NetApp AIPod Mini per ERAG - Fasi di deployment

NetApp artificial intelligence solutions

NetApp
February 12, 2026

Sommario

NetApp AIPod Mini per ERAG - Fasi di deployment

Questo documento fornisce una guida completa e dettagliata per l'implementazione di NetApp AIPod Mini per Enterprise RAG(ERAG) 2.0. Copre l'installazione e la configurazione end-to-end di tutti i componenti principali, inclusa la piattaforma Kubernetes, NetApp Trident per l'orchestrazione dello storage e lo stack ERAG 2.0 utilizzando i playbook ansible. Oltre al flusso di lavoro di implementazione, il documento include una guida dedicata alla risoluzione dei problemi che descrive i problemi più comuni riscontrati durante l'installazione, le relative cause principali e le soluzioni consigliate per supportare un'esperienza di implementazione fluida e affidabile.

Sathish Thyagarajan, Michael Oglesby, Arpita Mahajan NetApp

Ipotesi:

- L'utente di distribuzione dispone di autorizzazioni sufficienti per creare namespace e installare Helm charts.
- I server Xeon eseguono Ubuntu 22.04.
- Lo stesso nome utente è configurato su tutti i server Xeon.
- L'accesso amministrativo DNS è disponibile.
- ONTAP 9.16 distribuito con una SVM configurata per l'accesso S3.
- Il bucket S3 è creato e configurato.

Prerequisiti

Installa Git, Python3.11 e pip per Python3.11

Su 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
```

ERAG 2.0/2.0.1 Fasi di distribuzione

1. Estrarre la versione Enterprise RAG 2.0 da GitHub

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

Per ERAG 2.0.1, utilizzare il comando seguente

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

2. Installare i prerequisiti

```
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. Crea file di 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. Imposta SSH senza password su ogni nodo

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

Nota: se si utilizza un nodo di deploy per distribuire l'ERAG, assicurarsi che anche sul nodo di deploy sia configurato l'SSH senza password.

5. Verificare la connettività

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

Nota: se non hai configurato sudo senza password sui tuoi nodi, dovrai aggiungere --ask-become-pass a questo comando. Quando si utilizza --ask-become-pass, è fondamentale che l'utente ssh abbia la STESSA password su ogni nodo.

6. Modifica `config.yaml` file

Prepara la distribuzione modificando `inventory/<cluster-name>/config.yaml` per riflettere le specifiche del tuo ambiente.

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

Esempio di snippet:

```
…
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. Distribuire il cluster (con Trident)

Esegui ansible-playbook playbooks/infrastructure.yaml con i tag configure e install per distribuire il cluster e

Trident CSI.

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

Nota: - Se non hai configurato sudo senza password sui tuoi nodi, dovrai aggiungere `--ask-become-pass` a questo comando. Quando si utilizza `--ask-become-pass`, è fondamentale che l'utente ssh abbia la STESSA password su ogni nodo. - Consultare "NetApp Trident CSI Integration per Enterprise RAG" per i dettagli. Consultare "Documentazione di installazione Trident" per ulteriori dettagli.

8. Modificare il numero di descrittori aperti di iwatch

Fare riferimento a "iwatch open descriptors" per i dettagli.

9. Installa kubectl

Fare riferimento a "Installa Kubectl" se non è già installato. Recuperare il file kubeconfig da `<repository path>/deployment/inventory/<cluster-name>/artifacts/admin.conf`.

10. Installa MetalLB nel cluster Kubernetes

Installa MetalLB utilizzando helm sul tuo cluster Kubernetes.

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

Fare riferimento a "Installazione di MetalLB" per dettagli.

11. Configura MetalLB

MetalLB è stato configurato in modalità Layer 2 e le risorse IPAddressPool e L2Advertisement richieste sono state create in conformità con le linee guida di configurazione documentate.

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

Esempio di snippet:

```
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: - Utilizzare `metallb-system` come namespace per MetalLB IPAddressPool e L2Advertisement. - Il pool di indirizzi IP può includere qualsiasi IP non utilizzato all'interno della stessa subnet dei nodi Kubernetes. È richiesto un solo indirizzo IP per ERAG. - Fare riferimento a "Configurazione Layer2 di MetalLB" per i dettagli.

12. Aggiorna config.yaml con FQDN, modalità di accesso al volume, ingress e dettagli S3.

Modificare il file config.yaml situato in `inventory/<cluster-name>/config.yaml` per definire il deployment FQDN, impostare le modalità di accesso al volume, configurare l'esposizione ingress e integrare ONTAP S3.

