ALL DAY DEVOPS

TRACK: MODERN INFRASTRUCTURE

Ruling Kubernetes on the Cloud

•••

Huseyin BABAL

Software Development Team Lead, Hazelcast Cloud

About me

- Currently Implementing Hazelcast Cloud
- Ex-Sony and Ex-eBay Engineer (DevOps & Microservice Transformation Project Architect)
- Regularly Talk & Do Workshops about all the stuffs I know on public events

Outline

- Managed Service / On-premise
- Automation Preparation
- Networking
- Resource Management
- Persistence
- Monitoring
- Access Management

Managed Service / On-premise …

On-premise

- Due to regulations, you may need to stay in datacenters
- Kubespray to install and manage kubernetes clusters
- MetaLB for Load Balancer
- OpenEBS, Rook for Storage Provision

Managed Service

- All major cloud providers have Kubernetes-As-a-Service
- Eliminate operational costs
- Built-in providers for Load Balancer and Storage
- Cloud provider manages master nodes, you focus on the worker nodes

Automation Preparation …

Why Automation?

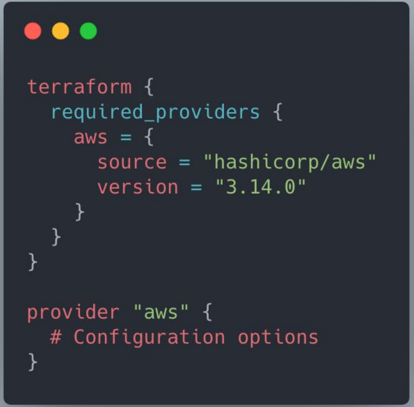
- To include k8s within a complete pipeline instead of using cloud specific UI
- Stress-free environment management
- Having the same apple in dev,stage, prod
-

Terraform

- Helps you to install components by using specific providers and their resources
- It stores the state in different kind of backends to have full control over resource creation
- It totally abstracts the way you connect cloud provider apis, you don’t need to control every call to check if it is finished or not

Providers

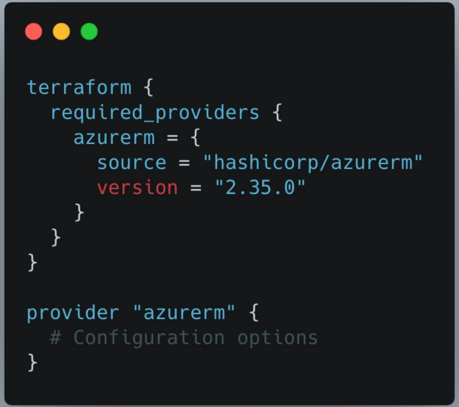
EKS



```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "3.14.0"
    }
  }
}

provider "aws" {
  # Configuration options
}
```

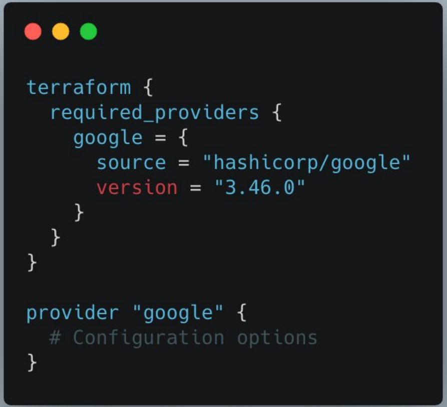
AKS



```
terraform {
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = "2.35.0"
    }
  }
}

provider "azurerm" {
  # Configuration options
}
```

GKE

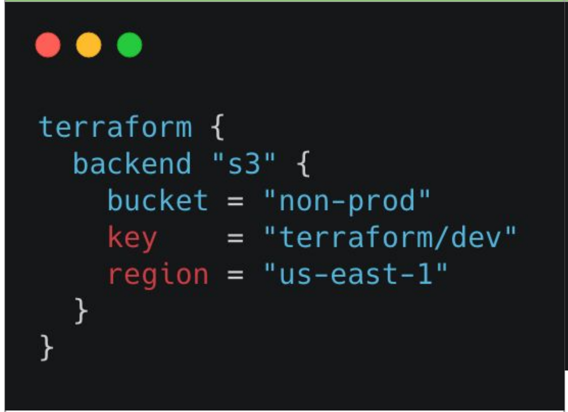


```
terraform {
  required_providers {
    google = {
      source = "hashicorp/google"
      version = "3.46.0"
    }
  }
}

provider "google" {
  # Configuration options
}
```

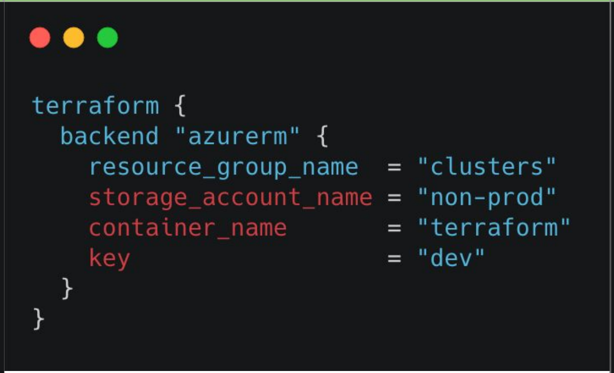
Backend Configuration

EKS



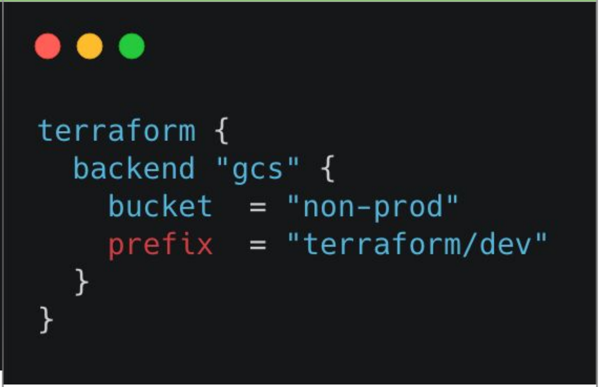
```
terraform {
  backend "s3" {
    bucket = "non-prod"
    key    = "terraform/dev"
    region = "us-east-1"
  }
}
```

AKS



```
terraform {
  backend "azurerm" {
    resource_group_name  = "clusters"
    storage_account_name = "non-prod"
    container_name       = "terraform"
    key                  = "dev"
  }
}
```

GKE



```
terraform {
  backend "gcs" {
    bucket  = "non-prod"
    prefix  = "terraform/dev"
  }
}
```

