



From Pets to Cattle in your Smarthome!

Disclaimer

Since the workshop time is limited some parts will be using the UI to ease the progress and increase understanding

English

Who doesn't know it: Over the years, countless Raspberry Pis with various software have accumulated in your house, and then the little box was carefully stowed away somewhere (out of sight, out of mind) and never touched again. The operating system and applications continue to run there (in the same version) for years #Security!

Each team will receive an ARM VM or Raspberry Pi, and together we will tackle the big problems:

- What was the IP again, and my username/password?
- How do I install the operating system again?
- How do I keep my applications and operating system up to date?
- For this purpose, we will use Rancher, Fleet, k3s, and Elemental to solve all problems in a cloud-native manner.

But the question of Kubernetes on the Edge arises not only at home but also in the corporate environment, which is why demanding topics such as hardware tampering will be presented on the same stack in the end.

German

Wer kennt es nicht: Über die Jahre haben sich im Haus unzählige Raspberry Pis mit verschiedenster Software angesammelt und dann wurde die kleine Kiste irgendwo sorgfältig (aus den Augen aus dem Sinn) verstaut und nie wieder angefasst. Das Betriebssystem und die Applikationen laufen dort weiterhin (also in der selben Version) seit Jahren #Security!

Jedes Team erhält 1 Raspberry Pis oder eine ARM VM und wir gehen zusammen die großen Probleme an:

- Was war doch gleich die IP und mein Nutzernamen/Password?
- Wie installiere ich doch gleich nochmal das Betriebssystem?
- Wie halte ich meine Applikationen und mein Betriebssystem aktuell?

Wir werden hierfür Rancher, Fleet, k3s und Elemental verwenden, um alle Probleme cloud-native zu lösen.

Aber nicht nur zu Hause stellt sich die Frage von Kubernetes on the Edge, sondern auch im Unternehmensumfeld findet k3s mehr und mehr Zulauf, weswegen abschließend anspruchsvolle Themen wie Hardware Tampering auf dem selben Stack aufgezeigt werden.



Installing Rancher

Requirements: 2 vCPU and 4GB of RAM

cloud-init

Tested with Ubuntu 22.04 and openSUSE Leap 15.4

```
#cloud-config
### System
locale: en_US.UTF-8
timezone: Europe/Berlin
### Users
user: rancher
ssh_authorized_keys:
- ssh-ed25519 .....
### Install
package_update: true
package_upgrade: true
package_reboot_if_required: true
packages: ["bash-completion", "fzf"]
### Files
write_files:
- path: /root/.bashrc
  content: |
    export KUBECONFIG=/etc/rancher/k3s/k3s.yaml
    alias k=kubectl
    complete -o default -F __start_kubectl k
    source <(kubectl completion bash | sed 's#"${requestComp}" 2>/dev/null#"${requestComp}" 2>/dev/null | head -n -1 | fzf --multi=0 #g')
  append: true
- path: /etc/sysctl.d/90-kubelet.conf
  content: |
    vm.panic_on_oom=0
    vm.overcommit_memory=1
    kernel.panic=10
    kernel.panic_on_oops=1
    kernel.keys.root_maxbytes=25000000
- path: /etc/sysctl.d/90-networking.conf
  content: |
    net.ipv4.conf.all.forwarding = 1
    net.ipv6.conf.all.disable_ipv6 = 1
    net.ipv6.conf.default.disable_ipv6 = 1
```

```
    net.ipv6.conf.lo.disable_ipv6 = 1
- path: /etc/rancher/k3s/config.yaml
  content: |
    protect-kernel-defaults: true
```

Sources

- <https://docs.k3s.io/installation/requirements>
- <https://ranchermanager.docs.rancher.com/pages-for-subheaders/installation-requirements>
- <https://docs.k3s.io/security/hardening-guide>





Installing k3s

1. Check the supported Kubernetes version for the rancher release (e.g. 1.25.9)
2. Install the matching k3s

```
# !!! For production this should be a HA installation !!!  
curl -sfL https://get.k3s.io | INSTALL_K3S_VERSION=v1.25.9+k3s1 sh -
```

3. Verify the installation

```
kubectl get node
```

4. Install Helm

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

Sources

- <https://docs.k3s.io/installation/uninstall>
- <https://docs.k3s.io/installation/configuration>



Installing Rancher Managment Server

1. Install Cert Manager

```
helm repo add jetstack https://charts.jetstack.io
helm repo update
helm install cert-manager jetstack/cert-manager \
--namespace cert-manager \
--create-namespace \
--version v1.11.0 \
--set installCRDs=true
```

