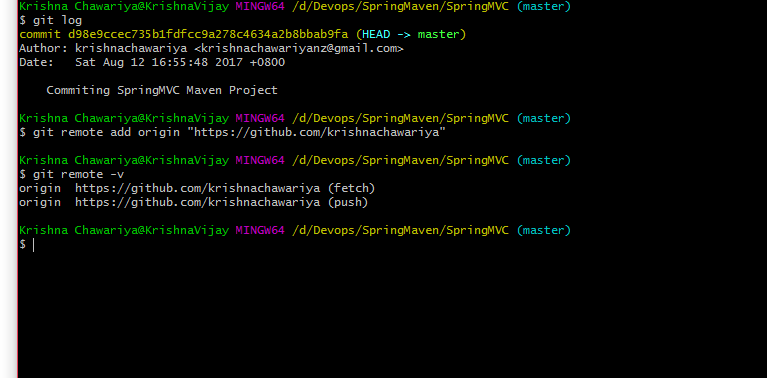
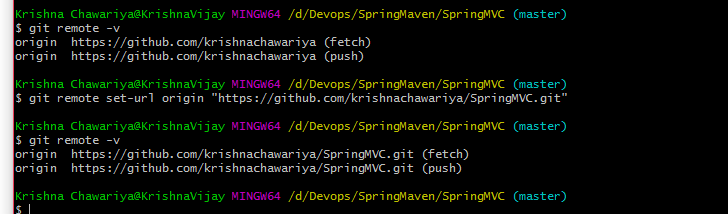
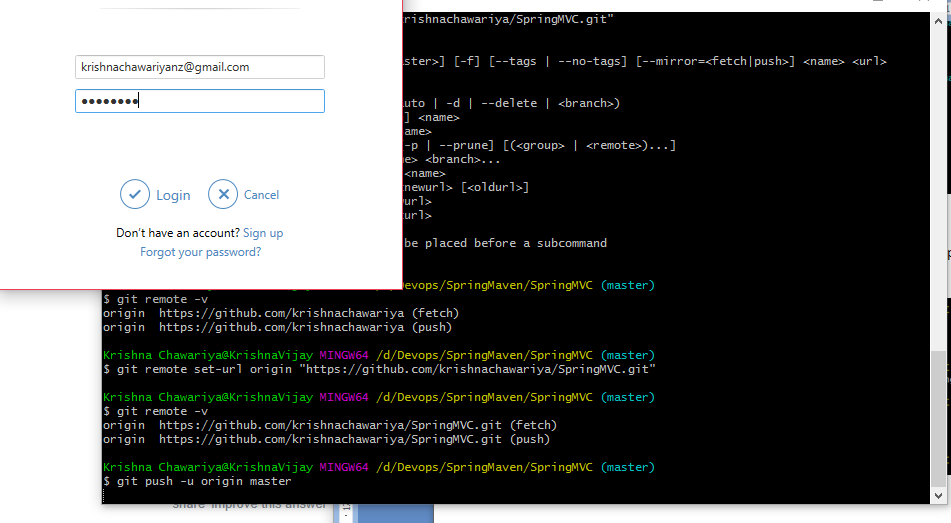
Add local repository to github:



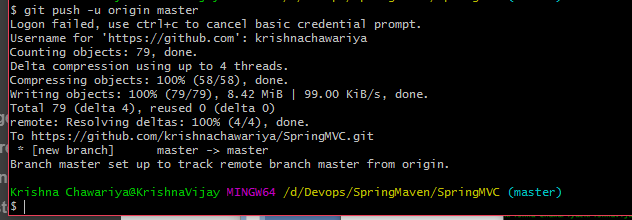
Map the local repo to remote repo

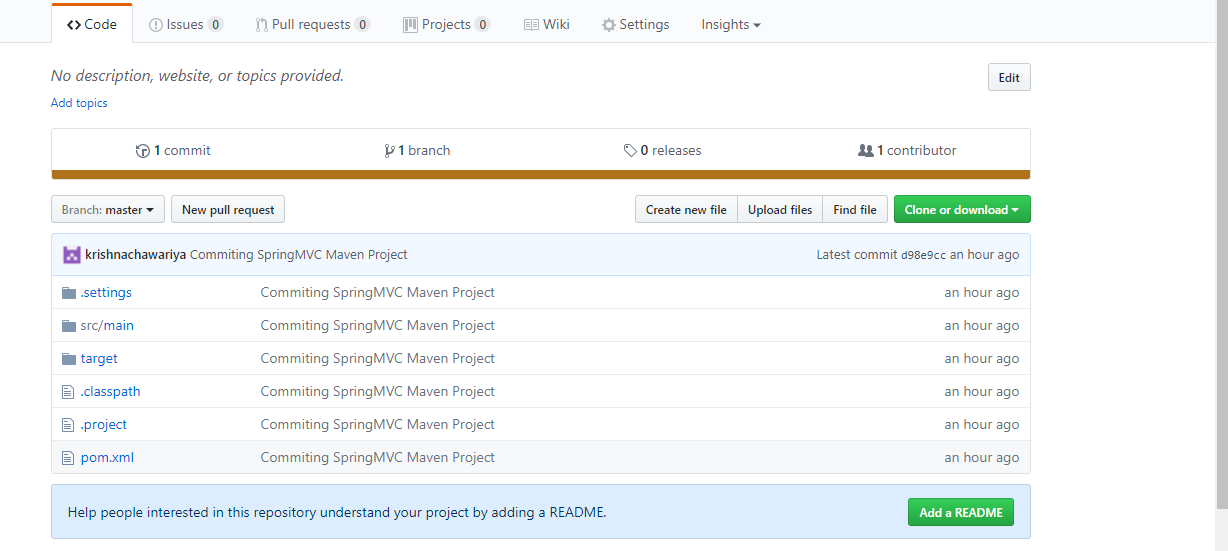
If we link our local repo by mistakenly to other remote repo, set the appropriate remote url using below command





