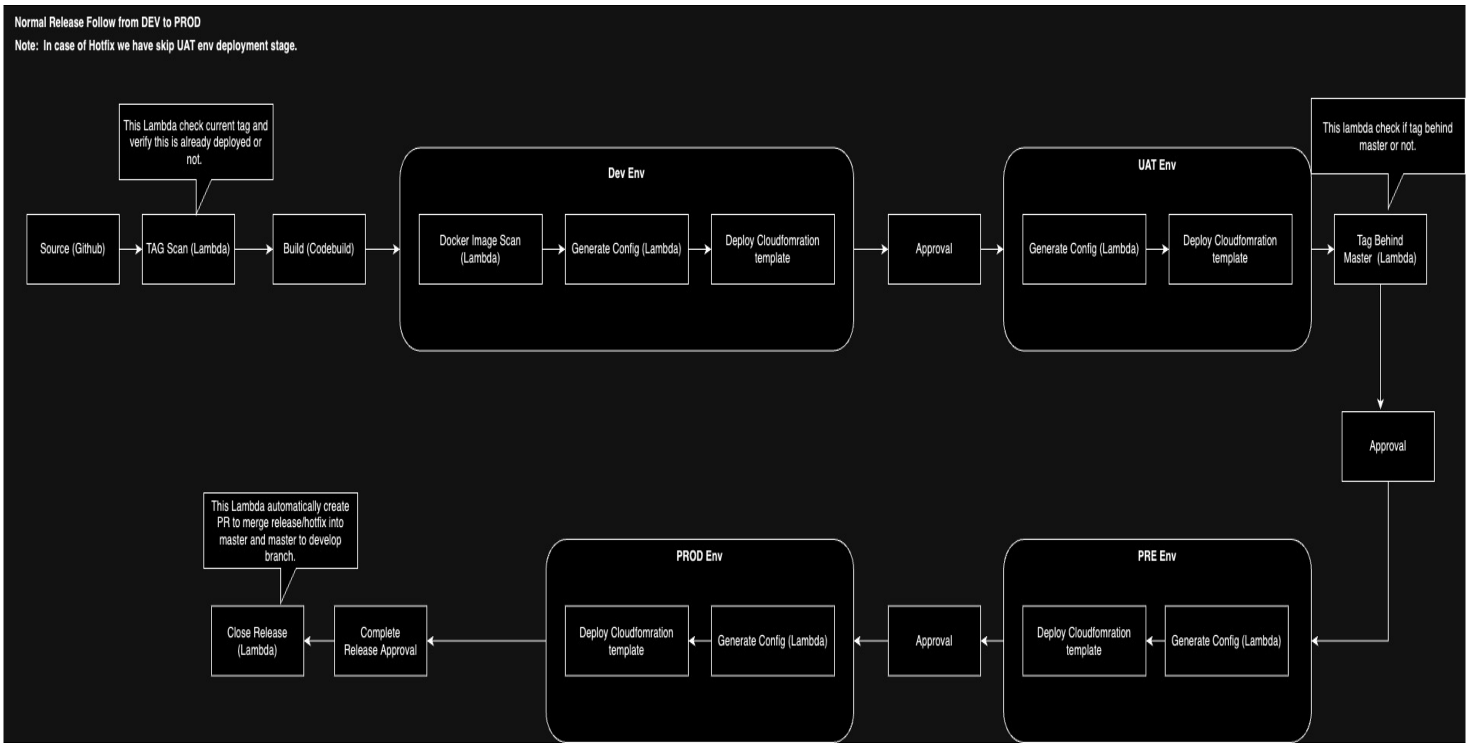
Object: Pipeline deployment for application from dev to prod.

Follow diagram:



Normal Release Follow from DEV to PROD
Note: In case of Hotfix we have skip UAT env deployment stage.
This Lambda check current tag and verify this is already deployed or not.
Source (Github)
TAG Scan (Lambda)
Build (Codebuild)
Dev Env
Docker Image Scan (Lambda)
Generate Config (Lambda)
Deploy Cloudfomration template
Approval
UAT Env
Generate Config (Lambda)
Deploy Cloudfomration template
This lambda check if tag behind master or not.
Tag Behind Master (Lambda)
Approval
PRE Env
Generate Config (Lambda)
Deploy Cloudfomration template
Approval
PROD Env
Generate Config (Lambda)
Deploy Cloudfomration template
Complete Release Approval
Close Release (Lambda)
This Lambda automatically create PR to merge release/hotfix into master and master to develop branch.

