Git Hands-On Labs — Solutions & Commands

This consolidated PDF contains step-by-step answers and commands for the 5 Git hands-on labs you uploaded. Each lab section includes objectives, required commands, expected behaviour and notes so you can run these on your machine (Git Bash on Windows / Terminal on macOS/Linux).

Lab 1 — Setup, configure Git and add a file

Objectives:
- Familiarize with git init, git status, git add, git commit, git push, git pull.

Steps & Commands (run in Git Bash):
1. Verify git installation:
   git --version

2. Configure user name and email:
   git config --global user.name "Your Name"
   git config --global user.email "you@example.com"

3. Verify configuration:
   git config --list

4. Create a repository folder and initialize:
   mkdir GitDemo
   cd GitDemo
   git init

5. Create a file and add content (example using echo):
   echo "Welcome to GitDemo" > welcome.txt
   cat welcome.txt

6. Check status, stage and commit:
   git status
   git add welcome.txt
   git commit -m "Add welcome.txt"

7. Link to remote (GitLab/GitHub) and push:
   git remote add origin https://gitlab.com/youruser/GitDemo.git
   git push -u origin master

8. To pull changes from remote:
   git pull origin master

Notes:
- Replace remote URL with your repository URL. If the remote branch is 'main' use main instead of master.

Lab 2 — .gitignore (ignore files/folders)

Objectives:
- Implement .gitignore to ignore unwanted files (e.g., .log files, log folders).

Steps & Commands:
1. In your repo root create .gitignore:
   echo "*.log" >> .gitignore
   echo "log/" >> .gitignore

2. Create sample files to test:
   mkdir log
   echo "log content" > app.log
   echo "inside log" > log/debug.log

3. Check git status — ignored files should not appear as untracked.
   git status --ignored

4. If a file was already tracked, remove it from tracking but keep locally:
   git rm --cached app.log
   git commit -m "Stop tracking app.log and add .gitignore"

Notes:
- Use `git status --ignored` to verify ignored files. Ensure .gitignore is committed.

Lab 3 — Branching and Merging

Objectives:
- Create a branch, make changes, merge to master and observe logs.

Steps & Commands:
1. Create and switch to a new branch:
   git branch GitNewBranch
   git checkout GitNewBranch
   # or git switch -c GitNewBranch

2. Make changes, add files, commit:
   echo "feature work" > feature.txt
   git add feature.txt
   git commit -m "Add feature.txt on GitNewBranch"

3. List branches:
   git branch -a

4. Switch back to master and merge:
   git checkout master
   git merge GitNewBranch

5. Visual log and cleanup:
   git log --oneline --graph --decorate --all
   git branch -d GitNewBranch

Notes:
- If merge produces conflicts, see Lab 4. Use `git switch` on newer Git versions.

Lab 4 — Conflict resolution during merge

Objectives:

- Create a conflict and resolve it using a 3-way merge tool or manually.

Steps & Commands (example):
1. Ensure master is clean:
   git status

2. Create branch and change a file:
   git checkout -b GitWork
   echo "Branch content A" > hello.xml
   git add hello.xml
   git commit -m "Add hello.xml on GitWork"

3. Switch to master and create conflicting change:
   git checkout master
   echo "Master content B" > hello.xml
   git add hello.xml
   git commit -m "Add hello.xml on master"

4. Merge branch to master (conflict expected):
   git merge GitWork

5. Resolve conflict manually or with merge tool (P4Merge example):
   git mergetool   # opens configured merge tool
   # After resolving, mark resolved:
   git add hello.xml
   git commit -m "Resolve merge conflict for hello.xml"

6. Cleanup and ignore backups:
   echo "*.orig" >> .gitignore
   git add .gitignore
   git commit -m "Ignore merge backup files"

Notes:
- Configure a merge tool: `git config --global merge.tool p4merge` and set path accordingly.

Lab 5 — Clean up and push back to remote

Objectives:
- Pull latest from remote, clean up branches, and push pending changes.

Steps & Commands:
1. Verify master is clean:
   git checkout master
   git status

2. List branches:
   git branch -a

3. Pull from remote:
   git pull origin master

4. Push pending local commits:

```
git push origin master
```

5. Delete merged remote branch (example):
```
git push origin --delete GitOldBranch
```

Notes:
- Use `git fetch --prune` to remove remote-tracking branches that no longer exist remotely.

Appendix — Useful Git commands & tips

- Set default editor to Notepad++ (Windows):
```
git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -nosession -notabbar"
```

- Create alias for notepad++ in bash (example .bashrc):
```
alias np="'/c/Program\ Files/Notepad++/notepad++.exe'"
```

- Common commands:
```
git stash            # save local changes temporarily
git restore <file>   # discard changes in working tree (newer git)
git reset --soft HEAD~1   # undo last commit but keep changes staged
```

If you want this document customized with screenshots, actual outputs or converted to slides, tell me which sections to expand.