**Modern Development Process:**

**SERVERLESS TECHNOLOGIES**

Simplify infrastructure management with serverless technologies. Reducing time spent on routine environment management frees time to focus on business logic. Serverless technologies let you build applications without thinking about the underlying infrastructure

**INFRASTRUCTURE AS CODE**

Standardize operations by modelling all applications and infrastructure as code (IaC). IaC allows you to model your application and all the supporting infrastructure resources as code. This code serves as the single source of truth for all your infrastructure, and updates to your infrastructure are made directly in code. This eliminates the need for manual updates to infrastructure which can be error-prone.

**MICROSERVICES**

Componentize applications using microservices. When applications are built with modular, independent components, called microservices, release velocity can increase because changes to any component are easier to make. Microservices make applications easier to scale and faster to develop, enabling innovation and accelerating time-to-market.

**CI/CD**

Update applications and infrastructure quickly by automating continuous integration and continuous delivery (CI/CD). Using CI/CD, you build, test, and deploy each code change with an automated process. Automated CI/CD practices help you release better features faster.

**SECURITY & COMPLIANCE**

Secure the entire application lifecycle by automating security. Building authentication, authorization, and compliance auditing directly into every component of your application, and securing your infrastructure from intrusions protects your users and your business. Automating the evaluation of security configuration and implementation throughout the development process and in production reduces the time it takes to secure your application.

**OBSERVABILITY**

Improve application performance by increasing observability. Collecting, aggregating, and correlating metrics, logs, and traces builds insights into application and customer behaviour. We call this observability. Observability allows you to rapidly detect and respond to issues and opportunities to improve application performance.

**CULTURE**

Enable experimentation by creating small autonomous teams. When teams own the complete application lifecycle, including taking customer input, planning the roadmap, and developing and operating the application, they have the impetus and autonomy to improve the customer experience.

**CI/CD, IAC and Monitoring:**

[INFRASTRUCTURE PROVISIONING](https://aws.amazon.com/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.

[CLOUD-BASED IDE](https://aws.amazon.com/cloud9/)

AWS Cloud9 is a cloud-based integrated development environment (IDE) that lets you write, run, and debug your code with just a browser. It includes a code editor, debugger, and terminal. AWS Cloud9 comes prepackaged with essential tools for popular programming languages so you don’t need to install files or configure your development machine to start new projects.

[USER AUDITING](https://aws.amazon.com/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.

[VERSION CONTROL](https://aws.amazon.com/codepipeline/)

AWS CodeCommit is a fully-managed [source control](https://aws.amazon.com/devops/source-control/) service that hosts secure Git-based repositories

[CONTINUOUS INTEGRATION](https://aws.amazon.com/codepipeline/)

AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy

[CONTINUOUS DELIVERY](https://aws.amazon.com/codepipeline/)

AWS CodePipeline is a fully managed [continuous delivery](https://aws.amazon.com/devops/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.

[MONITORING](https://aws.amazon.com/cloudwatch/)

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

[DEBUGGING](https://aws.amazon.com/xray/)

AWS X-Ray helps developers analyze and debug distributed applications in production or under development, such as those built using a microservices architecture. With X-Ray, you can understand how your application and its underlying services are performing so you can identify and troubleshoot the root cause of performance issues and errors.

**Customizing AMI’s:**

* If you use cfn-init the config is applied every time you run the CloudFormation script
* When you launch an instance in Amazon EC2, you have the option of passing **user data** to the instance that can be used to perform common automated configuration tasks and even run scripts after the instance starts
* By **default**, user data scripts and cloud-init directives **run only during the boot cycle when you first launch an instance**. You can update your configuration to ensure that your user data scripts and *cloud-init* directives run every time you restart your instance.
* You can pass two types of user data to Amazon EC2: **shell scripts and cloud-init directives**
* **Instance metadata** is data about your instance that you can use to configure or manage the running instance.
* To create a custom platform, you build an AMI from one of the supported operating systems— You can create your own Elastic Beanstalk platform using **Packer**, which is an open-source tool for creating machine images for many platforms, including AMIs for use with Amazon Elastic Compute Cloud (Amazon EC2) or Elastic Beanstalk.

