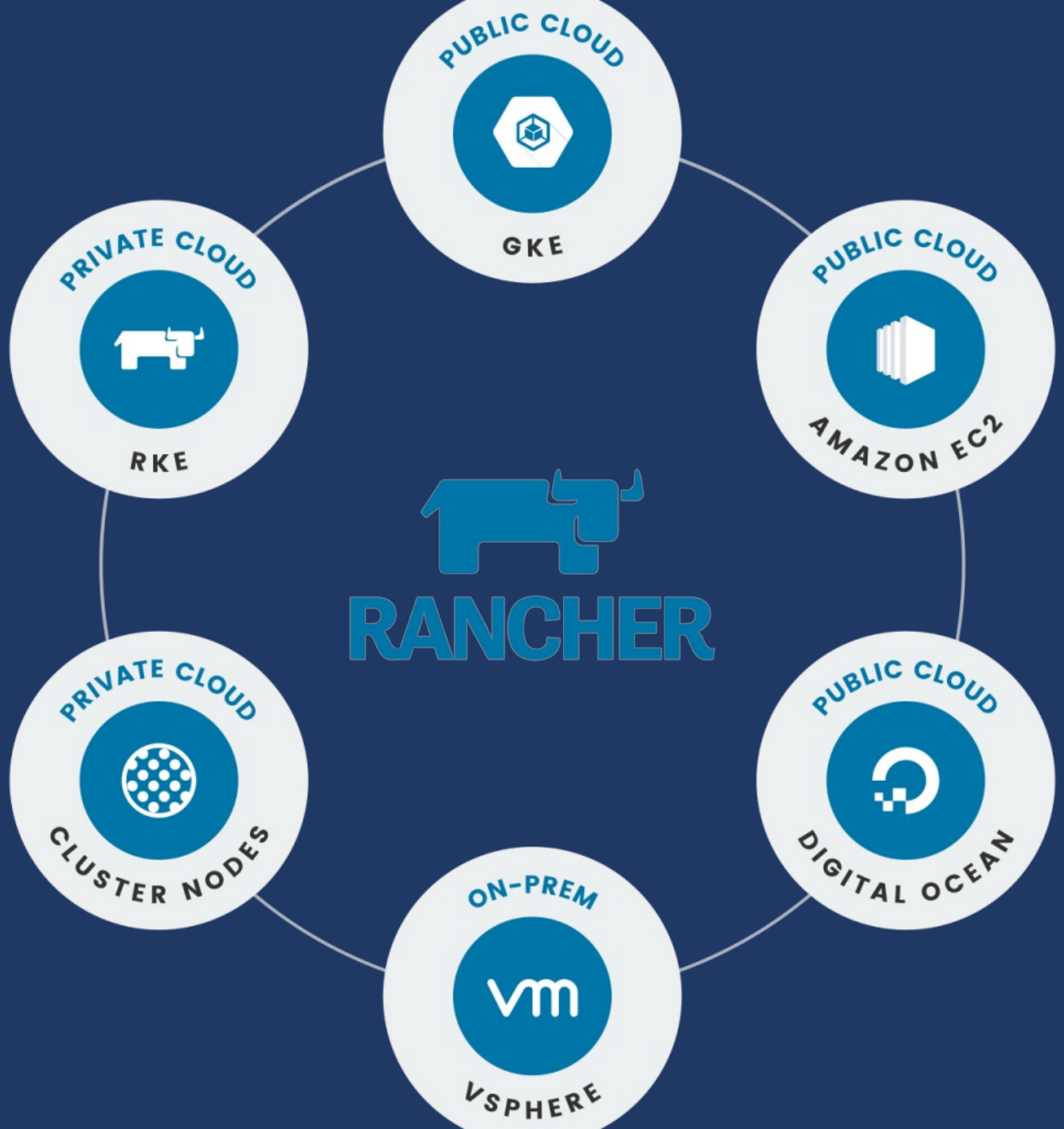

PUBLIC CLOUD
GKE
PRIVATE CLOUD
RKE
PUBLIC CLOUD
AMAZON EC2
RANCHER
PRIVATE CLOUD
CLUSTER NODES
PUBLIC CLOUD
DIGITAL OCEAN
ON-PREM
VSPHERE

Rancher 2.6.x

Kubernetes advantages:

- Consistently deliver a high level of reliability on any infrastructure
- Improve DevOps efficiency with standardized automation
- Ensure enforcement of security policies on any infrastructure

Kubernetes drawbacks (some of them):

- Without central visibility
- Without consistent security policies

Rancher:

- **Consistent Cluster Operations** – simplified Kubernetes upgrades, backups and deployments, from Data Center to the Edge.
- **Security Policy & User Management** – Consistent RBAC, Pod Security Policy (PSP), and user management.
- **Shared Tools & Services** – Out-of-the-box access to tools and services (e.g., Helm, Fleet)
- **External CI/CD and GitOps leveraging Fleet**

Build for production-grade kubernetes

Certified Kubernetes Distributions

Rancher Kubernetes Engine (RKE)

- Automate VM instance provisioning on many clouds using machine drivers.
- Install Kubernetes control plane and etcd database nodes.
- Provision worker nodes on Windows and Linux Arm64 and x86_64 nodes.
- Add or remove nodes in existing Kubernetes clusters.
- Upgrade Kubernetes clusters to new versions.
- Monitor the health of Kubernetes clusters.