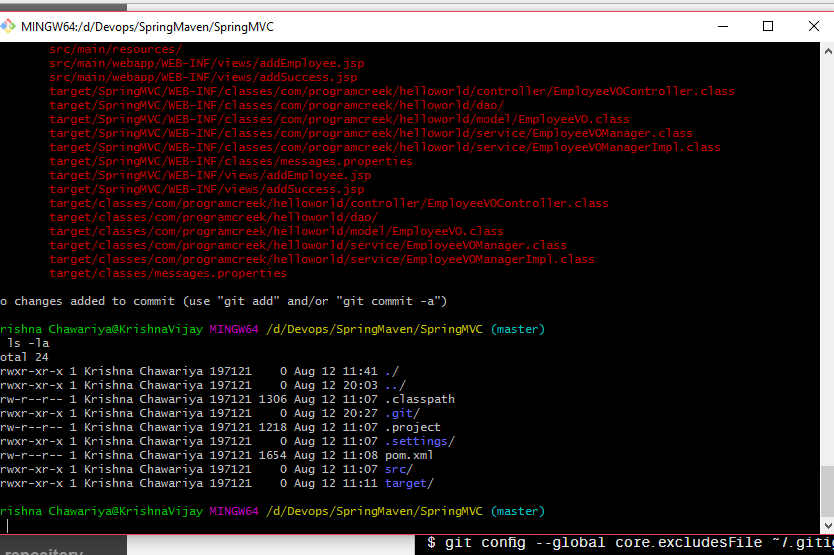
Once Push completed successfully

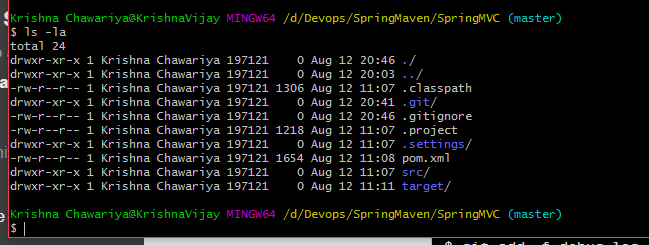


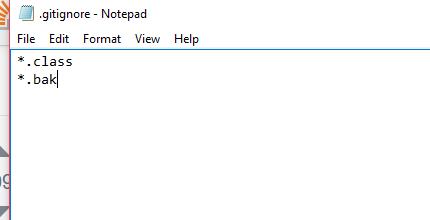


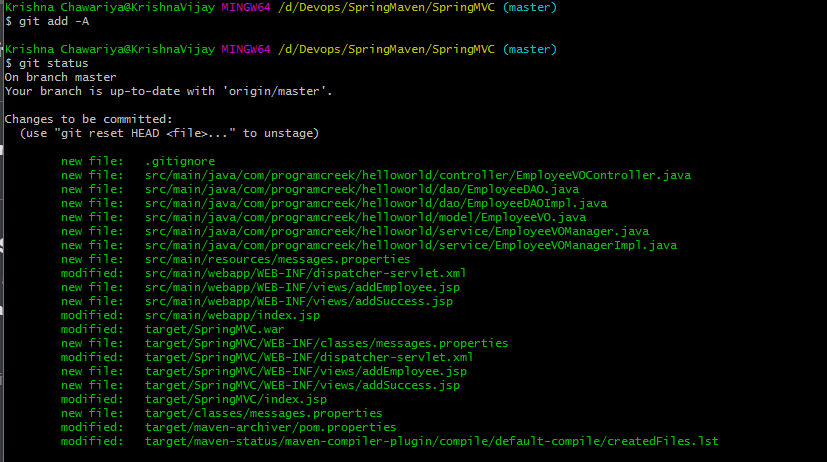
.gitIgnore

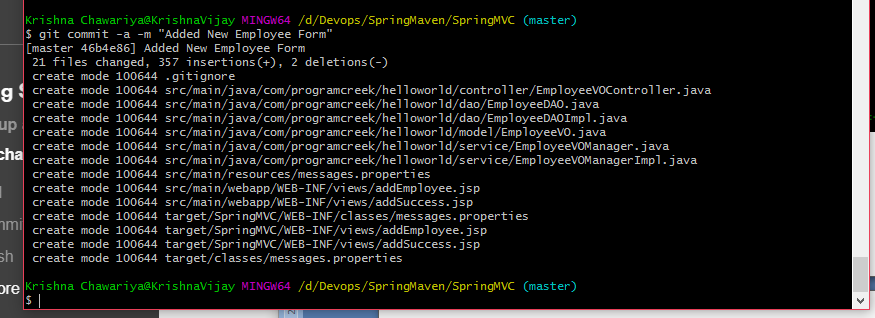
remove .class file

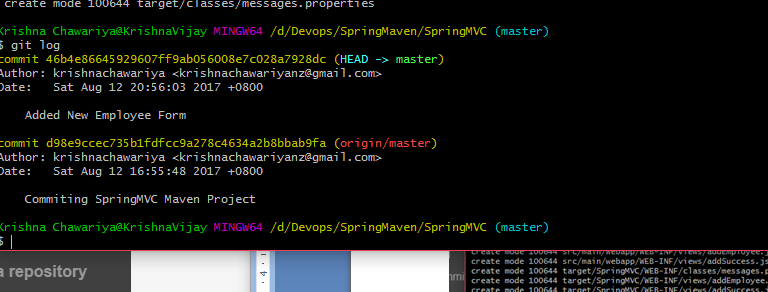




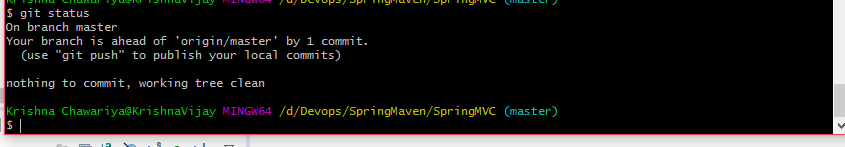


