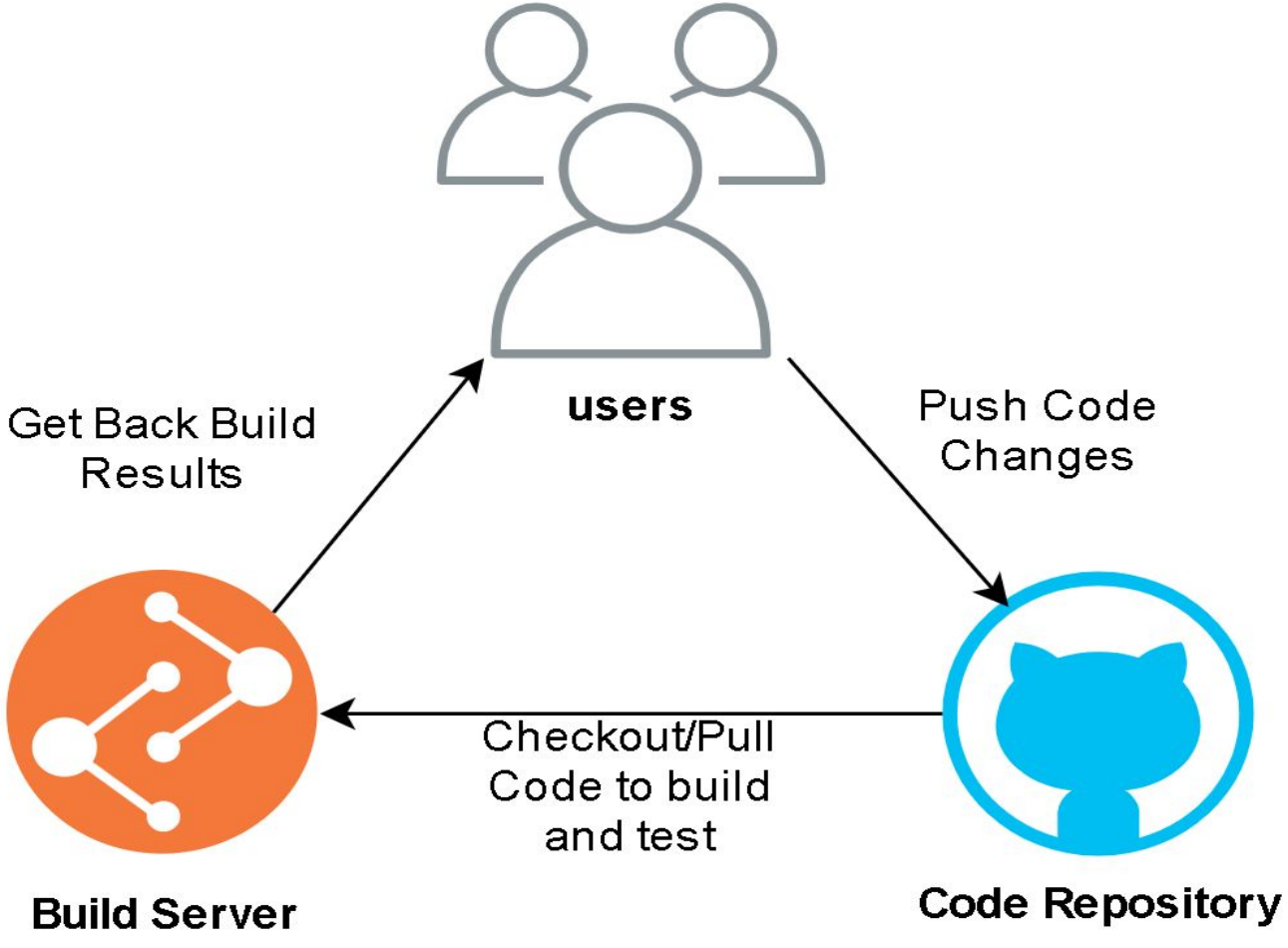
Continuous Integration



Code Repo:
Github/CodeCommit

What is AWS CodeBuild?

- AWS CodeBuild is a **fully managed build service**.
- **CodeBuild provides a compute environment in a Docker Container (Using a Docker Image stored in ECR/any registry)**
- It is a fully managed **continuous integration service** that **compiles source code, runs tests, and produces software packages (Artifacts)** that are ready to deploy.
- Execute instructions for **Infrastructure creation/update/delete** or any **aws cli commands**.
- **You cannot login inside the CodeBuild Docker Container.**
- CodeBuild Compute will **run commands** in **build specification file** (**buildspec.yml**).
- CodeBuild will run a **container** and once execution status is **Succeeded/Failed it means the compute environment will destroy itself at every execution**.