2. Install Rancher

```
helm repo add rancher-latest https://releases.rancher.com/server-charts/latest
helm repo update
helm install rancher rancher-latest/rancher \
--namespace cattle-system \
--create-namespace \
--set hostname=cloudland.giebert.dev \
--set global.cattle.psp.enabled=false \
--set ingress.tls.source=letsEncrypt \
--set letsEncrypt.email=me@giebert.dev \
--set replicas=1
```

3. Wait for the Rollout

```
kubectl -n cattle-system rollout status deploy/rancher
```

4. Navigate to <https://cloudland.giebert.dev>

Sources

- <https://github.com/rancher/rancher/>
- <https://ranchermanager.docs.rancher.com>



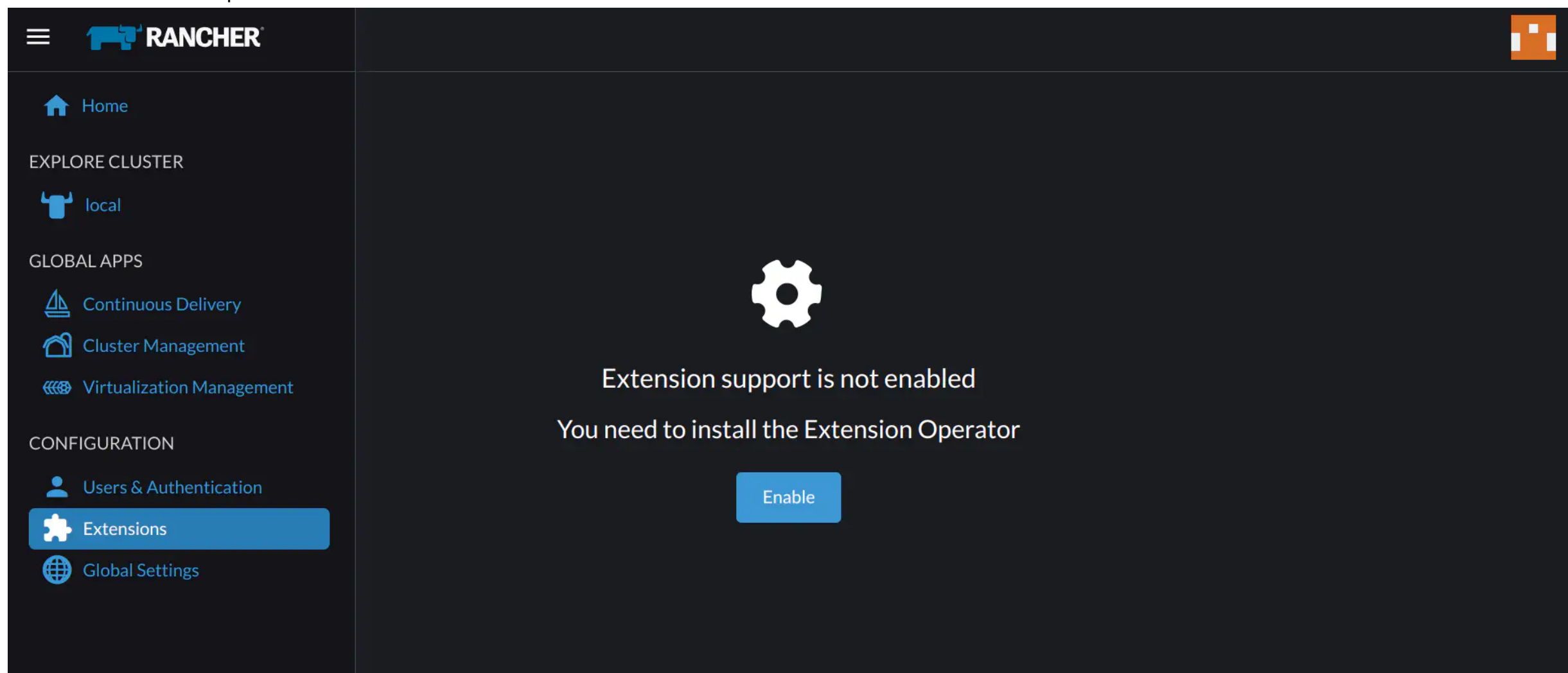
Enabling Rancher Extensions

New in Rancher v2.7.0


Rancher introduces extensions:


- Extend and enhance the Rancher UI.
- Independent of Rancher releases.
- Extensions are Helm charts that can only be installed once into a cluster

1. Enable the Extension Operator



2. Enable the Elemental Extension

RANCHER




Extensions

Installed

Available

Updates

All




Elemental

OS Management extension

1.1.0

Install



Kubewarden

Kubewarden extension for Rancher Manager

1.0.5

Install



Install the Elemental Operator

K3s includes a Helm Controller that manages installing, upgrading/reconfiguring, and uninstalling Helm charts using a HelmChart Custom Resource Definition (CRD). Paired with auto-deploying AddOn manifests, installing a Helm chart on your cluster can be automated by creating a single file on disk.

