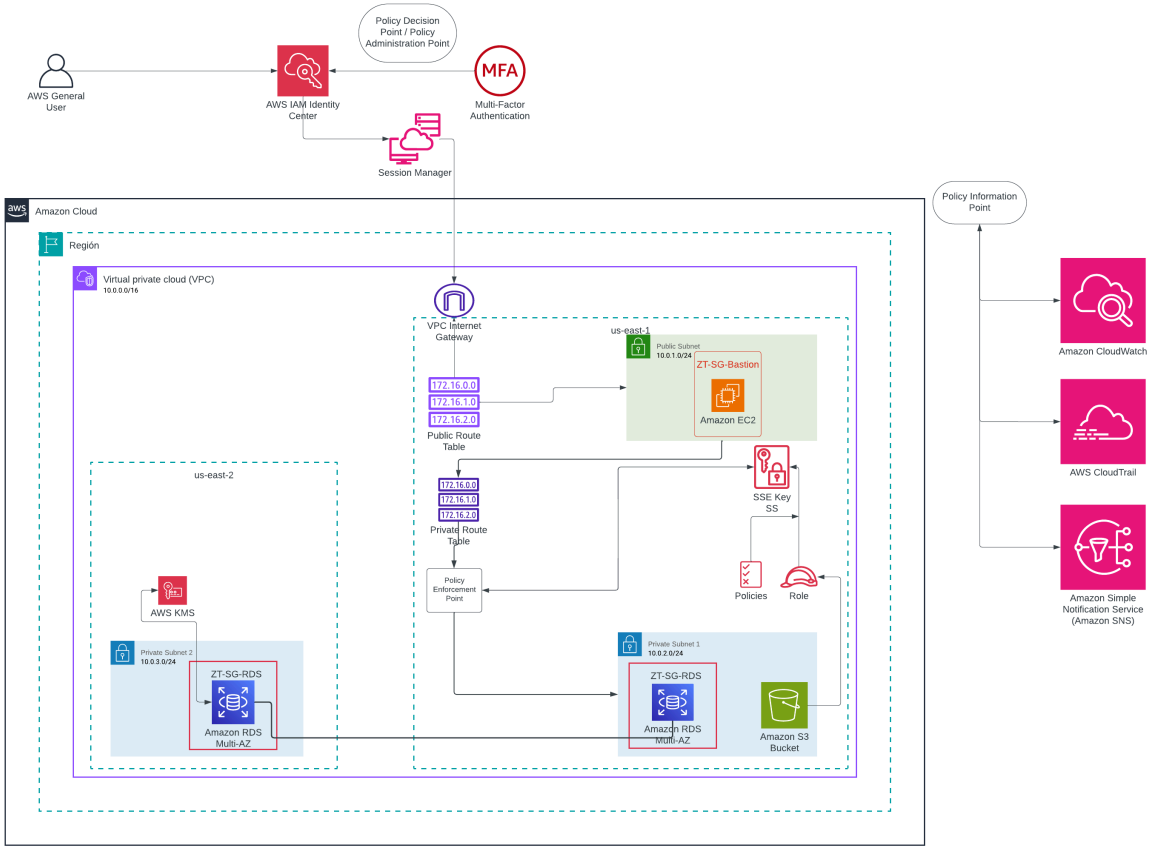
AWS Cloud Hardening & Monitoring (Free Tier Edition)

Por: Cristian Jiménez

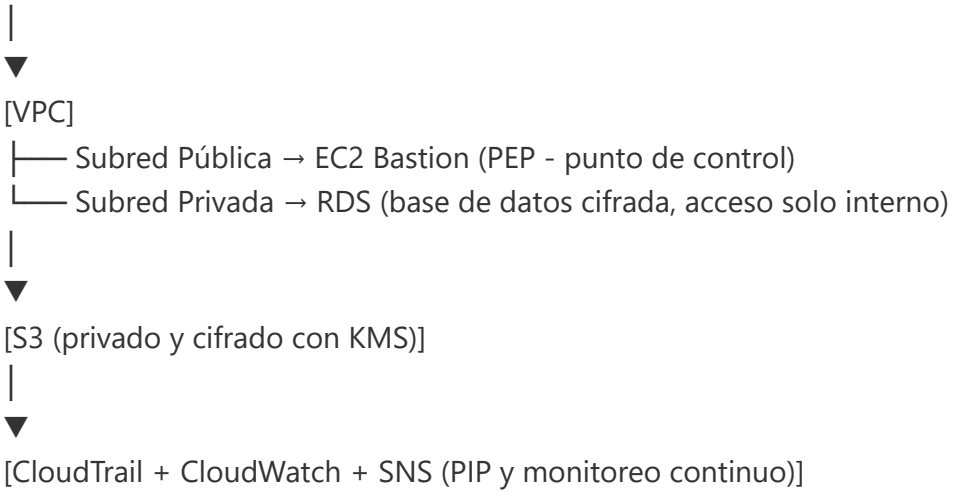
Objetivos

Implementar una arquitectura segura en AWS aplicando los **principios de Zero Trust Architecture (ZTA)** según el marco **NIST SP 800-207**, utilizando exclusivamente servicios disponibles en el **Free Tier de AWS**. El Proyecto demuestra cómo aplicar controles de acceso estrictos, segmentación de red, cifrado y monitoreo continuo sin depender de servicios premium como GuardDuty o Security Hub.
[Expandir un poco mas los objetivos y si es posible agregar una introducción]

Arquitectura General



[Usuario / Administrador]
|
▼
[IAM + MFA (PAP & PDP)]



🧱 Servicios utilizados

Servicio	Rol / Propósito
IAM	Control de identidades y políticas (PAP / PDP).
S3	Almacenamiento seguro con cifrado y acceso controlado.
KMS	Cifrado de datos en reposo para S3, RDS y EBS.
VPC	Microsegmentación de red (Zonas de confianza reducida).
EC2 (Bastion)	Punto de Acceso seguro a recursos internos (PEP).
RDS	Base de datos privada cifrada (recurso protegido)
CloudTrail	Auditoría de acciones (PIP: información de políticas).
CloudWatch	Detección de anomalías y generación de alertas
SNS	Notificaciones automáticas de seguridad.

⚙️ Pasos de creación del Entorno

1️⃣ Configuración de seguridad de identidad (IAM)

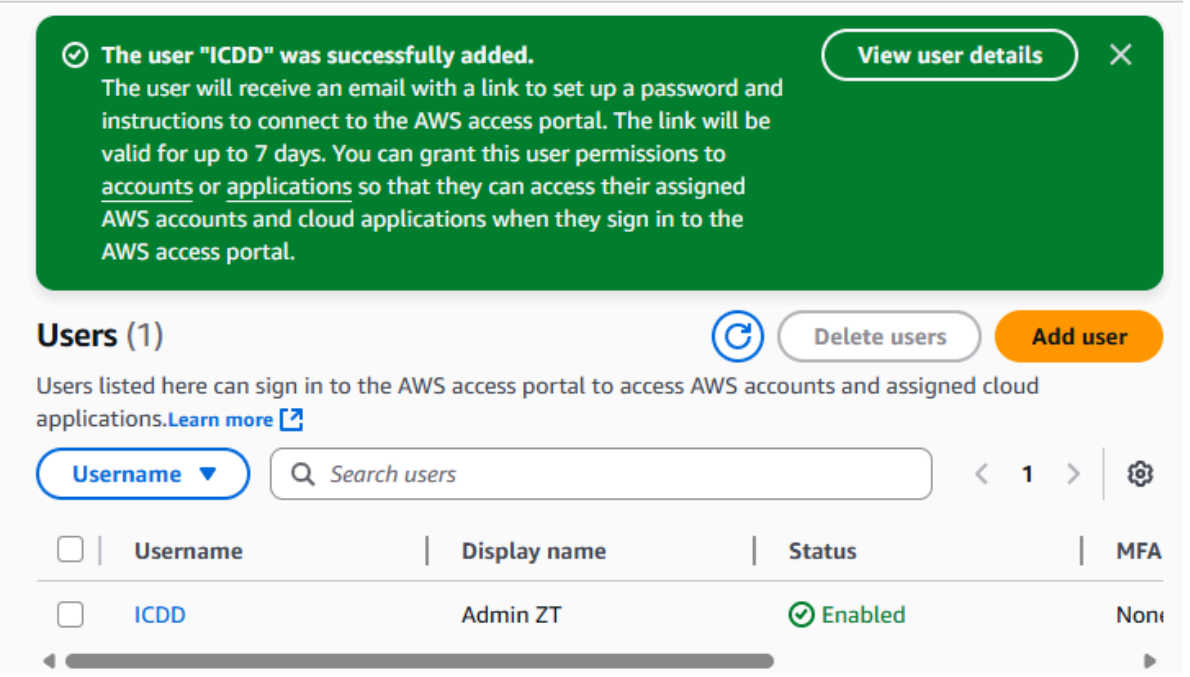
🎯 Objetivo

Configurar el entorno inicial con **Identity Center (antes AWS SSO)** para manejar acceso seguro basado en identidad y MFA.
Esta fase establece el **PDP (Policy Decision Point)** y **PAP (Policy Administration Point)** del modelo Zero Trust de nuestro Proyecto.

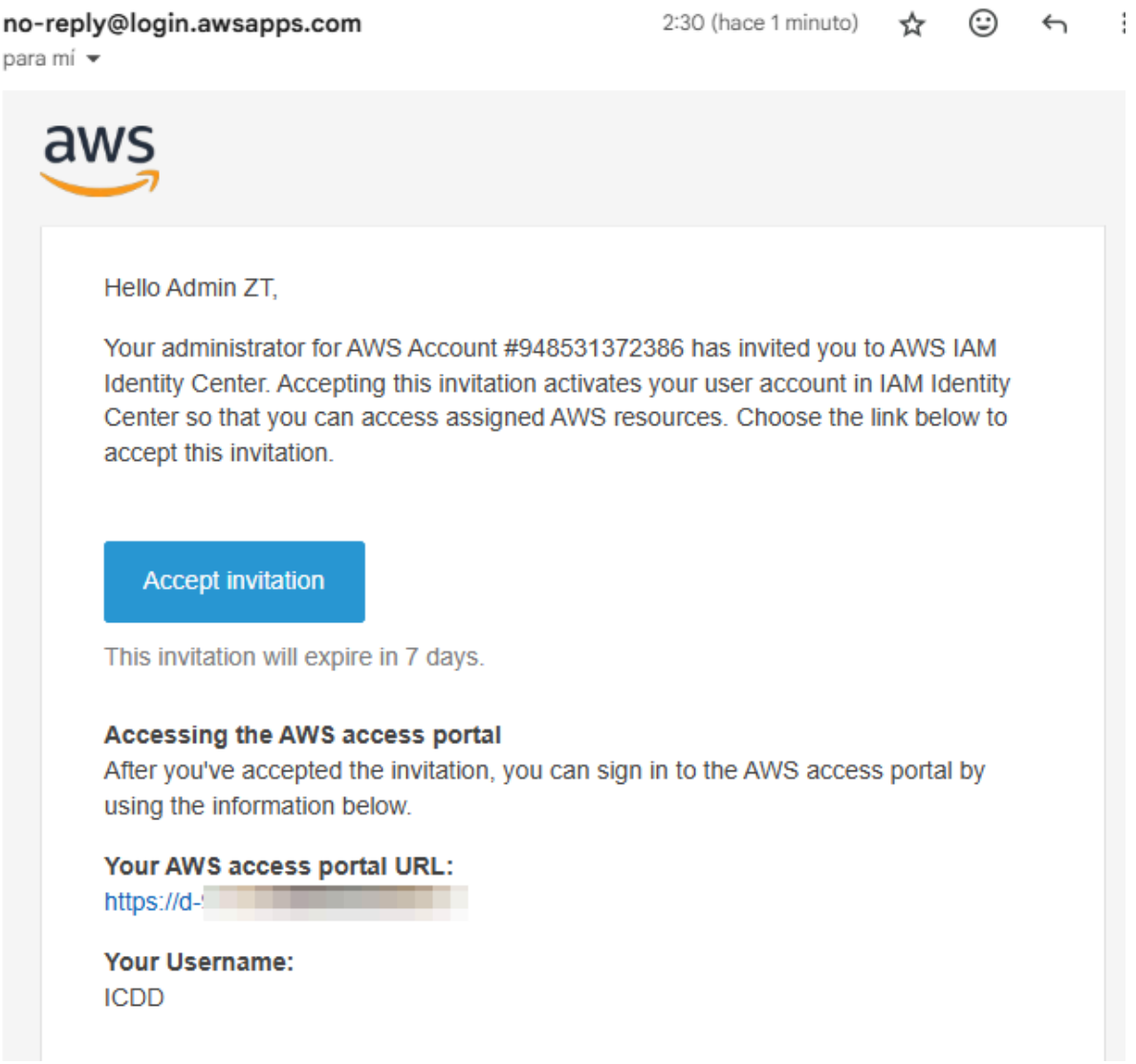
- Habilita **MFA (Multi-Factor Authentication)** para la cuenta root.

- Paso 1: Crear un usuario administrador separado en Identity Center.

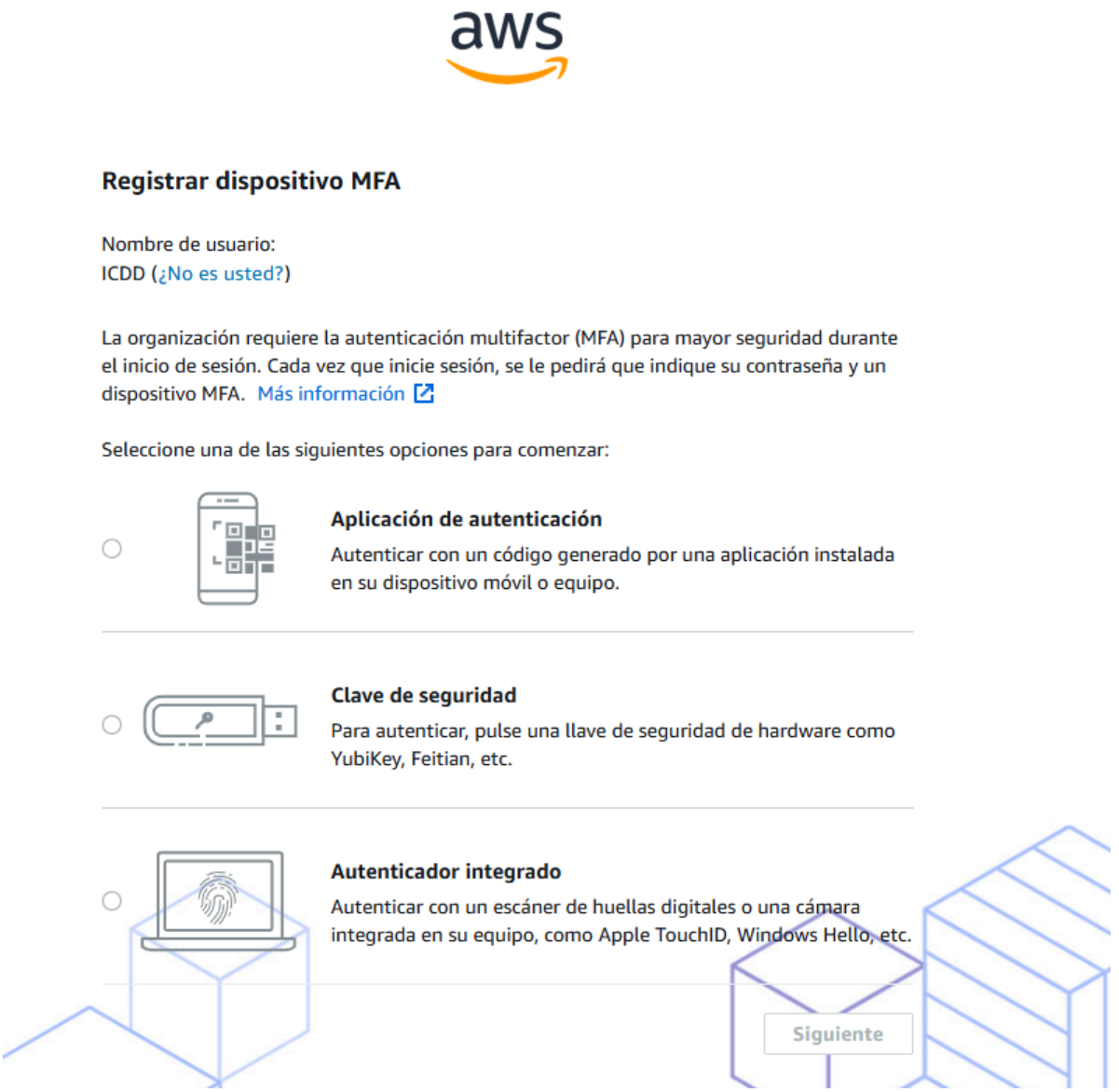
1. En el panel lateral, seleccionamos **Users** → **Add user**.
2. Introducimos los datos:
   a. User Name: el de nuestra preferencia
   b. Email Address: en mi caso utilicé mi correo personal (para recibir invitaciones de inicio de sesión)
   c. Password Options, aquí se dejó la opción "Send an email to the user to set their password".
3. (Opcional) Podemos asignar este usuario a un grupo llamado ZT-Admins.
4. Click en **Add User**.



📘 *Resultado esperado:*
El usuario recibe un **correo con un enlace de activación** para iniciar sesión en el portal Identity Center.



Una vez se acepta la invitación, el mismo portal de AWS le pedirá al nuevo usuario registrar un dispositivo MFA:



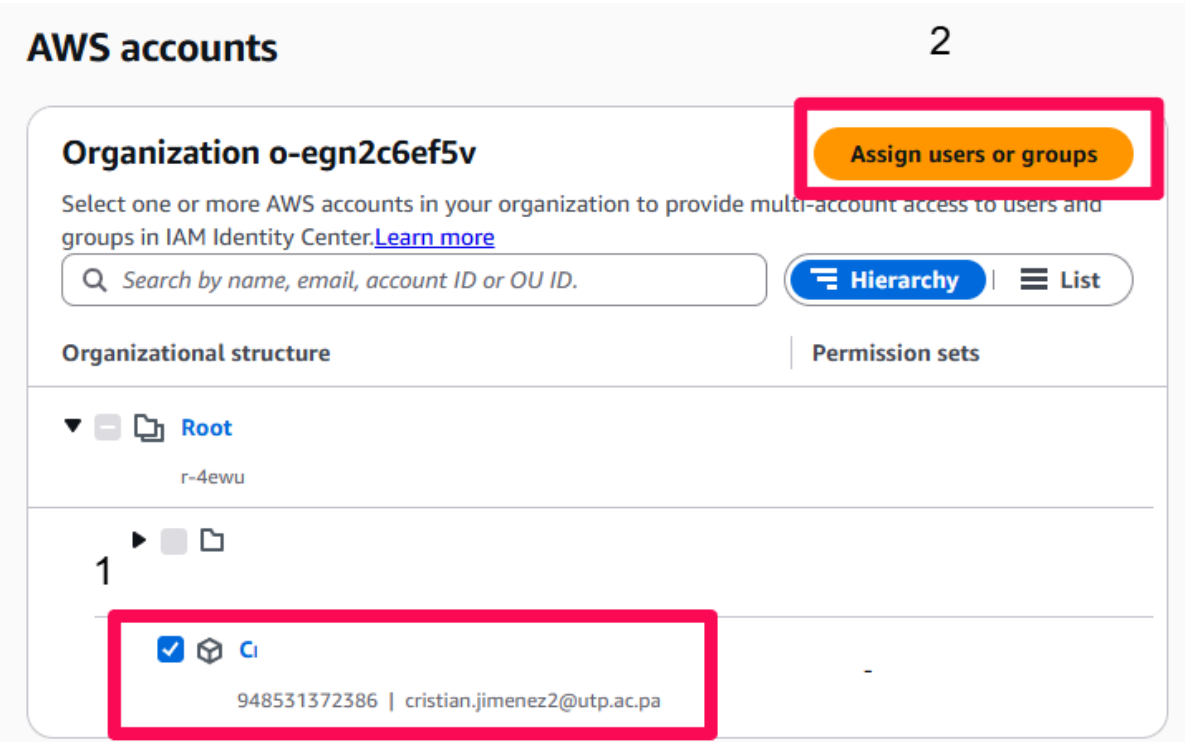
En este caso se usará la Aplicación de autenticación.

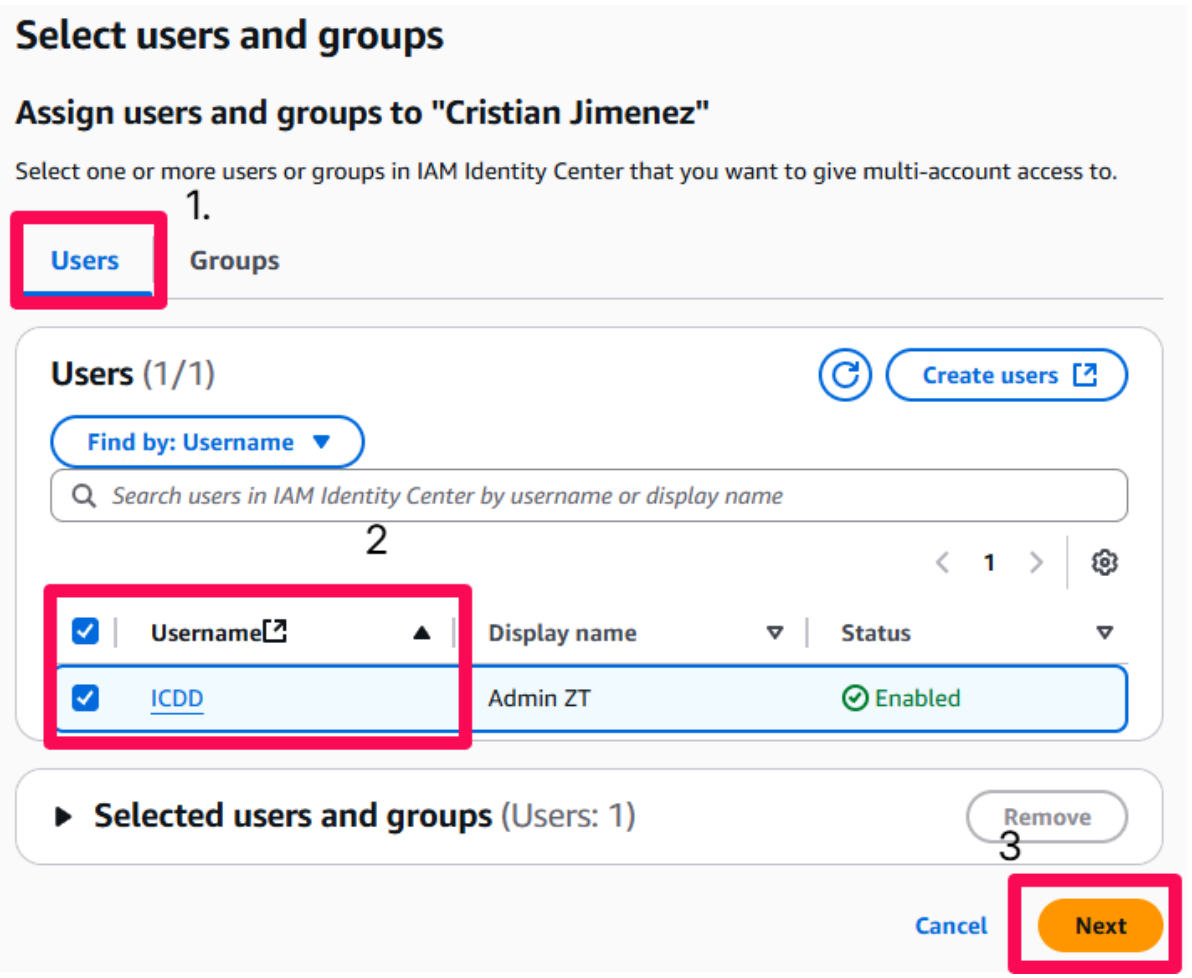
- Luego de seleccionarlo debemos tener instalada una aplicación de autenticación (Ejemplo: Authy, Duo Mobile o Google Authenticator).
- Luego escaneamos el código QR que nos aparece.
- Una vez escaneado colocamos el código de seis digitos de la aplicación de autenticación.

Una vez seguidos los pasos mencionados anteriormente deberiamos poder tener acceso a la consola AWS

- Paso 2: Asignar permisos al usuario

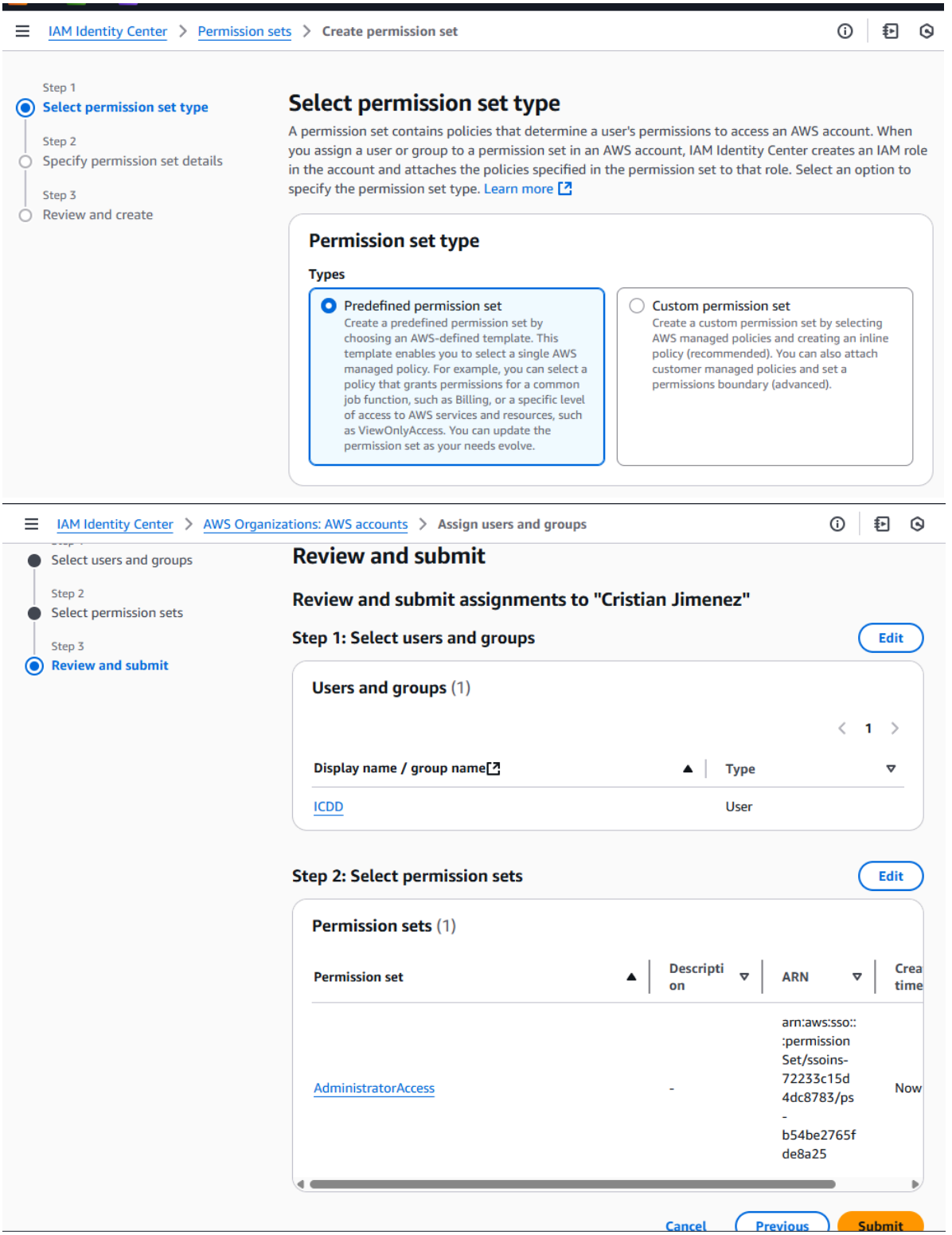
1. En el panel de Identity Center, seleccionamos **AWS Accounts** → **Assign users or groups**.
2. Escogemos nuestra cuenta AWS.
3. Hacemos click en Assign users, y seleccionamos el usuario que recién creamos.





4. Elegimos un **permission set**:

- Seleccionamos "AdministratorAccess" (Se puede crear un conjunto nuevo si se quiere probar personalización).



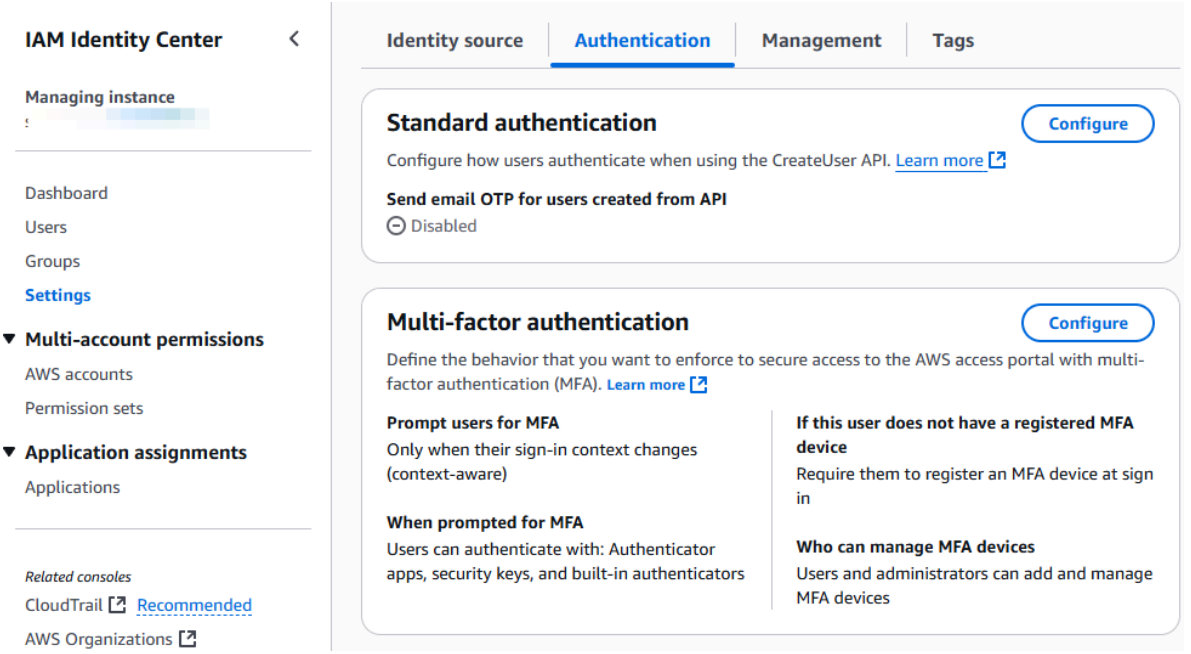
5. Confirmamos la asignación.

📘 *Resultado esperado:*

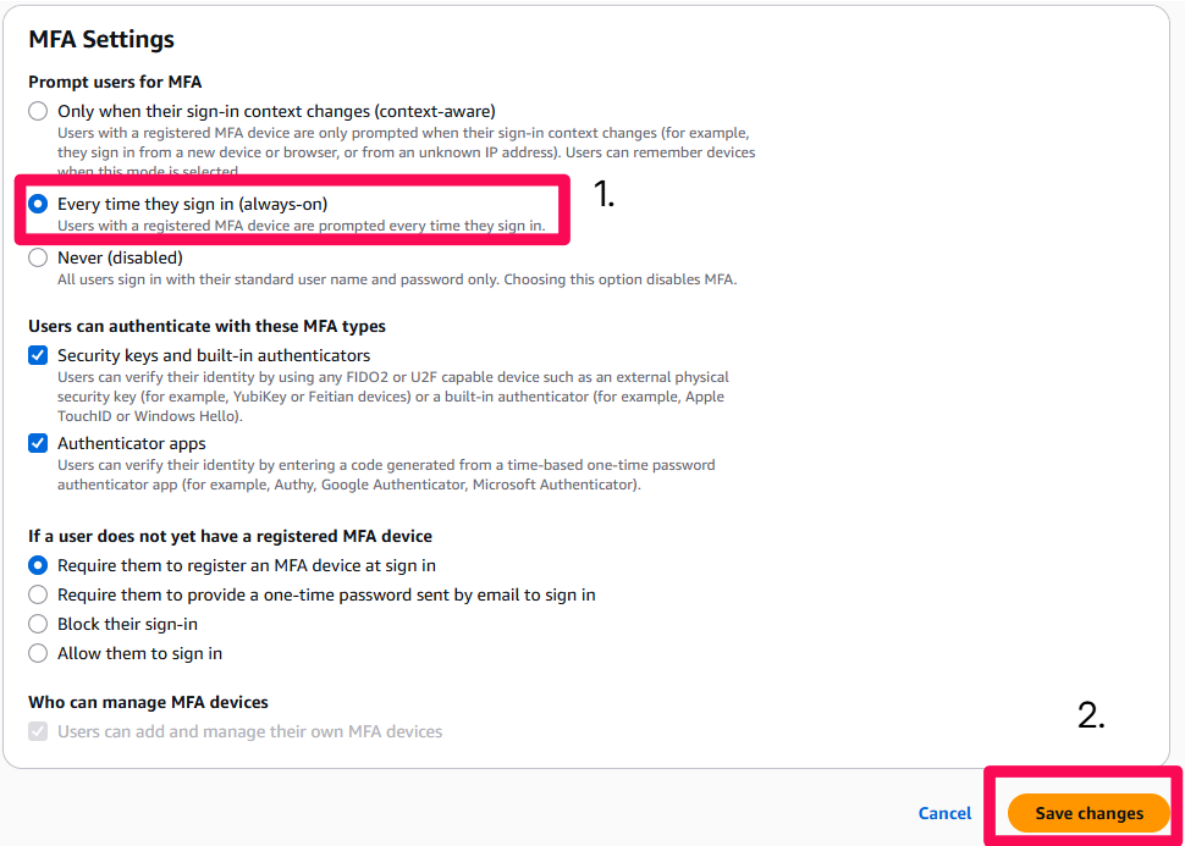
El usuario tiene permisos de administrador dentro de la cuenta, pero autenticación controlada por Identity Center (sin credenciales permanentes).

- Paso 3: Habilitamos MFA (Multi-Factor Authentication)

1. En el panel lateral de **IAM Identity Center**, entramos a **Settings→ Multi-factor authentication**.



2. Clickeamos en Configure, y una vez dentro seleccionamos la casilla que dice **Every time they sign in (always on)**



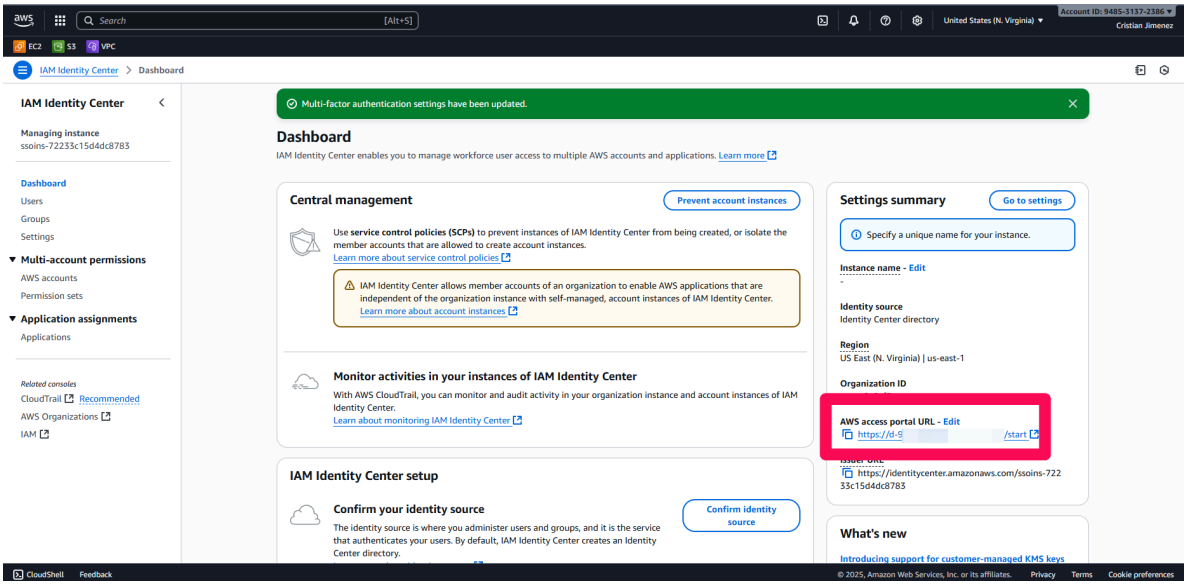
3. Guardamos los cambios.

📘 *Resultado esperado:*

Cada inicio de sesión al portal de AWS requerirá MFA, siguiendo uno de los principios fundamentales de Zero Trust “continuous verification”.

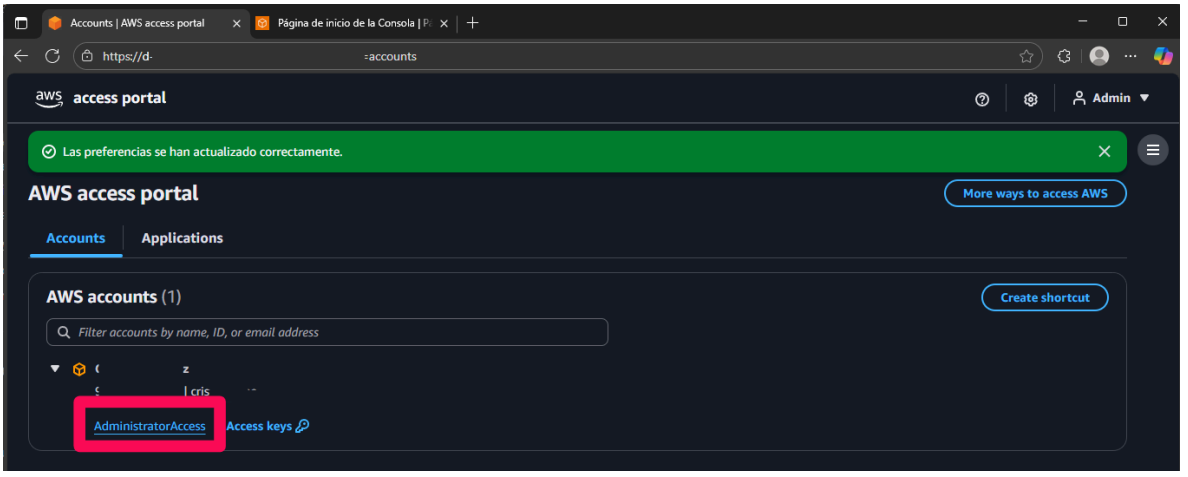
- Paso 4: Probar el inicio de sesión del nuevo usuario

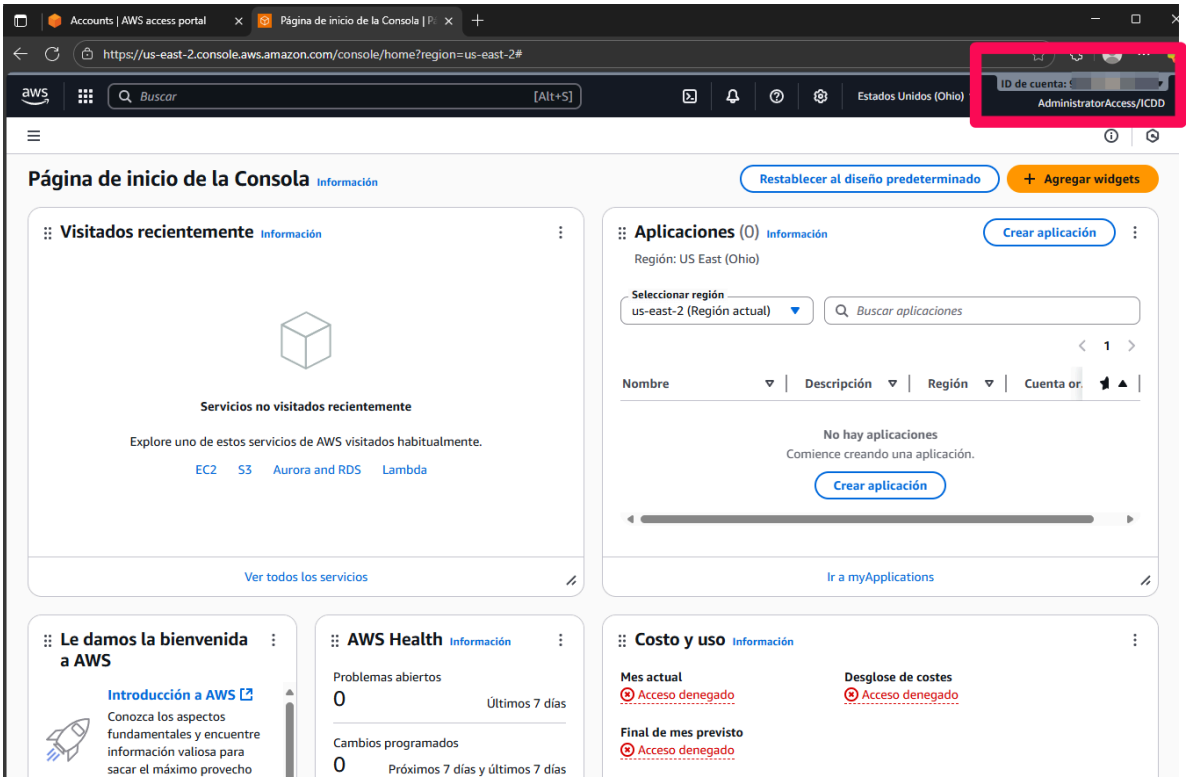
1. Vamos al **User Portal URL** (Identity Center muestra algo como https[:]//d-xxxxx.awsapps[.]com/start



2. Iniciamos sesión con el correo y la contraseña creados anteriormente.
3. Colocamos el código de 6 dígitos generado en nuestra app de autenticación.
    - Nota: Si llegas a este punto y aún no has iniciado sesión en la cuenta de administrador que acabamos de crear, entonces tienes que configurarla desde 0, crearle una contraseña y hacer set up del MFA.

4. Una vez dentro, veremos un panel con nuestros accesos asignados: "AWS Account - Administrator Access".





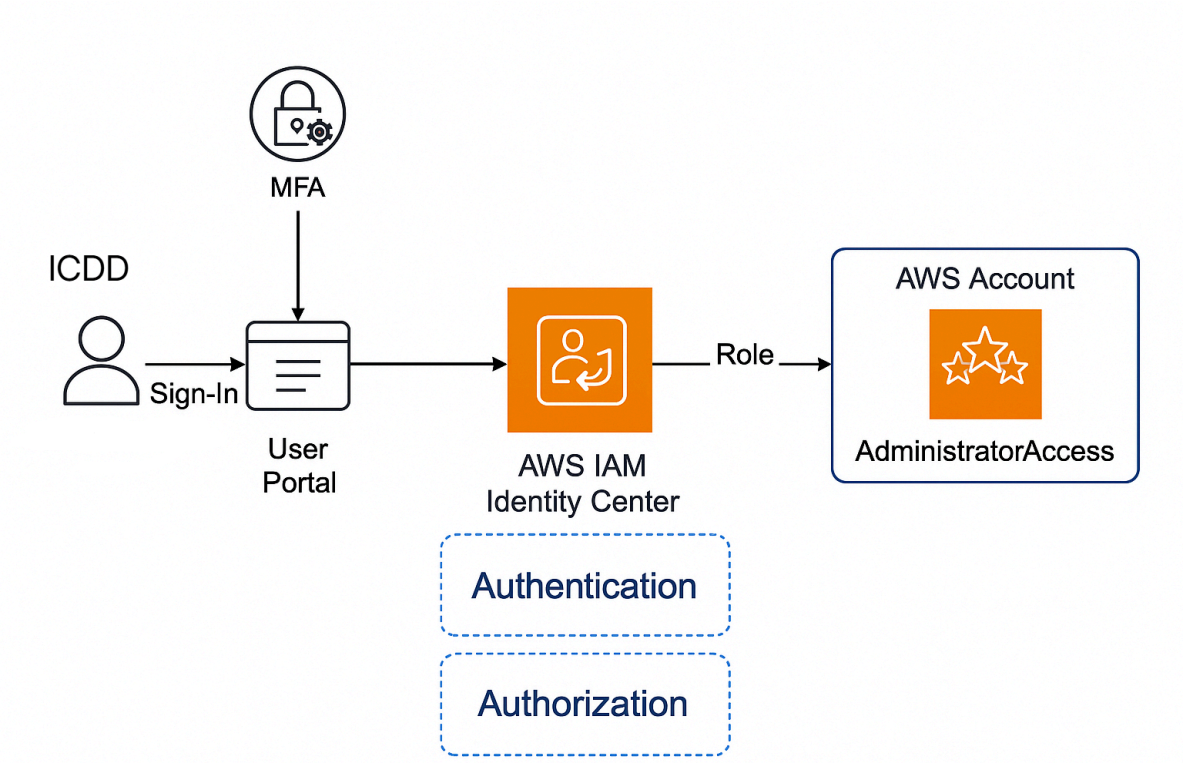
Nota: Asegurarnos de estar en la us-east-1 o la de tu preferencia.

📘 *Resultado esperado:*

El usuario `ICDD` inicia sesión mediante MFA y accede a la consola sin usar claves permanentes.

Resultados finales de la Fase 1

Elemento configurado	Propósito en Zero Trust
IAM Identity Center	Punto de administración de políticas (PAP)
MFA Habilitado	Autenticación continua
Roles Temporales	Sin credenciales permanentes
Permission sets	Aplicación granular de privilegios



Este diagrama muestra el flujo completo de autenticación y autorización basado en Zero Trust:

1. **Usuario: "ICDD"** → inicia sesión en el **User Portal**.
2. **MFA** → se requiere autenticación multifactor para validar identidad.
3. **AWS IAM Identity Center** → actúa como **PAP (Policy Administrator Point)** y **PDP (Policy Decision Point)**
4. **Role Assignment: AdministrationAccess** → se concede acceso temporal a la cuenta AWS.
5. Se indican claramente los procesos de **Authentication y Authorization**, que conforman la base del plano de identidad del proyecto Zero Trust.

Próximo Paso

Con la **identidad segura y MFA habilitado**, ya tenemos el **punto de entrada** al modelo Zero Trust.
El siguiente paso será entrar a la Fase 2: Auditoría y configuración inicial de identidad (IAM & roles), en donde vamos a:

- Auditar permisos efectivos.
- Definir políticas de acceso mínimo
- Preparar la base para la segmentación de recursos

Fase 2 - Identidad y Control de Acceso (IAM)

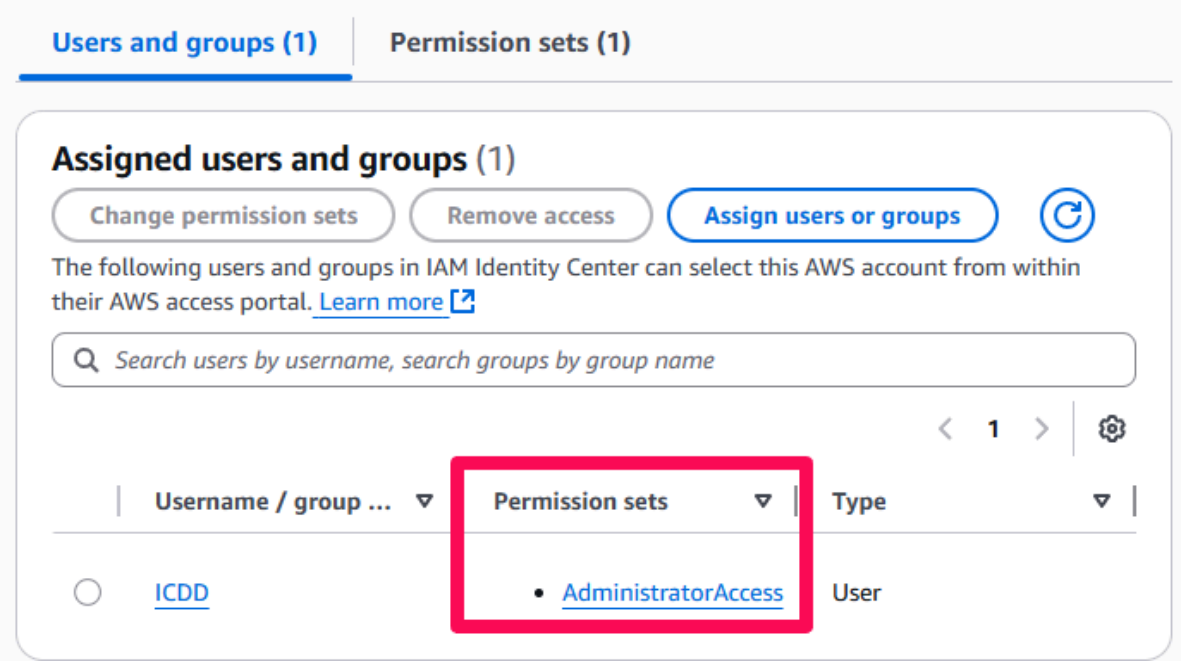
🎯 Objetivo

El objetivo en esta fase es aplicar el **principio de mínimo privlegio (Least Privilege principle)** y habilitar visibilidad completa sobre las identidades y roles de acceso dentro de nuestra cuenta AWS.

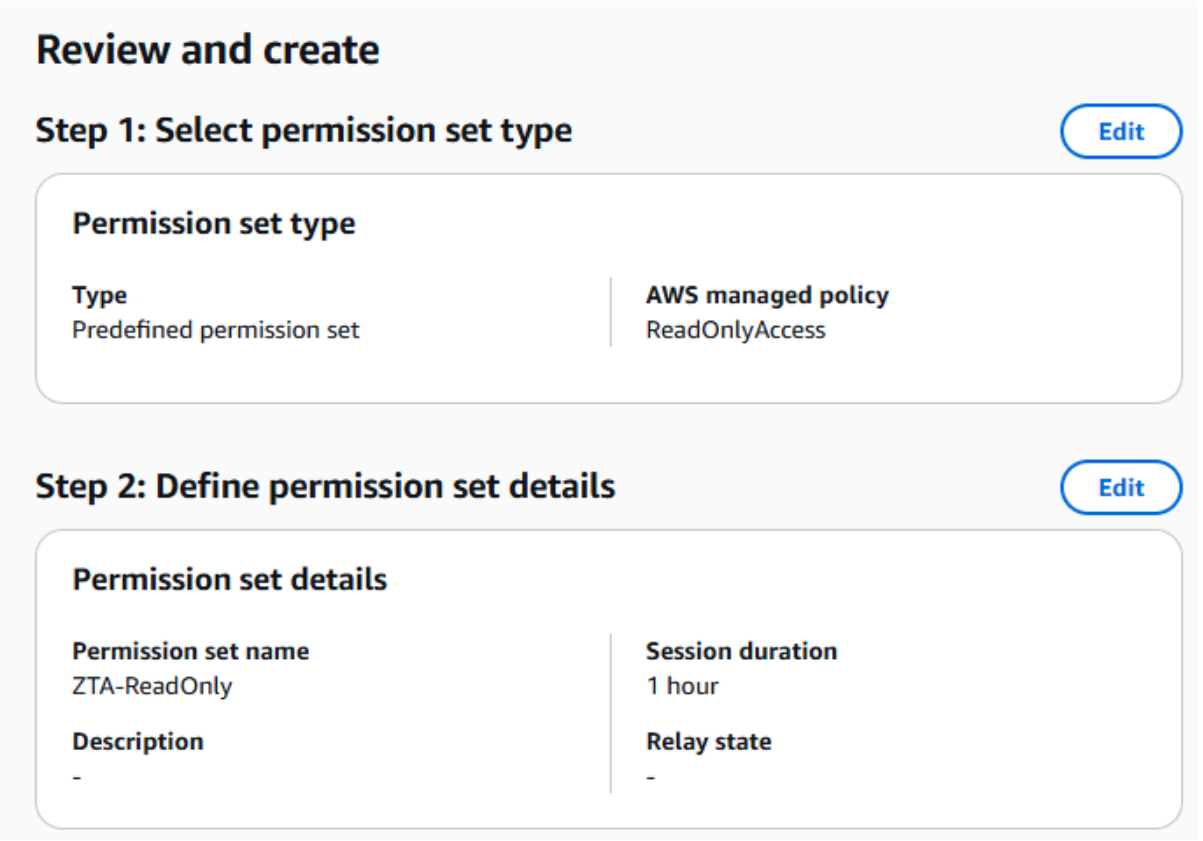
Esta fase representa la consolidación de los componentes **PAP** (Policy Administration Point) y **PDP** (Policy Decision Point) del modelo Zero Trust.

- Paso 1: Revisar la configuración de Identity Center

1. Vamos al panel de **IAM Identity Center → AWS Accounts → Assignments**.
2. Verificamos que nuestro usuario ICCD tenga asignado el permiso:
   - Permission set: **AdministratorAccess**



3. Si queremos probar distintos niveles de acceso (como por ejemplo, un usuario sin permisos de escritura), podemos crear otro permission set:
   - **Permission set name**: ZTA-ReadOnly
   - **Policies**: seleccionamos ReadOnlyAccess.



📘 *Resultado esperado:*

Tenemos acceso a al menos dos roles: uno con acceso administrativo y otro limitado (útil por si queremos simular un **PEP** más adelante).

- Paso 2: Crear roles y políticas específicas (desde AWS IAM)

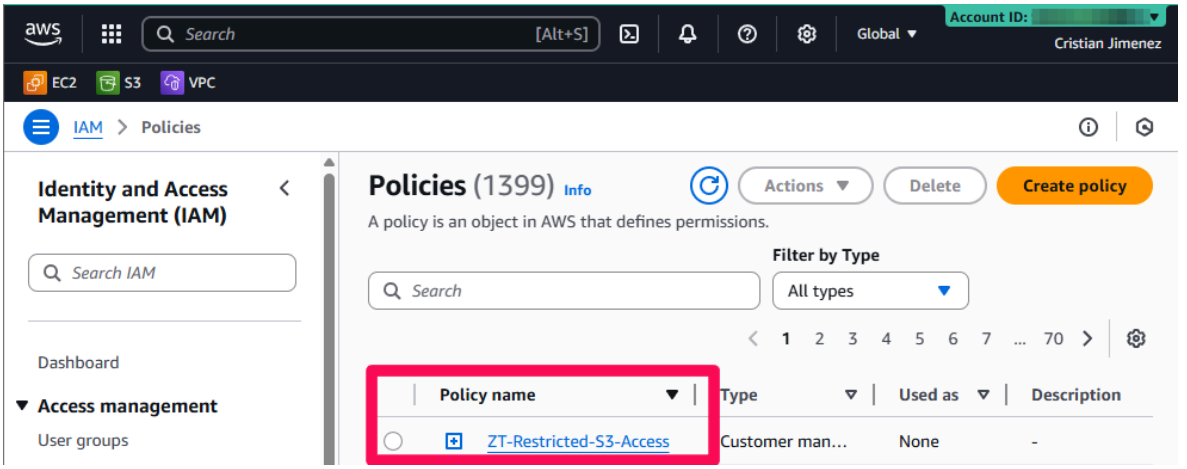
Por mas que usemos Identity Center, el servicio IAM sigue siendo donde definimos las políticas detalladas.
Estas políticas son la base del Enforcement Point (PEP) dentro de nuestra arquitectura.

1. En la consola, abrimos **IAM** → **Policies** → **Create policy**.
2. Creamos una política personalizada llamada ZT-Restricted-S3-Access:

```
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "AllowReadAccessToSpecificBucket",
"Effect": "Allow",
"Action": [
"s3:GetObject",
"s3:ListBucket"
```

```
],
"Resource": [
"arn:aws:s3:::zt-secure-bucket",
"arn:aws:s3:::zt-secure-bucket/*"
],
"Condition": {
"Bool": {
"aws:SecureTransport": "true"
}
}
}
]
}
```

3. Guardamos la política.

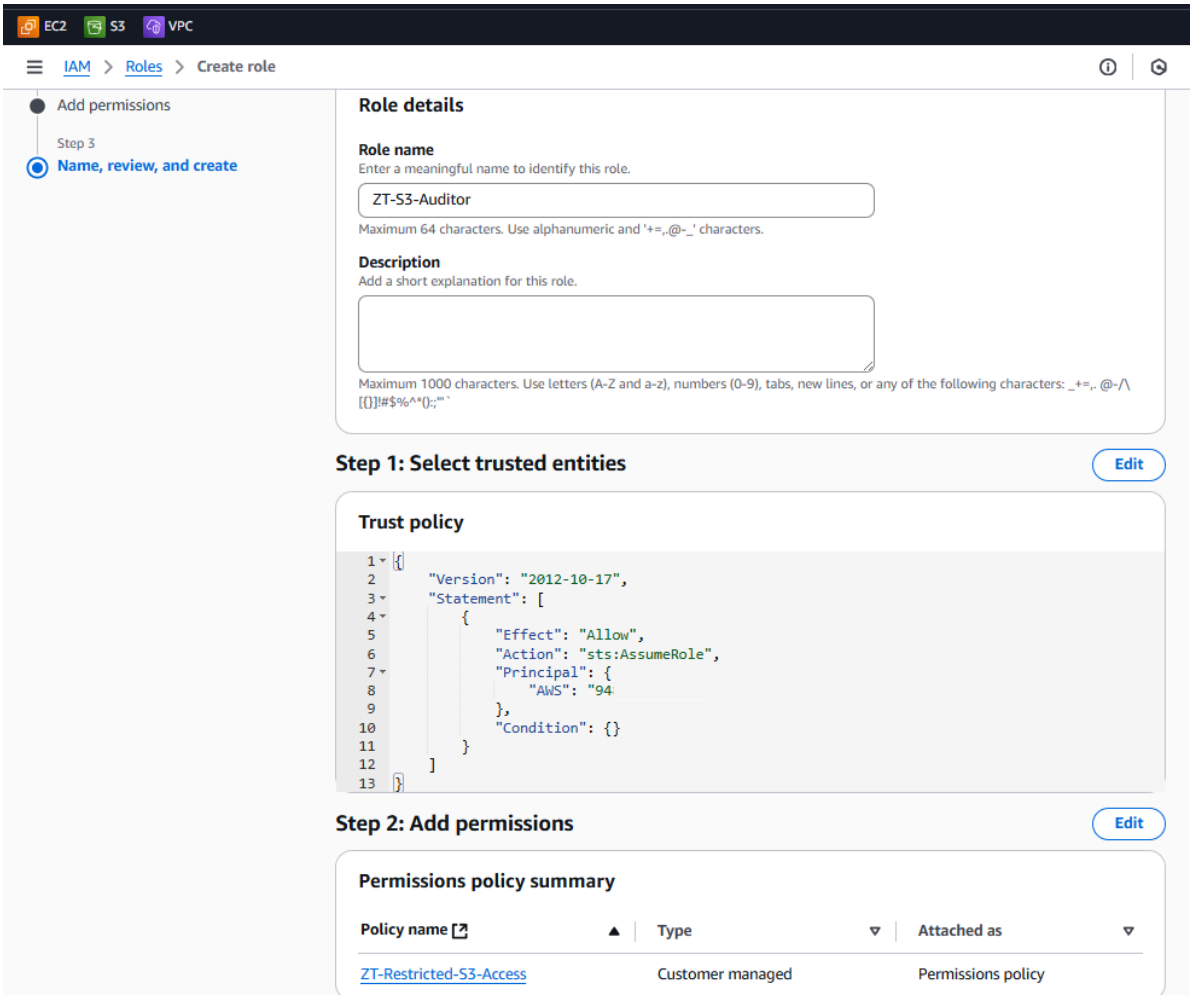


📘 *Resultado esperado:*

Tenemos una política IAM que aplica **acceso condicional** (HTTPS obligatorio), alineado con Zero Trust.

- Paso 3: Crear un rol IAM para acceso controlado

1. En **IAM** → **Roles** → **Create role**.
2. Tipo de entidad: **AWS Account** → selecciona “This account”.
3. Asignamos la política que creamos anteriormente "ZT-Restricted-S3-Access).
4. Le nombraremos ZT-S3-Auditor.

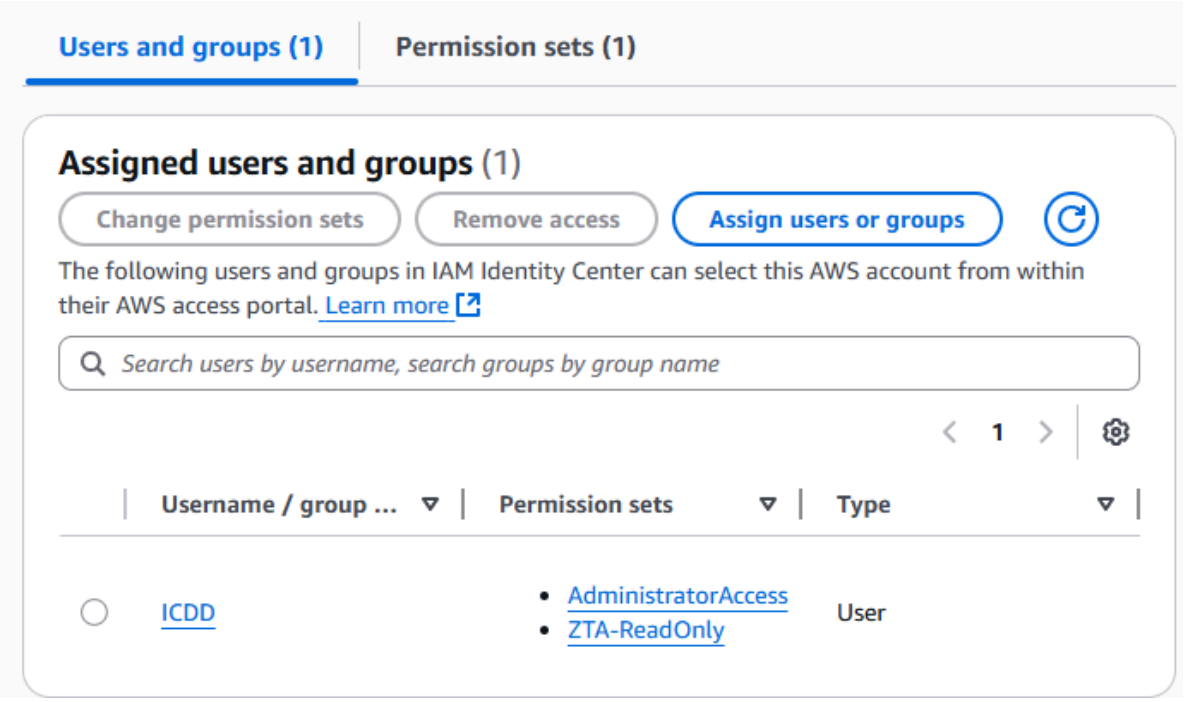


📘 *Resultado esperado:*

Creamos un rol con permisos mínimos que servirá como **Policy Enforcement Point (PEP)**

- Paso 4: Asignar ese rol a un usuario del Identity Center

1. Volvemos a **IAM Identity Center → AWS Accounts → Assign users**.
2. Escogemos nuestra cuenta AWS.
3. Seleccionamos a nuestro usuario ICDD
4. Asignamos un nuevo **Permission set** y elegimos el rol ZT-S3-Auditor



📘 *Resultado esperado:* Nuestro usuario ahora puede cambiar entre dos roles:

- `AdministratorAccess`
- `ZT-S3-Auditor` (acceso restringido a S3 bajo Zero Trust).

Esto refleja el modelo Zero Trust de separación de privilegios por contexto.

