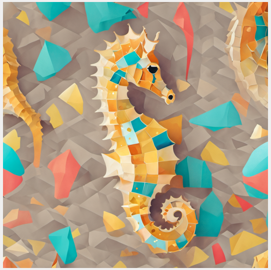
PVD903 Workshop RKE2

Pascal van Dam

November 2, 2023



Introduction Distro Galore

K8S Distro Galore

ACCORDION
AISTATION
ALVISTACK
aws
Microsoft Azure
bizkube
CANONICAL
CARS
cisco
Cocktail Cloud
Constellation
D2IQ
DaoCloud
desktop
DIAMANTI
EasyStack
elastisys
ERICSSON
Flexkube
FURY
Giant Swarm
GienTech
H3C
Hewlett Packard Enterprise
inspur
Intel Smart Edge Open
JD.COM
K0S
K3S
K8sPlus
Kamaji
KubeCube
KUBERMATIC
KUBESPHERE
Kubic
kublr
kurl
METALK8S
MicroK8s
MIRANTIS
ORACLE
palette
PLATFORM9
qbo
RANCHER
Red Hat OpenShift
RKE
ROBIN
TmaxA&C
Typhoon
vcluster
VMware Tanzu
whitestack
WNDRVR
Alibaba Cloud
aws
Baidu AI CLOUD
BIZMICRO
catalyst cloud
中国移动 China Mobile
China unicom
CIVO
CloudLink
DigitalOcean

K8S Distro Galore Series

Study and showcase of a pletora of K8S distributions:

- Kubeadm
- RKE2
- K3S
- K0S
- Kind
- Minikube
- K3D
- MicroK8S
- Charmed K8S

K8S Distro Galore Series

K8S in the large

- Mirantis Kubernetes Engine (MKE)
- Rancher
- OpenShift + MicroShift
- VMware Tanzu

K8S Distro Galore Series

Hosted K8S services

- AKS
- EKS
- GKE
- DOKS
- Alibaba Cloud Container Service for Kubernetes
- OKE

Introduction

About the Trainer

- Pascal van Dam, living in Nieuw Bergen (Limburg/NL)
- Owner of Poortier Management B.V / PASCALVANDAM.COM
- Trainer & Consultant Open-Source Solutions:
  - Kubernetes & Containers
  - Virtualization & Cloud
  - Go, Rust, NodeJS, C, C++, Perl
  - Cloud Automation & Orchestration
  - CI/CD Argo, Flux, Gitlab
  - Linux Kernel Internals

Pascal Van Dam

“Let us orchestrate your success!" #K8SMastery

About this workshop

- Introduction to RKE2 distro

Agenda

Agenda 1/2

Part I

- Introduction to RKE2
- Features
- Architecture
- Simple Installation
- Supported platforms
- Coffee break

Agenda 2/2

Part II

- RKE2 Add-ons
- Customization
- High available install
- RKE2 and Security
- Airgapped install
- RKE2 FIPS and CIS
- Conclusion
- Questions and Answers
- Next on PASCALVANDAM.COM

RKE2 Introduction

RKE2 Introduction

Facts:

- RKE2 - Rancher Kubernetes Engine 2
- Also known by RKE Government
- CNCF certified K8S distro
- Now owned and managed by SUSE

Origin

- RKE - Rancher Kubernetes Engine
- K3S - Kubernetes for Edge Computing
- Github: https://github.com/rancher/rke2
- As of Oct 2023 - ranking 1.1K stars
- Now owned and managed by SUSE

RKE2 Features

RKE2 FEATURES

- Simplified installation
- Security included (CIS/FIPS compliant)
- Based on containerd CRI/CRE
- Server/agent architecture
- Automatic upgrades possible
- Customizable
- Support for airgapped installs
- Ad-hoc and sched ETCD snapshots

Addons included in the box

Addons included out of the box

- Ingress-nginx
- Canal CNI
- Helm controller
- Metrics server

Add yourselves

What to add?

