Git

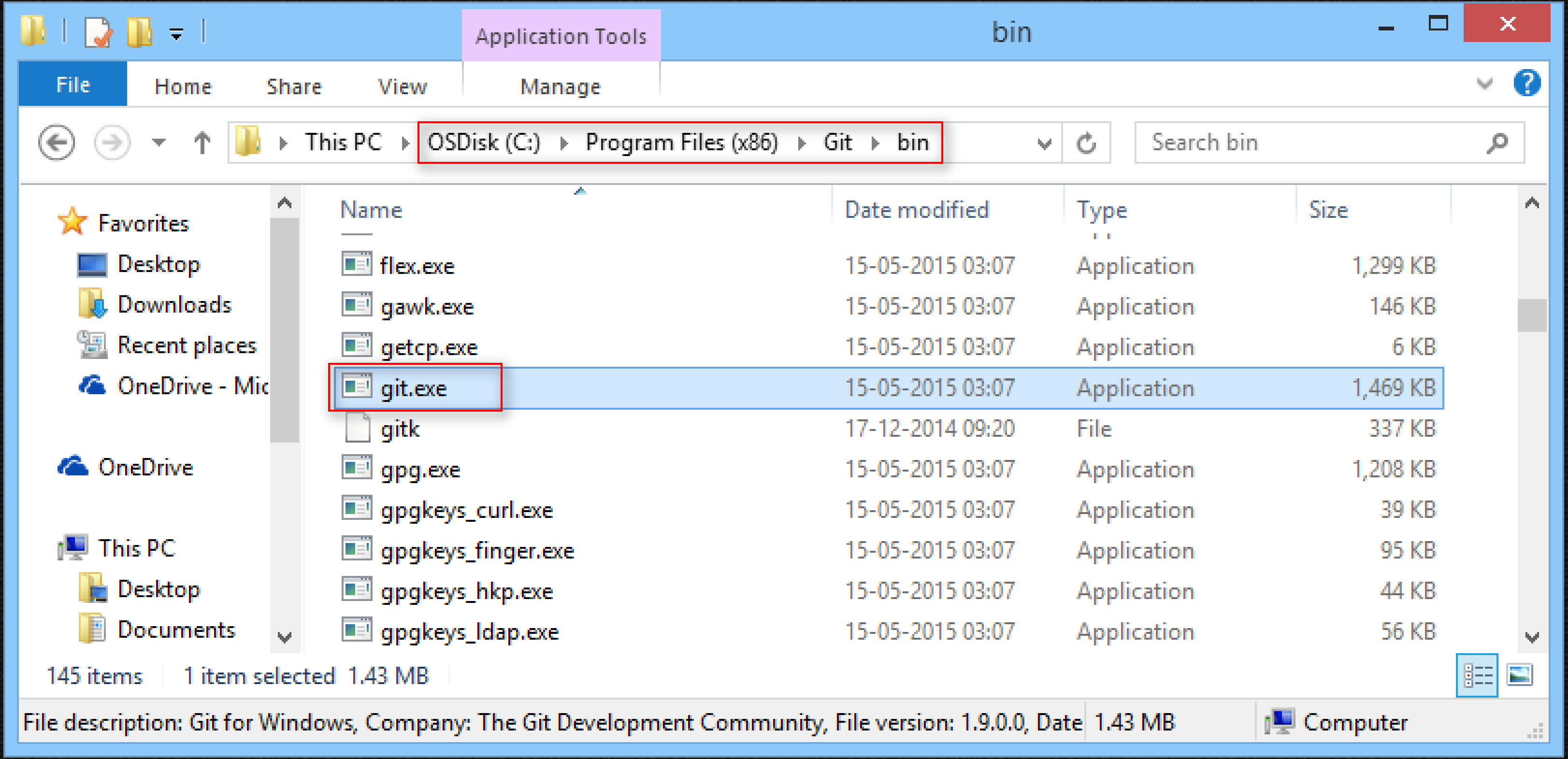
Vineel

Agenda

- Git Local
  - Git Installation
  - Git Vs CVS/SVN/SD/P4
  - How to initialize a git repository?
  - Git Basics – status, add, commit
  - Git Ignore
  - Git Configuration
  - Git HEAD
  - Git log
  - Git diff
  - Undo changes in Git
  - Git stash
  - Branching
  - Merging
  - Git rebase
- Git Remote
  - Git clone
  - Git fetch
  - Git pull
  - Git push
- Simple Daily Workflow
- Git Tips & Tricks
- Repo tool

Git Installation

- Git for Windows *https://git-scm.com/*
- Git will be installed to *C:\Program Files (x86)\Git*
- For teams working with OneBranch, Git is automatically installed via chocolatey
- Git is completely command line driven(99.99%)



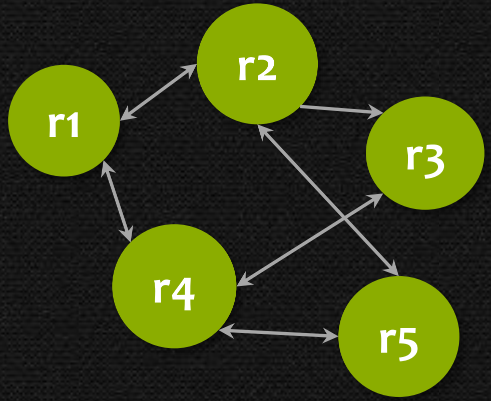
Name	Date modified	Type	Size
flex.exe	15-05-2015 03:07	Application	1,299 KB
gawk.exe	15-05-2015 03:07	Application	146 KB
getcp.exe	15-05-2015 03:07	Application	6 KB
git.exe	15-05-2015 03:07	Application	1,469 KB
gitk	17-12-2014 09:20	File	337 KB
gpg.exe	15-05-2015 03:07	Application	1,208 KB
gpgkeys_curl.exe	15-05-2015 03:07	Application	39 KB
gpgkeys_finger.exe	15-05-2015 03:07	Application	95 KB
gpgkeys_hkp.exe	15-05-2015 03:07	Application	44 KB
gpgkeys_ldap.exe	15-05-2015 03:07	Application	56 KB

145 items   1 item selected 1.43 MB

File description: Git for Windows, Company: The Git Development Community, File version: 1.9.0.0, Date 1.43 MB   Computer

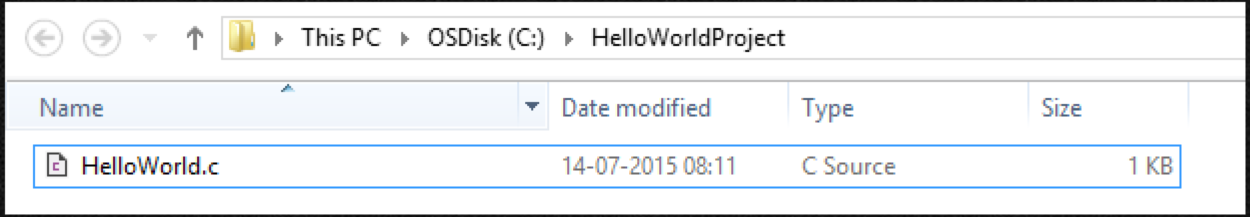
Git Vs CVS/SVN/SD/P4

- No client and server model
- No central remote database
- Think of Git as an intelligent *Local* File System to keep track of your source code
- All metadata related to the repository is stored in *.git* directory in root of the repository
  - This makes duplicating projects or transferring repositories super easy – Just *xcopy* the repository
- Git repositories communicate with each other via *push* and *pull* model – This make it a *distributed version control system*
- This simplified model has quite interesting implications



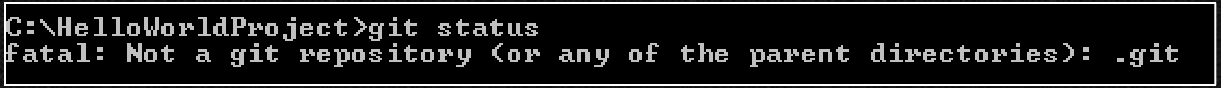
How to initialize a git repository?

- *git init .*



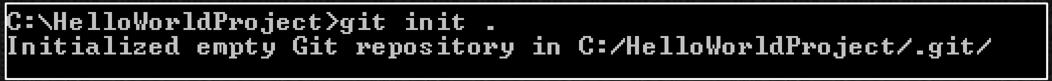
This PC › OSDisk (C:) › HelloWorldProject
Name | Date modified | Type | Size

Name	Date modified	Type	Size
HelloWorld.c	14-07-2015 08:11	C Source	1 KB

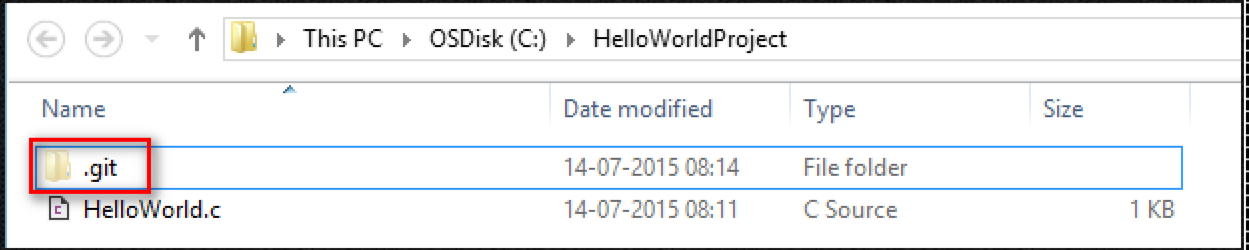


```
C:\HelloWorldProject>git status
fatal: Not a git repository (or any of the parent directories): .git
```

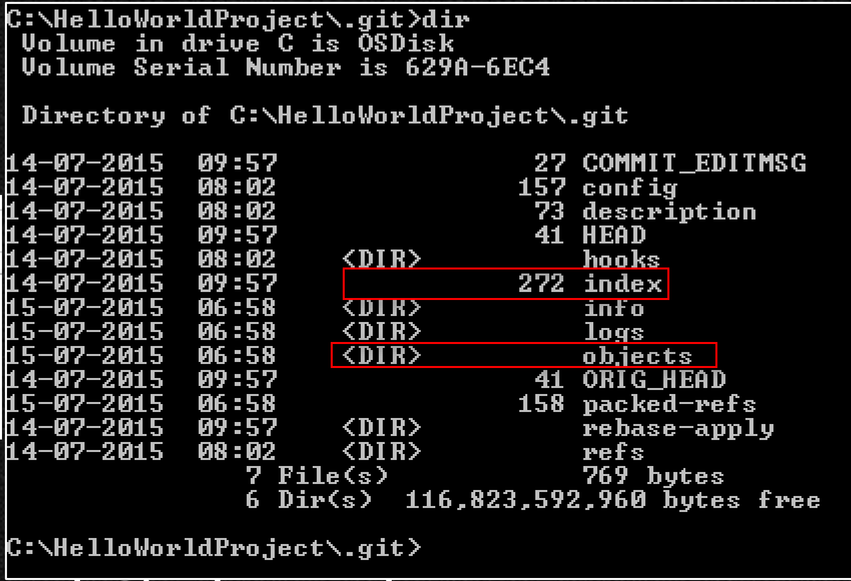
git init .



```
C:\HelloWorldProject>git init .
Initialized empty Git repository in C:/HelloWorldProject/.git/
```



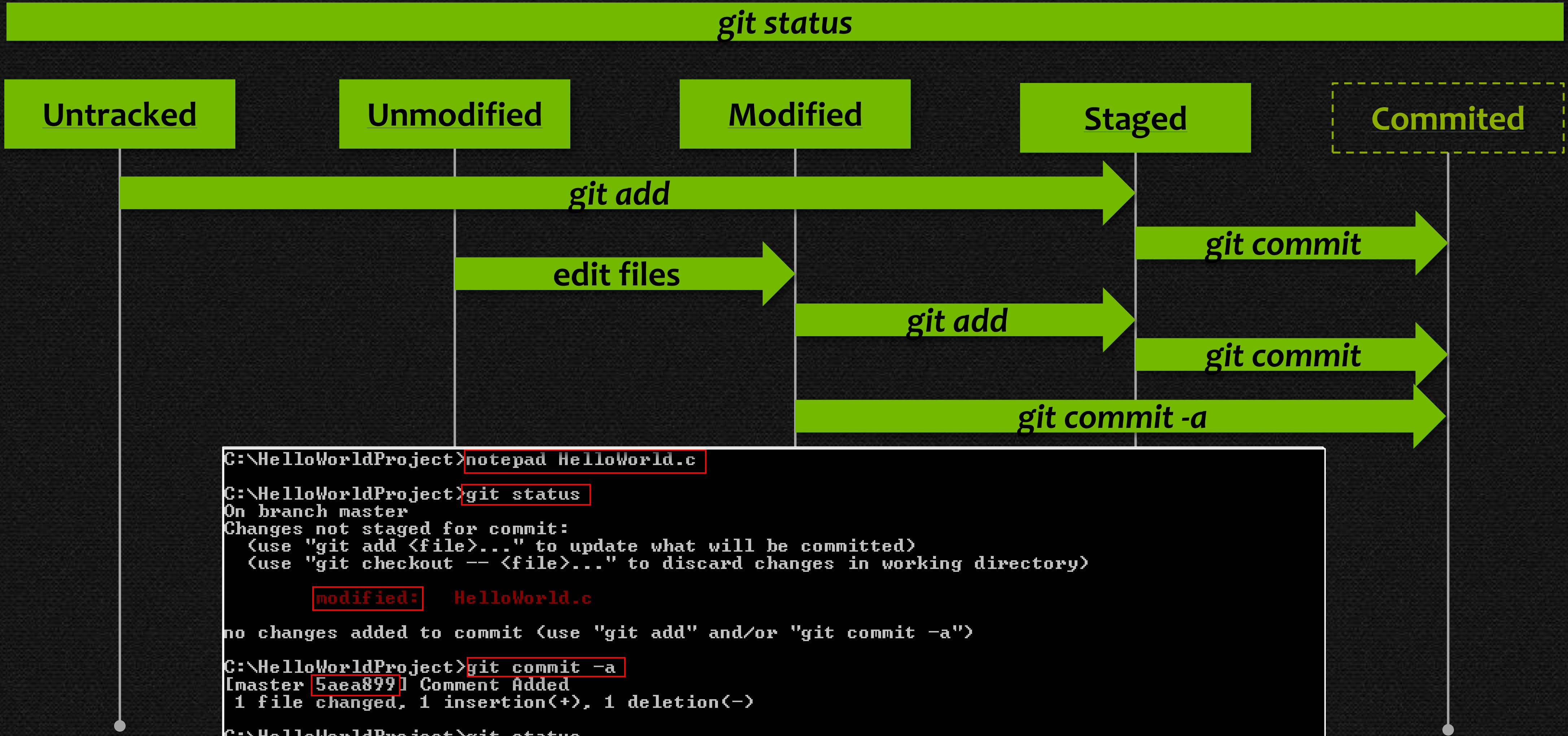
Name	Date modified	Type	Size
.git	14-07-2015 08:14	File folder	
HelloWorld.c	14-07-2015 08:11	C Source	1 KB

