Administrator's Guide

Purpose of this Guide

This guide serves as a focused documentation index for administrators looking for **step-by-step onboarding, deployment, configuration, and operational guidance**. This documentation is suitable for users deploying AccuKnox in real-world cloud-native environments. This curated guide is tailored for technical administrators and DevSecOps engineers who are looking for **concrete, task-oriented onboarding assets**, **installation steps**, and **configuration references**.

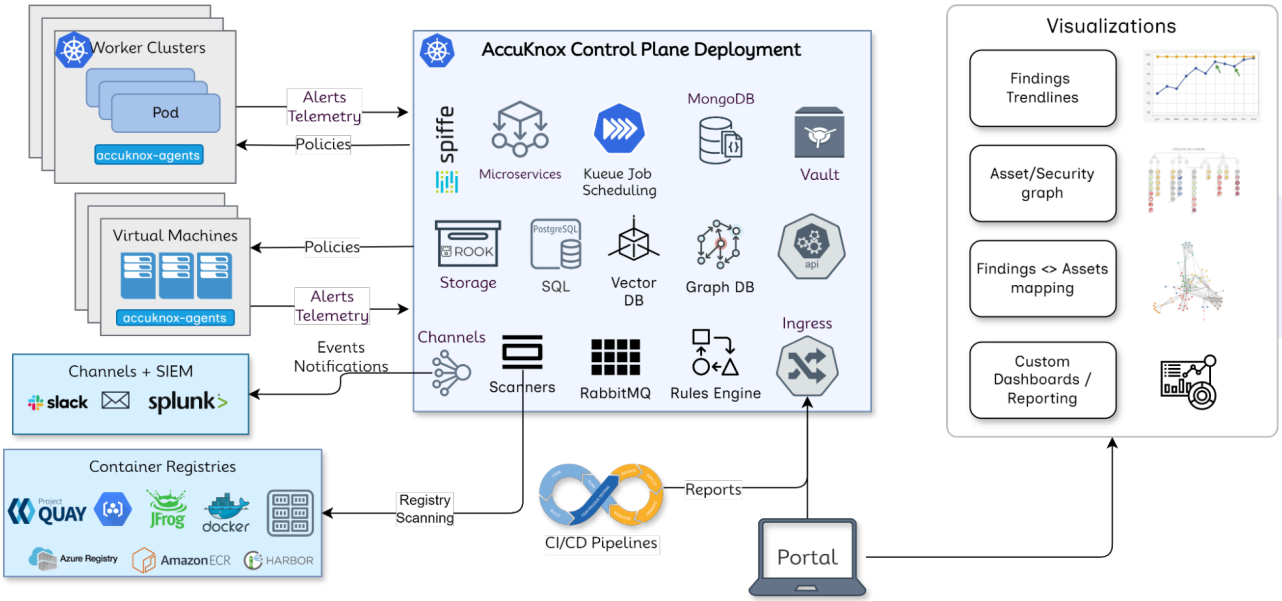
AccuKnox Enterprise Architecture

AccuKnox's Cloud-Native Application Protection Platform (CNAPP) offers a unified **AppSec + CloudSec** solution, integrating modules like ASPM, CSPM, CWPP, KIEM, and GRC. This architecture ensures comprehensive security across the software development lifecycle.

Core Components

Control Plane Architecture

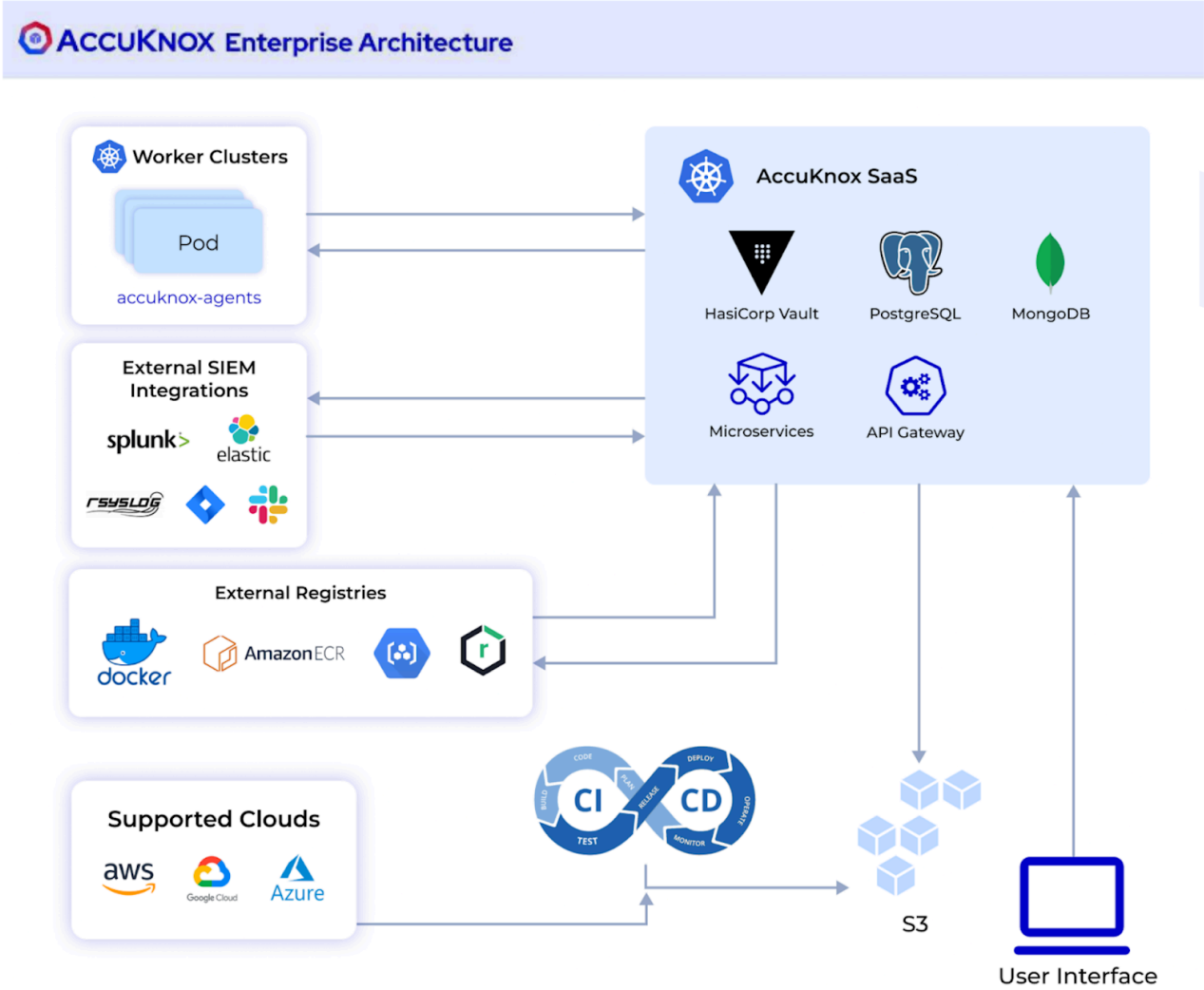
- **Microservices**:
  - *Divy*: Handles API requests.
  - *Celery*: Manages asynchronous tasks.
  - *Kueue*: Schedules Kubernetes-native jobs.
- **Parser Jobs**: Process asset and findings data, updating databases accordingly.
- **Alerts & Telemetry**: Ingested via RabbitMQ, processed for real-time insights.
- **Secure Onboarding**: Utilizes SPIFFE-based control plane for cluster onboarding.
- **Storage/Databases**:
  - *RDS*: Stores CSPM, KSPM, and ASPM data.
  - *MongoDB*: Handles streaming telemetry.
  - *Neo4j*: Manages metadata for KIEM.
- **Integrations**: Interfaces with SIEM tools (e.g., Splunk, Rsyslog) and ticketing systems (e.g., JIRA, Slack).



Key Components

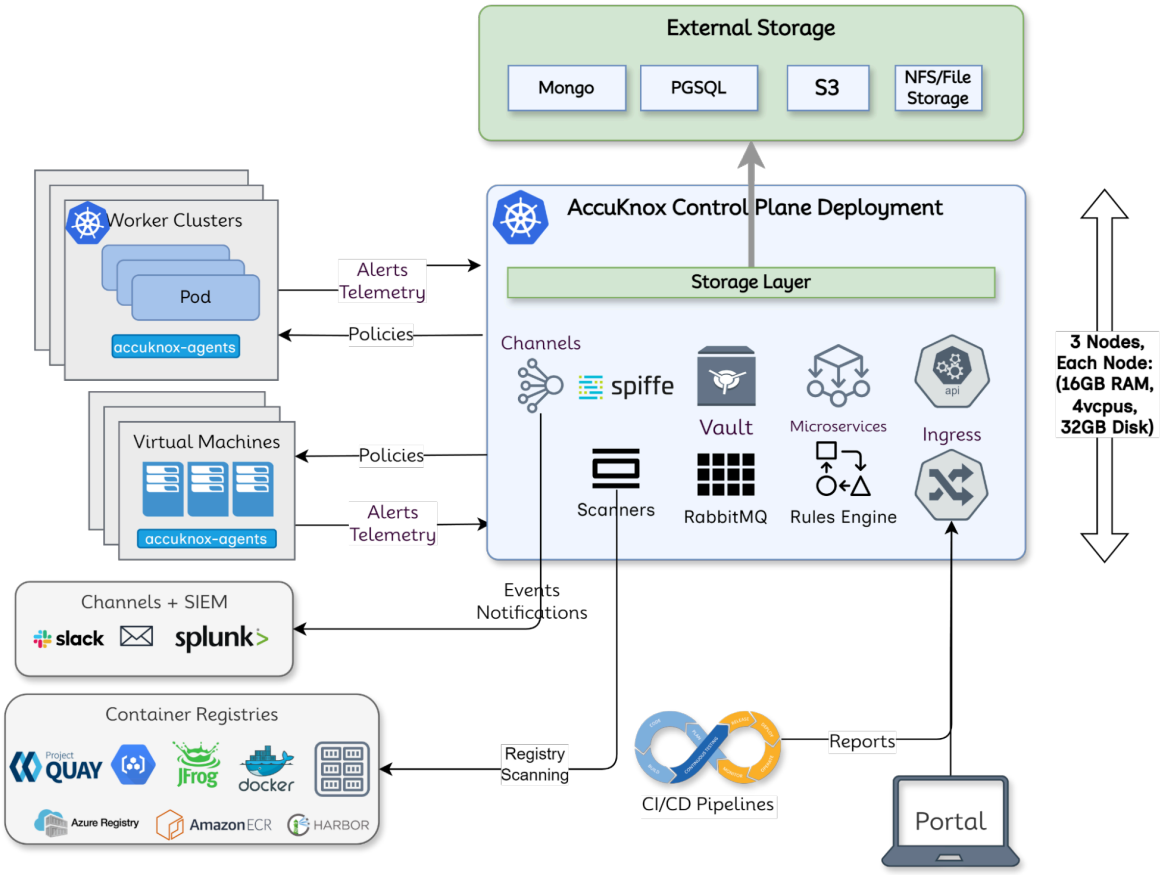
1. Playbook job scheduling: Microservices (Divy), Kueue scheduler, Celery tasks
2. Parser jobs for asset + findings database
3. Alerts and telemetry handling via RabbitMQ
4. SPIFFE-based secure cluster onboarding
5. Storage layer: RDS, MongoDB, Neo4j
6. External integrations & triggers handling

Cloud Architecture



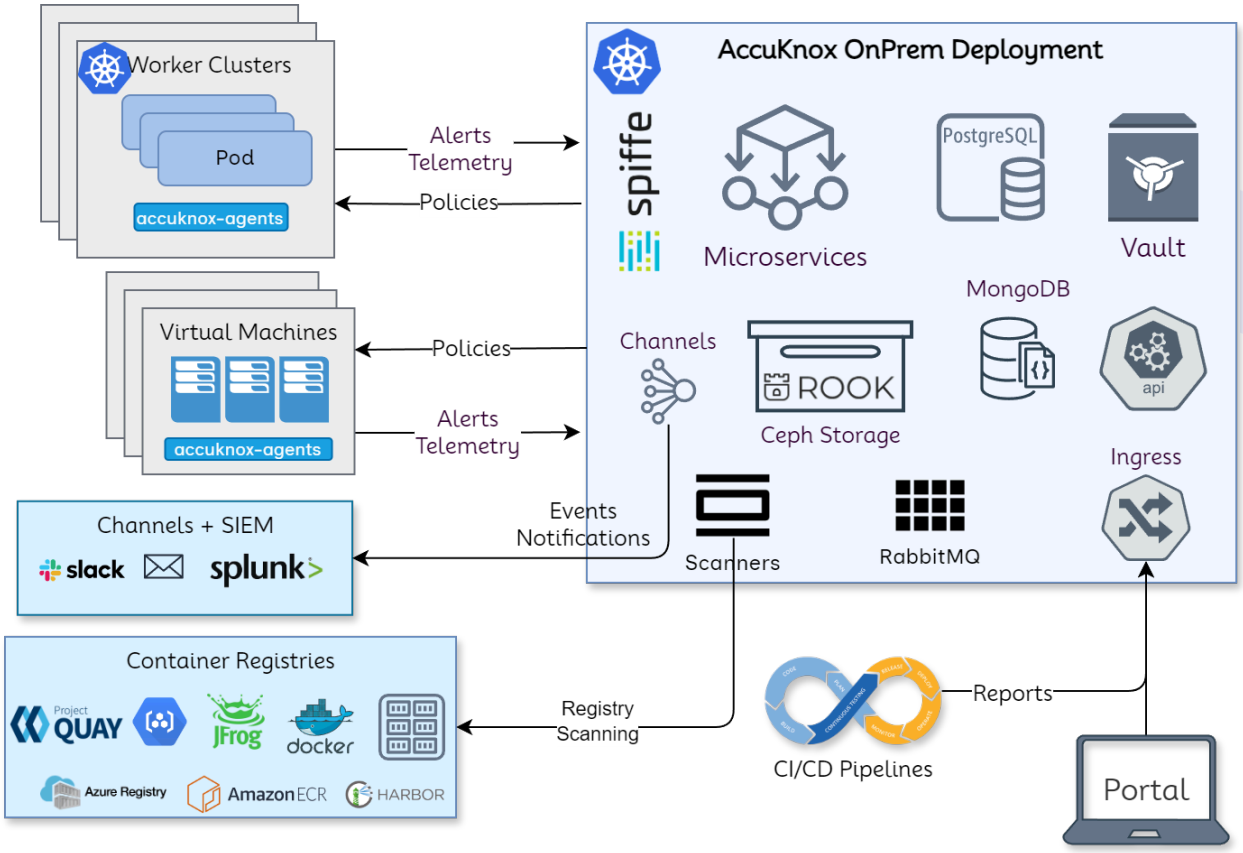
- SaaS and On-Prem support identical services (except AskADA AI Copilot – SaaS only)
- Tenant-level feature control
- Models:
    a. SaaS: AWS-managed (Aurora, S3)
    b. On-Prem: Full in-cluster setup (for air-gapped environments)
    c. Externalized: Uses customer DB/storage

Externalized Storage Architecture



- Supports deployments with customer-managed storage
- Enables hybrid cloud use cases
- Flexible DB integration (e.g., existing RDS, MongoDB, etc.)

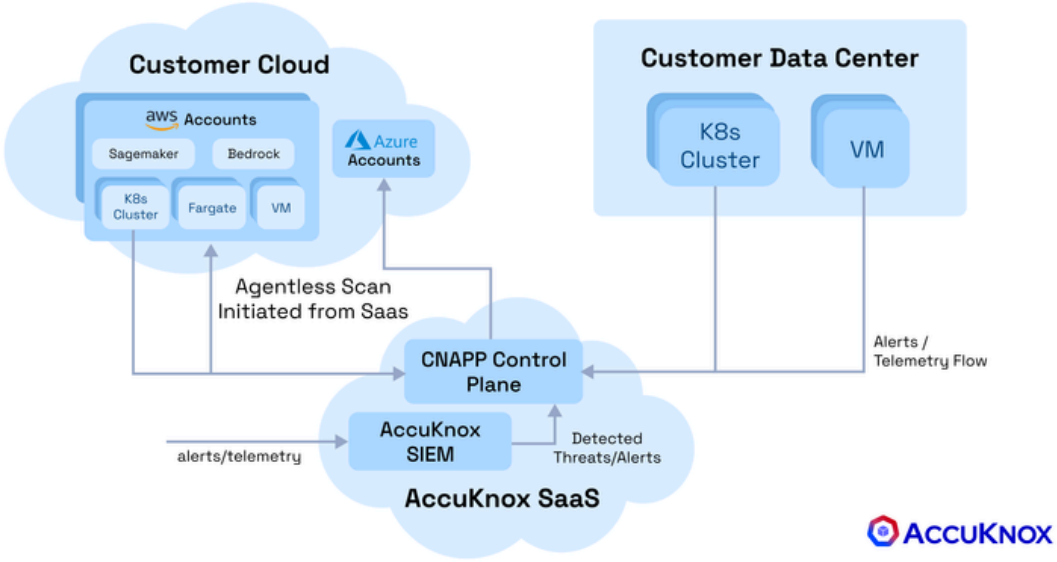
On-Premises Deployment Architecture



- K8s-native deployment
- No reliance on AWS managed services
- Designed for high-security & compliance environments

Deployment Details →

Scaling Considerations



Key Choke Points

1. **Playbook Jobs**: One AWS account = 272 jobs across regions
    - Kueue ensures tenant-aware resource allocation
2. **Parser Jobs**: Celery tasks parse reports & update DB
3. **Telemetry Overload**: Managed via thresholds & redirection to SIEM

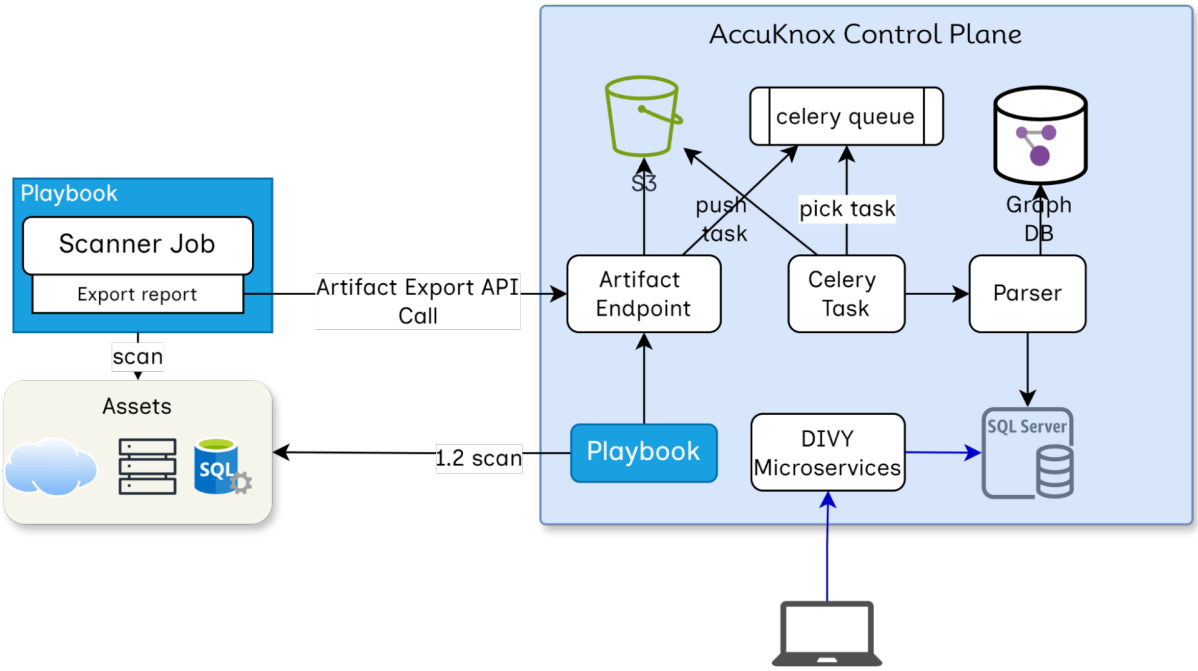
Noisy Neighbor Mitigation

- Celery replicated per tenant (currently manual)
- Kueue isolates playbook jobs per tenant
- RMQ overload handled by telemetry offload

Log & Data Storage

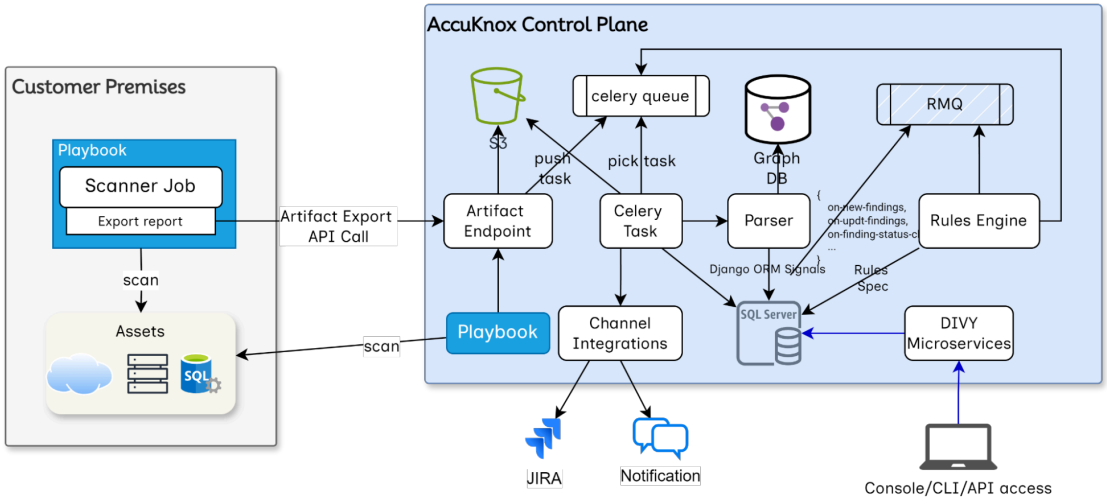
- **RDS**: CSPM, KSPM, ASPM (per-tenant tables)
- **MongoDB**: Telemetry logs (per-tenant collections)
- **Neo4j**: GraphDB for metadata (KIEM), expanding to assets/findings in v3.0

Customer Data Flow



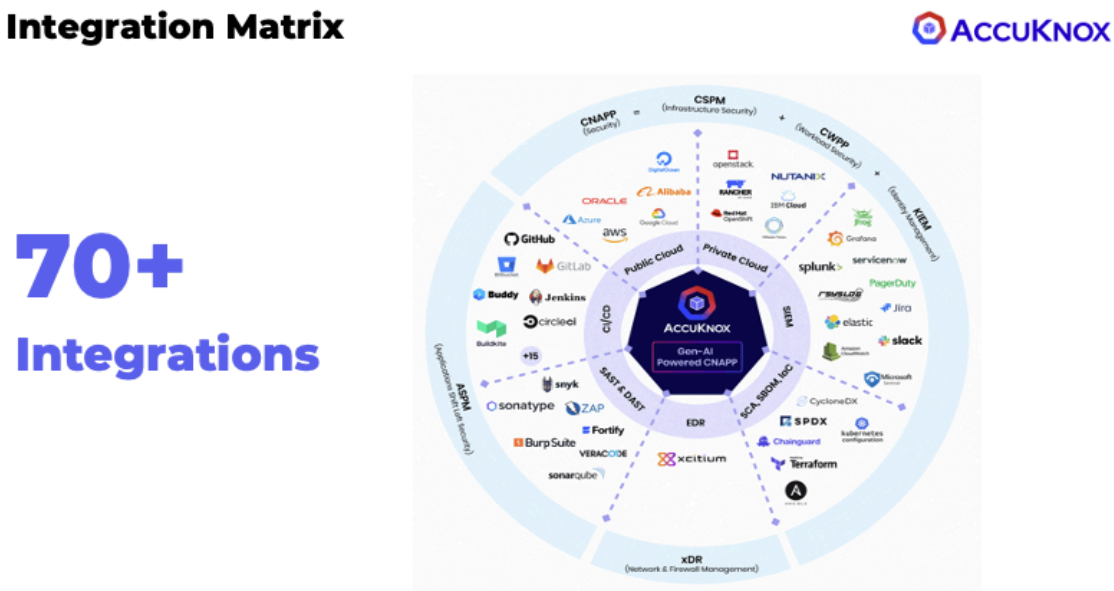
1. Playbook execution (on-prem or SaaS)
2. Report generated (assets/findings JSON)
3. Sent to control plane via Artifact API (token-based)
4. Saved in S3 + Celery task triggered
5. Celery pulls from S3 and parses
6. DB + Graph updated
7. UI fetches via Divy APIs

Rules Engine Architecture



- Parser emits events → Rules Engine evaluates
- Tenant-specific rule specs evaluated
- Actions (e.g., notifications, tickets) sent as Celery tasks
- Fully asynchronous, scalable via queues

Integrations Architecture



- **CLI-based**: TruffleHog, Sonarqube, Trivy, Zap, Kubebench
- **API-based**: Checkmarx, Nessus
- **SIEM**: One-way push (e.g., Splunk, Sentinel)

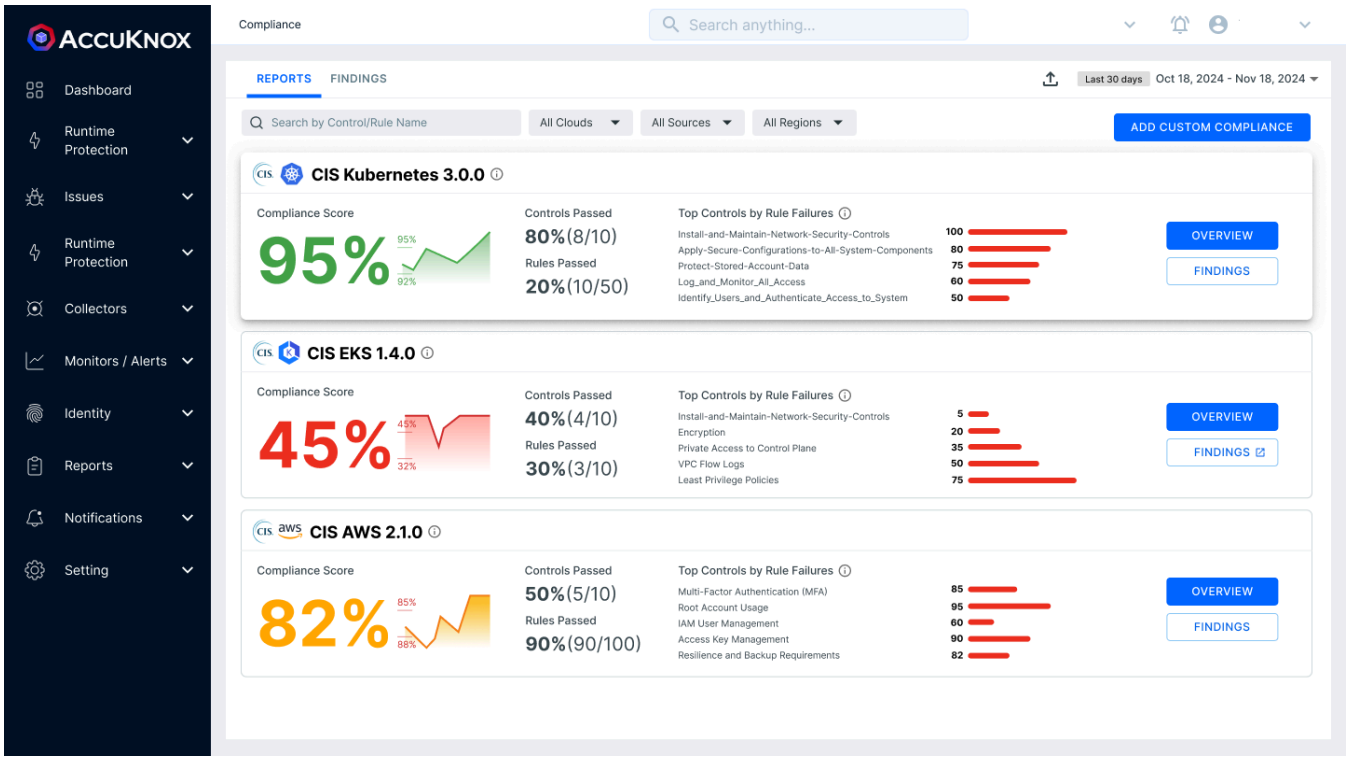
- **Ticketing**: Bidirectional (e.g., Jira, ServiceNow)

Integration Timelines

- CLI-based: 1 sprint
- API-based: 2–3 weeks
- SIEM: 1 sprint
- Ticketing: 3–5 sprints

Explore Integrations →

Compliance Frameworks



Supports over 30 regulatory standards, including:

- **General**: ISO 27001, PCI DSS, SOC2.
- **Industry-Specific**: HIPAA, GDPR.

Additional Resources

- Deployment Models
- Integrations Playbook
- Telemetry Logs
- On-Prem Installation Guide

Info

AccuKnox offers rapid protection for Kubernetes and other cloud workloads using Kernel Native Primitives like AppArmor, SELinux, and eBPF. For assistance in planning your cloud security strategy, feel free to reach out.

Getting Started With Technical Support

AccuKnox has active support teams across global regions. The Technical Support team is highly skilled in AccuKnox products and understands customer needs.

As a customer with AccuKnox Support, you're entitled to a number of predetermined technical support contacts who can help debug critical issues and provide solutions. These contacts must be specifically named individuals.

You can:

- Create support cases
- Search the AccuKnox Knowledge Base
- Review product documentation

Roles and Responsibilities

Role	Description
Customer	• Communicate business impacts of technical issues • Provide logs, debug data, diagnostic files, etc. • Respond timely to information or follow-up requests • Engage internal teams as needed • Have internet access for meetings

AccuKnox Solutions Engineer	• Understand business impact • Provide technical product expertise • Troubleshoot and resolve issues • Share timely status updates
AccuKnox Technical Support Manager	• Ensure high-level technical expertise in Support • Monitor critical issues
AccuKnox Customer Success Manager	• Understand customer requirements • Recommend matching AccuKnox solutions

Product Documentation

- 📘 AccuKnox Help Center
- 🎓 Certification & Training: On-demand and instructor-led sessions to enable your team

Email Support and Procedures

- Email: **support@accuknox.com**
- Or raise a support ticket via: AccuKnox Support Portal

Note:

- First-time users must sign up via Jira
- Try Incognito Mode if you face access issues
- Support responds within **<24 working hours**

Support Workflow

- Once a ticket is created, users can track the status via ticket ID

Priority Levels

Technical Priority	Description
P1 - Critical	Product is completely non-functional; critical business impact
P2 - High	Product is severely degraded; severe business impact

P3 - Medium	General errors; business still functional
P4 - Informational	Assistance or basic info; minimal/no business impact

🧠 Related article: *Technical Support Case Priorities* — visit the Knowledge Base for examples.

Case Information Required

Please have the following information ready when submitting a case:

1. Contact Name and Organization
2. Business Impact and project context
3. Affected Product
4. Priority Level
5. Relevant screenshots, logs, diagnostic files
6. Was it working before? When did it break? Any changes?
7. Error messages (if any)
8. Frequency and timing of the issue

Technical Support may ask for further info or coordinate with your technical team to isolate known issues.

Video Conferencing Options

- AccuKnox may initiate Zoom or Google Meet sessions.
- Sessions are scheduled for 30 minutes with a predefined agenda.
- If you're >5 minutes late or absent, the session may be rescheduled.
- Live troubleshooting will follow the session.

Case Resolution

A case is considered resolved when one of the following is provided:

- Official product behavior documentation
- A verified workaround
- A software update/patch
- A fix in documentation

Case Closure

A case is closed when:

- Customer confirms the resolution, or
- There's no response for a reasonable period

In rare cases (e.g., customer unresponsiveness or unprofessional behavior), AccuKnox may close the case independently.
Closed cases may be **reopened within 3 days**.

Resources

- 📄 On-Prem Deployment Guide
- 📘 Help Portal

FAQs

1. Can we engage on a messaging stream for continuous support?

Yes, we can create a **temporary Slack channel** for real-time communication.

2. What are the system requirements for On-Prem deployment?

Nodes	vCPUs	RAM (GB)	Disk (GB)
4	8	32	256
5	4	16	128

3. Is a completely air-gapped On-Prem environment supported?

✅ Yes, AccuKnox fully supports **air-gapped** environments.

4. How do upgrades work and how frequently are updates released?

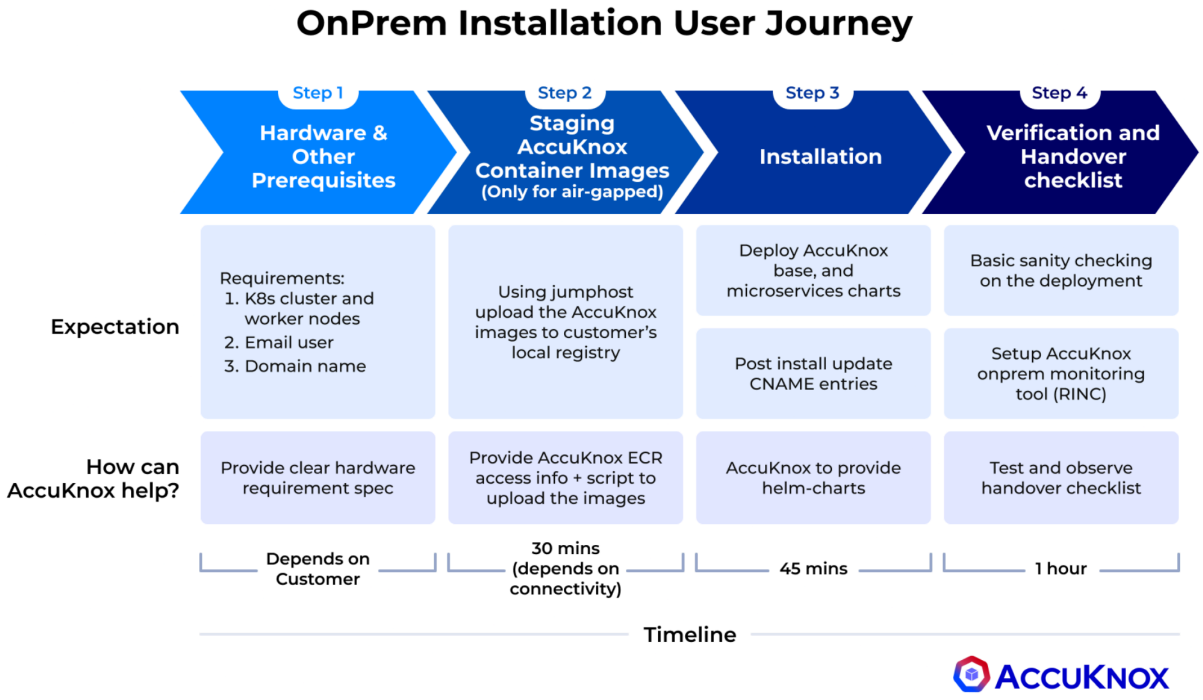
- Software updates are released **monthly**
- Latest package is shared with installation instructions
- AccuKnox Engineering/DevSecOps teams are available to assist if required

🔗 [Release Notes and FAQs]

AccuKnox OnPrem Deployment Guide

Onboarding Steps for AccuKnox

The onboarding process for AccuKnox's on-prem security solution consists of four key steps that the user must complete. Let's go through each step in a thorough, step-by-step manner:



Step 1: Hardware & Prerequisites

- Verify hardware, email user, and domain configurations.
- Ensure your environment meets all requirements.
- Time estimate: **Varies**, allocate sufficient time for review and adjustments.

Step 2: Staging AccuKnox Container Images *(For airgapped environments only)*

- Stage AccuKnox container images in the airgapped setup.
- Reconfirm hardware, email user, and domain requirements.
- Time estimate: **~1 hour**.

Step 3: Installation

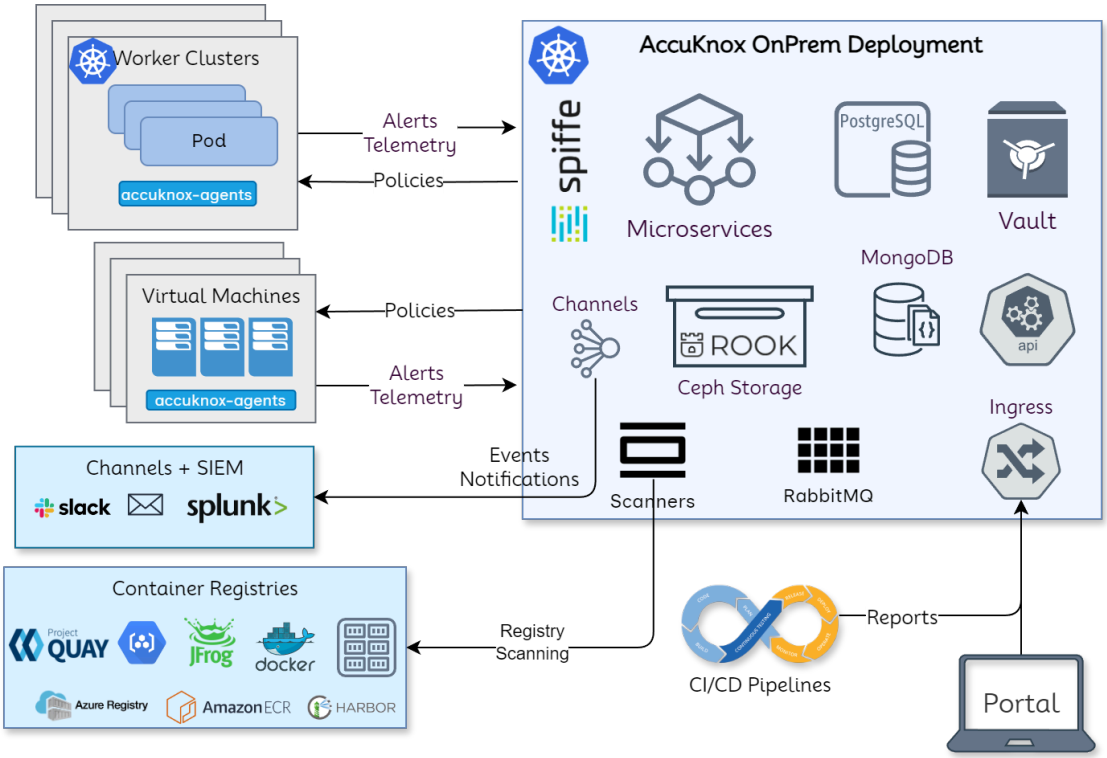
- Install the AccuKnox system within your environment.
- Ensure all prerequisites remain satisfied.
- Time estimate: **~45 minutes**.

Step 4: Verification/Validation

- Confirm all previous steps were completed successfully.
- Validate hardware, email user, and domain configurations.
- Time estimate: **~1 hour**.

AccuKnox onprem deployment is based on Kubernetes native architecture.

High-Level Architecture Overview



AccuKnox onprem deployment is based on Kubernetes native architecture.

AccuKnox OnPrem k8s components

Microservices

Microservices implement the API logic and provide the corresponding service endpoints. AccuKnox uses Golang-based microservices for handling streaming data (such as alerts and telemetry) and Python-based microservices for other control-plane services.

Databases

PostgreSQL is used as a relational database and MongoDB is used for storing JSON events such as alerts and telemetry. Ceph storage is used to keep periodic scanned reports and the Ceph storage is deployed and managed using the Rook storage operator.

Secrets Management

Within the on-prem setup, there are several cases where sensitive data and credentials have to be stored. Hashicorp's Vault is used to store internal (such as DB username/password) and user secrets (such as registry tokens). The authorization is

managed purely using the k8s native model of service accounts. Every microservice has its service account and uses its service account token automounted by k8s to authenticate and subsequently authorize access to the secrets.

Scaling

K8s native horizontal and vertical pod autoscaling is enabled for most microservices with upper limits for resource requirements.

AccuKnox-Agents

Agents need to be deployed in target k8s clusters and virtual machines that have to be secured at runtime and to get workload forensics. Agents use Linux native technologies such as eBPF for workload telemetry and LSMs (Linux Security Modules) for preventing attacks/unknown execution in the target workloads. The security policies are orchestrated from the AccuKnox onprem control plane. AccuKnox leverages SPIFFE/SPIRE for workload/node attestation and certificate provisioning. This ensures that the credentials are not hardcoded and automatically rotated. This also ensures that if the cluster/virtual machine has to be deboarded then the control lies with the AccuKnox control plane.

System Requirements

Worker Node Requirements

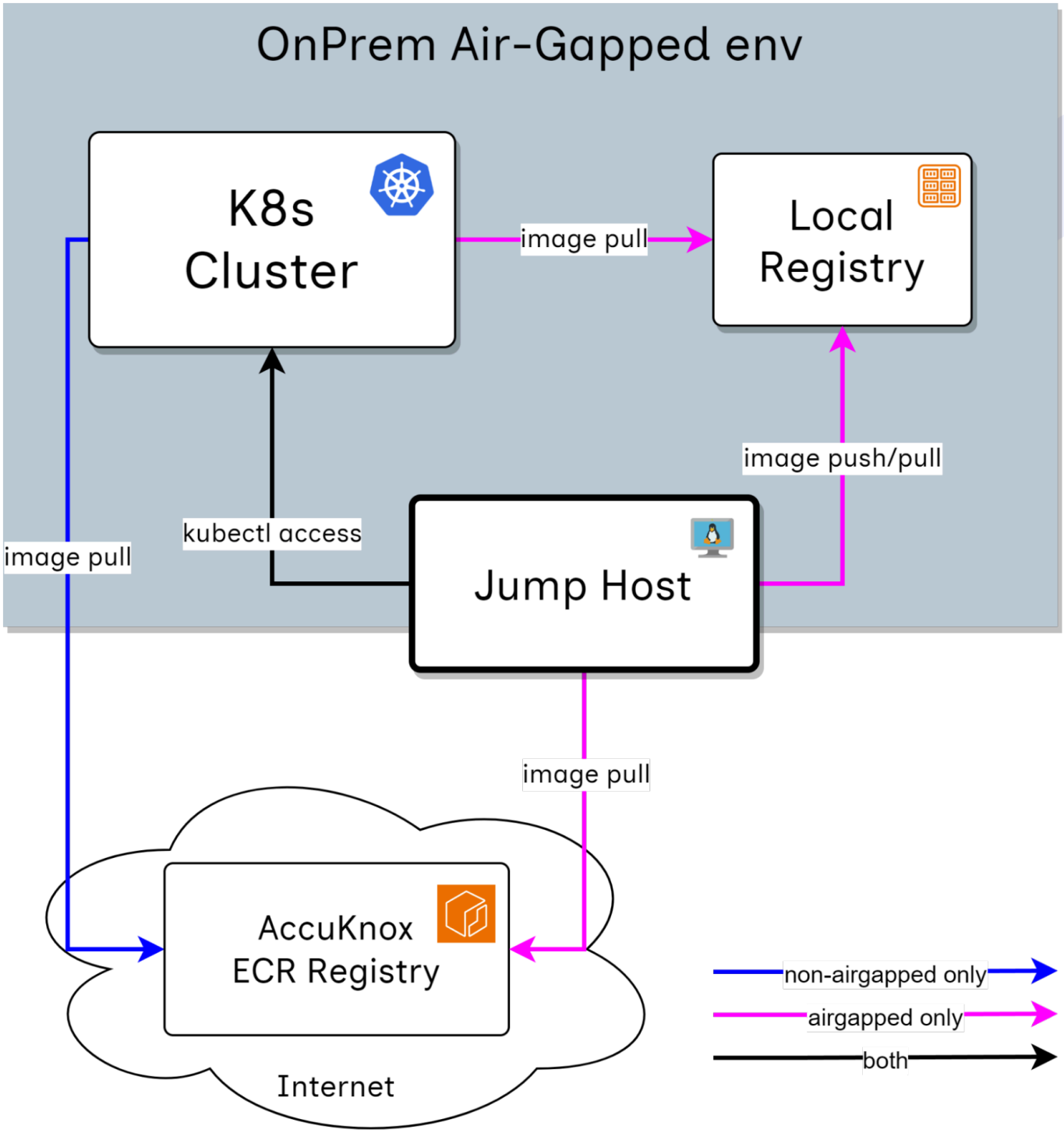
Nodes	vCPUs	RAM (GB)	Disk (GB)
6	4	16	256

Kubernetes Requirements

- **Ingress Controller (load balancers)**
    - For access to the application
- **Persistent Volumes (PV), provisioner/controller (block device/disks)**

    - Used as data storage for SQL, MongoDB, scanned artifacts
    - Other internal app usages
- **DNS CNAME provisioning**
    - Needed for application access & communication
    - Certs would use this CNAME so that address changes won't impact the cert validation
- **Email account configuration**
    - Need email username, and password
    - Used for user sign-in, password change, scan notification, sending reports

Jump Host



Jump Host Pre-requisites

Tool	Version	Install Command
jq	1.6	`apt install jq`
unzip	x.x	`apt install unzip`
yq	v4.40.x	`VERSION=v4.40.5 && BINARY=yq_linux_amd64 && wget https://github.com/mikefarah/yq/releases/download/${VERSION}/${BINARY}.tar.gz -O - | tar xz && mv ${BINARY} /usr/bin/yq`
helm	v3.x.x	`curl -s https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash`
kubectl	Supported by your k8s cluster	-
aws	v2	`curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip" && unzip awscliv2.zip && sudo ./aws/install --bin-dir /usr/local/bin --install-dir /usr/local/aws-cli --update`
docker	v20.xx	`apt install docker.io`

Storage	80GB	-

Installation Steps

- Onprem Deployment Installation Document (this document)
- Helm charts archive
- Kubectl and Helm tools are pre-requisite tools for using these helm charts

Use the following commands

```
tar xvf accuknox-helm-charts.tgz
cd Helm-charts
```

Use of Private/Local Container Registry (or air-gapped mode)

If you want to use your private/local registry as the exclusive source of images for the entire cluster, please install the accuknox-onprem-mgr component first.

Value	Description	Provider
registry.username	Registry User	Customer

registry.password	Registry Password	Customer
registry.address	The registry server address	Customer
ecr.user	Credential to pull images from AccuKnox registry	AccuKnox
ecr.password	Credential to pull images from AccuKnox registry	AccuKnox

```
cd airgapped-reg

# configure aws cli with AccuKnox  provided secrets
aws configure

# connect to docker Accuknox docker registry
aws ecr get-login-password --region us-east-2 | docker login --username AWS
--password-stdin 956994857092.dkr.ecr.us-east-2.amazonaws.com

# connect to airgapped registry
docker login <registry_address>

# upload images to private registry
./upload_images.sh <registry_address>
./upload_onboarding_images.sh <registry.address>

# upload helm charts to private registry
./upload_helm.sh <registry.address>

# create a namespace
MGR_NS="accuknox-onprem-mgr"
CERT_MGR_NS="cert-manager"
kubectl create ns $MGR_NS
kubectl create ns $CERT_MGR_NS
```

```
kubectl create secret docker-registry airgapped-reg     --docker-server=<registry.address>
--docker-username=<registry.username> --docker-password=<registry.password> -n
$MGR_NS

kubectl create secret docker-registry airgapped-reg --docker-server=<registry.address>
--docker-username=<registry.username> --docker-password=<registry.password> -n
$CERT_MGR_NS

# <registry_address> can include port as well

./install-certmanager.sh <registry_address>

./install-onprem-mgr.sh <registry_address>

kubectl apply -k .
kubectl apply -f onprem-mgr.yaml
```

Update the override-values.yaml

[ONLY FOR air-gapped/private registry ENVIRONMENT]: Set global.onprem.airgapped to true in override-values.yaml file.

Before you start

- set your domain name in the override values by changing by your domain
- set your ssl preferences in the override values by changing the ssl block
- If you wish to bring in your own MongoDB, PostgreSQL, NFS share or S3, disable global.postgres.airgapped and global.mongodb.enabled rookceph.enabled in override-values.yaml.

If the environment is OpenShift then set:

```
global:
  platform: "openshift"`
```

If environment is airgapped or using private registry make ssl.certmanager.install:"false"

```
ssl:
  certmanager:
    install: false
```

Auto-generated self-signed certificate

We auto generate the needed self signed certificates for the client. To enabled this option, the ssl section the override values file should be set as follow:

```
ssl:
  selfsigned: true
  customcerts: false
```

Certificate signed by a known authority

The client provides a certificate signed by a known signing authority To enable this option, the ssl section the override values file should be set as follow:

```
ssl:
  selfsigned: false
  customcerts: true
```

Self-signed certificates (provided by the customer)

The client provides a self signed certificate. To enabled this option, the ssl section the override values file should be set as follow:

```
ssl:
  selfsigned: true
  customcerts: true
```

AccuKnox installation package will contain override-values.yaml file that contains installation-specific options to be configured.

1. override to your domain
2. set your ssl preferences in the override values by changing the ssl block.

Install AccuKnox base dependencies

```
kubectl create namespace accuknox-chart
helm upgrade --install -n accuknox-chart accuknox-base accuknox-base-chart
--create-namespace -f override-values.yaml
```

IMPORTANT

Some resources deployed in the above step require some time to provision. If the user executes the next command without waiting for the proper provisioning of the previous command the installation may break and will need to start over.

Run the below script to make sure that the provisioning was done succesfully.

```
while true
do
  status=$(kubectl get cephcluster -n accuknox-ceph  rook-ceph
-o=jsonpath='{.status.phase}')
  [[$(echo $status | grep -v Ready | wc -l) -eq 0]] && echo "You can proceed" && break
 echo "wait for initialization"
  sleep 1
done
```

Install AccuKnox pre-chart

IMPORTANT

Contact your AccuKnox representative to acquire the credentials for `ecr.user` and `ecr.password` values.

Value	Description	Provider
email.user	Email user will send signup invites, reports, etc.	Customer
email.password	Email Password	Customer

email.host	The Email server address	Customer
email.from	The Email sender address (noreply@domain.com)	Customer
ecr.user	Credential to pull images from AccuKnox registry	AccuKnox
ecr.password	Credential to pull images from AccuKnox registry	AccuKnox
global.externalServices.postgres.user	Postgres username, if using an external DB	Customer
global.externalServices.postgres.password	Postgres password, if using an external DB	Customer
global.externalServices.postgres.host	Postgres host, if using an external DB	Customer
global.externalServices.mongo.user	Mongodb username, if using an external DB	Customer

global.externalServices.mongo.password	Mongodb password, if using an external DB	Customer
global.externalServices.mongo.host	Mongodb host, if using an external DB	Customer
global.externalServices.nfs.server	NFS server address	Customer
global.externalServices.s3.host	S3 datastore host	Customer
global.externalServices.s3.port	S3 datastore port	Customer
global.externalServices.s3.accessKey	S3 access key	Customer
global.externalServices.s3.secretKey	S3 secret access key	Customer
global.externalServices.s3.bucket	S3 bucket name	Customer

```
helm upgrade  --install  -n  accuknox-chart accuknox-pre pre-chart  --create-namespace -f
override-values.yaml --set global.email.from="" --set global.email.user="" --set
global.email.password="" --set global.email.host="" --set ecr.user="" --set ecr.password=""
```

Or, if using an external PostgreSQL or Mongo DB,

```
helm upgrade accuknox-pre pre-chart \
  --install \
  -namespace accuknox-chart \
  --create-namespace \
  -values override-values.yaml \
  --set global.email.user="" \
  --set global.email.password="" \
  --set global.email.host="" \
  --set ecr.user="" \
  --set ecr.password="" \
  --set global.externalServices.postgres.user="" \
  --set global.externalServices.postgres.password="" \
  --set global.externalServices.postgres.host="" \
  --set global.externalServices.mongo.user="" \
  --set global.externalServices.mongo.password="" \
  --set global.externalServices.mongo.host=""
```

Install AccuKnox microservices chart

Value	Description	Provider
email.user	Email user will send signup invites, reports, etc.	Customer
email.password	Email Password	Customer

email.host	The Email server address	Customer
email.from	The Email sender address (e.g., noreply@domain.com)	Customer

```
helm upgrade --install -n accuknox-chart accuknox-microservice accuknox-microservice-chart --set global.email.user="" --set global.email.from="" --set global.email.password="" --set global.email.host="" --create-namespace -f override-values.yaml
```

DNS Mapping

Run the following script to generate the records you should add to your DNS zone.

```
./generate_dns_entries.sh
```

Installing certificates

Certificates signed by known authority

```
./install_certs.sh <certificate_path> <certificate_key_path> <ca_path>
```

Self-signed certificates (provided by customer)

Install nginx ingress (if any other self-managed Kubernetes)

1. Install the nginx ingress chart

```
cd airgapped-reg/addons
```

```
helm upgrade --install ingress-nginx ingress-nginx \
    --repo https://kubernetes.github.io/ingress-nginx \
    --namespace ingress-nginx --create-namespace \
    --version 4.11.2 -f ingress-nginx.yaml
```

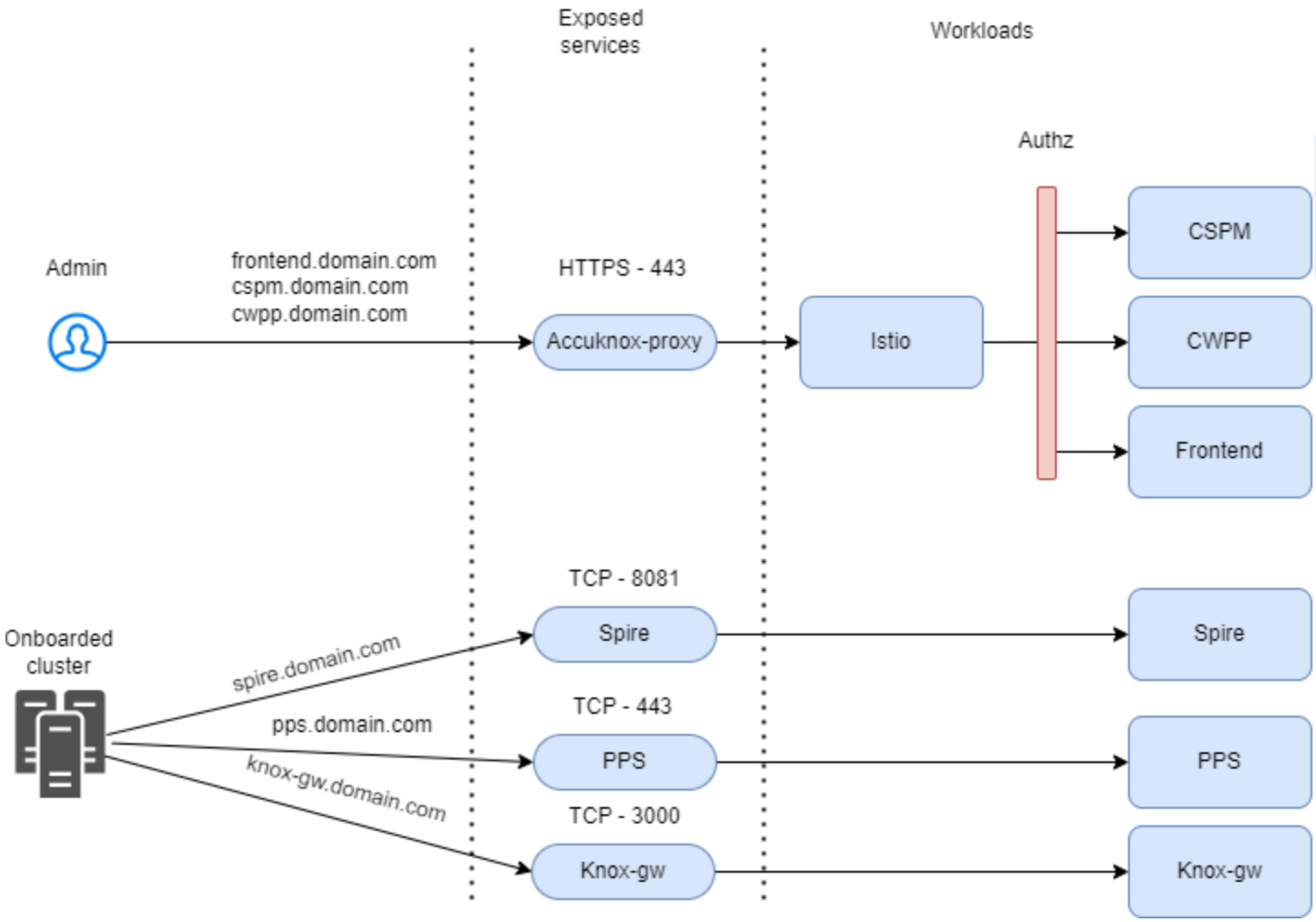
1. Update the domains in ingress.yaml and apply it

```
kubectl apply -f ingress.yaml
```

Verification of installation

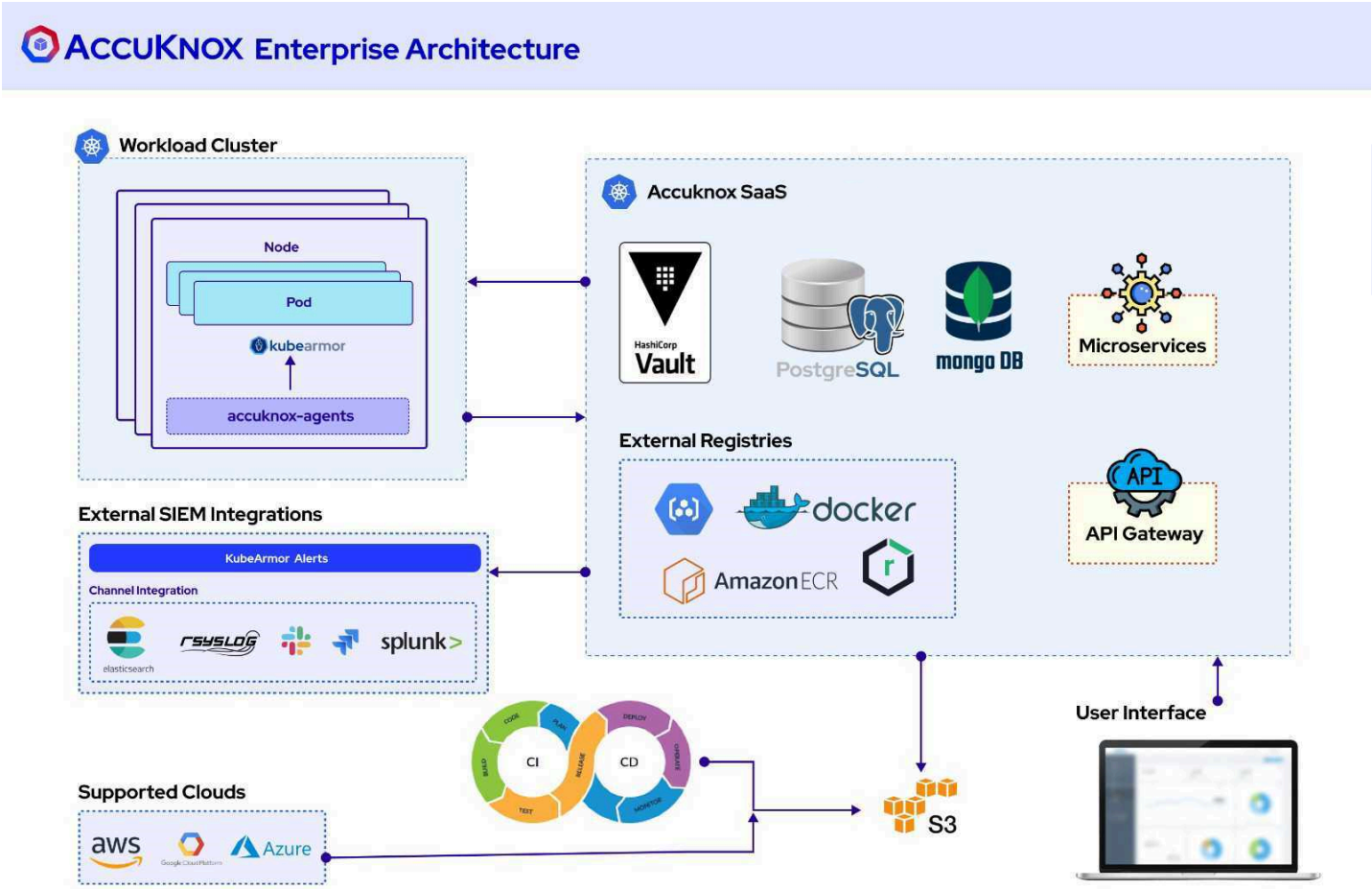
After successful installation, you should be able to access the following URLs:

- https://frontend.<your-domain.com>/ — Access the **Sign-in page**.
- https://cspm.<your-domain.com>/admin/ — Access the **CSPM Admin page**.
- https://cwpp.<your-domain.com>/cm/ — Access the **CWPP Configuration Management page**.



Runtime Security Prerequisites

In SaaS model of deployment the AccuKnox CNAPP will be hosted in our cloud environment and the agents deployed on the workloads will connect with the SaaS.



AccuKnox Agents

Deployments	Deployment Type
KubeArmor	DaemonSet

Shared Informer Agent	Deployment
Feeder Service	Deployment
Policy Enforcement	Deployment
Discovery Engine Agent	Deployment

- It is assumed that the user has some basic familiarity with Kubernetes, kubectl and helm. It also assumes that you are familiar with the AccuKnox opensource tool workflow. If you're new to AccuKnox itself, refer first to opensource installation
- It is recommended to have the following configured before onboarding:
    a. Kubectl
    b. Helm

Pre-requisites

Minimum Resource required

Deployments	Resource Usage	Ports	Connection Type	AccuKnox Endpoint
KubeArmor	CPU: 200 m, Memory: 200 Mi	-	-	-
Agents Operator	CPU: 50 m, Memory: 50 Mi	8081, 9090	Outbound	*.accuknox.com:8081 -→ SPIRE Access *.accuknox.com:9090 -→ SPIRE Health Check
Discovery Engine	CPU: 200 m, Memory: 200 Mi	-	-	-
Shared Informer Agent	CPU: 20 m, Memory: 50 Mi	3000	Outbound	*.accuknox.com:3000 -→ knox-gateway
Feeder Service	CPU: 50 m, Memory: 100 Mi	3000	Outbound	*.accuknox.com:3000 -→ knox-gateway
Policy Enforcement	CPU: 10 m, Memory: 20 Mi	443	Outbound	*.accuknox.com:443 -→ Policy Provider Service

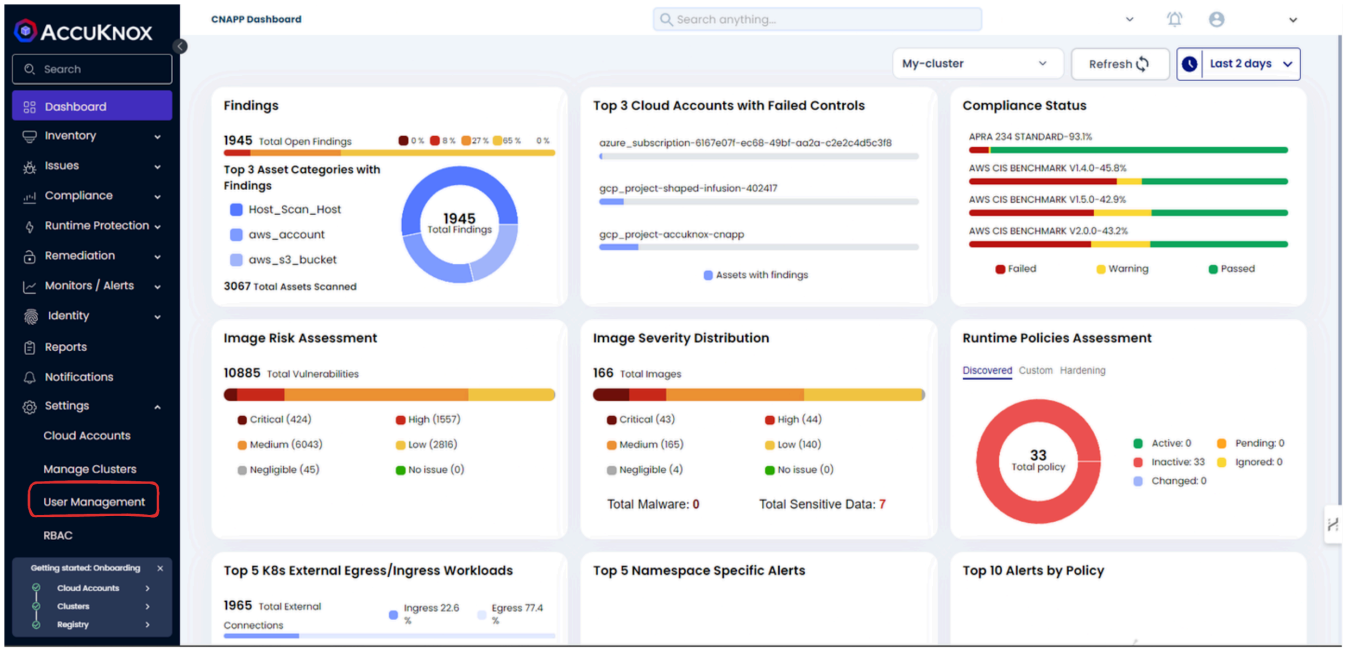
- These ports need to be allowed through firewall.

SSO Login Guide

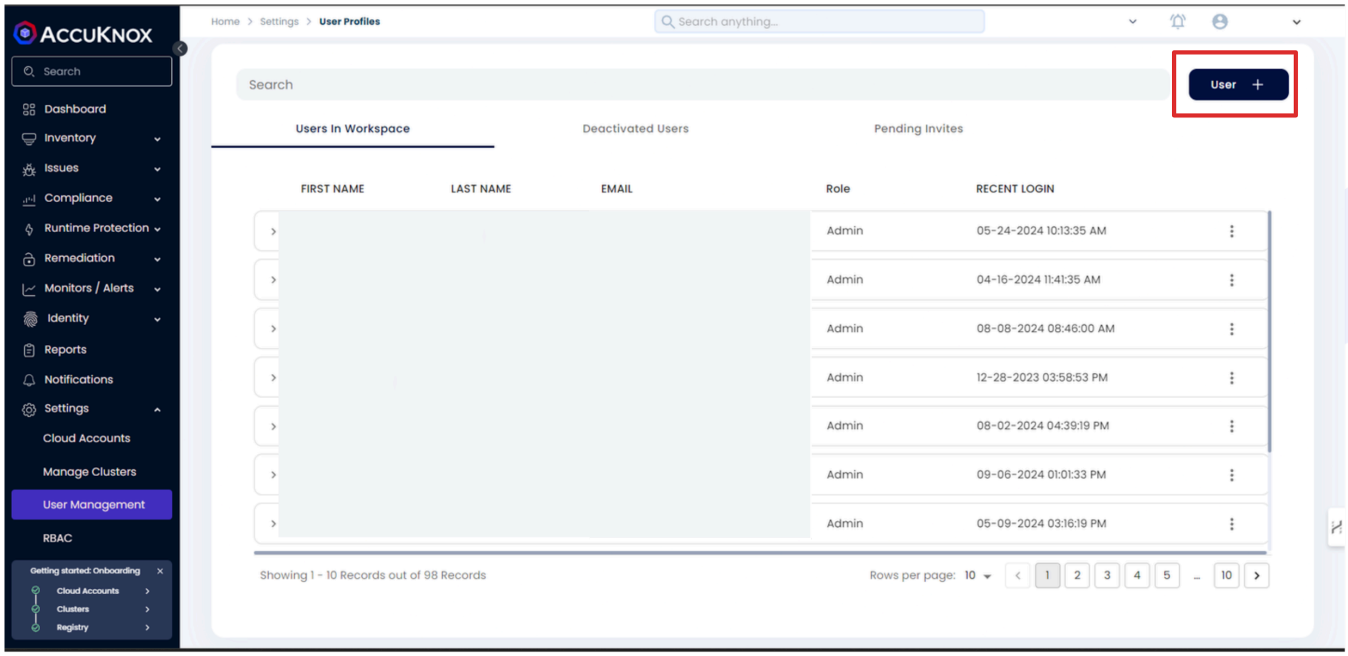
This guide covers the complete process from inviting a new user to logging in with SSO.

1. Inviting a New User

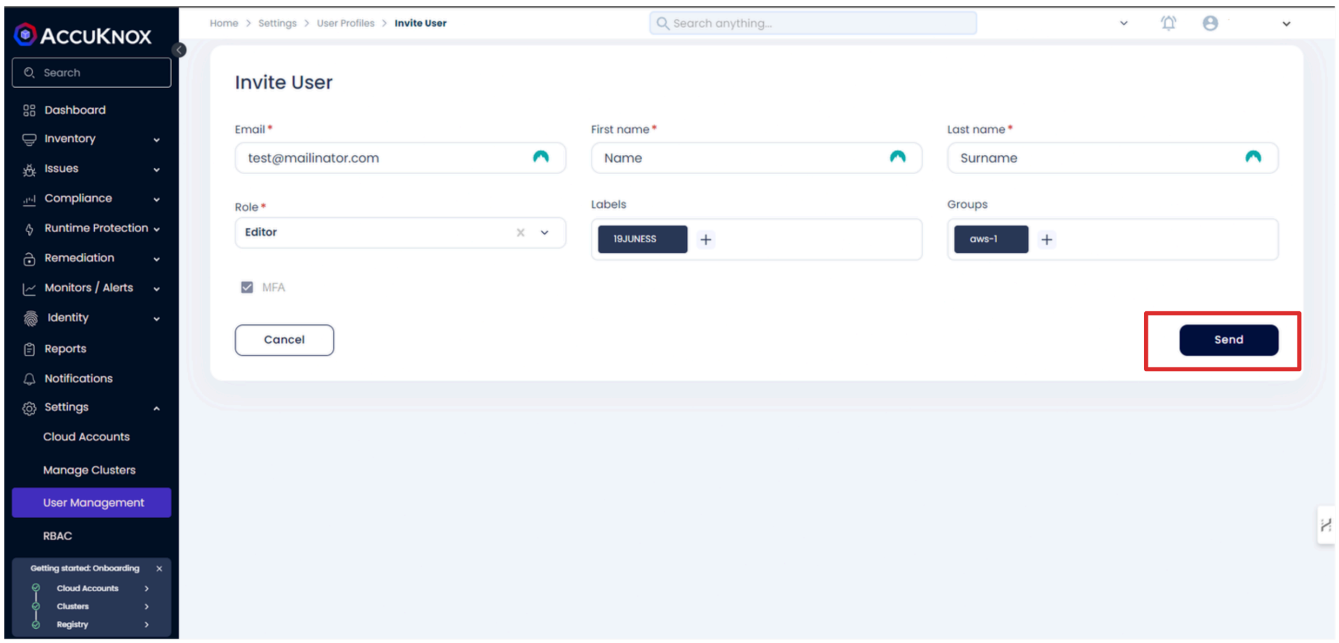
Log in to your AccuKnox dashboard.



Navigate to "User Management" in the left sidebar menu. Click the "User +" button in the top right corner of the Users page.



In the "Invite User" form, fill out the following details and hit send.

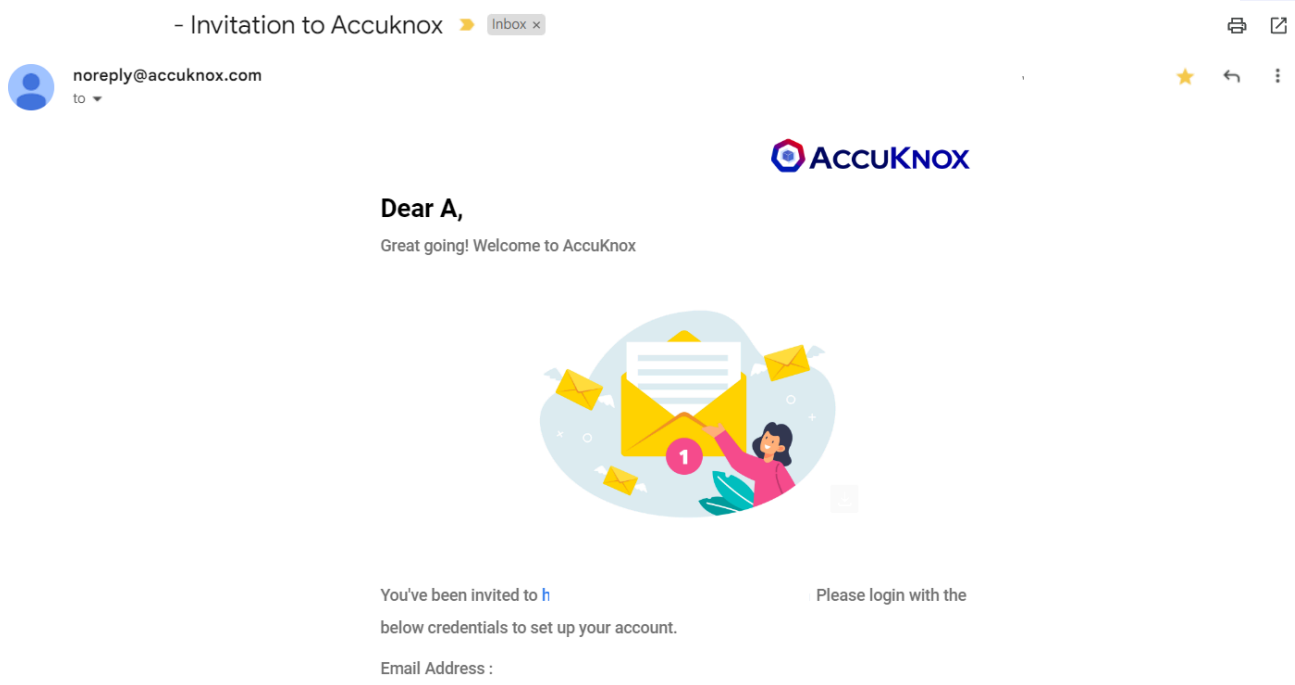


Note

You can view pending invitations in the "Pending Invites" tab on the Users page. You can resend or revoke invitations from this tab. Viewing all permissions of a user is possible via the main tab.

2. User Receives Invitation

The invited user will receive containing a link to accept the invitation and set up their account if they haven't already done so.



3. User Login Options

Users can log in to AccuKnox using two methods:

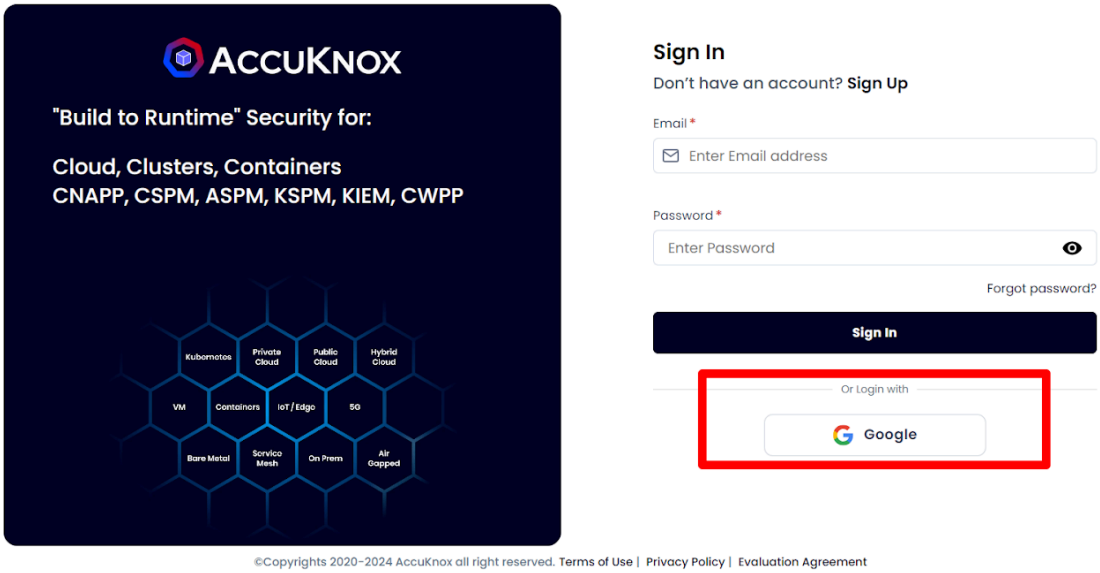
Option A: Traditional Login

1. Go to the AccuKnox login page.
2. Enter the email address and password.
3. Click "Sign In".

Note

This requires you to use the MFA (multi-factor authentication) code if it was enabled during the invitation process. MFA is required for every sign-in attempt.

Option B: Single Sign-On (SSO) with Google



1. Go to the AccuKnox login page.
2. Look for "Or login with" at the bottom of the form.
3. Click on the "Google" button.
4. If not already signed in to Google, enter Google account credentials.
5. Grant any necessary permissions for AccuKnox.

Note

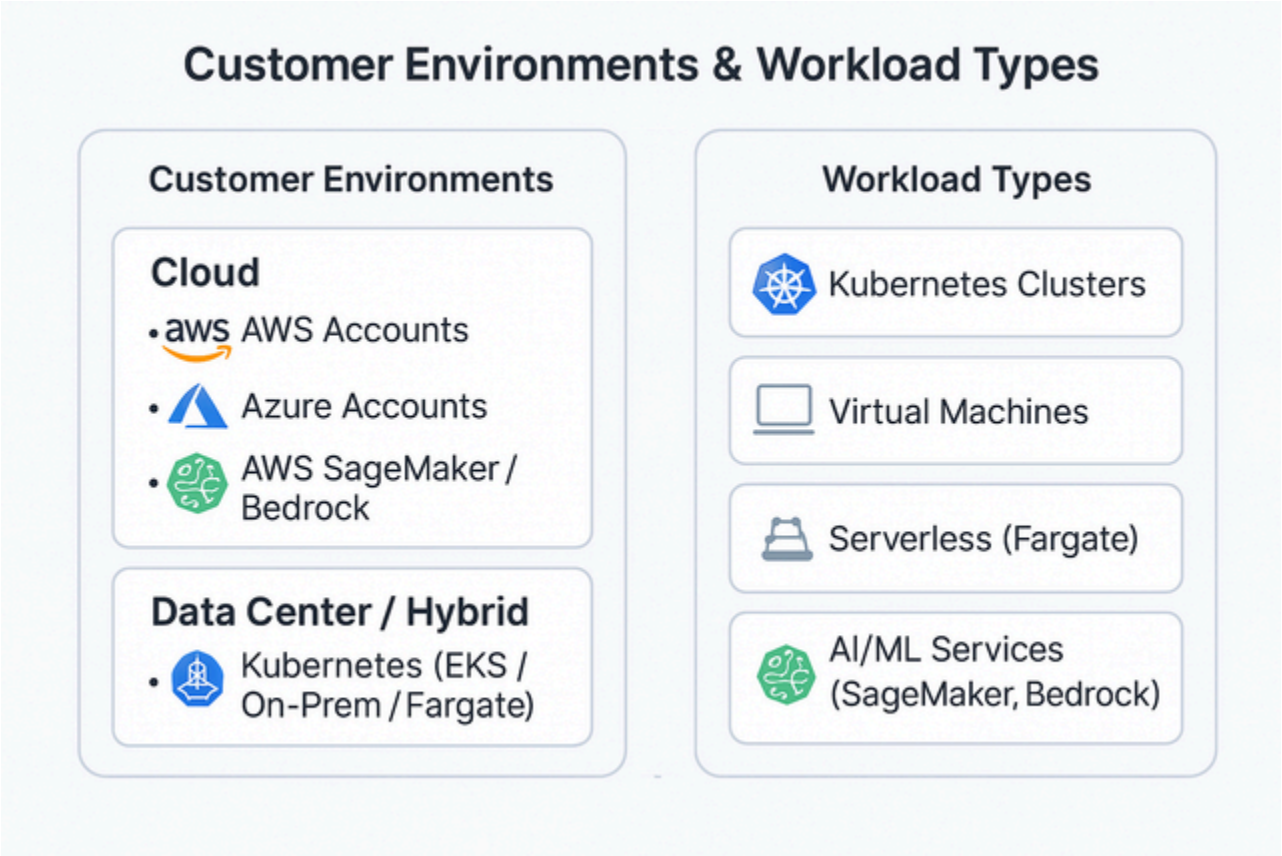
If you are already signed in to Google, you will be automatically logged in to AccuKnox. No need for MFA in this case.

