**Rancher Monitoring Detailed Documentation for Airgapped Kubernetes Environments**

**Overview**

This document provides an in-depth guide for deploying and managing the Rancher Monitoring stack on Kubernetes clusters managed by Rancher (including RKE2) in airgapped environments. It includes detailed component descriptions, configuration best practices, network port usage, and operational insights from the perspective of a Rancher architect and certified Kubernetes administrator.

**Table of Contents**

1. [Architecture Overview](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#architecture-overview)
2. [Component Deep Dive](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#component-deep-dive)
   * [Rancher Management Plane](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#rancher-management-plane)
   * [Node Exporter](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#node-exporter)
   * [Prometheus](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#prometheus)
   * [Alertmanager](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#alertmanager)
   * [Grafana](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#grafana)
   * [Thanos Components](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#thanos-components)
     + [Thanos Sidecar](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#thanos-sidecar)
     + [Thanos Store Gateway](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#thanos-store-gateway)
     + [Thanos Compactor](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#thanos-compactor)
     + [Thanos Query](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#thanos-query)
     + [Thanos Bucket Web](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#thanos-bucket-web)
     + [Thanos Ruler](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#thanos-ruler)
   * [NGINX Ingress Controller](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#nginx-ingress-controller)
   * [Persistent Volume Claims (PVC)](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#persistent-volume-claims-pvc)
   * [Internal Container Registry](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#internal-container-registry)
3. [Installation & Configuration](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#installation-configuration)
4. [Port and Network Configuration](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#port-and-network-configuration)
5. [Operational Best Practices](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#operational-best-practices)
6. [Troubleshooting & Debugging](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#troubleshooting-debugging)
7. [References & Further Reading](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#references-further-reading)

**Architecture Overview**

Rancher Monitoring in an airgapped Kubernetes environment is designed for full observability leveraging CNCF projects, orchestrated by Rancher. The stack includes metrics collection, alerting, dashboard visualization, long-term storage, and scalability via Thanos. PVC-based persistent storage is used instead of cloud object storage, making the stack fully on-premises compatible.

**Component Deep Dive**

**Rancher Management Plane**

* **Role**: Centralized operational control over Kubernetes clusters; manages lifecycle of cluster components, including monitoring.
* **Key Features**: Multi-cluster management, integrated with Helm and Kubernetes API.
* **Interaction**: Manages application deployment, image sync with internal registry, and cluster health monitoring.

**Node Exporter**

* **Purpose**: Exposes node-level Linux/Windows system metrics (CPU, memory, disk I/O, network) to Prometheus.
* **Deployment**: DaemonSet on each cluster node.
* **Ports**:
  + Default: **9100/TCP** for metrics scraping.
* **Metrics examples**: node\_cpu\_seconds\_total, node\_memory\_MemAvailable\_bytes, node\_network\_receive\_bytes\_total.
* **Security**: Should be in a restricted namespace with minimal privileges.

**Prometheus**

* **Purpose**: Time-series database and metrics collection engine. Scrapes Node Exporter and other exporters.
* **Key Features**: Alert rule evaluation, multi-target scraping, data retention management.
* **Persistence**: Local data retained on PVC (configured storage class).
* **Thanos sidecar**: Runs alongside Prometheus to upload blocks to long-term store.
* **Ports**:
  + **9090/TCP**: Prometheus web UI and API
  + Scraping ports vary depending on targets.
* **Retention**: Typical local retention configured (e.g., 15 days), offloaded to Thanos for long-term.

**Alertmanager**

* **Role**: Central alert routing and deduplication.
* **Components**:
  + Alert reception (API endpoint).
  + Silencing and grouping logic.
  + Notification integrations (email, Slack, PagerDuty, webhook).
* **Persistence**: Configured with PVC-backed storage for HA setups.
* **Ports**:
  + **9093/TCP**: Alertmanager HTTP API and Web UI.
* **Key Features**: Highly available by running in cluster with persistent state.
* **Configuration**: Alertmanager configmaps contain route and receiver info.

**Grafana**

* **Purpose**: Visualization of Prometheus/Thanos metrics and dashboards.
* **Features**: Customizable dashboards, alerting rules (via Grafana), variable querying.
* **Persistence**: PVC-backed persistence for dashboards and user settings.
* **Ports**:
  + **3000/TCP**: Grafana HTTP UI and API.
* **Dashboards**: Custom dashboards injected via Helm values or configmaps.