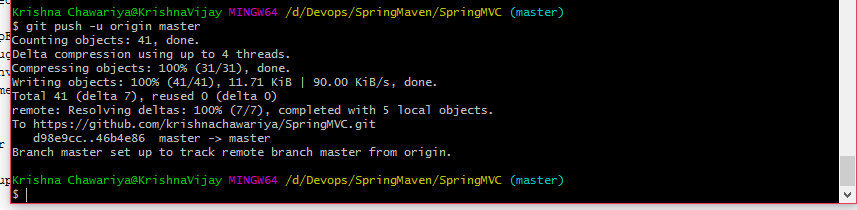


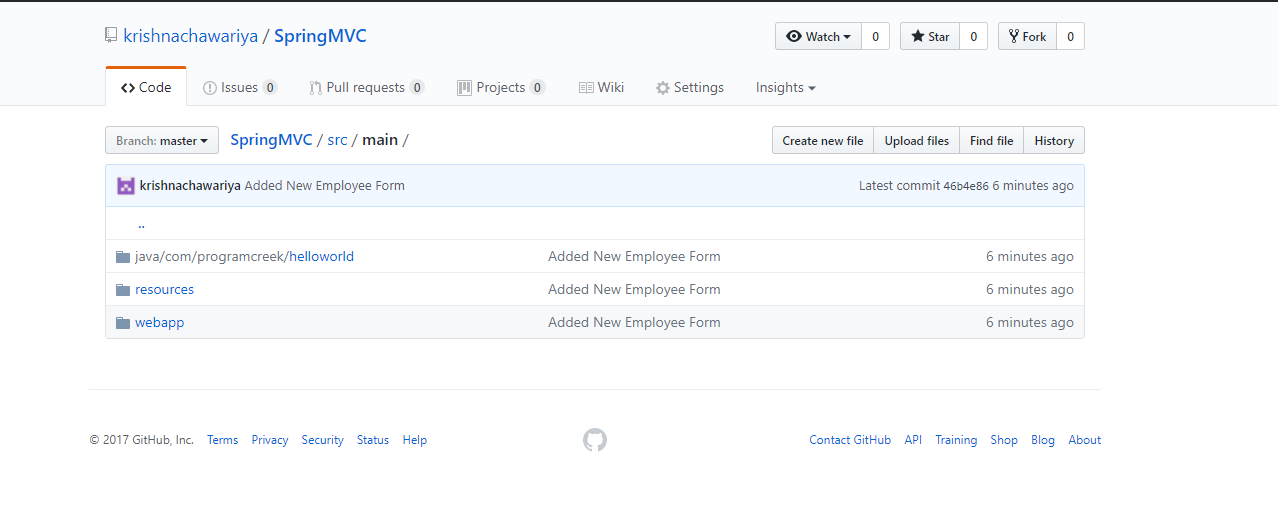




Before pushing changes to origin







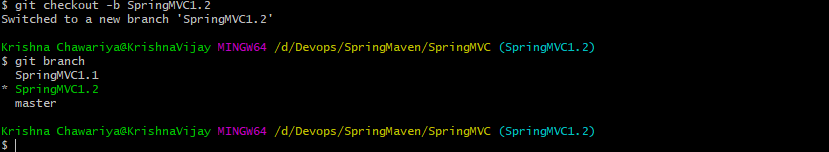
**Git Branching & Merging:**

**Created 2 branches under master in order to work parallel**

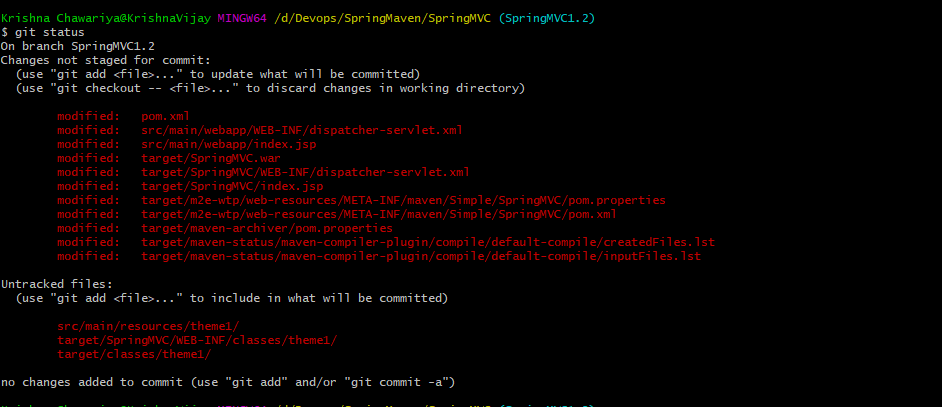
