Learning Outcome

## Able to Build a web application on modern cloud-based architectures and services

## 

## 

# Understanding DevOps, tools, DevOps services in Cloud

## DevOps

DevOps speeds delivery of higher quality software by combining and automating the work of software development and IT operations teams.

Let’s go to the good old days. Once in a time again, there are two separate teams, developers and operators. Developers develop an application. And, operators deploy this application to servers and maintain it on a server. These days, everyone was happy. After the application is developed and deployed, there is no need to update much. It would take too long to adapt to changes and make updates on our software. As a result of this, developers and operators are two different teams. But time changes like everything. The change is inevitable. In our modern times, we witness those applications are constantly updated. So, it leads the IT companies to a new culture, called DevOps.

### What is DevOps?

DevOps (a portmanteau of “development” and “operations”) is the combination of practices and tools designed to increase an organization’s ability to deliver applications and services faster than traditional software development processes. This speed enables organizations to better serve their customers and compete more effectively in the market.

In simple terms, DevOps is about removing the barriers between traditionally siloed teams, development and operations. Under a DevOps model, development and operations teams work together across the entire software application life cycle, from development and test through deployment to operations.



Image 1: DevOps

Reference: <https://aws.amazon.com/devops/what-is-devops/>

### DevOps Lifecycle

DevOps focuses on bringing all the development, operations, and IT infrastructure guys, including Developers, Testers, System Admins, and QAs, under one roof. Hence, all these people together are called DevOps Engineers.

DevOps Engineers share the end-to-end responsibility of gathering information, setting up the infrastructure, developing, testing, deploying, continuous monitoring, and fetching feedback from end-users. This process of developing, testing, deploying, and monitoring keeps on repeating for better results.

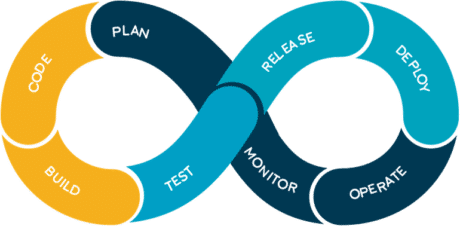


Image 2: DevOps Lifecycle

Reference: <https://aws.amazon.com/devops/what-is-devops/>

**Code:**

The first step in the DevOps life cycle is coding, where developers build the code on any platform

**Build:**

Developers build the version of their program in any extension depending upon the language they are using

**Test:**

For DevOps to be successful, the testing process must be automated using any automation tool like Selenium

**Release:**

A process for managing, planning, scheduling, and controlling the build in different environments after testing and before deployment

**Deploy:**

This phase gets all artifacts/code files of the application ready and deploys/executes them on the server

**Operate:**

The application is run after its deployment, where clients use it in real-world scenarios.

**Monitor:**

This phase helps in providing crucial information that basically helps ensure service uptime and optimal performance

**Plan:**

The planning stage gathers information from the monitoring stage and, as per feedback, implements the changes for better performance

### Different Lifecycle Stages



Image 3: Stages of DevOps Lifecycle

Reference: <https://intellipaat.com/blog/what-is-devops-lifecycle-and-process/>

The DevOps lifecycle provides a structure to the project in such a way that it gives the team or the person working on it a view of what comes next. By following this approach, one can develop a quality project in very little time and with high reliability. One cannot simply say that he/she knows DevOps without knowing its lifecycle. Here are the various stages of the DevOps lifecycle

* Continuous Development
* Continuous Integration
* Continuous Testing
* Continuous Monitoring
* Continuous Feedback
* Continuous Deployment
* Continuous Operations

**Continuous Development**

The continuous development phase involves planning and coding the product the team is developing. In this phase of the DevOps lifecycle, the vision and goal of the project are set, and developers start to code. The integration of development and operations teams helps in planning the work accordingly, increasing the productivity of the team. **In this phase, they use tools, such as Git, CVS, Slack, etc.**

Before DevOps, the concept of the cloud was in just its initial stages, and companies had to use fixed hardware and software allocations they had planned for the project. Now, with cloud services in place, they can plan to increase or decrease the resource allocation for the project using cloud resources within their budget.

With the adaptation of DevOps, there occurred an increase in the usage of good coding methodologies and versioning systems. Take Git, for example. Using Git and its commands, users can maintain version control for keeping track of the changes made to a set of files so that, whenever the newer versions have serious bugs or critical vulnerabilities in them, the team can revert to previous versions.

Similarly, Slack, Skype, and more recently Zoom are used for communication between the team members where they can send messages directly or hold virtual meetings to keep track of the progress of the project.



Image 4: Continuous Development

Reference: <https://intellipaat.com/blog/what-is-devops-lifecycle-and-process/>

**Continuous Integration**

In the continuous integration phase, the source code in the central repository is regularly updated by developers. This phase not only involves compilation but also unit testing, integration testing, code review, and the packaging of the code written by the developers. **The tools used in this DevOps process are Jenkins, GitLab, etc.**

Jenkins orchestrates a chain of actions that helps it achieve the continuous integration process in an automated fashion. It is a server-based application and uses servers like Apache Tomcat. The reason why it is used so much is it monitors the repeated tasks that arise during the development of a project and continually tests the builds to show errors, if any, in the early stages of development itself.



Image 5: Continuous Integration

Reference: <https://intellipaat.com/blog/what-is-devops-lifecycle-and-process/>

**Continuous Testing**

In the continuous testing phase, the code written by developers is sent to testers where they use automated tools to test it for bugs. The beauty of this phase is that they can schedule to run the tests automatically at a predefined time. The report generated in this phase is sent back to the developers where they make necessary updates to the code to remove the bugs.

**The tools used in this DevOps process are JUnit (to test the Java code), Selenium, and Docker to simulate a test environment in a container so that the rest of the code is not disturbed.**

Selenium is an automated testing framework used to validate applications across various browsers and platforms. You can create Selenium test cases using various programming languages, such as Java, Python, C#, etc. It is not just a single tool but a suite of software where each piece is used for the different QA testing needs of an organization.

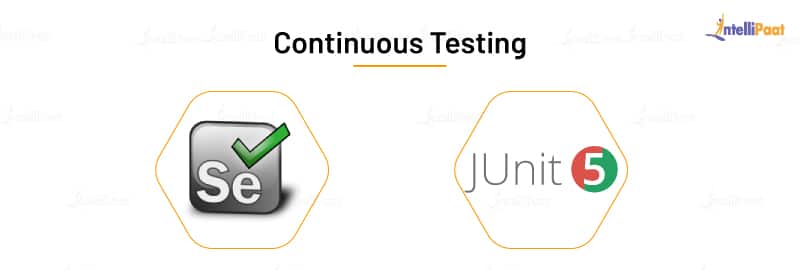


Image 6: Continuous Testing

Reference: <https://intellipaat.com/blog/what-is-devops-lifecycle-and-process/>

**Continuous Monitoring**

The continuous monitoring phase involves monitoring the health, performance, and reliability of the application or code, as well as the infrastructure, as the phases move from development to deployment. **The tools used in this phase are Nagios, Sensu, etc.**

Nagios is a platform that tracks the infrastructure, networks, and systems. It monitors and alerts services for servers, switches, software, etc. If there is a problem, it warns the users and notifies them again when the problem gets solved.

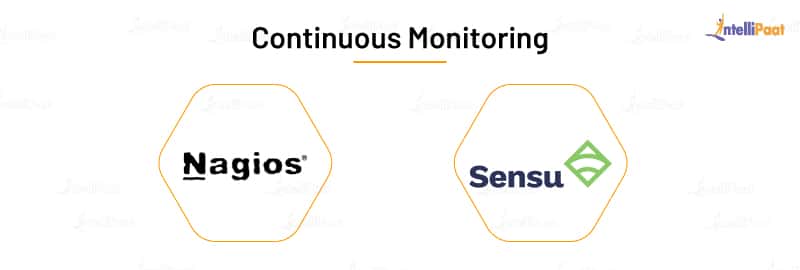


Image 7: Continuous Monitoring

Reference: <https://intellipaat.com/blog/what-is-devops-lifecycle-and-process/>

**Continuous Feedback**

In the continuous feedback phase of the DevOps lifecycle, the evaluation of the effect of each release on the user experience takes place, and this evaluation is reported back to the team to improve the future releases.

The feedback can be gathered in two methods: structured and unstructured. The structured method is applied through surveys, questionnaires, and focus groups. The unstructured feedback collection is done through social media platforms such as Twitter, Facebook, etc. Here, the users take part in this DevOps process by providing their feedback, just like how users provide app reviews on Google Playstore.

In this phase, the team uses **Pendo,** which is a product-analytics tool that helps organizations get customer views. It gives user insight, user guidance, user sentiment, and user feedback to an organization to know what its users want or what they are expecting.



Image 8: Continuous Feedback

Reference: <https://intellipaat.com/blog/what-is-devops-lifecycle-and-process/>

**Continuous Deployment**

In the continuous deployment phase of the DevOps lifecycle, the application is deployed on the production server to make it available for the intended users. **The tools used in this phase are AWS CodeDeploy, Octopus Deploy, Jenkins, etc.**

AWS CodeDeploy is a software deployment service that automates deployment to a variety of services. It makes it easier for organizations to rapidly release new features, avoiding downtime during the deployment, and it handles the complexity of the deployment process. It automates software deployment, eliminating the need for error-prone manual operations. It also scales the resources to match deployment needs.

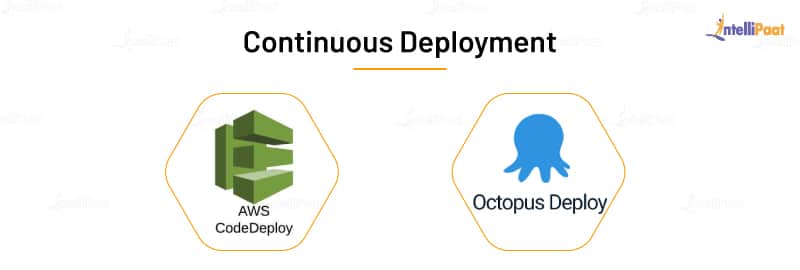


Image 9: Continuous Deployment

Reference: <https://intellipaat.com/blog/what-is-devops-lifecycle-and-process/>

**Continuous Operations**

The continuous operations phase involves the reduction or elimination of planned downtime like scheduled maintenance. The goal of this phase is to increase the uptime or the time the users can use the application. Companies use container management systems like Kubernetes or Swarm in this phase.

When developers want to make updates to the production server, usually they have to take it offline and make changes to it. This would increase the downtime of the software bringing loss to the organization.

To decrease and eliminate that downtime, they can use Kubernetes. They take a container with the software managed by Kubernetes and make the necessary changes to it, while Kubernetes runs another container containing the current version of the software. When the team deploys the software with changes, **Kubernetes** make those changes to all the containers present in the server without the team manually doing it.