K3s – Lightweight Kubernetes Distribution Built for IoT & the Edge

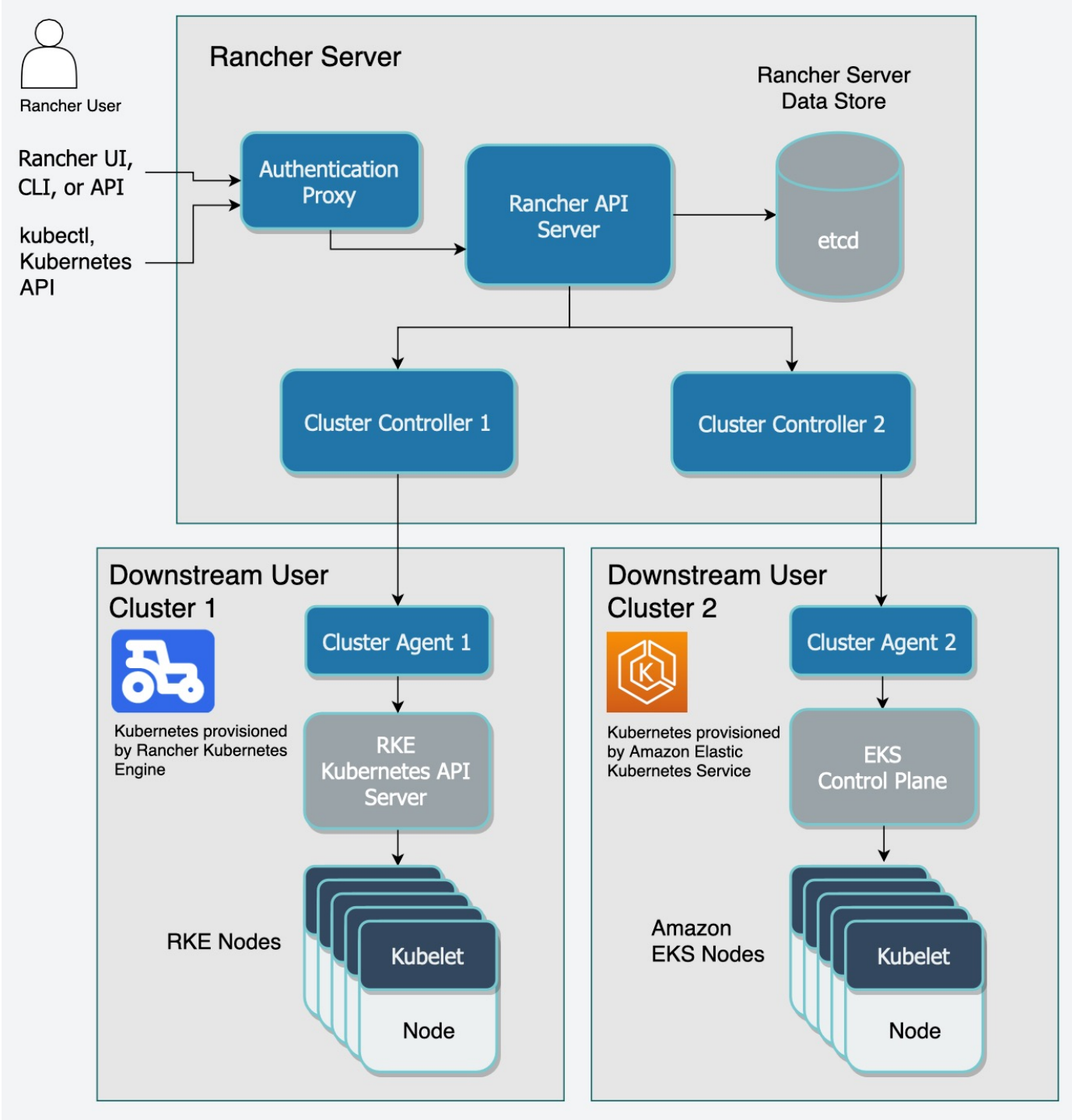
K3s is packaged as a single binary, which is about 50 megabytes in size. K3s bundles the Kubernetes components (kube-apiserver, kube-controller-manager, kube-scheduler, kubelet, kube-proxy) into combined processes that are presented as a simple server and agent model. K3s can run as a complete cluster on a single node or can be expanded into a multi-node cluster.

Besides the core Kubernetes components, it also run containerd, Flannel, CoreDNS, ingress controller and a simple host port-based service load balancer.

