

# SDS Platform

## Description

SDS or "**Solid Distributed Systems**" is a platform for end-to-end infrastructure automation. Initially we deploy the shared infrastructure Kubernetes cluster and the controller instances/VMs, either in public cloud (AWS, Azure or GCP) or on bare metal with any virtualization platform. From this core elements we deploy further Kubernetes clusters, cloud resources or ci/cd pipelines. Technologies used: Terragrunt with Terraform for public cloud resources, ArgoCD or FluxCD for Kubernetes plane configuration and Ansible for basic hosts configuration.

## Initial SDS platform bootstrap

### Create the controller instances/VMs - step 1

Deploy 2 VMs or instances with latest Ubuntu Server LTS version with the following requirements:

- access to internet to pull infrastructure code, docker images and OS patches (HTTPS - port 443)
  - same user that will be used on all hosts/VMs or instances
  - SUDO access for the user above with no password (see step 4.c bellow)
  - Network access to all nodes and to entire infrastructure (ssh and https)
  - 4vCPUs, 100 GB of fast storage and at least 8 GB of RAM
  - AWS, Azure or GPC permissions to deploy needed infrastructure
  - highly secured as this will contain all secrets, docker images and other critical platform data
- This hosts will be called "PLATFORM\_NAME-controller" and "PLATFORM\_NAME-controller-bkp" and will create/manage the entire platform infrastructure.

This hosts will run ansible if the platform is hosted on bare-metal servers.

This hosts will run core infrastructure elements:

- NGINX LB for services running on itself
- Hashicorp Vault
- Docker registry
- HAPROXY with Keepalived for the Kubernetes control plane

### Create the Kubernetes VMs for on-prem - step 2

**If on-prem hosting is used** you need 2 bare metal hosts with at least 18 CPU cores, 64 GB and 2TB storage each.

On these bare metal nodes you must setup a virtualization platform and the following VMs with latest Ubuntu Server LTS version as OS:

- 1 VMs on each bare metal host as Kubernetes control plane. hosts Each VM with 2vCPU, 50 GB of fast storage and 4GB RAM
- 2 VMs on each bare metal host as Kubernetes worker nodes. Each VM with 8vCPUs, 200 GB of storage and 16GB RAM
- Install on all 6 VMs latest Ubuntu Server LTS version and configure a platform wide used username (check step 4 bellow)

### Configure DNS domain - step 3

- Configure public DNS domain that will contain all hosts and services for the platform
- DNS service used must be updated programmatically from controller hosts and/or from external-dns service running inside Kubernetes cluster
- Manual updates for DNS service are not supported, all DNS updates must be automatic

### Bootstrap your initial SDS infrastructure via the controller hosts - step 4

4.a - After installing latest Ubuntu Server LTS on both controller VMs, connect with PLATFORM\_USERNAME and password to the first controller host

and generate platform wide ssh key:

```
ssh-keygen -o -a 100 -t ed25519 -f ~/.ssh/id_rsa -C "USER_EMAIL_ADDRESS"
```

4.b - Add the public SSH key to the SDS github account and confirm with SDS that infra code is prepared

4.c - Configure sudo access with NOPASSW on all controller hosts and Kubernetes nodes:

Scroll down till the end of the /etc/sudoers file and append the mentioned below line:

```
PLATFORM_USERNAME ALL=(ALL) NOPASSWD:ALL
```

4.d - Add platform wide SSH key to the second controller and to all Kubernetes cluster nodes:

```
ssh-copy-id PLATFORM_USERNAME@CONTROLLER_BKP_HOST
```

Also make sure you have to correct ~/.ssh/config and ~/.ssh/id\_rsa on both controllers  
git global configuration should also be configured correctly

4.e - On both controller hosts, in platform user home folder, clone the bootstrap repository:

```
git clone git@github.com:cbanciu667/sds-platform-bootstrap.git && cd sds-platform-bootstrap
```

4.f – Fill out the params file required for bootstrap according to example

4.f - Run ./bootstrap.sh

Alternatively you may run the commands blocks from the script one after the other to monitor each action.

Script also contains detailed explanations for each block.

4.g - Initialize and test the vault service

- Run docker-compose exec vault-server bash
- Run vault operator init and save vault init information in a safe secret store
- Run vault operator unseal UNSEAL\_KEY1..3 (use 3 of the init tokens)
- Run vault login ROOT\_TOKEN
- Run vault audit enable file file\_path=/vault/logs/audit.log
- Put test secret vault kv put secret/foo bar=precious
- Get it with vault kv get secret/foo

4.h - Optionally, install VPN clients on both controllers to allow secure connectivity from SDS HQ

5 – Acquire connectivity to the Kubernetes cluster according to platform and use the GitOps repositories to further deploy infrastructure and ci/cd pipelines. Flow: sds-kubernetes cluster (ArgoCD/FluxCD) >> Tekton ci/cd pipelines >> Terragrunt to deploy further infrastructure

#### SOME REFERENCES:

<https://technekey.com/kubespray-advanced-configuration-for-a-production-cluster/>

[https://schoolofdevops.github.io/ultimate-kubernetes-bootcamp/cluster\\_setup\\_kubespray/](https://schoolofdevops.github.io/ultimate-kubernetes-bootcamp/cluster_setup_kubespray/)

<https://blog.devops.dev/multi-node-kubernetes-cluster-deployment-with-kubespray-and-ansible-c83c2c3c8f7f>

<https://github.com/gruntwork-io/terragrunt-infrastructure-live-example>