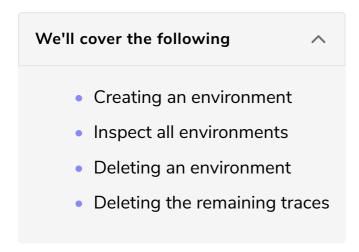
Controlling the Environments

This lesson covers the creation, retrieval and deletion of environments.



So far, we've seen that Jenkins X created three environments during its installation process.

- We acquired the development environment that runs the tools we need for continuous delivery as well as temporary Pods used during builds.
- We also acquired the **staging** environment where all the applications are promoted automatically whenever we push a change to the master branch.
- Finally, we have a production environment that is still a mystery.

Does all that mean that we are forced to use those three environments in precisely the way Jenkins X imagined?

The short answer is no. We can create as many environments as we need, we can update the existing one, and we can even delete them. So, let's start with the creation of a new environment.

Creating an environment

```
jx create env \
    --name pre-production \
    --label Pre-Production \
    --namespace jx-pre-production \
    --promotion Manual \
    --batch-mode
```

The arguments of the command should be self-explanatory. We just created a new Jenkins X environment called pre-production inside the Kubernetes Namespace jx-pre-production. We set its promotion policy to Manual, so new releases will not be installed on every push of the master branch of an application repository, but rather when we choose to promote it there.

If you take a closer look at the output, you'll see that the command also created a new GitHub repository, that it pushed the initial set of files, and that it created a webhook that will notify the system whenever we or the system pushes a change.

Inspect all environments

To be on the safe side, we'll list the environments and confirm that the newly created one is indeed available.



The output is as follows.

```
NAME
               LABEL
                               KIND
                                           PROMOTE NAMESPACE
                                                                      ORDER CLUSTER SOURCE REF
dev
               Development
                              Development Never
pre-production Pre-Production Permanent
                                           Manual jx-pre-production 100
                              Permanent
                                                                      100
staging
               Staging
                                           Auto
                                                   jx-staging
production
               Production
                               Permanent
                                           Manual jx-production
                                                                      200
```

Deleting an environment

Just as we can create an environment, we can also delete it.

```
jx delete env pre-production
```

As you can see from the output, that command did not remove the associated Namespace. But, it did output the kubectl delete command we can execute to finish the job. Please execute it.

Deleting the remaining traces

So, the jx delete env command will remove the references of the environment in Jenkins X, and it will delete the applications deployed in the associated Namespace. But, it does not remove the Namespace itself. That's not the only thing it didn't

remove. The repository is still in GitHub. By now, you should be used to the hub CLI. We'll use it to remove the last trace of the now non-existent environment.



That's it. We're done with the exploration of the environment. Or, to be more precise, we're finished with the environment with promotion policy set to Auto. Those set to Manual are coming soon.

Before we proceed, we'll go out of the environment-jx-rocks-staging directory.



Next, let's see if we are following all of the GitOps commandments.