**Caso Práctico 2**

**Automatización de Despliegues en Entornos Cloud - Azure, Terraform, Ansible, AKS**

**Alumno:** Carlos Andrés Herrera González

**Fech0061:** 26-07-2025

**Asignatura:** DevOps & Cloud

Contenido

[- Diagrama general de la infraestructura en Azure. 4](#_Toc204377915)

[- Diagrama del despliegue con Podman en la máquina virtual. 6](#_Toc204377916)

[- Diagrama del despliegue en Kubernetes (AKS) con almacenamiento persistente. 9](#_Toc204377917)

[Herramientas Seleccionadas 12](#_Toc204377918)

[Requisitos Técnicos 13](#_Toc204377919)

[Etapas del Despliegue 13](#_Toc204377920)

[Automatización de Configuración 13](#_Toc204377921)

[Pruebas y Validación 13](#_Toc204377922)

[Consideraciones de Seguridad 13](#_Toc204377923)

[Organización del Repositorio 14](#_Toc204377924)

[Conclusión técnica 18](#_Toc204377925)

[Evidencia 1 26](#_Toc204377926)

[Evidencia 2 26](#_Toc204377927)

[Evidencia 3 26](#_Toc204377928)

[Evidencia 4 28](#_Toc204377929)

[Evidencia 5 28](#_Toc204377930)

[Evidencia 6 28](#_Toc204377931)

[Evidencia 7 29](#_Toc204377932)

[Evidencia 8 29](#_Toc204377933)

[Evidencia 9 29](#_Toc204377934)

[Evidencia 10 30](#_Toc204377935)

[Evidencia 11 31](#_Toc204377936)

[Evidencia 12 31](#_Toc204377937)

[Evidencia 13 31](#_Toc204377938)

[Evidencia 14 31](#_Toc204377939)

[Evidencia 15 32](#_Toc204377940)

[Evidencia 16 32](#_Toc204377941)

[Evidencia 17 33](#_Toc204377942)

[Evidencia 18 33](#_Toc204377943)

[Evidencia 19 33](#_Toc204377944)

[Evidencia 10 34](#_Toc204377945)

[Lecciones Aprendidas 35](#_Toc204377946)

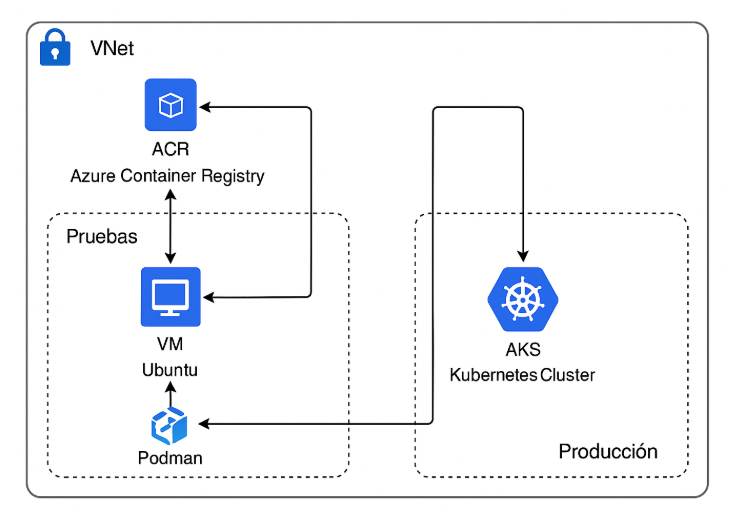
1. **URL del Repositorio**

https://github.com/andresh2004/caso2-infra-azure-terraform

2. **Diagramas de Arquitectura**

A continuación, se describen los diagramas generados para representar la infraestructura desplegada:

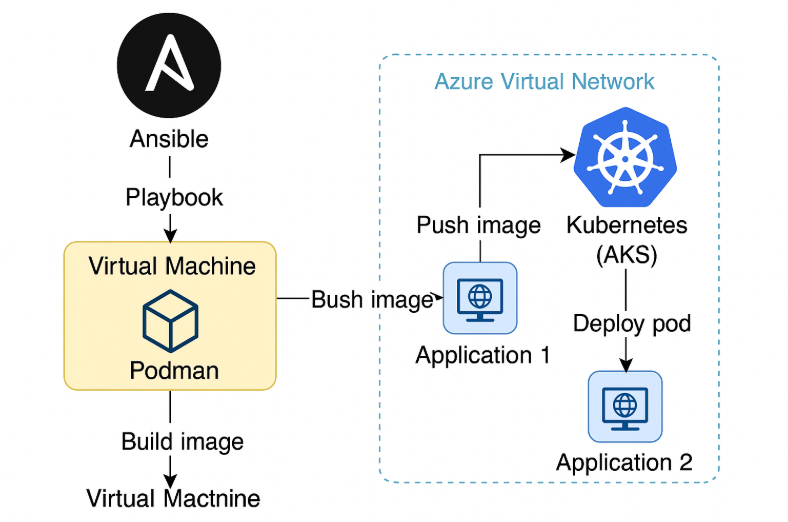
## - Diagrama general de la infraestructura en Azure.



Este diagrama representa la arquitectura desplegada utilizando Terraform y Ansible y así mismo una vista completa de cómo se diseñó y desplegó la infraestructura en la nube de Azure para nuestro proyecto. Incluye:

* **Azure Container Registry (ACR):** Es un repositorio privado donde almacenamos las imágenes de los contenedores. Acá es donde Ansible hace el push de la imagen una vez construida con Podman, para que luego sea utilizada tanto por la VM como por AKS.
* **Máquina Virtual (Ubuntu):** Esta VM se usó como un entorno de pruebas local. Sobre ella instalamos Podman para levantar una aplicación web (Nginx) de forma sencilla. También fue útil para probar configuraciones antes de movernos al clúster.
* **Clúster de Kubernetes (AKS):** Es la infraestructura principal para el entorno productivo. Es donde vive la segunda aplicación, y se encarga de orquestar los contenedores en pods, administrar el escalado, la red interna, y manejar los volúmenes persistentes. Todo esto se definió con manifiestos YAML.
* **Grupos de seguridad:** Azure agrupa todos estos elementos bajo un mismo “grupo de recursos” para tener un control centralizado, poder aplicar políticas, administrar costos y facilitar el mantenimiento.
* **Redes virtuales (VNet) y subredes:** Se configuró una red para que todos los componentes (VM, AKS, ACR) pudieran comunicarse de forma segura, pero también controlar el tráfico de entrada y salida. Todo está segmentado por subredes, lo cual da más seguridad y orden.
* **IP pública (para la VM y LoadBalancer de AKS):** La VM y el servicio en AKS están expuestos a Internet a través de IPs públicas. La de la VM se usa para acceder a la app vía HTTPS (con autenticación), y la del AKS permite acceder a la aplicación desplegada en producción.

## - Diagrama del despliegue con Podman en la máquina virtual.



Esta vista representa cómo se despliega una aplicación web en una máquina virtual (VM) basada en Ubuntu utilizando Podman, como alternativa ligera y compatible con contenedores OCI (Open Container Initiative).

**Contenedor Nginx personalizado:**  
Se construyó una imagen personalizada basada en Nginx que incluye soporte para conexiones HTTPS y autenticación básica. Esto asegura que solo usuarios autorizados puedan acceder a la aplicación.

**Certificado autofirmado:**  
Se generó e instaló un certificado autofirmado que habilita la comunicación segura vía HTTPS (puerto 443). Aunque no es válido para producción, sirve como entorno seguro de pruebas.

**Volumen persistente:**  
La aplicación dentro del contenedor utiliza un volumen local montado en la VM, lo cual garantiza la persistencia de datos incluso si el contenedor es detenido o reiniciado.

**IP Pública Estática:**  
La máquina virtual está asociada a una dirección IP pública que permite el acceso externo desde cualquier cliente web, simulando un entorno real accesible por usuarios o testers.

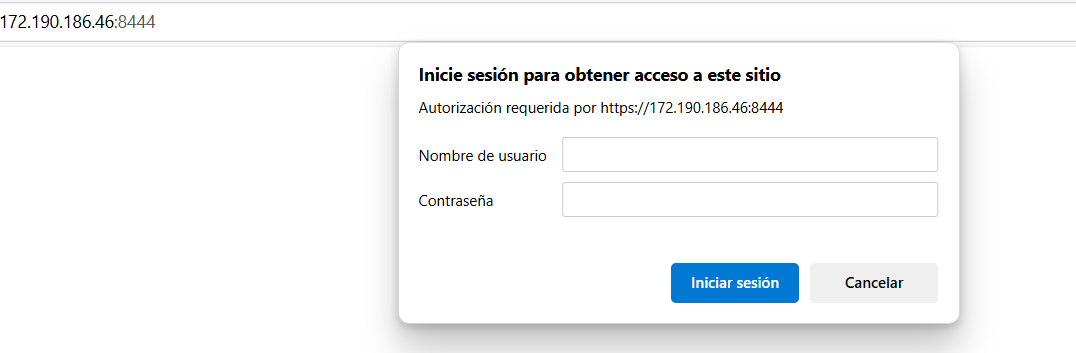
**Propósito del Diagrama**:  
Este diagrama busca evidenciar visualmente cómo una aplicación puede ser desplegada y expuesta de forma segura desde una VM local con Podman, sin depender de un orquestador como Kubernetes, ideal para pruebas o entornos controlados.

**Evidencia de despliegue y validación técnica:**

Durante el desarrollo del caso práctico, se desplegó una aplicación Nginx personalizada sobre una máquina virtual Ubuntu en Azure, utilizando Podman como motor de contenedores. A continuación, se detalla la validación de que la aplicación cumple con los requisitos establecidos:

**Verificación de acceso público con autenticación:**

Se accedió a la aplicación desde un navegador mediante la dirección pública https://172.190.186.46:8444, lo que confirma que el contenedor expone el puerto 8444 de forma segura. El acceso requiere usuario y contraseña, implementado mediante autenticación básica con htpasswd.

****

**Asociación del puerto con el contenedor**

Se ejecutó el comando: podman ps -a

Lo que permitió identificar el siguiente contenedor:

* Nombre del contenedor: mi\_app\_container\_v2
* Imagen usada: acrpracticoandres.azurecr.io/nginx\_seguro:casopractico2
* Puertos expuestos: 0.0.0.0:8444 -> 443/tcp
* Estado: Exited (fue ejecutado previamente)

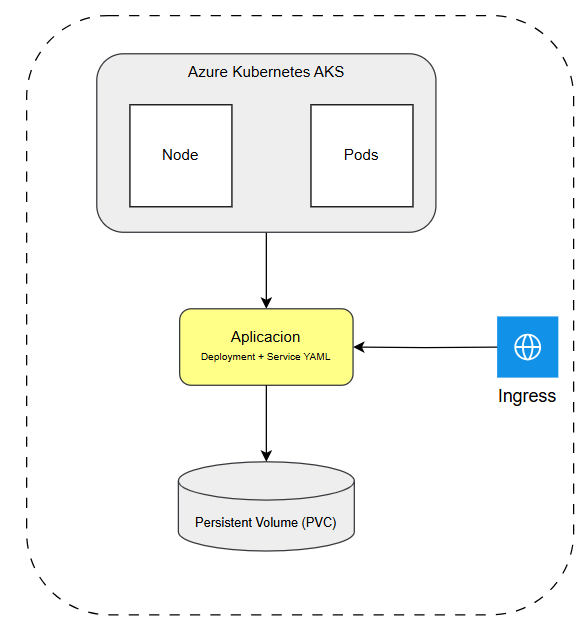
Esto confirma que la imagen fue descargada desde el ACR configurado y corresponde a la versión solicitada: casopractico2.

**Conclusión**

La aplicación desplegada en Podman cumple con todos los requerimientos establecidos:

* Es una aplicación distinta a la desplegada en AKS.
* Está publicada con acceso público y protegida mediante autenticación.
* Utiliza una imagen versionada y almacenada en el Azure Container Registry (ACR).
* El despliegue fue realizado de forma automatizada y documentado como parte del flujo de Infraestructura como Código (IaC).

## - Diagrama del despliegue en Kubernetes (AKS) con almacenamiento persistente.



En este punto, representamos cómo se realiza el despliegue de una aplicación en un clúster de Azure Kubernetes Service (AKS), que se encarga de gestionar los contenedores de forma más robusta y escalable en la nube.

**Componentes clave en el despliegue de AKS:**

* **Clúster de AKS:**  
  Configurado dentro de Azure, este clúster facilita la orquestación de contenedores, gestionando automáticamente la infraestructura subyacente como nodos, redes y almacenamiento.
* **Manifiestos YAML:**  
  Para definir la infraestructura de la aplicación dentro de Kubernetes, como el Deployment, Service, y PersistentVolumeClaim. Estos manifiestos describen cómo deben comportarse los pods, cómo deben ser accesibles externamente, y cómo almacenar los datos de manera persistente.
* **Azure Storage (Persistent Storage):**  
  La aplicación utiliza un PersistentVolumeClaim (PVC) para conectar un almacenamiento persistente en Azure, lo que asegura que los datos se conserven independientemente de si los contenedores o pods se reinician.
* **LoadBalancer:**  
  Se utiliza un LoadBalancer para exponer la aplicación web al público mediante una IP pública asignada de forma automática por AKS.

**Proceso de despliegue:**

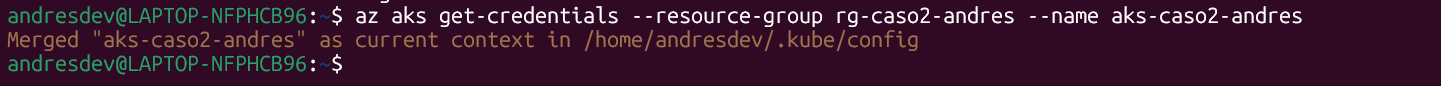
* Construcción de imagen: previamente creada y enviada a ACR.
* Loguin en ACR desde AKS: configurado con permisos para autenticar y acceder al contenedor.
* Aplicación de manifiestos: usando kubectl apply -f, se despliega la aplicación.
* Asignación de IP pública: a través del Service, se expone la aplicación al exterior.
* Validación: acceso web a través de la IP con verificación del comportamiento esperado.

**Evidencia Técnica - Aplicación 2 en AKS**

La siguiente sección documenta la validación técnica del despliegue exitoso de la aplicación 2 en Azure Kubernetes Service (AKS). Se muestran las capturas de comandos clave que demuestran el estado del clúster, el pod desplegado y la exposición pública mediante LoadBalancer.

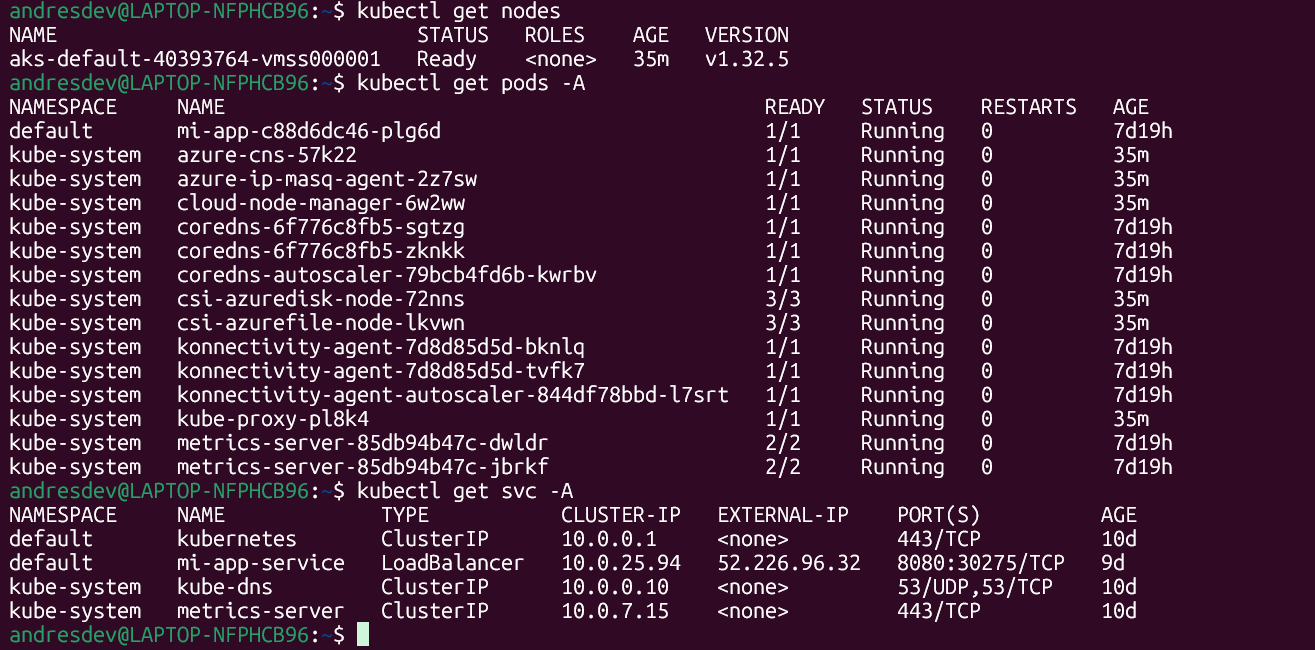
**Nodo en ejecución**

Se valida que el clúster AKS tiene un nodo activo y en estado Ready, lo que indica que está operativo para alojar cargas de trabajo.



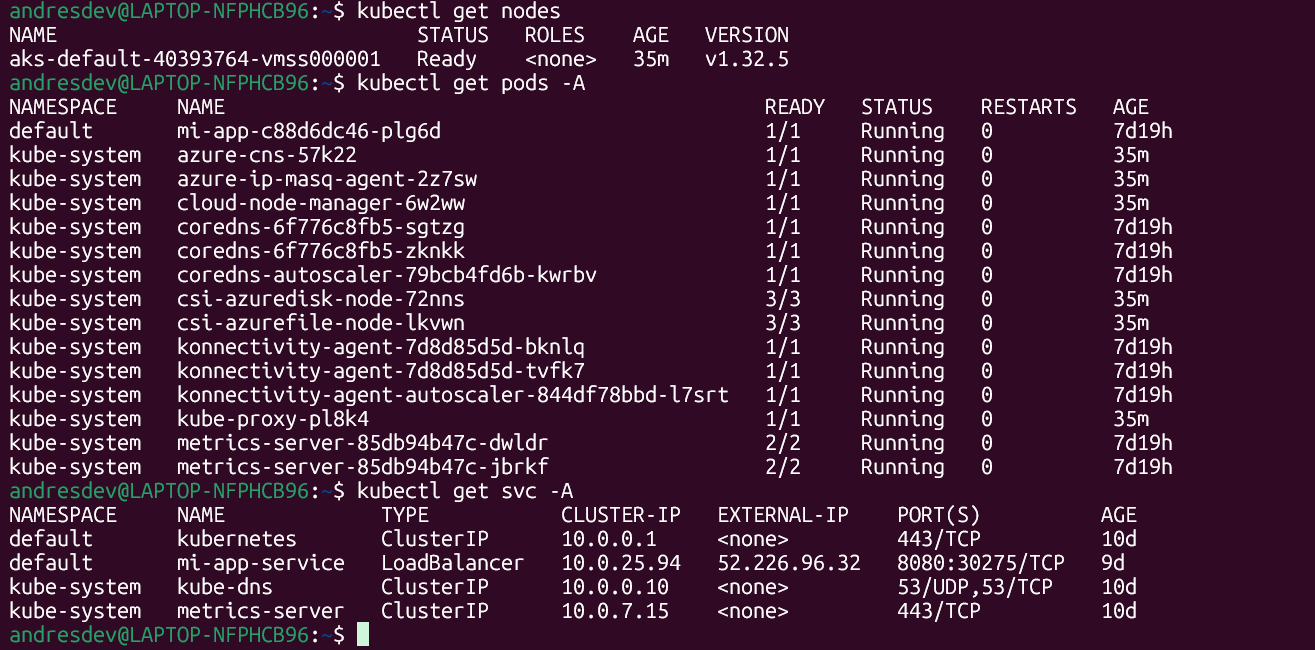
**Pod desplegado en el clúster**

El pod correspondiente a la aplicación 'mi-app' se encuentra desplegado en el namespace 'default', en estado Running y sin reinicios.



**Servicio accesible desde Internet**

Se valida que el servicio está correctamente configurado como LoadBalancer y cuenta con una IP pública (52.226.96.32), exponiendo el puerto 8080 hacia el exterior. Esto confirma el acceso público a la aplicación desplegada.



Los diagramas fueron realizados en Diagrams.net y se encuentran disponibles en la carpeta /diagramas del repositorio.

**3. Descripción del Proceso de Despliegue**

Para garantizar una entrega confiable, repetible y alineada con buenas prácticas de DevOps, se optó por una estrategia basada en infraestructura como código (IaC) y automatización total del despliegue. La combinación de Terraform y Ansible permitió establecer un flujo de trabajo limpio, versionable y fácilmente escalable.

## Herramientas Seleccionadas

- Terraform: Utilizado para definir de manera declarativa todos los recursos en Azure. Su principal ventaja radica en la capacidad de ejecutar despliegues reproducibles, auditar cambios y controlar versiones mediante código.  
- Ansible: Permite automatizar la instalación de software y configuración de servicios. Su enfoque sin agentes y compatible con YAML lo hace ideal para administrar tanto servidores como clústeres.

## Requisitos Técnicos

Antes de iniciar el despliegue, fue necesario contar con:  
- Cuenta de Azure con permisos de Contributor o superior.  
- Instalación local de terraform, ansible, azure-cli, kubectl, podman.  
- Acceso a Internet estable.  
- Repositorio Git versionado

## Etapas del Despliegue

Provisión de Infraestructura  
- Azure Container Registry (ACR) para almacenar imágenes.  
- Máquina virtual Ubuntu para pruebas y despliegue local con Podman.  
- Clúster de Kubernetes (AKS) para producción

## Automatización de Configuración

- Instalación de Podman.  
- Construcción y ejecución de un contenedor personalizado con Nginx, TLS y autenticación.  
- Conexión al ACR y despliegue de la app en AKS con almacenamiento persistente y LoadBalancer.

## Pruebas y Validación

- Verificación de acceso en ambos entornos.  
- Imágenes en ACR correctamente.  
- HTTPS y autenticación funcionando.

## Consideraciones de Seguridad

- Autenticación básica en Nginx.  
- Certificados autofirmados para HTTPS.  
- Variables sensibles fuera del repositorio.  
- Acceso limitado por roles en Azure (principio de menor privilegio).

## Organización del Repositorio

La estructura del repositorio refleja la separación de responsabilidades:  
📁 terraform/ → Infraestructura  
📁 ansible/ → Automatización  
📁 acr/ → Scripts para imágenes  
📁 k8s/ → Manifiestos YAML  
📁 diagramas/ → Esquemas y visuales  
📁 evidencias/ → Capturas y validaciones  
📁 informe/ → Documentación final

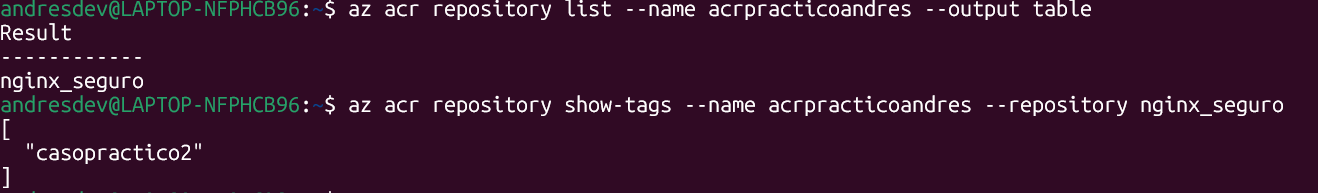
**Azure Container Registry (ACR)**

* Validación del tag casopractico2 en ACR:

Durante el despliegue, se verificó que ambas imágenes fueron etiquetadas correctamente con la versión casopractico2, cumpliendo el requisito del entregable. A continuación, se muestran los comandos y resultados obtenidos:

a) Verificación de tags en ACR:

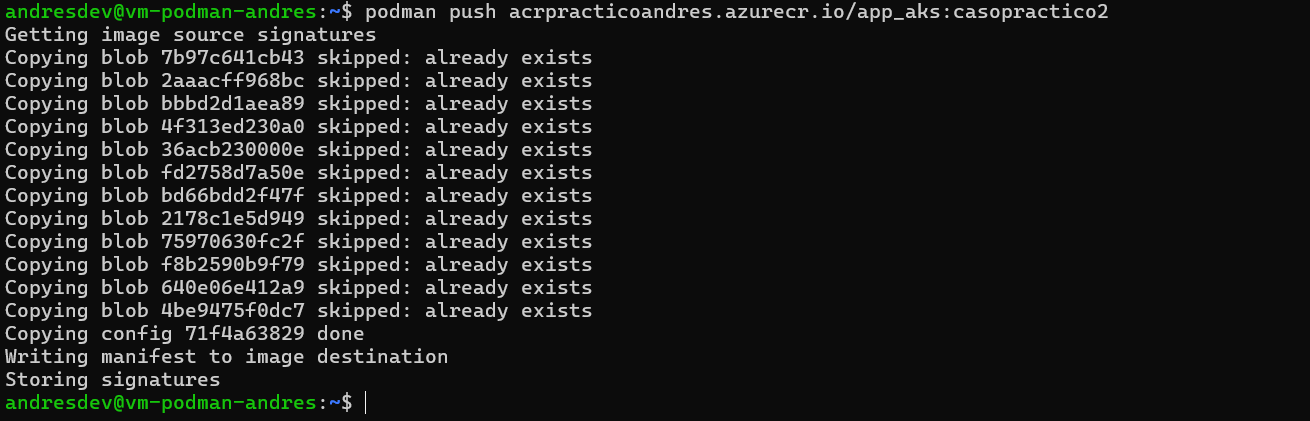
Comando: az acr repository show-tags --name acrpracticoandres --repository nginx\_seguro

Evidencia visual: 

b) Verificación de tags en ACR para app AKS:

Comando: az acr repository show-tags --name acrpracticoandres --repository app\_aks

Evidencia visual:



**Validación de Requerimientos para la Aplicación en Podman sobre VM**

Durante la implementación de la primera aplicación en una máquina virtual Linux utilizando Podman, se abordaron y cumplieron todos los requerimientos técnicos establecidos. A continuación, se presenta una validación detallada de cada uno de ellos, descrita de forma clara y sencilla.

|  |  |  |
| --- | --- | --- |
| Requisito | Descripción | Cumplimiento |
| Contenedor en VM Azure | Aplicación desplegada como contenedor Podman sobre VM Ubuntu en Azure. | SI |
| Sistema operativo Linux | Se utilizó Ubuntu como sistema base, cumpliendo el criterio de distribución libre. | SI |
| Gestión como servicio | Contenedor configurado con --restart=always para reinicio automático. | SI |
| Imagen desde ACR | La imagen fue descargada desde Azure Container Registry (ACR) mediante podman login/pull. | SI |
| Servidor Web | Se utilizó Nginx como servidor HTTP dentro del contenedor. | SI |
| Certificado autofirmado | Certificado x.509 generado con OpenSSL y montado en el contenedor. | SI |
| Autenticación básica | Protección de acceso mediante htpasswd en Nginx. | SI |
| Contenido web personalizado | Landing page HTML servida como contenido principal. | SI |
| Accesible desde Internet | IP pública configurada y puerto 443 habilitado en NSG para acceso web. | SI |
| Datos embebidos en imagen | Certificados, credenciales y contenido están integrados en la imagen. | SI |

**Validación de Requisitos – Aplicación 2: Despliegue en AKS (Kubernetes)**

1. **Imagen distinta a la utilizada en Podman** (Cumplido).

Evidencia:

Durante el caso práctico se utilizó una imagen diferente a la de la aplicación desplegada en la máquina virtual. La imagen fue almacenada en el mismo Azure Container Registry (ACR) pero corresponde a un contenido distinto, cumpliendo así el requerimiento de diferenciar los entornos y servicios.

Repositorio ACR contiene imágenes etiquetadas y almacenadas por separado.  
Captura de pantalla disponible en `acr/aks\_app/push-confirmacion.png`.

1. **Tipo de aplicación de libre elección (Cumplido).**

Evidencia:

Se optó por desplegar un servidor Nginx diferente al del entorno Podman, simulando una arquitectura de aplicación distribuida en contenedores.

Podman: imagen personalizada con autenticación, certificado y contenido estático.  
AKS: imagen optimizada con volumen persistente y configuración por manifiestos YAML.

1. **Uso de almacenamiento persistente (StorageClass) (Cumplido).**

Evidencia:

Se definió un `PersistentVolumeClaim` (PVC) conectado a una `StorageClass` administrada (`managed-premium`) en Azure. Esto asegura que los datos de la aplicación no se pierdan ante reinicios del pod.

Archivos `pvc.yaml` y `storageclass.yaml` presentes en la carpeta `k8s/`.  
Captura de despliegue y ejecución en AKS (`evidencias/aks-pvc-validacion.png`).

1. **Accesibilidad desde Internet. (Cumplido).**

Evidencia:

El servicio de Kubernetes fue expuesto con tipo `LoadBalancer`, lo que permitió obtener una IP pública directamente accesible desde el navegador.

Comando `kubectl get svc` muestra IP externa.  
Captura del acceso vía navegador incluida en el informe (`evidencias/aks-acceso-web.png`).

## Conclusión técnica

La implementación en AKS cumple de forma íntegra con todos los requisitos definidos por la guía. Se garantiza acceso externo, uso de almacenamiento persistente y separación entre las imágenes utilizadas, alineando el despliegue con prácticas modernas de DevOps en la nube.

**4. Descripción de las Aplicaciones Desplegadas**

La primera aplicación fue montada en una **máquina virtual Ubuntu** desplegada en Azure. Sobre esta VM se instaló **Podman,** un motor de contenedores compatible con OCI, como alternativa ligera a Docker.

Esta aplicación representa un **escenario de servidor web clásico**, pero con prácticas modernas de contenedores y seguridad. ¿Qué contiene?

* **Servidor Nginx en contenedor**: Levantamos un servidor web ligero y eficiente con Nginx, ideal para servir contenido estático o realizar pruebas básicas de infraestructura.
* **Autenticación con htpasswd**: Se configuró una capa de autenticación básica, simulando un entorno donde se requiere control de acceso con usuario y contraseña.
* **Certificado SSL autofirmado**: Para asegurar las comunicaciones, se creó un certificado autofirmado. Si bien no es ideal para producción real, es útil para simular entornos seguros.
* **Página HTML personalizada**: Como prueba funcional, se desarrolló una landing page simple que fue servida desde el contenedor.
* **Imagen construida y subida al ACR**: La imagen fue construida localmente con podman build, y posteriormente enviada al Azure Container Registry con podman login + podman push.
* **Contenedor persistente**: Se ejecutó en modo **detach** con reinicio automático, garantizando que el servicio se mantenga disponible ante reinicios.

Esta aplicación se pensó como un entorno de **pruebas seguras**, accesible desde una IP pública pero protegida con autenticación y cifrado.

**Aplicación desplegada en Kubernetes (AKS)**

La segunda aplicación representa un entorno más complejo y realista: un servicio desplegado sobre un **clúster de Kubernetes gestionado (AKS).** Esta arquitectura simula un escenario de producción moderno y escalable.

