DEPARTMENT OF STATISTICS AND ACTUARIAL SCIENCE

THE UNIVERSITY OF IOWA , USA

STAT-5400 : COMPUTING IN STATISTICS INSTRUCTIONAL PROJECT – FALL 2025

THE UNIVERSITY OF IOWA

INTRODUCTION TO GITHUB :

WEBPAGE DEVELOPING, VERSION CONTROL, GITHUB COLLABORATION & REPRODUCIBLE TEAM WORKFLOW

SUPERVISOR : PROF.BOXIANG WANG

AUTHORS :

SHOUHARDYO SARKAR

DYLAN DAY

TREVOR DEBUTCH

SHUO LI

NOVEMBER 7 ,2025

WHAT WE WILL SHOW IN THIS TALK!

- What is Git and Github?
- How to create a repository and invite collaborators?
- What Git and Github actually do
- Web Page developing using Github?
- How teams collaborate using branches,commits and pull requests?
- Version Control?
- How to connect a local folder to a Github repo and push/pull?
- How this supports small coding projects?
- Practical example on how to use basics of github?
- Some exercises to work on?

CONTRIBUTIONS OF EACH OF THE TEAM MEMBERS

Shouhardyo Sarkar

Primary Role: Serves as the foundation builder of the instructional project by setting up the GitHub repository, writing all core documentation, and designing the theoretical framework and presentation slides.

Technical Contributions

1. GitHub Repository Setup

Created and configured the project repository on GitHub

2. Core Documentation Development

Authored a comprehensive README.md file including:

- Project title and purpose
- Introduction to GitHub and its benefits
- Step-by-step setup instructions for beginners
- Overview of GitHub features: commits, branches, pull requests

Created supporting documentation:

- CONTRIBUTING.md – guidelines for collaboration
- GLOSSARY.md – definitions of key GitHub terms
- FEATURES.md – highlights of GitHub's advanced tools

Embedded a custom GitHub workflow diagram to visually explain the process

Creative and Instructional Contributions

3. Theoretical Framework and Introductory Content

Researched and wrote the conceptual introduction to GitHub

4. Presentation Slide Design

Designed and customized the team's PowerPoint slides - Structured the flow of the presentation:

- Introduction to GitHub
- Repository setup walkthrough
- Documentation overview
- Transition to webpage development and version control

Ensured clarity, consistency, and visual appeal across slides

Dylan Day

Primary Role: Responsible for transforming the instructional GitHub content into a live, accessible webpage using GitHub Pages. This contribution bridges the technical documentation with a user-friendly web interface, making the project visually engaging and easy to navigate.

Key Contributions:

Defined Webpage Development in GitHub Context Explained how static websites can be built and hosted directly from GitHub repositories, aligning with the project's instructional goals.

Highlighted GitHub's Strengths for Web Hosting Showcased GitHub as a collaborative, version-controlled platform ideal for publishing educational content and team projects.

Introduced GitHub Pages Demonstrated GitHub Pages as a free, built-in tool for hosting static sites — perfect for showcasing documentation, tutorials, and project portfolios.

Walked Through Setup Steps

- Created a dedicated repository for the webpage
- Added HTML, CSS, and JavaScript files
- Committed and pushed changes to GitHub
- Enabled GitHub Pages via repository settings
- Verified the live site URL and functionality

Covered Maintenance and Troubleshooting

- Explained how to update content and fix layout issues
- Provided tips for syncing changes and managing file structure

Summarized Key Benefits

- Emphasized collaboration, version control, and professional workflow
- Reinforced how GitHub Pages supports scalable and reproducible web publishing

Trevor Debutch

Primary Role: Responsible for demonstrating how GitHub supports professional collaboration through version control, issue tracking, and workflow management.

Explained Version Control with GitHub

Showed how GitHub tracks changes over time using commits
Demonstrated branching and merging to manage multiple versions of a project
Emphasized the importance of commit messages for clarity and traceability

Demonstrated Commit and Merge Workflow

Created branches to isolate new features or edits
Made commits to document changes
Submitted pull requests to propose updates
Merged branches into the main codebase after review

Used GitHub Issues for Task Management

Created issues to track bugs, tasks, and suggestions
Assigned issues to team members for accountability
Added labels and comments to organize and discuss work

Showcased Collaboration Tools

Used pull requests for peer review and feedback
Linked commits to issues for traceability
Highlighted how GitHub supports asynchronous teamwork

Reinforced Professional Workflow Practices

Encouraged clear commit history and structured branching
Demonstrated how GitHub helps teams avoid conflicts and maintain project integri
Connected version control to real-world software development standards

IOWA

Shuo Li

Primary Role: Focused on bringing the instructional content to life by enabling GitHub Pages, deploying practical code examples, and supporting the team's presentation design. This role bridges the gap between theory and application, showing how GitHub can be used to host and demonstrate real projects.

Technical Contributions

1. Enabled GitHub Pages for Live Hosting

Configured the repository to serve a static website using GitHub Pages

Selected the correct branch and folder (main or /docs) for deployment

Verified the live site URL and ensured it reflected the latest content

2. Uploaded Toy HTML and Python Code

Created and uploaded a basic index.html file to demonstrate static site hosting

Added a toy Python script to show how GitHub handles code versioning and collaboration

Used these examples to explain how GitHub supports both web and software development

3. Demonstrated Local-to-GitHub Workflow

Explained how to:

- Create and edit files on a local computer
- Use Git or GitHub Desktop to commit and push changes
- Sync local updates with the GitHub-hosted project

Highlighted the importance of version control and reproducibility in development

Creative Contributions

4. Designed and Shared Slide Templates

Created a clean, consistent slide template for the team presentation

Ensured visual alignment with GitHub's theme (colors, icons, layout)

Helped integrate diagrams, code snippets, and screenshots into the slides

Repository and README file Creation,Documentation and exploring Github

WHY GITHUB??

