Git Repository Creation and Project Setup

This document describes how to create a Git repository on GitHub and push the local `intelliDGAforge` project structure to it. Follow these steps to replicate the process and ensure your project is safely version-controlled in the cloud.

1. Verify Git Installation

1. Open your terminal (on macOS, Windows or Linux).
2. Run `git --version` to ensure Git is installed. The output should look something like `git version 2.x.x`.
3. If Git is not installed, install it using your operating system’s package manager (e.g., `brew install git` on macOS).

2. Configure Your Git Identity

Before making commits, set up your name and email so they are recorded in your commit history:

```sh
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

These settings only need to be configured once per machine.

3. Navigate to the Local Project

Change directory into the root of your local project. Replace the path below with the correct location for your machine:

```sh
cd ~/Desktop/intelliDGAforge
```

```

## 4. Initialize a Git Repository

Inside the project directory, run:

```sh
git init
```

This command creates a hidden `.git` directory and prepares your project for version control.

5. Stage and Commit Your Files

1. Add all files to the staging area:

   ```sh
   git add .
   ```

2. Commit the staged files with a descriptive message:

   ```sh
   git commit -m "Initial commit"
   ```

This records the current state of your project in the local repository.

6. Create a Remote Repository on GitHub

1. Open a web browser, visit [GitHub](https://github.com) and sign in.

2. From your dashboard, click the **+** icon (upper-right corner) and select **New repository**.
   ```

3. Enter a **Repository name** (e.g., `intelliDGAforge`), optionally provide a description, and choose whether it will be public or private.
4. **Do not** initialize the repository with a README—your local project already contains the files.
5. Click **Create repository**. GitHub will display a URL that looks like `https://github.com/<username>/<repository>.git`. Note this URL for the next step.

7. Link the Local Repository to the Remote

Add the GitHub repository as a remote named `origin`:

```sh
git remote add origin https://github.com/<username>/<repository>.git
```

Replace `<username>` and `<repository>` with your GitHub username and the repository name. If a remote named `origin` already exists, update it instead:

```sh
git remote set-url origin https://github.com/<username>/<repository>.git
```

8. Rename the Default Branch (Optional)

GitHub uses `main` as the default branch name. If your local branch is not already named `main`, rename it with:

```sh
git branch -M main
```

This step is optional but helps your local repository align with GitHub’s default conventions.

9. Push the Code to GitHub

Send your local commits to the remote repository using:

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

The `-u` flag tells Git to remember the upstream branch, so future pushes can be done simply with `git push`.

If prompted, authenticate using your GitHub credentials or a Personal Access Token (PAT).

10. Troubleshooting

* **Permission denied (403) errors:** Make sure you have write access to the remote repository or that you created the repo under your own account. If you are using an account without privileges, ask the repository owner to add you as a collaborator.
* **`remote origin already exists` errors:** This happens if you previously set a remote. Use `git remote set-url origin <new_url>` to update the existing remote instead of adding a new one.
* **Authentication issues:** If Git prompts you repeatedly for credentials, consider configuring a credential helper or generating a PAT via GitHub and using it when prompted.

11. Verify the Repository

After pushing your code, visit your repository’s URL in your browser. You should see all the folders and files from your local `intelliDGAforge` project (e.g., `assets`, `backend`, `frontend`, `deployment`, etc.) along with your commit history. This confirms that your push was successful.

Following these steps will create a GitHub repository and link your local project to it. You can

now collaborate with others, track changes, and manage your project using Git.