[root@centos kubernets]# terraform init

Initializing the backend...

Initializing modules...

Downloading registry.terraform.io/terraform-aws-modules/eks/aws 18.29.0 for eks...

- eks in .terraform/modules/eks

- eks.eks\_managed\_node\_group in .terraform/modules/eks/modules/eks-managed-node-group

- eks.eks\_managed\_node\_group.user\_data in .terraform/modules/eks/modules/\_user\_data

- eks.fargate\_profile in .terraform/modules/eks/modules/fargate-profile

Downloading registry.terraform.io/terraform-aws-modules/kms/aws 1.0.2 for eks.kms...

- eks.kms in .terraform/modules/eks.kms

- eks.self\_managed\_node\_group in .terraform/modules/eks/modules/self-managed-node-group

- eks.self\_managed\_node\_group.user\_data in .terraform/modules/eks/modules/\_user\_data

Downloading registry.terraform.io/terraform-aws-modules/vpc/aws 3.14.3 for vpc...

- vpc in .terraform/modules/vpc

Initializing provider plugins...

- Finding gavinbunney/kubectl versions matching ">= 1.14.0"...

- Finding hashicorp/helm versions matching ">= 2.6.0"...

- Finding hashicorp/aws versions matching ">= 3.63.0, >= 3.72.0"...

- Finding hashicorp/tls versions matching ">= 3.0.0"...

- Finding hashicorp/kubernetes versions matching ">= 2.10.0"...

- Finding hashicorp/cloudinit versions matching ">= 2.0.0"...

- Installing gavinbunney/kubectl v1.14.0...

- Installed gavinbunney/kubectl v1.14.0 (self-signed, key ID AD64217B5ADD572F)

- Installing hashicorp/helm v2.10.1...

- Installed hashicorp/helm v2.10.1 (signed by HashiCorp)

- Installing hashicorp/aws v5.2.0...

- Installed hashicorp/aws v5.2.0 (signed by HashiCorp)

- Installing hashicorp/tls v4.0.4...

- Installed hashicorp/tls v4.0.4 (signed by HashiCorp)

- Installing hashicorp/kubernetes v2.21.1...

- Installed hashicorp/kubernetes v2.21.1 (signed by HashiCorp)

- Installing hashicorp/cloudinit v2.3.2...

- Installed hashicorp/cloudinit v2.3.2 (signed by HashiCorp)

Partner and community providers are signed by their developers.

If you'd like to know more about provider signing, you can read about it here:

https://www.terraform.io/docs/cli/plugins/signing.html

Terraform has created a lock file .terraform.lock.hcl to record the provider

selections it made above. Include this file in your version control repository

so that Terraform can guarantee to make the same selections by default when

you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see

any changes that are required for your infrastructure. All Terraform commands

should now work.

If you ever set or change modules or backend configuration for Terraform,

rerun this command to reinitialize your working directory. If you forget, other

commands will detect it and remind you to do so if necessary.

[root@centos kubernets]# terraform apply

╷

│ Warning: Argument is deprecated

│

│ with module.vpc.aws\_eip.nat,

│ on .terraform/modules/vpc/main.tf line 1004, in resource "aws\_eip" "nat":

│ 1004: vpc = true

│

│ use domain attribute instead

╵

╷

│ Error: Unsupported argument

│

│ on .terraform/modules/vpc/main.tf line 27, in resource "aws\_vpc" "this":

│ 27: enable\_classiclink = var.enable\_classiclink

│

│ An argument named "enable\_classiclink" is not expected here.

╵

╷

│ Error: Unsupported argument

│

│ on .terraform/modules/vpc/main.tf line 28, in resource "aws\_vpc" "this":

│ 28: enable\_classiclink\_dns\_support = var.enable\_classiclink\_dns\_support

│

│ An argument named "enable\_classiclink\_dns\_support" is not expected here.

╵

╷

│ Error: Unsupported argument

│

│ on .terraform/modules/vpc/main.tf line 1237, in resource "aws\_default\_vpc" "this":

│ 1237: enable\_classiclink = var.default\_vpc\_enable\_classiclink

│

│ An argument named "enable\_classiclink" is not expected here.

╵

[root@centos kubernets]# ls -lrt

total 28

-rw-r--r-- 1 root root 361 Jun 10 11:39 main

-rw-r--r-- 1 root root 282 Jun 10 11:42 provider.tf

-rw-r--r-- 1 root root 801 Jun 10 12:23 security.tf

drwxr-xr-x 4 root root 38 Jun 10 13:24 old\_terraform

-rw-r--r-- 1 root root 991 Jun 10 13:29 eks.tf

-rw-r--r-- 1 root root 480 Jun 10 13:30 vpc.tf

-rw-r--r-- 1 root root 6574 Jun 10 13:31 old\_terraform.lock.hcl

[root@centos kubernets]#

Error due to vpc version after changed to 5.0.0 and issue fixed

root@centos kubernets]# cat vpc.tf

module "vpc" {

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

version = "5.0.0"

name = "main"

cidr = "10.0.0.0/16"

azs = ["us-east-1a", "us-east-1b"]

private\_subnets = ["10.0.0.0/19", "10.0.32.0/19"]

public\_subnets = ["10.0.64.0/19", "10.0.96.0/19"]

enable\_nat\_gateway = true

single\_nat\_gateway = true

one\_nat\_gateway\_per\_az = false

enable\_dns\_hostnames = true

enable\_dns\_support = true

tags = {

Environment = "staging"

}

}

[root@centos kubernets]#

[root@centos kubernets]# terraform init

Initializing the backend...

Initializing modules...

Downloading registry.terraform.io/terraform-aws-modules/vpc/aws 5.0.0 for vpc...