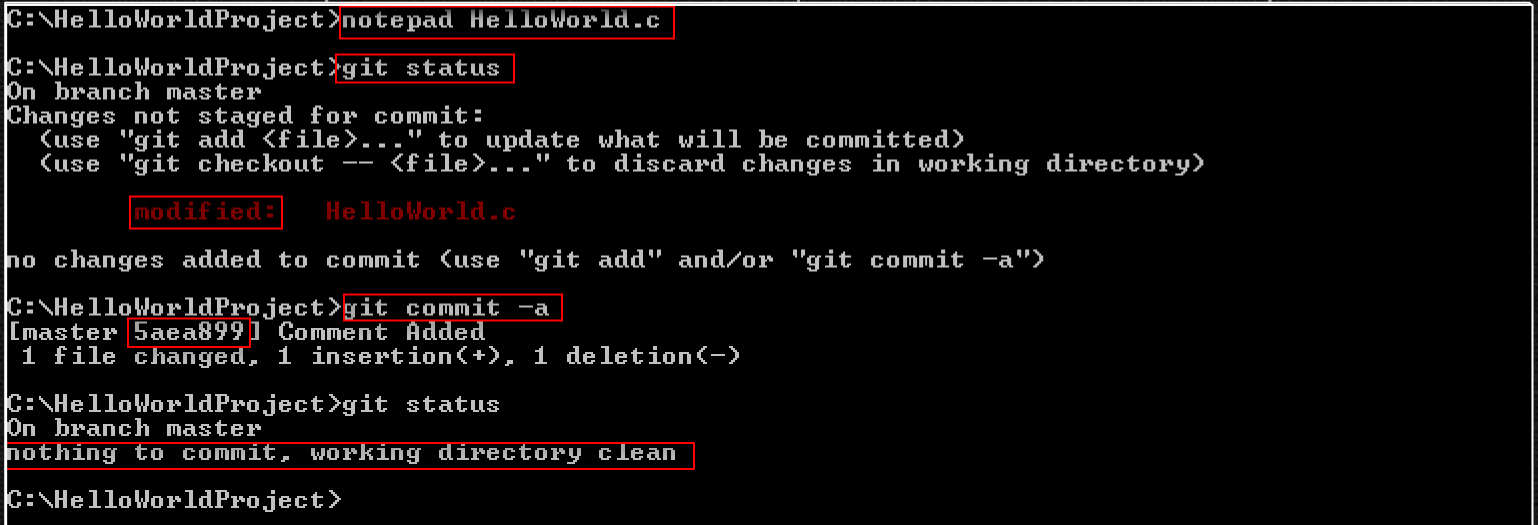


```
C:\HelloWorldProject\.git>dir
 Volume in drive C is OSDisk
 Volume Serial Number is 629A-6EC4

 Directory of C:\HelloWorldProject\.git

14-07-2015  09:57                27 COMMIT_EDITMSG
14-07-2015  08:02               157 config
14-07-2015  08:02                73 description
14-07-2015  09:57                41 HEAD
14-07-2015  08:02    <DIR>          hooks
14-07-2015  09:57               272 index
15-07-2015  06:58    <DIR>          info
15-07-2015  06:58    <DIR>          logs
15-07-2015  06:58    <DIR>          objects
14-07-2015  09:57                41 ORIG_HEAD
15-07-2015  06:58               158 packed-refs
14-07-2015  09:57    <DIR>          rebase-apply
14-07-2015  08:02    <DIR>          refs
               7 File(s)            769 bytes
               6 Dir(s)  116,823,592,960 bytes free

C:\HelloWorldProject\.git>
```

- What does .git folder contain?
- Unlike SD, All files are writable by default in Git and Git do not have "*sd edit*" step before editing files

Git Basics

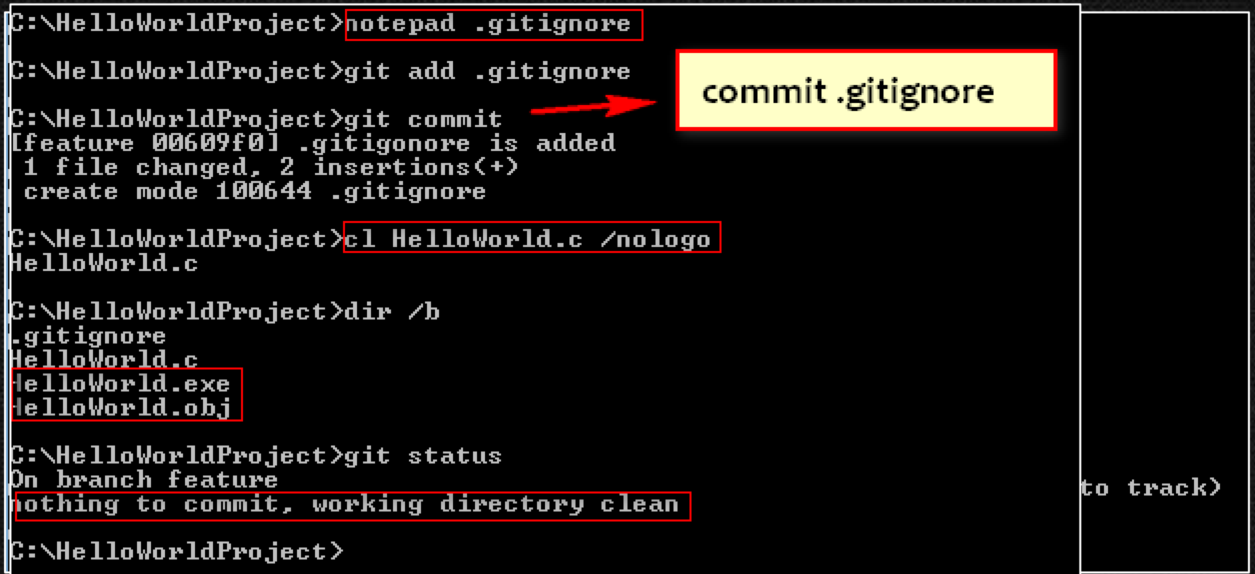




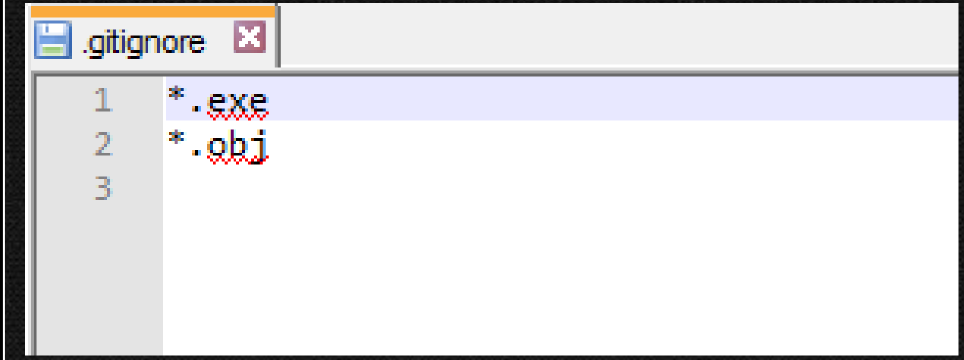
```
C:\HelloWorldProject>notepad HelloWorld.c

C:\HelloWorldProject>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   HelloWorld.c

no changes added to commit (use "git add" and/or "git commit -a")

C:\HelloWorldProject>git commit -a
[master 5aea899] Comment Added
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\HelloWorldProject>git status
On branch master
nothing to commit, working directory clean

C:\HelloWorldProject>
```

Git Ignore

- Some times you may want to tell git not to track any binary files or some temp files or folders inside your project
- *.gitignore* file in root directory contains patterns of file names to ignore
- Matching of files happen from root directory
  - *.exe   #ignore any exe file in the root directory
  - out/*.log   #ignore any logs files in out folder of root directory
  - obj/   #ignore obj folder in the root directory



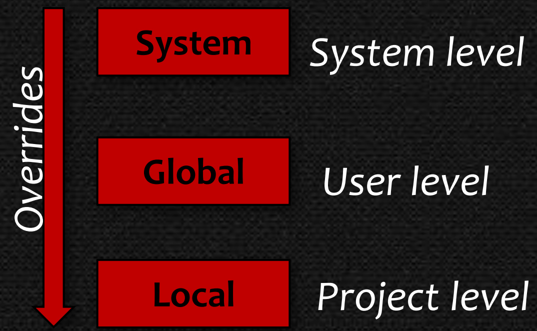
```
C:\HelloWorldProject>notepad .gitignore

C:\HelloWorldProject>git add .gitignore

C:\HelloWorldProject>git commit
[feature 00609f0] .gitignore is added
 1 file changed, 2 insertions(+)
 create mode 100644 .gitignore

C:\HelloWorldProject>cl HelloWorld.c /nologo
HelloWorld.c

C:\HelloWorldProject>dir /b
.gitignore
HelloWorld.c
HelloWorld.exe
HelloWorld.obj

C:\HelloWorldProject>git status
On branch feature
nothing to commit, working directory clean

C:\HelloWorldProject>
```

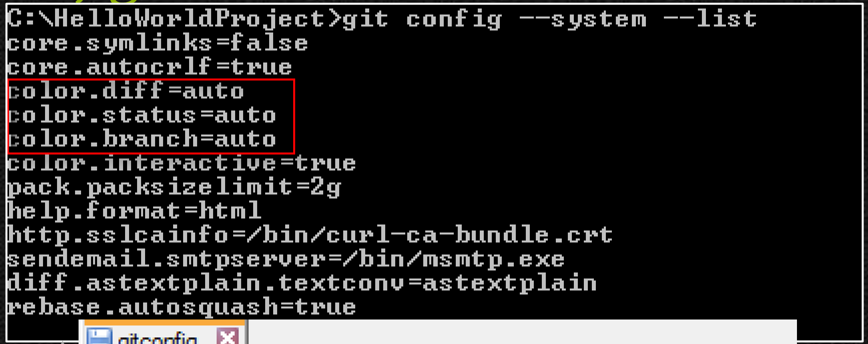


```
*.exe
*.obj
```

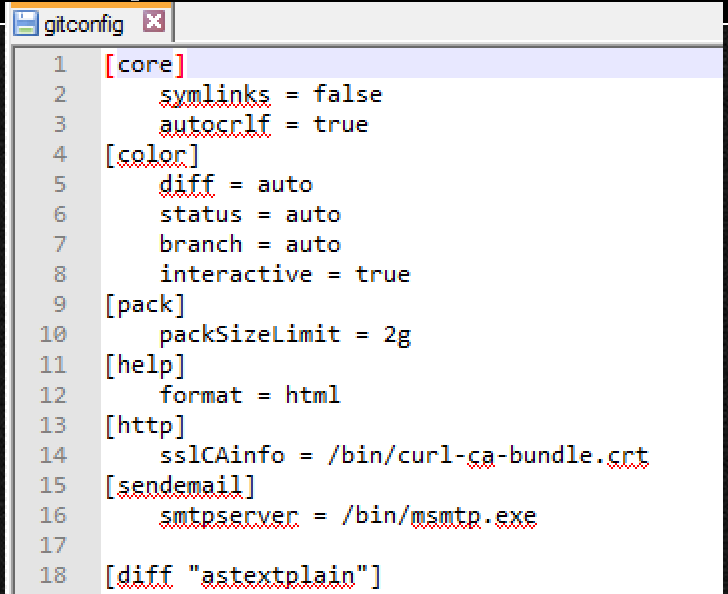
Git Configuration

- How to customize git configuration via .gitconfig
- Level of git configuration
  - System Level – *C:\Program Files (x86)\Git\gitconfig* file
  - User Level - *%USERPROFILE%\.gitconfig* file
  - Project Level - *.git\config* file
- *Setting config options*
  - *git config --system user.name “Vineel K”*
  - *git config --global user.name “Vineel K”*
  - *git config --local user.name “Vineel K”*
- *Reading config options*
  - *git config –list*
  - *git config --system --list*
  - *git config --global --list*
  - *git config --local --list*





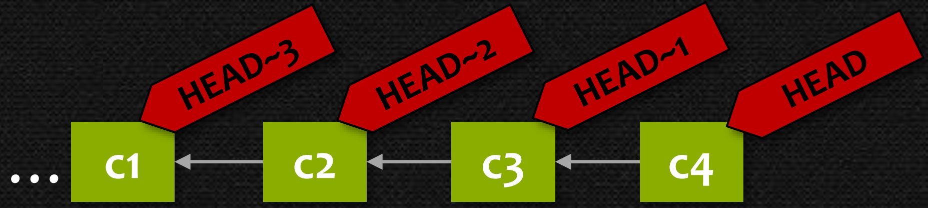
```
C:\HelloWorldProject>git config --system --list
core.symlinks=false
core.autocrlf=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
pack.packsizelimit=2g
help.format=html
http.sslcainfo=/bin/curl-ca-bundle.crt
sendemail.smtpserver=/bin/msmtp.exe
diff.astextplain.textconv=astextplain
rebase.autosquash=true
```



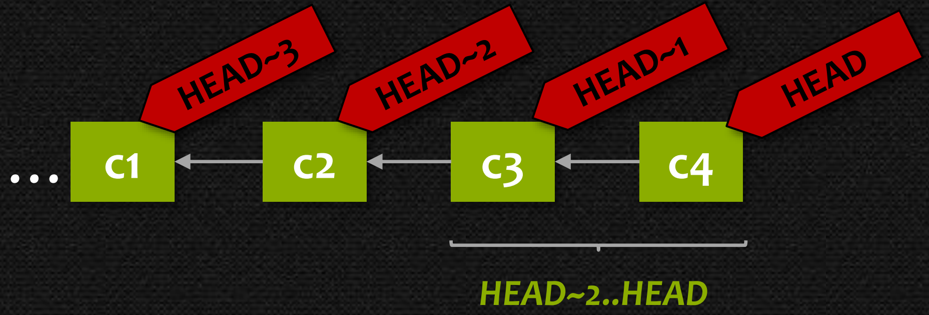
```
[core]
    symlinks = false
    autocrlf = true
[color]
    diff = auto
    status = auto
    branch = auto
    interactive = true
[pack]
    packSizeLimit = 2g
[help]
    format = html
[http]
    sslCAinfo = /bin/curl-ca-bundle.crt
[sendemail]
    smtpserver = /bin/msmtp.exe

[diff "astextplain"]
```

Git HEAD

- HEAD always refers to the latest commit on the current branch
- HEAD~1 always refers to the commit one before the latest commit
- HEAD~2, HEAD~3, …

