AWS Whitepaper

Introduction to DevOps on AWS

Introduction to DevOps on AWS: AWS Whitepaper

Table of Contents

Introduction to DevOps on AWS

Publication date: **April 7, 2023** (*Document Revisions*)

Today more than ever, enterprises are embarking on their digital transformation journey to build deeper connections with their customers, to achieve sustainable and enduring business value. Organizations of all shapes and sizes are disrupting their competitors and entering new markets by innovating more quickly than ever before. For these organizations, it is important to focus on innovation and software disruption, making it critical to streamline their software delivery. Organizations that shorten their time from idea to production making speed and agility a priority could be tomorrow's disruptors.

While there are several factors to consider in becoming the next digital disruptor, this whitepaper focuses on DevOps, and the services and features in the Amazon Web Services (AWS) platform that will help increase an organization's ability to deliver applications and services at a high velocity.

Introduction

DevOps is the combination of cultural philosophies, engineering practices, and tools which increase an organization's ability to deliver applications and services at high velocity and better quality. Over time, several essential practices have emerged when adopting DevOps: continuous integration (CI), continuous delivery (CD), Infrastructure as Code (IaC), and monitoring and logging.

This paper highlights AWS capabilities that help you accelerate your DevOps journey, and how AWS services can help remove the undifferentiated heavy lifting associated with DevOps adaptation. It also describes how to build a continuous integration and delivery capability without managing servers or build nodes, and how to use IaC to provision and manage your cloud resources in a consistent and repeatable manner.

- **Continuous integration:** A software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run.
- **Continuous delivery:** A software development practice where code changes are automatically built, tested, and prepared for a release to production.
- **Infrastructure as Code:** A practice in which infrastructure is provisioned and managed using code and software development techniques, such as version control, and continuous integration.
- **Monitoring and logging:** Enables organizations to see how application and infrastructure performance impacts the experience of their product’s end user.

- **Communication and collaboration:** Practices are established to bring the teams closer and by building workflows and distributing the responsibilities for DevOps.
- **Security:** Should be a cross cutting concern. Your continuous integration and continuous delivery (CI/CD) pipelines and related services should be safeguarded and proper access control permissions should be set up.

An examination of each of these principles reveals a close connection to the offerings available from AWS.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

Continuous integration

Continuous integration (CI) is a software development practice where developers regularly merge their code changes into a central code repository, after which automated builds and tests are run. CI helps find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

AWS offers the following services for continuous integration:

Topics

- AWS CodeCommit
- AWS CodeBuild
- AWS CodeArtifact

AWS CodeCommit

AWS CodeCommit is a secure, highly scalable, managed source control service that hosts private git repositories. CodeCommit reduces the need for you to operate your own source control system and there is no hardware to provision and scale or software to install, configure, and operate. You can use CodeCommit to store anything from code to binaries, and it supports the standard functionality of GitHub, allowing it to work seamlessly with your existing Git-based tools. Your team can also use CodeCommit's online code tools to browse, edit, and collaborate on projects. AWS CodeCommit has several benefits:

- **Collaboration** — AWS CodeCommit is designed for collaborative software development. You can easily commit, branch, and merge your code, which helps you easily maintain control of your team's projects. CodeCommit also supports pull requests, which provide a mechanism to request code reviews and discuss code with collaborators.
- **Encryption** — You can transfer your files to and from AWS CodeCommit using HTTPS or SSH, as you prefer. Your repositories are also automatically encrypted at rest through AWS Key Management Service (AWS KMS) using customer-specific keys.
- **Access control** — AWS CodeCommit uses AWS Identity and Access Management (IAM) to control and monitor who can access your data in addition to how, when, and where they can access it. CodeCommit also helps you monitor your repositories through AWS CloudTrail and Amazon CloudWatch.

High availability and durability — AWS CodeCommit stores your repositories in Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB. Your encrypted data is redundantly stored across multiple facilities. This architecture increases the availability and durability of your repository data.

- **Notifications and custom scripts** — You can now receive notifications for events impacting your repositories. Notifications will come as Amazon Simple Notification Service (Amazon SNS) notifications. Each notification will include a status message as well as a link to the resources whose event generated that notification. Additionally, using AWS CodeCommit repository cues, you can send notifications and create HTTP webhooks with Amazon SNS or invoke AWS Lambda functions in response to the repository events you choose.

AWS CodeBuild

AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. You don't need to provision, manage, and scale your own build servers. CodeBuild can use either of GitHub, GitHub Enterprise, BitBucket, AWS CodeCommit, or Amazon S3 as a source provider.

CodeBuild scales continuously and can process multiple builds concurrently. CodeBuild offers various pre-configured environments for various versions of Microsoft Windows and Linux. Customers can also bring their customized build environments as Docker containers. CodeBuild also integrates with open source tools such as Jenkins and Spinnaker.

CodeBuild can also create reports for unit, functional, or integration tests. These reports provide a visual view of how many tests cases were run and how many passed or failed. The build process can also be run inside an Amazon Virtual Private Cloud (Amazon VPC) which can be helpful if your integration services or databases are deployed inside a VPC.

AWS CodeArtifact

AWS CodeArtifact is a fully managed artifact repository service that can be used by organizations to securely store, publish, and share software packages used in their software development process. CodeArtifact can be configured to automatically fetch software packages and dependencies from public artifact repositories so developers have access to the latest versions.

