお勧めします。

  Git をインストールするには、Git のダウンロードなどのウェブサイトをお勧めします。

> **Note**
>
> Git は、定期的に更新されている、発展中のプラットフォームです。機能の変更により、CodeCommit での動作が影響を受ける場合があります。特定のバージョンの Git と CodeCommit で問題が発生した場合は、 の情報を確認してくださいトラブルシューティング

- Windows の場合:

  CodeCommit リポジトリのファイル、コミット、およびその他の情報を使用するには、ローカルマシンに Git をインストールする必要があります。CodeCommit は Git バージョン 1.7.9 以降をサポートしています。Git バージョン 2.28 は、初期コミットのブランチ名の設定をサポートしています。最新バージョンの Git を使用することをお勧めします。

  Git をインストールするには、Git for Windows などのウェブサイトをお勧めします。このリンクを使用して Git をインストールする場合、以下を除くすべてのインストールのデフォルト設定を使用できます。

  - [Adjusting your PATH environment (PATH 環境の調整)] ステップでプロンプトが表示されたら、[Use Git from the Windows Command Prompt (Windows コマンドプロンプトから Git を使用する)] オプションを選択します。
  - （オプション) CodeCommit の Git 認証情報を設定する AWS CLI 代わりに、 に含まれる認証情報ヘルパーで HTTPS を使用する場合は、「追加のオプションの設定」ページで、Git 認証情報マネージャーを有効にするオプションがオフになっていることを確認してください。Git 認証情報マネージャーは、IAM ユーザーが Git 認証情報を設定する場合のみ、CodeCommit と互換性があります。詳細については、「Git 認証情報を使用した HTTPS ユーザーのセットアップ」および「Git for Windows: Git for Windows をインストールしましたが、リポジトリへのアクセスが拒否されます (403)」を参照してください。

  > **Note**
  >
  > Git は、定期的に更新されている、発展中のプラットフォームです。機能の変更により、CodeCommit での動作が影響を受ける場合があります。特定のバージョンの Git と CodeCommit で問題が発生した場合は、 の情報を確認してくださいトラブルシューティング

CodeCommit は、HTTPS 認証と SSH 認証の両方をサポートしています。セットアップを完了するには、CodeCommit (HTTPS、ほとんどのユーザーに推奨）、CodeCommit (SSH) へのアクセスに使用する SSH キーペア、(フェデレーティッドアクセスを使用するユーザーに推奨）、または (HTTPS) に含まれる認証情報ヘルパーの Git 認証情報を設定する必要があります。 git-remote-codecommit AWS CLI

- サポートされているオペレーティングシステムすべての Git の認証情報については、「ステップ 3: CodeCommit への HTTPS 接続用の Git 認証情報を作成する」を参照してください。

- Linux、macOS、または UNIX の SSH については、SSH および Linux、macOS、または Unix: Git と CodeCommit 用にパブリックキーとプライベートキーをセットアップする を参照してください。
- Windows での SSH については、「ステップ 3: Git および CodeCommit 用のパブリックキーとプライベートキーをセットアップする」を参照してください。
- git-remote-codecommit の場合は、「git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」を参照してください。
- Linux、macOS、または Unix の認証情報ヘルパーについては、認証情報ヘルパーを設定する (Linux、macOS、または Unix) を参照してください。
- Windows での認証情報ヘルパーについては、「認証情報ヘルパーをセットアップする (Windows)」を参照してください。

ステップ 1: CodeCommit リポジトリを作成する

このセクションでは、CodeCommit コンソールを使用して、このチュートリアルの残りの部分で使用する CodeCommit リポジトリを作成します。を使用してリポジトリを作成するには、 AWS CLI「」を参照してくださいリポジトリを作成する (AWS CLI)。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタで、リポジトリ AWS リージョン を作成する を選択します。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。
3. [Repositories (リポジトリ)] ページで、[Create repository (リポジトリの作成)] を選択します。
4. [Create repository (リポジトリの作成)] ページの [Repository name (リポジトリ名)] に、リポジトリの名前を入力します。

   > **Note**
   >
   > リポジトリ名では大文字と小文字が区別されます。Amazon Web Services アカウントの名前は、 AWS リージョン 内で一意である必要があります。

5. (オプション) [Description (説明)] にリポジトリの説明を入力します。この説明は、お客様と他のユーザーがリポジトリの用途を識別するのに役立ちます。

> **Note**
>
> コンソールの説明フィールドに [Markdown] と表示され、すべての HTML 文字とすべての有効な Unicode 文字を使用できます。アプリケーションデベロッパーとして `GetRepository` または `BatchGetRepositories` API を使用していて、ウェブブラウザにレポジトリの説明フィールドを表示させる場合は、CodeCommit API リファレンスを参照してください。

6. (オプション) タグの追加 を選択して、1 つ以上のリポジトリタグ (AWS リソースの整理と管理に役立つカスタム属性ラベル) をリポジトリに追加します。詳細については、「でのリポジトリのタグ付け AWS CodeCommit」を参照してください。
7. (オプション) 追加設定を展開して、このリポジトリ内のデータの暗号化と復号にデフォルト AWS マネージドキー または独自のカスタマーマネージドキーを使用するかどうかを指定します。独自のカスタマーマネージドキーを使用する場合は、リポジトリを作成する AWS リージョン でそのキーが使用可能であること、およびキーがアクティブであることを確認する必要があります。詳細については、「AWS Key Management ServiceAWS CodeCommit リポジトリの および 暗号化」を参照してください。
8. (オプション) このリポジトリに Java または Python コードが含まれており、CodeGuru Reviewer で分析する場合は、[Enable Amazon CodeGuru Reviewer for Java and Python] (Java および Python 用に Amazon CodeGuru Reviewer を有効化) を選択します。CodeGuru Reviewer は、複数の機械学習モデルを使用して、コードの欠陥を検出し、プルリクエストの改善と修正を提案します。詳細については、Amazon CodeGuru Reviewer ユーザーガイドを参照してください。
9. [Create] を選択します。

作成されたリポジトリは、[リポジトリ] リストに表示されます。URL 列で、コピーアイコンを選択し、CodeCommit に接続するために使用するプロトコル (SSH または HTTPS) を選択します。URL をコピーします。

例えば、リポジトリの名前を *MyClonedRepository* とし、米国東部 (オハイオ) リージョンで HTTPS で Git 認証情報を使用している場合、URL は次のようになります。

```
https://git-codecommit.us-east-2.amazonaws.com/MyClonedRepository
```

この URL は後で ステップ 2: リポジトリのクローンを作成して CodeCommit リポジトリにプッシュする で必要になります。

ステップ 2: リポジトリのクローンを作成して CodeCommit リポジトリにプッシュする

このセクションでは、既存の Git リポジトリのクローン (ローカルリポジトリと呼ばれる) をローカルコンピュータに作成します。次に、ローカルリポジトリの内容を、先ほど作成した CodeCommit リポジトリにプッシュします。

1. ローカルコンピュータのターミナルまたはコマンドプロンプトから、git clone オプションを指定して `--mirror` コマンドを実行し、リモートリポジトリのベアコピーを *aws-codecommit-demo* という名前の新しいフォルダにクローンを作成します。これは移行にのみ使用するベアリポジトリです。CodeCommit の移行されたリポジトリとやり取りするためのローカルリポジトリではありません。ローカルリポジトリは、CodeCommit への移行が完了してから作成することができます。

   次の例では、GitHub (*https://github.com/awslabs/aws-demo-php-simple-app.git*) でホストされたデモアプリケーションを、*aws-codecommit-demo* というディレクトリのローカルリポジトリにクローンを作成します。

   ```
   git clone --mirror https://github.com/awslabs/aws-demo-php-simple-app.git aws-codecommit-demo
   ```

2. クローンを作成したディレクトリに変更します。

   ```
   cd aws-codecommit-demo
   ```

3. 送信先 CodeCommit リポジトリの URL と名前、および git push オプションを指定して、--all コマンドを実行します。(ここで指定するのは「ステップ 1: CodeCommit リポジトリを作成する」でコピーした URL です)。

   たとえば、リポジトリに *MyClonedRepository* という名前を付けて HTTPS を使用するように設定されている場合は、次のコマンドを実行します。

   ```
   git push https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyClonedRepository
    --all
   ```

> **Note**
>
> --all オプションは、リポジトリのすべてのブランチをプッシュします。タグのような他の参照をプッシュしません。タグをプッシュする場合は、最初のプッシュが完了するまで待機してから、--tags オプションを使用して、もう一度プッシュします。
>
> ```
> git push ssh://git-codecommit.us-east-2.amazonaws.com/v1/
> repos/MyClonedRepository --tags
> ```
>
> 詳細については、Git ウェブサイトで Git push のページを参照してください。大きなリポジトリをプッシュする方法については、特にすべての参照を一度にプッシュする場合 (例えば、--mirror オプションを指定する場合) は、リポジトリの段階的移行 を参照してください。

aws-codecommit-demo フォルダとそのコンテンツは、CodeCommit にリポジトリを移行した後で削除することができます。すべての正しいリファレンスを使用してローカルリポジトリを作成し、CodeCommit のリポジトリを操作するには、`git clone` オプションを指定せず `--mirror` コマンドを実行します。

```
git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyClonedRepository
```

ステップ 3: CodeCommit でファイルを表示する

ディレクトリの内容をプッシュした後、CodeCommit コンソールを使用して、そのリポジトリ内のすべてのファイルをすばやく表示できます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] で、リポジトリの名前 (例: *MyClonedRepository*) を選択します。
3. ブランチ、クローン URL、設定などのリポジトリ内のファイルを表示します。

ステップ 4: CodeCommit リポジトリを共有する

CodeCommit でリポジトリを作成すると、HTTPS 接続用と SSH 接続用の 2 つのエンドポイントが生成されます。どちらもネットワーク経由で安全な接続を提供します。ユーザーは、いずれかのプロトコルを使用できます。両方のエンドポイントは、どのプロトコルをユーザーに推奨するにしてもアクティブのままです。リポジトリを他のユーザーと共有する前に、リポジトリへのアクセスを他のユーザーに許可する IAM ポリシーを作成する必要があります。これらのアクセス指示をユーザーに提供します。

リポジトリのカスタマー管理ポリシーを作成します。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。
2. [ダッシュボード] ナビゲーションエリアで、[ポリシー] を選択し、次に [ポリシーの作成] を選択します。
3. [ポリシーの作成] ページで、[Import managed policy (マネージドポリシーのインポート)] を選択します。
4. [Import managed policies (マネージドポリシーのインポート)] ページの [フィルタポリシー] に「**`AWSCodeCommitPowerUser`**」と入力します。ポリシー名の横にあるボタンを選択し、[インポート] を選択します。
5. [Create policy] (ポリシーの作成) ページで [JSON] を選択します。次に示すように、CodeCommit アクションの `Resource` 行の「`*`」部分を、CodeCommit リポジトリの Amazon リソースネーム (ARN) に置き換えます。

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
```

```
]
```

> **Tip**
>
> CodeCommit リポジトリの ARN を確認するには、CodeCommit コンソールに移動し、リストからリポジトリ名を選択して [Settings] (設定) を選択します。詳細については、「リポジトリの詳細の表示」を参照してください。

このポリシーに複数のリポジトリを適用するには、リソースに ARN を指定して各リポジトリを追加します。次に示すように、各 Resource ステートメントはカンマで区切ります。

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
 "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo"
 ]
```

編集を完了したら、[ポリシーの確認] を選択します。

6. [Review Policy] (ポリシーの確認) ページの [Name] (名前) に、ポリシーの新しい名前 (例えば、*AWSCodeCommitPowerUser-MyDemoRepo*) を入力します。必要に応じて、このポリシーの説明を入力します。
7. [ポリシーの作成] を選択します。

リポジトリへのアクセスを管理するには、リポジトリユーザーの IAM グループを作成し、そのグループに IAM ユーザーを追加します。その後、前のステップで作成したカスタマー管理ポリシーをアタッチします。アクセスに必要なその他のポリシー (IAMUserSSHKeys または IAMSelfManageServiceSpecificCredentials など) をアタッチします。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。
2. [ダッシュボード] ナビゲーションエリアで、[グループ] を選択し、次に [Create New Group (新しいグループの作成)] を選択します。
3. [Set Group Name] (グループ名の設定) ページの [Group Name] (グループ名) に、グループの名前 (例: *MyDemoRepoGroup*) を入力し、[Next Step] (次のステップ) を選択します。ここで、グループ名の一部として、リポジトリ名を含めることを検討してください。

> **Note**
> この名前は、Amazon Web Services アカウント全体で一意である必要があります。

4. 前のセクションで作成したカスタマー管理ポリシー (例: AWSCodeCommitPowerUser-MyDemoRepo) の横にあるボックスをオンにします。
5. [Review] ページで、[Create Group] を選択します。IAM は、指定されたポリシーが既にアタッチされた状態でこのグループを作成します。このグループは、アマゾン ウェブ サービスアカウントに関連付けられたグループのリストに表示されます。
6. リストからグループを選択します。
7. グループの概要ページで、[ユーザー] タブを選択し、次に [Add Users to Group (グループにユーザーを追加)] を選択します。アマゾン ウェブ サービスアカウントに関連付けられているすべてのユーザーを示すリスト上で、CodeCommit リポジトリへのアクセスを許可するユーザーの横にあるボックスをオンにして、[Add Users] (ユーザーを追加) を選択します。

> **Tip**
> 検索ボックスに名前を入力して、ユーザーをすばやく見つけることができます。

8. ユーザーの追加が完了したら、IAM コンソールを閉じます。

設定したポリシーグループとポリシーを使用して、CodeCommit にアクセスするための IAM ユーザーを作成したら、リポジトリへの接続に必要な情報をそのユーザーに送信します。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタで、リポジトリ AWS リージョン が作成された を選択します。リポジトリは に固有です AWS リージョン。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。
3. [リポジトリ] ページで、共有するリポジトリを選択します。
4. [Clone URL] で、ユーザーが使用するプロトコルを選択します。接続プロトコルのクローン URL がコピーされます。
5. のインストール、プロファイルの設定、Git のインストールなど AWS CLI、他の手順とともにクローン URL をユーザーに送信します。接続プロトコルの設定情報を含めるようにしてください (HTTPS など)。

ローカルまたはバージョン管理されていないコンテンツを に移行する AWS CodeCommit

このトピックの手順では、コンピュータ上の既存のプロジェクトまたはローカルのコンテンツを CodeCommit リポジトリに移行する方法について説明します。このプロセスの一環として、次の作業を行います。

- CodeCommit に必要な初期セットアップを完了します。
- CodeCommit リポジトリを作成します。
- Git のバージョン管理対象になるローカルフォルダを配置し、そのフォルダの内容を CodeCommit リポジトリにプッシュします。
- CodeCommit リポジトリ内のファイルを表示します。
- CodeCommit リポジトリをチームと共有します。

トピック

- ステップ 0: CodeCommit へアクセスに必要なセットアップを行う
- ステップ 1: CodeCommit リポジトリを作成する
- ステップ 2: ローカルコンテンツを CodeCommit リポジトリに移行する

- ステップ 3: CodeCommit でファイルを表示する
- ステップ 4: CodeCommit リポジトリを共有する

ステップ 0: CodeCommit へアクセスに必要なセットアップを行う

ローカルコンテンツを CodeCommit に移行する前に、CodeCommit 用に IAM ユーザーを作成して設定し、アクセス用にローカルコンピュータを設定する必要があります。また、 AWS CLI をインストールして CodeCommit を管理する必要があります。ほとんどの CodeCommit タスクは CLI なしで実行できますが、 AWS CLI を使用することで Git での柔軟な作業が可能になります。

CodeCommit 用に既に設定されている場合は、ステップ 1: CodeCommit リポジトリを作成する にスキップできます。

IAM ユーザーを作成および設定して CodeCommit にアクセスするには

1. アマゾン ウェブ サービスアカウントを作成するには、http://aws.amazon.com にアクセスし、[Sign Up] (サインアップ) を選択します。
2. IAM ユーザーを作成するか、アマゾン ウェブ サービスアカウントに関連付けられた既存のユーザーを使用します。アクセスキー ID およびシークレットアクセスキーがその IAM ユーザーに関連付けられていることを確認します。詳細については、アマゾン ウェブ サービスアカウントの IAM ユーザーの作成を参照してください。

   > **Note**
   >
   > CodeCommit には が必要です AWS Key Management Service。既存の IAM ユーザーを使用している場合は、CodeCommit に必要な AWS KMS アクションを明示的に拒否するポリシーがユーザーにアタッチされていないことを確認してください。詳細については、「AWS KMS および 暗号化」を参照してください。

3. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。
4. IAM コンソールのナビゲーションペインで、[Users] (ユーザー) を選択し、続いて、CodeCommit へアクセスするために設定する IAM ユーザーを選択します。
5. [Permissions (アクセス許可)] タブで、[Add Permissions (アクセス許可の追加)] を選択します。
6. [Grant permissions (アクセス許可の付与)] で、[Attach existing policies directly (既存のポリシーを直接アタッチする)] を選択します。

7. ポリシーの一覧から、[AWSCodeCommitPowerUser] または CodeCommit アクセスの別の管理ポリシーを選択します。詳細については、「AWS CodeCommit の マネージドポリシー」を参照してください。

   アタッチするポリシーを選択したら、[Next: Review] (次へ: 確認) を選択して、IAM ユーザーにアタッチするポリシーのリストを表示します。リストが正しい場合は、[Add permissions (アクセス許可の追加)] を選択します。

   CodeCommit 管理ポリシーや、その他のグループおよびユーザーを含むリポジトリへのアクセス共有の詳細については、リポジトリの共有 および AWS CodeCommitの認証とアクセスコントロール を参照してください。

をインストールして設定するには AWS CLI

1. ローカルマシンで、 をダウンロードしてインストールします AWS CLI。これは、コマンドラインから CodeCommit とやり取りするための前提条件です。 AWS CLI バージョン 2 のインストールが推奨されます。これは の最新バージョン AWS CLI であり、すべての最新機能をサポートしています。これは、 でルートアカウント、フェデレーティッドアクセス、または一時的な認証情報の使用 AWS CLI をサポートする の唯一のバージョンですgit-remote-codecommit。

   詳細については、AWS 「 コマンドラインインターフェイスのセットアップ」を参照してください。

   > **Note**
   >
   > CodeCommit は、 AWS CLI バージョン 1.7.38 以降でのみ動作します。ベストプラクティスとして、 をインストールまたは最新バージョンにアップグレード AWS CLI します。インストール AWS CLI した のバージョンを確認するには、 aws --version コマンドを実行します。
   > の古いバージョン AWS CLI を最新バージョンにアップグレードするには、「 のインストール AWS Command Line Interface」を参照してください。

2. このコマンドを実行して、 の CodeCommit コマンド AWS CLI がインストールされていることを確認します。

```
aws codecommit help
```

   このコマンドは、CodeCommit コマンドのリストを返します。

3. 次のように、 configure コマンドを使用してプロファイル AWS CLI で を設定します。

```
aws configure
```

プロンプトが表示されたら、CodeCommit で使用する IAM ユーザーの AWS アクセスキーと AWS シークレットアクセスキーを指定します。また、 など、リポジトリが存在する AWS リージョン を必ず指定してください`us-east-2`。デフォルトの出力形式の入力を求められたら、`json` を指定します。例えば、IAM ユーザーのプロファイルを設定する場合は、次のようにします。

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

で使用するプロファイルの作成と設定の詳細については AWS CLI、以下を参照してください。

- 名前付きプロファイル
- での IAM ロールの使用 AWS CLI
- Set コマンド
- 認証情報をローテーションして AWS CodeCommit リポジトリに接続する

別のリポジトリまたはリソースに接続するには AWS リージョン、デフォルトのリージョン名 AWS CLI で を再設定する必要があります。CodeCommit でサポートされるデフォルトのリージョン名は以下のとおりです。

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2

- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

CodeCommit と の詳細については AWS リージョン、「」を参照してくださいリージョンと Git 接続エンドポイント。IAM、アクセスキー、シークレットキーに関する詳細については、認証情報を取得する方法および IAM ユーザーのアクセスキーの管理を参照してください。 AWS CLI および プロファイルの詳細については、「名前付きプロファイル」を参照してください。

次に、Git をインストールする必要があります。

- Linux、macOS、Unix の場合:

CodeCommit リポジトリのファイル、コミット、およびその他の情報を使用するには、ローカルマシンに Git をインストールする必要があります。CodeCommit は Git バージョン 1.7.9 以降をサポートしています。Git バージョン 2.28 は、初期コミットのブランチ名の設定をサポートしています。最新バージョンの Git を使用することをお勧めします。

Git をインストールするには、Git のダウンロードなどのウェブサイトをお勧めします。

> **Note**
>
> Git は、定期的に更新されている、発展中のプラットフォームです。機能の変更により、CodeCommit での動作が影響を受ける場合があります。特定のバージョンの Git と CodeCommit で問題が発生した場合は、 の情報を確認してくださいトラブルシューティング

- Windows の場合:

  CodeCommit リポジトリのファイル、コミット、およびその他の情報を使用するには、ローカルマシンに Git をインストールする必要があります。CodeCommit は Git バージョン 1.7.9 以降をサポートしています。Git バージョン 2.28 は、初期コミットのブランチ名の設定をサポートしています。最新バージョンの Git を使用することをお勧めします。

  Git をインストールするには、Git for Windows などのウェブサイトをお勧めします。このリンクを使用して Git をインストールする場合、以下を除くすべてのインストールのデフォルト設定を使用できます。

  - [Adjusting your PATH environment (PATH 環境の調整)] ステップでプロンプトが表示されたら、[Use Git from the Windows Command Prompt (Windows コマンドプロンプトから Git を使用する)] オプションを選択します。
  - (オプション) CodeCommit の Git 認証情報を設定する AWS CLI 代わりに、 に含まれる認証情報ヘルパーで HTTPS を使用する場合は、「追加のオプションの設定」ページで、Git 認証情報マネージャーを有効にするオプションがオフになっていることを確認してください。Git 認証情報マネージャーは、IAM ユーザーが Git 認証情報を設定する場合のみ、CodeCommit と互換性があります。詳細については、「Git 認証情報を使用した HTTPS ユーザーのセットアップ」および「Git for Windows: Git for Windows をインストールしましたが、リポジトリへのアクセスが拒否されます (403)」を参照してください。

> **Note**
>
> Git は、定期的に更新されている、発展中のプラットフォームです。機能の変更により、CodeCommit での動作が影響を受ける場合があります。特定のバージョンの Git と CodeCommit で問題が発生した場合は、 の情報を確認してください[トラブルシューティング]

CodeCommit は、HTTPS 認証と SSH 認証の両方をサポートしています。設定を完了するには、CodeCommit に使用する Git 認証情報 (HTTPS、ほとんどのユーザーに推奨)、CodeCommit へのアクセスに使用する SSH キーペア (SSH)、git-remote-codecommit (フェデレーティッドアクセスを使用するユーザーに推奨)、または AWS CLIに含まれる認証情報ヘルパーを設定する必要があります。

- サポートされているオペレーティングシステムすべての Git の認証情報については、「ステップ 3: CodeCommit への HTTPS 接続用の Git 認証情報を作成する」を参照してください。
- Linux、macOS、または UNIX の SSH については、SSH および Linux、macOS、または Unix: Git と CodeCommit 用にパブリックキーとプライベートキーをセットアップする を参照してください。
- Windows での SSH については、「ステップ 3: Git および CodeCommit 用のパブリックキーとプライベートキーをセットアップする」を参照してください。
- git-remote-codecommit の場合は、「git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」を参照してください。
- Linux、macOS、または Unix の認証情報ヘルパーについては、認証情報ヘルパーを設定する (Linux、macOS、または Unix) を参照してください。
- Windows での認証情報ヘルパーについては、「認証情報ヘルパーをセットアップする (Windows)」を参照してください。

ステップ 1: CodeCommit リポジトリを作成する

このセクションでは、CodeCommit コンソールを使用して、このチュートリアルの残りの部分で使用する CodeCommit リポジトリを作成します。を使用してリポジトリを作成するには、 AWS CLI「」を参照してください[リポジトリを作成する (AWS CLI)]。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタで、リポジトリ AWS リージョン を作成する を選択します。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。
3. [Repositories (リポジトリ)] ページで、[Create repository (リポジトリの作成)] を選択します。
4. [Create repository (リポジトリの作成)] ページの [Repository name (リポジトリ名)] に、リポジトリの名前を入力します。

> **Note**
> リポジトリ名では大文字と小文字が区別されます。Amazon Web Services アカウントの名前は、 AWS リージョン 内で一意である必要があります。

5. (オプション) [Description (説明)] にリポジトリの説明を入力します。この説明は、お客様と他のユーザーがリポジトリの用途を識別するのに役立ちます。

> **Note**
> コンソールの説明フィールドに [Markdown] と表示され、すべての HTML 文字とすべての有効な Unicode 文字を使用できます。アプリケーションデベロッパーとして `GetRepository` または `BatchGetRepositories` API を使用していて、ウェブブラウザにレポジトリの説明フィールドを表示させる場合は、CodeCommit API リファレンスを参照してください。

6. (オプション) タグの追加 を選択して、1 つ以上のリポジトリタグ (AWS リソースの整理と管理に役立つカスタム属性ラベル) をリポジトリに追加します。詳細については、「でのリポジトリのタグ付け AWS CodeCommit」を参照してください。
7. (オプション) 追加設定を展開して、このリポジトリ内のデータの暗号化と復号にデフォルト AWS マネージドキー または独自のカスタマーマネージドキーを使用するかどうかを指定します。独自のカスタマーマネージドキーを使用する場合は、リポジトリを作成する AWS リージョン でそのキーが使用可能であること、およびキーがアクティブであることを確認する必要があります。詳細については、「AWS Key Management ServiceAWS CodeCommit リポジトリの および 暗号化」を参照してください。
8. (オプション) このリポジトリに Java または Python コードが含まれており、CodeGuru Reviewer で分析する場合は、[Enable Amazon CodeGuru Reviewer for Java and Python] (Java および Python 用に Amazon CodeGuru Reviewer を有効化) を選択します。CodeGuru Reviewer

は、複数の機械学習モデルを使用して、コードの欠陥を検出し、プルリクエストの改善と修正を提案します。詳細については、Amazon CodeGuru Reviewer ユーザーガイドを参照してください。

9. [Create] (作成) を選択します。

作成されたリポジトリは、[リポジトリ] リストに表示されます。URL 列で、コピーアイコンを選択し、CodeCommit に接続するために使用するプロトコル (HTTPS または SSH) を選択します。URL をコピーします。

たとえば、リポジトリの名前を *MyFirstRepo* とし、HTTPS を使用している場合、URL は以下のようになります。

```
https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyFirstRepo
```

この URL は後で ステップ 2: ローカルコンテンツを CodeCommit リポジトリに移行する で必要になります。

ステップ 2: ローカルコンテンツを CodeCommit リポジトリに移行する

CodeCommit リポジトリを作成したので、ローカルコンピュータで Git リポジトリに変換するディレクトリを選択できます。git init コマンドは、既存のバージョン管理対象外のコンテンツを Git リポジトリに変換したり、ファイルやコンテンツがまだない場合は、新しい空のリポジトリを初期化したりするために使用できます。

1. ローカルコンピュータでターミナルまたはコマンドラインから、リポジトリのソースとして使用するディレクトリに移動します。
2. 次のコマンドを実行して、**main** という名前のデフォルトのブランチを使用するように Git を設定します。

   ```
   git config --local init.defaultBranch main
   ```

   このコマンドを実行して、新しく作成されたすべてのリポジトリについて、デフォルトのブランチ名を **main** に設定することもできます。

   ```
   git config --global init.defaultBranch main
   ```

3. git init コマンドを実行して、そのディレクトリで Git バージョン管理を初期化します。これにより、そのディレクトリのルートに .git サブディレクトリが作成され、バージョン管理の追跡が有効になります。.git フォルダには、リポジトリに必要なすべてのメタデータも含まれます。

```
git init
```

4. 次のコマンドを実行して、初期化されたディレクトリのステータスを確認します。

```
git status
```

バージョン管理の対象にするファイルを追加します。このチュートリアルでは、`git add` 指定子を付けて `.` コマンドを実行し、すべてのファイルをこのディレクトリに追加します。他のオプションについては、Git のドキュメントを参照してください。

```
git add .
```

5. 追加されたファイルのコミットとコミットメッセージを作成します。

```
git commit -m "Initial commit"
```

6. 送信先 CodeCommit リポジトリの URL と名前、および git push オプションを指定して、`--all` コマンドを実行します。(ここで指定するのは、「ステップ 1: CodeCommit リポジトリを作成する」でコピーした URL です)。

たとえば、リポジトリに *MyFirstRepo* という名前を付け、HTTPS を使用するように設定している場合は、以下のコマンドを実行します。

```
git push https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyFirstRepo --all
```

ステップ 3: CodeCommit でファイルを表示する

ディレクトリの内容をプッシュした後、CodeCommit コンソールを使用して、そのリポジトリ内のすべてのファイルをすばやく表示できます。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. [リポジトリ] で、リストからリポジトリの名前 (例: *MyFirstRepository*) を選択します。
3. ブランチ、クローン URL、設定などのリポジトリ内のファイルを表示します。

ステップ 4: CodeCommit リポジトリを共有する

CodeCommit でリポジトリを作成すると、HTTPS 接続用と SSH 接続用の 2 つのエンドポイントが生成されます。どちらもネットワーク経由で安全な接続を提供します。ユーザーは、いずれかのプロトコルを使用できます。両方のエンドポイントは、どのプロトコルをユーザーに推奨するにしてもアクティブのままです。リポジトリを他のユーザーと共有する前に、リポジトリへのアクセスを他のユーザーに許可する IAM ポリシーを作成する必要があります。これらのアクセス指示をユーザーに提供します。

リポジトリのカスタマー管理ポリシーを作成します。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。
2. [ダッシュボード] ナビゲーションエリアで、[ポリシー] を選択し、次に [ポリシーの作成] を選択します。
3. [ポリシーの作成] ページで、[Import managed policy (マネージドポリシーのインポート)] を選択します。
4. [Import managed policies (マネージドポリシーのインポート)] ページの [フィルタポリシー] に「**`AWSCodeCommitPowerUser`**」と入力します。ポリシー名の横にあるボタンを選択し、[インポート] を選択します。
5. [Create policy] (ポリシーの作成) ページで [JSON] を選択します。次に示すように、CodeCommit アクションの `Resource` 行の「*」部分を、CodeCommit リポジトリの Amazon リソースネーム (ARN) に置き換えます。

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
 ]
```

> **Tip**
>
> CodeCommit リポジトリの ARN を確認するには、CodeCommit コンソールに移動し、リストからリポジトリ名を選択して [Settings] (設定) を選択します。詳細については、「リポジトリの詳細の表示」を参照してください。

このポリシーに複数のリポジトリを適用するには、リソースに ARN を指定して各リポジトリを追加します。次に示すように、各 Resource ステートメントはカンマで区切ります。

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
 "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo"
 ]
```

編集を完了したら、[ポリシーの確認] を選択します。

6. [Review Policy] (ポリシーの確認) ページの [Name] (名前) に、ポリシーの新しい名前 (例えば、*AWSCodeCommitPowerUser-MyDemoRepo*) を入力します。必要に応じて、このポリシーの説明を入力します。
7. [ポリシーの作成] を選択します。

リポジトリへのアクセスを管理するには、リポジトリユーザーの IAM グループを作成し、そのグループに IAM ユーザーを追加します。その後、前のステップで作成したカスタマー管理ポリシーをアタッチします。アクセスに必要なポリシー (`IAMSelfManageServiceSpecificCredentials` または `IAMUserSSHKeys`) をアタッチします。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/:// www.com」で IAM コンソールを開きます。
2. [ダッシュボード] ナビゲーションエリアで、[グループ] を選択し、次に [Create New Group (新しいグループの作成)] を選択します。
3. [Set Group Name] (グループ名の設定) ページの [Group Name] (グループ名) に、グループの名前 (例: *MyDemoRepoGroup*) を入力し、[Next Step] (次のステップ) を選択します。ここで、グループ名の一部として、リポジトリ名を含めることを検討してください。

   > **Note**
   > この名前は、Amazon Web Services アカウント全体で一意である必要があります。

4. 前のセクションで作成したカスタマー管理ポリシー (例: AWSCodeCommitPowerUser-MyDemoRepo) の横にあるボックスをオンにします。
5. [Review] ページで、[Create Group] を選択します。IAM は、指定されたポリシーが既にアタッチされた状態でこのグループを作成します。このグループは、アマゾン ウェブ サービスアカウントに関連付けられたグループのリストに表示されます。
6. リストからグループを選択します。

7. グループの概要ページで、[ユーザー] タブを選択し、次に [Add Users to Group (グループにユーザーを追加)] を選択します。アマゾン ウェブ サービスアカウントに関連付けられているすべてのユーザーを示すリスト上で、CodeCommit リポジトリへのアクセスを許可するユーザーの横にあるボックスをオンにして、[Add Users] (ユーザーを追加) を選択します。

> **Tip**
> 検索ボックスに名前を入力して、ユーザーをすばやく見つけることができます。

8. ユーザーの追加が完了したら、IAM コンソールを閉じます。

設定したポリシーグループとポリシーを使用して、CodeCommit へのアクセスに使用する IAM ユーザーを作成したら、リポジトリへの接続に必要な情報をそのユーザーに送信します。

1. https://console.aws.amazon.com/codesuite/codecommit/home で CodeCommit コンソールを開きます。
2. リージョンセレクタで、リポジトリが作成された AWS リージョン を選択します。リポジトリは に固有です AWS リージョン。詳細については、「リージョンと Git 接続エンドポイント」を参照してください。
3. [リポジトリ] ページで、共有するリポジトリを選択します。
4. [Clone URL] で、ユーザーが使用するプロトコルを選択します。接続プロトコルのクローン URL がコピーされます。
5. のインストール、プロファイルの設定、Git のインストールなど AWS CLI、他の手順とともにクローン URL をユーザーに送信します。接続プロトコルの設定情報を含めるようにしてください (HTTPS など)。

リポジトリを段階的に移行する

に移行するときは AWS CodeCommit、断続的なネットワークの問題やネットワークパフォーマンスの低下によってプッシュ全体が失敗する可能性を減らすために、リポジトリを増分またはチャンクでプッシュすることを検討してください。次のようなスクリプトを使用して、段階的にプッシュすることで、移行を再開して、以前失敗したコミットのみプッシュすることができます。

このトピックの手順では、リポジトリの段階的な移行を行うスクリプトを作成して実行し、移行が完了するまで行われなかった段階的プッシュのみ再度行う方法について説明します。

これらの手順は、「設定 」および「リポジトリの作成」のステップを既に完了していることを前提としています。

トピック

- ステップ 0: 段階的に移行するかどうかを決める
- ステップ 1: 前提条件をインストールし、CodeCommit リポジトリをリモートとして追加する
- ステップ 2: 段階的移行に使用するスクリプトを作成する
- ステップ 3: スクリプトを実行し、CodeCommit に段階的に移行する
- 付録: サンプルスクリプト incremental-repo-migration.py

ステップ 0: 段階的に移行するかどうかを決める

リポジトリの全体サイズと段階的に移行するかどうかを決めるには、複数の要因を検討します。最も重要な要素は、リポジトリのアーティファクトの全体サイズです。リポジトリの累積履歴などの要素もサイズに関連します。各アセットのサイズは大きくなくても、長年の履歴を含むリポジトリやブランチのサイズは非常に大きくなります。これらのリポジトリの移行を単純にし、効率的にするための戦略は多数あります。たとえば、開発期間の長いリポジトリのクローン作成時に浅いクローン戦略を使用したり、大きなバイナリファイルの差分圧縮を無効にしたりできます。Git ドキュメントを確認してオプションを調査するか、段階的なプッシュをセットアップおよび設定して、このトピックのサンプルスクリプト `incremental-repo-migration.py` を使用してリポジトリを移行します。

以下の条件のいずれかに当てはまる場合は、段階的プッシュを設定します。

- 移行するリポジトリの履歴に 5 年以上含まれる
- インターネット接続は、断続的な停止や、削除されたパケットの中断、低速なレスポンスを招くだけでなく、その他のサービスへの停止にもつながります。
- リポジトリの全体サイズが 2 GB を超えているため、リポジトリ全体を移行することを予定します。
- リポジトリには、あまり圧縮されていない大きなアーティファクトやバイナリなどが含まれます。たとえば、追跡されたバージョンが 6 以上ある大きな画像ファイルがあります。
- 過去に CodeCommit への移行を試みて、「内部サービスエラー」メッセージを受け取っています。

上記の条件のいずれも当てはまらない場合でも、段階的なプッシュを行うことができます。

ステップ 1: 前提条件をインストールし、CodeCommit リポジトリをリモートとして追加する

独自のカスタムスクリプトを作成できます。これにより独自の前提条件を指定することができます。このトピックのサンプルを使用する場合は、次のことを行う必要があります。

- 前提条件を満たすこと。
- リポジトリのクローンをローカルコンピュータに作成します。
- 移行するリポジトリのリモートとして、CodeCommit リポジトリを追加します。

incremental-repo-migration.py をセットアップして実行する

1. ローカルコンピュータに Python 2.6 以降をインストールします。詳細と最新バージョンについては、「Python ウェブサイト」を参照してください。
2. 同じコンピュータに GitPython をインストールします。これは、Git リポジトリと通信するために使用する Python ライブラリです。詳細については、「GitPython ドキュメント」を参照してください。
3. git clone --mirror コマンドでは、ローカルコンピュータに移行するリポジトリのクローンを作成します。ターミナル (Linux、macOS、または Unix) またはコマンドプロンプト (Windows) より、git clone --mirror コマンドを使用して、該当リポジトリのローカルリポジトリ (例: ローカルリポジトリを作成するディレクトリ) を作成します。たとえば、URL (*https://example.com/my-repo/*) を持つ Git リポジトリ (*MyMigrationRepo*) のクローンをディレクトリ (*my-repo*) に作成するには、以下のように行います。

   ```
   git clone --mirror https://example.com/my-repo/MyMigrationRepo.git my-repo
   ```

   次のような出力が表示されます。これは、リポジトリのクローンが、ローカルリポジトリ (my-repo) に作成されたことを表します。

   ```
   Cloning into bare repository 'my-repo'...
   remote: Counting objects: 20, done.
   remote: Compressing objects: 100% (17/17), done.
   remote: Total 20 (delta 5), reused 15 (delta 3)
   Unpacking objects: 100% (20/20), done.
   Checking connectivity... done.
   ```

4. クローンを作成したばかりのリポジトリのローカルリポジトリ (例: *my-repo*) にディレクトリを変更します。そのディレクトリから git remote add ***DefaultRemoteName RemoteRepositoryURL*** コマンドを使用し、ローカルリポジトリのリモートリポジトリとして CodeCommit リポジトリを追加します。

> **Note**
>
> 大きなリポジトリをプッシュする場合は、HTTPS ではなく SSH を使用することを検討してください。大きな変更、多数の変更、大きなリポジトリのいずれかをプッシュすると、ネットワーク問題またはファイアウォール設定が原因で、長時間の HTTPS 接続は切断されることがあります。SSH 用に CodeCommit を設定する詳しい方法については、Linux、macOS、または Unix での SSH 接続の場合 または Windows で SSH 接続をセットアップする手順 を参照してください。

例えば、MyDestinationRepo という名前の CodeCommit リポジトリの SSH エンドポイントを `codecommit` という名前のリモートのリモートリポジトリとして追加するには、次のコマンドを使用します。

```
git remote add codecommit ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDestinationRepo
```

> **Tip**
>
> これはクローンのため、デフォルトのリモート名 (`origin`) は既に使用されています。別のリモート名を使用する必要があります。この例では `codecommit` を使用していますが、任意の名前を使用できます。git remote show コマンドでは、ローカルリポジトリに設定されているリモートを一覧表示します。

5. git remote -v コマンドでは、ローカルリポジトリのフェッチおよびプッシュの設定を表示し、これらの設定が正しいことを確認します。例:

```
codecommit  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDestinationRepo
 (fetch)
codecommit  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDestinationRepo
 (push)
```

> **Tip**
> 別のリモートリポジトリのフェッチおよびプッシュのエントリ (例: 送信元のエントリ) が依然として表示される場合は、git remote set-url --delete コマンドを使用してそれらのエントリを削除します。

ステップ 2: 段階的移行に使用するスクリプトを作成する

これらのステップでは、サンプルスクリプト (`incremental-repo-migration.py`) が既に使用されていることを前提としています。

1. テキストエディタを開き、「サンプルスクリプト」の内容を空のドキュメントに貼り付けます。
2. そのドキュメントをドキュメントディレクトリ (ローカルリポジトリの作業ディレクトリではありません) に保存し、ファイル名を `incremental-repo-migration.py` に変更します。選択するディレクトリが、ローカル環境またはパス変数で設定されているディレクトリであることを確認します。これで、コマンドラインまたはターミナルより、Python スクリプトを実行できるようになりました。

ステップ 3: スクリプトを実行し、CodeCommit に段階的に移行する

以上で作成された `incremental-repo-migration.py` スクリプトを使用して、ローカルリポジトリを CodeCommit リポジトリに段階的に移行できます。デフォルトでは、スクリプトは 1,000 コミットのバッチを使用したコミットをプッシュし、ローカルリポジトリおよびリモートリポジトリの設定として実行されるディレクトリの Git 設定を使用します。必要に応じて、`incremental-repo-migration.py` のオプションを使用し、他の設定を行うことができます。

1. ターミナルまたはコマンドプロンプトより、移行するローカルリポジトリにディレクトリを変更します。
2. そのディレクトリから、次のコマンドを実行します。

```
python incremental-repo-migration.py
```

3. スクリプトが実行され、ターミナルまたはコマンドプロンプトに進行状況が表示されます。リポジトリの大きさによっては、進行状況が遅れて表示される場合があります。単一プッシュが 3 度失敗すると、スクリプトは停止します。その後スクリプトを返します。失敗したバッチから開

始できます。すべてのプッシュが継続し、移行が完了するまで、スクリプトを返すことができます。

> **Tip**
>
> -l および -r オプションで、使用するローカル設定およびリモート設定が指定されている限り、任意のディレクトリより incremental-repo-migration.py を実行できます。たとえば、任意のディレクトリよりスクリプトを使用して、/tmp/*my-repo* のローカルリポジトリをリモート (*codecommit*) に移行するには、次のように行います。
>
> ```
> python incremental-repo-migration.py -l "/tmp/my-repo" -r "codecommit"
> ```
>
> また、-b オプションを使用して、段階的にプッシュする際に使用するデフォルトのバッチサイズを変更することもできます。たとえば、変更頻度の高い大きなバイナリファイルを含むリポジトリを定期的にプッシュし、ネットワーク帯域幅が制限された場所から操作する場合は、-b オプションを使用して、バッチサイズを 1,000 ではなく 500 に変更します。例:
>
> ```
> python incremental-repo-migration.py -b 500
> ```
>
> これにより、ローカルリポジトリは、500 コミットのバッチを使用して、段階的にプッシュされます。リポジトリ移行時にバッチサイズを再度変更する場合 (例: 試行失敗時にバッチサイズを小さくする場合) は、-b を使用してバッチサイズをリセットする前に、-c オプションを使用してバッチタグを削除してください。
>
> ```
> python incremental-repo-migration.py -c
> python incremental-repo-migration.py -b 250
> ```

> **Important**
>
> 失敗後にスクリプトを返す場合は、絶対に -c オプションを使用しないでください。-c オプションでは、コミットのバッチに使用するタグが削除されます。バッチサイズを変更して再開する場合、または今後スクリプトを使用しない場合は、-c オプションを使用します。