1. Create `/var/lib/rancher/k3s/server/manifests/` with the following content:

```
apiVersion: v1
kind: Namespace
metadata:
  name: cattle-elemental-system
---
apiVersion: helm.cattle.io/v1
kind: HelmChart
metadata:
  name: elemental-operator
  namespace: cattle-elemental-system
spec:
  chart: oci://registry.opensuse.org/isv/rancher/elemental/dev/charts/rancher/elemental-operator-chart
```

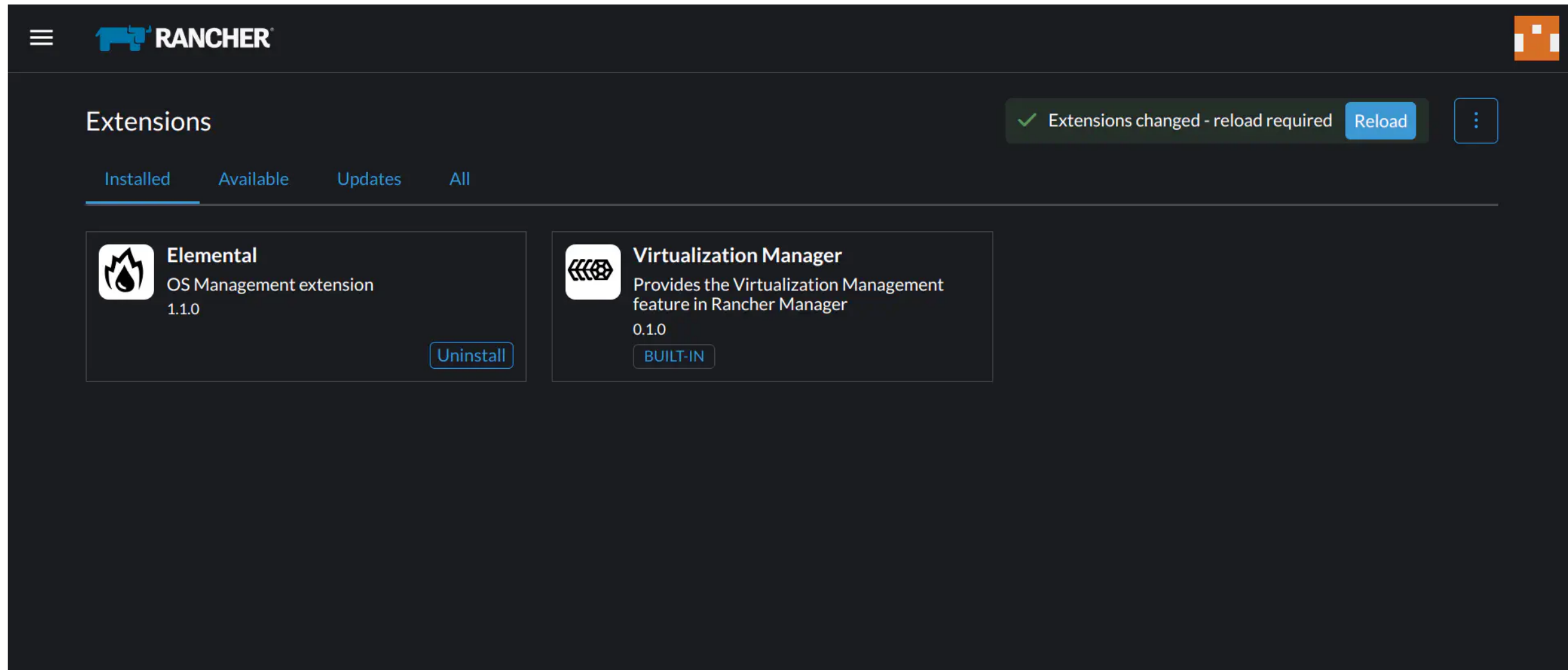
Sources

- <https://docs.k3s.io/helm>
- <https://elemental.docs.rancher.com/elementaloperatorchart-reference>

