

Getting Started With Rancher

MATTHEW MATTOX
PRINCIPAL SUPPORT ENGINEER, SUSE

CONTENTS

- What Is Rancher?
- Getting Started
 - Creating a RKE Cluster
 - Creating a K3s Cluster in Single-Node Mode
 - Installing Rancher on a RKE or K3s Cluster
 - Building a Downstream Cluster
- Configurations
 - Etcd Backups
 - Monitoring and Alerting
 - Logging
 - Compliance
 - Hardening a Cluster
 - Operational Backups
- Conclusion

What is Rancher? And how does it make Kubernetes crazy easy? Rancher is a complete Kubernetes stack that's easy to navigate — whether it's physical servers on-prem, VMs in the cloud, hosted Kubernetes clusters like EKS or GKE, and even on the edge. Rancher allows you to use open-source tools on a unified platform. For example, the same code deployed to a K3s cluster running on Raspberry Pis also deploys to any public and private cloud, including hybrid deployment.

This Refcard helps you get started with Rancher — from zero to fully production-ready. Finally, we'll cover the Day-2 operations, including managing your infrastructure, monitoring your applications, collecting logs, enforcing security policies, and protecting your data from disaster.

WHAT IS RANCHER?

Rancher is primarily a management and organization platform for Kubernetes clusters at scale. Rancher not only can deploy Enterprise Kubernetes on-prem using physical hardware or VMware's vSphere but also orchestrate any certified Kubernetes clusters, including Amazon's EKS, Google's GKE, Microsoft's AKS, etc., along with providing a unified platform. Whether it's a Raspberry Pi cluster sitting on your desk or an RKE cluster running on physical servers in your data center, or even a complete PaaS solution in AWS.

GETTING STARTED

The Rancher server is built on Kubernetes and runs as an application on any certified Kubernetes cluster, and, of course, Rancher is 100% open source with no license keys. Providing the primary controller

for managing downstream clusters, the Rancher server also provides access to your downstream clusters in a standardized web UI and API. Rancher is primarily deployed on two types of clusters, RKE and K3s. RKE is mainly used in more traditional data centers and cloud deployments, and K3s are primarily used in more edge and developer laptop deployments.

RKE (RANCHER KUBERNETES ENGINE)

RKE is a CNCF-certified Kubernetes distribution that runs entirely within Docker containers. It solves the common frustration of installation complexity with Kubernetes by removing most host dependencies and presenting a stable path for deployment, upgrades, and rollbacks. As long as you can run a supported Docker version, you can deploy and run Kubernetes with RKE.



