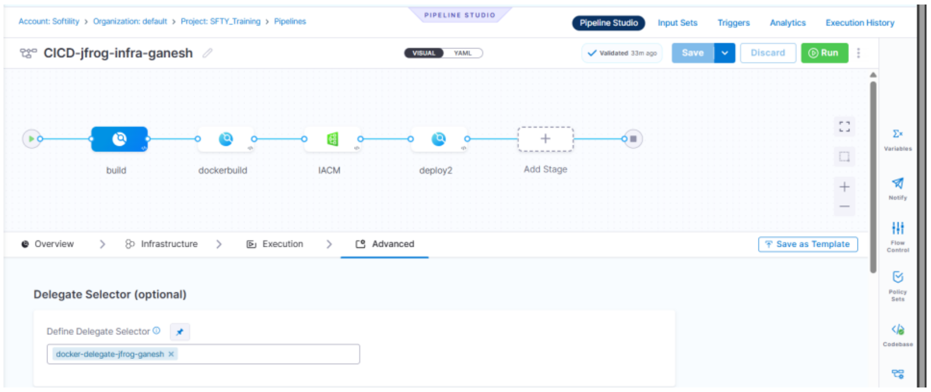
Harness jfrog connection

What I do :

- Here we create the .jar artifact and push it to jfrog
- From jfrog take the artifact (.jar) and create the Docker image
- And push the Docker image to Docker hub
- Next we create infra and to that infra we run the Docker image in it.
- We use (jfrog , Docker hub, vault [store the private key] , ansible – to install the requirements in vm)

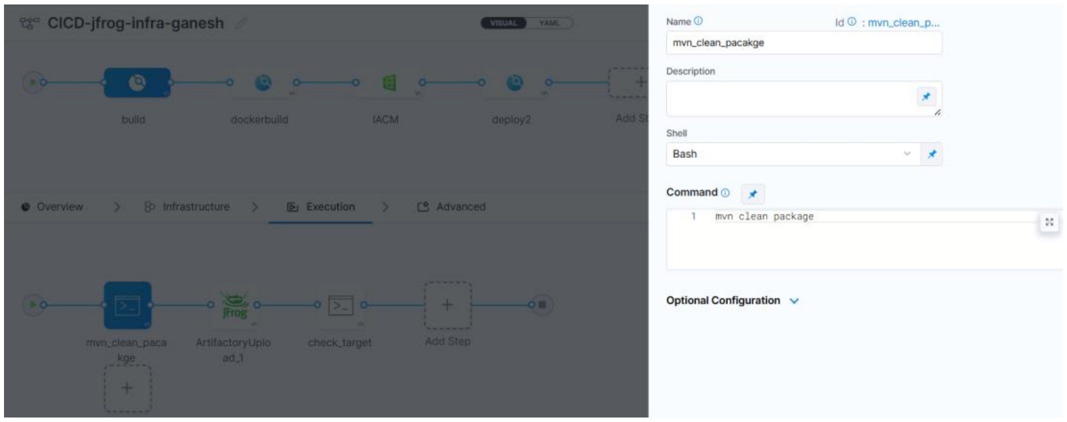


Delegate Selector (optional)

Define Delegate Selector: docker-delegate-jfrog-ganesh

stage 1: artifact build and push into jfrog

step 1: mvn clean package



Name: mvn_clean_pacakge

Id: mvn_clean_p...