Software development teams increasingly rely on open-source packages to perform common tasks in their application package. It has become critical for software development teams to maintain

control on a particular version of the open-source software to ensure the software is free of vulnerabilities. With CodeArtifact, you can set up controls to enforce this.

CodeArtifact works with commonly used package managers and build tools such as Maven, Gradle, npm, yarn, twine, and pip, making it easy to integrate into existing development workflows.

Continuous delivery

Continuous delivery (CD) is a software development practice where code changes are automatically prepared for a release to production. A pillar of modern application development, continuous delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage. When properly implemented, developers will always have a deployment-ready build artifact that has passed through a standardized test process.

Continuous delivery lets developers automate testing beyond just unit tests so they can verify application updates across multiple dimensions before deploying to customers.

These tests might include UI testing, load testing, integration testing, API reliability testing, and more. This helps developers more thoroughly validate updates and preemptively discover issues. Using the cloud, it is easy and cost-effective to automate the creation and replication of multiple environments for testing, which was previously difficult to do on-premises.

AWS offers the following services for continuous delivery:

- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline

Topics

AWS CodeDeploy

AWS CodeDeploy is a fully managed deployment service that automates software deployments to a variety of compute services such as Amazon Elastic Compute Cloud (Amazon EC2), AWS Fargate, AWS Lambda, and your on-premises servers. AWS CodeDeploy makes it easier for you to rapidly release new features, helps you avoid downtime during application deployment, and handles the complexity of updating your applications. You can use CodeDeploy to automate software deployments, reducing the need for error-prone manual operations. The service scales to match your deployment needs.

CodeDeploy has several benefits that align with the DevOps principle of continuous deployment:

- **Automated deployments** — CodeDeploy fully automates software deployments, allowing you to deploy reliably and rapidly.
- **Centralized control** — CodeDeploy enables you to easily launch and track the status of your application deployments through the AWS Management Console or the AWS CLI. CodeDeploy gives you a detailed report enabling you to view when and to where each application revision was deployed. You can also create push notifications to receive live updates about your deployments.
- **Minimize downtime** — CodeDeploy helps maximize your application availability during the software deployment process. It introduces changes incrementally and tracks application health according to configurable rules. Software deployments can easily be stopped and rolled back if there are errors.
- **Easy to adopt** — CodeDeploy works with any application, and provides the same experience across different platforms and languages. You can easily reuse your existing setup code. CodeDeploy can also integrate with your existing software release process or continuous delivery toolchain (for example, AWS CodePipeline, GitHub, Jenkins).

AWS CodeDeploy supports multiple deployment options. For more information, refer to the *Deployment strategies* section of this document.

AWS CodePipeline

AWS CodePipeline is a continuous delivery service that you can use to model, visualize, and automate the steps required to release your software. With AWS CodePipeline, you model the full release process for building your code, deploying to pre-production environments, testing your application, and releasing it to production. AWS CodePipeline then builds, tests, and deploys your application according to the defined workflow every time there is a code change. You can integrate partner tools and your own custom tools into any stage of the release process to form an end-to-end continuous delivery solution.

AWS CodePipeline has several benefits that align with the DevOps principle of continuous deployment:

- **Rapid delivery** — AWS CodePipeline automates your software release process, allowing you to rapidly release new features to your users. With CodePipeline, you can quickly iterate on feedback and get new features to your users faster.

- **Improved quality** — By automating your build, test, and release processes, AWS CodePipeline enables you to increase the speed and quality of your software updates by running all new changes through a consistent set of quality checks.
- **Easy to integrate** — AWS CodePipeline can easily be extended to adapt to your specific needs. You can use the pre-built plugins or your own custom plugins in any step of your release process. For example, you can pull your source code from GitHub, use your on-premises Jenkins build server, run load tests using a third-party service, or pass on deployment information to your custom operations dashboard.
- **Configurable workflow** — AWS CodePipeline enables you to model the different stages of your software release process using the console interface, the AWS CLI, AWS CloudFormation, or the AWS SDKs. You can easily specify the tests to run and customize the steps to deploy your application and its dependencies.

Deployment strategies

Deployment strategies define how you want to deliver your software. Organizations follow different deployment strategies based on their business model. Some choose to deliver software that is fully tested, and others might want their users to provide feedback and let their users evaluate under development features (such as Beta releases). The following section discusses various deployment strategies.

Topics

In-place deployments

In this strategy, the previous version of the application on each compute resource is stopped, the latest application is installed, and the new version of the application is started and validated. This allows application deployments to proceed with minimal disturbance to underlying infrastructure. With an in-place deployment, you can deploy your application without creating new infrastructure; however, the availability of your application can be affected during these deployments. This approach also minimizes infrastructure costs and management overhead associated with creating new resources. You can use a load balancer so that each instance is deregistered during its deployment and then restored to service after the deployment is complete. In-place deployments can be all-at-once, assuming a service outage, or done as a rolling update. AWS CodeDeploy and AWS Elastic Beanstalkoffer deployment configurations for one-at-a-time, half-at-a-time, and all-at-once.

Blue/green deployment

Blue/green deployment, sometimes referred to as red/black deployment, is a technique for releasing applications by shifting traffic between two identical environments running differing versions of the application. Blue/green deployment helps you minimize downtime during application updates, mitigating risks surrounding downtime and rollback functionality.

