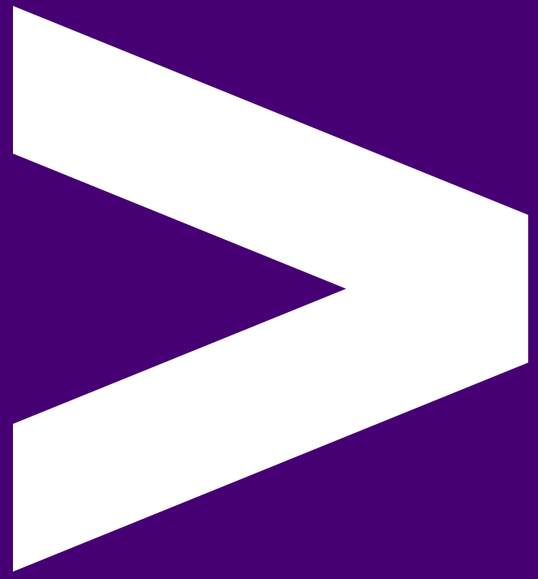
School of Tech
part of accenture

AWS 04 - Intro to DevOps with CloudFormation



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*

Overview

- DevOps - *what* it is and *why* it is
- Infrastructure as Code (IaC)
- CloudFormation concepts
- CloudFormation syntax and features
- S3 bucket setup with Cloudformation

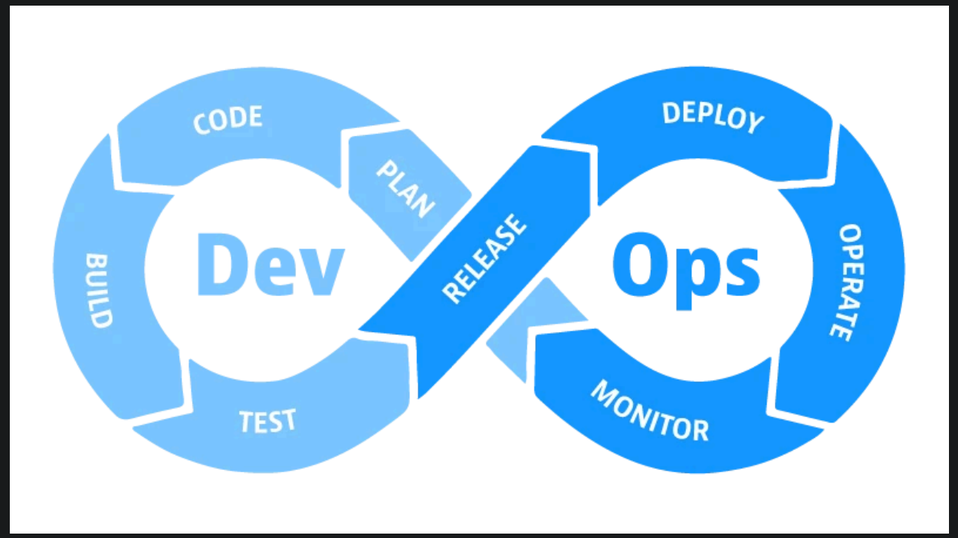
School of Tech
part of accenture

Learning Objectives

- Explain the role DevOps plays in modern software
- Explain why Infrastructure as Code is important
- Identify the parts of a CloudFormation Template
- Be able to write and deploy CloudFormation Templates that create S3 buckets in AWS

DevOps

Development and Operations



DevOps - A History

Developers and IT Ops professionals had separate (and often competing) objectives, department leadership, key performance indicators, and often worked on separate floors or even in separate buildings.

Developers looked after building the system, IT Ops looked after supporting and monitoring the system.

This resulted in siloed teams concerned only with their processes and releases.

These silos often meant miscommunication, delayed deliveries and strains on morale and relationships.

What is DevOps?

A combination of **culture, practises** and **tools** that increases an organisation's ability to deliver services at high velocity.

More succinctly, it is a set of practices that combines software development (Dev) and IT operations (Ops).

Under a DevOps model, "dev" and "ops" are no longer siloed.

Both are merged into a single team where engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.

School of Tech
part of accenture

DevOps Culture

- **Increased collaboration** as a team as opposed to silos
- **Share responsibility** across the whole team so no one process is looked after by specific people, thus spreading the knowledge and pain
- **No silos** between development and operations
- Give power to **autonomous teams** to enable them to make their own decisions in order to collaborate effectively, and remove convoluted decision making processes
- **Build quality into the development process**
- **Value feedback** to continuously improve ways of working
- **Automate** as much as you can

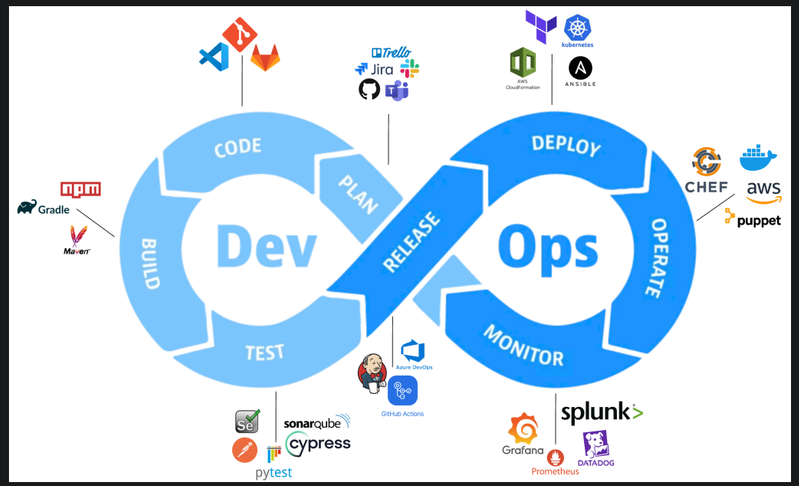
DevOps Practices

- Infrastructure as Code (IaC)
- Continuous Integration (CI)
- Continuous Delivery (CD)
- Smaller Apps and Web Servers (services) that are decoupled
- Monitoring and Logging
- Communication and Collaboration

School of Tech
part of accenture

DevOps Tools

- Source Control (Git etc)
- Collaboration / communication tools (Slack, Teams)
- Issue tracking (Jira, Trello, ZenDesk, MS Planner, GitHub Projects)
- Configuration tools (Ansible, Puppet, Chef)
- CI/CD tools (GitHub Actions, AWS Code Deploy)
- And plenty more!



Benefits of DevOps

Speed: Smaller apps and and continuous delivery lets teams take ownership of services and then release updates to them quicker.

Rapid delivery: Increase the frequency and pace of releases so you can innovate and improve your product faster.

The quicker you can release new features and fix bugs, the faster you can respond to your customers' needs and build competitive advantage.

Reliability: Quality of application updates and infrastructure changes so you can reliably deliver at a faster pace while maintaining a positive experience for end users.

Better internal culture: DevOps practises lead to better communication, increased productivity and agility.

Why does software need to change?

It changes all the time for many reasons:

- Features
- Bug fixes
- Security patches
- Contract changes
- Performance optimisations

What does software need to be able to operate?

Many parts might be required;

- Server, or, somewhere to run code
- Database
- Cache
- Storage
- APIs
- Networking

What software do we need?

To "do DevOps" we need tools that can setup our servers, deploy our software, create databases, and so-on.

The tool we will use today, and for the rest of our AWS sessions, is AWS CloudFormation.

CloudFormation

The Mystery Shopper ETL - revisited

Before migrating our final project pipelines to AWS, our client SuperCafe would like us to build a similar data pipeline focusing on mystery shop data from their branches.

> Mystery shopping is a method used by retail companies to help measure job performance and the quality of service being delivered to customers. Typically, a mystery shopper will pay a visit to a branch or store, mirroring the behaviours of a real customer and then submitting scores or feedback about their experience.

SuperCafe have identified some newly opened branches where quality of service could increase, and have introduced mystery shopping in order monitor and help improve customer experience.

School of Tech
part of accenture

The Mystery Shopper ETL - revisited

Similarly to the sales pipeline we are building, SuperCafe are recording the outcomes of mystery shop visits in CSV files:

- The CSV is uploaded at the end of each month, and represents all visits for the month across all branches in the initiative
- They have provided us a sample CSV containing 5 rows

mystery_shops_2024-03.csv

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

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

The Problem - revisited

- It is time consuming to collate data manually on all branches into one CSV
- Gathering meaningful data for the company on the whole is difficult, due to the limitations of the current solution
- Visualising trends is being done manually and is prone to human error
- Integers are represented inconsistently in the data
- `VisitDate` values have to be manually re-formatted to integrate with external spreadsheet software

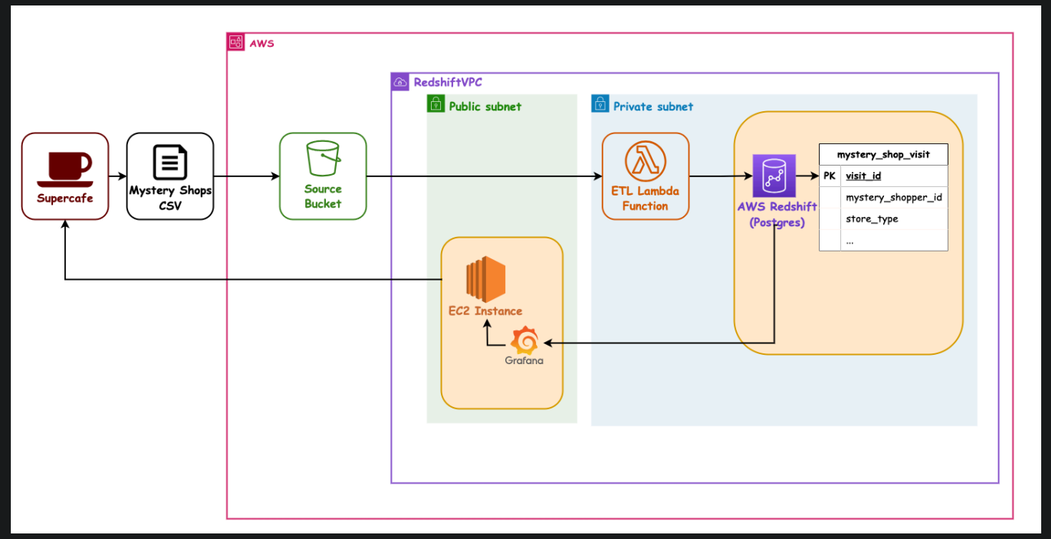
The Solution - revisited

After an initial discovery phase, we have agreed a plan to build a small cloud-based ETL pipeline to help SuperCafe solve some of these issues.

This has been identified as a great opportunity for us to learn and grow as an engineering team before moving onto the sales pipeline for our final project!

School of Tech
part of accenture

Proposed Pipeline Architecture - revisited



School of Tech
part of accenture

Our first user story - revisited

`As a` SuperCafe senior manager

`I want` a durable and available location to store monthly mystery shop data

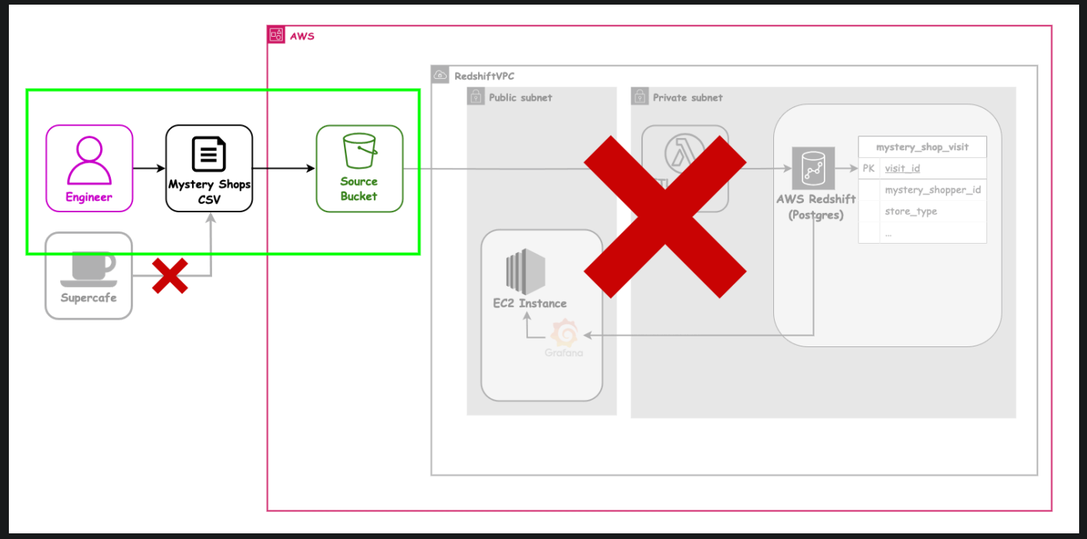
`So that` access to the data is securely configured

`And` the data can be automatically integrated with a downstream ETL pipeline

School of Tech
part of accenture

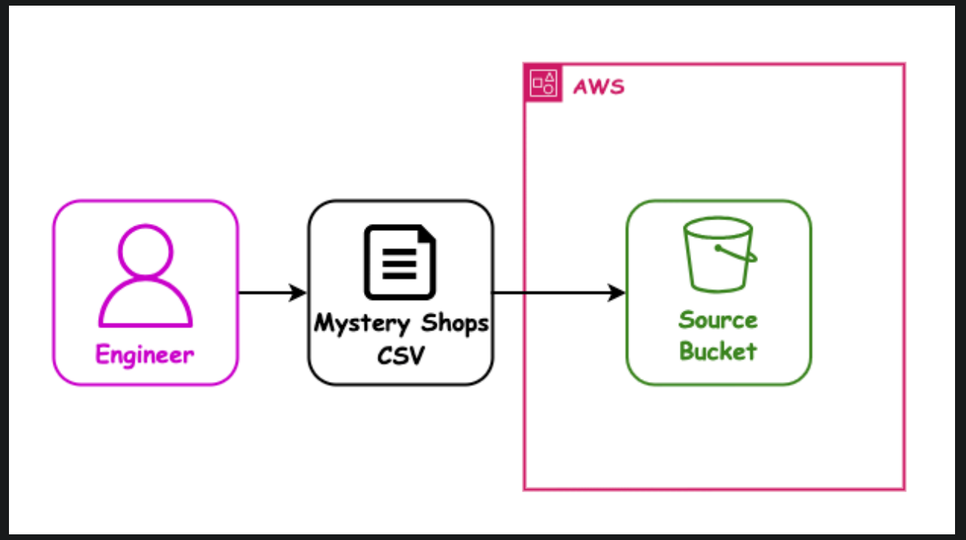
Our first user story - Architecture

In this session we can redo our manual work with IaC:



Our first user story - Architecture

So today we only need to do this bit (but properly, this time with IaC):



Infrastructure as Code (IaC)

> Question: What is Infrastructure as Code?

What is Infrastructure as Code?

- The management of *infrastructure* through version-controlled files
- Generates the exact same environment every time through a code file
- Used in conjunction with CI/CD pipeline
- Without IaC, teams must maintain the settings of all environments individually

What is "Infrastructure"?

In a traditional non-cloud context:

- Application Servers
- Virtual Machines
- Databases
- Firewalls
- etc

In an AWS cloud context, "Infrastructure" could be any component of any AWS service:

- S3 Buckets
- Lambda Functions
- EC2 compute instances
- Users
- Roles
- ... and much more

👉 Almost anything you can create through the AWS web console can be considered "infrastructure"!

Infrastructure as Code

It is an engineering principle in which we define **templates** for our **application and service infrastructure** to allow it to be created, deleted, re-created, or duplicated **consistently and predictably**

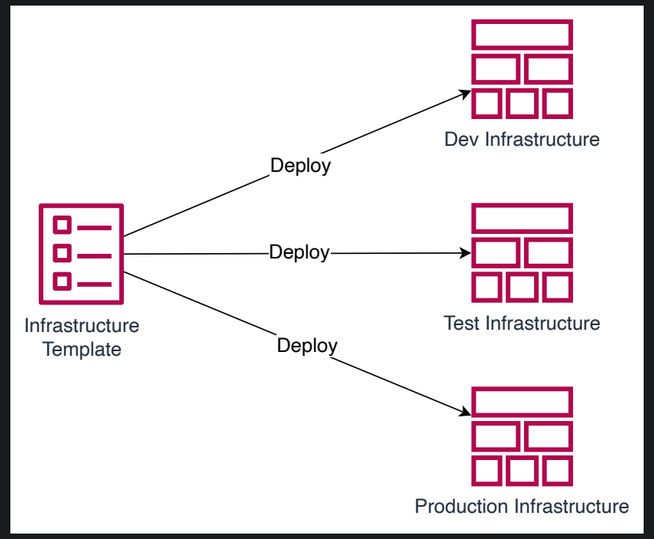
👉 There are many different IaC tools and technologies in the engineering world, but the goal and principle of IaC is always the same

Infrastructure as Code

IaC lets us take a template that contains everything we need for our application, and deploy multiple instances of that application side-by-side.

We don't build our application infrastructure directly, instead we build a template which can then create that infrastructure for us when deployed.

Infrastructure from Templates



- One template can be deployed any number of times
- Deployed instances can be updated by modifying and re-deploying the template

School of Tech
part of accenture

Without Infrastructure as Code

- Each new deployment requires lots of human work to provision 😩
- There may be mistakes that aren't noticed and cause problems 😵
- The instructions to build an application can be lost or forgotten! 😱

With Infrastructure as Code

- Any number of deployments created automatically with little work 😃
- Every deployment is identical, since it came from the same template 🤩
- The template is in source-control, so it can never be lost! 🎉

School of Tech
part of accenture

Quiz Time! 🤓

Why is Infrastructure as Code Important?

1. It reduces mistakes
2. It helps teams collaborate
3. It is automatable and repeatable
4. All of the above

Answer: 4

When is it okay to NOT use IaC?

1. When you need to get new features added quickly on a deadline
2. When you are exploring or prototyping new functionality
3. When the client doesn't mind if you use it or don't
4. Never OK

Answer: `2 (Sometimes!)`

When is it okay to NOT use IaC?

- AWS console can be used for intermittent purposes and where the created functionalities are not replicated.
- For all other scenarios IaC should be used, so that we create software that is redeployable.
- Client might not "mind", but as engineers it is OUR responsibility to recommend best-practice ways of working.
- Even for a "prototype" app it might still best to use IaC.

Infrastructure as Code Summary

- IaC makes deployments predictable and repeatable
- IaC makes collaborating with other people in your team on infrastructure easy
- You should **always** use IaC when building cloud apps and services

School of Tech
part of accenture

Emoji Check:

Do you feel you've understood the core concepts of IaC (Infrastructure as Code)?

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

AWS CloudFormation

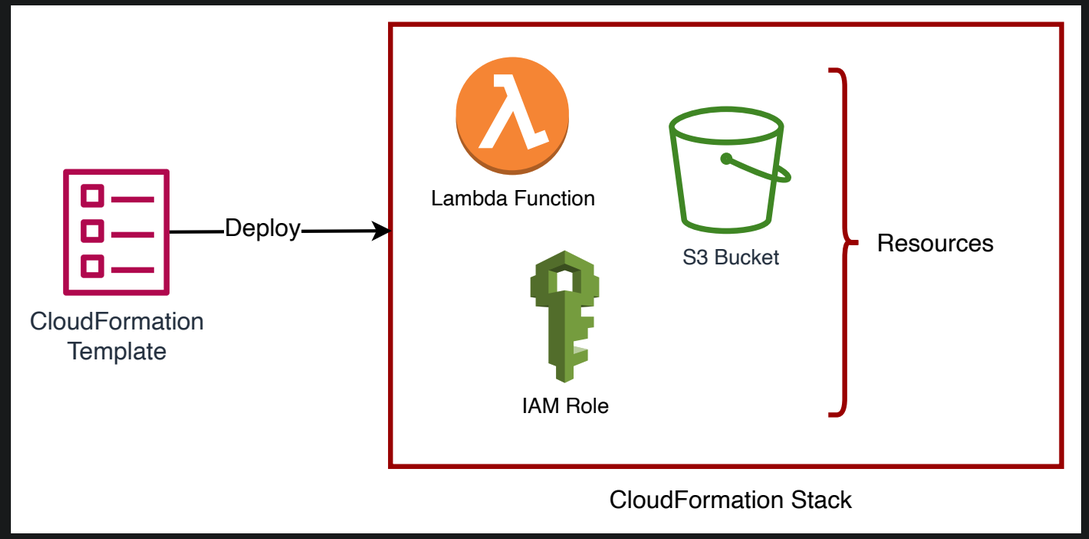
- CloudFormation is an IaC tool designed by AWS for use within AWS
- It natively understands and works with (almost!) all AWS infrastructure and services
- CloudFormation runs as a service within AWS, which can orchestrate our deployments
- It is an industry-standard and popular choice for AWS development

School of Tech
part of accenture

CloudFormation Key Concepts

- **Template** - A 'blueprint' for our infrastructure
- **Stack** - A deployed instance of a template
- **Resources** - Infrastructure components created inside a stack

CloudFormation Key Concepts



- When a template is deployed, it creates a stack containing the resources defined in that template

A CloudFormation Template

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Description: Template for a single S3 bucket

Resources:
  MyS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: academy-de-example-bucket
```

The above template creates a S3 Bucket `resource` called 'academy-de-example-bucket'

CloudFormation Anatomy (1)

```yaml
AWSTemplateFormatVersion: "2010-09-09" # <-- Version
Description: My S3 bucket # <-- Description
Resources:
  MyS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: academy-de-example-bucket
```

- *Version* (Optional)
  - Always "2010-09-09" (only one version exists 🤷)
- *Description* (Optional)
  - Human-readable description of what this template is for

CloudFormation Anatomy (2)

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Description: Template for a single S3 bucket

Resources: # <-- Resources block
  MyS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: academy-de-example-bucket
```

- *Resources* (Required)
  - The set of infrastructure to be created by this template
  - Could be S3 buckets, EC2 instances, roles, lambda functions, or any other supported AWS resource

School of Tech
part of accenture

CloudFormation Anatomy (3)

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Description: Template for a single S3 bucket

Resources:
  MyS3Bucket: # <-- Resource Name (for template reference)
    Type: AWS::S3::Bucket # <-- Resource Type
    Properties:
      BucketName: academy-de-example-bucket
```

- *Resource Name*
  - A label for this resource
  - Can be anything, as long as it is unique within the template
- *Resource Type*
  - The type of infrastructure this resource represents
  - Comes from a fixed list of possible resource types

School of Tech
part of accenture

CloudFormation Anatomy (4)

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Description: Template for a single S3 bucket

Resources:
  MyS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: my-bucket-name # <-- Resource Properties
```

- *Resource/Properties*
  - Configuration for this type of resource
  - Possible properties for a resource type can be found in the documentation
  - Some properties are required, while others are optional

CloudFormation Anatomy (4)

Possible resource properties for an S3 bucket for example may include things like the bucket name plus additional properties such as:

- Whether encryption is applied on the bucket
- Public or Private access to bucket files
- Use of the bucket to host a website

School of Tech
part of accenture

CloudFormation Anatomy (5)

```
Parameters:
  YourName:
    Type: String
    Description: Enter your name to customise your resource na
    Default: Alice Bloggs
```

- *Parameters*
  - A list of settings we can change
  - *Like* a variable in programming but the value of parameters can't be changed during code execution
- *YourName* is the name for the parameter
  - *Type* Number, String, Boolean, etc
  - *Description* Some free text describing the parameter
  - *Default* The value that will be used if not overridden

Points of Note

- Templates are YAML documents
- In YAML 👉 indentation is important 👈 (Just like in Python!)
- All possible AWS resource types and their possible properties can be found in the AWS docs

The Mystery Shopper ETL

After further discussions on the solution the mystery shopper pipeline, your tech leads have stipulated that we update the original user story to include a new piece of technical acceptance criteria:

`As a` SuperCafe senior manager

`I want` a durable and available location to store monthly mystery shop data

`So that` access to the data is securely configured

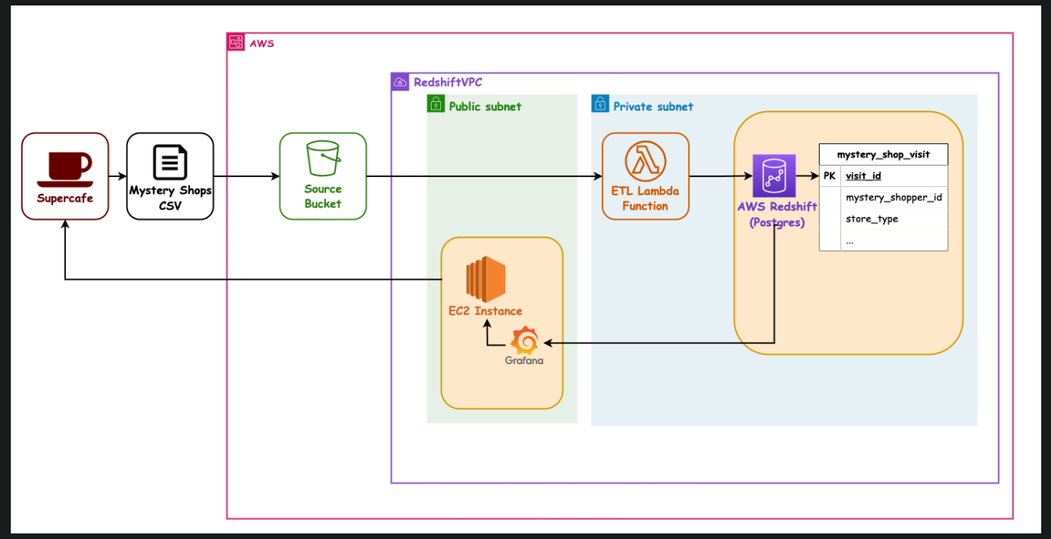
`And` the data can be automatically integrated with a downstream ETL pipeline

New acceptance criteria

The S3 bucket used to store the raw mystery shop data should be managed via Infrastructure as Code, so the team can easily manage the automation of creating it in AWS and provide flexibility later if multiple development and production versions of the pipeline are needed.

This approach also means the source code for how the bucket is defined can be committed to source control, hurrah!

Final outcome



Today's goal

Your instructor will give you a peek of our target yaml file!

> Lets look at the ./solutions/etl-stack.yml file - we will come back to this later.

Deploying a Template

We have a Template that defines our Resources.

How do we turn that template into a stack in AWS?

There are two main options:

1. Deploy via AWS Web Console
2. Deploy via the AWS Command Line Interface (CLI)
    - *the preferred way, we can automate this!*

Console Deployment

A stack can be created by uploading a template file via the Web interface, by going to:

- AWS Web Console
  - `CloudFormation` -> `Stacks` -> `Create Stack`

While a stack is deploying, the web console can be used to see what is happening.

The web console can also be used to check what resources have been created.

Deployment Errors

What if the deployment doesn't work?

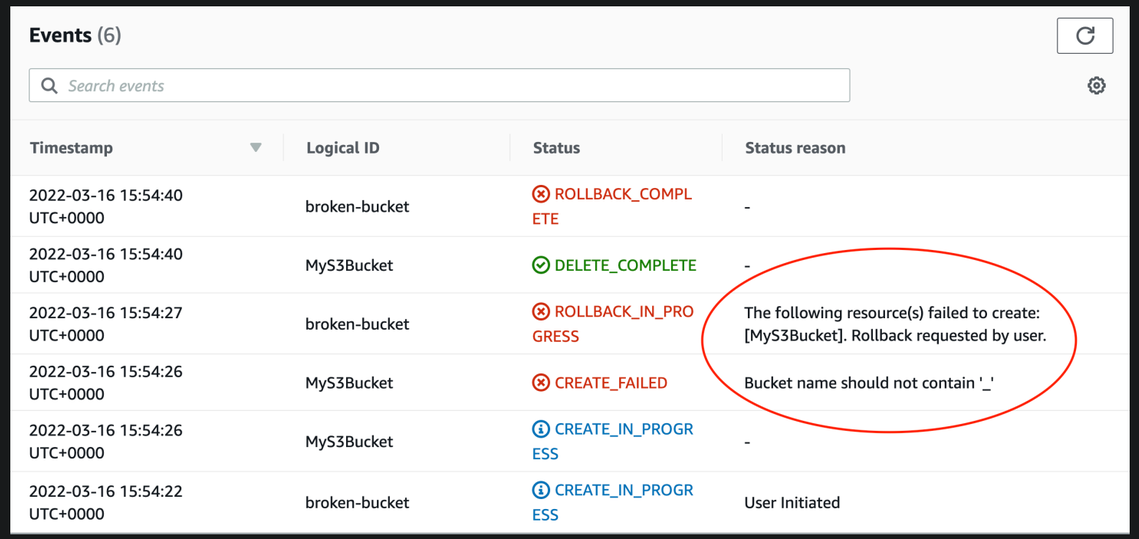
```
Resources:
  MyS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: academy_de_example_bucket
```

The above template has a problem. Errors in deployment can be viewed via the web console.

Remember: Identifying and understanding errors is a CRITICAL SKILL for an Engineer 👀

School of Tech
part of accenture

Deployment Errors



Events (6)

Search events

Timestamp	Logical ID	Status	Status reason
2022-03-16 15:54:40 UTC+0000	broken-bucket	ROLLBACK_COMPLETE	-
2022-03-16 15:54:40 UTC+0000	MyS3Bucket	DELETE_COMPLETE	-
2022-03-16 15:54:27 UTC+0000	broken-bucket	ROLLBACK_IN_PROGRESS	The following resource(s) failed to create: [MyS3Bucket]. Rollback requested by user.
2022-03-16 15:54:26 UTC+0000	MyS3Bucket	CREATE_FAILED	Bucket name should not contain '_'
2022-03-16 15:54:26 UTC+0000	MyS3Bucket	CREATE_IN_PROGRESS	-
2022-03-16 15:54:22 UTC+0000	broken-bucket	CREATE_IN_PROGRESS	User Initiated

Reasons for failure can be determined from the 'Events' pane in the console.

Any deployment errors can also be views using CLI commands in your terminal - but for once, this is easier to see in the web.

School of Tech
part of accenture

Quiz Time! 🤓

In a CloudFormation Template, what does the `Resources` section define?

1. Options that are passed into to the template
2. The set of infrastructure the template will create
3. Already-existing infrastructure that the template depends on

Answer: 2

Which of the following is FALSE?

1. A Template is a 'blueprint' for infrastructure
2. Templates can define one or more resources
3. A Stack is a deployed instance of a template
4. Stacks can be committed to source control

Answer: 4

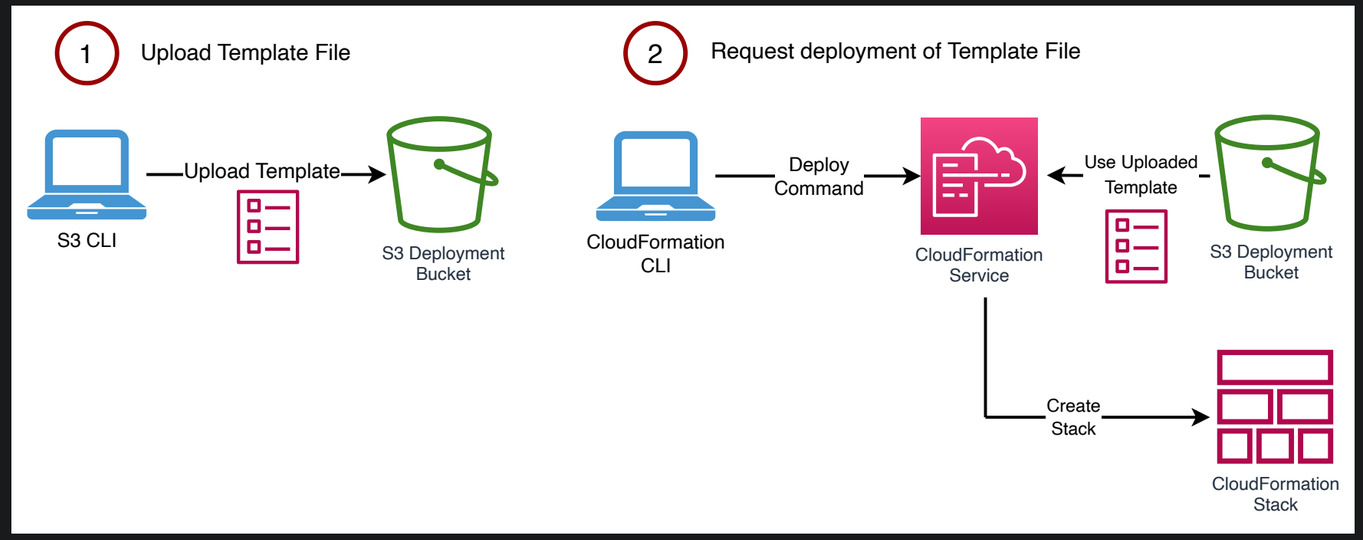
CLI Deployment

CloudFormation templates can also be deployed via the AWS Command Line Interface (CLI)

- Unlike the web interface, this can be easily invoked from a `shell script`
- This makes the CLI the preferred option to help automate deployments as part of a `CI/CD pipeline` 🤖

School of Tech
part of accenture

CLI Deployment Process

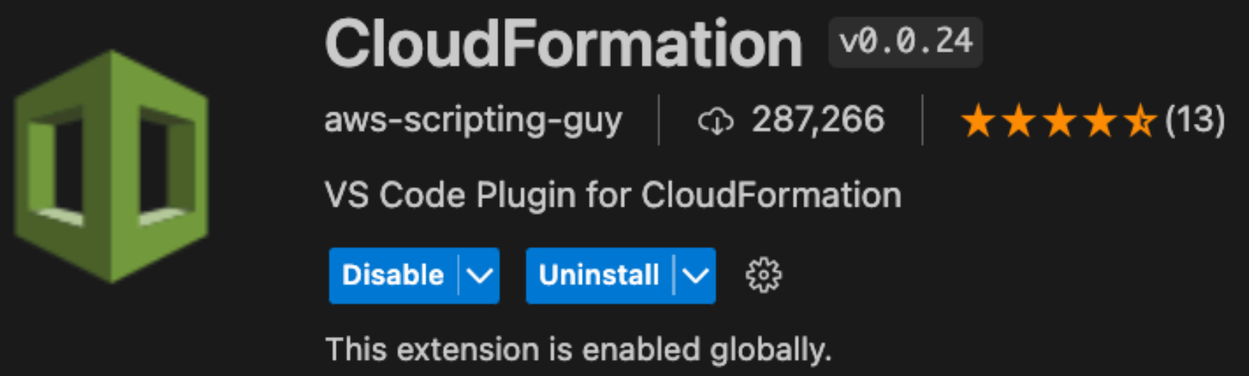


- S3 Deployment bucket stores the cloudformation template.
- Deployment bucket can be used to store any number of templates for different stacks.
- AWS CloudFormation is triggered to perform a deployment of the stored S3 template via the CLI.

School of Tech
part of accenture

Demo Prep

There is a handy CF plugin for VS Code that can remove some syntax errors we would otherwise get:



- Install the plugin
- Follow the steps to update your YAML editor config

Demo - the deployment script

> The instructor can now show you the ./handouts/deploy.sh.

This file has been written for us to run the deployment.

Demo - the blank stack file

> The instructor can now show you the starting ./handouts/etl-stack.yml.

This file needs filling in!

Exercise prep

> Have a read through the ./exercises/aws-04-cfn-intro-exercise.md file.
>
> This file contains the steps we need to do. Each step is on the following slides.

Code along - Check the parameter

> The parameter for `YourName` is already done, so no-one forgets it :-)
>
> This is passed in from the deployment script.

- `Parameters` section
  - Logical name
  - Then data `Type`, helpful `Description` and `Default` value

Code along - Add bucket

> Add a bucket with a dynamic name.

- Needs a `Resources` section
- Then a logical name e.g. `ShopperRawDataBucket`, to use within the template
- Then an AWS `Type`, e.g. `AWS::S3::Bucket`, which must be a valid value from the AWS docs
- Then some `Properties`, which is a key to hold more values

See next slide for more info.

AWS Bucket properties

These match what we saw in the AWS console in the last session; They all sit under the `Properties` key:

- A globally-unique `BucketName`, so we know which one is ours
- A set of `PublicAccessBlockConfiguration`(s), for security
  - By default, deny all public access!
- A `Name` Tag with a `Key` and `Value`, so it is labelled correctly as ours

School of Tech
part of accenture

Code along - Add policy

> Add a bucket policy with dynamic references.

- A logical name e.g. `ShopperRawDataBucketPolicy`, within the `Resources` section
- A specific `Type`, e.g. `AWS::S3::BucketPolicy`
- A set of `Properties`
  - With a dynamic reference back to our Bucket, e.g. `Bucket: !Ref ShopperRawDataBucket`
  - A `PolicyDocument` detailing our security rules (in this case - to make sure only secure traffic on HTTPS can be used, not HTTP)

See next slide for more info.

Policy Document Statements

These define the security rules for the Policy:

- A `Statement` block
- With an identifier `Sid`, which is like a unique human-readable name for the Statement
- Then an `Action`, to define what the rules apply to, like `s3:PutObject` or `s3:DeleteObject`, or `s3:*` for "all"
- A `Principal`, which is *who* the rule applies to or "*" for everyone
- The `Effect`, which is `Allow` or `deny`
- A list of AWS `Resource`(s), that the rule is guarding / protecting
- Any `Condition`(s), that turn the rule on or off

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
- Use these to deploy a stack called `your-name-shopper-etl-pipeline`

Code along - Deploy

> 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:

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

The results

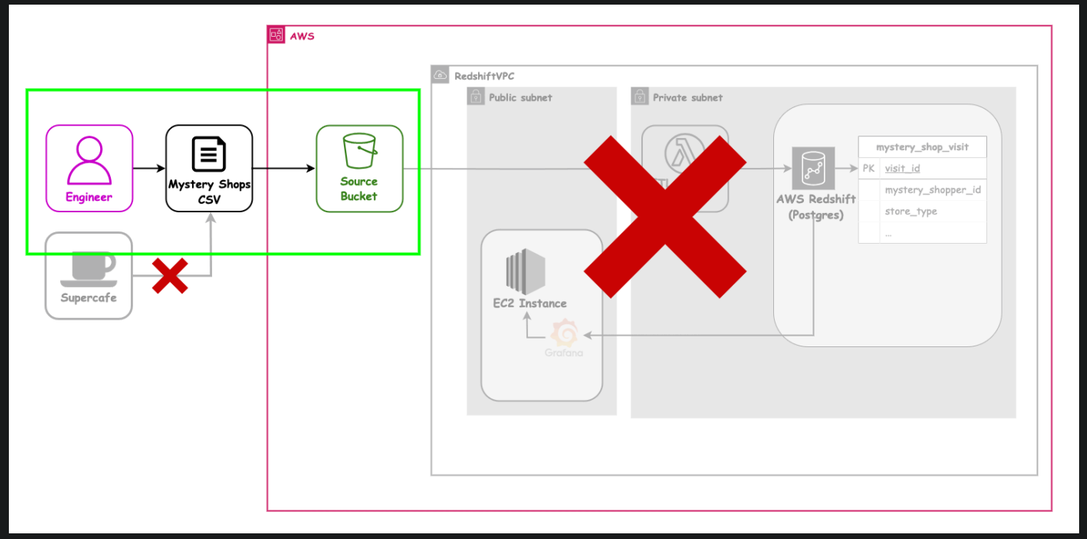
> In the `./solutions` folder there is a completed `etl-stack.yml` with extra comments, as a refresher of what we have assembled.

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

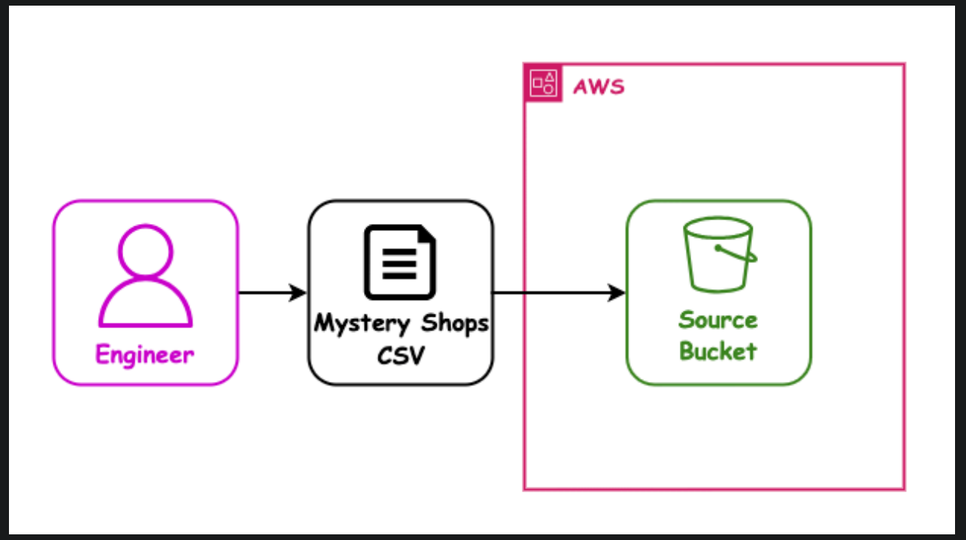
Our first user story - Architecture recap

What we did in this session was this, but properly:



Our first user story - Architecture recap

The bits we needed in this session were:



School of Tech
part of accenture

Overview - recap

- Infrastructure as Code
- CloudFormation concepts
- CloudFormation syntax and features
- S3 bucket setup with Cloudformation

School of Tech
part of accenture

Learning Objectives - recap

- Explain why Infrastructure as Code is important
- Identify the parts of a CloudFormation Template
- Be able to write and deploy CloudFormation Templates that create S3 buckets in AWS

Further Reading

- Cloudformation Docs
- CloudFormation Resource Types
- CloudFormation Functions
- CloudFormation Deploy

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