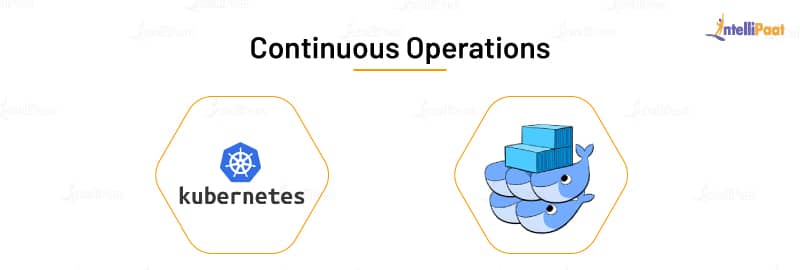


Image 10: Continuous Operations

Reference: <https://intellipaat.com/blog/what-is-devops-lifecycle-and-process/>

**DevOps lifecycle in IBMCloud**

The DevOps lifecycle (sometimes called the continuous delivery pipeline, when portrayed in a linear fashion) is a series of iterative, automated development processes, or workflows, executed within a larger, automated and iterative development lifecycle designed to optimize the rapid delivery of high-quality software. The name and number of workflows can differ depending on whom you ask, but they typically boil down to these six:

* Planning
* Development
* Integration (or build, or continuous Integration and continuous delivery (CI/CD)
* Deployment (usually called continuous deployment)
* Operations
* Learning

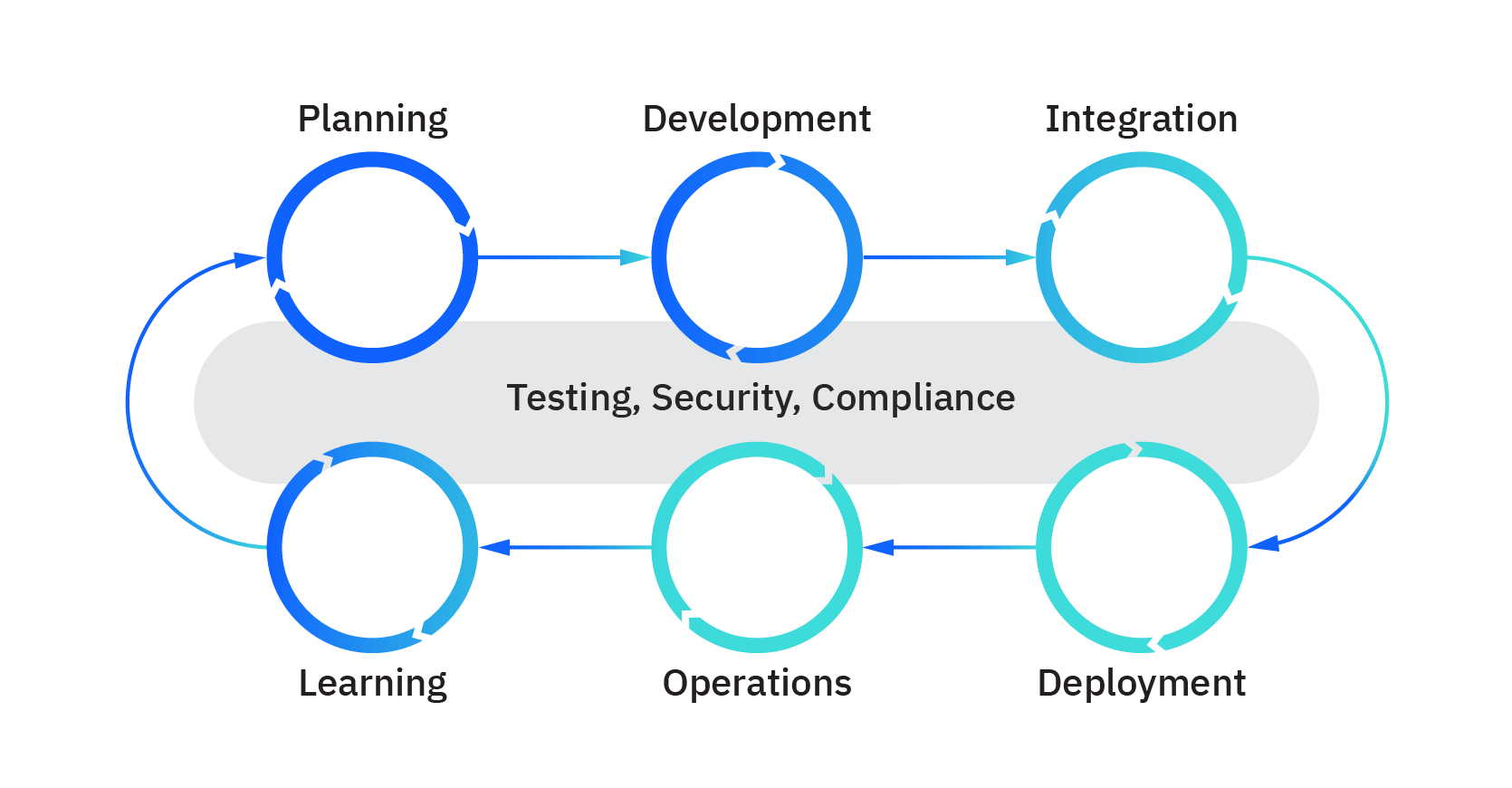


Image 11: DevOps lifecycle in Ibmcloud

Reference: <https://www.ibm.com/cloud/learn/devops-a-complete-guide>

**Planning (or ideation):** In this workflow, teams scope out new features and functionality in the next release, drawing from prioritized end-user feedback and case studies, as well as inputs from all internal stakeholders. The goal in the planning stage is to maximize the business value of the product by producing a backlog of features that when delivered produce a desired outcome that has value.

**Development:** This is the programming step, where developers test, code, and build new and enhanced features, based on user stories and work items in the backlog. A combination of practices such as test-driven development (TDD), pair programming, and peer code reviews, among others are common. Developers often use their local workstations to perform the “inner loop” of writing and testing code before sending it down the continuous delivery pipeline.

**Integration (or build, or continuous Integration and continuous delivery (CI/CD):** As noted above, in this workflow the new code is integrated into the existing code base, then tested and packaged into an executable for deployment. Common automation activities include merging code changes into a “master” copy, checking out that code from a source code repository, and automating the compile, unit test and packaging into an executable. Best practice is to store the output of the CI phase in a binary repository, for the next phase.

**Deployment (usually called continuous deployment):** Here the runtime build output (from integration) is deployed to a runtime environment - usually a development environment where runtime tests are executed for quality, compliance and security. If errors or defects are found, developers have a chance to intercept and remediate any problems before any end users see them. There are typically environments for development, test, and production, with each environment requiring progressively “stricter” quality gates. A good practice for deployment to a production environment is typically to deploy first to a subset of end users, and then eventually to all users once stability is established.

**Operations:** If getting features delivered to a production environment is characterized as “Day 1”, then once features are running in production “Day 2” operations occur. Monitoring feature performance, behavior, and availability ensures that the features are able to provide value add to end users. Operations ensures that features are running smoothly and that there are no interruptions in service - by making sure the network, storage, platform, compute and security posture are all healthy! If something goes wrong, operations ensure incidents are identified, the proper personnel are alerted, problems are determined, and fixes are applied.

**Learning (sometimes called continuous feedback):** This is the gathering of feedback from end users and customers on features, functionality, performance and business value to take back to planning for enhancements and features the next release. This would also include any learning and backlog items from the operations activities, that could empower developers to proactively avoid any past incidents that could happen again in the future. This is the point where the “wraparound” to the Planning phase happens and we “continuously improve!”

Three other important continuous workflows occur between these workflows:

**Continuous testing:** Classical DevOps lifecycles include a discrete “test” phase that occurs between integration and deployment. However, DevOps has advanced such that certain elements of testing can occur in planning (behavior-driven development), development (unit testing, contract testing), integration (static code scans, CVE scans, linting), deployment (smoke testing, penetration testing, configuration testing), operations (chaos testing, compliance testing), and learning (A/B testing). Testing is a powerful form of risk and vulnerability identification and provides an opportunity for IT to accept, mitigate, or remediate risks.

**Security:** While waterfall methodologies and agile implementations 'tack on' security workflows after delivery or deployment, DevOps strives to incorporate security from the start (Planning) - when security issues are easiest and least expensive to address - and continuously throughout the rest of the development cycle. This approach to security is referred to as shifting left.

**Compliance:** Regulatory compliance (governance and risk) are also best addressed early and throughout the development lifecycle. Regulated industries are often mandated to provide a certain level of observability, traceability and access of how features are delivered and managed in their runtime operational environment. This requires planning, development, testing, and enforcement of policies in the continuous delivery pipeline and in the runtime environment. Auditability of compliance measures is extremely important for proving compliance to 3rd party auditors.

### How DevOps works?

Under a DevOps model, development and operations teams are no longer “siloed.” Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.

In some DevOps models, quality assurance and security teams may also become more tightly integrated with development and operations and throughout the application lifecycle. When security is the focus of everyone on a DevOps team, this is sometimes referred to as DevSecOps.

These teams use practices to automate processes that historically have been manual and slow. They use a technology stack and tooling which help them operate and evolve applications quickly and reliably. These tools also help engineers independently accomplish tasks (for example, deploying code or provisioning infrastructure) that normally would have required help from other teams, and this further increases a team’s velocity.

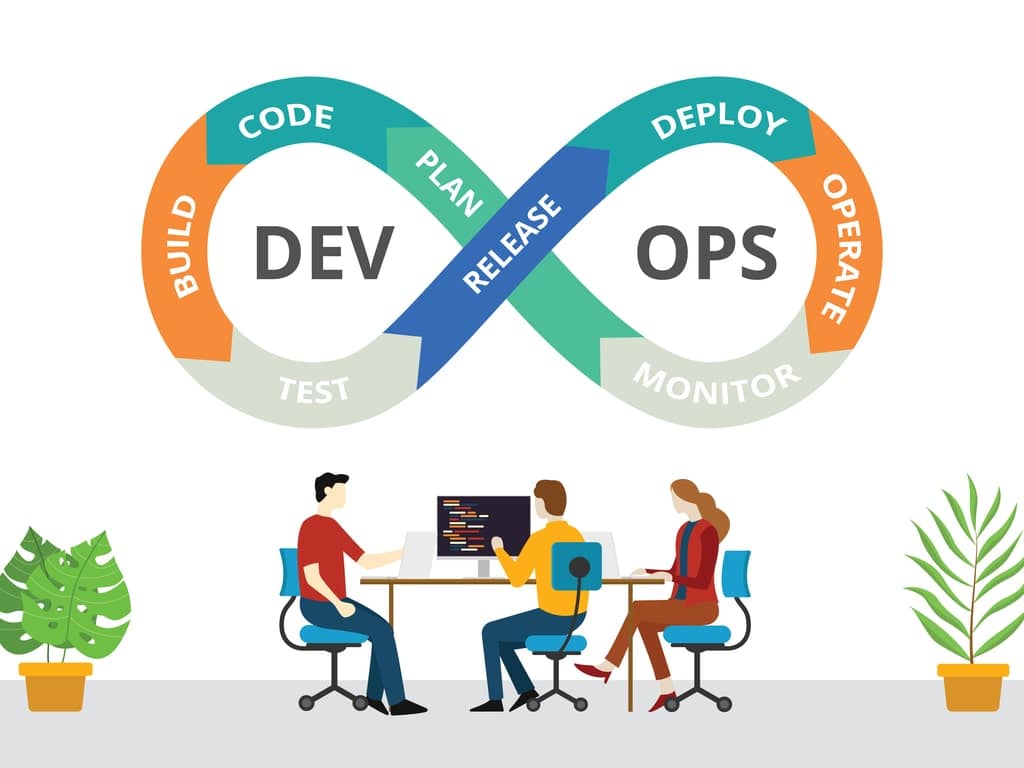


Image 12: How DevOps works?