**syntax:**

**git branch <branchname>**

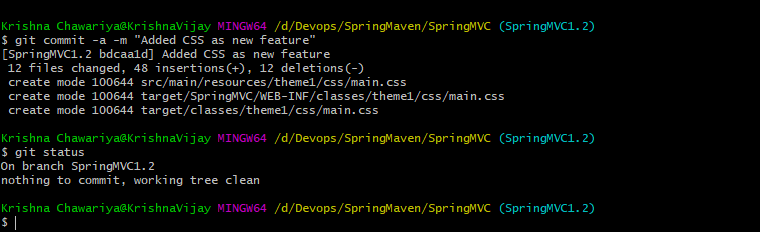
**git checkout -b <branchname>**

****

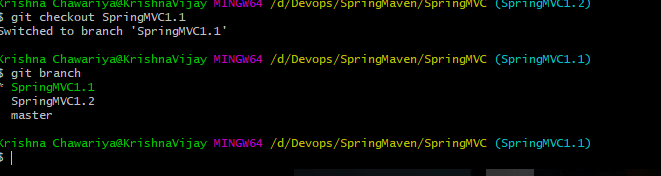
**1.2 contains modified and untracked files**

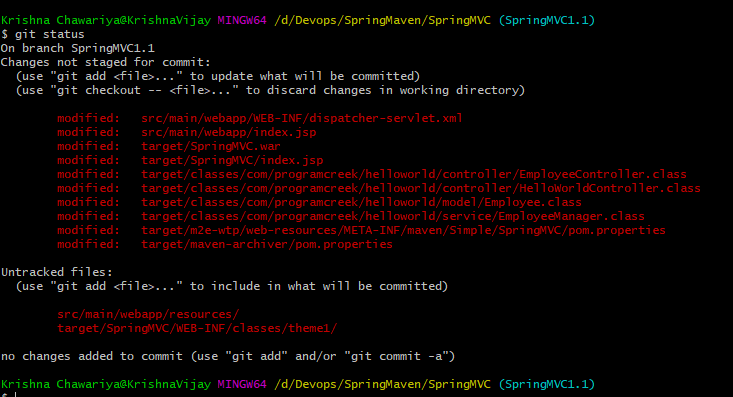
****

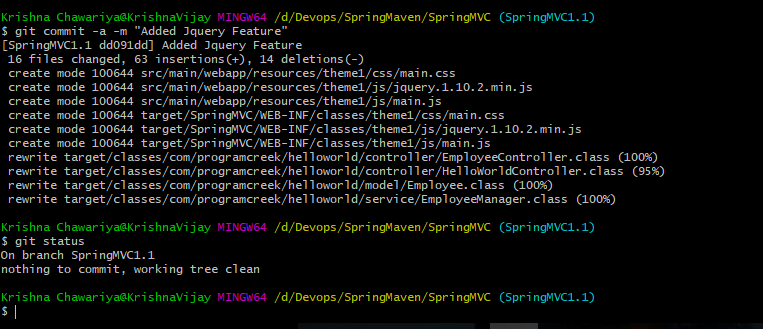
**Add and commit the files for 1.2 before switching to 1.1**

