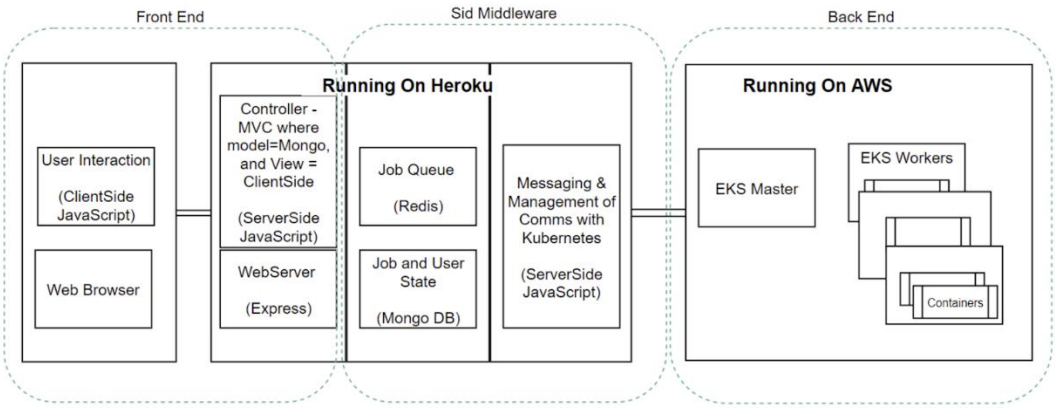
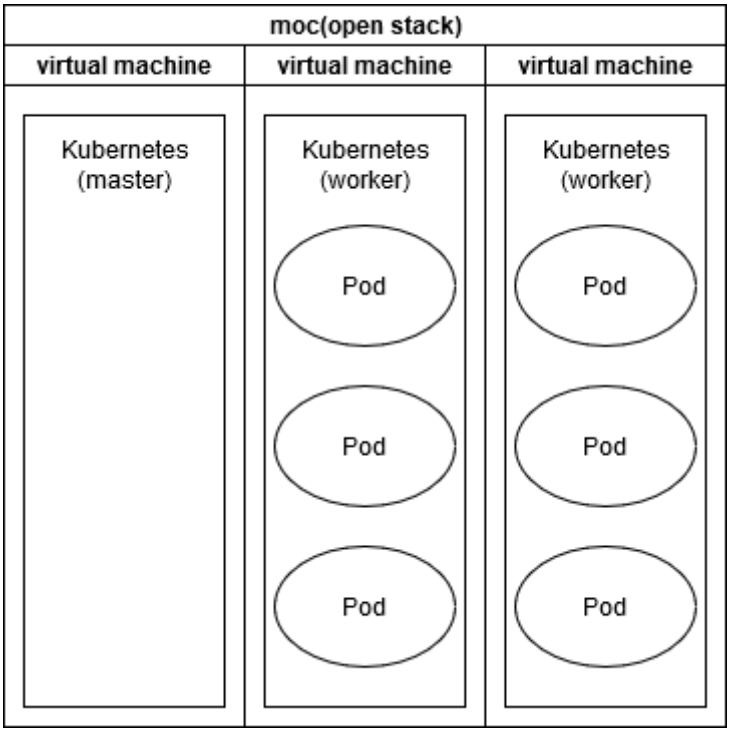
Create-Kubernetes-based-demand-research

Overview

Architecture:



We deployed all front end and middleware on openshift and backend on openstack.



This figure showed how sid will work on moc, sid will be deployed as an application in kubernetes' pod.

This documentation includes the tech we used in the project. including:

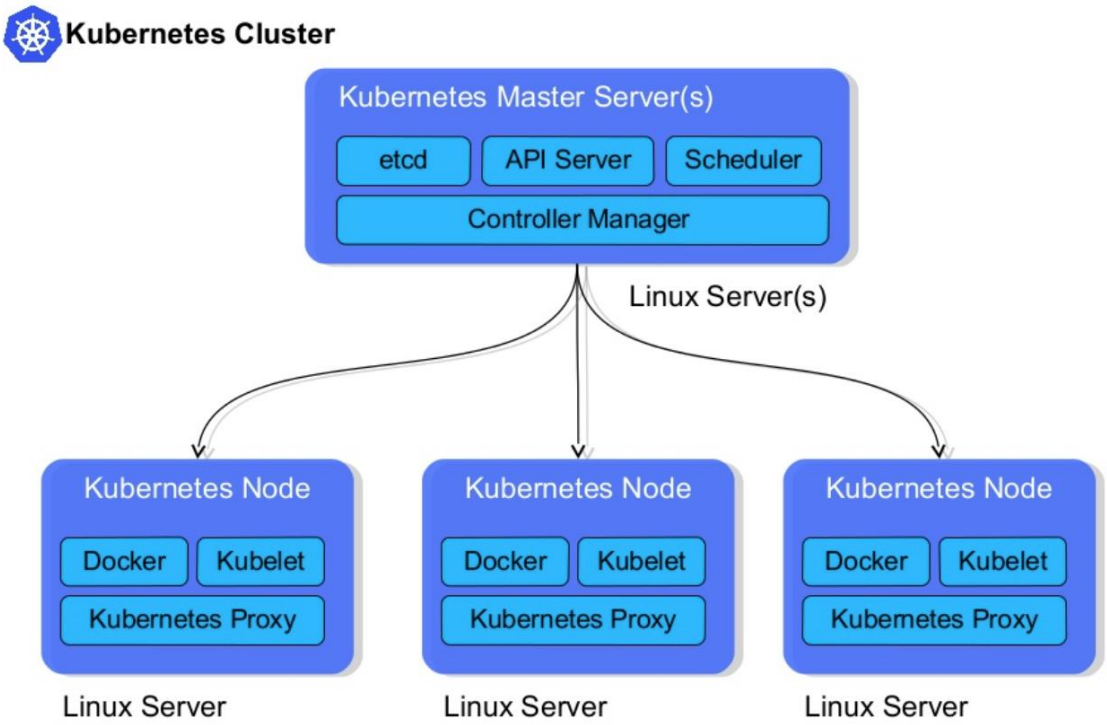
1. Kubernetes
2. Ansible
3. Kubespray
4. OpenStack
5. Terraform

