# NetApp AIPod Mini pour ERAG - Étapes de déploiement

## NetApp artificial intelligence solutions

NetApp
February 12, 2026

# Sommaire

# NetApp AIPod Mini pour ERAG - Étapes de déploiement

Ce document fournit un guide complet et détaillé pour le déploiement de NetApp AIPod Mini pour Enterprise RAG(ERAG) 2.0. Il couvre l'installation et la configuration de bout en bout de tous les composants principaux, y compris la plateforme Kubernetes, NetApp Trident pour l'orchestration du stockage, et la pile ERAG 2.0 à l'aide de playbooks ansible. En plus du workflow de déploiement, le document inclut un guide de dépannage dédié qui recense les problèmes courants rencontrés lors de l'installation, leurs causes profondes et les résolutions recommandées pour garantir une expérience de déploiement fluide et fiable.

Sathish Thyagarajan, Michael Oglesby, Arpita Mahajan NetApp

## Hypothèses :

- L'utilisateur chargé du déploiement dispose des autorisations suffisantes pour créer un espace de noms et installer des charts Helm.
- Les serveurs Xeon fonctionnent sous Ubuntu 22.04.
- Le même nom d'utilisateur est configuré sur tous les serveurs Xeon.
- L'accès administratif DNS est disponible.
- ONTAP 9.16 déployé avec une SVM configurée pour l'accès S3.
- Le compartiment S3 est créé et configuré.

## Prérequis

Installez Git, Python3.11 et pip pour Python3.11

Sur 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
```

# Étapes de déploiement ERAG 2.0/2.0.1

## 1. Récupérez la version Enterprise RAG 2.0 depuis GitHub

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

Pour ERAG 2.0.1, utilisez la commande ci-dessous

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

## 2. Installer les prérequis

```
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. Créer un fichier d’inventaire

```
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. Configurez une connexion SSH sans mot de passe pour chaque nœud

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

Remarque : Si un nœud de déploiement est utilisé pour déployer l'ERAG, assurez-vous que le protocole SSH sans mot de passe est également configuré sur le nœud de déploiement.

## 5. Vérifier la connectivité

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

Remarque : Si vous n'avez pas configuré l'accès sudo sans mot de passe sur vos nœuds, vous devrez ajouter --ask-become-pass à cette commande. Lorsque vous utilisez --ask-become-pass, il est impératif que l'utilisateur SSH possède le même mot de passe sur chaque nœud.

## 6. Modifier `config.yaml` file

Préparez le déploiement en modifiant `inventory/<cluster-name>/config.yaml` pour refléter les spécificités de votre environnement.

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

**Extrait d'exemple :**

```
…
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. Déployez le cluster K8s (avec Trident)

Exécutez ansible-playbook playbooks/infrastructure.yaml avec les balises configure et install pour déployer le

cluster et Trident CSI.

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

Remarque : - Si vous n'avez pas configuré sudo sans mot de passe sur vos nœuds, vous devrez ajouter `--ask-become-pass` à cette commande. Lors de l'utilisation de `--ask-become-pass`, il est essentiel que l'utilisateur ssh ait le MÊME mot de passe sur chaque nœud. - Consultez la "NetApp Trident CSI Intégration pour RAG d'entreprise" pour plus de détails. Consultez la "Documentation d'installation de Trident" pour plus de détails.

## 8. Modifier le nombre de descripteurs iwatch ouverts

Veuillez vous référer au "descripteurs ouverts iwatch" pour plus de détails.

## 9. Installez kubectl

Consultez le "Installer Kubectl" si ce n'est pas déjà installé. Récupérez le fichier kubeconfig depuis `<repository path>/deployment/inventory/<cluster-name>/artifacts/admin.conf`.

## 10. Installez MetalLB dans le cluster Kubernetes

Installez MetalLB à l'aide de helm sur votre cluster Kubernetes.

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

Veuillez vous référer au "Installation de MetalLB" pour plus de détails.

## 11. Configurer MetalLB

MetalLB a été configuré en mode Layer 2, et les ressources IPAddressPool et L2Advertisement requises ont été créées conformément aux directives de configuration documentées.

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

**Extrait d'exemple :**

```
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
```

Remarque : - Utilisez `metallb-system` comme espace de noms pour MetalLB IPAddressPool et L2Advertisement. - Le pool d'adresses IP peut inclure toutes les adresses IP inutilisées du même sous-réseau que les nœuds Kubernetes. Une seule adresse IP est requise pour ERAG. - Consultez "Configuration Layer2 de MetalLB" pour plus de détails.

## 12. Mettez à jour le fichier config.yaml avec le FQDN, le mode d'accès au volume, l'ingress et les détails S3.

Modifiez le fichier config.yaml situé à `inventory/<cluster-name>/config.yaml` pour définir le FQDN du déploiement, définir les modes d'accès au volume, configurer l'exposition d'ingress et intégrer ONTAP S3.

