



Deploy Rancher and K8s in Your Own Data Center with QCT's Azure Stack HCI solution

Written by Bono Hsueh

Last Update: 03/19/2020

Version: 1.3



Abstract

Azure Stack (AzS) HCI is an invitation-only program by Microsoft in which solution providers design hyper-converged infrastructures with Windows Server technologies.

After going through Microsoft's validation and requirements, all the systems in this build are preconfigured, tested, and certified by QCT.

Microsoft Cloud Ready Appliances integrate all of Microsoft Windows Server 2019 SDDC technologies—software-defined compute, storage and networking virtualization and security—into hyper-converged and disaggregated appliances. These optimized modular appliances are easy to use, agile, and scalable, creating a cloud-ready platform from four nodes to sixteen nodes in the same cluster to handle a variety of virtualization workloads and to address current needs while preparing for future requirements.

REVISIONS

Version	Date	Description	Authors
1.0	01/10/2020	First Published	Bono Hsueh
1.1	02/06/2020	Added Materials	Bono Hsueh
1.2	03/06/2020	Added Materials	Bono Hsueh
1.3	03/19/2020	Added Materials	Bono Hsueh



CONTENTS

Abstract	1
REVISIONS	2
CONTENTS	3
The Azure Stack HCI solution	4
Introducing the Azure Stack HCI Technical Use Cases	5
The best-performing HCI solution to suit your needs, amplified by hybrid services....	5
How to Build an Enterprise Kubernetes	7
A Multi-cluster Future for Cloud Applications	8
QCT Deployment Lab	9
1. Hardware for this Guide	9
2. Server Nodes / Network Diagram	10
3. Prepare the QCT physical server node	11
Partnering with Rancher on your Kubernetes Journey	12
Install Production Kubernetes Cluster with Rancher RKE	12
What is RKE?	12
Prepare Workstation machine	12
High Availability (HA) Install/ Production environments	13
Install Kubernetes with RKE&Rancher GUI	14
Step 1: Update your Linux System.....	14
Step 2: Create rke user	14
Step 3: Enable required Kernel modules:.....	15
Step 4: Disable swap and Modify sysctl entries	16
Step 5: Install Supported version of Docker	17
Step 6: Open Ports on firewall.....	17
Step 7: Allow SSH TCP Forwarding	18
Step 8: Generate RKE cluster configuration file.	18
Step 9: Deploy Kubernetes Cluster with RKE	24
Step 10: Accessing your Kubernetes cluster	24
Step 11: Accessing from Rancher GUI Web tool	25
About QCT	26

The Azure Stack HCI solution

AzS HCI is offered through Microsoft hardware partners, pre-built, and typically either preconfigured or bundled with simple configuration software. More than 175 AzS HCI solutions are available today from at least 20 Microsoft hardware partners.

These partners offer hardware configurations that Microsoft has validated to ensure optimal performance and reliability for AzS HCI.

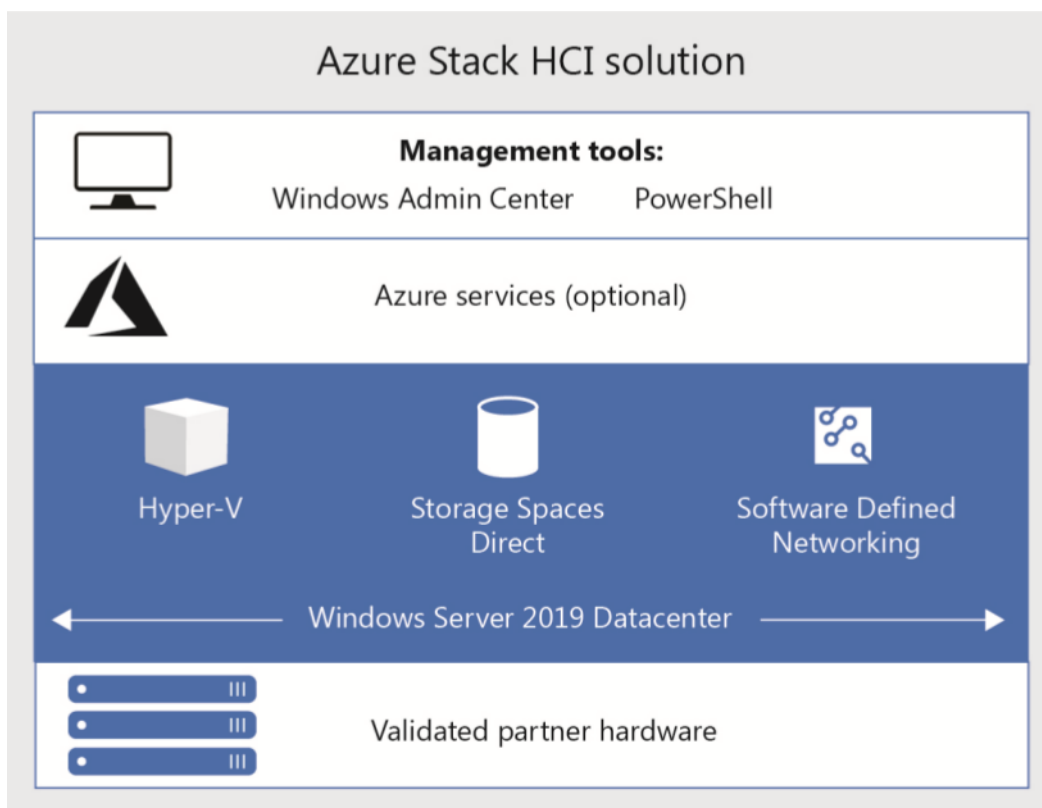
AzS HCI combines this pre-validated hardware from Microsoft partners with the following Windows Server 2019 Datacenter components and management tools:

Windows Server 2019 Datacenter roles and features:

- **Hyper-V** to run virtual machines (VMs) on all physical hosts
- **Software Defined Networking (SDN) (optional)** for network virtualization
- **Storage Spaces Direct** for software-defined storage

Management tools:







- **Windows Admin Center** for central, comprehensive management of local and remote servers through a graphical interface
- **Azure services** integrated into Windows Admin Center for optional offsite backups, site recovery, cloud-based monitoring, and other benefits
- **PowerShell** for scripting and automation



Introducing the Azure Stack HCI technical use cases

The Microsoft AzS HCI Technical Use Case program enables you to quickly find hardware configurations for AzS HCI that Microsoft has certified for various needs.

Through the Azure Stack HCI Technical Use Case program, Microsoft certifies AzS HCI solution offerings to validate the suitability of their hardware configurations in supporting any of these scenarios.

-  "Branch office and edge" (2 node resilient edge including IoT Edge)
-  "Virtual desktop infrastructure" (optimized for desktop serving)
-  "High-performance Microsoft SQL Server" (optimized for HA SQL on HCI)
-  "Trusted enterprise virtualization" (leverage VBS and Assurance AQ)
-  "Scale-out storage" (SAN replace, file aggregation and consolidation)
-  "General-purpose Kubernetes" (Kubernetes optimized on HCI)

The best-performing HCI solution to suit your needs, amplified by hybrid services

AzS HCI offers industry-leading performance for HCI at an affordable price, which is the simple cost of a Windows Server 2019 Datacenter license. AzS HCI is also the only HCI solution that includes optional cloud-based enhancements to ease administration.

You can use the Technical Use Case program to easily identify AzS HCI solutions that are built for your technical use case. Whether you need an HCI solution to run edge workloads, VDI, SQL Server, highly secure workloads, or a simple file server, you can find an AzS HCI solution that meets your requirements on premises while also offering optional hybrid services through Azure.

To find AzS HCI solutions from hardware vendors that have been approved for the most common customer needs for HCI, view the [Azure Stack HCI catalog](#).



Branch office and edge

Use case: Meet the evolving IT demands of branch offices, retail stores, and field locations. Deploy your container-built edge workloads and essential business applications in highly available virtual machines (VMs) and utilize Azure Monitor to get a global view of system health.



Virtual desktop infrastructure

Use remote desktop services, highly available virtual machines, and integrated scalable storage to support your large-scale virtual implementations. Connect your virtual desktops to the Azure Update Management solution to control your enterprise virtual desktop infrastructure configuration in the cloud.



High-performance Microsoft SQL Server

Deploy and manage high-performance, scalable Microsoft SQL Server running on hyperconverged infrastructure. Run your applications with high availability and leverage the flexibility of virtualization. Utilize Azure Backup and Azure Blob Storage to backup and restore your organization's data.



Trusted enterprise virtualization

Leverage Virtualization-based security and Assurance-certified hardware to protect your security sensitive workloads. Connecting this trusted infrastructure to Azure Security Center activates behavioral analytics and reports to account for rapidly changing workloads and threats.



Scale-out storage

Combine unmatched storage performance with validated hardware to centralize your organization's file shares. Azure File Sync keeps the flexibility, performance, and compatibility of an on-premises file server while replacing an aging (SAN) storage area network or aggregating files from multiple sources.



General-purpose Kubernetes

Enable automated deployment, scaling and management of containerized applications by running a Kubernetes cluster on your hyperconverged infrastructure. Utilize Azure Monitor for containers to deliver a comprehensive monitoring experience to understand the performance and health of your clusters, from infrastructure to applications.



How to Build an Enterprise Kubernetes

Organizations love Kubernetes because it helps significantly increase the agility and efficiency of their software development teams, enabling them to reduce the time and perils associated with putting new software into production.

Information technology operations teams love Kubernetes because it helps boost productivity, reduce costs and risks, and moves organizations closer to achieving their hybrid cloud goals.

Simply put, Kubernetes makes it easier to manage software complexity. As enterprise applications become more complex, development and operations (DevOps) teams need a tool that can orchestrate that complexity. They need a way to launch all the services dependent on these applications, making sure the applications and services are healthy and can connect to one another.

Containers have dramatically risen in popularity because they provide a consistent way to package application components and their dependencies into a single object that can run in any environment. By packaging code and its dependencies into containers, a development team can use standardized units of code as consistent building blocks. The container will run the same way in any environment and can start and terminate quickly, allowing applications to scale to any size.