TrilioVault for Kubernetes
DATA PROTECTION FOR SUSE RANCHER

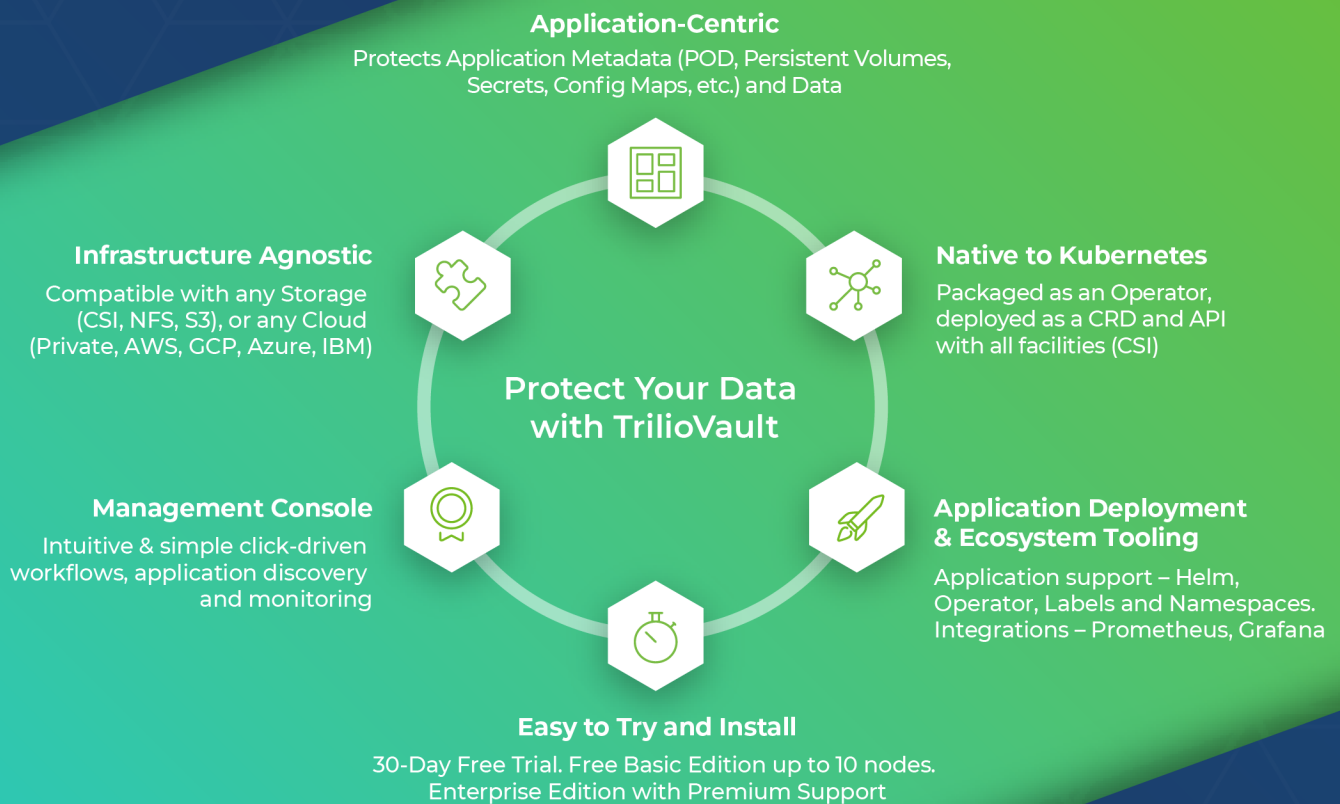
Get Started for Free

TRILIOVAULT | SUSE RANCHER

TrilioVault for Kubernetes

CLOUD-NATIVE DATA PROTECTION FOR SUSE RANCHER

TrilioVault for Kubernetes enables IT managers, administrators and developers to better manage their clouds with enterprise-grade, cloud-native backup and recovery for data and metadata.



Try TrilioVault for Free today!

Visit www.trilio.io

TRILIO  VAULT

 SUSE
RANCHER

K3S (5 LESS THAN K8S)

K3s is a lightweight certified Kubernetes distribution. All duplicate, redundant, and legacy code is removed and baked into a single binary that is less than 40MB and contains everything needed to run a Kubernetes cluster. This includes **etcd**, **traefik**, and all Kubernetes components. It is designed to run resource-constrained, remote locations, or inside IoT appliances. K3s have also been built to support ARM64 and ARMv7 nodes fully, so they can even be ran on a Raspberry Pi.

CREATING A RKE CLUSTER

REQUIREMENTS

Three Linux nodes with the following minimum specs:

- 2 vCPUs
- 8GB of RAM
- 20GB of SSD storage

INSTALLING DOCKER

You can either follow the Docker installation instructions or use Rancher's **install** scripts to install Docker.

Commands:

```
curl https://releases.rancher.com/install-docker/20.10.sh |sudo bash.
```

INSTALLING THE RKE BINARY

From your workstation or management server, download the current [latest RKE release](#).

Commands:

```
cd /tmp
wget https://github.com/rancher/rke/releases/download/v1.2.8/rke_linux-amd64
chmod +x rke_linux-amd64
sudo mv rke_linux-amd64 /usr/local/bin
```

INSTALLING THE KUBECTL BINARY

From your workstation or management server, download the current [latest kubectl release](#).

Commands:

```
cd /tmp
curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl
chmod +x kubectl
sudo mv kubectl /usr/local/bin/kubectl
```

CREATING THE CLUSTER CONFIG CONFIGURATION

RKE uses a **cluster.yml** file to define the nodes in the cluster and what roles each node should have. With three different roles that a node can have, the first is the **etcd** plane, the database for Kubernetes, and this role should be deployed in a HA configuration with an odd number of nodes and the default size of three nodes.

A five-member **etcd** cluster is the largest recommended size due to write performance suffering at scale. The second role being the control plane, which hosts the Kubernetes controllers and other related management services, should be deployed in a HA configuration with a minimum of two nodes.

Note: The control plane doesn't scale horizontally very well and scales more vertically.

The final role is the worker plane, which hosts your applications and related services. Nodes can support multiple roles, and in the default Rancher configuration, we'll be building a three-node cluster with all nodes running all roles.