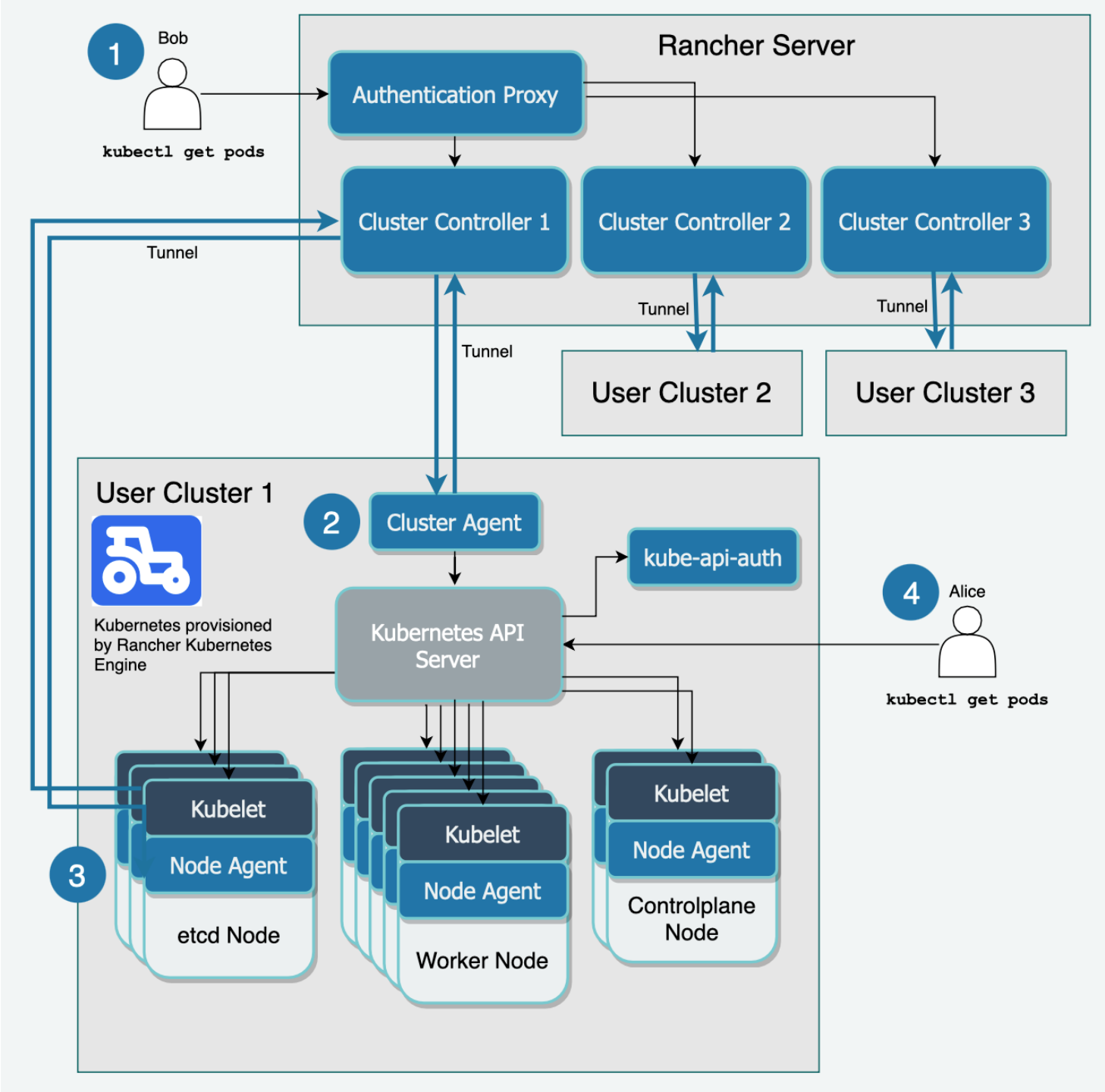
Shared Tools & Services

- Rancher UI does not attempt to hide the underlying Kubernetes concepts and introduces an application deployment framework different from Kubernetes. Rancher provides an easy-to-use UI for native Kubernetes resources like pods and deployments.
- The app catalog experience in Rancher is based on Helm charts. Rancher simplifies Helm chart deployment by exposing just the right set of variables and guiding the user through the process.
- Rancher works with any CI/CD systems that integrate with Kubernetes.
  - Jenkins, Drone, and GitLab
- Rancher works with any monitoring and logging systems that integrate with Kubernetes.
  - users can use the built-in Prometheus functionality or…
- Fleet is a Kubernetes cluster controller specifically designed to address the challenges of running thousands to millions of clusters worldwide. While it's designed for massive scale, the concepts still apply for even small deployments of less than 10 clusters. Fleet is lightweight enough to run on the smallest of deployments and even has merit in a single-node cluster managing only itself. The primary use case of Fleet is to ensure that deployments are consistent across clusters. You can deploy applications or easily enforce standards such as "every cluster must have X security tool installed."
- Fleet has two simple high-level concepts:
  - Cluster groups: A logical group of clusters that need to be targeted as a single entity.
  - Bundles: Collections of resources that are deployed to clusters.

Rancher Architecture (I)



Rancher Architecture (II)



Authentication Proxy

- The authentication proxy forwards all Kubernetes API calls to downstream clusters
- It integrates with authentication services like **local authentication**, **Active Directory**, and **GitHub**
- On every Kubernetes API call, the authentication proxy authenticates the caller and sets the proper Kubernetes impersonation headers before forwarding the call to Kubernetes masters
- By default, Rancher generates a **kubeconfig file** that contains credentials for proxying through the Rancher server to connect to the Kubernetes API server on a downstream user cluster
- The kubeconfig file (kube_config_cluster.yml) contains full access to the cluster

Cluster Controllers

There is one cluster controller and one cluster agent for each downstream cluster. Each cluster controller:

- Watches for resource changes in the downstream cluster
- Brings the current state of the downstream cluster to the desired state
- Configures access control policies to clusters and projects
- Provisions clusters by calling the required Docker machine drivers and Kubernetes engines, such as RKE and GKE

By default, to enable Rancher to communicate with a downstream cluster, the cluster controller connects to the cluster agent. If the cluster agent is not available, the cluster controller can connect to a **node agent** instead.

Cluster Agents

The cluster agent, also called cattle-cluster-agent, is a component that runs in a downstream user cluster. It performs the following tasks:

- Connects to the Kubernetes API of Rancher-launched Kubernetes clusters
- Manages workloads, pod creation and deployment within each cluster
- Applies the roles and bindings defined in each cluster's global policies
- Communicates between the cluster and Rancher server (through a tunnel to the cluster controller) about events, stats, node info, and health

Node Agents

If the cluster agent (also called **cattle-cluster-agent**) is not available, one of the node agents creates a tunnel to the cluster controller to communicate with Rancher.

- The cattle-node-agent is deployed using a DaemonSet resource to make sure it runs on every node in a Rancher-launched Kubernetes cluster.
- It is used to interact with the nodes when performing cluster operations.
- Examples of cluster operations include upgrading the Kubernetes version and creating or restoring etcd snapshots.

Authorized Cluster Endpoint (ACE)

An authorized cluster endpoint allows users to connect to the Kubernetes API server of a downstream cluster without having to route their requests through the Rancher authentication proxy.

There are two main reasons why a user might need the authorized cluster endpoint:

- To access a downstream user cluster while Rancher is down
- To reduce latency in situations where the Rancher server and downstream cluster are separated by a long distance

-> The **authorized cluster endpoint only works** on Rancher-launched Kubernetes clusters. In other words, it only works in clusters where Rancher used RKE to provision the cluster. The **ACE is NOT available** for clusters in a hosted Kubernetes provider, such as Amazon's EKS.

The ACE is available for registered RKE2 and K3s clusters as of Rancher v2.6.3.

