Software Delivery Pipeline Manual Book

I. Set up the environment.

- Ensure you have the AWS CDK installed and initialized in your local environment.

  ```
  npm install -g aws-cdk
  ```

- And for the Python environment, make sure you have the necessary CDK libraries.

II. Finish the CDK python file

- Write the CDK Application
  - Define Imports: Import required AWS CDK libraries for managing AWS resources.
- Define AWS Resources
  - **S3 Bucket**: This bucket is used to store the build artifacts

    ```
    artifact_bucket = s3.Bucket(self, "ArtifactBucket")
    ```

  - **CodeCommit Repository:** A Git repository to store the source code of the Java project.

    ```
    code_repo = codecommit.Repository(
      self, "JavaProjectRepo",
      repository_name="java-project",
     code=codecommit.Code.from_asset("finalProject/SoftwareDeliveryPipeline/java-project.zip")
    )
    ```

  - **CodeBuild Project:** Defines the build project that uses the source from CodeCommit and outputs to the S3 bucket.

    ```
    build_project = codebuild.Project(
      self, "JavaBuildProject",
      source=codebuild.Source.code_commit(repository=code_repo),
      build_spec=codebuild.BuildSpec.from_source_filename('buildspec.yml'),
      ...
    )
    ```

  - **IAM Role for CodePipeline:** A role that CodePipeline assumes to access other AWS services.

    ```
    pipeline_role = iam.Role(
      self, "CodePipelineServiceRole",
      assumed_by=iam.ServicePrincipal("codepipeline.amazonaws.com"),
      ...
    )
    ```

  - **CodePipeline:** Orchestrates the workflow of pulling code from CodeCommit, building it via CodeBuild, and handling artifacts.

    ```
    pipeline = codepipeline.Pipeline(
      self, "DeliveryPipeline",
      ...
    )
    ```

  - **Stages in Pipeline:** Define the stages and actions in the pipeline

```
source_output = codepipeline.Artifact()
build_output = codepipeline.Artifact()

pipeline.add_stage(...); pipeline.add_stage(...);
```

III. Finish the Deployment of the stack

- Run “cdk deploy” to deploy this stack to your AWS account.
- Validate: Check AWS Console to ensure all resources are created and linked correctly.
- Test the Pipeline: Make a commit to the CodeCommit repository and observe if the pipeline triggers a build and stores artifacts in S3.

IV. The difference between the new code from the code in our class lab

- The JSON CloudFormation template provides a static, declarative approach to infrastructure deployment, where every component and its configuration must be explicitly defined. This method is straightforward and ensures that resources are configured exactly as written, making it suitable for environments where changes are infrequent and predictability is crucial. However, it lacks flexibility, as any modifications require manual updates to the JSON code, which can be cumbersome and error-prone, especially as the complexity of the infrastructure grows.
- In contrast, the AWS CDK, using Python in this case, offers a dynamic, imperative approach that allows developers to use familiar programming constructs such as loops, conditions, and classes. This makes the CDK highly flexible and scalable, enabling developers to programmatically define complex environments, integrate infrastructure deployment tightly with application logic, and make modifications easily. This approach is particularly advantageous in agile development environments where infrastructure needs may evolve rapidly alongside application development.

V. Reference

- https://www.google.com/search?q=cdk+documentationn&rlz=1C5CHFA_enUS814US815&oq=cdk+documentationn&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIJCAEQABgNGIAEMgkIAhAAGA0YgAQyCAgDEAAYDRgeMggIBBAAGA0YHjIICAUQABgNGB4yCggGEAAYBRgNGB4yCggHEAAYBRgNGB4yCggIEAAYBRgNGB4yCggJEAAYBRgNGB7SAQg2NTEyajBqN6gCCLACAQ&sourceid=chrome&ie=UTF-8
- https://github.com/aws/aws-cdk
- https://jeromevdl.medium.com/document-your-aws-cdk-constructs-like-a-pro-696fc40ed39f