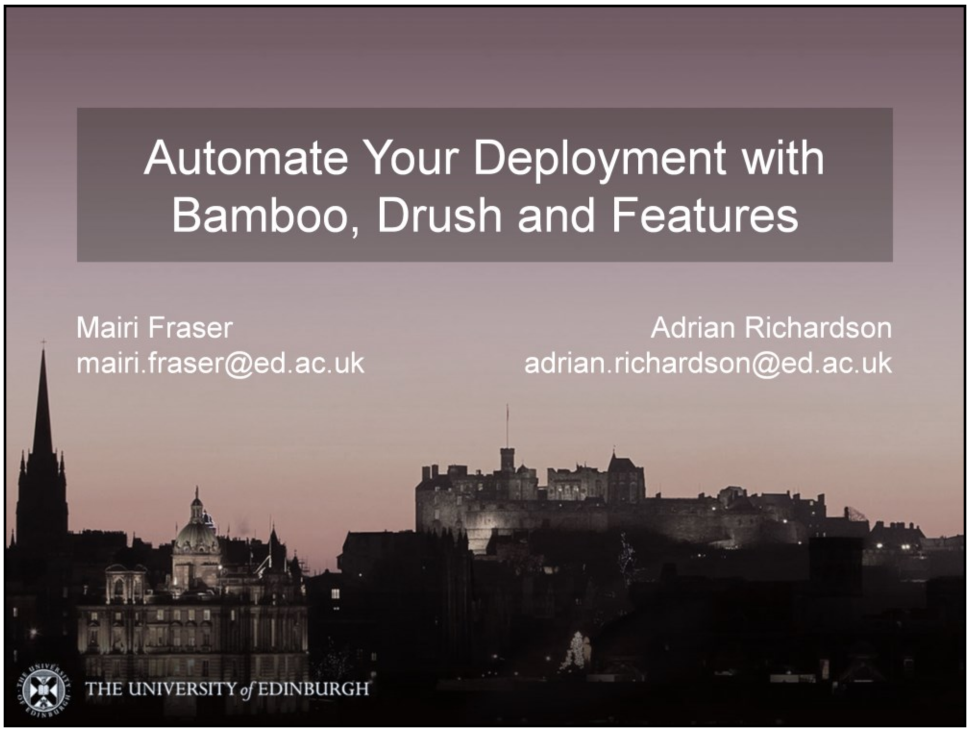
Automate Your Deployment with Bamboo, Drush and Features

DrupalCamp Scotland, 9th – 10th May 2014

THE UNIVERSITY of EDINBURGH

This presentation was originally given at DrupalCamp Scotland, 2014.