**Thanos Components**

**Thanos Sidecar**

* **Function**: Runs with Prometheus, uploads blocks to PVC storage, allows Prometheus to be horizontally scalable.
* **Key integration**: Uploads data to Thanos Store and exposes metrics for Thanos Query.
* **Ports**:
  + **10901/TCP**: gRPC metrics and sidecar communication.

**Thanos Store Gateway**

* **Role**: Serves queried historical data blocks stored on PVC as block objects.
* **Persistence**: PVC-backed storage for all historic metrics.
* **Ports**:
  + **10901/TCP**: gRPC serving data.

**Thanos Compactor**

* **Purpose**: Compacts and down-samples time-series blocks to reduce storage and improve query performance.
* **Persistence**: Uses PVC for intermediate and final compacted data.
* **Periodic Job**: Runs compaction and retention policy enforcement.

**Thanos Query**

* **Role**: Federates queries combining live Prometheus data and historic data from Store Gateways.
* **Ports**:
  + **10902/TCP**: HTTP query API.
* **Features**: Multi-source query aggregation, high availability querying.

**Thanos Bucket Web**

* **Purpose**: UI to browse stored metric blocks.
* **Usage**: Useful for troubleshooting storage state.

**Thanos Ruler**

* **Role**: Evaluates Prometheus recording and alerting rules at scale.
* **Persistence**: Stores rule results on PVC.
* **Ports**:
  + **10910/TCP**: HTTP API for rules results.

**NGINX Ingress Controller**

* Provides external HTTP(S) routing to services inside Kubernetes including Grafana, Alertmanager, Prometheus.
* Use TLS for secure access.
* Routes traffic based on hostname configured in hosts values.
* Terminates SSL offloading externally if needed.

**Persistent Volume Claims (PVC)**

* Storage class backing PVCs should support **ReadWriteOnce** mode.
* PVCs are allocated for Prometheus, Alertmanager, Grafana, and Thanos components.
* PVC-backed storage replaces cloud object storage typical in Thanos setups.
* Important for resilience across pod restarts, upgrades.

**Internal Container Registry**

* Airgapped environment requires mirroring all container images needed by Rancher Monitoring to the internal registry.
* Helm values should map image paths to this internal registry.
* ImagePullSecrets configured correspondingly in Kubernetes namespaces.

**Installation & Configuration**

1. **Mirror Rancher Monitoring images** to internal registry.
2. **Configure storageclass and PVCs** for all monitoring components.
3. Prepare a **values.yaml** as described in prior documentation, ensuring PVC persistence and internal registry image paths.
4. Deploy Rancher Monitoring Helm chart with:

bash

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

--namespace cattle-monitoring-system -f values.yaml

1. Validate deployment by checking pod status and logs.
2. Access UI endpoints via ingress hostnames over HTTPS.

**Port and Network Configuration**

| **Component** | **Port** | **Protocol** | **Description** |
| --- | --- | --- | --- |
| Node Exporter | 9100 | TCP | Metrics endpoint scraped by Prometheus |
| Prometheus | 9090 | TCP | Web UI, API, scraping endpoints |
| Alertmanager | 9093 | TCP | Alertmanager HTTP API and Web UI |
| Grafana | 3000 | TCP | Web UI and API |
| Thanos Sidecar | 10901 | TCP (gRPC) | Sidecar communication and metadata |
| Thanos Store | 10901 | TCP (gRPC) | Store Gateway API |
| Thanos Query | 10902 | TCP | Query API |
| Thanos Ruler | 10910 | TCP | Rules API and HTTP endpoints |
| NGINX Ingress | 80, 443 | TCP | External HTTP(S) ingress proxy |

**Operational Best Practices**

* Regularly update internal registry with latest images.
* Monitor PVC storage usage; scale storage class as necessary.
* Configure backups of PVC-backed volumes if data loss is critical.
* Secure ingress with TLS and restrict access by RBAC/network policies.
* Monitor alerting workflows end to end; test alert delivery regularly.
* Use Prometheus retention and Thanos compactor policies to optimize storage usage and cost.

**Troubleshooting & Debugging**

* **Failed image pulls**: Verify internal registry connectivity, image tags, and secrets.
* **Pod crash loops**: Check persistent storage mounts and logs for errors.
* **No metrics in Grafana**: Confirm Prometheus and Thanos Query API availability.
* **Alerts not firing**: Validate Prometheus alert rules and Alertmanager configs.

