**Step by Step PlatformOne BigBang DSOP Deployment**

Deploying Big Bang is a two stage process.

* Stage 1: Deployment of an RKE2 cluster (**DSOP-RKE2**)
* Stage 2: Deployment of Big Bang on the RKE2 cluster created in stage 1 (**DSOP-ENVIRONMENT**)

Mandatory tools and accounts access required for P1 Bigbang DSOP deployment.

* 1. **Accounts Access:**

|  |  |
| --- | --- |
| **Azure DevOps** | <https://azure.microsoft.com/en-us/services/devops/> |
| **Iron Bank Account** | **Existing Login:** <https://ironbank.dso.mil/repomap/products?page=1&sort=1>  **New Registration:** <https://login.dso.mil/auth/realms/baby-yoda/protocol/openid-connect/registrations?client_id=account&response_type=code> |
| **GitHub** | [**https://github.com/**](https://github.com/) |
| **Azure Portal** | Azure subscription with full access. Should be able to create resources as an Administrator.  <https://portal.azure.us/#home> |

* 1. **Required Tools:**

The following commands to install on **Windows PC / WLS 2 / Ubuntu 18.04 LTS**

**Tools scripts can be found:** [**https://github.com/benc-uk/tools-install**](https://github.com/benc-uk/tools-install)

**https://github.com/cheruvu1/dsop-environment-1.19.0/blob/main/dsop-tools.sh**

|  |  |
| --- | --- |
| **Bash** | (Linux / WSL2 / MacOS - Terminal) |
| **Terraform** | sudo apt-get update    wget <https://releases.hashicorp.com/terraform/1.1.4/terraform_1.1.4_linux_amd64.zip>  sudo apt-get install zip -y  unzip terraform\*.zip  sudo mv terraform /usr/local/bin  terraform version |
| **~~Chocolatey (Optional)~~** | [~~https://chocolatey.org/install~~](https://chocolatey.org/install)  [~~https://www.educba.com/linux-jq/~~](https://www.educba.com/linux-jq/)  ~~Install JQ: (JQ is a lightweight and flexible command-line JSON processor)~~   * + **~~chocolatey install jq~~** |
| **JQ** | **Below commands, install JQ:**  sudo apt-get update  sudo apt install jq  jq --version |
| **Kubectl**  >= 1.21.0 | curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"  curl -LO "https://dl.k8s.io/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"  echo "$(<kubectl.sha256) kubectl" | sha256sum --check  sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl  sudo chmod +x kubectl  sudo mkdir -p ~/.local/bin/kubectl  sudo mv ./kubectl ~/.local/bin/kubectl  # and then add ~/.local/bin/kubectl to $PATH  kubectl version |
| **Azure CLI** | curl -sL <https://aka.ms/InstallAzureCLIDeb> | sudo bash  az version |
| **gpg** | sudo apt-get install -y gpg  gpg --version |
| **Sops** | Download the latest sops version: **sops 3.7.1**  wget <https://github.com/mozilla/sops/releases/download/v3.7.1/sops_3.7.1_amd64.deb>  sudo dpkg -i sops\_3.7.1\_amd64.deb  sops --version  sops 3.7.1 (latest) (OUTPUT) |
| **Kustomize** | sudo curl -s "https://raw.githubusercontent.com/kubernetes-sigs/kustomize/master/hack/install\_kustomize.sh" | bash  sudo mv kustomize /usr/local/bin  kustomize version |
| **Flux** | curl -s <https://fluxcd.io/install.sh> | sudo bash |