Blue/green deployments enable you to launch a new version (green) of your application alongside the old version (blue), and monitor and test the new version before you reroute traffic to it, rolling back on issue detection.

Canary deployment

The purpose of a canary deployment is to reduce the risk of deploying a new version that impacts the workload. The method will incrementally deploy the new version, making it visible to new users in a slow fashion. As you gain confidence in the deployment, you will deploy it to replace the current version in its entirety.

Linear deployment

Linear deployment means traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.

All-at-once deployment

All-at-once deployment means all traffic is shifted from the original environment to the replacement environment all at once.

Deployment strategies matrix

The following matrix lists the supported deployment strategies for Amazon Elastic Container Service (Amazon ECS), AWS Lambda, and Amazon EC2/on-premises.

- Amazon ECS is a fully managed orchestration service.
- AWS Lambda lets you run code without provisioning or managing servers.
- Amazon EC2 enables you to run secure, resizable compute capacity in the cloud.

Deployment strategy	Amazon ECS	AWS Lambda	Amazon EC2/ on-premises
In-place	✓	✓	✓
Blue/green	✓	✓	✓*
Canary	✓	✓	X
Linear	✓	✓	X
All-at-once	✓	✓	X

Note

Blue/green deployment with EC2/on-premises works only with EC2 instances.

AWS Elastic Beanstalk deployment strategies

AWS Elastic Beanstalk supports the following type of deployment strategies:

- **All-at-once** Performs in place deployment on all instances.
- **Rolling** Splits the instances into batches and deploys to one batch at a time.
- **Rolling with additional batch** Splits the deployments into batches but for the first batch creates new EC2 instances instead of deploying on the existing EC2 instances.

- **Immutable** If you need to deploy with a new instance instead of using an existing instance.
- **Traffic splitting** Performs immutable deployment and then forwards percentage of traffic to the new instances for a pre-determined duration of time. If the instances stay healthy, then forward all traffic to new instances and shut down old instances.

Infrastructure as code

A fundamental principle of DevOps is to treat infrastructure the same way developers treat code. Application code has a defined format and syntax. If the code is not written according to the rules of the programming language, applications cannot be created. Code is stored in a version management or source control system that logs a history of code development, changes, and bug fixes. When code is compiled or built into applications, we expect a consistent application to be created, and the build is repeatable and reliable.

Practicing *infrastructure as code* means applying the same rigor of application code development to infrastructure provisioning. All configurations should be defined in a declarative way and stored in a source control system such as AWS CodeCommit, the same as application code. Infrastructure provisioning, orchestration, and deployment should also support the use of the infrastructure as code.

Infrastructure was traditionally provisioned using a combination of scripts and manual processes. Sometimes these scripts were stored in version control systems or documented step by step in text files or run-books. Often the person writing the run books is not the same person executing these scripts or following through the run-books. If these scripts or runbooks are not updated frequently, they can potentially become a show-stopper in deployments. This results in the creation of new environments not always being repeatable, reliable, or consistent.

In contrast, AWS provides a DevOps-focused way of creating and maintaining infrastructure. Similar to the way software developers write application code, AWS provides services that enable the creation, deployment and maintenance of infrastructure in a programmatic, descriptive, and declarative way. These services provide rigor, clarity, and reliability. The AWS services discussed in this paper are core to a DevOps methodology and form the underpinnings of numerous higher-level AWS DevOps principles and practices.

AWS offers the following services to define infrastructure as code.

Services

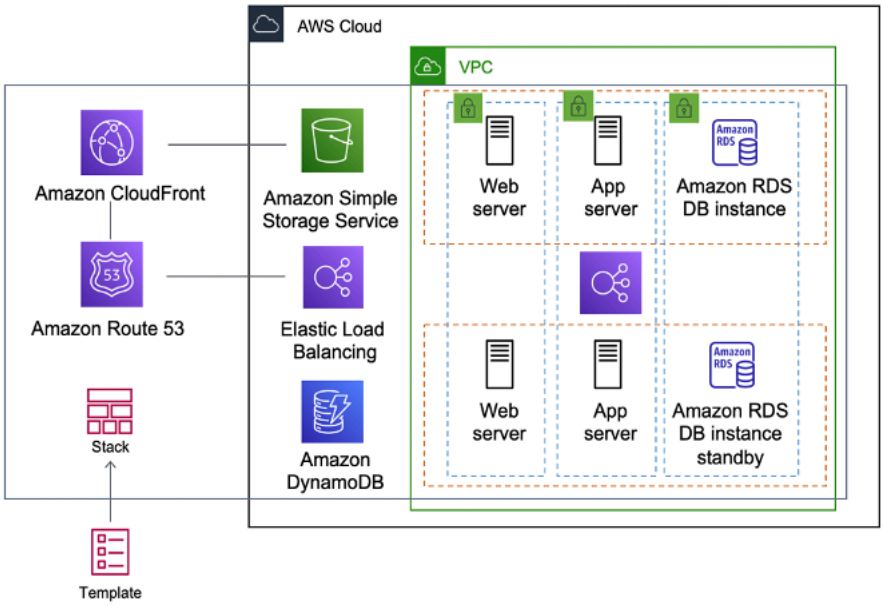
- AWS CloudFormation
- AWS Serverless Application Model
- AWS Cloud Development Kit (AWS CDK)
- AWS Cloud Development Kit for Kubernetes

- AWS Cloud Development Kit for Terraform
- AWS Cloud Control API

AWS CloudFormation

AWS CloudFormation is a service that enables developers to create AWS resources in an orderly and predictable fashion. Resources are written in text files using JSON or YAML format. The templates require a specific syntax and structure that depends on the types of resources being created and managed. You author your resources in JSON or YAML with any code editor such as AWS Cloud9, check it into a version control system, and then CloudFormation builds the specified services in safe, repeatable manner.

