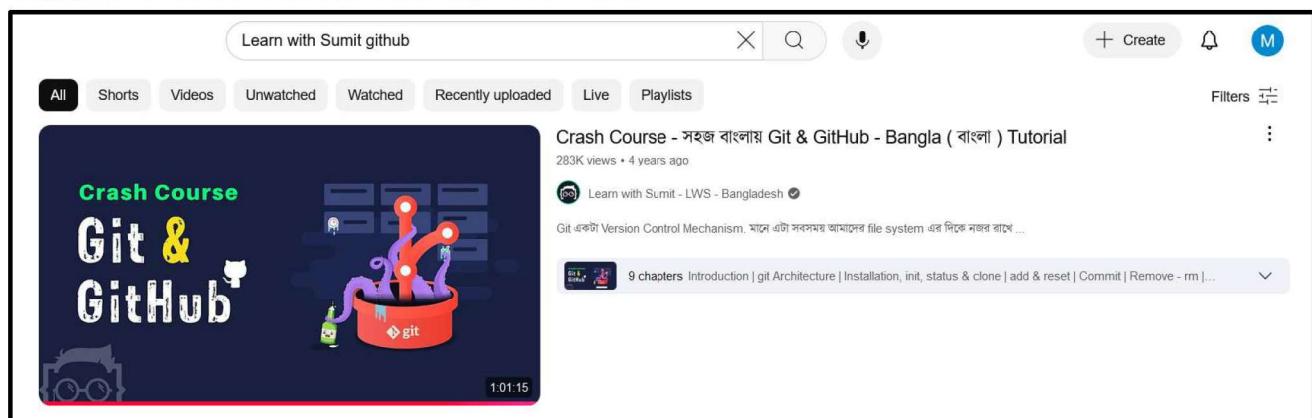


Class #10: Git and GitHub

1. Difference between Git and GitHub.
2. Installing and configuring Git.
3. Creating a GitHub account.
4. Creating repositories and deploying on GitHub.
5. Basic bash commands. Assignment #10:

Youtube এ বাংলায় এই লিংক থেকে Git শিখতে পারো।



1. Difference between Git and GitHub.

Git একটি ভার্সন কন্ট্রোল সিস্টেম (VCS) যা মূলত সফটওয়্যার ডেভেলপমেন্টে ব্যবহৃত হয়। এর মাধ্যমে কোড বা ডকুমেন্টের পরিবর্তনগুলোর ইতিহাস রেকর্ড করা হয় এবং একাধিক ডেভেলপার একসাথে কাজ করতে পারে। সহজ ভাষায় বললে, Git হলো একটি টুল যা তোমাকে এবং তোমার দলকে কাজের প্রতিটি পর্যায় ট্র্যাক করতে সাহায্য করে, যাতে যে কোন সময় আগের কাজগুলো ফিরিয়ে আনা যায় বা বুবাতে পারা যায় কী কী পরিবর্তন হয়েছে।

Git এর কিছু মূল ফিচার:

1. **Version Control:** এটি সব পরিবর্তন ট্র্যাক করে, অর্থাৎ তুমি কোন সময় কী পরিবর্তন করেছিলে তার রেকর্ড রাখা হয়। সে সকল পরিবর্তন দেখায়।
2. **Collaboration:** একাধিক ডেভেলপার একসাথে কাজ করতে পারে। সবাই নিজের নিজের কম্পিউটারে কাজ করতে পারে এবং পরে সেই কাজগুলো একত্রিত করা যায়।
3. **History:** Git এর মাধ্যমে অতীতের কাজের ইতিহাস দেখা যায়, যেন প্রয়োজনে আগের কোডে ফিরে যাওয়া যায়।
4. **Branching:** তুমি কোডে আলাদা আলাদা অংশে কাজ করতে পারো (যেমন নতুন ফিচার বানানো), এবং পরে সেগুলো একত্রিত করতে পারো। এতে প্রধান কোডের কোনো সমস্যা হওয়ার ঝুঁকি কমে যায়।

উদাহরণ:

ধৰা যাক, তুমি একটি ওয়েবসাইটের জন্য কোড লিখছো। প্রথমে তুমি index.html ফাইল বানাও। পরে তুমি একটি নতুন ফিচার যোগ করতে গিয়ে ভুল করে কিছু কোড মুছে ফেললে, তুমি Git ব্যবহার করে আগের সংস্করণ ফিরিয়ে আনতে পারো, যাতে তুমির কাজ হারিয়ে না যায়।

Git একটি শক্তিশালী টুল যা কোডের ইতিহাস এবং উন্নয়ন প্রক্রিয়া সহজে পরিচালনা করতে সাহায্য করে।

Git এবং GitHub এর মধ্যে পার্থক্য বুঝতে চাইলে, এটা একদম সহজভাবে বলতে পারি:

Git হলো এক ধরনের ভার্সন কন্ট্রোল সিস্টেম (VCS), যা কোডের পরিবর্তন ট্র্যাক করে এবং তোমার কম্পিউটারে কাজ করে। এটি এমন একটি টুল, যা তোমাকে কোডের ইতিহাস সংরক্ষণ করতে এবং কোডের বিভিন্ন সংস্করণ তৈরি করতে সাহায্য করে। ভাবো, Git হলো সাবান—যেটি তোমার নিজের কাছে থাকে এবং কোডের "পরিচ্ছন্নতা" বা ইতিহাস রাখতে কাজ করে।