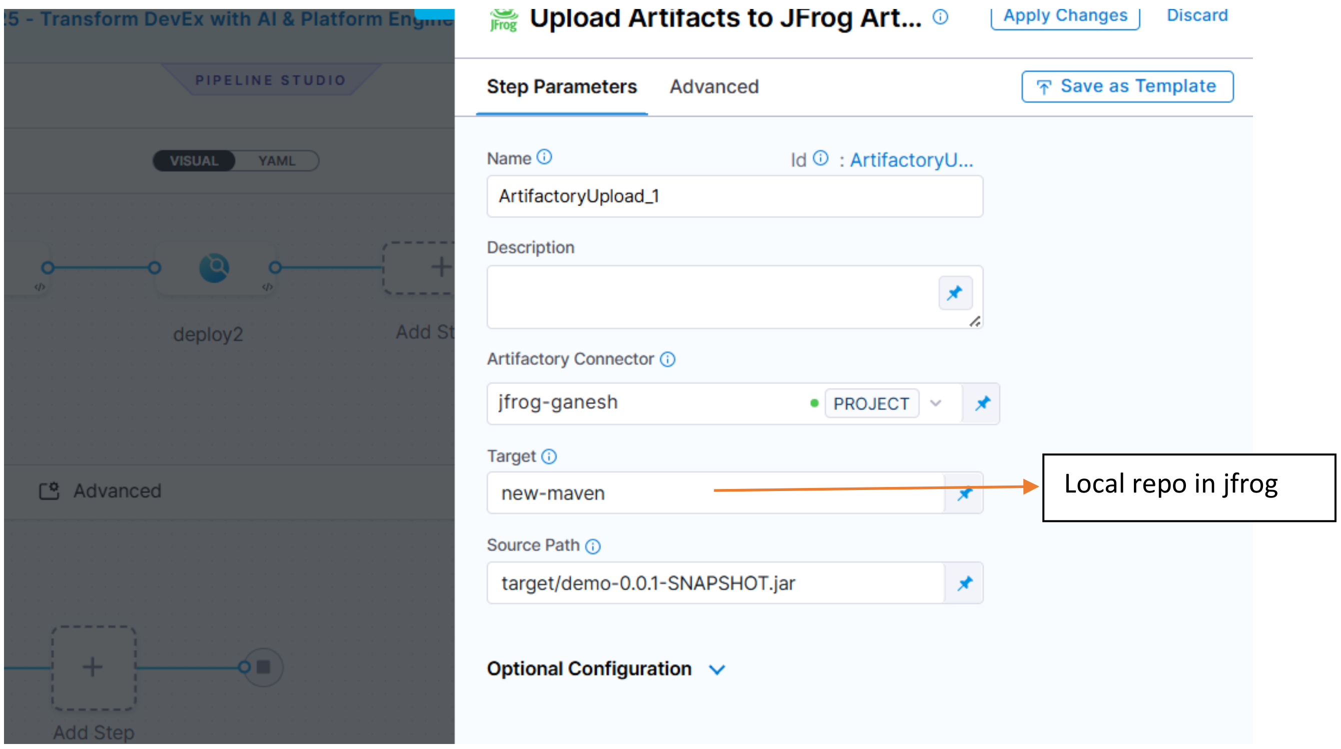
Description:

Shell: Bash

Command:

```
mvn clean package
```

step 2: jfrog push the artifact



Upload Artifacts to JFrog Art...

Step Parameters | Advanced

Name: ArtifactoryUpload_1 Id: ArtifactoryU...

Description:

Artifactory Connector: jfrog-ganesh PROJECT

Target: new-maven

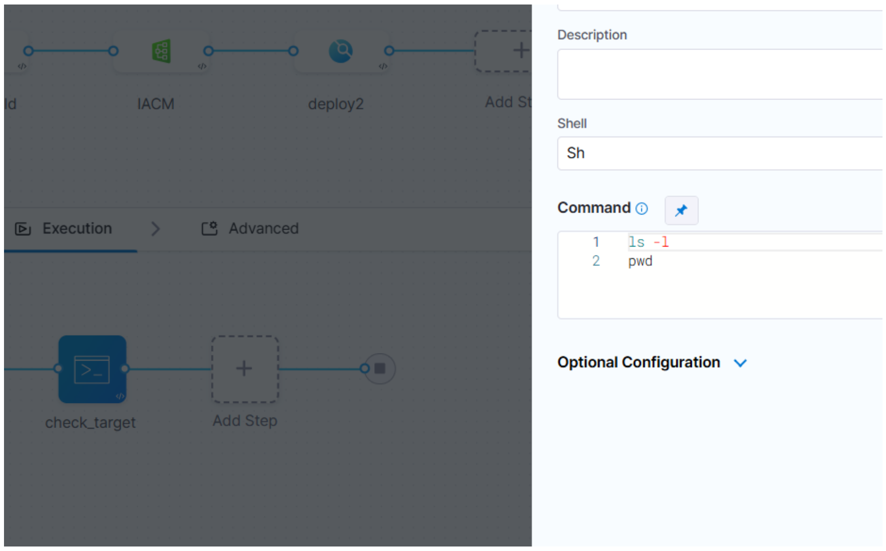
Local repo in jfrog

Source Path: target/demo-0.0.1-SNAPSHOT.jar

Optional Configuration

step 3:

check the path



Description:

Shell: Sh

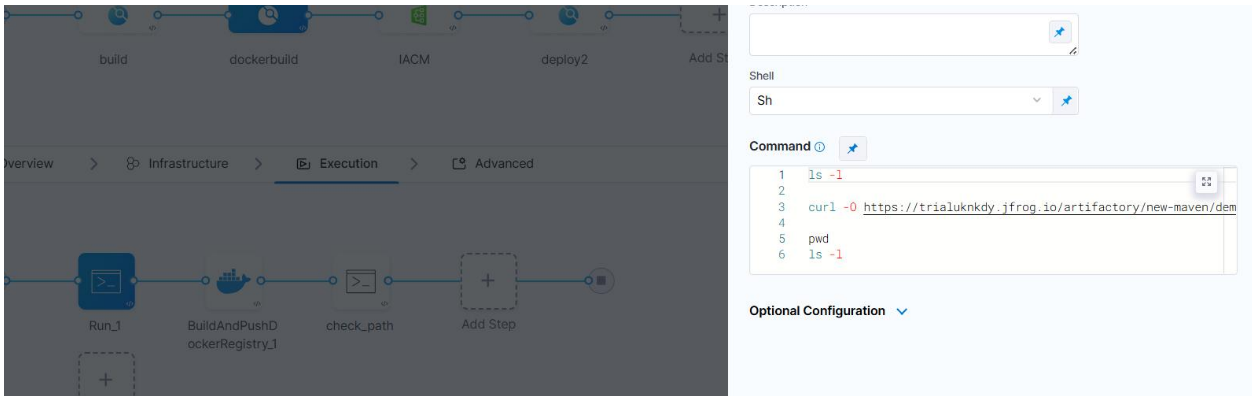
Command

```
ls -l
pwd
```

Optional Configuration

stage 2 : take the .jar artifact from jfrog and create Docker image

step 1:



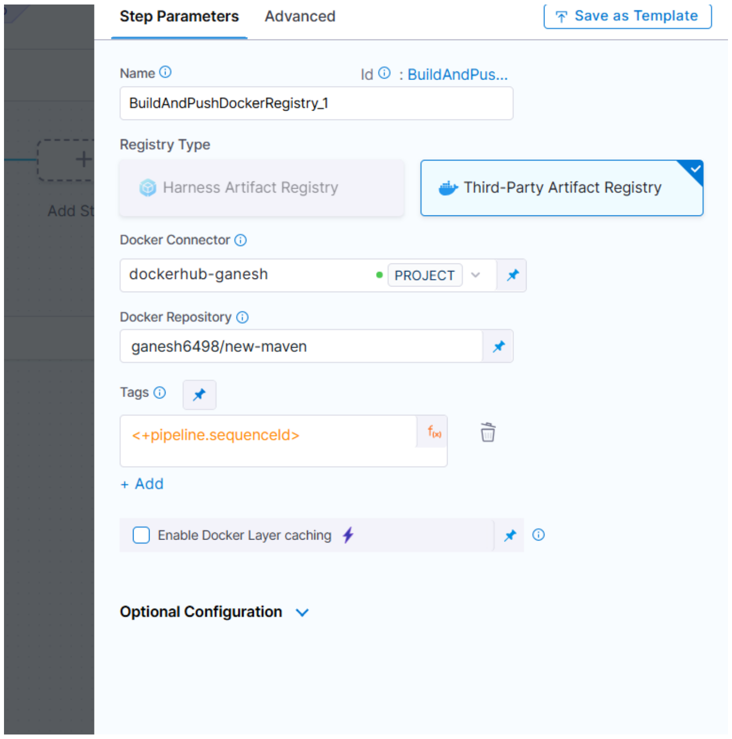
Shell: Sh

Command

```
ls -l

curl -O https://trialuknkdy.jfrog.io/artifactory/new-maven/dem

pwd
ls -l
```

```
ls -l

curl -O https://trialuknkdy.jfrog.io/artifactory/new-maven/demo-0.0.1-SNAPSHOT.jar

pwd
ls -l
```

