SESSION - 70

SA

====

-> Access the internal workloads by communication k8 API. SA can have role and rolebinding

-> SA can fetch secrets, create ALB, any AWS service based on the role attached

1. Make sure OIDC provider exists, creates trust and federation between IAM and others

2. Create a policy

3. We can create IAM Role and attach policy.

4. We can create SA and give IAM role in annotation

or

Create SA and role through eksctl command

5. Provide sa in the pod spec.

fetch secrets

==============

follow above steps

1. Make sure secrets exists and SA have access through role and policy

2. Create initcontainer that fetch the secret and write to the volume. init container should have volume mounts to persist.

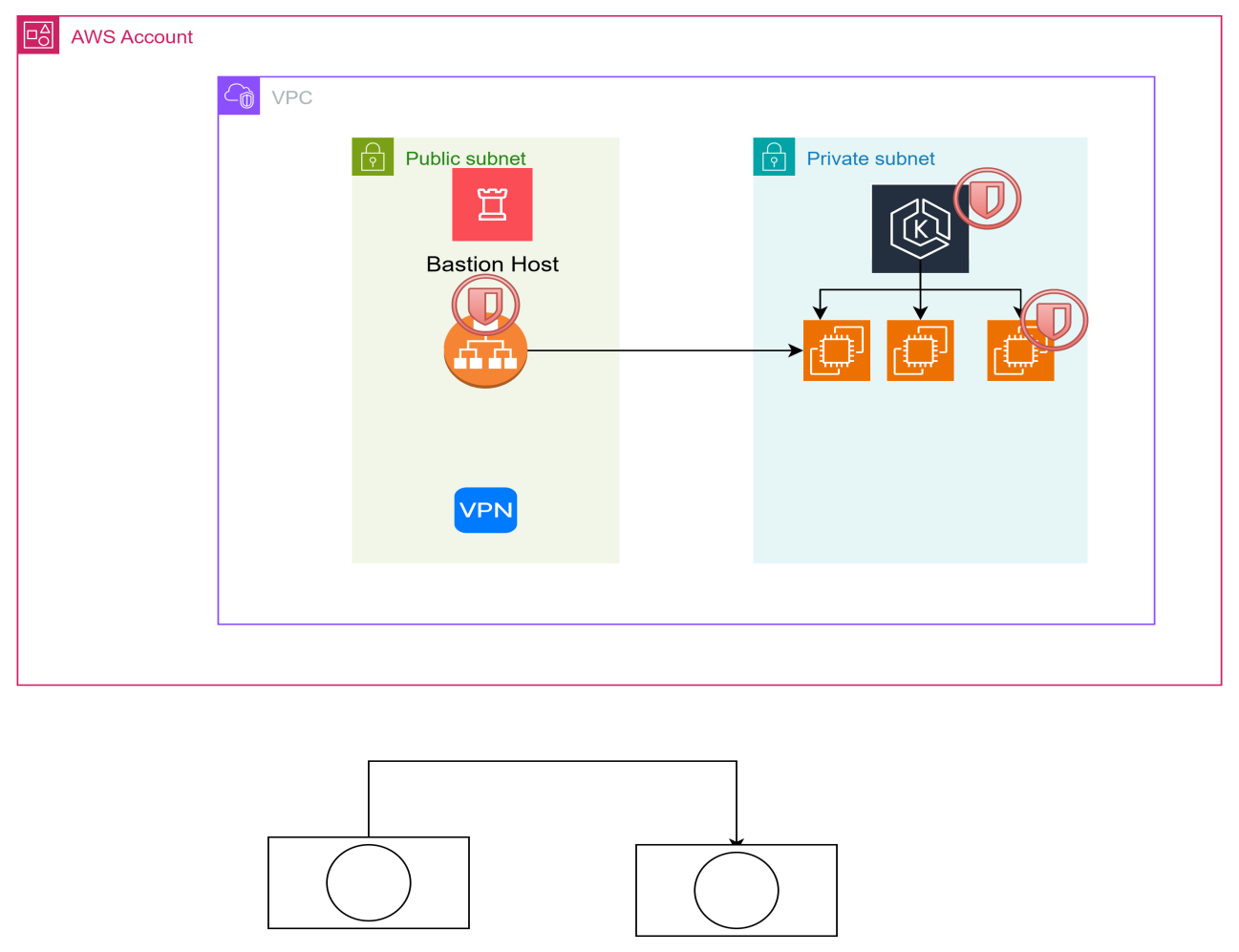
3. emptyDir is required to share the storage between containers in pod.

4. mount the same volume to the main container and access it..

5. before running command in container we need to read the secret and make it as env variable

6. now we can delete the file and start the container

--> Requests coming from load balancer.



--> create a directory terraform-aws-eks

**00-vpc/main.tf**

module "vpc" {

source = "git::https://github.com/daws-84s/terraform-aws-vpc.git?ref=main"

project = var.project

environment = var.environment

public\_subnet\_cidrs = var.public\_subnet\_cidrs

private\_subnet\_cidrs = var.private\_subnet\_cidrs

database\_subnet\_cidrs = var.database\_subnet\_cidrs

is\_peering\_required = true

}

/\* output "vpc\_ids" {

value = module.vpc.public\_subnet\_ids

} \*/

**00-vpc/parameters.tf**

resource "aws\_ssm\_parameter" "vpc\_id" {

name = "/${var.project}/${var.environment}/vpc\_id"

type = "String"

value = module.vpc.vpc\_id

}

resource "aws\_ssm\_parameter" "public\_subnet\_ids" {

name = "/${var.project}/${var.environment}/public\_subnet\_ids"

type = "StringList"

value = join(",", module.vpc.public\_subnet\_ids)

}

resource "aws\_ssm\_parameter" "private\_subnet\_ids" {

name = "/${var.project}/${var.environment}/private\_subnet\_ids"

type = "StringList"

value = join(",", module.vpc.private\_subnet\_ids)

}

resource "aws\_ssm\_parameter" "database\_subnet\_ids" {

name = "/${var.project}/${var.environment}/database\_subnet\_ids"

type = "StringList"

value = join(",", module.vpc.database\_subnet\_ids)

}

**00-vpc/variables.tf**

variable "project" {

default = "roboshop"

}

variable "environment" {

default = "dev"

}

variable "public\_subnet\_cidrs" {

default = ["10.0.1.0/24", "10.0.2.0/24"]

}

variable "private\_subnet\_cidrs" {

default = ["10.0.11.0/24", "10.0.12.0/24"]

}

variable "database\_subnet\_cidrs" {

default = ["10.0.21.0/24", "10.0.22.0/24"]

}

**00-vpc/provider.tf**

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "5.98.0"

}

}

backend "s3" {

bucket = "latest-84s-remote-state-dev"

key = "roboshop-dev-vpc"

region = "us-east-1"

encrypt = true

use\_lockfile = true

}

}

provider "aws" {

# Configuration options

region = "us-east-1"

}

--> cd /c/devops/daws-84s/repos/terraform-aws-eks/00-vpc

**--> terraform init -reconfigure**

**--> terraform plan**

**--> terraform apply -auto-approve**

1. **sg/data.tf**

data "aws\_ssm\_parameter" "vpc\_id" {

name = "/${var.project}/${var.environment}/vpc\_id"

}

1. **sg/locals.tf**

locals {

vpc\_id = data.aws\_ssm\_parameter.vpc\_id.value

}

1. **sg/parameters.tf**

resource "aws\_ssm\_parameter" "ingress\_alb\_sg\_id" {

name = "/${var.project}/${var.environment}/ingress\_alb\_sg\_id"

type = "String"

value = module.ingress\_alb.sg\_id

}

resource "aws\_ssm\_parameter" "eks\_control\_plane\_sg\_id" {

name = "/${var.project}/${var.environment}/eks\_control\_plane\_sg\_id"

type = "String"

value = module.eks\_control\_plane.sg\_id

}

resource "aws\_ssm\_parameter" "eks\_node\_sg\_id" {

name = "/${var.project}/${var.environment}/eks\_node\_sg\_id"

type = "String"

value = module.eks\_node.sg\_id

}

resource "aws\_ssm\_parameter" "bastion\_sg\_id" {

name = "/${var.project}/${var.environment}/bastion\_sg\_id"

type = "String"

value = module.bastion.sg\_id

}

resource "aws\_ssm\_parameter" "vpn\_sg\_id" {

name = "/${var.project}/${var.environment}/vpn\_sg\_id"

type = "String"

value = module.vpn.sg\_id

}

1. **sg/provider.tf**

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "5.98.0"

}

}

backend "s3" {

bucket = "latest-84s-remote-state-dev"

key = "roboshop-dev-sg"

region = "us-east-1"

encrypt = true

use\_lockfile = true

}

}

provider "aws" {

# Configuration options

region = "us-east-1"

}

1. **sg/main.tf**

module "ingress\_alb" {

#source = "../../terraform-aws-securitygroup"

source = "git::https://github.com/daws-84s/terraform-aws-securitygroup.git?ref=main"

project = var.project

environment = var.environment

sg\_name = "ingress-alb"

sg\_description = "for ingress alb"

vpc\_id = local.vpc\_id

}

module "bastion" {

#source = "../../terraform-aws-securitygroup"

source = "git::https://github.com/daws-84s/terraform-aws-securitygroup.git?ref=main"

project = var.project

environment = var.environment

sg\_name = var.bastion\_sg\_name

sg\_description = var.bastion\_sg\_description

vpc\_id = local.vpc\_id

}

module "vpn" {

#source = "../../terraform-aws-securitygroup"

source = "git::https://github.com/daws-84s/terraform-aws-securitygroup.git?ref=main"

project = var.project

environment = var.environment

sg\_name = "vpn"

sg\_description = "for vpn"

vpc\_id = local.vpc\_id

}

module "eks\_control\_plane" {

#source = "../../terraform-aws-securitygroup"

source = "git::https://github.com/daws-84s/terraform-aws-securitygroup.git?ref=main"

project = var.project

environment = var.environment

sg\_name = "eks\_control\_plane"

sg\_description = "for eks\_control\_plane"

vpc\_id = local.vpc\_id

}

module "eks\_node" {

#source = "../../terraform-aws-securitygroup"

source = "git::https://github.com/daws-84s/terraform-aws-securitygroup.git?ref=main"

project = var.project

environment = var.environment

sg\_name = "eks\_node"

sg\_description = "for eks\_node"

vpc\_id = local.vpc\_id

}

resource "aws\_security\_group\_rule" "ingress\_alb\_https" {

type = "ingress"

from\_port = 443

to\_port = 443

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

security\_group\_id = module.ingress\_alb.sg\_id

}

resource "aws\_security\_group\_rule" "bastion\_laptop" {

type = "ingress"

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

security\_group\_id = module.bastion.sg\_id

}

#VPN ports 22, 443, 1194, 943

resource "aws\_security\_group\_rule" "vpn\_ssh" {

type = "ingress"

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

security\_group\_id = module.vpn.sg\_id

}

resource "aws\_security\_group\_rule" "vpn\_https" {

type = "ingress"

from\_port = 443

to\_port = 443

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

security\_group\_id = module.vpn.sg\_id

}

resource "aws\_security\_group\_rule" "vpn\_1194" {

type = "ingress"

from\_port = 1194

to\_port = 1194

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

security\_group\_id = module.vpn.sg\_id

}

resource "aws\_security\_group\_rule" "vpn\_943" {

type = "ingress"

from\_port = 943

to\_port = 943

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

security\_group\_id = module.vpn.sg\_id

}

resource "aws\_security\_group\_rule" "eks\_control\_plane\_eks\_node" {

type = "ingress"

from\_port = 0

to\_port = 0

protocol = "-1"

source\_security\_group\_id = module.eks\_node.sg\_id

security\_group\_id = module.eks\_control\_plane.sg\_id

}

resource "aws\_security\_group\_rule" "eks\_node\_eks\_control\_plane" {

type = "ingress"

from\_port = 0

to\_port = 0

protocol = "-1"

source\_security\_group\_id = module.eks\_control\_plane.sg\_id

security\_group\_id = module.eks\_node.sg\_id

}

resource "aws\_security\_group\_rule" "eks\_control\_plane\_bastion" {

type = "ingress"

from\_port = 443

to\_port = 443

protocol = "tcp"

source\_security\_group\_id = module.bastion.sg\_id

security\_group\_id = module.eks\_control\_plane.sg\_id

}

resource "aws\_security\_group\_rule" "eks\_node\_bastion" {

type = "ingress"

from\_port = 22

to\_port = 22

protocol = "tcp"

source\_security\_group\_id = module.bastion.sg\_id

security\_group\_id = module.eks\_node.sg\_id

}

resource "aws\_security\_group\_rule" "eks\_node\_vpc" {

type = "ingress"

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["10.0.0.0/16"]

security\_group\_id = module.eks\_node.sg\_id

}

1. **sg/variables.tf**

variable "project" {

default = "roboshop"

}

variable "environment" {

default = "dev"

}

variable "bastion\_sg\_name" {

default = "bastion"

}

variable "bastion\_sg\_description" {

default = "created sg for bastion instance"

}

1. **sg/sg.yaml**

- name: Roboshop SG and Rules

security\_groups:

- DB:

- mongodb

- redis

- mysql

- rabbitmq

- APP:

- catalogue

- user

- cart

- shipping

- payment

- BACKEND\_ALB:

- FRONTEND:

- FRONTEND\_ALB:

- VPN:

- BASTION:

mongodb:

- name: mongodb\_vpn

purpose: mongodb should accept traffic on 22,27017 from vpn

- name: mongodb\_catalogue

purpose: mongodb should accept traffic on 27017 from catalogue

- name: mongodb\_user

purpose: mongodb should accept traffic on 27017 from user

redis:

- name: redis\_vpn

purpose: redis should accept traffic on 22,5679 from vpn

- name: redis\_user

purpose: redis should accept traffic on 5679 from user

- name: redis\_cart

purpose: redis should accept traffic on 5679 from cart

mysql:

- name: mysql\_vpn

purpose: mysql should accept traffic on 22,3306 from vpn

- name: mysql\_shipping

purpose: redis should accept traffic on 3306 from shipping

rabbitmq:

- name: rabbitmq\_vpn

purpose: rabbitmq should accept traffic on 22,5672 from vpn

- name: rabbitmq\_payment

purpose: redis should accept traffic on 5672 from payment

catalogue:

- name: catalogue\_vpn

purpose: catalogue should accept traffic on 22 from vpn

- name: catalogue\_backend\_alb

purpose: catalogue should accept traffic on 8080 from backend\_alb

**--> terraform init**

**--> terraform plan**

**--> terraform apply -auto-approve**

1. **bastion/data.tf**

data "aws\_ami" "joindevops" {

owners = ["973714476881"]

most\_recent = true

filter {

name = "name"

values = ["RHEL-9-DevOps-Practice"]

}

filter {

name = "root-device-type"

values = ["ebs"]

}

filter {

name = "virtualization-type"

values = ["hvm"]

}

}

data "aws\_ssm\_parameter" "bastion\_sg\_id" {

name = "/${var.project}/${var.environment}/bastion\_sg\_id"

}

data "aws\_ssm\_parameter" "public\_subnet\_ids" {

name = "/${var.project}/${var.environment}/public\_subnet\_ids"

}

**20-bastion/locals.tf**

locals{

ami\_id = data.aws\_ami.joindevops.id

bastion\_sg\_id = data.aws\_ssm\_parameter.bastion\_sg\_id.value

public\_subnet\_id = split ("," , data.aws\_ssm\_parameter.public\_subnet\_ids.value)[0]

common\_tags = {

Project = var.project

Environment = var.environment

Terraform = "true"

}

}

--> Check git hub account and add remaining scripts.

**--> terraform init**

**--> terraform plan**

**--> terraform apply -auto-approve**

1. **acm/locals.tf**

locals {

common\_tags = {

Project = var.project

Environment = var.environment

Terraform = "true"

}

}

**60-acm/main.tf**

resource "aws\_acm\_certificate" "daws84s" {

domain\_name = "\*.${var.zone\_name}"

validation\_method = "DNS"

tags = merge(

local.common\_tags,

{

Name = "${var.project}-${var.environment}"

}

)

lifecycle {

create\_before\_destroy = true

}

}

resource "aws\_route53\_record" "daws84s" {

for\_each = {

for dvo in aws\_acm\_certificate.daws84s.domain\_validation\_options : dvo.domain\_name => {

name = dvo.resource\_record\_name

record = dvo.resource\_record\_value

type = dvo.resource\_record\_type

}

}

allow\_overwrite = true

name = each.value.name

records = [each.value.record]

ttl = 60

type = each.value.type

zone\_id = var.zone\_id

}

resource "aws\_acm\_certificate\_validation" "daws84s" {

certificate\_arn = aws\_acm\_certificate.daws84s.arn

validation\_record\_fqdns = [for record in aws\_route53\_record.daws84s : record.fqdn]

}

--> Check git hub account and add remaining scripts.

**--> terraform init**

**--> terraform plan**

**--> terraform apply -auto-approve**

**70-frontend-alb/locals.tf**

locals {

vpc\_id = data.aws\_ssm\_parameter.vpc\_id.value

public\_subnet\_ids = split ("," , data.aws\_ssm\_parameter.public\_subnet\_ids.value)

ingress\_alb\_sg\_id = data.aws\_ssm\_parameter.ingress\_alb\_sg\_id.value

acm\_certificate\_arn = data.aws\_ssm\_parameter.acm\_certificate\_arn.value

common\_tags = {

Project = var.project

Environment = var.environment

Terraform = "true"

}

}

**70-frontend-alb/variables.tf**

variable "project" {

default = "roboshop"

}

variable "environment" {

default = "dev"

}

variable "zone\_id" {

default = "Z032558618100M4EJX8X4"

}

variable "zone\_name" {

default = "daws84s.site"

}

--> Check git hub account and add remaining scripts.

