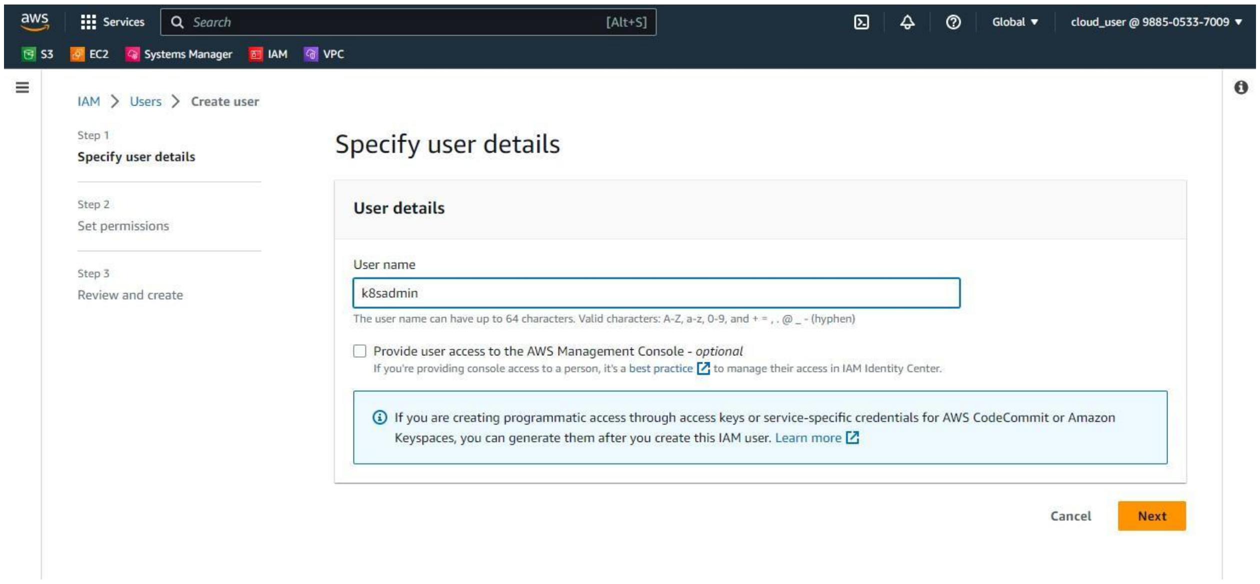
# Creando un pacman en k8s

Primero creamos una cuenta de iam con acceso programatico.



Specify user details

User details

User name

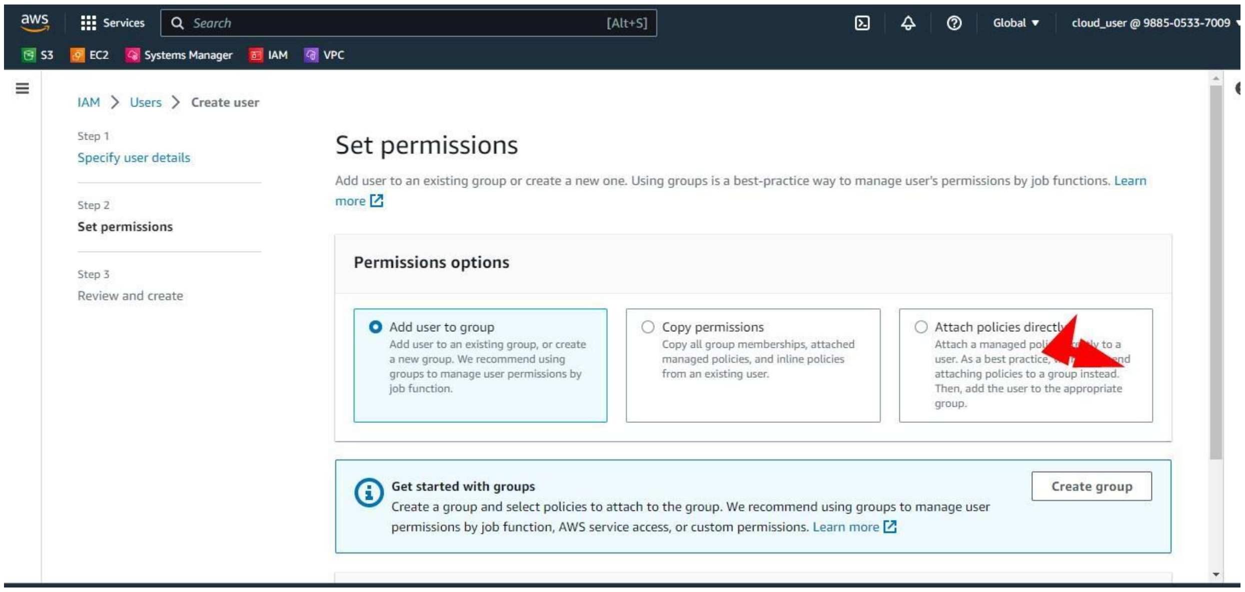
k8sadmin

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

☐ Provide user access to the AWS Management Console - *optional*
If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more

Ahora pedimos elegir una policy



Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more

Permissions options

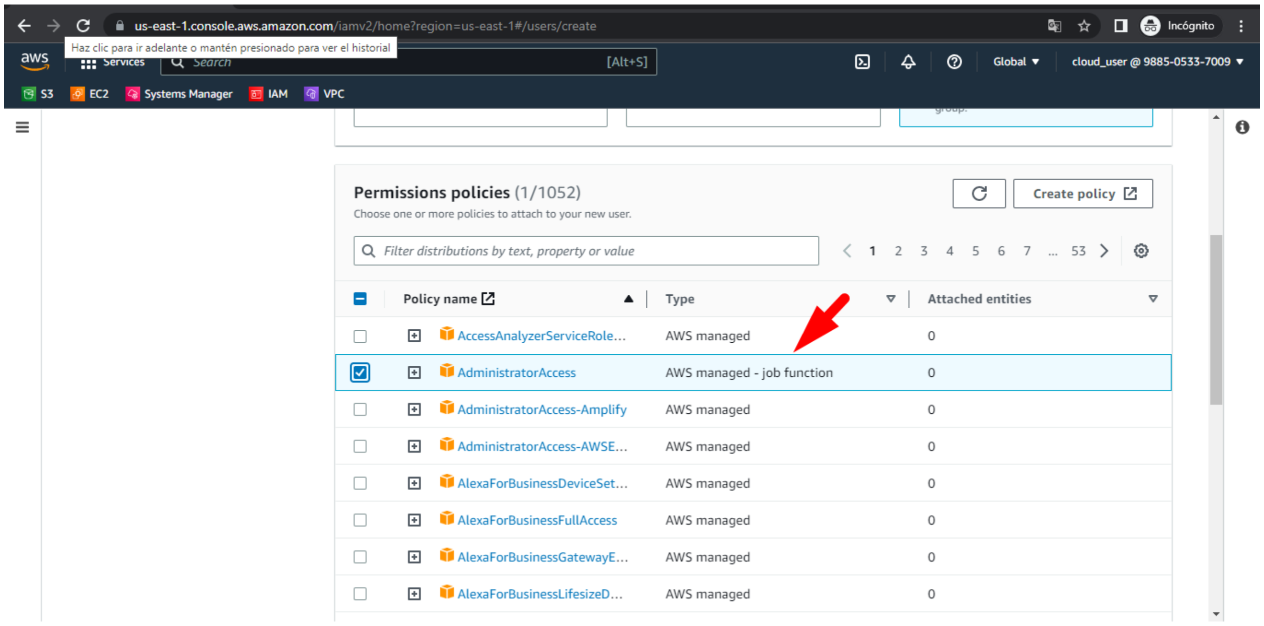
● Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

○ Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

○ Attach policies directly
Attach a managed pol... to a user. As a best practice, ... attaching policies to a group instead. Then, add the user to the appropriate group.