Notes

- SSO is currently only supported for Google accounts.
- Users must be invited with their Gmail address to use Google SSO.

- For the best experience, use the same email address for invitation and login.
- If you encounter any issues, contact your AccuKnox administrator or support team.
- Emails with + modifiers (e.g., test+stable@gmail.com or example+solutions@gmail.com) are not supported for SSO. Please use a base email address.

Onboarding Assets – High-Level Overview



Customer Environments

Cloud:

- AWS Accounts
- Azure Accounts
- AWS SageMaker / Bedrock

Data Center / Hybrid:

- Kubernetes Clusters (EKS / On-Prem / Fargate)
- Virtual Machines (EC2 / On-Prem)

Workload Types:

- K8s Clusters
- Virtual Machines
- Serverless (Fargate)
- AI/ML Services (SageMaker, Bedrock)

Security and Telemetry Flow:

- Agentless scan initiated from SaaS
- CNAPP control plane processes telemetry
- Alerts and detections sent to SIEM

Cloud Onboarding Options

- Fully Agentless Mode
- Account/Subscription Onboarding:
- CloudFormation (recommended)
- Terraform
- Manual
- AWS Organization Unit Onboarding:
- Using cross-account tenant roles

Kubernetes – AWS EKS / On-Prem / Fargate

Risk Assessment

- CIS Benchmarks
- Misconfigurations
- KIEM Policies
- Agentless methods:
- Remote scanning via `kubeconfig`
- Kubernetes job-based scanning

Runtime Security & Hardening

- Helm-based installation
- In-cluster image scanning:
- Operator and job-based deployment via Helm

Fargate Runtime

- Supported via sidecar model
- Deployable using Helm or Kubernetes manifests

Virtual Machines – EC2 / On-Prem

- Misconfiguration scanning via cloud account onboarding (agentless)
- Risk assessment / STIGs scanning requires lightweight VM agent

---

Container Registry

SaaS-Based Scanning

- Registry onboarded via control plane
- Credentials: Username + API Token

On-Prem Scanning

- Requires AccuKnox collector deployed on VM
- Local scanning of registries enabled

AI/ML Workloads – SageMaker / Bedrock

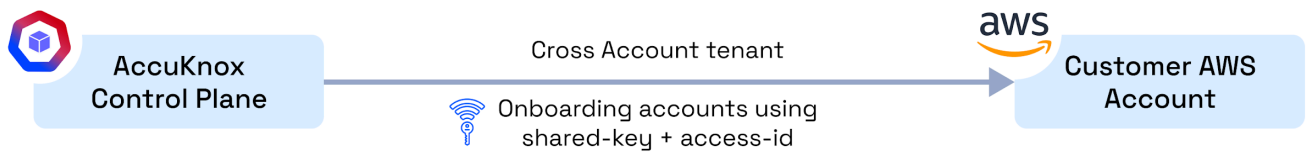
- Fully agentless
- Selectable during cloud account onboarding:
- General Cloud Assets
- General Cloud + AI/ML Assets

Deployment References

- Separate detailed documentation provided for Helm charts, job configurations, and onboarding automation (CloudFormation, Terraform).

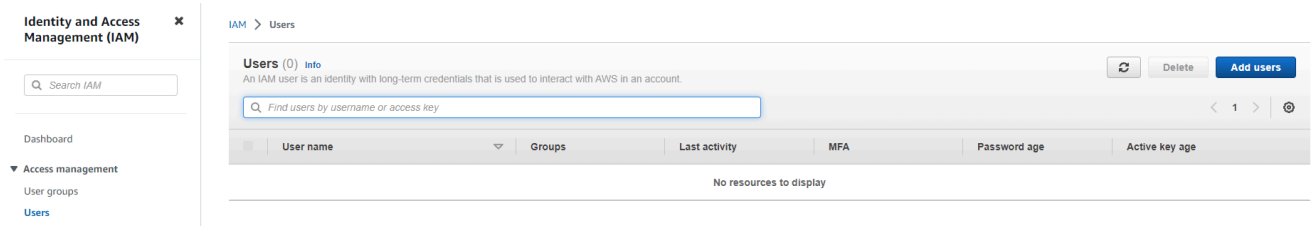
CSPM Pre-requisite for AWS

When the AccuKnox control plane is hosted in a cloud environment, scanning is performed using Cloud account Readonly Access permissions.

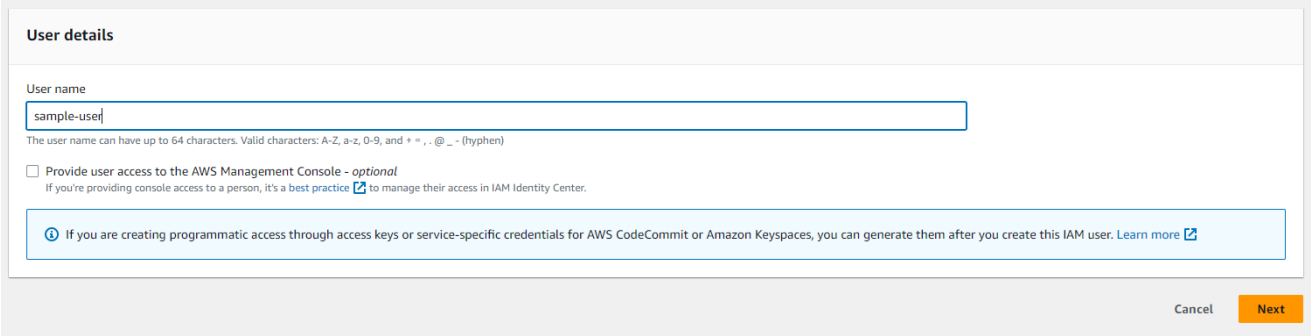


AWS onboarding requires creation of an IAM user. Please follow the following steps to provide a user with appropriate read access:

Step 1: Navigate to IAM → Users and click on Add Users



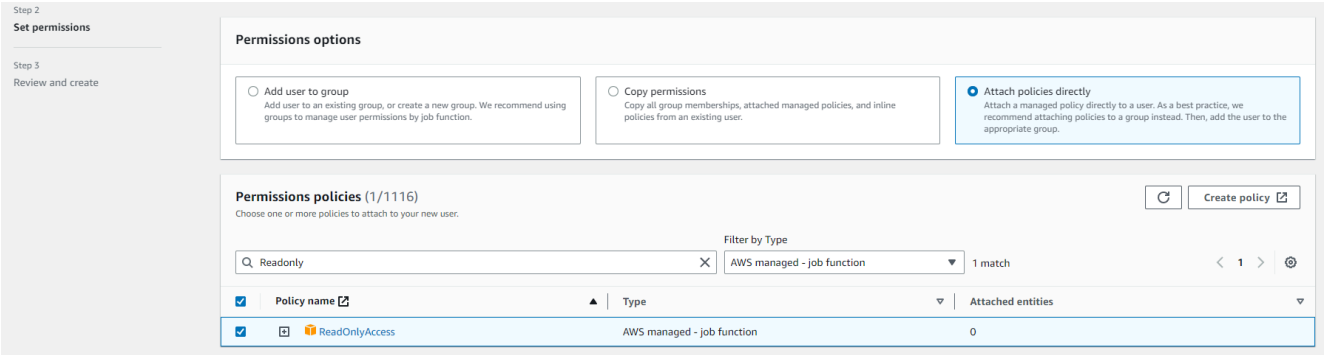
Step 2: Give a username to identify the user



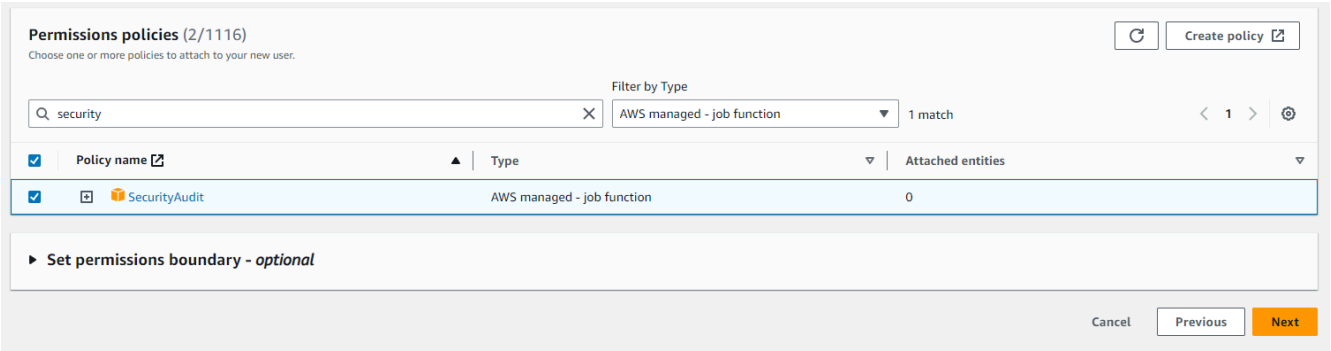
Step 3: In the "Set Permissions" screen:

a. Select "Attach policies directly"

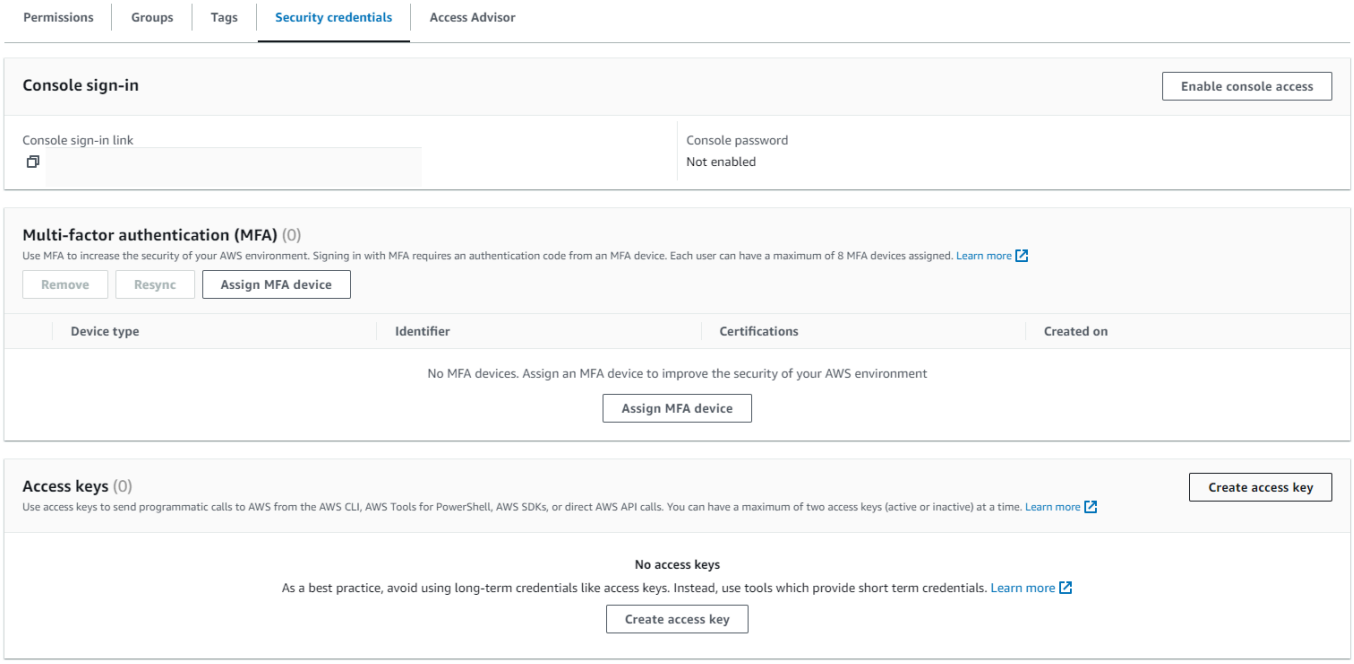
b. Search "ReadOnly", Filter by Type: "AWS managed - job function" and select the policy



c. Search "SecurityAudit", Filter by Type: "AWS managed - job function" and select the policy



Step 4: Finish creating the user. Click on the newly created user and create the Access key and Secret Key from the Security Credentials tab to be used in the AccuKnox panel



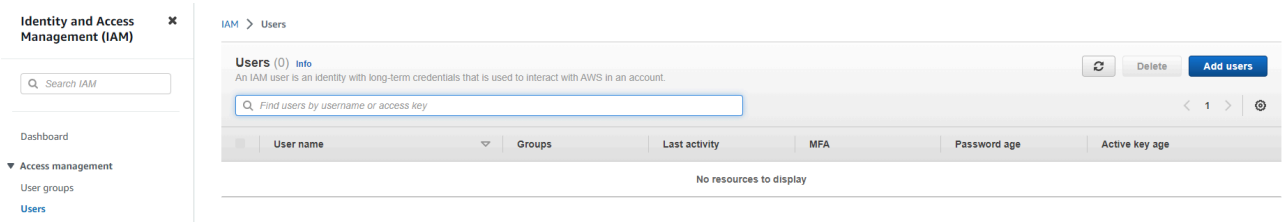
AWS Account onboarding

In this section we can find the steps to onboard an AWS cloud account to the AccuKnox SaaS platform.

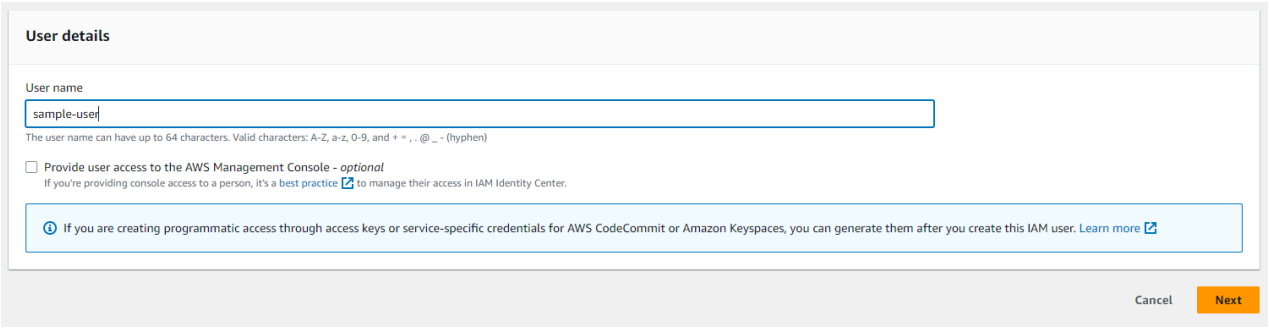
AWS IAM User Creation

Please follow the following steps to provide a user with appropriate read access:

Step 1: Navigate to IAM → Users and click on Add Users



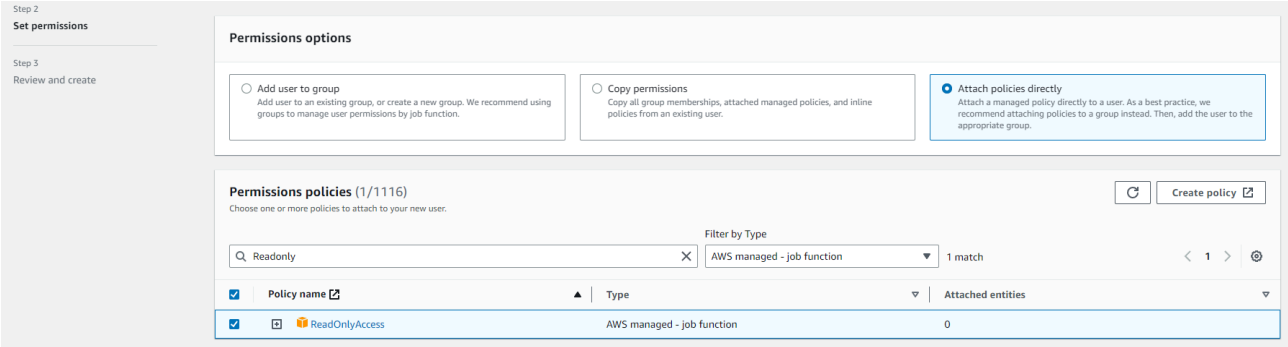
Step 2: Give a username to identify the user



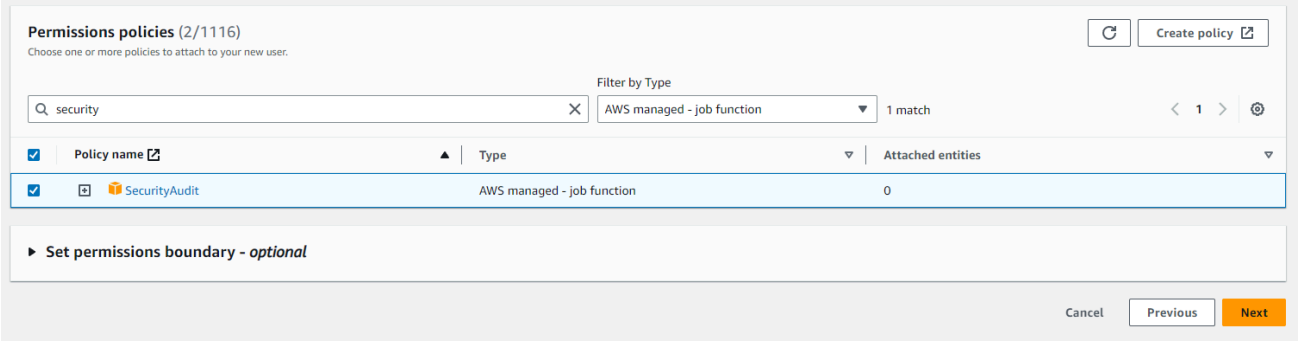
Step 3: In the "Set Permissions" screen:

a. Select "Attach policies directly"

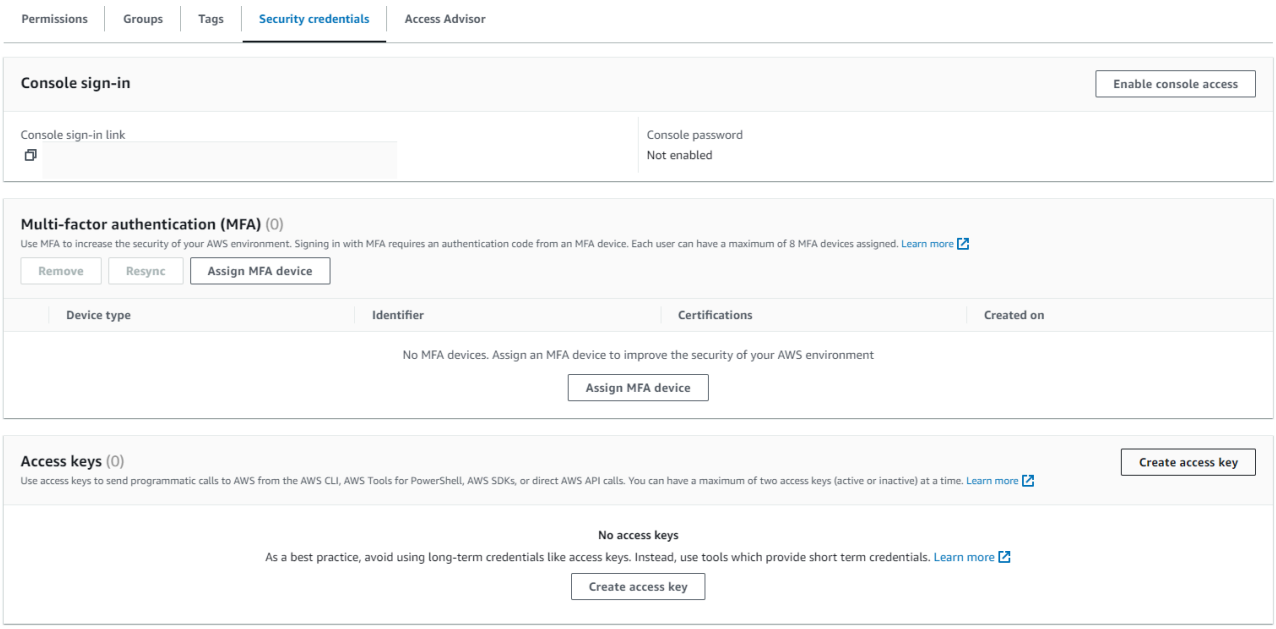
b. Search "ReadOnly", Filter by Type: "AWS managed - job function" and select the policy



c. Search "SecurityAudit", Filter by Type: "AWS managed - job function" and select the policy



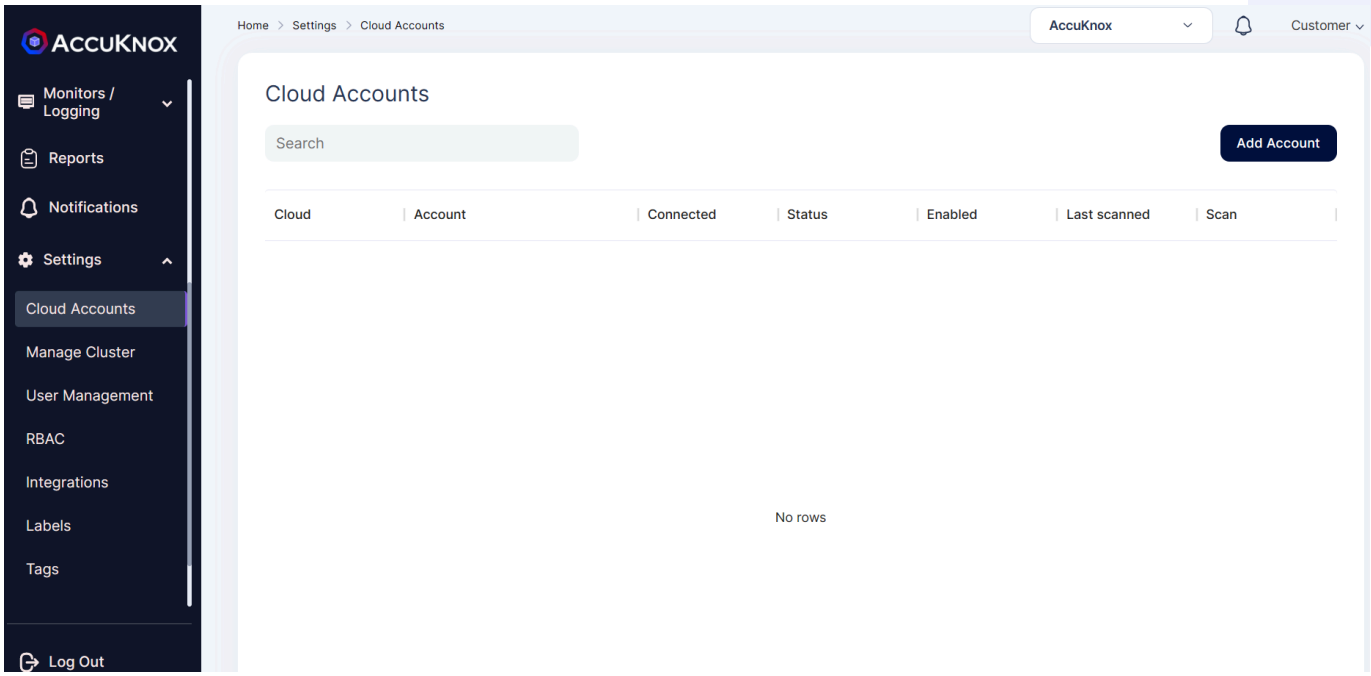
Step 4: Finish creating the user. Click on the newly created user and create the Access key and Secret Key from the Security Credentials tab to be used in the AccuKnox panel



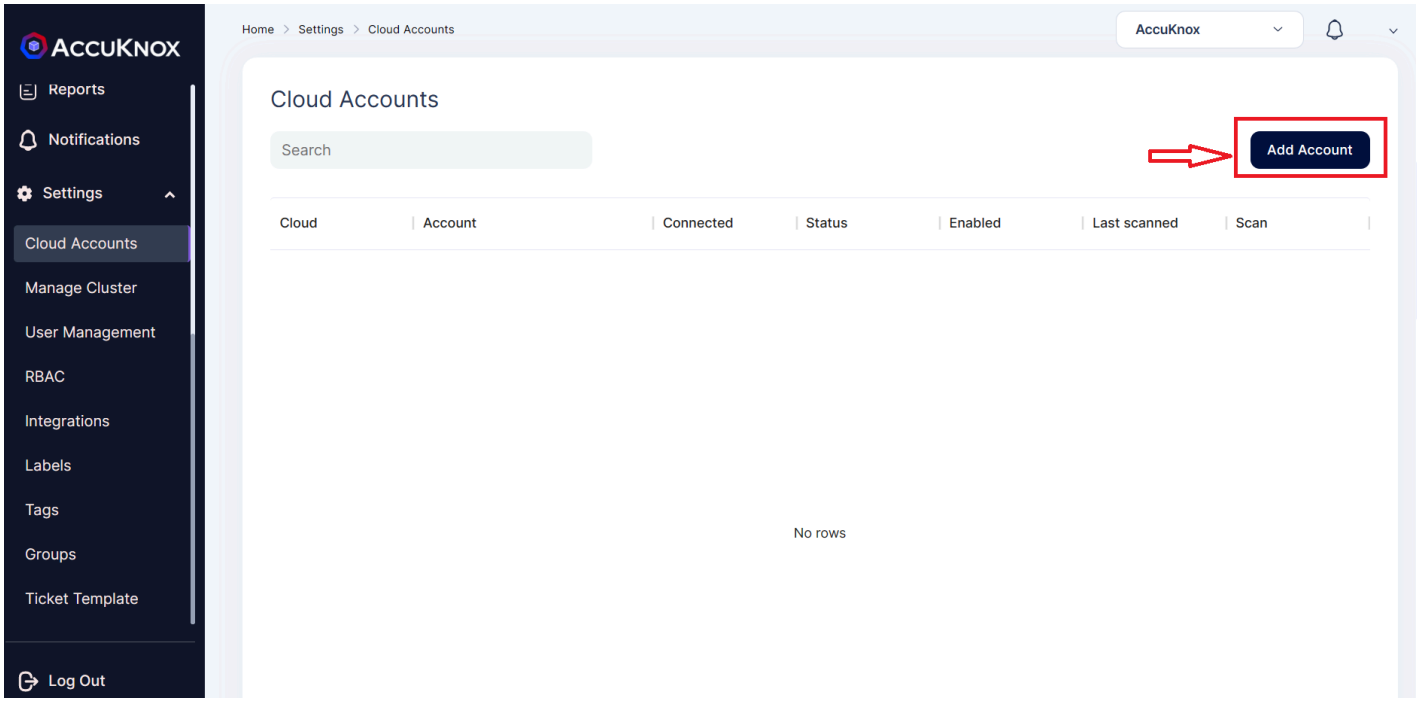
AWS Onboarding

In this example we are onboarding AWS account using the Access Keys method.

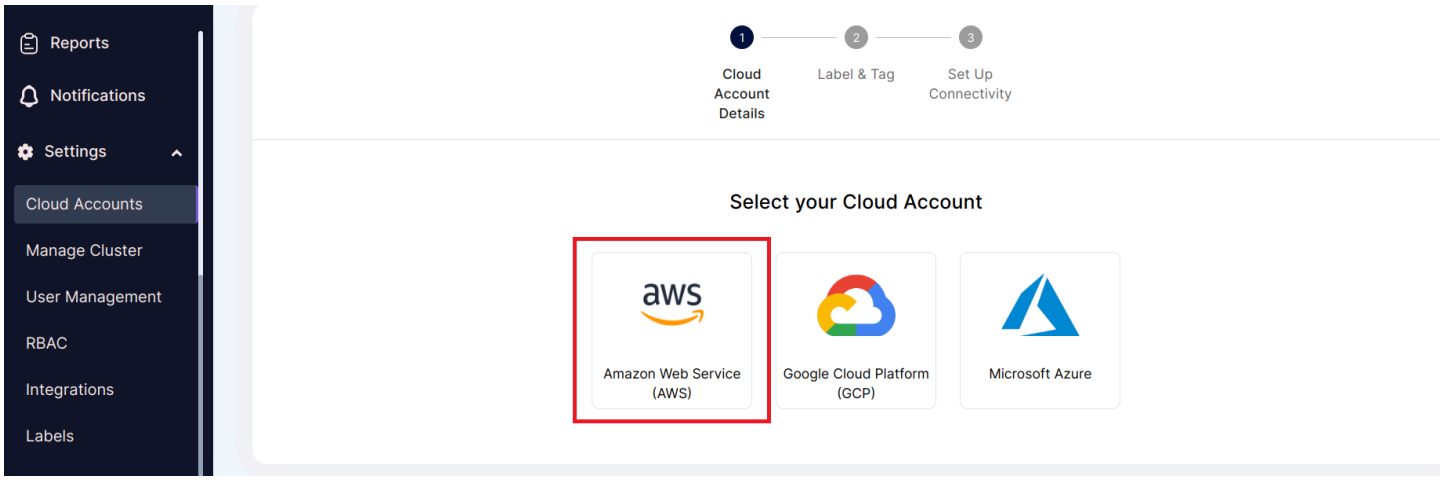
Step 1: To onboard Cloud Account Navigate to *Settings→cloud Accounts*



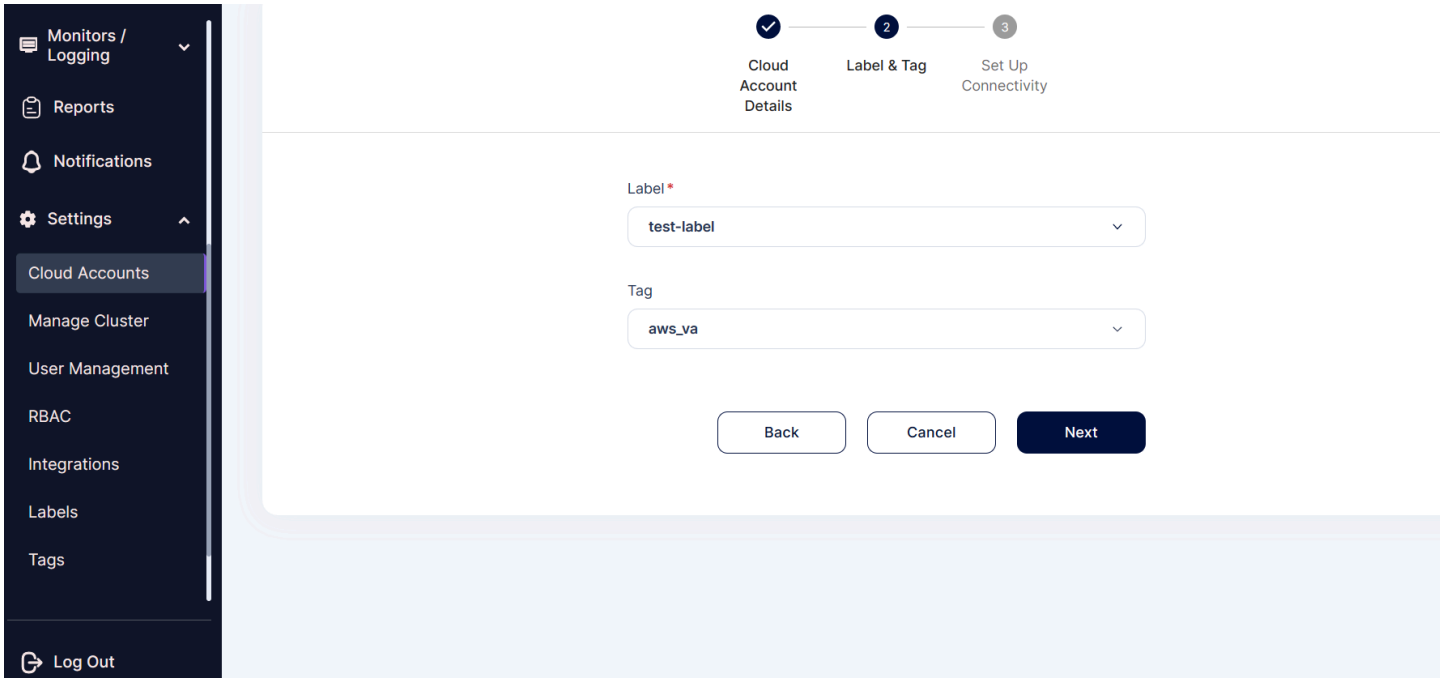
Step 2: In the Cloud Account Page select *Add Account* option



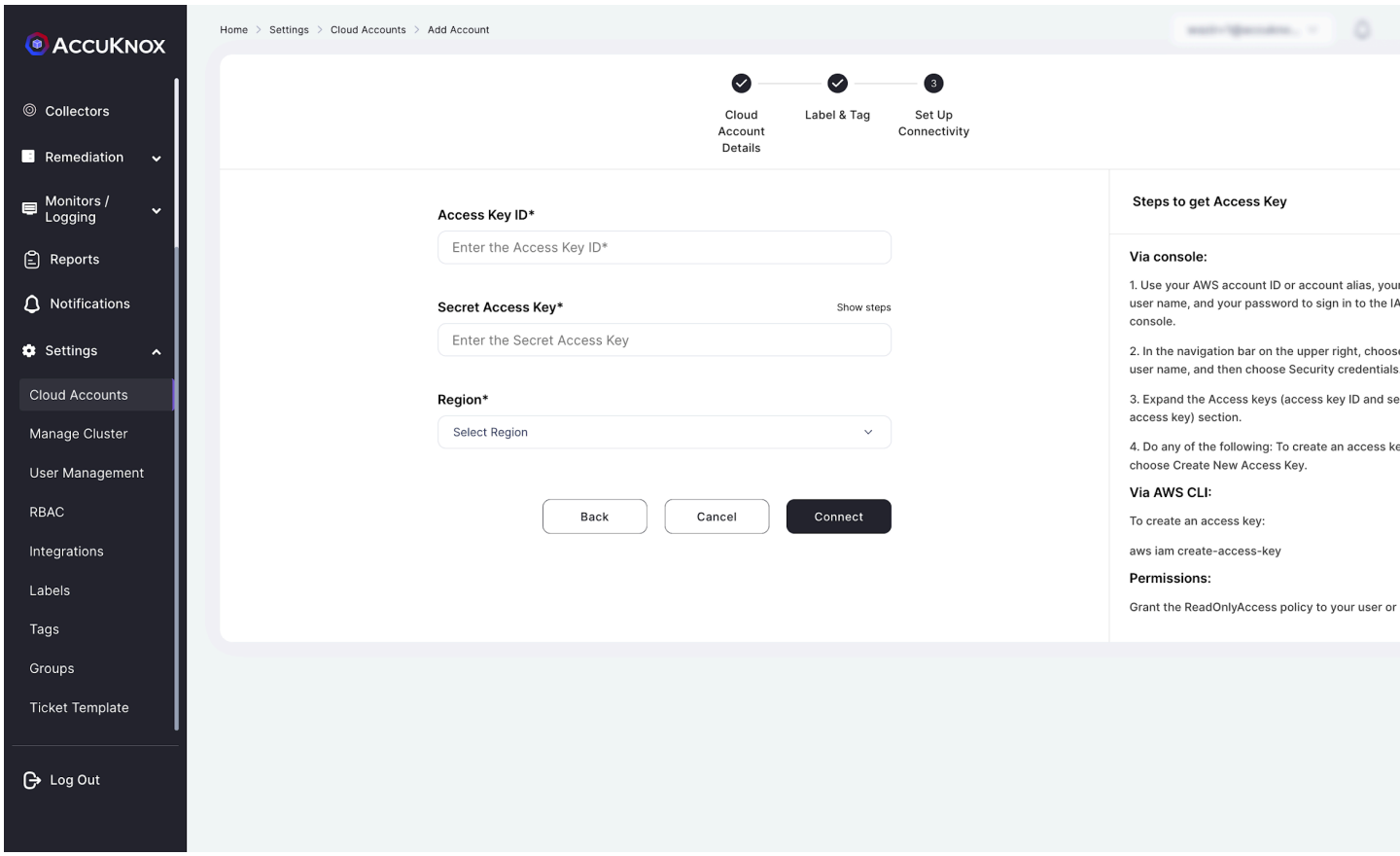
Step 3: Select the AWS option



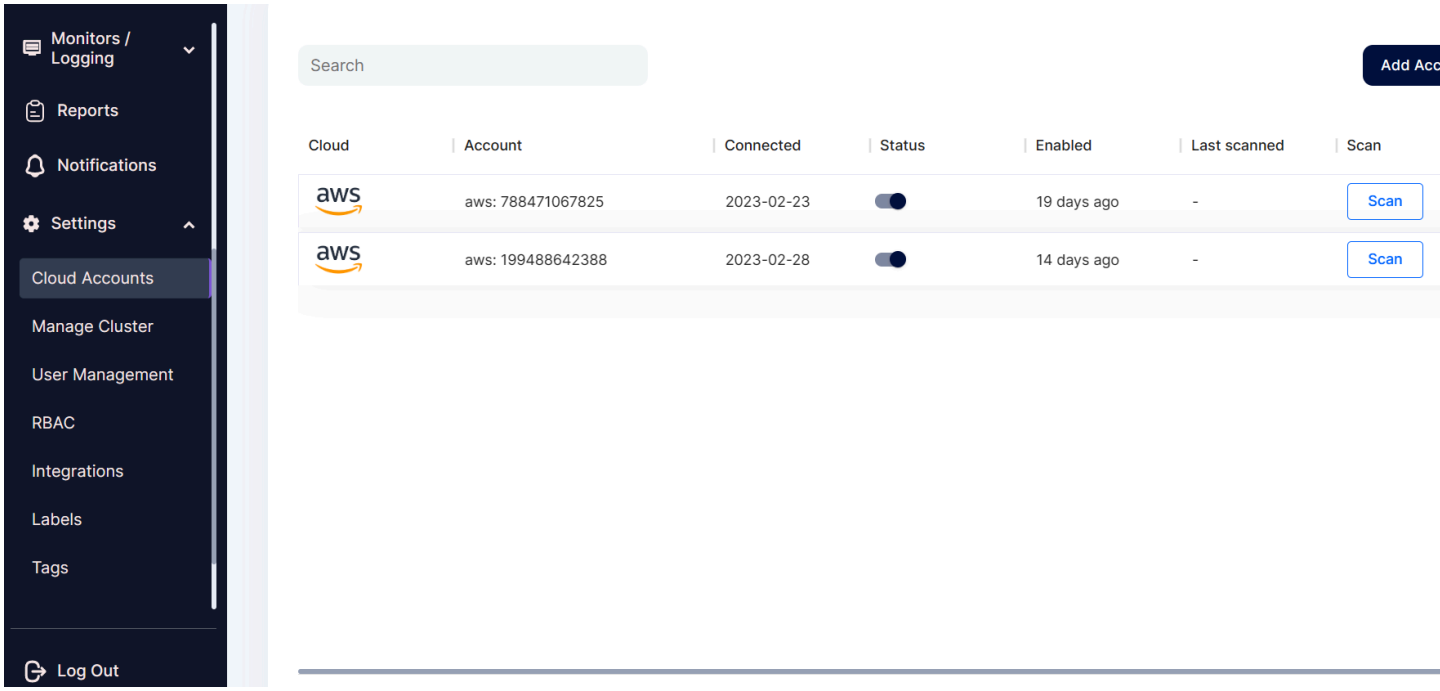
Step 4: In the next Screen select the labels and Tags field from the dropdown Menu.



Step 5: After giving labels and Tag in the Next Screen Provide the AWS account's Access Key and Secret Access Key ID and Select the Region of the AWS account.



Step 6: AWS account is added to the AccuKnox using Access Key Method. We can see the onboarded cloud account by navigating to Settings→cloud Accounts option.



Onboarding AWS Organization Accounts to AccuKnox

Managing security across multiple AWS accounts is complex. **AWS Organizations** simplifies this by grouping accounts under one structure. **AccuKnox** enhances this by enabling organization-level onboarding—removing the need to add accounts individually. This ensures centralized visibility, consistent policy enforcement, and automatic coverage for new accounts.

This guide explains how to onboard your **AWS Organization root account** to AccuKnox.

Prerequisites

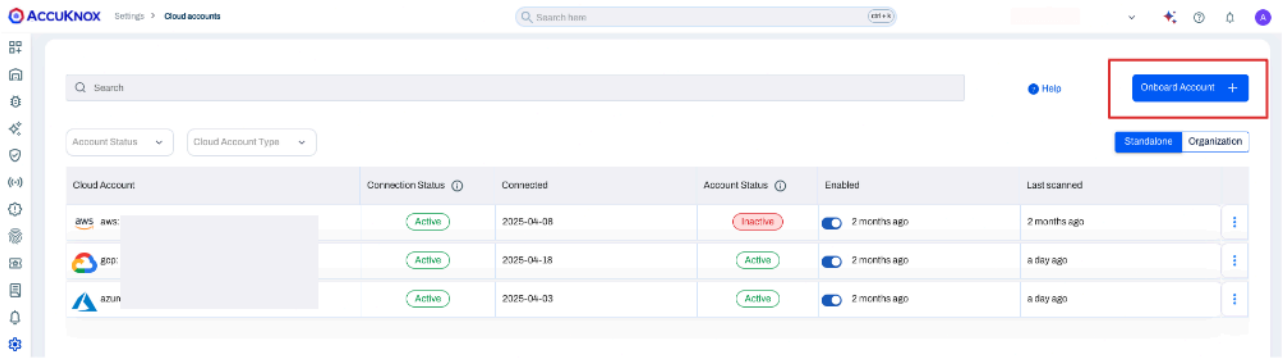
- You must have administrative access to your AWS Management Account and have permissions to deploy CloudFormation Stackset across the Organization.
- You need the AWS Organization ID of your root organization.

Step-by-Step Onboarding Process

Follow these steps to connect your AWS Organization to AccuKnox:

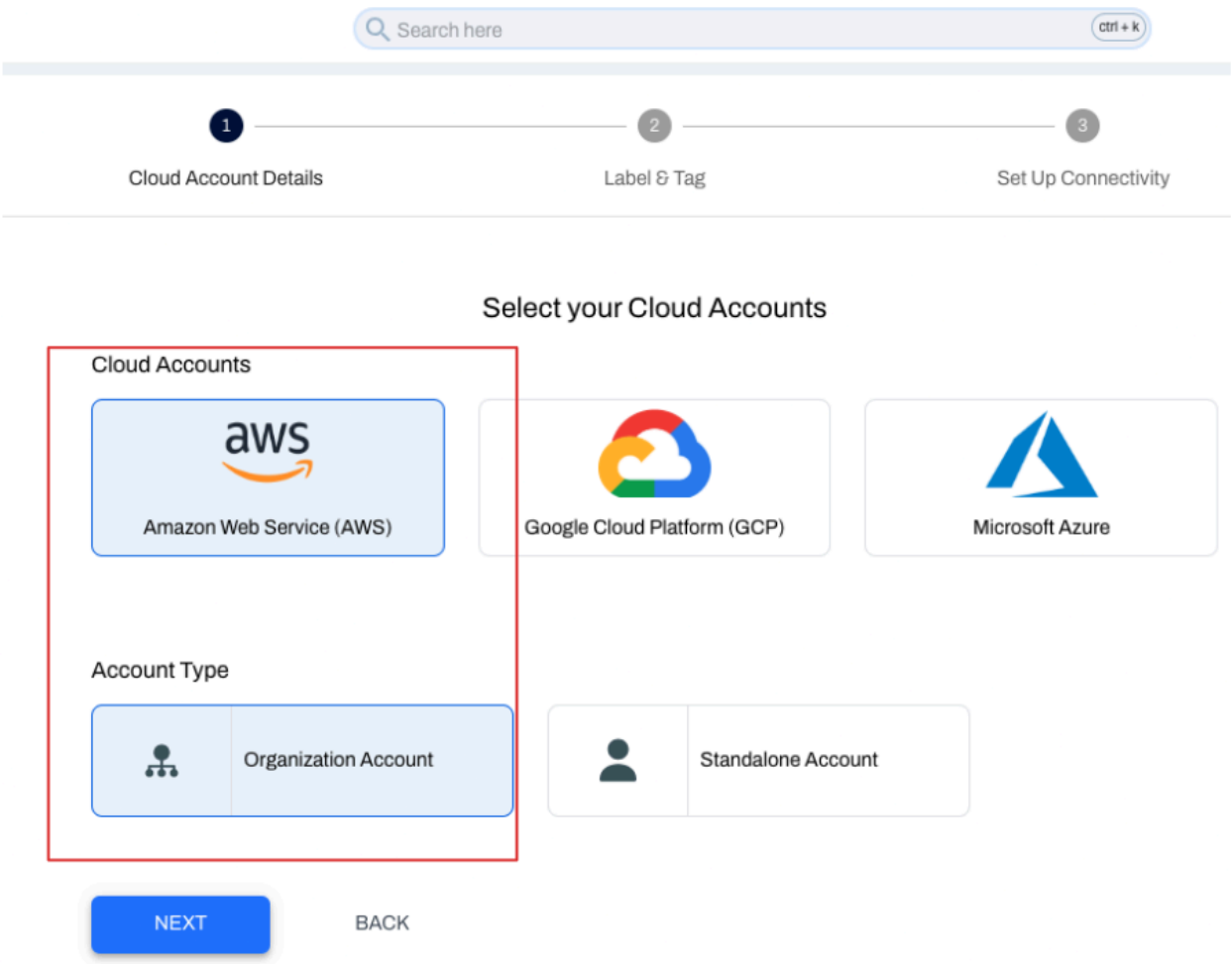
1. Initiate Account Onboarding

In the AccuKnox platform, navigate to **Cloud Security** → **Cloud Accounts** from the left-hand navigation menu. Select the **Organization** button, and then select **Onboard Account**.



2. Configure Organization Account Type and Labels

Select **Organization Account** as the account type.

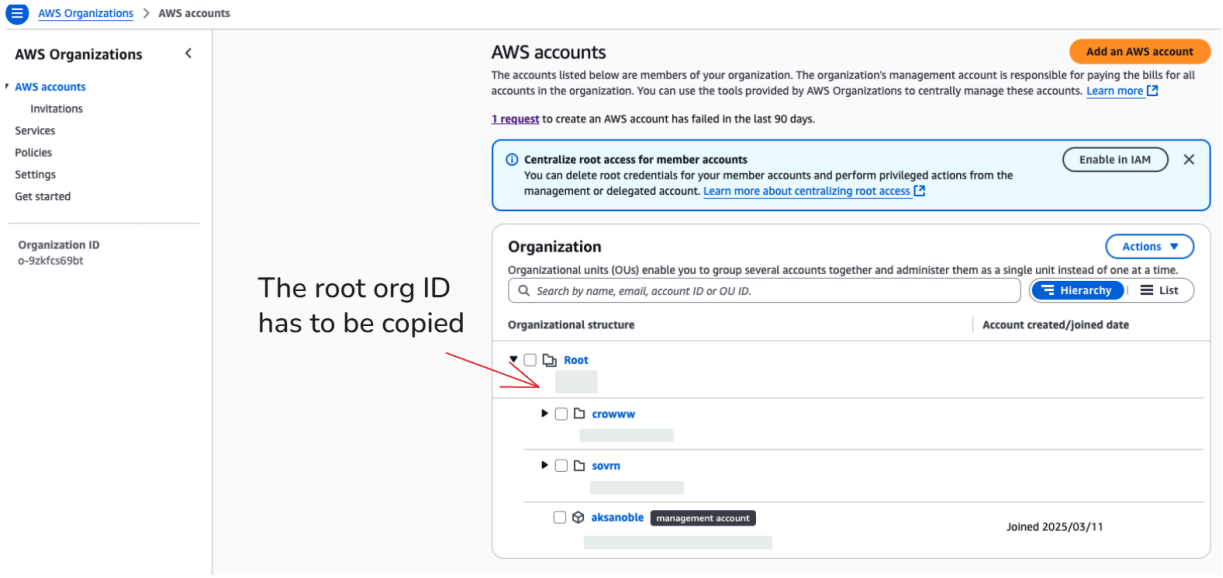


Next, select existing labels or create new ones to associate with all assets that will be discovered within this AWS Organization.

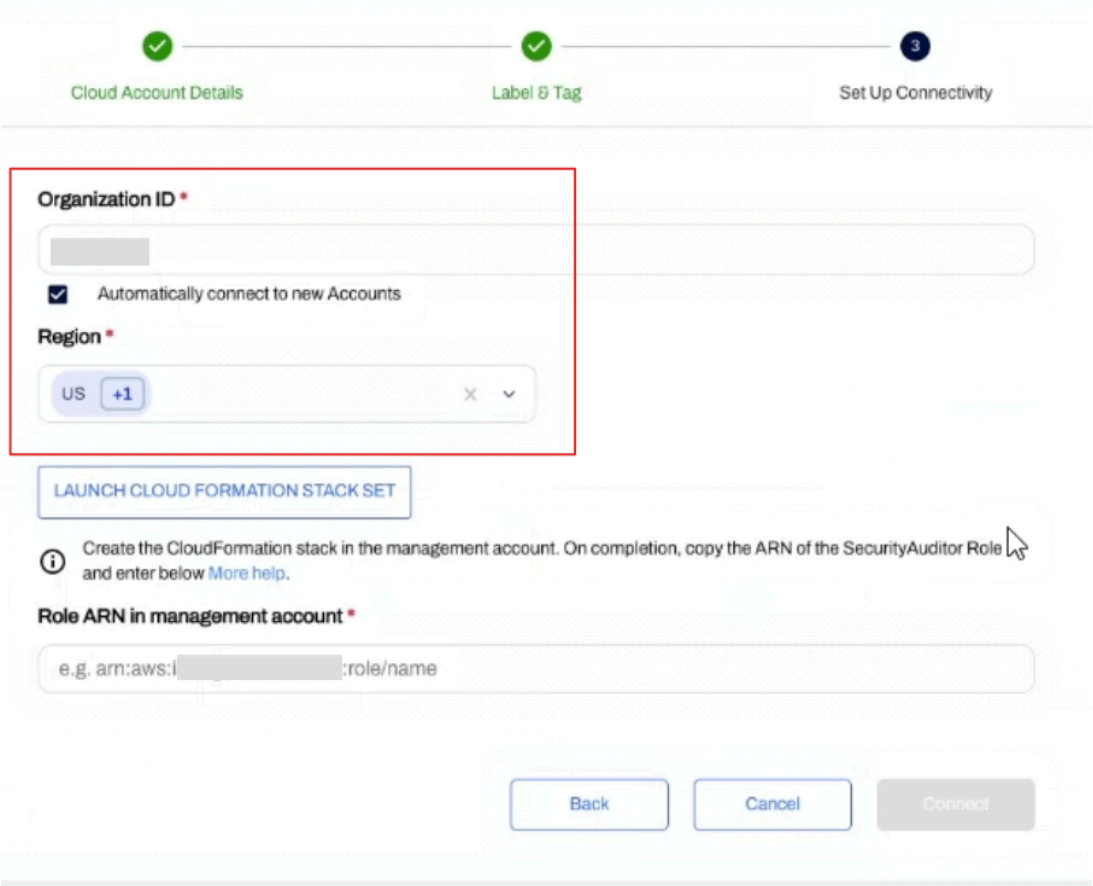
3. Enter AWS Organization Details

- Log in to the **AWS Console** → go to **AWS Organizations**.

- Copy your **Organization ID** (e.g., `r-xxxxxxxxxx`).



- You must use the **root organization account**.
- In AccuKnox, paste the ID into the **AWS Organization ID** field.
- Select the AWS regions where your assets are located.

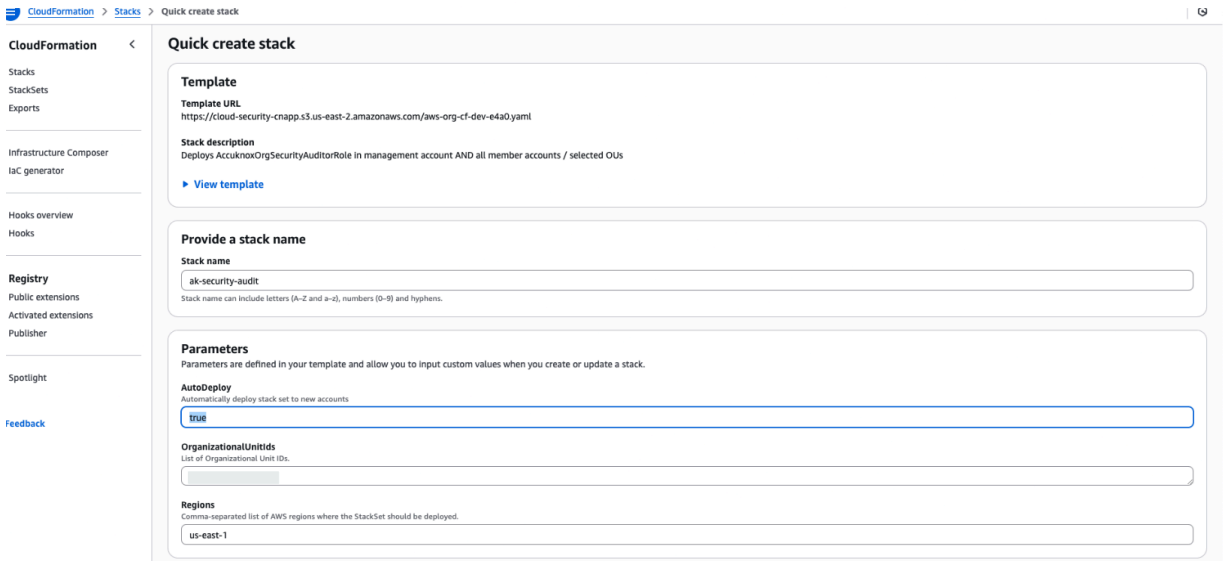


Note

At present, all assets discovered under this organization will inherit these selected labels. Granular labeling for individual assets will be an enhancement in future updates.

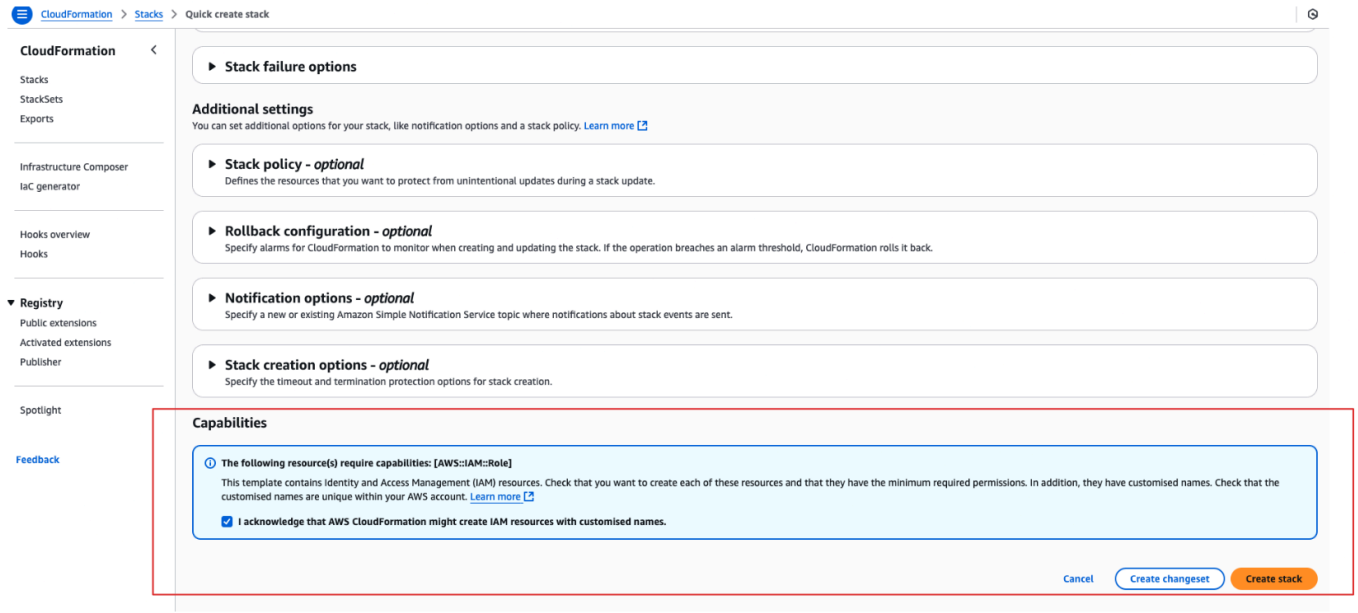
4. Enable Auto-Connect & Launch StackSet

- Toggle **Automatically connect to new accounts** (optional).
- Click **Launch CloudFormation StackSet** to open the AWS Console.



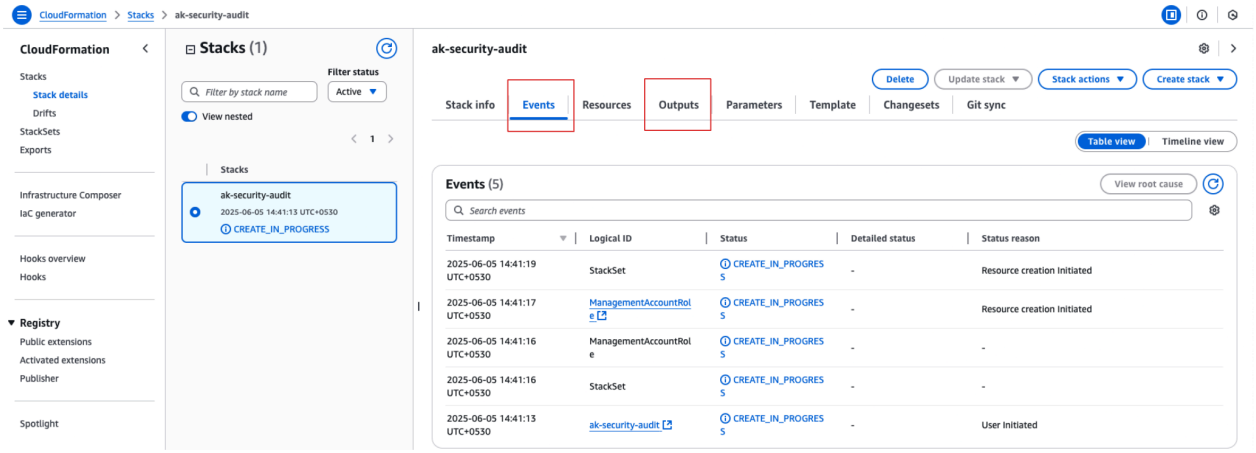
5. Create the Stack in AWS

- Scroll down, check the box: **"I acknowledge that AWS CloudFormation might create IAM resources..."**
- Click **Create stack**.



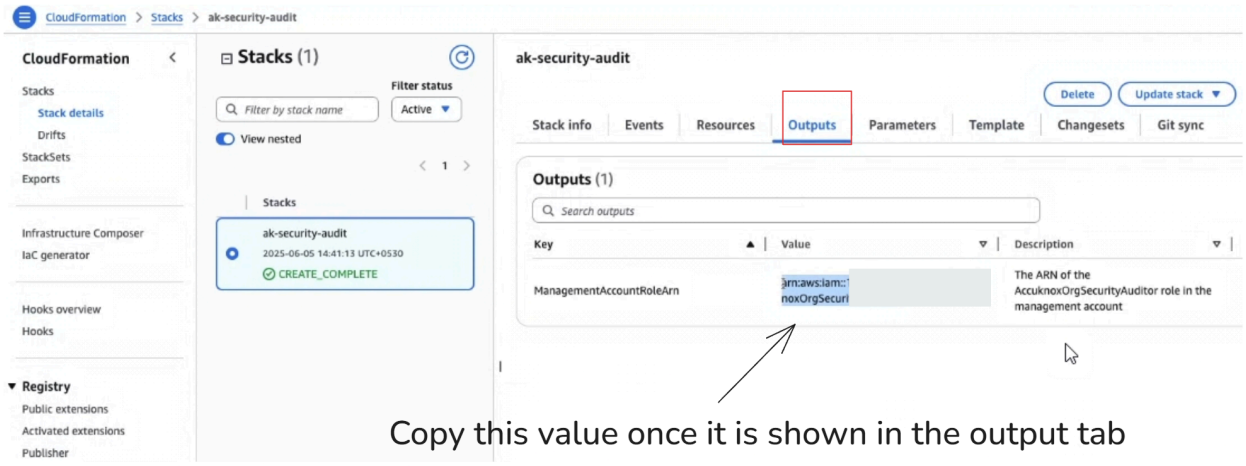
6. Wait for StackSet Deployment

- Wait until the status shows **CREATE_COMPLETE**.



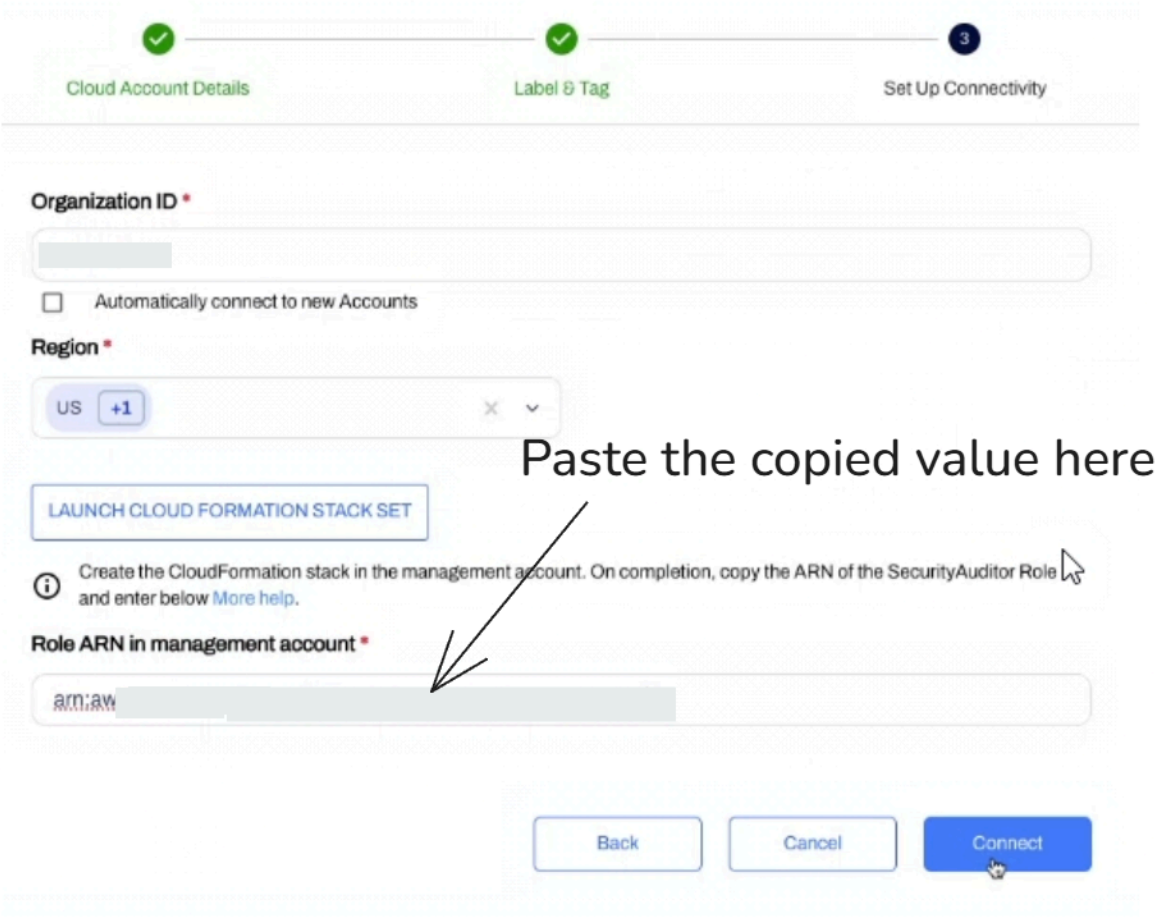
7. Copy Role ARN

- Go to the **Outputs** tab of the StackSet.
- Copy the value of `RoleArnInManagementAccount`.



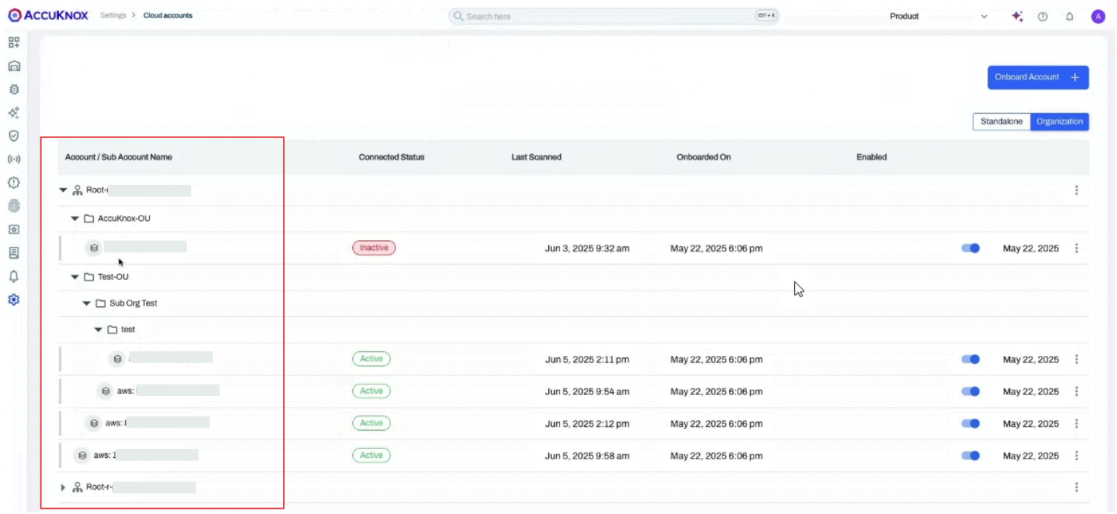
8. Connect in AccuKnox

- Paste the ARN in the **Role ARN** field.
- Click **Connect**.



9. Confirm Onboarding

- You'll be redirected to the **Cloud Accounts** page.
- Refresh the page to see your AWS Organization listed.



Post-Onboarding

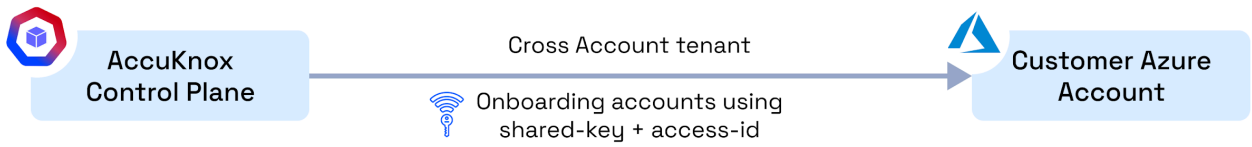
Once your AWS Organization is successfully onboarded:

- **Asset Discovery:** AccuKnox will start an inventory discovery process across all member accounts in the selected regions.
- **Security Scans:** Automated security scans will be scheduled to assess your cloud resources for misconfigurations, vulnerabilities, and compliance violations.
- **Dashboard Population:** Data will begin to populate your AccuKnox dashboards, providing insights into your organization's security posture. This may take some time depending on the size and complexity of your AWS environment.

You have now successfully onboarded your **AWS Organization** to **AccuKnox**, enabling comprehensive, centralized cloud security management.

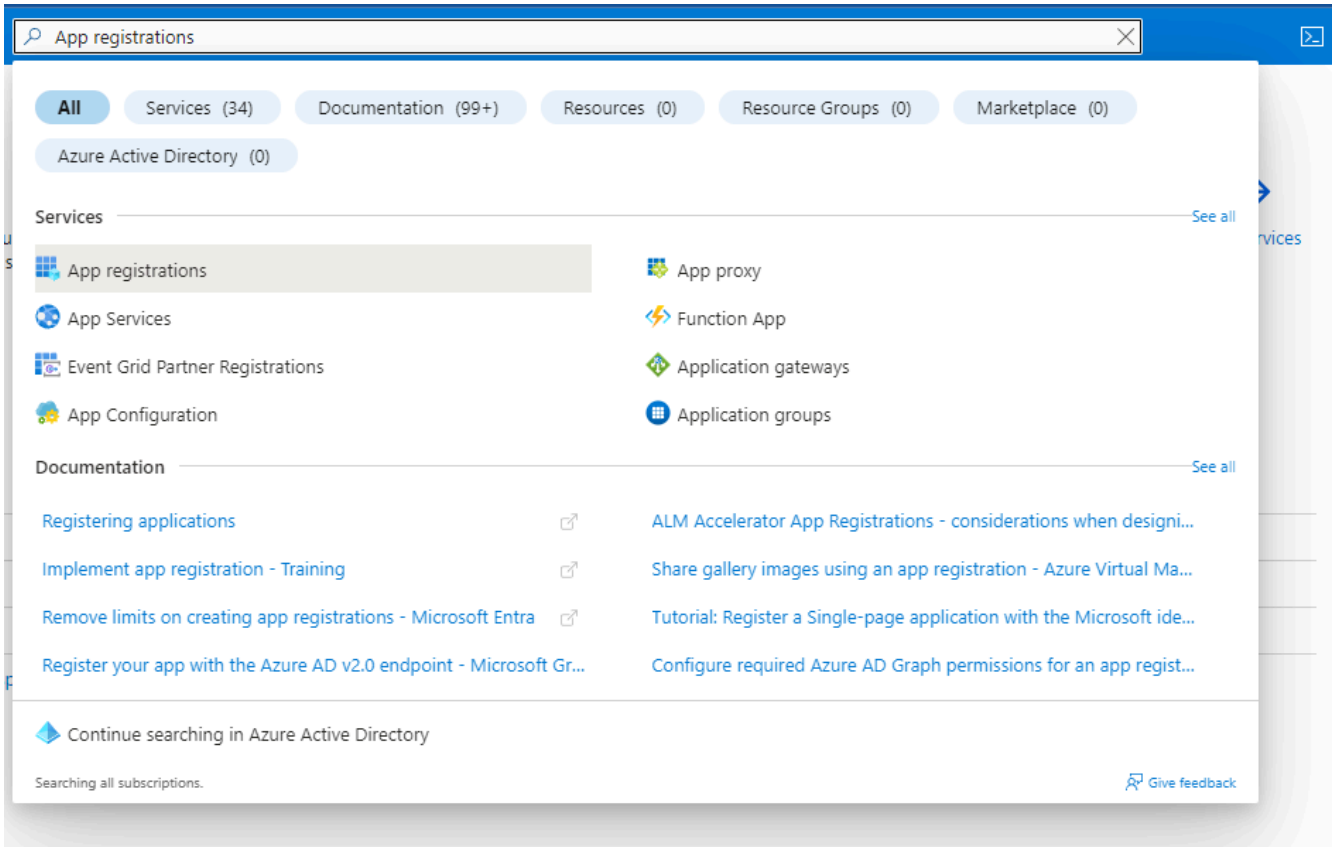
CSPM Pre-requisite for Azure

When the AccuKnox control plane is hosted in a cloud environment, scanning is performed using Cloud account Readonly Access permissions.

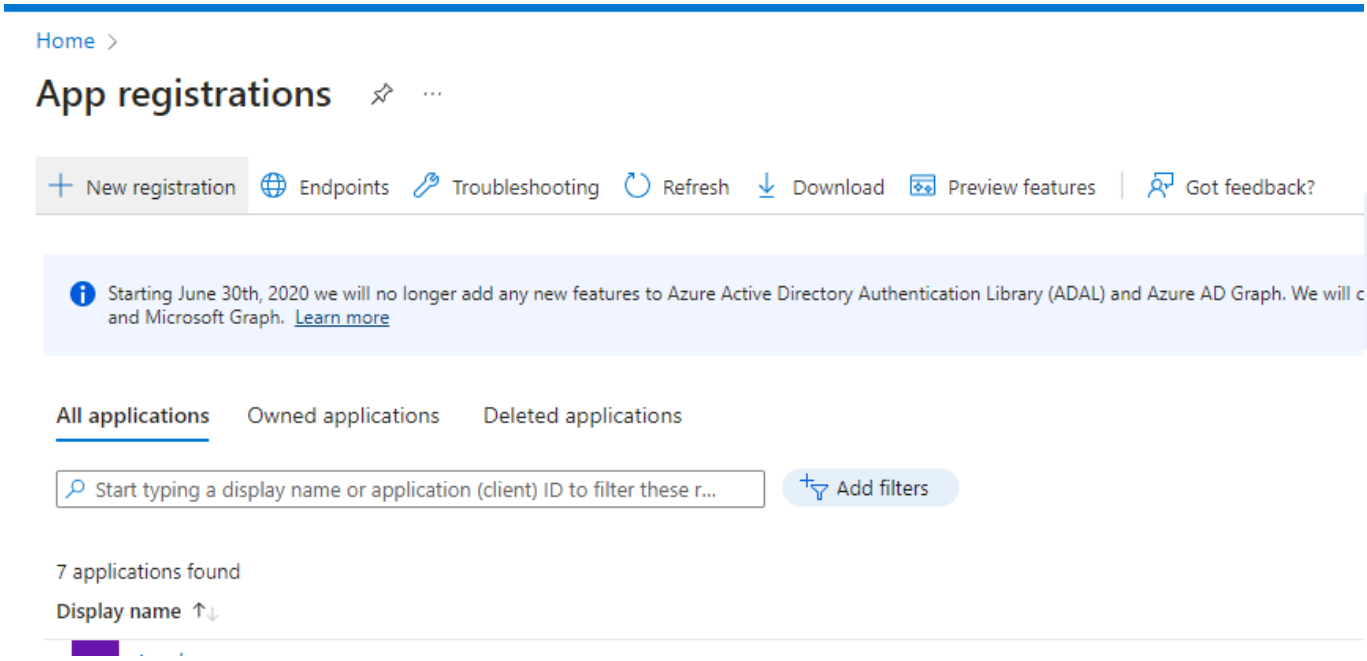


For Azure Onboarding it is required to register an App and giving Security read access to that App from the Azure portal.

Step 1: Go to your Azure Portal and search for *App registrations* and open it



Step 2: Here click on *New registration*



Step 3: Give your application a name, remember this name as it will be used again later, For the rest keep the default settings

Home > App registrations >

Register an application

* Name

The user-facing display name for this application (this can be changed later).

Accuknox-may-2023

Supported account types

Who can use this application or access this API?

- (●) Accounts in this organizational directory only (Default Directory only - Single tenant)
- ( ) Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- ( ) Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- ( ) Personal Microsoft accounts only

Help me choose...

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Select a platform | e.g. https://example.com/auth

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from Enterprise applications.

By proceeding, you agree to the Microsoft Platform Policies

Register

Step 4: Now your application is created, save *Application ID* and *Directory ID* as they will be needed to for onboarding on AccuKnox Saas and then click on 'Add a certificate or secret'

3

Delete | Endpoints | Preview features

Essentials

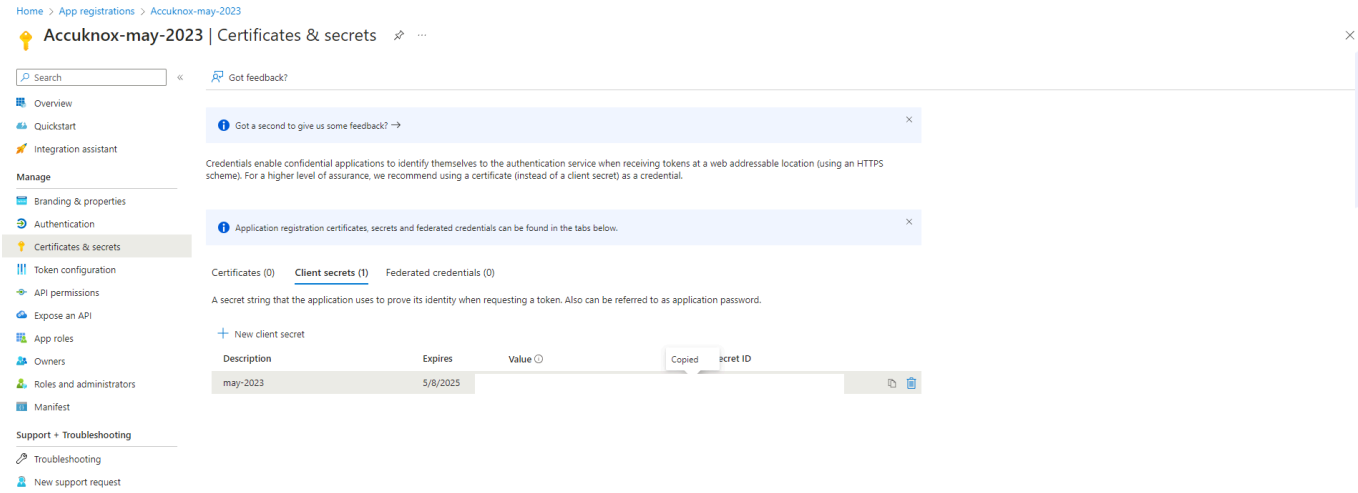
Display name	: Accuknox-may-2023	Client credentials	: Add a certificate or secret
Application (client) ID	:	Redirect URIs	: Add a Redirect URI
Object ID	:	Application ID URI	: Add an Application ID URI
Directory (tenant) ID	:	Managed application in l...	: Accuknox-may-2023
Supported account types	: My organization only		

Welcome to the new and improved App registrations. Looking to learn how it's changed from App registrations (Legacy)? Learn more

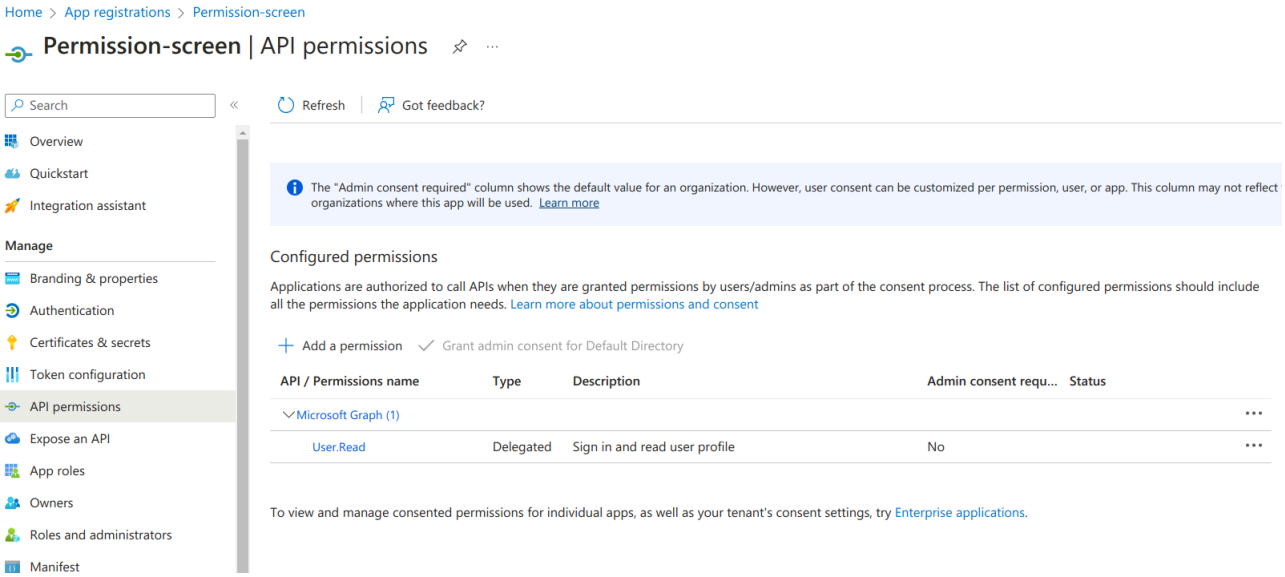
Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure AD Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications wi be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. Learn more

Get Started | Documentation

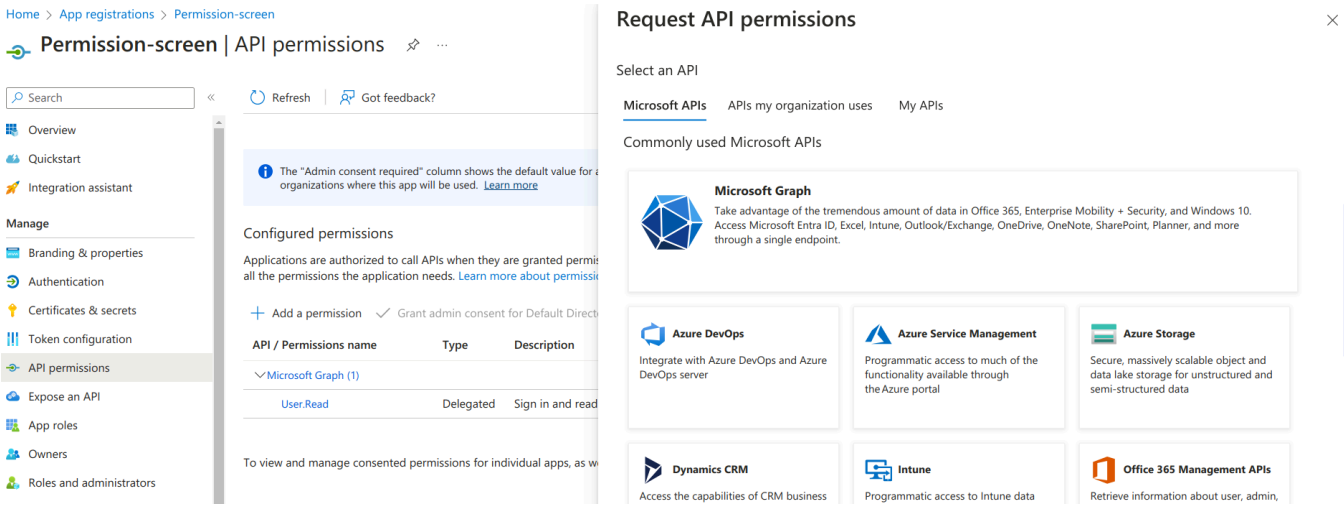
Step 5: Click on new client secret and enter the name and expiration date to get *secret id* and *secret value*, save this secret value as this will also be needed for onboarding.