A CloudFormation template is deployed into the AWS environment as a stack. You can manage stacks through the AWS Management Console, AWS Command Line Interface, or AWS CloudFormation APIs. If you need to make changes to the running resources in a stack you update the stack. Before making changes to your resources, you can generate a change set, which is a summary of your proposed changes. Change sets enable you to see how your changes might impact your running resources, especially for critical resources, before implementing them.



AWS CloudFormation creating an entire environment (stack) from one template

You can use a single template to create and update an entire environment, or separate templates to manage multiple layers within an environment. This enables templates to be modularized, and also provides a layer of governance that is important to many organizations.

When you create or update a stack in the CloudFormation console, events are displayed, showing the status of the configuration. If an error occurs, by default the stack is rolled back to its previous state. Amazon SNS provides notifications on events. For example, you can use Amazon SNS to track stack creation and deletion progress using email and integrate with other processes programmatically.

AWS CloudFormation makes it easy to organize and deploy a collection of AWS resources, and lets you describe any dependencies or pass in special parameters when the stack is configured.

With CloudFormation templates, you can work with a broad set of AWS services, such as Amazon S3, Auto Scaling, Amazon CloudFront, Amazon DynamoDB, Amazon EC2, Amazon ElastiCache, AWS Elastic Beanstalk, Elastic Load Balancing, IAM, AWS OpsWorks, and Amazon VPC. For the most recent list of supported resources, refer to AWS resource and property types reference.

AWS Serverless Application Model

The AWS Serverless Application Model (AWS SAM) is an open-source framework that you can use to build serverless applications on AWS.

AWS SAM integrates with other AWS services, so creating serverless applications with AWS SAM provides the following benefits:

- **Single-deployment configuration** — AWS SAM makes it easy to organize related components and resources, and operate on a single stack. You can use AWS SAM to share configuration (such as memory and timeouts) between resources, and deploy all related resources together as a single, versioned entity.
- **Extension of AWS CloudFormation** — Because AWS SAM is an extension of AWS CloudFormation, you get the reliable deployment capabilities of AWS CloudFormation. You can define resources by using AWS CloudFormation in your AWS SAM template.
- **Built-in best practices** — You can use AWS SAM to define and deploy your IaC. This makes it possible for you to use and enforce best practices such as code reviews.

AWS Cloud Development Kit (AWS CDK)

The AWS Cloud Development Kit (AWS CDK) is an open source software development framework to model and provision your cloud application resources using familiar programming languages. AWS CDK enables you to model application infrastructure using TypeScript, Python, Java, and .NET.

Developers can leverage their existing Integrated Development Environment (IDE), using tools such as autocomplete and in-line documentation to accelerate development of infrastructure.

AWS CDK utilizes AWS CloudFormation in the background to provision resources in a safe, repeatable manner. Constructs are the basic building blocks of CDK code. A construct represents a cloud component and encapsulates everything AWS CloudFormation needs to create the component. The AWS CDK includes the AWS Construct Library, containing constructs representing many AWS services. By combining constructs together, you can quickly and easily create complex architectures for deployment in AWS.

AWS Cloud Development Kit for Kubernetes

AWS Cloud Development Kit for Kubernetes is an open-source software development framework for defining Kubernetes applications using general-purpose programming languages.

Once you have defined your application in a programming language (as of the date of this publication, only Python and TypeScript are supported), cdk8s will convert your application description in to pre-Kubernetes YAML. This YAML file can then be consumed by any Kubernetes cluster running anywhere. Because the structure is defined in a programming language, you can use the rich features provided by the programming language. You can use the abstraction feature of the programming language to create your own boilerplate code, and reuse it across all of the deployments.

AWS Cloud Development Kit for Terraform

Built on top of the open source JSII library, CDK for Terraform (CDKTF) allows you to write Terraform configurations in your choice of C#, Python, TypeScript, Java, or Go and still benefit from the full ecosystem of Terraform providers and modules. You can import any existing provider or module from the Terraform Registry into your application, and CDKTF will generate resource classes for you to interact with in your target programming language.

With CDKTF, developers can set up their IaC without context switching from their familiar programming language, using the same tooling and syntax to provision infrastructure resources similar to the application business logic. Teams can collaborate in familiar syntax, while still using the power of the Terraform ecosystem and deploying their infrastructure configurations via established Terraform deployment pipelines.

AWS Cloud Control API

AWS Cloud Control API is a new AWS capability that introduces a common set of Create, Read, Update, Delete, and List (CRUDL) APIs to help developers manage their cloud infrastructure in an easy and consistent way. The Cloud Control API common APIs allow developers to uniformly manage the lifecycle of AWS and third-party services.

As a developer, you might prefer to simplify the way you manage the lifecycle of all your resources. You can use Cloud Control API's uniform resource configuration model with a pre-defined format to standardize your cloud resource configuration. In addition, you will benefit from uniform API behavior (response elements and errors) while managing your resources.

