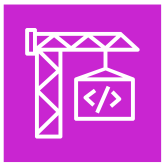


AWS Code***Build***

AWS CodeBuild

- **Purpose:** A fully managed build service that compiles source code, runs tests, and produces software packages that are ready to deploy.
- **Why Use It:** To streamline and automate the build process, ensuring code is always built and tested in a consistent environment.

AWS Code***Pipeline***

AWS CodePipeline

- **Purpose:** 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.
- **Why Use It:** To automate the entire release process, ensuring consistency and speed in delivering new features and updates.

AWS Code***Artifact***

AWS CodeArtifact

- **Purpose:** A fully managed artifact repository service that makes it easy to securely store, publish, and share software packages used in your software development process.
- **Why Use It:** To manage dependencies efficiently and securely across your development teams.

AWS Code***Deploy***

AWS CodeDeploy

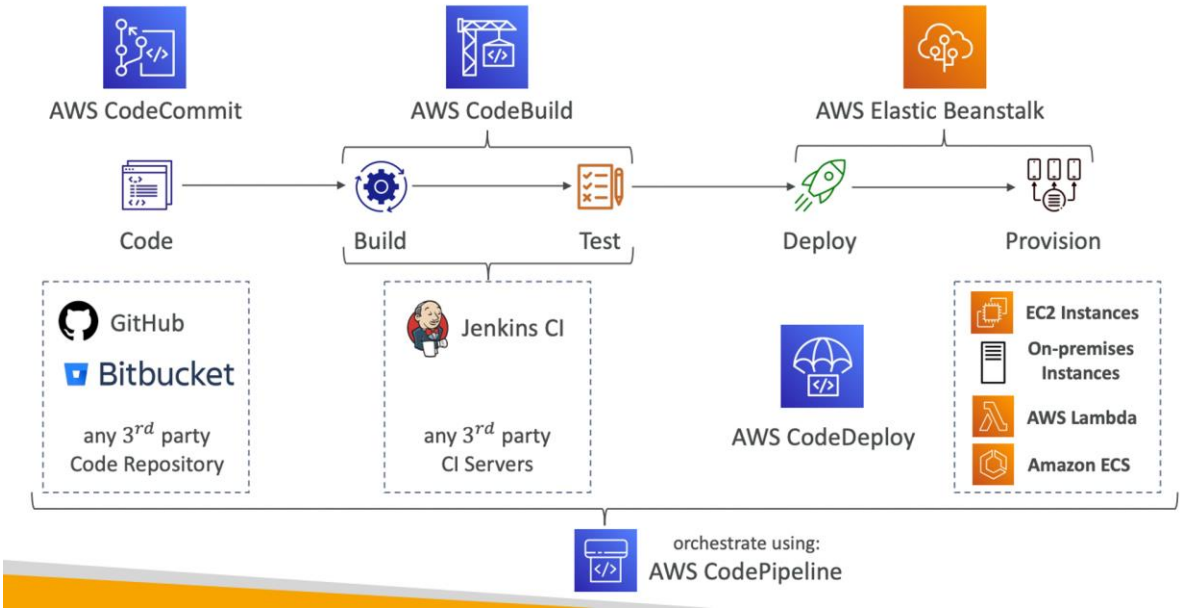
- **Purpose:** Automates code deployments to any instance, including Amazon EC2 instances and instances running on-premises.
- **Why Use It:** To deploy applications consistently and avoid downtime during the deployment process.
- manage code in a secure, scalable, and fully managed environment.

AWS CodeStar

AWS CodeStar

- **Purpose:** Provides a unified user interface to manage software development activities in one place, including CodeCommit, CodePipeline, CodeBuild, and CodeDeploy.
- **Why Use It:** To simplify the setup and management of the entire CI/CD workflow, allowing teams to start developing applications quickly.

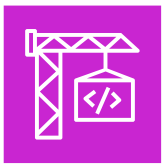
Overall Mapping of the Services for our understanding



- *AWS CodeCommit Service officially closed from the July last week of 2024*

Now you saw the basic components of CI CD Services of AWS – Now we will look into each service & understand a bit more …!!

Let's GOOOO ….!!!!



AWS Code***Build***

Intro of AWS CodeBuild

- **Fully Managed Service**: CodeBuild is a fully managed build service that compiles your source code, runs tests, and produces deployable software packages.
- **Scalability**: It scales continuously and processes multiple builds concurrently.

Build Environment

- **Compute Types and Settings**:
  - CodeBuild provides different compute types (e.g., general1.small, general1.medium, etc.) to choose the appropriate level of resources.
  - **Environment Variables**: You can specify environment variables to be used during the build.

Buildspec File (buildspec.yml)

- **Purpose**: This YAML file defines the build commands and settings. It includes the following sections:
  - **Phases**: Defines the sequence of commands to execute.
    - **install**: Install dependencies and tools.
    - **pre_build**: Pre-build commands, like setting up the environment.
    - **build**: The actual build commands.
    - **post_build**: Commands to run after the build, such as packaging or uploading artifacts.
  - **Artifacts**: Specifies the files and directories to store as build output.
  - **Reports**: Defines test report configuration, including formats and locations.
  - **Cache**: Specifies paths to cache dependencies and outputs to speed up subsequent builds.

