Version Controlled 2

What is DevOps?

https://aws.amazon.com/devops/what-is-devops/

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.



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. Quality assurance and security teams may also become more tightly integrated with development and operations and throughout the application lifecycle.

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.

# 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](https://aws.amazon.com/devops/what-is-devops/#microservices) and [continuous delivery](https://aws.amazon.com/devops/continuous-delivery/) let teams take ownership of services and then release updates to them quicker.

### **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](https://aws.amazon.com/devops/continuous-integration/) and [continuous delivery](https://aws.amazon.com/devops/continuous-delivery/)are practices that automate the software release process, from build to deploy.

### **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](https://aws.amazon.com/devops/continuous-integration/) and [continuous delivery](https://aws.amazon.com/devops/continuous-delivery/) to test that each change is functional and safe. [Monitoring and logging](https://aws.amazon.com/devops/what-is-devops/#monitoring) practices help you stay informed of performance in real-time.

### **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](https://aws.amazon.com/devops/what-is-devops/#iac) helps you manage your development, testing, and production environments in a repeatable and more efficient manner.

### **Improved Collaboration**

Build more effective teams under a DevOps cultural model, which emphasizes values such as ownership and accountability. Developers and operations teams [collaborate](https://aws.amazon.com/devops/what-is-devops/#communication) 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).

### **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](https://aws.amazon.com/devops/what-is-devops/#policyascode), you can define and then track compliance at scale.

# Why DevOps Matters

Software and the Internet have transformed the world and its industries, from shopping to entertainment to banking. Software no longer merely supports a business; rather it becomes an integral component of every part of a business. Companies interact with their customers through software delivered as online services or applications and on all sorts of devices. They also use software to increase operational efficiencies by transforming every part of the value chain, such as logistics, communications, and operations. In a similar way that physical goods companies transformed how they design, build, and deliver products using industrial automation throughout the 20th century, companies in today’s world must transform how they build and deliver software.

# How to Adopt a DevOps Model

## **DevOps Cultural Philosophy**

Transitioning to DevOps requires a change in culture and mindset. At its simplest, DevOps is about removing the barriers between two traditionally siloed teams, development and operations. In some organizations, there may not even be separate development and operations teams; engineers may do both. With DevOps, the two teams work together to optimize both the productivity of developers and the reliability of operations. They strive to communicate frequently, increase efficiencies, and improve the quality of services they provide to customers. They take full ownership for their services, often beyond where their stated roles or titles have traditionally been scoped by thinking about the end customer’s needs and how they can contribute to solving those needs. Quality assurance and security teams may also become tightly integrated with these teams. Organizations using a DevOps model, regardless of their organizational structure, have teams that view the entire development and infrastructure lifecycle as part of their responsibilities.

## **DevOps Practices**

There are a few key practices that help organizations innovate faster through automating and streamlining the software development and infrastructure management processes. Most of these practices are accomplished with proper tooling.

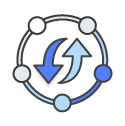
One fundamental practice is to perform very frequent but small updates. This is how organizations innovate faster for their customers. These updates are usually more incremental in nature than the occasional updates performed under traditional release practices. Frequent but small updates make each deployment less risky. They help teams address bugs faster because teams can identify the last deployment that caused the error. Although the cadence and size of updates will vary, organizations using a DevOps model deploy updates much more often than organizations using traditional software development practices.

Organizations might also use a microservices architecture to make their applications more flexible and enable quicker innovation. The microservices architecture decouples large, complex systems into simple, independent projects. Applications are broken into many individual components (services) with each service scoped to a single purpose or function and operated independently of its peer services and the application as a whole. This architecture reduces the coordination overhead of updating applications, and when each service is paired with small, agile teams who take ownership of each service, organizations can move more quickly.

However, the combination of microservices and increased release frequency leads to significantly more deployments which can present operational challenges. Thus, DevOps practices like continuous integration and continuous delivery solve these issues and let organizations deliver rapidly in a safe and reliable manner. Infrastructure automation practices, like infrastructure as code and configuration management, help to keep computing resources elastic and responsive to frequent changes. In addition, the use of monitoring and logging helps engineers track the performance of applications and infrastructure so they can react quickly to problems.

Together, these practices help organizations deliver faster, more reliable updates to their customers. Here is an overview of important DevOps practices.

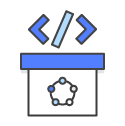
Continuous Integration



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. The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

[Learn more about continuous integration »](https://aws.amazon.com/devops/continuous-integration/)

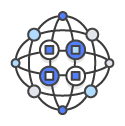
Continuous Delivery



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

[Learn more about continuous delivery and AWS CodePipeline »](https://aws.amazon.com/devops/continuous-delivery/)

Microservices



The microservices architecture is a design approach to build a single application as a set of small services. Each service runs in its own process and communicates with other services through a well-defined interface using a lightweight mechanism, typically an HTTP-based application programming interface (API). Microservices are built around business capabilities; each service is scoped to a single purpose. You can use different frameworks or programming languages to write microservices and deploy them independently, as a single service, or as a group of services.