**Stage 1**: Deployment of an RKE2 cluster (**DSOP-RKE2**)

**Azure DevOps Repository:** <https://azure-ecosystem.visualstudio.com/Azure%20Gov%20Engineering/_git/dsop-rke2>

OR

**GitHub:** <https://github.com/cheruvu1/dsop-rke2>

**Following tools required for the dsop-rke2:**

**Note:** Following commands are compatible with **Ubuntu Linux System [Windows PC / WLS 2 / Ubuntu 18.04 LTS]**

**Follow the below steps to install dsop-rke2:**

|  |  |
| --- | --- |
| Step1 | **Azure DevOps Repository:**  git clone <https://azure-ecosystem.visualstudio.com/Azure%20Gov%20Engineering/_git/dsop-rke2>  OR  **GitHub Repository:**  git clone <https://github.com/cheruvu1/dsop-rke2>  code .  (Open the dsop-rke2 in Visual Studio Code or preferred IDE)    For VNET Customization, use the following Repository:  git clone <https://github.com/cheruvu1/dsop-rke2-vnet-customization.git>  code .  (Open the dsop-rke2 in Visual Studio Code or preferred IDE) |
| Step2 | cd example (GoTo Example folder)  copy `terraform.tfvars.sample` to `terraform.tfvars`  Line Number 2: Change `cluster\_name` and other settings, but most can be left as the defaults  **Example: cluster\_name = "rke2-lmco-example" (Default Value: rke2-example)**  **Note:** Keep the middle name 3 or 4 characters length.  Make sure the following set to true.  **# Connectivity options**  server\_public\_ip = true  server\_open\_ssh\_public = true |
| Public IP | Default the public IP connection is true. Please change it to false, if you don’t want to expose your  Cluster to public.  # Connectivity options  server\_public\_ip = **false**  server\_open\_ssh\_public = **false** |
| Own VNET | If you want to use your own VNET instead of script created VNET, please replace the following 4 parameters.  # Networking  use\_external\_vnet = true  external\_vnet\_resource\_group = "rke2-lmi-example"  external\_vnet\_name = "rke2-lmi-vnet"  external\_vnet\_subnet\_name = "lmisubnet" |
| Step3 | **Login to Azure Portal using Command line:**  az cloud set --name AzureUSGovernment (Switch to Azure US Government, if pointing to Azure Commercial)  az cloud list --output table  az login  **Run Terrform commands: Go To dsop-rke2/example folder**  sudo terraform init  sudo terraform apply -auto-approve |
| Step4 | **Folder:** /dsop-rke2/example  terraform output -raw kv\_name  KV\_NAME=${1:-$(terraform output -raw kv\_name)}  echo $KV\_NAME    source ../scripts/fetch-kubeconfig.sh  FILE=$(realpath rke2.kubeconfig)  az keyvault secret show --name kubeconfig --vault-name $KV\_NAME | jq -r '.value' > $FILE  echo $FILE  export KUBECONFIG=$PWD/rke2.kubeconfig  echo $KUBECONFIG |
| Step5 | Download the Private Key  Copy the key vault name: **rke2-lmco-example-ao6** (replace below line)  az keyvault secret show --name node-key --vault-name rke2-example-alu | jq -r '.value' > rke2.priv\_key  cat rke2.priv\_key |
| ~~Step6~~ | ~~Execute the following command from terraform state folder..~~  ~~source ../scripts/fetch-kubeconfig.sh~~  ~~(script creates the file --> rke2.kubeconfig)~~ |
| Step7 – Is needed to ssh to the server. | sudo chmod 400 rke2.priv\_key  Go To Azure Portal Console  Open the resource group: rke2-lmco-example  Open type Public IP resource: rke2-lmco-example-wyf-pip (name may change)  ssh rke2@ 20.141.216.226 -p 5001 -i rke2.priv\_key (Optional)  kubectl get nodes  kubectl get nodes -A |
| **Final step** | This concludes the RKE2 cluster deploy. |

**Stage 2:** Deployment of Big Bang on the RKE2 / AKS cluster created in stage 1 (**DSOP-ENVIRONMENT**)

1. **DSOP-ENVIRONMENT Repo:**

**Azure DevOps Repository:** <https://azure-ecosystem.visualstudio.com/Azure%20Gov%20Engineering/_git/dsop-environment>

OR

**GitHub:** <https://github.com/cheruvu1/dsop-environment>

1. **Setup Instructions:**

**Option 1:** **Readme file contains step by step instructions for the PlatformOne DSOP installation...**

<https://azure-ecosystem.visualstudio.com/Azure%20Gov%20Engineering/_git/dsop-environment?path=/readme.md>

**Option 2: Follow below steps: PlatformOne BigBang Environment Setup:**

This is a set of manual pre-req steps that has to be done, and can't realistically be scripted

