1. **Account setup and Configuration, contributing to a project, Maintaining a project**

1. Account Setup & Configuration

- **Set user details:**
- git config --global user.name "Your Name"
- git config --global user.email "your@email.com"
- **Check config:**
- git config --list

2. Contributing to a Project

- **Fork & clone:**
- git clone repo_url
- **Create a feature branch:**
- git checkout -b feature-branch
- **Commit & push:**
- git add .
- git commit -m "Feature update"
- git push origin feature-branch
- Open a pull request (PR) on GitHub/GitLab.

3. Maintaining a Project

- **Fetch latest changes:**
- git fetch && git pull
- Review & merge PRs.
- **Tag releases:**
- git tag -a v1.0 -m "Version 1.0"
- git push origin v1.0

2. **Managing an organization, Scripting GitHub, Revision selection**

1. Managing an Organization

- Create an org on GitHub.
- Add members & set roles: Owner, Admin, Member.
- **Manage repositories & permissions:**
- Settings → Manage Access

2. Scripting GitHub (Automation)

- **Use GitHub CLI:**
- gh repo create myrepo --private
- gh issue list
- **Automate with API:**
- curl -H "Authorization: token YOUR_TOKEN" https://api.github.com/user/repos

3. Revision Selection in Git

- **View commit history:**
- git log --oneline
- **Checkout a specific commit:**
- git checkout <commit-hash>
- **Reset to an older commit:**
- git reset --hard <commit-hash>

3. Interactive staging, Git pull and merge conflicts, git fetch, Rebasing

Git Interactive Staging, Pull, Fetch, Merge, and Rebase

1. Interactive Staging
- Stage changes interactively:
- git add -i
- Use git add -p to review and stage hunks.

2. Git Pull & Merge Conflicts
- Fetch and merge changes:
- git pull origin main
- If conflicts occur, edit files, then:
- git add .
- git commit -m "Resolved merge conflict"

3. Git Fetch (Safe Update)
- Fetch updates without merging:

- git fetch
- View remote changes:
- git log origin/main --oneline

4. Rebasing (Linear History)

- Rebase onto main:
- git checkout feature-branch
- git rebase main
- If conflicts occur, resolve them, then:
- git rebase --continue

4. **Create a Git account on platforms like GitHub and Configure Git with your username and email**

Create a Git Account & Configure Git

1. Create a GitHub/GitLab/Bitbucket Account

- Go to GitHub, GitLab, or Bitbucket
- Sign up and verify your email

2. Configure Git Locally

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

3. Verify Configuration

git config --list

5. **Fork a project on GitHub, clone it to your local machine, make changes, stage them, commit, push to your fork and create a pull request.**

Fork, Clone, Edit, and Create a Pull Request on GitHub

1. Fork the Repository

- Go to the GitHub repo.
- Click Fork (top-right corner).

2. Clone Your Fork (Local Machine)

git clone https://github.com/your-username/repo-name.git
cd repo-name

3. Create a New Branch

git checkout -b feature-branch

4. Make Changes & Stage Them

git add .
git commit -m "Added new feature"

5. Push Changes to Your Fork

git push origin feature-branch

6. Create a Pull Request (PR)

- Go to your fork on GitHub.
- Click Compare & pull request.
- Add a description and submit the PR.

6. **Create and manage organizations on GitHub or other Git hosting platforms. Set permissions, manage teams, and oversee repository access.**

Create & Manage Organizations on GitHub (or Other Git Platforms)

1. Create an Organization

- Go to GitHub → Your profile → Your organizations → New organization
- Choose a plan and set the org name

2. Add Members & Manage Teams

- **Invite members:**
- Settings → People → Invite Member
- **Create teams & set roles:**
- Settings → Teams → New Team

3. Set Repository Access & Permissions

- **Org-wide access:**
- Settings → Member Privileges
- **Repo-specific roles (Read, Write, Admin):**
- Repo → Settings → Manage Access
- **Use branch protection:**
- Settings → Branches → Add Rule

7. **Using commands like git log, git show, and git diff to review commit history, view changes in specific commits, and select revisions for merging or reverting.**

1. View Commit History (git log)

- **Simple log:**
- git log --oneline
- **Detailed log with diffs:**

- git log -p
- **Filter by author:**
- git log --author="Your Name"

2. View Specific Commit (git show)

- **Show details of a commit:**
- git show <commit-hash>

3. Compare Changes (git diff)

- **Unstaged vs last commit:**
- git diff
- **Staged vs last commit:**
- git diff --staged
- **Between commits:**
- git diff commit1 commit2

8. **Use git add -p for interactive staging, allowing you to selectively stage changes within files.**

Interactive Staging with **git add -p**

1. Start Interactive Staging

git add -p

2. Review Each Change (Hunk)

Git will show changes in chunks (hunks) and prompt options:

- **y** → Stage this hunk
- **n** → Skip this hunk
- **s** → Split hunk for finer selection
- **e** → Edit the hunk manually
- **q** → Quit

3. Confirm Staged Changes

git status

9. **Pull changes from remote (git pull) and resolve merge conflicts using a merge tool (git merge tool) or manual editing. Commit the resolved conflicts.**

1. Pull Changes from Remote

git pull origin main

2. If a Merge Conflict Occurs:

- Git marks conflicting files with <<<<<<<, =======, and >>>>>>>.
- View conflicts:
- git status

3. Resolve Conflicts

- Use a Merge Tool (e.g., VS Code, Meld):
- git mergetool
- Or Manually Edit the Files
    - Keep the correct changes, then save.

4. Stage & Commit Resolved Conflicts

git add .

git commit -m "Resolved merge conflict"