Important files

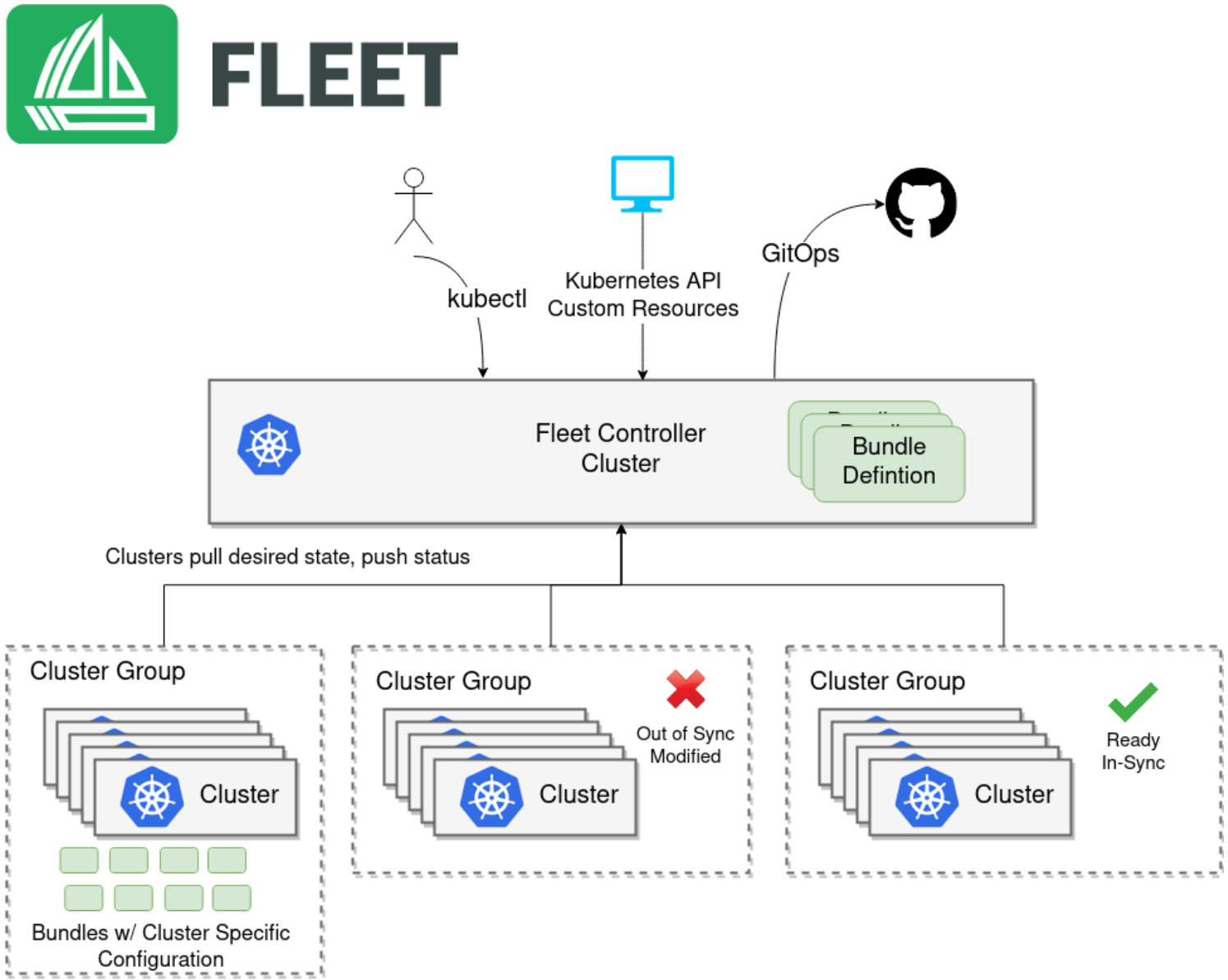
The files mentioned below are needed to maintain, troubleshoot and upgrade your cluster:

- rancher-cluster.yml: The RKE cluster configuration file.
- kube_config_cluster.yml: The Kubeconfig file for the cluster, this file contains credentials for full access to the cluster. You can use this file to authenticate with a Rancher-launched Kubernetes cluster if Rancher goes down.
- rancher-cluster.rkestate: The Kubernetes cluster state file. This file contains credentials for full access to the cluster. Note: This state file is only created when using RKE v0.2.0 or higher.

Fleet

- **Cluster engine**: Fleet is a container management and deployment engine designed to offer users more control on the local cluster and constant monitoring through GitOps. Fleet focuses not only on the ability to scale, but it also gives users a high degree of control and visibility to monitor exactly what is installed on the cluster.
- **GitOps at scale**: Fleet can manage up to a million clusters, but it's also lightweight enough that it works well for a single cluster. Fleet's capabilities are fully realized when it's used for large-scale projects. "Large scale" can mean a lot of clusters, a lot of deployments, or a lot of teams in a single organization.
- **Deployment management**: Fleet can manage deployments from git of raw Kubernetes YAML, Helm charts, Kustomize, or any combination of the three. Regardless of the source, all resources are dynamically turned into Helm charts, and Helm is used as the engine to deploy all resources in the cluster. As a result, users have a high degree of control, consistency, and auditability.

Fleet (II)

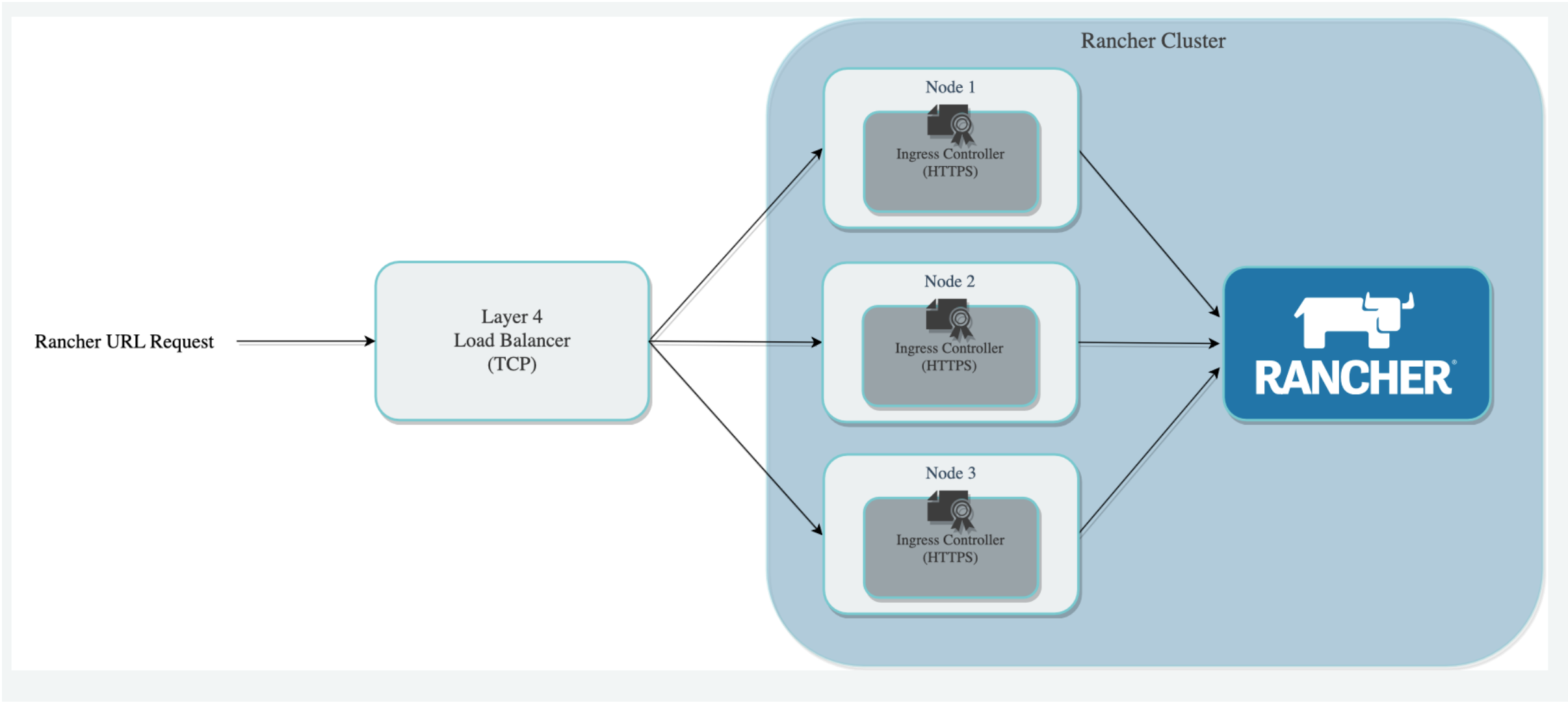

FLEET
kubectl
Kubernetes API
Custom Resources
GitOps
Fleet Controller
Cluster
Bundle
Defintion
Clusters pull desired state, push status
Cluster Group
Cluster
Bundles w/ Cluster Specific
Configuration
Cluster Group
Cluster
Out of Sync
Modified
Cluster Group
Cluster
Ready
In-Sync

Fleet (III)

Examples:

- https://fleet.rancher.io/examples/
- https://github.com/rancher/fleet-examples/tree/master/multi-cluster/helm
- https://github.com/rancher/fleet-examples/
- https://www.youtube.com/watch?v=PDOOoVp_icQ

Rancher HA


Rancher Cluster
Node 1
Ingress Controller
(HTTPS)
Node 2
Ingress Controller
(HTTPS)
Node 3
Ingress Controller
(HTTPS)
Rancher URL Request
Layer 4
Load Balancer
(TCP)
RANCHER

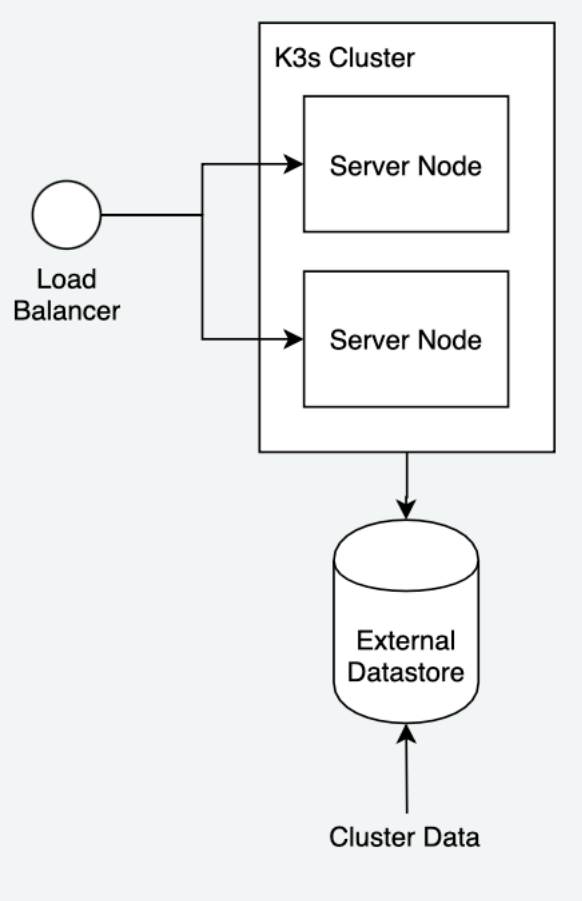
Rancher HA (II)

We recommend the following configurations for the load balancer and Ingress controllers:

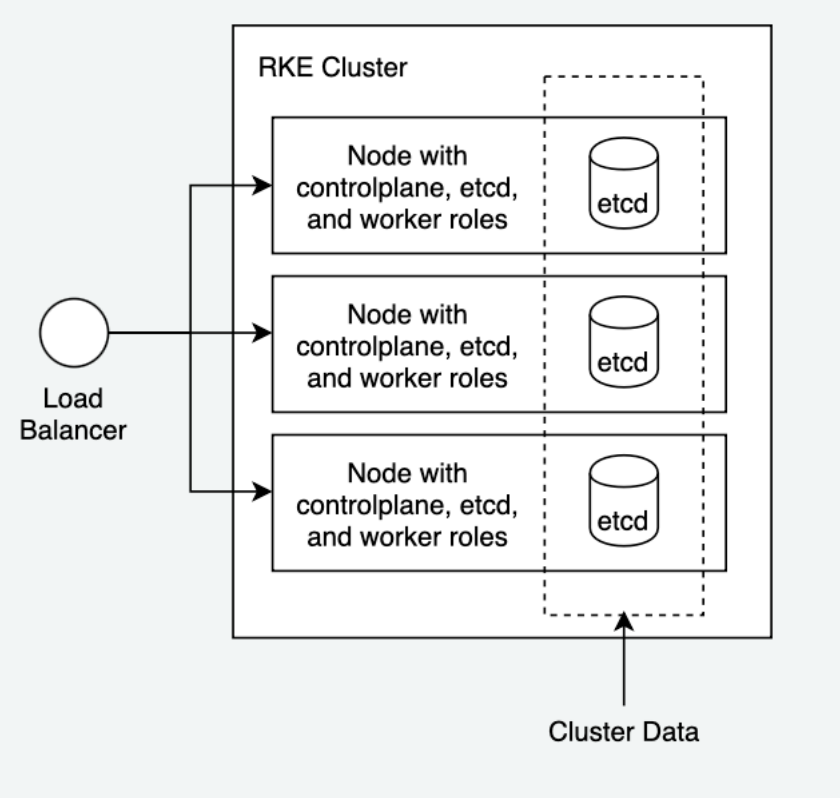
- The DNS for Rancher should resolve to a Layer 4 load balancer (TCP)
- The Load Balancer should forward port TCP/80 and TCP/443 to all 3 nodes in the Kubernetes cluster
- The Ingress controller will redirect HTTP to HTTPS and terminate SSL/TLS on port TCP/443
- The Ingress controller will forward traffic to port TCP/80 on the pod in the Rancher deployment

Rancher HA (III)

