Tyler Shepherd

Version 0

DWP Digital & Makers Academy

A portfolio to demonstrate my competence across the Knowledge, skills, and behaviours for the DevOps apprenticeship

DevOps L4 Portfolio

Table of contents

[Introduction 2](#_Toc102044243)

[Who am I? 2](#_Toc102044244)

[DWP Digital and my Team: Who are they and why should you care? 2](#_Toc102044245)

[Tickets 3](#_Toc102044246)

[(NSE-1853): Increased the Throttle rate and burst limit for the API GW 3](#_Toc102044247)

[Description 3](#_Toc102044248)

[Details 3](#_Toc102044249)

[KSB’S 6](#_Toc102044250)

­­­­

# Introduction

## Who am I?

My name is Tyler Shepherd, and I am currently employed by DWP Digital to pursue a DevOps apprenticeship. My previous history within employment has been quite short due to my age (20 as of writing) however, I have already completed a L3 Infrastructure Technician apprenticeship at a previous company!

In this role I was employed as a Support Engineer who dealt with end users’ technological issues, and I was here for 2 years before coming to DWP Digital. I moved from this position as I wanted to move up on the ladder of IT and completing an apprenticeship in DevOps seemed the perfect opportunity, I had already done it once and loved it!

Before all of this, I dropped out of college to pursue the first apprenticeship as I was done with formal education due to health issues. Alongside this, I knew what I wanted to do and being in a classroom delaying my start of where I wanted to be didn’t help, so I went headfirst into an apprenticeship which challenge all my preconceptions about it. Now I’m all for it and glad to be working for DWP Digital

## DWP Digital and my Team: Who are they and why should you care?

I am Part of the Digital Shared Channel Experience, in the Notifications Team for DWP Digital.

The Department for Work and Pensions (DWP) is responsible for welfare, pensions, and child maintenance policy. It is the UK’s biggest public service department and help manage around 20 million customers and users

My team is an agile team led by my Line Manager Tom Farrow. Our team has daily stand-up meetings where we express the day tasks ahead, and any help/requests that are needed. We assist in helping the automation and ease of notifications for DWP Shared Channels.

For example, DWP sends out around 200 text messages per second, and this needs to be managed and up taken using DevOps as any downtime could result in hundreds of thousands of appointments, password confirmations, etc. being lost.

By being in DWP, I am in handle of sensitive data, so I must have secure Computer use / mobile use policy in place. For my computer and my accounts, I must use 2FA Services such as OKTA and Twilio. This is to make sure that there is a second layer of defence encase my password is compromised.

# Tickets

## (NSE-1853): Increased the Throttle rate and burst limit for the API GW

### Description

The following ticket came into mention during the daily stand up’s we as a team have. Each morning our team has a meeting in which we talk about what we are carrying out for the day, and any help that is needed. The following ticket was highlighted as a quick fix that I and another apprentice could solve as a quick fix.

The APIs listed below had throttling set to 100TPBs which is well below what was required. The reason in which the throttling needs to be so high is so to help protect them from being overwhelmed by too many requests.

This therefore meant we required an update across all our private API GWs to update the throttling value of 10,000 TPS, with a 5,000 burst.

This impacts all private APIs across all environments so once configured it should boost and slow down traffic congestion for the API’s

#### Details

**The following Repositories with APIGWs that needed changing**

* SMS Notifications API:
  + SMS-notification-api-validation
* Channel Adapter API:
  + channel-adapter-otx-api
* Transformation API:
  + transformation-api-json-validation
  + transformation-api-xml-validation
* Bulk Upload API:
  + message-builder-api
* Transaction Log API - APP side:
  + transaction-record-validator
* Transaction Log API - API side:
  + transaction-logging-api-validation
* Callback API:
  + callback-service
* Department Switch API:
  + department-switch-api
* Subscriber Management API:
  + subscriber-management-api
* Template Management API:
  + template-api
* Template Registry API:
  + template-registry-api

To carry out the changes, I split into pair programming with another apprentice in which we split the work in half and configure the TF code in all our APIGW to have those settings for Throttling by going to the relevant repositories with Terraform code in and editing the values present.

While editing each API values terraform code, we must follow the following procedure

1. Create a local branch from develop with the name **bugfix/nse-1853-api-throttle-increase**  
2. Develop your code locally as usual.  
3. Push to the repo with **git add .**and then the commit message should be something descriptive **like git commit -m "fix (NSE-1853): Increased the Throttle rate and burst limit for the API GW"**  
4. Then push to the repo and create a Merge Request. Assign the MR to team member and add the other apprentice and myself as Reviewers.

To fully understand terraform code and how it works, I read over the terraform documentation as well as the documentation for AWS API Throttling

[**https://www.terraform.io/docs**](https://www.terraform.io/docs)

[**https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-request-throttling.html**](https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-request-throttling.html)

Below is an example of carrying out one of the required API for changing in the terraform code.

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Within the merge request there is a GitLab CI/CD pipeline, this pipeline carry’s out the following.

**DSE1** -> Developer Supported Environment. This is the one where things are tested and broken. The tf-apply for DSE1 will trigger automatically on every merge into develop branch.

**SIT1** -> System Integration Testing environment, where the testers perform most of the tests. This environment can be deployed only during a tag pipeline.

**PVT1** -> Volume Testing, where the testers perform Volumetric tests. This environment can be deployed only during a tag pipeline, so a tag in the repo is created to deploy in PVT1.

**PTE1** -> First of the production Environments, this is outside my scope and managed by SRE, this is also called Staging.

