Building Modern VFX Infrastructure

Natasha Kelkar
MPC Core Engineering
natasha-k@mpcfilm.com

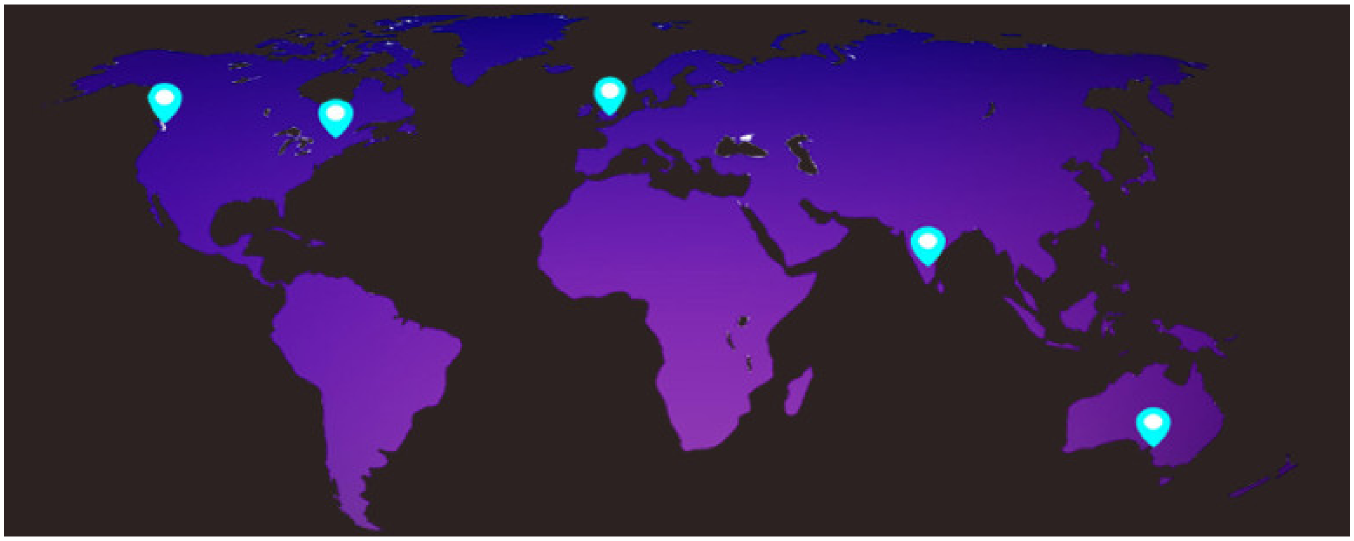

Figure 1: Present and Future sites that would use the VFX platform

ABSTRACT

In order to meet the rapidly evolving needs of the VFX industry, studios need to be able to adapt and upgrade quickly. However, the infrastructure stack at most companies is complex, usually set up over a number of years with significant customizations and proprietary software. Maintaining this stack requires dedicated teams. Upgrades can take months and are usually fraught with risk.

The engineering team at MPC drastically reduced this time to deployment from months to a few days by using cloud native solutions. Built on a foundation of microservices, the infrastructure stack provides an asset management system, storage, sync and compute capabilities. Within the first year it was deployed across two sites in different timezones, supporting up to 200 artists. Thus proving the ability for VFX studios to scale rapidly.

CCS CONCEPTS

• **Computing methodologies → Microservices**.

KEYWORDS

kubernetes, cloud native, microservices

ACM Reference Format:
Natasha Kelkar. 2019. Building Modern VFX Infrastructure. In *Proceedings of SIGGRAPH '19 Talks.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3214745.3214753

1 REQUIREMENTS

To support a growing global studio, the ability to leverage the public cloud for compute, storage and archival had to be considered when designing the platform. The policy was to assetize every artifact from the VFX artist to facilitate multi site collaboration and, when required, to utilise cloud providers to transparently extend render capacity. Automating the end-to-end maintenance and deployment of the platform was important to building scalable infrastructure. It would help bridge the gap between the development and infrastructure teams and ease maintenance of the platform. Also, tracking the metrics was crucial in order to be able to optimize and monitor performance of the infrastructure.

2 DESIGN

Amanda [Eenbergen 2014] provides a strong microservices framework and is the foundation of the new platform. MPC's software build tool, ion enables containerization of VFX software by creating isolated runtime environments. With a microservices framework and a containerised workflow in place, moving to a cloud native pipeline was a natural evolution. This allowed for the use of pre-existing open source technologies, such as Kubernetes [Hightower et al. 2017], to build a reusable, scalable and portable platform.

The design follows a client-server model with the platform acting as the 'server' and workstations as 'clients'.

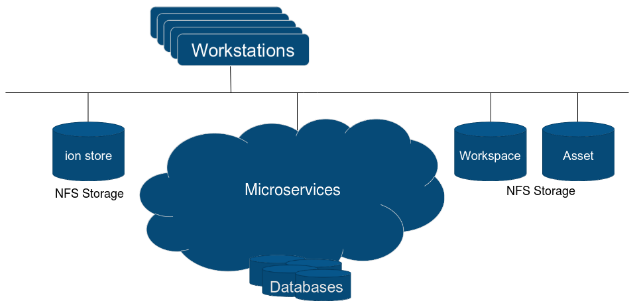

Figure 2: Core Platform infrastructure design

An artist logged into a workstation queries the platform for a show container and is assigned a workspace in the shared storage. At this point the geographic location of the workstation is irrelevant. It could be located on premise at the studio or even in the cloud.

The journey of an asset through the pipeline is recorded via the asset management system,'Tessa'. Tessa uses a unique asset identifier for tracking and automating the movement of assets between artists, insulating them from the complexity of the underlying data.

Tessa also provides a versatile API that can be leveraged to create modular, scalable components that can be replaced independently.

3 IMPLEMENTATION

Knowing that building the VFX infrastructure from the ground up would be a massive undertaking, it was important not to build in isolation and follow an iterative approach. The agile development process was instrumental in keeping the deliverables small and frequent. The first effort focussed on building a development platform that would be identical to a production set up. By running a local Kubernetes cluster using Minikube, the infrastructure team was able to open up the platform to pipeline and service developers without running the risk of breaking production. An increase in the number of early users to the platform helped the infrastructure team get feedback and apply the learnings to the development process.

To build a scalable production cluster, lightweight nodes without a persisted state were required, thereby making it possible to replace any of the nodes at any time. The need to build a cluster across multiple sites meant that the provisioning had to be automated to make the deployment easier, efficient and less prone to human error. A Kubernetes cluster was created comprising of multiple CoreOS nodes. CoreOS [Makam 2016] is a lightweight Linux designed for running containers and provides components necessary to run a distributed system. The nodes boot over HTTPS and configure themselves to form a cluster. By using Ansible [Hall 2013] for automation, the first site was deployed in three weeks. However, after further optimizations, the time to deploy a new site was reduced to 3 days, though the goal is in hours.

VFX show environments are complex comprising of multiple third-party applications, plug-ins and proprietary tools. Managing software upgrades and production environments often require full time support and dedicated teams across all disciplines. ion is used to build fully described runtime environments. It focuses on dependency management to build a containerized VFX software stack. The resulting show containers are managed by services running on the production cluster.

Tessa forms the backbone of the platform. It is implemented as a collection of microservices backed by persistent data storage. It's API and plug-in system allowed pipeline developers to leave some of MPC's legacy tool set behind and transition to a USD based pipeline.

An asset produced in one site is often an input to another asset in a different site. Tessa's power comes from it's ability to establish relationships between assets. A cross-site workflow was established by using these relationships to ensure that valid data was available to the right artist at the correct location. By making multiple service calls to check for availability and transfer, raw files and assets are synced across multiple sites.

Finally an on premise render farm was deployed adding compute capabilities to both sites.

4 LEARNINGS AND FUTURE WORK

Using open source solutions helped in being part of the larger technical community by being able to reach out when needed and contribute where possible. However, this meant that the team had to keep learning and stay up to date on the latest technologies. Mirroring the production set up on the development platform allowed the team to experiment with new techniques with minimal impact on production.

The second iteration will focus on extending the render capabilities to the cloud. With a new site deployment on the schedule, the next phase will see an emphasis on ease of deployment and scalability across multiple sites along with improving consistency and reliability of the platform by building tooling to monitor the cluster. Although still early to determine the success of the platform, the learnings are valuable. Future work will make it easier for the other business units to adopt, making it Technicolor's core VFX platform upon which efficient computer graphics pipelines can be built.

ACKNOWLEDGMENTS

Thanks to David Zuber, Robert Jaworzyn who were part of the core team that built this platform. Special thanks to David Stinson, Jozef Van Eenbergen and William Hall for their invaluable guidance and support.

REFERENCES

Jozef Van Eenbergen. 2014. Amanda: A New Generation of Distributed Services Framework *(EuroPython '14)*. https://archive.org/details/EuroPython_2014_zWSMQ8ep
Daniel Hall. 2013. *Ansible Configuration Management.* Packt Publishing.
Kelsey Hightower, Brendan Burns, and Joe Beda. 2017. *Kubernetes: Up and Running Dive into the Future of Infrastructure* (1st ed.). O'Reilly Media, Inc.
Sreenivas Makam. 2016. *Mastering CoreOS.* Packt Publishing.