CI/CD pipeline overview:

This documentation will explain the purpose and automation of all workflows provided, and any dependencies, services and technologies used and comparisons to alternatives.

Aim: Continuous Integration and Continuous Delivery pipelines are a vital tool used in DevOps. Continuous integration involves the ongoing process of integrating work, enabling quick detection of issues and reducing problems downstream (Vergardia, 2025). Continuous deployment (CD) is an extension of continuous integration, referring to preparing and packaging software. CD enhances the safety and efficiency of release processes (Vergardia, 2025). In DevOps, the cloud collaborates with DevOps by offering a service platform for businesses to establish CI/CD pipelines and supply infrastructure that accelerates DevOps processes (BrowserStack, 2025). Implementing DevOps pipelines through the cloud can boost development efficiency while providing continuous feedback to developers (BrowserStack, 2025).

In this app, CI/CD pipelines were used to automate testing and deploy the logs to Google Cloud, automate the building and deploying of the app to Google Cloud, create a downloadable zip file of the entire application, including workflows, and track releases as part of version control.

Key Features of the pipeline

1. Automated testing, building, and deployment of a MERN stack application
2. Test logs uploaded to Google Cloud buckets as shown in the image below

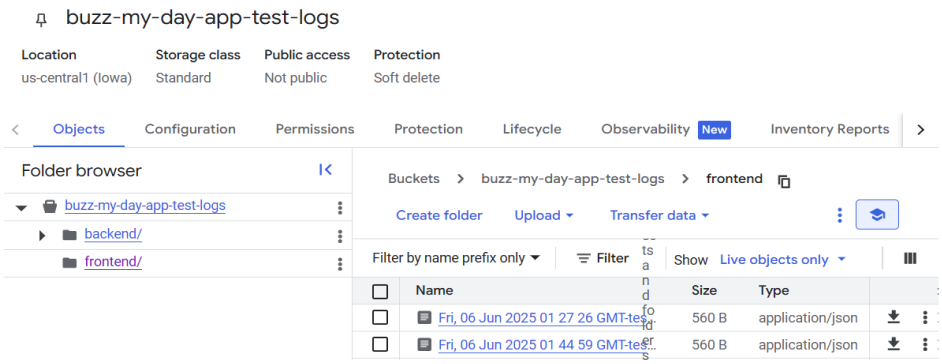


Image 1: Examples of test logs uploaded to Google Cloud Platform

3. A workflow that allows for GitHub releases to occur as shown in the image below:

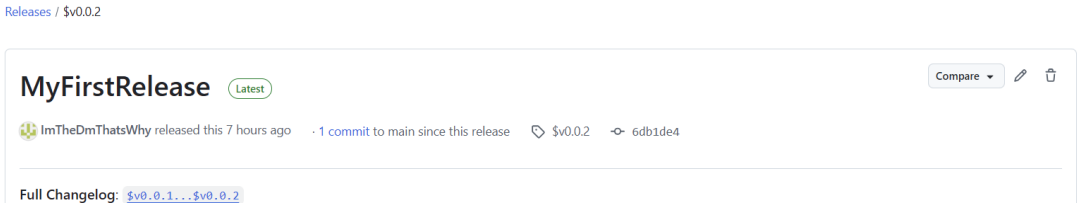
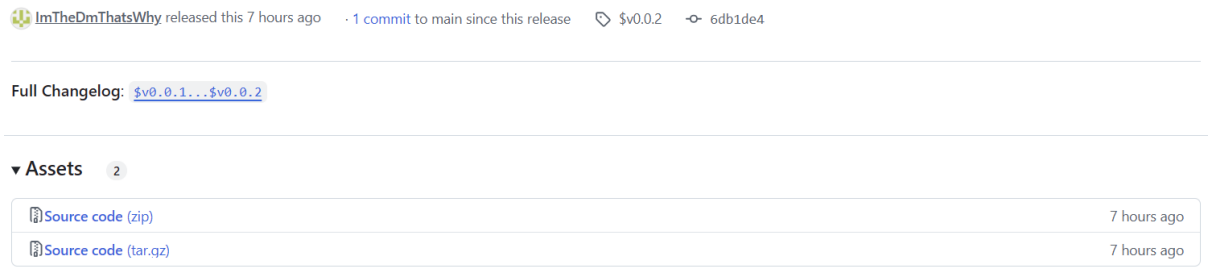


Image 2: The image above shows a picture of releases and versioning in GitHub

4. A workflow that allows the whole application to be downloadable as a zip file as shown in the image below:



Technology used:

To facilitate the automation of the Buzz My Day app technology, Docker, GitHub Actions, and Google Cloud were used. GitHub Actions is a continuous integration and delivery (CI/CD) platform that allows users to automate building, testing, and development pipelines (GitHub, 2024). GitHub Actions allows workflows to run while other events are happening, add labels, and automatically run workflows. Testing itself is an essential part of finding

bugs, and detecting any environmental defects (TestLambda, 2023). Automated testing using CI/CD has the following benefits according to TestLambda, 2023:

- Automation of repetitive tasks, which reduces manual labour
- Quick error identification in builds
- Greater amount of test coverage with increased accuracy
- Quick and timely feedback
- Generates multiple results for tests, ensuring consistency
- Enhances continuous delivery
- Ensures Quality Assurance remains continuous, reliable and agile
- Reduces the risk of decreased readability and problematic code

For these reasons automated testing has been implemented.

How does GitHub Actions work?

To use GitHub actions, a user configures a workflow to trigger when an event occurs in the repository, such as a commit; this event is specified in the workflow (GitHub, 2024). A workflow is an automated process defined by a YAML file checked into the repository (GitHub, 2024). In the workflows, jobs are created and can run sequentially or in parallel with each other, and run in a container or virtual instance (GitHub, 2024). In the case of the workflows used for this app, the jobs run in containers. In each job, there are steps that can run a script or an action that are reusable extensions of the workflow (GitHub, 2024). An action in a workflow is a custom application that can be found in GitHub Actions on the marketplace and used to pull code from a repository and reduce workflow repetition (GitHub, 2024). A runner is a server that the workflows run on when triggered and can only run a single one at a time (GitHub, 2024). In this assignment, the runner is Ubuntu latest.

Why GitHub Actions?

Many tools can be used for CI/CD automation aside from GitHub actions, and one of those providers is Jenkins. Jenkins is considered the grandfather of CI/CD and has been a cornerstone of automation for years (Kulkarni, 2023). Jenkins is advantageous in that it offers significant flexibility, supports multi-staged and intricate pipeline development, and has a large community and documentation (Kulkarni, 2023). However, Jenkins has some drawbacks, including the need for manual setup and configuration, as well as manual security setup and maintenance (admingeek, 2024). In comparison, GitHub Actions allows for seamless integration with GitHub, has a simplified set-up with YAML files, built-in security features, and offers prebuilt actions (admingeek, 2024). Furthermore, GitHub is ideal for developing workflows for moderate to smaller projects (Kulkarni, 2023). Therefore, GitHub was chosen instead of Jenkins due to its simplicity and seamless integration with GitHub.

Why Google Cloud?

There are three major cloud computing platforms: Google Cloud Platform (GCP), Amazon Web Services (AWS), and Azure (Jones, 2017). Since I have only used AWS and GCP as a coder, this section will focus on why GCP was preferred over AWS. Both platforms offer load balancing, have a similar range of network locations, and feature cutting-edge security (Jones, 2017). However, unlike AWS, GCP offers better free tier options, is easier to use and more developer-friendly (GeeksforGeeks, 2023); therefore, it is the chosen cloud service for this project.

Why Docker?

Docker is responsible for the containerisation of software, allowing for the easy building of an application that can be efficiently shipped to run on other machines. Docker also integrates seamlessly with CI/CD pipelines, enabling automated testing, building, and deployment (GeeksforGeeks, 2025). Docker also allows for efficient deployment and has security

benefits, as one container cannot access the data in another container without authorisation (GeeksforGeeks 2, 2025). Furthermore, using Docker allows deployment to be more cost-efficient as it scales down the need for infrastructure in development and requires less memory (GeeksforGeeks, 2, 2025). For these reasons, Docker will be used in the workflows to package the application before being deployed to the Google Cloud Platform.

Docker File Configuration: Some configurations were made to the Docker file as suggested by Dahal (2024), to allow seamless containerisation and deployment of a full stack MERN app. The additions were the following:

1. ARG VITE_API_ENDPOINT=/api: An argument with a default value so that an environmental variable can be provided as both the frontend and backend are running on the same host.
2. ENV VITE_API_ENDPOINT=${VITE_API_ENDPOINT} — This assumes that the project was created using Vite.
3. RUN VITE_API_ENDPOINT=${VITE_API_ENDPOINT} npm run build — Creates the production build for the front end

Other instructions followed not related to Docker, included the router suggestions, which allow the entire application to be built as one by redirecting URLs to the frontend. An image of the Docker file can be found below:

CI/CD pipelines overview diagram:

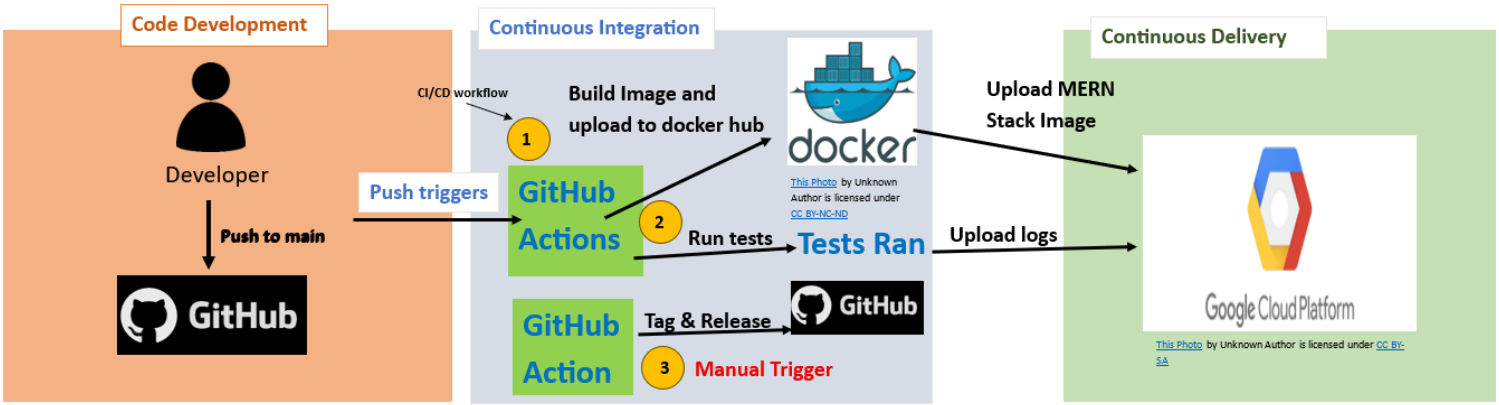


Diagram 1: The diagram above provides an overview of how the pipelines function. When a developer pushes code to GitHub, it triggers GitHub actions across four CI/CD pipelines. In the first pipeline, the Docker image is built and pushed to Docker Hub, then uploaded to Google Cloud for deployment. In the second pipeline, the Backend and Frontend tests are conducted. Once these tests have been completed, the results are then pushed to Google Cloud. In the third pipeline, a tag and release workflow was built, which is activated in GitHub, enabling version releases and the creation of a downloadable zip file.

Workflow 1:

The first workflow pipeline is to automate testing and push the logs to the Google Cloud Platform. The workflow has the following steps:

- **Push branch main:** This step tells the workflow to trigger when a developer pushes to main
- **GitHub actions:**
    1. **actions/checkout@v4.2.2:** This action checks out the repository under the GitHub workspace so the workflow can access it. According to the GitHub website, it is under the MIT license, which is present in the ReadMe. The MIT license is an open license; therefore, there are no ethical concerns in using this code. This code is used in all other workflows as well to copy the repository into the runner.
    2. actions/setup-node@v4.1.0 : This action has a built-in function for caching and restoring dependencies, and it also downloads the distribution of the requested Node.js version and is used in this workflow and all the other workflows to install Node.js into the runner so npm commands can be run. This code is also under the MIT license, so there are no ethical concerns.

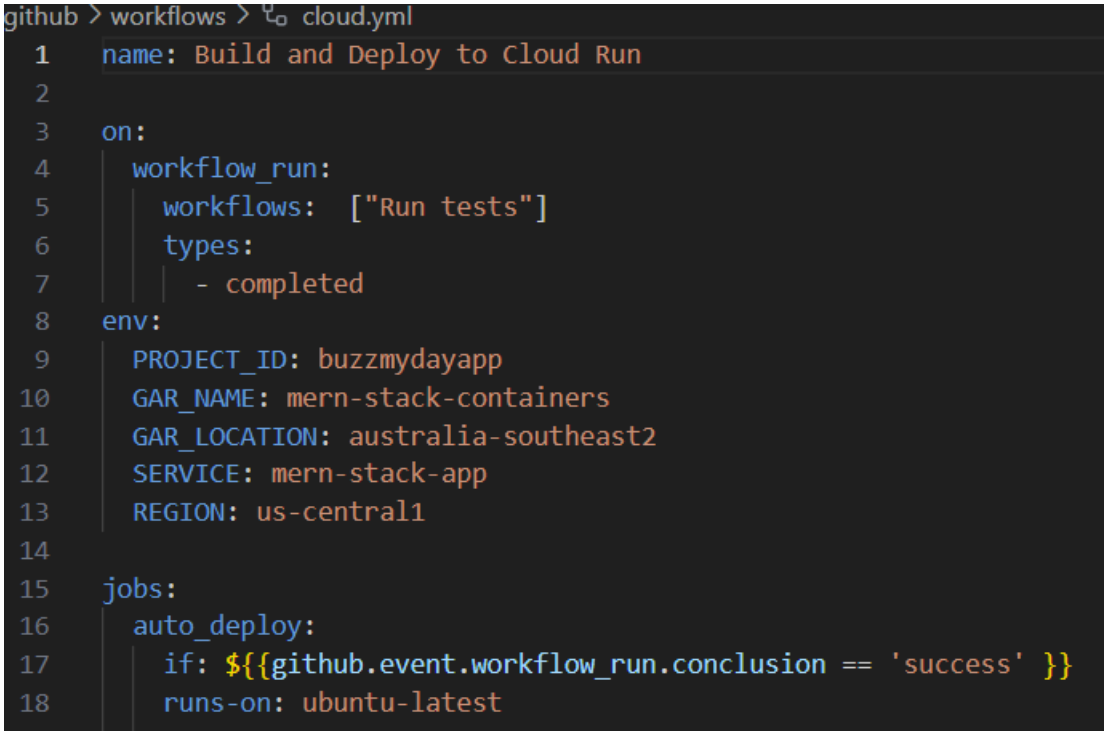
3. Install frontend and backend dependencies
4. Run tests using the command npm run tests
5. **Actions:**
6. **Apache license source:** Wikipedia Contributors (2020). *Apache License*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Apache_License.
7. 'google-github-actions/auth@v2': This action authenticates Google Cloud and is used in the workflow to log in to Google Cloud Platform and uses an authentication using the secret GCP credentials.
8. 'google-github-actions/upload-cloud-storage@v2' : This code provides a path to files that can be uploaded in the code; the destination specifies where the files will end up. In this case, the name of the destination is the name of the buckets in Google Cloud. This code was used to upload the test logs to those buckets.
9. Both actions are under the Apache License 2, which is a permissive free software license that allows users to freely distribute or use software under the license; therefore, there are no ethical concerns with using this license.

Why automated testing services? CI/CD are essential for companies that want to deliver high-quality software and stay competitive (LambdaTest, 2023). CI/CD helps streamline software processes like testing, allowing for effective team collaboration and the quick release of code changes (LambdaTest, 2023).

Workflow 2: The second workflow is a workflow that builds and deploys the MERN full-stack application container to Google Cloud Platform. This workflow was set up following the instructions from Dahal, S. (2024).

Steps:

1. This workflow triggers automatically after the tests are run using the code below



The workflow specifies which workflow needs to be completed before auto-completing, and in the jobs section, the condition is set to auto-deploy if the event workflow run for tests is a success.

2. The next steps are the same as the first workflow and use the same two actions as steps 1 in workflow 1.
3. The next step is to sign in to Google Cloud Platform, which works the same as step 8 in workflow 1.
4. The Docker auth code identifies Google Cloud Platform as a Docker credential helper that helps store Docker credentials
5. The next step builds and pushes the Docker container.
    a. The secret VITE_API_ENDPOINT is necessary to hide the API's connection endpoint; the Vite Api endpoint itself is the endpoint that connects Vite to the API server.

    b. All env. Variables refer to environmental variables that are created and identified in GitHub and are required for the code to work

6. The next step is to deploy to Cloud Run, which uses the action google-github-actions/deploy-cloudrun@v2. In the code, a JWT secret and a Database secret are used. The JWT secret protects the logins for the website, while the mongo secret protects the URL for the MongoDB. The other environmental variables (Project_ID, GAR_NAME, SERVICE are required to match the variables set in Google Cloud Platform to work. This action is under the Apache-2.0 licence, which has already been explained in this document; therefore, there is no ethical concerns in using this licence.
7. The last step is show output, which prints the URL that the website deploys to.

Why Automated Deployment Services? Automated deployment involves using tools and scripts to automate deployment (Farias, 2023). According to Farias (2023), automated deployment has several benefits, including:

- Reducing the occurrence of human error by implementing automated scripts, ensuring the same process is followed each time, enabling a reliable and consistent environment for testing and staging
- Ensures the rapid release of changes, with the most recent version being available for testing
- Allows for seamless rollbacks when mistakes occur
- Frees up developers' time to focus on testing, coding, and refining an application rather than on manual deployment.

For these reasons automated deployment was used.

Workflow 3: The third workflow creates a zip file of the entire application with downloadable items, including the workflows, and adds versioning.

1. The first instruction says on: workflow dispatch, meaning that the workflow must be manually triggered in GitHub actions.
2. The next step uses the actions/checkout@v4.2.2 to checkout code in the repository
3. The bump version and tag commit step creates a version tag with each commit and uses the action mathieudutour/github-tag-action@v6.2 to do so. This action requires the secret GH_TOKEN (Github token) to allow for permission to tag the repository. This action is under the MIT license, which has already been discussed therefore, there are no ethical concerns with using this action software.
4. The final step creates the GitHub release using the GitHub action elgohr/Github-Release-Action@release-20241111151247. This action generates a plain release on GitHub. The with: tag is employed to set the tag of the release. While the GH_TOKEN is used for publishing, it requires the correct permissions to be configured. This code is under the MIT license; therefore, there are no ethical concerns regarding its use. An image of the code can be found below:

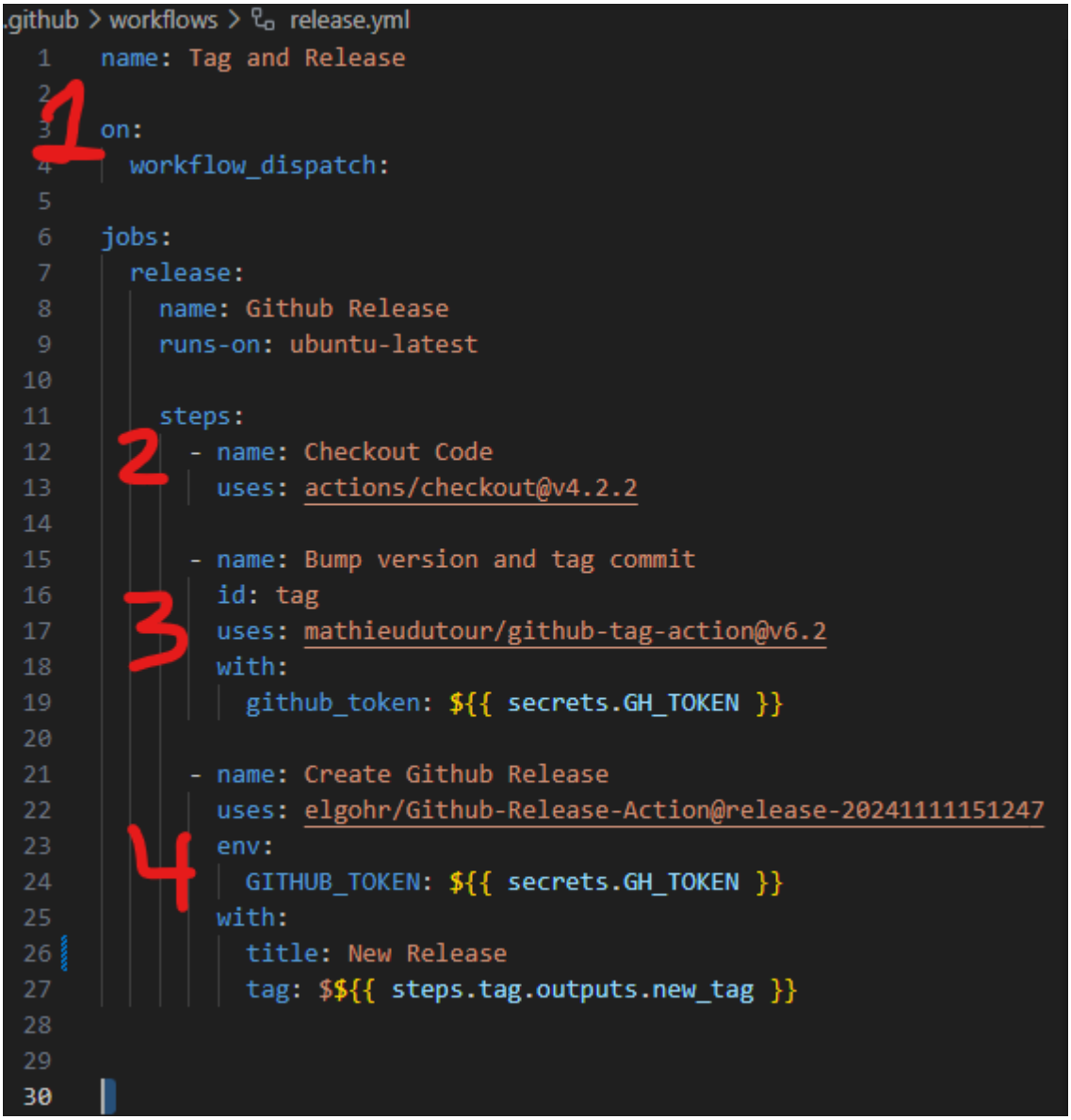


Image 3: The image above shows the code for the Tag and Release workflow, with the numbers corresponding to the steps outlined in the explanation above.

Why is a tag-and-release service important? The process of tagging and releasing in GitHub is a vital practice as it streamlines the process of deploying software while maintaining consistency and reliability Dumez, 2025. Automatic tagging also allows for traceability and helps to maintain a record of changes, making it easier to track history and facilitate rollbacks (Dumez, 2025). Tag and release also creates a zip file

of the application, which can be beneficial for downloading necessary items without needing to fork a repository. For these reasons, Tag and Release was used.

Summary: The image below provides a summary on how all the workflows work together:

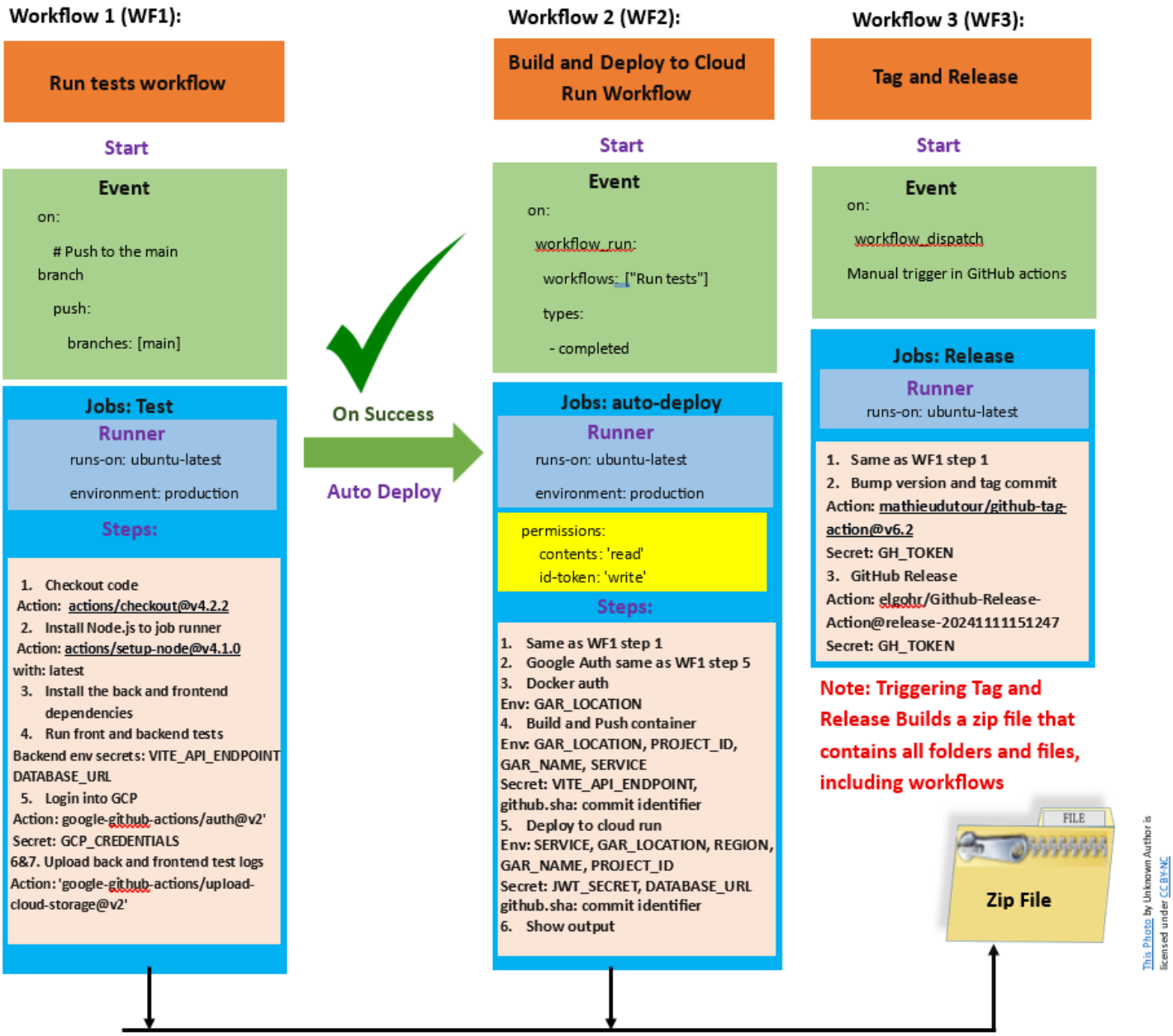


Image 4: This image shows an overview of each workflow, including the event they trigger on, what they run on, jobs, steps and key actions and secrets. Workflow 1 is the test workflow, and workflow 2, which deploys the app, is dependent on workflow 1 completing, as the tests must pass before deploying the application. Workflow 3 is separate and is in charge of releasing tag versions of the application upon manual triggering in GitHub. However, when the tag and release is triggered, it creates a downloadable zip file that

contains all folders and files, including the workflows, which can be downloaded from the zip, ensuring that the latest workflows are accessible.

References:

Vergardia, P. (2025). *DevOps and CI/CD on Google Cloud explained*. [online] Available at: https://cloud.google.com/blog/topics/developers-practitioners/devops-and-cicd-google-cloud-explained.

BrowserStack. (n.d.). *Cloud DevOps: How DevOps and Cloud work together*. [online] Available at: https://www.browserstack.com/guide/cloud-devops.

GitHub (2024). *Understanding GitHub Actions - GitHub Docs*. [online] GitHub Docs. Available at: https://docs.github.com/en/actions/about-github-actions/understanding-github-actions.

Kulkarni, P. (2023). *Jenkins vs. GitHub Actions: A Comprehensive Comparison for Effective CI/CD Automation*. [online] DEV Community. Available at: https://dev.to/pavankulkarni/jenkins-vs-github-actions-a-comprehensive-comparison-for-effective-cicd-automation-22n0.

admingeek (2024). *Jenkins Vs GitHub Actions: Which is Better? - Infotechys.com*. [online] Infotechys.com. Available at: https://infotechys.com/jenkins-vs-github-actions/ [Accessed 6 Jun. 2025].

Jones, E. (2017). Google Cloud vs AWS in 2019 (Comparing the Giants). [online] Kinsta Managed WordPress Hosting. Available at: https://kinsta.com/blog/google-cloud-vs-aws/.

GeeksforGeeks. (2023). *Google Cloud vs AWS: Which One Should You Choose?* [online] Available at: https://www.geeksforgeeks.org/gcp-vs-aws/.

GeeksforGeeks. (2025). *Kubernetes vs Docker*. [online] Available at: https://www.geeksforgeeks.org/kubernetes-vs-docker/.

GeeksforGeeks. (2025) 2. *Why Should You Use Docker - 7 Major Reasons!* [online] Available at: https://www.geeksforgeeks.org/why-should-you-use-docker-7-major-reasons/.

www.lambdatest.com. (2023). *CI/CD Testing: What, Why, and How*. [online] Available at: https://www.lambdatest.com/learning-hub/cicd-testing.

Dahal, S. (2024). *Deploy a Containerized MERN web app to Google Cloud Run using GitHub Actions*. [online] Medium. Available at: https://medium.com/@simonsd054/deploy-a-containerized-mern-web-app-to-google-cloud-run-using-github-actions-f1d0ae828f8e [Accessed 7 Jun. 2025].

saas.group, I. and Farias, F. (2023). *Finding the Right Balance: Automated vs. Manual Deployments for Different Environments*. [online] Deploybot.com. Available at: https://deploybot.com/blog/finding-the-right-balance-automated-vs-manual-deployments-for-different-environments [Accessed 8 Jun. 2025].

Keith, the Coder (2020). *Zip Code Base with Github Actions for Releases*. [online] YouTube. Available at: https://www.youtube.com/watch?v=yAkMgcfdok0 [Accessed 8 Jun. 2025]

Dumez, K. (2024). *How to automate tagging and release workflows in GitHub*. [online] Available at: https://graphite.dev/guides/how-to-automate-tagging-and-release-workflows-in-github [Accessed 8 Jun. 2025].