

**GUÍA DE LABORATORIO 15**

**“CloudFormation”**

**Estudiantes**

**Muñoz Herrera Stevan Pacheco**

***LABORATORIO***

## OBJETIVOS

* Comprender el funcionamiento de CloudFormation en AWS

## REQUERIMIENTOS

* + Leer las diapositivas del tema.

**PROCEDIMIENTO**

**(El siguientes laboratorio fue diseñado para ser desarrollado en grupos**

**de 2 personas)**

**Crear plantilla personalizada Stack-EC2**

{

"AWSTemplateFormatVersion" : "2010-09-09", "Resources": {

"Instancia3": {

"Type": "AWS::EC2::Instance", "Properties" : {

"ImageId" : "ami-0885b1f6bd170450c", "KeyName" : "key",

"InstanceType" : "t2.micro",

"SecurityGroupIds" : ["sg-0b5b241fca52ea5ed"]

}

},

"ElasticIP3" : {

"Type" : "AWS::EC2::EIP",

"Properties" : {

"InstanceId" : { "Ref" : "Instancia3" }

}

}

}

}

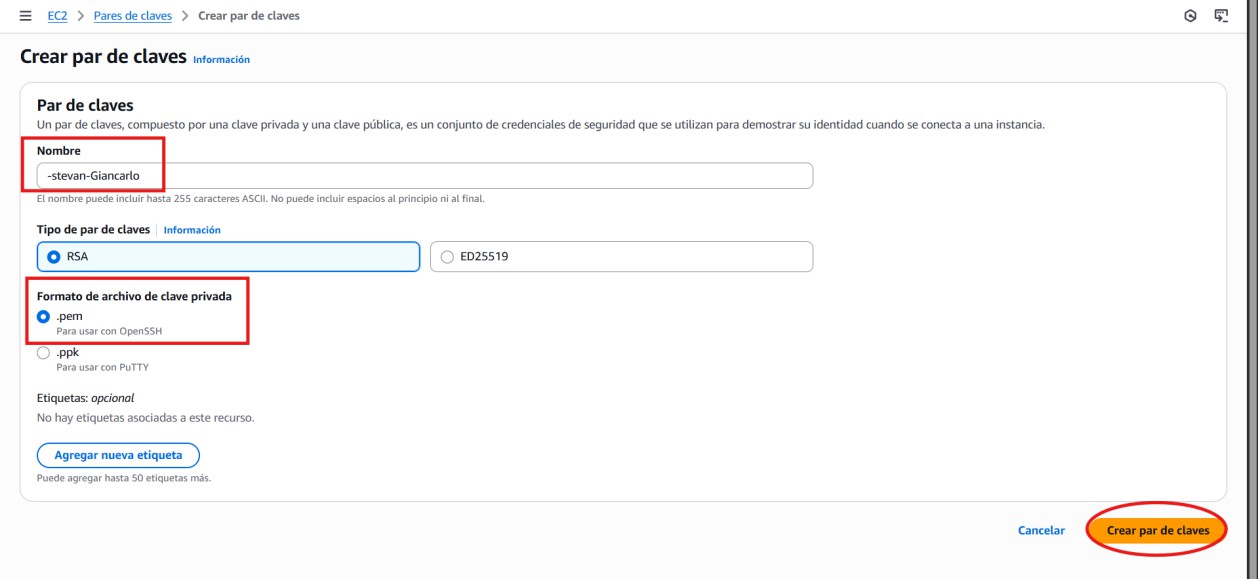
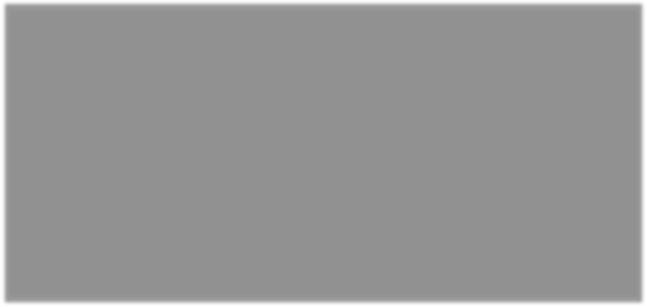
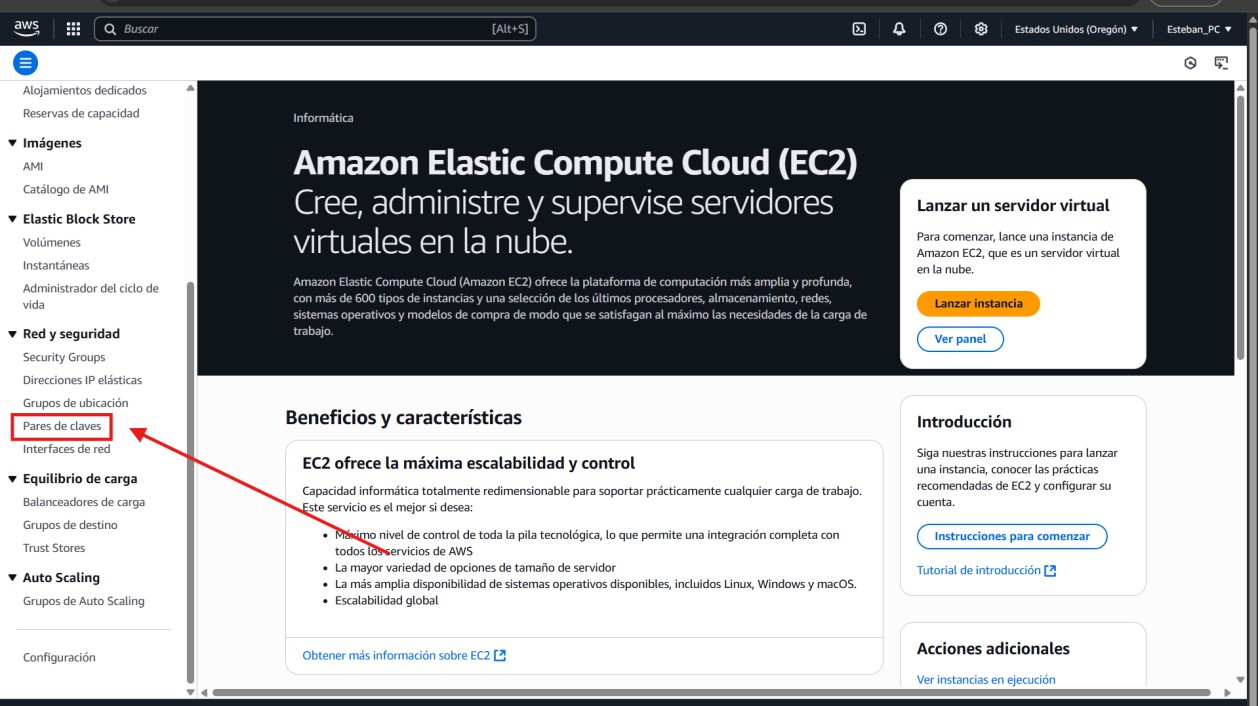
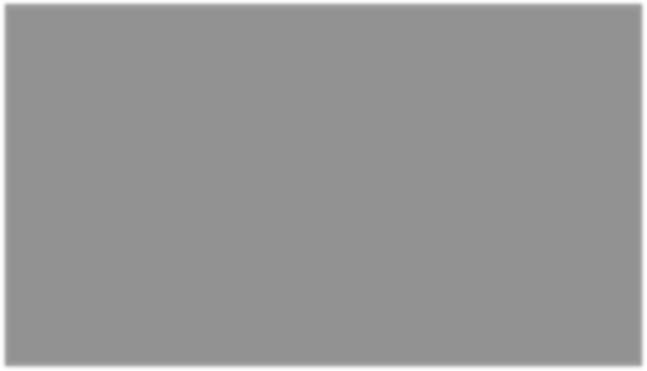
1. Ingresar al siguiente enlace y posteriormente realizar las actividades debajo: [http://docs.aws.amazon.com/es\_es/AWSCloudFormation/latest/UserGuide/sample-](http://docs.aws.amazon.com/es_es/AWSCloudFormation/latest/UserGuide/sample-templates-services-us-west-2.html) [templates-services-us-west-2.html](http://docs.aws.amazon.com/es_es/AWSCloudFormation/latest/UserGuide/sample-templates-services-us-west-2.html)
   1. Probar 6 plantillas de las brindadas por AWS, no más de una por tema, con los resultados obtenidos responda lo siguiente:
      1. Describa la estructura que pudo observar en el JSON, ¿qué le ha lla- mado la atención de la estructura?, ¿qué representa en ejecución la estructura planteada (recursos en uso, conexiones, etc)?
      2. Vea en el designer la plantilla y describa los aspectos importantes de la estructura planteada en modo designer. Agregue imágenes de ser ne- cesario.
      3. Haga el lanzamiento de la plantilla y describa para qué ha sido pensada,

¿qué requerimiento solucionaría de un equipo de desarrollo?, ¿cuándo se debería utilizar esta plantilla y cuándo no?

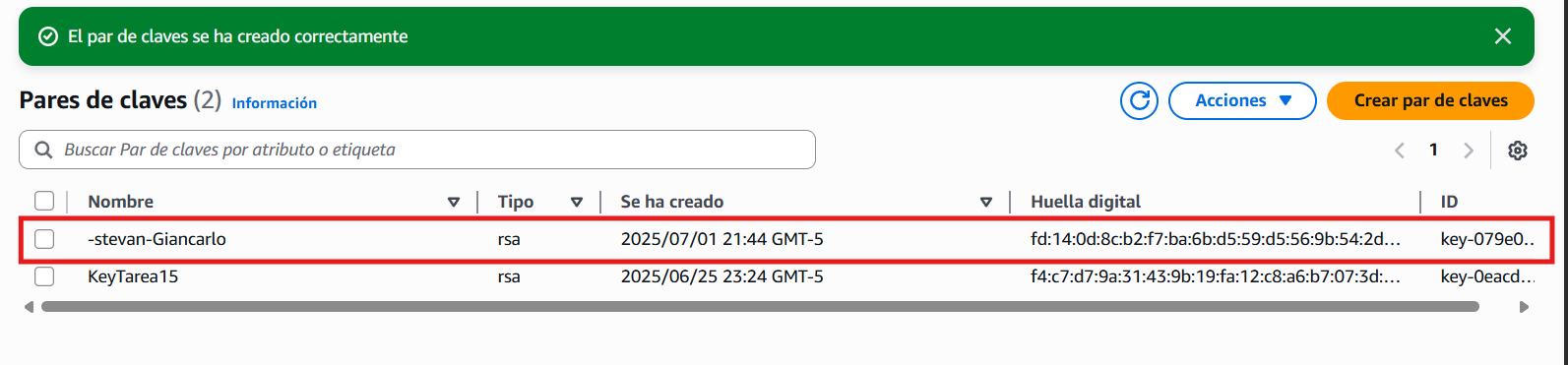
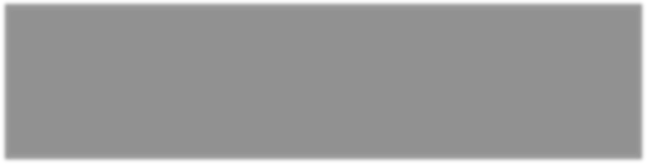
1. Describir las ventajas y desventajas que ha podido visualizar en el uso de CloudForma- tion.

# Desarrollo.

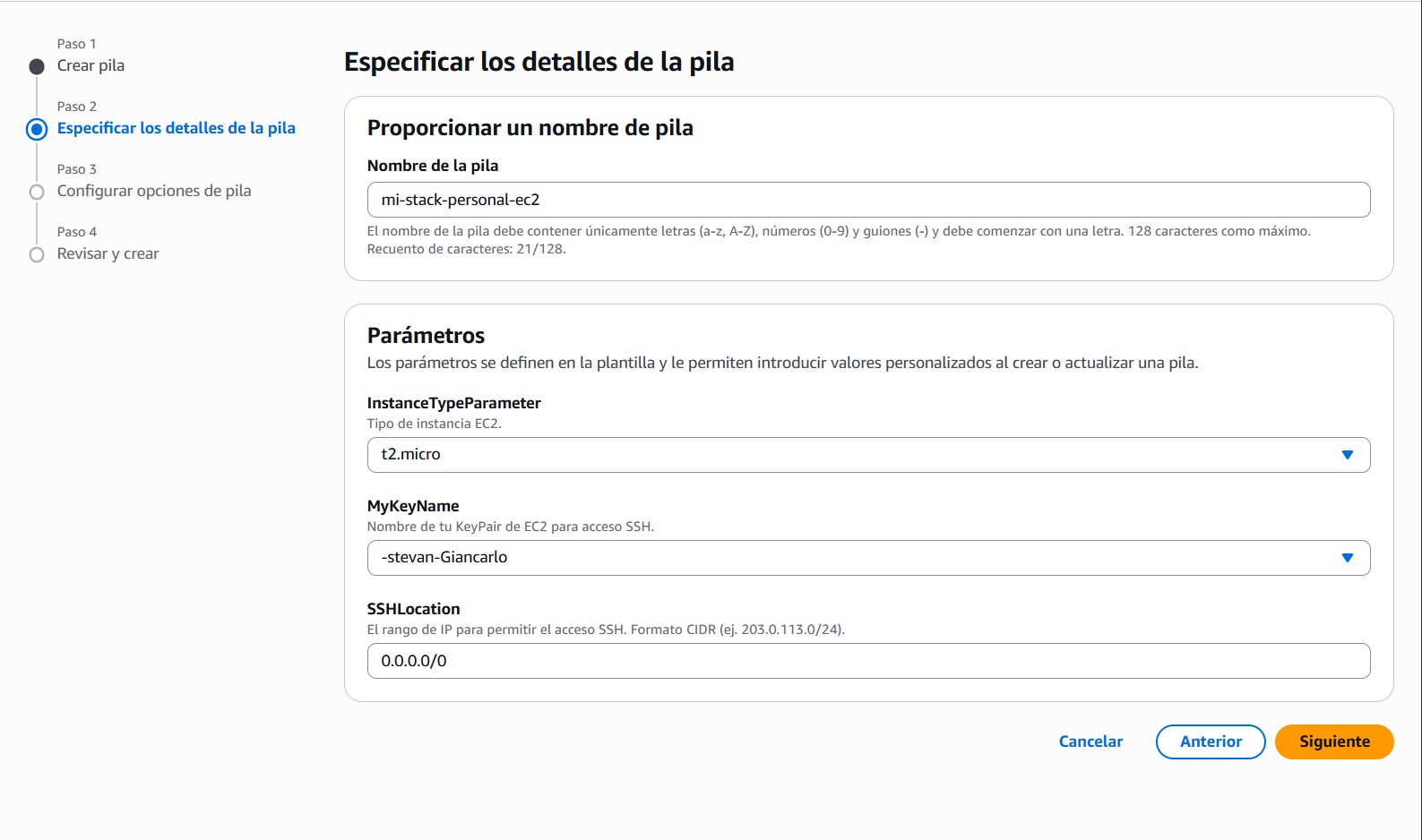
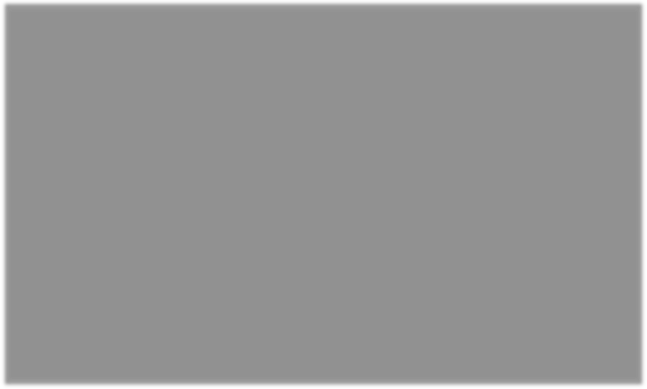
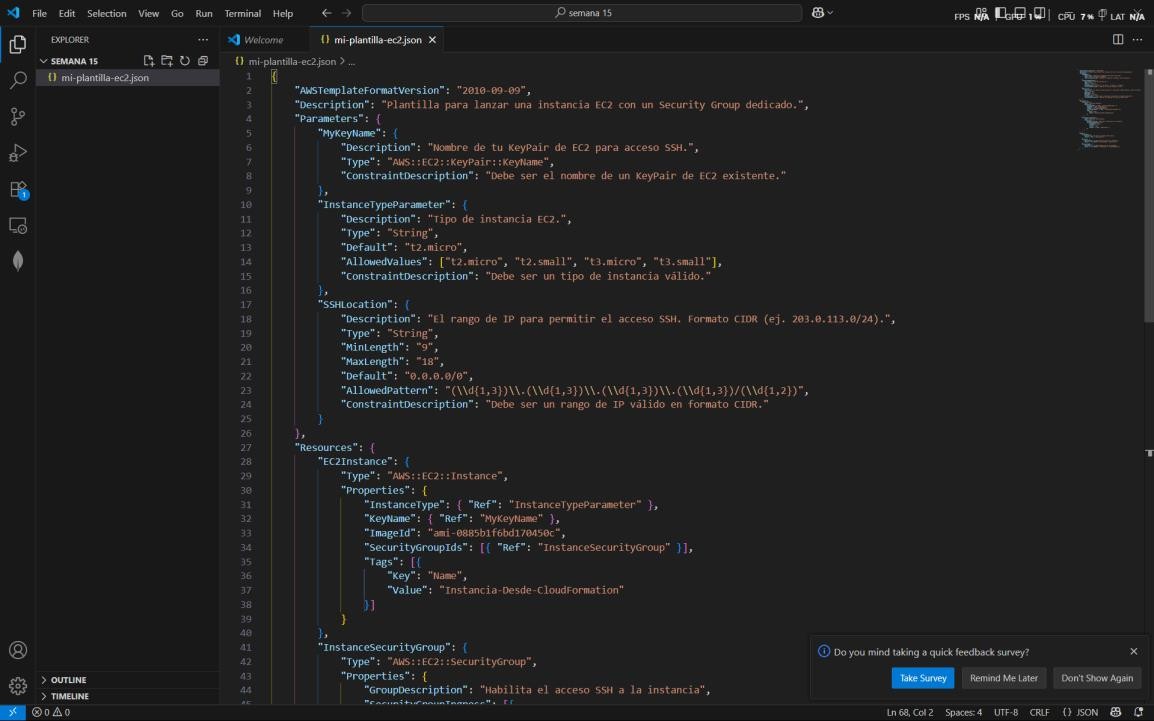
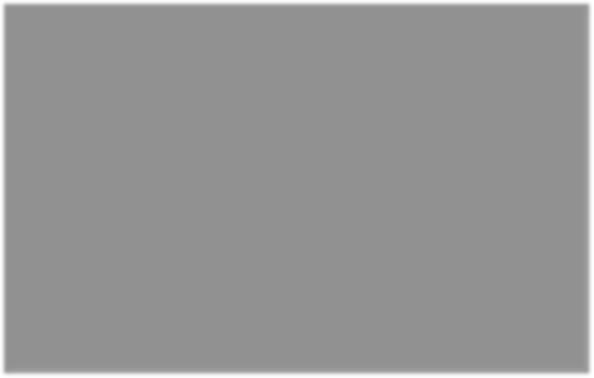
Primero necesitamos crear el par de claves.



