

Upgrading epic-app to a new version

Downloaded from Epic Games Confluence

Date: 2025-07-12 04:07:19

Original URL: <https://confluence-epicgames.atlassian.net/wiki/spaces/CDE/pages/81068732>

Document Level Classification

[300](#)


- [Introduction](#)
- [When do I need to upgrade epic-app?](#)
- [What happens if I don't upgrade epic-app?](#)
- [Kubernetes and epic-app compatibility](#)
 - [How to check Kubernetes cluster version?](#)
 - [How to check currently deployed epic-app version?](#)
 - [Check epic-app version using kubect!](#)
 - [Check epic-app version using the Deployment Status Tool](#)
- [Upgrade Scenarios](#)
 - [Upgrade to the latest version of epic-app from another version](#)
 - [Upgrade to the latest version of epic-app from another version due to a Kubernetes cluster upgrade that introduces an API Deprecation](#)

Introduction

Upgrading epic-app is typically as easy as updating the Chart.yaml's dependencies section of the app using the chart. However, there are cases

in which an upgrade could be destructive due to a breaking change. In



particular, versions marked with  in the [CHANGELOG](#), there is a chance of extra work involved to upgrade to that version. This document serves as guidance when upgrading epic-app and various scenarios that can be encountered.

When do I need to upgrade epic-app?

As a general rule, teams aren't required to keep upgrading to the latest version unless there's a specific feature or bugfix that they can benefit from or otherwise becomes mandatory due to special circumstances (for example, epic app 1.x -> 2.x transition due to an API deprecation). However, using the latest patch version is always encouraged as it reduces the friction to move to future upgrades and makes it easier to adopt new features such as KEDA, canary deployments, etc. For example, in the Chart.yaml file you would specify `version: ~3.1.0`. This will match version 3.1.0 and any patches to that release. Meaning any version ≥ 3.10 but < 3.20 . Reference [Tilde Range Comparison](#) table on github.

What happens if I don't upgrade epic-app?

As stated above, generally a team or service owner is not required to upgrade their epic-app version. If you don't upgrade app to the latest epic-app version, you will not receive the latest patches and/or benefit from added functionality in new releases. In some cases, if you do not upgrade epic-app you can also experience negative effects. For example, if you are on an old version of epic-app and your Kubernetes cluster is

scheduled to be upgraded, you will hold up the cluster upgrade process. If you go a long period of time without upgrading, epic-app may release more and more functionality that you may have to account for when upgrading to a new version. This will lead to extra work in order to account for changes that need to be made for your service to function. For example, the addition of mandatory values such as `requiredTags` that you will have to account for when upgrading to a new version of epic-app. These new features can compound.

Kubernetes and epic-app compatibility

Prior to upgrading epic-app to a new version it is recommended to check the current Kubernetes version running on your cluster and the currently deployed epic-app Helm chart version. Once you have obtained that information you can utilize the table below to determine if the Kubernetes cluster version and the epic-app version are compatible. The below table shows the Kubernetes Version followed by the Supported Epic App Version.

Kubernetes Version(s)	Supported Epic App Version(s)	Kubernetes API Deprecation	epic-app CHANGELOG
v1.26+	v2.5.1+	Kubernetes updates to HPA API	epic-app v2.5.1 CHANGELOG
v1.25	v2.47-v2.50	Kubernetes updates to	

		HPA and PDB API's	epic-app v2.5.0 CHANGELOG
v1.22	v2.0.0-v2.47	Kubernetes Updates to Ingress API	epic-app v2.0.0 CHANGELOG
v1.16	v1.0.1-v1.0.26		

In the following examples we are checking both the Kubernetes version running on the cluster and also the currently deployed epic-app version. The result is that we are running Kubernetes version 1.27 and epic-app version 3.4.1. If we use the compatibility table we can deduce that if we are running Kubernetes version 1.27, epic-app version 2.5.1 and beyond are compatible. However, if we had Kubernetes version 1.25, we would not be able to run that version due to incompatibility. We would need to be running a version of the app that is 2.47 through 2.50.

How to check Kubernetes cluster version?

1. Authenticate with AWS SSO and K8s SSO. Instructions [here](#).
2. Run the following command to check Kubernetes version:

1. `kubectl version`

2. The output should look similar to the below output

```
$ kubectl version
Client Version: v1.29.1
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: v1.27.10-eks-508b6b3 <-----
```

How to check currently deployed epic-app version?

