Assessment Method 1 Project and Practical Assessment Guidance

Accelerate People Level 4 End-point Assessment for DevOps Engineer ST0825/AP02

Qualification Number 610/2605/0

Document Control Information

Document Details	
Document Name	Assessment Method 1 Project and Practical Assessment Guidance V1 Accelerate People Level 4 End-point Assessment for DevOps Engineer ST0825/AP02.
Purpose of Document	Guidance document for Assessment Method 1 of the DevOps Engineer Level 4 apprenticeship.
Document Version Number	1
Document Status	Live
Document Owner	Product
Next Scheduled Review Date	June 2024

Version History		
Version Number	**Date Amended**	**Changes Made**
1	June 2023	Document created.

Contents

Contents 3
Component One: Project 4
Component Two: Practical Assessment 4
Form 5
Technical Breadth 5
Specific Technical Outputs 7
Plan for the Practical Assessment 8
Practical Assessment Delivery Plan 10
Final considerations 15

Apprentices have 13 weeks to complete a post-gateway project and submit their project and code to Accelerate People. At least 10 working days after, the apprentice will complete a 3-hour practical assessment. This practical assessment will focus on the grading criteria for this assessment method. **You should already have the practical assessment date agreed.**

Component One: Project

As an outcome of the project the apprentice must produce sufficient evidence of the **form**, **technical breadth**, and **specific technical outputs** of their work as outlined below:

- **On 'form'**: an architectural diagram (in a structured or ad-hoc notation) or other artefact which shows high level system structure.
- **On 'technical breadth'**: a short analysis, maximum 300 words, of which project areas provide evidence against which KSBs.
- **On 'specific technical outputs'**: the independent assessor will need to be provided with implementations which cover all techniques used. These may include source code, deployment/system build scripts or configuration files and should be communicated to the independent assessor through access to cloud services, an archive of files or in screenshots/videos/documents.

The following could be included in the piece of code:

- Building a piece of infrastructure and deploying an application to it.
- Building an element of a platform, resident on this infrastructure.
- Development of a new approach to a platform/infrastructure/deployment problem, i.e., novel tooling where no alternates are available.
- Development of tooling to automate common deployment/maintenance processes.
- Development/implementation of new CI/CD pipelines.
- Development of management/support processes.

To ensure the practical assessment is successful, apprentices must plan how they will meet all requirements during the 13-week project window. Further guidance is listed in component two.

Component Two: Practical Assessment

The **minimum expectations** of the 3-hour practical assessment are as follows:

- Demonstrate the piece of code that has been produced.
- Operating a performant, secure, and highly available platform.
- Satisfy the functional and non-functional requirements defined by the work-based project.
- A successful deployment of code from source to the end user.
- Meets all eight themes and associated grading criteria for this assessment method.

Apprentices will take full control of the 3-hour practical assessment. The independent assessor will guide the demonstration and ask questions where appropriate.

Further guidance for both components is provided over the following pages.

Form

An architectural diagram is a visual representation of the components, relationships, and interactions that make up a system or app. It shows how data moves across the architecture and how it interacts with the environment it resides in.

Typically, an architectural diagram will include shapes to represent different components and arrows to show relationships or interactions. Colour schemes are not mandatory but may help visualise how the system or app is built in a logical way.

Architectural diagrams can be created in any way. For example, hand drawn or being built in a software or application.

Technical Breadth

The table below is a suggested template to use. The 300-word count is only relevant for the third column titled 'Evidence From Project.' This column should include location identifiers only e.g., page numbers, appendix numbers, document names, or document numbers.

Theme	Criteria	Evidence From Project (300 words max)
Code Quality (K2, K5, K7, K14, S9, S11, S14, S17, S18, S20, S22)	Writes code, both general purpose and infrastructure-as-code (including cloud infrastructure) that is correctly versioned and easy to merge, while adhering to the principles of distributed Source Control	
	Demonstrates an iterative approach to evolving code consistent with cloud security best practice, evidenced by a lack of vulnerabilities and that all dependent components are present at run time.	
	Writes code around unit tests, including the appropriate use of test doubles and mocking strategies.	
	Explains troubleshooting methods used to identify and resolve issues and gives an example of identifying and remediating an issue that compromised code quality.	
Meeting User Needs (K4, K10, K21, S3)	Writes user stories that are understandable to a wide range of stakeholders, stand up to scrutiny and lend themselves to a solution based on common architectural patterns - i.e. reducing the number of moving/redundant parts; passes all acceptance tests.	
	The piece of code meets the 'must have' identified functional/non-functional user needs encapsulated in the acceptance criteria for the task.	

