RKE2 Automated Upgrades

How to install system-upgrade-controller for RKE2

One of the core ideas with RKE2 is that the cluster should be able to manage itself. This means that the cluster should be able to upgrade itself. RKE2 uses a tool called system-upgrade-controller to do this.

How does it work?

System Upgrade Controller is built around a controller running on the master node. This controller is responsible for spawning pods that will do the actual upgrade.

The `system-upgrade-controller` runs as a pod on the master node. That takes a CRD called `plans.upgrade.cattle.io` or `plan` for short. The `system-upgrade-controller` will watch for these plans, and if it finds one, it triggers an upgrade plan creation. The controller will build a list of nodes that have the label `rke2-upgrade=true`, and it will use that node list to create an upgrade plan for each node.

The plans are split into server-plan and agent-plan.

- server-plan plans are used to upgrade the master (etcd/control-plane) nodes.
- agent-plan plans are used to upgrade the agent (worker) nodes.

The system-upgrade-controller will start by upgrading the master nodes first. In this process, the controller will create a pod of one of the master nodes. This pod starts by cordoning and draining the node. Note: This pod ignores all taints. Once the drain is complete, the pod will trigger an upgrade of RKE2 on the node by running the same install script used to install RKE2. This will cause the binary to be downloaded and installed. Then the pod will trigger a restart of the rke2-server service. This will cause the RKE2 server to be restarted, and the node will disconnect from the cluster for a few minutes. During this time, the node will be unavailable for use by the cluster. Note: Even tho the node is unreachable, any pods still running on the node will still be running. But we drain the node before we upgrade it, so there shouldn’t be much running on the node. Once the node is back online, the pod will uncordon the node, and the node will be available for use by the cluster. The upgrade is complete, and the pod will go into the completed state. This will trigger the system-upgrade-controller to initiate a new pod for the next master node. And this process continues until all master nodes are upgraded.

While the master nodes are being upgraded, the system-upgrade-controller will create two pods for the worker nodes. Note: The default concurrency is 2. This means that the system-upgrade-controller will only make two pods simultaneously. This is to prevent the cluster from being overloaded. If you want to upgrade more than two nodes at a time, you can increase the concurrency to a higher number. But the agents will wait for all the master nodes to be upgraded before they start. At this point, the agent pods will begin the upgrade process like the master node, cycling through the worker nodes.

Install system-upgrade-controller

Install the controller with the following command:

```
kubectl apply -f https://github.com/rancher/system-upgrade-controller/releases/download/v0.13.2/system-upgrade-controller.yaml
```

Verify that the controller is running with the following command:

```
kubectl get pods -n system-upgrade
```

Add the plans

Example Master node plans:

```
apiVersion: upgrade.cattle.io/v1
kind: Plan
metadata:
  labels:
    rke2-upgrade: server
  name: server-plan
  namespace: system-upgrade
spec:
  concurrency: 1
  cordon: true
  drain:
    force: true
  nodeSelector:
    matchExpressions:
    - key: rke2-upgrade
      operator: Exists
    - key: rke2-upgrade
      operator: NotIn
      values:
      - disabled
      - "false"
    - key: node-role.kubernetes.io/control-plane
      operator: In
      values:
      - "true"
  serviceAccountName: system-upgrade
  tolerations:
  - key: CriticalAddonsOnly
    operator: Exists
  upgrade:
    image: rancher/rke2-upgrade
```

```
  version: v1.28.7+rke2r1
```

Note: You must update the version to the RKE2 version you want to upgrade. But it would be best if you double-checked that this is the correct version and is not lower than the current version, as the system-upgrade-controller will downgrade a node that can and will break the cluster.

Example Worker node plans:

```
apiVersion: upgrade.cattle.io/v1
kind: Plan
metadata:
  labels:
    rke2-upgrade: agent
  name: agent-plan
  namespace: system-upgrade
spec:
  concurrency: 2
  cordon: true
  drain:
    force: true
  nodeSelector:
    matchExpressions:
    - key: rke2-upgrade
      operator: Exists
    - key: rke2-upgrade
      operator: NotIn
      values:
      - disabled
      - "false"
    - key: node-role.kubernetes.io/control-plane
      operator: NotIn
      values:
      - "true"
  prepare:
    args:
    - prepare
    - server-plan
    image: rancher/rke2-upgrade
  serviceAccountName: system-upgrade
  tolerations:
  - key: CriticalAddonsOnly
    operator: Exists
  upgrade:
    image: rancher/rke2-upgrade
  version: v1.28.7+rke2r1
```

Note: For smaller clusters, you can use a lower concurrency. The default is 2. And for larger clusters, you can use a higher concurrency. But this number should be set to a number allowing you to upgrade as many nodes simultaneously without impacting the cluster. For example, if you have a cluster with ten nodes, you can use a concurrency of 2. If you have a cluster with 20 nodes, you can use a concurrency of 4.

Once the plans are created, you need to label the nodes with the following label rke2-upgrade=true. You can do this with the following command to label all nodes.

```
for node in `kubectl get node -o name | awk -F '/' '{print $2}'`; do kubectl label node ${node} rke2-upgrade=true --overwrite ; done
```

or step by step / node by node

```
kubectl label node <control-plane-node> rke2-upgrade=true --overwrite
```

```
kubectl label node <control-plane-node> rke2-upgrade=true --overwrite
```

Manual Upgrades

You can upgrade rke2 by using the installation script, by manually installing the binary of the desired version, or by using rpm upgrades in case of rpm installation.

Note: Upgrade the server nodes first, one at a time. Once all servers have been upgraded, you may then upgrade agent nodes.

Release Channels

Upgrades performed via the installation script or using our RKE2 Automated Upgrades feature can be tied to different release channels. The following channels are available:

Channal	Description
stable	(Default) Stable is recommended for production environments. These releases have been through a period of community hardening, and are compatible with the most recent release of Rancher.
latest	Latest is recommended for trying out the latest features. These releases have not yet been through a period of community hardening, and may not be compatible with Rancher.
v1.29 (example)	There is a release channel tied to each Kubernetes minor version, including versions that are end-of-life. These channels will select the latest patch available, not necessarily a stable release.

For an exhaustive and up-to-date list of channels, you can visit the rke2 channel service API. For more technical details on how channels work, you can see the channelserver project.

When attempting to upgrade to a new version of RKE2, the Kubernetes version skew policy applies. Ensure that your plan does not skip intermediate minor versions when upgrading. Nothing in the upgrade process will protect against unsupported changes to the Kubernetes version.

Upgrade rke2 Using the Installation Script