GitHub is necessary because it enables version control, seamless collaboration, and secure code management for developers and teams across the globe. It’s the backbone of modern software development workflows.

- **Why GitHub Is Indispensable in Today’s Development Landscape**

GitHub isn’t just a code hosting platform—it’s a comprehensive ecosystem that supports the entire lifecycle of software development. Here’s why it’s essential:

- **Version Control and Code History**

Built on Git, GitHub allows developers to track every change made to a project.We can revert to previous versions, compare changes, and understand who made what edits and why.This is critical for debugging, auditing, and maintaining clean, organized codebases.

- **Collaboration Across Teams**

GitHub supports **branching and merging**, enabling multiple developers to work on features or fixes without interfering with the main codebase.**Pull requests** and **code reviews** foster peer feedback and ensure quality control.
Teams can work asynchronously across time zones, making it ideal for remote and distributed setups.

- **Project Management Integration**

GitHub includes built-in **issue tracking**, **Kanban boards**, and **milestones** to manage tasks and progress.
It integrates with CI/CD pipelines, enabling automated testing and deployment.These features streamline workflows and reduce the need for external project management tools.

- **Security and Reliability**

GitHub offers **built-in security scanning**, dependency alerts, and secret detection.Enterprise-grade features ensure scalability and compliance for large organizations.Over 90% of Fortune 100 companies rely on GitHub for secure development.

- **Community and Open Source**

GitHub is the largest host of open-source projects, fostering a vibrant community of contributors.Developers can showcase their work, contribute to others’ projects, and build reputations through public repositories.

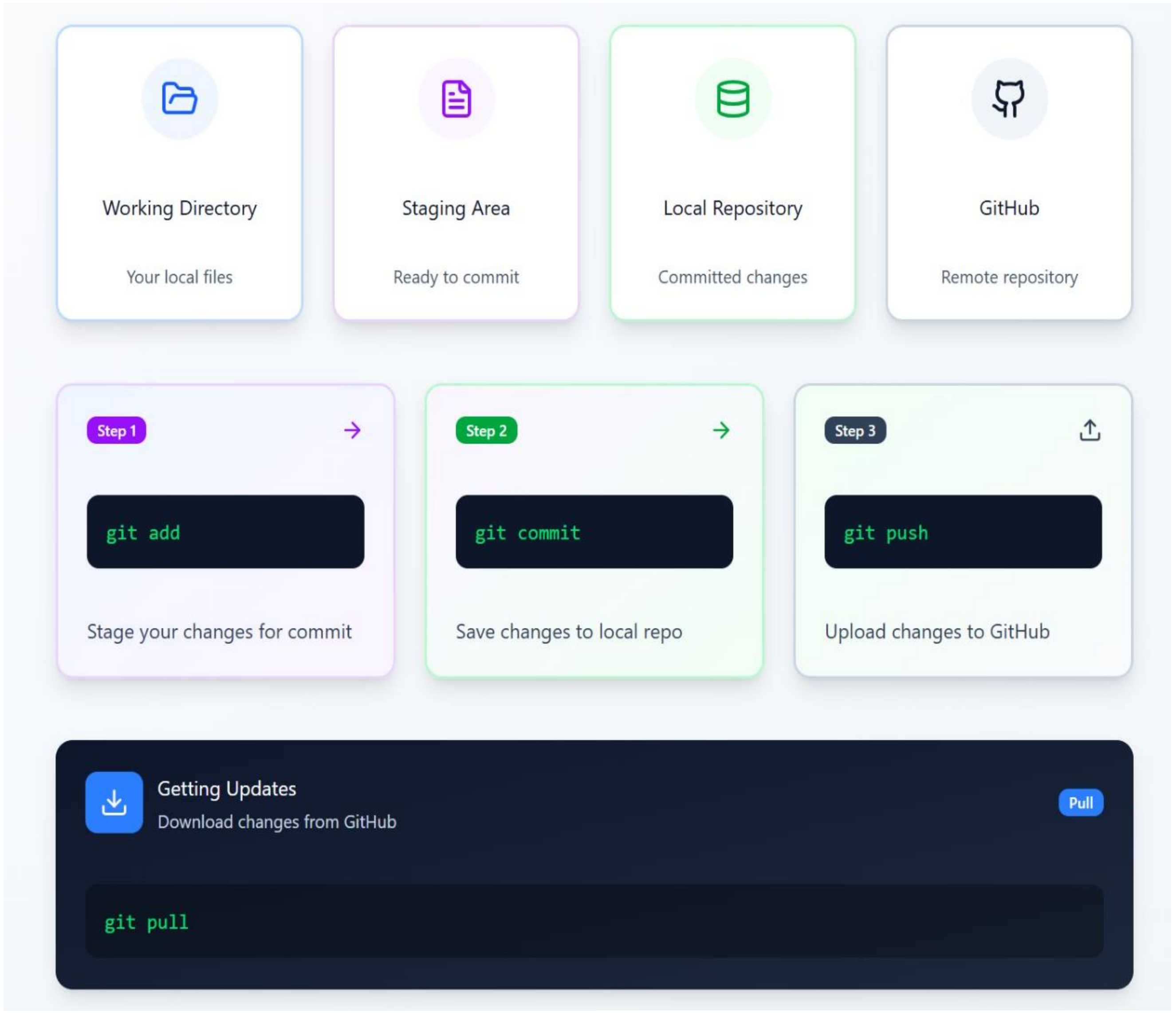
What Is Git?