Get started with groups
Create a group and select policies to attach to the group. We recommend using groups to manage user permissions by job function, AWS service access, or custom permissions. Learn more

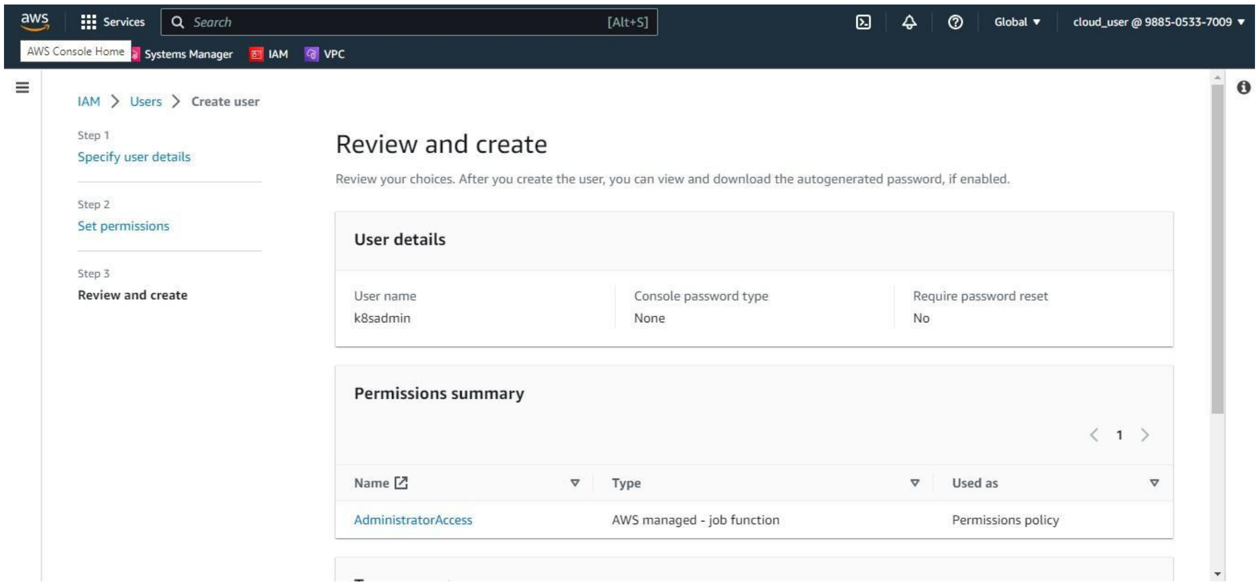
Le damos los permisos


Permissions policies (1/1052)

Choose one or more policies to attach to your new user.

| | Policy name | Type | Attached entities |
|---|---|---|---|
| ☐ | AccessAnalyzerServiceRole... | AWS managed | 0 |
| ✓ | AdministratorAccess | AWS managed - job function | 0 |
| ☐ | AdministratorAccess-Amplify | AWS managed | 0 |
| ☐ | AdministratorAccess-AWSE... | AWS managed | 0 |
| ☐ | AlexaForBusinessDeviceSet... | AWS managed | 0 |
| ☐ | AlexaForBusinessFullAccess | AWS managed | 0 |
| ☐ | AlexaForBusinessGatewayE... | AWS managed | 0 |
| ☐ | AlexaForBusinessLifesizeD... | AWS managed | 0 |

Ahora


Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

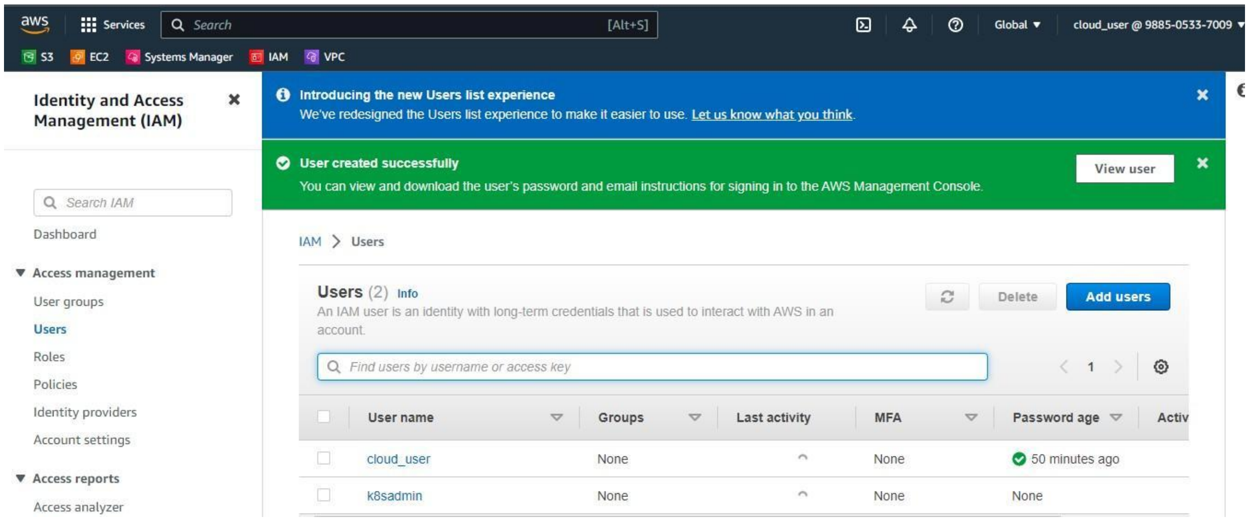
| User name | Console password type | Require password reset |
|---|---|---|
| k8sadmin | None | No |

Permissions summary

| Name | Type | Used as |
|---|---|---|
| AdministratorAccess | AWS managed - job function | Permissions policy |

Al acceso programático lo seteamos más adelante.

Luego de crear el user debería crear así....



**Introducing the new Users list experience**
We've redesigned the Users list experience to make it easier to use. Let us know what you think.

**User created successfully**
You can view and download the user's password and email instructions for signing in to the AWS Management Console.

View user

IAM > Users

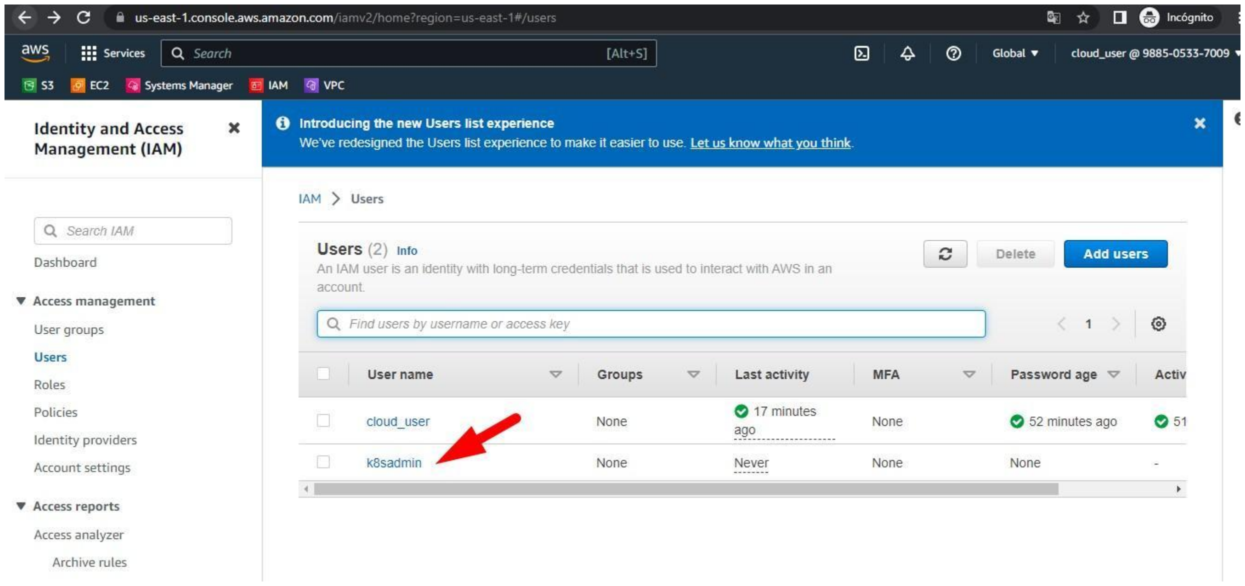
**Users** (2) Info
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Delete | Add users

Find users by username or access key

| User name | Groups | Last activity | MFA | Password age | Activ |
|---|---|---|---|---|---|
| cloud_user | None | | None | ✓ 50 minutes ago | |
| k8sadmin | None | | None | None | |

Seleccionamos ahí para darle el acceso programático



**Introducing the new Users list experience**
We've redesigned the Users list experience to make it easier to use. Let us know what you think.