Credentials

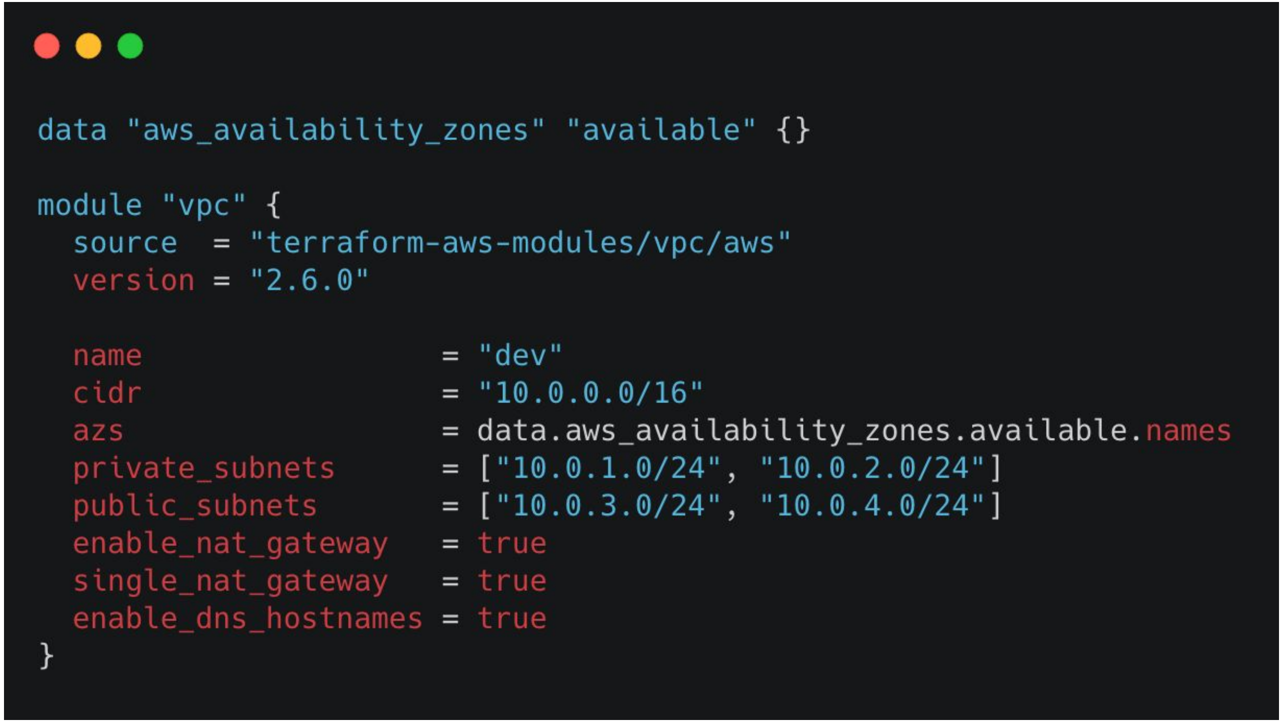
EKS	AKS	GKE
AWS_ACCESS_KEY_ID AWS_SECRET_ACCESS_KEY	ARM_TENANT_ID ARM_SUBSCRIPTION_ID ARM_CLIENT_ID ARM_CLIENT_SECRET ARM_ACCESS_KEY	GOOGLE_CREDENTIALS

Networking …

Network Topology

- Based on use-case, k8s clusters can live inside private or public VPCs
- If it is private, VPC Peering can be used to access from another VPC

EKS Networking

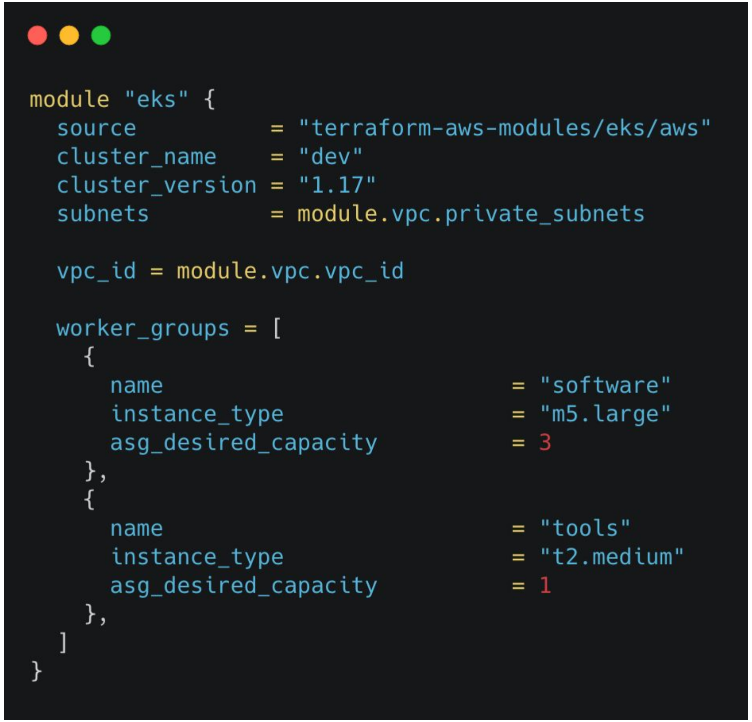


```
data "aws_availability_zones" "available" {}

module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "2.6.0"

  name                 = "dev"
  cidr                 = "10.0.0.0/16"
  azs                  = data.aws_availability_zones.available.names
  private_subnets      = ["10.0.1.0/24", "10.0.2.0/24"]
  public_subnets       = ["10.0.3.0/24", "10.0.4.0/24"]
  enable_nat_gateway   = true
  single_nat_gateway   = true
  enable_dns_hostnames = true
}
```

EKS Cluster

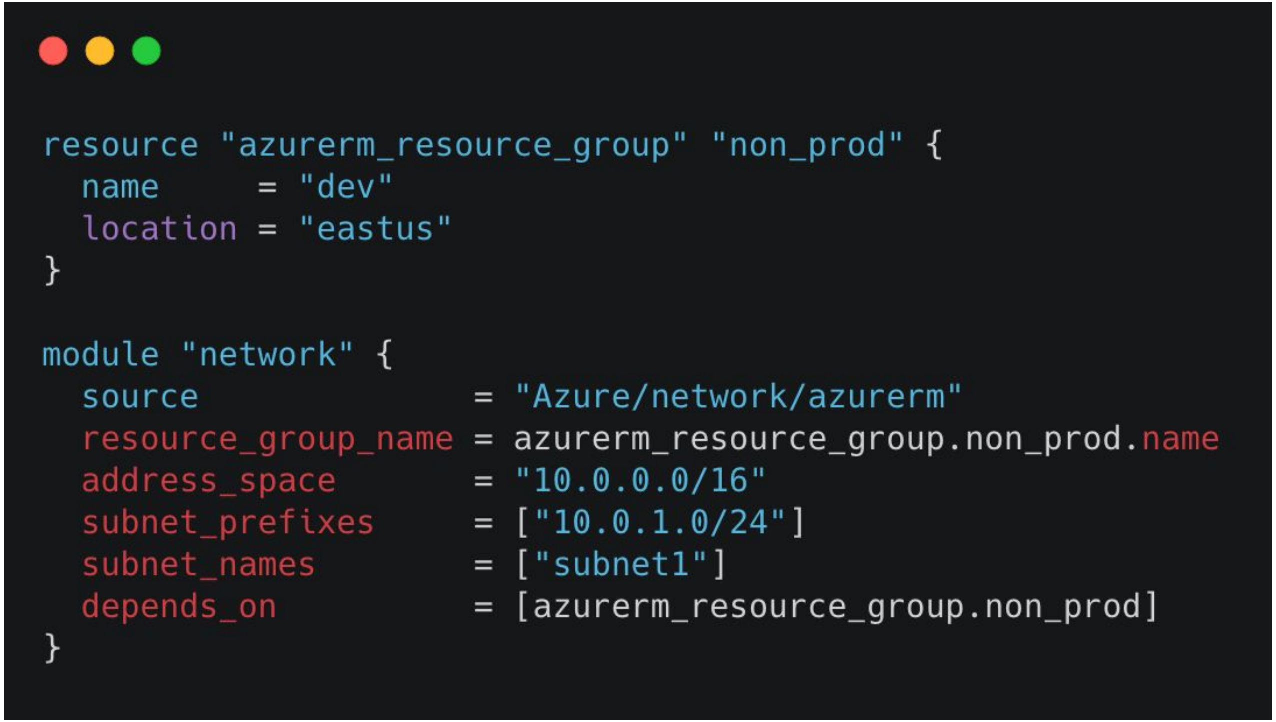


```
module "eks" {
  source          = "terraform-aws-modules/eks/aws"
  cluster_name    = "dev"
  cluster_version = "1.17"
  subnets         = module.vpc.private_subnets

  vpc_id = module.vpc.vpc_id

  worker_groups = [
    {
      name                          = "software"
      instance_type                 = "m5.large"
      asg_desired_capacity          = 3
    },
    {
      name                          = "tools"
      instance_type                 = "t2.medium"
      asg_desired_capacity          = 1
    },
  ]
}
```