Integration with Other AWS Services

- **AWS CodePipeline**: CodeBuild can be a stage in a CodePipeline to automate the build process as part of a CI/CD pipeline.
- **Source Repositories**: CodeBuild integrates with AWS CodeCommit, GitHub, Bitbucket, and Amazon S3 to fetch the source code.
- **Artifact Storage**: Build artifacts can be stored in Amazon S3.
- **Monitoring**: CodeBuild sends logs to Amazon CloudWatch Logs and metrics to Amazon CloudWatch Metrics.
- **Security**: Use AWS Identity and Access Management (IAM) roles and policies to manage permissions for accessing resources.

Security

- **IAM Roles and Policies**: Define permissions using IAM roles and policies to control access to resources during the build process.
- **Environment Variables**: Store sensitive information securely using environment variables.

Scaling and Performance

- **Concurrent Builds**: CodeBuild can run multiple builds concurrently based on account limits.
- **Build Duration Optimization**: Use caching to reduce build times, optimize the buildspec file, and choose appropriate compute types.

Monitoring and Logging

- **CloudWatch Logs**: Logs from the build process are sent to CloudWatch Logs for analysis and troubleshooting.
- **CloudWatch Metrics**: Monitor build performance metrics such as build duration, number of builds, and failure rates.
- **Alarms**: Set CloudWatch alarms to notify you of build failures or performance issues.

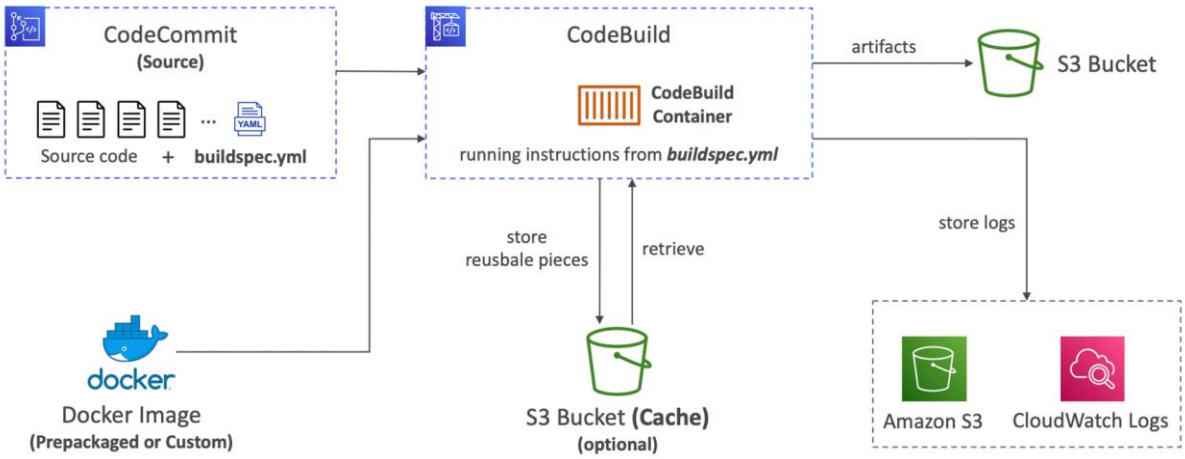
At a glance

- **AWS managed CI server for building and testing code**
- Build instructions are provided in **buildspec. yaml present in the root of the repo**
- Build logs can be stored in S3 or CloudWatch
- CW Metrics can be used to monitor build statistics and CW Events to detect failed builds
- Build projects can be defined in CodeBuild or CodePipeline
- Supported environment
    - Java
    - Ruby
    - Python
    - Go
    - Node.js
    - Android
    - .NET Core
    - PHP
    - Docker - to extend further
- **CodeBuild can be run locally** (requires Docker) using **CodeBuild Agent** for troubleshooting purposes.
- **By default, your CodeBuild containers are launched outside the VPC.** It cannot access resources within the VPC. When creating a CodeBuild project, we can specify the VPC where it should be launched. This way, the CodeBuild project can access resources within the VPC.
- CodeBuild creates a Docker container as the building or testing environment.
- CodeBuild projects can be added as stages in CodePipeline
- CodeBuild containers are deleted at the end of their execution (success or failure). You can't SSH into them, even while they're running.
- **CodeBuild scales automatically** to meet peak build requests. CodeBuild eliminates the need to provision, manage, and scale your own build servers.

CodePipeline and CodeBuild can be combined to build a single page application from MD files and copy the static files to an S3 bucket. This is because CodeBuild can run any commands, so you can use it to run commands including build a static website and copy your static web files to an S3 bucket; and CodePipeline will help make it into a pipeline.

Build Flow

Based on the environment (language), CodeBuild pulls an **AWS pre-packaged Docker image** or a custom image provided by us to build the source code into an artifact based on the instructions provided in buildspec.yaml. **CodeBuild can cache some files (eg. dependencies) in an S3 bucket to share between builds in order to speed up future builds.** The artifact created by the build stage is extracted out of the container and stored in the S3 bucket.