- vpc in .terraform/modules/vpc

Initializing provider plugins...

- Reusing previous version of hashicorp/aws from the dependency lock file

- Reusing previous version of hashicorp/tls from the dependency lock file

- Reusing previous version of hashicorp/kubernetes from the dependency lock file

- Reusing previous version of hashicorp/cloudinit from the dependency lock file

- Reusing previous version of hashicorp/helm from the dependency lock file

- Reusing previous version of gavinbunney/kubectl from the dependency lock file

- Using previously-installed hashicorp/helm v2.10.1

- Using previously-installed gavinbunney/kubectl v1.14.0

- Using previously-installed hashicorp/aws v5.2.0

- Using previously-installed hashicorp/tls v4.0.4

- Using previously-installed hashicorp/kubernetes v2.21.1

- Using previously-installed hashicorp/cloudinit v2.3.2

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see

any changes that are required for your infrastructure. All Terraform commands

should now work.

If you ever set or change modules or backend configuration for Terraform,

rerun this command to reinitialize your working directory. If you forget, other

commands will detect it and remind you to do so if necessary.

[root@centos kubernets]# terraform apply

module.eks.module.kms.data.aws\_caller\_identity.current: Reading...

module.eks.module.eks\_managed\_node\_group["general"].data.aws\_caller\_identity.current: Reading...

module.eks.data.aws\_partition.current: Reading...

module.eks.data.aws\_caller\_identity.current: Reading...

module.eks.module.eks\_managed\_node\_group["general"].data.aws\_partition.current: Reading...

module.eks.module.eks\_managed\_node\_group["spot"].data.aws\_caller\_identity.current: Reading...

module.eks.module.kms.data.aws\_partition.current: Reading...

module.eks.module.eks\_managed\_node\_group["spot"].data.aws\_partition.current: Reading...

module.eks.module.kms.data.aws\_partition.current: Read complete after 0s [id=aws]

module.eks.module.eks\_managed\_node\_group["spot"].data.aws\_partition.current: Read complete after 0s [id=aws]

module.eks.data.aws\_partition.current: Read complete after 0s [id=aws]

module.eks.module.eks\_managed\_node\_group["general"].data.aws\_partition.current: Read complete after 0s [id=aws]

module.eks.module.eks\_managed\_node\_group["spot"].data.aws\_iam\_policy\_document.assume\_role\_policy[0]: Reading...

module.eks.module.eks\_managed\_node\_group["general"].data.aws\_iam\_policy\_document.assume\_role\_policy[0]: Reading...

module.eks.module.eks\_managed\_node\_group["spot"].data.aws\_iam\_policy\_document.assume\_role\_policy[0]: Read complete after 0s [id=1734879000]

module.eks.module.eks\_managed\_node\_group["general"].data.aws\_iam\_policy\_document.assume\_role\_policy[0]: Read complete after 0s [id=1734879000]

module.eks.data.aws\_iam\_policy\_document.assume\_role\_policy[0]: Reading...

module.eks.data.aws\_iam\_policy\_document.assume\_role\_policy[0]: Read complete after 0s [id=1530481229]

module.eks.module.eks\_managed\_node\_group["spot"].data.aws\_caller\_identity.current: Read complete after 1s [id=667904429198]

module.eks.module.kms.data.aws\_caller\_identity.current: Read complete after 1s [id=667904429198]

module.eks.module.eks\_managed\_node\_group["general"].data.aws\_caller\_identity.current: Read complete after 1s [id=667904429198]

module.eks.data.aws\_caller\_identity.current: Read complete after 1s [id=667904429198]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the

following symbols:

+ create

<= read (data resources)

Terraform will perform the following actions:

# aws\_security\_group.all\_worker\_mgmt will be created

+ resource "aws\_security\_group" "all\_worker\_mgmt" {

+ arn = (known after apply)

+ description = "Managed by Terraform"

+ egress = (known after apply)

+ id = (known after apply)

+ ingress = [

+ {

+ cidr\_blocks = [

+ "10.0.0.0/8",

]

+ description = ""

+ from\_port = 22

+ ipv6\_cidr\_blocks = []

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_groups = []

+ self = false

+ to\_port = 22

},

]

+ name = (known after apply)

+ name\_prefix = "all\_worker\_management"

+ owner\_id = (known after apply)

+ revoke\_rules\_on\_delete = false

+ tags\_all = (known after apply)

+ vpc\_id = (known after apply)

}

# aws\_security\_group.worker\_group\_mgmt\_one will be created

+ resource "aws\_security\_group" "worker\_group\_mgmt\_one" {

+ arn = (known after apply)

+ description = "Managed by Terraform"

+ egress = (known after apply)

+ id = (known after apply)

+ ingress = [

+ {

+ cidr\_blocks = [

+ "10.0.0.0/8",

]

+ description = ""

+ from\_port = 22

+ ipv6\_cidr\_blocks = []

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_groups = []

+ self = false

+ to\_port = 22

},

]

+ name = (known after apply)

+ name\_prefix = "worker\_group\_mgmt\_one"

+ owner\_id = (known after apply)

+ revoke\_rules\_on\_delete = false

+ tags\_all = (known after apply)

+ vpc\_id = (known after apply)

}

# aws\_security\_group.worker\_group\_mgmt\_two will be created

+ resource "aws\_security\_group" "worker\_group\_mgmt\_two" {

+ arn = (known after apply)

+ description = "Managed by Terraform"

+ egress = (known after apply)

+ id = (known after apply)

+ ingress = [

+ {

+ cidr\_blocks = [

+ "10.0.0.0/8",

]

+ description = ""

+ from\_port = 22

+ ipv6\_cidr\_blocks = []

