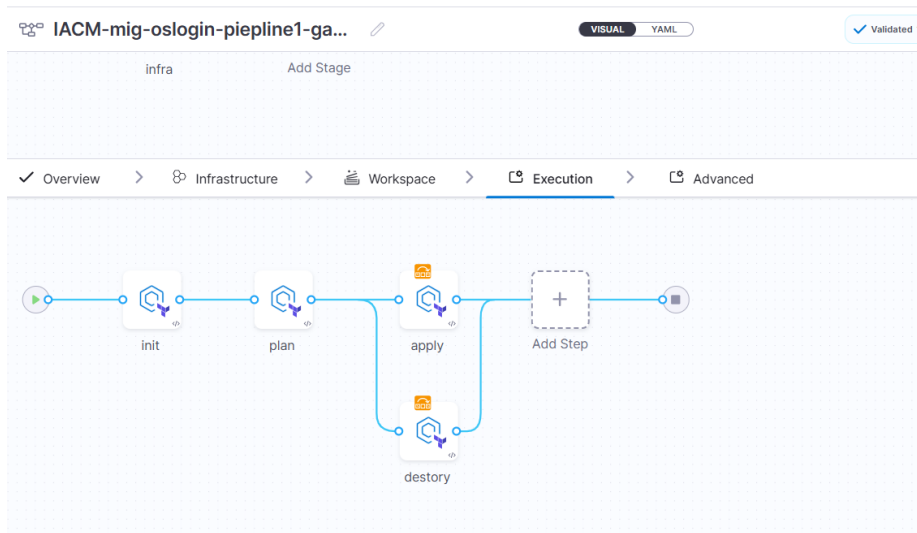


Pipeline 1:

The screenshot shows the Pipeline Studio interface for a pipeline named "IACM-mig-oslogin-pipeline1-ga...". The top navigation bar includes "Account: Softility", "Organization: default", "Project: SFTY_Training", and "Pipelines". The main header has tabs for "Pipeline Studio", "Input Sets", "Triggers", "Analytics", and "Execution". Below the header, there are buttons for "Validated 31s ago", "Save", "Discard", and "Run". The main workspace shows a visual pipeline diagram with a stage named "infra" and an "Add Stage" button. Below the diagram, there are tabs for "Overview", "Infrastructure", "Workspace", "Execution", and "Advanced". The "Infrastructure" tab is selected, showing a section titled "Infrastructure" with the instruction "Select the infrastructure where you want your provisioning to run". Below this instruction are three options: "Cloud", "Kubernetes", and "Local", with "Local" being selected.

Steps:

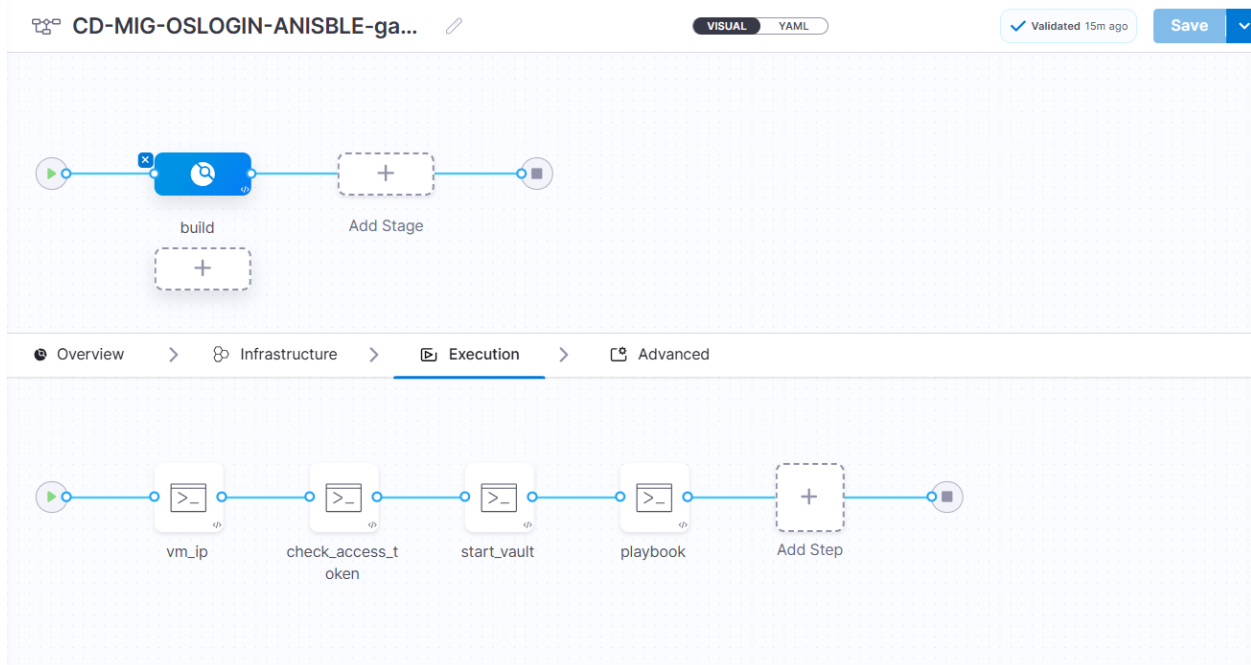


Here we just create the pipeline with variables choice
<https://github.com/ganesh-redy/mig-oslogin-ansible.git>

Pipeline 2:

Here we take ip's and store in ansible/hosts file and for ssh we take the private key from vault

Then ansible will connect to that system



Step 1: vm_ip

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

Here we take the ip's from gcp, instance named as okay because the mig instances base name is start with okay

And store in ansible/host file

step2: check_access_token

```
echo 'token <+secrets.getValue("oslogin-delegate-token")>'
```

here we have Docker delegate it consist token, that token was passed in runtime and place as variable in ansible this token was store in secrets.

Step 3: start vault

```

#!/bin/bash

# Set Vault address
export VAULT_ADDR='http://35.202.207.99:8200'

# 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

```

first we need export the vault ip of or local system then we can access ui

step 4 :

```
#!/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-reddy/mig-oslogin-ansible.git
cd mig-oslogin-ansible
# === [7] Run your playbook ===
ansible-playbook -e 'token=<+secrets.getValue("oslogin-delegate-token")>'
ansible.yaml

rm -f /tmp/privatekey
```