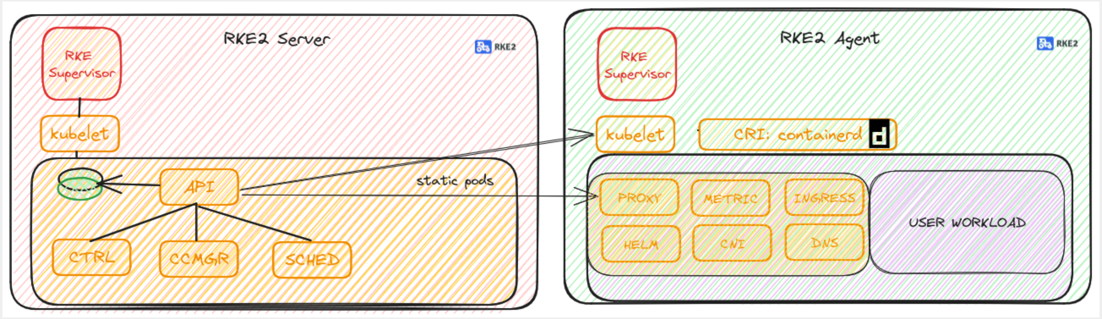
- Kubectl
- Storage provisioner
- Optional: node exporter
- Optional: Fluentd/fluent-bit/promtail

RKE2 Architecture

RKE2 Architecture

- Controlplane is composed of `servers`
- Workers are composed of `agents`
- Servers and agents are controlled by `systemd`
- RKE2 install binary for severs and agents is a static GO binary

RKE2 Architecture



RKE2 Single Controlplane Installation

Requirements RKE2 cluster

Pre-reqs:

- 1x Server for RKE2 server node on supported platform/OS
- 1x Server for RKE2 agent node on supported platform/OS
- Internet connection to download RKE2 binaries
- At least SUDO to root privileges for our install user

Install procedure single controlplane RKE2 cluster

Steps:

1. Install and configure RKE2 binary for server
2. Take note of the join token for new agents/servers
3. Create `config.yaml` with node token for agents
4. Install and configure RKE2 binary for agents
5. Install kubectl
6. Validate the cluster

Install procedure RKE2 server

On the RKE2 server, install and start RKE2 server.

code/rke2-server/rke2-server-sc.sh

```
# Step 1: Download and install the RKE2 server binary

curl -sfL https://get.rke2.io | sudo sh -

# Step 2: Enable and start rke2-server service to configure RKE2 server node

sudo systemctl enable rke2-server --now

# Step 3: Optionally verify RKE2 server logs with:

sudo journalctl -u rke2-server -lf

# Step 4: When install has finished copy kube config file

mkdir -p ~/.kube
sudo cp /etc/rancher/rke2/rke2.yaml ~/.kube/config
sudo chown ${USER}:${USER} ~/.kube/config

# Step 5: Retrieve and record the node-token for future node joins

sudo cat /var/lib/rancher/rke2/server/node-token
```

RKE2 server: Install and configure kubectl

On the RKE2 server install and configure kubectl

</> code/rke2-server/install-kubectl.sh </>

```sh
#!/bin/sh

ARCH="$(uname -m)"

case "$ARCH" in
    x86_64)
        ARCH="amd64"
        ;;
    aarch64)
        ARCH="arm64"
        ;;
esac

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/${ARCH}/kubectl"
sudo mv kubectl /usr/local/bin
sudo chmod +x /usr/local/bin/kubectl

kubectl get nodes
```

RKE2 SERVER: CONFIGURE AGENT CONFIG.YAML

On the RKE2 server create a file called `agent-config.yaml` with the following content:

code/rke2-server/agent-config-model.yaml

```yaml
server: https://<rke2-server-hostname>:9345
token: <node-token>
```

The `node-token` is copied from step 5 of the RKE2 server install, e.g:

code/rke2-server/agent-config.yaml

```yaml
server: https://k8sc903n01:9345
token: K10da6206e5e8b884c5c3e486349fdc26ceb6019297e088c276f36d83e3ed545418::server:f2e1c6a9e85c0d5e18ed977c2fa90983
```

Install procedure RKE2 agent

On the RKE2 agent:

code/rke2-agent/rke2-agent.sh

```
# Download and install the RKE2 agent binary

curl -sfL https://get.rke2.io | sudo INSTALL_RKE2_TYPE="agent" sh -

# Copy the agent-config.yaml file from rke2-server to rke2-agent's /etc

sudo mkdir -p /etc/rancher/rke2
sudo cp agent-config.yaml /etc/rancher/rke2/config.yaml
sudo systemctl enable rke2-agent.service --now

# Enable and start rke2-agent service to configure RKE2 agent node

sudo systemctl enable rke2-agent.service --now

# Optionally verify RKE2 agent logs with:

sudo journalctl -u rke2-agent -lf
```

You can add more rke2-agents this way, with the same token/config.yaml file.