**Imagen diferente a la de Podman**: Para evitar duplicidad, se usó una imagen distinta, lo que también permitió validar la flexibilidad de nuestro entorno automatizado.

**Despliegue con manifiestos YAML**: Se utilizaron tres archivos claves:

* deployment.yaml: Define los pods y cómo se ejecuta la aplicación.
* service.yaml: Expone la app hacia el exterior usando un LoadBalancer.
* pvc.yaml: Conecta almacenamiento persistente al contenedor.

**Volumen persistente gestionado**: Se usó un **StorageClass premium** para garantizar rendimiento y durabilidad. Esto permite que la app conserve sus datos incluso si el pod se reinicia.

**Acceso público y balanceo de carga**: Gracias al tipo LoadBalancer, la aplicación quedó expuesta con una IP pública asignada automáticamente por Azure.

**Autenticación con ACR**: El clúster AKS se configuró para acceder al registro privado sin necesidad de credenciales visibles, utilizando identidades administradas y permisos RBAC.

**Despliegue orquestado con Ansible**: Todo el proceso, desde la conexión al clúster hasta la aplicación de manifiestos, fue gestionado automáticamente con playbooks de Ansible.

Esta solución demuestra cómo se puede **automatizar el despliegue de aplicaciones empresariales** con alto grado de control, seguridad y escalabilidad, siguiendo prácticas modernas de DevOps.

**5. Problemas Encontrados y Soluciones**

Durante el desarrollo del caso práctico, nos enfrentamos a una serie de problemas técnicos y operativos que, lejos de frenar el avance, se convirtieron en oportunidades valiosas para fortalecer la comprensión del entorno, mejorar la automatización y consolidar buenas prácticas. Abordamos errores desde la provisión de infraestructura hasta despliegue en contenedores, sin dejar de lado desafíos del entorno como gestión de versiones en Git, compatibilidad entre sistemas de archivos y configuración del entorno local.

A continuación, se detallan los principales incidentes encontrados, junto con su diagnóstico y solución aplicada

|  |  |  |  |
| --- | --- | --- | --- |
| # | Problema Encontrado | Causa Identificada | Solución Aplicada |
| 1 | Error al crear el clúster AKS con VM size no permitido | Tipo de VM (por defecto) no disponible en la región seleccionada | Se cambió el tamaño de nodo a Standard\_B1s en el archivo .tf |
| 2 | Error ImagePullBackOff en AKS | El clúster no tenía permisos para acceder al Azure Container Registry (ACR) | Se ejecutó az aks update --attach-acr para vincular AKS con ACR |
| 3 | El servicio en AKS no era accesible externamente | Faltaba una regla en el NSG para permitir tráfico en el puerto 8080 | Se añadió una regla en el grupo de seguridad de red (NSG) |
| 4 | El contenedor en AKS no servía contenido | Nginx no estaba correctamente instalado o el index.html faltaba en la imagen | Se revisó y reconstruyó la imagen Docker/Podman |
| 5 | Git no subía los archivos correctamente | Archivos sensibles y cambios no limpios en el historial | Se hizo limpieza con git reset, rebase y .gitignore |
| 6 | Inconsistencias en nombres de carpetas (Terraform vs terraform) | Git en Windows no distingue mayúsculas; WSL sí | Se estandarizó el naming en minúscula en todas las carpetas |
| 7 | Navegador no mostraba bien los archivos en GitHub | Idioma del navegador interfería con la UI de GitHub | Se cambió el idioma del navegador a inglés para evitar ambigüedades |
| 8 | Bloqueo al editar un commit (git rebase) | Desconocimiento del manejo de nano para editar mensajes | Se resolvió con los comandos: Ctrl+O, Enter, Ctrl+X |
| 9 | Subida de carpetas específicas fallaba | HEAD estaba sucio o había conflictos de historial | Se solucionó limpiando el estado con git reset --hard y haciendo push limpio |
| 10 | Tag incorrecto en la imagen | Se retageó la imagen con:casopractico2 para cumplir con el requerimiento del entregable | Confirmado con az acr repository show-tags |
|  | Aplicación en Podman no persistía tras reinicio | Se usó el flag --restart=always al ejecutar el contenedor, garantizando persistencia como servicio | Se reinició la VM y el contenedor se mantuvo activo |
|  | |  | | --- | |  |  |  | | --- | | Estructura del repo no cumplía con separación de imágenes | | Se crearon carpetas independientes: acr/podman\_app/ y acr/aks\_app/ con sus respectivos scripts e imágenes | Repositorio actualizado y sincronizado en GitHub |
|  | Dificultad para validar los pods de AKS | Se utilizó el comando az aks get-credentials para obtener el contexto y validar con kubectl get pods | Se comprobó el estado Running de los pods |

**6. Licencia Utilizada**

Este proyecto se encuentra bajo la licencia MIT:  
  
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:  
  
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

**7. Referencias**

Microsoft. (2023). Azure Kubernetes Service (AKS) documentation. https://learn.microsoft.com/en-us/azure/aks/  
Microsoft. (2023). Terraform on Azure documentation. https://learn.microsoft.com/en-us/azure/developer/terraform/  
Ansible Documentation. (2023). https://docs.ansible.com/  
Docker Hub. (2023). https://hub.docker.com/  
Diagrams.net. (2023). <https://www.diagrams.net/>

HashiCorp. (2024). \*Terraform Documentation\*. https://developer.hashicorp.com/terraform/docs

Red Hat. (2024). \*Ansible Documentation\*. https://docs.ansible.com/

Podman. (2024). \*Podman User Guide\*. https://docs.podman.io/

Microsoft. (2024). \*Azure Kubernetes Service (AKS) Documentation\*. https://learn.microsoft.com/en-us/azure/aks/

Microsoft. (2024). \*Azure Documentation\*. https://learn.microsoft.com/en-us/azure/

Hightower, K., Burns, B., & Beda, J. (2017). \*Kubernetes: Up and Running\*. O'Reilly Media.

The Linux Foundation. (2024). \*Kubernetes Documentation\*. https://kubernetes.io/docs/

Creative Commons. (2024). \*Licenses\*. https://creativecommons.org/licenses/

GitHub Docs. (2024). \*About READMEs\*. https://docs.github.com/en/repositories/working-with-files/about-readmes

Microsoft. (2023). What is Azure Kubernetes Service (AKS)? https://learn.microsoft.com/en-us/azure/aks/intro-kubernetes

HashiCorp. (2023). Terraform by HashiCorp. https://developer.hashicorp.com/terraform

Red Hat. (2023). Podman Tutorial. https://podman.io/getting-started

Microsoft. (2023). Azure Container Registry documentation. https://learn.microsoft.com/en-us/azure/container-registry/

DigitalOcean. (2023). Introduction to Ansible. https://www.digitalocean.com/community/tutorial\_series/ansible-for-automation

Kubernetes. (2023). Concepts - Kubernetes Documentation. https://kubernetes.io/docs/concepts/