For example, you will find it simple to debug errors during CRUDL operations through uniform error codes surfaced by Cloud Control API that are independent of the resources you operate on. Using Cloud Control API, you will also find it simple to configure cross-resource dependencies. You will also no longer require to author and maintain custom code across multiple vendor tools and APIs to use AWS and third-party resources together.

Automation and tooling

Another core philosophy and practice of DevOps is *automation*. Automation focuses on the setup, configuration, deployment, and support of infrastructure and the applications that run on it. By using automation, you can set up environments more rapidly in a standardized and repeatable manner. The removal of manual processes is key to a successful DevOps strategy. Historically, server configuration and application deployment have been predominantly a manual process. Environments become non-standard, and reproducing an environment when issues arise is difficult.

The use of automation is critical to realizing the full benefits of the cloud. Internally, AWS relies heavily on automation to provide the core features of elasticity and scalability.

Manual processes are error prone, unreliable, and inadequate to support an agile business. Frequently, an organization may tie up highly skilled resources to provide manual configuration, when time could be better spent supporting other, more critical, and higher value activities within the business.

Modern operating environments commonly rely on full automation to eliminate manual intervention or access to production environments. This includes all software releasing, machine configuration, operating system patching, troubleshooting, or bug fixing. Many levels of automation practices can be used together to provide a higher level end-to-end automated process.

Automation has the following key benefits:

- Rapid changes
- Improved productivity
- Repeatable configurations
- Reproducible environments
- Elasticity
- Automatic scaling
- Automated testing

Automation is a cornerstone with AWS services and is internally supported in all services, features, and offerings.

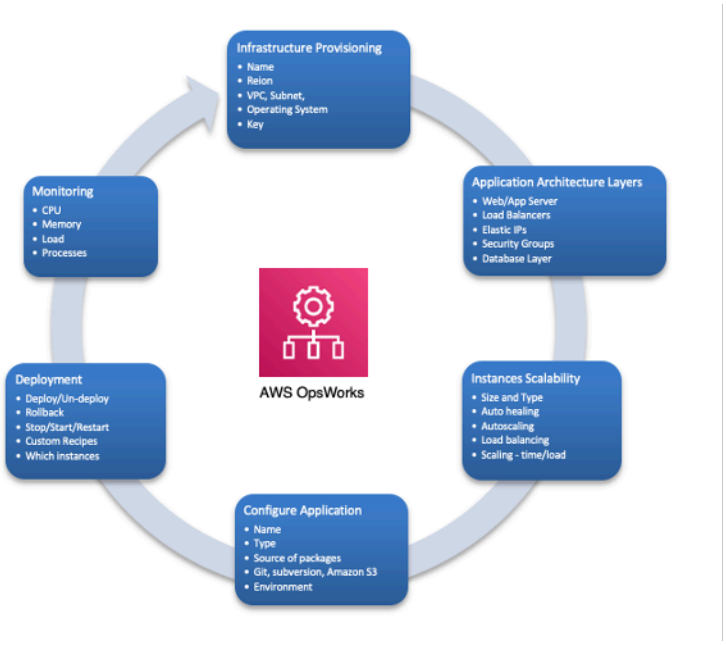
Topics

AWS OpsWorks

AWS OpsWorks takes the principles of DevOps even further than AWS Elastic Beanstalk. It can be considered an application management service rather than simply an application container. AWS OpsWorks provides even more levels of automation, with additional features such as integration with configuration management software (Chef) and application lifecycle management. You can use application lifecycle management to define when resources are set up, configured, deployed, un-deployed, or ended.

For added flexibility AWS OpsWorks has you define your application in configurable stacks. You can also select predefined application stacks. Application stacks contain all the provisioning for AWS resources that your application requires, including application servers, web servers, databases, and load balancers.

Application stacks are organized into architectural layers so that stacks can be maintained independently. Example layers could include web tier, application tier, and database tier. Out of the box, AWS OpsWorks also simplifies setting up AWS Auto Scaling groups and Elastic Load Balancing (ELB) load balancers, further illustrating the DevOps principle of automation. Just like AWS Elastic Beanstalk, AWS OpsWorks supports application versioning, continuous deployment, and infrastructure configuration management



AWS OpsWorks showing DevOps features and architecture

AWS OpsWorks also supports the DevOps practices of monitoring and logging (covered in the next section). Monitoring support is provided by Amazon CloudWatch. All lifecycle events are logged, and a separate Chef log documents any Chef recipes that are run, along with any exceptions.

AWS Elastic Beanstalk

AWS Elastic Beanstalk is a service to rapidly deploy and scale web applications developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, NGINX, Passenger, and IIS.

Elastic Beanstalk is an abstraction on top of Amazon EC2, Auto Scaling, and simplifies the deployment by giving additional features such as cloning, blue/green deployments, Elastic Beanstalk Command Line Interface (EB CLI) and integration with AWS Toolkit for Visual Studio, Visual Studio Code, Eclipse, and IntelliJ for increase developer productivity.

EC2 Image Builder

EC2 Image Builderis a fully managed AWS service that helps you to automate the creation, maintenance, validation, sharing, and deployment of customized, secure, and up-to-date Linux or Windows custom AMI. EC2 Image Builder can also be used to create container images. You can use the AWS Management Console, the AWS CLI, or APIs to create custom images in your AWS account.