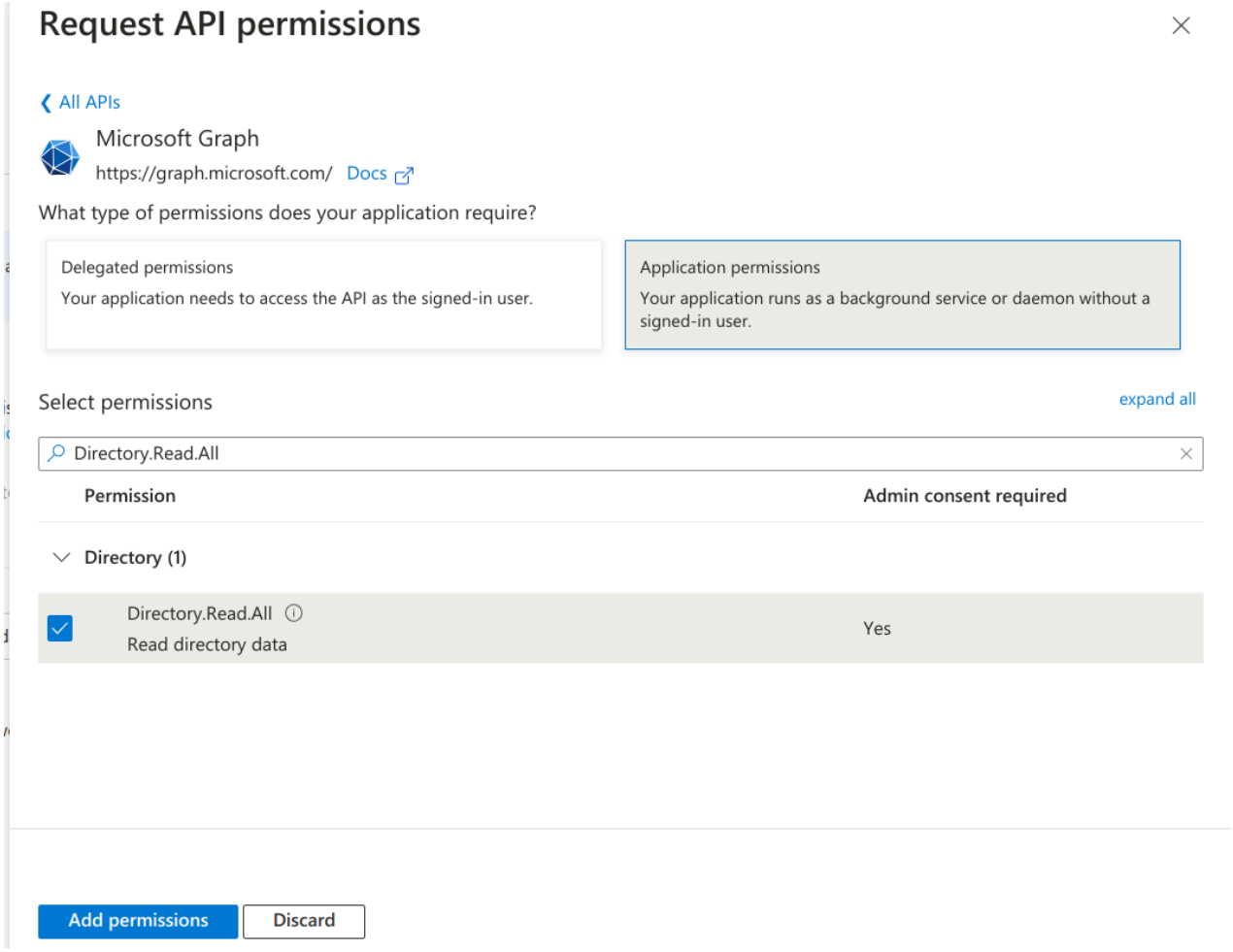
Step 6: Next, go to *API permissions* tab and click on 'Add permission'



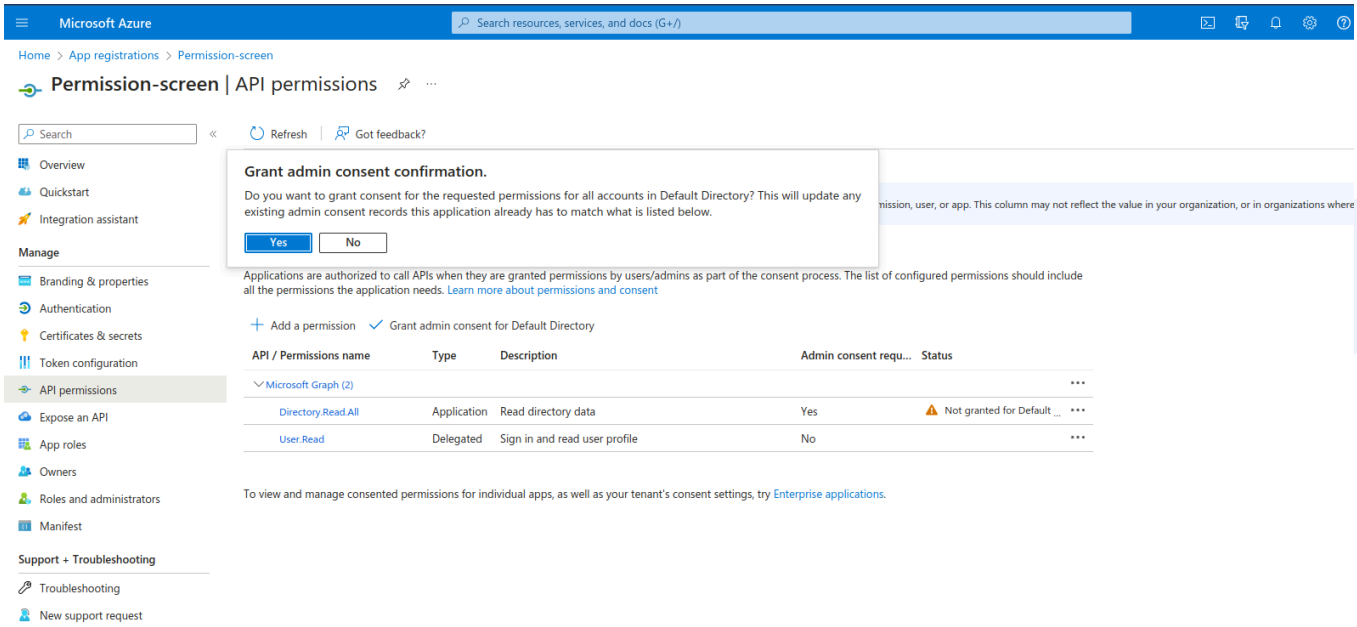
Step 7: On the screen that appears, click on 'Microsoft Graph'



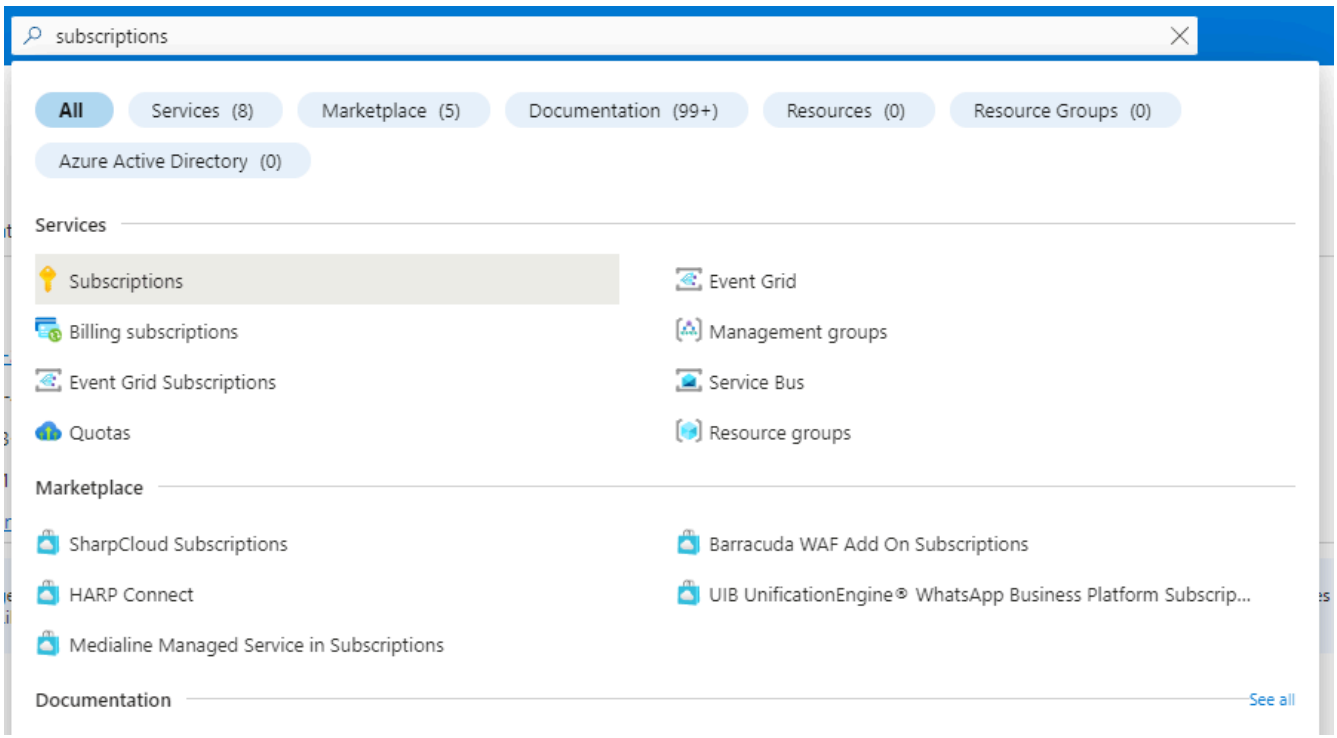
Step 8: Next, select Application Permissions and then search for Directory.Read.All and click on Add permissions



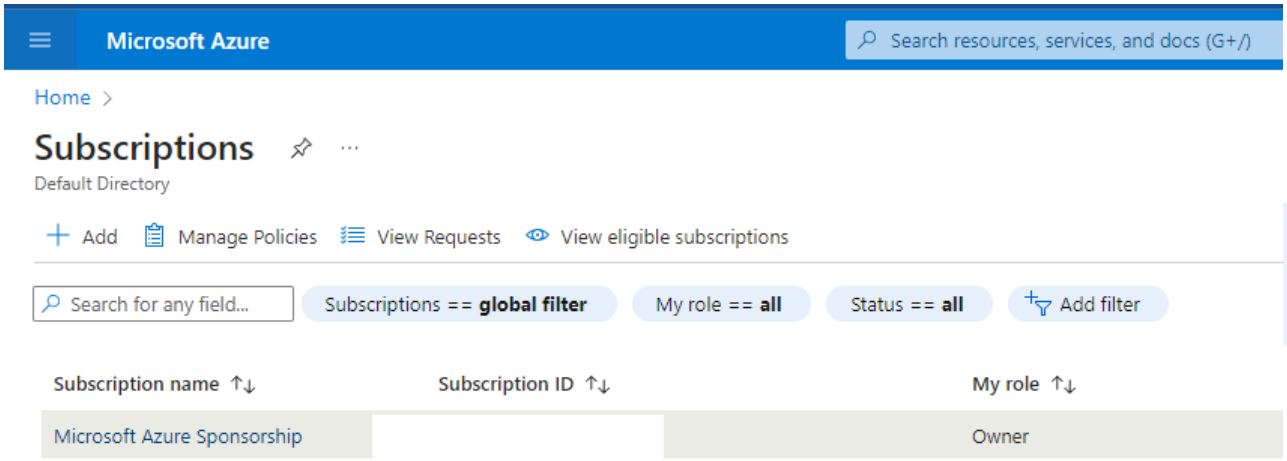
Step 9: Select 'Grant Admin Consent' for Default Directory and click on 'Yes'



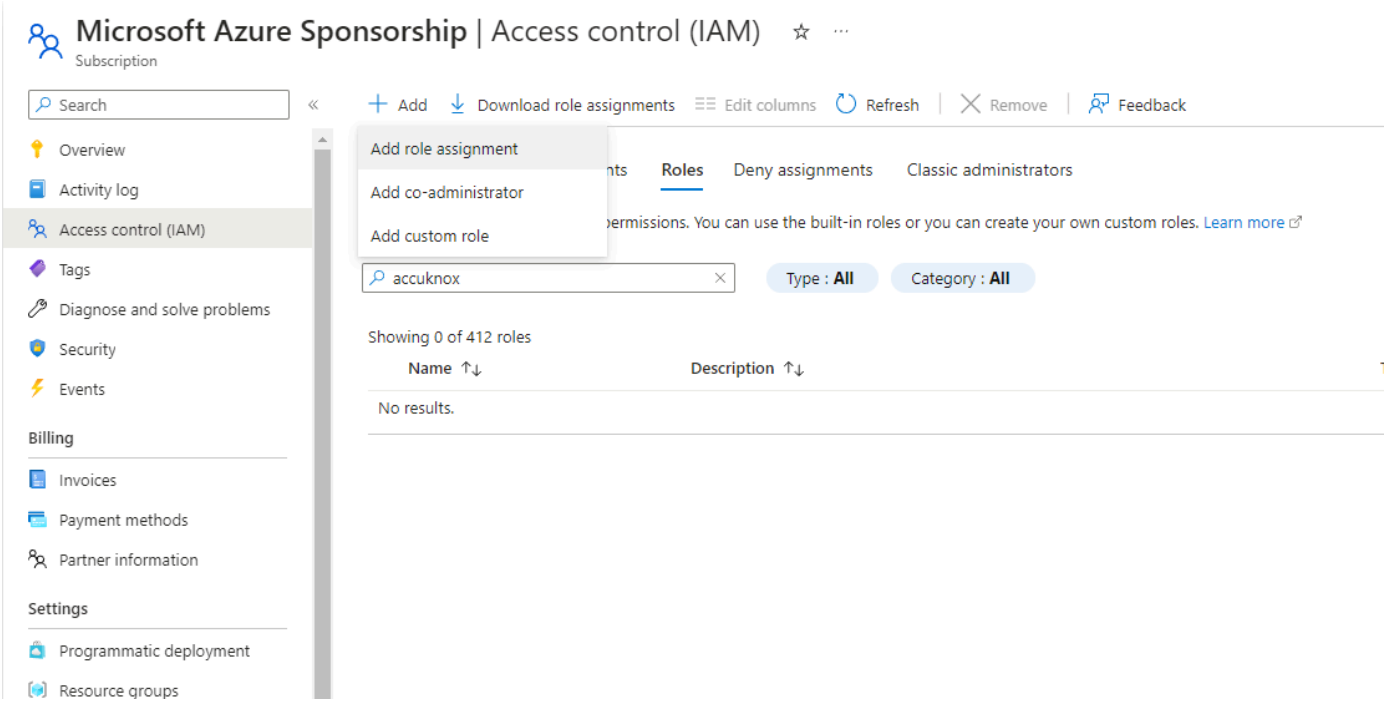
Step 10: Now we need to give Security read permissions to this registered Application , to do that go to subscriptions



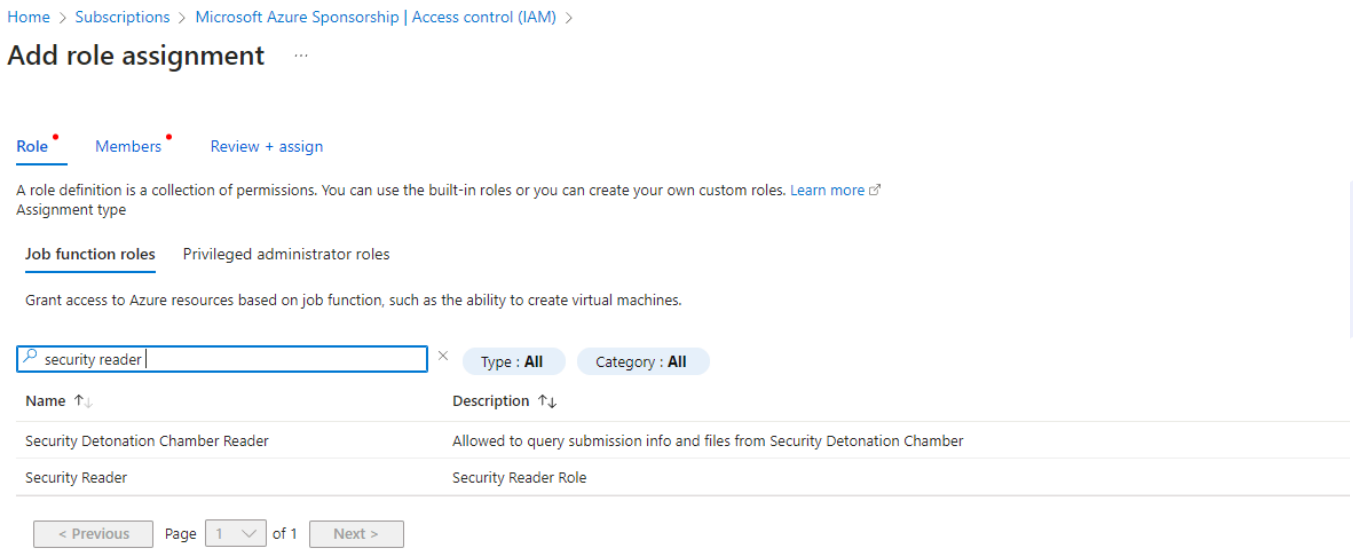
Step 11: First save the subscription ID and click on the subscription name , here it is "Microsoft Azure Sponsorship"



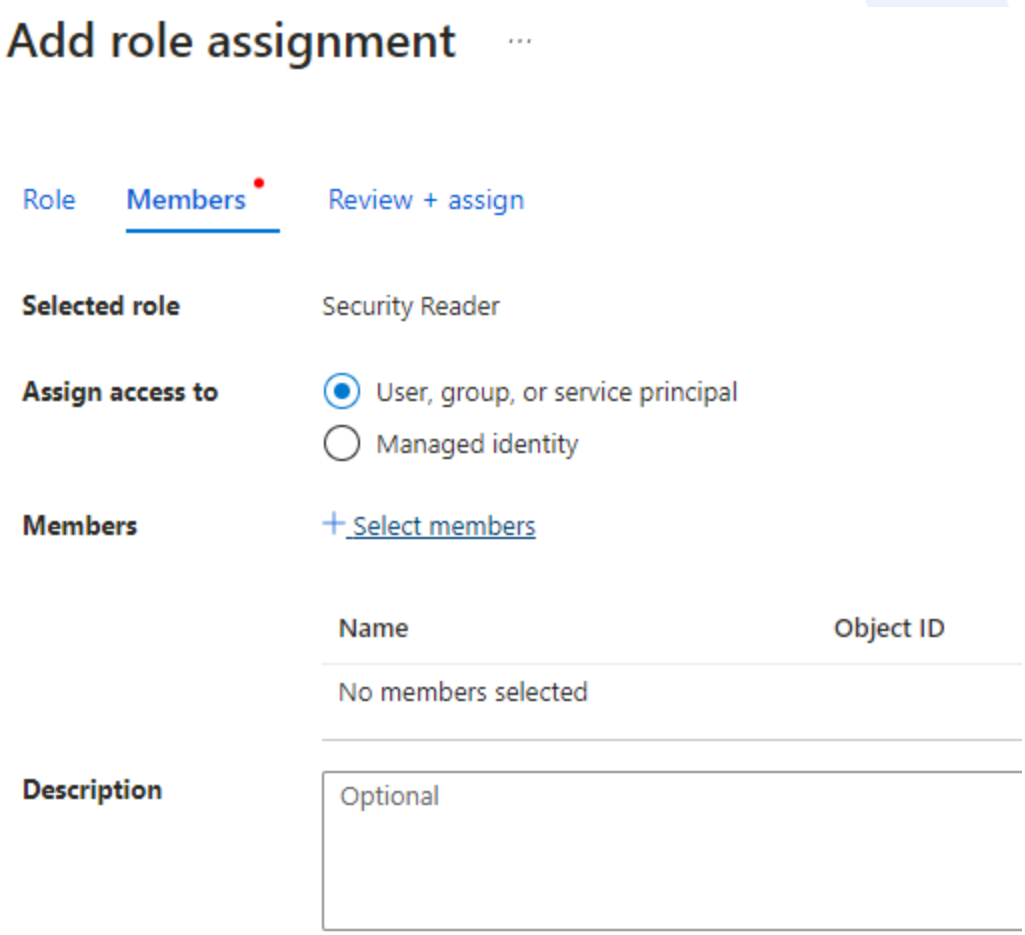
Step 12: Navigate to Access control(IAM) and go to Roles , here select Add and Add role assignment



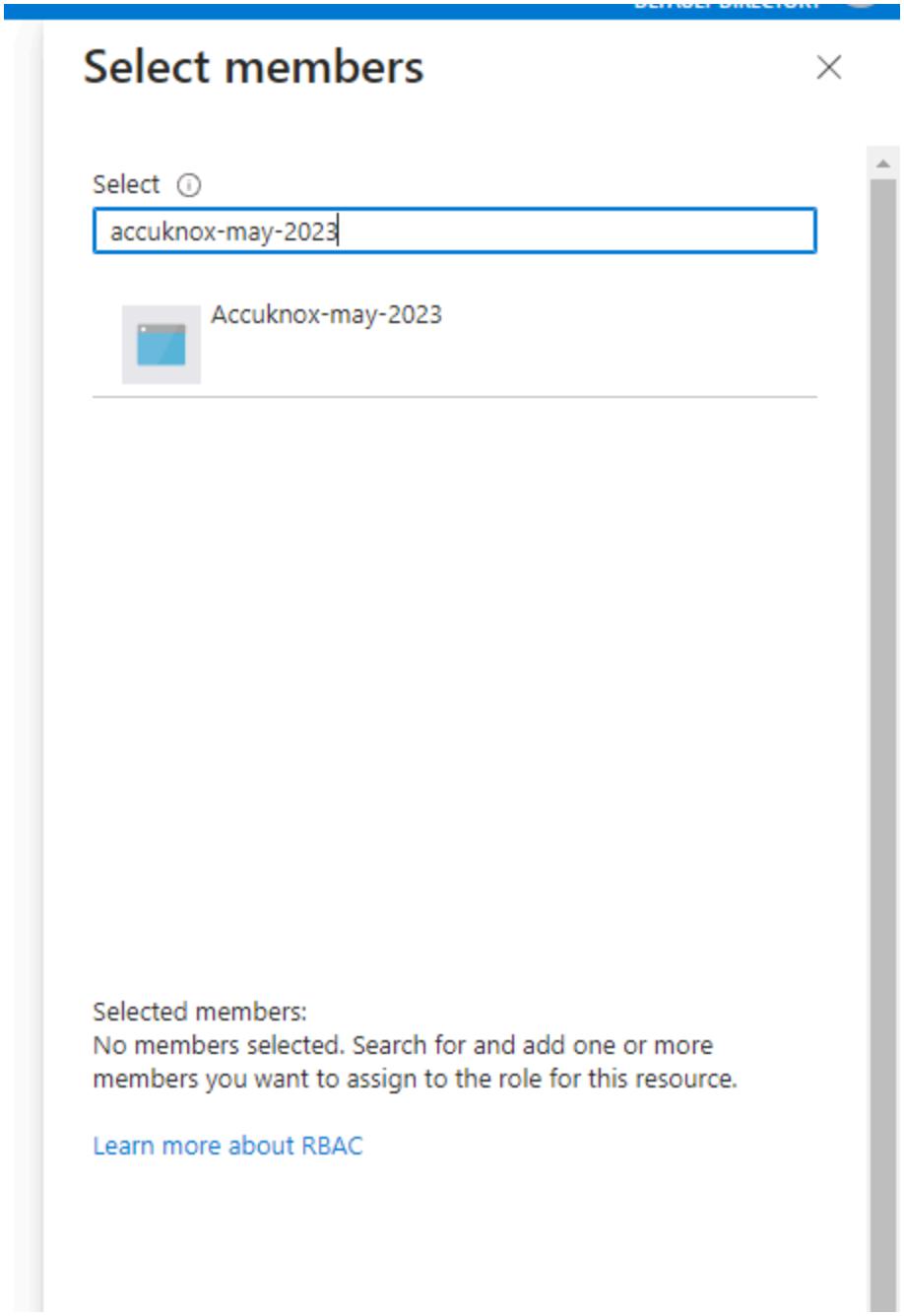
Step 13: Search for “Security Reader” Job function Role, select it and press *next*



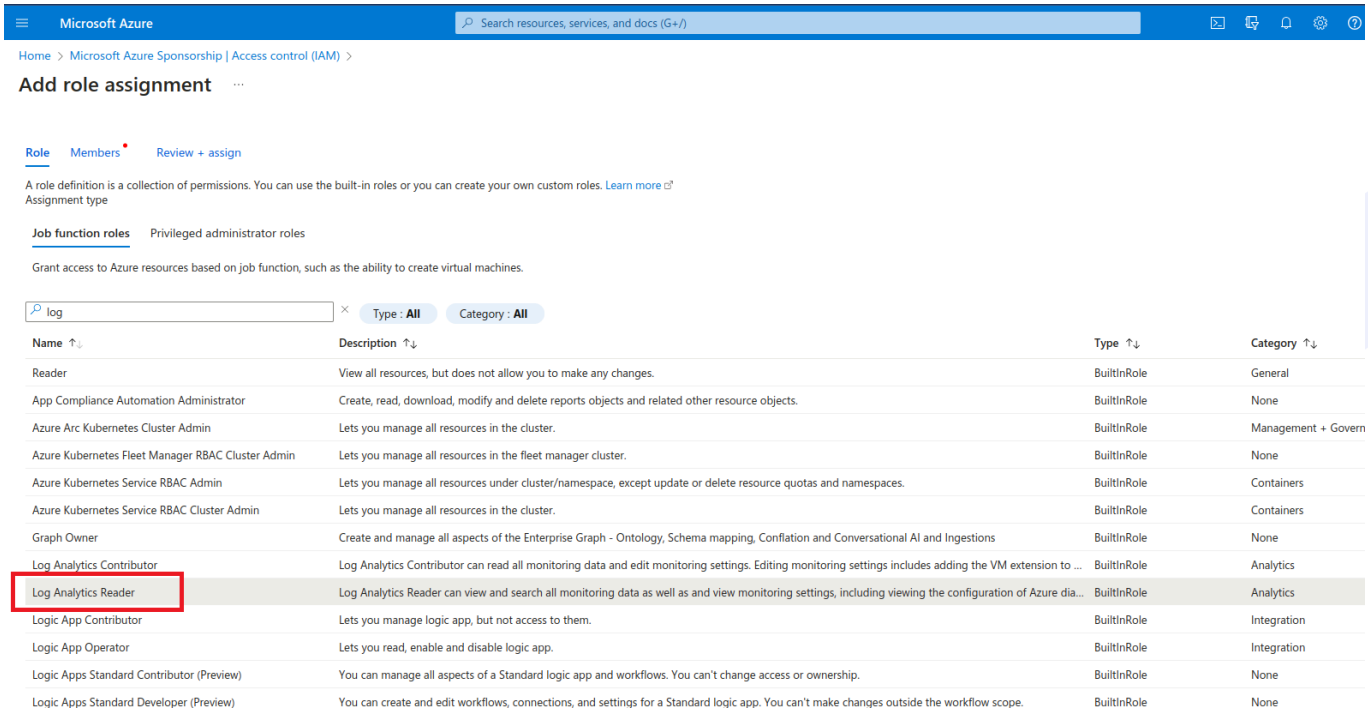
Step 14: In the member section click on Select *members* it will open a dropdown menu on the right hand side



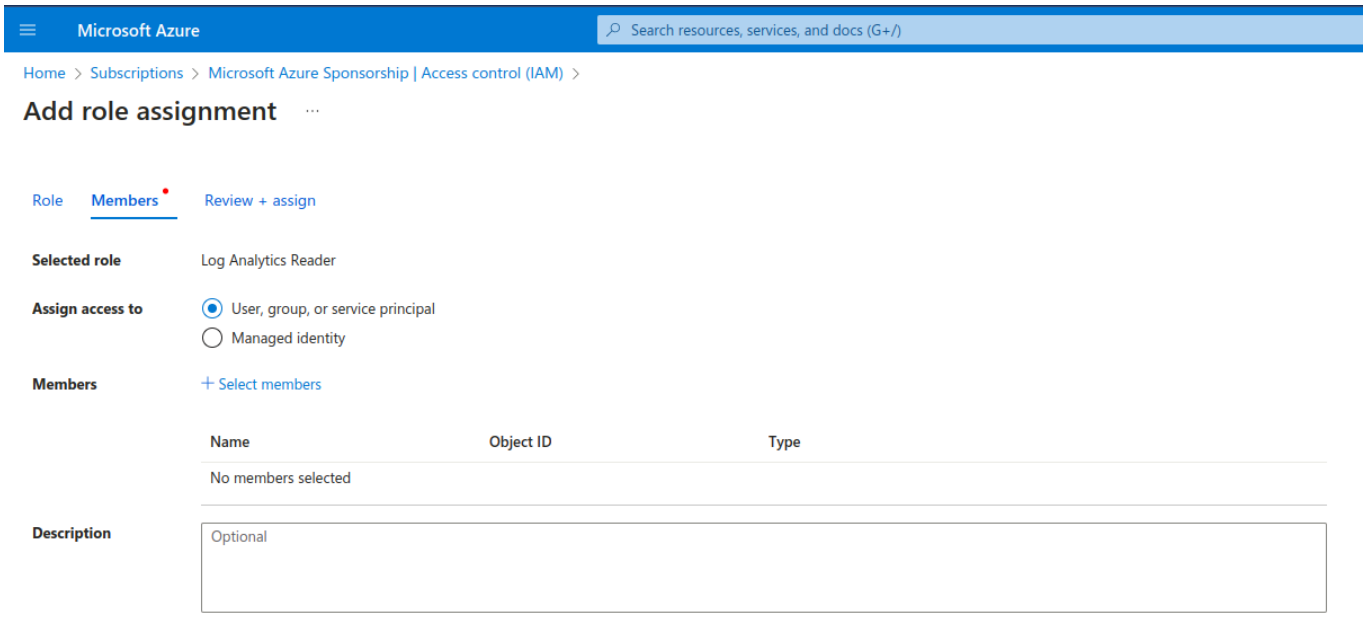
Step 15: Here search for the Application that you registered in the beginning , select the application and click on *review and assign*.



Step 16: Similarly, we have to add another role. This time, search for *Log Analytics Reader*. Select it and click *next*



Step 17: Now, click on *Select members*, select the application that was created similar to the previous role. Finally, click on *Review and Assign*.



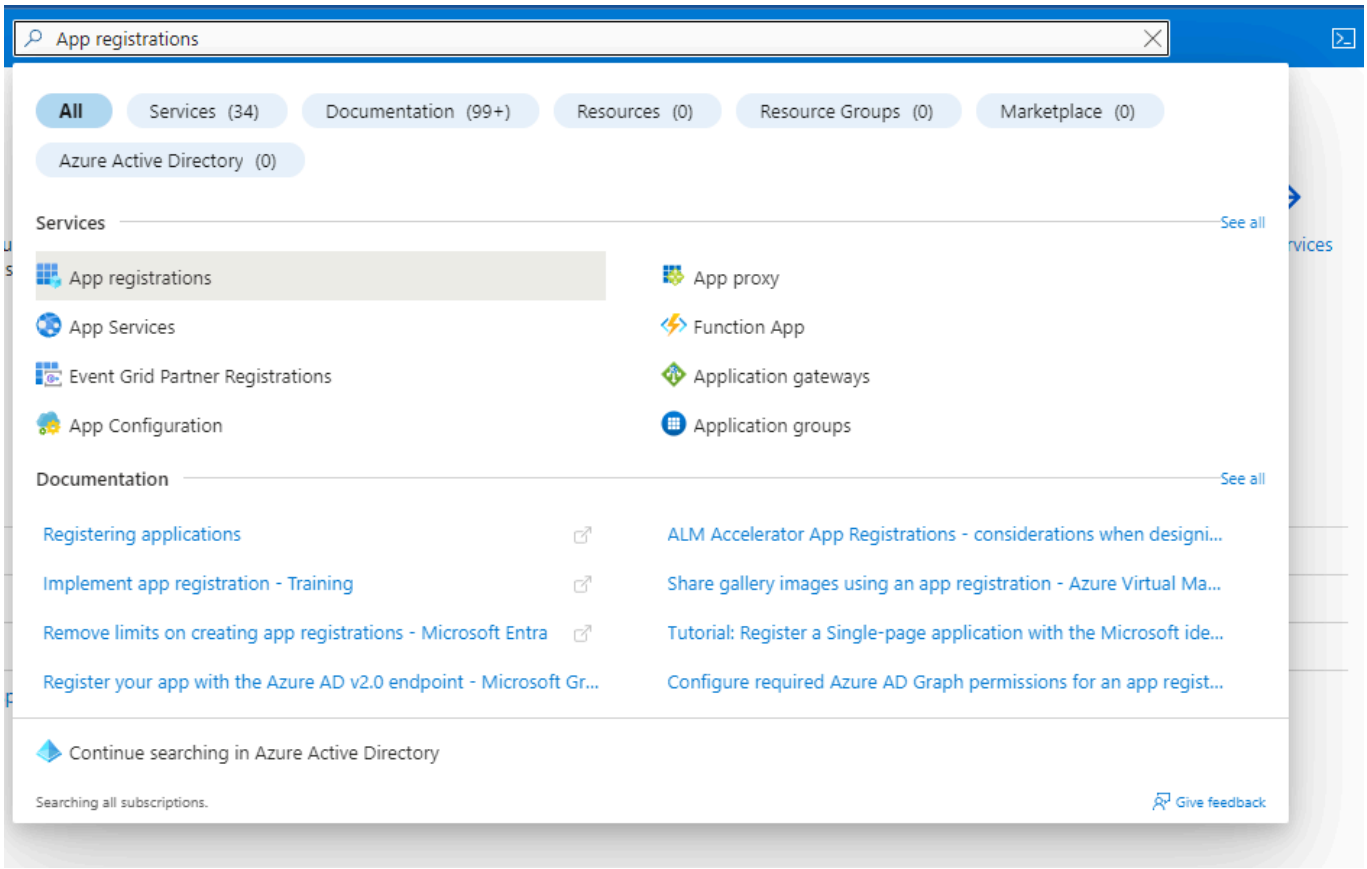
Azure Account onboarding

In this section we can find the steps to onboard an Azure cloud account to the AccuKnox SaaS platform

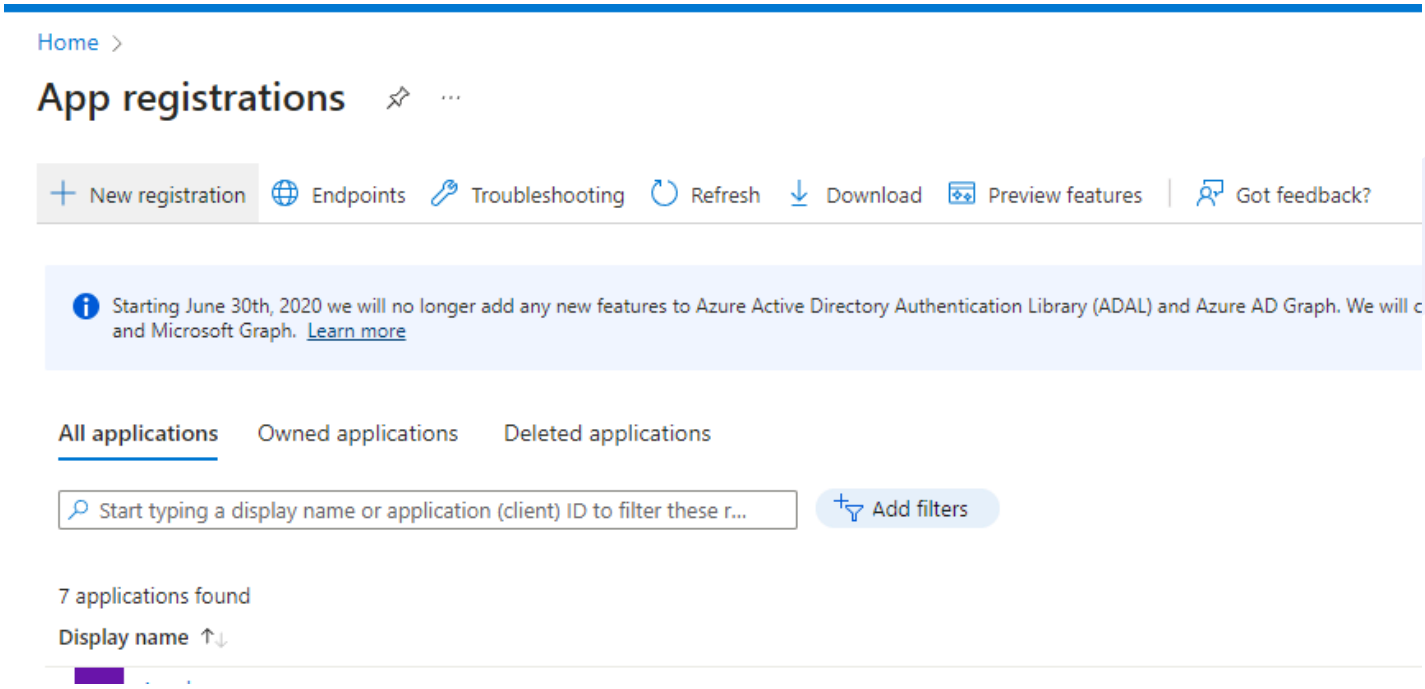
Rapid Onboarding (via Azure)

For Azure Onboarding it is required to register an App and giving Security read access to that App from the Azure portal.

Step 1: Go to your Azure Portal and search for *App registrations* and open it



Step 2: Here click on *New registration*



Step 3: Give your application a name, remember this name as it will be used again later, For the rest keep the default settings

Home > App registrations >

Register an application

* Name

The user-facing display name for this application (this can be changed later).

Accuknox-may-2023

Supported account types

Who can use this application or access this API?

- (●) Accounts in this organizational directory only (Default Directory only - Single tenant)
- ( ) Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- ( ) Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- ( ) Personal Microsoft accounts only

Help me choose...

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Select a platform | e.g. https://example.com/auth

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from Enterprise applications.

By proceeding, you agree to the Microsoft Platform Policies

Register

Step 4: Now your application is created, save *Application ID* and *Directory ID* as they will be needed to for onboarding on AccuKnox Saas and then click on 'Add a certificate or secret'

3

Delete | Endpoints | Preview features

Essentials

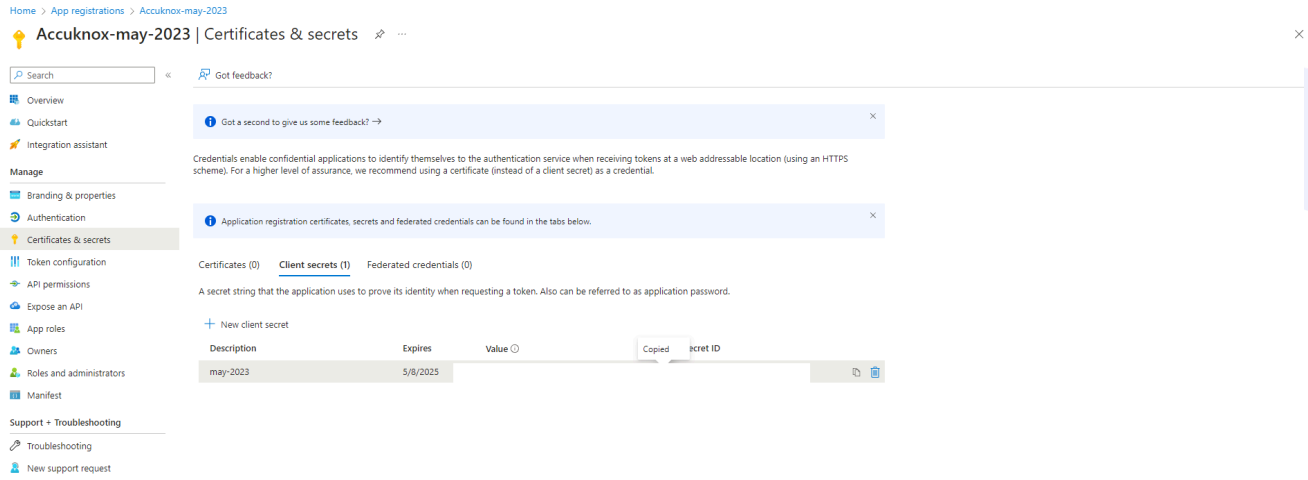
Display name	: Accuknox-may-2023	Client credentials	: Add a certificate or secret
Application (client) ID	:	Redirect URIs	: Add a Redirect URI
Object ID	:	Application ID URI	: Add an Application ID URI
Directory (tenant) ID	:	Managed application in l...	: Accuknox-may-2023
Supported account types	: My organization only		

Welcome to the new and improved App registrations. Looking to learn how it's changed from App registrations (Legacy)? Learn more

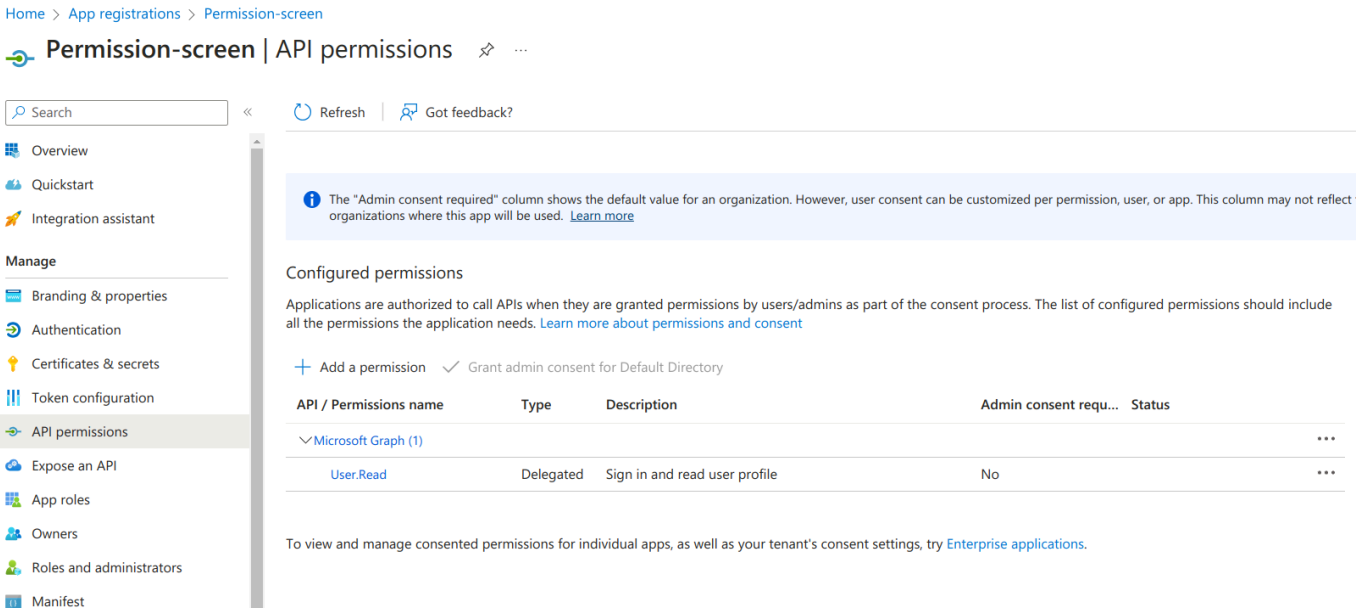
Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure AD Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications wi be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. Learn more

Get Started | Documentation

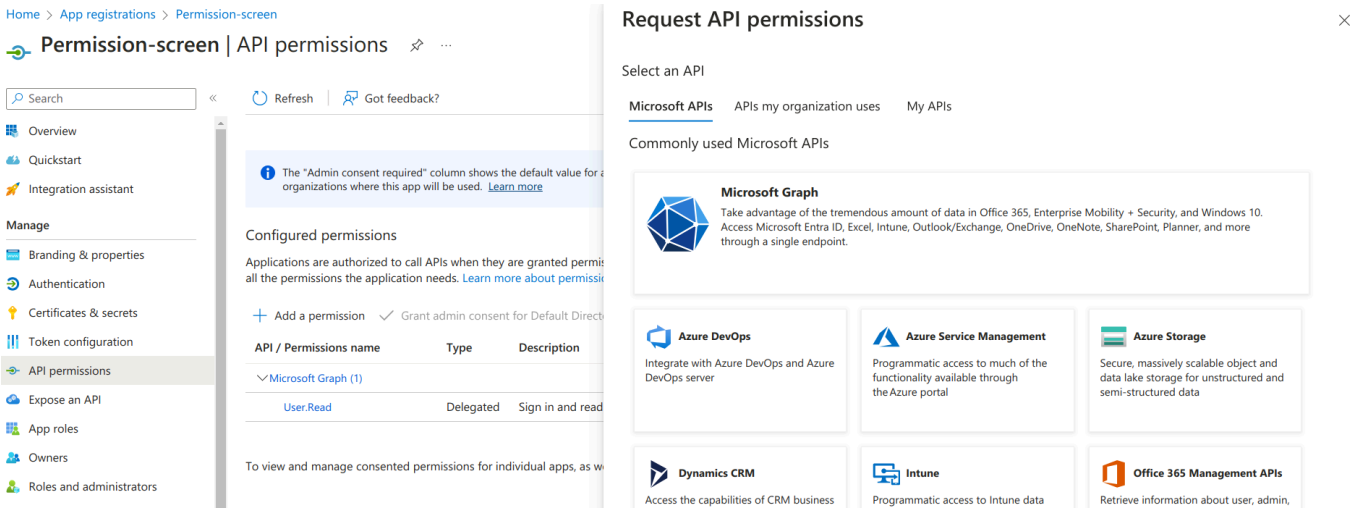
Step 5: Click on new client secret and enter the name and expiration date to get *secret id* and *secret value*, save this secret value as this will also be needed for onboarding.



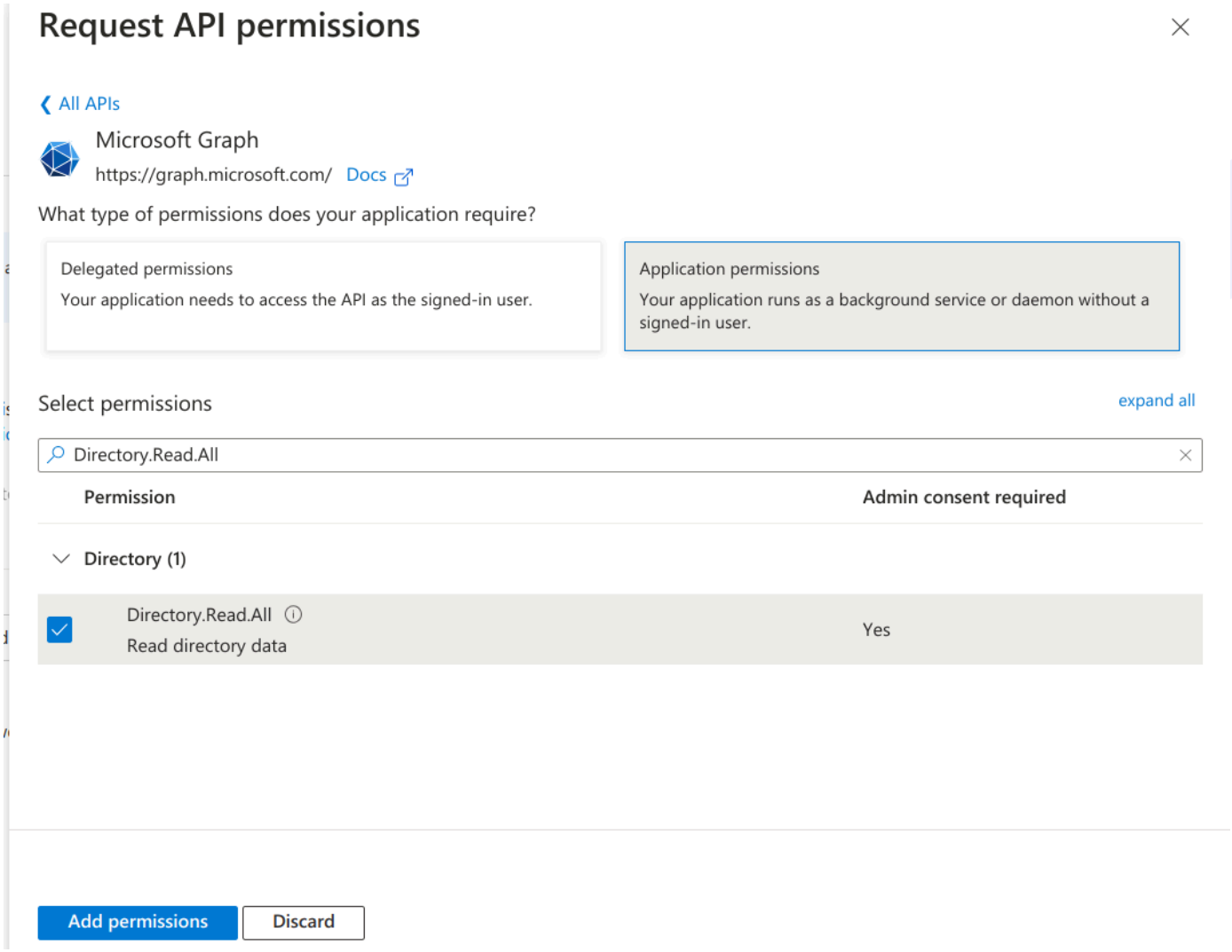
Step 6: Next, go to *API permissions* tab and click on 'Add permission'



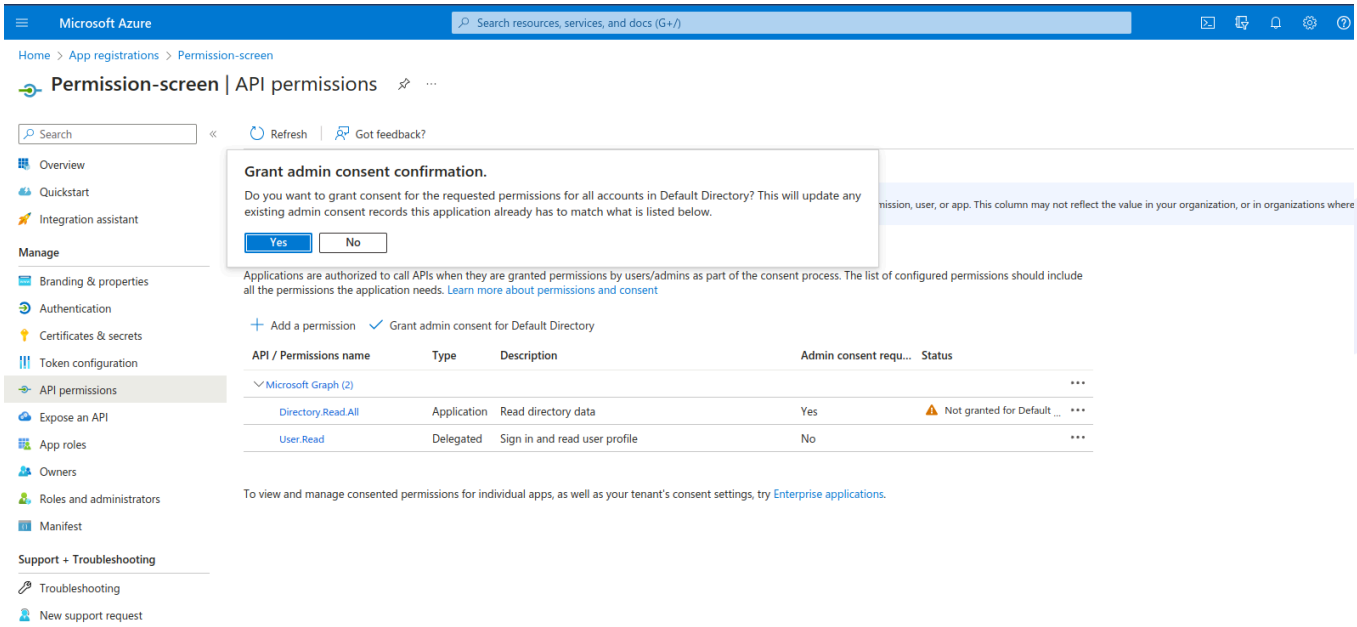
Step 7: On the screen that appears, click on 'Microsoft Graph'



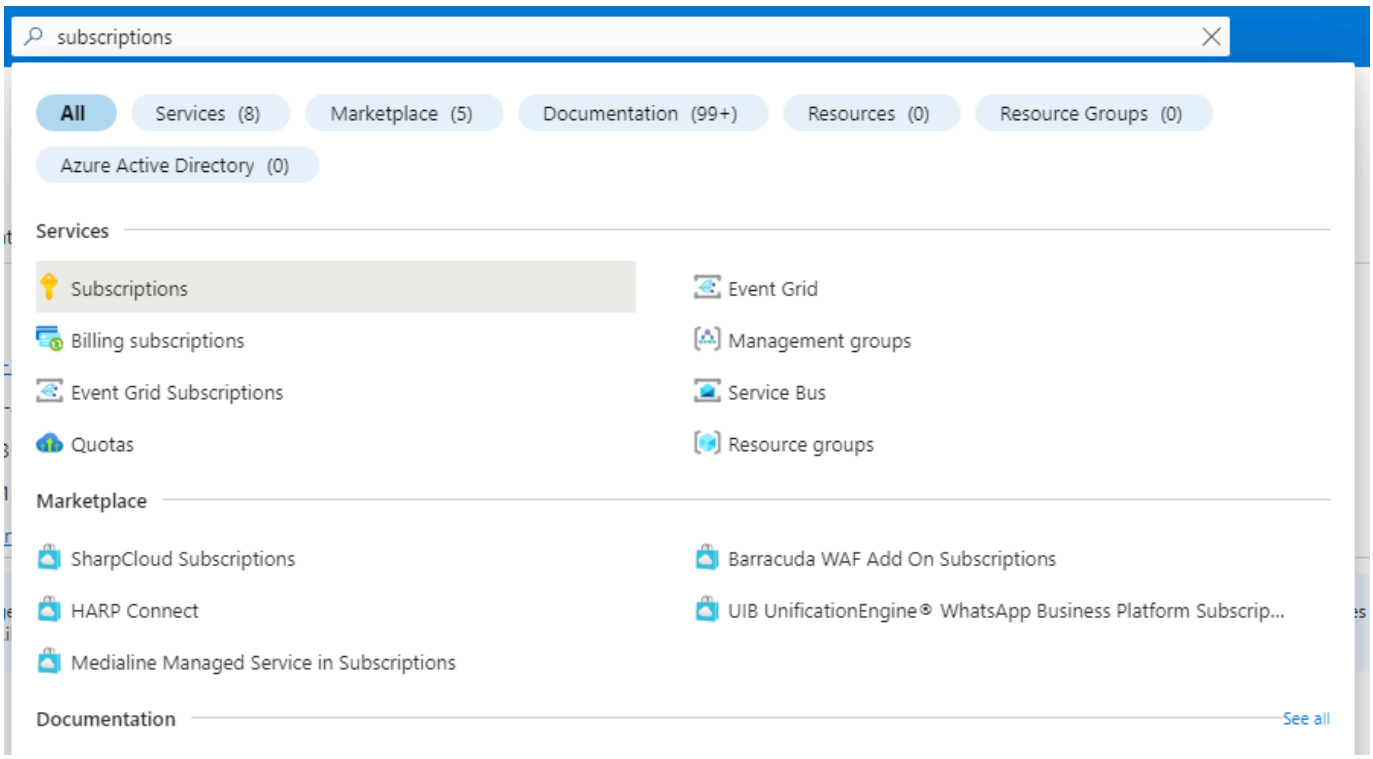
Step 8: Next, select Application Permissions and then search for Directory.Read.All and click on Add permissions



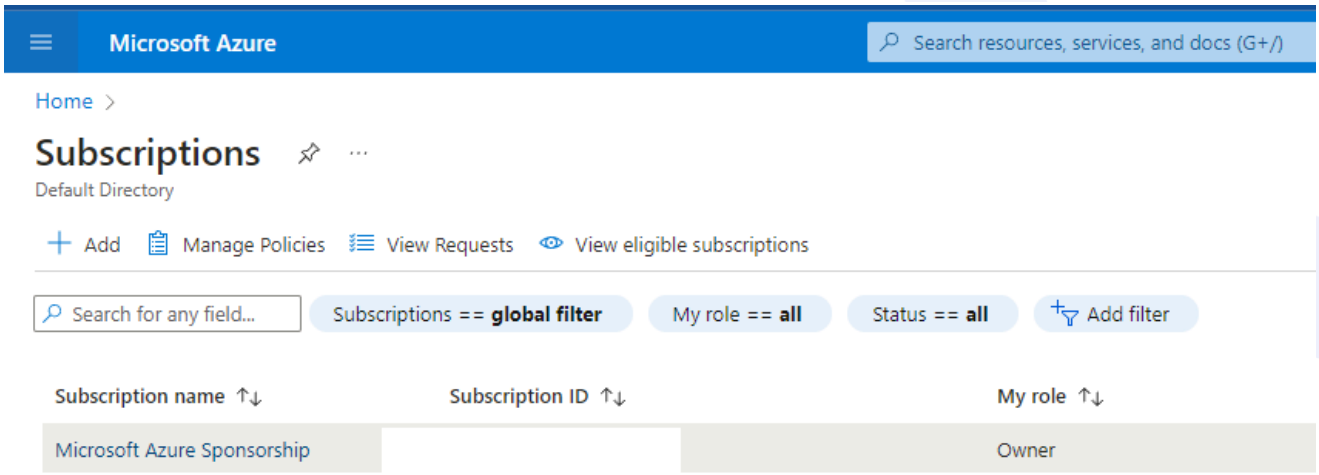
Step 9: Select 'Grant Admin Consent' for Default Directory and click on 'Yes'



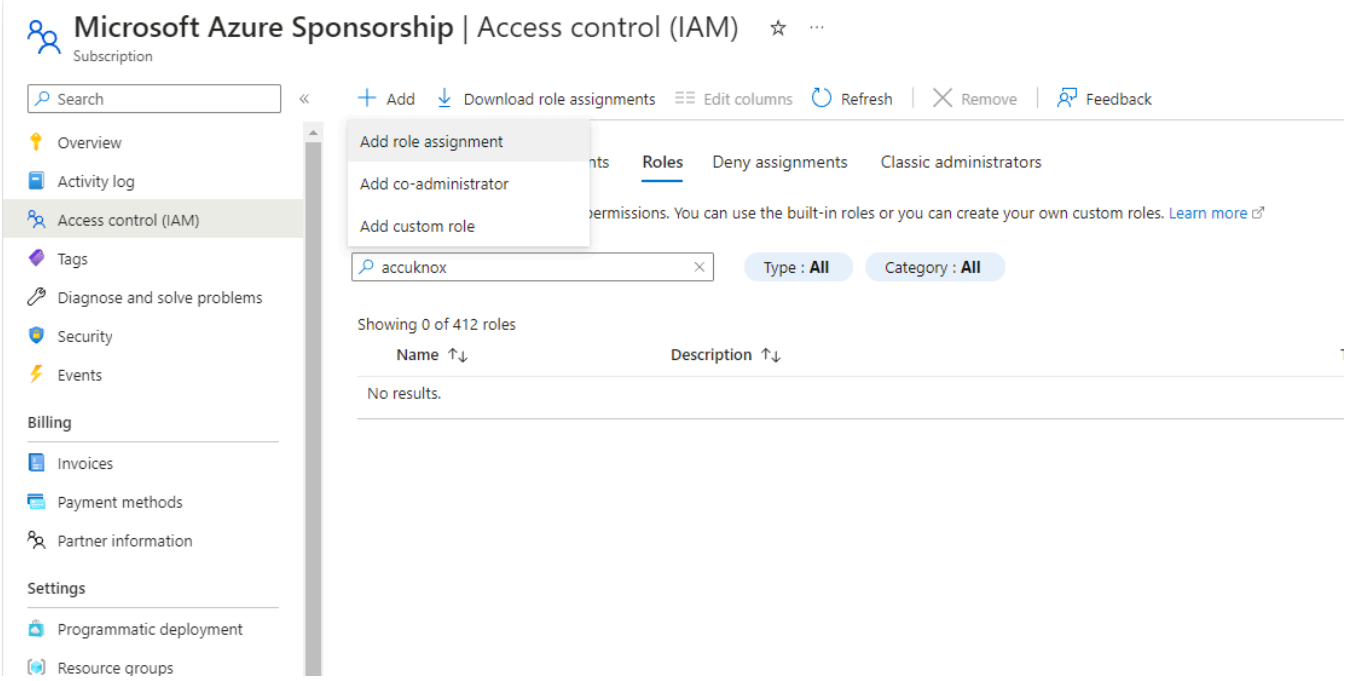
Step 10: Now we need to give Security read permissions to this registered Application , to do that go to subscriptions



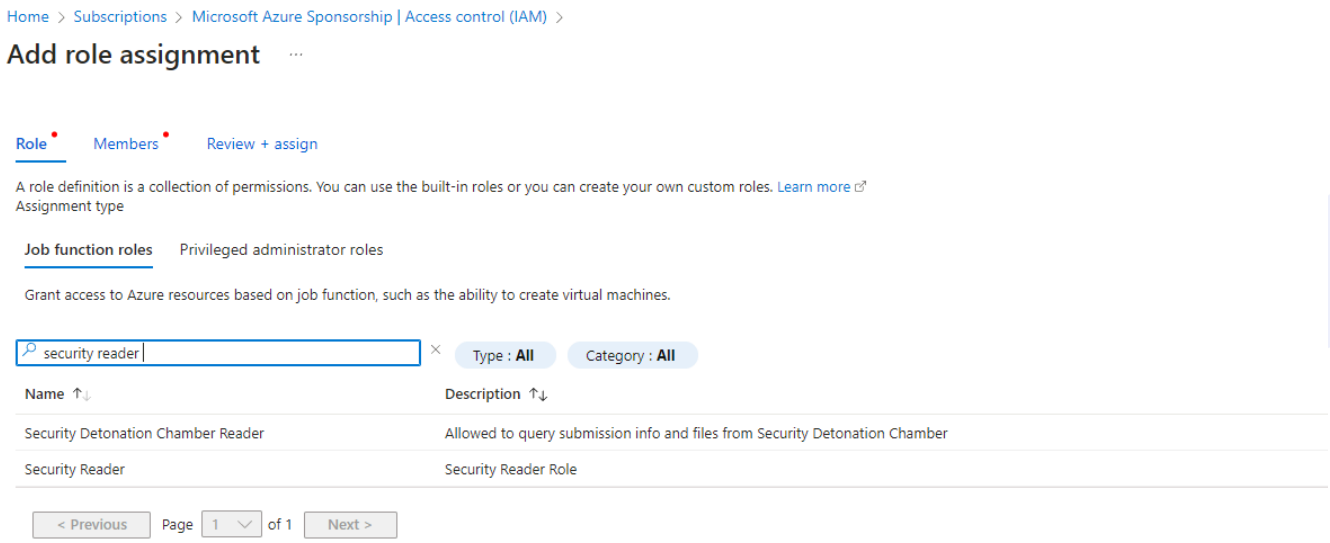
Step 11: First save the subscription ID and click on the subscription name , here it is "Microsoft Azure Sponsorship"



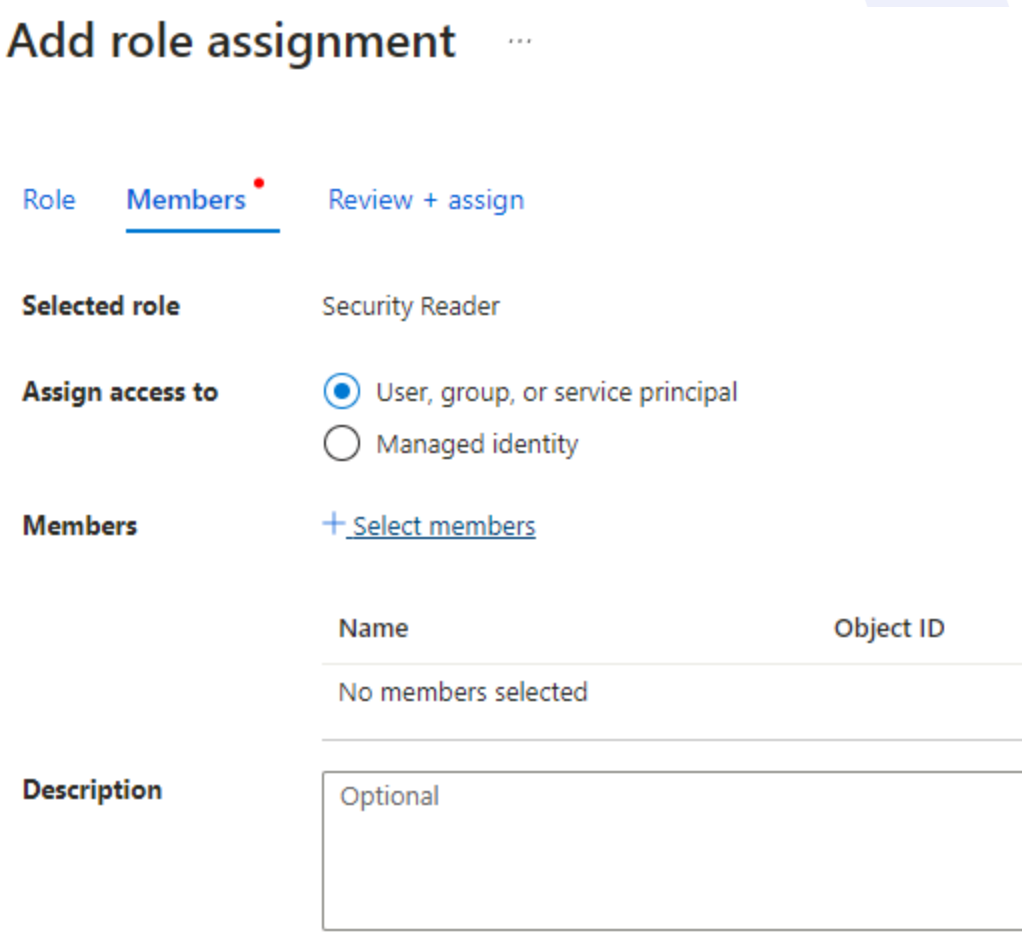
Step 12: Navigate to Access control(IAM) and go to Roles , here select Add and Add role assignment



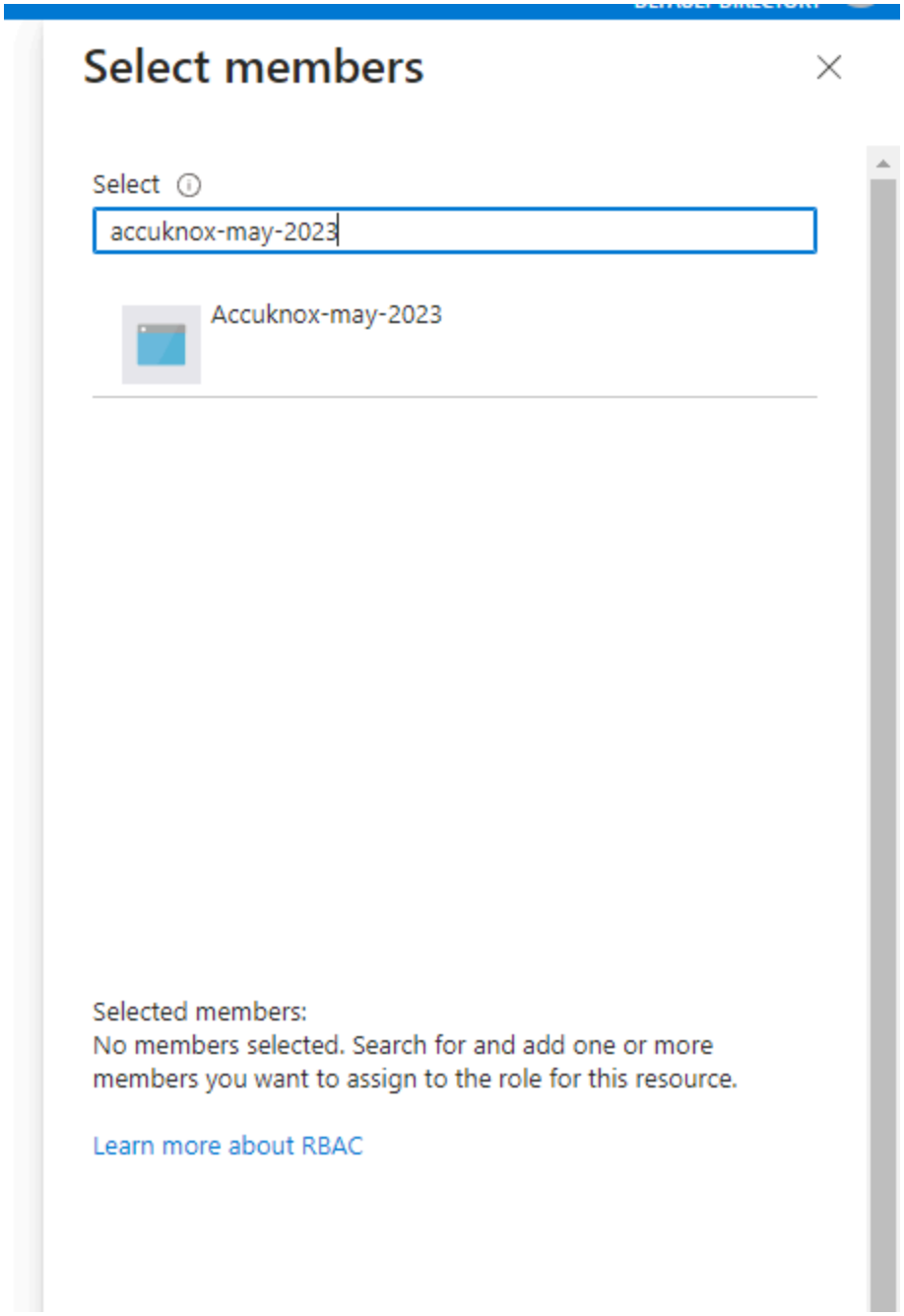
Step 13: Search for “Security Reader” Job function Role, select it and press *next*



Step 14: In the member section click on Select *members* it will open a dropdown menu on the right hand side



Step 15: Here search for the Application that you registered in the beginning , select the application and click on *review and assign*.



Step 16: Similarly, we have to add another role. This time, search for *Log Analytics Reader*. Select it and click *next*

Microsoft Azure

Search resources, services, and docs (G+/)

Home > Microsoft Azure Sponsorship | Access control (IAM) >

Add role assignment

Role | Members | Review + assign

A role definition is a collection of permissions. You can use the built-in roles or you can create your own custom roles. Learn more

Assignment type

Job function roles | Privileged administrator roles

Grant access to Azure resources based on job function, such as the ability to create virtual machines.

log | Type : All | Category : All

Name	Description	Type	Category
Reader	View all resources, but does not allow you to make any changes.	BuiltInRole	General
App Compliance Automation Administrator	Create, read, download, modify and delete reports objects and related other resource objects.	BuiltInRole	None
Azure Arc Kubernetes Cluster Admin	Lets you manage all resources in the cluster.	BuiltInRole	Management + Govern
Azure Kubernetes Fleet Manager RBAC Cluster Admin	Lets you manage all resources in the fleet manager cluster.	BuiltInRole	None
Azure Kubernetes Service RBAC Admin	Lets you manage all resources under cluster/namespace, except update or delete resource quotas and namespaces.	BuiltInRole	Containers
Azure Kubernetes Service RBAC Cluster Admin	Lets you manage all resources in the cluster.	BuiltInRole	Containers
Graph Owner	Create and manage all aspects of the Enterprise Graph - Ontology, Schema mapping, Conflation and Conversational AI and Ingestions	BuiltInRole	None
Log Analytics Contributor	Log Analytics Contributor can read all monitoring data and edit monitoring settings. Editing monitoring settings includes adding the VM extension to ...	BuiltInRole	Analytics
Log Analytics Reader	Log Analytics Reader can view and search all monitoring data as well as and view monitoring settings, including viewing the configuration of Azure dia...	BuiltInRole	Analytics
Logic App Contributor	Lets you manage logic app, but not access to them.	BuiltInRole	Integration
Logic App Operator	Lets you read, enable and disable logic app.	BuiltInRole	Integration
Logic Apps Standard Contributor (Preview)	You can manage all aspects of a Standard logic app and workflows. You can't change access or ownership.	BuiltInRole	None
Logic Apps Standard Developer (Preview)	You can create and edit workflows, connections, and settings for a Standard logic app. You can't make changes outside the workflow scope.	BuiltInRole	None

Step 17: Now, click on *Select members*, select the application that was created similar to the previous role. Finally, click on *Review and Assign*.

Microsoft Azure

Search resources, services, and docs (G+/)

Home > Subscriptions > Microsoft Azure Sponsorship | Access control (IAM) >

Add role assignment

Role | Members | Review + assign

Selected role Log Analytics Reader

Assign access to User, group, or service principal / Managed identity

Members + Select members

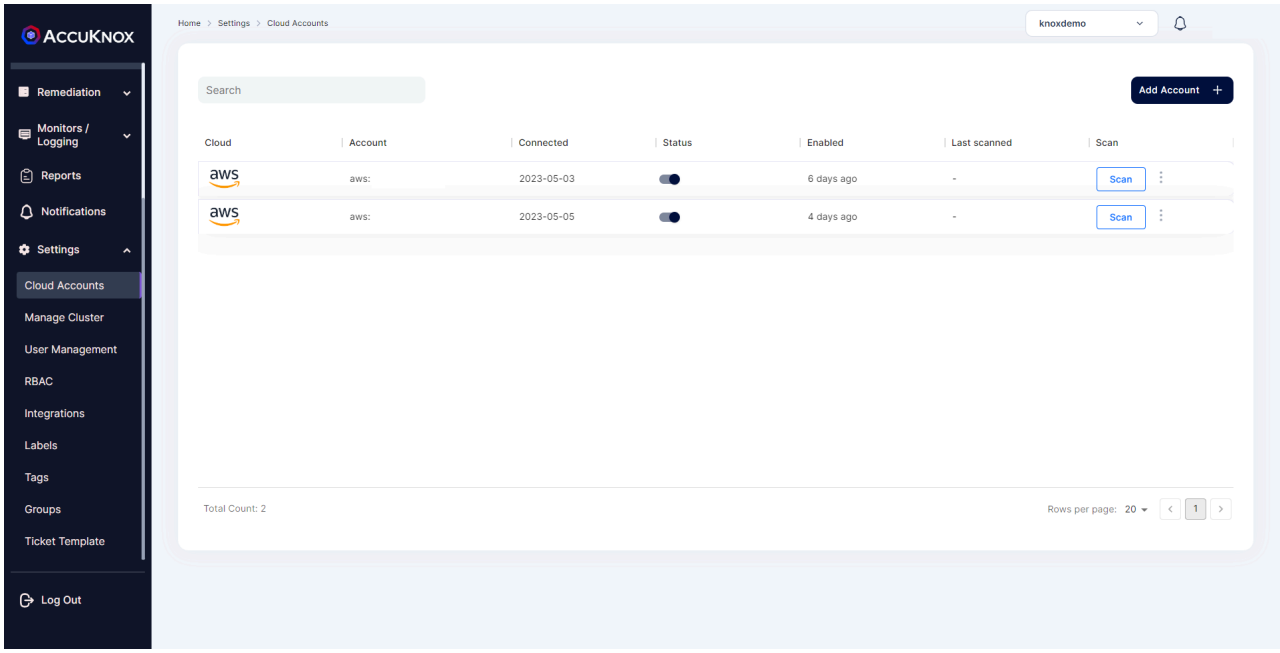
Name	Object ID	Type
No members selected		

Description Optional

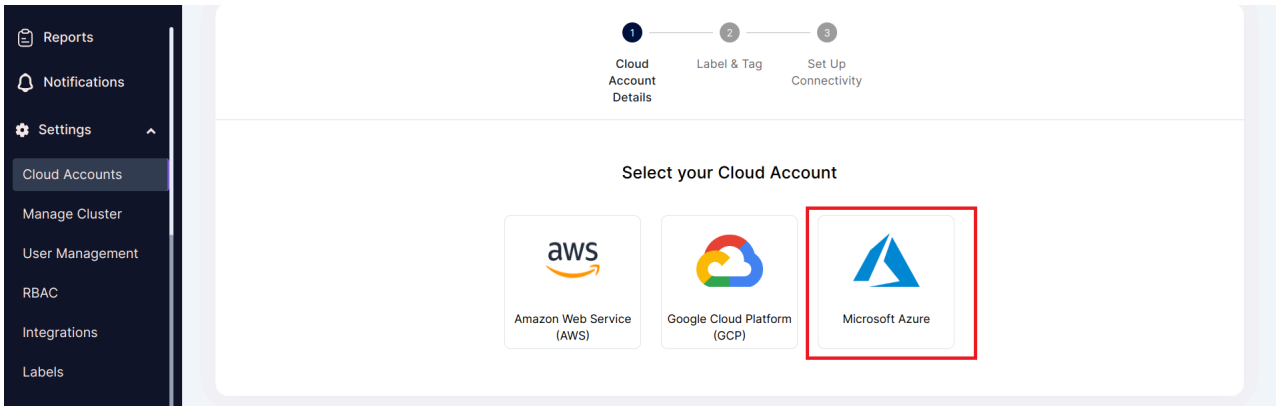
From AccuKnox SaaS UI

Configuring your Azure cloud account is complete, now we need to onboard the cloud account onto AccuKnox Saas Platform.