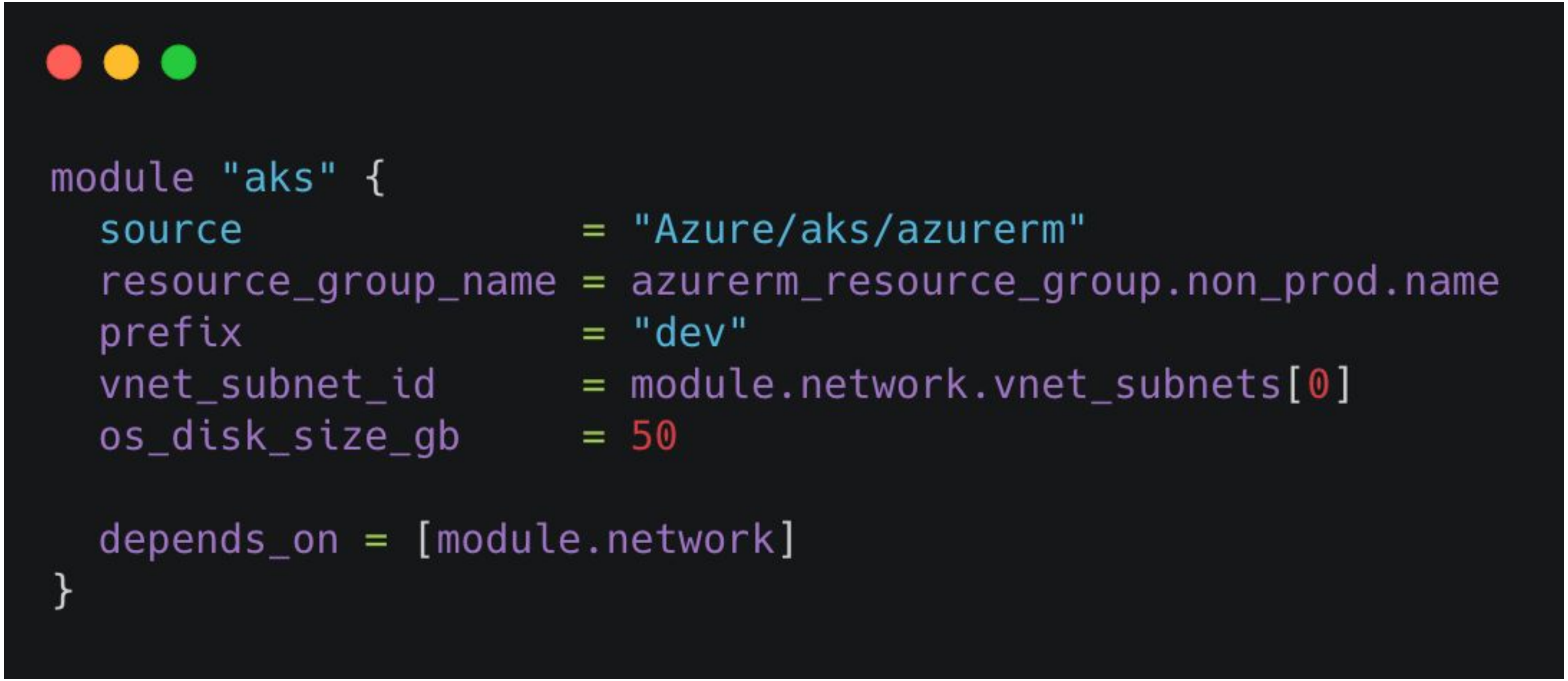
AKS Networking



```
resource "azurerm_resource_group" "non_prod" {
  name     = "dev"
  location = "eastus"
}

module "network" {
  source              = "Azure/network/azurerm"
  resource_group_name = azurerm_resource_group.non_prod.name
  address_space       = "10.0.0.0/16"
  subnet_prefixes     = ["10.0.1.0/24"]
  subnet_names        = ["subnet1"]
  depends_on          = [azurerm_resource_group.non_prod]
}
```

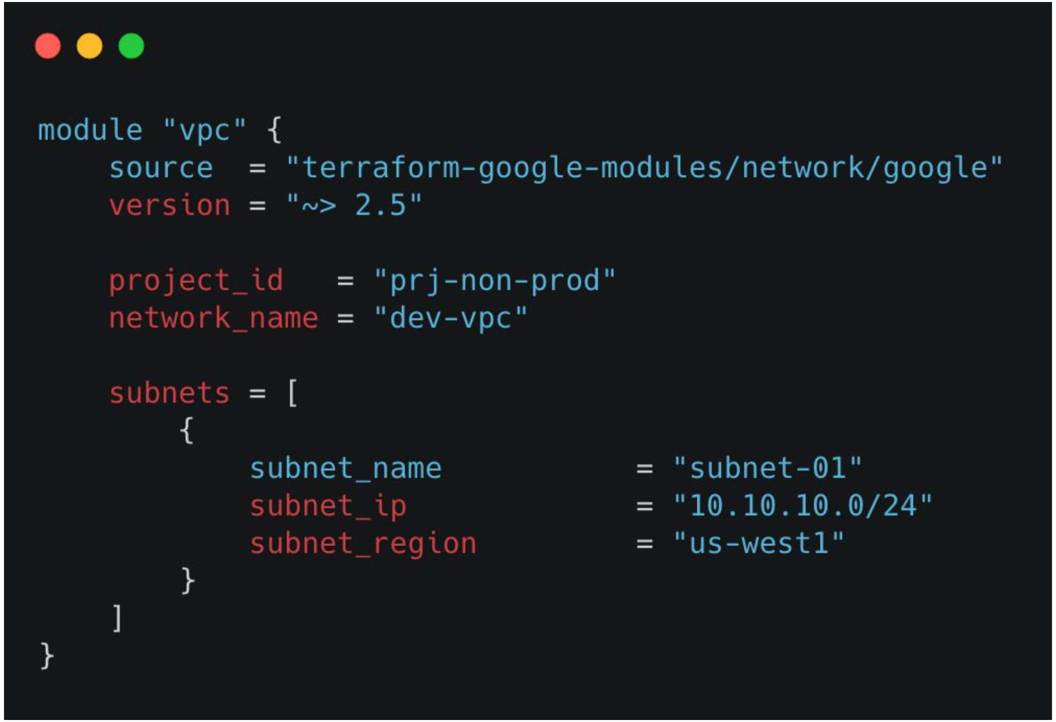
AKS Cluster



```
module "aks" {
  source              = "Azure/aks/azurerm"
  resource_group_name = azurerm_resource_group.non_prod.name
  prefix              = "dev"
  vnet_subnet_id      = module.network.vnet_subnets[0]
  os_disk_size_gb     = 50

  depends_on = [module.network]
}
```

GKE Networking

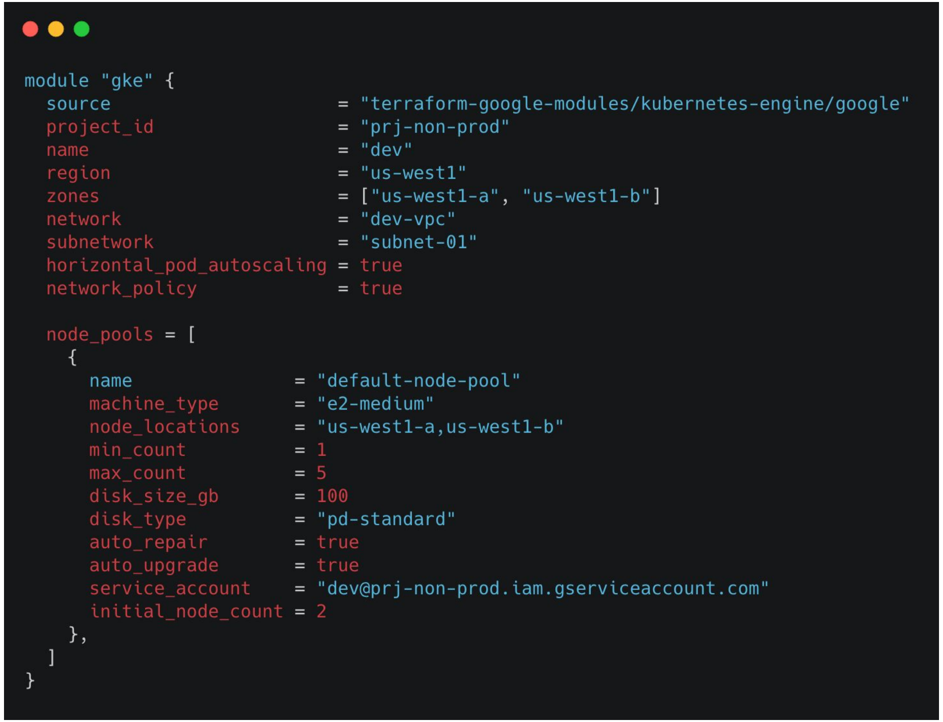


```
module "vpc" {
    source  = "terraform-google-modules/network/google"
    version = "~> 2.5"

    project_id   = "prj-non-prod"
    network_name = "dev-vpc"

    subnets = [
        {
            subnet_name           = "subnet-01"
            subnet_ip             = "10.10.10.0/24"
            subnet_region         = "us-west1"
        }
    ]
}
```