Reference Spec File

```yaml
version: 0.2

env:
  variables:
    JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"
  parameter-store:
    LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword

phases:
  install:
    commands:
      - echo "Entered the install phase..."
      - apt-get update -y
      - apt-get install -y maven
  pre_build:
    commands:
      - echo "Entered the pre_build phase..."
      - docker login -u User -p $LOGIN_PASSWORD
  build:
    commands:
      - echo "Entered the build phase..."
      - echo "Build started on `date`"
      - mvn install
  post_build:
    commands:
      - echo "Entered the post_build phase..."
      - echo "Build completed on `date`"

artifacts:
  files:
    - target/messageUtil-1.0.jar

cache:
  paths:
    - "/root/.m2/**/*"
```

From the above picture - buildspec.yaml

- Must be present at the root of repo
- env - environment variables
    - plain-text variables defined in the yaml file
    - parameter store
    - secrets manager
- phases - specify commands to run:
    - install - install dependencies needed for building
    - pre_build - commands to execute before build
    - build - build commands
    - post_build - post build commands (eg. push Docker image to repo, zip build files)
- artifacts - which file to upload to S3 as artifact (encrypted with KMS)
- cache - files to cache (usually dependencies) to S3 to speed up future builds

AWS Code***Pipeline***

Key Terms

1. **Pipeline Structure**:
   - **Stages**: Each pipeline consists of stages, and each stage can have one or more actions. Common stages include Source, Build, Test, and Deploy.
   - **Actions**: Individual tasks performed within a stage. Types of actions include Source, Build, Test, Deploy, Approval, and Invoke.
2. **Source Action**:
   - **Purpose**: Fetches the source code from a repository.
   - **Supported Providers**: AWS CodeCommit, GitHub, Bitbucket, Amazon S3.
3. **Build Action**:
   - **Purpose**: Compiles the source code, runs tests, and creates build artifacts.
   - **Common Services Used**: AWS CodeBuild, Jenkins.
4. **Test Action**:
   - **Purpose**: Runs automated tests on the build artifacts to ensure quality.
   - **Common Services Used**: AWS CodeBuild, third-party testing tools.
5. **Deploy Action**:
   - **Purpose**: Deploys the build artifacts to various environments.
   - **Common Deployment Targets**: AWS Elastic Beanstalk, AWS CodeDeploy, Amazon ECS, AWS Lambda, S3 (for static websites).
6. **Approval Action**:
   - **Purpose**: Pauses the pipeline execution until a manual approval is granted.
   - **Use Case**: Ensures that a human review is done before deploying to critical environments like production.
7. **Invoke Action**:
   - **Purpose**: Invokes an AWS Lambda function.
   - **Use Case**: Custom processing, notifications, or integration with other systems.

Integration with Other AWS Services

- **AWS IAM**: Define roles and policies to control access and permissions for CodePipeline and its integrated services.
- **Amazon CloudWatch**: Monitor pipeline metrics and set up alarms for pipeline execution statuses.
- **Amazon SNS**: Set up notifications for pipeline state changes and approvals.
- **AWS CodeBuild**: Automate the build process within a pipeline.
- **AWS CodeDeploy**: Automate the deployment process within a pipeline.
- **AWS CodeCommit**: Source repository for version control.

Pipeline Configuration

- **Pipeline Definition**: Pipelines can be defined using the AWS Management Console, AWS CLI, AWS SDKs, or AWS CloudFormation.
- **Artifacts**: Intermediate and output artifacts are stored and passed between actions in different stages.
- **Pipeline Triggers**: Pipelines can be triggered automatically by changes in the source repository or manually.

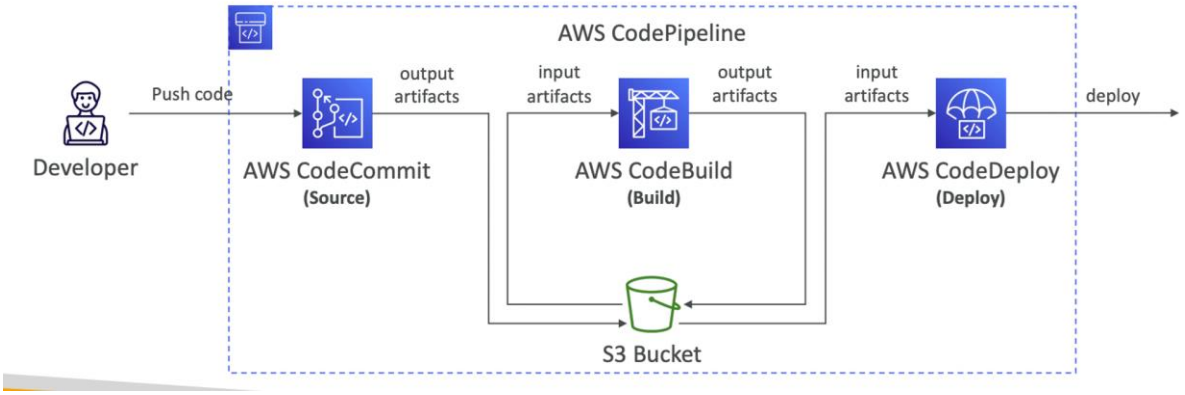
Security

- **IAM Roles and Policies**: Define and assign roles to control what CodePipeline and its actions can access and perform.
- **Encryption**: Ensure artifacts are encrypted at rest and in transit.