Example **cluster.yml** file:

```
nodes:
- address: 1.1.1.1
  user: root
  hostname_override: node01
  role: [controlplane,worker,etcd]
- address: 2.2.2.2
  user: root
  hostname_override: node02
  role: [controlplane,worker,etcd]
- address: 3.3.3.3
  user: root
  hostname_override: node03
  role: [controlplane,worker,etcd]
```

For more examples, check out the [Rancher documentation](#).

CREATING THE CLUSTER

After creating the **cluster.yml**, we need to run the command **rke** to build the cluster using the following steps:

1. Create an SSH tunnel to each node for Docker CLI access.
2. Generate SSL certificates for all the different Kubernetes components.
3. Create the **etcd** plane and config all the **etcd**-related services.
4. Create the control plane, which includes **kube-apiserver**, **kube-controller-manager**, and **kube-scheduler**.
5. Create the worker plane and join all the nodes to the cluster.

Once these steps are done, RKE will create the file `cluster.rkestate`; this file contains credentials and the current state of the cluster. RKE will also create the file `kube_config_cluster.yml`; this file is used by `kubectl` to access the cluster. To make access more manageable, we'll want to copy this file to `kubectl`'s config directory.

Commands:

```
mkdir -p ~/.kube/
cp kube_config_cluster.yml ~/.kube/config
Verify access:
kubectl get nodes
```

Example output:

```
mmtatx@a1wnthorp01:~$ kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
a1ubrancherp01     Ready    controlplane,etcd,worker   193d   v1.20.6
a1ubrancherp02     Ready    controlplane,etcd,worker   190d   v1.20.6
a1ubrancherp03     Ready    controlplane,etcd,worker   190d   v1.20.6
mmtatx@a1wnthorp01:~$
```

CREATING A K3S CLUSTER IN SINGLE-NODE MODE

REQUIREMENTS

One Linux node with the following minimum specs:

- 2 vCPUs
- 4GB of RAM
- 10GB of SSD storage

INSTALLING K3S

While SSH into the K3s node, we'll run the following commands:

```
sudo su -
curl -sfL https://get.k3s.io | sh -
Verify access:
k3s kubectl get node
```

INSTALLING RANCHER ON A RKE OR K3S CLUSTER

REQUIREMENTS

- Kubectl access to the cluster
- Helm installed on the workstation or management server

Note: For K3s clusters, update the command "kubectl" to "k3s kubectl".

INSTALLING THE HELM BINARY

From your workstation or management server, download the [latest helm release](#).

Commands:

```
sudo su -
curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash
```

CONFIGURING HELM

Using the command `helm repo add`, we'll add the Rancher charts to helm:

```
helm repo add rancher-latest https://releases.rancher.com/server-charts/latest
helm repo add jetstack https://charts.jetstack.io
```

INSTALLING CERT-MANAGER

`Cert-manager` will manage the SSL certificates for Rancher:

```
kubectl apply --validate=false -f https://github.com/jetstack/cert-manager/releases/download/v1.0.4/cert-manager.crds.yaml
kubectl create namespace cert-manager
helm install cert-manager jetstack/cert-manager --namespace cert-manager --version v1.0.4
```

Please see the `cert-manager`'s [documentation](#) for more details.

INSTALLING RANCHER

We're now going to install Rancher using the default settings and following commands:

```
kubectl create namespace cattle-system
helm install rancher rancher-latest/rancher --namespace cattle-system --set hostname=rancher.example.com
```

CONFIGURING DNS FOR A SINGLE NODE

In single-node mode, DNS is optional, and the node IP/Hostname can be used in place of the Rancher URL.

CONFIGURING THE FRONT-END LOAD BALANCER FOR HA

To provide a HA setup for Rancher, we'll want to create a Layer-4 (TCP mode) or Layer-7 (HTTP mode) load balancer for ports 80 and 443 sitting in front of and forwards traffic to all nodes in the cluster. The DNS record for the Rancher URL should be pointed at the load balancer.

For more details, please see Rancher's [documentation](#).

BUILDING A DOWNSTREAM CLUSTER

Downstream clusters in Rancher are RKE/RKE2/K3s clusters that Rancher manages for you. They can also be clusters that are built outside Rancher then imported. In this example, we'll be making a standard three-node with all nodes running all roles.

REQUIREMENTS

Three Linux nodes with the following minimum specs:

- 2 vCPUs
- 4GB of RAM
- 20GB of SSD storage

INSTALLING DOCKER ON ALL NODES

You can either follow these [Docker installation](#) instructions or use [Rancher's install scripts](#) to install Docker.

