4 Really Simple Ways To Improve Your Software Deployments

Craig Godden-Payne
Mar 8 · 4 min read ★

Let's use this hypothetical deployment process, and see what we can do:

1. Build software artifacts

2. Zip artifacts
3. Upload artifacts to server using SFTP
4. Send email notifying users of restart
5. Stop webserver
6. Tail server logs to check for stop
7. Backup files on webserver
8. Move new artifact files into webserver
9. Start webserver
10. Tail server logs to check for success
11. Send email notifying users of completed restarted

Seems like a pretty straightforward deployment pipeline. Unfortunately, life isn't always this kind, but we can interchange pretty much any deployment pipeline with the above using this guide. Regardless of how nasty the current deployment process is, you just need to work in small steps.

Step 1 — Stop doing manual deployments

Some companies *still* deploy software manually, and if you do, there isn't much you can do to improve your deployments. The first stage is to find ways to prevent yourself from doing manual deployments.

What is a manual deployment?
If it comes to release day, and you manually log onto a server and copy some files over, you fall into this category. There are many problems with manual deployments:

- **Difficult to spread the knowledge;** Sometimes one or a small number of people know how to deploy, what happens when the process changes, or *the deployment guy* is off sick, or leaves?
- **Lack of consistency;** What state is your live server in? If it has been configured manually, it is likely got an unknown amount of software on there, of which is probably grossly out of date

Step 2 — Script what you can and automate

It's better to script your deployment than to have a manual deployment.

This second stage is already a step in the right direction. By scripting parts of the deployment of your application, it means that eventually you can automate it.

Why script your deployment?
Scripting is the first step to automation. It also shares what is happening with whoever is running a deployment, it empowers others to be able to deploy, not just a select few.

With our example, we can create a bash or cmd script to automate at least some of the stages and combine them together.

The idea is just not to have a script that does it all, maybe consider splitting into pieces, where you would have a checkpoint of which you can rerun a deployment. Try to make the deployment idempotent also, so it can be ran multiple times without issues.

Your deployment scripts could now look something like:

1. Build

- Build software artifacts
- Zip artifacts
- Upload artifacts to server using SFTP

2. Backup

- Send email notifying users of restart.
- Stop webserver.
- Tail server logs to check for stop
- Backup files on webserver

3. Deploy

- Move new artifact files into webserver
- Start webserver
- Tail server logs to check for success
- Send email notifying users of completed restart.

To make the scripts more reliable, it may be worth splitting up more, or maybe changing the process around, to prevent the amount of downtime, or improve the reliability of the Stop/Start webserver.

Step 3 — Use a proper CI/CD solution

Using a CI/CD solution to track deployments not only leaves an audit trail of what has happened. It also has many other benefits, such as artifact versioning and backups, auto scheduled backups, it can hide credentials in a credentials store, etc.

There are many different CI/CD solutions out there, and each one will have its own use case, some are free, some are paid but generally, any CI/CD solution is better than none.

A few to mention:

- Jenkins
- Team City

- GOCD
- CircleCI

You can usually set these up to use the scripts you already are using, with more benefits such as increasing the build number and also hiding access keys etc. in a credentials store, so they are not exposed in your scripts.

Step 4 — Shout about your successes

Look how awesome you are, you have successfully deployed 100 times without a single failed deployment. Having the benefit of easy deployments, easy rollbacks, versioning etc. is just bliss, tell people about how awesome this is.

One of the places I used to work (it was a relatively small company, say 30 developers) and we deployed to production sometimes 100 times a day. By automating your deployments, you will start to give other parts of the business confidence in your ability to deploy, which is empowering, it makes everything move faster. You can usually template deployments, and next time a similar software needs to be deployed, it is just reusing the template, effectively speeding up your first iteration.

Software Development Continuous Delivery Deployment Automation Product Management