Concepts

- A **build project** defines how CodeBuild will run a build. It includes information such as **where to get the source code from**, which **build environment** to use, the **build commands** to run (**buildspec.yml**) , and where to **store the build output** (**S3**)
- CodeBuild Projects are **Region Specific**.
  - **Every build project will be having one or more build runs.**
- A **build environment** is the combination of ***operating system, programming language runtime***, *and* ***tools*** used by CodeBuild to run a build.
- The **build specification file (required)** is a **YAML file** that lets you choose the commands to run at each **phase** of the build and other settings.
  - Without a build spec, CodeBuild cannot successfully convert your build input into build output or locate the build output artifact in the build environment to upload to your output bucket.
  - If you include a build spec as part of the **source code**, by default, the build spec file must be named **buildspec.yml** and placed in the root of your source directory.
- A collection of input files is called **build input artifacts** or **build input** and a deployable version of a source code is called **build output artifact or build output.**

Concepts

Build environment compute types

You can choose from three levels of compute capacity that vary by the amount of CPU and memory to best suit to your development needs

Environment computeType	Memory	vCPUs	Disk Space (GB)
BUILD_GENERAL1_SMALL ***Free Tier : 100 build minutes** of **build.general1.small** per month*	3 GB	2	64
BUILD_GENERAL1_MEDIUM	7 GB	4	128
BUILD_GENERAL1_LARGE	15 GB	8	128
BUILD_GENERAL1_2XLARGE	145 GB	72	824 (SSD)

YAML Basics

- YAML is a **data serialisation language** designed to be directly writable and readable by humans.
- It's a strict superset of JSON, with the addition of **syntactically significant newlines and indentation**, like **Python**. **No curly Braces in YAML**.
- YAML is case sensitive.
- YAML does not allow the use of tabs while creating YAML files; **spaces are allowed instead**.
- **Conventional Block Format : Uses hyphen+space** to begin a **new item** in a specified **list**
- **Inline Format for List :** It is delimited with comma and space
- **Folded Text :** Folded text converts newlines to spaces and removes the leading whitespace
- **Extension:** YAML file should end with extensions like **.yaml** or **.yml**

YAML Basics

- **Comments & Key Value Pairs**
  - Colon and value should have a space (https://json2yaml.com/)

```
# Defining simple key value pairs
cloud: aws
compute: ec2
storage: s3
```

- Dictionary / Map
  - Set of properties grouped together after an item
  - Equal amount of blank space required for all the items under a dictionary

```
aws:
  compute: ec2
  storage: s3
  logging: cloudwatch
```

YAML Basics

```
aws: # Dictionary
  compute: ec2
  storage: s3
  logging1: # List
    - cloudwatch
    - cloudtrail
  logging2: [cloudwatch, cloudtrail]
# List with a different notation
```

- Array / Lists

- Multiple Lists

```
aws: # Dictionary
  compute: ec2
  storage: s3
  logging: # List
    - cloudwatch
    - cloudtrail
  logging: [cloudwatch, cloudtrail]
  linux:
    - name: AmazonLinux2
      provider: aws
    - name: Centos
      provider: Redhat
```

YAML Basics

- The structure which follows all the basic conventions of YAML is shown below -

```
AWS1: [EC2,S3,VPC,IAM]

AWS2:
  - EC2
  - S3
  - VPC
  - IAM
```

Synopsis of YAML Basic Elements

- Comments in YAML begins with the **(#)** character.
- **Indentation of whitespace is used to denote structure.**
- Tabs are not included as indentation for YAML files.
- List items are denoted by a leading hyphen (-).
- List items are enclosed in square brackets and separated by “**,**”
- YAML always requires **colons** and commas used as list separators followed by space with scalar(key value) values.

Steps in a Build Process

1. CodeBuild will create a **temporary compute container** of the class defined in the build project (**Ubuntu**, Amazon Linux)
2. CodeBuild loads it with the specified runtime environment.
3. CodeBuild downloads the source code (Github/Codecommit/S3)
4. CodeBuild executes the commands configured in the project build specification file (**buildspec.yml file**)
5. CodeBuild uploads the generated artifact to an S3 bucket.
6. Then it destroys the compute container.
7. All CodeBuild Execution Logs are store in **CloudWatch Logs**.

