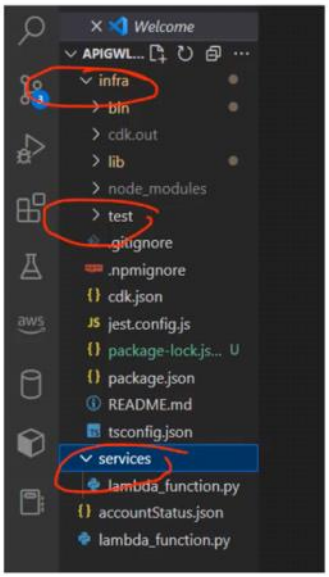
AWS CDK - 10 Best Practices based on my Cloud Migration Experience

1. Separate the Infrastructure and Application Code into separate folders



2. Single or Multi Stacks for an end to end application

- Separate out the **sensitive AWS Services** such as IAM Role, Security Group and NACL in a separate Repo
- **Rest of the AWS Services** go into a separate repo
- Build a **separate stack** for sensitive services
- Rest of the services can be deployed as **single** or multiple **stacks**
- AWS recommends keeping **stateful resources (like databases) in a separate stack** from stateless resources.
    - Turn on **termination protection** on the stateful stack.
    - Can freely destroy or create multiple copies of the stateless stack without risk of data loss.

3. Resource Naming Convention – AWS generated or customized

- AWS usually recommends to **auto-generate physical names** such as S3 bucket, APIGW and other services
- However, sometimes it's a good practice to be able to **co-relate the AWS Service Name to business unit** and **application, stage etc.**

  Naming an API GW
- **'$ (business unit name)-$ (application name)-$(stage)- apigw**
- business unit name, app name, stage etc. **can be referenced from the configuration file as an environment variable**

4. Changing the logical ID of stateful resources can impact the service due to replacement

- Changing the logical ID of a resource results in the **resource being replaced** with a new one at the next deployment.
- For **stateful resources** like databases and S3 buckets, or **persistent infrastructure** like an Amazon VPC, this may cause **serious issues if resource is replaced.**
- Make sure **refactoring of your AWS CDK code** does not impact the logical ID.
- Write **unit tests** that assert that the logical IDs of your stateful resources remain static.

5. Resource Retention policies and Log Retention

- Define a retention policy for your Storage Services – S3, RDS , EFS etc. in each Environment
- S3 default retention policy is 'Retain'
- CDK's default is to retain all logs forever

6. Application Deployment & CI-CD Pipeline is recommended to be in different AWS accounts

7. One repo across environments and deploy using the stage variable

- Create a single repository for your Infrastructure as Code and Application Code
- Deploy across the environments across the stages using the 'stage' variable in the configuration file

8. Use Secrets Manager and SSM for Storing Sensitive Values

- Use services like Secrets Manager and Systems Manager Parameter Store for sensitive values.
- Don't check in to source control, using the names or ARNs of those resources.

9. Custom constructs based on architecture patterns aligning to business domains

- Large Organizations create their own pattern to encapsulate all the resources and their default values inside a single higher-level L3 construct that can be shared.
- It can range from a simple rapper around creation of encrypted bucket or an architecture pattern
- These patterns help provision multiple resources based on common patterns with a limited knowledge in a precise manner at speed.

10. Measure everything

- Measure all aspects of your deployed resources, create metrics, alarms, and dashboards.
- Use CloudWatch, ELK/OpenSearch