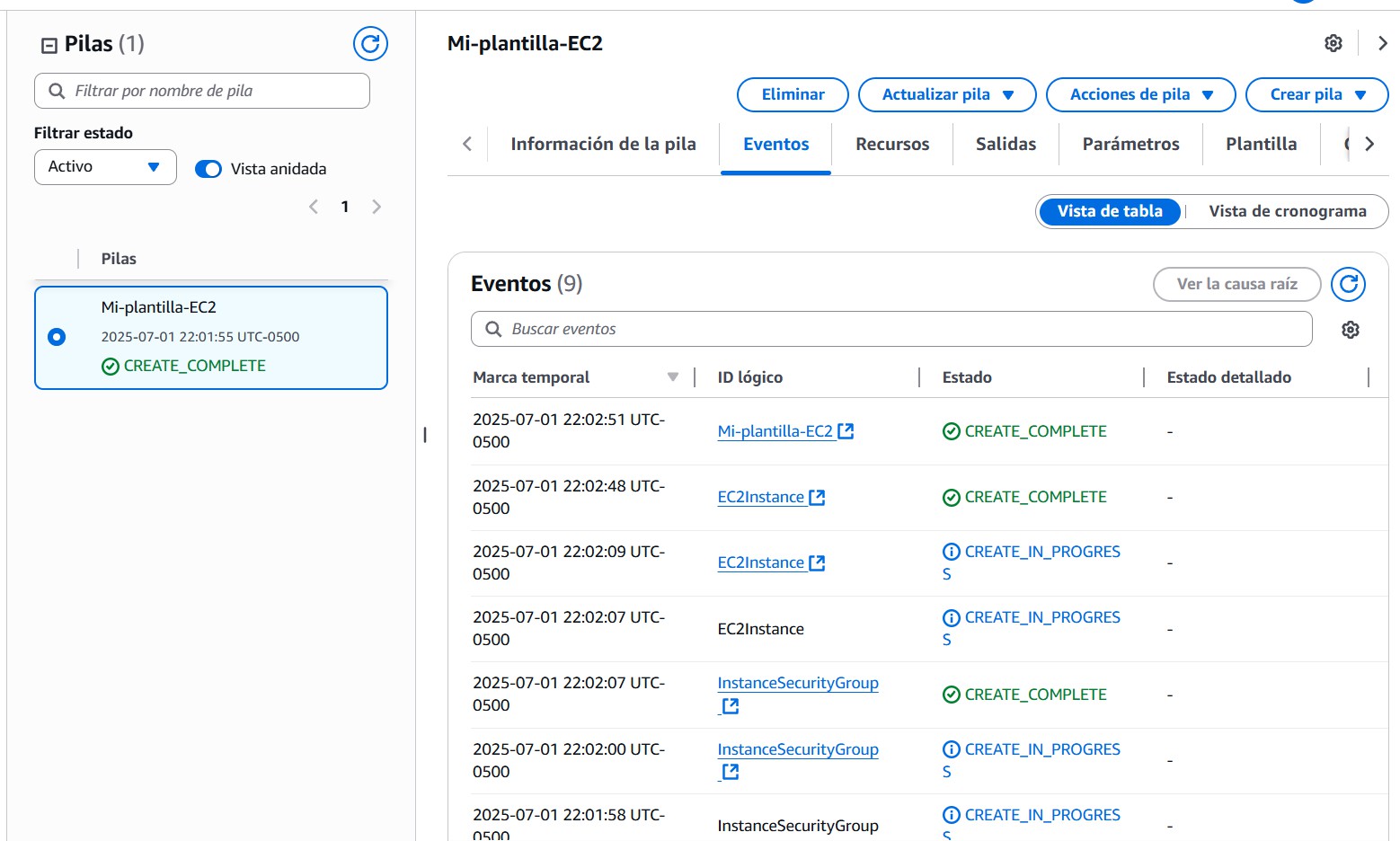
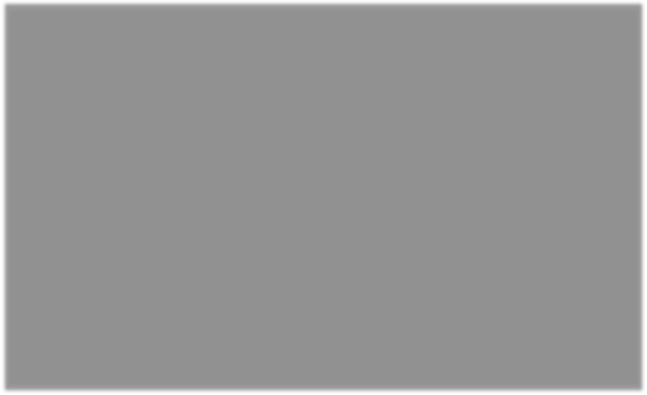
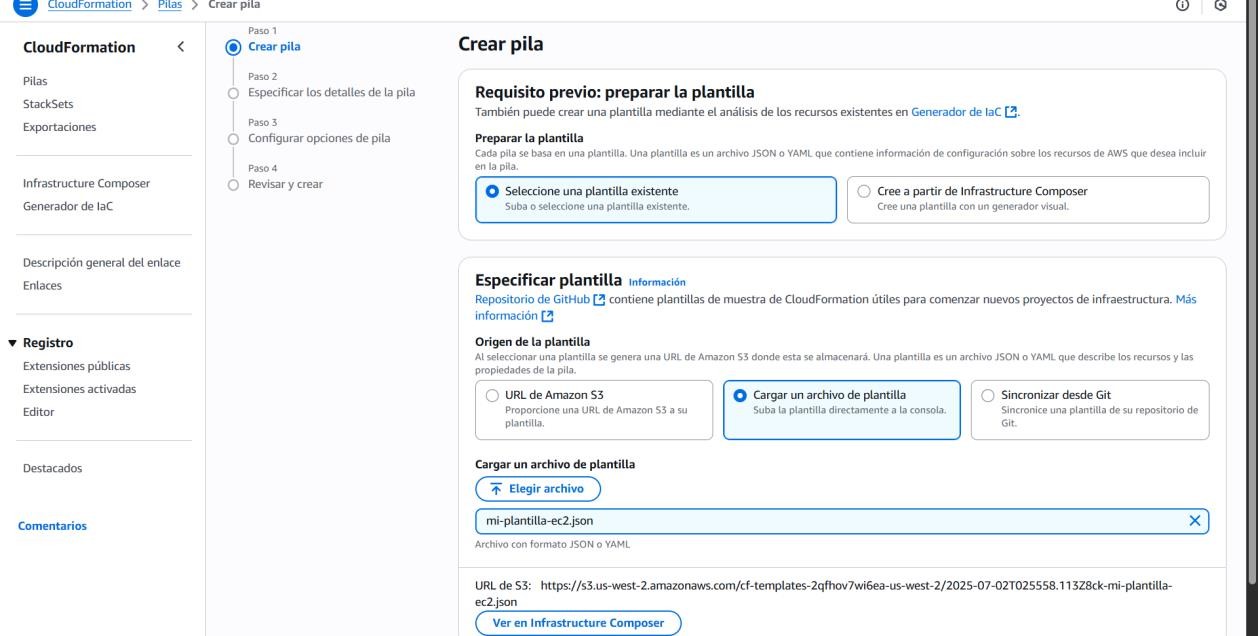
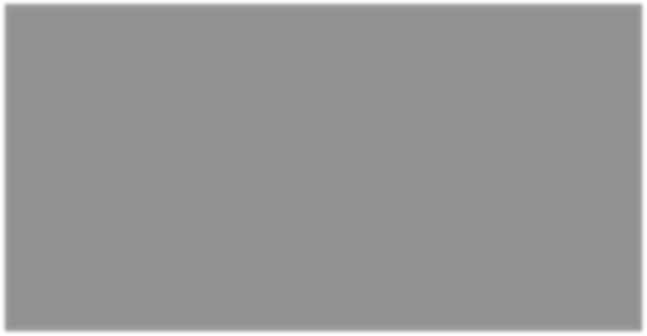
Plantilla JSON Mejorada



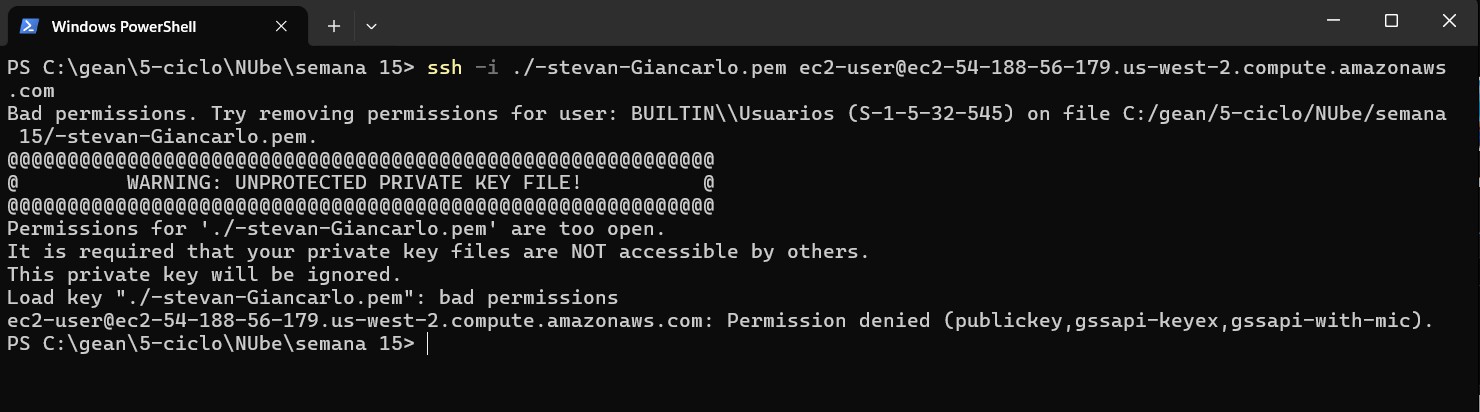
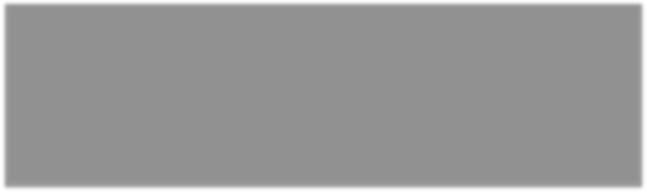
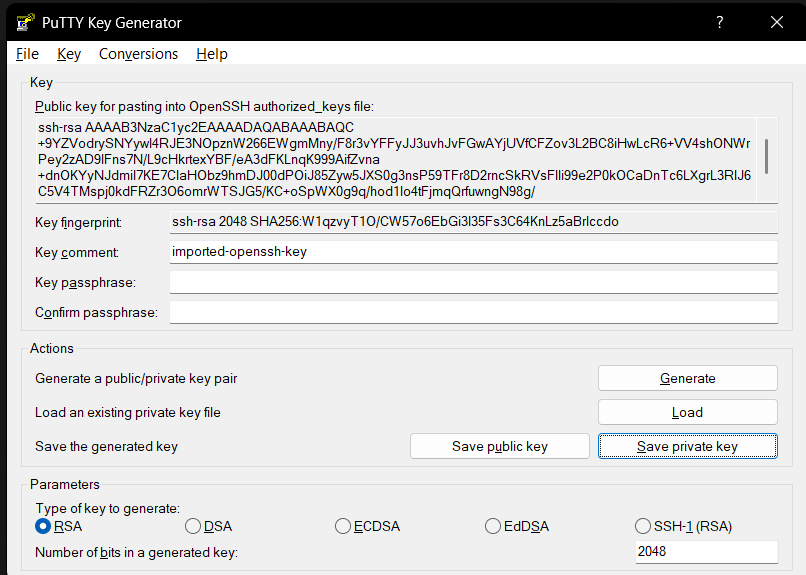
Ahora creamos la plantilla mejorada dada por la guía, pero mejorarla para que funcione correctamente.



Ahora creamos una pila con lo anteriormente mejorado.

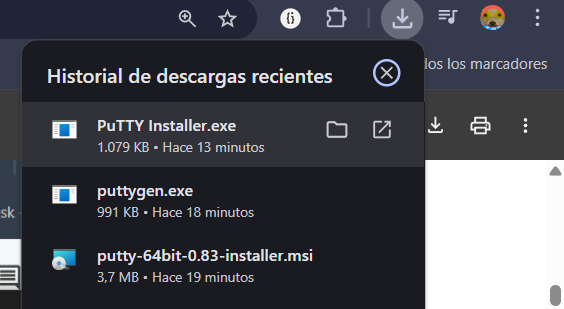
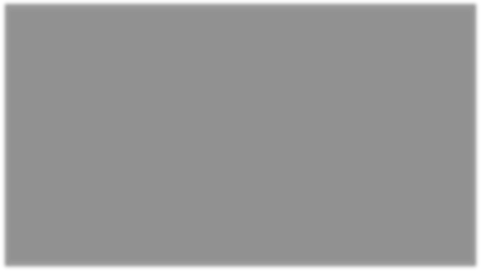


Para que de verdad funcione tenemos que colocar lo siguiente :

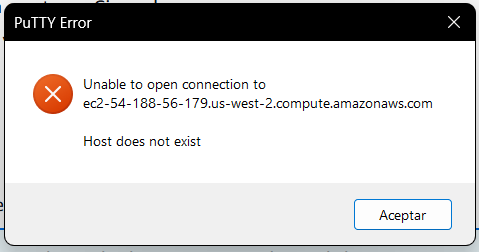
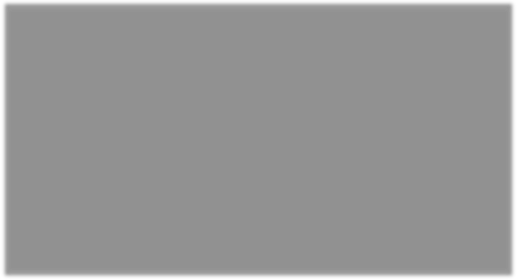


Pero se perdió completamente, no se pudo ingresar, no sabemos por qué motivo, pero si se llegó a completar todo.

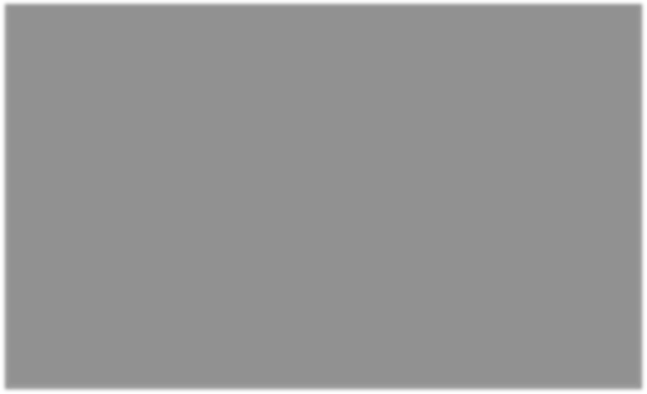
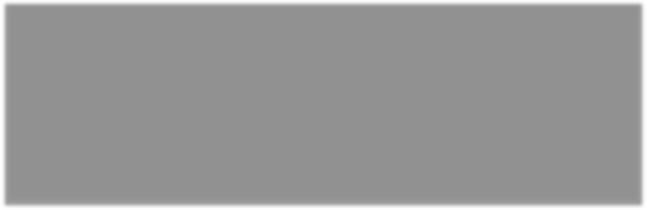
Los programas descargados como:



Pero al final nos dio error al tratar de modificar.

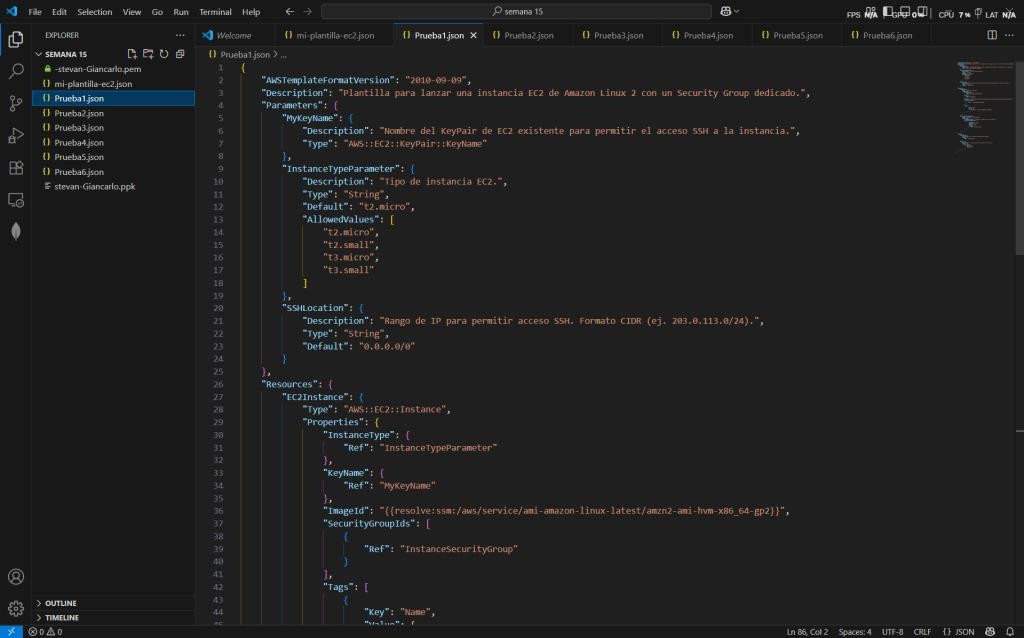
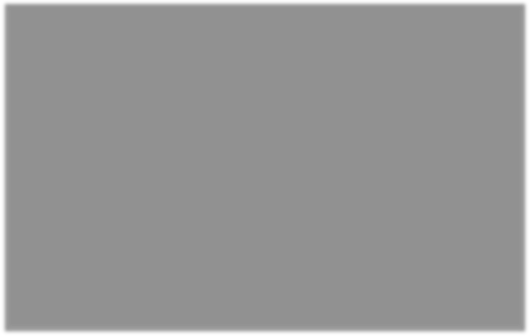


Al final probamos otra cosa para ver si funcionaba pero la opción que teníamos no esta funcional.

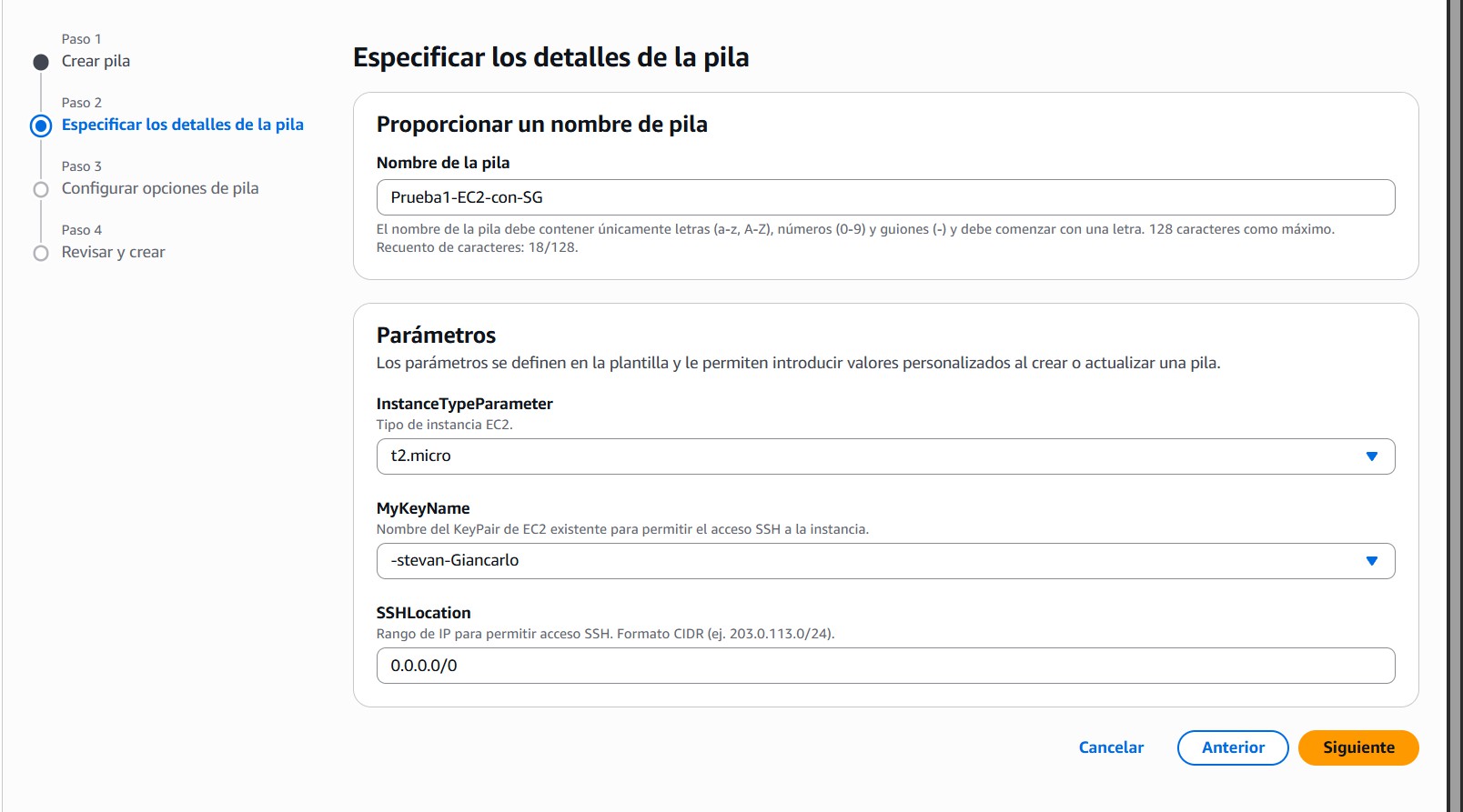
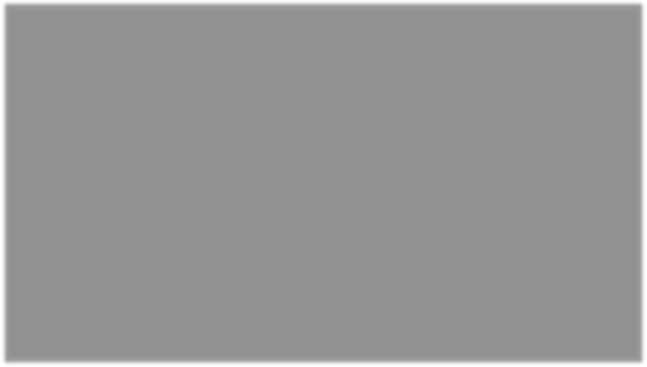


# Prueba 1: Despliegue de Instancia EC2 Básica

En esta prueba, utilicé una plantilla de CloudFormation para lanzar mi primera instancia EC2 de Amazon Linux de forma automatizada. La plantilla no solo creó la máquina virtual, sino también un **grupo de seguridad** asociado para permitir la conexión por SSH. Verifiqué que todo se creó correctamente revisando los recursos en la consola de EC2, donde confirmé que tanto la instancia como sus reglas de seguridad estaban activas.

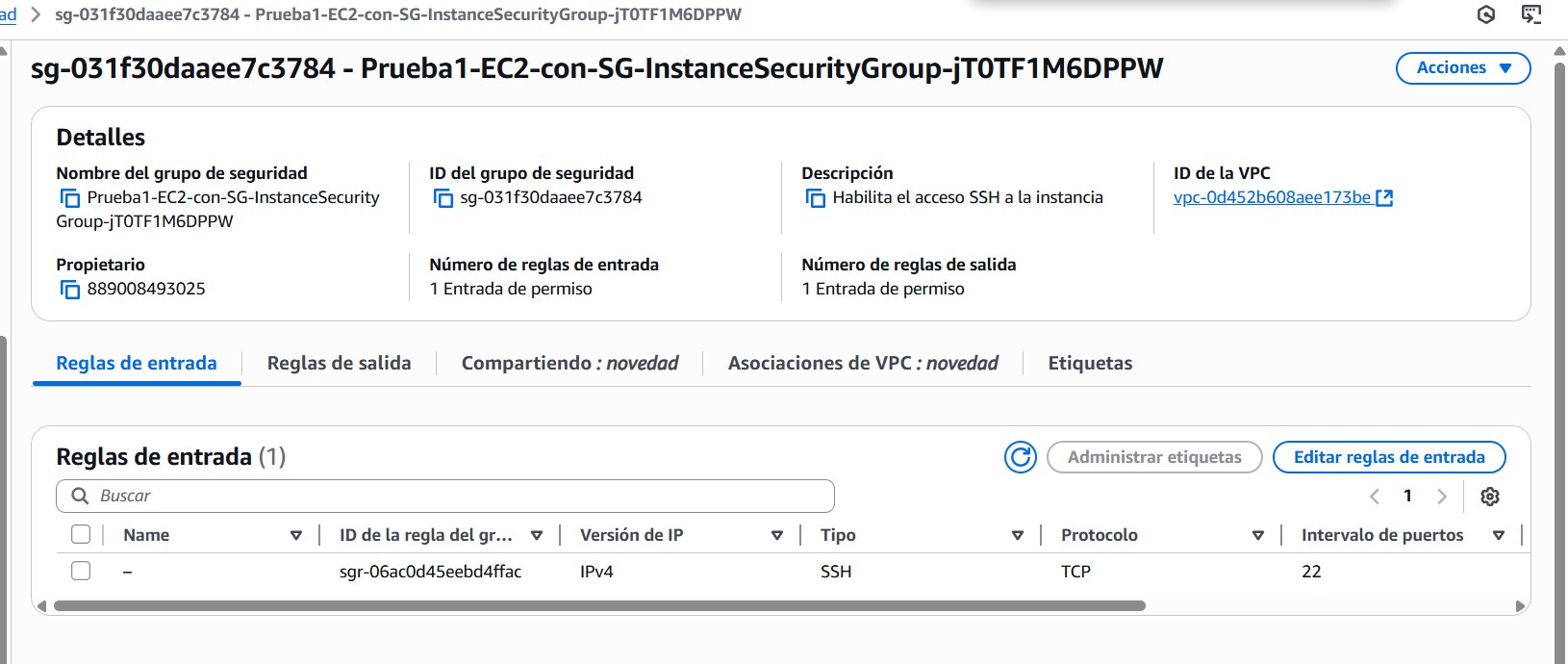
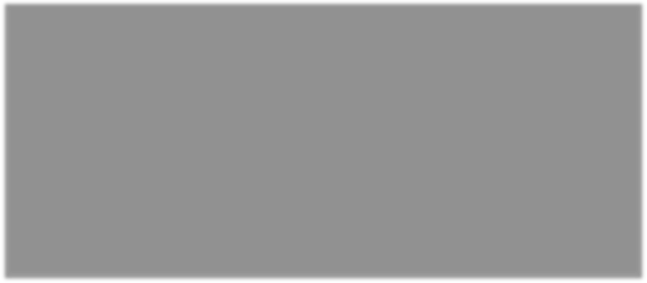
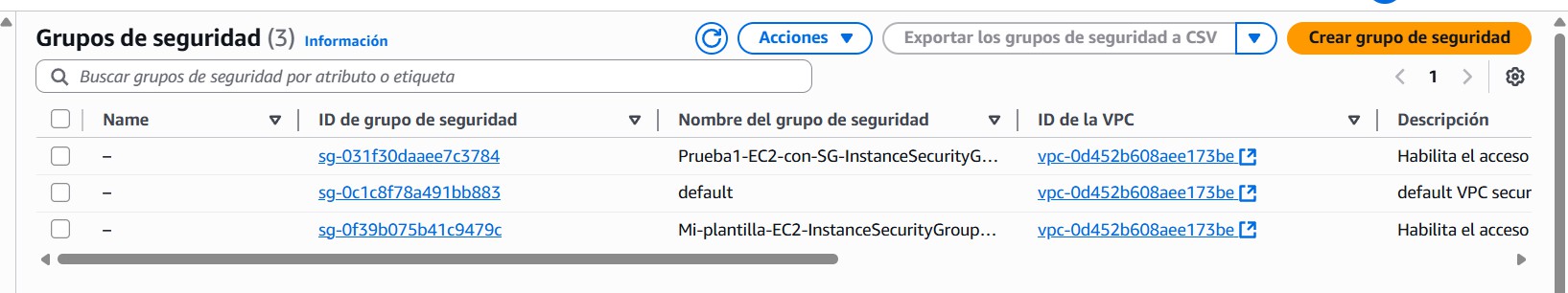
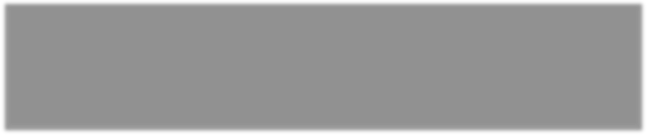
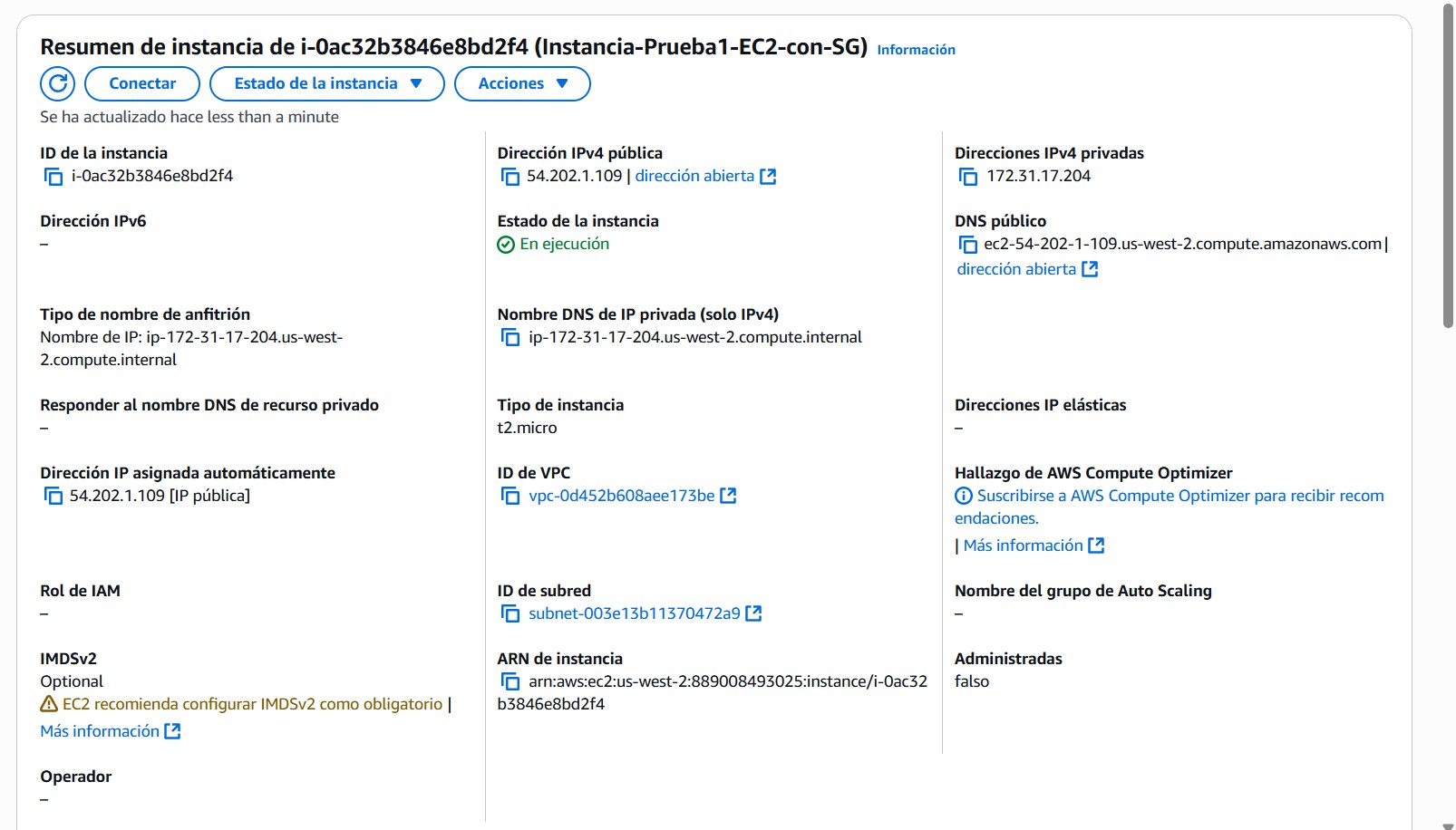
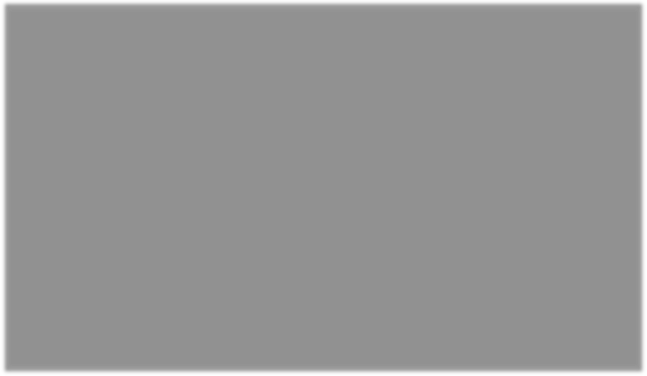
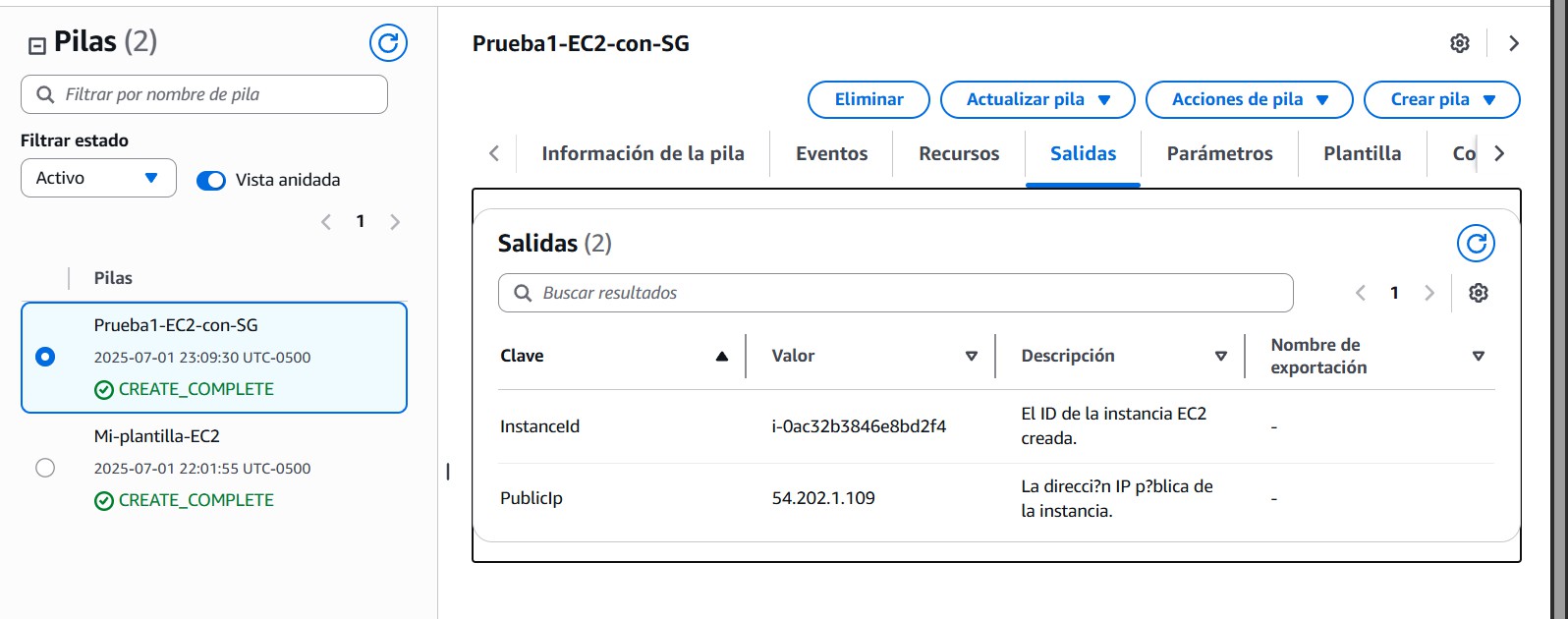
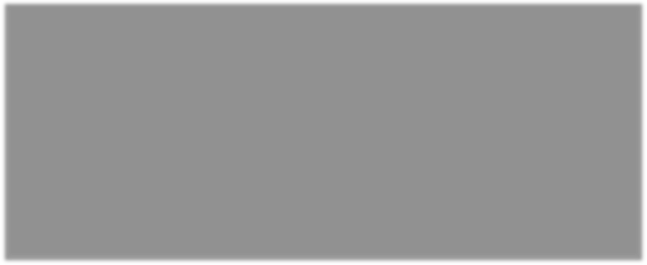
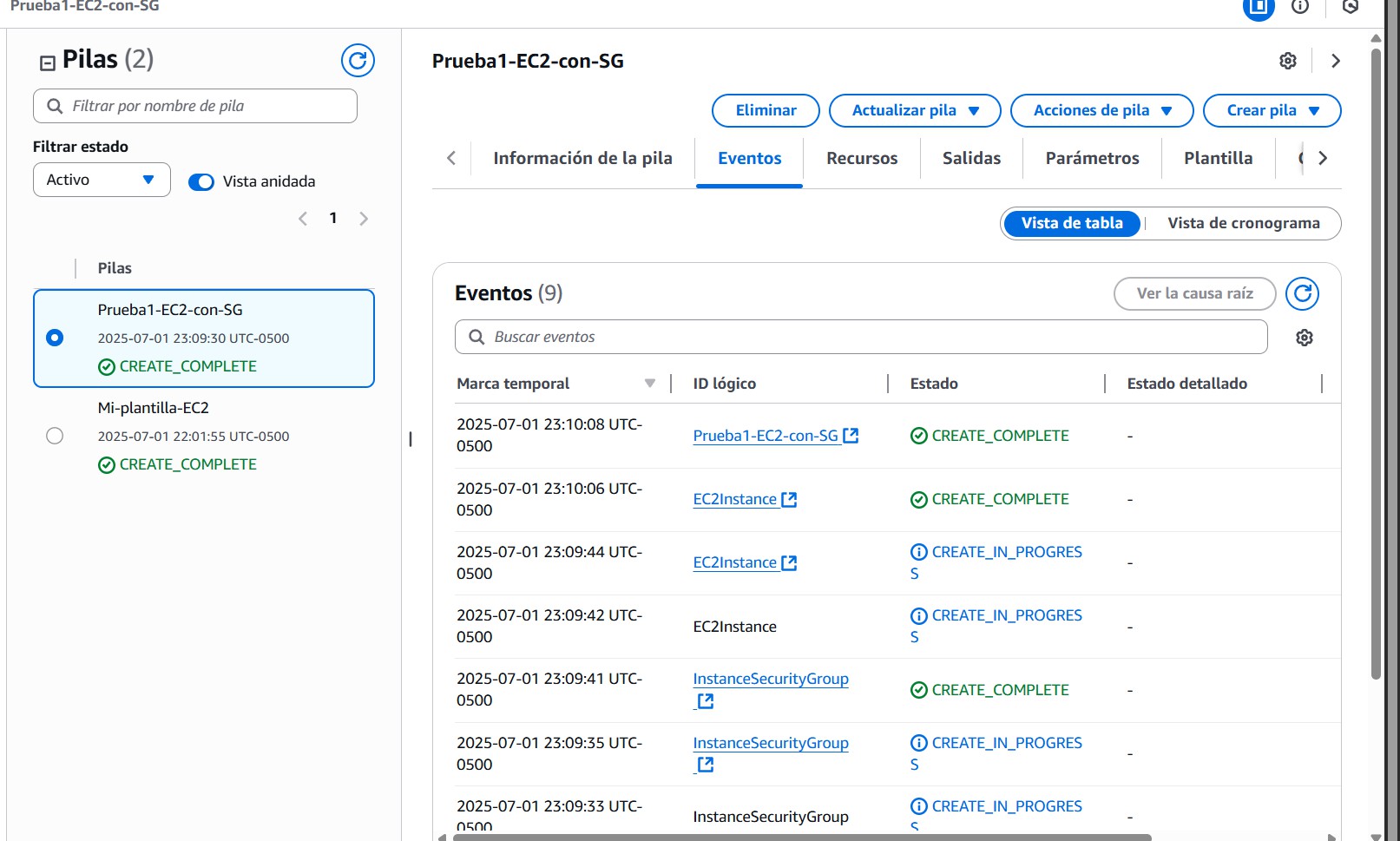
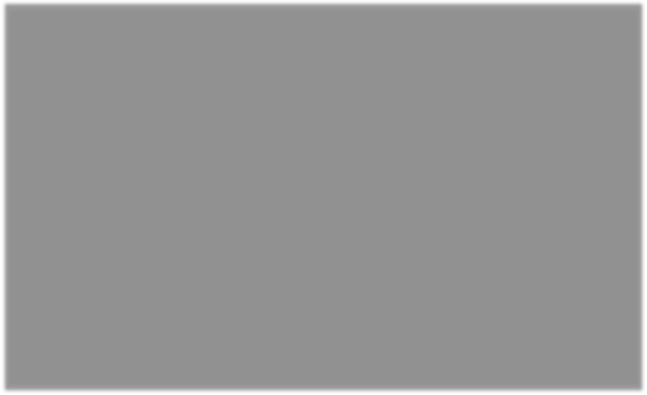


Verificamos si fue creada correctamente.



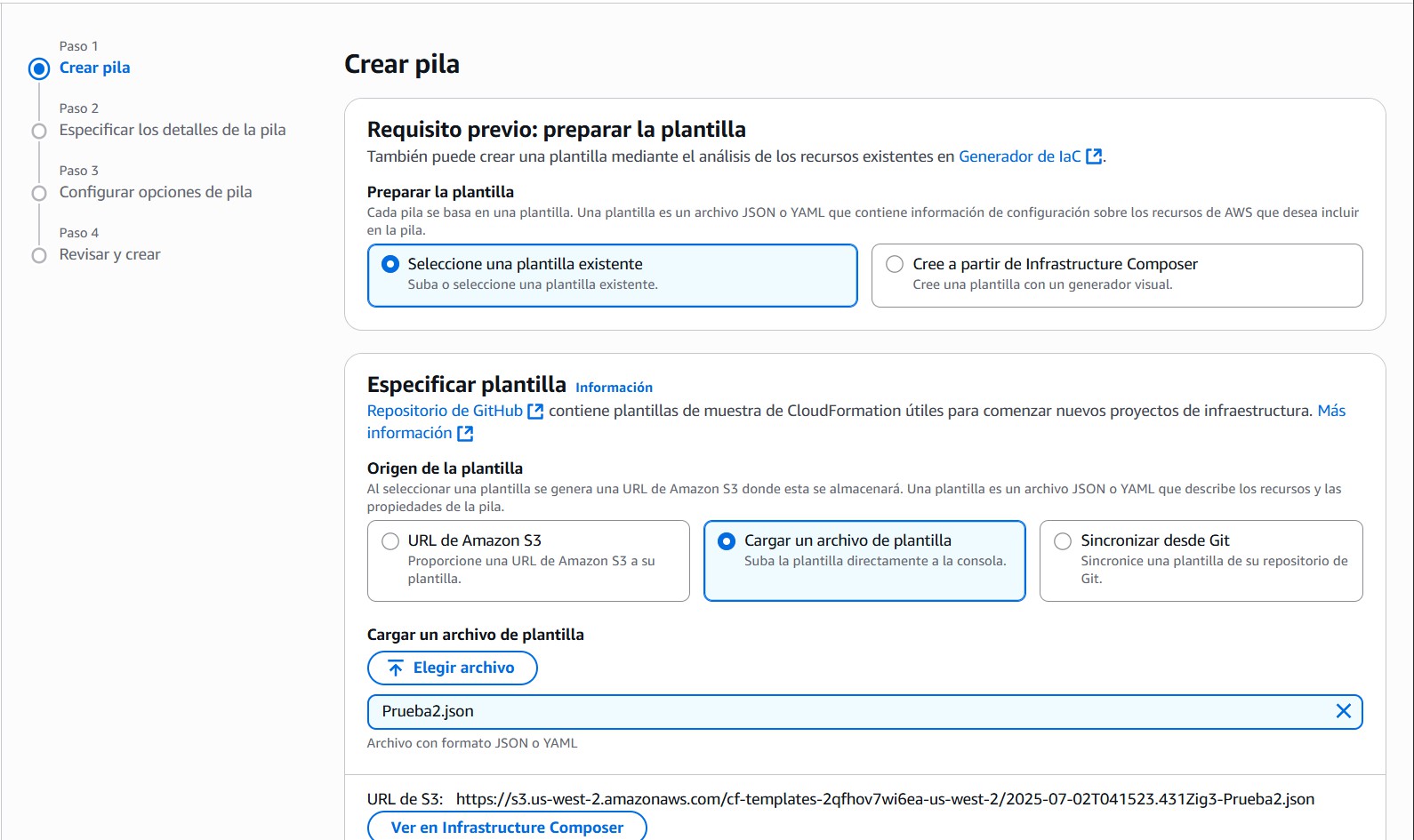
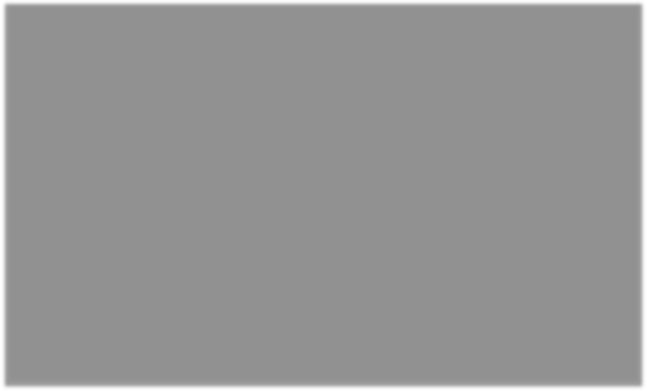
## Una Instancia EC2 con Grupo de Seguridad

* **a. Análisis de la Estructura (JSON)** El JSON se organiza en Parameters para personalizar la entrada (como el KeyPair), Resources donde se definen la instancia EC2 y su grupo de seguridad, y Outputs para exponer datos útiles como la IP pública. Lo más interesante fue ver cómo el recurso de la instancia usa una referencia (Ref) para conectarse directamente con el grupo de seguridad creado en la misma plantilla, demostrando la gestión de dependencias.
* **b. Análisis del Diseñador (Designer)** En el modo diseñador, la estructura es muy clara y simple: se ve un componente para la **EC2Instance** conectado directamente con una flecha al componente **InstanceSecurityGroup**. Esto resalta visualmente que la configuración de red de la instancia depende directamente de ese grupo de seguridad.
* **c. Propósito y Caso de Uso** Esta plantilla está pensada para el **despliegue rápido y automatizado de un servidor individual**. Soluciona el requerimiento de un equipo de desarrollo de aprovisionar velozmente un entorno de pruebas o un servidor de desarrollo con una configuración de red segura y predefinida. Es ideal para una única instancia que se necesita configurar de forma repetible.

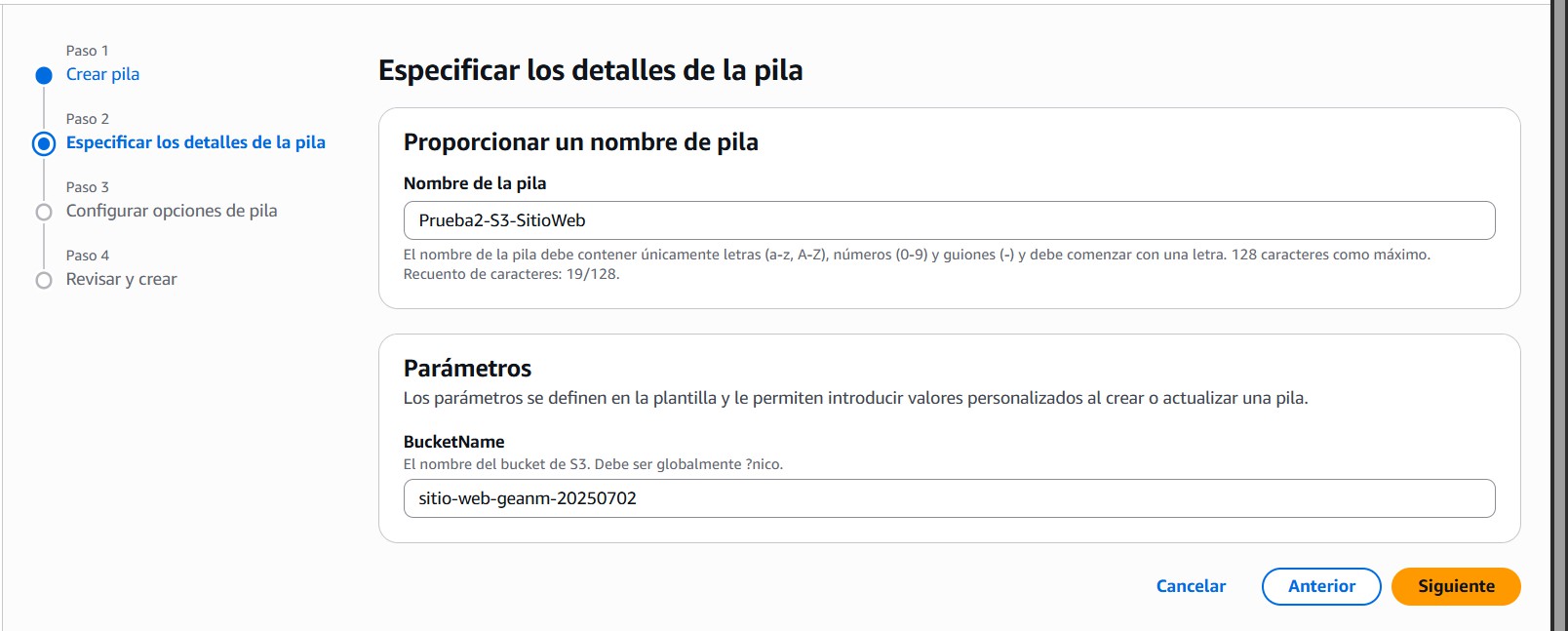
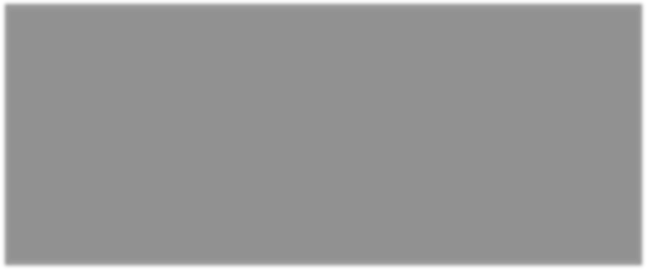


# Prueba 2: Creación de un Sitio Web Estático en S3

Aquí, desplegué un **bucket de S3** configurado para alojar un sitio web estático. La plantilla definió el bucket y una política pública para permitir el acceso a los archivos. Para que funcionara correctamente, fue crucial ajustar la **configuración de bloqueo de acceso público** que AWS aplica por defecto. La verificación final fue exitosa: subí un archivo index.html de prueba al bucket y accedí a la URL pública que CloudFormation me proporcionó.

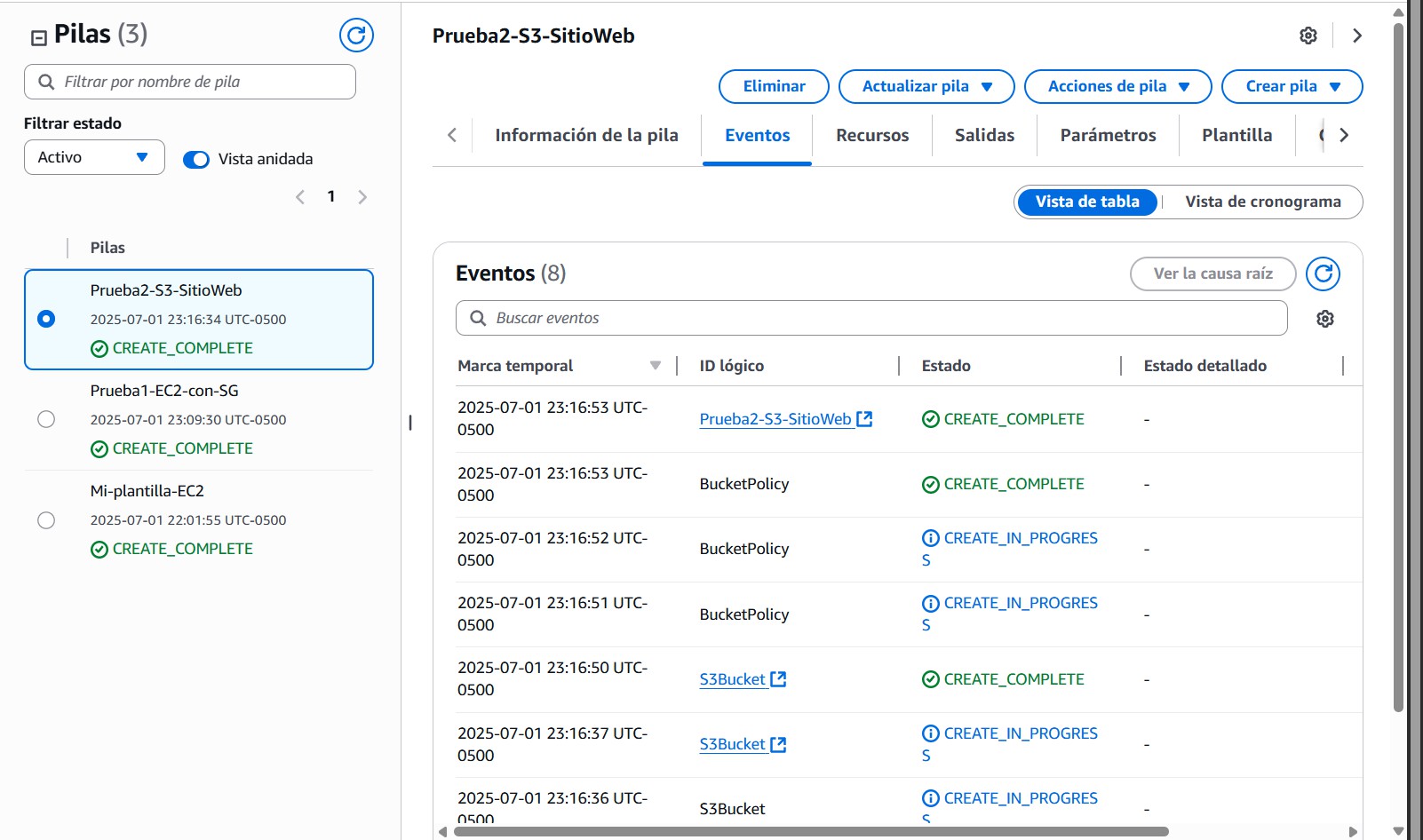
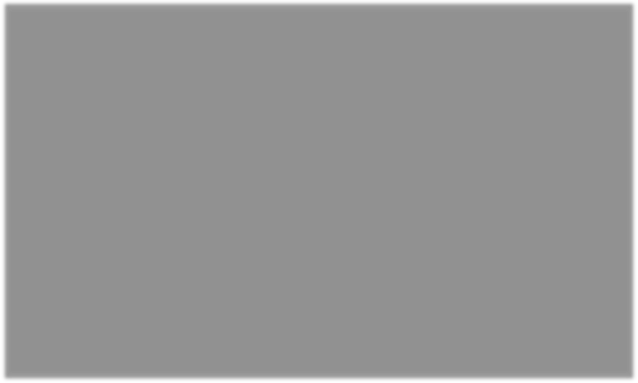


Ahora veremos que se creó correctamente esta pila.

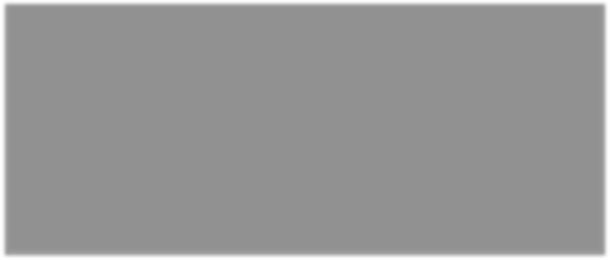
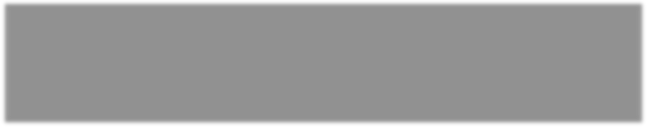
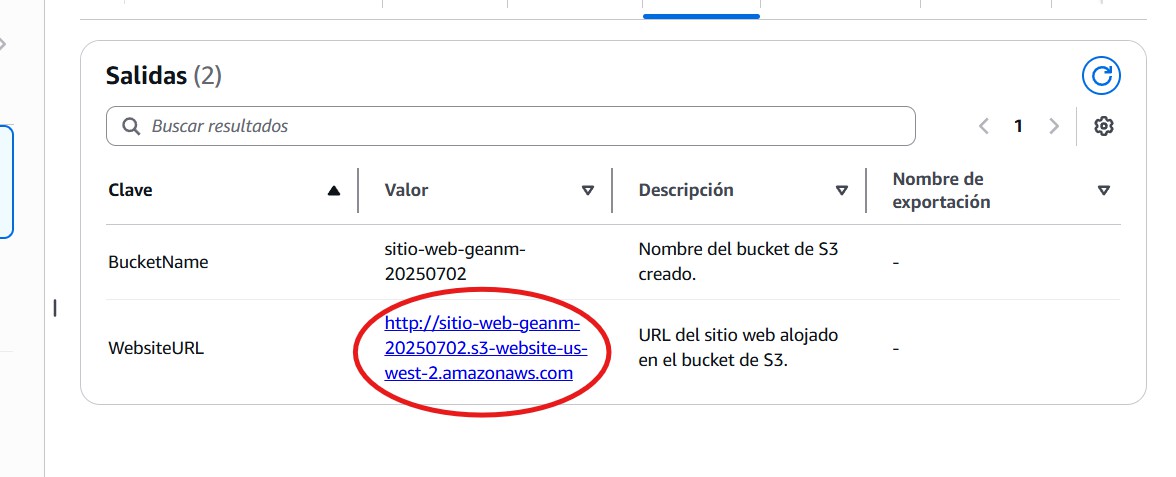
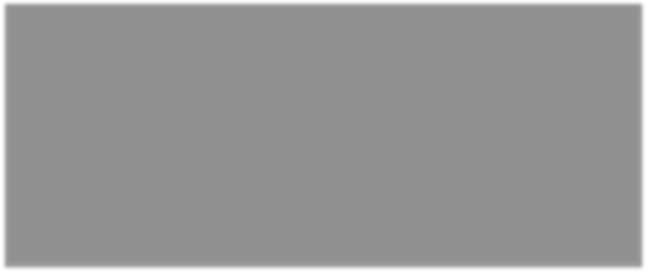
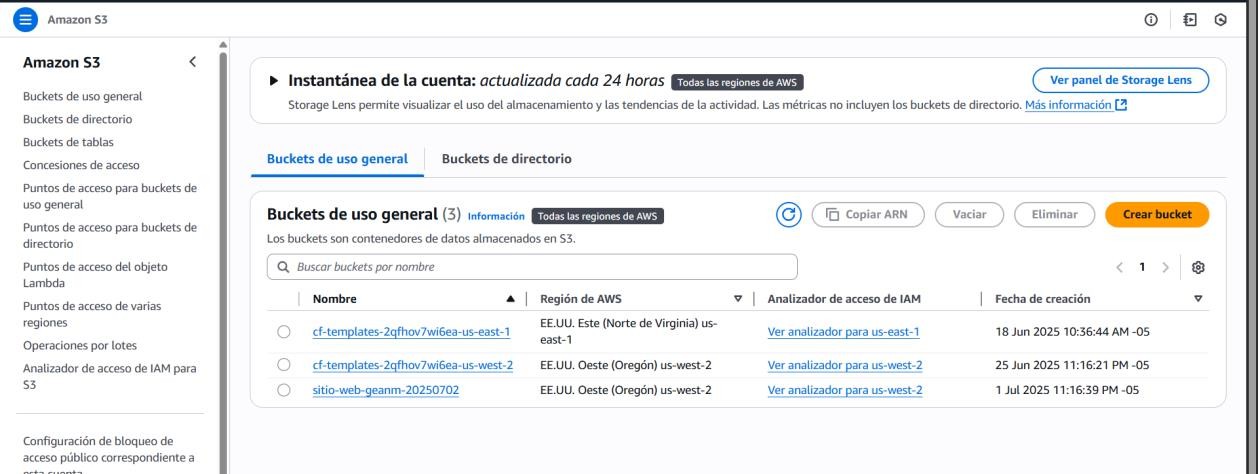
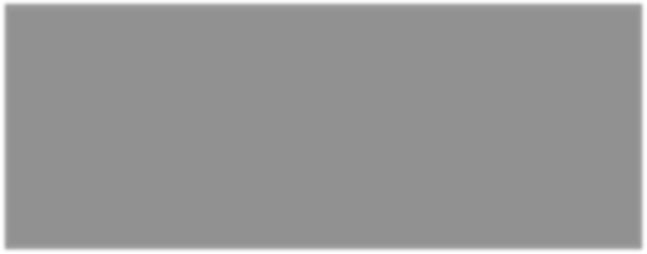


## Un Sitio Web Estático en S3

* **a. Análisis de la Estructura (JSON)** Identifiqué dos recursos principales: AWS::S3::Bucket para crear el bucket y AWS::S3::BucketPolicy para aplicar la política de acceso público. Lo más llamativo fue la propiedad **WebsiteConfiguration** dentro del bucket, que lo habilita como un sitio web al definir los documentos de índice y error. En ejecución, esto crea un bucket listo para servir contenido estático a través de una URL de AWS.
* **b. Análisis del Diseñador (Designer)** El diagrama es muy sencillo y muestra el recurso **S3Bucket** vinculado directamente a su política de acceso. Esto representa visualmente que la política se aplica directamente sobre el bucket para controlar sus permisos.
* **c. Propósito y Caso de Uso** La plantilla está diseñada para crear rápidamente un **repositorio de bajo costo para alojar sitios web estáticos**. Permite a un equipo desplegar un *frontend* (hecho en React, Angular, etc.) o una página de documentación de forma económica y sin gestionar servidores. Debe usarse para alojar contenido que no requiere procesamiento del lado del servidor.

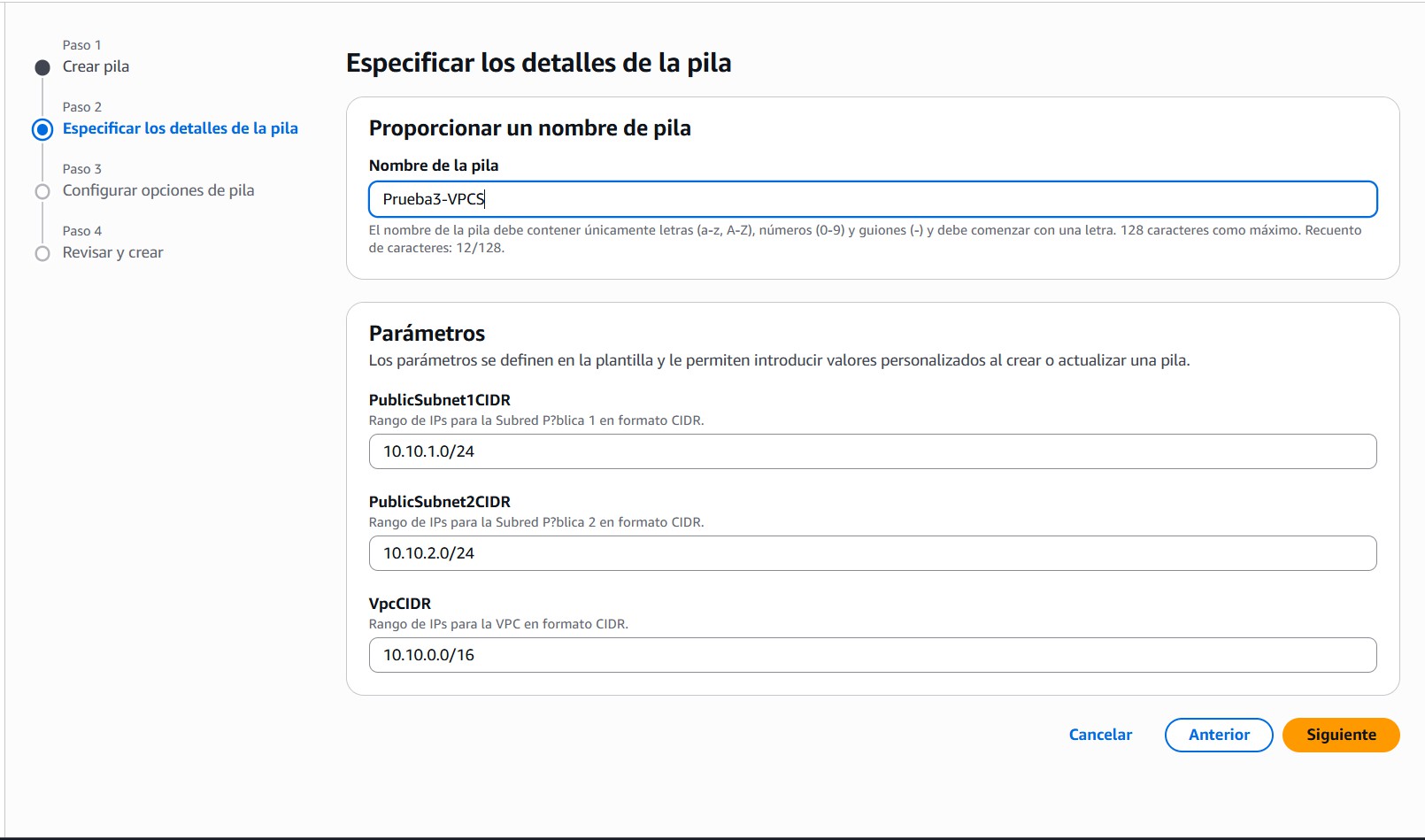
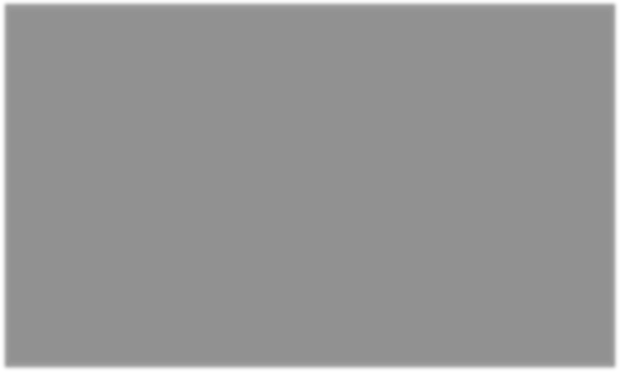
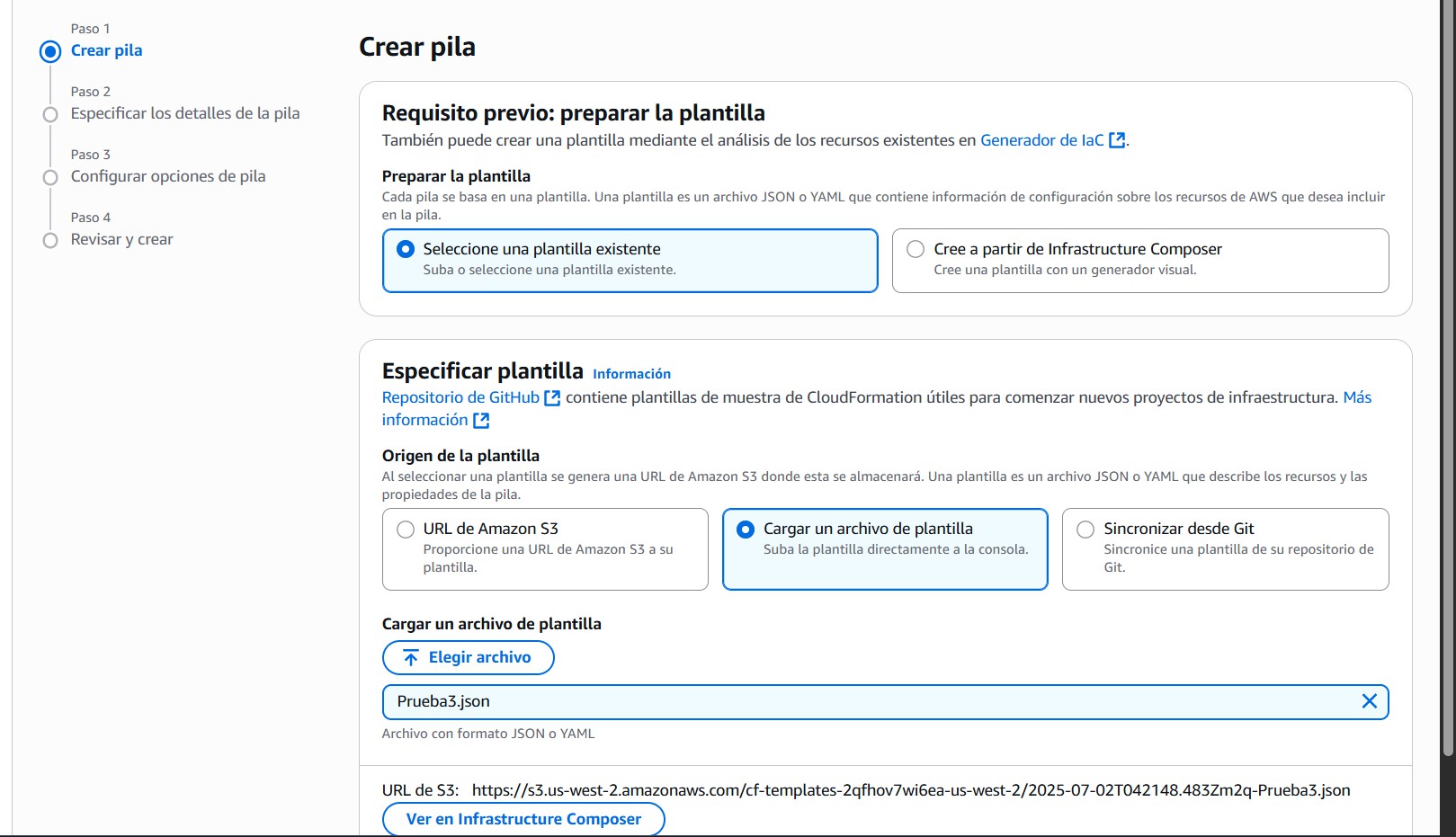
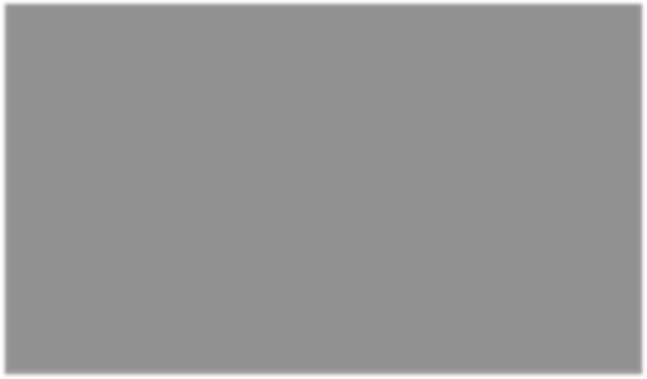


Para probar que de verdad funciona vemos que se cree el bucket y subir un index.html.

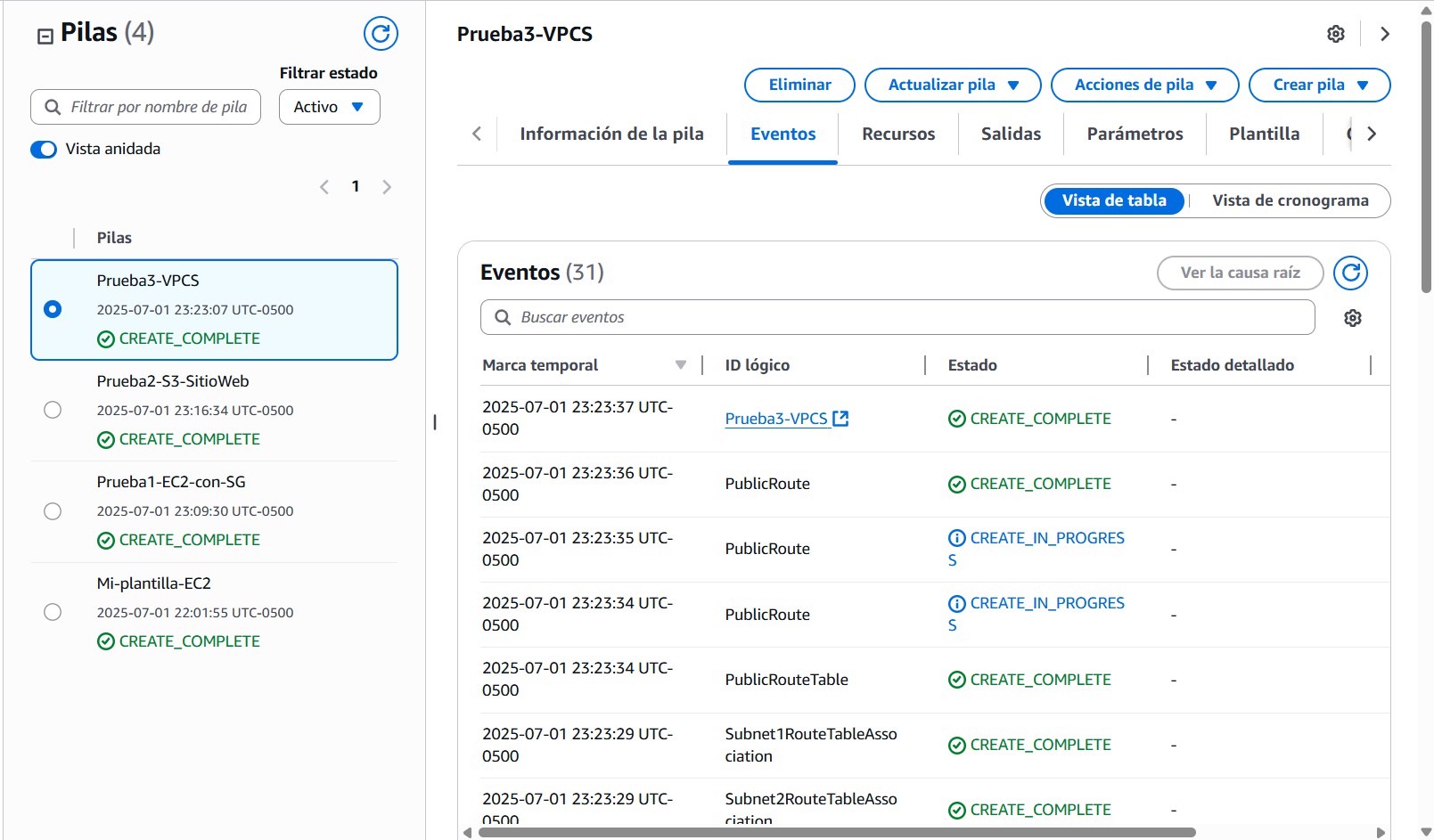
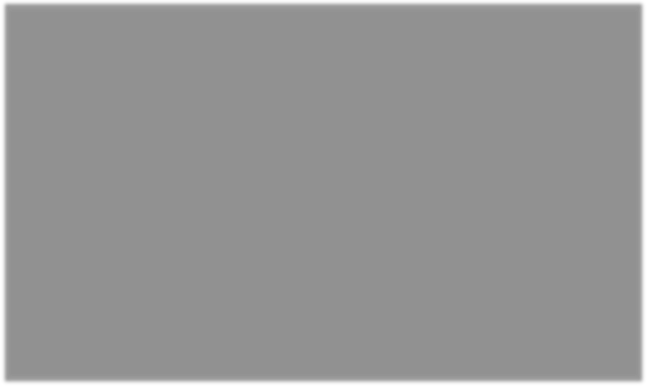


# Prueba 3: Implementación de una Red VPC Fundamental

En este caso, usé CloudFormation para construir la infraestructura de red base: una **Virtual Private Cloud (VPC)** con dos subredes públicas en diferentes zonas de disponibilidad. La plantilla también se encargó de configurar un **Internet Gateway** y las tablas de rutas necesarias para asegurar la conectividad a internet. Comprobé la creación de todos estos componentes directamente en la consola de VPC, confirmando que la red estaba bien estructurada.

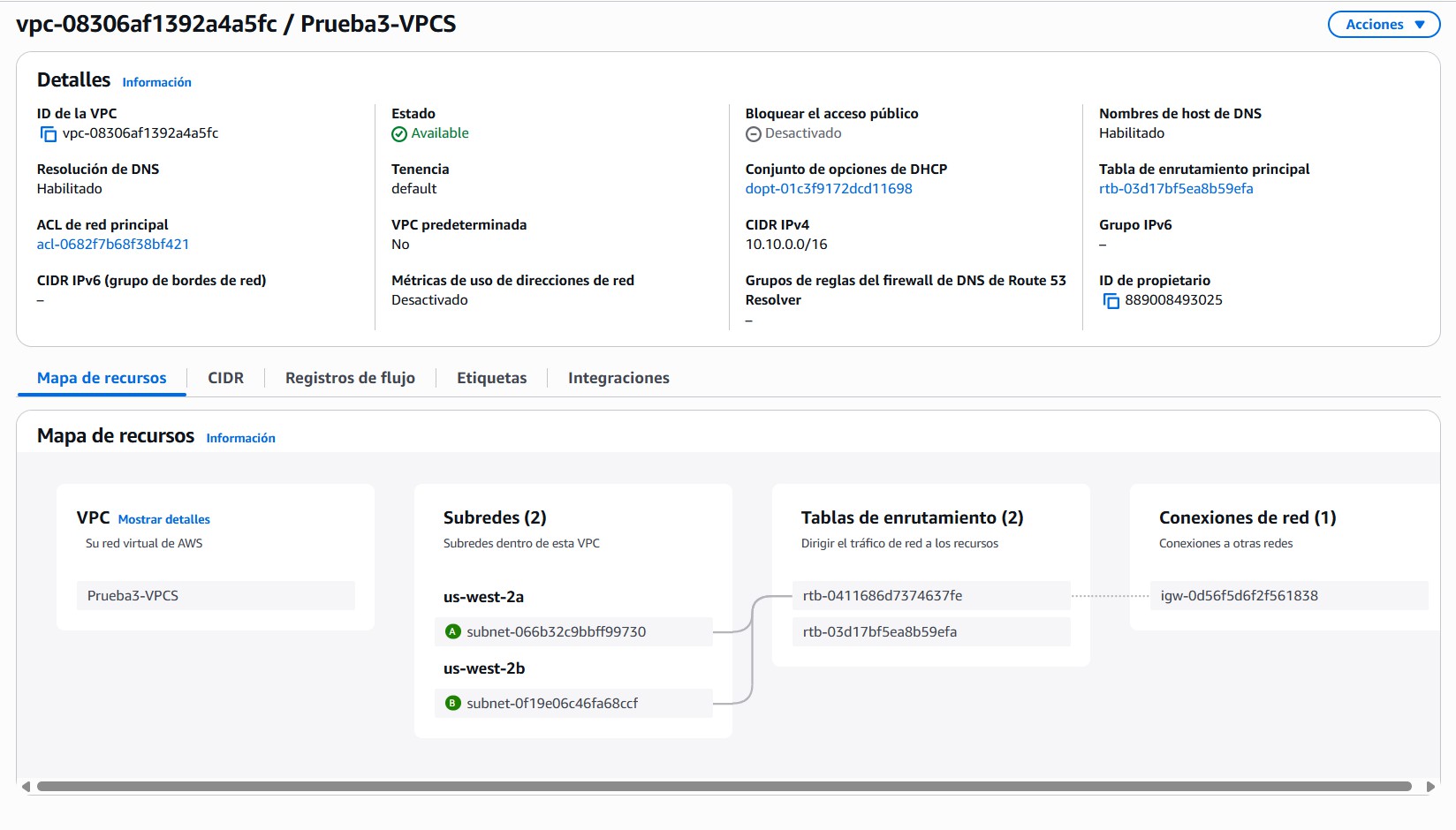
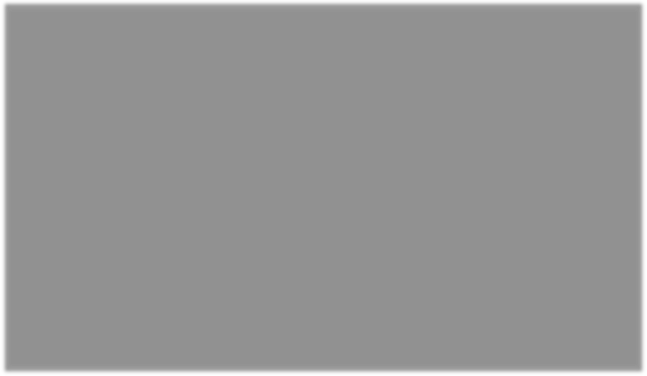
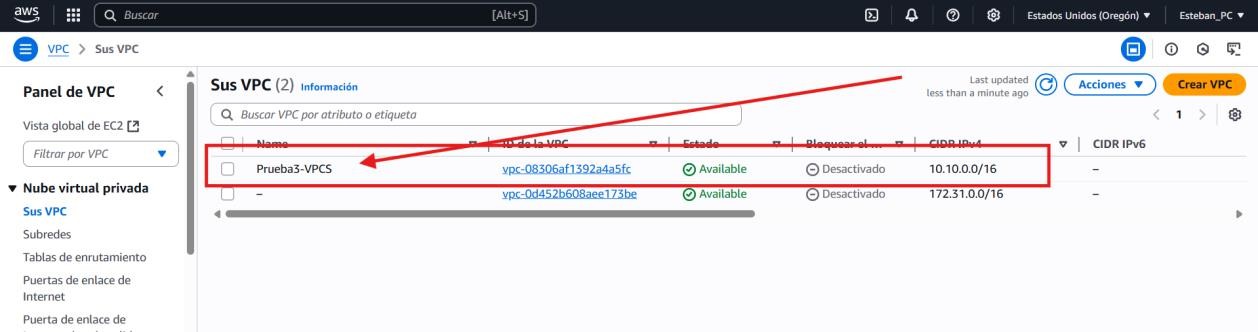
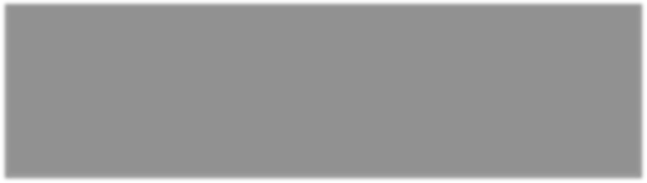


Ahora verificaremos si funciono correctamente todo.