Monitoring and Logging

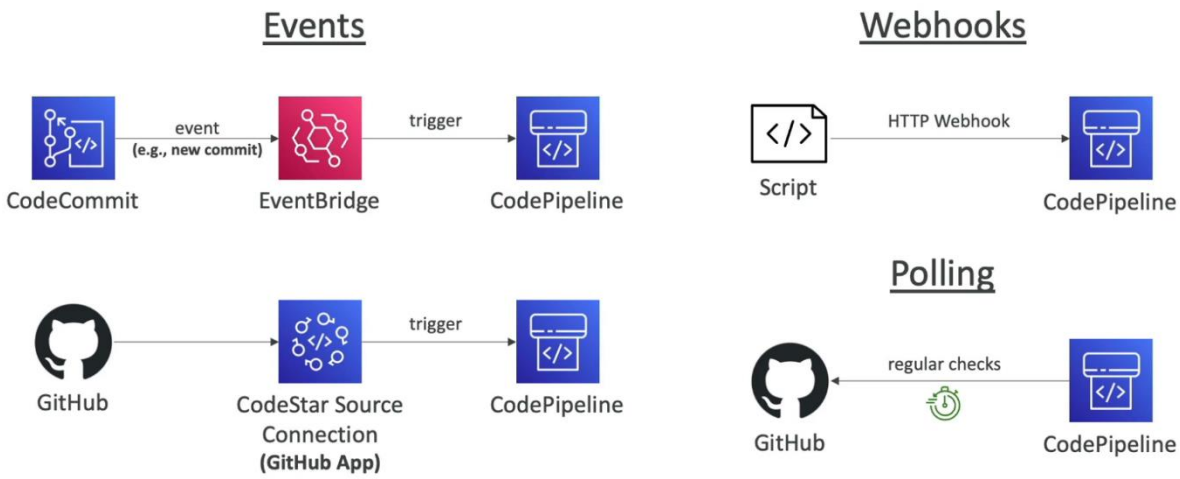
- **CloudWatch Logs**: Each action in the pipeline can send logs to CloudWatch Logs.
- **CloudWatch Metrics**: Monitor metrics such as pipeline execution time, success/failure counts, and more.
- **Alarms and Notifications**: Set up CloudWatch alarms and SNS notifications for pipeline events

- **Visual Workflow to orchestrate CICD pipelines.** Pipeline can be edited when needed to add or remove stages.
- **Each stage in the pipeline creates an artifact which is stored in S3 (Artifact Store).** The next stage uses this artifact as input.



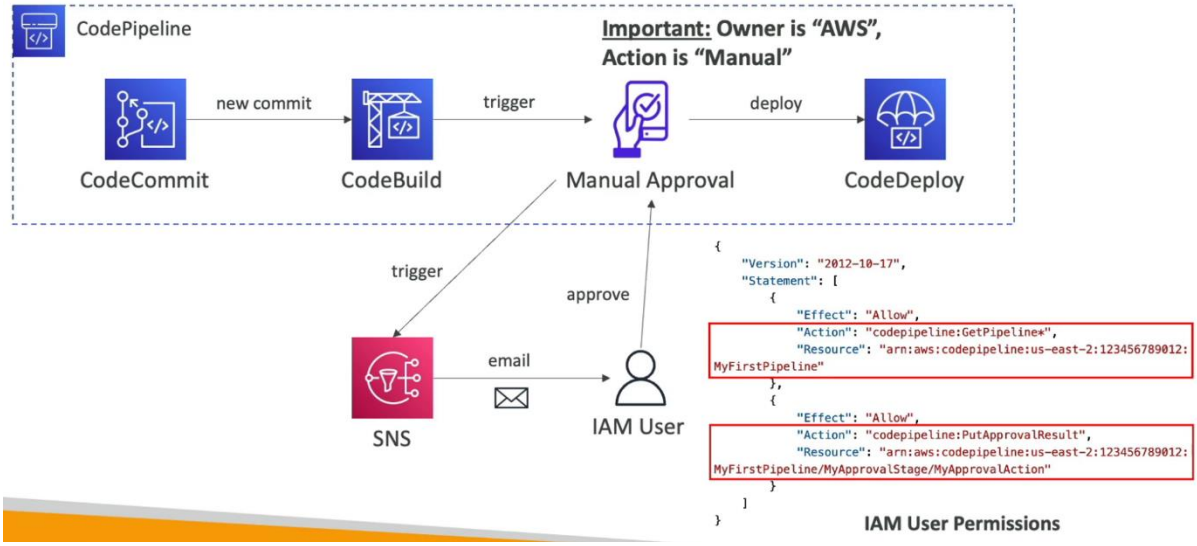
Events vs Webhooks vs Polling

- **Events**
  - **Default and recommended way**
  - Code change in CodeCommit generates an EB event which triggers CodePipeline.
  - For a version control application outside of AWS (eg. GitHub), need to use **CodeStar Source Connection** to trigger the CodePipeline on event from GitHub.
- **Webhooks**
  - Older method
  - CodePipeline provides a webhook URL which can be used to trigger CodePipeline with a payload
- **Polling**
  - CodePipeline regularly polls the version control application for changes (inefficient)

