What is AWS CodeBuild?

- AWS CodeBuild is a **fully managed build service**.

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

Concepts

- A **build project** defines how CodeBuild will run a build. It includes information such as where to get the source code, which build environment to use, the build commands to run, and where to store the build output.

- A **build environment** is the combination of *operating system, programming language runtime, and tools* used by CodeBuild to run a build.

- The **build specification** 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.**

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
  - **Build.general1.small** - 3GB memory, 2 vCPU
  - **Build.general1.medium** - 7GB memory, 4 vCPU
  - **Build.general1.large** - 15GB memory, 8 vCPU

- 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 GitHub Enterprise and Bitbucket to pull source code for builds.

- You can access your past build results through the console, CloudWatch, or the API.

Steps in a Build Process

1. CodeBuild will create a temporary compute container of the class defined in the build project.
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.
5. CodeBuild uploads the generated artifact to an S3 bucket.
6. Then it destroys the compute container.

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

Monitoring and Security

- You can specify a key stored in the AWS Key Management Service to encrypt your artifacts.

- CodeBuild provides security and separation at the infrastructure and execution levels.

- You can use Amazon CloudWatch 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

- **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
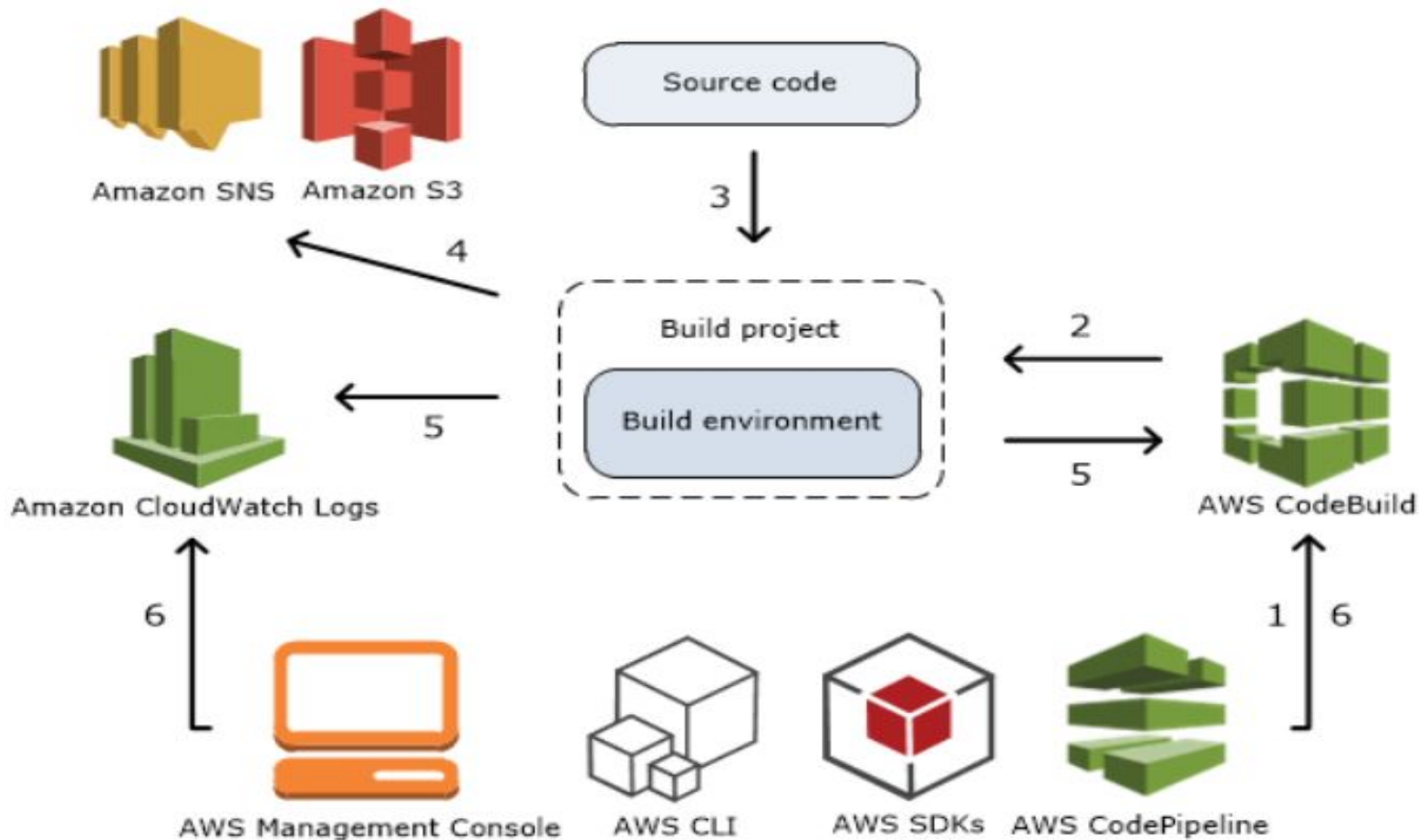
How CodeBuild Works

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:

- <u>Below are few of them:</u>
  - **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. <u>*This number is useful for semantic versioning of Build Versions.*</u>
  - **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, /tmp/src123456789/src).

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 successes 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