****

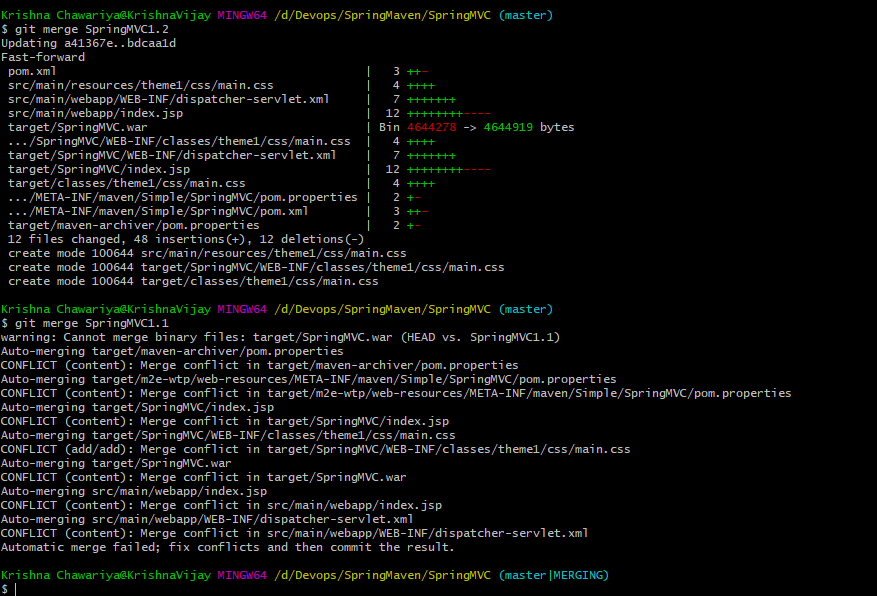
**Now switch to 1.1**

****

****

****

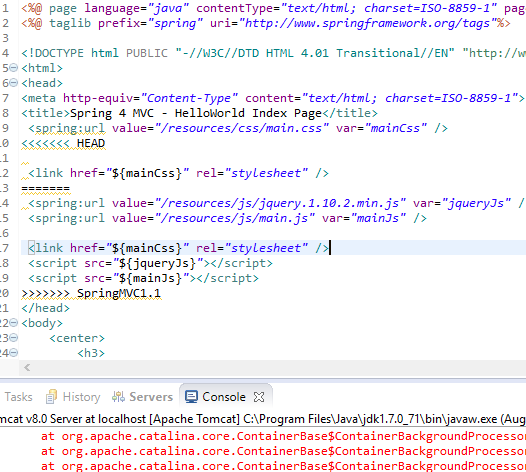
**Now checkout master, Merge 1.1 & 1.2 branches**

****

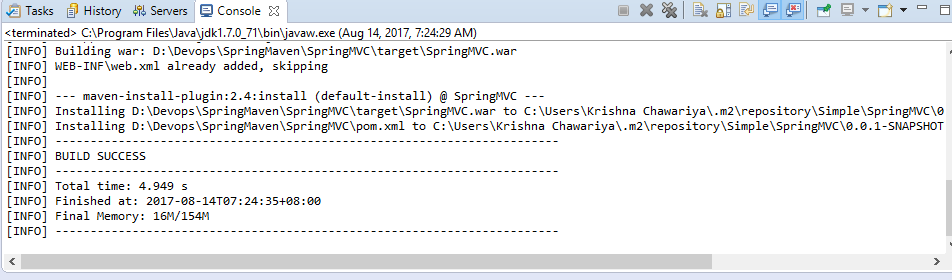
**Above screen shot, clearly shown the conflicts as i've performed the changes on common files**

**When you open the file in your text editor, you'll see the changes from the HEAD or base branch after the line <<<<<<< HEAD. Next, you'll see =======, which divides your changes from the changes in the other branch, followed by >>>>>>> BRANCH-NAME.**

**Decide if you want to keep only your branch's changes, keep only the other branch's changes, or make a brand new change, which may incorporate changes from both branches. Delete the conflict markers <<<<<<<, =======, >>>>>>> and make the changes you want in the final merge.**

