Cloud-Based Poly-Repo Is GitHub

S. Mohan
Assistant Professor of Computer Science and Engineering
V. S. B. College of Engineering Technical Campus, Kinathukadavu- Coimbatore -Tamil Nadu-India

***Abstract*—The GitHub is cloud-based platforms that fully support a poly-repo (multi-repository) architecture but it is not only a poly-repo tool. It serves as a hosting provide where you can choose to store your code-based as multiple isolated repository (poly-repo) or combine every thing in to a single, massive repository. the poly-repo setup means that every application service or libraries within an organization is assigned its own independent repository. The cloud-based poly-repo architecture is inherently built up on GitHub core abstraction because GitHub is fundamentally designed to be a repo- scoped platforms, every individual repository act as a isolated sandbox with its own access control .CI/CD automation and release reasoning every project, micro-service or infrastructure components gets own dedicated GitHub repository. the GitHub organization function as the parent container that aggregate all your decentralized poly-repo under one cloud identity. the organization build a master repository that coordinate code across different repositories using Git sub modules or customs script giving visibility without forcing code in to a single mono-repo a cloud-based poly-repo using GitHub means you manage a software project by splitting its code across multiple separate repositories hosted on GitHub cloud platforms.**

***Index Terms*—CI, CD, API,**

I. INTRODUCTION

A poly-repo (multi repository) approach means on organized split its code based in to a separate. independent repository based a project team, or micro-service each project has its own distinct Git repository team manage their own code setting and access control. repository have unique version visibilities, deployment pipe-line and release scheduled. the group related repositories under one control computer umbrella grant access to specific repository without exposing the entire code-base run separate CI/CD work flour tailored to each project needs. like separate GitHub repository together when dependencies exist. the development more faster without blocking other leafeater clone time and single code review. strong separation of concern reduced accidental code entanglements security flaws or broken build reaming isolated to one repository updating share the library across the multiple repository requires coordination. they might accidentally write identical utilizes code is isolation, different linking, testing, and formatting standard. the cross-repo changes feature update spanning multiple services required multiple pull request. CI/CD (customer integration/ customer deployment/deliveries). because most important to GitHub poly repository when the project is split across many independent repositories .CI/CD is vital. Because it automates building testing, and shipping code for each separate small service while GitHub started strictly as a Git based code responding and version control platforms it has ended in to a full DevSecOPs eco=system through GitHub action developer can build, test and deploy software directly from the code repository without needing external third-party service.

CI- (customer integration):
Automatically conflict code manager depended and run test suited every time a developer submits changes centering bugs before the search production.

CD- (customer deployment/ deliveries):
Automatically deploy successfully tested code directly to cloud provider (like AWS, Azure, or Google cloud) or internal source.
CI/CD is arguably more important in a poly-repo architecture then in a mono=repo because it serve as the glue that provide independent code responsible from different aspects and breathing overall systems while poly-repo give individual team incredible sperate and overall shipping. they introduced Hiten complexity like dependency hell fragmented testing

and in consistent development process that only robust automation can solve.

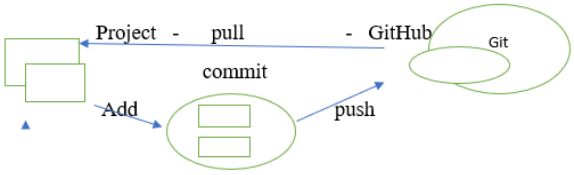
CI/CD admin Role:
For enterprises governance the GitHub provide CI/CD admin organized role. this specialized access level allows the specific team members to manage global pipe line without warding full administrative control over the entire compony repositories the safety configure API token deployments passwords and environment keys. this manage specific self-hub infrastructure and server group allocated for heavy build task enforce using permission and rules a what specific action workflow are allow to execute.

YAML (Yet Another Markup Language):
It is a human readable format used for comfortable files. the GitHub action uses YAML files inside (. GitHub/ workflow) folder to set update automated work flow test and deployment pipe-line.
GitHub means:

The GitHub work means refer to using GitHub a cross-based platforms to store code tracking changes and collaprete on software deployment project its fiction like a cloud based filling cabinet for dosing file (primary code) contained with a tracking history and social networking feature for developers. Digital project folder contains code documentation and historical file logs.

VERSION CONTROL: (Git)
A tracking system that records every change mode to file so you can UNDO mistakes or com [pare version. it have separately safe work space copies where you can write new code without breaking the main live projects.

PULL REQUEST: (PRs)



Formal submission asking project owner to review, discuss and safely mange your new code in to the main project.

REPOSITORIES:
a repository or (repo) is a storage place for a project it contains all the file their history and meta-data repositories can be possible (accessible to everyone) or private (partial access).

BRANCHES:
branches allow develop to work on different version of project simultaneously. the different branches are often called (MAIN) or (MASTER) develop can create new branches for feature or BUGS fixes and lateral marge them in to a main branch.

COMMIT:
a commit represents a change made to the code basedtech commit include a description exploring the changes, making it easily to track progress and revert is necessary.

PULL REQUEST:
Full request one way to purpose change to a repository they allow team members to review, discuss, and merge changes in the main branches.

COLLABRATE TOOLS:
GitHub provide tools like issues for tracking bugs and task, project board for workflow management and GitHub action for automatic task like testing and deployments.

II. FORKING AND CLONING:

forking create a personal copy of same one else s repository, allowing you to experiments or contribute, cloning down loads a repository to your local machine for off-lone work.
The GitHub significances collaboration by enabling multiple developer to work on the same project without over writing each other changes. it takes the entire history of project, making it easy to revert to previous version or understand why change, where made. it also same as a platform for open-source contribution. where developer worldwide can collaborate on project whether your significance of leaving the code, professional moving complex project or an open-source enthusiasts GitHub provider the tool to streamline your work flow and entrance productivity.

GitHub MEANS:
Git means a distribution version control system that help track and merge change in the code over time it allows multiple development to work on a same project, view prior version and restore changes when needed.

VERSION CONTROL:
Git keeps track of way change your make to your project file. you can go back to previous version is any requirements copies.

REPOSITORIES:
a Git repository or REPO is taking a project central HUBs when every thing related to your work is stored and managed
THERE are TWO main types:

LOCAL repository;
This is copy of the repository which is stored in your computer. you can work on your project and make change leave.

REMOTE repository:
This is stored on a server, like GitHub, when you and others can share and work a project.

REASON to CHOOSE Git:
Distributed system:
Every developer has a compute local copy of the project, allowing for off-line work and increased resilience.

Performance:
Git light is fast efficient handling large project with easy.

Branching AND merging:
Gits is weight branching and easy managing facility parallel development and feature isolation.

Collaboration:
Tools like GitHub enter to team collaborate, code reviews and project managements.

Track changes:
Git tracks changes and maintains a history, making it is easy to revert to previous version.

REPOSITORY means:
A repository is a place where thing is stored and can be found.in the context of the software developments, a repository typically refers to a storage location for software package code and other related files. to source as a central location where developed can collaborate merge and track changes to the codebase.

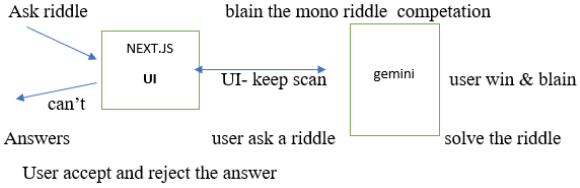
REPOSITORY in Gits:
When naming the repository in Git it is important to follow certain rules to ensure that the repository name is unique descriptive and every to understand here are some key points to consider.

Unique:
Name should be unique with your accounts this means that you cannot have two repositories with the same name under one account for examples: if you have repository named (my-best-repo) on others repository under open can have same name.

Descriptive AND concise:
The repository name should be considered and reflect the project purpose. it should be easy to remember and understanding for instant a good repository name good (air-soapAPI-csharp) where air represents the project name and soap API descript the project purpose and csharp indicate the programming language or frame work used.

Consistent naming conversion:
Following consistent naming conversion that align with your team performance and rules some people prepare using (_) hyphen to separate word while other new code -score (_) any character expect spares and tap can be used for separate.

III. MONO-REPO VS POLY-REPO



Mono-repo:
It is used by major tech compares like Google, face book, micro soft, Twitter, Uber, and Airbnb as well as

in the open-source [project and converted mono-repo tools.

Google:
Maintain of the largest mono-repo in the world containing code for most its project enabling shared library and atomic changes across multiple service.

Face book (meta):
User are requiring mono-repo with thousands of commits per week across hundreds of thousands of files, support multiple application and server.

Micro-soft:
Employees mono-repo for large scale project to manage dependencies and structure builders.

Twitter, Uber, Airbnb:
All user mono-repo to consolidate codebase. improve the collaboration and maintain the consistent versioning across multiple projects.

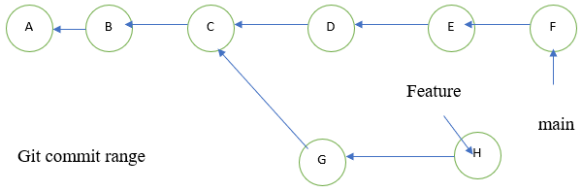
Mono-repo structure organization:

- Apps /: deployable application (web app, mobile app)
- Package/: shared library used across the project
- Tools /: internal script or loosely
- Configs/: centralized configuration files.

This structure allows team to build, test, and deploy project independently while keeping all code in a single repository, facilitating code resume, atomic changes and consistent dependency managements,

Poly-repo MEANE:
poly-repo is a type of repository (typically Git-repo) wand to merge small code using multiple repositories. Poly-repo to known many-repo or multi-repo. a poly-repo or cluster user multiple repositories return the once.

Examples: poly-repo can use 3 repo -one repo for web frond end another repo for mobile app yet another repo for a back-end node.
The smaller team that may across to tool for merging complexity of a large project including micro-service. if your project has large sub – project. it will be better to use multi-repo approach.
But a mono-repo is suitable for micro-service project including dozens of related small folders for examples AWS lambdas. now multi-repo approach with yield dozens of repos spread across -GitHub or Gitlab, will be a might have to handle co-ordination for smaller team.



Feature of MULTI-REPO:
Choose between a multi-repo and a mono-repo strategies depends on your project size, complexity, and team structure with each approach offering distinct advantages and challenges.

Multi-repo: this storage involves maintaining separate repository for each project components or service each repository has its own version history and life cycles allowing team to work independently a different part of the system.
Advantages: independent release more flexible deployment cycle.
Dis-advantages: cross-project co-ordinate overhead.

Mono-repo: in contrast mono-repo contain the code for multiple project or components with in a single repository. this approach processes a centralized version history and facilitate atomic change across he projects.
Advantages: simplified dependency management,
Dis-advantages: complex CI/CD.

Choose mono-repo:
IF ….
You highly independent and required frequent collaboration
Need to prefer light score referring across multiple project
Singly dependencies might and structured CI/CD process.

Choose multi-repo:
IF….
Your project relatively independent and can be developed and release separately.
Large team that benefits for autonomy and focused work on specific components.

Its time – gained across control and support for different part of your codebase.

IV. CONCLUSION

The Gits Ops journey understanding the concept of vital reporting -either mono-repo OR poly-repo which is transferring paradigm in modern software development. the GitHub is the single source fourth declaration implication and application. this approach reliability transferring and automation deployment. this continuous determine an adapting Git Ops how to structure your Git responsibly interact scalability collaboration and overall efficiency. poly-repo allows dresses architecture pattern adapting to complex organization structure. the support multi-cluster, multi-cloud or multi- Tanat environments providing a robust foundation for the advanced use cases. poly-repo certainly better for large, complex and clear separating of responsibility.

REFERENCES

[1] N. Gorelick, M. Hancher, M. Dixon, S. Ilyushchenko, D. Thau, and R. Moore, "Google Earth Engine: Planetary-scale geospatial analysis for everyone," *Remote Sensing of Environment*, vol. 202, pp. 18–27, Dec. 2017, doi: 10.1016/j.rse.2017.06.031.

[2] J. de La Beaujardière, "A geodata fabric for the 21st century," Eos, vol. 100, Nov. 2019, doi: 10.1029/2019EO136386.

[3] D. Medvedev, G. Lemson, and M. Rippin, "SciServer Compute: Bringing analysis close to the data," in Proc. 28th Int. Conf. Scientific and Statistical Database Management (SSDBM), 2016, doi: 10.1145/2949689.2949700.

[4] P. G. Brown, "Overview of SciDB," in Proc. ACM SIGMOD Int. Conf. Management of Data, 2010, doi: 10.1145/1807167.1807271.

[5] P. Baumann, A. Dehmel, P. Furtado, R. Ritsch, and N. Widmann, "The multidimensional database system rasdaman," in Proc. ACM SIGMOD Int. Conf. Management of Data, 1998, doi: 10.1145/276304.276386.

[6] M. D. Wilkinson et al., "The FAIR guiding principles for scientific data management and stewardship," Scientific Data, vol. 3, no. 1, Mar. 2016, doi: 10.1038/sdata.2016.18.

[7] C. Holmes, "Analysis ready data defined," Apr. 2018. [Online]. Available: https://medium.com/planet-stories/analysis-ready-data-defined-5694f6f4815.

[8] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, "Occupy the cloud: Distributed computing for the 99%," in Proc. ACM Symp. Cloud Computing (SoCC), pp. 445–451, 2017, doi: 10.1145/3127479.3128601.

[9] R. P. Signell and D. Pothina, "Analysis and visualization of coastal ocean model data in the cloud," Journal of Marine Science and Engineering, vol. 7, no. 4, pp. 110–121, Apr. 2019.

[10] P. Cornillon, J. Gallagher, and T. Sgouros, "OPeNDAP: Accessing data in a distributed, heterogeneous environment," Data Science Journal, vol. 2, pp. 164–174, 2003, doi: 10.2481/dsj.2.164.