### Set Up Git Repo:

|  |  |
| --- | --- |
| **Git Clone** | **Azure DevOps Repository:**  Clone this repo to your machine, you can use your personal Azure AD account to do this.  sudo git clone [https://azure-ecosystem.visualstudio.com/Azure Gov Engineering/\_git/dsop-environment](https://azure-ecosystem.visualstudio.com/Azure%20Gov%20Engineering/_git/dsop-environment)  sudo git clone <https://azure-ecosystem.visualstudio.com/Azure%20Gov%20Engineering/_git/dsop-environment>  OR  **GitHub Repository: /saic (Root folder, wherever dsop-rke2 rolder**  sudo git clone <https://github.com/cheruvu1/dsop-environment> |
| **GitOps need your own branch.**  Push branch | Create a new branch and name it, a suggestion is to place env/ as a prefix in front of the branch name, e.g. **env/saic** , to identify each developer's own environment branch  Push branch back to remote origin so it is tracked, e.g. git push --set-upstream origin {branch-name}  cd dsop-environment  sudo git checkout -b env/ver28 origin/main  sudo git branch -v  sudo git push --set-upstream origin env/ver28 |
| Generate Git Credentials | **Azure DevOps:**  Create a set of credentials to clone the repo, these will be used by Flux, you cannot use your Azure AD account or credentials. From [the Azure DevOps page for this repo](https://azure-ecosystem.visualstudio.com/Azure%20Gov%20Engineering/_git/dsop-environment)   * + Click 'Clone' button again   + Click 'Generate Git Credentials' button   + Make a note of the username and password generated, they are needed for secrets.sh   **GitHub:**  From your GitHub account,  go to Settings => Developer Settings => Personal Access Token => Generate New Token (Give your password) => Fill-up the form => click Generate token => Copy the generated Token |

**Generate wildcard certificate for your domain**

|  |  |
| --- | --- |
| **Self signed Certificate** | A certificate for non-production environments can be generated by executing the following steps:  **Folder Location: dsop-environment**  HOSTNAME=bigbang.dev  ./scripts/create-root-cert.sh  ./scripts/create-domain-cert.sh $HOSTNAME  ISTIO\_GW\_CRT=$(cat $HOSTNAME.crt | base64 -w0)  echo $ISTIO\_GW\_CRT  ISTIO\_GW\_KEY=$(cat $HOSTNAME.key | base64 -w0)  echo $ISTIO\_GW\_KEY |
| **Key Vault stored certificate** | export ISTIO\_GW\_CRT="<certificate id in keyvault>"  (copy the output from echo $ISTIO\_GW\_CRT)  export ISTIO\_GW\_KEY="<certificate id in keyvault>"  (copy the output from echo $ISTIO\_GW\_KEY)  **If the scripts already executed once,**  If your certificate is stored already as secrets in keyvault set ISTIO\_GW\_CRT and ISTIO\_GW\_KEY to the keyvault id of those secrets in secrets.sh  If your certificate is stored already as secrets in keyvault set USE\_KEYVAULT\_CERT to true on deploy-vars.sh Changing certificate If your certificate was changed change the value in secrets.sh and deploy-vars.sh them execute update-certs.sh |

### Configure For GitOps

|  |  |
| --- | --- |
| dev/bigbang.yaml file | **VisualStudio or IDE :**  Update the dev/bigbang.yaml file,   * 1. **LINE#14**: url: <https://github.com/cheruvu1/dsop-environment>   2. **LINE#16**: place your own branch name where it has \_\_CHANGE\_ME\_\_   Example: branch: env/saic |
| save and commit your change | Optional:  git config --global user.email "scheruvu@microsoft.com"  git config --global user.name "Bharadwaja Cheruvu"  git add dev/bigbang.yaml  git status  git commit -m "updated dev/bigbang.yaml"  git push OR  sudo git push --set-upstream origin env/saic |

