SCHRÖDINGER’S CLOUD STORAGE

Defining the Box Used

David Jericho – Solutions Architect, AARNet
HAVING STATE WITHOUT KNOWING THE STATE

- Clouds can be remarkably like the Schrödinger's cat thought experiment
- The cloud is both alive and dead, until the cloud is opened and examined inside
- Approaching the servers and components as cattle or even chickens.
  - No names
  - Are not pets
  - They aren’t individually important
- Not a new idea, merely rehashing what is done elsewhere
AREN’T YOU JUST COMPLICATING THINGS?

• In a word

  NO. The value proposition is simplifying the components for everyone involved.

• Ansible, Salt, Chef, Puppet, Yet-Another-Configuration-Management-System
  • They’re already complicated
  • They’re not for state or service orchestration

• Developers can do the job they’re paid to do
  • Map very neatly to visual diagrams
  • No need for whole stacks to be managed, rather just a single component
CLOUDSTOR IS GEOGRAPHICALLY LARGE...

- Australia is a huge continent
  - 47ms from Sydney to Perth
  - 27ms from Brisbane to Melbourne
  - 90ms from Darwin to Perth

- Bandwidth is not the problem
  - $\text{RTT} \times \text{TCP Window} = \text{Goodput}$
  - Campus networks
  - User equipment choices
...BECAUSE WE’RE SPREAD THIN

- The dark blue areas show 50% of our 24 million people
- 98% of the population live within 150 kilometres of the coast line
- Collaborations occur with groups thousands of kilometres apart
CLOUDSTOR SITES

• 3 major sites
  • Brisbane -> Melbourne is 22ms
  • Melbourne -> Perth is 43ms

• At least two geographically disperse replicas

• Network splits do occur despite redundancy

• At least 24 servers providing metadata, application, storage and additional compute in each site
  • This is excessive for our current needs
  • It is a large number of servers for a small team
• Traditional methods don’t scale
  • Too few staff
  • Vast knowledge required for whole stack
  • Common tools don’t do service management

• Reiterating earlier comments
  • Nothing new here
  • Leveraging existing knowledge of massive scale
  • Easier to test small components
IT’S ALL ABOUT THE SERVICE

• The focus is service
  • Not individual services

• Keeping things small
  • Reducing induced error
  • Keeping state management fast
  • Reducing monolithic blocks

• The industry buzzword is
  • ORCHESTRATION!
• Lots of small moving parts
  • Our current tool is Rancher controlling Docker
    • Kubernetes is a future possibility
• Everything in containers and orchestrated
  • Databases (data lives on permanent volumes)
  • Storage metadata
  • Storage disk servers
  • Web servers
  • Language specific environments
  • Caches
  • Routing daemons...
TOOL VIEW OF A TRAINING ENVIRONMENT
ENVIRONMENT TO SUIT TASK
ENVIRONMENT UPDATE

• Container hosts are minimal
  • Updated frequently
• Using Jenkins presently for CI
  • Dockerfiles are in Git
  • Configuration is in Git
  • Secrets live in a secure store
• Build based on
  • Git branch update
  • Schedule
  • Updates to package repositories
• Orchestrator pins versions

By Jenkins Project, CC, BY-SA 3.0, https://Jenkins.io/
THE UPGRADE BROKE!

- Rolling updates in production
  - With time scheduling
  - Spacing between events
  - Halting on failure
  - Trivial rollback of component state
  - Never having to log into a server
  - Event auditing

- Developers love having the responsibility without the burden
  - And the Get Out of Jail Free card

Michael D'Silva
@davidjericho how awesome is "Upgrade" on rancher!
• Everybody needs to understand the paradigm
  • There are no long lived environments
    • **EVERYTHING IS EPHEMERAL!**
  • There is no SSH into a server
  • There is no quick-fix without an audit trail
    • Quick-fixes can be deployed everywhere very quickly

• Sell on the value
  • And the handling of risk
WITHOUT DISCIPLINE

- This isn’t a solution for everywhere
  - The whole team needs to be in agreement
  - Adding a ssh daemon is punishable

- Take advantage of the tool features
  - Git branches and merges
  - Code review
  - Layering and inheritance
EVERYTHING IS EPHEMERAL

- Design your entire stack to be ephemeral
  - Your bootstrap maybe traditional
- docker-compose is not orchestration
  - It is a great rapid prototyping tool
- Database servers run fine in containers
  - THE DATA SHOULDN’T BE IN THE CONTAINER
  - A great way to lose your job
- Being ephemeral, keeping up with design patterns or fashion is far easier
• Continental VXLAN mesh for all components
• RancherOS or other minimal host operating system
• Complete CI and CD integration for commit, review and deploy
• Releasing containers code onto GitHub