**Setting Up Git and GitHub for Your Django Project**

**1. Modify .gitignore to Avoid Unwanted Files**

**Why?**

**Git tracks all files by default, but some files should not be committed to GitHub, such as:**

* **Virtual environments (env/) – These contain dependencies but can be recreated using requirements.txt.**
* **Database files (\*.sqlite3) – These are generated during Django development and shouldn’t be stored in Git.**
* **Compiled Python files (\_\_pycache\_\_/ and \*.pyc) – Python automatically generates these, so we don't need them in version control.**
* **Log files (logs/) – These may contain sensitive debugging data.**
* **Media files (media/) – User-uploaded files (e.g., recordings) should not be committed.**
* **History files (.history/) – These may be auto-generated by VS Code or other tools.**

**How?**

**Edit your .gitignore file (or create one if it doesn’t exist) and add the following lines:**

**# Ignore environment files**

**env/**

**\*.env**

**# Ignore database files**

**\*.sqlite3**

**# Ignore compiled Python files**

**\_\_pycache\_\_/**

**\*.pyc**

**# Ignore logs**

**logs/**

**# Ignore media/uploads (recordings)**

**media/**

**# Ignore history files**

**.history/**

**2. Remove Previously Tracked Unwanted Files**

**Even though we added entries to .gitignore, Git has already been tracking some of these files. We need to remove them from Git's tracking (but keep them locally).**

**Command:**

**git rm -r --cached logs/ media/ \_\_pycache\_\_/**

**What This Does:**

* **git rm → Removes files from Git tracking.**
* **-r → Recursively removes everything inside the specified directories.**
* **--cached → Only removes files from Git's tracking, not from your local system. Your files will still be on your computer but won't be pushed to GitHub.**

**3. Commit the .gitignore Changes**

**Now that we've removed the unwanted files, we need to commit the updated .gitignore file.**

**Command:**

**git commit -m "Update .gitignore to exclude logs, media, and compiled files"**

**What This Does:**

* **git commit -m "message" → Saves our changes to Git with a message describing what we did.**
* **This ensures that .gitignore is respected moving forward.**

**4. Push Your Code to GitHub**

**Now that everything is properly set up, we can push the project to GitHub.**

**Step 1: Connect to Your GitHub Repository**

**git remote add origin https://github.com/your-username/DjangoExample.git**

* **git remote add origin <repo\_url> → Links your local project to the GitHub repository.**

**\*\*Step 2: Rename the Default Branch to \*\*main**

**git branch -M main**

* **Some older versions of Git use master as the default branch, but main is the new standard.**

**Step 3: Push Your Project to GitHub**

**git push -u origin main**

* **git push → Uploads your commits to GitHub.**
* **-u origin main → Sets the default upstream branch so you can simply use git push next time.**

**Final Verification**

**After running these commands, go to your GitHub repository page and refresh. You should see your project files without unnecessary files like logs, media, or compiled files.**

**Next Steps**

**✅ Now, every time you make changes, you can stage, commit, and push updates with:**

**git add .**

**git commit -m "Describe your changes"**

**git push**

**Let me know if anything needs clarification! 🚀**

**Setting Up Git and GitHub for Your Django Project**

**1. Modify .gitignore to Avoid Unwanted Files**

**Why?**

Git tracks all files by default, but some files should **not** be committed to GitHub, such as:

* **Virtual environments (env/)** – These contain dependencies but can be recreated using requirements.txt.
* **Database files (\*.sqlite3)** – These are generated during Django development and shouldn’t be stored in Git.
* **Compiled Python files (\_\_pycache\_\_/ and \*.pyc)** – Python automatically generates these, so we don't need them in version control.
* **Log files (logs/)** – These may contain sensitive debugging data.
* **Media files (media/)** – User-uploaded files (e.g., recordings) should not be committed.
* **History files (.history/)** – These may be auto-generated by VS Code or other tools.

**How?**

Edit your .gitignore file (or create one if it doesn’t exist) and add the following lines:

# Ignore environment files

env/

\*.env

# Ignore database files

\*.sqlite3

# Ignore compiled Python files

\_\_pycache\_\_/

\*.pyc

# Ignore logs

logs/

# Ignore media/uploads (recordings)

media/

# Ignore history files

.history/

**2. Remove Previously Tracked Unwanted Files**

Even though we **added entries** to .gitignore, Git has already been tracking some of these files. We need to remove them from Git's tracking **(but keep them locally)**.

**Command:**

git rm -r --cached logs/ media/ \_\_pycache\_\_/

**What This Does:**

* git rm → Removes files from Git tracking.
* -r → Recursively removes everything inside the specified directories.
* --cached → **Only removes files from Git's tracking, not from your local system.** Your files will still be on your computer but won't be pushed to GitHub.

**3. Commit the .gitignore Changes**

Now that we've removed the unwanted files, we need to **commit** the updated .gitignore file.