- Paso 5: Auditar usuarios, roles y permisos activos

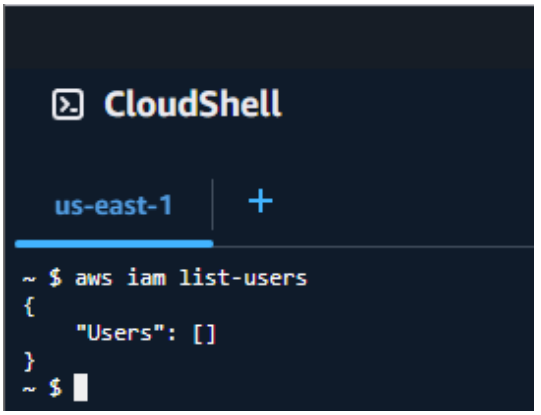
Con AWS CLI, ejecutamos los siguientes comandos:

```
aws iam list-users
aws iam list-roles
aws iam list-policies --scope Local
aws iam generate-credential-report
```

Esto nos da una visión de:

- Qué usuarios existen.
- Qué roles están definidos.
- Qué políticas están activas.

```
aws iam list-users
```

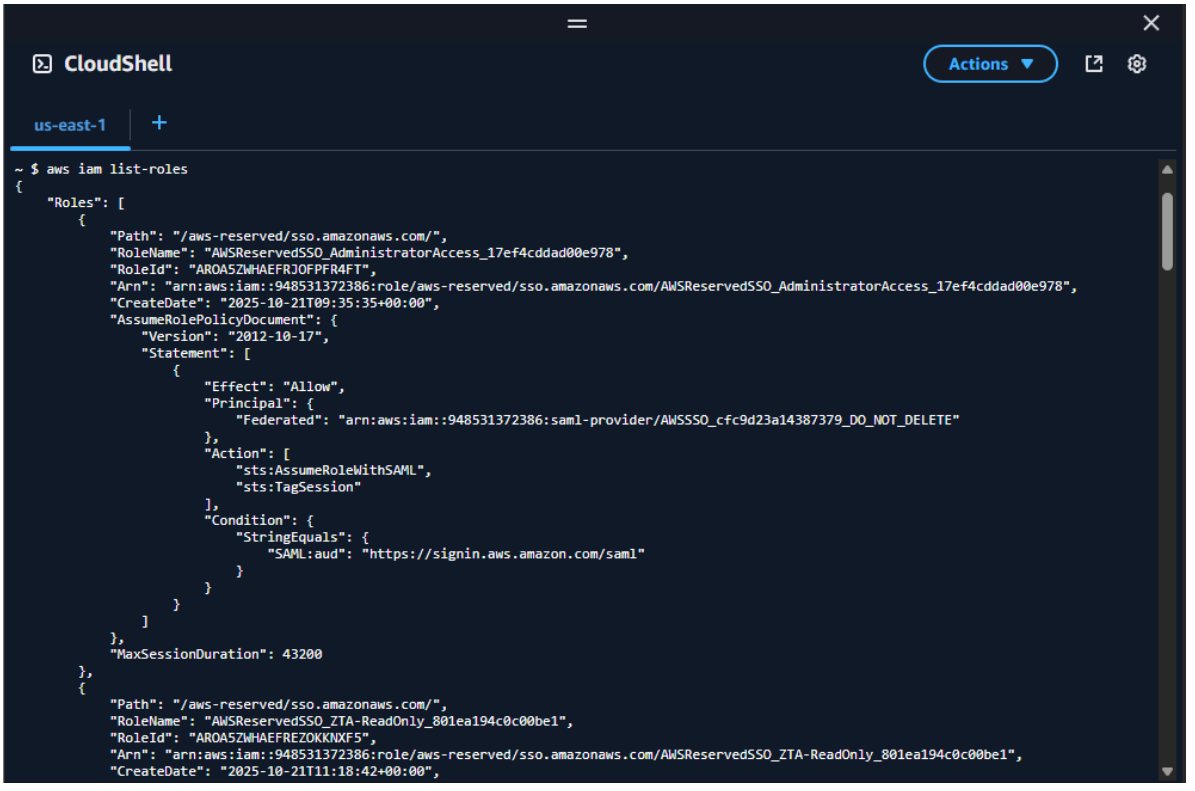


Esta salida indica que no hay usuarios IAM clásicos creados en la cuenta. Esta salida es correcta si estamos usando AWS Identity Center (SSO), porque:

- Los usuarios de Identity Center no se gestionan en IAM clásico, sino en su propio directorio interno.
- Por eso el comando aws iam list-users solo muestra usuarios IAM tradicionales, no los de Identity Center.

Podemos decir que nuestro entorno está limpio, sin usuarios IAM permanentes, lo cual es ideal para Zero Trust (evita credenciales estáticas).

```
aws iam list-roles
```

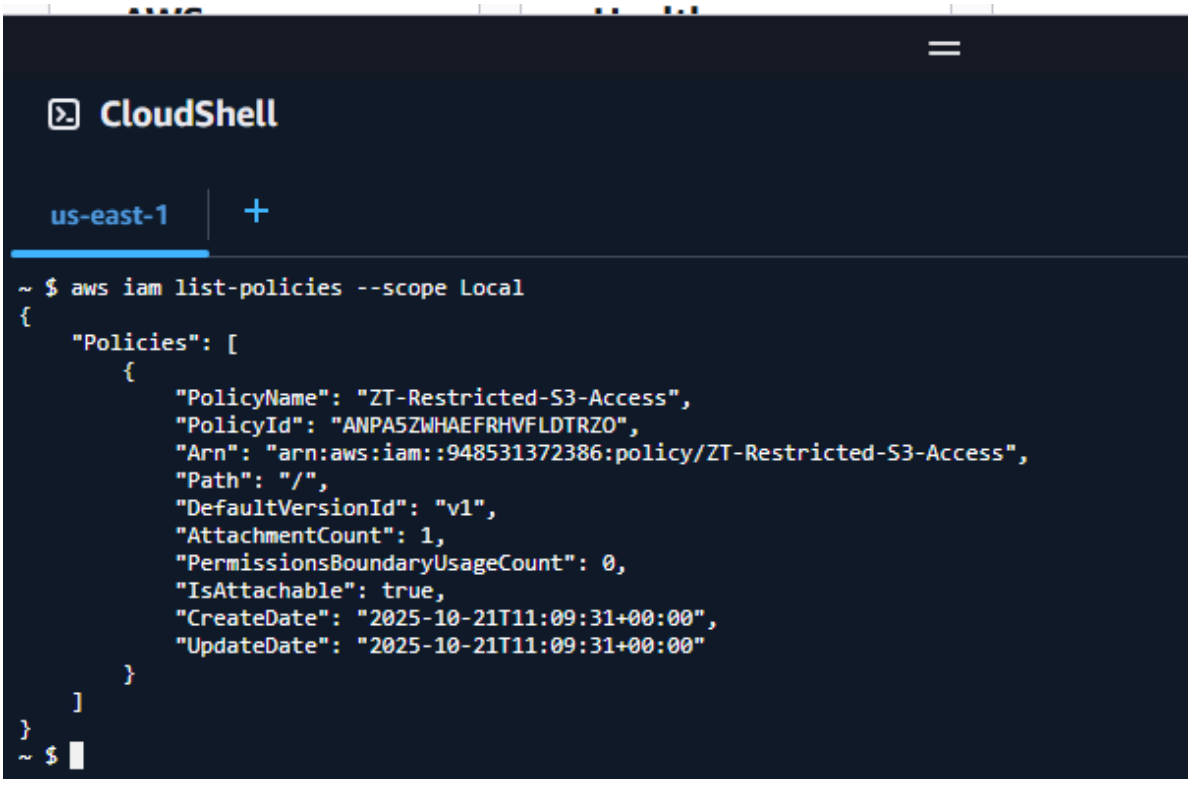


Este comando devuelve todos los roles IAM definidos en la cuenta, cada rol representa un conjunto de permisos que pueden asumir usuarios o servicios de AWS. Buscando bien en

la salida que nos dió podemos encontrar el **Permission set** que creamos hace un momento "ZTA-ReadOnly"

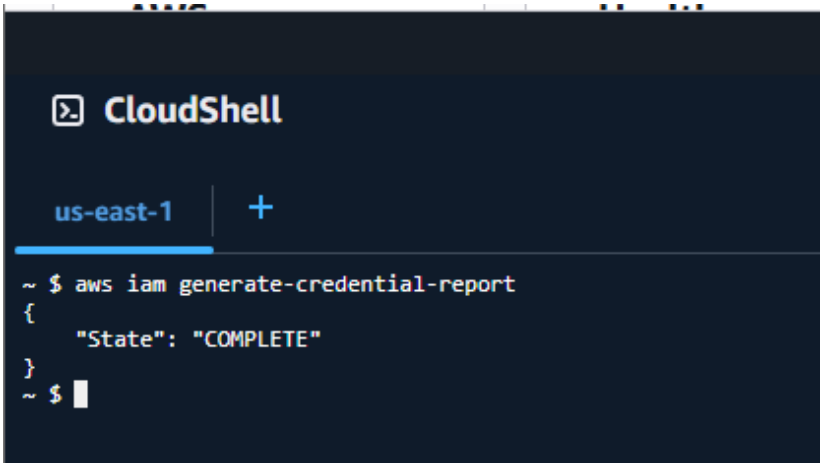

```
aws iam list-policies --scope Local
```

Este comando muestra solo las políticas personalizadas que hemos creado en esta cuenta (no las políticas predeterminadas de AWS). Este comando es super útil para auditar nuestras propias reglas de seguridad y asegurarnos de que solo existen las necesarias.



```
aws iam generate-credential-report
```

Este comando genera un reporte de credenciales con el estado de seguridad de todos los usuarios IAM (si tienen MFA, claves activas, etc).



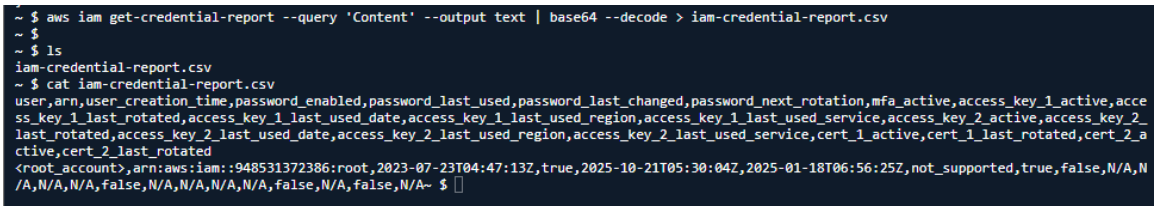
En nuestro caso, nuestra salida significa que AWS generó correctamente el reporte de credenciales, pero aún no lo descargamos. Lo podemos ver ejecutando el comando:
aws iam get-credential-report
Este comando devolverá un CSV codificado en Base64, algo como esto:

```
{
"Content": "QWNjb3VudCBOYW1lLENyZWF0aW9uIERhdGU...==",
"ReportFormat": "text/csv",
"GeneratedTime": "2025-10-21T12:34:56+00:00"
}
```

Podemos decodificarlo con el siguiente comando:

```
aws iam get-credential-report --query 'Content' --output text | base64 --
decode > iam-credential-report.csv
```

Luego de correr ese comando obtenemos lo siguiente:



Resultado de la Fase 2

Elemento Configurado	Componente de Zero Trust	Descripción
Identity Center	PAP/PDP	Administra políticas de acceso e identidad.
IAM Policies	PAP	Define permisos mínimos y condicionales.
IAM Roles	PEP	Aplica las políticas en recursos específicos.
MFA	PIP	Aporta información de autenticación contextual.
CLI Audit	CA-7 (NIST 800-53)	Valida configuración y cumplimiento.

Fase 3 - Protección de Datos (S3 + SSE-SS)

🎯 Objetivo

Configurar un almacenamiento seguro (S3) con:

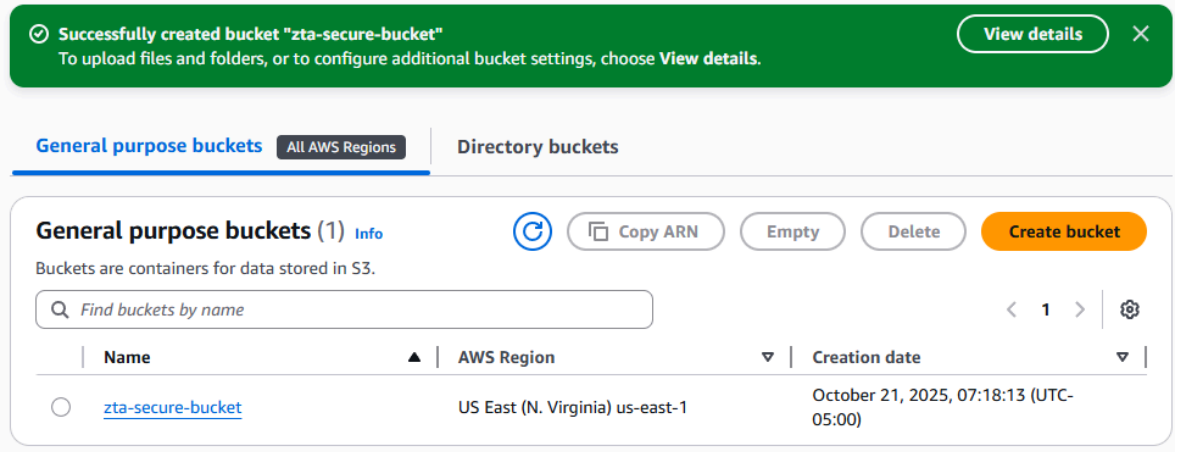
- Acceso restringido bajo políticas Zero Trust.
- Cifrado en reposo mediante KMS o SSE-S3
- Bloqueo total del acceso público.
- Auditoría del uso mediante CloudTrail

- Paso 1: Crear el Bucket Seguro

1. Desde CloudShell o desde la consola podemos crear un bucket nuevo, en este caso vamos a nombrarlo zta-secure-bucket, y lo crearemos con el siguiente comando:

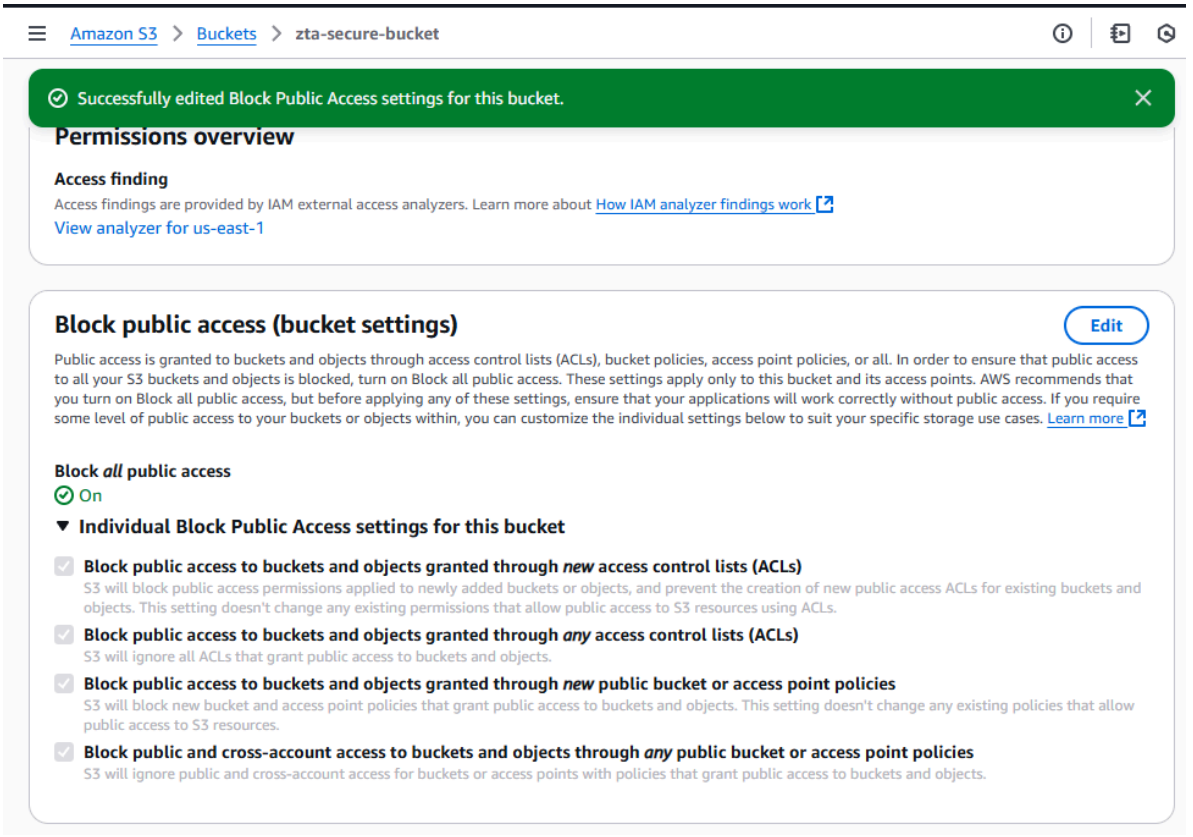
```
aws s3 mb s3://zta-secure-bucket --region us-east-1
```

O desde la consola vamos a S3 → **General purpose buckets** → **Create bucket.**



- Paso 2: Bloquear acceso público

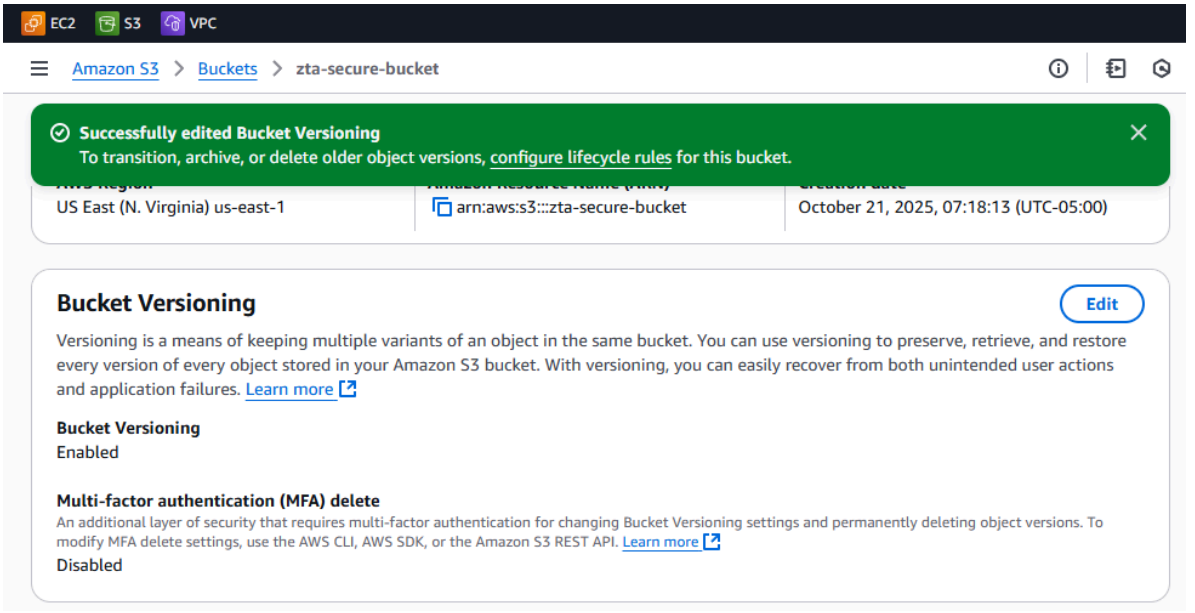
1. Vamos a la consola, → **Amazon S3** → **Nuestro bucket** → **Permissions** → **Block public access.**
2. Activamos todas las opciones de bloqueo:
   - Block all public access
   - Block ACLs
   - Block New public policies.



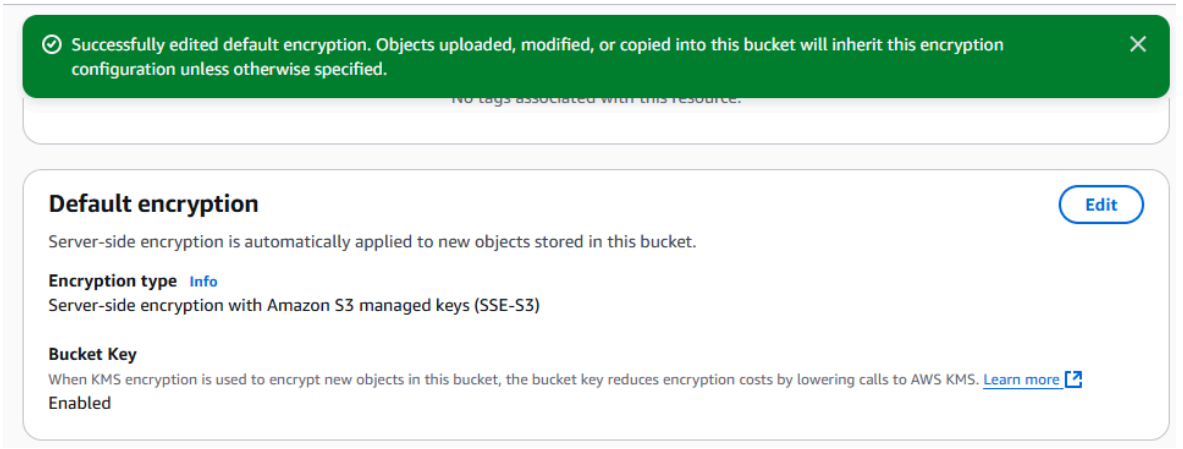
📘 Resultado**:**
Ningún objeto será accesible desde fuera de AWS (PEP: Enforcement de acceso).

- Paso 3: Habilitar Versioning y cifrado

1. Vamos nuevamente a la consola → **Amazon S3** → **Nuestro bucket** →**Properties.**
2. Bucket Versioning → Edit → Enable → Save Changes.



3. Default encryption → Edit → Server-side encryption with Amazon S3 managed keys (SSE-S3) → Save Changes



Nota: Opcional podemos activar el Server-side encryption with AWS Key Management Service keys (SSE-KMS), pero debemos de tener en cuenta de que cambiar las configuraciones de cifrado puede causar problemas de replicación y Batch replication. Eso puede fallar por la falta de permisos de parte de AWS KMS en los roles IAM, hay que tener en cuenta y verificar de que los roles IAM tengan los permisos de AWS KMS necesarios.

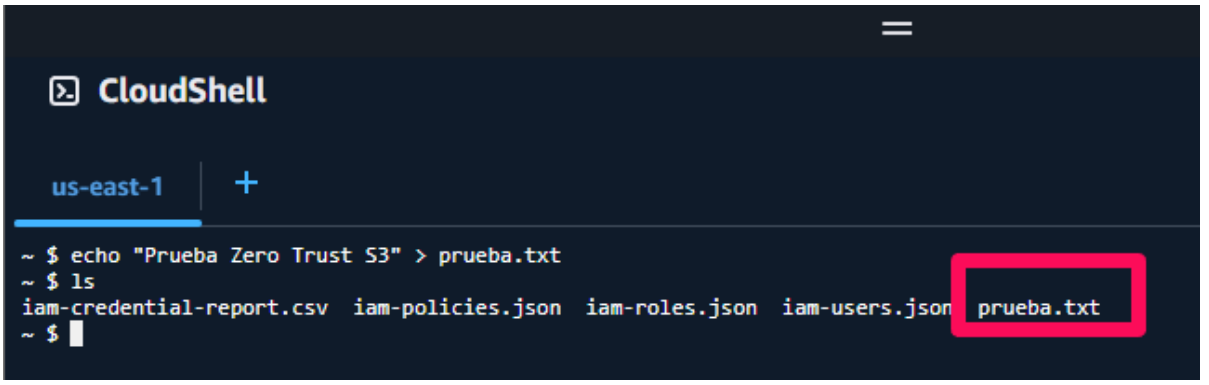
📘 Resultado**:**

Cada archivo que subamos se cifrará automáticamente con AES-256 en el lado del servidor.

- Paso 4: Subir un archivo de prueba y verificar su cifrado

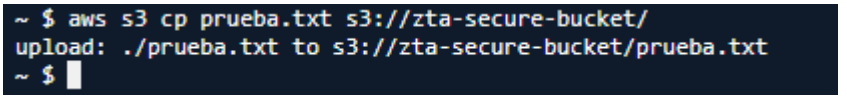
1. Creamos un archivo local en AWS CloudShell:

```
echo "Prueba Zero Trust S3" > prueba.txt
```



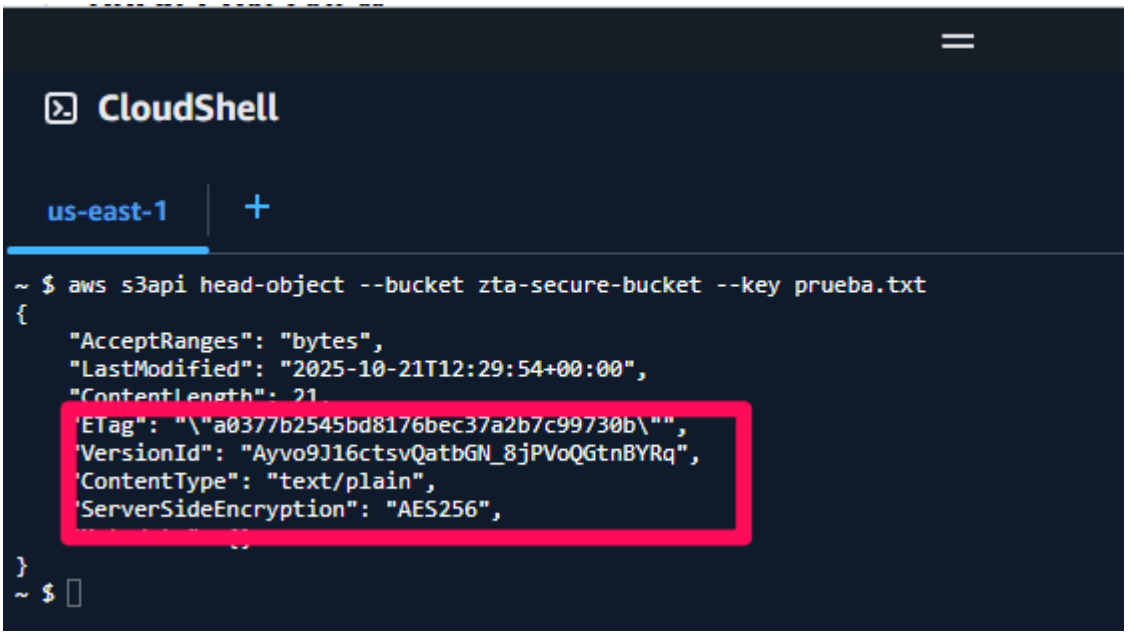
2. Subimos el archivo:

```
aws s3 cp prueba.txt s3://zta-secure-bucket/
```



3. Verificamos su cifrado:

```
aws s3api head-object --bucket zta-secure-bucket --key prueba.txt
```



📘 Resultado esperado:
El archivo se almacena cifrado y versionado.

- Paso 5: Crear Política de acceso Zero Trust para S3

1. Crearemos un archivo zt-s3-policy.json con el siguiente contenido:

```
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "DenyUnsecuredTransport",
"Effect": "Deny",
"Principal": "*",
"Action": "s3:*",
"Resource": [
"arn:aws:s3:::zta-secure-bucket",
"arn:aws:s3:::zta-secure-bucket/*"
],
"Condition": {
"Bool": {"aws:SecureTransport": "false"}
}
},
{
"Sid": "DenyUnencryptedUploads",
"Effect": "Deny",
"Principal": "*",
"Action": "s3:PutObject",
"Resource": "arn:aws:s3:::zta-secure-bucket/*",
"Condition": {
"StringNotEquals": {"s3:x-amz-server-side-encryption": "AES256"}
}
}
]
}
```

2. Aplicamos la política con el siguiente comando:

```
aws s3api put-bucket-policy --bucket zta-secure-bucket --policy file://zta-s3-policy.json
```

📘 Resultado:

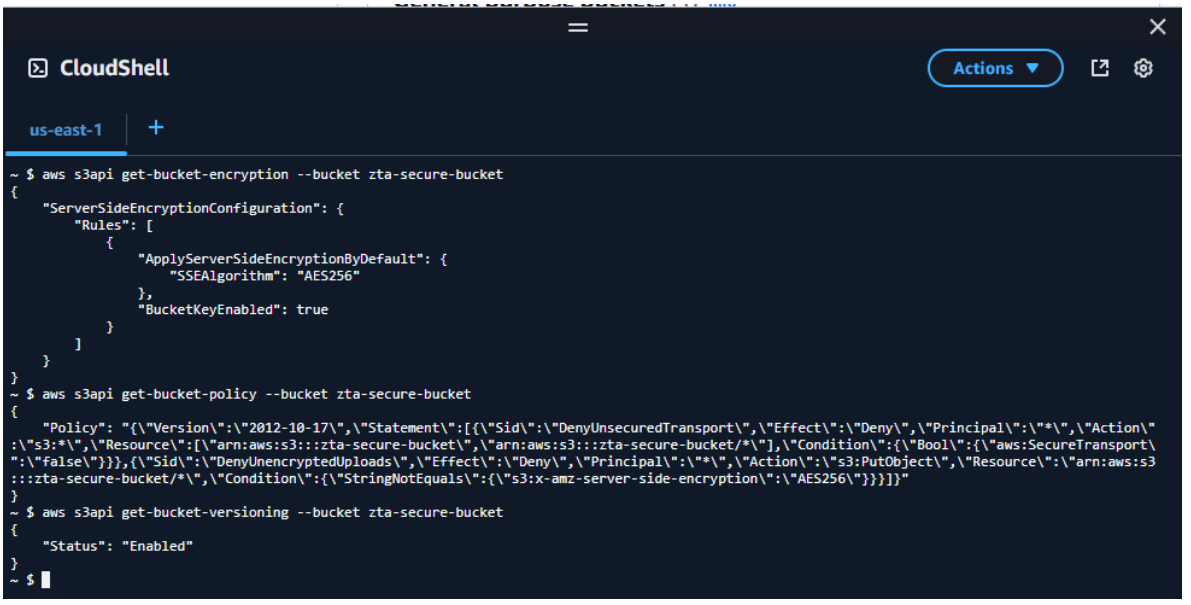
Solo se permiten conexiones seguras y objetos cifrados. Esto en definitiva es el **Zero Trust Enforcement Point (PEP)** en acción.

- Paso6: Validar la configuración

1. Verificamos que todo esté activo con los siguientes comandos:

```
aws s3api get-bucket-encryption --bucket zta-secure-bucket
aws s3api get-bucket-policy --bucket zta-secure-bucket
aws s3api get-bucket-versioning --bucket zta-secure-bucket
```

2. Salidas:

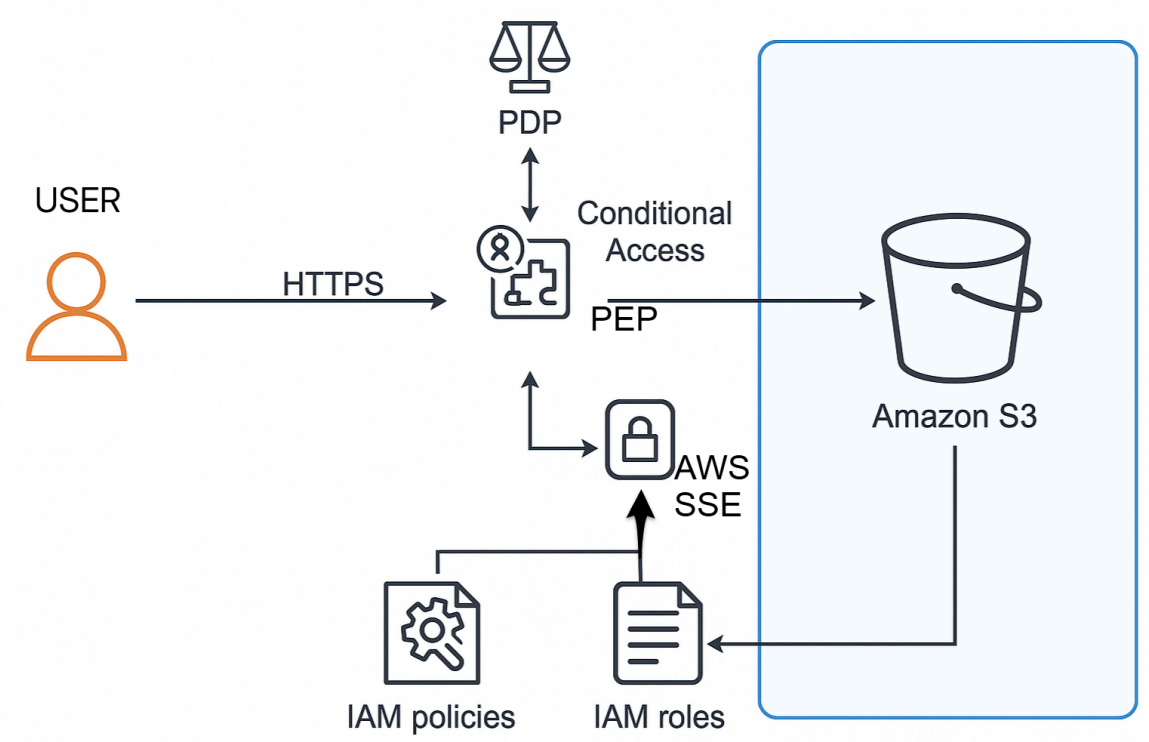


Resultados de la Fase 3

Control Aplicado	Propósito	Componente Zero Trust
Bloqueo Público	Impide acceso implícito	PEP
Cifrado SSE/KMS	Protege datos en reposo	PAP/PDP
Políticas condicionales	Exige HTTPS y cifrado	PDP/PEP
Versioning	Control de integridad	PIP
Auditoría con CLI	Evidencia de cumplimiento	CA-7 (NIST)

Hasta ahora tenemos:

- Identidad Segura.
- Almacenamiento cifrado y segmentado.