Kubernetes is an open source container orchestration platform, allowing large numbers of containers to work together in harmony, and reducing operational burdens. In fact, Kubernetes, originally developed by Google and now managed by the Cloud Native Computing Foundation (CNCF), has become a standard for cloud container orchestration, providing a platform for automating deployment, scaling and operations of application containers across multiple clusters of hosts.

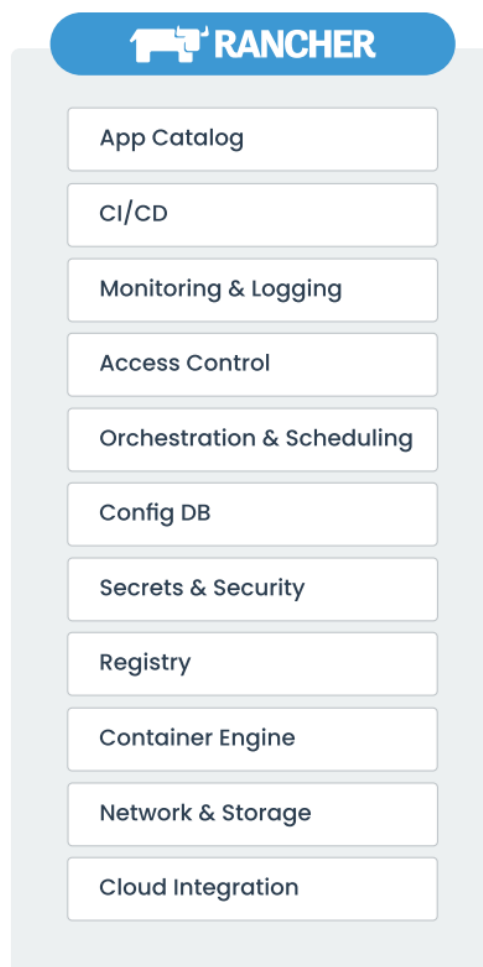
A Multi-cluster Future for Cloud Applications

Rancher's enterprise-friendly features include a built-in application catalog, integrated monitoring and logging, and superior RBAC.

Having these available from the beginning gets you off to a great start as you scale your deployment.

When you and your team negotiate spinning up your first Docker containers, you'll come to realize that running workloads on multiple servers isn't Docker's strength.

Instead, Kubernetes is the best tool when it comes to container cluster management: it runs the right containers at the right time, scales them up and down according to load, deals with hardware or container failure, and manages networking and storage.

