**GitHub Actions Yaml script for Creating Buckets:**

# This workflow installs the latest version of Terraform CLI and configures the Terraform CLI configuration file

# with an API token for the Terraform Cloud (app.terraform.io). On pull request events, this workflow will run

# `terraform init`, `terraform fmt`, and `terraform plan` (speculative plan via Terraform Cloud). On push events

# to the "main" branch, `terraform apply` will be executed.

name: 'Deploy Infra stacks in OCI'

on:

pull\_request:

types:

- closed

branches:

- "develope"

env:

PROVIDER\_NAME: "GitHub"

REGION: "us-phoenix-1"

BRANCH\_NAME: "feature-Bucket"

STACK\_NAME\_Bucket: "Stack\_Bucket\_GitHubActions"

WORKING\_DIRECTORY\_Bucket: "Bucket"

REPO\_URL: "https://github.com/Karthiga1700/DD-repo.git"

TF\_VERSION: "1.0.x"

permissions:

contents: read

jobs:

Stack\_Bucket:

runs-on: ubuntu-latest

steps:

- name: checkout

uses: actions/checkout@v2

- name: 'Writing config files'

run: |

mkdir ~/.oci

echo "[DEFAULT]" >> ~/.oci/config

echo "user=${{secrets.USER\_OCID}}" >> ~/.oci/config

echo "fingerprint=${{secrets.FINGERPRINT}}" >> ~/.oci/config

echo "compartment\_ocid=${{secrets.COMPARTMENT\_OCID}}" >> ~/.oci/config

echo "region=${{secrets.REGION}}" >> ~/.oci/config

echo "tenancy=${{secrets.TENANCY\_OCID}}" >> ~/.oci/config

echo "key\_file=~/.oci/key.pem" >> ~/.oci/config

echo "${{secrets.PRIVATE\_KEY\_FILE}}" >> ~/.oci/key.pem

echo "~/.oci/config file "

cat ~/.oci/config

echo " ~/.oci/key.pem file"

cat ~/.oci/key.pem

ls -lta ~/.oci

ls -lta ~

pwd

- name: 'Install OCI CLI'

run: |

curl -L -O https://raw.githubusercontent.com/oracle/oci-cli/master/scripts/install/install.sh

chmod +x install.sh

./install.sh --accept-all-defaults

#exec -l $SHELL

echo " Fixing file permissions and testing oci"

/home/runner/bin/oci setup repair-file-permissions --file ~/.oci/config

/home/runner/bin/oci setup repair-file-permissions --file ~/.oci/key.pem

echo " oci os ns get"

/home/runner/bin/oci os ns get

- name: 'Check for provider'

run: |

echo "SOURCE\_PROVIDER\_ID=$(/home/runner/bin/oci resource-manager configuration-source-provider list \

--compartment-id ${{secrets.COMPARTMENT\_OCID}} | jq '.data.items[] | select(."display-name"==env.PROVIDER\_NAME).id' -r)" >> $GITHUB\_ENV

- name: 'Check for existing stack compartments'

run: |

echo "STACK\_ID\_Buckets=$(/home/runner/bin/oci resource-manager stack list --all --compartment-id ${{secrets.COMPARTMENT\_OCID}} | jq '.data[] | select(."display-name"==env.STACK\_NAME\_Bucket).id' -r)" >> $GITHUB\_ENV

- name: 'Create Stack for Bucket'

if: ${{env.STACK\_ID\_Buckets == ''}}

run: |

echo "STACK\_ID\_Buckets=$(/home/runner/bin/oci resource-manager stack create-from-git-provider \

--compartment-id ${{secrets.COMPARTMENT\_OCID}} \

--config-source-configuration-source-provider-id $SOURCE\_PROVIDER\_ID \

--config-source-branch-name $BRANCH\_NAME \

--config-source-repository-url $REPO\_URL \

--display-name $STACK\_NAME\_Bucket \

--config-source-working-directory $WORKING\_DIRECTORY\_Bucket \

--terraform-version $TF\_VERSION \

--variables '{"compartment\_ocid": "${{secrets.COMPARTMENT\_OCID}}", "region": "${{env.REGION}}"}' \

