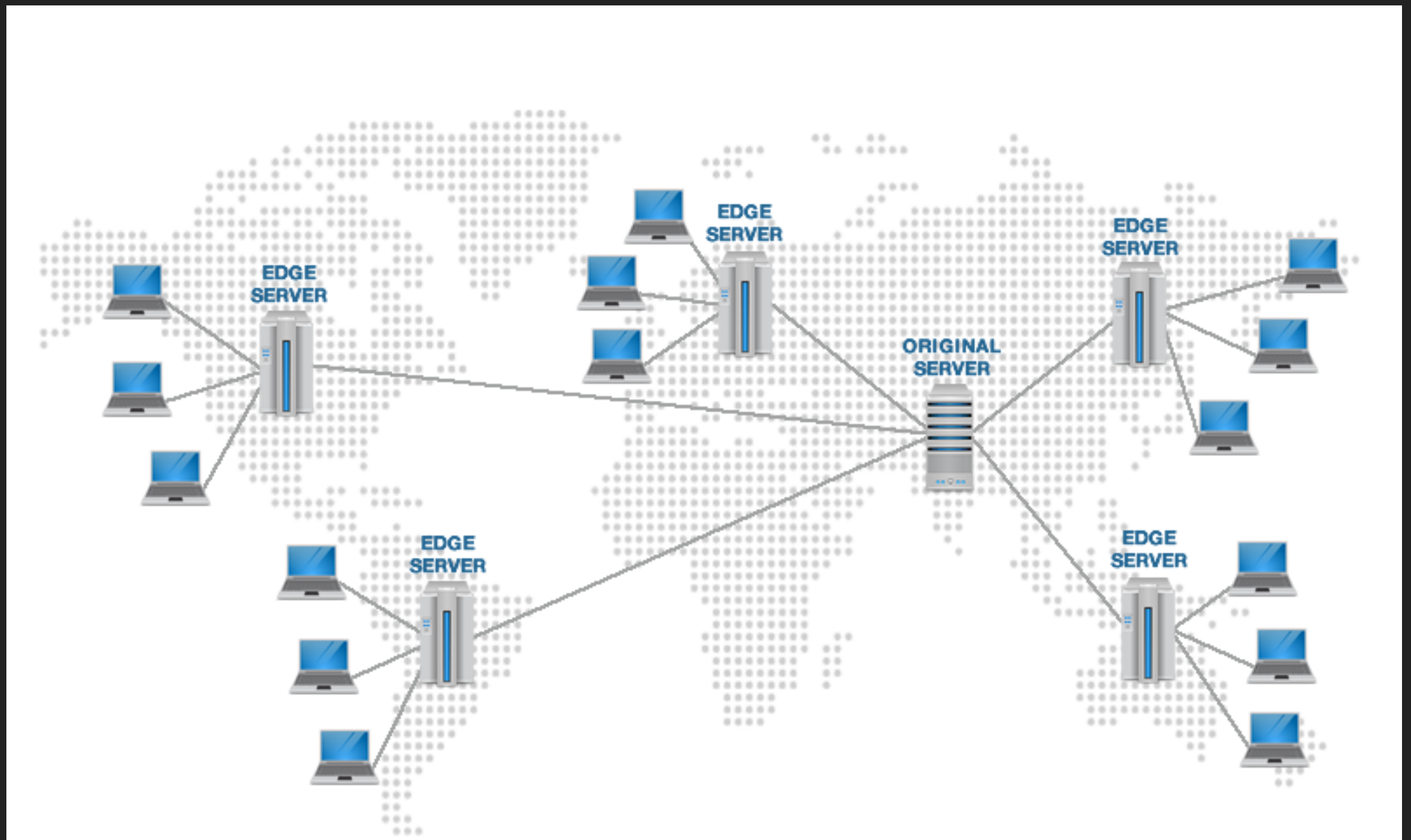
AN ENGINEERING PERSPECTIVE

BUILDING A CDN

HELLO

- murali
- cambridge
- dexecure.com

WHAT IS A CDN


EDGE SERVER
EDGE SERVER
EDGE SERVER
ORIGINAL SERVER
EDGE SERVER
EDGE SERVER

VALUE ADD