Validate RKE2 cluster

On the RKE2 server node:

code/rke2-server/rke2-validate.sh

```
kubectl get nodes
NAME          STATUS   ROLES                         AGE     VERSION

k8sc903n01    Ready    control-plane,etcd,master    2d23h   v1.28.1+rke2r1
k8sc903n02    Ready    control-plane,etcd,master    2d23h   v1.28.1+rke2r1
k8sc903n03    Ready    control-plane,etcd,master    2d23h   v1.28.1+rke2r1
k8sc903n04    Ready    <none>                        2d23h   v1.28.1+rke2r1

kubectl get pods -n kube-system
```

RKE2 SUPPORTED PLATFORMS

RKE2 SUPPORTED PLATFORMS

- Operating Systems
  - Linux (server- and agent nodes)
  - Windows (agent nodes only, experimental)
- CPU architectures
  - AMD64 (x86_64)
  - ARM64 (aarch64)

RKE2 Add-ons

RKE2 Add-ons

- CNI Canal
- Ingress NGINX
- Metric server
- Helm Controller

RKE2 Add-on: CNI Canal

- CANAL is Calico piggy backed on FLANNEL
- Workers virtually everywhere (no IPinIP req)
- Network policies from Calico available
- Automatically configured (IP Pools etc)
- Can be switched for another CNI
- Installed in `kube-system` namespace
- Upgraded with RKE2 upgrades

RKE2 Add-on: Ingress-nginx

- Default installed
- Additional Cert manager can be installed (helm)
- Additional Ingress controllers can be installed
- Installed in `kube-system` namespace
- Upgraded with RKE2 upgrades

Helm controller

- Helm charts are submitted to the controller using YAML
- Values file is submitted to the controller using YAML
- Installs helm-charts using a controller
- Used to install add-ons and extras on RKE2
- Installed in `kube-system` namespace
- Upgraded with RKE2 upgrades
- See also next RKE2 Customization chapter

RKE2 Customization

RKE2 Customization

There are 3 ways to customize RKE2 installs:

- Configure install by setting ENV VARs
- Configure install using YAML config files
- Add add-ons to RKE2 using the helm-controller

RKE2 CUSTOMIZATUON USING EXECUTABLE OPTIONS

You can configure RKE2 install using ENV VARS:

</> code/rke2-server/rke2-server-cust-env.sh </>

```
# Ex1: To install a specific version of RKE2/K8S set INSTALL_RKE2_VERSION ENV VAR
#      This needs to be done for ALL node installs, servers AND agents!

curl -sfL https://get.rke2.io | sudo INSTALL_RKE2_VERSION="v1.28.1+rke2r1"  sh -

# Ex2: To install the latest version from a RKE2 channel set INSTALL_RKE2_CHANNEL
# #    Channels available are: stable, tesing and latest
#      This needs to be done for ALL node installs, servers AND agents!


curl -sfL https://get.rke2.io | sudo INSTALL_RKE2_CHANNEL="testing"  sh -
```

RKE2 CUSTOMIZATUON USING CONFIG.YAML

You can configure RKE2 install by placing a config.yaml file in /etc/rancher/rke2 directory:

code/rke2-server/rke2-server-custom-config.yaml

```yaml
write-kubeconfig-mode: "0644"
tls-san:
  - "knoobz.org"
node-label:
  - "managedby=pascalvandam.com"
debug: true
system-default-registry: priv-sysreg.knoobz.org
private-registry: priv-reg.knoobz.org
```

RKE2 Customization using config.yaml

You can configure RKE2 install by placing a config.yaml file in /etc/rancher/rke2 directory:

code/rke2-server/rke2-server-custom.sh

```
curl -sfL https://get.rke2.io | sudo sh -

# Create directory and copy custom config file to /etc/rancher/rke2 directory
# prior to starting up rke2-server or rke2-agent
#
sudo mkdir -p /etc/rancher/rke2
sudo cp rke2-server-custom-config.yaml /etc/rancher/rke2/config.yaml

# Enabling and starting RKE2

sudo systemctl enable rke2-server.service --now
```

