Rancher Rodeo XII - Remote

Rancher Rodeo + RKE + Helm3

Step 1: Introduction

Welcome to the Rancher Rodeo RKE Edition.

In this scenario, we will be walking through installing Rancher in HA mode and deploying several workloads to a cluster provisioned by Rancher.

This scenario will be following the general HA installation instructions available here: High Availability (HA) Install

We will be using two virtual machines today, `cluster01` and `rancher01` which are located in the tabs in the panel to the right. `rancher01` will run a Kubernetes cluster and Rancher, and `cluster01` will run a Kubernetes cluster and the corresponding user workloads.

Note that there are two separate Kubernetes clusters at play here, the Rancher Kubernetes Cluster is dedicated to running Rancher, while the Workload Cluster is managed by Rancher and runs on a separate virtual machine.

Important Note: HobbyFarm will tear down your provisioned resources within 10 minutes if your laptop goes to sleep or you navigate off of the HobbyFarm page. Please ensure that you do not do this, for example, during lunch or you will need to restart your scenario.

There is Pause/Resume functionality built into HobbyFarm that will allow you to pause your scenario should you have to put your laptop to sleep temporarily. Please note that pausing the scenario will not extend the end of your resources beyond this Rodeo session.

Step 2: Generate an SSH Keypair for use with RKE

To start out, we will generate a new SSH Keypair and place this keypair on the node we will install Kubernetes for Rancher onto. As we will be using the `rancher01` node to run Rancher + Kubernetes, we will simply copy the public key into the `authorized_keys` file of this node.

The following command will generate the keypair and copy it into the file.

```
ssh-keygen -b 2048 -t rsa -f \
/home/ubuntu/.ssh/id_rsa -N ""
cat /home/ubuntu/.ssh/id_rsa.pub \
>> /home/ubuntu/.ssh/authorized_keys
```

Step 3: Download RKE

Rancher Kubernetes Engine (RKE) is an extremely simple, lightning fast Kubernetes installer that works everywhere.

In this step, we will download the `RKE CLI` to the `Rancher01` node.

```
sudo wget -O /usr/local/bin/rke \
https://github.com/rancher/rke/releases/download/v1.2.1/rke_linux-amd64
```

In order to execute RKE, we need to mark it as executable.

```
sudo chmod +x /usr/local/bin/rke
```

Next, let's validate that RKE is installed properly:

```
rke -v
```

You should have an output similar to:

```
rke version v1.2.1
```

If you receive the output as expected, you can continue on to the next step.

Step 4: Install kubectl

In order to interact with our Kubernetes cluster after we install it using `rke`, we need to install `kubectl`

The following command will add an apt repository and install `kubectl`.

```
sudo apt-get update
sudo apt-get install -y apt-transport-https
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg \
| sudo apt-key add -
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" \
| sudo tee -a /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubectl
```

After the `apt-get install -y kubectl` command finishes, we can test `kubectl` and make sure it is properly installed.

```
kubectl version --client
```

Step 5: Install Helm

Helm 3 is a very popular package manager for Kubernetes. It is used as the installation tool for Rancher when deploying Rancher onto a Kubernetes cluster. In order to download Helm, we need to download the Helm tar.gz,

move it into the appropriate directory, and mark it as executable. Finally, we will clean up the helm artifacts that are not necessary.

```
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
```

After a successful installation of Helm, we should check our installation to ensure that we are ready to install Rancher.

```
helm version --client
```

Step 6: Create a rancher-cluster.yml file

RKE CLI uses a YAML-formatted file to describe the configuration of our cluster. The following command will heredoc write the corresponding file onto your `rancher01` node, so that `RKE` will be able to install Kubernetes.

`RKE` uses SSH tunneling, which is why we generated the keypair in the first part of this scenario.

```
cat << EOF > rancher-cluster.yml
nodes:
  - address: ${vminfo:Rancher01:public_ip}
    internal_address: ${vminfo:Rancher01:private_ip}
    user: ubuntu
    role: [controlplane,etcd,worker]
addon_job_timeout: 120
EOF
```

Step 7: Run rke up

We are now ready to run `rke up` to install Kubernetes onto our `Rancher01` node.

The following command will run `rke up` which will install Kubernetes onto our node.

```
rke up --config rancher-cluster.yml
```

Step 8: Testing your cluster

RKE will have generated two important files:

- `kube_config_rancher-cluster.yml`
- `rancher-cluster.clusterstate`

in addition to your

- `rancher-cluster.yml`

All of these files are extremely important for future maintenance of our cluster. When running `rke` on your own machines to install Kubernetes/Rancher, you must make sure you have current copies of all 3 files otherwise you can run into errors when running `rke up`.

The `kube_config_rancher-cluster.yml` file will contain a `kube-admin` kubernetes context that can be used to interact with your Kubernetes cluster that you've installed Rancher on.

We can soft symlink the `kube_config_rancher-cluster.yml` file to our `/home/ubuntu/.kube/config` file so that `kubectl` can interact with our cluster:

```
mkdir -p /home/ubuntu/.kube
ln -s /home/ubuntu/kube_config_rancher-cluster.yml
/home/ubuntu/.kube/config
```

In order to test that we can properly interact with our cluster, we can execute two commands:

```
kubectl get nodes
```

```
kubectl get pods --all-namespaces
```

Step 9: Install cert-manager

cert-manager is a Kubernetes add-on to automate the management and issuance of TLS certificates from various issuing sources.

The following set of steps will install cert-manager which will be used to manage the TLS certificates for Rancher.

The following command will apply the cert-manager custom resource definitions as well as label the namespace that cert-manager runs in to disable validation.

```
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
```

Next, we'll add the helm repository for Jetstack

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

Update your helm repository cache

```
helm repo update
```

Now, we can install cert-manager version `0.15.0`

```
helm install \
  cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --version v0.15.0
```

Once the helm chart has installed, you can monitor the rollout status of both `cert-manager` and `cert-manager-webhook`

```
kubectl -n cert-manager rollout status deploy/cert-manager
```

You should eventually receive output similar to:

```
Waiting for deployment "cert-manager" rollout to finish: 0 of 1 updated replicas are available...

deployment "cert-manager" successfully rolled out
```

```
kubectl -n cert-manager rollout status deploy/cert-manager-webhook
```

You should eventually receive output similar to:

```
Waiting for deployment "cert-manager-webhook" rollout to finish: 0 of 1 updated replicas are available...

deployment "cert-manager-webhook" successfully rolled out
```

Step 10: Install Rancher

We will now install Rancher in HA mode onto our `Rancher01` Kubernetes cluster. The following command will add `rancher-latest` as a helm repository and update our local repository cache.

```
helm repo add rancher-stable https://releases.rancher.com/server-charts/stable
helm repo update
```

Next, we need to create the `cattle-system` namespace in our Kubernetes cluster to install Rancher into.

```
kubectl create namespace cattle-system
```

Finally, we can install Rancher using our `helm install` command.

```
helm install rancher rancher-stable/rancher \
  --namespace cattle-system \
  --set hostname=rancher.${vminfo:rancher01:public_ip}.on.hobbyfarm.io \
  --set replicas=1
```

Step 11: Verify Rancher is Ready to Access

Before we access Rancher, we need to make sure that `cert-manager` has signed a certificate using the `cattle-ca` in order to make sure our connection to Rancher does not get interrupted. The following bash script will check for the certificate we are looking for.

```
while true; do curl -kv
https://rancher.${vminfo:Rancher01:public_ip}.on.hobbyfarm.io 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";
```

Step 12: Accessing Rancher

Note: Rancher may not immediately be available at the link below, as it may be starting up still. Please continue to refresh until Rancher is available.

1. Access Rancher Server at https://rancher.${vminfo:Rancher01:public_ip}.on.hobbyfarm.io
2. Enter a password for the default `admin` user when prompted.
3. Select the default view of *"I want to create or manage multiple clusters"*
4. Make sure to agree to the Terms & Conditions
5. When prompted, the **Rancher Server URL** should be `rancher.${vminfo:Rancher01:public_ip}.on.hobbyfarm.io`, which is the hostname you used to access the server.
6. Once you log in, you'll see a message similar to "Waiting for server-url to be set". Click the ellipses on the right of the local cluster, click Edit, then click Save.

You will see the Rancher UI, with the `local` cluster in it. The `local` cluster is the cluster where Rancher itself runs, and should not be used for deploying your demo workloads.

Step 13: Creating A Kubernetes Lab Cluster within Rancher

In this step, we will be creating a Kubernetes Lab environment within Rancher. Normally, in a production case, you would create a Kubernetes Cluster with multiple nodes; however, with this lab environment, we will only be using one virtual machine for the cluster.

1. Hover over the top left dropdown, then click **Global**
    - *The current context is shown in the upper left, and should say 'Global'*
2. Click **Add Cluster**
    - Note the multiple types of Kubernetes cluster Rancher supports. We will be using **Existing nodes** for this lab, but there are a lot of possibilities with Rancher.
3. Click on the **Existing nodes** Cluster box
4. Enter a name in the **Cluster Name** box
5. Set the Kubernetes Version to a `v1.18` version
6. Click **Next** at the bottom.
7. Make sure the boxes **etcd**, **Control Plane**, and **Worker** are all ticked.
8. Click **Show advanced options** to the bottom right of the **Worker** checkbox
9. Enter the **Public Address** (`${vminfo:Cluster01:public_ip}`) and **Internal Address** (`${vminfo:Cluster01:private_ip}`)
    - **IMPORTANT:** It is VERY important that you use the correct External and Internal addresses from the **Cluster01** machine for this step, and run it on the correct machine. Failure to do this will cause the future steps to fail.
10. Click the clipboard to **Copy to Clipboard** the `docker run` command
11. Proceed to the next step of this scenario

Step 14: Start the Rancher Kubernetes Cluster Bootstrapping Process

IMPORTANT NOTE: Make sure you have selected the `Cluster01` tab in HobbyFarm in the window to the right. If you run this command on `Rancher01` you will cause problems for your scenario session.

1. Take the copied docker command and run it on `Cluster01`
2. Once the `docker run` command is complete, you should see a message similiar to `1 node has registered`
3. Within the Rancher UI click on `<YOUR_CLUSTER_NAME>` which is the name you entered during cluster creation.
4. You can watch the state of the cluster as your Kubernetes node `Cluster01` registers with Rancher here as well as the **Nodes** tab
5. Your cluster state on the Global page will change to **Active**
6. Once your cluster has gone to **Active** you can click on it and start exploring.

Step 15: Interacting with the Kubernetes Cluster

In this step, we will be showing basic interaction with our Kubernetes cluster.

1. Click into your newly `active` cluster.
2. Note the three dials, which illustrate cluster capacity.
3. Click the `Launch kubectl` button in the top right corner of the Cluster overview page, and enter `kubectl get pods --all-namespaces` and observe the fact that you can interact with your Kubernetes cluster using `kubectl`.

4. Also take note of the `Kubeconfig File` button which will generate a Kubeconfig file that can be used from your local desktop or within your deployment pipelines.
5. Click the Ellipses in the top right corner and note the various operational options available for your cluster. We will be exploring these in a later step.

Step 16: Enable Rancher Monitoring

To deploy the *Rancher Monitoring* feature, we will need to navigate to the Cluster Explorer.

1. On your newly-created cluster, click the "Explorer" button to open the Cluster Explorer.
2. Once the Cluster Explorer loads, use the dropdown in the upper-left section of the page, and navigate to "Apps & Marketplace."
3. Locate the "Monitoring" chart, and click on it
4. Select "Chart Options" on the left. Change Resource Limits > Requested CPU from `750m` to `250m`. This is required because our scenario virtual machine has limited CPU available.
5. Click "Install" at the bottom of the page, and wait for the `helm` install operation to complete.

Once Monitoring has been installed, you can click on that application under "Installed Apps" to view the various resources that were deployed.

Step 17: Working with Rancher Monitoring

Once Rancher Monitoring has been deployed, we can view the various components and interact with them.

1. In the dropdown in the upper-left corner of the Cluster Explorer, select "Monitoring"
2. On the Monitoring Dashboard page, identify the "Grafana" link. Clicking this will proxy you to the installed Grafana server

Once you have opened Grafana, feel free to explore the various dashboard and visualizations that have been setup by default.

These options can be customized (metrics and graphs), but doing so is out of the scope of this scenario.

Step 18: Create a Deployment And Service

In this step, we will be creating a Kubernetes Deployment and Kubernetes Service for an arbitrary workload. For the purposes of this lab, we will be using the docker image `rancher/hello-world:latest` but you can use your own docker image if you have one for testing.

When we deploy our container in a pod, we probably want to make sure it stays running in case of failure or other disruption. Pods by nature will not be replaced when they terminate, so for a web service or something we intend to be always running, we should use a Deployment.

The deployment is a factory for pods, so you'll notice a lot of similairities with the Pod's spec. When a deployment is created, it first creates a replica set, which in turn creates pod objects, and then continues to supervise those pods in case one or more fails.

Note: These steps will need to be executed within the Cluster Manager. To access, click on the Cluster Manager button at the top of the page.

---

1. Hover over the Dropdown next to the Rancher logo in the top left corner, hover over your cluster name, then select **Default** as the project.
2. Under the **Workloads** tab press **Deploy** in the top right corner and enter the following criteria:
    - **Name** - `helloworld`
    - **Docker Image** - `rancher/hello-world:latest`
    - Click **Add Port** and enter `80` for the container port
    - ** NOTE: ** Note the other capabilities you have for deploying your container. We won't be covering these in this Rodeo, but you have plenty of capabilities here.
3. Scroll down and click **Launch**
4. You should see one pod get deployed and a TCP endpoint under your workload name exposing the NodePort service that has been created.
5. Click the Arrow next to your workload that you just created, then note the `+` and `-` buttons under the replica count to the right. You can click these correspondingly and refresh your browser on the nodeport to see the changes in the pod name.

Step 19: Create a Kubernetes Ingress

In this step, we will be creating a Layer 7 ingress to access the workload we just deployed in the previous step. For this example, we will be using xip.io as a way to provide a DNS hostname for our workload. Rancher will automatically generate a corresponding workload IP.

1. Hover over the Dropdown next to the Rancher logo in the top left corner, hover over your cluster name, then select **Default** as the project.
2. In the **Default** project in the **Workloads** section, click on the **Load Balancing** tab
3. Click **Add Ingress** and enter the following criteria:
    - **Name** - `helloworld`
    - Leave '**Automatically generate a `.xip.io` hostname**' selected
    - Click the minus button on the right to remove the empty backend rule
    - Click the **+ Service** button
    - Pick the `helloworld-nodeport` service from the dropdown under **Target**
4. Click **Save** and wait for the `xip.io` hostname to register, you should see the rule become **Active** within a few minutes.
5. Click on the hostname and browse to the workload.

** Note: ** You may receive transient 404/502/503 errors while the workload stabilizes. This is due to the fact that we did not set a proper readiness probe on the workload, so Kubernetes is simply assuming the workload is healthy.

Step 20: Upgrading your Kubernetes Cluster

This step shows how easy it is to upgrade your Kubernetes clusters within Rancher.

1. Hover over the Dropdown next to the Rancher logo in the top left corner, then select your cluster.
2. Click the ellipses (...) next to the `Kubeconfig file` button
3. Click Edit
4. Scroll down and select the dropdown under the `Kubernetes Version`
5. Select a newer version of Kubernetes
6. Scroll down and hit `Save`

Observe that your Kubernetes cluster will now be upgraded.

Step 21: Congratulations

Congratulations, you have finished the Scenario. If you would like to tear down your lab environment, you can click the "Finish" button, otherwise, continue to work with your Kubernetes cluster while keeping HobbyFarm in the background.