Deploy K8s in Openstack with Terraform

Yunqi @ eCommoms
2026-04-29

Terraform - intro

- Terraform vs. OpenTofu
- Infrastructure as code software
- Core and plugins (providers and modules)

main.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

provider "aws" {
  region = "us-west-1"
}

resource "aws_s3_bucket" "example" {
  bucket = "my-example-bucket"
}
```

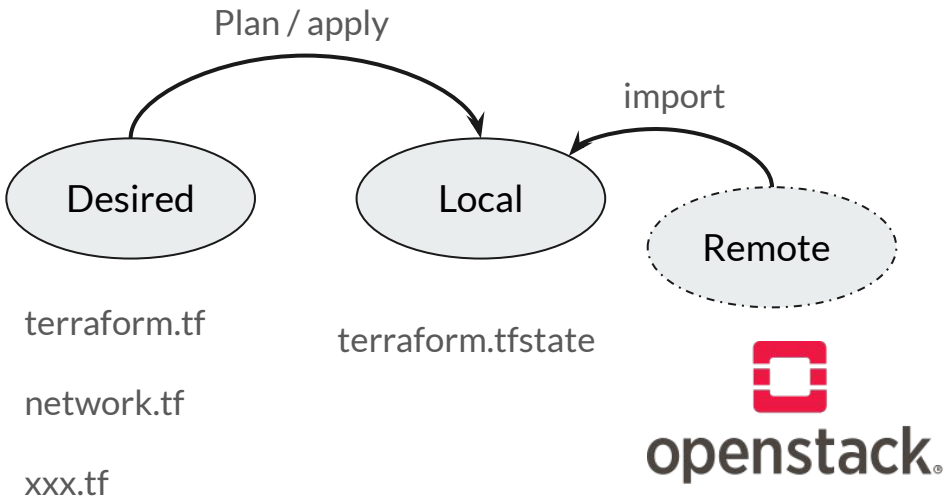
Terraform - syntax

- Pretty much JSON (can be serialized)
- Top level objects are special
  - Resources
  - Data
  - Local
  - ....
- Control flow is possible, but limited

```
data "openstack_networking_network_v2" "public" {
  name = "public1"
}

resource "openstack_networking_network_v2" "internal" {
  name = "ddls-k8s"
}

resource "openstack_networking_secgroup_v2" internal {
  name = "ddls-internal"
}

# allow any internal traffic
resource "openstack_networking_secgroup_rule_v2" "internal" {
  for_each                  = toset(["tcp", "udp", "icmp", "112"])
  security_group_id = openstack_networking_secgroup_v2.internal.id
  direction                 = "ingress"
  ethertype                = "IPv4"
  protocol                  = each.key
  remote_group_id   = openstack_networking_secgroup_v2.internal.id
}
```

Terraform - operations

- Resources are tracked with local state
- Resources can be imported
- Dependencies are tracked
- Can use data for external objects
- Demo: simple VM & network



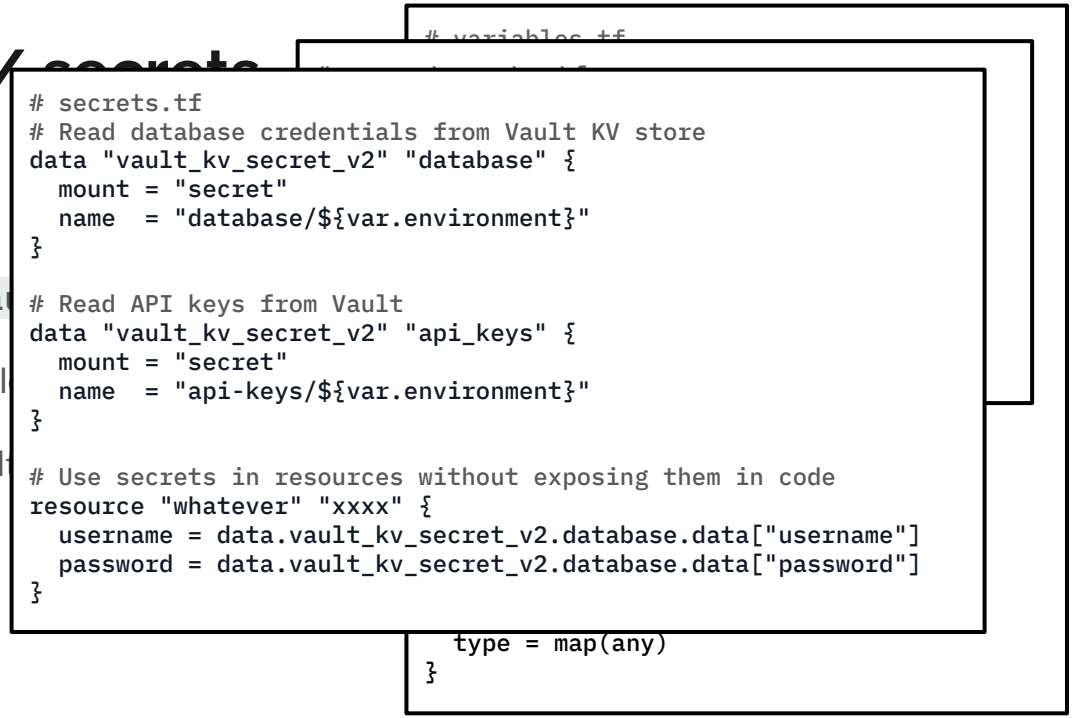
Terraform - local variables

- Iteration is somewhat limited to built-in keywords
- Usually need to implement logics as local variables
- List comprehension and string interpolation exist
- Functions exist

```
resource "ct_config" "nodes" {
  for_each = local.nodes
  snippets = each.value.snippets
}

locals {
  # ha proxy + keepalived
  keepalived_ign = [
    for i in range(var.count_control):

templatefile("butane/keepalived.yaml.tmpl", {
        prod_vip  = var.prod_vip
        test_vip  = var.test_vip
        priority  = 100 - i
      })
  ]

  nodes = merge(
    {
      for i in range(var.count_control) :
      "${var.cluster_name}-c${i + 1}" => {
        role = "control"
        ip = cidrhost(var.control_cidr, i + 1)
        snippets = [
         local.keepalived_ign[i],
         ...
```

Terraform - variables / secrets

- Variables are typed
- Reads `terraform.tfvars` and `*.au`
- Simple secret management (as variabl
- Proper secret management (data vault



Terraform: work-through with ddls cluster

Overall architecture: flat network & basic OS

-
- Services exposed as nodeports

Tasks for terraform

- Rocky 9 VMs
- Loadbalancers
- Firewall

Not covered here

- Actions (day 2 operations)
  - Trigger ansible playbooks, lambda functions, etc.
- Lifecycle options
  - Event triggers;
  - `create_before_destroy`, `prevent_destroy`;
  - `precondition`, `postcontdition`;

Extra note: cloud-init

- VM node is almost a bare-metal mode in this setup
- Initial setup still managed by cloud-init
- In openstack: config-drive or metadata server
- Keep the special routes when using metadata server

Deploy K8s in Openstack with ClusterAPI

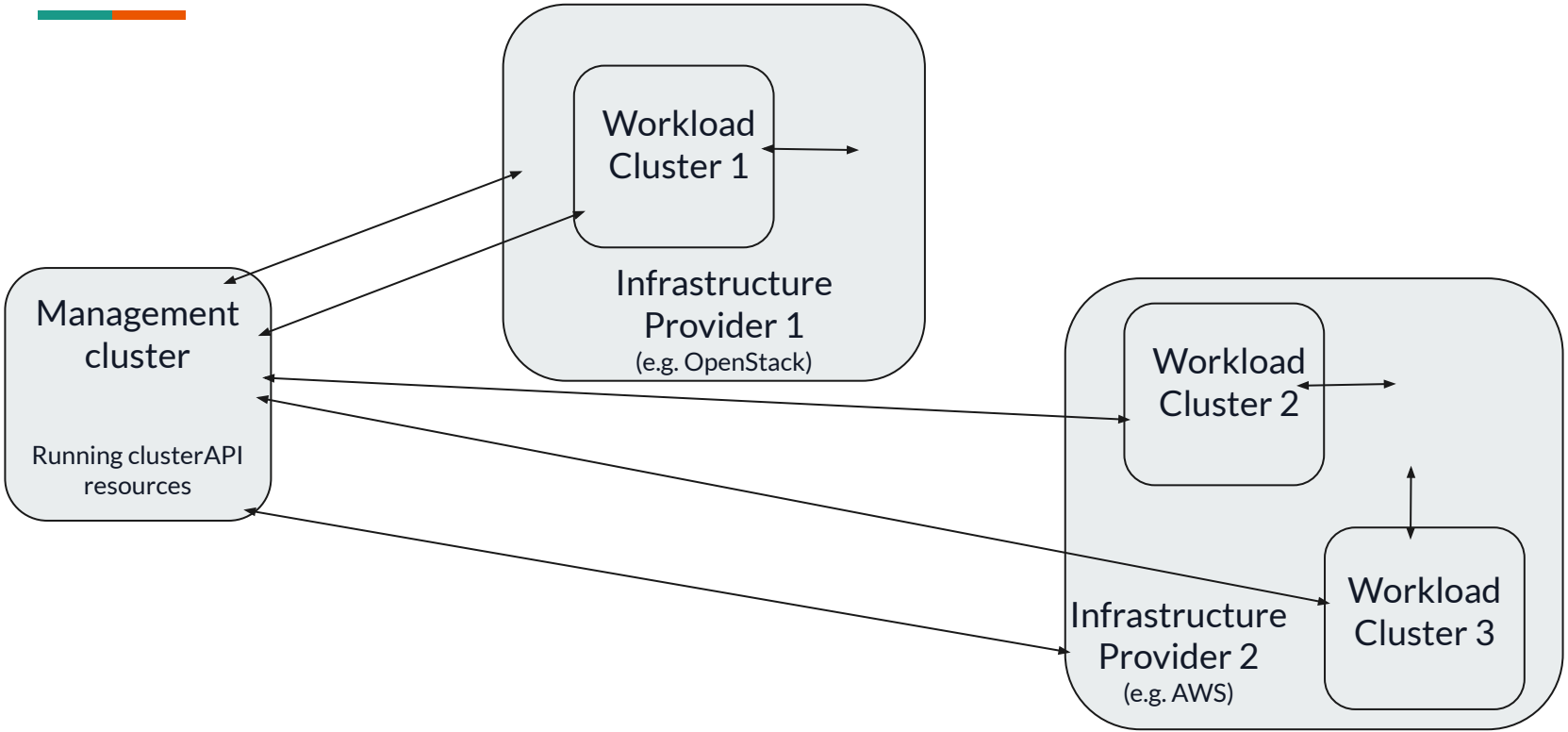
Franz Kirsten @ eCommoms
2026-04-27

What's ClusterAPI

- Cluster API is a Kubernetes sub-project focused on providing declarative APIs and tooling to simplify provisioning, upgrading, and operating multiple Kubernetes clusters.
- Check it out at https://cluster-api.sigs.k8s.io/
- Can be used to deploy & manage k8s-cluster(s) on very different infrastructure providers. Check https://cluster-api.sigs.k8s.io/user/quick-start#initialization-for-common-providers

What's ClusterAPI doing (in a nutshell)

It uses a k8s-cluster to deploy & run other k8s-clusters



What's required to have in OpenStack before you start

clusterAPI uses resources in OpenStack to run k8s

- K8s-ready images that will be used to spin up the VMs (can be created using imageBuilder)
- Loadbalancer service (amphora or OVN, we use OVN)
- External network (i.e. provider network; does not have to be public network [I think])
- Flavours; i.e. definitions of compute resource allocations for a VM
- ssh-keypair

Required config setup – for OpenStack

Setting environment variables for clusterAPI

- Access credentials – aka ‘application credentials’ (allows access via APIs) supplied as yaml
- Cloud.conf with cloud configuration; will be used by workload cluster to
  - Access the cloud services (e.g. loadbalancer)
  - Configure cloud services (e.g. loadblancer)
- Definitions of k8s-setup; e.g. k8s-version, network-IDs,... see later.

clusterctl as the CLI for ClusterAPI

It’s not super useful but you cannot go without it easily

- Sets up your seed cluster to be a management cluster
    - I.e. deploys required components and provider dependent resources
- Used to generate your workload cluster configuration
- Can help you diagnose issues with your workload cluster
    - But not really… it’s quite limited in its usefulness

Steps in brief

From no cluster to management + workload cluster

1. Create a seed cluster
2. Initiate management cluster
3. Generate your cluster configuration
   a. adjust this configuration as needed
4. Apply that config to your management cluster – `kubectl apply -f </path/to/config>`
5. Get access to the workload cluster that is being created – `clusterctl get kubeconfig <workloadClusterName>`
6. Generate secret with OpenStack access credentials in workload cluster
7. Deploy Cloud Controller OpenStack on workload cluster – to allow k8s-cluster to talk to OpenStack and create resources as needed (e.g. networks, loadbalancers…)
8. Deploy network solution; e.g. calico

Let’s do it! – bring up a new management cluster

- Create a seed cluster
  - `kind create cluster --name <seedClusterName> --kubeconfig ./seedClusterName.kubeconf`
- Initiate it as management cluster
  - `clusterctl init --infrastructure <someProvider> --kubeconfig ./seedClusterName.kubeconf`
  - For us, `someProvider = openstack`

Let’s do it! – generate cluster configuration

- Look at openststack-env.sh

```
export OPENSTACK_CLOUD=swesrc
export OPENSTACK_CONTROL_PLANE_MACHINE_FLAVOR=k8s-control
export OPENSTACK_NODE_MACHINE_FLAVOR=k8s-compute
export OPENSTACK_EXTERNAL_NETWORK_ID=0fe1e9e2-ad82-4ba2-b7ae-265e1c27ea2d
export KUBERNETES_VERSION=v1.31.4
export OPENSTACK_IMAGE_NAME=ubuntu-2204-kube-v1.31.4
export OPENSTACK_SSH_KEY_NAME=dev-k8s

export CLUSTER_NAME=seminar
export CONTROL_PLANE_MACHINE_COUNT=3
export WORKER_MACHINE_COUNT=3

export OPENSTACK_FAILURE_DOMAIN=nova
export OPENSTACK_DNS_NAMESERVERS="8.8.8.8"
```

Let's do it! – generate cluster configuration

- Look at cloud.yaml – you retreive that one from your project in OpenStack

```
clouds:
  swesrc:
    auth:
      auth_url: https://dev-cloud.swesrc.chalmers.se:5001
      application_credential_id:
      application_credential_secret:
    region_name: "RegionOne"
    interface: "public"
    identity_api_version: 3
    auth_type: "v3applicationcredential"
```

Let's do it! – generate cluster configuration

- Look at cloud.conf

```
[Global]
auth-url=https://dev-cloud.swesrc.chalmers.se:5001/v3
## lb app-credential
application-credential-id=
application-credential-secret=
region=RegionOne

#[Networking]
#internal-network-name=k8s-clusterapi-cluster-default-swesrc-k8s
#public-network-name=public1

[LoadBalancer]
#subnet-id=709bd4a0-9055-4b93-b360-bf2700d8c83a
#floating-network-id=d9455c82-001b-4593-a6bc-9dbb30d248b8
lb-provider=ovn
lb-method=SOURCE_IP_PORT
manage-security-groups=True
create-monitor=True
max-shared-lb=40
```

Let's do it! – generate cluster configuration

- Set up the environment variables needed by clusterctl
  - Create base64-encoded secrets from clouds.yaml: `source env.rc clouds.yaml <cloudName>`
  - Set OpenStack related env variables: `source openstack-env.sh`
- Generate the config
  - `clusterctl generate cluster <workloadClusterName> --infrastructure <someProvider> --kubeconfig ./seedClusterName.kubeconf > cluster.conf`
- Go through components in cluster configuration and adjust as needed

Let's do it! – cluster configuration; resources defined

```
---
apiVersion: cluster.x-k8s.io/v1beta2
kind: Cluster
metadata:
  name: swesrc-k8s-dev
  namespace: default
spec:
  clusterNetwork:
    pods:
      cidrBlocks:
      - 192.168.0.0/16
    serviceDomain: cluster.local
  controlPlaneRef:
    apiGroup: controlplane.cluster.x-k8s.io
    kind: KubeadmControlPlane
    name: swesrc-k8s-dev-control-plane
  infrastructureRef:
    apiGroup: infrastructure.cluster.x-k8s.io
    kind: OpenStackCluster
    name: swesrc-k8s-dev
```

Let's do it! – cluster configuration; resources defined

```yaml
---
apiVersion: cluster.x-k8s.io/v1beta2
kind: Cluster
metadata:
  name: swesrc-k8s-dev
  namespace: default
spec:
  clusterNetwork:
    pods:
      cidrBlocks:
      - 192.168.0.0/16
    serviceDomain: cluster.local
  controlPlaneRef:
    apiGroup: controlplane.cluster.x-k8s.io
    kind: KubeadmControlPlane
    name: swesrc-k8s-dev-control-plane
  infrastructureRef:
    apiGroup: infrastructure.cluster.x-k8s.io
    kind: OpenStackCluster
    name: swesrc-k8s-dev
```

```yaml
---
apiVersion: controlplane.cluster.x-k8s.io/v1beta2
kind: KubeadmControlPlane
metadata:
  name: swesrc-k8s-dev-control-plane
  namespace: default
spec:
  kubeadmConfigSpec:
    clusterConfiguration:
      controllerManager:
        extraArgs:
        - name: cloud-provider
          value: external
    initConfiguration:
      nodeRegistration:
        kubeletExtraArgs:
        - name: cloud-provider
          value: external
        - name: provider-id
          value:  openstack:///'{{ instance_id }}'
        - name: resolv-conf
          value: /etc/resolv.conf
        name: '{{ local_hostname }}'
    joinConfiguration:
      nodeRegistration:
        kubeletExtraArgs:
        - name: cloud-provider
          value: external
        - name: provider-id
          value:  openstack:///'{{ instance_id }}'
        - name: resolv-conf
          value: /etc/resolv.conf
        name: '{{ local_hostname }}'
  machineTemplate:
    spec:
      infrastructureRef:
        apiGroup: infrastructure.cluster.x-k8s.io
        kind: OpenStackMachineTemplate
        name: swesrc-k8s-control-plane-v1-34-6
  replicas: 3
  version: v1.34.6
```

Let's do it! – cluster configuration; resources defined

```yaml
---
apiVersion: cluster.x-k8s.io/v1beta2
kind: Cluster
metadata:
  name: swesrc-k8s-dev
  namespace: default
spec:
  clusterNetwork:
    pods:
      cidrBlocks:
      - 192.168.0.0/16
    serviceDomain: cluster.local
  controlPlaneRef:
    apiGroup: controlplane.cluster.x-k8s.io
    kind: KubeadmControlPlane
    name: swesrc-k8s-dev-control-plane
  infrastructureRef:
    apiGroup: infrastructure.cluster.x-k8s.io
    kind: OpenStackCluster
    name: swesrc-k8s-dev
```

```yaml
---
apiVersion: controlplane.cluster.x-k8s.io/v1beta2
kind: KubeadmControlPlane
metadata:
  name: swesrc-k8s-dev-control-plane
  namespace: default
spec:
  kubeadmConfigSpec:
    clusterConfiguration:
      controllerManager:
        extraArgs:
        - name: cloud-provider
          value: external
    initConfiguration:
      nodeRegistration:
        kubeletExtraArgs:
        - name: cloud-provider
          value: external
        - name: provider-id
          value:  openstack:///'{{ instance_id }}'
        - name: resolv-conf
          value: /etc/resolv.conf
        name: '{{ local_hostname }}'
    joinConfiguration:
      nodeRegistration:
        kubeletExtraArgs:
        - name: cloud-provider
          value: external
        - name: provider-id
          value:  openstack:///'{{ instance_id }}'
        - name: resolv-conf
          value: /etc/resolv.conf
        name: '{{ local_hostname }}'
  machineTemplate:
    spec:
      infrastructureRef:
        apiGroup: infrastructure.cluster.x-k8s.io
        kind: OpenStackMachineTemplate
        name: swesrc-k8s-control-plane-v1-34-6
  replicas: 3
  version: v1.34.6
```

```yaml
---
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: OpenStackCluster
metadata:
  name: swesrc-k8s-dev
  namespace: default
spec:
  apiServerLoadBalancer:
    enabled: true
    provider: ovn
  externalNetwork:
    id: 0fe1e9e2-ad82-4ba2-b7ae-265e1c27ea2d
  apiServerFloatingIP: 129.16.122.122
  identityRef:
    cloudName: swesrc
    name: swesrc-k8s-dev-cloud-config
  managedSecurityGroups:
    allNodesSecurityGroupRules:
    - description: Created by cluster-api-provider-openstac
      direction: ingress
      etherType: IPv4
      name: BGP (Calico)
      portRangeMax: 179
      portRangeMin: 179
      protocol: tcp
      remoteManagedGroups:
      - controlplane
      - worker
    - description: Created by cluster-api-provider-openstac
      direction: ingress
      etherType: IPv4
      name: IP-in-IP (calico)
      protocol: "4"
      remoteManagedGroups:
      - controlplane
      - worker
    - remoteIPPrefix: 10.7.0.0/16
      direction: ingress
      etherType: IPv4
      name: ssh
      protocol: tcp
      portRangeMax: 22
      portRangeMin: 22
      description: Allow ssh within the private network

  managedSubnets:
  - cidr: 10.7.0.0/24
    dnsNameservers:
    - 8.8.8.8
```

Let's do it! – cluster configuration; resources defined

```yaml
---
apiVersion: cluster.x-k8s.io/v1beta2
kind: Cluster
metadata:
  name: swesrc-k8s-dev
  namespace: default
spec:
  clusterNetwork:
    pods:
      cidrBlocks:
      - 192.168.0.0/16
    serviceDomain: cluster.local
  controlPlaneRef:
    apiGroup: controlplane.cluster.x-k8s.io
    kind: KubeadmControlPlane
    name: swesrc-k8s-dev-control-plane
  infrastructureRef:
    apiGroup: infrastructure.cluster.x-k8s.io
    kind: OpenStackCluster
    name: swesrc-k8s-dev
```

```yaml
---
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: OpenStackMachineTemplate
metadata:
  name: swesrc-k8s-control-plane-v1-34-6
  namespace: default
spec:
  template:
    spec:
      flavor: k8s-control
      image:
        filter:
          name: rocky9-kube-v1.34.6
      sshKeyName: dev-k8s
```

```yaml
---
apiVersion: controlplane.cluster.x-k8s.io/v1beta2
kind: KubeadmControlPlane
metadata:
  name: swesrc-k8s-dev-control-plane
  namespace: default
spec:
  kubeadmConfigSpec:
    clusterConfiguration:
      controllerManager:
        extraArgs:
        - name: cloud-provider
          value: external
    initConfiguration:
      nodeRegistration:
        kubeletExtraArgs:
        - name: cloud-provider
          value: external
        - name: provider-id
          value:  openstack:///'{{ instance_id }}'
        - name: resolv-conf
          value: /etc/resolv.conf
        name: '{{ local_hostname }}'
    joinConfiguration:
      nodeRegistration:
        kubeletExtraArgs:
        - name: cloud-provider
          value: external
        - name: provider-id
          value:  openstack:///'{{ instance_id }}'
        - name: resolv-conf
          value: /etc/resolv.conf
        name: '{{ local_hostname }}'
  machineTemplate:
    spec:
      infrastructureRef:
        apiGroup: infrastructure.cluster.x-k8s.io
        kind: OpenStackMachineTemplate
        name: swesrc-k8s-control-plane-v1-34-6
  replicas: 3
  version: v1.34.6
```

```yaml
---
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: OpenStackCluster
metadata:
  name: swesrc-k8s-dev
  namespace: default
spec:
  apiServerLoadBalancer:
    enabled: true
    provider: ovn
  externalNetwork:
    id: 0fe1e9e2-ad82-4ba2-b7ae-265e1c27ea2d
  apiServerFloatingIP: 129.16.122.122
  identityRef:
    cloudName: swesrc
    name: swesrc-k8s-dev-cloud-config
  managedSecurityGroups:
    allNodesSecurityGroupRules:
    - description: Created by cluster-api-provider-openstac
      direction: ingress
      etherType: IPv4
      name: BGP (Calico)
      portRangeMax: 179
      portRangeMin: 179
      protocol: tcp
      remoteManagedGroups:
      - controlplane
      - worker
    - description: Created by cluster-api-provider-openstac
      direction: ingress
      etherType: IPv4
      name: IP-in-IP (calico)
      protocol: "4"
      remoteManagedGroups:
      - controlplane
      - worker
    - remoteIPPrefix: 10.7.0.0/16
      direction: ingress
      etherType: IPv4
      name: ssh
      protocol: tcp
      portRangeMax: 22
      portRangeMin: 22
      description: Allow ssh within the private network

  managedSubnets:
  - cidr: 10.7.0.0/24
    dnsNameservers:
    - 8.8.8.8
```

Let's do it! – cluster configuration; resources defined

```
---
apiVersion: cluster.x-k8s.io/v1beta2
kind: MachineDeployment
metadata:
  name: swesrc-k8s-v1-33-10
  namespace: default
spec:
  clusterName: swesrc-k8s-dev
  replicas: 0
  selector:
    matchLabels: null
  template:
    spec:
      bootstrap:
        configRef:
          apiGroup: bootstrap.cluster.x-k8s.io
          kind: KubeadmConfigTemplate
          name: swesrc-k8s-dev-rocky
      clusterName: swesrc-k8s-dev
      failureDomain: nova
      infrastructureRef:
        apiGroup: infrastructure.cluster.x-k8s.io
        kind: OpenStackMachineTemplate
        name: swesrc-k8s-v1-33-10
      version: v1.33.10
```

Let's do it! – cluster configuration; resources defined

```yaml
---
apiVersion: cluster.x-k8s.io/v1beta2
kind: MachineDeployment
metadata:
  name: swesrc-k8s-v1-33-10
  namespace: default
spec:
  clusterName: swesrc-k8s-dev
  replicas: 0
  selector:
    matchLabels: null
  template:
    spec:
      bootstrap:
        configRef:
          apiGroup: bootstrap.cluster.x-k8s.io
          kind: KubeadmConfigTemplate
          name: swesrc-k8s-dev-rocky
      clusterName: swesrc-k8s-dev
      failureDomain: nova
      infrastructureRef:
        apiGroup: infrastructure.cluster.x-k8s.io
        kind: OpenStackMachineTemplate
        name: swesrc-k8s-v1-33-10
      version: v1.33.10
```

```yaml
---
apiVersion: bootstrap.cluster.x-k8s.io/v1beta2
kind: KubeadmConfigTemplate
metadata:
  name: swesrc-k8s-dev-rocky
  namespace: default
spec:
  template:
    spec:
      joinConfiguration:
        nodeRegistration:
          kubeletExtraArgs:
          - name: cloud-provider
            value: external
          - name: provider-id
            value:  openstack:///'{{ instance_id }}'
          - name: resolv-conf
            value: /etc/resolv.conf
          name: '{{ local_hostname }}'
```

Let's do it! – cluster configuration; resources defined

```yaml
---
apiVersion: cluster.x-k8s.io/v1beta2
kind: MachineDeployment
metadata:
  name: swesrc-k8s-v1-33-10
  namespace: default
spec:
  clusterName: swesrc-k8s-dev
  replicas: 0
  selector:
    matchLabels: null
  template:
    spec:
      bootstrap:
        configRef:
          apiGroup: bootstrap.cluster.x-k8s.io
          kind: KubeadmConfigTemplate
          name: swesrc-k8s-dev-rocky
      clusterName: swesrc-k8s-dev
      failureDomain: nova
      infrastructureRef:
        apiGroup: infrastructure.cluster.x-k8s.io
        kind: OpenStackMachineTemplate
        name: swesrc-k8s-v1-33-10
      version: v1.33.10
```

```yaml
---
apiVersion: bootstrap.cluster.x-k8s.io/v1beta2
kind: KubeadmConfigTemplate
metadata:
  name: swesrc-k8s-dev-rocky
  namespace: default
spec:
  template:
    spec:
      joinConfiguration:
        nodeRegistration:
          kubeletExtraArgs:
          - name: cloud-provider
            value: external
          - name: provider-id
            value:  openstack:///'{{ instance_id }}'
          - name: resolv-conf
            value: /etc/resolv.conf
          name: '{{ local_hostname }}'
```

```yaml
---
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: OpenStackMachineTemplate
metadata:
  name: swesrc-k8s-v1-33-10
  namespace: default
spec:
  template:
    spec:
      flavor: k8s-compute
      image:
        filter:
          name: rocky9-kube-v1.33.10
      sshKeyName: dev-k8s
```

Let's do it! – apply config to management cluster

I.e. create the workload cluster and get access to it

- Apply the configuration
  - `kubectl apply -f cluster.conf --kubeconfig ./seedClusterName.kubeconf`
- Get workload cluster kubeconfig
  - `clusterctl get kubeconfig <workloadClusterName> --kubeconfig ./seedClusterName.kubeconf > workloadClusterName.kubeconf`

Let’s do it! – finish the deployment

Deploy a network solution and enable the Cloud Control Manager

- Create secret enabling access to project in OpenStack
  - kubectl --kubeconfig=./workloadCluster.kubeconfig -n kube-system create secret generic cloud-config --from-file=cloud.conf
- Deploy Could Control Manager OpenStack
  - kubectl apply --kubeconfig=./workloadCluster.kubeconfig -f https://raw.githubusercontent.com/kubernetes/cloud-provider-openstack/master/manifests/controller-manager/cloud-controller-manager-roles.yaml
  - kubectl apply --kubeconfig=./workloadCluster.kubeconfig -f https://raw.githubusercontent.com/kubernetes/cloud-provider-openstack/master/manifests/controller-manager/cloud-controller-manager-role-bindings.yaml
  - kubectl apply --kubeconfig=./workloadCluster.kubeconfig -f https://raw.githubusercontent.com/kubernetes/cloud-provider-openstack/master/manifests/controller-manager/openstack-cloud-controller-manager-ds.yaml
- Deploy network for k8s-cluster
  - kubectl --kubeconfig=./capi-quickstart.kubeconfig apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/calico.yaml