****

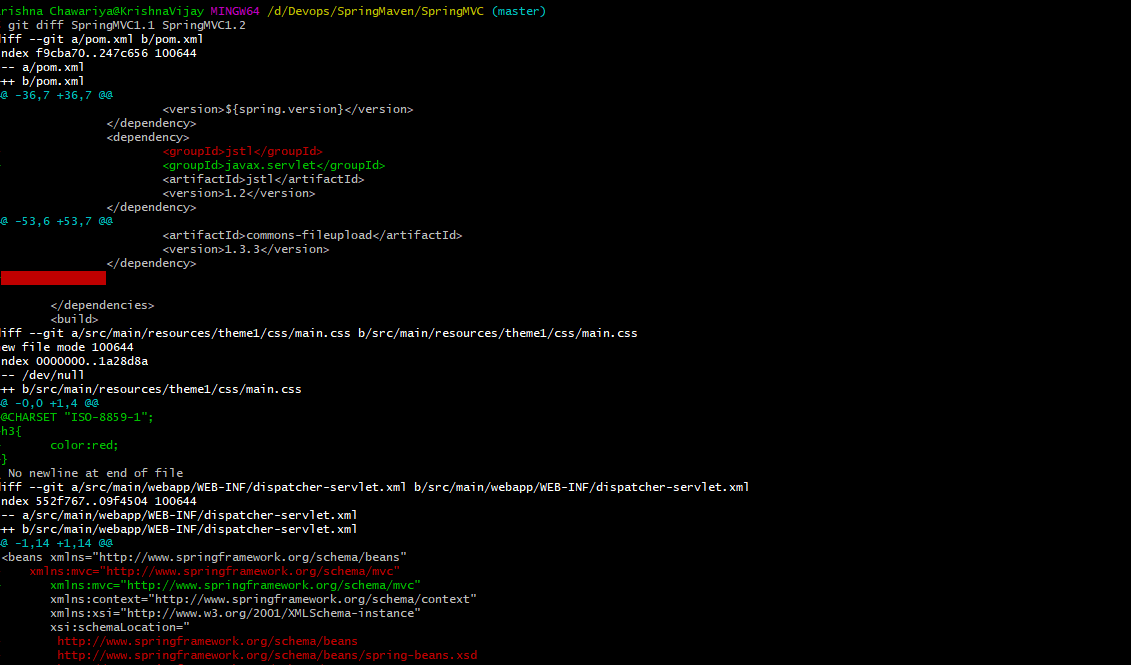
**Ensure build is success after resolving conflicts**

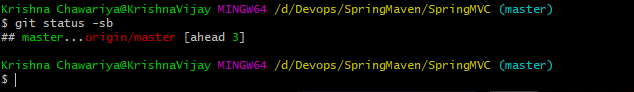
****

****

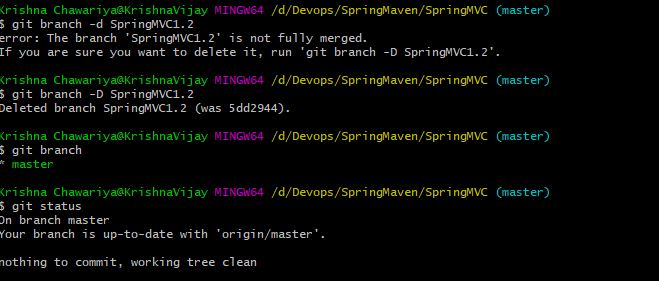
**Find the differences between two branches before merge using below command**

**git diff <branch1> <branch2>**

****

****

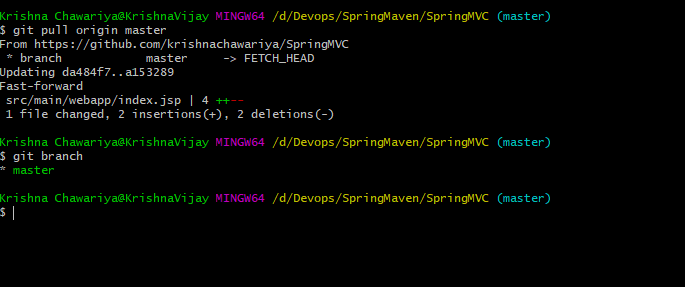
**Git delete brach:**

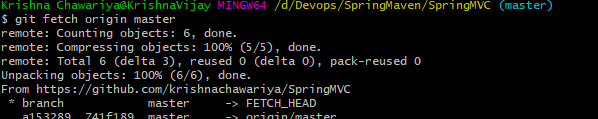
****

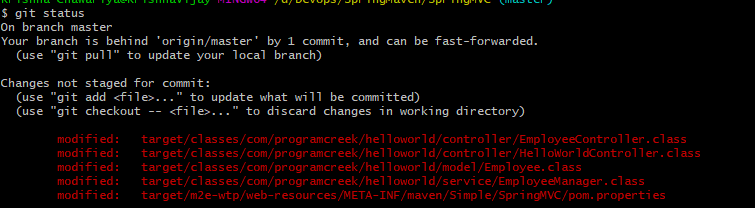
**Git Pull & Fetch:**

**in simplest terms git pull --> git fetch + git merge.**

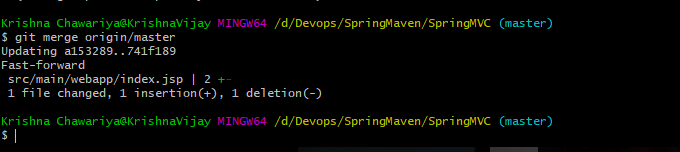
**Git Pull brings local branch upto date whereas fetch, Git gathers any commits from the target branch that do not exist in your current branch and**stores them in your local repository**. However,**it does not merge them with your current branch**. This is particularly useful if you need to keep your repository up to date, but are working on something that might break if you update your files. To integrate the commits into your master branch, you use merge.**