Este diagrama refleja lo siguiente:

- El usuario accediendo mediante HTTPS y autenticado con MFA (Aunque no se muestre en el diagrama)
- IAM Policies & Roles actuando como PAP (Policy Administrator Point) y PEP (Policy Enforcement Point)
- PDP (Policy Decision Point) tomando decisiones de acceso condicional.

- AWS SS3/KMS Proporciona cifrado gestionado de datos.
- Amazon S3 como almacenamiento protegido, con políticas de acceso cifrado y auditoría continua.

Fase 4: Segmentación de red con VPC y Security Groups

🎯 Objetivos

En esa fase vamos a construir y aislar recursos en una subred pública (bastion) y una subred provada (DB), y aplicaremos controles de red mínimamente necesarios y preparar todo para colocar una RDS.

Básicamente crearemos una red segura basada en Zero Trust: microsegmentación, acceso mínimo y controlado.

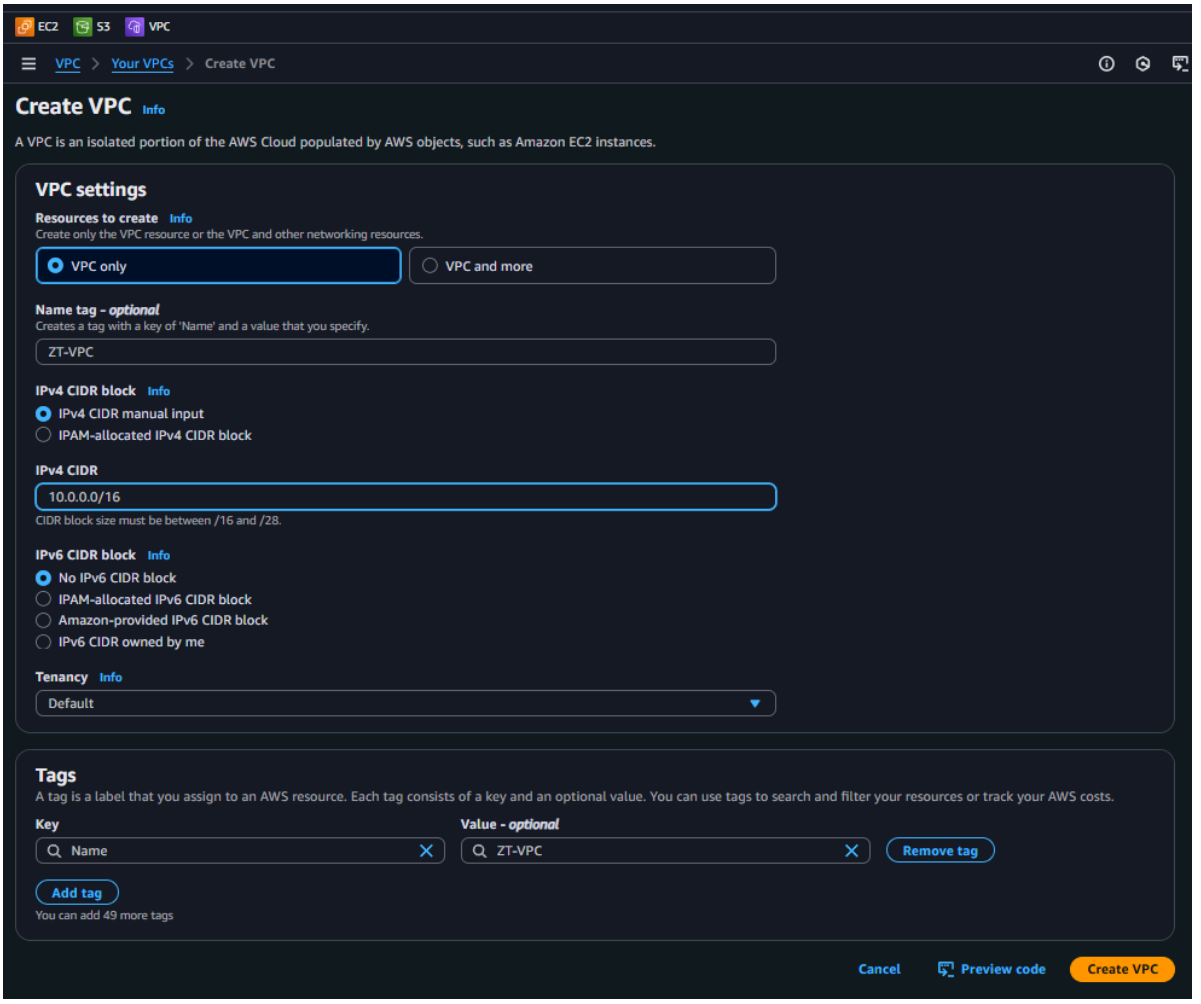
¿Qué vamos a construir?

- VPC (10.0.0.0/16)
  - Subred pública 10.0.1.0/24 → EC2 Bastion (o podemos acceder mediante SSM Session Manager para evitar abrir SSH)
  - Subred privada 10.0.2.0/24 → RDS (Sin IP Publica)
- Un Internet Gateway solo para la subred públca.
- Route tables tables separadas (Una con ruta pública a Internet, y otra privada sin salida pública).
- Security Groups (Este actuará como un PEP):
  - sg-bastion: este security group solo dará acceso desde nuestra IP pública.
  - sg-rds: este security group solo permitirá conexiones a los puertos 3306 y/o 5432 desde sg-bastion.

- Paso 1: Crear la VPC

1. En nuestra consola AWS, abrimos el servicio VPC.
2. En el menú lateral, elegimos **Your VPCs** → **Create VPC**
3. Marcamos la "opción VPC only"
4. Y configuramos:
   - **Name tag**: ZT-VPC
   - **IPv4 CIDR block**: 10.0.0.0/16
   - **Tenancy**: Default

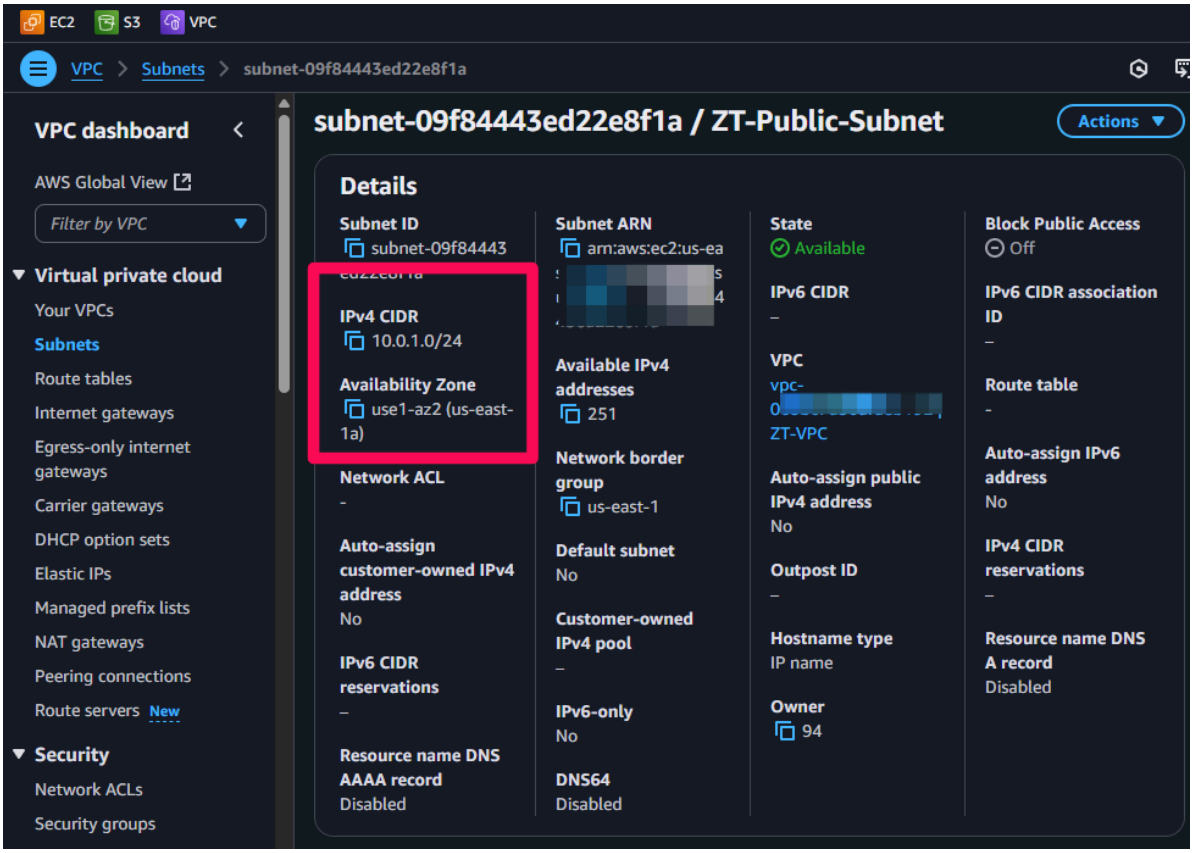
5. Luego de colocar la información, creamos la VPC haciendo click en **Create VPC**.



📘 Nota: Esta será nuestra red principal, en otras palabras podemos verla como una "caja de seguridad" que es donde se aislarán todos los recursos.

- Paso 2: Crear las subredes
- Subred pública

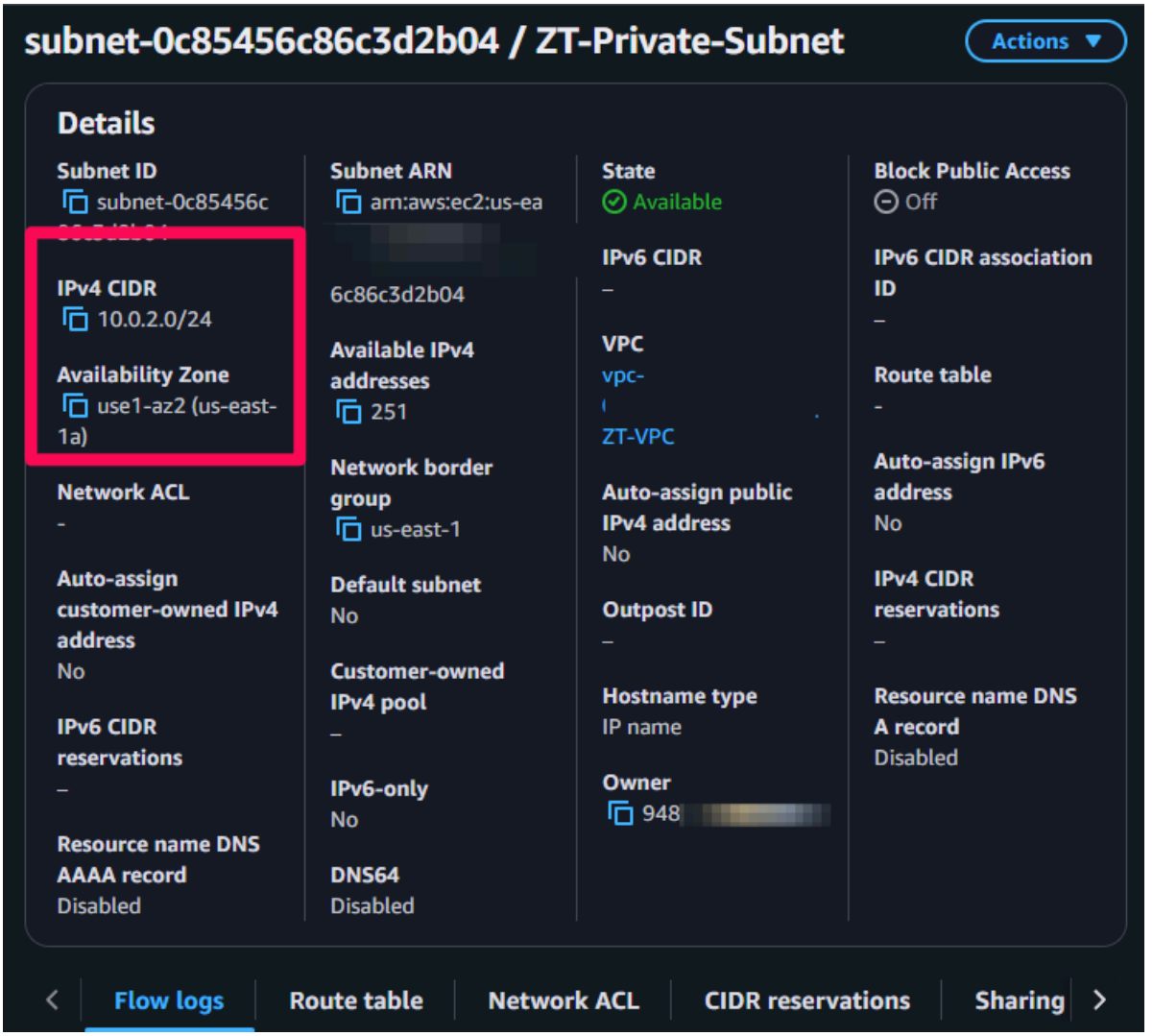
1. En el menú lateral, clickeamos en **Subnets** → **Create subnet**
2. Elegimos la VPC que acabamos de crear: ZT-VPC
3. Llenamos la siguiente información:
   - **Name tag**: ZT-Public-Subnet
   - **Availability Zone**: en este caso usaremos us-east-1a
   - **IPv4 CIDR block**: 10.0.1.0/24
4. Luego clickeamos en Create Subnet.



- Subred privada

1. Repetimos todos los pasos anteriores, pero esta vez colocamos la siguiente información:
   - **Name tag**: ZT-Private-Subnet
   - **Availability Zone**: us-east-1a
   - **CIDR block**: 10.0.2.0/24

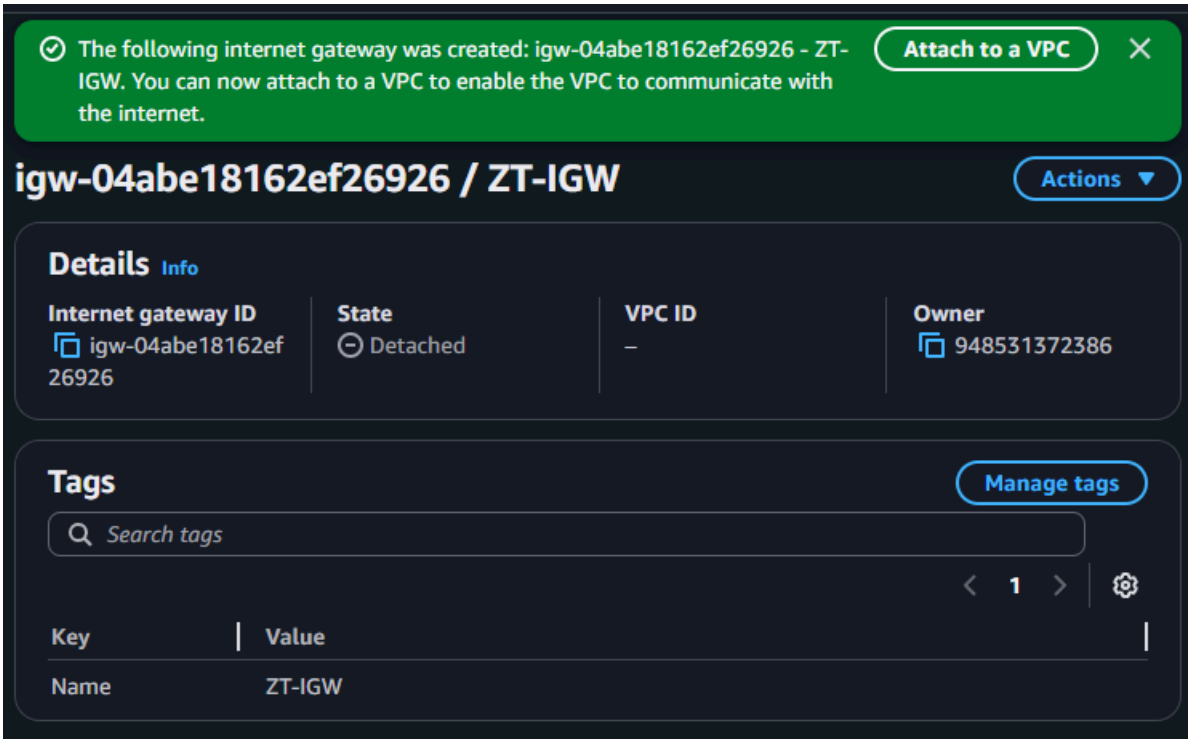
2. Y hacemos click en **Create Subnet**.



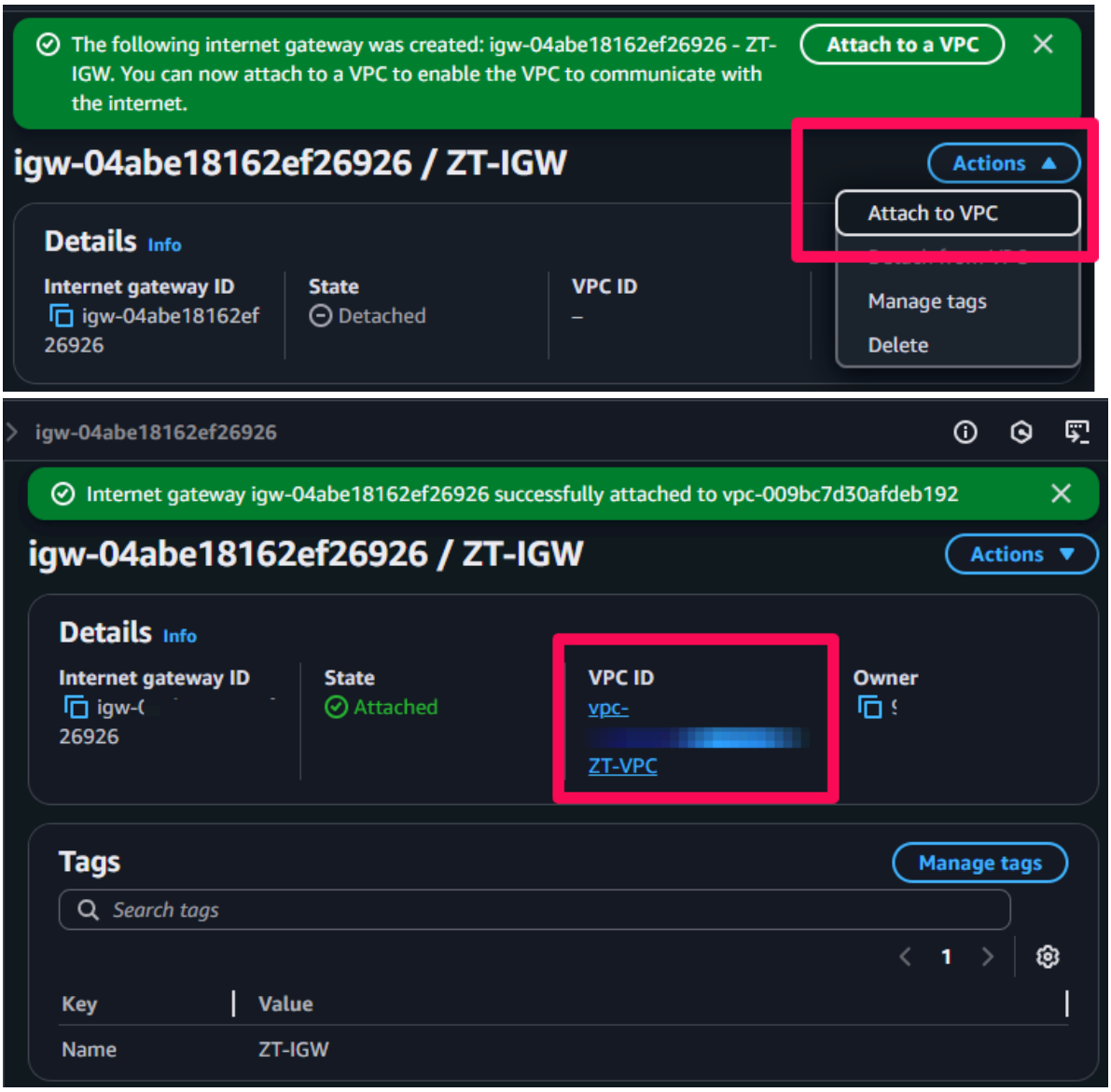
📘 La subred pública permitirá conexiones controladas, mientras que la subred privada servirá para nuestra base de datos, sin acceso a internet.

- Paso 3: Crear y asociar un Internet Gateway

1. En el panel lateral, clickeamos en **Internet Gateways** → **Create Internet gateway**
2. Configuramos:
   - Name tag: ZT-IGW



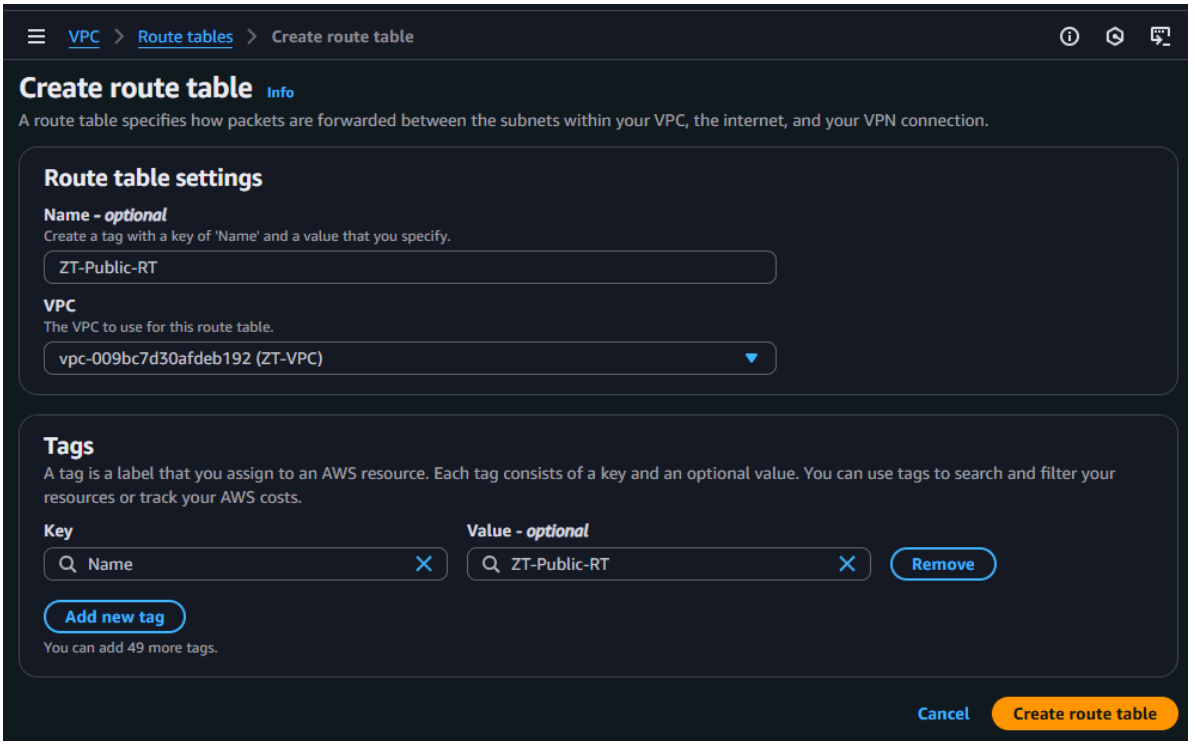
3. Clickeamos en **Create internet gateway**.
4. Luego lo seleccionamos nuevamente → **Actions** → **Attach to VPC** → **ZP-VPC**.



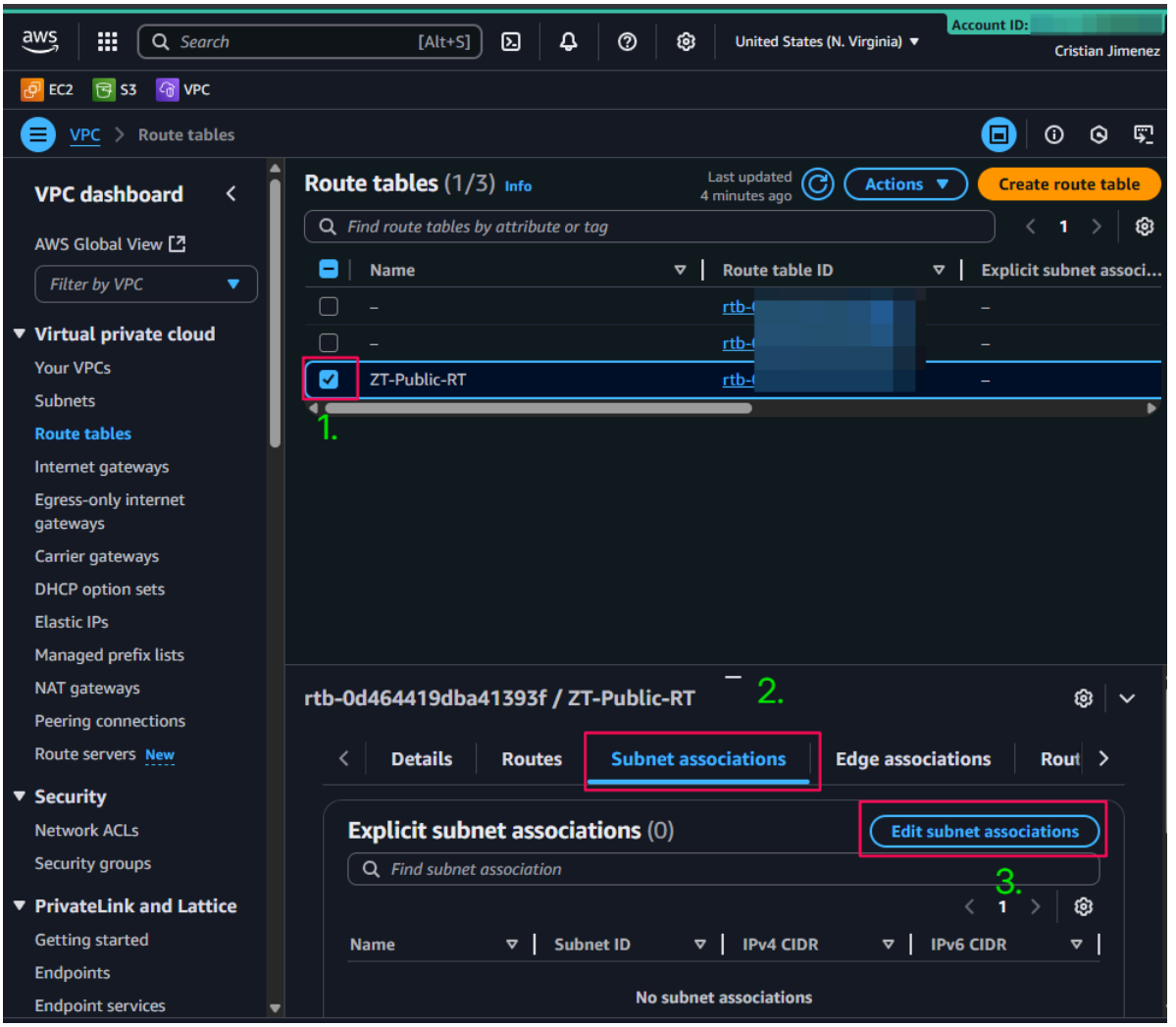
📘 Esto permitirá que los recursos de la subred pública puedan comunicarse con Internet (Todo esto depende de la configuración de los Security Group, podemos crearlos de tal forma que permitan o nieguen la comunicación con Internet).

