Git assignment:

Part 1: Basic Git Setup and Operations (30%)

1. Create a new project repository that will serve as a template for future microservices

2. Set up proper Git configuration including user information and core settings

3. Develop a .gitignore file suitable for your tech stack

4. Create an initial project structure with: ○ Documentation folder

○ Source code folder ○ Test folder ○ Infrastructure as code folder

5. Demonstrate proper commit message conventions for at least 10 commits

6. Create and maintain a changelog using Git logs

Answer:

**1. Create a new project repository**

**Since you want this on your GitHub account, follow these steps:**

1. **Go to** [**my GitHub profile**](https://github.com/ShantoITE01)**.**
2. **Click on "New" to create a new repository.**
3. **Name the repository something like microservice-template.**
4. **Add a description like: *"Template repository for microservices with Git best practices."***
5. **Select "Public" or "Private" based on preference.**
6. **Initialize with a README.md.**
7. **Click Create Repository.**

Ans-2. git config --list

user.name=Md Hasibul Hasan Shanto

[user.email=shantoaussie097@gmail.com](mailto:user.email=shantoaussie097@gmail.com)

Ans-3. touch .gitignore

oot@hasibul-virtual-machine:/home/hasibul/my-microservice-template# ls -la

total 12

drwxr-xr-x 3 root root 4096 Feb 23 12:53 .

drwxr-x--- 18 hasibul hasibul 4096 Feb 23 12:19 ..

drwxr-xr-x 7 root root 4096 Feb 23 12:19 .git

-rw-r--r-- 1 root root 0 Feb 23 12:53 .gitignore

root@hasibul-virtual-machine:/home/hasibul/my-microservice-template# vim .gitignore

root@hasibul-virtual-machine:/home/hasibul/my-microservice-template# cat .gitignore

# Python compiled files

\_\_pycache\_\_/

\*.pyc

\*.pyo

\*.pyd

# Virtual environment folders

venv/

.env/

# Logs and temporary files

logs/

\*.log

Ans-4. Create an initial project structure with: ○ Documentation folder

○ Source code folder ○ Test folder ○ Infrastructure as code folder

mkdir -p docs src tests infra

touch README.md CHANGELOG.md .gitignore

Output:

root@hasibul-virtual-machine:/home/hasibul/my-microservice-template# ls -la

total 32

drwxr-xr-x 7 root root 4096 Feb 23 13:46 .

drwxr-x--- 18 hasibul hasibul 4096 Feb 23 12:19 ..

-rw-r--r-- 1 root root 0 Feb 23 13:46 CHANGELOG.md

drwxr-xr-x 2 root root 4096 Feb 23 13:22 docs

drwxr-xr-x 7 root root 4096 Feb 23 12:19 .git

-rw-r--r-- 1 root root 139 Feb 23 13:46 .gitignore

drwxr-xr-x 2 root root 4096 Feb 23 13:22 infra

-rw-r--r-- 1 root root 0 Feb 23 13:46 README.md

drwxr-xr-x 2 root root 4096 Feb 23 13:22 src

drwxr-xr-x 2 root root 4096 Feb 23 13:22 tests

5. git commit -m "fix: resolve bug in login function"

On branch master

nothing to commit, working tree clean

root@hasibul-virtual-machine:/home/hasibul/my-microservice-template

# git commit -m "chore: update dependencies"

On branch master

nothing to commit, working tree clean

root@hasibul-virtual-machine:/home/hasibul/my-microservice-template

# git commit -m "docs: add API usage instructions"

On branch master

nothing to commit, working tree clean

Ans-6. root@hasibul-virtual-machine:/home/hasibul/my-microservice-template# git log --oneline --decorate --graph --all > CHANGELOG.md

root@hasibul-virtual-machine:/home/hasibul/my-microservice-template# git add CHANGELOG.md

root@hasibul-virtual-machine:/home/hasibul/my-microservice-template# git commit -m "docs: add changelog"

[master 390b30d] docs: add changelog

1 file changed, 1 insertion(+)

root@hasibul-virtual-machine:/home/hasibul/my-microservice-template# git log

commit 390b30dfcb9ff5d3df929e47b69fbc4f1255972b (HEAD -> master)

Author: Md Hasibul Hasan Shanto <shantoaussie097@gmail.com>

Date: Sun Feb 23 15:04:40 2025 +1100

docs: add changelog

commit 092fe4a6bd83ec74a0a0d58d6e3b42840e34c264

Author: Md Hasibul Hasan Shanto <shantoaussie097@gmail.com>

Date: Sun Feb 23 13:53:36 2025 +1100

chore: initial project setup with directory structure

**Part 2: Branching and Collaboration (40%)**

**1. Branching Strategy**

**Document your branching strategy:**

* **main → Production-ready code**
* **develop → Integration branch for ongoing development**
* **Feature branches → feature/authentication, feature/payment**
* **Release branch → release/v1.0**
* **Hotfix branch → hotfix/login-fix**

**2. Create and Manage Branches**

**git checkout -b develop**

**git push -u origin develop**

**git checkout -b feature/authentication develop**

**git push -u origin feature/authentication**

**3. Pull Request Templates**

**Create .github/PULL\_REQUEST\_TEMPLATE.md with sections:**

**## Description**

**<!-- Describe your changes here -->**

**## Checklist**

**- [ ] Unit tests added**

**- [ ] Documentation updated**

**You can also add different templates under .github/PULL\_REQUEST\_TEMPLATE/.**

**4. Simulate and Resolve a Merge Conflict**

* **Modify the same file in feature/authentication and develop**
* **Merge into develop:**

**git checkout develop**

**git merge feature/authentication**

* **Resolve conflicts manually and commit.**

**5. Merge vs. Rebase**

**Merge:**

**git checkout develop**

**git merge feature/authentication**

**Rebase:**

**git checkout feature/authentication**

**git rebase develop**

**6. Collaboration Guide**

**Document in docs/GIT\_WORKFLOW.md:**

* **How to branch**
* **When to rebase vs. merge**
* **Commit message conventions**

**Part 3: Advanced Git Implementation (30%)**

**1. Set Up Git Hooks**

**Pre-commit hook for linting (.git/hooks/pre-commit)**

**#!/bin/sh**

**eslint src/**

**Make it executable:**

**chmod +x .git/hooks/pre-commit**

**Commit message validation (.git/hooks/commit-msg)**

**#!/bin/sh**

**if ! grep -qE "^(feat|fix|chore|docs|test|refactor|perf|ci|build|revert): " "$1"; then**

**echo "❌ Commit message format incorrect!"**

**exit 1**

**fi**

**Enable:**

**chmod +x .git/hooks/commit-msg**

**2. Implement Git LFS for Large Files**

**git lfs install**

**git lfs track "\*.mp4" "\*.zip"**

**git add .gitattributes**

**git commit -m "chore: configure Git LFS"**

**3. Use Submodules for Common Utilities**

**git submodule add https://github.com/your-org/common-utils.git src/utils**

**git commit -m "chore: add common utilities submodule"**

**4. Security Measures**

**Signed Commits**

**git config --global user.signingkey YOUR\_GPG\_KEY**

**git config --global commit.gpgsign true**

**#Protected Branches (GitHub Settings)**

* **Require PR approvals before merging**
* **Require signed commits**
* **Enforce branch rules**

**Final Steps**

**Push all changes:**

**git push --all**