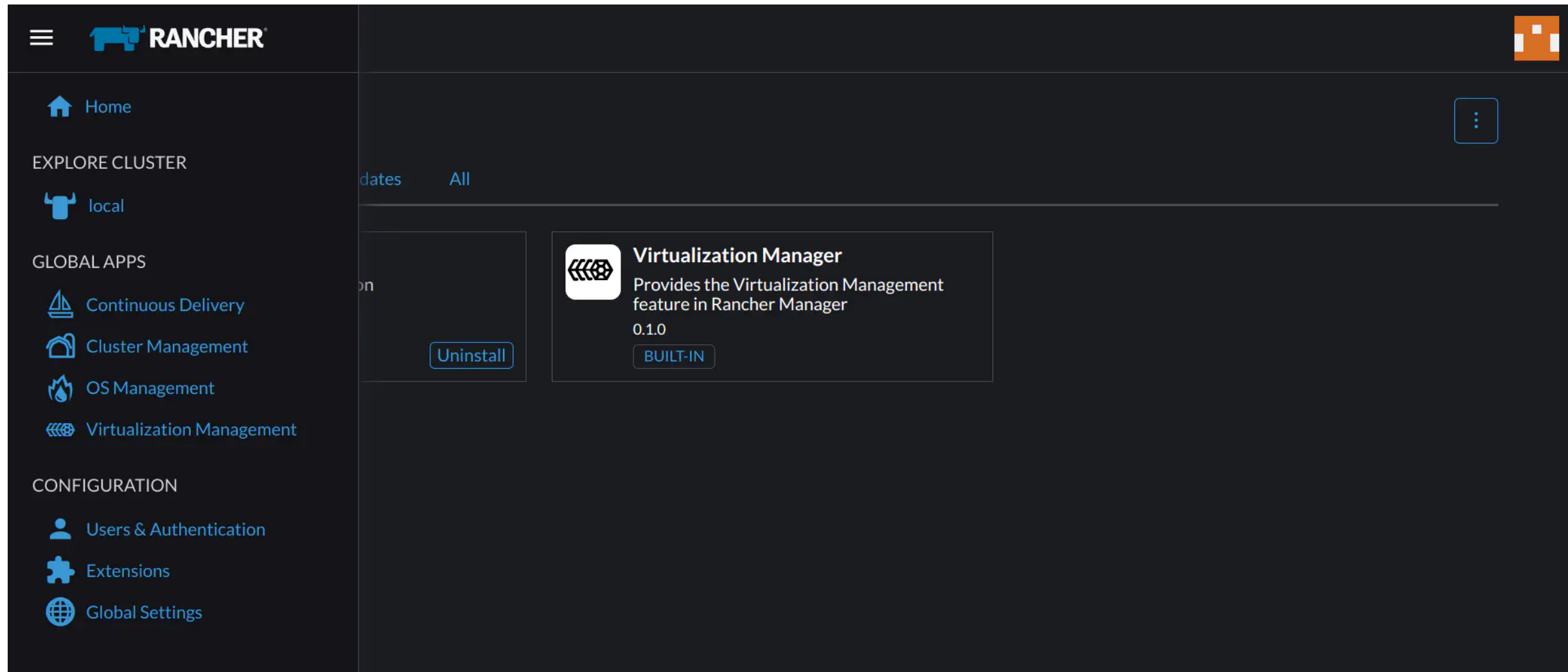


Elemental Extension

1. Reload



2. Navigate to OS Management

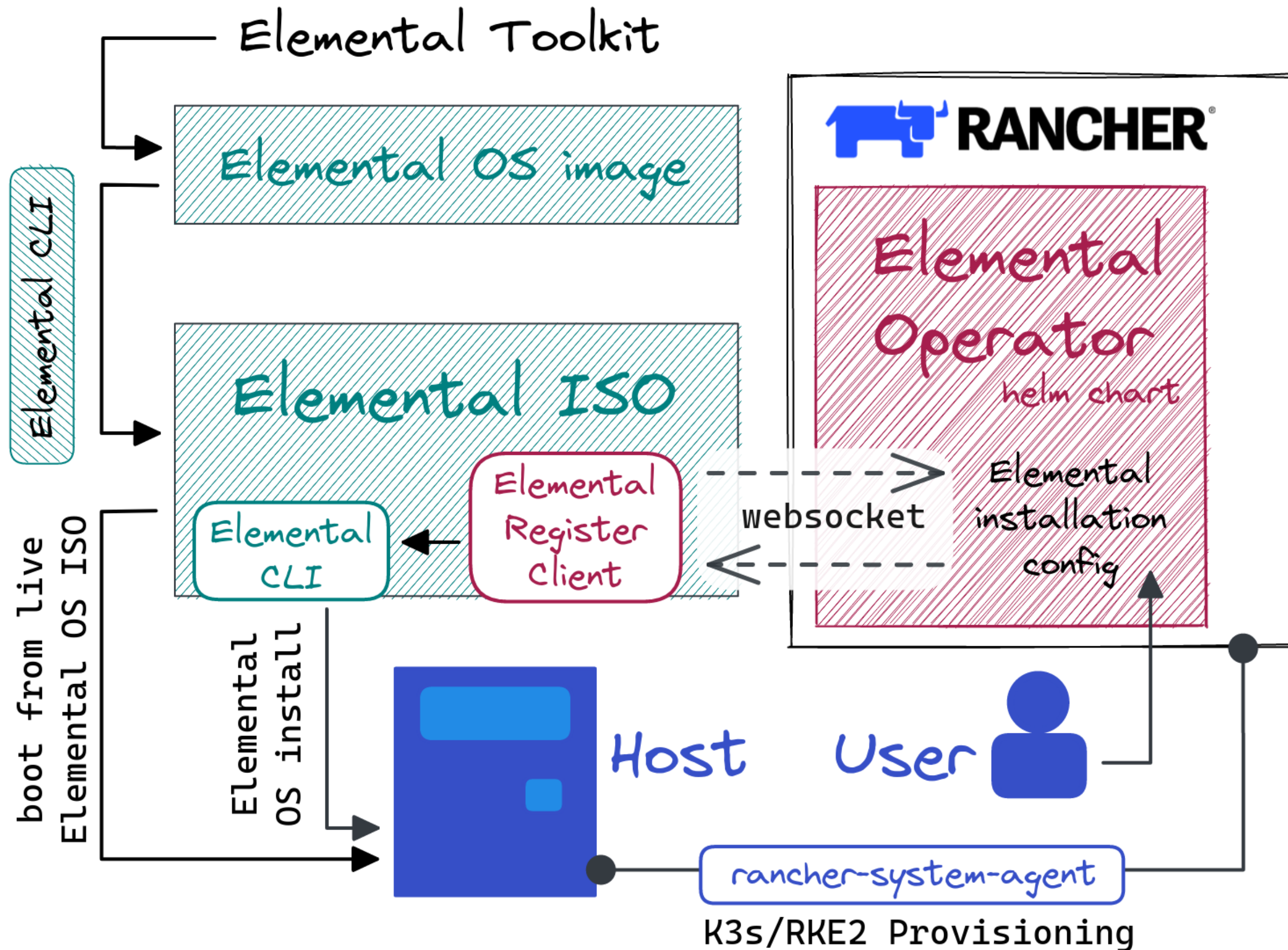


Sources

- <https://ranchermanager.docs.rancher.com/integrations-in-rancher/rancher-extensions>
- <https://elemental.docs.rancher.com/quickstart-ui>



Architecture



- **Elemental Operator** - connects to Rancher Manager and handles MachineRegistration and MachineInventory CRDs
- **Elemental Register Client** - registers machines via MachineRegistrations and installs them via elemental-cli
- **Elemental CLI** - installs any elemental-toolkit based derivative. Basically an installer based on our A/B install and upgrade system
- **Elemental ISO** - includes all the tools needed to perform a full node provisioning
- **Elemental OS image** - create a new image using a Dockerfile based on Elemental Teal image.
- **Elemental Toolkit** - includes a set of OS utilities to enable OS management via containers. Includes dracut modules, bootloader configuration, cloud-init style configuration services, etc.

Sources

- <https://elemental.docs.rancher.com/architecture>



Create the Registration Endpoints

OS Management

Dashboard

Registration Endpoints 0

Inventory of Machines 0

Advanced

Registration Endpoint: Create

Configuration

Name*

raspberry

Cloud Configuration

```
1 config:
2   cloud-config:
3     users:
4       - name: root
5         passwd: rancher
6   elemental:
7     install:
8       reboot: true
9       device: /dev/mmcblk0
10      debug: true
11      disable-boot-entry: true
12      registration:
13        emulate-tpm: true
14  machineInventoryLabels:
15    manufacturer: "${System Information/Manufacturer}"
16    productName: "${System Information/Product Name}"
17    serialNumber: "${System Information/Serial Number}"
18    machineUUID: "${System Information/UUID}"
```

