NetApp AIPod Mini für ERAG - Bereitstellungsschritte

NetApp artificial intelligence solutions

NetApp
February 12, 2026

Inhalt

NetApp AIPod Mini für ERAG - Bereitstellungsschritte

Dieses Dokument bietet eine umfassende Schritt-für-Schritt-Anleitung für die Bereitstellung von NetApp AIPod Mini für Enterprise RAG(ERAG) 2.0. Es behandelt die End-to-End-Installation und -Konfiguration aller Kernkomponenten, einschließlich der Kubernetes-Plattform, NetApp Trident für die Speicherorchestrierung und des ERAG 2.0-Stacks mithilfe von Ansible-Playbooks. Zusätzlich zum Bereitstellungs-Workflow enthält das Dokument einen speziellen Leitfaden zur Fehlerbehebung, der häufig auftretende Probleme während der Installation, deren Ursachen und empfohlene Lösungen erfasst, um eine reibungslose und zuverlässige Bereitstellung zu unterstützen.

Sathish Thyagarajan, Michael Oglesby, Arpita Mahajan NetApp

Annahmen:

- Der Deployment-Benutzer verfügt über ausreichende Berechtigungen, um einen Namespace zu erstellen und Helm-Charts zu installieren.
- Die Xeon-Server laufen mit Ubuntu 22.04.
- Auf allen Xeon-Servern ist derselbe Benutzername konfiguriert.
- DNS-Administrativzugriff ist verfügbar.
- ONTAP 9.16 mit einer für den S3-Zugriff konfigurierten SVM bereitgestellt.
- S3-Bucket ist erstellt und konfiguriert.

Voraussetzungen

Installieren Sie Git, Python3.11 und pip für Python3.11

Unter 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 Bereitstellungsschritte

1. Enterprise RAG 2.0-Release von GitHub herunterladen

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

Für ERAG 2.0.1 verwenden Sie den untenstehenden Befehl

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

2. Voraussetzungen installieren

```
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. Inventardatei erstellen

```
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. Richten Sie passwortloses SSH zu jedem Knoten ein

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

Hinweis: Wenn ein Deploy-Knoten für die Bereitstellung des ERAG verwendet wird, stellen Sie sicher, dass passwortloses SSH auch auf dem Deploy-Knoten konfiguriert ist.

5. Konnektivität prüfen

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

Hinweis: Falls auf Ihren Knoten kein passwortloses sudo eingerichtet ist, müssen Sie diesem Befehl die Option --ask-become-pass hinzufügen. Bei Verwendung von --ask-become-pass ist es unbedingt erforderlich, dass der SSH-Benutzer auf jedem Knoten dasselbe Passwort verwendet.

6. Datei `config.yaml` bearbeiten

Bereiten Sie die Bereitstellung vor, indem Sie `inventory/<cluster-name>/config.yaml` die Konfiguration an die Besonderheiten Ihrer Umgebung anpassen.

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

Beispiel-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. Stellen Sie den K8s-Cluster bereit (mit Trident)

Führen Sie ansible-playbook playbooks/infrastructure.yaml mit den Tags configure und install aus, um den

Cluster und Trident CSI bereitzustellen.

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

Hinweis: - Falls Sie auf Ihren Knoten kein passwortloses sudo eingerichtet haben, müssen Sie `--ask-become-pass` zu diesem Befehl hinzufügen. Bei der Verwendung von `--ask-become-pass` ist es unbedingt erforderlich, dass der SSH-Benutzer auf jedem Knoten dasselbe Passwort hat. - Siehe "NetApp Trident CSI Integration für Enterprise RAG" für Details. Siehe "Trident -Installationsdokumentation" für weitere Details.

8. Ändern Sie die Anzahl der iwatch-Open-Deskriptoren

Weitere Einzelheiten finden Sie in der "iwatch offene Beschreibungen".