GKE Cluster



```
module "gke" {
  source                     = "terraform-google-modules/kubernetes-engine/google"
  project_id                 = "prj-non-prod"
  name                       = "dev"
  region                     = "us-west1"
  zones                      = ["us-west1-a", "us-west1-b"]
  network                    = "dev-vpc"
  subnetwork                 = "subnet-01"
  horizontal_pod_autoscaling = true
  network_policy             = true

  node_pools = [
    {
      name               = "default-node-pool"
      machine_type       = "e2-medium"
      node_locations     = "us-west1-a,us-west1-b"
      min_count          = 1
      max_count          = 5
      disk_size_gb       = 100
      disk_type          = "pd-standard"
      auto_repair        = true
      auto_upgrade       = true
      service_account    = "dev@prj-non-prod.iam.gserviceaccount.com"
      initial_node_count = 2
    },
  ]
}
```

Resource Management

•••

Know Your K8s Node Limits

When you create a cluster on EKS, AKS, or GKE they actually spin up VMs on the background and they reserve certain amount of resource of that VMs for their internal usages.

If you need set of pods 16G in total, you cannot just use 2 m5.large on the background to install on top of that.

Know Your K8s Node Limits

AWS	Azure	GCP
M5.large (8G, 2 CPU)	D2 v3 (8G 2CPU)	n1-standard-2 (7.5 G, 2 CPU)
100M Hard Eviction Threshold	750M Hard Eviction Threshold	100M Hard Eviction Threshold
7G Usable memory for Pods	5.4G Usable memory for Pods	5.6G Usable memory for Pods
700M OS+Kubelet	1.8G OS+Kubelet	1.7G OS+Kubelet

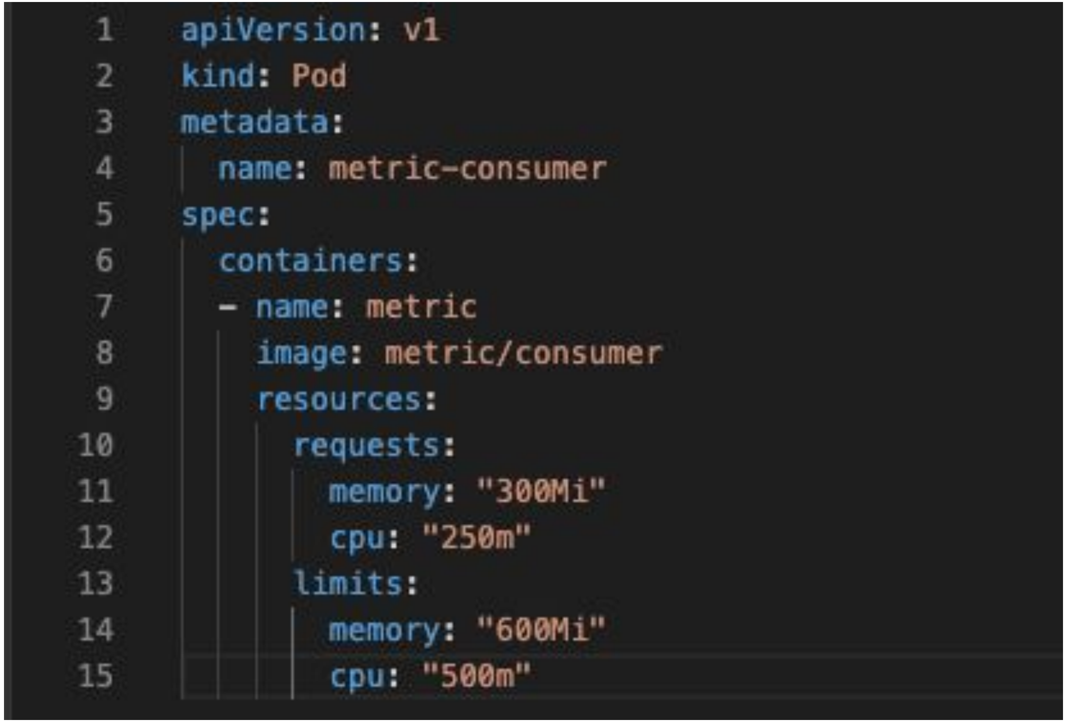
Pod Resource Management

To provide qualified service, there should be properly defined limit of an application inside kubernetes.
Applications are just workloads inside k8s and you can define requests / limits for them

Requests / Limits

Requests is for saying “How much memory / cpu needed” for this application.
Limits is for saying “Up to how much memory / cpu can be used” by this application.