--wait-for-state SUCCEEDED | jq '.data.id' -r)" >> $GITHUB\_ENV

- name: 'Update Stack for Bucket'

if: ${{env.STACK\_ID\_Buckets != ''}}

run: |

/home/runner/bin/oci resource-manager stack update-from-git-provider \

--config-source-configuration-source-provider-id $SOURCE\_PROVIDER\_ID \

--stack-id $STACK\_ID\_Buckets \

--config-source-branch-name $BRANCH\_NAME \

--config-source-repository-url $REPO\_URL \

--config-source-working-directory $WORKING\_DIRECTORY\_Bucket \

--variables file://Bucket/tfvars.jason \

--wait-for-state ACTIVE \

--force

- name: 'Terraform Plan for the stack Bucket'

if: ${{env.STACK\_ID\_Buckets != ''}}

run: |

echo "PLAN\_JOB\_ID=$(/home/runner/bin/oci resource-manager job create-plan-job \

--stack-id $STACK\_ID\_Buckets | jq '.data.id' -r)" >> $GITHUB\_ENV

- name: 'Waiting for Terraform Plan to complete'

if: ${{env.PLAN\_JOB\_ID != ''}}

run: |

while true

do

JOB\_STATUS=$(/home/runner/bin/oci resource-manager job get --job-id ${{env.PLAN\_JOB\_ID}} | jq '.data' | grep lifecycle-state | awk '{print $2}' | tr -d '"' | tr -d ',')

if [ "$JOB\_STATUS" == "SUCCEEDED" ]; then

echo "Terraform Plan: $JOB\_STATUS"

break

elif [ "$JOB\_STATUS" == "FAILED" ]; then

echo "Terraform Plan: $JOB\_STATUS"

exit 1

else

echo "Terraform Plan: $JOB\_STATUS"

sleep 10

fi

done

- name: "Terraform plan output to GitHUb"

if : ${{env.PLAN\_JOB\_ID != ''}}

run: |

/home/runner/bin/oci resource-manager job get-job-logs-content \

--job-id $PLAN\_JOB\_ID --output table > ~/tfoutput.log

- name: "Display the terraform plan on console output"

run: cat ~/tfoutput.log

- name: "Create a script to validate tf plan"

run: |

echo '#!/bin/sh' > ~/validate\_tfplan.sh

echo 'plan=$(grep -c "No changes. Infrastructure is up-to-date\|No changes. Your infrastructure matches the configuration" ~/tfoutput.log)' >> ~/validate\_tfplan.sh

echo 'if [ ${plan} -ne 0 ]' >> ~/validate\_tfplan.sh

echo 'then' >> ~/validate\_tfplan.sh

echo 'echo yes' >> ~/validate\_tfplan.sh

echo 'else' >> ~/validate\_tfplan.sh

echo 'echo no' >> ~/validate\_tfplan.sh

echo 'fi' >> ~/validate\_tfplan.sh

cat ~/validate\_tfplan.sh

chmod 755 ~/validate\_tfplan.sh

#- name: "Validate"

# run: echo "SKIP\_TF\_APPLY=$(sh ~/validate\_tfplan.sh)" >> $GITHUB\_ENV

#- name: "SKIP\_APPLY from env variables is "

# run: echo "${{env.SKIP\_TF\_APPLY}}"

#- name: 'Manually validate the terraform plan and approve the apply action'

# if: ${{env.SKIP\_TF\_APPLY == 'yes' }}

# uses: actions/checkout@v2

# with:

# secret: ${{ secrets.REPO\_TOKEN }}

# issue-title: "Stack Buckets: Deploying Terraform plan to apply"

- name: 'Terraform Apply for the stack compartments'

if: ${{env.PLAN\_JOB\_ID != ''}}

run: |

echo "APPLY\_JOB\_ID=$(/home/runner/bin/oci resource-manager job create-apply-job \

--execution-plan-strategy FROM\_PLAN\_JOB\_ID \

--execution-plan-job-id $PLAN\_JOB\_ID \

--stack-id $STACK\_ID\_Buckets \

| jq '.data.id' -r)" >> $GITHUB\_ENV

- name: 'Wait for Terraform Apply to complete'

if: ${{env.APPLY\_JOB\_ID != ''}}

run: |

while true

do

JOB\_STATUS=$(/home/runner/bin/oci resource-manager job get --job-id ${{env.APPLY\_JOB\_ID}} | jq '.data' | grep lifecycle-state | awk '{print $2}' | tr -d '"' | tr -d ',')

if [ "$JOB\_STATUS" == "SUCCEEDED" ]; then

echo "Terraform Apply: $JOB\_STATUS"

break

elif [ "$JOB\_STATUS" == "FAILED" ]; then

echo "Terraform Apply: $JOB\_STATUS"

exit 1

else

echo "Terraform Apply: $JOB\_STATUS"

sleep 30

fi

done