[Learn more about Amazon EC2 Container Service »](https://aws.amazon.com/ecs/)

[Learn more about AWS Lambda »](https://aws.amazon.com/lambda/)

Infrastructure as Code



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. The cloud’s API-driven model enables developers and system administrators to interact with infrastructure programmatically, and at scale, instead of needing to manually set up and configure resources. Thus, engineers can interface with infrastructure using code-based tools and treat infrastructure in a manner similar to how they treat application code. Because they are defined by code, infrastructure and servers can quickly be deployed using standardized patterns, updated with the latest patches and versions, or duplicated in repeatable ways.

[Learn to manage your infrastructure as code with AWS CloudFormation »](https://aws.amazon.com/cloudformation/)

Configuration Management

Developers and system administrators use code to automate operating system and host configuration, operational tasks, and more. The use of code makes configuration changes repeatable and standardized. It frees developers and systems administrators from manually configuring operating systems, system applications, or server software.

[Learn how you can configure and manage Amazon EC2 and on-premises systems with Amazon EC2 Systems Manager »](https://aws.amazon.com/ec2/systems-manager/)

[Learn to use configuration management with AWS OpsWorks »](https://aws.amazon.com/opsworks/)

Policy as Code

With infrastructure and its configuration codified with the cloud, organizations can monitor and enforce compliance dynamically and at scale. Infrastructure that is described by code can thus be tracked, validated, and reconfigured in an automated way. This makes it easier for organizations to govern changes over resources and ensure that security measures are properly enforced in a distributed manner (e.g. information security or compliance with PCI-DSS or HIPAA). This allows teams within an organization to move at higher velocity since non-compliant resources can be automatically flagged for further investigation or even automatically brought back into compliance.

[Learn how you can use AWS Config and Config Rules to monitor and enforce compliance for your infrastructure »](https://aws.amazon.com/config/)

Monitoring and Logging

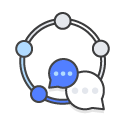


Organizations monitor metrics and logs to see how application and infrastructure performance impacts the experience of their product’s end user. By capturing, categorizing, and then analyzing data and logs generated by applications and infrastructure, organizations understand how changes or updates impact users, shedding insights into the root causes of problems or unexpected changes. Active monitoring becomes increasingly important as services must be available 24/7 and as application and infrastructure update frequency increases. Creating alerts or performing real-time analysis of this data also helps organizations more proactively monitor their services.

[Learn how you can use Amazon CloudWatch to monitor your infrastructure metrics and logs »](https://aws.amazon.com/cloudwatch/)

[Learn how you can use AWS CloudTrail to record and log AWS API calls »](https://aws.amazon.com/cloudtrail/)

Communication and Collaboration



Increased communication and collaboration in an organization is one of the key cultural aspects of DevOps. The use of DevOps tooling and automation of the software delivery process establishes collaboration by physically bringing together the workflows and responsibilities of development and operations. Building on top of that, these teams set strong cultural norms around information sharing and facilitating communication through the use of chat applications, issue or project tracking systems, and wikis. This helps speed up communication across developers, operations, and even other teams like marketing or sales, allowing all parts of the organization to align more closely on goals and projects.

DevOps Tools

The DevOps model relies on effective tooling to help teams rapidly and reliably deploy and innovate for their customers. 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. AWS provides services that are designed for DevOps and that are built first for use with the AWS cloud. These services help you use the DevOps practices described above.

[Learn about AWS DevOps services »](https://aws.amazon.com/devops/)

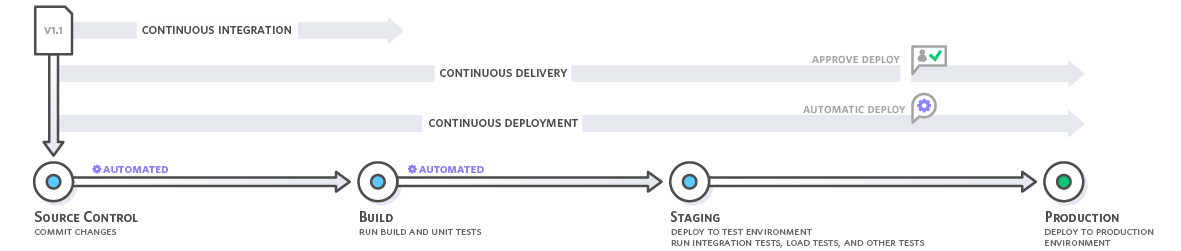
[Learn about AWS partner solutions »](https://aws.amazon.com/solutions/partners/dev-ops/)

# What is Continuous Integration?

Continuous integration is a [DevOps](https://aws.amazon.com/devops/) software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. Continuous integration most often refers to the build or integration stage of the software release process and entails both an automation component (e.g. a CI or build service) and a cultural component (e.g. learning to integrate frequently). The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

In the past, developers on a team might work in isolation for an extended period of time and only attempt to merge their changes to the master branch once their work was completed. This batched process made merging accumulated code changes difficult and time-consuming. This is compounded when small bugs accumulate for a long time without correction. These factors combined made it harder to deliver updates to customers quickly.