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_groups = []

+ self = false

+ to\_port = 22

},

]

+ name = (known after apply)

+ name\_prefix = "worker\_group\_mgmt\_two"

+ owner\_id = (known after apply)

+ revoke\_rules\_on\_delete = false

+ tags\_all = (known after apply)

+ vpc\_id = (known after apply)

}

# module.eks.data.tls\_certificate.this[0] will be read during apply

# (config refers to values not yet known)

<= data "tls\_certificate" "this" {

+ certificates = (known after apply)

+ id = (known after apply)

+ url = (known after apply)

}

# module.eks.aws\_cloudwatch\_log\_group.this[0] will be created

+ resource "aws\_cloudwatch\_log\_group" "this" {

+ arn = (known after apply)

+ id = (known after apply)

+ name = "/aws/eks/my-eks/cluster"

+ name\_prefix = (known after apply)

+ retention\_in\_days = 90

+ skip\_destroy = false

+ tags = {

+ "Environment" = "staging"

}

+ tags\_all = {

+ "Environment" = "staging"

}

}

# module.eks.aws\_ec2\_tag.cluster\_primary\_security\_group["Environment"] will be created

+ resource "aws\_ec2\_tag" "cluster\_primary\_security\_group" {

+ id = (known after apply)

+ key = "Environment"

+ resource\_id = (known after apply)

+ value = "staging"

}

# module.eks.aws\_eks\_cluster.this[0] will be created

+ resource "aws\_eks\_cluster" "this" {

+ arn = (known after apply)

+ certificate\_authority = (known after apply)

+ cluster\_id = (known after apply)

+ created\_at = (known after apply)

+ enabled\_cluster\_log\_types = [

+ "api",

+ "audit",

+ "authenticator",

]

+ endpoint = (known after apply)

+ id = (known after apply)

+ identity = (known after apply)

+ name = "my-eks"

+ platform\_version = (known after apply)

+ role\_arn = (known after apply)

+ status = (known after apply)

+ tags = {

+ "Environment" = "staging"

}

+ tags\_all = {

+ "Environment" = "staging"

}

+ version = "1.23"

+ kubernetes\_network\_config {

+ ip\_family = (known after apply)

+ service\_ipv4\_cidr = (known after apply)

+ service\_ipv6\_cidr = (known after apply)

}

+ timeouts {}

+ vpc\_config {

+ cluster\_security\_group\_id = (known after apply)

+ endpoint\_private\_access = true

+ endpoint\_public\_access = true

+ public\_access\_cidrs = [

+ "0.0.0.0/0",

]

+ security\_group\_ids = (known after apply)

+ subnet\_ids = (known after apply)

+ vpc\_id = (known after apply)

}

}

# module.eks.aws\_iam\_openid\_connect\_provider.oidc\_provider[0] will be created

+ resource "aws\_iam\_openid\_connect\_provider" "oidc\_provider" {

+ arn = (known after apply)

+ client\_id\_list = [

+ "sts.amazonaws.com",

]

+ id = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "my-eks-eks-irsa"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "my-eks-eks-irsa"

}

+ thumbprint\_list = (known after apply)

+ url = (known after apply)

}

# module.eks.aws\_iam\_role.this[0] will be created

+ resource "aws\_iam\_role" "this" {

+ arn = (known after apply)

+ assume\_role\_policy = jsonencode(

{

+ Statement = [

+ {

+ Action = "sts:AssumeRole"

+ Effect = "Allow"

+ Principal = {

+ Service = "eks.amazonaws.com"

}

+ Sid = "EKSClusterAssumeRole"

},

]

+ Version = "2012-10-17"

}

)

+ create\_date = (known after apply)

+ force\_detach\_policies = true

+ id = (known after apply)

+ managed\_policy\_arns = (known after apply)

+ max\_session\_duration = 3600

+ name = (known after apply)

+ name\_prefix = "my-eks-cluster-"

+ path = "/"

+ tags = {

+ "Environment" = "staging"

}

+ tags\_all = {

+ "Environment" = "staging"

}

+ unique\_id = (known after apply)

+ inline\_policy {

+ name = "my-eks-cluster"

+ policy = (known after apply)

}

}

# module.eks.aws\_iam\_role\_policy\_attachment.this["arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"] will be created

+ resource "aws\_iam\_role\_policy\_attachment" "this" {

+ id = (known after apply)

+ policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"

+ role = (known after apply)

}

# module.eks.aws\_iam\_role\_policy\_attachment.this["arn:aws:iam::aws:policy/AmazonEKSVPCResourceController"] will be created

+ resource "aws\_iam\_role\_policy\_attachment" "this" {

+ id = (known after apply)

+ policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSVPCResourceController"

+ role = (known after apply)

}

# module.eks.aws\_security\_group.cluster[0] will be created

+ resource "aws\_security\_group" "cluster" {

+ arn = (known after apply)

+ description = "EKS cluster security group"

+ egress = (known after apply)

+ id = (known after apply)

+ ingress = (known after apply)

+ name = (known after apply)

+ name\_prefix = "my-eks-cluster-"

+ owner\_id = (known after apply)

+ revoke\_rules\_on\_delete = false

+ tags = {

+ "Environment" = "staging"

+ "Name" = "my-eks-cluster"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "my-eks-cluster"

}

+ vpc\_id = (known after apply)

}

# module.eks.aws\_security\_group.node[0] will be created

+ resource "aws\_security\_group" "node" {

+ arn = (known after apply)

+ description = "EKS node shared security group"

+ egress = (known after apply)

+ id = (known after apply)

+ ingress = (known after apply)

+ name = (known after apply)