http://camp.drupalscotland.org/



Automate Your Deployment with Bamboo, Drush and Features

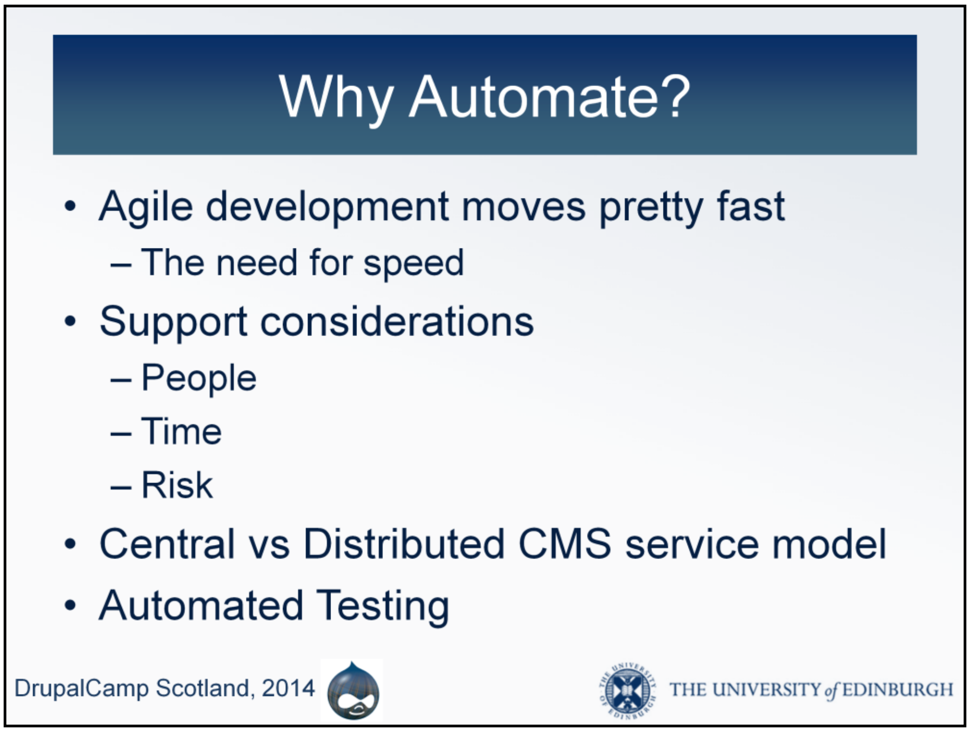
Mairi Fraser
mairi.fraser@ed.ac.uk

Adrian Richardson
adrian.richardson@ed.ac.uk

THE UNIVERSITY *of* EDINBURGH

We are 2 of the developers working on the University's ongoing project to develop a Drupal CMS service to manage the University's web presence. If you attended the DrupalCamp Business Day, you may have heard our colleague Stratos talking about the business factors in the University's decision to move to Drupal.

Today we're going to take a more technical viewpoint and talk about automated deployment, one of the key aspects of our development process, which will ultimately underpin the support of the University's Drupal service. I will give a high level overview of why we have taken the approach of automating our Drupal deployment, and how our process works, then Adrian will dig a bit more into the details of the deployment process.



Why Automate?

- Agile development moves pretty fast
  - The need for speed
- Support considerations
  - People
  - Time
  - Risk
- Central vs Distributed CMS service model
- Automated Testing

DrupalCamp Scotland, 2014

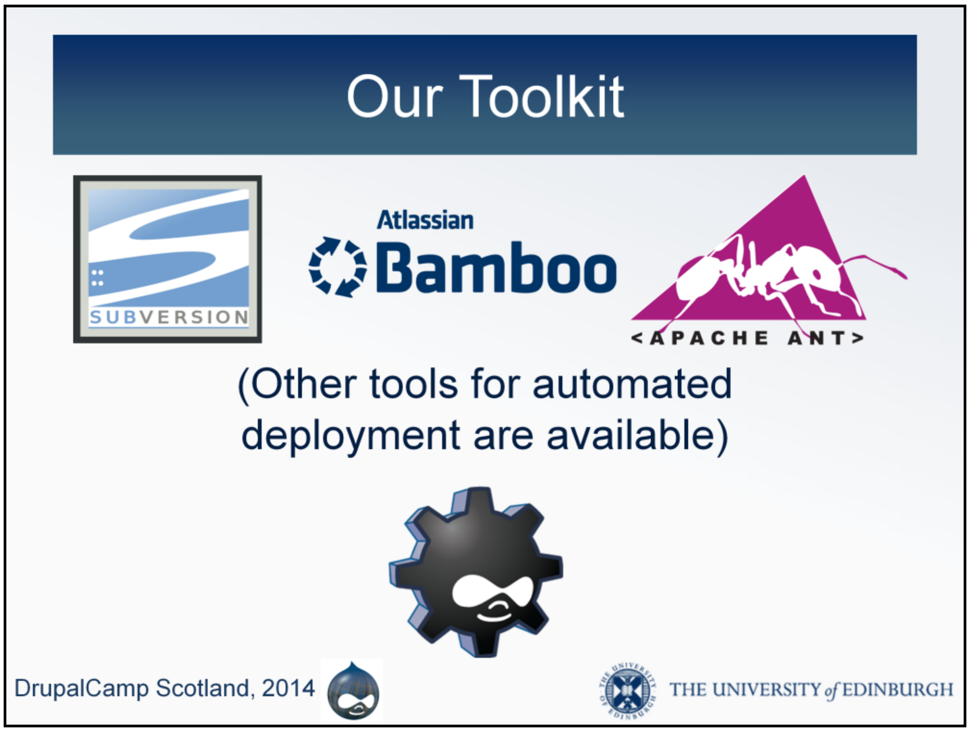
THE UNIVERSITY of EDINBURGH

Development considerations
- automated deployment facilitates greater frequency of deployments
- any time spent on deployment is time not spent working on new features
- over the past couple of years, we've seen the benefits of automated deployment in streamlining our development process