With continuous integration, developers frequently commit to a shared repository using a version control system such as Git. Prior to each commit, developers may choose to run local unit tests on their code as an extra verification layer before integrating. A continuous integration service detects commits to the shared repository, and automatically builds and runs unit tests on the new code changes to immediately surface any functional or integration errors.



*Continuous integration refers to the build and unit testing stages of the software release process. Every revision that is committed triggers an automated build and test.*

With [continuous delivery](https://aws.amazon.com/devops/continuous-delivery/), code changes are automatically built, tested, and prepared for a release to production. Continuous delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage.

## **Continuous Integration Benefits**



Improve Developer Productivity

Continuous integration helps your team be more productive by freeing developers from manual tasks and encouraging behaviors that help reduce the number of errors and bugs released to customers.



Find and Address Bugs Quicker

With more frequent testing, your team can discover and address bugs earlier before they grow into larger problems later.



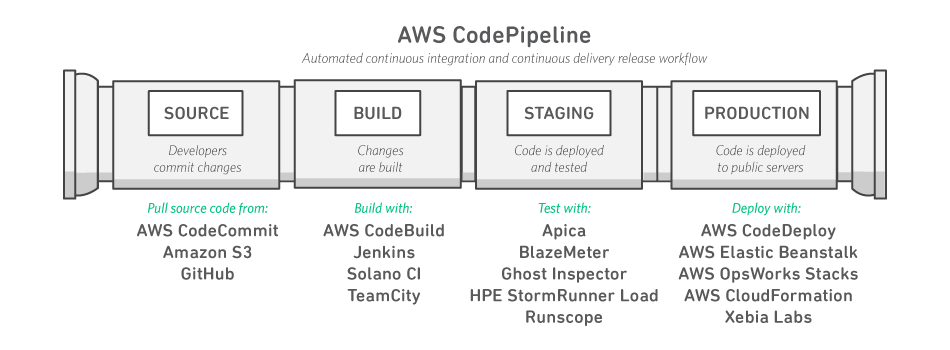
Deliver Updates Faster

Continuous integration helps your team deliver updates to their customers faster and more frequently.

# AWS CodePipeline

# https://aws.amazon.com/codepipeline/

AWS CodePipeline is a [continuous integration](https://aws.amazon.com/devops/continuous-integration/) and [continuous delivery](https://aws.amazon.com/devops/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. You can easily build out an end-to-end solution by using our pre-built plugins for popular third-party services like GitHub or integrating your own custom plugins into any stage of your release process. With AWS CodePipeline, you only pay for what you use. There are no upfront fees or long-term commitments.



Benefits



Rapid Delivery

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



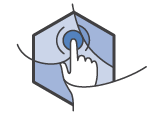
Improved Quality

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



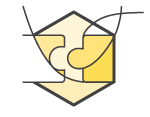
Configurable Workflow

AWS CodePipeline allows you to model the different stages of your software release process through a graphical user interface. You can specify the tests to run and 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

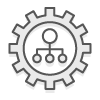
AWS CodePipeline can easily be extended to adapt to your specific needs. You can use our pre-built plugins or your own custom plugins in any step of your release process. For example, you can pull your source code from GitHub, use your on-premises Jenkins build server, run load tests using a third-party service, or pass on deployment information to your custom operations dashboard.

# AWS CodeBuild

<https://aws.amazon.com/codebuild/>

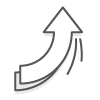
AWS CodeBuild is a fully managed build service that compiles source code, runs tests, and produces software packages that are ready to deploy. With CodeBuild, you don’t need to provision, manage, and scale your own build servers. CodeBuild scales continuously and processes multiple builds concurrently,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



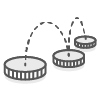
Fully Managed Build Service

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.



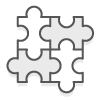
Continuous Scaling

AWS CodeBuild scales 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.



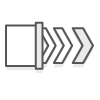
Pay as You Go

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](https://aws.amazon.com/products/developer-tools/), which you can use to create complete, automated software release workflows for [continuous integration](https://aws.amazon.com/devops/continuous-integration/) and [delivery](https://aws.amazon.com/devops/continuous-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](https://aws.amazon.com/kms/) (KMS). CodeBuild is integrated with [AWS Identity and Access Management](https://aws.amazon.com/iam/) (IAM), so you can assign user-specific permissions to your build projects.

