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/**
Azure Portal	Azure subscription with full access. Should be able to create resources as an Administrator. https://portal.azure.us/#home

2. 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

Bash	(Linux / WSL2 / MacOS - Terminal)
Terraform	sudo apt-get update wget https://releases.hashicorp.com/terraform/1.0.1/terraform_1.0.1_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://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 –client
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

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:** 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) echo $FILE az keyvault secret show --name kubeconfig --vault-name $KV_NAME | jq -r '.value' > $FILE export KUBECONFIG=$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-lmco-example-ao6** | 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@52.227.192.136 -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 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

2) **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 OR **GitHub Repository:** sudo git clone https://github.com/cheruvu1/dsop-environment
	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/dbowie , to identify each developer's own environment branch
GitOps need your own branch. Push branch	Push branch back to remote origin so it is tracked, e.g. git push --set-upstream origin {branch-name} cd dsop-environment git checkout -b env/bha origin/main

	git branch -v git push --set-upstream origin env/bha
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 ○ 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

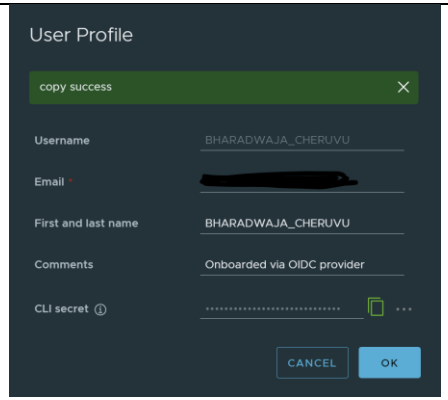
Configure For GitOps

dev/bigbang.yaml file	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/dbowie
save and commit your change	git add dev/bigbang.yaml `git status` git commit -m "updated dev/bigbang.yaml" git push OR sudo git push --set-upstream origin env/lm

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 (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 scripts **./deploy.sh** (Keep your Azure DevOps User ID and Password Handy, you might need, deployment will ask multiple times) **Note:**
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 using GitOps)

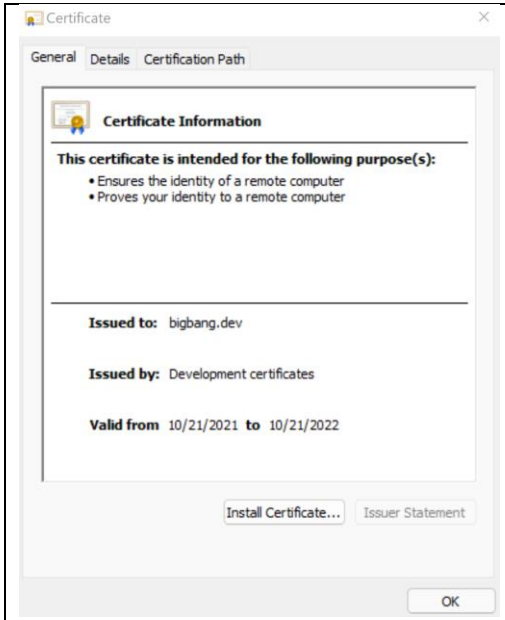
Configure local domain to IP address mapping:

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** ip=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.status.loadBalancer.ingress[0].ip}') **# get domains** domains=$(kubectl --kubeconfig rke2.kubeconfig get virtualservices -A -o jsonpath="{ .items[*].spec.hosts[*] }")

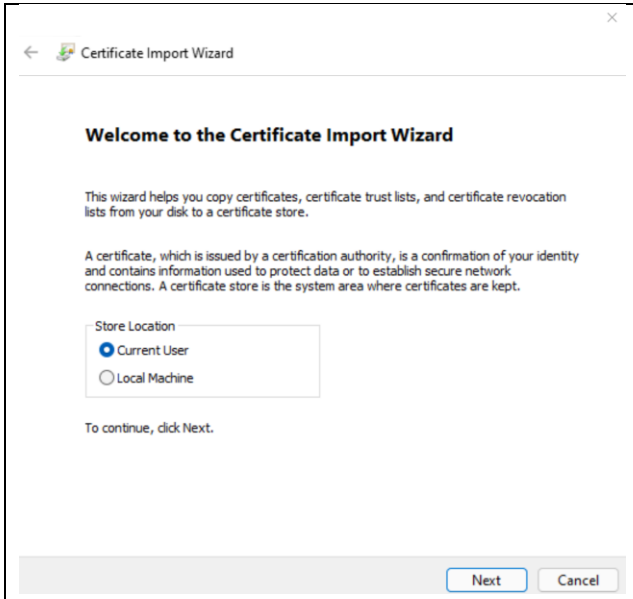
	echo $domains # add entry in /etc/hosts echo "$ip $domains" | sudo tee -a /etc/hosts
	Windows Hostfile Location **C:\Windows\System32\drivers\etc** 52.245.218.56 tracing.bigbang.dev 52.245.218.56 kiali.bigbang.dev 52.245.218.56 kibana.bigbang.dev 52.245.218.56 alertmanager.bigbang.dev 52.245.218.56 grafana.bigbang.dev 52.245.218.56 prometheus.bigbang.dev 52.245.218.56 twistlock.bigbang.dev 52.245.218.56 helloworld.bigbang.dev 52.245.218.56 currency-exchange.bigbang.dev