Modifica `config.yaml` e applica le seguenti modifiche alla configurazione:

- FQDN: specificare il fully qualified domain name utilizzato per accedere alla distribuzione.
- Modalità di accesso al volume: nella sezione gmc.pvc, impostare `accessMode: ReadWriteMany` per supportare l'accesso simultaneo ai volumi del modello su più pod.
- Configurazione di Ingress: configurare il service_type di Ingress come LoadBalancer per abilitare l'accesso esterno all'applicazione.
- Dettagli di archiviazione S3: impostare storageType su s3compatible e configurare i parametri ONTAP S3, inclusi regione, credenziali di accesso, endpoint interni ed esterni.
- Verifica del certificato SSL: impostare edpInternalCertVerify e edpExternalCertVerify su false solo quando ONTAP S3 è configurato con certificati autofirmati. Se i certificati sono emessi da una CA pubblicamente attendibile, questi parametri devono rimanere abilitati.

Esempio di snippet:

```
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: - Per impostazione predefinita, l® applicazione Intel® AI for Enterprise RAG acquisisce dati da tutti i bucket esistenti nella SVM. Se nella SVM sono presenti più bucket, è possibile modificare il `bucketNameRegexFilter` campo in modo che i dati vengano acquisiti solo da determinati bucket. - Consultare la documentazione "Implementazione RAG Intel® AI for Enterprise" per i dettagli.

13. Configurare le impostazioni di sincronizzazione pianificata

Quando si installa l'applicazione OPEA per Intel® AI for Enterprise RAG, abilitare `scheduledSync` in modo che l'applicazione acquisisca automaticamente file nuovi o aggiornati dai bucket S3.

Quando `scheduledSync` è abilitato, l'applicazione controlla automaticamente i bucket S3 di origine per file nuovi o aggiornati. Tutti i file nuovi o aggiornati rilevati durante questo processo di sincronizzazione vengono automaticamente acquisiti e aggiunti alla knowledge base RAG. L'applicazione controlla i bucket di origine in base a un intervallo di tempo preimpostato. L'intervallo di tempo predefinito è di 60 secondi, il che significa che l'applicazione verifica le modifiche ogni 60 secondi. Potresti voler modificare questo intervallo in base alle tue esigenze specifiche.

Per abilitare `scheduledSync` e impostare l'intervallo di sincronizzazione, impostare i seguenti valori in `deployment/components/edp/values.yaml`:

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

14. Distribuire Enterprise RAG 2.0/2.0.1

Prima dell'installazione, convalidare la disponibilità dell'infrastruttura seguendo le procedure descritte nel "Guida alla distribuzione dell'applicazione RAG Intel® AI for Enterprise". Questo passaggio garantisce che l'infrastruttura sottostante sia configurata correttamente e soddisfi tutti i prerequisiti richiesti per una riuscita installazione di Enterprise RAG Application.

Eseguire l'installazione utilizzando:

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

Nota: se non hai configurato sudo senza password sul tuo nodo di distribuzione (il laptop o l'host jump su cui stai eseguendo il comando ansible-playbook), dovrai aggiungere `--ask-become-pass` a questo comando. Quando si utilizza `--ask-become-pass`, è fondamentale che l'utente ssh abbia la STESSA password su ogni nodo.

15. Crea una voce DNS

Crea una voce DNS per la dashboard web di Enterprise RAG nel tuo DNS server. Per procedere, recupera

l'indirizzo IP esterno assegnato all'ingress di Enterprise RAG LoadBalancer:

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

Crea una voce DNS che punti a questo indirizzo IP per il fully qualified domain utilizzato nel passaggio 12.

Nota: - Il fully qualified domain utilizzato per la voce DNS DEVE corrispondere al fully qualified domain del file di configurazione.

16. Accedi all'interfaccia utente Enterprise RAG

Accedi all'interfaccia utente Enterprise RAG navigando fino a quel fully qualified domain (FQDN) nel tuo browser. Nota: puoi recuperare le credenziali predefinite dell'interfaccia utente da cat ansible-logs/default_credentials.txt

Guida alla risoluzione dei problemi

1. Problema: conflitto di installazione di Keycloak Helm

Scenario: durante la distribuzione di ERAG, l'installazione di Keycloak potrebbe non riuscire con il seguente errore:

```
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
```

Azione: se l'errore persiste dopo i nuovi tentativi, disinstallare la distribuzione ERAG, eliminare lo spazio dei nomi di autenticazione esistente utilizzando i comandi seguenti ed eseguire nuovamente la distribuzione.

```
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: uno stato di rilascio Helm obsoleto può bloccare le successive operazioni di installazione o aggiornamento.

2. Problema: versione della Helm Chart di Trident Operator non trovata

Scenario: durante l'implementazione di ERAG, l'installazione dell'operatore Trident potrebbe non riuscire a

causa di una mancata corrispondenza della versione della Helm chart. Potrebbe essere osservato il seguente errore:

```
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
```

Azione: se si verifica questo errore, aggiornare l'indice del repository Helm ed eseguire nuovamente il playbook di distribuzione.

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

Nota: questo è un problema noto nella versione 2.0 di ERAG. È stata inviata una correzione che sarà inclusa in una versione futura.