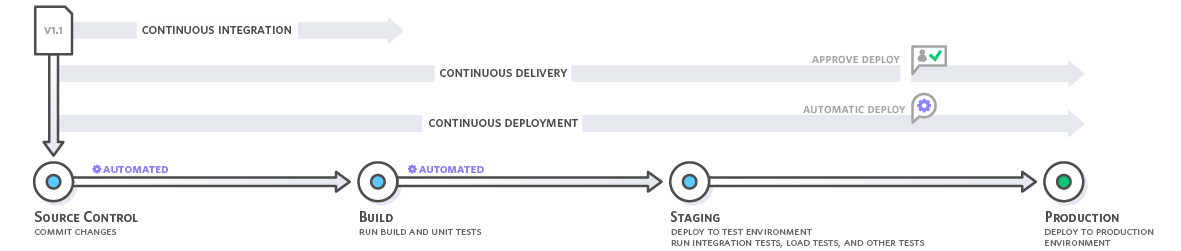
# What is Continuous Delivery?

<https://aws.amazon.com/devops/continuous-delivery/>

Continuous delivery is a [DevOps](https://aws.amazon.com/devops/) software development practice where code changes are automatically built, tested, and prepared for a release to production. It expands upon [continuous integration](https://aws.amazon.com/devops/continuous-integration/) by deploying all code changes to a testing environment and/or a production environment after the build stage. When continuous delivery is implemented properly, developers will always have a deployment-ready build artifact that has passed through a standardized test process.

With continuous delivery, every code change is built, tested, and then pushed to a non-production testing or staging environment. There can be multiple, parallel test stages before a production deployment. In the last step, the developer approves the update to production when they are ready. This is different from continuous deployment, where the push to  production happens automatically without explicit approval.

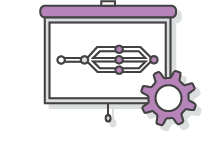
Continuous delivery lets developers automate testing beyond just unit tests so they can verify application updates across multiple dimensions before deploying to customers. These tests may include UI testing, load testing, integration testing, API reliability testing, etc. This helps developers more thoroughly validate updates and pre-emptively 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.



*Continuous delivery automates the entire software release process. Every revision that is committed triggers an automated flow that builds, tests, and then stages the update. The final decision to deploy to a live production environment is triggered by the developer.*

With continuous deployment, revisions are deployed to a production environment automatically without explicit approval from a developer, making the entire software release process automated.

Continuous Delivery Benefits



Automate the Software Release Process

Continuous delivery lets your team automatically build, test, and prepare code changes for release to production so that your software delivery is more efficient and rapid.



Improve Developer Productivity

These practices help your team be more productive by freeing developers from manual tasks and encouraging behaviors that help reduce the number of errors and bugs deployed to customers.



Find and Address Bugs Quicker

Your team can discover and address bugs earlier before they grow into larger problems later with more frequent and comprehensive testing. Continuous delivery lets you more easily perform additional types of tests on your code because the entire process has been automated.



Deliver Updates Faster

Continuous delivery helps your team deliver updates to customers faster and more frequently. When continuous delivery is implemented properly, you will always have a deployment-ready build artifact that has passed through a standardized test process.

### **Microservices**

The microservices architecture is a design approach to build a single application as a set of small services. Each service runs in its own process and communicates with other services through a well-defined interface using a lightweight mechanism, typically an HTTP-based application programming interface (API). Microservices are built around business capabilities; each service is scoped to a single purpose. You can use different frameworks or programming languages to write microservices and deploy them independently, as a single service, or as a group of services.

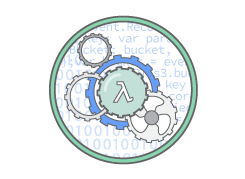
[Learn more about Amazon EC2 Container Service »](https://aws.amazon.com/ecs/)

[Learn more about AWS Lambda »](https://aws.amazon.com/lambda/)

# Amazon EC2 Container Service

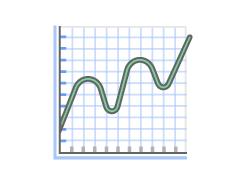
# AWS Lambda

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running. 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. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.



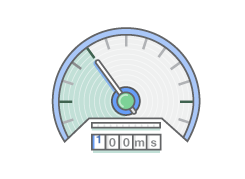
No Servers to Manage

AWS Lambda automatically runs your code without requiring you to provision or manage servers. Just write the code and upload it to Lambda.



