Introduction to DevOps on AWS

AWS Whitepaper

Introduction to DevOps on AWS: AWS Whitepaper

Table of Contents

Introduction to DevOps on AWS

Publication date: **October 16, 2020** (*Document Revisions* (p. 21))

Abstract

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 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, engineering practices and patterns, and tools that 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, Continuous Delivery, Infrastructure as Code, 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. We also highlight how to build a continuous integration and delivery capability without managing servers or build nodes, and how to leverage Infrastructure as Code to provision and manage your cloud resources in a consistent and repeatable manner.

- **Continuous Integration:** is 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:** is a software development practice where code changes are automatically built, tested, and prepared for a release to production.
- **Infrastructure as Code:** is 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 Amazon Web Services (AWS).

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 CodeCommit is a secure, highly scalable, managed source control service that hosts private git repositories. CodeCommit eliminates 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 Git, 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 enabling you to 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 triggers, 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 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 are increasingly relying on open-source packages to perform common tasks in their application package. It has now become critical for the software development teams to maintain control on a particular version of the open-source software is vulnerability free. With CodeArtifact, you can set up controls to enforce the preceding.

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

Continuous Delivery

Continuous delivery 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 may include UI testing, load testing, integration testing, API reliability testing, etc. This helps developers more thoroughly validate updates and preemptively discover issues. With 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 (p. 3)
- AWS CodeDeploy (p. 5)
- AWS CodePipeline (p. 6)

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, eliminating 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, see Deployment Strategies (p. 7).

AWS CodePipeline

AWS CodePipeline is a continuous delivery service that enables you 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 APN 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 may choose to deliver software that is fully tested, and others may want their users to provide feedback and let their users evaluate under development features (for example, beta releases). In the following section we will talk about various deployment strategies.

Topics

In-Place Deployments

In this strategy, the deployment is done line with the application on each instance in the deployment group is stopped, the latest application revision is installed, and the new version of the application is started and validated. 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 Beanstalk offer deployment configurations for one at a time, half at a time and all at once. These same deployment strategies for in-place deployments are available within blue/green deployments.

Blue/Green Deployments

Blue/green, sometimes referred to as red-black, deployment is a technique for releasing applications by shift traffic between two identical environments running differing versions of the application. Blue/green deployments help 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 Deployments

Traffic is shifted in two increments. A canary deployment is a blue/green strategy that is more risk-averse, in which a phased approach is used. This can be two step or linear in which new application code is deployed and exposed for trial, and upon acceptance rolled out either to the rest of the environment or in a linear fashion.

Linear Deployments

Linear deployments mean that 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 Deployments

All-at-once deployments mean that 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-Premise.

- 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.

	A	B	C	D
1	Deployment Strategies Matrix	Amazon ECS	AWS Lambda	Amazon EC2/On-Premise
2	In-Place	✓	✓	✓
3	Blue/Green	✓	✓	✓*
4	Canary	✓	✓	X
5	Linear	✓	✓	X
6	All-at-Once	✓	✓	X

Note
Blue/green deployment with EC2/On-Premise only works 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 to the preceding, 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 following services to define Infrastructure as a code.

- AWS CloudFormation (p. 10)
- AWS Cloud Development Kit (CDK) (p. 11)
- AWS Cloud Development Kit for Kubernetes (p. 12)

AWS CloudFormation

AWS CloudFormation is a service that enables developers create AWS resources in an orderly and predictable fashion. Resources are written in text files using JavaScript Object Notation (JSON) or Yet Another Markup Language (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.

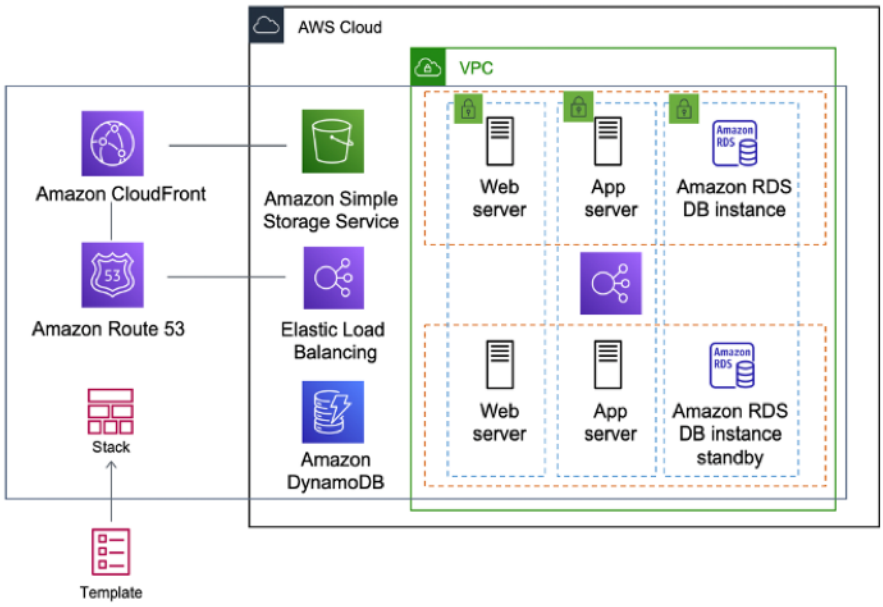

Figure 1 - AWS CloudFormation creating an entire environment (stack) from one template workflow

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 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 Simple Notification Service (Amazon SNS) provides notifications on events. For example, you can use Amazon SNS to track stack creation and deletion progress via 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, see AWS resource and property types reference.