অপরদিকে, GitHub হলো একটি অনলাইন প্ল্যাটফর্ম যেখানে Git রেপোজিটরিগুলো হোস্ট করা হয় এবং যেখানে ডেভেলপাররা একে অপরের সাথে কোড শেয়ার করতে পারে। এটি Git-এর মতো কাজ করলেও, একে ব্যবহার করা হয় মূলত অনলাইনে সহযোগিতা ও কোড শেয়ারিংয়ের জন্য। তুমি যেমন Lux সাবান কিনে ব্যবহার করো, তেমনি GitHub হলো Git এর সবচেয়ে জনপ্রিয় "ব্র্যান্ড" যেখানে অনেক ডেভেলপার তাদের কোডের রেপোজিটরি রাখে এবং একে অপরের সাথে সহযোগিতা করে।

তবে, Git এর আরও অনেক ব্র্যান্ড বা ভিন্ন ভিন্ন ভার্সন কন্ট্রোল সিস্টেম রয়েছে, যেমন Bitbucket, GitLab ইত্যাদি, যেগুলিও Git-এর মতো কাজ করে, কিন্তু GitHub-এর মতো জনপ্রিয় নয়।

সংক্ষেপে:

- ✚ Git হলো কোডের ইতিহাস ট্র্যাক করার টুল, যা তোমার কম্পিউটারে কাজ করে।
- ✚ GitHub হলো Git-এর উপর ভিত্তি করে তৈরি একটি অনলাইন প্ল্যাটফর্ম, যেখানে কোড শেয়ার এবং সহযোগিতা করা হয়।
- ✚ Git হচ্ছে "সাবান", এবং GitHub হলো সেই সাবানের "Lux ব্র্যান্ড" যা সবচেয়ে বেশি জনপ্রিয় এবং ব্যবহৃত।

2. Installing and configuring Git.

প্রথমে Git ডাউনলোড ও ইনস্টল করার জন্য নিচের ধাপগুলো অনুসরণ করো:

1. Git ডাউনলোড:

- ✚ আপনার কম্পিউটারের অপারেটিং সিস্টেম অনুযায়ী Git ডাউনলোড করতে হবে।
- ✚ নিচের লিঙ্কে যান: <https://git-scm.com/downloads>
- ✚ লিঙ্কে প্রবেশ করার পর, স্বয়ংক্রিয়ভাবে আপনার অপারেটিং সিস্টেমের জন্য সঠিক Git ভার্সন নির্বাচন করা হবে।
- ✚ যদি না হয়, তবে Windows, macOS বা Linux এর জন্য আলাদা ডাউনলোড অপশন দেখতে পারবেন।

2. Git ইনস্টলেশন:

- ⊕ ডাউনলোড করা Git ফাইলটি ওপেন করুন।
- ⊕ ইনস্টলেশনের জন্য নির্দেশনা অনুসরণ করুন। সাধারণত, এটি কয়েকটি সহজ ধাপে সম্পন্ন হয়:
 - License Agreement এর শর্তাবলী মেনে নিন।
 - ইনস্টলেশন ডিরেক্টরি নির্বাচন করুন (যতটুকু সম্ভব, ডিফল্ট রেখে দিন)।
 - কয়েকটি অপশন নির্বাচন করতে বলা হবে (যেমনঃ PATH environment variable সেট করা, অথবা Git Bash ও Git GUI ইনস্টল করা)। এগুলোতে ডিফল্ট অপশন রেখে এগিয়ে যান।
 - অবশ্যে ইনস্টলেশন প্রক্রিয়া সম্পূর্ণ হলে Finish বাটনে ক্লিক করুন।

3. Git ইনস্টল হয়েছে কি না চেক করুন:

- ⊕ ইনস্টলেশন শেষে, Command Prompt (Windows), Terminal (macOS/Linux) খুলুন।
- ⊕ টাইপ করুন:
- ⊕ git --version
- ⊕ এটি Git এর ইনস্টল হওয়া ভাস্বন দেখাবে, যেমন git version 2.47.1.windows.1

এখন আপনি Git ব্যবহার করার জন্য প্রস্তুত!

3. Creating a GitHub account.

GitHub-এ একাউন্ট তৈরি করার জন্য নিচের ধাপগুলো অনুসরণ করুন:

1. GitHub ওয়েবসাইটে প্রবেশ করুন:

- প্রথমে এই লিঙ্কে যান: <https://github.com/>

2. সাইন আপ পেজে যান:

- GitHub এর হোমপেজে গেলে, তান পাশে "Sign up" বাটনটি দেখতে পাবেন। সেটিতে ক্লিক করুন।

3. ব্যক্তিগত তথ্য পূরণ করুন:

- Username: আপনার পছন্দমত একটি ইউনিক ইউজারনেম দিন (যা আপনার GitHub অ্যাকাউন্টের পরিচিতি হবে)।
- Email Address: একটি বৈধ ইমেইল ঠিকানা প্রদান করুন।
- Password: একটি শক্তিশালী পাসওয়ার্ড দিন, যা কমপক্ষে 8টি অক্ষরের হতে হবে এবং ছোট-বড় অক্ষর, সংখ্যা ও বিশেষ চিহ্ন অন্তর্ভুক্ত থাকতে হবে।

