OPTIMISNG ARCHITECTURE FOR OPERATIONAL CONCERNS

PRESENTED BY:**RYAN FRENCH**
LEAD CLOUD PLATFORM ENGINEER
11TH MARCH 2020

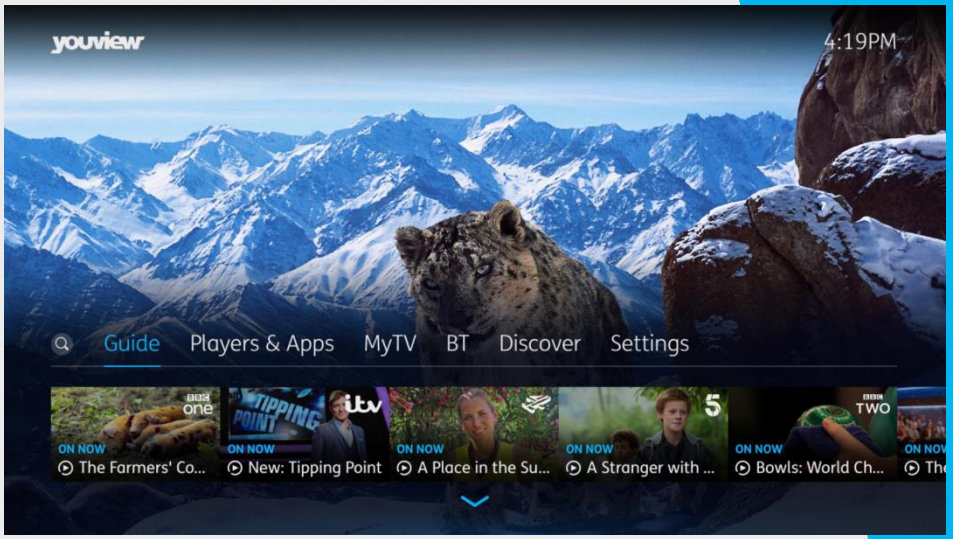
ABOUT YOUVIEW

- Joint venture between some of the UK's leading media and telecoms businesses
- Content discovery platform with content from some of the biggest names in media

ABOUT YOUVIEW

- Platform behind BT TV, TalkTalk and Sony Android TVs for millions of homes across the UK
- Cloud first company
- Third party integration

NEXT GENERATION USER EXPERIENCE



- In early 2015, we began a redesign of our platform, moving logic out of the box and in to the cloud
- In November 2016, YouView Launched our award winning re-engineered and redesigned platform
- As part of the redesign, we moved out of a physical datacentre and moved into AWS
- We also changed how we worked, moving in to small, cross-functional, agile teams

MY ROLE

- Lead Cloud Platform engineer with >10 years experience
- Joined YouView July 2015
- Designed and built cloud services for Next Gen platform
- Early 2017 became Tech Lead of Cloud Platform team

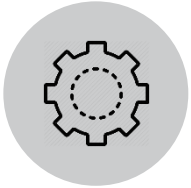
OUR PLATFORM

- Millions of Active Connected Devices
- 6B requests per day to edge
- 300M requests per day to origin
- 1.1M max TPM at origin
- 200k average TPM at origin
- 600+ servers
- 100+ micro services
- 40 cloud service developers
- 50+ AWS services
- 6 programming languages

youview

WHAT ARE OPERATIONAL CONCERNS?

