
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

Starting point

- Old hardware
- A lot of profitable legacy software

Starting point

- Old hardware
- A lot of profitable legacy software
- Openstack + bare metal

Starting point

- Old hardware
- A lot of profitable legacy software
- Openstack + bare metal
- Working CI/CD

Starting point

- Old hardware
- A lot of profitable legacy software
- Openstack + bare metal
- Working CI/CD
- Working configuration management

Starting point

- Old hardware
- A lot of profitable legacy software
- Openstack + bare metal
- Working CI/CD
- Working configuration management
- Small infrastructure team

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

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
- Developers are on call for production application issues

Cloud considerations

- Scaling
  - Cloud systems let you scale in smaller increments on demand

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

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

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

Vendor Choices

Vendor Choices

- Already using Docker

Vendor Choices

- Already using Docker
- Already moving to microservices

Vendor Choices

- Already using Docker
- Already moving to microservices
- Moving from Mesos to Kubernetes was easy

Vendor 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

Vendor 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
  - Savings mostly come from the lack of a dedicated operations group, and from being able to avoid some HA requirements

The technical research phase

- Lasted about half a year

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
  - Terminology

Why manual work?

- Familiarisation
  - Terminology
- Concepts

Why manual work?

- Familiarisation
  - Terminology
- Concepts
- **Discover limitations**
  - There are a lot of those
  - Some more interesting than others (load balancing, IPv6, DNS, ...)

Choosing automation tools

- Shell scripts
  - Via gcloud + gsutil

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

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

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
    - It has improved a lot since this slideset was first made

Configuration management

- Stateless systems implemented in a 12-factor style are best put in containers and managed via Kubernetes
  - Alternatively, use what Google calls managed groups and spin up VMs automatically in case of crashes

Configuration management

- Stateless systems implemented in a 12-factor style are best put in containers and managed via Kubernetes
  - Alternatively, use what Google calls managed groups and spin up VMs automatically in case of crashes
- We still need configuration management for systems which aren't in a container

Configuration management

- Stateless systems implemented in a 12-factor style are best put in containers and managed via Kubernetes
  - Alternatively, use what Google calls managed groups and spin up VMs automatically in case of crashes
- We still need configuration management for systems which aren't in a container
- Puppet was the obvious choice, because we were already using it
  - It doesn’t matter which specific tool you use, but use one.

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

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

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.
  - Getting your first bits and pieces working are harder than they should be, but the rest then follow pretty easily

Tools of choice

- Terraform
  - This is a pretty fast moving tool
  - They have good documentation
    - For some value of good.
  - Getting your first bits and pieces working are harder than they should be, but the rest then follow pretty easily
- Puppet
  - New Puppet repo, ignoring a lot of legacy.
  - Jumped Puppet version
  - Discarded large parts of the module approach recommended in Puppet documentation

Terraform

- Base network project
  - All network related things are done in this project

Terraform

- Base network project
  - All network related things are done in this project
- Other projects use instance groups with a mostly standard template
  - They reference network configs from the base project

