

PT. SYNNEX METRODATA INDONESIA

Methode of Procedure Rancher Implementation

Version: 2.6.1

Date: 2023-01-16



SUSE | RANCHER®

Conducted & Prepared By



PT Synnex Metrodata Indonesia

Jakarta - Indonesia

APL Tower 42nd Floor, Jl. Letjen S. Parman Kav 28, Jakarta Barat 11470

Kontrol Dokumen

Tanda Tangan

	Organisasi	Nama	Tanda Tangan
Penyusun :	PT SMI	M Ibnu Rusydan	
Kualitas Kontrol :	PT SMI	Ardian Saputra Kurniawan	

Sejarah Rilis

Versi	Tanggal Dibuat	Halaman	Catatan
1.0		Semua	Rilis Pertama

Kerahasiaan, hak cipta, dan hak kekayaan intelektual

Informasi yang terkandung bersifat rahasia dan eksklusif untuk PT Synnex Metrodata Indonesia. Tidak boleh diungkapkan atau ditransfer, secara langsung atau tidak langsung, kepada pihak ketiga tanpa izin tertulis eksplisit PT Synnex Metrodata Indonesia.

Semua hak cipta. Tidak ada bagian dari dokumen ini boleh direproduksi, disimpan dalam sistem pencarian, diterjemahkan, atau ditransmisikan dalam bentuk apapun atau dengan cara apapun, elektronik, mekanik, fotokopi, rekaman, atau sebaliknya, tanpa izin tertulis dari PT Synnex Metrodata Indonesia.

Hak cipta dan hak kekayaan intelektual lainnya dalam program asli, spesifikasi, laporan atau item lainnya yang timbul dalam perjalanan, atau yang dihasilkan dari proyek tersebut akan tetap menjadi milik PT Synnex Metrodata Indonesia.

DAFTAR ISI

Kontrol Dokumen.....	2
1 Pendahuluan.....	5
1.1 Tujuan.....	5
1.2 Ruang Lingkup.....	5
2 Daftar Node yang di Implementasi	6
3 Daftar Product yang di Implementasi.....	7
4 Topology	7
5 Dokumen Referensi.....	8
6 Install dan Konfigurasi	9
6.1 Deploy kubernetes cluster base-on RKE for Rancher Management Server cluster.....	9
6.1.1 <i>Initial config on all node.....</i>	9
6.1.2 <i>Install RKE.....</i>	10
6.1.3 <i>Deploy k8s cluster base-on RKE.....</i>	10
6.1.4 <i>Install kubectl.....</i>	11
6.2 Deploy Rancher Management Server for centralized management kubernetes.....	12
6.3 Deploy downstream k8s cluster from Rancher Management Server.....	13
6.4 Deploy Longhorn Distribute Block Storage from Rancher Management Server.....	14
6.4.1 <i>Minimum Recommended Hardware.....</i>	14
6.4.2 <i>Minimal Available Storage and Over-provisioning.....</i>	14
6.4.3 <i>Node Preparation.....</i>	15
6.4.4 <i>Deploy Longhorn.....</i>	16
6.5 Deploy Load Balancer on downstream k8s cluster from Rancher Management Server.....	20
6.6 Deploy Monitoring for downstream k8s cluster from Rancher Management Server.....	21
6.7 Deploy Wordpress on downstream k8s cluster from Rancher Management Server.....	22
6.8 Deploy Rancher Backups from Rancher Management Server.....	24
6.9 Backup & Restore Persistent Storage application with Longhorn.....	25
6.9.1 <i>How to Backup.....</i>	25
6.9.2 <i>How to Restore.....</i>	26
6.10 Disaster Recovery.....	29
6.10.1 <i>Concepct Section.....</i>	29
6.10.2 <i>Requiment.....</i>	30
6.10.3 <i>Kondisi Existing.....</i>	30
6.10.4 <i>Use Case.....</i>	30
6.10.5 <i>Operation.....</i>	31

1 Pendahuluan

1.1 Tujuan

Tujuan dari dokumen ini adalah untuk men-demo kan solusi terkait kebutuhan kubernetes environment dari SUSE Rancher.

Dokumen ini mencakup Methode of Procedure Implementation for Docker container engine, Rancher kubernetes engine, Rancher management server & Longhorn block storage on Suse Linux Enterprise Server.

1.2 Ruang Lingkup

Ruang lingkup dari dokumen ini mencakup:

- Topology
- Dokumen Referensi
- Deploy kubernetes cluster base-on RKE for Rancher Management Server cluster
- Deploy Rancher Management Server for centralized management kubernetes
- Deploy downstream k8s cluster from Rancher Management Server
- Deploy Longhorn Distribute Block Storage from Rancher Management Server
- Deploy Load Balancer on downstream k8s cluster from Rancher Management Server
- Deploy Monitoring for downstream k8s cluster from Rancher Management Server
- Deploy Wordpress on downstream k8s cluster from Rancher Management Server
- Deploy Rancher Backups from Rancher Management Server
- Backup & Restore Persistent Storage application with Longhorn
- Disaster Recovery

2 Daftar node yang di Implementasi

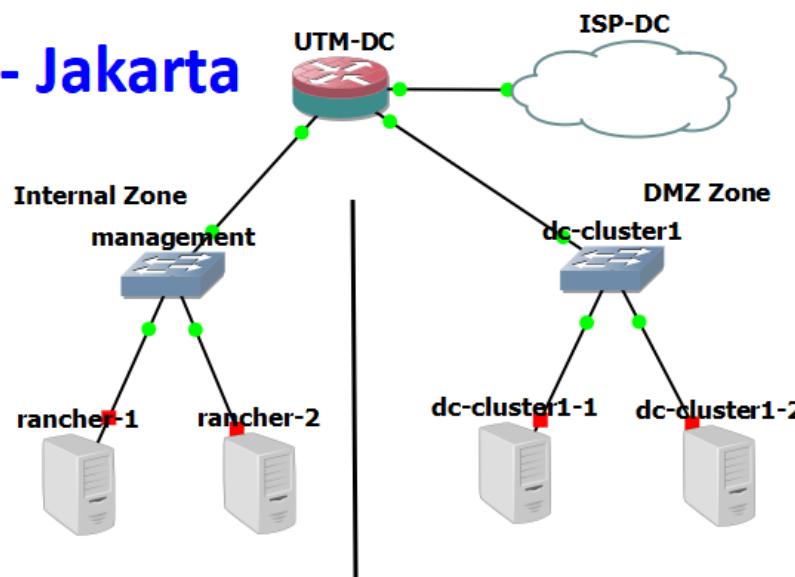
No	Node Name	Specification	Role	Description
1	rancher-1	cpu: 2vcpu, ram: 4gb, disk: 20gb	Controlplane, etcd, worker	RMS cluster node
2	rancher-2	cpu: 2vcpu, ram: 4gb, disk: 20gb	Controlplane, etcd, worker	RMS cluster node
3	dc-cluster1-1	cpu: 6vcpu, ram: 6gb, disk0-OS: 20gb, disk1-Longhorn: 5gb	Controlplane, etcd, worker	Downstream k8s dc-cluster1
4	dc-cluster1-2	cpu: 6vcpu, ram: 6gb, disk0-OS: 20gb, disk1-Longhorn: 5gb	Controlplane, etcd, worker	Downstream k8s dc-cluster1
5	dr-cluster1-1	cpu: 6vcpu, ram: 6gb, disk0-OS: 20gb, disk1-Longhorn: 5gb	Controlplane, etcd, worker	Downstream k8s dr-cluster1
6	dr-cluster1-2	cpu: 6vcpu, ram: 6gb, disk0-OS: 20gb, disk1-Longhorn: 5gb	Controlplane, etcd, worker	Downstream k8s dr-cluster1

3 Daftar Product yang di Implementasi

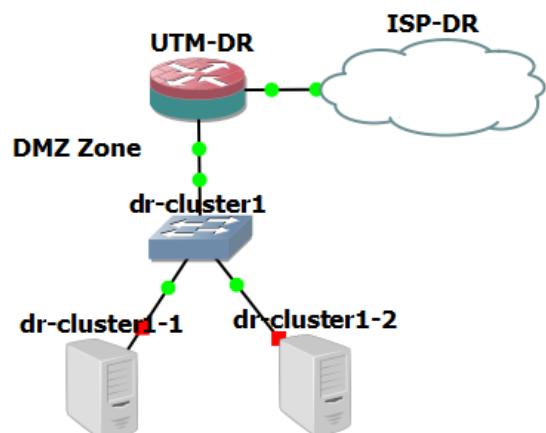
No	Product	Version
1	Suse Linux Enterprise Server	15 SP3
2	Docker Container Engine	20.10.9-ce
3	Rancher Kubernetes Engine	v1.21.7
4	Longhorn Distribute Block Storage	v1.2.3
5	Rancher Management Server	2.6.1

4 Topology

DC - Jakarta



DR - Purwakarta



Dev | Branch | Edge



5 Dokumen Referensi

- Rancher Docs : <https://ranchermanager.docs.rancher.com/v2.6>
- SUSE Rancher support matrix : <https://www.suse.com/suse-rancher/support-matrix/all-supported-versions/rancher-v2-6-1/>
- Port Requiment : <https://ranchermanager.docs.rancher.com/v2.6/getting-started/installation-and-upgrade/installation-requirements/port-requirements#downstream-kubernetes-cluster-nodes>

rancher.example.com

Status: Alive
Operating system:
 IP: 172.30.201.5
 MAC: 52:54:00:19:2B:5A
Manufacturer:
 NetBIOS:
 User:
 Type:
 Date:
Comments:

Port in Rancher

Service	Details
HTTP	308 Permanent Redirect (nginx reverse proxy)
Port 22 (TCP)	OpenSSH 7.2 protocol 2.0
Port 80 (TCP)	nginx reverse proxy
Port 443 (TCP)	Tunnel is ssl: nginx reverse proxy
Port 2379 (TCP)	Tunnel is ssl: unknown service
Port 2380 (TCP)	Tunnel is ssl: unknown service
Port 6443 (TCP)	Tunnel is ssl: unknown service
Port 8181 (TCP)	nginx reverse proxy
Port 8443 (TCP)	Tunnel is ssl: unknown service
Port 10250 (TCP)	Tunnel is ssl: Golang net/http server Go-IPFS json-rpc or InfluxDB API
Port 10251 (TCP)	
Port 10252 (TCP)	
Port 10254 (TCP)	Golang net/http server Go-IPFS json-rpc or InfluxDB API
Port 10257 (TCP)	Tunnel is ssl: unknown service
Port 10259 (TCP)	Tunnel is ssl: unknown service

192.168.137.100	
Status:	Alive
Operating system:	
IP:	192.168.137.100
MAC:	08:00:27:B4:9D:F4
Manufacturer:	PCS Systemtechnik GmbH
NetBIOS:	
User:	
Type:	Port in downstream k8s cluster
Date:	
Comments:	
Service	Details
Port 22 (TCP)	OpenSSH 8.4 protocol 2.0
Port 80 (TCP)	nginx reverse proxy
Port 443 (TCP)	Tunnel is ssl: nginx reverse proxy
Port 2379 (TCP)	Tunnel is ssl: unknown service
Port 2380 (TCP)	Tunnel is ssl: unknown service
Port 6443 (TCP)	Tunnel is ssl: unknown service
Port 7472 (TCP)	Golang net/http server Go-IPFS json-rpc or InfluxDB API
Port 7946 (TCP)	
Port 9796 (TCP)	
Port 10011 (TCP)	
Port 10012 (TCP)	
Port 10013 (TCP)	
Port 10014 (TCP)	
Port 10250 (TCP)	Tunnel is ssl: Golang net/http server Go-IPFS json-rpc or InfluxDB API
Port 10251 (TCP)	
Port 10252 (TCP)	
Port 10257 (TCP)	Tunnel is ssl: unknown service
Port 10259 (TCP)	Tunnel is ssl: unknown service
Port 30465 (TCP)	Apache httpd 2.4.52 (Unix) OpenSSL/1.1.1d PHP/7.4.27
Port 30892 (TCP)	Tunnel is Apache httpd SSL-only mode: Apache httpd 2.4.52 (Unix) OpenSSL/1.1.1d PHP/7.4.27

- Rancher Administration Docs (*Permit for Internal or Partner Use*) :
<https://drive.google.com/file/d/12KU6Jys4aPOaQKR1Yz49Z3d3p1usilUY/view?usp=sharing>
https://drive.google.com/file/d/1_QGLstVkltyDVJ5hmrNtBo9EEWDi_FIU/view?usp=sharing
- Longhorn Docs : <https://longhorn.io/docs/>
- RKE download : <https://github.com/rancher/rke/releases/>
- Rancher Management Server download :
if use kubernetes, please use package source with helm repo : <https://releases.rancher.com/server-charts/stable>
if use docker engine, please use this package : <https://github.com/rancher/rancher>