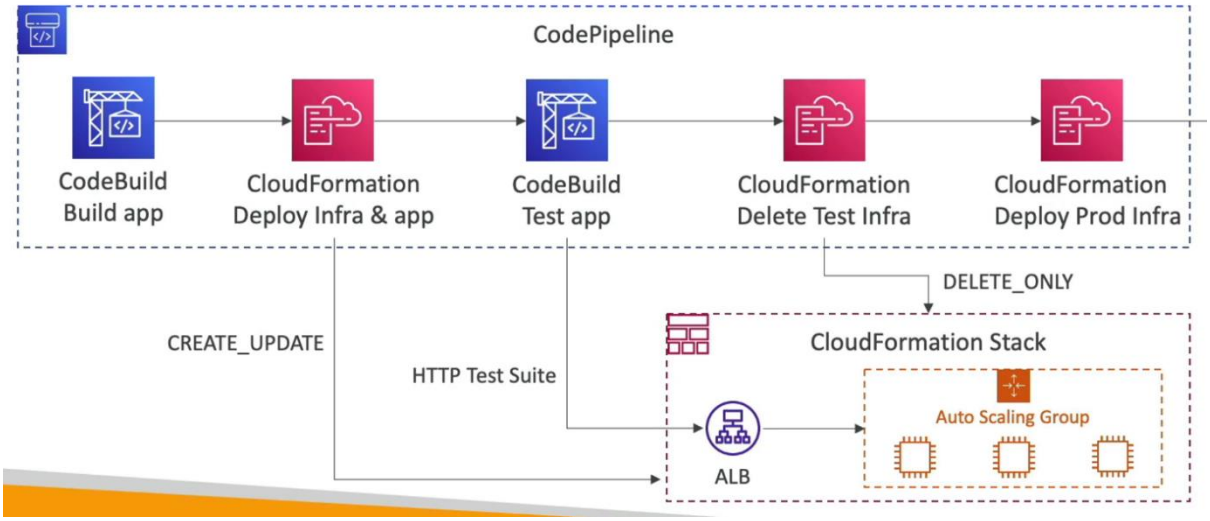


Manual Approval

- Manual approval can be defined at any stage of the pipeline
- **For manual approval, the owner must be AWS**
- We can setup an SNS topic to send an email to the user for manual approval.
- The user must have codepipeline:GetPipeline* permission to view the pipeline and codepipeline:PutApprovalResult permission to approve the pipeline.
-



CloudFormation Integration



CloudFormation can be integrated with CodePipeline to create a test stack and delete it after the tests have been run. If all the tests pass, another CloudFormation step can deploy the app on production.

AWS Code*Artifact*

Key Terms

1. **Domain**
   - **Definition**: A domain is a higher-level entity that contains one or more repositories.
   - **Purpose**: Domains allow organizational level management of repositories, enabling centralized access and control policies.
   - **Ownership**: Domains are owned by an AWS account, and each account can own multiple domains.
2. **Repository**
   - **Definition**: A repository is a collection of software packages.
   - **Purpose**: Repositories store and organize packages, and they can be linked to upstream repositories for pulling dependencies.
   - **Types**: Can be public or private.
3. **Package**
   - **Definition**: A package is a versioned collection of artifacts that include code and other dependencies.
   - **Package Versions**: Packages can have multiple versions stored within a repository.
4. **Upstream Repositories**
   - **Definition**: Repositories configured to pull packages from other repositories.
   - **Purpose**: To allow a repository to access packages from other repositories or external sources (e.g., npm registry, Maven Central).
5. **Package Version Lifecycle Policies**
   - **Definition**: Rules that automate the cleanup of old or unwanted package versions.
   - **Purpose**: To manage storage costs and keep the repository clean by removing obsolete versions.

6. **Domain and Repository Permissions**
    - **IAM Policies**: Use AWS Identity and Access Management (IAM) to control who can access and manage domains and repositories.
    - **Resource Policies**: Define fine-grained access controls for specific domains and repositories.

Integration with Other AWS Services

- **AWS CodeBuild**: Fetch dependencies from CodeArtifact during the build process.
- **AWS CodePipeline**: Use CodeArtifact to store build artifacts and dependencies.
- **AWS CloudWatch**: Monitor usage metrics and set up alarms.
- **AWS CloudTrail**: Track API calls and changes to CodeArtifact resources for auditing.

Security

- **Encryption**: CodeArtifact encrypts packages at rest using AWS Key Management Service (KMS).
- **IAM Roles and Policies**: Define who can publish, retrieve, and manage packages in CodeArtifact.
- **Cross-Account Access**: Set up cross-account access to share packages between different AWS accounts.

Best Practices

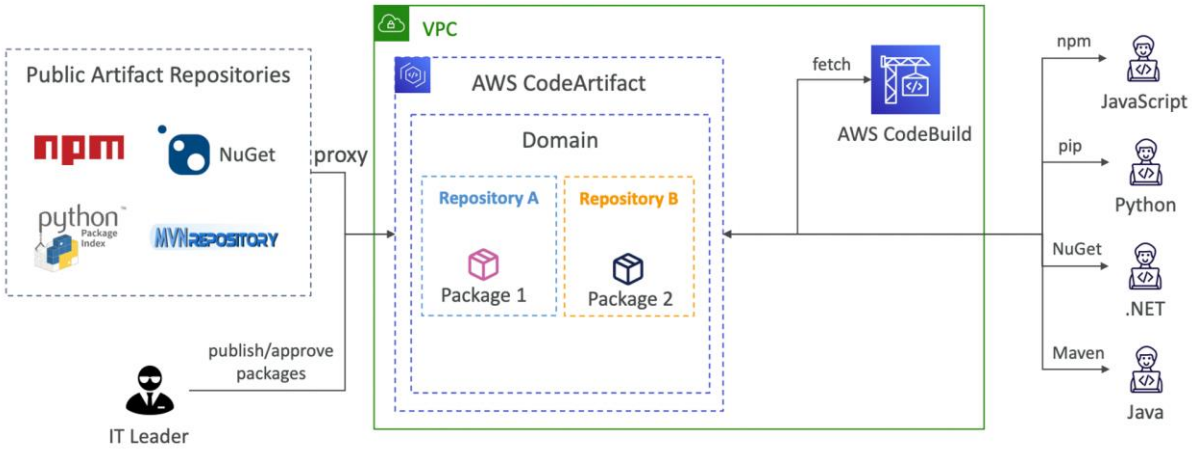
- **Organize Repositories**: Group related packages into repositories for better organization.
- **Use Domains**: Utilize domains to manage multiple repositories with shared settings and access controls.
- **Lifecycle Policies**: Implement package version lifecycle policies to automatically clean up old versions and manage storage costs.
- **Secure Access**: Use IAM policies and resource policies to enforce least privilege access to repositories and domains.

