Create a Quickstart Project

This lesson walks us through the process of creating a quickstart project.

We'll cover the following

Introduction to quickstart projects

Quickstart projects provide an easy way to start the development of a new application.

Traditionally, creating a new project is a tedious process that involves many different components. Obviously, we need to write the code of our new application and run many tests. On top of that, we need a mechanism to compile the code, to run the tests, to create a distribution, and so on and so forth. But it does not end there. Local development is only the beginning. We need to run performance, integration, and other types of tests that are too cumbersome to run locally. We need to deploy our application to different environments so that we can validate its readiness. We need to deal with branches and pull requests. A lot of things need to happen before a new release is deployed to production.

Quickstarts help us with those and some other tasks. They allow us to skip the tedious process and be up-and-running with a new project in a matter of minutes.

Later on, once we get a better understanding of what we need, we might need to modify the code and the configurations provided by quickstarts. That will be the subject of the follow-up writings. For now, our objective is to start a new project with the least possible effort, while still getting most of the things we need for local development as well as for the application's lifecycle that ends with the deployment to production.

That was enough of an introduction to Jenkins X quickstarts. We'll explore details through practical examples.

Creating a quickstart project

Like most other `jx` commands, we can create a quickstart project using the interactive or the batch mode. We'll take a look at the former first.

```
jx create quickstart
```

Git

First, we will be asked to confirm our Git user name as well as the organization.

Set repository name

Next, we need to type the name of the new repository that `jx` will create for us.

Select the quickstart project

After those questions, the output is as follows.

```
? select the quickstart you wish to create  [Use arrows to move, type to filter]
> android-quickstart
  angular-io-quickstart
  aspnet-app
  dlang-http
  golang-http
  jenkins-cwp-quickstart
  jenkins-quickstart
  node-http
  node-http-watch-pipeline-activity
  open-liberty
  python-http
  rails-shopping-cart
  react-quickstart
  rust-http
  scala-akka-http-quickstart
  spring-boot-http-gradle
  spring-boot-rest-prometheus
  spring-boot-watch-pipeline-activity
  vertx-rest-prometheus
```

We can see that there are quite a few types of projects we can create, all we have to do is select one of those.

While that is helpful at the beginning, I prefer running all the commands in the batch mode. If we proceed, we'd have to answer a few questions like the name of the project and what not. In the batch mode, instead of answering questions, we specify a few values as command arguments and, as a result, we end up with a documented way to reproduce our actions. It's easier and more reliable to have a README file with self-contained commands than to document the steps with specific instruction for each question.

> **Please cancel the current command by pressing *ctrl+c*.**

Creating a quickstart project using batch mode

We'll execute `jx create quickstart` again, but with a few additional arguments.

- We'll choose `golang-http` as the template.
- We'll name the project `jx-go`
- We'll use the `-b` (short for batch mode) argument to let `jx` know that there is no need to ask us any questions.

That does not mean that we will specify all the arguments we need, but rather those that differ from one project to another. When running in batch mode, `jx` will use the default values or those from previous executions.

> Don't worry if you do not work with Go, we'll use it only as an example. The principles we'll explore through practical exercises apply to any programming language.

Here we go.

```
jx create quickstart \
    --filter golang-http \
    --project-name jx-go
```

What did we get from this command?

The output is too big to be presented here, so I'll walk you through the steps `jx` performed while you're looking at your screen.

> You might be asked a couple of questions that you probably don't need guidance to answer.

We got the `jx-go` directory for our new Go application. Later on, it was converted into a Git repository, and `jx` copied the files from the pack (quickstart) dedicated to Go. Once it finished, it pushed the files to GitHub and created a project in Jenkins. As a result, the first build of the new Jenkins pipeline started running immediately. We'll explore Jenkins projects in more detail later.

If you're wondering where those quickstart projects come from, the answer is GitHub. The community created an organization called `jenkins-x-quickstarts` that contains the repositories hosting the quickstarts.

> ### A note to Windows users
>
> Git Bash might not be able to use the `open` command. If that's the case, replace `open` with `echo`. As a result, you'll get the full address that should be opened directly in your browser of choice.

```
open "https://github.com/jenkins-x-quickstarts"
```

Exploring the local buildpacks repository

Jenkins X also made a local copy of the repository in the *~/.jx/draft/packs/github.com/jenkins-x-buildpacks* directory. Let's see what's inside.

```
ls -1 ~/.jx/draft/packs/github.com/jenkins-x-buildpacks/jenkins-x-kubernetes/packs
```

The output is as follows.

```
C++
D
apps
```

```
appserver
charts
csharp

custom-jenkins
cwp
docker
docker-helm
dropwizard
environment
git
go
go-mongodb
gradle
helm
imports.yaml
javascript
jenkins
liberty
maven
maven-java11
ml-python-gpu-service
ml-python-gpu-training
ml-python-service
ml-python-training
nop
php
python
ruby
rust
scala
swift
typescript
```

We can see that it matches the output we got from the `jx create quickstart` command, even though the names of the directories are not the same.

Exploring the template files

Since we used the quickstart for **Go//88 language, we might as well take a look at its template files.

```
ls -1 ~/.jx/draft/packs/github.com/jenkins-x-buildpacks/jenkins-x-kubernetes/packs/go
```

The output is as follows.

```
Dockerfile
Makefile
charts
pipeline.yaml
preview
skaffold.yaml
watch.sh
```

I'll let you explore those files on your own, just remember that they are not used as-is, but rather serve as templates that are processed by the `jx create quickstart`

command in an attempt to create usable, yet customized, projects. Later on, we’ll also learn how to create custom quickstart packs.

Next, let’s explore the quickstart project files.