Example:

```
curl https://releases.rancher.com/install-docker/20.10.sh | sudo bash
```

CREATING THE CLUSTER IN THE RANCHER UI

1. From the **Clusters** page, click **Add Cluster**.
2. Choose **Custom**.
3. Enter a **Cluster Name**.
Note: This can be changed at a later date.
4. Click **Next**.
5. From **Node Role**, choose the roles that you want to be filled by a cluster node. You must provision at least one node for each role: **etcd**, **worker**, and **control plane**. In this example, we'll select all three roles.
6. Copy the command displayed on-screen to your clipboard.

ADDING THE NODES TO THE CLUSTER

We'll want to run the previous command on each node. Then once all three nodes have joined successfully, the cluster should be in an active state.

CONFIGURATIONS

ETCD BACKUPS

Snapshots of the **etcd** database can be taken and saved locally or to S3. Etcd backups are used to back up the state of the Kubernetes cluster. This backup includes all the **deployments**, **secrets**, and **configmaps** for the cluster.

Note: This does not have backups for any application volumes being used in the cluster. You'll need a third-party tool to back up your application data.

CONFIGURING LOCAL ETCD BACKUPS

1. From the **Clusters** page, click **Edit**.
2. Fill in the "etcd Snapshot Backup Target" section.
3. Click **Save**.

CONFIGURING S3 ETCD BACKUPS

1. From the **Clusters** page, click **Edit**.
2. Fill in the "etcd Snapshot Backup Target" section.
3. Click **Save**.

MONITORING AND ALERTING

Rancher is powered by [Prometheus](#), [Grafana](#), [Alertmanager](#), the [Prometheus Operator](#), and the [Prometheus adapter](#).

This monitoring stack allows you to:

- Monitor the state of your cluster, node, and Kubernetes components.
- Create custom dashboards to make it easy to visualize collected metrics via Grafana
- Configure alert-based notifications via Email, Slack, PagerDuty, etc. using Alertmanager.

INSTALLING MONITORING

1. From the **Cluster Explorer** page, select **Apps & Marketplace**.
2. Select **Monitoring** from the catalog.
3. Click **Install**.

ACCESS GRAFANA

From the **Cluster Explorer** page, select **Monitoring**.

COMPLIANCE

INSTALLING OPA GATEKEEPER

1. From the **Cluster Explorer** page, select **Apps & Marketplace**.
2. Select **OPA Gatekeeper** from the catalog.
3. Click **Install**.

CONFIGURING CONSTRAINTS

OPA Gatekeeper constraints are a set of policies that allow or deny particular behavior in a Kubernetes cluster. Below are some example policies that I usually recommend applying:

- Only images from a private Docker registry: <https://support.tools/post/opa-gatekeeper-allow-images-from-private-registry/>
- Require that the namespace have an owner label: <https://support.tools/post/opa-gatekeeper-require-labels/>

HARDENING A CLUSTER

By default, Kubernetes can be vulnerable to numerous security issues, including privilege escalation, allowing users to gain root access to the Kubernetes host servers. To address this issue, Rancher created a guide with a number of setting changes to lock down a cluster.

CONFIGURATION STEPS

Check out these [instructions](#) for hardening a production installation of a RKE cluster with Rancher.

INSTALLING CIS BENCHMARK

1. From the **Cluster Explorer** page, select **Apps & Marketplace**.
2. Select **CIS Benchmark** from the catalog.
3. Click **Install**.

CONFIGURING THE CIS SCANS

To verify the cluster hardening was applied correctly and hasn't changed, we configure a scheduled scan using this [guide](#).

