NetApp AIPod Mini para ERAG - Pasos de la implementación

NetApp artificial intelligence solutions

NetApp
February 12, 2026

Tabla de contenidos

NetApp AIPod Mini para ERAG - Pasos de la implementación

Este documento proporciona una guía completa, paso a paso, para el despliegue de NetApp AIPod Mini para Enterprise RAG(ERAG) 2.0. Cubre la instalación y configuración de extremo a extremo de todos los componentes principales, incluyendo la plataforma Kubernetes, NetApp Trident para la orquestación de almacenamiento y la pila ERAG 2.0 usando ansible playbooks. Además del flujo de trabajo de despliegue, el documento incluye una guía dedicada a la solución de problemas que recoge los problemas más comunes encontrados durante la instalación, sus causas raíz y las resoluciones recomendadas para apoyar una experiencia de despliegue fluida y confiable.

Sathish Thyagarajan, Michael Oglesby, Arpita Mahajan NetApp

Suposiciones:

- El usuario de despliegue tiene permisos suficientes para crear espacios de nombres e instalar Helm charts.
- Los servidores Xeon ejecutan Ubuntu 22.04.
- El mismo nombre de usuario está configurado en todos los servidores Xeon.
- El acceso administrativo de DNS está disponible.
- ONTAP 9.16 desplegado con una SVM configurada para acceso a S3.
- Se crea y configura el bucket S3.

Prerrequisitos

Instala Git, Python3.11 y pip para Python3.11

En Ubuntu 22.04:

```
add-apt-repository ppa:deadsnakes/ppa
apt update
apt upgrade
apt install python3.11
python3.11 --version
apt install python3.11-pip
python3.11 -m pip --version
```

Pasos de implementación de ERAG 2.0/2.0.1

1. Descarga la versión Enterprise RAG 2.0 desde GitHub

```
git clone https://github.com/opea-project/Enterprise-RAG.git
cd Enterprise-RAG/
git checkout tags/release-2.0.0
```

Para ERAG 2.0.1, usa el siguiente comando

```
git checkout tags/release-2.0.1
```

2. Instala los requisitos previos

```
cd deployment/
sudo apt-get install python3.11-venv
python3 -m venv erag-venv
source erag-venv/bin/activate
pip install --upgrade pip
pip install -r requirements.txt
ansible-galaxy collection install -r requirements.yaml --upgrade
```

3. Crear archivo de inventario

```
cp -a inventory/sample inventory/<cluster-name>
vi inventory/<cluster-name>/inventory.ini
# Control plane nodes
kube-3 ansible_host=<control_node_ip_address>

# Worker nodes
kube-1 ansible_host=<worker_node1_ip_address>
kube-2 ansible_host=<worker_node2_ip_address>

# Define node groups
[kube_control_plane]
kube-1
kube-2
kube-3

[kube_node]
kube-1
kube-2

[etcd:children]
kube_control_plane

[k8s_cluster:children]
kube_control_plane
kube_node

# Vars
[k8s_cluster:vars]
ansible_become=true
ansible_user=<ssh_username>
ansible_connection=ssh
```

4. Configura SSH sin contraseña para cada nodo

```
ssh-copy-id REMOTE_USER@MACHINE_IP
```

Nota: si se utiliza un nodo de despliegue para desplegar el ERAG, asegúrate de que SSH sin contraseña también esté configurado en el nodo de despliegue.

5. Verifica la conectividad

```
ansible all -i inventory/<cluster-name>/inventory.ini -m ping
```

Nota: Si no tienes configurado sudo sin contraseña en tus nodos, entonces necesitarás añadir --ask-become -pass a este comando. Cuando uses --ask-become-pass, es fundamental que el usuario ssh tenga la MISMA contraseña en cada nodo.

6. Edita `config.yaml` el archivo

Prepara el despliegue editando `inventory/<cluster-name>/config.yaml` para reflejar las características específicas de tu entorno.

```
vi inventory/<cluster-name>/config.yaml
```

Fragmento de muestra:

```
…
deploy_k8s: true
…
install_csi: "netapp-trident"
…
local_registry: false
…
trident_operator_version: "2510.0"    # Trident operator version (becomes
100.2506.0 in Helm chart)
trident_namespace: "trident"          # Kubernetes namespace for Trident
trident_storage_class: "netapp-trident" # StorageClass name for Trident
trident_backend_name: "ontap-nas"     # Backend configuration name
…
ontap_management_lif: "<ontap_mgmt_lif>"              # ONTAP management
LIF IP address
ontap_data_lif: "<ontap_nfs_data_lif>"                   # ONTAP data LIF
IP address
ontap_svm: "<ontap_svm>"                          # Storage Virtual Machine
(SVM) name
ontap_username: "<ontap_username>"                    # ONTAP username
with admin privileges
ontap_password: "<redacted>"                     # ONTAP password
ontap_aggregate: "<ontap_aggr>"                    # ONTAP aggregate name
for volume creation
…
kubeconfig: "<repository path>/deployment/inventory/<cluster-
name>/artifacts/admin.conf"
…
```