****

****

****

**In order to invoke changes of origin after fetch, need to merge with master**

****

**Git Tag:**

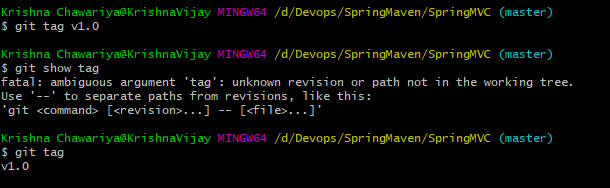
**Tag in Git refers to creating specific point s in history for repo. In common how we mark the release points v1.0, v1.1**

**checkout a branch where you want to create a tag.**

**Tags are 2 types:**

* **Light Weight Tag**
* **Annotated Tag**

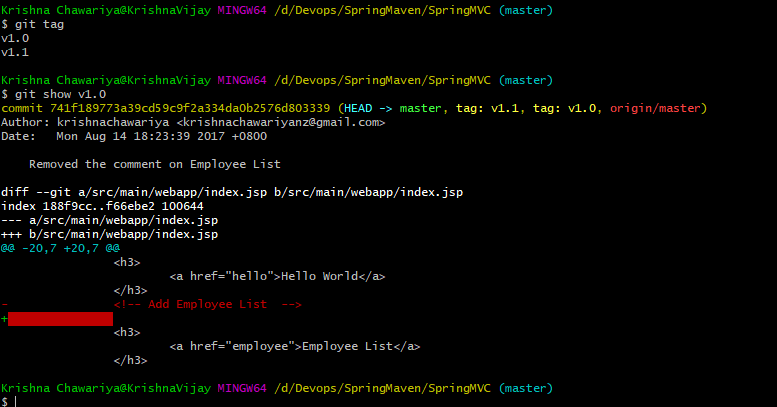
**In annotated tag, we can provide message which gives information about the tag**