EC2 Image Builder significantly reduces the effort of keeping images up-to-date and secure by providing a simple graphical interface, built-in automation, and AWS-provided security settings. With EC2 Image Builder, there are no manual steps for updating an image nor do you have to build your own automation pipeline.

AWS Proton

AWS Proton enables platform teams to connect and coordinate all the different tools your development teams need for infrastructure provisioning, code deployments, monitoring, and updates. AWS Proton enables automated infrastructure as code provisioning and deployment of serverless and container-based applications.

AWS Proton enables platform teams to define their infrastructure and deployment tools, while providing developers with a self-service experience to get infrastructure and deploy code. Through AWS Proton, platform teams provision shared resources and define application stacks, including CI/CD pipelines and observability tools. You can then manage which infrastructure and deployment features are available for developers.

AWS Service Catalog

AWS Service Catalog enables organizations to create and manage catalogs of IT services that are approved for AWS. These IT services can include everything from virtual machine images, servers, software, databases, and more to complete multi-tier application architectures. AWS Service Catalog lets you centrally manage deployed IT services, applications, resources, and metadata to achieve consistent governance of your IaC templates.

With AWS Service Catalog, you can meet your compliance requirements while making sure your customers can quickly deploy the approved IT services they need. End users can quickly deploy only the approved IT services they need, following the constraints set by your organization.

AWS Cloud9

AWS Cloud9 is a cloud-based IDE that lets you write, run, and debug your code with just a browser. It includes a code editor, debugger, and terminal. AWS Cloud9 comes prepackaged with essential tools for popular programming languages, including JavaScript, Python, PHP, and more, so you don't need to install files or configure your development machine to start new projects. Because your AWS Cloud9 IDE is cloud-based, you can work on your projects from your office, home, or anywhere using an internet-connected machine.

AWS CloudShell

AWS CloudShell is a browser-based shell that makes it easier to securely manage, explore, and interact with your AWS resources. AWS CloudShell is pre-authenticated with your console credentials. Common development and operations tools are pre-installed, so there's no need to install or configure software on your local machine.

Amazon CodeGuru

Amazon CodeGuru is a developer tool that provides intelligent recommendations to improve code quality and identify an application's most expensive lines of code. Integrate CodeGuru into your existing software development workflow to automate code reviews during application development and continuously monitor application's performance in production and provide recommendations and visual clues on how to improve code quality, application performance, and reduce overall cost. CodeGuru has two components:

- **Amazon CodeGuru Reviewer** — Amazon CodeGuru Reviewer is an automated code review service that identifies critical defects and deviation from coding best practices for Java and Python code. It scans the lines of code within a pull request and provides intelligent recommendations based on standards learned from major open-source projects as well as Amazon codebase.
- **Amazon CodeGuru Profiler** — Amazon CodeGuru Profiler analyzes the application runtime profile and provides intelligent recommendations and visualizations that guide developers on how to improve the performance of the most relevant parts of their code.

Monitoring and observability

Communication and collaboration are fundamental in a DevOps philosophy. To facilitate this, feedback is critical. This feedback is provided by our suite of monitoring and observability services.

AWS provides the following services for monitoring and logging:

Topics

- Amazon CloudWatch metrics
- Amazon CloudWatch Alarms
- Amazon CloudWatch Logs
- Amazon CloudWatch Logs Insights
- Amazon CloudWatch Events
- Amazon EventBridge
- AWS CloudTrail
- Amazon DevOps Guru
- AWS X-Ray
- Amazon Managed Service for Prometheus
- Amazon Managed Grafana

Amazon CloudWatch metrics

Amazon CloudWatch metrics automatically collect data from AWS services such as Amazon EC2 instances, Amazon EBS volumes, and Amazon RDS database (DB) instances. These metrics can then be organized as dashboards and alarms or events can be created to trigger events or perform Auto Scaling actions.

Amazon CloudWatch Alarms

You can set up alarms using Amazon CloudWatch alarms based on the metrics collected by Amazon CloudWatch metrics. The alarm can then send a notification to Amazon SNS topic, or initiate Auto Scaling actions. An alarm requires period (length of the time to evaluate a metric), evaluation

period (number of the most recent data points), and datapoints to alarm (number of data points within the evaluation period).

Amazon CloudWatch Logs

Amazon CloudWatch Logs is a log aggregation and monitoring service. AWS CodeBuild, CodeCommit, CodeDeploy and CodePipeline provide integrations with CloudWatch logs so that all of the logs can be centrally monitored. In addition, the previously mentioned services various other AWS services provide direct integration with CloudWatch.

With CloudWatch Logs you can:

- Query your log data
- Monitor logs from Amazon EC2 instances
- Monitor AWS CloudTrail logged events
- Define log retention policy

Amazon CloudWatch Logs Insights

Amazon CloudWatch Logs Insights scans your logs and enables you to perform interactive queries and visualizations. It understands various log formats and auto-discovers fields from JSON logs.

Amazon CloudWatch Events

Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. Using simple rules that you can quickly set up, you can match events and route them to one or more target functions or streams.

CloudWatch Events becomes aware of operational changes as they occur. CloudWatch Events responds to these operational changes and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.

You can configure rules in Amazon CloudWatch Events to alert you to changes in AWS services and integrate these events with other third-party systems using Amazon EventBridge. The following are the AWS DevOps related services that have integration with CloudWatch Events.