+ name\_prefix = "my-eks-node-"

+ owner\_id = (known after apply)

+ revoke\_rules\_on\_delete = false

+ tags = {

+ "Environment" = "staging"

+ "Name" = "my-eks-node"

+ "kubernetes.io/cluster/my-eks" = "owned"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "my-eks-node"

+ "kubernetes.io/cluster/my-eks" = "owned"

}

+ vpc\_id = (known after apply)

}

# module.eks.aws\_security\_group\_rule.cluster["egress\_nodes\_443"] will be created

+ resource "aws\_security\_group\_rule" "cluster" {

+ description = "Cluster API to node groups"

+ from\_port = 443

+ id = (known after apply)

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_group\_id = (known after apply)

+ security\_group\_rule\_id = (known after apply)

+ self = false

+ source\_security\_group\_id = (known after apply)

+ to\_port = 443

+ type = "egress"

}

# module.eks.aws\_security\_group\_rule.cluster["egress\_nodes\_kubelet"] will be created

+ resource "aws\_security\_group\_rule" "cluster" {

+ description = "Cluster API to node kubelets"

+ from\_port = 10250

+ id = (known after apply)

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_group\_id = (known after apply)

+ security\_group\_rule\_id = (known after apply)

+ self = false

+ source\_security\_group\_id = (known after apply)

+ to\_port = 10250

+ type = "egress"

}

# module.eks.aws\_security\_group\_rule.cluster["ingress\_nodes\_443"] will be created

+ resource "aws\_security\_group\_rule" "cluster" {

+ description = "Node groups to cluster API"

+ from\_port = 443

+ id = (known after apply)

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_group\_id = (known after apply)

+ security\_group\_rule\_id = (known after apply)

+ self = false

+ source\_security\_group\_id = (known after apply)

+ to\_port = 443

+ type = "ingress"

}

# module.eks.aws\_security\_group\_rule.node["egress\_cluster\_443"] will be created

+ resource "aws\_security\_group\_rule" "node" {

+ description = "Node groups to cluster API"

+ from\_port = 443

+ id = (known after apply)

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_group\_id = (known after apply)

+ security\_group\_rule\_id = (known after apply)

+ self = false

+ source\_security\_group\_id = (known after apply)

+ to\_port = 443

+ type = "egress"

}

# module.eks.aws\_security\_group\_rule.node["egress\_https"] will be created

+ resource "aws\_security\_group\_rule" "node" {

+ cidr\_blocks = [

+ "0.0.0.0/0",

]

+ description = "Egress all HTTPS to internet"

+ from\_port = 443

+ id = (known after apply)

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_group\_id = (known after apply)

+ security\_group\_rule\_id = (known after apply)

+ self = false

+ source\_security\_group\_id = (known after apply)

+ to\_port = 443

+ type = "egress"

}

# module.eks.aws\_security\_group\_rule.node["egress\_ntp\_tcp"] will be created

+ resource "aws\_security\_group\_rule" "node" {

+ cidr\_blocks = [

+ "0.0.0.0/0",

]

+ description = "Egress NTP/TCP to internet"

+ from\_port = 123

+ id = (known after apply)

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_group\_id = (known after apply)

+ security\_group\_rule\_id = (known after apply)

+ self = false

+ source\_security\_group\_id = (known after apply)

+ to\_port = 123

+ type = "egress"

}

# module.eks.aws\_security\_group\_rule.node["egress\_ntp\_udp"] will be created

+ resource "aws\_security\_group\_rule" "node" {

+ cidr\_blocks = [

+ "0.0.0.0/0",

]

+ description = "Egress NTP/UDP to internet"

+ from\_port = 123

+ id = (known after apply)

+ prefix\_list\_ids = []

+ protocol = "udp"

+ security\_group\_id = (known after apply)

+ security\_group\_rule\_id = (known after apply)

+ self = false

+ source\_security\_group\_id = (known after apply)

+ to\_port = 123

+ type = "egress"

}

# module.eks.aws\_security\_group\_rule.node["egress\_self\_coredns\_tcp"] will be created

+ resource "aws\_security\_group\_rule" "node" {

+ description = "Node to node CoreDNS"

+ from\_port = 53

+ id = (known after apply)

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_group\_id = (known after apply)

+ security\_group\_rule\_id = (known after apply)

+ self = true

+ source\_security\_group\_id = (known after apply)

+ to\_port = 53

+ type = "egress"

}

# module.eks.aws\_security\_group\_rule.node["egress\_self\_coredns\_udp"] will be created

+ resource "aws\_security\_group\_rule" "node" {

+ description = "Node to node CoreDNS"

+ from\_port = 53

+ id = (known after apply)

+ prefix\_list\_ids = []

+ protocol = "udp"

+ security\_group\_id = (known after apply)

+ security\_group\_rule\_id = (known after apply)

+ self = true

+ source\_security\_group\_id = (known after apply)

+ to\_port = 53

+ type = "egress"

}

# module.eks.aws\_security\_group\_rule.node["ingress\_cluster\_443"] will be created

+ resource "aws\_security\_group\_rule" "node" {

+ description = "Cluster API to node groups"

+ from\_port = 443

+ id = (known after apply)

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_group\_id = (known after apply)

+ security\_group\_rule\_id = (known after apply)

+ self = false

+ source\_security\_group\_id = (known after apply)

+ to\_port = 443

+ type = "ingress"

}

# module.eks.aws\_security\_group\_rule.node["ingress\_cluster\_kubelet"] will be created

+ resource "aws\_security\_group\_rule" "node" {

+ description = "Cluster API to node kubelets"

+ from\_port = 10250

+ id = (known after apply)

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_group\_id = (known after apply)