Note : Build Duration is calculated in minutes, from the time you submit your build until your build is terminated, rounded up to the nearest minute.

Features

- AWS CodeBuild runs your builds in preconfigured build environments that contain the operating system, programming language runtime, and build tools (such as **Apache Maven, Gradle, npm**) required to complete the task.
- You just specify your source codes location and select settings for your build, such as the build environment to use and the build commands to run during a build.
- AWS CodeBuild builds your code and stores the artifacts into an Amazon S3 bucket, or you can use a build command to upload them to an artifact repository.
- AWS CodeBuild provides build environments for:

  Java, Python, Node.js, Ruby, Go, Android, .NET Core for Linux, Docker

Features

- You can define the specific commands that you want AWS CodeBuild to perform, such as installing build tool packages, running unit tests, and packaging your code. You can choose from three levels of compute capacity that vary by the amount of CPU and memory to best suit to your development needs
- You can integrate CodeBuild into existing CI/CD workflows using its source integrations, build commands, or Jenkins integration.
- CodeBuild can connect to **AWS CodeCommit, S3, GitHub and Bitbucket** to pull source code for builds.
- You can access your past build results through the **console, CloudWatch, or the API.**

Monitoring and Security

- CodeBuild provides security and separation at the infrastructure and execution levels.
- **Amazon CloudWatch** is used to watch your builds, report when something is wrong, and take automatic actions when appropriate.
- You can monitor your builds at two levels:
  - At the project level: These metrics are for all builds in the specified project only.
  - At the AWS account level: These metrics are for all builds in one account
- **ProjectName** is the only AWS CodeBuild metrics dimension. If it is specified, then the metrics are for that project. If it is not specified, then the metrics are for the current AWS account.

AWS CodeBuild Pricing

- You are charged for **compute resources based on the duration it takes for your build to execute**. The per-minute rate depends on the compute type you use.
- For more information click here : **AWS CodeBuild pricing.**

CodeBuild Features

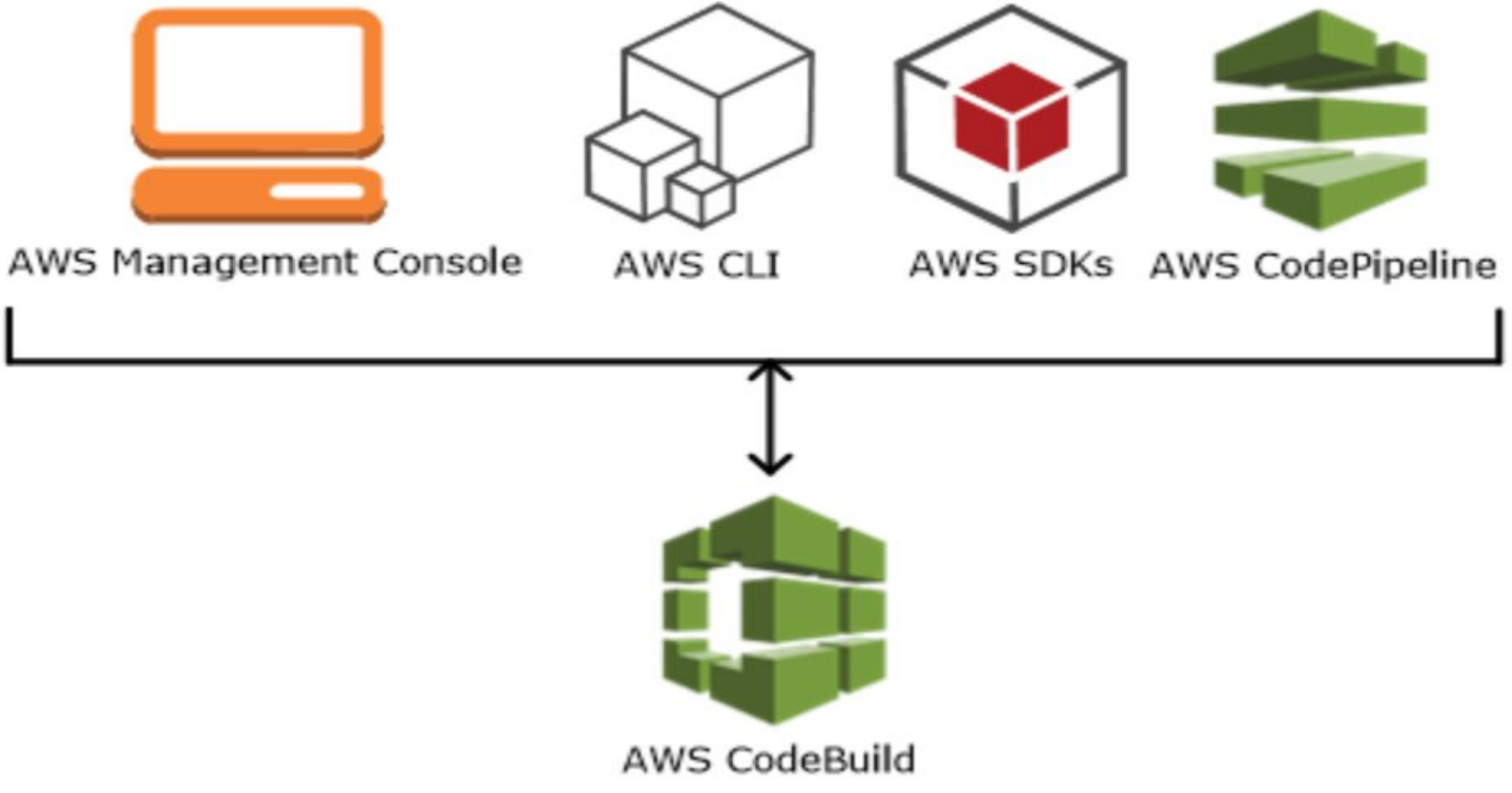
- Compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- Eliminates the need to provision, manage, and scale your own build servers.
- It provides prepackaged build environments for the most popular programming languages and build tools such as Apache Maven, Gradle, and more.
- We can also customize build environments in CodeBuild to use our own build tools.
- Scales automatically to meet peak build requests.