RKE2 Customization using Helm Controller

You can install and configure add-ons using helm-charts and the helm controller. This example will install a configure the fluent-bit daemonset in the `tools` namespace and forward the logs to ES instance logger:9200

code/rke2-server/fluent-bit.yaml

```yaml
apiVersion: helm.cattle.io/v1
kind: HelmChart
metadata:
  name: fluent-bit
  namespace: kube-system
spec:
  chart: fluent-bit
  repo: https://fluent.github.io/helm-charts
  targetNamespace: tools
  valuesContent: |
    backend:
      type: es
      es:
        host: logger
        port: 9200
```

RKE2 Customization using Helm Controller

Copy the CRD describing the helm-chart and providing the config to it in the proper directory:

code/rke2-server/install-rke2-server-helmcustom.sh

```
# Download the RKE2 binary for the RKE2 server

curl -sfL https://get.rke2.io | sudo sh -

# Create directory for the HELM charts
#
sudo mkdir -p /var/lib/rancher/rke2/server/manifests

# Copy the fluent-bit.yaml CRD for the Helm controller in RKE2

sudo cp fluent-bit.yaml /var/lib/rancher/rke2/server/manifests

# Enable and start the RKE2 server, the controller will bootstrap the HELM
# charts provided in /var/lib/rancher/rke2/server/manifests

sudo systemctl enable rke2-server.service --now
```

RKE2 Backup and Restore

Backing up a RKE2 cluster

Backing up RKE2

- All state is in the ETCD database
- RKE2 provides a built-in ETCD snapshotter
- Snapshots can be created ad-hoc or scheduled

Backing up a RKE2 cluster

code/rke2-server/ad-hoc-snapshot-etcd.sh

```
# Making an ad-hoc snapshot of RKE2's ETCD
sudo /usr/local/bin/rke2 etcd-snapshot save
INFO[0000] Managed etcd cluster bootstrap already complete and initialized
INFO[0000] Applying CRD helmcharts.helm.cattle.io
INFO[0000] Applying CRD helmchartconfigs.helm.cattle.io
INFO[0000] Applying CRD addons.k3s.cattle.io
INFO[0000] Creating rke2-supervisor event broadcaster
INFO[0000] Saving etcd snapshot to /var/lib/rancher/rke2/server/db/snapshots/on-demand-k8sc904n01-1698503438
INFO[0000] Reconciling etcd snapshot data in rke2-etcd-snapshots ConfigMap
INFO[0000] Reconciliation of snapshot data in rke2-etcd-snapshots ConfigMap complete

# Listing available ETCD snapshots
pascal@k8sc904n01:~$ sudo /usr/local/bin/rke2 etcd-snapshot ls
Name                                Location                                                                             Size    Created
etcd-snapshot-k8sc904n01-1698487203 file:///var/lib/rancher/rke2/server/db/snapshots/etcd-snapshot-k8sc904n01-
↪   1698487203 9699360 2023-10-28T12:00:03+02:00
on-demand-k8sc904n01-1698503438     file:///var/lib/rancher/rke2/server/db/snapshots/on-demand-k8sc904n01-
↪   1698503438     9699360 2023-10-28T16:30:38+02:00
etcd-snapshot-k8sc904n01-1698314403 file:///var/lib/rancher/rke2/server/db/snapshots/etcd-snapshot-k8sc904n01-
↪   1698314403 9699360 2023-10-26T12:00:03+02:00
etcd-snapshot-k8sc904n01-1698444003 file:///var/lib/rancher/rke2/server/db/snapshots/etcd-snapshot-k8sc904n01-
↪   1698444003 9699360 2023-10-28T00:00:04+02:00
```

Scheduling RKE2 ETCD snapshots

Scheduling snapshot creation of RKE2's ETCD. Default each 12h a snapshot is created:

code/rke2-server/etcd-snapshot.yaml

```yaml
tls-san:
  - "knoobz.org"
node-label:
  - "managedby=pascalvandam.com"
debug: true
etcd-snapshot-schedule-cron: "0 */4 * * *"
```

