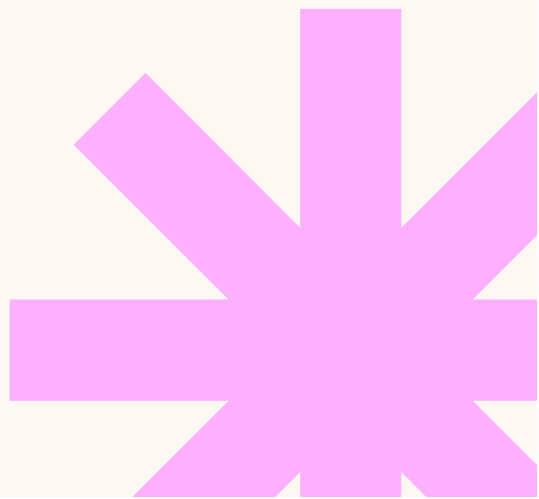
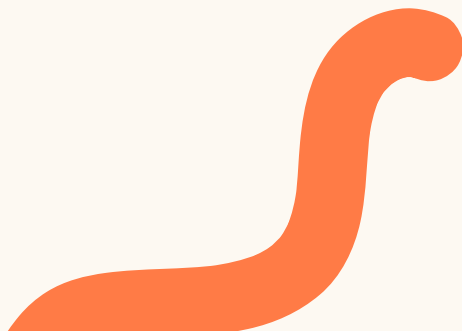
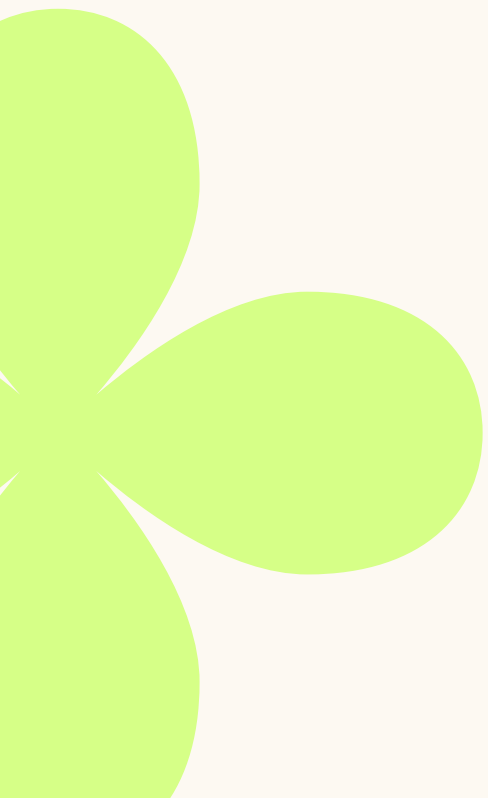
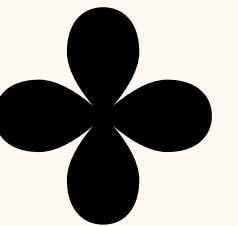


Do You Git it?

**Get ready to
understand Git.**





Welcome to Gitville!

Get ready to *explore* the
exciting world of **Git**!
Fasten your seatbelts as
we embark on a journey
to visit git and start on
our Dev journey.

What is Git?

Git is a **superhero *version control system*** that saves the day by tracking changes in your code. It's like a magical time machine for your files, allowing you to revisit any version of your project.



Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.

It is used for:

Tracking code changes

Tracking who made changes

Coding collaboration

Version Control System

a software tool that helps manage changes to files, documents, or any other collection of information over time. It enables multiple people to collaborate on projects by keeping track of every modification to files and providing mechanisms to coordinate and merge these changes.

Github

GitLab

BitBucket
(lol xD)

GitHub: The Social Hub!

Welcome to **GitHub**, the social hub for sharing and collaborating on code. It's like a bustling city where developers come together to build amazing projects.

Just search [Github.com](https://github.com)



Why Git?

- Over 70% of developers use Git!
- Developers can work together from anywhere in the world.
- Developers can see the full history of the project.
- Developers can revert to earlier versions of a project.

What is GitHub?

- Git is not the same as GitHub.
- GitHub makes tools that use Git.
- GitHub is the largest host of source code in the world,

**Do you have what it takes to
git it?**

To install Git, visit the official website
<https://git-scm.com/>
and download the appropriate version for your OS.

**open the terminal/cmd and
execute:
git --version**

```
PS C:\Users> git --version  
git version 2.37.2.windows.2  
PS C:\Users>
```

Set up Git on your laptops by first installing Git, then configuring your username and email address using the commands:

```
git config --global user.name "Your Name"
```

```
git config --global user.email "youremail@example.com"
```

CHECK:
git config --list

and look for the name and username you just entered





KEY TERMS

A "**repo**" is a tracked Git project folder where all the project files and version history are stored.

A "**commit**" is a snapshot of the project at a specific point in time.

The "**staging area**" or "**index**" is where changes are prepared before committing.

"**SHA**" is a unique identifier for each commit.
"**Untracked/Staged/Tracked**"

Untracked:

These files exist in your project directory but haven't been added to the Git repository yet. Git doesn't track changes to untracked files until you add them using **git add**

Staged:

Staged files are marked for inclusion in the next commit. You use **git add** to stage files, indicating changes you intend to include in the next commit.

Tracked:

Tracked files are those previously committed to the Git repository or staged for the next commit. They include modified files waiting to be staged and files already staged for commit.

More than 150 commands (....maybe)

But there is good news

Most developer experience is revolving around just a few of these commands:

push
pull
add
commit
status
log
fetch

local
repo

push ->
<- pull

remote
repo

check
out

↑↓ commit / reset

staging area

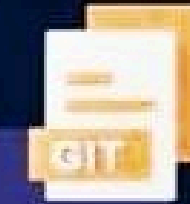


↑↓ add / restore

working tree

```
.  
├── go.mod  
└── main.go
```


Git Cheatsheet



```
git init // Initialize a new Git repo
git clone <repo-url> // Clone a repo from a URL

git status // Show active changes & status
git add <file> // Stage active changes
git commit -m "Message" // Commit changes with a message
git log // View commit history

git branch // List local branches
git branch <branch-name> // Create a new branch
git checkout <branch-name> // Switch to a branch
git checkout -b <branch-name> // Switch & create a new branch
git merge <branch-name> // Merge changes from a branch
git rebase <branch-name> // Rebase from a branch
git branch -D <branch-name> // Delete a branch

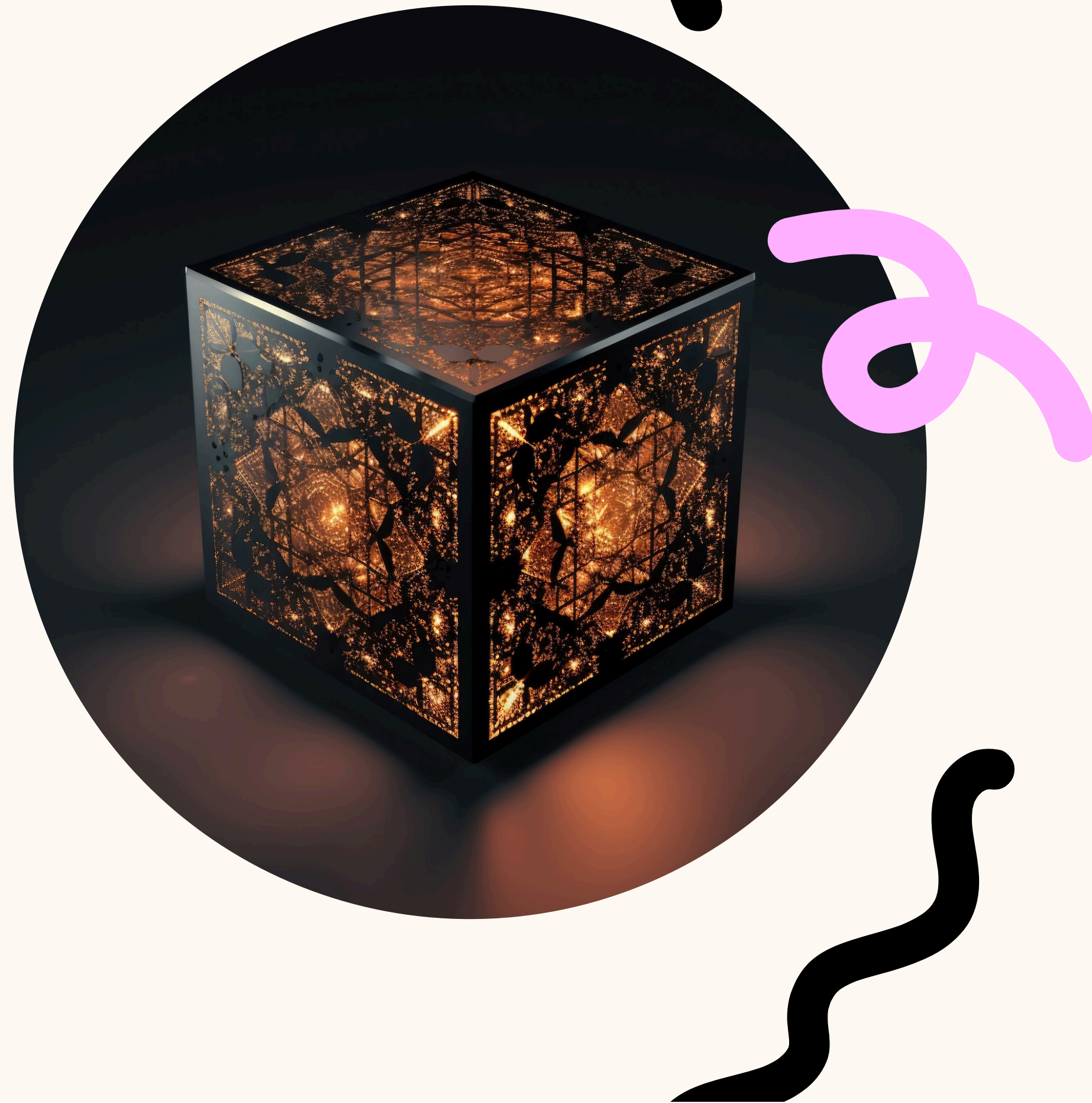
git remote // List remotes
git remote add <name> <url> // Add a remote
git push <remote> <branch> // Push changes to a remote
git push --delete <remote> <branch> // Delete a branch from remote
git pull <remote> <branch> // Pull changes from a remote


git pull // Fetch and merge changes
git fetch // Fetch changes without merging
git reset --hard HEAD // Discard active changes
git revert <commit-hash> // Revert changes in a commit
```

LETs GIT it DONE

Mastering Git: The Adventure Continues!

Congratulations on completing the fun guide to understanding Git! Now, it's time to embark on your own coding adventures with confidence, creativity, and a dash of Git magic.





The Git Finale: Happy Coding!

As our fun guide to understanding Git comes to an end, remember to keep coding with joy, curiosity, and the power of Git! Happy coding, and may your projects always be version-controlled and full of fun!



Thanks!



DO YOU HAVE ANY QUESTIONS?

<https://github.com/Bi7al>

<https://github.com/mutahir-muhammad>



alphaberoose