Modifiez `config.yaml` et appliquez les changements de configuration suivants :

- FQDN : Spécifiez le domaine complet utilisé pour accéder au déploiement.
- Mode d'accès aux volumes : dans la section gmc.pvc, définissez `accessMode: ReadWriteMany` pour prendre en charge l'accès simultané aux volumes du modèle sur plusieurs pods.
- Configuration Ingress : configurez le service_type Ingress sur LoadBalancer pour permettre l'accès externe à l'application.
- Détails du stockage S3 : définissez storageType sur s3compatible et configurez les paramètres ONTAP S3, notamment la région, les informations d'identification d'accès et les points de terminaison internes et externes.
- Vérification du certificat SSL : définissez edpInternalCertVerify et edpExternalCertVerify sur false uniquement lorsque ONTAP S3 est configuré avec des certificats auto-signés. Si les certificats sont émis par une autorité de certification publiquement reconnue, ces paramètres doivent rester activés.

**Extrait d'exemple :**

```
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
  …
```

Remarque : Par défaut, l'application Intel® AI for Enterprise RAG ingère les données de tous les compartiments existants dans votre SVM. Si vous avez plusieurs compartiments dans votre SVM, vous pouvez modifier le `bucketNameRegexFilter` champ afin que les données ne soient ingérées qu'à partir de certains compartiments. - Consultez la "Déploiement de Intel® AI for Enterprise RAG" documentation pour plus de détails.

## 13. Configurer les paramètres de synchronisation planifiée

Lors de l'installation de l'OPEA pour Intel® AI for Enterprise RAG, activez `scheduledSync` afin que l'application ingère automatiquement les fichiers nouveaux ou mis à jour de vos compartiments S3.

Quand `scheduledSync` est activé, l'application vérifie automatiquement vos buckets S3 sources pour les fichiers nouveaux ou mis à jour. Tous les fichiers nouveaux ou mis à jour trouvés dans le cadre de ce processus de synchronisation sont automatiquement ingérés et ajoutés à la base de connaissances RAG. L'application vérifie vos buckets sources en fonction d'un intervalle de temps prédéfini. L'intervalle de temps par défaut est de 60 secondes, ce qui signifie que l'application vérifie les modifications toutes les 60 secondes. Vous souhaiterez peut-être modifier cet intervalle en fonction de vos besoins spécifiques.

Pour activer `scheduledSync` et définir l'intervalle de synchronisation, définissez les valeurs suivantes dans `deployment/components/edp/values.yaml` :

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

## 14. Déployer Enterprise RAG 2.0/2.0.1

Avant l'installation, validez la préparation de l'infrastructure en suivant les procédures décrites dans le "Guide de déploiement de l'application Intel® AI for Enterprise RAG". Cette étape garantit que l'infrastructure sous-jacente est correctement configurée et répond à toutes les conditions préalables requises pour une installation réussie de l'application Enterprise RAG.

Exécutez l'installation en utilisant :

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

Remarque : Si vous n'avez pas configuré sudo sans mot de passe sur votre nœud de déploiement (l'ordinateur portable ou le serveur intermédiaire sur lequel vous exécutez la commande ansible-playbook), vous devrez ajouter `--ask-become-pass` à cette commande. Lors de l'utilisation de `--ask-become-pass`, il est essentiel que l'utilisateur ssh ait le MÊME mot de passe sur chaque nœud.

## 15. Créez une entrée DNS

Créez une entrée DNS pour le tableau de bord web Enterprise RAG sur votre serveur DNS. Pour continuer,

récupérez l'adresse IP externe attribuée à l'ingress LoadBalancer d'Enterprise RAG :

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

Créez une entrée DNS pointant vers cette adresse IP pour le FQDN que vous avez utilisé à l'étape 12.

Remarque : - Le FQDN utilisé pour l'entrée DNS DOIT correspondre au FQDN du fichier de configuration.

### 16. Accéder à l'interface utilisateur Enterprise RAG

Accédez à l'interface utilisateur Enterprise RAG en naviguant vers ce FQDN dans votre navigateur. Remarque : vous pouvez récupérer les identifiants UI par défaut à partir de cat ansible-logs/default_credentials.txt

## Guide de dépannage

### 1. Problème : Conflit d'installation de Keycloak Helm

Scénario : Lors du déploiement d'ERAG, l'installation de Keycloak peut échouer avec l'erreur suivante :

```
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
```

Action : Si l'échec persiste après plusieurs tentatives, désinstallez le déploiement ERAG, supprimez l'espace de noms auth existant à l'aide des commandes ci-dessous et relancez le déploiement.

```
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
```

Remarque : Un état obsolète de la version Helm peut bloquer les opérations d'installation ou de mise à niveau ultérieures.

### 2. Problème : version du Helm Chart de Trident Operator introuvable

Scénario : Lors du déploiement d'ERAG, l'installation de l'opérateur Trident peut échouer en raison d'une incompatibilité de version du chart Helm. L'erreur suivante peut être observée :

```
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
```

Action: Si cette erreur se produit, mettez à jour l'index du dépôt Helm et relancez le playbook de déploiement.

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

Remarque : Il s'agit d'un problème connu dans la version 2.0 d'ERAG. Un correctif a été soumis et sera inclus dans une future version.