**CD/CI:**

* Continuous integration (CI) is a software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run.
* Continuous delivery (CD) is a software development practice where code changes are automatically built, tested, and prepared for production release. It expands on continuous integration by deploying all code changes to a testing environment, a production environment, or both after the build stage has been completed

CI with CodeDeploy:

Resources

* [https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments.html](https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-rollback-and-redeploy.html)
* <https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-rollback-and-redeploy.html>
* <https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-structure-hooks.html>
* <https://aws.amazon.com/blogs/compute/implementing-safe-aws-lambda-deployments-with-aws-codedeploy/>

Key Points

* AppSpec file describes your build and can reference versions of lambda functions to use
* You bundle your deployable content and the AppSpec file into an archive file, and then upload it to an Amazon S3 bucket or a GitHub repository. This archive file is called an application revision (or simply a revision).
* The CodeDeploy agent is a software package that, when installed and configured on an instance, makes it possible for that instance to be used in CodeDeploy deployments
* CodeDeploy rolls back deployments by redeploying a previously deployed revision of an application as a new deployment. These rolled-back deployments are technically new deployments, with new deployment IDs, **rather than restored versions of a previous deployment.**
* Only deployments that use the EC2/On-Premises compute platform can use **in-place deployments**, The application on each instance in the deployment group is stopped, the latest application revision is installed, and the new version of the application is started and validated.
* Blue/green deployment reroutes traffic from your application's original environment to a replacement environment. Your environment depends on your CodeDeploy application's compute platform.

**CD with CodePipeline:**

Resources

* <https://docs.aws.amazon.com/codepipeline/latest/userguide/best-practices.html>
* <https://docs.aws.amazon.com/codepipeline/latest/userguide/concepts.html>

Key Points

* **A pipeline must contain at least two stages**.
* The first stage of a pipeline must **contain at least one source action**, and can only contain source actions.
* At least one stage in each pipeline must contain an action that is not a source action.
* These are the valid *actionTypeId* categories for CodePipeline:
* Source
* Build
* Approval
* Deploy
* Test
* Invoke
* When you create a pipeline, CodePipeline integrates with AWS products and services that act as action providers in each stage of your pipeline.
* A source action stage with a default name of “Source.”
* A build action stage with a default name of “Build.”
* A deploy action stage with a default name of “Staging.”
* You can create pipelines that integrate with third-party products such as GitHub and Jenkins.
* As a best practice, when you use a Jenkins build provider for your pipeline’s build or test action, install Jenkins on an Amazon EC2 instance and configure a separate EC2 instance profile.

**Serverless with ElasticBeanstalk:**

Resources

* <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/applications-sourcebundle.html>.
* [https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.rolling-version-deploy.html#environments-cfg-rollingdeployments-method](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.rolling-version-deploy.html) <https://docs.aws.amazon.com/lambda/latest/dg/best-practices.html>
* <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/ebextensions.html>
* <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.managing.ec2.html>

Key Points

* When you use the AWS Elastic Beanstalk console to deploy a new application or an application version, you'll need to upload a source bundle. Your source bundle must meet the following requirements:

- Consist of a single ZIP file or WAR file (you can include multiple WAR files inside your ZIP file)

- Not exceed 512 MB

- Not include a parent folder or top-level directory (subdirectories are fine)

* You can add AWS Elastic Beanstalk configuration files (.ebextensions) to your web application's source code to configure your environment and customize the AWS resources that it contains. AWS recommend using YAML for your configuration files, because it's more readable than JSON.
* Blue/green deployments require that your environment runs independently of your production database, if your application uses one. If your environment has an Amazon RDS DB instance attached to it, the data will not transfer over to your second environment and will be lost if you terminate the original environment.

**ElasticBeanstalk application deployment:**

*Environment creation creates the following resources:*

* *An Amazon Elastic Compute Cloud (Amazon EC2) virtual machine configured to run web apps on the platform that you choose.*
* *Instance security group – An Amazon EC2 security group configured to allow ingress on port 80 from the ELB.*
* *An Elastic Load Balancing load balancer configured to distribute requests to the instances running your application.*
* *Load balancer security group – An Amazon EC2 security group configured to allow ingress on port 80. This resource lets HTTP traffic from the internet reach the load balancer.*
* *Auto Scaling group – An Auto Scaling group configured to replace an instance if it is terminated or becomes unavailable.*
* *Amazon S3 bucket – A storage location for your source code, logs, and other artifacts that are created when you use Elastic Beanstalk.*
* *Amazon CloudWatch alarms – Two CloudWatch alarms that monitor the load on the instances in your environment and are triggered if the load is too high or too low. When an alarm is triggered, your Auto Scaling group scales up or down in response.*
* *AWS CloudFormation stack – Elastic Beanstalk uses AWS CloudFormation to launch the resources in your environment and propagate configuration changes.*
* *Domain name – A domain name that routes to your web app in the form subdomain.region.elasticbeanstalk.com.*