Reference: <https://www.romexsoft.com/wp-content/uploads/2019/09/shutterstock_1363648634-min.jpg>

**1. Continuous Development:**

This stage involves committing code to version control tools such as Git or SVN for maintaining the different versions of the code, and tools like Ant, Maven, Gradle for building/packaging the code into an executable file that can be forwarded to the QAs for testing.

**2. Continuous Integration:**

The stage is a critical point in the whole DevOps Lifecycle. It deals with integrating the different stages of the DevOps lifecycle and is, therefore, the key in automating the whole DevOps Process.

**3. Continuous Deployment:**

In this stage the code is built, the environment or the application is containerized and is pushed onto the desired server. The key processes in this stage are Configuration Management, Virtualization, and Containerization.

**4.** **Continuous Testing:**

The stage deals with automated testing of the application pushed by the developer. If there is an error, the message is sent back to the integration tool, this tool, in turn, notifies the developer of the error, If the test was a success, the message is sent to Integration-tool which pushes the build on the production server.

**5. Continuous Monitoring:**

The stage continuously monitors the deployed application for bugs or crashes. It can also be set up to collect user feedback. The collected data is then sent to the developers to improve the application.

### Why AWS for DevOps?

**Get Started Fast:**

Each AWS service is ready to use if you have an AWS account. There is no setup required or software to install.



Image 13: Get Started Fast

Reference: <https://aws.amazon.com/devops/>

**Fully Managed Services:**

These services can help you take advantage of AWS resources quicker. You can worry less about setting up, installing, and operating infrastructure on your own. This lets you focus on your core product.



Image 14: Fully Managed Services

Reference: <https://aws.amazon.com/devops/>

**Built for Scale:**

You can manage a single instance or scale to thousands using AWS services. These services help you make the most of flexible compute resources by simplifying provisioning, configuration, and scaling.



Image 15: Built for Scale

Reference: <https://aws.amazon.com/devops/>

**Programmable:**

You have the option to use each service via the AWS Command Line Interface or through APIs and SDKs. You can also model and provision AWS resources and your entire AWS infrastructure using declarative AWS CloudFormation templates.



Image 16: Programmable

Reference: <https://aws.amazon.com/devops/>

**Automation:**

AWS helps you use automation so you can build faster and more efficiently. Using AWS services, you can automate manual tasks or processes such as deployments, development & test workflows, container management, and configuration management.



Image 17: Automation

Reference: <https://aws.amazon.com/devops/>

**Secure:**

Use AWS Identity and Access Management (IAM) to set user permissions and policies. This gives you granular control over who can access your resources and how they access those resources.



Image 18: Secure

Reference: <https://aws.amazon.com/devops/>

**Large Partner Ecosystem:**

AWS supports a large ecosystem of partners which integrate with and extend AWS services. Use your preferred third-party and open-source tools with AWS to build an end-to-end solution. Visit here to learn more about our DevOps Partner Solutions.



Image 19: Large Partner Ecosystem

Reference: <https://aws.amazon.com/devops/>

**Pay-As-You-Go:**

With AWS purchase services as you need them and only for the period when you plan to use them. AWS pricing has no upfront fees, termination penalties, or long term contracts. The AWS Free Tier helps you get started with AWS.



Image 20: Pay-As-You-Go

Reference: <https://aws.amazon.com/devops/>

### Benefits of DevOps

**Speed:**

Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results. The DevOps model enables your developers and operations teams to achieve these results. For example, microservices and continuous delivery let teams take ownership of services and then release updates to them quicker.

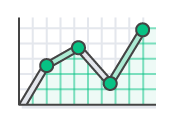


Image 21: Speed

Reference: <https://aws.amazon.com/devops/what-is-devops>

**Rapid Delivery:**

Increase the frequency and pace of releases so you can innovate and improve your product faster. The quicker you can release new features and fix bugs, the faster you can respond to your customers’ needs and build competitive advantage. Continuous integration and continuous delivery are practices that automate the software release process, from build to deploy.



Image 22: Rapid Delivery

Reference: <https://aws.amazon.com/devops/what-is-devops>

**Reliability:**

Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for end users. Use practices like continuous integration and continuous delivery to test that each change is functional and safe. Monitoring and logging practices help you stay informed of performance in real-time.

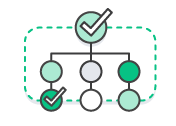


Image 23: Reliability

Reference: <https://aws.amazon.com/devops/what-is-devops>

**Scale:**

Operate and manage your infrastructure and development processes at scale. Automation and consistency help you manage complex or changing systems efficiently and with reduced risk. For example, infrastructure as code helps you manage your development, testing, and production environments in a repeatable and more efficient manner.

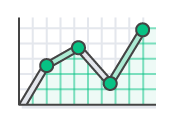


Image 24: Scale

Reference: <https://aws.amazon.com/devops/what-is-devops>

**Improved Collaboration:**

Build more effective teams under a DevOps cultural model, which emphasizes values such as ownership and accountability. Developers and operations teams collaborate closely, share many responsibilities, and combine their workflows. This reduces inefficiencies and saves time (e.g. reduced handover periods between developers and operations, writing code that takes into account the environment in which it is run).

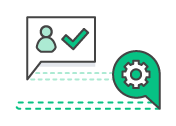


Image 25: Improved Collaboration

Reference: <https://aws.amazon.com/devops/what-is-devops>

**Security:**

Move quickly while retaining control and preserving compliance. You can adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques. For example, using infrastructure as code and policy as code, you can define and then track compliance at scale.

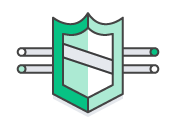


Image 26: Security

Reference: <https://aws.amazon.com/devops/what-is-devops>

### History of DevOps

Before DevOps, we had two approaches for software development namely the Waterfall and the Agile.

**Waterfall Model**

* The waterfall model is a software development model that is pretty straight forward and linear. This model follows a top-down approach.
* This model has various starting with Requirements gathering and analysis. This is the phase where you get the requirements from the client for developing an application. After this, you try to analyze these requirements.
* The next phase is the Design phase where you prepare a blueprint of the software. Here, you think about how the software is actually going to look like.
* Once the design is ready, you move further with the Implementation phase where you begin with the coding for the application. The team of developers works together on various components of the application.
* Once you complete the application development, you test it in the Verification phase. There are various tests conducted on the application such as unit testing, integration testing, performance testing, etc.
* After all the tests on the application are completed, it is deployed onto the production servers.
* At last, comes the Maintenance phase. In this phase, the application is monitored for performance. Any issues related to the performance of the application are resolved in this phase.

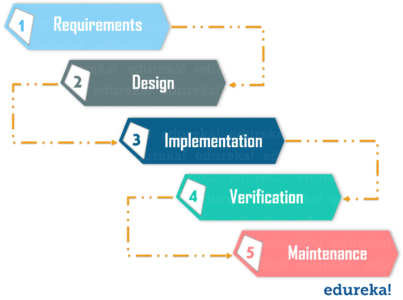


Image 27: Waterfall model

Reference: <https://www.edureka.co/blog/what-is-devops/>

**Advantages of the Waterfall Model:**

* Simple to understand and use
* Allows for easy testing and analysis
* Saves a significant amount of time and money
* Good for small projects if all requirements are clearly defined
* Allows for departmentalization & managerial control

**Disadvantages of Waterfall Model:**

* Risky and uncertain
* Lack of visibility of the current progress
* Not suitable when the requirements keep changing
* Difficult to make changes to the product when it is in the testing phase
* The end product is available only at the end of the cycle
* Not suitable for large and complex projects

**Agile Methodology**

Agile Methodology is an iterative based software development approach where the software project is broken down into various iterations or sprints. Each iteration has phases like the waterfall model such as Requirements Gathering, Design, Development, Testing, and Maintenance. The duration of each iteration is generally 2-8 weeks.

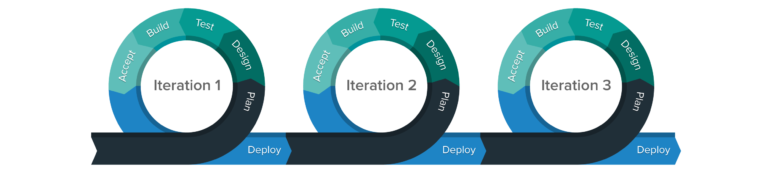


Image 28: Agile Methodology

Reference: <https://www.edureka.co/blog/what-is-devops/>

**Agile Process**

* In Agile, a company releases the application with some high priority features in the first iteration.
* After its release, the end-users or the customers give you feedback about the performance of the application.
* Then you make the necessary changes into the application along with some new features and the application is again released which is the second iteration.
* You repeat this entire procedure until you achieve the desired software quality.

**Advantages of Agile Model**

* It adaptively responds to requirement changes favorably
* Fixing errors early in the development process makes this process more cost-effective
* Improves the quality of the product and makes it highly error-free
* Allows for direct communication between people involved in software project
* Highly suitable for large & long-term projects
* Minimum resource requirements & very easy to manage

**Disadvantages of Agile Model**

* Highly dependent on clear customer requirements
* Quite Difficult to predict time and effort for larger projects
* Not suitable for complex projects
* Lacks documentation efficiency
* Increased maintainability risks

**DevOps vs Agile**

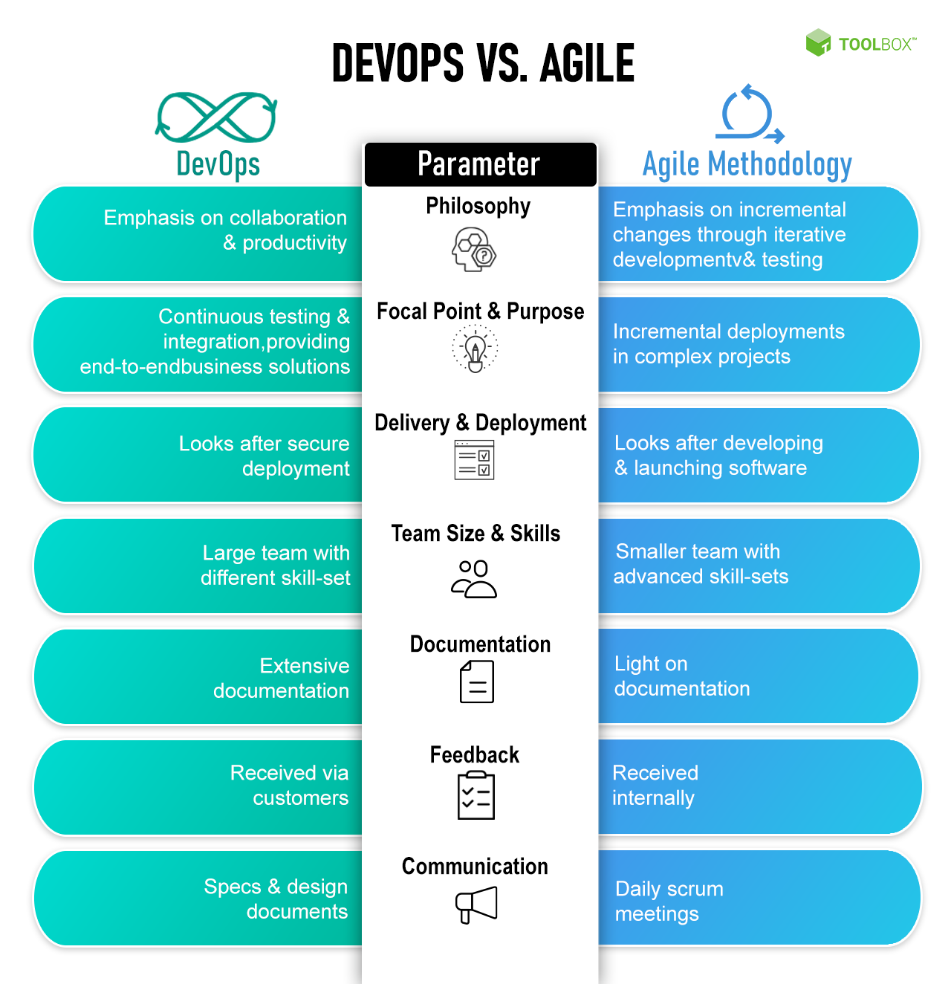


Image 29: DevOps vs Agile

Reference: <https://www.toolbox.com/tech/devops/articles/devops-vs-agile/>

### DevOps methodologies

DevOps is the direct descendant of agile software development, born from the need to keep up with increased software development velocity and throughput agile methods. Advancements in agile development highlighted the need for a more holistic approach to the software delivery life cycle, resulting in DevOps.

