UCI BioSci WordPress Multisite Deployment Automation

"An automated approach to update a complex multisite WordPress installation."

Submitted By

Carolina Manel Romero cmanel@uci.edu
Application Developer for the School of Biological Sciences.

Matthew R Martinez mrm@uci.edu
Director of Information Technology Service s for the School of Biological Sciences.

Contributors

Eric Sanchez eric.sanchez@uci.edu,
Network Administrator for the School of Biological Sciences.

Kameron Balutch kbalutch@uci.edu,
Technical Support Analyst for the Office of Information Technology.

Overview

The UCI School of Biological Sciences is a large, dynamic, and diverse academic unit. Having all the websites up and running at all times is critical as the websites are a primary connection to current, former, and prospective students; faculty and staff; and donors who rely upon the websites for their daily activities and current information.

The BioSci WordPress multisite installation includes a total of 31 websites, including the main BioSci site, the four academic BioSci departments, and other BioSci initiatives. An interruption of service might prevent staff members from posting crucial information and keep School operations running.

Keeping the multisite WordPress installation updated ensures that we have the latest security patches, combined with the best speed and performance, all while keeping the data secure, integrated, and functional.

In 2018 we started to implement an automation plan to ensure that all the BioSci multisite websites work without downtime, disruption, or data loss as a result of the WordPress deployment. Our plan was to use state-of–the-art software to update the multisite efficiently. Because of the constant updates to the platform, plugins and themes from public and private repositories; development of new features for each BioSci Department website; and maintenance of current code, automation was the key to our success. Our plan was not only to automate the deployment, but to implement testing for quality assurance and have a safe workflow that allows a rollback if needed. We created a process to replace the old manual deployment with a totally automated one with no human intervention. The project took 3 weeks to finish the implementation, from the Jenkins set up to writing the test cases, and we started seeing the benefits right away.

Approach

We accelerated the process to interact with the multisite using WP-CLI. This is a WordPress command line tool that performs the same tasks as using the admin panel but works almost

instantly. It not only helps us to automate the WordPress core and public plugins and themes, but also to test if the updates were successfully implemented.

Testing each of the 31 websites in the multisite after every deployment was the most time-consuming task when we did it manually but testing all of them was a must. We automated the process using Behat, which helps us to test the behavior of each website, the content, and links from the end user's point of view. Doing so, we made sure that all the data that our users are looking for is still available.

We use all the following technology to complete our automation plan, adding open source tools to the tools we already have:

- **GitHub Enterprise** as a source control with all the private repositories for plugins and themes.
- **Composer** to install all private repository dependencies.
- **Ansible** with asistrano to make the playbooks for the deployment.
- **Behat** as quality assurance to test the websites in the multisite.
- **Jenkins** to run a pipeline job to start and monitor the deployment process.
- **Microsoft Teams** to communicate about the deployment status.
- **Amazon S3** backup service.
- **WP CLI** command line to execute WordPress actions.

Timeline

Our first sprint for the automation plan using Jenkins started in September 2018. We ran our first Jenkins job for deployment in October 2018.