Check epic-app version using kubectl

1. Authenticate with AWS SSO and K8s SSO. Instructions [here](#).
2. On the cluster where your service is deployed, describe your deployment and look for the chart version. This can be done with the following command

1. `kubectl describe deployment <deployment-name>`

2. The output should look similar to the below output. Notice we are using grep to find "chart."

```
$ kubectl describe deploy cdp100-lab-03 | grep chart  
helm.sh/chart=epic-app-3.4.1 <-----
```

Check epic-app version using the Deployment Status Tool

1. Navigate to the [Deployment Status Tool](#).
2. Using the search box, you can search by typing the name of the application or the account in which the application resides.
3. After typing the name or account the search results will be filtered. You can see the currently deployed epic-app version in the Chart Version column.

Upgrade Scenarios

Upgrade to the latest version of epic-app from another version

In this scenario you would be performing a upgrade from the currently deployed version of epic-app to a new version but there are no Kubernetes deprecations involved.

1. Check your current Kubernetes cluster version and epic-app version.
2. Reference the [compatibility table](#) to make sure the epic-app version you want to upgrade to is compatible with the Kubernetes cluster version.
3. Review the epic-app [CHANGELOG](#) for guidance on breaking changes and make adjustments to values.yaml file if necessary. For example, if epic-app includes a new feature with mandatory values such as `requiredTags` those would have to be added to your values.yaml prior to deploying or you would encounter an error when deploying.
4. Update the Chart.yaml file's `dependency.version` to the version you want to upgrade to.
 1. **Note:** using the `~` before the version number will ensure you receive the latest patch update whenever you redeploy. For example, in the Chart.yaml file you would specify `version: ~3.1.0`. This will match version 3.1.0 and any patches to that release. Meaning any version ≥ 3.10 but < 3.20 . If you redeploy and there is a new version of epic-app, the new version will be used in the new deployment (AKA Automatic Upgrade). For this to work correctly you also need to make sure there is not a Chart.lock file present or it will use the version specified in the Chart.lock file. If that file exists it will need to be deleted when using [Tilde Range Comparison](#).
5. Redeploy your service. This will be done from either GitHub Actions or Codefresh depending on where your team's pipeline is set up.

Upgrade to the latest version of epic-app from another version due to a Kubernetes cluster upgrade that introduces an API Deprecation

In this scenario you would be performing an upgrade from the currently deployed version of epic-app to a new version with a breaking change due to a Kubernetes deprecation.

1. Check your current Kubernetes cluster version and epic-app version.
2. Reference the [compatibility table](#) to make sure the epic-app version you want to upgrade to is compatible with the Kubernetes cluster version.
3. Reference the [Kubernetes Deprecated API Migration Guide](#) to see what API's have been changed.
 1. Depending on what is changed, this could cause an issue where a Kubernetes object has immutable fields and a redeploy of that object fails. In this case it may be necessary to delete a Kubernetes object, `kubectl delete <object> <object-name>` before redeploying in step 6.
4. Review the epic-app [CHANGELOG](#) for guidance on breaking changes and make adjustments to values.yaml file if necessary. For example, if epic-app includes a new feature with mandatory values such as `requiredTags` those would have to be added to your values.yaml prior to deploying or you would encounter an error when deploying. The CHANGELOG may also include guidance on objects that may need to be deleted prior to redeploying. An example can be seen in the [epic-app v2.5.0 CHANGELOG](#).
5. Update the Chart.yaml file's `dependency.version` to the version you want to upgrade to.
 1. **Note:** using the `~` before the version number will ensure you receive the latest patch update whenever you redeploy. For example, in the Chart.yaml file you would specify `version: ~3.1.0`. This will match version 3.1.0 and any patches to that release. Meaning any version ≥ 3.10 but < 3.20 . If you redeploy and there is a new version of epic-app, the new version will be used in the new deployment (AKA Automatic Upgrade). For this to work correctly you also need to make sure there is not a Chart.lock file present or it will use the version specified in the Chart.lock file. If that file exists it will need to be deleted when using [Tilde Range Comparison](#).

6. Redeploy your service. This will be done from either GitHub Actions or Codefresh depending on where your team's pipeline is set up.

Page Information:

Page ID: 81068732

Space: Cloud Developer Platform

Downloaded: 2025-07-12 04:07:19