Continuous Scaling

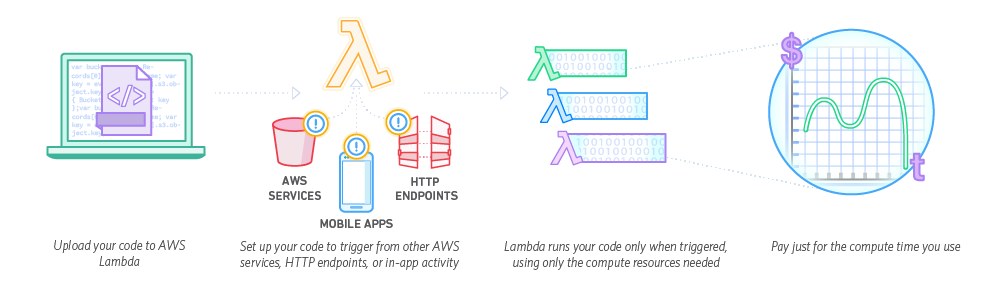
AWS Lambda automatically scales your application by running code in response to each trigger. Your code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload.



Subsecond Metering

With AWS Lambda, you are charged for every 100ms your code executes and the number of times your code is triggered. You don't pay anything when your code isn't running.

How It Works

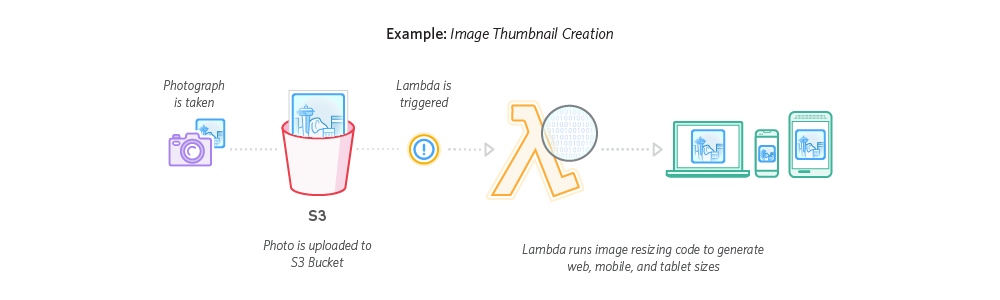


# Data Processing

You can use AWS Lambda to execute code in response to triggers such as changes in data, shifts in system state, or actions by users. Lambda can be directly triggered by AWS services such as S3, DynamoDB, Kinesis, SNS, and CloudWatch, allowing you to build a variety of real-time [serverless](https://aws.amazon.com/serverless/) data processing systems. You can process more complex jobs by coordinating multiple Lambda functions with [AWS Step Functions](https://aws.amazon.com/step-functions/).

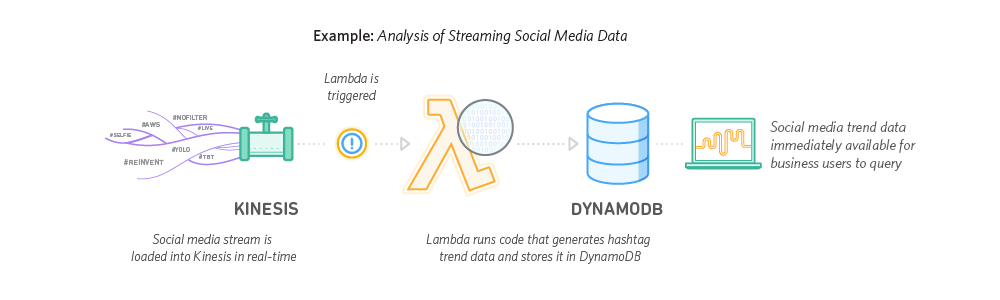
## **Real-time File Processing**

You can use Amazon S3 to trigger AWS Lambda to process data immediately after an upload. For example, you can use Lambda to thumbnail images, transcode videos, index files, process logs, validate content, and aggregate and filter data in real-time.



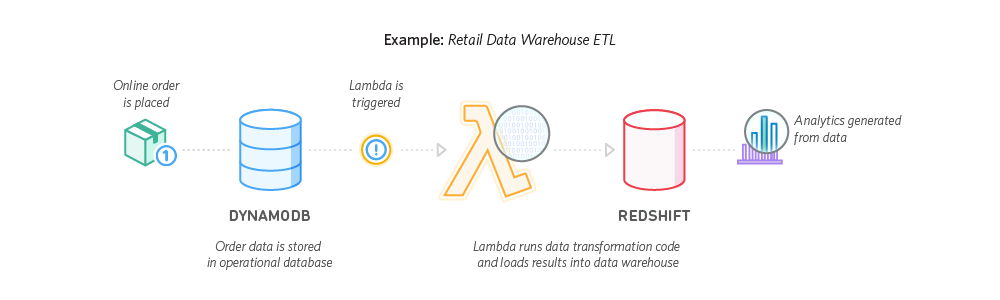
## **Real-time Stream Processing**

You can use AWS Lambda and Amazon Kinesis to process real-time streaming data for application activity tracking, transaction order processing, click stream analysis, data cleansing, metrics generation, log filtering, indexing, social media analysis, and IoT device data telemetry and metering. Reference Architecture: [Diagram](https://s3.amazonaws.com/awslambda-reference-architectures/stream-processing/lambda-refarch-streamprocessing.pdf) | [Sample Code](https://github.com/awslabs/lambda-refarch-streamprocessing)