step 2: push the image to Docker hub



Name: BuildAndPushDockerRegistry_1

Id : BuildAndPus...

Registry Type: Third-Party Artifact Registry

Docker Connector: dockerhub-ganesh PROJECT

Docker Repository: ganesh6498/new-maven

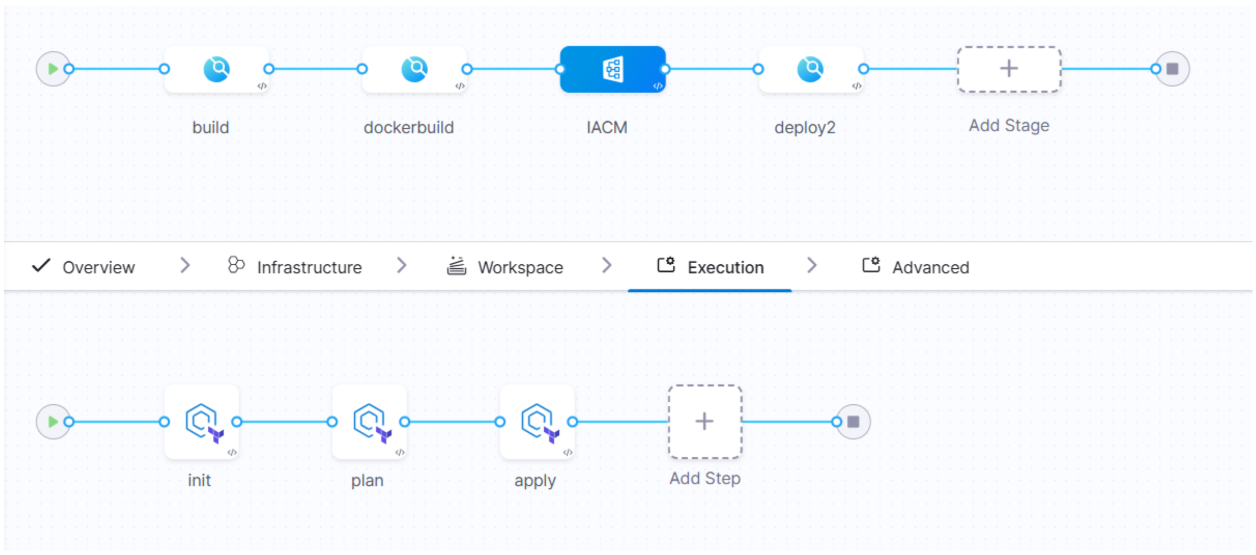
Tags: <+pipeline.sequenceId>

Enable Docker Layer caching [ ]