**References & Further Reading**

* [Rancher Monitoring Helm Chart (GitHub)](https://github.com/rancher/charts/tree/dev-v2.12/charts/rancher-monitoring)
* [Prometheus Official Docs](https://prometheus.io/docs/)
* [Thanos Documentation](https://thanos.io/tip/)
* [Grafana Documentation](https://grafana.com/docs/)
* [Kubernetes Persistent Storage Concepts](https://kubernetes.io/docs/concepts/storage/persistent-volumes/)
* [Certified Kubernetes Administrator (CKA) Guide](https://training.linuxfoundation.org/certification/certified-kubernetes-administrator-cka/)
* [Certified Kubernetes Security Specialist (CKS) Program](https://training.linuxfoundation.org/certification/certified-kubernetes-security-specialist-cks/)

This documentation provides the depth and clarity expected by Kubernetes and Rancher architects, ensuring that airgapped environments can achieve production-grade observability with Rancher Monitoring. For complete deployment success, tune storage sizing, ingress security, and alert management to your organization's needs.

**Rancher Monitoring Comprehensive Documentation for Airgapped Kubernetes Environments**

**(Including Detailed Architecture, Installation, Operation, and PTO Readiness)**

**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)
3. [Component Details](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#component-details)
4. [Installation and Configuration](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#installation-and-configuration)
5. [Networking and Security](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#networking-and-security)
6. [Operational Best Practices](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#operational-best-practices)
7. [PTO (Production Turnover) Readiness Checklist](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#pto-production-turnover-readiness-checklist)
8. [Troubleshooting and Maintenance](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#troubleshooting-and-maintenance)
9. [Reference Links](https://www.perplexity.ai/search/i-want-to-enable-alertmanager-sbtu2fNiS9OG2lOW247yRg#reference-links)

**Introduction**

This document outlines the design, deployment, and operational guidance for the Rancher Monitoring stack in airgapped Kubernetes environments managed by Rancher and RKE2. It focuses on a production-ready, PVC-backed persistent storage setup, internal container registries, and robust monitoring components including Prometheus, Alertmanager, Grafana, Node Exporter, and the full Thanos stack.

**Architecture Overview**

**Key Components**

| **Component** | **Purpose** | **Persistence** | **Ports** |
| --- | --- | --- | --- |
| Rancher Management | Cluster lifecycle, app management, monitoring deployment | N/A | N/A |
| Node Exporter | Node metrics exporter | None | 9100 TCP |
| Prometheus | Time-series metrics collection, Alert evaluation | PVC | 9090 TCP |
| Alertmanager | Alert ingestion, deduplication, routing | PVC | 9093 TCP |
| Grafana | Visualization of metrics via dashboards | PVC | 3000 TCP |
| Thanos Sidecar | Syncs Prometheus blocks to long-term storage | None | 10901 TCP (gRPC) |
| Thanos Store Gateway | Serves historic metrics stored on PVC | PVC | 10901 TCP (gRPC) |
| Thanos Compactor | Compacts and downsamples metrics blocks | PVC | 10901 TCP (gRPC) |
| Thanos Query | Query aggregation across data sources | None | 10902 TCP |
| Thanos Bucket Web | UI for browsing metric blocks | None | 10902 TCP |
| Thanos Ruler | Evaluates recording and alerting rules at scale | PVC | 10910 TCP |
| NGINX Ingress Ctrl | Routes external requests for services | N/A | 80/443 TCP |
| PVC Storage | Persistent volume backing for Prometheus, Alertmanager, Thanos, Grafana | PVC | N/A |
| Internal Registry | Stores required container images for airgapped environment | N/A | N/A |

**Component Details**

**1. Rancher Management Plane**

* Orchestrates cluster management and monitoring app deployment.
* Monitors health via integrated metrics and alerts.

**2. Node Exporter**

* DaemonSet deployed on all nodes collecting OS-level metrics.
* Scraped by Prometheus on port 9100.

**3. Prometheus**

* Scrapes metrics, evaluates alerting rules.
* Local data retention (e.g., 15 days).
* Runs a Thanos sidecar for long-term data sync.
* Web UI and API listen on 9090.

**4. Alertmanager**

* Group, silence, route alerts to notification endpoints.
* Clustered with PVC-backed storage for reliability.
* HTTP UI/API on 9093.

**5. Grafana**

* Visualizes metrics with user-defined dashboards.
* Persists settings on PVC.
* UI on port 3000.

**6. Thanos Components**

**Thanos Sidecar**

* Uploads Prometheus data blocks to PVC storage.
* Exposes data to Thanos Query over gRPC port 10901.