6 Install dan Konfigurasi

6.1 Deploy kubernetes cluster base-on RKE for Rancher Management Server cluster

6.1.1 Initial config on all node

```
-- All node, example use SUSE Linux Enterprise Server
LVM disk | no SWAP | Enable root user | Enable ssh | SSH user in docker group
systemctl stop firewalld
systemctl disable firewalld

# SSH config
vi /etc/ssh/sshd_config
AllowTcpForwarding yes

# activate RegCode
Request active code subscription *bisa trial mode, 60 hari
aktivasi di sisi OS :
SUSEConnect -r xxxxxxxxxxxx -e user@email.com & config NTP

# All node
vi /etc/hosts
ip-RMS/rancher-1    rancher.example.com        rancher          #to resolve RMS name
ip-rancher-1         rancher-1.example.com     rancher-1       #to resolve host self name
dst

# All node
Query :
1. Generate SSH (all node)
ssh-keygen

2. Copy keygen SSH antar server (di fungsikan agar ketika ssh, tidak perlu credentials)
ssh-copy-id -i ~/.ssh/id_rsa.pub <ip_localhost> dan <ip_server_tujuan>

# All node
Install Docker:
On SLES 15 :
regist your SLES OS
#Add container module from cli
SUSEConnect -p sle-module-containers/15.3/x86_64 -r "
#or Add container module from yast
yast -> software -> system extensions -> select Containers Module -> next -> accept -> ok -> finish
zypper -n install docker
systemctl start docker
systemctl enable docker
systemctl status docker
```

6.1.2 Install RKE

```
##-- Installing RKE | run on rancher-1 & rancher-2
wget https://github.com/rancher/rke/releases/download/v1.2.14/rke_linux-amd64
mv rke_linux-amd64 rke
chmod +x rke
mv rke /usr/local/bin/
rke --version
```

6.1.3 Deploy k8s cluster base-on RKE

```
##-- Create the cluter configuration File | run on rancher-1
rke config
sesuaikan konfigurasi nya
rancher-1:~ # rke config
[+] Cluster Level SSH Private Key Path [~/.ssh/id_rsa]:
[+] Number of Hosts [1]: 2
[+] SSH Address of host (1) [none]: 172.30.201.5
[+] SSH Port of host (1) [22]:
[+] SSH Private Key Path of host (172.30.201.5) [none]:
[-] You have entered empty SSH key path, trying fetch from SSH key parameter
[+] SSH Private Key of host (172.30.201.5) [none]:
[-] You have entered empty SSH key, defaulting to cluster level SSH key: ~/.ssh/id_rsa
[+] SSH User of host (172.30.201.5) [ubuntu]: root
[+] Is host (172.30.201.5) a Control Plane host (y/n)? [y]: y
[+] Is host (172.30.201.5) a Worker host (y/n)? [n]: y
[+] Is host (172.30.201.5) an etcd host (y/n)? [n]: y
[+] Override Hostname of host (172.30.201.5) [none]: rancher-1
[+] Internal IP of host (172.30.201.5) [none]:
[+] Docker socket path on host (172.30.201.5) [/var/run/docker.sock]:
[+] SSH Address of host (2) [none]: 172.30.201.6
[+] SSH Port of host (2) [22]:
[+] SSH Private Key Path of host (172.30.201.6) [none]:
[-] You have entered empty SSH key path, trying fetch from SSH key parameter
[+] SSH Private Key of host (172.30.201.6) [none]:
[-] You have entered empty SSH key, defaulting to cluster level SSH key: ~/.ssh/id_rsa
[+] SSH User of host (172.30.201.6) [ubuntu]: root
[+] Is host (172.30.201.6) a Control Plane host (y/n)? [y]: y
[+] Is host (172.30.201.6) a Worker host (y/n)? [n]: y
[+] Is host (172.30.201.6) an etcd host (y/n)? [n]: y
[+] Override Hostname of host (172.30.201.6) [none]: rancher-2
[+] Internal IP of host (172.30.201.6) [none]:
[+] Docker socket path on host (172.30.201.6) [/var/run/docker.sock]:
[+] Network Plugin Type (flannel, calico, weave, canal, aci) [canal]:
[+] Authentication Strategy [x509]:
[+] Authorization Mode (rbac, none) [rbac]:
[+] Kubernetes Docker image [rancher/hyperkube:v1.20.12-rancher1]:
[+] Cluster domain [cluster.local]: example.com
[+] Service Cluster IP Range [10.43.0.0/16]:
[+] Enable PodSecurityPolicy [n]:
[+] Cluster Network CIDR [10.42.0.0/16]: ■

##-- Deploy k8s with RKE
rke up
or
rke up --ignore-docker-version
```

6.1.4 Install kubectl

```
#-- Install Kubectl | run on rancher-1
zypper -n install kubernetes1.18-client

#-- Verify k8s cluster
root@poc00rke00:~# kubectl --kubeconfig kube_config_cluster.yml get node
NAME      STATUS   ROLES          AGE    VERSION
poc00rke00  Ready    controlplane,etcd,worker  30m   v1.17.4

#-- Setting kube-config
cd /root
cp kube_config_cluster.yml .kube/kube-rke
cat > kube-rke.sh <<EOF
export KUBECONFIG=/root/.kube/kube-rke
EOF
chmod +x kube-rke.sh
source kube-rke.sh
kubectl get node
```

6.2 Deploy Rancher Management Server for centralized management kubernetes

```
## Run on rancher-1

## Install Helm
sudo wget -O helm.tar.gz https://get.helm.sh/helm-v3.4.0-linux-amd64.tar.gz
sudo tar -zxf helm.tar.gz
sudo mv linux-amd64/helm /usr/local/bin/helm
sudo chmod +x /usr/local/bin/helm
sudo rm -rf linux-amd64
sudo rm -f helm.tar.gz
helm version --client

##-- Installing Rancher with Helm
# Install Cert Manager
kubectl create namespace cert-manager
kubectl apply --validate=false -f https://github.com/jetstack/cert-manager/releases/download/v0.15.0/cert-manager.crds.yaml
helm repo add jetstack https://charts.jetstack.io
helm repo update
helm install \
cert-manager jetstack/cert-manager \
--namespace cert-manager \
--version v0.15.0
kubectl get all -n cert-manager #pastikan smw pod ter-create terlebih dahulu
kubectl -n cert-manager rollout status deploy/cert-manager
kubectl -n cert-manager rollout status deploy/cert-manager-webhook

# Add Rancher package
helm repo add rancher-stable https://releases.rancher.com/server-charts/stable
helm repo update

kubectl create namespace cattle-system

# Install rancher
helm install rancher rancher-stable/rancher \
--namespace cattle-system \
--set hostname=rancher.example.com \
--version v2.6.1 \
--set replicas=2
*sesuaikan hostname untuk RMS nya dan sesuaikan replicas nya, contoh disini di set 2, karna menggunakan 2 node

# Verify rancher ready
while true; do curl -kv https://rancher.example.com 2>&1 | grep -q "dynamiclistener-ca"; if [ $? != 0 ]; then echo "Rancher isn't ready yet"; sleep 5; continue; fi; break; done; echo "Rancher is Ready";
```

Pastikan pod di cert-manager & cattle-system running/completed

```
rancher-1:~ # kubectl get pod -n cert-manager
NAME                               READY   STATUS    RESTARTS   AGE
cert-manager-86b8b4f4b7-sk4mp      1/1    Running   0          10m
cert-manager-cainjector-7f6686b94-g74vm  1/1    Running   0          10m
cert-manager-webhook-66d786db8c-wtbmc  1/1    Running   0          10m

rancher-1:~ # kubectl get pod -n cattle-system
NAME                               READY   STATUS    RESTARTS   AGE
helm-operation-g7j7v              0/2    Completed  0          2m56s
helm-operation-m6vkl              0/2    Completed  0          2m28s
helm-operation-nj66h              0/2    Completed  0          3m59s
helm-operation-plqmm              0/2    Completed  0          2m6s
helm-operation-xjs46              0/2    Completed  0          2m40s
rancher-7546f84756-gmwkp        1/1    Running   0          5m57s
rancher-7546f84756-lq6f9        1/1    Running   0          5m56s
rancher-webhook-6979fdb4bf-kfm7q  1/1    Running   0          2m19s

rancher-1:~ #
```

kubectl -n cattle-system rollout status deploy/rancher

Set A record atau dns di sisi laptop/client/lan :

ip-rancher-1	rancher.example.com
ip-rancher-2	rancher.example.com

access in web, <https://rancher.example.com>

6.3 Deploy downstream k8s cluster from Rancher Management Server

6.4 Deploy Longhorn Distribute Block Storage from Rancher Management Server

6.4.1 Minimum Recommended Hardware

- 3 nodes
- 4 vcpu per node
- 4 gb ram per node
- SSD/NVME or similar performance block device on the node for storage. We don't recommend using spinning disks with Longhorn, due to low IOPS.

6.4.2 Minimal Available Storage and Over-provisioning

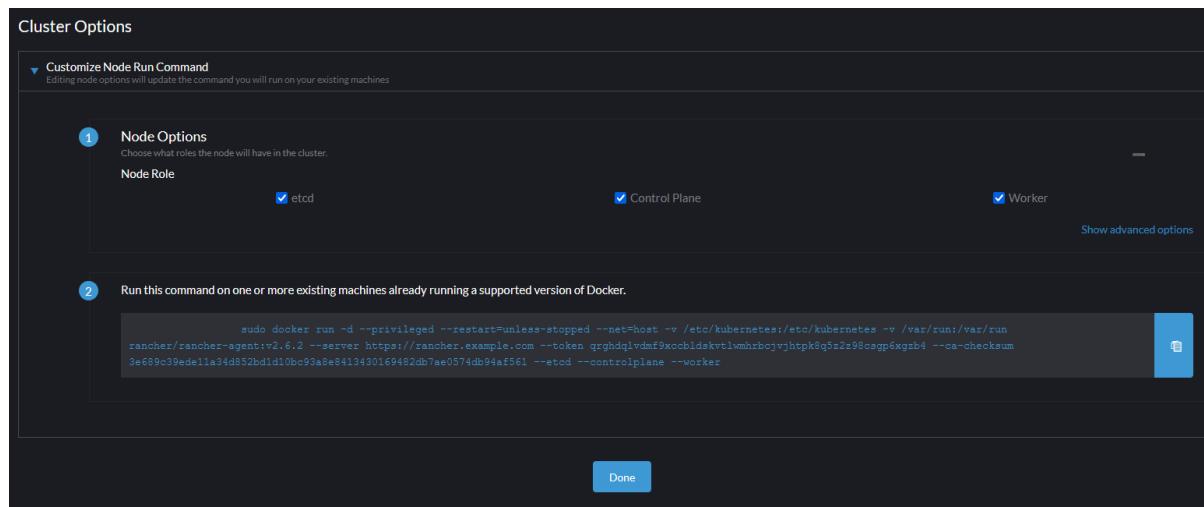
If you need to use the **root disk**, use the default minimal available storage percentage **setup which is 25%**, and set overprovisioning percentage to 200% to minimize the chance of DiskPressure.

If you're using a **dedicated disk for Longhorn**, you can lower the setting minimal available storage percentage to **10%**.

For the Over-provisioning percentage, it depends on how much space your volume uses on average. For example, if your workload only uses half of the available volume size, you can set the Over-provisioning percentage to 200, which means Longhorn will consider the disk to have twice the schedulable size as its full size minus the reserved space.

Go to Rancher Management Server -> cluster management -> create -> Use existing nodes and create a cluster using RKE -> insert cluster name & Leave other settings as the defaults -> next.

Sesuaikan role dan follow instruksi nya



The screenshot shows the 'Cluster Options' section of the Rancher UI. It includes a 'Customize Node Run Command' section with a note about updating the command for existing machines. Below this, the 'Node Options' section allows selecting roles: etcd (checked), Control Plane (checked), and Worker (checked). A 'Show advanced options' link is present. Step 2 indicates running a command on existing machines, with a code block showing the command:

```
sudo docker run -d --privileged --restart=unless-stopped --net=host -v /etc/kubernetes:/etc/kubernetes -v /var/run:/var/run
rancher/rancher-agent:v2.6.2 --server https://rancher.example.com --token grghdqlvdhf9xochldkvtlwhrbcjvjhtpk8g5z2z98csgp6xgsb4 --ca-checksum
3e689c39ede11a34d852bd1d10b93a8e8413430169482db7ae0574db94af561 --etcd --controlplane --worker
```

A 'Done' button is at the bottom right.

Langkah ini di gunakan untuk men-deploy dc-cluster1 dan dr-cluster1

Disk Space Management - Since Longhorn doesn't currently support sharding between the different disks, **we recommend using LVM to aggregate all the disks for Longhorn into a single partition**, so it can be easily extended in the future.

PT SYNNEX METRODATA INDONESIA

6.4.3 Node Preparation

```
## Run on dc-cluster1-1, dc-cluster1-2, dr-cluster1-1, dr-cluster1-2

# General info
default path : /var/lib/longhorn/
default disk stote : create partition with lvm
file system support : ext4, xfs, btrfs

# Make sure mandatory package
bash --version
curl --version
findmnt --version
grep --version
awk --version
blkid --version

# Install open-iscsi
zypper install open-iscsi
iscsid --version
systemctl start iscsid
systemctl enable iscsid
systemctl status iscsid

# Setup disk with LVM
lvmdiskscan
pvcreate /dev/sdb
vgcreate longhornvg /dev/sdb
lvcreate -l 100%FREE -n longhorn longhornvg -y
mkfs.xfs /dev/longhornvg/longhorn
echo "/dev/longhornvg/longhorn    /var/lib/longhorn/      xfs defaults 0 0" >> /etc/fstab
mkdir -p /var/lib/longhorn/
mount -a
df -hT /var/lib/longhorn
```

6.4.4 Deploy Longhorn

Deploy Longhorn

Goto Rancher Web -> Cluster management -> explore on your downstream k8s cluster -> project/namespaces -> create project, ex:
Storage (for longhorn system) -> apps & marketplace -> filter (search longhorn) -> install -> install into project: Storage -> next -> follow
this instructions.