9. Installieren Sie kubectl

Siehe "Installieren Sie Kubectl", falls es noch nicht installiert ist. Rufen Sie die kubeconfig-Datei von `<repository path>/deployment/inventory/<cluster-name>/artifacts/admin.conf` ab.

10. MetalLB im Kubernetes-Cluster installieren

Installieren Sie MetalLB mithilfe von helm auf Ihrem Kubernetes-Cluster.

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

Weitere Einzelheiten finden Sie in der "MetalLB-Installation".

11. MetalLB konfigurieren

MetalLB wurde im Layer-2-Modus konfiguriert, und die erforderlichen IPAddressPool- und L2Advertisement-Ressourcen wurden gemäß den dokumentierten Konfigurationsrichtlinien erstellt.

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

Beispiel-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
```

Hinweis: - Verwenden Sie `metallb-system` als Namespace für MetalLB IPAddressPool und L2Advertisement. - Der IP-Adresspool kann alle ungenutzten IPs innerhalb desselben Subnetzes wie die Kubernetes-Knoten enthalten. Für ERAG wird nur eine einzige IP-Adresse benötigt. - Siehe "MetalLB Layer2-Konfiguration" für Details.

12. Aktualisieren Sie die Datei config.yaml mit FQDN, Volume-Zugriffsmodus, Ingress und S3-Details.

Ändern Sie die Datei config.yaml unter `inventory/<cluster-name>/config.yaml` , um den FQDN für die Bereitstellung zu definieren, die Zugriffsmodi für die Volumes festzulegen, die Ingress-Exposition zu konfigurieren und ONTAP S3 zu integrieren.

Bearbeiten `config.yaml` und wenden Sie die folgenden Konfigurationsänderungen an:

- FQDN: Geben Sie die vollqualifizierte Domain an, die für den Zugriff auf die Bereitstellung verwendet wird.
- Zugriffsmodus für Volumes: Im Abschnitt gmc.pvc `accessMode: ReadWriteMany` so einstellen, dass der gleichzeitige Zugriff auf Modellvolumes über mehrere Pods hinweg unterstützt wird.
- Ingress-Konfiguration: Konfigurieren Sie den Ingress service_type als LoadBalancer, um externen Zugriff auf die Anwendung zu ermöglichen.
- S3-Speicherdetails: Setzen Sie storageType auf s3compatible und konfigurieren Sie ONTAP S3-Parameter, einschließlich Region, Zugangsdaten sowie interne und externe Endpunkte.
- SSL-Zertifikatsprüfung: Setzen Sie edpInternalCertVerify und edpExternalCertVerify nur dann auf false, wenn ONTAP S3 mit selbstsignierten Zertifikaten konfiguriert ist. Wenn Zertifikate von einer öffentlich vertrauenswürdigen CA ausgestellt werden, sollten diese Parameter aktiviert bleiben.

Beispiel-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
  …
```

Hinweis: - Standardmäßig erfasst die Intel AI for Enterprise RAG-Anwendung Daten aus allen vorhandenen Buckets in Ihrer SVM. Wenn Sie mehrere Buckets in Ihrer SVM haben, können Sie das `bucketNameRegexFilter` Feld so anpassen, dass Daten nur aus bestimmten Buckets erfasst werden. - Weitere Informationen finden Sie in der "Intel AI for Enterprise RAG-Bereitstellung" Dokumentation.

13. Konfigurieren Sie die Einstellungen für die geplante Synchronisierung

Aktivieren Sie bei der Installation der OPEA for Intel AI for Enterprise RAG-Anwendung `scheduledSync`, damit die Anwendung neue oder aktualisierte Dateien automatisch aus Ihren S3-Buckets aufnimmt.