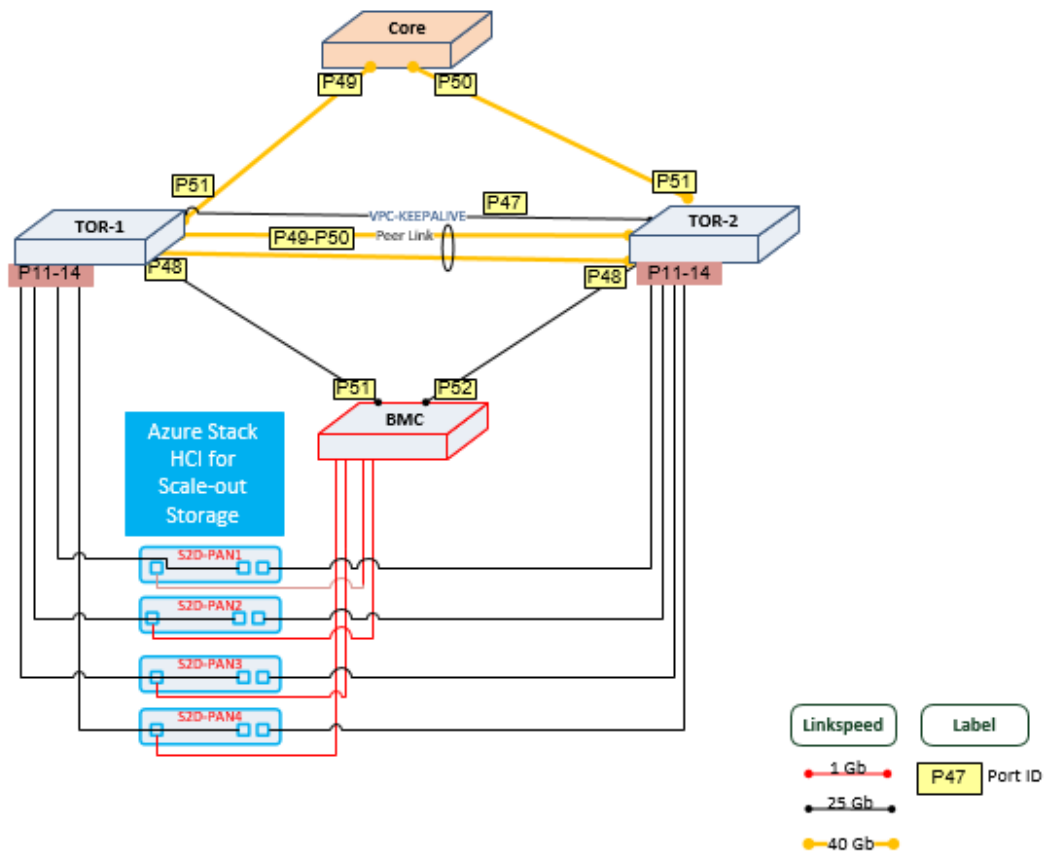


QCT Deployment Lab

1. Hardware for this Guide

QuantaGrid D52BQ-2U (2~4 nodes) (Alias name: S5BQ)				
SKU	Description	Qty per unit	Total	Version
Server Platform	D52BQ-2U	1	4	BIOS: S2P_3B10 BMC: 3.33
CPU	Intel Xeon Gold 5118 CPUs (2.3GHz, 12-core, 16.5MB cache)	2	8	
Memory	Samsung 32GB DDR4 2666MHz ECC-Register DIMMs	16	64	
Cache	Samsung 1.92TB 2.5" SATA SSD	4	16	104Q
Storage	Seagate 8TB 3.5" SATA HDD (ST8000NM0055)	8	32	PN01
Boot Drive	Intel 480G 2.5" SATA SSD	1	4	G2010140
HBA Card	QCT LSI SAS 9305-16i - IT firmware mode	1	4	FW: 1B03
NIC Card	Mellanox Quanta OCP Mezz CX4, Dual Port 25G	1	4	FW: 14.24.1000

2. Server Nodes / Network Diagram





3. Prepare the QCT physical server node

Best practices dictate that with a new server deployment, the first task is to review the system firmware and drivers relevant to the incoming operating system. If the system has the latest firmware and drivers installed, it will expedite tech support calls and may reduce the need for such calls.

<https://qct.io/product/index/Storage/Storage-Server/4U-Storage-Server/QuantaPlex-T21P-4U#download>



Partnering with Rancher on your Kubernetes Journey

Install Production Kubernetes Cluster with Rancher RKE

Kubernetes has gained much traction and is now the standard orchestration layer for containerized workloads. If you want an open-source system for automating deployment of containerized applications without worrying about scaling and management, then Kubernetes is the right tool for you.

This guide walks you through the simple steps for installing a production-grade Kubernetes cluster with RKE. We'll set up a 5-node cluster with Rancher Kubernetes Engine (RKE) and install the *Rancher chart* with the Helm package manager.

What is RKE?

Rancher Kubernetes Engine (RKE) is an extremely simple, lightning fast Kubernetes distribution that runs entirely within containers. Rancher is a container management platform built for organizations that deploy containers in production. Rancher makes it easy to run Kubernetes everywhere, meet IT requirements, and empower DevOps teams.

Prepare Workstation machine

A number of CLI Tools are required on your Workstation where deployment will be done. This can also be a virtual machine that is able to access cluster nodes.

1. kubectl:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s \
https://storage.googleapis.com/kubernetes-
release/release/stable.txt`/bin/linux/amd64/kubectl

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin/kubectl

kubectl version --client
```

2. rke

```
curl -s https://api.github.com/repos/rancher/rke/releases/latest | grep \
download_url | grep amd64 | cut -d '"' -f 4 | wget -qi -

chmod +x rke_linux-amd64

sudo mv rke_linux-amd64 /usr/local/bin/rke

rke --version
```

3. helm

```
curl -fsSL -o get_helm.sh \
https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3

chmod 700 get_helm.sh

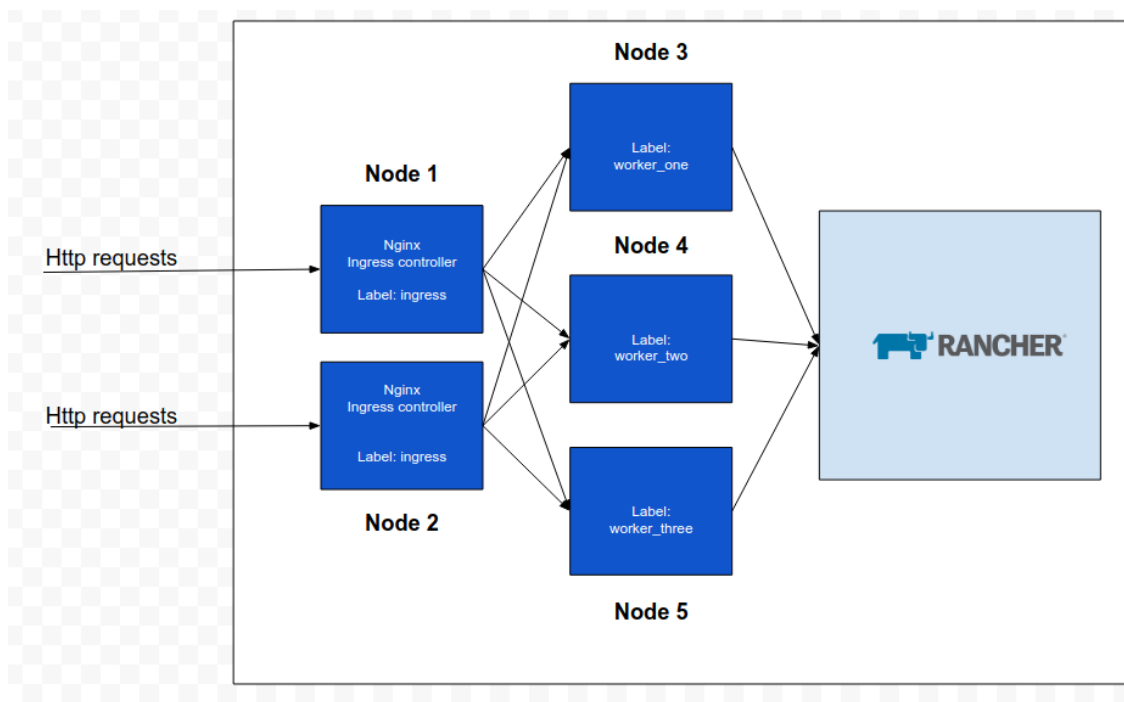
./get_helm.sh
```

High Availability (HA) Install/ Production environments

You can install rancher on a single node but this is only for development and testing as recommended by Rancher. Here we will install Rancher for production environments. Rancher recommends to have a least 3 nodes for HA installation, more details can be found [here](#).

In our case we will install Rancher with 5 nodes, so we will change the configuration made in the [official tutorial](#) by Rancher to support our case.

The following diagram shows our architecture:





Install Kubernetes with RKE&Rancher GUI

I'll be working on 5 nodes:

- **Master Nodes** – etcd and control plane (3 for HA)
- **Worker nodes** – Scale to meet your Workloads demand

These are the specifications for my setup.

- Master Nodes – **8GB** of RAM and **4** vcpus
- Worker Machines – **16GB** of RAM and **8** vcpus

Step 1: Update your Linux System

