Efficient Software Development with Jenkins X

This lesson lists the commonly faced problems in software development that can waste a lot of time and resources. It also states that many of these issues can be solved using Jenkins X.

We'll cover the following

What are the common problems in software development?

Software development is hard. It takes years to become a proficient developer, and the tech and the processes change so often. What was effective yesterday, is not necessarily effective today.

Multiple languages and frameworks

The number of languages we code in is increasing. While in the past, most developers would work in the same language throughout their whole career, today it is not uncommon for a developer to work on multiple projects written in different languages. We might work on a new project and code in Go while maintaining some other project written in Java. To be efficient, we need to install compilers, helper libraries, and quite a few other things.

Managing dependencies

No matter whether we write all the code in a single language or not, our applications will have different dependencies. One might need MySQL, while the other might use MongoDB as the data storage. We might also depend on applications developed by other teams working in parallel with us. No matter how good we become at writing mocks and stubs that replace those dependencies,

eventually, we'll need them running and accessible from our laptops. Historically, we've solved those problems by having a shared development environment, but that proved to be inefficient. Sharing development environments results is too much overhead. We'd need to coordinate changes, and those that we make would often break something and cause everyone to suffer. Instead, we need each developer to have the option to have its own environment where required dependencies for an application are running.

For the dependencies to be useful, we should run them in (almost) the same way we're running them in production, which means we should deploy them to Kubernetes as well. For that, we can choose **minikube** or Docker Desktop if we prefer a local cluster, or get a segment (Namespace) of a remote cluster.

Managing multiple tools

Unfortunately, compilers and dependencies are not everything we need to develop efficiently; we also need tools. Today that means that we need Docker or **kaniko** to build container images. We need `helm` and `kubectl` to deploy applications to Kubernetes. We need `skaffold` that combines the process of building images with deployment. There are quite a few other tools specific to a language and a framework that would need to be installed and configured as well.

Even if we do set up all those things, we are still missing more. We need to be able to push and pull artifacts from the container registry, **ChartMuseum**, **Nexus**, or any other registry that might be in use in our organization.

How can Jenkins X solve these issues?

As you can imagine, installing and configuring all that is not easy. It is not uncommon for a new hire to spend a week, or even more, on setting up their own development environment. Consider the following scenarios.

- *What happens if that person should move to a different project?*
- *What if he should work on multiple projects in parallel?*

We can continue with business as usual and install all the compilers and the tools on our laptops. We can dedicate time to setting them up and connecting them with the system. We can continue giving new hires long Word documents that walk them through all the actions they need to perform to be able to develop our applications. Or we can take a different approach. We might be able to create a

full-blown development environment on-demand , specific to each person. We can even make those environments application-specific. And we might be able to make it so fast and straightforward that anyone can do it with a single command in only a couple of minutes.

> Jenkins X allows us to spin up a project-based private development environment with all the tools, configurations, and environment variables we might need to work on any of our applications. That feature is called **DevPod**.

In the next lesson, let's explore the requirements of an efficient development environment.