QBN1 — QBN1 TASK 3: PROGRAM DEPLOYMENT

DEPLOYMENT — D602
PRFA — QBN1

COMPETENCIES

4162.1.3 : Implements a Function
The learner implements a function to call and receive information between multiple systems for deployment.

4162.1.4 : Deploys a Data Based Product
The learner deploys a data product based on project requirements.

INTRODUCTION

The data analyst's project is not complete until they have guided it through the process of bringing a data product from development to deployment.

In this task, you will write an API in either Python or R in GitLab. You will then create a Dockerfile that packages your API code and runs a web server to allow HTTP requests to your API. You will then explain how you wrote your code and the challenges you overcame. Finally, you will provide a video demonstrating the live API running from a deployed Docker container.

SCENARIO

Your leadership at the airline is pleased with the machine learning pipeline you generated and has passed it to other business units. To make route planning easier, the operations team has asked you to deploy the model in a format that is easy to integrate with other software tools they are already using. Specifically, they have asked you to create an API to provide departure delay predictions given a departure airport, an arrival airport served from that departure airport, a departure time, and an arrival time. This API should be deployed via a web server so that other software tools can send HTTP requests to the API and receive responses in JSON format. Both the model API and the web server should be deployed together using a Docker image, which is the company standard for deploying models for testing purposes. Also standard is implementation of unit testing of code to ensure it runs properly before including it in any Docker images.

Your code should be stored in the GitLab repository created for you, and you should provide multiple updates of your code within the repository so that users in other business units can track the changes you make to the code over time. Your GitLab repository has automation features implemented that will check your code upon submission, create a Docker image and store it in the GitLab Docker container registry, and then deploy that image to a running container. The live API may be accessed using the URL generated when the GitLab automation runs.

REQUIREMENTS

Your submission must be your original work. No more than a combined total of 30% of the submission and no more than a 10% match to any one individual source can be directly quoted or closely paraphrased from sources, even if cited correctly. The similarity report that is provided when you submit your task can be used as a guide.

You must use the rubric to direct the creation of your submission because it provides detailed criteria that will be used to evaluate your work. Each requirement below may be evaluated by more than one rubric aspect. The rubric aspect titles may contain hyperlinks to relevant portions of the course.

*Tasks may **not** be submitted as cloud links, such as links to Google Docs, Google Slides, OneDrive, etc., unless specified in the task requirements. All other submissions must be file types that are uploaded and submitted as attachments (e.g., .docx, .pdf, .ppt).*

A. Create your subgroup and project in GitLab using the provided web link and the "GitLab How-To" web link by doing the following:

- Clone the project to the IDE.
- Commit with a message and push when you complete each requirement listed in parts B and D.

  Note: You may commit and push whenever you want to back up your changes, even if a requirement is not yet complete.

- Submit a copy of the GitLab repository URL in the "Comments to Evaluator" section when you submit this assessment.
- Submit a copy of the repository branch history retrieved from your repository, which must include the commit messages and dates.

B. Write an API with the code templates provided in either the FastAPI package in Python or the plumber package in R that accepts the following HTTP endpoints. Submit *at least* **two** versions of your code to the GitLab repository demonstrating a progression of work on your code.

1. "/" should return a JSON message indicating that the API is functional.
2. "/predict/delays" should accept a GET request specifying the arrival airport, the local departure time, and the local arrival time. It should return a JSON response indicating the average departure delay in minutes.

C. Write *at least* **three** unit tests for your API code, using the pytest package in Python or the testthat package in R, that test features of endpoints given *both* correctly formatted and incorrectly formatted requests. Submit *at least* **two** versions of your code to the GitLab repository demonstrating a progression of work on your code.

D. Write a Dockerfile referencing the requirements.txt file as appropriate that packages your API code and runs a web server to allow HTTP requests to your API. Submit *at least* **two** versions of your Dockerfile to the GitLab repository demonstrating a progression of work on your code.

E. Provide an explanation of how you wrote your code, including any challenges you encountered and how you addressed those challenges.