“Agile development” is an umbrella term for several iterative software development methodologies, many of which have carried over to DevOps:

* **Scrum**-a framework in which people can address complex adaptive problems while delivering products of the highest possible value.
* **Kanban**—a method for managing the creation of products with an emphasis on continual delivery while not overburdening the development team. Like Scrum, Kanban is a process designed to help teams work together more effectively.
* **Scaled Agile Framework (SAFe)**-a set of organization and workflow patterns intended to guide enterprises in scaling lean and agile practices. SAFe is one of a growing number of frameworks that seek to address the problems encountered when scaling beyond a single team.
* **Lean development**-a translation of lean manufacturing principles and practices to the software development domain. Lean offers a conceptual framework, values, and principles, as well as best practices derived from experience, that support agile organizations.
* **Extreme programming (XP)**-a software development methodology intended to improve software quality and responsiveness to changing customer requirements. XP advocates frequent releases in short development cycles, intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted. Other elements of extreme programming include programming in pairs or doing extensive code review, unit testing of all code, not programming of features until they are needed, a flat management structure, code simplicity and clarity, expecting changes in the customer’s requirements as time passes and the problem is better understood, and frequent communication with the customer.

### DevOps Tools

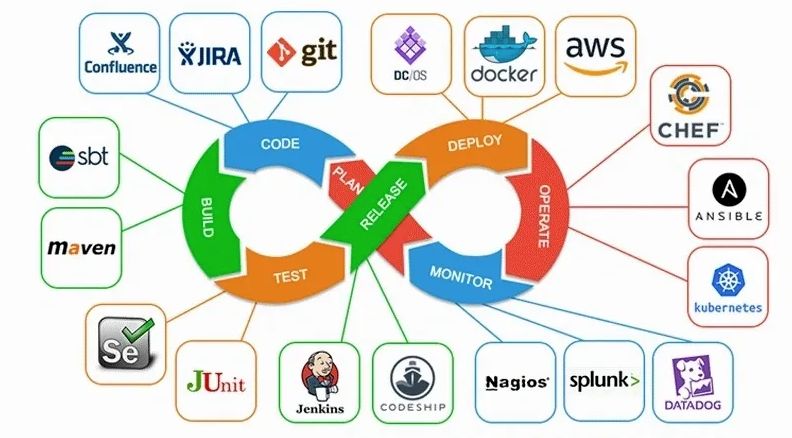


Image 30: DevOps Tools

Reference: <https://simpat.tech/wp-content/uploads/2021/09/devops-architecture-diagram.jpg>

DevOps tools lead to more collaborative teams, development pipelines that are faster, more robust applications and services, and more satisfied customers. We will be covering the following DevOps tools in this blog:

* Git
* Docker
* Selenium
* Jenkins
* Ansible
* Puppet
* Nagios
* Chef
* SVN
* Rational Clearcase
* Maven
* Apache Ant
* Kubernetes
* SignalFx
* Prometheus
* Splunk Cloud
* Raygun
* Catchpoint
* Splunk On-Call
* Gremlin
* JUnit

Quick and high-quality application delivery models have become the new standard of expectations. This triggered the rapid adoption of DevOps practices, which in turn, necessitated the high demand for DevOps tools. Here are some of the top DevOps tools:

**Git**

Git is an open-source, distributed, and the most popular software versioning system. Git has several sets of commands used to perform various operations. It works on a client-server model, which basically means that there is a central server that holds the main repository of code. Code can be downloaded from the main repository simultaneously by various clients or developers. Git is developed by Linus Torvalds. Git facilitates teams that are located at geographically different places to collaborate on the same project. Popular companies like Google, Facebook, Microsoft, Netflix extensively use Git in their CI/CD pipelines.

**Docker**

Docker is a containerization technology. Containers consist of all the applications with all of their dependencies. These containers can be deployed on any machine without caring about underlying host details. Containers can be a .net application or a website along with its dependencies like .net or lamp stack in case of a website application. These containers are used to automate the deployment process of applications in production and non-production environment.

**Selenium**

Selenium is an open-source, automated testing framework. Selenium is used majorly to automate the testing of web applications. We can define test in selenium which we want to test against our application. Selenium can repeatedly apply those tests to our application without manual intervention and generate reports. These reports can be used by testing individuals to see if the application is behaving correctly or not.

**Jenkins**

Jenkins is an open-source automation server written in java. Jenkins provides automation of the continuous delivery part. Jenkins is used in creating continuous delivery pipelines. To give you an example, here is a scenario wherein a Jenkins server will take our application container from the development environment and make it accessible to the testing environment, QA environment, or any other non-production environment in the beginning. Jenkins server is in the middle of the whole CI/CD pipeline. It automates this whole process which means whenever a developer commits a change in a code that code will automatically be visible to the testing server or QA team. They then can provide instantaneous feedback on these changes. Jenkins is used by Microsoft, Redhat, Rackspace, to name a few.

**Ansible**

Ansible is an open-source application that is used for automated software provisioning, configuration management, and application deployment. Ansible is the backbone of controlling an automated cluster environment consisting of many machines. Ansible works on the client-server model. The client acts as a master, which is the center point in our cluster and provides centralized control of all client machines (slaves) that are connected to it. We can give any command to any client machine or deploy any application to more than one machine from a single master machine. Ansible only requires SSH for communication so it does not need any software dependency to run. Ansible works on Unix.

**Puppet**

Puppet is an open-source software configuration management, automated provisioning tool. It is an alternative to Ansible and provides better control over client machines. Puppet comes up with GUI which makes it easier to use than Ansible. Puppet is cross-platform, it runs on both Unix and Microsoft Windows. Puppet uses a manifest file and applies those specifications across all machines. Unlike Ansible, Puppet is an agent-based tool. The Puppet master runs on a master machine and the Puppet agent runs on all client machines. Puppet is used by Microsoft, Google, Accenture, etc.

**Nagios**

Nagios is used for continuous monitoring of infrastructure. Nagios provides server monitoring, application monitoring, and network monitoring. With Nagios, we can monitor a whole data center from a single server. We can see whether switches are working correctly, servers are not having too much load, or if any part of the application is down. It provides a nice GUI interface to check various details like how much memory is used, what is the fan speed, routing tables of switches, or the state of the SQL server. Nagios has a modular design. It supports NRPE plugins which can be used to add monitoring parameters on existing Nagios. There are various plugins available on the internet which can be used freely to add features to Nagios. Nagios is the most popular tool in continuous monitoring.

**Chef**

Chef is a configuration management tool. Chef is used to manage configurations like creating or removing a user, adding an SSH key to a user present on multiple nodes, installing or removing a service, etc. We can manage up to 10,000 nodes by using chef. These changes are pushed by cookbooks or recipes. Chef has three components viz. Chef server, workstation, and nodes. The Chef server is a central point where all details of our Chef infrastructure reside. Chef workstation holds recipes or cookbooks which pushes particular configurations to our chef infrastructure. Nodes are simple machines that are configured using chef. Chef has API support from AWS, Azure, Rackspace, which makes it easy to use with infrastructure-as-code methodology.

**SVN**

SVN or Subversion is an open-source, centralized software versioning & revision control system. SVN is an alternative to GIT. SVN is a centralized system which means every time if a team member or client wants to make changes in the code, he has to inform the central server or repo about these changes. To use an SVN repo, the client needs to connect to the central server. It can check out the code to download the repo from the server. The client then moves on to make changes in this code and commit this change by informing the central repo. From the central repo, these changes will be visible to other team members or clients.

**Rational ClearCase**

Rational ClearCase manages changes across the software lifecycle. ClearCase is used in Software configuration management of source code. ClearCase is used in both hardware and software development. The center of ClearCase is a secure data repository. It has data that is shared by all users including accounting data and historical data on development processes themselves. It shows details like which user implemented which version, when, and why. There are 3 products of ClearCase—Rational ClearCase is for medium to large teams, Rational ClearCase LT is useful for small to medium teams, and Rational ClearCase multisite is used by geographically distributed teams.

**Maven**

Maven is a build automation tool. It automates the software build process & dependencies resolution. A Maven project is configured using a project object model or POM.XML file which describes the build process and the software project, its dependencies on external modules and components, build order, directories. Maven can dynamically download these external modules & Maven plugins during the build process itself. Maven can build and manage projects on Java, C#, Scala, Ruby, and other languages. Maven is majorly used by Apache Foundation to automate the build of some of its large projects like Apache Hadoop.

**Apache Ant**

Apache Ant is a software tool used for the automation of the software build process. Apache Ant is inspired by Unix make utility. Apache Ant uses an XML file build.xml in place of makefile which is used by make utility for build processes. It automates repetitive tasks in the build process and generates documentation. Ant builds are based on three blocks viz. tasks, targets, and extension points. Ant supports many third-party extensions like Eclipse IDE and NetBeans IDE.

**Kubernetes**

Kubernetes is an open-source container orchestration tool. It is developed by Google. It is used in continuous deployment and auto-scaling of container clusters. It increases fault tolerance, load balancing in a container cluster. Kubernetes maintains the desired state of a cluster; this desired state is described in the YAML file. YAML file contains the state of pods or slave nodes and replication unit for a cluster. Kubernetes uses this YAML file to maintain the desired state of the cluster. For example, in case, a pod is serving more requests than another pod, then it can automatically distribute the load to other pods. In case one machine fails, then it can configure another pod to replace it thus ensuring fault tolerance, load balancing, and high availability in a cluster. Kubernetes is used in high-performance data centers like that of Google, Facebook, and Amazon Web Services.

