School of Tech
part of accenture

AWS 06 - Lambda with IaC

School of Tech
part of accenture

AWS sessions list

- AWS 01 AWS + Cloud Intro ✅ *1.5hrs*
- AWS 02 AWS CLI Setup ✅ *1.5hrs*
- AWS 03 S3 Storage (Console) ✅ *1.5hrs*
- AWS 04 CloudFormation Intro + S3 Storage (IaC) ✅ *1.5hrs*
- AWS 05 Lambda Intro ✅ *1.5hrs*
- AWS 06 Lambda (IaC) ⬅ *1.5hrs*
- AWS 07 Redshift (IaC) *1.5hrs*
- AWS 08 EC2 (IaC) + Grafana setup *1.5hrs*

School of Tech
part of accenture

Overview

- Lambda as compute in AWS
- Packaging with CloudFormation
- Deployment with CloudFormation
- Event Triggers for Lambdas (with S3)

School of Tech
part of accenture

Learning Objectives

- How to add a lambda function using CloudFormation
- How to add a trigger between the S3 bucket and the Lambda
- How to process the incoming event in your Lambda function

Lambda & CloudFormation

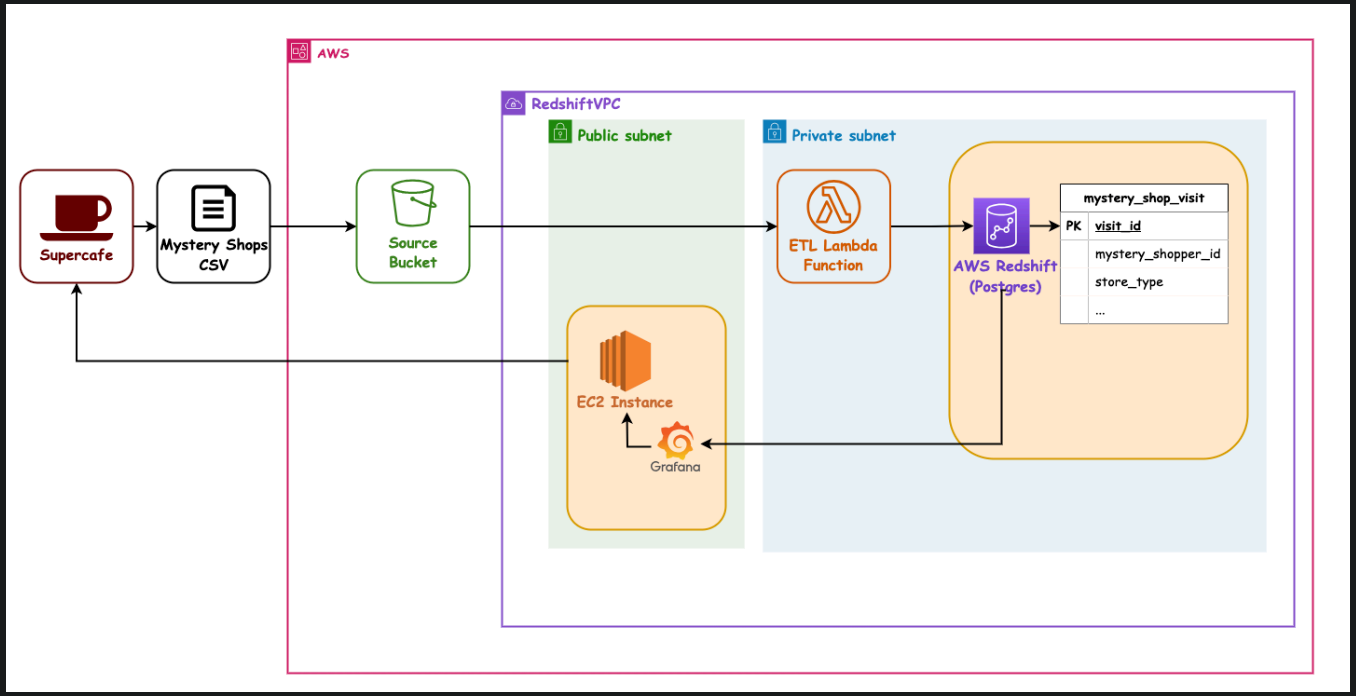
This...