7. Despliega el clúster K8s (con Trident)

Ejecuta ansible-playbook playbooks/infrastructure.yaml con las etiquetas configure e install para desplegar el

clúster y Trident CSI.

```
ansible-playbook playbooks/infrastructure.yaml --tags configure,install -i
inventory/<cluster-name>/inventory.ini -e @inventory/<cluster-
name>/config.yaml
```

Nota: - Si no tienes configurado sudo sin contraseña en tus nodos, entonces necesitas añadir `--ask-become-pass` a este comando. Cuando uses `--ask-become-pass`, es fundamental que el usuario ssh tenga la MISMA contraseña en cada nodo. - Consulta "NetApp Trident CSI Integration para RAG empresarial" para más detalles. Consulta "Documentación de instalación de Trident" para más detalles.

8. Cambia el número de descriptores abiertos de iwatch

Consulta el "iwatch descriptores abiertos" para más detalles.

9. Instala kubectl

Consulta "Instala Kubectl" si aún no está instalado. Recupera el archivo kubeconfig de `<repository path>/deployment/inventory/<cluster-name>/artifacts/admin.conf`.

10. Instala MetalLB en un clúster Kubernetes

Instala MetalLB usando helm en tu clúster de Kubernetes.

```
helm repo add metallb https://metallb.github.io/metallb
helm -n metallb-system install metallb metallb/metallb --create-namespace
```

Consulta el "Instalación de MetalLB" para más detalles.

11. Configura MetalLB

MetalLB se configuró en modo de Capa 2 y se crearon los recursos IPAddressPool y L2Advertisement requeridos según las directrices de configuración documentadas.

```
vi metallb-ipaddrpool-l2adv.yaml
kubectl apply -f metallb-ipaddrpool-l2adv.yaml
```

Fragmento de muestra:

```
vi metallb-ipaddrpool-l2adv.yaml
---
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: erag
  namespace: metallb-system
spec:
  addresses:
  - <IPAddressPool>
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: metallb-l2adv
  namespace: metallb-system
```

Nota: - Usa `metallb-system` como namespace para MetalLB IPAddressPool y L2Advertisement. - El pool de direcciones IP puede incluir cualquier IP no utilizada dentro de la misma subred que los nodos Kubernetes. Solo se requiere una única dirección IP para ERAG. - Consulta "Configuración de MetalLB Layer2" para detalles.

12. Actualiza config.yaml con FQDN, modo de acceso al volumen, ingreso y detalles de S3.

Modifica el archivo config.yaml ubicado en `inventory/<cluster-name>/config.yaml` para definir el FQDN de despliegue, establecer los modos de acceso al volumen, configurar la exposición de ingress e integrar ONTAP S3.

Edita `config.yaml` y aplica los siguientes cambios de configuración:

- FQDN: especifica el dominio completo que se usa para acceder a la implementación.
- Modo de acceso a volúmenes: en la sección gmc.pvc, configura `accessMode: ReadWriteMany` para admitir el acceso simultáneo a volúmenes de modelos en varios pods.
- Configuración de ingreso: configura el service_type de ingreso como LoadBalancer para permitir el acceso externo a la aplicación.
- Detalles de almacenamiento S3: establece storageType en s3compatible y configura los parámetros de ONTAP S3, incluida la región, las credenciales de acceso y los puntos finales internos y externos.
- Verificación de certificados SSL: establece edpInternalCertVerify y edpExternalCertVerify en false solo cuando ONTAP S3 está configurado con certificados autofirmados. Si los certificados son emitidos por una CA de confianza pública, estos parámetros deben permanecer habilitados.

Fragmento de muestra:

```
vi inventory/<cluster-name>/config.yaml
…
FQDN: "<FQDN>" # Provide the FQDN for the deployment
…
gmc:
  enabled: true
  pvc:
    accessMode: ReadWriteMany # AccessMode
    models:
      modelLlm:
        name: model-volume-llm
        storage: 100Gi
      modelEmbedding:
        name: model-volume-embedding
        storage: 20Gi
      modelReranker:
        name: model-volume-reranker
        storage: 10Gi
…
ingress:
  …
  service_type: LoadBalancer
  …
edp:
  …
  storageType: s3compatible
  …
  s3compatible:
    region: "us-east-1"
    accessKeyId: "<your_access_key>"
    secretAccessKey: "<your_secret_key>"
    internalUrl: "https://<IP-address>"
    externalUrl: "https://<IP-address>"
    bucketNameRegexFilter: ".*"
    edpExternalCertVerify: false
    edpInternalCertVerify: false
  …
```

Nota: - De forma predeterminada, la aplicación Intel® AI for Enterprise RAG ingiere datos de todos los buckets existentes en tu SVM. Si tienes varios buckets en tu SVM, puedes modificar el campo `bucketNameRegexFilter` para que los datos se ingieran solo de ciertos buckets. - Consulta la documentación de "Implementación de RAG de Intel® AI for Enterprise" para más detalles.