**SignalFx**

SignalFx, which recently got acquired by Splunk, is a complete observability tool. It can collect traces, metrics, and events from applications and infrastructure to help inform users of not only the system’s health but why it is behaving a certain way. This helps teams to fix issues faster as well as connect application and infrastructure monitoring with the requirements of the business. SignalFx is great to use for debugging and post-incident reviews through high cardinality analytics, service mapping, and detailed visualizations and dashboards.

**Prometheus**

Prometheus is an open-source, time-series database and monitoring tool that is mostly used by DevOps and IT teams. It generates alerts based on time-series data. One can generate precise alerts and visualizations to get business insights and engineering outcomes. Developers and IT practitioners can easily customize the tool for their own use cases. Following are some of its features:

* Functional sharding and federation helps with scaling
* Easy service instrumentation with numerous client libraries
* PromQL enables powerful reporting capabilities

**Splunk Cloud**

Splunk Enterprise and Splunk Cloud are log management, infrastructure monitoring, and application monitoring tools. It can collect data from services and devices as well as other monitoring tools such as DevOps monitoring tools. Splunk serves as the single source of truth for the health and performance of a system. Its powerful log search, filtration functionality, and informative visualizations and dashboards lead to quicker incident resolution. Splunk provides observability and provides the tools required to take action whether there is an on-premises architecture or in a cloud environment. Splunk when paired with VictorOps creates complete observability, data-driven incident management, and on-call incident response.

**Raygun**

Raygun generates real insights into the way users experience a service. It has the ability to provide detailed reports on everything like full-scale app crashes, downtime, performance metrics like load speeds, network latency, etc. Its real-user monitoring can identify and expose both client-side and server-side problems for users as well as help product teams give priority to engineering roadmaps to align with real problems. The APM tool works well with Raygun’s error management workflow. It automatically links errors back to the source code. This brings Development and Operations together through a single source of truth for the entire team—performance problems and the cause of errors.

**Catchpoint**

The Catchpoint monitoring tool combines synthetic monitoring, real-user monitoring, network monitoring, and endpoint monitoring for the detection of errors and incidents anywhere in an architecture. The ability to run synthetic metrics through the system is a unique advantage as compared to others. For smaller teams who don’t have large amounts of real metrics, it can help you find service reliability or performance problems as you scale before customers notice it. Catchpoint helps identify issues whether it’s caused by a user’s browser or device or an application or infrastructure problem.

**Splunk On-Call**

With Splunk On-Call, one can alert engineers and on-call responders about problems and incidents in real-time as well as provide contextual alert information and remediation instructions. In one single pane of glass, DevOps and IT practitioners can collaborate to drastically reduce the time taken to acknowledge and resolve incidents. With reports like MTTA/MTTR, Post-Incident Review, Incident Frequency, teams can drive swift problem resolution, manage alert noise, reduce burnout, etc.

**Gremlin**

Netflix popularized chaos engineering as a means to simulate chaos through systems in order to test the response to stress and unpredictable events. Gremlin lets you design experiments according to your liking to conduct or simply re-enact issues experienced in the past and run them through your applications and services to see how they handle it.

**JUnit**

JUnit is the de-facto test automation tool for Java. No matter if JUnit is used for writing unit or integration tests or even if another framework is used in conjunction with it - such as Selenium - the output results file can be uploaded to Zephyr Scale for generating reports and leveraging all the other capabilities

**What is the best DevOps tool?**

The best DevOps tools are the ones that serve the processes and people that form your DevOps culture. DevOps is not a product you can buy and implement and say, “Now we’ve got DevOps.” DevOps technologist Alex Honor published a post in the early days of DevOps titled “People Over Process Over Tools,” crystalizing the movement’s emphasis on culture over tooling.

That said, there are a number of notable products that help DevOps teams get the job done. A non-exhaustive list of prominent tools, both open-source and proprietary, includes:

* **Source code management:** Git (GitLab, GitHub), Bitbucket
* **Configuration management:** Puppet, Chef, Ansible, CFEngine
* **Release management:** Jenkins, Travis, CircleCl, TeamCity, Gradle, Bamboo
* **Orchestration:** Mesos, Zookeeper, Kubernetes
* **Monitoring, virtualization, containerization:** Nagios, Icignia, Monit, OpenStack, Vagrant, AWS, Docker, Kubernetes
* **Log and application lifecycle analytics:** Splunk is a leading log management tool ideal for DevOps.

**DevOps Tooling by AWS**

AWS provides services that help you practice DevOps at your company and that are built first for use with AWS. These tools automate manual tasks, help teams manage complex environments at scale, and keep engineers in control of the high velocity that is enabled by DevOps.

* Continuous Integration and Continuous Delivery
* Microservices
* Infrastructure as Code
* Monitoring and Logging
* Platform as a Service
* Version Control

**Continuous Integration and Continuous Delivery**

The AWS Developer Tools help you securely store and version your application's source code and automatically build, test, and deploy your application to AWS or your on-premises environment.

Start with AWS CodePipeline to build a continuous integration or continuous delivery workflow that uses AWS CodeBuild, AWS CodeDeploy, and other tools, or use each service separately.

**Software Release Workflows**

**AWS CodePipeline-**AWS CodePipeline is a continuous integration and continuous delivery service for fast and reliable application and infrastructure updates. CodePipeline builds, tests, and deploys your code every time there is a code change, based on the release process models you define. This enables you to rapidly and reliably deliver features and updates.

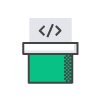


Image 31: Software Release Workflows

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**Build and Test Code**

**AWS CodeBuild**-AWS CodeBuild is a fully managed build service that compiles source code, runs tests, and produces software packages that are ready to deploy. With CodeBuild, you don’t need to provision, manage, and scale your own build servers. CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left waiting in a queue.

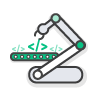


Image 32: Build and Test Code

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**Deployment Automation**

**AWS CodeDeploy-**AWS CodeDeploy automates code deployments to any instance, including Amazon EC2 instances and 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.



Image 33: Deployment Automation

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**Unified CI/CD Projects**

**AWS CodeStar-**AWS CodeStar enables you to quickly develop, build, and deploy applications on AWS. AWS CodeStar provides a unified user interface, enabling you to easily manage your software development activities in one place. With AWS CodeStar, you can set up your entire continuous delivery toolchain in minutes, allowing you to start releasing code faster.



Image 34: Unified CI/CD Projects

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**Microservices**

Build and deploy a microservices architecture using containers or serverless computing.

**Production Docker Platform**

**Amazon Elastic Container Service-**Amazon Elastic Container Service (ECS) is a highly scalable, high performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances.



Image 35: Production Docker Platform

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**Serverless Computing**

**AWS Lambda**-AWS Lambda lets you run code without provisioning or managing servers. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability.



Image 36: Serverless Computing

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**Infrastructure as Code**

Provision, configure, and manage your AWS infrastructure resources using code and templates. Monitor and enforce infrastructure compliance.

**Templated Infrastructure Provisioning**

**AWS CloudFormation**-AWS CloudFormation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion. You can use AWS CloudFormation’s sample templates or create your own templates.

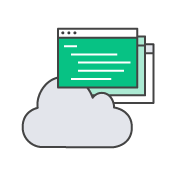


Image 37: Templated infrastructure Provisioning

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**Chef Configuration Management**

**AWS OpsWorks**-AWS OpsWorks is a configuration management service that uses Chef, an automation platform that treats server configurations as code. OpsWorks uses Chef to automate how servers are configured, deployed, and managed across your Amazon Elastic Compute Cloud (Amazon EC2) instances or on-premises compute environments. OpsWorks has two offerings, AWS Opsworks for Chef Automate, and AWS OpsWorks Stacks.

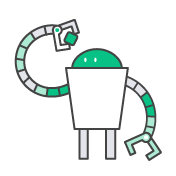


Image 38: Chef Configuration Management

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**Configuration Management**

**AWS Systems Manager**-AWS Systems Manager is a management service that helps you automatically collect software inventory, apply OS patches, create system images, and configure Windows and Linux operating systems. These capabilities help you define and track system configurations, prevent drift, and maintain software compliance of your EC2 and on-premises configurations.

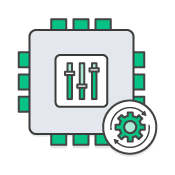


Image 39: Configuration Management

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**Policy as Code**

**AWS Config**-AWS Config is a fully managed service that provides you with an AWS resource inventory, configuration history, and configuration change notifications to enable security and governance. Config Rules enables you to create rules that automatically check the configuration of AWS resources recorded by AWS Config.



Image 40: Policy as Code

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**Monitoring and Logging**

Record logs and monitor application and infrastructure performance in near real-time.

**Cloud and Network Monitoring**

**Amazon CloudWatch**-Amazon CloudWatch is a monitoring service for AWS cloud resources and the applications you run on AWS. You can use Amazon CloudWatch to collect and track metrics, collect and monitor log files, set alarms, and automatically react to changes in your AWS resources.



Image 41: Cloud and Network Monitoring

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**Distributed Tracing**

**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.



Image 42: Distributed Racing

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**Activity & API Usage Tracking**

**AWS CloudTrail-**AWS CloudTrail is a web service that records AWS API calls for your account and delivers log files to you. The recorded information includes the identity of the API caller, the time of the API call, the source IP address of the API caller, the request parameters, and the response elements returned by the AWS service.



Image 43: Activity & API Usage Tracking

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**Platform as a Service**

Deploy web applications without needing to provision and manage the infrastructure and application stack.

**Run and Manage Web Apps**

**AWS Elastic Beanstalk**-AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.

You can simply upload your code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, auto-scaling to application health monitoring. At the same time, you retain full control over the AWS resources powering your application and can access the underlying resources at any time.



Image 44: Run and Manage Web Apps

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**Version Control**

Host secure, highly scalable Git repositories in the cloud.

**Private Git Hosting**

**AWS CodeCommit-**AWS CodeCommit is a fully-managed source control service that makes it easy for companies to host secure and highly scalable private Git repositories. You can use CodeCommit to securely store anything from source code to binaries, and it works seamlessly with your existing Git tools.

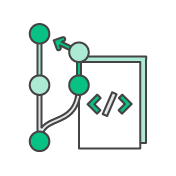


Image 45: Private Git Hosting

Reference: [https://aws.amazon.com/devops/](https://aws.amazon.com/devops/what-is-devops)

**IBMCloud DevOps tools: Building a DevOps toolchain**

The demands of DevOps and DevOps culture put a premium on tooling that supports asynchronous collaboration, seamlessly integrates DevOps workflows, and automates the entire DevOps lifecycle as much as possible. Categories of DevOps tools include:

**Project management tools:**

Tools that enable teams to build a backlog of user stories (requirements) that form coding projects, break them down into smaller tasks and track the tasks through to completion. Many supports agile project management practices such, as Scrum, Lean and Kanban, that developers bring to DevOps. Popular open-source options include GitHub Issues and Jira.

**Collaborative source code repositories:**

Version-controlled coding environments that that let multiple developers to work on the same code base. Code repositories should integrate with CI/CD, testing and security tools, so that when code is committed to the repository it can automatically move to the next step. Open-source code repositories include GiHub and GitLab.