	Creates a quality product in terms of Mean Time To Recovery (MTTR) - i.e. reduced time to fix bugs.	
	Distinction: Produces a piece of code that meets the 'should have' identified functional/non-functional user needs encapsulated in the acceptance criteria for the task.	
The CI-CD Pipeline (K1, K15, S15)	Builds a fully functioning, automated CI-CD pipeline with all tests passing.	
	Evidences a code commit progressing seamlessly from a build artefact to the end user.	
	Explains the pipeline capability, including the benefits of frequent merging of code, in terms of Continuous Integration/Delivery/Deployment.	
Refreshing and Patching (K8, S5)	Deploys immutable infrastructure that enables the regular recycling of servers and refreshing of associated software based on manual processes.	
	Distinction: Fully automates the refreshing and patching process.	
Operability (K11, S6, S19, B3)	Installs and manages monitoring and alerting tools that provide coverage of the infrastructure and applications, including RAM and CPU utilisation, application error rates and availability (health check).	
	Configures appropriate alerting thresholds and visualisations. Interprets these in terms of failure scenarios and remedial/follow up actions taken to deliver continuous improvement.	
	Distinction: Introduces custom metrics that provide additional improvement areas.	
	Distinction: Explains how these improvement areas may be interpreted, implemented and delivered	
Data Persistence (K12, S7)	Employs and operates an appropriate data persistence technology, such as database, configuration/infrastructure state management to meet non-functional and functional needs.	
	Explains troubleshooting steps taken to locate issues across the end-to-end service.	
Automation (K13, K17, S12)	Introduces process efficiencies by automating the setting up/deploying of the project (infrastructure and applications) from scratch, both locally, including all tests, and to a hosted environment.	
	Distinction: Identifies an additional opportunity and introduces automation that reduces overall effort.	

Data Security (K16, S10)	Builds in security so that all data in transit is encrypted and secure.	
	Explains the types of threats and the rationale behind the decision to either encrypt data at rest or not.	
		Total word count:

Specific Technical Outputs

This is where most of the evidence will exist for the project. The technical outputs are specific to the apprentice's own workplace and project. These outputs are unique and can cover the following:

- Source code.
- Deployment or system build scripts.
- Configuration files.

These should be communicated to the independent assessor in the following ways:

- Access to cloud services or archive of files.
- Screenshots, video demonstrations, or documents.

Any evidence produced must be related to the themes and grading criteria. An example is below using the following format:

- The theme and grading criteria are referenced separately with supporting practical evidence e.g., screenshots or videos.
- There is a narrative included which explains the work completed and how it relates to the project.
- There is a proposed plan on what will be covered within the 3-hour practical assessment for this grading criteria using the supplied evidence as a reference point.

Example: Theme - Meeting User Needs

Criteria: Creates a quality product in terms of Mean Time To Recovery (MTTR) - i.e. reduced time to fix bugs.

Sample evidence: Below is a screen shot for the Cypress Dashboard that I set up for the project. To do this I link the project ID from the Cypress dashboard back to the source code repo using the project ID in the cypress.json configuration file. You can see in Figure 2 how we as a team were able to run the full test suite and analyse the results in the dashboard after a change to the master branch (as shown in red). Ensuring that components and test names linked to requirements outlined on Jira using agreed internal naming conventions means that any failed tests could be easily traced back to specific components within the build. The testing frameworks (Cypress and Jest) that we explored during the workshops were new to the company, so part of my job was to ensure that we got the most out of each in terms of reducing MTTR.

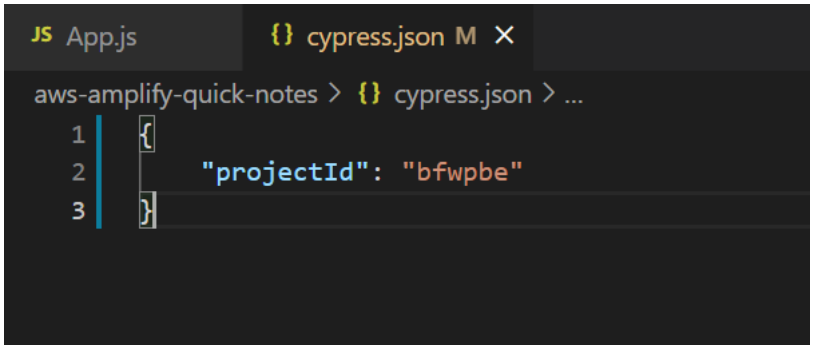

Figure 1 - Linking codebase to Cypress test suite dashboard.