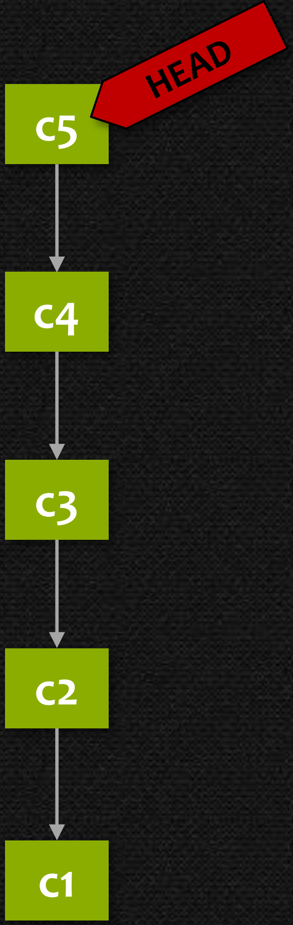


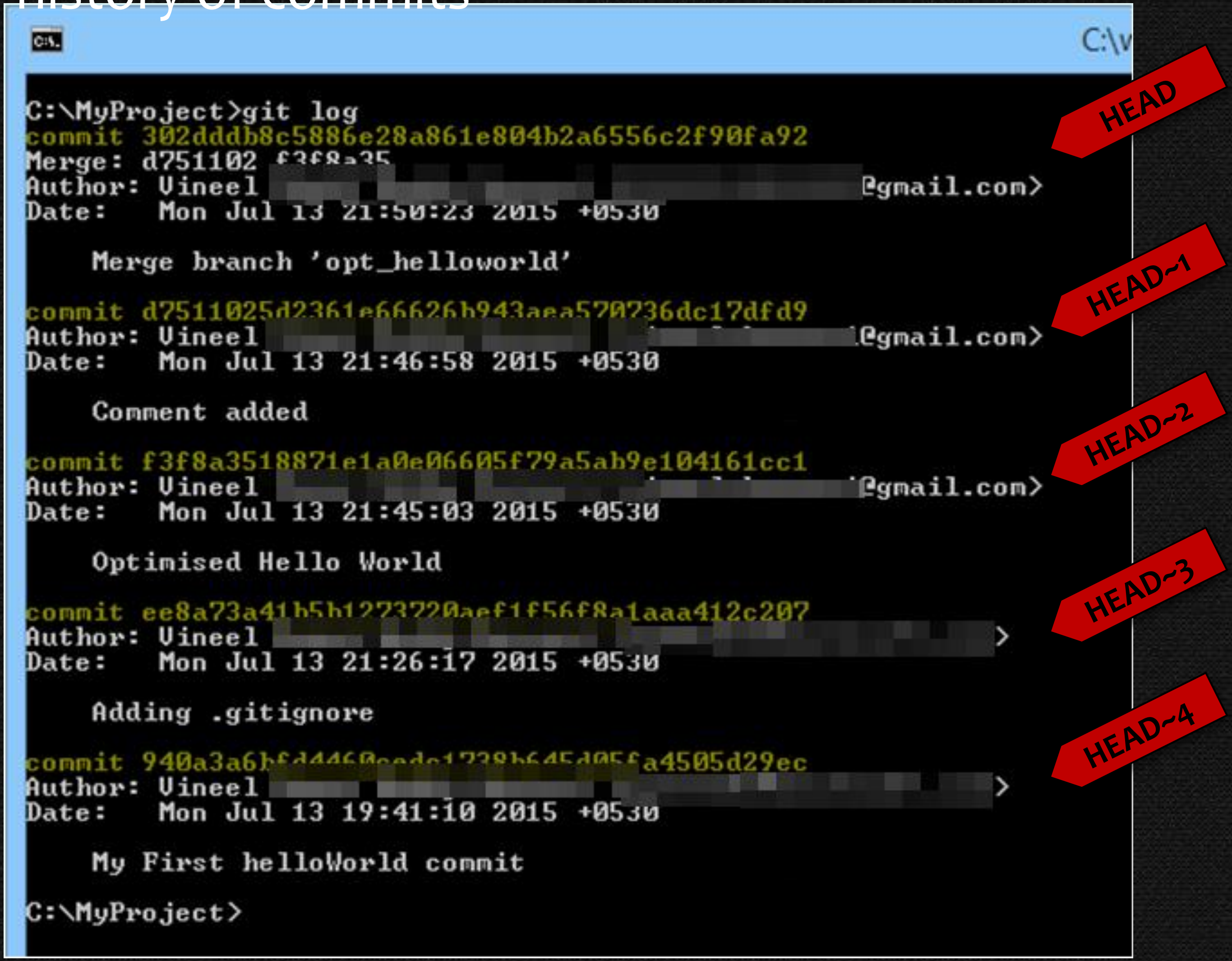
- .. Syntax(revision/range syntax) is used to refer a range of commits
- *HEAD~2..HEAD* means all commit b/w HEAD~2 and HEAD not including HEAD~2



Git log

- *git log* show history of commits

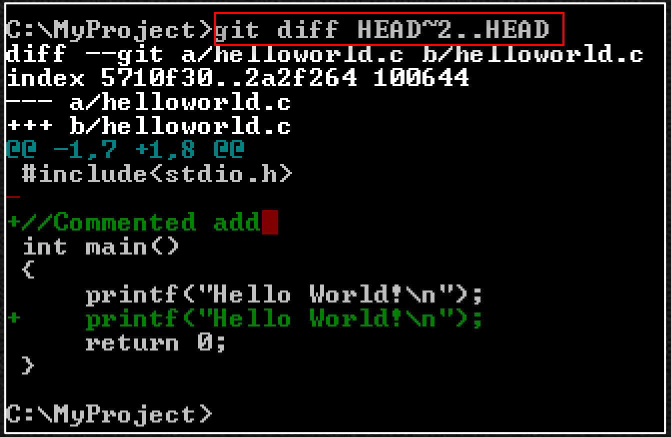




```
C:\MyProject>git log
commit 302dddb8c5886e28a861e804b2a6556c2f90fa92
Merge: d751102 f3f8a35
Author: Vineel [redacted]@gmail.com>
Date:   Mon Jul 13 21:50:23 2015 +0530

    Merge branch 'opt_helloworld'

commit d7511025d2361e66626b943aea570736dc17dfd9
Author: Vineel [redacted]@gmail.com>
Date:   Mon Jul 13 21:46:58 2015 +0530

    Comment added

commit f3f8a3518871e1a0e06605f79a5ab9e104161cc1
Author: Vineel [redacted]@gmail.com>
Date:   Mon Jul 13 21:45:03 2015 +0530

    Optimised Hello World

commit ee8a73a41b5b1273720aef1f56f8a1aaa412c207
Author: Vineel [redacted]>
Date:   Mon Jul 13 21:26:17 2015 +0530

    Adding .gitignore

commit 940a3a6bfd4460eede1738b645d05fa4505d29ec
Author: Vineel [redacted]>
Date:   Mon Jul 13 19:41:10 2015 +0530

    My First helloWorld commit

C:\MyProject>
```

Git diff

- *git diff* command is used to know the changes made to files between commits
- rev syntax can be used to specify range of commits to diff
  - *git diff HEAD~2..HEAD*



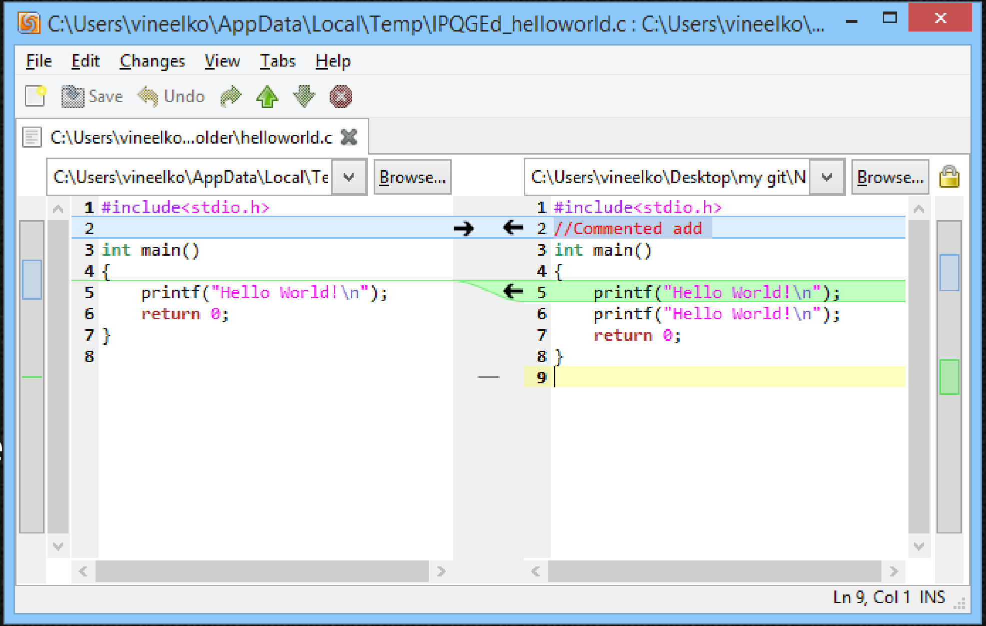
```
C:\MyProject>git diff HEAD~2..HEAD
diff --git a/helloworld.c b/helloworld.c
index 5710f30..2a2f264 100644
--- a/helloworld.c
+++ b/helloworld.c
@@ -1,7 +1,8 @@
 #include<stdio.h>
-
+//Commented add
 int main()
 {
     printf("Hello World!\n");
+    printf("Hello World!\n");
     return 0;
 }

C:\MyProject>
```

Git difftool

- *git difftool* is equivalent to running windiff in sd
- Lets give windiff a facelift with meld ☺
- http://sourceforge.net/projects/meld-installer/
- Copy below lines to your *%USERPROFILE%/.gitconfig*

```
[diff]
    tool = meld
[difftool "meld"]
    path = c:/Program Files (x86)/meld/meld/meld.exe
[difftool]
        prompt = false
```

- *git difftool HEAD~2..HEAD*



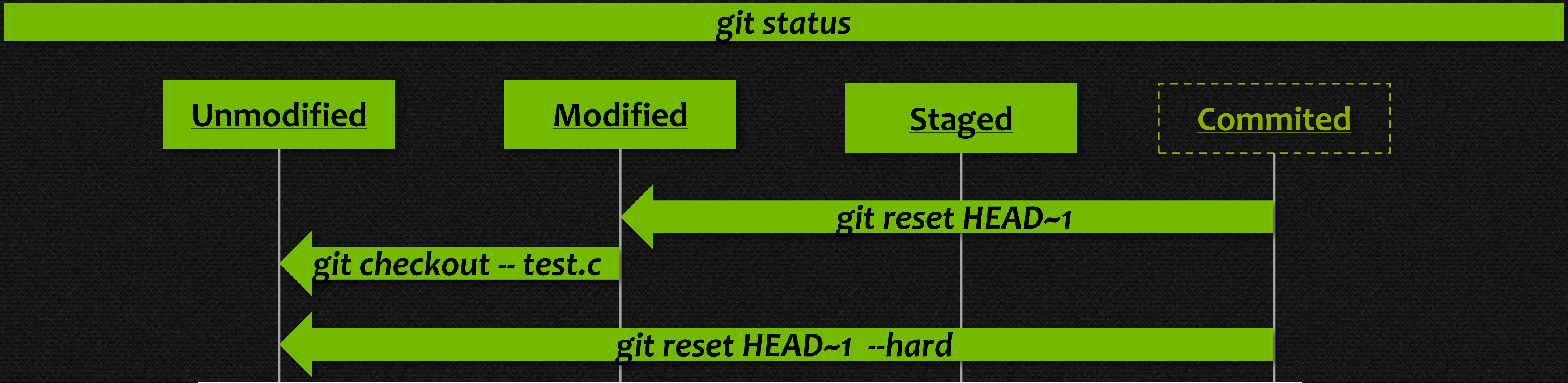
C:\Users\vineelko\AppData\Local\Temp\IPQGEd_helloworld.c : C:\Users\vineelko\...

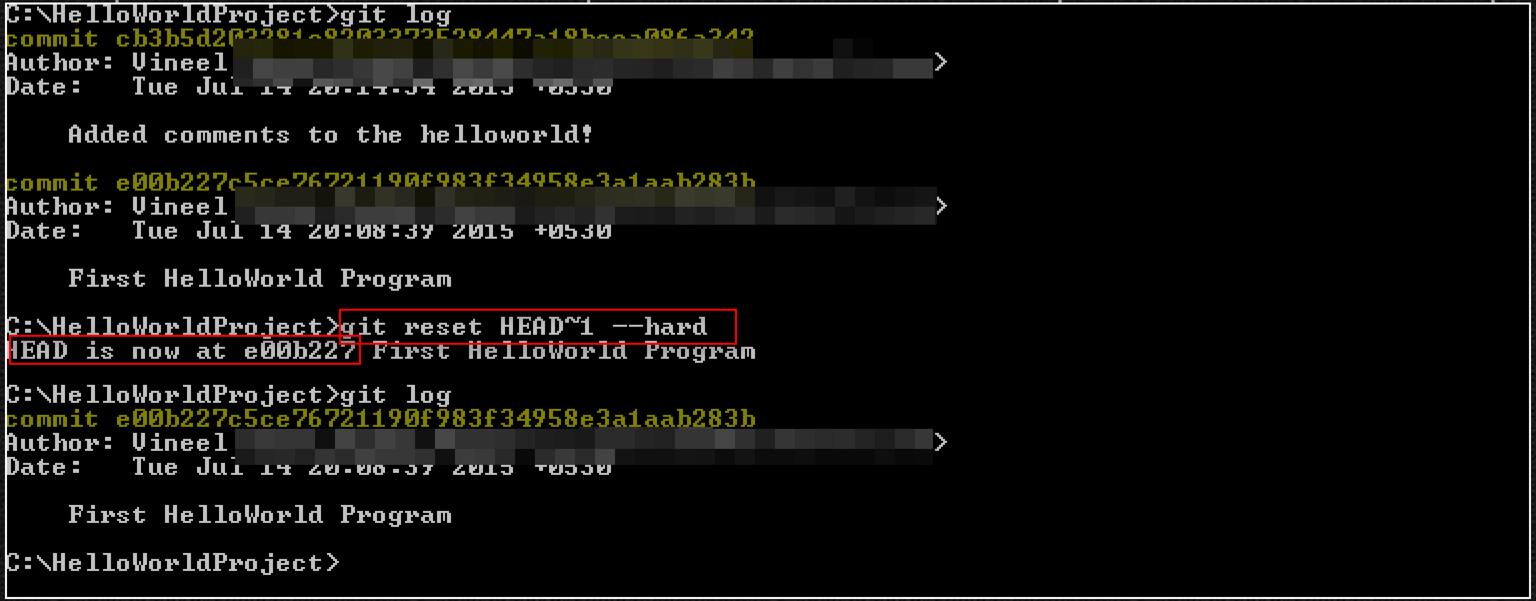
File Edit Changes View Tabs Help
Save Undo
C:\Users\vineelko...older\helloworld.c
C:\Users\vineelko\AppData\Local\Te
Browse...
C:\Users\vineelko\Desktop\my git\N
Browse...
Ln 9, Col 1 INS

```
#include<stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}
```

```
#include<stdio.h>
//Commented add
int main()
{
    printf("Hello World!\n");
    printf("Hello World!\n");
    return 0;
}
```