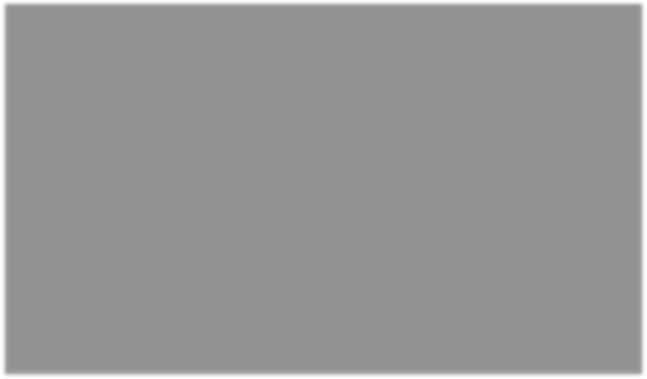
## Una Red VPC Fundamental

* **a. Análisis de la Estructura (JSON)** La plantilla define una red completa a través de múltiples recursos interconectados: VPC, Subredes, Internet Gateway y Tablas de Rutas. Lo más interesante fue ver cómo el recurso **AWS::EC2::SubnetRouteTableAssociation** es la pieza clave que convierte a las subredes en "públicas" al darles una ruta hacia el Internet Gateway.
* **b. Análisis del Diseñador (Designer)** El diseñador muestra la **VPC como un gran contenedor** que agrupa visualmente todos los demás componentes de red (subredes, tabla de rutas, etc.). Esto ilustra perfectamente el concepto de una red autocontenida y aislada.
* **c. Propósito y Caso de Uso** Está pensada para **establecer la infraestructura de red fundamental** y reutilizable para otros servicios. Es la solución para un equipo que necesita una red base estandarizada y segura sobre la cual desplegar aplicaciones complejas (por ejemplo, con servidores web públicos y bases de datos privadas). Es el punto de partida para casi cualquier arquitectura seria en AWS.

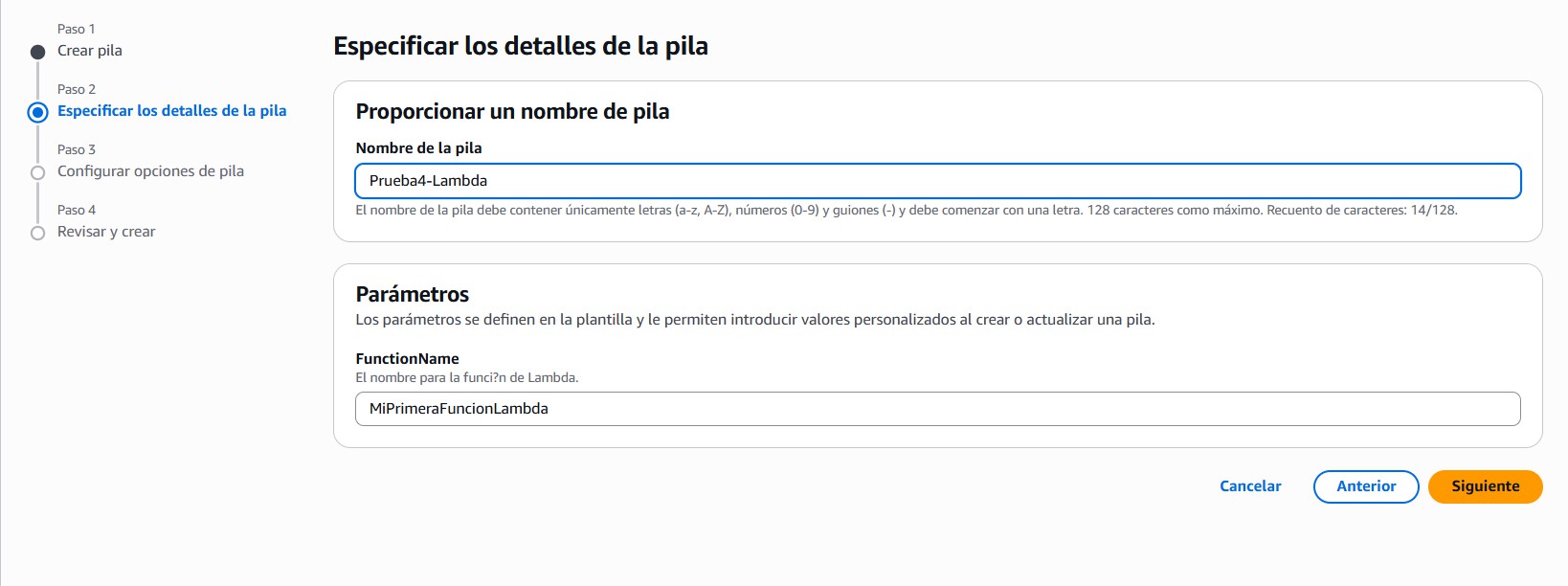
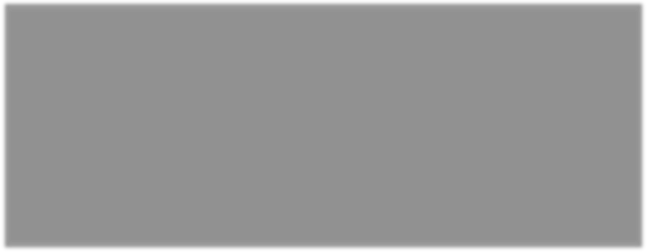


# Prueba 4: Despliegue de una Función Serverless (AWS Lambda)

Para esta prueba, desplegué mi primera función *serverless* con **AWS Lambda** usando CloudFormation. Lo más interesante fue ver cómo la plantilla se encargó de crear no solo la función con código Python embebido, sino también el **rol de IAM** con los permisos de ejecución necesarios. La verificación fue muy rápida: ejecuté un evento de prueba desde la propia consola de Lambda y confirmé que la función devolvía el mensaje de éxito esperado.



Ok ahora veremos si se creó correctamente.

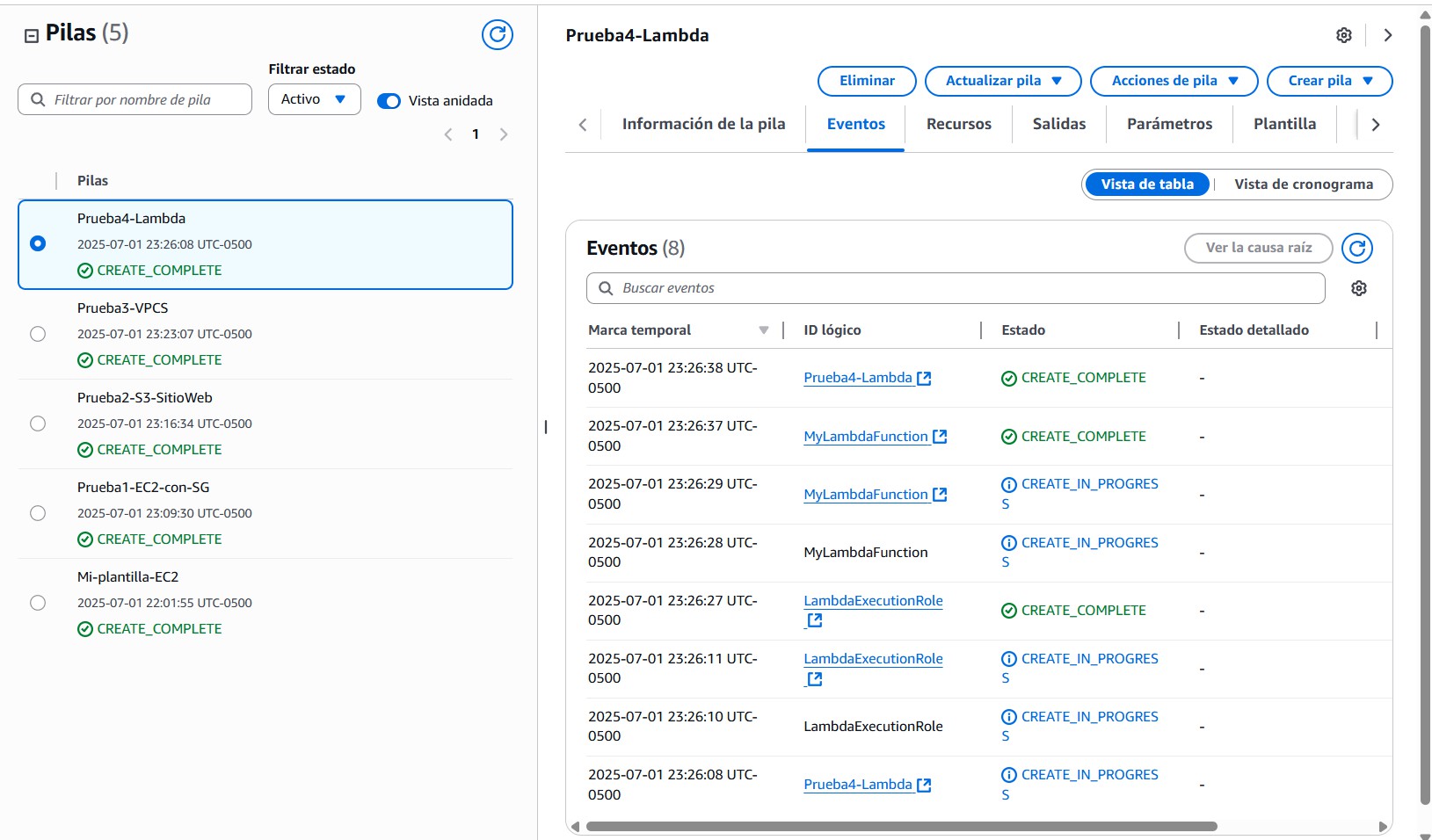
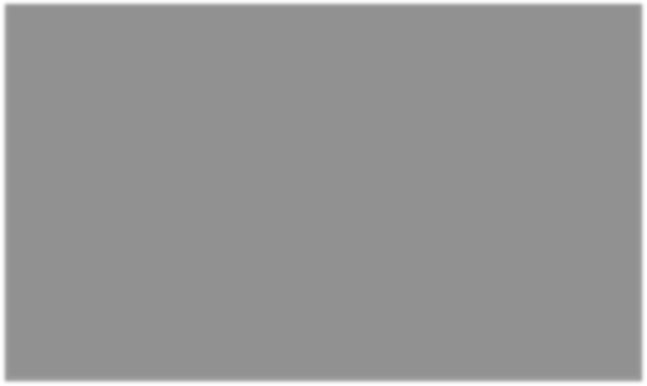


## Una Función Serverless (AWS Lambda)

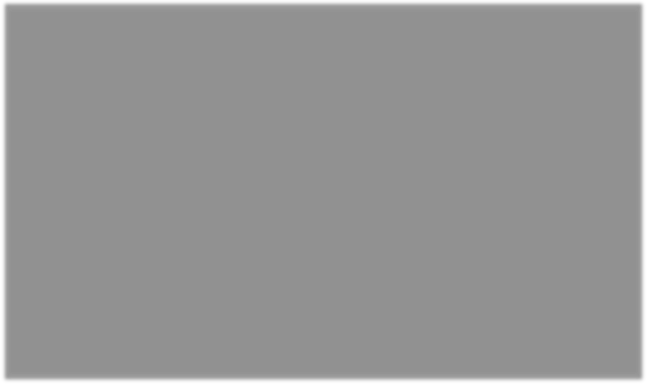
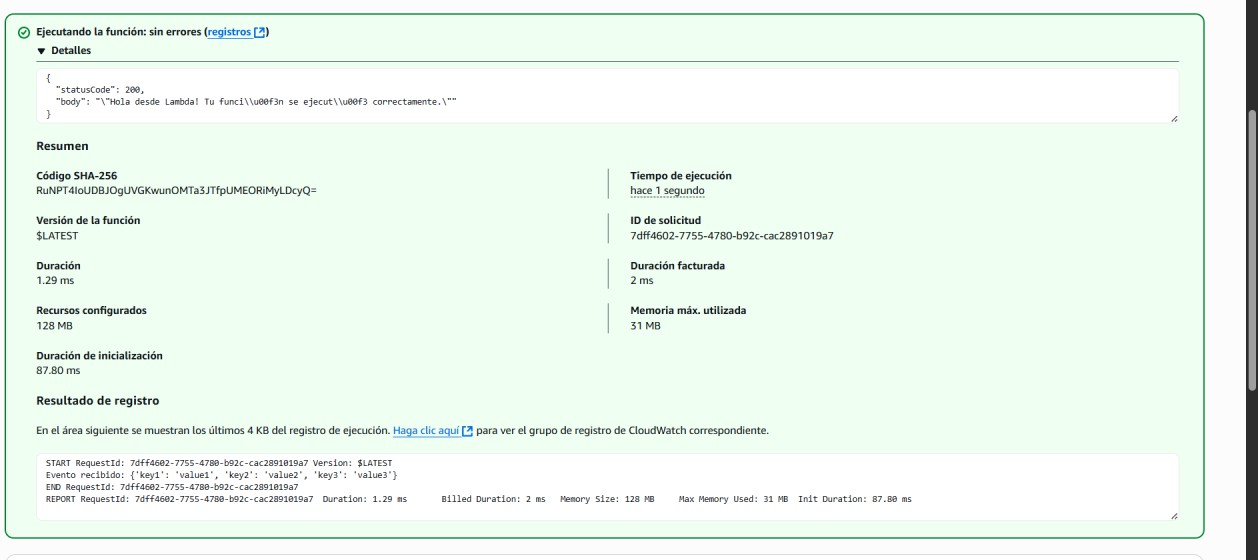
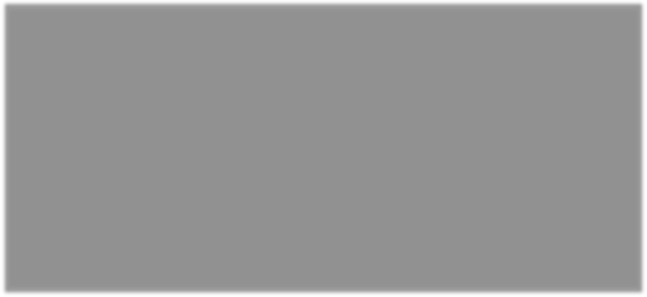
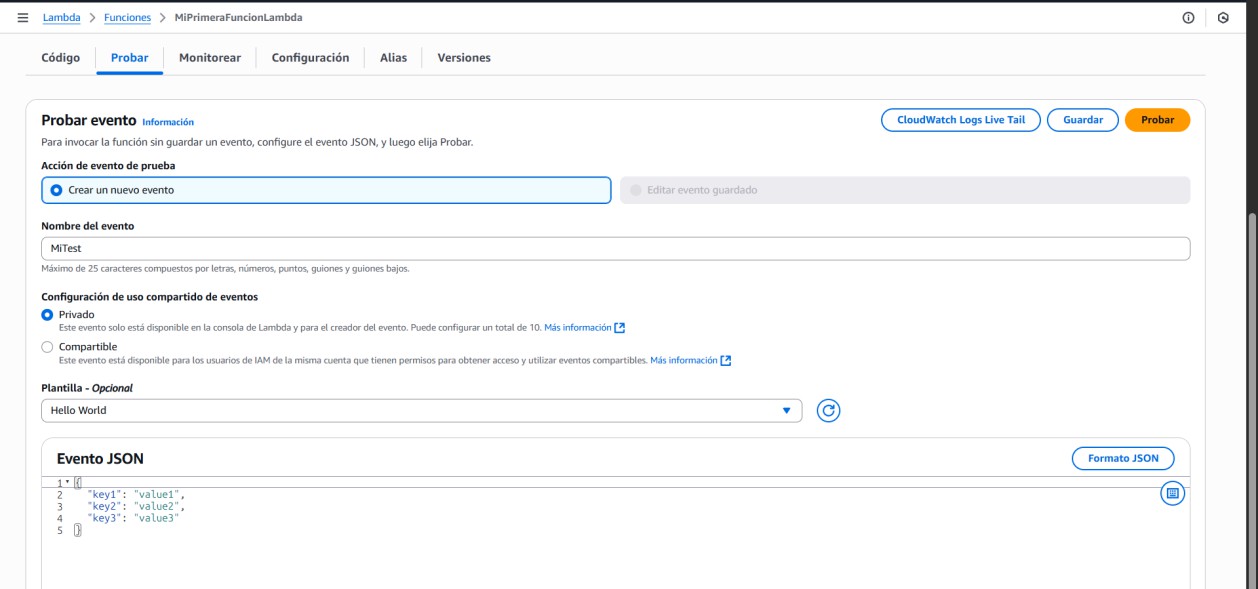
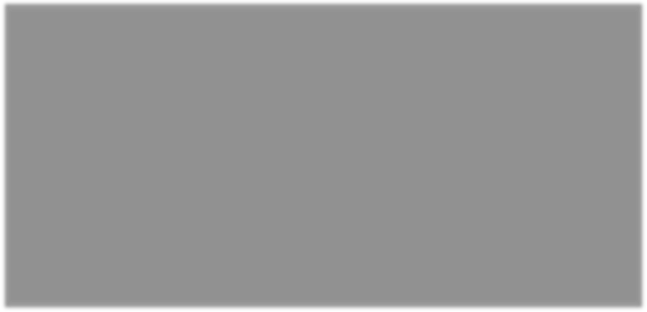
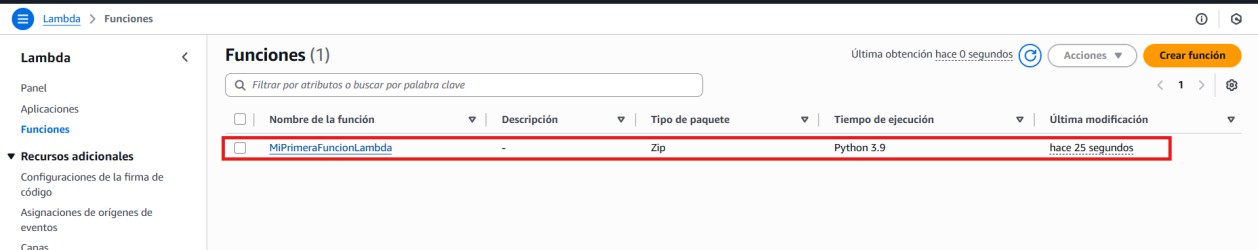
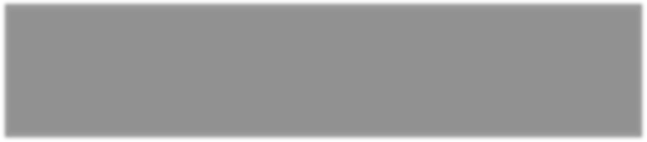
* **a. Análisis de la Estructura (JSON)** La plantilla se compone de dos recursos principales: AWS::IAM::Role para definir los permisos y AWS::Lambda::Function para

crear la función. Lo que más me llamó la atención fue la simplicidad con la que se puede **incrustar el código Python directamente en la plantilla** con ZipFile y asignar el rol de ejecución.

* **b. Análisis del Diseñador (Designer)** La vista del diseñador muestra claramente dos componentes: la **Función Lambda** y su **Rol de IAM** asociado, con una flecha que indica la asignación de permisos del rol hacia la función, haciendo la relación muy fácil de entender.
* **c. Propósito y Caso de Uso** Está diseñada para **desplegar rápidamente una pieza de lógica de negocio** en un entorno sin servidor (*serverless*). Un equipo la puede usar para crear microservicios, procesar archivos subidos a S3, ejecutar tareas programadas o crear un backend ligero para una API. Es ideal para tareas cortas y orientadas a eventos.

