Contribution ID: **63** Type: **not specified**

Schrodinger's cloud storage - Defining the box used

Tuesday 31 January 2017 12:20 (20 minutes)

Experiences of containerising a traditional software stack

Synopsis

This presentation will be about the experiences of AARNet of converting what is a traditional monolithic software stack to run inside a fully containerised and dynamically provisioned Docker based container system.

The entire stack, from the front end TLS proxies to the backend scale out storage, the metrics, reporting and orchestration all run inside containers.

History

AARNet has like many other NRENs, deployed a sync and share platform built on both ownCloud and File-Sender, and it has been adopted at scale by our users.

AARNet is in a somewhat unique position amongst NRENs, having a few clusters of dense population on a very large continent with tens of milliseconds between cities. This has meant that AARNet's software infrastructure needs to be spread in order to minimise network latencies between nodes, and this has created its own challenges in orchestrating and managing the environments.

Problem Scope

Growth of users and the type of usages has resulted in a substantial increase in the amount of hardware required to provide a reliable responsive service, and this is multiplied by 3 due to the 3 concurrent sites. The automatic connecting to the nearest node is managed by BGP Anycast, which means a user doesn't live at any one site, but all three simultaenously. Many of the research groups in Australia are geographically distant too, in some cases up to 90ms apart by network paths, yet they all want seamless and consistent performance experiences.

Managing the hardware and software required for this means keeping service state and versioning, to which using containers as a packaging format has become an obvious solution.

Solution

Containerisation through Docker, orchestrated via Rancher, has resulted in a stable and scalable software stack that allows tiered definitions of a software stack and all its constituent components. Ansible and Cobbler to deploy simplest-possible server environments, followed by automatic deployment onto compute resources as they become available has resulted in software upgrade times in the order of seconds with a single click of a mouse button, and instant rollback capability. The actual software is ephemeral and doesn't live permanently on any one server as an assigned task.

Due to the ephemeral nature of the environment, work has been put into ensuring logs and metrics are all centrally collected and visible, with servers themselves treated as entirely disposable.

By being entirely ephemeral, secret management also comes to the fore in the minds of developers, so security of hidden components is increased.

Additionally due to the advanced networking capabilities of Docker and Rancher, scaling out onto third-party clouds is a trivial activity.

Problems

The largest problem with such an infrastructure, strangely is also one of its strengths. The ephemeral nature of the software stacks, and the minimal tooling inside a container means that there can be extra steps or repeated steps to debug issues.

Another problem is advocating and modifying the thinking of developers to understand the benefits and changes in approach they have to adopt.

Summary

Containerising the software hasn't been an entirely smooth process, but this has been related to the software stack deployed and developer education, rather than a fault of the model. The benefits have far outweighed the difficulties, and the view is now that the task has been done there is no intention to revert the management model. Security has been increased, reproducibility and idempotency is much improved, and speed of deploying both updates and new applications has been massively improved.

Author: JERICHO, David (AARNet)

Presenter: JERICHO, David (AARNet)

Session Classification: Technology