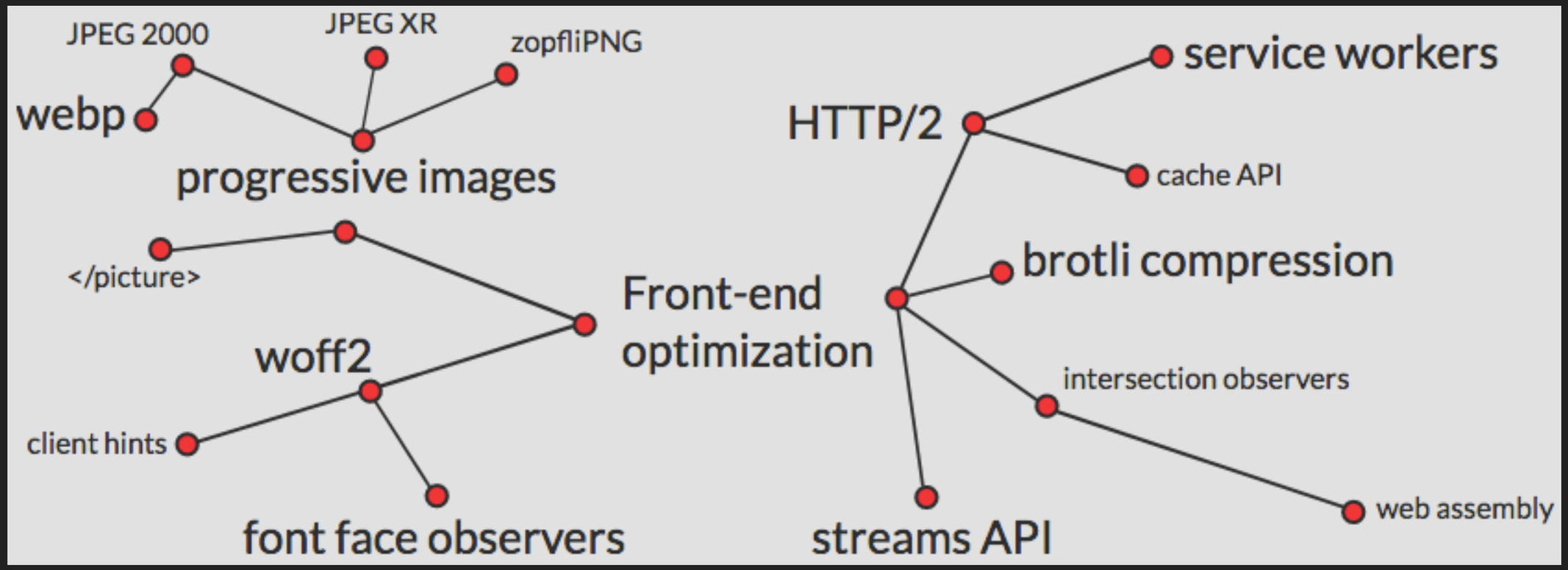
Image: Fire breathing "Jaipur Maharaja Brass Band" Chassepierre Belgium [5]

JPEG file size: 120.78 KB

WebP file size: 80.76 KB

- http://www.gstatic.com/webp/gallery/5.jpg
- http://www.gstatic.com/webp/gallery/5.webp

VALUE ADD

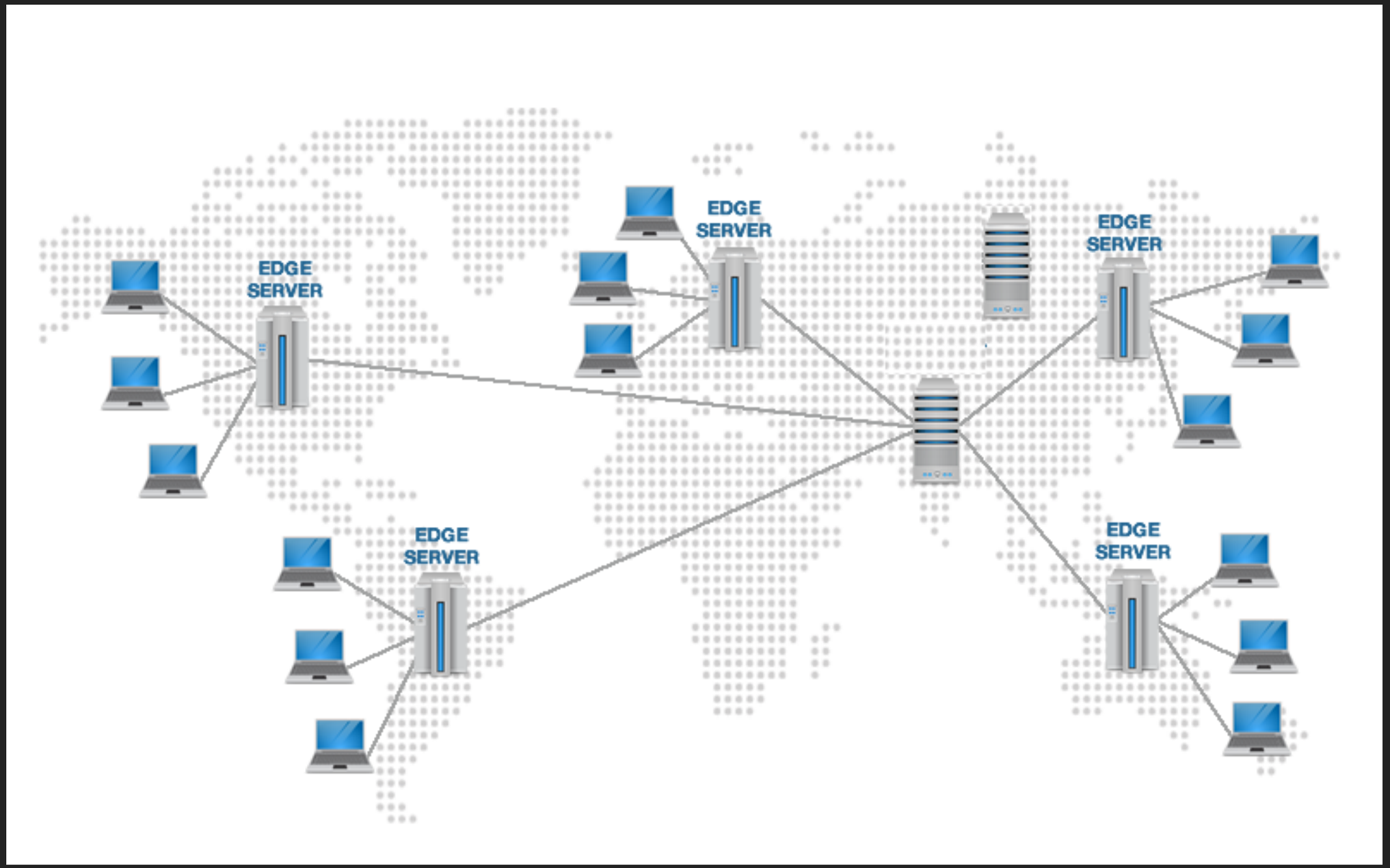


simply too much to bother with

THE DEADLINE IS SET

- A good salesperson does not need a product
- A desperate engineer does not need literature review
- Let’s make it up as we go

BASIC IDEA


EDGE SERVER
EDGE SERVER
EDGE SERVER
EDGE SERVER
EDGE SERVER

SIMPLE BEGINNINGS

```
const optimiseImageForChrome = (imageBuffer) => {
    return imageBuffer.toWebp();
}
```

```
const optimiseImageForOthers = (imageBuffer) => {
    return imageBuffer.toMozJPEG();
}

const requestHandler = (request, response) => {
  wget(request.url)
  .then(imageBuffer => {
    response.end(request.browser == 'chrome' ?
        optimiseImageForChrome(imageBuffer) : optimiseImageForOthers(imageBuffer));
  })
}
```

DISASTER STRIKES!

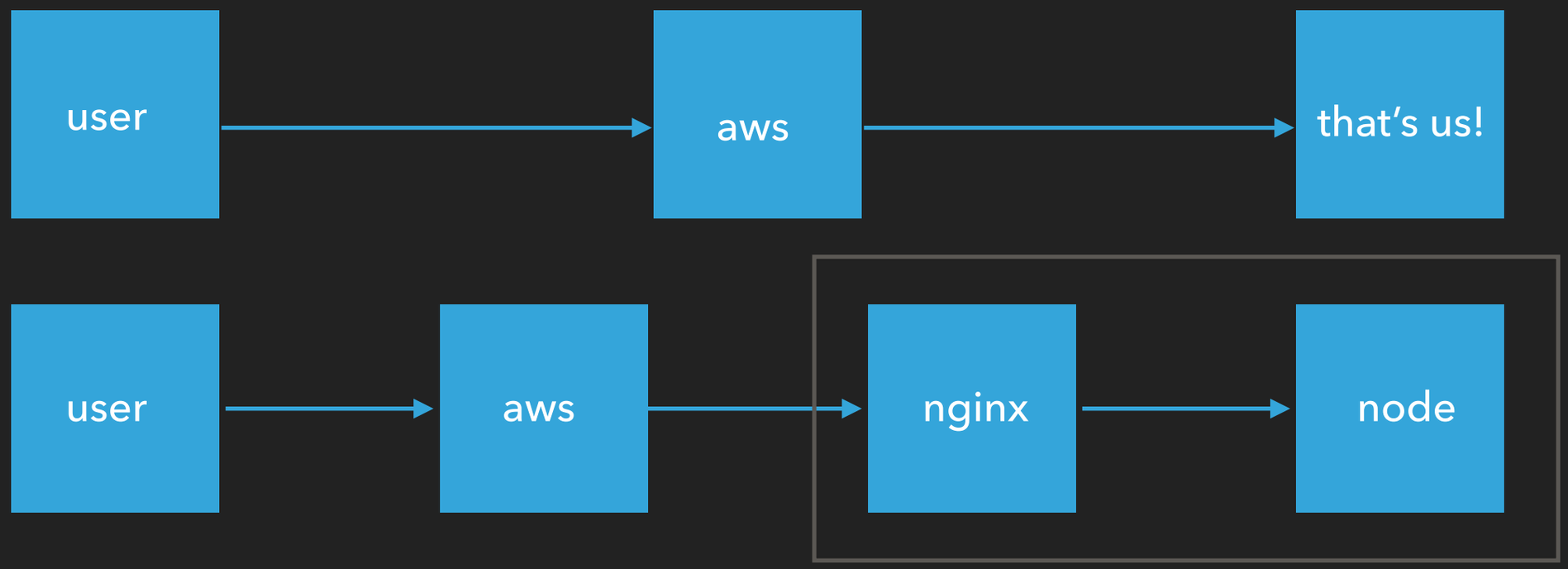
- Client deploys onto production
- Site crashed in 7 seconds
- Made British Airways IT guy proud

THINK ON YOUR FEET

- Complicated and intricate solution
- Spin up 10 load balanced machines
- Crisis averted! **JUST KIDDING, LOL**
- Lose 2$ for every 1$ made
- Uber stole my business model!

ET TU, AWS?

- Grow 100x.
- Machines are not enough. Very poor response times.
- Observe a lot of “cache eviction”



NGINX CONFIG

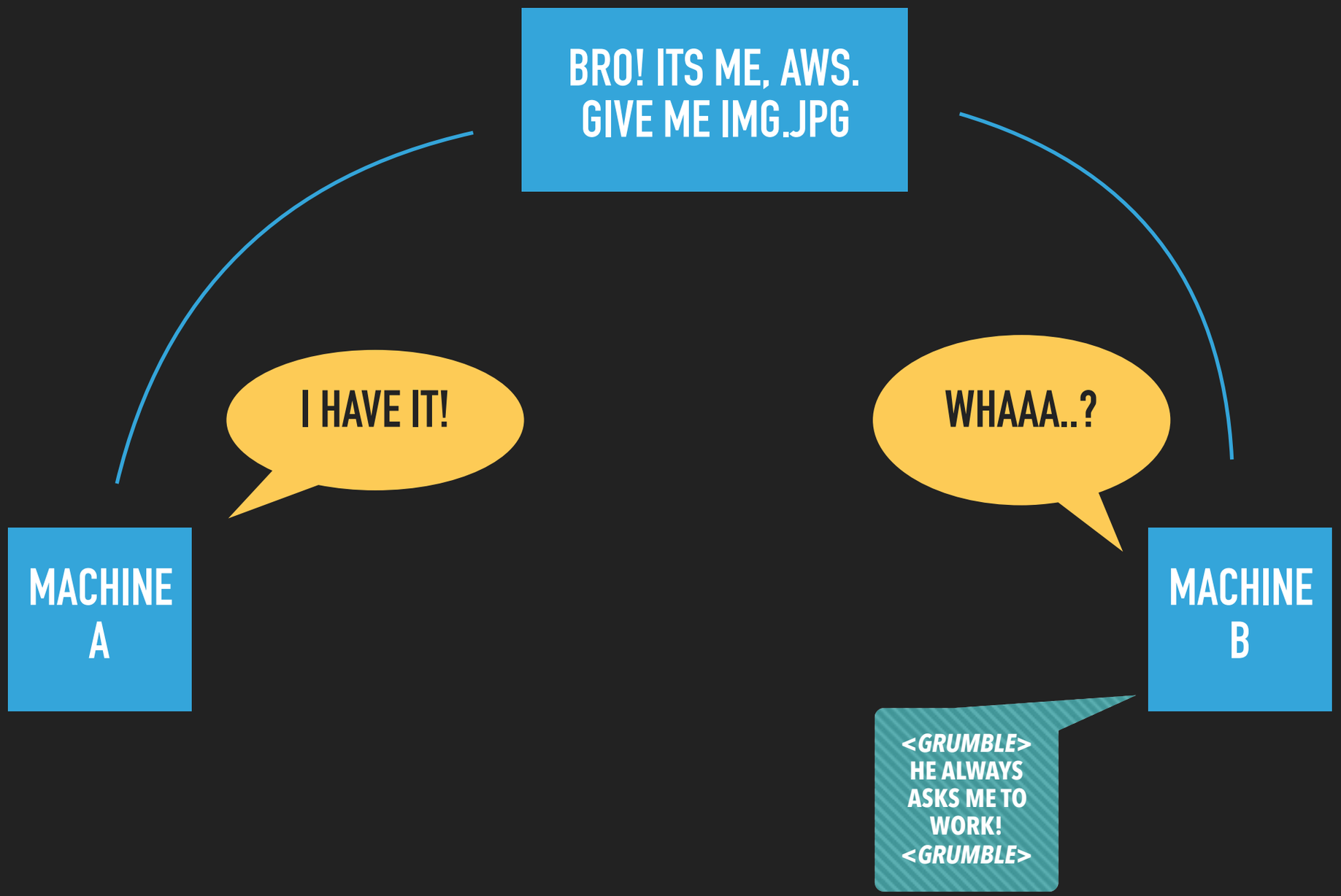
```
proxy_cache_path /tmp/nginx levels=1:2 keys_zone=my_zone:10m inactive=60m;
proxy_cache_key "$scheme$request_method$host$request_uri$browser";

server {
    ...

    location / {
        proxy_cache my_zone;
        add_header X-Proxy-Cache $upstream_cache_status;
        include proxy_params;
        proxy_pass http://localhost:3000;
    }

    ...
}
```

THE JOYS OF LOAD BALANCING



JUST GET IT WORKING

nginx tries to charge for centralised caching. how dare they try make money!

```
const requestHandler = (request, response) => {
  awsS3.get(request.url, request.browser)
  .then(imageBuffer => {
    response.end(imageBuffer)
  })
  .catch(err => {
    wget(request.url)
    .then(imageBuffer => {
        if(request.browser == 'chrome') {
            optimisedBuffer = optimiseImageForChrome(imageBuffer)
            awsS3.put(request.url, request.browser, optimisedBuffer)
            response.end(optimisedBuffer)
        } else {
            optimisedBuffer = optimiseImageForChrome(imageBuffer)
            awsS3.put(request.url, request.browser, optimisedBuffer)
            response.end(optimisedBuffer)
        }
    })
  })
}
```

IMPROVE LOOKUP TIME

- AWS dynamodb
- Low latency lookup table (64k)
- Check dynamodb before s3
- Still 1 extra RTT, but orders of magnitude faster

DEFER IT

- Grow 100x. again. Clog the ~~sewers~~ servers. again.
- Just add more machines? £££

```
const requestHandler = (request, response) => {
    dynamoDB.contains(request.url, request.browser)
    .then(s3URL => {
        // we have optimised image already
    })
    .catch(err => {
        wget(request.url)
        .then(originalImageBuffer => {
            response.end(originalImageBuffer)
            optimiseAndStore(originalImageBuffer, request.url, request.browser)
        })
    })
}
```

who am i kidding, that doesn't work ☹️

SERVERLESS

- AWS Lambda
- Run code in response to a trigger

```
exports.handleAnImage = function(event, context, callback) {
   wget(event.imageURL)
   .then(buffer =>{
       // optimise, store in dynamodb and s3, eat a sandwich
   })
}
```

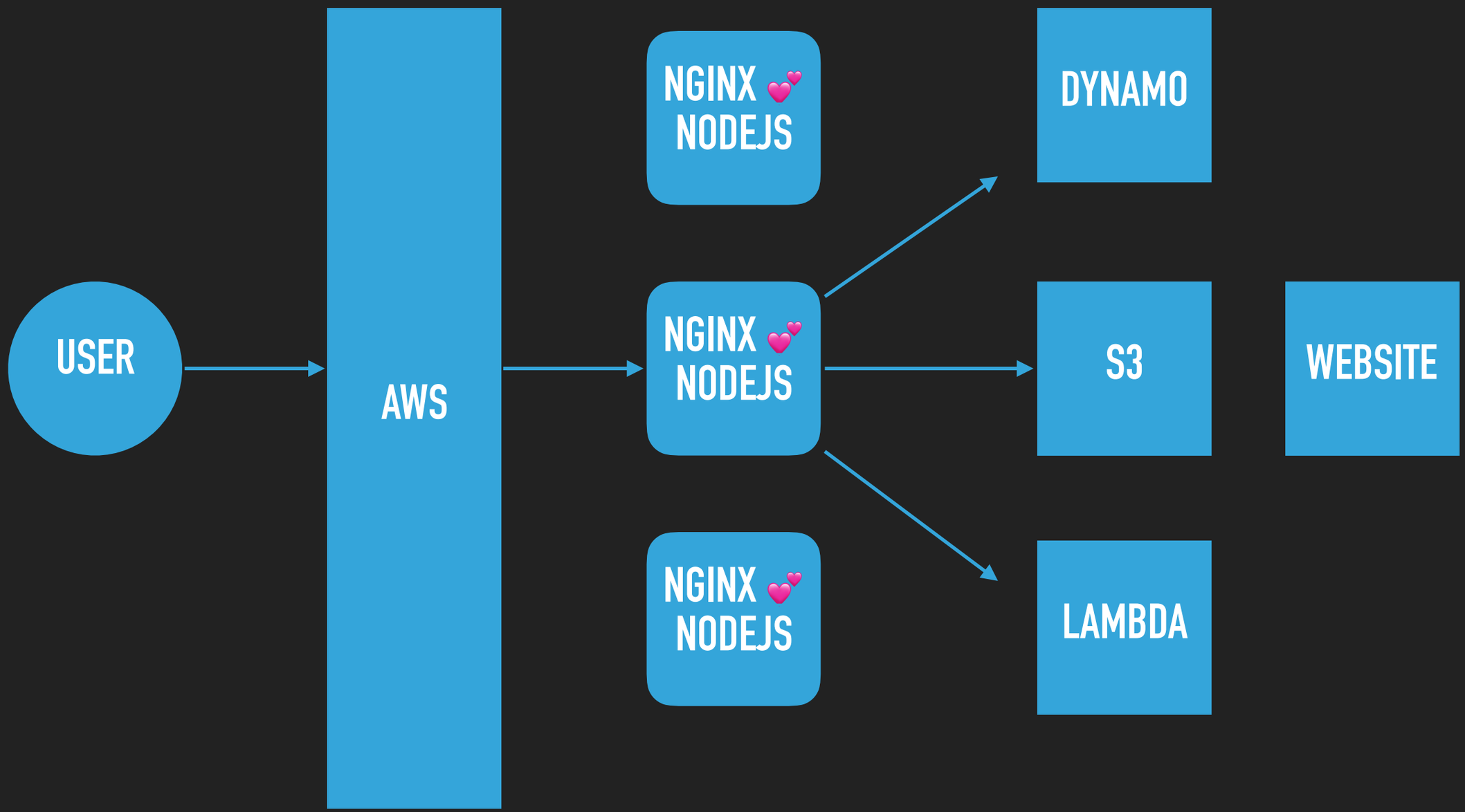
MAXIMISE THROUGHPUT

- A happy server is an idle server

```
const requestHandler = (request, response) => {
    dynamoDB.contains(request.url, request.browser)
    .then(s3URL => {
        // wget and send optimised image here
    })
    .catch(err => {
        // send original image to user without optimising here

        // then just ask lambda to do your work and sleep
        invokeLambdaForOptimisation(request.url, request.browser)
    })
}
```

THE FINAL PRODUCT



THE OUTCOME

- 4 weeks real life time spent
- 40 cans of red bull consumed
- 4,000 pages optimised
- 400,000,000 requests and growing
- pngquant, zopfli, mozjpeg, jpegxr, webp, image resizing, mobile quality prediction, progressive jpeg, service-workers, brotli, h2, imagemagick, alpha-channel removal, dssim selector..

THANK YOU

- dexecure.com
- murali
- make websites great again

- Intelligent optimisation layer over commodity cdn