Restoring RKE2 ETCD snapshots

Restoring an RKE2 ETCD snapshot

code/rke2-server/restore-snapshot-etcd.sh

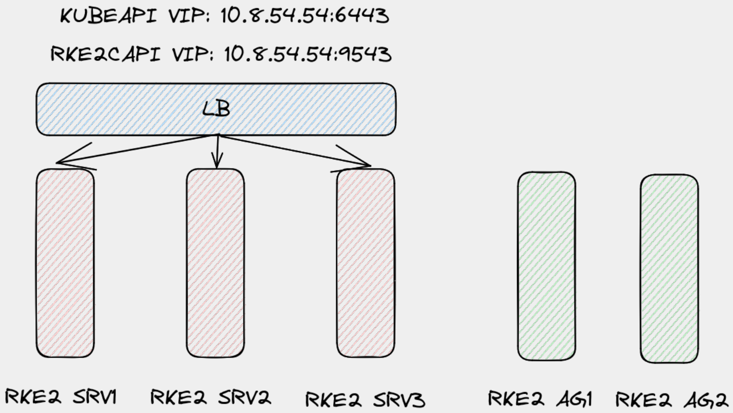
```
# Restoring an ETCD snapshot for RKE2 on existing NODEs

# Step 1: Stop RKE2 server on ALL server nodes
sudo systemctl stop rke2-server

# Step 2: On 'first' server node, restore ETCD snapshot
sudo rke2 server \
  --cluster-reset \
  --cluster-reset-restore-path=<PATH-TO-SNAPSHOT>

sudo systemctl start rke2-server
# On the OTHER server nodes prior to (re)starting rke2-server remove
# the ETCD data in /var/lib/rancher/rke2/server/db

sudo rm -f /var/lib/rancher/rke2/server/db/*
# Restarting the rke2-server service will start replication of the ETCD database
# from the 'FIRST' server.

sudo systemctl start rke2-server
```

High Available Controlplane Installation

High Available Controlplane

- Multiple RKE2-server nodes (min. 3)
- Single K8SAPI (port 6443) EndPoint using LB
- Single RKE2 CAPI (port 9345) EndPoint using LB
- Requirement: Hard or Software LoadBalancer
- Software LB: HAPROXY, NGINX, KUBEVIP

High Available Controlplane



Steps to install HA RKE2

1. Install and configure LB (HAPROXY)
2. Install and configure first RKE2 server
3. Install and configure second RKE2 server
4. Install and configure third RKE2 server
5. Install and configure any RKE2 agents

HAPROXY CONFIG EXAMPLE

HAPROXY config example front-ends:

code/rke2-server/haproxy.cfg

```
# K8S/RKE2 Master frontends
frontend k8s-api
    bind 0.0.0.0:6443
    mode tcp
    default_backend rke2-servers-6443

frontend rancher-ui-api
    bind 0.0.0.0:9345
    mode tcp
    default_backend rke2-servers-9345
```

HAPROXY CONFIG EXAMPLE

HAPROXY config example back-ends:

code/rke2-server/haproxy.cfg

```
# K8S/RKE2 Master backends
backend rke2-servers-6443
    mode tcp
    balance roundrobin
    option ssl-hello-chk
    server k3sc903n01 10.8.59.95:6443 check
    server k8sc903n02 10.8.59.140:6443 check
    server k8sc903n03 10.8.59.190:6443 check

backend rke2-servers-9345
    mode tcp
    balance roundrobin
    option ssl-hello-chk
    server k3sc903n01 10.8.59.95:9345 check
    server k8sc903n02 10.8.59.140:9345 check
    server k8sc903n03 10.8.59.190:9345 check
```

Create config file for HA controlplane install

Create a file called `rke2-server-ha-config.yaml` with the following content:

code/rke2-server/rke2-server-config-ha.yaml

```yaml
tls-san:
  - pascalvandam.com
  - k8sc903lb01
node-taint:
  - "CriticalAddonsOnly=true:NoExecute"
```

Install first RKE2 server of HA controlplane

code/rke2-server/rke2-ha-install.sh

```
# Download and install RKE2 server
curl -sfL https://get.rke2.io | sudo INSTALL_RKE2_VERSION=v1.28.1+rke2r1 sh -

# Copy the configured

sudo mkdir -p /etc/rancher/rke2
sudo cp rke2-server-config-ha.yaml /etc/rancher/rke2/config.yaml

sudo mkdir -p /var/lib/rancher/rke2/server/manifests
sudo systemctl enable rke2-server.service --now
mkdir -p ~/.kube
sudo cp /etc/rancher/rke2/rke2.yaml ~/.kube/config
sudo chown ${USER}:${USER} ~/.kube/config
echo "Servers and Agents can be joined with node-token: \c"
sudo cat /var/lib/rancher/rke2/server/node-token
echo
```