Terraform

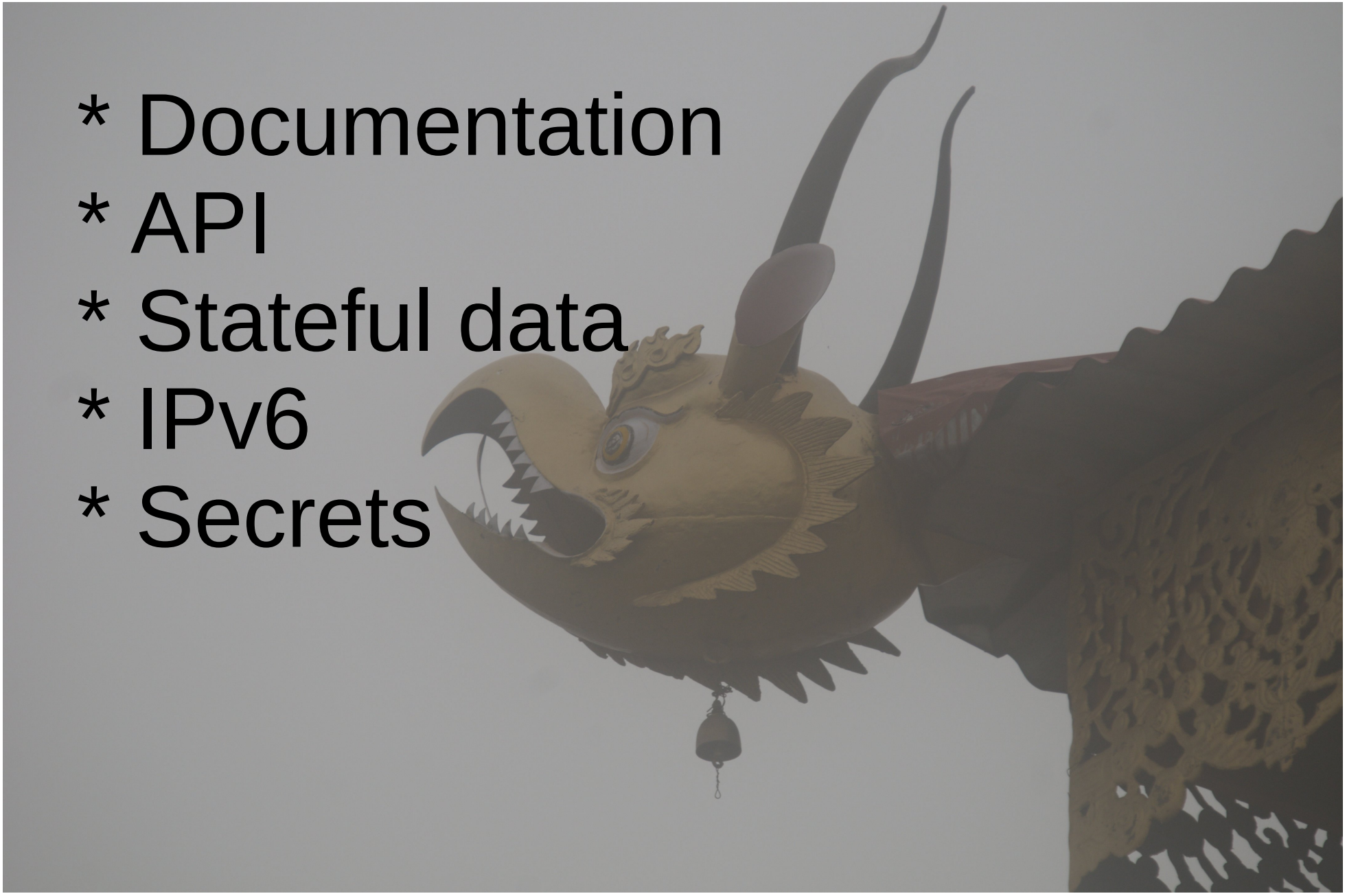
- Base network project
  - All network related things are done in this project
- Other projects use an instance group with a mostly standard template
  - They reference network configs from the base project
- Google metadata is used to tie together Puppet and Terraform

Shared backends

- We started with a simple backend for Terraform, with no remote state.
  - This does not scale to many users, but for the initial proof of concept was useful.

Shared backends

- We started with a simple backend for Terraform, with no remote state.
  - This does not scale to many users, but for the initial proof of concept was useful.
- We then spent a few days very carefully refactoring this into per project state, with the shared state being remote in a cloud storage bucket.
  - https://charity.wtf/2016/03/30/terraform-vpc-and-why-you-want-a-tfstate-file-per-env/ is a pretty good horror story of what could go wrong


* Documentation
* API
* Stateful data
* IPv6
* Secrets

Google Cloud Documentation

- Lags behind software

Google Cloud Documentation

- Lags behind software
- Is often inconsistent

Google Cloud Documentation

- Lags behind software
- Is often inconsistent
- This has not changed in about three years
  - This is not limited to Google though.

API

- Quite inconsistent in some regards
  - Particularly about referencing other properties
  - Name or reference?

