
Going to the
CLOUD!

DISCLAIMER:
This talk is about work in progress. Completeness and accuracy aren't guaranteed beyond best effort

Starting point

- Old hardware
- A lot of profitable legacy software
- Openstack + bare metal
- Working CI/CD
- Working configuration management
- Small infrastructure team
- Software is an essential business component, but our business is not software

Cloud considerations

- Scaling
  - Cloud systems let you scale in smaller increments on demand
- Variability in demand
  - Low variability in demand for computing resources supports staying in-house
  - Highly variable systems benefit from moving to the cloud far more
- Legal issues
  - Privacy regulations in the EU itself
    - Also different laws between different EU countries
  - Brexit
- Software design
  - Observability must be built into the software

Choices

- Already using Docker
- Already moving to microservices
- Moving from Mesos to Kubernetes was easy
- This made Google's Cloud offering a slightly better choice than Amazon
  - Google being cheaper helped a bit
- Neither was cheaper than running our own hardware

The technical research phase

- Lasted about half a year
- Focus on two main areas:
  - How to manage infrastructure manually at the vendor
  - Tooling and automation

Why manual work?

- Familiarisation
- Concepts
- **Discover limitations**

Choosing automation tools

- Shell scripts
  - Via gcloud + gsutil
- Ansible
  - We had Ansible experience
  - Built some systems with ansible
  - Very limited in what it can do without using gcloud
- Puppet
  - Was not a serious contender six months ago
- Terraform
  - The best of the lot

Configuration management

- We still need configuration management for systems which aren't in a container
  - Stateless systems implemented in a 12-factor style are best put in containers and managed via Kubernetes
- Puppet was the obvious choice, because we were already using it

Inventory

- There isn't a nice CMDB out there yet, which can automagically provision VMs in the cloud and provide information to config-mgmt and orchestration tools
  - We currently hack our way around this by using tags and the Google API

Moving into high speed

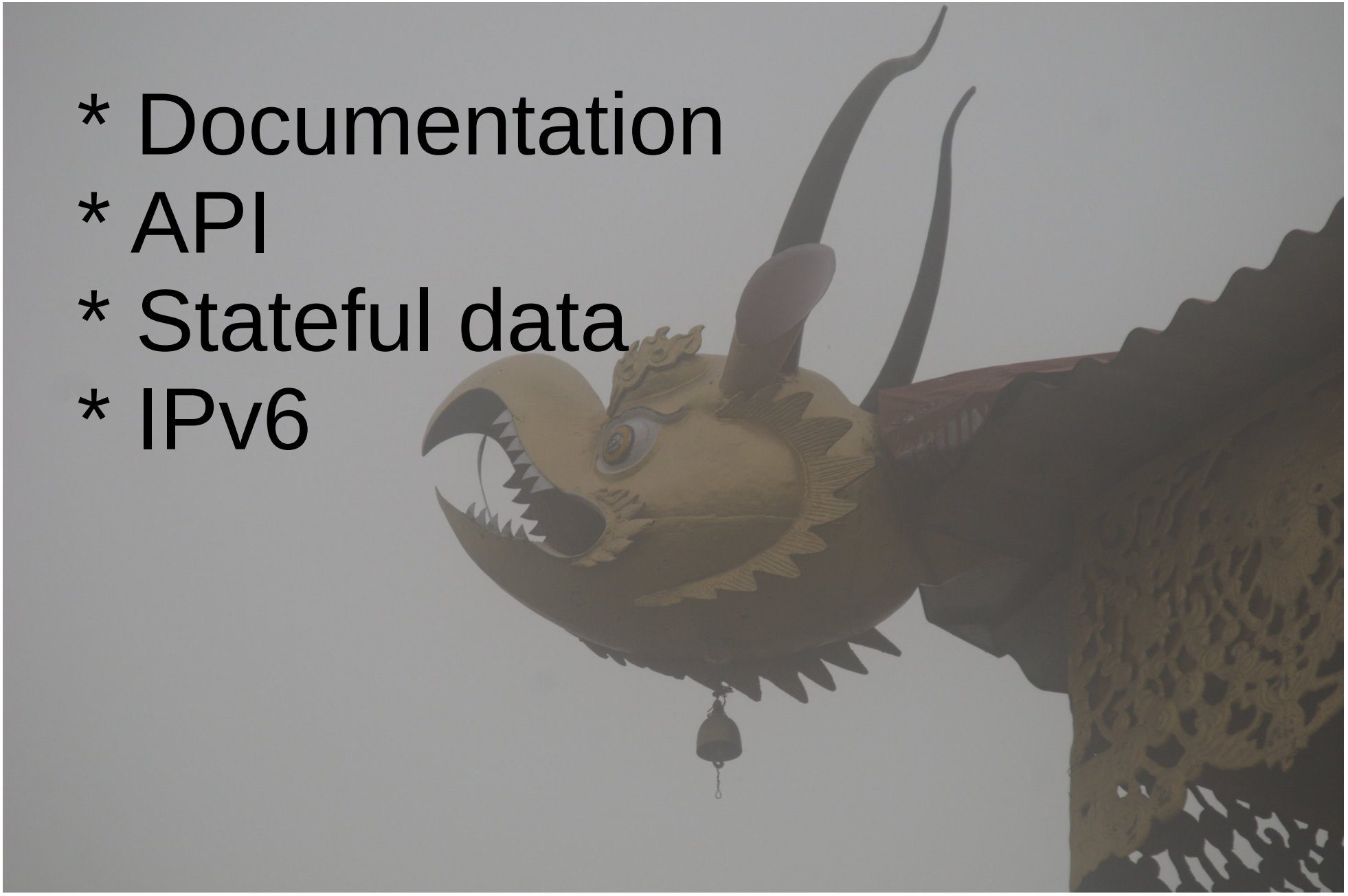
- One meeting
  - Three people
  - Thirty minutes
- Decided on goals for a proof of concept
  - Complete automation
  - Custom tooling around the application
  - Fixed target application for a test deployment
- Took us about three months of full time effort to wrap up the PoC