OPERATIONAL BACKUPS

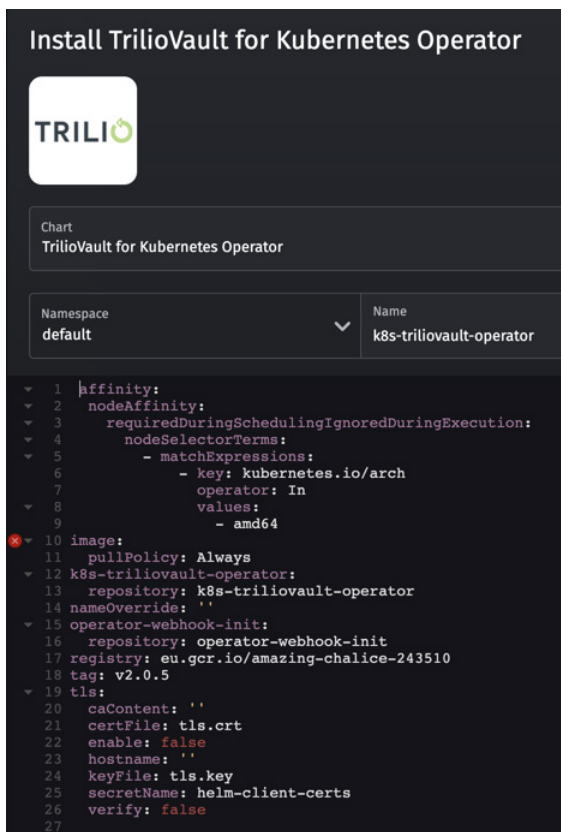
By default, Rancher clusters have a scheduled backup job that takes an `etcd` backup every 12 hours. But this is only backing up the `etcd` database and not backing up any volume data. It's also designed to restore a whole cluster without restoring individual objects and rolling the whole cluster back. This is where a third-party tool can be used to take volume and object-level backups.

For more details on the Rancher `etcd` backup, please see this [documentation](#).

INSTALLATION STEPS

To install a third-party data protection tool, like TrilioVault for example, on a Rancher cluster, we'll want to follow the official tool [install guide](#).

See example below.



The screenshot shows the installation configuration for TrilioVault for Kubernetes Operator. The interface includes a chart name, namespace selection (default), and a list of values to be set. The values are as follows:

```

1 affinity:
2   nodeAffinity:
3     requiredDuringSchedulingIgnoredDuringExecution:
4       nodeSelectorTerms:
5         - matchExpressions:
6           - key: kubernetes.io/arch
7             operator: In
8             values:
9               - amd64
10 image:
11   pullPolicy: Always
12 k8s-triliovault-operator:
13   repository: k8s-triliovault-operator
14   nameOverride: ''
15 operator-webhook-init:
16   repository: operator-webhook-init
17   registry: eu.gcr.io/amazing-chalice-243510
18   tag: v2.0.5
19 tls:
20   caContent: ''
21   certFile: tls.crt
22   enable: false
23   hostname: ''
24   keyFile: tls.key
25   secretName: helm-client-certs
26   verify: false
27

```

RESTORE STEPS

We'll want to follow the example application to deploy a WordPress site with a MySQL database with an attached volume. See [here](#).

Then, to kick off a restore, we'll need to create a restore job that can be on the same cluster or restored on a different cluster (Great of a DR plan) following [these steps](#).

CONCLUSION

This getting started with Rancher Refcard provides a step-by-step guide for installing Rancher, addressing standard Day-2 tasks and making your Kubernetes cluster production-ready.

ADDITIONAL DOCUMENTATION AND GUIDES

- This repo has several Kubernetes Masterclasses, covering a range of topics: <https://github.com/mattmattox/Kubernetes-Master-Class>
- The official Kubernetes documentation: <https://kubernetes.io>
- This is the Unofficial Kubernetes documentation, which goes into a lot more detail than the official documentation: <https://unofficial-kubernetes.readthedocs.io>
- Rancher's Official Documentation: <https://rancher.com/docs/>
- Rancher's Knowledge: <https://support.rancher.com/hc/en-us>

WRITTEN BY MATTHEW MATTOX,
 PRINCIPAL SUPPORT ENGINEER, SUSE



Matthew Mattox is a Principal Support Engineer at SUSE who specializes in providing customer-focused support. Matthew has experience in both Engineering and DevOps and in-depth knowledge of Kubernetes, Docker, and the surrounding ecosystem — Rancher, Longhorn, and the OPA Gatekeeper. Along with designing custom solutions to solve every changing problem, his most recent award was "Bullfighter of the Year", praising excellence in his support at Rancher Labs. His number one goal: IT should be a profit center within your company, not a hole you throw money into.



DZone, a Devada Media Property, is the resource software developers, engineers, and architects turn to time and again to learn new skills, solve software development problems, and share their expertise. Every day, hundreds of thousands of developers come to DZone to read about the latest technologies, methodologies, and best practices. That makes DZone the ideal place for developer marketers to build product and brand awareness and drive sales. DZone clients include some of the most innovative technology and tech-enabled companies in the world including Red Hat, Cloud Elements, Sensu, and Sauce Labs.

Devada, Inc.
 600 Park Offices Drive
 Suite 150
 Research Triangle Park, NC 27709
 888.678.0399 | 919.678.0300

Copyright © 2021 Devada, Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means of electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.