Undo changes in Git

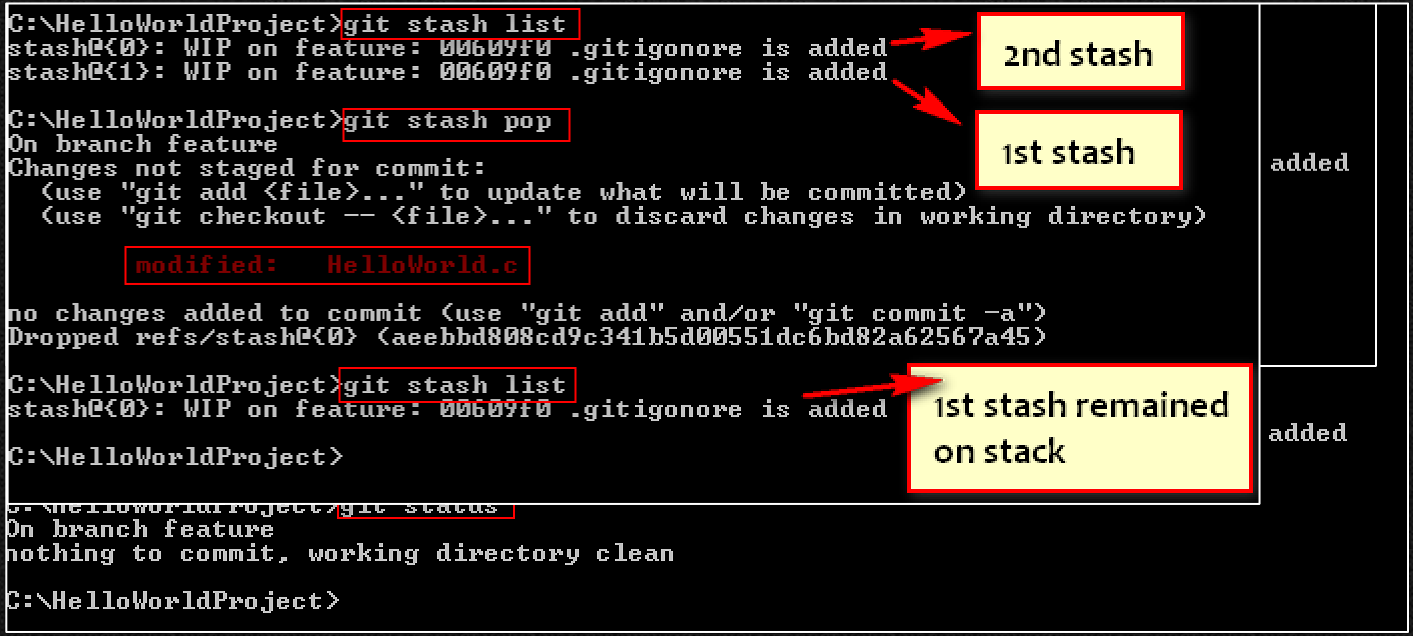




```
C:\HelloWorldProject>git log
commit cb3b5d2[redacted]
Author: Vineel [redacted]>
Date:   Tue Jul [redacted]

    Added comments to the helloworld!

commit e00b227c5ce76721190f983f34958e3a1aab283b
Author: Vineel [redacted]>
Date:   Tue Jul 14 20:08:39 2015 +0530

    First HelloWorld Program

C:\HelloWorldProject>git reset HEAD~1 --hard
HEAD is now at e00b227 First HelloWorld Program

C:\HelloWorldProject>git log
commit e00b227c5ce76721190f983f34958e3a1aab283b
Author: Vineel [redacted]>
Date:   Tue Jul [redacted]

    First HelloWorld Program

C:\HelloWorldProject>
```

Stashing your changes

- *git stash* command is used to temporarily store your modification on a stack
- *git stash list* will show all the stashed changed
- *git stash apply* will just apply the top of the stack stash
- *git stash pop* will pop previously saved modifications from stack
- *git stash drop* will drop the topmost stash from the stack



```
C:\HelloWorldProject>git stash list
stash@{0}: WIP on feature: 00609f0 .gitignore is added
stash@{1}: WIP on feature: 00609f0 .gitignore is added

C:\HelloWorldProject>git stash pop
On branch feature
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   HelloWorld.c

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (aeebbd808cd9c341b5d00551dc6bd82a62567a45)

C:\HelloWorldProject>git stash list
stash@{0}: WIP on feature: 00609f0 .gitignore is added

C:\HelloWorldProject>
```

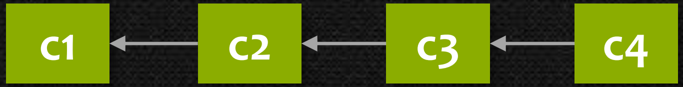
```
added
added
On branch feature
nothing to commit, working directory clean

C:\HelloWorldProject>
```

Demo

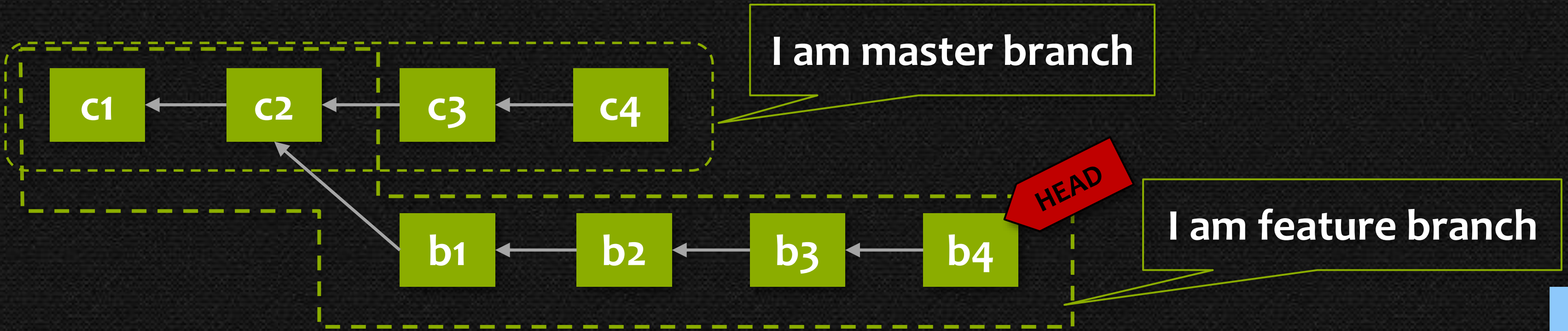
- status
- add
- commit
- config
- log
- diff/difftool
- reset
- stash

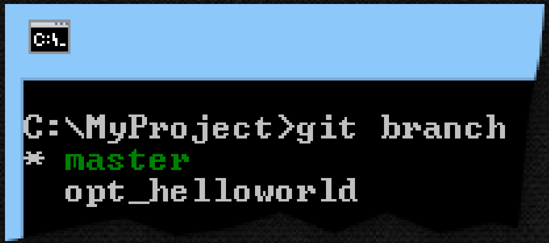
What are branches and why should I care?

- Branch is just a sequence of commits with a parent child relationship
- The default branch is always referred as *master*



- Branching helps in working with multiple features independently
- At any given point in time, There can be only one *active* branch in a repository





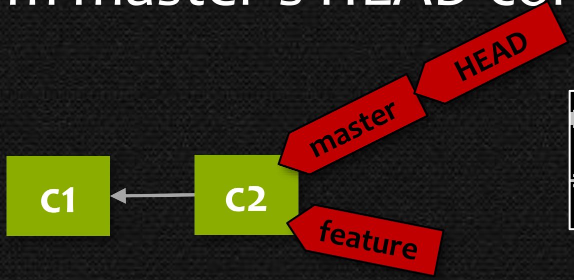
```
C:\MyProject>git branch
* master
  opt_helloworld
```

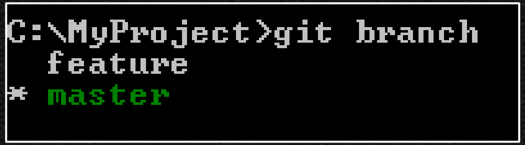
- The content of the file and folder structure of the repo is determined by the commits on current *active* branch
- *git branch* will show *all branches and highlights the current active branch

* = local ☺

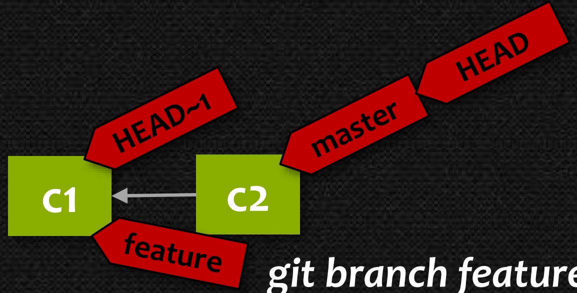
Branching

- *git branch feature master* will create a new branch named '*feature*' from master's HEAD commit



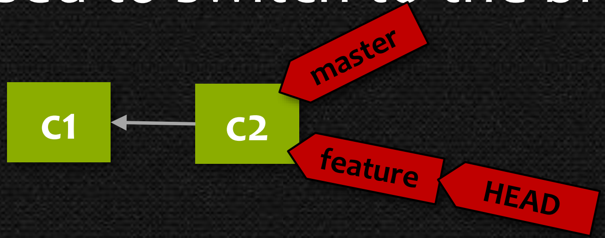


```
C:\MyProject>git branch
  feature
* master
```

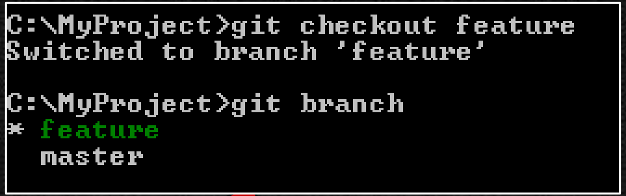


git branch feature HEAD~1

- *git checkout feature* is used to switch to the branch named '*feature*'

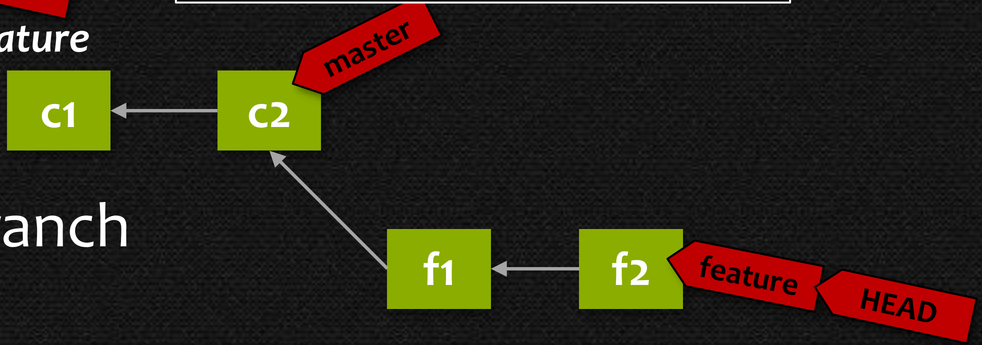


HEAD after checkout of feature



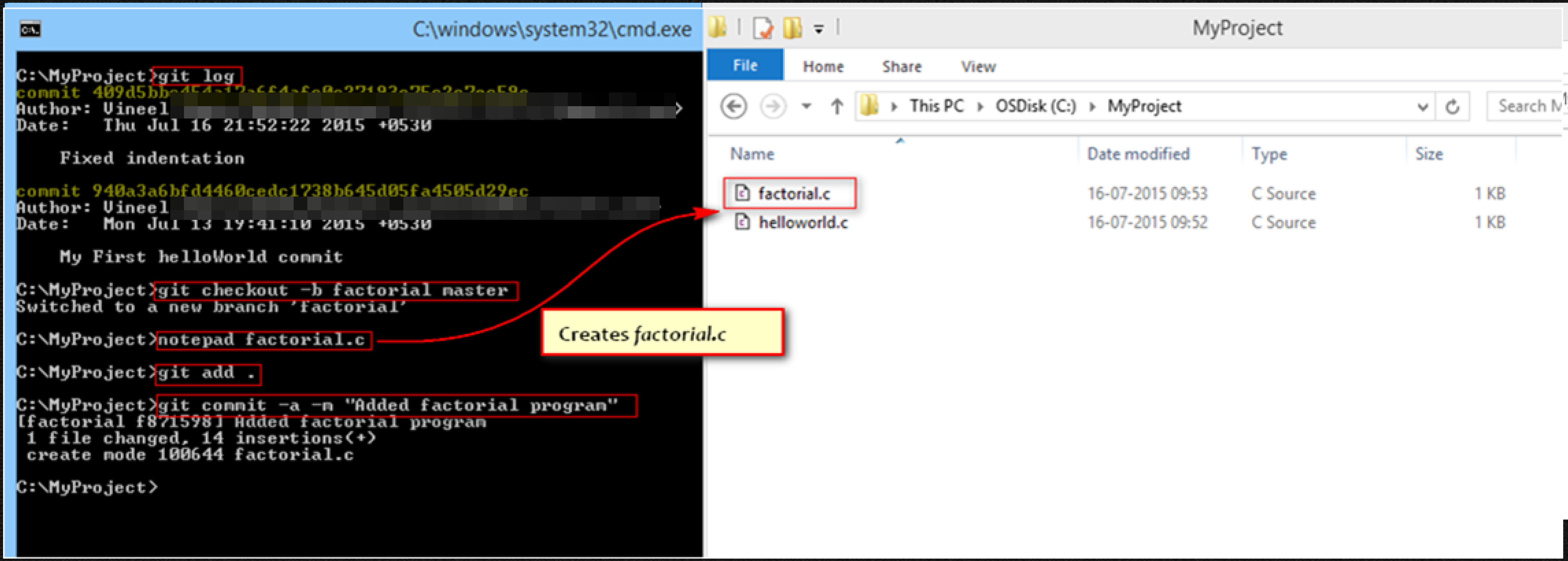
```
C:\MyProject>git checkout feature
Switched to branch 'feature'

C:\MyProject>git branch
* feature
  master
```

- With each commit on the feature branch,
  The HEAD moves forward on the feature branch



git checkout –b feature master = git branch feature master + git checkout feature

Gist of Git branching


C:\windows\system32\cmd.exe
MyProject
File
Home
Share
View
This PC › OSDisk (C:) › MyProject
Search M