Support considerations
With our current proprietary central CMS, we have been able to automate parts of the deployment process, but there is still a lengthy series of manual steps that fills in the gaps between automated stages. The problem here is threefold.
- Too many people have to be involved for each a manual deployment to ensure everything is covered; with automated deployment, those people only need to be involved once in signing off the aspects of the deployment relevant to them.
- It takes more than a day of elapsed time to update everything, including back-end CMS and front servers, followed by up to 5 or 6 days of manual testing to check the website, depending on the complexity of the changes deployed.
- Regardless of the skill and competence of those involved in the deployment, things are inevitably missed, and it can be difficult to detect where something has gone wrong. Up against scheduled deployment windows & with 650+ editors and publishers relying on the central CMS service, the risk involved in manual deployments is high. Automation makes the process repeatable, reducing risk.

Central vs Distributed CMS service model
- The University's highly devolved structure means we have to consider those in the University who manage their own websites outside of the central website.
- The primary focus of the CMS development work we're doing is to build a CMS to replace the existing central CMS. However, another aspect of our vision for Drupal is to provide a University Drupal distribution which can be re-used around the University. The idea is to have a Drupal profile that other Drupal developers in the University can download, or alternatively they might pick & choose parts of that distribution, such as the theming layer, to use in their own sites.
- Automated deployment will support the delivery of the central CMS, and we will be able to use the same automated deployment tools to support how we push updates to the Schools & Units that are making use of the University's Drupal distribution.

Automated Testing
- Including automated tests in our deployments for integration and regression testing reduces the need for manual regression testing, so is partly a support consideration. However, it also means we can quickly see following deployment whether we have broken something, which is crucial when several people are committing code to the repository.



Our Toolkit

(Other tools for automated deployment are available)

DrupalCamp Scotland, 2014

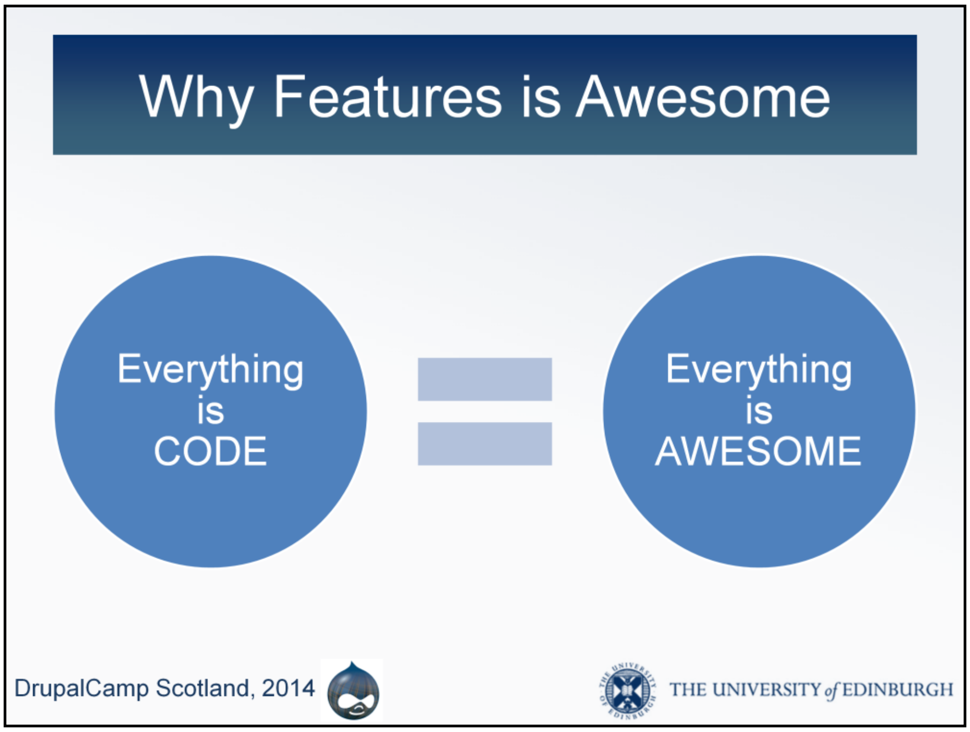
THE UNIVERSITY *of* EDINBURGH

Our Toolkit
Before adopting Drupal, we already had a successful toolkit for automated deployments in place as a result of earlier Agile projects. So, what tools do we need to build an automated deployment process?

