GIT and CI/CD

Renu
(Renugopal Sivaprakasam)

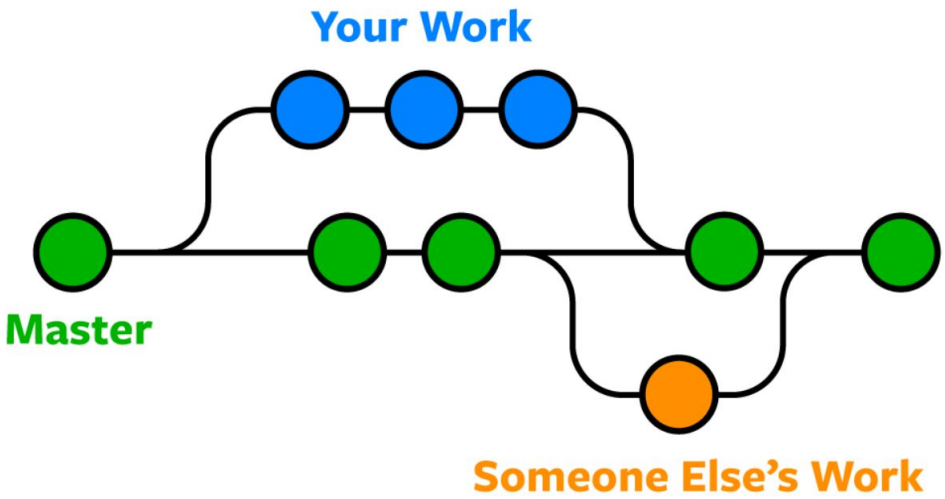
2/25/2025

Agenda

- GIT Workflow
- GIT Activity
- CI / CD Workflow
- CI/CD Activity

GIT

- Popular Version Control System
- Github, Bitbucket, Gitlab, etc.. are Cloud based Git hosting platforms.
- Helps in collaboration



GIT POP QUIZ 1

1. **What happens when you create a new branch in Git?**

   A. The entire repository is duplicated into a separate folder.

   B. A pointer is created that references a specific commit in the history.

   C. The main branch is replaced by the new branch.

   D. Changes from the main branch are automatically merged into the new branch.

Common GIT Commands

git clone or init

Git clone
<repository link>

Staging - git add .

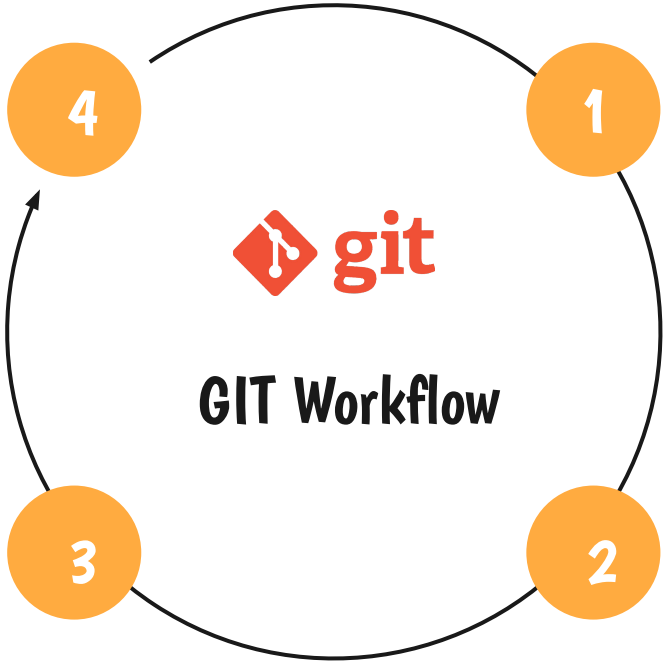
git add <file name>

git commit

git commit -m
“Commit Message”