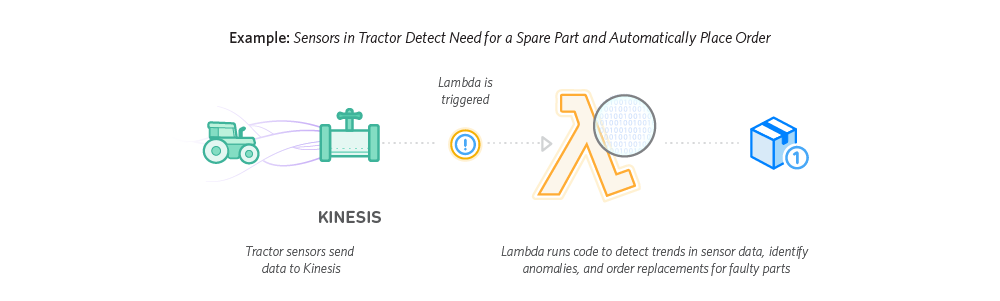
Extract, Transform, Load

You can use AWS Lambda to perform data validation, filtering, sorting, or other transformations for every data change in a DynamoDB table and load the transformed data to another data store. Reference Architecture: [Diagram](https://s3.amazonaws.com/awslambda-reference-architectures/extract-transform-load/lambda-refarch-etl.pdf)



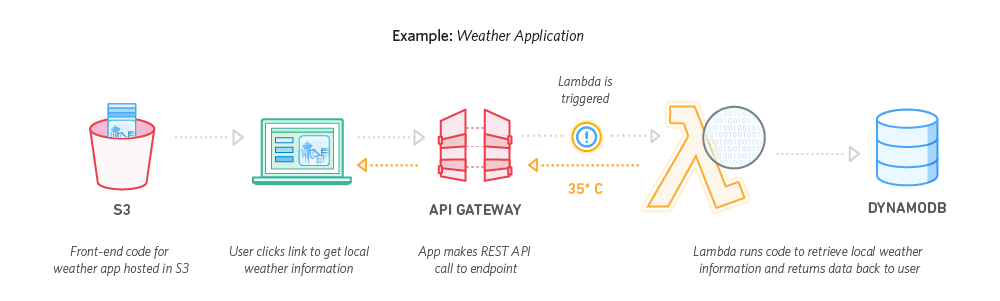
IoT Backends

You can build backends using AWS Lambda and Amazon Kinesis for Internet of Things (IoT) device data telemetry and analysis. Reference Architecture: [Diagram](https://s3.amazonaws.com/awslambda-reference-architectures/iot-backend/lambda-refarch-iotbackend.pdf) | [Sample Code](https://github.com/awslabs/lambda-refarch-iotbackend)



Web Applications

By combining AWS Lambda with other AWS services, developers can build powerful web applications that automatically scale up and down and run in a highly available configuration across multiple data centers – with zero administrative effort required for scalability, back-ups or multi-data center redundancy. Reference Architecture: [Diagram](https://s3.amazonaws.com/awslambda-reference-architectures/web-app/lambda-refarch-webapp.pdf) | [Sample Code](https://github.com/awslabs/lambda-refarch-webapp/)



## **Infrastructure as Code**

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

# 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](https://aws.amazon.com/cloudformation/aws-cloudformation-templates/) or create your own templates to describe the AWS resources, and any associated dependencies or runtime parameters, required to run your application. You don’t need to figure out the order for provisioning AWS services or the subtleties of making those dependencies work. CloudFormation takes care of this for you. After the AWS resources are deployed, you can modify and update them in a controlled and predictable way, in effect applying version control to your AWS infrastructure the same way you do with your software. You can also visualize your templates as diagrams and edit them using a drag-and-drop interface with the [AWS CloudFormation Designer](https://aws.amazon.com/cloudformation/details/#designer).

You can deploy and update a template and its associated collection of resources (called a stack) by using the AWS Management Console, AWS Command Line Interface, or APIs. CloudFormation is available at no additional charge, and you pay only for the AWS resources needed to run your applications.