- Version control
  - We use Subversion for version control (we do use Git when working with Drupal community code, but at the moment SVN is our internal version control tool)
- A tool to run the deployment plans & tests
  - We use Bamboo to drive the deployment process, running build plans and tests, providing a live view of deployment process and a full history of completed deployments including logs
- A workhorse to actually carry out the deployment tasks
  - Apache Ant is the workhorse in the process as our deployments are built using Ant XML scripts

Other automated deployment tools are available:
- Bamboo was the most obvious choice for us because it integrates well with other tools we use such as JIRA, Hipchat, and Confluence. However, there are other options such as Jenkins in the open source community which can be used to do the equivalent of what we do.
- Tools like Maven can be used in place of Ant.
- Basically, as long as the individual components are present, you can swap any of them for something else that works better in your particular situation.

In order to plug Drupal into this toolkit, we make use of drush, Drupal's command-line tool. As Adrian will demonstrate, drush is extremely convenient for use with Ant.



Why Features is Awesome

Everything is CODE = Everything is AWESOME

DrupalCamp Scotland, 2014

THE UNIVERSITY *of* EDINBURGH

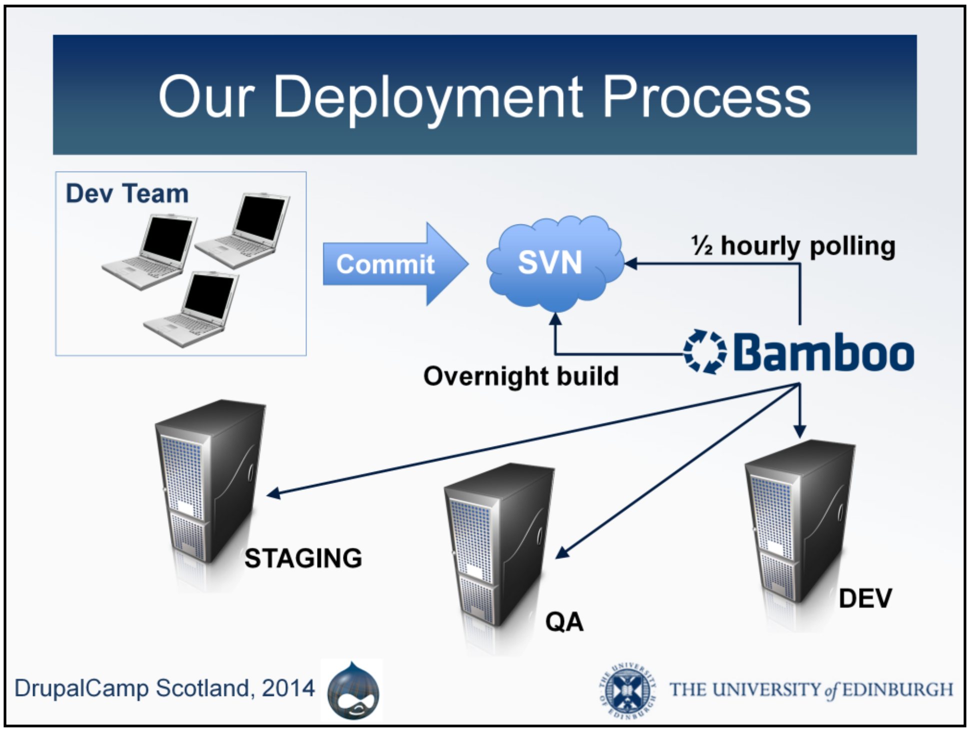
Where Features comes in
The command-line tool drush is a powerful tool to support automated deployment., but it's not enough for us to achieve everything we need with automated deployment.

For that, we needed to add Features to the mix.

A key factor in using Drupal for us is control over the configuration settings that are deployed. This applies to both the central CMS service and the Drupal distribution that we package for wider use in the University. We don't want manual intervention in Drupal's administration panels to be necessary to set up a site with the University's default Drupal configuration. This means we need everything to be encapsulated in code.

The Features module allows us to export related configuration settings into code as custom features. These can then be deployed along with all of our other Drupal code, and we can revert features during the automated deployment process to enforce the required configuration settings for the University's central CMS. Using Features in this way allows us to deploy everything as code, and also to see in the Drupal admin interface where any deployed configuration settings have been overridden. This latter aspect of Features will be extremely useful when supporting use of our Drupal distribution around the University.