Tools of choice

- Terraform
  - This is a pretty fast moving tool
  - They have good documentation
    - For some value of good.
- Puppet
  - New Puppet repo, ignoring a lot of legacy.
  - Jumped Puppet version

Terraform

- Base network project, all network related things are done in this project
- Other projects use an instance group with a mostly standard template
  - They reference network configs from the base project
- Google metadata is used to tie together Puppet and Terraform


* Documentation
* API
* Stateful data
* IPv6

Google Cloud Documentation

- Lags behind software
- Is often inconsistent
- This has not changed in about three years

API

- Quite inconsistent in some regards
  - Particularly about referencing other properties
  - Name or reference?
- Needs actual examples
  - A lot of examples

Stateful data

- There are no good answers
- Google offers multiple options for storage
- Some of these are more reliable than others
- Maintenance can cause outages, but automatic failover for CloudSQL needs a whole zone to fail

IPv6

- Google does not put it's money where it's mouth is wrt IPv6
  - IPv6 support is very limited in the compute environment
- We started off by routing IPv6 traffic to our loadbalancers in the legacy environment and then proxying to IPv4 in Google

Monitoring

- Stackdriver looks promising for log management
  - It has quite a few retention limitations
  - New pricing makes it cheaper to run an ELK stack
- There isn't a really good alternative to running your own time-series database
  - Especially if you use that data for alerting
- Stackdriver is a good replacement for the ELK stack, but not for high quality analytics/monitoring

Legacy code

- Plan on migrating it wholesale
  - Even if you plan to rewrite it
    - Rewrites will take longer than you plan for
- This does not benefit from moving to the cloud
- Database migrations are “interesting”

Spectre/Meltdown impact

- CPU utilisation doubles
  - We are currently on rather over-provisioned hardware, so actual impact is minimal
- Anything which does a lot of system calls is slowed quite a bit
  - Large data import went from 26 hours to 56

Summary

- Cloud migration is a business decision, but remember that costs will probably increase
- Outsourcing your L1 operations team to people who do not care about your business needs still has the same problems as a decade or two ago
- Choosing which provider to go with often involves small differences based on your existing stack
- The tooling available is still very raw, and we are still discovering operational design patterns
- Migrating to the cloud may require a wholesale change in process