

Exploring Quickstart Project Files

This lesson explores the various files inside the jx-go repository.

We'll cover the following



- Exploring the Github repository
- Exploring the local project repository
 - The Makefile
 - The Dockerfile
 - Skaffold
 - The charts directory
 - The jx-go directory
 - The jenkins-x.ymlfile

Exploring the Github repository

Let's see what Jenkins X created and pushed to GitHub.



Please replace [...] with your GitHub username before executing the commands that follow.

```
GH_USER=[...]
```

```
open "https://github.com/$GH_USER/jx-go"
```



We can see that Jenkins X created quite a few files.

vfarcic / jx-goPrivate

Unwatch1Star0Fork0

<> Code

Issues0

Pull requests0

Actions

Projects0

Wiki

Security

Insights

Settings

No description, website, or topics provided.

Edit

Manage topics

2 commits

1 branch

0 packages

0 releases

0 contributors

Branch: masterNew pull request

Create new fileUpload filesFind fileClone or download

Viktor FarcicDraft createLatest commit 4aa256e 25 seconds ago

charts	Draft create	25 seconds ago
.dockerignore	Draft create	25 seconds ago
.gitattributes	Initial import	27 seconds ago
.gitignore	Initial import	27 seconds ago
.helmignore	Draft create	25 seconds ago
Dockerfile	Draft create	25 seconds ago
Makefile	Draft create	25 seconds ago
OWNERS	Draft create	25 seconds ago
OWNERS_ALIASES	Draft create	25 seconds ago
README.md	Initial import	27 seconds ago
curlloop.sh	Initial import	27 seconds ago
jenkins-x.yml	Draft create	25 seconds ago
main.go	Initial import	27 seconds ago
scaffold.yaml	Draft create	25 seconds ago
watch.sh	Initial import	27 seconds ago

GitHub repository with a newly created Jenkins X quickstart project

Exploring the local project repository

The repository was also created locally, so let's take a closer look at the generated files.

```
cd jx-go  
  
ls -l
```

The output is as follows:

```
Dockerfile  
Makefile  
OWNERS  
OWNERS_ALIASES  
README.md  
charts
```

```
charts  
curlloop.sh  
jenkins-x.yml  
  
main.go  
skaffold.yaml  
watch.sh
```

The **Makefile**

Let's go through each of those files and directories and explore what we got. The first in line is **Makefile**.

```
cat Makefile
```

I won't go into much detail about the **Makefile** since it is made specifically for **Go** applications, and that might not be your favorite language. Not all quickstart packs use **Makefile**. Instead, each tends to leverage methods appropriate for the given language and programming framework to accomplish the required tasks. In this case, the **Makefile** has targets to perform operations to **build**, **test**, **install**, and so on.

The **Dockerfile**

The next in line is **Dockerfile**.

```
cat Dockerfile
```

The output is as follows:

```
FROM scratch  
EXPOSE 8080  
ENTRYPOINT ["/jx-go"]  
COPY ./bin/ /
```

In the case of **Go**, there's not much needed. It uses a very lightweight base image (**scratch**), exposes a port, creates an entrypoint that will execute the binary (**jx-go**), and, finally, it copies that binary.

Skaffold

Unlike Dockerfile, **Skaffold** might be a tool you haven't used before.

Skaffold handles the workflow for building, pushing, and deploying applications to Kubernetes clusters, as well as for local development. We'll explore it in more detail later. For now, we'll take a brief look at the **skaffold.yaml** file.

detail later. For now, we'll take a brief look at the `scaffold.yaml` file.

```
cat scaffold.yaml
```

What matters, for now, is the `build` section that defines the `template` with the tag of the image we'll build in our pipelines. It consists of variables `DOCKER_REGISTRY` and `VERSION` whose values will be set by our pipeline at runtime.

The `charts` directory

Next, we have the `charts` directory that contains Helm definitions that will be used to deploy our application to Kubernetes. We won't go into much detail about Helm, but only the bits necessary to understand what Jenkins X does. If you have never used Helm, I recommend consulting the [official documentation](#) or read **The DevOps 2.4 Toolkit: Continuous Deployment to Kubernetes** book I published previously. For now, I'll summarize it by stating that Helm is a package manager for Kubernetes.

Let's take a look at what's inside the `charts` folder.

```
ls -l charts
```

The output is as follows.

```
jx-go  
preview
```

There are two subdirectories.

- `jx-go`: This contains the Helm definition of the application we'll deploy to different environments (e.g., staging, production).
- `preview`: This is mostly used with pull requests.

The reason for this separation lies in the ability to differentiate one from the other. We might need to customize `preview` with different variables or add temporary dependencies. We'll explore the `preview` charts in more depth later. Right now, we'll focus on the `jx-go` chart.

The `jx-go` directory

```
ls -l charts/jx-go
```

```
Chart.yaml
Makefile
README.md
charts
templates
values.yaml
```

If you used Helm, the structure should be familiar. If that's not the case, you might want to stop here and explore Helm in more detail. **The DevOps 2.4 Toolkit: Continuous Deployment to Kubernetes** might be a good read if you have time, otherwise, check the official documents.

The `jenkins-x.yml` file

The last file in the `jx-go` directory is `jenkins-x.yml`.

```
cat jenkins-x.yml
```

We will go into more detail about this alter, but for now, think of it as a pipeline definition that points to the pipeline, go, defined in a different repository and used with all applications written in GoLang.

Jenkins X did not only create a Git project, but also a pipeline in the cluster, as well as a GitHub webhook that will trigger it.

```
open "https://github.com/$GH_USER/jx-go/settings/hooks"
```

The screenshot shows the GitHub repository settings for 'vfarci / jx-go' (Private). The 'Settings' tab is selected, and the 'Webhooks' section is active. The left sidebar contains a list of settings: Options, Collaborators, Branches, Webhooks (highlighted), Notifications, Integrations & services, Deploy keys, Autolink references, Secrets, and Actions. The main content area is titled 'Webhooks' and includes an 'Add webhook' button. Below this, a text block explains that webhooks allow external services to be notified of certain events. A single webhook is listed with a green checkmark, the URL 'http://hook.jx.34.206.148.101.nip.io/hook (all events)', and 'Edit' and 'Delete' buttons.

vfarci / jx-go Private

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

Options

Collaborators

Branches

Webhooks

Notifications

Integrations & services

Deploy keys

Autolink references

Secrets

Actions

Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ <http://hook.jx.34.206.148.101.nip.io/hook> (all events) Edit Delete

A GitHub webhook created as part of Jenkins X quickstart process

From now on, every time we push a change to the repository, that webhook will trigger a build in Jenkins X.

Next, let's look at how we can retrieve almost any information related to Jenkins X in the command line through `jx`.