**Use Jenkins and git-ftp to deploy a website to shared webhosting**

Gone are the days of manually deploying code to servers! I recently started using Jenkins to deploy several simple websites that I’ve built over the years. However, some of these websites are hosted on a shared webhosting. This means: no shell access available for pushing a git repository to it. In this post I’ll show you how to configure Jenkins to deploy websites with FTP.

Jenkins?

Let’s quickly go over what [Jenkins](https://jenkins-ci.org/) is and why you should use it:

Jenkins is an award-winning, cross-platform, continuous integration and continuous delivery application that increases your productivity.

Sounds sweet right? You can set it to automatically build, unit test and deploy websites after each git commit.

Deploy to shared webhost?

I know what you’re thinking: shared webhosting? Really? Why would I need to deploy to a shared webhosting if [I’m hosting my website on Amazon S3](https://www.savjee.be/2013/02/howto-host-jekyll-blog-on-amazon-s3/) and if [I’m working with serverless applications](https://www.savjee.be/2016/01/Building-serverless-anagram-solver-with-aws-dynamodb-lambda-s3-cloudfront-api-gateway/)?

Well, I work as a volunteer for a non-profit and they don’t have a big budget for the website. They chose to use shared webhosting because it’s very cheap and sufficient enough for their Laravel application. Sadly though, every time the websites needs an update, I have to use my FTP client, connect to the server and initialize a sync. Not really ideal. So let’s automate that!

The idea is simple: the entire codebase of the website is tracked by git. Everytime something changes I make a commit and that commit should land on the server.

Getting started

I’m assuming that you’ve already set up Jenkins and git on your server. If not, follow the instructions [here](https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins). It’s as simple as running an apt-get install, yum install, docker pull or java -jar jenkins.war. I installed Jenkins on [my tiny and cheap RamNode VPS with just 128mb RAM](https://clientarea.ramnode.com/aff.php?aff=1321) (and -surprisingly- it works extremely well!).

Jenkins has built-in support for deploying the contents of a git repository to a FTP server. Great! Except that it’s rather limited. Each builds uploads **all files** to the FTP server which could take a long time if you have a lot of assets or if you have a slow server. Additionally, the built-in FTP deployment **doesn’t remove a file** if you have removed it from the git repository. To fix this you would need to remove all files from the server and re-upload them every time you want to deploy! This could cause several minutes of downtime depending again on the size of your website.

Enter git-ftp

A few Google searches later, I found the perfect tool for the job: [git-ftp](http://git-ftp.github.io/git-ftp/). This shell script does one thing: it deploys all **changed** files to a FTP server. The tool uploads all the files that where changes and deletes all the files that where removed between commits.

I started by installing it onto my Jenkins server (Ubuntu server):

sudo apt-get install git-ftp

The tool has two commands that we’re interested in: init and push. The init command is used only once and uploads all the files in your git repository to a FTP server. It also creates a small file on the server (.git-ftp.log) that contains the checksum of the commit that is currently deployed on the server. You can run it like this:

git ftp init --user USERNAME --passwd PASSWORD ftp://YOUR-FTP-SERVER-ADDRESS/path/to/website/

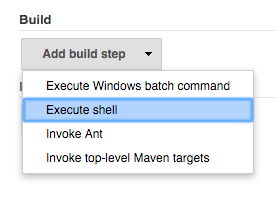
After initializing the FTP server you can make a new commit in git and deploy the changes with git-ftp using the push command:

git ftp push --user USERNAME --passwd PASSWORD ftp://YOUR-FTP-SERVER-ADDRESS/path/to/website/

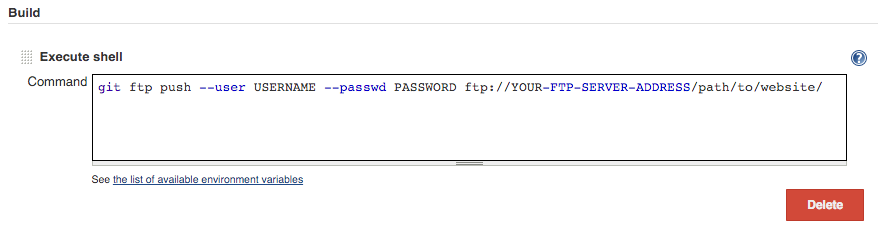
Integrating git-ftp in Jenkins

The last thing we need to do is tell Jenkins to use git-ftp after each successful build to deploy the changes to the FTP server.

Modify your Jenkins job and add a Execute shell step to your build process:

*Add a “Execute shell” build step to Jenkins*

A new textfield appears where you can write your shell command that should be executed. Fill it in and save the changes.

*You can execute any shell command. Here I use git-ftp to deploy to the server.*

That’s it! I don’t have to use my FTP client anymore and I can rest assured that the code on the server is kept up-to-date!