Adrian will talk more about how we use Features…

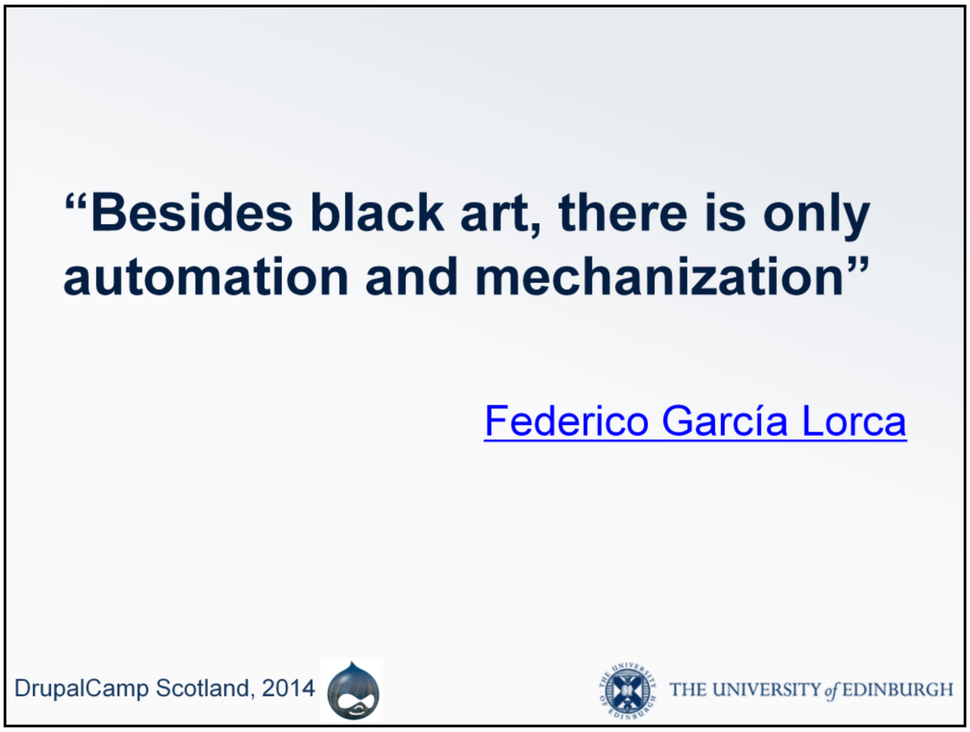


Our deployment process

- The development team work with a local Drupal installation that is built from our SVN environment
- We have two bamboo plans
    - a ½ hourly plan which polls SVN for changes & deploys any changes committed to the Dev environment, integrating changes by all developers
        - a sub-set of the automated tests relating to the work for the current iteration is run at this stage
    - an additional plan which runs overnight, **only** if the last Dev plan was successful, which updates our QA environment with a completely clean build and runs full automated test suite
        - there is manual intervention following the overnight QA update to push the changes to the Staging environment since we don't want that to happen completely automatically (Staging is updated without clearing down the database)

It's not shown in the diagram, but ultimately there will be a further manual step following any successful push to Staging which will update the Live environment.

This process is by no means set in stone. We have a lot of work still to do to finalise the process for managing the University's Drupal distribution & build this into our deployment process. Also, we are currently only using drush to deploy our own code via the automated deployment – updates from drupal.org are not part of the automated process. We still have to manually carry out the updates locally and commit any changes to our SVN repository. Ultimately, support of the central CMS will be much smoother if we can extend our use of Bamboo & drush to automate the Drupal update process on our Dev servers & notify the support team when changes are ready to be evaluated.



"Besides black art, there is only automation and mechanization"

[Federico García Lorca]

DrupalCamp Scotland, 2014

THE UNIVERSITY *of* EDINBURGH

Regardless of the scale at which you're operating, you can benefit from the efficiency gains from automating your deployments, and Adrian can show you a bit more about how drush & features can help with that…

This is a link to Adrian's prezi from the day:
http://prezi.com/2w7vppwwdzfs/deploying-drupal-drupalcamp-scotland/

There are some limited notes on Adrian's prezi, but the presentation on its own should give an idea of what his part of the talk covered, and it includes some code samples for the Ant scripts, drush commands and automated tests that we are using.

THE UNIVERSITY *of* EDINBURGH