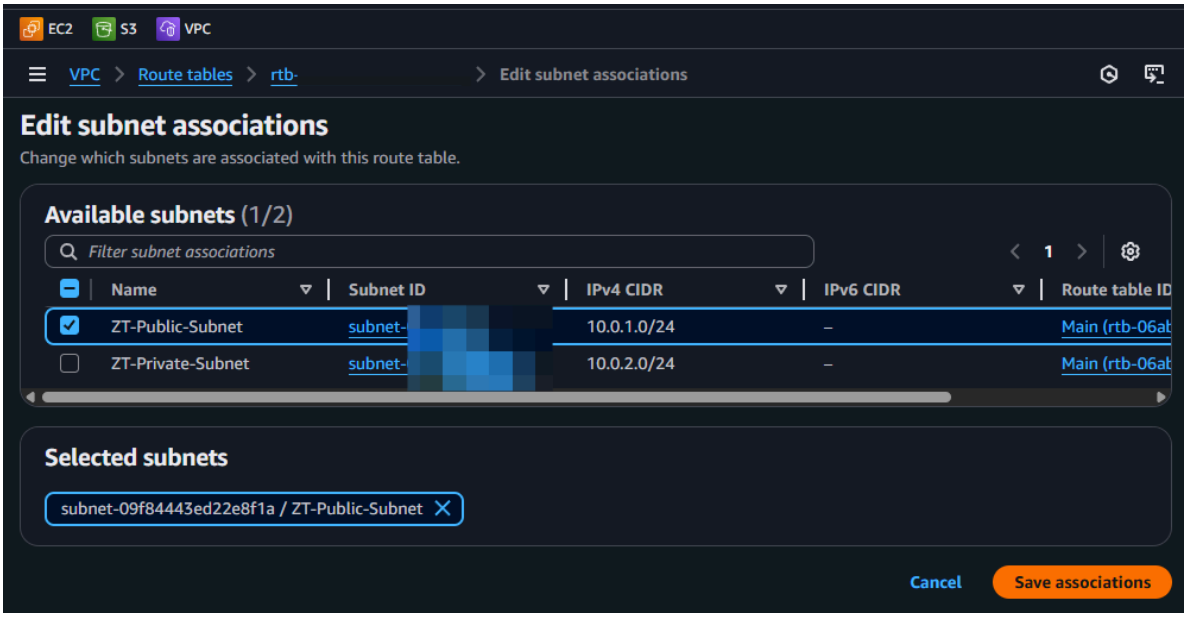
- Paso 4: Configurar las Route Tables
- Route Table pública

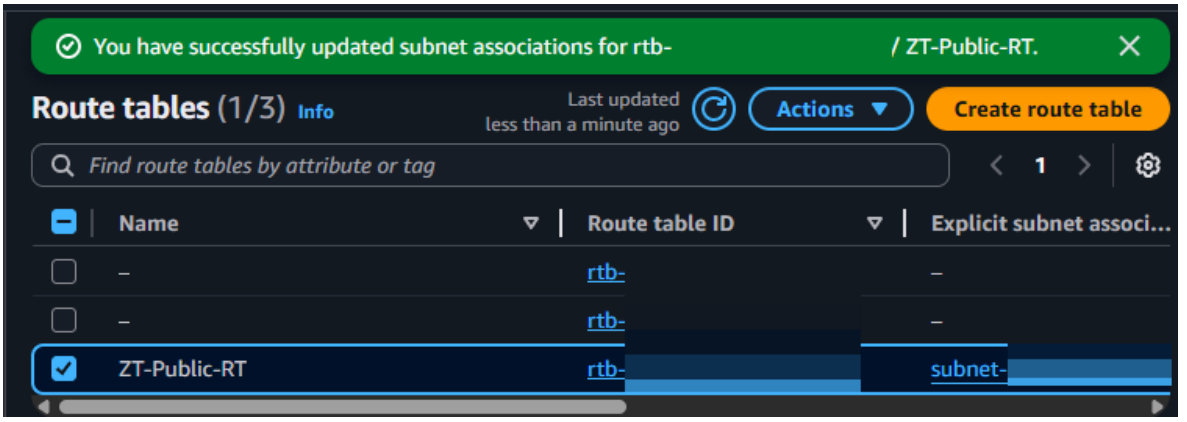
1. En el menú lateral, clickeamos en **Route Tables** → **Create route table**.
2. Configuramos:
   - **Name tag**: ZT-Public-RT
   - **VPC:** ZT-VPC



3. Clickeamos en **Create route table**.
4. Una vez creada, la seleccionamos → pestaña de **Routes** → **Edit routes** → **Edit subnet associations**
    - Asociamos la subred **ZT-Public-Subnet**


aws
Search
[Alt+S]
United States (N. Virginia)
Account ID:
Cristian Jimenez
EC2
S3
VPC
VPC > Route tables
VPC dashboard
AWS Global View
Filter by VPC
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
NAT gateways
Peering connections
Route servers New
Security
Network ACLs
Security groups
PrivateLink and Lattice
Getting started
Endpoints
Endpoint services
Route tables (1/3) Info
Last updated 4 minutes ago
Actions
Create route table
Find route tables by attribute or tag
Name
Route table ID
Explicit subnet associ...
ZT-Public-RT
1.
rtb-0d464419dba41393f / ZT-Public-RT
2.
Details
Routes
Subnet associations
Edge associations
Rout
Explicit subnet associations (0)
Edit subnet associations
Find subnet association
3.
Name
Subnet ID
IPv4 CIDR
IPv6 CIDR
No subnet associations

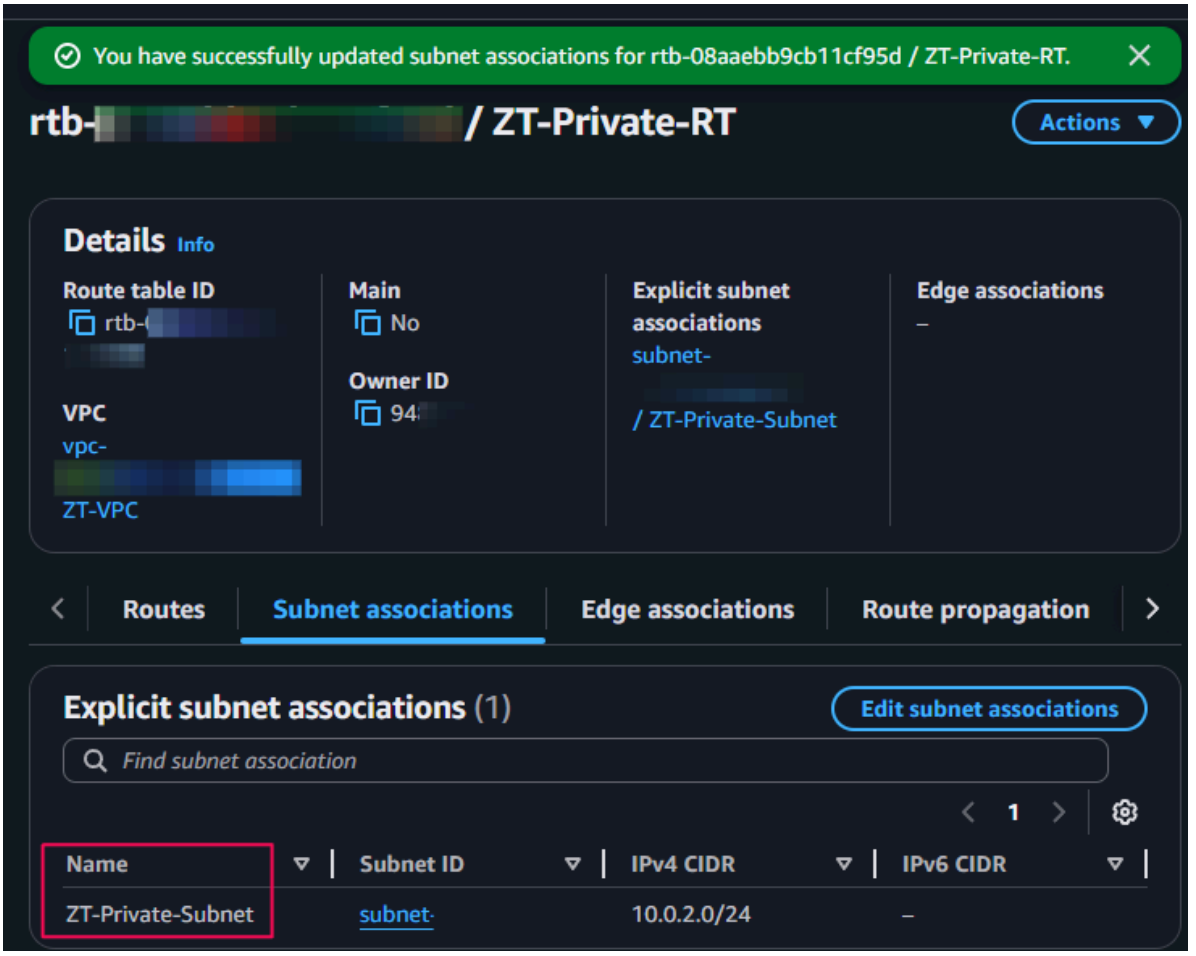

EC2
S3
VPC
VPC > Route tables > rtb-
Edit subnet associations
Edit subnet associations
Change which subnets are associated with this route table.
Available subnets (1/2)
Filter subnet associations
Name
Subnet ID
IPv4 CIDR
IPv6 CIDR
Route table ID
ZT-Public-Subnet
10.0.1.0/24
Main (rtb-06ab
ZT-Private-Subnet
10.0.2.0/24
Main (rtb-06ab
Selected subnets
subnet-09f84443ed22e8f1a / ZT-Public-Subnet
Cancel
Save associations



- Crear Route Table privada

1. Para crear la nueva Route Table seguimos los mismos pasos que hicimos anteriormente, pero con la siguiente información:
   - **Name tag**: ZT-Private-RT
   - **VPC**: ZT-VPC

2. En esta route table no agregaremos ninguna ruta pública.
3. En Subnet associations, vamos a asociar ZT-Private-Subnet.

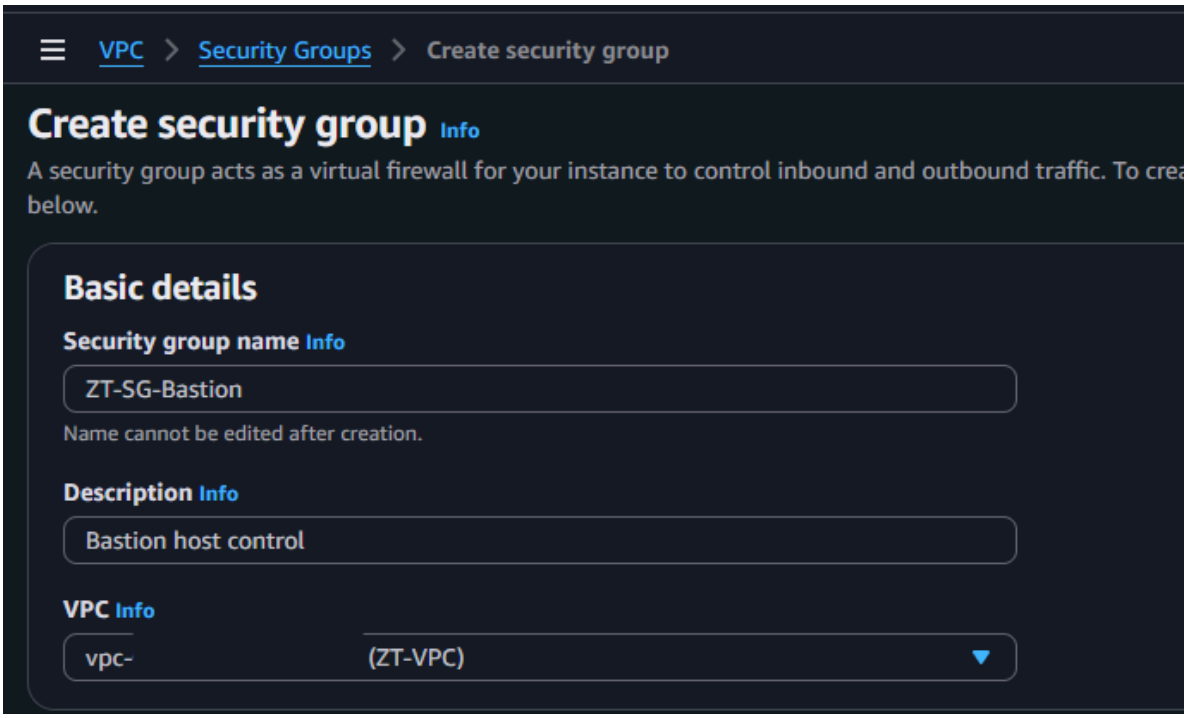


📘 Como resultado tenemos que la subred pública es la única que tiene salida a Internet, mientras que la subred privada está completamente aislada.

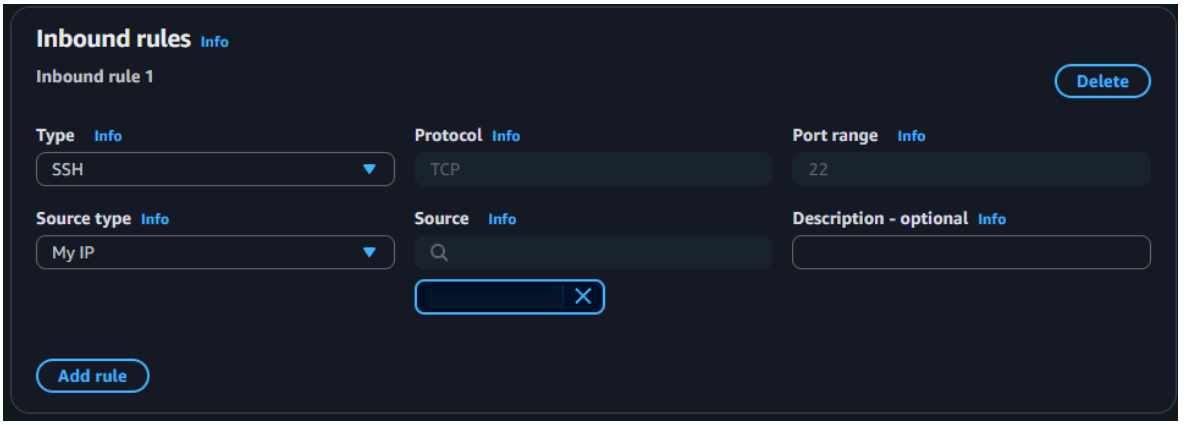
- Paso 5: Crear los Security Groups
- Security Group para Bastion Host

1. En **VPC** → nos ubicamos en el panel de la izquierda y clickeamos **Security Groups** → **Create security group**.
2. Configuramos:

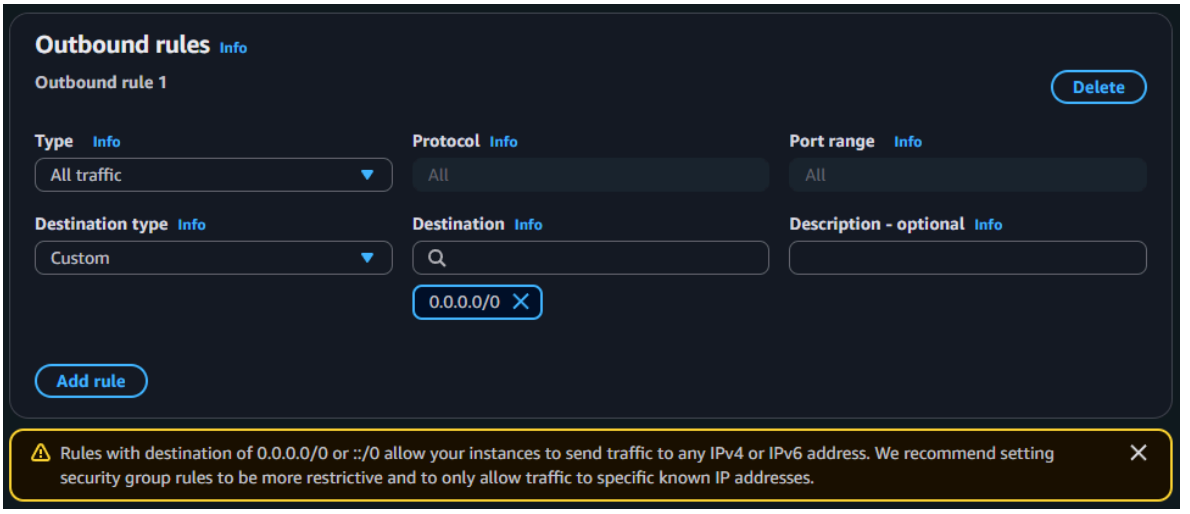
- **Name**: ZT-SG-Bastion
- **Descripción**: Bastion host control
- **VPC**: ZT-VPC



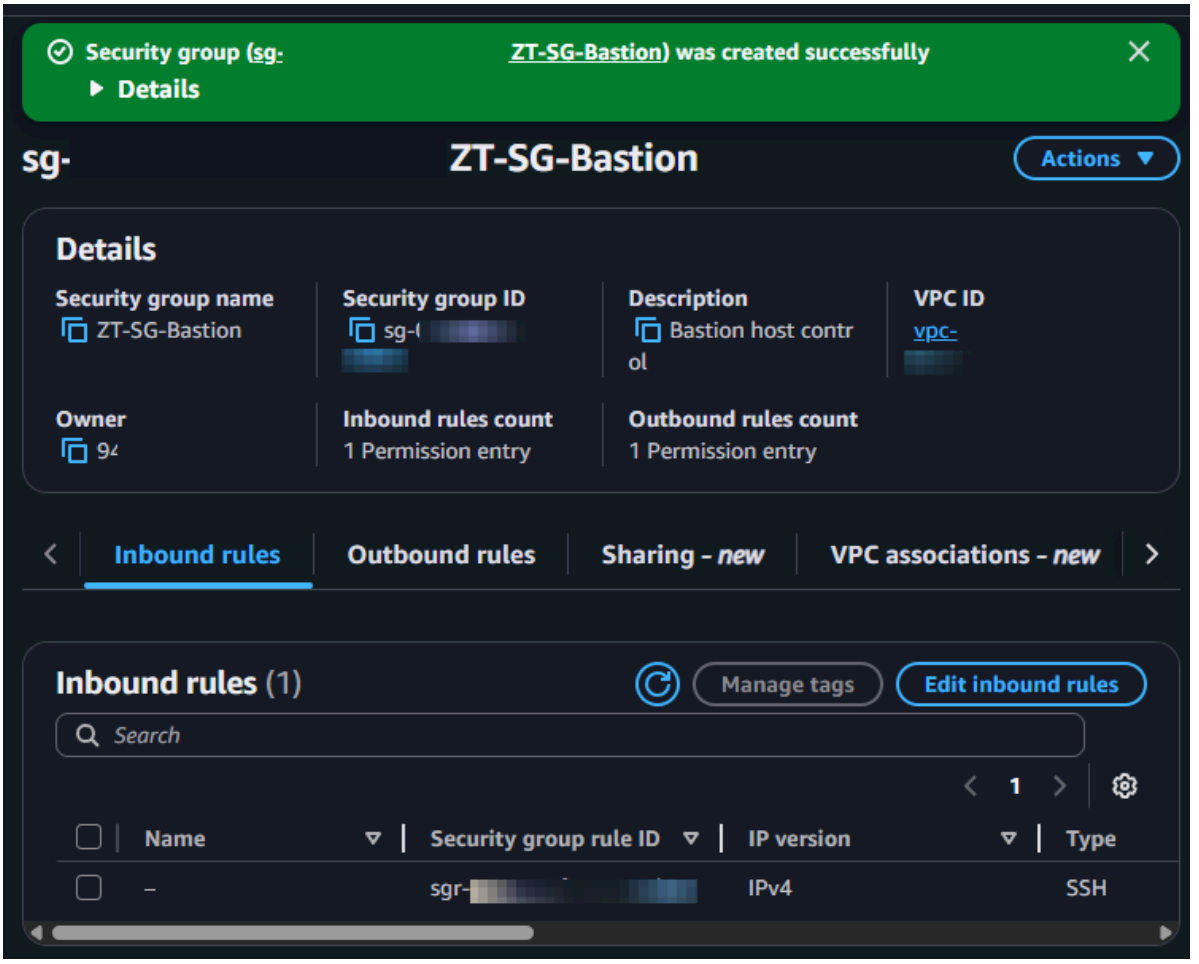
3. Justo abajo, en la sección **Inbound rules** → clickeamos en **Add rule**:
   - **Type**: SSH
   - **Port**: 22
   - **Source**: My IP (Nuestra IP Pública actual)



4. Luego, en **Outbound rules**, dejamos el tráfico abierto (0.0.0.0/0)



5. Guardamos.

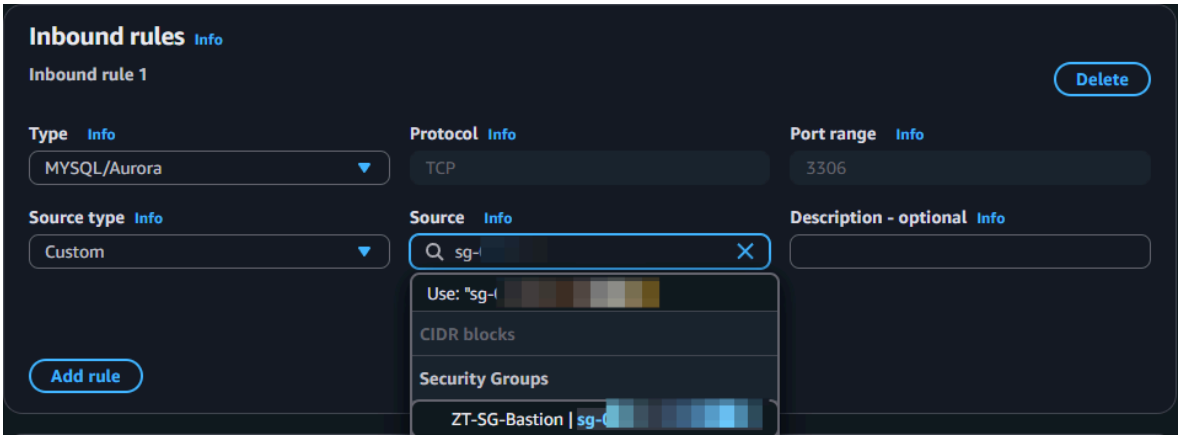


📘 Esta configuración permitirá únicamente conexión SSH desde nuestra IP, evitando accesos externos.

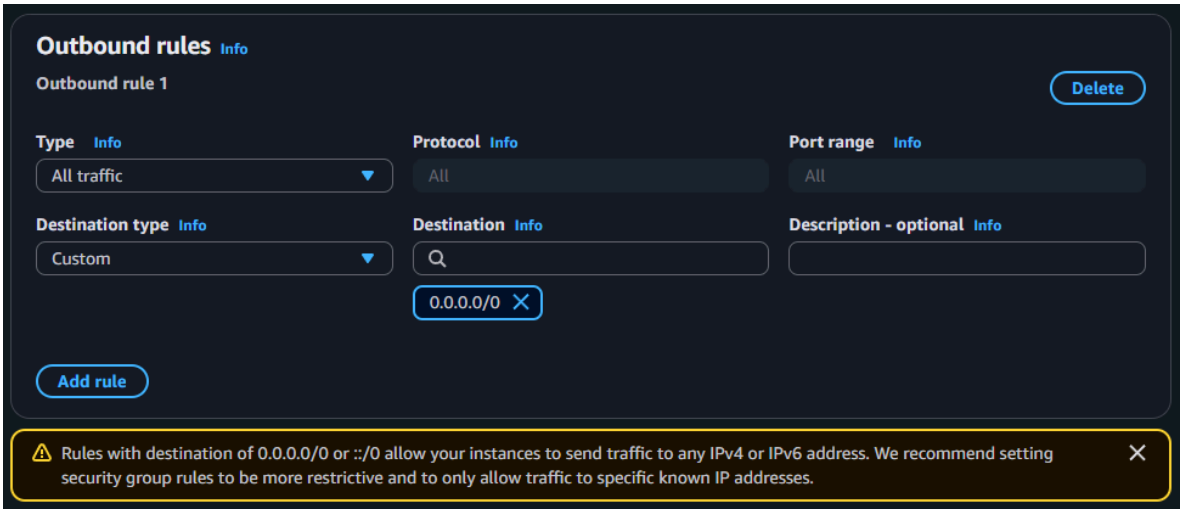
- Security Group para la Base de Datos (RDS)

1. Creamos otro Security Group:
   - **Name**: ZT-SG-RDS
   - **Descripción**: RDS Private access
   - **VPC**: ZT-VPC

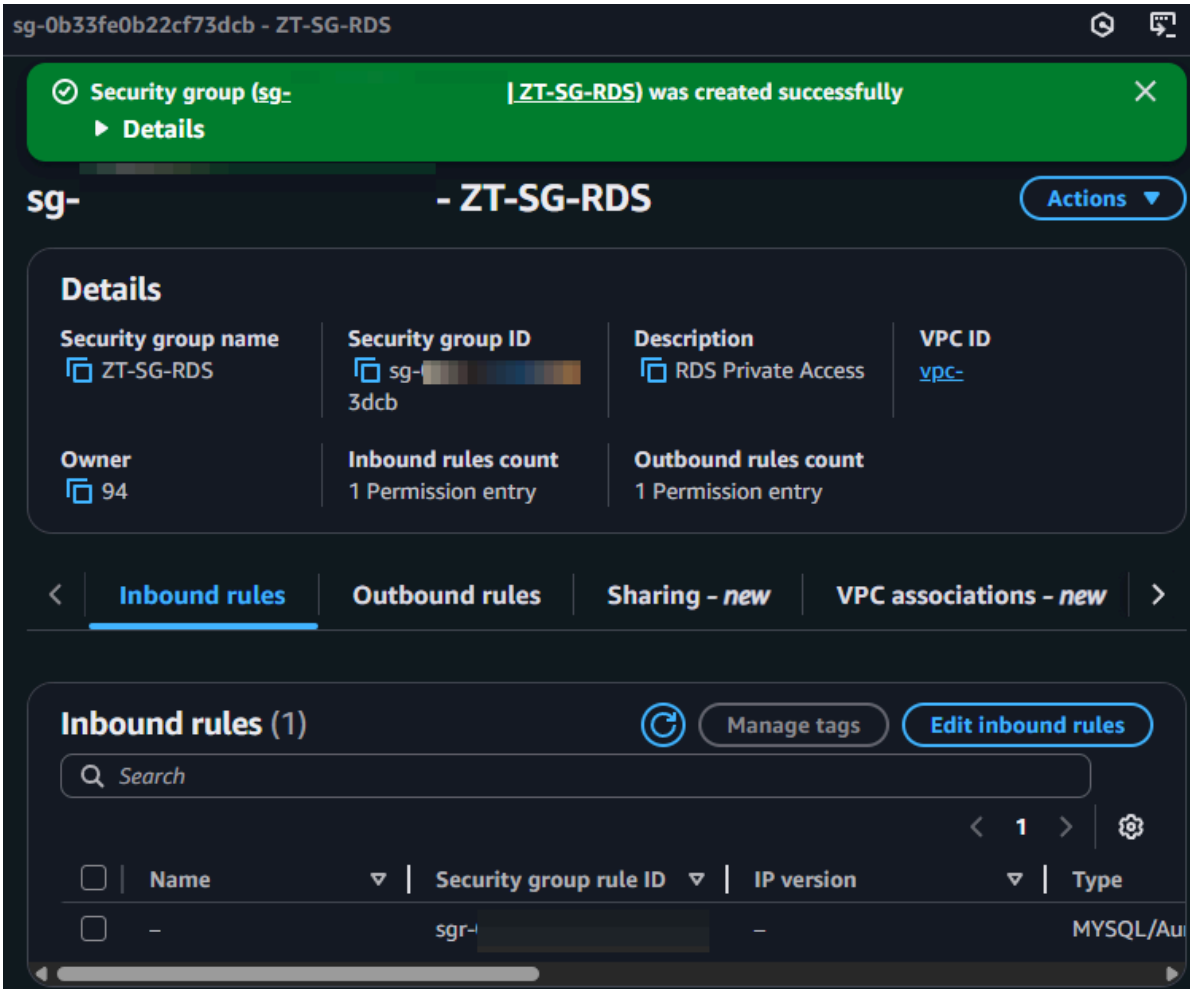
2. En **Inbound Rules** → click en **Add rule**:
   - **Type**: MySQL/Aurora
   - **Port**: 3306 o 5432 (en nuestro caso usaremos el 3306)
   - **Source**: seleccionamos ZT-SG-Bastion (NO CIDR)



3. En **Outbound**, podemos dejarlo por defecto.



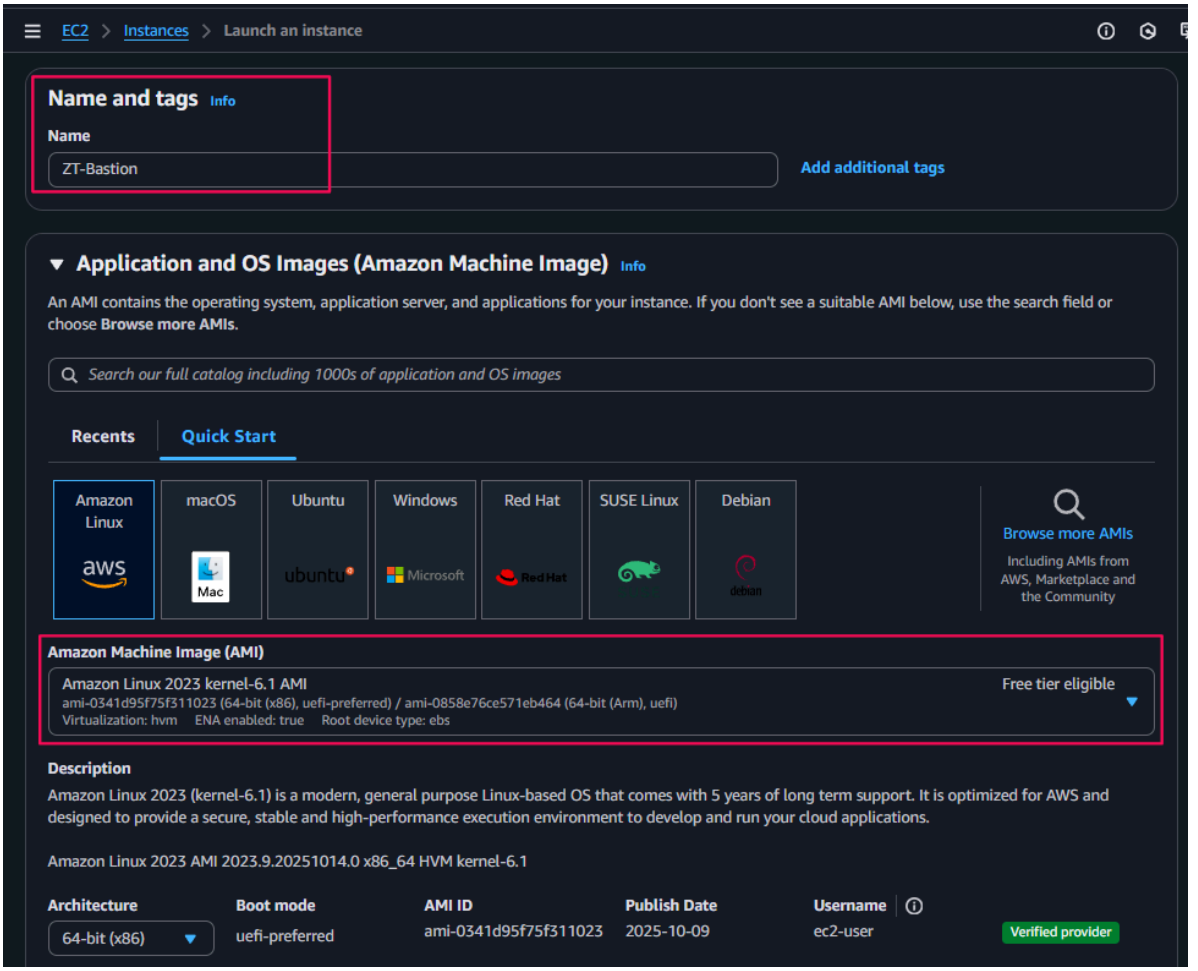
📘 La base de datos solo aceptará conexiones desde el bastion host, no desde Internet.

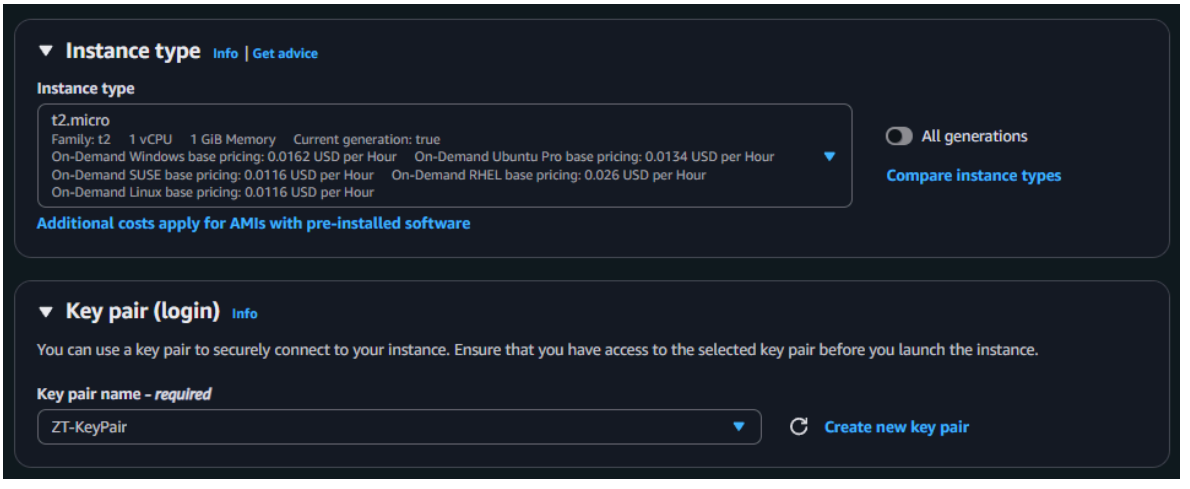


- Paso 6: Crear instancia EC2 Bastion (dentro de la subred pública)

1. Primero abrimos el servicio **EC2** → **Instances** → **Launch instance**.
2. Configuramos así:

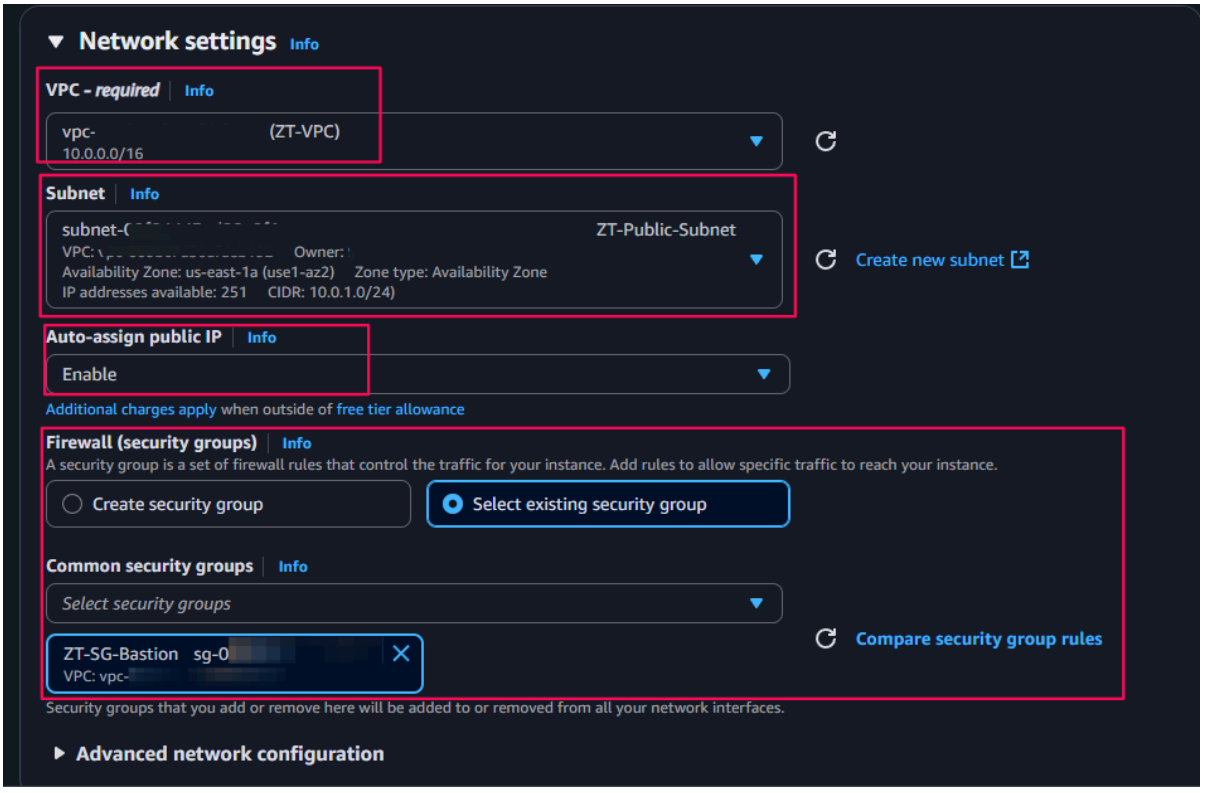
- **Name**: ZT-Bastion
- **AMI**: Amazon Linux 2 (64-bit x86)
- **Instance Type**: t2.micro
- **Key pair**: Seleccionamos o creamos una nueva (en nuestro caso crearemos un nuevo par).





