What Goes into Promoting a Release to Production?

This lesson sheds some light on the reasoning that goes behind promoting a release to production.

We reached the final stage, at least from the application lifecycle point of view.

- We saw how to import an existing project and how to create a new one.
- We saw how to develop build packs that will simplify those processes for the types of applications not covered with the existing build packs or for those that deviate from them.
- Once we added our app to Jenkins X, we explored how it implements GitOps processes through environments (e.g., `staging` and `production`).
- Then we moved into the application development phase and explored how DevPods help us to set a personal application-specific environment that simplifies the "traditional" setup that forced us to spend countless hours setting it on our laptop and, at the same time, that avoids the pitfalls of shared development environments.
- Once the development of a feature, a change, or a bug fix is finished, we created a pull request, we executed automatic validations, and we deployed the release candidate to a PR-specific preview environment so that we can check it manually as well.
- Once we were satisfied with the changes we made, we merged it to the master branch, and that resulted in deployment to the environments set to receive automatic promotions (e.g., staging) as well as in another round of testing. Now that we are comfortable with the changes we did, all that's left is to promote our release to production.

The critical thing to note is that promotion to production is not a technical decision. By the time we reach this last step in the software development lifecycle, we should already know that the release is working as expected. We already

gathered all the information we need to decide to go live. Therefore, the choice is business-related. *“When do we want our users to see the new release?”* We know that every release that passed all the steps of the pipeline is production-ready, but we do not know when to release it to our users. But, before we discuss when to release something to production, we should decide who does that. The actor will determine when is the right time, but does a person approve a pull request, or is it a machine?

Who decides when to release to production?

Business, marketing, and management might be decision-makers in charge of promotion to production. In that case, we cannot initiate the process when the code is merged to the master branch (as with the staging environment), and that means that we need a mechanism to start the process manually through a command. If the executing command is too complicated and confusing, it should be easy to add a button (we’ll explore that through the UI later). There can also be the case when no one decides to promote something to production. Instead, we can promote each change to the master branch automatically. In both cases, the command that initiates the promotion is the same. The only difference is in the actor that executes it. Is it us (humans) or Jenkins X (machines)?

Who approves the PR?

At the moment, our production environment is set to receive manual promotions. As such, we are employing continuous delivery that has the whole pipeline fully automated, requiring a single manual action to promote a release to production. All that’s left is to click a button or execute a single command. We could have added the step to promote to production to `jenkins-x.yaml`, and in that case, we’d be practicing continuous deployment, not the delivery. That would result in the deployment of every merge or push to the master branch. But, we aren’t practicing continuous deployment today, and we’ll stick with the current setup and jump into the last stage of continuous delivery. We’ll promote our latest release to production.

Let’s get started by creating a Kubernetes cluster (if you deleted the previous one).