# FRONT END & MIDDLEWARE

Most of the deployment is done with Terraform; however, you must first push build and push the Node servers to the MOC's Docker Registry.

## Prerequisites

- In order to push to the Docker registry on MOC you need to request access via a ticket
- MOC provides X509 self signing certs so the following must be added to **/etc/docker/daemon.json:**
  *{*
  *"insecure-registries" : ["docker-registry-default.k-apps.osh.massopen.cloud"]*
  *}*
- The Kubernetes provider in Terraform needs a login token generated from the openshift frontend (see image below) or the Openshift CLI tool using (oc whoami -t). This token should be placed in **$PROJECT_DIR/terraform_oc/provider.tf**

## Project Setup

```
#install the two git projects
git clone $PROJECT_GIT_URL
git clone $SID_PROJECT_URL
#move the docker files to sid
cp $PROJECT_DIR/scripts/docker/* $SID_PROJ_DIR/

#cd to the sid directory
cd $SID_PROJ_DIR

#build, and tag images to MOC
docker build -t <MOC_DOCKER_URL>/<PROJECT_NAME>/gulp:latest -f Dockerfile-gulp .
docker build -t <MOC_DOCKER_URL>/<PROJECT_NAME>/worker:latest -f Dockerfile-worker .

#push images to MOC
docker push <MOC_DOCKER_URL>/<PROJECT_NAME>/gulp:latest
docker push <MOC_DOCKER_URL>/<PROJECT_NAME>/worker:latest

#cd back to project
cd $PROJECT_DIR/terraform-oc/
#modify provider token
Vim provider.tf
```

## Using Terraform

After running the terraform apply command in the following folder the openshift console will display your deployment.

```
cd $PROJECT_DIR/terraform-oc
#run terraform init to download provider files
terraform init
#terraform apply will apply the current config
terraform apply
#terraform destroy will remove the current config
```

```
terraform destroy
```

## BACKEND

Install Terraform on your machine(recommend: brew install terraform)

clone/download repo：https://github.com/BU-CLOUD-S20/MOC-Research-Computing

```
cd /PATH/TO/MOC-Research-Computing-master/terraform-backend-sid

sudo pip3 install -r requirements.txt
```

Add configuration detail at ~/.config/openstack/clouds.yaml in following format:

```
clouds:
  openstack:
    auth:
      auth_url: https://kaizen.massopen.cloud:13000/v3
      username: <USERNAME>
      project_name: <PROJECT_NAME>
      project_id: <PROJECT_ID>
      user_domain_id: <USER_DOMAIN_ID>
      password: <PASSWORD>
    region_name: "moc-kzn"
    interface: "public"
    identity_api_version: 3
```

```
cd /PATH/TO/MOC-Research-Computing-master/terraform-backend-sid/inventory/k8-test-cluster
```

Open (or vim) cluster.tfvars, and add value for "public_key_path"

```
export OS_CLOUD=openstack

terraform init ../../contrib/terraform/openstack

terraform apply -var-file=cluster.tfvars ../../contrib/terraform/openstack
```

When you get a success message, check on OpenStack Dashboard to see if instances/networks/routers/security groups are created and configured.

Open (or vim) /PATH/TO/MOC-Research-Computing-master/terraform-backend-sid/inventory/k8-test-cluster/group_vars/all/all.yml and add value for "openstack_password"

```
eval $(ssh-agent -s)

ssh-add ~/.ssh/<YOUR SSH ADDRESS>

cd /PATH/TO/MOC-Research-Computing-master/terraform-backend-sid

#OPTIONAL PING:
ansible -i inventory/k8-test-cluster/hosts -m ping all

ansible-playbook --become -i inventory/k8-test-cluster/hosts cluster.yml
```

You should see it run checks. Could take up to 20 min.It is only successful if you see no error message.

## For Teardown:

```
cd /PATH/TO/MOC-Research-Computing-master/terraform-backend-sid/inventory/k8-test-cluster

terraform destroy -var-file=cluster.tfvars ../../contrib/terraform/openstack
```