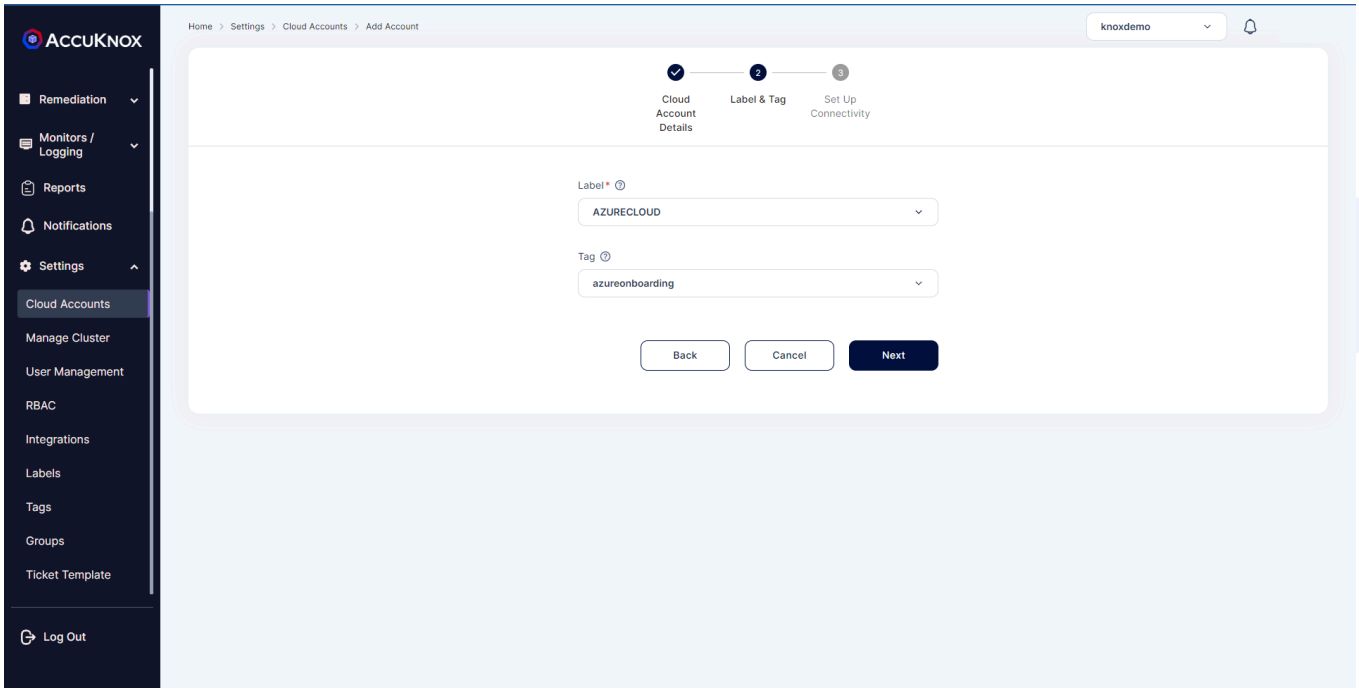
Step 1: Go to settings→ Cloud Account and click on Add Account



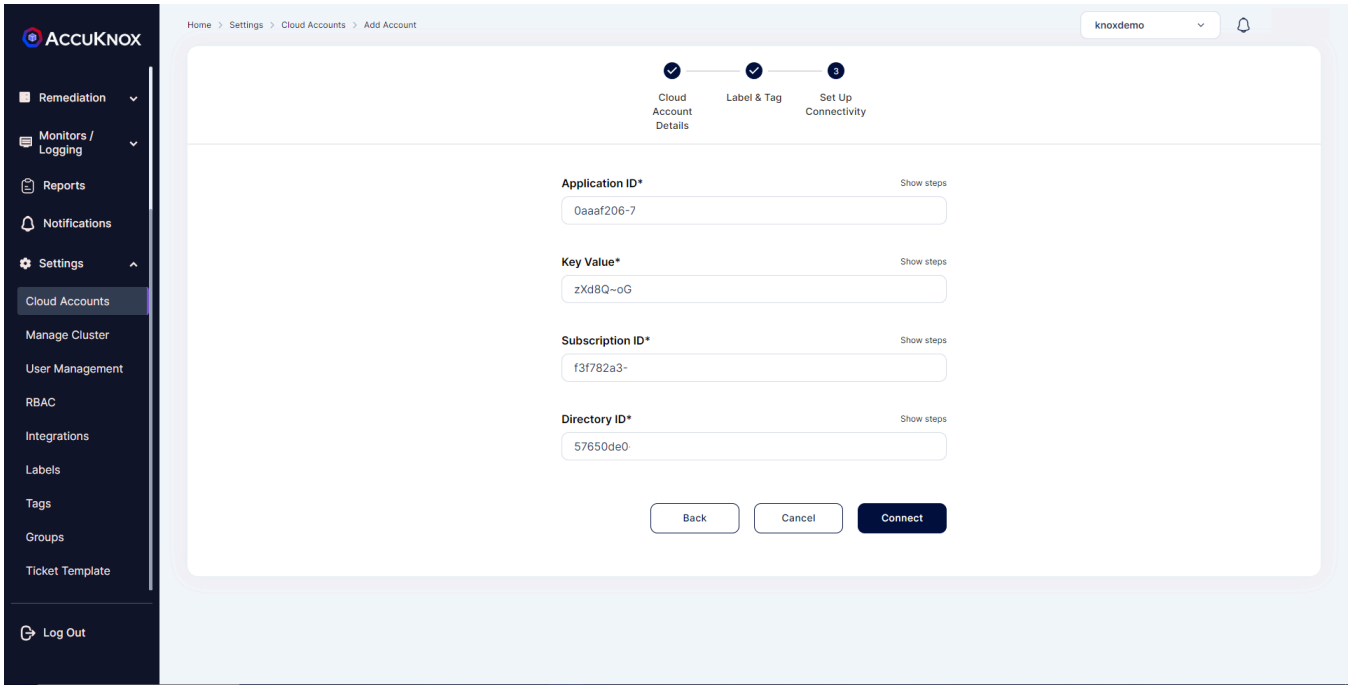
Step 2: Select Microsoft Azure as Cloud Account Type



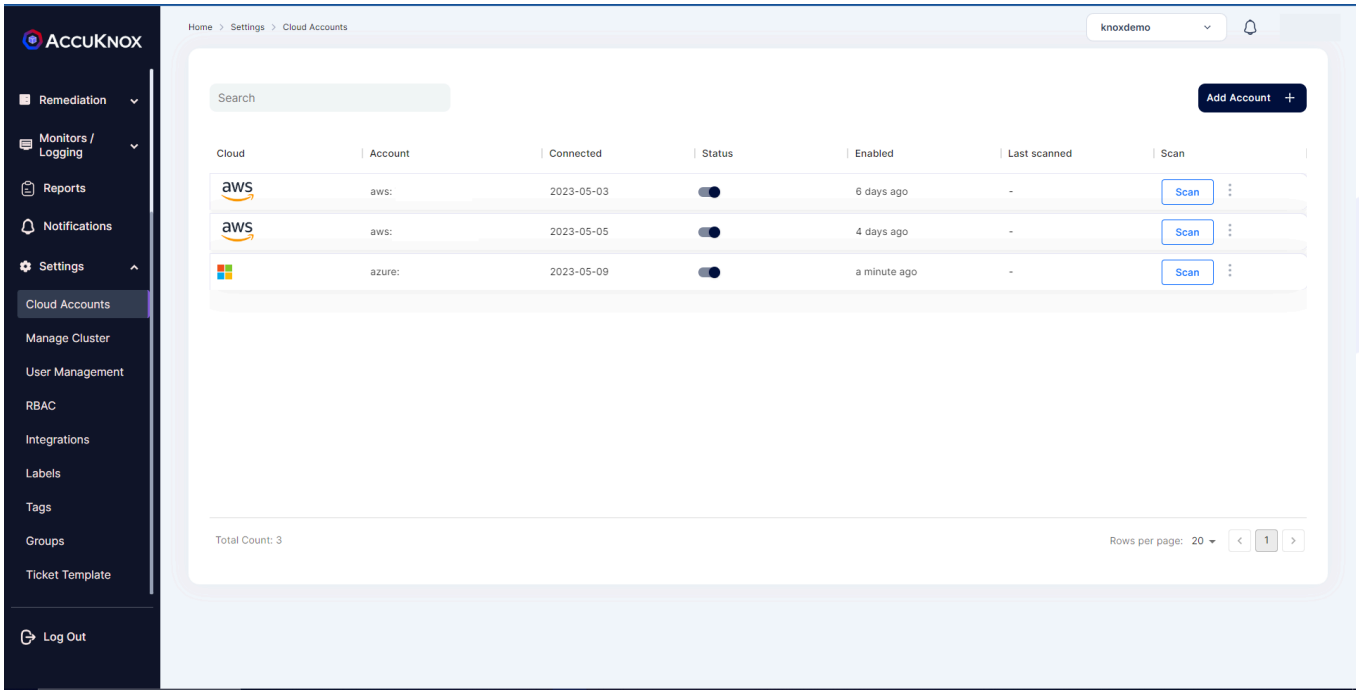
Step 3: Select or create label and Tags that will be associated with this Cloud Account



Step 4: Enter the details that we saved earlier during the steps for app registration and subscription id from subscriptions in azure portal and click on connect

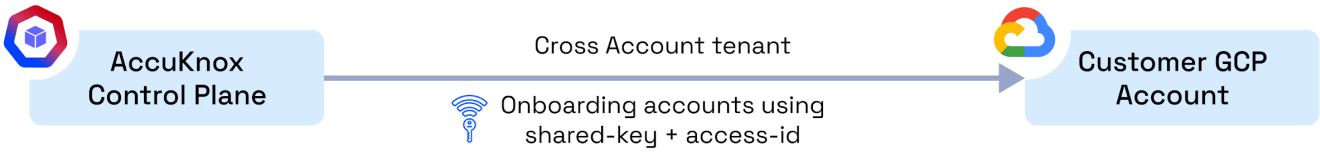


Step 5: After successfully connecting your cloud account will show up in the list



CSPM Pre-requisite for GCP

When the AccuKnox control plane is hosted in a cloud environment, scanning is performed using Cloud account Readonly Access permissions.

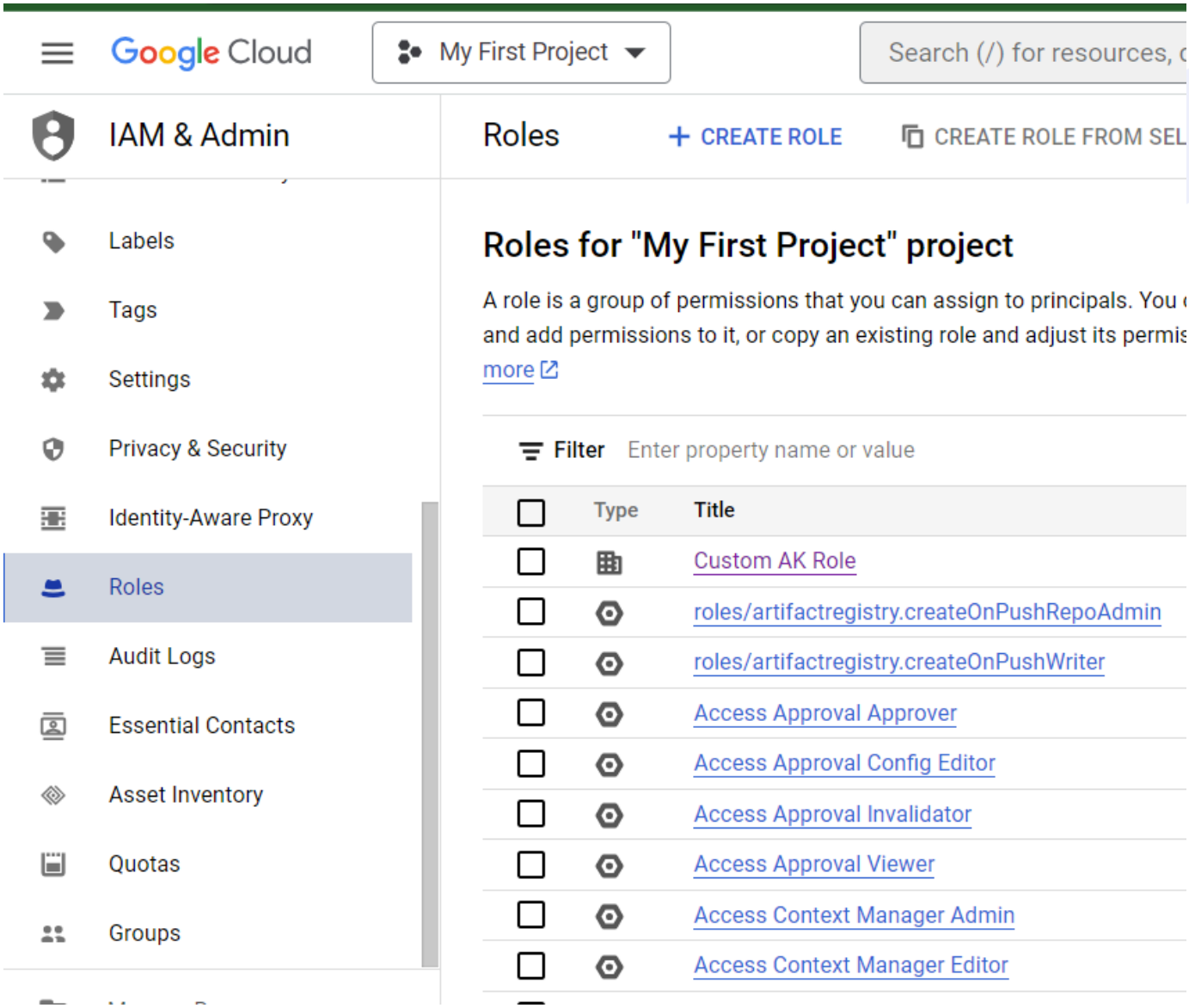


Note: Make sure the Below API Library is enabled in your GCP Account for onboarding into AccuKnox SaaS:

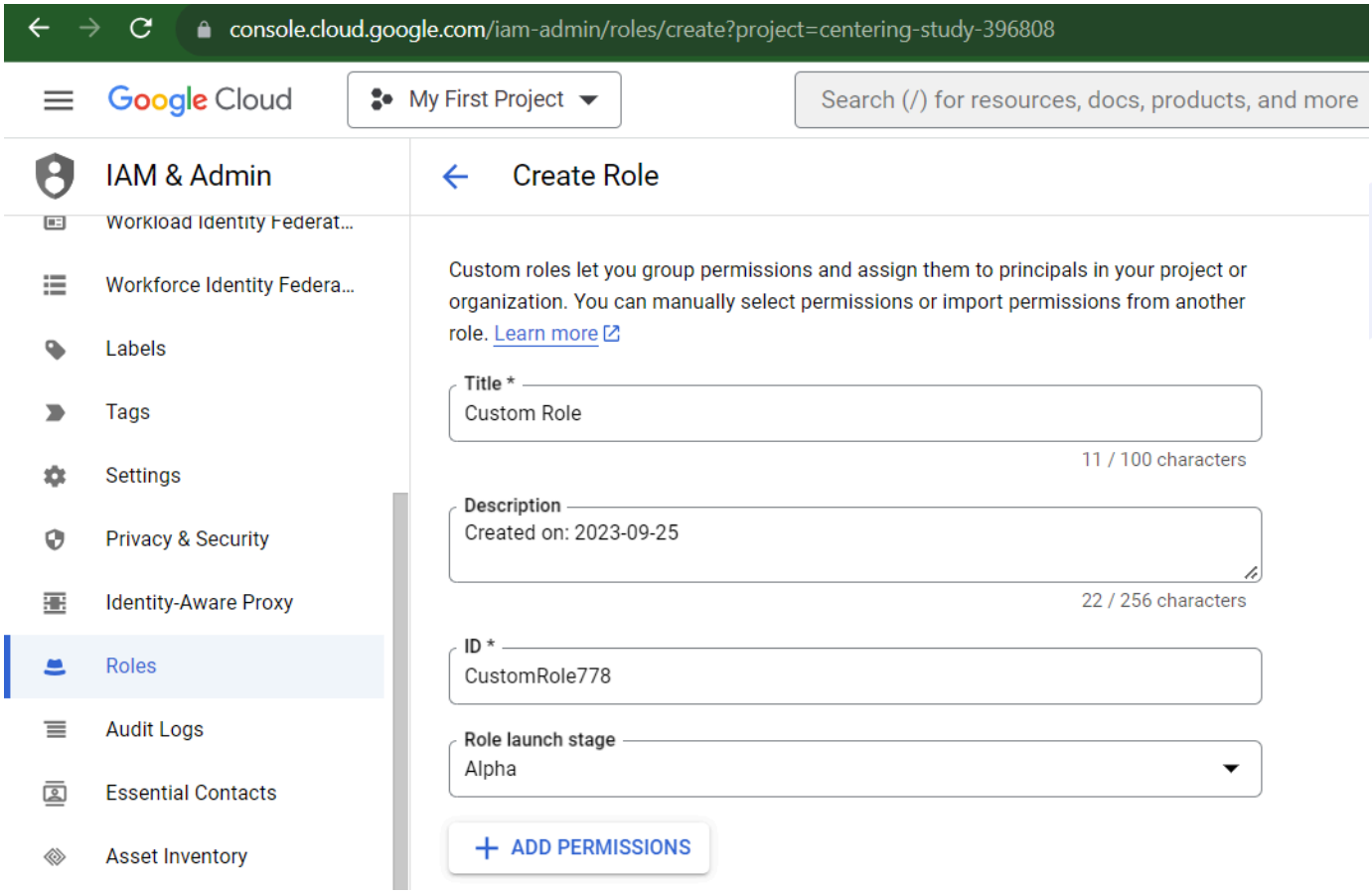
1. Compute Engine API
2. Identity and Access Management (IAM) API
3. Cloud Resource Manager API
4. Cloud Functions API
5. KMS API
6. Kubernetes API
7. Cloud SQL Admin API

For GCP there is a requirement for IAM Service Account Access.

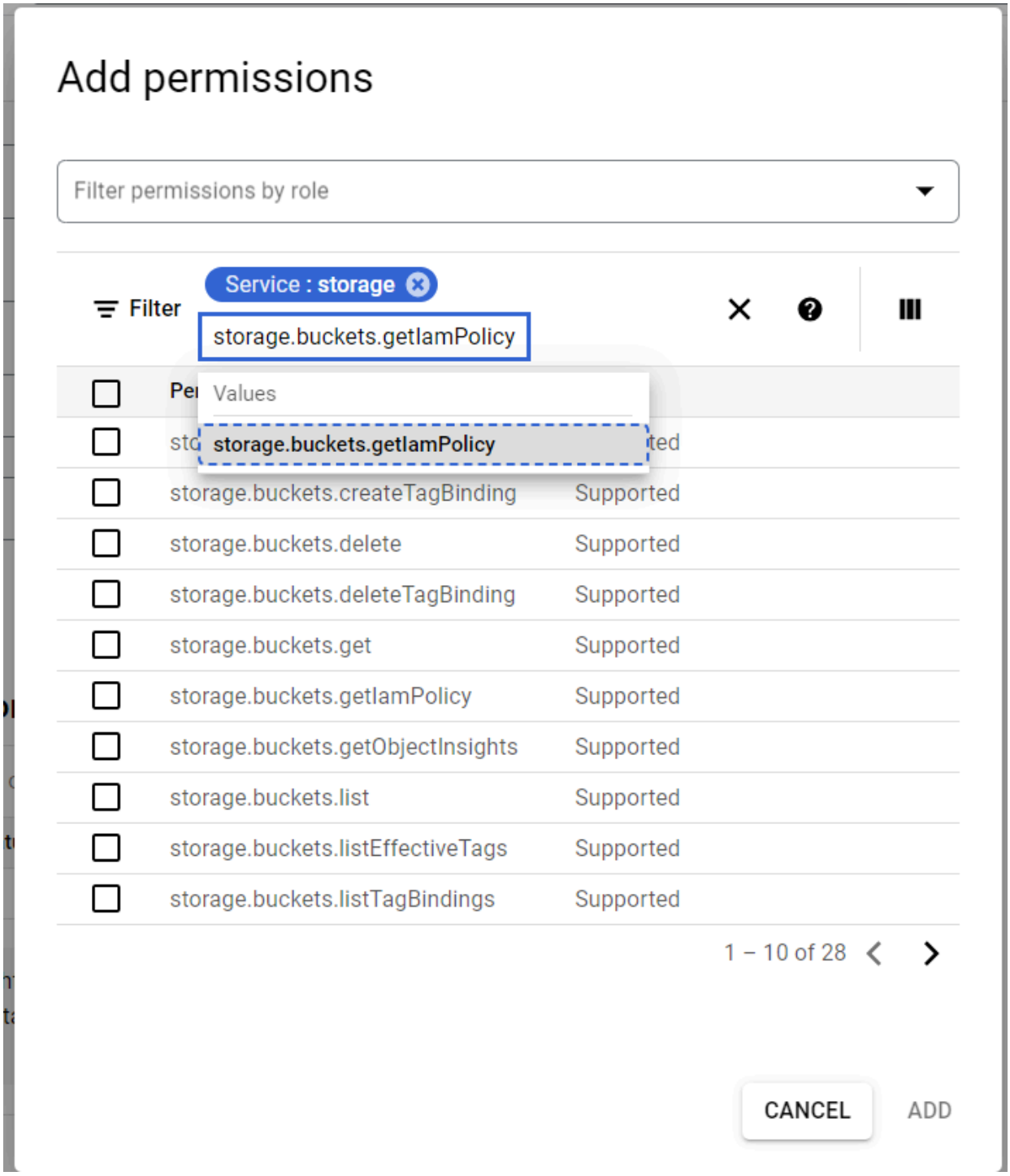
Step 1: Log into your Google Cloud console and navigate to IAM & Admin choose “Roles“ and Click “Create Role“



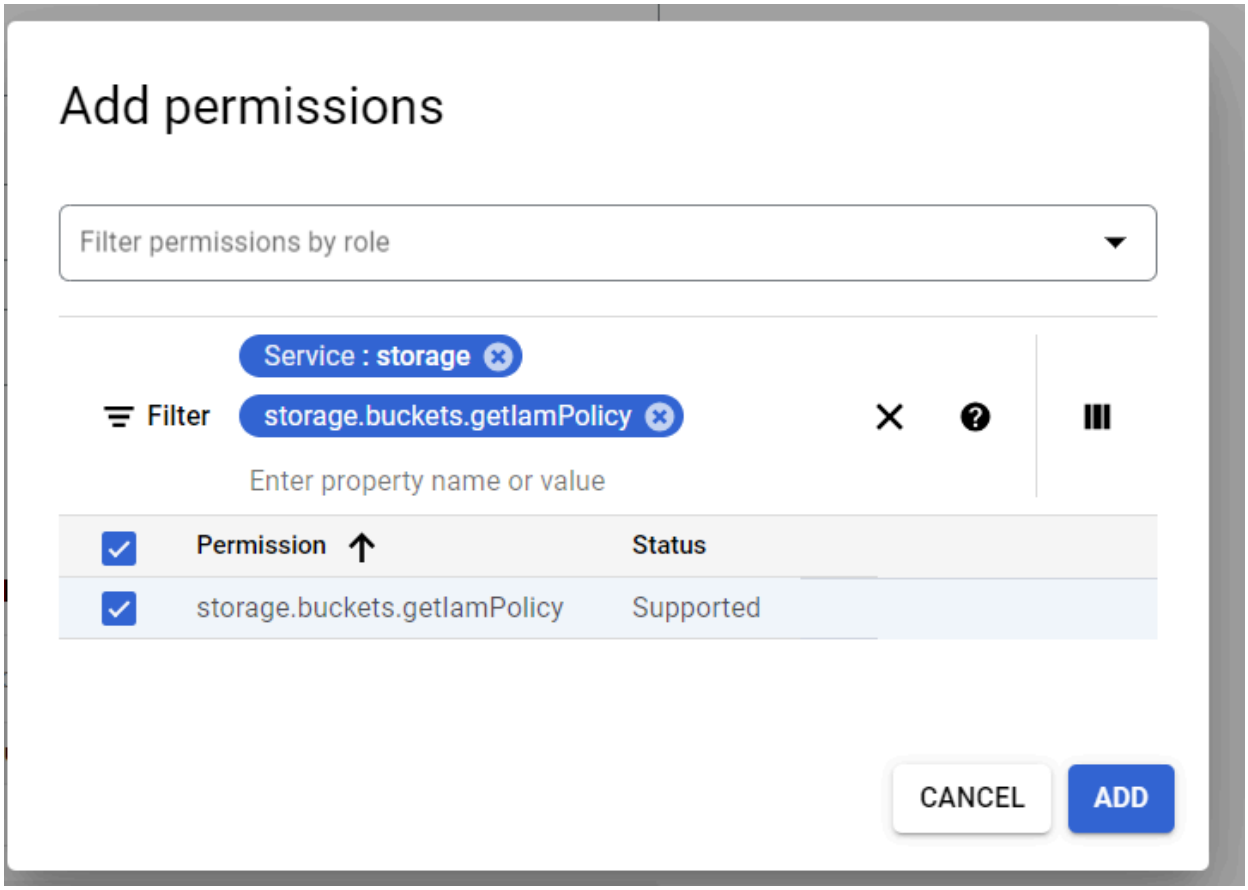
Step 2: Name the “Role” and Click “Add Permission”



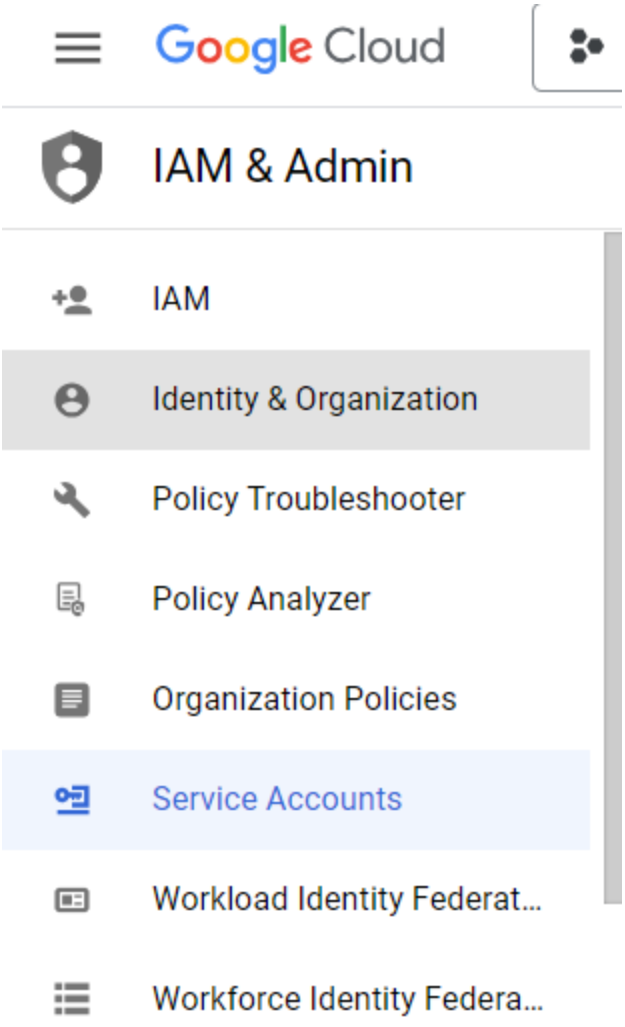
Step 3: Use the Service: storage filter then value as “storage.buckets.getIamPolicy“



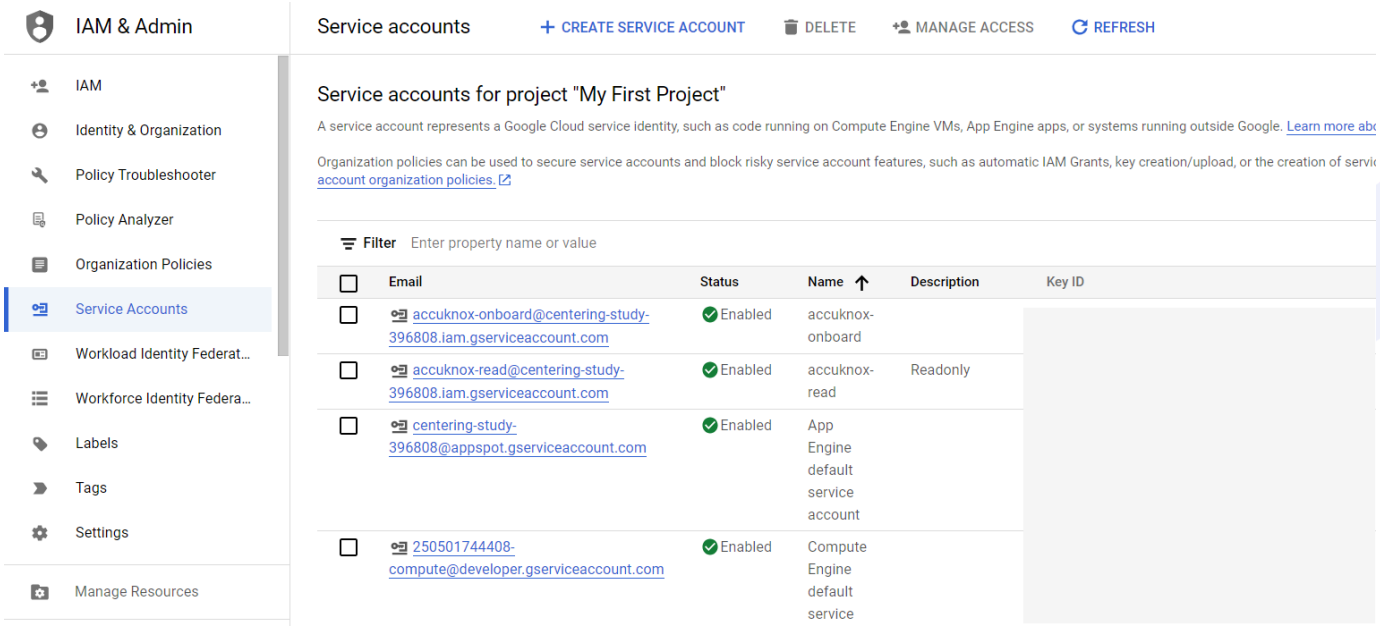
Step 4: Choose the permission and Click “Add“ then Click Create in the same page.



Step 5: In the Navigation Panel, navigate to IAM Admin > Service Accounts.

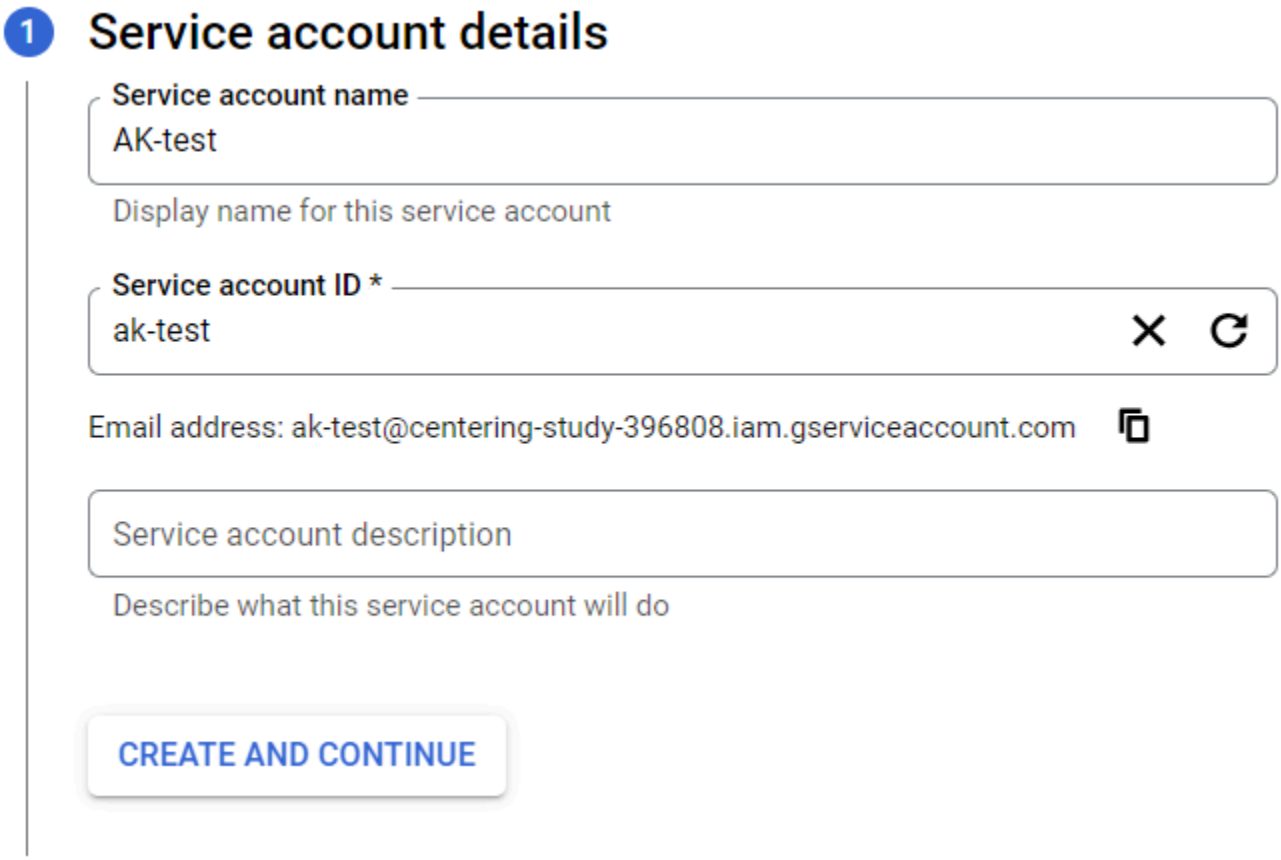


Step 6: Click on "Create Service Account"

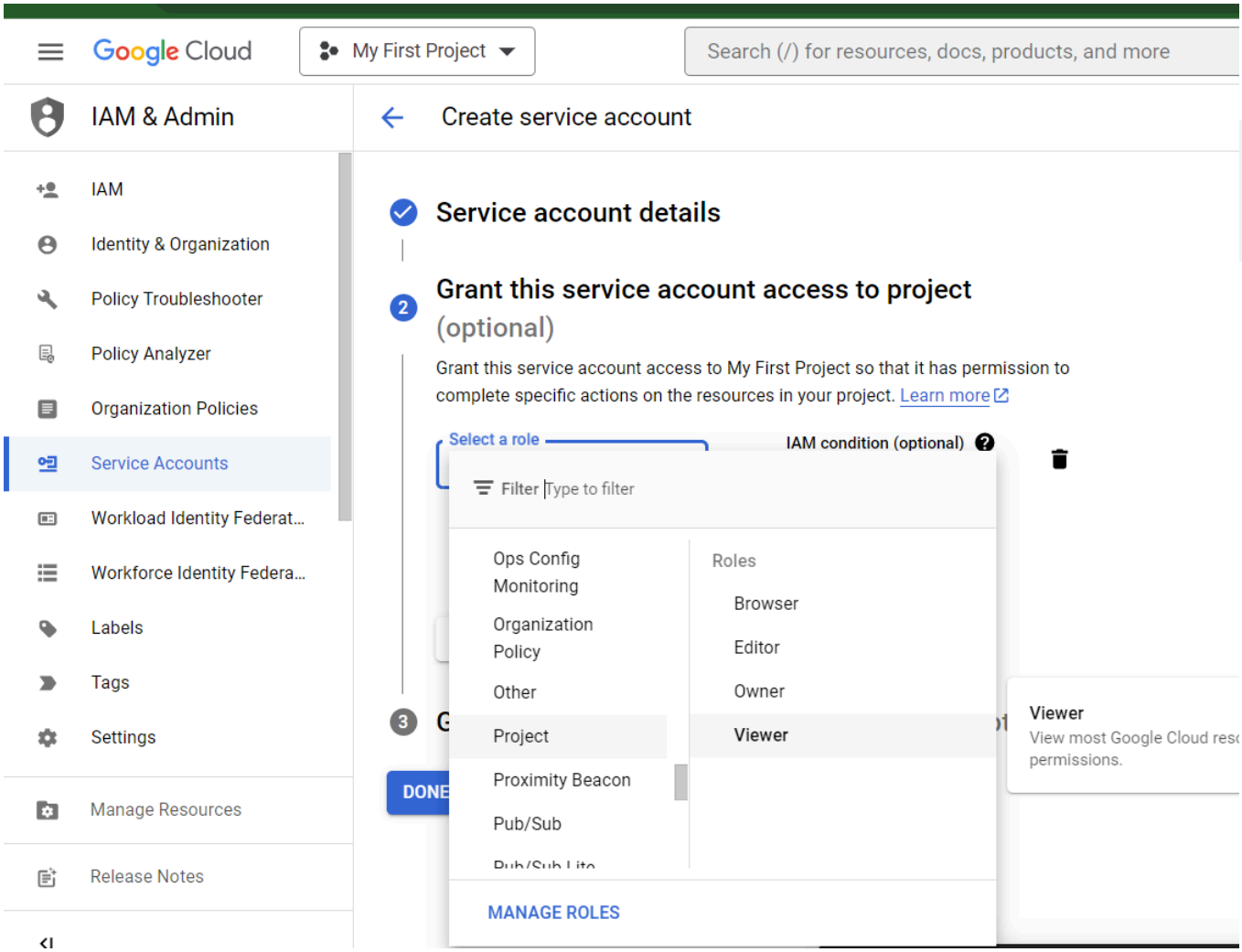


Step 7: Enter any name that you want on Service Account Name.

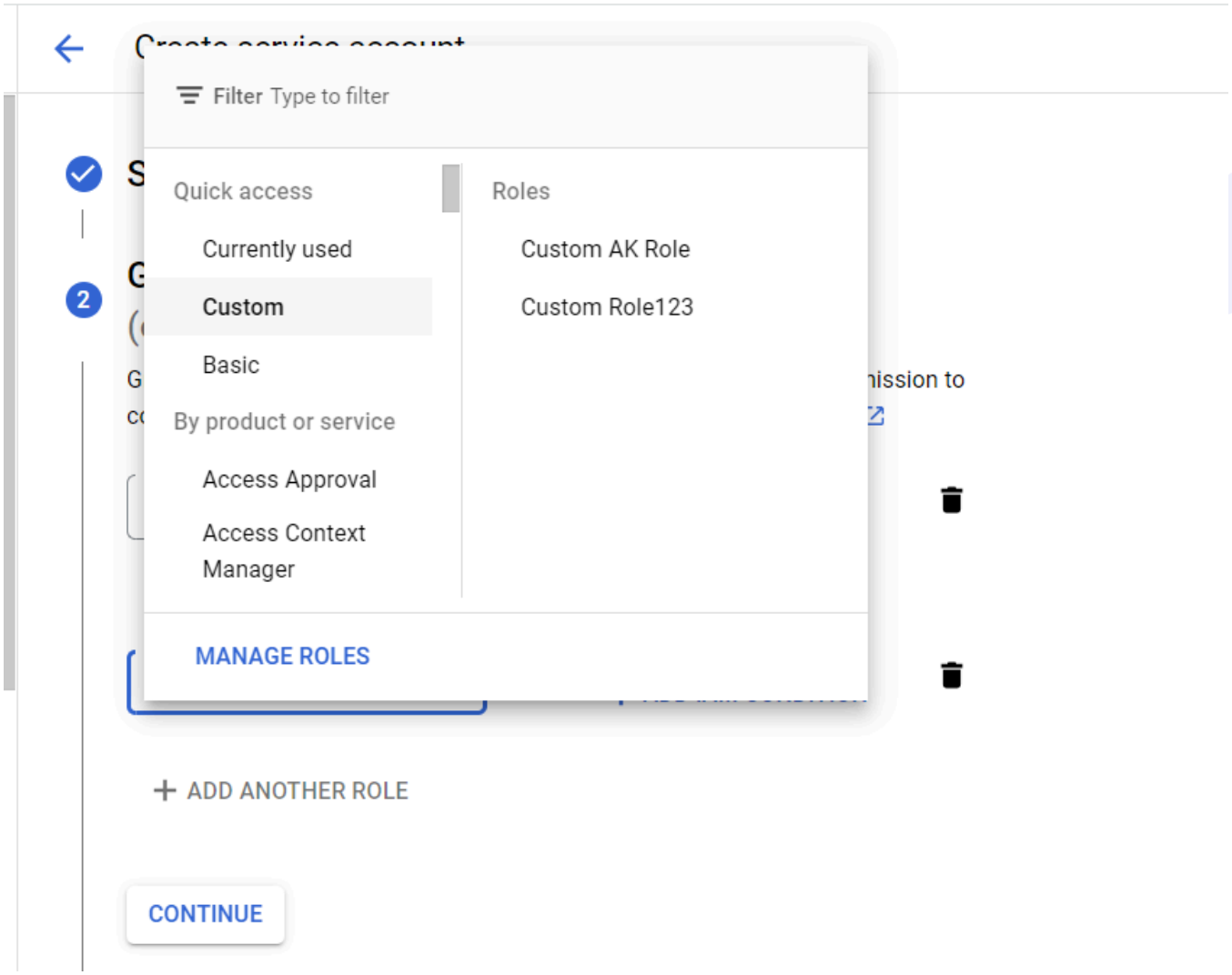
Step 8: Click on Continue.



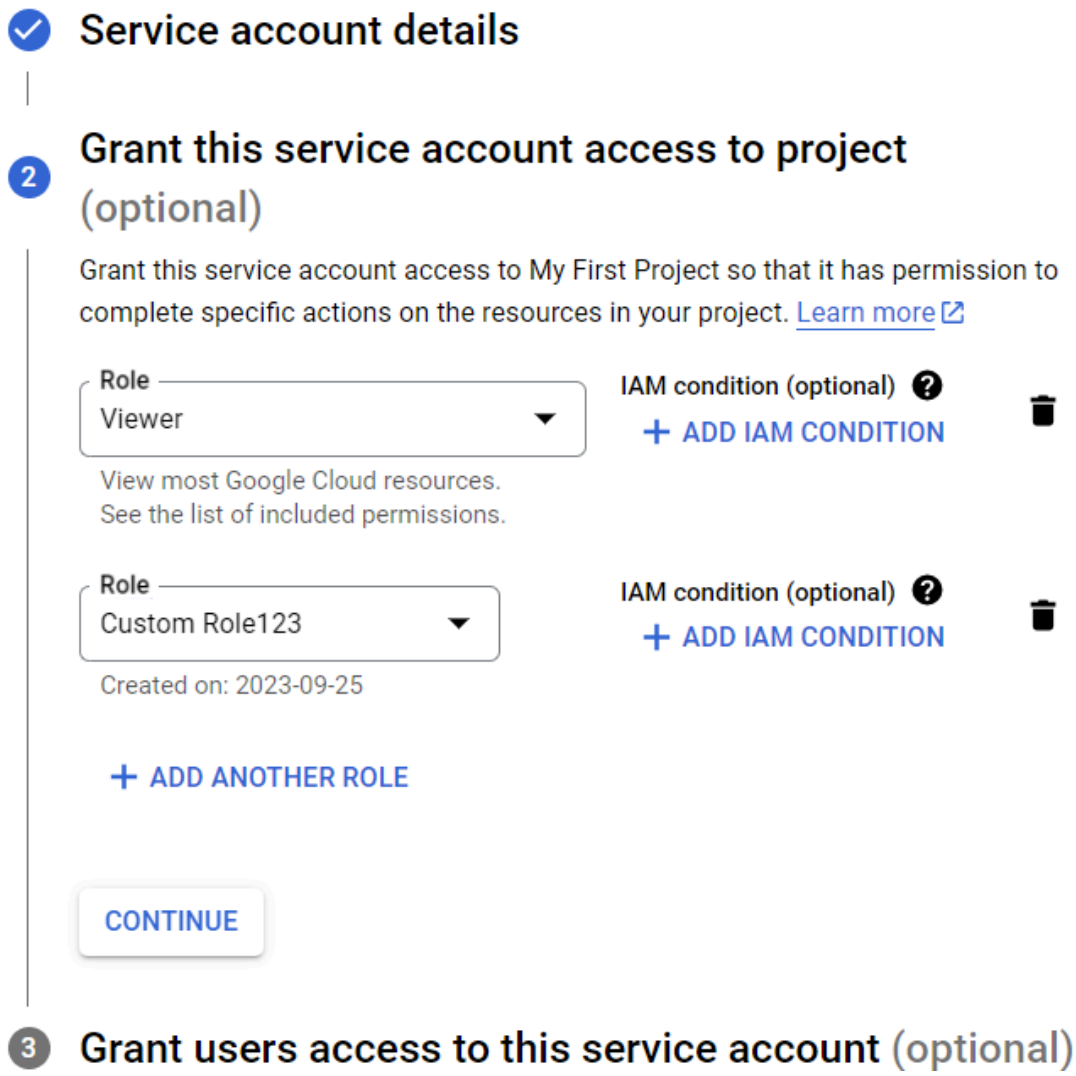
Step 9: Select the role: Project > Viewer and click Add another Role.



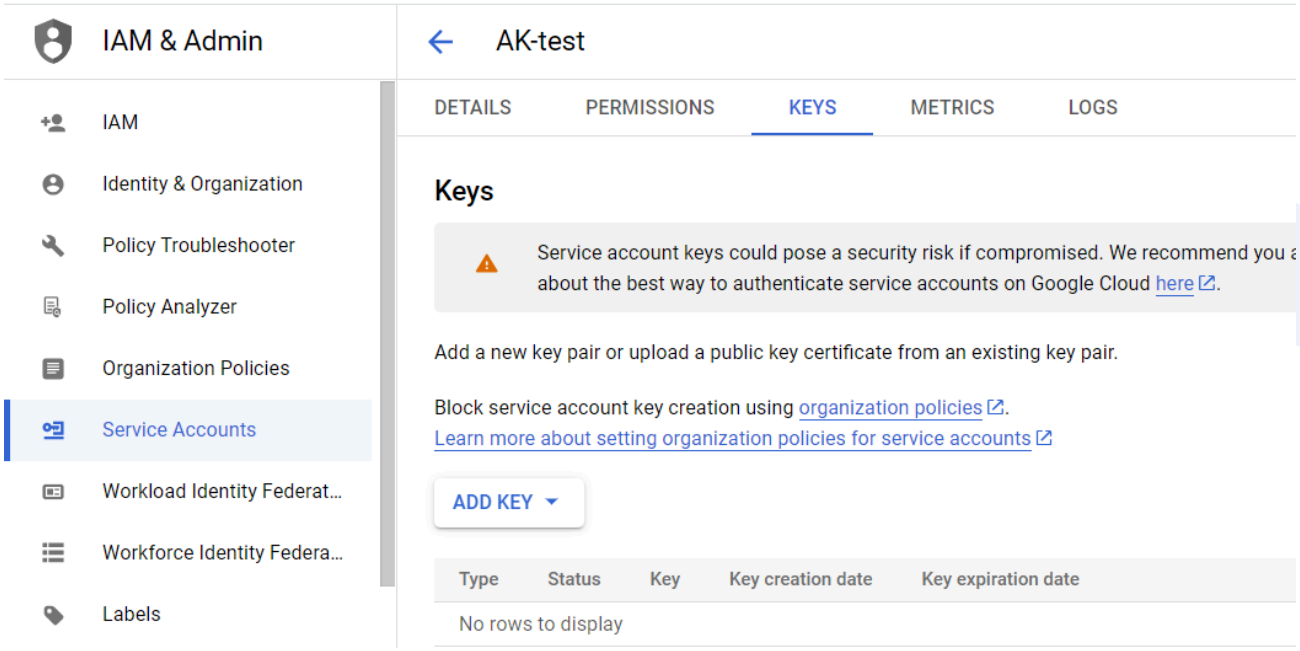
Step 10: Click “Add Another Role” Choose “Custom“ Select the created Custom Role.



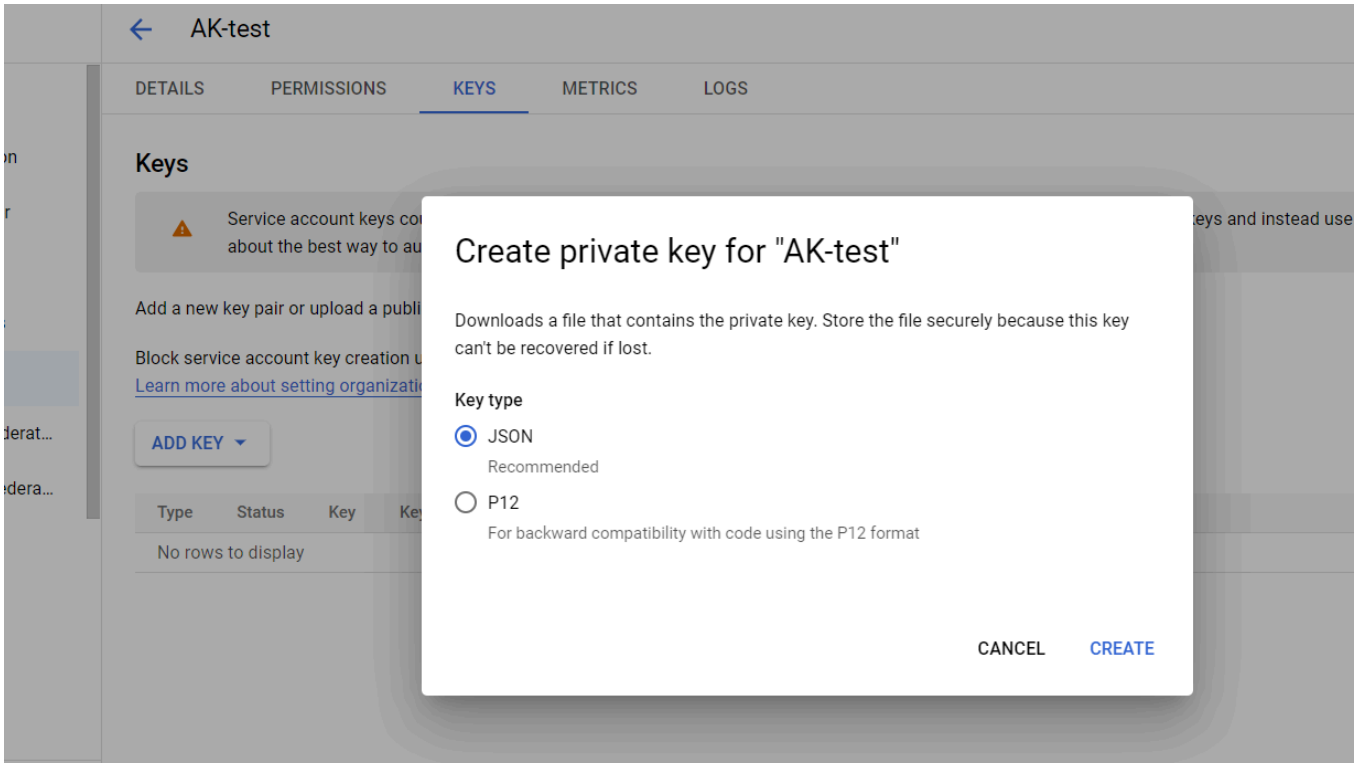
Step 11: Click on “Continue“ and ”Done”



Step 12: Go to the created Service Account, click on that Service Account navigate to the “Keys“ section.



Step 13: Click the “Add key“ button and “Create new key “ . Chosen Key type should be JSON format.



Step 14: Click the “Create“ button it will automatically download the JSON key.

GCP Account onboarding

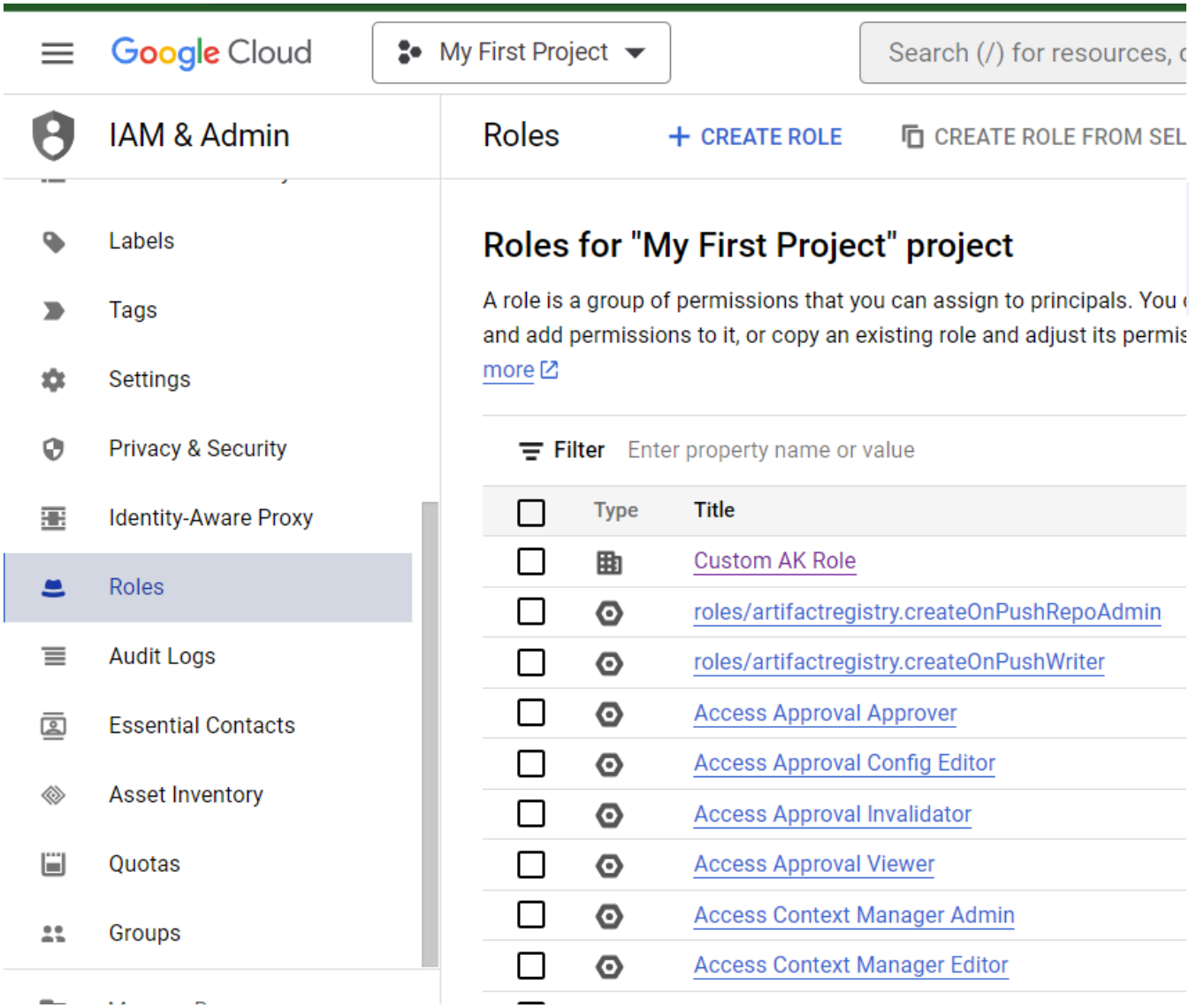
Here, we will see the steps to onboard a GCP cloud account to the AccuKnox SaaS platform

Note: Make sure the Below API Library is enabled in your GCP Account for onboarding into AccuKnox SaaS:

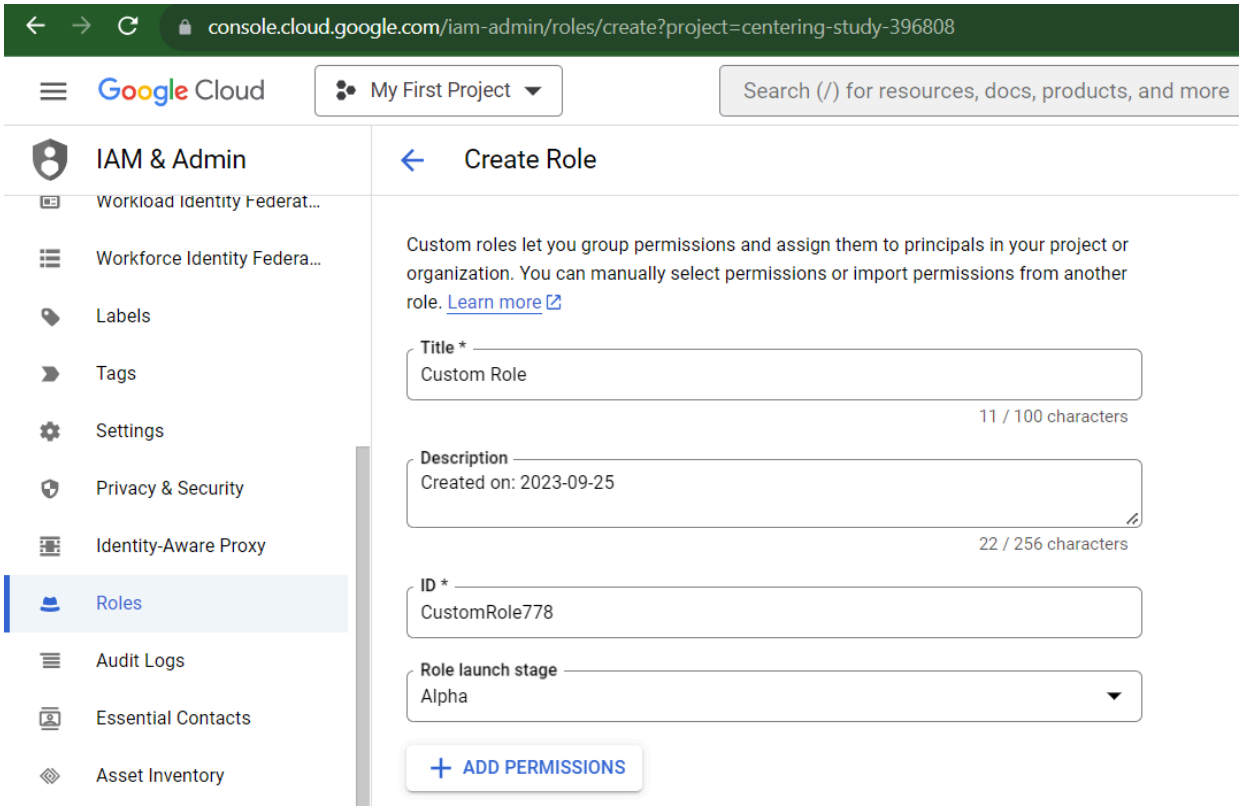
1. Compute Engine API
2. Identity and Access Management (IAM) API
3. Cloud Resource Manager API
4. Cloud Functions API
5. KMS API
6. Kubernetes API
7. Cloud SQL Admin API

For GCP there is a requirement for IAM Service Account Access.

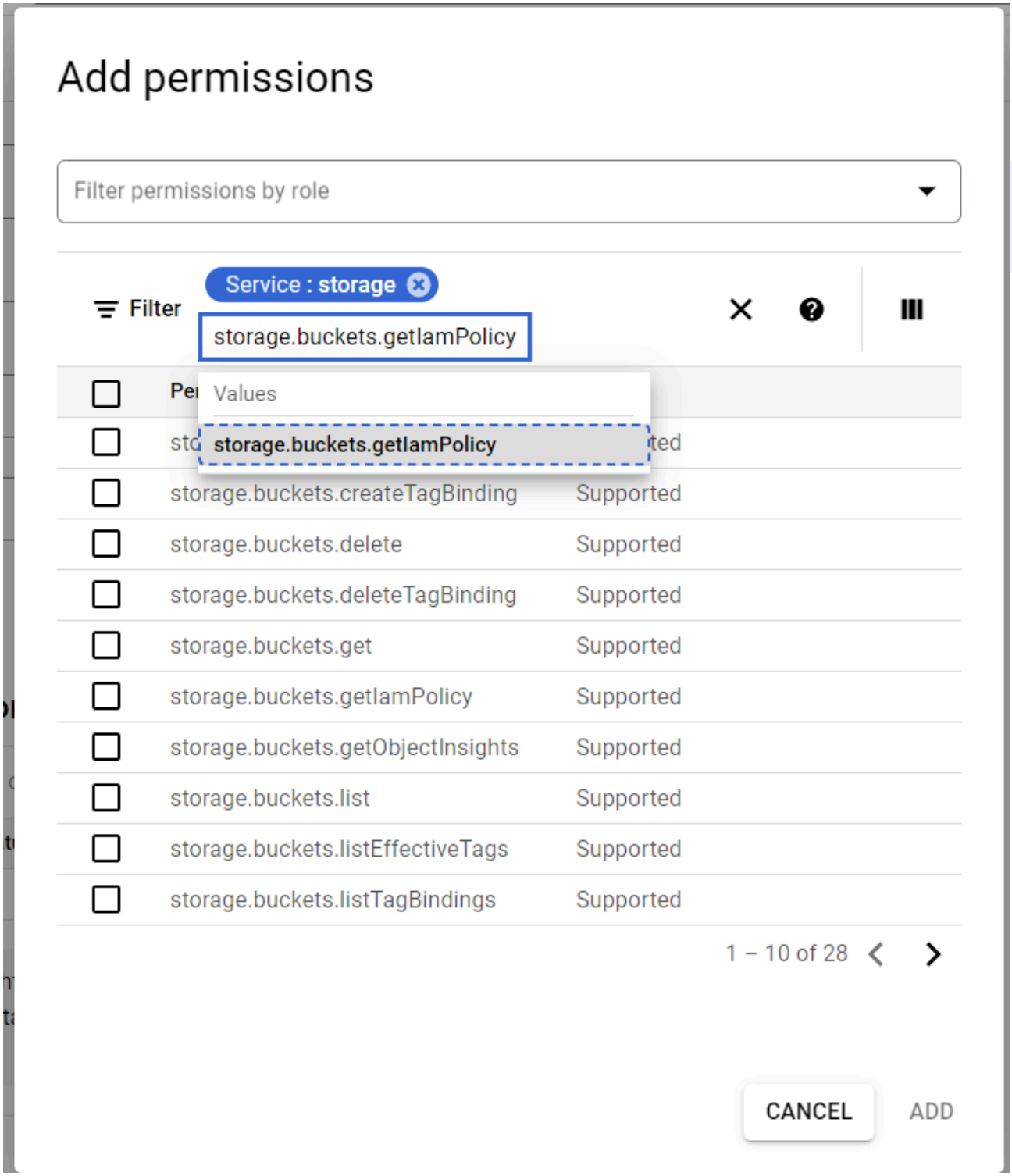
Step 1: Log into your Google Cloud console and navigate to IAM & Admin choose “Roles“ and Click “Create Role“



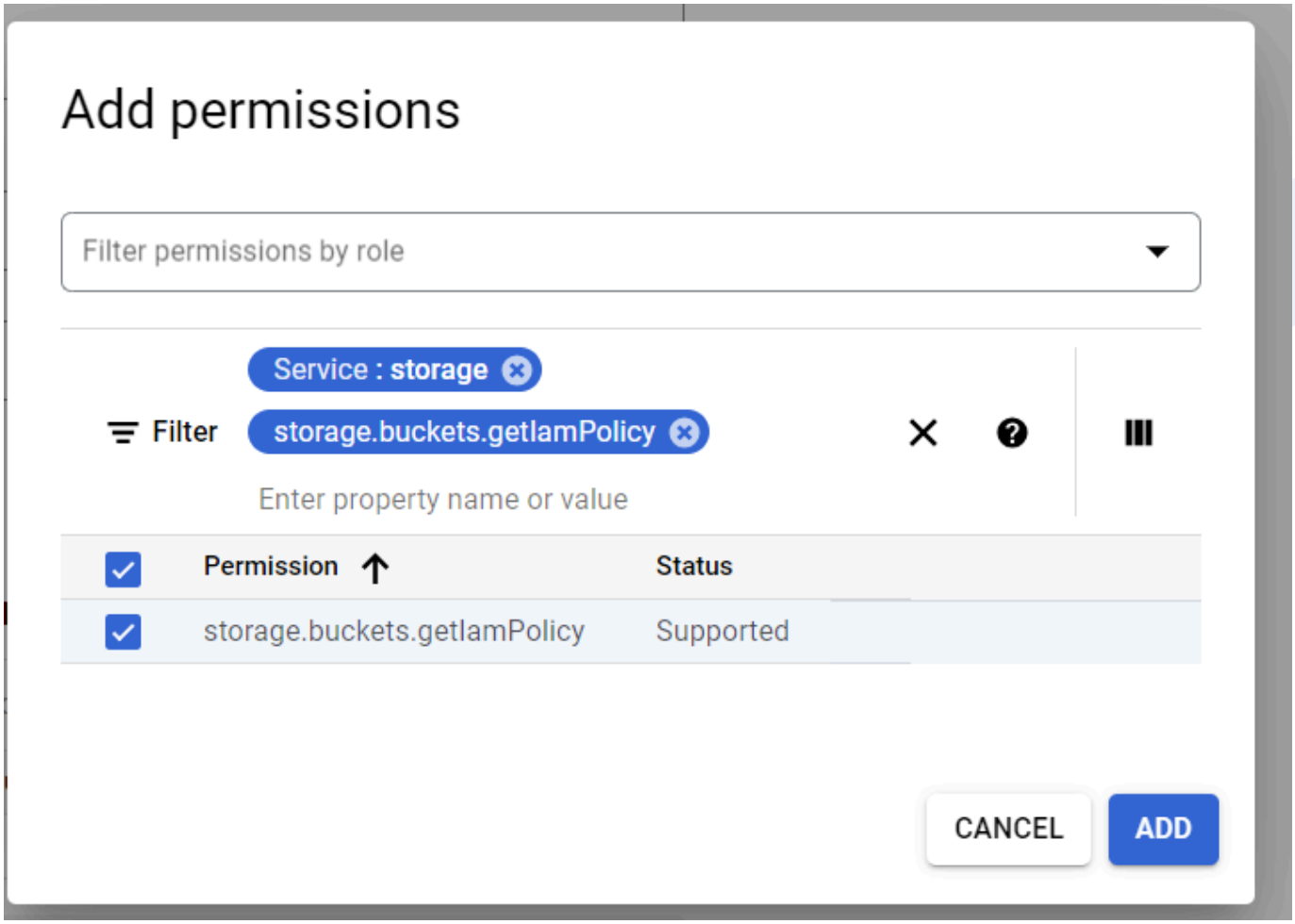
Step 2: Name the “Role” and Click “Add Permission”



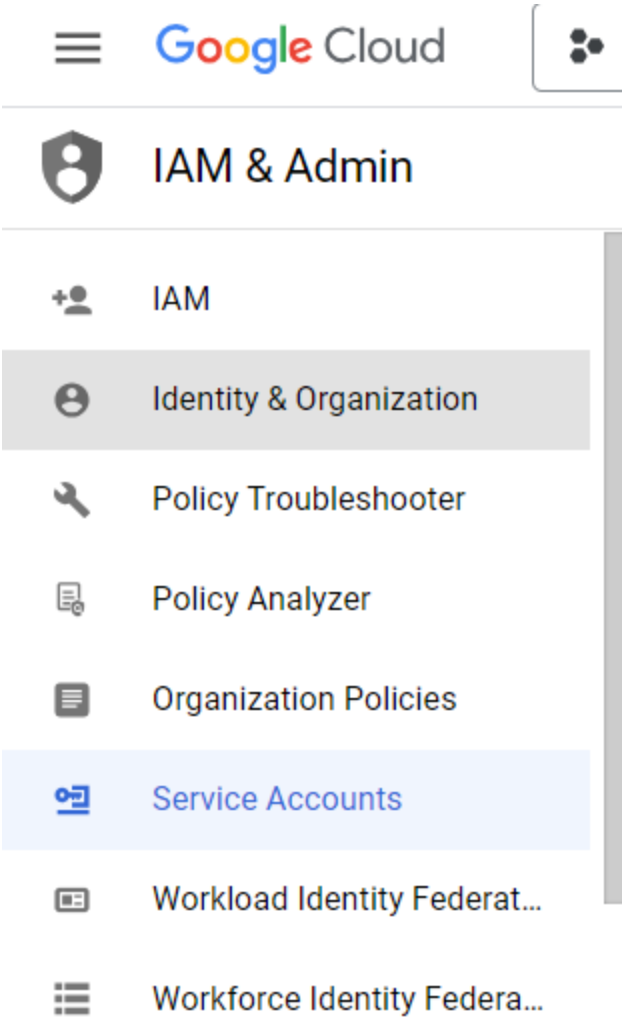
Step 3: Use the Service: storage filter then value as “storage.buckets.getIamPolicy“



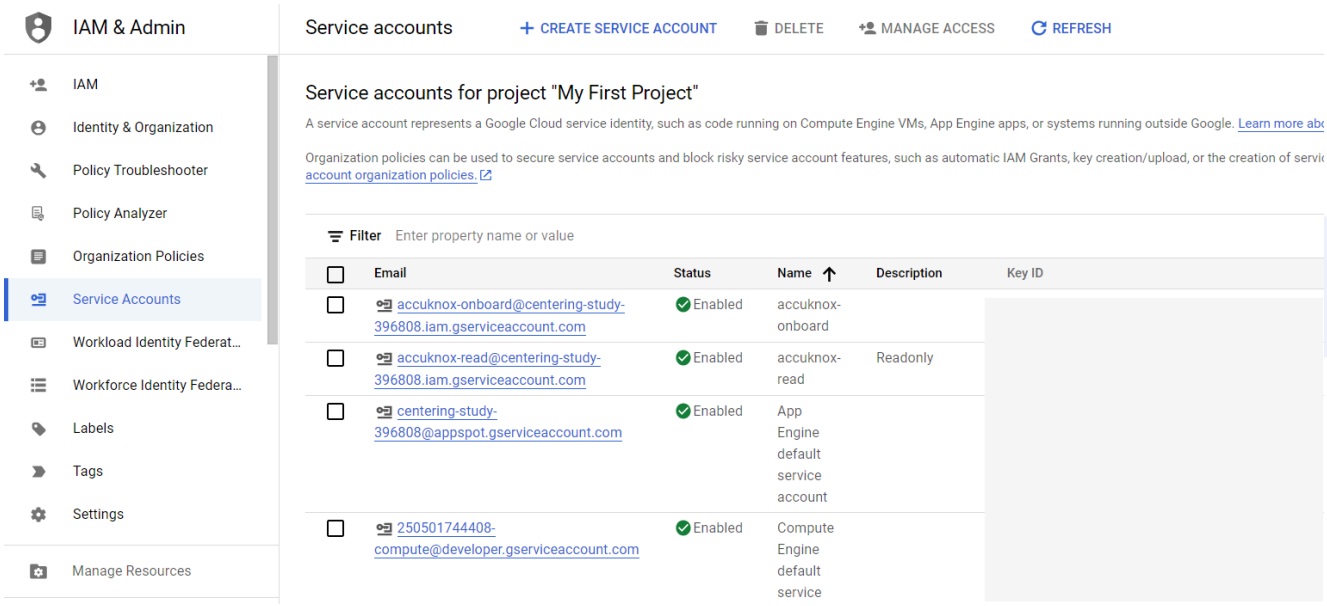
Step 4: Choose the permission and Click “Add“ then Click Create in the same page.



Step 5: In the Navigation Panel, navigate to IAM Admin > Service Accounts.

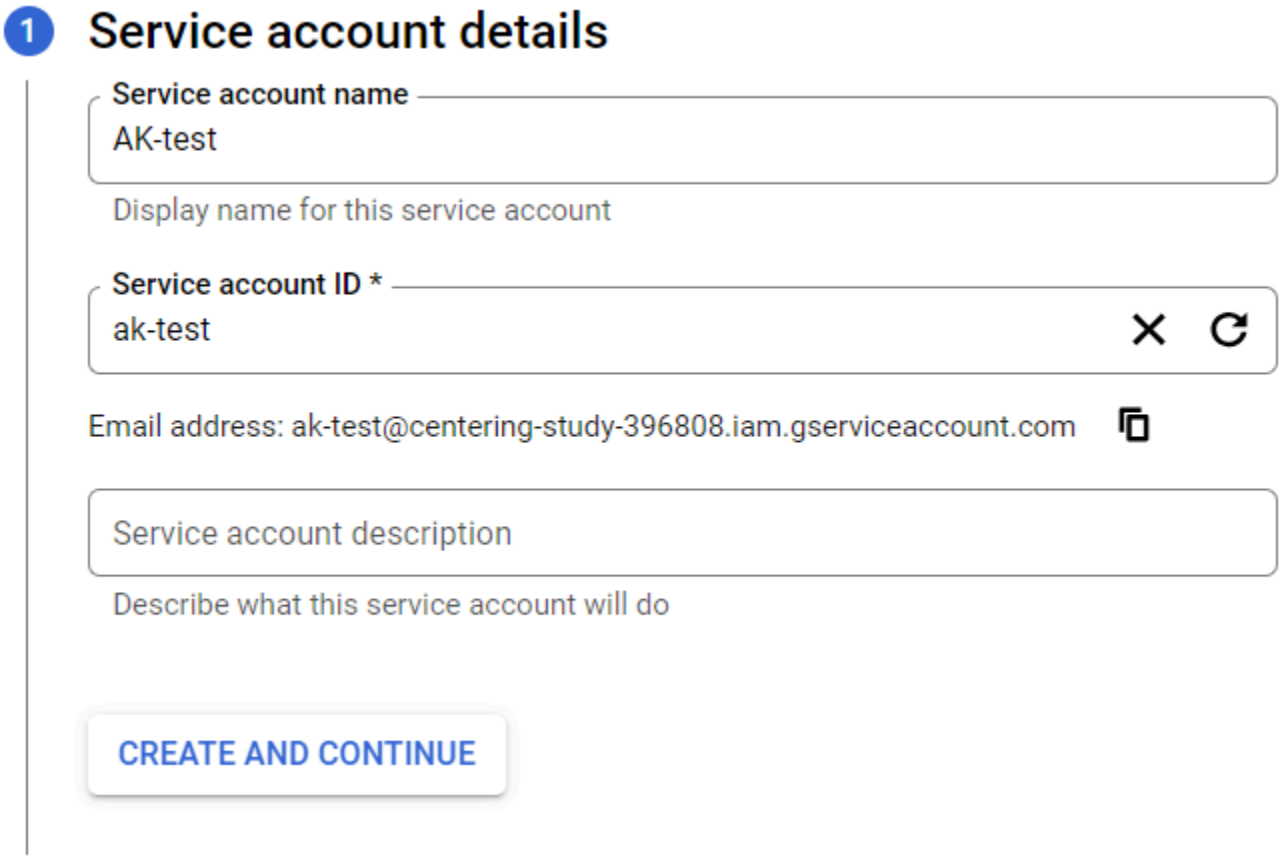


Step 6: Click on "Create Service Account"

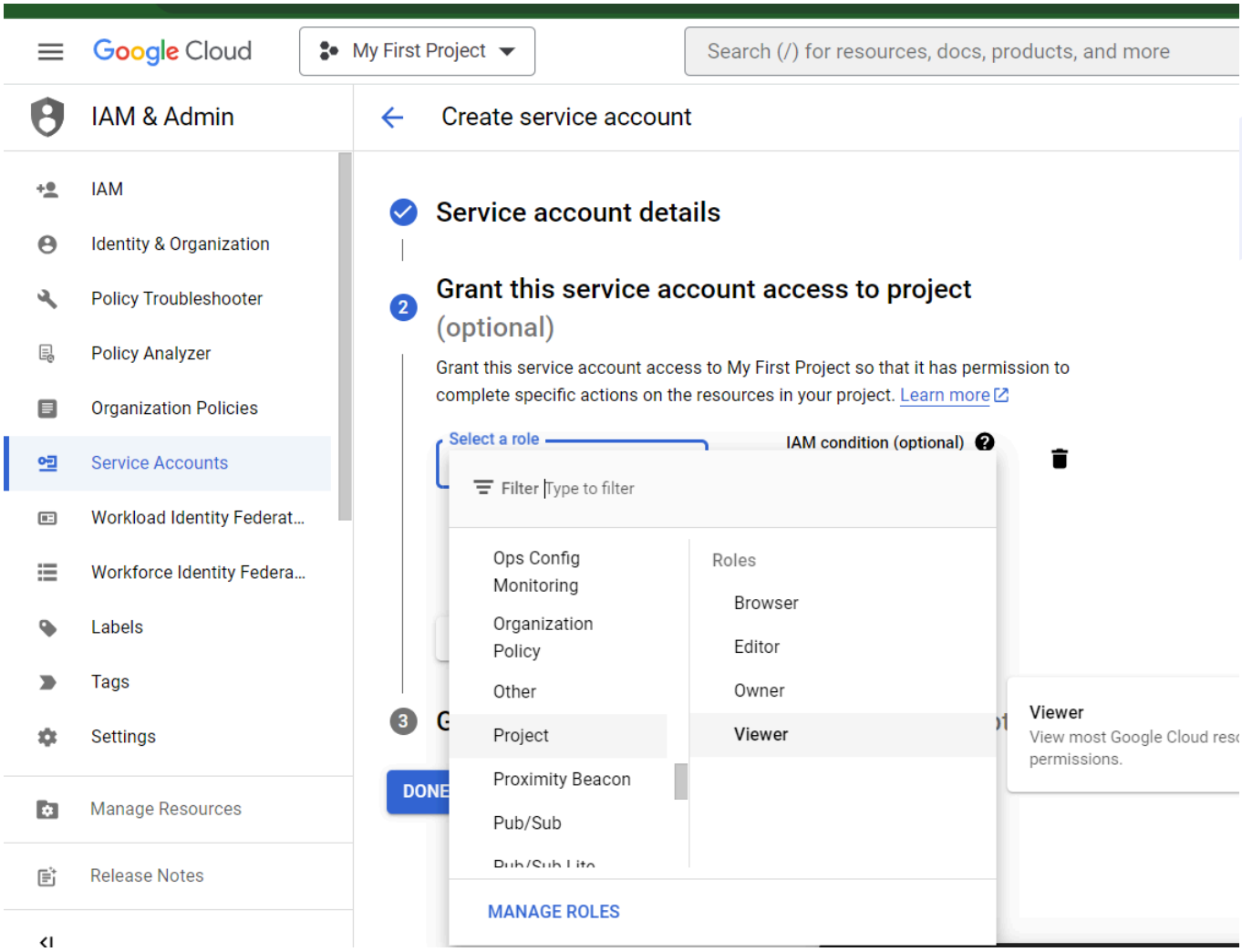


Step 7: Enter any name that you want on Service Account Name.