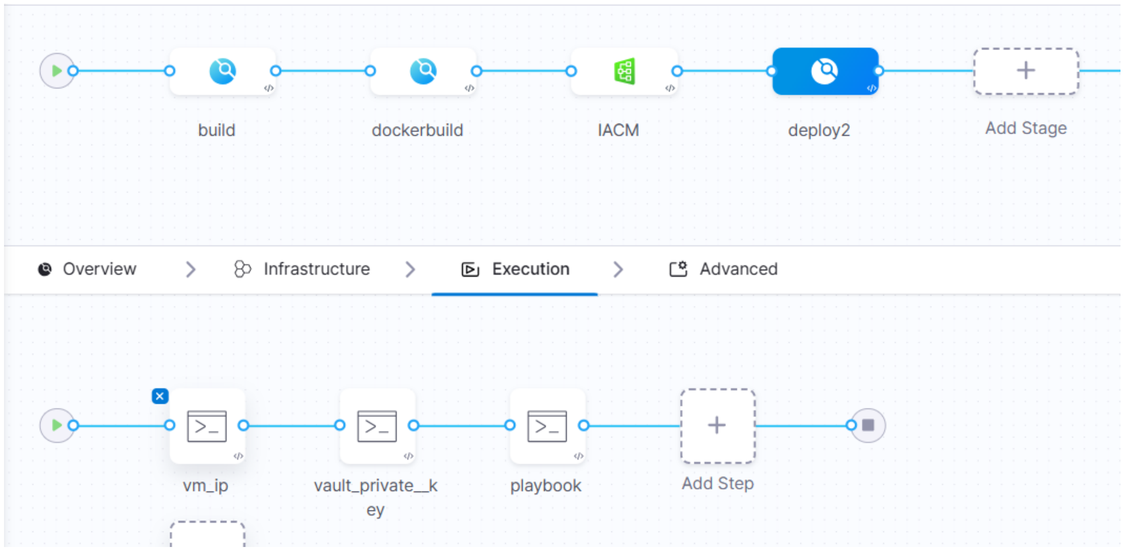
step 3: check the path

```
ls -l

pwd
```

stage 3: create infra with terraform code



stage 4: take the vm details and run the playbook



step 1: take vm_ip

we take the ip base on instance name or tags

```
gcloud compute instances list   --filter="name~'^inst-5'"   --
format="value(networkInterfaces.accessConfigs.natIP)" | tr -d "[]'"  > /etc/ansible/hosts
cat /etc/ansible/hosts
```

and these ip are store /etc/ansible/host which ws use as local vm where the delegate was running in that system it should have ansible.

step 2: from vault take the private key .

```
#!/bin/bash


# Set Vault address
export VAULT_ADDR='http://34.135.53.114:8200'

# Stop any running Vault process
pkill vault || echo "No Vault process found."

# Start Vault server in the background
nohup vault server -config=/etc/vault.d/vault.hcl > /var/log/vault.log 2>&1 &

# Wait for Vault to start
sleep 5

# Write unseal keys to /root/text
echo '<+secrets.getValue("ganesh-vault-unseal")>' > /root/text

cat /root/text

# Unseal Vault using the keys
vault operator unseal "$(cut -d ' ' -f1 /root/text)"
vault operator unseal "$(cut -d ' ' -f2 /root/text)"
vault operator unseal "$(cut -d ' ' -f3 /root/text)"

# Login to Vault
vault login '<+secrets.getValue("vault-usertoken-ganesh")>'
vault status
# List secrets at path 'my/'
vault kv list my/
vault kv get my/private_key

# Retrieve the private key and save it securely
vault kv get -field=private-key my/private_key > /tmp/privatekey
chmod 600 /tmp/privatekey
```

step 3: run the play book

```
#!/bin/bash

# Fail on error
set -e

ls -l /tmp/privatekey


# === [1] Variables ===
# Replace with your actual playbook and VM user if different
ANSIBLE_DIR="/etc/ansible"
INVENTORY_FILE="$ANSIBLE_DIR/hosts"
ANSIBLE_CFG="$ANSIBLE_DIR/ansible.cfg"
PRIVATE_KEY_PATH="/tmp/privatekey"
VM_USER="sa_106301816075024666979"



# === [4] Write Ansible config ===
cat <<EOF > "$ANSIBLE_CFG"
[defaults]
inventory = $INVENTORY_FILE
host_key_checking = False
retry_files_enabled = False
remote_user = $VM_USER
private_key_file = $PRIVATE_KEY_PATH
EOF

# === [5] Optional: Set environment to use this config ===
export ANSIBLE_CONFIG="$ANSIBLE_CFG"

# === [6] Run Ansible ping to test connection ===
ansible all -m ping
git clone https://github.com/ganesh-redy/harness-jfrog-inst.git

cd  harness-jfrog-inst
# === [7] Run your playbook ===
ansible-playbook -e "build_number=<+pipeline.sequenceId>" ansible.yaml


rm -f /tmp/privatekey
```