Learn Continuous Deployment with Bitbucket Pipelines

We'll see in this guide how you can implement a continuous deployment pipeline with Bitbucket Pipelines.

BY STEN PITTET

In software development you often have to do hard tradeoffs when developing an application. If you want to go faster, the general belief is that you will have to trade on the quality of your releases. But there's a development practice that can allow you to save time while releasing faster, and that is continuous deployment.

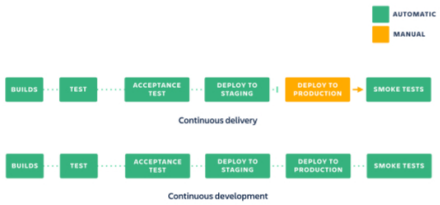
With continuous deployment you remove the stress of deploying software by making it an automated process. Your development team doesn't have to stop anymore and switch context for a release – code is shipped to your customers as soon as a developer has finished their work.

We'll see in this guide how you can implement a continuous deployment pipeline with Bitbucket Pipelines.

Continuous delivery vs. continuous deployment

Continuous delivery is the practice of making sure that your code is always ready to release even if you are not deploying every change to production. It is recommended to update your production as often as possible to make sure that you keep the scope of the changes small, but ultimately you're in control the rhythm of your releases.

In continuous deployment, new changes pushed to the repository are automatically deployed to production if they pass the tests. This puts more emphasis (read: pressure) on your testing culture, but it's a great way to accelerate the feedback loop with your customers.



Requirements

Time:

30 minutes

Audience:

You are new to continuous deployment and/or Bitbucket Pipelines

Prerequisite:

- Create a Bitbucket Account
- Follow the continuous integration tutorial
- Follow the continuous delivery tutorial

Try it free

Setting up a continuous deployment pipeline

Continuous deployment is a great way for teams to accelerate development. It removes the impediments related to the release approval process, and it allows developers to get feedback from customers as soon as they're done with their work. Issues are easier to identify and fix, and there's less context switching as there's

no release time anymore.

Configuring a continuous deployment pipeline with Bitbucket Pipelines is very easy. We will see how to do it with a simple Hello World application that goes through a staging environment and acceptance tests before getting released automatically to production.

You can find the source of the Hello World app in the repository linked below.

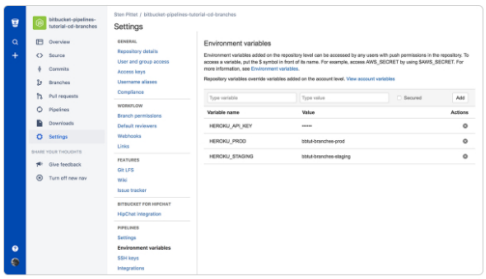
- Hello World application

Preparing the deployment to Heroku

Before we begin we need to add a few environment variables to be able to deploy our application on Heroku:

- HEROKU_API_KEY: You can find your API key in your Heroku account
- HEROKU_STAGING: name of your staging environment
- HEROKU_PROD: name of your production environment

Go to *Pipelines > Environment variables* in your repository settings to add the variables.



Setting up environment variables to deploy to Heroku

> We're using Heroku in this guide, it is certainly possible to adapt this example to other hosting services. You can find other deployment guides in the documentation.

Setting up the pipeline

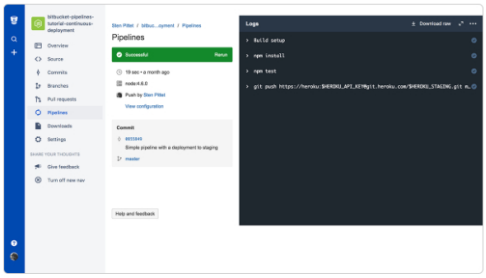
Our workflow will be straightforward:

- Build the application.
- Run tests on the build.
- Deploy to staging.
- Run some acceptance tests on staging.
- Deploy to production.

You'll see that it's very easy to create the corresponding pipeline configuration. We'll start by adding our automatic deployment to the staging environment to make sure that it's happening properly on every push.

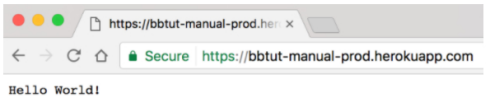
```
image: node:4.6.0   # Doing a full clone to be able to pu
```

You'll notice that we're using a full clone to make sure that Heroku doesn't reject our push. Then we're also using a branch-specific pipeline to make sure that we only deploy staging for changes pushed on main and not on other branches. You can push this configuration to Bitbucket to see it in action.



Automatic deployment to staging is completed

At this stage, we now have our *Hello World* application deployed on staging.



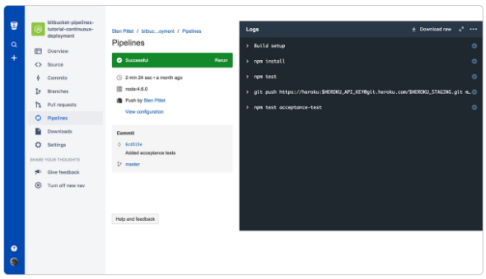
We can now move on to the next step and add our acceptance tests. Acceptance tests are here to make sure that our application behaves as expected in a production-like environment (staging). The goal is to remove uncertainties due to differences in configuration between the environment used to test the build and your production.

If you look at the code of our app our test is doing something extremely simple as it's just looking for the presence of the string "Hello World" in the page. In a real application we recommend to have acceptance tests that go much further and verify that all the underlying services used by your application (database, cache, 3rd party, etc.) are working properly.

So let's add our test right after our deployment to staging.

```
image: node:4.6.0   # Doing a full clone to be able to pu
```

After pushing this new configuration to Bitbucket we can see our pipeline completing successfully.

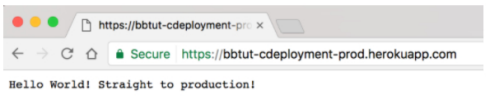


Bitbucket Pipelines now runs acceptance tests after deploying to staging

All that is left now is to add our production deployment at the end to complete our continuous deployment pipeline.

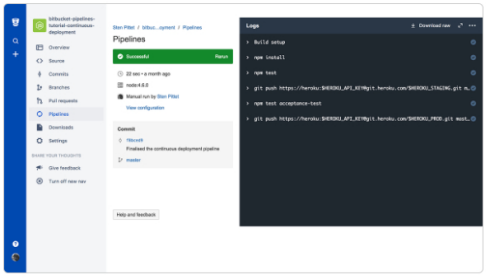
```
image: node:4.6.0   # Doing a full clone to be able to pu
```

A final push to Bitbucket will take our code changes through all the pipeline, building and testing the code then deploying to production after successfully verifying that staging works. In this case the homepage has been updated with a different message to make sure that it would get deployed all the way to production.



Our new message went to production as expected!

You can check the pipeline to make sure that it went properly through all the different phases.



Checking that the code went through the pipeline

It is important to stress out the importance of having a

good test suite, as well as having real-time monitoring in place before you go ahead and implement continuous deployment for your own repositories. From now on changes will go straight to production as soon as they're pushed to the main branch so it's all the more important to