Requests / Limits

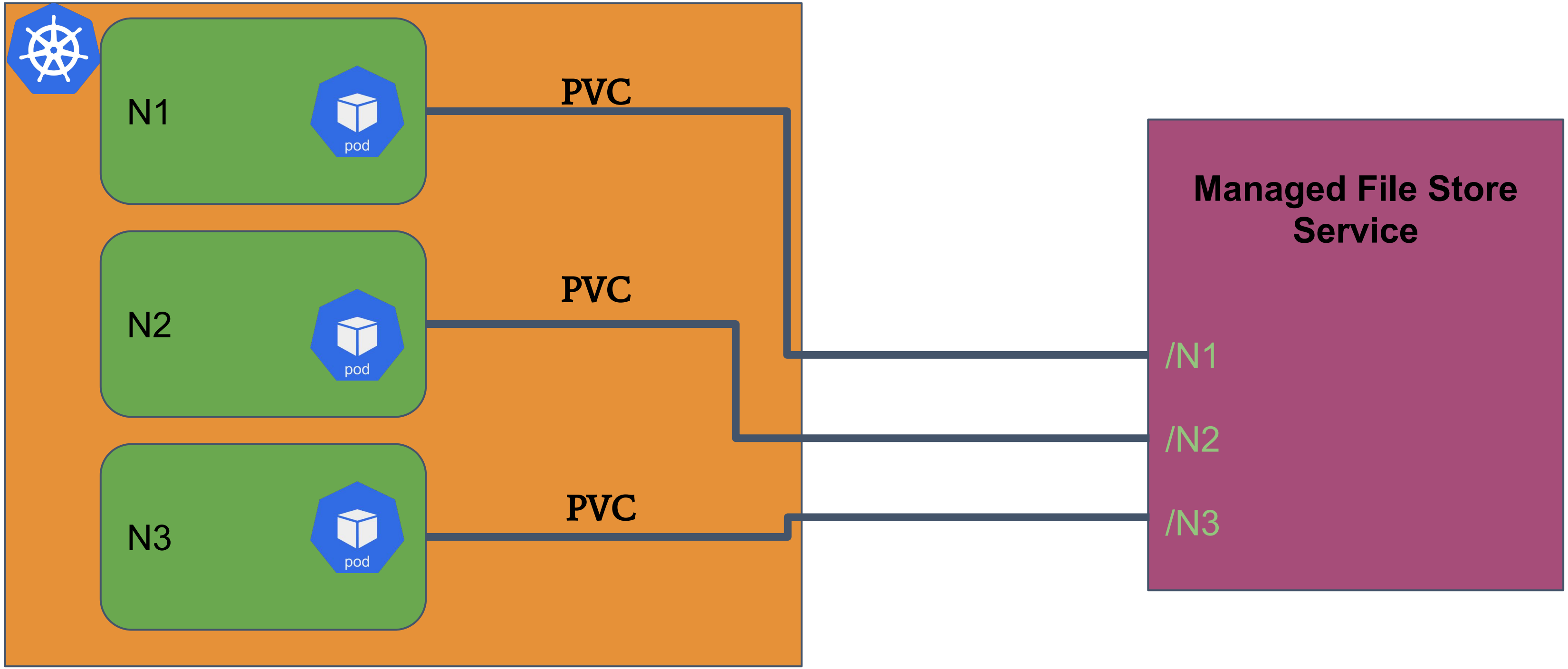


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: metric-consumer
spec:
  containers:
  - name: metric
    image: metric/consumer
    resources:
      requests:
        memory: "300Mi"
        cpu: "250m"
      limits:
        memory: "600Mi"
        cpu: "500m"
```

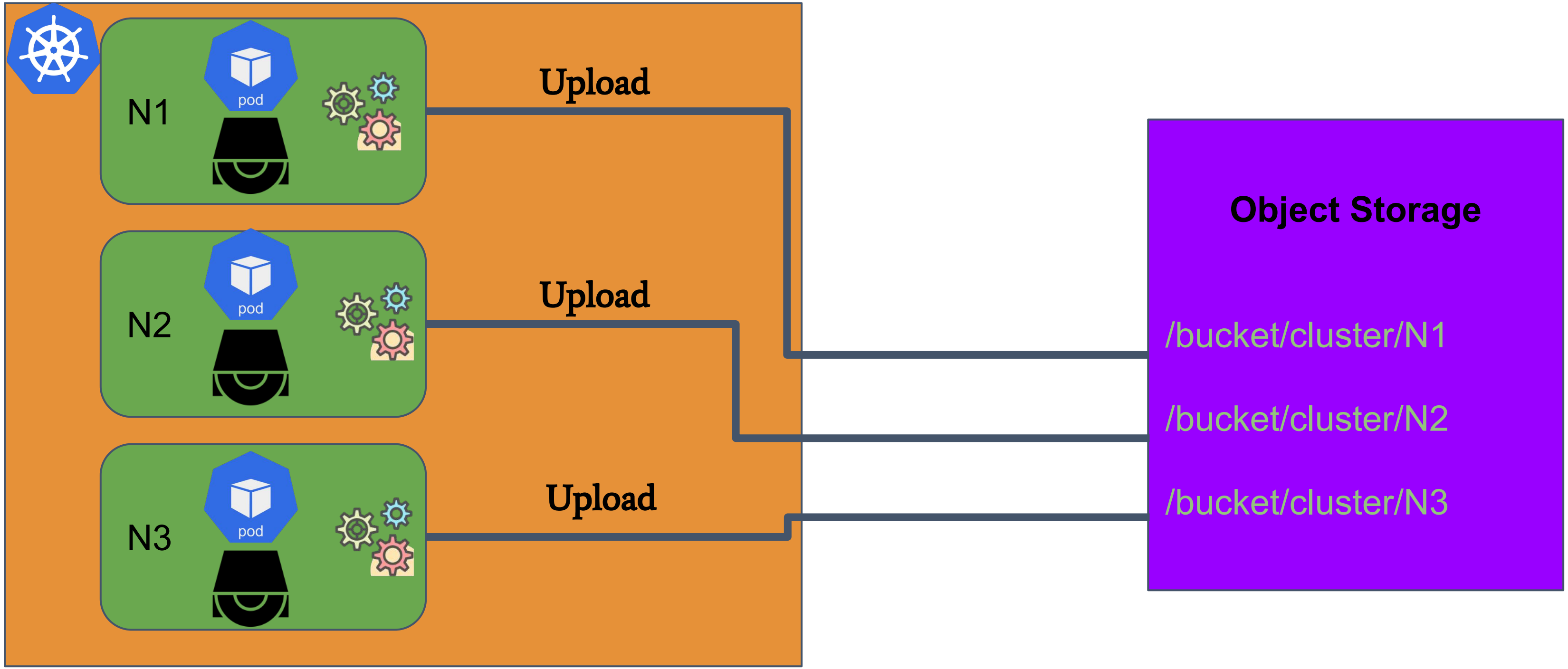
Persistence

…

With Managed File Store Service



Without Managed File Store Service



Motivation of Persistence

- Taking Snapshot data at time T
- Restore from snapshot in for disaster recovery
- Clone a technology (lives in pods) by creating new cluster and provide snapshot data during startup

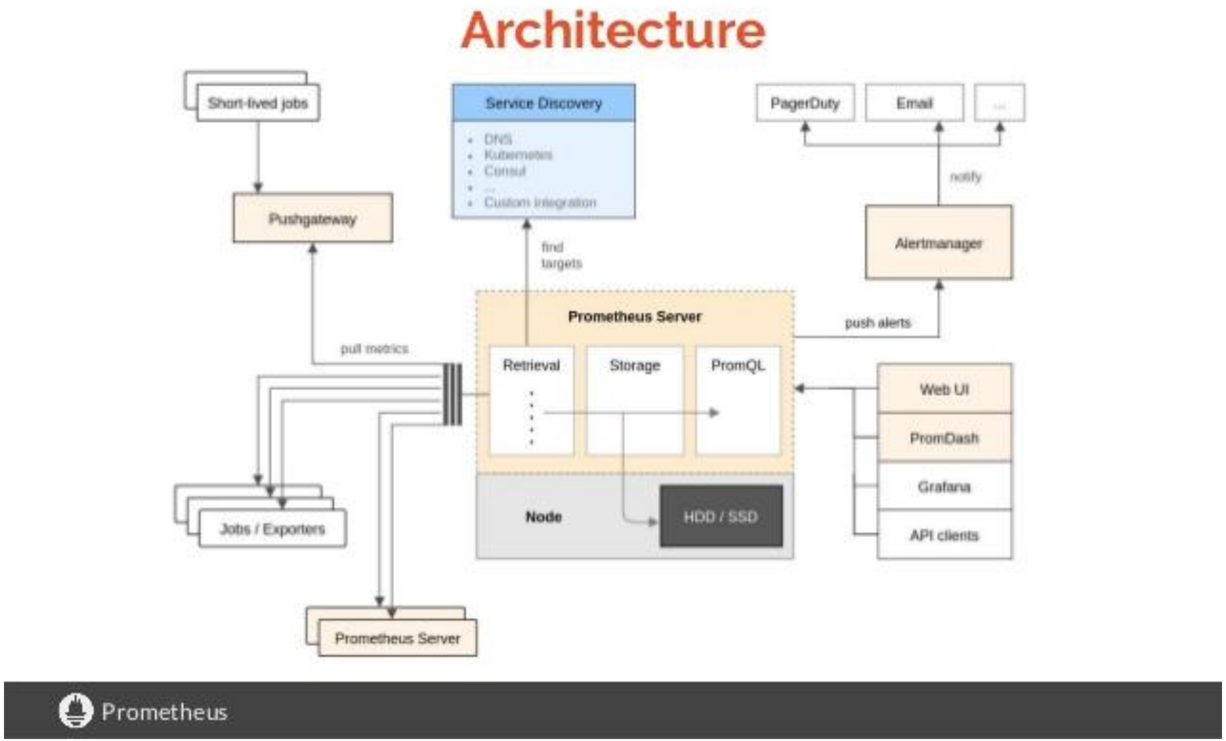
Persistence Components

- There is a daemonset to have a uploader agent on every node to upload data for snapshot operations
- There is a daemonset to have a downloader agent on every node to download data in advance for restore or clone operations
