AWS CDK development recommendations within AWS Organizational Structure  
 Restricted Environments + Cross-Account Deployment Pipelines

Prerequisites. All these instructions only apply to CDK V2 development and all CDK code examples and recommendations have only been applied to CDK C# development.

1. Bootstrapping An Account for CDK
2. Traditionally we have bootstrapped CDK with vanilla ‘cdk bootstrap’ command. CDK bootstrap creates several roles used within CDK commands (synth, lookup, deploy CloudFormation, deploy assets, publish ECR images, etc.). The default roles will not work in the ORG environment because of following reasons.
   1. several of those roles default to Administrator-level policies
   2. The default roles are not given Trusted permissions in other accounts.
      1. This is a problem for Cross-Account Deployments
   3. The default roles are not automatically named with correct qualifier prefix.
3. Executing **cdk --show-template** should yield you the default CloudFormation template that bootstrapping would normally use to create the necessary roles. This template needs to be saved and modified. Later we will feed the modified template back into cdk bootstrap via the **--template** argument.
   1. Apply the following change to the resource naming standard throughout:
      1. example  
         from: cdk-${**Qualifier**}-assets-key  
         to **:** ${**Qualifier**}-cdk-assets-key
   2. Note that “Qualifier” comes first.
   3. Again, apply this new standard throughout the file.
4. Bootstrap The source account and the target account. Source is where your pipeline will run, Target will the application environment and it will vary depending on the project and the environment i.e., dev, qm, staging, prod, etc.
5. Create KMS key
   1. Alias must conform to the same qualifier used throughout the project.
   2. Key policy
6. Create Pipeline Artifacts Bucket
   1. Bucket policy
7. Create Parameter Store parameters per environment for your vpc, subnets.
   1. Do not hardcode vpcid, subnet ids, security group ids into your code.
      1. You will not be able to properly change those in a HiTrust Environment
8. Cross-Account Parameter Store Patterns
   1. Custom Resource Lambda to write parameters to another account via STS
9. Modify the CDK Bootstrap Default Synthesizer in your CDK Code
10. Limit or Eliminate use of ContextHelper.cs class
    1. Traditionally it has been used to get Subnet, VPC ids by environment but it is mostly an anti-pattern.
    2. Most of it should be replaced by Template Output-Input pattern or Parameter store values.
    3. Limit StringParameter.Lookup to only when necessary, it hardcodes values in the cdk.context.json file.
11. Eliminate use of Secondary stacks that are not tied to a pipeline.
    1. This leads to confusion and scenarios where code changes are not auto deployed.
    2. A Github repo should have at least one pipeline pointing to it
       1. An exception to this would be a repo that is solely used to build a class library that is then consumed via a nuget package
    3. A github repo could have multiple pipelines pointing to it but this should be limited.
12. Override the DefaultSyntheizer