API

- Quite inconsistent in some regards
  - Particularly about referencing other properties
  - Name or reference?
- Needs actual examples
  - A lot of examples
    - This has not really improved since I first wrote this talk

Stateful data

- There are no good answers for high availability

Stateful data

- There are no good answers for high availability
- Google offers multiple options for storage
  - Some of these are more reliable than others
  - But they are more complex to use
  - Or involve code changes

Stateful data

- There are no good answers for high availability
- Google offers multiple options for storage
  - Some of these are more reliable than others
  - But they are more complex to use
  - Or involve code changes
- Maintenance can cause outages
  - automatic failover for CloudSQL needs a whole zone to fail, so a maintenance can cause an unexpected outage

Stateful data

- There are no good answers for high availability
- Google offers multiple options for storage
  - Some of these are more reliable than others
  - But they are more complex to use
  - Or involve code changes
- Maintenance can cause outages
  - automatic failover for CloudSQL needs a whole zone to fail, so a maintenance can cause an unexpected outage
- You may need to run your own database systems for more reliable access to structured data

IPv6

- Google does not put it's money where it's mouth is wrt IPv6
  - IPv6 support is very limited in the compute environment

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
  - This is no longer needed

Secrets

- If you have containers, Google supports encrypted secrets.

Secrets

- If you have containers, Google supports encrypted secrets.
- Using Vault from Hashicorp looks like a good option, but you still need to code applications to use those secrets instead of reading from a config file

Secrets

- If you have containers, Google supports encrypted secrets.
- Using Vault from Hashicorp looks like a good option, but you still need to code applications to use those secrets instead of reading from a config file
- Anything else which works with your configuration management system is a good idea (eyaml with Puppet, for example)
  - You still have the problem of managing a few master encryption keys

Secrets

- If you have containers, Google supports encrypted secrets.
- Using Vault from Hashicorp looks like a good option, but you still need to code applications to use those secrets instead of reading from a config file
- Anything else which works with your configuration management system is a good idea (eyaml with Puppet, for example)
  - You still have the problem of managing a few master encryption keys
- We tested hiera-vault, but performance was terrible

Loadbalancing

- Google’s load balancer offering is limited in some ways as compared to more advanced tools like F5s, etc
- We chose to replace the hardware LBs with simple IP based load balancer + nginx proxies
  - Note that code which tracks IP addresses or does geolocation needs to change to handle this.

Monitoring

- Stackdriver looks promising for log management
  - It has quite a few retention limitations
  - New pricing makes it cheaper to run an ELK stack, depending on log volume

Monitoring

- Stackdriver looks promising for log management
  - It has quite a few retention limitations
  - New pricing makes it cheaper to run an ELK stack, depending on log volume
- Stackdriver is a good replacement for the ELK stack, but not for high quality analytics/monitoring

Monitoring

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

Legacy code

- Plan on migrating it wholesale
  - Even if you plan to rewrite it
    - Rewrites will take longer than you plan for
  - Even your planned migrations will take longer than expected, because of environmental assumptions.

Legacy code

- Plan on migrating it wholesale
  - Even if you plan to rewrite it
    - Rewrites will take longer than you plan for
  - Even your planned migrations will take longer than expected, because of environmental assumptions.
- This does not benefit from moving to the cloud
  - You are just running it in an environment with different assumptions on latency and reliability

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
  - Monitor your costs closely, you will discover a number of ways in which money is wasted in the cloud (debug logging, for example).

Summary

- Cloud migration is a business decision, but remember that costs will probably increase
  - Monitor your costs closely, you will discover a number of ways in which money is wasted in the cloud (debug logging, for example).
- 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

Summary

- Cloud migration is a business decision, but remember that costs will probably increase
  - Monitor your costs closely, you will discover a number of ways in which money is wasted in the cloud (debug logging, for example).
- 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

Summary

- Cloud migration is a business decision, but remember that costs will probably increase
  - Monitor your costs closely, you will discover a number of ways in which money is wasted in the cloud (debug logging, for example).
- 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

Summary

- Cloud migration is a business decision, but remember that costs will probably increase
  - Monitor your costs closely, you will discover a number of ways in which money is wasted in the cloud (debug logging, for example).
- 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
  - If you are in a large ITIL shop, that will require a huge change.