PREPARING FOR ADDITIONAL RKE2-SERVER NODES

To enable the other rke2-server nodes to join the leader we need to craft a special `rke2-join-server-config-ha.yaml`
This file needs to be copied to the other rke2-servers prior to install of rke2-server software.

code/rke2-server/rke2-join-server-config-ha.yaml

```yaml
server: https://k8sc903lb01:9345
token: K1066bf857b5cb1b9a40d111ace22fac1177a4bdc19e6424c2a678e0b4273fb8cf5::server:ff544d6ba9b39ac62a817199d4249e39
tls-san:
  - pascalvandam.com
  - k8sc903lb01
node-taint:
  - "CriticalAddonsOnly=true:NoExecute"
```

Have 2nd rke2-server join HA controlplane

To have the 2nd rke2-server node join the HA controlplane execute:

code/rke2-server/join-rke2-server-ha.sh

```
curl -sfL https://get.rke2.io | sudo INSTALL_RKE2_VERSION=v1.28.1+rke2r1 sh -

# Copy the HA config.yaml file from the first master
sudo mkdir -p /etc/rancher/rke2
sudo cp rke2-join-server-config-ha.yaml /etc/rancher/rke2/config.yaml
sudo systemctl enable rke2-server.service --now

# Copy the kubeconf file for kubectl
mkdir -p ~/.kube
sudo cp /etc/rancher/rke2/rke2.yaml ~/.kube/config
sudo chown ${USER}:${USER} ~/.kube/config
```

Have 3rd rke2-server node join the HA controlplane

Repeat for the 3rd rke2-server node:

</> *code/rke2-server/join-rke2-server-ha.sh* </>

```
curl -sfL https://get.rke2.io | sudo INSTALL_RKE2_VERSION=v1.28.1+rke2r1 sh -

# Copy the HA config.yaml file from the first master
sudo mkdir -p /etc/rancher/rke2
sudo cp rke2-join-server-config-ha.yaml /etc/rancher/rke2/config.yaml
sudo systemctl enable rke2-server.service --now

# Copy the kubeconf file for kubectl
mkdir -p ~/.kube
sudo cp /etc/rancher/rke2/rke2.yaml ~/.kube/config
sudo chown ${USER}:${USER} ~/.kube/config
```

Verify if all 3 rke2-servers online and ready

On the first rke2-server node:

code/rke2-server/kubectl-ha.out

```
kubectl get nodes

NAME          STATUS   ROLES                       AGE      VERSION
k8sc903n01    Ready    control-plane,etcd,master   16m12s   v1.28.1+rke2r1
k8sc903n02    Ready    control-plane,etcd,master   11m53s   v1.28.1+rke2r1
k8sc903n03    Ready    control-plane,etcd,master   4m8s     v1.28.1+rke2r1
```

Adding any RKE2-agent nodes to the HA controlplane

On the RKE2 server create a file called `agent-config.yaml` with the following content:

code/rke2-server/agent-config-model-ha.yaml

```yaml
server: https://<rke2-lb>:9345
token: <node-token>
tls-san:
  - «rke2-lb»
```

The `node-token` is copied from step 5 of the RKE2 server install, e.g:

code/rke2-server/agent-config-ha.yaml

```yaml
server: https://k8sc903lb01:9345
token: K1066bf857b5cb1b9a40d111ace22fac1177a4bdc19e6424c2a678e0b4273fb8cf5::server:ff544d6ba9b39ac62a817199d4249e39
tls-san:
  - pascalvandam.com
  - k8sc903lb01
```

Install procedure RKE2 agent

On the RKE2 agent:

code/rke2-agent/rke2-agent-ha.sh

```
# Download and install the RKE2 agent binary

curl -sfL https://get.rke2.io | sudo INSTALL_RKE2_TYPE="agent" INSTALL_RKE2_VERSION=v1.28.1+rke2r1 sh -

# Copy the agent-config.yaml file from rke2-server to rke2-agent's /etc

sudo mkdir -p /etc/rancher/rke2
sudo cp agent-config.yaml /etc/rancher/rke2/config.yaml
sudo systemctl enable rke2-agent.service --now

# Enable and start rke2-agent service to configure RKE2 agent node

sudo systemctl enable rke2-agent.service --now

# Optionally verify RKE2 agent logs with:

sudo journalctl -u rke2-agent -lf
```

You can add more rke2-agents this way, with the same token/config.yaml file.

