Understand and set up a simple deployment pipeline

Read this article first:

http://martinfowler.com/bliki/DeploymentPipeline.html

Deployment pipelines can become very complex - typically the larger the project is and the more teams are involved, the more complicated the automatic deployment process is. For our purpose, we are going to adopt a simple but effective deployment pipeline for solo dev or small team situations.

1) Follow the Github Flow process, and after you complete a feature, merge the pull request back to the master branch.

2) Run your entire test suite. This is very important and you should always do this before pushing code to Github. The master branch should always have working code. Developers call this "don't break the build".

3) Deploy your code to the staging environment and test on your staging server manually to make sure that the new feature works. Again, your staging server should be as close to your production server as possible.

4) Deploy code to production. You want to deploy code to production as often as possible to get feedback as fast as possible. If something breaks, it's much easier to fix it with your mind still fresh with the feature you worked on. In some teams this step is out of your control and there are strict production deployment times. In that situation you'd go back to step 1 and build your next feature.

When you deploy to the staging server or the production server, effectively what you are doing is to push code from your local master branch to the corresponding Heroku server's master branch.

Why is it important to also push/deploy from your local master branch? The problem with deploying from a feature branch is that you miss the step of integration with other developers. For example, while you work on your feature, I might have push out my feature as well. If you deploy from your branch even if it does work you didn't test if your feature would integrate with the feature that I just deployed, AFTER you branched off. By forcing everyone to integrate with master first, now every time you push, you HAVE to integrate with everyone else's push at that moment. If someone pushed code after you branched off, this would force you to do a pull first on master, then integrate with it, otherwise you can't push.

Note that you should never deploy to production before you go back home or go to sleep. Make sure you can be around your computer for some time to fix anything that could happen after a production deploy. Your error monitoring app will alert you with emails if they do happen.

In summary, our simple deployment pipeline process is `run entire test suite locally -> deploy to staging and test -> deploy to production`. This is probably good enough for solo developers or small teams. We'll talk about Continuous Integration and Continuous Deployment with more automation in the next several topics.

You marked this topic or exercise as completed.