RELIABILITY



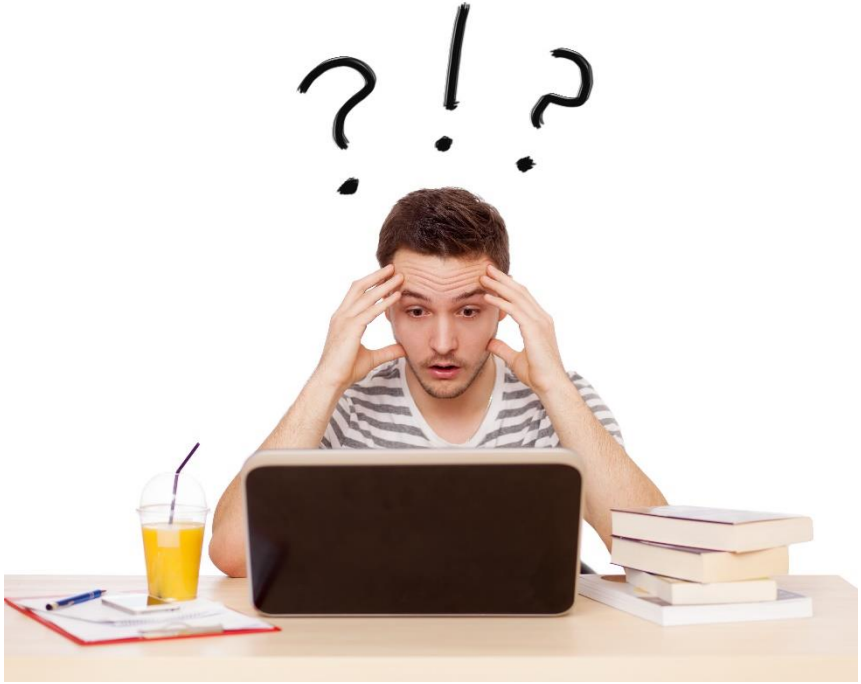
MAINTENANCE

OBSERVABILITY

COST

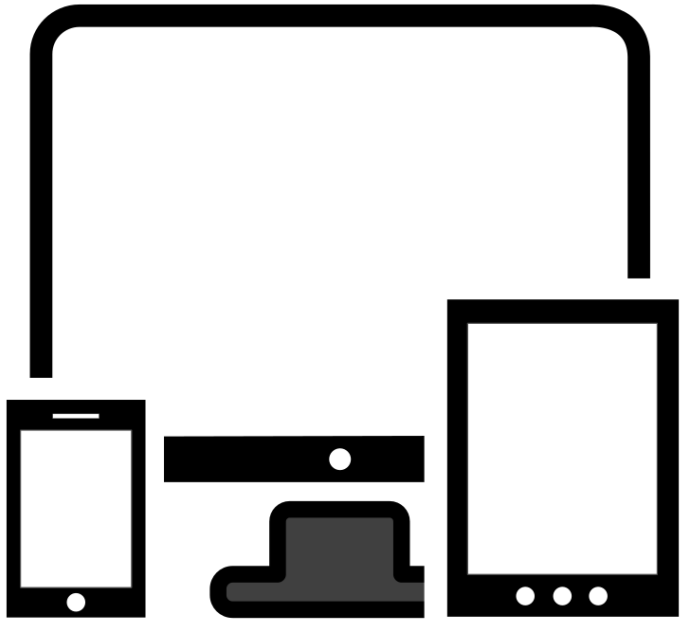
SECURITY

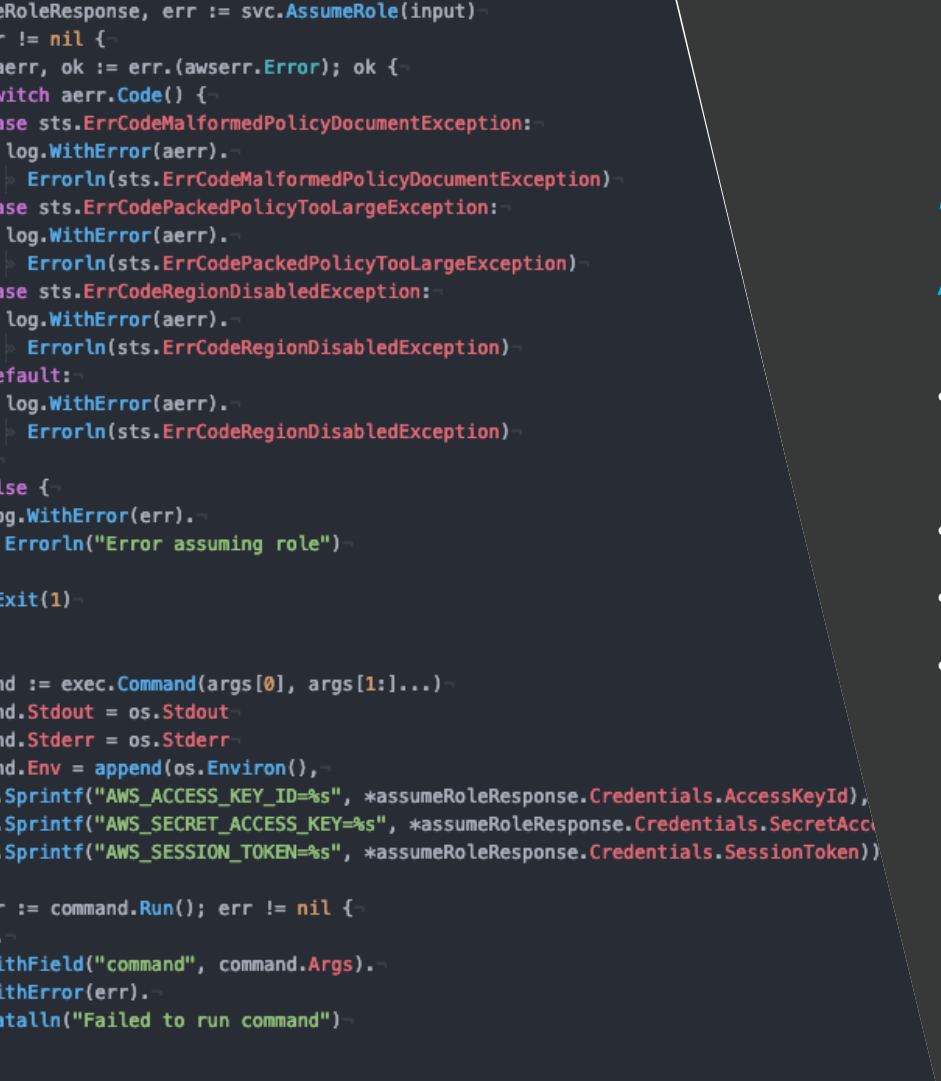
SO WHERE DO YOU START?