Read from File

1. Populate the Cloud Configuration with this example

```
config:
  cloud-config:
    users:
      - name: root
    ssh_authorized_keys:
      - ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOY5nEt0qssNTouZzN4LPg8M30yDAwGDDvreTUMA6hQ5
```

```
elemental:
  install:
    poweroff: true
    device: /dev/mmcblk0
    disable-boot-entry: true
  registration:
    emulate-tpm: true
    emulated-tpm-seed: -1
machineInventoryLabels:
  manufacturer: "${System Information/Manufacturer}"
  productName: "${System Information/Product Name}"
  serialNumber: "${System Information/Serial Number}"
  machineUUID: "${System Information/UUID}"
```

2. Click create and “Download the Configuration File”

3. Create a folder in `rpi-image` named `overlay` and place the configuration file as `livecd-cloud-config.yaml`



Building a custom OS Image

Preparing your machine

1. Install [docker](#) or [buildah](#)
2. Install emulators to build cross-platform [source](#)

```
docker run --privileged --rm tonistiigi/binfmt --install arm64
```

```
podman run --privileged --rm docker.io/tonistiigi/binfmt --install arm64
```

Your first OS Image

1. Use the [wifi.connection.example](#) and place it in the same folder named `wifi.connection`
2. Create a Dockerfile

```
FROM registry.opensuse.org/isv/rancher/elemental/stable/teal53/15.4/rancher/elemental-teal/5.3:latest AS build
```

```
# e.g check https://en.opensuse.org/Package_repositories
```

```
RUN rpm --import http://download.opensuse.org/distribution/leap/15.4/repo/oss/gpg-pubkey-3dbdc284-53674dd4.asc && \
    zypper addrepo --refresh http://download.opensuse.org/distribution/leap/15.4/repo/oss/ oss && \
    zypper --non-interactive rm k9s kernel-firmware* && \
    zypper --non-interactive in ncdu htop && \
    # Add for Raspberry
    zypper --non-interactive in kernel-firmware-usb-network kernel-firmware-brcm kernel-firmware-bnx2 && \
    # Hack needed for Hetzner
    # zypper --non-interactive rm selinux-tools && \
    zypper --non-interactive update && \
    zypper clean --all
```

```
COPY --chmod=0600 wifi.connection /etc/NetworkManager/system-connections/wifi.connection
```

```
# IMPORTANT: /etc/os-release is used for versioning/upgrade. The
```

```
# values here should reflect the tag of the image currently being built
```

```
ARG IMAGE_REPO=norepo
```

```
ARG IMAGE_TAG=latest
```

```
RUN echo "IMAGE_REPO=${IMAGE_REPO}" > /etc/os-release && \
    echo "IMAGE_TAG=${IMAGE_TAG}" >> /etc/os-release && \
    echo "IMAGE=${IMAGE_REPO}:${IMAGE_TAG}" >> /etc/os-release
```

```
# Flatten the image to reduce the final size since we are not interested in layers
FROM scratch as os
COPY --from=build / /
```

3. Customize and add packages by appending to the zypper install or add complete custom Dockerfile commands

4. Login to registry

```
docker login
```

```
buildah login docker.io
```

5. Export the configuration and adapt it to your username and tag version

```
export IMAGE_REPO=dgiebert/rpi-os-image
export IMAGE_TAG=v0.0.1
```

6. Build the OS Image

```
docker buildx build . \
  --push \
  --platform linux/arm64 \
  --build-arg IMAGE_REPO=${IMAGE_REPO} \
  --build-arg IMAGE_TAG=${IMAGE_TAG} \
  --tag ${IMAGE_REPO}:${IMAGE_TAG} \
  --target os

buildah manifest create ${IMAGE_REPO}:${IMAGE_TAG}
buildah build \
  --platform linux/arm64,linux/amd64 \
  --build-arg IMAGE_REPO=${IMAGE_REPO} \
  --build-arg IMAGE_TAG=${IMAGE_TAG} \
  --manifest ${IMAGE_REPO}:${IMAGE_TAG} \
  --target os
buildah manifest push --all "localhost/${IMAGE_REPO}:${IMAGE_TAG}" "docker://docker.io/${IMAGE_REPO}:${IMAGE_TAG}"
```

Sources

- <https://github.com/dgiebert/elemental-iso-builder/tree/main>
- <https://elemental.docs.rancher.com/customizing>



Building a the ISO

Note Make sure to check [the requirements](#)