### Deploy:

|  |  |
| --- | --- |
| Modify Scripts/secrets.sh file – Add credentials info. | 1. Folder: DSOP\_ENVIRONMENT/Scripts.   Copy**secrets.sh.sample to secrets.sh** and edit to with your own values and secrets as follows:   * Set IRON\_BANK\_USER & IRON\_BANK\_PAT with the Username and CLI secret from your User Profile on [https://registry1.dso.mil](https://registry1.dso.mil/)  (After logging in click your username in the upper righthand corner). * Set AZDO\_USER & AZDO\_PASSWORD with the credentials you generated in step 2 * Set ISTIO\_GW\_CRT & ISTIO\_GW\_KEY with the certificates from step 3.   To get the IronBank credentials:  Login: <https://registry1.dso.mil/harbor/projects>  IRON\_BANK\_USER = **Username**  IRON\_BANK\_PAT = **CLI Secret** |
| Application specific changes, ex: NAMESPACE="bigbang" | **Folder: DSOP\_ENVIRONMENT/Scripts**  Copy deploy-vars.sh.sample to deploy-vars.sh and configure as you wish |
| Run the automated deployment script | cd dsop-environment/scripts  **sudo ./deploy.sh**  (Keep your Azure DevOps User ID and Password Handy, you might need, deployment will ask multiple times) |
| This script will carry out the following: | 1. One time creation of GPG keys and update to .sops.yaml if keys are found to exist, this step is skipped. 2. Creation/update of secrets.enc.yaml and pushed with git 3. *OPTIONAL: Deployment of AKS cluster.* 4. *OPTIONAL: Connection to AKS cluster for kubectl etc* 5. Creation of namespaces: bigbang and flux-system 6. Creation of secrets: sops-gpg, private-registry & private-git 7. Deployment of Flux from the main bigbang repo which will be cloned and scripts/install\_flux.sh run. This can be disabled by setting DEPLOY\_FLUX=false. 8. Removes network policies which block Flux being scraped 9. Deploys the dev/bigbang.yaml to the cluster 10. Validates the status of the deployment |
| Status of what was just deployed. | kubectl get gitrepositories,ks,hr -A  kubectl get pods -A  kubectl get nodes  kubectl get vs -A (Show the Hosts Information)  kubectl get all -n hello-world (Run this command, if hello-world deployed DONE using GitOps)  kubectl get all -n istio-system  kubectl get hr -A ?  flux check  flux get all - A  flux get hr bigbang -A  flux logs --kind=Kustomization -n bigbang  kubectl get hr,po -A |

### Configure local domain to IP address mapping:

|  |  |
| --- | --- |
| IP address mapping | In dev, when using a domain name not recorded in a DNS server, if we want to access the virtual services created by Bigbang, we can add the IP address - domain mapping to /etc/hosts running the following commands: |
| **Get IP Address** | Go To the folder: /mnt/c/Work/lmco/dsop-rke2/example (**RKE2 setup location**)  **# get istio gateway ip**  bigbang v < 1.13  ip=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.status.loadBalancer.ingress[0].ip}')  bigbang v >= 1.13  ip=$(kubectl -n istio-system get service public-ingressgateway -o jsonpath='{.status.loadBalancer.ingress[0].ip}')  echo $ip  **# get domains**  domains=$(kubectl --kubeconfig rke2.kubeconfig get virtualservices -A -o jsonpath="{ .items[\*].spec.hosts[\*] }")  echo $domains  # add entry in /etc/hosts  echo "$ip $domains" | sudo tee -a /etc/hosts |
|  | Windows hostfile Location  **C:\Windows\System32\drivers\etc**  52.245.213.246 tracing.bigbang.dev  52.245.213.246 kiali.bigbang.dev  52.245.213.246 kibana.bigbang.dev  52.245.213.246 alertmanager.bigbang.dev  52.245.213.246 grafana.bigbang.dev  52.245.213.246 prometheus.bigbang.dev  52.245.213.246 twistlock.bigbang.dev  52.245.213.246 helloworld.bigbang.dev  52.245.218.56 currency-exchange.bigbang.dev  52.245.218.56 **sonarqube.bigbang.dev** |