IAM > Users

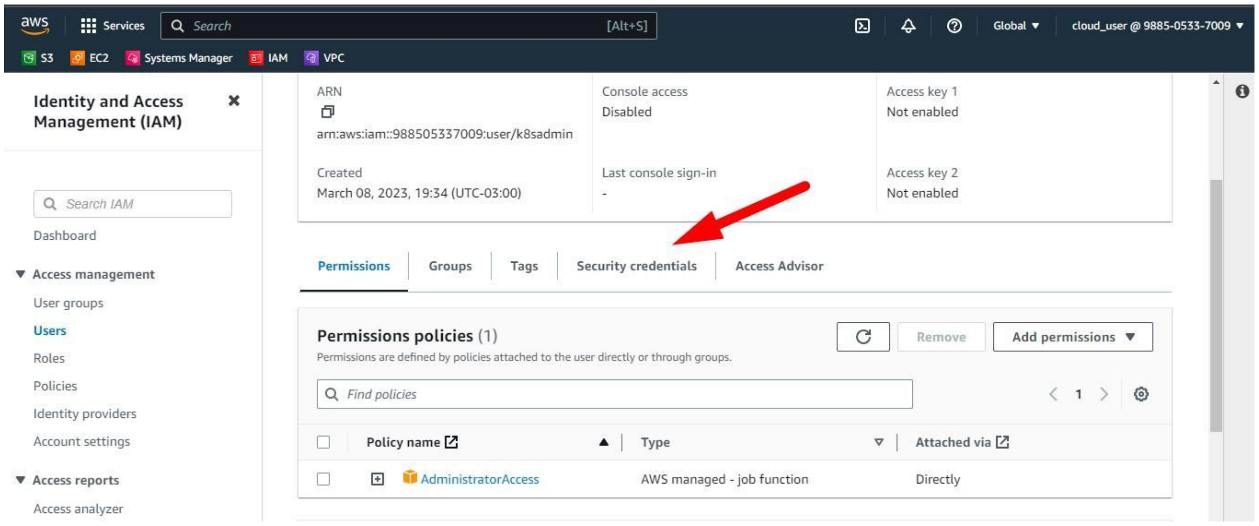
**Users** (2) Info
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Delete | Add users

Find users by username or access key

| User name | Groups | Last activity | MFA | Password age | Activ |
|---|---|---|---|---|---|
| cloud_user | None | ✓ 17 minutes ago | None | ✓ 52 minutes ago | ✓ 51 |
| k8sadmin | None | Never | None | None | - |

Luego:

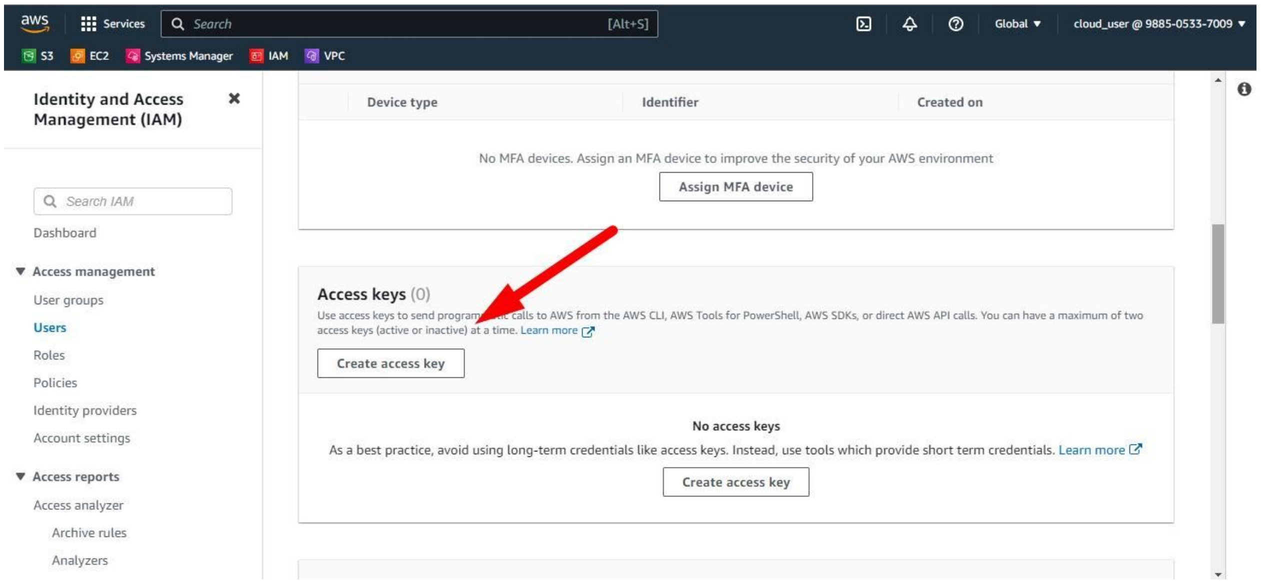


| ARN | Console access | Access key 1 |
|---|---|---|
| arn:aws:iam::988505337009:user/k8sadmin | Disabled | Not enabled |
| Created | Last console sign-in | Access key 2 |
| March 08, 2023, 19:34 (UTC-03:00) | - | Not enabled |

Permissions policies (1)
Permissions are defined by policies attached to the user directly or through groups.

| Policy name | Type | Attached via |
|---|---|---|
| AdministratorAccess | AWS managed - job function | Directly |

Le damos en access key

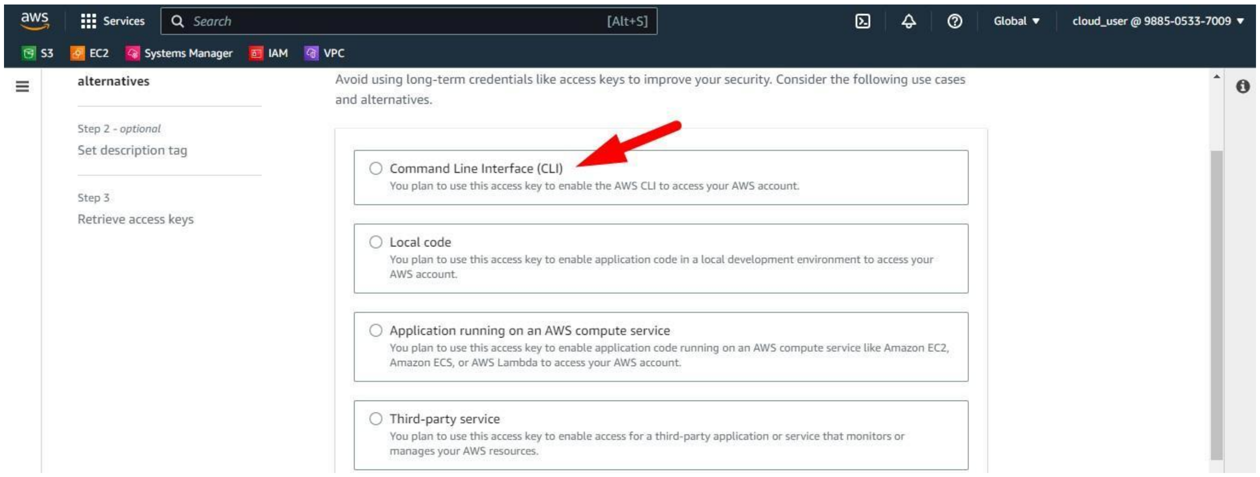


No MFA devices. Assign an MFA device to improve the security of your AWS environment

Access keys (0)
Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

No access keys
As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more

Luego



alternatives

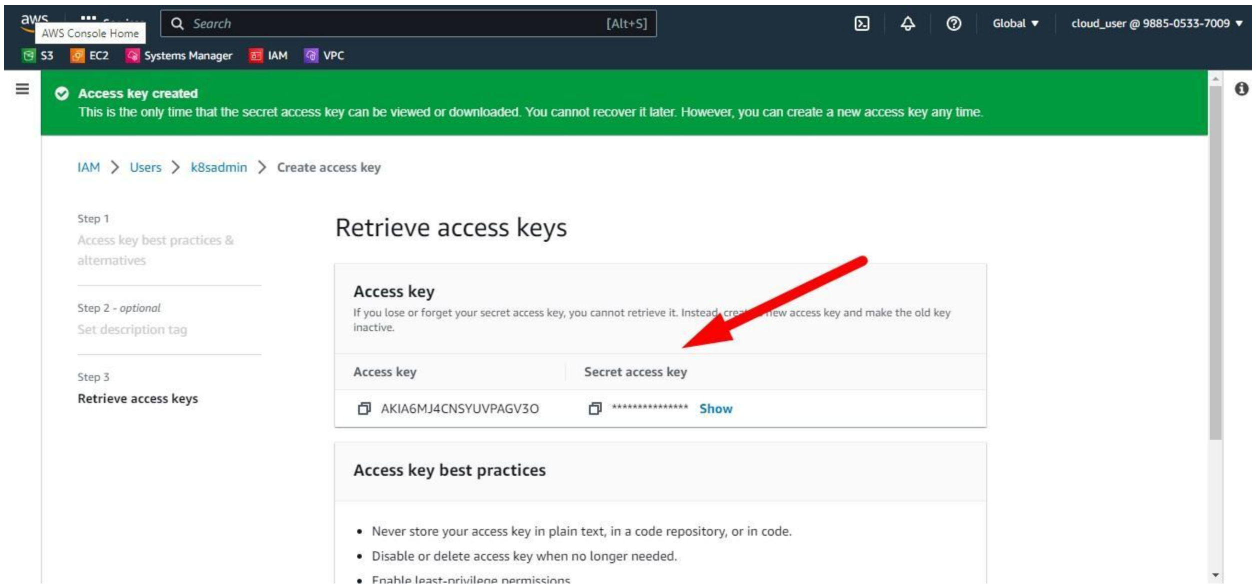
Step 2 - *optional*
Set description tag

Step 3
Retrieve access keys

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

- ○ Command Line Interface (CLI)
  You plan to use this access key to enable the AWS CLI to access your AWS account.
- ○ Local code
  You plan to use this access key to enable application code in a local development environment to access your AWS account.
- ○ Application running on an AWS compute service
  You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- ○ Third-party service
  You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

aceptamos los dos checkbox y nos aparecen las credentiales



**Access key created**
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

IAM > Users > k8sadmin > Create access key

Step 1
Access key best practices & alternatives

Step 2 - *optional*
Set description tag

Step 3
**Retrieve access keys**

Retrieve access keys

**Access key**
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

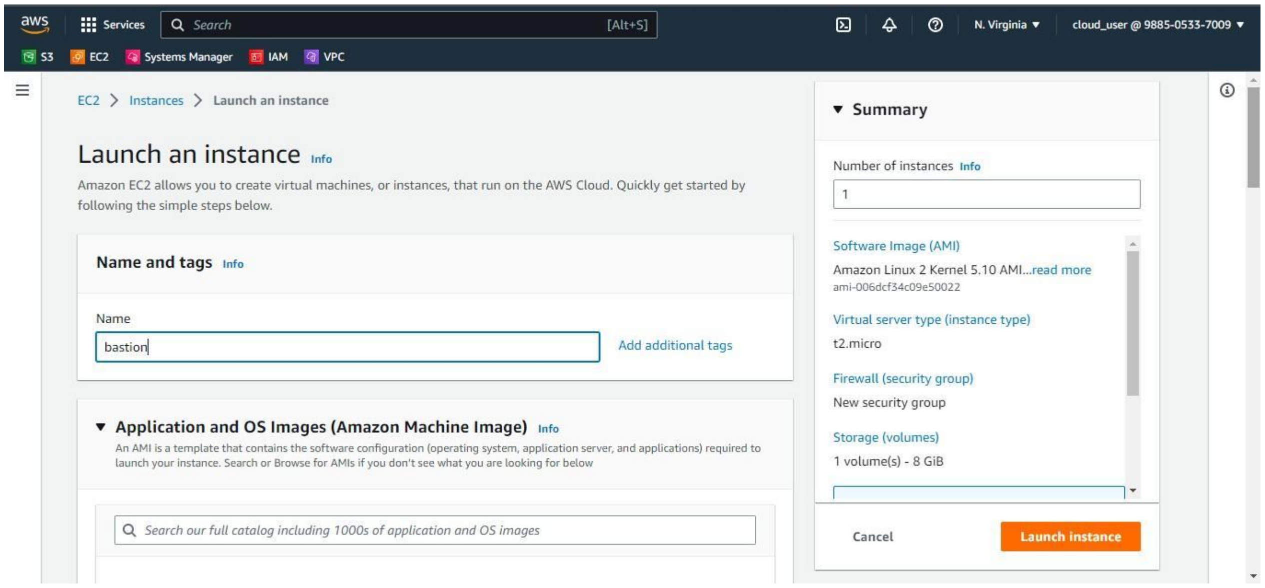
| Access key | Secret access key |
|---|---|
| AKIA6MJ4CNSYUVPAGV3O | *************** Show |

**Access key best practices**

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions

Las copiamos.

Creamos una instancia.


EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: bastion

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Summary

Number of instances: 1

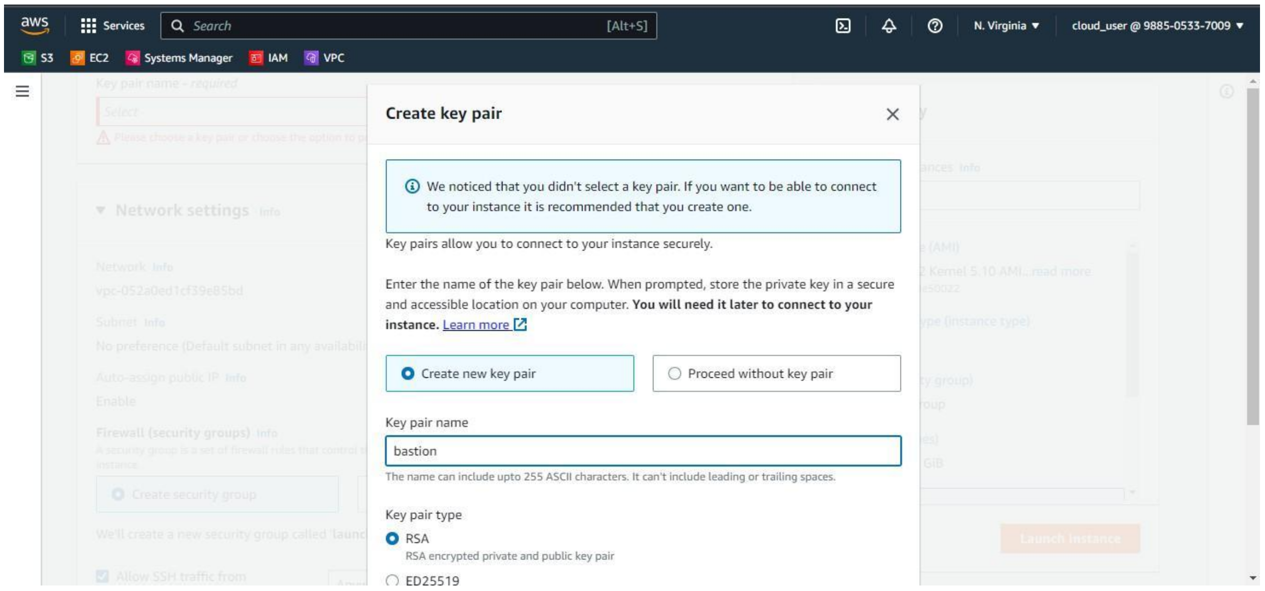
Software Image (AMI): Amazon Linux 2 Kernel 5.10 AMI...read more ami-006dcf34c09e50022

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Generamos el par key (bastion.pem)


Create key pair

We noticed that you didn't select a key pair. If you want to be able to connect to your instance it is recommended that you create one.

Key pairs allow you to connect to your instance securely.

Enter the name of the key pair below. When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more

[✓ Create new key pair] Proceed without key pair

Key pair name: bastion

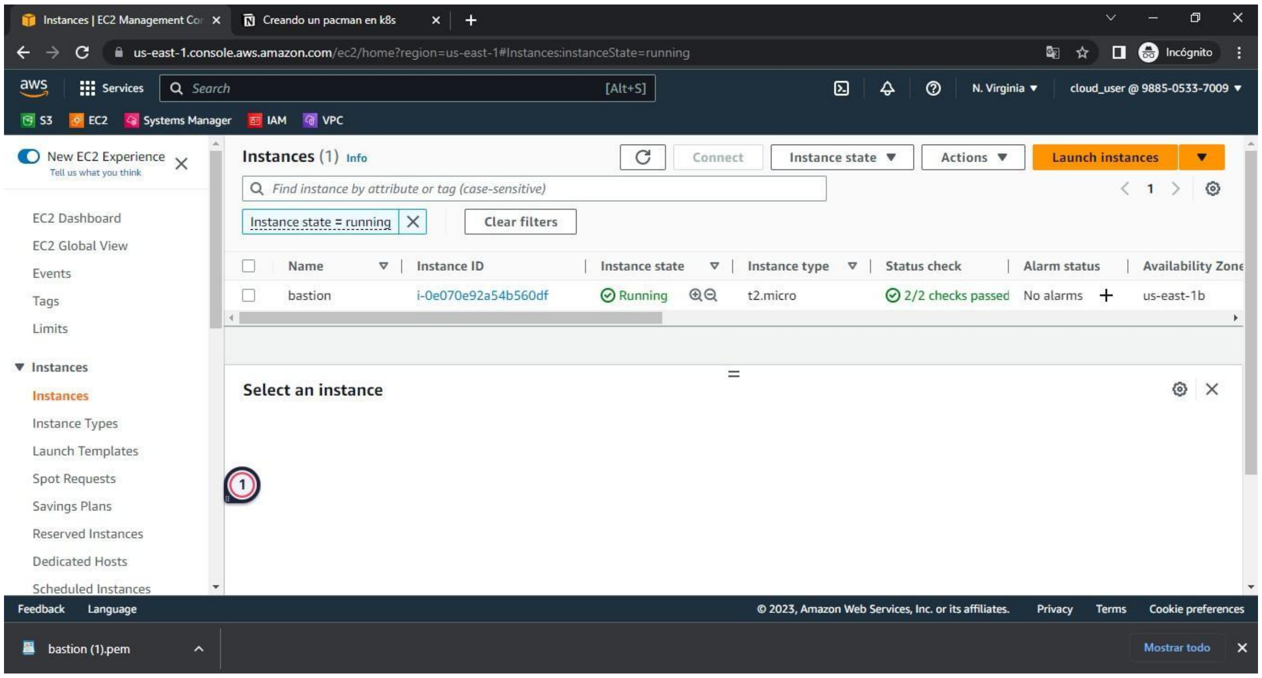
The name can include upto 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

[✓ RSA] RSA encrypted private and public key pair

ED25519

Esperamos que se cree y entramos.



| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone |
|---|---|---|---|---|---|---|
| bastion | i-0e070e92a54b560df | Running | t2.micro | 2/2 checks passed | No alarms | us-east-1b |

Ahora nos logueamos con la clave generada

```
ssh -i bastion.pem ec2-user@54.90.140.133
Warning: Permanently added '54.90.140.133' (RSA) to the list of known hosts.
X11 forwarding request failed on channel 0

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
4 package(s) needed for security, out of 6
available Run "sudo yum update" to apply all
updates.
```

y empezamos a instalar las tooles que necesitamos....

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip" && unzip awscliv2.zip
```

Luego

```
sudo ./aws/install --bin-dir /usr/bin --install-dir /usr/bin/aws-cli --update
 aws --version
```

Luego

Configuramos aws con la data programatica

```
aws
configure
```

Ahora bajamos kubectl

```
curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.16.8/2020-04-16/bin/linux/amd64/kubectl
```

Lo seteamos

```
chmod +x ./kubectl
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin
kubectl version
```

Bajamos las configs que necesitamos

```
wget https://github.com/pluralsight-cloud/content-deploying-and-managing-a-web-application-in-kubernetes-with-terraform/raw/main/eks.zip
```

descomprimimos

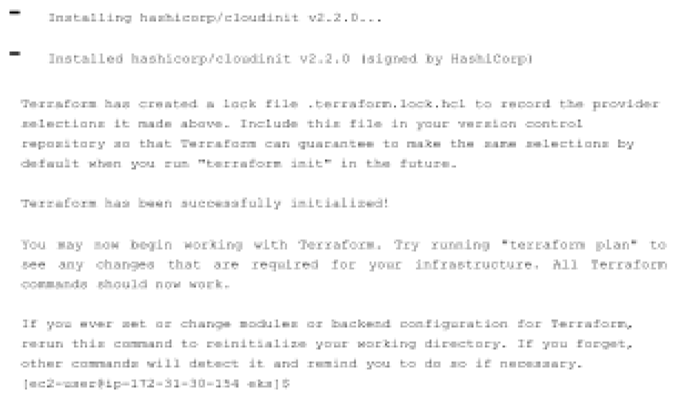
```
[ec2-user@ip-172-31-30-154 ~]$ unzip
eks.zip Archive: eks.zip
   creating: eks/
  inflating:
  eks/eks-cluster.tf
  inflating: eks/main.tf
  inflating: eks/outputs.tf
  inflating:
  eks/terraform.tf
  inflating:
  eks/variables.tf
  inflating: eks/vpc.tf
```

Luego instalar el cliente terrarform

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
sudo yum -y install terraform git
```

y finalmente la secuencia de siempre....

```
]ec2-user@ip-172-31-58-158 eks]# terraform
init Initializing modules...
Downloading registry.terraform.io/terraform-aws-modules/eks/aws 18.5.4 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 1.1.0 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 3.14.2 for vpc...

-   vpc in


    .terraform/modules/vpc

    Initializing the

    backend...

Initializing provider plugins...

-   Finding hashicorp/kubernetes versions matching ">= 2.10.0, ~> 2.16.1"...
- Finding hashicorp/aws versions matching ">= 3.63.0, >= 3.72.0, >= 4.45.0, ~> 4.46.0"...

-   Finding hashicorp/random versions matching "~> 3.4.3"...

-   Finding hashicorp/tls versions matching ">= 3.0.0, ~> 4.0.4"...

-   Finding hashicorp/cloudinit versions matching ">= 2.0.0, ~> 2.2.0"...

-   Installing hashicorp/kubernetes v2.16.1...

-   Installed hashicorp/kubernetes v2.16.1 (signed by HashiCorp)

-   Installing hashicorp/aws v4.46.0...
```



```
-   Installing hashicorp/cloudinit v2.2.0...
-   Installed hashicorp/cloudinit v2.2.0 (signed by HashiCorp)

 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.
 [ec2-user@ip-172-31-30-154 eks]$
```

