NAME: FRIDASAM KARIMI MURIITHI

TOPIC: GIT AND GITLAB

**1. What is version control, and why is GitHub popular?**

Version control tracks changes in code, allowing multiple people to collaborate without overwriting each other’s work. GitHub is popular because it provides cloud storage for repositories, collaboration tools, issue tracking, and CI/CD integration. It makes teamwork smoother by allowing branching, pull requests, and version history.

**2. How does version control maintain project integrity?**

It prevents data loss, keeps track of who changed what, allows reverting to previous versions, and supports collaboration without conflicts. This ensures that code remains organized and reliable.

**3. How do you set up a new repository on GitHub?**

* Sign in to GitHub and click the "+" icon → "New repository."
* Choose a name and description.
* Decide if it should be public or private.
* Initialize with a README (optional but recommended).
* Add a .gitignore file if needed.
* Choose a license (optional but useful for open-source projects).
* Click “Create repository.”

**4. Why is a README file important?**

A README explains the project’s purpose, setup instructions, usage, and contributions. It’s often the first thing visitors see, making collaboration easier by giving clear guidance.

**5. Public vs. Private repositories: Pros & Cons**

| **Feature** | **Public Repo** | **Private Repo** |
| --- | --- | --- |
| Visibility | Open to everyone | Restricted access |
| Collaboration | Great for open-source | Good for private teams |
| Security | Code is exposed | Code is protected |
| Use Case | Open-source, portfolios | Company projects, sensitive data |

**6. Making your first commit**

A commit is a snapshot of your project at a point in time. Steps:

1. Initialize Git (git init)
2. Add files (git add .)
3. Commit changes (git commit -m "Initial commit")
4. Link to GitHub (git remote add origin <repo-url>)
5. Push the commit (git push -u origin main)

Commits track changes and allow versioning.

**7. What is branching, and why is it useful?**

Branching lets developers work on new features or fixes without affecting the main code.

* Create a branch: git branch feature-branch
* Switch to it: git checkout feature-branch
* Merge changes back: git merge feature-branch

It’s crucial for collaboration, testing, and avoiding conflicts.

**8. What are pull requests, and how do they help?**

A pull request (PR) is a proposal to merge code into the main branch. It allows for code review, discussions, and catching bugs before merging.  
Typical steps:

1. Push code to a branch.
2. Open a PR on GitHub.
3. Request reviews.
4. Address feedback.
5. Merge when approved.

**9. Forking vs. Cloning**

* **Forking** creates a personal copy of a repo under your account, useful for contributing to open-source projects.
* **Cloning** downloads a repo to your local machine for editing.

Use **forking** when you don’t have write access but want to contribute.

**10. How do GitHub issues and project boards help?**

Issues track bugs, feature requests, and improvements. Project boards organize tasks like a Kanban board. Example:

* Issue: "Fix login bug"
* Board columns: "To Do," "In Progress," "Done"

They keep projects organized and improve teamwork.

**11. Common GitHub challenges & best practices**

**Challenges:**

* Merge conflicts
* Forgetting to pull before pushing
* Messy commit history