****

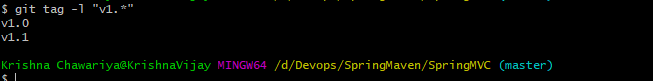
**Annotated Tag**

****

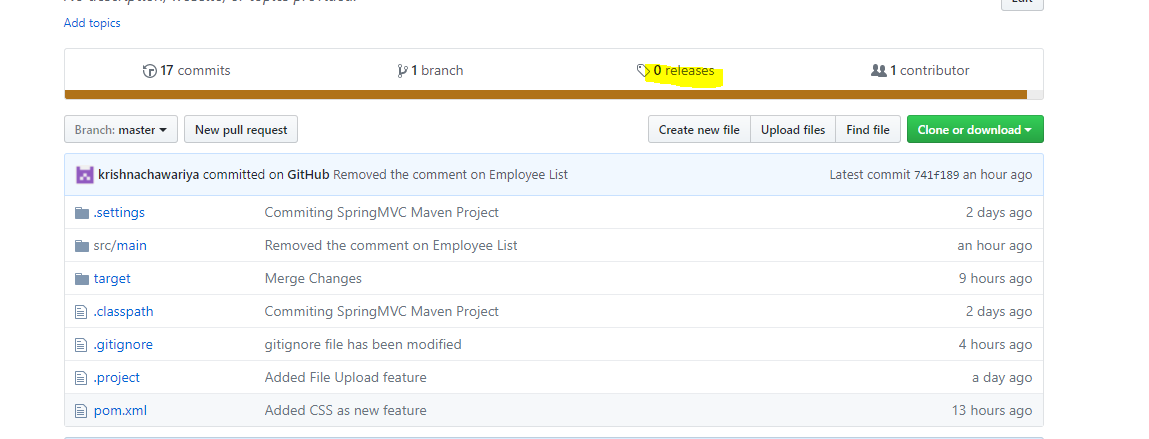
**Show/Display Tags**

****

**By default Tag created on the last commit on the bracnh**

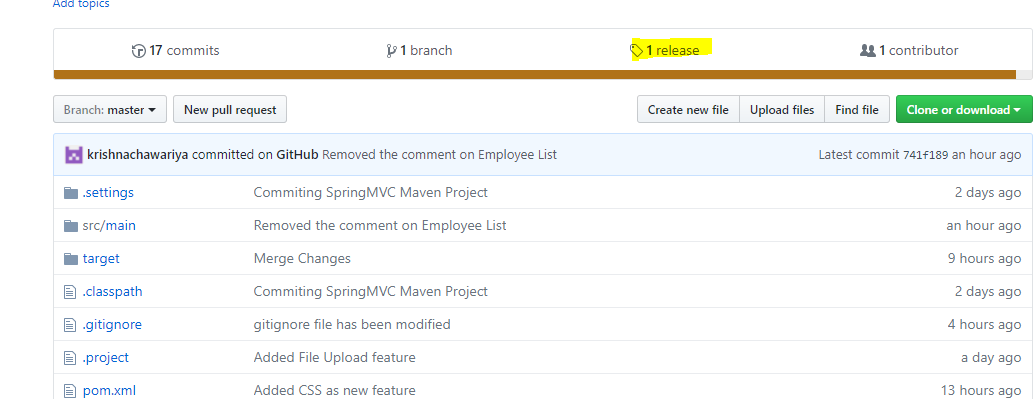
****

**On my sample project have 0 releases**

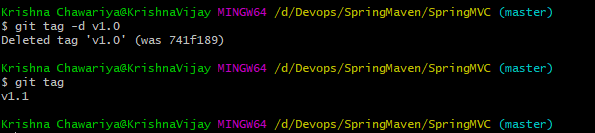
****

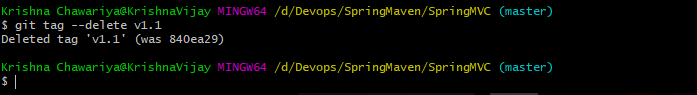
**Push Tags to remote**

****

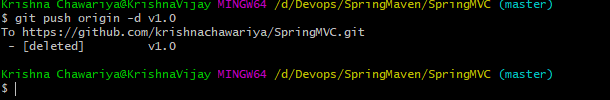
****

**Delete Tag:**

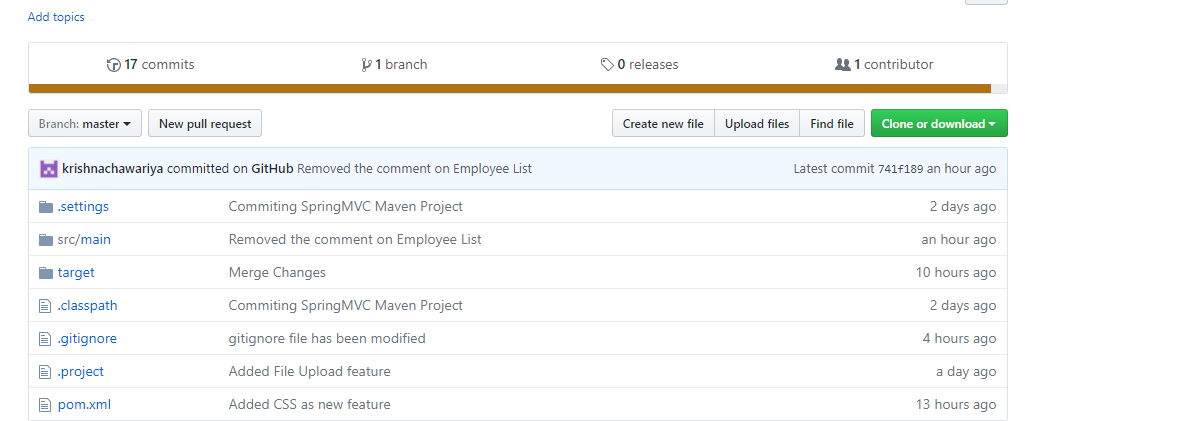
****

****

**But still tags would exist in remote repo. To delete remote tags**

****

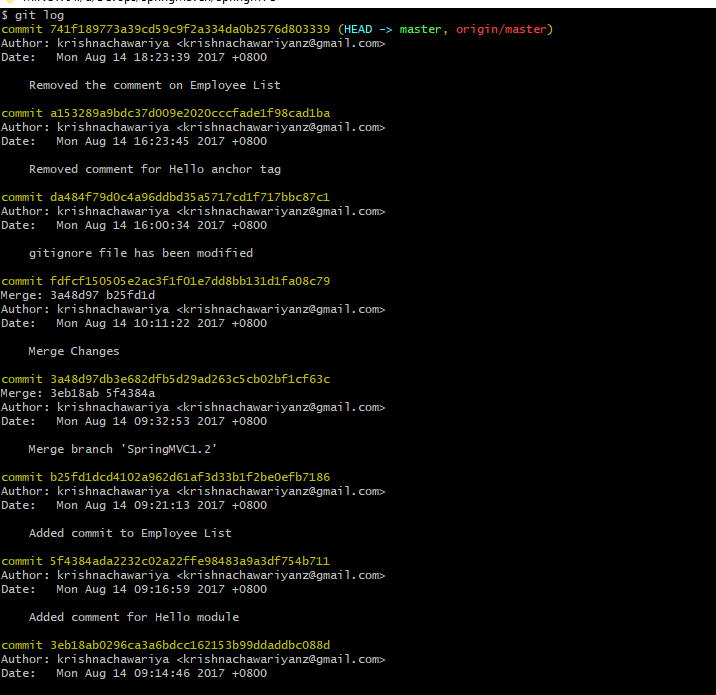
**Now, don't have any tags under my central repo**

****

**In real scenario, will create a tag on the commit where the repository is stable and push the tag to remote server.**

**After that historic point of commit, continue with other commits and create another tag where out next release goes.**

**Every commit generates a unique 40 digit checksum number generated by using SHA algorithm.**

****

**We can't checkout Tags like how we checkout branches. So what we do**

**we create a branch from the tag and checkout the branch.**