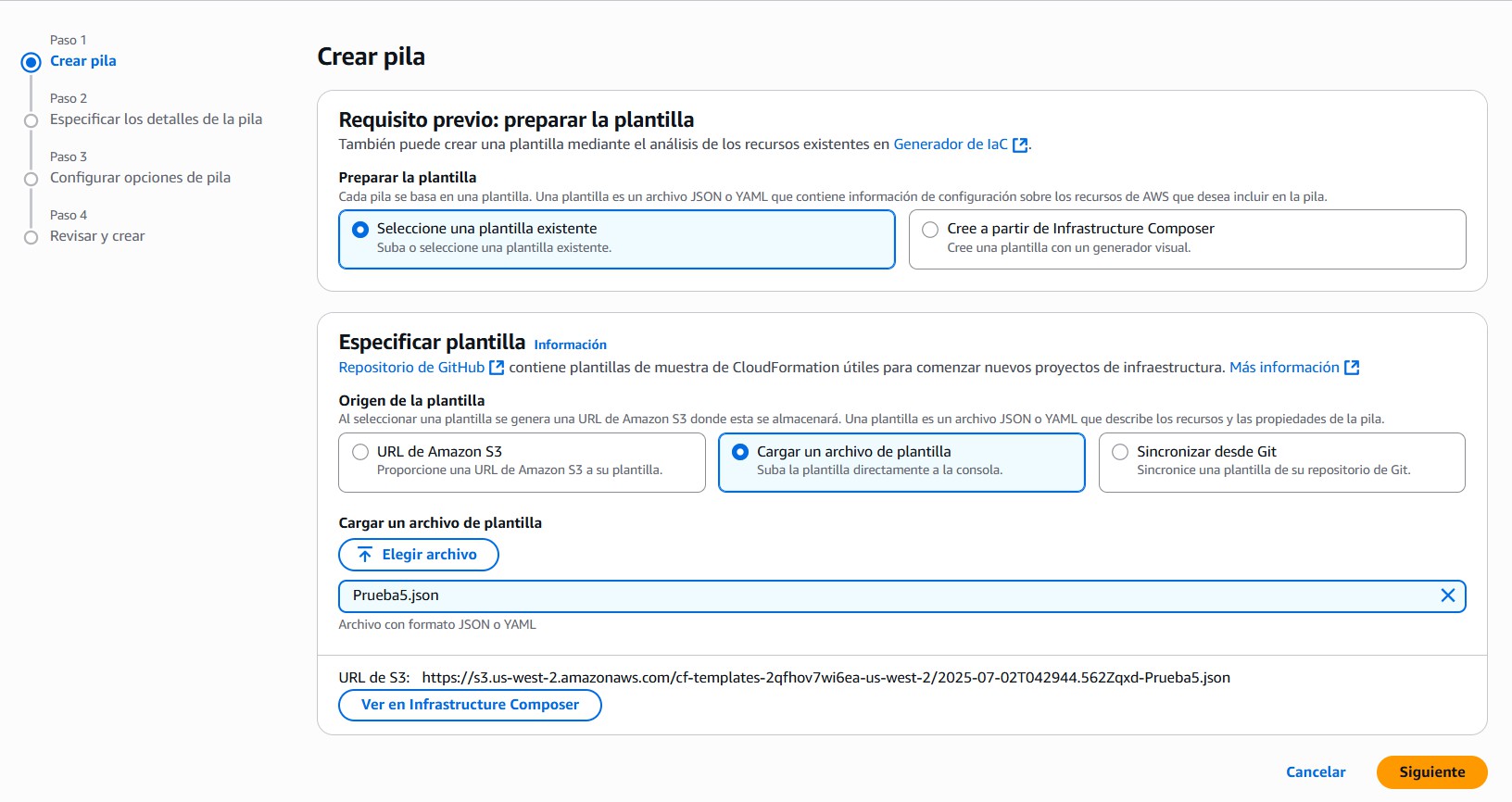
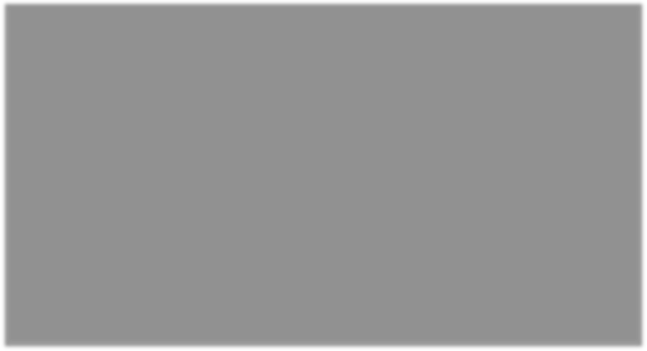


Ahora vemos que se creo correctamente.



# Prueba 5: Creación de una Base de Datos Privada en RDS

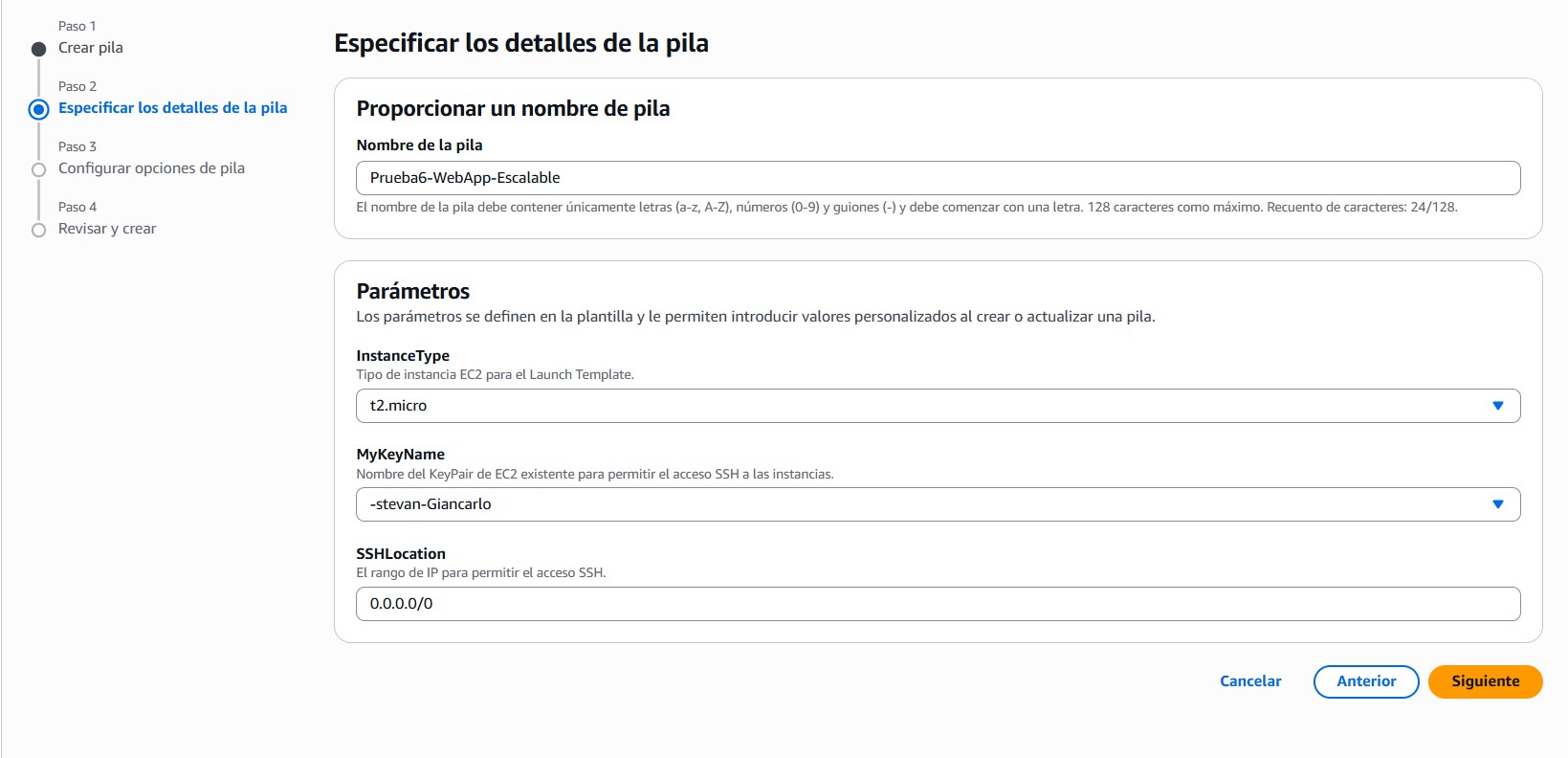
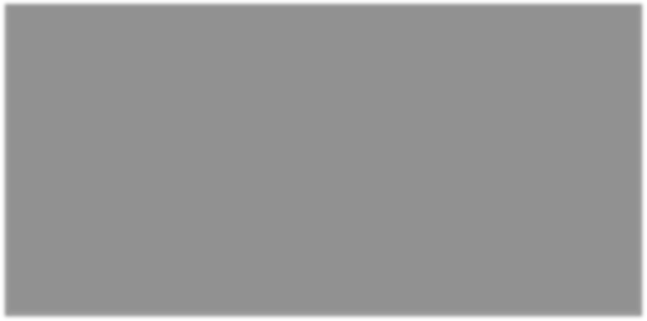
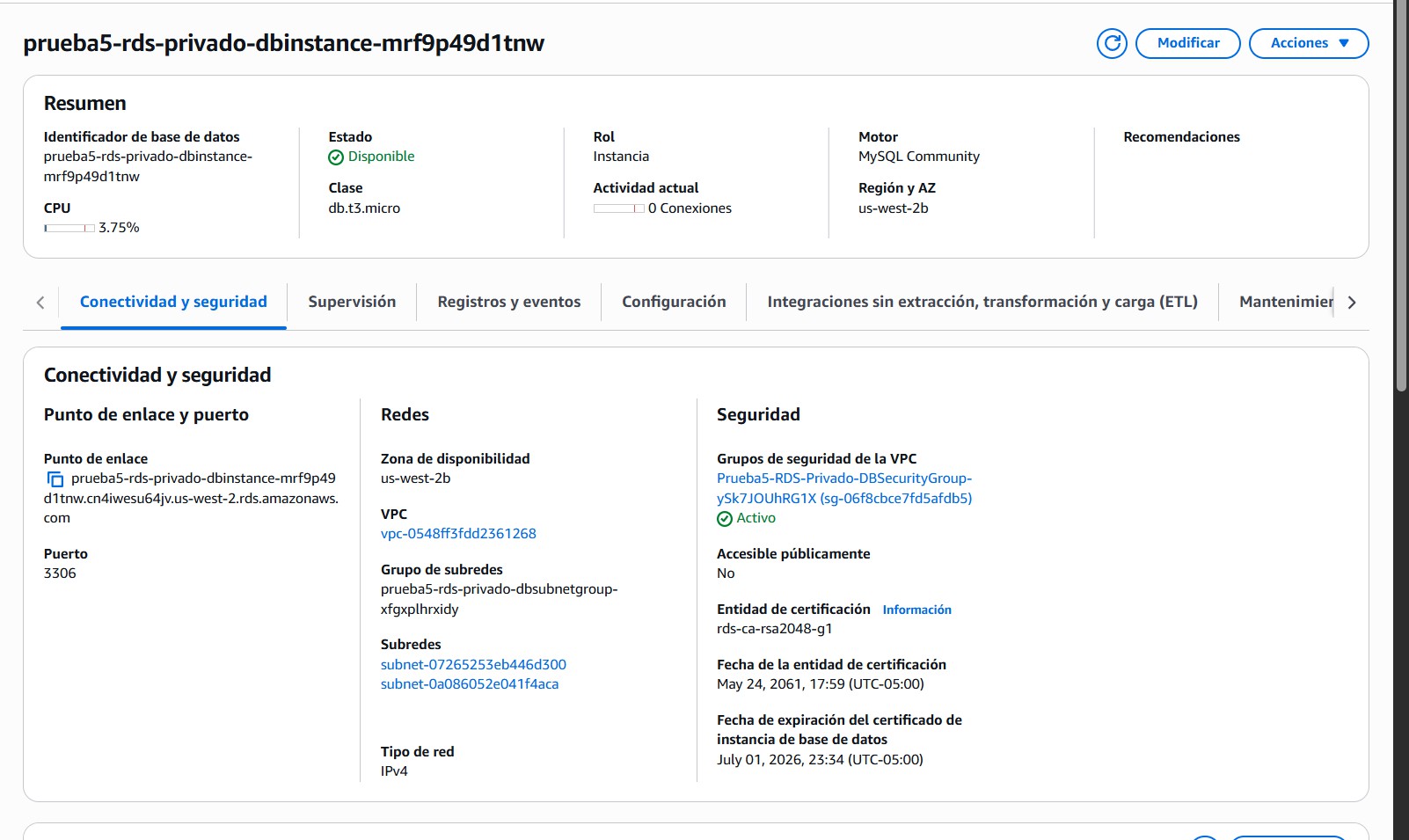
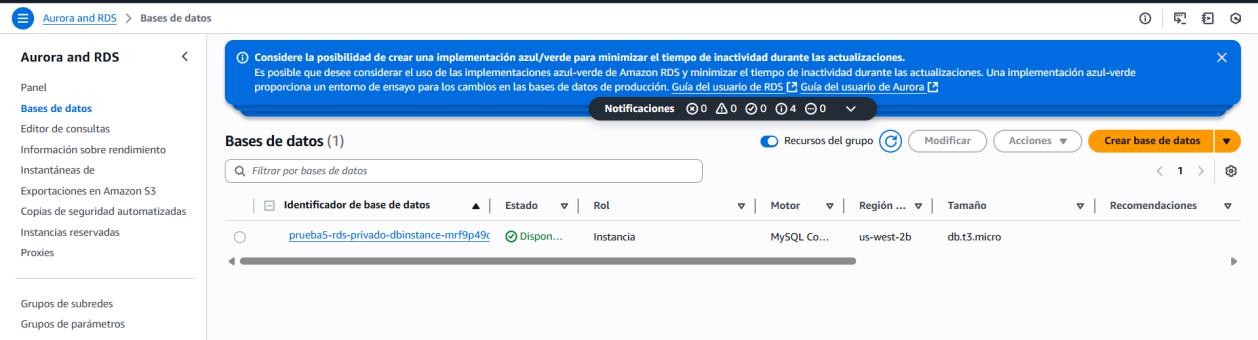
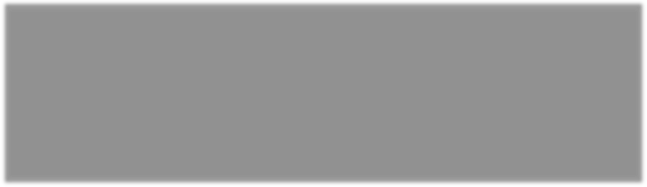
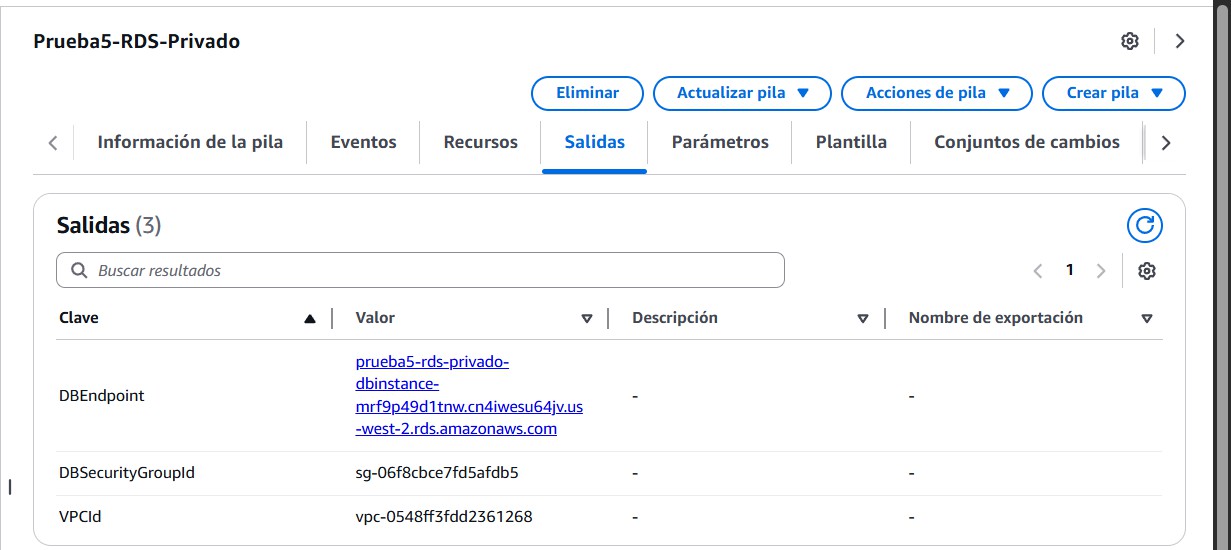
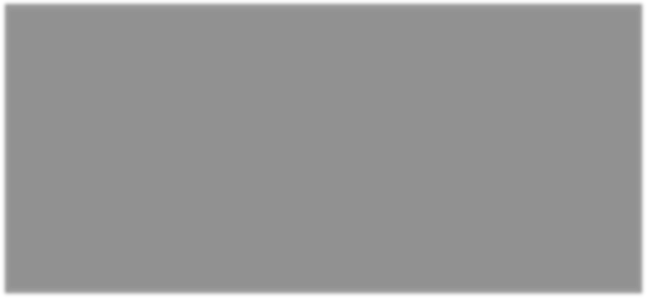
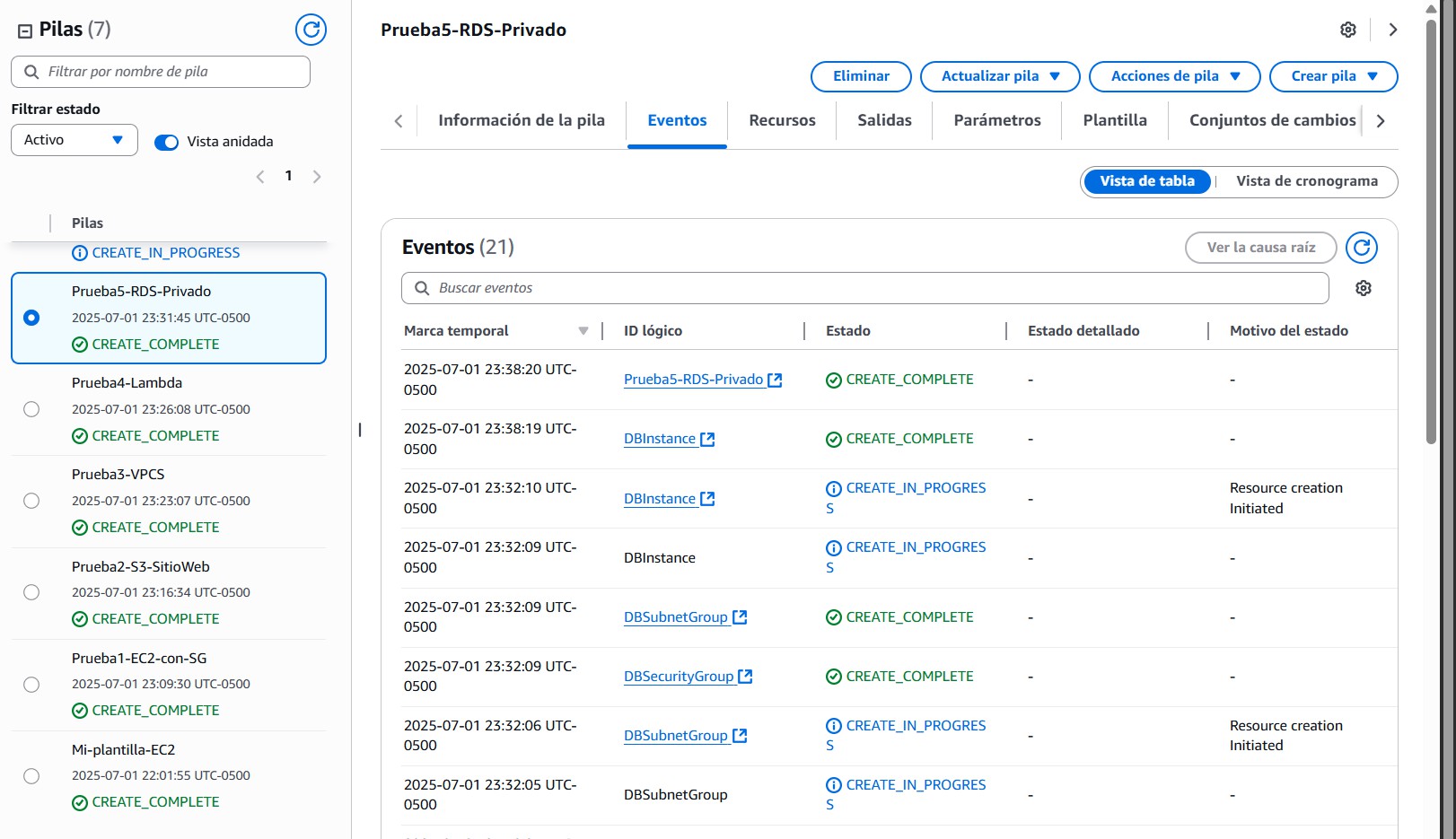
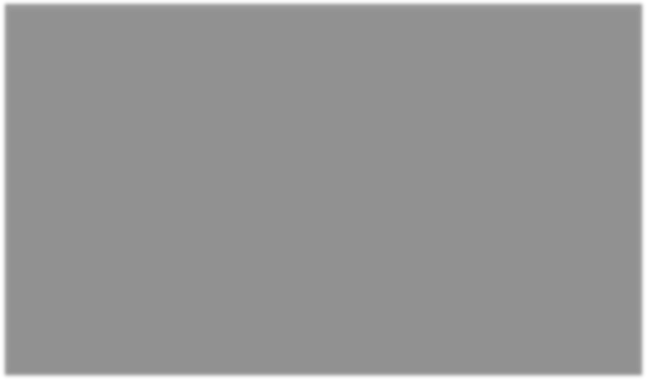
Esta fue la prueba que más tiempo tomó. Desplegué una instancia de base de datos MySQL en **Amazon RDS**. La plantilla fue muy completa, ya que primero creó una VPC nueva y subredes privadas para garantizar que la base de datos estuviera completamente aislada y segura. Verifiqué su creación en la consola de RDS, donde confirmé que el estado era "Disponible" y que la opción **"Públicamente accesible" estaba configurada en "No"**, cumpliendo el objetivo de la plantilla.