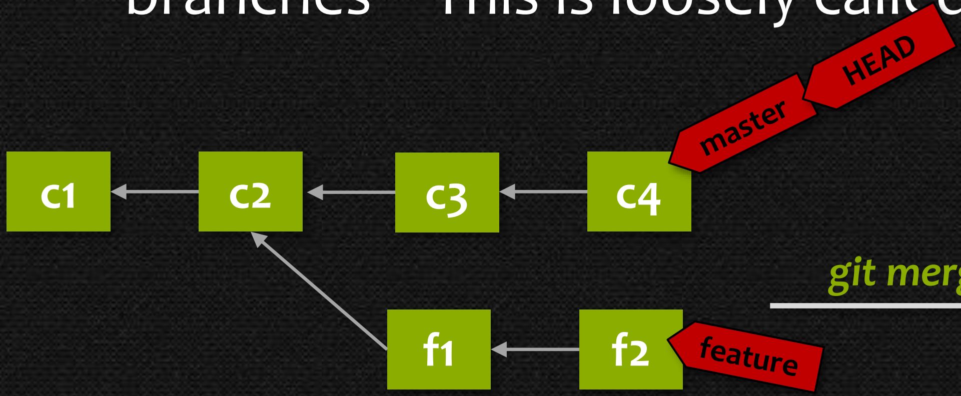
```
C:\MyProject>git log
commit 409d5bb[redacted]
Author: Vineel [redacted]
Date:   Thu Jul 16 21:52:22 2015 +0530

    Fixed indentation

commit 940a3a6bfd4460cedc1738b645d05fa4505d29ec
Author: Vineel [redacted]
Date:   Mon Jul 13 19:41:10 2015 +0530

    My First helloWorld commit

C:\MyProject>git checkout -b factorial master
Switched to a new branch 'factorial'

C:\MyProject>notepad factorial.c

C:\MyProject>git add .

C:\MyProject>git commit -a -m "Added factorial program"
[factorial f871598] Added factorial program
 1 file changed, 14 insertions(+)
 create mode 100644 factorial.c

C:\MyProject>
```

Creates *factorial.c*

Name	Date modified	Type	Size
factorial.c	16-07-2015 09:53	C Source	1 KB
helloworld.c	16-07-2015 09:52	C Source	1 KB

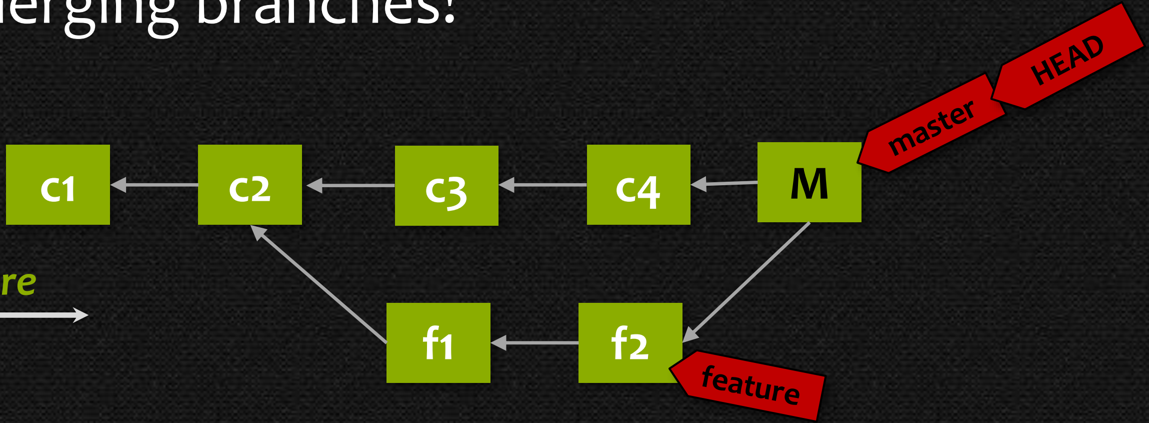
Merging

- *git merge* is used to create a merge commit between two or more branches – This is loosely called as merging branches!

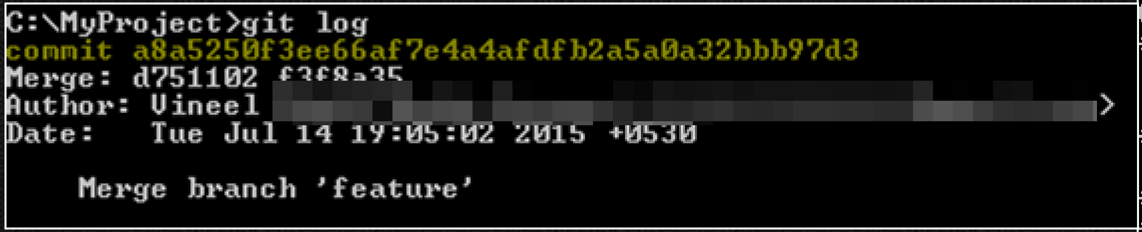


HEAD on master branch after checkout of master

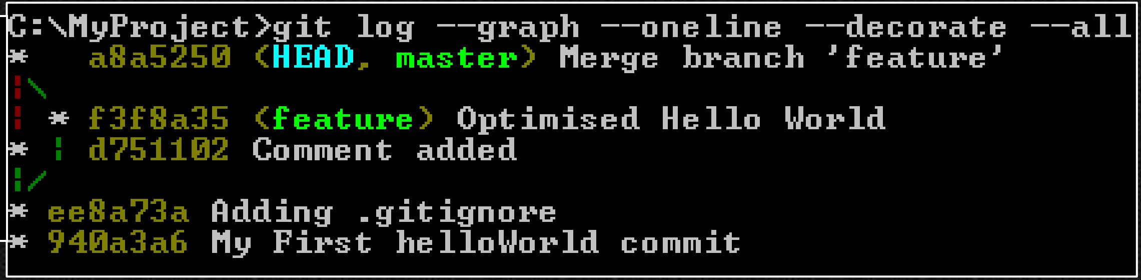
git merge feature



Merge commit created on master after git merge command



```
C:\MyProject>git log
commit a8a5250f3ee66af7e4a4afdfb2a5a0a32bbb97d3
Merge: d751102 f3f8a35
Author: Vineel [redacted]>
Date:   Tue Jul 14 19:05:02 2015 +0530

    Merge branch 'feature'
```



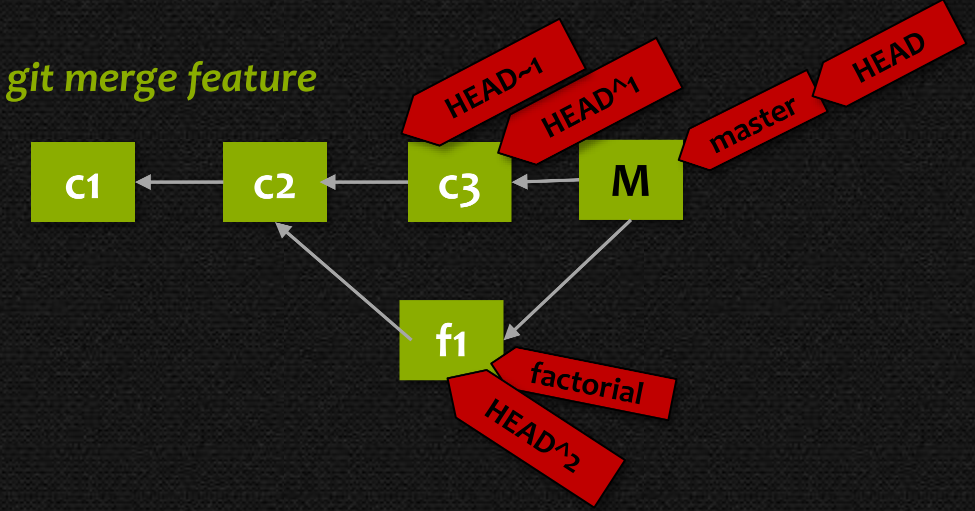
```
C:\MyProject>git log --graph --oneline --decorate --all
*   a8a5250 (HEAD, master) Merge branch 'feature'
|\
| * f3f8a35 (feature) Optimised Hello World
* | d751102 Comment added
|/
* ee8a73a Adding .gitignore
* 940a3a6 My First helloWorld commit
```

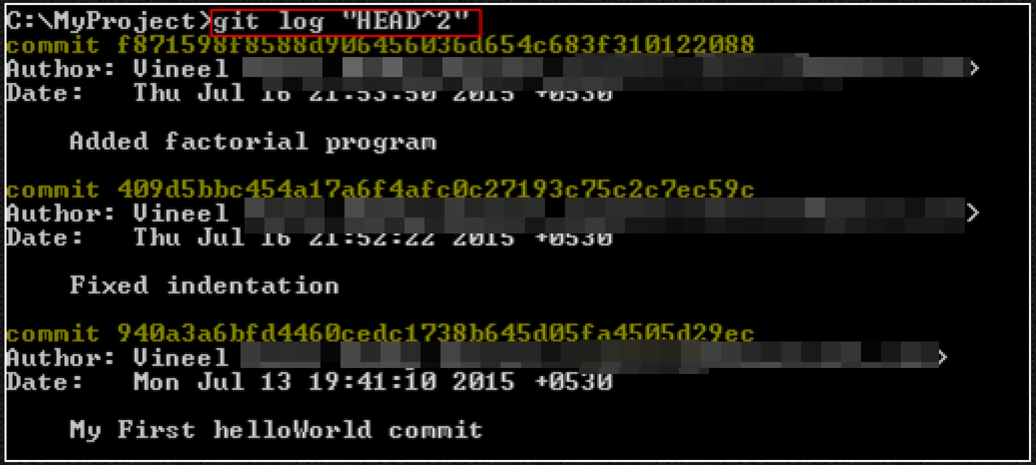
- In the above workflow the important point to note is, merge commit M is created on master branch and not on feature branch

Git log and Git HEAD revisited

- Of all commits, Merge commit **M** is little special, it has multiple parents

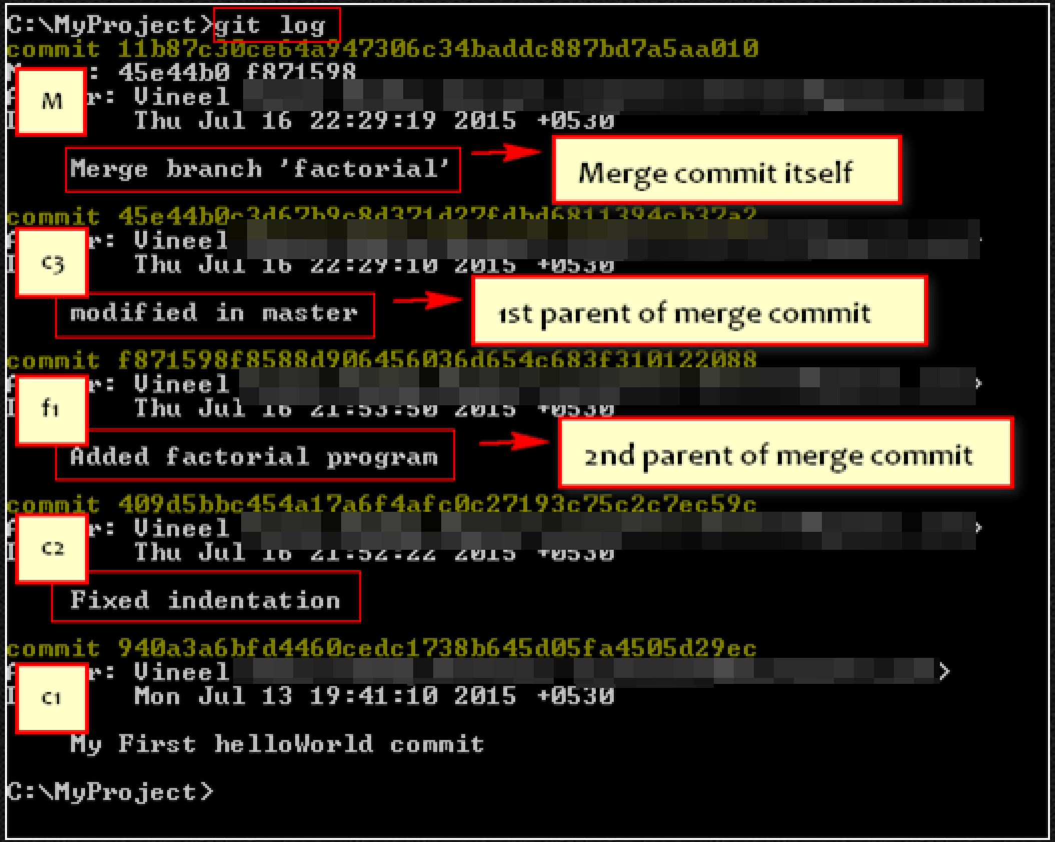
git merge feature





```
C:\MyProject>git log "HEAD^2"
commit f871598f8588d906456036d654c683f310122088
Author: Vineel [redacted]
Date:   Thu Jul 16 21:53:50 2015 +0530

    Added factorial program

commit 409d5bbc454a17a6f4afc0c27193c75c2c7ec59c
Author: Vineel [redacted]
Date:   Thu Jul 16 21:52:22 2015 +0530

    Fixed indentation

commit 940a3a6bfd4460cedc1738b645d05fa4505d29ec
Author: Vineel [redacted]
Date:   Mon Jul 13 19:41:10 2015 +0530

    My First helloWorld commit
```

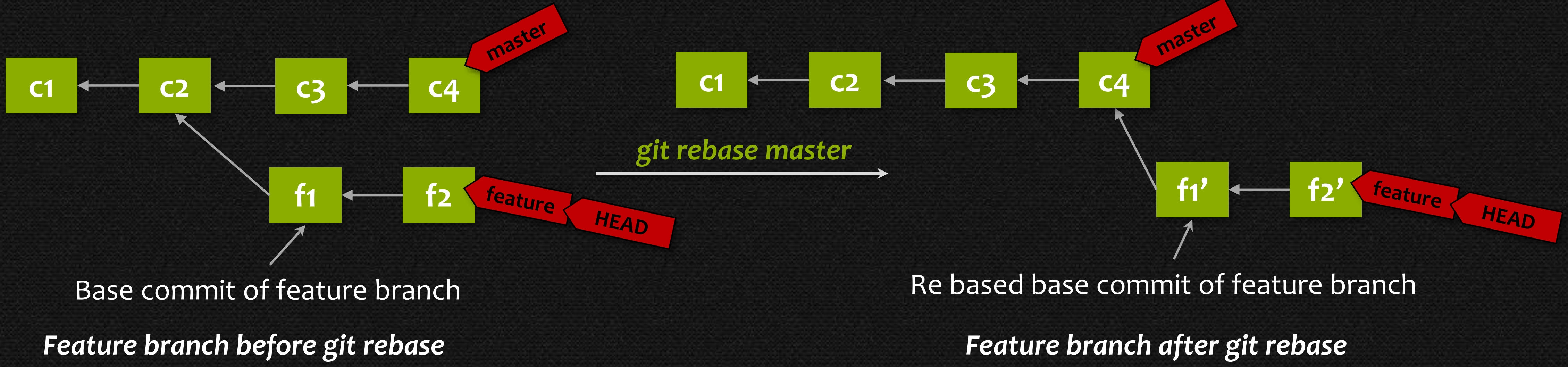
git log [merge commit]



```
C:\MyProject>git log
commit 11b87c30ce64a947306c34baddc887bd7a5aa010
M    : 45e44b0 f871598
f    r: Vineel [redacted]
I        Thu Jul 16 22:29:19 2015 +0530

    Merge branch 'factorial'

commit 45e44b0c3d67b9c8d371d27fdbd6811394cb37a2
f    r: Vineel [redacted]
I        Thu Jul 16 22:29:10 2015 +0530

    modified in master

commit f871598f8588d906456036d654c683f310122088
f    r: Vineel [redacted]
I        Thu Jul 16 21:53:50 2015 +0530

    Added factorial program

commit 409d5bbc454a17a6f4afc0c27193c75c2c7ec59c
f    r: Vineel [redacted]
I        Thu Jul 16 21:52:22 2015 +0530

    Fixed indentation

commit 940a3a6bfd4460cedc1738b645d05fa4505d29ec
f    r: Vineel [redacted]
I        Mon Jul 13 19:41:10 2015 +0530

    My First helloWorld commit

C:\MyProject>
```