**Install the Certificates:**

|  |
| --- |
| * + Open the file **bigbang.dev.cert & ca.cert** from the location: \dsop-environment)   + Using Windows Explorer right mouse à Open à bigbang.dev.crt & ca.crt files |
| * + Install Certificate button |
|  |
| **Note:** You might need to restart the pc, to take certificate changes effect. |

## **Test Bigbang deployment Using Browser:**

How to get credentials:

<https://repo1.dso.mil/platform-one/big-bang/bigbang/-/blob/master/docs/guides/using_bigbang/default_credentials.md>

|  |  |
| --- | --- |
| Grafana | <https://grafana.bigbang.dev/login> |
| Kiali | <https://kiali.bigbang.dev/kiali> |
| Kibana | <https://kibana.bigbang.dev/login?next=%2F> |
| TwistLock | <https://twistlock.bigbang.dev/#!/login> |
| Prometheus alert manager | <https://alertmanager.bigbang.dev/#/alerts> |
| Prometheus Graph | <https://prometheus.bigbang.dev/graph> |
| Jaeger | <https://tracing.bigbang.dev/search> |
| HelloWorld – GitOps Flux | <https://helloworld.bigbang.dev/login> |

|  |  |
| --- | --- |
| Grafana | User ID: admin / Password: prom-operator |
| Kiali | kubectl get secret -n kiali | grep kiali-service-account-token | awk '{print $1}' | xargs kubectl get secret -n kiali -o go-template='{{.data.token | base64decode}}' |
| Kibana | User ID: elastic  kubectl get secrets -n logging logging-ek-es-elastic-user -o go-template='{{.data.elastic | base64decode}}' |
| TwistLock | <https://twistlock.bigbang.dev/#!/login>  Create Account, after login. |
| Prometheus alert manager | <https://alertmanager.bigbang.dev/#/alerts> |
| Prometheus Graph | <https://prometheus.bigbang.dev/graph> |
| Jaeger | <https://tracing.bigbang.dev/search> |
| HelloWorld – GitOps Flux | <https://helloworld.bigbang.dev/login> |

## **Test Bigbang deployment Using Python Scripts :**

|  |  |
| --- | --- |
| **Install the Python3** | Go to à cd dsop-environment/tests  This test required, Python 3.8.10:  sudo apt install python3.8-venv  python3 –version |
|  | Tests are written in python; in order to run them follow the steps below in the  dsop-environment directory: /dsop-environment |
| 1. Create virtual environment | sudo apt-get install python3-venv  sudo apt-get install pip  sudo /usr/bin/python3 -m venv ./venv |
| 2. Activate environment | source ./venv/bin/activate |
| 3.Install requirements | cd .. (Should be in the folder location: dsop-environment)  pip install -r requirements.txt |
| 4.Run tests | pytest ./tests -v |
| **5.Test Output** |  |

**Troubleshooting:**

|  |  |
| --- | --- |
| Error | Solution |
| Error! kubectl is not pointing at a cluster, configure KUBECONFIG or /home/bharadwaja/.kube/config | Go To the folder: dsop-rke2/example  cp rke2.kubeconfig /home/bharadwaja/.kube/config  cd dsop-environment/scripts  **./deploy.sh** |
| You are connected to Kubenetes: https://20.141.202.94:6443  Do you want to proceed and deploy BigBang (y/n) ? y  ### 🔑 Key bigbang-sops (0212AA298D2930DBB9E7A9E0A91709726FD29DCA) already exists, skipping creation  ### 📝 Creating & encrypting dev/secrets.enc.yaml  [PGP] WARN[0000] Deprecation Warning: GPG key fetching from a keyserver within sops will be removed in a future version of sops. See https://github.com/mozilla/sops/issues/727 for more information.  Could not generate data key: [failed to encrypt new data key with master key "FALSE\_KEY\_HERE": could not encrypt data key with PGP key: golang.org/x/crypto/openpgp error: key with fingerprint FALSE\_KEY\_HERE is not available in keyring and could not be retrieved from keyserver; GPG binary error: gpg binary failed with error: exit status 2, gpg: 'FALSE\_KEY\_HERE' is not a valid long keyID  ] | Copy the Key bigbang-sops key to line #4 in the file dsop-environment à  .sops.yaml file |