weaveworks

CASE STUDY

MediaMarktSaturn Technology

INDUSTRY: Retail

LOCATION: Germany

HIGHLIGHTS

- No downtime when deploying
- Fast disaster recovery
- New deployment process using existing toolchain

KEY BENEFITS

- Full visibility and observability across deployments
- GitOps and Flagger provides completely automated canary releasing
- Increased team confidence around deployments
- Near 100% reliability of systems

The customer

MediaMarktSaturn Retail group, based out of Germany, is Europe's leading commerce company for Consumer Electronics with total sales of over €20 billion. The organization runs around 1,000 stores in 13 countries with about 53,000 employees, and their integrated online and store offerings reach millions of customers every day.

MediaMarktSaturn has already adopted DevOps practices and are operating parts of their stack in the cloud. The central platform team manages Kubernetes clusters on Google Kubernetes Engine, where GitOps operators are provisioned via Terraform. The cloud-native applications are structured as microservices and use Helm charts to implement cloud services. Continuous integration and testing is executed using GitHub Actions and since the team was an early adopter of GitOps, Flux automates deployments the GitOps way.

The group's internal systems are as diverse as the teams that operate them. Each team has autonomy to choose their own tech stack, and yet, teams share best practices with each other. Systems see a wide range of loads according to season, and type of service. Some services have 300 operations per second (OPS), while others have 1,500 OPS, and during peak times like Black Friday, this number can shoot up to extremely high volume like 2 to 3,000 OPS.

Challenges

▸ Downtime during deployment

The biggest issue MediaMarktSaturn faced was downtime when manually deploying updates and new releases. The team was forced to put significant time into deployment planning that was only executable during off-peak hours. The team also lacked visibility into deployments, which made it harder to recover from failures. The fear of new deployments and potential issues led to stagnant systems and slow improvements for the sake of stability.

As Charity Majors put it, 'Fear of deploys is the ultimate technical debt.'

Since the platform was already using Flux to manage their systems the GitOps way, a natural evolution was to adopt Flagger for progressive delivery.

Flagger is a GitOps-based tool that enables progressive delivery of applications. Flagger can run and manage canary releases, blue-green, A/B testing and other advanced deployments with service meshes for traffic shifting. MediaMarktSaturn were excited to hear about the possibility of controlled Canary releases.

This is a process where a new version of the service (canary) is deployed next to an existing service, and traffic is gradually shifted over, while measuring key performance indicators like HTTP requests success ssrate, requests average duration and pods health. Based on the set thresholds, a canary is either promoted or aborted. If aborted, Flagger automatically reverts to the previous one, giving the teams a chance to fix the issues without affecting end users in production.

Solution

▸ A new deployment process for the existing pipeline

Since the team were already using GitOps, they were able to instantly deploy changes into different environments. For example, when a pull request was created, they would instantly deploy any bug fixes to production, new feature versions to a test environment, and new major versions only to development environments.

Flagger adds the ability to automate canary releases in a GitOps pipeline. Whenever the GitOps operator, in this case Flux, detects a new version, Flagger spins up a new canary release and shifts traffic gradually over from an existing service to the new version while Flagger's load generator simulates customer traffic. Every 5 minutes, 5% of the traffic is moved from the primary to the canary version, which goes up to a 50% traffic split.

At this point, Flagger shadows all deployments, config maps, and secrets, and only proceeds with the deployment if the canary is stable. If the canary is less stable than the primary detected by default or custom performance metrics, Flagger stops the deployment and rolls back to the primary version.

▸ Deployment confidence through observability

Flagger's dashboard allows the team at MediaMarktSaturn to monitor deployments in real time. For additional troubleshooting and faster root cause analysis, the team have added custom dashboards from Flagger data and log extracts using Loki.

Monitoring data is collected and digested with Prometheus and Grafana, and threshold alerts for failed canaries are directly sent to Rocket.Chat. This allows the team to get proactively notified about potential errors instead of manually checking the status of a deployment on a regular basis.

Traffic routing is essential for canary releasing, and Istio is the service mesh of choice by the MediaMarktSaturn team. The team leverages standard Istio metrics such as request duration and HTTP error rates. Beyond this, they also use custom metrics that compare the performance of the canary with the primary version. Custom metrics are used to enforce policies, for example, that the new version must not be slower than 10% compared to the old version.

> "Adding Flagger to our stack was a really simple task since we were already using Helm charts."— **Bernd Stübinger, Backend Engineer**

Results

1. **▸ Confidence to deploy:** The single biggest benefit of Flagger, according to MediaMarktSaturn, is the increased confidence in their deployment results. Stable rollouts at high deployment speed empowers the team to now deploy during high-traffic times without issues.

> "With Flagger, we merge a pull request to deploy a new version, and basically forget about it because we feel so safe and sure about it not breaking anything in production." — **Bernd Stübinger, MediaMarktSaturn**

2. **▸ Almost no downtime:** Before Flagger, it was the norm to experience outages during a deployment. Now, instead of outages, they have automatic rollbacks. They even recall an incident where a corrupt configuration that made it to production did not raise an alert because the release was so quickly rolled back.

3. **▸ Faster disaster recovery:** Even when the inevitable failure happens, they can recover quickly. The team can now spin up an identical full-stack production system within five minutes and recover from disaster.

Results

4

▶ **Faster & more frequent releases:** GitOps with Flagger speeds up deployments even more, as it reduces the amount of manual effort required from developers. This is possible because of Flagger's ability to manage automatic traffic shifting. Doing this the manual way would require a lot of precise calculation, scripting, hacking, and monitoring, and still be difficult to implement.

5

▶ **Deeper & wider observability into deployments:** The team can now directly quantify and qualify impact between the new and the old version of a service. They have dashboards with a side by side comparison of latencies, error rates, and more. All data can be observed in real-time, using their monitoring application of choice, whether that's Prometheus, Grafana, Loki, or Rocket.Chat.

6

▶ **Better collaboration between business and tech teams:** A lesser known benefit of Flagger is that it can help software delivery teams when talking to business stakeholders. At MediaMarktSaturn, there are some stores that have legacy stacks and still do completely manual deployments. Those stores have a fear of deployments. However, when they learn that Flagger enables automated regression tests with every new deployment, it eases their fear. This goes a long way in enabling better collaboration between both the technical and business teams.

> "Right now, our systems are nearly 100% reliable."
> — **Florian Heubeck, MediaMarktSaturn**