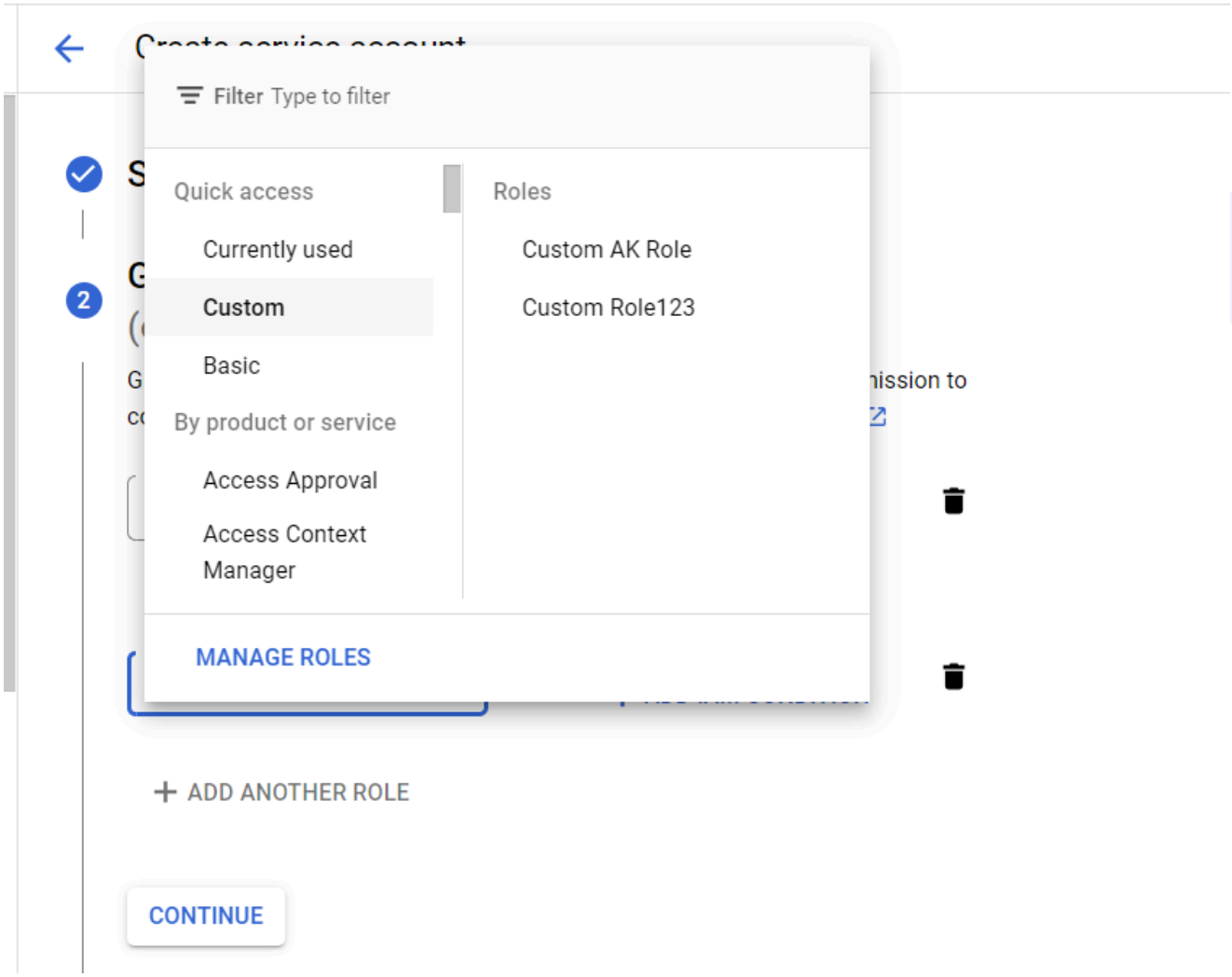
Step 8: Click on Continue.



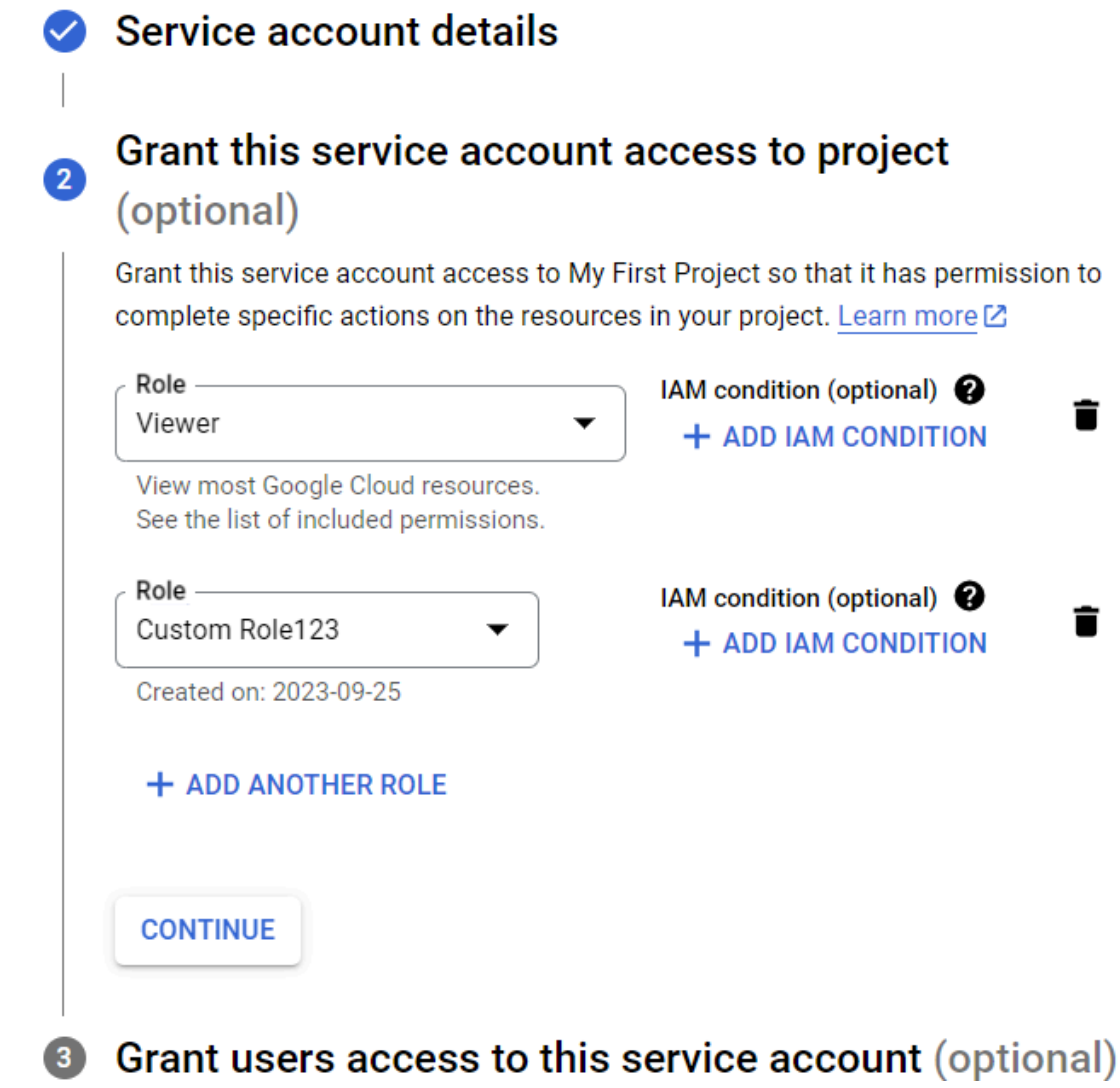
Step 9: Select the role: Project > Viewer and click Add another Role.



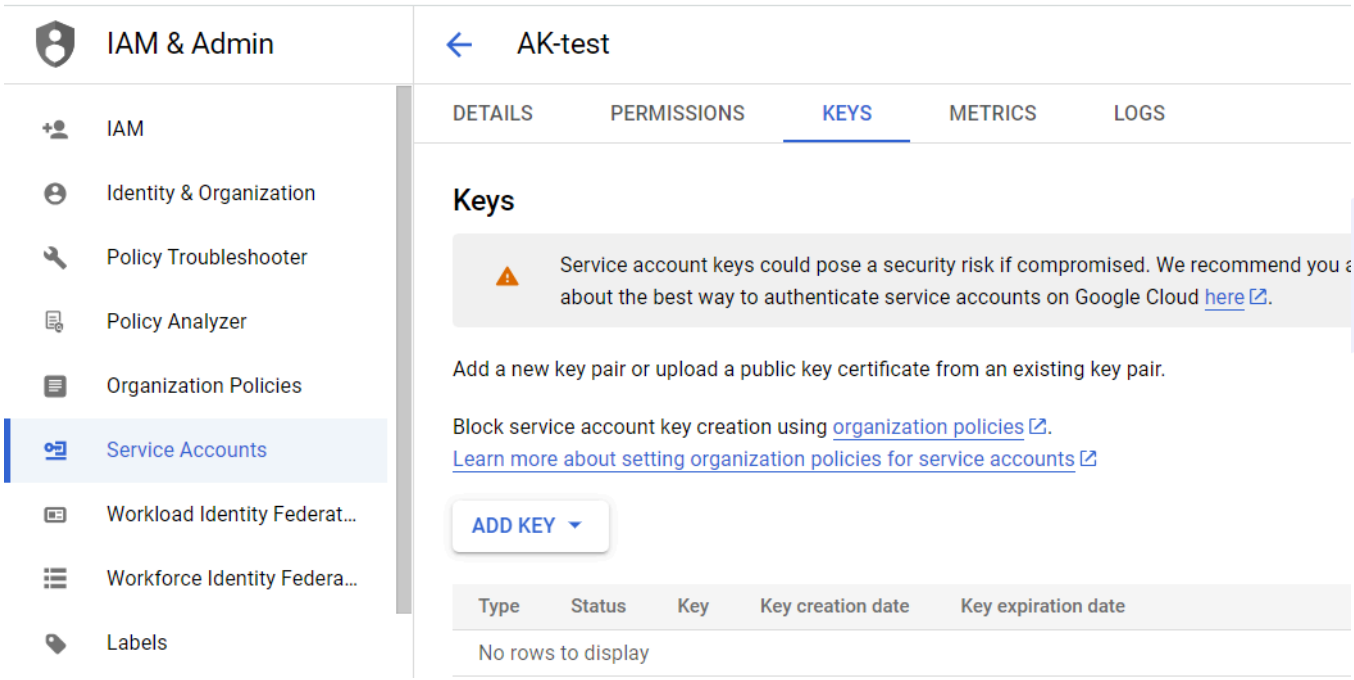
Step 10: Click “Add Another Role” Choose “Custom“ Select the created Custom Role.



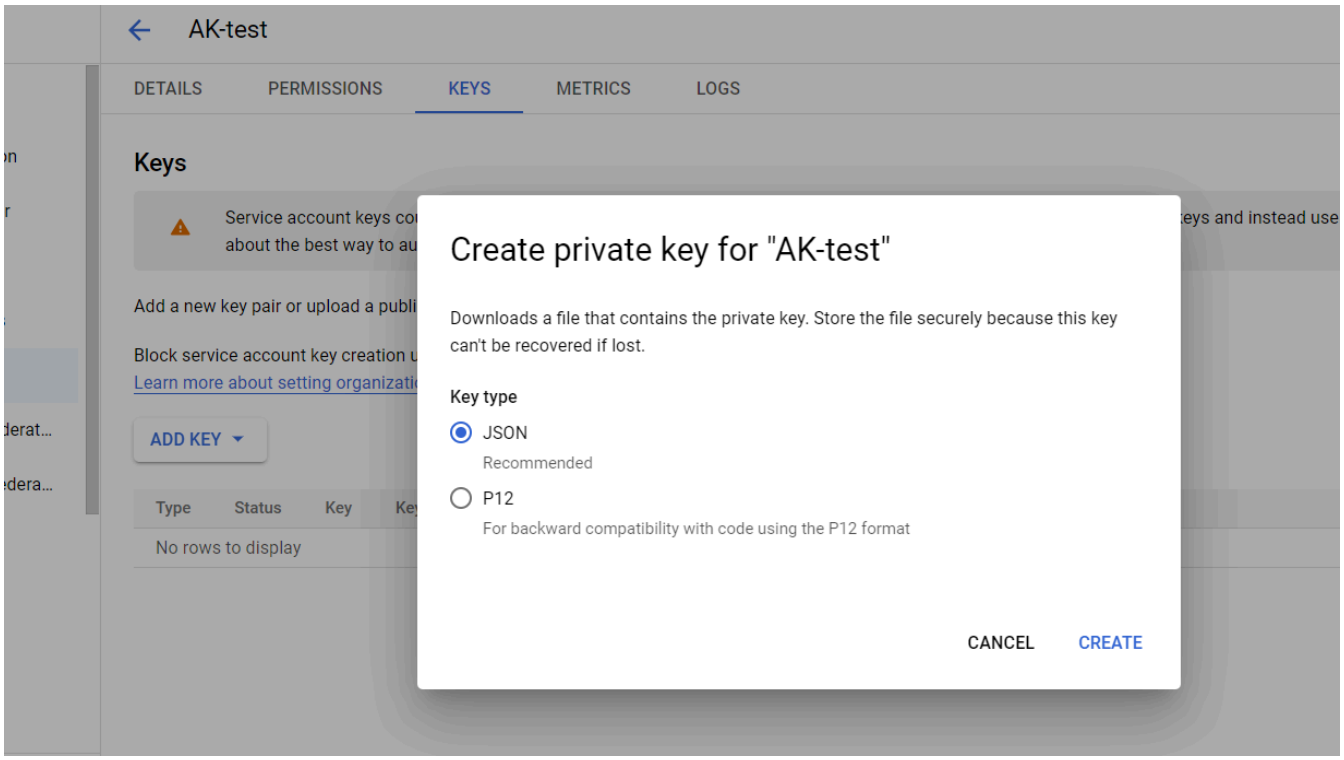
Step 11: Click on “Continue“ and ”Done”



Step 12: Go to the created Service Account, click on that Service Account navigate to the “Keys“ section.



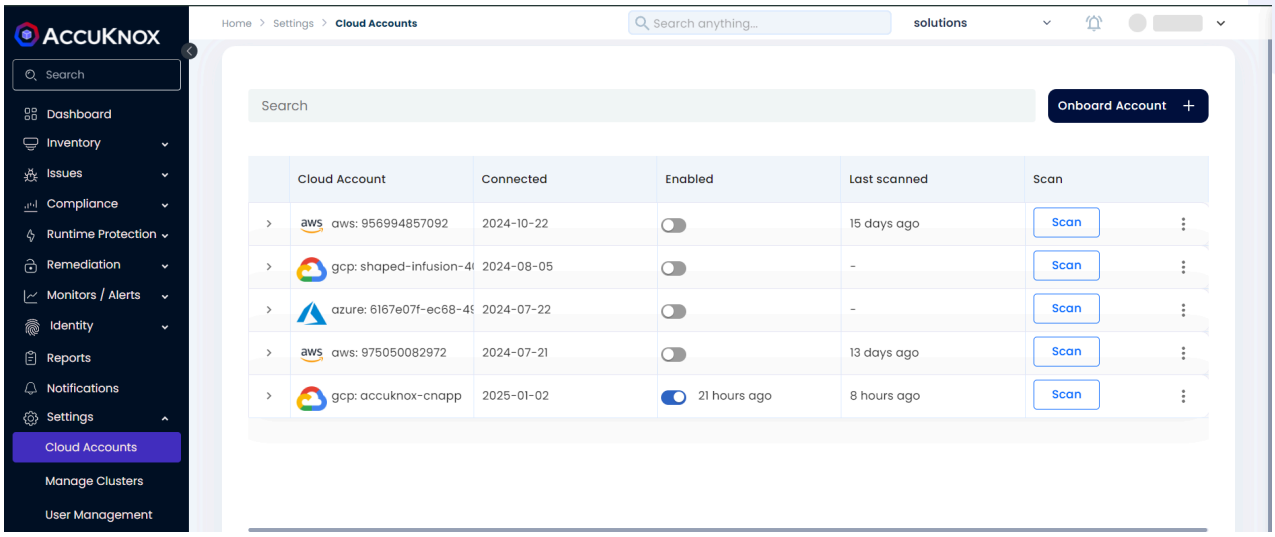
Step 13: Click the “Add key“ button and “Create new key “ . Chosen Key type should be JSON format.



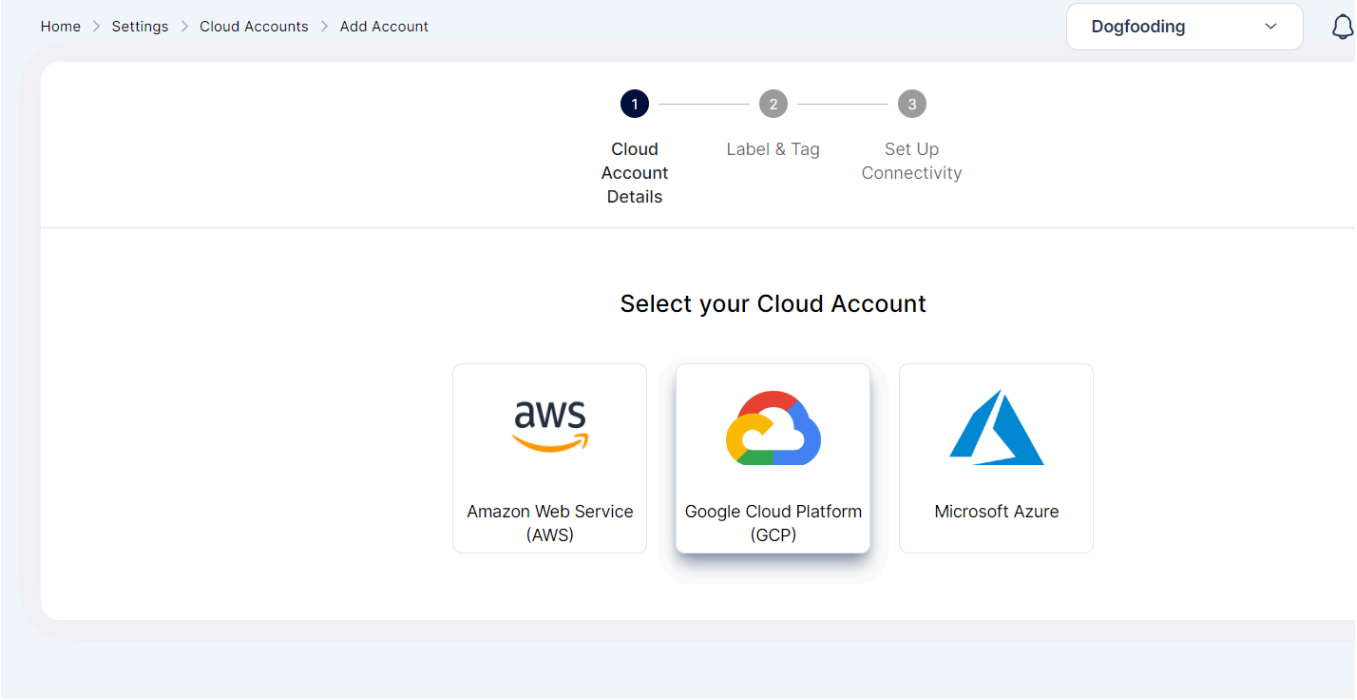
Step 14: Click the “Create“ button it will automatically download the JSON key.

From AccuKnox SaaS UI

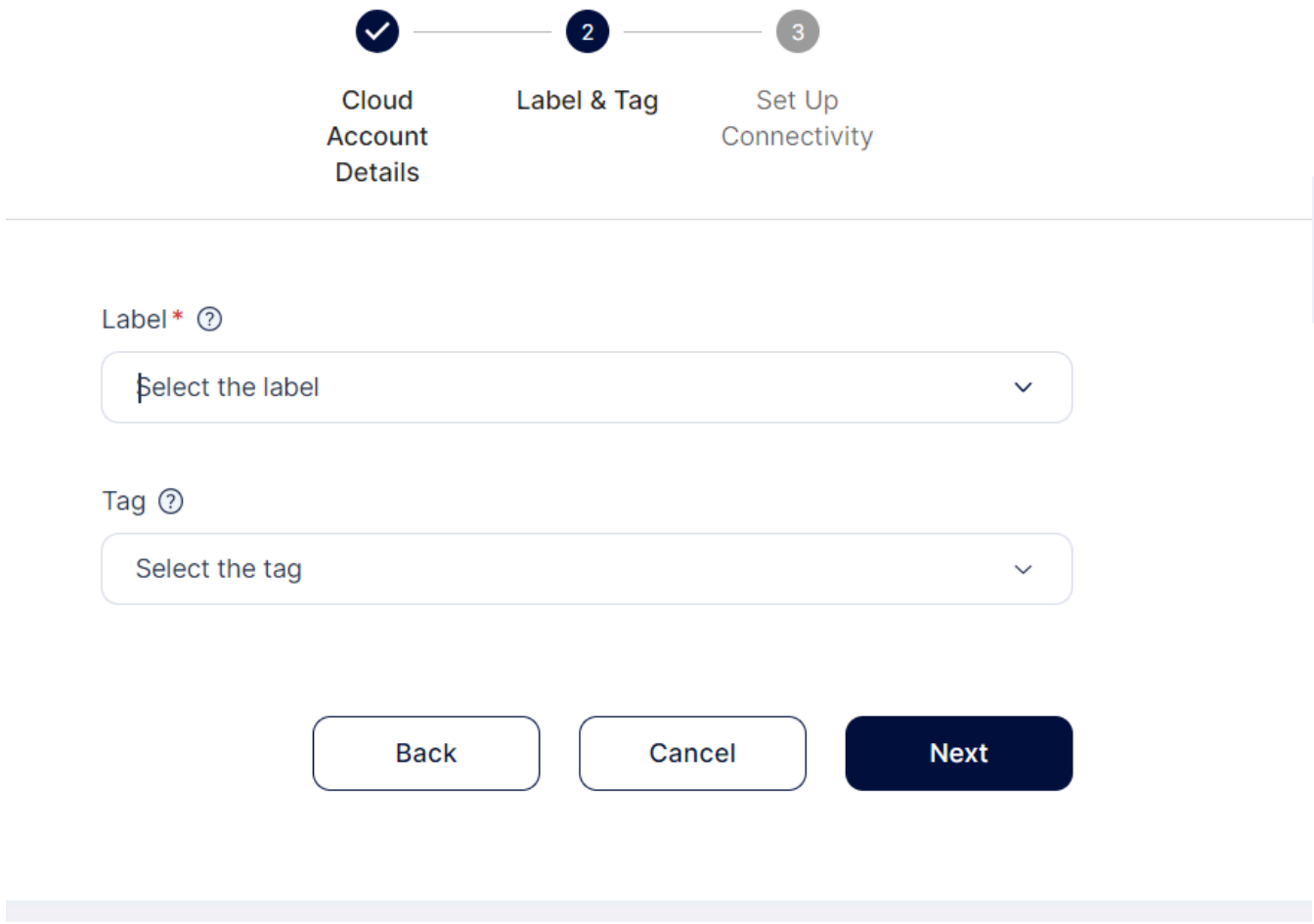
Step 1: Go to the AccuKnox SaaS. Navigate to the “Settings” → “Cloud Accounts” then “Add Account”.



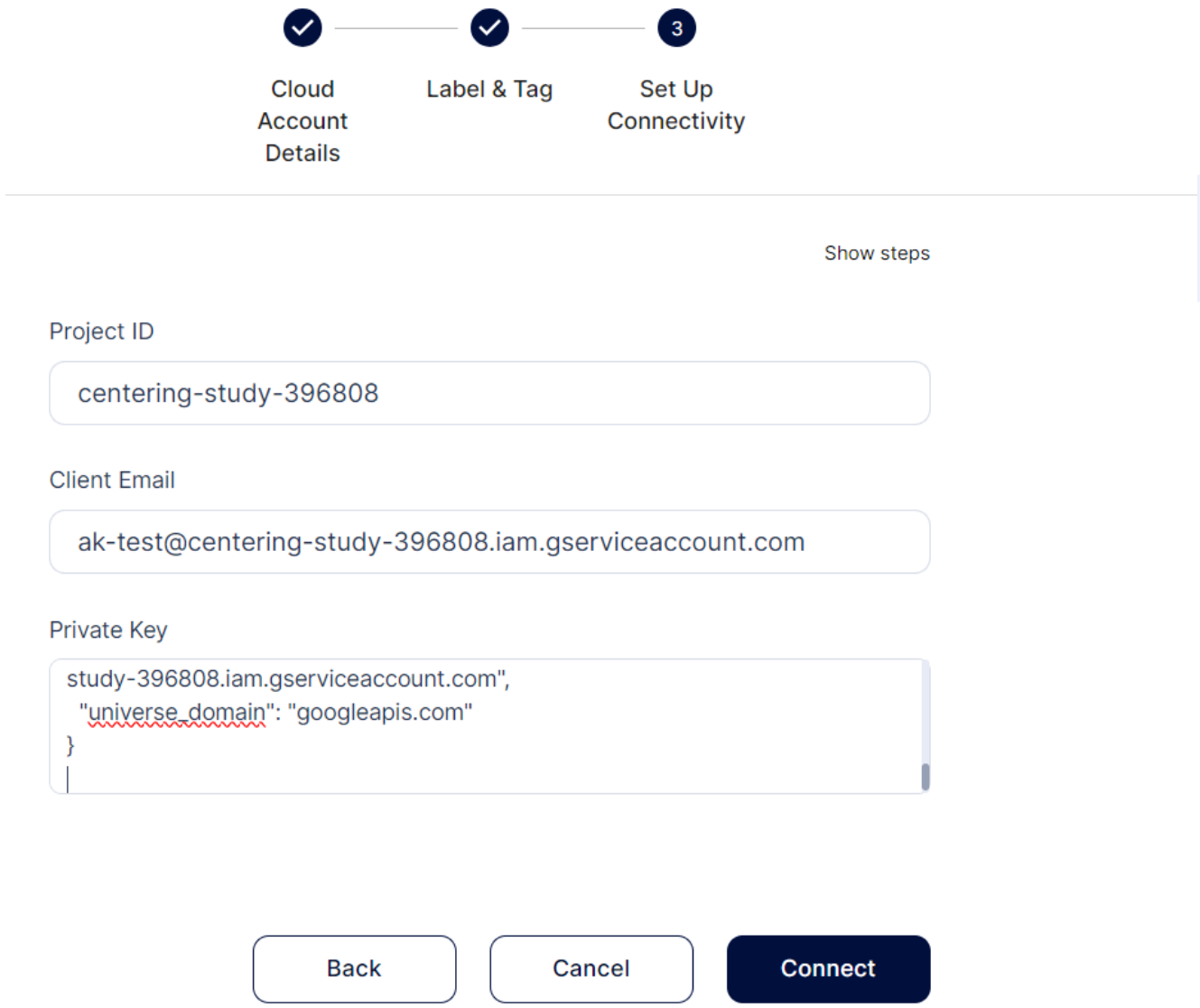
Step 2: Click the “GCP Platform”



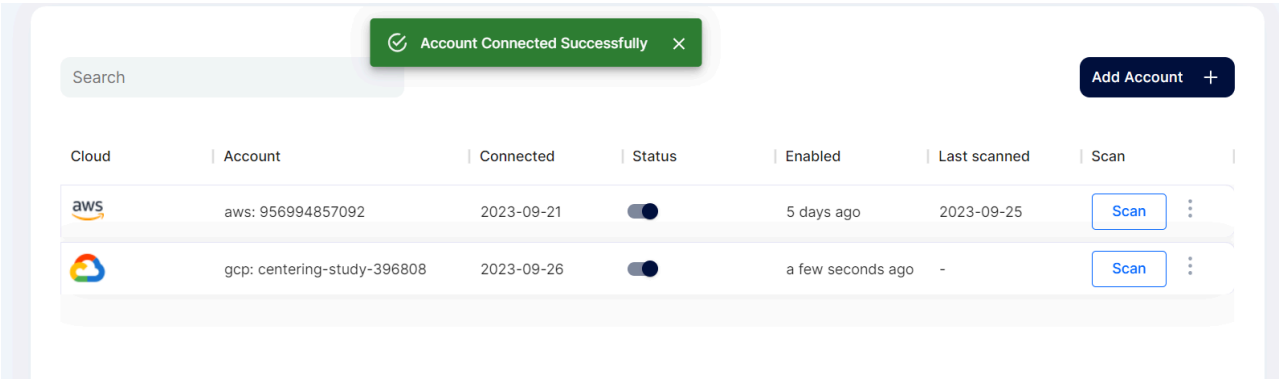
Step 3: Create New Label and Add the Label for identifying the assets inside this account and add a Tag optionally.



Step 4: Enter the “Project ID“, “Client Email”(The Service Account mail ID) and “Private Key” from the downloaded File. Copy paste the entire downloaded file into the ”Private Key” field . Then Click “Connect“



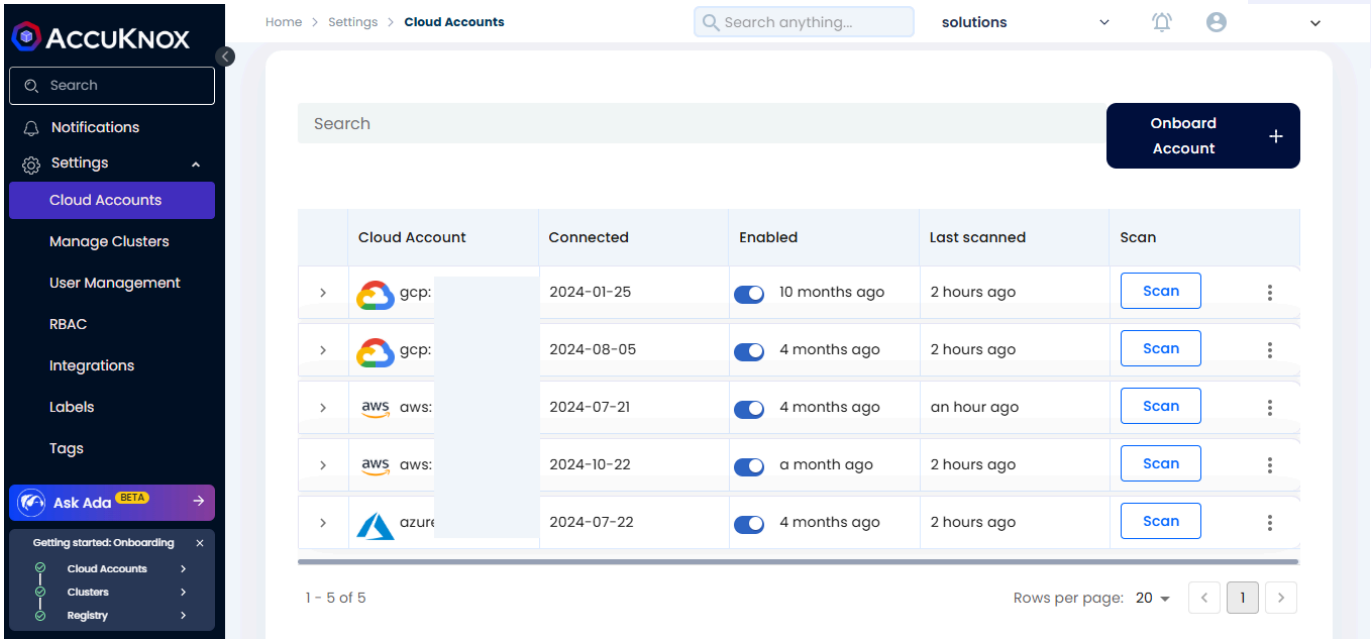
The cloud account has been onboarded successfully



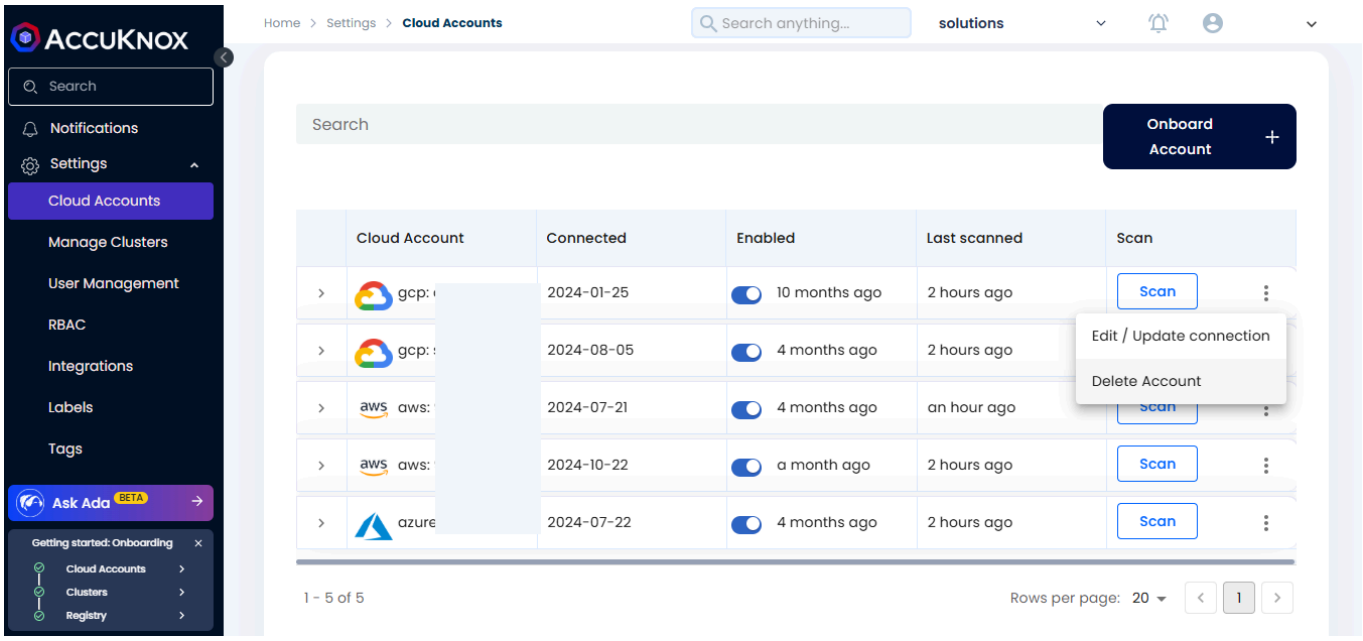
How to Deboard a Cloud Account

This guide outlines the steps for offboarding a cloud account from AccuKnox SaaS.

Step 1: Login to AccuKnox SaaS and Go to Cloud Accounts under Settings.



Step 2: Select the cloud account and click “Delete” to delete the account from SaaS.



This will delete the cloud account from AccuKnox SaaS.

Kubernetes Security Onboarding

Features Supported for Kubernetes

- Supported on managed (EKS, AKS, OCI) and on-prem Kubernetes clusters
- Works on Kubernetes versions >= 1.18
- All features are modular and can be enabled independently
- Available via AccuKnox SaaS and On-Prem Control Plane with identical UX
- Runtime Security requires Linux kernel >= 4.15
- Only egress connectivity from K8s cluster to control plane is required

K8s Runtime Visibility and Security

Deployment Mode: DaemonSet via Operator (default) or Kubernetes manifests

Helm Command:

```
helm upgrade --install agents oci://public.ecr.aws/k9v9d5v2/agents-chart \
--version "v0.10.0" \
--set joinToken="[TOKEN]" \
--set spireHost="spire.demo.accuknox.com" \
--set ppsHost="pps.demo.accuknox.com" \
--set knoxGateway="knox-gw.demo.accuknox.com:3000" \
--set admissionController.enabled=false \
--set kyverno.enabled=false \
-n agents --create-namespace
```

Features:

- File, process, and network visibility
- MITRE-based policy enforcement (FIM, cryptojacking protection, etc.)
- Auto-discovery of ingress/egress and whitelisting policies

Control Plane Access:

- PPS: Port 443
- SPIRE: Port 443
- Knox Gateway: Port 3000

K8s Misconfiguration Scanning

Deployment Mode: Kubernetes cronjob

Helm Command:

```
helm upgrade --install k8s-risk-assessment-job
oci://public.ecr.aws/k9v9d5v2/k8s-risk-assessment-job \
--set accuknox.tenantID="[TENANTID]" \
--set accuknox.authToken="[AUTHTOKEN]" \
--set accuknox.cronTab="30 9 * * *" \
--set accuknox.clusterName="[CLUSTERNAME]" \
--set accuknox.URL="cspm.demo.accuknox.com" \
--set accuknox.label="[LABEL]" \
--version=v1.1.3
```

Features:

- Detection of misconfigurations and insecure configurations
- Includes checks for root containers, privilege escalation, and 100+ other rules

Control Plane Access:

- HTTPS access to Artifact Endpoint

K8s Identity & Entitlements Management

Deployment Mode: Kubernetes cronjob

Helm Command:

```
helm upgrade --install kiem-job oci://public.ecr.aws/k9v9d5v2/kiem-job \
--set accuknox.label="[LABEL]" \
--version v1.1.3 \
--set accuknox.URL="cspm.demo.accuknox.com" \
--set accuknox.authToken="[AUTHTOKEN]" \
--set accuknox.cronTab="30 9 * * *" \
--set accuknox.clusterName="[CLUSTERNAME]" \
--set accuknox.tenantID="[TENANTID]"
```

Features:

- Identifies overly permissive role bindings
- Graph-based identity view
- Detection of dangling service accounts and cross-namespace access

Control Plane Access:

- HTTPS access to Artifact Endpoint

K8s CIS Benchmarking

Deployment Mode: Kubernetes cronjob

Helm Command:

```
helm upgrade --install cis-k8s-job oci://public.ecr.aws/k9v9d5v2/cis-k8s-job \
--set accuknox.url="cspm.demo.accuknox.com" \
--set accuknox.tenantId="[TENANTID]" \
--set accuknox.authToken="[AUTHTOKEN]" \
--set accuknox.cronTab="30 9 * * *" \
--set accuknox.clusterName="[CLUSTERNAME]" \
```

```
--set accuknox.label="[LABEL]" \
--version v1.1.3
```

Features:

- Benchmarks support for:
- Kubernetes (generic)
- EKS
- AKS
- GKE
- OKE not currently supported

Control Plane Access:

- HTTPS access to Artifact Endpoint

DISA STIGs Support

Deployment Mode: Kubernetes cronjob

Helm Command:

```
helm upgrade --install k8s-stig-job oci://public.ecr.aws/k9v9d5v2/k8s-stig-job \
--set accuknox.url="cspm.demo.accuknox.com" \
--set accuknox.tenantId="[TENANTID]" \
--set accuknox.authToken="[AUTHTOKEN]" \
--set accuknox.cronTab="30 9 * * *" \
--set accuknox.clusterName="[CLUSTERNAME]" \
--set accuknox.label="[LABEL]" \
--version v1.1.3
```

Features:

- DISA Special Technical Implementation Guidelines (STIGs) compliance

Control Plane Access:

- HTTPS access to Artifact Endpoint

In-Cluster Container Image Scanning

Deployment Mode: CronJob (per node job)

Helm Command:

```
helm install kubeshield kubeshield-chart \
--set scan.tenantId="<TENANTID>" \
--set scan.artifactToken="<TOKEN>" \
--set scan.artifactEndpoint="https://cspm.demo.accuknox.com/api/v1/artifact/" \
--set scan.label="<LABEL>"
```

Features:

- Direct in-cluster image scanning (no registry access required)
- Scans cached images on nodes
- Reports sent to AccuKnox console for triage

Control Plane Access:

- HTTPS access to Artifact Endpoint

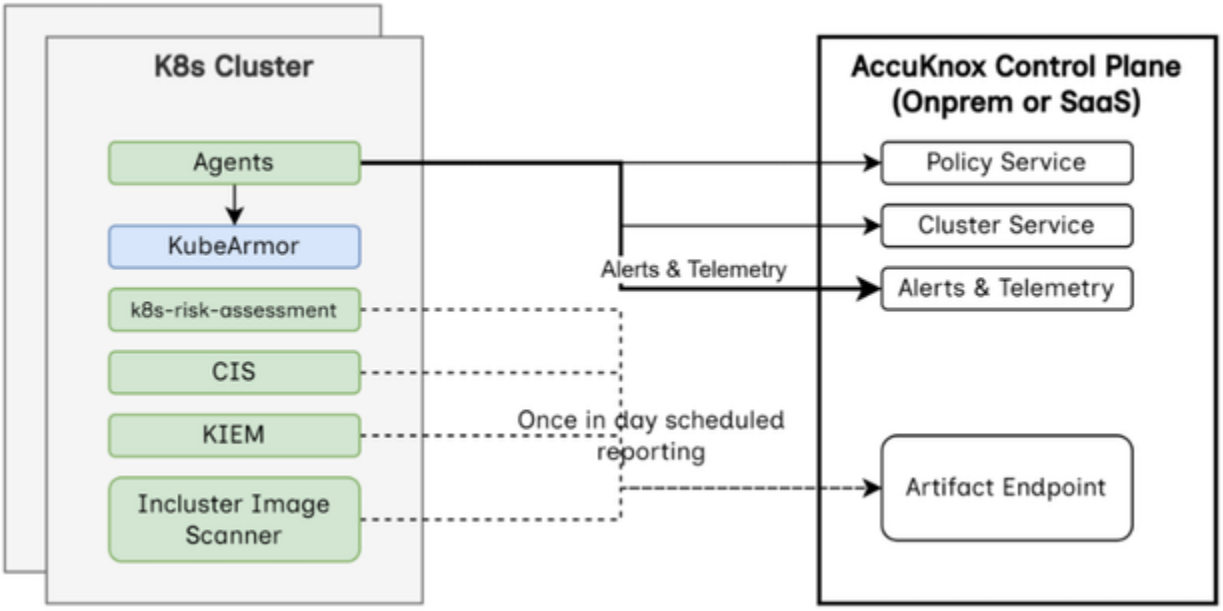
Admission Controller Support

AccuKnox Admission Controller enforces:

1. Trusted registry enforcement for images
2. Deployment compliance with security best practices (no root, no host mounts, etc.)
3. Violations reported to AccuKnox Control Plane (visible under Monitors & Alerts)

Cluster Access to Control Plane

Each feature requires outbound (egress) HTTPS access only. Refer to the access notes under each feature for exact service and port requirements.



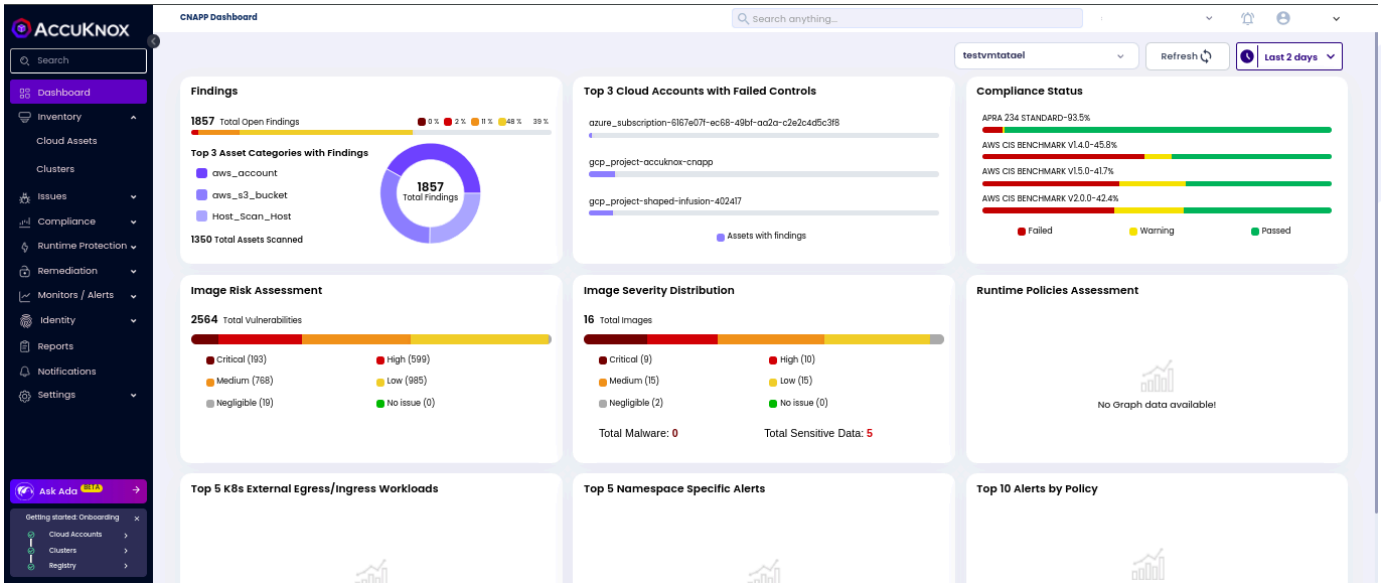
Cluster Onboarding

This is a detailed guide on how to onboard clusters to the AccuKnox SaaS platform. The guide covers the installation of KubeArmor and AccuKnox agents in the cluster to connect to the AccuKnox SaaS application.

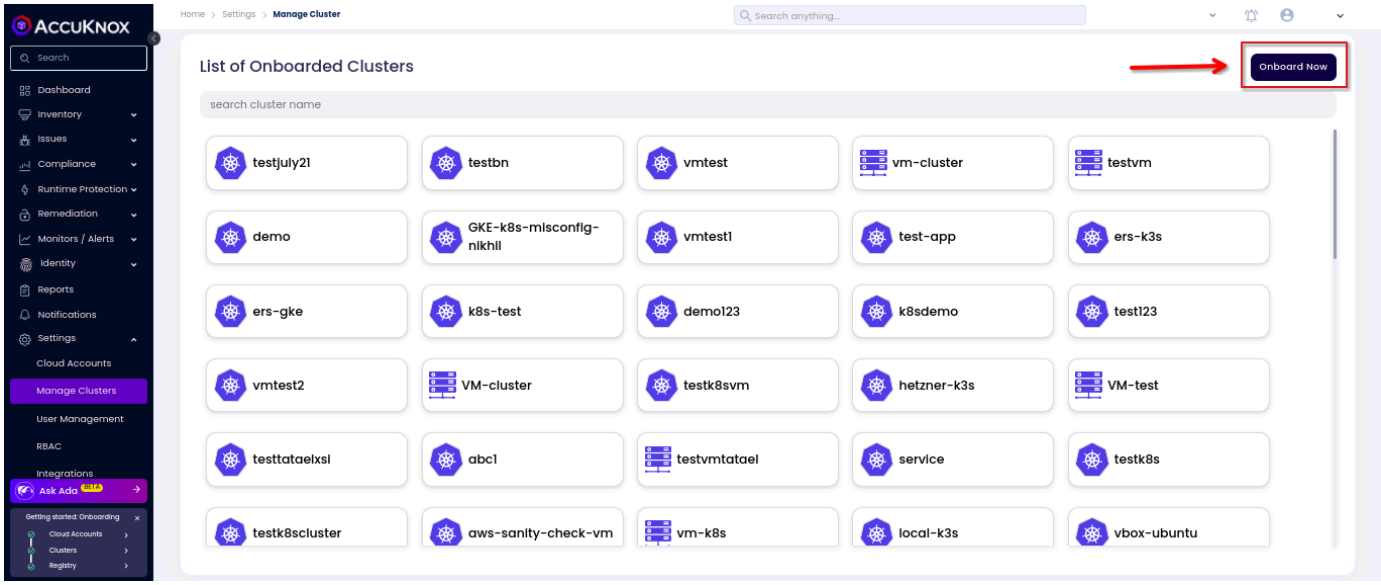
Below shown image is from an k3s cluster running in a local machine with Kali Linux Operating System. We can onboard this cluster by following the steps shown below

```
└─$ kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
nginx-demo   1/1     Running   0          22s
redis-demo   1/1     Running   0          14s
```

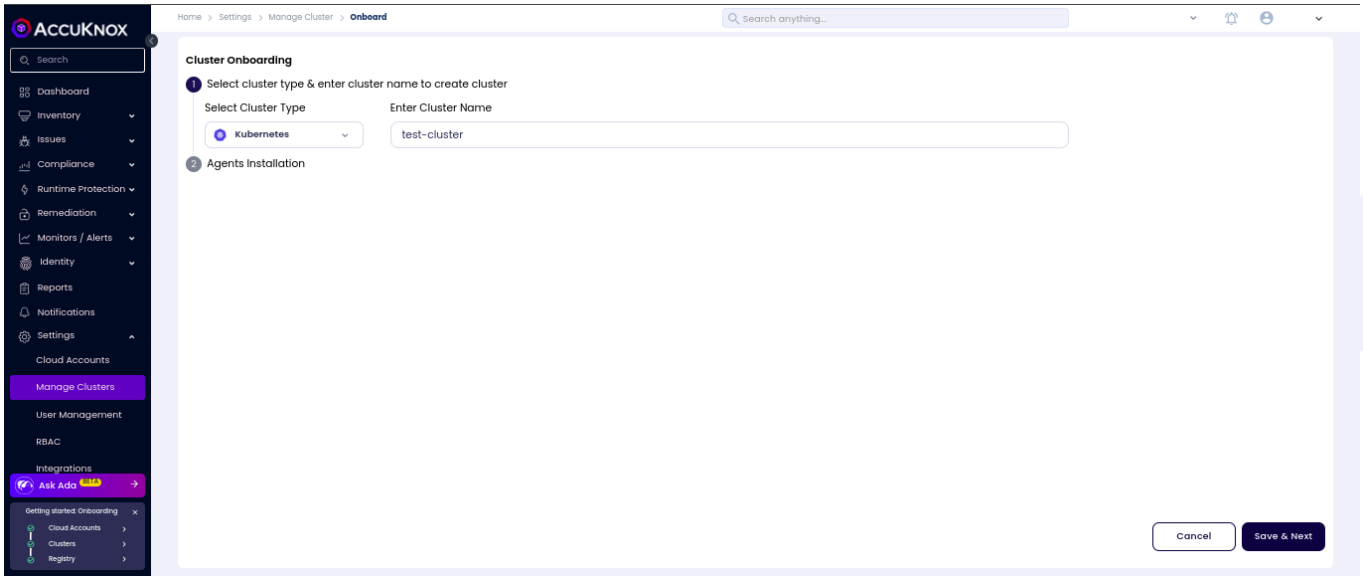
Step 1: As a first time user, the management console will show up the CNAPP dashboard without any data mentioned in widgets, since the cloud account and cluster onboarding is not done.



Step 2: Navigate to Manage Cluster from Settings Tab: From this page we can onboard the clusters running in various cloud platforms like GCP,AWS and Azure. We can onboard locally setup clusters using an cloud option. To onboard cluster select onboard now option



Step 3: In this screen, give any name to the cluster that you are going to onboard now.



Step 4: Installing KubeArmor and AccuKnox agents

We are going to install KubeArmor and AccuKnox-agents to connect to the AccuKnox SaaS application. For the agent installation selection click on the Runtime Visibility & Protection.

Step 4.1 KubeArmor Installation

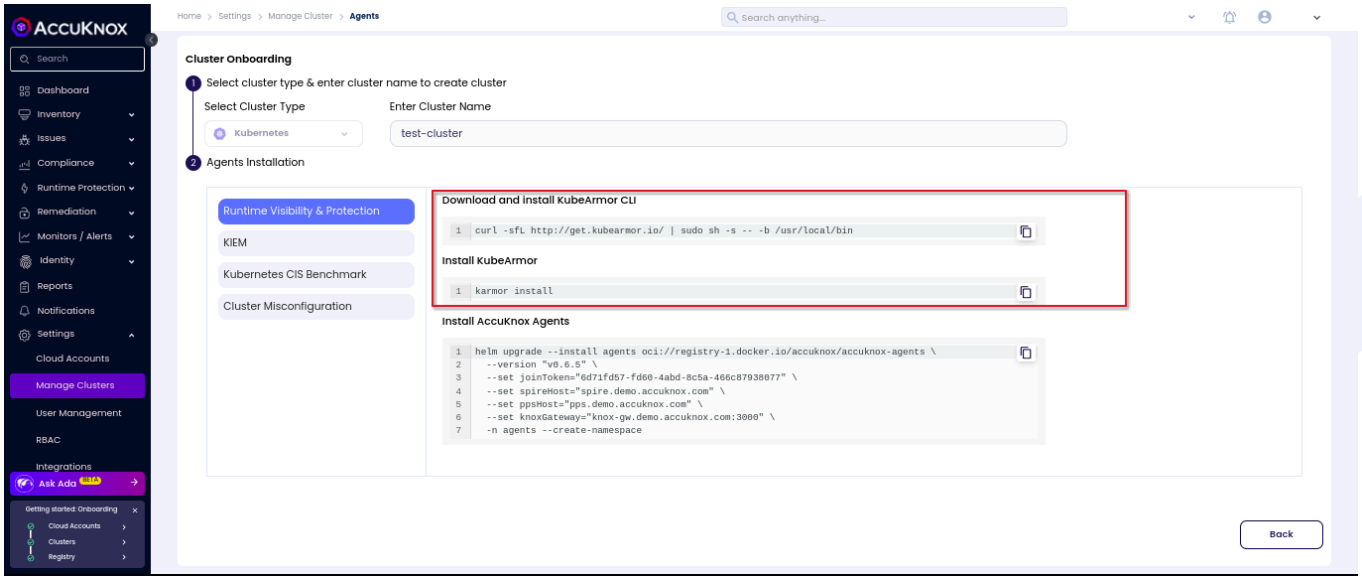
KubeArmor

KubeArmor is a cloud-native runtime security enforcement system that restricts the behavior (such as process execution, file access, and networking operation) of containers and nodes at the system level.

With KubeArmor, a user can:

- Restrict file system access for certain processes
- Restrict what processes can be spawned within the pod
- Restrict the capabilities that can be used by the processes within the pod

KubeArmor differs from seccomp-based profiles, wherein KubeArmor allows to dynamically set the restrictions on the pod. With seccomp, the restrictions must be placed during the pod startup and cannot be changed later. KubeArmor leverages Linux Security Modules (LSMs) to enforce policies at runtime.



KubeArmor is installed using the following commands:

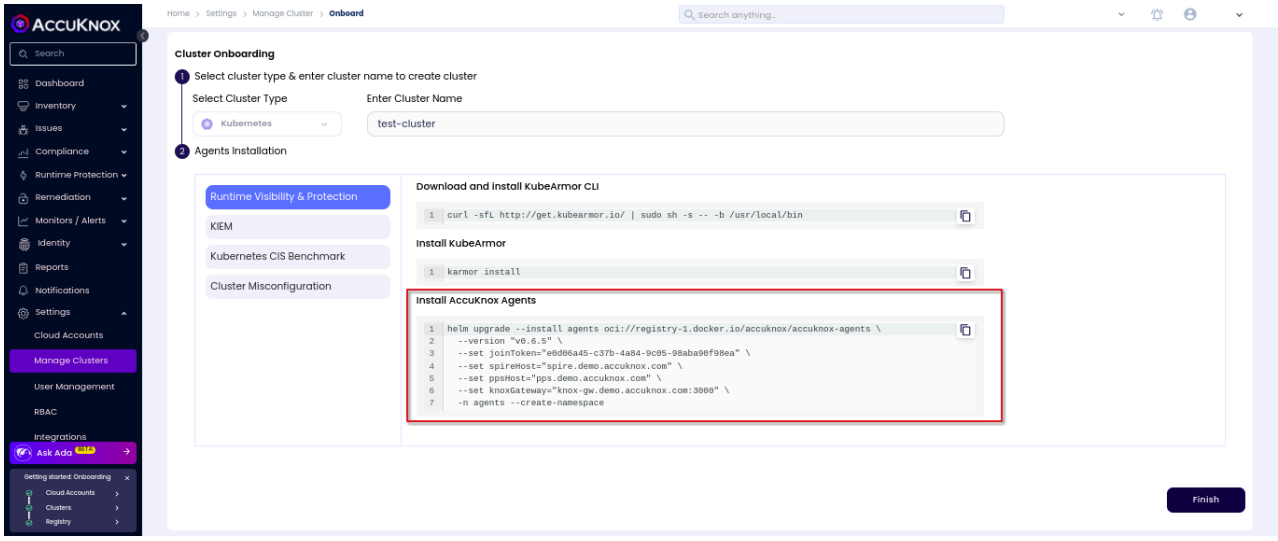
```
curl -sfL http://get.kubearmor.io/ | sudo sh -s -- -b /usr/local/bin && karmor install
```

Step 4.2: AccuKnox-Agents installation

After installing KubeArmor we are going to install AccuKnox Agents in the cluster.

AccuKnox Agents

1. **KubeArmor**: KubeArmor is a cloud-native runtime security enforcement system that restricts the behavior (such as process execution, file access, and networking operation) of containers and nodes at the system level. KubeArmor dynamically set the restrictions on the pod. KubeArmor leverages Linux Security Modules (LSMs) to enforce policies at runtime.
2. **Feeder Service:** It collects the feeds from kubeArmor and relays to the app.
3. **Shared Informer Agent:** It collects information about the cluster like pods, nodes, namespaces etc.,
4. **Policy Discovery Engine:** It discovers the policies using the workload and cluster information that is relayed by a shared informer Agent.



AccuKnox Agents can be installed using the following command:

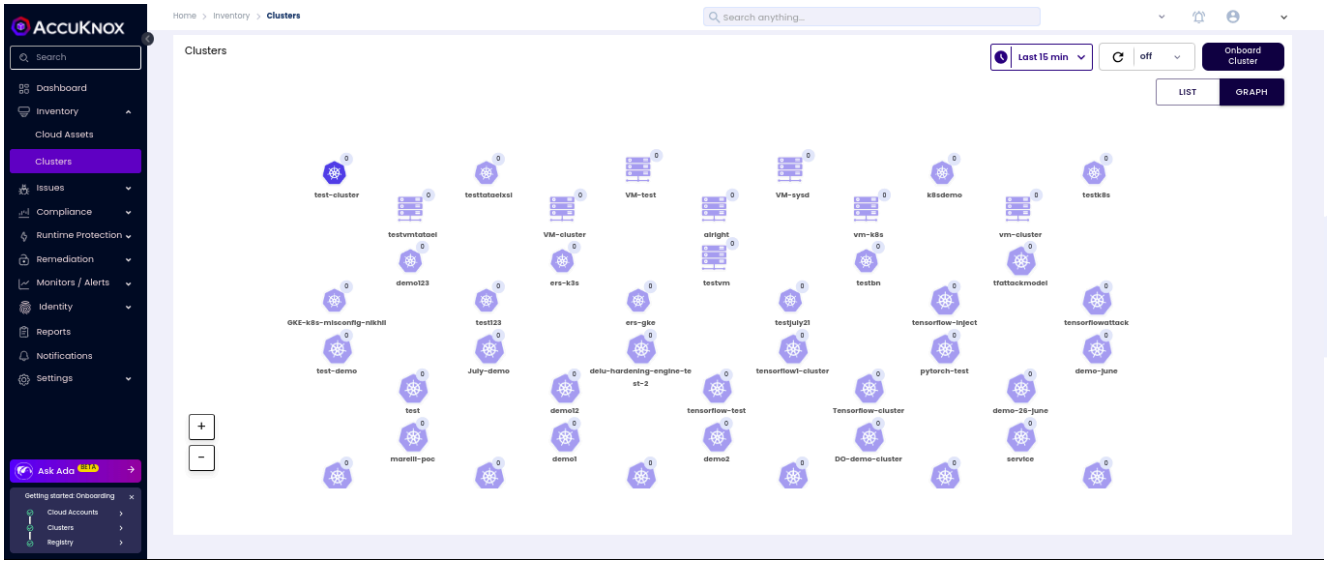
```
helm upgrade --install agents oci://registry-1.docker.io/accuknox/accuknox-agents
--version "v0.6.5"
--set joinToken="**********_**********_**********"
--set spireHost="spire.demo.accuknox.com"
--set ppsHost="pps.demo.accuknox.com"
--set knoxGateway="knox-gw.demo.accuknox.com:3000"
-n agents --create-namespace
```

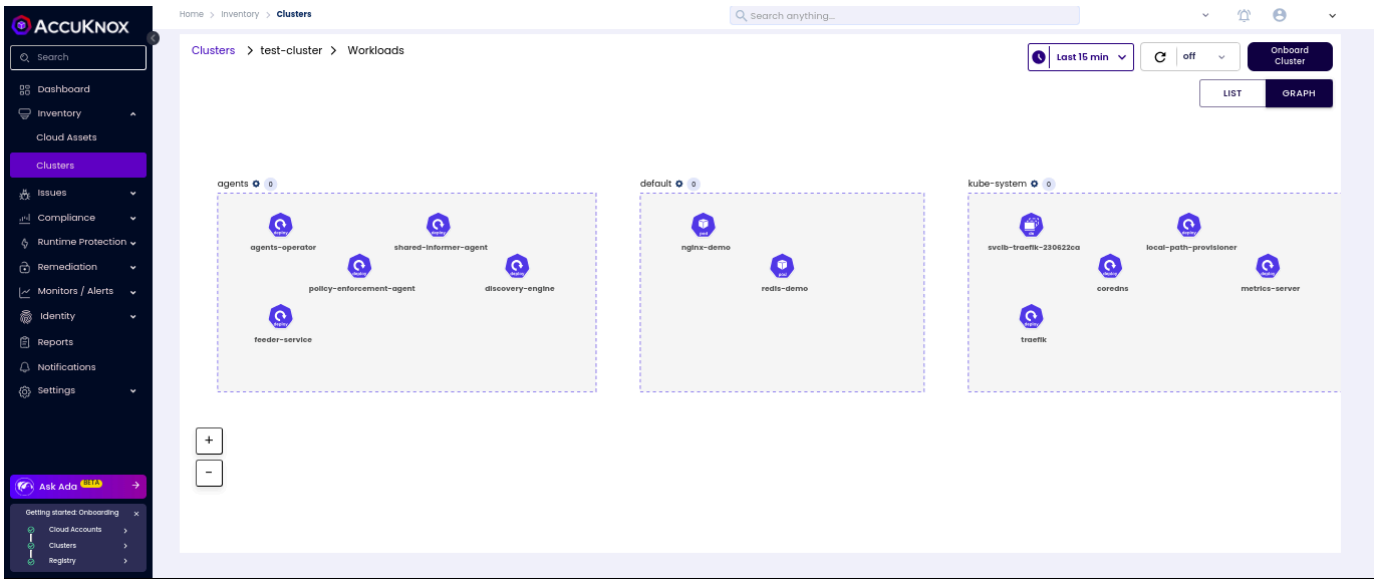
Note

In the above command joinToken is specific to this example and it will vary based on the cluster

Step 5: Onboarded Cluster

After installing all the AccuKnox agents the cluster is onboarded successfully into the SaaS application. We can see the workload details of the onboarded cluster by Navigating to Inventory→cloud Workloads option. There all the onboarded clusters will be listed out and all the inactive ones would be grayed out. By Double clicking on the active cluster user can get a more detailed view of the cluster.





Cluster Onboarding with Access Keys

Streamlining cluster onboarding is made easy with access keys, allowing users to onboard multiple clusters using the same key. Additionally, users can set expiration times for these keys and specify the number of clusters each key can onboard. This process can be performed directly from the CLI if the access key is already created, offering enhanced flexibility and convenience

Pre-requisite:

1. Kubernetes (managed/un-manager) cluster

2. AccuKnox CNAPP login access
3. One or more clusters to onboard
4. Access Key (See how to create)

Onboarding

In the case of the Access key onboarding method, the User can directly onboard the VMs from the CLI, To Onboard a new cluster follow the below steps:

Step1: Install KubeArmor

```
curl -sfL http://get.kubearmor.io/ | sudo sh -s -- -b /usr/local/bin
karmor install
```

Output:

```
kubearmor/kubearmor-client info checking GitHub for latest tag
kubearmor/kubearmor-client info found version: 1.3.0 for v1.3.0/linux/amd64
kubearmor/kubearmor-client info installed /usr/local/bin/karmor
kubearmor/kubearmor-client info karmor is installed in /usr/local/bin
kubearmor/kubearmor-client info invoke /usr/local/bin/karmor or move karmor to your
desired PATH

$ karmor install
🛡️    Installed helm release : kubearmor-operator
😄    KubeArmorConfig created
⌚    This may take a couple of minutes
🥳    KubeArmor Snitch Deployed!
🥳    KubeArmor Daemonset Deployed!
🥳    Done Checking , ALL Services are running!
⌚    Execution Time : 58.615464051s

🔧    Verifying KubeArmor functionality (this may take upto a minute)...

    🛡️    Your Cluster is Armored Up!
```

Step2: Install AccuKnox Agents

AccuKnox-Agents:

The AccuKnox Agent is a K8s operator that installs the following agents:

- Feeder service: It collects KubeArmor feeds.
- Shared-informer-agent: This agent authenticates with your cluster and collects information regarding entities like nodes, pods, and namespaces.
- Policy-enforcement-agent: This agent authenticates with your cluster and enforces labels and policies.
- Discovery Engine: Discovery Engine discovers the security posture for your workloads and auto-discovers the policy set required to put the workload in least-permissive mode. The engine leverages the rich visibility provided by KubeArmor to auto-discover systems and network security postures.

The agent-operator also manages the agents' resource limits. The operator is in charge of spawning the agents based on the size of the cluster. If the cluster size changes, i.e., new nodes are added or existing nodes are deleted, then the operator scales up or down the resources accordingly.

AccuKnox Agents can be installed using the following command:

```
helm upgrade --install agents oci://registry-1.docker.io/accuknox/accuknox-agents \
    --version "v0.5.11" \
    --set spireHost="spire.demo.accuknox.com" \
    --set ppsHost="pps.demo.accuknox.com" \
    --set knoxGateway="knox-gw.demo.accuknox.com:3000" \
    --set tokenURL="cwpp.demo.accuknox.com" \
    --set clusterName="accuknoxcluster" \
    --set accessKey="<token>" \
    -n accuknox-agents --create-namespace
```

Note

In the commands above, substitute **--set clusterName** with the desired cluster name, and replace the `<token>` with the **Access Keys** generated from UI. Adjust the URLs if required

Note

Please check for the value of --version "v0.0.0" from the UI steps of cluster onboarding to make sure you are using the latest image tags

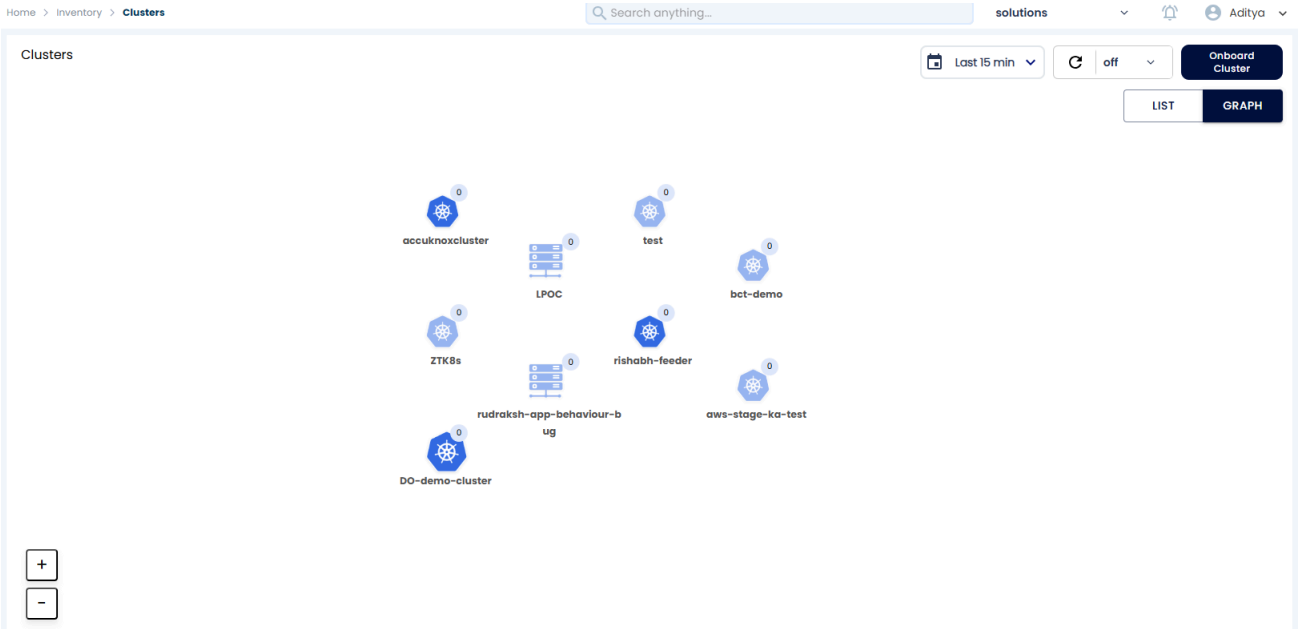
Output

```
Release "agents" does not exist. Installing it now.
Pulled: registry-1.docker.io/accuknox/accuknox-agents:v0.5.11
Digest: sha256:6b7870020c0470741b7a89f47fd6f4e85882521721ce50407351d231508c6aaf
NAME: agents
LAST DEPLOYED: Thu Jan  2 19:05:38 2025
NAMESPACE: accuknox-agents
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

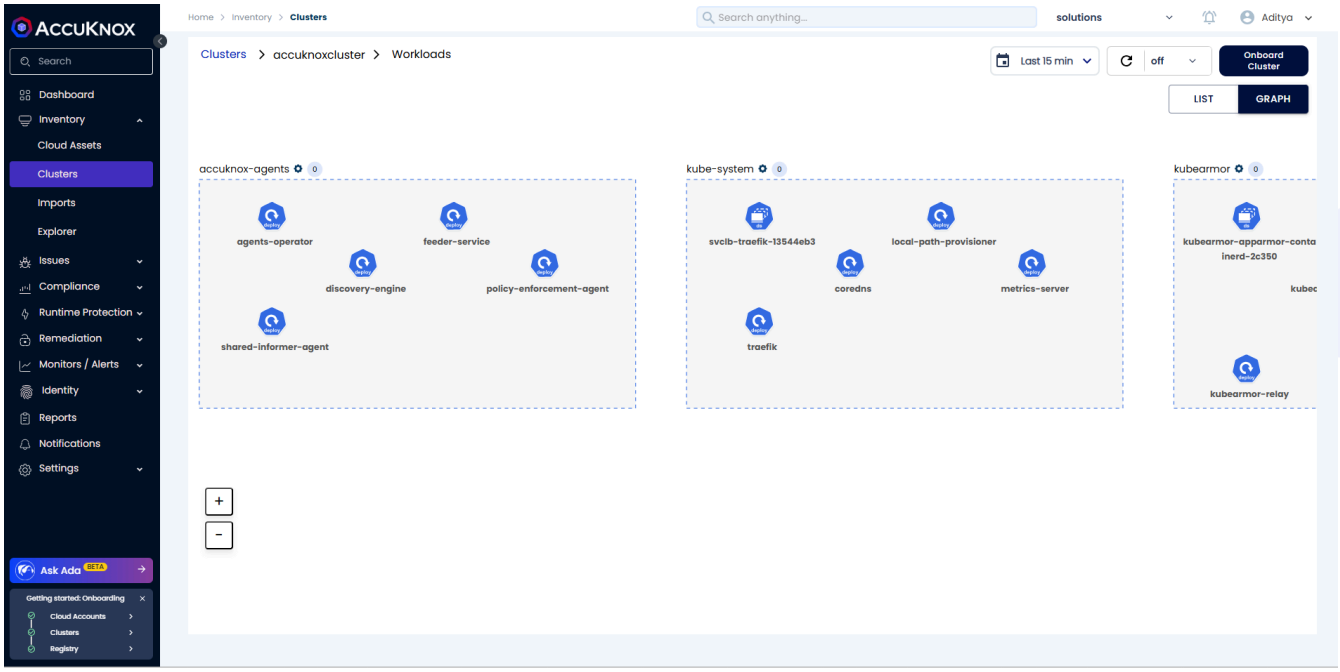
To verify please use