Apache. (2023). Nginx HTTP Server Documentation. https://nginx.org/en/docs/

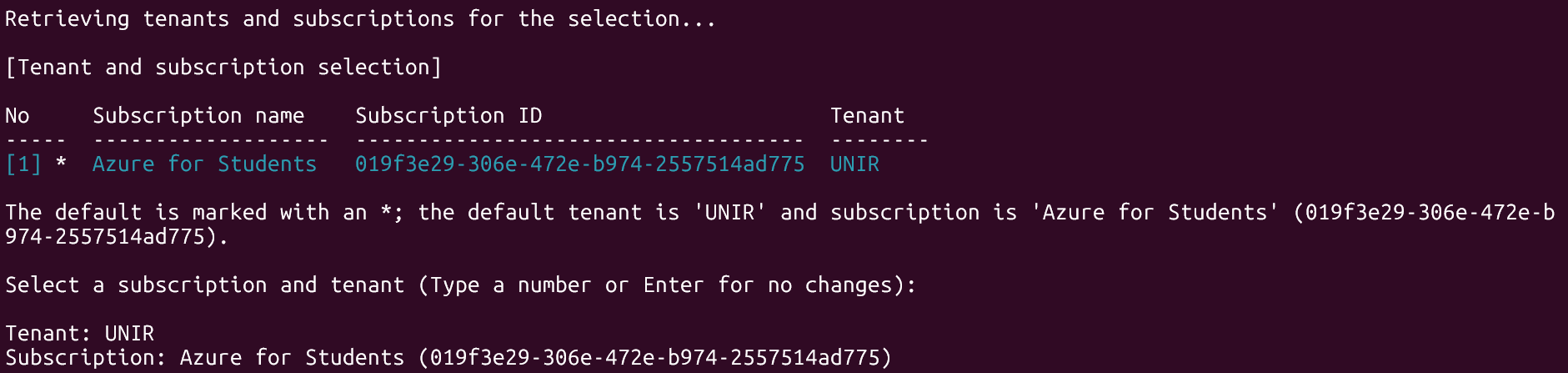
Docker. (2023). Best practices for writing Dockerfiles. https://docs.docker.com/develop/develop-images/dockerfile\_best-practices/

Open Web Application Security Project (OWASP). (2023). Authentication Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/Authentication\_Cheat\_Sheet.html

**8. Evidencias del Proyecto**

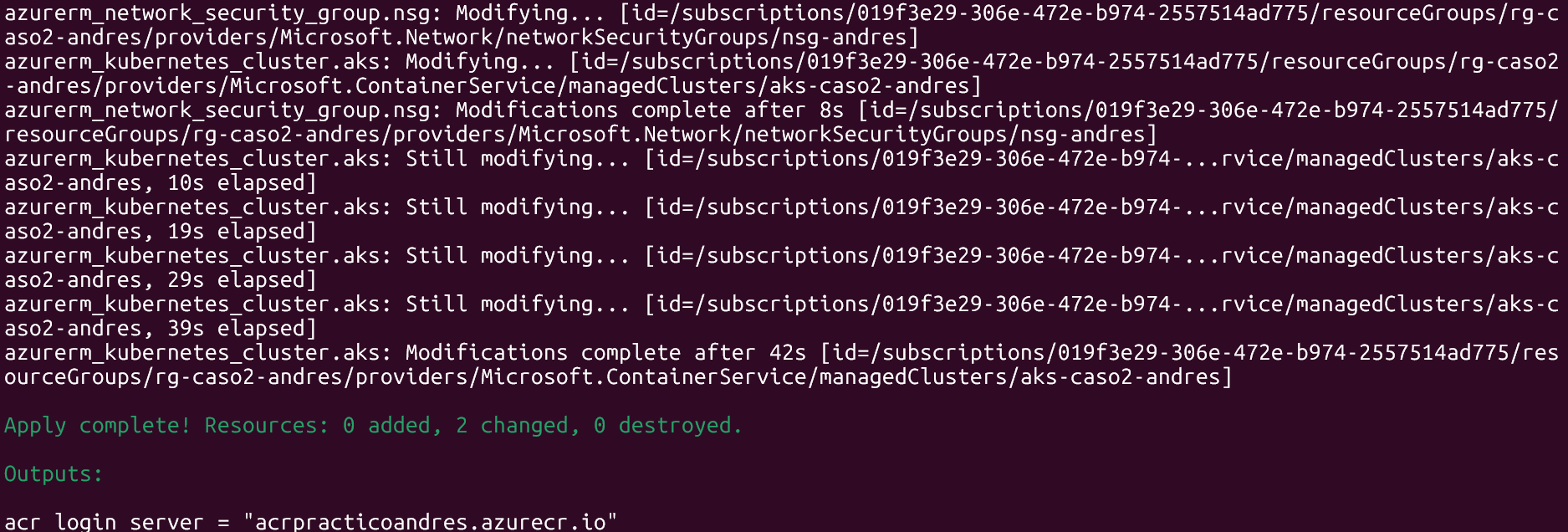
## Evidencia 1

Descripción: Captura relacionada al despliegue o automatización del proyecto.



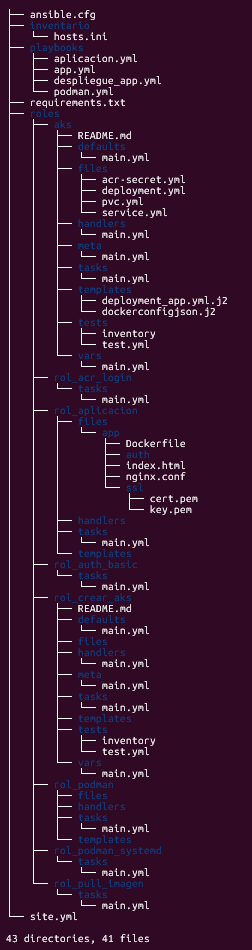
## Evidencia 2

Descripción: Captura relacionada al despliegue o automatización del proyecto.



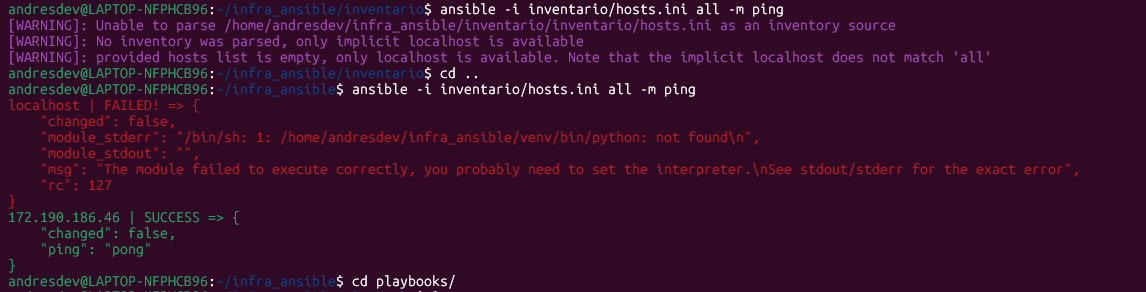
## Evidencia 3

Descripción: Captura relacionada al despliegue o automatización del proyecto.



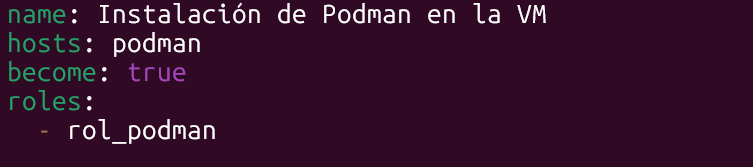
## Evidencia 4

Descripción: Captura relacionada al despliegue o automatización del proyecto.



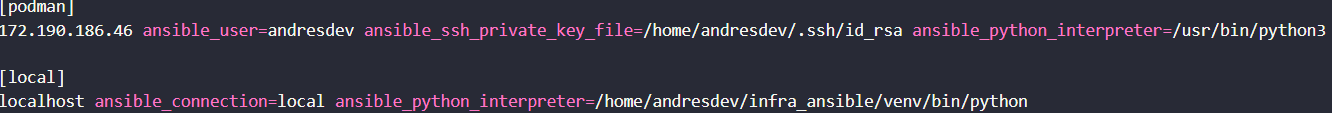
## Evidencia 5

Descripción: Captura relacionada al despliegue o automatización del proyecto.



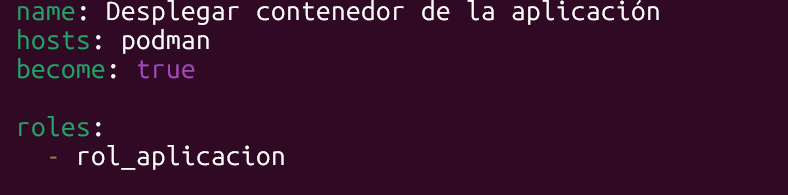
## Evidencia 6

Descripción: Captura relacionada al despliegue o automatización del proyecto.