```
$ sudo yum -y update
$ sudo reboot
```

Step 2: Create rke user

If using Red Hat Enterprise Linux, Oracle Enterprise Linux or CentOS, you cannot use the root user as SSH user due to Bugzilla 1527565. For this reason, we'll create a user account called rke for deployment purposes.

Login to each of your cluster nodes and create rke user.

```
$ sudo useradd rke
$ sudo passwd rke
```

Enable passwordless sudo for the user:

```
$ sudo vi /etc/sudoers.d/rke
rke ALL=(ALL:ALL) NOPASSWD: ALL
```

Copy your ssh public key to the user's ~/.ssh/authorized_keys file.

```
$ for i in rke-master-01 rke-master-02 rke-master-03 rke-worker-01 \
    rke-worker-02; do
    ssh-copy-id rke@$i
done
```

Confirm you can login from your workstation:

```
$ ssh rke@rke-master-01
Warning: Permanently added 'rke-master-01,x.x.x.x' (ECDSA) to the list of known
hosts.
[rke@rke-master-01 ~]$ sudo su -      # No password prompt
[root@rke-master-01 ~]# exit
[rke@rke-master-01 ~]$ exit

logout
Connection to rke-master-01 closed.
```

Step 3: Enable required Kernel modules:

Login to each host and enable Kernel modules required to run Kubernetes.

```
for module in br_netfilter ip6_udp_tunnel ip_set ip_set_hash_ip
ip_set_hash_net iptable_filter iptable_nat iptable_mangle iptable_raw
nf_conntrack_netlink nf_conntrack nf_conntrack_ipv4 nf_defrag_ipv4 nf_nat
nf_nat_ipv4 nf_nat_masquerade_ipv4 nfnetlink udp_tunnel veth vxlan x_tables
xt_addrtype xt_conntrack xt_comment xt_mark xt_multiport xt_nat xt_recent
xt_set xt_statistic xt_tcpudp;
do
    if ! lsmod | grep -q $module; then
        echo "module $module is not present";
        modprobe $module;
    fi;
done
```

Step 4: Disable swap and Modify sysctl entries

The recommendation of Kubernetes is to disable swap and add some sysctl values.

Swap:

```
$ sudo vim /etc/fstab
# Add comment to swap line
$ sudo swapoff -a
```

Sysctl:

```
$ sudo tee -a /etc/sysctl.d/99-kubernetes.conf <<EOF
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
$ sysctl --system
```

Confirm SWAP is disabled:

```
$ free -h
```

	total	used	free	shared	buff/cache	available
Mem:	7.6G	180M	6.8G	8.5M	633M	7.2G
Swap:	0B	0B	0B			



Step 5: Install Supported version of Docker

Each Kubernetes version supports different Docker versions. The Kubernetes release notes contain the current list of validated Docker versions.

You can either follow the [Docker installation](#) instructions or use one of Rancher's [install scripts](#) to install Docker. To install the latest supported version:

```
curl https://releases.rancher.com/install-docker/18.09.2.sh | sudo bash -
```

Start and enable docker service:

```
sudo systemctl enable --now docker
$ sudo docker version --format '{{.Server.Version}}'
18.09.2
```

Add **rke** user to docker group.

```
$ sudo usermod -aG docker rke
$ id rke
uid=1000(rke) gid=1000(rke) groups=1000(rke),994(docker)
```

Step 6: Open Ports on firewall

For a single-node installation, you only need to open the ports required to enable Rancher to communicate with downstream user clusters.

For a high-availability installation, the same ports need to be opened, as well as additional ports required to set up the Kubernetes cluster that Rancher is installed on.

Check all ports used in the [requirements page](#)

Firewalld TCP ports:

```
for i in 22 80 443 179 5473 6443 8472 2376 8472 2379-2380 9099 10250 10251
10252 10254 30000-32767; do
    sudo firewall-cmd --add-port=${i}/tcp --permanent
done
sudo firewall-cmd --reload
```

Firewalld UDP ports:

```
for i in 8285 8472 4789 30000-32767; do
  sudo firewall-cmd --add-port=${i}/udp --permanent
done
```

Step 7: Allow SSH TCP Forwarding

You need to enable your SSH server system-wide TCP forwarding.

Open the ssh configuration file located at `/etc/ssh/sshd_config`:

```
$ sudo vi /etc/ssh/sshd_config
AllowTcpForwarding yes
```

Restart ssh services after making the change.

```
$ sudo systemctl restart sshd
```

Step 8: Generate RKE cluster configuration file

RKE uses a cluster configuration file, referred to as `cluster.yml` to determine what nodes will be in the cluster and how to deploy Kubernetes.

There are [many configuration options](#) that can be set in the `cluster.yml`. This file can be created from [minimal example](#) templates or generated with the `rke config` command.

Run the `rke config` command to create a new `cluster.yml` in your current directory.

```
rke config --name cluster.yml
```

This command will prompt you for all the information needed to build a cluster.

If you want to create an empty template `cluster.yml` file instead, specify the `--empty` flag.

```
rke config --empty --name cluster.yml
```

The sample of a cluster configuration file looks like :

```
# https://rancher.com/docs/rke/latest/en/config-options/  
nodes:  
- address: 10.106.6.101  
  internal_address:  
  hostname_override: rke-master-01  
  role: [controlplane, etcd]  
  user: rke  
- address: 10.106.5.102  
  internal_address:  
  hostname_override: rke-master-02  
  role: [controlplane, etcd]  
  user: rke  
- address: 10.106.5.103  
  internal_address:  
  hostname_override: rke-master-03  
  role: [controlplane, etcd]  
  user: rke  
- address: 10.106.5.104  
  internal_address:  
  hostname_override: rke-worker-01  
  role: [worker]  
  user: rke
```

```
- address: 10.106.5.105

internal_address:

hostname_override: rke-worker-02

role: [worker]

user: rke

# using a local ssh agent

# Using SSH private key with a passphrase - eval `ssh-agent -s` && ssh-add

ssh_agent_auth: true

# SSH key that access all hosts in your cluster

ssh_key_path: ~/.ssh/id_rsa

# By default, the name of your cluster will be local

# Set different Cluster name

cluster_name: rke

# Fail for Docker version not supported by Kubernetes

ignore_docker_version: false

# prefix_path: /opt/custom_path
```

```
# Set kubernetes version to install:
https://rancher.com/docs/rke/latest/en/upgrades/#listing-supported-kubernetes-versions

# Check with -> rke config --list-version --all

kubernetes_version:

# Etcd snapshots

services:

  etcd:

    backup_config:

      interval_hours: 12

      retention: 6

    snapshot: true

    creation: 6h

    retention: 24h

  kube-api:

    # IP range for any services created on Kubernetes

    # This must match the service_cluster_ip_range in kube-controller

    service_cluster_ip_range: 10.43.0.0/16

    # Expose a different port range for NodePort services

    service_node_port_range: 30000-32767

    pod_security_policy: false
```

kube-controller:

CIDR pool used to assign IP addresses to pods in the cluster

cluster_cidr: 10.42.0.0/16

IP range for any services created on Kubernetes

This must match the service_cluster_ip_range in kube-api

service_cluster_ip_range: 10.43.0.0/16

kubelet:

Base domain for the cluster

cluster_domain: cluster.local

IP address for the DNS service endpoint

cluster_dns_server: 10.43.0.10

Fail if swap is on

fail_swap_on: false

Set max pods to 150 instead of default 110

extra_args:

max-pods: 150

Configure network plug-ins

KE provides the following network plug-ins that are deployed as add-ons:
flannel, calico, weave, and canal

After you launch the cluster, you cannot change your network provider.

Setting the network plug-in

network:

plugin: canal

options:

canal_flannel_backend_type: vxlan

```
# Specify DNS provider (coredns or kube-dns)

dns:
  provider: coredns

# Currently, the only authentication strategy supported is x509.
# You can optionally create additional SANs (hostnames or IPs) to
# add to the API server PKI certificate.
# This is useful if you want to use a load balancer for the
# control plane servers.
authentication:
  strategy: x509
  sans:
    - "k8s.ws19demo.qct"

# Set Authorization mechanism
authorization:
  # Use `mode: none` to disable authorization
  mode: rbac

# Currently only nginx ingress provider is supported.
# To disable ingress controller, set `provider: none`
# `node_selector` controls ingress placement and is optional
ingress:
  provider: nginx
  options:
    use-forwarded-headers: "true"
```



Step 9: Deploy Kubernetes Cluster with RKE

Once you've created the cluster.yml file, you can deploy your cluster with a simple command.

```
$ rke up --config ./rancher_cluster.yml
```

Step 10: Accessing your Kubernetes cluster

As part of the Kubernetes creation process, a **kubeconfig** file has been created and written at **kube_config_cluster.yml**.

Set **KUBECONFIG** variable to the file generated.

