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 hosts or cluster - 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)

If public cloud is used we will deploy the required infrastructure on step 5, after finishing 3 and 4.

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

Controller hosts - step 4

4.a - After installing latest Ubuntu Server LTS on both 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 corectly
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 - Run cd sds-platform-bootstrap && ./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

From now on, ALL operations, manual or automatic, will be performed only on the controller instances/VMs!

Kubernetes cluster - step 5

Option 1 - ON PREM with BARE METAL HOSTING
5.a - If on-prem hosting is used run on the controller in home folder:
cd kubespray
5.b - Update the ansible code and create the Kubernetes cluster with Kubespray

5.c -

Option 2 - PUBLIC CLOUD HOSTING

5.c - Confirm with SDS that Terragrunt code is updated and create EKS/AKS or GKE cluster by running in home folder:

```
cd sds-cloud && deploy.sh
```

5.d - Update kubeconfig file on both controller hosts

5.e - Bootstrap Kubernetes cluster with GitOps by running on controller in home folder:

```
cd sds-kubernetes && init.sh
```

REFERENCES:

https://technekey.com/kubespray-advanced-configuration-for-a-production-cluster/
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
https://github.com/hifis-net/ansible-role-keepalived
https://github.com/hifis-net/ansible-role-haproxy