

A background image showing the silhouettes of a crowd of people standing on a beach or open field, looking towards the horizon. The sun is low in the sky, creating a bright orange and yellow glow with lens flare effects. The people are mostly seen from the back or side, with some wearing hats and casual clothing. The overall mood is serene and communal.

# youview

## OPTIMISING 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

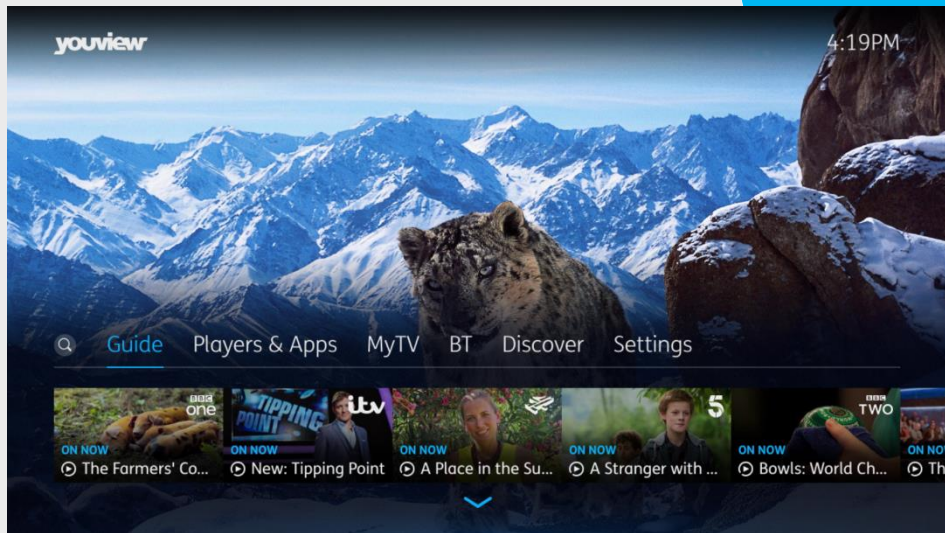


# ABOUT YOUVIEW

- 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





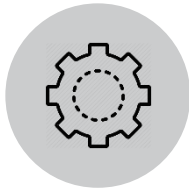
# OUR PLATFORM

- 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

# WHAT ARE OPERATIONAL CONCERNS?



RELIABILITY



MAINTENANCE



OBSERVABILITY

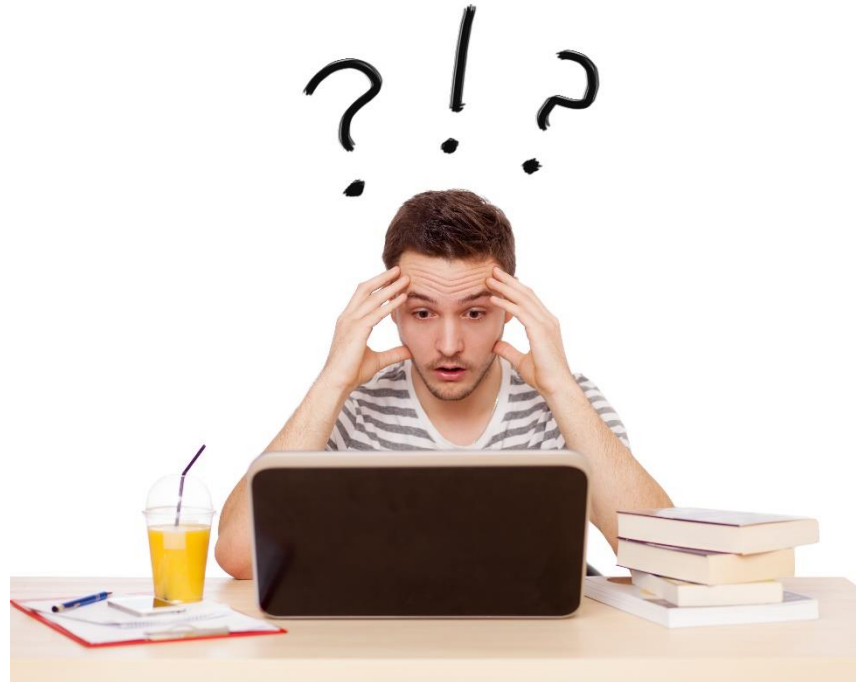


COST



SECURITY

# SO WHERE DO YOU START?





# THE CLIENT

- **Communication**

- Request/Response
- Eventually Consistent
- Push Messaging
- Avoid synchronized calls
- HTTP/2

- **Error handling**

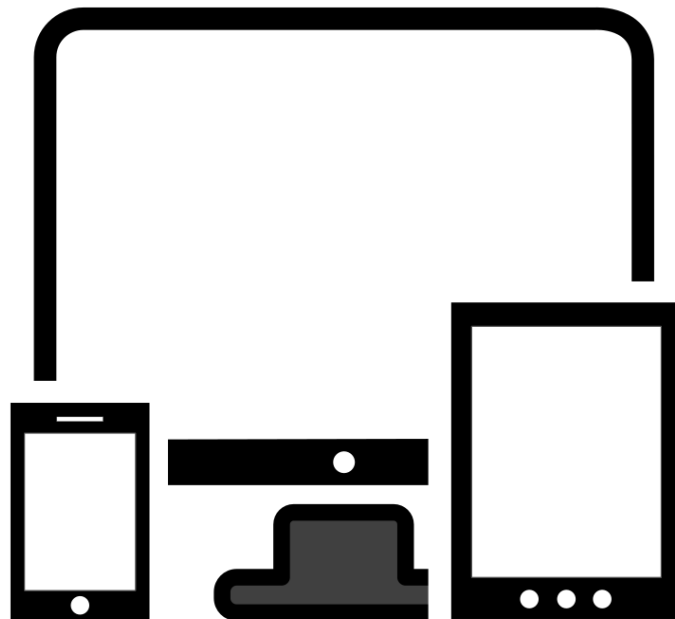
- Retry logic with exponential backoff
- Randomisation with Jitter

- **Page Composition**

- Separate static assets from personalised data

- **Pre-emptive painting**

- Disconnect UI feedback from the backend



# 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

```

assumeRoleResponse, err := svc.AssumeRole(input)
if err != nil {
    // Handle error
    aerr, ok := err.(awserr.Error); ok {
        // Handle specific AWS error
        switch aerr.Code() {
            case sts.ErrCodeMalformedPolicyDocumentException:
                log.WithError(aerr).
                    Errorln(sts.ErrCodeMalformedPolicyDocumentException)
            case sts.ErrCodePackedPolicyToolargeException:
                log.WithError(aerr).
                    Errorln(sts.ErrCodePackedPolicyToolargeException)
            case sts.ErrCodeRegionDisabledException:
                log.WithError(aerr).
                    Errorln(sts.ErrCodeRegionDisabledException)
            default:
                log.WithError(aerr).
                    Errorln(sts.ErrCodeRegionDisabledException)
        }
    }
    // Handle generic error
    log.WithError(err).
        Errorln("Error assuming role")
    Exit(1)
}

cmd := exec.Command(args[0], args[1:]...)
cmd.Stdout = os.Stdout
cmd.Stderr = os.Stderr
cmd.Env = append(os.Environ(),
    Sprintf("AWS_ACCESS_KEY_ID=%s", *assumeRoleResponse.Credentials.AccessKeyId),
    Sprintf("AWS_SECRET_ACCESS_KEY=%s", *assumeRoleResponse.Credentials.SecretAccessKey),
    Sprintf("AWS_SESSION_TOKEN=%s", *assumeRoleResponse.Credentials.SessionToken))

err := cmd.Run(); err != nil {
    // Handle error
    log.WithField("command", cmd.Args).
        WithError(err).
        Fatalln("Failed to run command")
}

```

# 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

# 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

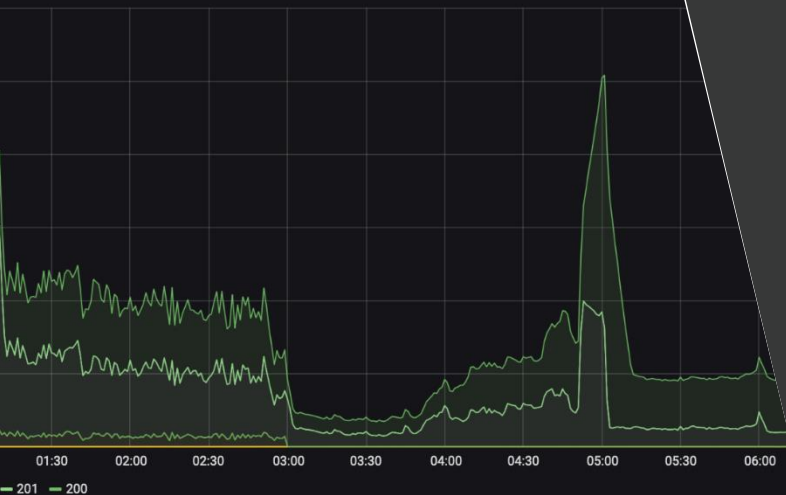
## The Solutions

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



# 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



status ○ Last 24 hours

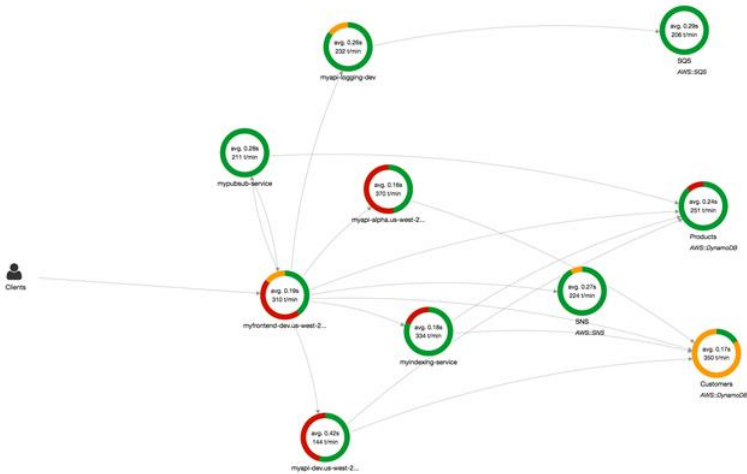
hostname.keyword	Count ▾
yv1-api.youview.tv	12.74 K
yv1-api.youview.tv	1.87 K
yv1-api.youview.tv	1.61 K
api.youview.tv	586.00
trials-configuration.youview.tv	396.00

status ○ Last 24 hours

Count ▾
2.67 K
2.26 K
1.19 K
377.00
289.00



# DISTRIBUTED TRACING



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

# COST



DATABASES



DEVELOPER  
EXPERIMENTS



LARGE TRAFFIC  
SPIKES



OVER-PROVISIONED  
SERVERS

A man wearing a dark suit, a white shirt, a red tie, and a black fedora is running across a rooftop. He is wearing white sneakers and has his arms outstretched. The background shows a city skyline at sunset, with the sun low on the horizon and buildings silhouetted against the orange and yellow sky. The rooftop has a concrete wall and some yellow paint markings.

# youview

THANK YOU