Kubernetes version and version skew support policy

This document describes the maximum version skew supported between various Kubernetes components. Specific cluster deployment tools may place additional restrictions on version skew.

Supported versions

Kubernetes versions are expressed as **x.y.z**, where **x** is the major version, **y** is the minor version, and **z** is the patch version, following Semantic Versioning terminology. For more information, see Kubernetes Release Versioning.

The Kubernetes project maintains release branches for the most recent three minor releases (1.20, 1.19, 1.18). Kubernetes 1.19 and newer receive approximately 1 year of patch support. Kubernetes 1.18 and older received approximately 9 months of patch support.

Applicable fixes, including security fixes, may be backported to those three release branches, depending on severity and feasibility. Patch releases are cut from those branches at a regular cadence, plus additional urgent releases, when required.

The Release Managers group owns this decision.

For more information, see the Kubernetes patch releases page.

Supported version skew

kube-apiserver

In highly-available (HA) clusters, the newest and oldest `kube-apiserver` instances must be within one minor version.

Example:

- newest `kube-apiserver` is at **1.20**
- other `kube-apiserver` instances are supported at **1.20** and **1.19**

kubelet

`kubelet` must not be newer than `kube-apiserver`, and may be up to two minor versions older.

Example:

- `kube-apiserver` is at **1.20**
- `kubelet` is supported at **1.20**, **1.19**, and **1.18**

> **Note:** If version skew exists between `kube-apiserver` instances in an HA cluster, this narrows the allowed `kubelet` versions.

Example:

- `kube-apiserver` instances are at **1.20** and **1.19**
- `kubelet` is supported at **1.19**, and **1.18** (**1.20** is not supported because that would be newer than the `kube-apiserver` instance at version **1.19**)

kube-controller-manager, kube-scheduler, and cloud-controller-manager

`kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` must not be newer than the `kube-apiserver` instances they communicate with. They are expected to match the `kube-apiserver` minor version, but may be up to one minor version older (to allow live upgrades).

Example:

- `kube-apiserver` is at **1.20**
- `kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` are supported at **1.20** and **1.19**

> **Note:** If version skew exists between `kube-apiserver` instances in an HA cluster, and these components can communicate with any `kube-apiserver` instance in the cluster (for example, via a load balancer), this narrows the allowed versions of these components.

Example:

- `kube-apiserver` instances are at **1.20** and **1.19**
- `kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` communicate with a load balancer that can route to any `kube-apiserver` instance
- `kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` are supported at **1.19** (**1.20** is not supported because that would be newer than the `kube-apiserver` instance at version **1.19**)

kubectl

`kubectl` is supported within one minor version (older or newer) of `kube-apiserver`.

Example:

- `kube-apiserver` is at **1.20**
- `kubectl` is supported at **1.21**, **1.20**, and **1.19**

> **Note:** If version skew exists between `kube-apiserver` instances in an HA cluster, this narrows the supported `kubectl` versions.

Example:

- `kube-apiserver` instances are at **1.20** and **1.19**
- `kubectl` is supported at **1.20** and **1.19** (other versions would be more than one minor version skewed from one of the `kube-apiserver` components)

Supported component upgrade order

The supported version skew between components has implications on the order in which components must be upgraded. This section describes the order in which components must be upgraded to transition an existing cluster from version **1.19** to version **1.20**.

kube-apiserver

Pre-requisites:

- In a single-instance cluster, the existing `kube-apiserver` instance is **1.19**
- In an HA cluster, all `kube-apiserver` instances are at **1.19** or **1.20** (this ensures maximum skew of 1 minor version between the oldest and newest `kube-apiserver` instance)
- The `kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` instances that communicate with this server are at version **1.19** (this ensures they are not newer than the existing API server version, and are within 1 minor version of the new API server version)
- `kubelet` instances on all nodes are at version **1.19** or **1.18** (this ensures they are not newer than the existing API server version, and are within 2 minor versions of the new API server

version)

- Registered admission webhooks are able to handle the data the new `kube-apiserver` instance will send them:
  - `ValidatingWebhookConfiguration` and `MutatingWebhookConfiguration` objects are updated to include any new versions of REST resources added in **1.20** (or use the `matchPolicy: Equivalent` option available in v1.15+)
  - The webhooks are able to handle any new versions of REST resources that will be sent to them, and any new fields added to existing versions in **1.20**

Upgrade `kube-apiserver` to **1.20**

> **Note:** Project policies for API deprecation and API change guidelines require `kube-apiserver` to not skip minor versions when upgrading, even in single-instance clusters.

kube-controller-manager, kube-scheduler, and cloud-controller-manager

Pre-requisites:

- The `kube-apiserver` instances these components communicate with are at **1.20** (in HA clusters in which these control plane components can communicate with any `kube-apiserver` instance in the cluster, all `kube-apiserver` instances must be upgraded before upgrading these components)

Upgrade `kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` to **1.20**

kubelet

Pre-requisites:

- The `kube-apiserver` instances the `kubelet` communicates with are at **1.20**

Optionally upgrade `kubelet` instances to **1.20** (or they can be left at **1.19** or **1.18**)

> **Note:** Before performing a minor version `kubelet` upgrade, drain pods from that node. In-place minor version `kubelet` upgrades are not supported.

> **Warning:**
> Running a cluster with `kubelet` instances that are persistently two minor versions behind `kube-apiserver` is not recommended:
>
> - they must be upgraded within one minor version of `kube-apiserver` before the control plane can be upgraded
> - it increases the likelihood of running `kubelet` versions older than the three maintained minor releases

kube-proxy

- `kube-proxy` must be the same minor version as `kubelet` on the node.
- `kube-proxy` must not be newer than `kube-apiserver`.
- `kube-proxy` must be at most two minor versions older than `kube-apiserver`.

Example:

If `kube-proxy` version is **1.18**:

- `kubelet` version must be at the same minor version as **1.18**.
- `kube-apiserver` version must be between **1.18** and **1.20**, inclusive.

Feedback

Was this page helpful?

Yes No

Last modified January 14, 2021 at 3:05 PM PST: Clarify that nodes must be drained before minor version kubelet upgrades (8781aceb6)