Lanzamos el terraform plan

```
[ec2-user@ip-172-31-30-154 eks]$ terraform plan data.aws_availability_zones.available:
Reading... module.eks.module.eks_managed_node_group["one"].data.aws_partition.current:
Reading... module.eks.data.aws_partition.current: Reading...
module.eks.module.eks_managed_node_group["two"].data.aws_partition.current: Reading...
module.eks.module.eks_managed_node_group["two"].data.aws_caller_identity.current: Reading...
module.eks.module.kms.data.aws_caller_identity.current: Reading...
module.eks.module.kms.data.aws_partition.current: Reading...
module.eks.data.aws_caller_identity.current: Reading... module.eks.data.aws_partition.current:
Read complete after 0s [id=aws]
module.eks.module.eks_managed_node_group["two"].data.aws_partition.current: Read complete after 0s [id=aws]
module.eks.module.eks_managed_node_group["one"].data.aws_partition.current: Read complete after 0s [id=aws]
module.eks.module.kms.data.aws_partition.current: Read complete after 0s [id=aws]
module.eks.module.eks_managed_node_group["two"].data.aws_caller_identity.current: Read complete after 0s [id=988505337009]
module.eks.module.eks_managed_node_group["one"].data.aws_iam_policy_document.assume_role_policy[0]: Reading...
module.eks.module.eks_managed_node_group["one"].data.aws_caller_identity.current: Reading...
module.eks.module.eks_managed_node_group["two"].data.aws_iam_policy_document.assume_role_policy[0]: Reading...
module.eks.data.aws_iam_policy_document.assume_role_policy[0]: Reading...
module.eks.data.aws_caller_identity.current: Read complete after 0s [id=988505337009] module.eks.module.kms.data.aws_caller_identity.current: Read
complete after 0s [id=988505337009] module.eks.module.eks_managed_node_group["one"].data.aws_caller_identity.current: Read complete after 0s
[id=988505337009] module.eks.data.aws_iam_policy_document.assume_role_policy[0]: Read complete after 0s [id=2764486067]
module.eks.module.eks_managed_node_group["one"].data.aws_iam_policy_document.assume_role_policy[0]: Read complete after 0s [id=2560088296]
module.eks.module.eks_managed_node_group["two"].data.aws_iam_policy_document.assume_role_policy[0]: Read complete after 0s [id=2560088296]
data.aws_availability_zones.available: Read complete after 0s [id=us-east-1]

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: #
  random_string.suffix will be created
  + resource "random_string" "suffix" {
      + id          = (known after apply)
      + length      = 8
      + lower       = true
      + min_lower = 0
      + min_numeric = 0
      + min_special = 0
      + min_upper = 0
      + number      = true
      + numeric     = true
      + result      = (known after apply)
      + special     = false
      + upper       = true
    }

  # 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              = (known after apply)
      + name_prefix       = (known after apply)
      + retention_in_days = 90
      + skip_destroy      = false
      + tags_all          = (known after apply)
    }

  # 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                      = (known after apply)
      + platform_version          = (known after apply)
      + role_arn                  = (known after apply)
      + status                    = (known after apply)
      + tags_all                  = (known after apply)
      + version                   = "1.24"

      + encryption_config {
          + resources = [
              + "secrets",
            ]

          + provider {
              + key_arn = (known after apply)
            }
        }

      + 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            = (known after apply)
      + tags_all        = (known after apply)
      + thumbprint_list = (known after apply)
      + url             = (known after apply)
    }

  # module.eks.aws_iam_policy.cluster_encryption[0] will be created
  + resource "aws_iam_policy" "cluster_encryption" {
      + arn         = (known after apply)
      + description = "Cluster encryption policy to allow cluster role to utilize CMK provided"
      + id          = (known after apply)
      + name        = (known after apply)
      + name_prefix = (known after apply)
      + path        = "/"
      + policy      = (known after apply)
      + policy_id = (known after apply)
      + tags_all    = (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            = (known after apply)
      + path                   = "/"
      + tags_all               = (known after apply)
      + unique_id              = (known after apply)

      + inline_policy {
          + name = (known after apply)
          + policy = jsonencode(
                {
                  + Statement = [
                      + {
                          + Action = [
                              + "logs:CreateLogGroup",
                            ]
                          + Effect = "Deny"
                          + Resource = "*"
                        },
                    ]
                  + Version = "2012-10-17"
                }
            )
        }
    }

  # module.eks.aws_iam_role_policy_attachment.cluster_encryption[0] will be created
  + resource "aws_iam_role_policy_attachment" "cluster_encryption" {
      + id         = (known after apply)
      + policy_arn = (known after apply)
      + role       = (known after apply)
    }

  # module.eks.aws_iam_role_policy_attachment.this["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["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            = (known after apply)
      + owner_id               = (known after apply)
      + revoke_rules_on_delete = false
      + tags                   = (known after apply)
      + tags_all               = (known after apply)
      + 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            = (known after apply)
      + owner_id               = (known after apply)
      + revoke_rules_on_delete = false
      + tags                   = (known after apply)
      + tags_all               = (known after apply)
      + vpc_id                 = (known after apply)
    }

  # 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)
      + 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_all"] will be created
  + resource "aws_security_group_rule" "node" {
      + cidr_blocks              = [
          + "0.0.0.0/0",
        ]
      + description              = "Allow all egress"
      + from_port                = 0
      + id                       = (known after apply)
      + prefix_list_ids          = []
      + protocol                 = "-1"
      + security_group_id        = (known after apply)
      + security_group_rule_id = (known after apply)
      + self                     = false
      + source_security_group_id = (known after apply)
      + to_port                  = 0
      + 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_8443_webhook"] will be created
  + resource "aws_security_group_rule" "node" {
      + description              = "Cluster API to node 8443/tcp webhook"
      + from_port                = 8443
      + 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                  = 8443
      + type                     = "ingress"
    }

  # module.eks.aws_security_group_rule.node["ingress_cluster_9443_webhook"] will be created
  + resource "aws_security_group_rule" "node" {
      + description              = "Cluster API to node 9443/tcp webhook"
      + from_port                = 9443
      + 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                  = 9443
      + 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_nodes_ephemeral"] will be created
  + resource "aws_security_group_rule" "node" {
      + description              = "Node to node ingress on ephemeral ports"
      + from_port                = 1025
      + 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                  = 65535
      + 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 UDP"
      + 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_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                = (known after apply)
      + 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                  = {
          + "Name" = "guru-vpc-us-east-1a"
        }
      + tags_all              = {
          + "Name" = "guru-vpc-us-east-1a"
        }
      + vpc                   = true
    }

  # 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     = {
          + "Name" = "guru-vpc"
        }
      + tags_all = {
          + "Name" = "guru-vpc"
        }
      + 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)
      + 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                  = {
          + "Name" = "guru-vpc-us-east-1a"
        }
      + tags_all              = {
          + "Name" = "guru-vpc-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             = {
          + "Name" = "guru-vpc-private"
        }
      + tags_all         = {
          + "Name" = "guru-vpc-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             = {
          + "Name" = "guru-vpc-public"
        }
      + tags_all         = {
          + "Name" = "guru-vpc-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.private[2] 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_route_table_association.public[2] 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                                     = "10.0.1.0/24"
      + 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                                           = (known after apply)
      + tags_all                                       = (known after apply)
      + 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                                     = "10.0.2.0/24"
      + 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                                           = (known after apply)
      + tags_all                                       = (known after apply)
      + vpc_id                                         = (known after apply)
    }

  # module.vpc.aws_subnet.private[2] will be created
  + resource "aws_subnet" "private" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "us-east-1c"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "10.0.3.0/24"
      + 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                                           = (known after apply)
      + tags_all                                       = (known after apply)
      + 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                                     = "10.0.4.0/24"
      + 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                        = true
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + tags                                           = (known after apply)
      + tags_all                                       = (known after apply)
      + 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                                     = "10.0.5.0/24"
      + 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                        = true
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + tags                                           = (known after apply)
      + tags_all                                       = (known after apply)
      + vpc_id                                         = (known after apply)
    }

