



Tech Winter Break



Google Developer Groups

On Campus GEC Arwal



Google Developer Groups

On Campus GEC Arwal

About the Speaker: Kirtpreet Kaur

Kirtpreet is an emerging tech enthusiast with a passion for cybersecurity and automation. With her diverse experience at leading organizations like Nokia, Samsung, and IIT Hyderabad, she has honed her skills in various programming languages and cutting-edge technologies. She has also been a Kavach 2023 finalist and a silver medalist at the AIU National Women Parliament, showcasing her dedication and leadership in the tech community

.Key Achievements:

- Intern at Nokia Solutions and Networks, focusing on CNS Automation.
- Ex-R&D Intern at Samsung Prism and IIT Hyderabad.
- Kavach 2023 Finalist and Silver Medalist at the AIU National Women Parliament.
- Proficient in C, C++, Python, SQL, and Java.
- We are thrilled to have Kirtpreet share her knowledge and insights with us today as part of our Git & GitHub Workshop!



Kirtpreet Kaur

NCOM CNS Automation Intern,
Nokia | Ex-R&D Intern, Samsung
Prism | Ex-Intern, IIT Hyderabad |



Google Developer Groups

On Campus GEC Arwal

Welcome to the Git & GitHub Workshop!



git



On Campus GEC Arwal

Today's agenda:

1. Git - Introduction and basic commands

Learn how to **install Git**, **navigate Git Bash**, and use **basic commands** along with creating **branches** and **merging changes**.

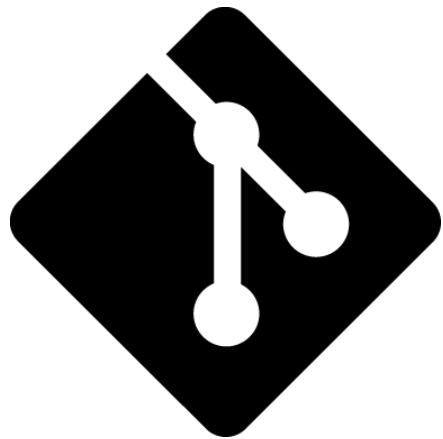
1. GitHub - Contribute and Collaborate

Discover how to **create and clone repositories**, collaborate by **pushing changes**, **fork** repositories, create **pull requests**, and manage contributions from a maintainer's perspective.

1. Q&A and Networking



Part 1: **Git**



```
lookup.KeyValue  
f.constant(['en  
=tf.constant([G  
lookup.StaticV  
_buckets=5)
```

```
child: Column(  
  crossAxisAlignment: CrossAxisAlignment.  
  children: [  
    /*2*/  
    Conta  
    pad  
    chi  
    's  
    )  
  ),  
  Text(  
    'Ka  
    sty  
    c  
    ),  
  ),  
),
```



What is Git?

- Git is officially defined as a *distributed version control system* (VCS).
- It's a system that tracks changes to our project files over time
- It is used to efficiently work together and collaborate on team projects



Google Developer Groups On Campus

How about **GitHub**?

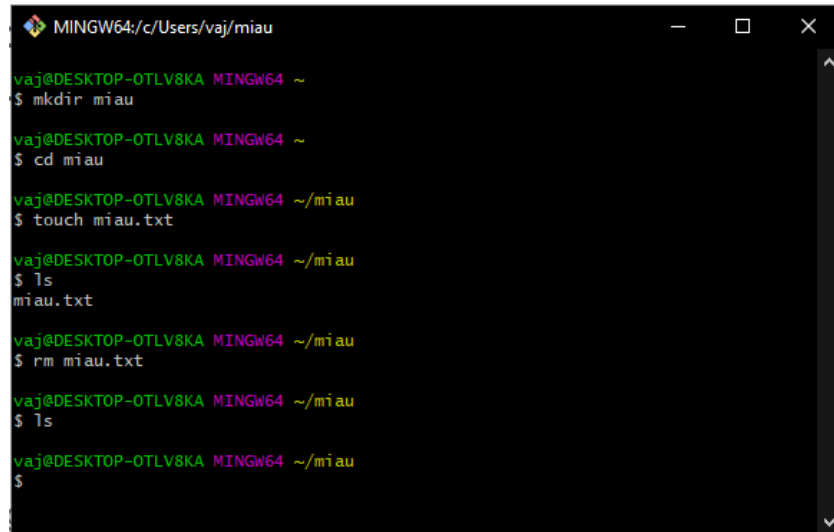
- GitHub is a hosting platform for Git repositories
- Using GitHub, we can upload a local project repository to a remote cloud-based GitHub Repository
- GitHub is also a popular way developers for developers to publish their portfolio online



```
lookup.KeyValue  
f.constant(['em  
=tf.constant([G  
lookup.StaticV  
_buckets=5)
```

Useful **navigation commands** in the command-line interface

- **mkdir** – creates directory
- **touch** – creates file
- **cd** – changes directory
- **rm** – removes directory



```
MINGW64: c:/Users/vaj/miau

vaj@DESKTOP-OTLV8KA MINGW64 ~
$ mkdir miau

vaj@DESKTOP-OTLV8KA MINGW64 ~
$ cd miau

vaj@DESKTOP-OTLV8KA MINGW64 ~/miau
$ touch miau.txt

vaj@DESKTOP-OTLV8KA MINGW64 ~/miau
$ ls
miau.txt

vaj@DESKTOP-OTLV8KA MINGW64 ~/miau
$ rm miau.txt

vaj@DESKTOP-OTLV8KA MINGW64 ~/miau
$ ls

vaj@DESKTOP-OTLV8KA MINGW64 ~/miau
$
```


Setup instructions

- Install git on your computers. Go to <https://git-scm.com/downloads>
- After installing it, start your terminal and type the following command:

```
git --version
```

Configuring

- `git config --global user.name "Your Name"`
- `git config --global user.email your@email.com`
- not used to log in anywhere, used to track who made what changes

```
lookup.KeyValue  
f.constant(['em  
=tf.constant([G  
lookup.StaticV  
_buckets=5)
```

Repositories

- A Git repository is a **container** for a project that is tracked by Git
 - **Local repository** = an isolated repository stored on your own computer
 - **Remote repository** = generally stored outside of your isolated local system, usually on a remote server; useful for working in teams
- **Initializing a repository (git init)**
 - If you want to use git to track the changes in a folder you have to initialize it first
- **Checking the status (git status)**
 - used to check changes and tracked/untracked files

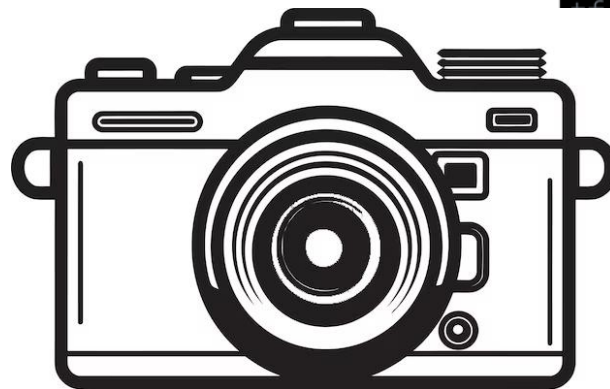


git add

- Adds files to the staging area, which allows git to track those files
- syntax:
 - **git add file.js** – one file
 - **git add file.js file2.js file3.js** – multiple files
 - **git add .** – all the files and folders
- Checking the status (**git status**)
 - used to check changes and tracked/untracked files

git commit

- Saves a snapshot of your code at a particular time
- syntax:
 - **git commit -m „Commit message”**



```
child: Column(  
  crossAxisAlignment: CrossAxisAlignment.  
  children: [  
    /*2*/  
    Conta  
    pad  
    chi  
    '  
    s  
    )  
  ),  
  Text(  
    'Ka  
    sty  
    c  
    ),  
  ),  
),
```



git log

- used to see the commit history

git checkout

- Used to go to a previous state of your code that you committed
 - syntax: **git checkout <commit-hash>**
 - You can see the commit hashes when you use git log
- To go back you can use:
 - git checkout master

 Google Developer Groups

git rebase (Teo's favourite command)

- A more advanced command often used in professional workflows to keep commit history clean.
- Integrates changes from one branch into another while keeping a linear history
- Unlike git merge, it rewrites commit history to eliminate unnecessary merge commits.
- Use it to update your branch with the latest changes from the main branch:

```
git fetch origin
```

```
git rebase origin/main
```

- Resolve conflicts and continue with:

```
git rebase --continue
```

- Avoid rebasing public branches to prevent rewriting shared history.

```
child: Column(  
  crossAxisAlignment: CrossAxisAlignment.  
  children: [  
    /*2*/  
    Conta  
    pad  
    chi  
    '  
    s  
  )  
),  
),  
Text(  
  'Ka  
  sty  
  c  
),  
),
```

Branches

- Individual timelines of our project commits, where versions of our project code can exist and be tracked in parallel
 - **git branch** - lists the branches
 - **git branch <new-branch-name>** - creates a new branch
- *Changing branches:*
 - **git checkout <branch-name>** (hence the git checkout master from before)



Google Developer Groups On Campus

Merging branches (git merge)

- Used to implement the code changes that you made in an individual branch to a different branch (usually on master)

- syntax:

git merge <branch-name>

- Deleting branches

- syntax:

git branch -d <branch-name>



Google Developer Groups

```
/*1*/  
child: Column(  
  crossAxisAlignment: CrossAxisAlignment.  
  children: [  
    /*2*/  
    Container(  
      padding: const EdgeInsets.  
      child: const Text(  
        'Oeschinen Lake Campg  
        style: TextStyle(  
          fontWeight: FontWeig  
      ),  
    ),  
  ],  
)
```

Part 2: GitHub

Step-by-step Guide to Collaboration and
Advanced GitHub Features



```
lookup.KeyValue  
f.constant(['en  
=tf.constant([G  
lookup.StaticV  
_buckets=5)
```



```
children: [
  Icon(icon, color: color),
  Container(
    margin: const EdgeInsets.all(10),
    child: Text(
      label,
      style: TextStyle(
        color: color,
        fontSize: 16,
        fontStyle: FontStyle.normal,
      ),
    ),
  ),
],
```

Create a GitHub account if you don't already have one

Welcome to GitHub!
Let's begin the adventure

Enter your email*

✓ gdg.unstpb@gmail.com

Create a password*

✓

Enter a username*

→ gdg-unstpb

Continue

gdg-unstpb is available.

Step 1: Create a GitHub Repository

- **Log in:** Go to GitHub and log into your account.
- **New Repository:** Click the + icon (top-right) → New repository.
- **Name:** Enter a name for your repository.
- **Visibility:** Choose *Public* (anyone can see) or *Private* (restricted access).
- **Optional:** Add a description, README, .gitignore, or license.
- **Create:** Click Create repository.
- **Push Code (Optional):** Use the commands provided to push your local code.



Step 2: Add Collaborators

- Go to your repository → **Settings**
- Click **Manage access** → **Invite a collaborator**
- Enter their GitHub username or email → **Add**
- Collaborator accepts the invitation. Done!

```
lookup.KeyValue  
f.constant(['em  
=tf.constant([G  
.lookup.StaticV  
_buckets=5)
```

Step 3: Clone the Repository

1. Copy the repository URL (HTTPS, SSH, or GitHub CLI).
2. Open your terminal.
3. Run: **git clone <repo-url>**
4. Navigate to the cloned repository folder.

```
cd repo-name
```

```
Lookup.KeyValue  
f.constant(['em  
=tf.constant([G  
.lookup.StaticV  
_buckets=5)
```

Step 4: Create Your Own Branch

1. Run the command:

```
git branch <branch-name>
```

1. Switch to your branch:

```
git checkout <branch-name>/git switch <branch-name>
```

1. Verify the branch:

```
git branch
```

1. Push the branch:

```
git push -u origin <branch-name>
```

```
Lookup.KeyValue  
f.constant(['en  
=tf.constant([G  
.lookup.StaticV  
_buckets=5)
```



- Save your work
 - **git stash**
- Come back later
 - **git stash pop**
- See what's saved
 - **git stash list**

```
er(
l(32),

/*1*/
child: Column(
    crossAxisAlignment: CrossAxisAlignment
children: [
    /*2*/
    Container(
        padding: const EdgeInsets
child: const Text(
    'Oeschinen Lake Campg
    style: TextStyle(
        fontWeight: FontWei
    ),
    ),
)
```

Step 6: Push Changes

- Create a new file and add content.
 - `git add <file-name> / git add . (all)`
 - `git restore --staged <file-name> / .(all)`
- Commit:
 - `git commit -m "message"`
- Push changes:
 - `git push origin <branch-name>`

Step 7: Pull Request and Merge

1. Go to the GitHub repository.
2. Click on '**Pull Requests**' and select '**New Pull Request**'.
3. **Compare changes** and **select your branch**.
4. Add a description and click '**Create Pull Request**'.
5. **Merge** the request after review.

Step 8: Fork a Repository

1. Go to the repository you want to fork.
2. Click the '**Fork**' button on the *top right*.
3. Choose your account to fork the repository.
4. **Clone** the forked repository and *start contributing*.

OVERVIEW

3 scenarios

```
lookup.KeyValue  
f.constant(['en  
=tf.constant([G  
lookup.StaticV  
_buckets=5)
```

Scenario 1: Working Solo on Your Own Projects

(I really don't know what's wrong with the indents on these slides. They look fine on PowerPoint and Google Slides, but they get messed up when uploaded here)

1. Create a directory on your computer:

```
mkdir <project-directory>  
cd <project-directory>
```
2. Initialize a local repository:

```
git init
```
3. Create a GitHub repository and connect it as the remote origin:

```
git remote add origin <repo-url>
```
4. Make changes to your files and track them:

```
git add .  
git commit -m "Describe changes"
```
5. Push changes to GitHub:

```
git push -u origin master
```

Your project is now on GitHub, and you can track its progress and share it with others if needed.

Scenario 2: Collaborating with Friends

Repository Owner:

Add collaborators on GitHub: Navigate to **Settings > Access > Collaborators > Add People**

Collaborators:

Clone the repository:

```
git clone <repo-url>
```

Everyone:

Create and switch to your own branch:

```
git branch <branch-name>
```

```
git checkout <branch-name>
```

Make changes, stage, and commit:

```
git add .
```

```
git commit -m "Describe changes"
```

Push your branch to GitHub:

```
git push -u origin <branch-name>
```

Open a pull request on GitHub:

Go to **Pull Requests > Compare & pull request**

Select your branch to compare with the main branch.

Resolve conflicts (if any) and merge the pull request.

Useful VS Code extensions:

Git Graph: Visualize commit history and branch structure.

GitHub Pull Requests: Manage PRs and resolve conflicts in VS Code.

GitLens: View file history, authorship, and commit details.

Scenario 3: Collaborating to a larger project

(where you are not collaborator)

1. Fork the repository:
Click the **Fork** button in the top-right corner of the repository on GitHub.
2. Clone your forked repository:
`git clone <your-fork-url>`
3. Create and switch to a new branch:
`git branch <branch-name>`
`git checkout <branch-name>`
4. Make changes, stage, and commit:
`git add .`
`git commit -m "Describe changes"`
5. Push changes to your fork:
`git push origin <branch-name>`
6. Open a pull request:
Go to the original repository on GitHub.
Navigate to **Pull Requests > Compare & pull request**
Select your branch from your fork and propose it for the base branch of the original repository.



Q&A (we forgot to show this part)

The background is a solid blue color with various white abstract elements. There are several thin, wavy white lines that meander across the frame. Scattered throughout are small white geometric shapes, including triangles and circles. In the bottom left corner, there is a small cluster of white dots. The overall aesthetic is modern and minimalist.

THANK YOU