**Rancher Monitoring Documentation for Airgapped Kubernetes Environments**

**Overview**

This documentation provides a comprehensive guide for architecting, installing, and operating the Rancher Monitoring stack in airgapped Kubernetes environments. It covers all major components including Prometheus, Alertmanager, Grafana, Node Exporter, and the complete Thanos stack for scalable, durable monitoring and observability. The guidance assumes use of Rancher as the cluster management platform in offline/isolation scenarios with PVC-backed persistent storage and internal container registries.

**Table of Contents**

1. [Introduction](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#introduction)
2. [Architecture Overview](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#architecture-overview)
   1. [Key Components](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#key-components)
   2. [Data Flow](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#data-flow)
3. [Pre-requisites](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#pre-requisites)
4. [Installation](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#installation)
   1. [Prepare Airgapped Environment](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#prepare-airgapped-environment)
   2. [Configure Persistent Storage](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#configure-persistent-storage)
   3. [Setup Internal Container Registry](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#setup-internal-container-registry)
   4. [Configure Helm Chart Values](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#configure-helm-chart-values)
   5. [Deploy Rancher Monitoring Stack](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#deploy-rancher-monitoring-stack)
5. [Component Details](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#component-details)
   1. [Prometheus](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#prometheus)
   2. [Alertmanager](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#alertmanager)
   3. [Grafana](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#grafana)
   4. [Node Exporter](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#node-exporter)
   5. [Thanos](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#thanos)
6. [Operational Considerations](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#operational-considerations)
7. [Troubleshooting](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#troubleshooting)
8. [Reference Links](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#reference-links)

**Introduction**

Rancher Monitoring leverages Prometheus and related CNCF projects to provide a full-featured monitoring and alerting stack integrated into Kubernetes clusters managed by Rancher. This documentation is tailored for airgapped environments—networks isolated from the public internet—requiring on-premise image hosting and PVC-based storage.

**Architecture Overview**

**Key Components**

* **Rancher Management Plane**: Central cluster management and orchestration.
* **Prometheus**: Core metrics collector, scraping targets and evaluating alert rules.
* **Alertmanager**: Alert routing and notification management.
* **Grafana**: Visualization layer with custom dashboards.
* **Node Exporter**: Metrics exporter running on each node.
* **Thanos**: Provides horizontally scalable, durable long-term storage and global querying via:
  + Sidecar running alongside Prometheus uploads data.
  + Store Gateway serving data blocks from PVC.
  + Compactor for downsampling and block compaction.
  + Query API aggregating data across Prometheus and stores.
  + Bucket Web UI for browsing stored metrics.
  + Ruler for scalable evaluation of alerting rules.
* **PVC-backed Storageclass**: Kubernetes native persistent storage for data retention.
* **Internal Registry**: Storing container images locally for airgapped deployments.
* **NGINX Ingress Controller**: Provides external user access via hostname routing.

**Data Flow Diagram**

![Architecture Diagram](your\_architecture\_diagram

*Metrics flow from Node Exporter → Prometheus → Thanos Sidecar → Thanos Store → Thanos Query → Grafana. Alerts flow Prometheus → Alertmanager. External access via Ingress.*

**Pre-requisites**

* Kubernetes cluster managed by Rancher (version compatible with Rancher Monitoring v105+).
* PVC storage class configured to support ReadWriteOnce access mode.
* Internal container registry available with Rancher Monitoring images pushed.
* NGINX Ingress controller installed in the cluster.
* Helm version 3.x installed on deployment workstation.
* Access to Rancher CLI and Kubernetes API.

**Installation**

**Prepare Airgapped Environment**

* Mirror Rancher Monitoring container images (Prometheus, Alertmanager, Grafana, Thanos, Node Exporter) to internal registry.
* Create required Kubernetes namespaces and RBAC roles.

**Configure Persistent Storage**

* Create or verify existing PVC storage class with adequate storage and access modes.
* Define PVC volumes via Helm values for Prometheus, Alertmanager, Grafana, and Thanos components.

**Setup Internal Container Registry**

* Configure imagePullSecrets in Kubernetes namespaces to pull from internal registry.
* Ensure Helm chart values reference proper internal image URLs.

**Configure Helm Chart Values**

Create a values.yaml with entries such as:

text

global:

systemDefaultRegistry: "<internal\_registry>"

alertmanager:

enabled: true

ingress:

enabled: true

hosts:

- alertmanager.example.com

alertmanagerSpec:

externalUrl: "https://alertmanager.example.com"

storage:

volumeClaimTemplate:

spec:

storageClassName: "<your-storage-class>"

accessModes: ["ReadWriteOnce"]

resources:

requests:

storage: "10Gi"

prometheus:

ingress:

enabled: true

hosts:

- prometheus.example.com

prometheusSpec:

externalUrl: "https://prometheus.example.com"

storageSpec:

volumeClaimTemplate:

spec:

storageClassName: "<your-storage-class>"

accessModes: ["ReadWriteOnce"]

resources:

requests:

storage: "50Gi"

retention: "15d"

thanos:

create: true

image: "<internal\_registry>/thanos/thanos:v0.35.1"

version: "v0.35.1"

logLevel: info

objectStorageConfig:

name: thanos-objstore-config

key: thanos.yaml

grafana:

enabled: true

ingress:

enabled: true

hosts:

- grafana.example.com

url: "https://grafana.example.com"

persistence:

enabled: true

type: pvc

storageClassName: "<your-storage-class>"

accessModes: ["ReadWriteOnce"]

size: "10Gi"

dashboards:

enabled: true

customDashboards:

my-namespace-dashboard:

gnetId: 1860

datasource: Prometheus

prometheus-node-exporter:

enabled: true

thanos:

enabled: true

objstoreConfig: |

type: FILESYSTEM

config:

directory: /data/thanos

persistence:

enabled: true

storageClass: "<your-storage-class>"

accessModes: ["ReadWriteOnce"]

size: "100Gi"

compact:

enabled: true

persistence:

enabled: true

storageClass: "<your-storage-class>"

accessModes: ["ReadWriteOnce"]

size: "20Gi"

storegateway:

enabled: true

persistence:

enabled: true

storageClass: "<your-storage-class>"

accessModes: ["ReadWriteOnce"]

size: "20Gi"

query:

enabled: true

bucketweb:

enabled: true

ruler:

enabled: true

persistence:

enabled: true

storageClass: "<your-storage-class>"

accessModes: ["ReadWriteOnce"]

size: "20Gi"

images:

alertmanager: "<internal\_registry>/prometheus/alertmanager:v0.25.0"

prometheus: "<internal\_registry>/prometheus/prometheus:v2.45.0"

grafana: "<internal\_registry>/grafana/grafana:10.2.0"

nodeExporter: "<internal\_registry>/prometheus/node-exporter:v1.6.0"

thanos: "<internal\_registry>/thanos/thanos:v0.35.1"

**Deploy Rancher Monitoring Stack**

1. Add Rancher monitoring helm repo if needed.
2. Use Helm CLI to deploy using custom values:

bash

helm upgrade --install rancher-monitoring rancher-monitoring-chart \

--namespace cattle-monitoring-system \

-f values.yaml

1. Verify all pods are running:

bash

kubectl get pods -n cattle-monitoring-system

**Component Details**

**Prometheus**

* Scrapes cluster and node metrics.
* Has configured retention of 15 days locally.
* Runs a Thanos sidecar to upload blocks to Thanos PVC storage.
* Evaluates alerting rules and forwards alerts to Alertmanager.

**Alertmanager**

* Receives alerts from Prometheus and routes notifications to configured receivers (email, Slack, etc).
* Highly available and persistent through PVC.

**Grafana**

* Provides dashboards and visualization of metrics sourced from Prometheus or Thanos Query APIs.
* Supports custom dashboards configured via Helm values or ConfigMaps.

**Node Exporter**

* Runs as a DaemonSet on each node.
* Collects OS and hardware-level metrics for Prometheus scraping.

**Thanos**

* Extends Prometheus with horizontally scalable long-term storage and federation.
* Components:
  + Sidecar for data upload alongside Prometheus.
  + Store Gateway for serving stored metrics.
  + Compactor for efficient data management.
  + Query API merges data from multiple sources.
  + Bucket Web UI for storage browsing.
  + Ruler for scalable rule evaluation.
* Uses PVC-backed storage configured with storageclass for persistence in airgapped mode.

**Operational Considerations**

* Plan storage sizing carefully based on retention policies.
* Keep the internal registry fully synchronized for all required images.
* Verify ingress hostnames and TLS setups for secure external access.
* Monitor component logs and alerting workflows regularly.
* Perform regular backups of persistent volumes if possible.

**Troubleshooting**

* **Pods not starting**: Check PVC and storage class configurations.
* **Image pull errors**: Ensure imagePullSecrets are correct and registry accessible.
* **Alert routing failures**: Validate Alertmanager config and reachable notification endpoints.
* **Data missing in Grafana**: Confirm Prometheus and Thanos are up and integrated properly.

**Reference Links**

* [Rancher Monitoring Helm Chart GitHub](https://github.com/rancher/charts/tree/dev-v2.12/charts/rancher-monitoring)
* [Prometheus Official Documentation](https://prometheus.io/docs/)
* [Thanos Official Documentation](https://thanos.io/tip/)
* [Grafana Official Documentation](https://grafana.com/docs/)
* [Rancher Docs - Monitoring & Alerting](https://ranchermanager.docs.rancher.com/)
* [Kubernetes Persistent Volumes](https://kubernetes.io/docs/concepts/storage/persistent-volumes/)

This document is suitable as a Confluence wiki page or internal technical reference to simplify the deployment and operation of Rancher Monitoring in disconnected airgapped environments, ensuring observability best practices and resilience.

If needed, detailed architecture diagrams and flowcharts can be added as supporting visual material.

Add to follow-up

Check sources