Upgrading the Cluster and Local Binaries

This lesson gives a practical exercise on upgrading the cluster and local binaries.

We'll cover the following

Inspecting the current version

Let's take a quick look at the current version before we upgrade our cluster.

```
jx version
```

In my case, the output is as follows.

```
NAME               VERSION
jx                 2.0.151
jenkins x platform 2.0.108
Kubernetes cluster v1.12.7-gke.10
kubectl            v1.14.2
helm client        Client: v2.14.0+g05811b8
git                git version 2.20.1 (Apple Git-117)
Operating System   Mac OS X 10.14.4 build 18E226
```

You might be asked whether you want to upgrade `jx` to the new release. That is a safe operation since it will upgrade only `jx` CLI and leave the apps running in the cluster intact. If you were creating the cluster using the provided Gists, you already upgraded the CLI quite a few times, so that should not be anything new.

It might be worth mentioning that `jx` CLI can also be upgraded through the `jx upgrade cli` command. The end result is the same, except that `jx upgrade cli` does not output all the versions, but directly updates only the CLI.

What matters, for now, is the `jenkins x platform` version from the output. In my

What matters, for now, is the `jenkins x platform` version from the output. In my case, it is `2.0.108`. If we take a look at the jenkins-x-platform.yml file, we can see

that quite a few versions of the platform were created in the meantime. At the time of this writing (May 2019), the current version is `2.0.330`. I am 22 versions behind. While that might sound like a lot, it really isn't since Jenkins X has a very high frequency of releases.

What is the Jenkins X Platform?

The Jenkins X platform is a bundle of quite a few applications already running in our cluster. If you are running static Jenkins X, Jenkins is one of the components of the platform. **ChartMuseum** is there, just as **Nexus**, **Monocular**, **Docker Registry**, and quite a few others. At this point, you might think that the Jenkins X platform is everything related to Jenkins X, but that would not be true. There are quite a few other applications installed as add-ons, extensions, apps, CRDs, and so on. We'll go through the process of upgrading them all, but, for now, we'll limit ourselves to the platform.

Let's take a quick look with the help of the `jx upgrade platform` command.

```
jx upgrade platform --help
```

The output shows us all the arguments we can set. That one that you should always be using is `-v` or `--version`. Even though most of the time you'll want to upgrade to the latest release, you should still specify the version. That way you can be sure that you'll upgrade production to the same version you'll test before that. Otherwise, the Jenkins X community might make a new release of the platform after you created the test environment and before you start the process of upgrading production.

Nevertheless, we will not use the `--version` argument in the exercise that follows because there are likely many new versions since the time of this writing. So, even though we'll skip `--version`, I expect you to use it when applying the exercises from this chapter in the "real" cluster. The same is true for all other `jx upgrade` commands we'll run later. We can say the same for `jx install` and `jx create cluster` commands. Using a specific version gives you control and a better understanding of the problems when things go wrong.

Always use the `--version` argument when installing or upgrading Jenkins X components, even if the examples in this book are ignoring it.

Upgrading the platform

Let's see what we'll get when we upgrade the platform.

```
jx upgrade platform --batch-mode
```

If you are already running the latest platform, you'll see a message notifying you that the command will skip the upgrade process. Otherwise, you'll see a detailed log with a long list of resources that were updated. It was an uneventful experience, so we can move on and check the versions one more time.

```
jx version
```

This time my `jenkins x platform` version is `2.0.330` (yours will be different), thus confirming that the upgrade process was successful.

There's much more to upgrades than keeping the platform up-to-date. But, before we get into that, let's take a look at what we are currently running in the cluster.

Getting the add-ons to check versions

> We haven't explored add-ons yet. We'll do that in one of the next chapters. For now, please note that they provide, as their name suggests, additional functionalities.

```
jx get addons
```

The output will significantly depend on whether you are running static on serverless Jenkins X and whether you installed add-ons outside those coming through the "standard" installation. I can only assume that you did not install add-ons on your own given that we did not cover them just yet. If that's the case, you are likely going to see an empty list if you're using static Jenkins X, and a few instances of `jx-prow` and `tekton` if you prefer the serverless flavor.

In the case of serverless Jenkins X, the output, without the repeated add-ons, is as follows.

```
NAME    CHART             ENABLED STATUS   VERSION
jx-prow jenkins-x/prow            DEPLOYED 0.0.620
...
tekton  jenkins-x/tekton          DEPLOYED 0.0.38
...
```

We can see that I'm running `jx-prow` version `0.0.647` and `tekton` version `0.0.38`.

What versions would we get if we upgrade those add-ons? We can check that quickly by visiting the jenkins-x/jenkins-x-versions repository. If you do, open the *charts/jenkins-x* directory and select the file that represents one of the add-ons you're interested in. For example, opening prow.yml shows that, at the time of this writing, the current version is `0.0.647`.

Upgrading the add-ons

Let's upgrade the add-ons.

> You might not have any add-ons installed, or those that you do might already be at the latest version. If that's the case, feel free to skip the command that follows.

```
jx upgrade addons
```

You'll see a long output with the list of things that changed and those that stayed the same.

Let's see what we've got.

```
jx get addons
```

You should see the add-ons running the latest versions.

> Please note that we could have upgraded a single addon by adding the name to the command. For example, `jx upgrade addon tekton` would upgrade only Tekton.

The same pattern can be followed with `app`, `crd`, and `extensions` upgrades. We haven't explored them just yet. When we do, you'll already know that they can be upgraded as well. Nevertheless, there should be no need to go through those as well since all you have to do is execute `jx upgrade`.

The last upgradable type of components is `ingress`. But, unlike other `upgrade` types we explored, this one does much more than what you might have guessed. We will cover it in the next lesson.