Sesuaikan point berikut :

Edit options -> Longhorn default settings -> customize default settings

 Storage minimal available percentage

 Default replica count

 Concurrent replica rebuild per node limit

Longhorn CSI Driver Settings

Longhorn Storage Class Settings

 Default storage class replica count

Install

**dalam demo ini menggunakan 2 worker node, jadi replica di set 2, untuk di sebar ke 2 worker node*

More Resources

Namespace: longhorn-system

	Deployed	longhorn	longhorn:100.1.0+up1.2.2	16	9 mins
	Deployed	longhorn-crd	longhorn-crd:100.1.0+up1.2.2	15	10 mins

v2.6.2

Install longhorn-system:longhorn

```

Thu, Nov 18 2021 8:37:39 pm      LAST DEPLOYED: Thu Nov 18 13:35:39 2021
Thu, Nov 18 2021 8:37:39 pm      NAME: longhorn-system
Thu, Nov 18 2021 8:37:39 pm      STATUS: deployed
Thu, Nov 18 2021 8:37:39 pm      REVISION: 1
Thu, Nov 18 2021 8:37:39 pm      TEST SUITE: None
Thu, Nov 18 2021 8:37:39 pm      NOTES:
Thu, Nov 18 2021 8:37:39 pm      Longhorn is now installed on the cluster!
Thu, Nov 18 2021 8:37:39 pm      Please wait a few minutes for other Longhorn components such as CSI deployments, Engine Images, and Instance Managers to be initialized.
Thu, Nov 18 2021 8:37:39 pm      Visit our documentation at https://longhorn.io/docs/
Thu, Nov 18 2021 8:37:39 pm      SUCCESS: helm upgrade --install=true --namespace=longhorn-system --timeout=10m0s --values=/home/shell/helm/values-longhorn-100.1.0-up1.2.2.yaml --version=100.1.0+up1.2.2 --wait=true longhorn
Thu, Nov 18 2021 8:37:39 pm      /home/shell/helm/longhorn-100.1.0-up1.2.2.tgz
Thu, Nov 18 2021 8:37:39 pm

```

Project: longhorn

Nodes	Name	Roles	Version	External/Internal IP	OS	CPU	RAM	Pods	Age
dr01	All	v1.2.6	192.168.137.101	Linux	22%	67%	20%	1 day	

v2.6.2

Kubectl: dr-cluster1

```

NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE   SELECTOR
service/csi-attacher   ClusterIP   <none>        12345/TCP   9m64s   app=csi-attacher
service/csi-provisioner ClusterIP   <none>        12346/TCP   9m64s   app=csi-provisioner
service/csi-resizer     ClusterIP   <none>        12347/TCP   9m64s   app=csi-resizer
service/csi-snapshotter ClusterIP   <none>        12348/TCP   9m64s   app=csi-snapshotter
service/longhorn-backend ClusterIP   <none>        9500/TCP   1m      app=longhorn-manager
service/longhorn-frontend ClusterIP   <none>        80/TCP     10s     app=longhorn-ui

NAME           DESIRED   CURRENT  READY   AGE   SELECTOR
demonset.apps/engine-image-el-0422ab0c 1/1     1/1     1/1     1m    engine-image-el-0422ab0c   rancher/mirrored-longhornio-longhorn-engine:v1.2.2
longhorn.io/component-engine-image.longhorn.io/engine-image-el-0422ab0c 1/1     1/1     1/1     1m    engine-image-el-0422ab0c   rancher/mirrored-longhornio-longhorn-engine:v1.2.2
demonset.apps/node-driver-registrar 1/2     1/2     1/2     9m44s  node-driver-registrar.longhorn-csi-plugin   rancher/mirrored-longhornio-csi-node-driver-registrar:v2.3.0,rancher/mirrored-longhornio-longhorn-csi-node-driver-registrar:v2.3.0
demonset.apps/app-longhorn-csi-plugin 1/1     1/1     1/1     9m44s  app-longhorn-csi-plugin.longhorn-csi-plugin   rancher/mirrored-longhornio-longhorn-csi-plugin:v1.2.2
demonset.apps/longhorn-manager 1/1     1/1     1/1     1m    longhorn-manager.longhorn-manager   rancher/mirrored-longhornio-longhorn-manager:v1.2.2

NAME           DESIRED   CURRENT  READY   AGE   SELECTOR
deployment.apps/csi-attacher 1/1     1/1     1/1     9m64s  csi-attacher.rancher/mirrored-longhornio-csi-attacher:v3.2.1   app=csi-attacher
deployment.apps/csi-provisioner 1/1     1/1     1/1     9m64s  csi-provisioner.rancher/mirrored-longhornio-csi-provisioner:v2.1.2   app=csi-provisioner
deployment.apps/csi-resizer 1/1     1/1     1/1     9m64s  csi-resizer.rancher/mirrored-longhornio-csi-resizer:v1.2.0   app=csi-resizer
deployment.apps/csi-snapshotter 1/1     1/1     1/1     9m64s  csi-snapshotter.rancher/mirrored-longhornio-csi-snapshotter:v1.3   app=csi-snapshotter
deployment.apps/longhorn-driver-deployer 1/1     1/1     1/1     1m    longhorn-driver-deployer.rancher/mirrored-longhornio-longhorn-driver-deployer:v1.2.2   app=longhorn-driver-deployer
deployment.apps/longhorn-ui 1/1     1/1     1/1     1m    longhorn-ui.rancher/mirrored-longhornio-longhorn-ui:v1.2.2   app=longhorn-ui

NAME           DESIRED   CURRENT  READY   AGE   CONTAINERS   IMAGES
replicaset.apps/csi-attacher-56c54648 1/1     1/1     1/1     9m64s  csi-attacher   rancher/mirrored-longhornio-csi-attacher:v3.2.1   app=csi-attacher,pod-template-hash=56c54648
replicaset.apps/csi-provisioner-76594c87b 1/1     1/1     1/1     9m64s  csi-provisioner   rancher/mirrored-longhornio-csi-provisioner:v2.1.2   app=csi-provisioner,pod-template-hash=76594c87b
replicaset.apps/csi-resizer-76594c87b 1/1     1/1     1/1     9m64s  csi-resizer   rancher/mirrored-longhornio-csi-resizer:v1.2.0   app=csi-resizer,pod-template-hash=76594c87b
replicaset.apps/csi-snapshotter-7f647ccb97 1/1     1/1     1/1     9m64s  csi-snapshotter   rancher/mirrored-longhornio-csi-snapshotter:v3.0.3   app=csi-snapshotter,pod-template-hash=7f647ccb97
replicaset.apps/longhorn-driver-deployer-bbf754577 1/1     1/1     1/1     1m    longhorn-driver-deployer   rancher/mirrored-longhornio-longhorn-driver-deployer:v1.2.2   app=longhorn-driver-deployer,pod-template-hash=bbf754577

```

Clear Connected

Longhorn installed

! 3-worker02.example.com

```

worker02:~ # ls -l /var/lib/longhorn/
total 4
drwxr-xr-x 3 root root 64 Nov 18 17:49 engine-binaries
-rw-r--r-- 1 root root 51 Nov 18 17:43 longhorn-disk.cfg
drwxr-xr-x 2 root root 6 Nov 18 17:43 replicas
worker02:~ # df -hT /var/lib/longhorn/
Filesystem           Type  Size  Used Avail Use% Mounted on
/dev/mapper/longhornvg-longhorn xfs   20G  79M  20G   1% /var/lib/longhorn
worker02:~ #

```

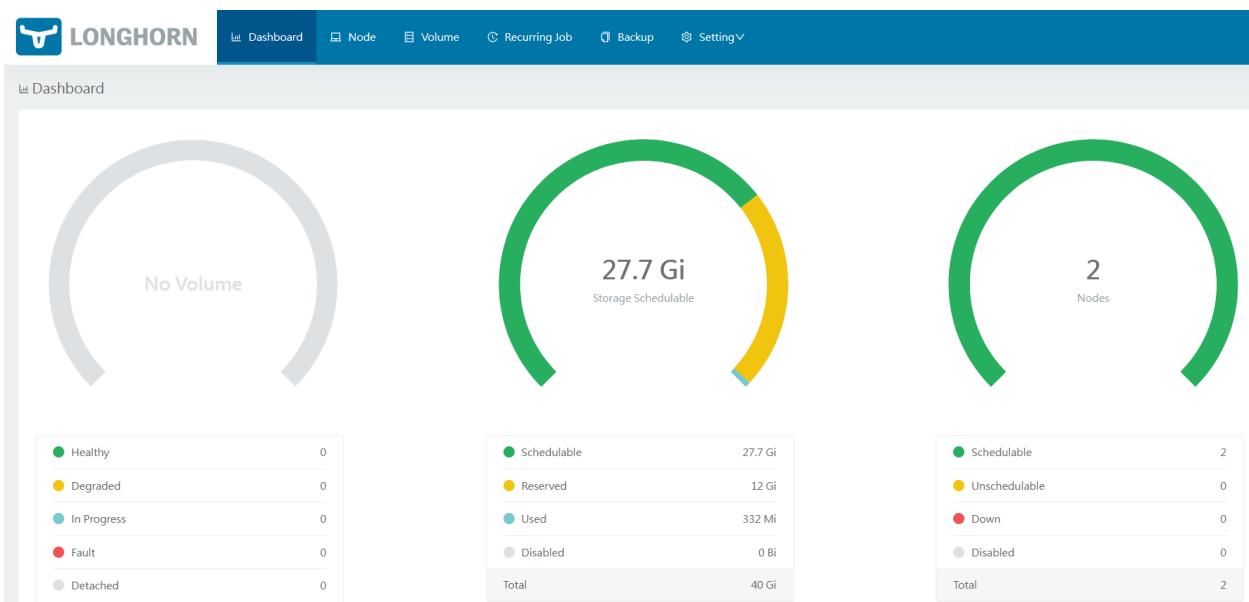
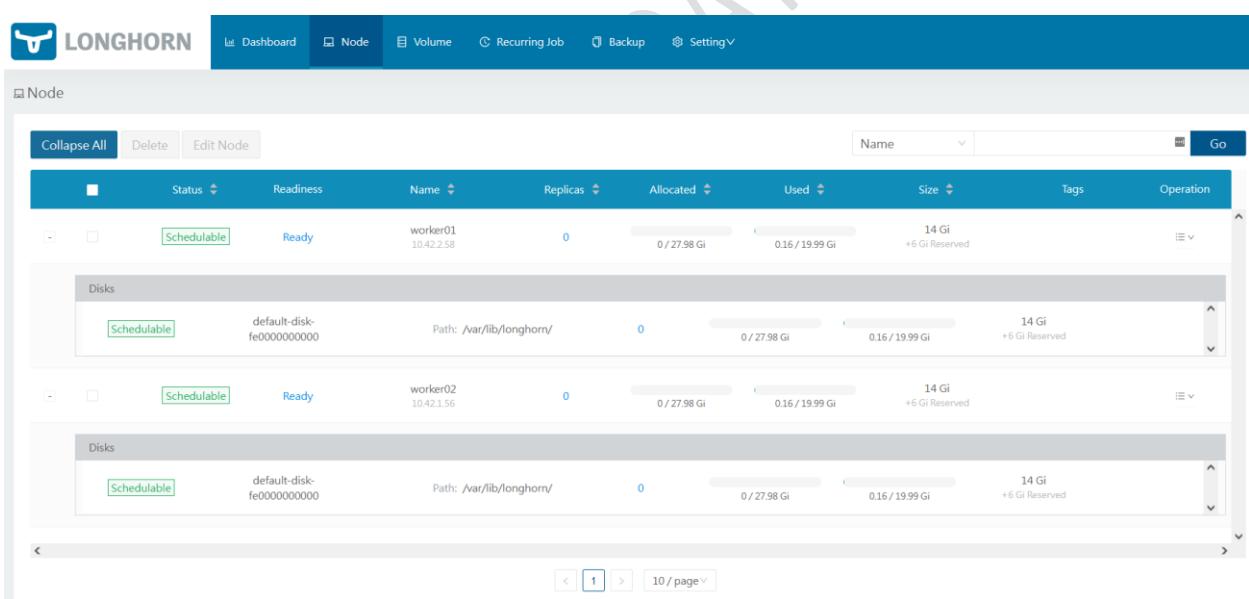
! 2-worker01.example.com

```

worker01:~ # ls -l /var/lib/longhorn/
total 4
drwxr-xr-x 3 root root 64 Nov 18 17:46 engine-binaries
-rw-r--r-- 1 root root 51 Nov 18 17:43 longhorn-disk.cfg
drwxr-xr-x 2 root root 6 Nov 18 17:43 replicas
worker01:~ # df -hT /var/lib/longhorn/
Filesystem           Type  Size  Used Avail Use% Mounted on
/dev/mapper/longhornvg-longhorn xfs   20G  79M  20G   1% /var/lib/longhorn
worker01:~ #

```

Longhorn on File System node

Node

Node List:

Status	Readiness	Name	Replicas	Allocated	Used	Size	Tags	Operation
Schedulable	Ready	worker01 10.42.2.58	0	0 / 27.98 Gi	0.16 / 19.99 Gi	14 Gi +6 Gi Reserved		
Schedulable	Ready	worker02 10.42.1.56	0	0 / 27.98 Gi	0.16 / 19.99 Gi	14 Gi +6 Gi Reserved		

Disks (for each node):

Path	Capacity	Used	Free
/var/lib/longhorn/	0 / 27.98 Gi	0.16 / 19.99 Gi	14 Gi +6 Gi Reserved

