Github link: <https://github.com/HaydenDuong/SIT323_Cloud_Native_Application_Development/tree/main/5.2D>

Note: a README.md specific to this task is placed within folder “5.2D”

A computer screen with white text

AI-generated content may be incorrect.

Accessing the assigned Google Cloud Project by the following steps:

1. Log in Google Cloud via Google Cloud SDK Shell through command “gcloud auth login”.
2. Enter Deakin Email account in the pop-up Google login page.
3. Enter command “gcloud project list” to check which projects are assigned to the login accound.
4. Type in “gcloud config set project <project\_id>”, in this case, “sit323-25t1-duong-10255c” to switch to that project.
5. (Optional) Re-check the current project by “gcloud config get-value project”.

Using Artifact Registry

1. Create a repository:
   * Type for “Artifact Repository” on Search box and click enable API.
   * Click on the Pin icon right-next to “Artifact Registry” upon the page as shown in the image below (Image 1).

A screenshot of a computer

AI-generated content may be incorrect.

Image 1 – “Artifact Registry” main page

* + Choose “Create repository” with the following selections:
    1. Name: “calculator-repo”.
    2. Format: “Docker”.
    3. Mode: “Standard”.
    4. Location Type: “Region”.
       1. Region: “australia-southeast2 (Melbourne)”
       2. Description: “Repository for Calculator application (Task 4.2C).”.
    5. Leave other selections as their initial state and click “Create” button.

A screenshot of a computer

AI-generated content may be incorrect.

Image 2 – calculator-repo is successfully created.

1. Access permission to created Artifact Registry

* (Optional) Checking the current active account with “gcloud auth list”. (Image 3)

A screen shot of a computer

AI-generated content may be incorrect.

Image 3 – Checking the current active account.

* Artifact Registry is a private registry which will requiring permission-granted users and systems to be able to push and pull image from repositories created by this method.
* To grant that permission to client application, Docker, enter the following command: “gcloud auth configure-docker <Region\_selected>-docker.pkg.dev”, which is **australia-southeast2** in this case.
  + This command will using the login account, from “gcloud auth login” to create a credential helper and store the credential info into configuration file of Docker (~/.docker/config.json) so that whenever Docker application need to interact with Google Cloud like push and pull images, Docker will “contact” with this platform via Credential Helper “gcloud” to get the access token. (Image 4)
  + Note: this permission is only granted to the input region in the command, and will be required to re-enter it again with new <Region\_selected> input that matching the target repository in Artifact Registry.

A computer screen with white text

AI-generated content may be incorrect.

Image 4 – Authorize the accessibility of local Docker to Artifact Registry repository.

1. Pushing Docker Image to Artifact Registry (AR) repository

* (Optional) Re-checking if the target image is still present in the local host by starting up Docker Desktop & run command “docker images” in the terminal. (Image 5)

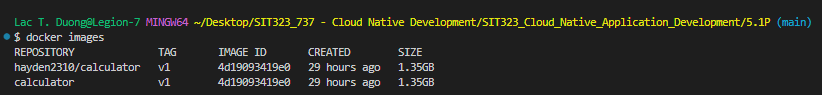


Image 5 – Images that are currently present in localhost

* Change the current tag of the target image to meet Artifact Registry standard which is “<Region\_selected>-docker.pkg.dev/<project-id>/<AR-repo-name>/<image-name>:<tag>” by command “docker tag …” (Image 6)

A screen shot of a computer code

AI-generated content may be incorrect.

Image 6 – New image is created through “docker tag …” command.

* Push the newly created image to “calculator-repo” created on Google Cloud platform through command “docker push <image-name>”. (Image 7, 8)

A screen shot of a computer

AI-generated content may be incorrect.

Image 7 – Image of the original “calculator” had successfully pushed to Google Cloud.

A screenshot of a computer

AI-generated content may be incorrect.

Image 8 – The image is found in “calculator-repo”.

1. (Optional) Pulling the pushed image back to local host
   * Remove the newly created (tagged) image in the localhost through command “docker rmi <image-name”.
   * The pushed image in Google Cloud can be pulled by one of these two ways:

A close-up of a computer screen

AI-generated content may be incorrect.

Image 9 – Ways to pull the image down to the local host.

* Results (Image 10, 11):

A black screen with colorful text

AI-generated content may be incorrect.

Image 10 – Image pulled by “Pull by tag” method. (Ideally)

A black screen with many colorful text

AI-generated content may be incorrect.

Image 11 – Image pulled by “Pull by digest” method.

* It was found that pulling the image via “Pull by tag” is more convenient because the pulled image will tagged with the designated tag represents in input docker command – make it easier to be interacted with (v1 vs @sha256:0f3b0761ca471aad2300579c870c7a179c58c6c125fa2dff84241dc91d36ea28).

1. Build a container from the pulled image
   * Use command “docker run …” upon the pulled image to generate a container from this image. (Image 12)



Image 12 – A container is successfully created from the pulled image.

* Check the container with Docker Desktop application to make sure it is running and be able to access to through the provided link in “Port(s)” section of this container. (Image 13).

A computer screen shot of a black screen

AI-generated content may be incorrect.

Image 13 – A container named “image-from-gc”, generated from pulled image is currently running and accessible via “https:localhost:3041”