4. Verification:

- Email verification: GitHub আপনার দেওয়া ইমেইলে একটি ভেরিফিকেশন লিংক পাঠাবে। আপনার ইমেইল ইনবক্সে গিয়ে সেই লিঙ্কে ক্লিক করে আপনার ইমেইল ঠিকানা নিশ্চিত করুন।

5. ধাপ ২ (নির্বাচন):

- পরবর্তী ধাপে, GitHub আপনার পছন্দের এক্সট্রা সেটিংস এবং সার্ভিস সম্পর্কিত কিছু প্রশ্ন করবে। আপনি এগুলো ডিফল্ট রেখে বা প্রয়োজনীয় অনুসারে নির্বাচন করতে পারেন।

6. Account Creation সম্পূর্ণ করুন:

- সব তথ্য সঠিকভাবে পূর্ণ করার পর, "Create Account" বাটনে ক্লিক করুন। এখন আপনার GitHub অ্যাকাউন্ট তৈরি হয়ে যাবে।

7. অ্যাকাউন্ট সাইন ইন করুন:

- সাইন আপ প্রক্রিয়া সম্পূর্ণ হওয়ার পর, আপনার GitHub অ্যাকাউন্টে সাইন ইন করতে পারবেন। এর জন্য "Sign in" বাটনে ক্লিক করে আপনার ইমেইল ঠিকানা এবং পাসওয়ার্ড দিয়ে লগইন করুন।

এখন আপনি GitHub ব্যবহার করার জন্য প্রস্তুত!

4. Creating repositories and deploying on GitHub.

GitHub-এ Repository তৈরি করার প্রক্রিয়া:

GitHub-এ একটি repository তৈরি করা মানে হল একটি নতুন প্রোজেক্ট বা ফাইল সংগ্রহের জায়গা তৈরি করা যেখানে আপনি কোড সংরক্ষণ, শেয়ার এবং পরিচালনা করতে পারবেন। এখানে কিভাবে আপনি GitHub-এ repository তৈরি করবেন তার বিস্তারিত ব্যাখ্যা দেওয়া হলো:

১. GitHub-এ লগইন করুন

প্রথমেই আপনার GitHub অ্যাকাউন্টে লগইন করতে হবে:

- [GitHub.com](https://github.com) ওয়েবসাইটে যান।
- আপনার ইউজারনেম এবং পাসওয়ার্ড দিয়ে লগইন করুন।

২. নতুন Repository তৈরি করুন

লগইন করার পর, নতুন repository তৈরি করতে এই ধাপগুলো অনুসরণ করুন:

1. Dashboard-এ যান:

- GitHub-এ লগইন করার পর, উপরের ডান দিকের কোণায় আপনার প্রোফাইল ছবির পাশে থাকা + চিহ্নে ক্লিক করুন এবং তারপর "New repository" নির্বাচন করুন। <https://github.com/new>

2. Repository এর নাম দিন:

- Repository name: এখানে আপনার প্রোজেক্ট বা কোডের জন্য একটি নাম দিন। এটি একে একে ইউনিক (অন্যান্য রেপোজিটরির সঙ্গে মিলবে না) হতে হবে। যেমন: my-first-project।

3. Description (অপশনাল):

- আপনি চাইলে আপনার repository এর বর্ণনা দিতে পারেন, যেমন "My first project on GitHub" বা "This repository contains my website code"।

4. Public/Private নির্বাচন করুন:

- Public: এই repository সবার জন্য উন্মুক্ত হবে, সবাই এটি দেখতে এবং কপি (fork) করতে পারবে।
- Private: শুধুমাত্র আপনি এবং আপনার নির্বাচিত ব্যবহারকারীরা এটি দেখতে পারবেন।

5. Initialize this repository with a README:

- আপনি যদি চাইলে README ফাইল দিয়ে repository শুরু করতে পারেন। README ফাইলটি সাধারণত প্রজেক্টের বর্ণনা থাকে এবং এটি GitHub-এর repository পৃষ্ঠায় দেখানো হয়।
- এই অপশনটি নির্বাচন করলে GitHub স্বয়ংক্রিয়ভাবে একটি README.md ফাইল তৈরি করবে।

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * **Repository name ***

mminhzulkabir / my-first-project my-first-project is available.

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-journey](#) ?

Description (optional)

Initialize this repository with:

Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

ⓘ You are creating a public repository in your personal account.

Create repository

Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

© 2025 GitHub, Inc.

6. Add .gitignore (অপশনাল):

- .gitignore ফাইলের মাধ্যমে আপনি বলে দিতে পারেন যে কোন ফাইলগুলি GitHub এ আপলোড হবে না। উদাহরণস্বরূপ, যদি আপনি কোনো প্রোগ্রামিং ভাষায় কাজ করেন এবং কোনো নির্দিষ্ট ফাইল (যেমনঃ .log, .tmp ইত্যাদি) চান না GitHub-এ আপলোড হোক, তবে সেগুলো .gitignore ফাইলে উল্লেখ করা যাবে।

7. Choose a license (অপশনাল):

- আপনি যদি চান, তবে repository-এর জন্য একটি লাইসেন্স নির্বাচন করতে পারেন। লাইসেন্সের মাধ্যমে আপনি নির্ধারণ করবেন, অন্যরা আপনার কোড কীভাবে ব্যবহার করতে পারবে।

- এটি একেবারে অপশনাল, তবে যেকোনো ওপেন সোর্স প্রজেক্টের জন্য লাইসেন্স থাকা গুরুত্বপূর্ণ।

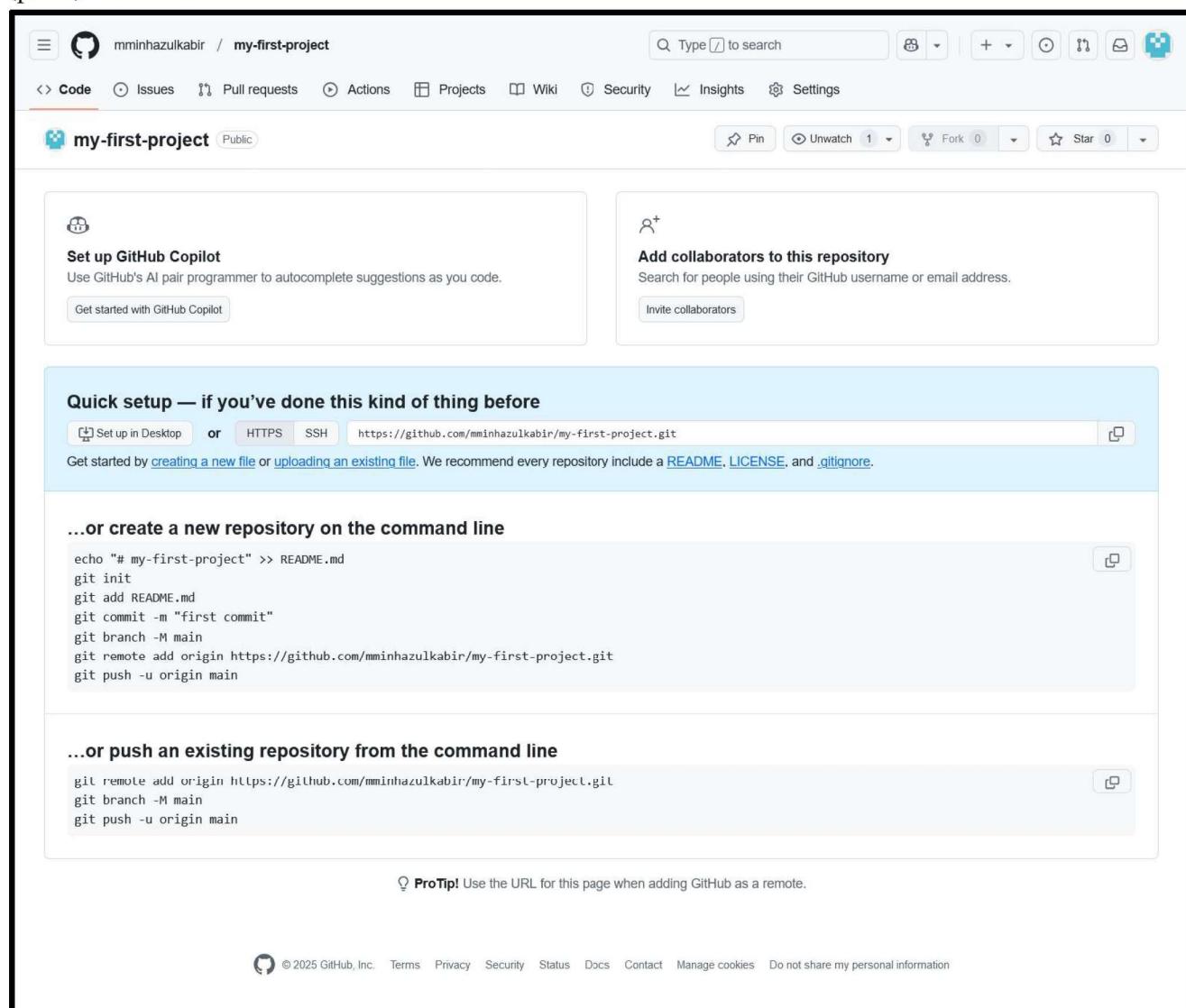
৩. Create Repository বাটনে ক্লিক করুন

সব তথ্য পূর্ণ করার পর, নিচে "Create repository" বাটনে ক্লিক করুন। এটি আপনার নতুন repository তৈরি করবে।

৪. Repository ব্যবহার শুরু করুন

একবার repository তৈরি হয়ে গেলে, আপনি এখন কোড সংরক্ষণ, ফাইল আপলোড, এবং অন্যান্য কার্যক্রম শুরু করতে পারবেন। আপনার repository-র পৃষ্ঠায় আপনি কোড লিখতে পারবেন, এবং আপনার সহকর্মীদের বা কমিউনিটির সাথে এটি শেয়ার করতে পারবেন।

এছাড়া, আপনি GitHub Desktop অথবা Git CLI (Command Line Interface) ব্যবহার করে আপনার কম্পিউটারের লোকাল ফোল্ডারে কাজ করতে পারেন এবং তারপর পরিবর্তনগুলো GitHub repository তে পুশ (push) করতে পারবেন।



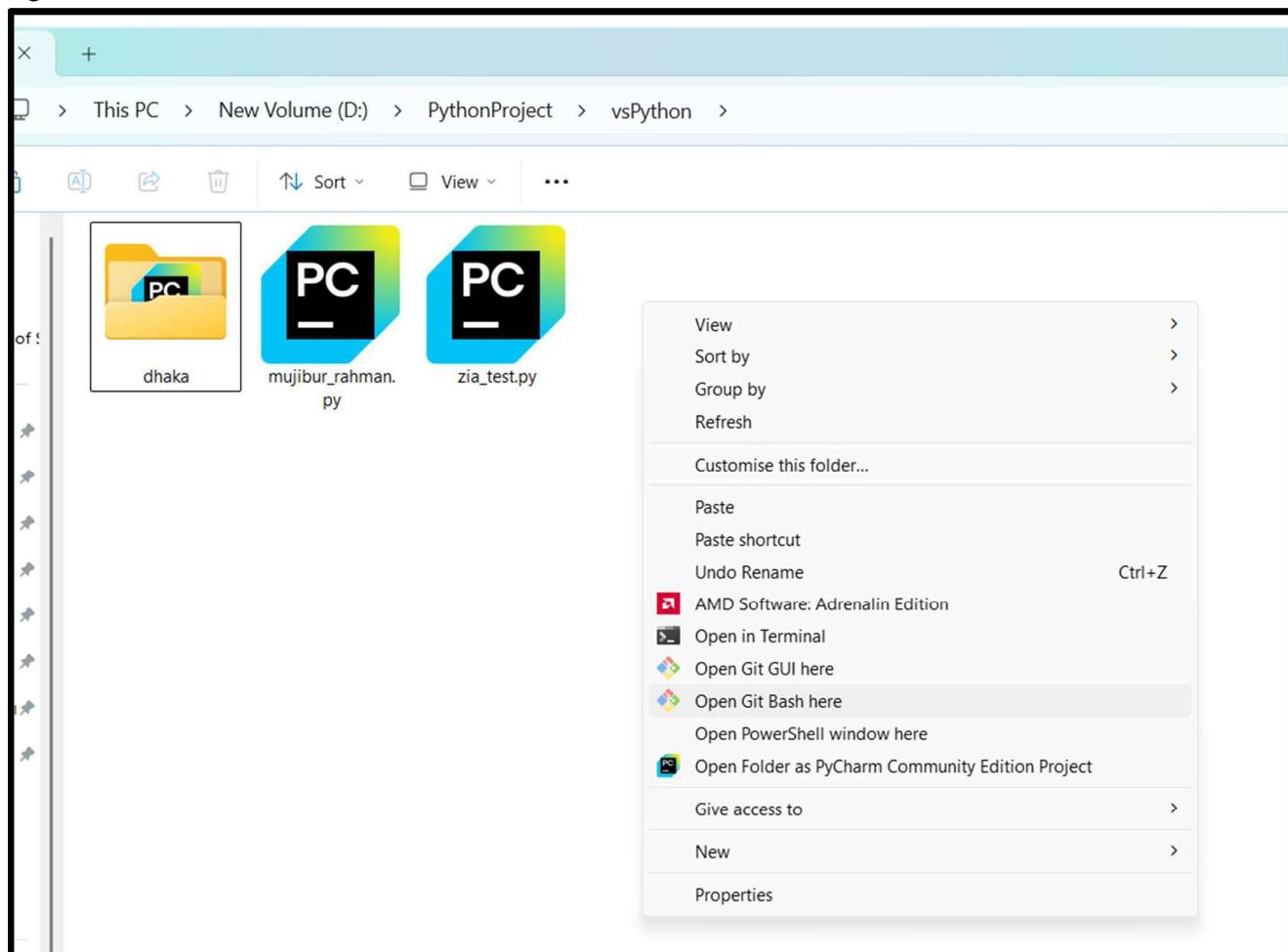
উপরে লিখা পর্যন্ত করলে পারলে আমাকে জানাও যে তুমি সেটা করতে পেরেছ।

সারাংশ:

- GitHub-এ Repository তৈরি করার প্রক্রিয়া হল একটি নতুন জায়গা তৈরি করা, যেখানে আপনি আপনার কোড বা প্রজেক্ট সংরক্ষণ করতে পারবেন এবং অন্যান্যদের সঙ্গে শেয়ার করতে পারবেন।
- এই repository তৈরি করার সময় নাম, বর্ণনা, পাবলিক বা প্রাইভেট নির্বাচন, README ফাইল ইত্যাদি নির্ধারণ করা হয়।

এভাবে আপনি GitHub-এ একটি repository তৈরি করতে পারেন এবং আপনার কোড বা প্রজেক্টের জন্য একটি নতুন স্থান পেতে পারেন।

এখন, একটা ফোল্ডারে এর মধ্য কিছু পাইথন ফাইল রাখো। এবং, কি-বোর্ড এর shift button চাপ দিয়ে মাউসের right এ ক্লিক করো। এমন দেখতে পাবে।



Open Git Bash here এ আমরা নির্বাচন করবো।

GitHub Folder/Repository-তে কোড পুশ/upload করা (Deploy)

1. লোকাল ফোল্ডারে Git Folder/Repository ইনিশিয়ালাইজ করুন: আপনার কম্পিউটারে যে প্রজেক্ট বা ফাইলগুলো আপনি GitHub-এ আপলোড করতে চান, সেই ফোল্ডারে যান। যদি Git repository না থাকে, তবে git init কমান্ড দিয়ে একটি নতুন Git folder/repository শুরু(initial) করুন।

তুমি নিচের লিখাটা লিখবে

```
git init
```

এই কমান্ডটি চালানোর পর আপনি নিচের বার্তাটি পাবেন:

```
Initialized empty Git repository in D:/PythonProject/vsPython/.git/
```

2. ফাইলগুলো Git-এ যোগ করুন: আপনার সমস্ত ফাইল যোগ করতে, git add কমান্ড ব্যবহার করুন।
যে ফাইলগুলি আপনি GitHub-এ আপলোড করতে চান, সেগুলি আপনার লোকাল রিপোজিটরিতে/ফোল্ডারে যুক্ত/add
করতে হবে। তুমি নিচের লিখাটা লিখবে।

```
git add .
```

এই কমান্ডটি চালানোর পর, git সে ফোল্ডারের অধীনে সকল file এবং folder তার নজরদারিতে রাখা শুরু করবে।
এই কমান্ডটি সমস্ত পরিবর্তন এবং নতুন ফাইলকে Git-এ যোগ করবে।

3. Commit করুন: পরিবর্তনগুলো কমিট করতে, git commit কমান্ড ব্যবহার করুন। কমিট মেসেজ দিন,
যেমন: তুমি নিচের লিখাটা লিখবে

```
git commit -m "Initial commit minhaz"
```

এই কমান্ডটি চালানোর পর আপনি নিচের বার্তাটি পাবেন:

```
$ git commit -m "Initial commit minhaz"
[master (root-commit) aa7c499] Initial commit minhaz
 4 files changed, 41 insertions(+)
  create mode 100644 dhaka/cantonment.py
  create mode 100644 dhaka/mirpur.py
  create mode 100644 mujibur_rahman.py
  create mode 100644 zia_test.py
```

4. আপনার রিপোজিটরির ডিফল্ট ব্রাঞ্চের নাম master। আপনি যদি এটি main এ পরিবর্তন করতে চান,
তাহলে নিচের কমান্ডটি ব্যবহার করুন:

```
git branch -M main
```

এই কমান্ডটি গিট রিপোজিটরির ব্রাঞ্চ নাম পরিবর্তন করতে ব্যবহৃত হয়। এখানে, git branch ব্রাঞ্চ তৈরি বা ম্যানেজ
করতে ব্যবহার হয়, -M অপশনটি গিটকে ব্রাঞ্চের নাম পরিবর্তন করতে বলে, এবং main হল নতুন নাম, যা এখন
গিট রিপোজিটরির ডিফল্ট ব্রাঞ্চ হবে (যেমন আগে master ছিল)।

5. GitHub Folder/Repository-কে রিমোট হিসেবে যুক্ত করুন: যদি আপনি নিজের ল্যাপটপ/কম্পিউটারে
GitHub-এ তৈরি করা রিপোজিটরিকে রিমোট হিসেবে যোগ করতে চান, তাহলে নিচের কমান্ডটি ব্যবহার
করুন:

```
git remote add origin https://github.com/mminhazulkabir/my-first-project.git
```

এই কমান্ডটি দিয়ে আপনি নির্দিষ্ট GitHub রিপোজিটরির লিংককে রিমোট হিসেবে যুক্ত করবেন। remote এর মাধ্যমে নির্ধারণ করা হয়, কোন লিংকে আপনার প্রকল্প আপলোড করা হবে।

- ফাইল internet GitHub-এ পুশ/upload করুন: এখন, আপনার ফাইলগুলো GitHub-এ পুশ (আপলোড) করতে, নিচের কমান্ডটি ব্যবহার করুন:

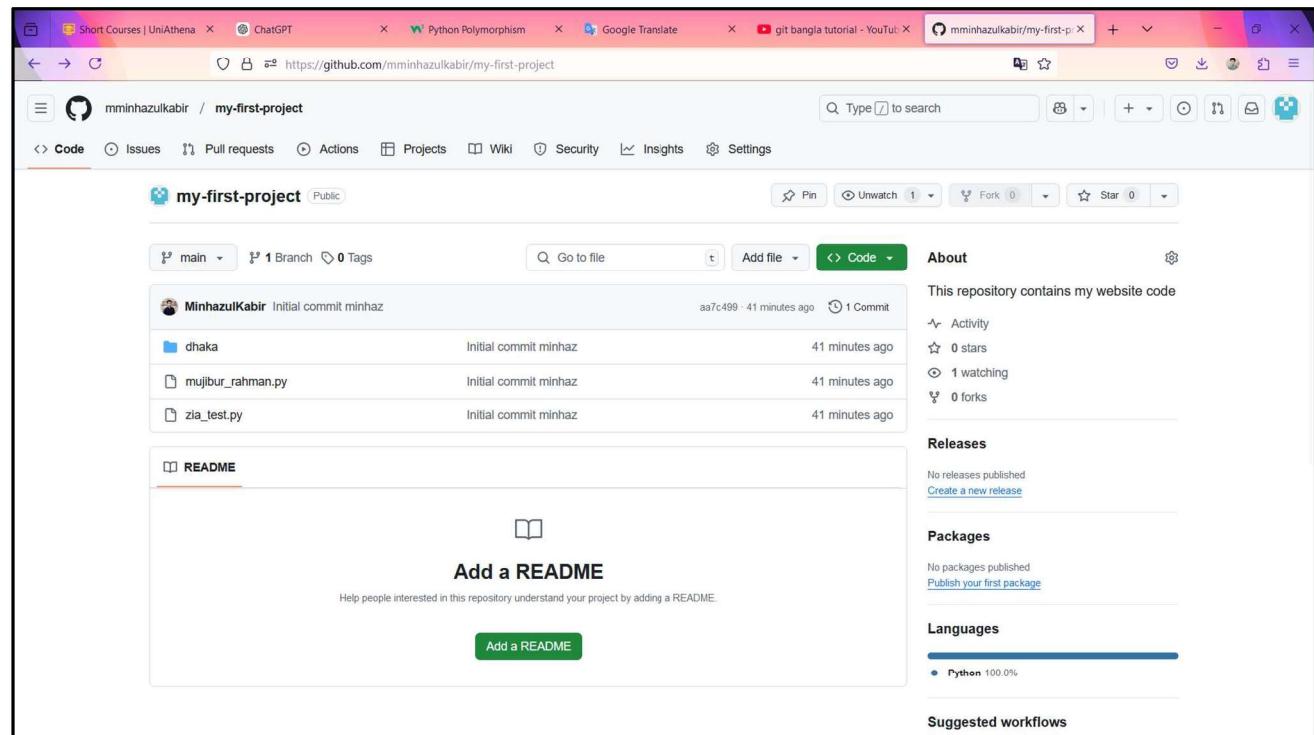
যদি আপনার ব্রাউজের নাম main হয় (যেটি নতুন GitHub রেপোজিটরি দ্বারা স্বয়ংক্রিয়ভাবে সেট করা হয়), তবে:

```
git push -u origin main
```

এই কমান্ডটি চালানোর পর আপনি নিচের বার্তাটি পাবেন:

```
$ git push -u origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.08 KiB | 1.08 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/mminhazulkabir/my-first-project.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Browser এ গেলে এটা দেখতে পাবে।



All of git command

```

MINGW64:/d/PythonProject/vsPython
Minhazul Kabir@DESKTOP-ESEK66M MINGW64 /d/PythonProject/vsPython
$ git init
init      instaweb

Minhazul Kabir@DESKTOP-ESEK66M MINGW64 /d/PythonProject/vsPython
$ git init
Initialized empty Git repository in D:/PythonProject/vsPython/.git/

Minhazul Kabir@DESKTOP-ESEK66M MINGW64 /d/PythonProject/vsPython (master)
$ git add .

Minhazul Kabir@DESKTOP-ESEK66M MINGW64 /d/PythonProject/vsPython (master)
$ git commit -m "Initial commit minhaz"
[master (root-commit) aa7c499] Initial commit minhaz
 4 files changed, 41 insertions(+)
 create mode 100644 dhaka/cantonment.py
 create mode 100644 dhaka/mirpur.py
 create mode 100644 mujibur_rahman.py
 create mode 100644 zia_test.py

Minhazul Kabir@DESKTOP-ESEK66M MINGW64 /d/PythonProject/vsPython (master)
$ git remote add origin https://github.com/mminhazulkabir/my-first-project.git

Minhazul Kabir@DESKTOP-ESEK66M MINGW64 /d/PythonProject/vsPython (master)
$ git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/mminhazulkabir/my-first-project.git'

Minhazul Kabir@DESKTOP-ESEK66M MINGW64 /d/PythonProject/vsPython (master)
$ git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/mminhazulkabir/my-first-project.git'

Minhazul Kabir@DESKTOP-ESEK66M MINGW64 /d/PythonProject/vsPython (master)
$ git branch -M main

Minhazul Kabir@DESKTOP-ESEK66M MINGW64 /d/PythonProject/vsPython (main)
$ git remote add origin https://github.com/mminhazulkabir/my-first-project.git
error: remote origin already exists.

Minhazul Kabir@DESKTOP-ESEK66M MINGW64 /d/PythonProject/vsPython (main)
$ git push -u origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.08 KiB | 1.08 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/mminhazulkabir/my-first-project.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

Minhazul Kabir@DESKTOP-ESEK66M MINGW64 /d/PythonProject/vsPython (main)
$ |

```

5. Basic bash commands.

নিচে Git ব্যবহার করার জন্য কিছু বেসিক Bash কমান্ড দেওয়া হলো:

1. **git init:** একটি নতুন Git রিপোজিটরি তৈরি করতে ব্যবহৃত হয়।

উদাহরণ: `git init`

2. **git clone:** একটি বিদ্যমান Git রিপোজিটরি ক্লোন (কপি) করতে ব্যবহৃত হয়।

উদাহরণ: `git clone https://github.com/username/repository.git`

3. git status: আপনার রিপোজিটরির বর্তমান অবস্থা দেখতে ব্যবহৃত হয় (যেমন, কোন ফাইল পরিবর্তিত হয়েছে বা যোগ করা হয়েছে)।

উদাহরণ: git status

4. git add: ফাইল বা ডিরেক্টরি গিট স্টেজিং এরিয়াতে যোগ করতে ব্যবহৃত হয় (কমিটের জন্য প্রস্তুত করা)।

উদাহরণ: git add filename.txt

সমস্ত পরিবর্তন যোগ করতে: git add .

5. git commit: স্টেজ করা পরিবর্তন গিট রিপোজিটরিতে কমিট (সংরক্ষণ) করতে ব্যবহৃত হয়।

উদাহরণ: git commit -m "Commit message"

6. git push: লোকাল রিপোজিটরি থেকে পরিবর্তনগুলো রিমোট রিপোজিটরিতে পুশ (আপলোড) করতে ব্যবহৃত হয়।

উদাহরণ: git push origin main

7. git pull: রিমোট রিপোজিটরি থেকে পরিবর্তনগুলো লোকাল রিপোজিটরিতে আনতে ব্যবহৃত হয়।

উদাহরণ: git pull origin main

8. git branch: গিট রিপোজিটরির ব্রাঞ্চ দেখানোর জন্য ব্যবহৃত হয় বা নতুন ব্রাঞ্চ তৈরি করতে ব্যবহৃত হয়।

ব্রাঞ্চ দেখাতে: git branch

নতুন ব্রাঞ্চ তৈরি করতে: git branch new-branch

9. git checkout: একটি নির্দিষ্ট ব্রাঞ্চে স্যুইচ করতে ব্যবহৃত হয়।

উদাহরণ: git checkout new-branch

10. git merge: একটি ব্রাঞ্চের পরিবর্তন অন্য একটি ব্রাঞ্চে মার্জ করতে ব্যবহৃত হয়।

উদাহরণ: git merge new-branch

11. git log: রিপোজিটরির ইতিহাস (কমিট ইতিহাস) দেখতে ব্যবহৃত হয়।

উদাহরণ: git log

12. git remote: রিমোট রিপোজিটরি সম্পর্কিত তথ্য দেখতে বা সেটিংস পরিবর্তন করতে ব্যবহৃত হয়।

রিমোট রিপোজিটরি দেখতে: git remote -v

রিমোট রিপোজিটরি যোগ করতে: git remote add origin <https://github.com/username/repository.git>

13. git fetch: রিমোট রিপোজিটরি থেকে তথ্য নিয়ে আসে কিন্তু লোকাল ব্রাঞ্চে মার্জ করে না।

উদাহরণ: git fetch origin

14. git reset: কমিটগুলিকে রিভার্স (অথবা আনস্টেজ) করতে ব্যবহৃত হয়।

উদাহরণ: git reset --hard HEAD

15. git diff: ফাইলের মধ্যে পরিবর্তনগুলি দেখতে ব্যবহৃত হয়।

উদাহরণ: git diff

16. git rm: রিপোজিটরি থেকে ফাইল মুছে ফেলতে ব্যবহৃত হয়।

উদাহরণ: git rm filename.txt

17. git tag: একটি নির্দিষ্ট পয়েন্টে ট্যাগ তৈরি করতে ব্যবহৃত হয় (যেমন, রিলিজ ভের্শন)।

উদাহরণ: git tag v1.0

18. git stash: সাময়িক পরিবর্তনগুলি সেভ করে রাখতে ব্যবহৃত হয়, যাতে কাজের মধ্যে ব্রাঞ্চ সুইচ করা যায়।

উদাহরণ: git stash

19. git show: কোন কমিটের বিস্তারিত দেখতে ব্যবহৃত হয়।

উদাহরণ: git show <commit-id>

এগুলি Git ব্যবহারের জন্য কিছু বেসিক Bash কমান্ড। এগুলি আপনার গিট রিপোজিটরির কার্যক্রম সহজে পরিচালনা করতে সহায়তা করবে।

নিচে git এর main এবং master ব্রাঞ্চের মধ্যে পার্থক্যটি উপস্থাপন করা হলো:

বিষয়	main ব্রাঞ্চ	master ব্রাঞ্চ
ডিফল্ট নাম	বর্তমানে অধিকাংশ Git রিপোজিটরিতে ব্যবহৃত।	পুরনো Git রিপোজিটরির ডিফল্ট নাম।
ইতিহাস	২০২০ সালে GitHub এবং অন্যান্য প্ল্যাটফর্মে পরিবর্তন শুরু হয়।	Git এর প্রথম সংস্করণে ছিল ডিফল্ট নাম।
নেতিবাচক অর্থ	কোনো নেতিবাচক বা অপমানজনক কনটেক্টের নেই।	"master" শব্দটি ইতিহাসে কিছু নেতিবাচক কনটেক্টের সাথে যুক্ত ছিল।
সামাজিক প্রভাব	বৈচিত্র্য এবং অন্তর্ভুক্তির লক্ষ্যে পরিবর্তন করা হয়েছে।	এখনো কিছু প্রজেক্টে ব্যবহৃত হয়, তবে পরিবর্তনের উদ্যোগ চলমান।
ব্যবহার	অধিকাংশ আধুনিক প্রজেক্টে ব্যবহৃত হয়।	পুরনো বা ঐতিহ্যবাহী প্রজেক্টে এখনও ব্যবহৃত হয়।