weaveworks

CASE STUDY

THE CUSTOMER

Axel Springer SE is the largest publisher in Europe, with numerous digital and multimedia news brands, such as Bild, Die Welt, and Fakt. Headquartered in Berlin, Germany, the company is active in more than 40 countries, employs more than 15,000 people and has a total revenue in excess of €3.3 billion.

The infrastructure team at Axel Springer is responsible for managing Kubernetes clusters and its ecosystem of cloud native tools. The team recently adopted GitOps in combination with GitHub Actions as a way to spin up fully functional test environments within minutes.

"The benefit of GitOps for us is to get rid of old-fashioned tools like Jenkins, having complex CICD pipelines and discussions within the teams 'who is responsible for which part of this complex CICD platform." **- Andreas Prang, Team Lead Infrastructure Services, Axel Springer SE**

CHALLENGES

Axel Springer is at the beginning of their cloud native transformation with approximately 10 EKS clusters running on site and in production. With multiple teams of engineers rolling out new services and updates to its digital news assets, the infrastructure team needed an automated and secure method to test, and review new features and updates before they get deployed to production. In order to support the speed of changes to the news portals, the team had to modernize their continuous deployment tools with a focus on automation and stability of test environments. Both developers and product owners needed a production like environment to test and approve new features before deploying them to the live site.

Complex and error prone CICD tools

As the team embraced modern operations with cloud native tools such as Docker, Kubernetes and Helm, the operations team quickly realized that delivery lagged behind because of complex, error prone and sometimes manual workflows. Old fashioned automation and CICD tools like Jenkins needed to be replaced in order to keep up with the speed of development, testing and deployment.

axel springer

Industry: Publishing
Location: Germany

HIGHLIGHTS

- Easy set up and maintenance
- Streamlined GitOps pipelines
- Automatic test environments

KEY BENEFITS

- Modern GitOps tooling for high performing teams
- Automated test and deployment pipelines
- Reduced cognitive load for developers

CONTACT US

www.weave.works

sales@weave.works

Automate test environment creation

The infrastructure team needed to give developers the opportunity to run services easily for their individual platform. That means they had to put guardrails in place so developers and product owners can easily and independently create test environments for developing and running acceptance tests before securely deploying updated Docker images to production clusters.

Reduce cost overhead

Like most IT organizations, keeping costs down is always an important consideration. One of the challenges in this case was to only create test environments when needed, rather than keeping them running constantly in the background.

Solution

The infrastructure team set up a Kubernetes platform that allows developers to create test environments on demand. Using a combination of GitHub Actions, Container Registry, and GitOps, developers can automatically spin up test environments in minutes. These automatic instant test environments are referred to as “Phoenix environments”.

Two main stakeholders depend on the Phoenix environments:

1. The developers who want to test their code changes in an environment that is as close as possible to production.
2. The product owners who want to view and approve (acceptance test) any new features before they go live.

Since all of the infrastructure definitions are kept in Git, and reconciled with the production clusters on a continual basis with Flux, the team can use GitOps to create and manage test environments that are replicas of what is running in production.

On demand test environments identical to production

The team takes advantage of GitOps and Flux to manage and initialize a new test environment. When a developer starts a new feature, the first thing they do is create a new feature branch in Git. Once the feature branch is created, a GitHub Action triggers an `onCreate` event.

“The end goal is for each environment to be as close as possible to what’s running in production. We had considered creating a new namespace on the cluster to deploy a few services in order to test its functionality, but instead chose to create a copy of the live environment that includes the entire toolchain, including all monitoring, logging and other services in place for our production system.”
– Andreas Prang

The GitHub Action event gets the feature branch and the project name, and then sends that data to a more centralized GitHub Action. This action pulls the live environment’s YAML definitions from the Flux repository, and then copies and updates the definitions with the feature branch and pushes them back to the Flux repository.

Flux notices the changes and pushes them to a newly created feature namespace on the cluster. It also sets up some of the other Kubernetes resources like secrets in the configuration map. The Helm operator then deploys all the monitoring and logging services needed to run in this new test environment within the namespace.

Automating test environments and making them available when needed avoids the problem of having a bespoke QA cluster running at all times. Instead, test environments are created on demand and are available during the development of the new feature. Once a feature has been tested by the developer, the product owners are sent a unique URL to the test environment so that new updates can be reviewed and approved before going live in production.

In addition to this, after the pull request is merged by the developer, the branch is automatically deleted and the test environment is destroyed. This avoids incurring any extra computing costs.

Results

With less than 200 lines of code, the infrastructure team at Axel Springer was able to use GitOps with GitHub Actions to completely automate the creation of test environments. Developers and product owners can now easily test and approve new features on a production like environment before deploying them to the live site.

Reduced cognitive overload

Because GitOps uses familiar tools and workflows, setting up a new test environment does not require the team to learn a whole new set of tools and eliminates context switching. Environments are spun up and torn down straight from Git making it fast and simple for the development team to test new features as they are being developed.

Cost reduction

Because Phoenix test environments are fully automated, environments are spun and destroyed on demand which saves the costs of running them continuously.

Automated continuous deployment pipelines

Developers increase their productivity with fully automated pipelines. Once the update or new feature is tested and reviewed, developers build the Docker image with the new feature and deposit it to the GitHub Container Registry.

Flux notices the new image and automatically and securely deploys it to the production cluster without the developer having to log into Kubernetes.

It is worth mentioning that this workflow also allows teams to view the history of all deployments in git and can serve as an audit trail.

“*We achieved a setup with less than 200 lines of code and with this setup which is very dynamic, we can manage many product projects and services. It’s completely automated and it’s identical to production.”* **– Andreas Prang**

Watch the full talk from GitHub Universe 2020.