**Thanos Store Gateway**

* Serves historical block data from PVC for query.
* Runs on port 10901 gRPC.

**Thanos Compactor**

* Compacts and downsamples metrics for storage efficiency.
* Runs periodic compaction jobs.

**Thanos Query**

* Aggregates data across Thanos Store, Thanos Sidecar, and Prometheus endpoints.
* HTTP API on 10902.

**Thanos Bucket Web**

* User interface to inspect metrics blocks.

**Thanos Ruler**

* Evaluates alerting and recording rules at scale.
* PVC-backed persistence stores rule results.

**Installation and Configuration**

1. Mirror container images for all components (Prometheus, Alertmanager, Grafana, Node Exporter, Thanos components) to internal registry.
2. Define appropriate **Persistent Volume Claims (PVCs)** leveraging Kubernetes storageclass supporting ReadWriteOnce.
3. Customize the Helm chart values.yaml with PVC storage, internal registry images, NGINX ingress hosts, and enable components:

text

global:

systemDefaultRegistry: "<internal\_registry>"

prometheus:

prometheusSpec:

storageSpec:

volumeClaimTemplate:

spec:

storageClassName: "<storageclass>"

resources:

requests:

storage: "50Gi"

retention: "15d"

thanos:

create: true

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

grafana:

persistence:

storageClassName: "<storageclass>"

size: "10Gi"

alertmanager:

alertmanagerSpec:

storage:

volumeClaimTemplate:

spec:

storageClassName: "<storageclass>"

resources:

requests:

storage: "10Gi"

thanos:

persistence:

storageClass: "<storageclass>"

size: "100Gi"

...

1. Deploy monitoring stack with helm and verify pod readiness.
2. Configure ingress DNS, TLS, and access rules.

**Networking and Security**

* Confirm firewall rules permit internal cluster communication on required ports (e.g., 9100, 9090, 9093, 10901).
* Ingress TLS offloading recommended for secure external access.
* Image pull secrets configured for private registry.
* RBAC and Network Policies applied to control access.

**Operational Best Practices**

* Schedule regular compaction and retention maintenance.
* Monitor PVC storage utilization and expand as needed.
* Implement backup strategies for critical PVCs.
* Test alerting pipelines end-to-end before production cutover.
* Maintain image registries up to date with security patches.

**PTO (Production Turnover) Readiness Checklist**

| **Item** | **Status** | **Remarks** |
| --- | --- | --- |
| All components deployed |  | Prometheus, Alertmanager, Grafana, Thanos, Node Exporter |
| PVCs attached & validated |  | Ensure ReadWriteOnce PVC bound |
| Internal registry configured |  | All images mirrored and accessible |
| Network ports open & tested |  | Ingress and inter-component ports |
| Monitoring dashboards tested |  | Dashboards reflect cluster state |
| Alerts tested & acknowledged |  | Alertmanager properly routes alerts |
| Backup mechanism in place |  | PVC snapshots or external backup |
| Access & RBAC validated |  | Secure access to monitoring UIs |
| Documentation updated |  | Include all configurations & diagrams |
| Team trained & aware |  | Handed off to support personnel |

**Troubleshooting and Maintenance**

* **PVC issues**: Check volume claims, storageclass compatibility, pod logs.
* **Image pull failures**: Verify secrets, registry accessibility.
* **No metrics in Grafana**: Inspect Prometheus and Thanos data sources.
* **Alerts not firing**: Verify alert rules and Alertmanager configuration.
* **Ingress 404s or cert errors**: Validate ingress hosts and TLS certs.

**References & Further Reading**

* [Rancher Monitoring Chart GitHub](https://github.com/rancher/charts/tree/dev-v2.12/charts/rancher-monitoring)
* [Prometheus Documentation](https://prometheus.io/docs/)
* [Thanos Official Documentation](https://thanos.io/tip/)
* [Grafana Official Documentation](https://grafana.com/docs/)
* [Kubernetes Persistent Volumes](https://kubernetes.io/docs/concepts/storage/persistent-volumes/)
* [Certified Kubernetes Administrator (CKA) Guide](https://training.linuxfoundation.org/certification/certified-kubernetes-administrator-cka/)
* [Certified Kubernetes Security Specialist (CKS) Guide](https://training.linuxfoundation.org/certification/certified-kubernetes-security-specialist-cks/)

This documentation adheres to production-grade standards essential for a successful PTO, ensuring a scalable, secure, and observable Rancher Monitoring installation in fully airgapped environments.