Validate RKE2 cluster

On the RKE2 server node:

code/rke2-server/kubectl-ha-all.out

```
kubectl get nodes

NAME          STATUS   ROLES                        AGE      VERSION
k8sc903n01    Ready    control-plane,etcd,master    24m12s   v1.28.1+rke2r1
k8sc903n02    Ready    control-plane,etcd,master    19m53s   v1.28.1+rke2r1
k8sc903n03    Ready    control-plane,etcd,master    12m8s    v1.28.1+rke2r1
k8sc903n04    Ready    <none>                       3m41s    v1.28.1+rke2r1
k8sc903n05    Ready    <none>                       2m5s     v1.28.2+rke2r1

kubectl get pods -n kube-system
```

RKE2 Security

RKE2 SECURITY

- Provides out of the box near CIS-1.23 K8S compliancy
- Based on immutable infrastructure
- K8S components run in containers
- Deploy using full CIS-1.23 compliancy possible
- FIPS 140-2 Enablement (with Canal as CNI)
- Support for airgapped install
- Support for install behind PROXY

RKE2 Airgapped Install

Airgapped installs

Airgapped installs means install without any internet access (NO PROXY)
Pre-requisites:

- Pre-download specific tarball with RKE2 container images
- Or Private registry with RKE2 images of RKE2 release to be deployed
- Install binaries for RKE

Install RKE2 server in an airgapped environment using TARBALL

code/rke2-server/airgap-tarball.sh

```
# Create directory for rke2 artefacts

mkdir /root/rke2-artifacts && cd /root/rke2-artifacts/

# Download images and RKE2 install binary

curl -OLs https://github.com/rancher/rke2/releases/download/v1.28.1%2Brke2r1/rke2-images.linux-amd64.tar.zst
curl -OLs https://github.com/rancher/rke2/releases/download/v1.28.1%2Brke2r1/rke2.linux-amd64.tar.gz
curl -OLs https://github.com/rancher/rke2/releases/download/v1.28.1%2Brke2r1/sha256sum-amd64.txt
curl -sfL https://get.rke2.io --output install.sh

# Start the installer

INSTALL_RKE2_ARTIFACT_PATH=/root/rke2-artifacts sh install.sh

systemctl enable rke2-server.service --now
```

Preparation RKE2 server in an airgapped environment using priv-reg

- Deploy a private registry (Harbor, registry v2 etc)
- Copy relevant RKE2 images to it (Tip: `Skopeo`)
- Create `registries.yaml` file

code/rke2-server/registries.yaml

```yaml
mirrors:
  docker.io:
    endpoint:
      - "https://privreg-n01.spikweien08.nest"
configs:
  "privreg-n01.spikweien08.nest":
    auth:
      username: student01 # this is the registry username
      password: Welkom01 # this is the registry password
        #tls:
        #insecure_skip_verify:
```

Install RKE2 server(s) in an airgapped environment using priv-reg

Install RKE2 server in airgapped environment:

code/rke2-server/install-rke2-server-airgapped

```
HTTPS_PROXY=http://k8sc903lb01.spikweien08.nest:3128 curl -sfL https://get.rke2.io | \
   sudo HTTPS_PROXY=http://k8sc903lb01.spikweien08.nest:3128 INSTALL_RKE2_VERSION="v1.28.1+rke2r1" \
   sh -s -- --system-default-registry privreg-n01.spikweien08.nest

sudo mkdir -p /etc/rancher/rke2
sudo cp registries.yaml /etc/rancher/rke2

sudo systemctl enable rke2-server.service --now
mkdir -p ~/.kube
sudo cp /etc/rancher/rke2/rke2.yaml ~/.kube/config
sudo chown ${USER}:${USER} ~/.kube/config
echo "Agents can be joined with node-token:"
sudo cat /var/lib/rancher/rke2/server/node-token
echo
```

Install RKE2 agent in an airgapped environment using priv-reg

Install RKE2 agent(s) in airgapped environment:

</> *code/rke2-server/install-rke2-agent-airgapped* </>

```
HTTPS_PROXY=http://k8sc903lb01.spikweien08.nest:3128 curl -sfL https://get.rke2.io | \
   sudo HTTPS_PROXY=http://k8sc903lb01.spikweien08.nest:3128 INSTALL_RKE2_VERSION="v1.28.1+rke2r1"  INSTALL_RKE2_TYPE="agent" \
   sh -s -- --system-default-registry privreg-n01.spikweien08.nest

sudo mkdir -p /etc/rancher/rke2
sudo cp registries.yaml /etc/rancher/rke2
sudo cp rke2-join-agent-config.yaml /etc/rancher/rke2/config.yaml
sudo sysctl user.max_inotify_instances=1024
sudo systemctl enable rke2-agent.service --now
```

RKE2 FIPS and CIS

Hardened install of RKE2

Pre-requistes for CIS-1.23 installation:

- Host level setup:
  - Create `etcd:etcd` user group
  - Set hardened Kernel paramaters
- RKE2 setup
  - `profile: "cis-1.23"`
  - Post deploy configuration

Host level setup

Host level setup:

code/rke2-server/install-rke2-hardened-host

```
# To create the etcd:etcd user

sudo useradd -r -c "etcd user" -s /sbin/nologin -M etcd -U

# On TARBALL installed hosts:

sudo cp -f /usr/local/share/rke2/rke2-cis-sysctl.conf /etc/sysctl.d/60-rke2-cis.conf
sudo systemctl restart systemd-sysctl
```

Host level setup

This config.yaml needs to be copied/adapted prior to rke2-server and rke2-agent install:

code/rke2-server/config-hardened.yaml

```yaml
write-kubeconfig-mode: "0644"
profile: "cis-1.23"
tls-san:
  - "knoobz.org"
node-label:
  - "managedby=pascalvandam.com"
system-default-registry: priv-sysreg.knoobz.org
private-registry: priv-reg.knoobz.org
```

Post deploy hardening 1/4

Post deploy hardening: POD SECURITY POLICY

- Restricted policy for namespaces: `kube-system` and `cis-operator-system`
- For additionally created NS: operator must intervene

Post deploy hardening 2/4

Post deploy hardening: DEFAULT NETWORK SECURITY POLICY

- Default Network Security Policy: only intra namespace network conn. allowed
- Installed in NS: `kube-system`, `kube-public`, `kube-node-lease`, and `default`
- For additionally created NS: operator must act

Post deploy hardening 3/4

Post deploy hardening: Default Service Account

- Set `automountServiceAccountToken` to `false` for default service accounts
- Out of the box realized for namespaces: `kube-system` and `default`
- Operator must act for additionally created namespaces

Post deploy hardening 4/4

Post deploy hardening: API Server audit configuration

- By default API loggins is enabled but configured with `level: None`
- Adapt `/etc/rancher/rke2/audit-policy.yaml` and restart rke2-server(s)

Upgrading RKE2

Upgrading RKE2

Two different methods:

- By upgrading the server or agent ’packages’
- By using the Rancher Upgrade Controller

Upgrading RKE2 server and agent packages

- Upgrade using YUM/DNF/ZYPPER on RPM based systems
- Upgrade using curl/yar on other systems
- First upgrade servers
- Then upgrade agents

Upgrading RKE2 using Rancher Upgrade Controller

- Install the Rancher Upgrade Controller
- Write `plans` for upgrading server and agent nodes
- Enable upgrades to nodes by setting proper labels on nodes
- `kubectl label nodes <node-name-1> <node-name-2> ... rke.cattle.io/upgrade="true"`
- Watch the upgrade progress

Conclusions and Resume

RKE2 Resume: the Good

Resume: the Good

- Production ready
- Easy setup
- Rel. easy hardening
- Very customizable
- Hardened by default
- FIPS and CIS compliancy possible
- Install in Airgapped environment
- Support for ARM64/AARCH64 arch
- Integratetable with Rancher 2

RKE2 Resume: the Bad

Resume: the Bad

- Documentation lacks accuracy
- No RISCV64 support (but no ones has)

Question and Answers

- Questions?
- Vragen?
- Preguntas?
- Fragen?

Coming NEXT on PASCALVANDAM.COM

Next to be planned, watch our website!

- K3S
- K0S
- Programming in the large with Go