**Jose Rizal Memorial State University**

**Gov. Guading Adaza St., Sta. Cruz,**

**Dapitan City, Philippines**

**GSU Gateway User Manual: Updating Docker Setup After New GitHub Code Push**

By:

Edrean Balonga

Bernadette Bantillan

Bryan Bantillan

Patrick Joshua Nicholas

Jenifer Saguindang

May 19, 2025

**General Service Office**

**Overview**

This user manual provides clear, step-by-step instructions for updating the Dockerized environment of the **GSU Gateway web application** after new code changes are pushed to GitHub. It is intended for developers, deployment specialists, and system administrators involved in maintaining the live or development environment of the application.

**Purpose**

The goal of this manual is to ensure a smooth and reliable process for syncing the local or production Docker setup with the latest version of the source code. It helps prevent configuration drift, broken builds, and outdated container states, ensuring your services always reflect the latest committed code from the repository.

**Importance**

Keeping your Docker containers updated with the latest pushed code is critical for:

* Ensuring all bug fixes, features, and security updates are live
* Avoiding compatibility issues between frontend and backend services
* Preventing deployment of outdated assets or broken builds
* Supporting team collaboration and CI/CD workflows

Without properly updating Docker images and containers, your environment may run stale code, leading to debugging confusion, service inconsistency, or even production downtime.

**Prerequisites**

Before using this manual, make sure you have the following:

* Git installed and configured on your local machine
* Docker Desktop installed and running
* Access to the GSU Gateway GitHub repository
* Terminal or shell access to the environment where Docker is set up

**Tools Needed**

* Docker Desktop installed and running
* Git installed
* Docker Compose configured (from initial setup)
* Access to the GitHub repository

*Step 1: Pull Latest Code from GitHub*

Open your terminal and run the following codes one at a time:

Cd C:\Users\DELL\Downloads\project-docker\GS

git clone -b <branch-name>

https://github.com/<username>/<repository-name>.git

Replace C:\Users\DELL\Downloads\project-docker\GS with the path to the folder you want the clone code to be saved, and replace the URL with the actual git repository HTTPS URL of your repository with your actual branch name. This updates your local copy with the latest changes from GitHub.

*Step 2: Rebuild Docker Images and Restart the Docker Containers*

Since the code changed, you need to stop, remove, and clean the previous docker container and networks, then rebuild it again, and bring the services back up with the updated images. Use the code below:

Docker-compose down

Docker-compose up –build -d

This will reprocess the Dockerfile for all services, incorporating any new code. The -d means "detached mode" (runs in background) and it starts the containers with the latest images.

*Step 3: Verify the Update*

Check if the containers are running correctly through opening your docker app, and click the **localhost URL with a port.** The front-end should load or it should show the welcome page or login page of your frontend, and there should be no error messages in the browser console or white screen. And for backend, it should also show the Laravel welcome page, API response, or JSON (if it's an API) or a status code 200 OK or relevant success message.

You can also run:

docker compose logs -f

To view logs and confirm the new version is active. If there is an error in the container, fix it.

**Common Docker Commands**

The following are common docker commands that can be use for managing and troubleshooting Docker containers:

| **Step** | **Command** | **Description** |
| --- | --- | --- |
| *1. Change Directory* | cd C:\Users\DELL\Downloads\project-docker\GS | Navigate to the directory where you want to clone the project. |
| *2. Clone Repo* | git clone -b <branch-name> https://github.com/<username>/<repository-name>.git | Clone the latest code from GitHub. Replace <branch-name>, <username>, and <repository-name>. |
| *3. Stop & Remove Containers* | docker-compose down | Stops and removes current containers and networks. |
| *4. Rebuild & Start Containers* | docker-compose up --build -d | Rebuilds images and starts containers in detached mode (background). |
| *5. Check Logs* | docker compose logs -f | Streams logs from containers to verify successful startup or detect errors. |
| *6. View Running Containers* | docker ps | Lists all currently running containers. |
| *7. Stop Running Containers* | docker stop <container-name> | Stops a specific running container. |
| *8. Start a Stopped Container* | docker start <container-name> | Starts a previously stopped container. |
| *9. Restart a Container* | docker restart <container-name> | Restarts a container. |
| *10. Remove a Container* | docker rm <container-name> | Removes a stopped container from your system. |
| *11. View Container Logs* | docker logs <container-name> | Displays the logs of a specific container. |
| *12. Access Container Terminal* | docker exec -it <container-name> /bin/bash | Opens an interactive terminal session inside a running container. |
| *13. View All Containers (Running and Stopped)* | docker ps -a | Lists all containers, whether running or stopped. |
| *14. List All Docker Images* | docker images | Shows all the Docker images on your machine. |
| *15. Remove a Docker Image* | docker rmi <image-name> | Removes a Docker image from your local system. |

### ****sConclusion****

This user manual ensures that all developers and deployment team members can reliably update the Dockerized environment of the GSU Gateway web application after new code has been pushed to GitHub. By following the outlined steps—pulling the latest code, rebuilding Docker images, and verifying the application status—you can keep both the front-end and back-end services in sync with the latest development changes.

Maintaining up-to-date containers is essential for avoiding deployment errors, minimizing downtime, and ensuring users and team members always interact with the most current version of the application. With this repeatable and structured process, you help preserve development integrity, reinforce CI/CD practices, and reduce the risk of service inconsistencies across environments.

By using this guide consistently, your team supports a streamlined and professional DevOps workflow for the GSU Gateway project.