  # module.vpc.aws_subnet.public[2] will be created
  + resource "aws_subnet" "public" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "us-east-1c"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "10.0.6.0/24"
      + 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                        = true
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + tags                                           = (known after apply)
      + tags_all                                       = (known after apply)
      + vpc_id                                         = (known after apply)
    }

  # module.vpc.aws_vpc.this[0] will be created
  + resource "aws_vpc" "this" {
      + arn                                  = (known after apply)
      + assign_generated_ipv6_cidr_block     = false
      + 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_classiclink                   = (known after apply)
      + enable_classiclink_dns_support       = (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                                 = {
          + "Name" = "guru-vpc"
        }
      + tags_all                             = {
          + "Name" = "guru-vpc"
        }
    }

  # module.eks.module.eks_managed_node_group["one"].aws_eks_node_group.this[0] will be created
  + resource "aws_eks_node_group" "this" {
      + ami_type                = "AL2_x86_64"
      + arn                     = (known after apply)
      + capacity_type           = (known after apply)
      + cluster_name            = (known after apply)
      + disk_size               = (known after apply)
      + id                      = (known after apply)
      + instance_types          = [
          + "t3.small",
        ]
      + node_group_name         = (known after apply)
      + node_group_name_prefix = "node-group-1-"
      + 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                    = {
          + "Name" = "node-group-1"
        }
      + tags_all                = {
            + "Name" = "node-group-1"
        }
      + version                 = "1.24"

      + launch_template {
          + id      = (known after apply)
          + name    = (known after apply)
          + version = (known after apply)
        }

      + scaling_config {
          + desired_size = 2
          + max_size     = 3
          + min_size     = 1
        }

      + timeouts {}

      + update_config {
          + max_unavailable_percentage = 33
        }
    }

  # module.eks.module.eks_managed_node_group["one"].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            = "node-group-1-eks-node-group-"
      + path                   = "/"
      + tags_all               = (known after apply)
      + unique_id              = (known after apply)

      + inline_policy {
          + name = (known after apply)
          + policy = (known after apply)
        }
    }

  # module.eks.module.eks_managed_node_group["one"].aws_iam_role_policy_attachment.this["arn:aws:iam::aws:policy/AmazonEC2ContainerRegistry
  + 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["one"].aws_iam_role_policy_attachment.this["arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
  + 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["one"].aws_iam_role_policy_attachment.this["arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"] wil
  + 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["one"].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 node-group-1 EKS managed node group"
      + id                     = (known after apply)
      + latest_version         = (known after apply)
      + name                   = (known after apply)
      + name_prefix            = "one-"
      + tags_all               = (known after apply)
      + update_default_version = true
      + vpc_security_group_ids = (known after apply)

      + metadata_options {
          + http_endpoint               = "enabled"
          + http_protocol_ipv6          = "disabled"
          + http_put_response_hop_limit = 2
          + http_tokens                 = "required"
          + instance_metadata_tags      = "disabled"
        }

      + monitoring {
          + enabled = true
        }

      + tag_specifications {
          + resource_type = "instance"
          + tags          = {
              + "Name" = "node-group-1"
            }
        }
      + tag_specifications {
          + resource_type = "network-interface"
          + tags          = {
              + "Name" = "node-group-1"
            }
        }
      + tag_specifications {
          + resource_type = "volume"
          + tags          = {
              + "Name" = "node-group-1"
            }
        }
    }

  # module.eks.module.eks_managed_node_group["two"].aws_eks_node_group.this[0] will be created
  + resource "aws_eks_node_group" "this" {
      + ami_type               = "AL2_x86_64"
      + arn                    = (known after apply)
      + capacity_type          = (known after apply)
      + cluster_name           = (known after apply)
      + disk_size              = (known after apply)
      + id                     = (known after apply)
      + instance_types         = [
          + "t3.small",
        ]
      + node_group_name        = (known after apply)
      + node_group_name_prefix = "node-group-2-"
      + 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                    = {
          + "Name" = "node-group-2"
        }
      + tags_all                = {
          + "Name" = "node-group-2"
        }
      + version                 = "1.24"

      + launch_template {
          + id      = (known after apply)
          + name    = (known after apply)
          + version = (known after apply)
        }

      + scaling_config {
          + desired_size = 1
          + max_size     = 2
          + min_size     = 1
        }

      + timeouts {}

      + update_config {
          + max_unavailable_percentage = 33
        }
    }

  # module.eks.module.eks_managed_node_group["two"].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           = "node-group-2-eks-node-group-"
      + path                  = "/"
      + tags_all              = (known after apply)
      + unique_id             = (known after apply)

      + inline_policy {
          + name = (known after apply)
          + policy = (known after apply)
        }
    }

  # module.eks.module.eks_managed_node_group["two"].aws_iam_role_policy_attachment.this["arn:aws:iam::aws:policy/AmazonEC2ContainerRegistry
  + 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["two"].aws_iam_role_policy_attachment.this["arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
  + 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["two"].aws_iam_role_policy_attachment.this["arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"] wil
  + 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["two"].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 node-group-2 EKS managed node group"
      + id                     = (known after apply)
      + latest_version         = (known after apply)
      + name                   = (known after apply)
      + name_prefix            = "two-"
      + tags_all               = (known after apply)
      + update_default_version = true
      + vpc_security_group_ids = (known after apply)

      + metadata_options {
          + http_endpoint               = "enabled"
          + http_protocol_ipv6          = "disabled"
          + http_put_response_hop_limit = 2
          + http_tokens                 = "required"
          + instance_metadata_tags      = "disabled"
        }

      + monitoring {
          + enabled = true
        }

      + tag_specifications {
          + resource_type = "instance"
          + tags          = {
              + "Name" = "node-group-2"
            }
        }
      + tag_specifications {
          + resource_type = "network-interface"
          + tags          = {
              + "Name" = "node-group-2"
            }
        }
      + tag_specifications {
          + resource_type = "volume"
          + tags          = {
              + "Name" = "node-group-2"
            }
        }
    }

  # module.eks.module.kms.data.aws_iam_policy_document.this[0] will be read during apply #
  (config refers to values not yet known)
 <= data "aws_iam_policy_document" "this" {
      + id                        = (known after apply)
      + json                      = (known after apply)
      + override_policy_documents = []
      + source_policy_documents = []

      + statement {
          + actions = [
              + "kms:CancelKeyDeletion",
              + "kms:Create*",
              + "kms:Delete*",
              + "kms:Describe*",
              + "kms:Disable*",
              + "kms:Enable*",
              + "kms:Get*",
              + "kms:List*",
              + "kms:Put*",
              + "kms:Revoke*",
              + "kms:ScheduleKeyDeletion",
              + "kms:TagResource",
              + "kms:UntagResource",
              + "kms:Update*",
            ]
           + resources = [
              + "*",
            ]
          + sid       = "KeyAdministration"

          + principals {
              + identifiers = [
                  + "arn:aws:iam::988505337009:user/k8sadmin",
                ]
              + type        = "AWS"
            }
        }
      + statement {
          + actions = [
              + "kms:Decrypt",
              + "kms:DescribeKey",
              + "kms:Encrypt",
              + "kms:GenerateDataKey*",
              + "kms:ReEncrypt*",
            ]
          + resources = [
              + "*",
            ]
          + sid       = "KeyUsage"

          + principals {
              + identifiers = [
                  + (known after apply),
                ]
              + type        = "AWS"
            }
        }
    }

  # module.eks.module.kms.aws_kms_alias.this["cluster"] will be created
  + resource "aws_kms_alias" "this" {
      + arn            = (known after apply)
      + id             = (known after apply)
      + name           = (known after apply)
      + name_prefix    = (known after apply)
      + target_key_arn = (known after apply)
      + target_key_id = (known after apply)
    }

  # module.eks.module.kms.aws_kms_key.this[0] will be created
  + resource "aws_kms_key" "this" {
      + arn                                = (known after apply)
      + bypass_policy_lockout_safety_check = false
      + customer_master_key_spec           = "SYMMETRIC_DEFAULT"
      + description                        = (known after apply)
      + enable_key_rotation                = true
      + id                                 = (known after apply)
      + is_enabled                         = true
      + key_id                             = (known after apply)
      + key_usage                          = "ENCRYPT_DECRYPT"
      + multi_region                       = false
      + policy                             = (known after apply)
      + tags_all                           = (known after apply)
    }
Plan: 54 to add, 0 to change, 0 to destroy.
Changes to Outputs:
  + cluster_endpoint          = (known after apply)
  + cluster_name              = (known after apply)
  + cluster_security_group_id = (known after apply)
  + region                    = "us-east-1"

─────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you
run "terraform apply" now.
```