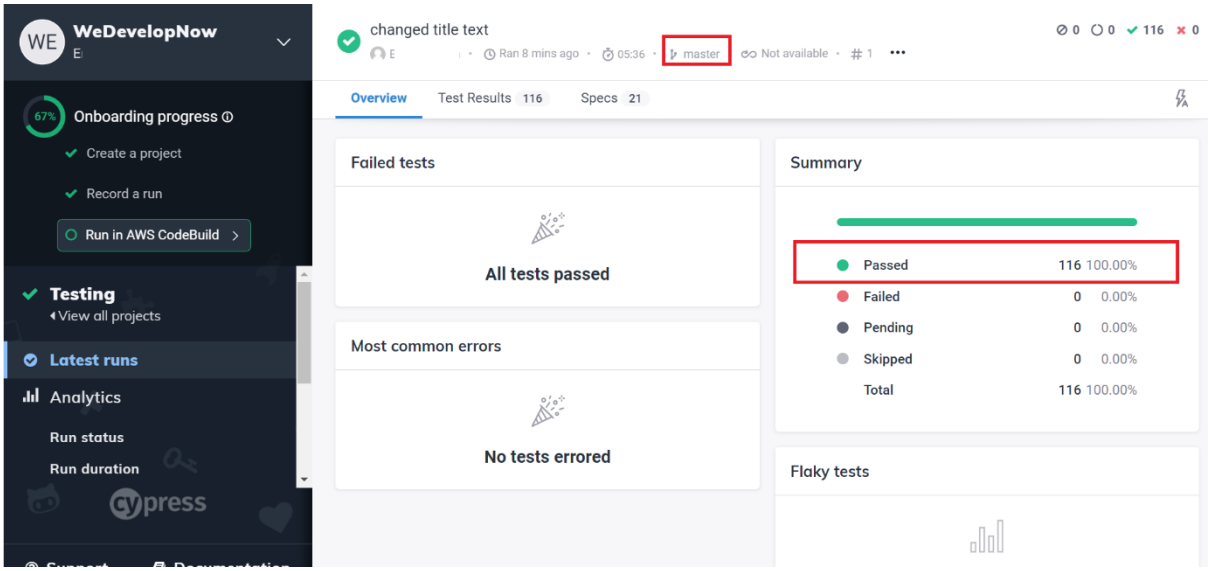

Figure 2 - All tests passing after update to master branch.

Plan for practical assessment: In the practical assessment, I plan to do a full walkthrough on how functional and non-functional requirements were automated into the build pipeline. I will consider how the logs from both testing frameworks and the pipeline are used to ensure a high-quality product is delivered. This links to the themes on CI/CD Pipeline, Automation, and Code Quality. I plan to show how I have helped contribute to building a library of internal regression tests that can be used in future client projects.

Plan for the Practical Assessment

The practical assessment will cover the following:

- Operating a performant, secure and highly available platform.
- Satisfy the functional and non-functional requirements defined by the work-based project.
- Meets the KSBs mapped to this assessment method.

The following task must be observed by the independent assessor during the 3-hour practical assessment:

- A successful deployment of code from source to the end user.

This is not a mandatory requirement but is provided as a suggestion to help with assessment preparation	
Action	**Preparation**
Operating a performant, secure and highly available platform.	This should be discussed on the day of the practical assessment and the code submitted with the project should be presented along the other outputs to **demonstrate availability of the built infrastructure**. Any considerations for security (dependencies, built assets, and environment variables. Also, any integration across platforms or front and backend and how this is considered. This may include preparation for providing security through available libraries, services, and resources within cloud platforms (e.g., AWS Identity and Access Management (IAM) to manage access control) can be outlined and documented). In total, the context of the code provided with the project should be discussed. The code should be used to define and build a working pipeline to **demonstrate a performant platform.**
Satisfy the functional and non-functional requirements defined by the work-based project.	The overall environment conforms to the **'Form'** component of the project. The practical evidence showing the CI/CD pipeline will support the architectural diagram when introduced during the practical assessment. The code should link to building the correct infrastructure that satisfies the requirements of the project in relation to their role. Prepare to discuss and validate this during the practical assessment.
Meets the KSBs mapped to this assessment method.	The apprentice's submission for the **'Technical Breadth'** component of the project will support this. Apprentices should ensure they know all the themes (eight in total) that need to be met, plus the associated grading criteria for each. Evidence should be supplied that covers all grading criteria as set out in the **'Specific Technical Outputs'** component of the project.
A successful deployment of code from source to the end user.	Evidence of this must be supplied as part of the **'Specific Technical Outputs'** component of the project.

	During the practical assessment, apprentices **must** have access to systems that will enable them to demonstrate how they have been involved in the deployment and CI/CD process. Apprentices should show and explain contributions to kicking off a code deployment (this could be a demo of everything up to manually starting a deployment or discuss automation and scheduling to do this). Apprentices can then use the system to explain the process referring to practical examples if needed i.e., walking through the stages of a pipeline and show a successful deployed build and explain the process of how a built environment is made available to the end user. Apprentices **must** be able to show within the system a successful deployment for the independent assessor to consider after discussing their contribution to trigger this. Preparation in advance could take the shape of a condensed video or documentation to outline the process. **However, this will need to be observed by the independent assessor live during the 3-hour practical assessment.**

Practical Assessment Delivery Plan

Apprentices should come prepared for the 3-hour practical assessment to ensure they present and cover all themes in a timely manner. The independent assessor will guide the conversation based on the evidence apprentices have supplied in their project and demonstrated during the practical demonstration.