**--> terraform init**

**--> terraform plan**

**--> terraform apply -auto-approve**

1. You create LB from K8

2. You create LB outside of K8 and provide the info to K8

--> This is part of infra, road balancer is a not of EKS cluster.

--> you create it and then let kubernates can connect

**80-eks/data.tf**

data "aws\_ssm\_parameter" "vpc\_id" {

name = "/${var.project}/${var.environment}/vpc\_id"

}

data "aws\_ssm\_parameter" "private\_subnet\_ids" {

name = "/${var.project}/${var.environment}/private\_subnet\_ids"

}

data "aws\_ssm\_parameter" "eks\_control\_plane\_sg\_id" {

name = "/${var.project}/${var.environment}/eks\_control\_plane\_sg\_id"

}

data "aws\_ssm\_parameter" "eks\_node\_sg\_id" {

name = "/${var.project}/${var.environment}/eks\_node\_sg\_id"

}

**80-eks/locals.tf**

locals {

vpc\_id = data.aws\_ssm\_parameter.vpc\_id.value

eks\_node\_sg\_id = data.aws\_ssm\_parameter.eks\_node\_sg\_id.value

eks\_control\_plane\_sg\_id = data.aws\_ssm\_parameter.eks\_control\_plane\_sg\_id.value

private\_subnet\_ids = split ("," , data.aws\_ssm\_parameter.private\_subnet\_ids.value)

common\_tags = {

Project = var.project

Environment = var.environment

Terraform = "true"

}

}

**80-eks/main.tf**

module "eks" {

source = "terraform-aws-modules/eks/aws"

version = "~> 21.0" # this is module version

name = "${var.project}-${var.environment}"

kubernetes\_version = "1.33"

addons = {

coredns = {}

eks-pod-identity-agent = {

before\_compute = true

}

kube-proxy = {}

vpc-cni = {

before\_compute = true

}

metrics-server= {}

}

# Optional

endpoint\_public\_access = false

# Optional: Adds the current caller identity as an administrator via cluster access entry

enable\_cluster\_creator\_admin\_permissions = true

vpc\_id = local.vpc\_id

subnet\_ids = local.private\_subnet\_ids

control\_plane\_subnet\_ids = local.private\_subnet\_ids

create\_node\_security\_group = false

create\_security\_group = false

security\_group\_id = local.eks\_control\_plane\_sg\_id

node\_security\_group\_id = local.eks\_node\_sg\_id

# EKS Managed Node Group(s)

eks\_managed\_node\_groups = {

/\* blue = {

# Starting on 1.30, AL2023 is the default AMI type for EKS managed node groups

ami\_type = "AL2023\_x86\_64\_STANDARD" # user name is ec2-user

instance\_types = ["m5.xlarge"]

min\_size = 2

max\_size = 10

desired\_size = 2

} \*/

# iam\_role\_additional\_policies = {

# AmazonEBS = "arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy"

# AmazonEFS = "arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy"

# AmazonEKSLoad = "arn:aws:iam::aws:policy/AmazonEKSLoadBalancingPolicy"

# }

green = {

# Starting on 1.30, AL2023 is the default AMI type for EKS managed node groups

ami\_type = "AL2023\_x86\_64\_STANDARD" # user name is ec2-user

instance\_types = ["m5.xlarge"]

min\_size = 2

max\_size = 10

desired\_size = 2

iam\_role\_additional\_policies = {

AmazonEBS = "arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy"

AmazonEFS = "arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy"

AmazonEKSLoad = "arn:aws:iam::aws:policy/AmazonEKSLoadBalancingPolicy"

}

/\* taints = {

upgrade = {

key = "upgrade"

value = "true"

effect = "NO\_SCHEDULE"

}

} \*/

}

}

tags = merge(

local.common\_tags,

{

Name = "${var.project}-${var.environment}"

}

)

}

Create amazon elastic kubernates

--> create EKS cluster

--> Check git hub account and add remaining scripts.

**--> terraform init**

**--> terraform plan**

**--> terraform apply -auto-approve**