Monitoring and Logging

- **CloudWatch Metrics**: Monitor repository metrics such as the number of requests, data transfer, and storage usage.
- **CloudTrail Logs**: Track actions performed on CodeArtifact resources for security auditing and compliance.

Workflow

- CodeArtifact proxies requests to public artifact repositories and caches publicly available artifacts. So, even if the artifact is deleted from the public repository, it will still be available in CodeArtifact.
- Custom artifacts can also be hosted on CodeArtifact
- Developers and CodeBuild fetch all the artifacts from a single source



- CodeArtifact proxies requests to public artifact repositories and caches publicly available artifacts. So, even if the artifact is deleted from the public repository, it will still be available in CodeArtifact.
- Custom artifacts can also be hosted on CodeArtifact
- Developers and CodeBuild fetch all the artifacts from a single source

Upstream Repositories

Upstream repositories allow you to configure your repositories to pull packages from external sources. This enables your internal CodeArtifact repository to automatically retrieve and cache packages from external repositories, such as public package registries or other CodeArtifact repositories.

An upstream repository is an external package repository that your CodeArtifact repository is configured to pull packages from when they are not found locally.

- A CodeArtifact repository can have other CodeArtifact repositories as Upstream Repositories. This allows the developer to access the packages contained in multiple repositories from a single repository endpoint.
- A repository can have **max 10 upstream repositories**
- A repository can only have one external connection with a public repository. Artifacts fetched from public repositories will be cached (stored in the domain and have reference to the artifact) at the CodeArtifact repository connected to it.

Working Method:

Step 1: Client Request:

- A client (e.g., npm client, pip, or Maven) requests a package from your CodeArtifact repository.

Step 2: CodeArtifact Repository:

- CodeArtifact checks its own storage to see if the requested package is available.

Step 3: Upstream Check:

- If the package isn't found, CodeArtifact internally makes a request to the configured upstream public or external repository.

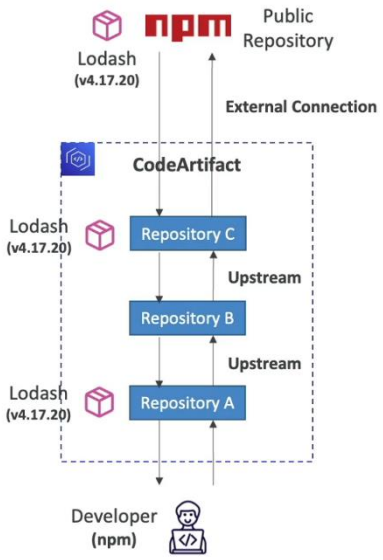
Step 4: External Repository Response:

- The external/public repository (like npmjs.com) responds with the package data.

Step 5: Caching and Response:

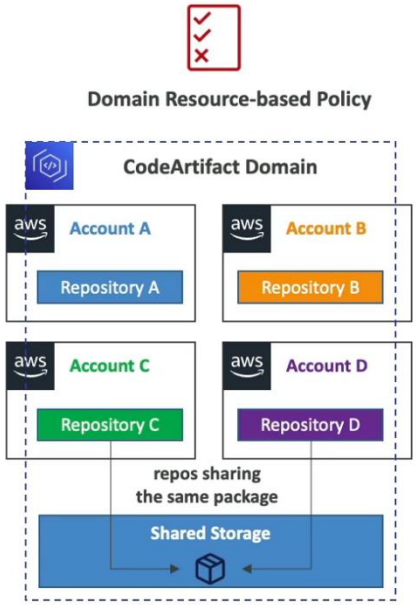
- CodeArtifact caches the package and immediately serves it to the client.

Retention



- If a requested package (artifact) is found in an upstream repo, a reference to it is stored in the downstream repo closest to the developer (who requested the package).
- If the package is fetched from a public repo, a reference to it will be stored in the repo connected to the external repo.
- Intermediate repos that were involved in resolving the package do not store reference to the package.
- The package cached in the downstream repo is not affected by changes to the package in the upstream repository.