- Application Auto Scaling Events

- CodeBuild Events
- CodeCommit Events
- CodeDeploy Events
- CodePipeline Events

Amazon EventBridge

> **Note**
>
> Amazon CloudWatch Events and EventBridge are the same underlying service and API, however, EventBridge provides more features.

Amazon EventBridgeis a serverless event bus that enables integrations between AWS services, Software as a services (SaaS), and your applications. In addition to build event driven applications, EventBridge can be used to notify about the events from the services such as CodeBuild, CodeDeploy, CodePipeline, and CodeCommit.

AWS CloudTrail

To embrace the DevOps principles of collaboration, communication, and transparency, it's important to understand who is making modifications to your infrastructure. In AWS, this transparency is provided by AWS CloudTrail. All AWS interactions are handled through AWS API calls that are monitored and logged by AWS CloudTrail. All generated log files are stored in an Amazon S3 bucket that you define. Log files are encrypted using Amazon S3 server-side encryption (SSE). All API calls are logged whether they come directly from a user or on behalf of a user by an AWS service. Numerous groups can benefit from CloudTrail logs, including operations teams for support, security teams for governance, and finance teams for billing.

Amazon DevOps Guru

Amazon DevOps Guru is a service powered by machine learning (ML) that is designed to make it easy to improve an application's operational performance and availability. DevOps Guru helps detect behaviors that deviate from normal operating patterns, so you can identify operational issues long before they impact your customers.

DevOps Guru uses ML models informed by years of Amazon.com and AWS operational excellence to help identify anomalous application behavior (for example, increased latency, error rates, resource constraints, and others) and surface critical issues that could cause potential outages or service disruptions.

When DevOps Guru identifies a critical issue, it saves debugging time by fetching relevant and specific information from a large number of data sources and automatically sends an alert and provides a summary of related anomalies, and context for when and where the issue occurred.

AWS X-Ray

AWS X-Ray helps developers analyze and debug production, distributed applications, such as those built using a microservices architecture. With X-Ray, you can understand how your application and its underlying services are performing to identify and troubleshoot the root cause of performance issues and errors. X-Ray provides an end-to-end view of requests as they travel through your application, and shows a map of your application's underlying components. X-Ray makes it easy for you to:

- **Create a service map** – By tracking requests made to your applications, X-Ray can create a map of services used by your application. This provides you with a view of connections among services in your application, and enables you to create a dependency tree, detect latency or errors when working across AWS Availability Zones or Regions, zero in on services not operating as expected, and so on.
- **Identify errors and bugs** – X-Ray can automatically highlight bugs or errors in your application code by analyzing the response code for each request made to your application. This enables easy debugging of application code without requiring you to reproduce the bug or error.
- **Build your own analysis and visualization apps** – X-Ray provides a set of query APIs you can use to build your own analysis and visualizations apps that use the data that X-Ray records.

Amazon Managed Service for Prometheus

Amazon Managed Service for Prometheus is a serverless monitoring service for metrics compatible with open-source Prometheus, making it easier for you to securely monitor and alert on container environments. Amazon Managed Service for Prometheus reduces the heavy lifting required to get started with monitoring applications across Amazon Elastic Kubernetes Service, Amazon Elastic Container Service, and AWS Fargate, as well as self-managed Kubernetes clusters.

Amazon Managed Grafana

Amazon Managed Grafana is a fully managed service with rich, interactive data visualizations to help customers analyze, monitor, and alarm on metrics, logs, and traces across multiple data sources. You can create interactive dashboards and share them with anyone in your organization with an automatically scaled, highly available, and enterprise-secure service.

Communication and Collaboration

Whether you are adopting DevOps Culture in your organization or going through a DevOps cultural transformation, communication and collaboration are an important part of your approach. At Amazon, we have realized that there was a need to bring a change to the mindset of our teams and thus adopted the concept of *Two-Pizza Teams*.

Topics

Two-Pizza Teams

"We try to create teams that are no larger than can be fed by two pizzas," said Bezos. "We call that the two-pizza team rule."

The smaller the team, the better the collaboration. Collaboration is very important, as software releases are moving faster than ever. And a team's ability to deliver the software can be a differentiating factor for your organization against your competition. Imagine a situation in which a new product feature needs to be released or a bug needs to be fixed. You want this to happen as quickly as possible, so you can have a smaller go-to-market timed. You don't want the transformation to be a slow-moving process; you want an agile approach where waves of changes start to make an impact.

Communication between teams is also important as you move toward the shared responsibility model and start moving out of the siloed development approach. This brings the concept of ownership to the team, and shifts their perspective to look at the process as an end-to-end venture. Your team should not think about your production environments as black boxes where they have no visibility.

Cultural transformation is also important, because you might be building a common DevOps team or have a DevOps focused member in your team. Both of these approaches introduce Shared Responsibility in to the team.

AWS CodeStar

AWS CodeStar provides the tools you need to quickly develop, build, and deploy applications on AWS. With AWS CodeStar, you can use a variety of project templates to start developing applications on Amazon EC2, AWS Lambda, and AWS Elastic Beanstalk. AWS CodeStar allows you to accelerate application delivery by providing a pre-configured continuous delivery toolchain for developing, building, testing, and deploying your projects on AWS.

The project dashboard in AWS CodeStar makes it easy to centrally monitor application activity and manage day-to-day development tasks such as recent code commits, builds, and deployments. Because AWS CodeStar integrates with Atlassian JIRA, a third-party issue tracking and project management tool, you can create and manage JIRA issues in the AWS CodeStar dashboard.