+ security\_group\_rule\_id = (known after apply)

+ self = false

+ source\_security\_group\_id = (known after apply)

+ to\_port = 10250

+ type = "ingress"

}

# module.eks.aws\_security\_group\_rule.node["ingress\_self\_coredns\_tcp"] will be created

+ resource "aws\_security\_group\_rule" "node" {

+ description = "Node to node CoreDNS"

+ from\_port = 53

+ id = (known after apply)

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_group\_id = (known after apply)

+ security\_group\_rule\_id = (known after apply)

+ self = true

+ source\_security\_group\_id = (known after apply)

+ to\_port = 53

+ type = "ingress"

}

# module.eks.aws\_security\_group\_rule.node["ingress\_self\_coredns\_udp"] will be created

+ resource "aws\_security\_group\_rule" "node" {

+ description = "Node to node CoreDNS"

+ from\_port = 53

+ id = (known after apply)

+ prefix\_list\_ids = []

+ protocol = "udp"

+ security\_group\_id = (known after apply)

+ security\_group\_rule\_id = (known after apply)

+ self = true

+ source\_security\_group\_id = (known after apply)

+ to\_port = 53

+ type = "ingress"

}

# module.vpc.aws\_default\_network\_acl.this[0] will be created

+ resource "aws\_default\_network\_acl" "this" {

+ arn = (known after apply)

+ default\_network\_acl\_id = (known after apply)

+ id = (known after apply)

+ owner\_id = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "main-default"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "main-default"

}

+ vpc\_id = (known after apply)

+ egress {

+ action = "allow"

+ from\_port = 0

+ ipv6\_cidr\_block = "::/0"

+ protocol = "-1"

+ rule\_no = 101

+ to\_port = 0

}

+ egress {

+ action = "allow"

+ cidr\_block = "0.0.0.0/0"

+ from\_port = 0

+ protocol = "-1"

+ rule\_no = 100

+ to\_port = 0

}

+ ingress {

+ action = "allow"

+ from\_port = 0

+ ipv6\_cidr\_block = "::/0"

+ protocol = "-1"

+ rule\_no = 101

+ to\_port = 0

}

+ ingress {

+ action = "allow"

+ cidr\_block = "0.0.0.0/0"

+ from\_port = 0

+ protocol = "-1"

+ rule\_no = 100

+ to\_port = 0

}

}

# module.vpc.aws\_default\_route\_table.default[0] will be created

+ resource "aws\_default\_route\_table" "default" {

+ arn = (known after apply)

+ default\_route\_table\_id = (known after apply)

+ id = (known after apply)

+ owner\_id = (known after apply)

+ route = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "main-default"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "main-default"

}

+ vpc\_id = (known after apply)

+ timeouts {

+ create = "5m"

+ update = "5m"

}

}

# module.vpc.aws\_default\_security\_group.this[0] will be created

+ resource "aws\_default\_security\_group" "this" {

+ arn = (known after apply)

+ description = (known after apply)

+ egress = (known after apply)

+ id = (known after apply)

+ ingress = (known after apply)

+ name = (known after apply)

+ name\_prefix = (known after apply)

+ owner\_id = (known after apply)

+ revoke\_rules\_on\_delete = false

+ tags = {

+ "Environment" = "staging"

+ "Name" = "main-default"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "main-default"

}

+ vpc\_id = (known after apply)

}

# module.vpc.aws\_eip.nat[0] will be created

+ resource "aws\_eip" "nat" {

+ allocation\_id = (known after apply)

+ association\_id = (known after apply)

+ carrier\_ip = (known after apply)

+ customer\_owned\_ip = (known after apply)

+ domain = "vpc"

+ id = (known after apply)

+ instance = (known after apply)

+ network\_border\_group = (known after apply)

+ network\_interface = (known after apply)

+ private\_dns = (known after apply)

+ private\_ip = (known after apply)

+ public\_dns = (known after apply)

+ public\_ip = (known after apply)

+ public\_ipv4\_pool = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "main-us-east-1a"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "main-us-east-1a"

}

+ vpc = (known after apply)

}

# module.vpc.aws\_internet\_gateway.this[0] will be created

+ resource "aws\_internet\_gateway" "this" {

+ arn = (known after apply)

+ id = (known after apply)

+ owner\_id = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "main"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "main"

}

+ vpc\_id = (known after apply)

}

# module.vpc.aws\_nat\_gateway.this[0] will be created

+ resource "aws\_nat\_gateway" "this" {

+ allocation\_id = (known after apply)

+ association\_id = (known after apply)

+ connectivity\_type = "public"

+ id = (known after apply)

+ network\_interface\_id = (known after apply)

+ private\_ip = (known after apply)

+ public\_ip = (known after apply)

+ subnet\_id = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "main-us-east-1a"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "main-us-east-1a"

}

}

# module.vpc.aws\_route.private\_nat\_gateway[0] will be created

+ resource "aws\_route" "private\_nat\_gateway" {

+ destination\_cidr\_block = "0.0.0.0/0"

+ id = (known after apply)

+ instance\_id = (known after apply)

+ instance\_owner\_id = (known after apply)

+ nat\_gateway\_id = (known after apply)

+ network\_interface\_id = (known after apply)

+ origin = (known after apply)

+ route\_table\_id = (known after apply)

+ state = (known after apply)

+ timeouts {

+ create = "5m"

}

}

# module.vpc.aws\_route.public\_internet\_gateway[0] will be created

+ resource "aws\_route" "public\_internet\_gateway" {

+ destination\_cidr\_block = "0.0.0.0/0"

+ gateway\_id = (known after apply)

+ id = (known after apply)

+ instance\_id = (known after apply)