付録: サンプルスクリプト incremental-repo-migration.py

参考用に、Python のサンプルスクリプト incremental-repo-migration.py を用意しています。このスクリプトでは、リポジトリを段階的にプッシュします。このスクリプトは、オープンソースコードのサンプルで、現状のまま提供されています。

```
# Copyright 2015 Amazon.com, Inc. or its affiliates. All Rights Reserved. Licensed
 under the Amazon Software License (the "License").
# You may not use this file except in compliance with the License. A copy of the
 License is located at
#    http://aws.amazon.com/asl/
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 KIND, express or implied. See the License for
# the specific language governing permissions and limitations under the License.

#!/usr/bin/env python

import os
import sys
from optparse import OptionParser
from git import Repo, TagReference, RemoteProgress, GitCommandError


class PushProgressPrinter(RemoteProgress):
    def update(self, op_code, cur_count, max_count=None, message=""):
        op_id = op_code & self.OP_MASK
        stage_id = op_code & self.STAGE_MASK
        if op_id == self.WRITING and stage_id == self.BEGIN:
            print("\tObjects: %d" % max_count)


class RepositoryMigration:
    MAX_COMMITS_TOLERANCE_PERCENT = 0.05
    PUSH_RETRY_LIMIT = 3
    MIGRATION_TAG_PREFIX = "codecommit_migration_"

    def migrate_repository_in_parts(
        self, repo_dir, remote_name, commit_batch_size, clean
    ):
        self.next_tag_number = 0
        self.migration_tags = []
        self.walked_commits = set()
        self.local_repo = Repo(repo_dir)
```

```
        self.remote_name = remote_name
        self.max_commits_per_push = commit_batch_size
        self.max_commits_tolerance = (
            self.max_commits_per_push * self.MAX_COMMITS_TOLERANCE_PERCENT
        )

        try:
            self.remote_repo = self.local_repo.remote(remote_name)
            self.get_remote_migration_tags()
        except (ValueError, GitCommandError):
            print(
                "Could not contact the remote repository. The most common reasons for
this error are that the name of the remote repository is incorrect, or that you do not
have permissions to interact with that remote repository."
            )
            sys.exit(1)

        if clean:
            self.clean_up(clean_up_remote=True)
            return

        self.clean_up()

        print("Analyzing repository")
        head_commit = self.local_repo.head.commit
        sys.setrecursionlimit(max(sys.getrecursionlimit(), head_commit.count()))

        # tag commits on default branch
        leftover_commits = self.migrate_commit(head_commit)
        self.tag_commits([commit for (commit, commit_count) in leftover_commits])

        # tag commits on each branch
        for branch in self.local_repo.heads:
            leftover_commits = self.migrate_commit(branch.commit)
            self.tag_commits([commit for (commit, commit_count) in leftover_commits])

        # push the tags
        self.push_migration_tags()

        # push all branch references
        for branch in self.local_repo.heads:
            print("Pushing branch %s" % branch.name)
            self.do_push_with_retries(ref=branch.name)
```

```
        # push all tags
        print("Pushing tags")
        self.do_push_with_retries(push_tags=True)

        self.get_remote_migration_tags()
        self.clean_up(clean_up_remote=True)

        print("Migration to CodeCommit was successful")

    def migrate_commit(self, commit):
        if commit in self.walked_commits:
            return []

        pending_ancestor_pushes = []
        commit_count = 1

        if len(commit.parents) > 1:
            # This is a merge commit
            # Ensure that all parents are pushed first
            for parent_commit in commit.parents:
                pending_ancestor_pushes.extend(self.migrate_commit(parent_commit))
        elif len(commit.parents) == 1:
            # Split linear history into individual pushes
            next_ancestor, commits_to_next_ancestor = self.find_next_ancestor_for_push(
                commit.parents[0]
            )
            commit_count += commits_to_next_ancestor
            pending_ancestor_pushes.extend(self.migrate_commit(next_ancestor))

        self.walked_commits.add(commit)

        return self.stage_push(commit, commit_count, pending_ancestor_pushes)

    def find_next_ancestor_for_push(self, commit):
        commit_count = 0

        # Traverse linear history until we reach our commit limit, a merge commit, or
 an initial commit
        while (
            len(commit.parents) == 1
            and commit_count < self.max_commits_per_push
            and commit not in self.walked_commits
        ):
            commit_count += 1
```

```
            self.walked_commits.add(commit)
            commit = commit.parents[0]

        return commit, commit_count

    def stage_push(self, commit, commit_count, pending_ancestor_pushes):
        # Determine whether we can roll up pending ancestor pushes into this push
        combined_commit_count = commit_count + sum(
            ancestor_commit_count
            for (ancestor, ancestor_commit_count) in pending_ancestor_pushes
        )

        if combined_commit_count < self.max_commits_per_push:
            # don't push anything, roll up all pending ancestor pushes into this
 pending push
            return [(commit, combined_commit_count)]

        if combined_commit_count <= (
            self.max_commits_per_push + self.max_commits_tolerance
        ):
            # roll up everything into this commit and push
            self.tag_commits([commit])
            return []

        if commit_count >= self.max_commits_per_push:
            # need to push each pending ancestor and this commit
            self.tag_commits(
                [
                    ancestor
                    for (ancestor, ancestor_commit_count) in pending_ancestor_pushes
                ]
            )
            self.tag_commits([commit])
            return []

        # push each pending ancestor, but roll up this commit
        self.tag_commits(
            [ancestor for (ancestor, ancestor_commit_count) in pending_ancestor_pushes]
        )
        return [(commit, commit_count)]

    def tag_commits(self, commits):
        for commit in commits:
            self.next_tag_number += 1
```

```
            tag_name = self.MIGRATION_TAG_PREFIX + str(self.next_tag_number)

            if tag_name not in self.remote_migration_tags:
                tag = self.local_repo.create_tag(tag_name, ref=commit)
                self.migration_tags.append(tag)
            elif self.remote_migration_tags[tag_name] != str(commit):
                print(
                    "Migration tags on the remote do not match the local tags. Most
likely your batch size has changed since the last time you ran this script. Please run
this script with the --clean option, and try again."
                )
                sys.exit(1)

    def push_migration_tags(self):
        print("Will attempt to push %d tags" % len(self.migration_tags))
        self.migration_tags.sort(
            key=lambda tag: int(tag.name.replace(self.MIGRATION_TAG_PREFIX, ""))
        )
        for tag in self.migration_tags:
            print(
                "Pushing tag %s (out of %d tags), commit %s"
                % (tag.name, self.next_tag_number, str(tag.commit))
            )
            self.do_push_with_retries(ref=tag.name)

    def do_push_with_retries(self, ref=None, push_tags=False):
        for i in range(0, self.PUSH_RETRY_LIMIT):
            if i == 0:
                progress_printer = PushProgressPrinter()
            else:
                progress_printer = None

            try:
                if push_tags:
                    infos = self.remote_repo.push(tags=True, progress=progress_printer)
                elif ref is not None:
                    infos = self.remote_repo.push(
                        refspec=ref, progress=progress_printer
                    )
                else:
                    infos = self.remote_repo.push(progress=progress_printer)

                success = True
                if len(infos) == 0:
```

```
                    success = False
                else:
                    for info in infos:
                        if (
                            info.flags & info.UP_TO_DATE
                            or info.flags & info.NEW_TAG
                            or info.flags & info.NEW_HEAD
                        ):
                            continue
                        success = False
                        print(info.summary)

                if success:
                    return
            except GitCommandError as err:
                print(err)

        if push_tags:
            print("Pushing all tags failed after %d attempts" %
(self.PUSH_RETRY_LIMIT))
        elif ref is not None:
            print("Pushing %s failed after %d attempts" % (ref, self.PUSH_RETRY_LIMIT))
            print(
                "For more information about the cause of this error, run the following
command from the local repo: 'git push %s %s'"
                % (self.remote_name, ref)
            )
        else:
            print(
                "Pushing all branches failed after %d attempts"
                % (self.PUSH_RETRY_LIMIT)
            )
        sys.exit(1)

    def get_remote_migration_tags(self):
        remote_tags_output = self.local_repo.git.ls_remote(
            self.remote_name, tags=True
        ).split("\n")
        self.remote_migration_tags = dict(
            (tag.split()[1].replace("refs/tags/", ""), tag.split()[0])
            for tag in remote_tags_output
            if self.MIGRATION_TAG_PREFIX in tag
        )
```

```
    def clean_up(self, clean_up_remote=False):
        tags = [
            tag
            for tag in self.local_repo.tags
            if tag.name.startswith(self.MIGRATION_TAG_PREFIX)
        ]

        # delete the local tags
        TagReference.delete(self.local_repo, *tags)

        # delete the remote tags
        if clean_up_remote:
            tags_to_delete = [":" + tag_name for tag_name in
 self.remote_migration_tags]
            self.remote_repo.push(refspec=tags_to_delete)


parser = OptionParser()
parser.add_option(
    "-l",
    "--local",
    action="store",
    dest="localrepo",
    default=os.getcwd(),
    help="The path to the local repo. If this option is not specified, the script will
 attempt to use current directory by default. If it is not a local git repo, the script
 will fail.",
)
parser.add_option(
    "-r",
    "--remote",
    action="store",
    dest="remoterepo",
    default="codecommit",
    help="The name of the remote repository to be used as the push or migration
 destination. The remote must already be set in the local repo ('git remote add ...').
 If this option is not specified, the script will use 'codecommit' by default.",
)
parser.add_option(
    "-b",
    "--batch",
    action="store",
    dest="batchsize",
    default="1000",
```

```
    help="Specifies the commit batch size for pushes. If not explicitly set, the
 default is 1,000 commits.",
)
parser.add_option(
    "-c",
    "--clean",
    action="store_true",
    dest="clean",
    default=False,
    help="Remove the temporary tags created by migration from both the local repo
 and the remote repository. This option will not do any migration work, just cleanup.
 Cleanup is done automatically at the end of a successful migration, but not after a
 failure so that when you re-run the script, the tags from the prior run can be used to
 identify commit batches that were not pushed successfully.",
)

(options, args) = parser.parse_args()

migration = RepositoryMigration()
migration.migrate_repository_in_parts(
    options.localrepo, options.remoterepo, int(options.batchsize), options.clean
)
```

のセキュリティ AWS CodeCommit

のクラウドセキュリティが最優先事項 AWS です。 AWS カスタマーは、最もセキュリティの影響を受けやすい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を活用できます。

セキュリティは、 AWS とユーザーの間で共有される責任です。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説明しています。

- クラウドのセキュリティ – AWS クラウドで AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。 AWS また、 では、安全に使用できるサービスも提供しています。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、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。が適用されるコンプライアンスプログラムの詳細については AWS CodeCommit、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウド内のセキュリティ — お客様の責任は、使用する AWS サービスによって決まります。また、ユーザーは、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、CodeCommit を使用する際の責任共有モデルの適用方法を理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するように CodeCommit を設定する方法について説明します。また、CodeCommit リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

- でのデータ保護 AWS CodeCommit
- の Identity and Access Management AWS CodeCommit
- の耐障害性 AWS CodeCommit
- のインフラストラクチャセキュリティ AWS CodeCommit

でのデータ保護 AWS CodeCommit

マネージドサービスとして、 は AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS「 クラウドセキュリティ」を参照してください。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して AWS 環境を設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS 公開された API コールを使用して、ネットワーク経由で にアクセスします。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)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。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。

また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられている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) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

保存時の CodeCommit リポジトリは自動的に暗号化されます。お客様の操作は必要ありません。CodeCommit は、転送中のリポジトリデータも暗号化します。CodeCommit リポジトリでは、HTTPS プロトコル、SSH プロトコル、またはその両方を使用できます。詳細については、「のセットアップ AWS CodeCommit」を参照してください。CodeCommit リポジトリへの cross-account アクセス を設定することもできます。

トピック

- AWS Key Management ServiceAWS CodeCommit リポジトリの および 暗号化
- 認証情報をローテーションして AWS CodeCommit リポジトリに接続する

AWS Key Management ServiceAWS CodeCommit リポジトリの および 暗号化

CodeCommit リポジトリ内のデータは、転送中と不使用時のいずれも暗号化されます。データが CodeCommit リポジトリにプッシュされると (git push を呼び出すことによってプッシュされる場合など)、リポジトリに格納されると同時に、CodeCommit は受信データを暗号化します。データが CodeCommit リポジトリからプルされると (例: git pull を呼び出す)、CodeCommit は、データを復号し、発信者にその復号データを送信します。これは、プッシュリクエストまたはプルリクエストに関連付けられた IAM ユーザーが によって認証されていることを前提としています AWS。送信データまたは受信データは、HTTPS または SSH で暗号化されたネットワークプロトコルを使用して送信されます。

AWS マネージドキー またはカスタマーマネージドキーを使用して、リポジトリ内のデータを暗号化および復号できます。カスタマーマネージドキーと AWS マネージドキーの違いの詳細については、「カスタマーキーと AWS マネージドキー」を参照してください。カスタマーマネージドキーを指定しない場合、CodeCommit はリポジトリ内のデータの暗号化と復号に AWS マネージドキー を使用します。この AWS マネージドキー は AWS アカウントで自動的に作成されます。 AWS リージョン Amazon Web Services アカウントの新しい に CodeCommit リポジトリを初めて作成するときに、カスタマーマネージドキーを指定しない場合、CodeCommit は AWS マネージドキー () の同じ AWS リージョン に AWS Key Management Service （ `aws/codecommit`キー) を作成しますAWS KMS。この `aws/codecommit` キーは CodeCommit によってのみ使用されます。これは、アマゾン ウェブサービスアカウントに保存されます。指定した内容に応じて、CodeCommit はカスタマーマネージドキーまたは AWS マネージドキー を使用してリポジトリ内のデータを暗号化および復号します。

> **Important**
>
> CodeCommit は、リポジトリ内のデータの暗号化と復号に使用される AWS KMS キーに対して次の AWS KMS アクションを実行します。を使用している場合 AWS マネージドキー、ユーザーはこれらのアクションに対する明示的なアクセス許可を必要としませんが、`aws/codecommit`キーのこれらのアクションを拒否するポリシーをアタッチすることはできません。 AWS アカウント ID がそのキーのポリシープリンシパルとして設定されたカスタマーマネージドキーを使用している場合は、これらのアクセス許可を明示的に に設定する必要がありますallow。具体的には、最初のリポジトリを作成し、リポジトリのキーを更新する場合、 を使用している場合denyは、次のアクセス許可を に設定しないでください。ポリシープリンシパルでカスタマーマネージドキーを使用している場合allowは AWS マネージドキー、 を に設定する必要があります。
>
> - `"kms:Encrypt"`
> - `"kms:Decrypt"`
> - `"kms:ReEncrypt"` (コンテキストによっては、これには `kms:ReEncryptFrom`、
>
> `kms:ReEncryptTo`、または `kms:ReEncrypt*` が拒否に設定されていないことが必要になる場合があります。)
> - `"kms:GenerateDataKey"`
> - `"kms:GenerateDataKeyWithoutPlaintext"`
> - `"kms:DescribeKey"`

独自のカスタマーマネージドキーを使用する場合は、リポジトリが存在する AWS リージョン でキーが使用可能である必要があります。CodeCommit は、単一リージョンとマルチリージョンの両方のカスタマーマネージドキーの使用をサポートします。キーマテリアルのオリジンタイプはすべてサポートされていますが、デフォルトの [KMS] オプションを使用することをお勧めします。[外部キーストア] オプションを使用する場合、ストアプロバイダによる遅延が発生する可能性があります。加えて: CodeCommit では、カスタマーマネージドキーを使用する場合、次の要件があります。

- CodeCommit は対称キーの使用のみをサポートします。
- キーの使用タイプは [暗号化と復号化] に設定する必要があります。

カスタマーマネージドキーの作成の詳細については、「概念」および「キーの作成」を参照してください。

CodeCommit によって AWS マネージドキー 生成された に関する情報を表示するには、次の手順を実行します。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/kms://www.com で AWS Key Management Service（AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. サービスナビゲーションペインで、[AWS マネージドキー] を選択します。キー AWS リージョン を確認する にサインインしていることを確認します。
4. 暗号化キーのリストで、エイリアス aws/codecommit AWS マネージドキー を持つ を選択します。に関する基本情報 AWS 所有のキー が表示されます。

これを変更または削除することはできません AWS マネージドキー。

暗号化アルゴリズムを使用してリポジトリデータを暗号化する方法

CodeCommit では、2 つの異なる方法でデータを暗号化します。6 MB 未満の個々の Git オブジェクトは、データ整合性の検証を提供する AES-GCM-256 を使用して暗号化されます。1 つの blob の 6 MB から最大 2 GB のオブジェクトは、AES-CBC-256 を使用して暗号化されます。CodeCommit は常に暗号化コンテキストを検証します。

暗号化コンテキスト

と統合されている各サービスは、暗号化オペレーションと復号オペレーションの両方の暗号化コンテキスト AWS KMS を指定します。暗号化コンテキストは、データの整合性を調べるために AWS KMS で使用される追加の認証情報です。暗号化オペレーションに対して暗号化コンテキストを指定

するときは、復号オペレーションに対しても指定する必要があります。そうしないと、復号は失敗します。CodeCommit は、暗号化コンテキストに CodeCommit リポジトリ ID を使用します。get-repository コマンドまたは CodeCommit コンソールを使用してリポジトリ ID を見つけることができます。 AWS CloudTrail ログで CodeCommit リポジトリ ID を検索して、CodeCommit リポジトリ内のデータを暗号化または復号 AWS KMS する のキーに対して実行された暗号化オペレーションを理解します。

詳細については AWS KMS、「 AWS Key Management Service デベロッパーガイド」を参照してください。

認証情報をローテーションして AWS CodeCommit リポジトリに接続する

IAM ユーザーを設定したり、アクセスキーとシークレットキーを使用したりすることなく、 AWS CodeCommit リポジトリへのアクセスをユーザーに許可できます。フェデレーテッドアイデンティティ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) 用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、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。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。また、IAM ユーザーが別のアマゾン ウェブ サービスアカウントの CodeCommit リポジトリにアクセスするためのロールベースのアクセスを設定することもできます (cross-account アクセスと呼ばれるテクニック)。リポジトリへのクロスアカウントアクセスを設定するためのウォークスルーは、「ロールを使用して AWS CodeCommit リポジトリへのクロスアカウントアクセスを設定する」を参照してください。

以下のような方法で認証する必要のあるユーザーに、アクセスを設定できます。

- Security Assertion Markup Language (SAML)
- 多要素認証 (MFA)
- フェデレーション
- Login with Amazon
- Amazon Cognito
- Facebook
- Google
- OpenID Connect (OIDC) 対応の ID プロバイダー

> **Note**
>
> 以下の情報は、CodeCommit リポジトリに接続するための git-remote-codecommitまたは AWS CLI 認証情報ヘルパーの使用にのみ適用されます。CodeCommit への一時アクセスまたはフェデレーティッドアクセスに対して推奨されるアプローチは git-remote-codecommit を設定することであるため、このトピックでは、このユーティリティの使用例を示します。詳細については、「git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」を参照してください。
> アクセス認証情報の更新や一時的なアクセス認証情報で CodeCommit リポジトリに接続するのに、SSH あるいは Git 認証情報および HTTPS を使用することはできません。

以下のすべての要件が当てはまる場合には、これらのステップを実行する必要はありません。

- Amazon EC2 インスタンスにサインインしている。
- AWS CLI 認証情報ヘルパーで Git と HTTPS を使用して、Amazon EC2 インスタンスから CodeCommit リポジトリに接続します。
- Amazon EC2 インスタンスに、 AWS CLI 認証情報ヘルパーを使用した Linux、macOS、または Unix での HTTPS 接続の場合 または AWS CLI 認証情報ヘルパーを使用した Windows での HTTPS 接続の場合 で説明されているアクセス許可を含む IAM インスタンスプロファイルがアタッチされている。
- AWS CLI 認証情報ヘルパーを使用した Linux、macOS、または Unix での HTTPS 接続の場合 または AWS CLI 認証情報ヘルパーを使用した Windows での HTTPS 接続の場合 で説明されているように、Amazon EC2 インスタンスに Git 認証情報ヘルパーがインストールおよび設定されている。

上記の要件を満たす Amazon EC2 インスタンスは、お客様に代わって CodeCommit に一時アクセス認証情報を提供するように、既に設定されています。

> **Note**
>
> Amazon EC2 インスタンスでは、git-remote-codecommit を設定および使用することができます。

ユーザーに CodeCommit リポジトリへの一時アクセスを許可するには、以下の手順を実行します。

ステップ 1: 前提条件を満たす

設定ステップを実行し、認証情報の更新を使用して CodeCommit リポジトリへのアクセスをユーザーに許可します。

- cross-account アクセスについては、チュートリアル: IAM ロールを使用したアマゾン ウェブ サービスアカウント間でのアクセスの委任および ロールを使用して AWS CodeCommit リポジトリへのクロスアカウントアクセスを設定する を参照してください。
- SAML とフェデレーションについては、「組織の認証システムを使用して AWS リソースへのアクセスを許可する」および「SAML 2.0 AWS STS ベースのフェデレーションについて」を参照してください。
- MFA については、AWSでの多要素認証 (MFA) デバイスの使用と IAM ユーザーにアクセスを許可する一時的なセキュリティ認証情報の作成を参照してください。
- Login with Amazon、Amazon Cognito、Facebook、Google、または OIDC 互換 ID プロバイダーについては、AWS STS 「ウェブ ID フェデレーションについて」を参照してください。

AWS CodeCommitの認証とアクセスコントロール の情報を使用して、ユーザーに付与する CodeCommit アクセス許可を指定します。

ステップ 2: ロール名またはアクセス認証情報を取得する

ユーザーがロールを引き受けることによってリポジトリにアクセスできるようにするには、そのロールの Amazon リソースネーム (ARN) をユーザーに提供します。それ以外の場合、アクセス権の設定方法に応じて、ユーザーは次のいずれかの方法で認証情報の更新を取得できます。

- クロスアカウントアクセスの場合は、the AWS CLI assume-role コマンドを呼び出すか AWS STS AssumeRole API を呼び出します。
- SAML の場合は、 AWS CLI assume-role-with-saml コマンドまたは AWS STS AssumeRoleWithSAML API を呼び出します。
- フェデレーションの場合は、the AWS CLI assume-role または get-federation-token コマンド、AWS STS AssumeRole または GetFederationToken APIs。
- MFA の場合は、 AWS CLI get-session-token コマンドまたは AWS STS GetSessionToken API を呼び出します。
- Login with Amazon、Amazon Cognito、Facebook、Google、または OIDC 互換の ID プロバイダーの場合は、 AWS CLI assume-role-with-web-identity コマンドまたは AWS STS AssumeRoleWithWebIdentity API を呼び出します。

ステップ 3: git-remote-codecommit をインストールして を設定する AWS CLI

AWS CLIで git-remote-codecommit をインストールしてプロファイルを設定することで、アクセス認証情報を使用するようにローカルコンピュータを設定する必要があります。

1. 「設定 」の指示に従って、 AWS CLIを設定します。1 つまたは複数のプロファイルを設定するには、aws configure コマンドを使用します。認証情報の更新を使用して CodeCommit リポジトリに接続するときに使用する、名前付きプロファイルを作成することを検討してください。
2. 次のいずれかの方法で、認証情報をユーザー AWS CLI の名前付きプロファイルに関連付けることができます。

   - CodeCommit にアクセスするロールを引き受ける場合は、そのロールを引き受けるのに必要な情報を含む名前付きプロファイルを設定します。例えば、Amazon Web Services アカウント 111111111111「」で *CodeCommitAccess* という名前のロールを引き受ける場合、他の AWS リソースを操作するときに使用するデフォルトのプロファイルと、そのロールを引き受けるときに使用する名前付きプロファイルを設定できます。次のコマンドは、*CodeCommitAccess* という名前のロールを引き受ける、名前付きプロファイルの *CodeAccess* を作成します。*Maria_Garcia* というユーザー名はセッションに関連付けられ、デフォルトのプロファイルはその AWS 認証情報のソースとして設定されます。

     ```
     aws configure set role_arn arn:aws:iam::111111111111:role/CodeCommitAccess --
     profile CodeAccess
     aws configure set source_profile default --profile CodeAccess
     aws configure set role_session_name "Maria_Garcia" --profile CodeAccess
     ```

     変更を確認する場合は、`~/.aws/config` ファイル (Linux の場合) または `%UserProfile%.aws\config`ファイル (Windows の場合) を手動で表示または編集し、名前付きプロファイルの情報を確認します。たとえば、ファイルは次のようになります。

     ```
     [default]
     region = us-east-1
     output = json

     [profile CodeAccess]
     source_profile = default
     role_session_name = Maria_Garcia
     role_arn = arn:aws:iam::111111111111:role/CodeCommitAccess
     ```

名前付きプロファイルを設定したら、名前付きプロファイルを使用して git-remote-codecommit ユーティリティで CodeCommit リポジトリのクローンを作成できます。例えば、*MyDemoRepo* という名前のリポジトリのクローンを作成するには、次のようにします。

```
git clone codecommit://CodeAccess@MyDemoRepo
```

- ウェブ ID フェデレーションと OpenID Connect (OIDC) を使用している場合は、ユーザーに代わって AWS Security Token Service（AWS STS) `AssumeRoleWithWebIdentity` API コールを行い、一時的な認証情報を更新する名前付きプロファイルを設定します。aws configure set コマンドを使用するか、 `~/.aws/credentials` ファイル (Linux の場合) または `%UserProfile%.aws\credentials` ファイル (Windows の場合) を手動で編集して、必要な設定値を持つ AWS CLI 名前付きプロファイルを追加します。例えば、*CodeCommitAccess* ロールを引き受け、ウェブ ID トークンファイル *~/my-credentials/my-token-file* を使用するプロファイルを作成するには、次のようにします。

```
[CodeCommitWebIdentity]
role_arn = arn:aws:iam::111111111111:role/CodeCommitAccess
web_identity_token_file=~/my-credentials/my-token-file
role_session_name = Maria_Garcia
```

詳細については、AWS Command Line Interface ユーザーガイドのAWS Command Line Interfaceの設定とAWS CLIでの IAM ロールの使用を参照してください。

ステップ 4: CodeCommit リポジトリにアクセスする

ユーザーが リポジトリへの接続 の指示に従って CodeCommit リポジトリに接続したとすると、そのユーザーは git-remote-codecommit によって提供される拡張機能と Git を使用して git clone、git push、および git pull を呼び出し、アクセスが許可されている CodeCommit リポジトリに対して、クローン作成、プッシュ、プルを行います。たとえば、リポジトリのクローンを作成するには次のようにします。

```
git clone codecommit://CodeAccess@MyDemoRepo
```

Git のコミット、プッシュ、プルコマンドは、通常の Git 構文を使用します。

ユーザーが を使用して、ローテーションするアクセス認証情報に関連付けられた AWS CLI 名前付きプロファイルを指定する AWS CLI と、そのプロファイルの範囲が指定された結果が返されます。

の Identity and Access Management AWS CodeCommit

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 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に CodeCommit リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス 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 CodeCommitの認証とアクセスコントロール
- が IAM と AWS CodeCommit 連携する方法
- CodeCommit のリソースベースのポリシー
- CodeCommit タグに基づく認可
- CodeCommit IAM ロール
- AWS CodeCommit アイデンティティベースのポリシーの例
- AWS CodeCommit ID とアクセスのトラブルシューティング

対象者

AWS Identity and Access Management (IAM) の使用方法は、CodeCommit で行う作業によって異なります。

[Service user] (サービスユーザー) – CodeCommit サービスを使用してジョブを実行する場合は、必要なアクセス許可と認証情報を管理者が用意します。さらに多くの CodeCommit 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解すると、管理者から適切なアクセス許可をリクエストするのに役に立ちます。CodeCommit の機能にアクセスできない場合は、AWS CodeCommit ID とアクセスのトラブルシューティング を参照してください。

サービス管理者 – 社内の CodeCommit リソースを担当している場合は、通常、CodeCommit へのフルアクセスがあります。サービスのユーザーがどの CodeCommit 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を確認して、IAM の基本概念を理解してください。会社で CodeCommit を使用して IAM を利用する方法の詳細については、が IAM と AWS CodeCommit 連携する方法 を参照してください。

IAM 管理者 – 管理者は、CodeCommit へのアクセスを管理するポリシーの作成方法の詳細について確認する場合があります。IAM で使用できる CodeCommit アイデンティティベースのポリシーの例を表示するには、AWS CodeCommit アイデンティティベースのポリシーの例 を参照してください。

アイデンティティを使用した認証

認証は、ID 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 ロールを引き受けることによって認証（ にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド 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 認証情報は、フェデレーティッド ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。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ることになります。

ユーザーの種類に応じて、 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を暗号化して署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに自分で署名する推奨方法の使用については、「IAM ユーザーガイド」の「API リクエストに対するAWS Signature Version 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、 AWS では、多要素認証 (MFA) を使用してアカウントのセキュリティを向上させる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」および「IAM ユーザーガイド」の「IAM のAWS 多要素認証」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することでアクセスできます。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「ルートユーザー認証情報が必要なタスク」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一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保有する 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 ユーザーでの長期的な認証情報が必要な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、「IAM 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ために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 グループは、IAM ユーザーの集団を指定するアイデンティティです。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、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

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、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

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 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。で 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。ロールを引き受けるには、 または AWS API オペレーションを AWS CLI 呼び出すか、カスタム URL を使用します。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ための各種方法」を参照してください。

IAM ロールと一時的な認証情報は、次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) 用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、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。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 ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで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、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では 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におけるロールとリソースベースのポリシーの違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。
- クロスサービスアクセス — 一部の は他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。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と 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アクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除することができま

す。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを作成する」を参照してください。

- 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、サービスリンク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 EC2 で実行されているアプリケーション – 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 インスタンス内でのアクセスキーの保存に推奨されます。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、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM 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 IAM ロールを使用して許可を付与する」を参照してください。

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 AWS には、ポリシーを作成し、ID AWS 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 のオブジェクトです。 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 ポリシー概要を参照してください。

管理者は JSON AWS ポリシーを使用して、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。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、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方法を問わず、アクションの許可を定義します。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 API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 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。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

アイデンティティベースのポリシーは、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。管理ポリシーは、 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。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理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のいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

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、リソースに添付する 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には例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。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制御できます。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、フェデレーティッドユーザー、または 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では、IAM の AWS マネージドポリシーを使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) 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ための許可を持つかを制御します。ACL 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、 AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、「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) の概要」を参照してください。

その他のポリシータイプ

AWS は、追加のあま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ポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、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アクセス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、アクセス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 (SCP) を一部またはすべてのアカウントに適用できます。SCP 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を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー (SCP)」を参照してください。
- リソースコントロールポリシー (RCP) – RCP は、所有する各リソースにアタッチされた IAM ポリシーを更新することなく、アカウント内のリソースに利用可能な最大数のアクセス許可を設定するために使用できる JSON ポリシーです。RCP は、メンバーアカウントのリソースに対するアクセス許可を制限し、組織に属するかどうかにかかわらず AWS アカウントのルートユーザー、 を含む ID に対する有効なアクセス許可に影響を与える可能性があります。RCP AWS のサービス をサポートする のリストを含む Organizations と RCPs「リソースコントロールポリシー (RCPs」を参照してください。 AWS Organizations
- 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もあります。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、「IAM 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。

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係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、IAM 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。

AWS CodeCommitの認証とアクセスコントロール

へのアクセスには認証情報 AWS CodeCommit が必要です。これらの認証情報には、CodeCommit リポジトリや IAM ユーザーなどの AWS リソースにアクセスするためのアクセス許可が必要です。IAM ユーザーを使用して、Git 認証情報や Git 接続に使用する SSH パブリックキーを管理します。以下のセクションでは、リソースへの安全なアクセスのために、AWS Identity and Access Management (IAM) と CodeCommit を役立てる方法について詳しく説明します。

- 認証
- アクセスコントロール

認証

CodeCommit リポジトリは Git ベースで Git 認証情報を含む Git の基本機能をサポートしているので、CodeCommit で作業する場合は、IAM ユーザーを使用することをお勧めします。他のアイデンティティタイプで CodeCommit にアクセスすることができますが、その他のアイデンティティタイプは、以下に説明するように制限されることがあります。

アイデンティティタイプ

- IAM ユーザー – IAM ユーザーは、単に特定のカスタムアクセス許可を持つ、Amazon Web Services アカウント内のアイデンティティです。例えば、IAM ユーザーは、CodeCommit リポジトリにアクセスするための Git 認証情報を作成および管理するためのアクセス許可を持つことができます。以下は、CodeCommit での作業に推奨されるユーザータイプです。IAM ユーザー名とパスワードを使用して、、 AWS ディスカッションフォーラムAWS Management Console、 AWS サポート センターなどの安全な AWS ウェブページにサインインできます。

  Git の認証情報を生成するか SSH パブリックキーを IAM ユーザーに関連付ける、または git-remote-codecommit をインストールして設定することができます。これらは、CodeCommit リポジトリと連携するように Git を設定する最も簡単な方法です。Git 認証情報で、IAM の静的ユーザー名とパスワードを生成します。HTTPS 接続のためのこれらの認証情報は、Git だけでな

く、Git のユーザー名およびパスワード認証をサポートするサードパーティーツールでも使用します。SSH 接続では、SSH 認証用に Git および CodeCommit が使用するパブリックキーとプライベートキーのファイルをローカルマシンで作成します。パブリックキーを IAM ユーザーに関連付けて、プライベートキーをローカルマシンに保存します。git-remote-codecommit は Git 自体を拡張するため、ユーザーの Git 認証情報を設定する必要はありません。

また、各ユーザーのアクセスキーを生成することができます。 AWS サービスにプログラムでアクセスするときは、AWS SDKs のいずれかまたは AWS Command Line Interface（AWS CLI）を使用してアクセスキーを使用します。SDK と CLI ツールでは、アクセスキーを使用してリクエストが暗号で署名されます。 AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。CodeCommit では、署名バージョン 4 がサポートされています。これは、インバウンド API リクエストを認証するためのプロトコルです。リクエストの認証の詳細については、『』の「署名バージョン 4 の署名プロセスAWS 全般のリファレンス」を参照してください。

- Amazon Web Services アカウントのルートユーザー – にサインアップするときに AWS、Amazon Web Services アカウントに関連付けられた E メールアドレスとパスワードを指定します。これらは ルート認証情報であり、これらの情報を使用すると、すべての AWS リソースへの完全なアクセスが可能になります。ルートアカウントのユーザーは、一部の CodeCommit 機能を利用できません。さらに、ルートアカウントで Git を使用する唯一の方法は、 をインストールして設定するか git-remote-codecommit (推奨）、 に含まれている AWS 認証情報ヘルパーを設定することです AWS CLI。ルートアカウントのユーザーは、Git 認証情報または SSH パブリック/プライベートキーペアを使用できません。これらの理由により、CodeCommit で作業するときにルートアカウントのユーザーを使用することはお勧めしません。

> **Important**
>
> セキュリティ上の理由から、ルート認証情報は、 AWS アカウントへの完全なアクセス許可を持つ管理者ユーザー (IAM ユーザー) を作成するためにのみ使用することをお勧めします。その後、この管理者ユーザーを使用して、制限されたアクセス許可を持つ他の IAM ユーザーとロールを作成できます。詳細については、[IAM ユーザーガイド]で[IAM ベストプラクティス]および[管理者ユーザーおよびグループの作成]を参照してください。

- IAM Identity Center と IAM Identity Center のユーザー – AWS IAM Identity Center の機能を拡張 AWS Identity and Access Management し、ユーザーの管理と AWS アカウント およびクラウドアプリケーションへのアクセスを一元的に行える場所を提供します。IAM Identity Center は、 AWS を使用するほとんどのユーザーにベストプラクティスとして推奨されていますが、現在のところ、Git 認証情報や SSH キーペアのメカニズムは提供されていません。これらのユーザーは、git-remote-codecommit をインストールして CodeCommit リポジトリのクローンをローカルに作成す

るように設定できますが、すべての統合開発環境 (IDE) が git-remote-codecommit を使用したクローン作成、プッシュ、またはプルをサポートしているわけではありません。

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、一時的な認証情報を使用して にアクセスするために ID プロバイダーとのフェデレーション AWS のサービス を使用することを要求します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリ、または ID ソースを介して提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID がアクセスすると AWS アカウント、ロールを引き受け、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、 AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成するか、独自の ID ソースのユーザーとグループのセットに接続して同期し、すべての AWS アカウント とアプリケーションで使用できます。IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「What is IAM Identity Center?」(IAM Identity Center とは) を参照してください。

- IAM ロール – IAM ユーザーと同様に、IAM ロールは、アカウントに作成できる IAM アイデンティティで、特定のアクセス許可を付与できます。

  IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。で 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。ロールを引き受けるには、 または AWS API オペレーションを AWS CLI 呼び出すか、カスタム URL を使用します。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ための各種方法」を参照してください。

  IAM ロールと一時的な認証情報は、次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) 用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、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。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 ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで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、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では 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におけるロールとリソースベースのポリシーの違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。
- クロスサービスアクセス — 一部の は他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。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と 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アクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除することができます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを作成する」を参照してください。
  - 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、サービスリンク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 EC2 で実行されているアプリケーション – 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 インスタンス内でのアクセスキーの保存に推奨されま

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、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM 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 IAM ロールを使用して許可を付与する」を参照してください。

> **Note**
>
> フェデレーティッドユーザーは、Git 認証情報または SSH パブリック/プライベートキーペアを使用できません。さらに、ユーザー設定はフェデレーティッドユーザーでは使用できません。フェデレーティッドアクセスを使用して接続を設定する方法については、「git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」を参照してください。

アクセスコントロール

有効な認証情報があればリクエストを認証できますが、アクセス許可がなければ CodeCommit リソースの作成やアクセスはできません。たとえば、リポジトリの表示、コードのプッシュ、Git の認証情報の作成および管理などにはアクセス許可が必要です。

以下のセクションでは、CodeCommit のアクセス許可を管理する方法について説明します。最初に概要のセクションを読むことをお勧めします。

- CodeCommit リソースに対するアクセス許可の管理の概要
- CodeCommit でのアイデンティティベースのポリシー (IAM ポリシー) の使用
- CodeCommit アクセス許可リファレンス

CodeCommit リソースに対するアクセス許可の管理の概要

すべての AWS リソースは、Amazon Web Services アカウントによって所有されます。リソースを作成またはアクセスするためのアクセス許可は、アクセス許可ポリシーで管理されます。アカウント管理者は、IAM アイデンティティ (つまり、ユーザー、グループ、ロール) に許可ポリシーをアタッチできます。などの一部の サービスでは AWS Lambda、 リソースへのアクセス許可ポリシーのアタッチもサポートされています。

> **Note**
> アカウント管理者 (または管理者ユーザー) は、管理者権限を持つユーザーです。詳細については、IAM ユーザーガイドの IAM ベストプラクティスを参照してください。

アクセス許可を付与する場合、アクセス許可を取得するユーザー、取得するアクセス許可の対象となるリソース、およびそれらのリソースに対して許可される特定のアクションを決定します。

トピック

- CodeCommit のリソースとオペレーション
- リソース所有権についての理解
- リソースへのアクセスの管理
- CodeCommit でのリソースのスコープ
- ポリシー要素 (リソース、アクション、効果、プリンシパル) の指定
- ポリシーでの条件を指定する

CodeCommit のリソースとオペレーション

CodeCommit では、プライマリリソースがリポジトリとなります。リソースにはそれぞれ、一意の Amazon リソースネーム (ARN) が関連付けられています。ポリシーで Amazon リソースネーム (ARN) を使用して、ポリシーを適用するリソースを識別します。ARN の詳細については、AWSの「Amazon リソースネーム (ARN) およびAmazon Web Services 全般のリファレンスサービスの名前空間」を参照してください。CodeCommit では、現在、サブリソースと呼ばれる他のリソースタイプはサポートされていません。

次の表で、CodeCommit リソースを指定する方法について説明します。

リソースタイプ	ARN 形式
リポジトリ	arn:aws:codecommit:*region*:*account-id* :*repository-name*
すべての CodeCommit リポジトリ	arn:aws:codecommit:*
指定された 内の指定されたアカウント	arn:aws:codecommit:*region*:*account-id* :*

リソースタイプ	ARN 形式
が所有するすべての CodeCommit リポジトリ AWS リージョン	

> **Note**
> ほとんどの AWS サービスは、ARN のARNs。ただし、CodeCommit では、リソースパターンおよびルールで完全一致が必要です。イベントパターンの作成時に正しい ARN 文字を使用して、リソース内の ARN 構文とそれらの文字が一致する必要があります。

たとえば、以下の要領で ARN を使用して、ステートメント内で特定のリポジトリ (*MyDemoRepo*) を指定することができます。

```
"Resource": "arn:aws:codecommit:us-west-2:111111111111:MyDemoRepo"
```

特定のアカウントに属するすべてのリポジトリを指定するには、以下のようにワイルドカード文字 (*) を使用します。

```
"Resource": "arn:aws:codecommit:us-west-2:111111111111:*"
```

すべてのリソースを指定する場合、または特定の API アクションが ARN をサポートしていない場合は、以下の要領で、`Resource` エレメント内でワイルドカード文字 (*) を使用します。

```
"Resource": "*"
```

また、ワイルドカード文字 (*) を使用して、リポジトリ名の一部に一致するすべてのリソースを指定できます。例えば、以下の ARN は、`us-east-2` AWS リージョン内の Amazon Web Services アカウント 111111111111 に登録されている名前 `MyDemo` で始まる CodeCommit リポジトリを指定しています。

```
arn:aws:codecommit:us-east-2:111111111111:MyDemo*
```

CodeCommit リソースで動作する使用可能なオペレーションのリストについては、[CodeCommit アクセス許可リファレンス] を参照してください。

リソース所有権についての理解

Amazon Web Services アカウントは、誰がリソースを作成したかにかかわらず、アカウントで作成されたリソースを所有します。具体的には、リソース所有者は、リソースの作成リクエストを認証するプリンシパルエンティティ (ルートアカウント、IAM ユーザー、または IAM ロール) の Amazon Web Services アカウントです。以下の例では、このしくみを示しています。

- Amazon Web Services アカウントに IAM ユーザーを作成し、そのユーザーに CodeCommit リソースを作成するためのアクセス許可を付与する場合、そのユーザーは CodeCommit リソースを作成できます。ただし、ユーザーが属する Amazon Web Services アカウントは CodeCommit リソースを所有しているとします。
- Amazon Web Services アカウントのルートアカウントの認証情報を使用してルールを作成する場合、 Amazon Web Services アカウントは CodeCommit リソースの所有者です。
- CodeCommit リソースを作成するためのアクセス許可を持つ Amazon Web Services アカウントに IAM ロールを作成する場合は、ロールを引き受けることのできるいずれのユーザーも CodeCommit リソースを作成できます。ロールが属する Amazon Web Services アカウントは CodeCommit リソースを所有しているとします。

リソースへのアクセスの管理

AWS リソースへのアクセスを管理するには、アクセス許可ポリシーを使用します。アクセスポリシーは、誰が何に対するアクセス権を持っているのかを説明します。以下のセクションで、アクセス権限のポリシーを作成するためのオプションについて説明します。

> **Note**
>
> このセクションでは、CodeCommit のコンテキストでの IAM の使用について説明します。これは、IAM サービスに関する詳細情報を取得できません。詳細については、IAM ユーザーガイドの IAM とは を参照してください。IAM ポリシー構文の詳細と説明については、IAM ユーザーガイドの IAM ポリシーの参照 を参照してください。

IAM アイデンティティにアタッチされたアクセス許可ポリシーは、アイデンティティベースのポリシー (IAM ポリシー) と呼ばれます。リソースにアタッチされたアクセス許可ポリシーは、リソースベースのポリシーと呼ばれます。現在、CodeCommit では、アイデンティティベースのポリシー (IAM ポリシー) のみがサポートされています。

トピック

- アイデンティティベースのポリシー (IAM ポリシー)
- リソースベースのポリシー

アイデンティティベースのポリシー (IAM ポリシー)

AWS リソースへのアクセスを管理するには、IAM ID にアクセス許可ポリシーをアタッチします。CodeCommit で、アイデンティティベースのポリシーを使用して、リポジトリへのアクセスを制御します。例えば、次の操作を実行できます。

- アカウントのユーザーまたはグループにアクセス許可ポリシーをアタッチする – CodeCommit コンソールに CodeCommit リソースを表示するアクセス許可をユーザーに付与するには、ユーザーまたはユーザーが属するグループにアイデンティティベースのアクセス許可ポリシーをアタッチします。
- アクセス許可ポリシーをロールにアタッチする (クロスアカウントのアクセス許可を付与) – クロスアカウントアクセスを付与するときなど、委任には、リソースを所有するアカウント (信頼するアカウント) と、リソースへアクセスする必要のあるユーザーを含むアカウント (信頼されたアカウント) の間に信頼を設定することが含まれます。アクセス許可ポリシーは、リソースに対して目的のタスクを実行するために必要なアクセス許可をロールのユーザーに付与します。信頼ポリシーは、どの信頼されたアカウントのユーザーに、ロールを引き受ける権限を与えるかを指定します。詳細については、IAM の用語と概念を参照してください。

  クロスアカウントのアクセス許可を付与するには、アイデンティティベースのアクセス許可ポリシーを IAM ロールにアタッチします。たとえば、アカウント A の管理者は、次のように、別の Amazon Web Services アカウント (アカウント B など) または AWS サービスにクロスアカウントアクセス許可を付与するロールを作成できます。

  1. アカウント A の管理者は、IAM ロールを作成して、アカウント A のリソースに許可を付与するロールに許可ポリシーをアタッチします。
  2. アカウント A の管理者は、アカウント B をそのロールを引き受けるプリンシパルとして識別するロールに、信頼ポリシーをアタッチします。
  3. アカウント B の管理者は、アカウント B のユーザーにロールを引き受けるアクセス許可を委任できます。これにより、アカウント B のユーザーはアカウント A のリソースを作成またはアクセスできます。ロールを引き受けるアクセス許可を AWS サービスに付与する場合、信頼ポリシーのプリンシパルは AWS サービスプリンシパルになることもできます。詳細については、IAM の用語と概念の Delegation (委任) を参照してください。

  IAM を使用した許可委任の詳細については、IAM ユーザーガイドのアクセス管理を参照してください。

次のポリシーの例では、ユーザーが *MyDemoRepo* という名前のリポジトリでブランチを作成することを許可します。

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codecommit:CreateBranch"
      ],
      "Resource" : "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
    }
  ]
}
```

アカウントのユーザーがアクセスできる呼び出しとリソースを制限するには、特定の IAM ポリシーを作成した後、IAM ユーザーにそれらのポリシーをアタッチします。IAM ロールを作成する方法、および CodeCommit の IAM ポリシーステートメントの例を調べる方法の詳細については、カスタマー管理のアイデンティティポリシーの例 を参照してください。

リソースベースのポリシー

Amazon S3 などのいくつかのサービスでは、リソースベースのアクセス許可ポリシーもサポートされています。たとえば、リソースベースのポリシーを S3 バケットにアタッチして、そのバケットに対するアクセス権限を管理できます。CodeCommit はリソースベースのポリシーをサポートしていませんが、タグを使用してリソースを識別し、IAM ポリシーで使用できます。タグベースのポリシーの例については、アイデンティティベースのポリシー (IAM ポリシー) を参照してください。

CodeCommit でのリソースのスコープ

CodeCommit では、CodeCommit のリソースとオペレーション で説明されているように、アイデンティティベースのポリシーとリソースへのアクセス許可の範囲を指定できます。ただし、リソースへの `ListRepositories` アクセス許可の範囲を指定することはできません。代わりに、すべてのリソースを範囲とします (ワイルドカード `*` を使用)。そうしない場合、そのアクションは失敗します。

他のすべての CodeCommit アクセス許可は、リソースにスコープできます。

ポリシー要素 (リソース、アクション、効果、プリンシパル) の指定

リソースへのユーザーのアクセスを許可または拒否するポリシー、または、それらのリソースに対してユーザーが特定のアクションを実行するのを許可または拒否するポリシーを作成できます。CodeCommit は、ユーザーが CodeCommit コンソール、 SDKs、、 AWS CLIまたはそれらの API を直接呼び出すことによって、サービスを操作する方法を定義する一連のパブリック APIs オペレーションを定義します。これらの API 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ために、CodeCommit ではポリシーに一連のアクションを定義できます。

API オペレーションによっては、複数のアクションに対するアクセス許可が必要になる場合があります。リソースおよび API オペレーションに関する詳細については、「CodeCommit のリソースとオペレーション」および「CodeCommit アクセス許可リファレンス」を参照してください。

以下に、ポリシーの基本的な要素を示します。

- リソース – ポリシーを適用するリソースを識別するには、Amazon リソースネーム (ARN) を使用します。詳細については、「CodeCommit のリソースとオペレーション」を参照してください。
- アクション – 許可または拒否するリソースオペレーションを識別するには、アクションキーワードを使用します。例えば、指定された `Effect` に応じて、`codecommit:GetBranch` アクセス許可は、CodeCommit リポジトリのブランチの詳細を取得する `GetBranch` オペレーションをユーザーが実行するのを許可または拒否します。
- 効果 – ユーザーが特定のアクションをリクエストする際の効果の実行について、許可または拒否のいずれかを指定します。リソースへのアクセスを明示的に許可していない場合、アクセスは暗黙的に拒否されます。また、明示的にリソースへのアクセスを拒否すると、別のポリシーによってアクセスが許可されている場合でも、ユーザーはリソースにアクセスできなくなります。
- プリンシパル – CodeCommit がサポートしている唯一のポリシータイプであるアイデンティティベースのポリシー (IAM ポリシー) で、ポリシーがアタッチされているユーザーが黙示的なプリンシパルとなります。

IAM ポリシーの構文の詳細については、IAM ユーザーガイドの IAM ポリシーリファレンスを参照してください。

すべての CodeCommit API アクションとそれらが適用されるリソースの表については、CodeCommit アクセス許可リファレンス を参照してください。

ポリシーでの条件を指定する

アクセス許可を付与するとき、IAM のアクセスポリシー言語を使用してポリシーの適用条件を指定できます。たとえば、特定の日付の後にのみ適用されるポリシーが必要になる場合があります。ポリ

シー言語での条件の指定の詳細については、IAM ユーザーガイドの条件およびポリシーの文法を参照してください。

条件を表すには、あらかじめ定義された条件キーを使用します。CodeCommit に固有の条件キーはありません。ただし、必要に応じて使用できる条件キーは AWS広範囲に存在します。 AWS全体のキーの完全なリストについては、IAM ユーザーガイドの「条件に使用可能なキー」を参照してください。

CodeCommit でのアイデンティティベースのポリシー (IAM ポリシー) の使用

以下のアイデンティティベースのポリシーの例では、アカウント管理者が IAM ID (ユーザー、グループ、およびロール) にアクセス許可ポリシーをアタッチし、CodeCommit リソースに対する操作を実行するためのアクセス許可を付与する方法を示します。

> **Important**
>
> 初めに、CodeCommit リソースへのアクセスを管理するための基本概念と使用可能なオプションについて説明する概要トピックを確認することをお勧めします。詳細については、「CodeCommit リソースに対するアクセス許可の管理の概要」を参照してください。

トピック

- CodeCommit コンソールを使用するために必要なアクセス許可
- コンソールでのリソースの表示
- AWS CodeCommit の マネージドポリシー
- カスタマーマネージドポリシーの例

以下は、アイデンティティベースのアクセス許可ポリシーの例です。

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codecommit:BatchGetRepositories"
      ],
      "Resource" : [
        "arn:aws:codecommit:us-east-2:111111111111:MyDestinationRepo",
```

```
        "arn:aws:codecommit:us-east-2:111111111111:MyDemo*"
      ]
    }
  ]
}
```