Longhorn node menu

Edit Node and Disks

Node Scheduling	Eviction Requested	
<input checked="" type="radio"/> Enable	<input type="radio"/> Disable	
<input type="radio"/> True	<input checked="" type="radio"/> False	
Engine Manager CPU Request	Replica Manager CPU Request	
0 m	0 m	
Node Tags		
+ New Node Tag		
Conditions		
<input checked="" type="radio"/> MountPropagation <input checked="" type="radio"/> Ready <input checked="" type="radio"/> Schedulable		
+ New Disk Tag		
Conditions		
<input checked="" type="radio"/> Ready <input checked="" type="radio"/> Schedulable		
Storage Available	Storage Scheduled	Storage Max
19.82Gi	0Gi	19.99Gi
Name		
default-disk-fe0000000000		
Path		
/var/lib/longhorn/		
Storage Reserved	Scheduling	Eviction Requested
6 Gi	<input checked="" type="radio"/> Enable	<input type="radio"/> Disable
<input type="radio"/> True	<input checked="" type="radio"/> False	Edit
Add Disk		
Cancel Save		

Node and disk info

LONGHORN

- [Dashboard](#)
- [Node](#)
- [Volume](#) (Current)
- [Recurring Job](#)
- [Backup](#)
- [Setting](#)

Volume / pvc-9d65e9cc-20c7-402c-b1c3-867bad2efb9d

Volume Details

State: Attached
Health: Healthy
Ready for workload: Ready
Conditions: restore scheduled

Frontend: Block Device
Attached Node & Endpoint:
worker02
/dev/longhorn/pvc-9d65e9cc-20c7-402c-b1c3-867bad2efb9d

Size: 1 Gi
Actual Size: 58 Mi
Data Locality: disabled

Replicas

 867bad2efb9d-r-77d38530 Healthy worker01 Node instance-manager-r-7f12fd70 /var/lib/longhorn/ Running	 867bad2efb9d-r-842f1ef4 Healthy worker02 Node instance-manager-r-c7d871bc /var/lib/longhorn/ Running
---	--

[Take Snapshot](#)
[Create Backup](#)

Show System Hidden:

Snapshots and Backups

Volume pada longhorn di replicate menjadi 2

6.5 Deploy Load Balancer on downstream k8s cluster from Rancher Management Server

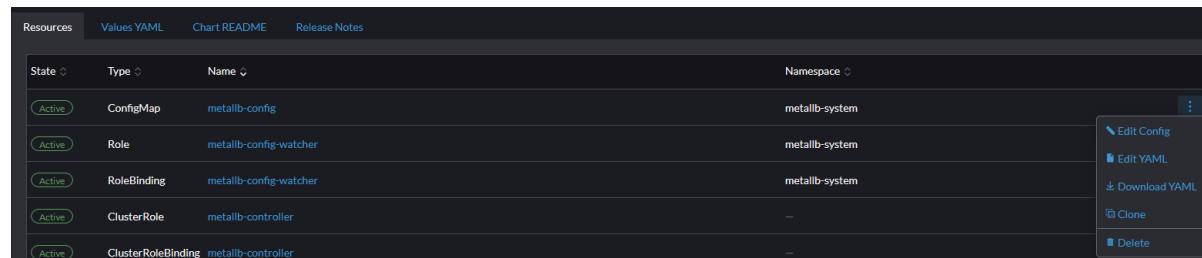
Ingress sudah default ter-install pada k8s cluster base-on RKE. Untuk load balancer masuk ke dalam optional install. Secara best practice kami merekomendasikan load balancer dari enterprise product seperti F5/Citrix/etc, dalam demo ini kami menggunakan MetallB untuk keperluan load balancer nya.

```
# Add repository
Goto Rancher Web -> Cluster management -> explore on your downstream k8s cluster -> apps & marketplace -> repositories -> create -> follow this instructions.

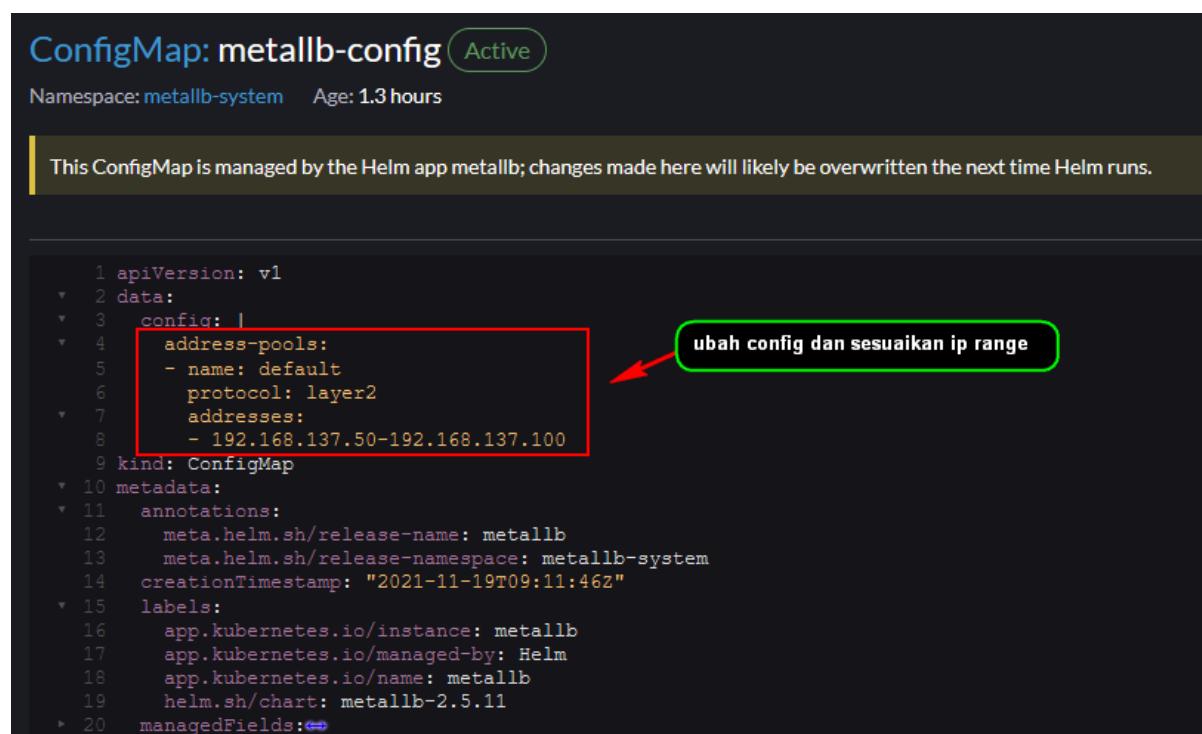
Name : bitnami
Index URL : https://charts.bitnami.com/bitnami

# Deploy MetalLB
Create namespace with metallb-system on Project System
Go to apps & marketplace -> charts -> metallb -> namespace: metallb-system -> next -> install
Make sure metallb already deployed via CLI
    kubectl get all -n metallb-system

# Config IP Pool for Load Balancer
Go to apps & marketplace -> installed apps -> metallb
Edit yaml on ConfigMap -> save
```



State	Type	Name	Namespace
Active	ConfigMap	metallb-config	metallb-system
Active	Role	metallb-config-watcher	metallb-system
Active	RoleBinding	metallb-config-watcher	metallb-system
Active	ClusterRole	metallb-controller	-
Active	ClusterRoleBinding	metallb-controller	-



```
1 apiVersion: v1
2 data:
3   config: |
4     address-pools:
5       - name: default
6         protocol: layer2
7         addresses:
8           - 192.168.137.50-192.168.137.100
9   kind: ConfigMap
10 metadata:
11   annotations:
12     meta.helm.sh/release-name: metallb
13     meta.helm.sh/release-namespace: metallb-system
14   creationTimestamp: "2021-11-19T09:11:46Z"
15   labels:
16     app.kubernetes.io/instance: metallb
17     app.kubernetes.io/managed-by: Helm
18     app.kubernetes.io/name: metallb
19     helm.sh/chart: metallb-2.5.11
20   managedFields:[]
```

6.6 Deploy Monitoring for downstream k8s cluster from Rancher Management Server

Cluster management -> explore on your downstream k8s cluster -> project/namespaces -> create project, ex: Monitoring -> apps & marketplace -> filter (search monitoring) -> install -> install into project Monitoring -> next -> follow this instructions.

Sesuaikan point berikut :

Edit options -> Prometheus

Retention Size

Check list Persistent Storage for Prometheus

Size

Storage Class Name: longhorn

Install

6.7 Deploy Wordpress on downstream k8s cluster from Rancher Management Server

```
# Deploy Wordpress
```

Cluster management -> explore on your downstream k8s cluster -> project/namespaces -> create project, ex: TokonyaKita -> create namespace: tokonyakita-ui -> apps & marketplace -> filter (search wordpress) -> install -> install into namespace: tokonyakita-ui -> next -> follow this instructions.

Sesuaikan point berikut :

Edit YAML & search this point :

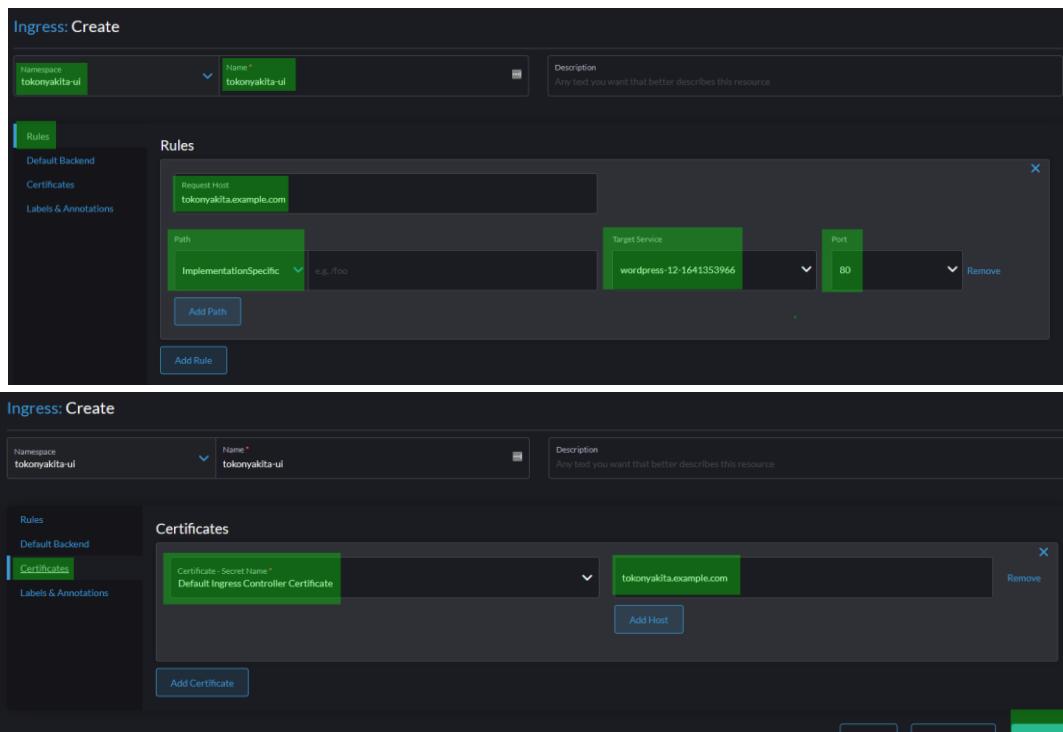
```
wordpressUsername      #for wordpress admin use: admin
wordpressPassword      #for wordpress admin password: linux
wordpressBlogName      #for your apps name: TokonyaKita
10Gi                   #this size for apps: 1Gi
8Gi                    #this size for database: 1Gi
database: bitnami_wordpress
existingSecret: ""      #this secret db: linux
host: localhost
password: ""           #this secret db: linux
port: 3306
user: bn_wordpress

database: bitnami_wordpress
password: ""           #this secret db: linux
rootPassword: ""        #this secret db: linux
username: bn_wordpress
storageClass: ""        #gunakan storageClass untuk persistent storage nya
```

Install

```
# Ingress config
```

Cluster management -> explore on your downstream k8s cluster -> Service Discovery -> Ingress -> Create -> follow this picture -> if already, klik create.