**CI/CD pipelines:**

Tools that automate code checkout, building, testing and deployment. Jenkins is the most popular open-source tool in this category; many previously open-source alternatives, such as CircleCI, are now available in commercial versions only. When it comes to continuous deployment (CD) tools, Spinnaker straddles between application and infrastructure as code layers. ArgoCD is another popular open-source choice for Kubernetes native CI/CD.

**Test automation frameworks:**

These include software tools, libraries and best practices for automating unit, contract, functional, performance, usability, penetration and security tests. The best of these tools support multiple languages; some use artificial intelligence (AI) to automatically reconfigure tests in response to code changes. The expanse of test tools and frameworks is far and wide! Popular open-source test automation frameworks include Selenium, Appium, Katalon, Robot Framework, and Serenity (formerly known as Thucydides).

**Configuration management (infrastructure as code) tools:**

These enable DevOps engineers to configure and provision fully versioned and fully documented infrastructure by executing a script. Open-source options include Ansible (Red Hat), Chef, Puppet and Terraform. Kubernetes performs the same function for containerized applications (see 'DevOps and cloud-native development,' below).

**Monitoring tools:**

These help DevOps teams identify and resolve system issues; they also gather and analyze data in real time to reveal how code changes impact application performance. Open-source monitoring tools include Datadog, Nagios, Prometheus and Splunk.

**Continuous feedback tools:**

Tools that gather feedback from users, either through heatmapping (recording users' actions on screen), surveys, or self-service issue ticketing.

### DevOps Services

**AWS**

* AWS CodePipeline
* AWS CodeBuild
* AWS CodeDeploy
* AWS CodeStar

**AWS CodePipeline**

AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define. This enables you to rapidly and reliably deliver features and updates. You can easily integrate AWS CodePipeline with third-party services such as GitHub or with your own custom plugin. With AWS CodePipeline, you only pay for what you use. There are no upfront fees or long-term commitments

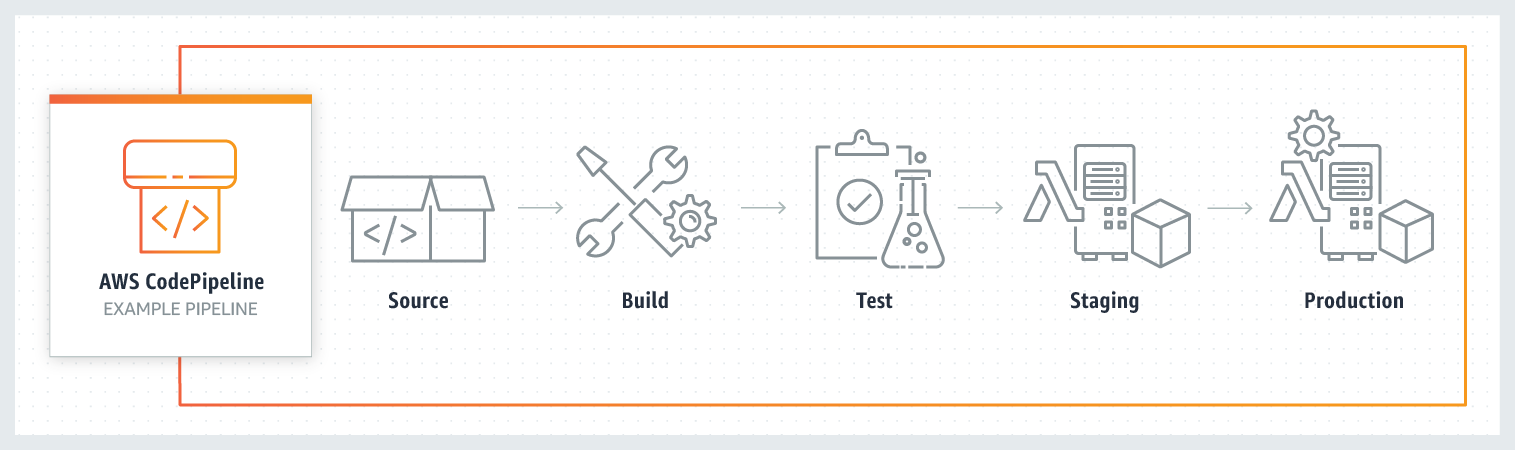


Image 46: Code pipeline

Reference: <https://aws.amazon.com/codepipeline/>

**Benefits**

**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.

Automating your build, test, and release process allows you to quickly and easily test each code change and catch bugs while they are small and simple to fix. You can ensure the quality of your application or infrastructure code by running each change through your staging and release process.

**Configurable workflow**

AWS CodePipeline allows 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.

**Get started fast**

With AWS CodePipeline, you can immediately begin to model your software release process. There are no servers to provision or set up. CodePipeline is a fully managed continuous delivery service that connects to your existing tools and systems.

**Easy to integrate**

AWS CodePipeline can easily be extended to adapt to your specific needs. You can use our 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.

**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. With CodeBuild, you don’t need to provision, manage, and scale your own build servers. CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left waiting in a queue. You can get started quickly by using prepackaged build environments, or you can create custom build environments that use your own build tools. With CodeBuild, you are charged by the minute for the compute resources you use.

**Benefits**

Fully managed build service

AWS CodeBuild eliminates the need to set up, patch, update, and manage your own build servers and software. There is no software to install or manage.

**Continuous scaling**

AWS CodeBuild scales up and down automatically to meet your build volume. It immediately processes each build you submit and can run separate builds concurrently, which means your builds are not left waiting in a queue.

**Pay as you go**

With AWS CodeBuild, you are charged based on the number of minutes it takes to complete your build. This means you no longer have to worry about paying for idle build server capacity.

**Extensible**

You can bring your own build tools and programming runtimes to use with AWS CodeBuild by creating customized build environments in addition to the prepackaged build tools and runtimes supported by CodeBuild.

**Enables continuous integration and delivery**

AWS CodeBuild belongs to a family of AWS Code Services, which you can use to create complete, automated software release workflows for continuous integration and delivery (CI/CD). You can also integrate CodeBuild into your existing CI/CD workflow. For example, you can use CodeBuild as a worker node for your existing Jenkins server setup for distributed builds.

**Secure**

With AWS CodeBuild, your build artifacts are encrypted with customer-specific keys that are managed by the AWS Key Management Service (KMS). CodeBuild is integrated with AWS Identity and Access Management (IAM), so you can assign user-specific permissions to your build projects

**AWS CodeDeploy**

AWS CodeDeploy is a fully managed deployment service that automates software deployments to a variety of compute services such as 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 AWS CodeDeploy to automate software deployments, eliminating the need for error-prone manual operations. The service scales to match your deployment needs.

**Benefits**

**Automated deployments**

AWS CodeDeploy fully automates your software deployments, allowing you to deploy reliably and rapidly. You can consistently deploy your application across your development, test, and production environments whether deploying to Amazon EC2, AWS Fargate, AWS Lambda, or your on-premises servers. The service scales with your infrastructure.

**Minimize downtime**

AWS 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.

**Centralized control**

AWS CodeDeploy allows 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 allowing 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.

**Easy to adopt**

AWS CodeDeploy is platform and language agnostic, works with any application, and provides the same experience whether you’re deploying to Amazon EC2, AWS Fargate, or AWS Lambda. You can easily reuse your existing setup code. CodeDeploy can also integrate with your existing software release process or continuous delivery toolchain (e.g., AWS CodePipeline, GitHub, Jenkins).

**AWS CodeStar**

AWS CodeStar enables you to quickly develop, build, and deploy applications on AWS. AWS CodeStar provides a unified user interface, enabling you to easily manage your software development activities in one place. With AWS CodeStar, you can set up your entire continuous delivery toolchain in minutes, allowing you to start releasing code faster. AWS CodeStar makes it easy for your whole team to work together securely, allowing you to easily manage access and add owners, contributors, and viewers to your projects. Each AWS CodeStar project comes with a project management dashboard, including an integrated issue tracking capability powered by Atlassian JIRA Software. With the AWS CodeStar project dashboard, you can easily track progress across your entire software development process, from your backlog of work items to teams’ recent code deployments. Visit here to learn more.

There is no additional charge for using AWS CodeStar. You only pay for the AWS resources that you provision for developing and running your application (for example, Amazon EC2 instances).

**Benefits**

**Start developing on AWS in minutes**

AWS CodeStar makes it easy for you to set up your entire development and continuous delivery toolchain for coding, building, testing, and deploying your application code. To start a project, you can choose from a variety of AWS CodeStar templates for Amazon EC2, AWS Lambda, and AWS Elastic Beanstalk. You have the option to choose AWS CodeCommit or GitHub to use as your project’s source control. You also have the option to edit your source code using one of several options including AWS Cloud9, Microsoft Visual Studio, or Eclipse. After you make your selections the underlying AWS services are provisioned in minutes, allowing you to quickly start coding and deploying your applications.

**Manage software delivery in one place**

AWS CodeStar provides an easy way to coordinate your day-to-day development activities through a unified user interface, reducing the need to switch between various service consoles. AWS CodeStar’s project dashboard lets you monitor application activity, and track progress across all stages of your software development process, including code commits, builds, tests, and deployments, from a central place. AWS CodeStar integrates Atlassian JIRA Software, a third-party issue tracking and project management tool, allowing you to easily manage JIRA issues directly in the AWS CodeStar dashboard.

**Work across your team securely**

AWS CodeStar enables you to collaborate on projects across your team in a secure manner. You can easily manage access for project owners, contributors, and viewers without needing to manually configure your own policy for each service. AWS CodeStar simplifies the process of setting up project access for teams by providing built-in role-based policies that follow AWS Identity and Access Management best practices.

**Choose from a variety of project templates**

With AWS CodeStar project templates, you can easily develop a variety of applications such as websites, web applications, web services, and Alexa skills. AWS CodeStar project templates include the code for getting started on supported programming languages including Java, JavaScript, PHP, Ruby, C#, and Python.

**DevOps services on IBM Cloud**

**IBM Cloud Continuous Delivery**

Embrace enterprise-ready DevOps. Create toolchains, automate builds, tests, deployments and more.

**IBM Cloud App Configuration**

Centralize feature management and configuration.

**IBM Cloud Schematics**

Configure and automate management of IBM Cloud resources.

**IBM Key Protect**

Get visibility and control of entire key lifecycle.

**IBM Cloud Secrets Manager**

Centrally manage secrets used in your apps and services.

**IBM Kubernetes Service**

Deploy and operate a Kubernetes cluster on IBM Cloud

**IBM Cloud Satellite™**

Run IBM Cloud services on your infrastructure with consistency.

**IBM Cloud Code Engine**

Run your application, batch jobs or container on a managed serverless platform.

**IBM Cloud Monitoring**

Get in-depth visibility into infrastructure and app performance.

**IBM Cloud Security and Compliance Center**

Govern cloud resource configurations and centrally manage your compliance with organization and regulatory guidelines.

**IBM Cloud Activity Tracker**

Manage compliance controls within the IBM Cloud platform.