1. Kubernetes

1.1. Overview

We use Kubernetes to manage all the containers on which sid can work.
Kubernetes is an open-source system that allows organizations to deploy and manage containerized applications like platforms as a service (PaaS), batch processing workers, and microservices in the cloud at scale. Through an abstraction layer created on top of a group of hosts, development teams can let Kubernetes manage a host of functions--including load balancing, monitoring and controlling resource consumption by team or application, limiting resource consumption and leveraging additional resources from new hosts added to a cluster, and other workflows.

1.2. Kubernetes Architecture (Master, Worker)



The Kubernetes master is responsible for maintaining the desired state for your cluster. The master can also be replicated for availability and redundancy. When you interact with Kubernetes, eg. via the kubectl command-line interface, you're communicating with the master. The worker nodes in a cluster are the machines (VMs, physical servers, etc) that run your applications and cloud workflows. The master controls each node; you'll rarely interact

with nodes directly.

2. Terraform

2.1 Overview

We use terraform to automatically create new virtual machine on openstack.
Terraform is an open-source infrastructure as code software tool created by HashiCorp. It enables users to define and provision a datacenter infrastructure using a high-level configuration language known as Hashicorp Configuration Language (HCL), or optionally JSON.

2.2 Deploying a compute cluster in OpenStack via Terraform

a) Log in to the OpenStack dashboard, create Application Credentials and save clouds.yaml.(we need to do it because terraform only support Single sign-on while we have no id.)
b) Create a providers.tf file with the following contents

```
provider "openstack" {
  cloud = "openstack"
}
```

c) Create a key pair in openstack, save the key.pem file.
you can find the public key by running the command

```
$ ssh-keygen -y -f /path/to/key.pem
```

d) Create a new file called main.tf with the following structure. In public_key, write the complete value of your public key.

```
resource "openstack_compute_keypair_v2" "my-cloud-key" {
  name       = "my-key"
  public_key = "ssh-rsa AAAAB3Nz..."
}
```

e) Configure openstack instance in main.tf in this format

```
resource "openstack_compute_instance_v2" "test" {
  name            = "test-vm"
  image_name      = "denbi-centos7-j10-2e08aa4bfa33-master"
  flavor_name     = "m1.tiny"
  key_pair        = "${openstack_compute_keypair_v2.my-cloud-
key.name}"
  security_groups = ["default"]

  network {
    name = "public"
  }
}
```

d) Deploy new cluster by running these commands

```
brew install terraform
terraform init
terraform plan
terraform apply
```

3. Ansible

3.1 Overview

We use ansible to automatically deploy Kubernetes on openstack virtual machine.
Ansible is a configuration management platform that automates storage, servers, and networking. When you use Ansible to configure these components, difficult manual tasks become repeatable and less vulnerable to error.

4. Kubespray

4.1 Overview

We use kubespray to automatically deploy Kubernetes on openstack virtual machine.
Kubespray is a composition of Ansible playbooks, inventory, provisioning tools, and domain knowledge for generic OS/Kubernetes clusters configuration management tasks.

4.2 Kubernetes on Openstack with Terraform

Just follow kuberspray's github
https://github.com/kubernetes-sigs/kubespray/tree/master/contrib/terraform/openstack

a) Add ssh key

```
$ eval $(ssh-agent -s)
$ ssh-add ~/.ssh/id_rsa
```

b) Open group_vars/all/all.yml and add value for "openstack_password" and set

```
cloud_provider: openstack.
```

c) configure cluster variable

Open sample-inventory/cluster.tfvars, and modify these variables:

```
network_name = "<network>"

external_net = "<UUID>"

subnet_cidr = "<cidr>"

floatingip_pool = "<pool>"

bastion_allowed_remote_ips = ["0.0.0.0/0"]
```

d) Deploy Kubernetes
cd back to terraform-backend-sid directory, and run "ansible-playbook --become -i inventory/$CLUSTER/hosts cluster.yml"