Rebasing

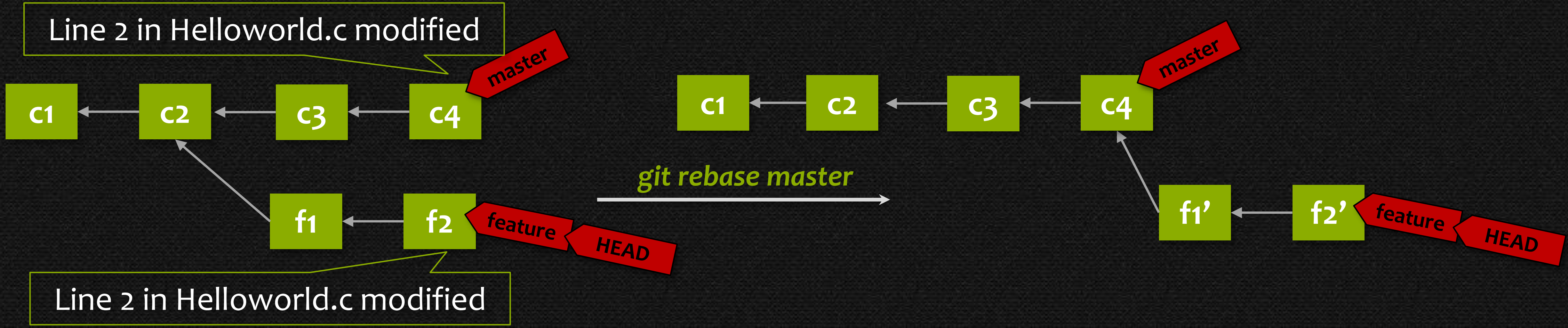
- *git rebase* realigns the base commit of the current branch with other branch

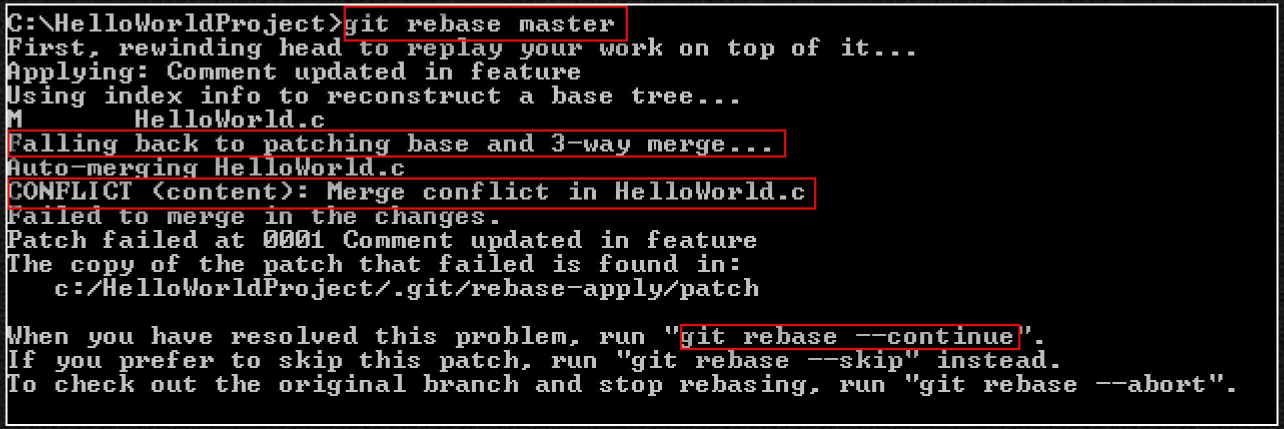

c1
c2
c3
c4
master
f1
f2
feature
HEAD
Base commit of feature branch
Feature branch before git rebase
git rebase master
c1
c2
c3
c4
master
f1'
f2'
feature
HEAD
Re based base commit of feature branch
Feature branch after git rebase

f1	Contains changes made before rebase
f1'	May not contain the same changes as f1 because of merge conflicts

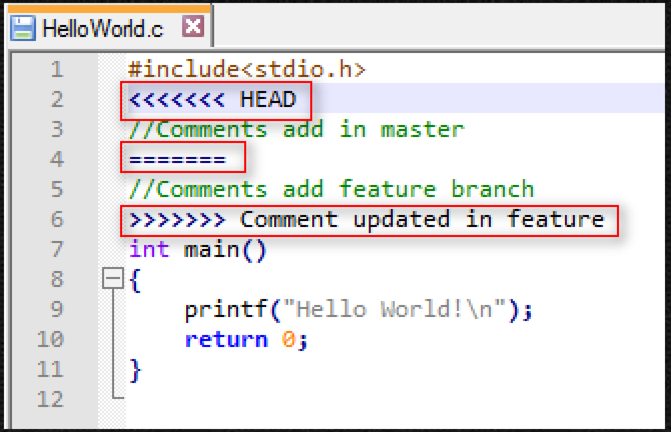
Resolving conflicts manually in Git

- *git merge* and *git rebase* can sometime lead to merge conflicts





```
C:\HelloWorldProject>git rebase master
First, rewinding head to replay your work on top of it...
Applying: Comment updated in feature
Using index info to reconstruct a base tree...
M       HelloWorld.c
Falling back to patching base and 3-way merge...
Auto-merging HelloWorld.c
CONFLICT (content): Merge conflict in HelloWorld.c
Failed to merge in the changes.
Patch failed at 0001 Comment updated in feature
The copy of the patch that failed is found in:
   c:/HelloWorldProject/.git/rebase-apply/patch

When you have resolved this problem, run "git rebase --continue".
If you prefer to skip this patch, run "git rebase --skip" instead.
To check out the original branch and stop rebasing, run "git rebase --abort".
```



HelloWorld.c

```
#include<stdio.h>
<<<<<<< HEAD
//Comments add in master
=======
//Comments add feature branch
>>>>>>> Comment updated in feature
int main()
{
    printf("Hello World!\n");
    return 0;
}
```

Demo

- branch
- merge
- rebase
- conflict

Remote

- Even though git did not have the concept of a central server to control it, it does have the concept of *local* and *remote*

- A remote repository is just any other repository that is not your current working repository

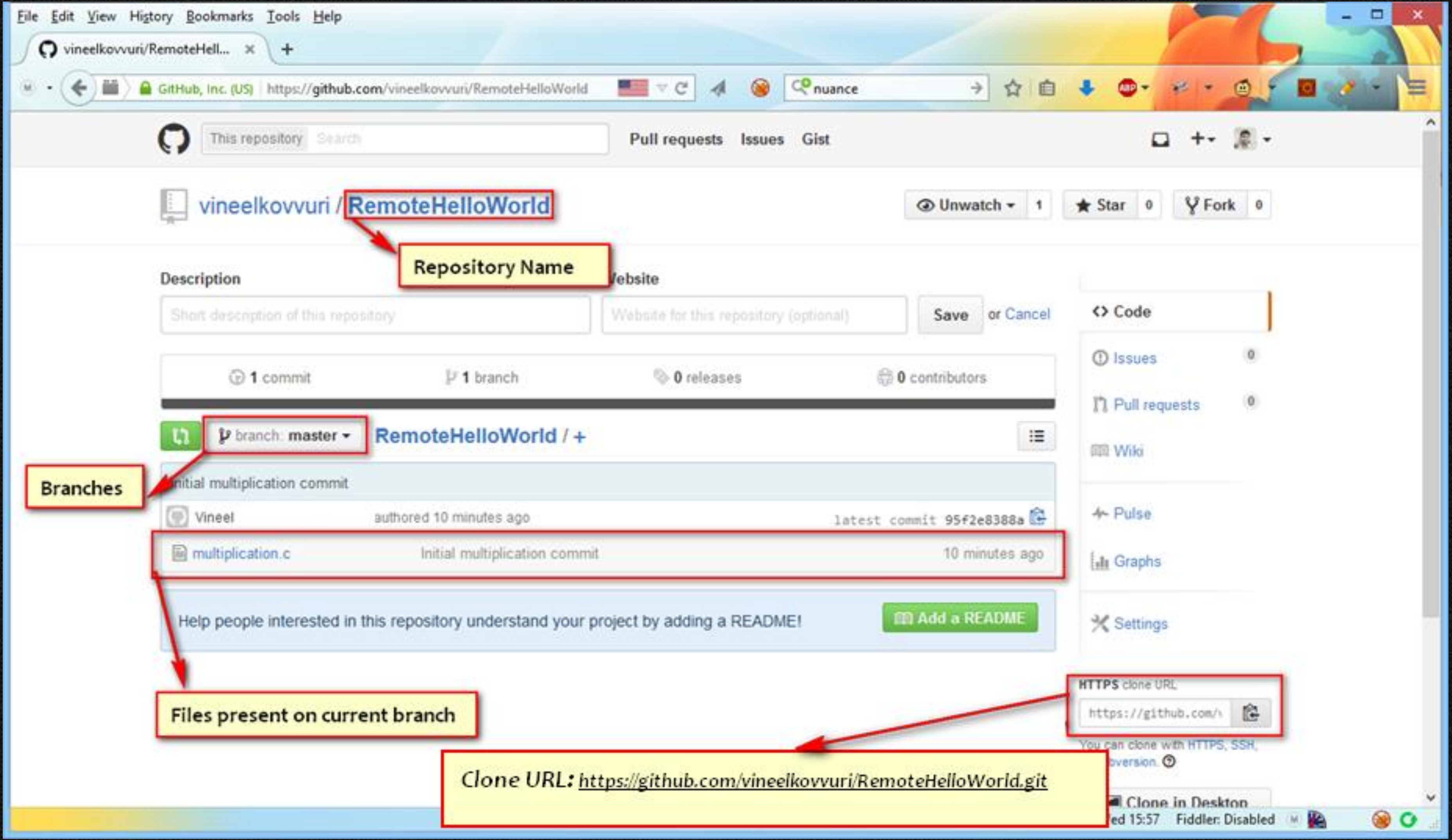
remote do not necessarily mean some server or cloud repository

- Remote can be another git repository present in your local hard drive!

- In git, repositories talk to each other by *pushing* and *pulling* branches from each other

Remote Repository ≡ Local Repository

- https://github.com/vineelkovvuri/RemoteHelloWorld



vineelkovvuri / RemoteHelloWorld

Repository Name

Description

Website

1 commit | 1 branch | 0 releases | 0 contributors

branch: master

RemoteHelloWorld / +

Branches

Initial multiplication commit

Vineel authored 10 minutes ago latest commit 95f2e8388a

multiplication.c Initial multiplication commit 10 minutes ago

Help people interested in this repository understand your project by adding a README!

Files present on current branch

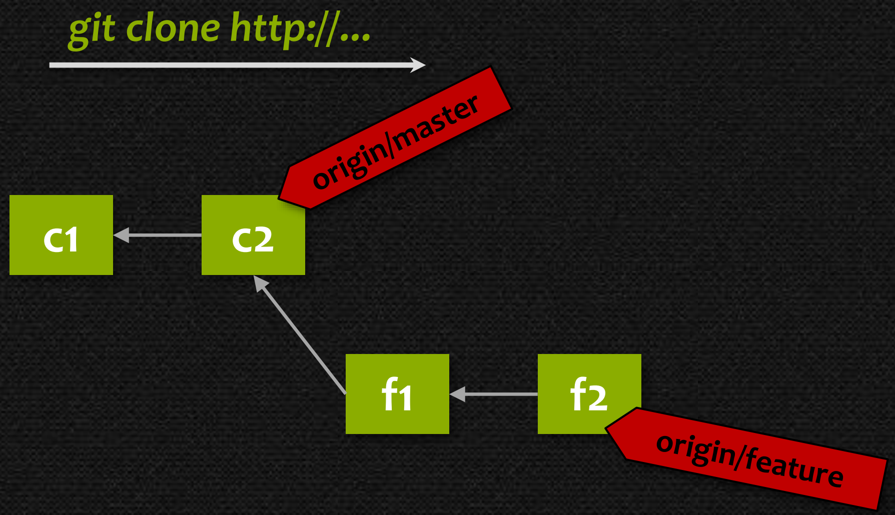
HTTPS clone URL

Clone URL: https://github.com/vineelkovvuri/RemoteHelloWorld.git

File Edit View History Bookmarks Tools Help
vineelkovvuri/RemoteHell...
GitHub, Inc. (US) https://github.com/vineelkovvuri/RemoteHelloWorld
nuance
This repository Search
Pull requests Issues Gist
Unwatch 1 Star 0 Fork 0
Save or Cancel
Code
Issues 0
Pull requests 0
Wiki
Pulse
Graphs
Settings
Add a README
Clone in Desktop
Fiddler: Disabled

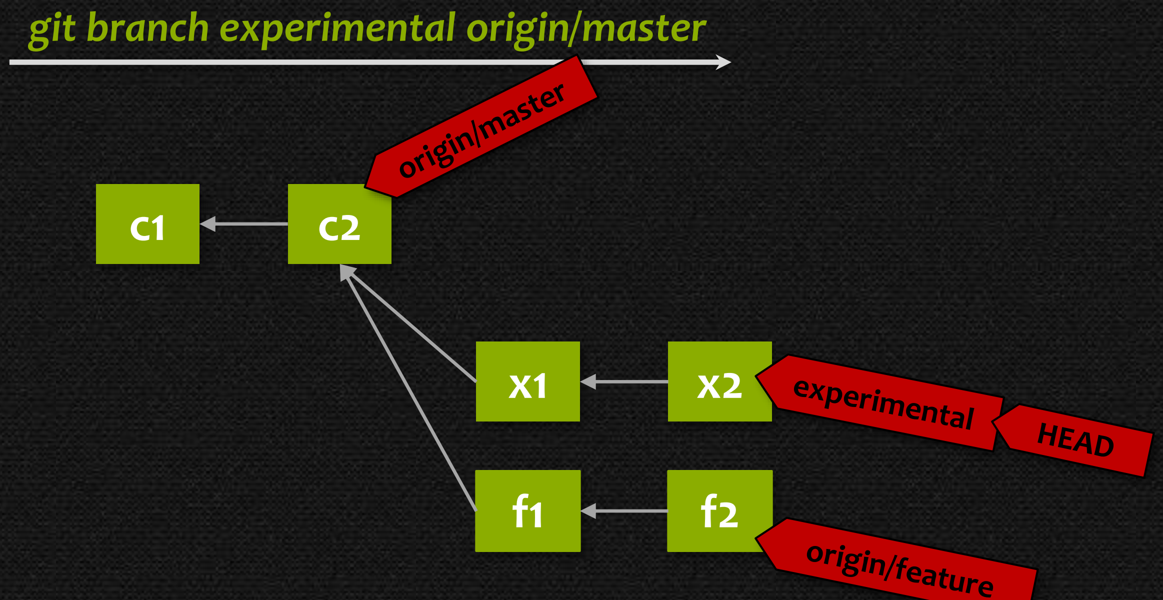
Cloning a remote repository

- *git clone* is used to create a new copy of remote repository in local machine
- Git clone completely copies all the branches from the remote repository
- By default git clone bookmarks the URL of the remote repo as *origin*