```
export KUBECONFIG=./kube_config_cluster.yml
```

Check list of nodes in the cluster.

```
$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
rke-master-01	Ready	controlplane,etcd	16m	v1.17.0
rke-master-02	Ready	controlplane,etcd	16m	v1.17.0
rke-master-03	Ready	controlplane,etcd	16m	v1.17.0
rke-worker-01	Ready	worker	6m33s	v1.17.0
rke-worker-02	Ready	worker	16m	v1.17.0

You can copy this file to `$HOME/.kube/config` if you don't have any other kubernetes cluster.

Step 11: Accessing from Rancher GUI Web tool

Before installing rancher, let's install cert-manager to automatically provision and manage TLS certificates in Kubernetes

```
# Install custom resource definition

kubectl apply -f https://raw.githubusercontent.com/jetstack/cert-
manager/release-0.11/deploy/manifests/00-crds.yaml

# Create namespace

kubectl create namespace cert-manager

# Label namespace to disable resource validation

kubectl label namespace cert-manager certmanager.k8s.io/disable-
validation=true

# Add the Jetstack repo

helm repo add jetstack https://charts.jetstack.io

# Update repo

helm repo update

# Install cert manager

helm install --name cert-manager --namespace cert-manager --version v0.9.1
jetstack/cert-manager

# Check if cert manager is deployed and running correctly

kubectl get pods --namespace cert-manager
```

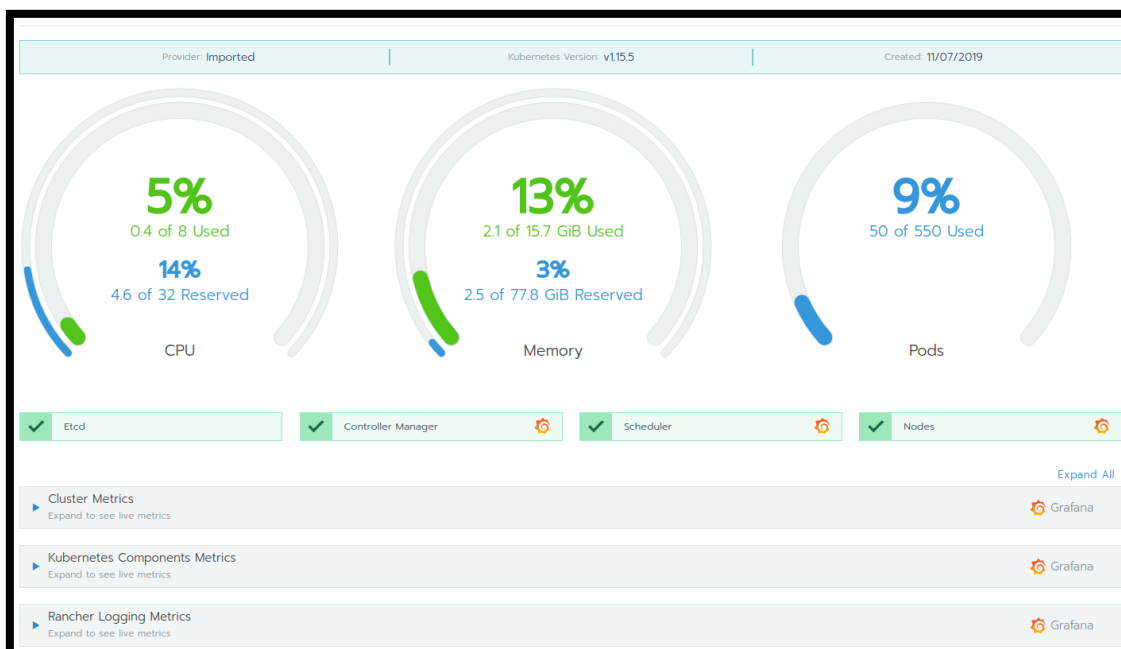
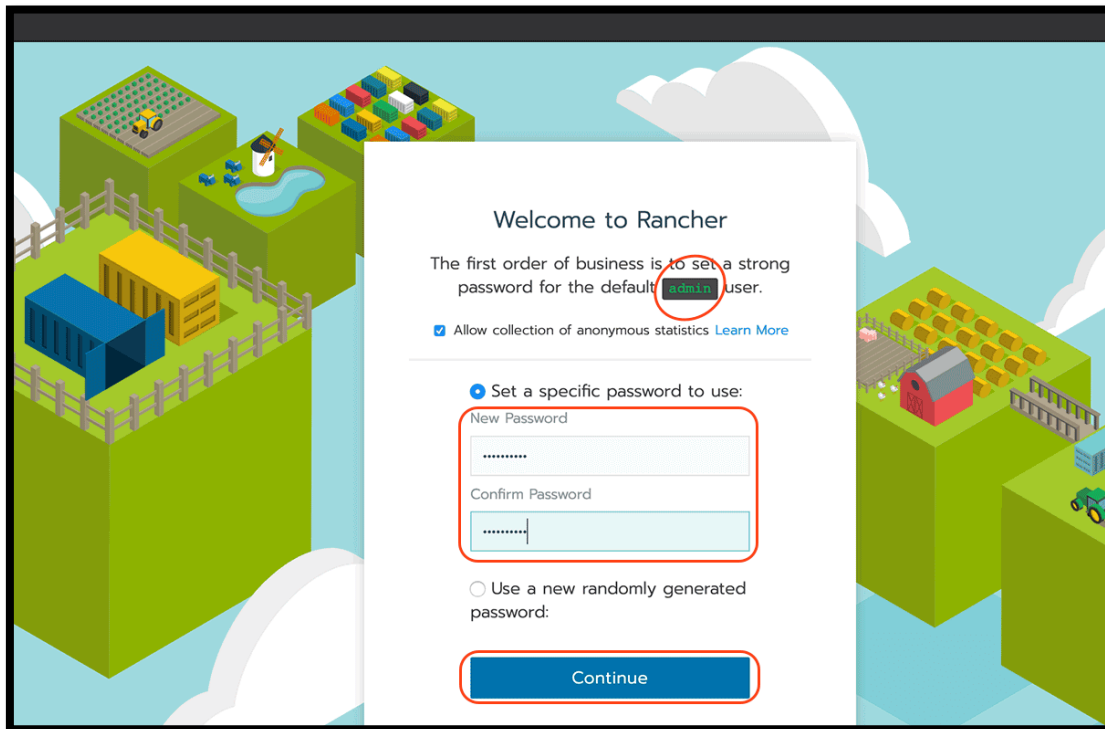