Example workflow

Apps <-> svc-loadbalancer cluster-ip <-> svc-loadbalancer external-ip <-> ingress <-> user access

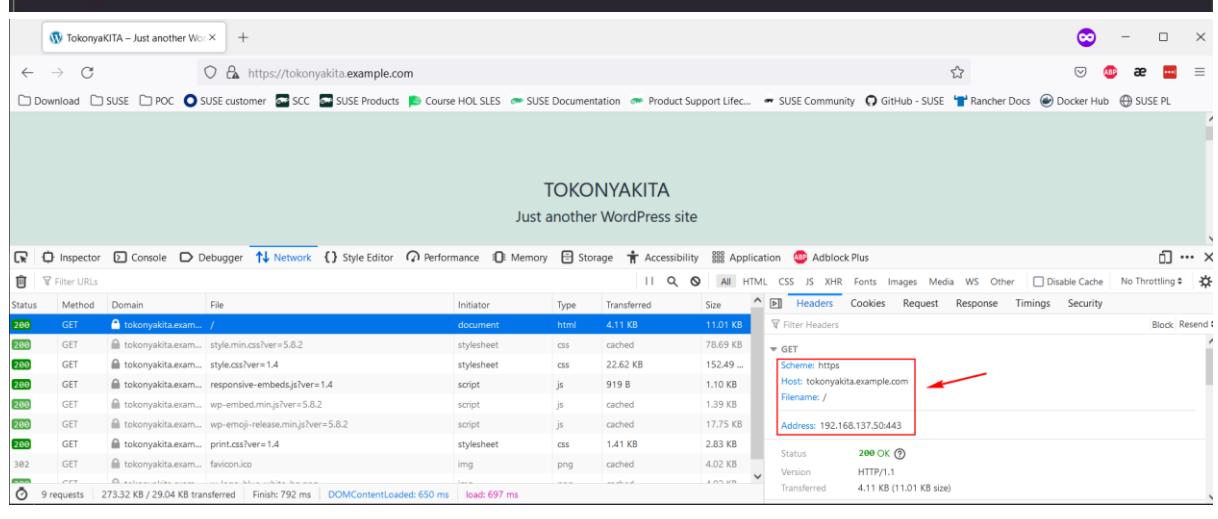
```
master01:~ # kubectl get all -n tokonyakita-ui
NAME                                     READY   STATUS    RESTARTS   AGE
pod/tokonyakita-ui-mariadb-0            1/1     Running   0          4m56s
pod/tokonyakita-ui-wordpress-58f54d5765-2thn9 0/1     Running   0          4m56s

NAME           TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)         AGE
service/tokonyakita-ui-mariadb   ClusterIP   10.43.204.80   <none>        3306/TCP       4m56s
service/tokonyakita-ui-wordpress LoadBalancer   10.43.103.145  192.168.137.50  80:31008/TCP,443:30273/TCP  4m56s

NAME              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/tokonyakita-ui-wordpress   0/1     1           0          4m56s

NAME           DESIRED  CURRENT   READY   AGE
replicaset.apps/tokonyakita-ui-wordpress-58f54d5765 1         1         0         4m56s

NAME           READY   AGE
statefulset.apps/tokonyakita-ui-mariadb  1/1     4m56s
master01:~ # kubectl get services -o wide --all-namespaces | grep --color=never -E '(LoadBalancer|NAMESPACE)'
NAMESPACENAMESPACE   NAME           TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)         AGE
tokonyakita-ui      tokonyakita-ui-wordpress LoadBalancer   10.43.103.145  192.168.137.50  80:31008/TCP,443:30273/TCP  12m   SELECTOR
=wordpress
master01:~ # kubectl get ingress -A
NAME      CLASS      HOSTS          ADDRESS          PORTS   AGE
rancher  rancher   <none>        rancher.example.com  80, 443  2d2h
tokonyakita-ui tokonyakita-ui <none>        tokonyakita.example.com 192.168.137.21,192.168.137.22 80, 443  4m53s
master01:~ #
```



6.8 Deploy Rancher Backups from Rancher Management Server

```
## Example we use local storage with persistent volume
# Create persistent volume
mkdir -p /persistentstorage
cat > pv-rancher-backup.yml <<EOF
kind: PersistentVolume
apiVersion: v1
metadata:
  name: rancher-backup          #name pv
  labels:
    type: local                #define for local storage type
spec:
  capacity:
    storage: 5Gi               #size
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/persistentstorage"  #folder or path location
EOF
kubectl apply -f pv-rancher-backup.yml

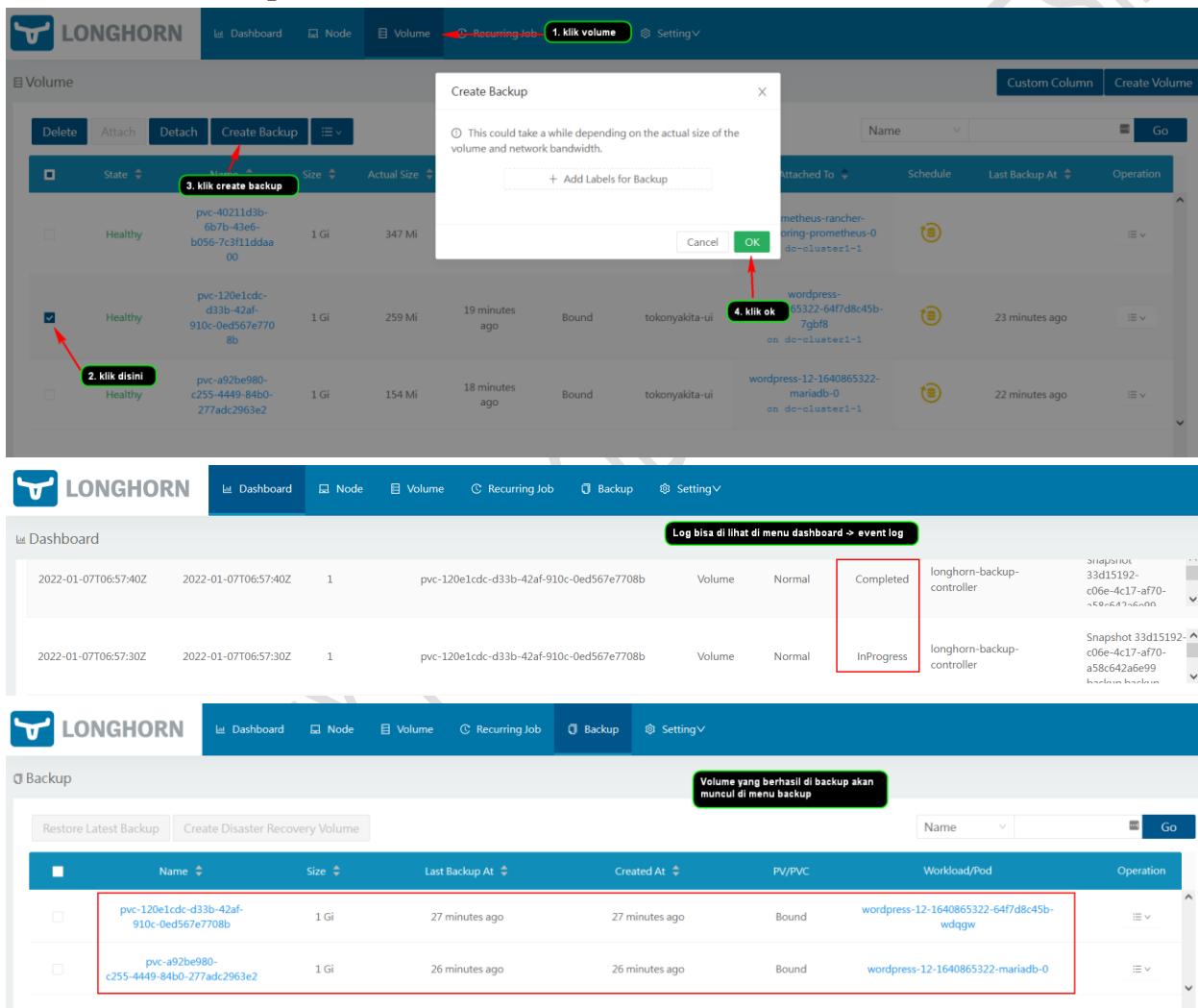
# Deploy Rancher Backups
Cluster management -> explore on your rancher cluster -> project/namespaces -> create project, ex: Backup -> apps & marketplace -> filter (search rancher backups) -> install -> install into project Backup -> next -> follow this instructions.

Sesuaikan point berikut :
Edit options -> Chart Options
  Select: Use an existing persistent volume -> rancher-backup -> install
```

6.9 Backup & Restore Persistent Storage application with Longhorn

Pre-requiment, volume untuk data aplikasi pada longhorn telah di backup. Target backup yang dibutuhkan adalah external storage, bisa dengan NFS maupun S3. Kondisi disini backup target untuk longhorn sudah ready.

6.9.1 How to Backup



The screenshots illustrate the steps to create a backup in the Longhorn UI:

- Step 1: Click on the Volume tab.**
- Step 2: Select the checkbox next to the volume you want to backup.**
- Step 3: Click the "Create Backup" button.**
- Step 4: Click the "OK" button in the confirmation dialog.**

Dashboard View (Top Screenshot):

Name	Attached To	Schedule	Last Backup At	Operation
pvc-40211d3b-6b7b-43e6-b056-7c3f1ddaa00	metheus-rancher-ing-prometheus-0 on de-clusterx1-1			
pvc-120e1cdc-d33b-42af-910c-0ed567e7708b	wordpress-35322-64f7d8c45b-7gbf8 on de-clusterx1-1		23 minutes ago	
pvc-a92be980-c255-4449-84b0-277adc2963e2	wordpress-12-1640865322-mariadb-0 on de-clusterx1-1		22 minutes ago	

Event Log View (Middle Screenshot):

Event	Timestamp	Volume	Status
longhorn-backup-controller	2022-01-07T06:57:40Z	pvc-120e1cdc-d33b-42af-910c-0ed567e7708b	Completed
longhorn-backup-controller	2022-01-07T06:57:30Z	pvc-120e1cdc-d33b-42af-910c-0ed567e7708b	InProgress

Backup View (Bottom Screenshot):

Volume yang berhasil di backup akan muncul di menu backup

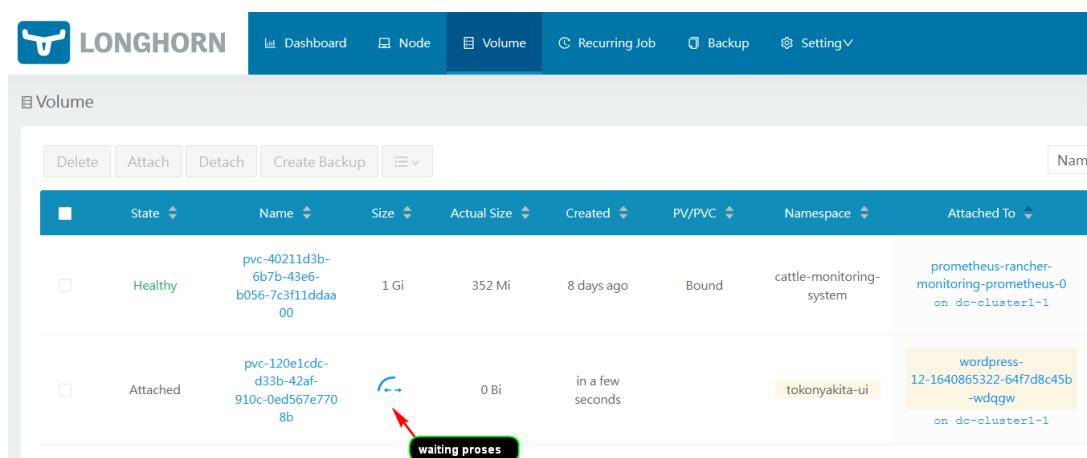
Name	Size	Last Backup At	Created At	PV/PVC	Workload/Pod	Operation
pvc-120e1cdc-d33b-42af-910c-0ed567e7708b	1 Gi	27 minutes ago	27 minutes ago	Bound	wordpress-12-1640865322-64f7d8c45b-wdqgw	
pvc-a92be980-c255-4449-84b0-277adc2963e2	1 Gi	26 minutes ago	26 minutes ago	Bound	wordpress-12-1640865322-mariadb-0	

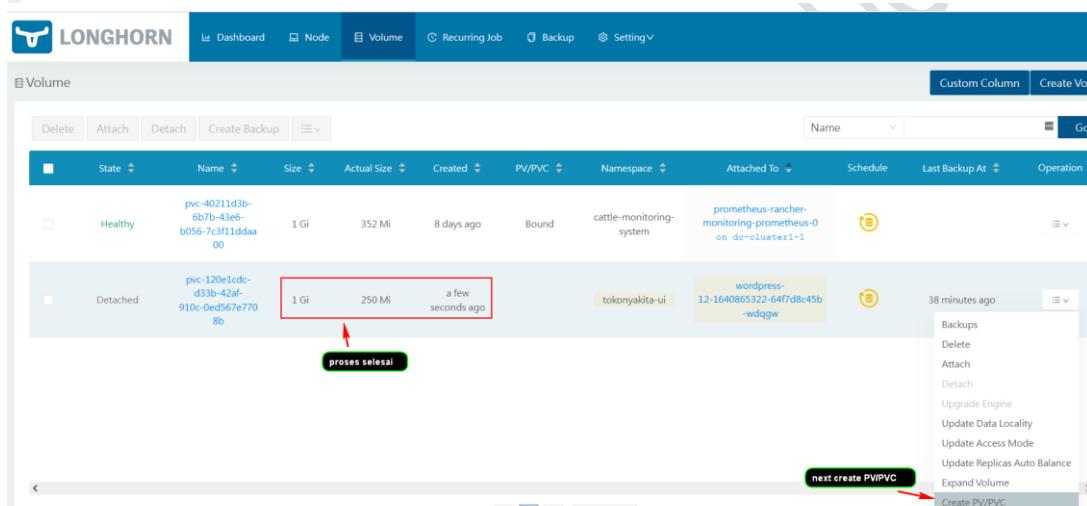
6.9.2 How to Restore

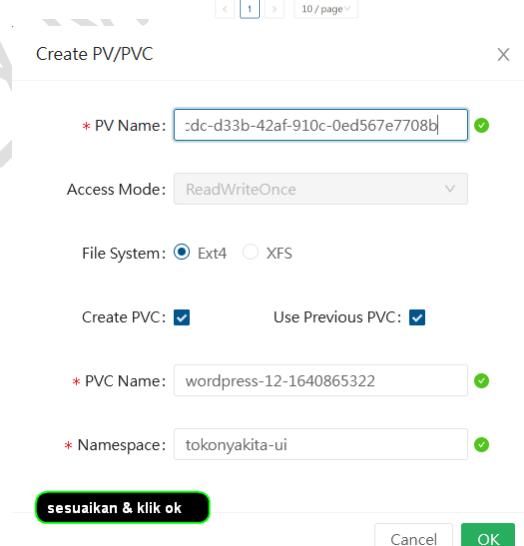
```
rancher-1:~ # kubectl get pod -n tokonyakita-ui
NAME                               READY   STATUS    RESTARTS   AGE
wordpress-12-1640865322-64f7d8c45b-7gbf8   1/1    Running   2          24m
wordpress-12-1640865322-mariadb-0           1/1    Running   0          19m
rancher-1:~ # kubectl scale deployment wordpress-12-1640865322 --replicas 0 -n tokonyakita-ui
deployment.apps/wordpress-12-1640865322 scaled
rancher-1:~ # kubectl scale statefulset wordpress-12-1640865322-mariadb --replicas 0 -n tokonyakita-ui
statefulset.apps/wordpress-12-1640865322-mariadb scaled
rancher-1:~ #
```

The screenshot illustrates the steps to restore a Persistent Volume (PV) and its corresponding Persistent Volume Claim (PVC) using the Longhorn UI.