**Production** -> The real Live environment again in a different AWS account managed by SRE team.

Carrying out these simple tasks allowed me to learn a lot, how the structure of repositories work in the DWP Environments. The pipeline’s used and how they interact with each other, testing/code reviewing processes, and working in a pair programming situation to evenly distribute work!

By completing the configuring of throttling to the API’s I was able to make sure that the API’s traffic load didn’t become overloaded by too many requests thus saving future availability.

Offload responsibilities to GDPR.

##### KSB’S

**Knowledge**

[K1](#K1)

[K2](#K2)

[K7](#K7)

[K20](#K20)

**Skills**

[**S5**](#S5)

[**S8**](#S8)

[**S13**](#S13)

**Behaviour**

[B1](#B1)

**Evidence**

**https://dwpdigital.atlassian.net/browse/NSE-1853**

|  |  |
| --- | --- |
| **K1** | Continuous Integration - the benefits of frequent merging of code, the creation of build artefacts and ensuring all tests pass, with  automation throughout - including common tooling. |
| **K2** | The principles of distributed Source Control, including how to exploit the features of the tool, such as branching. |
| **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. |
| **K4** | The business value of DevOps in terms of Time, Cost, Quality, with an emphasis on building in internal Quality throughout  the lifetime of the product. |
| **K5** | A range of modern security tools and techniques - e.g., threat modelling, vulnerability scanning and dependency checking, with a general  awareness of penetration testing - in order to deal with threats and attack vectors within code and across the cyber domain. |
| **K6** | A range of problem-solving techniques appropriate to the task at hand, such as affinity mapping, impact maps, plan-do-check-act/Deming. |
| **K7** | General purpose programming and infrastructure-as-code. |
| **K8** | Immutable infrastructure and how it enables continuous refreshing of software, namely the updating of the operating system,  container and security patching. |
| **K9** | Different organisational cultures, the development frameworks utilised and how they can both complement each other and introduce  constraints on delivery. |
| **K10** | How the user experience sits at the heart of modern development practices in terms of strategies to understand diverse user needs,  accessibility and how to drive adoption. |
| **K11** | Monitoring and alerting technologies and an awareness of the insights that can be derived from the infrastructure and applications -  collecting logs and metrics, configuring alerting thresholds, firing alerts, and visualising data. |
| **K12** | The persistence/data layer, including which database/storage technologies are appropriate to each platform type and application when  considering non-functional and functional needs, e.g., monolith, microservice, read heavy, write heavy, recovery plans. |
| **K13** | Automation techniques, such as scripting and use of APIs. |
| **K14** | Test Driven Development and the Test Pyramid. How the practice is underpinned by unit testing, the importance of automation,  appropriate use of test doubles and mocking strategies, reducing a reliance on end-to-end testing. |
| **K15** | The principles and application of Continuous Integration, Continuous Delivery and Continuous Deployment, including the  differences between them. |
| **K16** | How best to secure data, e.g., encryption in transit, encryption at rest and access control lists (ACL). |
| **K17** | What an API is, how to find them and interpret the accompanying documentation. |
| **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. |
| **K21** | Architecture principles, common patterns, and common strategies for translating user needs into both cloud infrastructure and  application code. |
| **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. |
|  |  |
|  |  |
|  |  |
| **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 |
| **S3** | Translate user needs into deliverable tasks, writing clear, concise, and unambiguous user stories that the whole team can understand. |
| **S4** | Initiate and facilitate knowledge sharing and technical collaboration |
| **S5** | Deploy immutable infrastructure |
| **S6** | Install, manage, and troubleshoot monitoring tools |
| **S7** | Navigate and troubleshoot stateful distributed systems, to locate issues across the end-to-end service. |
| **S8** | Work in agile, multi-disciplinary delivery teams, taking a flexible, collaborative, and pragmatic approach to delivering tasks. |
| **S9** | Application of a range of cloud security tools and techniques - e.g., threat modelling, vulnerability scanning, dependency checking,  reducing attack surface area - incorporating these tools and techniques into the automated pipeline wherever possible. |
| **S10** | Assess identified and potential security threats and take appropriate action based on likelihood v impact. |
| **S11** | Employ a systematic approach to solving problems, using logic and hypotheses / experimentation to identify the source of issues. |
| **S12** | Automate tasks where it introduces improvements to the efficiency of business processes and reduces waste, considering the effort  and cost of automation. |
| **S13** | Engage in productive pair/mob programming. |
| **S14** | Write tests and follow Test Driven Development discipline in various contexts. |
| **S15** | Release automation and orchestration as part of a Continuous Integration workflow and Continuous Delivery pipeline, automating  the delivery of code from source control to the end users. |
| **S16** | Invest in continuous learning, both your own development and others, ensuring learning activities dovetail with changing job  requirements. Keep up with cutting edge. |
| **S17** | Code in a general-purpose programming language. |
| **S18** | Specify cloud infrastructure in an infrastructure-as-code domain-specific language. |
| **S19** | Interpret logs and metrics data within the appropriate context to identify issues and make informed decisions. |
| **S20** | Writing code in such a way that makes merging easier and facilitates branching by abstraction - i.e., feature toggling. |
| **S21** | Application of lightweight modelling techniques, such as whiteboarding, to gain consensus as a team on evolving architecture. |
| **S22** | Incremental refactoring by applying small behaviour-preserving code changes to evolve the architecture. |
|  |  |
|  |  |
|  |  |
| **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. |
| **B3** | Displays a commitment to the mantra 'You build it, you run it', taking ownership of deployed code and being accountable for its  continual improvement, learning from experience and taking collective responsibility when things fail. |
| **B4** | Is inclusive, professional and maintains a blameless culture. |