Security

Whether you are going through a DevOps transformation or implementing DevOps principles for the first time, you should think about Security as integrated in your DevOps processes. This should be cross cutting concern across your build, test deployment stages.

Before exploring security in DevOps on AWS, this paper looks at the AWS Shared Responsibility Model.

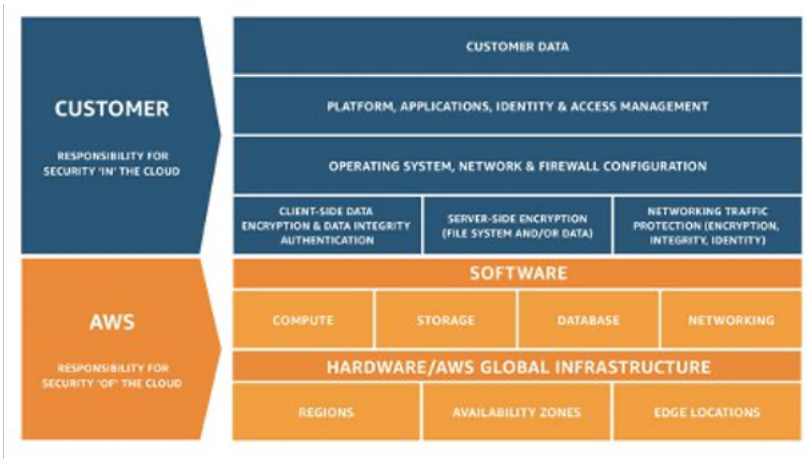
Topics

AWS Shared Responsibility Model

Security is a shared responsibility between AWS and the customer. The different parts of the Shared Responsibility Model are:

- **AWS responsibility "Security of the Cloud"** - AWS is responsible for protecting the infrastructure that runs all of the services offered in the AWS Cloud. This infrastructure is composed of the hardware, software, networking, and facilities that run AWS Cloud services.
- **Customer responsibility "Security in the Cloud"** – Customer responsibility is determined by the AWS Cloud services that a customer selects. This determines the amount of configuration work the customer must perform as part of their security responsibilities.

This shared model can help relieve the customer's operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the service operates. This is critical in the cases where customer want to understand the security of their build environments.



AWS Shared Responsibility Model

For DevOps, assign permissions based on the least-privilege permissions model. This model states that “a user (or service)should have the exact access rights necessary to complete their role's responsibilities—no more, no less. ”.

Permissions are maintained in IAM. You can use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

Identity and Access Management

AWS Identity and Access Management (IAM) defines the controls and polices that are used to manage access to AWS resources. Using IAM you can create users and groups and define permissions to various DevOps services.

In addition to the users, various services may also need access to AWS resources. For example, your CodeBuild project might need access to store Docker images in Amazon Elastic Container Registry (Amazon ECR) and need permissions to write to Amazon ECR. These types of permissions are defined by a special type role know as service role.

IAM is one component of the AWS security infrastructure. With IAM, you can centrally manage groups, users, service roles and security credentials such as passwords, access keys, and permissions policies that control which AWS services and resources users can access. IAM Policy lets you define the set of permissions. This policy can then be attached to either a role, user, or a service to define their permission.

You can also use IAM to create roles that are used widely within your desired DevOps strategy. In some cases, it can make perfect sense to programmatically AssumeRole instead of directly getting

the permissions. When a service or user assumes roles, they are given temporary credentials to access a service that they normally don't have access to.

Conclusion

To make the journey to the cloud smooth, efficient, and effective, technology companies should embrace DevOps principles and practices. These principles are embedded in AWS, and form the cornerstone of numerous AWS services, especially those in the deployment and monitoring offerings.

Begin by defining your infrastructure as code using the service AWS CloudFormation or AWS CDK. Next, define the way in which your applications are going to use continuous deployment with the help of services like AWS CodeBuild, AWS CodeDeploy, AWS CodePipeline, and AWS CodeCommit. At the application level, use containers such as AWS Elastic Beanstalk, Amazon ECS, or Amazon Elastic Kubernetes Service (Amazon EKS). Use AWS OpsWorks to simplify the configuration of common architectures. Using these services also makes it easy to include other important services such as Auto Scaling and Elastic Load Balancing.

Finally, use the DevOps strategy of monitoring such as Amazon CloudWatch, and solid security practices such as IAM.

With AWS as your partner, your DevOps principles bring agility to your business and IT organization and accelerate your journey to the cloud.

Document Revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Updated	Updated	April 7, 2023
Updated sections to include new services	Updated sections to include new services	October 16, 2020
Initial publication	Whitepaper first published	December 1, 2014

Contributors

Contributors to this document include:

- Abhra Sinha, Solutions Architect
- Anil Nadiminti, Solutions Architect
- Muhammad Mansoor, Solutions Architect
- Ajit Zadgaonkar, World Wide Tech Leader, Modernization
- Juan Lamadrid, Solutions Architect
- Darren Ball, Solutions Architect
- Rajeswari Malladi, Solutions Architect
- Pallavi Nargund, Solutions Architect
- Bert Zahniser, Solutions Architect
- Abdullahi Olaoye, Cloud Solutions Architect
- Mohamed Kiswani, Software Development Manager
- Tara McCann, Manager, Solutions Architect

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.