Getting Started with Custom Build Packs

This lesson teaches us about build packs and outlines the process we will follow to create custom build packs.

We'll cover the following

- The driving idea behind build packs
- How can you help?
- An overview of the process

The driving idea behind build packs

I stand by my claim that *"you do not need to understand Kubernetes to **use Jenkins X**."* To be more precise, those who do not want to know Kubernetes and its ecosystem in detail can benefit from Jenkins X's ability to simplify the processes around the software development lifecycle. That's the promise or, at least, one of the driving ideas behind the project. Nevertheless, for that goal to reach as wide of an audience as possible, we need a variety of build packs. The more we have, the more use cases can be covered with a single `jx import` or `jx quickstart` command.

> The problem is that there is an infinite number of types of applications and combinations we could have. Not all can be covered with community-based packs. No matter how much effort the community puts into creating build packs, they will always be a fraction of what we might need. **That's where you come in.**

How can you help?

The fact that Jenkins X build packs cannot fully cover all our use cases does not mean that we shouldn't work on reducing the gap. Some of our applications will have a perfect match with one of the build packs. Others will require only slight modifications. Still, no matter how many packs we create, there will always be a

case when the gap between what we need and what build packs offer is more significant.

Given the diversity in languages and frameworks we use to develop our applications, it is hard to avoid the need for understanding how to create new build packs. Those who want to expand the available build packs need to know at least the basics behind Helm and a few other tools, but that doesn't mean everyone needs to possess that knowledge. We don't want different teams to reinvent the wheel and we do not wish for every developer to spend endless hours trying to figure out all the details behind the ever-growing Kubernetes ecosystem. It would be a waste of time for everyone to go through steps of importing their applications into Jenkins X, only to discover that they need to perform a set of changes to adapt the result to their own needs.

An overview of the process

Our next mission is to streamline the process of importing projects for the teams in charge of applications that share common design choices yet don't have a build pack that matches all their needs.

We'll explore how to create a custom build pack that could be highly beneficial for a company. We'll imagine that there are multiple applications written in Go that depend on MongoDB and that there is a high probability that new ones based on the same stack will be created in the future. We'll explore how to create such a build pack with the least possible effort.

We'll need to decide on what should be included in the build pack, and what should be left to developers to add. Finally, we'll need to brainstorm whether the result of our work might be useful to others outside of our organization, or whether what we did is helpful only to our teams. If we conclude that the fruits of our work are useful to the community, we should contribute back by creating a pull request.

To do all that, we'll continue using the *go-demo-6* application since we are already familiar with it, and since we already went through the exercise of discovering which changes are required to the `go` build pack.

Let's get started and create a Kubernetes cluster with Jenkins X (if you already deleted the previous one).