Then install rancher using [Let's Encrypt](#) to generate our certificate, change **rancher.my.org** to your domain name, and the Email **me@example.org** to your Email

```
helm repo add rancher-latest https://releases.rancher.com/server-charts/latest
helm repo update
helm install rancher-latest/rancher \
  --name rancher \ --namespace cattle-system \
  --set hostname=rancher.my.org \
  --set ingress.tls.source=letsEncrypt \
  --set letsEncrypt.email=me@example.org
# Wait for rancher to rollout
kubectl -n cattle-system rollout status deploy/rancher
Waiting for deployment "rancher" rollout to finish: 0 of 3 updated replicas are
available...
deployment "rancher" successfully rolled out
```

Accessing Rancher GUI

Once the Rancher container is up and running, access the URL and create your password for default account admin. Type-in your own defined password and click [Continue] to set the password.



~ End



About QCT

QCT (Quanta Cloud Technology) is a global data center solution provider extending the power of hyperscale datacenter design in standard and open SKUs to all datacenter customers.



Product lines include servers, storage, network switches, integrated rack systems and cloud solutions, all delivering hyperscale efficiency, scalability, reliability, manageability, serviceability and optimized performance for each workload.

QCT offers a full spectrum of datacenter products and services from engineering, integration and optimization to global supply chain support, all under one roof.

The parent of QCT is Quanta Computer Inc., a Fortune Global 500 technology engineering and manufacturing company.

<http://www.QCT.io>

United States QCT LLC., Silicon Valley office
1010 Rincon Circle, San Jose, CA 95131
TOLL-FREE: 1-855-QCT-MUST
TEL: +1-510-270-6111
FAX: +1-510-270-6161
Support: +1-510-270-6216

QCT LLC., Seattle office
13810 SE Eastgate Way, Suite 190, Building 1,
Bellevue, WA 98005
TEL: +1-425-633-1620
FAX: +1-425-633-1621

China 云达科技, 北京办公室 (Quanta Cloud Technology)
北京市朝阳区东三环中路 1 号 · 环球金融中心东楼 1508 室
Room 1508, East Tower 15F, World Financial Center
No.1, East 3rd Ring Zhong Rd., Chaoyang District, Beijing, China
TEL: +86-10-5920-7600
FAX: +86-10-5981-7958

云达科技, 杭州办公室 (Quanta Cloud Technology)
浙江省杭州市西湖区古墩路浙商财富中心 4 号楼 303 室
Room 303 · Building No.4 · ZheShang Wealth Center
No. 83 GuDun Road, Xihu District, Hangzhou, Zhejiang , China
TEL: +86-571-2819-8660

Japan Quanta Cloud Technology Japan 株式会社
日本国東京都港区芝大門二丁目五番八号
牧田ビル 3 階
Makita Building 3F, 2-5-8, Shibadaimon ,
Minato-ku, Tokyo 105-0012, Japan
TEL: +81-3-5777-0818
FAX: +81-3-5777-0819

Taiwan 雲達科技 (Quanta Cloud Technology)
桃園市龜山區文化二路 211 號 1 樓
1F, No. 211 Wenhua 2nd Rd., Guishan Dist.,
Taoyuan City 33377, Taiwan
TEL: +886-3-286-0707
FAX: +886-3-327-0001

Other regions Quanta Cloud Technology
No. 211 Wenhua 2nd Rd., Guishan Dist.,
Taoyuan City 33377, Taiwan
TEL: +886-3-327-2345
FAX: +886-3-397-4770

All specifications and figures are subject to change without prior notice. Actual products may look different from the photos.

QCT, the QCT logo, Rackgo, Quanta, and the Quanta logo are trademarks or registered trademarks of Quanta Computer Inc.

All trademarks and logos are the properties of their representative holders.

Copyright © 2014-2015 Quanta Computer Inc. All rights reserved.