+ instance\_owner\_id = (known after apply)

+ network\_interface\_id = (known after apply)

+ origin = (known after apply)

+ route\_table\_id = (known after apply)

+ state = (known after apply)

+ timeouts {

+ create = "5m"

}

}

# module.vpc.aws\_route\_table.private[0] will be created

+ resource "aws\_route\_table" "private" {

+ arn = (known after apply)

+ id = (known after apply)

+ owner\_id = (known after apply)

+ propagating\_vgws = (known after apply)

+ route = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "main-private"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "main-private"

}

+ vpc\_id = (known after apply)

}

# module.vpc.aws\_route\_table.public[0] will be created

+ resource "aws\_route\_table" "public" {

+ arn = (known after apply)

+ id = (known after apply)

+ owner\_id = (known after apply)

+ propagating\_vgws = (known after apply)

+ route = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "main-public"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "main-public"

}

+ vpc\_id = (known after apply)

}

# module.vpc.aws\_route\_table\_association.private[0] will be created

+ resource "aws\_route\_table\_association" "private" {

+ id = (known after apply)

+ route\_table\_id = (known after apply)

+ subnet\_id = (known after apply)

}

# module.vpc.aws\_route\_table\_association.private[1] will be created

+ resource "aws\_route\_table\_association" "private" {

+ id = (known after apply)

+ route\_table\_id = (known after apply)

+ subnet\_id = (known after apply)

}

# module.vpc.aws\_route\_table\_association.public[0] will be created

+ resource "aws\_route\_table\_association" "public" {

+ id = (known after apply)

+ route\_table\_id = (known after apply)

+ subnet\_id = (known after apply)

}

# module.vpc.aws\_route\_table\_association.public[1] will be created

+ resource "aws\_route\_table\_association" "public" {

+ id = (known after apply)

+ route\_table\_id = (known after apply)

+ subnet\_id = (known after apply)

}

# module.vpc.aws\_subnet.private[0] will be created

+ resource "aws\_subnet" "private" {

+ arn = (known after apply)

+ assign\_ipv6\_address\_on\_creation = false

+ availability\_zone = "us-east-1a"

+ availability\_zone\_id = (known after apply)

+ cidr\_block = "10.0.0.0/19"

+ enable\_dns64 = false

+ enable\_resource\_name\_dns\_a\_record\_on\_launch = false

+ enable\_resource\_name\_dns\_aaaa\_record\_on\_launch = false

+ id = (known after apply)

+ ipv6\_cidr\_block\_association\_id = (known after apply)

+ ipv6\_native = false

+ map\_public\_ip\_on\_launch = false

+ owner\_id = (known after apply)

+ private\_dns\_hostname\_type\_on\_launch = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "main-private-us-east-1a"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "main-private-us-east-1a"

}

+ vpc\_id = (known after apply)

}

# module.vpc.aws\_subnet.private[1] will be created

+ resource "aws\_subnet" "private" {

+ arn = (known after apply)

+ assign\_ipv6\_address\_on\_creation = false

+ availability\_zone = "us-east-1b"

+ availability\_zone\_id = (known after apply)

+ cidr\_block = "10.0.32.0/19"

+ enable\_dns64 = false

+ enable\_resource\_name\_dns\_a\_record\_on\_launch = false

+ enable\_resource\_name\_dns\_aaaa\_record\_on\_launch = false

+ id = (known after apply)

+ ipv6\_cidr\_block\_association\_id = (known after apply)

+ ipv6\_native = false

+ map\_public\_ip\_on\_launch = false

+ owner\_id = (known after apply)

+ private\_dns\_hostname\_type\_on\_launch = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "main-private-us-east-1b"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "main-private-us-east-1b"

}

+ vpc\_id = (known after apply)

}

# module.vpc.aws\_subnet.public[0] will be created

+ resource "aws\_subnet" "public" {

+ arn = (known after apply)

+ assign\_ipv6\_address\_on\_creation = false

+ availability\_zone = "us-east-1a"

+ availability\_zone\_id = (known after apply)

+ cidr\_block = "10.0.64.0/19"

+ enable\_dns64 = false

+ enable\_resource\_name\_dns\_a\_record\_on\_launch = false

+ enable\_resource\_name\_dns\_aaaa\_record\_on\_launch = false

+ id = (known after apply)

+ ipv6\_cidr\_block\_association\_id = (known after apply)

+ ipv6\_native = false

+ map\_public\_ip\_on\_launch = false

+ owner\_id = (known after apply)

+ private\_dns\_hostname\_type\_on\_launch = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "main-public-us-east-1a"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "main-public-us-east-1a"

}

+ vpc\_id = (known after apply)

}

# module.vpc.aws\_subnet.public[1] will be created

+ resource "aws\_subnet" "public" {

+ arn = (known after apply)

+ assign\_ipv6\_address\_on\_creation = false

+ availability\_zone = "us-east-1b"

+ availability\_zone\_id = (known after apply)

+ cidr\_block = "10.0.96.0/19"

+ enable\_dns64 = false

+ enable\_resource\_name\_dns\_a\_record\_on\_launch = false

+ enable\_resource\_name\_dns\_aaaa\_record\_on\_launch = false

+ id = (known after apply)

+ ipv6\_cidr\_block\_association\_id = (known after apply)

+ ipv6\_native = false

+ map\_public\_ip\_on\_launch = false

+ owner\_id = (known after apply)

+ private\_dns\_hostname\_type\_on\_launch = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "main-public-us-east-1b"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "main-public-us-east-1b"

}

+ vpc\_id = (known after apply)

}

# module.vpc.aws\_vpc.this[0] will be created

+ resource "aws\_vpc" "this" {

+ arn = (known after apply)