このポリシーには、`MyDestinationRepo` リージョンにある `MyDemo` という名前の CodeCommit リポジトリおよび **`us-east-2`** という名前で始まるすべての CodeCommit リポジトリに関する情報を取得することをユーザーに許可するステートメントがあります。

CodeCommit コンソールを使用するために必要なアクセス許可

各 CodeCommit API オペレーションの必要なアクセス許可を確認するには、また、CodeCommit のオペレーションの詳細については、CodeCommit アクセス許可リファレンス を参照してください。

ユーザーが CodeCommit コンソールを使用できるようにするには、管理者は CodeCommit アクションのアクセス許可をユーザーに付与する必要があります。たとえば、AWSCodeCommitPowerUser マネージドポリシーまたはそれに相当するユーザーまたはグループにアタッチできます。

CodeCommit では、アイデンティティベースのポリシーによってユーザーに付与されるアクセス許可に加えて、 AWS Key Management Service （ AWS KMS) アクションのアクセス許可が必要です。IAM ユーザーは、これらのアクションに対して明示的な `Allow` アクセス許可を必要としませんが、以下のアクセス許可を `Deny` に設定するポリシーはアタッチされません。

```
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt",
        "kms:GenerateDataKey",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:DescribeKey"
```

暗号化と CodeCommit の詳細については、AWS KMS および 暗号化 を参照してください。

コンソールでのリソースの表示

CodeCommit コンソールには、サインイン AWS リージョン している で Amazon Web Services アカウントのリポジトリのリストを表示する`ListRepositories`アクセス許可が必要です。このコンソールには、大文字と小文字を区別しない検索をリソースに対して迅速に実行するための [Go to resource (リソースに移動)] 機能も含まれています。この検索は、サインイン AWS リージョン している の Amazon Web Services アカウントで実行されます。次のリソースは、以下のサービス全体で表示されます。

- AWS CodeBuild: ビルドプロジェクト
- AWS CodeCommit: リポジトリ
- AWS CodeDeploy: アプリケーション
- AWS CodePipeline: パイプライン

この検索をすべてのサービスのリソースにわたって実行するには、次のアクセス権限が必要です。

- CodeBuild: `ListProjects`
- CodeCommit: `ListRepositories`
- CodeDeploy: `ListApplications`
- CodePipeline: `ListPipelines`

あるサービスに対するアクセス権限がない場合、そのサービスのリソースに関して結果は返されません。リソースを表示するためのアクセス権限がある場合でも、特定のリソースの表示に対する明示的な `Deny` がある場合にはそれらのリソースは返されません。

AWS CodeCommit の マネージドポリシー

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、自分でポリシーを記述するよりも AWS 管理ポリシーを使用する方が簡単です。チームに必要な権限のみを提供する IAM カスタマーマネージドポリシーを作成するには時間と専門知識が必要です。すぐに開始するには、 AWS マネージドポリシーを使用できます。これらのポリシーは、一般的なユースケースをターゲット範囲に含めており、 AWS アカウントで利用できます。 AWS 管理ポリシーの詳細については、IAM ユーザーガイドの「 AWS 管理ポリシー」を参照してください。

AWS サービスは、 AWS 管理ポリシーを維持および更新します。 AWS 管理ポリシーのアクセス許可は変更できません。サービスでは新しい機能を利用できるようにするために、 AWS マネージドポリシーに権限が追加されることがあります。この種類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、グループおよびロール) に影響を与えます。新しい機能が立ち上げられた場合や、新しいオペレーションが使用可能になった場合に、各サービスが AWS マネージドポリシーを更新する可能性が最も高くなります。サービスは AWS 管理ポリシーからアクセス許可を削除しないため、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ることはありません。

さらに、 は、複数の サービスにまたがる職務機能用の マネージドポリシー AWS をサポートしています。例えば、ReadOnlyAccess AWS 管理ポリシーは、すべての AWS サービスとリソースへの読み取り専用アクセスを提供します。サービスが新機能を起動すると、 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 AWS を追加します。ジョブ機能ポリシーのリストと説明については、IAM ユーザーガイドのジョブ機能のAWS 管理ポリシーを参照してください。

AWS は、 によって作成および管理されるスタンドアロン IAM ポリシーを提供することで、多くの一般的なユースケースに対処します AWS。これらの AWS 管理ポリシーは、一般的なユースケースに必要なアクセス許可を付与します。また、CodeCommit の管理ポリシーは、該当するポリシーが付与されたユーザーの責任の必要に応じて、IAM、Amazon SNS、および Amazon CloudWatch Events などの他のサービスのオペレーションを実行するアクセス許可を提供します。例えば、AWSCodeCommitFullAccess ポリシーは管理レベルのユーザーポリシーで、このポリシーを持つユーザーは、リポジトリの CloudWatch Events ルール (名前にプレフィックス `codecommit` が付いているルール) とリポジトリ関連イベントに関する通知の Amazon SNS トピック (名前にプレフィックス `codecommit` が付いているトピック) を作成および管理でき、また、CodeCommit のリポジトリを管理できます。

アカウントのユーザーにアタッチできる以下の AWS 管理ポリシーは、CodeCommit に固有のものです。

トピック

- AWS マネージドポリシー: AWSCodeCommitFullAccess
- AWS マネージドポリシー: AWSCodeCommitPowerUser
- AWS マネージドポリシー: AWSCodeCommitReadOnly
- CodeCommit の管理ポリシーと通知
- AWS CodeCommit マネージドポリシーと Amazon CodeGuru Reviewer
- CodeCommit による AWS マネージドポリシーの更新

AWS マネージドポリシー: AWSCodeCommitFullAccess

`AWSCodeCommitFullAccess` ポリシーを IAM アイデンティティにアタッチできます。このポリシーにより、CodeCommit へのフルアクセスが付与されます。リポジトリを削除する権限も含め、Amazon Web Services アカウントの CodeCommit リポジトリおよび関連するリソースを完全に制御する権限を付与したい管理レベルのユーザーにのみこのポリシーを適用します。

AWSCodeCommitFullAccess ポリシーには、次のポリシーステートメントが含まれます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
            "Effect": "Allow",
            "Action": [
              "codecommit:*"
            ],
            "Resource": "*"
          },
          {
            "Sid": "CloudWatchEventsCodeCommitRulesAccess",
            "Effect": "Allow",
            "Action": [
              "events:DeleteRule",
              "events:DescribeRule",
              "events:DisableRule",
              "events:EnableRule",
              "events:PutRule",
              "events:PutTargets",
              "events:RemoveTargets",
              "events:ListTargetsByRule"
            ],
            "Resource": "arn:aws:events:*:*:rule/codecommit*"
          },
          {
            "Sid": "SNSTopicAndSubscriptionAccess",
            "Effect": "Allow",
            "Action": [
              "sns:CreateTopic",
              "sns:DeleteTopic",
              "sns:Subscribe",
              "sns:Unsubscribe",
              "sns:SetTopicAttributes"
            ],
            "Resource": "arn:aws:sns:*:*:codecommit*"
          },
          {
            "Sid": "SNSTopicAndSubscriptionReadAccess",
            "Effect": "Allow",
            "Action": [
              "sns:ListTopics",
              "sns:ListSubscriptionsByTopic",
              "sns:GetTopicAttributes"
            ],
            "Resource": "*"
          },
          {
```

```
            "Sid": "LambdaReadOnlyListAccess",
            "Effect": "Allow",
            "Action": [
              "lambda:ListFunctions"
            ],
            "Resource": "*"
          },
          {
            "Sid": "IAMReadOnlyListAccess",
            "Effect": "Allow",
            "Action": [
              "iam:ListUsers"
            ],
            "Resource": "*"
          },
          {
            "Sid": "IAMReadOnlyConsoleAccess",
            "Effect": "Allow",
            "Action": [
              "iam:ListAccessKeys",
              "iam:ListSSHPublicKeys",
              "iam:ListServiceSpecificCredentials"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
          },
          {
            "Sid": "IAMUserSSHKeys",
            "Effect": "Allow",
            "Action": [
              "iam:DeleteSSHPublicKey",
              "iam:GetSSHPublicKey",
              "iam:ListSSHPublicKeys",
              "iam:UpdateSSHPublicKey",
              "iam:UploadSSHPublicKey"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
          },
          {
            "Sid": "IAMSelfManageServiceSpecificCredentials",
            "Effect": "Allow",
            "Action": [
              "iam:CreateServiceSpecificCredential",
              "iam:UpdateServiceSpecificCredential",
              "iam:DeleteServiceSpecificCredential",
```

```
            "iam:ResetServiceSpecificCredential"
          ],
          "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
          "Sid": "CodeStarNotificationsReadWriteAccess",
          "Effect": "Allow",
          "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:DeleteNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
          ],
          "Resource": "*",
          "Condition": {
            "StringLike": {
              "codestar-notifications:NotificationsForResource": "arn:aws:codecommit:*"
            }
          }
        },
        {
          "Sid": "CodeStarNotificationsListAccess",
          "Effect": "Allow",
          "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource",
            "codestar-notifications:ListEventTypes"
          ],
          "Resource": "*"
        },
        {
          "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
          "Effect": "Allow",
          "Action": [
            "sns:CreateTopic",
            "sns:SetTopicAttributes"
          ],
          "Resource": "arn:aws:sns:*:*:codestar-notifications*"
        },
        {
          "Sid": "AmazonCodeGuruReviewerFullAccess",
```

```
          "Effect": "Allow",
          "Action": [
            "codeguru-reviewer:AssociateRepository",
            "codeguru-reviewer:DescribeRepositoryAssociation",
            "codeguru-reviewer:ListRepositoryAssociations",
            "codeguru-reviewer:DisassociateRepository",
            "codeguru-reviewer:DescribeCodeReview",
            "codeguru-reviewer:ListCodeReviews"
          ],
          "Resource": "*"
        },
        {
          "Sid": "AmazonCodeGuruReviewerSLRCreation",
          "Action": "iam:CreateServiceLinkedRole",
          "Effect": "Allow",
          "Resource": "arn:aws:iam::*:role/aws-service-role/codeguru-
reviewer.amazonaws.com/AWSServiceRoleForAmazonCodeGuruReviewer",
          "Condition": {
            "StringLike": {
              "iam:AWSServiceName": "codeguru-reviewer.amazonaws.com"
            }
          }
        },
        {
          "Sid": "CloudWatchEventsManagedRules",
          "Effect": "Allow",
          "Action": [
            "events:PutRule",
            "events:PutTargets",
            "events:DeleteRule",
            "events:RemoveTargets"
          ],
          "Resource": "*",
          "Condition": {
            "StringEquals": {
              "events:ManagedBy": "codeguru-reviewer.amazonaws.com"
            }
          }
        },
        {
          "Sid": "CodeStarNotificationsChatbotAccess",
          "Effect": "Allow",
          "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
```

```
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
          "Resource": "*"
        },
        {
            "Sid": "CodeStarConnectionsReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "codestar-connections:ListConnections",
                "codestar-connections:GetConnection"
            ],
            "Resource": "arn:aws:codestar-connections:*:*:connection/*"
        }
      ]
    }
```

AWS マネージドポリシー: AWSCodeCommitPowerUser

`AWSCodeCommitPowerUser` ポリシーを IAM アイデンティティにアタッチできます。このポリシーでは、ユーザーが CodeCommit およびリポジトリ関連のリソースのすべての機能にアクセスすることを許可します。ただし、CodeCommit リポジトリを削除したり、Amazon CloudWatch Events などの他の AWS サービスでリポジトリ関連のリソースを作成または削除したりすることはできません。ほとんどのユーザーにこのポリシーを適用することをお勧めします。

AWSCodeCommitPowerUser ポリシーには、次のポリシーステートメントが含まれます。

```
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "codecommit:AssociateApprovalRuleTemplateWithRepository",
            "codecommit:BatchAssociateApprovalRuleTemplateWithRepositories",
            "codecommit:BatchDisassociateApprovalRuleTemplateFromRepositories",
            "codecommit:BatchGet*",
            "codecommit:BatchDescribe*",
            "codecommit:Create*",
            "codecommit:DeleteBranch",
            "codecommit:DeleteFile",
            "codecommit:Describe*",
            "codecommit:DisassociateApprovalRuleTemplateFromRepository",
            "codecommit:EvaluatePullRequestApprovalRules",
```

```
            "codecommit:Get*",
            "codecommit:List*",
            "codecommit:Merge*",
            "codecommit:OverridePullRequestApprovalRules",
            "codecommit:Put*",
            "codecommit:Post*",
            "codecommit:TagResource",
            "codecommit:Test*",
            "codecommit:UntagResource",
            "codecommit:Update*",
            "codecommit:GitPull",
            "codecommit:GitPush"
          ],
          "Resource": "*"
        },
        {
          "Sid": "CloudWatchEventsCodeCommitRulesAccess",
          "Effect": "Allow",
          "Action": [
            "events:DeleteRule",
            "events:DescribeRule",
            "events:DisableRule",
            "events:EnableRule",
            "events:PutRule",
            "events:PutTargets",
            "events:RemoveTargets",
            "events:ListTargetsByRule"
          ],
          "Resource": "arn:aws:events:*:*:rule/codecommit*"
        },
        {
          "Sid": "SNSTopicAndSubscriptionAccess",
          "Effect": "Allow",
          "Action": [
            "sns:Subscribe",
            "sns:Unsubscribe"
          ],
          "Resource": "arn:aws:sns:*:*:codecommit*"
        },
        {
          "Sid": "SNSTopicAndSubscriptionReadAccess",
          "Effect": "Allow",
          "Action": [
            "sns:ListTopics",
```

```
            "sns:ListSubscriptionsByTopic",
            "sns:GetTopicAttributes"
          ],
          "Resource": "*"
        },
        {
          "Sid": "LambdaReadOnlyListAccess",
          "Effect": "Allow",
          "Action": [
            "lambda:ListFunctions"
          ],
          "Resource": "*"
        },
        {
          "Sid": "IAMReadOnlyListAccess",
          "Effect": "Allow",
          "Action": [
            "iam:ListUsers"
          ],
          "Resource": "*"
        },
        {
          "Sid": "IAMReadOnlyConsoleAccess",
          "Effect": "Allow",
          "Action": [
            "iam:ListAccessKeys",
            "iam:ListSSHPublicKeys",
            "iam:ListServiceSpecificCredentials"
          ],
          "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
          "Sid": "IAMUserSSHKeys",
          "Effect": "Allow",
          "Action": [
            "iam:DeleteSSHPublicKey",
            "iam:GetSSHPublicKey",
            "iam:ListSSHPublicKeys",
            "iam:UpdateSSHPublicKey",
            "iam:UploadSSHPublicKey"
          ],
          "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
```

```
            "Sid": "IAMSelfManageServiceSpecificCredentials",
            "Effect": "Allow",
            "Action": [
              "iam:CreateServiceSpecificCredential",
              "iam:UpdateServiceSpecificCredential",
              "iam:DeleteServiceSpecificCredential",
              "iam:ResetServiceSpecificCredential"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
          },
          {
            "Sid": "CodeStarNotificationsReadWriteAccess",
            "Effect": "Allow",
            "Action": [
              "codestar-notifications:CreateNotificationRule",
              "codestar-notifications:DescribeNotificationRule",
              "codestar-notifications:UpdateNotificationRule",
              "codestar-notifications:Subscribe",
              "codestar-notifications:Unsubscribe"
            ],
            "Resource": "*",
            "Condition": {
              "StringLike": {
                "codestar-notifications:NotificationsForResource": "arn:aws:codecommit:*"
              }
            }
          },
          {
            "Sid": "CodeStarNotificationsListAccess",
            "Effect": "Allow",
            "Action": [
              "codestar-notifications:ListNotificationRules",
              "codestar-notifications:ListTargets",
              "codestar-notifications:ListTagsforResource",
              "codestar-notifications:ListEventTypes"
            ],
            "Resource": "*"
          },
          {
            "Sid": "AmazonCodeGuruReviewerFullAccess",
            "Effect": "Allow",
            "Action": [
              "codeguru-reviewer:AssociateRepository",
              "codeguru-reviewer:DescribeRepositoryAssociation",
```

```
            "codeguru-reviewer:ListRepositoryAssociations",
            "codeguru-reviewer:DisassociateRepository",
            "codeguru-reviewer:DescribeCodeReview",
            "codeguru-reviewer:ListCodeReviews"
          ],
          "Resource": "*"
        },
        {
          "Sid": "AmazonCodeGuruReviewerSLRCreation",
          "Action": "iam:CreateServiceLinkedRole",
          "Effect": "Allow",
          "Resource": "arn:aws:iam::*:role/aws-service-role/codeguru-
reviewer.amazonaws.com/AWSServiceRoleForAmazonCodeGuruReviewer",
          "Condition": {
            "StringLike": {
              "iam:AWSServiceName": "codeguru-reviewer.amazonaws.com"
            }
          }
        },
        {
          "Sid": "CloudWatchEventsManagedRules",
          "Effect": "Allow",
          "Action": [
            "events:PutRule",
            "events:PutTargets",
            "events:DeleteRule",
            "events:RemoveTargets"
          ],
          "Resource": "*",
          "Condition": {
            "StringEquals": {
              "events:ManagedBy": "codeguru-reviewer.amazonaws.com"
            }
          }
        },
        {
          "Sid": "CodeStarNotificationsChatbotAccess",
          "Effect": "Allow",
          "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
          "Resource": "*"
        },
```

```
        {
            "Sid": "CodeStarConnectionsReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "codestar-connections:ListConnections",
                "codestar-connections:GetConnection"
            ],
            "Resource": "arn:aws:codestar-connections:*:*:connection/*"
        }
      ]
    }
```

AWS マネージドポリシー: AWSCodeCommitReadOnly

`AWSCodeCommitReadOnly` ポリシーを IAM アイデンティティにアタッチできます。このポリシーは、他の AWS サービスの CodeCommit およびリポジトリ関連のリソースへの読み取り専用アクセス、および独自の CodeCommit 関連のリソース (リポジトリへのアクセス時に IAM ユーザーが使用する Git 認証情報や SSH キーなど) を作成および管理するための機能を付与します。リポジトリのコンテンツを読み込む機能 (コンテンツを変更させない) を付与したいユーザーにこのポリシーを適用します。

AWSCodeCommitReadOnly ポリシーには、次のポリシーステートメントが含まれます。

```
    {
       "Version":"2012-10-17",
       "Statement":[
          {
             "Effect":"Allow",
             "Action":[
                "codecommit:BatchGet*",
                "codecommit:BatchDescribe*",
                "codecommit:Describe*",
                "codecommit:EvaluatePullRequestApprovalRules",
                "codecommit:Get*",
                "codecommit:List*",
                "codecommit:GitPull"
             ],
             "Resource":"*"
          },
          {
             "Sid":"CloudWatchEventsCodeCommitRulesReadOnlyAccess",
             "Effect":"Allow",
             "Action":[
```

```
                "events:DescribeRule",
                "events:ListTargetsByRule"
            ],
            "Resource":"arn:aws:events:*:*:rule/codecommit*"
        },
        {
            "Sid":"SNSSubscriptionAccess",
            "Effect":"Allow",
            "Action":[
                "sns:ListTopics",
                "sns:ListSubscriptionsByTopic",
                "sns:GetTopicAttributes"
            ],
            "Resource":"*"
        },
        {
            "Sid":"LambdaReadOnlyListAccess",
            "Effect":"Allow",
            "Action":[
                "lambda:ListFunctions"
            ],
            "Resource":"*"
        },
        {
            "Sid":"IAMReadOnlyListAccess",
            "Effect":"Allow",
            "Action":[
                "iam:ListUsers"
            ],
            "Resource":"*"
        },
        {
            "Sid":"IAMReadOnlyConsoleAccess",
            "Effect":"Allow",
            "Action":[
                "iam:ListAccessKeys",
                "iam:ListSSHPublicKeys",
                "iam:ListServiceSpecificCredentials",
                "iam:GetSSHPublicKey"
            ],
            "Resource":"arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid":"CodeStarNotificationsReadOnlyAccess",
```

```
            "Effect":"Allow",
            "Action":[
               "codestar-notifications:DescribeNotificationRule"
            ],
            "Resource":"*",
            "Condition":{
               "StringLike":{
                  "codestar-
notifications:NotificationsForResource":"arn:aws:codecommit:*"
               }
            }
         },
         {
            "Sid":"CodeStarNotificationsListAccess",
            "Effect":"Allow",
            "Action":[
               "codestar-notifications:ListNotificationRules",
               "codestar-notifications:ListEventTypes",
               "codestar-notifications:ListTargets"
            ],
            "Resource":"*"
         },
         {
            "Sid": "AmazonCodeGuruReviewerReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
               "codeguru-reviewer:DescribeRepositoryAssociation",
               "codeguru-reviewer:ListRepositoryAssociations",
               "codeguru-reviewer:DescribeCodeReview",
               "codeguru-reviewer:ListCodeReviews"
            ],
            "Resource": "*"
         },
         {
           "Sid": "CodeStarConnectionsReadOnlyAccess",
           "Effect": "Allow",
           "Action": [
               "codestar-connections:ListConnections",
               "codestar-connections:GetConnection"
           ],
           "Resource": "arn:aws:codestar-connections:*:*:connection/*"
        }
    ]
```

```
}
```

CodeCommit の管理ポリシーと通知

AWS CodeCommit は、リポジトリへの重要な変更をユーザーに通知できる通知をサポートしています。CodeCommit の管理ポリシーには、通知機能のポリシーステートメントが含まれます。詳細については、通知とはを参照してください。

フルアクセスマネージドポリシーの通知に関連するアクセス許可

`AWSCodeCommitFullAccess` マネージドポリシーには、通知へのフルアクセスを許可する次のステートメントが含まれています。この管理ポリシーが適用されたユーザーは、通知の Amazon SNS トピックの作成と管理、トピックへのユーザーのサブスクライブとサブスクライブ解除、通知ルールのターゲットとして選択するトピックの一覧表示、Slack 用に設定されたチャットアプリケーションクライアントの Amazon Q Developer の一覧表示を行うこともできます。

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:DeleteNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codecommit:*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource,"
            "codestar-notifications:ListEventTypes"
        ],
        "Resource": "*"
```

```
    },
    {
        "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
        "Effect": "Allow",
        "Action": [
            "sns:CreateTopic",
            "sns:SetTopicAttributes"
        ],
        "Resource": "arn:aws:sns:*:*:codestar-notifications*"
    },
    {
        "Sid": "CodeStarNotificationsChatbotAccess",
        "Effect": "Allow",
        "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
       "Resource": "*"
    }
```

読み取り専用マネージドポリシーの通知に関連するアクセス許可

AWSCodeCommitReadOnlyAccess 管理ポリシーには、通知への読み取り専用アクセスを許可する以下のステートメントが含まれています。この管理ポリシーが適用されたユーザーは、リソースの通知を表示することはできますが、リソースの作成や管理、リソースへのサブスクライブを行うことはできません。

```
   {
        "Sid": "CodeStarNotificationsPowerUserAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:DescribeNotificationRule"
        ],
        "Resource": "*",
        "Condition" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codecommit:*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
```

```
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListEventTypes",
            "codestar-notifications:ListTargets"
        ],
        "Resource": "*"
    }
```

その他の管理ポリシーの通知に関連するアクセス許可

AWSCodeCommitPowerUser 管理ポリシーには、ユーザーが通知を作成、編集、サブスクライブできるようにする次のステートメントが含まれています。ユーザーは通知ルールを削除したり、リソースのタグを管理したりすることはできません。

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:DeleteNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codecommit*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource",
            "codestar-notifications:ListEventTypes"
        ],
        "Resource": "*"
    },
    {
        "Sid": "SNSTopicListAccess",
```

```
        "Effect": "Allow",
        "Action": [
            "sns:ListTopics"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsChatbotAccess",
        "Effect": "Allow",
        "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
       "Resource": "*"
    }
```

IAM と通知の詳細については、「AWS CodeStar Notifications の Identity and Access Management」を参照してください。

AWS CodeCommit マネージドポリシーと Amazon CodeGuru Reviewer

CodeCommit は、プログラム分析と機械学習を使用して一般的な問題を検出し、Java または Python コードにおける修正点を提案する自動化されたコードレビューサービスである Amazon CodeGuru Reviewer をサポートしています。CodeCommit の管理ポリシーには、CodeGuru Reviewer 機能のポリシーステートメントが含まれます。詳細については、Amazon CodeGuru Reviewer とは を参照してください。

AWSCodeCommitFullAccess の CodeGuru Reviewer に関連するアクセス許可

`AWSCodeCommitFullAccess` 管理ポリシーには、CodeGuru Reviewer が CodeCommit リポジトリに関連付けられ、および関連付けが解除されることを許可する、次のステートメントが含まれています。このマネージドポリシーが適用されたユーザーは、CodeCommit リポジトリと CodeGuru Reviewer との関連付けステータスを表示したり、プルリクエストのレビュージョブのステータスを表示したりすることもできます。

```
    {
      "Sid": "AmazonCodeGuruReviewerFullAccess",
      "Effect": "Allow",
      "Action": [
        "codeguru-reviewer:AssociateRepository",
        "codeguru-reviewer:DescribeRepositoryAssociation",
        "codeguru-reviewer:ListRepositoryAssociations",
```

```
        "codeguru-reviewer:DisassociateRepository",
        "codeguru-reviewer:DescribeCodeReview",
        "codeguru-reviewer:ListCodeReviews"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AmazonCodeGuruReviewerSLRCreation",
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/codeguru-
reviewer.amazonaws.com/AWSServiceRoleForAmazonCodeGuruReviewer",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "codeguru-reviewer.amazonaws.com"
        }
      }
    },
    {
      "Sid": "CloudWatchEventsManagedRules",
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events:DeleteRule",
        "events:RemoveTargets"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "events:ManagedBy": "codeguru-reviewer.amazonaws.com"
        }
      }
    }
```

AWSCodeCommitPowerUser の CodeGuru Reviewer に関連するアクセス許可

`AWSCodeCommitPowerUser` マネージドポリシーには、リポジトリと CodeGuru Reviewer の関連付けや関連付けの解除、関連付けステータスの表示、プルリクエストのレビュージョブのステータスの表示をユーザーに許可するための以下ステートメントが含まれています。

```
    {
      "Sid": "AmazonCodeGuruReviewerFullAccess",
```

```
      "Effect": "Allow",
      "Action": [
        "codeguru-reviewer:AssociateRepository",
        "codeguru-reviewer:DescribeRepositoryAssociation",
        "codeguru-reviewer:ListRepositoryAssociations",
        "codeguru-reviewer:DisassociateRepository",
        "codeguru-reviewer:DescribeCodeReview",
        "codeguru-reviewer:ListCodeReviews"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AmazonCodeGuruReviewerSLRCreation",
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/codeguru-
reviewer.amazonaws.com/AWSServiceRoleForAmazonCodeGuruReviewer",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "codeguru-reviewer.amazonaws.com"
        }
      }
    },
    {
      "Sid": "CloudWatchEventsManagedRules",
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events:DeleteRule",
        "events:RemoveTargets"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "events:ManagedBy": "codeguru-reviewer.amazonaws.com"
        }
      }
    }
```

AWSCodeCommitReadOnly の CodeGuru Reviewer に関連するアクセス許可

`AWSCodeCommitReadOnlyAccess` マネージドポリシーには、CodeGuru Reviewer の関連付けステータスやプルリクエストのレビュージョブのステータスを表示するための読み取り専用アクセスを許可する以下のステートメントが含まれています。このマネージドポリシーが適用されたユーザーは、リポジトリを関連付けたり関連付け解除できません。

```
    {
     "Sid": "AmazonCodeGuruReviewerReadOnlyAccess",
     "Effect": "Allow",
     "Action": [
           "codeguru-reviewer:DescribeRepositoryAssociation",
           "codeguru-reviewer:ListRepositoryAssociations",
           "codeguru-reviewer:DescribeCodeReview",
           "codeguru-reviewer:ListCodeReviews"
     ],
     "Resource": "*"
   }
```

Amazon CodeGuru Reviewer のサービスにリンクされたロール

リポジトリを CodeGuru Reviewer に関連付けると、CodeGuru Reviewer がプルリクエストの Java または Python コードの問題を検出し、修正点を提案できるように、サービスにリンクされたロールが作成されます。サービスにリンクされたロールの名前は、AWSServiceRoleForAmazonCodeGuruReviewer です。詳細については、Amazon CodeGuru Reviewer でのサービスにリンクされたロールの使用を参照してください。

詳細については、IAM ユーザーガイドの AWS 管理ポリシーを参照してください。

CodeCommit による AWS マネージドポリシーの更新

このサービスがこれらの変更の追跡を開始してからの CodeCommit の AWS マネージドポリシーの更新に関する詳細を表示します。このページの変更に関する自動通知については、AWS CodeCommit ユーザーガイドのドキュメント履歴 の RSS フィードを購読してください。

変更	説明	日付
AWS マネージドポリシー: AWSCodeCommitFullAccess と AWS マネージドポリシー:	CodeCommit は、チャットアプリケーションで Amazon Q Developer を使用する追加の	2023 年 5 月 16 日

変更	説明	日付
AWSCodeCommitPowerUser – 既存のポリシーに対する更新	通知タイプをサポートするために、これらのポリシーにアクセス許可を追加しました。 AWSCodeCommitPowerUser ポリシーと AWSCodeCommitFullAccess ポリシーが変更され、アクセス許可 `chatbot:ListMicrosoftTeamsChannelConfigurations` が追加されました。	
AWS マネージドポリシー: AWSCodeCommitReadOnly – 既存ポリシーへの更新	CodeCommit がポリシーから重複したアクセス許可を削除しました。 AWSCodeCommitReadOnly が変更され、重複したアクセス許可 `"iam:ListAccessKeys"` が削除されました。	2021 年 8 月 18 日
CodeCommit が変更の追跡を開始しました	CodeCommit は、 AWS 管理ポリシーの変更の追跡を開始しました。	2021 年 8 月 18 日

カスタマーマネージドポリシーの例

独自のカスタム IAM ポリシーを作成して、CodeCommit アクションとリソースのための権限を許可することもできます。これらのカスタムポリシーは、それらのアクセス許可が必要な IAM ユーザーまたはグループにアタッチできます。CodeCommit と他の AWS のサービスを統合するための独自のカスタム IAM ポリシーを作成することもできます。

トピック

- カスタマー管理のアイデンティティポリシーの例

- カスタマー管理の統合ポリシーの例

カスタマー管理のアイデンティティポリシーの例

次の IAM ポリシーの例では、さまざまな CodeCommit アクションのアクセス許可を付与します。それらを使用して、IAM ユーザーおよびロールに対して CodeCommit アクセスを制限します。これらのポリシーは、CodeCommit コンソール、API、 AWS SDKs、または を使用してアクションを実行する機能を制御します AWS CLI。

> **Note**
>
> すべての例で、米国西部 (オレゴン) リージョン (us-west-2) を使用し、架空のアカウント ID を使用しています。

例

- 例 1: ユーザーが 1 つの で CodeCommit オペレーションを実行できるようにする AWS リージョン
- 例 2: 1 つのリポジトリで Git を使用することをユーザーに許可する
- 例 3: 指定した IP アドレス範囲から接続するユーザーにリポジトリへのアクセスを許可する
- 例 4: ブランチに対するアクションを許可または拒否する
- 例 5: タグを使用してリポジトリに対するアクションを許可または拒否する

例 1: ユーザーが 1 つの で CodeCommit オペレーションを実行できるようにする AWS リージョン

次のアクセス許可ポリシーでは、ワイルドカード文字 (`"codecommit:*"`) を使用して、他の AWS リージョンからではなく、us-east-2 リージョンですべての CodeCommit アクションを実行できるようにしています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codecommit:*",
            "Resource": "arn:aws:codecommit:us-east-2:111111111111:*",
            "Condition": {
```

```
                "StringEquals": {
                    "aws:RequestedRegion": "us-east-2"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "codecommit:ListRepositories",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": "us-east-2"
                }
            }
        }
    ]
}
```

例 2: 1 つのリポジトリで Git を使用することをユーザーに許可する

CodeCommit では、`GitPull` IAM ポリシーのアクセス許可は、git fetch、git clone など、CodeCommit からデータを取得する Git クライアントコマンドに適用されます。同様に、`GitPush` IAM ポリシーアクセス許可は、CodeCommit にデータを送信する Git クライアントコマンドに適用されます。例えば、`GitPush` IAM ポリシーアクセス許可が `Allow` に設定されている場合、ユーザーは Git プロトコルを使用してブランチの削除をプッシュできます。そのプッシュは、その IAM ユーザーの `DeleteBranch` オペレーションに適用されているどのアクセス許可の影響も受けません。アクセス`DeleteBranch`許可は、コンソール、、 SDKs、API AWS CLIで実行されるアクションに適用されますが、Git プロトコルには適用されません。

以下の例では、指定したユーザーが `MyDemoRepo` という名前の CodeCommit リポジトリに対してプルおよびプッシュできるようにしています。

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codecommit:GitPull",
        "codecommit:GitPush"
      ],
      "Resource" : "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
```

```
    }
  ]
}
```

例 3: 指定した IP アドレス範囲から接続するユーザーにリポジトリへのアクセスを許可する

IP アドレスが特定の IP アドレス範囲にある場合にのみ、ユーザーが CodeCommit リポジトリに接続することを許可するポリシーを作成できます。これには、有効なアプローチが 2 つあります。ユーザーの IP アドレスが特定のブロック内にない場合に CodeCommit オペレーションを拒否する `Deny` ポリシーを作成するか、または、ユーザーの IP アドレスが特定のブロック内にある場合に CodeCommit オペレーションを許可する `Allow` ポリシーを作成することができます。

特定の IP 範囲にないすべてのユーザーのアクセスを拒否する `Deny` ポリシーを作成することができます。例えば、AWSCodeCommitPowerUser 管理ポリシーとカスタマー管理ポリシーをリポジトリへのアクセスを必要とするすべてのユーザーにアタッチできます。次のポリシーの例では、IP アドレスが指定された IP アドレスのブロック 203.0.113.0/16 にないユーザーの CodeCommit アクセス許可をすべて拒否します。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Deny",
         "Action": [
            "codecommit:*"
         ],
         "Resource": "*",
         "Condition": {
            "NotIpAddress": {
               "aws:SourceIp": [
                  "203.0.113.0/16"
               ]
            }
         }
      }
   ]
}
```

次のポリシーの例では、MyDemoRepo という名前の CodeCommit リポジトリに、指定されたユーザーが、IP アドレスが指定された 203.0.113.0/16 のアドレスブロック内である場合に、同等のアクセス許可である AWSCodeCommitPowerUser 管理ポリシーでアクセスするのを許可します。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "codecommit:BatchGetRepositories",
            "codecommit:CreateBranch",
            "codecommit:CreateRepository",
            "codecommit:Get*",
            "codecommit:GitPull",
            "codecommit:GitPush",
            "codecommit:List*",
            "codecommit:Put*",
            "codecommit:Post*",
            "codecommit:Merge*",
            "codecommit:TagResource",
            "codecommit:Test*",
            "codecommit:UntagResource",
            "codecommit:Update*"
         ],
         "Resource": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
         "Condition": {
            "IpAddress": {
               "aws:SourceIp": [
                  "203.0.113.0/16"
               ]
            }
         }
      }
   ]
}
```

例 4: ブランチに対するアクションを許可または拒否する

1 つ以上のブランチで、指定したアクションに対するユーザーのアクセス許可を拒否するポリシーを作成できます。あるいは、リポジトリの他のブランチにはないような、1 つ以上のブランチに対してアクションを許可するポリシーを作成することもできます。これらのポリシーは、適切な管理 (事前定義済み) ポリシーとともに使用できます。詳細については、「のブランチへのプッシュとマージを制限する AWS CodeCommit」を参照してください。

例えば、*MyDemoRepo* という名前のリポジトリで、そのブランチを削除することを含め、ユーザーが main という名前のブランチに変更を加えることを拒否する Deny ポリシーを作成できます。このポリシーは、AWSCodeCommitPowerUser 管理ポリシーとともに使用できます。これら 2 つのポリシーが適用されたユーザーは、ブランチの作成と削除、プルリクエストの作成、および AWSCodeCommitPowerUser で許可されているその他すべてのアクションを実行できますが、main というブランチに変更をプッシュしたり、CodeCommit コンソールの main ブランチのファイルを追加または編集したり、ブランチまたはプルリクエストを main ブランチにマージしたりすることはできません。Deny に GitPush が適用されているので、ユーザーがローカルリポジトリからプッシュしたときに最初の Null 呼び出しの有効性を分析できるように、ポリシーに GitPush ステートメントを含める必要があります。

> **Tip**
>
> Amazon Web Services アカウントのすべてのリポジトリで main という名前のすべてのブランチに適用するポリシーを作成する場合、Resource では、リポジトリ ARN の代わりにアスタリスク (*) を指定します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "codecommit:GitPush",
                "codecommit:DeleteBranch",
                "codecommit:PutFile",
                "codecommit:Merge*"
            ],
            "Resource": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
            "Condition": {
                "StringEqualsIfExists": {
                    "codecommit:References": [
                        "refs/heads/main"
                    ]
                },
                "Null": {
                    "codecommit:References": "false"
                }
            }
```

```
        }
    ]
}
```

次のポリシー例では、ユーザーは Amazon Web Services アカウントのすべてのリポジトリ内の main というブランチに変更を加えることができます。他のブランチへの変更は許可されません。このポリシーを AWSCodeCommitReadOnly 管理ポリシーとともに使用して、main ブランチにあるリポジトリへの自動プッシュを許可することができます。効果は `Allow` であるため、この例のポリシーは AWSCodeCommitPowerUser などの管理ポリシーでは機能しません。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codecommit:GitPush",
                "codecommit:Merge*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "codecommit:References": [
                        "refs/heads/main"
                    ]
                }
            }
        }
    ]
}
```

例 5: タグを使用してリポジトリに対するアクションを許可または拒否する

これらのリポジトリに関連付けられた AWS タグに基づいてリポジトリに対するアクションを許可または拒否するポリシーを作成し、IAM ユーザーを管理するために設定した IAM グループにそれらのポリシーを適用できます。例えば、 AWS タグキー Status と Secret のキー値を持つリポジトリですべての CodeCommit アクションを拒否するポリシーを作成し、そのポリシーを一般的な開発者 (###) 用に作成した IAM グループに適用できます。次に、上記のタグ付けされたリポジトリ上で作業するデベロッパーが一般的な######グループのメンバーではなく、代わりに制限されたポリシーが

適用されていない別の IAM グループに属していること (SecretDevelopers) を確認する必要があります。

次の例では、Status キーと Secret のキー値でタグ付けされたリポジトリ上のすべての CodeCommit アクションを拒否しています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codecommit:Associate*",
        "codecommit:Batch*",
        "codecommit:CancelUploadArchive",
        "codecommit:CreateBranch",
        "codecommit:CreateCommit",
        "codecommit:CreatePullRequest*",
        "codecommit:CreateRepository",
        "codecommit:CreateUnreferencedMergeCommit",
        "codecommit:DeleteBranch",
        "codecommit:DeleteCommentContent",
        "codecommit:DeleteFile",
        "codecommit:DeletePullRequest*",
        "codecommit:DeleteRepository",
        "codecommit:Describe*",
        "codecommit:DisassociateApprovalRuleTemplateFromRepository",
        "codecommit:EvaluatePullRequestApprovalRules",
        "codecommit:GetBlob",
        "codecommit:GetBranch",
        "codecommit:GetComment*",
        "codecommit:GetCommit",
        "codecommit:GetDifferences*",
        "codecommit:GetFile",
        "codecommit:GetFolder",
        "codecommit:GetMerge*",
        "codecommit:GetObjectIdentifier",
        "codecommit:GetPullRequest*",
        "codecommit:GetReferences",
        "codecommit:GetRepository*",
        "codecommit:GetTree",
        "codecommit:GetUploadArchiveStatus",
        "codecommit:Git*",
```

```
        "codecommit:ListAssociatedApprovalRuleTemplatesForRepository",
        "codecommit:ListBranches",
        "codecommit:ListPullRequests",
        "codecommit:ListTagsForResource",
        "codecommit:Merge*",
        "codecommit:OverridePullRequestApprovalRules",
        "codecommit:Post*",
        "codecommit:Put*",
        "codecommit:TagResource",
        "codecommit:TestRepositoryTriggers",
        "codecommit:UntagResource",
        "codecommit:UpdateComment",
        "codecommit:UpdateDefaultBranch",
        "codecommit:UpdatePullRequest*",
        "codecommit:UpdateRepository*",
        "codecommit:UploadArchive"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Status": "Secret"
        }
      }
    }
  ]
}
```

すべてのリポジトリではなく、特定のリポジトリをリソースとして指定することで、この戦略をさらに精密にすることができます。また、特定のタグによってタグ付けされていないすべてのリポジトリ上の CodeCommit アクションを許可するポリシーを作成することもできます。例えば、次のポリシーは特定のタグによってタグ付けされていないリポジトリでCodeCommit アクションのみを許可することを除き、CodeCommit アクションに対する AWSCodeCommitPowerUser 許可に相当する許可を付与します。

> **Note**
>
> このポリシーの例には、CodeCommit のアクションのみが含まれます。AWSCodeCommitPowerUser 管理ポリシーに含まれる他の AWS サービスのアクションは含まれません。詳細については、「」を参照してください[AWS マネージドポリシー: AWSCodeCommitPowerUser](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:Associate*",
        "codecommit:Batch*",
        "codecommit:CancelUploadArchive",
        "codecommit:CreateBranch",
        "codecommit:CreateCommit",
        "codecommit:CreatePullRequest*",
        "codecommit:CreateRepository",
        "codecommit:CreateUnreferencedMergeCommit",
        "codecommit:DeleteBranch",
        "codecommit:DeleteCommentContent",
        "codecommit:DeleteFile",
        "codecommit:DeletePullRequest*",
        "codecommit:Describe*",
        "codecommit:DisassociateApprovalRuleTemplateFromRepository",
        "codecommit:EvaluatePullRequestApprovalRules",
        "codecommit:GetBlob",
        "codecommit:GetBranch",
        "codecommit:GetComment*",
        "codecommit:GetCommit",
        "codecommit:GetDifferences*",
        "codecommit:GetFile",
        "codecommit:GetFolder",
        "codecommit:GetMerge*",
        "codecommit:GetObjectIdentifier",
        "codecommit:GetPullRequest*",
        "codecommit:GetReferences",
        "codecommit:GetRepository*",
        "codecommit:GetTree",
        "codecommit:GetUploadArchiveStatus",
        "codecommit:Git*",
        "codecommit:ListAssociatedApprovalRuleTemplatesForRepository",
        "codecommit:ListBranches",
        "codecommit:ListPullRequests",
        "codecommit:ListTagsForResource",
        "codecommit:Merge*",
        "codecommit:OverridePullRequestApprovalRules",
        "codecommit:Post*",
```

```
        "codecommit:Put*",
        "codecommit:TagResource",
        "codecommit:TestRepositoryTriggers",
        "codecommit:UntagResource",
        "codecommit:UpdateComment",
        "codecommit:UpdateDefaultBranch",
        "codecommit:UpdatePullRequest*",
        "codecommit:UpdateRepository*",
        "codecommit:UploadArchive"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceTag/Status": "Secret",
          "aws:ResourceTag/Team": "Saanvi"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:CreateApprovalRuleTemplate",
        "codecommit:GetApprovalRuleTemplate",
        "codecommit:ListApprovalRuleTemplates",
        "codecommit:ListRepositories",
        "codecommit:ListRepositoriesForApprovalRuleTemplate",
        "codecommit:UpdateApprovalRuleTemplateContent",
        "codecommit:UpdateApprovalRuleTemplateDescription",
        "codecommit:UpdateApprovalRuleTemplateName"
      ],
      "Resource": "*"
    }
  ]
}
```

カスタマー管理の統合ポリシーの例

このセクションでは、CodeCommit と他の AWS サービスの統合のためのアクセス許可を付与するカスタマー管理ユーザーポリシーの例を示します。CodeCommit リポジトリへのクロスアカウントアクセスを許可するポリシーの特定の例については、ロールを使用して AWS CodeCommit リポジトリへのクロスアカウントアクセスを設定する を参照してください。

> **Note**
> すべての例では、 AWS リージョン が必要な場合に米国西部 (オレゴン) リージョン (us-west-2) を使用し、架空のアカウント IDs が含まれています。

例

- 例 1: Amazon SNS トピックへのクロスアカウントアクセスを有効にするポリシーを作成する
- 例 2: Amazon CloudWatch Events がトピックに CodeCommit イベントを発行できるようにする Amazon Simple Notification Service (Amazon SNS) トピックポリシーを作成する
- 例 3: CodeCommit トリガーと AWS Lambda 統合するためのポリシーを作成する

例 1: Amazon SNS トピックへのクロスアカウントアクセスを有効にするポリシーを作成する

コードプッシュや他のイベントが Amazon Simple Notification Service (Amazon SNS) からの通知の送信などのアクションをトリガーするように CodeCommit リポジトリを設定できます。CodeCommit リポジトリの作成に使用したものと同じアカウントで Amazon SNS トピックを作成する場合は、追加の IAM ポリシーまたはアクセス許可を設定する必要はありません。トピックを作成し、リポジトリのトリガーを作成できます。詳細については、「Amazon SNS トピック用のトリガーを作成する」を参照してください。

ただし、トリガーを設定して、別の Amazon Web Services アカウントで Amazon SNS トピックを使用する場合は、まず CodeCommit がそのトピックにパブリッシュできるようにするポリシーを使用してトピックを設定する必要があります。他のアカウントから、Amazon SNS コンソールを開き、リストからトピックを選択し、[Other topic actions] (他のトピックアクション) で [Edit topic policy] (トピックポリシーの編集) を選択します。[Advanced] (アドバンスト) タブで、トピックのポリシーを変更して、CodeCommit がそのトピックに発行できるようにします。例えば、ポリシーがデフォルトポリシーの場合、ポリシーに以下のように変更して、######で示している項目がリポジトリ、Amazon SNS トピック、アカウントの値に一致するようにします。

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
```

```
        "AWS": "*"
      },
      "Action": [
        "sns:Subscribe",
        "sns:ListSubscriptionsByTopic",
        "sns:DeleteTopic",
        "sns:GetTopicAttributes",
        "sns:Publish",
        "sns:RemovePermission",
        "sns:AddPermission",        "sns:SetTopicAttributes"
      ],
      "Resource": "arn:aws:sns:us-east-2:111111111111:NotMySNSTopic",
      "Condition": {
        "StringEquals": {
          "AWS:SourceOwner": "111111111111"
        }
      }
     },
     {
      "Sid": "CodeCommit-Policy_ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "codecommit.amazonaws.com"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:111111111111:NotMySNSTopic",
      "Condition": {
        "StringEquals": {
          "AWS:SourceArn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
          "AWS:SourceAccount": "111111111111"
        }
      }
    }
  ]
}
```