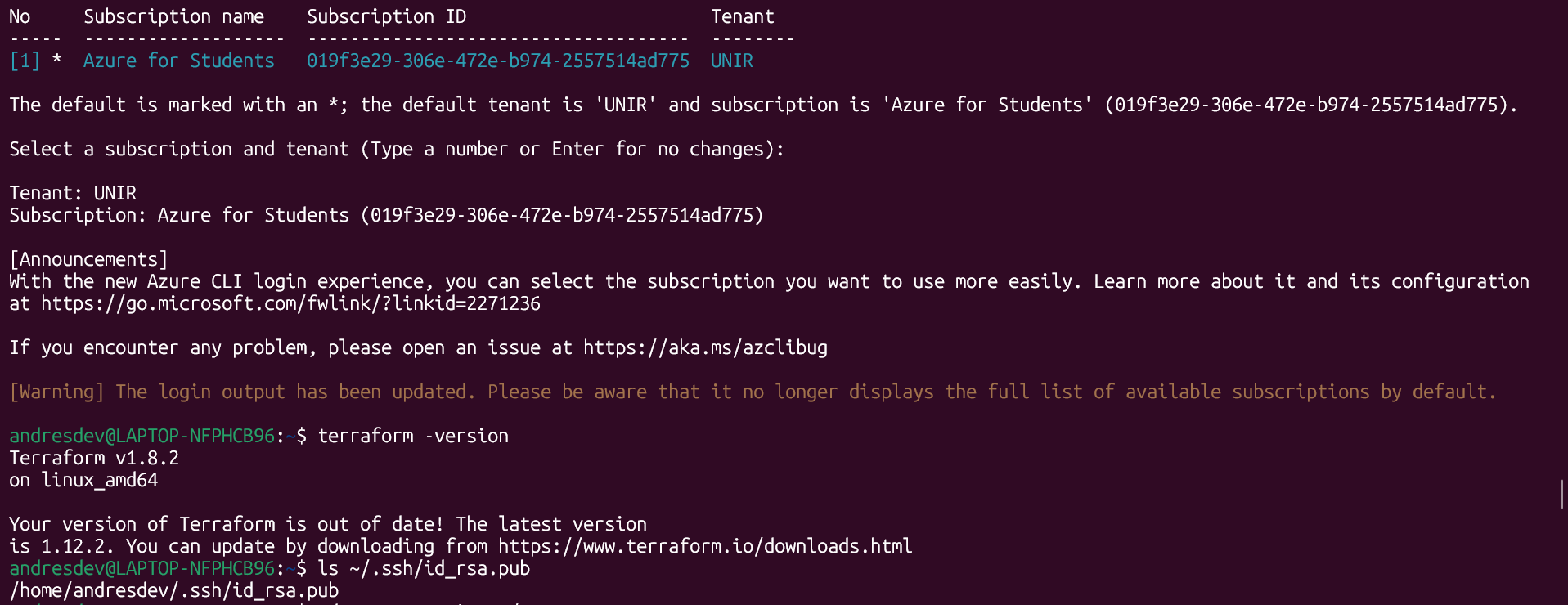
## Evidencia 7

Descripción: Captura relacionada al despliegue o automatización del proyecto.



## Evidencia 8

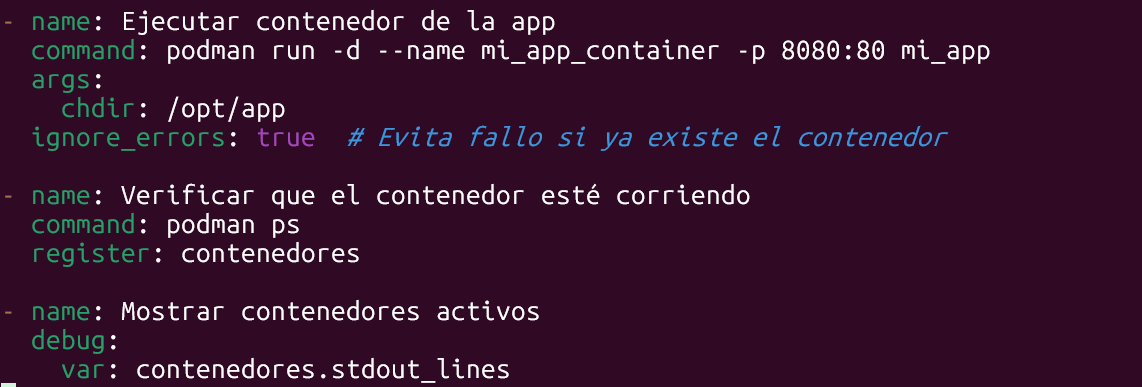
Descripción: Captura relacionada al despliegue o automatización del proyecto.



## Evidencia 9

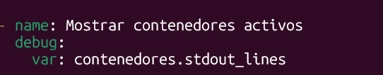
Descripción: Captura relacionada al despliegue o automatización del proyecto.





## Evidencia 10

Descripción: Captura relacionada al despliegue o automatización del proyecto.



## Evidencia 11

Descripción: Captura relacionada al despliegue o automatización del proyecto.



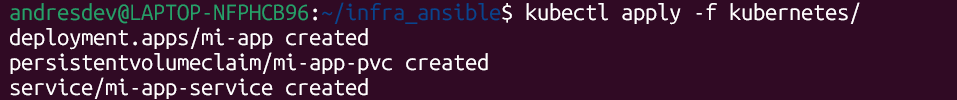
## Evidencia 12

Descripción: Captura relacionada al despliegue o automatización del proyecto.



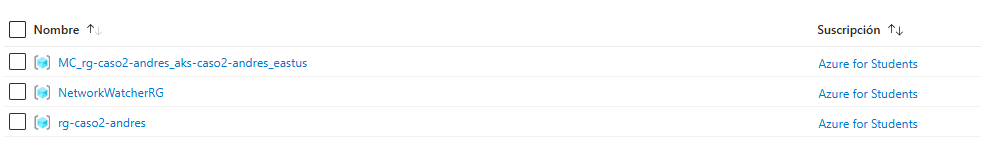
## Evidencia 13

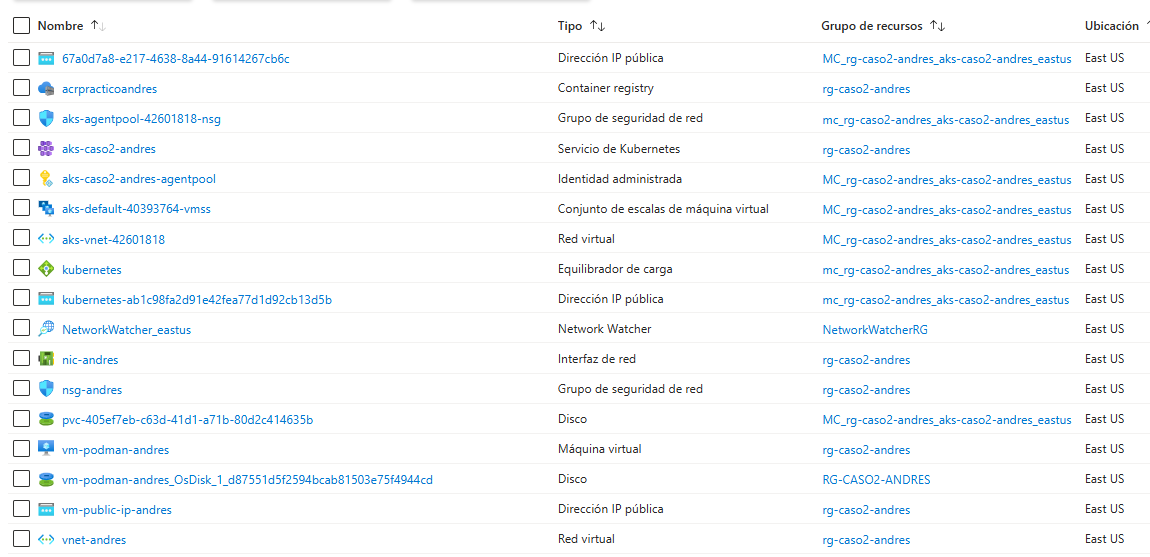
Descripción: Captura relacionada al despliegue o automatización del proyecto.



