Hands-On Lab (HOL)\:Understanding Git Diff – Comparing Working Directory, Staging Area, and Repository

Author

Dr. Sandeep Kumar Sharma

Learning Objective

The objective of this Hands-On Lab is to help learners clearly understand the **git diff** command by visualizing and comparing changes between the Working Directory, Staging Area (Index), and the latest committed version (HEAD). This lab simplifies the Git diff concept using a real file and incremental changes.

Learning Outcome

After completing this lab, learners will be able to:

- Explain what `git diff` does in simple terms
- Differentiate between unstaged and staged changes
- Use `git diff`, `git diff HEAD`, and `git diff --staged` confidently
- Map Git diff outputs to Git internal architecture
- Debug file changes before committing

Pre-requisite Understanding (Very Important)

Before running `git diff`, always remember Git has **three comparison points**:

1. **Working Directory (WD)** – Current file changes
2. **Staging Area (Index)** – Changes added using `git add`
3. **Repository (HEAD)** – Last committed snapshot

```
Working Directory  <-->  Staging Area  <-->  Repository (HEAD)
```

`git diff` simply compares **two of these areas**.

Step-by-Step Hands-On Lab: Git Diff

Initial State Assumption

- `login.php` already exists
- File is already committed at least once
- Working tree is clean

Step 1: Modify the File (Working Directory Change)

```
vi login.php
```

What you do:

- Press `i` (insert mode)
- Modify or add new content
- Press `Esc` → `:wq`

Git Internal State:

- File changed only in **Working Directory**
- Git has NOT tracked this change yet

Step 2: Check Git Status

```
git status
```

Explanation:

- Git shows `login.php` as **modified**
- Changes exist only in Working Directory

Step 3: Stage the File

```
git add login.php
```

Explanation:

- Current snapshot moved to **Staging Area**
- Working Directory and Index are now in sync

Step 4: Verify Staged Status

```
git status
```

Explanation:

- File shows as **Changes to be committed**

Step 5: Modify the File Again (After Staging)

```
vi login.php
```

Explanation:

- File modified again after staging

Important Concept:

- Staging Area has **old version**
- Working Directory has **new version**

Step 6: Check Git Status Again

```
git status
```

Explanation:

- File appears in **two states**:
- Staged changes
- Unstaged changes

Step 7: Compare Working Directory vs Staging Area

```
git diff login.php
```

What this shows:

- Difference between **Working Directory** and **Staging Area**

In simple words:

"What did I change AFTER `git add`?"

Step 8: Compare Working Directory vs Repository (HEAD)

```
git diff HEAD login.php
```

What this shows:

- Difference between **Working Directory** and **Last Commit**

In simple words:

"What all changes have I made since my last commit?"

Step 9: Compare Staging Area vs Repository (HEAD)

```
git diff --staged HEAD login.php
```

What this shows:

- Difference between **Staging Area** and **Repository**

In simple words:

"What exactly will go into my next commit?"

Quick Comparison Table

Command	Comparison
`git diff`	Working Directory ↔ Staging Area
`git diff HEAD`	Working Directory ↔ Repository
`git diff --staged HEAD`	Staging Area ↔ Repository

Very Common Interview Questions

Q1: Why does ***_git diff_* show nothing sometimes?**
➡️Because there is no difference between WD and Index

Q2: Can Git track multiple versions of the same file?
➡️Yes, simultaneously in WD, Index, and Repository

Q3: Which diff is checked before commit?
➡️`git diff --staged`

Conclusion (Trainer Note)

Think of `git diff` as a **comparison microscope**. Git never guesses—Git compares snapshots. Once learners understand *what is being compared*, `git diff` becomes one of the most powerful debugging and validation tools in Git.