git push

git push origin
main



GIT Exercise

1. Navigate to this URL -
https://github.com/regostar/semaphore-demo-javascript

2. Fork The Repository

3. Clone This Repository on local
Open VS Code
Open terminal
git clone <url>

4. Create a new Branch and checkout to it
git checkout -b <your_branch_name>

5. Add a File test1.txt

GIT Exercise

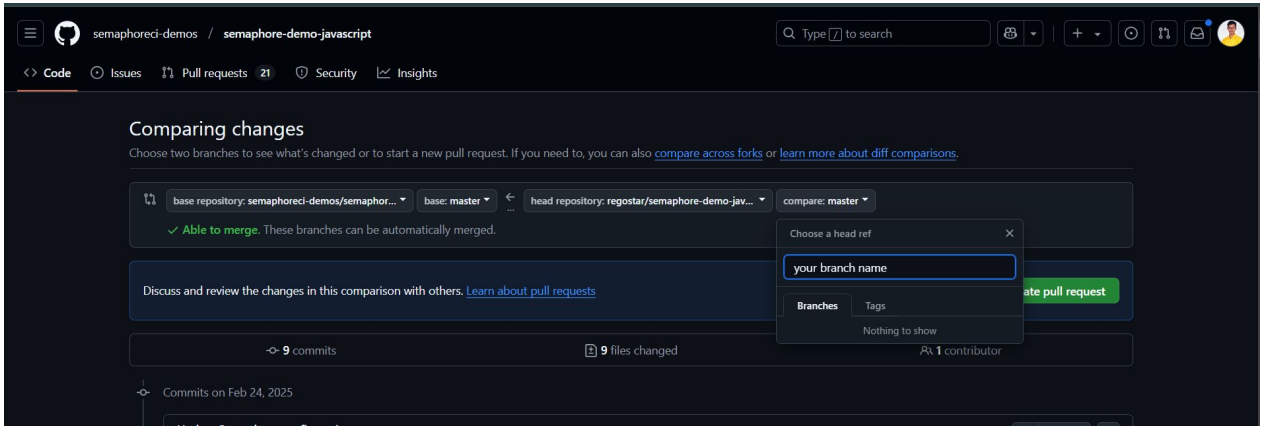
6. Add to GIT Staging area
git add test1.txt

7. GIT Commit
git commit -m “added new test file - this is your message”

8. GIT PUSH
git push origin <your new branch name>

9. Open A Pull Request on Github from your branch to master

10. Merge it



Other GIT Commands

Identity :
git config --global user.name "Mike Riethmuller"
git config --global user.email mike@madebymike.com.au

Creating empty GIT Repository
git init
git remote add origin
https://github.com/MadeByMike/madebymike.madebymike.git

Status
git status

Create a Branch
git branch name-of-branch

Checkout a branch
git checkout name-of-branch

Other GIT Commands

Staging files
git add .
git add ./my-folder/

Unstaging files
git reset ./path-to/file.js

Commit
git commit -m "Add a useful commit message here"

Push to a remote branch
git push origin develop

Pull a remote branch or merge remote with local
git pull origin develop

Other GIT Commands

Stash
git stash
git stash list
git stash pop
git stash apply stash@{1}

For an explanation refer -
https://www.madebymike.com.au/writing/how-to-git/

GIT POP QUIZ 2

A developer needs to securely integrate OpenAI API keys into their project while collaborating via Git. Which approach follows security best practices for CI/CD pipelines?

Options:-

a) Hardcode the key directly in main.py and commit it
b) Store the key in config.json but exclude it in .gitignore
c) Use environment variables with a .env file listed in .gitignore
d) Encrypt the key using git-crypt and commit the encrypted file

Answer

The most secure and scalable solution is c) Use environment variables with a .env file listed in .gitignore, combined with:

Never committing secrets: .gitignore prevents accidental exposure

CI/CD integration: Configure secrets in pipeline settings (GitHub Actions Variables/AWS Secrets Manager)

Local development: Load via dotenv or similar packages

Why others are insecure:
a) Exposes keys permanently in Git history
b) Risk of forgetting to exclude config.json
d) Requires team-wide setup for decryption

GIT POP QUIZ 3

What is the primary advantage of using Git rebase over Git merge?

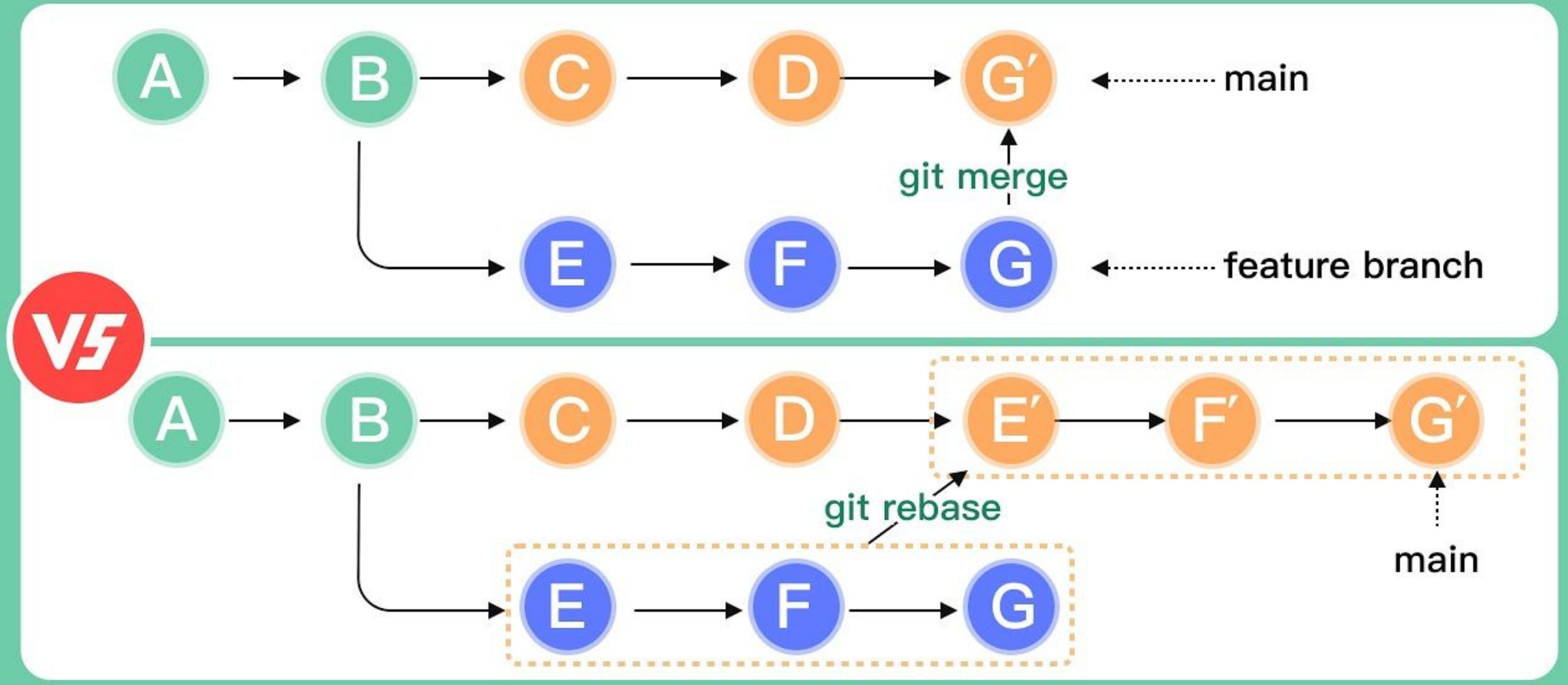
A. It creates a more linear, cleaner commit history.

B. It eliminates the need for resolving conflicts entirely.

C. It always preserves the exact branch history with merge commits.

D. It automatically synchronizes your local and remote branches.

Git MERGE vs REBASE



GIT REBASE VS MERGE

Identifying Commits: Git rebase finds the commits in your current branch that are not in the target branch.

Replaying Commits: These commits are then reapplied one by one on top of the target branch's latest commit.

New History: The commits get new commit hashes since their parent commit has changed, resulting in a linear, cleaner history.

Continuous Integration

Using a dustbin, few balls, and a sticky note:

- Balls = Code Changes (or Commits)
- Dustbin = Central Repository (or Codebase)
- Sticky Note = Unit Test/Build Status

The process:
Developer's Action: Every time you throw a ball, think of it as a developer committing code to the central repository.

CI in Action: After each ball lands in the dustbin, check the sticky note (test results).

The **dustbin remains clean** (like your main codebase) because each time someone adds something, it's verified by the sticky note (tests).
Continuous Integration ensures that **multiple people can add their balls** (code) into the dustbin smoothly without causing chaos or mess.

Continuous Integration

CI (Continuous Integration): Automates the process of integrating code changes regularly from multiple contributors into a shared repository.

Steps:

1. Developer **commits** code.
2. Code is **pushed** to version control (e.g., Git).
3. Automated **build** process starts (e.g., Jenkins).
4. Unit **tests** and other automated tests run.
5. **Feedback** is given to the developer immediately.

Key Tools: Jenkins, Travis CI, CircleCI, GitLab CI.

Continuous Delivery

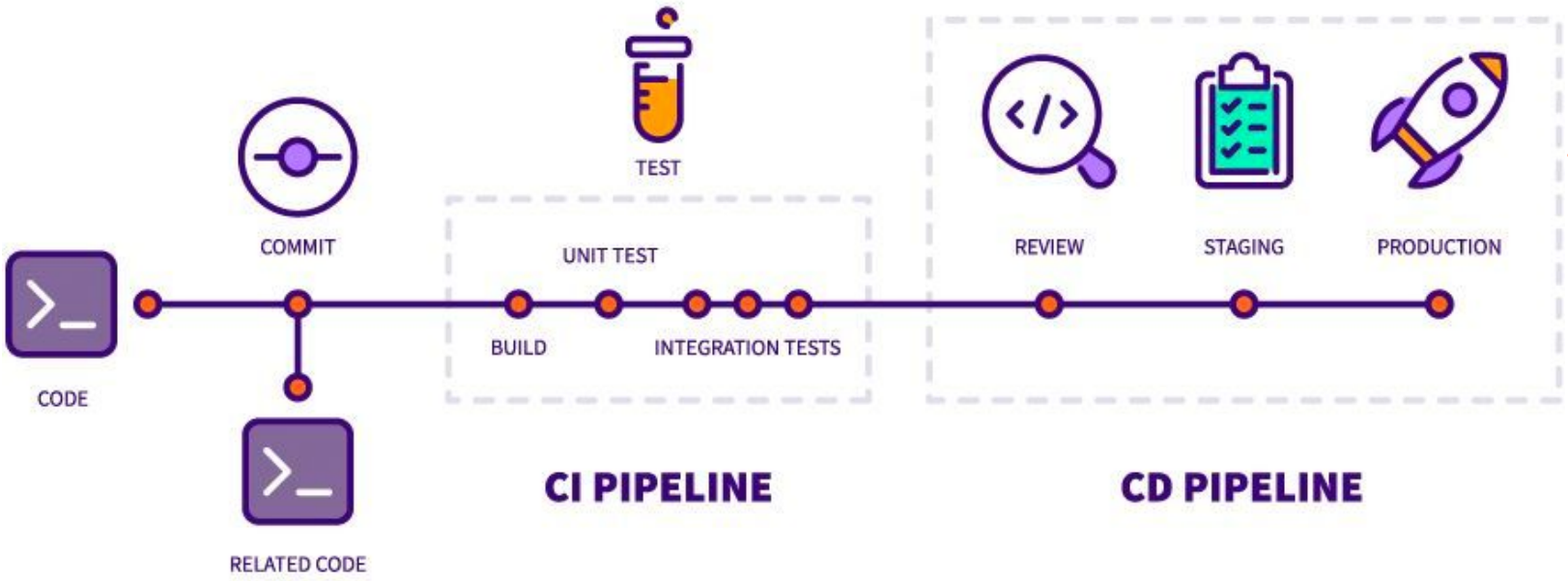
CD (Continuous Delivery/Deployment): Automates delivering applications to production after successful integration, with either manual or automatic deployment.

Steps:

- Post-CI, artifacts are generated.
- **Deployment** to staging for testing (Continuous Delivery).
- Automated deployment to **production** (Continuous Deployment).
- **Monitor** for performance and feedback.

Key Tools: Docker, GCP Cloud Build, Kubernetes, AWS CodeDeploy, ArgoCD.

CI/CD Pipeline

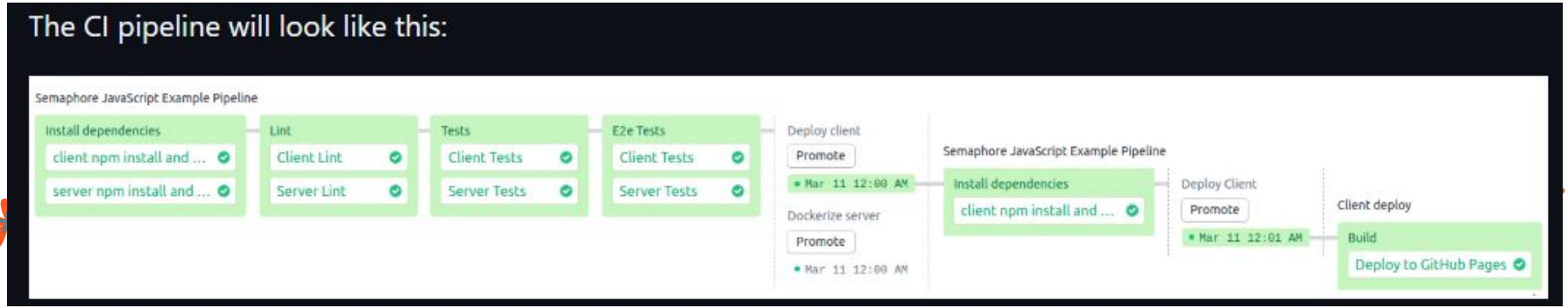


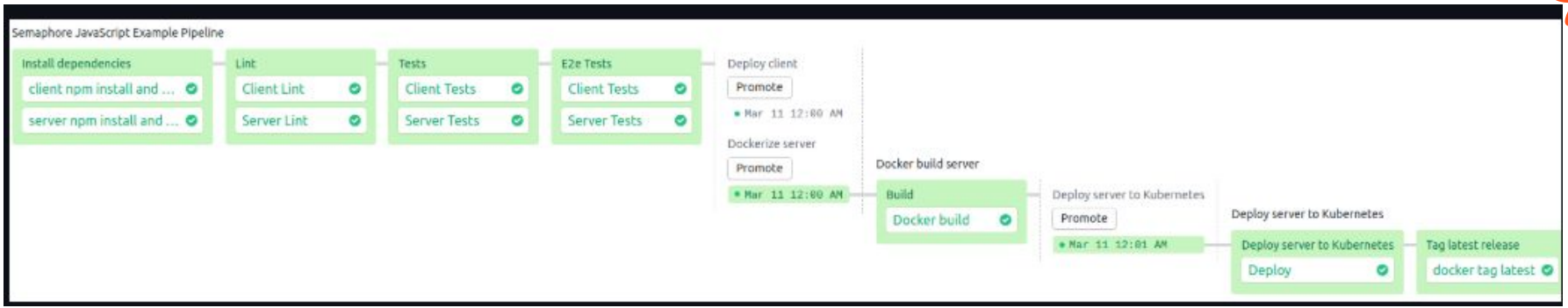
CI/CD Activity

1. Visit Semaphore CI and Create an Account with your Github
2. Create a New Project
3. Choose Repository and give access
   https://github.com/regostar/semaphore-demo-javascript
4. Proceed with the creation
5. Run the Workflow
6. View and Make changes
7. Go to Repository -> your branch -> make file changes
8. Open a Pull Request to Master - See the Build status there

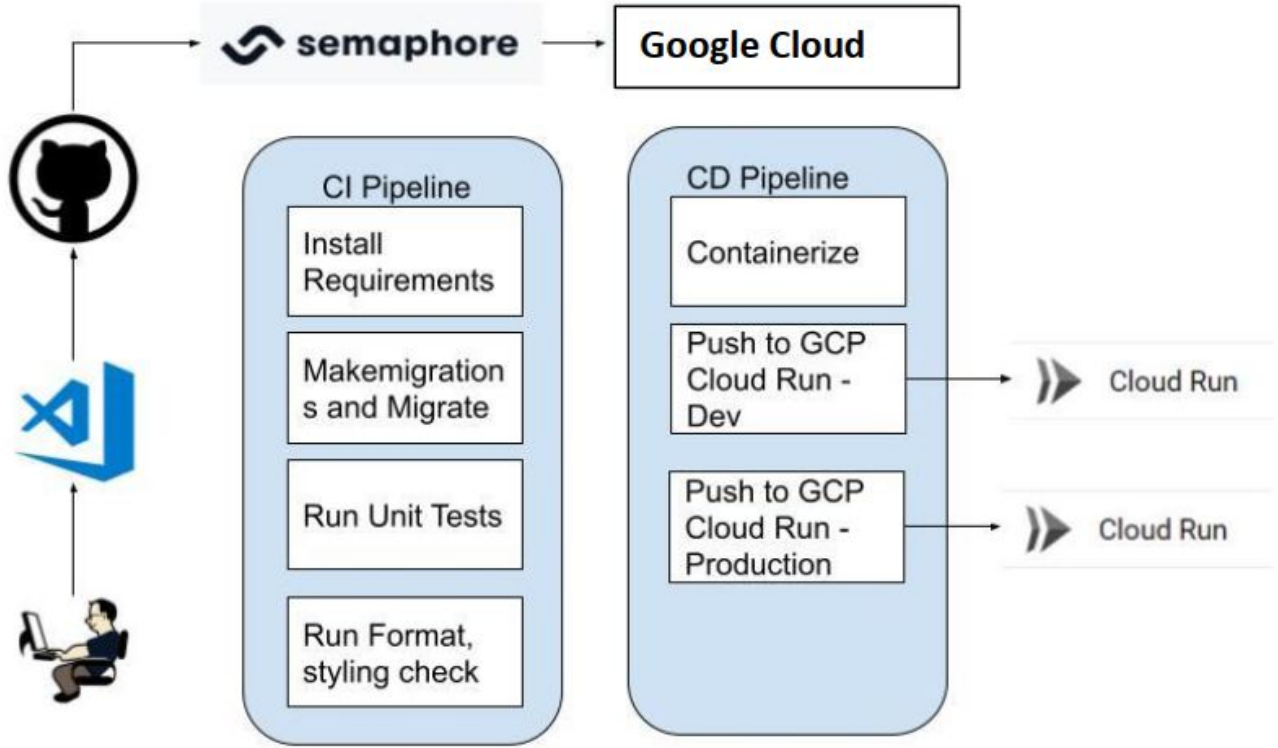
CI/CD Activity

The CI pipeline will look like this:

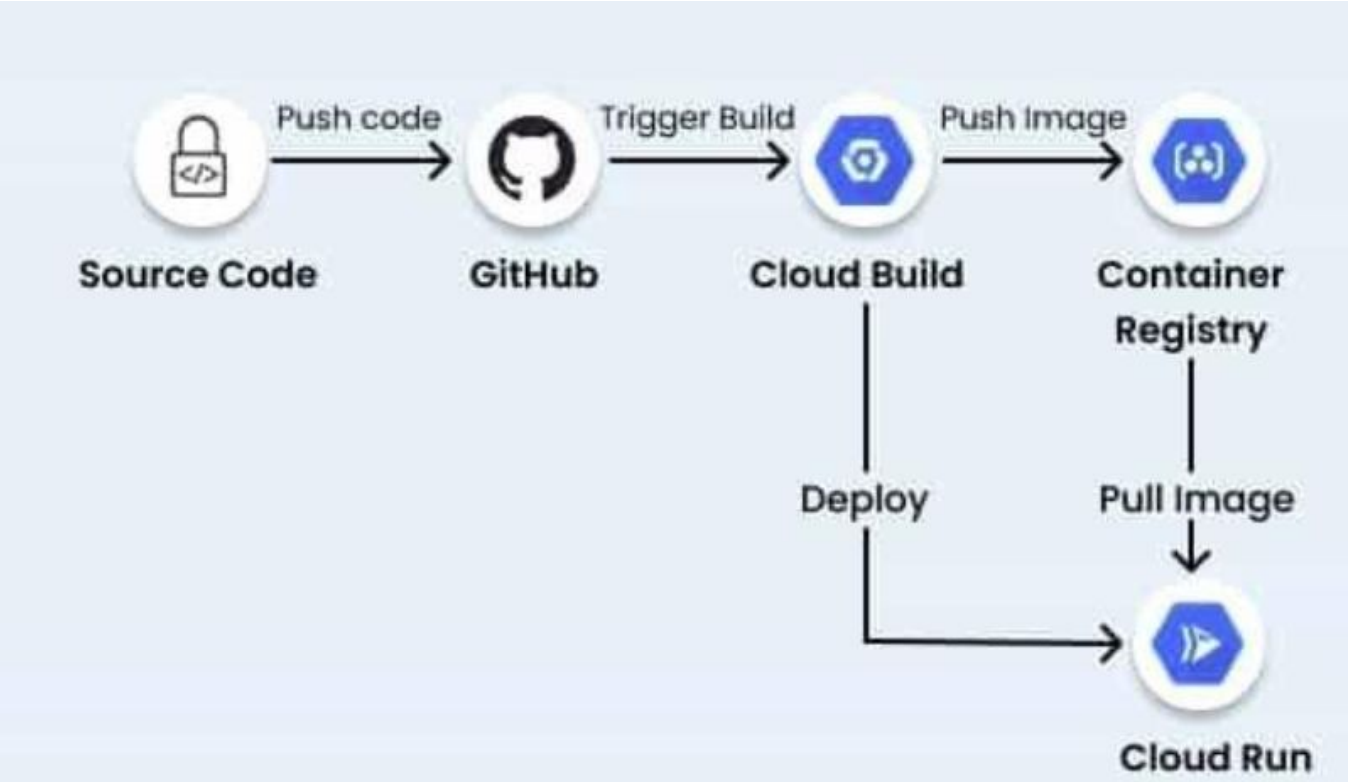




How I implemented CI/CD



A deeper look into my CD pipeline



POP QUIZ

A developer creates a feature branch, adds a new API endpoint, and opens a pull request. The CI/CD pipeline runs these steps:

1. Lint code
2. Run unit tests
3. Build Docker image
4. Deploy to staging

The PR passes all checks but can't be merged due to a **merge conflict**. What should the developer do first to resolve this?

Options :

a) Re-run the CI/CD pipeline

b) git fetch origin main && git rebase main

c) Delete the feature branch and start over

d) Force push to main branch

Answer

The correct answer is b) git fetch origin main && git rebase main.
This fetches the latest main branch changes and replays the feature branch commits on top of them, resolving the conflict locally before pushing the updated history. The CI/CD pipeline would automatically re-run after the rebase and push.

Exercise 3

Fork the Repository (and give a star :P)
https://github.com/regostar/flask_react_app

Walkthrough of ci_cd.yml file

Resources

Github actions and github pages-
https://medium.com/@pathirage/step-in-to-ci-cd-a-hands-on-guide-to-building-ci-cd-pipeline-with-github-actions-7490d6f7d8ff
Deploy on Netlify
https://docs.netlify.com/api/get-started/
CI/CD on Digital Ocean
https://medium.com/@its-andrerebonato/setting-up-a-basic-ci-cd-pipeline-for-deploying-a-node-js-application-on-digitalocean-8797aa2c8049
Semaphore CI/CD with nodejs react app
https://github.com/regostar/semaphore-demo-javascript/
Semaphore CI/CD with nodejs and Kube Clusters for deployment
https://github.com/semaphoreci-demos/semaphore-demo-nodejs-k8s

Closing Thoughts

“The biggest risk is not taking any risk... In a world that’s changing really quickly, the only strategy that is guaranteed to fail, is not taking risks.”

—Mark Zuckerberg, Co-founder of Facebook

“If you don't innovate fast, disrupt your industry, disrupt yourself, you'll be left behind.”

—John Chambers, CEO of Cisco