- Lakukan scale out pod**: A green box highlights the command to scale out the WordPress deployment.
- 1. klik volume**: A red arrow points to the "Volume" tab in the Longhorn UI.
- 2. pilih volume**: A red box highlights the selection of a detached PV named "pvc-120e1cdc-d33b-42af-910c-0ed567e7708b".
- 3. klik ok**: A red arrow points to the "OK" button after confirming the deletion of the selected volumes.
- 1. klik backup**: A red arrow points to the "Backup" tab in the Longhorn UI.
- 2. pilih volume dan klik restore latest backup**: A red box highlights the selection of the same PV ("pvc-120e1cdc-d33b-42af-910c-0ed567e7708b") and a red arrow points to the "Restore Latest Backup" option in the context menu.
- 3. klik use previous name**: A red arrow points to the "Use Previous Name" checkbox in the "Restore Backup" dialog.
- 4. sesuaikan access mode dengan policy sebelumnya**: A red arrow points to the "Access Mode" dropdown set to "ReadWriteOnce". A yellow callout box provides a note about the restore volume name being the same as the backup volume.
- 5. klik ok**: A red arrow points to the "OK" button in the "Restore Backup" dialog.







Screenshot of a Kubernetes interface showing two Persistent Volumes (PVs) and their associated Persistent Volume Claims (PVCs).

	State	Name	Size	Actual Size	Created	PV/PVC	Namespace	Attached To	Schedule	Last Backup At	Operation
<input type="checkbox"/>	Detached	pvc-120e1cdc-d33b-42af-910c-0ed567e7708b	1 Gi	250 Mi	a minute ago	Bound	tokonyakita-ui			39 minutes ago	<input type="button" value="Backups"/> <input type="button" value="Delete"/> <input type="button" value="Attach"/> <input type="button" value="Detach"/> <input type="button" value="Upgrade Engine"/> <input type="button" value="Update Data Locality"/> <input type="button" value="Update Access Mode"/> <input type="button" value="Update Replicas Auto Balance"/> <input type="button" value="Expand Volume"/> <input type="button" value="Create PV/PVC"/>
<input type="checkbox"/>	Healthy	pvc-40211d3b-6b7b-43e6-b056-7c3f11ddaa00	1 Gi	352 Mi	8 days ago	Bound	cattle-monitoring-system	prometheus-rancher-monitoring-prometheus-0 on dc-cluster1-1			<input type="button" value="next, lakukan attach ke node"/> <input type="button" value="Attach"/> <input type="button" value="Detach"/> <input type="button" value="Upgrade Engine"/> <input type="button" value="Update Data Locality"/> <input type="button" value="Update Access Mode"/> <input type="button" value="Update Replicas Auto Balance"/> <input type="button" value="Expand Volume"/> <input type="button" value="Create PV/PVC"/>

Attach to host

* Host: dc-cluster1-1

Maintenance:

Cancel

bounding pv/pvc, done

prometheus-rancher-processes attach to host, done

```
rancher-1:~ # kubectl get pod -n tokonyakita-ui
No resources found in tokonyakita-ui namespace.
rancher-1:~ # kubectl scale statefulset wordpress-12-1640865322-mariadb --replicas 1 -n tokonyakita-ui
statefulset.apps/wordpress-12-1640865322-mariadb scaled
rancher-1:~ # kubectl scale deployment wordpress-12-1640865322 --replicas 1 -n tokonyakita-ui
deployment.apps/wordpress-12-1640865322 scaled
rancher-1:~ # kubectl get pod -n tokonyakita-ui
NAME                               READY   STATUS    RESTARTS   AGE
wordpress-12-1640865322-64f7d8c45b-2sqnq   0/1     Running   0          38s
wordpress-12-1640865322-mariadb-0           1/1     Running   0          85s
rancher-1:~ #
```

LONGHORN

Volume

Delete Attach Detach Create Backup

	State	Name	Size	Actual Size	Created	PV/PVC	Namespace	Attached To	Schedule	Last Backup At	Operation
<input type="checkbox"/>	Healthy	pvc-40211d3b-6b7b-43e6-b056-7c3f11ddaa00	1 Gi	353 Mi	8 days ago	Bound	cattle-monitoring-system	prometheus-rancher-monitoring-prometheus-0 on dc-cluster1-1			<input type="button" value="restore selesai"/>
<input type="checkbox"/>	Healthy	pvc-120e1cdc-d33b-42af-910c-0ed567e7708b	1 Gi	259 Mi	6 minutes ago	Bound	tokonyakita-ui	wordpress-12-1640865322-64f7d8c45b-2sqnq on dc-cluster1-1		44 minutes ago	<input type="button" value="Custom Column"/> <input type="button" value="Create Volume"/>
<input type="checkbox"/>	Healthy	pvc-a92be980-c255-4449-84b0-277adc2963e2	1 Gi	153 Mi	3 minutes ago	Bound	tokonyakita-ui	wordpress-12-1640865322-mariadb-0 on dc-cluster1-1		43 minutes ago	<input type="button" value="Go"/>

6.10 Disaster Recovery

6.10.1 Concept Section

A disaster recovery (DR) volume is a special volume that stores data in a backup cluster in case the whole main cluster goes down. DR volumes are used to increase the resiliency of Longhorn volumes.

Because the main purpose of a DR volume is to restore data from backup, this type of volume doesn't support the following actions before it is activated:

- Creating, deleting, and reverting snapshots
- Creating backups
- Creating persistent volumes
- Creating persistent volume claims

A DR volume can be created from a volume's backup in the backup store. After the DR volume is created, Longhorn will monitor its original backup volume and incrementally restore from the latest backup. A backup volume is an object in the backupstore that contains multiple backups of the same volume.

If the original volume in the main cluster goes down, the DR volume can be immediately activated in the backup cluster, so it can greatly reduce the time needed to restore the data from the backup store to the volume in the backup cluster.

When a DR volume is activated, Longhorn will check the last backup of the original volume. If that backup has not already been restored, the restoration will be started, and the activate action will fail. Users need to wait for the restoration to complete before retrying.

The Backup Target in the Longhorn settings cannot be updated if any DR volumes exist.

After a DR volume is activated, it becomes a normal Longhorn volume and it cannot be deactivated.

6.10.2 Requiment

Di sisi aplikasi :

- Arsitektur, deployment dan credentials di buat sama antara DC-DR

Di sisi Persistent Storage with longhorn :

- K8s downstream cluster pada DC-DR menggunakan longhorn untuk persistent storage
- DC-DR menggunakan target backup yang sama, support NFS atau S3
- Memiliki backup volume, baik dengan metode sekali backup maupun berulang

Memiliki inventory terkait persistent volume (PV) & persistent volume claim (PVC) yang related dengan aplikasi.

6.10.3 Kondisi Existing

- DC running apps tokonyakita.example.com dengan persistent storage with longhorn, load mengarah ke DC

```
> kubectl get all -n tokonyakita-ui
NAME                                     READY   STATUS    RESTARTS   AGE
pod/tokonyakita-ui-mariadb-0            1/1     Running   0          2m48s
pod/tokonyakita-ui-wordpress-6fd589bf7-n2jkk 1/1     Running   0          3m11s

NAME           TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)         AGE
service/tokonyakita-ui-mariadb   ClusterIP   10.43.122.73   <none>          3306/TCP       22h
service/tokonyakita-ui-wordpress LoadBalancer   10.43.150.116  192.168.137.50  80:31254/TCP,443:30995/TCP  22h

NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/tokonyakita-ui-wordpress 1/1     1           1           22h

NAME          DESIRED  CURRENT  READY   AGE
replicaset.apps/tokonyakita-ui-wordpress-6fd589bf7 1        1         1         22h

NAME          READY   AGE
statefulset.apps/tokonyakita-ui-mariadb 1/1     22h
>
> kubectl get pvc -A
NAMESPACE   NAME          STATUS    VOLUME   CAPACITY  ACCESS MODES  STORAGECLASS  AGE
cattle-monitoring-system   prometheus-rancher-monitoring-prometheus-db-prometheus-rancher-monitoring-prometheus-0   Bound    pvc-40211d3b-6b7b-43e6-b056-7c3f11ddaa00  1Gi     RWO          longhorn    11d
tokonyakita-ui   data-tokonyakita-ui-mariadb-0   Bound    pvc-87c2de61-b35a-1f43-9c49-e9ad4c3d0b64  1Gi     RWO          longhorn    22h
tokonyakita-ui   tokonyakita-ui-wordpress   Bound    pvc-d5c0c256-ac10-4414-af29-8f0bfbcecea95  1Gi     RWO          longhorn    22h
```

- DR standby apps tokonyakita.example.com

```
> kubectl get all -n tokonyakita-ui
NAME                                     READY   STATUS    RESTARTS   AGE
service/tokonyakita-ui-mariadb   ClusterIP   10.43.113.2   <none>          3306/TCP       22h
service/tokonyakita-ui-wordpress LoadBalancer   10.43.186.187  192.168.137.60  80:30465/TCP,443:30892/TCP  22h

NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/tokonyakita-ui-wordpress 0/0     0           0           22h

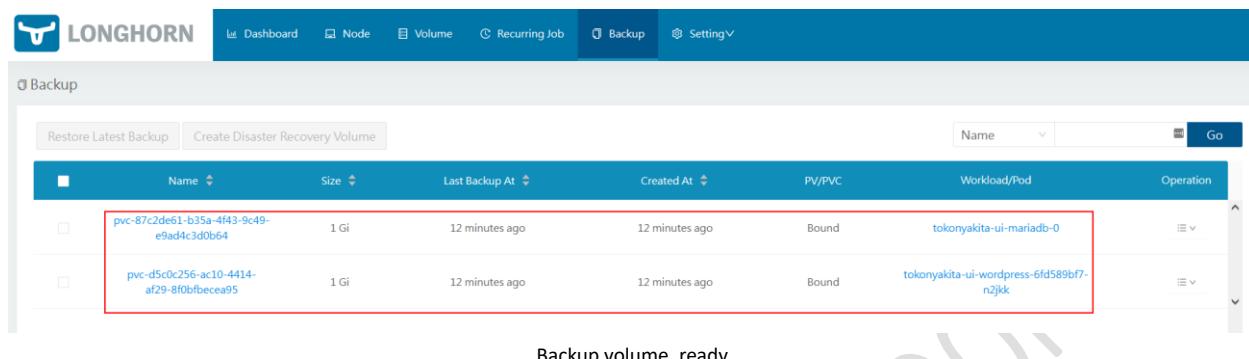
NAME          DESIRED  CURRENT  READY   AGE
replicaset.apps/tokonyakita-ui-wordpress-6fd589bf7 0        0         0         22h

NAME          READY   AGE
statefulset.apps/tokonyakita-ui-mariadb 0/0     22h
>
> kubectl get pvc -A
NAMESPACE   NAME          STATUS    VOLUME   CAPACITY  ACCESS MODES  STORAGECLASS  AGE
cattle-monitoring-system   prometheus-rancher-monitoring-prometheus-db-prometheus-rancher-monitoring-prometheus-0   Bound    pvc-32e1d75a-68f0-4bac-93d0-70b2cf860794  1Gi     RWO          longhorn    11d
> |
```