- Created by **Linus Torvalds** (2005).Distributed, fast, and open-source
- **Common commands:**
- **git push(Upload) / git pull(Download)**: sync with remote
- **git branch / git merge**: manage branches
- **git add:** adds changes to staging area
- **git commit:** takes snapshots and saves changes locally

- **Git** is a *version control system* — a tool that tracks changes in your files over time. It lets you:
- Save snapshots of your project (called commits)
- Revert to earlier versions if something breaks
- Work on different features in parallel using branches
- Think of Git as a time machine for your code.

What Is GitHub?

- **GitHub** is a *cloud-based platform* that hosts Git repositories online. It adds collaboration tools like:
- Sharing code with others
- Reviewing changes via pull requests
- Tracking issues and tasks
- Hosting websites with GitHub Pages
  - Git is the engine; GitHub is the garage where your team works together.
- **Sources: Git vs GitHub explained**



```
git add
```

```
git commit
```

```
git push
```

```
git pull
```

IOWA

Key Git and GitHub Terms

Term	Definition
Repository (repo)	A folder that contains your project files and Git history. Think of it as your project’s home.
README.md	A markdown file that introduces your project — what it does, how to use it, and setup steps.
Commit	A saved snapshot of your project at a specific point in time. Includes a message describing the change.
Branch	A separate line of development. You can create branches to work on features without affecting the main code.
Push	Uploads your local commits to GitHub so others can see your changes.
Pull	Downloads the latest changes from GitHub to your local machine.
Pull Request (PR)	A request to merge changes from one branch into another. Used for code review and collaboration.
Merge	Combines changes from one branch into another (usually into `main`).
Issue	A task, bug, or suggestion tracked in the GitHub repo. Can be assigned and discussed.

This step-by-step guide walks you through setting up a GitHub repository, writing documentation.

Step 1: Create a GitHub Account

Go to https://github.com

Click **Sign up**

Choose a username, enter your email, and create a password

Verify your email and complete the setup

Step 2: Create a New Repository

After logging in, click the **+** icon in the top-right corner → **New repository**

Fill in:

- **Repository name** (e.g., github-intro)
- **Description** (optional but helpful)
- Choose **Public** (anyone can see it) or **Private** (only invited collaborators)
- Check **Add a README file**

Click **Create repository**

Step 3: Write and Edit the README.md

The README.md is the front page of your project.Click on README.md in your repo. Click the **pencil icon** to edit

Add:

- Project title and purpose
- Write your content of the project
- Setup instructions
- Key GitHub features (commits, branches, pull requests).Scroll down and click **Commit changes**

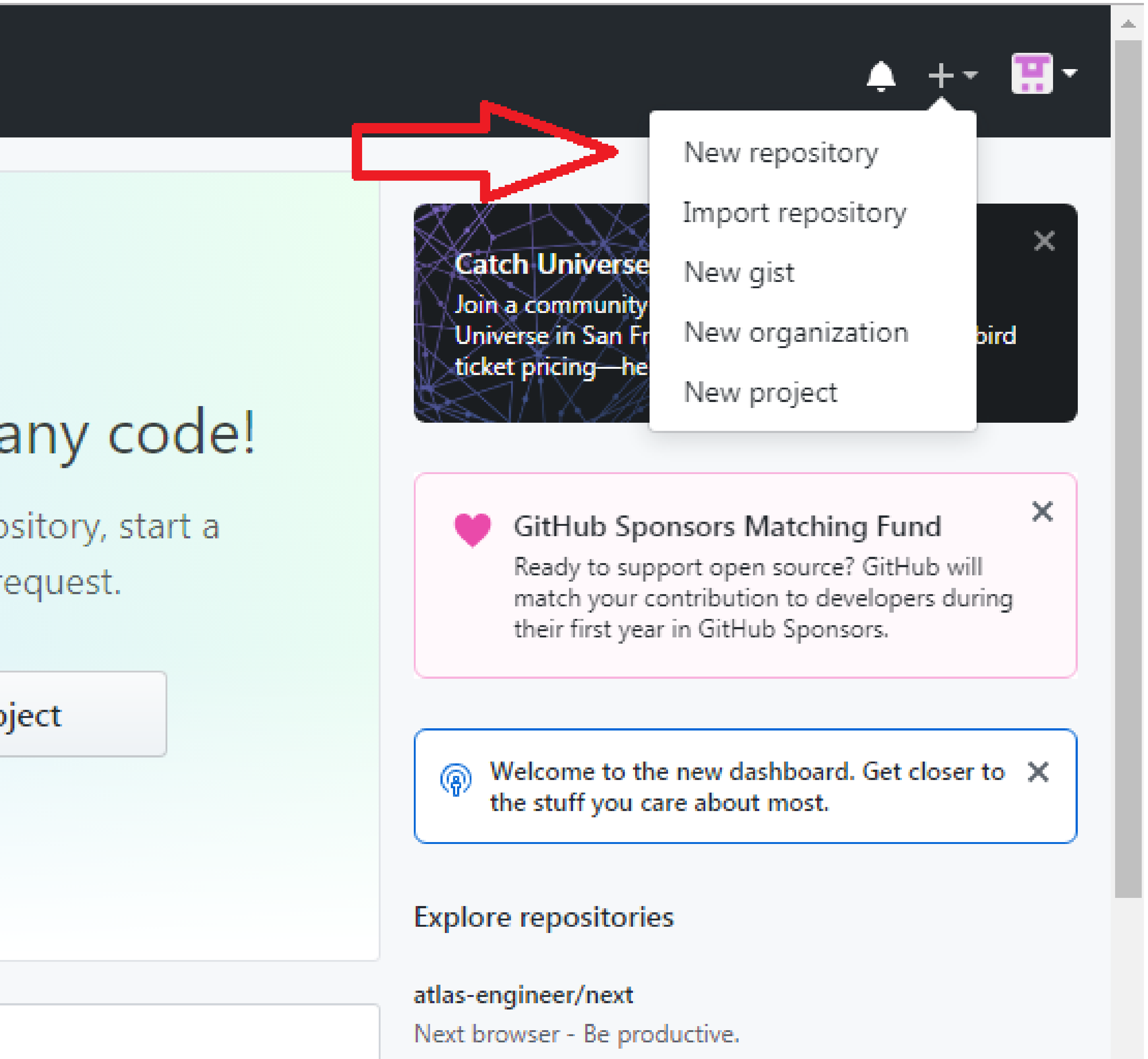
Feature	What It Does
Commit	Saves a snapshot of your changes with a message (e.g., “Added intro section”)
Branch	Creates a separate version of your project to work on new features safely
Push	Uploads your local commits to GitHub
Pull	Downloads the latest changes from GitHub to your computer
Pull Request	Proposes changes from one branch to another (used for collaboration)
Merge	Combines changes from one branch into another
Issue	Tracks bugs, tasks, or suggestions