**Command:**

git commit -m "Update .gitignore to exclude logs, media, and compiled files"

**What This Does:**

* git commit -m "message" → Saves our changes to Git with a message describing what we did.
* This ensures that .gitignore is respected moving forward.

**4. Push Your Code to GitHub**

Now that everything is properly set up, we can push the project to GitHub.

**Step 1: Connect to Your GitHub Repository**

git remote add origin https://github.com/your-username/DjangoExample.git

* git remote add origin <repo\_url> → Links your local project to the GitHub repository.

**\*\*Step 2: Rename the Default Branch to \*\*main**

git branch -M main

* Some older versions of Git use master as the default branch, but main is the new standard.

**Step 3: Push Your Project to GitHub**

git push -u origin main

* git push → Uploads your commits to GitHub.
* -u origin main → Sets the default upstream branch so you can simply use git push next time.

**Final Verification**

After running these commands, **go to your GitHub repository page and refresh**. You should see your project files **without** unnecessary files like logs, media, or compiled files.

**Next Steps**

✅ Now, every time you make changes, you can **stage, commit, and push** updates with:

git add .

git commit -m "Describe your changes"

git push

Let me know if anything needs clarification! 🚀

**Setting Up Git and GitHub for Your Django Project**

**1. Modify .gitignore to Avoid Unwanted Files**

**Why?**

Git tracks all files by default, but some files should **not** be committed to GitHub, such as:

* **Virtual environments (env/)** – These contain dependencies but can be recreated using requirements.txt.
* **Database files (\*.sqlite3)** – These are generated during Django development and shouldn’t be stored in Git.
* **Compiled Python files (\_\_pycache\_\_/ and \*.pyc)** – Python automatically generates these, so we don't need them in version control.
* **Log files (logs/)** – These may contain sensitive debugging data.
* **Media files (media/)** – User-uploaded files (e.g., recordings) should not be committed.
* **History files (.history/)** – These may be auto-generated by VS Code or other tools.

**How?**

Edit your .gitignore file (or create one if it doesn’t exist) and add the following lines:

# Ignore environment files

env/

\*.env

# Ignore database files

\*.sqlite3

# Ignore compiled Python files

\_\_pycache\_\_/

\*.pyc

# Ignore logs

logs/

# Ignore media/uploads (recordings)

media/

# Ignore history files

.history/

**2. Remove Previously Tracked Unwanted Files**

Even though we **added entries** to .gitignore, Git has already been tracking some of these files. We need to remove them from Git's tracking **(but keep them locally)**.

**Command:**

git rm -r --cached logs/ media/ \_\_pycache\_\_/

**What This Does:**

* git rm → Removes files from Git tracking.
* -r → Recursively removes everything inside the specified directories.
* --cached → **Only removes files from Git's tracking, not from your local system.** Your files will still be on your computer but won't be pushed to GitHub.

**3. Commit the .gitignore Changes**

Now that we've removed the unwanted files, we need to **commit** the updated .gitignore file.

**Command:**

git commit -m "Update .gitignore to exclude logs, media, and compiled files"

**What This Does:**

* git commit -m "message" → Saves our changes to Git with a message describing what we did.
* This ensures that .gitignore is respected moving forward.

**4. Push Your Code to GitHub**

Now that everything is properly set up, we can push the project to GitHub.

**Step 1: Connect to Your GitHub Repository**

git remote add origin https://github.com/your-username/DjangoExample.git

* git remote add origin <repo\_url> → Links your local project to the GitHub repository.

**\*\*Step 2: Rename the Default Branch to \*\*main**

git branch -M main

* Some older versions of Git use master as the default branch, but main is the new standard.

**Step 3: Push Your Project to GitHub**

git push -u origin main

* git push → Uploads your commits to GitHub.
* -u origin main → Sets the default upstream branch so you can simply use git push next time.

**Final Verification**

After running these commands, **go to your GitHub repository page and refresh**. You should see your project files **without** unnecessary files like logs, media, or compiled files.

**Next Steps**

✅ Now, every time you make changes, you can **stage, commit, and push** updates with:

git add .

git commit -m "Describe your changes"

git push

Let me know if anything needs clarification! 🚀

**Setting Up Git and GitHub for Your Django Project**

**1. Modify .gitignore to Avoid Unwanted Files**

**Why?**

Git tracks all files by default, but some files should **not** be committed to GitHub, such as:

* **Virtual environments (env/)** – These contain dependencies but can be recreated using requirements.txt.
* **Database files (\*.sqlite3)** – These are generated during Django development and shouldn’t be stored in Git.
* **Compiled Python files (\_\_pycache\_\_/ and \*.pyc)** – Python automatically generates these, so we don't need them in version control.
* **Log files (logs/)** – These may contain sensitive debugging data.
* **Media files (media/)** – User-uploaded files (e.g., recordings) should not be committed.
* **History files (.history/)** – These may be auto-generated by VS Code or other tools.