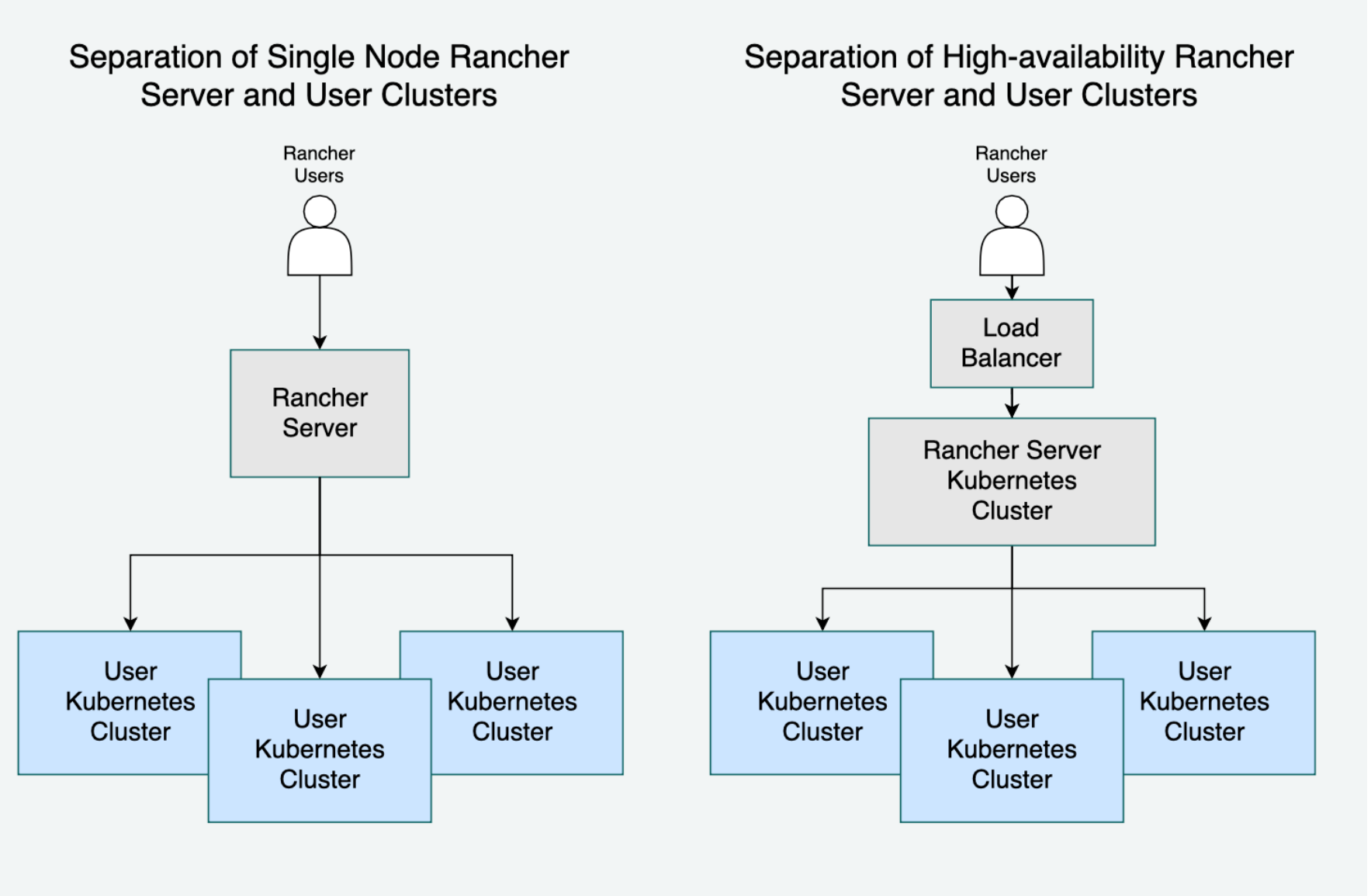
K3s



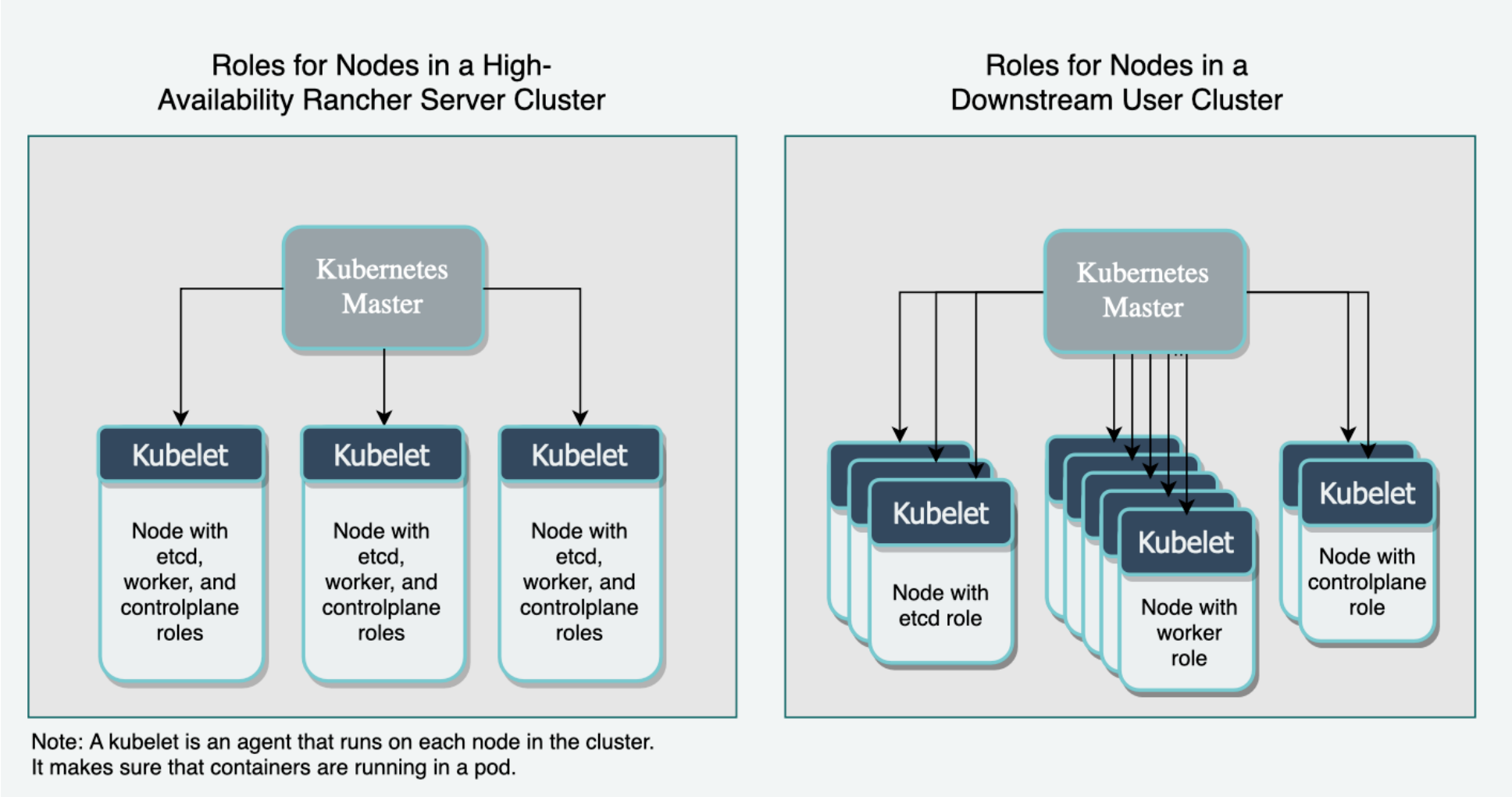
RKE



Separation of Rancher User Clusters



Roles for nodes within the cluster



Note: A kubelet is an agent that runs on each node in the cluster.
It makes sure that containers are running in a pod.

Roles for nodes within the cluster (II)

We recommend that downstream user clusters should have at least:

- **Three nodes with only the etcd role** to maintain a quorum if one node is lost, making the state of your cluster highly available
- **Two nodes with only the control plane role** to make the master component highly available
- **One or more nodes with only the worker role** to run the Kubernetes node components, as well as the workloads for your apps and services

