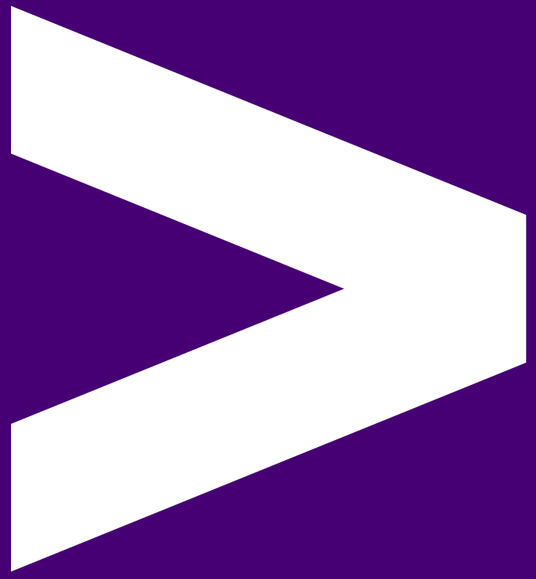
Next Gen Engineering
School of Tech - INDIA

AWS Lambda with CDK

Running functions in AWS



Overview

- Monoliths vs Microservices: Why Lambda?
- What is AWS Lambda?
- Sample Lambda application structures
- Create and run a JavaScript Lambda manually
- Create TypeScript Lambdas with CDK
- Extend our Lambda code

Objectives

- Understand the difference between monolithic and microservice architecture
- Use Lambda in the AWS Console
- Create & Invoke a Lambda
- Create a Lambda with CDK
- See the Event and Context objects
- Use environment variables and payload parameters
- Consider error handling

AWS sessions list

- AWS + Cloud intro 01 ✅ 1.5hrs
- AWS + Cloud intro 02 ✅ 1.5hrs
- AWS 01 S3 - storage (manual) ✅ 1.5hrs
- AWS 02 CDK intro - with S3 ✅ 3.0hrs
- AWS 03 Cloudfront - get files out of s3 ✅ 1.5hrs
- AWS 04 Lambda - running code ⬅ 3.0hrs
- AWS 05 Api Gateway - put an API in front of Lambda 3.0hrs
- AWS 06 Aurora Serverless Postgres - relational db 3.0hrs
- AWS 07 DynamoDB - non-relational db 3.0hrs

What is server architecture?

There are different ways we can organise and structure our backend code on the server.

If we look at two of these, Monoliths and Microservices, these lead us to the structure of a Lambda-based application.

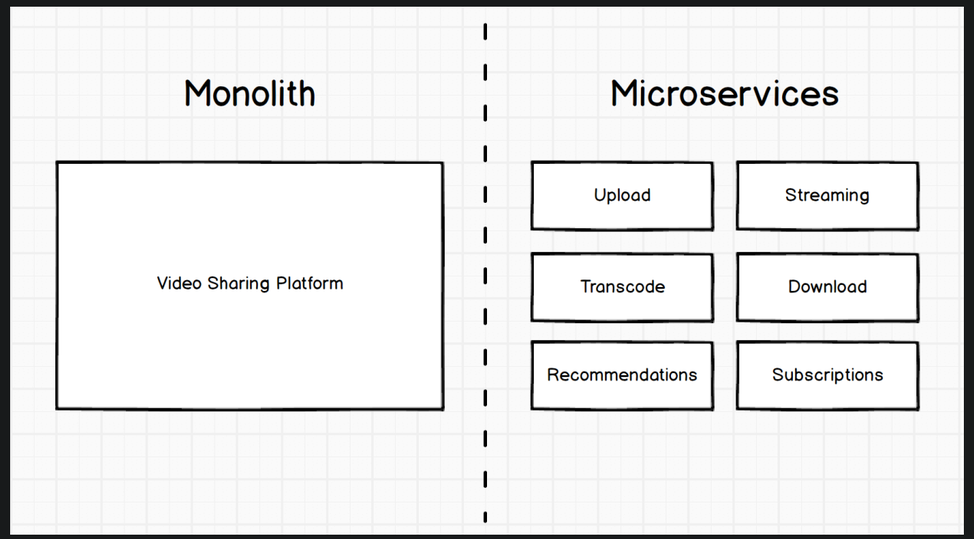
Once upon a time, software was deployed on dedicated servers as monolithic applications. This means your entire application was deployed in a single 'chunk'.

What if we had 10s or even 100s of modules in our monolith?

Monolithic codebases can be massive, with tightly coupled components, making them difficult to test and debug...

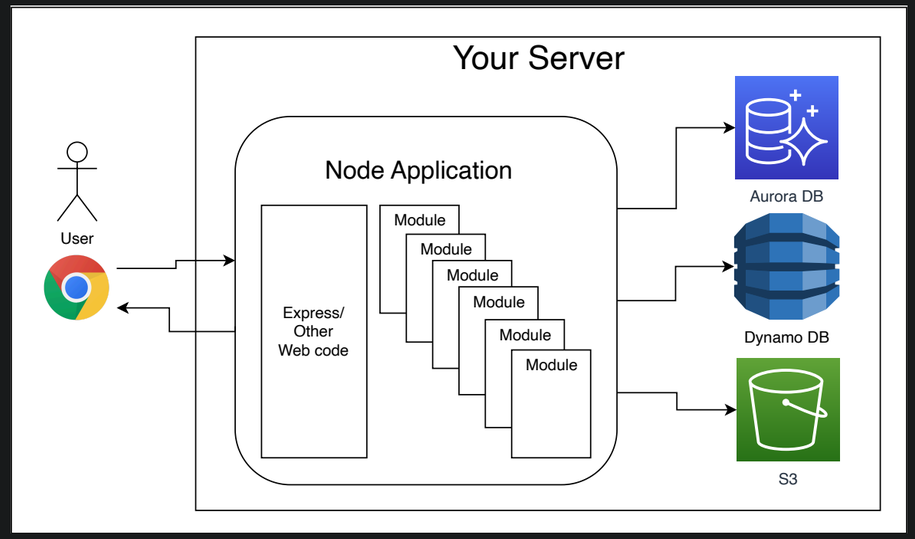
Monoliths vs Microservices

Over time, with advances in server and code technologies, we have been able to deploy smaller services that talk to each other, making the same overall set of features much more maintainable:



A 'classic' application

Whether running locally or in AWS, here is what a monolithic node app might look like:



We still see plenty of these about, however their dominance has been challenged by the rise of the microservice.

We can go further though...

We can split each module in to smaller parts that run independently and are even more granular.

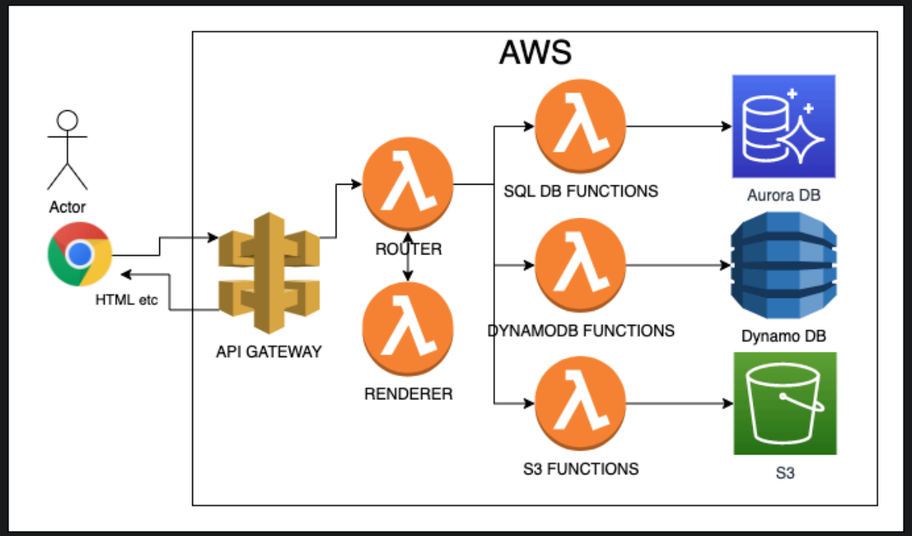
If we can deploy on the function level, we can make very granular components that are each much easier to maintain.

Lambda Functions are AWS's answer to this.

> Question: What might a Lambda application look like?

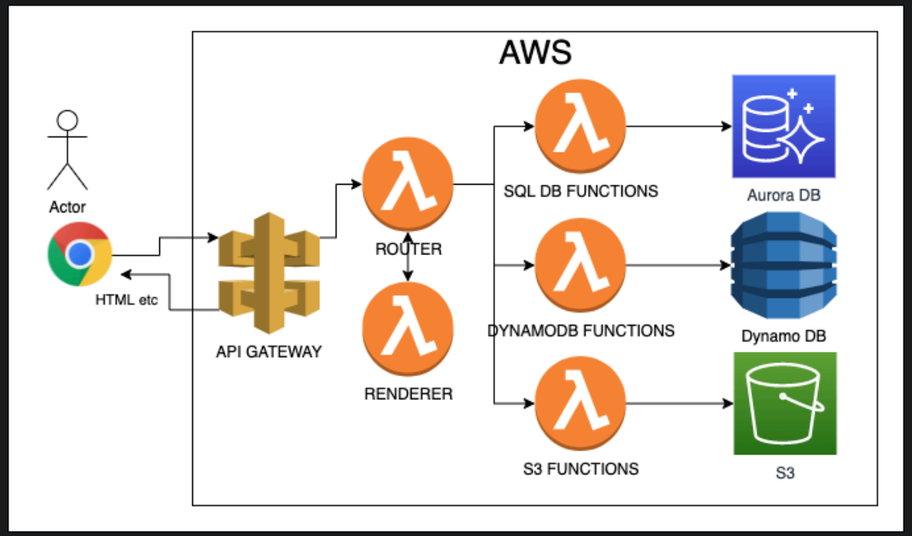
A Lambda Application

An application running in Lambda might look like this:



A Lambda Application

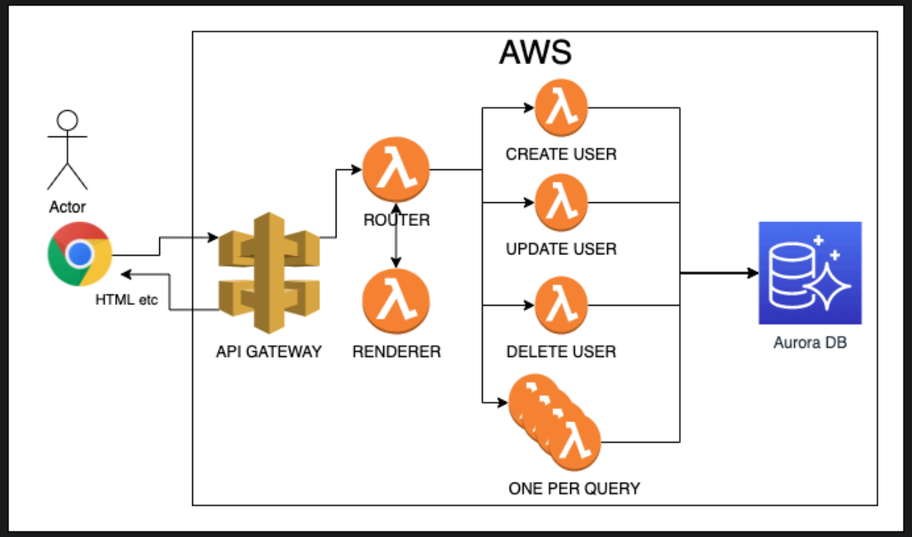
An application running in Lambda might look like this:



But wait! This still has a Lambda per 'module'... we can to go further and have a Lambda per `function`!

A Lambda Application

A Lambda per `function`!



(Other modules not shown - we would do the same for them)

Lambda per function

> By having many small functions running, we have many independently upgradeable parts, making our application very maintainable

Of course there are pros and cons - we will have to manage all these components and keep track of them.

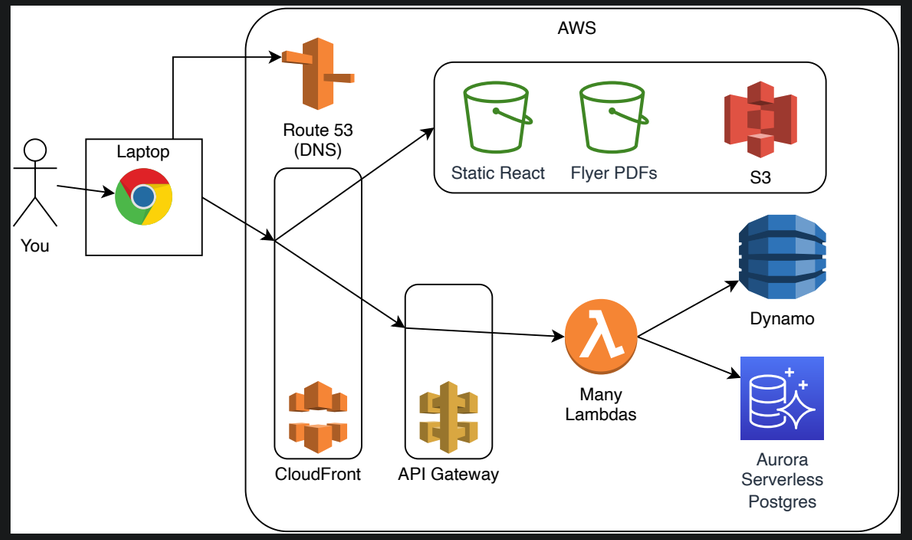
In practice, this is the same complexity as managing the interactions in any codebase, so overall the flexibility this gives us is a win.

Infini-Gigs revisited

Perhaps there's an example in the Infini Gigs architecture?

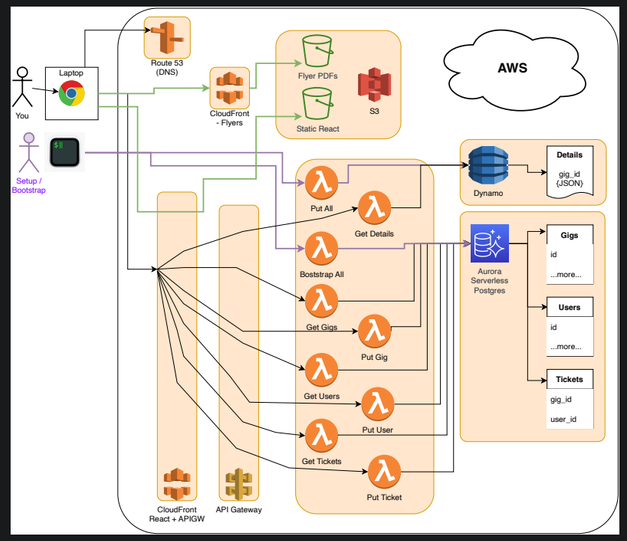
Infini Gigs Lambdas

In the fully finished InfiniGigs app we have a number of Lambdas:



Infini Gigs Lambdas

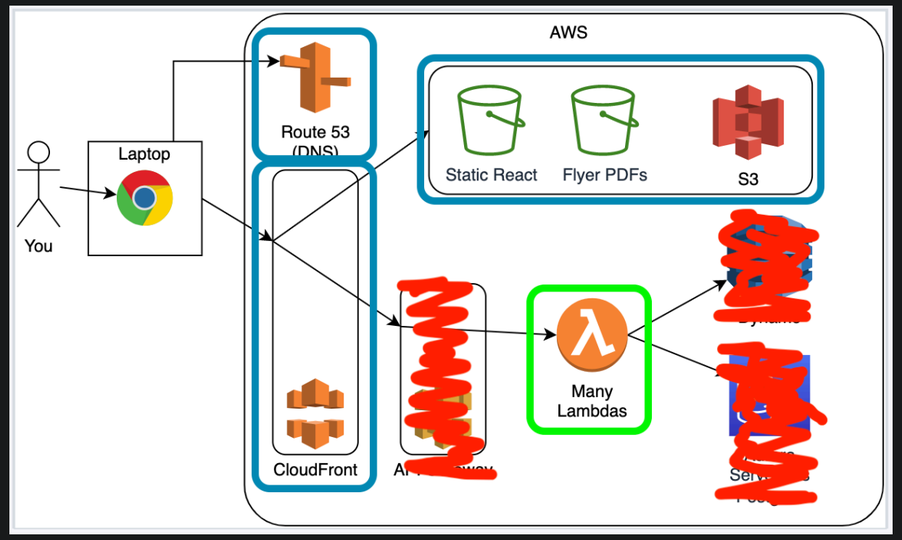
InfiniGigs has many specific Lambdas:



(This slide is here for illustration - we're not going to try and do all of that this session.)

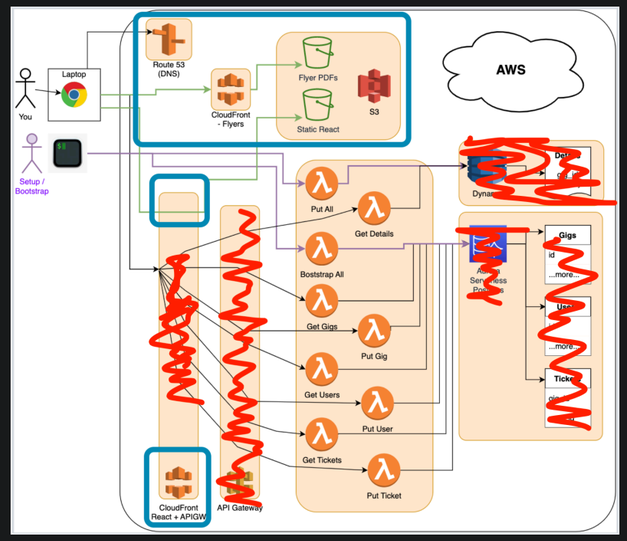
InfiniGigs revisited

In this session we are going to look only at the Lambda parts:



InfiniGigs revisited

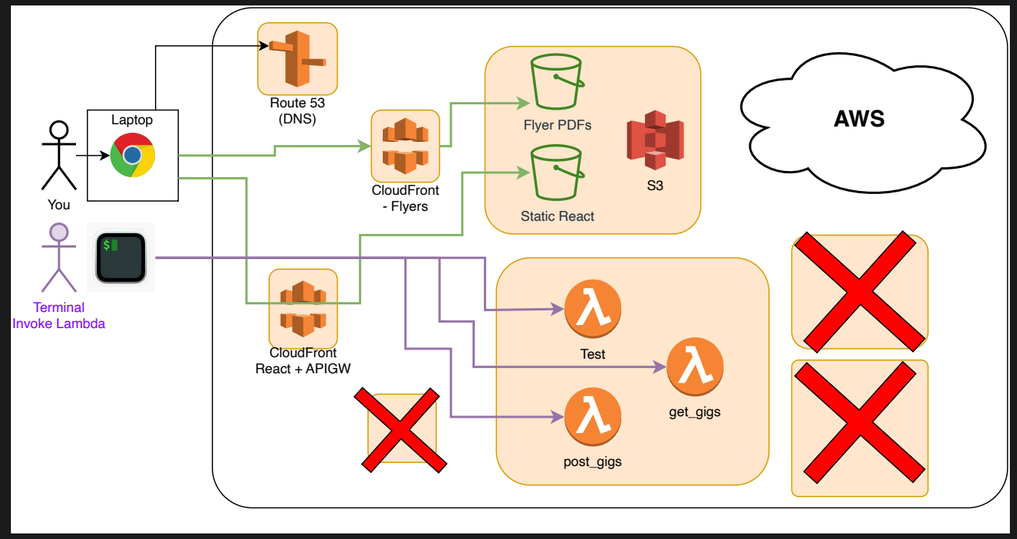
In detail that would be:



But all of that wouldn't fit in this session and needs tools we haven't looked at yet (API GW, Aurora Serverless, DynamoDB).

InfiniGigs cut down to size

As this is the Lambda introduction session, we're going to keep it basic:



In later sessions we'll add some of the missing parts.

So - what is AWS Lambda?

- A compute service (runs our code)
- Free when not being used
- On-demand only
- Event driven (something has to start it - more on this later)
- Highly available & Fault Tolerant (Multi-AZ)
- Load Balanced by AWS

And ideally

- Small functions that run quickly (for short, cheap executions)
- Non-blocking functions (i.e. they do work and put the result on a queue)

Quiz: Why choose Lambda?

> Given the previous slides, why might we choose Lambda over other options?

Quiz: Why choose Lambda?

> Given the previous slides, why might we choose Lambda over other options?

Smaller components to deploy

Quiz: Why choose Lambda?

> Given the previous slides, why might we choose Lambda over other options?

Smaller components to deploy

Can run fast as the code is small

Quiz: Why choose Lambda?

> Given the previous slides, why might we choose Lambda over other options?

Smaller components to deploy

Can run fast as the code is small

Auto-scaling managed by AWS

Quiz: Why choose Lambda?

> Given the previous slides, why might we choose Lambda over other options?

Smaller components to deploy

Can run fast as the code is small

Auto-scaling managed by AWS

No cost when not running

Quiz: Why choose Lambda?

> Given the previous slides, why might we choose Lambda over other options?

Smaller components to deploy

Can run fast as the code is small

Auto-scaling managed by AWS

No cost when not running

Cheaper IaC (less DevOps work to make and run applications)

Lambda Limits & Quotas

There are many limits applied by AWS to the use of Lambdas, due to the architecture they run on.

> What might these limits include?

Lambda Limits & Quotas

There are many limits applied by AWS to the use of Lambdas, due to the architecture they run on.

> What might these limits include?

Function Resources (CPU, Memory, Threads, etc)

Lambda Limits & Quotas

There are many limits applied by AWS to the use of Lambdas, due to the architecture they run on.

> What might these limits include?

Function Resources (CPU, Memory, Threads, etc)

Maximum concurrent executions

Lambda Limits & Quotas

There are many limits applied by AWS to the use of Lambdas, due to the architecture they run on.

> What might these limits include?

Function Resources (CPU, Memory, Threads, etc)

Maximum concurrent executions

Maximum invocation requests per Region

Lambda Limits & Quotas

There are many limits applied by AWS to the use of Lambdas, due to the architecture they run on.

> What might these limits include?

Function Resources (CPU, Memory, Threads, etc)

Maximum concurrent executions

Maximum invocation requests per Region

Storage limits (disk etc)

Lambda Limits & Quotas

There are many limits applied by AWS to the use of Lambdas, due to the architecture they run on.

> What might these limits include?

Function Resources (CPU, Memory, Threads, etc)

Maximum concurrent executions

Maximum invocation requests per Region

Storage limits (disk etc)

Limited network connections (ENIs per VPC)

What does a Lambda app look like?

When using Lambda, we have some design considerations:

- Lambdas should be stateless (the state should be stored outside the code)
- They should be short running
- They can invoke each other synchronously (blocking)
- They can send data to queues (non-blocking)
- They can be triggered (activated) by many different events
- They can effectively run any code you like (anything you could write in TS elsewhere)

Quiz: Lambda limitations?

> Given the previous slides and it being "Functions as a service", what limitations might Lambda have?

Quiz: Lambda limitations?

> Given the previous slides and it being "Functions as a service", what limitations might Lambda have?

A Memory Limit (10 GB)

Quiz: Lambda limitations?

> Given the previous slides and it being "Functions as a service", what limitations might Lambda have?

A Memory Limit (10 GB)

A code size limit (50 MB)

Quiz: Lambda limitations?

> Given the previous slides and it being "Functions as a service", what limitations might Lambda have?

A Memory Limit (10 GB)

A code size limit (50 MB)

Lumpy Autoscaling (per 500)

Quiz: Lambda limitations?

> Given the previous slides and it being "Functions as a service", what limitations might Lambda have?

A Memory Limit (10 GB)

A code size limit (50 MB)

Lumpy Autoscaling (per 500)

Per-Account concurrency limit

Quiz: Lambda limitations?

> Given the previous slides and it being "Functions as a service", what limitations might Lambda have?

A Memory Limit (10 GB)

A code size limit (50 MB)

Lumpy Autoscaling (per 500)

Per-Account concurrency limit

Cold and Warm starts

Quiz: Lambda limitations?

> Given the previous slides and it being "Functions as a service", what limitations might Lambda have?

A Memory Limit (10 GB)

A code size limit (50 MB)

Lumpy Autoscaling (per 500)

Per-Account concurrency limit

Cold and Warm starts

Lambda Warmers may be required

Quiz: Lambda limitations?

> What else might be a major Lambda limitation?

Quiz: Lambda limitations?

> What else might be a major Lambda limitation?

Node Lambdas run JavaScript, not Typescript

Quiz: Lambda limitations?

> What else might be a major Lambda limitation?

Node Lambdas run JavaScript, not Typescript

We can use JavaScript directly in the Web Console (which we will do soon)

Quiz: Lambda limitations?

> What else might be a major Lambda limitation?

Node Lambdas run JavaScript, not Typescript

We can use JavaScript directly in the Web Console (which we will do soon)

But to run TypeScript, we need to transpile...

Quiz: Lambda limitations?

> What else might be a major Lambda limitation?

Node Lambdas run JavaScript, not Typescript

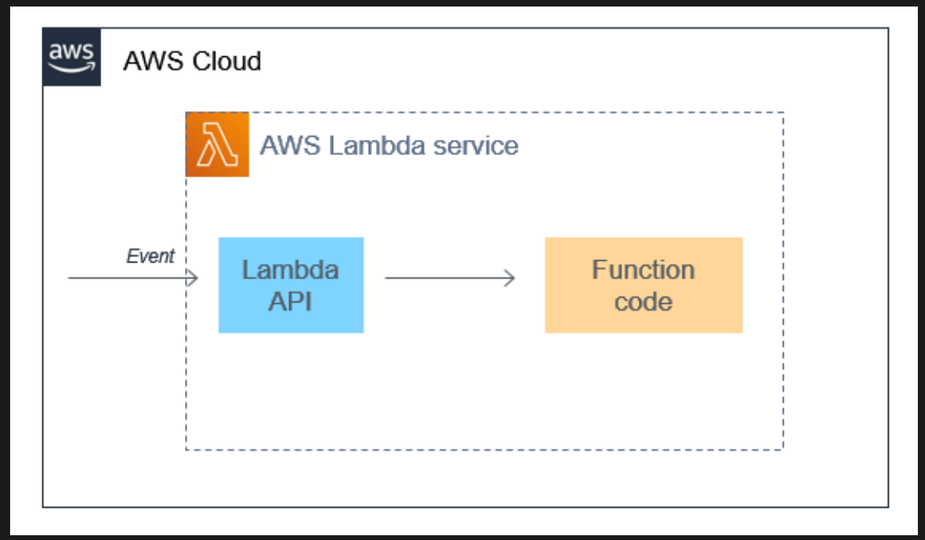
We can use JavaScript directly in the Web Console (which we will do soon)

But to run TypeScript, we need to transpile...

...which CDK can do for us (which we will do later)

How do we start a Lambda then?

If the code isn't always running, some external event has to "activate" or "trigger" it - these are referred to as `events`:



What sort of events?

In AWS there are many types of events we can register as the trigger for out Lambdas, like:

- An HTTP request
- A file being loaded into S3
- A log file being written to
- A message arriving in a queue
- A direct invocation (we'll do this later)
- Another Lambda
- And many more

Handling the event

A `handler` JavaScript function is the starting point for all Lambda invocations:

```
exports.handler = async (event) => {
    //TODO: Mode code here!
    return {
        statusCode: 200,
        body: JSON.stringify(
          { message: 'Hello from Lambda!' }),
    };
};
```

Sample script available at `examples/default-node-lambda.js`

Handling the event

A `handler` JavaScript function is the starting point for all Lambda invocations:

```
exports.handler = async (event) => {
    //TODO: Mode code here!
    return {
        statusCode: 200,
        body: JSON.stringify(
          { message: 'Hello from Lambda!' }),
    };
};
```

Sample script available at `examples/default-node-lambda.js`

We receive an `event` object, use it, and return something, usually json.

Demo: Create and run a NodeJS Lambda

We will follow some steps from the starting Node tutorial for Lambda

- The instructor will go through this step by step
- Follow along to run your own node Lambda
- Make sure you uniquely name your Lambda i.e. `yourname-first-lambda`

Task on next slide - don't start yet!

Code Along: create a Lambda - 5 mins

- Open the Lambda console
- Choose Create function
- Configure the following settings:
  - Name – `yourname-first-lambda`
  - Runtime – `Node.js 18.x`
  - Role – Choose the existing role `nja-lambda-execution-role`
- Choose Create function
- To configure a test event, choose Test
- For Event name, enter `test-event`
- Choose Save changes
- To invoke the function, choose Test

Emoji Check:

Do you understand why Lambdas need a handler (entry point)?

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😀 Yes, enough to start working on it collaboratively

Code Along: CloudWatch logs - 5 mins

All our `console.log()`s go into CloudWatch.

You get one Log Group named for your Lambda, i.e. `/aws/lambda/yourname-first-lambda`.

Within that Group you get a Log Stream per Lambda instance - so logs for new Lambda instances go in a new stream.

More details are in the docs at How CloudWatch structures logs if you want to know more.

> Let's go to CloudWatch and find our Lambda logs

Emoji Check:

Do you understand the key activities we did to run a Lambda and see the logs?

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😀 Yes, enough to start working on it collaboratively

Repeat invoking Lambdas

- Invoke your Lambda a few more times
- What is the Init and Billed duration?
- Is there a pattern?

Lambda Invocation patterns

Lambda Invocation patterns

Init duration - should be unspecified after first invocation. This is the "Cold Start".

Lambda Invocation patterns

Init duration - should be unspecified after first invocation. This is the "Cold Start".

Billed duration - likely be the longer for 1st vs 2nd invocation, should be similar for the subsequent operations (Warm start).

Lambda Invocation patterns

Init duration - should be unspecified after first invocation. This is the "Cold Start".

Billed duration - likely be the longer for 1st vs 2nd invocation, should be similar for the subsequent operations (Warm start).

There's a 15 minute cool-down where a Lambda will not be garbage collected/terminated.

Lambda Invocation patterns

Init duration - should be unspecified after first invocation. This is the "Cold Start".

Billed duration - likely be the longer for 1st vs 2nd invocation, should be similar for the subsequent operations (Warm start).

There's a 15 minute cool-down where a Lambda will not be garbage collected/terminated.

So if nothing triggers anything for 15 mins, you end up with 0 running, and it has to Cold start again

Lambda Invocation patterns

Init duration - should be unspecified after first invocation. This is the "Cold Start".

Billed duration - likely be the longer for 1st vs 2nd invocation, should be similar for the subsequent operations (Warm start).

There's a 15 minute cool-down where a Lambda will not be garbage collected/terminated.

So if nothing triggers anything for 15 mins, you end up with 0 running, and it has to Cold start again

You get one Log Group per Lambda and one Log Stream per Lambda instance - so logs for new Lambda instances go in a new stream.

Code Along: CloudWatch again - 5 mins

From the AWS docs:

When a Lambda Function is invoked, the AWS Lambda service routes the request to a Lambda Function Instance. It launches a new Instance if:

- No Lambda Function instances are running, or
- The handler code or configuration has changed, or
- All running instances are currently servicing other requests

Each Lambda Function Instance logs to a distinct Log Stream within the named Log Group.

> Let's go to CloudWatch and find our Lambda logs

Node Handler Functions

The `handler` function is your entrypoint - this activates the rest of your code. Following Lambda best practice, we make separate function(s) to do work, i.e:

```
exports.handler = async function(event) {
    doSomethingElse(event)
    return {
        'status': 200,
        'result': 'ok'
    }
}

function doSomethingElse(event) {
    console.log(`EVENT: ${JSON.stringify(event)}`)
    console.log("more code here")
}
```

Task: Update the Handler - Breakout 5 mins

Update your Lambda to use another function to log out sections of your event object.

- The instructor will show you what this would look like
- Use the example file `./examples/handler-example-01.js`
- You will need to use the `Deploy` button to save the updates
- Then invoke it with your test event
- And check the CloudWatch logs - we redeployed so there should be a new stream.

Emoji Check:

On a high level, do you get the idea of running small bits of code or "Lambda Functions" in AWS?

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😀 Yes, enough to start working on it collaboratively

Layer 8 issues anyone?

Layer 8 issues anyone?

Is making a whole system by hand in the Console a good idea?

Layer 8 issues anyone?

Is making a whole system by hand in the Console a good idea?

Making all this stuff by hand is very error prone!

Layer 8 issues anyone?

Is making a whole system by hand in the Console a good idea?

Making all this stuff by hand is very error prone!

Can't we use CDK to make this reliable?

Layer 8 issues anyone?

Is making a whole system by hand in the Console a good idea?

Making all this stuff by hand is very error prone!

Can't we use CDK to make this reliable?

Well, of course we can!

Infini gigs to the rescue!

> Lets get gigging...

Code Along - InfiniGigs files!

Download and extract the `./exercises` zip for this session...

> Make sure it's the `aws-iac-04-cdk-lambda` zip file!

This is a cut-down version like the AWS 03 Cloudfront session, with Flyers and a Client.

Code along - folder check!

Open a terminal in the extracted `./exercises` folder for this session:

- Run `pwd`
  - you should see the right `aws-iac-04-cdk-lambda/exercises` folder path
- Run `ls -la`
  - You should see the `gig-flyers`, `client` and `cdk` folders
- Open this set of files in VS Code

Code Along - Check your env vars

Run the following in your terminal:

```
echo $GIGS_STACK_NAME
```

And make sure it gives you some output.

Speak now if it does not - we need this!

The Infini-Gigs front end

> In this session, we have a bare-bones Infini Gigs front end - until we add more in following sessions, it wouldn't fully "work" anyway!

To build it, change to the `aws-iac-04-cdk-lambda/exercises/client` folder and run this. Make sure you run this in git bash:

```
chmod a+x build.sh
./build.sh
ls ./dist/*
```

Code along - install packages

In the terminal in the `./exercises` folder for this session:

Run `./install.sh` This will run `npm i` in the following locations:

- client
- cdk
- cdk/functions

Code Along - initial deploy

To get everyone's stacks to the same starting point let's deploy now from the `./exercises/cdk` folder:

```
cd cdk # if you're not there already
aws-logon # or use the full version
npx cdk deploy
```

Where does the TS code go?

In order to make CDK deploy a Lambda, we first need some code to use!

> By convention we will put this code in a `functions` folder inside the `cdk` folder, i.e. `./exercises/cdk/functions/*.ts`

Caveat: In the `exercises/cdk/functions` folder, we have provided the usual TS boilerplate of a `package.json` and `tsconfig.json` file, so CDK knows what to do. It's not completely magic!

A TypeScript Lambda

From the AWS docs here -> "Async Handlers", like this:

```
import { Context, APIGatewayProxyResult, APIGatewayEvent }
  from 'aws-lambda'

export const lambdaHandler = async
  (event: APIGatewayEvent, context: Context):
    Promise<APIGatewayProxyResult> => {
    console.log(`Event: ${JSON.stringify(event, null, 2)}`)
    return {
        statusCode: 200,
        body: JSON.stringify({
            message: 'hello world',
        }),
    }
}
```

Time to make our own code

> Now we know what sample TS Lambda code looks like, we can start making our own.

In the provided `./exercises/cdk/` folder, we have the skeleton of our CDk stack. This should be familiar from previous sessions.

Code Along - standard imports

Put the default imports we need at the top of file `./exercises/cdk/functions/utility-functions.ts`:

```
import { Context, APIGatewayProxyResult, APIGatewayEvent }
  from 'aws-lambda'
```

Code Along - Lambda code

Let's add a function to return a list of gigs. Put this under the imports in the same file:

```
export const gigsListHandler =
  async (event: APIGatewayEvent, context: Context):
    Promise<APIGatewayProxyResult> => {
  console.log('gigsListHandler invoked')
  return {
    status: 'ok',
    list: [ 'Rolling Stones', 'The Doors', 'The Beatles' ],
  }
}
```

Next, some CDK

That has defined some code for us to run - next we can add the CDK constructs we need to deploy that code.

> As mentioned before, CDK will do the transpilation for us - by using the same construct we would for JavaScript, CDK will detect the use of a TypeScript file and fix up the rest, as we have provided the `package.json` and `tsconfig.json` files

Code Along - Lambda import

We need to use the `lambda` and `node` CDK utilities, so import these at the top of your `./exercises/cdk/lib/*.ts` file:

```
import * as lambda from 'aws-cdk-lib/aws-lambda'
import * as nodejs from 'aws-cdk-lib/aws-lambda-nodejs'
```

Interjection - CDK Bundling

CDK will 'bundle" our code and constructs into a deployable artefact.

The Node Lambda runtime includes the default `aws-sdk` library, so we do not need CDK to include this for us.

We can tell CDK a list of the modules that are external to our code, i.e. already available in the runtime.

More info can be found in the AWS CDK docs.

Code Along - CDK Bundling

Add this to the end of your constructor in the lib file:

```
const bundling = {
  externalModules: ['aws-sdk'],
}
```

We will use this to help configure all our Node Lambdas.

Code Along - Lambda deployment

We need the construct to deploy a new Lambda function - this goes at the end of your `constructor`:

```
const gigsListLambda = new nodejs.NodejsFunction(this,
  'gigs-list-lambda',
  {
    functionName: `${props.subDomain}-gigs-list-lambda`,
    runtime: lambda.Runtime.NODEJS_16_X, // the latest
    entry: './functions/utility-lambdas.ts', // filename
    handler: 'gigsListHandler', // function name
    bundling, // our externals list
  })
```

Code Along - outputs

So we can confirm what the Lambda name is, lets add an output for that:

```
new cdk.CfnOutput(this, 'GigsListLambdaName', {
  value: gigsListLambda.functionName,
})
```

If we had not specified the name, CDK would auto-generate an incomprehensible one.

Code Along - Lambda deployment

Time to redeploy!

Run

- `aws-logon` or the full thing
- `npx cdk deploy`

Invoking our Lambda

So far we have only been invoking our Lambda from the console using the test functionality, but we also want to know how to trigger it from our own laptop.

To do this we can use the `awscli` to run a Lambda specific command:

- `aws lambda invoke [arguments]` - call a specified Lambda with the provided event data

We will now try this out on the command line (next slide) - luckily we made the Lambda Function name deterministic and unique.

Code Along: Invoke Lambda from CLI

To invoke your Lambda we can run the following:

```
aws lambda invoke --function-name YOUR_LAMBDA_NAME \
    --payload '{"gigLocation": "Leeds", "gigId": "007"}' \
    --cli-binary-format raw-in-base64-out \
    \
    invoke-output.txt
```

An example is in `./examples/invoke-lambda.sh` - you can open this in a new terminal and run it with:

- `./invoke-lambda.sh YOUR_LAMBDA_NAME`

Invocation results

Hopefully we all see something like this?

```
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}

{"status":"ok",
 "list": ["Rolling Stones","The Doors","The Beatles"]}
```

Code along - logs in AWS

All our Lambda logs go into CloudWatch, AWS's logging service. Let's go find them:

- Open the AWS (web) Console
- Go to the CloudWatch service
- Find your Log Group
- Find your invocation logs (in the unique Log Stream)
- See your `console.log()`s!

Emoji Check:

How do we feel, on a high level, about our Lambdas and the deployment?

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😀 Yes, enough to start working on it collaboratively

The Event object

Some of you will have spotted an object called `event` - but what is this?

- The event object includes data you can use to drive your logic (remember your test invocation payload/event contained what you sent to your Lambda)
- Depending on what 'activates' your Lambda however this event might be different
- The Lambda docs from AWS don't have examples as they vary so much between services.

Our Event object - sample

As ours comes direct from our invocation, it only contains the object we sent it:

```
{
  "gigLocation": "Leeds",
  "gigId": "007"
}
```

But we haven't logged this yet in our Lambda, so we can't see it... yet.

Event objects vary

Different triggers send different data - An event that's been sent from API Gateway looks very different to ours:

```
{
    "resource": "/",
    "path": "/",
    "httpMethod": "GET",
    "requestContext": { "path": "/Prod/", ... },
    "headers": { "accept": ... },
    "multiValueHeaders": { ... },
    "queryStringParameters": null,
    ...
    "stageVariables": null,
    "body": "{\"key1\":\"value1\",\"key2\":\"value2\"}",
    "isBase64Encoded": false
}
```

An SQS event object

And an event from the Simple Queue Service is different again:

```
{ "Records": [
    {
    "messageId": "19dd0b57-b21e-4ac1-bd88",
    "receiptHandle": "MessageReceiptHandle",
    "body": "Hello from SQS!",
    "attributes": {
        "ApproximateReceiveCount": "1",
        ...
    },
    "messageAttributes": {},
    "eventSource": "aws:sqs",
    "awsRegion": "ap-south-1",
    ...
  ], ...
}
```

Caller aware code

> Our Lambda code has to be "caller aware" - that is, we will have to code it to expect specific shapes of json from the events that trigger it

And

> The `event.body` is typically a string, or stringified-json, so we will very likely need to `JSON.parse()` it back into a JS object.

The Context object

We just looked at the event object. There is second parameter object that we are given called `context`.

This contains metadata relating to our specific invocation which can be pulled out if needed:

- Has useful info like `logGroupName` and `functionVersion`
- `getRemainingTimeInMillis()` is the remaining time until the max timeout (15 mins or less)
- For reference, you can see what we get in node in the AWS docs here

Task: Add logging - Breakout 10 mins

Go back to your `gigsListHandler` function:

- Your function params are the `event` and the `context`, with their types
- Add a utility function that accepts both of those
- In the new function, `console.log()` the event object
- Also `console.log()` the context object
- Re-deploy your cdk
- Invoke your Lambda using the script
- Check the logs in CloudWatch

In case you get stuck there is a sample at `./examples/handler-gigs-example-01.ts`

Our Event object - sample

```
{
  "gigLocation": "Leeds",
  "gigId": "007"
}
```

Our Context object - sample

```
{
    "callbackWaitsForEmptyEventLoop": true,
    "functionVersion": "$LATEST",
    "functionName": "markm-gigs-gigs-list-lambda",
    "logGroupName": "/aws/lambda/markm-gigs-gigs-list-lambda",
    "memoryLimitInMB": "128",
    ...
    "awsRequestId": "de35b98c-6279-4c76-bb21-03be5c8674c4"
}
```

Emoji Check:

Do you understand the key data of the Event and Context objects?

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😀 Yes, enough to start working on it collaboratively

Environment variables

Our Lambdas can be configured with environment variables to aid configuration.

Environment variables

Our Lambdas can be configured with environment variables to aid configuration.

Question: What sort of things could go in env vars?

Environment variables

Our Lambdas can be configured with environment variables to aid configuration.

Question: What sort of things could go in env vars?

Question: What sort of things should go in env vars?

Environment variables

Our Lambdas can be configured with environment variables to aid configuration.

Question: What sort of things could go in env vars?

Question: What sort of things should go in env vars?

Question: What sort of things should not go in env vars?

Environment variables

Our Lambdas can be configured with environment variables to aid configuration.

Question: What sort of things could go in env vars?

Question: What sort of things should go in env vars?

Question: What sort of things should not go in env vars?

- We will add one and use it in our code (next slides)

Env vars in CDK

We can configure these in the construct properties:

```
const gigsListLambda = new nodejs.NodejsFunction(this,
  'gigs-list-lambda',
  {
    // ... more ...
    environment: {
      FAVOURITE_GIG: 'Duran Duran in Leeds'
    }
  })
```

Using env vars

In our Lambda handler we can update our code to use the value:

```
console.log(`Best gig is ${process.env.FAVOURITE_GIG}`)
```

Then when invoked we will see this in the logs in CloudWatch.

Task - Breakout 10 mins

- Update the `gigsListHandler` code
  - Add code to log `process.env.FAVOURITE_GIG`
- Update the cdk construct for `gigsListLambda`
  - Add the env var like so:
  - `environment: { FAVOURITE_GIG: 'Duran Duran in Leeds' }`
- Re-deploy
- Re-test (invoke on the command line)
- Check the logs in CloudWatch

Sample: Env var logged

You code could look something like `examples/handler-gigs-example-02-env-var.ts`.

In Cloudwatch your new logs should look something like this:

```
INFO    EVENT: {"key1":"value1","key
INFO    CONTEXT: {"callbackWaitsFor
INFO    Best gig is Duran Duran
```

Emoji Check:

Do you manage to add and use an env var?

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😀 Yes, enough to start working on it collaboratively

The payload

Inside our event JSON is the `payload` that was sent in.

The payload

Inside our event JSON is the `payload` that was sent in.

Question: What might we put in a payload?

The payload

Inside our event JSON is the `payload` that was sent in.

Question: What might we put in a payload?

Query parameters or url parameters

The payload

Inside our event JSON is the `payload` that was sent in.

Question: What might we put in a payload?

Query parameters or url parameters

Resources ids (user id, player id, pokemon id, cart id)

The payload

Inside our event JSON is the `payload` that was sent in.

Question: What might we put in a payload?

Query parameters or url parameters

Resources ids (user id, player id, pokemon id, cart id)

I.e. information that varies between calls or is volatile

Using the payload

We'll now add code to use keys and values from the `payload`.

Given our direct invocation, we can grab this directly as it is available on the event.

- Other event sources have arbitrarily complex structures to use

Using the payload

We can have some code to log out the keys of the payload in the event:

```
const gigLocation = event.gigLocation || 'NOT_SET'
const gigId = event.gigId || 'NOT_SET'
console.log(`PARAMS: gigId = ${gigId}`)
console.log(`PARAMS: gigLocation = ${gigLocation}`)
```

See also file examples/handler-example-03-params.js

Breakout: Using the payload - 15 mins

Now we will add a new `postGigsHandler` function which captures the payload, and the construct to deploy it.

(Details on next two slides)

Breakout part 1/2

- Make a new handler function `postGigsHandler`
- In it, log that the function was called
- Return a basic `{ status: 'ok' }` response
- Add the code below to log the keys of the event

```
const gigId = event.gigId || 'NOT_SET'
const gigLocation = event.gigLocation || 'NOT_SET'
console.log(`PARAMS: gigId = ${gigId}`)
console.log(`PARAMS: gigLocation = ${gigLocation}`)
```

Breakout part 2/2

Then:

- Add a new node CDK construct called `postGigsLambda` with id `post-gigs-lambda`
- Add an output of `postGigsLambda.functionName`
- Invoke your Lambda with `./invoke-lambda.sh <lambda-name>`
- Check the logs in Cloudwatch

Sample code

You code might look like file `examples/handler-gigs-example-03-params.ts`

Emoji Check:

Do you understand that Lambda payloads are how we get volatile data in?

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😀 Yes, enough to start working on it collaboratively

Leaking Data

Leaking Data

What happens if we get an error from a database with data like "can't insert card id 1234-5678-9012-1234" in it?

Leaking Data

What happens if we get an error from a database with data like "can't insert card id 1234-5678-9012-1234" in it?

How a about an error like "User name abcde or pwd ***** not valid"?

Leaking Data

What happens if we get an error from a database with data like "can't insert card id 1234-5678-9012-1234" in it?

How a about an error like "User name abcde or pwd ***** not valid"?

We have to make sure that we don't leak this data back to the client.

Leaking Data

What happens if we get an error from a database with data like "can't insert card id 1234-5678-9012-1234" in it?

How a about an error like "User name abcde or pwd ***** not valid"?

We have to make sure that we don't leak this data back to the client.

(This applies not just to Lambdas, but APIs in general.)

Node Errors

In Lambda executions, any uncaught errors will terminate our execution.

If so, clients will have to handle unpredictable results.

We should always catch any errors and return a suitable HTTP Code (5xx) and meaningful message to the caller.

What about both errors and not leaking data?

> We need to combine both not leaking information and handling errors gracefully.

Node Errors - example

So we need to always do this:

- Add a try-catch block to the main method
- Log the real error for ourselves
- Always return a custom response so we don't leak data - never return the actual error!

Sample file at `examples/bad-lambda.ts` - open this now.

Task - breakout 10 mins

- Add a new Lambda `badHandler`
- Include a `try-catch` block
- Make it throw an error when called i.e. `throw new Error('DB called failed')`
- Log the real error in the `catch` block
- In the `catch`, send a "please try again" message back to the caller
- Add a new `NodejsFunction` construct in the CDK, and a `CfnOutput` for it's name
- Re-deploy
- Test it (i.e. invoke it)
- Check the response is sanitised
- Check the CloudWatch logs have the real error

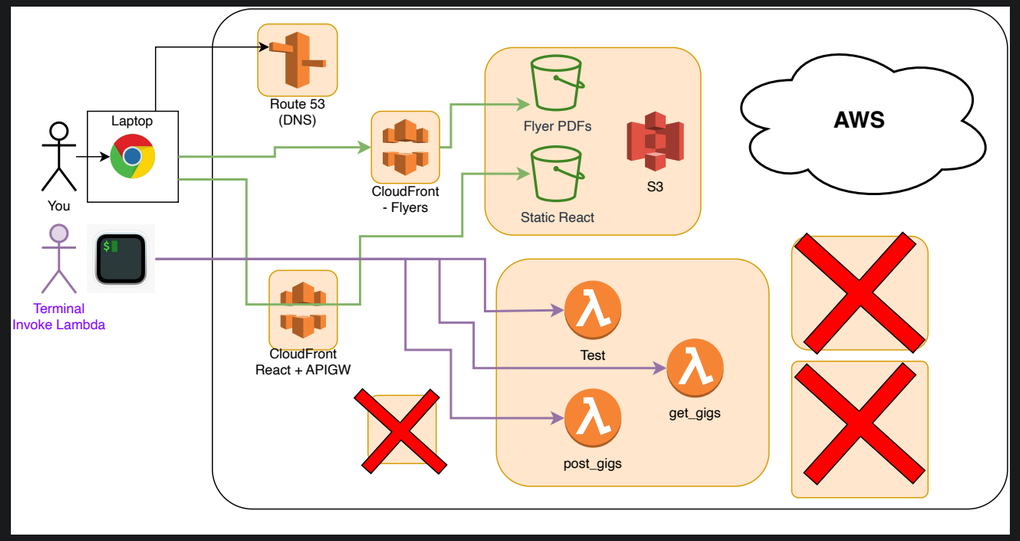
Emoji Check:

On a high level, do you understand not leaking data to the caller by catching the real error?

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😀 Yes, enough to start working on it collaboratively

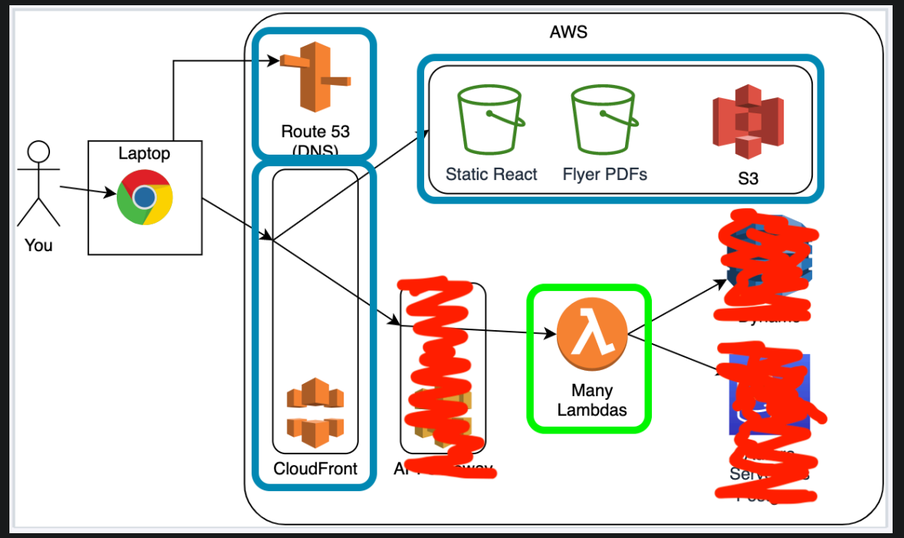
What we just made

Well done! We didn't make all the Lambdas yet - but we started 😀



What's missing?

Invoking Lambdas directly is one thing - and a valid use case - but in our "real world" InfiniGigs example we want, for example, our client code running in our browsers to be able to access some sort of "web based api"... that in this diagram we crossed out...



...this is coming in the following sessions - you may notice we didn't use the UI at all in this session 😀

Overview - recap

- Monoliths vs Microservices: Why Lambda?
- What is AWS Lambda?
- Sample Lambda application structures
- Create and run a JavaScript Lambda manually
- Create TypeScript Lambdas with CDK
- Extend our Lambda code

Objectives - recap

- Understand the difference between monolithic and microservice architecture
- Use Lambda in the AWS Console
- Create & Invoke a Lambda
- Create a Lambda with CDK
- See the Event and Context objects
- Use environment variables and payload parameters
- Consider error handling

Further reading

- AWS docs - BUilding Apps with CDK
- AWS docs - How CloudWatch structures logs
- AWS docs - What is AWS Lambda
- AWS docs - Lambda core concepts
- AWS docs - Lambda function handler in TypeScript
- AWS Blog: React & Lambda & API Gateway
- AWS Blog: 10 Things Serverless Architects Should Know
    - AWS re:Invent 2018: A Serverless Journey: AWS Lambda Under the Hood (SRV409-R1) - 1 hour
- Introduction to AWS Lambda & Serverless Applications - 1 hour

Emoji Check:

On a high level, do you think you understand the main concepts of this session? Say so if not!

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😀 Yes, enough to start working on it collaboratively