Git Cheat Sheet with Practice Scenarios

1. Setup Git

Scenario: Configure Git for the first time.

```
git config --global user.name "Your Name"
git config --global user.email "youremail@example.com"
git config --global core.editor "code --wait"  # Example for VS Code
git config --list
```

2. Create a Repository

Scenario: Create a new local repository.

```
mkdir my-repo
cd my-repo
git init
```

Clone a Public Repository:

```
git clone https://github.com/public/repo.git
cd repo
```

3. Add and Commit Files

Scenario: Add and save changes.

```
echo "Hello Git" > file.txt
git status
git add file.txt
git commit -m "Initial commit"
```

4. Branching

Scenario: Work on a feature without affecting the main branch.

```
git branch feature-branch
git checkout feature-branch
echo "Feature work" >> feature.txt
git add feature.txt
git commit -m "Add feature work"
git log --oneline --graph
```

Switch Branches:

```
git checkout main
```

5. Merging

Scenario: Merge feature work into the main branch.

```
git checkout main
git merge feature-branch
git log --oneline --graph
```

6. Undoing Changes

Scenario: Revert a committed change.

```
git revert <commit-hash>
```

Scenario: Reset to an earlier commit.

```
git reset --hard <commit-hash>
```

7. Working with Remotes

Scenario: Push changes to a remote repository.

git remote add origin https://github.com/your/repo.git

git push -u origin main

Pull Changes:

git pull origin main

8. Stashing

Scenario: Temporarily save changes.

git stash

git stash list

git stash pop

9. Resolving Conflicts

Scenario: Merge branches with conflicts.

 # Simulate a conflict

echo "Main branch change" >> file.txt

git commit -am "Change in main branch"

git checkout feature-branch

echo "Feature branch change" >> file.txt

git commit -am "Change in feature branch"

git checkout main

git merge feature-branch

 # Resolve conflict in file.txt

git add file.txt

git commit -m "Resolve merge conflict"

10. Logs and Diffs

Scenario: Review history and changes.

```
git log --oneline --graph
git diff HEAD~1 HEAD
```

11. Tags

Scenario: Create and push a tag for releases.

```
git tag v1.0
git push origin v1.0
```

12. Git Ignore

Scenario: Ignore specific files.

```
echo "*.log" > .gitignore
git add .gitignore
git commit -m "Add .gitignore"
```

13. Collaboration

Scenario: Review changes from others.

```
git fetch
git merge origin/main
```

14. Advanced Commands

Scenario: Squash commits.

```
git rebase -i HEAD~3
```

Scenario: Bisect a bug.

```
git bisect start
git bisect bad
```

git bisect good <commit-hash>

Practice Steps:

1. **Create a GitHub repository and practice all commands above.**
2. **Fork a public repository, make changes, and submit a pull request.**
3. **Create conflicts intentionally and resolve them.**