F. Provide a video demonstrating the live API running from a deployed Docker container. You must issue *at least* 1 well-formatted request and 1 ill-formatted request and demonstrate that the API responds appropriately.

G. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

H. Demonstrate professional communication in the content and presentation of your submission.

File Restrictions

File name may contain only letters, numbers, spaces, and these symbols: ! - _ . * ' ( )
File size limit: 200 MB
File types allowed: doc, docx, rtf, xls, xlsx, ppt, pptx, odt, pdf, csv, txt, qt, mov, mpg, avi, mp3, wav, mp4, wma, flv, asf, mpeg, wmv, m4v, svg, tif, tiff, jpeg, jpg, gif, png, zip, rar, tar, 7z

RUBRIC

A:GITLAB REPOSITORY

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A GitLab repository is not provided.	The subgroup and project are created in GitLab, but 1 or more of the given actions are not completed, or they are completed incorrectly.	The subgroup and project are created in GitLab correctly, and all of the given actions are completed correctly.

B1:API ENDPOINT "/"

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
An API code is not provided.	In the API code provided, "/" does not return a JSON message.	In the API code provided, "/", returns a JSON message indicating that the API is functional.

B2:API ENDPOINT "/PREDICT/DELAYS"

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
An API code is not provided.	In the API code provided, "/predict/delays" does not provide arrival airport, local departure time, local arrival time, or a	In the API code provided, "/predict/delays" provides the arrival airport, the local departure time, the local arrival time, and a

	JSON response indicating the average departure delay in minutes.	JSON response indicating the average departure delay in minutes.

C:API TEST

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
Unit tests for the API code are not provided.	Only 1 or 2 unit tests were written, or the tests do not feature correctly formatted requests, or the tests do not feature incorrectly formatted requests. Or only 1 version of the code is provided.	3 unit tests are written, and they contain *both* correctly and incorrectly formatted requests. *At least* 2 versions of the code are provided.

D:DOCKERFILE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A Dockerfile is not provided.	The Dockerfile does not package the API code or does not run a web server to allow HTTP requests to the API. Or only 1 version of the Dockerfile is provided.	The Dockerfile packages the API code and runs a web server to allow HTTP requests to the API. *At least* 2 versions of the Dockerfile are provided.

E:EXPLANATION

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
An explanation of how the code was written is not provided.	The explanation does not include how *all* the code was written, or it does not address challenges that were encountered, or it does not explain how the challenges were addressed, or it contains errors.	The explanation of how *all* the code was written, including challenges encountered and how they were addressed, is logical and free of errors.

F:DEMONSTRATION

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A video demonstrating the live API running is not provided.	The video demonstration does not show 1 ill-formatted re-	A video demonstration shows 1 ill-formatted request and 1 well-

	quest, or it does not show 1 well-formatted request, or the API does not respond appropriately.	formatted request, and the API responds appropriately.

G:SOURCES

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The submission does not include both in-text citations and a reference list for sources that are quoted, paraphrased, or summarized.	The submission includes in-text citations for sources that are quoted, paraphrased, or summarized and a reference list; however, the citations or reference list is incomplete or inaccurate.	The submission includes in-text citations for sources that are properly quoted, paraphrased, or summarized and a reference list that accurately identifies the author, date, title, and source location as available.

H:PROFESSIONAL COMMUNICATION

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
Content is unstructured, is disjointed, or contains pervasive errors in mechanics, usage, or grammar. Vocabulary or tone is unprofessional or distracts from the topic.	Content is poorly organized, is difficult to follow, or contains errors in mechanics, usage, or grammar that cause confusion. Terminology is misused or ineffective.	Content reflects attention to detail, is organized, and focuses on the main ideas as prescribed in the task or chosen by the candidate. Terminology is pertinent, is used correctly, and effectively conveys the intended meaning. Mechanics, usage, and grammar promote accurate interpretation and understanding.

WEB LINKS

- WGU GitLab Environment FAQ
- Bureau of Transportation Statistics
- GitLab Environment