

We All Live In A Yellow (Serverless) Submarine

Welcome.

This is Your Captain Speaking



Luke DeWitt

WEB TEAM LEAD / DAD /
SMOKER OF VARIOUS MEATS /
SAD BLUE JAYS FAN

This is Your Captain Speaking



ARRRRR!!

Luke DeWitt

WEB TEAM LEAD / DAD /
SMOKER OF VARIOUS MEATS /
SAD BLUE JAYS FAN

This is Your Captain Speaking



Luke DeWitt

WEB TEAM LEAD / DAD /
SMOKER OF VARIOUS MEATS /
SAD BLUE JAYS FAN


RED
space™


turner


SONY
PICTURES


MTV


CN
CARTOON NETWORK


PBS
KIDS


Harper
Collins


nick


NBC


Bravo

corus.


CBC


BET


SHOWTIME


IBM


sesameworkshop


H
HISTORY

Serverless

Serverless



Serverless = Server-Less

Serverless

A cloud computing model where-by a server dynamically allocates machine resources to execute code on-demand.

Serverless



Serverless

- Also referred to FaaS (Functions as a Service)
- Originator: Zimki - 2006
- Introduced by Amazon in 2014 (AWS Lambda)

Serverless





Google Cloud Platform





Serverless











Why?

Why Serverless?

- No need to deal with f*%&ing servers

Why Serverless?

- Let’s you concentrate on what’s important to your clients and users

Why Serverless?

- Cheap
- 128MB
  - $0.000000208 / 100ms
- 1024MB
  - $0.000001667 / 100ms
- Function at 1024MB, runs 1000X at 80ms / exec
  - $0.1667 = 17 cents

Why Serverless?

- Auto-Scaling

Why Serverless?

- Less complex development

Why Serverless?

- Prototyping

Why Serverless?

- Get sh*t done faster

When?

When Serverless?

- Websites / APIs

When Serverless?

- Event Streaming w/ Pub/Sub

When Serverless?

- IOT

When Serverless?

- Continuous Integration

When Serverless?

- Image / Video Manipulation



Why?

Why Not?

Why Not Serverless?

- Kind of a pain in the ass getting started
- All of the providers are different

Why Not Serverless?

- No server = No control
- Difficult to test
- Logging and Error Handling

Why Not Serverless?

- Cost model may not make sense
- Your language may not be supported
- Execution time limits

Why Not Serverless?

- Cold start

Why Not Serverless?

- Cold start

Serverless

And then ... ?

What's Next?

- Serverless Databases
- Containers

Talk is cheap...

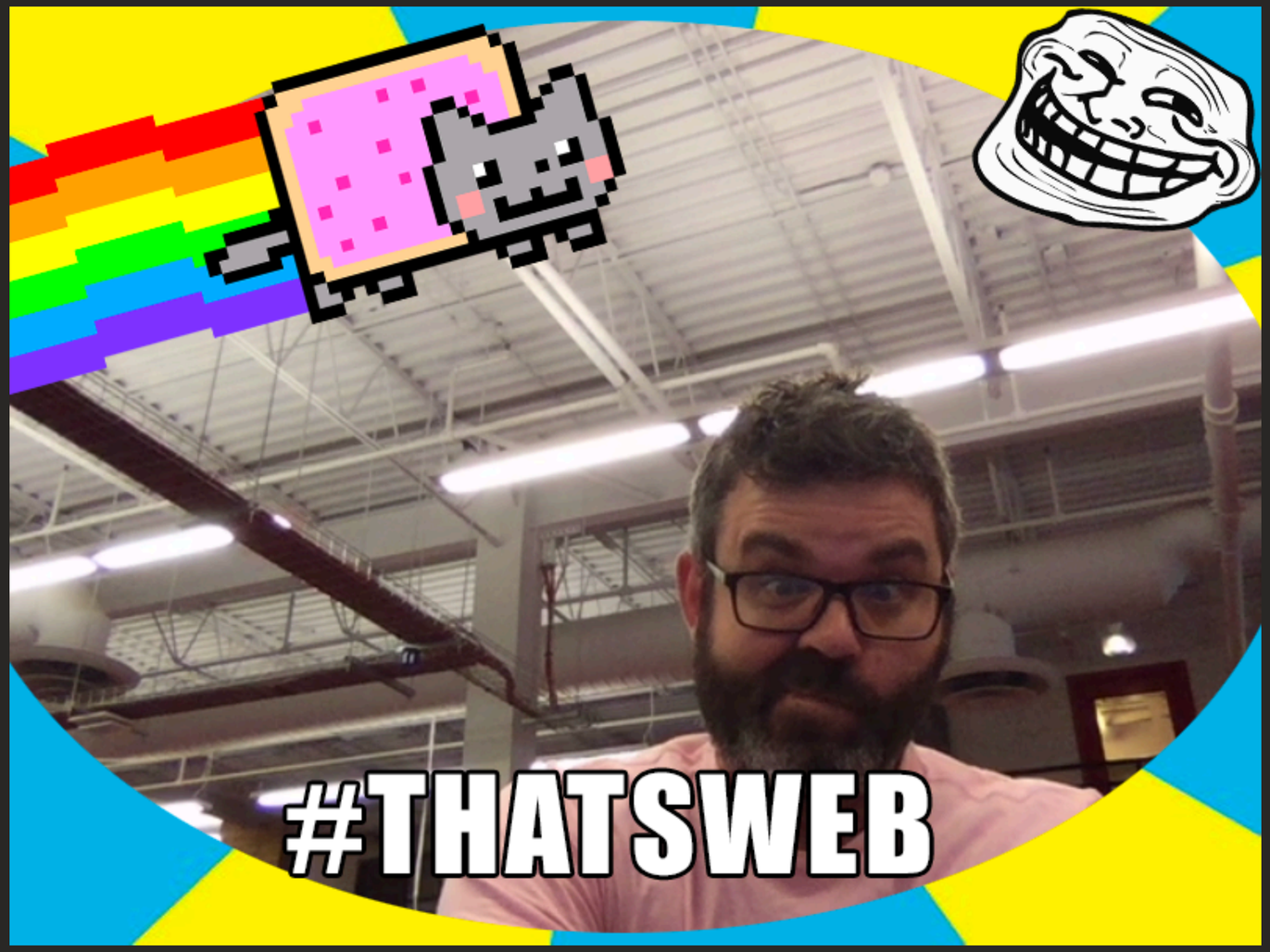
Demo time!

Serverless Demo

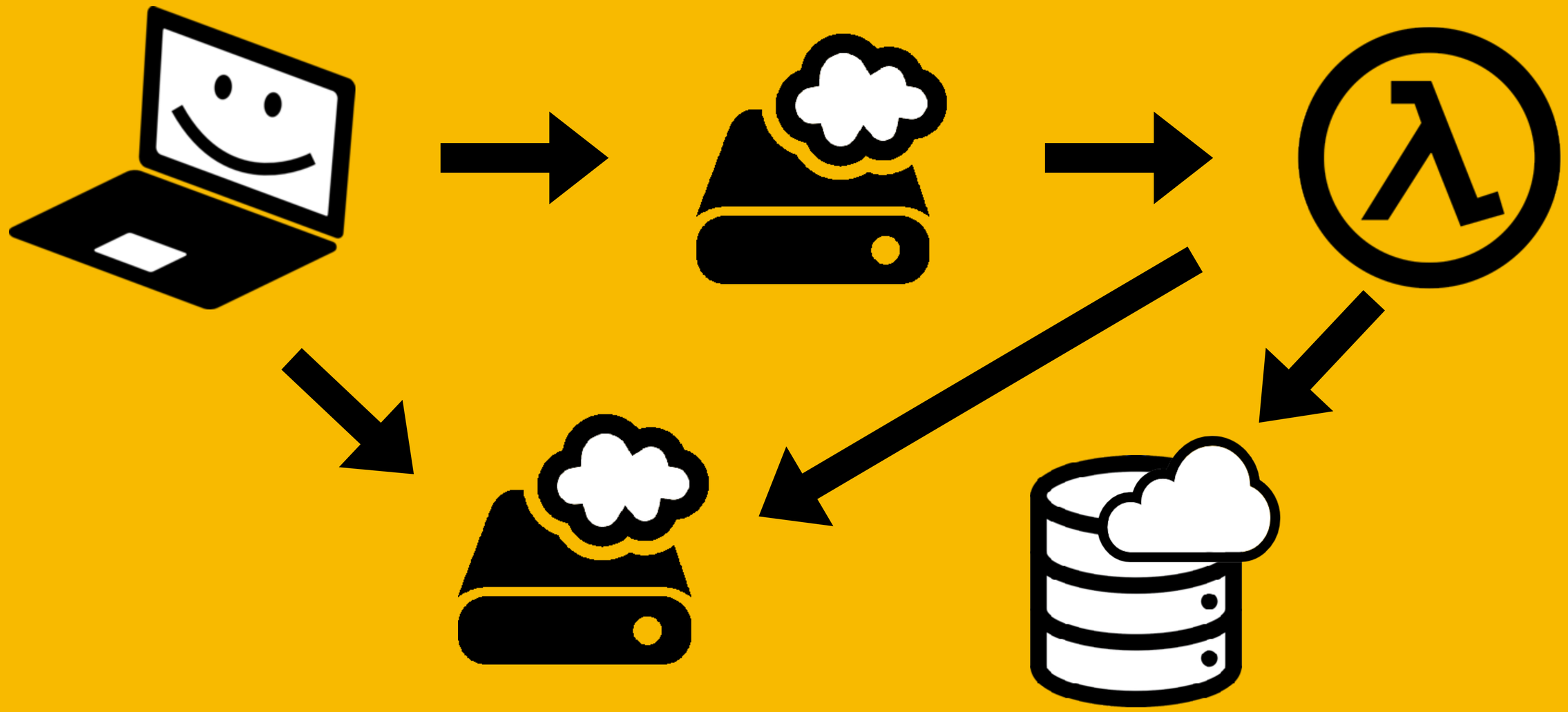
- React App
- User Uploads a Photo to S3
- Triggers Lambda Function
- Image Modified, Saved to S3
- Share to Social Media

- Meme Generator

Serverless Demo



Serverless Demo



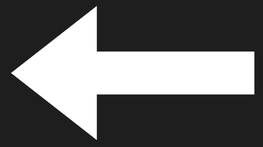

React App
NOT A REACT TALK

Upload to S3

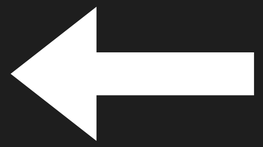
```
Amplify.configure({
  Auth: {
    identityPoolId: process.env.IDENTITY_POOL_ID,
    region: process.env.AWS_REGION
  },
  Storage: {
    bucket: process.env.UPLOAD_BUCKET,
    region: process.env.AWS_REGION
  }
});

let filename = shortid.generate();
this.setState({
  photoUrl: `//www.thatsweb.ca/uploads/${filename}.png`
});

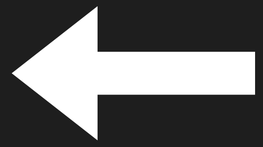
Amplify.Storage.put(
  `${filename}.png`,
  this.dataURItoBlob(this.photo.current.src),
  {
    contentType: "image/png",
    customPrefix: {
      public: process.env.UPLOAD_FOLDER
    }
  }
).then(result => this.searchForMeme());
```



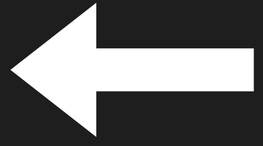
AWS Amplify



“Unique” ID

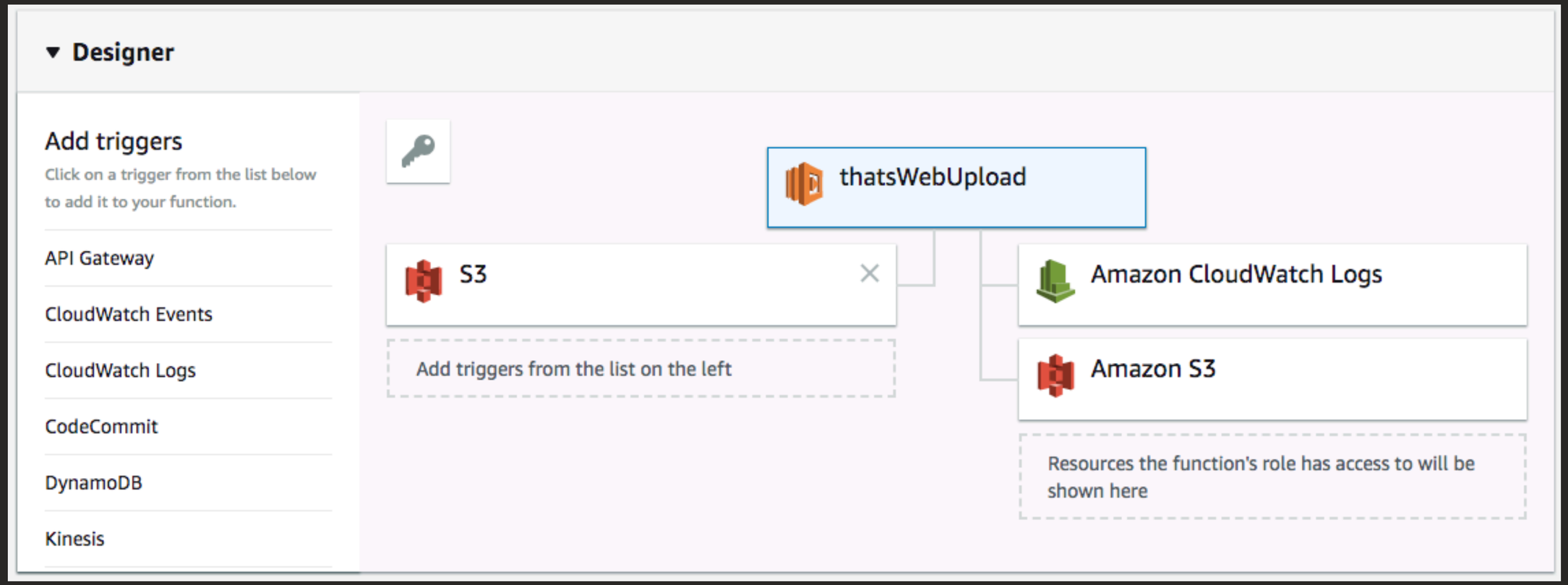


Amplify push to S3



Pull from S3

Trigger Lambda Function



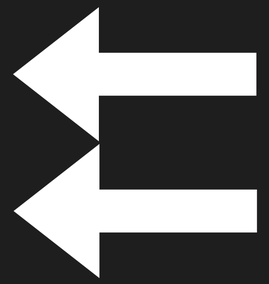
Lambda Function Code

```
const s3 = new AWS.S3();

exports.handler = (event, context, callback) => {
  const key = event.Records
    ? event.Records[0].s3.object.key
    : process.env.TEST_IMAGE;

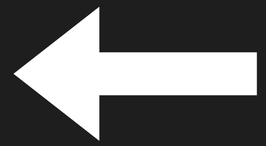
  const inputParams = {
    Bucket: process.env.TRIGGER_BUCKET,
    Key: key
  };

  // download image from s3 that triggered the lambda event
  return s3.getObject(inputParams, (err, data) => {
```



Invocation Function

Get uploaded photo



Get the image

Lambda Function Code

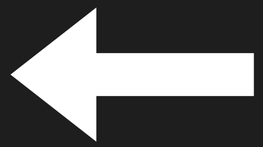
```
const input = data.Body;

// composite the meme image on top of the uploaded image
gm(input, "input.png")
  .composite("./thatsweb.png")
  .stream("png", (err, stdout, stderr) => {
    let buf = new Buffer("");

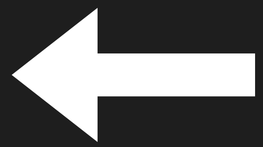
    stdout.on("data", data => {
      buf = Buffer.concat([buf, data]);
    });

    stdout.on("end", data => {
      // upload the meme to s3
      const destinationParams = {
        Bucket: process.env.DESTINATION_BUCKET,
        Key: key,
        Body: buf,
        ContentType: "image/png"
      };

      s3.putObject(destinationParams, (err, res) => {
```



Build the meme



Push to public S3

Lambda Function Code

```
const connection = mysql.createConnection({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASS,
  database: process.env.DB_NAME
});

connection.connect(function(err) {
  if (err) {
    callback(err);
  }
});

const record = {
  uuid: recordId[1],
  ip: event.Records
    ? event.Records[0].requestParameters.sourceIPAddress
    : "testing-ip"
};

connection.query("INSERT into thatsweb SET ?", record, err => {
  if (err) {
    callback(err);
  }

  connection.end(() => callback());
```

Connect to DB

Create Object

Save Record

End Lambda Function

Endpoint Polling

```
searchForMeme() {
    let img = new Image();

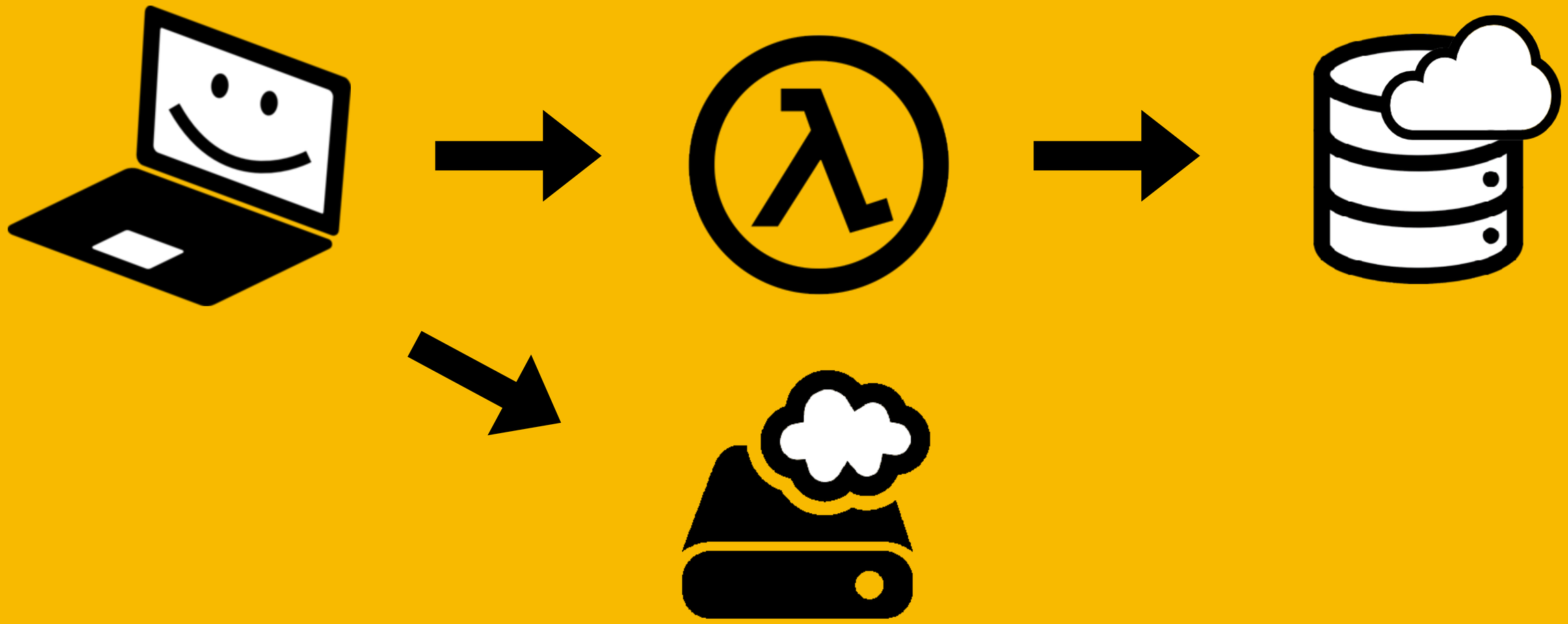
    img.onload = () => {
      this.photo.current.src = this.state.photoUrl;
      this.setState({
        stage: "share"
      });
    };

    img.onerror = () => {
      // try again in half a second...
      img = null;
      return setTimeout(this.searchForMeme, 500);
    };

    img.src = this.state.photoUrl;
  }
```

BONUS?

Fetch Flow



Fetch Memes



Get Endpoint

```
connection.query("SELECT COUNT(*) AS count FROM thatsweb", (err, results) => {
    const { count } = results[0];

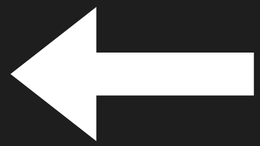
   let query = "SELECT uuid FROM thatsweb ORDER BY (id * RAND())";

    if (count > 20) {
      query += " LIMIT 20;";
    }

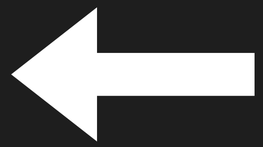
    connection.query(query, (err, results) => {
      if (err) {
        callback(err);
      }

      function shuffle(a) {
        for (let i = a.length - 1; i > 0; i--) {
          const j = Math.floor(Math.random() * (i + 1));
          [a[i], a[j]] = [a[j], a[i]];
        }
        return a;
      }

      connection.end(() =>
        callback(null, {
          memes: shuffle(results.map(({ uuid }) => uuid))
        })
      );
    });
```



Get Results



Send back memes

Fetch Memes

```
fetch("https://api.thatsweb.ca/v1/memes", {
  headers: {
    "x-api-key": process.env.API_GATEWAY_KEY
  }
})
  .then(resp => resp.json())
  .then(resp =>
    this.setState({
      memes: resp.memes
    })
  );
```

```
<div className="thatsweb-list__memes">
  {memes.map((m, idx) => (
    <MemeThumb uuid={m} key={idx} />
  ))}
</div>
```

```
const MemeThumb = ({ uuid }) => {
  return (
    <div className="thatsweb-list__meme">
      <Link to={`/memes/${uuid}`}>
        <img
          src={
            typeof uuid === "undefined"
              ? placeholder
              : `//www.thatsweb.ca/uploads/${uuid}.png`
          }
          alt="uploaded meme thumbnail"
        />
      </Link>
    </div>
  );
};
```

Your Turn

https://www.thatsweb.ca

https://www.github.com/whatadewitt/thatsweb

Thank you for coming!

Thank You!

RED space™

redspace.com / **T** (902) 444.3490

LUKE DEWITT @whatadewitt

FACEBOOK REDspace

TWITTER @theREDspace

LINKEDIN The REDspace

Thank You!

Oh, by the way, we're hiring! https://www.redspace.com/jobs