**1. Create a new GitHub repository**

1. Go to GitHub, click **New → Repository**, give it a name (e.g. flask‑project), optionally add a README, and click **Create**.
2. Copy the repo’s **SSH** URL, which looks like:

scss

CopyEdit

git@github.com:YOUR-USERNAME/YOUR-REPO.git

You’ll use this to clone over SSH (instead of HTTPS).

**2. Generate an SSH key (if you don't already have one)**

**On macOS/Linux:**

1. Open Terminal/Bash and check for existing keys:

bash

CopyEdit

ls -al ~/.ssh

1. If you don’t have id\_ed25519 or id\_rsa, generate a new key with:

bash

CopyEdit

ssh-keygen -t ed25519 -C "your\_email@example.com"

(or use -t rsa -b 4096 -C "email" as a fallback) [GitHub Docs+1Popupsmart Community+1](https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository?utm_source=chatgpt.com)[phoenixNAP | Global IT Services+1TheServerSide+1](https://phoenixnap.com/kb/git-clone-ssh?utm_source=chatgpt.com)[Medium+5GitHub Docs+5YouTube+5](https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent?utm_source=chatgpt.com)[Stack Overflow](https://stackoverflow.com/questions/41714882/git-how-to-clone-with-ssh-key-username?utm_source=chatgpt.com)

1. When prompted, accept the default file location and optionally set a passphrase.

**3. Add your key to the SSH agent**

1. Start the agent:

bash

CopyEdit

eval "$(ssh-agent -s)"

1. Add your private key:

bash

CopyEdit

ssh-add ~/.ssh/id\_ed25519

(Replace with ~/.ssh/id\_rsa if using RSA key) [GitHub Docs+1Medium+1](https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent?utm_source=chatgpt.com)[phoenixNAP | Global IT Services+1GitHub Docs+1](https://phoenixnap.com/kb/git-clone-ssh?utm_source=chatgpt.com)[Stack Overflow](https://stackoverflow.com/questions/41714882/git-how-to-clone-with-ssh-key-username?utm_source=chatgpt.com)

**4. Add your public key to GitHub**

1. Copy the public key:

bash

CopyEdit

pbcopy < ~/.ssh/id\_ed25519.pub # macOS

# or

cat ~/.ssh/id\_ed25519.pub | clip # Windows/Git Bash

1. In GitHub: Profile → **Settings → SSH and GPG keys → New SSH key**, give it a title, paste the key, and save [TheServerSide+4GitHub Docs+4Stack Overflow+4](https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account?utm_source=chatgpt.com).

**5. Test your SSH connection**

Run:

bash

CopyEdit

ssh -T git@github.com

You should see something like:  
Hi YOUR-USERNAME! You've successfully authenticated...  
This confirms SSH is set up correctly. [Medium](https://medium.com/%40kyledeguzmanx/quick-step-by-step-guide-to-generating-an-ssh-key-in-github-d3c6f7e185bb?utm_source=chatgpt.com)[YouTube+7phoenixNAP | Global IT Services+7Stack Overflow+7](https://phoenixnap.com/kb/git-clone-ssh?utm_source=chatgpt.com)

**6. Clone the repo via SSH**

Choose a directory and run:

bash

CopyEdit

git clone git@github.com:YOUR-USERNAME/YOUR-REPO.git

cd YOUR-REPO

This uses your SSH key for secure access. [GitHub Docs](https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository?utm_source=chatgpt.com)

**7. Create a new branch**

Make a branch named after your username, e.g.:

bash

CopyEdit

git checkout -b Tutedude

This creates and checks out the Tutedude branch.

**8. Add your Flask project files**

Copy or create your Flask app files (e.g., app.py, requirements.txt, templates, etc.) into the project folder. Then add and commit them:

bash

CopyEdit

git add .

git commit -m "Add initial Flask project files"

**9. Merge your branch into main**

1. Switch to main:

bash

CopyEdit

git checkout main

1. Merge your branch:

bash

CopyEdit

git merge Tutedude

* + If there are no conflicts, your changes go into main.

1. Push the updated main branch back to GitHub:

bash

CopyEdit

git push origin main

**✅ Summary of what each step does:**

| **Step** | **Command** | **What it does** |
| --- | --- | --- |
| 1 | git clone ... | Downloads your GitHub repo to your computer via SSH |
| 2 | git checkout -b Tutedude | Creates a new branch for you to safely work on |
| 3 | git add ., git commit | Saves your Flask files into the branch |
| 4 | git checkout main | Returns you to the default branch |
| 5 | git merge Tutedude | Merges your work into the main branch |
| 6 | git push origin main | Uploads your updated main branch to GitHub |

**1. 🔀 Create the new branch**

bash

CopyEdit

git checkout main

git pull origin main

git checkout -b YOURNAME\_new

This creates and switches to the YOURNAME\_new branch—where you'll update the JSON content for your /api route.

**2. ✍️ Update the JSON file**

* Open the file (e.g., data.json) that your Flask /api route uses.
* Make your changes.
* Save the file.
* Stage and commit:

bash

CopyEdit

git add data.json

git commit -m "Update JSON content for /api in YOURNAME\_new branch"

**3. 🔄 Merge into main and resolve conflicts (if any)**

Switch to main:

bash

CopyEdit

git checkout main

git pull origin main

Merge your branch:

bash

CopyEdit

git merge YOURNAME\_new

**▶️ If there are no merge conflicts, simply:**

bash

CopyEdit

git push origin main

**❗ If merge conflicts occur:**

1. **Locate conflict markers** in data.json (you’ll see <<<<<<<, =======, >>>>>>>).
2. **Resolve** conflicts by choosing the version from YOURNAME\_new:

text

CopyEdit

<<<<<<< HEAD

old content

=======

new content from YOURNAME\_new

>>>>>>> YOURNAME\_new

Keep only the “new content” block.

1. **Stage the resolved file**:

bash

CopyEdit

git add data.json

1. **Commit the merge**:

bash

CopyEdit

git commit -m "Merge YOURNAME\_new into main, resolved conflicts using branch changes"

1. **Push to remote**:

bash

CopyEdit

git push origin main

**Why this works:**

When merging branches, Git integrates changes from both branches. If the same line differs, a conflict marker appears. You're explicitly choosing the branch version by resolving conflicts in favor of YOURNAME\_new, then staging and committing. This completes the merge [geeksforgeeks.org+5stackoverflow.com+5stackoverflow.com+5](https://stackoverflow.com/questions/9189867/after-a-git-merge-conflict-a-lot-of-files-i-didnt-touch-become-changes-to-be-c?utm_source=chatgpt.com)[stackoverflow.com+4fastercapital.com+4geeksforgeeks.org+4](https://fastercapital.com/topics/setting-up-your-development-environment-for-php.html?utm_source=chatgpt.com).

**📝 Summary**

* Created YOURNAME\_new branch
* Modified the JSON file and committed
* Merged into main
* Resolved any conflicts by taking the branch’s version
* Staged, committed the resolution, and pushed updates

**1. Branch Creation**

bash

CopyEdit

# Ensure you're on the main branch and it's up to date

git checkout main

git pull origin main

# Create the new branches from main

git checkout -b master\_1

git push -u origin master\_1

git checkout main

git checkout -b master\_2

git push -u origin master\_2

**✅ 2. Feature Development in master\_1 (Frontend)**

**Structure a To-Do Page (React Example)**

**src/pages/TodoPage.js**

jsx

CopyEdit

import React, { useState } from 'react';

import axios from 'axios';

const TodoPage = () => {

const [itemName, setItemName] = useState('');

const [itemDescription, setItemDescription] = useState('');

const handleSubmit = async (e) => {

e.preventDefault();

try {

await axios.post('/submittodoitem', { itemName, itemDescription });

alert('To-Do item submitted!');

setItemName('');

setItemDescription('');

} catch (error) {

console.error('Error submitting To-Do item:', error);

}

};

return (

<form onSubmit={handleSubmit}>

<div>

<label>Item Name:</label>

<input value={itemName} onChange={(e) => setItemName(e.target.value)} />

</div>

<div>

<label>Item Description:</label>

<textarea value={itemDescription} onChange={(e) => setItemDescription(e.target.value)} />

</div>

<button type="submit">Submit</button>

</form>

);

};

export default TodoPage;

**Push changes**

bash

CopyEdit

git add .

git commit -m "Add To-Do form page in frontend"

git push origin master\_1

**✅ 3. Backend API in master\_2**

**Setup a route /submittodoitem**

**routes/todo.js**

js

CopyEdit

const express = require('express');

const router = express.Router();

const TodoItem = require('../models/TodoItem'); // Mongoose model

router.post('/submittodoitem', async (req, res) => {

const { itemName, itemDescription } = req.body;

try {

const newItem = new TodoItem({ itemName, itemDescription });

await newItem.save();

res.status(200).json({ message: 'Item saved successfully' });

} catch (err) {

res.status(500).json({ error: 'Failed to save item' });

}

});

module.exports = router;

**models/TodoItem.js**

js

CopyEdit

const mongoose = require('mongoose');

const todoSchema = new mongoose.Schema({

itemName: { type: String, required: true },

itemDescription: { type: String, required: true }

});

module.exports = mongoose.model('TodoItem', todoSchema);

**In your server entry file (e.g., server.js)**

js

CopyEdit

const express = require('express');

const mongoose = require('mongoose');

const todoRoutes = require('./routes/todo');

const app = express();

app.use(express.json());

app.use('/', todoRoutes);

// Connect to MongoDB

mongoose.connect('mongodb://localhost:27017/todoapp', {

useNewUrlParser: true,

useUnifiedTopology: true,

}).then(() => console.log('MongoDB connected'))

.catch(err => console.error(err));

const PORT = 5000;

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

**Push backend changes**

bash

CopyEdit

git add .

git commit -m "Add /submittodoitem POST route to store items in MongoDB"

git push origin master\_2

**✅ 4. Merging Changes into main**

**Step 1: Merge master\_1 into main**

bash

CopyEdit

git checkout main

git pull origin main

git merge master\_1

git push origin main

**Step 2: Merge master\_2 into main**

bash

CopyEdit

git merge master\_2

git push origin main

**1. Switch to master\_1 and Enhance the To-Do Form**

bash

CopyEdit

git checkout master\_1

**🔹 First Commit: Add Item ID field**

Edit your form (TodoPage.js) to include **Item ID**:

jsx

CopyEdit

const [itemId, setItemId] = useState('');

...

<div>

<label>Item ID:</label>

<input value={itemId} onChange={(e) => setItemId(e.target.value)} />

</div>

bash

CopyEdit

git add .

git commit -m "Add Item ID field to To-Do form"

**🔹 Second Commit: Add Item UUID field**

Add another input to the form:

jsx

CopyEdit

const [itemUUID, setItemUUID] = useState('');

...

<div>

<label>Item UUID:</label>

<input value={itemUUID} onChange={(e) => setItemUUID(e.target.value)} />

</div>

bash

CopyEdit

git add .

git commit -m "Add Item UUID field to To-Do form"

**🔹 Third Commit: Add Item Hash field**

Add one more input:

jsx

CopyEdit

const [itemHash, setItemHash] = useState('');

...

<div>

<label>Item Hash:</label>

<input value={itemHash} onChange={(e) => setItemHash(e.target.value)} />

</div>

bash

CopyEdit

git add .

git commit -m "Add Item Hash field to To-Do form"

**✅ 2. Merge master\_1 into main**

bash

CopyEdit

git checkout main

git pull origin main

git merge master\_1

git push origin main

**✅ 3. Git Reset to Roll Back to "Item ID" Only**

bash

CopyEdit

# Check history

git log --oneline

Look for the commit message:  
➡️ **"Add Item ID field to To-Do form"**

Then:

bash

CopyEdit

# Replace <commit\_hash> with the hash for the "Add Item ID field..." commit

git reset --soft <commit\_hash>

git status # Changes should be staged

git commit -m "Restore state with only Item ID field"

git push --force origin main

**✅ 4. Rebase master\_1 onto Updated main (Preserving Commit History)**

bash

CopyEdit

git checkout master\_1

git fetch origin

git rebase main

**You may see this in terminal:**

vbnet

CopyEdit

First, rewinding head to replay your work on top of it...

Applying: Add Item UUID field to To-Do form

Applying: Add Item Hash field to To-Do form

✅ Each commit is applied **individually**, preserving commit history. Do **not squash**.

If no conflicts, you’re done!

**✅ Summary of Git Flow**

| **Branch** | **Action** |
| --- | --- |
| master\_1 | Added 3 fields in 3 commits |
| main | Merged, then reset to keep only "Item ID" |
| master\_1 | Rebases main to stay up-to-date |
| Commit log | Preserves 3 logical changes as individual commits |