All branches from remote repository are Cloned in to local repo after a git clone

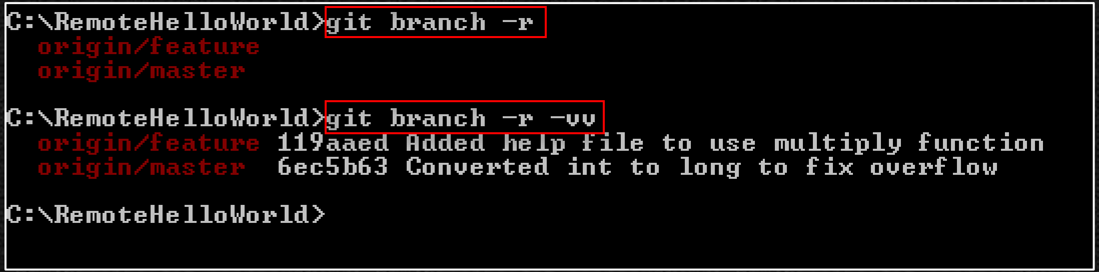
Create local branch with remote branch reference



New experimental branch created from origin/master

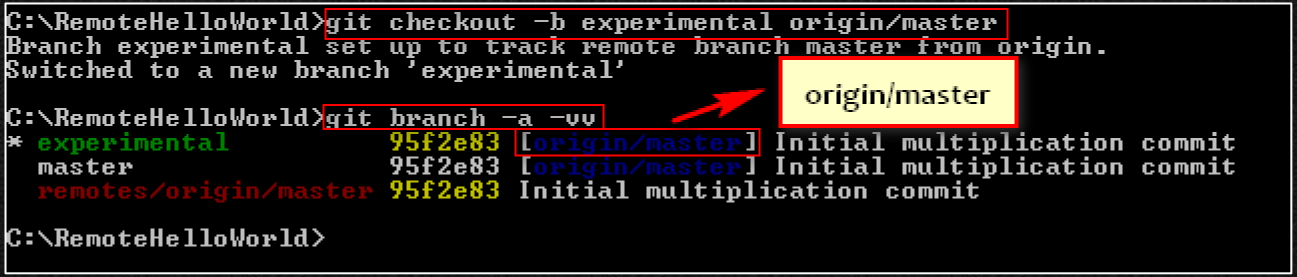
Listing local and remote branches

- *git branch –r* can be used to list only remote branches



```
C:\RemoteHelloWorld>git branch -r
  origin/feature
  origin/master

C:\RemoteHelloWorld>git branch -r -vv
  origin/feature 119aaed Added help file to use multiply function
  origin/master  6ec5b63 Converted int to long to fix overflow

C:\RemoteHelloWorld>
```

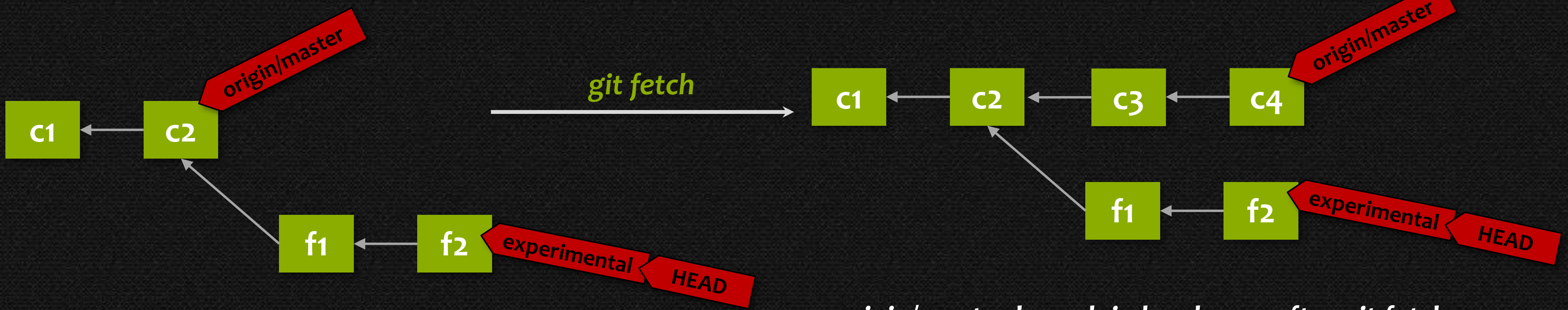
- *git branch –a –vv* list all(-a) branches(both local and remote) with tracking information(-vv)



```
C:\RemoteHelloWorld>git checkout -b experimental origin/master
Branch experimental set up to track remote branch master from origin.
Switched to a new branch 'experimental'

C:\RemoteHelloWorld>git branch -a -vv
* experimental          95f2e83 [origin/master] Initial multiplication commit
  master                95f2e83 [origin/master] Initial multiplication commit
  remotes/origin/master 95f2e83 Initial multiplication commit

C:\RemoteHelloWorld>
```

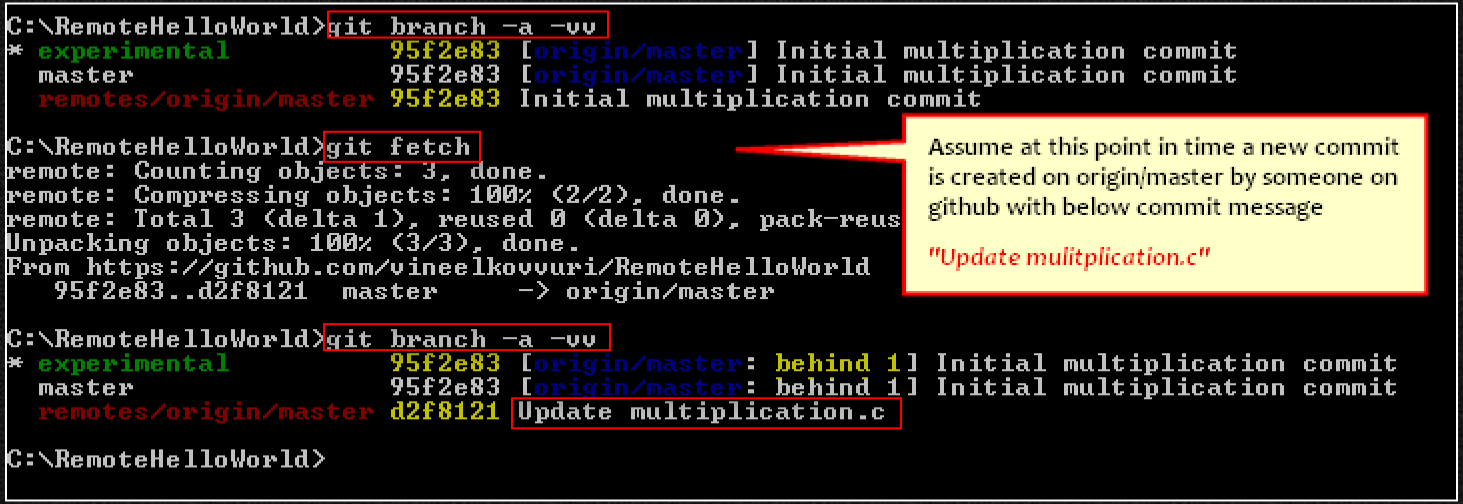
Fetching

- *git fetch* gets all the remote objects(commits/branches)
- It will not update any local branches



origin/master branch in local repo before git fetch

origin/master branch in local repo after git fetch



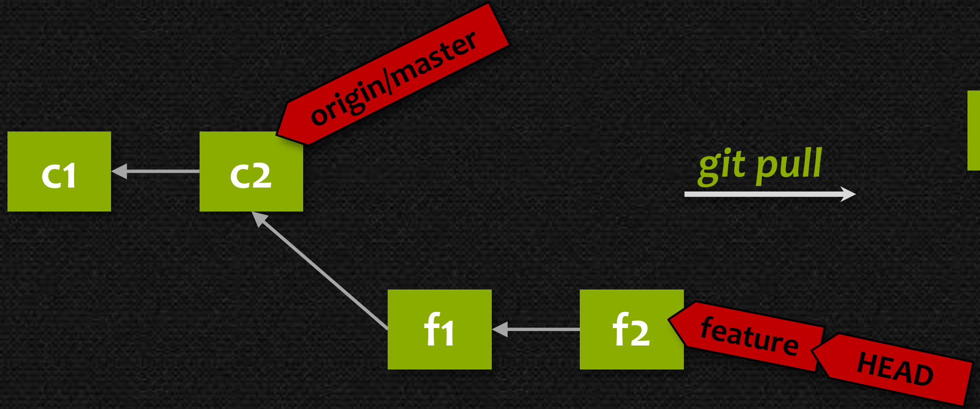
```
C:\RemoteHelloWorld>git branch -a -vv
* experimental                95f2e83 [origin/master] Initial multiplication commit
  master                      95f2e83 [origin/master] Initial multiplication commit
  remotes/origin/master       95f2e83 Initial multiplication commit

C:\RemoteHelloWorld>git fetch
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reus
Unpacking objects: 100% (3/3), done.
From https://github.com/vineelkovvuri/RemoteHelloWorld
   95f2e83..d2f8121  master     -> origin/master

C:\RemoteHelloWorld>git branch -a -vv
* experimental                95f2e83 [origin/master: behind 1] Initial multiplication commit
  master                      95f2e83 [origin/master: behind 1] Initial multiplication commit
  remotes/origin/master       d2f8121 Update multiplication.c

C:\RemoteHelloWorld>
```

Assume at this point in time a new commit is created on origin/master by someone on github with below commit message

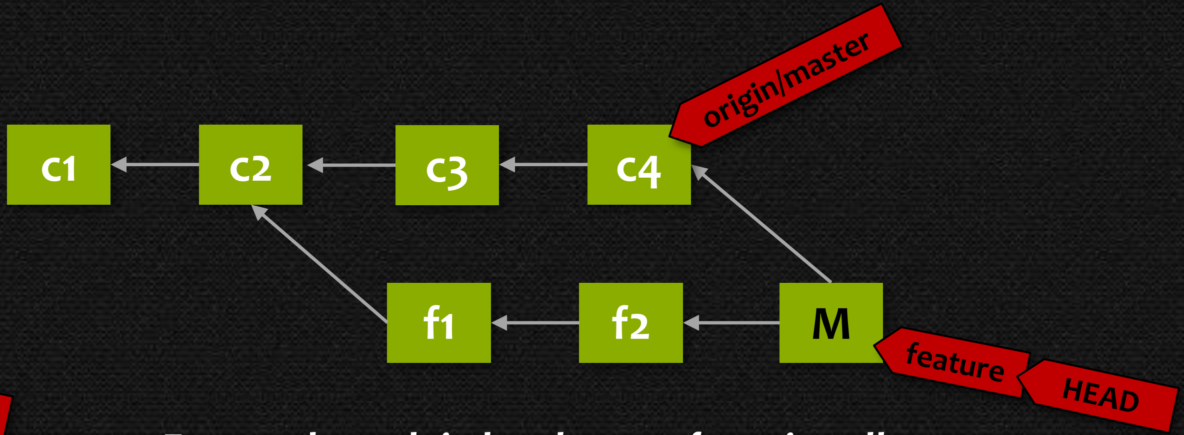
"Update mulitplication.c"

How to pull from remote repository?

- *git pull* will do exactly what git fetch does and creates an additional merge commit with remote branch on to the current branch



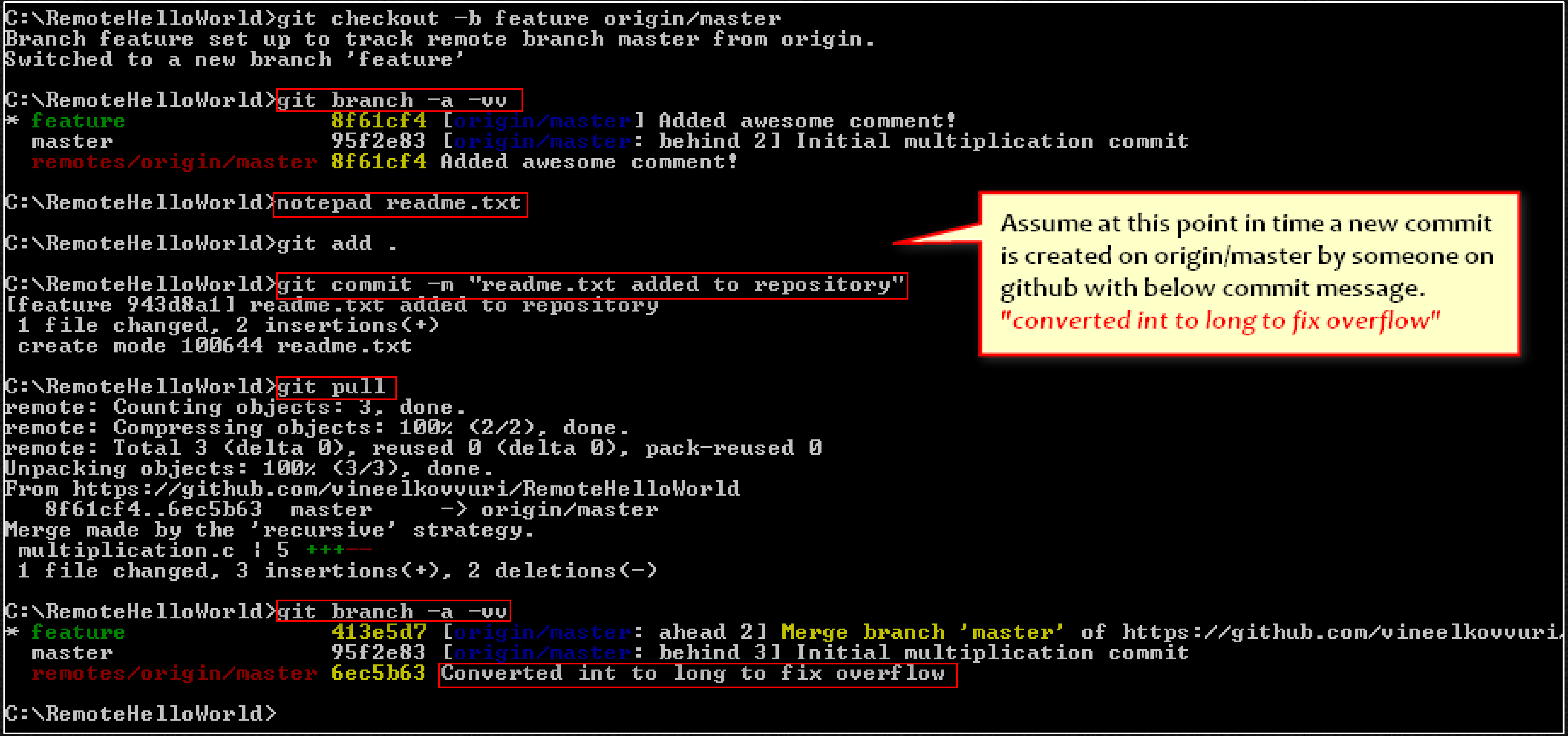
Feature branch in local repo before git pull



Feature branch in local repo after git pull

***git pull = git fetch + git merge**(on current branch)*

How to pull from remote repository?