- Since those agents within same node with actual technology you provide, they must be as tiny as possible
- Best practice to use workload identity for agents to be able to access object storage without any kind of credentials

Monitoring

●●●

Prometheus

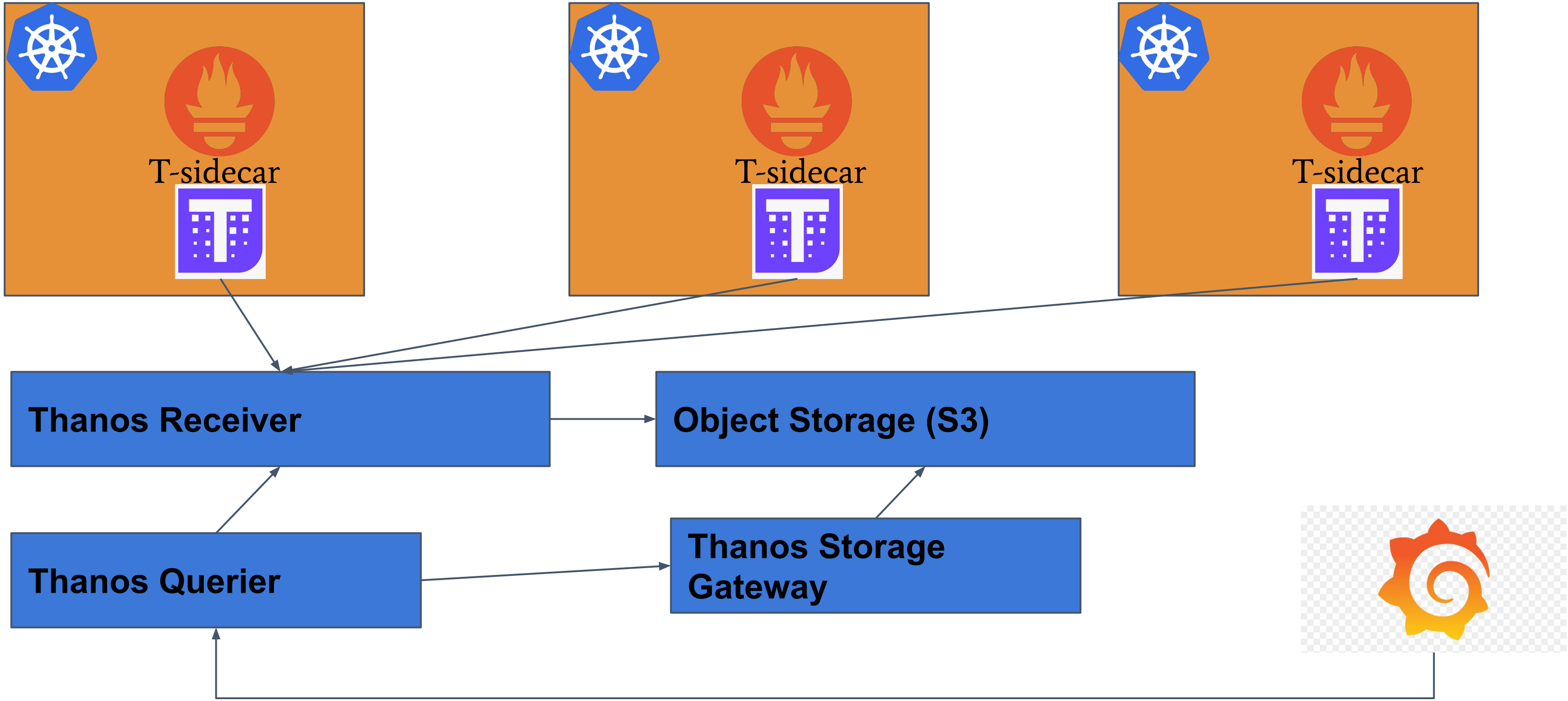


*https://www.slideshare.net/QAware/cloud-monitoring-with-prometheus

Monitoring

- Prometheus for monitoring to collect metrics from targets
- Implement your own metric exporter to be scraped by Prometheus
- Define Prometheus Rules to let AlertManager to send notifications
- Introduce a central monitoring system to handle metrics coming from different clusters in one place
- Use Thanos to have scalable monitoring system

Monitoring Multi-Cluster



Alert Rules

Through Thanos Querier, you can get built-in metrics and custom ones. By using those metrics, you can also trigger alerts like;