Important: Independent assessors can ask a maximum of 16 questions (two per theme) throughout the entire 3-hour period. Apprentices should ensure they plan to cover all grading criteria in their submitted evidence and during the 3-hour practical assessment.

A clear structure and plan will help the apprentice to prepare and identify how best to present their evidence.

Apprentices should structure their project evidence to focus on each of the themes included in the assessment plan. To do this, apprentices may wish to use the table below to document and build a delivery plan:

This is not a mandatory requirement but is provided as a suggestion to help with assessment preparation			
Theme	**Criteria**	**Action**	**Duration** (3 hours total)
Code Quality (K2, K5, K7, K14, S9, S11, S14, S17, S18, S20, S22)	Writes code, both general purpose and infrastructure-as-code (including cloud infrastructure) that is correctly versioned and easy to merge, while adhering to the principles of distributed Source Control	*Briefly outline how you plan to cover this criteria. Consider how best to use time to show evidence and answer questions. An example of your planning could be:* • *When demonstrating the pipeline built, I will discuss the infrastructure-as-code and how this contributed to the environment.* • *Demonstrate the unit tests as part of the pipeline to show my contribution to the general-purpose code created as part of the build process (link to criteria below).* • *Whilst doing the build, I will show how the code has been managed and versioned within GitHub.*	
	Demonstrates an iterative approach to evolving code consistent with cloud security best practice, evidenced by a lack of vulnerabilities and that all dependent components are present at run time.		
	Writes code around unit tests, including the appropriate use of test doubles and mocking strategies.		
	Explains troubleshooting methods used to identify and resolve issues and gives an example of identifying and remediating an issue that compromised code quality.		
Meeting User Needs	Writes user stories that are understandable to a wide range of		

(K4, K10, K21, S3)	stakeholders, stand up to scrutiny and lend themselves to a solution based on common architectural patterns - i.e. reducing the number of moving/redundant parts; passes all acceptance tests.		
	The piece of code meets the 'must have' identified functional/non-functional user needs encapsulated in the acceptance criteria for the task.		
	Creates a quality product in terms of Mean Time To Recovery (MTTR) - i.e. reduced time to fix bugs.		
	Distinction: Produces a piece of code that meets the 'should have' identified functional/non-functional user needs encapsulated in the acceptance criteria for the task.		
The CI-CD Pipeline (K1, K15, S15)	Builds a fully functioning, automated CI-CD pipeline with all tests passing.		
	Evidences a code commit progressing seamlessly from a build artefact to the end user.		
	Explains the pipeline capability, including the benefits of frequent merging of code, in terms of Continuous Integration/Delivery/Deployment.		
Refreshing and Patching (K8, S5)	Deploys immutable infrastructure that enables the regular recycling of servers and refreshing of associated software based on manual processes.		
	Distinction: Fully automates the refreshing and patching process.		
Operability	Installs and manages monitoring and alerting tools that provide coverage of		

(K11, S6, S19, B3)	the infrastructure and applications, including RAM and CPU utilisation, application error rates and availability (health check).		
	Configures appropriate alerting thresholds and visualisations. Interprets these in terms of failure scenarios and remedial/follow up actions taken to deliver continuous improvement.		
	Distinction: Introduces custom metrics that provide additional improvement areas.		
	Distinction: Explains how these improvement areas may be interpreted, implemented and delivered		
Data Persistence (K12, S7)	Employs and operates an appropriate data persistence technology, such as database, configuration/infrastructure state management to meet non-functional and functional needs.		
	Explains troubleshooting steps taken to locate issues across the end-to-end service.		
Automation (K13, K17, S12)	Introduces process efficiencies by automating the setting up/deploying of the project (infrastructure and applications) from scratch, both locally, including all tests, and to a hosted environment.		
	Distinction: Identifies an additional opportunity and introduces automation that reduces overall effort.		
Data Security	Builds in security so that all data in transit is encrypted and secure.		

(K16, S10)	Explains the types of threats and the rationale behind the decision to either encrypt data at rest or not.		

Final considerations

Here are a few final points to consider and to clarify for this assessment method.

- Apprentices have 13 weeks to complete their project and submit all required documentation; apprentices are unable to introduce new written evidence after this deadline.
- The practical assessment will run for a maximum of 3-hours.
    - Apprentices will deliver the practical assessment. The independent assessor will guide the discussion and questioning.
- The grade for this assessment method is based on all evidence provided, including questions and answers.
- The templates supplied in this document are for guidance. Apprentices can use any format.