**How?**

Edit your .gitignore file (or create one if it doesn’t exist) and add the following lines:

# Ignore environment files

env/

\*.env

# Ignore database files

\*.sqlite3

# Ignore compiled Python files

\_\_pycache\_\_/

\*.pyc

# Ignore logs

logs/

# Ignore media/uploads (recordings)

media/

# Ignore history files

.history/

**2. Remove Previously Tracked Unwanted Files**

Even though we **added entries** to .gitignore, Git has already been tracking some of these files. We need to remove them from Git's tracking **(but keep them locally)**.

**Command:**

git rm -r --cached logs/ media/ \_\_pycache\_\_/

**What This Does:**

* git rm → Removes files from Git tracking.
* -r → Recursively removes everything inside the specified directories.
* --cached → **Only removes files from Git's tracking, not from your local system.** Your files will still be on your computer but won't be pushed to GitHub.

**3. Commit the .gitignore Changes**

Now that we've removed the unwanted files, we need to **commit** the updated .gitignore file.

**Command:**

git commit -m "Update .gitignore to exclude logs, media, and compiled files"

**What This Does:**

* git commit -m "message" → Saves our changes to Git with a message describing what we did.
* This ensures that .gitignore is respected moving forward.

**4. Push Your Code to GitHub**

Now that everything is properly set up, we can push the project to GitHub.

**Step 1: Connect to Your GitHub Repository**

git remote add origin https://github.com/your-username/DjangoExample.git

* git remote add origin <repo\_url> → Links your local project to the GitHub repository.

**\*\*Step 2: Rename the Default Branch to \*\*main**

git branch -M main

* Some older versions of Git use master as the default branch, but main is the new standard.

**Step 3: Push Your Project to GitHub**

git push -u origin main

* git push → Uploads your commits to GitHub.
* -u origin main → Sets the default upstream branch so you can simply use git push next time.

**Final Verification**

After running these commands, **go to your GitHub repository page and refresh**. You should see your project files **without** unnecessary files like logs, media, or compiled files.

**Next Steps**

✅ Now, every time you make changes, you can **stage, commit, and push** updates with:

git add .

git commit -m "Describe your changes"

git push

Let me know if anything needs clarification! 🚀

**Setting Up Git and GitHub for Your Django Project**

**1. Modify .gitignore to Avoid Unwanted Files**

**Why?**

Git tracks all files by default, but some files should **not** be committed to GitHub, such as:

* **Virtual environments (env/)** – These contain dependencies but can be recreated using requirements.txt.
* **Database files (\*.sqlite3)** – These are generated during Django development and shouldn’t be stored in Git.
* **Compiled Python files (\_\_pycache\_\_/ and \*.pyc)** – Python automatically generates these, so we don't need them in version control.
* **Log files (logs/)** – These may contain sensitive debugging data.
* **Media files (media/)** – User-uploaded files (e.g., recordings) should not be committed.
* **History files (.history/)** – These may be auto-generated by VS Code or other tools.

**How?**

Edit your .gitignore file (or create one if it doesn’t exist) and add the following lines:

# Ignore environment files

env/

\*.env

# Ignore database files

\*.sqlite3

# Ignore compiled Python files

\_\_pycache\_\_/

\*.pyc

# Ignore logs

logs/

# Ignore media/uploads (recordings)

media/

# Ignore history files

.history/

**2. Remove Previously Tracked Unwanted Files**

Even though we **added entries** to .gitignore, Git has already been tracking some of these files. We need to remove them from Git's tracking **(but keep them locally)**.

**Command:**

git rm -r --cached logs/ media/ \_\_pycache\_\_/

**What This Does:**

* git rm → Removes files from Git tracking.
* -r → Recursively removes everything inside the specified directories.
* --cached → **Only removes files from Git's tracking, not from your local system.** Your files will still be on your computer but won't be pushed to GitHub.

**3. Commit the .gitignore Changes**

Now that we've removed the unwanted files, we need to **commit** the updated .gitignore file.

**Command:**

git commit -m "Update .gitignore to exclude logs, media, and compiled files"

**What This Does:**

* git commit -m "message" → Saves our changes to Git with a message describing what we did.
* This ensures that .gitignore is respected moving forward.

**4. Push Your Code to GitHub**

Now that everything is properly set up, we can push the project to GitHub.

**Step 1: Connect to Your GitHub Repository**

git remote add origin https://github.com/your-username/DjangoExample.git

* git remote add origin <repo\_url> → Links your local project to the GitHub repository.

**\*\*Step 2: Rename the Default Branch to \*\*main**

git branch -M main

* Some older versions of Git use master as the default branch, but main is the new standard.

**Step 3: Push Your Project to GitHub**

git push -u origin main

* git push → Uploads your commits to GitHub.
* -u origin main → Sets the default upstream branch so you can simply use git push next time.

**Final Verification**

After running these commands, **go to your GitHub repository page and refresh**. You should see your project files **without** unnecessary files like logs, media, or compiled files.

**Next Steps**

✅ Now, every time you make changes, you can **stage, commit, and push** updates with:

git add .

git commit -m "Describe your changes"

git push

Let me know if anything needs clarification! 🚀