CodeBuild Pricing and Usage

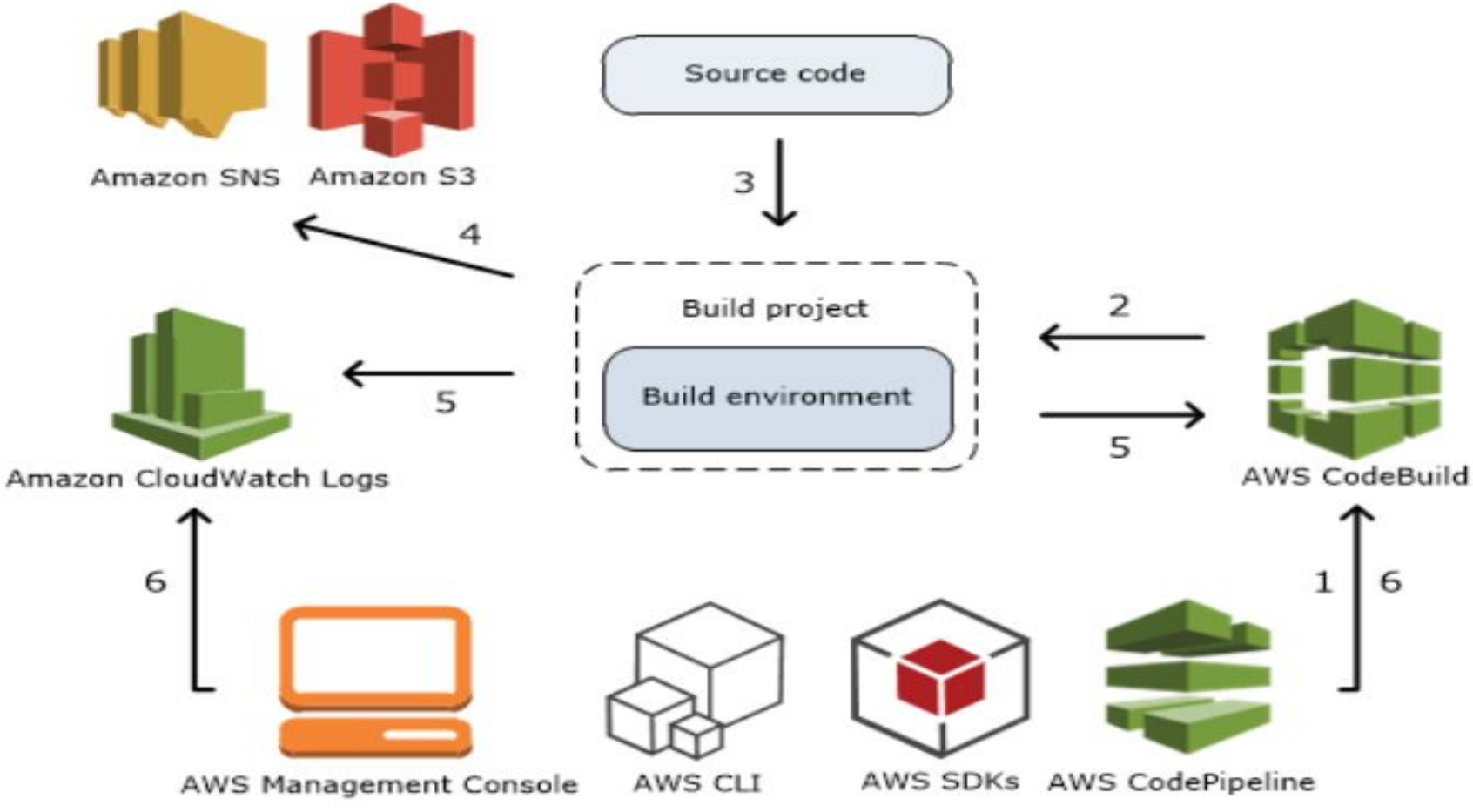
- **Pay for usage**: the time it takes to complete the builds
  - https://aws.amazon.com/codebuild/pricing/
  -
- **Docker** : CodeBuild leverages Docker under the hood for reproducible builds.
- **Secure:** CodeBuild Uses IAM Service as Build Permissions, Integration with KMS for encryption of build artifacts, and VPC for network security, CloudTrail for API calls logging.