```
kubectl get po -n accuknox-agents
```

After installing all the AccuKnox agents, the cluster is onboarded successfully into the SaaS application. We can see the workload details of the onboarded cluster by Navigating to Inventory-> Clusters



View the workloads



Note

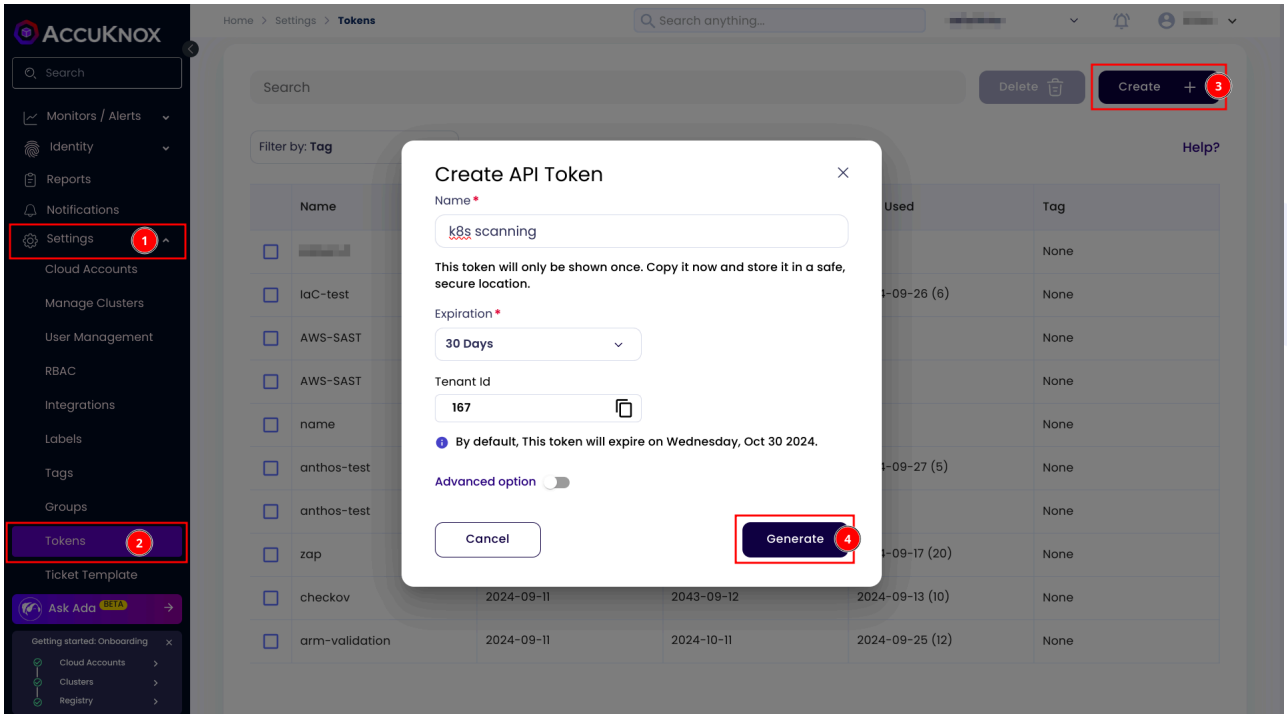
You can repeat the same command with different **"clusterName"** to onboard multiple cluster using access keys

Onboard Cluster for Misconfiguration Scanning

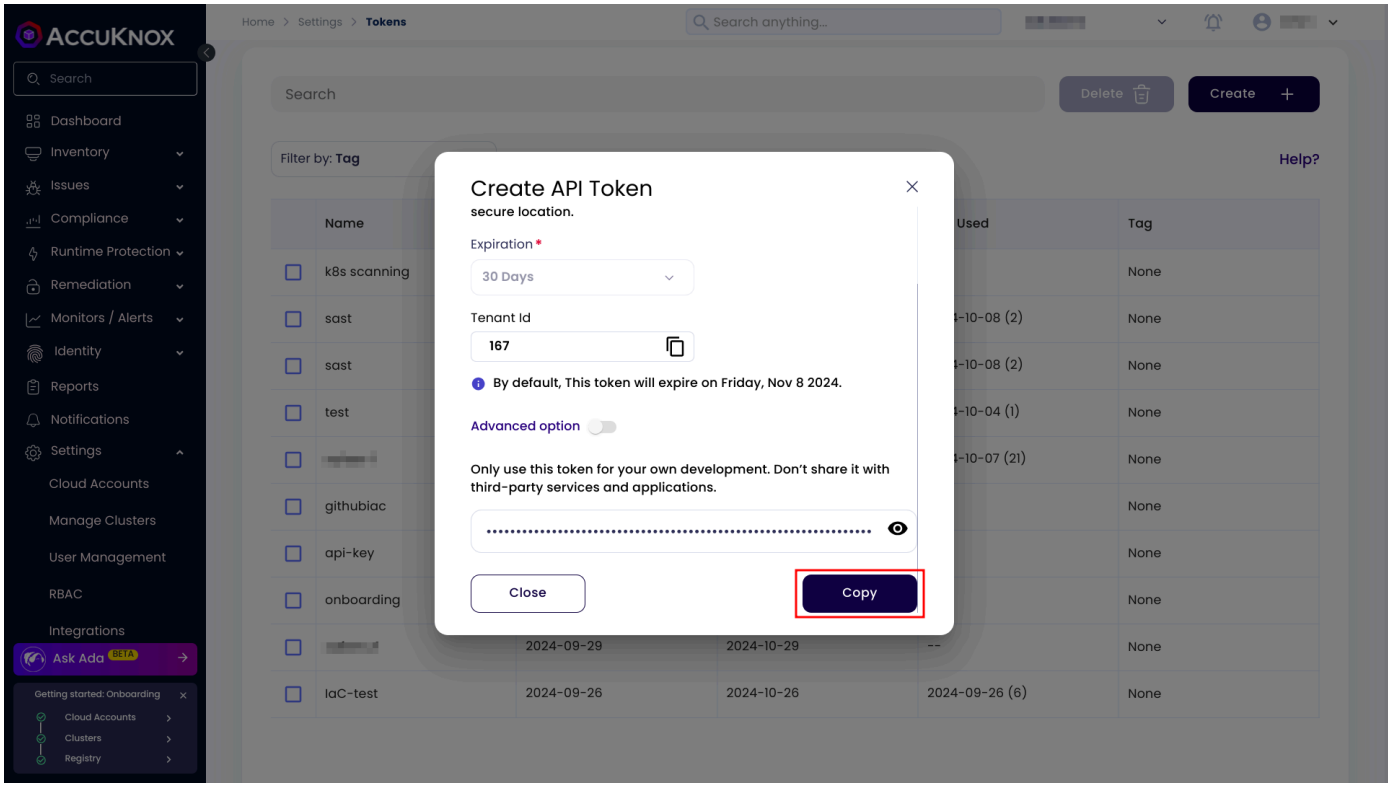
This guide outlines the steps for onboarding a cluster to AccuKnox SaaS for scanning cluster misconfigurations.

For onboarding a cluster and for scanning for misconfigurations you need to create a token first. For creating follow these steps:

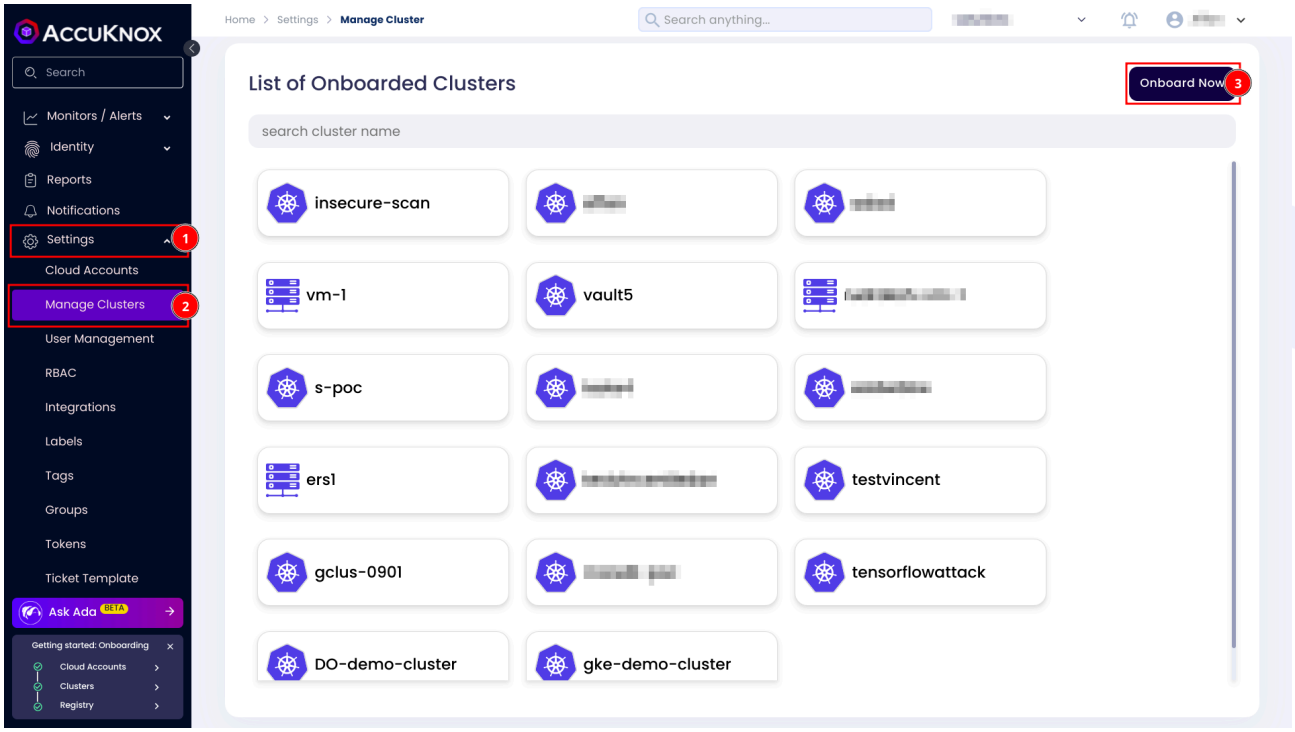
Go to `Settings > Tokens` and click on the create button. Give your token a name and click on generate button.



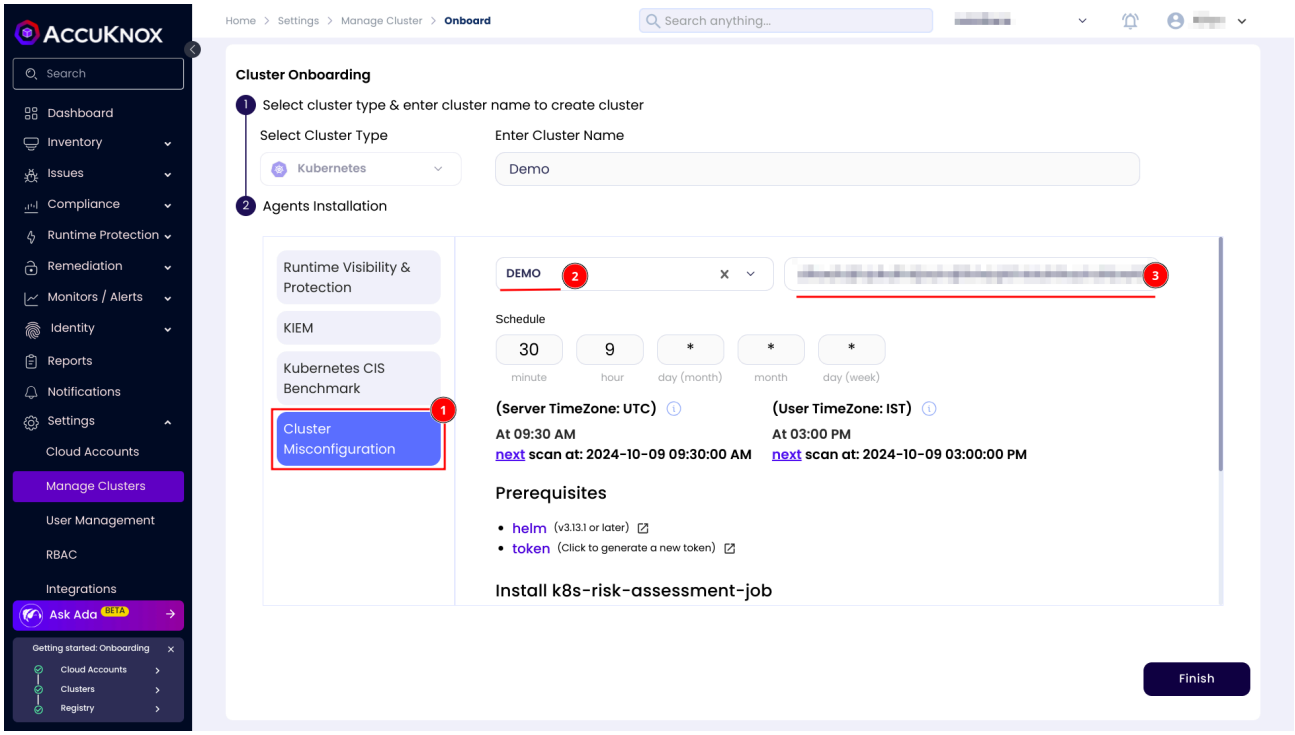
Once the token is generated, copy it and take a note of it.



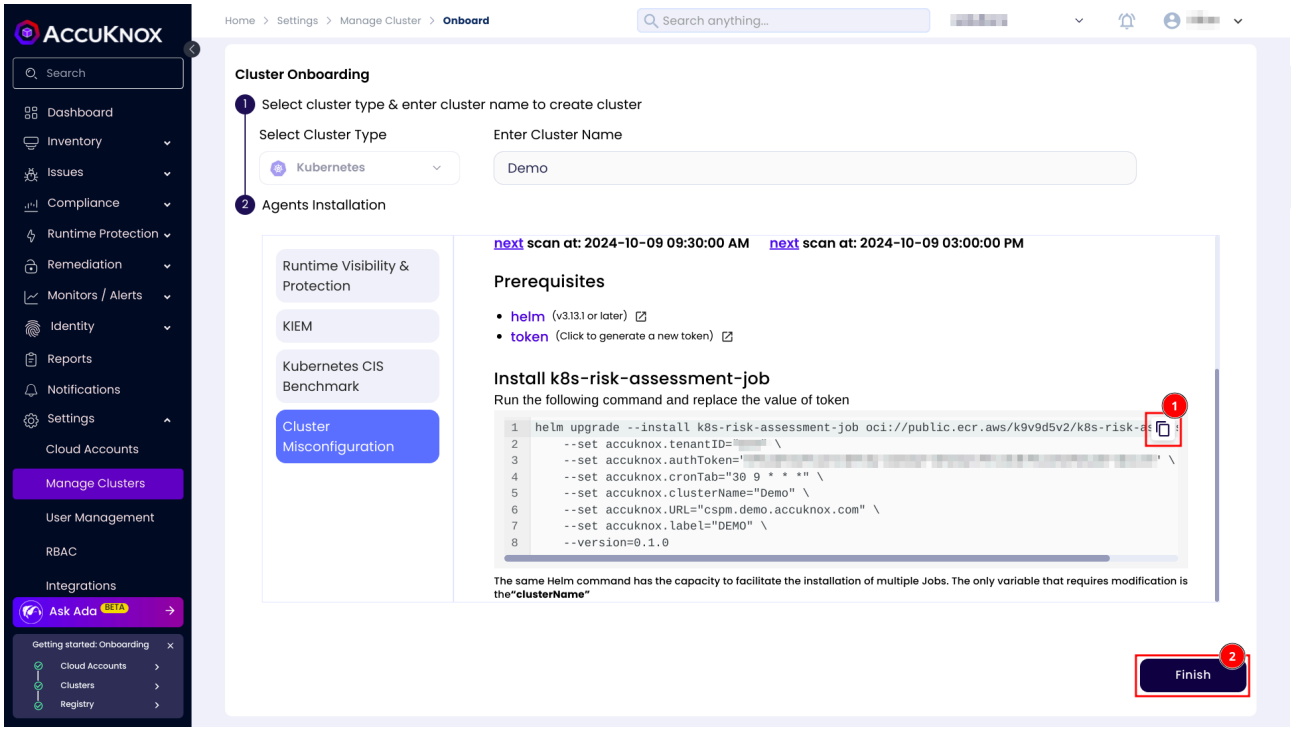
Now go to Settings > Manage Clusters, click on onboard now button or select an existing cluster.



Give your cluster a name. Under the Agents Installation section select Cluster Misconfiguration. Select a label and paste your token.

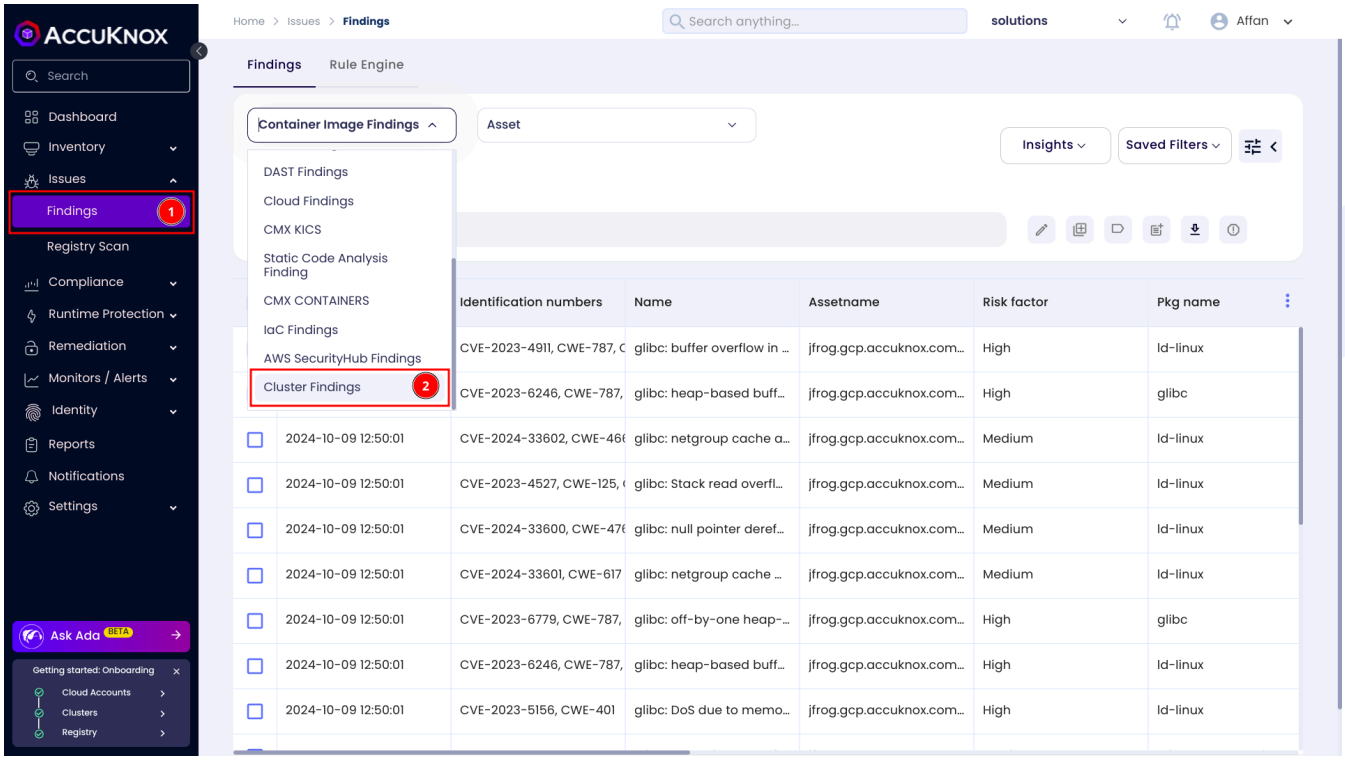


You can also change the schedule as per your requirement. Then next scan will happen based on the schedule. Scroll down and copy the helm command and run it inside a terminal. Then click on Finish button.

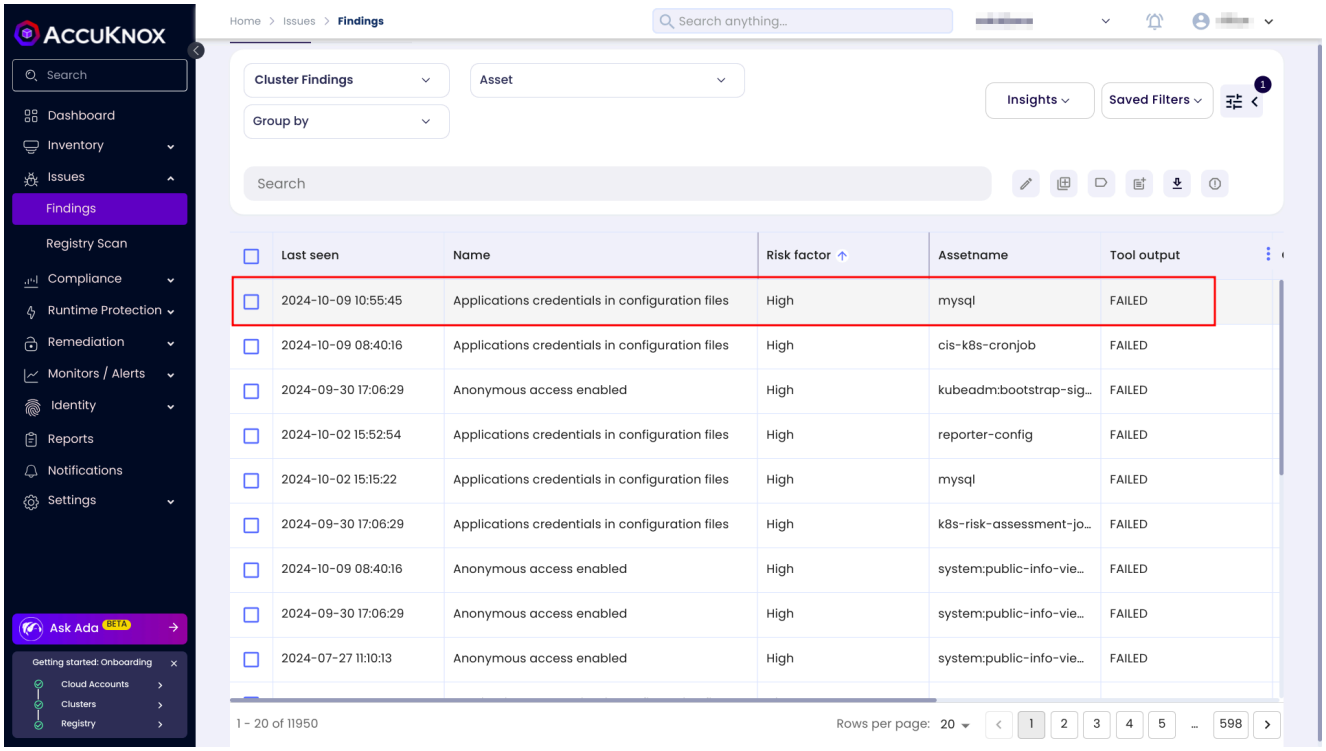


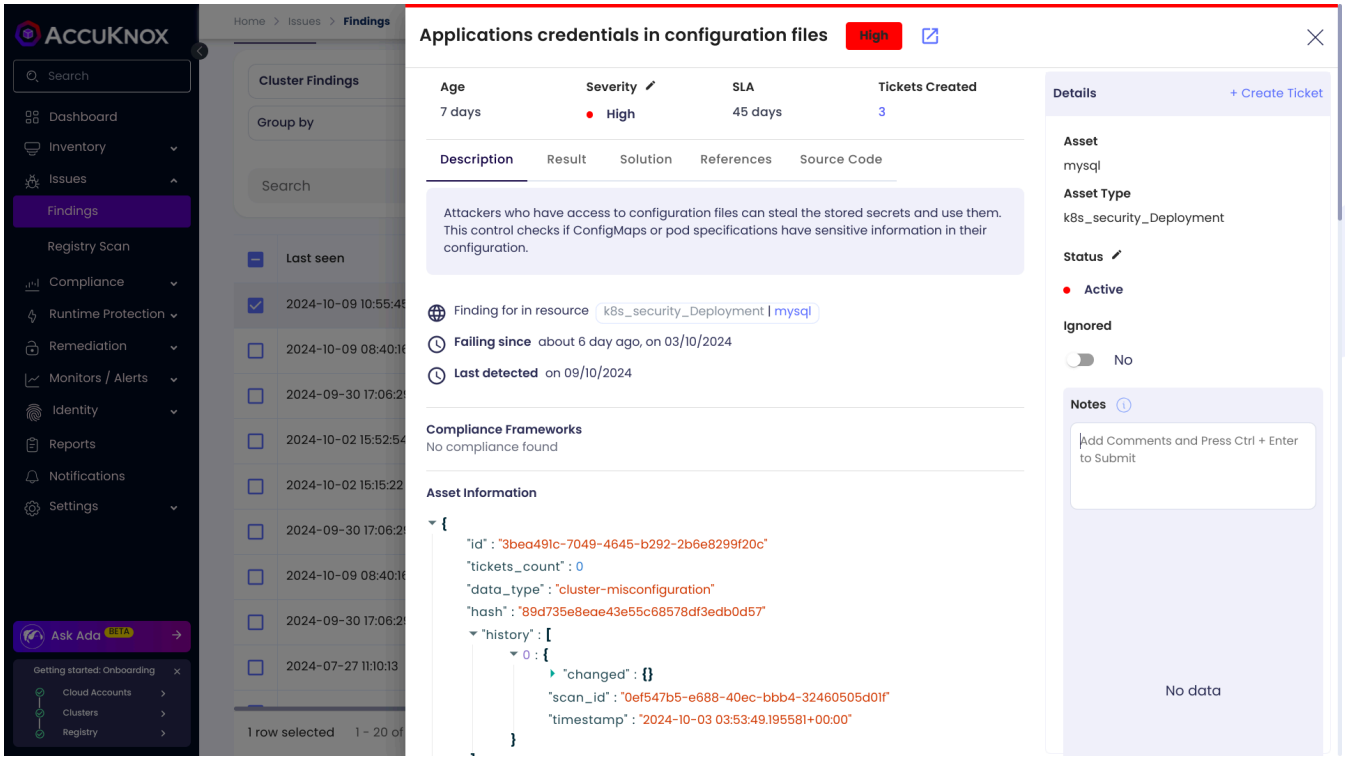
Once the scan is completed you can see the results on the findings page.

1. Go to the Issues > Findings page.
2. Select the Cluster Finding from the drop down.



Click on any of the findings to see more details.





CIS Benchmarking Compliance Scan Onboarding

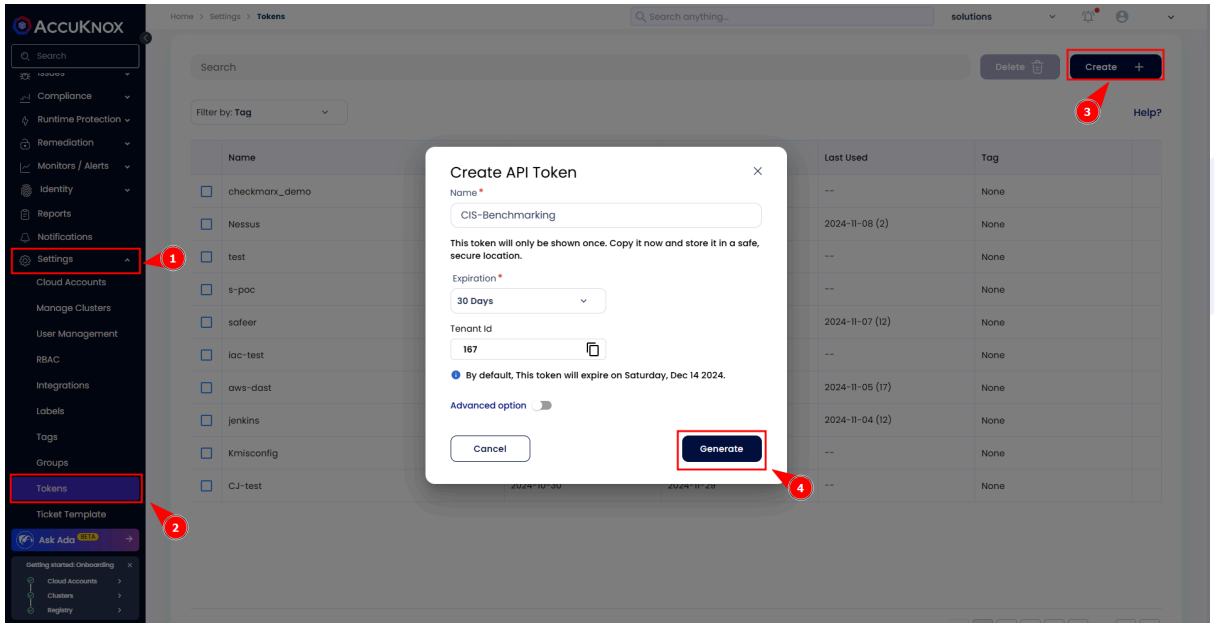
This guide details the steps to onboard a Kubernetes cluster to Accuknox SaaS for CIS Benchmarking compliance scanning, enabling you to monitor and improve cluster security in line with CIS standards.

Step 1: Generate an Access Token

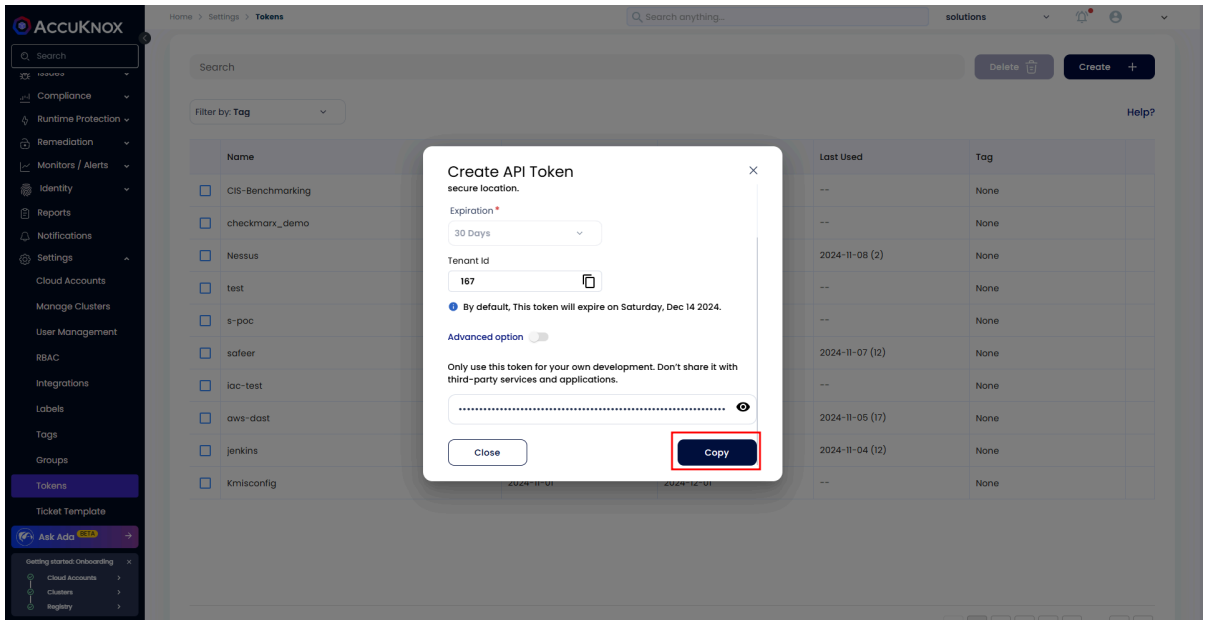
To begin, create a token that will authenticate your cluster for scanning. Follow these steps:

1. Navigate to **Settings** > **Tokens** in the Accuknox platform and Click on the **Create** button, give your token a descriptive name (e.g.,

"CIS-Compliance-Token"), and click **Generate**.

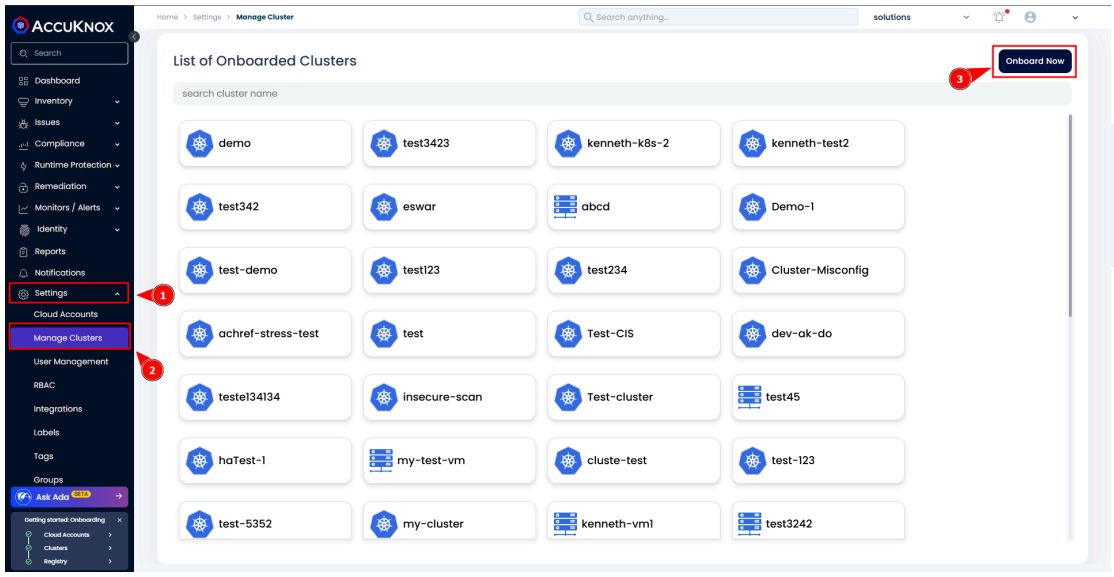


2. Once the token is generated, copy it and securely save it for later use.

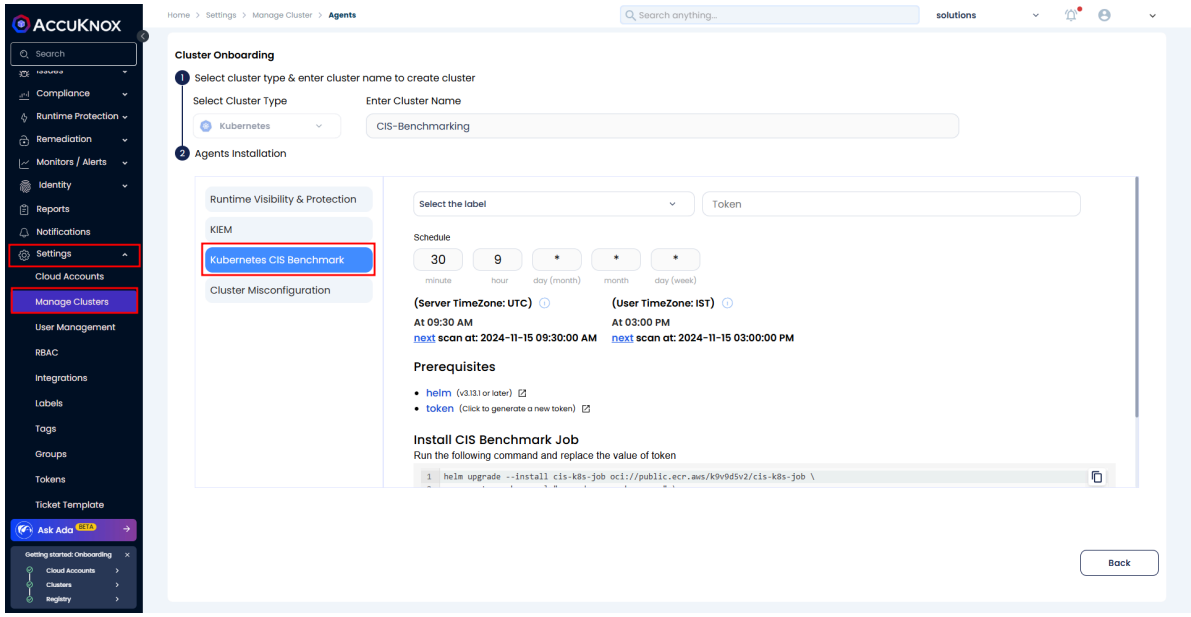


Step 2: Onboard Your Cluster

1. Go to **Settings** > **Manage Clusters** and Click **Onboard Now** or select an existing cluster if you're updating a previously onboarded cluster.

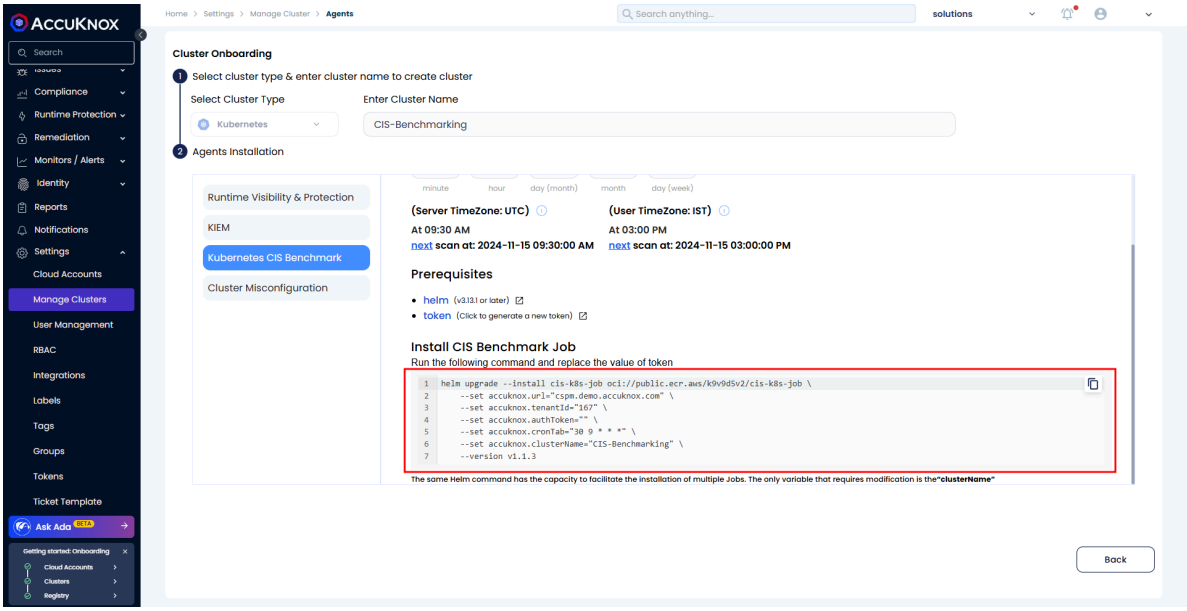


2. Enter a name for your cluster to identify it in Accuknox. From the scan type, choose **CIS Benchmarking.**
3. Select a label for easy identification and paste the token you generated in Step 1. Set a scan schedule based on your requirements. Accuknox will automatically run scans according to the selected schedule.



Step 3: Deploy the Scanner Using Helm

1. Scroll down to the **Helm Command** section and copy the provided command.



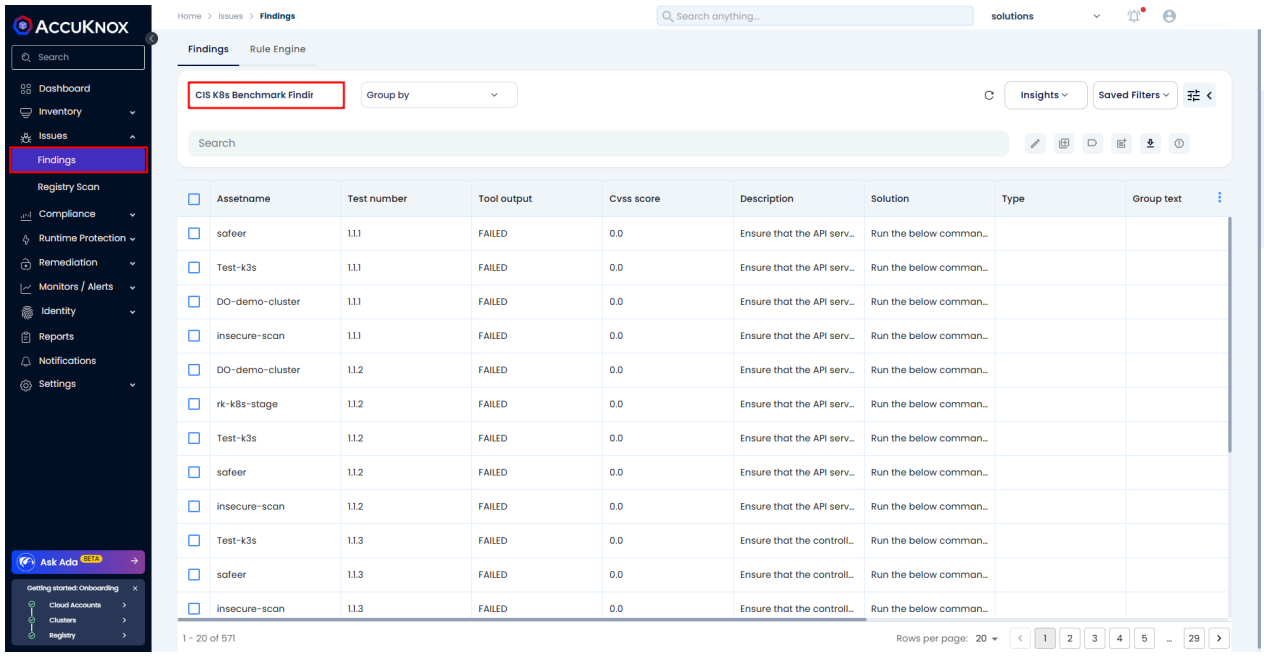
2. Run this command in your terminal on a machine that has access to your Kubernetes cluster. The command will schedule the scan for CIS Benchmarking compliance.
3. Once the Helm installation is complete, return to the Accuknox platform and click **Finish**.

Step 4: View Compliance Findings

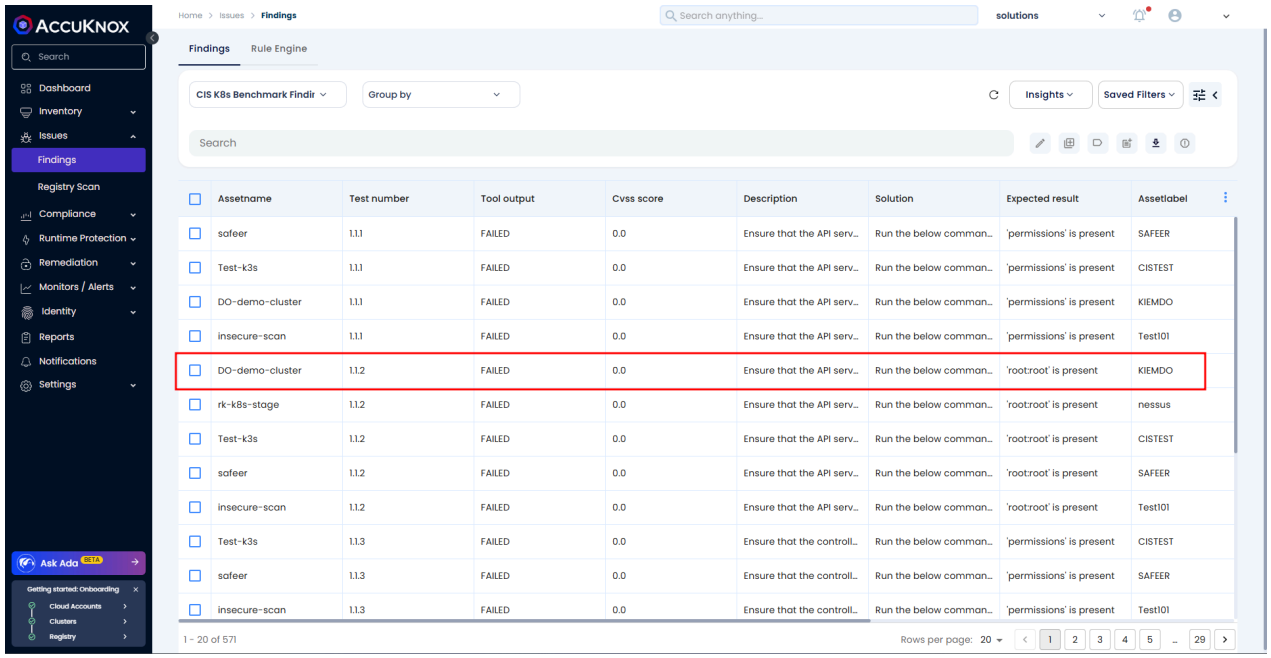
After the initial scan is completed, you can view the compliance results:

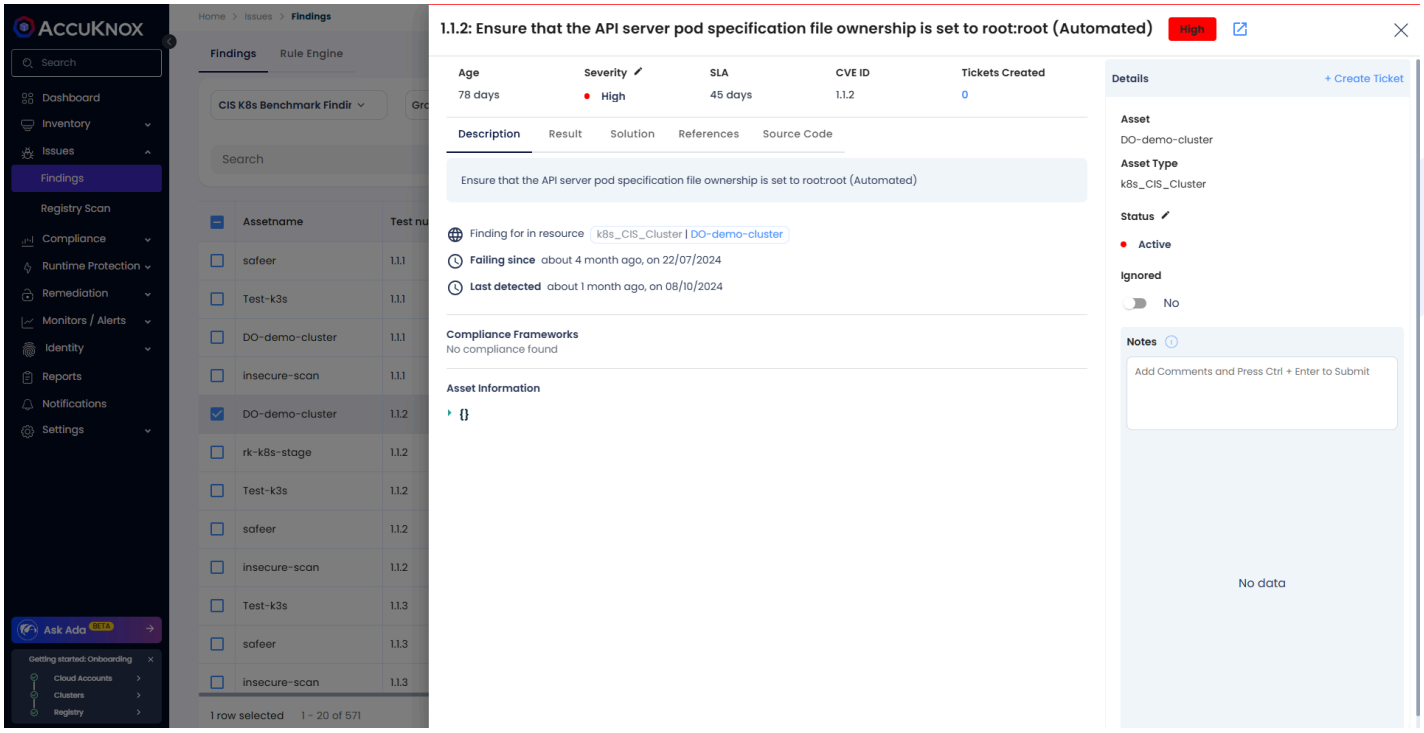
1. Go to **Issues** > **Findings** in Accuknox.

2. Use the **Findings** dropdown to filter and select CIS k8s Benchmarking finding results.



3. Each result will provide details on specific CIS controls and any non-compliant configurations detected.





This completes the onboarding process for CIS Benchmarking compliance scanning. You can review findings regularly to maintain and improve your cluster's CIS compliance.

Cluster Offboarding

This guide outlines the steps for offboarding a cluster from AccuKnox SaaS. The process involves uninstalling the agents from the cluster and deleting the cluster from AccuKnox SaaS.

Below, you will find detailed instructions for agent uninstallation from your cluster CLI and deleting the cluster from AccuKnox SaaS. These steps apply to all clusters.

Agents Uninstallation

Uninstall AccuKnox agents using the following commands:

```
helm uninstall agents -n agents && kubectl delete ns agents;
```

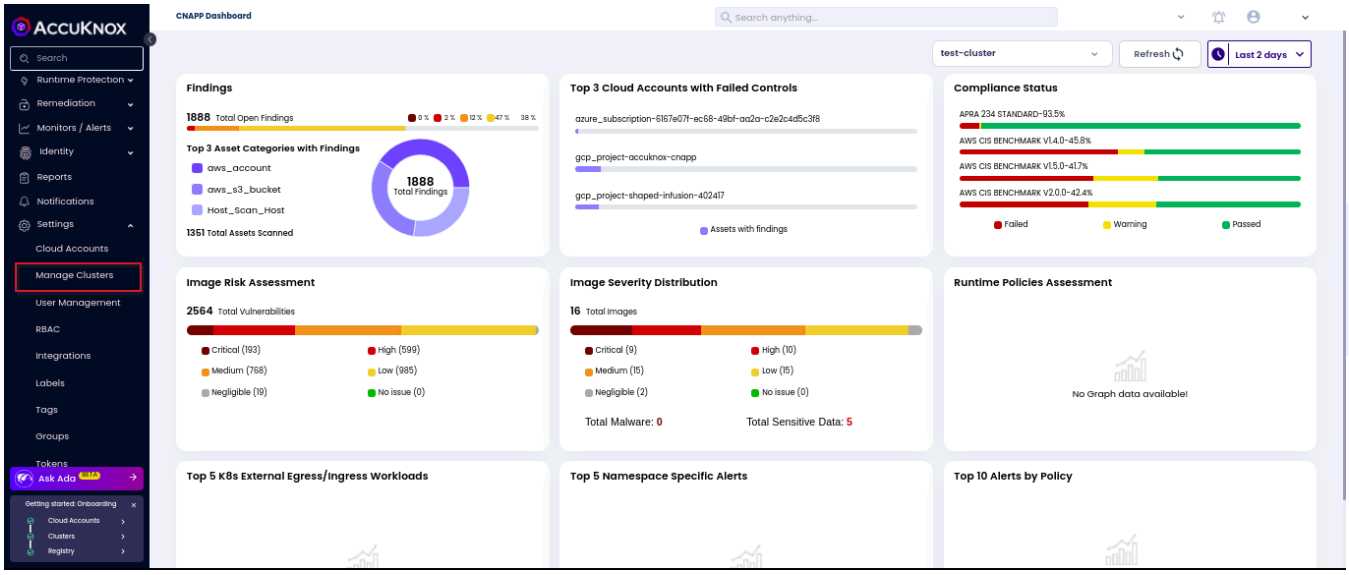
```
helm uninstall cis-k8s-job;

helm uninstall kiem-job;

helm uninstall k8s-risk-assessment-job
```

Sample for Uninstalling Runtime Visibility & Protection agents

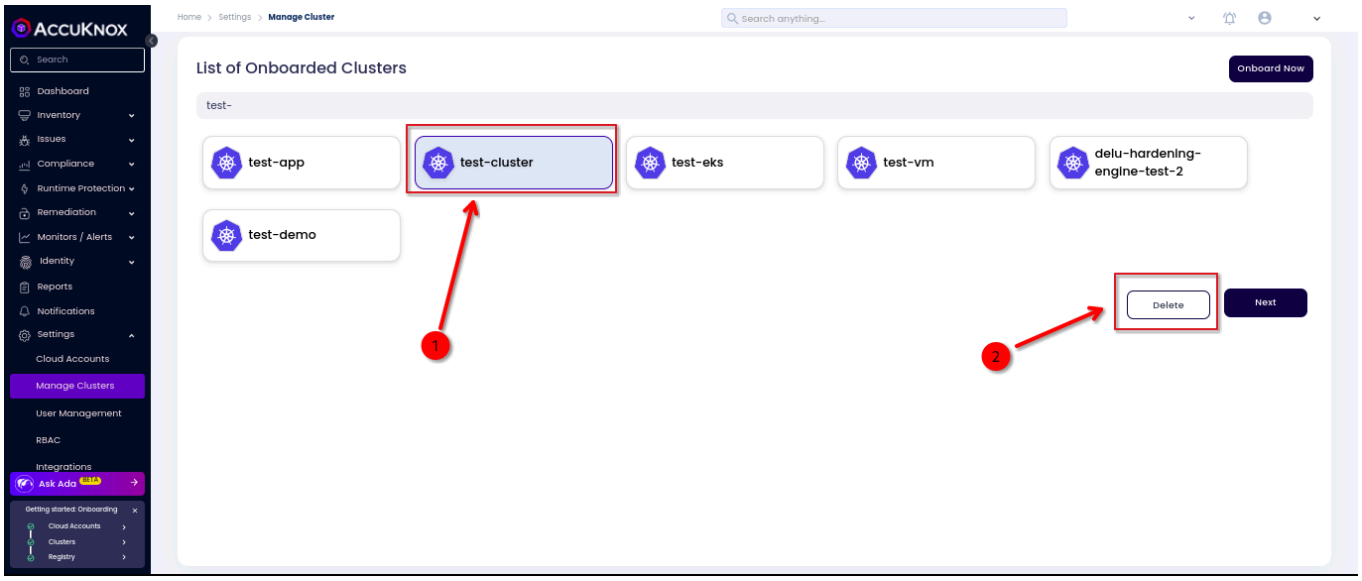
```
(Accuknox㉿kali)-[~]

└─$ helm uninstall agents -n agents && kubectl delete ns agents

WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /etc/rancher/k3s/k3s.yaml

WARNING: Kubernetes configuration file is world-readable. This is insecure. Location: /etc/rancher/k3s/k3s.yaml

release "agents" uninstalled

namespace "agents" deleted
```

Cluster Deletion

Step 1: Login to AccuKnox SaaS and Go to Manage Cluster under Settings



Step 2: Select the cluster and click Delete to delete the cluster from SaaS.



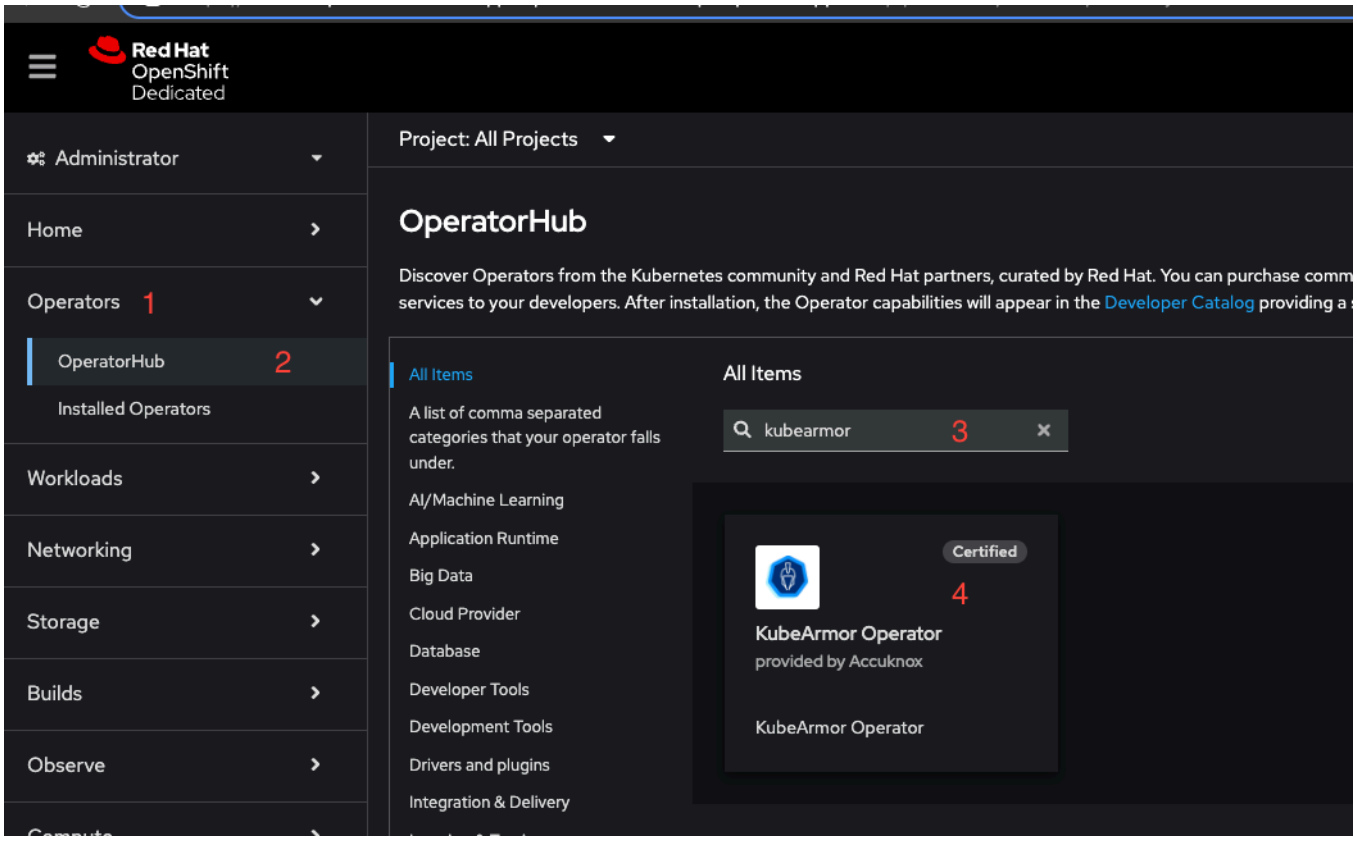
This will delete the cluster from AccuKnox SaaS.

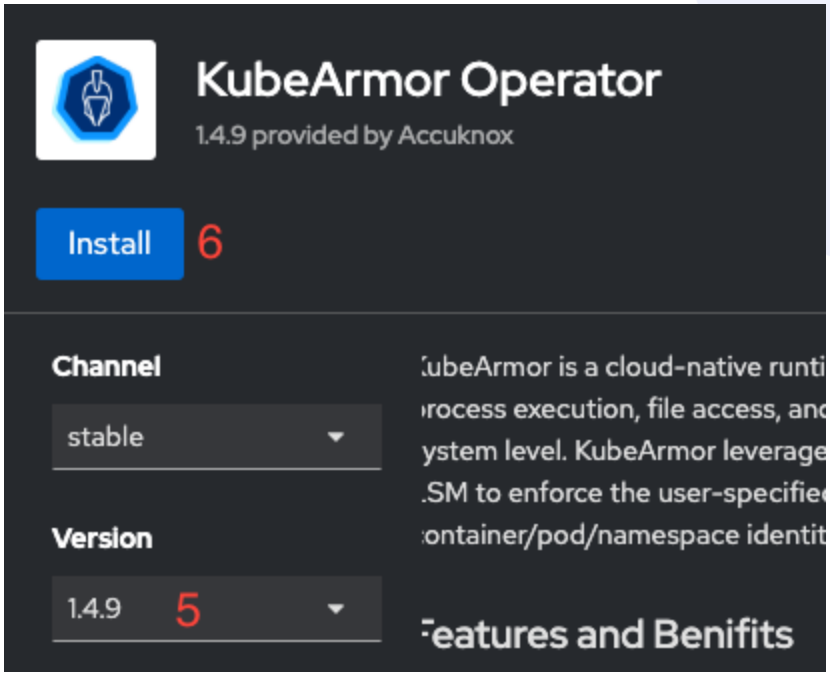
Runtime Security Deployment for Openshift

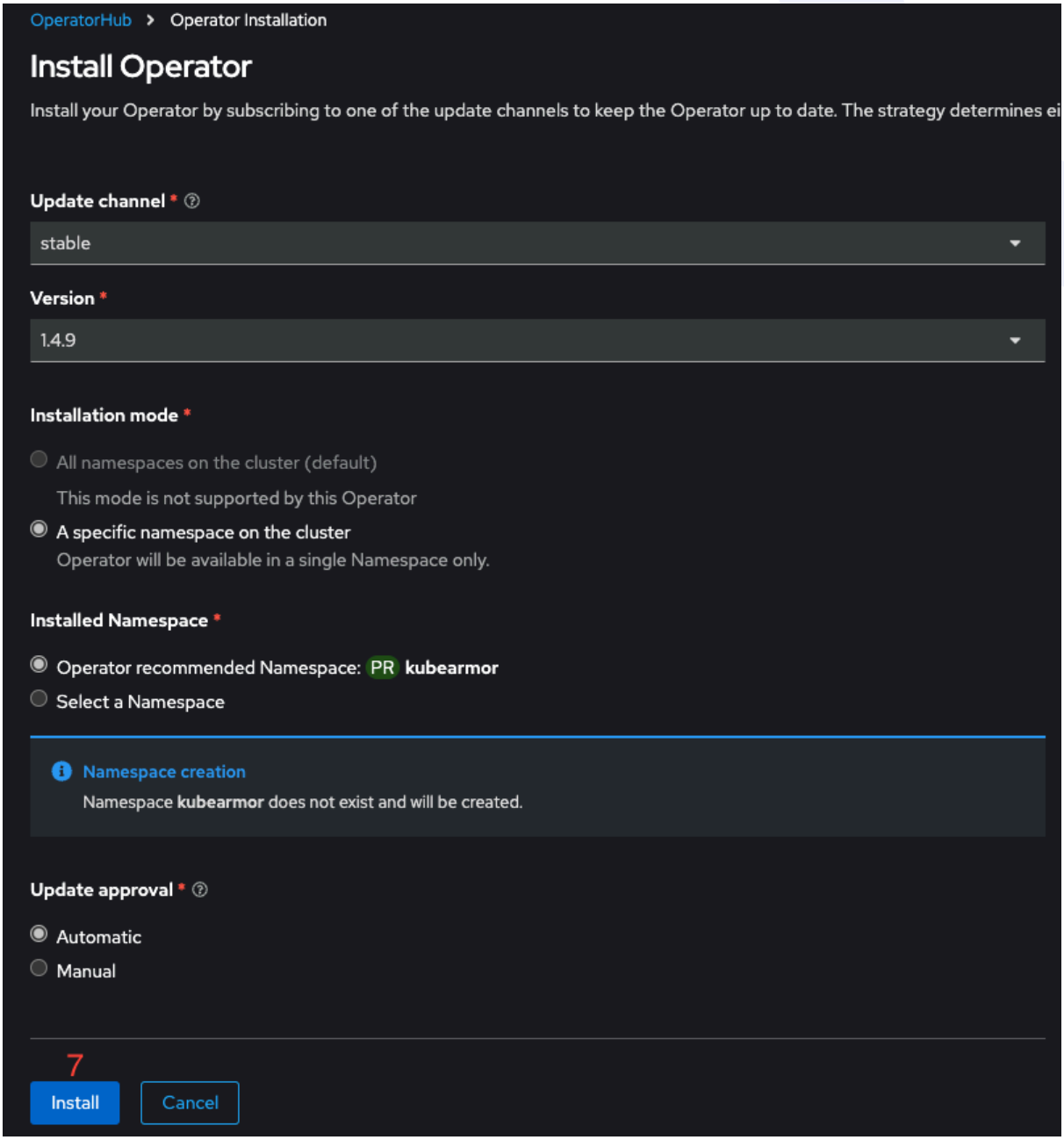
Operator Installation

In the OpenShift console, install KubeArmor operator by following the instructions below:

- Under operators (1) select Operator Hub (2).
- Search for the word "kubearmor" (3) and select "KubeArmor Operator" (4).
- Install KubeArmor version "1.4.9" with default configurations (5, 6, 7).




KubeArmor Operator
1.4.9 provided by Accuknox
Install
6
Channel
stable
Version
1.4.9
5
ubeArmor is a cloud-native runti
rocess execution, file access, and
ystem level. KubeArmor leverage
SM to enforce the user-specifie
ontainer/pod/namespace identit
eatures and Benifits



ElasticSearch Integration

To integrate KubeArmor with Elasticsearch, the following inputs are required:

- **Username/Password**: If the Elasticsearch server requires authentication.
- **CA Certificate**: If Elasticsearch security is enabled.
- **URL of Elasticsearch**: Including protocol and port.

Steps to Install

Username/Password Installation

If the server does not require authentication, you can skip this step. To use username/password authentication with Elasticsearch, a Kubernetes secret called `elastic-secret` needs to be created in the `kubearmor` namespace.

Run the following command, replacing `<elastic-user>` and `<elastic-password>` with appropriate values:

```
kubectl create secret generic elastic-secret -n kubearmor --from-literal
username=<elastic-user> --from-literal password=<elastic-password>
```

CA Certificate Installation

To use HTTPS communication between the agents and Elasticsearch, a Kubernetes secret called `elastic-ca` needs to be created in the `kubearmor` namespace.

- Acquire the CA certificate used by Elasticsearch. If acquiring the certificate is not possible, set the `allowInsecureTLS` flag to `true` in the next steps.
- Save the certificate in a file and run the following command:

```
kubectl create secret generic elastic-ca -n kubearmor --from-file ca.crt=<cacert file name>
```

KubeArmor Instance Installation

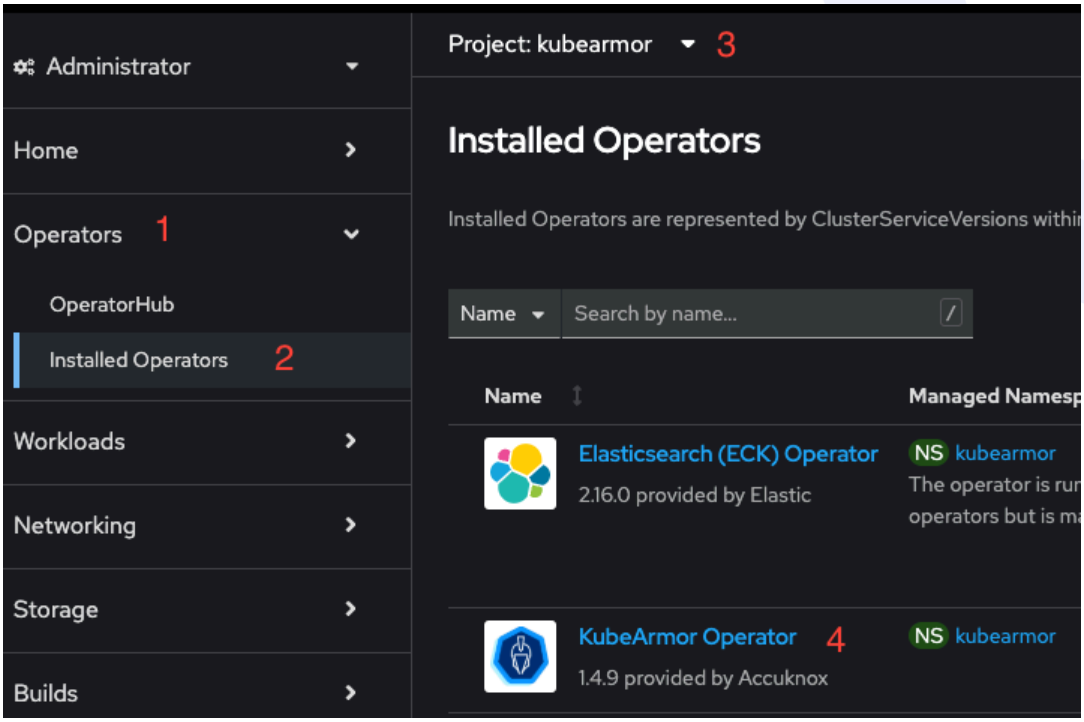
Once the steps in the previous chapter are completed, proceed with the agent installation from the OpenShift console.

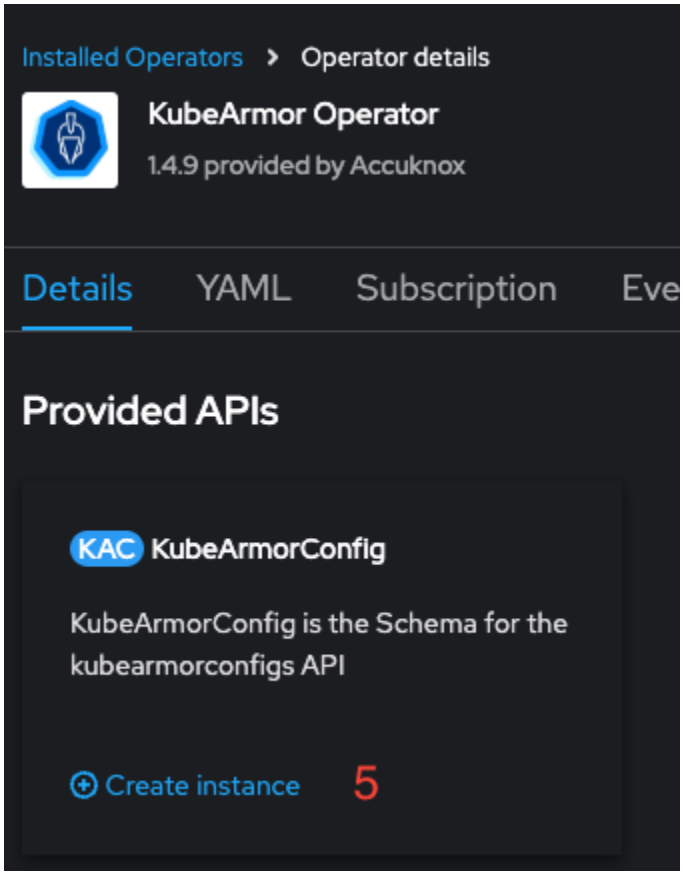
Steps to Install

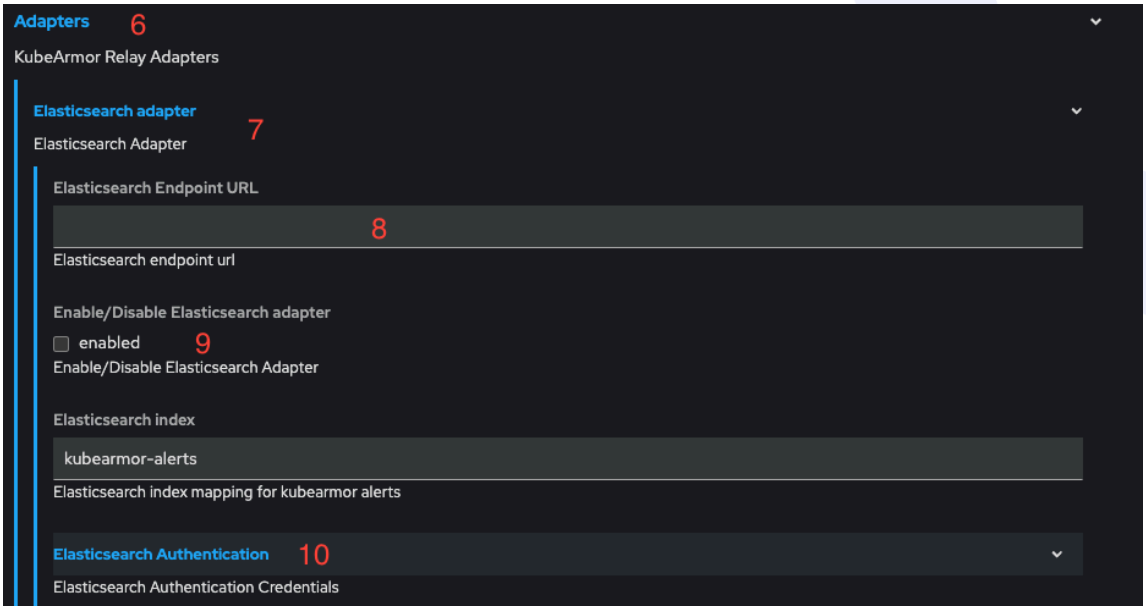
1. Install the required SCC using the following command:

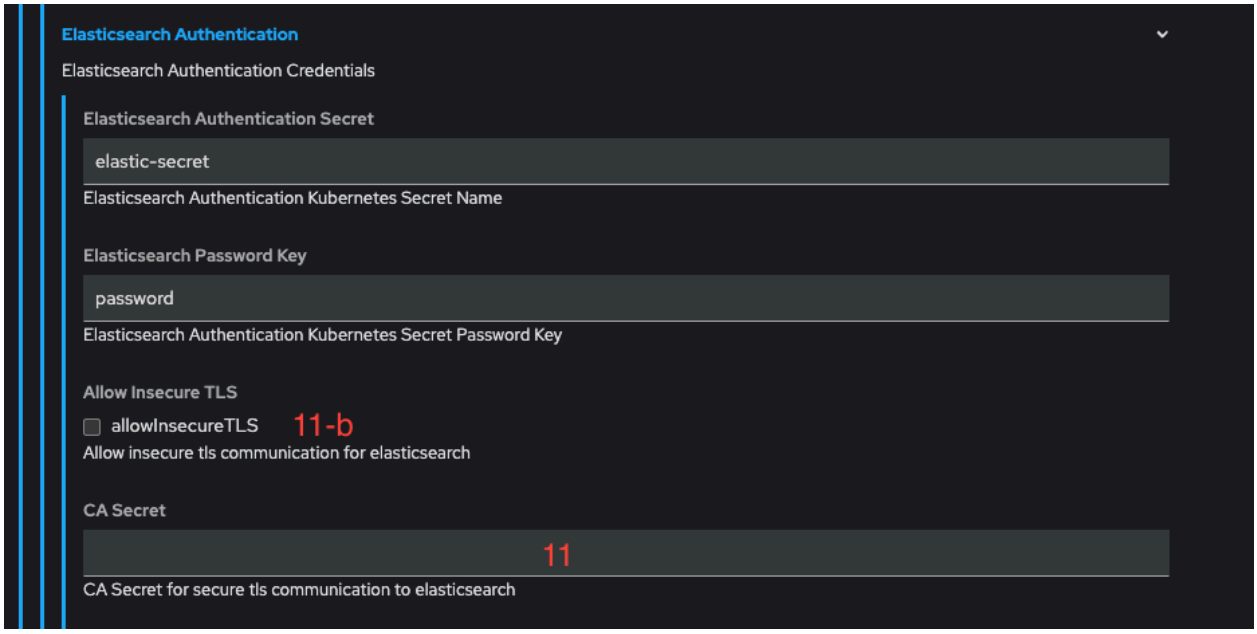
```
oc create -f
https://raw.githubusercontent.com/kubearmor/KubeArmor/main/pkg/KubeArmorOperator/co
nfig/rbac/kubearmor-scc.yaml
```

1. In the OpenShift console:
2. Under Operators (1), go to Installed Operators (2).
3. Select `kubearmor` (3) as the project.
4. Click on the KubeArmor Operator (4).
5. Create a `KubeArmorConfig` Instance (5).
6. In the form view:
7. Select **Adapters** (6) -> **Elasticsearch Adapter** (7).
8. Perform the following steps:
    - Enter the Elasticsearch URL in the field (8).
    - Enable Elasticsearch adapter by checking the checkbox (9).
    - Click on **Elasticsearch Authentication** (10) and:
    - Set the CA secret field (11) to `elastic-ca`.
    - To enable insecure TLS communication (if no certificate is available), check the `allowInsecureTLS` checkbox (11-b) and leave the field (11) empty.
9. Create the instance. The `KubeArmorConfig` Instance controls the installation of the agents in the entire cluster, and only one instance should be created per cluster.


Administrator
Home
Operators 1
OperatorHub
Installed Operators 2
Workloads
Networking
Storage
Builds
Project: kubearmor 3
Installed Operators
Installed Operators are represented by ClusterServiceVersions withi
Name
Search by name...
Name
Managed Namesp
Elasticsearch (ECK) Operator
2.16.0 provided by Elastic
NS kubearmor
The operator is run
operators but is ma
KubeArmor Operator 4
1.4.9 provided by Accuknox
NS kubearmor


Installed Operators > Operator details
KubeArmor Operator
1.4.9 provided by Accuknox
Details
YAML
Subscription
Eve
Provided APIs
KAC KubeArmorConfig
KubeArmorConfig is the Schema for the kubearmorconfigs API
Create instance 5



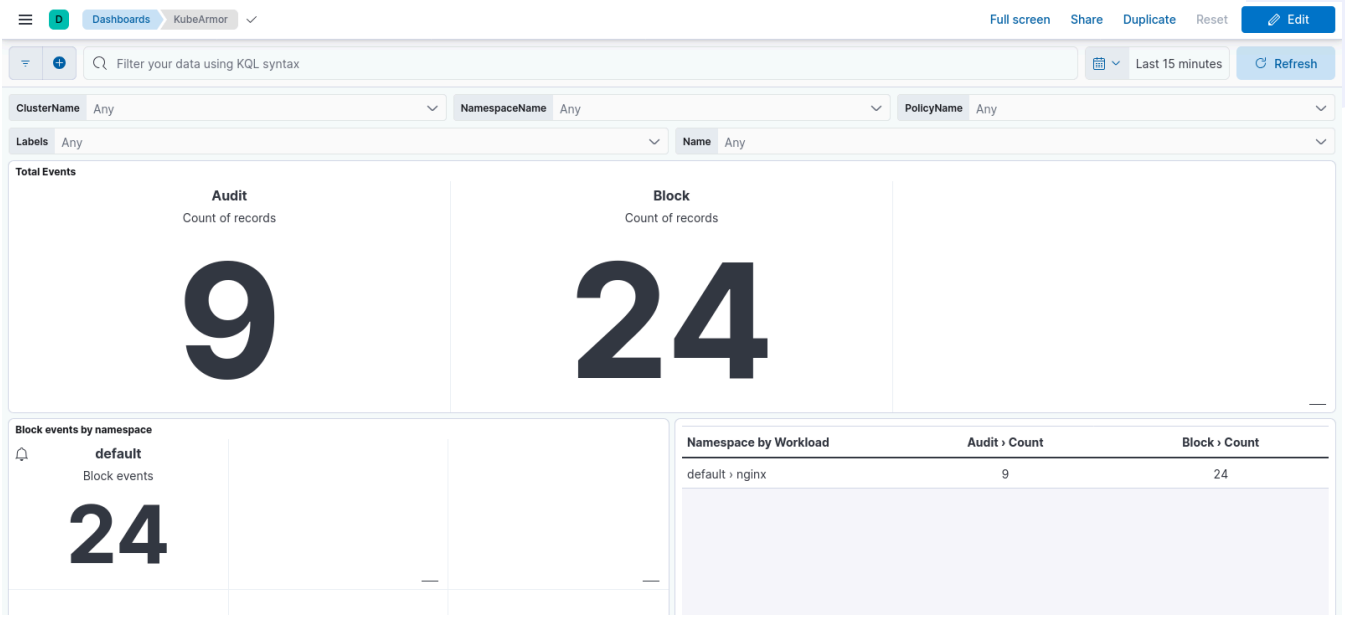


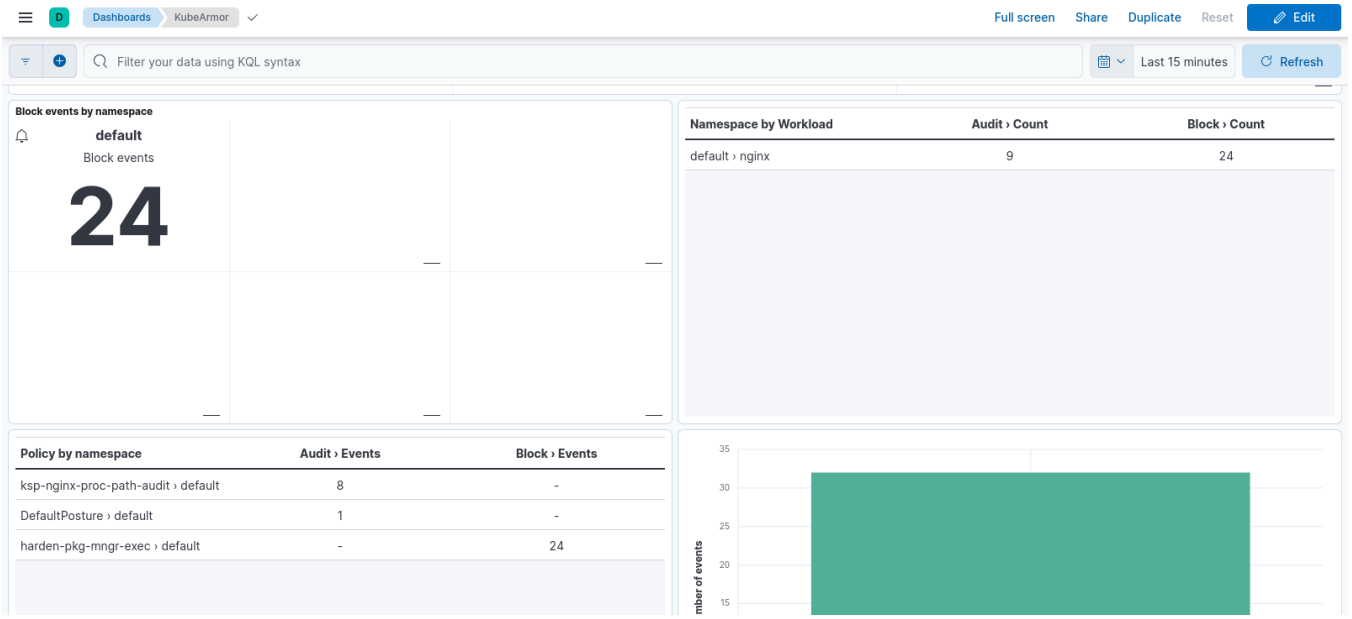
Kibana Dashboard Setup

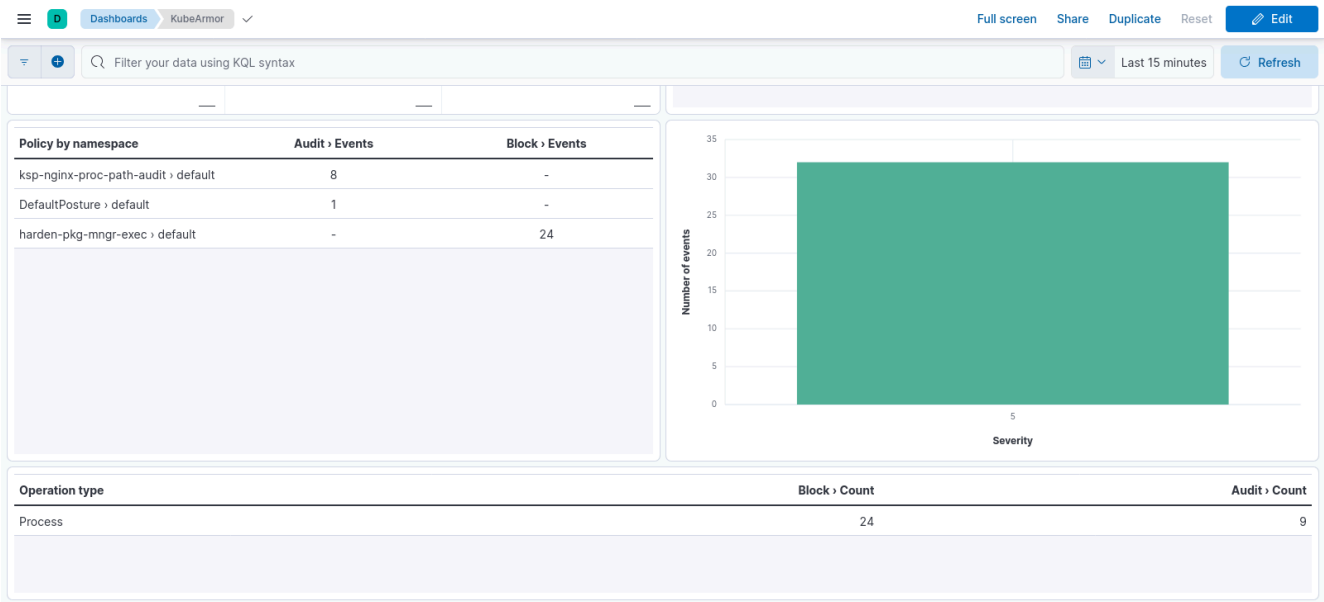
Steps to Install

Along with this document, a file called `kubearmor-dashboard.ndjson` has been shared. Follow these steps to import the dashboard:

1. Under the **Management** tab, select **Stack Management**.
2. Navigate to **Saved Objects** under Kibana.
3. Click **Import** and select `kubearmor-dashboard.ndjson`.







Onboarding and Deboarding VMs with Docker

Docker

Docker v19.0.3 and Docker Compose v1.27.0+ are required. Follow the latest Install Docker Engine for downloading. Ensure you also add your user to the docker user group: Linux post-installation steps for Docker Engine.

Linux Kernel v5.8+ with BPF LSM support is needed. See how to enable BPF LSM.

If the environment does not support Linux v5.8+ or BPF LSM and instead uses AppArmor, host enforcement will still work out of the box. However, to protect containers, new containers must be created with special options. Refer to the "Support for Non-Orchestrated Containers" documentation for more details.

Resource Requirements

Node Type	vCPU	Memory	Disk
Control Plane Node	2	4 GB	24 GB
Worker Node	2	2 GB	12 GB

Network Requirements

Connectivity between control plane node and worker nodes is a must. They should either be:

- Part of the same private network **(recommended & secure)**
- Control plane has a public IP (not recommended)

Ports required on the control plane VM:

Component	Type	Ports	Endpoint	Purpose
Knox-Gateway	Outbound to SaaS	3000	`knox-gw.<env>.accuknox.com:3000`	For Knox-Gat service

PPS	Outbound to SaaS	443	pps.<env>.accuknox.com	For PPS (Poli Provisioning
Spire-Server	Outbound to SaaS	8081, 9090	spire.<env>.accuknox.com	For Spire-Ser communicat
KubeArmor Relay Server	Inbound in Control Plane	32768	-	For KubeArm server on con
Shared Informer Agent	Inbound in Control Plane	32769	-	For Shared In agent on con
Policy Enforcement Agent (PEA)	Inbound in Control Plane	32770	-	For Policy En Agent on con
Hardening Module	Inbound in Control Plane	32771	-	For Discovery Hardening M
VM Worker Nodes	Outbound from worker node to Control Plane	32768-32771	-	For VM worke to connect to control plane

By default, the network created by onboarding commands reserves the subnet `172.20.32.0/27`. If you want to change it for your environment, you can use the `--network-cidr` flag.

You can check the connectivity between nodes using curl. Upon a successful connection, the message returned by curl will be:

```
$ curl <control-plane-addr>:32770
curl: (1) Received HTTP/0.9 when not allowed
```

Onboarding

Navigate to the onboarding page (Settings → Manage Cluster → Onboard Now) and choose the "VM" option on the instructions page. Then, provide a name for your cluster. You will be presented with instructions to download accuknox-cli and onboard your cluster.

The following agents are installed:

1. **Feeder-service** which collects KubeArmor feeds.
2. **Shared-informer-agent** authenticates with your VMs and collects information regarding entities like hosts, containers, and namespaces.
3. **Policy-enforcement-agent** authenticates with your VMs and enforces labels and policies.

Install knoxctl/accuknox-cli

```
curl -sfL https://knoxctl.accuknox.com/install.sh | sudo sh -s -- -b /usr/bin
```

Onboarding Control Plane

The command may look something like this:

```
$ knoxctl onboard vm Control Plane-node \
  --version "v0.2.10" \
  --join-token="843ef458-cecc-4fb9-b5c7-9f1bf7c34567" \
  --spire-host="spire.dev.accuknox.com" \
  --pps-host="pps.dev.accuknox.com" \
  --knox-gateway="knox-gw.dev.accuknox.com:3000"
```

The above command will emit the command to onboard worker nodes. You may also use the `--Control Plane-node-addr` flag to specify the address that other nodes will use to connect with your cluster.

By default, the network created by onboarding commands reserves the subnet `172.20.32.0/27` for the `accuknox-net` Docker network. If you want to change it for your environment, you can use the `--network-cidr` flag.

Onboarding Worker Nodes

The second command will be for onboarding worker nodes. It may look something like this:

```
knoxctl onboard vm node --Control Plane-node-addr=<control-plane-addr>
```

Example:

```
$ knoxctl onboard vm node --Control Plane-node-addr=192.168.56.106
Pulling kubearmor-init      ... done
Pulling kubearmor           ... done
Pulling kubearmor-vm-adapter ... done
Creating network "accuknox-config_accuknox-net" with the default driver
Creating kubearmor-init ... done
Creating kubearmor      ... done
Creating kubearmor-vm-adapter ... done
onboard-vm-node.go:41: VM successfully joined with control-plane!
```

Troubleshooting

If you encounter any issues while onboarding, use the commands below to debug:

```
docker logs spire-agent -f
docker logs shared-informer-agent -f
docker logs kubearmor-init -f
docker logs kubearmor -f
```

Deboarding

Deboard the cluster from SaaS first.

To deboard the worker-vm/Node:

```
knoxctl deboard vm node
```

To deboard the Control-Plane VM:

```
knoxctl deboard vm Control Plane-node
```

Sample Output:

```
$ knoxctl deboard vm Control Plane-node
[+] Running 10/10
 ✔ Container shared-informer-agent      Removed                                    0.6s
 ✔ Container feeder-service             Removed                                    0.6s
 ✔ Container policy-enforcement-agent   Removed                                    0.8s
 ✔ Container wait-for-it                Removed                                    0.0s
 ✔ Container kubearmor-vm-adapter       Removed                                    5.6s
 ✔ Container kubearmor-relay-server     Removed                                    1.5s
 ✔ Container spire-agent                Removed                                    0.5s
 ✔ Container kubearmor                  Removed                                    10.4s
 ✔ Container kubearmor-init             Removed                                    0.0s
 ✔ Network accuknox-config_accuknox-net  Removed                                   0.3s
Please remove any remaining resources at /home/user/.accuknox-config
Control plane node deboarded successfully.
```

After that cleanup the ~/.accuknox-config directory

```
sudo rm -rf ~/.accuknox-config
```

Onboarding and Deboarding VMs with Systemd

Systemd

Systemd is a core component of modern Linux systems responsible for managing services and processes. It ensures that essential services start automatically during boot, remain running, and restart if they fail. In simple terms, systemd acts like a **controller** that organizes and oversees everything needed to keep the system stable and functional.

Currently, **root/sudo** permissions are needed for onboarding systemd. This is because KubeArmor requires privileges to protect the host and systemd services, packages are currently installed on the root directory.

Only in case of the control plane node, a working RabbitMQ server is required. This can be installed using Docker.

