Aim 1: Clone, Create, Commit, and Push to Remote

> GitHub Version (with cloning)

```
$ mkdir Aim1 && cd Aim1
$ git clone <repository_url>
$ cd <repo_name>
$ git checkout -b feature-branch
$ echo "Some content for the file" > newfile.txt
$ git add newfile.txt
$ git commit -m "Add newfile.txt with initial content"
$ git push origin feature-branch
```

> Local Repository Version (without cloning)

```
$ mkdir Aim1 && cd Aim1
$ git init
$ git checkout -b feature-branch
$ echo "Some content for the file" > newfile.txt
$ git add newfile.txt
$ git commit -m "Add newfile.txt with initial content"
```

Aim 2: Create and Merge Branches

> GitHub Version (with cloning)

```
$ mkdir Aim2 && cd Aim2
$ git clone <repository_url>
$ cd <repo_name>
$ git checkout -b my-feature
$ echo "New feature line" >> README.md
```

$ git add README.md

$ git commit -m "Update README with new feature info"

$ git checkout main

$ git merge my-feature

> Local Repository Version (without cloning)

$ mkdir Aim2 && cd Aim2

$ git init

$ touch README.md && git add README.md && git commit -m "Initial commit"

$ git checkout -b my-feature

$ echo "New feature line" >> README.md

$ git add README.md

$ git commit -m "Update README with new feature info"

$ git checkout main

$ git merge my-feature

Aim 3: Revert a Commit

> GitHub Version (with cloning)

$ mkdir Aim3 && cd Aim3

$ git clone <repository_url>

$ cd <repo_name>

$ echo "Commit 1" > file.txt

$ git add file.txt

$ git commit -m "First commit"

$ echo "Commit 2" >> file.txt

$ git commit -am "Second commit"

$ echo "Commit 3" >> file.txt

$ git commit -am "Third commit"

$ git log --oneline

$ git revert <commit_hash_of_second_commit>

$ git log --oneline

> Local Repository Version (without cloning)

$ mkdir Aim3 && cd Aim3

$ git init

$ echo "Commit 1" > file.txt

$ git add file.txt

$ git commit -m "First commit"

$ echo "Commit 2" >> file.txt

$ git commit -am "Second commit"

$ echo "Commit 3" >> file.txt

$ git commit -am "Third commit"

$ git log --oneline

$ git revert <commit_hash_of_second_commit>

$ git log --oneline

Aim 4: Modify a Commit Message (Amend Last Commit)

> GitHub Version (with cloning)

$ mkdir Aim4 && cd Aim4

$ git clone <repository_url>

$ cd <repo_name>

$ echo "Initial content" > file.txt

$ git add file.txt

$ git commit -m "Initial mesage"

$ git commit --amend -m "Initial message corrected"

$ git log --oneline

> Local Repository Version (without cloning)

$ mkdir Aim4 && cd Aim4

$ git init

$ echo "Initial content" > file.txt

$ git add file.txt

$ git commit -m "Initial mesage"

$ git commit --amend -m "Initial message corrected"

$ git log --oneline

Aim 5: Stash, Apply, and Pop Changes

> GitHub Version (with cloning)

$ mkdir Aim5 && cd Aim5

$ git clone <repository_url>

$ cd <repo_name>

$ echo "Temporary changes" >> file.txt

$ git stash

$ git checkout -b another-branch

$ git checkout main

$ git stash apply

$ # or git stash pop

$ git add file.txt

$ git commit -m "Apply and commit stashed changes"

> Local Repository Version (without cloning)

```
$ mkdir Aim5 && cd Aim5

$ git init

$ touch file.txt && git add file.txt && git commit -m "Initial commit"

$ echo "Temporary changes" >> file.txt

$ git stash

$ git checkout -b another-branch

$ git checkout main

$ git stash apply

$ # or git stash pop

$ git add file.txt

$ git commit -m "Apply and commit stashed changes"
```

Aim 6: Resolve Merge Conflicts

> GitHub Version (with cloning)

```
$ mkdir Aim6 && cd Aim6

$ git clone <repository_url>

$ cd <repo_name>

$ git checkout -b branch-1

$ echo "Conflict from branch-1" > conflict.txt

$ git add conflict.txt

$ git commit -m "Add conflicting change in branch-1"

$ git checkout main

$ git checkout -b branch-2

$ echo "Conflict from branch-2" > conflict.txt

$ git add conflict.txt

$ git commit -m "Add conflicting change in branch-2"

$ git checkout branch-1
```

$ git merge branch-2

$ # Resolve conflict manually in conflict.txt

$ git add conflict.txt

$ git commit -m "Resolve merge conflict between branch-1 and branch-2"

> Local Repository Version (without cloning)

$ mkdir Aim6 && cd Aim6

$ git init

$ touch conflict.txt && git add conflict.txt && git commit -m "Initial commit"

$ git checkout -b branch-1

$ echo "Conflict from branch-1" > conflict.txt

$ git add conflict.txt

$ git commit -m "Add conflicting change in branch-1"

$ git checkout main

$ git checkout -b branch-2

$ echo "Conflict from branch-2" > conflict.txt

$ git add conflict.txt

$ git commit -m "Add conflicting change in branch-2"

$ git checkout branch-1

$ git merge branch-2

$ # Resolve conflict manually in conflict.txt

$ git add conflict.txt

$ git commit -m "Resolve merge conflict between branch-1 and branch-2"

Aim 7: Work with Git Submodules

> GitHub Version (with cloning)

$ mkdir Aim7 && cd Aim7

```
$ git clone --recurse-submodules <repository_url>

$ cd <repo_name>

$ # If not using --recurse-submodules

$ git submodule init

$ git submodule update

$ cd path/to/submodule

$ echo "Submodule edit" >> subfile.txt

$ git add subfile.txt

$ git commit -m "Edit inside submodule"

$ cd ../..

$ git add path/to/submodule

$ git commit -m "Update submodule reference"
```

> Local Repository Version (without cloning)

```
$ mkdir Aim7 && cd Aim7

$ git init

$ # Add a submodule if required: git submodule add <repo_url> path/to/submodule

$ git submodule init

$ git submodule update

$ cd path/to/submodule

$ echo "Submodule edit" >> subfile.txt

$ git add subfile.txt

$ git commit -m "Edit inside submodule"

$ cd ../..

$ git add path/to/submodule

$ git commit -m "Update submodule reference"
```

Aim 8: Cherry-pick a Commit

> GitHub Version (with cloning)

```
$ mkdir Aim8 && cd Aim8
$ git clone <repository_url>
$ cd <repo_name>
$ git checkout -b feature-branch
$ echo "First feature" > file.txt
$ git add file.txt
$ git commit -m "First feature commit"
$ echo "Second feature" >> file.txt
$ git commit -am "Second feature commit"
$ git log --oneline
$ git checkout main
$ git cherry-pick <commit_hash>
$ git log --oneline
```

> Local Repository Version (without cloning)

```
$ mkdir Aim8 && cd Aim8
$ git init
$ git checkout -b feature-branch
$ echo "First feature" > file.txt
$ git add file.txt
$ git commit -m "First feature commit"
$ echo "Second feature" >> file.txt
$ git commit -am "Second feature commit"
$ git log --oneline
$ git checkout main
```

```
$ git cherry-pick <commit_hash>

$ git log --oneline
```