# AWS OpsWorks

### **[AWS OpsWorks for Chef Automate](https://aws.amazon.com/opsworks/chefautomate/)**

AWS OpsWorks for Chef Automate provides a fully managed Chef server and suite of automation tools that give you workflow automation for continuous deployment, automated testing for compliance and security, and a user interface that gives you visibility into your nodes and their status. The Chef server gives you full stack automation by handling operational tasks such as software and operating system configurations, package installations, database setups, and more. The Chef server centrally stores your configuration tasks and provides them to each node in your compute environment at any scale, from a few nodes to thousands of nodes. OpsWorks for Chef Automate is completely compatible with tooling and cookbooks from the Chef community and automatically registers new nodes with your Chef server.

With OpsWorks for Chef Automate, you are charged based on the number of nodes connected to your Chef server and the time those nodes are running, and you also pay for the underlying Amazon Elastic Compute Cloud (Amazon EC2) instance running your Chef server.

### **[AWS OpsWorks Stacks](https://aws.amazon.com/opsworks/stacks/)**

AWS OpsWorks Stacks lets you manage applications and servers on AWS and on-premises. With OpsWorks Stacks, you can model your application as a stack containing different layers, such as load balancing, database, and application server. You can deploy and configure Amazon EC2 instances in each layer or connect other resources such as Amazon RDS databases. OpsWorks Stacks lets you set automatic scaling for your servers based on preset schedules or in response to changing traffic levels, and it uses lifecycle hooks to orchestrate changes as your environment scales. You run Chef recipes using Chef Solo, allowing you to automate tasks such as installing packages and programming languages or frameworks, configuring software, and more.

There is no additional charge to use OpsWorks Stacks for Amazon EC2, and you only pay for the underlying resources created using OpsWorks Stacks.

# Amazon EC2 Systems Manager

Amazon EC2 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. By providing a management approach that is designed for the scale and agility of the cloud but extends into your on-premises data center, EC2 Systems Manager makes it easier for you to seamlessly bridge your existing infrastructure with AWS.

EC2 Systems Manager is easy to use. Simply access EC2 Systems Manager from the EC2 Management Console, select the instances you want to manage, and define the management tasks you want to perform. EC2 Systems Manager is available now at no cost to manage both your EC2 and on-premises resources.

## **Monitoring and Logging**

# 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. Amazon CloudWatch can monitor AWS resources such as Amazon EC2 instances, Amazon DynamoDB tables, and Amazon RDS DB instances, as well as custom metrics generated by your applications and services, and any log files your applications generate. You can use Amazon CloudWatch to gain system-wide visibility into resource utilization, application performance, and operational health. You can use these insights to react and keep your application running smoothly.

Features & Benefits



Monitor Amazon EC2

View metrics for CPU utilization, data transfer, and disk usage activity from Amazon EC2 instances (Basic Monitoring) for no additional charge. For an additional charge, CloudWatch provides Detailed Monitoring for EC2 instances with higher resolution and metric aggregation. No additional software needs to be installed.

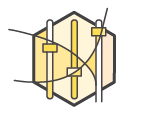
[Learn more »](https://aws.amazon.com/cloudwatch/details#amazon-ec2-monitoring)



Monitor Other AWS Resources

Monitor metrics on Amazon DynamoDB tables, Amazon EBS volumes, Amazon RDS DB instances, Amazon Elastic MapReduce job flows, Elastic Load Balancers, Amazon SQS queues, Amazon SNS topics, and more for no additional charge. No additional software needs to be installed.

[Learn more »](https://aws.amazon.com/cloudwatch/details#other-aws-resource-monitoring)



Monitor Custom Metrics

Submit Custom Metrics generated by your own applications via a simple API request and have them monitored by Amazon CloudWatch. You can send and store metrics that are important to your application’s operational performance to help you troubleshoot and spot trends.

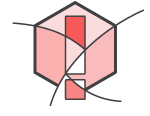
[Learn more »](https://aws.amazon.com/cloudwatch/details#custom-metrics-monitoring)



Monitor and Store Logs

You can use CloudWatch Logs to monitor and troubleshoot your systems and applications using your existing system, application, and custom log files. You can send your existing system, application, and custom log files to CloudWatch Logs and monitor these logs in near real-time. This can help you better understand and operate your systems and applications, and you can store your logs using highly durable, low-cost storage for later access.

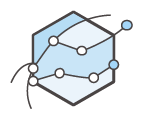
[Learn more »](https://aws.amazon.com/cloudwatch/details#log-monitoring)



Set Alarms

Set alarms on any of your metrics to send you notifications or take other automated actions. For example, when a specific Amazon EC2 metric crosses your alarm threshold, you can use Auto Scaling to dynamically add or remove EC2 instances or send you a notification.