```
# Latest RabbitMQ 3.13
docker run -it --rm --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3.13-management
```

Alternatively, you can install RabbitMQ using a package manager:

- **Linux, BSD, UNIX**: Debian, Ubuntu | RHEL, CentOS Stream, Fedora | Generic binary build | Solaris
- **Windows**: Chocolatey package | Windows Installer | Binary build
- **MacOS**: Homebrew | Generic binary build
- Erlang/OTP for RabbitMQ

BTF support is needed. Any kernel version which has this should work. Check if BTF info is present with the script below:

```
if [ ! -e "/sys/kernel/btf/vmlinux" ]; then
  echo "BTF info not present"
else
  echo "BTF info present"
fi
```

If the script returns "BTF info not present," BTF support is not available, and you should run the script below to build the required files on your system:

```
# Download KubeArmor
git clone https://github.com/kubearmor/KubeArmor/
cd KubeArmor/KubeArmor/packaging
./post-install.sh
```

Note

For detailed instructions specific to SystemD Based Non-BTF Environments, please refer to this guide.

Container Protection Requirements (Optional)

If container protection is needed, a Linux Kernel with **BPF LSM** is desired. Generally, it is present in v5.8+. Here's a guide on enabling BPF LSM: KubeArmor Getting Started FAQ.

If BPF LSM is not available, AppArmor should still work out of the box for host policy application. However, follow the guide Support for non orchestrated containers for each container.

Resource Requirements

Control Plane Node (Minimum)

Resource	Requirement
CPU	2 vCPU
Memory	4 GB
Disk	1 GB

Worker Node (Minimum)

Resource	Requirement
CPU	2 vCPU
Memory	2 GB
Disk	500 MB

Network Requirements

Connectivity between control plane node and worker nodes is a must. They should either be:

- Part of the same private network **(recommended & secure)**
- Control plane has a public IP (not recommended)

Ports required on the control plane VM:

Component	Type	Ports	Endpoint	Purpose
Knox-Gateway	Outbound to SaaS	3000	knox-gw.<env>.accuknox.com:3000	For Knox-Gat service

PPS	Outbound to SaaS	443	pps.<env>.accuknox.com	For PPS (Poli Provisioning
Spire-Server	Outbound to SaaS	8081, 9090	spire.<env>.accuknox.com	For Spire-Ser communicat
KubeArmor Relay Server	Inbound in Control Plane	32768	-	For KubeArm server on con
Shared Informer Agent	Inbound in Control Plane	32769	-	For Shared In agent on con
Policy Enforcement Agent (PEA)	Inbound in Control Plane	32770	-	For Policy En Agent on con
Hardening Module	Inbound in Control Plane	32771	-	For Discovery Hardening M
VM Worker Nodes	Outbound from worker node to Control Plane	32768-32771	-	For VM worke to connect to control plane

Check the CWPP documentation for more details on the network requirements.

You can check the connectivity between nodes using curl. Upon a successful connection, the message returned by curl will be:

```
$ curl <control-plane-addr>:32770
curl: (1) Received HTTP/0.9 when not allowed
```

Onboarding

Navigate to the onboarding page (Settings → Manage Cluster → Onboard Now) and choose the "VM" option on the instructions page. Then, provide a name for your cluster. You will be presented with instructions to download accuknox-cli and onboard your cluster.

The following agents will be installed:

1. **Feeder-service** which collects KubeArmor feeds.
2. **Shared-informer-agent** authenticates with your VMs and collects information regarding entities like hosts, containers, and namespaces.
3. **Policy-enforcement-agent** authenticates with your VMs and enforces labels and policies.

Install knoxctl/accuknox-cli

```
curl -sfL https://knoxctl.accuknox.com/install.sh | sudo sh -s -- -b /usr/bin
```

Onboarding Control Plane

The command may look something like this:

```
$ knoxctl onboard vm cp-node \
  --version "v0.2.10" \
  --join-token="843ef458-cecc-4fb9-b5c7-9f1bf7c34567" \
  --spire-host="spire.dev.accuknox.com" \
  --pps-host="pps.dev.accuknox.com" \
  --knox-gateway="knox-gw.dev.accuknox.com:3000"
```

Note

By default, if Docker is not found, systemd mode of installation would be used. If you want to explicitly onboard using systemd services, add the `--vm-mode=systemd` flag to the above command.

The above command will emit the command to onboard worker nodes. You may also use the `--cp-node-addr` flag to specify the address that other nodes will use to connect with your cluster.

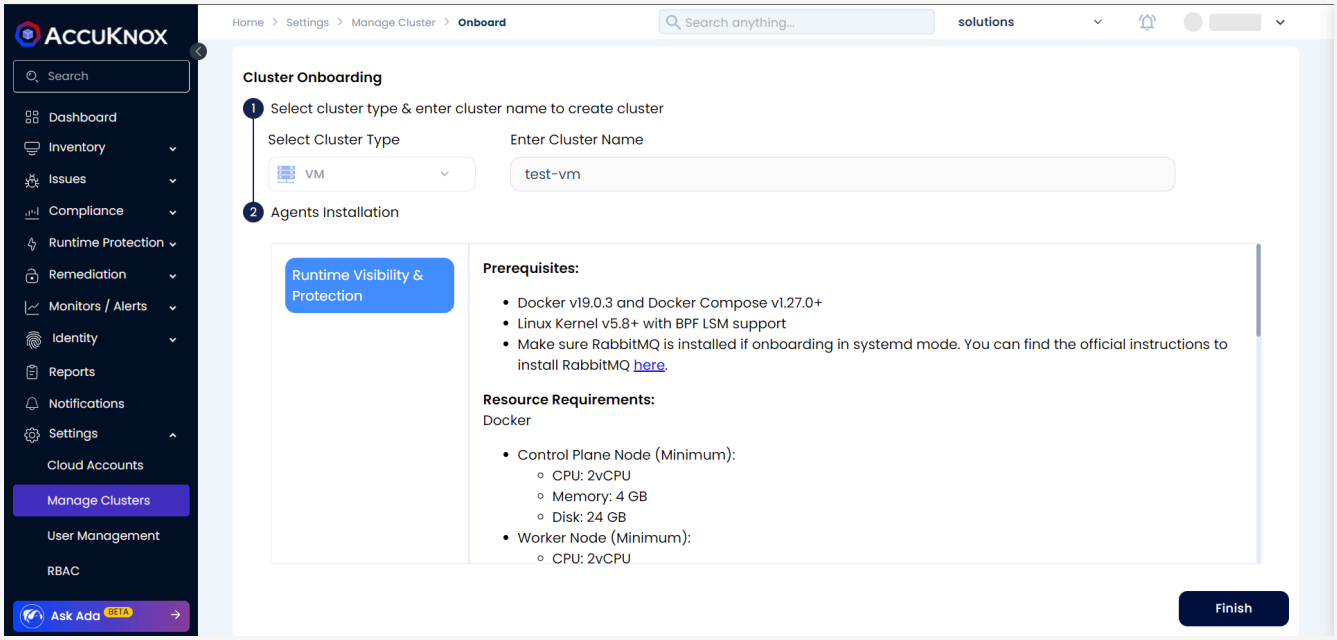
Onboarding Worker Nodes

The second command will be for onboarding worker nodes. It may look something like this:

```
knoxctl onboard vm node --cp-node-addr=<control-plane-addr>
```

Example:

```
$ knoxctl onboard vm node --cp-node-addr=192.168.56.106
Pulling kubearmor-init       ... done
Pulling kubearmor            ... done
Pulling kubearmor-vm-adapter ... done
Creating network "accuknox-config_accuknox-net" with the default driver
Creating kubearmor-init ... done
Creating kubearmor      ... done
Creating kubearmor-vm-adapter ... done
onboard-vm-node.go:41: VM successfully joined with control-plane!
```



Troubleshooting

If you encounter any issues while onboarding, use the commands below to debug:

`sudo journalctl -xeu <service-name>.service`

Replace `<service-name>` with one of the following:

- `kubearmor`: Logs show policy enforcement and monitor Kubernetes workloads; useful for debugging misconfigurations or runtime issues.
- `kubearmor-relay-server`: Bridges KubeArmor clients with external log systems; logs debug communication or relay errors.
- `kubearmor-vm-adapter`: Tracks policy enforcement in VMs; logs diagnose policy application on non-Kubernetes workloads.
- `accuknox-policy-enforcement-agent`: Enforces security policies; logs troubleshoot policy errors or conflicts.
- `accuknox-shared-informer-agent`: Shares Kubernetes resource data; logs debug metadata collection issues.
- `accuknox-sumengine`: Processes telemetry data; logs resolve performance or data processing errors.
- `accuknox-discover-agent`: Discovers potential policies; logs analyze policy suggestions.
- `spire-agent`: Manages workload identities; logs debug identity issuance and attestation issues.

- accuknox-hardening-agent: Automates system hardening; logs troubleshoot configuration and hardening conflicts.

Deboarding

Deboard the cluster from SaaS first.

To deboard the worker-vm/Node:

```
knoxctl deboard vm node
```

To deboard the Control-Plane VM:

```
knoxctl deboard vm cp-node
```

Sample Output:

```
$ knoxctl deboard vm cp-node
[+] Running 10/10
 ✔ Container shared-informer-agent      Removed                                    0.6s
 ✔ Container feeder-service             Removed                                    0.6s
 ✔ Container policy-enforcement-agent   Removed                                    0.8s
 ✔ Container wait-for-it                Removed                                    0.0s
 ✔ Container kubearmor-vm-adapter       Removed                                    5.6s
 ✔ Container kubearmor-relay-server     Removed                                    1.5s
 ✔ Container spire-agent                Removed                                    0.5s
 ✔ Container kubearmor                  Removed                                    10.4s
 ✔ Container kubearmor-init             Removed                                    0.0s
 ✔ Network accuknox-config_accuknox-net  Removed                                   0.3s
Please remove any remaining resources at /home/user/.accuknox-config
Control plane node deboarded successfully.
```

After that cleanup the ~/.accuknox-config directory

```
sudo rm -rf ~/.accuknox-config
```

SystemD Based Non-BTF Environments

Compiling system monitor

Some Kernels don't have BTF information available which is required by KubeArmor's system monitor to work out of the box. Thus, the monitor has to be built either on the target machine or on a machine which matches the kernel version of the target machine.

There are two ways to do it, you can chose either one:

Compile system monitor using Docker (Recommended and reliable)

1. Dependencies:
    - Make sure you have docker installed
    - Make sure you have linux-headers installed for your package
2. Run the kubearmor-init container using the below command which will generate the file `/tmp/system_monitor.bpf.o`.

```
sudo docker run --rm -d --name=kubearmor-init --privileged \
-v "/tmp:/opt/kubearmor/BPF:rw" \
-v "/lib/modules:/lib/modules:ro" \
-v "/sys/kernel/security:/sys/kernel/security:ro" \
-v "/sys/kernel/debug:/sys/kernel/debug:ro" \
-v "/media/root/etc/os-release:/media/root/etc/os-release:ro" \
-v "/usr/src:/usr/src" \
kubearmor/kubearmor-init:stable
```

Compile system monitor directly (Might not work for some versions)

Get the KubeArmor version from Release v1.4.3 - kubearmor/KubeArmor

Fetch and install KubeArmor by running

```
VER="1.4.3" # set according to the latest version
```

```
curl -sfLO
<https://github.com/kubearmor/KubeArmor/releases/download/v${VER}/kubearmor_${VER}_linux-amd64.deb>
sudo apt install ./kubearmor_${VER}_linux-amd64.deb
```

The above will generate the system monitor file at `/opt/kubearmor/BPF/system_monitor.bpf.o`. Copy it to some other path.

Onboard the node

Once you've compiled the monitor, you can specify it while onboarding the control plane/node.

Install `knoxctl` - the accuknox CLI by running the below command

```
curl -sfL <https://knoxctl.accuknox.com/install.sh> | sudo sh -s -- -b /usr/local/bin
```

Onboard your node/control plane by running the respective command with the below additional flags

```
sudo knoxctl onboard vm cp-node \
... usual flags
--skip-btf-check=true \

--system-monitor-path=/tmp/system_monitor.bpf.o
```

VM Onboarding using Access Keys

Overview

The access key method simplifies the onboarding of multiple VMs as control plane VMs. The process mirrors that of SystemD mode and Docker Container mode. Using

the access key, users can onboard a VM directly from the CLI without needing to access the AccuKnox SaaS interface.

Users can select either SystemD or Docker Container mode for onboarding, as the same access key works for both. Moreover, the access key provides enhanced flexibility, enabling the onboarding of multiple control plane VMs with a single key

Here we will follow the `SystemD` mode of onboarding

Pre-requisites

1. Access Key
2. Resource requirements
3. Network requirements
4. BTF support is enabled in the VM
5. RabbitMQ should be installed

Onboarding

In the case of the Access key onboarding method User can directly onboard the VMs from the CLI

NOTE

We don't need to follow AccuKnox UI for the access key method of the VM onboarding; we will be using a command to do the same from the CLI.

Install knoxctl/accuknox-cli

```
curl -sfL https://knoxctl.accuknox.com/install.sh | sudo sh -s -- -b /usr/bin
```

Onboarding Control Plane

The command may look something like this:

```
knoxctl onboard vm cp-node \
 --version v0.8.1 \
 --spire-host=spire.demo.accuknox.com \
 --pps-host=pps.demo.accuknox.com \
 --knox-gateway=knox-gw.demo.accuknox.com:3000 \
 --vm-name="accuknox-vm" \
 --access-key-url="cwpp.demo.accuknox.com" \
 --access-key="access-token"
```

In the above command, You need to replace the `--access-token` value with the created access key, and substitute `--vm-name` with the desired vm name. After replacing the value the command will look like this:

By default, if Docker is not found, systemd mode of installation would be used. If you want to explicitly onboard using systemd services, add the `--vm-mode=systemd` flag to the above command.

Output

```
Downloading agents...
Downloading Agent - kubearmor | Image - docker.io/kubearmor/kubearmor-systemd:1.4.3_linux-amd64
kubearmor version 1.4.3_linux-amd64 downloaded successfully

Downloading Agent - vm-adapter | Image - docker.io/accuknox/vm-adapter-systemd:0.1.4_linux-amd64
vm-adapter version 0.1.4_linux-amd64 downloaded successfully

Downloading Agent - kubearmor-relay-server | Image - docker.io/accuknox/kubearmor-relay-server-systemd:0.0.4_linux-amd64
kubearmor-relay-server version 0.0.4_linux-amd64 downloaded successfully

Downloading Agent - spire-agent | Image - docker.io/accuknox/spire-agent-systemd:1.9.4_linux-amd64
spire-agent version 1.9.4_linux-amd64 downloaded successfully

Downloading Agent - accuknox-shared-informer-agent | Image - docker.io/accuknox/accuknox-shared-informer-agent-systemd:0.7.3_linux-amd64
accuknox-shared-informer-agent version 0.7.3_linux-amd64 downloaded successfully

Downloading Agent - accuknox-policy-enforcement-agent | Image - docker.io/accuknox/accuknox-policy-enforcement-agent-systemd:0.6.4_linux-amd64
accuknox-policy-enforcement-agent version 0.6.4_linux-amd64 downloaded successfully

Downloading Agent - accuknox-feeder-service | Image - docker.io/accuknox/accuknox-feeder-service-systemd:0.7.4_linux-amd64
accuknox-feeder-service version 0.7.4_linux-amd64 downloaded successfully

Downloading Agent - accuknox-sumengine | Image - docker.io/accuknox/accuknox-sumengine-systemd:0.2.4_linux-amd64
accuknox-sumengine version 0.2.4_linux-amd64 downloaded successfully

Downloading Agent - accuknox-discover | Image - docker.io/accuknox/accuknox-discover-systemd:0.2.4_linux-amd64
accuknox-discover version 0.2.4_linux-amd64 downloaded successfully

Downloading Agent - accuknox-hardening-agent | Image - docker.io/accuknox/accuknox-hardening-agent-systemd:0.2.4_linux-amd64
accuknox-hardening-agent version 0.2.4_linux-amd64 downloaded successfully

All agents downloaded successfully.

Configuring services...

Enabling services...
Started kubearmor.service
Started kubearmor-vm-adapter.service
Started kubearmor-relay-server.service
Started spire-agent.service
Started accuknox-shared-informer-agent.service
Started accuknox-policy-enforcement-agent.service
Started accuknox-feeder-service.service
Started accuknox-sumengine.service
Started accuknox-discover.service
Started accuknox-hardening-agent.service

Cleaning up downloaded assets...
VM successfully onboarded!

Now run the below command to onboard any worker nodes.
Please assign appropriate IP address to --cp-node-addr to make sure
that worker nodes can connect to this node
knoxctl onboard vm node --vm-mode="systemd"  --version=v0.8.1 --cp-node-addr=<address-of-this-node>
```

The above command will emit the command to onboard worker nodes. You may also use the `--cp-node-addr` flag to specify the address that other nodes will use to connect with your cluster.

NOTE

The user needs to repeat the CLI onboarding command to onboard multiple control plane VMs using the access key

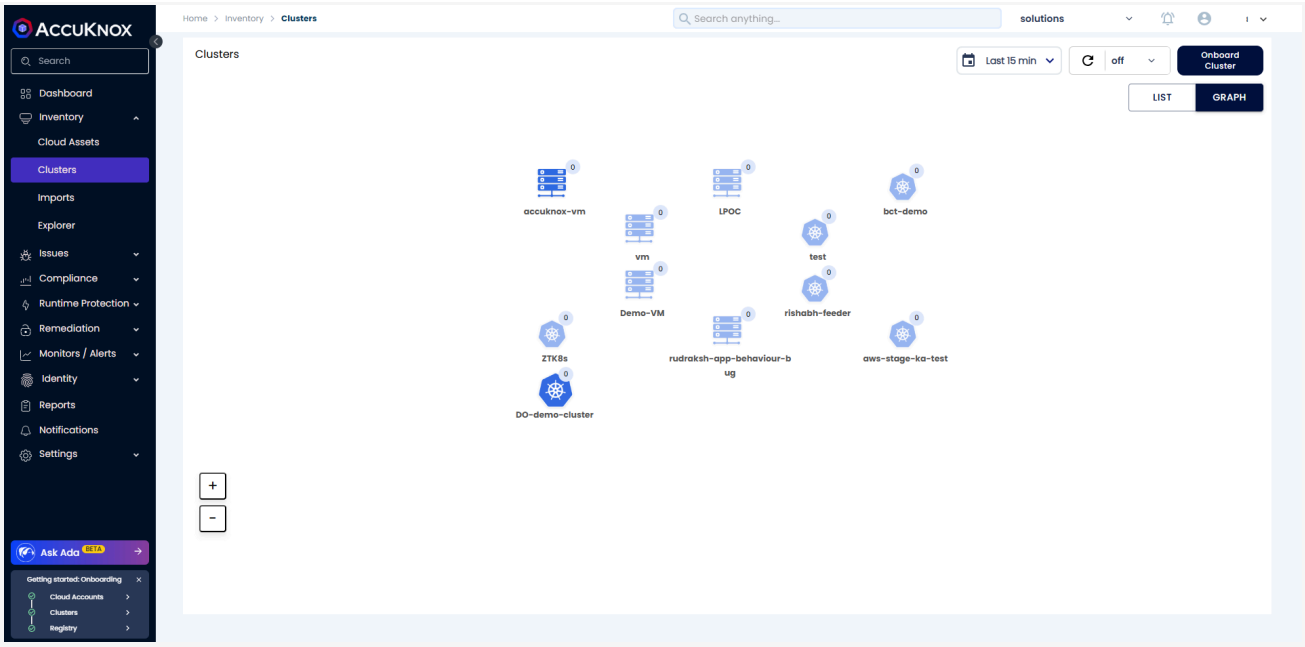
Onboarding Worker Nodes

The second command will be for onboarding worker nodes. It may look something like this:

```
knoxctl onboard vm node --vm-mode="systemd" --version=v0.8.1
--cp-node-addr=<control-plane-addr>
```

Example:

```
$ knoxctl onboard vm node --vm-mode="systemd"  --version=v0.8.1
--cp-node-addr=192.168.56.106
Pulling kubearmor-init       ... done
Pulling kubearmor            ... done
Pulling kubearmor-vm-adapter ... done
Creating network "accuknox-config_accuknox-net" with the default driver
Creating kubearmor-init ... done
Creating kubearmor      ... done
Creating kubearmor-vm-adapter ... done
onboard-vm-node.go:41: VM successfully joined with control-plane!
```



Troubleshooting

If you encounter any issues while onboarding, use the commands below to debug:

```
sudo journalctl -xeu <service-name>.service
```

Replace `<service-name>` with one of the following:

- `kubearmor`: Logs show policy enforcement and monitor Kubernetes workloads; useful for debugging misconfigurations or runtime issues.
- `kubearmor-relay-server`: Bridges KubeArmor clients with external log systems; logs debug communication or relay errors.
- `kubearmor-vm-adapter`: Tracks policy enforcement in VMs; logs diagnose policy application on non-Kubernetes workloads.
- `accuknox-policy-enforcement-agent`: Enforces security policies; logs troubleshoot policy errors or conflicts.
- `accuknox-shared-informer-agent`: Shares Kubernetes resource data; logs debug metadata collection issues.
- `accuknox-sumengine`: Processes telemetry data; logs resolve performance or data processing errors.
- `accuknox-discover-agent`: Discovers potential policies; logs analyze policy suggestions.
- `spire-agent`: Manages workload identities; logs debug identity issuance and attestation issues.
- `accuknox-hardening-agent`: Automates system hardening; logs troubleshoot configuration and hardening conflicts.

Deboarding

Deboard the cluster from SaaS first.

To deboard the worker-vm/Node:

```
knoxctl deboard vm node
```

To deboard the Control-Plane VM:

```
knoxctl deboard vm cp-node
```

Sample Output:

```
$ knoxctl deboard vm cp-node
[+] Running 10/10
 ✔ Container shared-informer-agent      Removed                                    0.6s
 ✔ Container feeder-service             Removed                                    0.6s
 ✔ Container policy-enforcement-agent   Removed                                    0.8s
 ✔ Container wait-for-it                Removed                                    0.0s
 ✔ Container kubearmor-vm-adapter       Removed                                    5.6s
 ✔ Container kubearmor-relay-server     Removed                                    1.5s
 ✔ Container spire-agent                Removed                                    0.5s
```

```
✔ Container kubearmor              Removed                                    10.4s
✔ Container kubearmor-init         Removed                                     0.0s
✔ Network accuknox-config_accuknox-net  Removed                                0.3s
Please remove any remaining resources at /home/user/.accuknox-config
Control plane node deboarded successfully.
```

After that cleanup the ~/.accuknox-config directory

```
sudo rm -rf ~/.accuknox-config
```

In-Cluster Image Scanning with Helm

AccuKnox offers an in-cluster container image scanning solution designed to periodically inspect container images deployed within your Kubernetes (K8s) environment. This automated scanning process detects known vulnerabilities, promoting compliance and enhancing your cluster's overall security. All scan results, including detailed vulnerability insights, are automatically sent to the AccuKnox Control Plane, where they can be viewed and managed through an intuitive user interface.

🛠️ Installation Guide

Follow these steps to deploy the in-cluster image scanner using Helm:

1. Create a Label

In the AccuKnox Control Plane, create a unique **Label**. This will be associated with the container image scan reports.

2. Generate a Token

From the AccuKnox Control Plane:

- Generate an **Artifact Token**

- Note down both the **Token** and your **Tenant ID**

3. Schedule and Deploy the Scanner via Helm

Use the following Helm command to install the scanner in your Kubernetes cluster:

```
helm install kubeshield oci://public.ecr.aws/k9v9d5v2/kubeshield-chart -n agents
--create-namespace \
 --set scan.tenantId="" \
 --set scan.authToken="" \
 --set scan.url="" \
 --set scan.label="" \
 --set scan.cronTab="30 9 * * *" \
 --version "v0.1.2"
```

Replace the parameters („ , and ` `) with the appropriate values.

Sample Output

```
Pulled: public.ecr.aws/k9v9d5v2/kubeshield-chart:v0.1.1
Digest: sha256:a4c1a8948db7a24d8990b71b53184f564960b2b39dbd6cba1cd6104c12addd75
NAME: kubeshield
LAST DEPLOYED: Mon May  5 10:08:24 2025
NAMESPACE: agents
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

⚙️ Parameters:

Variable	Sample Value	Description

tenantId	11	AccuKnox Tenant ID
authToken	eyJhbGc...	AccuKnox Token
url	cspm.accuknox.com	AccuKnox CSPM API Endpoint
label	kubeshield	AccuKnox Label
cronTab	30 9 * * *	Schedule in Cron

Note: Deploy the Scanner via Helm (One Time) If you don't want to schedule and just want to trigger scan for one time, remove this flag `--set scan.cronTab`

✅ Post-Installation

Once the scanner is deployed and completes a scan cycle, results will be visible in the **Findings** or **Registry Scan** sections within the AccuKnox Control Plane.

- Navigate to **Issues -> Findings**
- Switch to **Findings** tab
- Select **Container Image Findings** & do **Group by** based on **Label Name**
- You should be able to see the data for the **Label** used in above command

---

🧪 Scan Status from Cluster

🔧 Check if `kubeshield-controller-manager` is running fine or not

```
kubectl get po -n kubeshield
NAME                                    READY   STATUS    RESTARTS   AGE
kubeshield-controller-manager-5dd5cbc6d4-8xg8k   1/1     Running   0          22s
```

STATUS should be **Running**

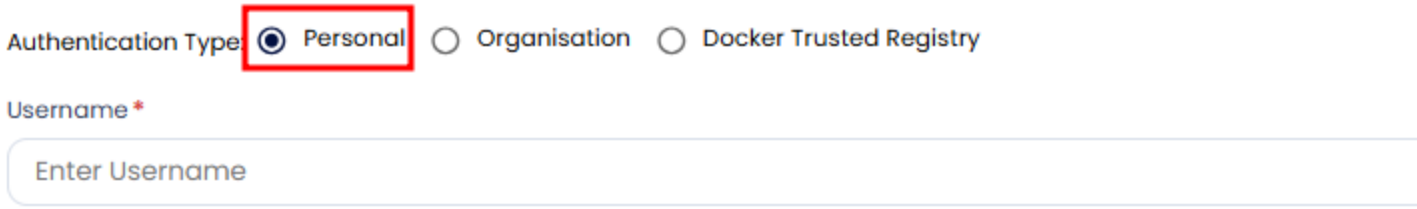
Dockerhub Registry Onboarding

Docker Hub is a cloud-based repository for storing, sharing, and managing Docker container images. It's like a library for container images, where you can find and download pre-built images or upload your own.

Prerequisites

Personal Account

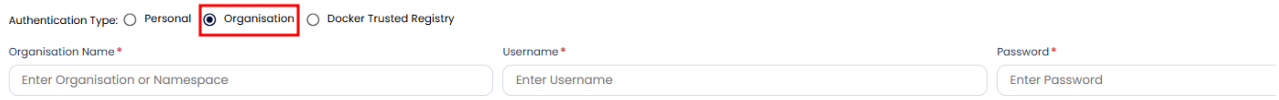
- **Requires:**
    - **Username**
    - **Password**
- **Explanation: A personal account is used by individual users who own or manage their own Docker Hub repositories. These credentials authenticate access to the user's personal space in Docker Hub.**



Organization Account

- **Requires:**
  - **Organization Name**
  - **Username**
  - **Password**
- **Explanation: An organization account is suitable for teams and enterprises managing shared Docker Hub repositories. It allows multiple users to collaborate under a unified organization while maintaining individual user roles and permissions.**

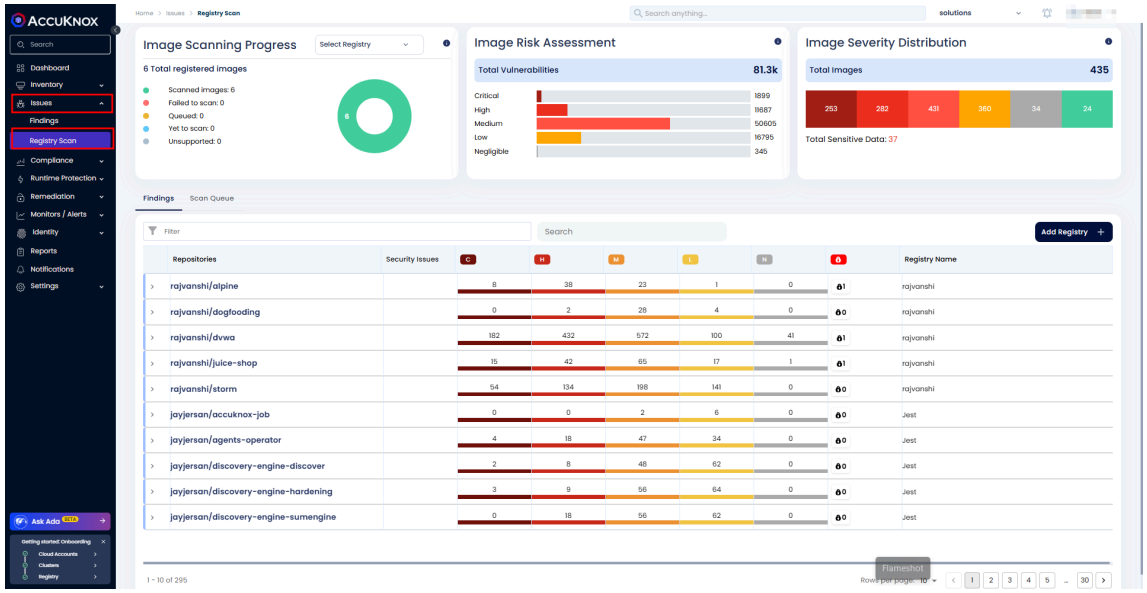
Note: Users must have pull permissions to access images stored in the enterprise repositories.



Steps to Add a Registry

1. Navigate to the Registry Scan Section

- **Go to Issues > Registry Scan.**

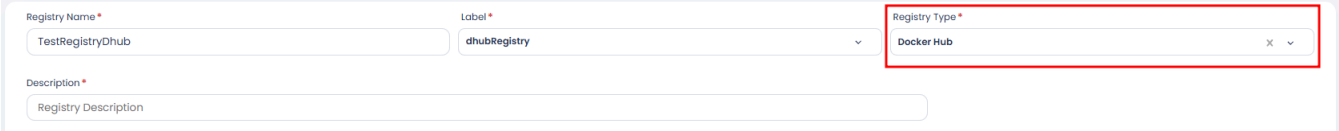


2. Add a New Registry

- **Click on Add Registry.**

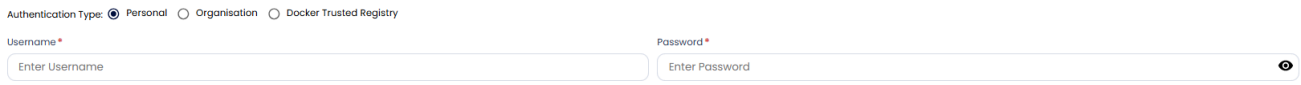
3. Provide Registry Details

- **Registry Name: Enter a name for your registry.**
- **Label: Add a label to associate findings to a particular label.**
- **Description: Provide additional information about the registry.**
- **Registry Type: Select Docker Hub from the dropdown menu.**

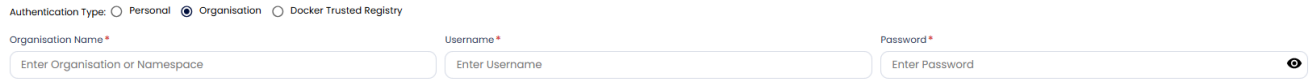


4. Authentication Type

- **Choose an appropriate authentication type based on your Docker Hub configuration:**
- **Personal: Requires your Docker Hub Username and Password.**



- **Organization: Requires your Organization Name, Username, and Password.**



5. Configure Advanced Settings

Image Updated Within Last

Choose one of the following options:

- **X Days: Scans only images updated within the last X days.**
- **All: Scans all images, regardless of the update time.**

Image Pulled Within Last

Choose one of the following options:

- **X Days: Scans only images pulled within the last X days.**
- **All: Scans all images, regardless of the pull time.**

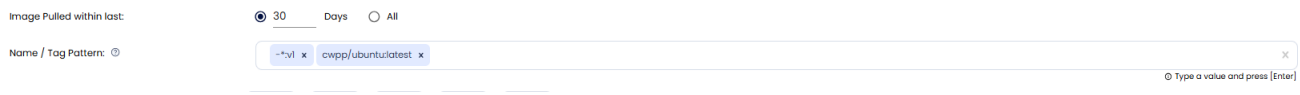
Name/Tag Pattern:

Specify patterns to include or exclude images for scanning. Use the `-` symbol to explicitly exclude patterns.

By default, images are excluded unless explicitly included through patterns.

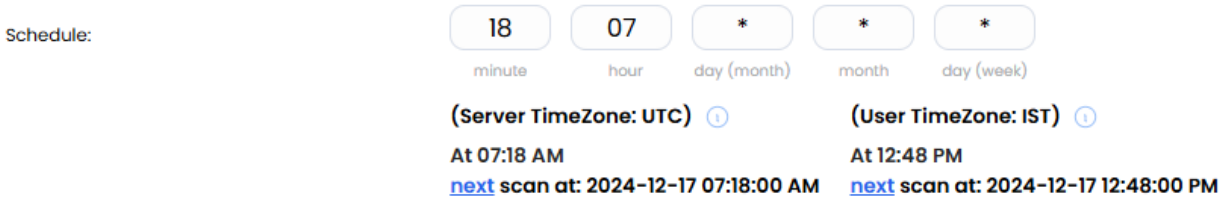
To exclude specific images, use the `-` symbol. For example: - To exclude `cwpp/ubuntu:v1`, use the pattern `-*:v1`. - To include `cwpp/ubuntu:latest`, specify a pattern like `*:latest`.

Note: Only images matching the pattern will be scanned. For instance, using `*:latest` ensures only images with the latest tags are scanned.



Schedule and Certificate

Set the scan schedule using a CRON expression. For example: - CRON Expression: `18 minute 07 hour * day (month) * month * day (week)`.



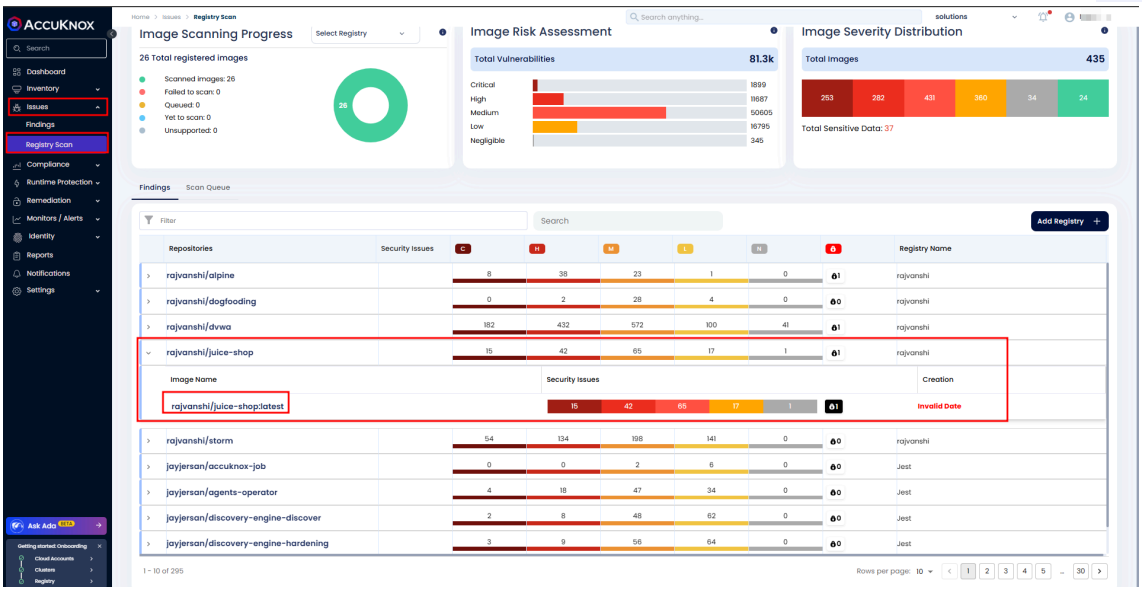
Toggle Trigger Scan on Save to directly initiate the scan for the first time without waiting for the scheduled time.

Viewing Registry Scan Details

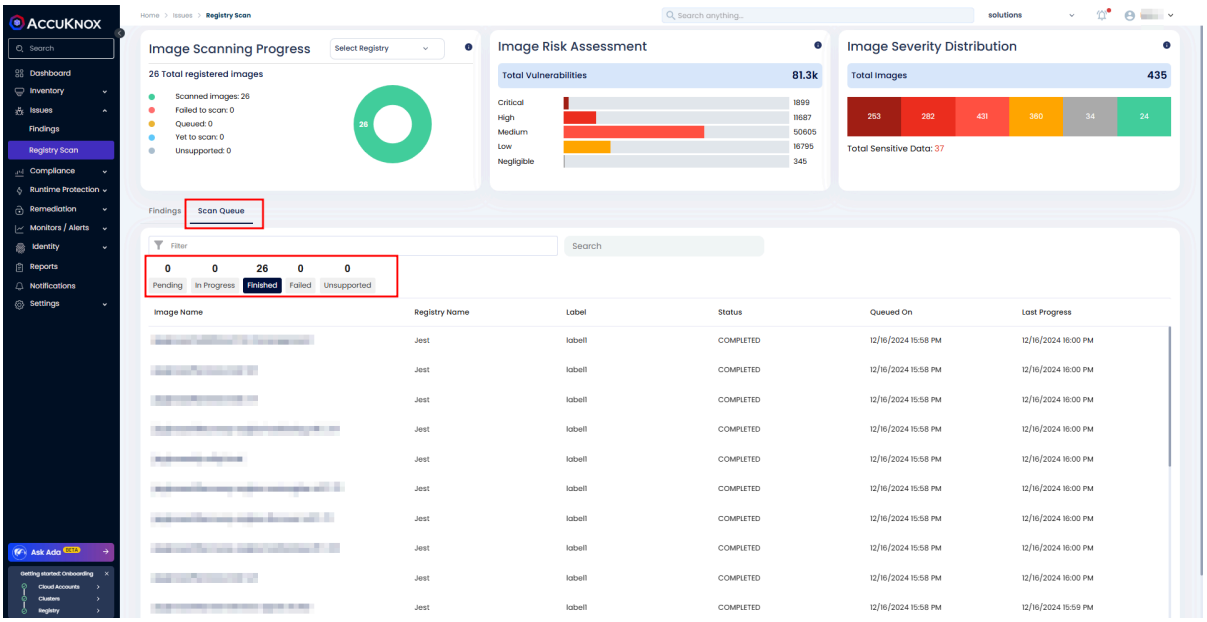
Once the configuration is complete, your registry is ready for scanning. Scans will occur based on the defined schedule and criteria. Ensure all advanced settings align with your organizational requirements for optimal results.

To view the scan results:

1. **Navigate to Issues > Registry Scan.**
2. **Find your repository to view the findings.**



3. **Alternatively, select Scan Queue to check the scan status.**



JFrog Container Registry Onboarding

JFrog Container Registry is a secure, universal repository manager specifically optimized for storing and managing container images. Widely adopted by DevOps and software teams, it supports Docker and Helm images, offering seamless integration with CI/CD pipelines to enhance workflows and ensure image security and traceability.

JFrog Artifactory offers two primary deployment options:

1. **Cloud-Based**: Managed by JFrog, offering scalability and minimal maintenance for teams preferring a ready-to-use solution.
2. **Self-Hosted**: On-premise for strict security needs, giving organizations control over configurations, with support for deployment in isolated networks.

AccuKnox Support for JFrog Container Registry Scanning

AccuKnox provides robust security scanning for container images stored in the JFrog Container Registry, regardless of deployment type. Supporting both cloud-based and self-hosted JFrog instances.

- **Cloud-Based JFrog Scanning**: For the JFrog Container Registry deployed in the cloud, AccuKnox connects seamlessly to scan images and detect vulnerabilities in real time.
- **Self-Hosted JFrog Scanning**: AccuKnox also supports self-hosted JFrog Container Registry deployments, providing vulnerability scanning for images in private, on-premise environments.
    - **Isolated Network Support**: AccuKnox can connect to self-hosted JFrog instances in isolated or air-gapped networks. This enables secure scanning in environments with strict compliance or network restrictions, ensuring continuous monitoring without compromising security.

The following steps outline how to onboard your JFrog Container Registry into the AccuKnox platform for ongoing security scanning, giving you real-time insights into vulnerabilities and risks within your container images.

Scanning an Isolated Registry

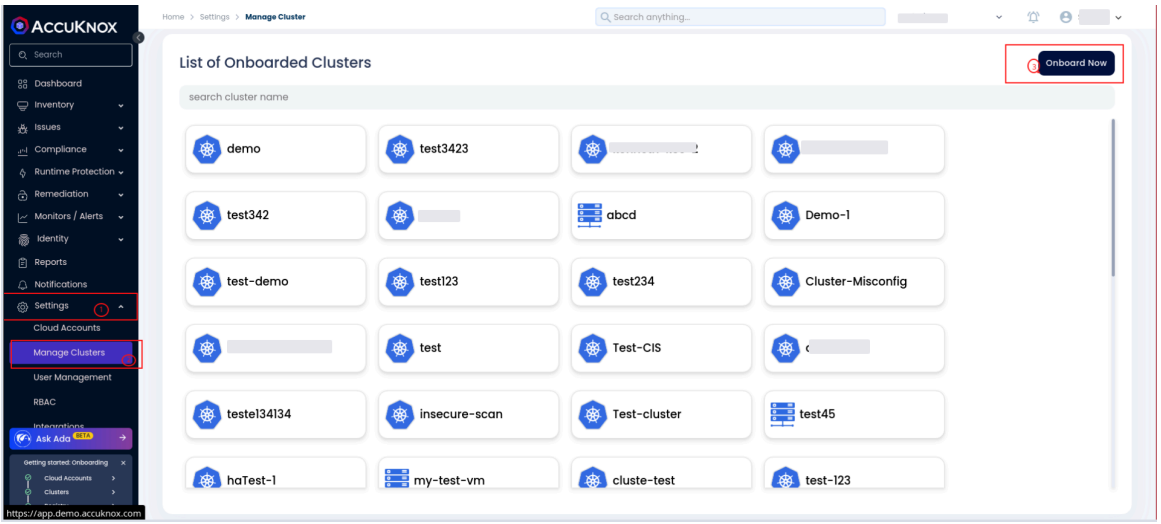
Important: If you're using a non-isolated JFrog Container Registry (cloud-based or non-isolated self-hosted), you can skip this section. This part applies **only** to **isolated JFrog instances**.

To get started with scanning a JFrog isolated container registry, ensure the following prerequisites are met:

1. Set up an **isolated JFrog container registry**.
2. Ensure you have access to a Kubernetes cluster where the **AccuKnox agents** can be onboarded.

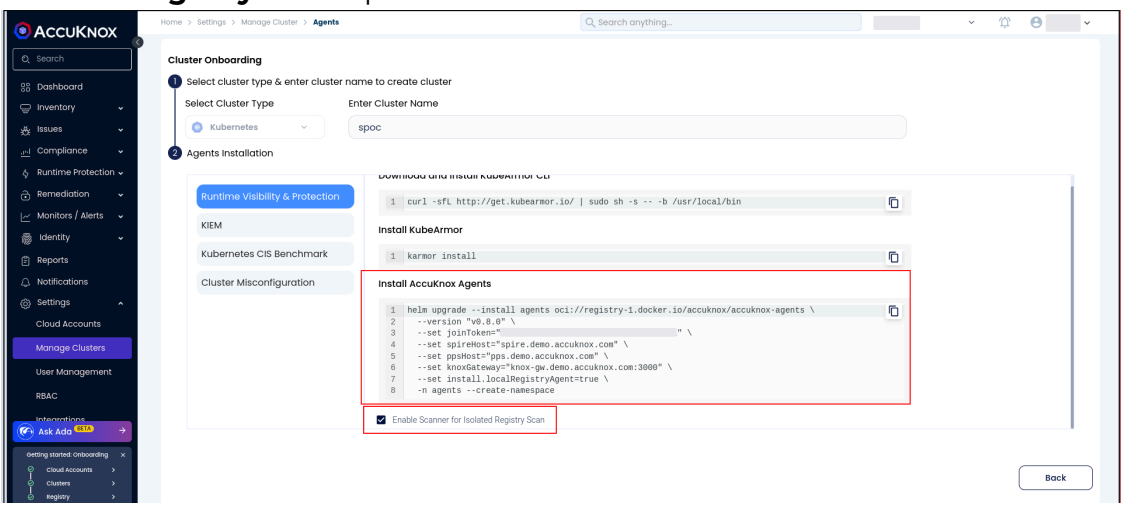
Once your registry is set up, the next step is to onboard the AccuKnox agents to your Kubernetes cluster.

1. Navigate to **Settings** > **Manage Cluster** in the AccuKnox platform.
2. Click **Onboard Now** to begin the process.



3. Provide an appropriate name for your cluster in the form that appears. During the agent installation process, ensure that the **Scanner for**

Isolated Registry Scan option is enabled.



4. Run the following Helm command to install the AccuKnox agents

```
helm upgrade --install agents oci://registry-1.docker.io/accuknox/accuknox-agents \
 --version "v0.8.0" \
 --set joinToken="<TOKEN>" \
 --set spireHost="spire.demo.accuknox.com" \
 --set ppsHost="pps.demo.accuknox.com" \
 --set knoxGateway="knox-gw.demo.accuknox.com:3000" \
 --set install.localRegistryAgent=true \
 -n agents --create-namespace
```

1. Verify the installation of the agents by running the following command:

```
kubectl get pods -n agents
```

```
➜  ~ kubectl get pods -n agents
NAME                                    READY   STATUS    RESTARTS   AGE
agents-operator-7645bccd5c-t5tx2        1/1     Running   0          63s
feeder-service-5f7b45884c-sbppz         1/1     Running   0          43s
local-registry-agent-7cb7484f5b-nkd8w   1/1     Running   0          43s
policy-enforcement-agent-544d59cf8-mrpkf 1/1    Running   0          42s
rabbitmq-755c547b88-tfpkp               1/1     Running   0          63s
shared-informer-agent-8589b8f6cf-82nwc  1/1     Running   0          42s
➜  ~
```

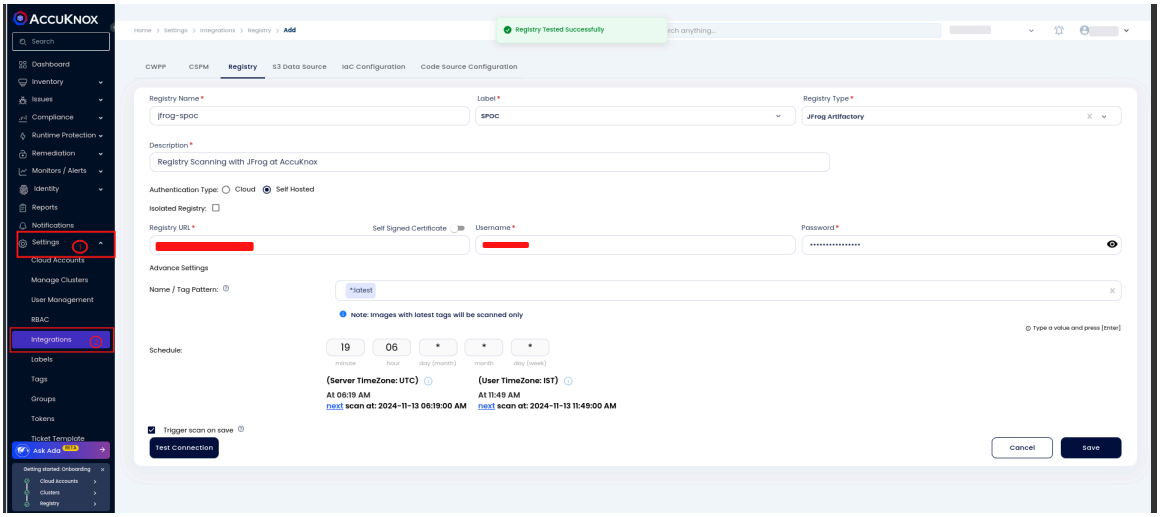
Once the agents are installed, navigate to the Cluster View in AccuKnox to ensure that your onboarded cluster is live and ready for scanning. This completes the onboarding process for scanning an isolated container registry in AccuKnox. The next step is to configure the registry scanning, as outlined in the previous sections.

Configuring the **JFrog** Registry

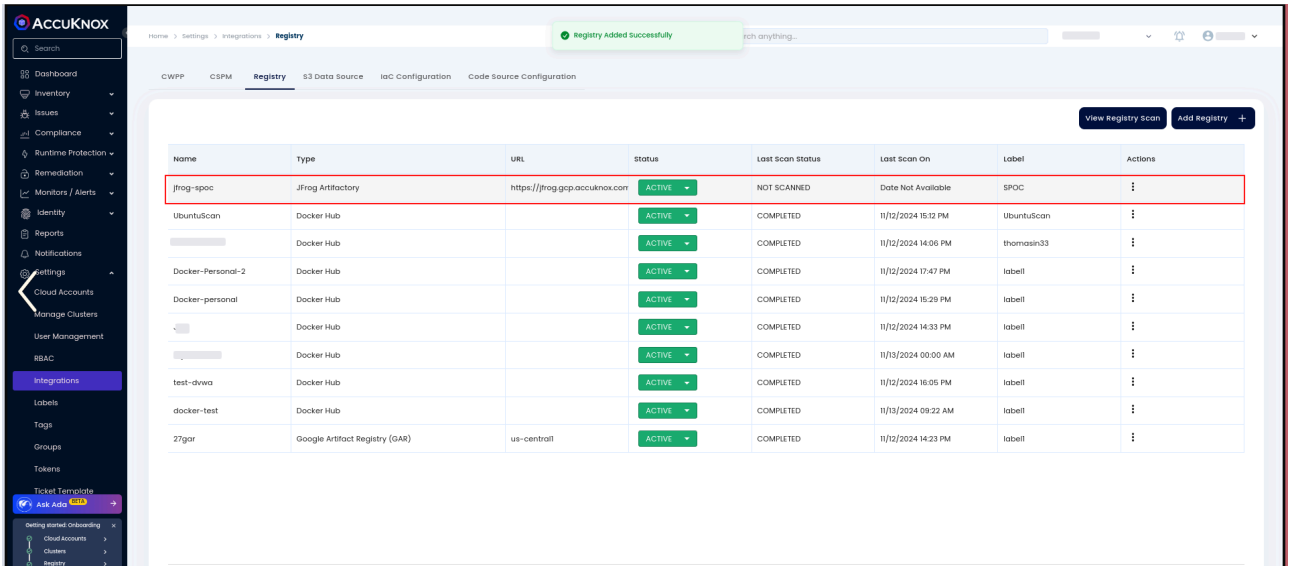
For this example, we'll proceed with **JFrog Self-hosted**.

Next, configure the self-hosted registry to begin scanning. Choose between **JFrog Cloud** or **Self-hosted**.

1. Go to **Settings** -> **Integration** -> **Registry**.
2. Click on the **Add Registry** button
3. Fill out the required fields such as:
    a. Name
    b. Description
    c. Registry Type
    d. URL
    e. Credentials
    f. Cron Expression (for scheduled scans)
4. If your JFrog Container Registry is in an isolated mode, ensure that the **Isolated Registry** flag is enabled in the onboarding form
5. Test the connection. If the configuration is correct, you will receive a successful response.



6. Once the connection is verified, save the form and create the registry. After the registry is configured and connected, it will appear as **Active** in the registry list.



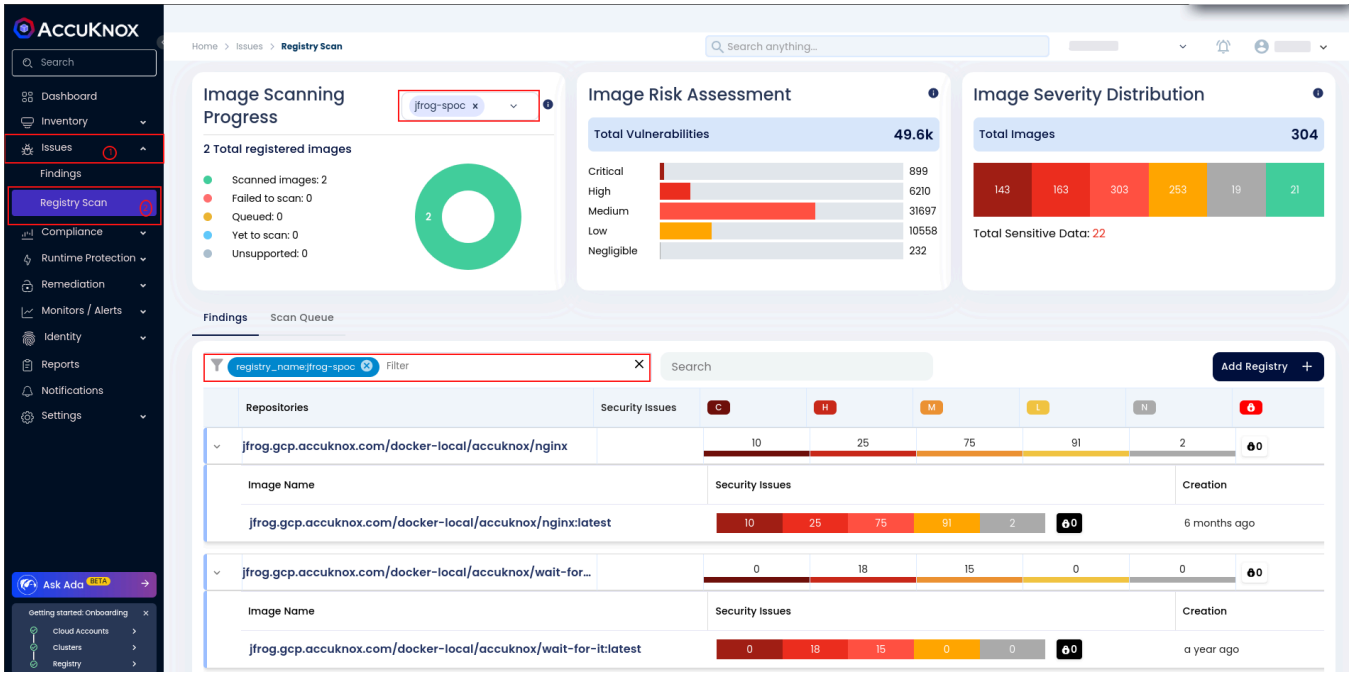
AccuKnox will begin scanning at the scheduled time specified during the configuration or If you've enabled the **Trigger scan on the save** option, the first scan will start immediately. Once the scan completes, navigate to the registry page to view the results.

Viewing Scan Details

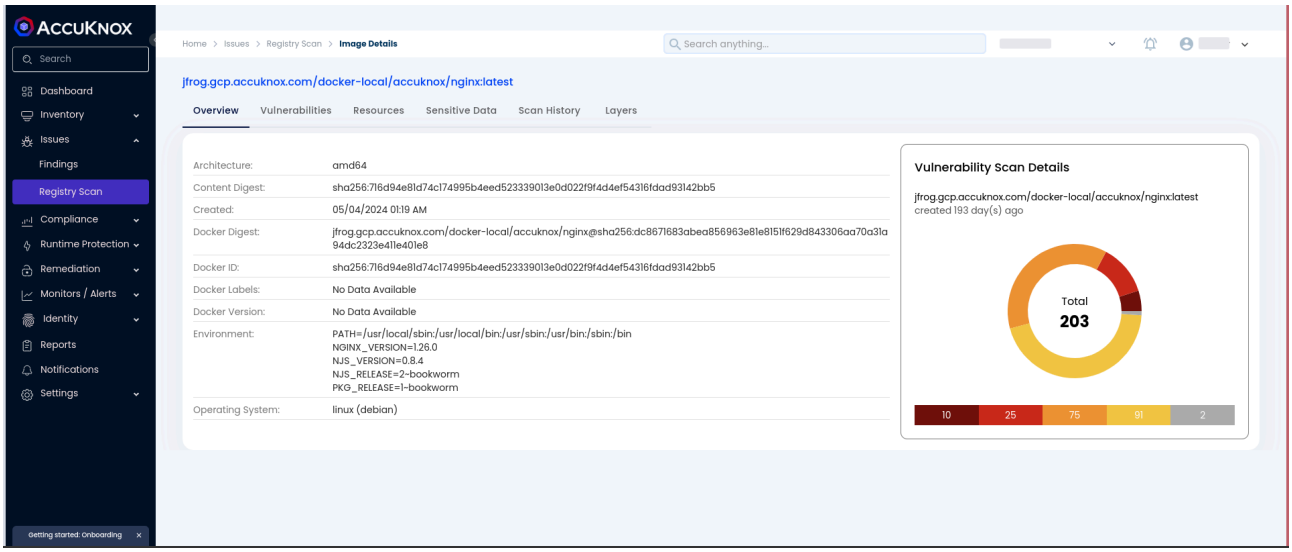
After the scan is completed, you can explore detailed information about the registry:

1. Go to **Issues** -> **Findings** -> **Registry Scan**.
2. Filter the results to view the onboarded registry.
3. Click on an image to see a detailed view of the metadata, vulnerabilities, and other scan details.

In the **JFrog Self-hosted Registry** that we onboarded to AccuKnox during this presentation, there is a specific package, **accuknox/nginx**. Below, you can see the associated vulnerabilities for this image, as highlighted in the following screenshots.



To get more detailed information about the vulnerabilities associated with the image, simply click on the container image in the AccuKnox dashboard. This will allow you to view the metadata, including any embedded secrets and a comprehensive list of the vulnerabilities identified in the image. You will also be able to explore the severity of these vulnerabilities, CVSS scores, and recommended remediation actions.



Integrating JFrog Container Registry with AccuKnox ensures continuous security scanning for container images, whether cloud-based or self-hosted. For isolated networks, AccuKnox provides secure, compliance-friendly scanning, helping you detect and address vulnerabilities efficiently.

CWPP Report Generation

Understand the Regex to Select the Cluster Name and Namespace

The CWPP report generation utilizes regular expressions (regex) to specify and filter cluster names and namespaces. The syntax for regex follows a particular pattern to ensure accurate selection.

Regex

Regex Syntax Format: Cluster Name Selection / Namespace Selection

Rules for Regular Expression

Excluding

- To exclude a specific cluster or namespace, prefix it with a hyphen (-).

NOTE

To exclude any cluster or namespace, it must be included in the selection first.

Select all

- Use an asterisk (*) to select all clusters or namespaces.

Delimiter

- A forward slash (/) is used to delimit the cluster name selection from the namespace selection.

Examples

- cluster1/ns1: Include only namespace ns1 from cluster cluster1.
- cluster1/*: Include all namespaces from cluster cluster1.
- cluster1/ns*: Include namespaces starting with ns from cluster cluster1.
- -cluster1/ns3: Exclude namespace ns3 from cluster cluster1.
- */ns1: Include namespace ns1 from all clusters.
- */*: Include all namespaces from all clusters.

Reports Configuration

Reports can be configured in two ways: On Demand and Scheduled.

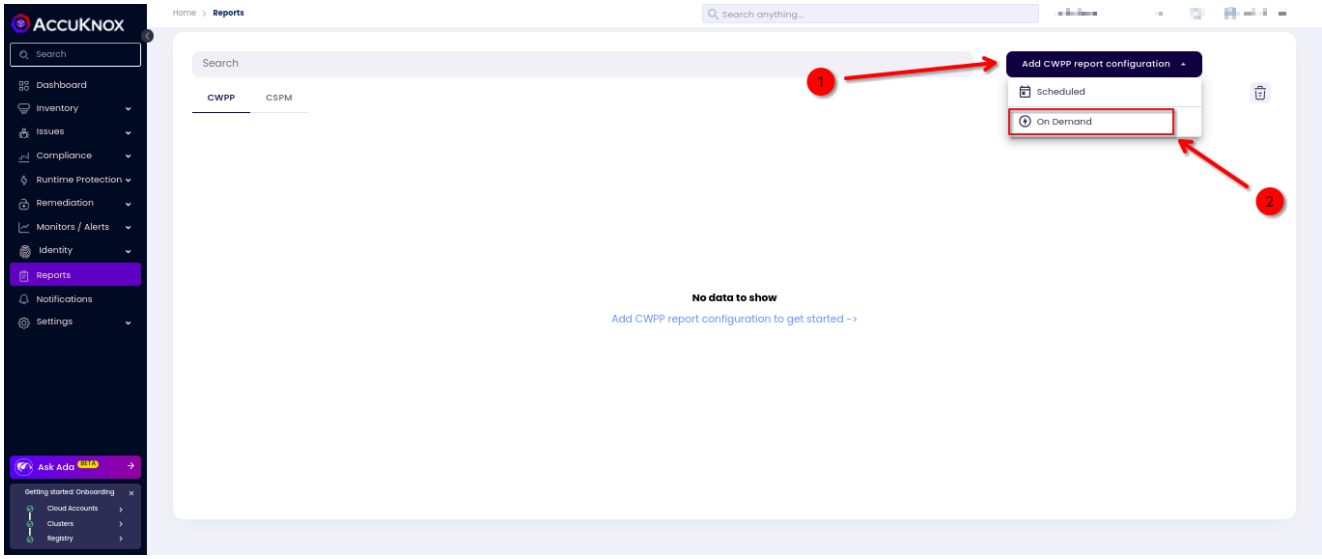
1. On Demand Report Configuration

In On Demand Report, you can generate the report for the clusters shortly after the configuration is completed.

To generate On Demand reports:

Step 1: Add CWPP Report Configuration

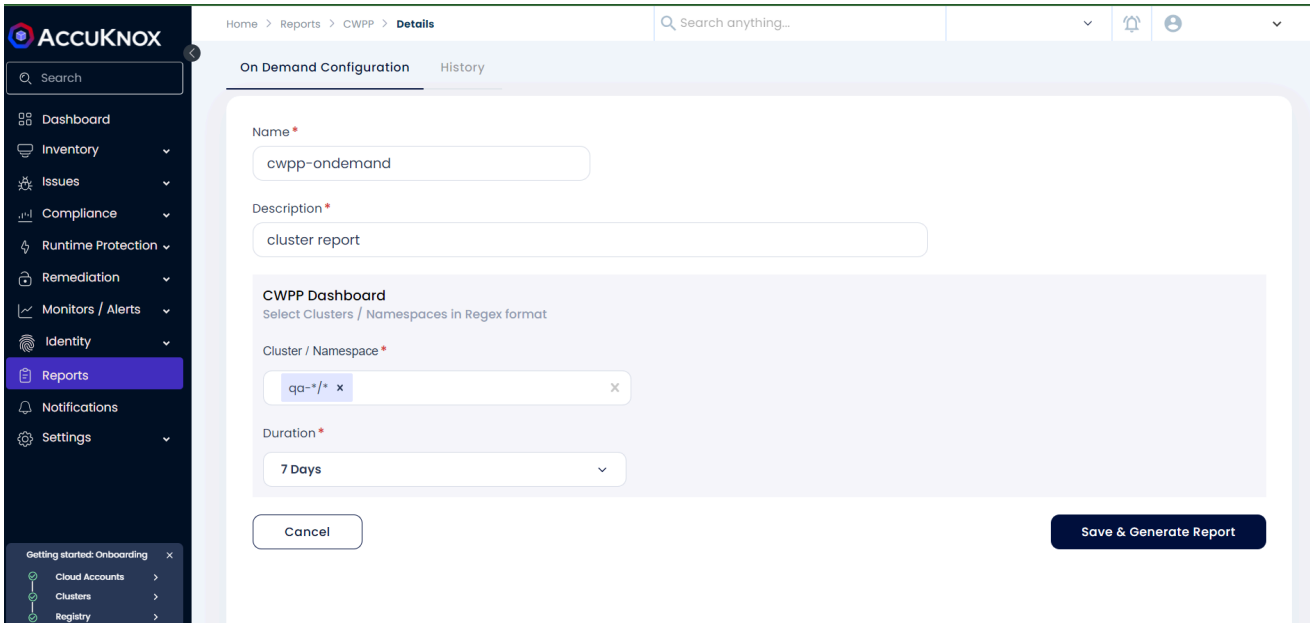
- Go to the Reports section in AccuKnox SaaS.
- Choose "On Demand" from the drop-down menu.



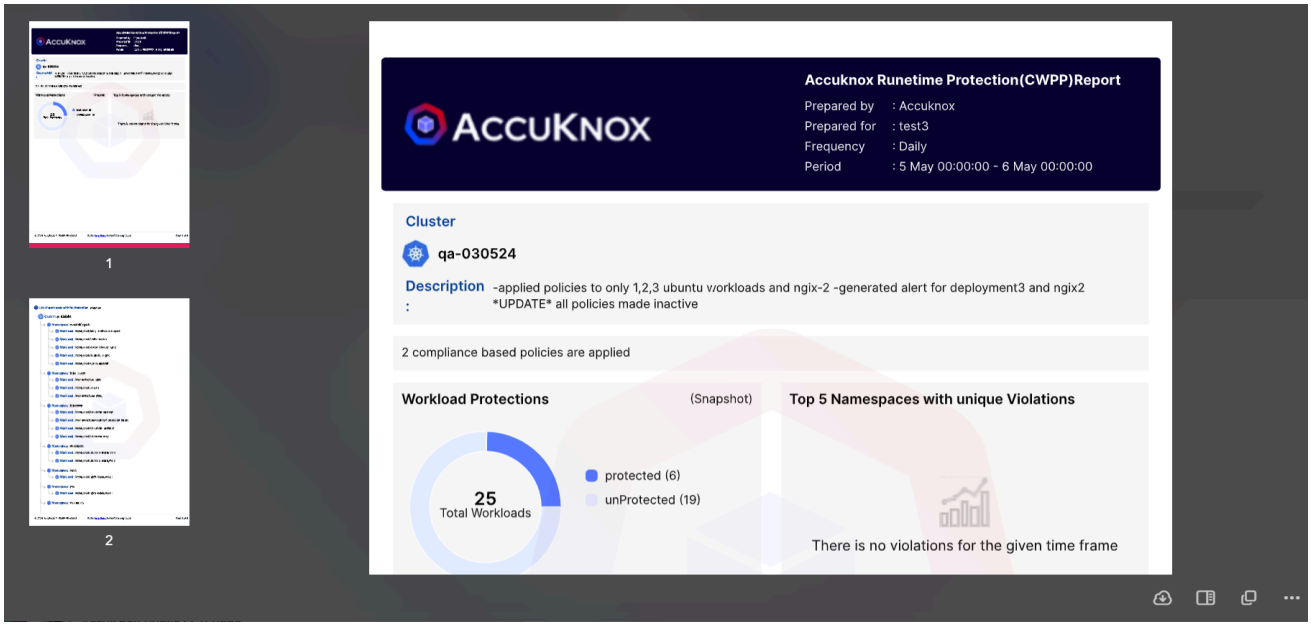
Step 2: In the Configuration user needs to provide the details about Name, Description and Cluster and NameSpace.

NOTE

The cluster field drop-down will show all the clusters that are active during the report generation.



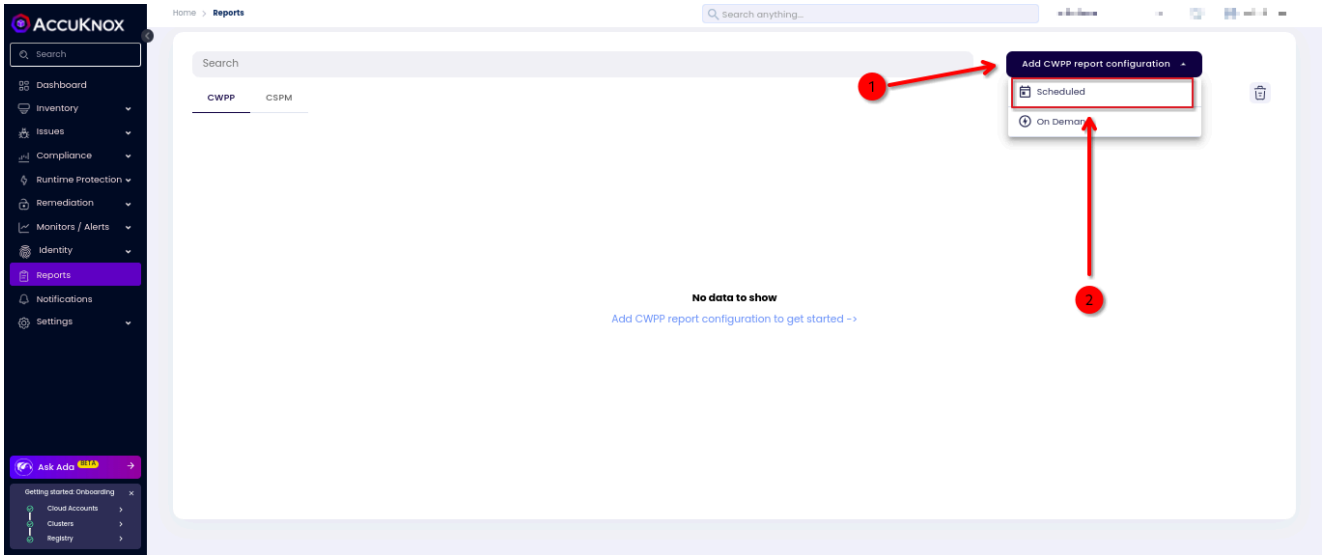
By clicking Save and Generate Report it will generate the report in the PDF format as per the selected duration.



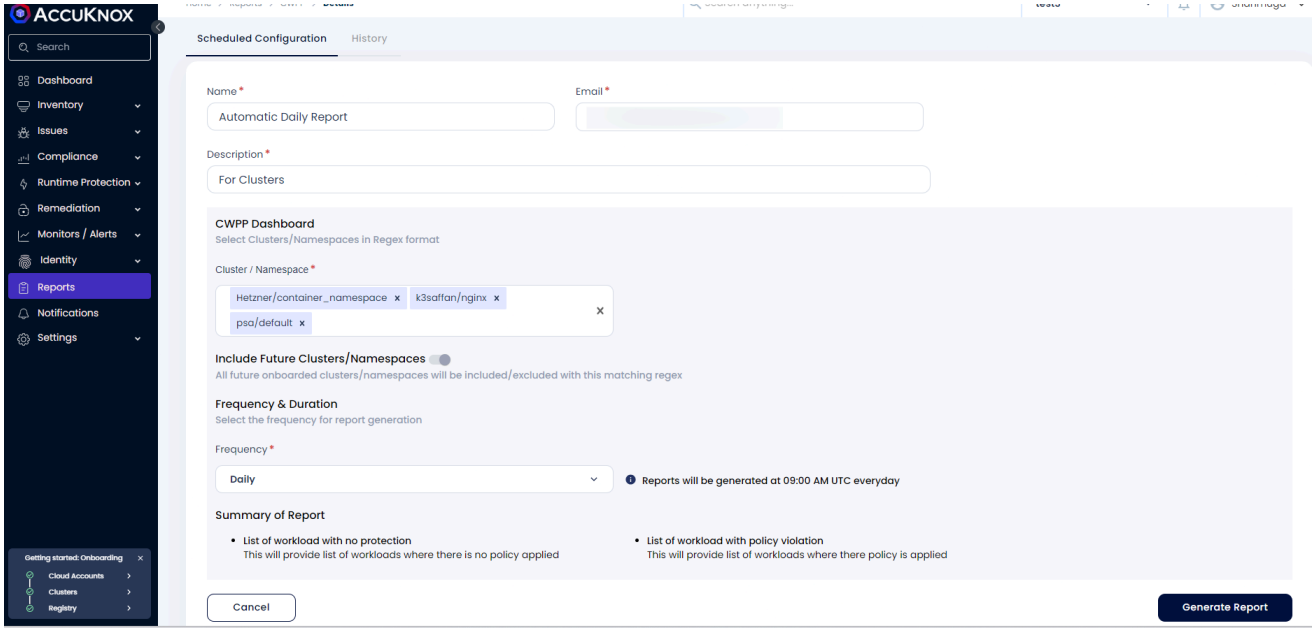
2. Scheduled Report Configuration

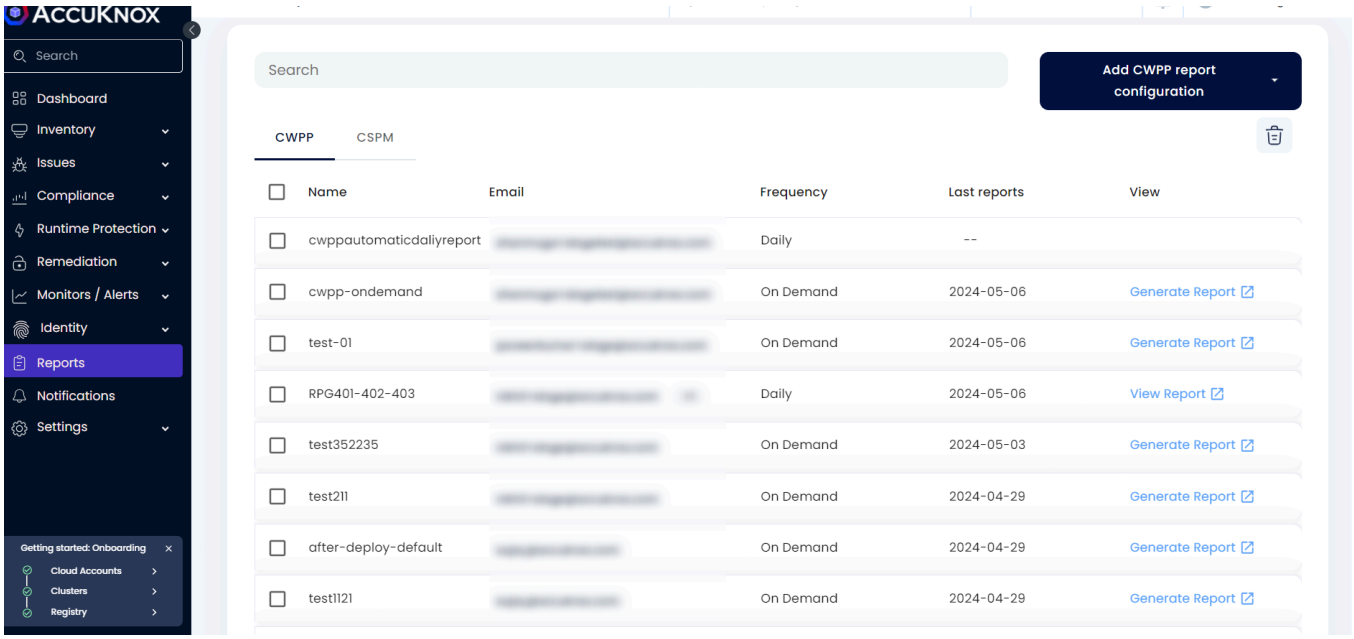
To get the report of the clusters automatically as per the frequency that choosen .i.e by weekly or by monthly or daily this is the go to way.

Step 1: To Add CWPP report configuration as Scheduled and choose the Scheduled option from the drop down.

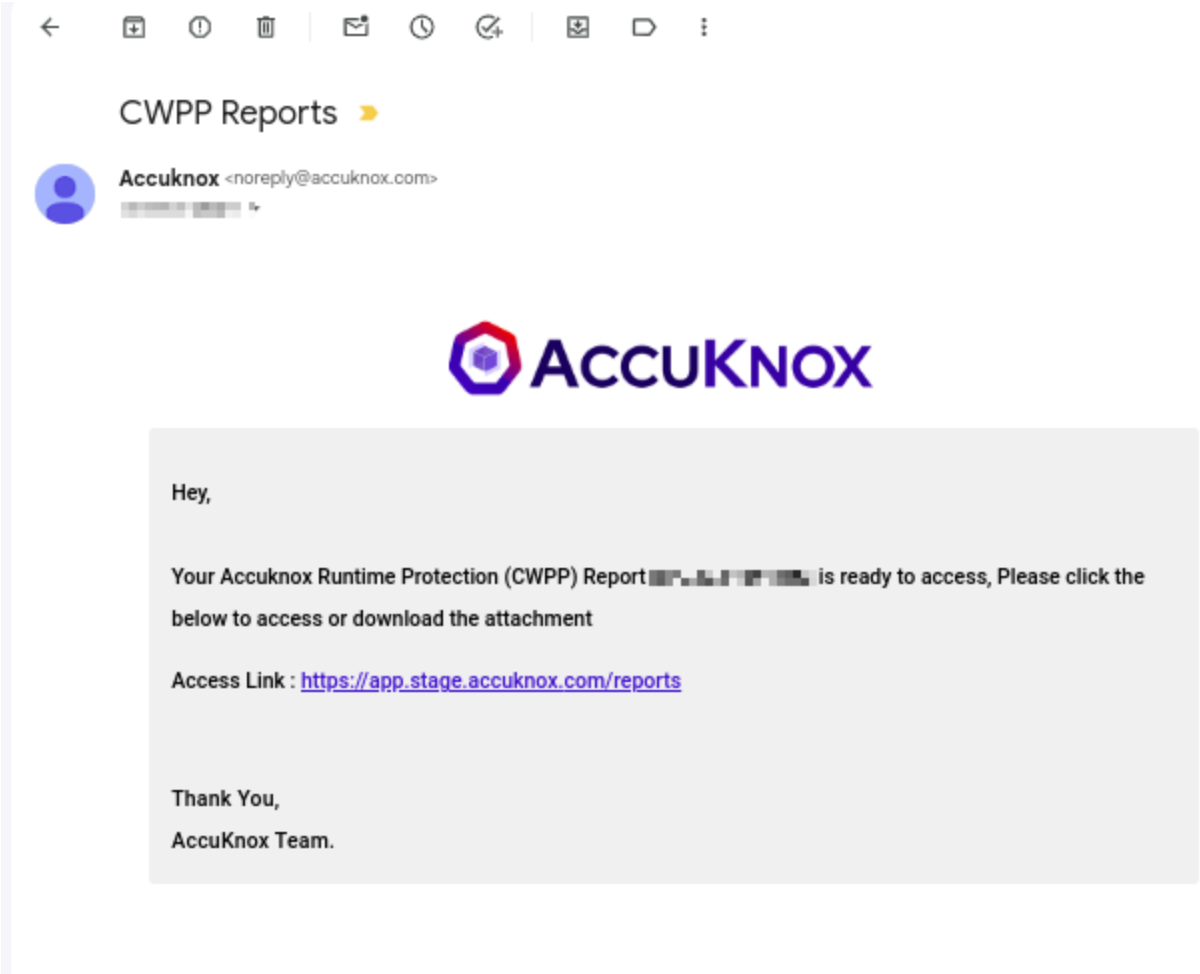


Step 2: In the Configuration user needs to provide the details about their Name, Email, Selecting the Cluster, Namespace in the regex format and Frequency of the report then click the Generate Report.





Step 3: After finishing the configuration the report would be scheduled to be sent to you in the email. Users can reconfigure the past configurations by clicking on them to edit the configuration.