Deployment steps: Lets' assume we are using AWS as a cloud platform.

I am going to create two pipelines: one for development and another for production deployment.

The developer pipeline will only pick up code from the develop branch and deploy it in the dev environment for testing.

The production pipeline will be used to deploy from dev to prod.

Note: We will need a maintain release page where every developer and tester can input their application components' details, such as branch and version. Any other prerequisites that need to be followed during prod deployment should also be documented. Another tester needs to provide sign-off at every stage of deployment, including UAT, PRE, and Prod.

I am adding a source stage where the code comes from the release branch on GitHub (it can be any repo). Whenever a developer cuts a release branch with a version number (e.g., release/1.0.0) to deploy the application in prod, the developer provides application name and release branch details.

Source: Code is picked up from the GitHub repository, specifically from the develop branch.

TAG Scan: I am going to deploy a lambda function that will scan whether this tagged branch has already been deployed in prod. If it has already been deployed, it will reject the deployment. This helps us avoid mistakes and prevents deploying the same version multiple times.

Build: I will use a CodeBuild project to build the final artifact.

Dev Env deployment: In this section I am including three stage.

Docker image scan: It will scan all Docker images used in our project and provide the current vulnerability status. If there are any critical or high vulnerabilities, it will halt the deployment. This ensures we secure our deployment from a security standpoint.

Generate config: Here, I am including environment-specific configuration details.

Deploy: I am using a CloudFormation template stack to deploy our application in the DEV environment with all our config requirements.

UAT Approval stage: This stage is used to start UAT deployment after manual approval. Before approval, we will require tester signoff for the dev environment. If the application is working as expected in dev, then it will deploy in UAT.

UAT Env deployment: In this section I am including two stage.

Generate config: Here I am including environment-specific configuration details.

Deploy: I am using a CloudFormation template stack to deploy our application in the UAT environment with all our config requirements.

PRE Approval stage: This stage is used to start PRE deployment after manual approval. Before approval, we will require tester signoff for the UAT environment. If the application is working as expected in UAT, then it will deploy in PRE.

PRE Env deployment: In this section I am including two stage.

Generate config: Here I am including environment specific configuration details.

Deploy: I am using a CloudFormation template stack to deploy our application in the PRE environment with all our config requirements.

PROD Approval stage: This stage is used to start PROD deployment after manual approval. Before approval, we will require tester signoff for the PRE environment. If the application is working as expected in PRE, then it will deploy in PROD.

PROD Env deployment: In this section I am including two stage.

Generate config: Here I am including environment specific configuration details.

Deploy: I am using a CloudFormation template stack to deploy our application in the PROD environment with all our config requirements.

Complete release approval stage: This stage automatically approves after 2 days' time. I am giving two days' time because two days are required to check in prod, which we deployed is working as expected, and we didn't observe any issues in two days. Now we can close this release happily.

Close release: This stage has a lambda function, which we will create a new tag and pull request for release/1.0.0 or Hotfix/1.0.0 to master & master to develop. Then developers need to verify PR and approve, merge. So we can have proper sync between master and develop.

Note: In case of any issues occurring in production, we utilize the hotfix branch (Hotfix/1.0.0) to address the problem. In a hotfix scenario, we skip the UAT deployment stage and proceed to deploy directly in the PRE environment. We thoroughly test in the PRE environment, and if everything is satisfactory, we proceed with the deployment to the Production environment. All other stages remain unchanged.

Benefits:

Efficiency and Control: The pipeline streamlines deployment, providing centralized control and monitoring for a more efficient workflow.

Error Identification: Quick identification and resolution of errors ensure minimal downtime and improved system reliability.

Enhanced Visibility: Transparent tracking at each stage offers improved visibility into the deployment process.

Auditable Track Records: Comprehensive logs and track records provide auditability for compliance and historical analysis.

Multiple Deployment Instances: Supports parallel deployments of different application versions for increased flexibility.

Versioning and Tagging: Enforces a structured approach to code releases through versioning and tagging.

Efficient Collaboration: Encourages collaboration among development, testing, and operations teams.