### Disadvantages of DevOps

* Demands proper mindset across the company
* Lowered business security by outsourcing the DevOps operations
* Dealing with the legacy system is a challenge
* Practicing security for CI/CD is a separate affair
* Getting the right pool of DevOps expertise is a challenge
* Challenges with the number of tools and switching tools
* Transition challenges (organizational and technical)

# AWS Code Commit, Deploy and Pipeline

## AWS CodePipeline

AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define. This enables you to rapidly and reliably deliver features and updates. You can easily integrate AWS CodePipeline with third-party services such as GitHub or with your own custom plugin. With AWS CodePipeline, you only pay for what you use. There are no upfront fees or long-term commitments

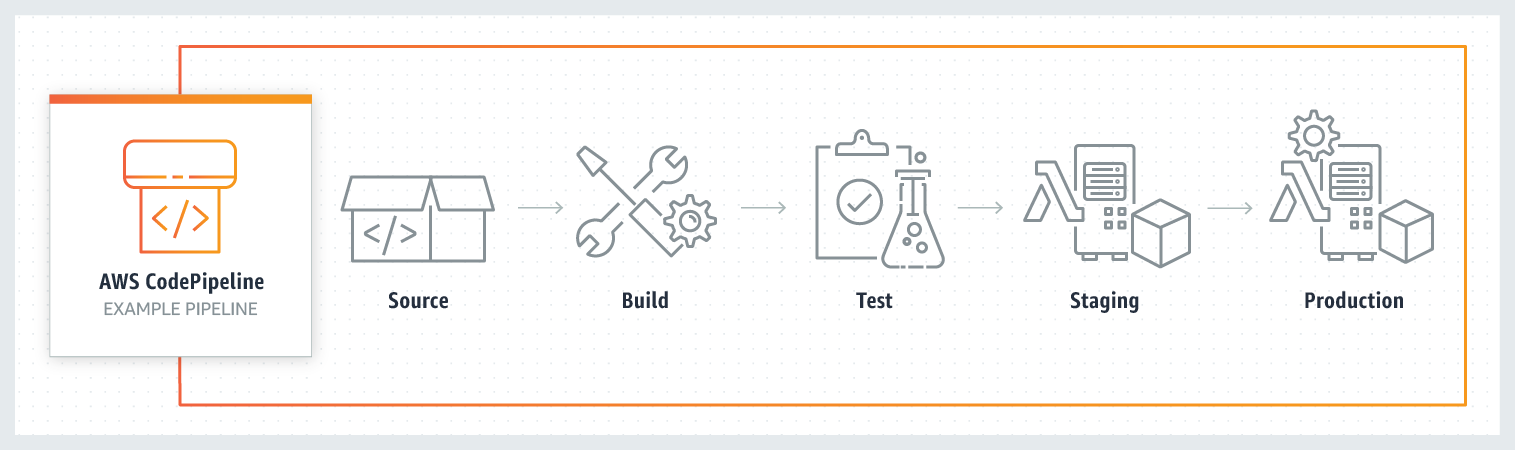


Image 47: Code pipeline

Reference: <https://aws.amazon.com/codepipeline/>

**Benefits**

**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.

Automating your build, test, and release process allows you to quickly and easily test each code change and catch bugs while they are small and simple to fix. You can ensure the quality of your application or infrastructure code by running each change through your staging and release process.

**Configurable workflow**

AWS CodePipeline allows 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.

**Get started fast**

With AWS CodePipeline, you can immediately begin to model your software release process. There are no servers to provision or set up. CodePipeline is a fully managed continuous delivery service that connects to your existing tools and systems.

**Easy to integrate**

AWS CodePipeline can easily be extended to adapt to your specific needs. You can use our 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.

AWS CodePipeline is similar to the Jenkins Pipeline which helps to have a visual view of the end-to-end delivery process.

So, in a CodePipeline, you will typically configure the following

* **Source Code Repository** – So your source code would need to be either in AWS CodeCommit or GitHub repository.
* **Build Service** – AWS CodeBuild details will be configured as part of the pipeline.
* **Deploy** – AWS CodeDeploy will be configured into the pipeline.
* During the deploy process to different environments if any approvals are needed they could be configured as well

So if there is a code change by the developer the visual representation of Build and Deploy can be seen to be automated.

Source code repository configuration in AWS CodePipeline

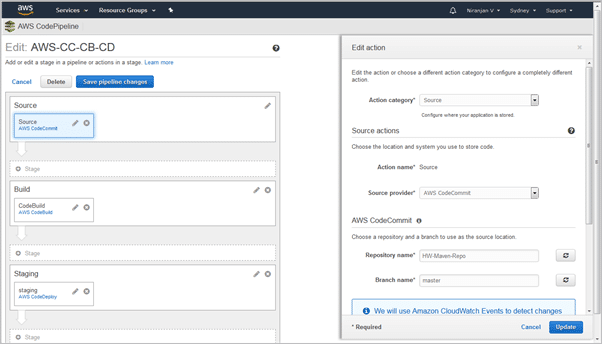
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/image016-1.png)

Image 48: Code pipeline -Configuration

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

Build configuration in AWS CodePipeline which uses Maven build

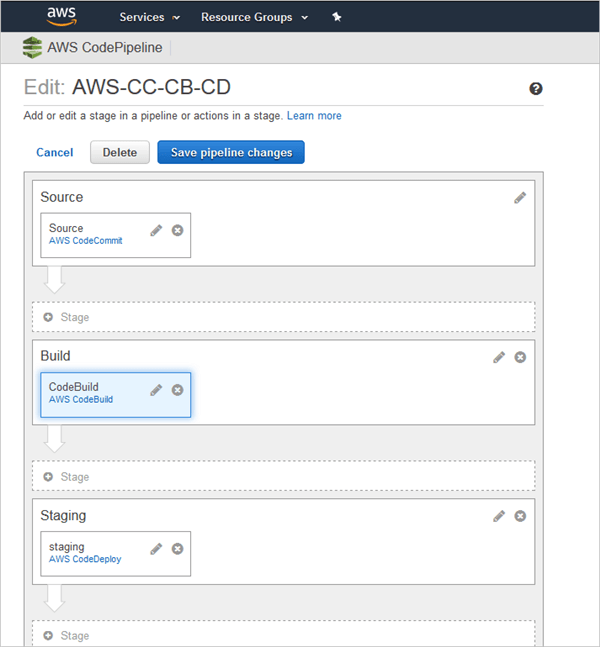
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/image017.png)

Image 49: Code pipeline Maven build

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

Deployment configuration in AWS CodePipeline

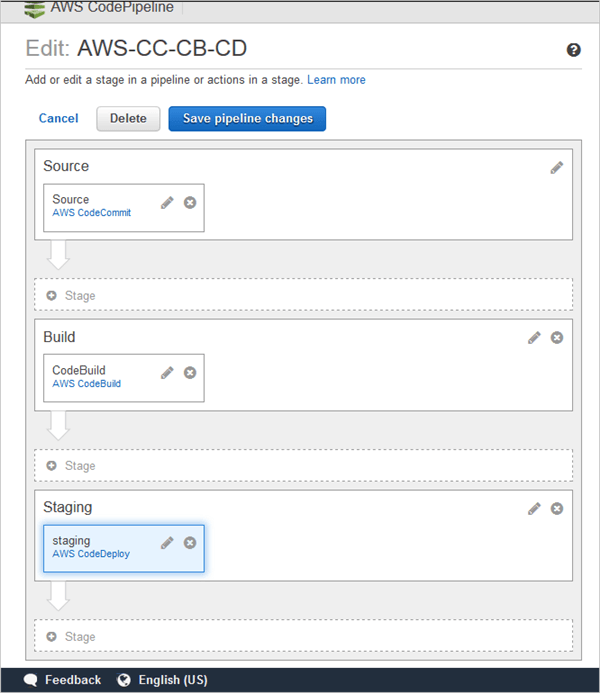
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/image018.png)

Image 50: Code pipeline-Deployment

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

Complete Execution is seen in AWS CodePipeline

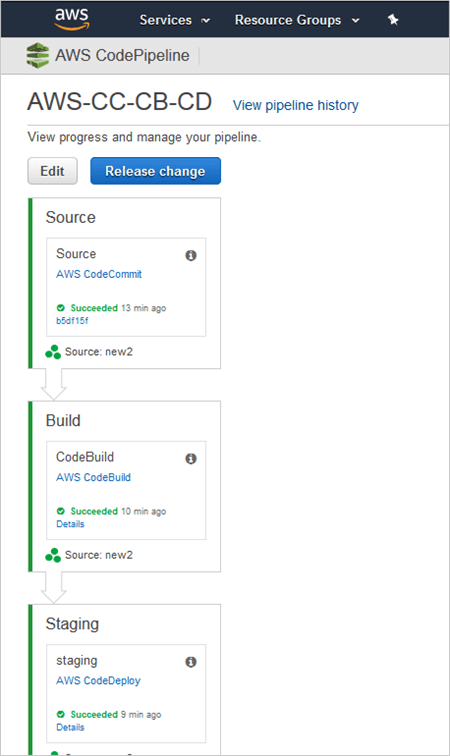
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/image019.png)

Image 51: Code pipeline Execution

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

## AWS CodeCommit

AWS CodeCommit is a secure, highly scalable, managed source control service that hosts private Git repositories. It makes it easy for teams to securely collaborate on code with contributions encrypted in transit and at rest. CodeCommit eliminates the need for you to manage your own source control system or worry about scaling its infrastructure. You can use CodeCommit to store anything from code to binaries. It supports the standard functionality of Git, so it works seamlessly with your existing Git-based tools.

Benefits

**Fully managed**

AWS CodeCommit eliminates the need to host, maintain, back up, and scale your own source control servers. The service automatically scales to meet the growing needs of your project.

**Secure**

AWS CodeCommit automatically encrypts your files in transit and at rest. CodeCommit is integrated with AWS Identity and Access Management (IAM) allowing you to customize user-specific access to your repositories.

**High availability**

AWS CodeCommit has a highly scalable, redundant, and durable architecture. The service is designed to keep your repositories highly available and accessible.

**Collaborate on code**

AWS CodeCommit helps you collaborate on code with teammates via pull requests, branching, and merging. You can implement workflows that include code reviews and feedback by default, and control who can make changes to specific branches.

**Faster development lifecycle**

AWS CodeCommit keeps your repositories close to your build, staging, and production environments in the AWS cloud. You can transfer incremental changes instead of the entire application. This allows you to increase the speed and frequency of your development lifecycle.

**Use your existing tools**

AWS CodeCommit supports all Git commands and works with your existing Git tools. You can keep using your preferred development environment plugins, continuous integration/continuous delivery systems, and graphical clients with CodeCommit.

AWS CodeCommit is a secure online version control service which hosts private Git repositories. A team need not maintain their own version control repository instead they use AWS CodeCommit to store their source code or even binaries like the WAR/JAR/EAR files generated out of the build.

With AWS CodeCommit you create a repository and every developer will clone it to their local machine, add files to it and push it back to the AWS CodeCommit repository. One uses the standard GIT commands with the AWS CodeCommit repository.

For E.g. once the AWS CodeCommit repository is cloned to local machine you would use commands like ‘git pull’, ‘git add’, ‘git commit’, ‘git push’ etc..