...and this

School of Tech
part of accenture

Proposed Pipeline Architecture

Let's revisit our Mystery Shopper target setup:



School of Tech
part of accenture

Our next user story (same as last session)

`As a` SuperCafe senior manager

`I want` the Mystery Shopper data processed automatically

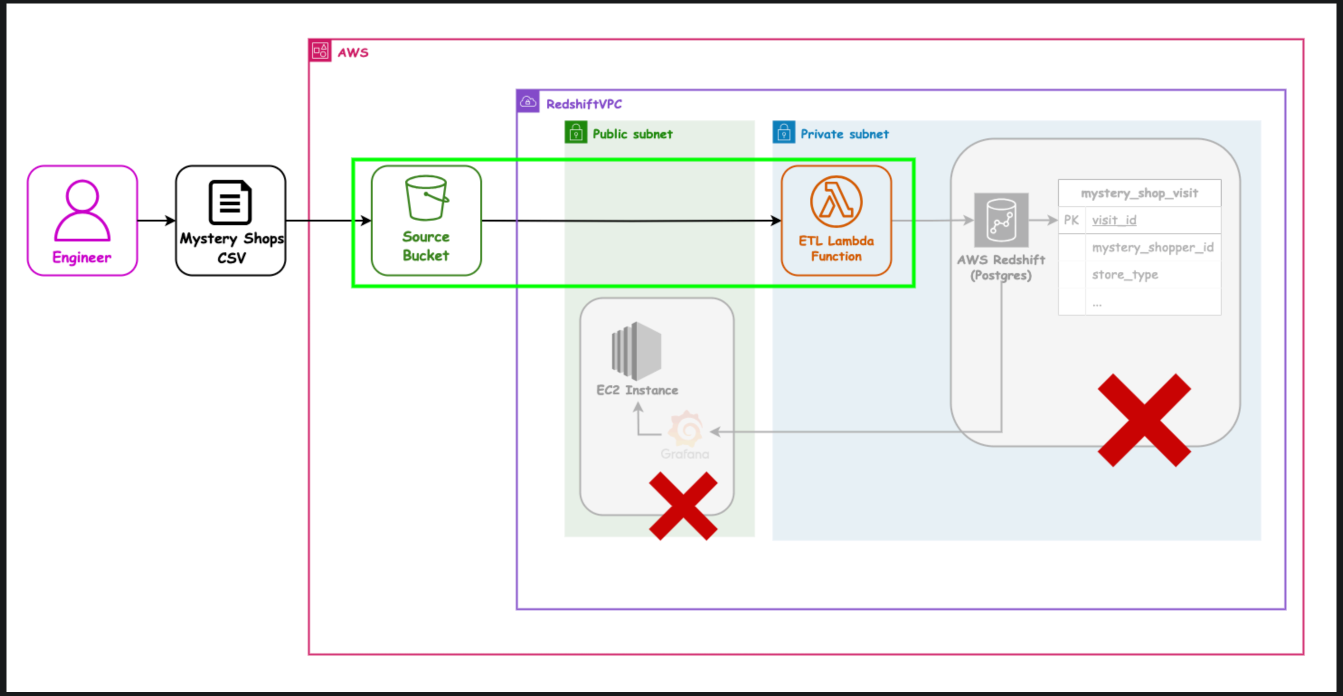
`So that` the data can be analysed

`And` the pipeline can run daily

School of Tech
part of accenture

Our next user story - Architecture

Now that we know Lambda a bit, we will deploy a Lambda "properly" using IaC:

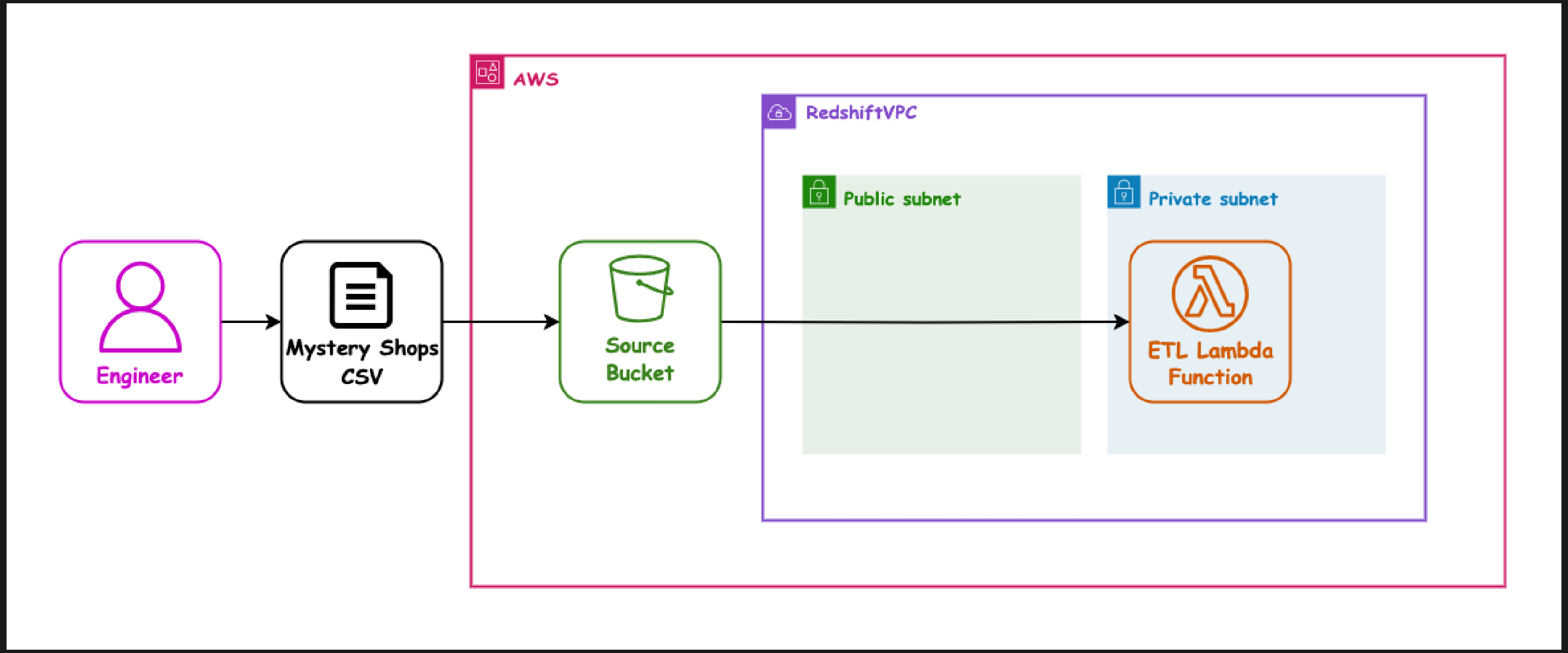


> This session - we will set it up with IaC.

