SESSION - 71

**--> terraform plan**

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

--> connect the AWS server.

--> aws configure

**--> aws sts get-caller-identity**

**--> aws eks update-kubeconfig --region us-east-1 --name roboshop-dev**

**--> kubectl get nodes**

**--> git clone https://**

**--> cd k8-resources**

**--> ls -l**

**--> kubectl apply -f 14-service-lb.yaml**

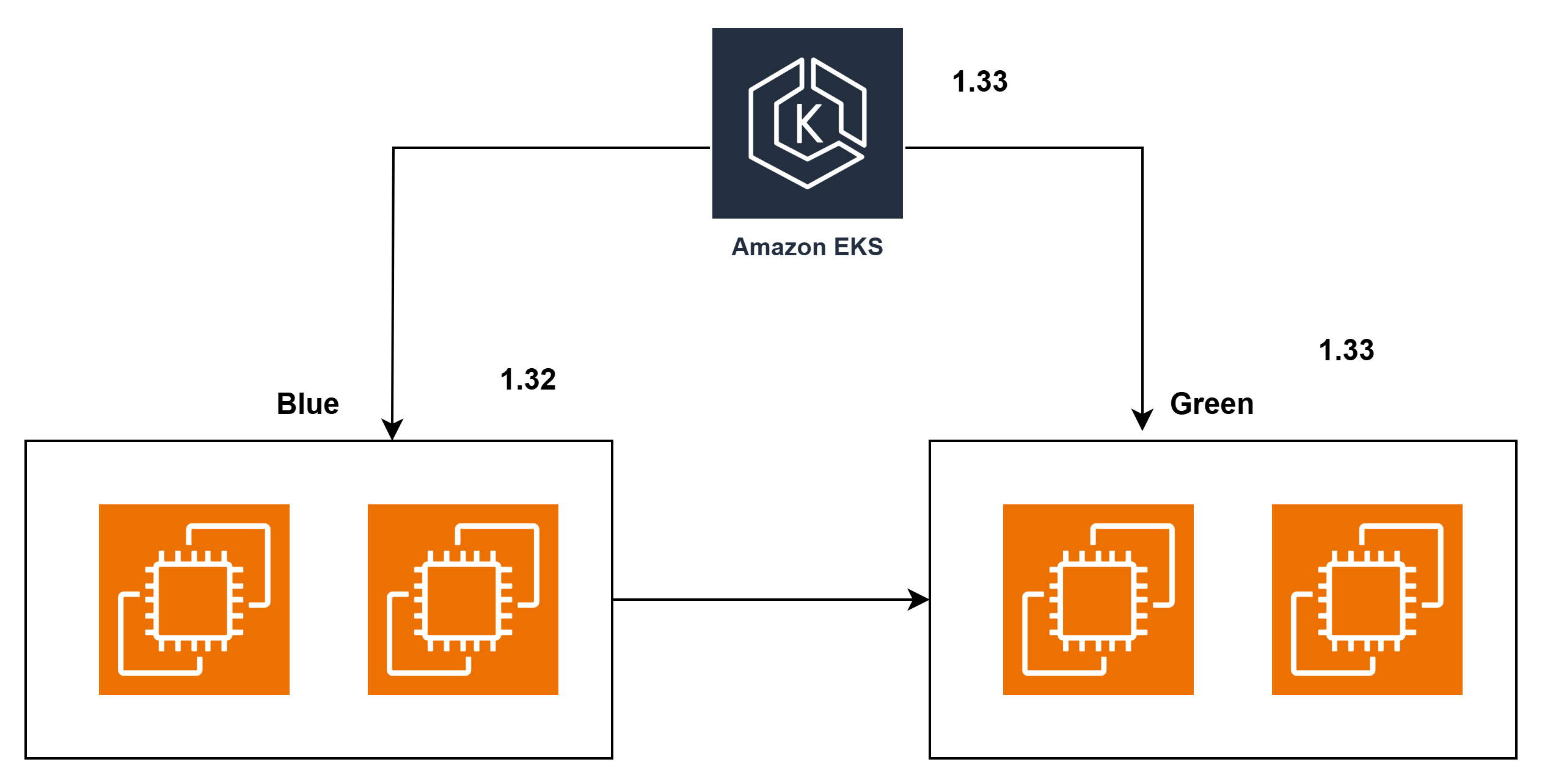
**--> kubectl get svc**

**--> terraform plan**

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

--> Load balancer was created will see in load balancers

--> check target instances



**Minimum Downtime or Zero Downtime**

**==================================**

Announce Downtime while platform upgrade. Take 3hours downtime. Make sure changes in firewalls so that other teams can't connect to cluster. We make sure no changes to apps happen while upgrade is going on.

--> 1. EKS control plane and NG is fully functional.

2. Create one NG blue/green with same capacity. Taint the nodes

3. Upgrade controlplane to 1.33

4. New NG also should be upgraded to same 1.33. Old node group is still in 1.32

5. Cordon old nodes. Untaint new NG

6. Start draining the nodes. Workloads shift to new NG

7. Delete old NG, and manually edit the controlplane to 1.33

8. Create firewalls back and announce upgr ade is completed. Ask app team to check their applications.

**Blue Green**

**==========**

zero downtime deployment strategy, where we create the same capacity of infra if blue is running version we create green and vice versa, test it and switch the workloads to new infra, delete old infra

**--> kubectl get nodes**

**--> kubectl describe node file-name**

**-->** check clusters in aws account

**Rolling Update**

**===========**

4 pods

2min

create 5th pod with new version

delete one old old

3 old 1 new

creaet 6th pod with new version

delete 2nd old pod

2 old 2 new

creaet 7th pod with new version

delete 3rd old pod

1 old 3 new

creaet 8th pod with new version

delete 4th old pod

0 old 4 new

at the same time users may see 2 diff versions, but code should handle this

**Blue Green**

**===========**

LB -> Listener -> Rule -> TG -> VM

present blue TG is running

create new Green TG, add the nodes to this

Create new internal LB, attach Listener, Rule and new TG

Test the application with internal URL.

Edit Main LB rule to send traffic to new TG

We can remove old TG, but let's keep it as backup/rollback.

**--> kubectl get nodes**

**--> kubectl taint nodes log-name upgrade=true:NoSchedule-**

**--> kubectl cordon log-name**

**--> kubectl get nodes**

**--> kubectl drain --ignore-daemonsets --delete-emptydir-data log-name**

**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/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}"

}

)

}

**--> cd terraform-aws-eks**

**--> terraform init**

**--> terraform plan**

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

**-->**