Illustrative AWS CodeCommit empty repository created

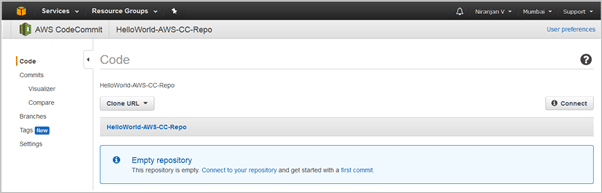
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/image020.png)

Image 52: Code commit Empty Repository

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

Clone the repository to the local machine

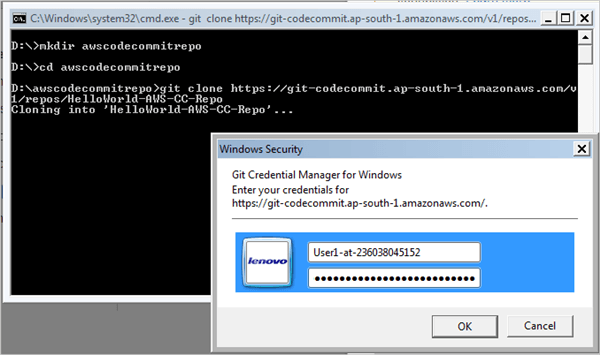
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/image021.png)

Image 53: Clone Repository

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

Files added to AWS CodeCommit repository

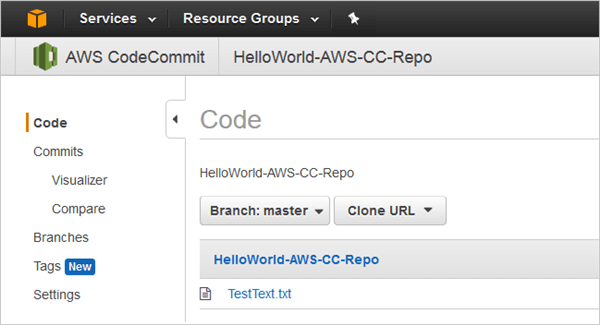
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/image022.png)\

Image 54: Code Commit Files Add

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

## 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. With CodeBuild, you don’t need to provision, manage, and scale your own build servers. CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left waiting in a queue. You can get started quickly by using prepackaged build environments, or you can create custom build environments that use your own build tools. With CodeBuild, you are charged by the minute for the compute resources you use.

**Benefits**

Fully managed build service

AWS CodeBuild eliminates the need to set up, patch, update, and manage your own build servers and software. There is no software to install or manage.

**Continuous scaling**

AWS CodeBuild scales up and down automatically to meet your build volume. It immediately processes each build you submit and can run separate builds concurrently, which means your builds are not left waiting in a queue.

**Pay as you go**

With AWS CodeBuild, you are charged based on the number of minutes it takes to complete your build. This means you no longer have to worry about paying for idle build server capacity.

**Extensible**

You can bring your own build tools and programming runtimes to use with AWS CodeBuild by creating customized build environments in addition to the prepackaged build tools and runtimes supported by CodeBuild.

**Enables continuous integration and delivery**

AWS CodeBuild belongs to a family of AWS Code Services, which you can use to create complete, automated software release workflows for continuous integration and delivery (CI/CD). You can also integrate CodeBuild into your existing CI/CD workflow. For example, you can use CodeBuild as a worker node for your existing Jenkins server setup for distributed builds.

**Secure**

With AWS CodeBuild, your build artifacts are encrypted with customer-specific keys that are managed by the AWS Key Management Service (KMS). CodeBuild is integrated with AWS Identity and Access Management (IAM), so you can assign user-specific permissions to your build projects

To implement Continuous Integration AWS CodeBuild like Jenkins fetches the latest changes of the source code from AWS CodeCommit or GitHub repository as configured and based on the build specification YAML file (created as buildspec.yml) the commands are run based on the four phases like Install, Pre-build, Build and Post-build.

Once the build is completed the artifacts (WAR/ZIP/JAR/EAR) are stored in the AWS Storage which is an S3 bucket.

Samplebuildspec.yml file

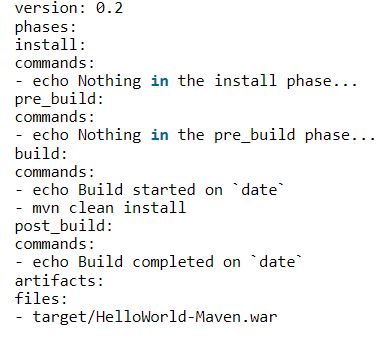


Image 55: Yaml file

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

Sample AWS Codebuild project

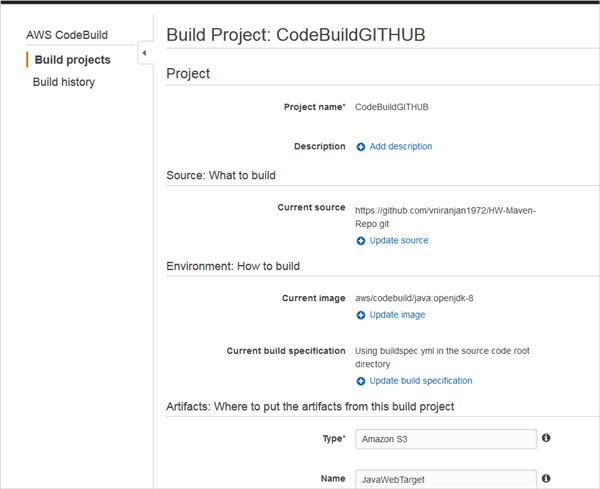
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/image023.png)

Image 56: Code Build Sample

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

Build Success

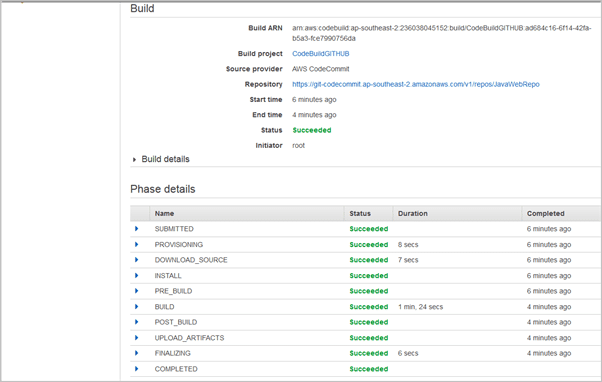
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/image024.png)

Image 57: Code Build Success

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

Artifact (WAR file) copied to S3 bucket

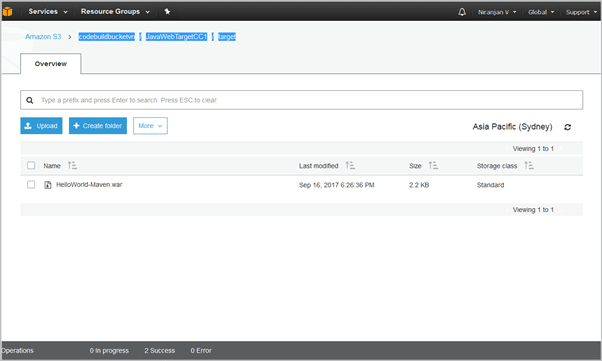
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/image025-1.png)

Image 58: WAR File copied to s3 bucket

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

## AWS CodeDeploy

AWS CodeDeploy is a fully managed deployment service that automates software deployments to a variety of compute services such as 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 AWS CodeDeploy to automate software deployments, eliminating the need for error-prone manual operations. The service scales to match your deployment needs.

**Benefits**

**Automated deployments**

AWS CodeDeploy fully automates your software deployments, allowing you to deploy reliably and rapidly. You can consistently deploy your application across your development, test, and production environments whether deploying to Amazon EC2, AWS Fargate, AWS Lambda, or your on-premises servers. The service scales with your infrastructure.

**Minimize downtime**

AWS 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.

**Centralized control**

AWS CodeDeploy allows 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 allowing 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.

**Easy to adopt**

AWS CodeDeploy is platform and language agnostic, works with any application, and provides the same experience whether you’re deploying to Amazon EC2, AWS Fargate, or AWS Lambda. You can easily reuse your existing setup code. CodeDeploy can also integrate with your existing software release process or continuous delivery toolchain (e.g., AWS CodePipeline, GitHub, Jenkins).

As the name suggests AWS Codedeploy is the deployment service which automates the deployment of the application (in this case WAR file) to the Amazon EC2 Linux or Windows instances.

Since we now have the artifacts stored in S3 bucket which was completed using AWS CodeBuild the artifacts are then picked up from the S3 bucket and deployed appropriately to the app server Tomcat or JBoss etc. in the AWS EC2 instance provisioning.

AWS CodeDeploy depends on a YAML file called appspec.yml which has instructions on the deployment to the EC2 instances.

Sample appspec.yml file where the index.html file is copied and deployed to the Apache server

**before\_install** script

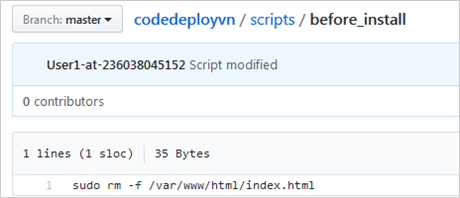
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/image026.png)

Image 59: Before Install script

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

**restart\_server** script

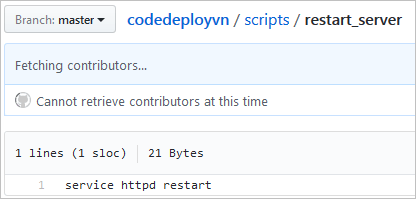
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/image027.png)

Image 60: Restart Server script

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

GitHub repo of all files needed to run AWS CodeDeploy

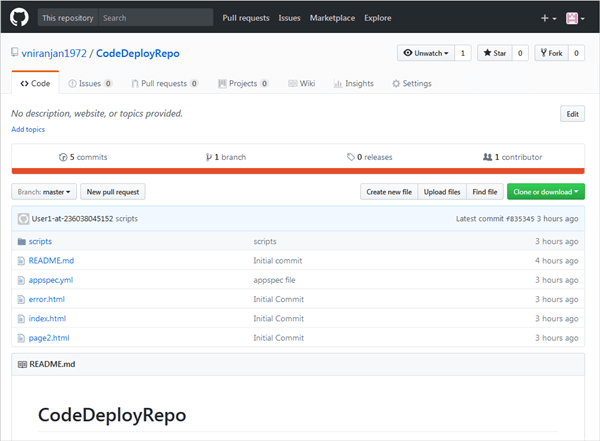
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/image028.png)

Image 61: GitHub Repository

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

Deployment execution in AWS CodeDeploy

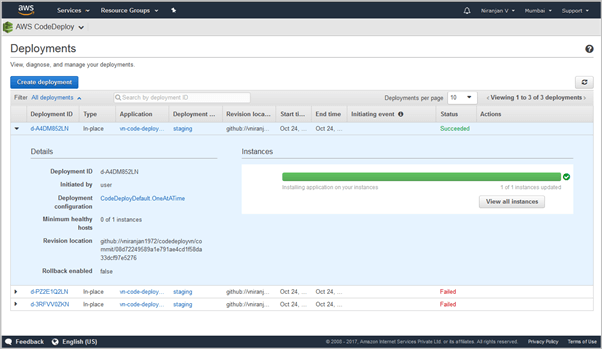
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/image029.png)

Image 62: Code Deploy-Deployment Execution

Reference: <https://www.softwaretestinghelp.com/aws-devops-tools/>

References

1. https://aws.amazon.com/devops/
2. <https://www.ibm.com/cloud/devops>