+ cidr\_block = "10.0.0.0/16"

+ default\_network\_acl\_id = (known after apply)

+ default\_route\_table\_id = (known after apply)

+ default\_security\_group\_id = (known after apply)

+ dhcp\_options\_id = (known after apply)

+ enable\_dns\_hostnames = true

+ enable\_dns\_support = true

+ enable\_network\_address\_usage\_metrics = (known after apply)

+ id = (known after apply)

+ instance\_tenancy = "default"

+ ipv6\_association\_id = (known after apply)

+ ipv6\_cidr\_block = (known after apply)

+ ipv6\_cidr\_block\_network\_border\_group = (known after apply)

+ main\_route\_table\_id = (known after apply)

+ owner\_id = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "main"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "main"

}

}

# module.eks.module.eks\_managed\_node\_group["general"].aws\_eks\_node\_group.this[0] will be created

+ resource "aws\_eks\_node\_group" "this" {

+ ami\_type = (known after apply)

+ arn = (known after apply)

+ capacity\_type = "ON\_DEMAND"

+ cluster\_name = "my-eks"

+ disk\_size = (known after apply)

+ id = (known after apply)

+ instance\_types = [

+ "t3.small",

]

+ labels = {

+ "role" = "general"

}

+ node\_group\_name = (known after apply)

+ node\_group\_name\_prefix = "general-"

+ node\_role\_arn = (known after apply)

+ release\_version = (known after apply)

+ resources = (known after apply)

+ status = (known after apply)

+ subnet\_ids = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "general"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "general"

}

+ version = "1.23"

+ launch\_template {

+ id = (known after apply)

+ name = (known after apply)

+ version = (known after apply)

}

+ scaling\_config {

+ desired\_size = 1

+ max\_size = 10

+ min\_size = 1

}

+ timeouts {}

}

# module.eks.module.eks\_managed\_node\_group["general"].aws\_iam\_role.this[0] will be created

+ resource "aws\_iam\_role" "this" {

+ arn = (known after apply)

+ assume\_role\_policy = jsonencode(

{

+ Statement = [

+ {

+ Action = "sts:AssumeRole"

+ Effect = "Allow"

+ Principal = {

+ Service = "ec2.amazonaws.com"

}

+ Sid = "EKSNodeAssumeRole"

},

]

+ Version = "2012-10-17"

}

)

+ create\_date = (known after apply)

+ description = "EKS managed node group IAM role"

+ force\_detach\_policies = true

+ id = (known after apply)

+ managed\_policy\_arns = (known after apply)

+ max\_session\_duration = 3600

+ name = (known after apply)

+ name\_prefix = "general-eks-node-group-"

+ path = "/"

+ tags = {

+ "Environment" = "staging"

}

+ tags\_all = {

+ "Environment" = "staging"

}

+ unique\_id = (known after apply)

}

# module.eks.module.eks\_managed\_node\_group["general"].aws\_iam\_role\_policy\_attachment.this["arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"] will be created

+ resource "aws\_iam\_role\_policy\_attachment" "this" {

+ id = (known after apply)

+ policy\_arn = "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"

+ role = (known after apply)

}

# module.eks.module.eks\_managed\_node\_group["general"].aws\_iam\_role\_policy\_attachment.this["arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"] will be created

+ resource "aws\_iam\_role\_policy\_attachment" "this" {

+ id = (known after apply)

+ policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"

+ role = (known after apply)

}

# module.eks.module.eks\_managed\_node\_group["general"].aws\_iam\_role\_policy\_attachment.this["arn:aws:iam::aws:policy/AmazonEKS\_CNI\_Policy"] will be created

+ resource "aws\_iam\_role\_policy\_attachment" "this" {

+ id = (known after apply)

+ policy\_arn = "arn:aws:iam::aws:policy/AmazonEKS\_CNI\_Policy"

+ role = (known after apply)

}

# module.eks.module.eks\_managed\_node\_group["general"].aws\_launch\_template.this[0] will be created

+ resource "aws\_launch\_template" "this" {

+ arn = (known after apply)

+ default\_version = (known after apply)

+ description = "Custom launch template for general EKS managed node group"

+ id = (known after apply)

+ latest\_version = (known after apply)

+ name = (known after apply)

+ name\_prefix = "general-"

+ tags = {

+ "Environment" = "staging"

}

+ tags\_all = {

+ "Environment" = "staging"

}

+ update\_default\_version = true

+ vpc\_security\_group\_ids = (known after apply)

+ metadata\_options {

+ http\_endpoint = "enabled"

+ http\_protocol\_ipv6 = (known after apply)

+ http\_put\_response\_hop\_limit = 2

+ http\_tokens = "required"

+ instance\_metadata\_tags = (known after apply)

}

+ monitoring {

+ enabled = true

}

+ tag\_specifications {

+ resource\_type = "instance"

+ tags = {

+ "Environment" = "staging"

+ "Name" = "general"

}

}

+ tag\_specifications {

+ resource\_type = "network-interface"

+ tags = {

+ "Environment" = "staging"

+ "Name" = "general"

}

}

+ tag\_specifications {

+ resource\_type = "volume"

+ tags = {

+ "Environment" = "staging"

+ "Name" = "general"

}

}

}

# module.eks.module.eks\_managed\_node\_group["general"].aws\_security\_group.this[0] will be created

+ resource "aws\_security\_group" "this" {

+ arn = (known after apply)

+ description = "EKS managed node group security group"

+ egress = (known after apply)

+ id = (known after apply)

+ ingress = (known after apply)

+ name = (known after apply)

+ name\_prefix = "general-eks-node-group-"

+ owner\_id = (known after apply)