[Learn more »](https://aws.amazon.com/cloudwatch/details#alarms)



View Graphs and Statistics

Amazon Cloudwatch Dashboards enable you to create re-usable graphs of AWS resources and custom metrics so you can quickly monitor operational status and identify issues at a glance. Metric data is kept for a period of fifteen months enabling you to view up to the minute data and also historical data. Amazon CloudWatch can load all the metrics in your account for search and graphing with the AWS Management Console. This includes logs, AWS resource metrics, and application metrics that you provide.

[Learn more »](https://aws.amazon.com/cloudwatch/details/#graphs-and-statistics)



Monitor and React to Resource Changes

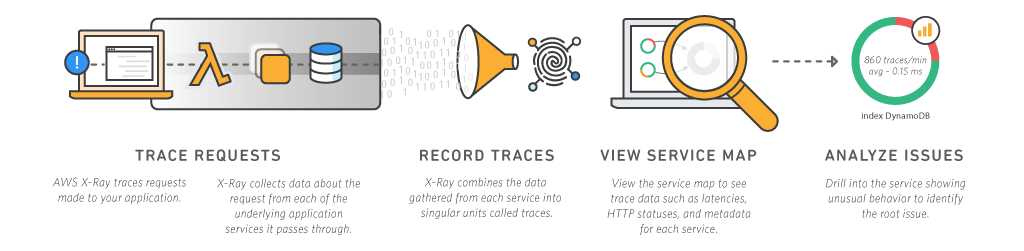
CloudWatch Events provides a stream of events describing changes to your AWS resources. You can easily build workflows that automatically take actions you define, such as invoking an AWS Lambda function, when an event of interest occurs.

[Learn more »](https://aws.amazon.com/cloudwatch/details/#events)

# AWS X-Ray

AWS X-Ray helps developers analyze and debug production, distributed applications, such as those built using a microservices architecture. With X-Ray, you can understand how your application and its underlying services are performing to identify and troubleshoot the root cause of performance issues and errors. X-Ray provides an end-to-end view of requests as they travel through your application, and shows a map of your application’s underlying components. You can use X-Ray to analyze both applications in development and in production, from simple three-tier applications to complex microservices applications consisting of thousands of services.

How It Works



Benefits



Review Request Behavior

AWS X-Ray traces user requests as they travel through your entire application. It aggregates the data generated by the individual services and resources that make up your application, providing you an end-to-end view of how your application is performing.



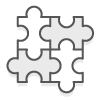
Discover Application Issues

With AWS X-Ray, you can glean insights into how your application is performing and discover root causes. With X-Ray’s tracing features, you can follow request paths to pinpoint where in your application and what is causing performance issues. X-Ray provides annotations so you can append metadata to traces, making it possible to tag and filter trace data so you can discover patterns and diagnose issues.



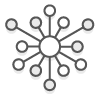
Improve Application Performance

AWS X-Ray helps you identify performance bottlenecks. X-Ray’s service maps let you see relationships between services and resources in your application in real time. You can easily detect where high latencies are occurring, visualize node and edge latency distribution for services, and then drill down into the specific services and paths impacting application performance.



Ready to Use with AWS

AWS X-Ray works with [Amazon EC2](https://aws.amazon.com/ec2/), [Amazon EC2 Container Service](https://aws.amazon.com/ecs/) (Amazon ECS), [AWS Lambda](https://aws.amazon.com/lambda/), and [AWS Elastic Beanstalk](https://aws.amazon.com/elasticbeanstalk/). You can use X-Ray with applications written in Java, Node.js, and .NET that are deployed on these services.



Designed for a Variety of Applications

AWS X-Ray works for both simple and complex applications, either in development or in production. You can analyze simple asynchronous event calls, three-tier web applications, or complex microservices applications consisting of thousands of services. With X-Ray, you can trace requests made to applications that span multiple AWS accounts, AWS Regions, and Availability Zones.

## **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

WS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, [.NET](https://aws.amazon.com/net/), PHP, Node.js, Python, Ruby, Go, and [Docker](https://aws.amazon.com/docker/) on familiar servers such as Apache, Nginx, Passenger, and [IIS](https://aws.amazon.com/windows/).

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.

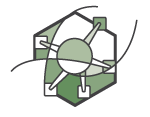
There is no additional charge for Elastic Beanstalk - you pay only for the AWS resources needed to store and run your applications.

## **Version Control**

Host secure, highly scalable Git repositories in the cloud.

AWS CodeCommit is a fully-managed source control service that makes it easy for companies to host secure and highly scalable private Git repositories. CodeCommit eliminates the need to operate your own source control system or worry about scaling its infrastructure. You can use CodeCommit to securely store anything from source code to binaries, and it works seamlessly with your existing Git 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 assign user-specific permissions to your repositories.



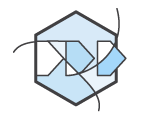
High Availability

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



Store Anything

AWS CodeCommit allows you to store any type of file, and there are no repository size limits. This allows you to store and version application assets, like images and libraries, along with your code.



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.