Ch11

CI/CD on Google Cloud for Data Engineers

CHAPTER ASSIGNMENT

Name _____ Class _____ No. _____ Date _____

> Answer every question in the space given. Show your working. You may use the chapter page to help you.

Questions from *Data Engineering with Google Cloud Platform*, 1st edition, Adi Wijaya (Packt, 2022), and from the Guru Jazzy chapter page, for classroom use.

Exercises from the book

Ex1 **Exercise ? implementing continuous integration using Cloud Build**

Create a Continuous Integration (CI) pipeline using Cloud Source Repository, Cloud Build, and Google Container Registry (`chapter-11/code/cicd_basic`):

(1) Creating a GitHub repository using Cloud Source Repository: create a new repository in Cloud Source Repository and clone it to Cloud Shell (`gcloud source repos clone [your repository name] --project=[your project id]`).

(2) Developing the code and Cloud Build scripts: copy `calculate.py`, `Dockerfile`, `cloudbuild.yaml`, and `tests/test_calculate.py` into your repository. Explain what each of the three steps in `cloudbuild.yaml` (`Build Image`, `Validation Test`, `Push Image to GCR`) does, and what `name`, `id`, `entrypoint`, and `args` specify.

(3) Creating the Cloud Build trigger: create a trigger in Cloud Build invoked on Push to a branch (`.*`) for your Cloud Source Repository.

(4) Pushing the code to the GitHub repository: commit and push the code, verify the build in Cloud Build History and the image in Container Registry, then intentionally change `result = arg_1 + arg_2` to `result = arg_1 * arg_2` in `calculate.py` and push again. Describe what happens when `Validation Test` fails.

Ex2 **Exercise ? deploying Cloud Composer jobs using Cloud Build**

Create a CI/CD pipeline in Cloud Build that validates and deploys a Cloud Composer Airflow DAG (`chapter-11/code/cicd_cloud_composer`):

(1) Preparing the CI/CD environment: create a Cloud Source Repository, clone it in Cloud Shell, and connect a Cloud Build Trigger.

(2) Preparing the `cloudbuild.yaml` configuration file: explain the role of each of the four steps in `cloudbuild.yaml` (`Build Airflow DAGs Builder` with `--cache-from`, `Validation Test` running `tests/dag_tests.py` with `AIRFLOW__CORE__DAGS_FOLDER=/workspace/dags`, `Push Image to GCR`, and `Deploy DAGs` using `gsutil -m rsync -r -c -x`).

(3) Pushing the DAG and checking the CI/CD result: push the code and verify that `level_1_dag.py` is synced to the Cloud Composer GCS `dags/` folder. Then change `schedule_interval='0 5 * * *'` to `schedule_interval='This is wrong'` and push again. Record the error message in the Cloud Build logs and explain how this protects the production Airflow environment.

Sorting activities from the chapter page

S1 **Two practices, not one**

Continuous integration automatically brings many engineers' code together and checks it. Continuous deployment automatically ships the final code to production. They are related but independent.

Options: **CI** ? **CD**

No.	Scenario	Answer
1	Run the unit tests automatically on every push.	
2	Copy the validated DAG file into the Composer environment’s dags/ folder.	
3	Build a container image containing the project’s dependencies.	
4	Roll back the production application automatically when an issue is detected.	

S2 **Does CI/CD fit here?**

Any code-based data application can use it. Fully UI-driven tools cannot. And you do not always need it.

Options: **Fits** ? **Does not fit**

No.	Scenario	Answer
1	Airflow DAG files in a repository, edited by several engineers.	
2	A dashboard built entirely by clicking around in a UI-only reporting tool.	
3	Terraform scripts for the data platform.	
4	Apache Beam pipeline code for Dataflow and a PySpark job for Dataproc.	

S3 **Does CI/CD fit here?**

Any code-based data application can use it. Fully UI-driven tools cannot. And you do not always need it.

Options: **Worth it** ? **Overkill**

No.	Scenario	Answer
1	A solo data engineer building a short-term project that will rarely change.	
2	A large team with hundreds of commits a day on shared pipelines.	

Quiz from the chapter page: Check yourself

1 What is the main benefit of continuous integration?

A. Code is never reviewed

B. Changes from many engineers are integrated and checked quickly, so errors are found early and easy to locate

C. It replaces Git

D. It reduces cloud costs to zero

2. Do you have to implement both CI and CD?

A. Yes, they are one indivisible thing

B. No: teams often automate CI but deploy to production by hand because automating deployment feels too risky

C. Only CD is needed

D. Neither is ever optional

---

3. In a Cloud Build pipeline a test step fails. What happens next?

A. The remaining steps run anyway

B. The run stops, so later steps such as deployment never execute

C. The failed step is skipped

D. Composer is restarted

---

4. Why is the "push image" step optional when deploying DAGs to Composer?

A. Composer needs an image

B. A DAG is deployed as a Python file copied to Cloud Storage, so the image is only kept for reuse as a build cache

C. Images are not allowed

D. DAGs deploy from BigQuery

---

5. What does the name parameter of a Cloud Build step specify?

A. A label for the log

B. Which container image (environment) runs that step

C. The build owner

D. The region

---

6. As a data engineer, who usually builds the whole CI/CD system?

A. The data engineer, always

B. Usually a DevOps or infrastructure team; you need to understand the concept and follow the system

C. The CFO

D. Nobody

End of assignment · Check your answers against the chapter at gurujazzy.com/subjects/gcp-data-engineering/chapter-11-cicd-for-data-engineers/