- Network settings:
  - **VPC**: ZT-VPC
  - **Subnet**: ZT-Public-Subnet

- **Auto-assign Public IP**: Enable
- **Security Group**: ZT-SG-Bastion



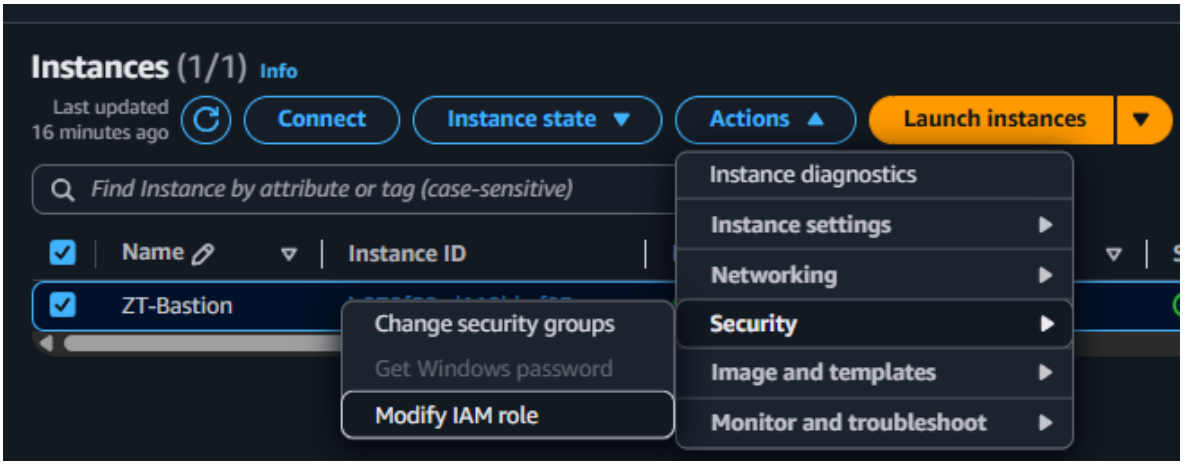
3. En **Advanced details** podemos dejar todo por defecto.
4. Hacemos click en **Launch instance**.

📘 Con lo que acabamos de hacer, creando el Bastion host, nos permitirá conectarnos de forma segura al entorno interno.

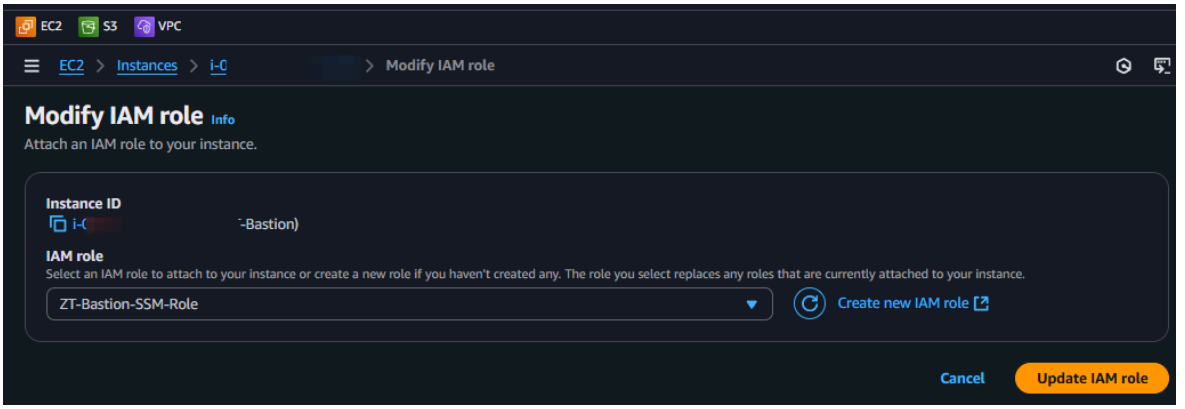
- Paso 7: Habilitar Session Manager (Sin Abrir SSH) (Recomendado)

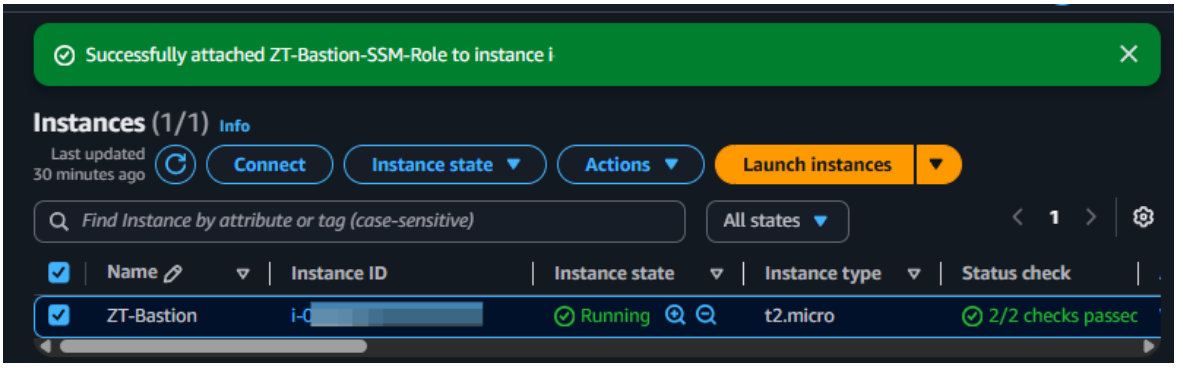
El objetivo de Habilitar el SSM es para conectarnos a nuestra instancia EC2 sin abrir el puerto 22 ni usar claves SSH.
El acceso ocurre mediante AWS Systems Manager (SSM), que usa autenticación IAM + MFA, cumpliendo el principio "nunca confíes, siempre verifica" de Zero Trust.

1. Vamos a **EC2** → **Instances** → Seleccionamos nuestra instancia → **Actions** → **Security** → **Modify IAM role**.

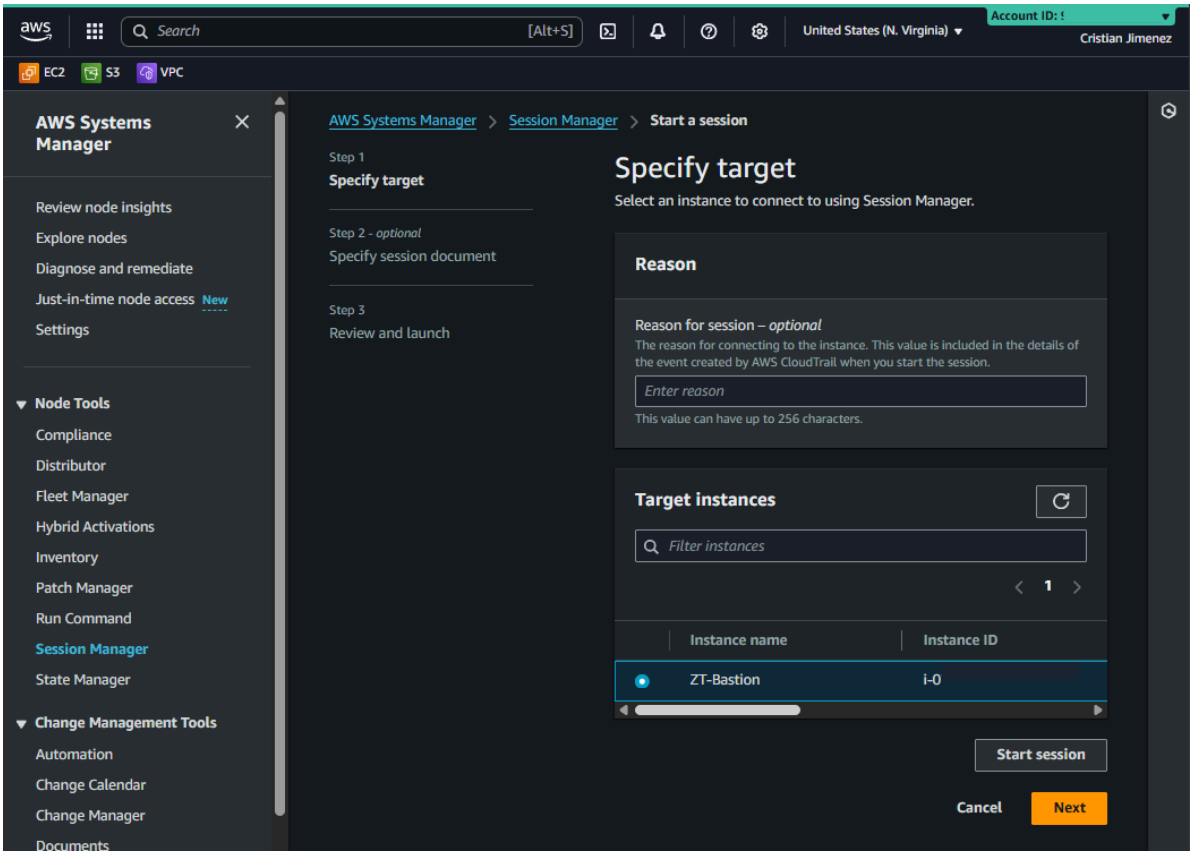


2. Asignamos el rol administrado **AmazonSSMManagedInstanceCore**.
   Nota: Si no tenemos el rol, podemos crearlo directamente desde el servicio IAM. Para tomar en cuenta, en tipo de identidad elegimos AWS Service, en Use case, EC2, y luego asignamos la politica AmazonSSMManagedInstanceCore, le damos un nombre y lo asociamos con la instancia EC2.





3. Abrimos el **Systems Manager** → **Session Manager** → **Start Session** → Seleccionamos nuestra instancia.

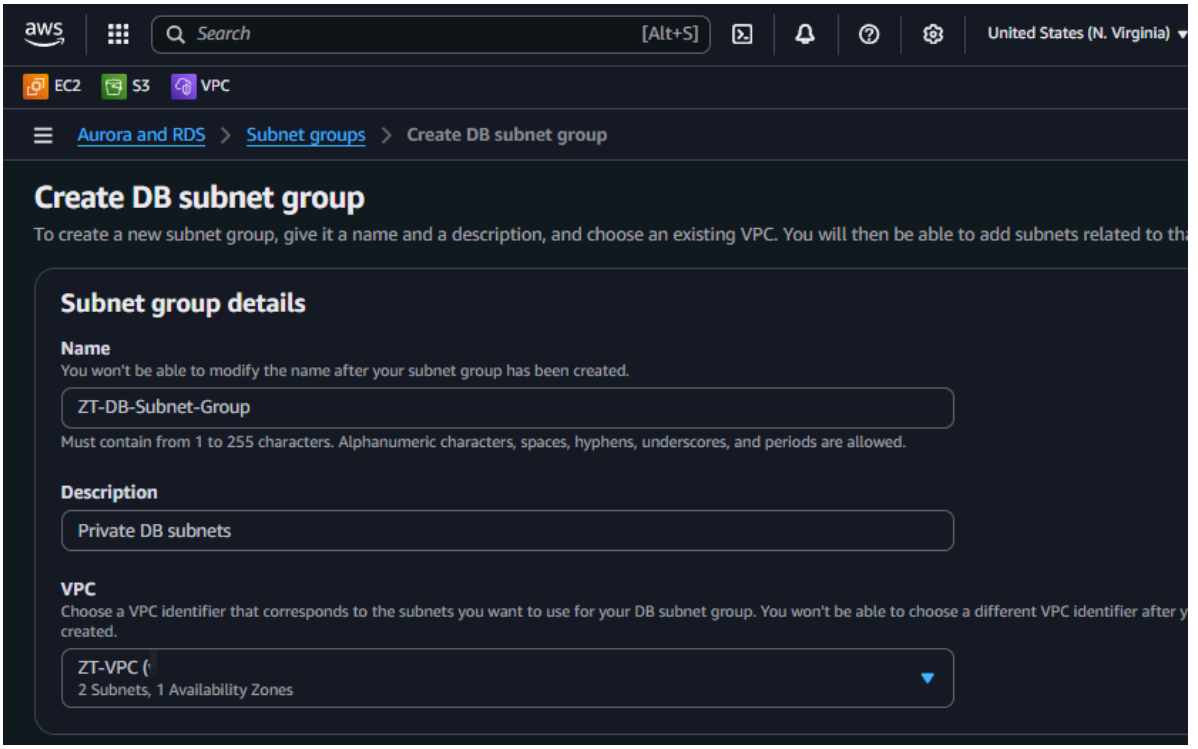


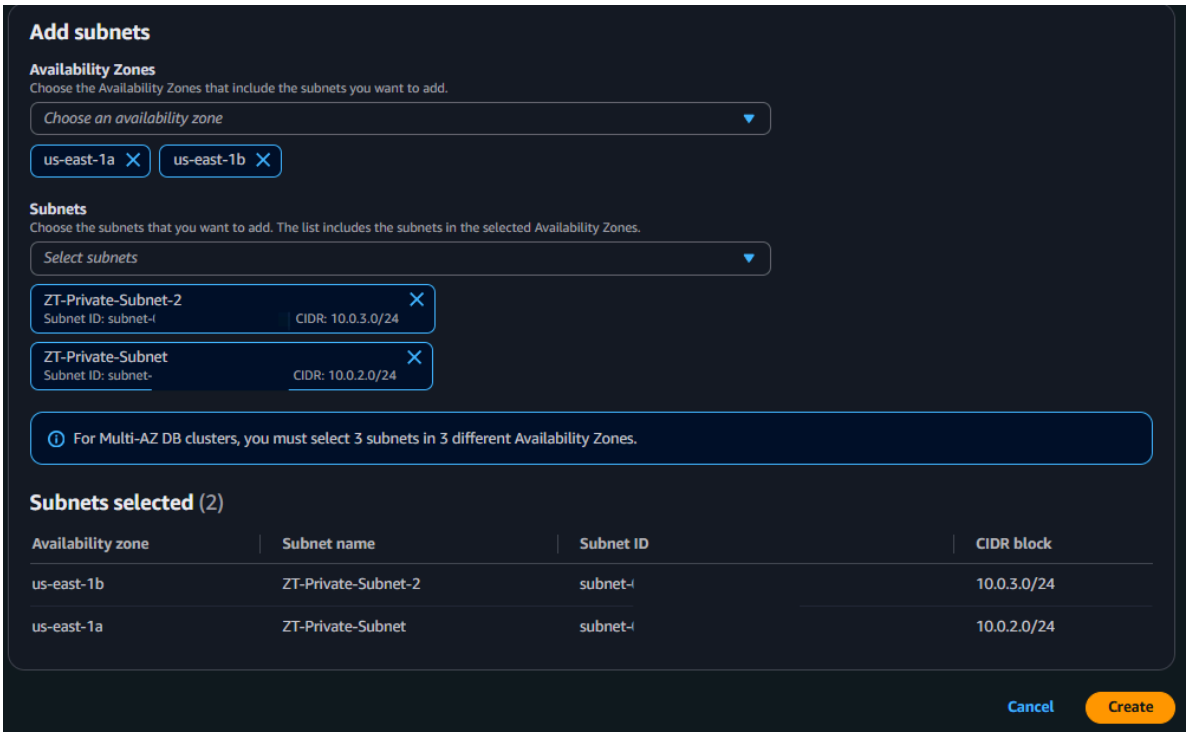
4. Hacemos click en **Start Session**. Esto nos abre otra pestaña en nuestro navegador y deberiamos poder ver la terminal.

📘 Ahora podemos acceder sin SSH, y tampoco sin IP expuesta, perfecto para Zero Trust.

- Paso 8: Crear un grupo de subredes para RDS

1. Abrimos el servicio RDS → Subnet groups → Create DB subnet group.
2. Configuramos:
   - **Name**: ZT-DB-Subnet-Group
   - Descripción: Private DB subnets
   - VPC: ZT-VPC
   - Subnets: seleccionamos ZT-Private-Subnet

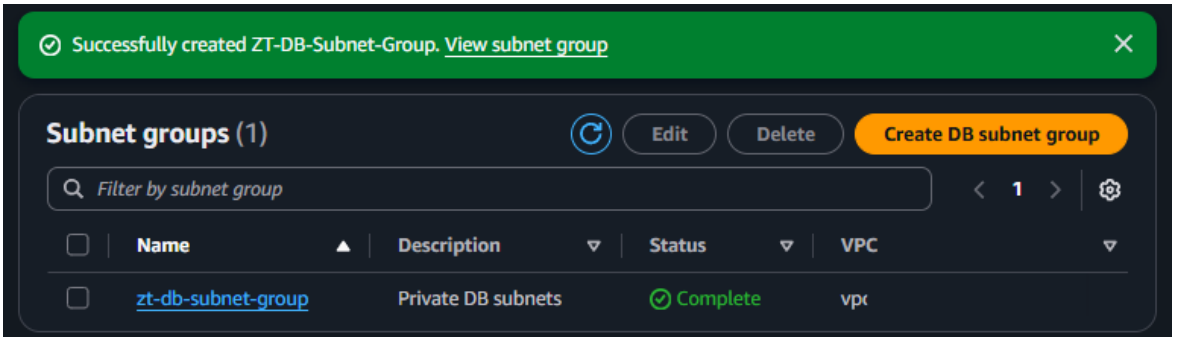




Nota: A la hora de crear esta subnet group, el mismo servicio nos pedía explicitamente tener más de una AZ. Aunque no utilicemos Multi-AZ, AWS reserva la capacidad de mover la instancia si hay una falla en la AZ, por eso siempre pide al menos 2 subredes privadas de diferentes AZs para alta disponibilidad.

Por eso, en este caso, creamos una nueva subred con CIDR 10.0.3.0/24, luego la asociamos la subred a nuestra private route table (ZT-Private-RT) y una vez creado volvemos a crear el subnet group en RDS.

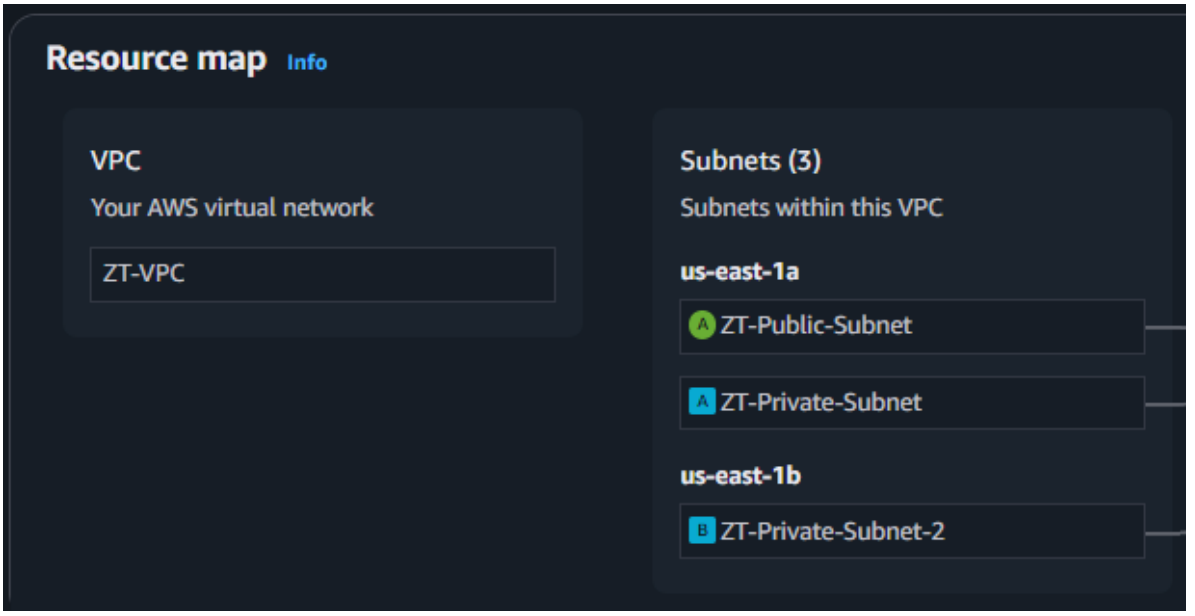
3. Clickeamos en **Create**.



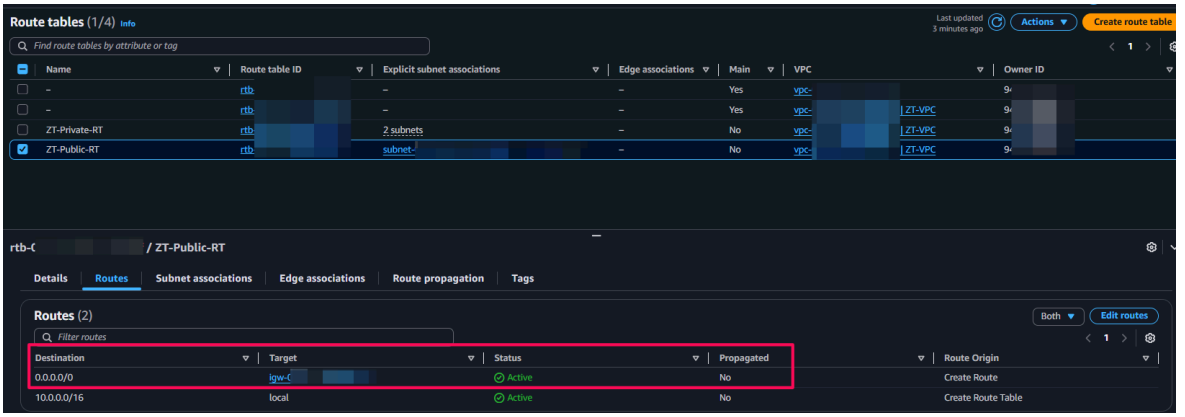
- Paso 9: Validar configuración

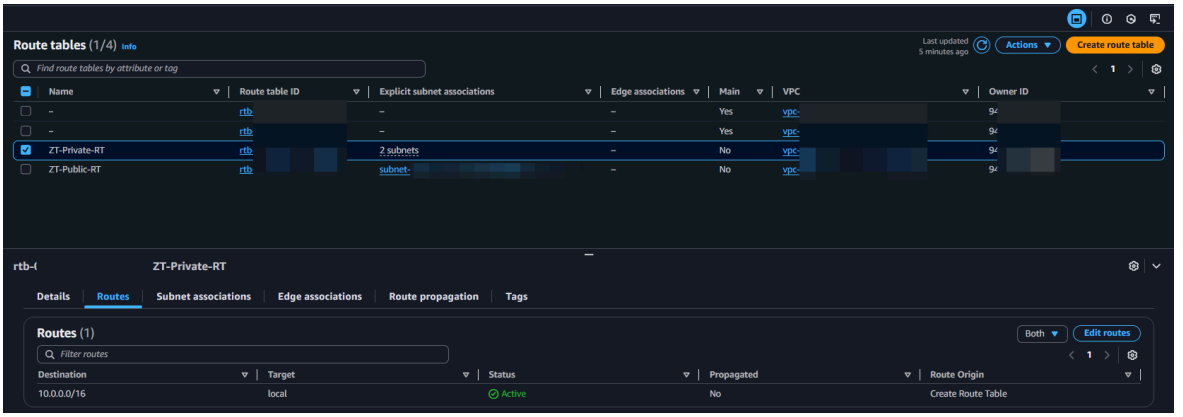
Vamos a revisar visualmente lo que hemos hecho; VPC, Route Tables, Security Groups, EC2, y SSM.

- En el panel de VPC → Your VPCs: ZT-VPC con sus subredes

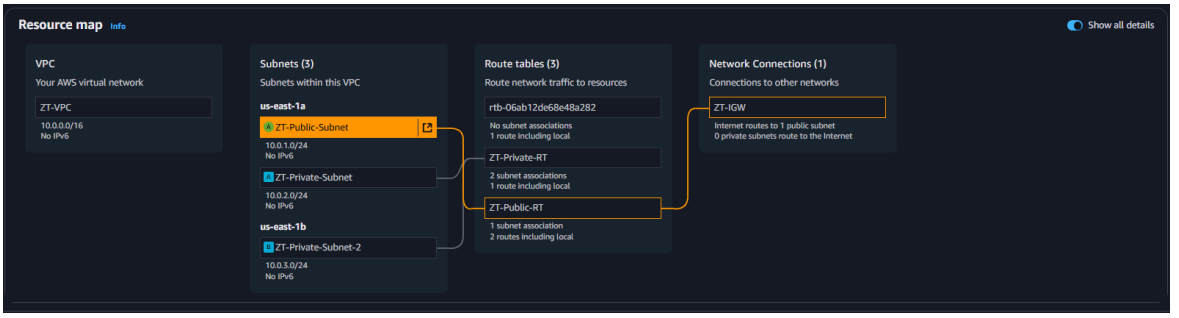


- Route Tables:
  - Route Table Pública con ruta 0.0.0.0/0 → IGW
  - Route Table Privada sin rutas externas

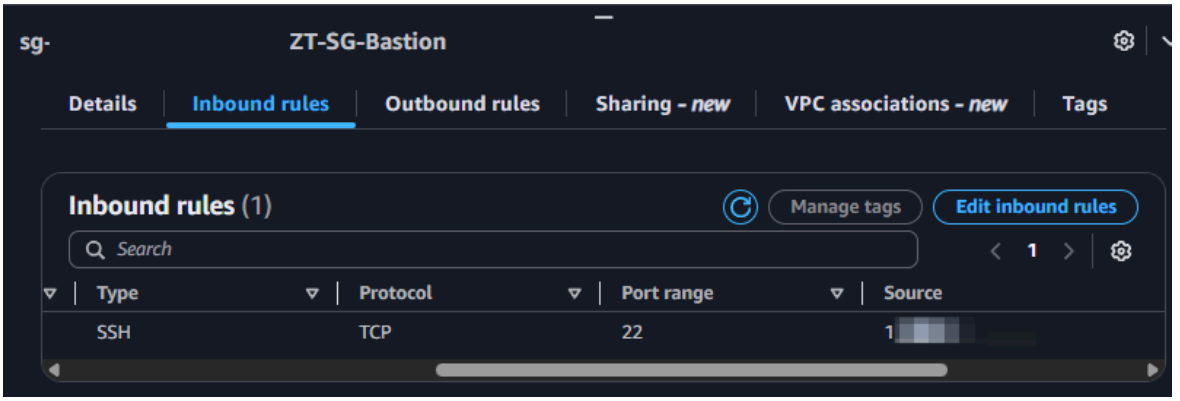


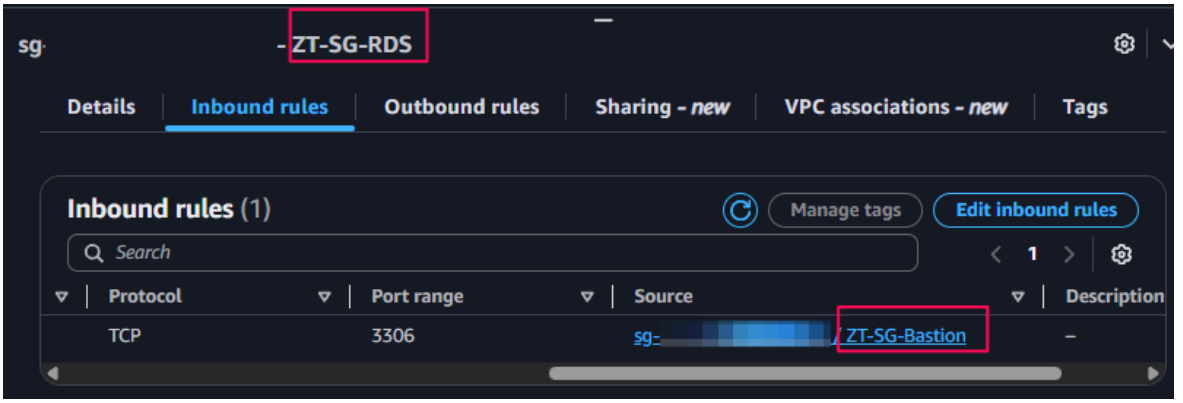


Resource map

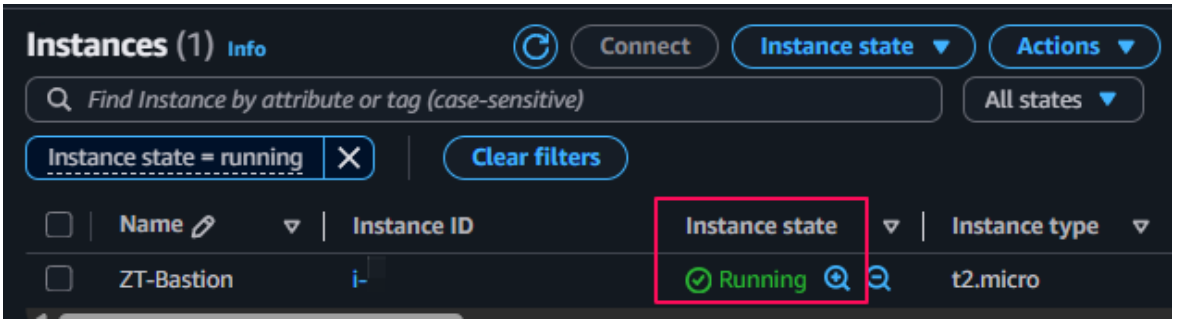


- En Security Groups:
  - ZT-SG-Bastion: Solo nuestra IP
  - ZT-SG-RDS: Solo acepta conexiones del bastion.

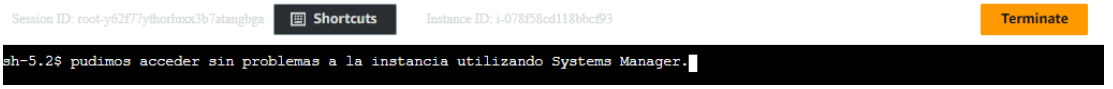




- En **EC2** → **Instances**: Verificamos que nuestra instancia esté corriendo.