例 2: Amazon CloudWatch Events がトピックに CodeCommit イベントを発行できるようにする Amazon Simple Notification Service (Amazon SNS) トピックポリシーを作成する

CodeCommit イベントなど、イベントが発生したときに Amazon SNS トピックに発行するように CloudWatch Events を設定できます。これを行うには、トピックのポリシーを作成するか、次のようなトピックの既存のポリシーを変更して、CloudWatch Events に Amazon SNS トピックへのイベントの発行を許可するアクセス許可を付与する必要があります。

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic",
      "Condition": {
        "StringEquals": {
          "AWS:SourceOwner": "123456789012"
        }
      }
    },
    {
      "Sid": "Allow_Publish_Events",
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

CodeCommit および CloudWatch Events の詳細については、サポートされている各サービスからの CloudWatch Events イベントの例を参照してください。IAM とポリシー言語の詳細については、IAM JSON ポリシー言語の文法を参照してください。

例 3: CodeCommit トリガーと AWS Lambda 統合するためのポリシーを作成する

コードがプッシュされるように、または他のイベントでアクション (AWS Lambdaでの関数の呼び出しなど) がトリガーされるように、CodeCommit リポジトリを設定できます。詳細については、「Lambda 関数のトリガーを作成する」を参照してください。この情報は CloudWatch Events ではなく、トリガーに固有のものです。

トリガーで Lambda 関数を直接実行し (Amazon SNS トピックを使用して Lambda 関数を呼び出す代わりに)、Lambda コンソールでトリガーを設定しない場合は、関数のリソースベースのポリシーに次のようなステートメントを含める必要があります。

```
{
  "Statement":{
     "StatementId":"Id-1",
     "Action":"lambda:InvokeFunction",
     "Principal":"codecommit.amazonaws.com",
     "SourceArn":"arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
     "SourceAccount":"111111111111"
  }
}
```

Lambda 関数を呼び出す CodeCommit トリガーを手動で設定する場合は、Lambda AddPermission コマンドを使用して、関数を呼び出すアクセス許可を CodeCommit に付与する必要もあります。例については、「CodeCommit でLambda 関数を実行できるようにするには」の「既存の Lambda 関数のトリガーを作成する」セクションを参照してください。

Lambda 関数のリソースポリシーの詳細については、AWS Lambda デベロッパーガイドの AddPermission およびプル/プッシュイベントモデルを参照してください。

CodeCommit アクセス許可リファレンス

次の表には、各 CodeCommit API オペレーション、アクセス許可を付与できる対応するアクション、およびアクセス許可を付与するためのリソース ARN の形式が一覧で示されています。CodeCommit API は、その API により許可されたアクションの範囲に基づいてテーブルに分類されています。アクセスコントロール をセットアップし、IAM アイデンティティ (アイデンティティベースのポリシー) にアタッチできるアクセス許可ポリシーを作成する際、参照してください。

アクセス許可ポリシーを作成するときに、ポリシーの `Action` フィールドでアクションを指定します。ポリシーの `Resource` フィールドで、ワイルドカード文字 (*) を使用して、または使用せずに、ARN としてリソース値を指定します。

CodeCommit ポリシーの条件を表すには、 AWS全体の条件キーを使用します。 AWS全体のキーの完全なリストについては、IAM ユーザーガイドの「使用可能なキー」を参照してください。IAM ポリシーの CodeCommit のアクション、リソース、条件キーの詳細については、「AWS CodeCommit のアクション、リソース、条件キー」を参照してください。

> **Note**
>
> アクションを指定するには、API オペレーション名 (例えば、`codecommit:` や `codecommit:GetRepository`) の前に `codecommit:CreateRepository` プレフィックスを使用します。

ワイルドカードの使用

複数のアクションまたはリソースを指定するには、ARN でワイルドカード文字 (*) を使用します。例えば、`codecommit:*` はすべての CodeCommit アクションを指定し、`codecommit:Get*` は `Get` という単語で始まるすべての CodeCommit アクションを指定します。次の例では、`MyDemo` で始まる名前のすべてのレポジトリへのアクセスを許可します。

```
arn:aws:codecommit:us-west-2:111111111111:MyDemo*
```

次のテーブルに示されている *`repository-name`* リソースでのみワイルドカードを使用できます。ワイルドカードを *`region`* または *`account-id`* リソースで使用することはできません。ワイルドカードの詳細については、IAM ユーザーガイドの IAM ID を参照してください。

トピック

- Git クライアントのコマンドに必要なアクセス許可
- ブランチに対するアクションのアクセス許可
- マージに対するアクションのアクセス許可
- プルリクエストに対するアクションのアクセス許可
- 承認ルールテンプレートに対するアクションのアクセス許可
- 個別のファイルに対するアクションのアクセス許可
- コメントに対するアクションのアクセス許可
- コミットされたコードに対するアクションのアクセス許可
- リポジトリに対するアクションのアクセス許可
- タグに対するアクションのアクセス許可
- トリガーに対するアクションのアクセス許可
- CodePipeline 統合でのアクションのアクセス許可

Git クライアントのコマンドに必要なアクセス許可

CodeCommit では、`GitPull` IAM ポリシーのアクセス許可は、git fetch、git clone など、CodeCommit からデータを取得する Git クライアントコマンドに適用されます。同様に、`GitPush` IAM ポリシーアクセス許可は、CodeCommit にデータを送信する Git クライアントコマンドに適用されます。例えば、`GitPush` IAM ポリシーアクセス許可が `Allow` に設定されている場合、ユーザーは Git プロトコルを使用してブランチの削除をプッシュできます。そのプッシュは、その IAM ユーザーの `DeleteBranch` オペレーションに適用されているどのアクセス許可の影響も受けません。アクセス`DeleteBranch`許可は、コンソール、、 SDKs、API AWS CLIで実行されるアクションに適用されますが、Git プロトコルには適用されません。

`GitPull` と `GitPush` は IAM ポリシーアクセス許可です。API アクションではありません。

Git クライアントのコマンドへのアクションに CodeCommit で必要なアクセス許可

GitPull

アクション: `codecommit:GitPull`

CodeCommit リポジトリからローカル repo に情報をプルするのに必要です。これは単なる IAM ポリシーのアクセス許可であり、API アクションではありません。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`

GitPush

アクション: `codecommit:Git Push`

ローカルリポジトリから CodeCommit リポジトリに情報をプッシュするのに必要です。これは単なる IAM ポリシーのアクセス許可であり、API アクションではありません。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`

ブランチに対するアクションのアクセス許可

次のアクセス許可では、CodeCommit リポジトリのブランチに対するアクションを許可または拒否します。これらのアクセス許可は、CodeCommit コンソールで実行されるアクションと CodeCommit API を使用して実行されるアクション、および、 AWS CLIを使用して実行されるコマンドについてのみ関係します。Git プロトコルを使用して実行される同様のアクションには関係しません。たとえば、git show-branch -r コマンドは、Git プロトコルを使用してリポジトリとコミットの

リモートブランチのリストを表示します。CodeCommit の `ListBranches` オペレーションのアクセス許可による影響はありません。

ブランチのポリシーの詳細については、「のブランチへのプッシュとマージを制限する AWS CodeCommit および カスタマーマネージドポリシーの例」を参照してください。

CodeCommit API オペレーションおよびブランチのアクションで必要なアクセス許可

CreateBranch

アクション: `codecommit:CreateBranch`

CodeCommit リポジトリにブランチを作成するために必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

DeleteBranch

アクション: `codecommit:DeleteBranch`

CodeCommit リポジトリからブランチを削除するために必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

GetBranch

アクション: `codecommit:GetBranch`

CodeCommit リポジトリ内のブランチに関する詳細を取得するのに必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

ListBranches

アクション: `codecommit:ListBranches`

CodeCommit リポジトリ内のブランチのリストを取得するために必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

MergeBranchesByFastForward

アクション: `codecommit:MergeBranchesByFastForward`

CodeCommit リポジトリで早送りマージ戦略を使用した 2 つのブランチのマージに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

MergeBranchesBySquash

アクション: `codecommit:ListBranches`

CodeCommit リポジトリでスカッシュマージ戦略を使用した 2 つのブランチのマージに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

MergeBranchesByThreeWay

アクション: `codecommit:ListBranches`

CodeCommit リポジトリで 3 方向マージ戦略を使用した 2 つのブランチのマージに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

UpdateDefaultBranch

アクション: `codecommit:UpdateDefaultBranch`

CodeCommit リポジトリのデフォルトブランチを変更するのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

マージに対するアクションのアクセス許可

次のアクセス許可では、CodeCommit リポジトリのマージに対するアクションを許可または拒否します。これらのアクセス許可は、CodeCommit コンソールと CodeCommit API で実行されるアクション、および、 AWS CLIを使用して実行されるコマンドに関係します。Git プロトコルを使用して実行される同様のアクションには関係しません。ブランチでのアクセス権限の詳細については、「ブランチに対するアクションのアクセス許可」を参照してください。プルリクエストでのアクセス権限については、「プルリクエストに対するアクションのアクセス許可」を参照してください。

マージコマンドのアクションにおける CodeCommit API オペレーションと必要なアクセス許可

BatchDescribeMergeConflicts

アクション: `codecommit:BatchDescribeMergeConflicts`

CodeCommit リポジトリでのコミット間のマージにおける競合に関する情報を返すために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

CreateUnreferencedMergeCommit

アクション: `codecommit:CreateUnreferencedMergeCommit`

比較して潜在的な競合を識別するために、CodeCommit リポジトリでの 2 つのブランチあるいはコミット間の非参照のコミットを作成するために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

DescribeMergeConflicts

アクション: `codecommit:DescribeMergeConflicts`

CodeCommit リポジトリでの潜在的なマージに対して、ファールのベース、送信元、送信先のバージョン間におけるマージ競合に関する情報を返すために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetMergeCommit

アクション: `codecommit:GetMergeCommit`

CodeCommit リポジトリで送信元と送信先のコミット間のマージに関する情報を返すために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetMergeOptions

アクション: `codecommit:GetMergeOptions`

CodeCommit リポジトリでの 2 つのブランチまたはコミット識別子間で使用できるマージオプションに関する情報を返すために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

プルリクエストに対するアクションのアクセス許可

次のアクセス許可では、CodeCommit リポジトリにおけるプルリクエストに対するアクションを許可または拒否します。これらのアクセス許可は、CodeCommit コンソールと CodeCommit API で実行されるアクション、および、 AWS CLIを使用して実行されるコマンドに関係します。Git プロトコ

ルを使用して実行される同様のアクションには関係しません。コメントに関連するアクセス権限については、「コメントに対するアクションのアクセス許可」を参照してください。

CodeCommit API オペレーションおよびプルリクエストのアクションで必要なアクセス許可

BatchGetPullRequests

アクション: `codecommit:BatchGetPullRequests`

CodeCommit リポジトリで 1 つ以上のプルリクエストに関する情報を返すのに必要です。これは IAM ポリシーのアクセス許可であり、呼び出すことのできる API アクションではありません。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

CreatePullRequest

アクション: `codecommit:CreatePullRequest`

CodeCommit リポジトリにプルリクエストを作成するために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

CreatePullRequestApprovalRule

アクション: `codecommit:CreatePullRequestApprovalRule`

CodeCommit リポジトリのプルリクエストの承認ルールを作成するために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

DeletePullRequestApprovalRule

アクション: `codecommit:DeletePullRequestApprovalRule`

CodeCommit リポジトリのプルリクエストの承認ルールを削除するために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

DescribePullRequestEvents

アクション: `codecommit:DescribePullRequestEvents`

CodeCommit リポジトリの 1 つ以上のプルリクエストイベントに関する情報を返すのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

EvaluatePullRequestApprovalRules

アクション: `codecommit:EvaluatePullRequestApprovalRules`

プルリクエストが、CodeCommit リポジトリの関連する承認ルールで指定されたすべての条件を満たしているかどうかを評価するために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetCommentsForPullRequest

アクション: `codecommit:GetCommentsForPullRequest`

プルリクエストでのコメントを返すのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetCommitsFromMergeBase

アクション: `codecommit:GetCommitsFromMergeBase`

潜在的なマージのコンテキストにおけるコミット間の違いに関する情報を返す場合は必須。これは IAM ポリシーのアクセス許可であり、呼び出すことのできる API アクションではありません。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetMergeConflicts

アクション: `codecommit:GetMergeConflicts`

プルリクエストのソースブランチと送信先ブランチとのマージの競合に関する情報を返す場合は必須。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetPullRequest

アクション: `codecommit:GetPullRequest`

CodeCommit リポジトリのプルリクエストに関する情報を返すのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetPullRequestApprovalStates

アクション: `codecommit:GetPullRequestApprovalStates`

指定したプルリクエストの承認状態に関する情報を返すために必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

GetPullRequestOverrideState

アクション: `codecommit:GetPullRequestOverrideState`

プルリクエストの承認ルールが破棄されている (上書きされている) かどうかと、上書きされている場合に、プルリクエストのルールと要件を上書きしたユーザーまたはアイデンティティの Amazon リソースネーム (ARN) に関する情報を返すために必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

ListPullRequests

アクション: `codecommit:ListPullRequests`

リポジトリでプルリクエストをリスト表示するのに必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

MergePullRequestByFastForward

アクション: `codecommit:MergePullRequestByFastForward`

プルリクエストをクローズして、早送りマージ戦略を使用してプルリクエストの送信元ブランチを送信先ブランチに統合するのに必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

MergePullRequestBySquash

アクション: `codecommit:MergePullRequestBySquash`

プルリクエストをクローズして、スカッシュマージ戦略を使用してプルリクエストの送信元ブランチを送信先ブランチに統合するのに必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

MergePullRequestByThreeWay

アクション: `codecommit:MergePullRequestByThreeWay`

プルリクエストをクローズして、3 方向マージ戦略を使用してプルリクエストの送信元ブランチを送信先ブランチに統合するのに必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

OverridePullRequestApprovalRules

アクション: `codecommit:OverridePullRequestApprovalRules`

CodeCommit リポジトリのプルリクエストのすべての承認ルール要件を確保するために必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

PostCommentForPullRequest

アクション: `codecommit:PostCommentForPullRequest`

CodeCommit リポジトリでプルリクエストにコメントを投稿するのに必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

UpdatePullRequestApprovalRuleContent

アクション: `codecommit:UpdatePullRequestApprovalRuleContent`

CodeCommit リポジトリのプルリクエストの承認ルールの構造を変更するために必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

UpdatePullRequestApprovalState

アクション: `codecommit:UpdatePullRequestApprovalState`

CodeCommit リポジトリのプルリクエストの承認の状態を更新するために必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

UpdatePullRequestDescription

アクション: `codecommit:UpdatePullRequestDescription`

CodeCommit リポジトリのプルリクエストの説明を変更するのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

UpdatePullRequestStatus

アクション: `codecommit:UpdatePullRequestStatus`

CodeCommit リポジトリのプルリクエストのステータスを変更するのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

UpdatePullRequestTitle

アクション: `codecommit:UpdatePullRequestTitle`

CodeCommit リポジトリのプルリクエストのタイトルを変更するのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

承認ルールテンプレートに対するアクションのアクセス許可

次のアクセス許可は、CodeCommit リポジトリの承認ルールテンプレートに対するアクションを許可または拒否します。これらのアクセス許可は、CodeCommit コンソールと CodeCommit API で実行されるアクション、および、 AWS CLIを使用して実行されるコマンドについてのみ関係します。Git プロトコルを使用して実行される同様のアクションには関係しません。プルリクエストでのアクセス権限については、「プルリクエストに対するアクションのアクセス許可」を参照してください。

承認ルールテンプレートに対するアクションの CodeCommit API オペレーションと必要なアクセス許可

AssociateApprovalRuleTemplateWithRepository

アクション: `codecommit:AssociateApprovalRuleTemplateWithRepository`

Amazon Web Services アカウントの指定されたリポジトリにテンプレートを関連付けるために必要です。関連付けられると、指定したリポジトリで作成されるすべてのプルリクエストのテンプレート条件と一致する承認ルールが自動的に作成されます。

リソース: *

BatchAssociateApprovalRuleTemplateWithRepositories

アクション: `codecommit:BatchAssociateApprovalRuleTemplateWithRepositories`

Amazon Web Services アカウントの 1 つ以上の指定されたリポジトリにテンプレートを関連付けるために必要です。

リソース: *

BatchDisassociateApprovalRuleTemplateFromRepositories

アクション: `codecommit:BatchDisassociateApprovalRuleTemplateFromRepositories`

Amazon Web Services アカウントの 1 つ以上の指定したリポジトリからテンプレートの関連付けを解除するために必要です。

リソース: *

CreateApprovalRuleTemplate

アクション: `codecommit:CreateApprovalRuleTemplate`

承認ルールのテンプレートを作成するために必要です。このテンプレートは、Amazon Web Services アカウントの 1 つ以上のリポジトリに関連付けることができます。

リソース: *

DeleteApprovalRuleTemplate

アクション: `codecommit:DeleteApprovalRuleTemplate`

AWS アカウントから承認ルールテンプレートを削除するのに必要です。

リソース: *

DisassociateApprovalRuleTemplateFromRepository

アクション: `codecommit:DisassociateApprovalRuleTemplateFromRepository`

Amazon Web Services アカウントのリポジトリから指定したテンプレートとの関連付けを解除するために必要です。テンプレートですでに作成されたプルリクエストの承認ルールは削除されません。

リソース: *

GetApprovalRuleTemplate

アクション: `codecommit:GetApprovalRuleTemplate`

Amazon Web Services アカウントの承認ルールテンプレートに関する情報を返す場合に必要です。

リソース: *

ListApprovalRuleTemplates

アクション: `codecommit:ListApprovalRuleTemplates`

Amazon Web Services アカウントに承認ルールテンプレートを一覧表示するために必要です。

リソース: *

ListAssociatedApprovalRuleTemplatesForRepository

アクション: `codecommit:ListAssociatedApprovalRuleTemplatesForRepository`

Amazon Web Services アカウントの指定したリポジトリに関連付けられているすべての承認ルールテンプレートを一覧表示するために必要です。

リソース: *

ListRepositoriesForApprovalRuleTemplate

アクション: `codecommit:ListRepositoriesForApprovalRuleTemplate`

Amazon Web Services アカウントの指定された承認ルールテンプレートに関連付けられているすべてのリポジトリを一覧表示するために必要です。

リソース: *

UpdateApprovalRuleTemplateContent

アクション: `codecommit:UpdateApprovalRuleTemplateContent`

Amazon Web Services アカウントの承認ルールテンプレートの内容を更新するために必要です。

リソース: *

UpdateApprovalRuleTemplateDescription

アクション: `codecommit:UpdateApprovalRuleTemplateDescription`

Amazon Web Services アカウントの承認ルールテンプレートの説明を更新するのに必要です。

リソース: *

UpdateApprovalRuleTemplateName

アクション: `codecommit:UpdateApprovalRuleTemplateName`

AWS アカウントの承認ルールテンプレートの名前を更新するのに必要です。

リソース: *

個別のファイルに対するアクションのアクセス許可

次のアクセス許可では、CodeCommit リポジトリの個別のファイルに対するアクションを許可または拒否します。これらのアクセス許可は、CodeCommit コンソールと CodeCommit API で実行されるアクション、および、 AWS CLIを使用して実行されるコマンドについてのみ関係します。Git プロトコルを使用して実行される同様のアクションには関係しません。例えば、`git push` コマンドは、Git プロトコルを使用して新規の変更されたファイルを CodeCommit リポジトリにプッシュします。CodeCommit の `PutFile` オペレーションのアクセス許可による影響はありません。

CodeCommit API オペレーションおよび個別ファイルにおけるアクションで必要なアクセス許可

DeleteFile

アクション: `codecommit:DeleteFile`

CodeCommit コンソールから、CodeCommit リポジトリ内の指定されたブランチから指定されたファイルを削除するために必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

GetBlob

アクション: `codecommit:GetBlob`

CodeCommit コンソールの CodeCommit リポジトリにある個々のファイルのエンコードされた内容を表示するために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetFile

アクション: `codecommit:GetFile`

CodeCommit コンソールから CodeCommit リポジトリ内の指定されたファイルとそのメタデータのエンコードされた内容を表示するために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetFolder

アクション: `codecommit:GetFolder`

CodeCommit コンソールから CodeCommit リポジトリ内の指定されたフォルダの内容を表示するために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

PutFile

アクション: `codecommit:PutFile`

CodeCommit コンソール、CodeCommit API、または AWS CLIから CodeCommit リポジトリに新しいファイルまたは変更されたファイルを追加するために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

コメントに対するアクションのアクセス許可

次のアクセス許可では、CodeCommit リポジトリのコメントに対するアクションを許可または拒否します。これらのアクセス許可は、CodeCommit コンソールと CodeCommit API で実行されるアクション、および、 AWS CLIを使用して実行されるコマンドに関係します。プルリクエストのコメントに関連するアクセス権限については、「プルリクエストに対するアクションのアクセス許可」を参照してください。

CodeCommit API オペレーションおよびリポジトリのアクションで必要なアクセス許可

DeleteCommentContent

アクション: `codecommit:DeleteCommentContent`

リポジトリ内の変更、ファイル、コミットに対してなされたコメントの内容を削除するのに必要です。コメントは削除できませんが、ユーザーにこのアクセス許可がある場合はコメントの内容を削除できます。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetComment

アクション: `codecommit:GetComment`

CodeCommit リポジトリ内の変更、ファイル、コミットに対してなされたコメントに関する情報を返すのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetCommentReactions

アクション: `codecommit:GetCommentReactions`

CodeCommit リポジトリ内の変更、ファイル、コミットに対するコメントへの絵文字リアクションの情報を返すために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetCommentsForComparedCommit

アクション: `codecommit:GetCommentsForComparedCommit`

CodeCommit リポジトリ内の 2 つのコミット間の比較に対するコメントに関する情報を返すのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

PostCommentForComparedCommit

アクション: `codecommit:PostCommentForComparedCommit`

CodeCommit リポジトリ内の 2 つのコミット間の比較に対するにコメントするのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

PostCommentReply

アクション: `codecommit:PostCommentReply`

コミット間の比較に対する、または CodeCommit リポジトリ内のプルリクエストのコメントに返信を作成するのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

PutcommentReaction

アクション: `codecommit:PutCommentReaction`

CodeCommit リポジトリ内のコミットやプルリクエストに対するコメントに絵文字を使用して返信するために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

UpdateComment

アクション: `codecommit:UpdateComment`

コミット間の比較に対する、またはプルリクエストのコメントを編集するのに必要です。コメントの作成者だけがコメントを編集できます。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

コミットされたコードに対するアクションのアクセス許可

次のアクセス許可では、CodeCommit リポジトリにコミットされたコードに対するアクションを許可または拒否します。これらのアクセス許可は、CodeCommit コンソールと CodeCommit API で実行されるアクション、および、 AWS CLIを使用して実行されるコマンドに関係します。Git プロトコルを使用して実行される同様のアクションには関係しません。たとえば、git commit コマンドは、Git プロトコルを使用してリポジトリのブランチに対するコミットを作成します。CodeCommit の `CreateCommit` オペレーションのアクセス許可による影響はありません。

これらのアクセス許可を明示的に拒否すると、CodeCommit コンソールで予期しない結果が発生する可能性があります。たとえば、`GetTree` を `Deny` に設定すると、ユーザーがコンソールでリポジトリの内容を操作することは禁止されますが、ユーザーがリポジトリ内のファイルの内容を表示することはブロックされません (ユーザーが E メールでファイルへのリンクを受け取った場合など)。`GetBlob` を `Deny` に設定すると、ユーザーがファイルの内容を表示することは禁止されますが、ユーザーがリポジトリの構造を参照することはブロックされません。`GetCommit` を `Deny` に設定すると、ユーザーがコミットに関する詳細を取得することを禁止します。`GetObjectIdentifier` を `Deny` に設定すると、コードの参照機能のほとんどがブロックさ

れます。これらの 3 つのアクションのすべてをポリシーで Deny に設定すると、そのポリシーが適用されるユーザーは、CodeCommit コンソールでコードを参照できません。

CodeCommit API オペレーションおよびコミットされたコードのアクションで必要なアクセス許可

BatchGetCommits

アクション: `codecommit:BatchGetCommits`

CodeCommit リポジトリ内の 1 つまたは複数のコミットに関する情報を返すために必要です。これは IAM ポリシーのアクセス許可であり、呼び出すことのできる API アクションではありません。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

CreateCommit

アクション: `codecommit:CreateCommit`

コミットを作成するために必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

GetCommit

アクション: `codecommit:GetCommit`

コミットに関する情報を返すのに必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

GetCommitHistory

アクション: `codecommit:GetCommitHistory`

リポジトリに対するコミットの履歴に関する情報を返すのに必要です。これは IAM ポリシーのアクセス許可であり、呼び出すことのできる API アクションではありません。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

GetDifferences

アクション: `codecommit:GetDifferences`

コミット指定子 (ブランチ、タグ、HEAD、コミット ID、または他の完全修飾参照) の違いに関する情報を返すのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetObjectIdentifier

アクション: `codecommit:GetObjectIdentifier`

BLOB、ツリー、コミットをそれらの識別子に解決すのに必要です。これは IAM ポリシーのアクセス許可であり、呼び出すことのできる API アクションではありません。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetReferences

アクション: `codecommit:GetReferences`

ブランチやタグなどのすべての参照を返すのに必要です。これは IAM ポリシーのアクセス許可であり、呼び出すことのできる API アクションではありません。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetTree

アクション: `codecommit:GetTree`

CodeCommit コンソールから、CodeCommit リポジトリ内の指定されたツリーの内容を表示するために必要です。これは IAM ポリシーのアクセス許可であり、呼び出すことのできる API アクションではありません。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

リポジトリに対するアクションのアクセス許可

次のアクセス許可では、CodeCommit リポジトリに対するアクションを許可または拒否します。これらのアクセス許可は、CodeCommit コンソールと CodeCommit API で実行されるアクション、および、 AWS CLIを使用して実行されるコマンドに関係します。Git プロトコルを使用して実行される同様のアクションには関係しません。

CodeCommit API オペレーションおよびリポジトリのアクションで必要なアクセス許可

BatchGetRepositories

アクション: `codecommit:BatchGetRepositories`

Amazon Web Services アカウント内の複数の CodeCommit リポジトリに関する情報を取得するのに必要です。`Resource` では、ユーザーが情報の取得を許可 (または拒否) されるすべての CodeCommit リポジトリの名前を指定する必要があります。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

CreateRepository

アクション: `codecommit:CreateRepository`

CodeCommit リポジトリを作成するために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

DeleteRepository

アクション: `codecommit:DeleteRepository`

CodeCommit リポジトリを削除するために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetRepository

アクション: `codecommit:GetRepository`

1 つの CodeCommit リポジトリに関する情報を取得するのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

ListRepositories

アクション: `codecommit:ListRepositories`

Amazon Web Services アカウントに登録されている複数の CodeCommit リポジトリの名前とシステム ID のリストを取得するのに必要です。このアクションで `Resource` に許可される値は、すべてのリポジトリ (*) のみです。

リソース: `*`

UpdateRepositoryDescription

アクション: `codecommit:UpdateRepositoryDescription`

CodeCommit リポジトリの説明を変更するために必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

UpdateRepositoryName

アクション: `codecommit:UpdateRepositoryName`

CodeCommit リポジトリの名前を変更するために必要です。`Resource` では、変更が許可されている CodeCommit リポジトリと新しいリポジトリ名の両方を指定する必要があります。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

タグに対するアクションのアクセス許可

次のアクセス許可は、CodeCommit リソースの AWS タグに対するアクションを許可または拒否します。

CodeCommit API オペレーションおよびタグのアクションに必要なアクセス許可

ListTagsForResource

アクション: `codecommit:ListTagsForResource`

CodeCommit のリソースで設定された AWS タグに関する情報を返すために必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

TagResource

アクション: `codecommit:TagResource`

リポジトリの AWS タグを追加または編集するために必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

UntagResource

アクション: `codecommit:UntagResource`

CodeCommit のリソースから AWS タグを削除するために必要です。

リソース: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

トリガーに対するアクションのアクセス許可

次のアクセス許可では、CodeCommit リポジトリのトリガーに対するアクションを許可または拒否します。

CodeCommit API オペレーションおよびトリガーのアクションに必要なアクセス許可

GetRepositoryTriggers

アクション: `codecommit:GetRepositoryTriggers`

リポジトリのために設定されたトリガーに関する情報を返すのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

PutRepositoryTriggers

アクション: `codecommit:PutRepositoryTriggers`

リポジトリのトリガーを作成、編集、または削除するのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

TestRepositoryTriggers

アクション: `codecommit:TestRepositoryTriggers`

トリガー用に設定されたトピックまたは関数にデータを送ることで、リポジトリトリガーの機能をテストするのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

CodePipeline 統合でのアクションのアクセス許可

CodePipeline がパイプラインのソースアクションで CodeCommit リポジトリを使用するには、次のテーブルに示されているすべてのアクセス許可を CodePipeline のサービスロールに付与する必要があります。これらのアクセス許可がサービスロールで設定されていないか、**Deny** に設定されている場合、リポジトリに変更が加えられてもパイプラインは自動的に実行されないため、変更を手動でリリースすることはできません。

CodeCommit API オペレーションと CodePipeline 統合でのアクションに必要なアクセス許可

GetBranch

アクション: `codecommit:GetBranch`

CodeCommit リポジトリ内のブランチに関する詳細を取得するのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetCommit

アクション: `codecommit:GetCommit`

コミットに関する情報を返すのに必要です。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

UploadArchive

アクション: `codecommit:UploadArchive`

CodePipeline のサービスロールに対し、変更をパイプラインにアップロードすることを許可するために必要です。これは IAM ポリシーのアクセス許可であり、呼び出すことのできる API アクションではありません。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

GetUploadArchiveStatus

アクション: `codecommit:GetUploadArchiveStatus`

アーカイブのアップロードのステータス (進行中か、完了したか、キャンセル済みか、エラーが発生したかどうか) を判断するのに必要です。これは IAM ポリシーのアクセス許可であり、呼び出すことのできる API アクションではありません。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

CancelUploadArchive

アクション: `codecommit:CancelUploadArchive`

パイプラインへのアーカイブのアップロードをキャンセルするのに必要です。これは IAM ポリシーのアクセス許可であり、呼び出すことのできる API アクションではありません。

リソース: `arn:aws:codecommit:region:account-id:repository-name`

が IAM と AWS CodeCommit 連携する方法

IAM を使用して CodeCommit へのアクセスを管理する前に、CodeCommit で使用できる IAM 機能について理解しておく必要があります。CodeCommit およびその他の AWS のサービスが IAM と連携

する方法の概要を把握するには、IAM ユーザーガイドのAWS 「IAM と連携する のサービス」を参照してください。

トピック

- 条件キー
- 例

条件キー

管理者は JSON AWS ポリシーを使用して、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。

`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` 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を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` 要素を指定する場合、または 1 つの `Condition` 要素に複数のキーを指定する場合、 AWS では `AND` 論理演算子を使用してそれらを評価します。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 を評価します。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。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、「IAM ユーザーガイド」の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートしています。すべての AWS グローバル条件キーを確認するには、IAM 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。

CodeCommit は独自の条件キーを定義し、一部のグローバル条件キーの使用をサポートしています。すべての AWS グローバル条件キーを確認するには、IAM 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。

CodeCommit アクションの中には、`codecommit:References` 条件キーをサポートするものもあります。このキーを使用するポリシーの例については、「例 4: ブランチに対するアクションを許可または拒否する」を参照してください。

CodeCommit 条件キーのリストを確認するには、IAM ユーザーガイドの AWS CodeCommitの条件キーを参照してください。条件キーを使用できるアクションとリソースについては、「 で定義されるアクション AWS CodeCommit」を参照してください。

例

CodeCommit アイデンティティベースのポリシーの例については、AWS CodeCommit アイデンティティベースのポリシーの例 を参照してください。

CodeCommit のリソースベースのポリシー

CodeCommit では、リソースベースのポリシーはサポートされていません。

CodeCommit タグに基づく認可

CodeCommit リソースにタグをアタッチしたり、リクエスト内のタグを CodeCommit に渡したりできます。タグに基づいてアクセスを制御するには、`codecommit:ResourceTag/`*`key-name`*、`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提供します。CodeCommit リソースのタグ付けの詳細については、例 5: タグを使用してリポジトリに対するアクションを許可または拒否する を参照してください。タグ付け戦略の詳細については、AWS 「リソースのタグ付け」を参照してください。

CodeCommit では、セッションタグに基づくポリシーもサポートしています。詳細については、「セッションタグ」を参照してください。

CodeCommit で ID 情報を提供するためのタグの使用

CodeCommit では、セッションタグの使用がサポートされています。セッションタグは、IAM ロールを引き受けるとき、一時的な認証情報を使用するとき、または AWS Security Token Service (AWS STS) でユーザーをフェデレートするときに渡すキーと値のペアの属性です。タグを IAM ユーザーに関連付けることもできます。これらのタグで提供される情報を使用して、変更を加えた人やイベントを発生させたユーザーを簡単に特定できます。CodeCommit では、次のキー名を持つタグの値が CodeCommit イベントに含まれます。

キー名	値
`displayName`	表示してユーザーに関連付ける人間が判読可能な名前 (たとえば、Mary Major や Saanvi Sarkar) 。

キー名	値
emailAddress	ユーザーに対して表示し、関連付ける E メールアドレス (例: mary_major@example.com または saanvi_sarkar@example.com) 。

この情報が提供された場合、CodeCommit は Amazon EventBridge および Amazon CloudWatch Events に送信されるイベントにその情報を含めます。詳細については、「Amazon EventBridge および Amazon CloudWatch Events の CodeCommit イベントのモニタリング」を参照してください。

セッションタグを使用するには、ロールに、sts:TagSession アクセス許可を Allow に設定したポリシーが含まれている必要があります。フェデレーティッドアクセスを使用している場合は、設定の一部として表示名と E メールタグ情報を構成できます。たとえば、Azure Active Directory を使用している場合は、次のクレーム情報を提供できます。

クレーム名	値
https://aws.amazon.com/SAML/Attributes/PrincipalTag:displayNam e	user.displayname
https://aws.amazon.com/SAML/Attributes/PrincipalTag:emailAddress	user.mail

を使用して AWS CLI 、 displayNameと のセッションタグemailAddressを渡すことができますAssumeRole。たとえば、*Developer* という名前のロールを引き受け、*Mary Major* という名前を関連付けたいユーザーは、次のような assume-role コマンドを使用できます。

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Developer \
--role-session-name Mary-Major \
--tags Key=displayName,Value="Mary Major"
 Key=emailAddress,Value="mary_major@example.com" \
--external-id Example987
```

詳細については、「AssumeRole」を参照してください。

AssumeRoleWithSAML オペレーションを使用して、displayName タグと emailAddress タグを含む一時的な認証情報のセットを返すことができます。これらのタグは、CodeCommit リポジトリにアクセスするときに使用できます。これには、会社またはグループがサードパーティーの SAML ソリューションを既に統合していることが必要です AWS。その場合は、SAML 属性をセッションタグとして渡すことができます。たとえば、*Saanvi Sarkar* という名前のユーザーの表示名と E メールアドレスの ID 属性をセッションタグとして渡す場合は、次のようにします。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:displayName">
  <AttributeValue>Saanvi Sarkar</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:emailAddress">
  <AttributeValue>saanvi_sarkar@example.com</AttributeValue>
</Attribute>
```

詳細については、「AssumeRoleWithSAML を使用したセッションタグの受け渡し」を参照してください。

AssumeRoleWithIdentity オペレーションを使用して、displayName タグと emailAddress タグを含む一時的な認証情報のセットを返すことができます。これらのタグは、CodeCommit リポジトリにアクセスするときに使用できます。OpenID Connect (OIDC) からセッションタグを渡すには、JSON ウェブトークン (JWT) にセッションタグを含める必要があります。例: *Li Juan* という名前のユーザーの AssumeRoleWithWebIdentity セッションタグ displayName とセッションタグが含まれる呼び出し emailAddress に使用されるデコードされた JWP トークン。

```
{
    "sub": "lijuan",
    "aud": "ac_oic_client",
    "jti": "ZYUCeREXAMPLE",
    "iss": "https://xyz.com",
    "iat": 1566583294,
    "exp": 1566583354,
    "auth_time": 1566583292,
    "https://aws.amazon.com/tags": {
        "principal_tags": {
            "displayName": ["Li Juan"],
            "emailAddress": ["li_juan@example.com"],
        },
        "transitive_tag_keys": [
            "displayName",
            "emailAddress"
        ]
```

```
    }
}
```

詳細については、「AssumeRoleWithWebIdentity を使用したセッションタグの受け渡し」を参照してください。

GetFederationToken オペレーションを使用して、displayName タグと emailAddress タグを含む一時的な認証情報のセットを返すことができます。これらのタグは、CodeCommit リポジトリにアクセスするときに使用できます。たとえば、 を使用して、 タグdisplayNameとemailAddress タグを含むフェデレーショントークン AWS CLI を取得するには、次のようにします。

```
aws sts get-federation-token \
--name my-federated-user \
--tags key=displayName,value="Nikhil Jayashankar"
 key=emailAddress,value=nikhil_jayashankar@example.com
```

詳細については、「GetFederationToken を使用したセッションタグの受け渡し」を参照してください。

CodeCommit IAM ロール

IAM ロールは、特定のアクセス許可を持つ、Amazon Web Services アカウント内のエンティティです。

CodeCommit での一時的な認証情報の使用

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する、IAM 役割を引き受ける、またはクロスアカウント役割を引き受けることができます。一時的なセキュリティ認証情報を取得するには、AssumeRole や GetFederationToken などの AWS STS API オペレーションを呼び出します。

CodeCommit では、一時認証情報の使用をサポートしています。詳細については、「認証情報をローテーションして AWS CodeCommit リポジトリに接続する」を参照してください。

サービスにリンクされた役割

サービスにリンクされたロールを使用すると、 AWS サービスは他の 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。サービスリンクロールは IAM アカウント内に表示され、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。

CodeCommit 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役割を引き受けることが許可されます。この役割により、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。サービス役割は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 管理者はこの役割の権限を変更できます。ただし、これを行うことにより、サービスの機能が損なわれる場合があります。

CodeCommit はサービスロールを使用しません。

AWS CodeCommit 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、IAM ユーザーおよびロールには、CodeCommit リソースを作成または変更するアクセス許可はありません。また、 AWS Management Console、 AWS CLI、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、ユーザーとロールに必要な、指定されたリソースで特定の API オペレーションを実行する権限をユーザーとロールに付与する 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 IAM 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。

ポリシーの例については、以下のトピックを参照してください。

- 例 1: ユーザーが 1 つの で CodeCommit オペレーションを実行できるようにする AWS リージョン
- 例 2: 1 つのリポジトリで Git を使用することをユーザーに許可する
- 例 3: 指定した IP アドレス範囲から接続するユーザーにリポジトリへのアクセスを許可する
- 例 4: ブランチに対するアクションを許可または拒否する
- 例 5: タグを使用してリポジトリに対するアクションを許可または拒否する
- ロールを使用して AWS CodeCommit リポジトリへのクロスアカウントアクセスを設定する

JSON ポリシードキュメントのこれらの例を使用して、IAM アイデンティティ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「JSON タブでのポリシーの作成」を参照してください。

トピック

- ポリシーに関するベストプラクティス
- CodeCommit コンソールの使用

- ユーザーが自分の許可を表示できるようにする
- タグに基づく CodeCommit リポジトリの表示

ポリシーに関するベストプラクティス

ID ベースのポリシーは、ユーザーのアカウント内の CodeCommit リソースを誰かが作成、アクセス、または削除できる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 管理ポリシーを開始し、最小特権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の付与を開始するには、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 管理ポリシーを使用します。これらは で使用できます 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 マネージドポリシー」または「ジョブ機能の AWS マネージドポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定する場合は、タスクの実行に必要な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 を使用して許可を適用する方法の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーとアクセス許可」を参照してください。
- IAM ポリシーで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、アクションや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 SSL を使用して送信するように指定できます。条件を使用して、 などの特定の を通じてサービスアクションが使用される場合に AWS のサービス、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、「IAM ユーザーガイド」の「IAM JSON ポリシー要素:条件」を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer でポリシーを検証する」を参照してください。
- 多要素認証 (MFA) を要求する – 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、セキュリティを強化するために MFA を有効にします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細

については、「IAM ユーザーガイド」の「MFA を使用した安全な API アクセス」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。

CodeCommit コンソールの使用

AWS CodeCommit コンソールにアクセスするには、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、Amazon Web Services アカウントの CodeCommit リソースの詳細をリストおよび表示できます。最小限必要な許可よりも厳しく制限されたアイデンティティベースポリシーを作成すると、そのポリシーを添付したエンティティ (IAM ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

これらのエンティティが CodeCommit コンソールを引き続き使用できるようにするには、エンティティに次の AWS 管理ポリシーもアタッチします。詳細については、IAM ユーザーガイドのユーザーへのアクセス許可の追加を参照してください。

詳細については、「CodeCommit でのアイデンティティベースのポリシー (IAM ポリシー) の使用」を参照してください。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザーが自分の許可を表示できるようにする

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API 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。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
```

```
                "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 CodeCommit #####の表示

アイデンティティベースのポリシーの条件を使用して、タグに基づいて CodeCommit リソースへのアクセスを制御できます。これを行う方法を示すポリシーの例については、「例 5: タグを使用してリポジトリに対するアクションを許可または拒否する」を参照してください。

詳細については、IAM ユーザーガイドの IAM JSON ポリシー要素: 条件を参照してください。

AWS CodeCommit ID とアクセスのトラブルシューティング

次の情報は、CodeCommit と IAM の使用に伴って発生する可能性がある一般的な問題の診断や修復に役立ちます。

トピック

- CodeCommit でアクションを実行する権限がない
- iam:PassRole を実行する認可がない
- アクセスキーを表示したい

- 管理者として CodeCommit へのアクセスを他のユーザーに許可したい
- 自分の Amazon Web Services アカウント以外のユーザーに CodeCommit リソースへのアクセスを許可したい

CodeCommit でアクションを実行する権限がない

がアクションを実行する権限がないと AWS Management Console 通知した場合は、管理者に連絡して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。

詳細については、「CodeCommit コンソールを使用するために必要なアクセス許可」を参照してください

iam:PassRole を実行する認可がない

`iam:PassRole` アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して CodeCommit にロールを渡せるようにする必要があります。

一部の AWS のサービス 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、そのサービスに既存のロールを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。

以下の例のエラーは、`marymajor` という IAM ユーザーがコンソールを使用して CodeCommit でアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。メアリーには、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 のポリシーを更新してメアリーに `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

アクセスキーを表示したい

IAM ユーザーアクセスキーを作成した後は、いつでもアクセスキー ID を表示できます。ただし、シークレットアクセスキーを再表示することはできません。シークレットアクセスキーを紛失した場合は、新しいアクセスキーペアを作成する必要があります。

アクセスキーは、アクセスキー ID (例: `AKIAIOSFODNN7EXAMPLE`) とシークレットアクセスキー (例: `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`) の 2 つで構成されています。ユーザー名とパスワードと同様に、リクエストを認証するために、アクセスキー ID とシークレットアクセスキーの両方を使用する必要があります。ユーザー名とパスワードと同様に、アクセスキーは安全に管理してください。

> **Important**
>
> 正規のユーザー ID を確認するためであっても、アクセスキーを第三者に提供しないでください。これにより、 への永続的なアクセス権をユーザーに付与できます AWS アカウント。

アクセスキーペアを作成する場合、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するように求めるプロンプトが表示されます。このシークレットアクセスキーは、作成時にのみ使用できます。シークレットアクセスキーを紛失した場合、IAM ユーザーに新規アクセスキーを追加する必要があります。アクセスキーは最大 2 つまで持つことができます。既に 2 つある場合は、新規キーペアを作成する前に、いずれかを削除する必要があります。手順を表示するには、IAM ユーザーガイドの「アクセスキーの管理」を参照してください。

管理者として CodeCommit へのアクセスを他のユーザーに許可したい

CodeCommit へのアクセスを他のユーザーに許可するには、アクセスが必要なユーザーまたはアプリケーションにアクセス許可を付与する必要があります。 AWS IAM Identity Center を使用してユーザーとアプリケーションを管理する場合は、アクセスレベルを定義するアクセス許可セットをユーザーまたはグループに割り当てます。アクセス許可セットは、ユーザーまたはアプリケーションに関連付けられている IAM ロールに自動的に IAM ポリシーを作成して割り当てます。詳細については、「AWS IAM Identity Center ユーザーガイド」の「アクセス許可セット」を参照してください。