Domains



- Single shared storage for all the repositories across multiple accounts.
- **De-duplicated storage**: Artifacts are stored only once in the shared storage of the domain and repositories only store references to these artifacts. The artifacts are not duplicated across repositories.
- **Fast copying**: when a repository pulls an artifact from an upstream repository, it only copies the metadata (containing the reference to the artifact stored in the domain).
- **Easy sharing across repositories and teams**: all the assets and the metadata in a domain are encrypted with a single KMS key

→ Domain Resource-based Policy: domain administrator can apply policy across the domain such as:

- Restricting which accounts have access to repositories in the domain

- Who can configure connections to public repositories to use as sources of packages

AWS Code***Deploy***

Key Terms

1. **Deployment Types**
    - **In-Place Deployment**: Updates the application on the existing instances without replacing them. This deployment type temporarily stops the application.
    - **Blue/Green Deployment**: Deploys the new version alongside the old version, allowing for testing before switching traffic to the new version. This approach minimizes downtime.
2. **Application and Deployment Groups**
    - **Application**: A logical entity that represents the CodeDeploy application.
    - **Deployment Group**: A set of individual instances or Lambda functions targeted for deployment. It includes configuration settings like deployment type, deployment configuration, and optional triggers and alarms.
3. **Deployment Configurations**
    - **Predefined Configurations**:
        - CodeDeployDefault.OneAtATime
        - CodeDeployDefault.HalfAtATime
        - CodeDeployDefault.AllAtOnce
    - **Custom Configurations**: Define the minimum number of healthy instances and other parameters.
4. **AppSpec File**
    - **Definition**: A YAML or JSON file that specifies how CodeDeploy deploys the application. It includes:
        - **Version**: Version of the AppSpec file.
        - **Resources**: The files and scripts to be deployed.
        - **Hooks**: Lifecycle event hooks that specify deployment actions at different stages (e.g., BeforeInstall, AfterInstall, ApplicationStart, ValidateService).

5. **Lifecycle Event Hooks**
    - **Hooks:** Scripts or commands executed at specific points in the deployment lifecycle.
    - **Common Hooks:**
        - BeforeInstall
        - AfterInstall
        - ApplicationStart
        - ApplicationStop
        - ValidateService
6. **Rollback and Redeployments**
    - **Automatic Rollbacks:** Configured to automatically revert to the previous version if deployment fails.
    - **Manual Rollbacks:** Triggered by the user.

Integration with Other AWS Services

- **AWS CodePipeline:** Use CodeDeploy as a deployment stage in a CI/CD pipeline.
- **Amazon CloudWatch:** Monitor deployment metrics and set alarms.
- **AWS Lambda:** Automate function updates.
- **Amazon EC2:** Deploy applications to EC2 instances.

Security

- **IAM Roles and Policies:** Define roles for CodeDeploy to access other AWS resources securely.
- **Encryption:** Encrypt data at rest and in transit.

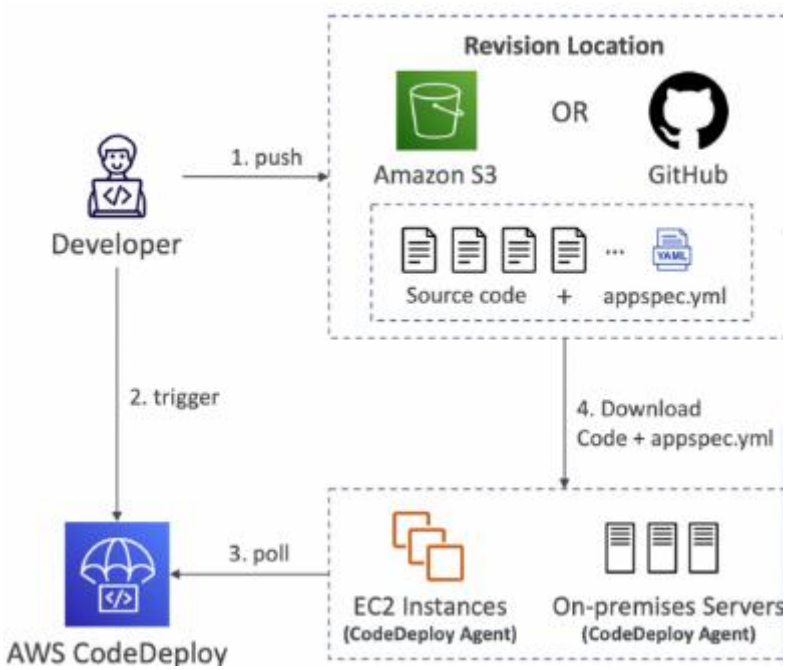
Monitoring and Logging

- **CloudWatch Metrics:** Monitor deployment status, success/failure rates, and other metrics.
- **CloudWatch Alarms:** Set up alarms for deployment failures.
- **Logging:** Access deployment logs for troubleshooting.

Components

- Application - the build artifact (archived) we want to deploy
- **Compute Platform** - what platform the application will be deployed to
    - EC2 or On-Premises (**CodeDeploy Agent** must be installed)
    - AWS Lambda
    - Amazon ECS
- Deployment Configuration - set of deployment rules for successful deployment
    - EC2/On-premises - specify the minimum number of healthy instances for the deployment
    - AWS Lambda or Amazon ECS - specify how traffic is routed to your updated versions
- **Deployment Group** - group of tagged EC2 instances (allows to deploy gradually, ex: first deploy to dev, then test and then prod)
- **IAM Instance Profile** - give EC2 instances the permissions to access S3 or GitHub
- **Application Revision** = build artifact + appspec.yaml file
- **Service Role** - IAM Role for CodeDeploy to perform operations on EC2 instances, ASGs, ELBs for deployment
- Target Revision - the most recent revision that you want to deploy to a Deployment Group

Workflow



- The developer or CI server builds and pushes the application revision (build artifact + appspec.yaml) on S3 or GitHub.
- The developer triggers a new deployment in CodeDeploy.
- The CodeDeploy Agent running on the compute resource continuously polls CodeDeploy to check if a new deployment needs to be done.