School of Tech
part of accenture

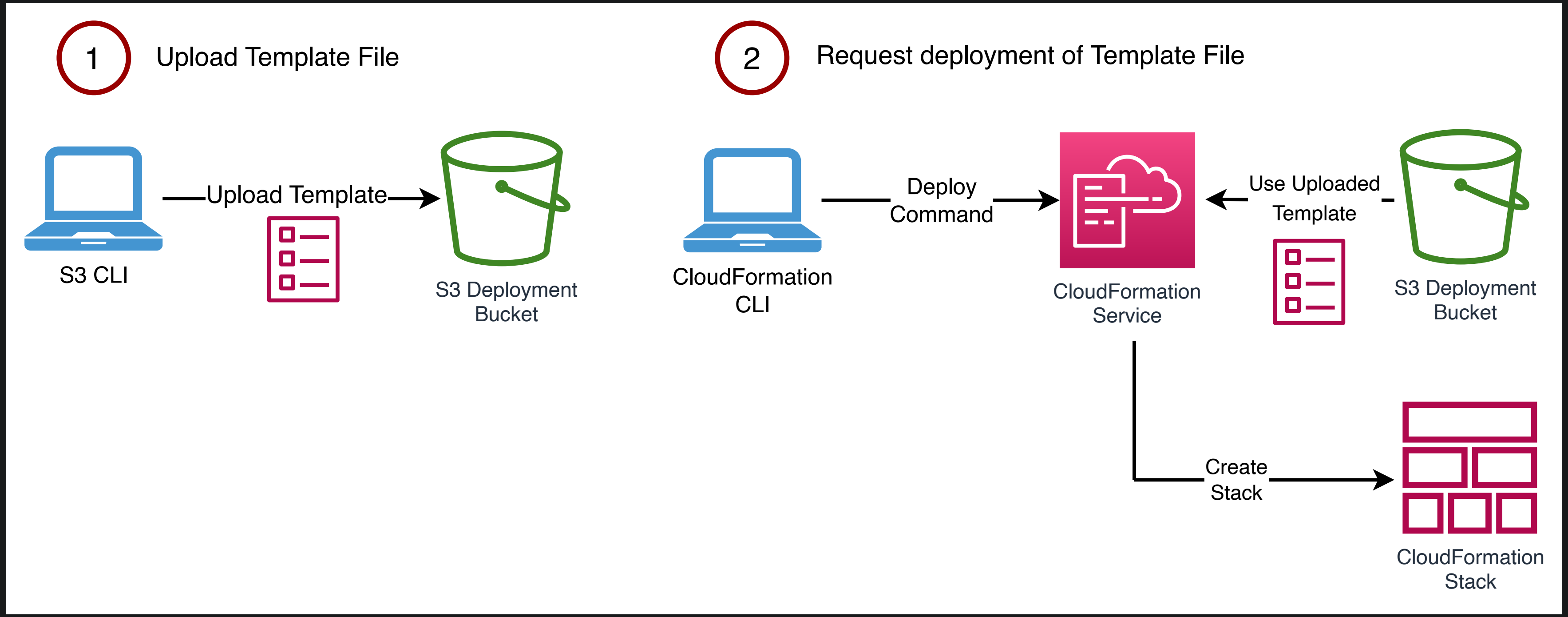
Our next user story - Architecture

These are the pieces we will need this session:



Cloudformation Deployment

This is a complex process with a few stages:



Some notes on the next slides.

Deployment Bucket

Our lambda code can get very big, especially with added dependencies.

It is common practice to do the following when deploying lambdas from IaC:

- Install any dependencies locally, into the same folder as our python code
- Zip up the Lambda code folder, including the above dependencies
- Upload the Zip to a "Deployment Bucket"
- The Lambda is then deployed from the Zip in the Deployment bucket into the Lambda service

School of Tech
part of accenture

Packaging vs Deployment

Packaging is the act of getting CloudFormation to:

- Bundle our Lambda code into Zip files,
- Upload the zip files somewhere ready to use later (S3)
- Update our template YAML files to point to the Zip, so that CF knows what to do in AWS

> We can see this in the ./handouts/deploy.sh file.

Our Data Bucket

In our previous session(s) we set up a *data* bucket to put our CSV files in.

This is separate to the *Deployment* Bucket we need to put our zips of lambda code into.

Demo - Starting point

> We need to start from the partially complete file ../handouts/etl-stack.yml.
>
> This continues from our previous sessions.

Demo - The provided Lambda

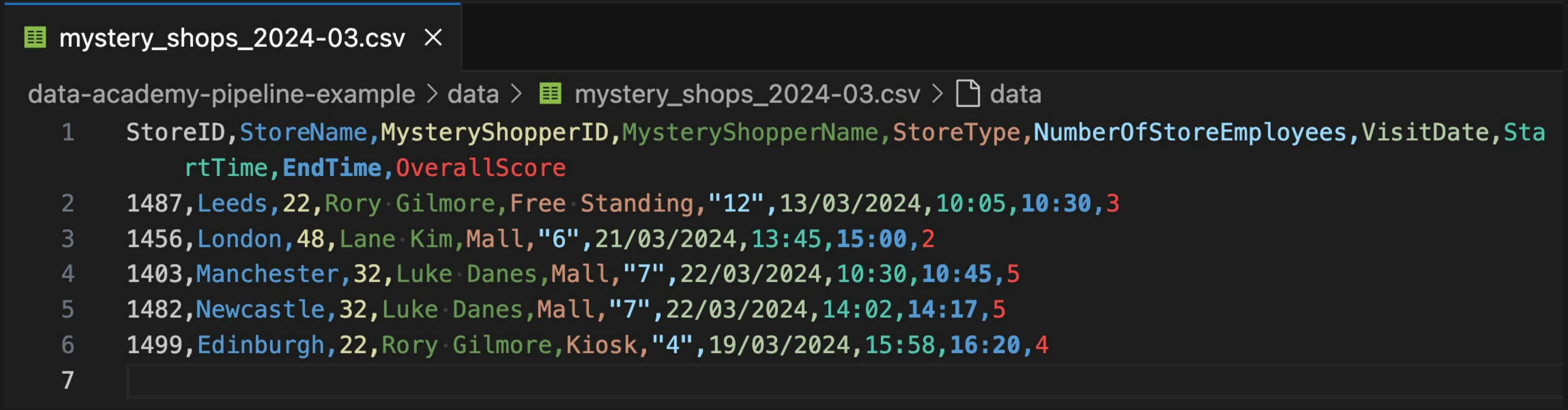
> The Lambda code to process the CSV has been written for us by our team mates - we can see this in file ./handouts/src/mystery_shop_etl_lambda.py.

This session is about using CloudFormation for Lambda, as distinct from writing the workings of a python lambda ourselves.

School of Tech
part of accenture

Demo - The provided data

> A sample mystery_shops_2024-03.csv file is also provided.



mystery_shops_2024-03.csv

data-academy-pipeline-example > data > mystery_shops_2024-03.csv > data

```
StoreID,StoreName,MysteryShopperID,MysteryShopperName,StoreType,NumberOfStoreEmployees,VisitDate,StartTime,EndTime,OverallScore
1487,Leeds,22,Rory Gilmore,Free Standing,"12",13/03/2024,10:05,10:30,3
1456,London,48,Lane Kim,Mall,"6",21/03/2024,13:45,15:00,2
1403,Manchester,32,Luke Danes,Mall,"7",22/03/2024,10:30,10:45,5
1482,Newcastle,32,Luke Danes,Mall,"7",22/03/2024,14:02,14:17,5
1499,Edinburgh,22,Rory Gilmore,Kiosk,"4",19/03/2024,15:58,16:20,4

```

Code along - Parameter

> Add a parameter for Network Stack Name, so we know where to put the lambda (so that in a later session it can talk to RedShift).

- In the `Parameters` section
- With logical name `NetworkStackName`
  - With a `Type` of `String`
  - A `Default` value of `project-networking`
  - And a helpful `Description`

Code along - lambda

> Add a Lambda with a dynamic name (from `YourName`), so all our lambdas are unique.

- In the `Resources` section
- With a logical name like `EtlLambdaFunction`
- And a specific `Type` of `AWS::Lambda::Function`
- And many `Properties`...

See next slide for more.

School of Tech
part of accenture

AWS Lambda properties

There are many that we can set, we need at least the following:

- A unique and dynamic `FunctionName`, using `YourName`
- An up to date `Runtime`, `python3.12`
- A `Handler` to specify the file name and function name to run
- The `Code` setting, to specify which folder our source code is in e.g. `./src`
- A `Role`, to assume for security so we are allowed to talk to RedShift and the S3 bucket
- A `Timeout` value in seconds e.g. `30`, high enough for our E-T-L to run but not time out
- A `VpcConfig`, to put our lambda in the same networking as RedShift so it can see the DB
- A `Tag` with value `Name` to further identify our lambda

Code along - Wake the Lambda

> Add a Notification Configuration to the CSV data bucket, so that files arriving there wake up the lambda.

We need to extend the `ShopperRawDataBucket` configuration `Properties`, like so:

- Add a new `NotificationConfiguration` property
- With a child property of `LambdaConfigurations`
    - This has a child list of Event & Function tuples
    - Add an `Event` of type `s3:ObjectCreated:*`
    - With a `Function` (lambda) reference to `!GetAtt EtlLambdaFunction.Arn`

S3 as a source

As mentioned in the previous session aws-03-console-s3, S3 can be a source of our data:

- We've set up S3 to send a notification or "event" to our Lambda to wake it
- This event will tell us the *bucket* and *file name* (but not the content / payload)
- If those systems fail to respond, some of them will receive a retry - for example, if Lambdas are *throttled*, S3 will retry the event for up to 6 hours

School of Tech
part of accenture

Code along - Dependencies

> We will tell CloudFormation that the Bucket depends on the permissions and the lambda.

```yaml
ShopperRawDataBucket:
  ...
  DependsOn:
    - ShopperRawDataBucketPermission
    - EtlLambdaFunction
```

Most of the time, CloudFormation will work these out for it's self. However we have found in this stack, the build order is more reliable with this hint added.

Code along - Because Security

> Add a Source Bucket Permission, so the Lambda is allowed to read from in the bucket when it is invoked.

- We need a new `Resource` called `ShopperRawDataBucketPermission`
- With a `Type` of `AWS::Lambda::Permission`
- The `Properties` of it are
  - An `Action`, which is `lambda:InvokeFunction`, for when the lambda is activated
  - The `FunctionName`, by reference to our lambda, e.g. `!Ref EtlLambdaFunction`
  - For the specific `Principal` that is `s3.amazonaws.com`
  - Allowing the `SourceArn` by name so `!Sub 'arn:aws:s3:::${YourName}-shopper-raw-data'`

Code along - Log into AWS

> Make sure you are logged into AWS in your terminal
>
> `aws-azure-login --profile sot-academy`

- Windows users may need to use Powershell

Demo - the Deploy script - 5 mins

> The deploy script ./handouts/deploy.sh is done for you, so that it will reliably work.
>
> Instructor to show the file.

It does the following:

- Collect your `aws-profile` and `your-name` from the command line
- Deploy a stack called `your-name-shopper-deployment-bucket`
- Install the Lambda's dependencies in the `src` folder
- Package the `your-name-shopper-etl-pipeline` stack with Lambda Zip in S3
- Deploy a stack called `your-name-shopper-etl-pipeline`

School of Tech
part of accenture

Code along - Deployment

> Let's all deploy our stacks. This may take some time!

- Windows users may need to do this in GitBash
- `YourName` should be entered `lower-case-with-dashes`, as it will be used in the S3 Bucket names

Run the ./handouts/deploy.sh script like this:

```
cd handouts
./deploy.sh <aws-profile> <your-name>
# e.g.
./deploy.sh sot-academy rory-gilmore
```

Code along - Trigger the lambda

> Upload the sample CSV file mystery_shops_2024-03.csv into your data bucket.
>
> This should trigger your lambda.

School of Tech
part of accenture

Code along - Check the logs

> Find your Log Group in CloudWatch and check the latest Log Stream.
>
> ...Do you see some nice useful logs?

The results

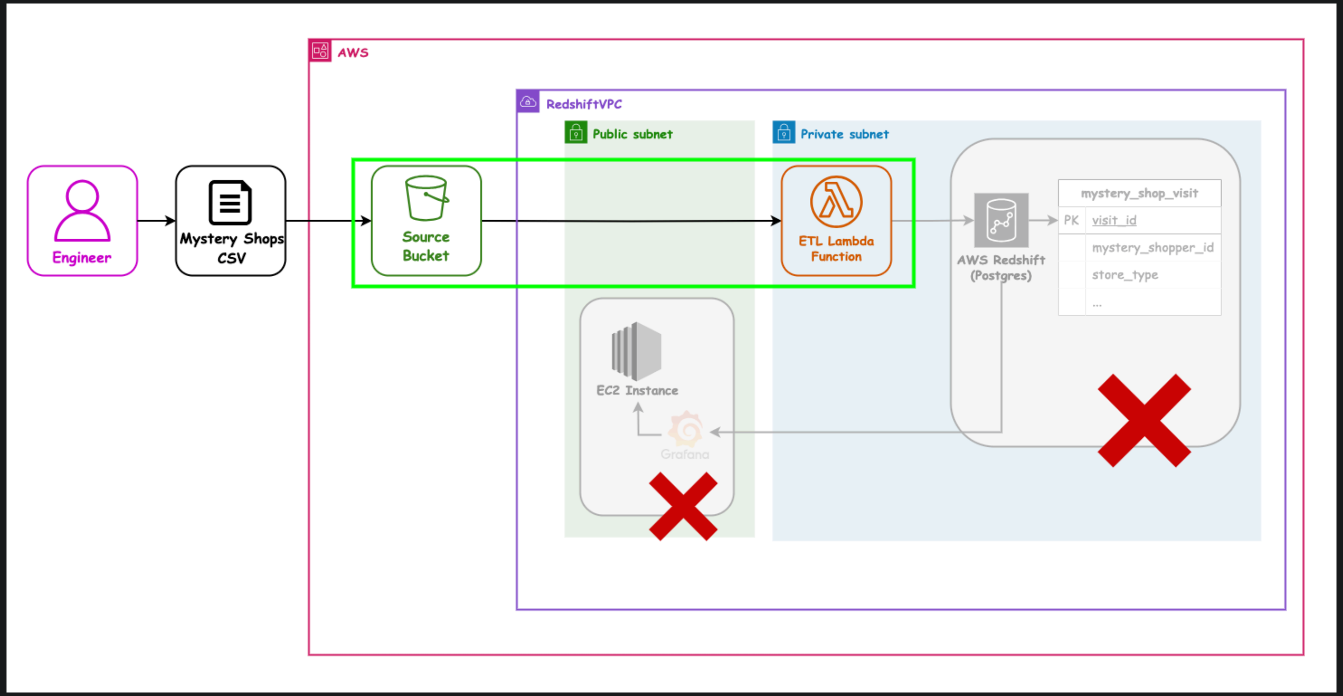
> In the `./solutions` folder there is a completed `etl-stack.yml` with extra comments, as a refresher of what we have assembled.

This is provided so that after the session you can cross-reference what we put together with the slides.

School of Tech
part of accenture

Our next user story - Architecture

This is what we just did with IaC:

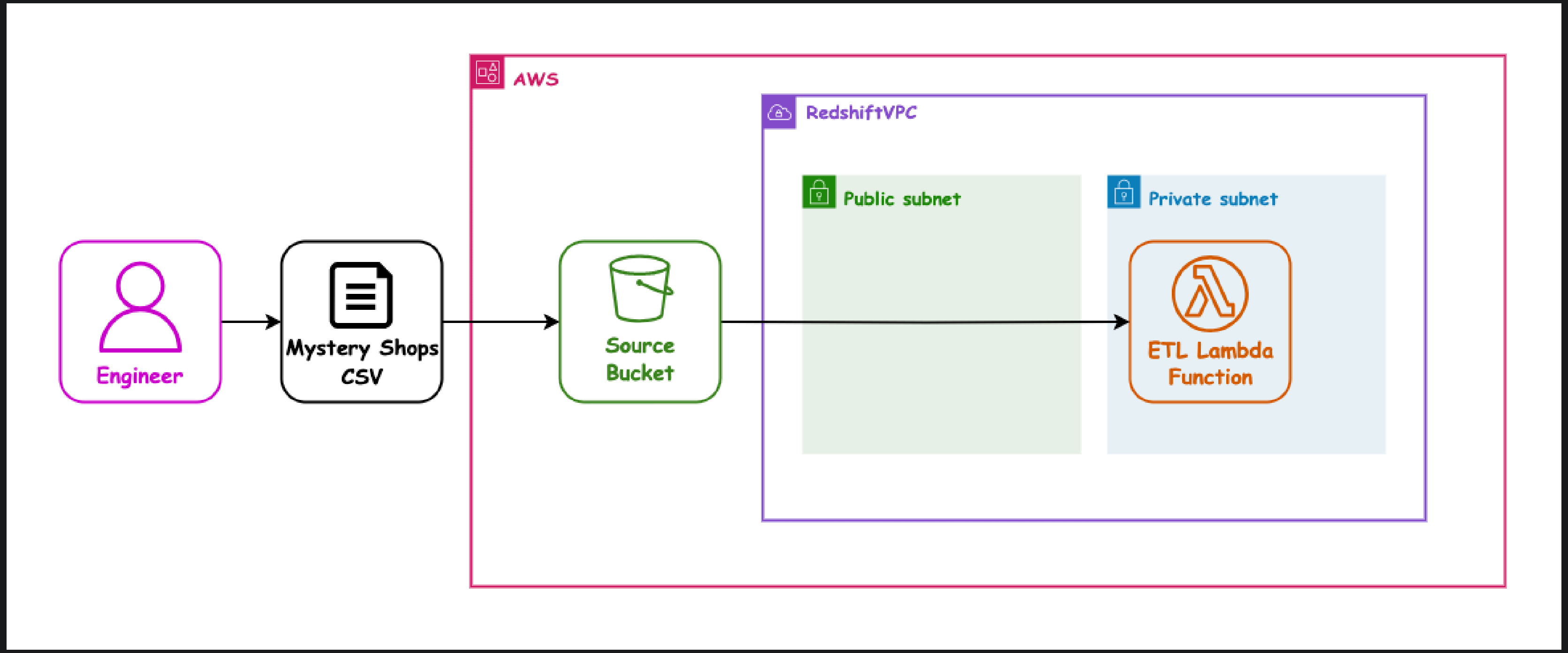


> This session - we set it up with IaC instead of manually.

School of Tech
part of accenture

Our next user story - Architecture

These are the bits we used:



School of Tech
part of accenture

Terms and Definitions - recap

- Lambda
- Event Driven
- Event Trigger
- Handler function
- S3
- Package
- Deployment

Overview - recap

- Lambda as compute in AWS
- Packaging with CloudFormation
- Deployment with CloudFormation
- Event Triggers for Lambdas (with S3)

School of Tech
part of accenture

Learning Objectives - recap

- How to add a lambda function using CloudFormation
- How to add a trigger between the S3 bucket and the Lambda
- How to process the incoming event in your lambda function

School of Tech
part of accenture

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