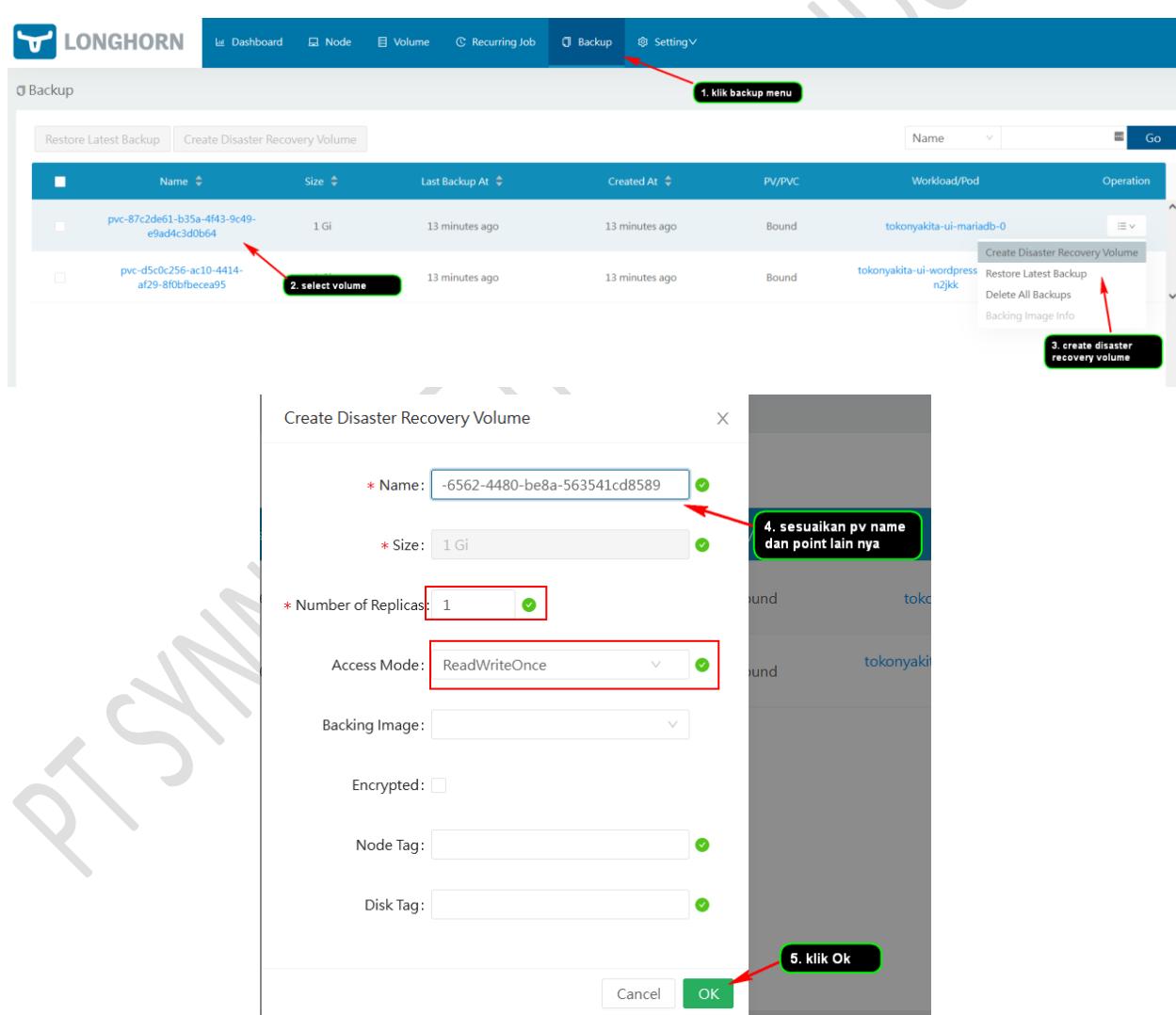
6.10.4 Use Case

- Metode aktif-passive
- DC down, perlu tindakan untuk up system di sisi DR
- Untuk peralihan koneksi bisa dengan metode pergantian ip publik dc/dr pada main domain

6.10.5 Operation



Backup volume, ready



1. klik backup menu

2. select volume

3. create disaster recovery volume

4. sesuaikan pv name dan point lain nya

5. klik Ok

Create volume pada disaster recovery, dari volume DC yang sudah ter-backup

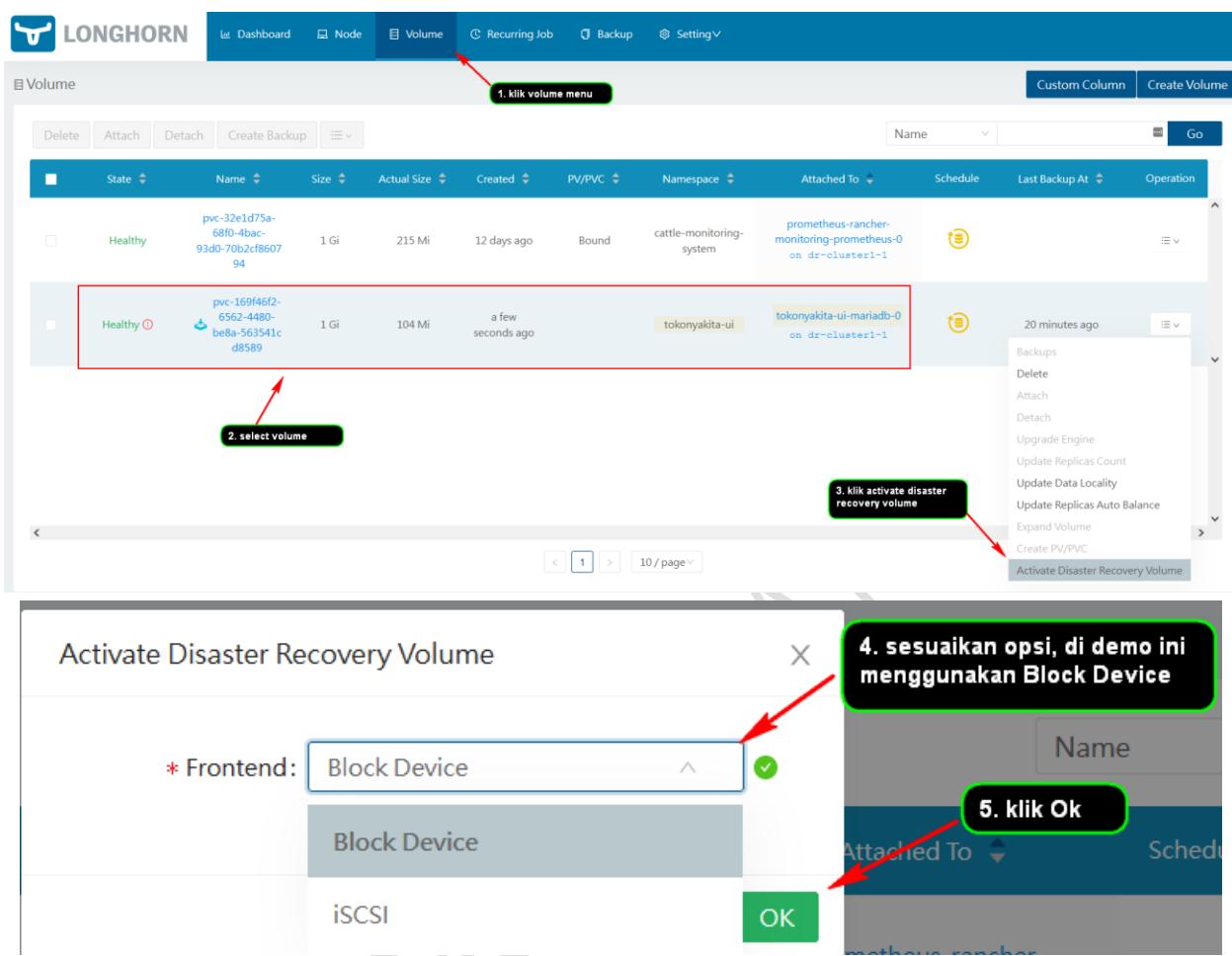
1. klik volume menu

2. select volume

3. klik activate disaster recovery volume

4. sesuaikan opsi, di demo ini menggunakan Block Device

5. klik Ok



The screenshot shows the LONGHORN interface with the 'Volume' tab selected. A red box highlights a volume entry, and a red arrow points to the 'Activate Disaster Recovery Volume' button in the context menu. Below, a modal dialog titled 'Activate Disaster Recovery Volume' shows a dropdown menu set to 'Block Device'. A red arrow points to the 'OK' button. The background shows a list of volumes with columns like State, Name, Size, Actual Size, Created, PV/PVC, Namespace, Attached To, Schedule, Last Backup At, and Operation.

1. klik volume menu

2. select volume

3. create pv/pvc

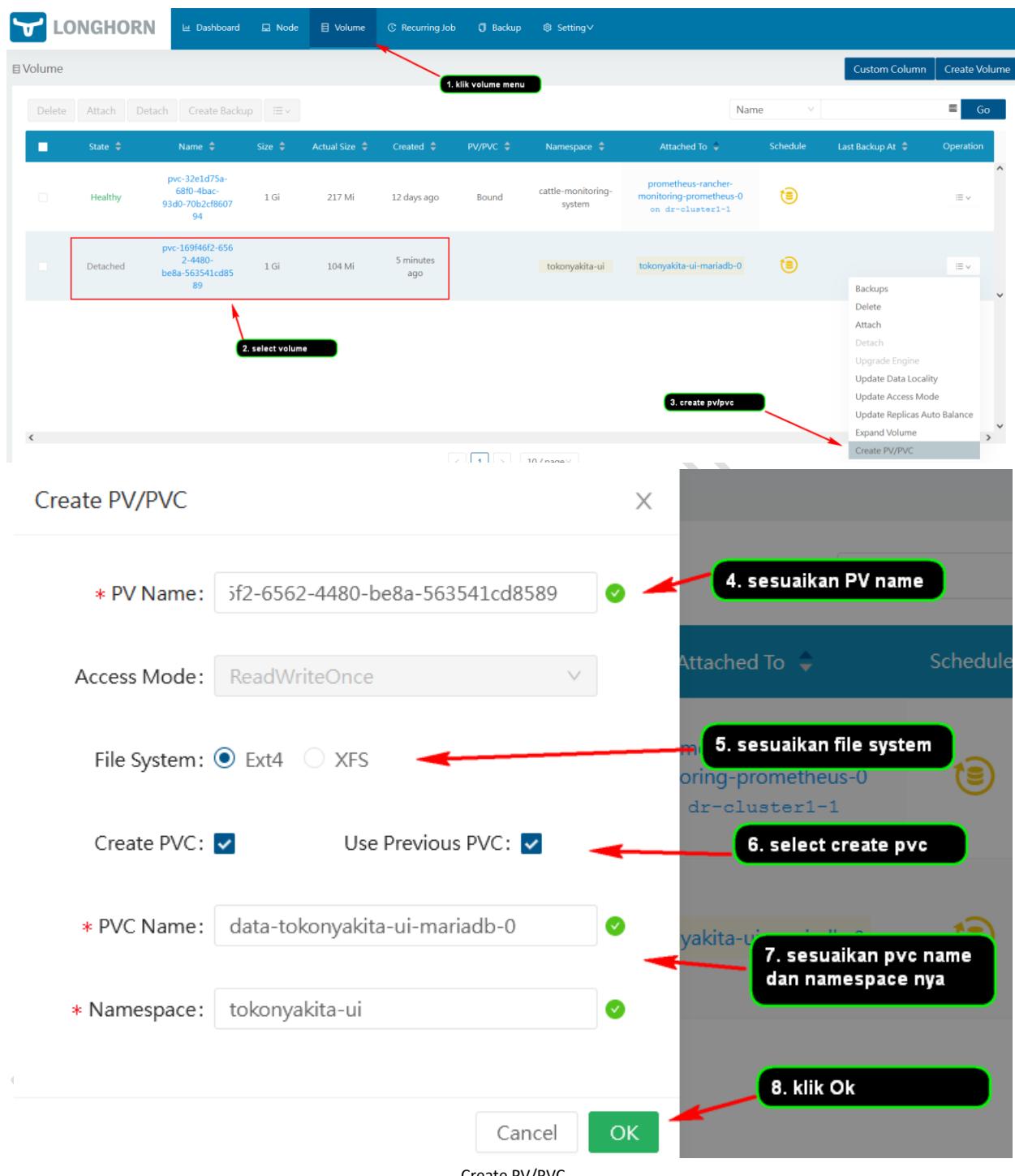
4. sesuaikan PV name

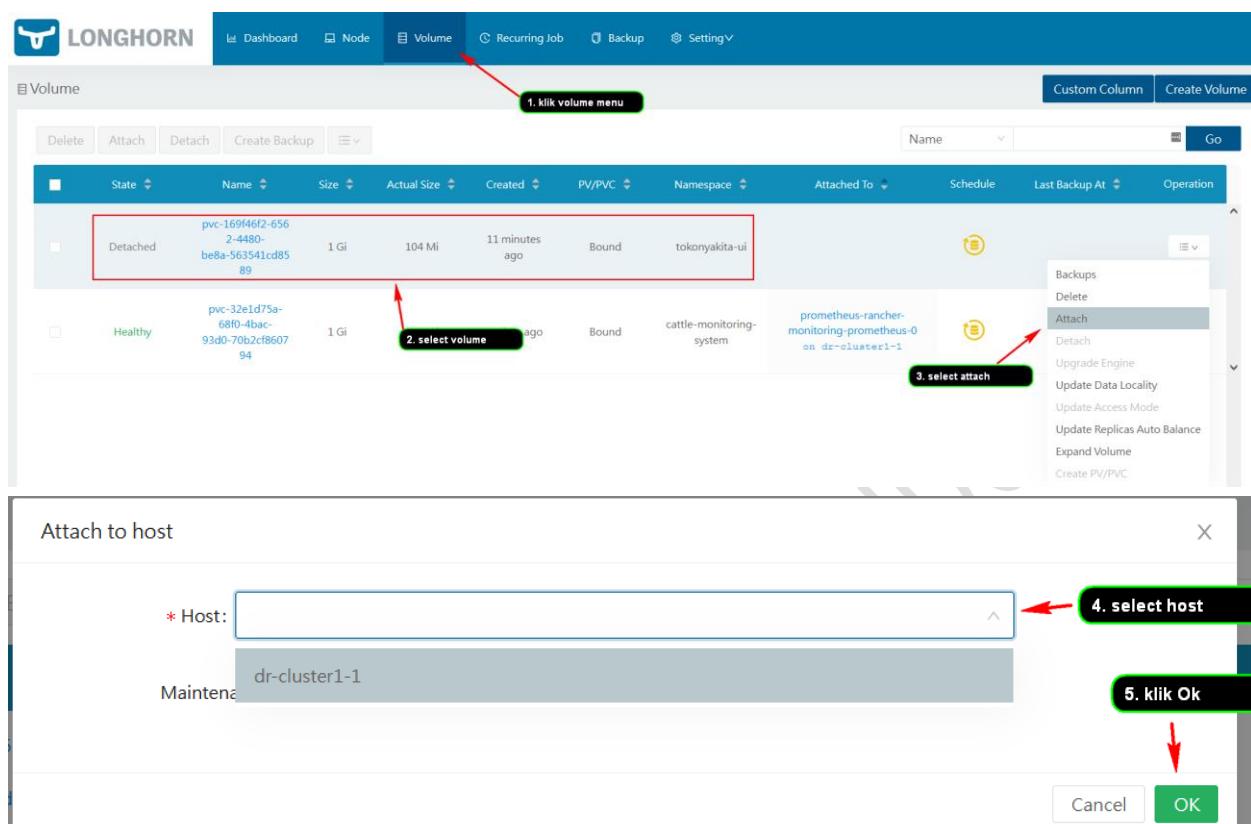
5. sesuaikan file system

6. select create pvc

7. sesuaikan pvc name dan namespace nya

8. klik Ok





The screenshot shows the LONGHORN UI interface for managing volumes. At the top, there are tabs: Dashboard, Node, Volume (highlighted with a red arrow), Recurring Job, Backup, and Setting. Below the tabs is a search bar with 'Volume' and buttons for Custom Column and Create Volume.