Wann `scheduledSync` aktiviert ist, überprüft die Anwendung Ihre Quell-S3-Buckets automatisch auf neue oder aktualisierte Dateien. Alle neuen oder aktualisierten Dateien, die im Rahmen dieses Synchronisierungsprozesses gefunden werden, werden automatisch aufgenommen und der RAG-Wissensdatenbank hinzugefügt. Die Anwendung überprüft Ihre Quell-Buckets basierend auf einem voreingestellten Zeitintervall. Das Standardzeitintervall beträgt 60 Sekunden, was bedeutet, dass die Anwendung alle 60 Sekunden nach Änderungen sucht. Möglicherweise möchten Sie dieses Intervall Ihren speziellen Anforderungen entsprechend ändern.

Um `scheduledSync` und das Synchronisierungsintervall zu aktivieren und festzulegen, setzen Sie die folgenden Werte in `deployment/components/edp/values.yaml`:

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

14. Enterprise RAG 2.0/2.0.1 bereitstellen

Vor der Installation validieren Sie die Infrastrukturbereitschaft, indem Sie den im "Intel AI for Enterprise RAG-Anwendungsbereitstellungsleitfaden" beschriebenen Verfahren folgen. Dieser Schritt stellt sicher, dass die zugrunde liegende Infrastruktur korrekt konfiguriert ist und alle Voraussetzungen für eine erfolgreiche Enterprise RAG Application-Installation erfüllt.

Führen Sie die Installation mit aus:

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

Hinweis: Falls auf Ihrem Bereitstellungsknoten (dem Laptop oder Jump-Host, auf dem Sie den Befehl ansible-playbook ausführen) kein passwortloses sudo eingerichtet ist, müssen Sie `--ask-become-pass` zu diesem Befehl hinzufügen. Bei der Verwendung von `--ask-become-pass` ist es unbedingt erforderlich, dass der SSH-Benutzer auf jedem Knoten das gleiche Passwort verwendet.

15. Erstellen Sie einen DNS-Eintrag

Erstellen Sie einen DNS-Eintrag für das Enterprise RAG-Web-Dashboard auf Ihrem DNS-Server. Um fortzufahren, rufen Sie die dem Enterprise RAG-Ingress zugewiesene externe IP-Adresse ab LoadBalancer:

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

Erstellen Sie einen DNS-Eintrag, der auf diese IP-Adresse für den FQDN zeigt, den Sie in Schritt 12 verwendet haben.

Hinweis: - Der für den DNS-Eintrag verwendete FQDN MUSS mit dem FQDN aus der config file übereinstimmen.

16. Zugriff auf die Enterprise RAG UI

Greifen Sie auf die Enterprise RAG-Benutzeroberfläche zu, indem Sie in Ihrem Browser zu diesem FQDN navigieren. Hinweis: Sie können die Standard-Benutzeroberflächen-Anmeldeinformationen mit abrufen.

Leitfaden zur Fehlerbehebung

1. Problem: Keycloak Helm Installationskonflikt

Szenario: Bei der Bereitstellung von ERAG kann die Keycloak-Installation mit folgendem Fehler fehlschlagen:

```
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
```

Maßnahme: Wenn der Fehler nach wiederholten Versuchen weiterhin besteht, deinstallieren Sie die ERAG-Bereitstellung, löschen Sie den vorhandenen auth-Namespace mit den untenstehenden Befehlen und führen Sie die Bereitstellung erneut aus.

```
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
```

Hinweis: Ein veralteter Helm-Release-Status kann nachfolgende Installations- oder Upgrade-Vorgänge blockieren.

2. Problem: Trident Operator Helm Chart-Version nicht gefunden

Szenario: Während der ERAG-Bereitstellung kann die Installation des Trident-Operators aufgrund einer Helm-Chart-Versionsinkompatibilität fehlschlagen. Der folgende Fehler kann auftreten:

```
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
```

Maßnahme: Wenn dieser Fehler auftritt, aktualisieren Sie den Helm-Repository-Index und führen Sie das Deployment-Playbook erneut aus.

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

Hinweis: Dies ist ein bekanntes Problem in ERAG Version 2.0. Ein Fix wurde eingereicht und wird in einer zukünftigen Version enthalten sein.