Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA   Batch - 61
EADC Assignment 1

AWS CodePipeline

1. CodePipeline

- CodePipeline is a continuous delivery service that automates the release process of software applications.
- CodePipeline integrates with various AWS and third-party services to build, test, and deploy code changes.
- CodePipeline consists of pipelines, stages, actions, artifacts, and transitions.

2. CI/CD (Continuous Integration and Continuous Delivery)

- Continuous delivery is a practice of releasing software changes frequently and reliably.
- Continuous integration is a practice of merging code changes from multiple developers into a shared repository and running automated tests.
- CodePipeline supports both practices by enabling fast and consistent delivery of code changes to different environments.

3. Advantages of CodePipeline

- CodePipeline allows you to create custom pipelines that suit your specific needs and preferences.
- CodePipeline provides predefined action types for common tasks such as source, build, test, deploy, and approval.
- CodePipeline also supports custom action types for integrating with

your own tools and services.

- CodePipeline enables you to monitor and control your pipelines through the console, CLI, or API.

4. <u>CodePipeline components overview</u>

- A pipeline is a workflow that defines how your code changes are delivered.
- A stage is a logical group of actions that perform a specific task in the pipeline.
- An action is an individual task that operates on artifacts (files or data) in a stage.
- A transition is a link between two stages that controls the flow of artifacts and executions.
- An artifact is a file or set of files that is used as input or output of an action.
- A source revision is a specific version of an artifact that is processed by the pipeline.

5. <u>Using the CodePipeline</u>

- To get started with CodePipeline, you need to create a pipeline and configure its stages and actions.
- You can create a pipeline using the console, the CLI, the API, or CloudFormation.
- You can configure your pipeline using the pipeline settings page in the console, or by editing the pipeline JSON file.
- You can also edit, delete, or clone an existing pipeline.

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA   Batch - 61
EADC Assignment 1

a) Working with pipelines

- To work with pipelines, you need to understand the pipeline structure, the pipeline execution, and the pipeline artifacts.
- The pipeline structure defines the stages and actions that make up your pipeline, and the transitions that connect them.
- The pipeline execution is the process of running your pipeline and performing the actions in each stage.
- The pipeline artifacts are the files or data that are used as input or output of the actions in your pipeline.

b) Working with stages

- To work with stages, you need to understand the stage structure, the stage transitions, and the stage retries.
- The stage structure defines the actions that are performed in a stage, and the order and concurrency of the actions.
- The stage transitions control the flow of artifacts and executions between stages, and can be enabled or disabled manually or automatically.
- The stage retries allow you to retry a failed stage or action without restarting the entire pipeline.

c) Working with actions

- To work with actions, you need to understand the action structure, the action categories, the action providers, and the action variables.
- The action structure defines the name, type, configuration, input, output, and run order of an action.

- The action categories are the logical groups of actions based on their purpose, such as source, build, test, deploy, approval, or invoke.
- The action providers are the services or tools that perform the actions, such as AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, AWS Lambda, or Jenkins.
- The action variables are the dynamic values that are passed between actions in a pipeline, such as source revision IDs, artifact locations, or output values.

d) <u>Working with artifacts</u>

- To work with artifacts, you need to understand the artifact structure, the artifact store, and the artifact encryption.
- The artifact structure defines the name, location, and revision of an artifact that is used as input or output of an action.
- The artifact store is the Amazon S3 bucket where the artifacts are stored and retrieved by the pipeline.
- The artifact encryption is the process of encrypting and decrypting the artifacts using AWS KMS keys.

e) <u>Working with transitions</u>

- To work with transitions, you need to understand the transition structure, the transition types, and the transition settings.
- The transition structure defines the source and destination stages of a transition, and the status of the transition.
- The transition types are the ways of controlling the flow of artifacts and executions between stages, such as automatic, manual, or disabled.

- The transition settings are the options that affect the behavior of the transitions, such as stage retry mode, stage transition timeout, or transition change detection options.

6. Pipeline execution and flow

A pipeline execution is a set of changes released by a pipeline. CodePipeline can process multiple executions at the same time, using different execution modes to handle how each execution progresses through the pipeline. The execution modes are:

- SUPERSEDED mode : This is the default mode, where a more recent execution can overtake an older one. An older execution can be superseded by a newer one at the point between stages, where the stage is locked by another execution. The superseded execution stops and does not continue.

- QUEUED mode : In this mode, executions are processed one by one in the order that they are queued. Executions wait for the stage to be unlocked before entering. This mode ensures that every execution is processed in sequence and no execution is skipped or superseded.

- PARALLEL mode : In this mode, executions run simultaneously and independently of one another. Executions do not wait for other runs to complete before starting or finishing. This mode allows for faster and more frequent deployments, but may require more resources and coordination.

The flow of pipeline executions can also be controlled by transitions, approval actions, and failures. A transition is the point where a pipeline execution moves to the next stage in the pipeline. A transition can be disabled to prevent executions from entering a stage. An approval action is an action that requires manual intervention to continue the pipeline. A failure is an event that causes an action or a stage to not complete successfully, which stops the execution from moving forward.

7. Types of CodePipelines

- V1 type : A pipeline type that has a JSON structure with standard parameters for pipeline, stage, and action levels. It supports action-level variables, but not pipeline-level variables or source revision overrides. It does not allow triggers and filtering based on Git tags, pull requests, branches, or file paths. It also does not support PARALLEL and QUEUED execution modes.

- V2 type : A pipeline type that has the same structure as V1 type, but with additional parameters for release safety and trigger configuration. It supports both action-level and pipeline-level variables, as well as source revision overrides. It allows triggers and filtering based on Git tags, pull requests, branches, or file paths. It also supports PARALLEL and QUEUED execution modes.

Between the two types of pipelines, V1 and V2, that have different features and pricing models, the main differences are :

Structure : Both types have a JSON structure with standard parameters for pipelines, stages, and actions. V2 type also has additional parameters for release safety and trigger configuration.

Use cases : V1 type is suitable for standard deployments that do not need advanced features. V2 type is ideal for complex deployments that need runtime configuration of variables, initiation based on Git events, or parallel execution modes.

Execution modes : V1 type only supports SUPERSEDED mode, where a newer execution can overtake an older one. V2 type supports PARALLEL and QUEUED modes, where executions can run simultaneously or sequentially.

Cost : V1 type charges based on active pipelines per month, regardless of the number of executions. V2 type charges based on the number of executions, regardless of the number of pipelines. The cost depends on the usage and budget of the user.