The main area displays a table of volumes:

	State	Name	Size	Actual Size	Created	PV/PVC	Namespace	Attached To	Schedule	Last Backup At	Operation
<input checked="" type="checkbox"/>	Detached	pvc-169f46f2-6562-4480-be8a-563541cd8589	1 Gi	104 Mi	11 minutes ago	Bound	tokonyakita-ui				
<input type="checkbox"/>	Healthy	pvc-32e1d75a-68f0-4bac-93d0-70b2cf860794	1 Gi		ago	Bound	cattle-monitoring-system	prometheus-rancher-monitoring-prometheus-0 on dr-cluster1-1			

A context menu is open for the first volume (pvc-169f46f2-6562-4480-be8a-563541cd8589). The menu items are:

- Backups
- Delete
- Attach** (highlighted with a green box)
- Detach
- Upgrade Engine
- Update Data Locality
- Update Access Mode
- Update Replicas Auto Balance
- Expand Volume
- Create PV/PVC

Below the table, a modal window titled "Attach to host" is displayed:

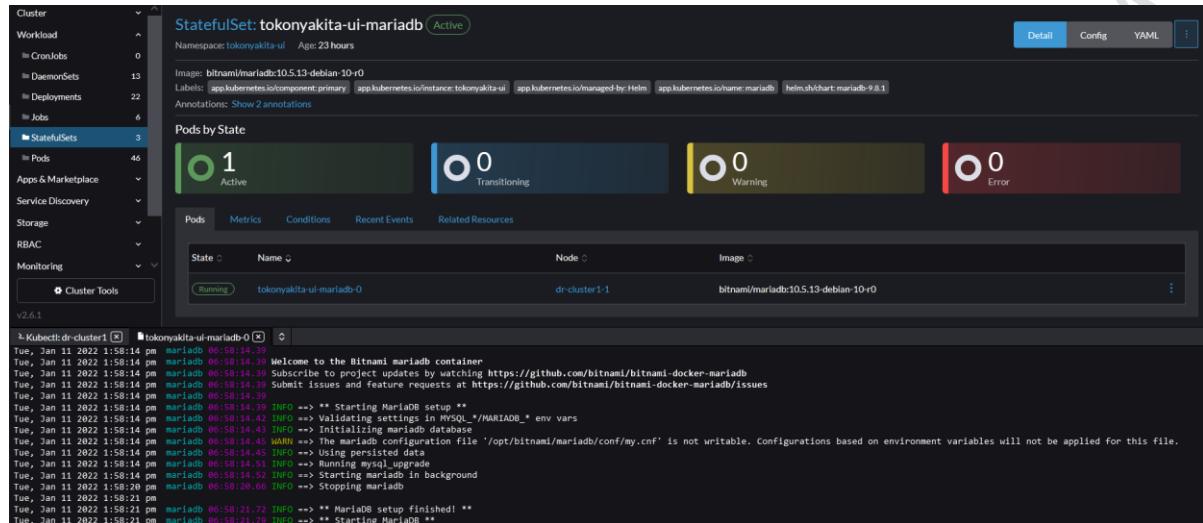
* Host: 

dr-cluster1-1

Cancel 

The modal has a red arrow pointing to the "Select host" input field labeled "4. select host". A green arrow points to the "OK" button labeled "5. klik Ok".

```
> kubectl get pod -n tokonyakita-ui
No resources found in tokonyakita-ui namespace.
>
> kubectl scale statefulset tokonyakita-ui-mariadb --replicas 1 -n tokonyakita-ui
statefulset.apps/tokonyakita-ui-mariadb scaled
>
> kubectl get pod -n tokonyakita-ui
NAME           READY   STATUS    RESTARTS   AGE
tokonyakita-ui-mariadb-0  1/1     Running   0          91s
>
```



StatefulSet: tokonyakita-ui-mariadb (Active)

Namespace: tokonyakita-ui Age: 23 hours

Image: bitnami/mariadb:10.5.13-debian-10-r0

Labels: app.kubernetes.io/component: primary, app.kubernetes.io/instance: tokonyakita-ui, app.kubernetes.io/managed-by: Helm, app.kubernetes.io/name: mariadb, helm.sh/chart: mariadb-9.1

Annotations: Show 2 annotations

Pods by State

State	Count	Status
Active	1	1 Active
Transitioning	0	0 Transitioning
Warning	0	0 Warning
Error	0	0 Error

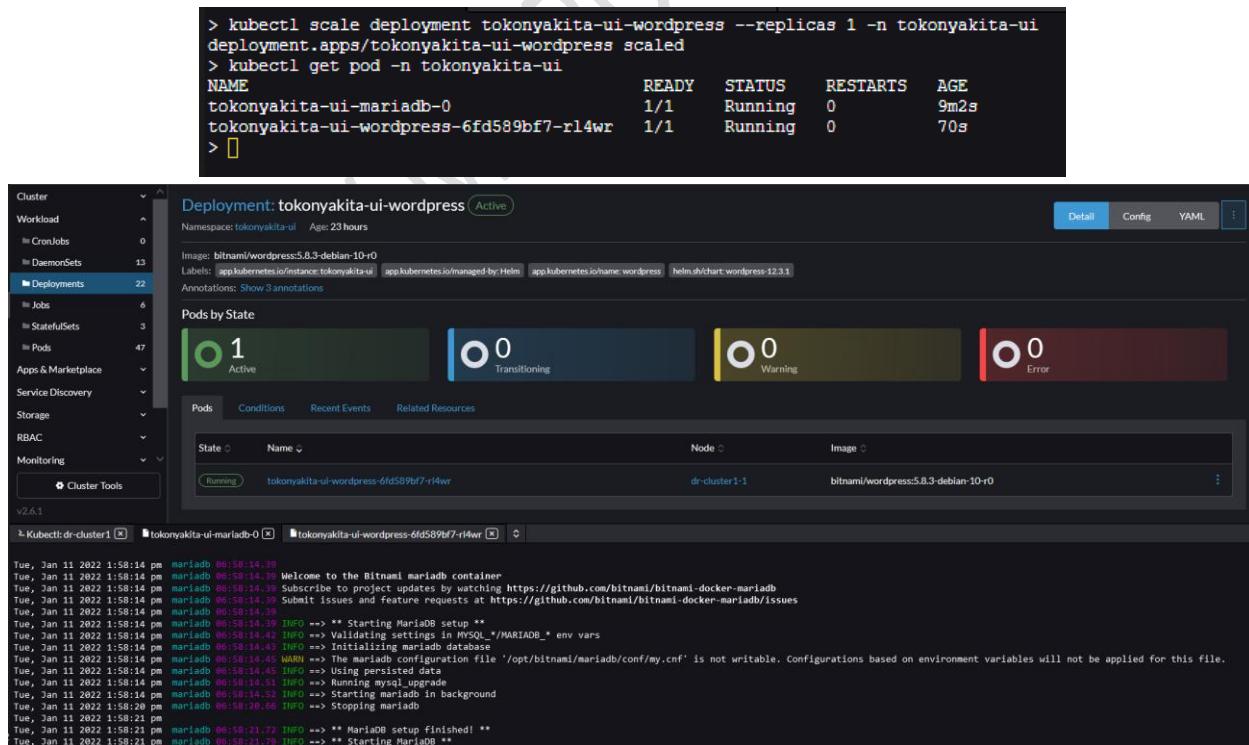
Pods Metrics Conditions Recent Events Related Resources

State	Name	Node	Image
Running	tokonyakita-ui-mariadb-0	dr-cluster1-1	bitnami/mariadb:10.5.13-debian-10-r0

```

Tue, Jan 11 2022 1:58:14 pm mariadb 06:58:14.39 Welcome to the Bitnami mariadb container
Tue, Jan 11 2022 1:58:14 pm mariadb 06:58:14.39 Subscribe to project updates by watching https://github.com/bitnami/bitnami-docker-mariadb
Tue, Jan 11 2022 1:58:14 pm mariadb 06:58:14.39 Submit issues and feature requests at https://github.com/bitnami/bitnami-docker-mariadb/issues
Tue, Jan 11 2022 1:58:14 pm mariadb 06:58:14.39 WARN >>> Using persisted data
Tue, Jan 11 2022 1:58:14 pm mariadb 06:58:14.39 INFO >>> Using mysql_upgrade
Tue, Jan 11 2022 1:58:14 pm mariadb 06:58:14.51 INFO >>> Starting mariadb in background
Tue, Jan 11 2022 1:58:20 pm mariadb 06:58:20.00 INFO >>> Stopping mariadb
Tue, Jan 11 2022 1:58:21 pm mariadb 06:58:21.72 INFO >>> ** MariaDB setup finished! **
Tue, Jan 11 2022 1:58:21 pm mariadb 06:58:21.78 INFO >>> ** Starting MariaDB **
```

Scale up database



Deployment: tokonyakita-ui-wordpress (Active)

Namespace: tokonyakita-ui Age: 23 hours

Image: bitnami/wordpress:5.8.3-debian-10-r0

Labels: app.kubernetes.io/instance: tokonyakita-ui, app.kubernetes.io/managed-by: Helm, app.kubernetes.io/name: wordpress, helm.sh/chart: wordpress-12.3.1

Annotations: Show 3 annotations

Pods by State

State	Count	Status
Active	1	1 Active
Transitioning	0	0 Transitioning
Warning	0	0 Warning
Error	0	0 Error

Pods Conditions Recent Events Related Resources

State	Name	Node	Image
Running	tokonyakita-ui-wordpress-6fd589bf7-rl4wr	dr-cluster1-1	bitnami/wordpress:5.8.3-debian-10-r0

```

> kubectl scale deployment tokonyakita-ui-wordpress --replicas 1 -n tokonyakita-ui
deployment.apps/tokonyakita-ui-wordpress scaled
> kubectl get pod -n tokonyakita-ui
NAME           READY   STATUS    RESTARTS   AGE
tokonyakita-ui-mariadb-0  1/1     Running   0          9m2s
tokonyakita-ui-wordpress-6fd589bf7-rl4wr  1/1     Running   0          70s
>
```

Tue, Jan 11 2022 1:58:14 pm mariadb 06:58:14.39 Welcome to the Bitnami mariadb container
Tue, Jan 11 2022 1:58:14 pm mariadb 06:58:14.39 Subscribe to project updates by watching https://github.com/bitnami/bitnami-docker-mariadb
Tue, Jan 11 2022 1:58:14 pm mariadb 06:58:14.39 Submit issues and feature requests at https://github.com/bitnami/bitnami-docker-mariadb/issues
Tue, Jan 11 2022 1:58:14 pm mariadb 06:58:14.39 INFO >>> Using persisted data
Tue, Jan 11 2022 1:58:14 pm mariadb 06:58:14.45 INFO >>> Using mysql_upgrade
Tue, Jan 11 2022 1:58:14 pm mariadb 06:58:14.51 INFO >>> Starting mariadb in background
Tue, Jan 11 2022 1:58:20 pm mariadb 06:58:20.00 INFO >>> Stopping mariadb
Tue, Jan 11 2022 1:58:23 pm mariadb 06:58:23.72 INFO >>> ** MariaDB setup finished! **
Tue, Jan 11 2022 1:58:23 pm mariadb 06:58:23.78 INFO >>> ** Starting MariaDB **

Lakukan langkah yang sama untuk up aplikasi

Health	Name	Size	Storage Class	Last Updated	Status	Owner	Annotations	Image
Healthy	pvc-169f46f2-6562-4480-be8a-563541cd8589	1 Gi	126 Mi	27 minutes ago	Bound	tokonyakita-ui	tokonyakita-ui-mariadb-0 on dr-cluster1-1	
Healthy	pvc-afbd4254-dc5a-46be-bb66-7c9d33d08ea0	1 Gi	262 Mi	7 minutes ago	Bound	tokonyakita-ui	tokonyakita-ui-wordpress-6fd589bf7-rl4wr on dr-cluster1-1	

Volume Ok, sudah dalam status healthy

New Arrivals

SHOE COLLECTION

Pi Demo ini, DC menggunakan ip 50, DR menggunakan ip 60



Network & Security			Server & Storage			Data Protection		
Data & Analytics			IaaS (Infrastructure as a Service)			Printing, Peripheral & Tools		
Devices & Smartphone			Memory, Flash & Components			Electronics		
PC End Computing								
Gaming & Accessories						Rack, Power & Cooling		
Education			EC-Council			Indonesia Technology Solution		