- Source Code from **S3/ GitHub / CodeCommit / CodePipeline.**
- Build instructions are defined in code (***buildspec.yml file***)
- CodeBuild Outputs the Execution logs to Amazon S3 & AWS CloudWatch Logs Metrics to monitor CodeBuild statistics.
- Use CloudWatch Alarms to notify if you need “thresholds” for failures
- SNS notifications
- Use CloudWatch Events to detect failed builds and trigger notifications
- Alternative to other build tools such as Jenkins.

How to run CodeBuild



How CodeBuild Works



Build Environment Reference for CodeBuild

- A build environment contains a **Docker image.**
- When you provide information to CodeBuild about the build environment, you specify the identifier of a Docker image in a supported repository type that can be :

1. *CodeBuild Docker image repository*
2. *Publicly available images in Docker Hub*
3. *Amazon Elastic Container Registry (Amazon ECR) repositories that your AWS account has permissions to access*

Official AWS Github CodeBuild repository for managed Docker images.

https://github.com/aws/aws-codebuild-docker-images

Environment Variables in Build Environment

- AWS CodeBuild provides several environment variables that you can use in your build commands:
- Below are few of them:
  - **AWS_REGION**: The AWS Region where the build is running (for example, us-east-1).
  - **CODEBUILD_BUILD_NUMBER:** The current build number for the project. *This number is useful for semantic versioning of Build Versions.*
  - **CODEBUILD_RESOLVED_SOURCE_VERSION:** An identifier for the version of a build's source code. Its format depends on the source code repository:
  - The URL to the input artifact or source code repository.

    For S3, this is s3:// followed by the bucket name and path to the input artifact.

    For CodeCommit and GitHub, this is the repository's clone URL.

    For CodePipeline, then this might be empty.
  - **CODEBUILD_SRC_DIR**: The directory path that CodeBuild uses for the build (for example, **/codebuild/output/src241206719/src/git-codecommit.us-east-1.amazonaws.com/v1/repos/codebuild-cf-stack-test**).

Source Version Sample with CodeBuild

- To list all of the available environment variables in a build environment, you can run the **printenv** command (for Linux Based Images) in **buildspec.yml** file
- To specify a **GitHub/CodeCommit** repository version with a commit ID and reference.
- In Source version, enter **refs/heads/master^{046e8b67481d53bdc86c3f6affdd5d1afae6d369}.**

This is the same commit ID and a reference to a branch in the format **refs/heads/branchname^{full-commit-SHA}.**

Create a Notification Rule

- You can use notification rules to notify users when important changes, such as build **succeeds and failures**, occur. Notification rules specify both the events and the Amazon SNS topic that is used to send notifications.
- On the build project page, choose **Notify**, and then choose **Create notification rule**.
- In **Events that trigger notifications**, select the events for which we want to send notifications.

Category	Events
Build state	Failed Succeeded In-progress Stopped
Build phase	Failure Success