Create a multi-stage

1. Append to the Dockerfile to create a multi-stage Dockerfile

```
FROM registry.opensuse.org/isv/rancher/elemental/stable/teal53/15.4/rancher/elemental-builder-image/5.3:latest AS builder
ARG TARGETARCH
WORKDIR /iso
COPY --from=os / rootfs
COPY overlay overlay

# Fix needed for buildah
RUN rm -rf rootfs/etc/resolv.conf && \
    ln -s /var/run/netconfig/resolv.conf rootfs/etc/resolv.conf && \
    mkdir output && \
    elemental build-iso \
        dir:rootfs \
        --bootloader-in-rootfs \
        --squash-no-compression \
        -n "elemental-teal-${TARGETARCH}" && \
    xorriso -indev "elemental-teal-${TARGETARCH}.iso" -outdev "output/elemental-teal-${TARGETARCH}.iso" -map overlay / -boot_image any req

FROM scratch AS iso
COPY --from=builder /iso/output .
```

2. Export the configuration and adapt it to your username and tag version

```
export IMAGE_REPO=dgiebert/rpi-os-image
export IMAGE_TAG=v0.0.1
```

3. Build the ISO

```
docker buildx build . \
    --platform linux/arm64 \
    --build-arg IMAGE_REPO=${IMAGE_REPO} \
    --build-arg IMAGE_TAG=${IMAGE_TAG} \
```

```
--target iso \
--output .
```

```
buildah build \
  --platform linux/arm64 \
  --build-arg IMAGE_REPO=${IMAGE_REPO} \
  --build-arg IMAGE_TAG=${IMAGE_TAG} \
  --target iso \
  --output .
```

4. For the Raspberry Pi the image now needs to be modified please use the provided rpi.sh script in the folder holding the elemental-teal-arm64.iso

```
wget https://raw.githubusercontent.com/dgiebert/cloudland-2023/master/assets/rpi.sh
chmod +x rpi.sh
./rpi.sh
```

5. Write the resulting elemental-teal-rpi.iso to the USB key

```
sudo dd if=elemental-teal-rpi.iso of=/dev/sda status=progress
```

Sources

- <https://github.com/dgiebert/elemental-iso-builder/tree/main>
- <https://elemental.docs.rancher.com/customizing>



Running on Hetzner

1. Create a registration with the following configuration

```
config:
  cloud-config:
    users:
      - name: root
        ssh_authorized_keys:
          - ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOY5nEt0qssNTouZzN4LPg8M30yDAwGDDvreTUMA6hQ5
    elemental:
      install:
        device: /dev/sda
        system-uri: dgiebert/hetzner-os-image:v0.0.1
      registration:
        emulate-tpm: true
        emulated-tpm-seed: -1
  machineInventoryLabels:
    registrationEndpoint: hetzner
    manufacturer: "${System Information/Manufacturer}"
    productName: "${System Information/Product Name}"
    serialNumber: "${System Information/Serial Number}"
    machineUUID: "${System Information/UUID}"
```

2. Put the vServer or Server into rescue mode

3. Export your registration url

```
export REGISTRATION_URL=https.....
```

4. SSH and execute the commands to register the nodes

```
# Install dependencies
apt update
apt install -y golang libssl-dev
# Get the Elemental CLI
wget https://github.com/rancher/elemental-cli/releases/download/v0.3.1/elemental-v0.3.1-Linux-arm64.tar.gz
tar -xf elemental-v0.3.1-Linux-arm64.tar.gz
mv elemental /usr/local/bin
# Create the needed folders, binaries
ln -s /usr/bin/grub-editenv /usr/bin/grub2-editenv
mkdir /oem
```

```
# Build the elemental-register commands
git clone https://github.com/rancher/elemental-operator.git
cd elemental-operator/
make register
# Register and reboot
./build/elemental-register --registration-url ${REGISTRATION_URL} --emulate-tpm true --emulated-tpm-seed -1
reboot
```



Upgrade OS

1. Export the configuration and adapt it to your username and tag version

```
export IMAGE_REPO=dgiebert/rpi-os-image
export IMAGE_TAG=v0.0.2
```

2. Build a new version

```
docker buildx build . \
  --push \
  --platform linux/arm64 \
  --build-arg IMAGE_REPO=${IMAGE_REPO} \
  --build-arg IMAGE_TAG=${IMAGE_VERSION} \
  --tag ${IMAGE_REPO}:${IMAGE_VERSION} \
  --target os

buildah manifest create ${IMAGE_REPO}:${IMAGE_VERSION}
buildah build \
  --platform linux/arm64 \
  --build-arg IMAGE_REPO=${IMAGE_REPO} \
  --build-arg IMAGE_TAG=${IMAGE_VERSION} \
  --manifest ${IMAGE_REPO}:${IMAGE_VERSION} \
  --target os
buildah manifest push --all "localhost/${IMAGE_REPO}:${IMAGE_VERSION}" "docker://docker.io/${IMAGE_REPO}:${IMAGE_VERSION}"
```

