Exploring and Adapting the Staging Environment

This lesson explores the staging environment and describes how tests are carried out in this environment.

We'll cover the following

Exploring the `staging` environment

Now that we have seen the environments and their general purposes, let's explore what's inside them. We already saw that they are linked to Git repositories, so we'll clone them and check what's inside.

> ⚠ If you are inside the *go-demo-6* or any other repository, please move to the parent directory by executing `cd ..` command.

Cloning the repository

Let's clone the *environment-jx-rocks-staging* repository that contains the definition of our staging environment.

> ⚠ Please replace `[...]` with your GitHub user before executing the commands that follow.

```
GH_USER=[...]

git clone \
    https://github.com/$GH_USER/environment-jx-rocks-staging.git

cd environment-jx-rocks-staging
cd environment-jx-rocks-staging
```

What do we have there?

```
ls -1
```

The output is as follows.

```
Jenkinsfile
LICENSE
Makefile
README.md
env
jenkins-x.yml
```

As you can see, there aren't many files in that repository. So, we should be able to explore them all relatively fast. The first one in line is `Makefile`.

```
cat Makefile
```

As you can probably guess by reading the code, the `Makefile` has targets used to build, install, and delete **Helm** charts.

Tests for the `staging` environment

Tests are missing. Jenkins X doesn't know if we want to run tests against applications in the `staging` environment and, if we do, what tests we want to run.

The `staging` environment is the place where all interconnected applications reside. That's the place where we deploy new releases in a production-like setting, and we'll see soon where that information about new releases is stored. For now, we'll focus on adding tests that will validate that a new release of any of the applications meets our system-wide quality requirements.

While you can run any type of test when deploying to the `staging` environment, I recommend to keep them light. We'll have all sorts of tests specific to applications inside their pipelines. For example, we'll run unit tests, functional tests, and whichever other types of application-specific tests we might have. We should assume that the application works on the functional level long before it is deployed to `staging`. With that in mind, all that's left to test in `staging` are cases that can be validated only when the whole system (or a logical and independent part of it) is up and running. Those can be integration, performance, or any other type of system-wide validations.

To run our tests, we need to know the addresses of the applications deployed to staging. Usually, that would be an easy thing since we'd use "real" domains. Our *go-demo-6* application could have a hard-coded domain *go-demo-6.staging.acme.com*. But, that's not the case since we're relying on dynamic domains assembled through a combination of the load balancer IP and nip.io. Fortunately, it is relatively easy to find out the address by querying the associated **Ingress**.

Once we have the address, we could extend `jenkins-x.yml` by adding a step that would execute the tests we'd like to run.

However, we won't do that just yet. We still need to explore Jenkins X pipelines in more detail.