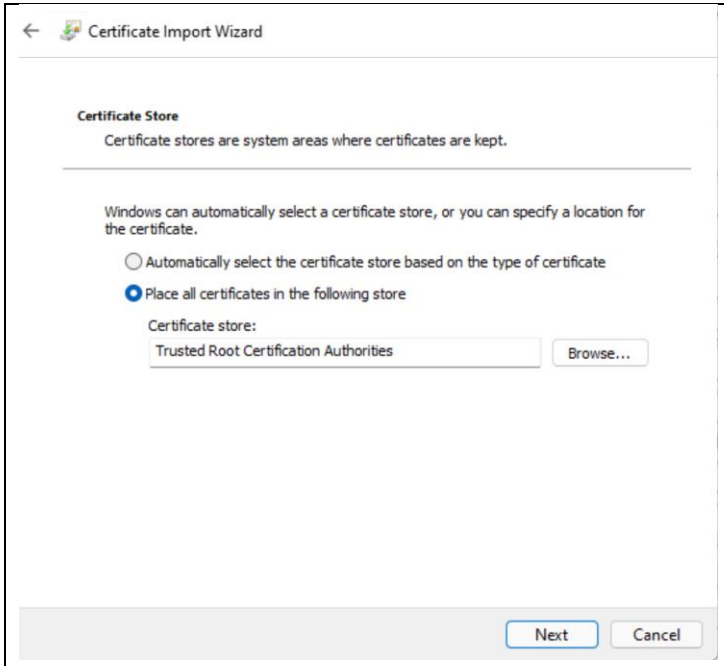
Install the Certificates:

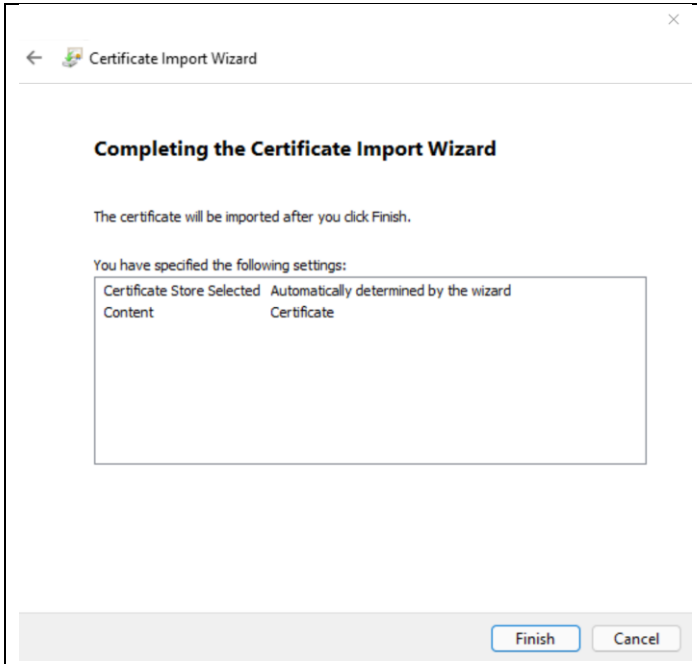
- ➔ Open the file bigbang.dev.cert & ca.cert from the location: \dsop-environment)
- ➔ Using Windows Explorer right mouse → Open →

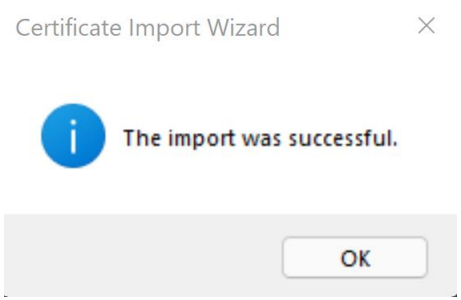


➔ Install Certificate button


Certificate Import Wizard
Welcome to the Certificate Import Wizard
This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.
A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.
Store Location
Current User
Local Machine
To continue, click Next.
Next
Cancel


Certificate Import Wizard
Certificate Store
Certificate stores are system areas where certificates are kept.
Windows can automatically select a certificate store, or you can specify a location for the certificate.
Automatically select the certificate store based on the type of certificate
Place all certificates in the following store
Certificate store:
Trusted Root Certification Authorities
Browse...
Next
Cancel





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 → 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	❯ pytest ./tests -v ========== test session starts ========== platform linux -- Python 3.8.10, pytest-6.2.4, py-1.10.0, pluggy-0.13.1 -- /mnt/c/Work/lmco/lmco/dsop-environment/venv/bin/python3 cachedir: .pytest_cache rootdir: /mnt/c/Work/lmco/lmco/dsop-environment collected 5 items tests/test_bigbang_k8s_resources.py::test_namespaces_created PASSED [20%] tests/test_bigbang_k8s_resources.py::test_successful_pod_status PASSED [40%] tests/test_bigbang_k8s_resources.py::test_successful_deployment_status PASSED [60%] tests/test_bigbang_k8s_resources.py::test_virtual_services_deployed PASSED [80%] tests/test_bigbang_services.py::test_services_are_reachable PASSED [100%]