The build artifact and appspec.yaml are downloaded to the compute resource and the CodeDeploy Agent runs the deployment instructions mentioned in appspec.yaml.

appspec.yaml

```yaml
version: 0.0
os: linux

files:
  - source: Config/config.txt
    destination: /webapps/Config
  - source: source
    destination: /webapps/myApp

hooks:
  BeforeInstall:
    - location: Scripts/UnzipResourceBundle.zip
    - location: Scripts/UnzipDataBundle.zip
  AfterInstall:
    - location: Scripts/RunResourceTests.sh
      timeout: 180
  ApplicationStart:
    - location: Scripts/RunFunctionalTests.sh
      timeout: 3600
  ValidateService:
    - location: Scripts/MonitorService.sh
      timeout: 3600
      runas: codedeployuser
```

- Specifies how the build artifact should be deployed.
- Should be present at the root of the bundle.
- files - what to copy from S3 or GitHub into the deployment server
- hooks - set of instructions to deploy the new application version:
    - **ValidateService** (checks if the application is running properly after installation)
- Can have timeouts in any step (hook)
- hooks section for EC2 instances is shown to the right.

- Order of hooks for different compute resources

  - Lambda: BeforeAllowTraffic > AfterAllowTraffic

  - EC2: BeforeInstall → AfterInstall → ApplicationStart → ValidateService

  - ECS: BeforeInstall → AfterInstall → AfterAllowTestTraffic → BeforeAllowTraffic → AfterAllowTraffic

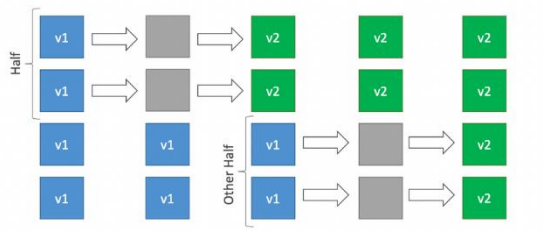
- The content in the `resources` section varies, depending on the compute platform of your deployment. For Lambda functions, it has `name`, `alias`, `currentversion` and `targetversion`.

```
resources:
  - name-of-function-to-deploy:
      type: "AWS::Lambda::Function"
      properties:
        name: name-of-lambda-function-to-deploy
        alias: alias-of-lambda-function-to-deploy
        currentversion: version-of-the-lambda-function-traffic-currently-points-to
        targetversion: version-of-the-lambda-function-to-shift-traffic-to
```

Deployment Type

It is the method used to deploy the application to a Deployment Group.

- **In-place Deployment** - supports EC2, On-Premise Server
  - **One at a time - one EC2 instance at time, if one instance fails then deployment stops**
  - **Half at a time - 50% healthy instances even if the deployment fails**
  - **All at once - quick but has downtime (good for dev)**
  - **Custom - specify the minimum percentage of healthy hosts**
- **Blue/Green Deployment** - supports EC2 instances, AWS Lambda, and Amazon ECS



Lambda functions don't support in-place deployment

Failures & Rollbacks

- Once deployment fails, EC2 instances stay in failed state
- New deployments will first be deployed to failed instances
- To rollback, redeploy old deployment or enable automated rollback on failures
- If a rollback happens, CodeDeploy redeploys the last known good revision with a new deployment ID (does not restore to a previous version).

Troubleshooting

- InvalidSignatureException - the date or time in CodeDeploy (AWS) does not match that in the EC2 instance where the application is to be deployed
- If the deployment fails on EC2, check deployment log files at /opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log

AWS CodeStar

- An abstraction over CICD services in AWS
- Used to **quickly create CICD-ready projects** for **EC2, Lambda and Elastic Beanstalk**
- One dashboard to view all the components
- Supported languages: C#, Go, HTML, Java, NodeJS, PHP, Python and Ruby
- Issue tracking integration with Jira and GitHub Issues
- Free service, pay for the underlying resources
- Limited customization
- Supports CodeCommit and GitHub as code repo
- Integrates with **Cloud9** (cloud-based IDE)

Project Templates

- **Definition**: Pre-configured project templates that help you start development quickly.
- **Types**: Templates for different programming languages (Java, Python, Node.js, etc.) and frameworks (Spring, Express, etc.).
- **Usage**: Select a template based on your application type, and CodeStar sets up the necessary resources and CI/CD pipeline.

Integrated Services

- **CodeCommit**: Source code repository.
- **CodeBuild**: Build service.
- **CodeDeploy**: Deployment service.
- **CodePipeline**: CI/CD pipeline.
- **CloudWatch**: Monitoring and logging.
- **CloudFormation**: Infrastructure as code (IaC) for setting up resources.

Project Dashboard

- **Purpose**: Provides an overview of the project's status, including recent commits, build results, deployment status, and application metrics.
- **Features**: Access to project resources, pipeline status, and team collaboration tools.

There is one more service we need keep it in mind is AWS CodeGuru (Belongs to AWS AI Category but helps us in AWS CI & CD Services)

AWS CodeGuru

AWS Guru, also known as **Amazon DevOps Guru**, is a service designed to help you improve the availability and operational performance of your applications by using machine learning (ML) to identify operational issues, anomalies, and potential risks before they affect your applications. It is part of Amazon's broader suite of tools aimed at making DevOps practices more efficient and automated.

AWS CI & CD Scenario based Questions

https://lisireddy.medium.com/aws-ci-cd-services-scenario-based-questions-539c9b13f234

Wish you the best …! Happy Learning ..! Yours' Love

(@lisireddy across all the platforms)