3. Create a list with the available versions in the following JSON format

```
[
  {
    "metadata": {
      "name": "v0.0.1"
    },
    "spec": {
      "version": "v0.0.1",
      "type": "container",
      "metadata": {
        "upgradeImage": "dgiebert/rpi-os-image:v0.0.1"
      }
    }
  }
]
```

```

    },
    {
      "metadata": {
        "name": "v0.0.2"
      },
      "spec": {
        "version": "v0.0.2",
        "type": "container",
        "metadata": {
          "upgradeImage": "dgiebert/rpi-os-image:v0.0.2"
        }
      }
    }
  ]
}

```

4. Append to the Dockerfile to create a multi-stage Dockerfile

```

FROM busybox AS versions
COPY versions.json /versions.json
CMD ["/bin/cp", "/versions.json", "/data/output"]

```

5. Create and upload the needed image

```

docker buildx build . \
  --target versions \
  --platform linux/arm64 \
  --push \
  -t ${IMAGE_REPO}-versions

buildah manifest create ${IMAGE_REPO}-versions
buildah build \
  --platform linux/arm64 \
  --target versions \
  --manifest ${IMAGE_REPO}-versions
buildah manifest push --all "localhost/${IMAGE_REPO}-versions" "docker://docker.io/${IMAGE_REPO}-versions"

```

6. Create a Version Channel to list available images

```

apiVersion: elemental.cattle.io/v1beta1
kind: ManagedOSVersionChannel
metadata:
  name: dgiebert-rpi-versions
  namespace: fleet-default

```

```
spec:
  options:
    image: dgiebert/rpi-os-image-versions
  syncInterval: 1h
  type: custom
```

7. Create the Upgrade

```
apiVersion: elemental.cattle.io/v1beta1
kind: ManagedOSImage
metadata:
  name: rpi
  namespace: fleet-default
spec:
  clusterTargets:
    - clusterName: elemental
  managedOSVersionName: v0.0.2
  osImage: ''
```




Create from existing nodes

UI

- 1. Click the three dots and then select Create Elemental Cluster

OS Management

Dashboard

Registration Endpoints1

Inventory of Machines2

Advanced

Inventory of Machines

Create Elemental Cluster

Download YAML

Delete

Create from YAML

Add Filter

State	Name	Cluster	Age
Namespace: fleet-default			
Active	m-1f873db6-be4f-4df3-a0f7-c0d64815f109	elemental	
Active	m-1566600f-d1df-4c3b-ba12-427351aca730	elemental	

Edit YAML

Clone

Download YAML

Delete

Create Elemental Cluster

v2.7.4

2. Provided the needed details

Cluster: Create Elemental

Cluster Name *

example

Cluster Description

Any text you want that better describes this cluster

Machine Pools

pool1

Pool Name *

pool1

Machine Count *

1

Roles

☒ etcd

☒ Control Plane

☒ Worker

Inventory of Machines Selector Template

Key

create-cluster-selector

▼

Operator

in list

▼

Value

GNshQbl6ZmwAGOB1ISV91IOS

Remove

Add Rule

Matches 1 of 2 existing Inventory of Machines: "m-1f873db6-be4f-4df3-a0f7-c0d64815f109"

Show Advanced

+

−

Cluster Configuration

- Basics
- Member Roles
- Add-On Config
- Agent Environment Vars
- etcd
- Labels & Annotations
- Networking
- Registries
- Upgrade Strategy
- Advanced

Basics

Kubernetes Version

v1.25.9+k3s1

☐ Show deprecated Kubernetes patch versions ⓘ

Security

Pod Security Policies have been removed in Kubernetes v1.25, use Pod Security Admission instead.

Pod Security Admission Configuration Template

(None)

☒ Encrypt Secrets
☐ Project Network Isolation

System Services

☒ CoreDNS
☒ Klipper Service LB
☒ Traefik Ingress
☒ Local Storage
☒ Metrics Server

Using HelmChart

```

apiVersion: helm.cattle.io/v1
kind: HelmChart
metadata:
  name: elemental-cluster-1
  namespace: fleet-default
spec:
  chart: oci://ghcr.io/dgiebert/cluster-template-examples
  version: 0.0.3
  valuesContent: |-
    cloudprovider: elemental
    kubernetesVersion: "v1.24.13+k3s1"
    rke:
      machineGlobalConfig:

```

```
      secrets-encryption: true
machineSelectorConfig:
- config:
  docker: false
  protect-kernel-defaults: true
  selinux: false
nodepools:
- name: elemental-server
  etcd: true
  controlplane: true
  worker: true
  matchLabels:
    registrationEndpoint: rpi
```