13. Configura los ajustes de sincronización programada

Al instalar la aplicación OPEA para Intel® AI for Enterprise RAG, habilita `scheduledSync` para que la aplicación ingiera automáticamente archivos nuevos o actualizados desde tus buckets de S3.

Cuando `scheduledSync` está habilitado, la aplicación verifica automáticamente sus buckets S3 de origen en busca de archivos nuevos o actualizados. Cualquier archivo nuevo o actualizado que se encuentre como parte de este proceso de sincronización se incorpora automáticamente y se agrega a la base de conocimiento de RAG. La aplicación verifica sus depósitos de origen según un intervalo de tiempo preestablecido. El intervalo de tiempo predeterminado es de 60 segundos, lo que significa que la aplicación comprueba si hay cambios cada 60 segundos. Es posible que desee cambiar este intervalo para adaptarlo a sus necesidades específicas.

Para habilitar `scheduledSync` y establecer el intervalo de sincronización, establece los siguientes valores en `deployment/components/edp/values.yaml`:

```
vi components/edp/values.yaml
…
presignedUrlCredentialsSystemFallback: "true"
…
celery:
  …
  config:
    …
    scheduledSync:
      enabled: true
      syncPeriodSeconds: "60"
…
```

14. Implementa Enterprise RAG 2.0/2.0.1

Antes de la instalación, valida la preparación de la infraestructura siguiendo los procedimientos descritos en "Guía de implementación de aplicaciones Intel® AI for Enterprise RAG". Este paso asegura que la infraestructura subyacente esté configurada correctamente y cumpla todos los requisitos previos necesarios para una instalación exitosa de Enterprise RAG Application.

Ejecuta la instalación usando:

```
ansible-playbook -u $USER playbooks/application.yaml --tags
configure,install -e @inventory/<cluster-name>/config.yaml
```

Nota: Si no tienes configurado sudo sin contraseña en tu nodo de despliegue (el portátil o el host de salto donde estás ejecutando el comando ansible-playbook), entonces necesitarás añadir `--ask-become-pass` a este comando. Cuando uses `--ask-become-pass`, es fundamental que el usuario ssh tenga la MISMA contraseña en cada nodo.

15. Crear una entrada DNS

Crea una entrada DNS para el panel web de Enterprise RAG en tu servidor DNS. Para continuar, recupera la dirección IP externa asignada al ingress de Enterprise RAG LoadBalancer:

```
kubectl -n ingress-nginx get svc ingress-nginx-controller
```

Crea una entrada DNS que apunte a esta dirección IP para el FQDN que usaste en el paso 12.

Nota: - El FQDN usado para la entrada DNS DEBE coincidir con el FQDN del archivo de configuración.

16. Accede a la interfaz RAG de Enterprise

Accede a Enterprise RAG UI navegando a ese FQDN en tu navegador. Nota: puedes recuperar las credenciales predeterminadas de UI desde cat ansible-logs/default_credentials.txt

Guía de resolución de problemas

1. Problema: conflicto en la instalación de Keycloak Helm

Escenario: al desplegar ERAG, la instalación de Keycloak puede fallar con el siguiente error:

```
FAILED - RETRYING: [localhost]: Install Keycloak Helm chart (5 retries
left).
Failure when executing Helm command. Exited 1.
    stdout:
    stderr: Error: UPGRADE FAILED: another operation
(install/upgrade/rollback) is in progress
```

Acción: si el fallo persiste después de reintentos, desinstala la implementación de ERAG, elimina el espacio de nombres auth existente usando los siguientes comandos y vuelve a ejecutar la implementación.

```
ansible-playbook playbooks/application.yaml --tags uninstall -e
@inventory/<cluster-name>/config.yaml

helm -n auth uninstall keycloak
kubectl -n auth get pvc # confirm all PVCs are gone; if any are left,
delete them
kubectl delete ns auth
```

Nota: el estado obsoleto de la versión de Helm puede bloquear posteriores operaciones de instalación o actualización.

2. Problema: no se encuentra la versión del Helm Chart del operador de Trident

Escenario: durante el despliegue de ERAG, la instalación del operador Trident puede fallar debido a un

desajuste en la versión del Helm chart. Se puede observar el siguiente error:

```
TASK [netapp_trident_csi_setup : Install Trident operator via Helm]
fatal: [localhost]: FAILED! => changed=false
  command: /usr/local/bin/helm --version=100.2510.0 show chart 'netapp-
trident/trident-operator'
  msg: |-
    Failure when executing Helm command. Exited 1.
    stdout:
    stderr: Error: chart "trident-operator" matching 100.2510.0 not found
in netapp-trident index.
            (try 'helm repo update'): no chart version found for trident-
operator-100.2510.0
```

Acción: si se produce este error, actualiza el índice del repositorio Helm y vuelve a ejecutar el playbook de despliegue.

```
helm repo update
ansible-playbook playbooks/application.yaml -e @inventory/<cluster-
name>/config.yaml
```

Nota: este es un problema conocido en la versión 2.0 de ERAG. Se ha enviado una solución y se incluirá en una futura versión.