- If used_memory > 80 then fire alarm to notify customer
- If used_memory < 40 then fire alarm to scale down
- If used_memory > 90 then fire alarm to scale up

Inside prometheus operator, you can find custom resource PrometheusRule

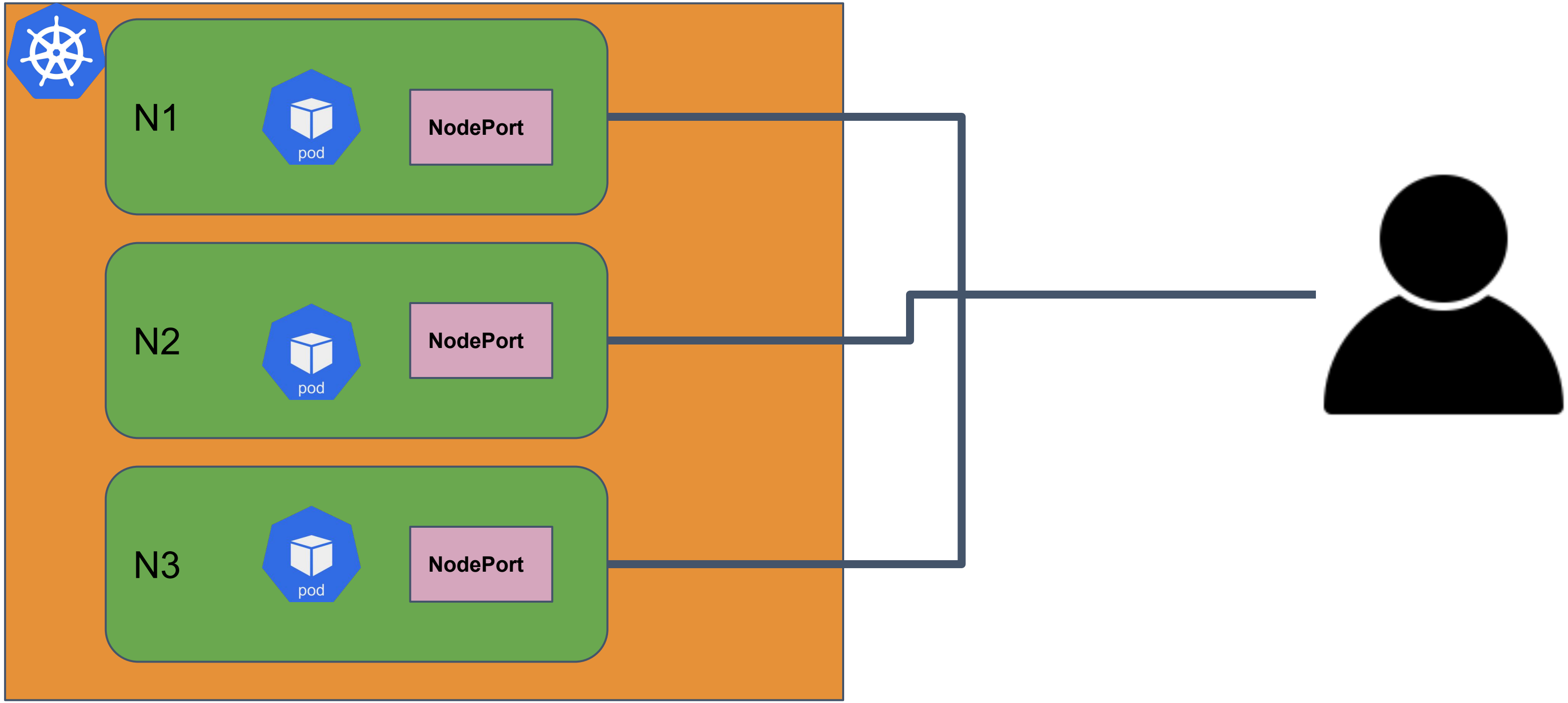
Access Management

...

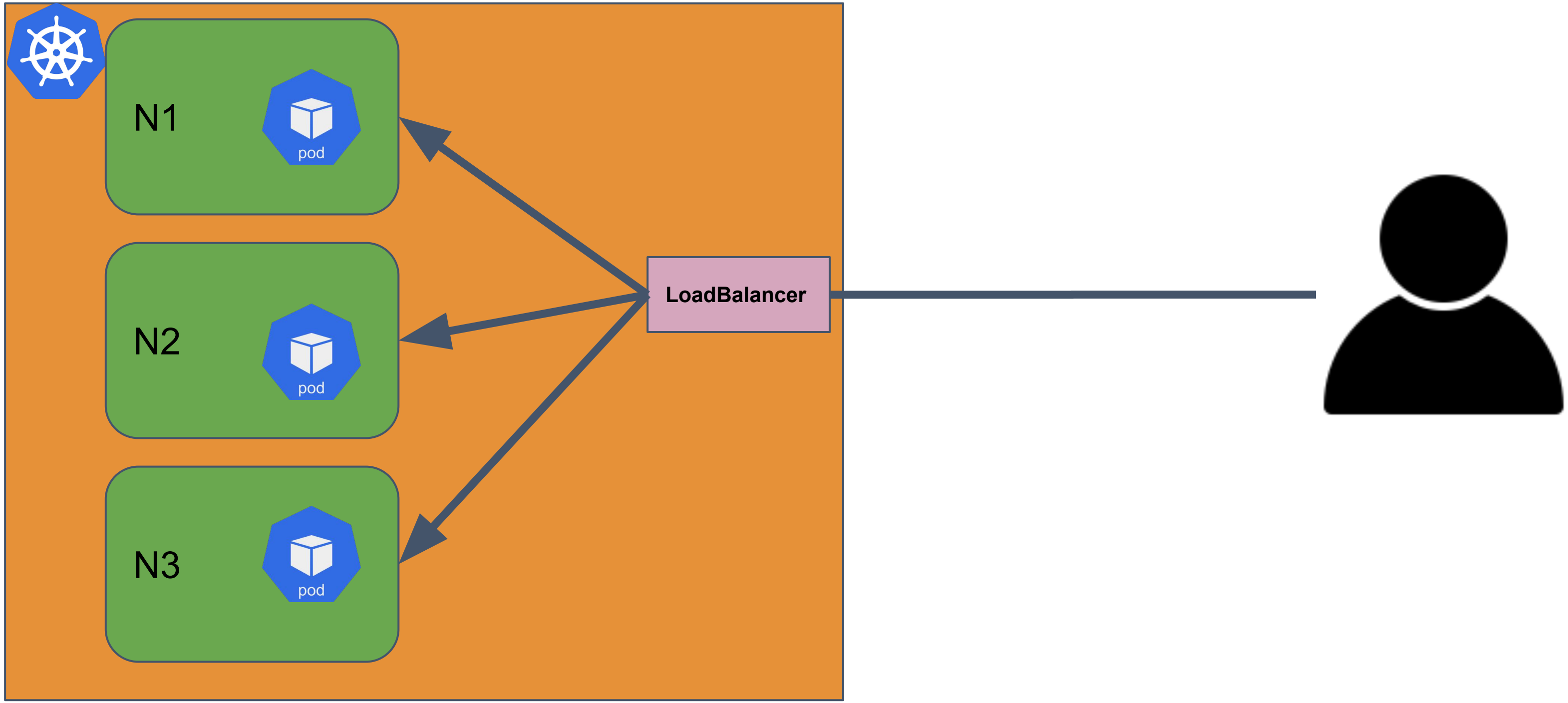
Network Topologies

According to business needs, you may want to setup cluster in a private or public network.
On all cloud providers, they provide network topology type to put k8s in desired network type.
Accessing public cluster is ok, but private one is a bit challenging

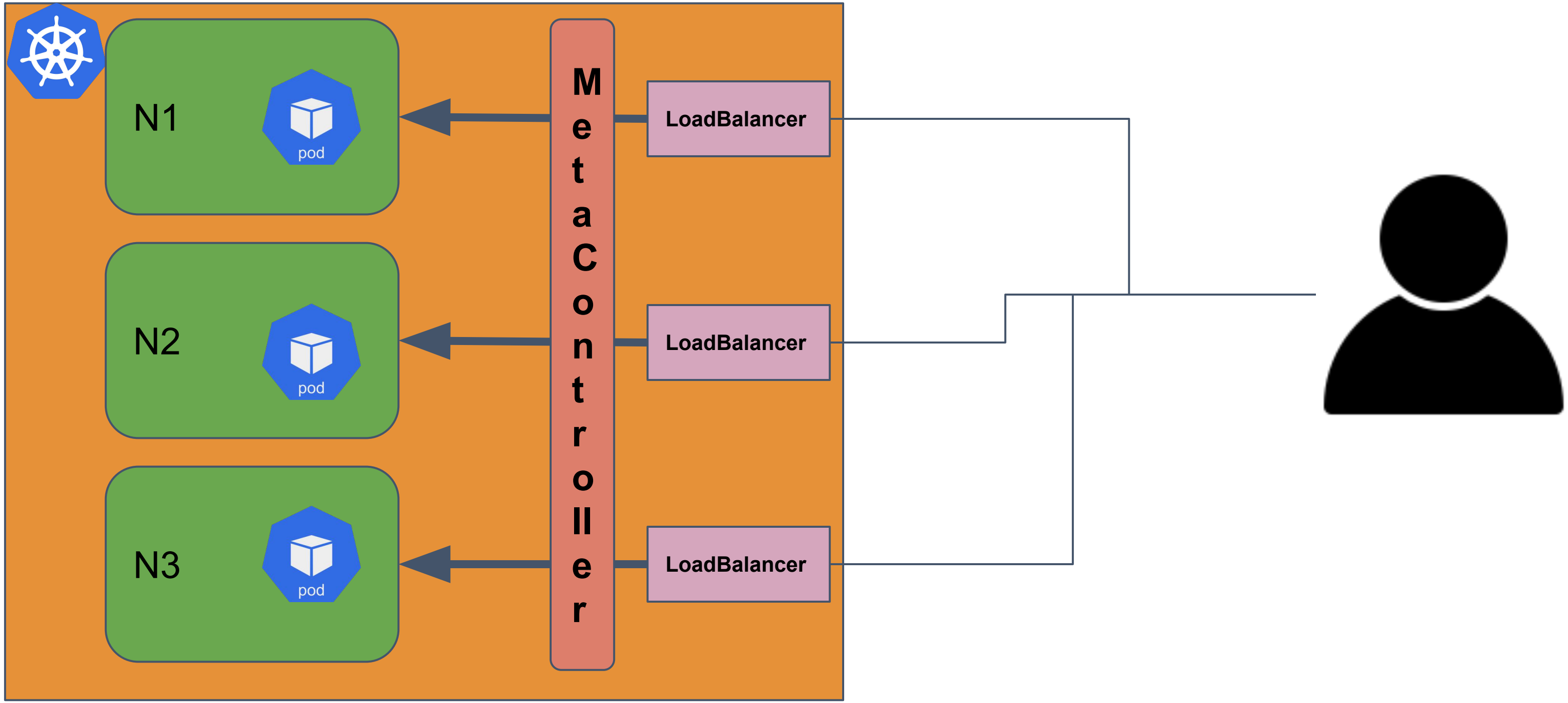
Service Exposal with NodePort



Service Exposal with LoadBalancer



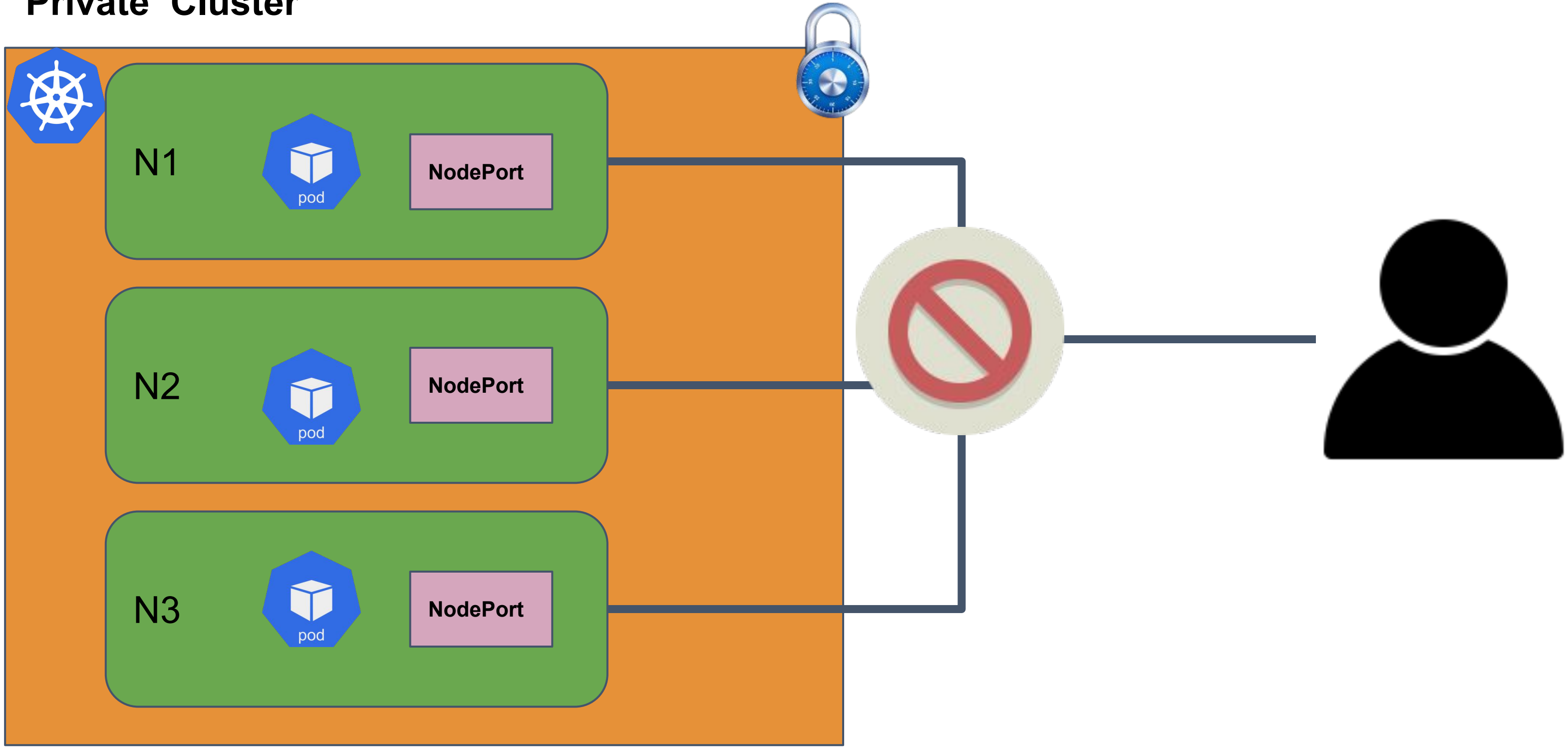
Service Exposal with NodePort as LoadBalancer



Private Clusters

It is easy to connect to cluster which is inside k8s cluster that has public network topology.