Step 5: Understand and Use GitHub Features

Step 6: Collaborate with Others

- Go to **Settings** → **Manage access**
- Click **Invite a collaborator**
- Enter their GitHub username or email
- They’ll receive an invite to join your repo

IOWA



New repository
Import repository
New gist
New organization
New project

Catch Universe
Join a community
Universe in San Fr
ticket pricing—he
bird

any code!
ository, start a
request.

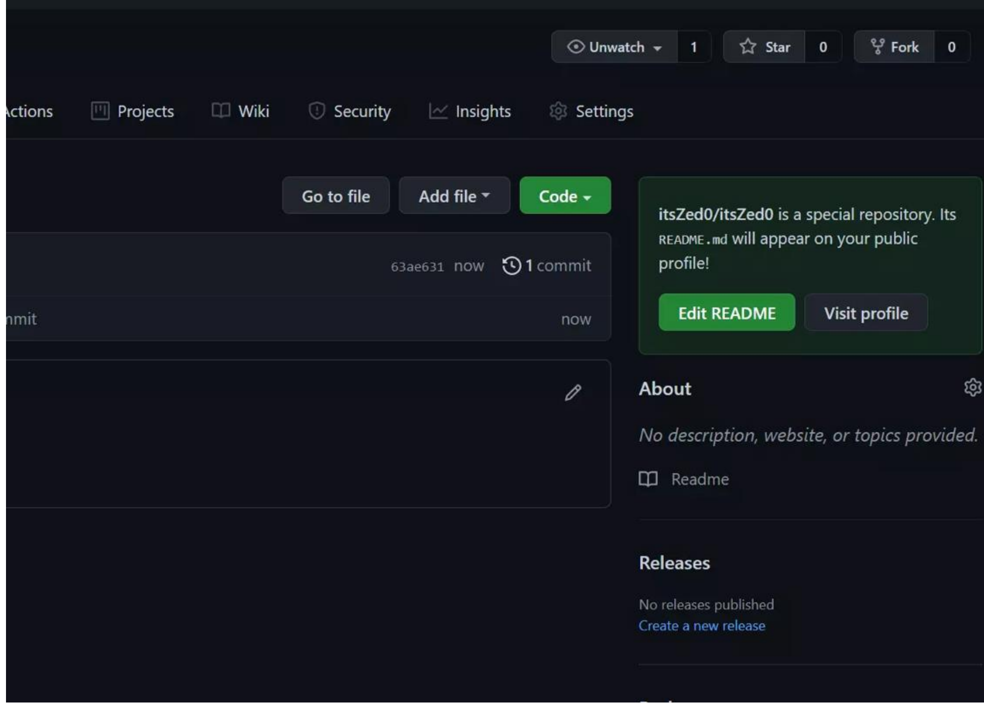
oject

GitHub Sponsors Matching Fund
Ready to support open source? GitHub will match your contribution to developers during their first year in GitHub Sponsors.

Welcome to the new dashboard. Get closer to the stuff you care about most.

Explore repositories

atlas-engineer/next
Next browser - Be productive.

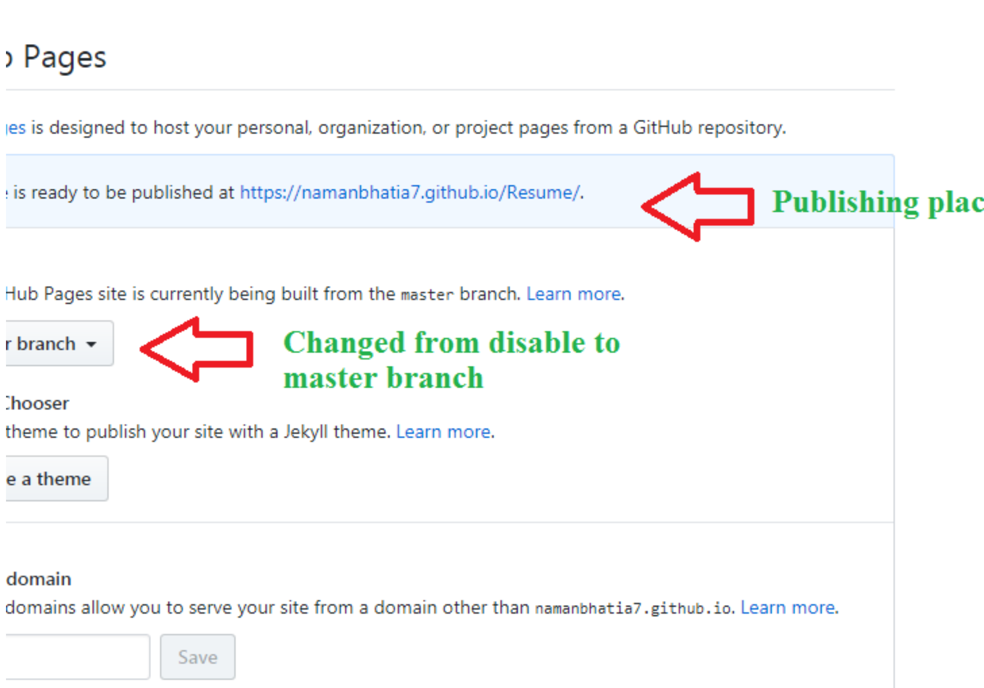


Unwatch 1 Star 0 Fork 0
ctions Projects Wiki Security Insights Settings
Go to file Add file Code
63ae631 now 1 commit
mit now

itsZed0/itsZed0 is a special repository. Its README.md will appear on your public profile!
Edit README Visit profile

About
No description, website, or topics provided.
Readme

Releases
No releases published
Create a new release



Pages

es is designed to host your personal, organization, or project pages from a GitHub repository.

is ready to be published at https://namanbhatia7.github.io/Resume/.

Publishing plac

Hub Pages site is currently being built from the master branch. Learn more.

r branch

Changed from disable to master branch

hooser
theme to publish your site with a Jekyll theme. Learn more.
e a theme

domain
domains allow you to serve your site from a domain other than namanbhatia7.github.io. Learn more.
Save

Webpage Development and GitHub Pages Integration

Webpage development refers to creating and designing websites using technologies like HTML, CSS, and JavaScript. Within GitHub, webpage development plays a key role in:

Presenting project content visually

Hosting documentation or tutorials

Creating portfolios or instructional sites

GitHub allows developers to store, version, and publish these web files directly from a repository.

Why GitHub Is an Effective Platform for Web Hosting and Collaboration

GitHub is ideal for hosting and collaborating on web projects because:

It provides **version control** via Git, so every change is tracked

Multiple contributors can **collaborate asynchronously**

It supports **GitHub Pages**, a free static site hosting service

It integrates with tools like Markdown, Jekyll, and custom domains

This makes GitHub a powerful platform for both development and deployment.

Introduce GitHub Pages as the Publishing Tool for Static Sites

GitHub Pages is a feature that lets you publish static websites directly from a GitHub repository.

Static site = a website built with HTML, CSS, and JS (no backend server).

Benefits:

Free hosting

Easy setup

Automatic updates from your repo

Customizable with themes or frameworks

Here’s how to build and publish a webpage using GitHub:

a. Create a Repository

Go to GitHub → Click **New repository**
Name it (e.g., web-demo)
Add a description
Initialize with a README (optional)

b. Add HTML/CSS/JS Files

Click **Add file** → **Upload files**
Upload your index.html, style.css, and script.js

c. Commit and Push Changes

Commit = save a snapshot of your changes
Push = send those changes to GitHub

d. Enable GitHub Pages

Go to your repo → **Settings** → **Pages**
Under **Source**, select main branch and /root or /docs folder
Click **Save**
GitHub will generate a live URL

e. Verify the Live Site

Visit the URL provided by GitHub Pages
Check that your HTML content appears correctly
Test links, layout, and responsiveness

IOWA

Cover Maintenance and Troubleshooting Basics

- **Maintenance Tips:**
- Edit files locally or on GitHub
- Commit and push updates regularly
- Use branches for major changes
- **Troubleshooting Tips:**
- Check file names and folder structure
- Ensure index.html is in the root or /docs
- Re-enable GitHub Pages if the site doesn't load
- Use browser dev tools to inspect layout issues

Benefit	Description
Collaboration	Multiple team members can edit and review content
Version Control	Every change is tracked, reversible, and documented
Professional Workflow	Mimics real-world development practices with commits, branches, and deployment

Version Control

What Is Version Control?

Version control records changes to files over time so you can recall specific versions later.
Think of it as a time line for your projects.
Key ideas:
- Track and revert changes
- Compare versions
- Restore previous states

Why Version Control Matters?

- Prevents data loss
- Enables team collaboration without overwriting
- Encourages experimentation with branches
- Keeps a complete project history

Collaboration and the Power of GitHub:

- GitHub turns version control into team control.
- Collaboration features:
- Branches: Safely develop new ideas
- Pull Requests: Review before merging
- Code Reviews: Improve quality
- Issues & Project Boards: Manage progress
- Actions: Automate testing and deployment

The Power of Collaboration:

- Git + GitHub combine distributed control with a social layer:
- Global team contributions
- Every change is tracked, attributed, and reversible
- Documentation and commits form shared knowledge
- Open-source communities thrive
- Version control can reduce errors and improve team efficiency.

Codespaces & Working Offline:

GitHub Codespaces brings your dev environment to the cloud.

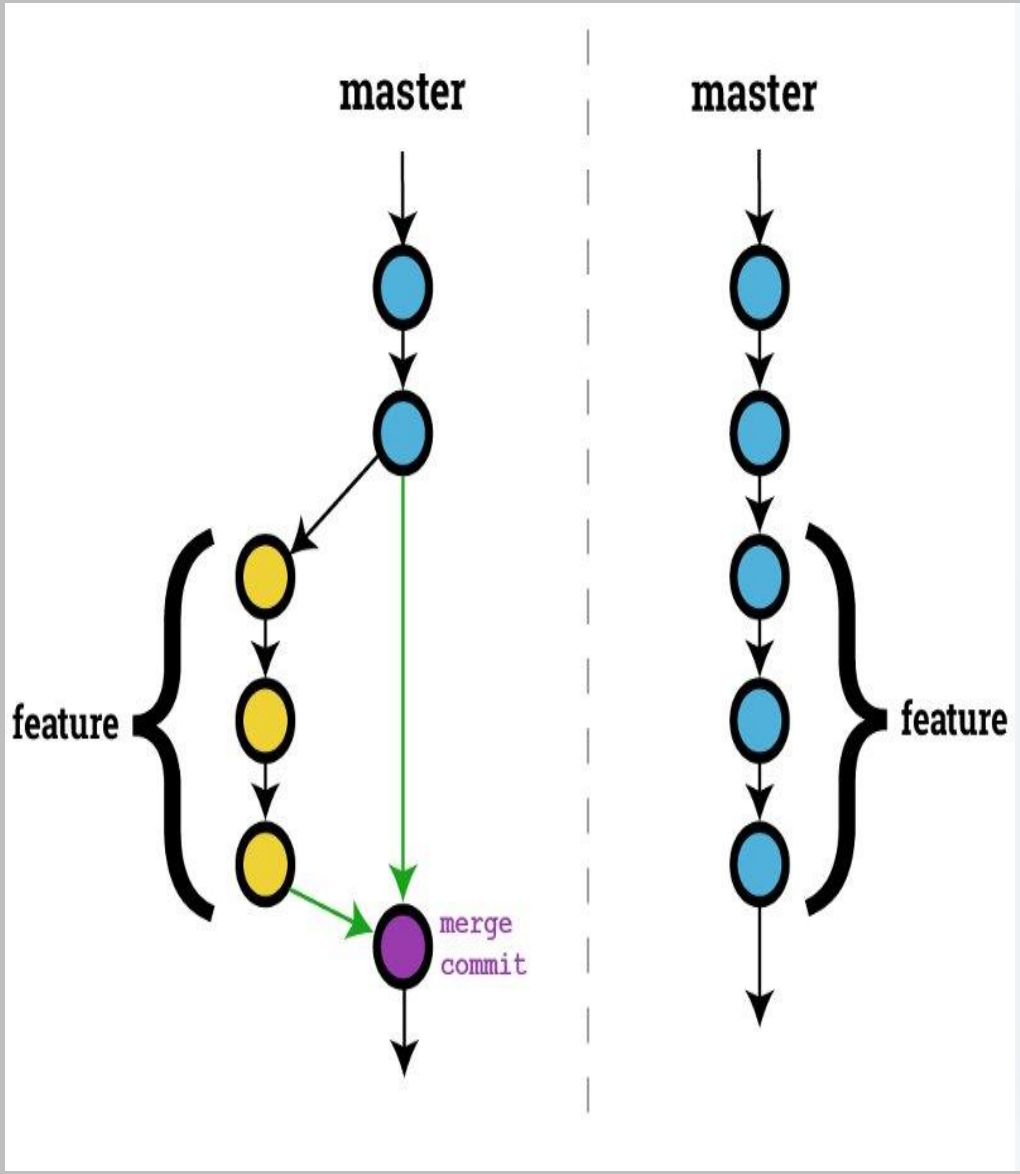
- **Benefits:**
- Launch in seconds
- Reproducible environments
- Seamless sync with repos

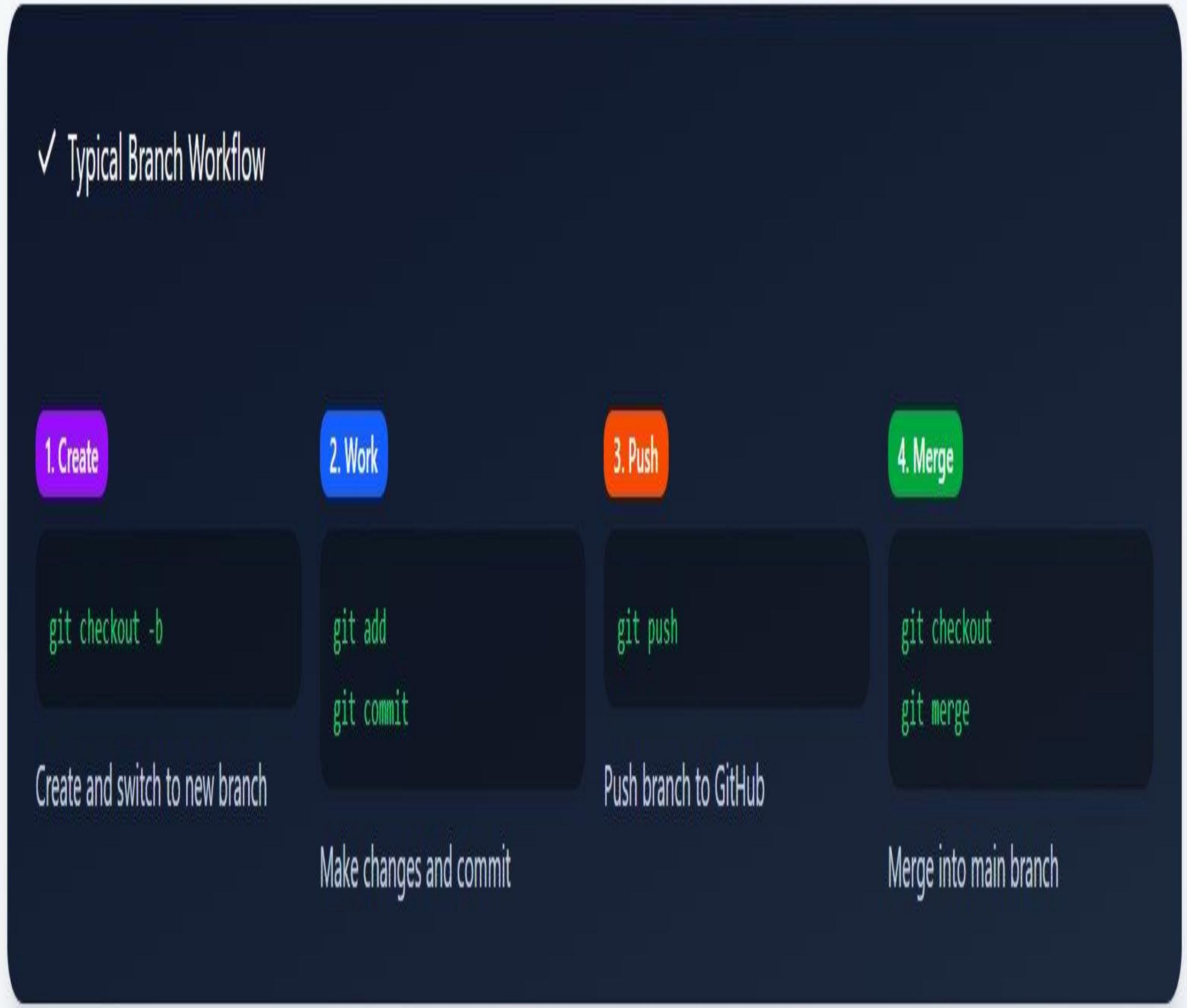
Even without Wi-Fi:

- Edit, commit, branch
- Review history
- Push later

“Version control doesn’t stop when Wi-Fi does.”

Version Control as a Timeline:
Think of version control as a timeline of your project’s evolution each commit operates as a checkpoint.
Each branch can serve to create different features without affecting the “main” product




✓ Typical Branch Workflow

1. Create

```
git checkout -b
```

Create and switch to new branch

2. Work

```
git add
git commit
```

Make changes and commit

3. Push

```
git push
```

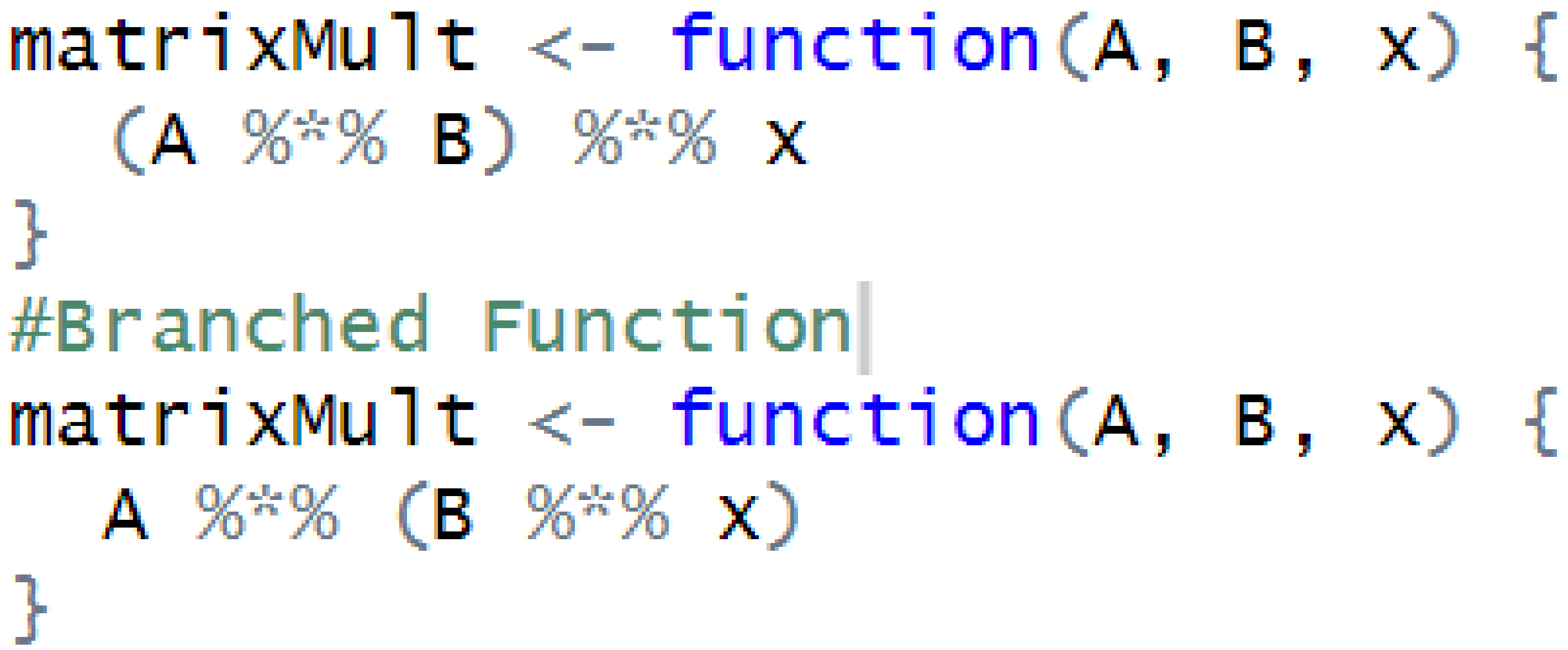
Push branch to GitHub

4. Merge

```
git checkout
git merge
```

Merge into main branch

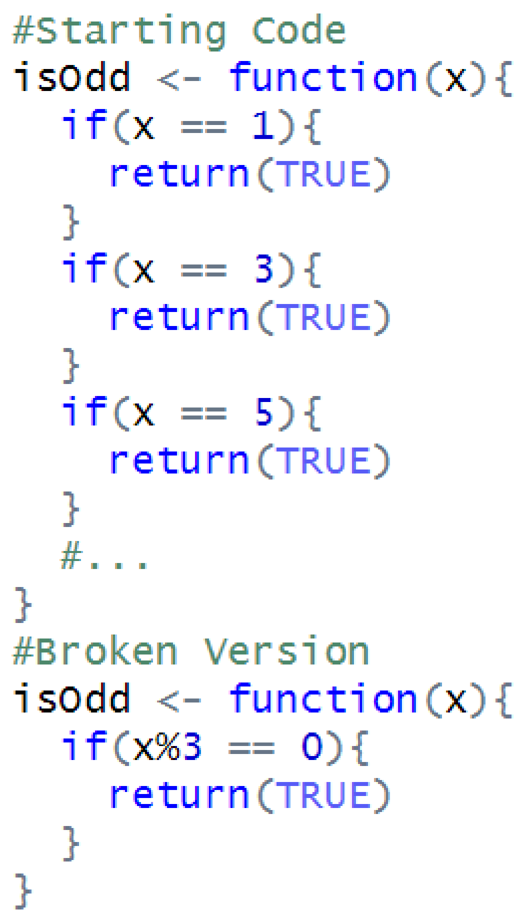
EXAMPLES - Branching



```r
matrixMult <- function(A, B, x) {
   (A %*% B) %*% x
}
#Branched Function
matrixMult <- function(A, B, x) {
   A %*% (B %*% x)
}
```

- See the above code; Say you have matrixMult as your current useable multiplication function for everyone (you or coworkers) who is using your functions. In a separate branch you can update the code to make it more efficient and test it to make sure it works without causing problems for any other code that calls the function, then you can merge the branch to main and update the function with the improved speed without ever breaking all other code in the repository.

EXAMPLES - Commit



```
#Starting Code
isOdd <- function(x){
  if(x == 1){
    return(TRUE)
  }
  if(x == 3){
    return(TRUE)
  }
  if(x == 5){
    return(TRUE)
  }
  #...
}
#Broken Version
isOdd <- function(x){
  if(x%3 == 0){
    return(TRUE)
  }
}
```

- Here we have an inefficient very long starting code that we want to improve, we hastily created the new isOdd function that doesn't work correctly, instead of having to recode all the previous code or worrying about creating a new working function so that people can use isOdd function we can go to our previous commits and revert to a previous form of our code. (This is why you commit often so you have plenty of snapshots you can go back to)

- **Use Git to control change.**
- **Use GitHub to collaborate and deploy.**
- **With version control, every change is accountable, reversible, and shared.
  It's the key to reproducible, reliable, and collaborative coding.**

Reference :

- For Branch image
- Ismail, H. A. (2017, January 28). *Learning how to Git: Merging branches and resolving conflict*. Medium. https://haydar-ai.medium.com/learning-how-to-git-merging-branches-and-resolving-conflict-61652834d4b0

- **GitHub Docs – Official Documentation** - https://docs.github.com/

- **Git vs GitHub – Kinsta Blog -https://kinsta.com/blog/git-vs-github**

Now we will show some practical applications on the basics of using Github!!

THANK YOU !!

IOWA