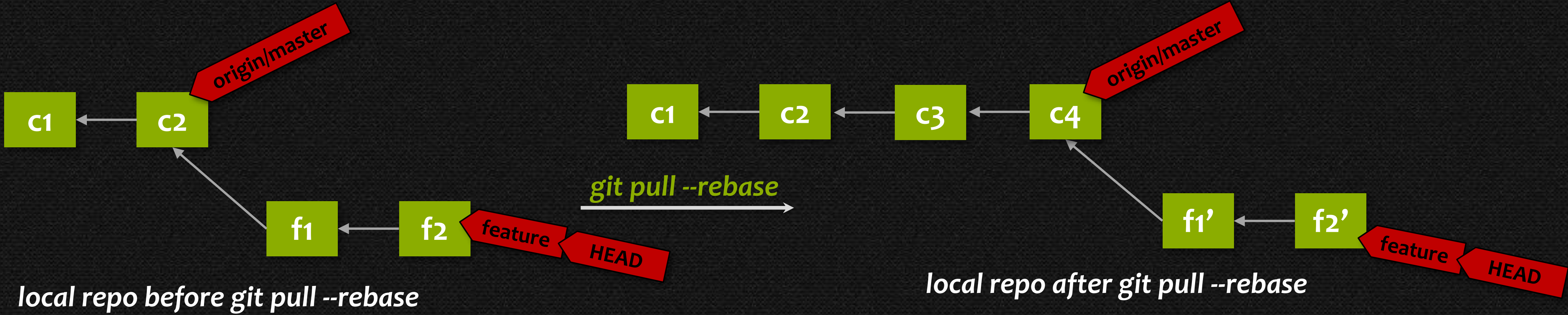


```
C:\RemoteHelloWorld>git checkout -b feature origin/master
Branch feature set up to track remote branch master from origin.
Switched to a new branch 'feature'

C:\RemoteHelloWorld>git branch -a -vv
* feature                 8f61cf4 [origin/master] Added awesome comment!
  master                  95f2e83 [origin/master: behind 2] Initial multiplication commit
  remotes/origin/master 8f61cf4 Added awesome comment!

C:\RemoteHelloWorld>notepad readme.txt

C:\RemoteHelloWorld>git add .

C:\RemoteHelloWorld>git commit -m "readme.txt added to repository"
[feature 943d8a1] readme.txt added to repository
 1 file changed, 2 insertions(+)
 create mode 100644 readme.txt

C:\RemoteHelloWorld>git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/vineelkovvuri/RemoteHelloWorld
   8f61cf4..6ec5b63  master     -> origin/master
Merge made by the 'recursive' strategy.
 multiplication.c | 5 +++--
 1 file changed, 3 insertions(+), 2 deletions(-)

C:\RemoteHelloWorld>git branch -a -vv
* feature                 413e5d7 [origin/master: ahead 2] Merge branch 'master' of https://github.com/vineelkovvuri
  master                  95f2e83 [origin/master: behind 3] Initial multiplication commit
  remotes/origin/master 6ec5b63 Converted int to long to fix overflow

C:\RemoteHelloWorld>
```

Assume at this point in time a new commit is created on origin/master by someone on github with below commit message.
"converted int to long to fix overflow"

git pull --rebase

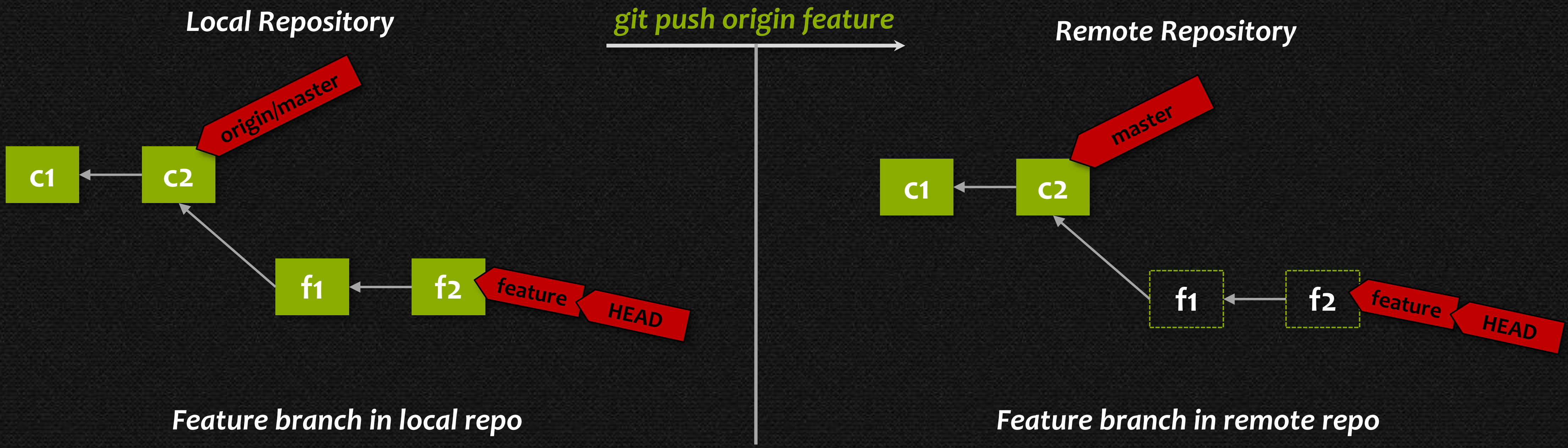
- *git pull ––rebase* donot recreate the merge commit, instead, after the fetch it rebases the current branch with the origin/master



git pull ––rebase = git fetch + git rebase(current branch)

How to push to remote repository?

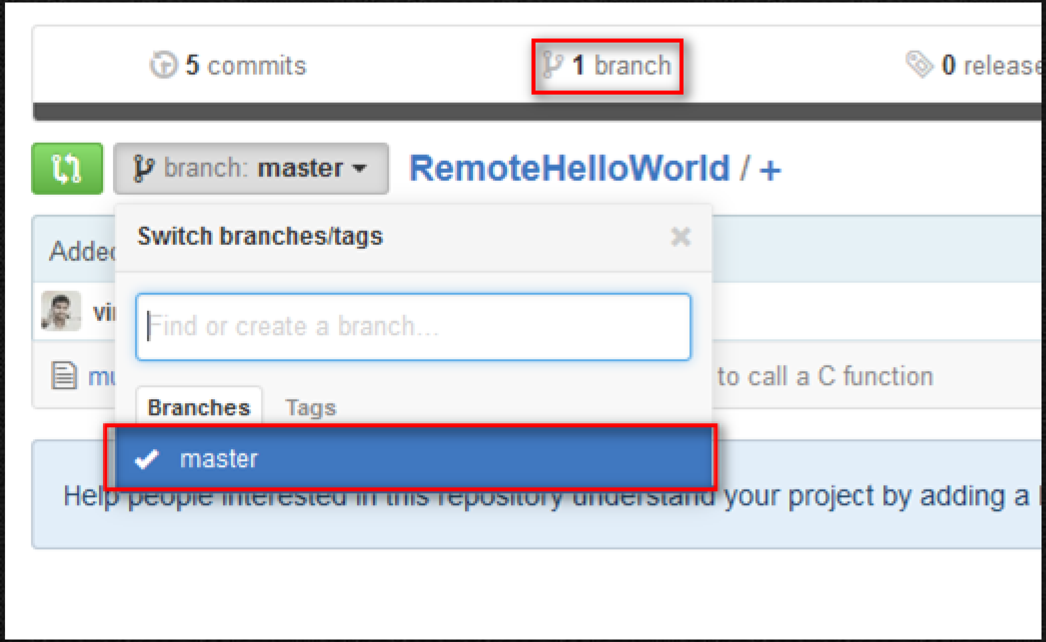
- *git push origin <branchname>*
  - Push the commits from the local branch with name <branchname> to remote branch with the *same name*
    - If remote do not have a branch with same name git tries to create it and then push the commits
- *git push origin HEAD:<RemoteBranch>*



How to push to remote repository?

git push origin feature

Remote Repository branches before push



5 commits 1 branch 0 release

branch: master RemoteHelloWorld / +

Switch branches/tags

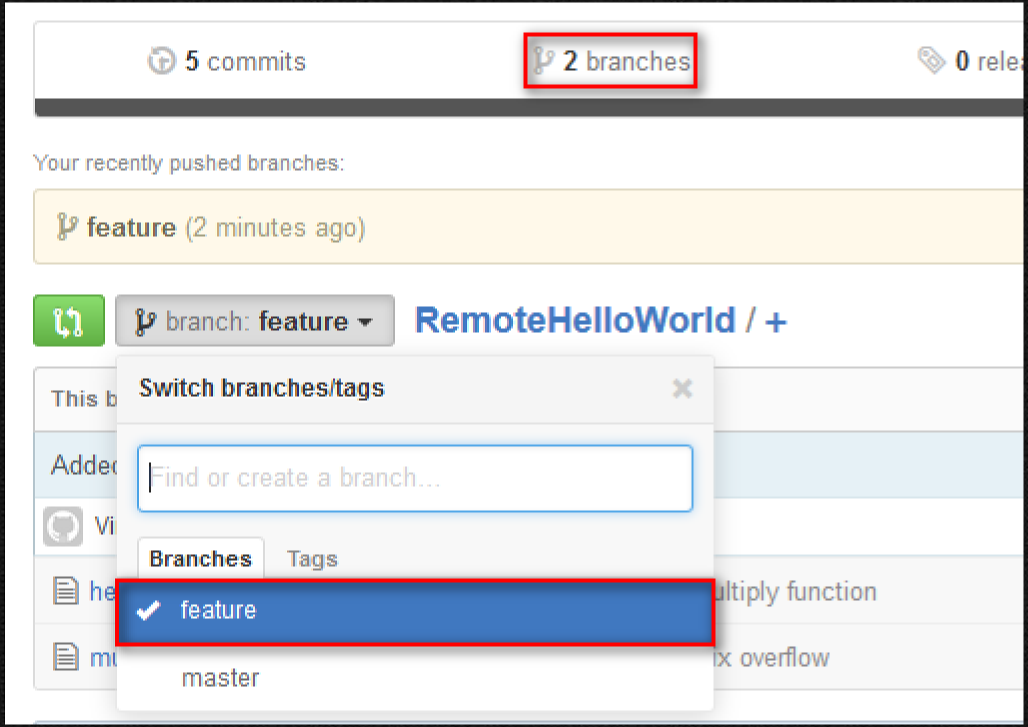
Find or create a branch…

Branches Tags

✓ master

Help people interested in this repository understand your project by adding a

Remote Repository branches after push



5 commits 2 branches 0 rele

Your recently pushed branches:

feature (2 minutes ago)

branch: feature RemoteHelloWorld / +

Switch branches/tags

Find or create a branch…

Branches Tags

✓ feature

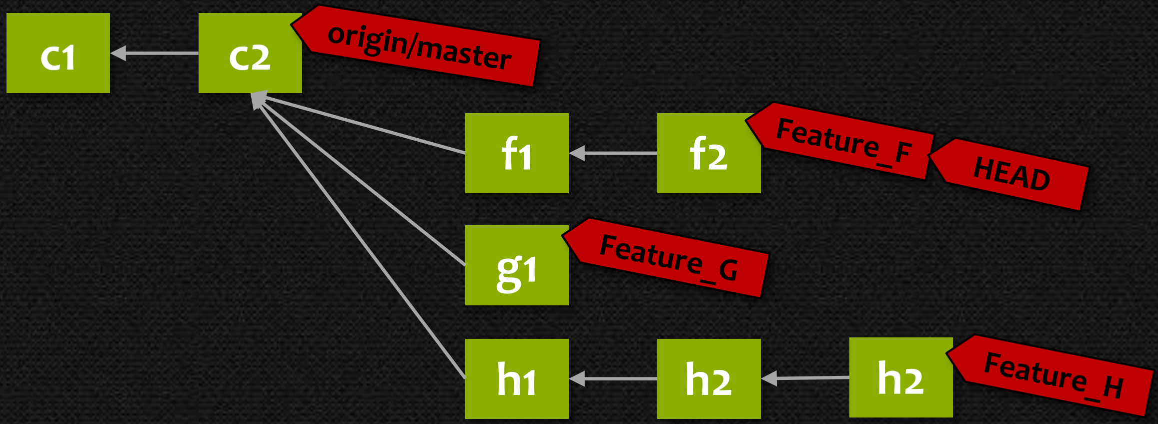
master

Demo

- clone
- fetch
- pull
- pull –rebase
- push

Simple Daily Workflow

- Step 1: *git branch feature <remotename/remotebranch>* create feature branch
- Step 2: *git checkout feature* checkout feature branch
- Step 3: make changes in feature branch and create commits
- Step 4: *git pull ––rebase* will make sure your feature branch is up to date with remotebranch
- Step 5: *git push origin HEAD:<remotebranch>* will push all your changes on the current local branch to the specified remotebranch
- Step 6: for a new feature repeat Step 1 ☺



Git Tips & Tricks

- git add -i
- git commit –-amend
- git log –p
- git log –-name-only
- git diff –-cached
- git difftool –-dir-diff
- git branch –D
- git format-patch -<n>
- git am <patch>
- git revert
- git show stash@{0}
- git rebase –i
- git grep
- git blame
- git bisect
- git help

What is repo tool?

- Git only handles one project it do not have the concept of multiple git projects or the concept of sub git projects
- This becomes mandatory if we are working on large scale projects with multiple sub projects
- To address this issue, Google has created a python wrapper script called *repo* for managing Android source code
- https://source.android.com/source/using-repo.html

Git Advice

- In many ways the learning curve for Git is comparable to Vi editor
- Learning Git with *hash/tree/blob* objects is like learning vi editor with vimscript! So never start there!
- Start with basic *add, commit, log, reset, stash* commands that do your job
  - Its like starting vi editor with *I, esc, :wq* keystrokes
- Day by day you will start to build your muscle memory with more git commands and workflows
- Try not to use GUI tools. They will hide some important useful details

References

https://git-scm.com/book/en/v2
http://gitref.org/
https://www.kernel.org/pub/software/scm/git/docs/

```
git help <command>
```

SD To Git Cheat Sheet

Thank You