**BLZ Bootstrap Core Products & Services Pipeline Build Strategy**

**Repository Structure**: Instead of creating branches or folders for each environment, it’s ideal for us to use a hierarchical folder structure where each product has its own folder, and within each product folder, there are sub-folders for different environments. The configuration files for each environment would live in their respective sub-folders and also for any sensitive information it would include the Keys for the HashicorpVault/CSMVault/AWS ASM which can be pulled by the Pipeline and injected during the runtime.

A screenshot of a computer program

Description automatically generated with low confidence

**\*config.yaml** - Contains environment-specific config for corresponding Product, including account number any keys for sensitive Information that can be pulled from strategic CSM Vault

**CI/CD Pipeline**: In Jenkins pipeline, we can add a stage that can dynamically read the product and environment config values i.e. changed from the commit message, then target the deployment for that specific Product and environments based on the change. This way, you can have a single Jenkinsfile for all products and environments.

**Webhooks**: Configure Bitbucket to send webhooks to Jenkins whenever a push event occurs. In Jenkins, create a job that gets triggered by these webhooks and deploys the affected products to the appropriate environments

1. **Jenkins Jobs**:
   * **Automated Deployment Job**: This job is triggered by a Bitbucket webhook. It inspects the changes made in the commit to determine which product and environments to deploy and will do the parallel deployment across environments.
     + 1. If Product/Products Config is Altered – Deploy to All Environments for those Altered Products
       2. If Env Config is Altered – Deploy to that specific Env only for those Altered Products
   * **Manual Deployment Job**: This job is parameterized and allows you to manually trigger a deployment and select the products and environments you want to target. This gives us the flexibility to override the automated deployment process when necessary.

**\*\*Prod Environment (Additional Controls) :- CR Prompt**: For production deployments, add an additional step in your Jenkins pipeline that prompts for ServiceNow CR information. This ensures that every production deployment is accompanied by the necessary change request documentation. This needs to be done for both Manual and Automated Jobs defined above

**# Monitoring/Report/Alert Helper Jobs**

* + **Email Reminder Job**: This job periodically checks for any pending approval steps and sends an email reminder to the relevant users across all the DS CoreDevOps pipelines so any pending pipelines awaiting Inputs for CR Prompts can be actioned upon making the deployment almost automated and reducing the configuration Drift. Another Ideal thing given the nature of our service offering is to have one running CR all the time for Product provisioning, support..etc.

**Repository Naming conventions:**

blz-dseng-bootstrap-core – For Core Products

blz-dseng-bootstrap-svc – For Services

blz-dseng-bootstrap-iam – For Services

blz-dseng-lambda-transform-handlers – For Tranform Macro Lambda Handlers

blz-dseng-lambda-function-utilities – Lambda Function Utilities and Lamba’s for edge-case handlings

blz-dseng-sharedlibrary/dseng-sharedlibrary

# Jenkins Stuff

First, you need to create a shared library. Let's call it **deployUtils**. Create a new Git repository for the shared library and add a file named **vars/deploy.groovy** with the following content:

// vars/deploy.groovy

def call(String product, String environment) {

// Deployment logic goes here...

echo "Deploying product ${product} to environment ${environment}"

}

Next, configure your Jenkins server to use this shared library. Go to "Manage Jenkins" > "Configure System" > "Global Pipeline Libraries", and add a new library with the name **deployUtils** and the default version **master**.

1. For the **automated deployment job**, you can create a Jenkinsfile with the following content:

See File <JenkinsFile> in the Parent Folder

1. For the **manual deployment job**, you can create a parameterized Jenkins job with choice parameters for the product and environment. In the pipeline script for this job, you can call the **deploy** function with the chosen product and environment: