Devops Portfolio

# MICHAEL PLASOM-SCOTT

For DWP Digital and Makers Academy

## Contents

##### About

[Intro To Me](#_Intro_To_Me)

[The Team](#_The_Team)

[My Role](#_My_Role)

[Tech Stack](#_Tech_Stack)

##### Portfolio

[Ticket Name](#_Ticket_Name)

[Ticket Description](#_Ticket_Description)

[Ticket Process](#_Ticket_Details)

[Knowledge](#_Knowledge)

[Skill](#_Skill)

[Behaviour](#_Behaviour)

##### DUTIES

[Knowledge](#_Knowledge)

[Skill](#_Skill)

[Behaviour](#_Behaviour)

## About

##### Intro To Me

I'm Michael Plasom-Scott. I studied as a Graphic Designer, worked as a teaching assistant and technician in a school, and I'm now re-training as a DevOps Engineer. I come from a creative and collaborative background, that although does not scream DevOps has been vital to my development in the role.

This course is a brilliant opportunity to have a glimpse into a fascinating industry, completely unlike anything I've tackled before. Every day is a new challenge, and I haven't stopped learning since day one.

##### The Team

My team is an agile team led by Tom Farrow. Along with my Line Manager, Shazad Azam, they have been extremely helpful in helping me get to grips with the day-to-day life and work ethic of a DevOps engineer. The team has daily morning standups in which we talk about what was achieved yesterday, what we plan to achieve in the coming day, and any blockers that might stand in our way. It’s also an opportunity to ask any questions or for advice. Tickets are assigned at the start of a sprint, and completed as quickly as possible, with as much communication as possible. I spent a lot of my time at the start of my placement shadowing Luis Castromil, another DevOps engineer, who moved on to giving me tickets to try myself.

##### My Role

My role is that of Apprentice DevOps Engineer in the Digital Shared Channels Experience (DSCE) at DWP. I’m in the Notifications team within this, however some projects I do affect multiple teams within the DSCE Directorate.

##### Tech Stack

AWS

S3

Lambda

Cloudwatch

DynamoDB

Terraform

Ansible

Prometheus

Grafana

Gitlab

## Portfolio

Project #1

##### Ticket Name

NSE-2264

##### Ticket Description

Create an S3 Bucket with the local name Data Migration Changes. S3 access then provided to Data Specialist who will pull the files Users.CSV and Org\_Node.CSV that will be uploaded to SharePoint. Users.CSV needs to be encrypted. Files to be exported to S3 location Wed/Fri by 8am.

This ticket was assigned to me as a way of getting me used to an agile workflow and working alongside developers to roll out some changes.

##### Ticket process

Meeting with Luis  
The process started with a call with Luis where he outlined what would need to be done, and then updated the ticket with a bullet pointed list of everything. He also said one of the first things to tackle would be the creation of a GitLab CI/CD pipeline. Tyler and I said we’d split up the basic work, he’d focus on writing the terraform of the Lambda and I’d focus on the pipeline.   
[S8](#_Skill)

Creating the pipeline  
I spent a while on the initial stage of the pipeline. It was my first time creating a GitLab CI/CD pipeline at DWP, so I was unaware of the protocol used. I looked at other examples of pipelines from different repositories that the team had worked on and spent a while trying to decipher how they worked. Working on my own branch in the repository, I pushed up a few very simple versions of what I understood, to try and see if I could get something minimal off the ground and build it up. I couldn’t understand why these were failing so I set up a call with Luis and we workshopped around the setup of a pipeline. He explained about the use of fragments, and the ‘extends’ function in a Gitlab CI file. We did some productive pair programming, building up a pipeline that would pass correctly. After that it became a lot clearer and I eventually pushed up a merge request with a pipeline that worked as it should, with the correct steps in it.   
[S13](#_Skill.)

Code review by Luis  
Once the pipeline was working as expected, Luis took the time to review the code that Tyler and I had in the repo. This was an incredibly helpful step, and one of the key parts of a smooth and agile DevOps environment. Often it was small things like not following the correct naming conventions (fig. 1), however Luis would often take the time to highlight important DevOps principles, as a learning opportunity for me (fig. 2).   
[B1](#_Behaviour)Graphical user interface, text, application, email

Description automatically generated

Figure 1 - Code Review comment from Luis

Graphical user interface, text, application

Description automatically generated

Figure 2 - Code Review comment highlighting DevOps Principle

Fixing Changes  
Following the code review I worked by myself to make sure the suitable changes were applied. This involved a lot of simple changes to variables within the file, which were simple. However, there were some elements that I was unfamiliar with, particularly the scheduling of Lambda functions, as was required by the ticket - Files to be exported to S3 location Wed/Fri by 8am - so I researched around ways to do this and looked in some other repos from the team to find examples of cron triggers. I settled on setting up the cron expression (fig. 3), so the file was exported on Wednesday and Friday at 7:30 and ran this by one of the [Business Analysts/Architects] to make sure that was a suitable time.

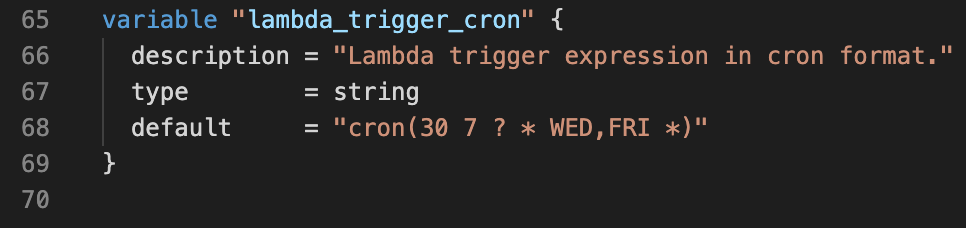
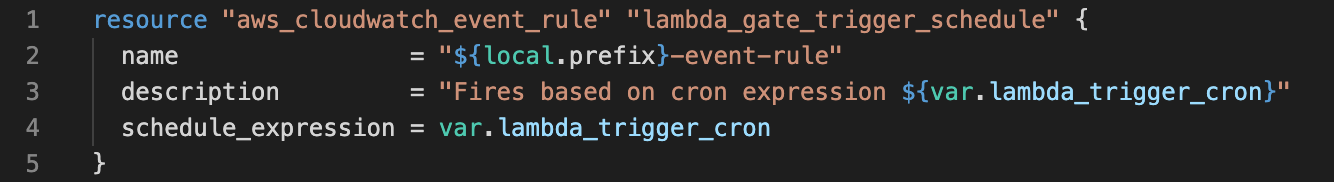


Figure 3 - Cron Trigger

Debugging with Developers  
Once the setup was all complete (in theory) from the DevOps side, a chat was created with me, Luis, Daniel, and Ritesh. Daniel and Ritesh were the developers adding some code in, using the pipeline I’d created. At this point some of the steps in the CI/CD pipeline started to fail – mainly catching errors in the code and doing what the pipeline is supposed to do in part, i.e. catch errors before going to deployment. However, one of the times it failed was due to an error in the Terraform code – the handler path in the Lambda config had been copied across from another repository and was incorrect, so I updated the code and pushed again. Another error that slipped through the net was that the Lambda wasn’t given enough time to connect to the database, having 3 seconds as a default. This gave the Devs an error message, which they fed back to me, and I fixed in the Terraform code (fig. 4).   
[K23](#_Knowledge)Graphical user interface, text

Description automatically generated

Figure 4 – Devs highlighting error

Deploying to DSE-1  
Having fixed the bugs and got the pipeline running smoothly, we took finished for the weekend. On Monday Luis started a call with me and demonstrated to me the way that we deploy code into the DSE-1 (Develop Supported) environment.

Business Impact  
The changes we made created an AWS Lambda that retrieves data from a bucket, encrypts it using a public key, and places it in another bucket. A user can then access that bucket through Sharepoint, download the data, and decrypt it using a private key. This automates a process that was previously done manually and makes it a lot more secure with the use of encryption. The pipeline created means that developers can constantly make incremental changes to this code safely.

##### Knowledge

K23 - The importance of continual improvement within a blameless culture.

##### Skill

S8 - Work in agile, multi-disciplinary delivery teams, taking a flexible, collaborative and pragmatic approach to delivering tasks.

S13 - Engage in productive pair/mob programming.

##### Behaviour

B1 - Exhibits enthusiasm, openness and an aptitude for working as part of a collaborative community; e.g. sharing best practice, pairing with team members, learning from others and engaging in peer review practices.

Project #2

##### Ticket Name

NSE-

##### Ticket Description

The project is a large job that will affect multiple teams within the SCE Directorate. Within the directorate there are multiple silos of technical stacks, for example Notifications, and ECMP, etc. with multiple teams working within them. The task would be to create an accessible single view location across the entire directorate. This location would have dashboards covering various aspects and display monitoring for all the projects, with all issues highlighted. These issues would then be triaged effectively by the respective team. Effectively this would mean that any issue can be compared across all the projects and give visibility to whether it was a lone issue or affecting multiple projects. It would be a huge update and improvement on a system already in place that does not house the monitoring in one place.

##### Ticket Process

Workshop with Tom Farrow  
Tyler and I went into the Manchester office for a workshop with the DevOps lead for our team, Tom. It being the first time I’d been in the building to work, Tom led us through the basic health and safety procedures, including fire evacuation. We then went through the requirements for the project and came up with an idea of the architecture required to get it to work. The basic structure would be a User accessing Grafana via a load balancer to query Prometheus and Alert Manager. In the workshop we worked on a diagram, and Tyler digitised the diagram to this (fig. 5) that displays our initial attempt at the architecture required to continue with the project.   
[K19](#_Knowledge.), [S21](#_Skill.)

*Diagram

Description automatically generated*

Figure 5 - Initial Diagram after Workshop

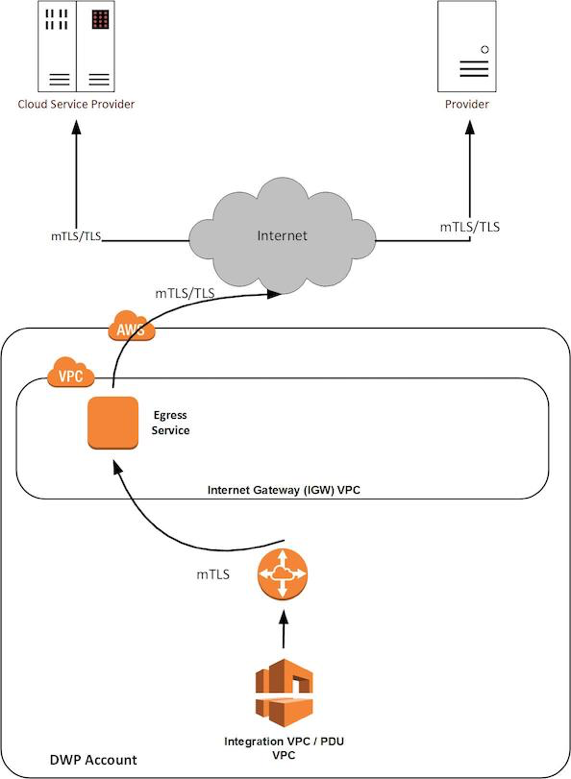
Architecture Call with Matthew Mann  
Following our workshop with Tom, we were told we had to run the architecture design past Matthew Mann, the architect for the Notifications project. Matthew accessed our proposal for compatibility with DWP data protection and highlighted the importance of keeping citizen information secure to comply with GDPR. He advised us to use a different pattern (fig. 6) instead of the transit gateway that we pitched, and he linked us to the correct pattern.   
[K3](#_Knowledge.)

Figure 6 - Internet Egress Pattern provided by Architect

End of the Project  
The project then was put on hold, as our Line Manager had been working to find some way to outsource the whole project, and finally found a solution that would require minimal work from us. It was an interesting learning journey, however and I took a lot from it. This cut down a lot of man-hours and freed up the team to focus on other more pressing tickets and bugs, enforcing the DevOps ideal of not wasting time and money.   
[K24](#_Knowledge.)

##### Knowledge.

K3 - How to use data ethically and the implications for wider society, with respect to the use of data, automation and artificial intelligence within the context of relevant data protection policy and legislation.

K19 - Different methods of communication and choosing the appropriate one - e.g. face-to-face (synchronous, high bandwidth), instant messaging, email (asynchronous, low bandwidth), visualisations vs. words.

K24 - The difference between Software-as-a-Service (SaaS) v bespoke v enterprise tooling and how to make an informed choice that suits each use case.

##### Skill.

S21 - Application of lightweight modelling techniques, such as whiteboarding, in order to gain consensus as a team on evolving architecture.

##### Behaviour.

Project #3

##### Ticket Name

NSE-2483

##### Ticket Description

After deploying the TMA-APP the pipeline should refresh the ASG so the new version of the APP would become available without further intervention.

##### Ticket Process

Meeting with Tom  
We went through what the ticket was asking, and Tom outlined the idea of creating a new step in the Gitlab CI pipeline which would manually refresh the Auto-Scaling groups (fig. 7). There were a few ways to successfully complete this task, but this seemed to me the most useful, and efficient way of creating a reusable piece of infrastructure. The idea to make the refresh a manual button was driven from a user experience standpoint, making it into an easy task executable by anyone.

Graphical user interface, text, application

Description automatically generated

Figure 7 - The desired result, showing completed manual 'refresh' jobs

Creating the script  
I wrote the initial script using some documentation that Tom had suggested I look at. The script had to do three major things: access the correct environment; assume the correct role; start the instance refresh. Having written that I hit various errors, that made me realise I’d have to install pip3 and the AWS CLI in the script as well.

Creating the fragment  
Once the script was working within the pipeline, refreshing the DSE1 ASG as expected, I could then get to work extracting the code to a fragment. The first thing I did was message my colleague Luis with my script, and some questions about fragments. He reviewed my code and suggested some changes, and other elements I might want to include in the process. He also linked some documentation and example repos with fragments.

I went away and did some learning, and then set up a call with myself and some other apprentices that I knew had worked with some fragments before (fig. 8). This was helpful, and we discussed the different uses of fragments between our teams. No one there had actually created one themselves yet, but after the knowledge sharing I felt more confident to go and experiment myself.   
[S4](#_Skill…), [B4](#_Behaviour…)Text

Description automatically generated

Figure 8 - Setting up a call with apprenticeship colleagues to share knowledge

I extracted the versatile code to a hidden job at the start of the GitLab CI file and used an ‘include’ statement in the ‘asg-refresh’ job I had set up. I then took out as much code from the refresh job to the hidden job, using variables where appropriate. Once I’d done that, I moved the hidden job to a file in another repo, to store it with other fragments. I then included the file by using the filepath at the start of the CI file, and extended the fragment in my original ‘refresh’ job. I tested the script still worked, then opened a merge request for both new additions.

Business Impact  
The fragment I created enables DevOps engineers to refresh the Auto-Scaling Groups manually, without having to redeploy the whole pipeline every time a minor change is made. It also means that SRE engineers don’t have to be involved as they normally would, making the whole workflow smoother (fig. 10). This embodies the DevOps value of building on internal quality, reducing waste and effort. A picture containing text

Description automatically generated

Figure 10 - a fellow DevOps engineer letting me know when he used my solution!

##### Knowledge…

##### Skill…

S4 - Initiate and facilitate knowledge sharing and technical collaboration

##### Behaviour…

B4 - Is inclusive, professional and maintains a blameless culture.

## KSBs

##### Knowledge

K3 How to use data ethically and the implications for wider society, with respect to the use of data, automation and artificial intelligence within the context of relevant data protection policy and legislation.

K6 A range of problem solving techniques appropriate to the task at hand, such as affinity mapping, impact maps, plan-do-check-act/Deming.

K9 Different organisational cultures, the development frameworks utilised and how they can both complement each other and introduce constraints on delivery.

K18 Roles within a multidisciplinary team and the interfaces with other areas of an organisation.

K19 Different methods of communication and choosing the appropriate one - e.g. face-to-face (synchronous, high bandwidth), instant messaging, email (asynchronous, low bandwidth), visualisations vs. words.

K20 Pair/mob programming techniques and when to use each technique.

K22 How their occupation fits into the wider digital landscape and any current or future regulatory requirements.

K23 The importance of continual improvement within a blameless culture.

K24 The difference between Software-as-a-Service (SaaS) v bespoke v enterprise tooling and how to make an informed choice that suits each use case.

K25 Maintain an awareness of cloud certification requirements.

##### Skill

S1 Communicate credibly with technical and non-technical people at all levels, using a range of methods; e.g. ‘Show and Tell’ and ‘Demonstrations’.

S2 Work within different organisational cultures with both internal and external parties

S4 Initiate and facilitate knowledge sharing and technical collaboration

S8 Work in agile, multi-disciplinary delivery teams, taking a flexible, collaborative and pragmatic approach to delivering tasks.

S13 Engage in productive pair/mob programming.

S16 Invest in continuous learning, both your own development and others, ensuring learning activities dovetail with changing job requirements. Keep up with cutting edge.

S21 Application of lightweight modelling techniques, such as whiteboarding, in order to gain consensus as a team on evolving architecture.

##### Behaviour

B1 Exhibits enthusiasm, openness and an aptitude for working as part of a collaborative community; e.g. sharing best practice, pairing with team members, learning from others and engaging in peer review practices.

B2 Invests time and effort in their own development, recognising that technology evolves at a rapid rate.

B4 Is inclusive, professional and maintains a blameless culture.