CWPP Reports

Accuknox <noreply@accuknox.com>

ACCUKNOX

Hey,

Your Accuknox Runtime Protection (CWPP) Report [redacted] is ready to access, Please click the below to access or download the attachment

Access Link : https://app.stage.accuknox.com/reports

Thank You,
AccuKnox Team.

NOTE

The report will be sent to the Email-ID daily at 09.00AM UTC.

How to Configure Custom Reports

AccuKnox's latest feature update provides new custom reporting feature capabilities that can help users get the reports customized as per their requirements.

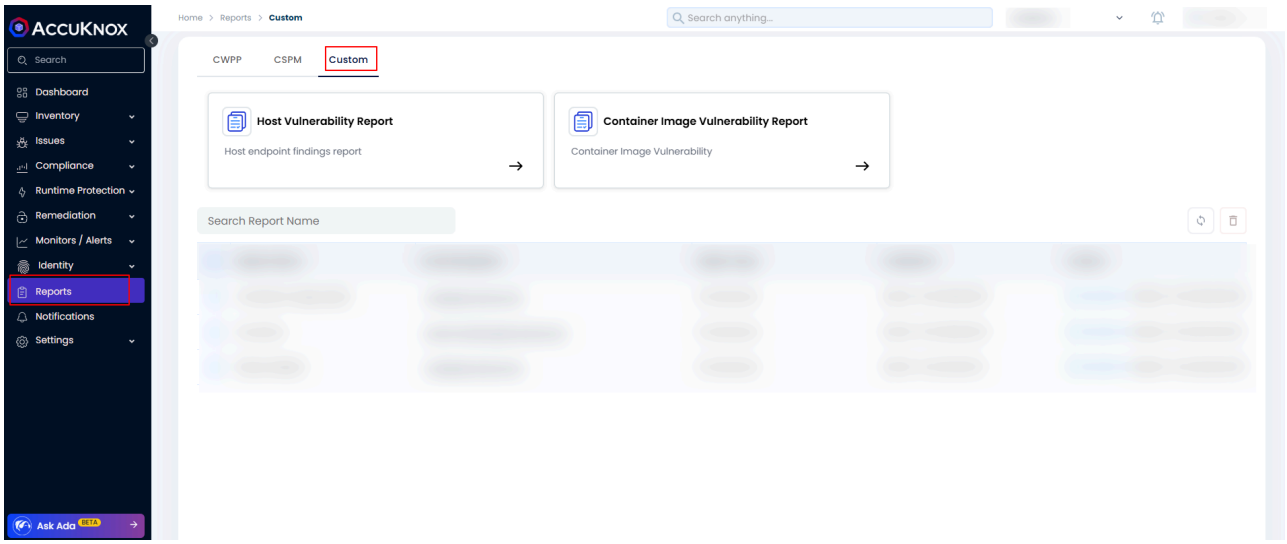
NOTE

For this feature to be enabled the customers need to inform the Support team(support@accuknox.com) regarding their requirements for custom reporting. Then the AccuKnox Support team can configure the report template from the backend. After which the users can generate an on-demand report or configure a scheduled report.

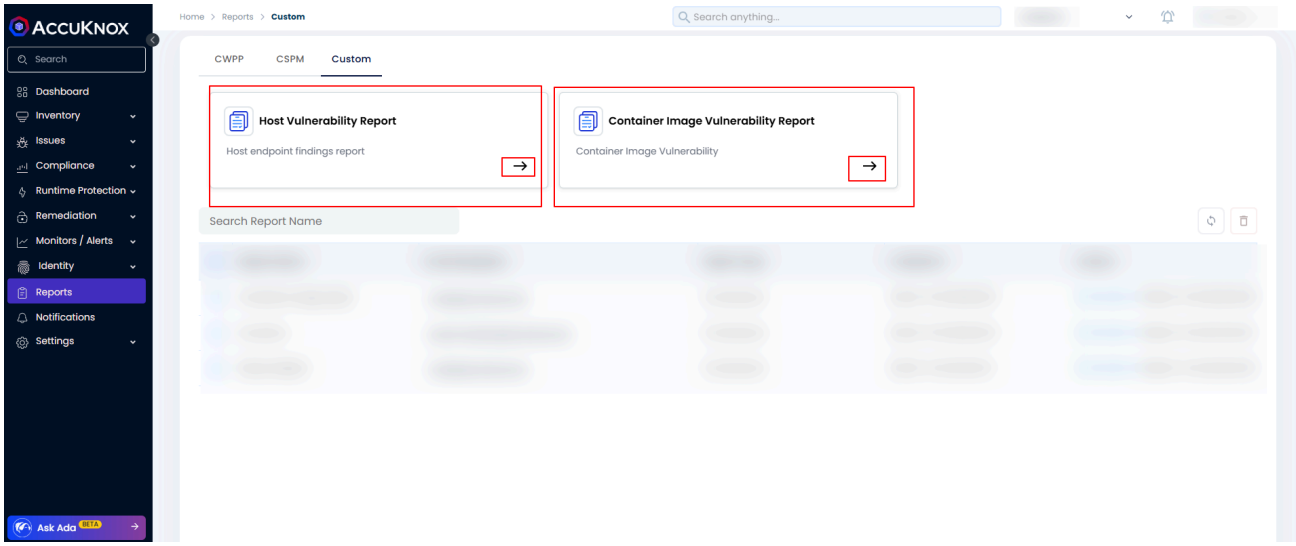
To generate an on-demand or scheduled report, users must follow the steps below.

On-demand custom Report generation

Step 1: Users will need to navigate to the Reports->Custom Reports Section.

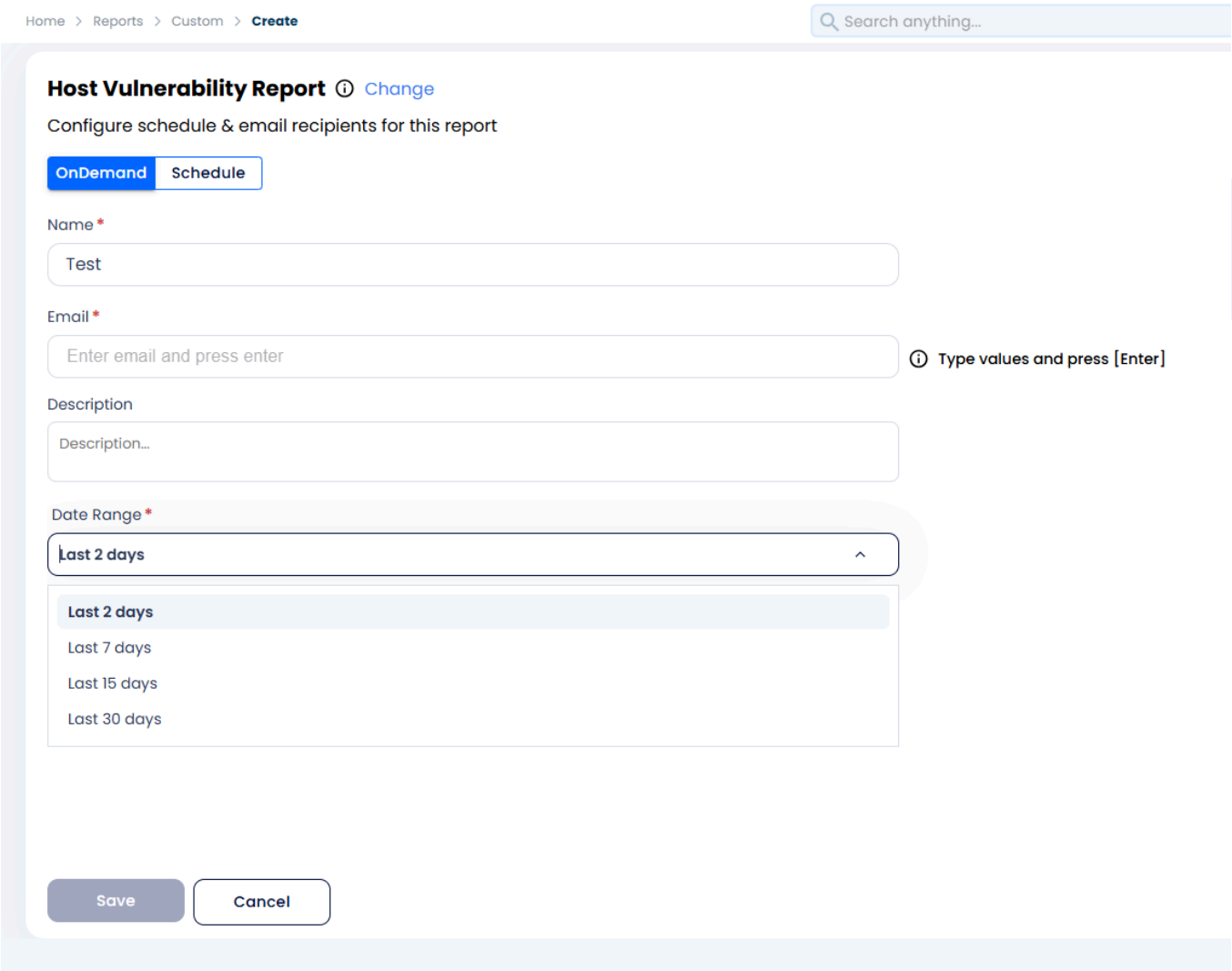


Step 2: Now the users will need to select any one report which they want to configure from the customized reports that are shown in the UI.

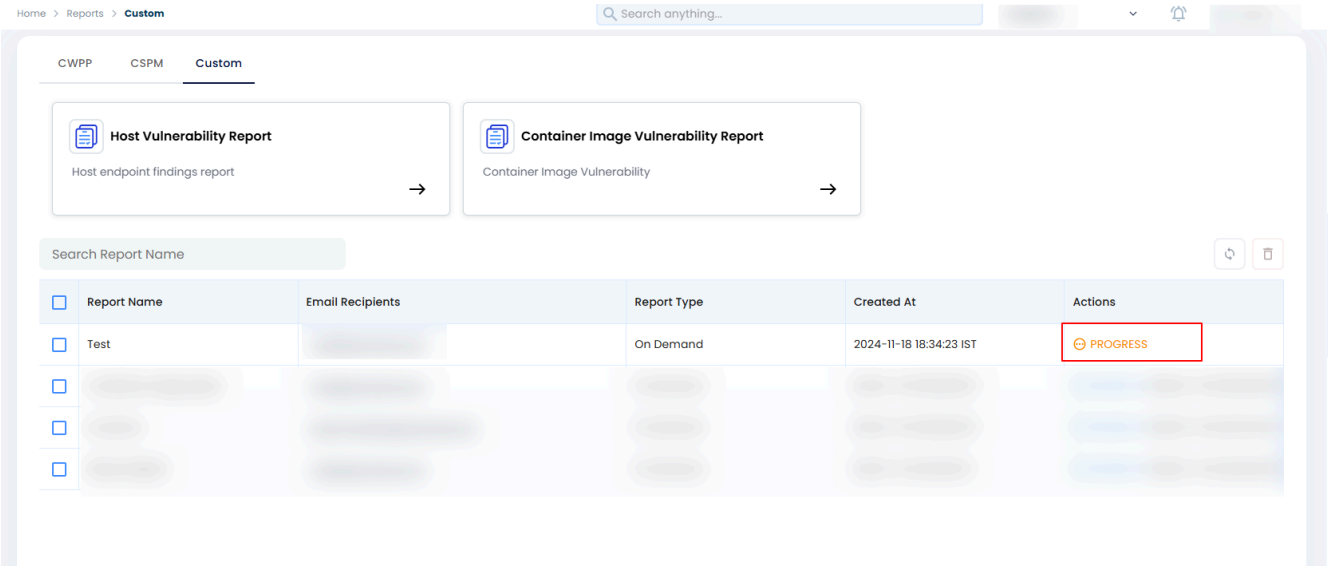


Step 3: Users can configure the report as a scheduled report or generate it as an on-demand one. Users can select any one option and fill out the necessary details. Like if it is an on-demand report the users will need to fill in the following fields

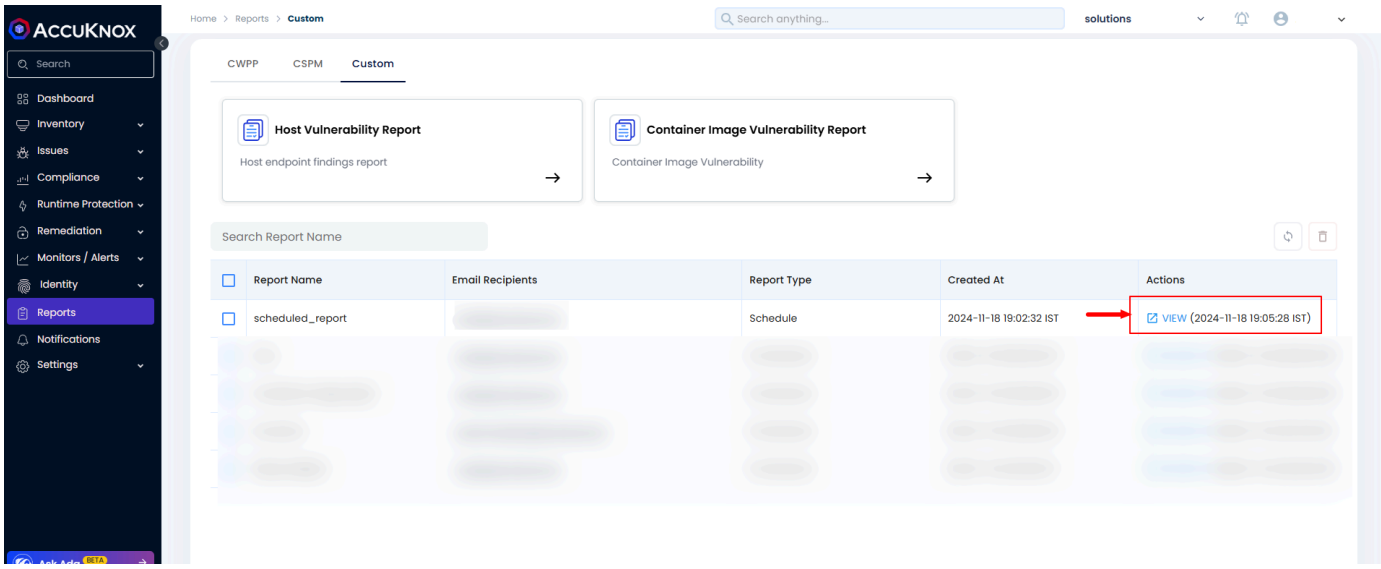
Like the report name, an email address where the report needs to be sent, and the duration for which the report needs to be generated from the drop-down list options shown in the UI. After filling out these options the save button will be enabled and users can save it.



Step 4: Once the on-demand report is saved the users can see the report in the UI with the progress state mentioned

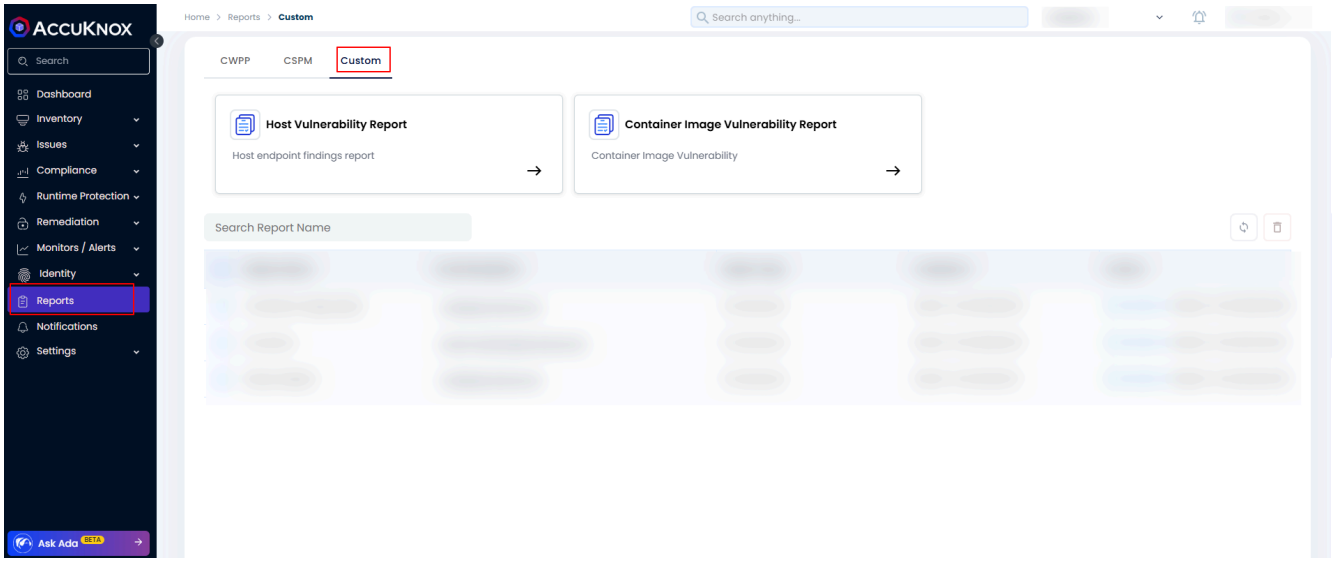


Step 5: After the report generation is completed you can see the Generate option in the UI as well as the report will be mailed to the email address. If the user wants to see the report in the UI they can click on the Generate report.

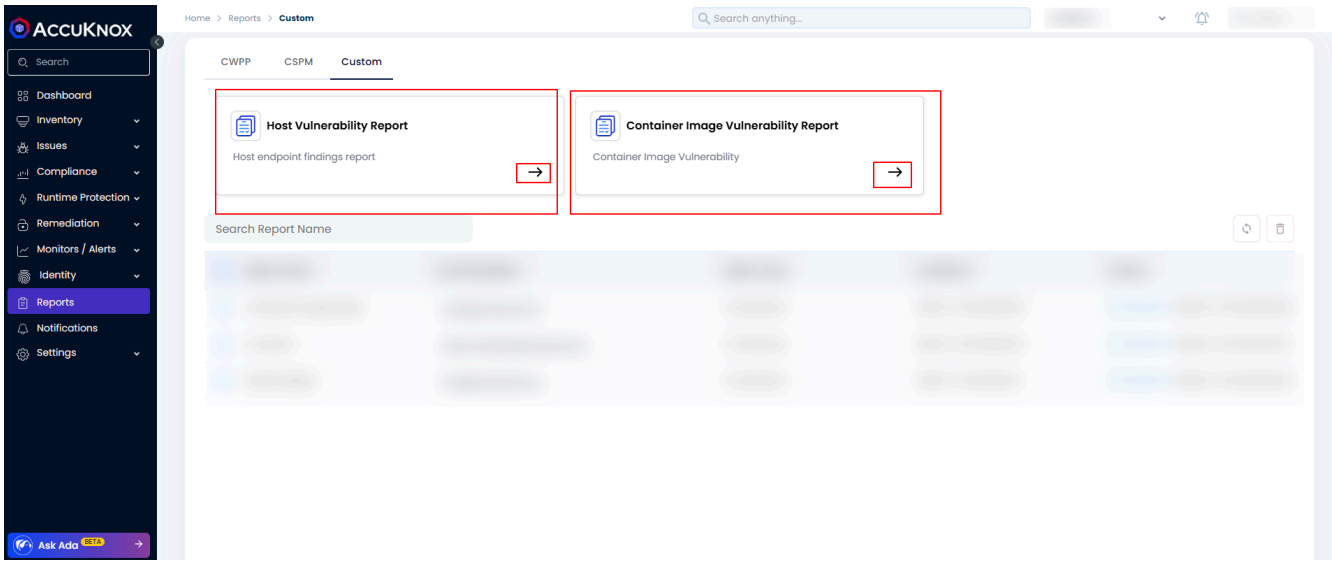


Scheduling Custom Report

Step 1: Users will need to navigate to the Reports->Custom Reports Section.

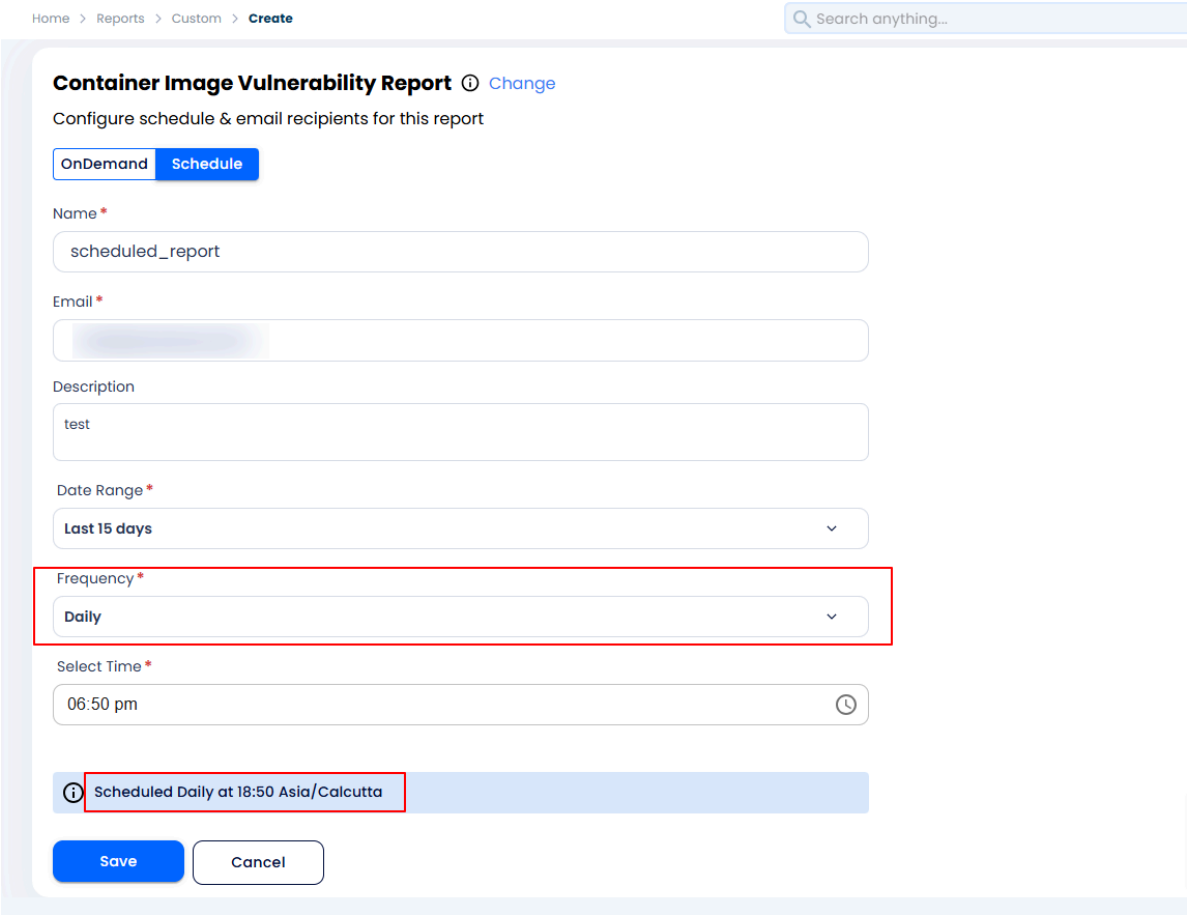


Step 2: Now the users will need to select any one report which they want to configure from the customized reports that are shown in the UI.

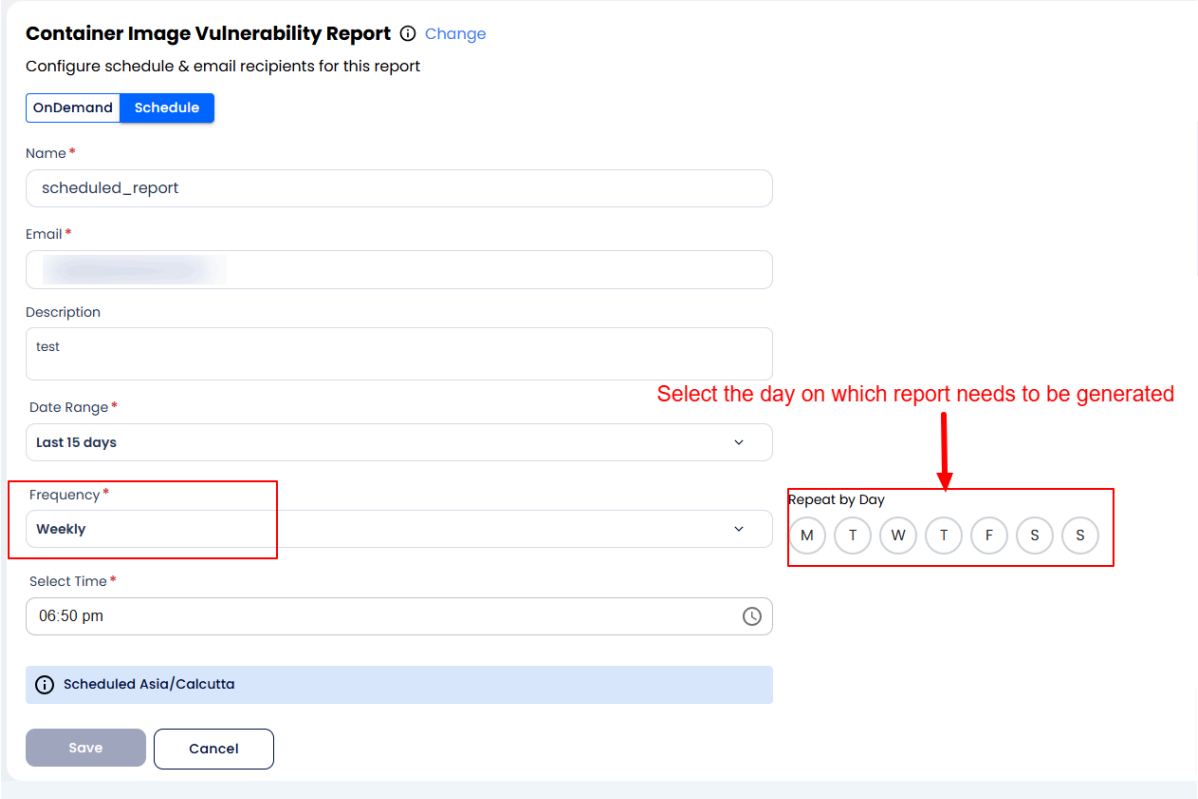


Step 3: Now the users will have the option to configure the report as a scheduled report or generate it as an on-demand one. Users can select any one option and fill out the necessary details. If the users want to schedule a custom report then they will have to fill out the following details like name, duration, and scheduling frequency. AccuKnox provides 3 scheduling frequency options.

1. Daily Report: users can select the frequency as daily to receive the report every day at the configured time.

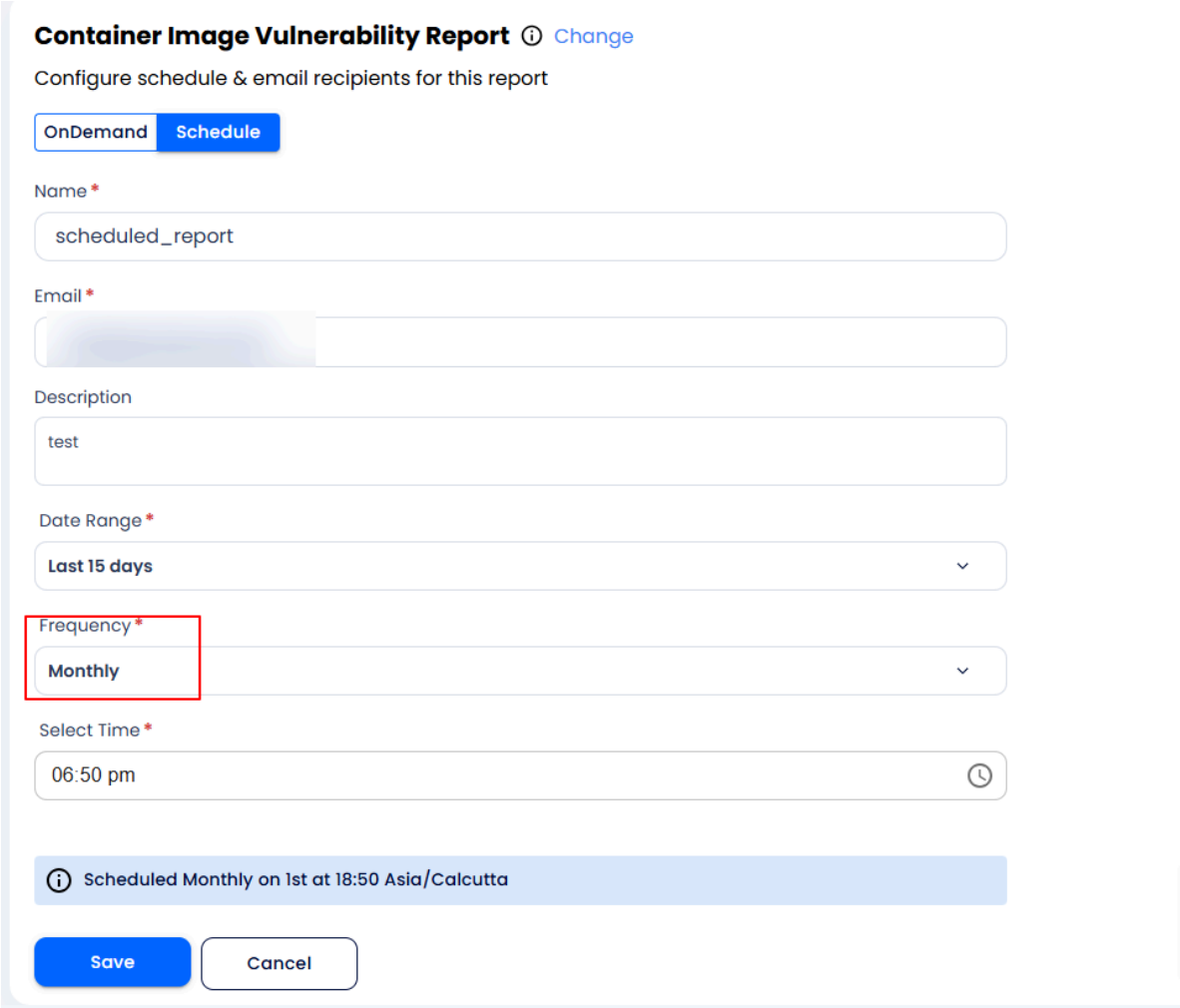


2. Weekly: Users can also schedule the report weekly and select the day on a week when the report needs to be generated.

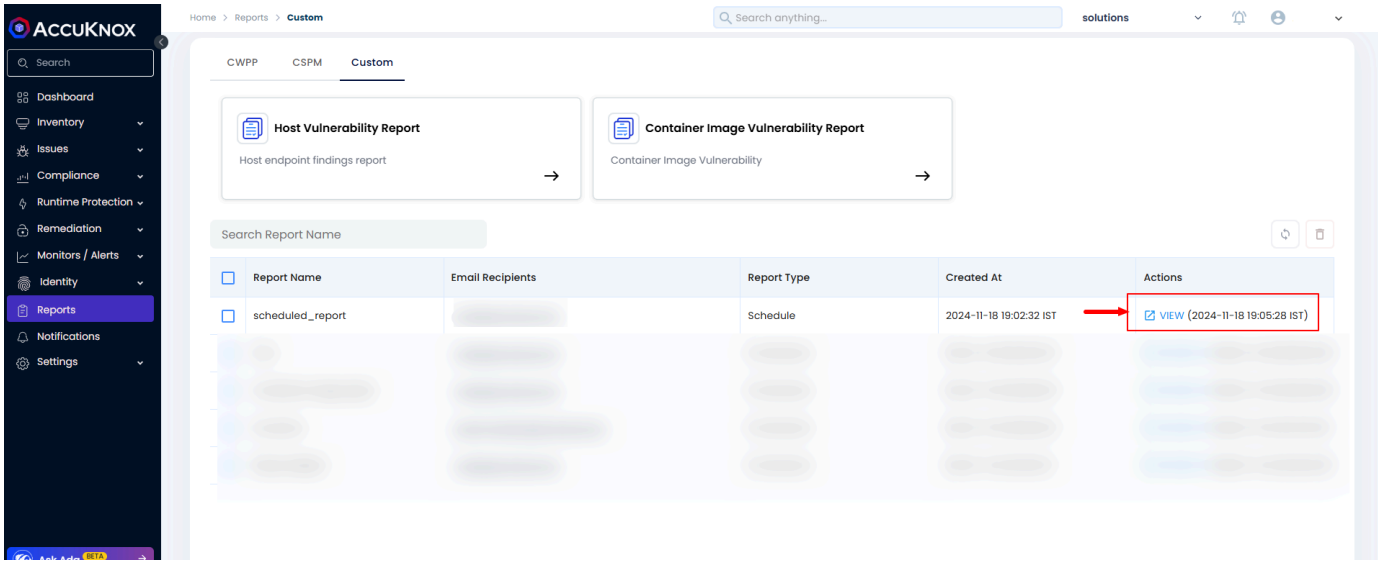


3. Monthly: Users can also configure the report duration as monthly where they will be getting the report on the 1st of every month. It will soon be

configurable as the user-defined date as well.



Step 4: Once the report generation is completed you can see the View option in the UI as well as the report will be mailed to the email address. If the user wants to see the report in the UI they can click on the View.



RINC

RINC (short for "Reporting IN Cluster") is a simple and lightweight reporting tool that provides insights into the status of a Kubernetes cluster, as well as other services running within it.

It includes built-in alerting capabilities, allowing users to define alerts using an expression language. RINC comes with a set of practical and sensible pre-configured alerts, which are included in the provided Helm charts. If you need to customize or extend these alerts, you can easily do so using our expression language, which is powered by the gval Go library.

RINC also supports email integration, allowing you to receive alerts via email.

Supported reports

- Kubernetes deployment and statefulset status reports
- Long-running job reports
- Registry scan job status reports
- Supports reporting jobs where the module container has succeeded but the artifact-api container has failed.
- Kubernetes deployment and statefulset image tag reports

- RabbitMQ metrics reports
- CEPH metrics reports
- Pod status reports
- PV Utilization report
- Pod & Node resource utilization report
- Token expiry report
- Nodes' time-in-sync report
- Connectivity & Status checks for,
- Vault
- MongoDB
- Redis/KeyDB
- Neo4j
- Postgresql
- Prometheus
- Metabase
- AWS RDS
- Weaviate
- Onboarded registries status report
- Kueue workload status report
- Supports reporting jobs where the module container has succeeded but the artifact-api container has failed.

Installation

We recommend installing RINC through our provided helm charts.

Note: RINC uses MongoDB as its data store and creates a new collection called "rinc" upon launch. It is recommended that you create a separate MongoDB user with R/W access to the "rinc" collection. See the section on Minimum Required Database Permissions.

```
VERSION=0.9.0

helm show values oci://public.ecr.aws/k9v9d5v2/accuknox-rinc --version "$VERSION" > values.yaml
```

The file `values.yaml` is well-documented and includes all configurable options for `RINC`. Please go through it and adjust the values as needed to suit your preferences. See passing database/vault credentials to RINC.

By default, all reports are disabled and can be enabled by setting `enable` to `true` in the Helm chart values. For example, to enable the RabbitMQ report, set:

```yaml
config:
  rabbitmq:
    enable: true
```

If you are using our Accuknox Helm charts, we provide an `accuknox-values.yaml` file with most of the values pre-configured.

```
helm pull oci://public.ecr.aws/k9v9d5v2/accuknox-rinc --version "$VERSION"
tar xvzf "accuknox-rinc-$VERSION.tgz"
less accuknox-rinc/accuknox-values.yaml
```

RINC supports reading secrets directly from Vault. If you are using Hashicorp's Vault, please refer to the section on vault.

After customizing the values to your preferences, run the Helm install command below to deploy `RINC` in your cluster:

```
NAMESPACE="accuknox-rinc"

helm upgrade rinc oci://public.ecr.aws/k9v9d5v2/accuknox-rinc \
  --install \
  --namespace "$NAMESPACE" \
  --create-namespace \
  --version "$VERSION" \
  --values values.yaml
```

To check if everything is healthy, run:

```
watch kubectl -n "$NAMESPACE" get pod,job,cronjob,secret,configmap
```

If everything appears healthy and running, congratulations! RINC has been successfully installed on your cluster.

Passing Database Credentials

Database credentials are used for connectivity checks. There are 3 ways to pass your database credentials to RINC,

1. **Using Helm**:

Set `secretConfig.create` to true in the helm values and fill the secrets below to let Helm create a Kubernetes Secret that is mounted into RINC.

```yaml
secretConfig:
  create: true
  config:
    mongodb:
      ###        ###
      ### REDACTED ###
      ###        ###
```

2. **Manually Creating a Secret**:

Below is a template for the Secret manifest,

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: credentials
  namespace: accuknox-rinc
type: Opaque
stringData:
  secret.yaml: |-
    # Please fill in the configuration below if you have set `vault.use` to
    # true above.
    vault:
      auth:
        # vault auth type
        #
        # Possible values: "token", "kubernetes"
        type: ""
        # Token used to authenticate to vault. Required when auth type is set to
        # "token".
        token: ""
```

```yaml
    # Role name used to authenticate to vault. Required when auth type is set
    # to "kubernetes".
    role: ""
  # Service-specify credentials.
  #
  # It is recommended to create a dedicated `rinc` user for each of the
  # services.
  mongodb:
    username: ""
    password: ""
  email:
    smtp:
      host: ""
      username: ""
      password: ""
      port: 587
  rabbitmq:
    management:
      # basic auth username for the management api.
      username: ""
      # basic auth password for the management api.
      password: ""
  ceph:
    # ceph reporter uses ceph's dashboard API to scrape ceph status and
    # metrics.
    dashboardAPI:
      # username to authenticate with ceph dashboard API.
      username: ""
      # password to authenticate with ceph dashboard API.
      password: ""
  connectivity:
    neo4j:
      # neo4j basic auth username
      username: ""
      # neo4j basic auth password
      password: ""
    postgres:
      # postgresql auth username.
      username: ""
      # postgresql auth password.
      password: ""
    rds:
      # aws access key id
      accessKeyId: ""
      # aws secret access key
      secretAccessKey: ""
  tokenExpiry:
    # list of token whose expiry need to be checked.
    #
    # It is recommended to NOT specify the token value here as it will remain
    # static. If you are using Vault, you can specify the vault `path` as
    # documented in the `config` section. If you are NOT using Vault, you can
    # use ExternalSecrets that will periodically sync the token value.
```

```
    tokens: []
      # - name: ""
      #   value: ""
  cloudScan:
    onboardedRegistries:
      postgres:
        # postgresql auth username.
        username: ""
        # postgresql auth password.
        password: ""
```

```
kubectl apply -f credentials.yaml
```

This secret must then be referenced in the helm chart values,

```
# This section is for specifying an existing Kubernetes Secret that the Helm
# chart should reference
existingSecret:
  # name of the existing Secret in the Kubernetes cluster
  name: "credentials"
  # key within the Secret, which corresponds to the specific value to be used.
  key: "secret.yaml"
```

3. **Reading credentials directly from Vault**

RINC can read credentials directly from Vault. To configure RINC to connect to Vault, specify the connection details in the Helm values under `secretConfig.config.vault` and ensure that `secretConfig.create` is set to `true`. Helm will pass the Vault credentials to RINC via the created Kubernetes Secret, allowing RINC to use these credentials to connect to Vault and read the remaining credentials directly from it.

See the section on Vault for setting up the required Vault policies.

```
secretConfig:
  create: true
  config:
    # Please fill in the configuration below if you have set `vault.use` to
    # true above.
    vault:
      auth:
        # vault auth type
        #
        # Possible values: "token", "kubernetes"
        type: ""
        # Token used to authenticate to vault. Required when auth type is set to
```

```
    # "token".
    token: ""
    # Role name used to authenticate to vault. Required when auth type is set
    # to "kubernetes".
    role: ""
```

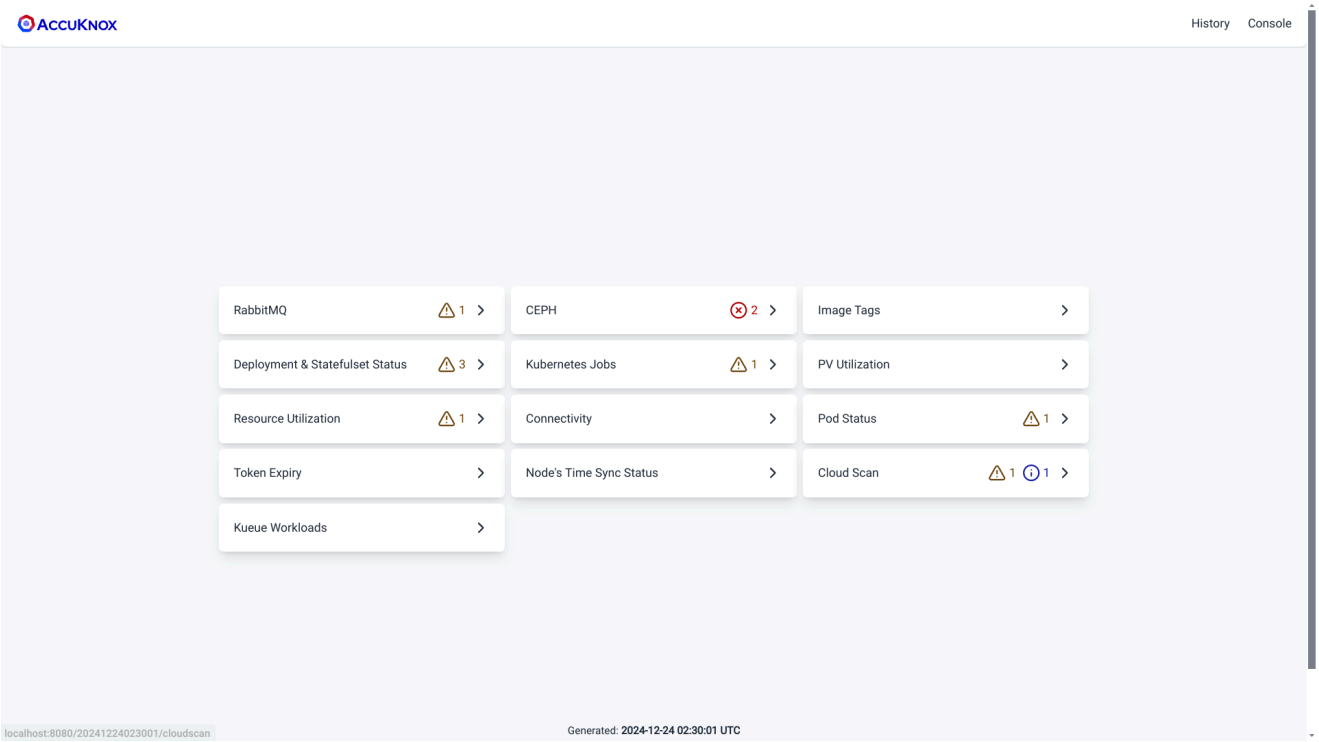
Accessing RINC's web interface

By default, RINC is not exposed to the outside world. To access RINC's web interface, port-forward to the `rinc-web` service:

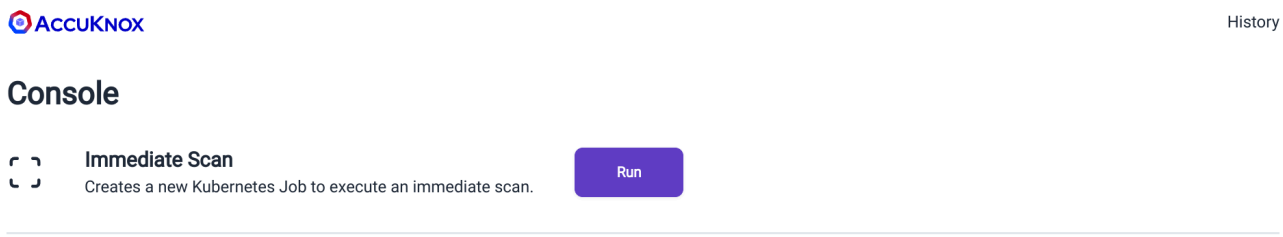
```
kubectl -n "$NAMESPACE" port-forward svc/rinc-web 8080:80
```

Now open `<http://localhost:8080>` in your browser.

An overview of RINC's web interface



If you open RINC's web interface immediately after installation, the reporting cronjob might not have scheduled yet, so you may see an empty welcome screen instead of the dashboard. However, don't worry - you can go to the Console by clicking on the top-right section of the page and start an "on-demand scan".



This will immediately launch a Kubernetes Job to aggregate all the metrics and generate a report for you. The job will take some time depending on the size of your cluster and workloads. Once the job is completed, you will see a dashboard similar to the example above.

An overview of the reports generated by RINC

RabbitMQ (2024-12-24 10:30:06 UTC)

Alerts

RabbitMQ ready messages count has reached 124222

Summary

Total messages in Queues	124240
Unacknowledged messages in Queues	18
Ready messages in Queues	124222
RabbitMQ Version	3.12.13
Total channels	1858
Total connections	1848
Total consumers	38
Total exchanges	217
Total queues	222

Nodes

Name	Running	CPU	Mem Used	Free Disk	Processes	Sockets	FD Used	Network Partitions	Uptime	Plugins
rabbit@rabbitmq-server-0.rabbitmq-nodes.rabbitmq	true	4	1026.34 MiB 1064.96 MiB high watermark	92.51 GiB 1.86 GiB low watermark	10545	692	766	No	2547h37m1s	rabbitmq_peer_discovery_k8s rabbitmq_prometheus rabbitmq_management
rabbit@rabbitmq-server-1.rabbitmq-nodes.rabbitmq	true	4	767.76 MiB 1064.96 MiB high watermark	92.53 GiB 1.86 GiB low watermark	7060	492	552	No	311h4m38s	rabbitmq_management rabbitmq_peer_discovery_k8s rabbitmq_prometheus rabbitmq_shovel rabbitmq_shovel_management
rabbit@rabbitmq-server-2.rabbitmq-nodes.rabbitmq	true	4	885.02 MiB 1064.96 MiB high watermark	92.51 GiB 1.86 GiB low watermark	9300	664	763	No	603h21m42s	rabbitmq_peer_discovery_k8s rabbitmq_prometheus rabbitmq_management

Queues

Name	State	Messages	Unacknowledged Messages	Ready Messages	Durable

Above is an example RabbitMQ report.

Every report begins with an **Alerts** section, displaying any fired alerts. The alerts are color-coded based on their severity:

1. **Red** - Indicates a critical alert.
2. **Yellow** - Indicates a warning.
3. **Info** - Provides useful information.

Critical alerts typically require immediate action. Warning alerts, if not addressed in time, may impact operations. Info alerts provide useful details, such as the number of onboarded registries and nodes.

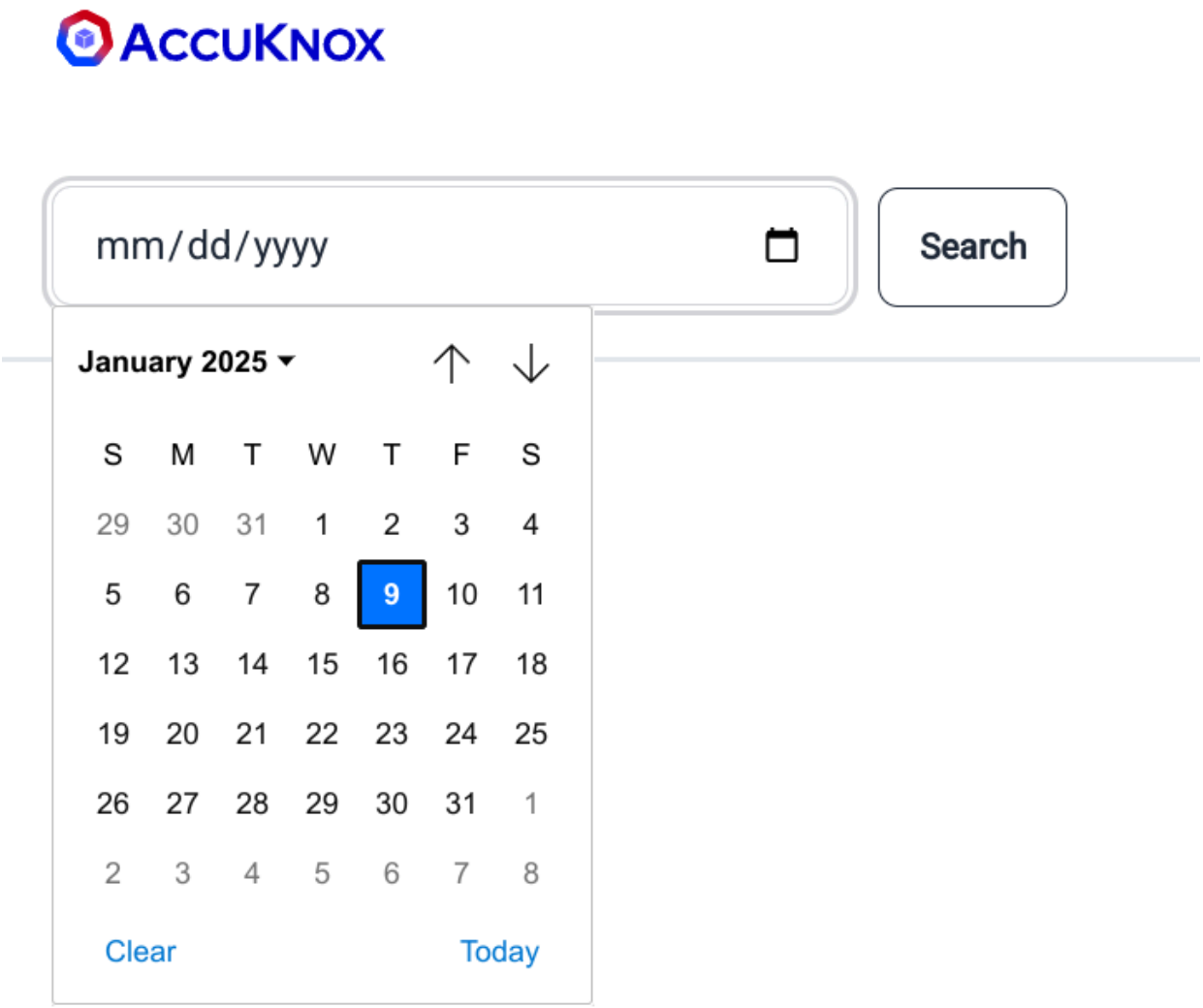
As a cluster operator, ensure there are no critical alerts.

Note: As described earlier, RINC supports email integration, allowing you to receive these alerts via email. Refer to the `email` section in the Helm chart to configure email integration.

The rest of the report varies depending on the type of report and includes insights about the cluster/service.

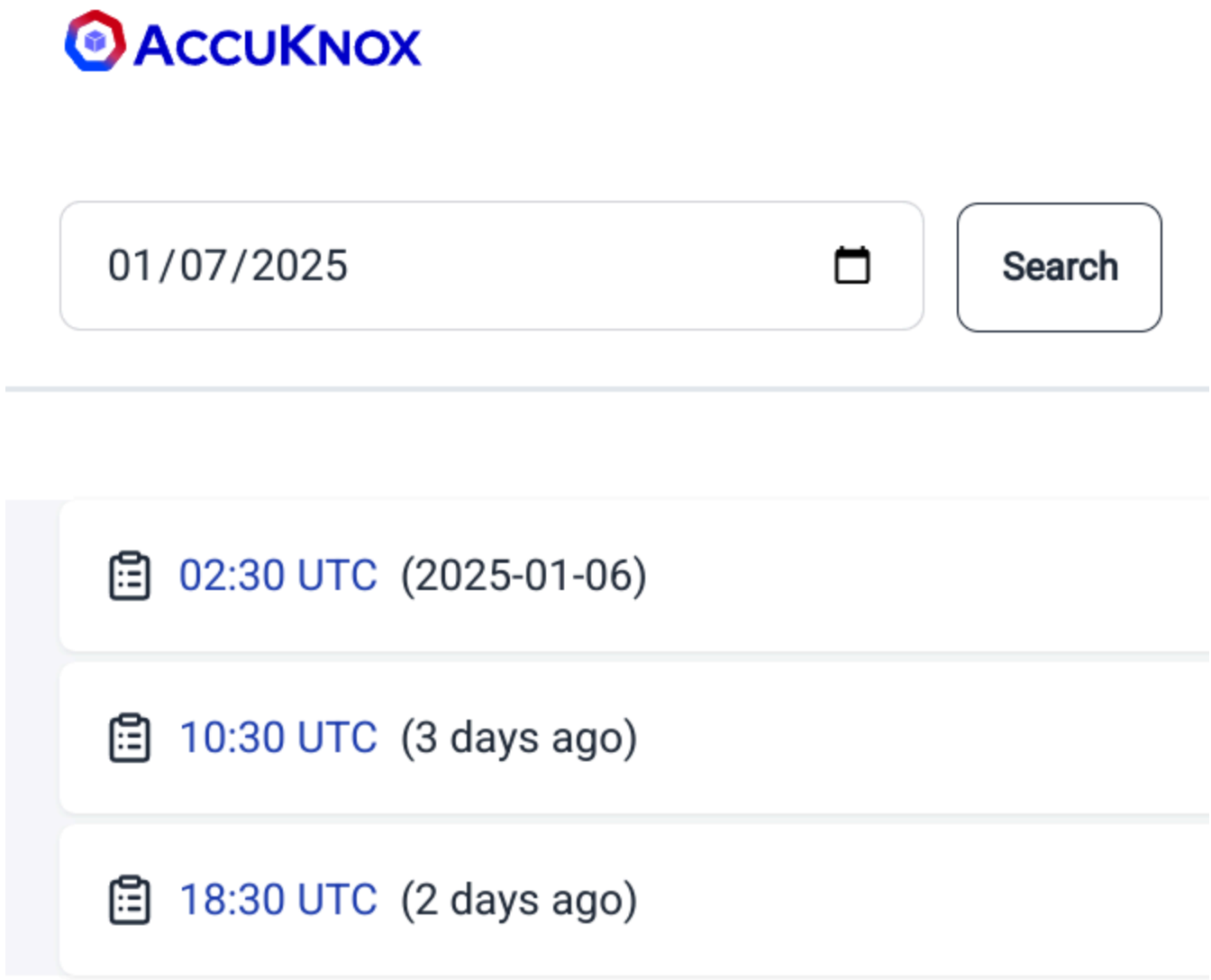
Fetching Old Reports

RINC retains old reports for the duration specified in `config.maintenance.metricsRetention` in the Helm values. To retrieve old reports, click on `History` at the top-right of the web interface to access the history page.



History Page

On this page, select the desired date to fetch the reports and click `Search`.



History Search Results - *All times are in UTC.*

Advanced

Minimum Required Database Permissions for RINC to Generate Reports

MongoDB:

RW access to the `rinc` collection

Postgresql:

SELECT access to the following within the `cwpp` schema (within the `accuknox` database) tables,

1. registry_scan_details
2. registries
3. image_scan_details
4. registry_configuration
5. workspaces
6. clusters
7. node

The query below creates a user named `rinc` with SELECT access to the listed tables under the `cwpp` schema.

```
CREATE USER rinc WITH PASSWORD 'tryguessingthis';
GRANT CONNECT ON DATABASE accuknox TO rinc;
GRANT USAGE ON SCHEMA cwpp TO rinc;
GRANT SELECT ON
  cwpp.registry_scan_details,
  cwpp.registries,
  cwpp.image_scan_details,
  cwpp.registry_configuration,
  cwpp.workspaces,
  cwpp.clusters,
  cwpp.nodes
TO rinc;
```

Neo4j:

Neo4j requires authentication to ping the database. It is recommended you created a separate database called "rinc" and a user, also called "rinc". This database is not going to be used and is only present to allow RINC to authenticate with neo4j in order to test the connectivity.

Vault Policy

If you are using Vault with Kubernetes auth, create a role and attach the necessary policy to allow reading your configured secrets.

Example, vault policy:

```
path "/accuknox/k8s/*" {
  capabilities = ["read"]
}

path "/accuknox/aws/*" {
  capabilities = ["read"]
}

path "/accuknox/artifacts/microservices/token" {
  capabilities = ["read"]
}
```

You also need to bind the role to the service accounts and namespace. RINC helm charts creates three service accounts. You can list them using,

```
kubectl -n "$NAMESPACE" get serviceaccounts
```

You should associate the role with all three service account names.

Once the role is created, refer to it in the Vault section of the Helm chart.

CWPP Troubleshooting

If the user faces any issue related to clusters, then they should provide the logs information of their clusters for troubleshooting purposes.

Requirements

Getting Kubearmor Sysdump

Users can get the kubeArmor sysdump by using the following command:

```
karmor sysdump
```

Getting logs from AccuKnox Agents

Along with KubeArmor Sysdump users will be required to send the logs of AccuKnox Agents running inside their cluster. To get the logs of each agent use the following commands:

```
kubectl logs -n accuknox-agents discovery-engine-xxxx-xxxx > discovery-engine-logs.txt
kubectl logs -n accuknox-agents feeder-service-xxxx-xxx > feeder-service-logs.txt
kubectl logs -n accuknox-agents policy-enforcement-agent-xxxx-xxx > PEA-logs.txt
kubectl logs -n accuknox-agents shared-informer-agent-XXX-XXx > SIA-logs.txt
```

Note: In the above command replace the xxx-xxxx with your respective pod name that is running in accuknox-agents namespace.

The users will have to send this Karmor sysdump file and AccuKnox Agents logs to AccuKnox Solutions team for debugging the issue.

Script To automate this process

- This script will save all the output Txt files in a single zip file
- karmor sysdump will run independently as it creates a separate zip file on it's own

```
#!/bin/bash

# Function to get the pod name for a given deployment
get_pod_name() {
  local namespace=$1
  local deployment=$2
  kubectl get po -n "$namespace" -o=name | grep "$deployment" | awk -F/ '{print $2}'
}

# Function to fetch logs for a given pod and save them to a file
fetch_and_save_logs() {
  local namespace=$1
  local pod=$2
  local output_file=$3
  kubectl logs -n "$namespace" "$pod" > "$output_file"
}

# Main script starts here
```

```
# Set your desired namespace here
namespace="accuknox-agents"

# Get the pod names and store them in variables
discovery_engine_pod=$(get_pod_name "$namespace" "discovery-engine")
feeder_service_pod=$(get_pod_name "$namespace" "feeder-service")
pea_pod=$(get_pod_name "$namespace" "policy-enforcement-agent")
sia_pod=$(get_pod_name "$namespace" "shared-informer-agent")

# Create a temporary directory to store the log files
temp_dir=$(mktemp -d 2>/dev/null || mktemp -d -t 'mytmpdir')

# Fetch and save the logs to separate files in the temporary directory
fetch_and_save_logs "$namespace" "$discovery_engine_pod"
"$temp_dir/discovery-engine-logs.txt"
fetch_and_save_logs "$namespace" "$feeder_service_pod" "$temp_dir/feeder-service-logs.txt"
fetch_and_save_logs "$namespace" "$pea_pod" "$temp_dir/PEA-logs.txt"
fetch_and_save_logs "$namespace" "$sia_pod" "$temp_dir/SIA-logs.txt"

# Create a ZIP archive of all the log files
zip_file="agents_logs_archive.zip"
zip -j "$zip_file" "$temp_dir"/*.txt

# Clean up the temporary directory
rm -rf "$temp_dir"

echo "Logs have been fetched and saved to the ZIP archive: $zip_file"

# Execute 'karmor sysdump'
karmor sysdump

echo "karmor sysdump executed."
```

Users can now send the zip files generated for troubleshooting.

Note: Need to install zip as a pre-requisite in linux before running the above script.

```
sudo apt install zip
```

Output

```
:~/logs$ ./script.sh
  adding: discovery-engine-logs.txt (deflated 95%)
  adding: feeder-service-logs.txt (deflated 64%)
  adding: PEA-logs.txt (deflated 43%)
  adding: SIA-logs.txt (deflated 55%)
Logs have been fetched and saved to the ZIP archive: agents_logs_archive.zip
getting logs from kubearmor-p4rlf
tar: removing leading '/' from member names
Sysdump at karmor-sysdump-Thu Oct 19 07_45_31 UTC 2023.zip
karmor sysdump executed.
:~/logs$ ls
agents_logs_archive.zip  'karmor-sysdump-Thu Oct 19 07_45_31 UTC 2023.zip'   script.sh
:~/logs$
```

CSPM Troubleshooting Guide

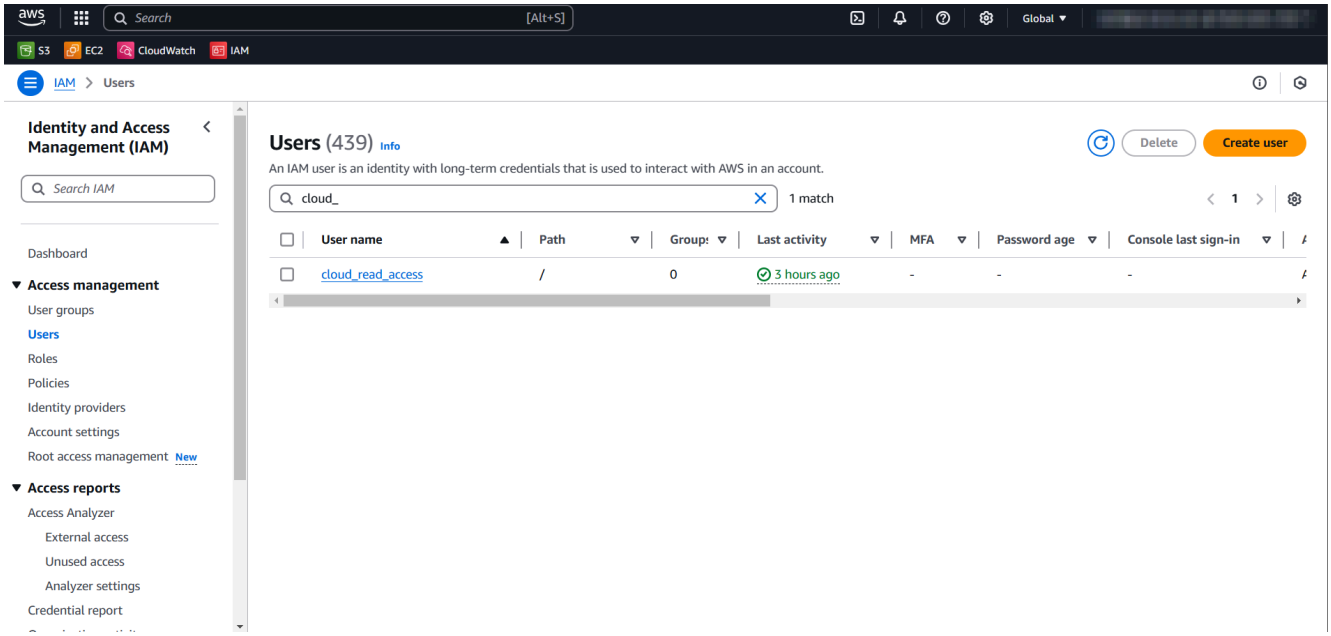
This guide helps troubleshoot onboarding and scanning issues for the Accuknox CNAPP SaaS deployment across AWS, Azure, and GCP.

Step 1: Validate Prerequisites

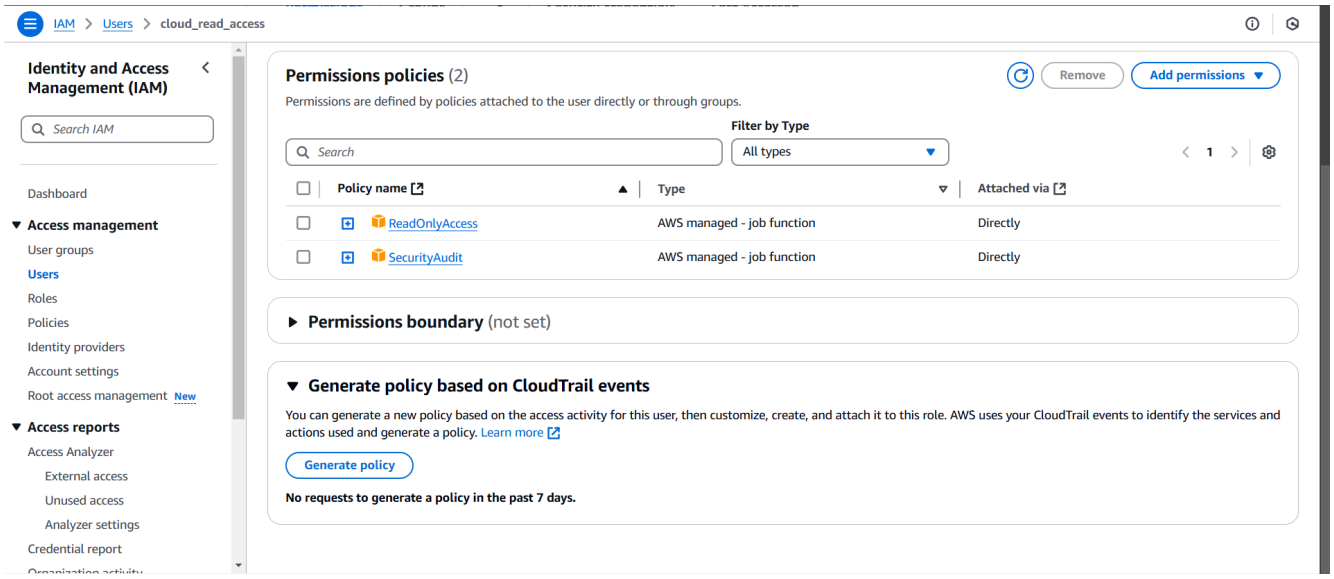
Ensure the required permissions are granted to the user or application for the respective cloud account.

AWS Permissions

1. **Login to AWS Console**.
2. Navigate to **IAM > Users**.
3. Select the user created for AccuKnox onboarding.



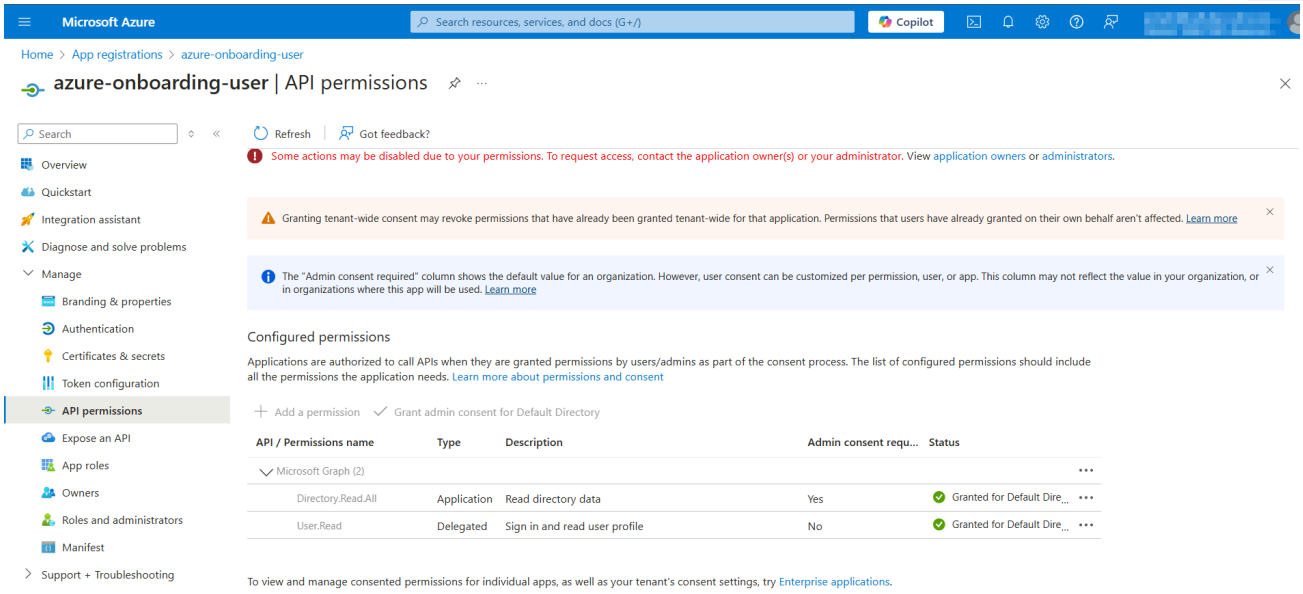
1. Go to the **Permissions** tab:
    - Confirm the following policies are attached:
        - ReadOnlyAccess (AWS Managed - Job Function)
        - SecurityAudit (AWS Managed - Job Function)



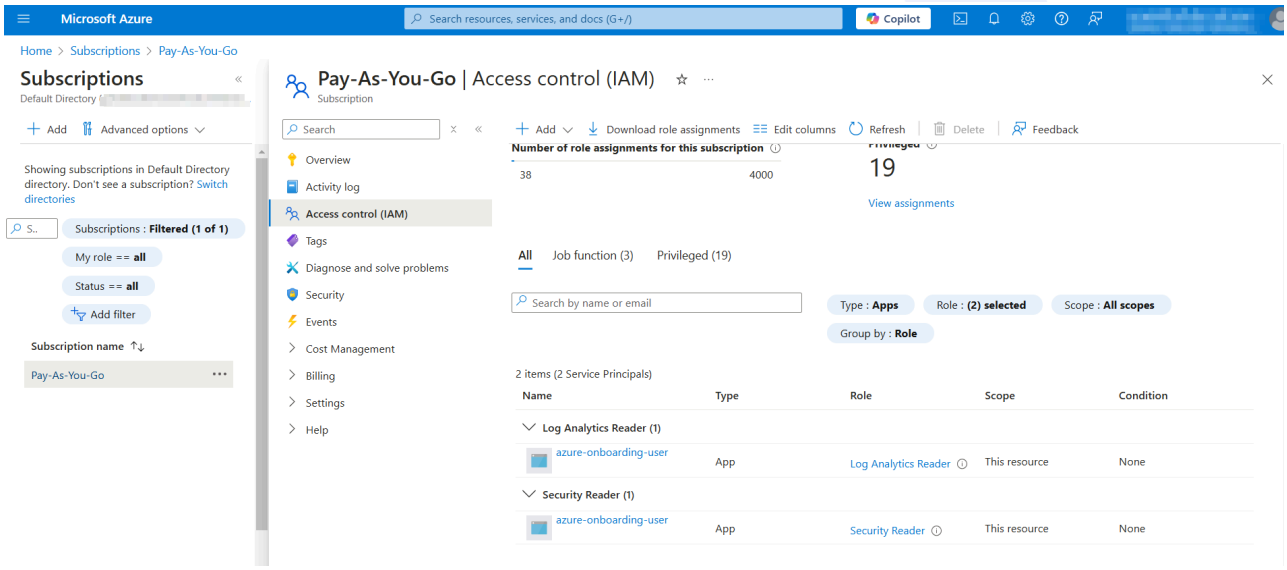
Azure Permissions

1. **Login to Azure Portal**.
2. Navigate to **App Registrations**:

- Select the application registered for onboarding.
- Go to the **API Permissions** tab and verify:
    - Directory.Read.All is listed under **Application Permissions**.

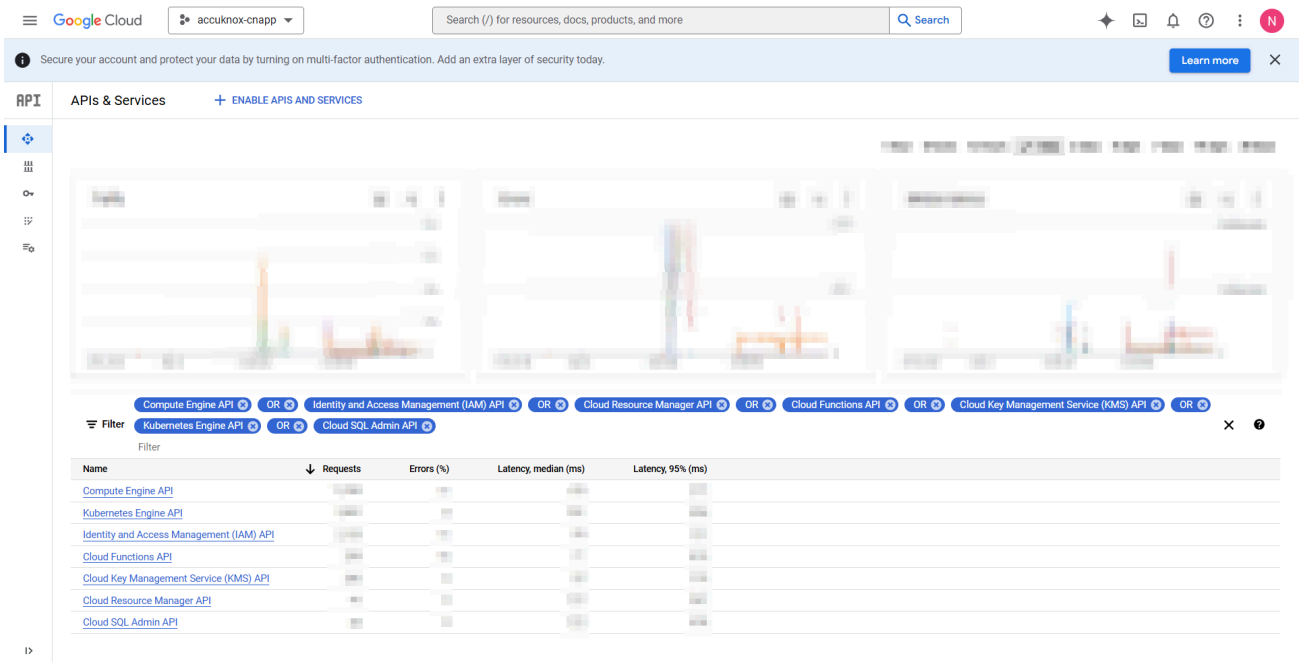


1. Navigate to **Subscriptions**:
    - Select the relevant subscription.
    - Go to **Manage > Access control (IAM)**.
    - Verify the registered application has the following roles assigned:
        - Security Reader (Job Function Role for subscriptions)
        - Log Analytics Reader (Job Function Role for subscriptions)



GCP Permissions

1. **Login to Google Cloud Console**.
2. Navigate to **IAM & Admin > IAM**:
    - Find the service account created for onboarding.
    - Verify the following roles are assigned:
        a. `roles/viewer` (Viewer Role)
        b. `roles/iam.securityReviewer` (Security Reviewer Role)
        c. `roles/logging.viewer` (Log Viewer Role)
3. Navigate to **APIs & Services > Library**:
    - Ensure the following APIs are enabled:
        a. Compute Engine API
        b. Identity and Access Management (IAM) API
        c. Cloud Resource Manager API
        d. Cloud Functions API
        e. KMS API
        f. Kubernetes API
        g. Cloud SQL Admin API



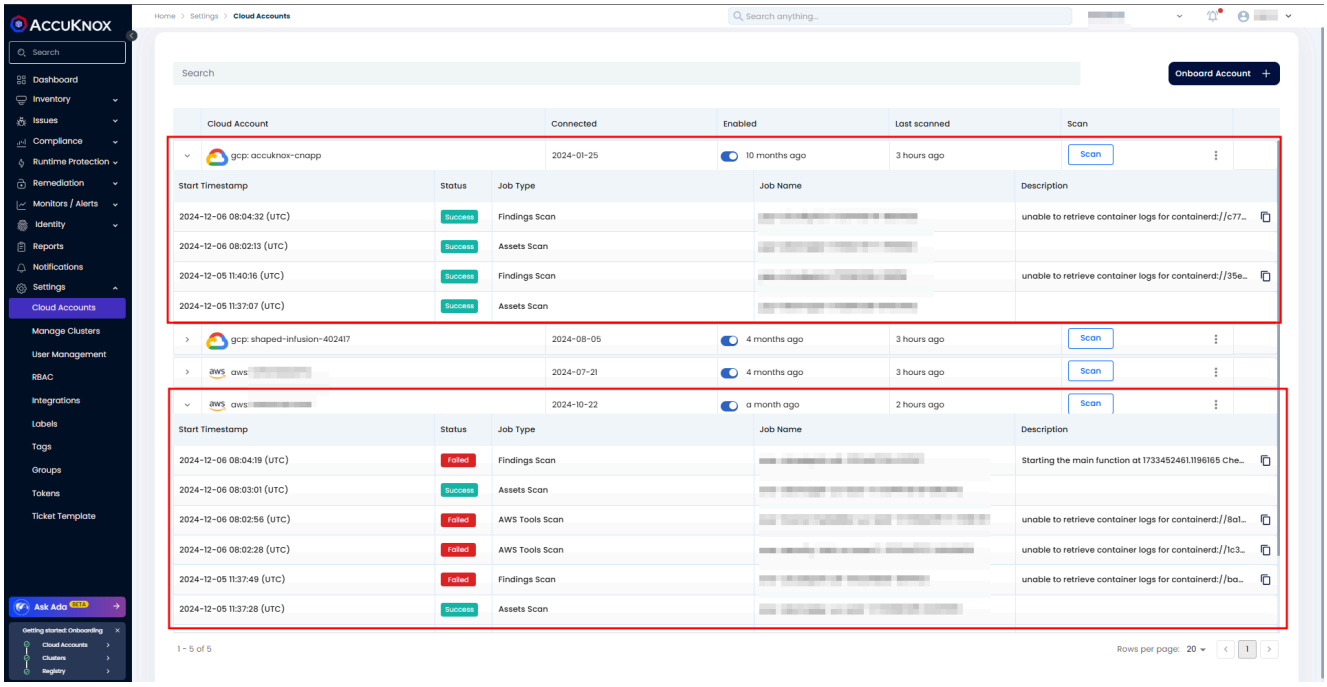
If permissions and APIs are configured correctly, proceed to the next step.

Refer to the prerequisites for more info:

- AWS Onboarding Prerequisites
- Azure Onboarding Prerequisites
- GCP Onboarding Prerequisites

Step 2: Verify Cloud Scan Status

1. Log in to the **AccuKnox SaaS platform**.
2. Navigate to **Settings > Cloud Account**.
3. Select the **specific cloud account** in question.
4. Review the **status of the cloud scan**:


ACCUKNOX
Search
Dashboard
Inventory
Issues
Compliance
Runtime Protection
Remediation
Monitors / Alerts
Identity
Reports
Notifications
Settings
Cloud Accounts
Manage Clusters
User Management
RBAC
Integrations
Labels
Tags
Groups
Tokens
Ticket Template
Ask Ada
Home > Settings > Cloud Accounts
Search anything...
Search
Onboard Account
Cloud Account
Connected
Enabled
Last scanned
Scan
gcp: accuknox-cnapp
2024-01-25
10 months ago
3 hours ago
Scan
Start Timestamp
Status
Job Type
Job Name
Description
2024-12-06 08:04:32 (UTC)
Success
Findings Scan
unable to retrieve container logs for containerd://c77...
2024-12-06 08:02:13 (UTC)
Success
Assets Scan
2024-12-05 11:40:16 (UTC)
Success
Findings Scan
unable to retrieve container logs for containerd://35e...
2024-12-05 11:37:07 (UTC)
Success
Assets Scan
gcp: shaped-infusion-402417
2024-08-05
4 months ago
3 hours ago
Scan
aws:
2024-07-21
4 months ago
3 hours ago
Scan
aws:
2024-10-22
a month ago
2 hours ago
Scan
Start Timestamp
Status
Job Type
Job Name
Description
2024-12-06 08:04:19 (UTC)
Failed
Findings Scan
Starting the main function at 1733452461.1196165 Che...
2024-12-06 08:03:01 (UTC)
Success
Assets Scan
2024-12-06 08:02:56 (UTC)
Failed
AWS Tools Scan
unable to retrieve container logs for containerd://8a1...
2024-12-06 08:02:28 (UTC)
Failed
AWS Tools Scan
unable to retrieve container logs for containerd://1c3...
2024-12-05 11:37:49 (UTC)
Failed
Findings Scan
unable to retrieve container logs for containerd://ba...
2024-12-05 11:37:28 (UTC)
Success
Assets Scan
1 - 5 of 5
Rows per page: 20
1