It is safe to use all three roles on three nodes when setting up the Rancher server because:

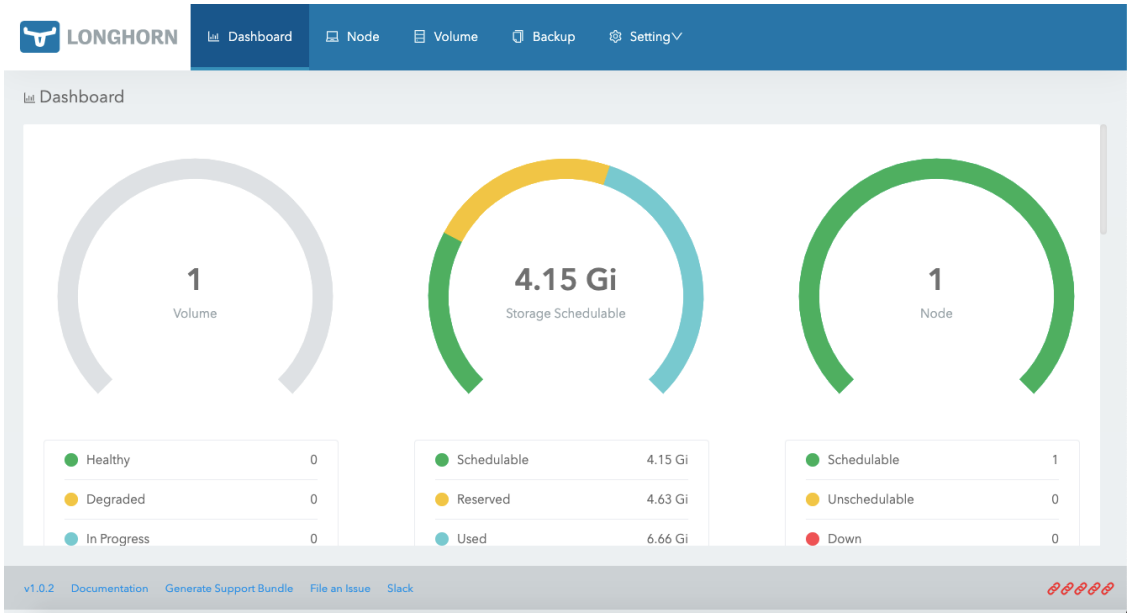
- It allows one etcd node failure
- It maintains multiple instances of the master components by having multiple control plane nodes
- No other workloads than Rancher itself should be created on this cluster

Checklist for production ready clusters
-> https://rancher.com/docs/rancher/v2.6/en/cluster-provisioning/production/
Best practices guide
-> https://rancher.com/docs/rancher/v2.6/en/best-practices/

Longhorn

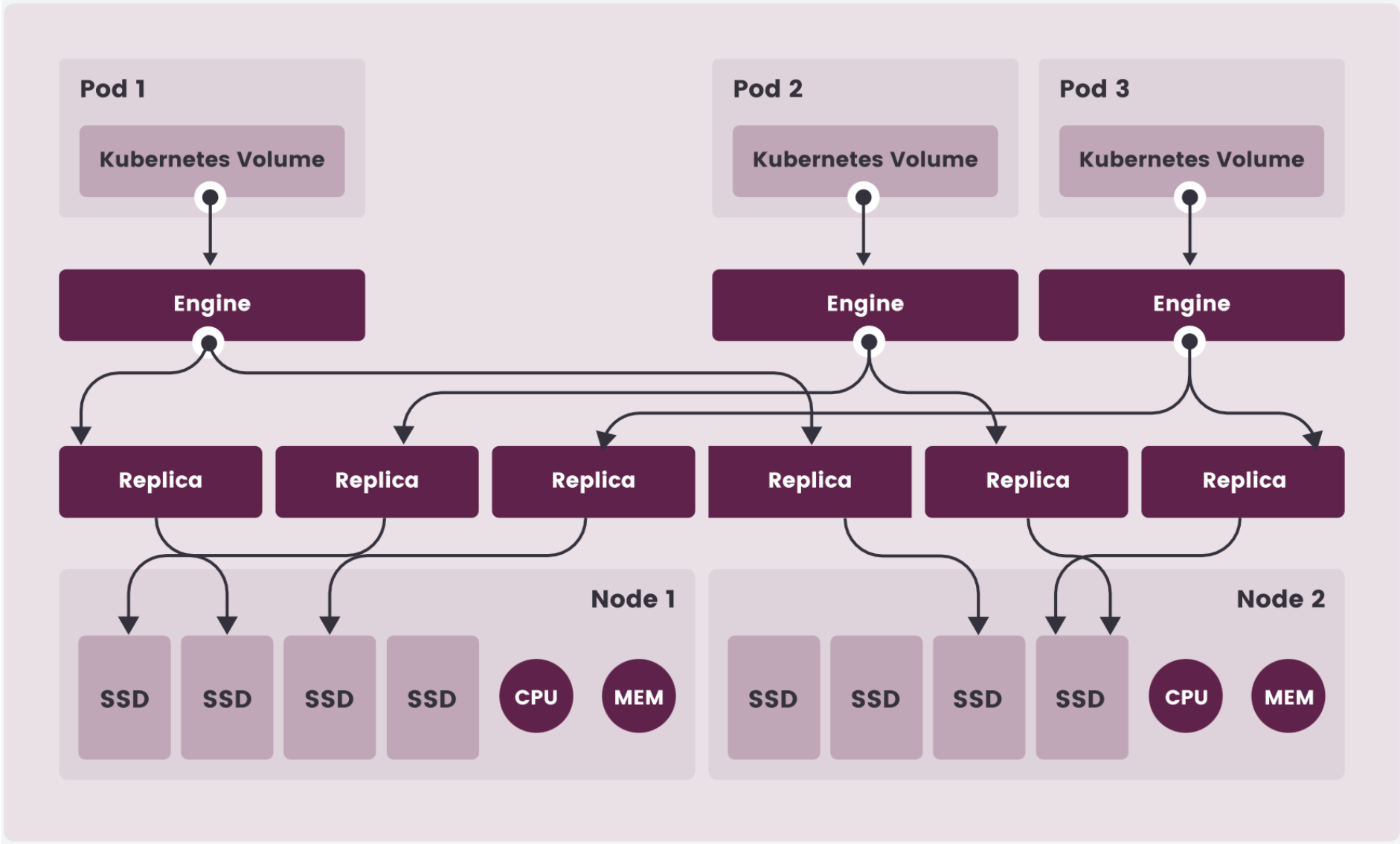
Longhorn is a Cloud native distributed block storage for Kubernetes.

- Volume provisioning through StorageClass
- Using Container Storage Interface (CSI)
- Cloud-native application
- Multiple replicas
- Snapshots
- Backup and Restore functionalities



https://longhorn.io/docs/1.0.2/concepts/

Longhorn Architecture



Longhorn Requirements

We recommend the following setup for deploying Longhorn in production.

Minimum Recommended Hardware

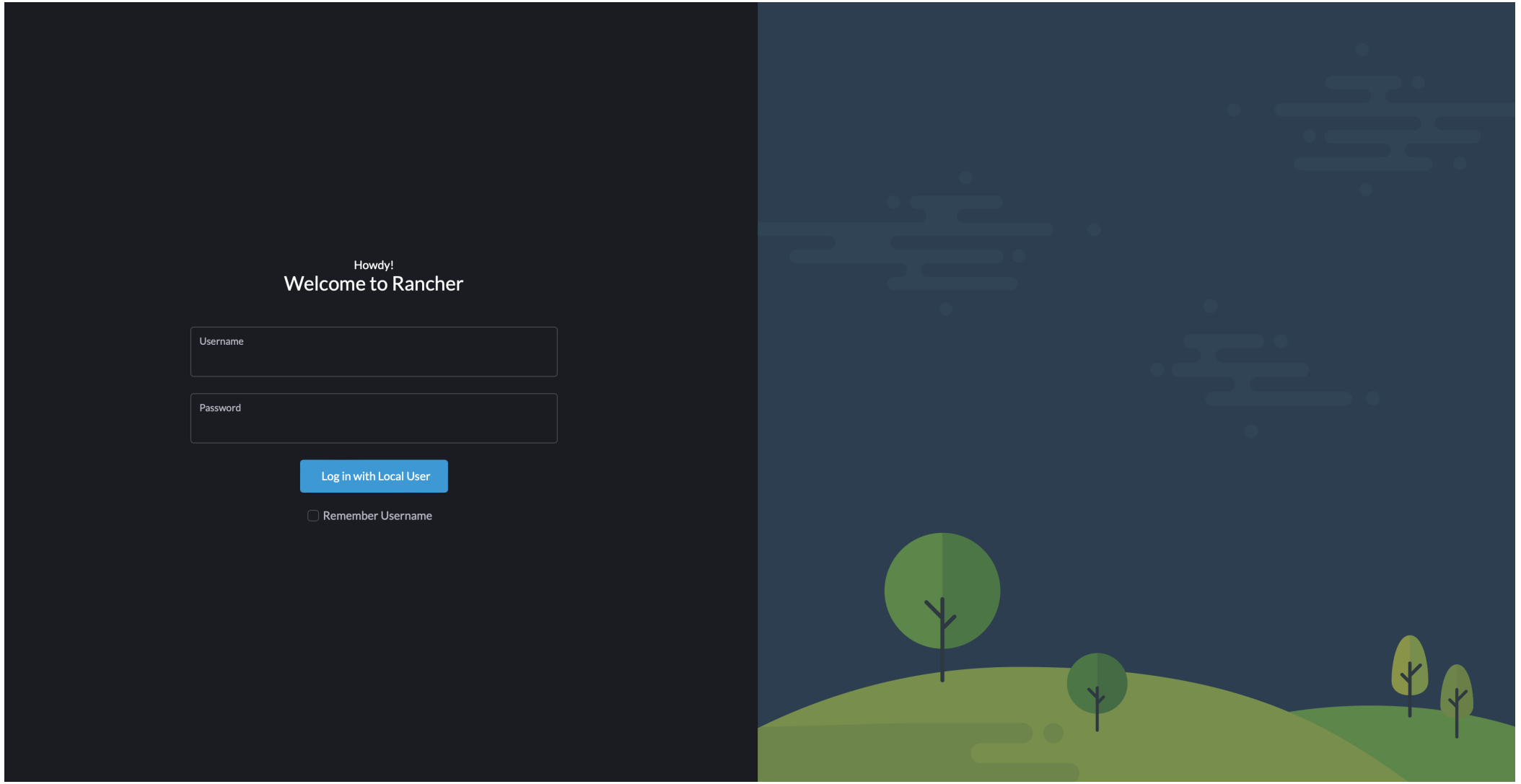
- 3 nodes
- 4 vCPUs per node
- 4 GiB per node
- SSD/NVMe or similiar performance block device on the node for storage
  - We don't recommend using spinning disks with Longhorn, due to low IOPS.

Software

It's recommended to run an OS from the following list for every node of your Kubernetes cluster:

1. Ubuntu 18.04
2. CentOS 7/8

Demo Time


Howdy!
Welcome to Rancher
Username
Password
Log in with Local User
Remember Username