The process

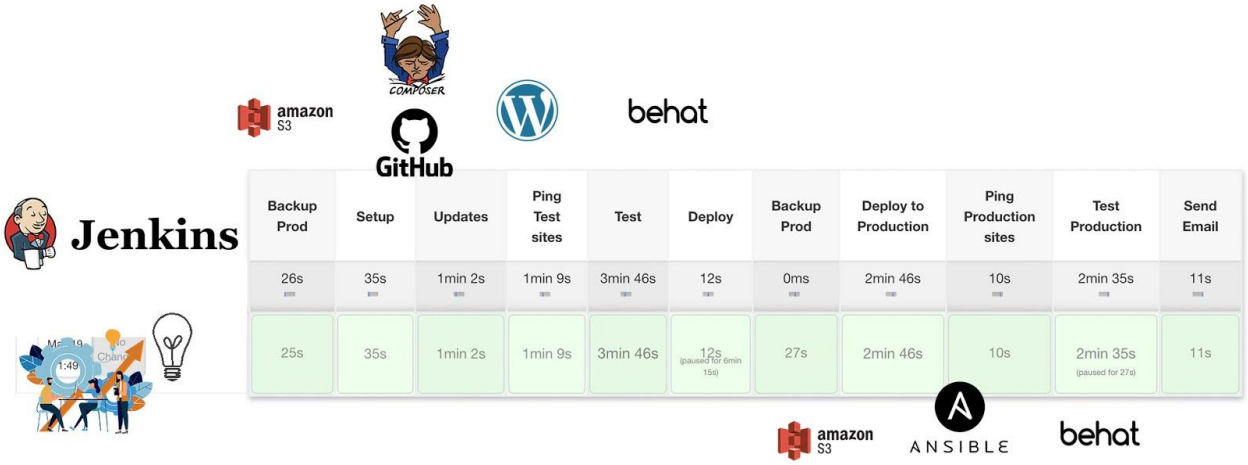
The School of Biological Sciences has 2 servers we deploy to (staging and production) that hold a test and production version of the BioSci WordPress Multisite. With collaboration from the Office of Information Technology and the network administrator of BioSci, we set up a Windows server with Jenkins for secure server interconnectivity with the BioSci servers.

Jenkins runs the following pipeline:

Pipeline

1. **Monitor -** Jenkins runs a cron job in the production server to check for any updates in the core, theme, or plugins. If there are updates, then a backup job will run the next step. If not, it finishes here.
2. **Backup -** Because the staging server has important test data for staff who may be testing, we run a backup of the database and send this backup to Amazon S3 storage service using a rest API call.
3. **Building a new installation -** A job that makes a new and clean multisite installation using git and composer in the staging server.
4. **Update -** A job that updates the core, plugins, or themes that reside in the WordPress public repositories. We update the private components with Composer. If there were changes, then it will push to the version control repository and add a commit message with the updates and versions that were done. This is very important for tracking purposes.

5. **Test -** A shell script will run a series of test cases built in Behat. The test cases have scenarios to check for the header, footer, text and important elements. We run a minimum of 5 test cases per site. If any of the tests fails, it will prevent the deployment to production and will stop before step 7.
6. **Microsoft Teams -** It will connect to MS Teams using a common communication channel to let the admin know about the process status.
7. **Backup -** Data is one of the most valuable assets we have in the school. We backup our production database and send the backup to Amazon S3 before doing any deployments on the production server.
8. **Deployment -** At this point we know the updates are pushed to the repository and it is safe to install the package in production because we did a backup and completed testing. It is time to deploy to production. This job runs a playbook in Ansible with the Ansistrano deployment role to set up the updated installation on the production server.
9. **Test -** The test on production is a series of sub-jobs to run the step 5 (above) again, but this time for the production server and includes a ping of all the sites in the multisite.
10. **MS Teams -** Admins and site administrators are notified about which components (plugins, themes, core) were updated.
11. **Rollback -** Has something gone wrong? We have never had a bad deployment at this point, however there is always a way to rollback in seconds using Ansistrano Rollback.



Measurements of Success

Reduce down time to zero due to WordPress update deployments.
Reduce the time to apply WordPress update deployments by 90%, from about 3.5 hours for a manual deployment to 15 minutes.
Reduce the time to test all the sites in the multisite by 98.4%, from about 2.5 hours of manual testing each to 4 minutes average in total, running approximately 200 test cases.

Conclusion

Since we have implemented this Automation, we have been able to:

Execute and handle data backups and backouts accurately.
Have zero data breaches due to security vulnerabilities.
Catch errors in custom code in time before it goes to production and causes downtime.

Reduce human errors during WordPress update deployments.
Increase capacity to manage more sites with minimal effort and completely integrate them in the automation process.
Reuse the Ansistrano playbook to implement an automated deployment of 3 more custom BioSci Symfony web applications.
Improve developers' productivity using their time on other important projects.

The software and functions are readily deployable using available and open source software at no additional costs to other groups on campus and throughout UC who run WordPress multisites.