AWS Cloud Development Kit

The AWS Cloud Development Kit (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), leveraging tools like 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 (cdk8s), 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 date of publication only Python and TypeScript are supported) cdk8s will convert your application description in to pre-Kubernetes YML. This YML 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 boiler-plate code and re-use it across all of the deployments.

Automation

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 a key to a successful DevOps strategy. Historically, server configuration and application deployment have been predominantly a manual process. Environments become nonstandard, 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
- Leveraged elasticity
- Leveraged 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 like 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, undeployed, or shut down.

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.

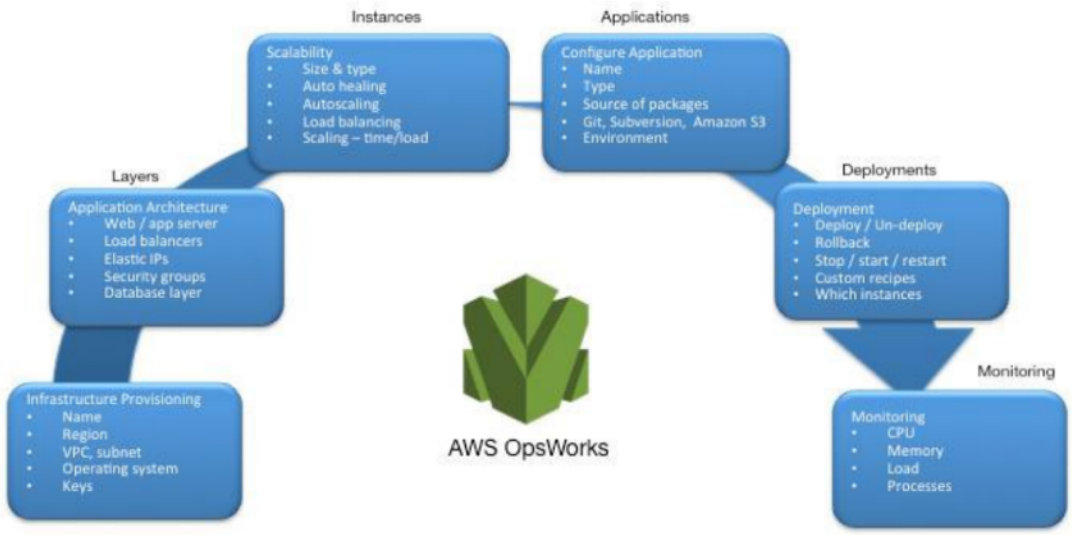

Figure 2 - AWS OpsWorks showing DevOps features and architecture

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 Auto Scaling groups and Elastic Load Balancing 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 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.

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 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

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

In order 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 service. 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.

Communication and Collaboration

Whether you are adopting DevOps Culture in your organization or going through a DevOps cultural transformation communication, and collaboration is an important part of you approach. At Amazon, we have realized that there is need to bring a changed in the mindset of the teams and hence 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 also very important as the 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. Image 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. This is also important as you don't want the transformation to be a slow-moving process rather than an agile approach where waves of changes start to make an impact.

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

Cultural transformation is also important as you may be building a common DevOps team or the other approach is that you have one or more DevOps-focused members on your team. Both of these approaches do introduce shared responsibility in to the team.

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 we talk about Security in DevOps on AWS let's look 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 explained below:

- **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.

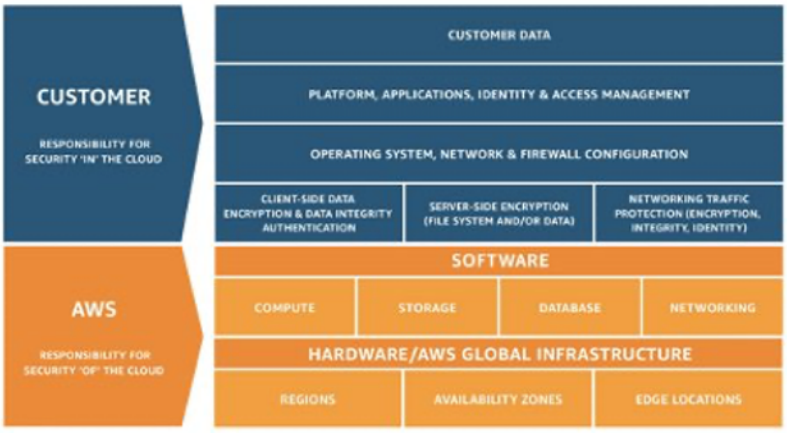

Figure 3 - AWS Shared Responsibility Model

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 you normally don't have access.

Conclusion

In order to make the journey to the cloud smooth, efficient, and effective, technology companies should embrace DevOps principles and practices. These principles are embedded in the AWS platform. Indeed, they 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 Cloud Development Kit (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 like AWS Elastic Beanstalk, Amazon Elastic Container Service (Amazon ECS), or Amazon Elastic Kubernetes Service (Amazon EKS), and AWS OpsWorks to simplify the configuration of common architectures. Using these services also makes it easy to include other important services like Auto Scaling and Elastic Load Balancing. Finally, use the DevOps strategy of monitoring such as Amazon CloudWatch, and solid security practices such as AWS 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.

update-history-change	update-history-description	update-history-date
Restored missing Contributors section (p. 22)	Restored missing Contributors section and minor text changes	November 21, 2020
Updated sections to include new services (p. 21)	Updated sections to include new services	October 16, 2020
Initial publication (p. 21)	Whitepaper first published	December 1, 2014

Contributors

Contributors to this document include:

- 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