+ revoke\_rules\_on\_delete = false

+ tags = {

+ "Environment" = "staging"

+ "Name" = "general-eks-node-group"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "general-eks-node-group"

}

+ vpc\_id = (known after apply)

}

# module.eks.module.eks\_managed\_node\_group["spot"].aws\_eks\_node\_group.this[0] will be created

+ resource "aws\_eks\_node\_group" "this" {

+ ami\_type = (known after apply)

+ arn = (known after apply)

+ capacity\_type = "SPOT"

+ cluster\_name = "my-eks"

+ disk\_size = (known after apply)

+ id = (known after apply)

+ instance\_types = [

+ "t3.micro",

]

+ labels = {

+ "role" = "spot"

}

+ node\_group\_name = (known after apply)

+ node\_group\_name\_prefix = "spot-"

+ node\_role\_arn = (known after apply)

+ release\_version = (known after apply)

+ resources = (known after apply)

+ status = (known after apply)

+ subnet\_ids = (known after apply)

+ tags = {

+ "Environment" = "staging"

+ "Name" = "spot"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "spot"

}

+ version = "1.23"

+ launch\_template {

+ id = (known after apply)

+ name = (known after apply)

+ version = (known after apply)

}

+ scaling\_config {

+ desired\_size = 1

+ max\_size = 10

+ min\_size = 1

}

+ taint {

+ effect = "NO\_SCHEDULE"

+ key = "market"

+ value = "spot"

}

+ timeouts {}

}

# module.eks.module.eks\_managed\_node\_group["spot"].aws\_iam\_role.this[0] will be created

+ resource "aws\_iam\_role" "this" {

+ arn = (known after apply)

+ assume\_role\_policy = jsonencode(

{

+ Statement = [

+ {

+ Action = "sts:AssumeRole"

+ Effect = "Allow"

+ Principal = {

+ Service = "ec2.amazonaws.com"

}

+ Sid = "EKSNodeAssumeRole"

},

]

+ Version = "2012-10-17"

}

)

+ create\_date = (known after apply)

+ description = "EKS managed node group IAM role"

+ force\_detach\_policies = true

+ id = (known after apply)

+ managed\_policy\_arns = (known after apply)

+ max\_session\_duration = 3600

+ name = (known after apply)

+ name\_prefix = "spot-eks-node-group-"

+ path = "/"

+ tags = {

+ "Environment" = "staging"

}

+ tags\_all = {

+ "Environment" = "staging"

}

+ unique\_id = (known after apply)

}

# module.eks.module.eks\_managed\_node\_group["spot"].aws\_iam\_role\_policy\_attachment.this["arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"] will be created

+ resource "aws\_iam\_role\_policy\_attachment" "this" {

+ id = (known after apply)

+ policy\_arn = "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"

+ role = (known after apply)

}

# module.eks.module.eks\_managed\_node\_group["spot"].aws\_iam\_role\_policy\_attachment.this["arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"] will be created

+ resource "aws\_iam\_role\_policy\_attachment" "this" {

+ id = (known after apply)

+ policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"

+ role = (known after apply)

}

# module.eks.module.eks\_managed\_node\_group["spot"].aws\_iam\_role\_policy\_attachment.this["arn:aws:iam::aws:policy/AmazonEKS\_CNI\_Policy"] will be created

+ resource "aws\_iam\_role\_policy\_attachment" "this" {

+ id = (known after apply)

+ policy\_arn = "arn:aws:iam::aws:policy/AmazonEKS\_CNI\_Policy"

+ role = (known after apply)

}

# module.eks.module.eks\_managed\_node\_group["spot"].aws\_launch\_template.this[0] will be created

+ resource "aws\_launch\_template" "this" {

+ arn = (known after apply)

+ default\_version = (known after apply)

+ description = "Custom launch template for spot EKS managed node group"

+ id = (known after apply)

+ latest\_version = (known after apply)

+ name = (known after apply)

+ name\_prefix = "spot-"

+ tags = {

+ "Environment" = "staging"

}

+ tags\_all = {

+ "Environment" = "staging"

}

+ update\_default\_version = true

+ vpc\_security\_group\_ids = (known after apply)

+ metadata\_options {

+ http\_endpoint = "enabled"

+ http\_protocol\_ipv6 = (known after apply)

+ http\_put\_response\_hop\_limit = 2

+ http\_tokens = "required"

+ instance\_metadata\_tags = (known after apply)

}

+ monitoring {

+ enabled = true

}

+ tag\_specifications {

+ resource\_type = "instance"

+ tags = {

+ "Environment" = "staging"

+ "Name" = "spot"

}

}

+ tag\_specifications {

+ resource\_type = "network-interface"

+ tags = {

+ "Environment" = "staging"

+ "Name" = "spot"

}

}

+ tag\_specifications {

+ resource\_type = "volume"

+ tags = {

+ "Environment" = "staging"

+ "Name" = "spot"

}

}

}

# module.eks.module.eks\_managed\_node\_group["spot"].aws\_security\_group.this[0] will be created

+ resource "aws\_security\_group" "this" {

+ arn = (known after apply)

+ description = "EKS managed node group security group"

+ egress = (known after apply)

+ id = (known after apply)

+ ingress = (known after apply)

+ name = (known after apply)

+ name\_prefix = "spot-eks-node-group-"

+ owner\_id = (known after apply)

+ revoke\_rules\_on\_delete = false

+ tags = {

+ "Environment" = "staging"

+ "Name" = "spot-eks-node-group"

}

+ tags\_all = {

+ "Environment" = "staging"

+ "Name" = "spot-eks-node-group"

}

+ vpc\_id = (known after apply)

}

Plan: 58 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: