RANCHER

September 2018

Rancher: Technical Architecture

Version 2.1

RANCHER

Contents

Background 3

Kubernetes is Everywhere 3

Rancher 2.0: Built on Kubernetes 3

- Rancher Kubernetes Engine (RKE) 4
- Unified Cluster Management 4
- Application Workload Management 5

High-level Architecture 5

Rancher Server Components 6

- Rancher API Server 6
- Management Controllers 7
- User Cluster Controllers 7
- Authentication Proxy 7

Rancher Agent Components 7

- Cluster Agents 7
- Node Agents 8

Upgrade 8

High Availability 8

Scalability 8

- Scalability of Kubernetes Clusters 8
- Scalability of Rancher Server 9

Background

We developed the Rancher container management platform to address the need to manage containers in production. Container technologies are developing quickly and, as a result, the Rancher architecture continues to evolve.

When Rancher 1.0 shipped in early 2016, it included an easy-to-use container orchestration framework called Cattle. It also supported a variety of industry-standard container orchestrators, including Swarm, Mesos, and Kubernetes. Early Rancher users loved the idea of adopting a management platform that gave them the choice of container orchestration frameworks.

In the last year, however, the growth of Kubernetes has far outpaced other orchestrators. Rancher users are increasingly demanding better user experience and more functionality on top of Kubernetes. We have, therefore, decided to re-architect Rancher 2.0 to focus solely on Kubernetes technology.

Kubernetes is Everywhere

When we started to build Kubernetes support into Rancher in 2015, the biggest challenge we faced was how to install and set up Kubernetes clusters. Off-the-shelf Kubernetes scripts and tools were difficult to use and unreliable. Rancher made it easy to set up a Kubernetes cluster with a click of a button. Better yet, Rancher enabled you to set up Kubernetes clusters on any infrastructure, including public cloud, vSphere clusters, and bare metal servers. As a result, Rancher quickly became one of the most popular ways to launch Kubernetes clusters.

In early 2016, numerous off-the-shelf and third-party installers for Kubernetes became available. The challenge was no longer how to install and configure Kubernetes, but how to operate and upgrade Kubernetes clusters on an on-going basis. Rancher made it easy to operate and upgrade Kubernetes clusters and its associated etcd database.

By the end of 2016, we started to notice that the value of Kubernetes operations software was rapidly diminishing. Two factors contributed to this trend. First, open-source tools, such as Kubernetes Operations (kops), have reached a level of maturity that made it easy for many organizations to operate Kubernetes on AWS. Second, Kubernetes-as-a-service started to gain popularity. A Google Cloud user, for example, no longer needed to set up and operate their own clusters. They could use Google Container Engine (GKE) instead.

The popularity of Kubernetes continues to grow in 2017 and 2018. The momentum is not slowing. Amazon Web Services (AWS) announced GA of Elastic Container Service for Kubernetes (EKS) in June 2018. Kubernetes-as-a-service is available from all major cloud providers. Unless they use VMware clusters and bare metal servers, DevOps teams will no longer need to operate Kubernetes clusters themselves. The only remaining challenge will be how to manage and utilize Kubernetes clusters, which are available everywhere.

Rancher 2.0: Built on Kubernetes

Rancher 2.0 is a complete container management platform built on Kubernetes. As illustrated in Figure 1, Rancher 2.0 contains three major components.

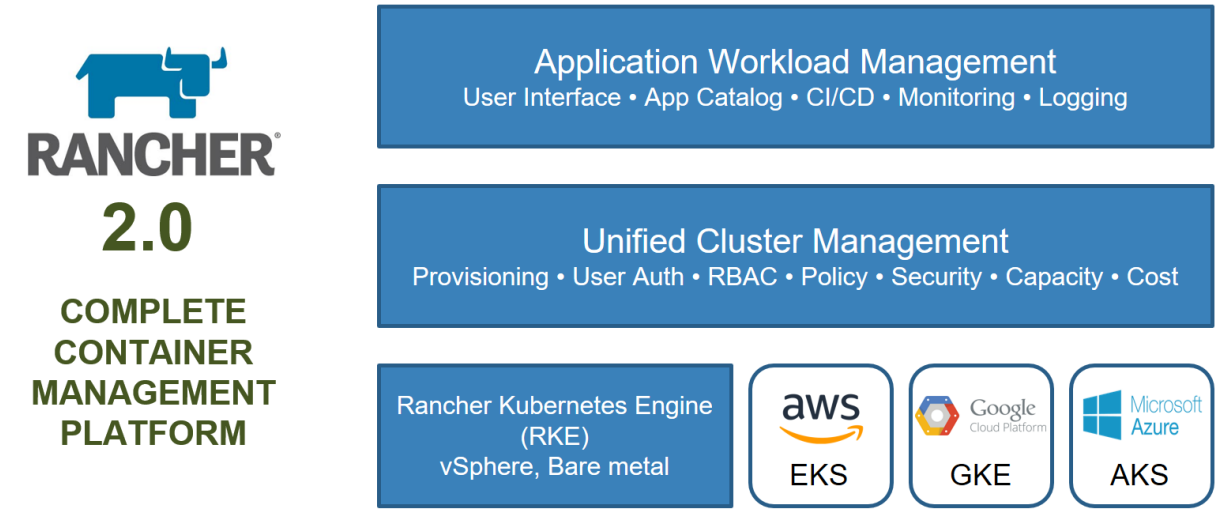

Figure 1 Overview of Rancher 2.0

Rancher Kubernetes Engine (RKE)

RKE is an extremely simple, lightning fast Kubernetes installer that works everywhere. RKE is particularly useful in standing up Kubernetes clusters on VMware clusters, bare metal servers, and VM instances on clouds that do not yet support Kubernetes service. In addition, many people use RKE in cloud providers that already support Kubernetes services so that they have a consistent Kubernetes implementation everywhere.

RKE manages the complete lifecycle of Kubernetes clusters from initial install to on-going maintenance. Rancher users can:

1. Automate VM instance provisioning on many clouds using Docker machine drivers.
2. Install Kubernetes masters, etcd nodes, and worker nodes.
3. Add or remove nodes in existing Kubernetes clusters.
4. Upgrade Kubernetes clusters to new versions.
5. Monitor the health of Kubernetes clusters.

Unified Cluster Management

With Rancher 2.0, you can choose to manage your own Kubernetes clusters, or use Kubernetes clusters managed by a cloud. You can use managed Kubernetes services such as GKE, EKS, and AKS, or provision and operate RKE Kubernetes clusters on any cloud, virtualized, or bare metal infrastructure.

Unified cluster management starts with centralized authentication. Rancher 2.0 provides an authentication proxy for all Kubernetes clusters under management. This is a crucial capability for enterprise IT to adopt cloud Kubernetes services like GKE. GKE normally requires its users to authenticate using their Google credentials. With Rancher 2.0, an enterprise developer can sign into a GKE cluster using the Active Directory credentials managed by corporate IT.

Rancher 2.0 further supports centralized access control policies built on top of centralized authentication. Leveraging the native RBAC capabilities of Kubernetes, Rancher 2.0 enables IT

administrators to configure and enforce access control and security policies across multiple Kubernetes clusters.

Rancher 2.0 introduces a new concept called Projects. A project is one or more Kubernetes namespaces and their associated policies such as RBAC rules, resource quota, etc. Users can create projects and assign other users or groups to the projects he owns. Users can be assigned different roles across multiple projects. For example, a developer can be given full create/read/update/delete privilege in a "dev" project but just read-only access in the staging and production projects. Users can only create, modify, or delete namespaces in the projects they are a member of. This greatly enhances the multi-tenant self-service functionality of Kubernetes.

In a future release, Rancher will provide visibility into capacity and cost of underlying resources consumed by Kubernetes clusters.

Application Workload Management

Rancher 2.0 offers an intuitive user interface for managing application workloads. Users can create Kubernetes workloads without having to learn all the Kubernetes concepts up front. As much as possible, the easy-to-use Cattle experience in Rancher 1.0 is adapted to work on the powerful Kubernetes orchestration engine.

Rancher 2.0 UI does not attempt to hide the underlying Kubernetes concepts and introduce an application deployment framework that is different from Kubernetes. Rancher provides an easy-to-use UI for native Kubernetes resources like pods and deployments.

The app catalog in Rancher 1.0 has been adapted to support Helm charts. Helm is a powerful templating mechanism for deploying applications on Kubernetes. But users still need to read through lengthy documentation to understand exactly what variables to set and the right values for these variables. This is an error-prone process. Rancher simplifies Helm chart deployment by exposing just the right set of variables and by guiding the user through the process. Rancher catalog guides the user by asking the right questions and presenting sensible defaults and multiple choice values.

Rancher 2.0 works with any CI/CD systems that integrate with Kubernetes. For example, Jenkins, Drone, and GitLab will continue to work with Rancher 2.0 as they did with Rancher 1.0. Rancher 2.0 additionally includes a managed CI/CD service built on Jenkins. The Rancher 2.0 CI/CD service seamless integrate with Rancher UI.

Rancher 2.0 works with any monitoring and logging systems that integrates with Kubernetes. For example, DataDog, Sysdig, and ELK will continue to work with Rancher 2.0 as they did with Rancher 1.0. Rancher 2.0 includes a managed Fluentd service for log aggregation. Rancher 2.0 includes a managed Prometheus alert manager. In the future, Rancher will also include a managed Prometheus service for out-of-the-box monitoring.

High-level Architecture

Rancher 2.0 software consists of two parts. The Rancher server components manages the entire Rancher deployment. Rancher also deploys agent components into Kubernetes clusters and nodes.

Figure 2 illustrates the high-level architecture of Rancher 2.0. The figure depicts a Rancher server installation that manages two Kubernetes clusters: one Kubernetes cluster created by RKE and another Kubernetes cluster created by GKE.

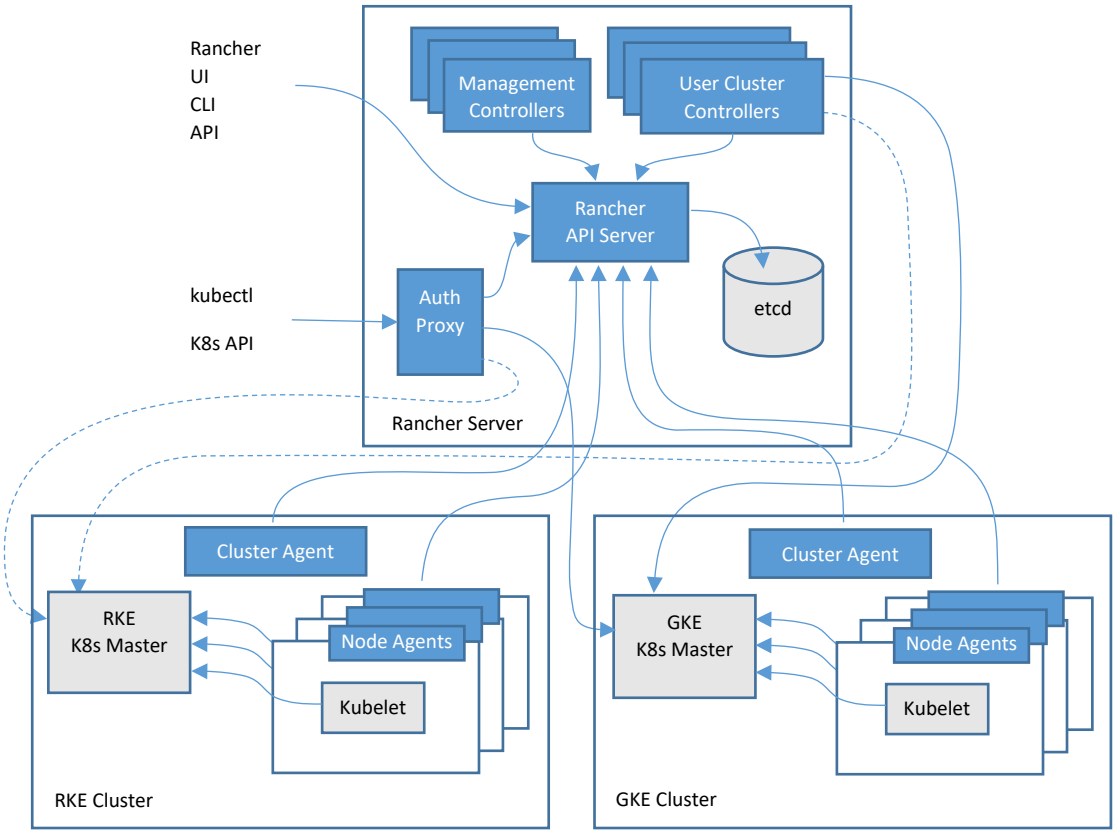

Figure 2 Rancher 2.0 High Level Architecture

Rancher Server Components

In this section we describe the functionalities of each Rancher server components.

Rancher API Server

Rancher API server is built on top of an embedded Kubernetes API server and etcd database. All Rancher specific resources that are being created using Rancher API, get translated to CRD (Custom Resource Definition) objects, with their lifecycle being managed by one or several Rancher controllers.

Rancher API Server is the foundational layer for all controllers in the Rancher server. It includes the following functionalities:

- User facing API schema generation with an ability to plug custom formatters and validators.
- Controller interfaces generation for CRDs and native Kubernetes objects types
- Object lifecycle management framework
- Conditions management framework
- Simplified generic controller implementation by encapsulating TaskQueue and SharedInformer logic into a single interface

Management Controllers

The management controllers perform the activities that happen at the Rancher server level, not specific to an individual cluster. The activities include:

1. Configuring access control policies to clusters and projects
2. Managing pod security policy templates
3. Provisioning clusters by invoking the necessary Docker machine drivers and invoking Kubernetes engines like RKE and GKE
4. Managing users – CRUD operations on users
5. Managing global-level catalog, fetch content of the upstream Helm repo, etc.
6. Managing cluster and project-level catalogs
7. Aggregating and displaying cluster stats and events
8. Managing of node drivers, node templates, and node pools
9. Managing cluster cleanup when cluster is removed from Rancher

User Cluster Controllers

User cluster controllers perform activities specific to a cluster. Activities include:

1. Managing workloads, which includes, for example, creating pods and deployments in each cluster
2. Applying roles and bindings that are defined in global policies into every cluster
3. Propagating information from cluster to rancher server: events, stats, node info, and health
4. Managing network policies
5. Managing alerts, log aggregation, and CI/CD pipelines
6. Managing resource quota.
7. Propagating secrets down from Rancher server to individual clusters

User cluster controllers connect to API servers in GKE clusters directly, but tunnel through the cluster agent to connect to API servers in RKE clusters.

Authentication Proxy

The authentication proxy proxies all Kubernetes API calls. 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. Rancher communicates with Kubernetes clusters using a service account.

The authentication proxy connects to API servers in GKE clusters directly, but tunnels through the cluster agent to connect to API servers in RKE clusters.

Rancher Agent Components

In this section, we describe software components deployed in Kubernetes clusters managed by Rancher.

Cluster Agents

Rancher deploys one cluster agent for each Kubernetes cluster under management. The cluster agent opens a WebSocket tunnel back to Rancher server so that the user cluster controllers and authentication proxy can communicate with user cluster Kubernetes API server. Note that only RKE

clusters and imported clusters utilize the cluster agent to tunnel Kubernetes API. Cloud Kubernetes services like GKE already exposes API endpoint on the public Internet and therefore does not require the cluster agent to function as a tunnel.

Cluster agents serve two additional functions:

1. They serve as a proxy for other services in the cluster, like Rancher's built-in alert, log aggregation, and CI/CD pipelines. In fact, any services running in user clusters can be exposed through the cluster agents. This capability is sometimes called "the magic proxy."
2. During registration, cluster agents get service account credentials from the Kubernetes cluster and send the service account credentials to the Rancher server.

Node Agents

Node agents are primarily used by RKE to deploy the components during initial install and follow-on upgrades. Node agents, however, are deployed on cloud Kubernetes clusters like GKE even though they are not needed for Kubernetes install and upgrade. Node agents serve several additional functions for all clusters:

1. Fallback for cluster agents: if the cluster agent is not available for any reason, Rancher server will use node agent to connect to the Kubernetes API server.
2. Proxy for `kubectl` shell. Rancher server connects through node agents to tunnel the `kubectl` shell in the UI. Node agent runs with more privileges than cluster agent, and that additional privilege is required to tunnel the `kubectl` shell.

Upgrade

Rancher 1.0 was built on Docker, and cannot be upgraded to a Kubernetes cluster without disrupting the workload. Rancher 1.0 users must setup a separate Rancher 2.0 cluster, migrate their workload, and decommission the Rancher 1.0 cluster.

Users can upgrade to new versions Rancher 2.0 by upgrading the Rancher server. Rancher 2.0 handles RKE cluster upgrades. Rancher 2.0 integrates with cloud providers like GKE to upgrade GKE clusters. Rancher 2.0 does not attempt to upgrade imported Kubernetes clusters.

High Availability

Rancher server can be deployed on an existing Kubernetes cluster. In that case, the Rancher server is highly available as long as there the underlying Kubernetes cluster has no single point of failure.

Users may use a dedicated RKE cluster to run the Rancher server. The standard Rancher 2.0 installation guide, for example, creates an RKE deployment with 3 nodes, each running one instance of the API server and the etcd database. Rancher server automatically imports the Kubernetes cluster it runs on. It is called "the local cluster."

Scalability

Scalability of Kubernetes Clusters

As of Kubernetes version 1.6, A Kubernetes cluster can scale to 5,000 nodes and 150,000 pods. User can expect Rancher 2.0 to manage clusters up to that scale as well.

Scalability of Rancher Server

There is no inherent limit on how many Kubernetes clusters each Rancher server can manage. We do not expect an issue for Rancher 2.0 to manage up to 1,000 clusters.

The real scalability limits of Rancher server are:

1. Total nodes across all clusters
2. Users and groups
3. Events collected from all clusters

Rancher server stores all the above entities in its own database. We will improve scalability along these dimensions over time to meet user needs.