Finalmente terraform apply (Lo cual nos va a llevar unos cuantos minutos).

```
terraform apply --auto-approve
```

Finalmente

```
Apply complete! Resources: 54 added, 0 changed, 0 destroyed.

Outputs:

cluster_endpoint = "https://43F725EF95A10F6FF084D18A8BDFA2BD.gr7.us-east-1.eks.amazonaws.com"
cluster_name = "guru-eks-eks1o8mn"
cluster_security_group_id =
"sg-0b3b25f18a24fdf0d" region = "us-east-1"
```

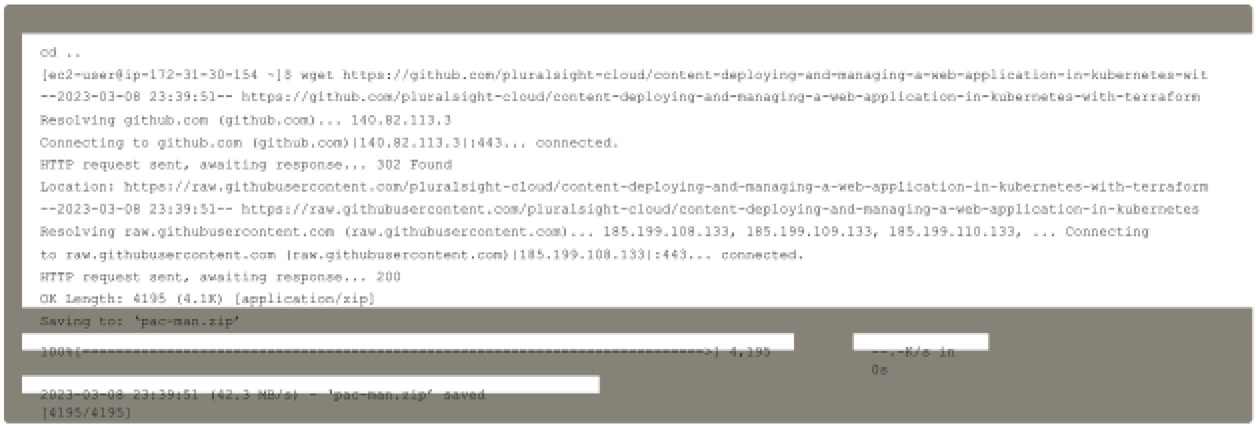
ahora agregamos el context

```
aws eks --region $(terraform output -raw region) update-kubeconfig --name $(terraform output -raw cluster_name)
```

vemos que está todo ok, seguimos

```
[ec2-user@ip-172-31-30-154 eks]$ aws eks --region $(terraform output -raw region) update-kubeconfig --name $(terraform output -raw cluster_
Added new context arn:aws:eks:us-east-1:988505337009:cluster/guru-eks-eks1o8mn to /home/ec2-user/.kube/config
[ec2-user@ip-172-31-30-154 eks]$ kubectl get nodes
NAME                          STATUS   ROLES    AGE     VERSION
ip-10-0-1-196.ec2.internal    Ready    <none>   5m31s   v1.24.10-eks-48e63af
ip-10-0-2-128.ec2.internal    Ready    <none>   5m10s   v1.24.10-eks-48e63af
ip-10-0-2-135.ec2.internal    Ready    <none>   5m21s   v1.24.10-eks-48e63af
```

## Bajamos la aplicacion del pacman



```
cd ..
[ec2-user@ip-172-31-30-154 ~]$ wget https://github.com/pluralsight-cloud/content-deploying-and-managing-a-web-application-in-kubernetes-wit
--2023-03-08 23:39:51-- https://github.com/pluralsight-cloud/content-deploying-and-managing-a-web-application-in-kubernetes-with-terraform
Resolving github.com (github.com)... 140.82.113.3
Connecting to github.com (github.com)|140.82.113.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/pluralsight-cloud/content-deploying-and-managing-a-web-application-in-kubernetes-with-terraform
--2023-03-08 23:39:51-- https://raw.githubusercontent.com/pluralsight-cloud/content-deploying-and-managing-a-web-application-in-kubernetes
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ... Connecting
to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200
OK Length: 4195 (4.1K) [application/zip]
Saving to: 'pac-man.zip'
100%[=============================================================================>] 4,195       --.-K/s in
                                                                                                  0s
2023-03-08 23:39:51 (42.3 MB/s) - 'pac-man.zip' saved
[4195/4195]
```

ahora descomprimo

```
[ec2-user@ip-172-31-30-154 ~]$ unzip
pac-man.zip Archive: pac-man.zip
   creating:
  pac-man/modules/
  inflating:
  pac-man/pac-man.tf
   creating: pac-man/modules/mongo/
   creating: pac-man/modules/pac-man/
  inflating:
  pac-man/modules/mongo/mongo-deployment.tf
  inflating: pac-man/modules/mongo/mongo-pv.tf
  inflating: pac-man/modules/mongo/mongo-pvc.tf
  inflating: pac-man/modules/mongo/mongo-sc.tf
```

```
 inflating:
pac-man/modules/mongo/mongo-service.tf
extracting: pac-man/modules/mongo/variables.tf
 inflating:
 pac-man/modules/pac-man/pac-man-deployment.tf
 inflating: pac-man/modules/pac-man/pac-man-service.tf
extracting: pac-man/modules/pac-man/variables.tf
```

Veo los directorios:

```
[ec2-user@ip-172-31-30-154 ~]$ cd
pac-man/ [ec2-user@ip-172-31-30-154
pac-man]$ ls -R
.:
modules pac-man.tf

./modules:
mongo pac-man

./modules/mongo:
mongo-deployment.tf mongo-pvc.tf mongo-pv.tf mongo-sc.tf mongo-service.tf variables.tf

./modules/pac-man:
pac-man-deployment.tf pac-man-service.tf variables.tf
```

tenemos que agregar la imagen de docker de pacman. vamos a hacerlo!

Linea 30, agregamos.

```
sp
ec
{     container {
        name = "pac-man"
        image =
        "docker.io/jessehoch/pacman-nodejs-app:latest"
```

Salvamos y vamos al modulo de mongodb

Editamos el archivo

```
[ec2-user@ip-172-31-30-154 pac-man]$ vim pac-man.tf
```

agregamos al final de todo esto

en kubernetes_namespace le faltaba el “pac-man”

```
module "mongo" {
  source = "./modules/mongo"
  kubernetes_namespace = "pac-man"
}

module "pac-man" {
  source = "./modules/pac-man"
  kubernetes_namespace =
  "pac-man" depends_on =
  [module.mongo]
}
```

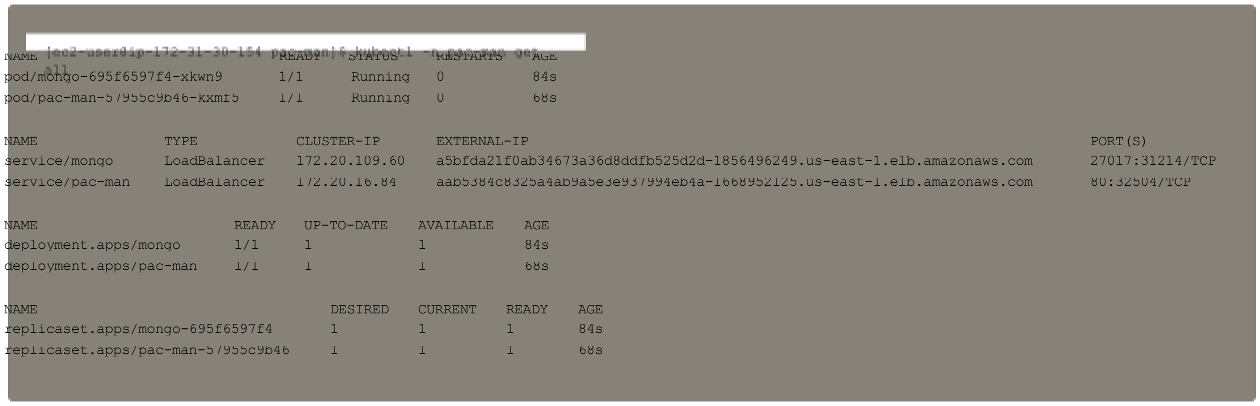
Y salvamos

```
terraform init && terraform plan && terraform apply
```

Esperamos y vemos si todo salio como esperabamos

```
kubectl -n pac-man get all
```

Vemos que la salida es....



```
[ec2-user@ip-172-31-30-154 pac-man]$ kubectl -n pac-man get
all
NAME                             READY   STATUS    RESTARTS   AGE
pod/mongo-695f6597f4-xkwn9       1/1     Running   0          84s
pod/pac-man-57955c9b46-kxmf5     1/1     Running   0          68s

NAME              TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)
service/mongo     LoadBalancer   172.20.109.60   a5bfda21f0ab34673a36d8ddfb525d2d-1856496249.us-east-1.elb.amazonaws.com    27017:31214/TCP
service/pac-man   LoadBalancer   172.20.16.84    aab5384c8325a4ab9a5e3e937994eb4a-1668952125.us-east-1.elb.amazonaws.com    80:32504/TCP

NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mongo     1/1     1            1           84s
deployment.apps/pac-man   1/1     1            1           68s

NAME                                 DESIRED   CURRENT   READY   AGE
replicaset.apps/mongo-695f6597f4     1         1         1       84s
replicaset.apps/pac-man-57955c9b46   1         1         1       68s
```

Lo que nos interesa es

LB de aca. veamos....

```
kubectl get svc -n pacman
```