IAM アイデンティティセンターを使用していない場合は、アクセスを必要としているユーザーまたはアプリケーションの IAM エンティティ (ユーザーまたはロール) を作成する必要があります。次に、CodeCommit の適切なアクセス許可を付与するポリシーを、そのエンティティにアタッチする必要があります。アクセス許可が付与されたら、ユーザーまたはアプリケーション開発者に認証情報を提供します。これらの認証情報を使用して AWSにアクセスします。IAM ユーザー、グループ、ポリシー、アクセス許可の作成の詳細については、「IAM ユーザーガイド」の「IAM アイデンティティ」と「IAM のポリシーとアクセス許可」を参照してください。

自分の Amazon Web Services アカウント以外のユーザーに CodeCommit リソースへのアクセスを許可したい

詳細については、「ロールを使用して AWS CodeCommit リポジトリへのクロスアカウントアクセスを設定する」を参照してください。

の耐障害性 AWS CodeCommit

AWS グローバルインフラストラクチャは、 AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。 は、低レイテンシー、高スループット、および高度に冗長なネットワークで接続された、物理的に分離および分離された複数のアベイラビリティーゾーン AWS リージョン を提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく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。アベイラビリティーゾーンは、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が高く、フォールトトレラントで、スケーラブルです。

CodeCommit リポジトリまたは CodeCommit 承認ルールテンプレートは、それが作成された AWS リージョン に存在します。詳細については、「のリージョンと Git 接続エンドポイント AWS CodeCommit」を参照してください。リポジトリの復元性を高めるために、2 つのリポジトリに同時にプッシュするように Git クライアントを設定できます。詳細については、「追加の Git リポジトリにコミットをプッシュする」を参照してください。

AWS リージョン およびアベイラビリティーゾーンの詳細については、AWS 「グローバルインフラストラクチャ」を参照してください。

のインフラストラクチャセキュリティ AWS CodeCommit

マネージドサービスである AWS CodeCommit は、ホワイトペーパー「Amazon Web Services: セキュリティプロセスの概要」に記載されている AWS グローバルネットワークセキュリティ手順で保護されています。

AWS 公開された API コールを使用して、ネットワーク経由で CodeCommit にアクセスします。クライアントで Transport Layer Security (TLS) 1.0 以降がサポートされている必要があります。TLS 1.2 以降が推奨されています。また、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもクライアントでサポートされている必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

リクエストは、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットの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) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

これらの API オペレーションは、任意のネットワークロケーションから呼び出すことができます。ただし、CodeCommit は送信元 IP アドレスに基づく制限をサポートします。また、CodeCommit ポリシーを使用して、特定の Amazon Virtual Private Cloud (Amazon VPC) エンドポイントまたは特定の VPC からのアクセスを制御することもできます。これにより、実質的に、ネットワーク内の特定の VPC からのみ、特定の CodeCommit リソースへの AWS ネットワークアクセスが分離されます。

詳細については、次を参照してください。

- 例 1: ユーザーが 1 つの で CodeCommit オペレーションを実行できるようにする AWS リージョン
- 例 3: 指定した IP アドレス範囲から接続するユーザーにリポジトリへのアクセスを許可する
- インターフェイス VPC エンドポイント AWS CodeCommit での の使用

モニタリング AWS CodeCommit

モニタリングは、CodeCommit およびその他の AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。 には、CodeCommit をモニタリングし、問題が発生したときに報告し、必要に応じて自動アクションを実行するための以下のモニタリングツール AWS が用意されています。

- Amazon EventBridge を使用して AWS サービスを自動化し、アプリケーションの可用性の問題やリソースの変更などのシステムイベントに自動的に対応できます。 AWS のサービスからのイベントは、ほぼリアルタイムに EventBridge に提供されます。簡単なルールを記述して、注目するイベントと、イベントがルールに一致した場合に自動的に実行するアクションを指定できます。詳細については、Amazon EventBridge ユーザーガイドおよび Amazon EventBridge および Amazon CloudWatch Events の CodeCommit イベントのモニタリング を参照してください。
- Amazon CloudWatch Events は、 AWS リソースの変更を示すシステムイベントをほぼリアルタイムのストリームとして提供します。CloudWatch Events は、特定のイベントを監視し、これらのイベントが発生したときに他の AWS サービスで自動アクションをトリガーするルールを記述できるため、自動イベント駆動型コンピューティングを有効にします。詳細については、Amazon CloudWatch Events ユーザーガイドおよび Amazon EventBridge および Amazon CloudWatch Events の CodeCommit イベントのモニタリング を参照してください。
- Amazon CloudWatch Logs は、CloudTrail や他のソースからログファイルを監視、保存、およびアクセスするために使用できます。CloudWatch Logs は、ログファイル内の情報を監視し、特定のしきい値が満たされたときに通知します。高い耐久性を備えたストレージにログデータをアーカイブすることも可能です。詳細については、「Amazon CloudWatch Logs ユーザーガイド」を参照してください。
- AWS CloudTrail は、Amazon Web Services アカウントによって、または Amazon Web Services アカウントに代わって行われた API コールおよび関連イベントをキャプチャし、指定した Amazon S3 バケットにログファイルを配信します。 AWSを呼び出したユーザーとアカウント、呼び出し元のソース IP アドレス、および呼び出しの発生日時を特定できます。詳細については、AWS CloudTrail ユーザーガイド、および を使用した AWS CodeCommit API コールのログ記録 AWS CloudTrail を参照してください。

Amazon EventBridge および Amazon CloudWatch Events の CodeCommit イベントのモニタリング

EventBridge で AWS CodeCommit イベントをモニタリングできます。EventBridge は、独自のアプリケーション、software-as-a-service (SaaS) アプリケーション、および AWS サービスからリアルタイムデータのストリームを配信します。 EventBridge EventBridge は、そのデータを AWS Lambda や Amazon Simple Notification Service などのターゲットにルーティングします。これらのイベントはAmazon CloudWatch Events に表示されるイベントと同じです。Amazon CloudWatch Events は、 AWS リソースの変更を記述するシステムイベントのほぼリアルタイムのストリームを提供します。

次の例は、CodeCommit のイベントを示しています。

> **Note**
>
> CodeCommit は、イベント内のセッションタグに含まれる `displayName` および `emailAddress` の情報の提供をサポートします (その情報が利用可能な場合)。詳細については、「セッションタグ」および「CodeCommit で ID 情報を提供するためのタグの使用」を参照してください。

トピック

- referenceCreated イベント
- referenceUpdated イベント
- referenceDeleted イベント
- unreferencedMergeCommitCreated イベント
- commentOnCommitCreated イベント
- commentOnCommitUpdated イベント
- commentOnPullRequestCreated イベント
- commentOnPullRequestUpdated イベント
- pullRequestCreated イベント
- pullRequestSourceBranchUpdated イベント
- pullRequestStatusChanged イベント

- pullRequestMergeStatusUpdated イベント
- approvalRuleTemplateCreated イベント
- approvalRuleTemplateUpdated イベント
- approvalRuleTemplateDeleted イベント
- approvalRuleTemplateAssociatedWithRepository イベント
- approvalRuleTemplateDisassociatedWithRepository イベント
- approvalRuleTemplateBatchAssociatedWithRepositories イベント
- approvalRuleTemplateBatchDisassociatedFromRepositories イベント
- pullRequestApprovalRuleCreated イベント
- pullRequestApprovalRuleDeleted イベント
- pullRequestApprovalRuleOverridden イベント
- pullRequestApprovalStateChanged イベント
- pullRequestApprovalRuleUpdated イベント
- reactionCreated イベント
- reactionUpdated イベント

referenceCreated イベント

この例では、myBranch という名前のブランチが MyDemoRepo というリポジトリに作成されています。

```
{
   "version": "0",
   "id": "01234567-EXAMPLE",
   "detail-type": "CodeCommit Repository State Change",
   "source": "aws.codecommit",
   "account": "123456789012",
   "time": "2019-06-12T10:23:43Z",
   "region": "us-east-2",
   "resources": [
     "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
   ],
   "detail": {
     "event": "referenceCreated",
```

```
      "repositoryName": "MyDemoRepo",
      "repositoryId": "12345678-1234-5678-abcd-12345678abcd",
      "referenceType": "branch",
      "referenceName": "myBranch",
      "referenceFullName": "refs/heads/myBranch",
      "commitId": "3e5983DESTINATION"
    }
  }
```

referenceUpdated イベント

この例のイベントでは、myBranch という名前のブランチが MyDemoRepo という名前のリポジトリ内のマージによって更新されています 。

```
{
   "version": "0",
   "id": "01234567-EXAMPLE",
   "detail-type": "CodeCommit Repository State Change",
   "source": "aws.codecommit",
   "account": "123456789012",
   "time": "2019-06-12T10:23:43Z",
   "region": "us-east-2",
   "resources": [
     "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
   ],
   "detail": {
     "event": "referenceUpdated",
     "repositoryName": "MyDemoRepo",
     "repositoryId": "12345678-1234-5678-abcd-12345678abcd",
     "referenceType": "branch",
     "referenceName": "myBranch",
     "referenceFullName": "refs/heads/myBranch",
     "commitId": "7f0103fMERGE",
     "oldCommitId": "3e5983DESTINATION",
     "baseCommitId": "3e5a9bf1BASE",
     "sourceCommitId": "26a8f2SOURCE",
     "destinationCommitId": "3e5983DESTINATION",
     "mergeOption": "THREE_WAY_MERGE",
     "conflictDetailsLevel": "LINE_LEVEL",
     "conflictResolutionStrategy": "AUTOMERGE"
   }
}
```

referenceDeleted イベント

この例では、myBranch という名前のブランチが MyDemoRepo というリポジトリで削除されています。

```
{
  "version": "0",
  "id": "01234567-EXAMPLE",
  "detail-type": "CodeCommit Repository State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-06-12T10:23:43Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "event": "referenceDeleted",
    "repositoryName": "MyDemoRepo",
    "repositoryId": "12345678-1234-5678-abcd-12345678abcd",
    "referenceType": "branch",
    "referenceName": "myBranch",
    "referenceFullName": "refs/heads/myBranch",
    "oldCommitId": "26a8f2EXAMPLE"
  }
}
```

unreferencedMergeCommitCreated イベント

この例のイベントでは、参照されていないマージコミットが MyDemoRepo という名前のリポジトリに作成されています。

```
{
  "version": "0",
  "id": "01234567-EXAMPLE",
  "detail-type": "CodeCommit Repository State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-06-12T10:23:43Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
```

```
  ],
  "detail": {
    "event": "unreferencedMergeCommitCreated",
    "repositoryName": "MyDemoRepo",
    "repositoryId": "12345678-1234-5678-abcd-12345678abcd",
    "commitId": "7f0103fMERGE",
    "baseCommitId": "3e5a9bf1BASE",
    "sourceCommitId": "26a8f2SOURCE",
    "destinationCommitId": "3e5983DESTINATION",
    "mergeOption": "SQUASH_MERGE",
    "conflictDetailsLevel": "LINE_LEVEL",
    "conflictResolutionStrategy": "AUTOMERGE"
  }
}
```

commentOnCommitCreated イベント

この例では、Mary_Major という名前のフェデレーティッドユーザーがコミットについてコメントしています。この例では、フェデレーテッド ID プロバイダーにより displayName と emailAddress のセッションタグが設定されています。その情報はイベントに含まれます。

```
{
  "version": "0",
  "id": "e9dce2e9-EXAMPLE",
  "detail-type": "CodeCommit Comment on Commit",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-09-29T20:20:39Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "beforeCommitId": "3c5dEXAMPLE",
    "repositoryId": "7dd1EXAMPLE...",
    "inReplyTo": "695bEXAMPLE...",
    "notificationBody": "A comment event occurred in the following repository:
 MyDemoRepo. The display name for the user is Mary Major. The email address for
 the user is mary_major@example.com. The user arn:aws:sts::123456789012:federated-
user/Mary_Major made a comment. The comment was made on the following comment ID:
 463bEXAMPLE.... For more information, go to the AWS CodeCommit console at https://us-
east-2.console.aws.amazon.com/codecommit/home?region=us-east-2#/repository/MyDemoRepo/
compare/3c5dEXAMPLE...f4d5EXAMPLE#463bEXAMPLE....",
```

```
    "commentId": "463bEXAMPLE...",
    "afterCommitId": "f4d5EXAMPLE",
    "event": "commentOnCommitCreated",
    "repositoryName": "MyDemoRepo",
    "callerUserArn": "arn:aws:sts::123456789012:federated-user/Mary_Major",
    "displayName": "Mary Major",
    "emailAddress": "mary_major@example.com"
  }
}
```

commentOnCommitUpdated イベント

この例のイベントでは、Admin というセッション名を持つ Mary_Major というロールを引き受けるユーザーがコミットについてのコメントを編集しました。この例では、ロールに displayName と emailAddress の設定済みセッションタグが含まれていました 。その情報はイベントに含まれます。

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Comment on Commit",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "afterCommitId": "53812581",
    "beforeCommitId": "03314446",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "commentId": "a7e5471e-EXAMPLE",
    "event": "commentOnCommitUpdated",
    "inReplyTo": "bdb07d47-EXAMPLE",
    "notificationBody": "A comment event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. The display name for the user is Mary
 Major. The email address for the user is mary_major@example.com. The user
 arn:aws:sts::123456789012:federated-user/Mary_Major updated a comment or
 replied to a comment. The comment was made on the following comment ID:
 bdb07d47-6fe9-47b0-a839-b93cc743b2ac:468cd1cb-2dfb-4f68-9636-8de52431d1d6.
 For more information, go to the AWS CodeCommit console https://us-
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/
```

```
compare/033144464617842958996982309670958225176 8/.../5381258150293783361471680277136017291382?
region\u003dus-east-2",
    "repositoryId": "12345678-1234-1234-1234-123456789012",
    "repositoryName": "MyDemoRepo",
    "displayName": "Mary Major",
    "emailAddress": "mary_major@example.com"
  }
}
```

commentOnPullRequestCreated イベント

この例のイベントでは、Saanvi_Sarkar という名前のフェデレーティッドユーザーがプルリクエストにコメントしました。この例では、フェデレーテッド ID プロバイダーにより displayName と emailAddress のセッションタグが設定されています。その情報はイベントに含まれます。

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Comment on Pull Request",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "beforeCommitId": "3c5dEXAMPLE",
    "repositoryId": "7dd1EXAMPLE...",
    "inReplyTo": "695bEXAMPLE...",
    "notificationBody": "A comment event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. The display name for the user is Saanvi
 Sarkar. The email address for the user is saanvi_sarkar@example.com. The user
 arn:aws:sts::123456789012:federated-user/Saanvi_Sarkar made a comment. The comment
 was made on the following Pull Request: 201. For more information, go to the AWS
 CodeCommit console https://us-east-2.console.aws.amazon.com/codecommit/home?region=us-
east-2#/repository/MyDemoRepo/pull-request/201/activity#3276EXAMPLE...",
    "commentId": "463bEXAMPLE...",
    "afterCommitId": "f4d5EXAMPLE",
    "event": "commentOnPullRequestCreated",
    "repositoryName": "MyDemoRepo",
    "callerUserArn": "arn:aws:sts::123456789012:federated-user/Saanvi_Sarkar",
    "pullRequestId": "201",
```

```
    "displayName": "Saanvi Sarkar",
    "emailAddress": "saanvi_sarkar@example.com"
  }
}
```

commentOnPullRequestUpdated イベント

この例では、Saanvi_Sarkar という名前のフェデレーティッドユーザーがプルリクエストのコメントを編集しました。この例では、フェデレーテッド ID プロバイダーにより displayName と emailAddress のセッションタグが設定されています。その情報はイベントに含まれます。

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Comment on Pull Request",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "afterCommitId": "96814774EXAMPLE",
    "beforeCommitId": "6031971EXAMPLE",
    "callerUserArn": "arn:aws:sts::123456789012:federated-user/Saanvi_Sarkar",
    "commentId": "40cb52f0-EXAMPLE",
    "event": "commentOnPullRequestUpdated",
    "inReplyTo": "1285e713-EXAMPLE",
    "notificationBody": "A comment event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. The display name for the user is Saanvi
 Sarkar. The email address for the user is saanvi_sarkar@example.com. The user
 arn:aws:sts::123456789012:federated-user/Saanvi_Sarkar updated a comment or
 replied to a comment. The comment was made on the following Pull Request:
 1. For more information, go to the AWS CodeCommit console https://us-
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/1/activity#40cb52f0-aac7-4c43-b771-601eff02EXAMPLE",
    "pullRequestId": "1",
    "repositoryId": "12345678-1234-1234-1234-123456789012",
    "repositoryName": "MyDemoRepo"
  }
}
```

pullRequestCreated イベント

この例のイベントでは、セッション名が MyDemoRepo の Admin というロールを引き受けたユーザーによって指定された Mary_Major リポジトリにプルリクエストが作成されています。セッションタグ情報が提供されていないため、情報はイベントに含まれません。

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Pull Request State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "author": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "creationDate": "Tue Feb 9 2019 10:18:42 PDT ",
    "description": "An example description.",
    "destinationCommit": "12241970EXAMPLE",
    "destinationReference": "refs/heads/main",
    "event": "pullRequestCreated",
    "isMerged": "False",
    "lastModifiedDate": "Tue Feb 9 2019 10:18:42 PDT",
    "notificationBody": "A pull request event occurred in the following AWS CodeCommit
 repository: MyDemoRepo. User: arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major.
 Event: Created. The pull request was created with the following information: Pull
 Request ID as 1 and title as My Example Pull Request. For more information, go to the
 AWS CodeCommit console https://us-east-2.console.aws.amazon.com/codesuite/codecommit/
repositories/MyDemoRepo/pull-requests/1",
    "pullRequestId": "1",
    "pullRequestStatus": "Open",
    "repositoryNames": ["MyDemoRepo"],
    "revisionId": "bdc0cb9bEXAMPLE",
    "sourceCommit": "2774290EXAMPLE",
    "sourceReference": "refs/heads/test-branch",
    "title": "My Example Pull Request"
  }
}
```

pullRequestSourceBranchUpdated イベント

この例のイベントでは、セッション名が Admin の Mary_Major という名前のロールを引き受けたユーザーが、ID が 1 のプルリクエストの test-branch という名前のソースブランチを更新しました。

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Pull Request State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "author": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "creationDate": "Tue Feb 9 2019 10:18:42 PDT",
    "description": "An example description.",
    "destinationCommit": "7644990EXAMPLE",
    "destinationReference": "refs/heads/main",
    "event": "pullRequestSourceBranchUpdated",
    "isMerged": "False",
    "lastModifiedDate": "Tue Feb 9 2019 10:18:42 PDT",
    "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:sts::123456789012:assumed-role/
Admin/Mary_Major. Event: Updated.  The user updated the following pull request:
 1. The pull request was updated with one or more commits to the source branch:
 test-branch. For more information, go to the AWS CodeCommit console https://us-
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/1?region\u003dus-east-2",
    "pullRequestId": "1",
    "pullRequestStatus": "Open",
    "repositoryNames": ["MyDemoRepo"],
    "revisionId": "bdc0cb9b4EXAMPLE",
    "sourceCommit": "64875001EXAMPLE",
    "sourceReference": "refs/heads/test-branch",
    "title": "My Example Pull Request"
  }
```

```
}
```

pullRequestStatusChanged イベント

この例のイベントでは、セッション名が Admin の Mary_Major という名前のロールを引き受け、ID が 1 のプルリクエストを閉じました。プルリクエストはマージされませんでした。

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Pull Request State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "author": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "creationDate": "Tue Jun 18 10:34:20 PDT 2019",
    "description": "An example description.",
    "destinationCommit": "95149731EXAMPLE",
    "destinationReference": "refs/heads/main",
    "event": "pullRequestStatusChanged",
    "isMerged": "False",
    "lastModifiedDate": "Tue Jun 18 10:34:20 PDT 2019",
    "notificationBody": "A pull request event occurred in the following AWS CodeCommit
 repository: MyDemoRepo. arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major
 updated the following PullRequest 1. The pull request status has been updated. The
 status is closed. For more information, go to the AWS CodeCommit console https://
us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/1?region\u003dus-east-2",
    "pullRequestId": "1",
    "pullRequestStatus": "Closed",
    "repositoryNames": ["MyDemoRepo"],
    "revisionId": "bdc0cb9bEXAMPLE",
    "sourceCommit": "4409936EXAMPLE",
    "sourceReference": "refs/heads/test-branch",
    "title": "My Example Pull Request"
  }
```

```
}
```

pullRequestMergeStatusUpdated イベント

この例のイベントでは、セッション名が Admin の Mary_Major という名前のロールを引き受けるユーザーが、ID 1 のプルリクエストをマージしました。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "CodeCommit Pull Request State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-06-12T10:23:43Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "author": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "creationDate": "Mon Mar 11 14:42:31 PDT 2019",
    "description": "An example description.",
    "destinationCommit": "4376719EXAMPLE",
    "destinationReference": "refs/heads/main",
    "event": "pullRequestMergeStatusUpdated",
    "isMerged": "True",
    "lastModifiedDate": "Mon Mar 11 14:42:31 PDT 2019",
    "mergeOption": "FAST_FORWARD_MERGE",
    "notificationBody": "A pull request event occurred in the following AWS CodeCommit
 repository: MyDemoRepo. arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major
 updated the following PullRequest 1. The pull request merge status has been updated.
 The status is merged. For more information, go to the AWS CodeCommit console https://
us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/1?region\u003dus-east-2",
    "pullRequestId": "1",
    "pullRequestStatus": "Closed",
    "repositoryNames": ["MyDemoRepo"],
    "revisionId": "bdc0cb9beEXAMPLE",
    "sourceCommit": "0701696EXAMPLE",
    "sourceReference": "refs/heads/test-branch",
    "title": "My Example Pull Request"
  }
```

```
}
```

approvalRuleTemplateCreated イベント

この例のイベントでは、Mary_Major というユーザー名の IAM ユーザーが 2-approvers-required-for-main という名前の承認規則テンプレートを作成しました。

```
{
    "version": "0",
    "id": "f7702227-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:02:27Z",
    "region": "us-east-2",
    "resources": [],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "d7385967-EXAMPLE",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:02:14 UTC 2019",
        "event": "approvalRuleTemplateCreated",
        "lastModifiedDate": "Wed Nov 06 19:02:14 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user/Mary_Major.
  Additional information: An approval rule template with the following name has been
 created: 2-approvers-required-for-main. The ID of the created template is: d7385967-
EXAMPLE. For more information, go to the AWS CodeCommit console.",
        "repositories": {}
    }
}
```

approvalRuleTemplateUpdated イベント

この例のイベントでは、Mary_Major というユーザー名の IAM ユーザーが 2-approvers-required-for-main という名前の承認規則テンプレートを編集しました。承認ルールテンプレートは、どのリポジトリにも関連付けられていません。

```
{
  "version": "0",
  "id": "66403118-EXAMPLE",
```

```
  "detail-type": "CodeCommit Approval Rule Template Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-11-12T23:03:30Z",
  "region": "us-east-2",
  "resources": [

  ],
  "detail": {
    "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
    "approvalRuleTemplateId": "c9d2b844-EXAMPLE",
    "approvalRuleTemplateName": "2-approvers-required-for-main",
    "callerUserArn": "arn:aws:iam::123456789012:user\Mary_Major",
    "creationDate": "Tue Nov 12 23:03:06 UTC 2019",
    "event": "approvalRuleTemplateDeleted",
    "lastModifiedDate": "Tue Nov 12 23:03:20 UTC 2019",
    "notificationBody": "A approval rule template event occurred in the following AWS
 CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user\Mary_Major.
 Additional information: An approval rule template with the following name has been
 deleted: 2-approvers-required-for-main. The ID of the updated template is: c9d2b844-
EXAMPLE. For more information, go to the AWS CodeCommit console.",
    "repositories": {}
  }
}
```

approvalRuleTemplateDeleted イベント

この例のイベントでは、Mary_Major というユーザー名の IAM ユーザーが 2-approvers-required-for-main という名前の承認規則テンプレートを削除しました。承認ルールテンプレートは、どのリポジトリにも関連付けられていません。

```
{
  "version": "0",
  "id": "66403118-EXAMPLE",
  "detail-type": "CodeCommit Approval Rule Template Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-11-12T23:03:30Z",
  "region": "us-east-2",
  "resources": [],
  "detail": {
    "approvalRuleTemplateContentSha256": "4f3de6632EXAMPLE",
    "approvalRuleTemplateId": "c9d2b844-EXAMPLE",
```

```
    "approvalRuleTemplateName": "2-approvers-required-for-main",
    "callerUserArn": "arn:aws:iam::123456789012:user\Mary_Major",
    "creationDate": "Tue Nov 12 23:03:06 UTC 2019",
    "event": "approvalRuleTemplateUpdated",
    "lastModifiedDate": "Tue Nov 12 23:03:20 UTC 2019",
    "notificationBody": "A approval rule template event occurred in the following AWS
 CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user\Mary_Major.
 Additional information: An approval rule template with the following name has
 been updated: 2-approvers-required-for-main. The ID of the updated template is:
 c9d2b844-EXAMPLE. The after rule template content SHA256 is 4f3de663EXAMPLE. For more
 information, go to the AWS CodeCommit console.",
    "repositories": {}
  }
}
```

approvalRuleTemplateAssociatedWithRepository イベント

この例のイベントでは、Mary_Major というユーザー名の IAM ユーザーが、2-approvers-required-for-main という名前の承認規則テンプレートを MyDemoRepo という名前のリポジトリに関連付けました。

```
{
    "version": "0",
    "id": "ea1c6d73-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:02:27Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "d7385967-EXAMPLE",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:02:14 UTC 2019",
        "event": "approvalRuleTemplateAssociatedWithRepository",
        "lastModifiedDate": "Wed Nov 06 19:02:14 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user/Mary_Major.
 Additional information: An approval rule template has been associated with the
```

```
 following repository: [MyDemoRepo]. For more information, go to the AWS CodeCommit
 console.",
        "repositories": {
            "MyDemoRepo": "92ca7bf2-d878-49ed-a994-336a6cc7c574"
        }
    }
}
```

approvalRuleTemplateDisassociatedWithRepository イベント

この例のイベントでは、Mary_Major というユーザー名の IAM ユーザーが、2-approvers-required-for-main という名前の承認規則テンプレートを MyDemoRepo という名前のリポジトリから関連付けを解除しました。

```
{
    "version": "0",
    "id": "ea1c6d73-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:02:27Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "d7385967-EXAMPLE",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:02:14 UTC 2019",
        "event": "approvalRuleTemplateDisassociatedFromRepository",
        "lastModifiedDate": "Wed Nov 06 19:02:14 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user/Mary_Major.
 Additional information: An approval rule template has been disassociated from the
 following repository: [MyDemoRepo]. For more information, go to the AWS CodeCommit
 console.",
        "repositories": {
            "MyDemoRepo": "92ca7bf2-d878-49ed-a994-336a6cc7c574"
        }
    }
```

```
}
```

approvalRuleTemplateBatchAssociatedWithRepositories イベント

この例のイベントでは、Mary_Major というユーザー名の IAM ユーザーが、2-approvers-required-for-main という名前の承認規則テンプレートを MyDemoRepo という名前のリポジトリと MyTestRepo という名前のリポジトリに一括で関連付けしました。

```
{
    "version": "0",
    "id": "0f861e5b-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-12T23:39:09Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "c71c1fe0-EXAMPLE",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Tue Nov 12 23:38:57 UTC 2019",
        "event": "batchAssociateApprovalRuleTemplateWithRepositories",
        "lastModifiedDate": "Tue Nov 12 23:38:57 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user\Mary_Major.
 Additional information: An approval rule template has been batch associated with the
 following repository names: [MyDemoRepo, MyTestRepo]. For more information, go to the
 AWS CodeCommit console.",
        "repositories": {
            "MyDemoRepo": "MyTestRepo"
        }
    }
}
```

approvalRuleTemplateBatchDisassociatedFromRepositories イベント

この例のイベントでは、Mary_Major というユーザー名の IAM ユーザーが、2-approvers-required-for-main という名前の承認規則テンプレートを MyDemoRepo という名前のリポジトリと MyTestRepo という名前のリポジトリから一括で関連付けを解除しました。

```
{
    "version": "0",
    "id": "e08fc996-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-12T23:39:09Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "c71c1fe0-ff91-4db4-9a45-a86a7b6c474f",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Tue Nov 12 23:38:57 UTC 2019",
        "event": "batchDisassociateApprovalRuleTemplateFromRepositories",
        "lastModifiedDate": "Tue Nov 12 23:38:57 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user/Mary_Major.
 Additional information: An approval rule template has been batch disassociated from
 the following repository names: [MyDemoRepo, MyTestRepo]. For more information, go to
 the AWS CodeCommit console.",
        "repositories": {
            "MyDemoRepo": "MyTestRepo"
        }
    }
}
```

pullRequestApprovalRuleCreated イベント

この例のイベントでは、Mary_Major というユーザー名の IAM ユーザーが 1-approver-needed という名前の承認規則を ID 227 のプルリクエストに作成しました。

```
{
```

```
    "version": "0",
    "id": "ad860f12-EXAMPLE",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleContentSha256": "f742eebbEXAMPLE",
        "approvalRuleId": "0a9b5dfc-EXAMPLE",
        "approvalRuleName": "1-approver-needed",
        "author": "arn:aws:iam::123456789012:user/Mary_Major",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleCreated",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/Mary_Major.
 Event: Updated. Pull request: 227. Additional information: An approval rule has been
 created with the following name: 1-approver-needed. For more information, go to the
 AWS CodeCommit console https://us-east-2.console.aws.amazon.com/codesuite/codecommit/
repositories/MyDemoRepo/pull-requests/227?region=us-east-2",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecab3EXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

pullRequestApprovalRuleDeleted イベント

この例のイベントでは、Mary_Major というユーザー名の IAM ユーザーが ID 1-approver-needed のプルリクエストの 227 という名前の承認規則を削除しました。名前が Saanvi_Sarkar である IAM ユーザーが、最初に承認ルールを作成しました。


```
{
    "version": "0",
    "id": "c1c3509d-EXAMPLE",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleContentSha256": "f742eebbEXAMPLE",
        "approvalRuleId": "0a9b5dfc-EXAMPLE",
        "approvalRuleName": "1-approver-needed",
        "author": "arn:aws:iam::123456789012:user/Saanvi_Sarkar",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleDeleted",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/Mary_Major.
 Event: Created. Pull request: 227. Additional information: An approval rule has been
 deleted: 1-approver-needed was deleted. For more information, go to the AWS CodeCommit
 console https://us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/
MyDemoRepo/pull-requests/227?region=us-east-2",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
```

```
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

pullRequestApprovalRuleOverridden イベント

この例のイベントでは、IAM ユーザー名 Mary_Major のユーザーによって、プルリクエストの承認規則の要件が無効にされています (OVERRIDE)。プルリクエストは、Li_Juan という IAM ユーザー名を持つユーザーによって作成されました。

```
{
    "version": "0",
    "id": "52d2cb73-EXAMPLE",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "author": "arn:aws:iam::123456789012:user/Li_Juan",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleOverridden",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/Mary_Major.
 Event: Updated. Pull request name: 227. Additional information: An override
 event has occurred for the approval rules for this pull request. Override status:
 OVERRIDE. For more information, go to the AWS CodeCommit console https://us-
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/227?region=us-east-2",
        "overrideStatus": "OVERRIDE",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
```

```
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

この例のイベントでは、プルリクエストの承認規則要件が回復されています (REVOKE)。

```
{
    "version": "0",
    "id": "2895482d-13eb-b783-270d-76588e6029fa",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "author": "arn:aws:iam::123456789012:user/Li_Juan",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleOverridden",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following
 AWS CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/
Mary_Major. Event: Updated. Pull request name: 227. Additional information: An
 override event has occurred for the approval rules for this pull request. Override
 status: REVOKE. For more information, go to the AWS CodeCommit console https://
us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/227?region=us-east-2",
        "overrideStatus": "REVOKE",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
```

```
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

pullRequestApprovalStateChanged イベント

この例のイベントでは、IAM ユーザー名が Mary_Major のユーザーによってプルリクエストが承認されています。

```
{
    "version": "0",
    "id": "53e5d7e9-986c-1ebf-9d8b-ebef5596da0e",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalStatus": "APPROVE",
        "author": "arn:aws:iam::123456789012:user/Li_Juan",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalStateChanged",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following
 AWS CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/
Mary_Major. Event: Updated. Pull request name: 227. Additional information:
 A user has changed their approval state for the pull request. State change:
 APPROVE. For more information, go to the AWS CodeCommit console https://us-
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/227?region=us-east-2",
```

```
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

この例のイベントでは、IAM ユーザー名が Mary_Major のユーザーによってプルリクエストの承認が取り消されています。

```
{
    "version": "0",
    "id": "25e183d7-d01a-4e07-2bd9-b2d56ebecc81",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalStatus": "REVOKE",
        "author": "arn:aws:iam::123456789012:user/Li_Juan",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalStateChanged",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/Mary_Major.
 Event: Updated. Pull request name: 227. Additional information: A user has changed
 their approval state for the pull request. State change: REVOKE. For more information,
 go to the AWS CodeCommit console https://us-east-2.console.aws.amazon.com/codesuite/
codecommit/repositories/MyDemoRepo/pull-requests/227?region=us-east-2",
```

```
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

pullRequestApprovalRuleUpdated イベント

この例のイベントでは、IAM ユーザー名が Mary_Major のユーザーによってプルリクエストの承認規則が編集されています。このユーザーは、プルリクエストを作成したユーザーでもあります。

```
{
    "version": "0",
    "id": "21b1c819-2889-3528-1cb8-3861aacf9d42",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleContentSha256": "f742eebbEXAMPLE",
        "approvalRuleId": "0a9b5dfc-EXAMPLE",
        "approvalRuleName": "1-approver-needed",
        "author": "arn:aws:iam::123456789012:user/Mary_Major",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleUpdated",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following
 AWS CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/
```

```
Mary_Major. Event: Updated. Pull request name: 227. The content of an approval
 rule has been updated for the pull request. The name of the updated rule is: 1-
approver-needed. For more information, go to the AWS CodeCommit console https://
us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/227?region=us-east-2",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecab3EXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

reactionCreated イベント

この例のイベントでは、Mary_Major というユーザー名の IAM ユーザーによって、コメントに対するリアクションが追加されています。

```
{
   "version":"0",
   "id":"59fcccd8-217a-32ce-2b05-561ed68a1c42",
   "detail-type":"CodeCommit Comment Reaction Change",
   "source":"aws.codecommit",
   "account":"123456789012",
   "time":"2020-04-14T00:49:03Z",
   "region":"us-east-2",
   "resources":[
      "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
   ],
   "detail":{
      "callerUserArn":"arn:aws:iam::123456789012:user/Mary_Major",
      "commentId":"28930161-EXAMPLE",
      "event":"commentReactionCreated",
      "notificationBody":"A comment reaction event occurred in the following AWS
 CodeCommit Repository: MyDemoRepo. The user: arn:aws:iam::123456789012:user/Mary_Major
 made a comment reaction # to the comment with comment ID: 28930161-EXAMPLE",
      "reactionEmojis":["#"],
      "reactionShortcodes":[":thumbsdown:"],
      "reactionUnicodes":["U+1F44E"],
```

```
      "repositoryId":"12345678-1234-5678-abcd-12345678abcd",
      "repositoryName":"MyDemoRepo"
   }
}
```

reactionUpdated イベント

この例のイベントでは、Mary_Major というユーザー名の IAM ユーザーによって、コメントに対するリアクションが更新されています。ユーザーは各自のリアクションのみを更新できます。

```
{
   "version":"0",
   "id":"0844ed99-a53f-3bdb-6048-4de315516889",
   "detail-type":"CodeCommit Comment Reaction Change",
   "source":"aws.codecommit",
   "account":"123456789012",
   "time":"2020-04-22T23:19:42Z",
   "region":"us-east-2",
   "resources":[
      "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
   ],
   "detail":{
      "callerUserArn":"arn:aws:iam::123456789012:user/Mary_Major",
      "commentId":"28930161-EXAMPLE",
      "event":"commentReactionUpdated",
      "notificationBody":"A comment reaction event occurred in the following AWS
 CodeCommit Repository: MyDemoRepo. The user: arn:aws:iam::123456789012:user/Mary_Major
 updated a reaction :smile: to the comment with comment ID: 28930161-EXAMPLE",
      "reactionEmojis":[
         "#"
      ],
      "reactionShortcodes":[
         ":smile:"
      ],
      "reactionUnicodes":[
         "U+1F604"
      ],
      "repositoryId":"12345678-1234-5678-abcd-12345678abcd",
      "repositoryName":"MyDemoRepo"
   }
}
```

を使用した AWS CodeCommit API コールのログ記録 AWS CloudTrail

CodeCommit は AWS CloudTrail、CodeCommit のユーザー、ロール、または のサービスによって実行されたアクションを記録する AWS サービスCodeCommitと統合されています。CloudTrail は、CodeCommit のすべての API 呼び出しをイベントとしてキャプチャします。これには、CodeCommit コンソール、Git クライアント、CodeCommit API へのコード呼び出しからの呼び出しが含まれます。証跡を作成する場合は、CodeCommit のイベントなど、Amazon S3 バケットへの CloudTrail 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 [Event history (イベント履歴)] で最新のイベントを表示できます。CloudTrail で収集された情報を使用して、CodeCommit 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。

CloudTrail に関する詳細は、AWS CloudTrail ユーザーガイドを参照してください。

CloudTrail の CodeCommit 情報

CloudTrail は、アカウント作成時に アマゾン ウェブ サービスアカウントで有効になります。CodeCommit でアクティビティが発生すると、そのアクティビティはイベント履歴の他の AWS サービスイベントとともに CloudTrail イベントに記録されます。AWS アカウントでの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、「CloudTrail イベント履歴でのイベントの表示」を参照してください。

CodeCommit のイベントなど、アマゾン ウェブ サービスアカウント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処理するように、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 がサポートするサービスと統合
- CloudTrail 用 Amazon SNS 通知の構成
- 「複数のリージョンからCloudTrailログファイルを受け取る」および「複数のアカウントからCloudTrailログファイルを受け取る」

Amazon Web Services アカウントで CloudTrail ログ記録が有効になっている場合、CodeCommit アクションに対して行われた API コールは CloudTrail ログファイルで追跡され、他の AWS サービスレコードで書き込まれます。CloudTrail は、期間とファイルサイズに基づいて、新しいファイルをいつ作成して書き込むかを決定します。

すべての CodeCommit アクションは CloudTrail によってログに記録されます。これには、AWS CodeCommit API リファレンスに現時点では記載されていないものの、アクセス許可として参照され、CodeCommit アクセス許可リファレンス に記載されているもの (`GetObjectIdentifier` など) もあります。例えば、 `ListRepositories` (、`aws codecommit list-repositories`) AWS CLI、 `CreateRepository` (`aws codecommit create-repository`) 、 `PutRepositoryTriggers` (`aws codecommit put-repository-triggers`) アクションを呼び出すと、CloudTrail ログファイルにエントリが生成され、 `GitPull` および への Git クライアント呼び出しも生成されます`GitPush`。さらに、CodePipeline でパイプラインのソースとして CodeCommit リポジトリが設定されている場合は、CodePipeline からの `UploadArchive` などの CodeCommit アクセス許可アクションの呼び出しが表示されます。CodeCommit は を使用してリポジトリ AWS Key Management Service の暗号化と復号を行うため、CodeCommit から への呼び出し`Encrypt`と からの`Decrypt`アクションも CloudTrail ログ AWS KMS に表示されます。

各ログエントリには、リクエストの生成者に関する情報が含まれます。ログエントリのユーザー ID 情報は、次のことを確認するのに役立ちます。

- 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ティッドユーザーの一時的なセキュリティ資格情報で行われたか、または想定されたロールによって行われたか
- 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。

必要な場合はログファイルを自身の Amazon S3 バケットに保存できますが、ログファイルを自動的にアーカイブまたは削除するように Amazon S3 ライフサイクルルールを定義することもできます。デフォルトでは Amazon S3 のサーバー側の暗号化 (SSE) を使用して、ログファイルが暗号化されます。

CodeCommit ログファイルエントリの理解

CloudTrail ログファイルには、1 つ以上のログエントリを含むことができます。各エントリには、複数の JSON 形式のイベントがリストされます。ログイベントは任意のソースからの1つ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、リクエストのパラメーターなどに関する

情報が含まれます。ログエントリは、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。

> **Note**
>
> この例は、読みやすくするために書式設定しています。CloudTrail ログファイルでは、すべてのエントリとイベントが 1 行に連結されます。また、この例では 1 つの CodeCommit エントリに限定しています。実際の CloudTrail ログファイルには、複数の AWS のサービスからのエントリとイベントが記録されます。

目次

- 例: CodeCommit リポジトリをリストするためのログエントリ
- 例: CodeCommit リポジトリを作成するためのログエントリ
- 例: CodeCommit リポジトリへの Git プルコールを表すログエントリ
- 例: CodeCommit リポジトリへのプッシュの成功を表すログエントリ

例: CodeCommit リポジトリをリストするためのログエントリ

以下の例は、`ListRepositories` アクションを示す CloudTrail ログエントリです。

> **Note**
>
> `ListRepositories` はリポジトリのリストを返しますが、変更不可のレスポンスは CloudTrail ログに記録されないため、`responseElements` はログファイルに `null` として記録されます。

```
{
  "eventVersion":"1.05",
  "userIdentity": {
    "type":"IAMUser",
    "principalId":"AIDACKCEVSQ6C2EXAMPLE",
    "arn":"arn:aws:iam::444455556666:user/Mary_Major",
    "accountId":"444455556666",
    "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
```

```
  "eventTime":"2016-12-14T17:57:36Z",
  "eventSource":"codecommit.amazonaws.com",
  "eventName":"ListRepositories",
  "awsRegion":"us-east-1",
  "sourceIPAddress":"203.0.113.12",
  "userAgent":"aws-cli/1.10.53 Python/2.7.9 Windows/8 botocore/1.4.43",
  "requestParameters":null,
  "responseElements":null,
  "requestID":"cb8c167e-EXAMPLE",
  "eventID":"e3c6f4ce-EXAMPLE",
  "readOnly":true,
  "eventType":"AwsApiCall",
  "apiVersion":"2015-04-13",
  "recipientAccountId":"444455556666"
}
```

例: CodeCommit リポジトリを作成するためのログエントリ

次の例は、米国東部 (オハイオ) リージョンでの `CreateRepository` アクションを示す CloudTrail ログエントリを示しています。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
  },
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
  "eventName": "CreateRepository",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "aws-cli/1.10.53 Python/2.7.9 Windows/8 botocore/1.4.43",
  "requestParameters": {
    "repositoryDescription": "Creating a demonstration repository.",
    "repositoryName": "MyDemoRepo"
  },
  "responseElements": {
    "repositoryMetadata": {
```

```
      "arn": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "creationDate": "Dec 14, 2016 6:19:14 PM",
      "repositoryId": "8afe792d-EXAMPLE",
      "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
      "repositoryName": "MyDemoRepo",
      "accountId": "111122223333",
      "cloneUrlHttp": "https://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
      "repositoryDescription": "Creating a demonstration repository.",
      "lastModifiedDate": "Dec 14, 2016 6:19:14 PM"
    }
  },
  "requestID": "d148de46-EXAMPLE",
  "eventID": "740f179d-EXAMPLE",
  "readOnly": false,
  "resources": [
    {
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
    }
  ],
  "eventType": "AwsApiCall",
  "apiVersion": "2015-04-13",
  "recipientAccountId": "111122223333"
}
```

例: CodeCommit リポジトリへの Git プルコールを表すログエントリ

以下の例では、ローカルリポジトリが既に最新である `GitPull` アクションを表す CloudTrail ログエントリを示しています。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
```

```
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
  "eventName": "GitPull",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "git/2.11.0.windows.1",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "protocol": "HTTP",
    "dataTransferred": false,
    "repositoryName": "MyDemoRepo",
    "repositoryId": "8afe792d-EXAMPLE",
    },
  "requestID": "d148de46-EXAMPLE",
  "eventID": "740f179d-EXAMPLE",
  "readOnly": true,
  "resources": [
    {
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
      }
    ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

以下の例では、ローカルリポジトリが最新でなく、データが CodeCommit リポジトリからローカルリポジトリに転送される `GitPull` アクションを表す CloudTrail ログエントリを示しています。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
```

```
  "eventName": "GitPull",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "git/2.10.1",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "protocol": "HTTP",
    "capabilities": [
      "multi_ack_detailed",
      "side-band-64k",
      "thin-pack"
      ],
    "dataTransferred": true,
    "repositoryName": "MyDemoRepo",
    "repositoryId": "8afe792d-EXAMPLE",
    "shallow": false
    },
  "requestID": "d148de46-EXAMPLE",
  "eventID": "740f179d-EXAMPLE",
  "readOnly": true,
  "resources": [
    {
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
      }
    ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

例: CodeCommit リポジトリへのプッシュの成功を表すログエントリ

以下の例では、GitPush アクションの成功を表す CloudTrail ログエントリを示しています。プッシュの成功を表すログエントリには GitPush アクションが 2 回、含まれています。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
```

```
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
  "eventName": "GitPush",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "git/2.10.1",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "protocol": "HTTP",
    "dataTransferred": false,
    "repositoryName": "MyDemoRepo",
    "repositoryId": "8afe792d-EXAMPLE",
    },
  "requestID": "d148de46-EXAMPLE",
  "eventID": "740f179d-EXAMPLE",
  "readOnly": false,
  "resources": [
    {
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
      }
    ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
},
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
  "eventName": "GitPush",
```

```
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "git/2.10.1",
  "requestParameters": {
    "references": [
      {
        "commit": "100644EXAMPLE",
        "ref": "refs/heads/main"
        }
      ]
    },
  "responseElements": null,
  "additionalEventData": {
    "protocol": "HTTP",
    "capabilities": [
      "report-status",
      "side-band-64k"
      ],
    "dataTransferred": true,
    "repositoryName": "MyDemoRepo",
    "repositoryId": "8afe792d-EXAMPLE",
    },
  "requestID": "d148de46-EXAMPLE",
  "eventID": "740f179d-EXAMPLE",
  "readOnly": false,
  "resources": [
    {
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
      }
    ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

を使用した CodeCommit リソースの作成 AWS CloudFormation

AWS CodeCommit は と統合されています。これは AWS CloudFormation、 AWS リソースとインフラストラクチャの作成と管理に費やす時間を短縮できるように、リソースのモデル化とセットアップに役立つサービスです。必要なすべての AWS リソース (リポジトリなど) を記述するテンプレートを作成し、それらのリソースを AWS CloudFormation プロビジョニングして設定します。

を使用すると AWS CloudFormation、テンプレートを再利用して CodeCommit リソースを一貫して繰り返しセットアップできます。リソースを 1 回記述し、同じリソースを複数の AWS アカウントおよびリージョンで何度もプロビジョニングします。

CodeCommit と AWS CloudFormation テンプレート

CodeCommit および関連サービスのリソースをプロビジョニングして設定するには、AWS CloudFormation テンプレートについて理解しておく必要があります。テンプレートは、JSON や YAML でフォーマットされたテキストファイルです。これらのテンプレートは、 AWS CloudFormation スタックでプロビジョニングするリソースを記述します。JSON または YAML に慣れていない場合は、 AWS CloudFormation デザイナー を使用して AWS CloudFormation テンプレートの使用を開始できます。詳細については、「AWS CloudFormation ユーザーガイド」の「AWS CloudFormation Designer とは」を参照してください。

CodeCommit は、 でのリポジトリの作成をサポートしています。コンソールまたはコマンドラインからリポジトリを作成する AWS CloudFormation 場合とは異なり、 AWS CloudFormation を使用してリポジトリを作成し、Amazon S3 バケット内の指定された .zip ファイルから新しく作成されたリポジトリにコードを自動的にコミットできます。リポジトリの JSON および YAML テンプレートの例を含む詳細については、AWS::CodeCommit::Repository を参照してください。

を使用して CodeCommit リポジトリを作成する場合 AWS CloudFormation、AWS:CodeCommit::CodeCommit Repository Code でプロパティを設定することで、アーカイブが 20 MB 未満である限り、作成プロセスの一環としてそのリポジトリにコードをコミットするオプションがあります。コードが格納される Amazon S3 バケットを指定し、任意で BranchName プロパティを使用して、そのコードの初期コミットで作成されるデフォルトのブランチの名前を指定できます。これらのプロパティは最初のリポジトリ作成でのみ使用され、スタックの更新時には無視されます。これらのプロパティを使用して、リポジトリに追加のコミットを行ったり、最初のコミットが行われた後にデフォルトブランチの名前を変更したりすることはできません。

> **Note**
>
> 2021 年 1 月 19 日に、 は CodeCommit のデフォルトブランチの名前を master から main AWS に変更しました。この名前の変更は、CodeCommit コンソール、CodeCommit APIs、AWS SDKs、および を使用してリポジトリの初期コミットを作成するときの CodeCommit のデフォルトの動作に影響します AWS CLI。 AWS CloudFormation または を使用して作成されたリポジトリ AWS CDK で、作成の一環としてコードの初期コミットが 2021 年 3 月 4 日以降の変更と一致します。この変更は、既存のリポジトリまたはブランチには影響しません。ローカル Git クライアントを使用して初期コミットを作成するお客様は、これらの Git クライアントの設定に続いてデフォルトのブランチ名を持つことになります。詳細については、ブランチを操作する、コミットを作成する、およびブランチ設定を変更するを参照してください。

また、関連リソース (リポジトリの 通知ルール、AWS CodeBuild ビルドプロジェクト、AWS CodeDeploy アプリケーションウォン、AWS CodePipeline パイプラインなど) を作成するためのテンプレートを作成することもできます。

テンプレートの例

以下の例では、*MyDemoRepo* という名前の CodeCommit リポジトリを作成します。新しく作成されたリポジトリには、*MySourceCodeBucket* という名前の Amazon S3 バケットに保存されたコードが入力され、そのリポジトリは、*development* という名前のブランチ (リポジトリのデフォルトのブランチ) に配置されます。

> **Note**
>
> 新しいリポジトリにコミットされるコンテンツが含まれた ZIP ファイルを含む Amazon S3 バケットの名前は、ARN またはアマゾン ウェブ サービスアカウント内のバケットの名前を使用して指定できます。Amazon S3 オブジェクトキーは、Amazon S3 デベロッパーガイドで定義されています。

JSON:

```
{
    "MyRepo": {
        "Type": "AWS::CodeCommit::Repository",
```

```
        "Properties": {
            "RepositoryName": "MyDemoRepo",
            "RepositoryDescription": "This is a repository for my project with code
 from MySourceCodeBucket.",
            "Code": {
                "BranchName": "development",
                "S3": {
                    "Bucket": "MySourceCodeBucket",
                    "Key": "MyKey",
                    "ObjectVersion": "1"
                }
            }
        }
    }
}
```

YAML:

```
MyRepo:
  Type: AWS::CodeCommit::Repository
  Properties:
    RepositoryName: MyDemoRepo
    RepositoryDescription: This is a repository for my project with code from
 MySourceCodeBucket.
    Code:
      BranchName: development
      S3:
        Bucket: MySourceCodeBucket,
        Key: MyKey,
        ObjectVersion: 1
```

その他の例については、AWS::CodeCommit::Repository を参照してください。

AWS CloudFormation、CodeCommit、および AWS Cloud Development Kit (AWS CDK)

を使用して作成されたリポジトリは、作成時に AWS CloudFormation 機能 AWS CDK を使用します。 AWS CloudFormation テンプレートが CodeCommit リソースとどのように連携するかを理解することは、 AWS CDK コードの作成と管理に役立ちます。の詳細については AWS CDK、「 AWS Cloud Development Kit (AWS CDK) デベロッパーガイド」とAWS CDK 「 API リファレンス」を参照してください。

次の AWS CDK Typescript の例では、*MyDemoRepo* という名前の CodeCommit リポジトリを作成します。新しく作成されたリポジトリには、*MySourceCodeBucket* という名前の Amazon S3 バケットに保存されたコードが入力され、そのリポジトリは、*development* という名前のブランチ (リポジトリのデフォルトのブランチ) に配置されます。

```
import * as cdk from '@aws-cdk/core';
import codecommit = require('@aws-cdk/aws-codecommit');
export class CdkCodecommitStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);
    // The code creates a CodeCommit repository with a default branch name development
      new codecommit.CfnRepository(this, 'MyRepoResource', {
            repositoryName: "MyDemoRepo",
            code: {
              "branchName": "development",
              "s3": {
                "bucket": "MySourceCodeBucket",
                "key": "MyKey"
              }
            },
        }
     );
  }
}
```

の詳細 AWS CloudFormation

詳細については AWS CloudFormation、次のリソースを参照してください。

- AWS CloudFormation
- AWS CloudFormation ユーザーガイド
- AWS CloudFormation コマンドラインインターフェイスユーザーガイド

トラブルシューティング AWS CodeCommit

以下の情報は、 AWS CodeCommitでの一般的な問題のトラブルシューティングに役立ちます。

トピック

- への Git 認証情報と HTTPS 接続のトラブルシューティング AWS CodeCommit
- git-remote-codecommit と のトラブルシューティング AWS CodeCommit
- への SSH 接続のトラブルシューティング AWS CodeCommit
- 認証情報ヘルパーと への HTTPS 接続のトラブルシューティング AWS CodeCommit
- Git クライアントと のトラブルシューティング AWS CodeCommit
- アクセスエラーと のトラブルシューティング AWS CodeCommit
- 設定エラーと のトラブルシューティング AWS CodeCommit
- コンソールエラーと のトラブルシューティング AWS CodeCommit
- トリガーと のトラブルシューティング AWS CodeCommit
- デバッグを有効にする

への Git 認証情報と HTTPS 接続のトラブルシューティング AWS CodeCommit

以下の情報は、Git 認証情報と HTTPS を使用して AWS CodeCommit リポジトリに接続する際の一般的な問題のトラブルシューティングに役立ちます。

トピック

- Git 認証情報 AWS CodeCommit: ターミナルまたはコマンドラインで CodeCommit リポジトリに接続すると、認証情報の入力プロンプトが引き続き表示されます。
- Git 認証情報 AWS CodeCommit: Git 認証情報をセットアップしたが、システムが使用していない

Git 認証情報 AWS CodeCommit: ターミナルまたはコマンドラインで CodeCommit リポジトリに接続すると、認証情報の入力プロンプトが引き続き表示されます。

問題: ターミナルまたはコマンドラインから、CodeCommit リポジトリへのプッシュ、リポジトリからのプル、またはこのリポジトリとの接続を試みると、ユーザー名およびパスワードの入力を求められるため、IAM ユーザーの Git 認証情報を入力する必要がある。

解決方法: このエラーの最も一般的な原因としては、ローカルコンピュータにおいて、認証情報管理をサポートしていないオペレーティングシステムを実行していること、または認証情報マネジメントユーティリティがインストールされていないこと、またはこれらの認証情報管理システムのいずれかに IAM ユーザーの Git 認証情報が保存されていないことが考えられます。オペレーティングシステムおよびローカル環境によっては、認証情報マネージャーのインストール、オペレーティングシステム内の認証情報マネージャーの設定、またはローカル環境のカスタマイズを行い、認証情報ストレージを使用する必要がある場合があります。例えば、コンピュータで macOS が実行されている場合は、Keychain Access ユーティリティを使用して認証情報を保存できます。コンピュータで Windows が実行されている場合は、Git for Windows と一緒にインストールされている Git 認証情報マネージャーを使用できます。詳細については、Git 認証情報を使用した HTTPS ユーザーのセットアップ、および Git ドキュメントの認証情報ストレージを参照してください。

Git 認証情報 AWS CodeCommit: Git 認証情報をセットアップしたが、システムが使用していない

問題: Git クライアントで CodeCommit を使用しようとすると、クライアントは IAM ユーザーの Git 認証情報を使用していないように見える。

解決方法: このエラーの最も一般的な原因は、以前に AWS CLIに含まれている認証情報ヘルパーを使用するようにコンピューターをセットアップしたことです。.gitconfig ファイルで、次のような設定セクションを確認し、それらを削除します。

```
[credential "https://git-codecommit.*.amazonaws.com"]
    helper = !aws codecommit credential-helper $@
    UseHttpPath = true
```

ファイルを保存したら、新しいコマンドラインセッションまたはターミナルセッションを開いてから、再度接続を試みます。

また、複数の資格情報ヘルパーまたはマネージャーがコンピューターにセットアップされていて、システムが別の設定にデフォルト設定されている可能性があります。デフォルトとして使用する認証情報ヘルパーをリセットするには、--system コマンドの実行時に --global オプションを使用します (--local や git config は使用しません)。

詳細については、Git 認証情報を使用した HTTPS ユーザーのセットアップ、および Git ドキュメントの認証情報ストレージを参照してください。

git-remote-codecommit と のトラブルシューティング AWS CodeCommit

以下の情報は、git-remote-codecommit リポジトリに接続するときに、 AWS CodeCommit の問題のトラブルシューティングに役立つ場合があります。

トピック

- 次のエラーが表示される: git: 'remote-codecommit' is not a git command
- 次のエラーが表示される: fatal: Unable to find remote helper for 'codecommit'
- クローンエラー: IDE から CodeCommit リポジトリのクローンを作成できない
- プッシュまたはプルエラー: IDE から CodeCommit リポジトリにコミットをプッシュまたはプルできない
- インストールエラー: git-remote-codecommitexternally-managed-environmentエラーが表示されます

次のエラーが表示される: git: 'remote-codecommit' is not a git command

問題: git-remote-codecmit を使用しようとすると、git-remote-codecmit は git コマンドではないというエラーが表示されます。「git --help」を参照してください。

解決方法： このエラーの最も一般的な理由は、git-remote-codecommit 実行可能ファイルを PATH に追加していないか、文字列に構文エラーが含まれていることです。このエラーは、git と remote-codecmit の間にハイフンがない場合や、git-remote-codecmit の前に余分な git が挿入されている場合に発生します。

また、ローカルインストールをサポートされていないバージョンに更新した場合も、このエラーが発生することがあります。サポートされている Python のバージョンの詳細については、git-remote-codecommit」を参照してください。

git-remote-codecmit の設定と使用の詳細については、「git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」を参照してください。

次のエラーが表示される: fatal: Unable to find remote helper for 'codecommit'

問題: git-remote-codecmit を使用しようとすると、「致命的: 'codecmit'のリモートヘルパーが見つかりません」というエラーが表示されます。

解決方法: このエラーの最も一般的な原因は次のとおりです。

- git-remote-codecommit の設定が完了していない
- git-remote-codecommit をパスにないロケーションにインストールしたか、`Path` 環境変数の一部として設定されていない
- Python がパスにないか、`Path` 環境変数の一部として設定されていない
- git-remote-codecommit のインストールが完了してから再起動していないターミナルまたはコマンドラインウィンドウを使用している

git-remote-codecmit の設定と使用の詳細については、「git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」を参照してください。

クローンエラー: IDE から CodeCommit リポジトリのクローンを作成できない

問題: IDE で CodeCommit リポジトリのクローンを作成しようとすると、エンドポイントまたは URL が無効であることを示すエラーが表示される。

考えられる修正: すべての IDE が、クローン作成中に git-remote-codecommit によって使用される URL をサポートしているわけではありません。端末またはコマンドラインからリポジトリのクローンをローカルに作成し、そのローカルリポジトリを IDE に追加します。詳細については、「ステップ 3: CodeCommit コンソールに接続し、リポジトリのクローンを作成する」を参照してください。

プッシュまたはプルエラー: IDE から CodeCommit リポジトリにコミットをプッシュまたはプルできない

問題: IDE からコードをプルまたはプッシュしようとすると、接続エラーが表示される。

考えられる修正: このエラーの最も一般的な理由は、IDE が git-remote-codecommit などの Git リモートヘルパーと互換性がないことです。IDE 機能を使用してコードをコミット、プッシュ、プルする代わりに、Git コマンドを使用してコマンドラインまたはターミナルからローカルリポジトリを手動で更新します。

リモートヘルパーと Git の詳細については、Git のドキュメントを参照してください。

インストールエラー: git-remote-codecommitexternally-managed-environmentエラーが表示されます

問題： `pip install git-remote-codecommit` コマンドを実行しようとすると、環境が外部で管理されていることを示すエラーが表示されます。

解決方法： このエラーの最も一般的な理由は、EXTERNALLY-MANAGED マーカーファイルを定義する Python の配布 (ディストリビューション) を実行していることです。そのための最善の解決策は、仮想環境を作成して使用することです。

外部管理環境と Python の詳細については、Python ドキュメントの「外部管理環境」および「pip と venv を使用して仮想環境にパッケージをインストールする」を参照してください。

への SSH 接続のトラブルシューティング AWS CodeCommit

以下の情報は、SSH を使用して CodeCommit リポジトリに接続する際に発生する一般的な問題のトラブルシューティングに役立つ場合があります。

トピック

- アクセスエラー: パブリックキーが IAM に正常にアップロードされたが、Linux、macOS、または Unix システムでは接続が失敗する。
- アクセスエラー: パブリックキーは正常に IAM および SSH にアップロードされたが、Windows システム上で接続できない。
- 認証の問題: CodeCommit リポジトリへの接続時にホストの信頼性が確立できない
- IAM エラー: 「無効な形式」(IAM へのパブリックキーの追加時)
- SSH 認証情報を使用して複数の Amazon Web Services アカウントの CodeCommit リポジトリにアクセスする必要がある
- Windows 上の Git: SSH を使用して接続を試みると、Bash エミュレーターまたはコマンドラインはフリーズします。

- パブリックキー形式は、Linux の一部のディストリビューションにおいて指定する必要があります。
- アクセスエラー: CodeCommit リポジトリへの接続時に SSH パブリックキーが拒否される

アクセスエラー: パブリックキーが IAM に正常にアップロードされたが、Linux、macOS、または Unix システムでは接続が失敗する。

問題: 接続をテストする場合、またはリポジトリのクローンを作成する場合に CodeCommit リポジトリと通信するために SSH エンドポイントに接続しようとすると、接続に失敗するか、拒否される。

解決方法: IAM のパブリックキーに割り当てられた SSH キー ID が接続試行時に関連付けられていない可能性があります。設定ファイルが構成されていない、設定ファイルへのアクセス許可が付与されていない、別の設定が原因で設定ファイルが読み取れない、キー ID ではなく IAM ユーザーの ID を入力している、のいずれかが考えられます。

SSH キー ID は、IAM ユーザーのプロフィールの IAM コンソールで確認できます。

> **Note**
>
> 複数の SSH キー ID をアップロードしている場合、キーは、アップロードの日付ではなく、キー ID のアルファベット順にリストされます。正しいアップロードの日付に関連付けられたキー ID をコピーしていることを確認します。

次のコマンドを使用して、接続をお試しください。

```
ssh Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com
```

接続確認後に成功メッセージが表示される場合、お使いの SSH キー ID は有効です。設定ファイルを編集し、接続試行を IAM のパブリックキーと関連付けます。設定ファイルを編集しない場合は、すべてのリポジトリへの接続試行の前で SSH キー ID を使用することができます。たとえば、接続試行を関連付けるための設定ファイルを変更せずに、リポジトリ (*MyDemoRepo*) のクローンを作成する場合は、次のコマンドを実行します。

```
git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo my-demo-repo
```

詳細については、「Linux、macOS、または Unix での SSH 接続の場合」を参照してください。

アクセスエラー: パブリックキーは正常に IAM および SSH にアップロードされたが、Windows システム上で接続できない。

問題: SSH エンドポイントを使用して、クローンを作成するか、CodeCommit リポジトリと接続しようとすると、エラーメッセージ `No supported authentication methods available` が表示される。

解決方法: このエラーの最も一般的な原因は、SSH 使用時に Windows に別のプログラムを使用すると指示する Windows システムの環境変数セットが含まれることです。たとえば、ツール (plink.exe) の PuTTY セットのいずれかにポイントするように GIT_SSH 変数を設定しているとします。これはレガシーな設定であるか、コンピュータにインストールされている他のプログラムに必須である可能性があります。この環境変数が必須ではないことがわかっている場合は、システムプロパティを開き、この環境変数を削除できます。

この問題を対処するには、Bash エミュレーターを開いて 再度 SSH 接続を試みますが、プリフィックスとして `GIT_SSH_COMMAND="SSH"` を含みます。たとえば、SSH を使用してリポジトリのクローンを作成するには次のように行います。

```
GIT_SSH_COMMAND="ssh" git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo my-demo-repo
```

お使いの Windows バージョンによって、Windows コマンドラインで SSH から接続しているときに接続文字列の一部として SSH キー ID を含める必要がある場合にも、同様のエラーが発生する場合があります。再度接続をお試しください。今回は、コマンドの一部として、IAM からコピーした SSH キー ID を含めます。例:

```
git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo my-demo-repo
```

認証の問題: CodeCommit リポジトリへの接続時にホストの信頼性が確立できない

問題: SSH エンドポイントを使用して、CodeCommit リポジトリと接続しようとすると、警告メッセージ `The authenticity of host 'host-name' can't be established.` が表示される

解決方法: 認証情報が正しく設定されていない可能性があります。「Linux、macOS、または Unix での SSH 接続の場合」または「Windows で SSH 接続をセットアップする手順」の手順に従います。

このステップ通りに行っても問題が解決しない場合は、中間者 (MITM) 攻撃を受けている可能性があります。次のメッセージが表示されたら、no と入力し、[Enter] キーを押します。

```
Are you sure you want to continue connecting (yes/no)?
```

接続を継続する前に、CodeCommit 接続に必要なフィンガープリントやパブリックキーが、SSH セットアップのトピックのものと一致していることを確認します。

CodeCommit のパブリックフィンガープリント

サーバー	暗号ハッシュタイプ	フィンガープリント
git-codecommit.us-east-2.amazonaws.com	MD5	a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e
git-codecommit.us-east-2.amazonaws.com	SHA256	3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ
git-codecommit.us-east-1.amazonaws.com	MD5	a6:9c:7d:bc:35:f5:d4:5f:8b:ba:6f:c8:bc:d4:83:84
git-codecommit.us-east-1.amazonaws.com	SHA256	eLMY1j0DKA4uvDZcl/KgtIayZANwX6t8+8isPtotBoY

サーバー	暗号ハッシュタイプ	フィンガープリント
git-codecommit.us-west-2.amazonaws.com	MD5	a8:68:53:e3:99:ac:6e:d7:04:7e:f7:92:95:77:a9:77
git-codecommit.us-west-2.amazonaws.com	SHA256	0pJx9SQpkbPUAHwy58UVIq0IHcyo1fwCpOOuVgcAWPo
git-codecommit.eu-west-1.amazonaws.com	MD5	93:42:36:ea:22:1f:f1:0f:20:02:4a:79:ff:ea:12:1d
git-codecommit.eu-west-1.amazonaws.com	SHA256	tKjRkOL8dmJyTmSbeSdN1S8F/f0iql3RlvqgTOP1UyQ
git-codecommit.ap-northeast-1.amazonaws.com	MD5	8e:a3:f0:80:98:48:1c:5c:6f:59:db:a7:8f:6e:c6:cb
git-codecommit.ap-northeast-1.amazonaws.com	SHA256	Xk/WeYD/K/bnBybzhiuu4dWpBJtXPf7E30jHU7se4Ow
git-codecommit.ap-southeast-1.amazonaws.com	MD5	65:e5:27:c3:09:68:0d:8e:b7:6d:94:25:80:3e:93:cf
git-codecommit.ap-southeast-1.amazonaws.com	SHA256	ZIsVa7OVzxrTIf+Rk4UbhPv6Es22mSB3uTBojfPXIno
git-codecommit.ap-southeast-2.amazonaws.com	MD5	7b:d2:c1:24:e6:91:a5:7b:fa:c1:0c:35:95:87:da:a0

サーバー	暗号ハッシュタイプ	フィンガープリント
git-codecommit.ap-southeast-2.amazonaws.com	SHA256	`nYp+gHas80HY3DqbP4yanCDFhqDVjseefVbHEXqH2Ec`
git-codecommit.ap-southeast-3.amazonaws.com	MD5	`64:d9:e0:53:19:4f:a8:91:9a:c3:53:22:a6:a8:ed:a6`
git-codecommit.ap-southeast-3.amazonaws.com	SHA256	`ATdkGSFhpqIu7RqUVT/1RZo6MLxxxUW9NoDVMbAc/6g`
git-codecommit.me-central-1.amazonaws.com	MD5	`bd:fa:e2:f9:05:84:d6:39:6f:bc:d6:8d:fe:de:61:76`
git-codecommit.me-central-1.amazonaws.com	SHA256	`grceUDWubo4MzG1NoaKZKUfrgPvfN3ijliOnQr1lTZA`
git-codecommit.eu-central-1.amazonaws.com	MD5	`74:5a:e8:02:fc:b2:9c:06:10:b4:78:84:65:94:22:2d`
git-codecommit.eu-central-1.amazonaws.com	SHA256	`MwGrkiEki8QkkBtlAgXbYt0hoZYBnZF62VY5RzGJEUY`
git-codecommit.ap-northeast-2.amazonaws.com	MD5	`9f:68:48:9b:5f:fc:96:69:39:45:58:87:95:b3:69:ed`
git-codecommit.ap-northeast-2.amazonaws.com	SHA256	`eegAPQrWY9YsYo9ZHIKOmxetfXBHzAZd8Eya53Qcwko`

サーバー	暗号ハッシュタイプ	フィンガープリント
git-codecommit.sa-east-1.amazonaws.com	MD5	`74:99:9d:ff:2b:ef:63:c6:4b:b4:6a:7f:62:c5:4b:51`
git-codecommit.sa-east-1.amazonaws.com	SHA256	`kW+VKB0jpRaG/ZbXkgbtMQbKgEDK7JnISV3SVoyCmzU`
git-codecommit.us-west-1.amazonaws.com	MD5	`3b:76:18:83:13:2c:f8:eb:e9:a3:d0:51:10:32:e7:d1`
git-codecommit.us-west-1.amazonaws.com	SHA256	`gzauWTWXDK2u5KuMMi5vbKTmfyerdIwgSbzYBODLpzg`
git-codecommit.eu-west-2.amazonaws.com	MD5	`a5:65:a6:b1:84:02:b1:95:43:f9:0e:de:dd:ed:61:d3`
git-codecommit.eu-west-2.amazonaws.com	SHA256	`r0Rwz5k/IHp/QyrRnfiM9j02D5UEqMbtFNTuDG2hNbs`
git-codecommit.ap-south-1.amazonaws.com	MD5	`da:41:1e:07:3b:9e:76:a0:c5:1e:64:88:03:69:86:21`
git-codecommit.ap-south-1.amazonaws.com	SHA256	`hUKwnTj7+Xpx4Kddb6p45j4RazIJ4IhAMD8k29itOfE`
git-codecommit.ap-south-2.amazonaws.com	MD5	`bc:cc:9f:15:f8:f3:58:a2:68:65:21:e2:23:71:8d:ce`

サーバー	暗号ハッシュタイプ	フィンガープリント
git-codecommit.ap-south-2.amazonaws.com	SHA256	`Xe0CyZEOvgR5Xa2YUGqf+jn8/Ut7l7nX/CmslSFNEig`
git-codecommit.ca-central-1.amazonaws.com	MD5	`9f:7c:a2:2f:8c:b5:74:fd:ab:b7:e1:fd:af:46:ed:23`
git-codecommit.ca-central-1.amazonaws.com	SHA256	`Qz5puafQdANVprLlj6r0Qyh4lCNsF6ob61dGcPtFS7w`
git-codecommit.eu-west-3.amazonaws.com	MD5	`1b:7f:97:dd:d7:76:8a:32:2c:bd:2c:7b:33:74:6a:76`
git-codecommit.eu-west-3.amazonaws.com	SHA256	`uw7c2FL564jVoFgtc+ikzILnKBsZz7t9+CFdSJjKbLI`
git-codecommit.us-gov-west-1.amazonaws.com	MD5	`9f:6c:19:3b:88:cd:e8:88:1b:9c:98:6a:95:31:8a:69`
git-codecommit.us-gov-west-1.amazonaws.com	SHA256	`djXQoSIFcg8vHe0KVH1xW/gOF9X37tWTqu4Hkng75x4`
git-codecommit.us-gov-east-1.amazonaws.com	MD5	`00:8d:b5:55:6f:05:78:05:ed:ea:cb:3f:e6:f0:62:f2`
git-codecommit.us-gov-east-1.amazonaws.com	SHA256	`fVb+R0z7qW7minenW+rUpAABRCRBTCzmETAJEQrg98`

サーバー	暗号ハッシュタイプ	フィンガープリント
git-codecommit.eu-north-1.amazonaws.com	MD5	8e:53:d8:59:35:88:82:fd:73:4b:60:8a:50:70:38:f4
git-codecommit.eu-north-1.amazonaws.com	SHA256	b6KSK7xKq+V8jl7iuAcjqXsG7zkqoUZZmmhYYFBq1wQ
git-codecommit.me-south-1.amazonaws.com	MD5	0e:39:28:56:d5:41:e6:8d:fa:81:45:37:fb:f3:cd:f7
git-codecommit.me-south-1.amazonaws.com	SHA256	O+NToCGgjrHekiBuOl0ad7ROGEsz+DBLXOd/c9wc0JU
git-codecommit.ap-east-1.amazonaws.com	MD5	a8:00:3d:24:52:9d:61:0e:f6:e3:88:c8:96:01:1c:fe
git-codecommit.ap-east-1.amazonaws.com	SHA256	LafadYwUYW8hONoTRpojbjNs9IRnbEwHtezD3aAIBX0
git-codecommit.cn-north-1.amazonaws.com.cn	MD5	11:7e:2d:74:9e:3b:94:a2:69:14:75:6f:5e:22:3b:b3
git-codecommit.cn-north-1.amazonaws.com.cn	SHA256	IYUXxH2OpTDsyYMLIp+JY8CTLS4UX+ZC5JVZXPRaxc8
git-codecommit.cn-northwest-1.amazonaws.com.cn	MD5	2e:a7:fb:4c:33:ac:6c:f9:aa:f2:bc:fb:0a:7b:1e:b6

サーバー	暗号ハッシュタイプ	フィンガープリント
git-codecommit.cn-northwest-1.amazonaws.com.cn	SHA256	wqjd6eHd0+mOBx+dCNuL0omUoCNjaDtZiEpWj5TmCfQ
git-codecommit.eu-south-1.amazonaws.com	MD5	b9:f6:5d:e2:48:92:3f:a9:37:1e:c4:d0:32:0e:fb:11
git-codecommit.eu-south-1.amazonaws.com	SHA256	lyXrWbCg3uQmJrl1XxB/ASR7ugW1Ysf5yzYOJbudHsI
git-codecommit.ap-northeast-3.amazonaws.com	MD5	25:17:40:da:b9:d4:18:c3:b6:b3:fb:ed:1c:20:fe:29
git-codecommit.ap-northeast-3.amazonaws.com	SHA256	2B815B9F0AvwLnRxSVxUz4kDYmtEQUGGdQYP8OQLXhA
git-codecommit.af-south-1.amazonaws.com	MD5	21:a0:ba:d7:c1:d1:b5:39:98:8d:4d:7c:96:f5:ca:29
git-codecommit.af-south-1.amazonaws.com	SHA256	C34ji3x/cnsDZjUpyNGXdE5pjHYimqJrQZ3leTgqJHM
git-codecommit.il-central-1.amazonaws.com	MD5	04:74:89:16:98:7a:61:b1:69:46:42:3c:d1:b4:ac:a9
git-codecommit.il-central-1.amazonaws.com	SHA256	uFxhp51kUWhleTLeYbxQVYm4RnNLNZ5Dbdm1cgdSl/8

IAM エラー: 「無効な形式」(IAM へのパブリックキーの追加時)

問題: IAM において、CodeCommit で SSH を使用するようにセットアップを試みると、パブリックキー追加時にエラーメッセージ `Invalid format` が表示される。

解決方法: IAM では、パブリックキーを ssh-rsa 形式または PEM 形式でエンコードする必要があります。OpenSSH 形式のパブリックキーのみを受け入れ、さらに「IAM ユーザーガイド」の「CodeCommit で SSH キーを使用する」で説明されている追加の要件があります。別の形式でパブリックキーを指定した場合やキーが必要なビット数ではない場合にこのエラーが表示されます。

- SSH パブリックキーをコピーすると、オペレーティングシステムによって改行が挿入される場合があります。IAM に追加するパブリックキーに改行がないことを確認してください。
- 一部の Windows オペレーティングシステムでは、OpenSSH 形式を使用しません。キーペアを生成して、IAM に必要な OpenSSH 形式をコピーする方法については、the section called “ステップ 3: Git および CodeCommit 用のパブリックキーとプライベートキーをセットアップする” を参照してください。

IAM における SSH キーの要件の詳細については、IAM ユーザーガイドの CodeCommit で SSH キーを使用するを参照してください。

SSH 認証情報を使用して複数の Amazon Web Services アカウントの CodeCommit リポジトリにアクセスする必要がある

問題: 複数のアマゾン ウェブ サービスアカウントで CodeCommit リポジトリへの SSH アクセスを設定する必要がある。

解決方法: アマゾン ウェブ サービスアカウントごとに一意の SSH パブリック/プライベートキーペアを作成し、各キーを使用して IAM を設定できます。次に、パブリックキーに関連付けられた各 IAM ユーザー ID に関する情報を使用して ~/.ssh/config ファイルを設定できます。例:

```
Host codecommit-1
    Hostname git-codecommit.us-east-1.amazonaws.com
    User SSH-KEY-ID-1 # This is the SSH Key ID you copied from IAM in Amazon Web
 Services account 1 (for example, APKAEIBAERJR2EXAMPLE1).
    IdentityFile ~/.ssh/codecommit_rsa # This is the path to the associated public key
 file, such as id_rsa.  We advise creating CodeCommit specific _rsa files.

Host codecommit-2
```

```
    Hostname git-codecommit.us-east-1.amazonaws.com
    User SSH-KEY-ID-2 # This is the SSH Key ID you copied from IAM in Amazon Web
 Services account 2 (for example, APKAEIBAERJR2EXAMPLE2).
    IdentityFile ~/.ssh/codecommit_2_rsa # This is the path to the other associated
 public key file.  We advise creating CodeCommit specific _rsa files.
```

この設定では、'git-codecmit.us-east-1.amazonaws.com' を 'codecmit-2' に置き換えることができます。例えば、2 番目の Amazon Web Services アカウントにリポジトリを複製するには、次のようにします。

```
git clone ssh://codecommit-2/v1/repos/YourRepositoryName
```

リポジトリのリモートを設定するには、git remote add を実行します。例:

```
git remote add origin ssh://codecommit-2/v1/repos/YourRepositoryName
```

その他の例については、こちらのフォーラムの投稿と GitHub でのこちらの投稿を参照してください。

Windows 上の Git: SSH を使用して接続を試みると、Bash エミュレーターまたはコマンドラインはフリーズします。

問題: Windows で SSH アクセスを設定し、コマンドラインやターミナルで接続できることを確認した後で、コマンドプロンプトまたは Bash エミュレーターで git pull、git push、git clone などのコマンドを使用すると、サーバーのホストキーがレジストリにキャッシュされていないことを示すメッセージが表示されたり、キーをキャッシュに保存するための入力画面がフリーズしたりする (y/n/ の戻り値入力が受け付けられない)。

解決方法: このエラーの最も一般的な原因は、お使いの Git 環境が OpenSSH 以外 (おそらく PuTTY) を使用して認証するように設定されていることです。設定によっては、キーのキャッシングに関する問題が発生することで知られています。この問題を解決するには、以下のいずれかをお試しください。

- Bash エミュレーターを開き、`GIT_SSH_COMMAND="ssh"` パラメータを追加してから、Git コマンドを使用します。たとえば、リポジトリにプッシュする場合は、git push ではなく、次のように入力します。

  ```
  GIT_SSH_COMMAND="ssh" git push
  ```

- PuTTY がインストールされている場合は、PuTTY を開き、[Host Name (or IP address)] (ホスト名 (または IP アドレス)) で、到達する CodeCommit エンドポイント (例えば、git-codecommit.us-east-2.amazonaws.com) を入力します。[Open] を選択します。PuTTY セキュリティ警告によりプロンプトが表示されたら、[はい] を選択して、完全にキーをキャッシュします。
- 今後使用しない `GIT_SSH` 環境変数の名前を変更するか、削除します。次に、新しいコマンドプロンプト、または Bash エミュレーターセッションを開き、コマンドを再度実行してみてください。

その他のソリューションについては、スタックのオーバーフローの「キャッシュの保存キーでの Git のクローン/プルのフリーズ」を参照してください。

パブリックキー形式は、Linux の一部のディストリビューションにおいて指定する必要があります。

問題: パブリックキーとプライベートキーのペアを構成しようとすると、エラーが発生する。

解決方法: Linux の一部のディストリビューションでは、`~/.ssh/config` ファイルで、受け入れられるパブリックキーの種類を指定する追加の設定を行う必要があります。詳細については、`PubkeyAcceptedKeyTypes` に関するディストリビューションのドキュメントを参照してください。

アクセスエラー: CodeCommit リポジトリへの接続時に SSH パブリックキーが拒否される

問題: SSH エンドポイントを使用して、CodeCommit リポジトリと接続しようとすると、エラーメッセージ `Error: public key denied` が表示される。

解決方法: このエラーの最も一般的な原因は、SSH 接続のセットアップを完了していないことです。SSH キー (パブリックキーとプライベートのペア) を設定し、続いて、そのパブリックキーを IAM ユーザーと関連付けます。SSH の設定に関する詳細については、「Linux、macOS、または Unix での SSH 接続の場合」および「Windows で SSH 接続をセットアップする手順」を参照してください。

認証情報ヘルパーと への HTTPS 接続のトラブルシューティング AWS CodeCommit

以下の情報は、 AWS CLI および HTTPS に含まれている認証情報ヘルパーを使用して CodeCommit リポジトリに接続する際の一般的な問題のトラブルシューティングに役立つ場合があります。

> **Note**
>
> 認証情報ヘルパーは、フェデレーティッドアクセス、ID プロバイダー、または一時的な認証情報を使用して CodeCommit に接続するためにサポートされている方法ですが、git-remote-codecommit ユーティリティをインストールして使用する方法が推奨されます。詳細については、「git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」を参照してください。

トピック

- git config コマンドを実行して認証情報ヘルパーを設定しようとするとエラーが発生する
- リポジトリのクローンを作成しようとすると、ユーザー名の読み取りエラーが表示される
- 認証情報ヘルパーを使用すると Windows でコマンドが見つかりませんというエラーが表示される
- CodeCommit リポジトリへの接続時にユーザー名の入力を求められる
- Git for macOS: 認証情報ヘルパーは正常に設定できましたが、リポジトリへのアクセスが拒否されます (403)
- Git for Windows: Git for Windows をインストールしましたが、リポジトリへのアクセスが拒否されます (403)

`git config` コマンドを実行して認証情報ヘルパーを設定しようとするとエラーが発生する

問題: Git config コマンドを実行して CodeCommit リポジトリと通信するように認証情報ヘルパーを設定しようとすると、引数が少なすぎるというエラーや、Git config コマンドと構文の使用に関するプロンプトが表示される。

解決方法: このエラーの最も一般的な原因は、Windows オペレーティングシステムでコマンドに一重引用符が使用されている、もしくは Linux、macOS、UNIX オペレーティングシステムでコマンドに二重引用符が使用されていることです。正しい構文は次のとおりです。

- Windows: `git config --global credential.helper "!aws codecommit credential-helper $@"`
- Linux、macOS、または Unix: `git config --global credential.helper '!aws codecommit credential-helper $@'`

リポジトリのクローンを作成しようとすると、ユーザー名の読み取りエラーが表示される

問題： 認証情報ヘルパーを使用してリポジトリのクローンを作成しようとすると、システムがリポジトリ URL のユーザー名を読み取れなかったことを示すエラーが表示されます。

解決方法： このエラーの最も一般的な理由は、.gitconfig ファイルがコンピュータで正しく作成または設定されていないことです。.gitconfig ファイルを開き、認証情報ヘルパーが正しく設定されていることを確認します。Linux、macOS、または Unix を実行しているコンピュータを使用している場合`$HOME` は、 の値がシステムに対して正しく設定されていることを確認してください。

認証情報ヘルパーを使用すると Windows でコマンドが見つかりませんというエラーが表示される

問題： CLI AWS を更新した後、CodeCommit リポジトリへの認証情報ヘルパー接続は で失敗します`aws codecommit credential-helper $@ get: aws: command not found`。

原因: このエラーの最も一般的な理由は、CLI AWS バージョンが Python 3 を使用するバージョンに更新されたことです。MSI パッケージに関する既知の問題があります。該当するバージョンのいずれかを持っているかどうかを確認するには、コマンドラインを開き、次のコマンドを実行します: `aws --version`

出力された Python のバージョンが 3 で始まる場合は、該当するバージョンがあります。例:

```
aws-cli/1.16.62 Python/3.6.2 Darwin/16.7.0 botocore/1.12.52
```

解決方法: この問題を回避するには、次のいずれかを実行します。

- MSI の代わりに Python と pip を使用して Windows AWS CLI に をインストールして設定します。詳細については、「Windows AWS CLI での Python、pip、および のインストール」を参照してください。
- 手動で `.gitconfig` ファイルを編集して、ローカルコンピュータの `[credential]` セクションを明示的に `aws.cmd` を指すように変更します。例:

```
[credential]
    helper = !"\C:\\Program Files\\Amazon\\AWSCLI\\bin\\aws.cmd\" codecommit
 credential-helper $@
    UseHttpPath = true
```

- git config コマンドを実行して `.gitconfig` を明示的に参照するように `aws.cmd` ファイルを更新し、必要に応じてコマンドへのパスを含めるように PATH 環境変数を手動で更新します。例:

```
git config --global credential.helper "!aws.cmd codecommit credential-helper $@"
git config --global credential.UseHttpPath true
```

CodeCommit リポジトリへの接続時にユーザー名の入力を求められる

問題: 認証情報ヘルパーを使用して CodeCommit リポジトリに接続しようとすると、ユーザー名を求めるメッセージが表示される。

解決方法： AWS プロファイルを設定するか、使用しているプロファイルが CodeCommit の使用用に設定したプロファイルであることを確認します。セットアップの詳細については、「AWS CLI 認証情報ヘルパーを使用して Linux、macOS、または Unix 上の AWS CodeCommit リポジトリへの HTTPS 接続を設定する手順 または AWS CLI 認証情報ヘルパーを使用した Windows 上の AWS CodeCommit リポジトリへの HTTPS 接続のセットアップ手順」を参照してください。IAM、アクセスキー、シークレットキーに関する詳細については、IAM ユーザーのアクセスキーの管理および認証情報を取得する方法を参照してください。

Git for macOS: 認証情報ヘルパーは正常に設定できましたが、リポジトリへのアクセスが拒否されます (403)

問題: macOS で、認証情報ヘルパーによって認証情報が正常にアクセスまたは使用されていないように見える。原因として 2 つの問題が考えられます。

- AWS CLI は、リポジトリが存在する AWS リージョン とは異なる 用に設定されています。
- Keychain Access ユーティリティで保存されている認証情報の有効期限が切れています。

解決方法： AWS CLI が正しいリージョンに設定されているかどうかを確認するには、 aws configure コマンドを実行し、表示される情報を確認します。CodeCommit リポジトリが に示されているもの AWS リージョン とは異なる にある場合は AWS CLI、 aws configure コマンドを実行し、そのリージョンに適した値に変更する必要があります。詳細については、「ステップ 1: CodeCommit の初期設定」を参照してください。

OS X および macOS でリリースされているデフォルトバージョンの Git では、Keychain Access ユーティリティを使用して、生成された認証情報を保存します。セキュリティ上の理由により、CodeCommit リポジトリへのアクセス用に生成されるパスワードは一時的なものであり、約 15

分後にキーチェーンに保存されている認証情報は機能しなくなります。Git に CodeCommit でのみアクセスする場合は、以下のことをお試しください。

1. ターミナルで git config コマンドを実行し、Keychain Access ユーティリティが定義されている Git 設定ファイル (`gitconfig`) を見つけます。ローカルシステムおよび設定によっては、複数の `gitconfig` ファイルが存在する場合があります。

   ```
   git config -l --show-origin | grep credential
   ```

   このコマンドの出力で、次のような結果を検索します。

   ```
   file:/path/to/gitconfig  credential.helper=osxkeychain
   ```

   この行の先頭に示されているファイルが、編集する必要がある Git 設定ファイルです。

2. Git 設定ファイルを編集するには、プレーンテキストエディタを使用するか、次のコマンドを実行します。

   ```
   nano /usr/local/git/etc/gitconfig
   ```

3. 次のいずれかの方法を使用して、設定を変更します。

   - `helper = osxkeychain` が含まれている認証情報セクションをコメントアウトまたは削除します。例:

     ```
     # helper = osxkeychain
     ```

   - `aws credential helper` と `osxkeychain` の両方の認証情報ヘルパーセクション更新して内容を持たせます。例えば、`osxkeychain` を使用して GitHub に認証する場合:

     ```
     [credential "https://git-codecommit.us-east-1.amazonaws\.com"]
       helper = !aws --profile CodeCommitProfile codecommit credential-helper $@
       UseHttpPath = true
     [credential "https://github.com"]
       helper = osxkeychain
     ```

     この設定の場合、Git は、リモートホストが "`osxkeychain`" に一致したときに `https://github.com` ヘルパーを使用し、リモートホストが "`https://git-codecommit\.us-east-1\.amazonaws.com`" に一致したときに認証情報ヘルパーを使用します。

- 認証情報ヘルパーの前に空の文字列ヘルパーを含めます。例えば、*CodeCommitProfile* という名前のプロファイルを CLI で使用するときにosxkeychainヘルパーを使用しない場合：

```
[credential]
  helper =
  helper = !aws --profile CodeCommitProfile codecommit credential-helper $@
  UseHttpPath = true
```

> **Tip**
>
> 空の文字列ヘルパー行の後に、すべてのプロファイルを除外する場合に CodeCommit と一致しないように行を設定することもできます。
>
> ```
> helper = !aws codecommit credential-helper $@
> ```

または、Keychain Access ユーティリティを引き続き使用して他の Git リポジトリの認証情報をキャッシュする場合は、行をコメントアウトする代わりにヘッダーを変更します。たとえば、GitHub の認証情報のキャッシュを許可するには、次のようにヘッダーを変更できます。

```
[credential "https://github.com"]
   helper = osxkeychain
```

Git を使用して他のリポジトリにアクセスする場合は、CodeCommit リポジトリの認証情報が提供されないように、Keychain Access ユーティリティを設定できます。Keychain Access ユーティリティを設定するには、以下のように行います。

1. Keychain Access ユーティリティを開きます。(Finder を使用して位置を指定できます)
2. *us-east-2* を検索`git-codecommit.`*us-east-2*`.amazonaws.com`し、リポジトリが存在する AWS リージョン に置き換えます。行をハイライト表示し、コンテキスト (右クリック) メニューを開いてから、[Get Info] を選択します。
3. [Access Control] タブを選択します。
4. [Confirm before allowing access] で、[`git-credential-osxkeychain`] を選択し、マイナス記号を選択してリストから削除します。

> **Note**
>
> リストから `git-credential-osxkeychain` を削除すると、Git コマンドを実行するたびにダイアログボックスが表示されます。[Deny] を選択して続行します。ポップアップが不要な場合は、以下のように行います。
>
> - HTTPS で認証情報ヘルパーの代わりに SSH または Git 認証情報を使用して CodeCommit に接続する 詳細については、Linux、macOS、または Unix での SSH 接続の場合および Git 認証情報を使用した HTTPS ユーザーのセットアップを参照してください。
> - Keychain Access ユーティリティで、`git-codecommit.us-east-2.amazonaws.com` の [Access Control] (アクセスコントロール) タブから、[Allow all applications to access this item (access to this item is not restricted)] (すべてのアプリケーションがこのアイテムにアクセスすることを許可 (このアイテムへのアクセスは制限されない)) オプションを選択します。これにより、ポップアップは表示されませんが、認証情報はやがて無効になり (およそ 15 分後)、403 エラーメッセージが表示されます。この場合、キーチェーンアイテムを削除して機能を復元する必要があります。
> - デフォルトでキーチェーンを使用しない Git バージョンをインストールします。
> - キーチェーンアイテムを削除するスクリプトソリューションを検討してください。スクリプトソリューションのコミュニティで作成されたサンプルを表示するには、製品およびサービスの統合 の OS X 認証ストアのキャッシュされた認証情報を定期的に削除する Mac OS X スクリプトを参照してください。

Git で Keychain Access ユーティリティを完全に使用しないように、osxkeychain が Git で認証情報ヘルパーとして使用されないように設定することができます。たとえば、ターミナルを開いて `git config --system credential.helper` コマンドを実行し、`osxkeychain` が返される場合、Git は Keychain Access ユーティリティを使用するように設定されています。これを変更するには、次のコマンドを実行します。

```
git config --system --unset credential.helper
```

`--system` オプションを指定してこのコマンドを実行すると、すべてのユーザーに対して Git の動作が変更されるため、他のユーザーに意図しない結果が生じるか、CodeCommit だけでなく他のリポジトリサービスを使用している場合は、他のリポジトリにも同様の結果が生じることがある点に注意

してください。また、このアプローチでは、sudo の使用が必要になり、この変更を適用するための十分なアクセス許可がお客様のアカウントに付与されていない場合があります。このコマンドが正常に適用されていることを確認するには、`git config --system credential.helper` コマンドを再度実行します。詳細については、「Git のカスタマイズ - Git の設定」および「Stack Overflow m に関するこの記事」を参照してください。

Git for Windows: Git for Windows をインストールしましたが、リポジトリへのアクセスが拒否されます (403)

問題: Windows で、認証情報ヘルパーによって認証情報が正常にアクセスまたは使用されていないようです。原因としてさまざまな問題が考えられます。

- AWS CLI は、リポジトリが存在する AWS リージョン とは異なる 用に設定されています。
- デフォルトでは、Git for Windows は、認証情報ヘルパーを使用する CodeCommit 接続と互換性のない Git AWS 認証情報マネージャーユーティリティをインストールします。インストールすると、認証情報ヘルパーが でインストール AWS CLI され、CodeCommit への接続用に設定されていても、リポジトリへの接続が失敗します。
- Git for Windows の一部のバージョンは、「RFC 2617」および「RFC 4559」との完全な互換性がない場合があります。これにより、Git 認証情報と AWS CLIに含まれている認証情報ヘルパーの両方で問題が生じることがあります。詳細については、「バージョン 2.11.0(3) でユーザー名/パスワードの入力画面が表示されない」を参照してください。

解決方法:

- に含まれている認証情報ヘルパーを使用しようとする場合は AWS CLI、認証情報ヘルパーを使用する代わりに、HTTPS 経由で Git 認証情報を使用して接続することを検討してください。IAM ユーザーに設定されている Git 認証情報は、 の認証情報ヘルパーとは異なり、Windows 用 Git 認証情報マネージャーと互換性があります AWS CodeCommit詳細については、「Git 認証情報を使用した HTTPS ユーザーのセットアップ」を参照してください。

  認証情報ヘルパーを使用する場合は、 が正しく設定 AWS CLI されているかどうかを確認するには AWS リージョン、 aws configure コマンドを実行し、表示される情報を確認します。CodeCommit リポジトリが に示されているものと AWS リージョン 異なる にある場合は AWS CLI、 aws configure コマンドを実行し、そのリージョンに適した値に変更する必要があります。詳細については、「ステップ 1: CodeCommit の初期設定」を参照してください。
- 可能な場合は、Git for Windows をアンインストールしてから再インストールします。Git for Windows をインストールする際、Git 認証情報マネージャーユーティリティをインストールする

オプションのチェックボックスをオフにします。この認証情報マネージャーは、 の認証情報ヘルパーとは互換性がありません AWS CodeCommit Git 認証情報マネージャー、または別の認証情報管理ユーティリティをインストール済みで、アンインストールしない場合は、次のように `.gitconfig` ファイルを変更し、CodeCommit の特定の認証情報管理を追加します。

1. [Control Panel] (コントロールパネル) を開き、[Credential Manager] (認証情報マネージャー) を選択して、CodeCommit の保存された認証情報を削除します。
2. Notepad など、任意のプレーンテキストエディタで `.gitconfig` ファイルを開きます。

> **Note**
>
> 複数の Git プロファイルを操作する場合は、ローカルとグローバルの `.gitconfig` ファイルを両方使用します。必ず、編集するファイルが正しいことを確認します。

3. ここで次のコードを `.gitconfig` ファイルに追加します。

```
[credential "https://git-codecommit.*.amazonaws.com"]
    helper = !aws codecommit credential-helper $@
    UseHttpPath = true
```

4. ファイルを保存したら、新しいコマンドラインセッションを開いてから、再度接続を試みます。

また、CodeCommit リポジトリに接続する AWS CodeCommit ときに の認証情報ヘルパーを使用し、GitHub リポジトリなどの他のホストされたリポジトリに接続するときに別の認証情報管理システムを使用する場合にも、このアプローチを使用できます。

デフォルトとして使用する認証情報ヘルパーをリセットするには、--system コマンドの実行時に --global オプションを使用します (--local や git config は使用しません)。

- Windows コンピュータで Git 認証情報を使用している場合、RFC 非準拠に関する問題を回避するには、Git 認証情報のユーザー名を接続文字列の一部として含めることができます。例えば、この問題を回避するには、米国東部 (オハイオ) リージョンのリポジトリ (*MyDemoRepo*) のクローンを作成します。

```
git clone https://Your-Git-Credential-Username@git-codecommit.us-
east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
```

> **Note**
>
> Git 認証情報のユーザー名に @ 文字が含まれていると、このアプローチは機能しません。必ず、この文字の URL エンコード (URL エスケープまたはパーセントエンコードとも呼ばれる) を行ってから使用します。

Git クライアントと のトラブルシューティング AWS CodeCommit

以下の情報は、Git を AWS CodeCommit リポジトリと共に使用するときに発生する一般的な問題のトラブルシューティングに役立つ場合があります。HTTPS または SSH を使用している場合の Git クライアント関連の問題のトラブルシューティングについては、Git 認証情報 (HTTPS) のトラブルシューティング、SSH 接続のトラブルシューティング、認証情報ヘルパー (HTTPS) のトラブルシューティングを参照してください。

トピック

- Git エラー: error: RPC failed; result=56, HTTP code = 200 fatal: リモートエンドが予期せず切断されました
- Git エラー: リファレンスアップデートコマンドが多すぎる
- Git エラー: Git の一部のバージョンで HTTPS 経由のプッシュが切断される
- Git エラー: 'gnutls_handshake() failed'
- Git エラー: Git で CodeCommit リポジトリを見つけることができない、またはリポジトリへのアクセス許可がない
- Windows 上の Git: 利用できる認証メソッド (パブリックキー) がサポートされていない

Git エラー: error: RPC failed; result=56, HTTP code = 200 fatal: リモートエンドが予期せず切断されました

問題: 大きな変更、多数の変更、大きなリポジトリのいずれかをプッシュすると、ネットワーク問題またはファイアウォール設定が原因で、長時間実行されている HTTPS 接続が切断されることがあります。

解決方法: SSH を使用してプッシュするか、大きなリポジトリに移行する場合は、リポジトリの段階的移行 の手順を実行します。また、個々のファイルのサイズ制限を超えていないことを確認してください。詳細については、「クォータ」を参照してください。

Git エラー: リファレンスアップデートコマンドが多すぎる

問題: 1 回のプッシュあたりの最大リファレンスアップデート数は 4,000 回です。プッシュにリファレンスアップデートが 4,001 回以上が含まれていると、このエラーが表示されます。

解決方法: `git push --all` および `git push --tags` を使用して、ブランチおよびタグのプッシュを個別にお試しください。タグが多数ある場合は、タグを複数のプッシュに分割します。詳細については、「クォータ」を参照してください。

Git エラー: Git の一部のバージョンで HTTPS 経由のプッシュが切断される

問題: curl を 7.41.0 にアップデートする際の問題が原因で、SSPI ベースのダイジェスト認証が失敗することがあります。影響のある Git の既知バージョンのひとつに、1.9.5.msysgit.1 があります。Git for Windows のバージョンによっては、「RFC 2617」および「RFC 4559」との完全な互換性がない場合があります。これにより、Git 認証情報、または AWS CLIに含まれている認証情報ヘルパーを使用する HTTPS 接続で問題が生じることがあります。

解決方法: 既知の問題の可能性についてお使いの Git バージョンを確認するか、前後のバージョンを使用します。mysysgit に関する詳細については、GitHub フォーラムの「HTTPS へのプッシュが切断される」を参照してください。Git for Windows バージョンの問題の詳細については、「バージョン 2.11.0(3) でユーザー名/パスワードの入力画面が表示されない」を参照してください。

Git エラー: 'gnutls_handshake() failed'

問題: Linux で、Git を使用して CodeCommit リポジトリと接続しようとすると、エラーメッセージ `error: gnutls_handshake() failed` が表示される。

解決方法: OpenSSL に対して Git をコンパイルします。アプローチについては、Ask Ubuntu フォーラムの「HTTPS サーバーへの接続時のエラー: gnutls_handshake() failed」を参照してください。

あるいは、HTTPS の代わりに SSH を使用して、CodeCommit リポジトリと接続します。

Git エラー: Git で CodeCommit リポジトリを見つけることができない、またはリポジトリへのアクセス許可がない

問題: 接続文字列の末尾にスラッシュを使用すると、接続試行が失敗する原因になります。

解決方法: 正しいリポジトリ名および接続文字列が入力されていること、末尾にスラッシュがないことを確認します。詳細については、「リポジトリへの接続」を参照してください。

Windows 上の Git: 利用できる認証メソッド (パブリックキー) がサポートされていない

問題: Windows 向けに SSH アクセスを設定後、git pull、git push、または git clone などのコマンドの使用を試みると、アクセス拒否エラーが表示されます。

解決方法: このエラーの最も一般的な原因は、GIT SSH 環境変数がコンピュータに存在しており、PuTTY などの別の接続ユーティリティをサポートするように設定されていることです。この問題を解決するには、以下のいずれかをお試しください。

- Bash エミュレーターを開き、`GIT_SSH_COMMAND="ssh"` パラメータを追加してから、Git コマンドを使用します。例えば、リポジトリのクローンを作成する場合は、git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo ではなく、次を実行します。

```
GIT_SSH_COMMAND="ssh" git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo my-demo-repo
```

- 今後使用しない `GIT_SSH` 環境変数の名前を変更するか、削除します。次に、新しいコマンドプロンプト、または Bash エミュレーターセッションを開き、コマンドを再度実行してみてください。

SSH を使用しているときに Windows で Git の問題をトラブルシューティングする方法の詳細については、「SSH 接続のトラブルシューティング」を参照してください。

アクセスエラーと のトラブルシューティング AWS CodeCommit

以下の情報は、 AWS CodeCommit リポジトリに接続するときに、アクセスエラーのトラブルシューティングに役立つ場合があります。

トピック

- アクセスエラー: Windows から CodeCommit リポジトリへ接続する際、ユーザー名とパスワードの入力を求められる
- アクセスエラー: CodeCommit リポジトリへの接続時にパブリックキーが拒否される
- アクセスエラー: CodeCommit リポジトリへの接続時の「レートが超過しました」または「429」メッセージ

アクセスエラー: Windows から CodeCommit リポジトリへ接続する際、ユーザー名とパスワードの入力を求められる

問題: Git を使用して CodeCommit リポジトリに接続しようとすると、ユーザー名およびパスワードの入力を求めるダイアログボックスが表示される。

解決方法: Windows 用の組み込みの認証情報管理システムである可能性があります。設定により、以下のいずれかを実行します。

- Git 認証情報で HTTPS を使用している場合は、Git の認証情報は、まだシステムに保存されていません。Git の認証情報を入力して続行します。メッセージが再度表示されることはありません。詳細については、「Git 認証情報を使用した HTTPS ユーザーのセットアップ」を参照してください。
- 認証情報ヘルパーで HTTPS を使用している場合 AWS CodeCommit、Windows 認証情報管理システムと互換性がありません。[Cancel] を選択します。

  また、Git for Windows をインストールしたときに Git 認証情報マネージャーをインストールした可能性があります。Git 認証情報マネージャーは、 AWS CLIに含まれる CodeCommit の認証情報ヘルパーとの互換性がありません。Git 認証情報マネージャーのアンインストールを検討してください。CodeCommit の認証情報ヘルパーを使用する代わりに、git-remote-codecommit をインストールして設定することもできます。

  詳細については、「git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」、「 AWS CLI 認証情報ヘルパーを使用した Windows での HTTPS 接続の場合」、および「Git for Windows: Git for Windows をインストールしましたが、リポジトリへのアクセスが拒否されます (403)」を参照してください。

アクセスエラー: CodeCommit リポジトリへの接続時にパブリックキーが拒否される

問題: SSH エンドポイントを使用して、CodeCommit リポジトリと接続しようとすると、エラーメッセージ `Error: public key denied` が表示される。

解決方法: このエラーの最も一般的な原因は、SSH 接続のセットアップを完了していないことです。SSH キー (パブリックキーとプライベートのペア) を設定し、続いて、そのパブリックキーを IAM ユーザーと関連付けます。SSH の設定に関する詳細については、「Linux、macOS、または

Unix での SSH 接続の場合」および「Windows で SSH 接続をセットアップする手順」を参照してください。

アクセスエラー: CodeCommit リポジトリへの接続時の「レートが超過しました」または「429」メッセージ

問題: CodeCommit リポジトリと通信しようとすると、「レートが超過しました」というメッセージまたはエラーコード「429」が表示される。通信は大幅に遅くなるか失敗します。

原因： アプリケーション、、Git クライアント AWS CLI、または のいずれからの CodeCommit へのすべての呼び出しには AWS Management Console、1 秒あたりのリクエストの最大数とアクティブなリクエスト全体が適用されます。どの でも、Amazon Web Services アカウントの最大許容リクエストレートを超えることはできません AWS リージョン。リクエストが最大レートを超えると、エラーが表示され、それ以降の呼び出しは一時的にアマゾン ウェブ サービスアカウントに制限されます。スロットリング期間中は、CodeCommit への接続が遅くなり、失敗する可能性があります。

解決方法: CodeCommit への接続または呼び出しの数を減らすか、リクエストを分散させるためのステップを実行してください。留意が必要なアプローチ:

- リクエスト、特に定期的なポーリングリクエストにジッターを実装する

  CodeCommit を定期的にポーリングしているアプリケーションがあり、このアプリケーションが複数の Amazon EC2 インスタンスで実行されている場合は、異なる Amazon EC2 インスタンスが同じ秒にポーリングしないようにジッター (ランダムな遅延量) を導入します。1 分間にポーリングメカニズムを均等に分散させるには、0 ～ 59 秒の乱数をお勧めします。
- ポーリングではなくイベントベースのアーキテクチャーを使用する

  ポーリングではなく、イベントベースのアーキテクチャーを使用して、イベントが発生したときにのみ呼び出しが行われるようにします。AWS CodeCommit イベントの CloudWatch Events 通知を使用して、ワークフローをトリガーすることを検討してください。
- API と自動 Git アクションのためのエラー再試行とエクスポネンシャルバックオフの実装

  エラーの再試行とエクスポネンシャルバックオフは、呼び出しの頻度を制限するのに役立ちます。各 AWS SDK は自動再試行ロジックおよび指数エクスポネンシャルバックオフアルゴリズムを実装します。自動 Git push と Git pull の場合は、独自の再試行ロジックを実装する必要があります。詳細については、「エラーの再試行とエクスポネンシャルバックオフ AWS」を参照してください。
- AWS サポートセンターで CodeCommit サービスクォータの引き上げをリクエストする

サービスの制限の引き上げを受けるには、エラーの再試行やエクスポネンシャルバックオフ方法の実装など、ここで示されている提案に従っていることを確認する必要があります。リクエストでは、 AWS リージョン、Amazon Web Services アカウント、およびスロットリングの問題の影響を受ける期間も指定する必要があります。

設定エラーと のトラブルシューティング AWS CodeCommit

以下の情報は、 AWS CodeCommit リポジトリに接続するときに発生する可能性のある設定エラーのトラブルシューティングに役立ちます。

トピック

- 設定エラー: macOS で AWS CLI 認証情報を設定できません

設定エラー: macOS で AWS CLI 認証情報を設定できません

問題： を実行して を設定する`aws configure`と AWS CLI、`ConfigParseError`メッセージが表示されます。

解決方法: このエラーの最も一般的な原因は、認証情報ファイルが既に存在していることです。参照先に ~/.aws を指定し、ファイル (`credentials`) を探します。ファイルの名前を変更するか、削除し、再度 aws configure を実行します。

コンソールエラーと のトラブルシューティング AWS CodeCommit

以下の情報は、 AWS CodeCommit リポジトリを使用するときに、コンソールエラーのトラブルシューティングに役立つ場合があります。

トピック

- アクセスエラー: コンソールまたは から CodeCommit リポジトリの暗号化キーアクセスが拒否されました AWS CLI
- 暗号化エラー:リポジトリを復号できない
- コンソールエラー: コンソールから CodeCommit リポジトリのコードを参照できない
- 表示エラー: ファイルまたはファイル間の比較を表示できません

アクセスエラー: コンソールまたは から CodeCommit リポジトリの暗号化キーアクセスが拒否されました AWS CLI

問題： コンソールまたは から CodeCommit にアクセスしようとすると AWS CLI、`EncryptionKeyAccessDeniedException`または というフレーズを含むエラーメッセージが表示されます`User is not authorized for the KMS default key for CodeCommit 'aws/codecommit' in your account`。

解決方法： このエラーの最も一般的な原因は AWS Key Management Service、Amazon Web Services アカウントが CodeCommit に必須である にサブスクライブされていないことです。 AWS KMS コンソールを開き、AWS マネージドキーを選択し、今すぐ開始するを選択します。現在 AWS Key Management Service サービスにサブスクライブしていないというメッセージが表示された場合は、そのページの手順に従ってサブスクライブします。CodeCommit と の詳細については AWS Key Management Service、「」を参照してください[AWS KMS および 暗号化](#)。

暗号化エラー:リポジトリを復号できない

問題： コンソールまたは から CodeCommit リポジトリにアクセスしようとすると AWS CLI、 というフレーズを含むエラーメッセージが表示されます`Repository can't be decrypted`。

解決方法： このエラーの最も一般的な原因は、このリポジトリのデータの暗号化と復号に使用される AWS KMS キーがアクティブではないか、削除が保留中であることです。CodeCommit には、のアクティブキー AWS マネージドキー またはカスタマーマネージドキー AWS Key Management Serviceが必要です。 AWS KMS コンソールを開き、 AWS マネージドキーまたはカスタマーマネージドキーを選択し、リポジトリに使用されるキーがリポジトリが存在する AWS リージョン に存在し、その状態がアクティブであることを確認します。CodeCommit と の詳細については AWS Key Management Service、「」を参照してください[AWS KMS および 暗号化](#)。

> **Important**
>
> リポジトリのデータの暗号化と復号に使用されたキーが完全に削除されたか、その他の理由でアクセスできない場合、そのキーで暗号化されたリポジトリ内のデータにはアクセスできません。

コンソールエラー: コンソールから CodeCommit リポジトリのコードを参照できない

問題: コンソールからリポジトリの内容を参照しようとすると、エラーメッセージが表示され、アクセスが拒否される。

解決方法: このエラーの最も一般的な原因としては、CodeCommit コンソールからコードを参照するために必要な複数のアクセス許可が、アマゾン ウェブ サービスアカウントに適用されている IAM ポリシーによって拒否されているということが考えられます。CodeCommit のアクセス許可と参照の詳細については、AWS CodeCommitの認証とアクセスコントロール を参照してください。

表示エラー: ファイルまたはファイル間の比較を表示できません

問題: CodeCommit コンソールでファイルまたはファイルの 2 つのバージョンの比較を表示しようとすると、ファイルまたは差異が大きすぎて表示できないというエラーが表示されます。

解決方法: このエラーの最も一般的な原因は、ファイルが大きすぎて表示できないか、ファイル内の 1 行の文字制限を超える行が 1 つまたは複数含まれているか、ファイルの 2 つのバージョンの相違が行制限を超えていることが考えられます。詳細については、クォータ を参照してください。ファイルまたはファイルのバージョン間の相違を表示するには、選択した IDE でファイルをローカルに開くか、Git 差分ツールを使用するか、git diff コマンドを実行します。

トリガーと のトラブルシューティング AWS CodeCommit

以下の情報は、 AWS CodeCommitで表示されるトリガーの問題のトラブルシューティングに役立ちます。

トピック

- トリガーエラー: リポジトリのトリガーが、想定されているとおりに実行されない

トリガーエラー: リポジトリのトリガーが、想定されているとおりに実行されない

問題: リポジトリについて設定された 1 つ以上のトリガーが実行されないか、想定どおりに実行されない。

解決方法: トリガーのターゲットが AWS Lambda 関数の場合は、CodeCommit によりアクセスするために必要な関数のリソースポリシーが設定されていることを確認します。詳細については、「例

3: CodeCommit トリガーと AWS Lambda 統合するためのポリシーを作成する」を参照してください。

または、トリガーを編集し、アクションをトリガーするイベントが選択されていること、アクションに対するレスポンスが表示されるブランチがトリガーのブランチに含まれていることを確認します。トリガーの設定を [All repository events] (すべてのリポジトリイベント) および [All branches] (すべてのブランチ) に変更し、トリガーのテストをお試しください。詳細については、「リポジトリのトリガーを編集する」を参照してください。

デバッグを有効にする

問題: デバッグを有効にして、リポジトリの詳細と Git でコマンドが実行される様子を確認したい

解決方法: 以下をお試しください。

1. ローカルマシンのターミナルまたはコマンドプロンプトから、以下のコマンドを実行してから Git コマンドを実行します。

   Linux、macOS、Unix の場合:

   ```
   export GIT_TRACE_PACKET=1
   export GIT_TRACE=1
   export GIT_CURL_VERBOSE=1
   ```

   Windows の場合:

   ```
   set GIT_TRACE_PACKET=1
   set GIT_TRACE=1
   set GIT_CURL_VERBOSE=1
   ```

   > **Note**
   >
   > HTTPS 接続のみにするには、`GIT_CURL_VERBOSE` に設定する方法が便利です。SSH では、`libcurl` ライブラリは使用しません。

2. Git リポジトリの詳細については、最新バージョンの git-sizer をインストールすることをお勧めします。オペレーティングシステムと環境に適したユーティリティをインタールする手順に従います。インストールしたら、コマンドラインまたはターミナルでディレクトリをローカルリポジトリに変更し、次のコマンドを実行します。

```
git-sizer --verbose
```

> **Tip**
>
> コマンドの出力をファイルに保存して、特に時間の経過とともに問題をトラブルシューティングするときに他のユーザーと簡単に共有できるようにすることを検討してください。

AWS CodeCommit リファレンス

以下のリファレンストピックは、CodeCommit、Git AWS リージョン、サービスの制限などを理解するのに役立ちます。

トピック

- のリージョンと Git 接続エンドポイント AWS CodeCommit
- インターフェイス VPC エンドポイント AWS CodeCommit での の使用
- のクォータ AWS CodeCommit
- AWS CodeCommit コマンドラインリファレンス
- 基本的な Git コマンド

のリージョンと Git 接続エンドポイント AWS CodeCommit

各 CodeCommit リポジトリは に関連付けられています AWS リージョン。CodeCommit は、リージョンのエンドポイントを提供してサービスへのリクエストを行います。さらに、CodeCommit は、CodeCommit が利用可能なすべてのリージョンで SSH プロトコルと HTTPS プロトコルの両方に Git 接続エンドポイントを提供します。

このガイドに示すすべての例では、米国東部 (オハイオ) の同じ Git 用エンドポイント URL (`git-codecommit.us-east-2.amazonaws.com`) を使用しています。ただし、Git を使用して接続を設定するときは、CodeCommit リポジトリをホストする AWS リージョン と一致する Git 接続エンドポイントを選択するようにしてください。例えば、米国東部 (バージニア北部) のリポジトリに接続する場合は、エンドポイント URL として `git-codecommit.us-east-1.amazonaws.com` を使用します。これは、API コールにも当てはまります。 AWS CLI または SDKs を使用して CodeCommit リポジトリに接続する場合は、リポジトリに正しいリージョンエンドポイントを使用していることを確認してください。

トピック

- CodeCommit AWS リージョン でサポート
- Git 接続エンドポイント
- CodeCommit のサーバーフィンガープリント

CodeCommit AWS リージョン でサポート

CodeCommit リポジトリを作成して使用できます AWS リージョン。

- 米国東部(オハイオ)
- 米国東部 (バージニア北部)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (パリ)
- 欧州 (フランクフルト)
- 欧州 (ストックホルム)
- 欧州 (ミラノ)
- アフリカ (ケープタウン)
- イスラエル (テルアビブ)
- アジアパシフィック (東京)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (ジャカルタ)
- 中東 (アラブ首長国連邦)
- アジアパシフィック (ソウル)
- アジアパシフィック (大阪)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (ハイデラバード)
- アジアパシフィック (香港)
- 南米 (サンパウロ)
- 中東 (バーレーン)
- カナダ (中部)
- 中国 (北京)
- 中国 (寧夏)

- AWS GovCloud (米国西部)
- AWS GovCloud (米国東部)

CodeCommit は、一部のリージョンで連邦情報処理規格 (FIPS) 出版物 140-2 の政府規格に対するサポートを追加しました。FIPS および 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) 140-2 の概要」を参照してください。FIPS をサポートする Git 接続エンドポイントについては、「Git 接続エンドポイント」を参照してください。

CodeCommit へのリージョンエンドポイント AWS CLI、サービス、API コールの詳細については、AWS CodeCommit 「エンドポイントとクォータ」を参照してください。

Git 接続エンドポイント

CodeCommit リポジトリへの Git 接続を設定するときは、次の URL を使用してください。

の Git 接続エンドポイント AWS CodeCommit

リージョン名	リージョン	エンドポイント URL	プロトコル
米国東部 (オハイオ)	us-east-2	https://git-codecommit.us-east-2.amazonaws.com	HTTPS
米国東部 (オハイオ)	us-east-2	ssh://git-codecommit.us-east-2.amazonaws.com	SSH
米国東部 (オハイオ)	us-east-2	https://git-codecommit-fips.us-east-2.amazonaws.com	HTTPS
米国東部 (バージニア北部)	us-east-1	https://git-codecommit.us-east-1.amazonaws.com	HTTPS
米国東部 (バージニア北部)	us-east-1	ssh://git-codecommit.us-east-1.amazonaws.com	SSH

リージョン名	リージョン	エンドポイント URL	プロトコル
米国東部 (バージニア北部)	us-east-1	https://git-codecommit-fips.us-east-1.amazonaws.com	HTTPS
米国西部 (オレゴン)	us-west-2	https://git-codecommit.us-west-2.amazonaws.com	HTTPS
米国西部 (オレゴン)	us-west-2	ssh://git-codecommit.us-west-2.amazonaws.com	SSH
米国西部 (オレゴン)	us-west-2	https://git-codecommit-fips.us-west-2.amazonaws.com	HTTPS
米国西部 (北カリフォルニア)	us-west-1	https://git-codecommit.us-west-1.amazonaws.com	HTTPS
米国西部 (北カリフォルニア)	us-west-1	ssh://git-codecommit.us-west-1.amazonaws.com	SSH
米国西部 (北カリフォルニア)	us-west-1	https://git-codecommit-fips.us-west-1.amazonaws.com	HTTPS
欧州 (アイルランド)	eu-west-1	https://git-codecommit.eu-west-1.amazonaws.com	HTTPS
欧州 (アイルランド)	eu-west-1	ssh://git-codecommit.eu-west-1.amazonaws.com	SSH

リージョン名	リージョン	エンドポイント URL	プロトコル
アジアパシフィック (東京)	ap-northeast-1	https://git-codecommit.ap-northeast-1.amazonaws.com	HTTPS
アジアパシフィック (東京)	ap-northeast-1	ssh://git-codecommit.ap-northeast-1.amazonaws.com	SSH
アジアパシフィック (シンガポール)	ap-southeast-1	https://git-codecommit.ap-southeast-1.amazonaws.com	HTTPS
アジアパシフィック (シンガポール)	ap-southeast-1	ssh://git-codecommit.ap-southeast-1.amazonaws.com	SSH
アジアパシフィック (シドニー)	ap-southeast-2	https://git-codecommit.ap-southeast-2.amazonaws.com	HTTPS
アジアパシフィック (シドニー)	ap-southeast-2	ssh://git-codecommit.ap-southeast-2.amazonaws.com	SSH
アジアパシフィック (ジャカルタ)	ap-southeast-3	https://git-codecommit.ap-southeast-3.amazonaws.com	HTTPS
アジアパシフィック (ジャカルタ)	ap-southeast-3	ssh://git-codecommit.ap-southeast-3.amazonaws.com	SSH
中東 (アラブ首長国連邦)	me-central-1	https://git-codecommit.me-central-1.amazonaws.com	HTTPS

リージョン名	リージョン	エンドポイント URL	プロトコル
中東 (アラブ首長国連邦)	me-central-1	ssh://git-codecommit.me-central-1.amazonaws.com	SSH
欧州 (フランクフルト)	eu-central-1	https://git-codecommit.eu-central-1.amazonaws.com	HTTPS
欧州 (フランクフルト)	eu-central-1	ssh://git-codecommit.eu-central-1.amazonaws.com	SSH
アジアパシフィック (ソウル)	ap-northeast-2	https://git-codecommit.ap-northeast-2.amazonaws.com	HTTPS
アジアパシフィック (ソウル)	ap-northeast-2	ssh://git-codecommit.ap-northeast-2.amazonaws.com	SSH
南米 (サンパウロ)	sa-east-1	https://git-codecommit.sa-east-1.amazonaws.com	HTTPS
南米 (サンパウロ)	sa-east-1	ssh://git-codecommit.sa-east-1.amazonaws.com	SSH
欧州 (ロンドン)	eu-west-2	https://git-codecommit.eu-west-2.amazonaws.com	HTTPS
欧州 (ロンドン)	eu-west-2	ssh://git-codecommit.eu-west-2.amazonaws.com	SSH

リージョン名	リージョン	エンドポイント URL	プロトコル
アジアパシフィック (ムンバイ)	ap-south-1	https://git-codecommit.ap-south-1.amazonaws.com	HTTPS
アジアパシフィック (ムンバイ)	ap-south-1	ssh://git-codecommit.ap-south-1.amazonaws.com	SSH
アジアパシフィック (ハイデラバード)	ap-south-2	https://git-codecommit.ap-south-2.amazonaws.com	HTTPS
アジアパシフィック (ハイデラバード)	ap-south-2	ssh://git-codecommit.ap-south-2.amazonaws.com	SSH
カナダ (中部)	ca-central-1	https://git-codecommit.ca-central-1.amazonaws.com	HTTPS
カナダ (中部)	ca-central-1	ssh://git-codecommit.ca-central-1.amazonaws.com	SSH
カナダ (中部)	ca-central-1	https://git-codecommit-fips.ca-central-1.amazonaws.com	HTTPS
欧州 (パリ)	eu-west-3	https://git-codecommit.eu-west-3.amazonaws.com	HTTPS
欧州 (パリ)	eu-west-3	ssh://git-codecommit.eu-west-3.amazonaws.com	SSH

リージョン名	リージョン	エンドポイント URL	プロトコル
AWS GovCloud (米国西部)	us-gov-west-1	https://git-codecommit.us-gov-west-1.amazonaws.com	HTTPS
AWS GovCloud (米国西部)	us-gov-west-1	ssh://git-codecommit.us-gov-west-1.amazonaws.com	SSH
AWS GovCloud (米国西部)	us-gov-west-1	https://git-codecommit-fips.us-gov-west-1.amazonaws.com	HTTPS
AWS GovCloud (米国東部)	us-gov-east-1	https://git-codecommit.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (米国東部)	us-gov-east-1	ssh://git-codecommit.us-gov-east-1.amazonaws.com	SSH
AWS GovCloud (米国東部)	us-gov-east-1	https://git-codecommit-fips.us-gov-east-1.amazonaws.com	HTTPS
欧州 (ストックホルム)	eu-north-1	https://git-codecommit.eu-north-1.amazonaws.com	HTTPS
欧州 (ストックホルム)	eu-north-1	ssh://git-codecommit.eu-north-1.amazonaws.com	SSH
中東 (バーレーン)	me-south-1	https://git-codecommit.me-south-1.amazonaws.com	HTTPS

リージョン名	リージョン	エンドポイント URL	プロトコル
中東 (バーレーン)	me-south-1	ssh://git-codecommit.me-south-1.amazonaws.com	SSH
アジアパシフィック (香港)	ap-east-1	https://git-codecommit.ap-east-1.amazonaws.com	HTTPS
アジアパシフィック (香港)	ap-east-1	ssh://git-codecommit.ap-east-1.amazonaws.com	SSH
中国 (北京)	cn-north-1	https://git-codecommit.cn-north-1.amazonaws.com.cn	HTTPS
中国 (北京)	cn-north-1	ssh://git-codecommit.cn-north-1.amazonaws.com.cn	SSH
中国 (寧夏)	cn-northwest-1	https://git-codecommit.cn-northwest-1.amazonaws.com.cn	HTTPS
中国 (寧夏)	cn-northwest-1	ssh://git-codecommit.cn-northwest-1.amazonaws.com.cn	SSH
ヨーロッパ (ミラノ)	eu-south-1	https://git-codecommit.eu-south-1.amazonaws.com	HTTPS
ヨーロッパ (ミラノ)	eu-south-1	ssh://git-codecommit.eu-south-1.amazonaws.com	SSH

リージョン名	リージョン	エンドポイント URL	プロトコル
アジアパシフィック (大阪)	ap-northeast-3	https://git-codecommit.ap-northeast-3.amazonaws.com	HTTPS
アジアパシフィック (大阪)	ap-northeast-3	ssh://git-codecommit.ap-northeast-3.amazonaws.com	SSH
アフリカ (ケープタウン)	af-south-1	https://git-codecommit.af-south-1.amazonaws.com	HTTPS
アフリカ (ケープタウン)	af-south-1	ssh://git-codecommit-af-south-1.amazonaws.com	SSH
イスラエル (テルアビブ)	il-central-1	https://git-codecommit.il-central-1.amazonaws.com	HTTPS
イスラエル (テルアビブ)	il-central-1	ssh://git-codecommit.il-central-1.amazonaws.com	SSH

CodeCommit のサーバーフィンガープリント

以下の表には、CodeCommit の Git 接続エンドポイントのパブリックフィンガープリントが一覧表示されています。これらのサーバーフィンガープリントは、既知のホストファイルにエンドポイントを追加するための検証プロセスの一環として表示されています。

CodeCommit のパブリックフィンガープリント

サーバー	暗号ハッシュタイプ	フィンガープリント
git-codecommit.us-east-2.amazonaws.com	MD5	a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e
git-codecommit.us-east-2.amazonaws.com	SHA256	3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ
git-codecommit.us-east-1.amazonaws.com	MD5	a6:9c:7d:bc:35:f5:d4:5f:8b:ba:6f:c8:bc:d4:83:84
git-codecommit.us-east-1.amazonaws.com	SHA256	eLMY1j0DKA4uvDZcl/KgtIayZANwX6t8+8isPtotBoY
git-codecommit.us-west-2.amazonaws.com	MD5	a8:68:53:e3:99:ac:6e:d7:04:7e:f7:92:95:77:a9:77
git-codecommit.us-west-2.amazonaws.com	SHA256	0pJx9SQpkbPUAHwy58UVIq0IHcyo1fwCpOOuVgcAWPo
git-codecommit.eu-west-1.amazonaws.com	MD5	93:42:36:ea:22:1f:f1:0f:20:02:4a:79:ff:ea:12:1d
git-codecommit.eu-west-1.amazonaws.com	SHA256	tKjRkOL8dmJyTmSbeSdN1S8F/f0iql3RlvqgTOP1UyQ
git-codecommit.ap-northeast-1.amazonaws.com	MD5	8e:a3:f0:80:98:48:1c:5c:6f:59:db:a7:8f:6e:c6:cb

サーバー	暗号ハッシュタイプ	フィンガープリント
git-codecommit.ap-northeast-1.amazonaws.com	SHA256	`Xk/WeYD/K/bnBybzhiuu4dWpBJtXPf7E30jHU7se4Ow`
git-codecommit.ap-southeast-1.amazonaws.com	MD5	`65:e5:27:c3:09:68:0d:8e:b7:6d:94:25:80:3e:93:cf`
git-codecommit.ap-southeast-1.amazonaws.com	SHA256	`ZIsVa7OVzxrTIf+Rk4UbhPv6Es22mSB3uTBojfPXIno`
git-codecommit.ap-southeast-2.amazonaws.com	MD5	`7b:d2:c1:24:e6:91:a5:7b:fa:c1:0c:35:95:87:da:a0`
git-codecommit.ap-southeast-2.amazonaws.com	SHA256	`nYp+gHas80HY3DqbP4yanCDFhqDVjseefVbHEXqH2Ec`
git-codecommit.ap-southeast-3.amazonaws.com	MD5	`64:d9:e0:53:19:4f:a8:91:9a:c3:53:22:a6:a8:ed:a6`
git-codecommit.ap-southeast-3.amazonaws.com	SHA256	`ATdkGSFhpqIu7RqUVT/1RZo6MLxxxUW9NoDVMbAc/6g`
git-codecommit.me-central-1.amazonaws.com	MD5	`bd:fa:e2:f9:05:84:d6:39:6f:bc:d6:8d:fe:de:61:76`
git-codecommit.me-central-1.amazonaws.com	SHA256	`grceUDWubo4MzG1NoaKZKUfrgPvfN3ijliOnQr1lTZA`

サーバー	暗号ハッシュタイプ	フィンガープリント
git-codecommit.eu-central-1.amazonaws.com	MD5	`74:5a:e8:02:fc:b2:9c:06:10:b4:78:84:65:94:22:2d`
git-codecommit.eu-central-1.amazonaws.com	SHA256	`MwGrkiEki8QkkBtlAgXbYt0hoZYBnZF62VY5RzGJEUY`
git-codecommit.ap-northeast-2.amazonaws.com	MD5	`9f:68:48:9b:5f:fc:96:69:39:45:58:87:95:b3:69:ed`
git-codecommit.ap-northeast-2.amazonaws.com	SHA256	`eegAPQrWY9YsYo9ZHIKOmxetfXBHzAZd8Eya53Qcwko`
git-codecommit.sa-east-1.amazonaws.com	MD5	`74:99:9d:ff:2b:ef:63:c6:4b:b4:6a:7f:62:c5:4b:51`
git-codecommit.sa-east-1.amazonaws.com	SHA256	`kW+VKB0jpRaG/ZbXkgbtMQbKgEDK7JnISV3SVoyCmzU`
git-codecommit.us-west-1.amazonaws.com	MD5	`3b:76:18:83:13:2c:f8:eb:e9:a3:d0:51:10:32:e7:d1`
git-codecommit.us-west-1.amazonaws.com	SHA256	`gzauWTWXDK2u5KuMMi5vbKTmfyerdIwgSbzYBODLpzg`
git-codecommit.eu-west-2.amazonaws.com	MD5	`a5:65:a6:b1:84:02:b1:95:43:f9:0e:de:dd:ed:61:d3`

サーバー	暗号ハッシュタイプ	フィンガープリント
git-codecommit.eu-west-2.amazonaws.com	SHA256	`r0Rwz5k/IHp/QyrRnfiM9j02D5UEqMbtFNTuDG2hNbs`
git-codecommit.ap-south-1.amazonaws.com	MD5	`da:41:1e:07:3b:9e:76:a0:c5:1e:64:88:03:69:86:21`
git-codecommit.ap-south-1.amazonaws.com	SHA256	`hUKwnTj7+Xpx4Kddb6p45j4RazIJ4IhAMD8k29itOfE`
git-codecommit.ap-south-2.amazonaws.com	MD5	`bc:cc:9f:15:f8:f3:58:a2:68:65:21:e2:23:71:8d:ce`
git-codecommit.ap-south-2.amazonaws.com	SHA256	`Xe0CyZEOvgR5Xa2YUGqf+jn8/Ut7l7nX/CmslSFNEig`
git-codecommit.ca-central-1.amazonaws.com	MD5	`9f:7c:a2:2f:8c:b5:74:fd:ab:b7:e1:fd:af:46:ed:23`
git-codecommit.ca-central-1.amazonaws.com	SHA256	`Qz5puafQdANVprLlj6r0Qyh4lCNsF6ob61dGcPtFS7w`
git-codecommit.eu-west-3.amazonaws.com	MD5	`1b:7f:97:dd:d7:76:8a:32:2c:bd:2c:7b:33:74:6a:76`
git-codecommit.eu-west-3.amazonaws.com	SHA256	`uw7c2FL564jVoFgtc+ikzILnKBsZz7t9+CFdSJjKbLI`

サーバー	暗号ハッシュタイプ	フィンガープリント
git-codecommit.us-gov-west-1.amazonaws.com	MD5	`9f:6c:19:3b:88:cd:e8:88:1b:9c:98:6a:95:31:8a:69`
git-codecommit.us-gov-west-1.amazonaws.com	SHA256	`djXQoSIFcg8vHe0KVH1xW/gOF9X37tWTqu4Hkng75x4`
git-codecommit.us-gov-east-1.amazonaws.com	MD5	`00:8d:b5:55:6f:05:78:05:ed:ea:cb:3f:e6:f0:62:f2`
git-codecommit.us-gov-east-1.amazonaws.com	SHA256	`fVb+R0z7qW7minenW+rUpAABRCRBTCzmETAJEQrg98`
git-codecommit.eu-north-1.amazonaws.com	MD5	`8e:53:d8:59:35:88:82:fd:73:4b:60:8a:50:70:38:f4`
git-codecommit.eu-north-1.amazonaws.com	SHA256	`b6KSK7xKq+V8jl7iuAcjqXsG7zkqoUZZmmhYYFBq1wQ`
git-codecommit.me-south-1.amazonaws.com	MD5	`0e:39:28:56:d5:41:e6:8d:fa:81:45:37:fb:f3:cd:f7`
git-codecommit.me-south-1.amazonaws.com	SHA256	`O+NToCGgjrHekiBuOl0ad7ROGEsz+DBLXOd/c9wc0JU`
git-codecommit.ap-east-1.amazonaws.com	MD5	`a8:00:3d:24:52:9d:61:0e:f6:e3:88:c8:96:01:1c:fe`

サーバー	暗号ハッシュタイプ	フィンガープリント
git-codecommit.ap-east-1.amazonaws.com	SHA256	`LafadYwUYW8hONoTRpojbjNs9IRnbEwHtezD3aAIBX0`
git-codecommit.cn-north-1.amazonaws.com.cn	MD5	`11:7e:2d:74:9e:3b:94:a2:69:14:75:6f:5e:22:3b:b3`
git-codecommit.cn-north-1.amazonaws.com.cn	SHA256	`IYUXxH2OpTDsyYMLIp+JY8CTLS4UX+ZC5JVZXPRaxc8`
git-codecommit.cn-northwest-1.amazonaws.com.cn	MD5	`2e:a7:fb:4c:33:ac:6c:f9:aa:f2:bc:fb:0a:7b:1e:b6`
git-codecommit.cn-northwest-1.amazonaws.com.cn	SHA256	`wqjd6eHd0+mOBx+dCNuL0omUoCNjaDtZiEpWj5TmCfQ`
git-codecommit.eu-south-1.amazonaws.com	MD5	`b9:f6:5d:e2:48:92:3f:a9:37:1e:c4:d0:32:0e:fb:11`
git-codecommit.eu-south-1.amazonaws.com	SHA256	`lyXrWbCg3uQmJrl1XxB/ASR7ugW1Ysf5yzYOJbudHsI`
git-codecommit.ap-northeast-3.amazonaws.com	MD5	`25:17:40:da:b9:d4:18:c3:b6:b3:fb:ed:1c:20:fe:29`
git-codecommit.ap-northeast-3.amazonaws.com	SHA256	`2B815B9F0AvwLnRxSVxUz4kDYmtEQUGGdQYP8OQLXhA`

サーバー	暗号ハッシュタイプ	フィンガープリント
git-codecommit.af-south-1.amazonaws.com	MD5	`21:a0:ba:d7:c1:d1:b5:39:98:8d:4d:7c:96:f5:ca:29`
git-codecommit.af-south-1.amazonaws.com	SHA256	`C34ji3x/cnsDZjUpyNGXdE5pjHYimqJrQZ3leTgqJHM`
git-codecommit.il-central-1.amazonaws.com	MD5	`04:74:89:16:98:7a:61:b1:69:46:42:3c:d1:b4:ac:a9`
git-codecommit.il-central-1.amazonaws.com	SHA256	`uFxhp51kUWhleTLeYbxQVYm4RnNLNZ5Dbdm1cgdSl/8`

インターフェイス VPC エンドポイント AWS CodeCommit での の使用

Amazon Virtual Private Cloud (Amazon VPC) を使用して AWS リソースをホストする場合は、VPC と CodeCommit の間にプライベート接続を確立できます。この接続を使用すると、CodeCommit はパブリックインターネットを経由せずに、VPC のリソースと通信できます。

Amazon VPC は、定義した仮想ネットワークで AWS リソースを起動するために使用できる AWS サービスです。VPC を使用すると、IP アドレス範囲、サブネット、ルートテーブル、ネットワークゲートウェイなどのネットワーク設定を制御できます。VPC エンドポイントでは、VPC と AWS サービス間のルーティングは AWS ネットワークによって処理され、IAM ポリシーを使用してサービスリソースへのアクセスを制御できます。

VPC を CodeCommit に接続するには、CodeCommit のインターフェイス VPC エンドポイントを定義します。インターフェイスエンドポイントは、サポートされている AWS サービス宛てのトラフィックのエントリポイントとして機能するプライベート IP アドレスを持つ Elastic Network Interface です。このエンドポイントは、インターネットゲートウェイ、ネットワークアドレス変換 (NAT) インスタンス、および VPN 接続を必要とせず、信頼性が高くスケーラブルな CodeCommit へ

の接続を提供します。詳細については、Amazon VPC ユーザーガイドの Amazon VPC とはを参照してください。

> **Note**
>
> VPC サポートを提供し、 などの CodeCommit と統合する他の AWS サービスは AWS CodePipeline、その統合に Amazon VPC エンドポイントを使用することをサポートしていない場合があります。例えば、CodePipeline と CodeCommit の間のトラフィックを VPC サブネット範囲に制限することはできません。AWS Cloud9 など、統合をサポートするサービスでは、 AWS Systems Managerなど、追加のサービスが必要になる場合があります。

インターフェイス VPC エンドポイントは、プライベート IP アドレスを持つ Elastic Network Interface を使用して AWS サービス間のプライベート通信を可能にする AWS テクノロジーである AWS PrivateLink を利用しています。詳細については、「AWS PrivateLink」を参照してください。

以下の手順は、Amazon VPC のユーザー向けです。詳細については、『Amazon VPC ユーザーガイド』の「開始方法」を参照してください。

可用性

CodeCommit は現在、以下の VPC エンドポイントをサポートしています AWS リージョン。

- 米国東部(オハイオ)
- 米国東部 (バージニア北部)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (パリ)
- 欧州 (フランクフルト)
- 欧州 (ストックホルム)
- 欧州 (ミラノ)
- アフリカ (ケープタウン)
- イスラエル (テルアビブ)

- アジアパシフィック (東京)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (ジャカルタ)
- 中東 (アラブ首長国連邦)
- アジアパシフィック (ソウル)
- アジアパシフィック (大阪)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (ハイデラバード)
- アジアパシフィック (香港)
- 南米 (サンパウロ)
- 中東 (バーレーン)
- カナダ (中部)
- 中国 (北京)
- 中国 (寧夏)
- AWS GovCloud (米国西部)
- AWS GovCloud (米国東部)

CodeCommit 用の VPC エンドポイントを作成する

VPC で CodeCommit の使用を開始するには、CodeCommit のインターフェイス VPC エンドポイントを作成します。CodeCommit には、Git オペレーションと CodeCommit API オペレーション用に別々のエンドポイントが必要です。ビジネスニーズに応じて、複数の VPC エンドポイントを作成する必要がある場合があります。CodeCommit 用の VPC エンドポイントを作成するときは、[AWS サービス] を選択し、[サービス名] で、次のオプションから選択します。

- com.amazonaws.***region***.git-codecommit: CodeCommit リポジトリを使用した Git オペレーション用の VPC エンドポイントを作成する場合は、このオプションを選択してください。例えば、ユーザーが Git クライアントを使用し、CodeCommit リポジトリと対話するときに `git pull`、`git commit`、および `git push` などのコマンドを使用する場合は、このオプションを選択します。
- com.amazonaws.***region***.git-codecommit-fips: 連邦情報処理規格 (FIPS) 出版物 140-2 の米国政府規格に準拠した CodeCommit リポジトリの Git オペレーション用の VPC エンドポイントを作成する場合は、このオプションを選択します。

> **Note**
>
> Git の FIPS エンドポイントは、すべての AWS リージョンで利用できるわけではありません。詳細については、「Git 接続エンドポイント」を参照してください。

- com.amazonaws.*region*.codecommit: CodeCommit API オペレーション用の VPC エンドポイントを作成する場合は、このオプションを選択します。たとえば、ユーザーが AWS CLI、CodeCommit API、または AWS SDKs を使用して `CreateRepository`、、 などのオペレーションで CodeCommit を操作する場合は`ListRepositories`、このオプションを選択しますPutFile。
- com.amazonaws.*region*.codecommit-fips: 連邦情報処理規格 (FIPS) 出版物 140-2 の米国政府規格に準拠した CodeCommit API オペレーション用の VPC エンドポイントを作成する場合は、このオプションを選択してください。

> **Note**
>
> FIPS エンドポイントは、すべての AWS リージョンで利用できるわけではありません。詳細については、連邦情報処理規格 (FIPS) 140-2 の概要 AWS CodeCommit の エントリを参照してください。 https://aws.amazon.com/compliance/fips/

CodeCommit 用の VPC エンドポイントポリシーを作成する

CodeCommit には Amazon VPC エンドポイントのポリシーを作成することができます。このポリシーでは、以下の内容を指定できます。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- 自身に対してアクションを実行できたリソース。

たとえば、ある企業がリポジトリへのアクセスを VPC のネットワークアドレス範囲に制限する場合があります。このようなポリシーの例はこちらに表示されています。例 3: 指定した IP アドレス範囲から接続するユーザーにリポジトリへのアクセスを許可する 同社は、米国東部 (オハイオ) リージョンに次の 2 つの Git VPC エンドポイントを設定しました。`com.amazonaws.us-east-2.codecommit` および `com-amazonaws.us-east-2.git-codecommit-fips` です。彼らは、FIPS 準拠のエンドポイントでのみ、*MyDemoRepo* という名前の CodeCommit リポジトリ

へのコードプッシュを許可したいと考えています。これを実施するには、`com.amazonaws.us-east-2.codecommit` エンドポイントで次のようなポリシーを設定し、Git プッシュアクションを明確に拒否します。

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "codecommit:GitPush",
            "Effect": "Deny",
            "Resource": "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo",
            "Principal": "*"
        }
    ]
}
```

> **Important**
>
> グローバル条件キー`aws:VpcSourceIp`は、 `git push` コマンドの IAM ポリシーで CodeCommit リポジトリをサポートしていません。

詳細については、『Amazon VPC ユーザーガイド』の「インターフェイスエンドポイントの作成」を参照してください。

のクォータ AWS CodeCommit

次の表では、CodeCommit のクォータについて説明します。変更できるクォータの詳細については、「AWS CodeCommit エンドポイントとクォータ」を参照してください。クォータ引き上げのリクエストの情報については、AWS サービスクォータ を参照してください。Git およびその他のソフトウェアの必要なバージョンについては、「CodeCommit、Git、および他のコンポーネントの互換性」を参照してください。

承認ルールおよび承認ルールテンプレート名	1 ～ 100 文字の長さの、文字、数字、ピリオド、スペース、アンダースコア、ダッシュの任意の組み合わせ。名前では、大文字と小文字が区別されます。名前は .git で終わることはできません。また、次の文字を含めることはできません: ! ? @ # $ % ^ & * ( ) + = { } [ ] | \ / > < ~ ` ' " ; :
承認ルールのコンテンツの長さ	3000 文字
承認ルールテンプレートの説明の長さ	1000 文字
承認ルールテンプレートの送信先リファレンス	100
承認ルールテンプレート	で 1000 AWS リージョン
プルリクエストの承認ルール	最大 30 個まで。これらのうち、最大 25 個が承認ルールテンプレートからのものです。
承認ルールテンプレートから作成されるプルリクエストの承認ルール	25
プルリクエストの承認	200
承認プールの承認者	50
ブランチ名	使用できる文字の組み合わせの長さは 1 ~ 256 文字です。ただし、例外として、厳密に 40 桁の 16 進数文字列はブランチ名に使用できません。ブランチ名は以下のようにはできません。 • 先頭および末尾にスラッシュ (/) またはピリオド (.) • @ を 1 文字含める • 2 つ以上の連続するピリオド (..)、スラッシュ (//)、または次の文字の組み合わせ :@{ を含める

	• スペースまたは以下の文字を含める: ? ^ * [\ ~ : ブランチ名は参照です。ブランチ名の制限の多くは、Git 参照標準に基づいています。詳細については、「Git 内部」と「git-check-ref-format」を参照してください。
コメントの長さ	最大 10,240 文字。
で CodeCommit リポジトリを初期化するための圧縮アーカイブサイズ AWS CloudFormation	圧縮時の最大 20 GB。
カスタムデータトリガー	これは、これらに限定される文字列フィールドで 1,000 文字まで入力できます。これは動的パラメータを渡すために使用することはできません。
コンソールの表示	次のような場合、ファイルまたはファイル間の比較がコンソールに表示されないことがあります。 • ファイルが 2 MB を超えている • ファイルの 1 行に 25,000 文字を超える文字が含まれている • 比較に合計 6,500 行を超える差異が含まれている
コンソールで作成したコミットの E メールアドレス	使用できる文字の組み合わせの長さは 1 ~ 256 文字です。E メールアドレスは検証されません。

ファイルパス	使用できる文字の組み合わせの長さは 1 ~ 4,096 文字です。ファイルパスは、ファイルおよびそのファイルの正確な場所を特定する間違えのない名前にしてください。ファイルパスは、深さが 20 ディレクトリを超えることはできません。また、ファイルパスでは以下を行うことはできません。
	• 空の文字列を含むこと
	• 相対ファイルパスであること
	• 次の文字のあらゆる組み合わせが含まれていること。
	`/./`
	`/../`
	`//`
	• 末尾にスラッシュまたはバックスラッシュがあること
	
	ファイル名とパスは完全修飾である必要があります。ローカルコンピュータのファイルへの名前とパスは、オペレーティングシステムの基準に従う必要があります。CodeCommit リポジトリのファイルへのパスを指定する場合、Amazon Linux の標準を使用します。

ファイルサイズ	CodeCommit コンソール、API あるいは AWS CLIを使用する場合の個別ファイルでは最大 6 MB。

Git の blob サイズ	最大 2 GB です。 Note メタデータが 6 MB を超えず、単一の blob が 2 GB を超えない限り、単一コミット内のすべてのファイルの数または合計サイズに制限はありません。
Commit Visualizer のブランチのグラフ表示	1 ページあたり 35 です。単一ページに 35 以上のブランチがある場合、グラフは表示されません。
コミットのメタデータ	CodeCommit コンソール、API あるいは AWS CLIを使用する場合の合計最大 20 MB のコミットの メタデータの組み合わせ (例えば、作成者の情報、日付、親コミットリスト、コミットメッセージ)。 Note メタデータが 6 MB を超えず、個別ファイルが 6 MB を超えず、単一の blob が 2 GB を超えない限り、単一コミット内のすべてのファイルの数または合計サイズに制限はありません。
コミット内のファイル数	最大 100。
開いているプルリクエストの数	最大 1,000。
一回のプッシュでの参照数	最大 4,000 (作成、削除、および更新を含む)。リポジトリ内の参照の総数に制限はありません。

リポジトリの数	Amazon Web Services アカウントあたり最大 5,000。この制限は変更が可能です。詳細については、「AWS CodeCommit エンドポイントとクォータ」と「AWS のサービスクォータ」を参照してください。
リポジトリ内のトリガーの数	最大 10

リージョン	CodeCommit は、以下にあります AWS リージョン。 • 米国東部(オハイオ) • 米国東部 (バージニア北部) • 米国西部 (北カリフォルニア) • 米国西部 (オレゴン) • 欧州 (アイルランド) • 欧州 (ロンドン) • 欧州 (パリ) • 欧州 (フランクフルト) • 欧州 (ストックホルム) • 欧州 (ミラノ) • アフリカ (ケープタウン) • イスラエル (テルアビブ) • アジアパシフィック (東京) • アジアパシフィック (シンガポール) • アジアパシフィック (シドニー) • アジアパシフィック (ジャカルタ) • 中東 (アラブ首長国連邦) • アジアパシフィック (ソウル) • アジアパシフィック (大阪) • アジアパシフィック (ムンバイ) • アジアパシフィック (ハイデラバード) • アジアパシフィック (香港) • 南米 (サンパウロ) • 中東 (バーレーン) • カナダ (中部) • 中国 (北京) • 中国 (寧夏) • AWS GovCloud (米国西部)

	• AWS GovCloud (米国東部) 詳細については、「リージョンと Git 接続エンドポイント」を参照してください。
リポジトリの説明	使用できる文字の組み合わせの長さは 0 ~ 1,000 文字です。リポジトリの説明はオプションです。
リポジトリ名	1 ~ 100 文字の長さの、文字、数字、ピリオド、アンダースコア、ダッシュの任意の組み合わせ。名前では、大文字と小文字が区別されます。リポジトリ名は .git で終わることはできません。また、次の文字を含めることはできません: ! ? @ # $ % ^ & * ( ) + = { } [ ] | \ / > < ~ ` ' " ; :
リポジトリのタグキー名	任意の組み合わせで使用できる文字は、Unicode 文字、数字、スペース、および許可されている UTF-8 文字 (1 ~ 128 文字) です。使用できる文字は次のとおりです: + - = . _ : / @ タグキー名は一意である必要があり、各キーに使用できる値は 1 つのみです。タグは以下のようにはできません。 • aws: から始まる • 空白文字のみで構成されている • 末尾にスペースを使用する • 絵文字、または以下の文字を含める: ? ^ * [\ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;

リポジトリのタグ値	任意の組み合わせで使用できる文字は、Unicode 文字、数字、スペース、および許可されている UTF-8 文字 (1～256 文字) です。使用できる文字は次のとおりです: + - = . _ : / @ キーに使用できる値は 1 つのみですが、多数のキーと同じ値を含めることができます。タグは以下のようにはできません。 • 空白文字のみで構成されている • 末尾にスペースを使用する • 絵文字、または以下の文字を含める: ? ^ * [\ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;
リポジトリタグ	タグでは、大文字と小文字が区別されます。リソースあたり最大 50。厳密に 40 桁の 16 進数文字列はタグ名に使用できません。
トリガー名	1 ～ 100 文字の長さの、文字、数字、ピリオド、アンダースコア、ダッシュの任意の組み合わせ。トリガー名にスペースやカンマを含めることはできません。
コンソールで作成したコミットのユーザー名	使用できる文字の組み合わせの長さは 1 ～ 1,024 文字です。

AWS CodeCommit コマンドラインリファレンス

このリファレンスは、 AWS CLIの使用方法について説明します。

をインストールして設定するには AWS CLI

1. ローカルマシンで、 をダウンロードしてインストールします AWS CLI。これは、コマンドラインから CodeCommit とやり取りするための前提条件です。 AWS CLI バージョン 2 のインストールが推奨されます。これは の最新バージョン AWS CLI であり、すべての最新機能をサポー

トしています。これは、 でルートアカウント、フェデレーティッドアクセス、または一時的な認証情報の使用 AWS CLI をサポートする の唯一のバージョンですgit-remote-codecommit。

詳細については、AWS 「 コマンドラインインターフェイスのセットアップ」を参照してください。

> **Note**
>
> CodeCommit は AWS CLI バージョン 1.7.38 以降でのみ動作します。ベストプラクティスとして、 をインストールまたは最新バージョンにアップグレード AWS CLI します。インストール AWS CLI した のバージョンを確認するには、 aws --version コマンドを実行します。
> の古いバージョン AWS CLI を最新バージョンにアップグレードするには、「 のインストール AWS Command Line Interface」を参照してください。

2. このコマンドを実行して、 の CodeCommit コマンド AWS CLI がインストールされていることを確認します。

```
aws codecommit help
```

このコマンドは、CodeCommit コマンドのリストを返します。

3. 次のように、 configure コマンドを使用して プロファイル AWS CLI で を設定します。

```
aws configure
```

プロンプトが表示されたら、CodeCommit で使用する IAM ユーザーの AWS アクセスキーと AWS シークレットアクセスキーを指定します。また、 など、リポジトリが存在する AWS リージョン を必ず指定してくださいus-east-2。デフォルトの出力形式の入力を求められたら、json を指定します。例えば、IAM ユーザーのプロファイルを設定する場合は、次のようにします。

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

で使用するプロファイルの作成と設定の詳細については AWS CLI、以下を参照してください。

- 名前付きプロファイル
- での IAM ロールの使用 AWS CLI
- Set コマンド
- 認証情報をローテーションして AWS CodeCommit リポジトリに接続する

別の のリポジトリまたはリソースに接続するには AWS リージョン、デフォルトのリージョン名 AWS CLI で を再設定する必要があります。CodeCommit でサポートされるデフォルトのリージョン名は以下のとおりです。

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1

- ap-east-1

- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

CodeCommit と の詳細については AWS リージョン、「」を参照してくださいリージョンと Git 接続エンドポイント。IAM、アクセスキー、シークレットキーに関する詳細については、認証情報を取得する方法および IAM ユーザーのアクセスキーの管理を参照してください。 AWS CLI および プロファイルの詳細については、「名前付きプロファイル」を参照してください。

使用可能なすべての CodeCommit コマンドのリストを表示するには、次のコマンドを実行します。

```
aws codecommit help
```

特定の CodeCommit コマンドに関する情報を表示するには、次のコマンドを実行します。ここで、*command-name* はコマンドの名前です (例: create-repository)。

```
aws codecommit command-name help
```

AWS CLIのコマンドの説明と使用例については、以下を参照してください。

- associate-approval-rule-template-with-repository
- batch-associate-approval-rule-template-with-repositories
- batch-disassociate-approval-rule-template-from-repositories
- batch-describe-merge-conflicts
- batch-get-commits
- batch-get-repositories
- create-approval-rule-template
- create-branch
- create-commit

- create-pull-request
- create-pull-request-approval-rule
- create-repository
- create-unreferenced-merge-commit
- delete-approval-rule-template
- delete-branch
- delete-comment-content
- delete-file
- delete-repository
- describe-merge-conflicts
- delete-pull-request-approval-rule
- describe-pull-request-events
- disassociate-pull-request-approval-rule-template-from-repository
- evaluate-pull-request-approval-rules
- get-approval-rule-template
- get-blob
- get-branch
- get-comment
- get-comment-reactions
- get-comments-for-compared-commit
- get-comments-for-pull-request
- get-commit
- get-differences
- get-merge-commit
- get-merge-conflicts
- get-merge-options
- get-pull-request
- get-pull-request-approval-states
- get-pull-request-override-state

- get-repository
- get-repository-triggers
- list-approval-rule-templates
- list-associated-approval-rule-templates-for-repository
- list-branches
- list-pull-requests
- list-repositories
- list-repositories-for-approval-rule-template
- list-tags-for-resource
- merge-branches-by-fast-forward
- merge-branches-by-squash
- merge-branches-by-three-way
- merge-pull-request-by-fast-forward
- merge-pull-request-by-squash
- merge-pull-request-by-three-way
- override-pull-request-approval-rules
- post-comment-for-compared-commit
- post-comment-for-pull-request
- post-comment-reply
- put-comment-reaction
- put-file
- put-repository-triggers
- タグリソース
- test-repository-triggers
- タグなしリソース
- update-approval-rule-template-content
- update-approval-rule-template-description
- update-approval-rule-template-name
- update-comment

- update-default-branch
- update-pull-request-approval-rule-content
- update-pull-request-approval-state
- update-pull-request-description
- update-pull-request-status
- update-pull-request-title
- update-repository-description
- update-repository-name

基本的な Git コマンド

ローカルリポジトリと、ローカルリポジトリに接続されている CodeCommit リポジトリでの作業には Git を使用できます。

以下に示しているのは、よく使用する Git コマンドの基本的な例です。

他のオプションについては、Git のドキュメントを参照してください。

トピック

- 設定変数
- リモートリポジトリ
- コミット
- ブランチ
- [タグ]

設定変数

設定変数を一覧表示します。	`git config --list`
ローカル設定変数のみを一覧表示します。	`git config --local -l`
システム設定変数のみを一覧表示します。	`git config --system -l`
グローバル設定変数のみを一覧表示します。	`git config --global -l`

指定した設定ファイル内の設定変数を設定します。	`git config [--local | --global | --system] variable-name variable-value`
デフォルトのブランチを持たないリポジトリに初期コミットが行われたときに、すべてのローカルリポジトリのデフォルトのブランチ名を main に設定します	git config --global init.defaultBranch main
設定ファイルを直接編集します。特定の設定ファイルの場所を検出するためにも使用できます。編集モードを終了するには、通常、「`:q`」(変更を保存せずに終了) または「`:wq`」(変更を保存して終了) と入力し、Enter キーを押します。	`git config [--local | --global | --system] --edit`

リモートリポジトリ

CodeCommit リポジトリに接続するための準備として、ローカルリポジトリを初期化します。	`git init`
ローカルリポジトリで CodeCommit リポジトリに使用するニックネームと、CodeCommit リポジトリへの URL を指定して、ローカルリポジトリとリモートリポジトリ (CodeCommit リポジトリなど) との間の接続を設定できます。	`git remote add remote-name remote-url`
指定した URL にある CodeCommit リポジトリをコピーすることで、ローカルマシン上の現在のフォルダの指定したサブフォルダに、ローカルリポジトリを作成します。このコマンドは、複製された CodeCommit リポジトリ内のブランチごとにリモート追跡ブランチを作成し、複製された CodeCommit リポジトリ内の現在のデフォルトブランチから分岐される最初のブランチを作成してチェックアウトします。	`git clone remote-url local-subfolder-name`

ローカルリポジトリが CodeCommit リポジトリに使用するニックネームを表示します。	`git remote`
ローカルリポジトリが CodeCommit リポジトリに対するフェッチとプッシュに使用するニックネームと URL を表示します。	`git remote -v`
ローカルリポジトリが CodeCommit に使用するニックネームとブランチを指定して、ローカルリポジトリから CodeCommit リポジトリに確定済みのコミットをプッシュします。また、プッシュ中にローカルリポジトリのアップストリーム追跡情報を設定します。	`git push -u remote-name branch-name`
アップストリーム追跡情報の設定後、ローカルリポジトリから CodeCommit に確定済みのコミットをプッシュします。	`git push`
ローカルリポジトリが CodeCommit に使用するニックネームとブランチを指定して、CodeCommit リポジトリからローカルリポジトリに確定済みのコミットをプルします。	`git pull remote-name branch-name`
アップストリーム追跡情報の設定後、CodeCommit リポジトリからローカルリポジトリに確定済みのコミットをプルします。	`git pull`
ローカルリポジトリが CodeCommit リポジトリに使用するニックネームを指定して、CodeCommit リポジトリからローカルリポジトリを切断します。	`git remote rm remote-name`

コミット

ローカルリポジトリ内の保留中のコミットに何が追加されて何が追加されていないかを表示します。	`git status`
ローカルリポジトリ内の保留中のコミットに何が追加されて何が追加されていないかを簡潔な形式で表示します。 (M = 変更、A = 追加、D = 削除など)	`git status -sb`
ローカルリポジトリ内の保留中のコミットと最新のコミットとの間の変更を表示します。	`git diff HEAD`
ローカルリポジトリ内の保留中のコミットに特定のファイルを追加します。	`git add [file-name-1 file-name-2 file-name-N | file-pattern ]`
ローカルリポジトリ内の保留中のコミットにすべての新規、変更済み、削除済みのファイルを追加します。	`git add`
ローカルリポジトリ内の保留中のコミットを確定して、エディタにコミットメッセージを表示します。メッセージが入力されると、保留中のコミットが確定されます。	`git commit`
ローカルリポジトリ内の保留中のコミットを確定すると同時に、コミットメッセージを指定します。	`git commit -m "Some meaningful commit comment"`
ローカルリポジトリ内の最新のコミットを一覧表示します。	`git log`
ローカルリポジトリ内の最新のコミットをグラフ形式で示します。	`git log --graph`
ローカルリポジトリ内の最新のコミットを、定義済みの短縮形式で示します。	`git log --pretty=oneline`

ローカルリポジトリ内の最新のコミットを、定義済みの短縮形式でグラフと共に示します。	`git log --graph --pretty=oneline`
ローカルリポジトリ内の最新のコミットを、カスタム形式でグラフと共に示します。 （その他のオプションについては、Git ドキュメントの「履歴を理解する：コミット」を参照してください）。	`git log --graph --pretty=format:"%H (%h) : %cn : %ar : %s"`

ブランチ

ローカルリポジトリ内のすべてのブランチを一覧表示します。現在のブランチの横にはアスタリスク (*) が表示されます。	`git branch`
CodeCommit リポジトリ内のすべての既存のブランチに関する情報をローカルリポジトリにプルします。	`git fetch`
ローカルリポジトリ内のすべてのブランチと、ローカルリポジトリ内のリモート追跡ブランチを一覧表示します。	`git branch -a`
ローカルリポジトリ内のリモート追跡ブランチのみを一覧表示します。	`git branch -r`
指定したブランチ名を使用して、ローカルリポジトリ内で新しいブランチを作成します。	`git branch` *`new-branch-name`*
指定したブランチ名を使用して、ローカルリポジトリ内で別のブランチに切り替えます。	`git checkout` *`other-branch-name`*
指定したブランチ名を使用して、ローカルリポジトリ内で新しいブランチを作成し、そのブランチに切り替えます。	`git checkout -b` *`new-branch-name`*

ローカルリポジトリが CodeCommit リポジトリに使用するニックネームとブランチ名を指定して、ローカルリポジトリから CodeCommit リポジトリに新しいブランチをプッシュします。また、プッシュ中にローカルリポジトリ内のブランチのアップストリーム追跡情報を設定します。	`git push -u remote-name new-branch-name`
指定したブランチ名を使用して、ローカルリポジトリ内で新しいブランチを作成します。その後、ローカルリポジトリが CodeCommit に使用するニックネームとブランチ名を指定して、ローカルリポジトリ内の新しいブランチを CodeCommit リポジトリ内の既存のブランチに接続します。	`git branch --track new-branch-name remote-name /remote-branch-name`
ローカルリポジトリ内の別のブランチからローカルリポジトリ内の現在のブランチに変更をマージします。	`git merge from-other-branch-name`
マージされていない作業が含まれていない限り、ローカルリポジトリ内のブランチを削除します。	`git branch -d branch-name`
ローカルリポジトリが CodeCommit リポジトリに使用するニックネームとブランチ名を指定して、CodeCommit リポジトリのブランチを削除します。(コロン (:) の使用に注意してください)。	`git push remote-name :branch-name`

[タグ]

ローカルリポジトリ内のすべてのタグを一覧表示します。	`git tag`

CodeCommit リポジトリからローカルリポジトリにすべてのタグをプルします。	`git fetch --tags`
ローカルリポジトリ内の特定のタグに関する情報を表示します。	`git show tag-name`
ローカルリポジトリで、「軽量な」タグを作成します。	`git tag tag-name commit-id-to-point-tag-at`
ローカルリポジトリが CodeCommit リポジトリに使用するニックネームとタグ名を指定して、ローカルリポジトリから CodeCommit リポジトリに特定のタグをプッシュします。	`git push remote-name tag-name`
ローカルリポジトリが CodeCommit リポジトリに使用するニックネームを指定して、ローカルリポジトリから CodeCommit リポジトリにすべてのタグをプッシュします。	`git push remote-name --tags`
ローカルリポジトリ内のタグを削除します。	`git tag -d tag-name`
ローカルリポジトリが CodeCommit リポジトリに使用するニックネームとタグ名を指定して、CodeCommit リポジトリのタグを削除します。(コロン (:) の使用に注意してください)。	`git push remote-name :tag-name`

AWS CodeCommit ユーザーガイドのドキュメント履歴

以下の表は CodeCommit ドキュメントの重要な変更点をまとめたものです。このドキュメントの更新に関する通知については、RSS フィードにサブスクライブできます。

- API バージョン: 2015-04-13

変更	説明	日付
AWS CodeCommit は、新規顧客には利用できなくなりました。	AWS CodeCommit は、新規顧客には利用できなくなりました。 AWS CodeCommit の既存のお客様は、通常どおりサービスを引き続き使用できます。詳細はこちら	2024 年 7 月 25 日
認証情報ヘルパーのトラブルシューティング情報を更新しました	不完全または誤って設定された .gitconfig ファイルに起因するエラーのトラブルシューティングに関する情報を追加しました。	2024 年 7 月 9 日
のトラブルシューティング情報を更新しました `git-remote-codecommit`	外部管理環境のエラーのトラブルシューティングに関する追加情報を追加しました。	2024 年 7 月 9 日
への参照を削除しました AWS Toolkit for Eclipse	有効期限 AWS Toolkit for Eclipse に達し、サポートされなくなりました。このツールキットへの参照は削除されました。	2024 年 7 月 9 日
CodeCommit がカスタマーマネージドキーの使用をサポート	カスタマーマネージドキーまたは を使用して AWS マネージドキー 、リポジトリ内のデータを暗号化および復号できるようになりました。詳細	2023 年 12 月 21 日

	については、「AWS KMS と暗号化」、「リポジトリの作成」、および「リポジトリ設定の変更」を参照してください。	
CodeCommit がイスラエル (テルアビブ) で利用可能になりました。	CodeCommit がイスラエル (テルアビブ) で利用可能になりました。詳細については、リージョンおよび Git 接続エンドポイントを参照してください。	2023 年 8 月 28 日
CodeCommit の管理ポリシーの変更	AWSCodeCommitPowerUser ポリシーと AWSCodeCommitFullAccess ポリシーが更新され、アクセス許可が追加されました。詳細については、CodeCommit updates to AWS managed policies」を参照してください。	2023 年 5 月 16 日
CodeCommit が 3 つの追加で利用可能になりました AWS リージョン	CodeCommit が、アジアパシフィック (ジャカルタ) AWS リージョン、中東 (アラブ首長国連邦) 、アジアパシフィック (ハイデラバード) の 3 つの追加で利用可能になりました。詳細については、リージョンおよび Git 接続エンドポイントを参照してください。	2023 年 2 月 28 日

CodeCommit がアフリカ (ケープタウン) で利用可能になりました	CodeCommit がアフリカ AWS リージョン (ケープタウン) で利用可能になりました。詳細については、リージョンおよび Git 接続エンドポイントを参照してください。	2021 年 9 月 15 日
CodeCommit の管理ポリシーの変更	CodeCommit の AWS 管理ポリシーの更新に関する詳細が利用可能になりました。詳細については、CodeCommit updates to AWS managed policies」を参照してください。	2021 年 8 月 18 日
CodeCommit がアジアパシフィック (大阪) で利用可能に	CodeCommit が、 AWS リージョンアジアパシフィック (大阪) の追加で利用可能になりました。詳細については、リージョンおよび Git 接続エンドポイントを参照してください。	2021 年 4 月 14 日

AWS CloudFormation CodeCommit のデフォルトブランチの命名動作 AWS Cloud Development Kit (AWS CDK) を変更する および の変更	コードの初期コミット AWS CDK で AWS CloudFormation または を使用して作成されたリポジトリは、デフォルトのブランチ名 main を使用するようになりました。この変更は、既存のリポジトリまたはブランチには影響しません。ローカル Git クライアントを使用して初期コミットを作成するお客様は、これらの Git クライアントの設定に続いてデフォルトのブランチ名を持つことになります。詳細については、「 を使用した CodeCommit リソースの作成 AWS CloudFormation」を参照してください。	2021 年 3 月 4 日
CodeCommit がデフォルトブランチの名前付け動作を変更	2021 年 1 月 19 日現在、CodeCommit リポジトリへの最初のコミットによって作成されるデフォルトのブランチ名は main です。この変更は、既存のリポジトリまたはブランチには影響しません。ローカル Git クライアントを使用して初期コミットを作成するお客様は、これらの Git クライアントの設定に続いてデフォルトのブランチ名を持つことになります。詳細については、ブランチを操作する、コミットを作成する、およびブランチ設定を変更するを参照してください。	2021 年 1 月 19 日

CodeCommit が欧州 (ミラノ) で利用可能に	CodeCommit が、 AWS リージョン欧州 (ミラノ) の追加で利用可能になりました。詳細については、リージョンおよび Git 接続エンドポイントを参照してください。	2020 年 9 月 16 日
CodeCommit が、コメントへの絵文字リアクションのサポートを追加	CodeCommit では、他のユーザーからのコメントに対する絵文字リアクションがサポートされるようになりました。詳細については、コミットへのコメントとプルリクエストの確認を参照してください。	2020 年 6 月 24 日
CodeCommit が中国 (北京) と中国 (寧夏) で利用可能に	CodeCommit が、中国 (北京) と中国 (寧夏) AWS リージョンの 2 つの追加で利用可能になりました。詳細については、リージョンおよび Git 接続エンドポイントを参照してください。	2020 年 4 月 23 日

CodeCommit で git-remote-codecommit のサポートが追加	CodeCommit は、Git を変更するユーティリティ git-remote-codecommit で HTTPS 経由の CodeCommit リポジトリへの接続をサポートします。これは、CodeCommit リポジトリへのフェデレーティッドアクセス接続または一時的なアクセス接続に推奨されるアプローチです。IAM ユーザーで git-remote-codecommit を使用することもできます。git-remote-codecommit では、ユーザーの Git 認証情報を設定する必要はありません。詳細については、git-remote-codecommit AWS CodeCommit を使用した への HTTPS 接続のセットアップ手順」を参照してください。	2020 年 3 月 4 日

CodeCommit がセッションタグをサポート	CodeCommit では、セッションタグの使用がサポートされています。セッションタグは、IAM ロールを引き受けるとき、一時的な認証情報を使用するとき、または AWS Security Token Service (AWS STS) でユーザーをフェデレートするときに渡すキーと値のペアの属性です。これらのタグで提供される情報を使用して、変更を加えた人やイベントを発生させたユーザーを簡単に特定できます。詳細については、CodeCommit のモニタリングおよびタグを使用して CodeCommit の ID 情報を提供するを参照してください。	2019 年 12 月 19 日
CodeCommit がアジアパシフィック (香港) で利用可能に	CodeCommit がアジアパシフィック (香港) で利用可能になりました。Git 接続エンドポイントなど、詳細については、リージョンを参照してください。	2019 年 12 月 11 日

CodeCommit が Amazon CodeGuru Reviewer をサポート	CodeCommit は、プログラム分析と機械学習を使用して一般的な問題を検出し、Java または Python コードにおける修正点を提案する自動化されたコードレビューサービスである Amazon CodeGuru Reviewer をサポートしています。詳細については、Amazon CodeGuru Reviewer に対するリポジトリの関連付けまたは関連付け解除およびプルリクエストの操作を参照してください。	2019 年 12 月 3 日
CodeCommit が承認ルールをサポート	承認ルールを使用して、異なるブランチがプルリクエストに対して適切なレベルの承認とコントロールを持つように、リポジトリ間で開発ワークフローをカスタマイズできるようになりました。詳細については、承認ルールテンプレートの操作およびプルリクエストの操作を参照してください。	2019 年 11 月 20 日
CodeCommit が通知ルールをサポート	通知ルールを使用して、リポジトリの重要な変更をユーザーに通知できるようになりました。この機能により、2019 年 11 月 5 日より前に作成された通知は置き換わります。詳細については、通知ルールを作成するを参照してください。	2019 年 11 月 5 日

CodeCommit が中東 (バーレーン) で利用可能に	CodeCommit が中東 (バーレーン) で利用可能になりました。Git 接続エンドポイントなど、詳細については、リージョンを参照してください。	2019 年 10 月 30 日
CodeCommit に、複数のコミットに関する情報を取得するためのサポートが追加	AWS CLIの batch-get-commits コマンドを使用して、複数のコミットに関する情報を取得できます。詳細については、コミットの詳細表示を参照してください。	2019 年 8 月 15 日
CodeCommit が欧州 (ストックホルム) で利用可能に	CodeCommit が欧州 (ストックホルム) で利用可能になりました。Git 接続エンドポイントなど、詳細については、リージョンを参照してください。	2019 年 7 月 31 日
CodeCommit が CodeCommit コンソールでのリポジトリのタグ付けのサポートを追加	リポジトリのタグを追加、管理、削除して、CodeCommit コンソールから AWS リソースを管理できるようになりました。詳細については、リポジトリのタグ付けを参照してください。	2019 年 7 月 2 日
CodeCommit が Git マージ戦略へのサポートをさらに追加	CodeCommit でプルリクエストをマージするときに、Git マージ戦略から選択できるようになりました。また、CodeCommit コンソールでマージの競合を解決することもできます。詳細については、「プルリクエストの使用」を参照してください。	2019 年 6 月 10 日

CodeCommit が AWS GovCloud (米国東部) で利用可能に	AWS GovCloud (米国東部) で CodeCommit を使用できるようになりました。Git 接続エンドポイントなど、詳細については、リージョンを参照してください。	2019 年 5 月 31 日
リポジトリをタグ付けするサポートが CodeCommit に追加	リポジトリのタグを追加、管理、削除して、 AWS リソースの管理に役立てることができるようになりました。詳細については、リポジトリのタグ付けを参照してください。	2019 年 5 月 30 日
コンソールでリソースを検索する	リポジトリ、ビルドプロジェクト、デプロイアプリケーション、パイプラインなどのリソースをすばやく検索することができるようになりました。[Go to resource (リソースに移動)] または / キーを押して、リソースの名前を入力します。詳細については、CodeCommit チュートリアルを参照してください。	2019 年 5 月 14 日
CodeCommit が AWS GovCloud (米国西部) で利用可能に	AWS GovCloud (米国西部) で CodeCommit を使用できるようになりました。Git 接続エンドポイントなど、詳細については、リージョンを参照してください。	2019 年 4 月 18 日

CodeCommit が Amazon VPC エンドポイントのサポートを追加	これで、VPC と CodeCommit との間でプライベート接続を確立できます。詳細については、インターフェイス VPC エンドポイントでの CodeCommit の使用を参照してください。	2019 年 3 月 7 日
CodeCommit が新しい API を追加	CodeCommit はコミットを作成するための API を追加しました。詳細については、コミットの作成を参照してください。	2019 年 2 月 20 日
コンテンツの更新	このガイドの内容は、わずかな修正とトラブルシューティングガイダンスが追加されて更新されています。	2019 年 1 月 2 日
コンテンツの更新	このガイドのコンテンツは、新しい CodeCommit コンソールのエクスペリエンスに対応するために更新されました。	2018 年 10 月 30 日
CodeCommit および連邦情報処理標準 (FIPS)	CodeCommit は、一部のリージョンで連邦情報処理規格 (FIPS) 出版物 140-2 の政府規格に対するサポートを追加しました。FIPS および 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) 140-2 の概要」を参照してください。Git 接続エンドポイントの詳細については、リージョンを参照してください。	2018 年 10 月 25 日

CodeCommit が 3 つの API を追加	CodeCommit で、ファイルの操作をサポートするために 3 つの API が追加されました。Git 接続のエンドポイントの詳細については、個別のファイルに対するアクションのアクセス許可およびAWS CodeCommit API リファレンスを参照してください。	2018 年 9 月 27 日
CodeCommit のドキュメント履歴通知が RSS フィード経由で利用可能に	RSS フィードにサブスクライブして、CodeCommit ドキュメントの更新に関する通知を受信できるようになりました。	2018 年 6 月 29 日

以前の更新

以下の表は、2018 年 6 月 29 日以前のドキュメントの重要な変更点をまとめたものです。

変更	説明	変更日
新しいトピック	「ブランチに対するプッシュとマージの制限」トピックが追加されました。「CodeCommit アクセス許可リファレンス」トピックが更新されました。	2018 年 5 月 16 日
新規セクション	「AWS CodeCommit リポジトリ内のファイルの使用」 セクションが追加されました。「CodeCommit アクセス許可リファレンス」および「の開始方法 AWS CodeCommit」トピックが更新されました。	2018 年 2 月 21 日
新しいトピック	「ロールを使用して AWS CodeCommit リポジトリへのクロスアカウントアクセスを設定する」トピックが追加されました。	2018 年 2 月 21 日

変更	説明	変更日
新しいトピック	「AWS Cloud9 との統合 AWS CodeCommit」トピックが追加されました。製品およびサービスの統合 トピックがに関する情報で更新されました AWS Cloud9。	2017 年 1 月 12 日
新規セクション	「AWS CodeCommit リポジトリのプルリクエストを操作する」 セクションが追加されました。プルリクエストとコメントのアクセス権限に関する情報で「AWS CodeCommitの認証とアクセスコントロール」が更新されました。更新された管理ポリシーのステートメントも含まれています。	2017 年 11 月 20 日
トピックの更新	製品およびサービスの統合 トピックが更新され、既存のパイプラインを更新し、Amazon CloudWatch Events を使用して CodeCommit リポジトリが変更されたときにパイプラインを開始したいと思っているお客様向けのリンクが追加されました。	2017 年 10 月 11 日
新しいトピック	「AWS CodeCommitの認証とアクセスコントロール」 セクションが追加されました。これは、アクセス権限リファレンスのトピックと置き換わります。	2017 年 9 月 11 日
トピックの更新	リポジトリのトリガーの管理 セクションは、トリガー設定の変更を反映して更新されました。ガイド全体にわたり、ナビゲーションバーの変更を反映するようにトピックとイメージが更新されています。	2017 年 8 月 29 日
新しいトピック	「ユーザー設定の操作」トピックが追加されました。「タグの詳細の表示」トピックが更新されました。製品およびサービスの統合 トピックが更新され、Amazon CloudWatch Events との統合に関する情報が更新されました。	2017 年 8 月 3 日
新しいトピック	Eclipse を AWS CodeCommit および と統合するVisual Studio を と統合する AWS CodeCommitトピックが追加されました。	2017 年 6 月 29 日

変更	説明	変更日
トピックの更新	CodeCommit は、アジアパシフィック (ムンバイ) とカナダ (中部) の 2 つの追加リージョンで利用可能になりました。「リージョンと Git 接続エンドポイント」トピックが更新されました。	2017 年 6 月 29 日
トピックの更新	CodeCommit は、アジアパシフィック (ソウル)、南米 (サンパウロ)、米国西部 (北カリフォルニア)、欧州 (ロンドン) の 4 つの追加リージョンで利用可能になりました。「リージョンと Git 接続エンドポイント」トピックが更新されました。	2017 年 6 月 6 日
トピックの更新	CodeCommit は、アジアパシフィック (東京)、アジアパシフィック (シンガポール)、アジアパシフィック (シドニー)、欧州 (フランクフルト) の 4 つの追加リージョンで利用可能になりました。リージョンと Git 接続エンドポイント トピックが更新され、CodeCommit の Git 接続エンドポイントとサポートされているリージョンに関する情報が提供されました。	2017 年 5 月 25 日
新しいトピック	「ブランチの比較とマージ」トピックが追加されました。ブランチの操作 セクションは、CodeCommit コンソールを使用したリポジトリのブランチでの作業についての情報が更新されています。	2017 年 5 月 18 日
新しいトピック	「コミットの比較」トピックには、コミットの比較に関する情報が追加されています。ユーザーガイドの構造は、リポジトリ、コミット、ブランチ で作業するために更新されました。	2017 年 3 月 28 日
トピックの更新	コミットの詳細の表示 トピックは、コンソール内のコミットとその親の違いを表示し、get-differences を使用してコミット間の相違を表示するために AWS CLIコマンドを使用する情報が更新されました。	2017 年 1 月 24 日

変更	説明	変更日
新しいトピック	を使用した AWS CodeCommit API コールのログ記録 AWS CloudTrail トピックが追加され、 AWS CloudFormationを使用して CodeCommit への接続をログ記録するための情報が提供されました。	2017 年 1 月 11 日
新しいトピック	Git 認証情報を使用した HTTPS ユーザーのセットアップ トピックが追加され、HTTPS 経由の Git 認証情報を使用して CodeCommit への接続を設定するための情報が提供されました。	2016 年 22 月 12 日
トピックの更新	製品およびサービスの統合 トピックが更新され、 との統合に関する情報が追加されました AWS CodeBuild。	2016 年 5 月 12 日
トピックの更新	CodeCommit は、もう 1 つのリージョン (欧州 (アイルランド)) で利用可能になりました。リージョンと Git 接続エンドポイント トピックが更新され、CodeCommit の Git 接続エンドポイントとサポートされているリージョンに関する情報が提供されました。	2016 年 11 月 16 日
トピックの更新	CodeCommit は、もう 1 つのリージョン (米国西部 (オレゴン)) で利用可能になりました。リージョンと Git 接続エンドポイント トピックが更新され、CodeCommit の Git 接続エンドポイントとサポートされているリージョンに関する情報が提供されました。	2016 年 11 月 14 日
新しいトピック	Lambda 関数のトリガーを作成する トピックが更新され、Lambda 関数の作成の一部として CodeCommit トリガーを作成する機能が反映されました。この簡素化されたプロセスは、トリガーの作成を合理化し、CodeCommit が Lambda 機能を呼び出すのに必要なアクセス許可でトリガーを自動的に設定します。既存の Lambda 関数のトリガーを作成する トピックが追加され、CodeCommit コンソールに既存の Lambda 機能のトリガーを作成するための情報が含まれました。	2016 年 10 月 19 日

変更	説明	変更日
新しいトピック	CodeCommit は、もう 1 つのリージョン (米国東部 (オハイオ)) で利用可能になりました。リージョンと Git 接続エンドポイント トピックが追加され、CodeCommit の Git 接続エンドポイントとサポートされているリージョンに関する情報が提供されました。	2016 年 10 月 17 日
トピックの更新	製品およびサービスの統合 トピックが更新され、 との統合に関する情報が追加されました AWS Elastic Beanstalk 。	2016 年 10 月 13 日
トピックの更新	製品およびサービスの統合 トピックが更新され、 との統合に関する情報が追加されました AWS CloudFormation。	2016 年 10 月 6 日
トピックの更新	「Windows で SSH 接続をセットアップする手順」トピック改訂され、PuTTY スイートの代わりに Windows で SSH 接続用の Bash エミュレーターを使用するためのガイダンスが提供されました。	2016 年 9 月 29 日
トピックの更新	コミットの詳細の表示 および CodeCommit の開始方法 のトピックが更新され、CodeCommit コンソールの Commit Visualizer に関する情報が含まれました。「クォータ」トピックが更新され、一回のプッシュで許可された参照の数が増えました。	2016 年 9 月 14 日
トピックの更新	コミットの詳細の表示 および CodeCommit の開始方法 トピックが更新され、CodeCommit コンソールにコミットの履歴を表示するための情報が含まれました。	2016 年 7 月 28 日
新しいトピック	「Git リポジトリを に移行する AWS CodeCommit」および「ローカルまたはバージョン管理されていないコンテンツを に移行する AWS CodeCommit」 トピックが追加されました。	2016 年 6 月 29 日
トピックの更新	「トラブルシューティング」および「 AWS CLI 認証情報ヘルパーを使用した Windows での HTTPS 接続の場合」トピックのマイナーアップデートが行われました。	2016 年 6 月 22 日

変更	説明	変更日
トピックの更新	製品およびサービスの統合 およびアクセス許可リファレンスのトピックが更新され、CodePipeline との統合に関する情報が含まれました。	2016 年 4 月 18 日
新しいトピック	「リポジトリのトリガーの管理」 セクションが追加されました。新しいトピックには、ポリシーとコードのサンプル、トリガーの作成、編集、および削除方法の例が含まれています。	2016 年 3 月 7 日
新しいトピック	「製品およびサービスの統合」トピックが追加されました。「トラブルシューティング」にマイナーな更新が加えられました。	2016 年 3 月 7 日
トピックの更新	MD5 サーバーのフィンガープリントに加えて、CodeCommit の SHA256 サーバーのフィンガープリントが Linux、macOS、または Unix での SSH 接続の場合 および Windows で SSH 接続をセットアップする手順 に追加されました。	2015 年 12 月 9 日
新しいトピック	「リポジトリでのファイルの参照」トピックが追加されました。新しい問題が、「トラブルシューティング」に追加されました。マイナー改善と修正がユーザーガイドに加えられました。	2015 年 10 月 5 日
新しいトピック	「を使用していない SSH ユーザーの場合 AWS CLI」トピックが追加されました。この「設定 」セクションのトピックが効率化されました。ユーザーがオペレーティングシステムおよび推奨プロトコルに従う手順を決定する手助けとなるガイダンスが提供されています。	2015 年 8 月 5 日
トピックの更新	「SSH および Linux、macOS、または Unix: Git と CodeCommit 用にパブリックキーとプライベートキーをセットアップする」と「ステップ 3: Git および CodeCommit 用のパブリックキーとプライベートキーをセットアップする」の SSH キー ID ステップに説明と例が追加されました。	2015 年 7 月 24 日

変更	説明	変更日
トピックの更新	ステップ 3: Git および CodeCommit 用のパブリックキーとプライベートキーをセットアップする のステップが、IAM の問題に対処し、パブリックキーファイルを保存するように更新されました。	2015 年 7 月 22 日
トピックの更新	トラブルシューティング「」がナビゲーションの支援で更新されました。認証情報のキーチェーンの問題に関するその他のトラブルシューティング情報が追加されました。	2015 年 7 月 20 日
トピックの更新	アクセス AWS Key Management Service 許可の詳細については、AWS KMS および 暗号化「」および「アクセス許可リファレンス」トピックに追加されています。	2015 年 7 月 17 日
トピックの更新	の問題のトラブルシューティングに関する情報を含む別のセクションが、「トラブルシューティング」に追加されました。。 AWS Key Management Service	2015 年 7 月 10 日
初回リリース	これは、CodeCommit ユーザーガイドの初版です。	2015 年 7 月 9 日

AWS 用語集

最新の AWS 用語については、 AWS の用語集 リファレンスのAWS 用語集を参照してください。