Deploying Models To the Masses

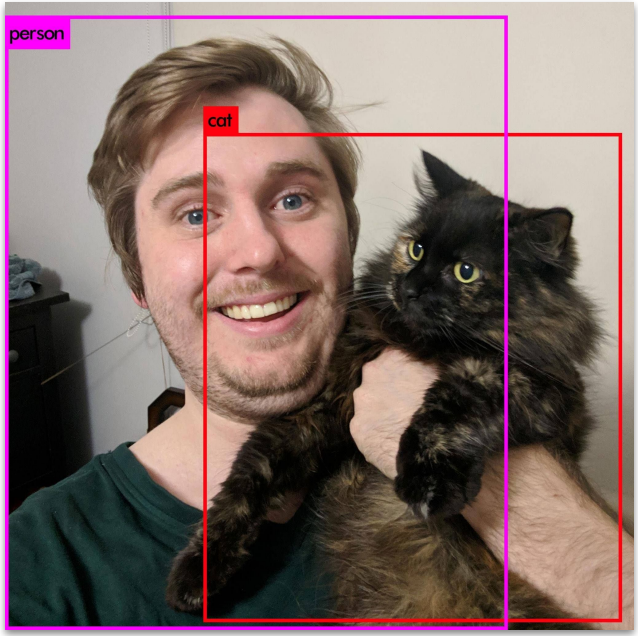
*A couple times I nearly spent a lot of money really fast

Nathan Glover - PMLG - 31/10/2019

@nathangloverAUS

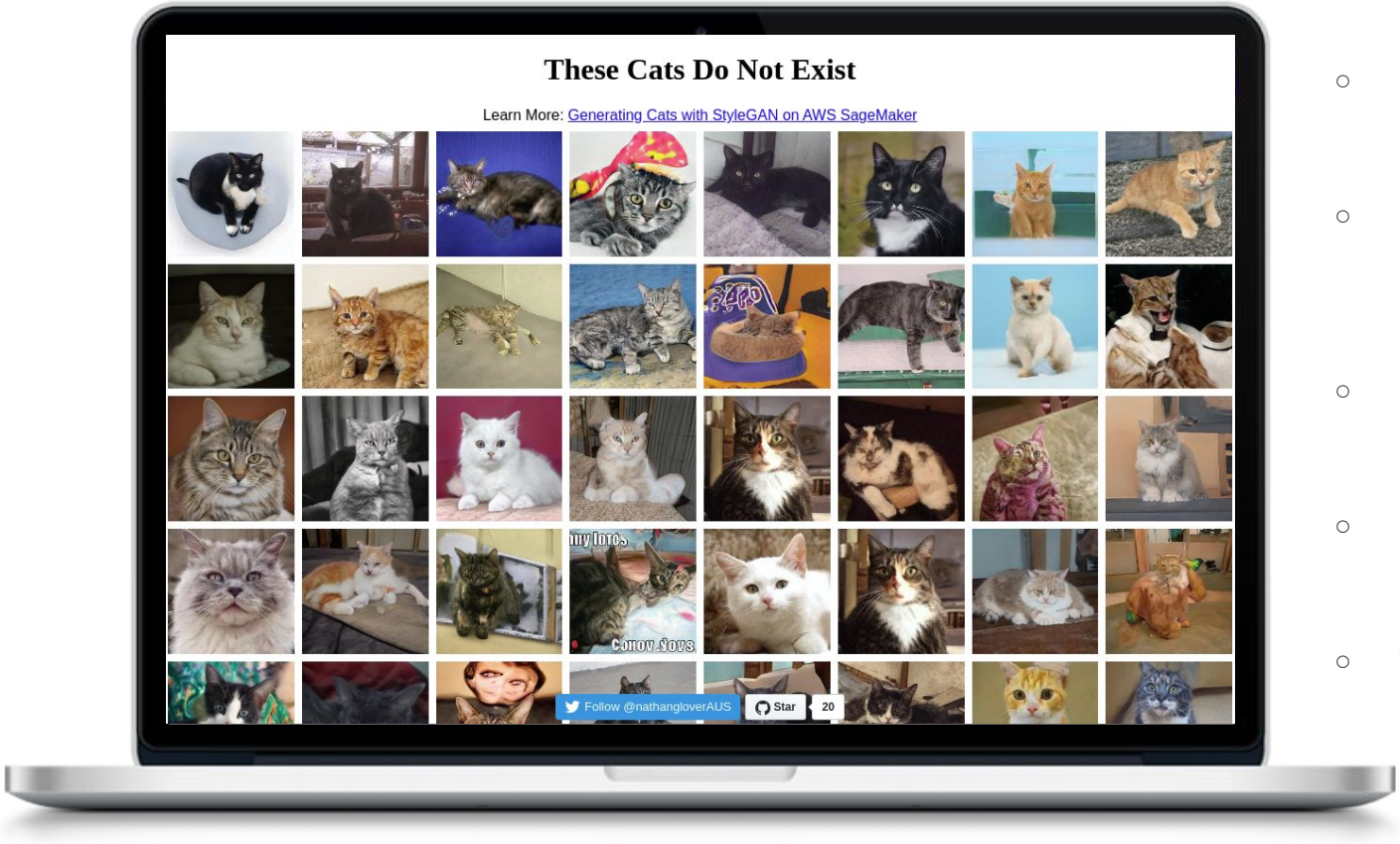
Hi

- **DevOps** Consultant @mechanicalrock_
- Got into Machine Learning after working around Data Scientists
  - Helping setup notebooks
  - Data platforms
- Got hooked
- Love building **weird** / **fun** things that make people smile ❤️



The Good, the Bad and the (very) Ugly

- **Accidently** had a couple things go semi-viral
- **Sharing** is **Caring** is - I'll save you some money
- Don't do some things I talk about (maybe) 🤦‍♀️
- I tackle most of these projects from an Ops perspective *(this is me saying I'm not very good at explaining ML concepts sometimes)*



- Leveraged **NVlabs/stylegan**
- Started out as just a learning exercise for SageMaker
- Over the week, **50,000~** visitors
- Got **Very Lucky** with Timing
- **We didn't even do it Well!**

Timeline

Feb 11th 2019

Model Launch

NVIDIA Labs released StyleGAN

NVlabs/stylegan

Feb 16th 2019

Learning SageMaker

Just started to learn how to use SageMaker, wanted to do something fun with it.

Saw StyleGAN and pestered my infinitely more qualified friend; Stephen for help

Feb 17th 2019

Website

Registered the domain… Before writing any code...

Feb 18th 2019

Generated Cats

On Stephen's GPU cluster (his desktop computer in Summer)

This becomes a bad habit for us

Site Launched that evening

Feb 19th 2019

Awoke to craziness

Hackernews, Reddit, Twitter, Futurism, CNN (lol what?)

How?

- Lots of people asked **hard** questions
- How'd you **train** the model?
  - *We didn't*
- What **batch size** did you use?
  - *A dozen 🍞?*
- How'd you run **inference** so quickly?
  - *You won't like the answer*

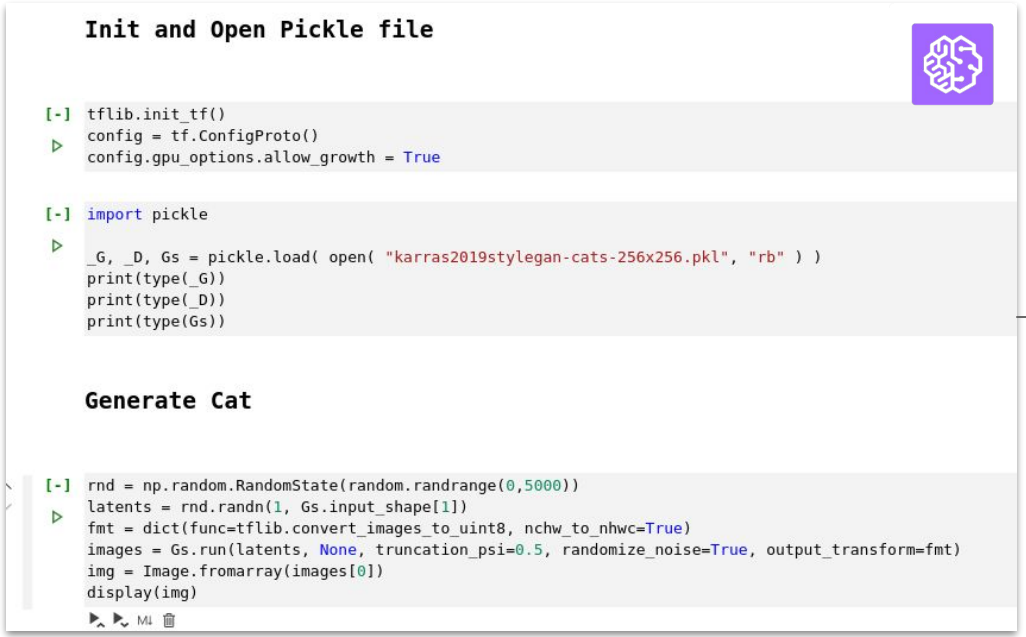
```
<script>
    function getImage() {
        let url = "https://d2ph5fj80uercy.cloudfront.net/"   + getRandomCat();
        return url;
    }
    function getRandomCat() {
        var folderNumber = Math.floor((Math.random() *   3) + 1);
        switch (folderNumber) {
            case 1:
                var catNumber = Math.floor((Math.random() *   8000) + 1);
                var path = "0" + folderNumber +  "/cat" + catNumber +  ".jpg";
                return path;
                Break;
           case 2:
                var catNumber = Math.floor((Math.random() *   8000) + 1);
                var path = "0" + folderNumber +  "/cat" + catNumber +  ".jpg";
                return path;
                break;
            case 3:
                var catNumber = Math.floor((Math.random() *   8000) + 1);
                var path = "0" + folderNumber +  "/cat" + catNumber +  ".jpg";
                return path;
                break;
        }
    }
</script>
```

HTML is the best Data Science Language

don't @ me

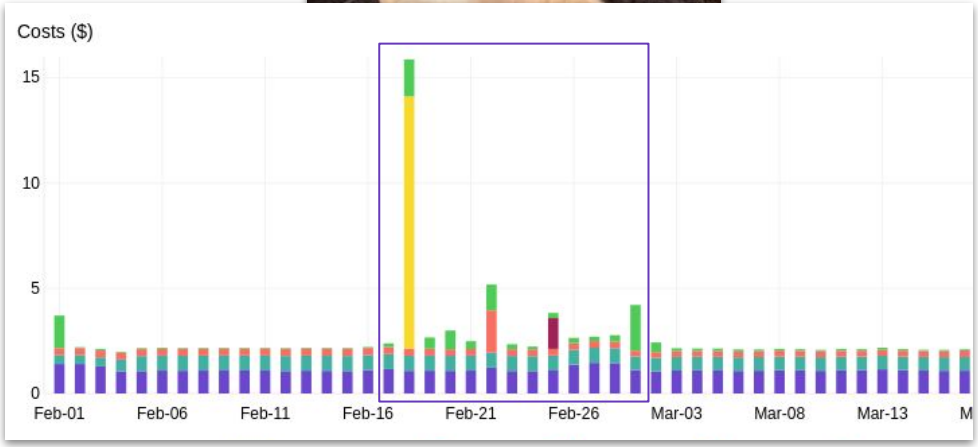
No Really, How?

- We just exported **thousands and thousands** of images
- Saved to S3
- Served over CloudFront

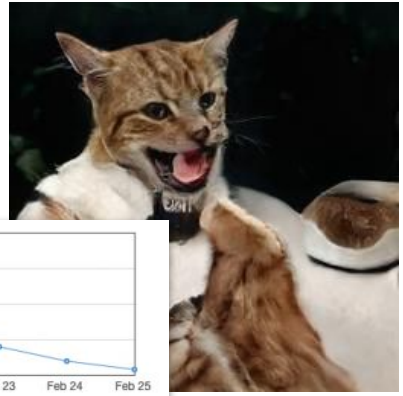


Costs

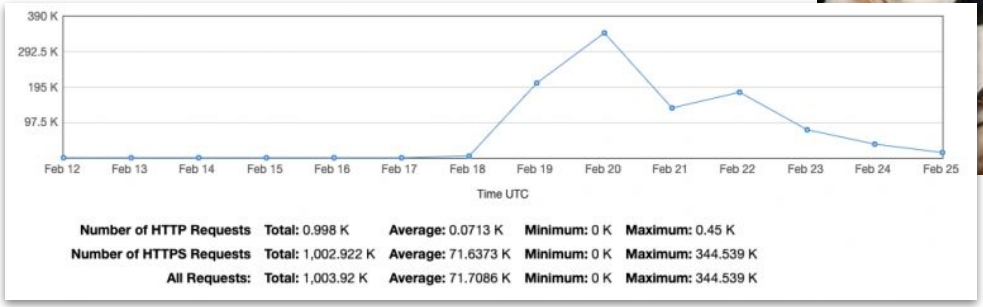
- Wallet didn't suffer
- Most expensive cost?
  - **Domain** ¯_(ツ)_/¯
- Then?
  - S3 *(image hosting)*
  - CloudFront *(image serving)*
  - SageMaker *(inference)*
- Since launch, almost all hosting has been free-tier



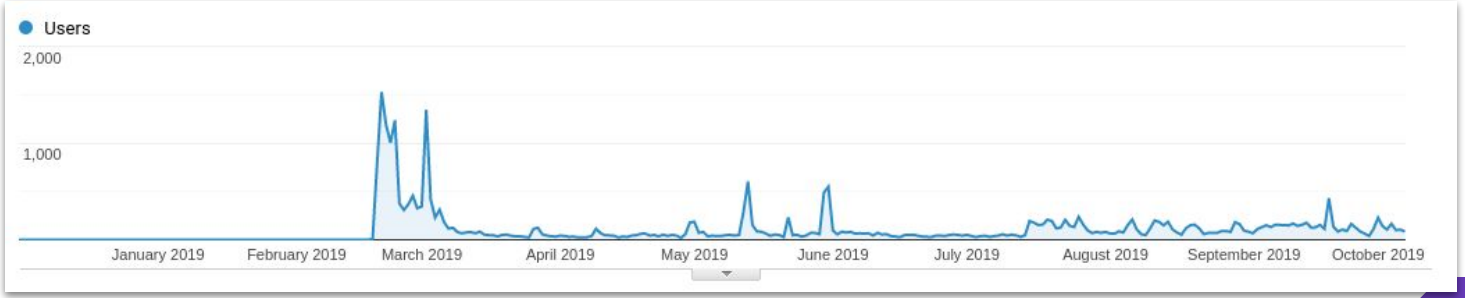
Stats



- +1,000,000 requests to images
- **20GB+** of image cache hits
- Site has seen a constant stream of traffic since



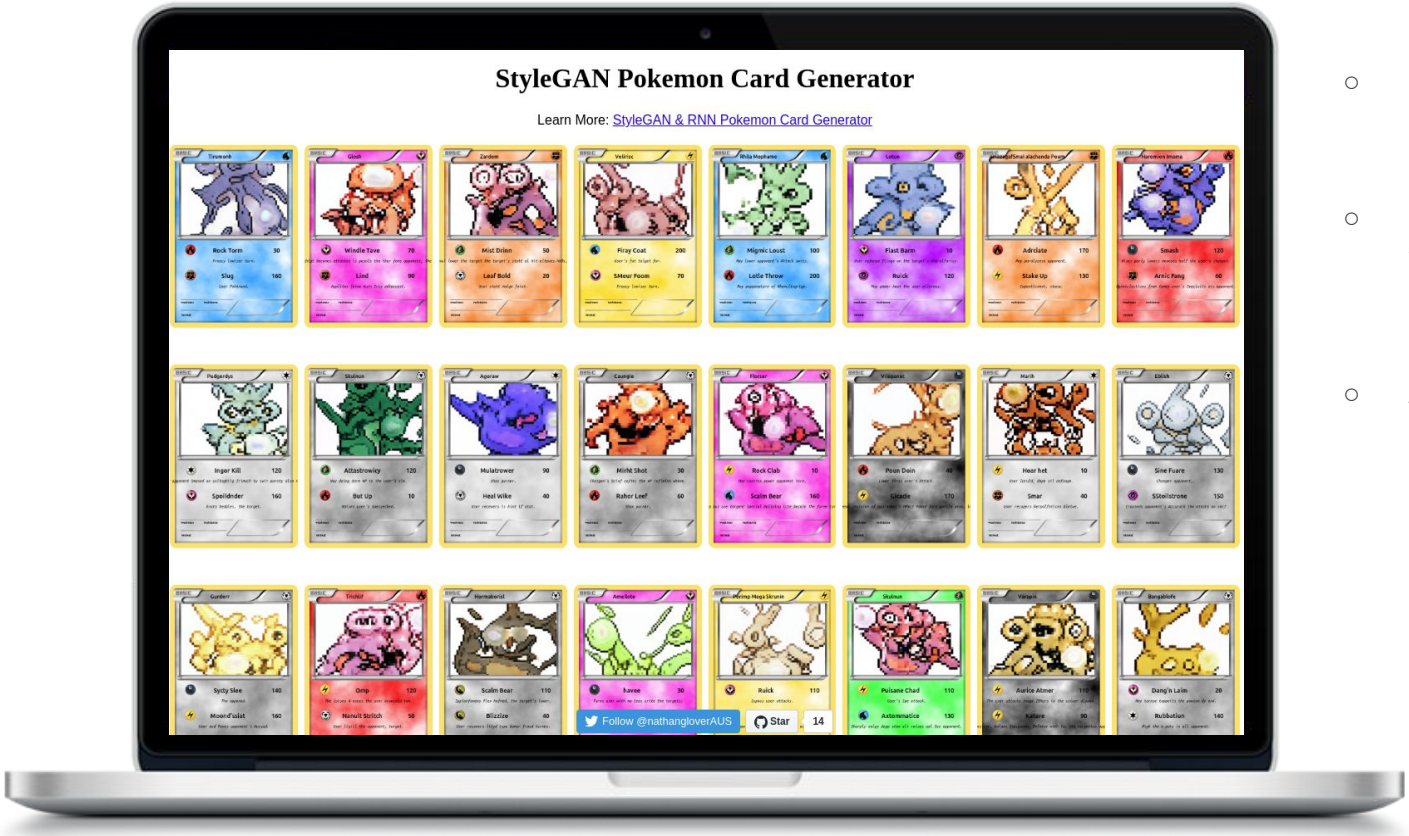
CloudFront Requests



Google Analytics

The Learnings

- The more I learnt, the less I knew.
- Was spoilt by **pre-trained models**
- Google Ads are very very hard to qualify for.
- Deploying these large models **is HARD**
- **CloudFront** and **S3** can be quite expensive, I unfortunately didn't recognise this at the time...



- **StyleGAN**, **LSTM**, **RNN**
- Less popular, but a lot more work
- Actually **trained a model myself** this time

Totally Random

- **RNN** - Generated move sets and names
- **StyleGAN** - Generated Images
- Trained on images from **gen1-5**

Unholy amount of FFMPEG

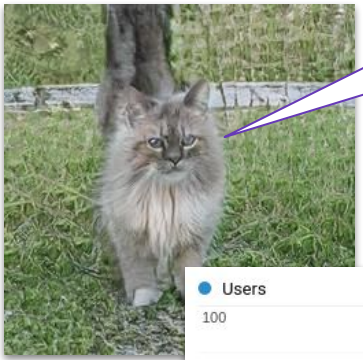
```
# Trim
convert $6 -trim $6

# Crop
convert $6 -resize 390x291^ -gravity Center -extent 390x291 -crop
390x291+25+0 $6

# Overlay Pokemon Image
convert img/templates/$CARD_TYPE.png $6 -geometry +35+55 \
    -compose DstOver -composite \
    $6

# Pokemon Name Insert
convert $6 -font Ubuntu-Medium -pointsize 17 -gravity North -background
none -splice 0x32 \
    -annotate -45+60 "$POKEMON_NAME" \
    $6

# Pokemon Attack 1 Insert
convert $6 -font Ubuntu-Medium -pointsize 20 -gravity South -background
none -splice 0x32 \
    -annotate -20+270 "$POKEMON_ATK1" \
    -annotate +140+270 "$POKEMON_POW1" \
    $6
```

```
# Pokemon Attack 1 Desc
convert $6 -font Ubuntu-Mono-Italic -pointsize 14 -gravity South -background none \
    -annotate +0+240 "$POKEMON_ATK1_DESC" \
    $6

# Overlay Attack 1 Symbol
convert $6 img/templates/s$POKEMON_ATK1_TYPE.png -background none -alpha set -gravity South \
    -geometry -140+265 -define -compose -composite \
    $6

# Pokemon Attack 2 Insert
convert $6 -font Ubuntu-Medium -pointsize 20 -gravity South -background none -splice 0x32 \
    -annotate -20+220 "$POKEMON_ATK2" \
    -annotate +140+220 "$POKEMON_POW2" \
    $6

# Pokemon Attack 2 Desc
convert $6 -font Ubuntu-Mono-Italic -pointsize 14 -gravity South -background none \
    -annotate +0+190 "$POKEMON_ATK2_DESC" \
    $6

# Overlay Attack 1 Symbol
convert $6 img/templates/s$POKEMON_ATK2_TYPE.png -background none -alpha set -gravity South \
    -geometry -140+215 -define -compose -composite \
    $6
```

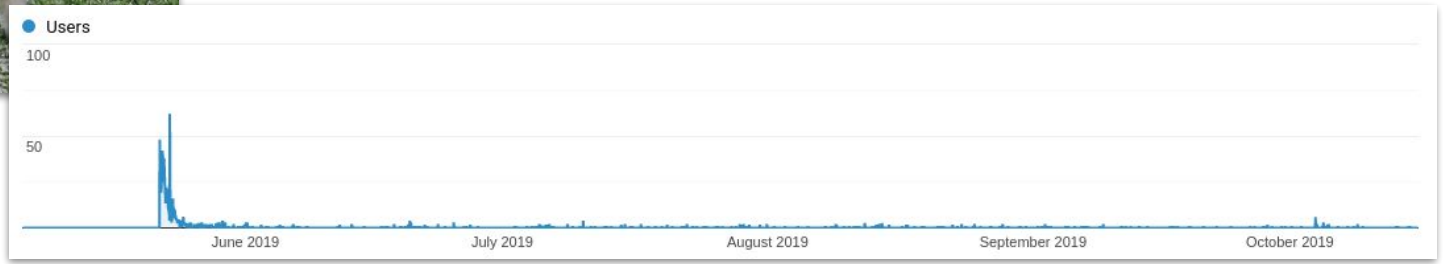
And its all done dynamically in real time yes?

SHHHH YOU

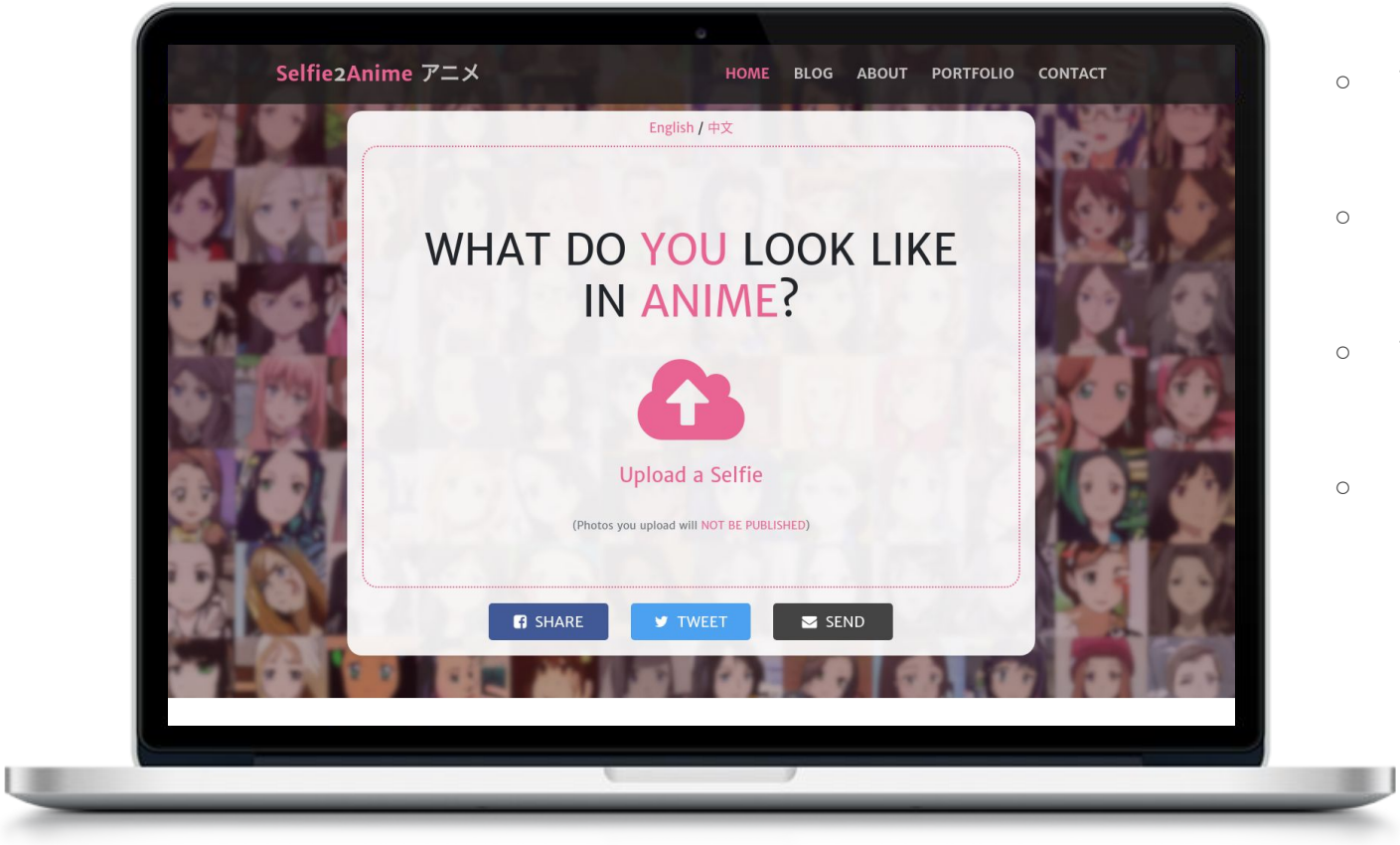
Stats



Wait... so riding a hype-wave doesn't always work?



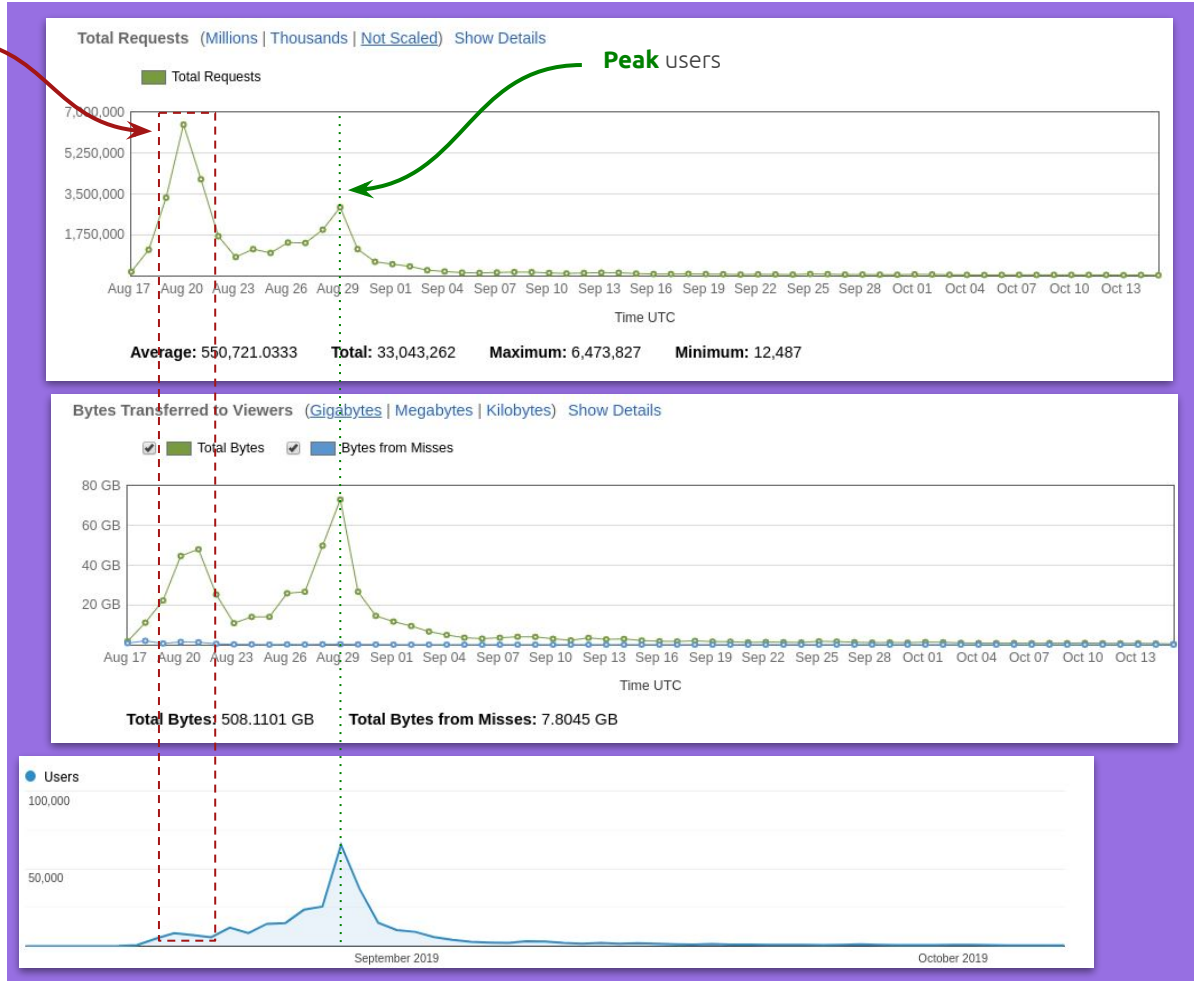
Google Analytics



- That week I **didn't sleep good**
- Broke the bank
- Tested the limits of a friendship
- Learnt about some **hidden limits** on Cloud providers

The **$20 a day** mistake

- **>500GB** of cached requests.
- Peaked at **6.4 million** requests to our site
  - (though this metric has a story to it)
- **615,000 selfies recieved**, converted and emailed.
- **No monetization strategy**



Ok, but what is it?

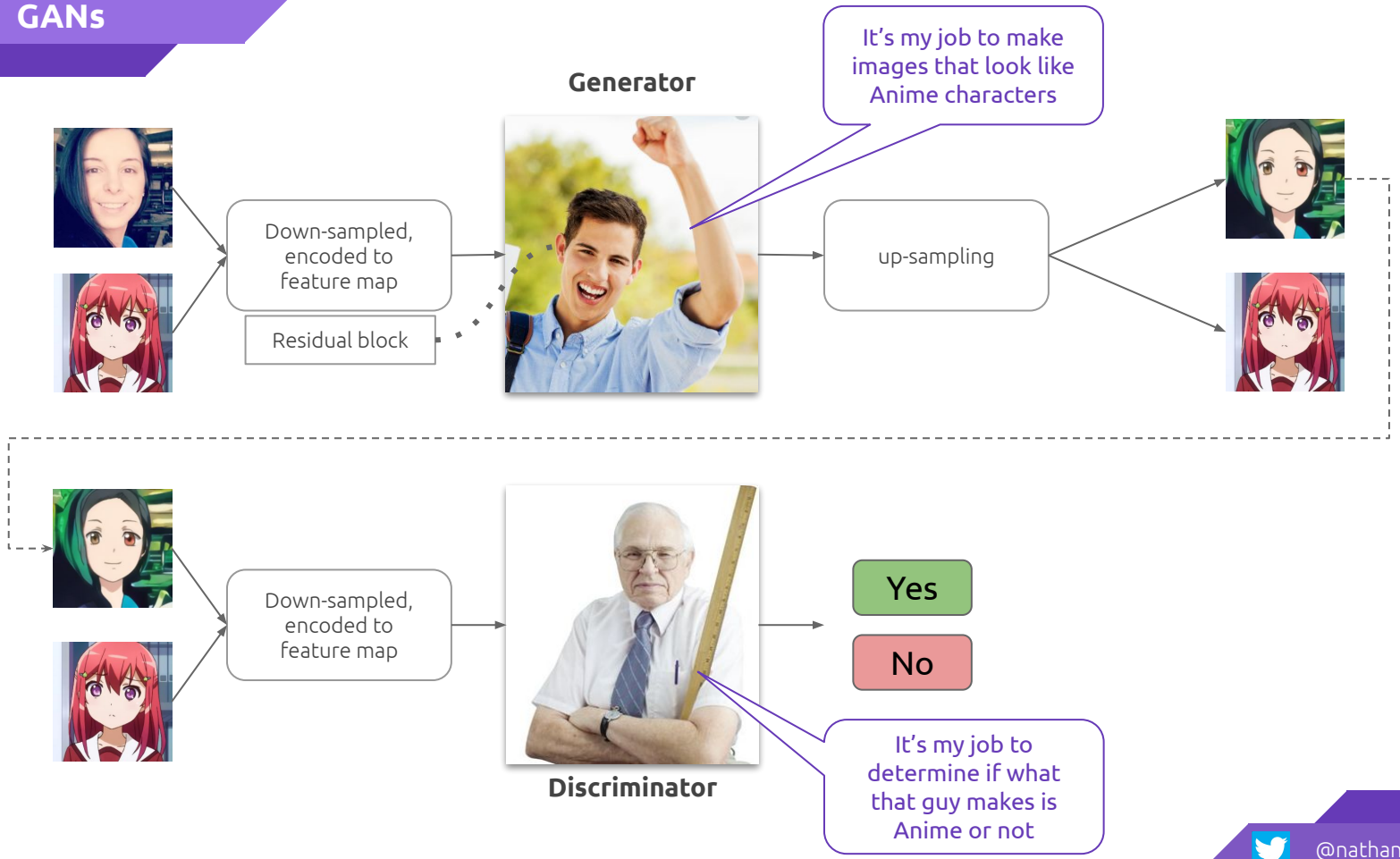
U-GAT-IT (Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation)

Put simply, it's a **GAN** (Generative Adversarial Network)

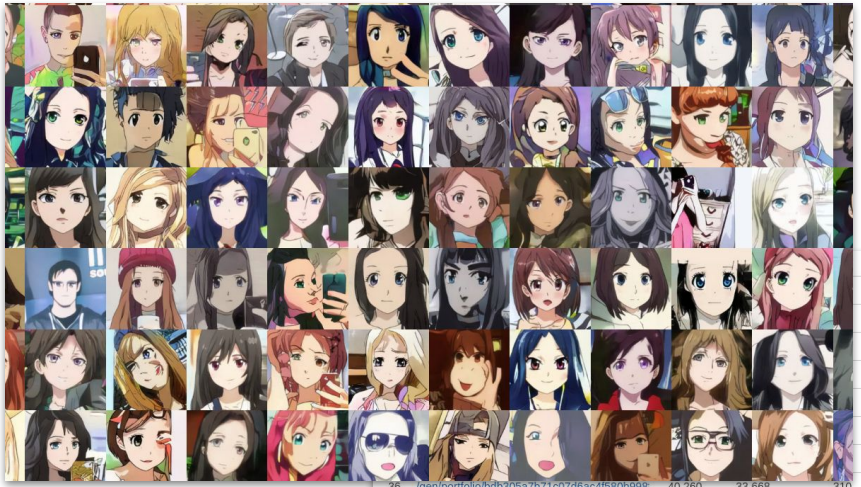
+

Attention maps that guide the network to better results

GANs


Generator
It's my job to make images that look like Anime characters
Down-sampled, encoded to feature map
Residual block
up-sampling
Down-sampled, encoded to feature map
Yes
No
Discriminator
It's my job to determine if what that guy makes is Anime or not

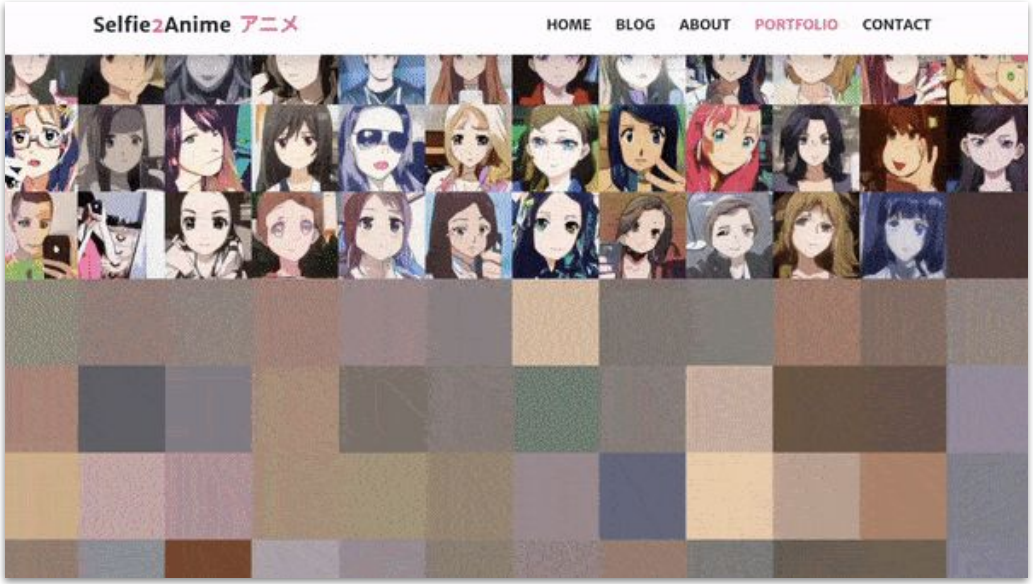
The $20 a day mistake



					80.45%	2.66 MB	256.28 MB
					89.71%	2.18 MB	217.19 MB
					81.72%	2.09 MB	211.87 MB
					89.22%	2.17 MB	207.10 MB
					80.17%	1.92 MB	202.37 MB
					80.58%	1.99 MB	199.87 MB
					80.33%	1.79 MB	187.31 MB
					83.38%	1.73 MB	182.81 MB
36	/gen/portfolio/bdb305a7b71c07d6ac4f580b998	40,260	33,668	310	83.63%	1.69 MB	176.29 MB
37	/gen/portfolio/f238abee1f1e7674a8745f9c9b8b	40,736	33,591	347	82.46%	1.87 MB	172.41 MB
38	/gen/portfolio/d9d1d30cc7c4d4a7f9d2ae42156	39,383	32,107	262	81.53%	1.47 MB	168.56 MB
39	/gen/portfolio/dde32adb3e41b6fd4739e1d16e2	39,545	32,422	316	81.99%	1.68 MB	166.51 MB
40	/gen/portfolio/fe7faf30408c52c9a928c785cfd81	38,268	33,779	347	88.27%	1.66 MB	159.00 MB
41	/gen/portfolio/e9f9dfff970cfa2a140845948f4ff3	37,310	30,207	247	80.96%	1.33 MB	154.90 MB
42	/gen/portfolio/c238be52d527a97246e8463d0d	38,542	32,270	274	83.73%	1.38 MB	152.88 MB
43	/gen/portfolio/a408e69c3729141224504ae29d	37,173	30,684	264	82.54%	1.36 MB	149.37 MB
44	/gen/portfolio/e1fcf6ab2adb522399f14018c9bc	37,694	33,250	313	88.21%	1.44 MB	147.44 MB
45	/gen/portfolio/ed699e5ddc77a6d3362e01da55	40,225	33,085	304	82.25%	1.38 MB	143.64 MB
46	/gen/portfolio/ab7ebc811fb069a35334addac81	37,220	29,954	252	80.48%	1.31 MB	143.05 MB

- **180** Images (90 selfie, 90 anime) made up a beautiful collage
- HTTPS Requests went through the roof
- I learnt a lot about **cost explorer**!

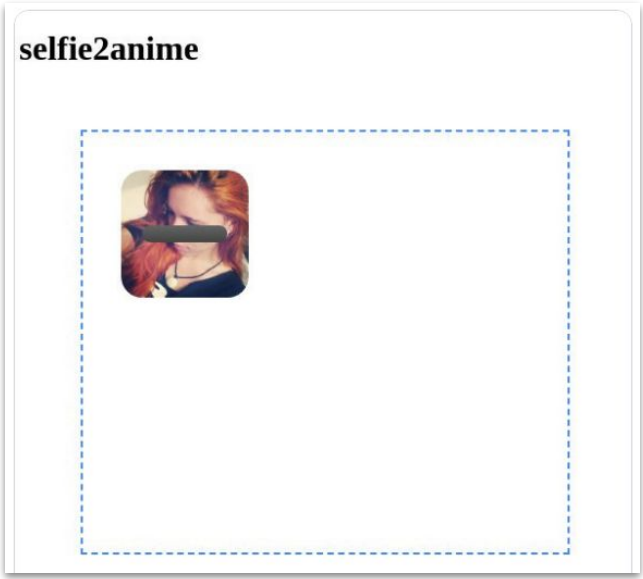
The $20 a day mistake (Fix)



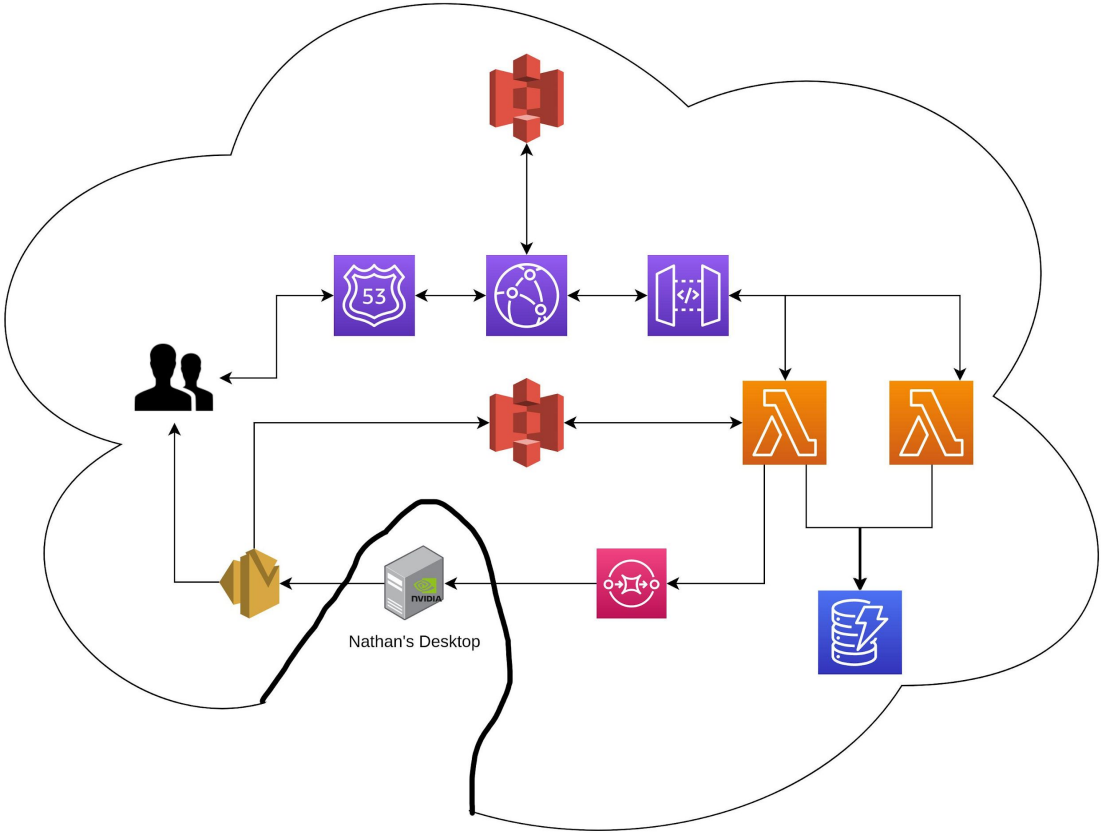
Lazy Load your images!

Architecture (v0.1)

- **v0.1** was a Docker container running on my desktop
- Worlds worst Flask webapp
- Having images of people land on my disk made me **uncomfortable**
- **Single threaded**
- *Launched the first public model on Kaggle at the time, along with the containerized app*



Architecture (v0.2)



- **Serverless** up to a point where it became too hard 😂
- UGATIT fork with an SQS processor.
- **80% elegant**, **20% s**t**

Bugs

```
while True:
    messages = get messages from_queue()
    for message in messages:
        body = json.loads(message[ 'Body'])
        print(body)
        bucket = body[ 'bucket name' ]
        bucket key = body[ 'bucket key' ]
        file name = body[ 'file name' ]
        email addr = body[ 'email']
        crop = body[ 'crop' ]

        image = download_image(bucket, bucket_key)

        # Crop params
        x = crop[ 'x']
        y = crop[ 'y']
        width = crop[ 'width']
        height = crop[ 'height']
        crop img = image[y:y+height, x:x+width]
        # Change color space
        crop_img = cv2.cvtColor(crop_img, cv2.COLOR_RGB2BGR)

        # Resize image
        crop_img = cv2.resize(crop_img, dsize=( 256, 256))

        # do some fancy processing here....
        fake_img = gan.test_endpoint(crop_img)

        # Upload to S3
        image_url = upload_image(fake_img, file_name)

        # Send Email
        email.send_email(email_addr, image_url)
    time.sleep( 10)
```

- **SQS** queue processing worked well
- Core Python loop for processing was VERY unreliable
- **No try, catching**
- **No error handling**
- **No in depth understanding** of what was happening behind the curtains

AHHHHHHHHHH

- **SQS** processing hit a point where my PC just wasn't enough anymore
- Had to look into other (cheap) options
- Got **$3k GCP credits during startup weekend**
- **Google Cloud notebooks** to the rescue
- **K80 GPU** attached for extra **(((S P E E D)))**
  - Wasn't necessary but we didn't know this at the time.

UGATIT

Pull Dataset

```
In [ ]: !gsutil cp gs://devopstar/projects/data-science/UGATIT/*.zip ./dataset
        !unzip -qq dataset/selfie2anime.zip -d ./dataset
```

Pull Existing Training

```
In [ ]: !mkdir samples
        !mkdir checkpoint
        !gsutil -m rsync -d -r gs://devopstar/projects/data-science/UGATIT/samples samples
        !gsutil -m rsync -d -r gs://devopstar/projects/data-science/UGATIT/checkpoint checkpoint
```

Train

```
In [ ]: !python3 main.py --dataset selfie2anime --phase train
```

Test

```
In [ ]: !python3 main.py --dataset selfie2anime --phase test
```

Runner

```
In [ ]: !pip3 install boto3 flask-dropzone flask-uploads requests jsonpickle flask Pillow opencv-python
```

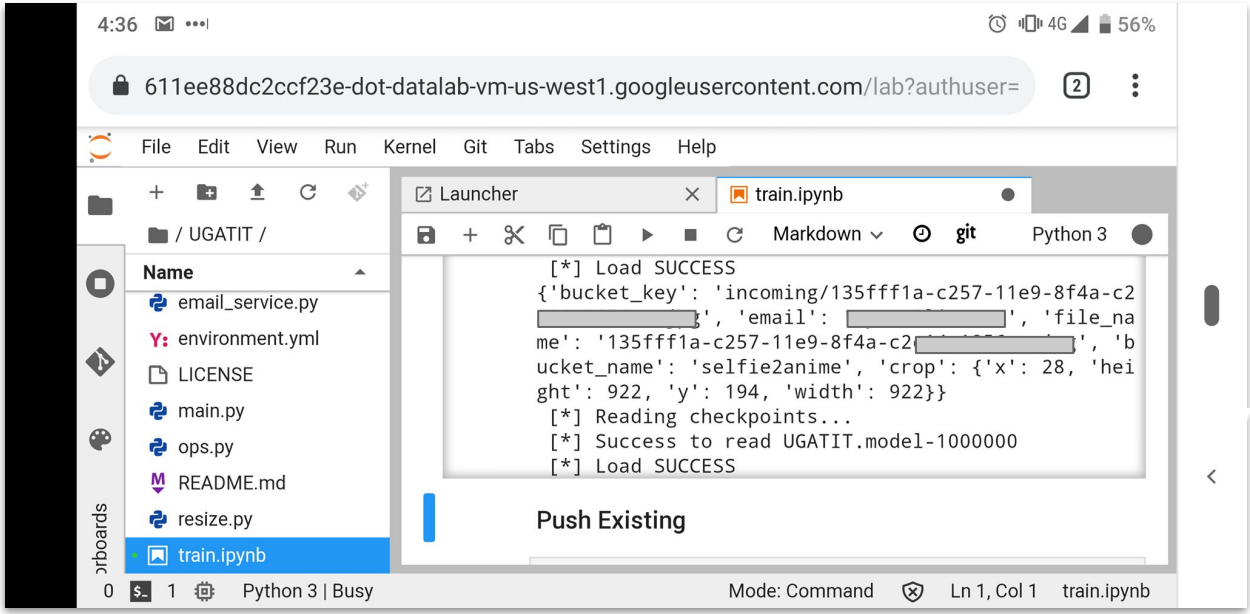
```
In [ ]: !QUEUE_NAME=selfie2anime BUCKET_NAME=selfie2anime SENDER_EMAIL=noreply@selfie2anime.com python3
        main.py --dataset selfie2anime --phase runner
```

Push Existing

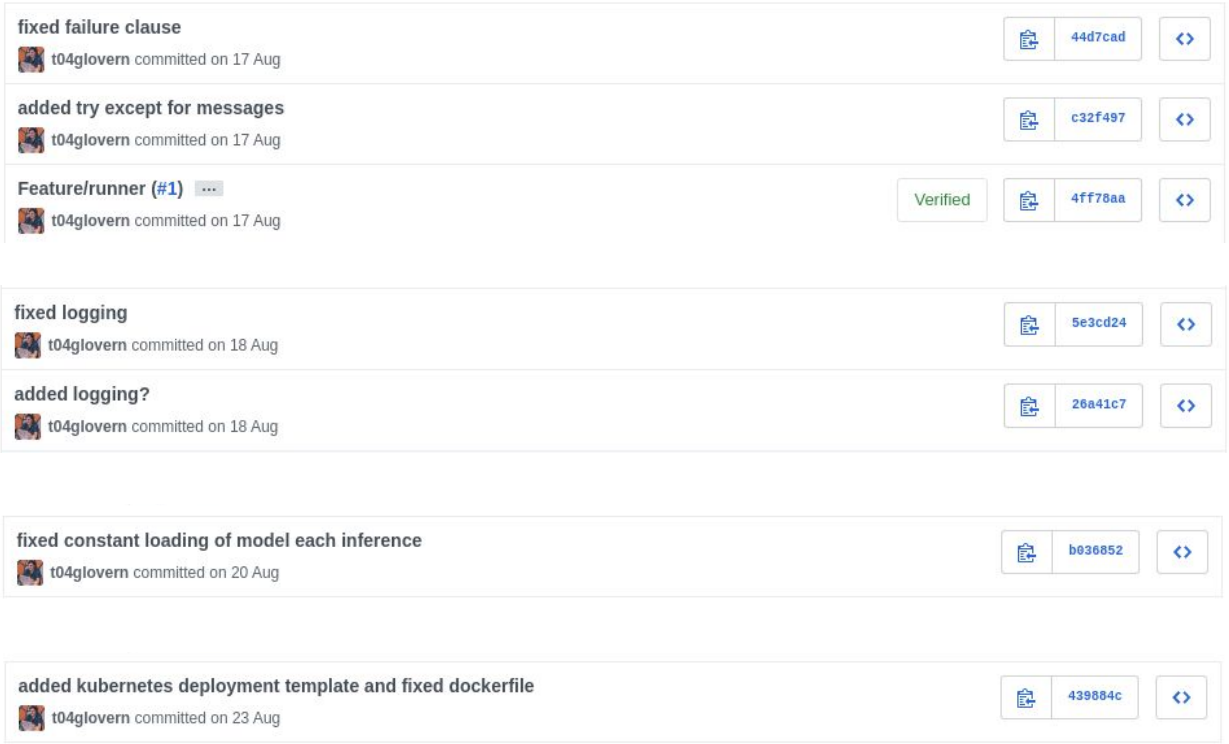
```
In [ ]: !gsutil -m rsync -d -r samples gs://devopstar/projects/data-science/UGATIT/samples
        !gsutil -m rsync -d -r checkpoint gs://devopstar/projects/data-science/UGATIT/checkpoint
```

The Things you do for uptime

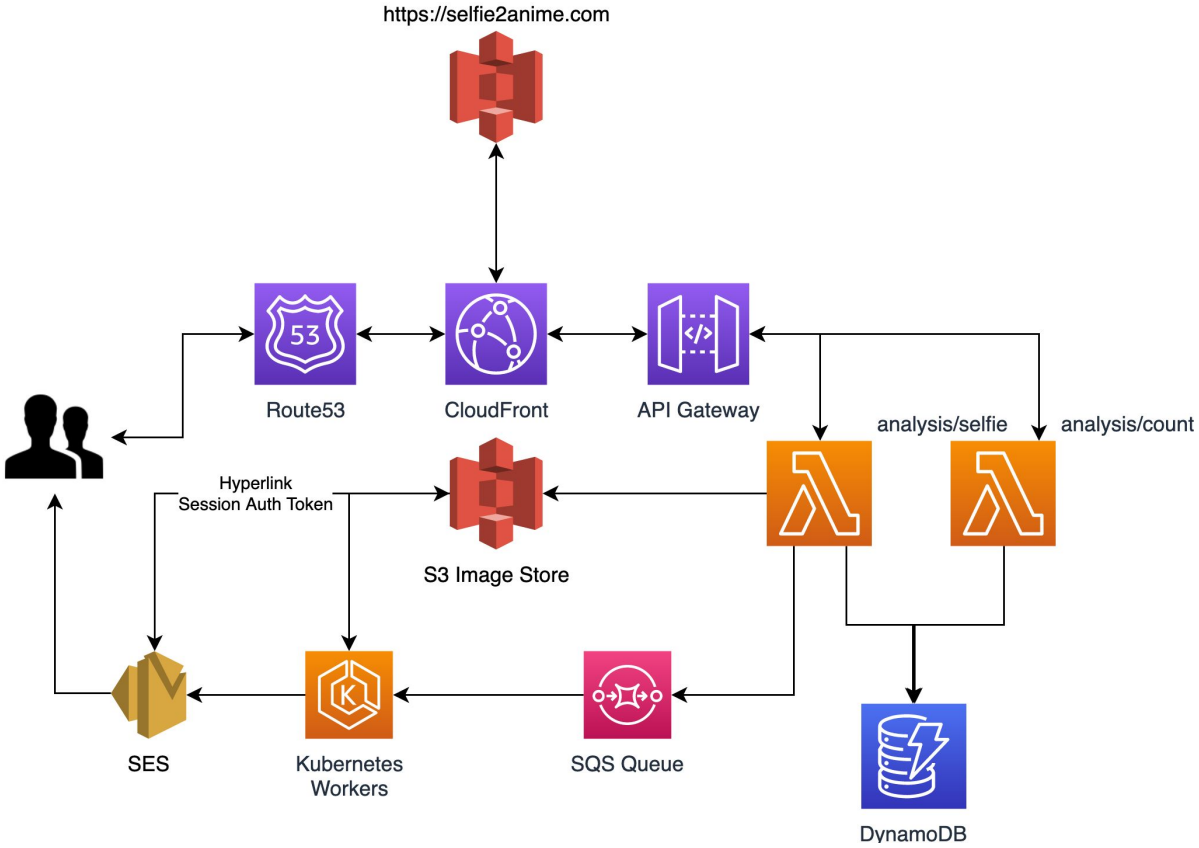
Detacting from the notebook just wasn't possible due to bugs



Paying off the debt

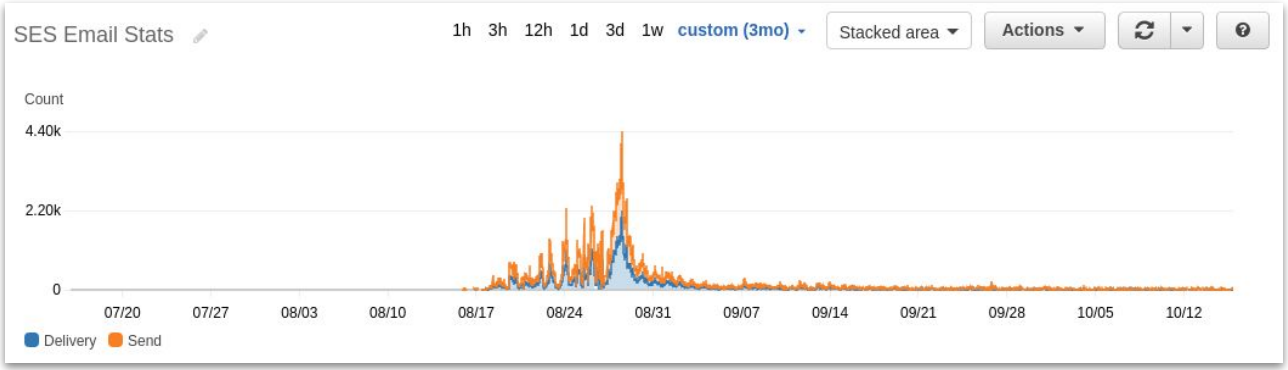


Architecture (v1.0)

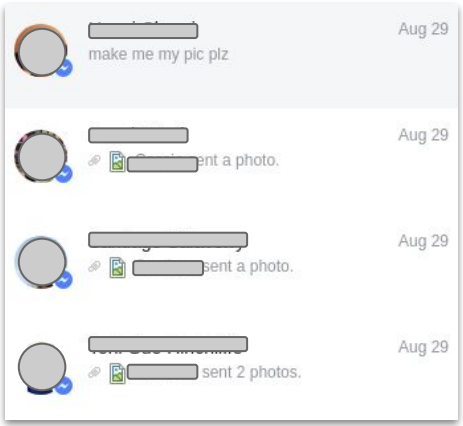


- Could **finally** get a good nights sleep
- Scaling up and down was now easy *(compared to notebook instances)*
- Being able to scale to meet demand meant daily views were increasing very quickly

SES Send Limits



- Hit **SES** send limits for the first time
- Totally blocked from meeting demand until our rates got increased
- Scaled back processing, 1-2 hour delays on images for people



People craved a better service

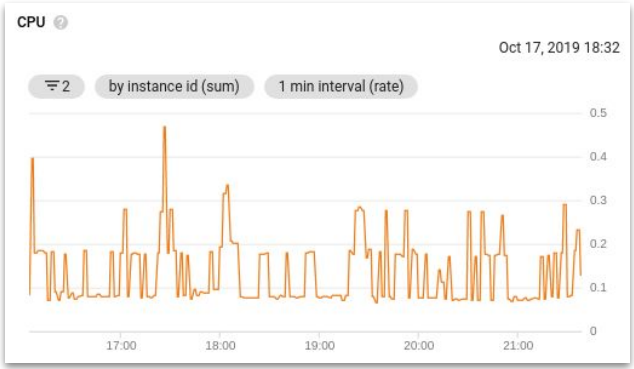
Selfie2Anime running on:

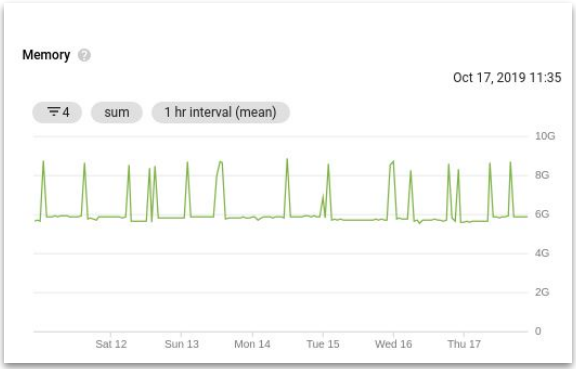
Compute Instance

Kubernetes

Bursty Workload by design

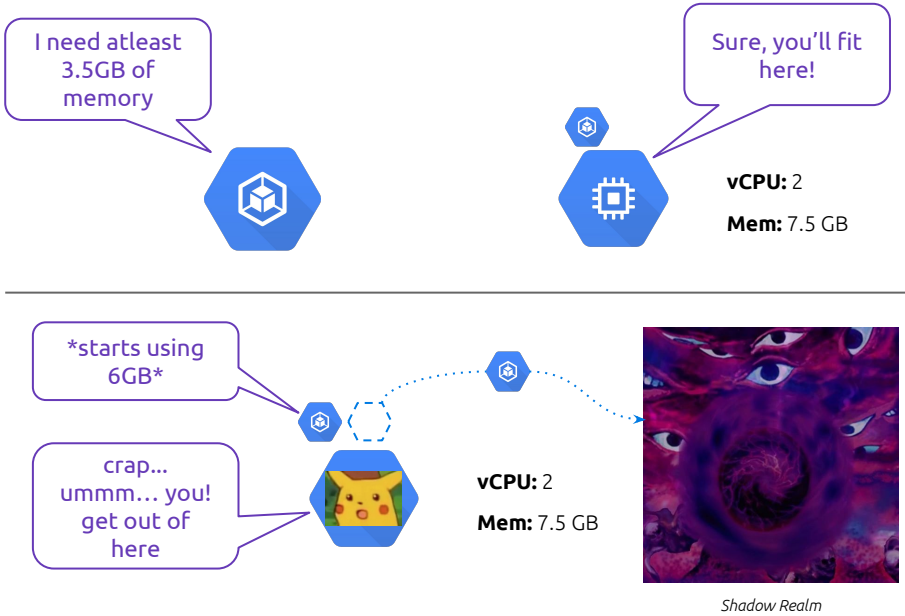
- Inference was very bursty
- Pod **eviction occured regularly**
  - Jobs were being lost to the void 🕳
- Pod headroom required more money





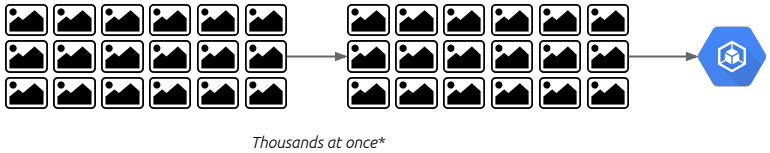
The Scheduler was too smart

- **Not** a Kubernetes specific “problem”
- /proc/sys/vm/overcommit_memory = 0 || 1
  - **OOM-Killer** (out of memory killer)
- Undercommit means pods get evicted
- Overcommit means pods never get scheduled
  - unless new nodes come up 💰🚽

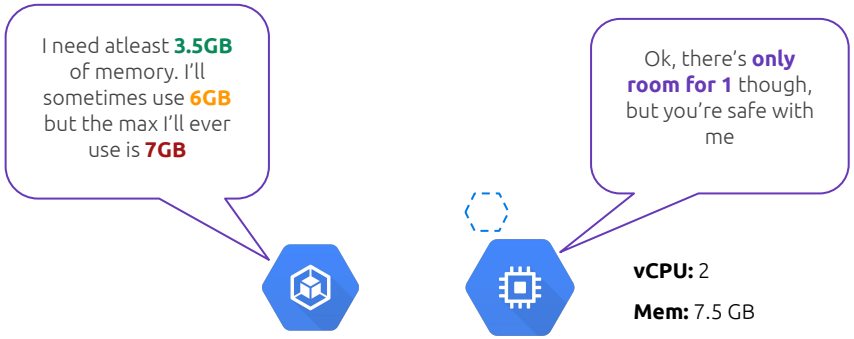


The “Fix”?

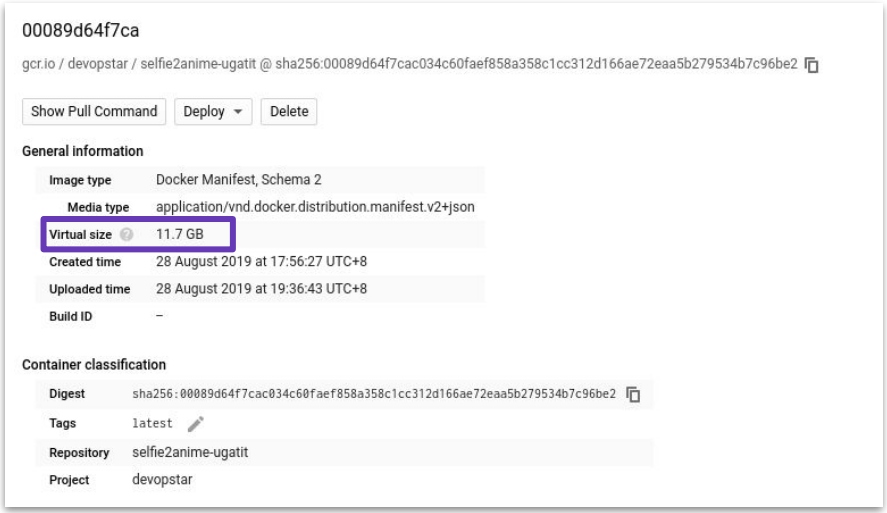
- **Scheduling** is hard on a budget
- Ended up in a position where using a compute instance *might* have been a smarter decision.
- Batching the queues, short lived nodes to deal with chunks **would** have been most effective



*Thousands at once**

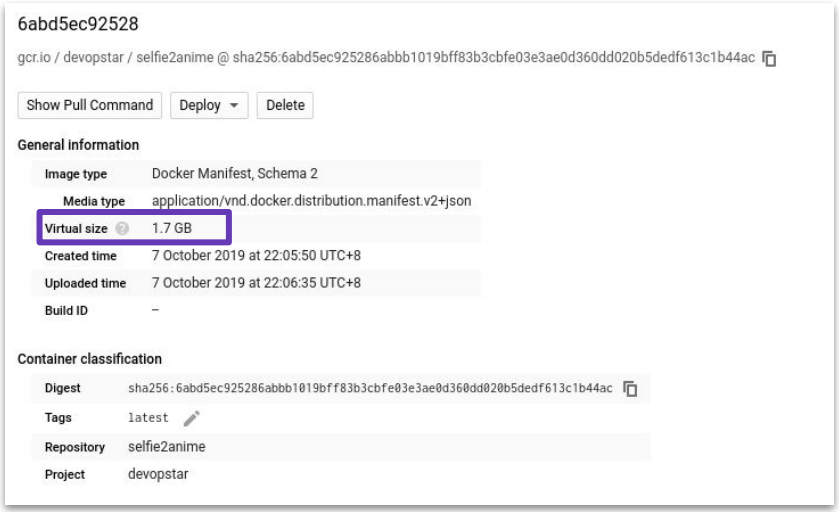


Giant Containers



- **Ignored** this problem while trying to scale
- **Spot instances on Nodes** to save money
  - When nodes were taken away, 5 minute delays while node populated the container.
- We could do better

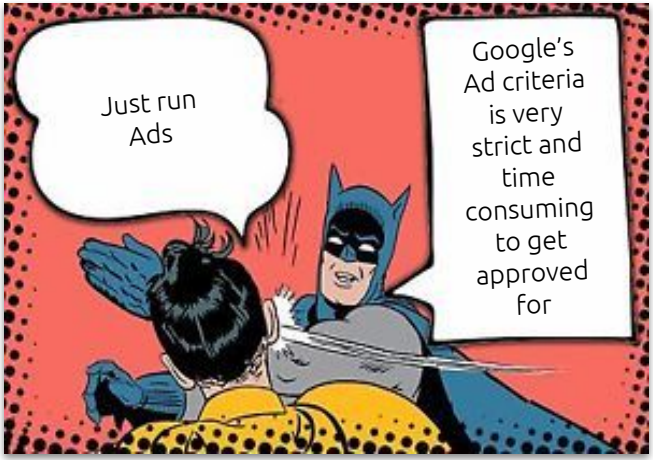
Giant Containers (Quick Fix)




```
RUN apt-get update && apt-get install -y gcsfuse

RUN mkdir -p /app/checkpoint

COPY . /app
WORKDIR /app

ENTRYPOINT [ "python3" ]
CMD [ "main.py" ]
```

```
  lifecycle:
    postStart:
      exec:
        command: ["gcsfuse", "-o", "nonempty", "selfie2anime", "/app/checkpoint"]
    preStop:
      exec:
        command: ["fusermount", "-u", "/app/checkpoint"]
```

How to make money?



Google AdSense	You need to fix some issues before your site is ready for AdSense
Google AdSense	You need to fix some issues before your site is ready for AdSense
Google AdSense	You need to fix some issues before your site is ready for AdSense
Google AdSense	You need to fix some issues before your site is ready for AdSense
Google AdSense	You need to fix some issues before your site is ready for AdSense
Google AdSense	You need to fix some issues before your site is ready for AdSense
Google AdSense	You need to fix some issues before your site is ready for AdSense

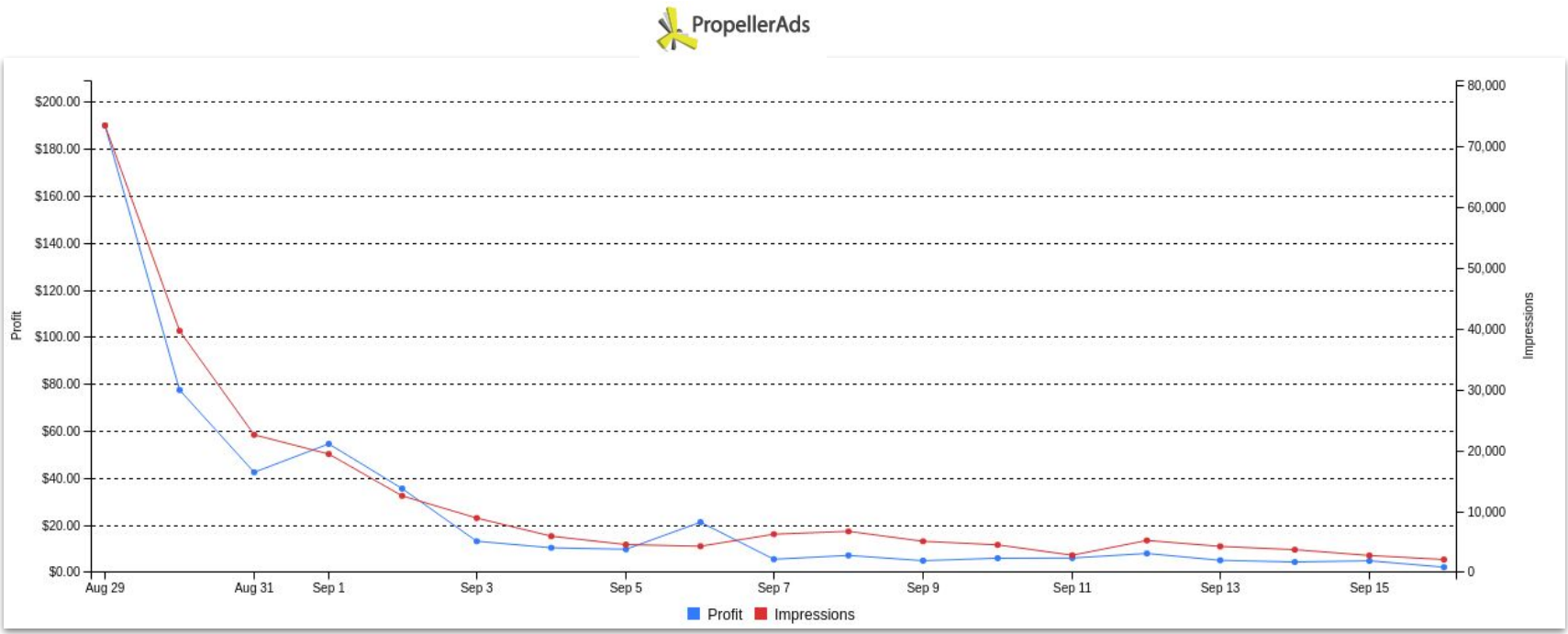
- **Google Ads** are very difficult to get approved
- Site needs to **"have more content"** most of the time
- There's a lot of optimizing needed (for auto-approval)

Some Tips

- **Start a Blog** (on the same domain)
- Make sure:
  - Blog is in sitemap.xml
  - Fix 302 redirects (temporarily moved)
  - 301 for best SEO
- Can be achieved with CloudFront Lambda@Edge
- It's hard, that's why people pay for SEO

```
'use strict';

const path = require('path')

function redirect301(url) {
    return {
        status: '301',
        statusDescription:'Moved Permanently',
        headers: {
            location: [{
                key: 'Location',
                value: url,
            }],
        },
    };
}

exports.handler = (event, context, callback) => {
    const {
        request
    } = event.Records[0].cf
    const url = request.uri;

    if (url.endsWith('/')) {
        return callback(null, request);
    }

    if (url.endsWith('/index.html')) {
        return callback(null, redirect301(url.replace('/index.html', '/')));
    }

    if (path.extname(url).length >0) {
        return callback(null, request);
    }

    return callback(null, redirect301(url +'/'));
};
```

PropellerAds are awful, but so is going broke - **Nathan Glover**



Aftermath

- **Free-tier** usage
  - Still running 1-2 workers on some spare GCP credits
- We did *just* make the money we spent back.
- Had 4 people reach out asking to have their data removed
  - We built a way for us to comply!
  - **Had a lawyer** write up our privacy policy early on
    - Remember to date it!
- We learnt so much under pressure