Ahora que creamos esta veremos si es funcional.

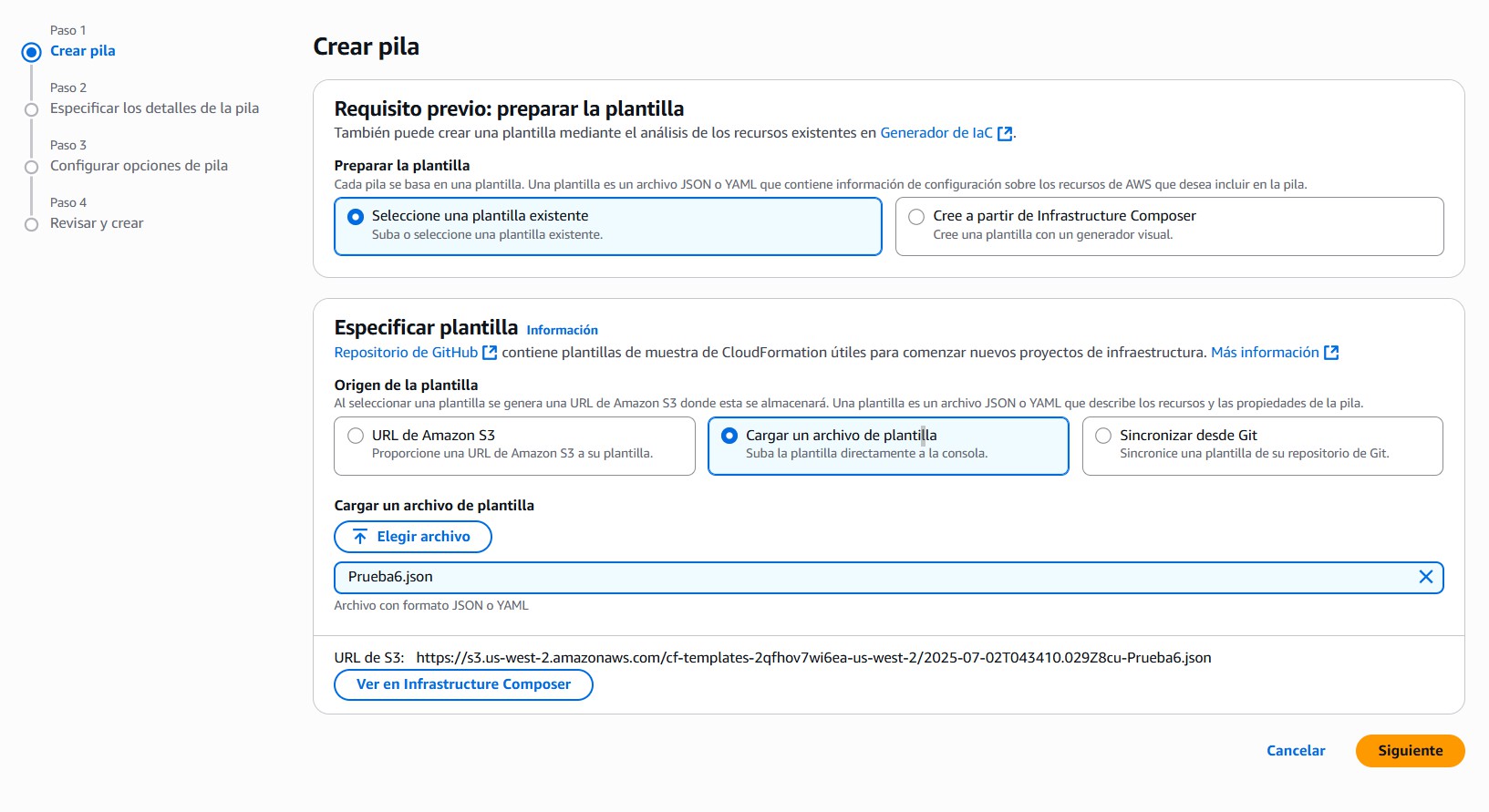
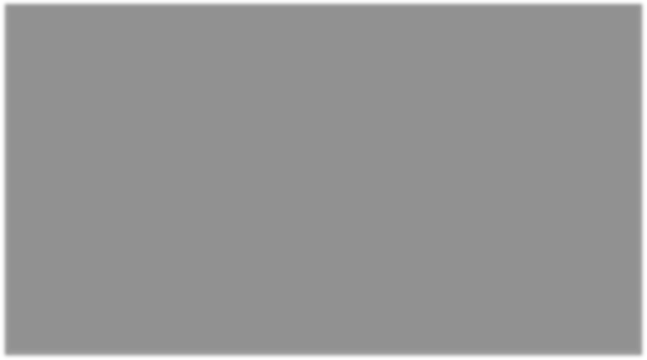
## Una Base de Datos Privada en RDS

* **a. Análisis de la Estructura (JSON)** Esta plantilla es compleja y primero crea una red (VPC y subredes privadas) y luego, dentro de ella, los recursos de RDS. Lo más relevante fue ver cómo se usa la propiedad **PubliclyAccessible: false** y se asocia la instancia a un grupo de seguridad y un grupo de subredes para garantizar el máximo aislamiento y seguridad.
* **b. Análisis del Diseñador (Designer)** El diagrama muestra que la **DBInstance está contenida dentro de las PrivateSubnets**, las cuales a su vez están dentro de la VPC. También se ven claramente las dependencias de la instancia con su DBSecurityGroup y DBSubnetGroup.
* **c. Propósito y Caso de Uso** Su propósito es **desplegar una base de datos relacional de manera segura y aislada** de la red pública. Resuelve el requerimiento crítico de tener una base de datos para producción que almacene datos sensibles y no deba ser expuesta a Internet. Es la práctica recomendada para cualquier base de datos en entornos productivos.



# Prueba 6: Arquitectura Web Escalable con Balanceador de Carga y Auto Scaling

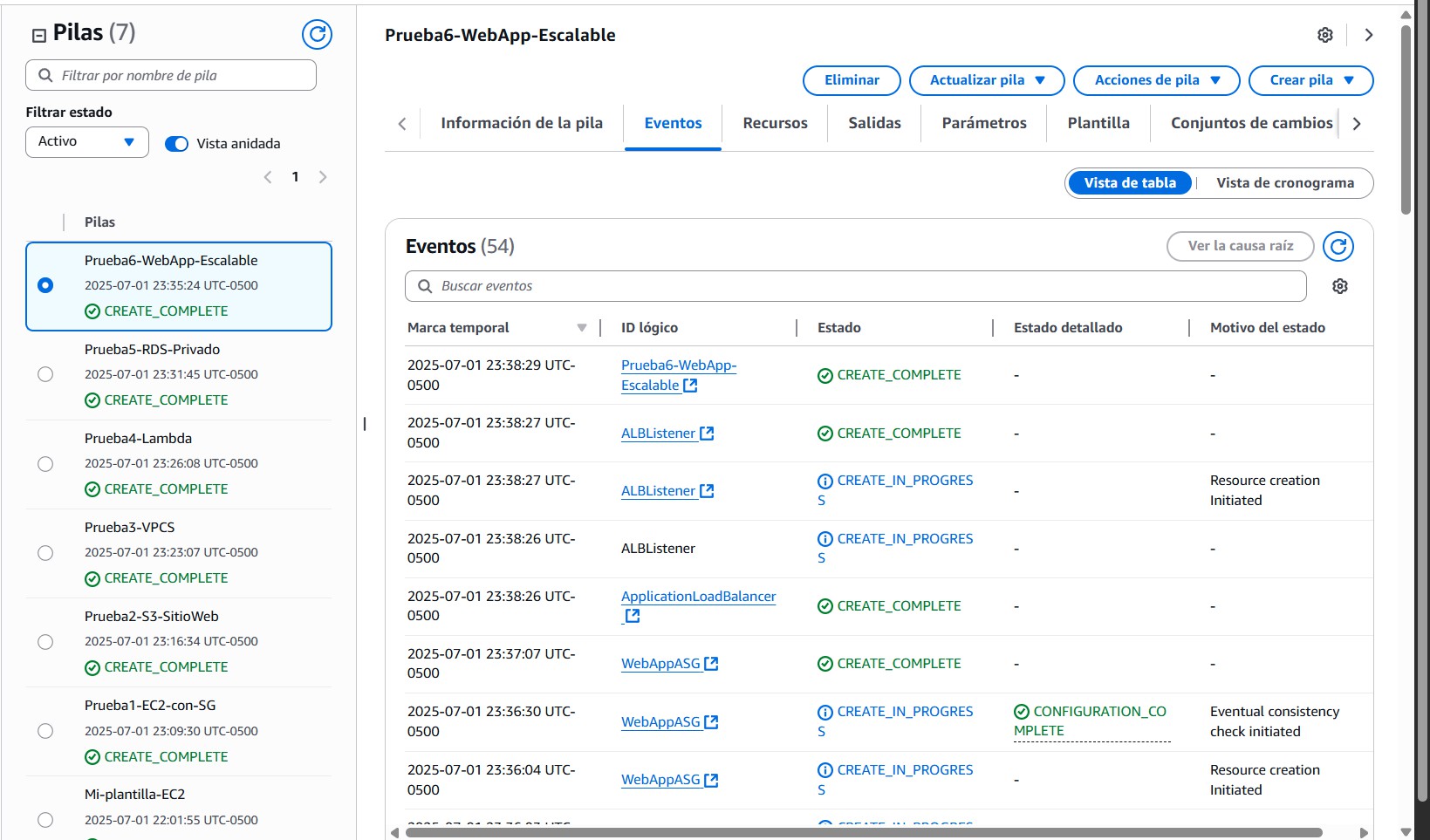
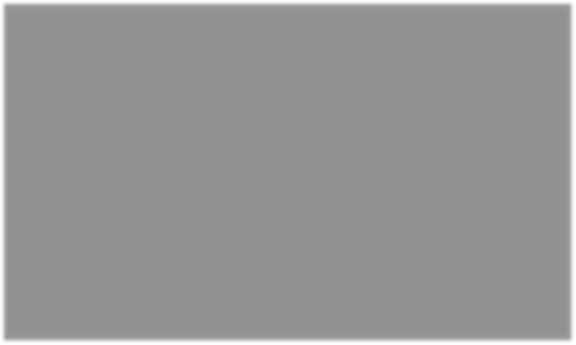
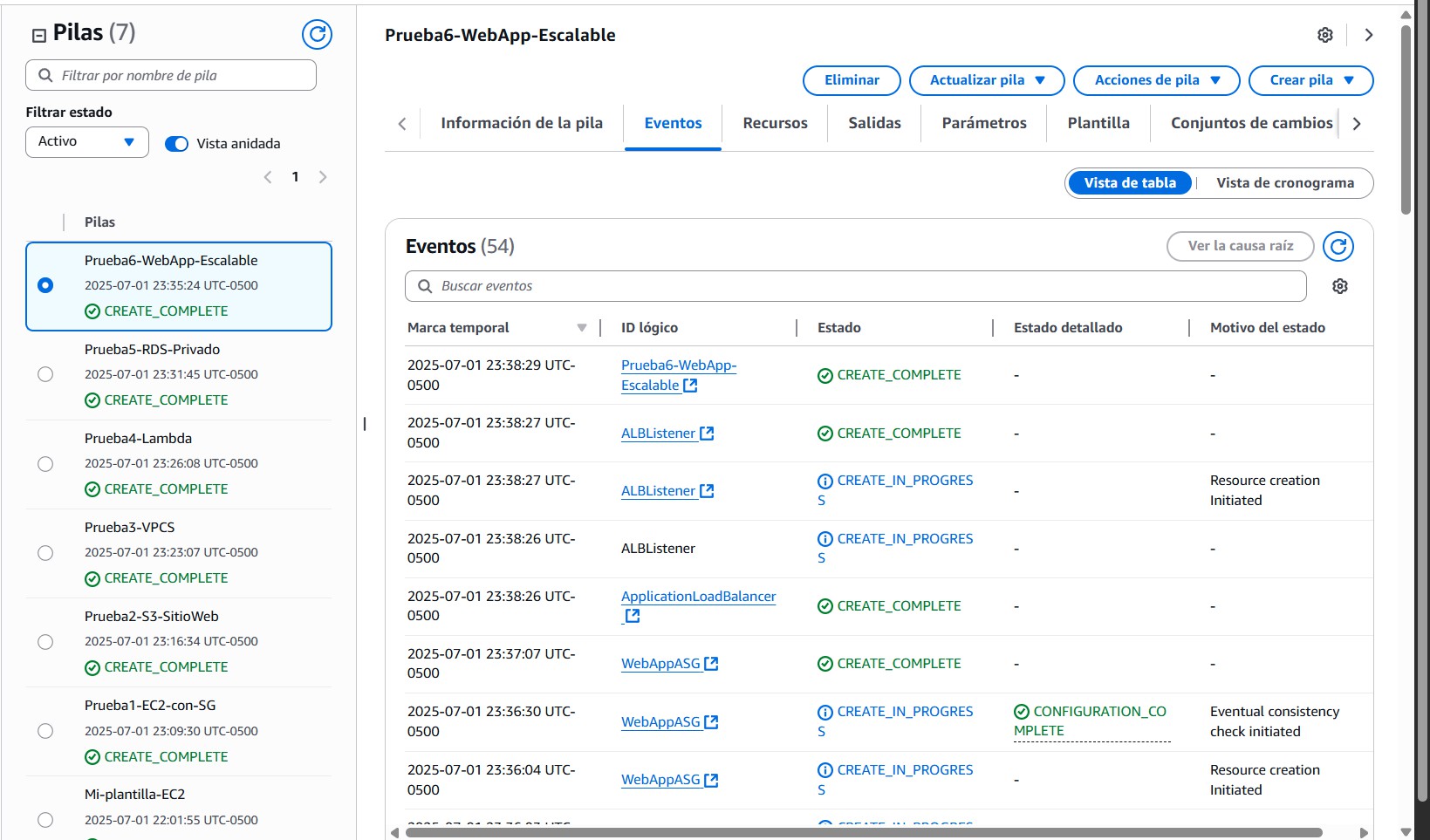
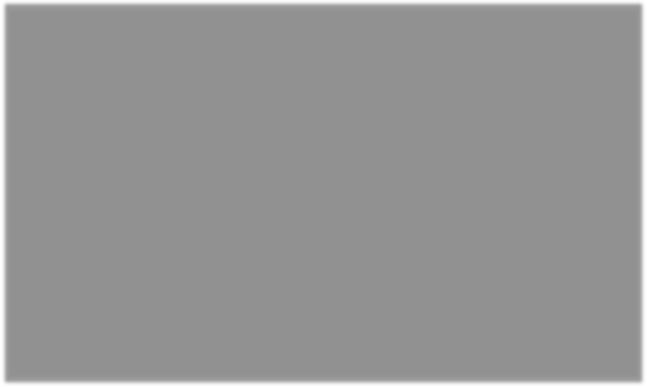
En la prueba final, implementé una arquitectura web completa y de alta disponibilidad. La plantilla automatizó la creación de una VPC, un **Application Load Balancer (ALB)**, un Launch Template que instalaba un servidor web, y un **Grupo de Auto Scaling** para gestionar las instancias. La verificación fue la más satisfactoria: accedí al DNS público del balanceador de carga y vi la página "Hola Mundo", demostrando que todo el sistema funcionaba en conjunto perfectamente.



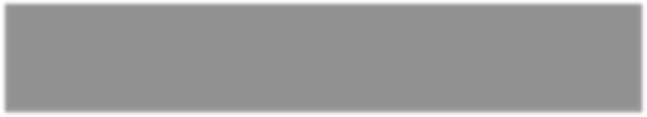
Ahora vemos si esto funciona correctamente.

## Arquitectura Web Escalable con Balanceador de Carga y Auto Scaling

* **a. Análisis de la Estructura (JSON)** Es la plantilla más compleja, con una clara cadena de dependencias: una VPC como base, un Launch Template para definir las instancias (incluyendo un script UserData para instalar Apache), un AutoScalingGroup para gestionar el número de instancias, y un LoadBalancer para distribuir el tráfico.
* **b. Análisis del Diseñador (Designer)** El diagrama muestra la interconexión de todos los recursos en una cadena lógica: el tráfico llega al **ALB**, que lo envía al **Target Group**, que a su vez es el destino del **Auto Scaling Group**, el cual utiliza el **Launch Template** para crear las instancias EC2. Todo esto reside dentro de una VPC.
* **c. Propósito y Caso de Uso** Su propósito es **construir una aplicación web con alta disponibilidad y escalabilidad automática**. Soluciona la necesidad de un equipo de desplegar una aplicación en producción que debe soportar picos de tráfico sin intervención manual y mantenerse disponible incluso si una de las instancias falla. Es la arquitectura estándar para aplicaciones web de producción.



## Conclusiones:



Indicar las conclusiones a las que llegó después de los temas tratados de manera práctica en este laboratorio.

1.- Se demostró la capacidad de CloudFormation para automatizar el despliegue de diversos recursos de AWS, desde una simple instancia EC2 y grupos de seguridad hasta una arquitectura web escalable completa con balanceador de carga y auto scaling.

2.- Las plantillas JSON utilizadas en el laboratorio estructuran los recursos y sus interconexiones, lo que nos permite mayor comprensión y la reutilización de las configuraciones de infraestructura.

3.- Las diferentes pruebas mostraron cómo plantillas específicas pueden solucionar requerimientos como el despliegue rápido de entornos de prueba (EC2 básica) , el alojamiento de frontends estáticos (S3) , el establecimiento de redes seguras (VPC) , la creación de microservicios sin servidor (Lambda) , y la implementación de bases de datos seguras y escalables (RDS).

4.- La representación gráfica de las plantillas en el diseñador ayuda a comprender las relaciones y dependencias entre los diferentes recursos de AWS, facilitando así el diseño y la depuración de arquitecturas.

5.- A pesar de que la conexión SSH a la instancia EC2 tuvo problemas debido a permisos , el laboratorio resalta la importancia de configurar correctamente los pares de claves y los roles de IAM para permitir el acceso y la ejecución de los recursos de manera segura.