## Evidencia 14

Descripción: Captura relacionada al despliegue o automatización del proyecto.





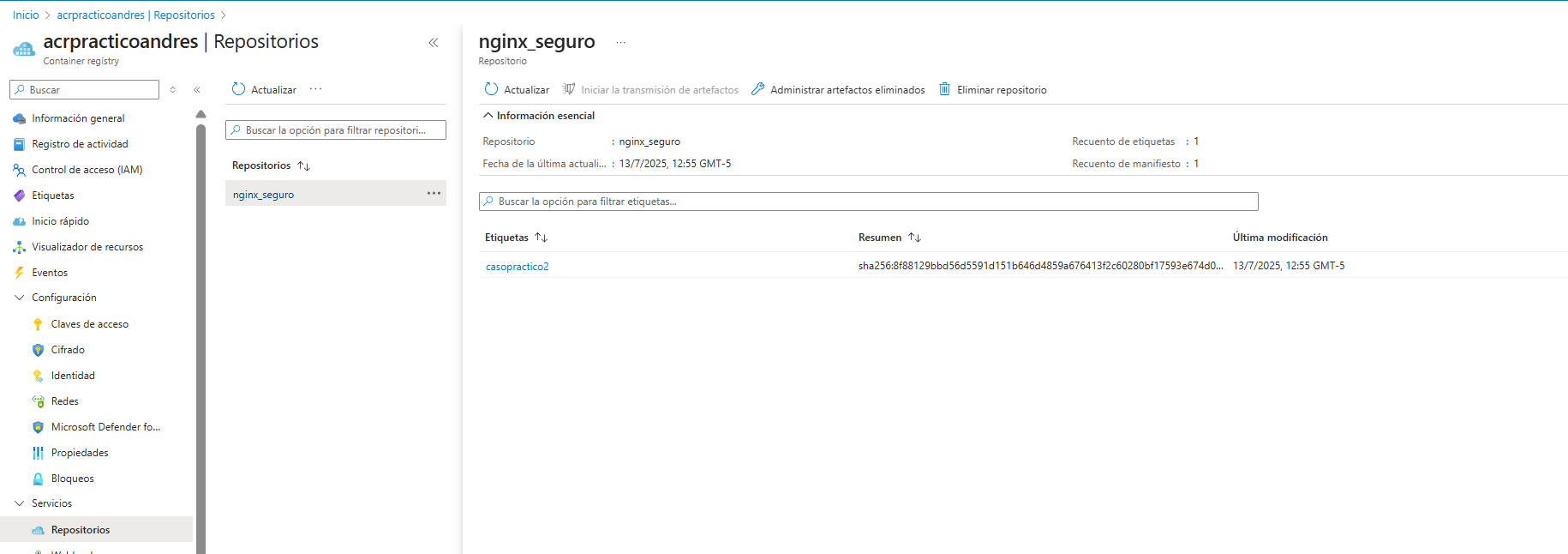
## Evidencia 15

Descripción: Captura relacionada al despliegue o automatización del proyecto.



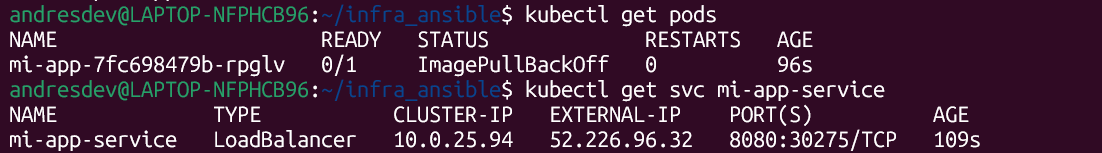
## Evidencia 16

Descripción: Captura relacionada al despliegue o automatización del proyecto.



## Evidencia 17

Descripción: Captura relacionada al despliegue o automatización del proyecto.



## Evidencia 18

Descripción: Captura relacionada al despliegue o automatización del proyecto.



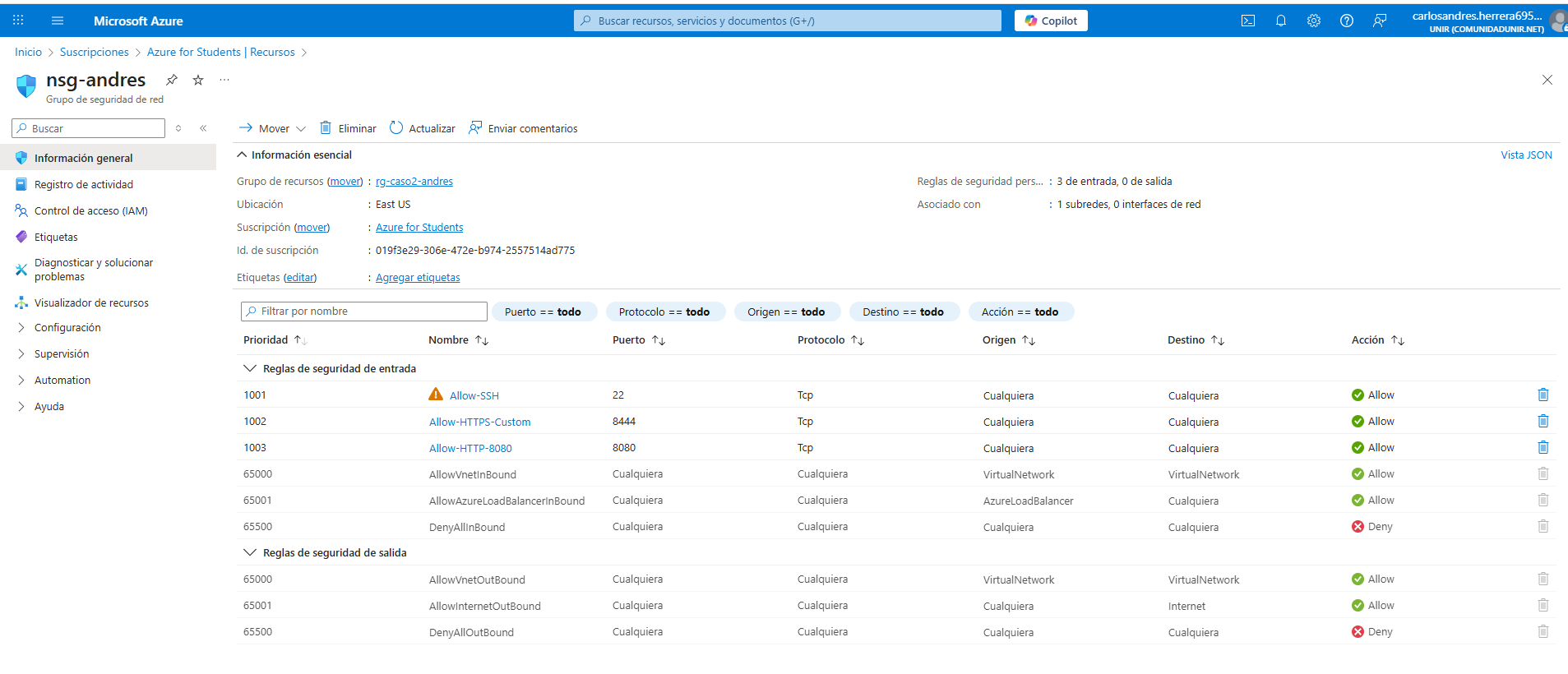
## Evidencia 19

Descripción: Captura relacionada al despliegue o automatización del proyecto.



## Evidencia 20

Descripción: Captura relacionada al despliegue o automatización del proyecto.



## 

## Lecciones Aprendidas

Este proyecto permitió reforzar conocimientos en infraestructura como código, manejo de errores comunes en AKS, trabajo colaborativo con Git, automatización con Ansible y despliegue de contenedores tanto en entornos locales como orquestados. Además, se destacó la importancia de documentar cada error y su solución para futuras referencias.