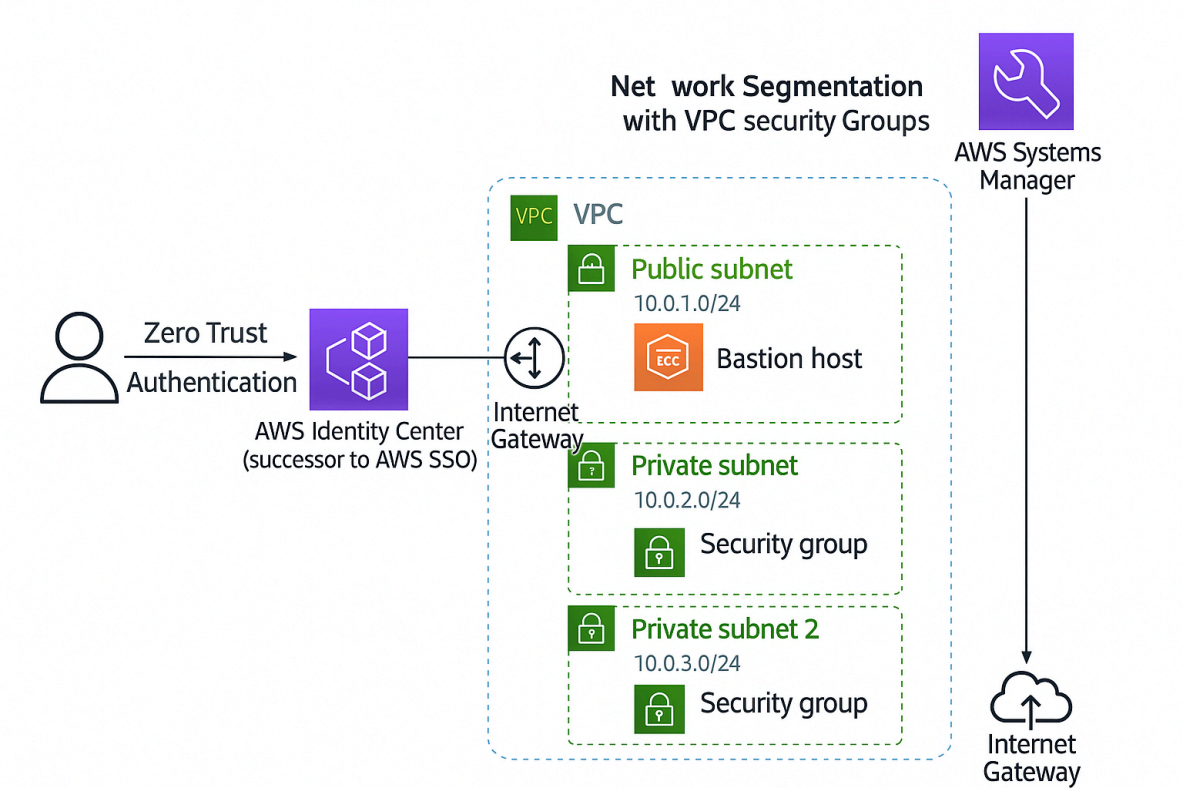


- En SSM: Verificamos que podemos abrir la sesión segura sin SSH.



Zero Trust aplicado en esta fase

Principio	Implementación
Microsegmentación	Subred pública y privadas separadas
Mínimo privilegio	SG del RDS solo permite acceso del bastion (EC2)
No implicit trust	RDS sin IP Pública, acesso remoto interno
Verificación continua	MFA + Session Manager aseguran identidad



Los elementos principales de este diagrama, basado en lo que se ha hecho hasta ahora:

- **AWS Identity Center (PAP/PDP):** Autentica usuarios antes de cualquier acceso.
- **VPC con segmentación**:
  - Subred pública con la instancia Bastion Host.
  - Dos subredes privadas para bases de datos u otros recursos internos.
- **Internet Gateway (IGW)**: Solo la subred pública tiene salida a internet.
- **AWS Systems Manager:** Es el canal seguro de gestión bajo políticas Zero Trust.
- **Security Groups**: Actúan como Policy Enforcement Points (PEP) que validan las conexiones entre capas.

FASE 5: Base de datos RDS Privada + KMS + IAM bajo Zero Trust

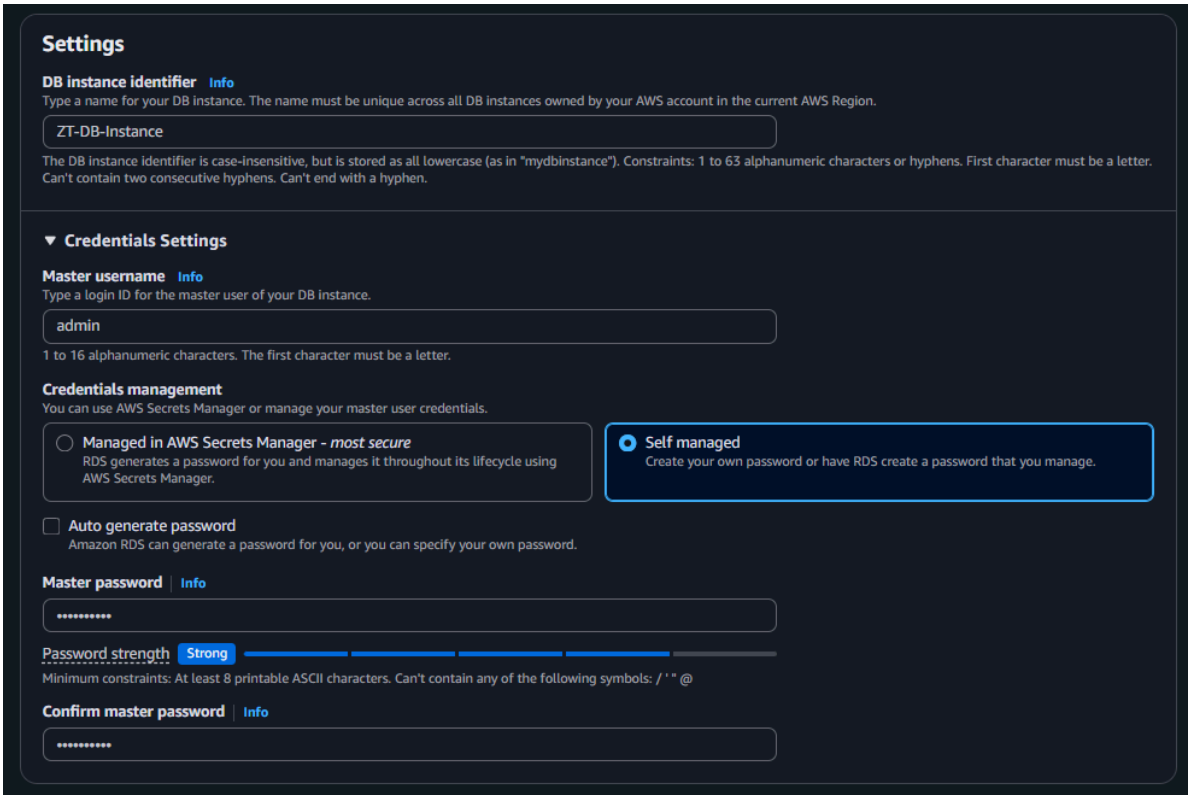
🎯 Objetivo
El objetivo en esta fase es implementar una base de datos Amazon RDS en la subred privada que creamos en la fase anterior, cifrada con AWS KMS, y accesible solo a través de la instancia bastion mediante políticas y security groups restrictivos.
Esta fase refuerza la **confidencialidad**, **integridad** y **control de acceso granular** de los datos en reposo, alineado al modelo Zero Trust.

- Paso 1: Crear instancia RDS Privada

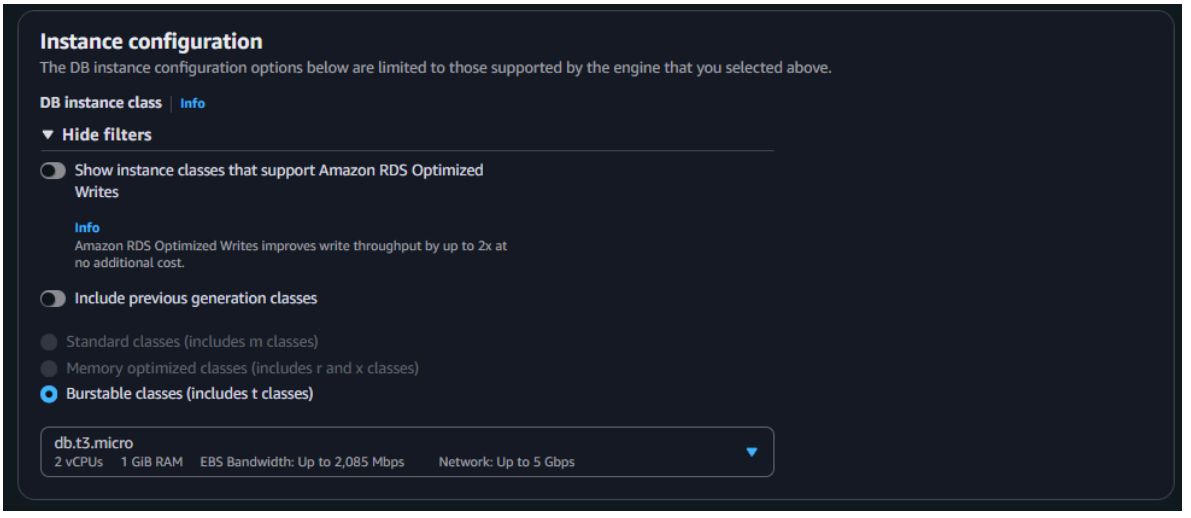
1. Vamos al panel **RDS** → **Databases** → **Create databases**
2. Seleccionamos lo siguiente:
   - **Engine type**: MySQL (o PostgreSQL)
   - **Templates**: Free Tier

3. Luego configuramos:
   - DB instance identifier: ZT-DB-Instance

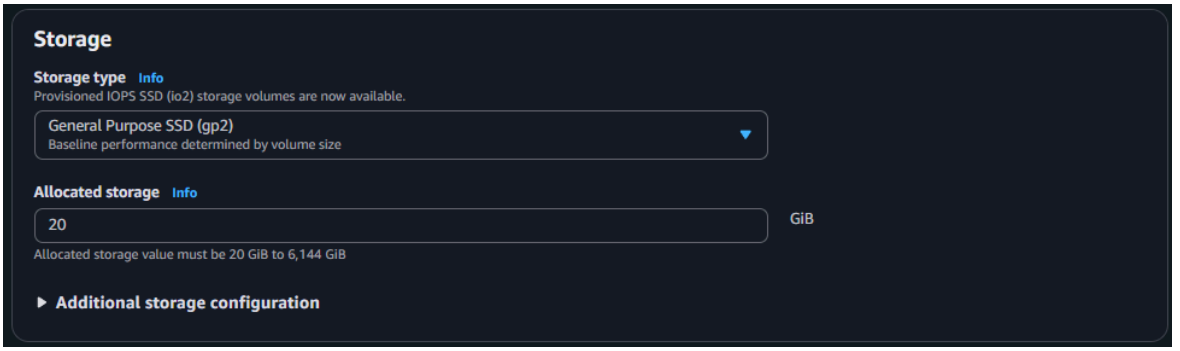
- Master username: admin
- Password: creamos una contraseña segura y la guardamos.



4. En la DB instance size, elegimos, db.t3.micro (free tier elegible)

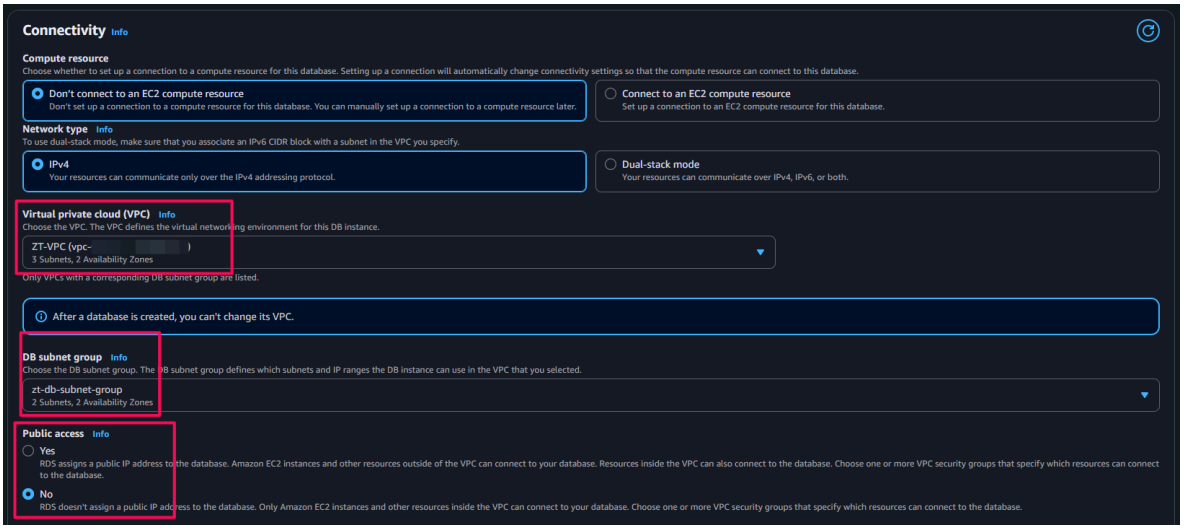


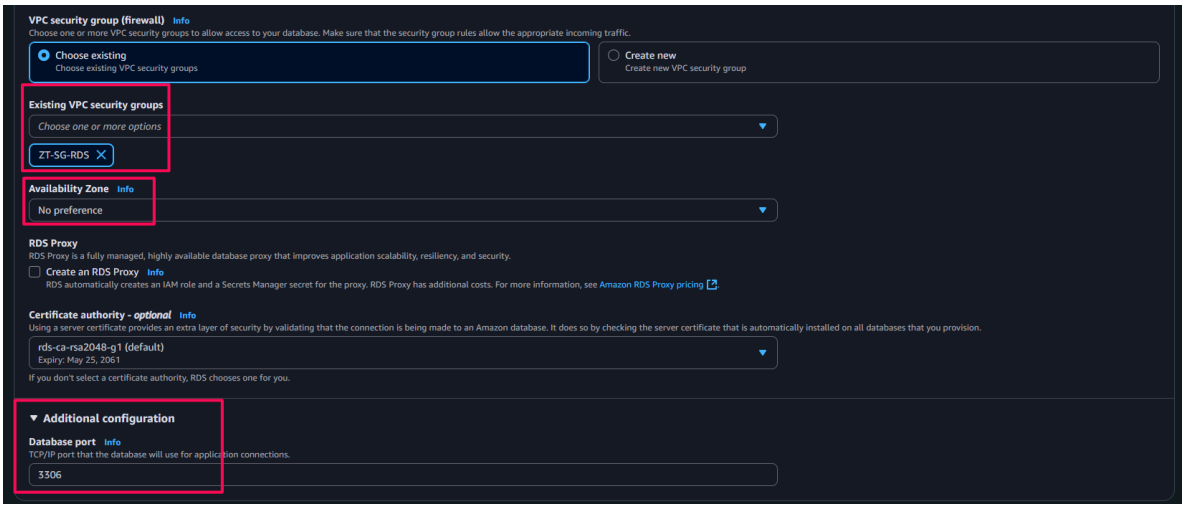
5. En Storage, seleccionamos:
   - **Type**: General Purpose (SSD)
   - **Size**: 20GB



- Paso 2: Configurar red y seguridad

1. En la sección Connectivity:
   - **VPC**: seleccionamos ZT-VPC
   - **Subnet Group**: ZT-DB-Subnet-Group (Creado en fase 3)
   - **Public access**: No
   - **VPC Security group**: seleccionamos o crear uno llamado ZT-RDS-SG
     - Reglas:
       - **Inbound**: Permitir trafico TCP al puerto 3306, que es el puerto que seleccionamos.
       - **Source**: el security group del bastion (ZT-Bastion-SG)



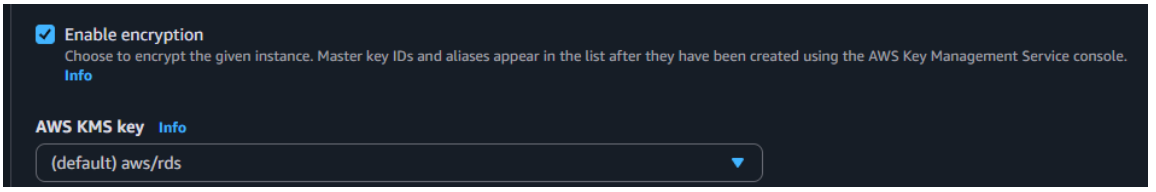


2. Availability Zone: Esta parte no importa mucho, ya que se distribuirá en las dos subredes privadas.

- Paso 3: Cifrar con KMS

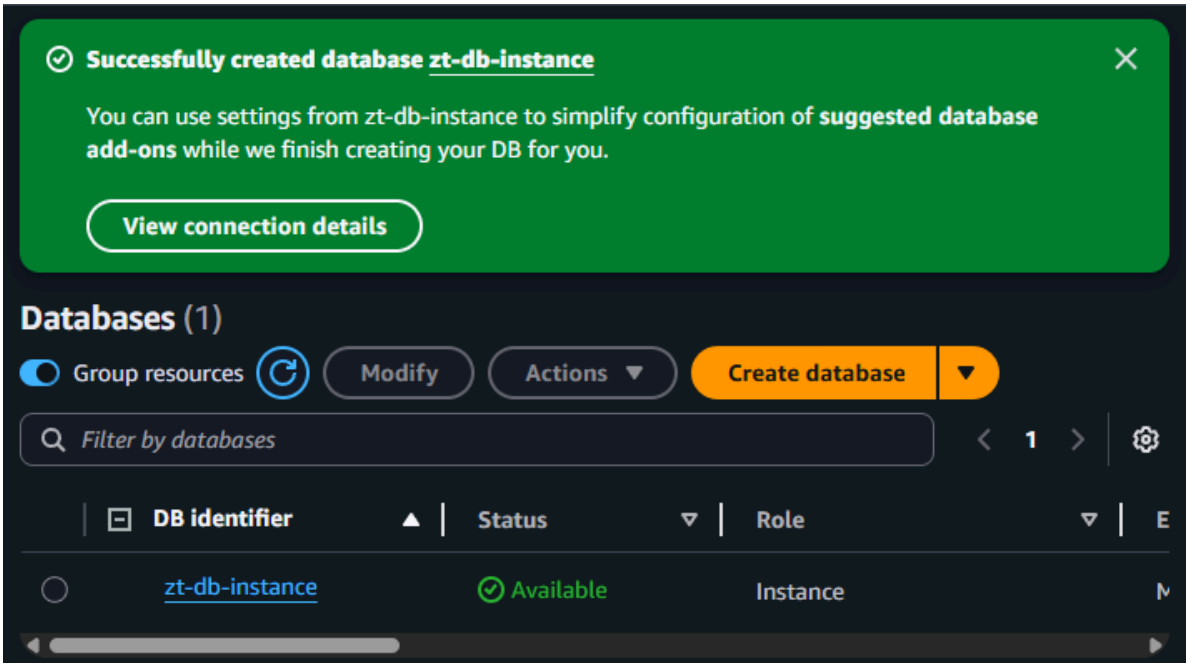
1. En la sección Encryption:
   - Enable encryption: marcamos la casilla.
   - AWS KMS Key: Elegimos aws/rds o creamos una llave nueva asignando los permisos unicamente a nuestro usuario. En nuestro caso solo usaremos la llave default aws/rds.

2. Seleccionamos la clave aws/rds.



- Paso 4: Finalizar la creación de la base de datos.

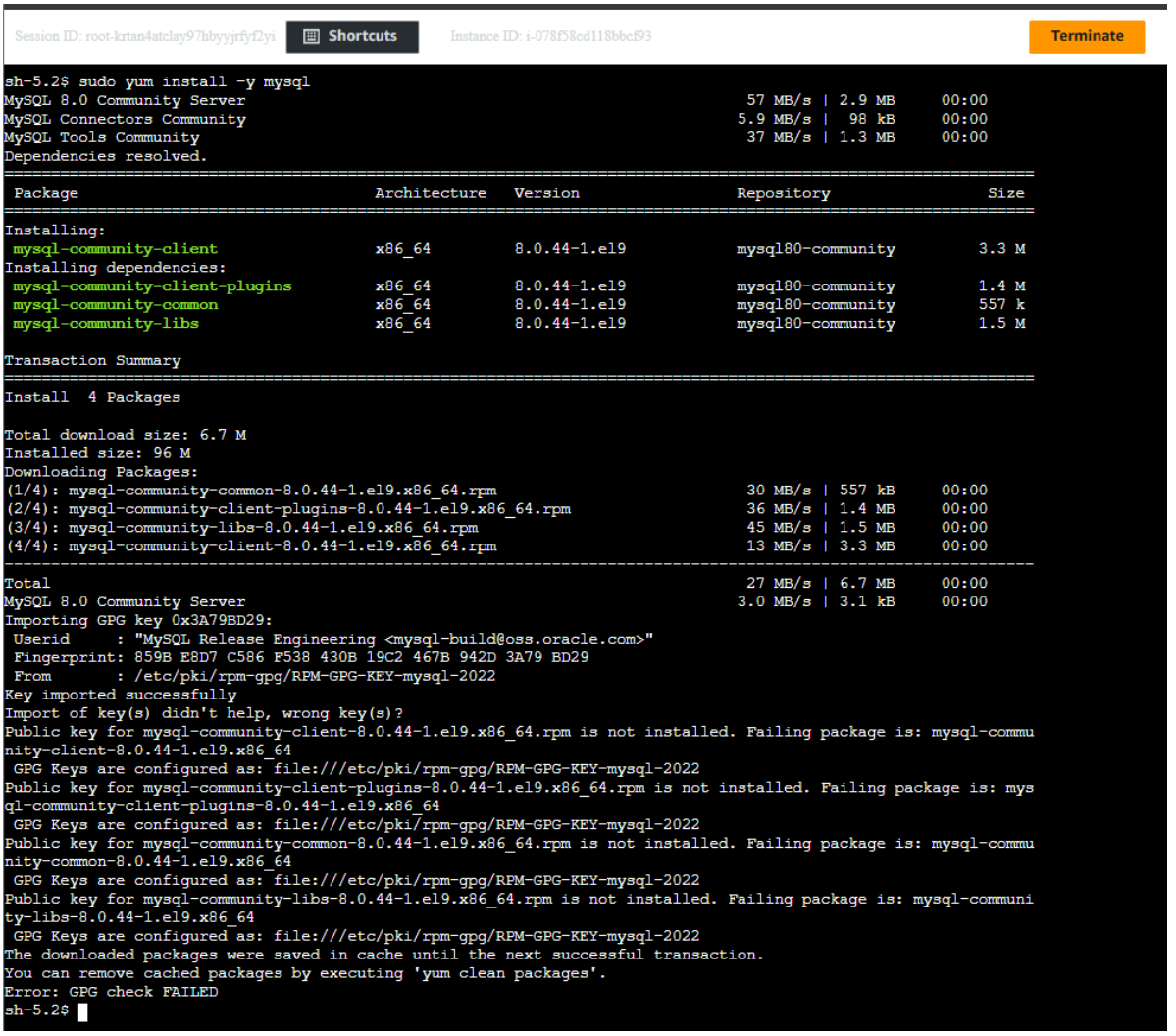
Solo hacemos click en **Create database** y esperamos a que cambio de estado a Available.



- Paso 5: Probamos la conectividad segura

1. Desde el Session Manager, en Systems Manager, abrimos sesión en la instancia Bastion.
2. y ejecutamos el siguiente comando:

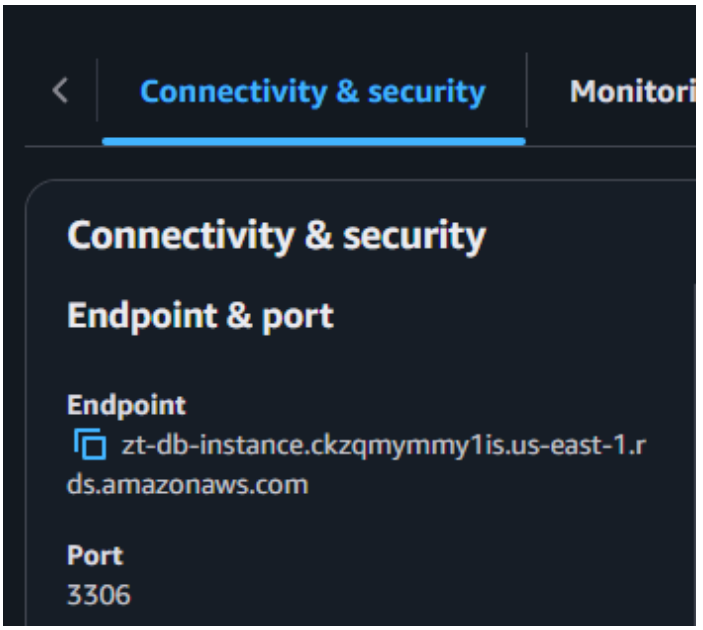
```
sudo yum install -y mysql
```



3. Luego nos conectamos ejecutando el siguiente comando:

```
mysql -h zt-db-instance.ckzqmymmy1is.us-east-1.rds.amazonaws.com -u admin -p
```

(Encuentramos el `<endpoint_RDS>` en RDS → Databases → tu instancia → Connectivity & security)

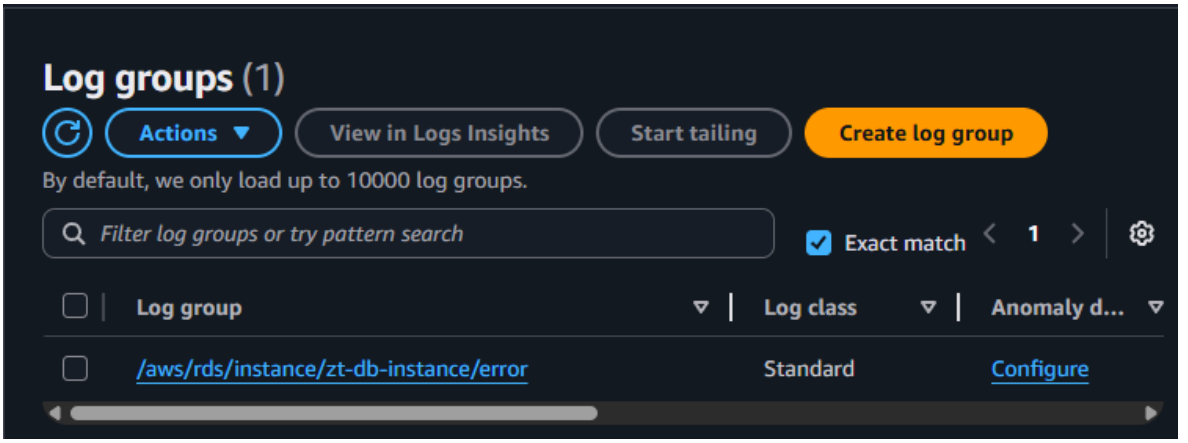



```
sh-5.2$ mysql -h zt-db-instance.ckzqmymmy1is.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 30
Server version: 8.0.42 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

Una vez conectados podemos verificar de que tenemos acceso seguro, cifrado y solo desde el bastión (EC2).

- Paso 6: Activar Auditoría y trazabilidad

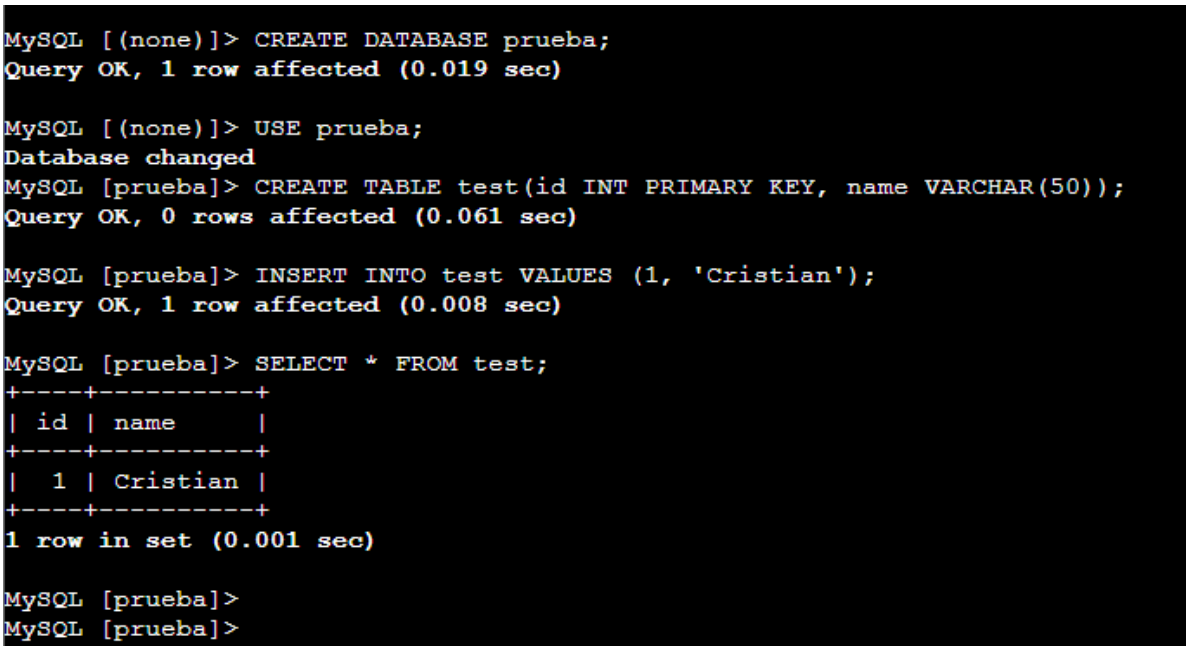
1. Vamos a **CloudWatch** → **Logs** → **Log groups** → /aws/rds/instance/...



Nota: En esta primera captura vemos un solo Log group ya que ese /aws/rds/instance/zt-db-instance/error se activa con cualquier error interno del motor, para generar automáticamente otros Log Groups es ejecutando queries reales desde el bastion, como por ejemplo cuando empecemos a crear tablas o consultas).

En este caso, realizamos unas queries, nos conectamos a la instancia desde Systems Manager y entramos a la base de datos, una vez logeados, ejecutamos el siguiente query:

```
CREATE DATABASE prueba;
USE prueba;
CREATE TABLE test(id INT PRIMARY KEY, name VARCHAR(50));
INSERT INTO test VALUES (1, 'Cristian');
SELECT * FROM test;
```



Una vez realizado el query, esperamos unos 2-3 minutos y luego revisamos CloudWatch → Log Groups y deberíamos poder ver los grupos nuevos que se generen.

- y así obtenemos trazabilidad centralizada de eventos.

¿Qué hemos logrado hasta ahora?

Componente	Propósito
RDS en subred privada	Sin Acceso público, solo interno
KMS encryption	Cifrado en reposo gestionado por clave simétrica
IAM y SG	Control de acceso granular entre Bastion y DB
Session Manager	Acceso sin SSH, con trazabilidad
CloudWatch	Monitoreo y auditoría bajo Zero Trust

- **FASE 6: Monitoreo Continuo (CloudWatch + CloudTrail + SNS)**

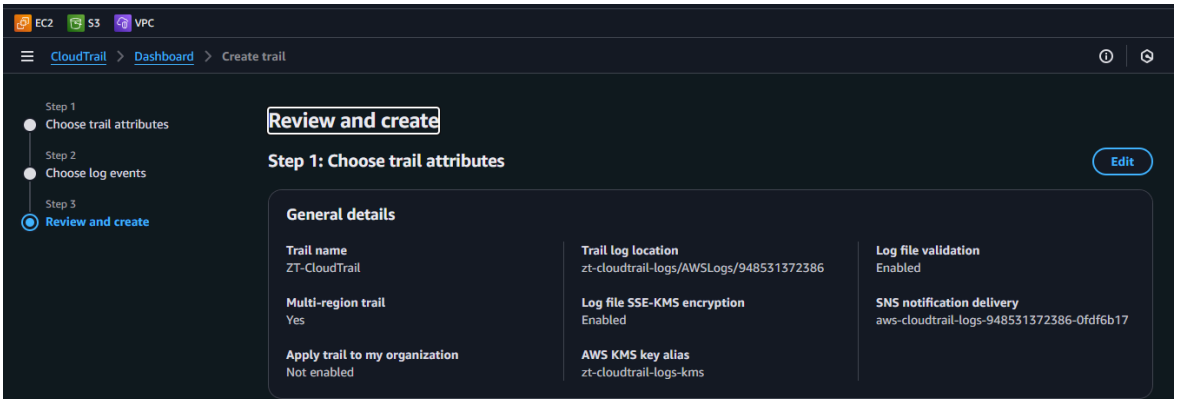
Objetivo
El objetivo en esta fase es el habilitar la visibilidad total de eventos críticos (inicios de sesión, cambios IAM, accesos fallidos) y recibir alertas automáticas usando SNS.

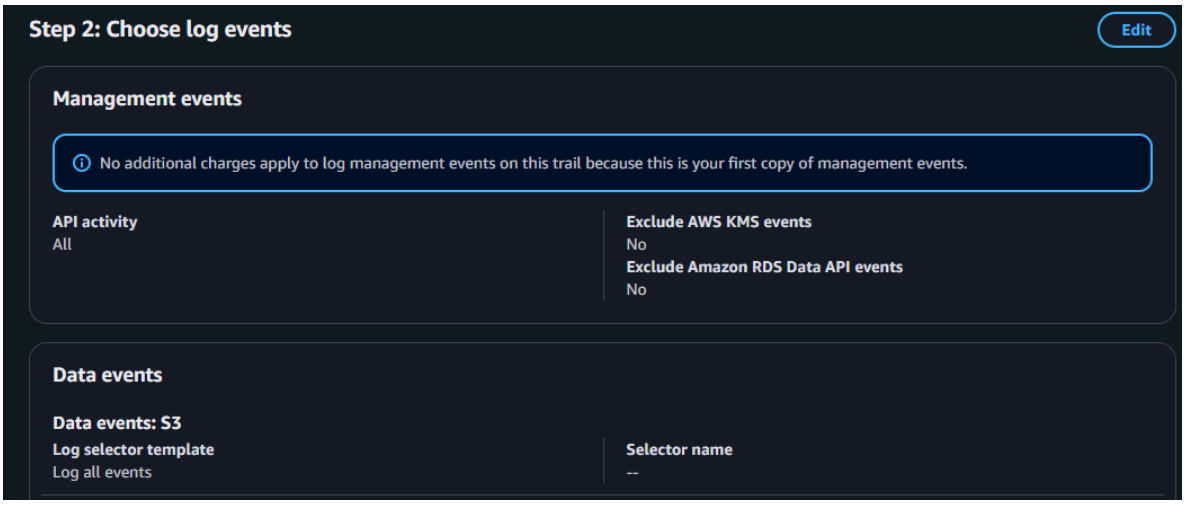
- Paso 1: Habilitar CloudTrail

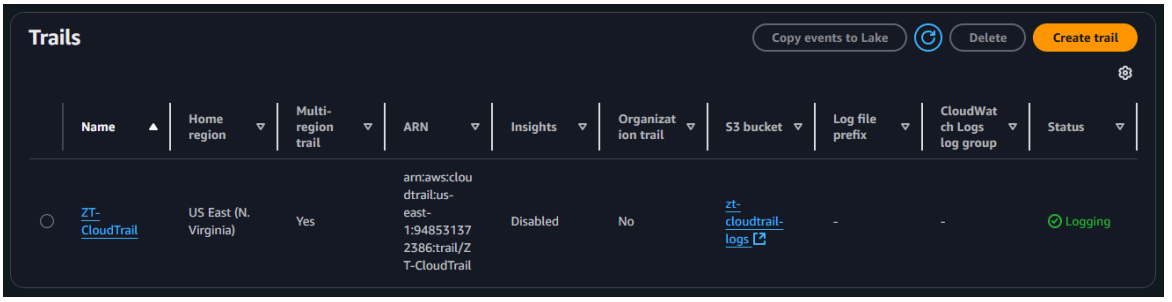
1. Vamos a CloudTrail -> Trails -> Create Trail

2. Configuramos:
   - **Trail name**: ZT-CloudTrail
   - **Storage location**: creamos un nuevo bucket S3 -> zt-cloudtrail-logs
   - **Apply CloudTrail to all regions**: marcamos el check point para activarlo.
   - **Log File SSE encryption**: Aqui podemos usar la clave aws/s3 o nuestra llave KMS por si queremos más control.
   - **Log Events**:
     - **Management events** -> Read/Write
     - **Data events** -> S3
     - **Insights events** -> Esta opcion es más opcional, por si queremos un análisis más anómalo.

3. Clickeamos en **Create CloudTrail**.

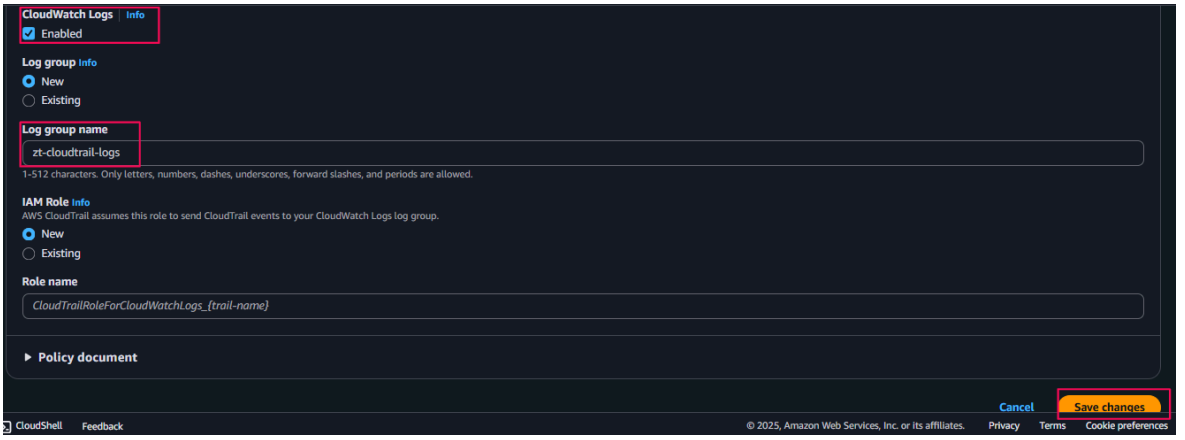






- ## Paso 2: Enviar logs a CloudWatch

1. Luego de crear el Trail, podemos clickear en nuestro Trail recién creado -> CloudWatch logs -> Edit -> Activamos CloudWatch Logs
2. Configuramos:
   - **Log Group Name**: zt-cloudtrail-logs
   - **IAM Role**: CloudTrail se encarga de crear un nuevo rol automaticamente llamado CloudTrail_CloudWatchLogs_Role
   - Clickeamos en **Save Changes**



- **Paso 3: Crear filtro de métrica en CloudWatch**

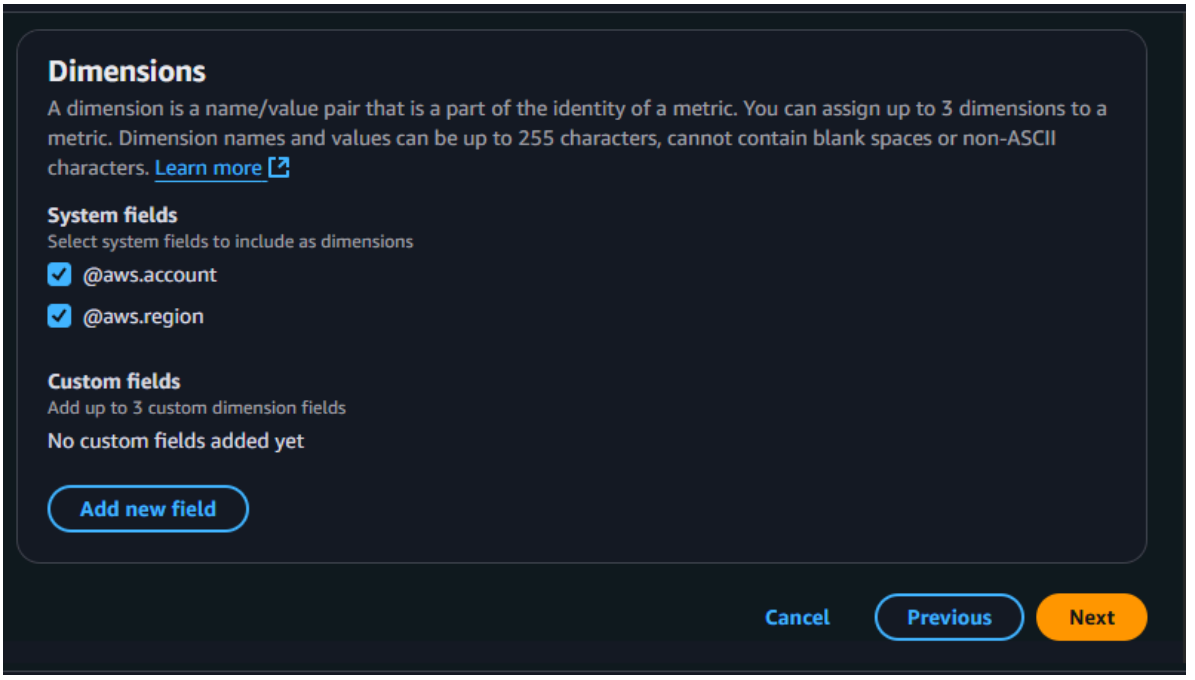
1. Vamos a **Cloudwatch** -> **Log groups** -> **/aws/cloudtrail/..**
2. En la pestaña Metric Filters, hacemos click en:
   - **Create Metric Filter**

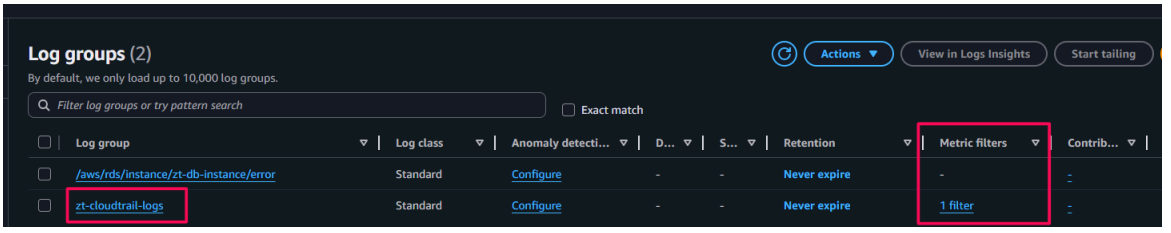
3. Pegamos este patrón:

```
{ ($.eventName = "ConsoleLogin") && ($.errorMessage = "Failed
authentication") }
```

4. Hacemos click en Next y lo nombramos **FailedConsoleLogins**
5. Y creamos una métrica bajo el namespace: **SecurityMetrics**
   - **Metric name:**
   - **Metric Value: 1**

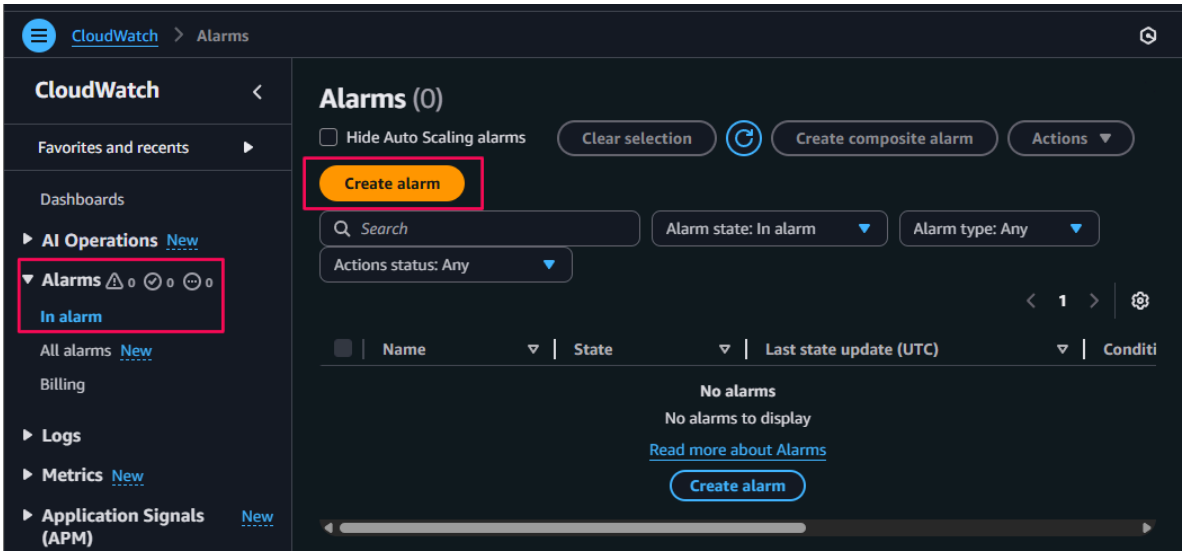
6. En Dimensions marcamos ambas casillas como se muestra a continuación y clickeamos next.



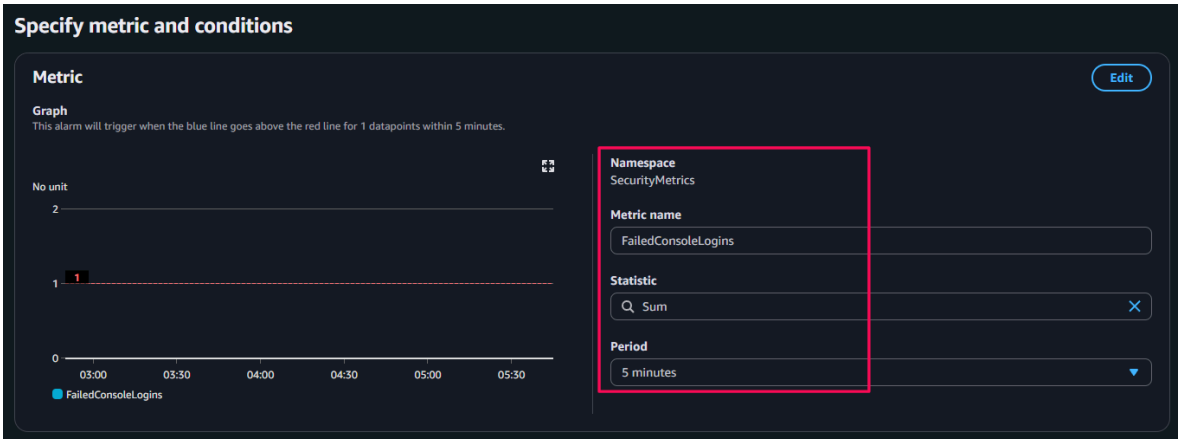


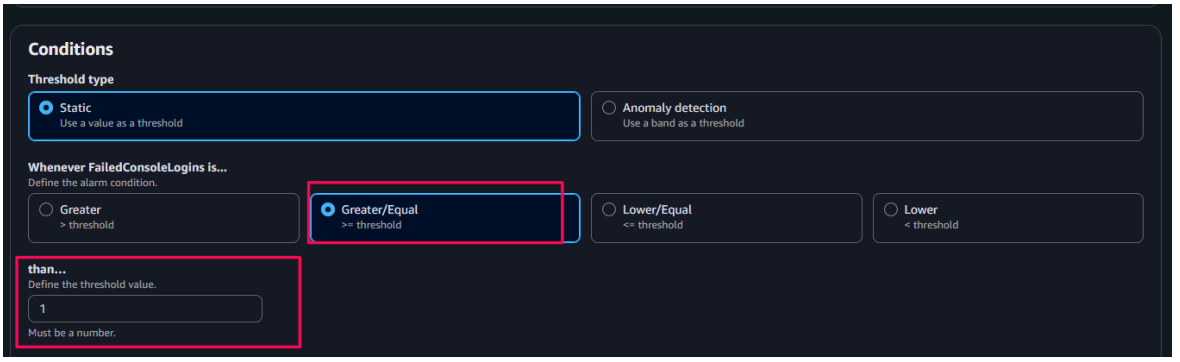
- **Paso 4: Crear Alarma y SNS**

1. En **CloudWatch**, ubicamos el panel izquierdo → **Alarms** → **Create Alarm**.

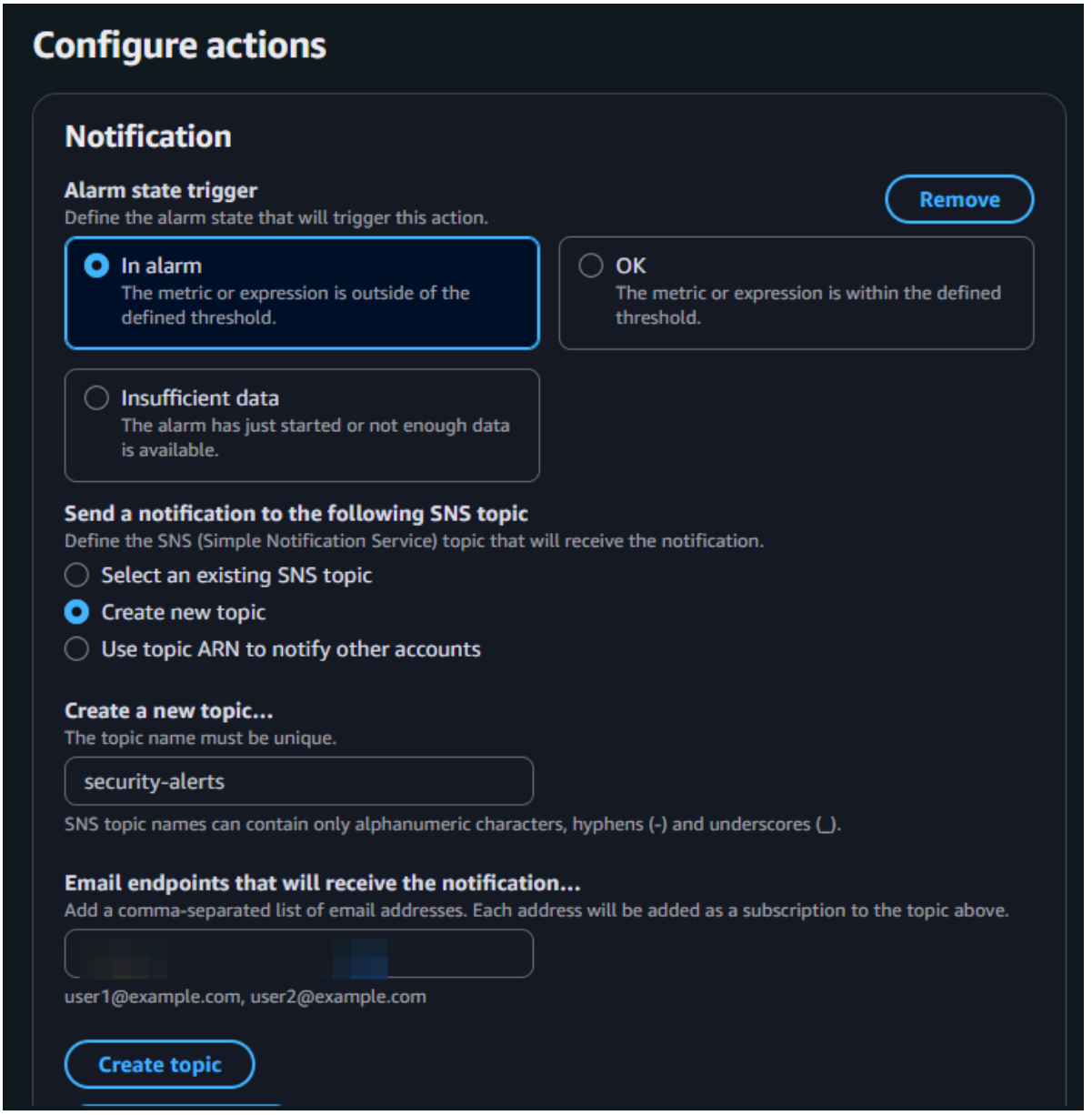


2. Seleccionamos la métrica FailedConsoleLogins
3. Condición:
    - Threshold: >= 1
    - Period: 5 Minutes

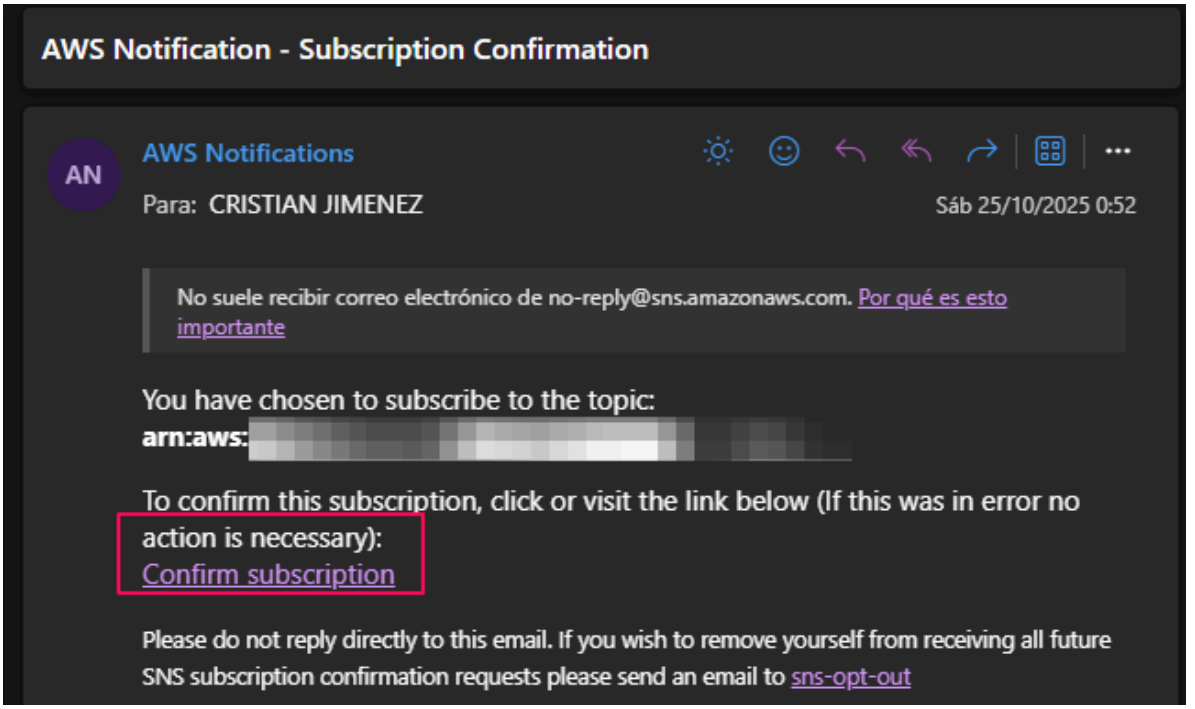


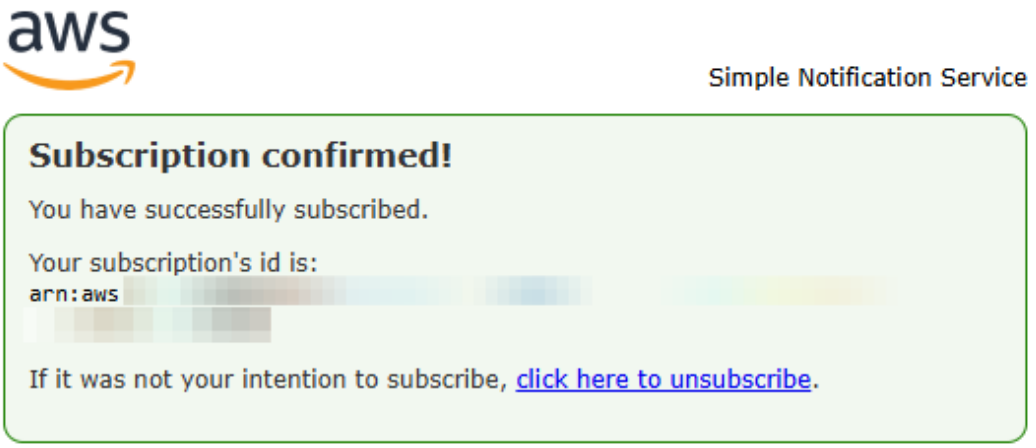


4. En notificación, creamos un nuevo SNS topic
    - Name: security-alerts
    - Email endpoint: nuestro correo

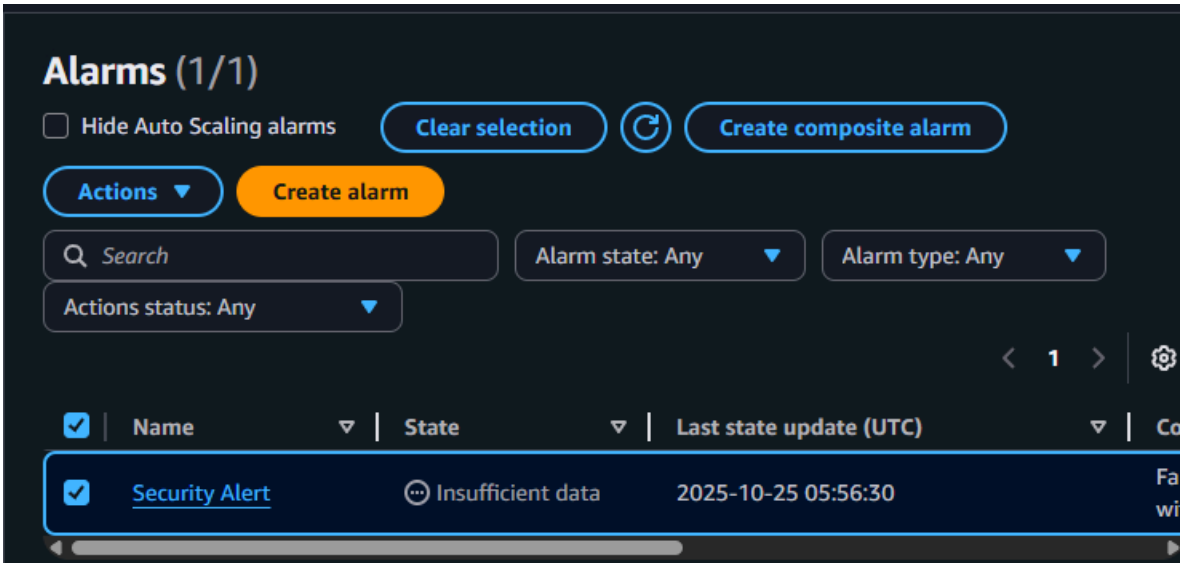


5. Confirmamos nuestro correo desde nuestra bandeja de entrada.



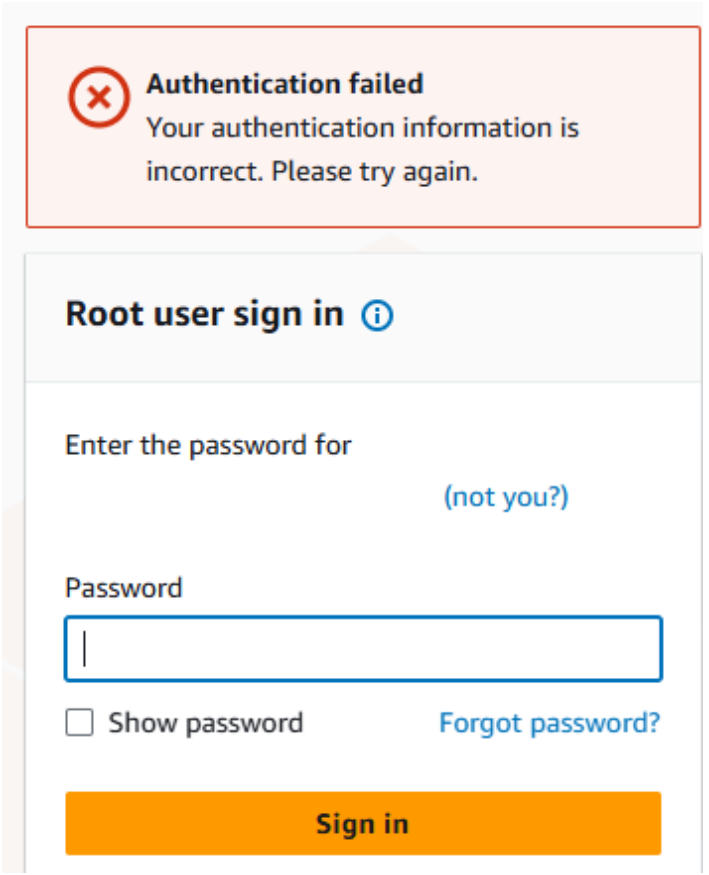


6. Click en Create alarm.



- Paso 5: Simulación

1. Intentamos iniciar sesión a AWS con una contraseña errónea.

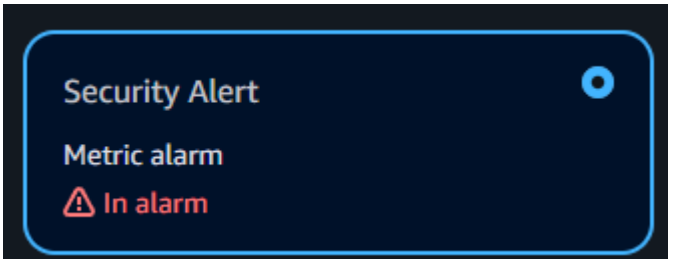


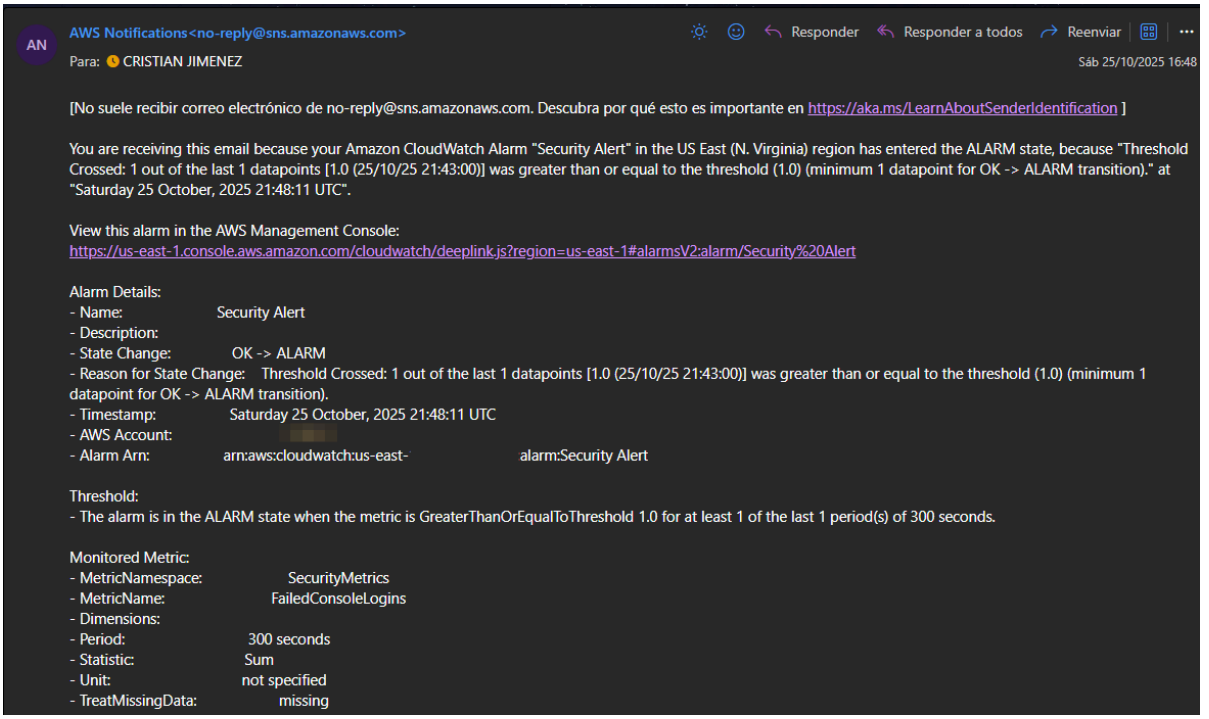
2. Esperamos de 2 a 5 minutos.

Alerta generada en Cloudwatch:

3. Verificamos que nos llega el correo de alerta.

Alerta en estado "In Alarm"





Efectivamente verificamos que el correo nos llegó unos minutos después del evento.

- **Conclusión**

La implementación realizada permitió construir un entorno AWS seguro y trazable alineado a los principios del modelo Zero Trust. Durante las fases se configuraron los servicios fundamentales (VPC, EC2, RDS, IAM, CloudTrail, CloudWatch y SNS) garantizando control de acceso, visibilidad de eventos y respuesta ante incidentes. La arquitectura resultante integra administración de políticas (IAM y KMS), evaluación y monitoreo continuo (CloudWatch y CloudTrail) y mecanismos de alerta (SNS), cumpliendo los lineamientos del NIST SP 800-207. Este entorno sirve como base para futuras pruebas de detección y respuesta, así como para prácticas de automatización de políticas adaptativas y análisis de seguridad en la nube.