What about private ones?

Private Cluster



Peering Scenario

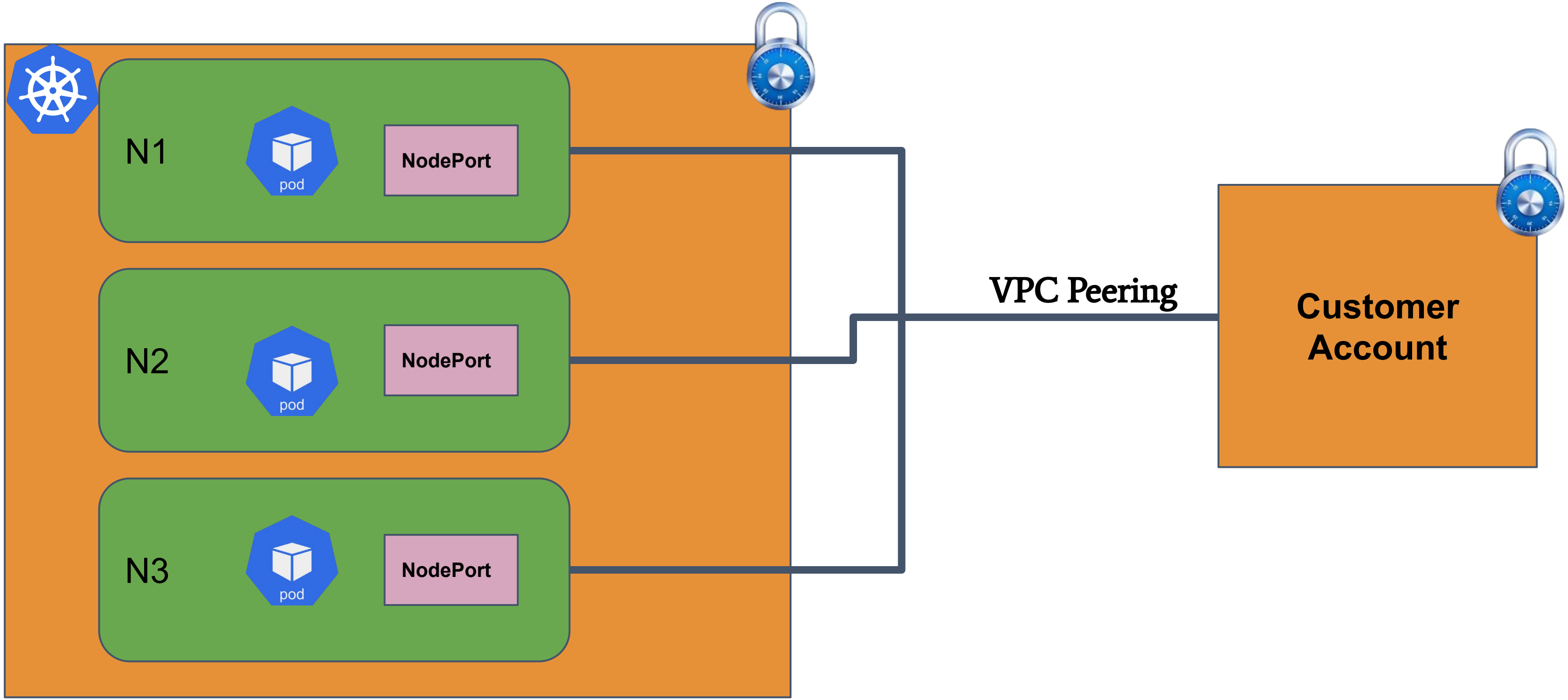
- In Console UI, customer get pre-generated cli command
- Initiates VPC Peering on customer side
- CLI prompts necessary parameters like VPC ID, Subnet, etc…
- It creates VPC Peering connection on customer side and sends request to Control Plane to create same records to your system to verify

Cloud Specific Peering Terms

	AWS	Azure	GCP
Name	VPC Peering	vNet Peering	VPC Network Peering
Requested Parameters	Account ID VPC ID Subnet ID	vNet ID	Project ID Network Name

n AWS, you can also use Private Link to convert your service into a VPC Endpoint Service. It is also best practice for enabling your service in AWS Marketplace

VPC Peering


N1
pod
NodePort
N2
pod
NodePort
N3
pod
NodePort
VPC Peering
Customer Account

Any Question?

/huseyinbabal

/huseyinbabal

https://huseyinbabal.com