THE CLIENT

- **Communication**
  - Request/Response
  - Eventually Consistent
  - Push Messaging
  - Avoid synchronized calls
  - HTTP/2
- **Error handling**
  - Retry logic with exponential backoff
  - Randomisation with Jitter
- **Page Composition**
  - Separate static assets from personailised data
- **Pre-emptive painting**
  - Disconnect UI feedback from the backend





youview

MICROSERVICE ARCHITECTURE

- Define service boundaries – it's better to go too big than too small
- Strong contract testing is a must
- Lines of Code **DO NOT MATTER**
- Loosely couple with pub/sub and replicated data

youview

SIGNS YOU GOT YOUR BOUNDARIES WRONG

- Your service has a runtime dependency on another service
- Your service has a deployment-time dependency on another service
- You cannot contract test your service in isolation
- If your service goes down, another service will fail
- Adding or fixing existing functionality requires modifying multiple repos
- If your service reads another services data store

youview

INTER-SERVICE COMMUNICATION

- Don’t daisy-chain or aggregate requests whenever it can be avoided
  - Every hop in the network is a chance to fail
  - If any request in a daisy-chain or aggregation fails, it’s a failed request
- Adopt an eventually consistent model
- Data stream technologies, like Kafka, or Pub/Sub technologies, like AMQP, allow for easy decoupling of services
- If your service requires data from another service to work, it should have it’s own local copy
- If you simply can’t duplicate your data, or avoid a runtime dependency, then return what data you can, and have your client retrieve the rest of the data

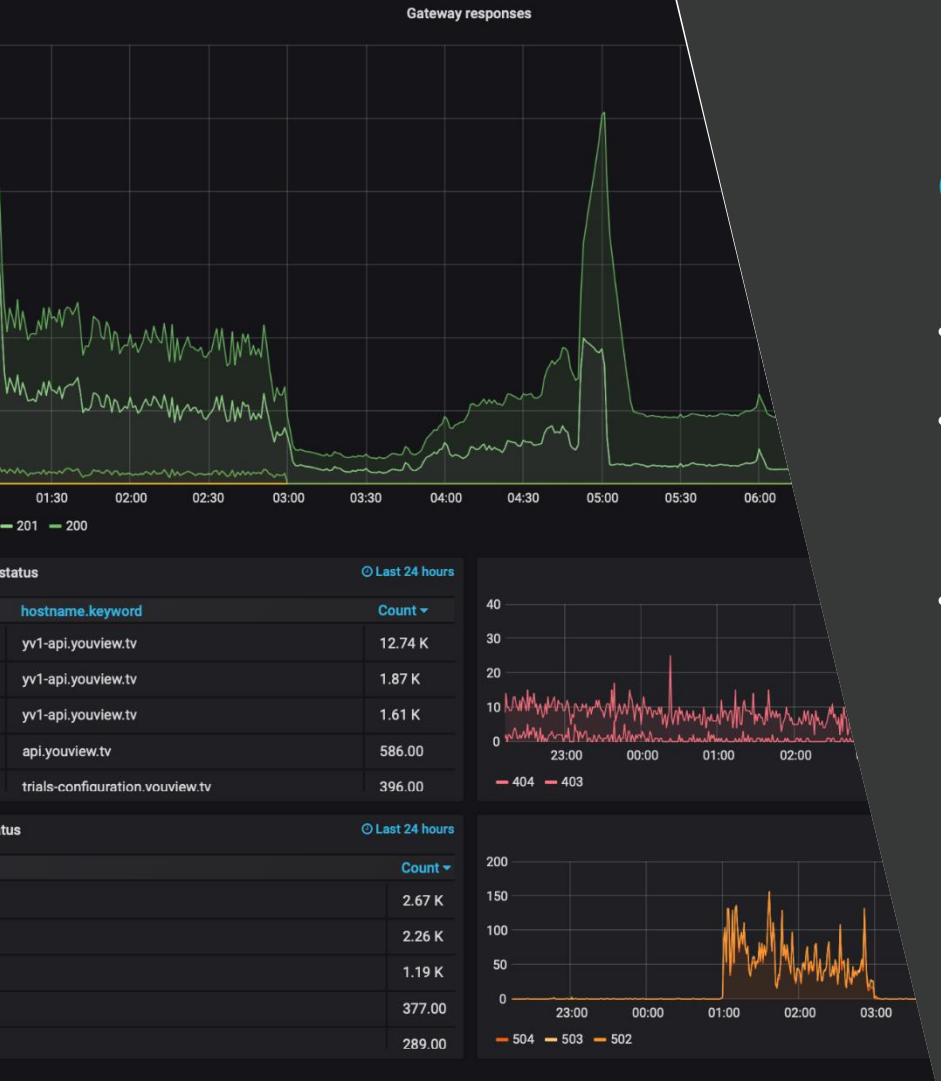
MICROSERVICES – NOT A SILVER BULLET

The Problems

- Duplicate code
  - Boilerplate code
  - Infrastructure as Code
  - Deployment pipelines
- Inconsistencies in implementation
- Large maintenance overhead
- Shared bugs

The Solutions

- Shared libraries
- Generate new services from templates
- Remote templates for infrastructure as code
- Centralised Ownership
- Automated CI/CD

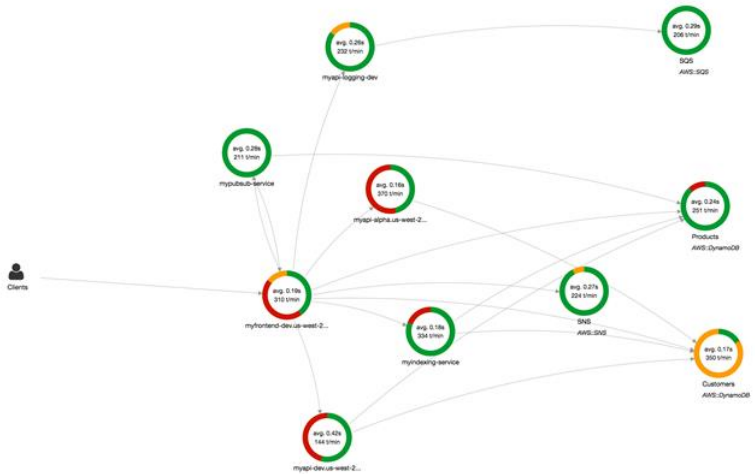


youview

OBSERVABILITY

- Monitor everything
- Expect to spend 10-20% of your operating budget on monitoring
- Differentiate between what should be real time, and what can be delayed

DISTRIBUTED TRACING

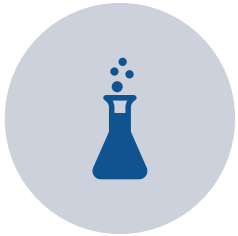


- Improves observability
- Easily find bottlenecks and broken services
- A service map can be a good indicator of accidental coupling

youview

COST

DATABASES



DEVELOPER EXPERIMENTS

LARGE TRAFFIC SPIKES

OVER-PROVISIONED SERVERS


youview
THANK YOU