Microsoft

WorkshopPLUS: Essentials – Git

Branching and Merging

Microsoft Services

Module Overview

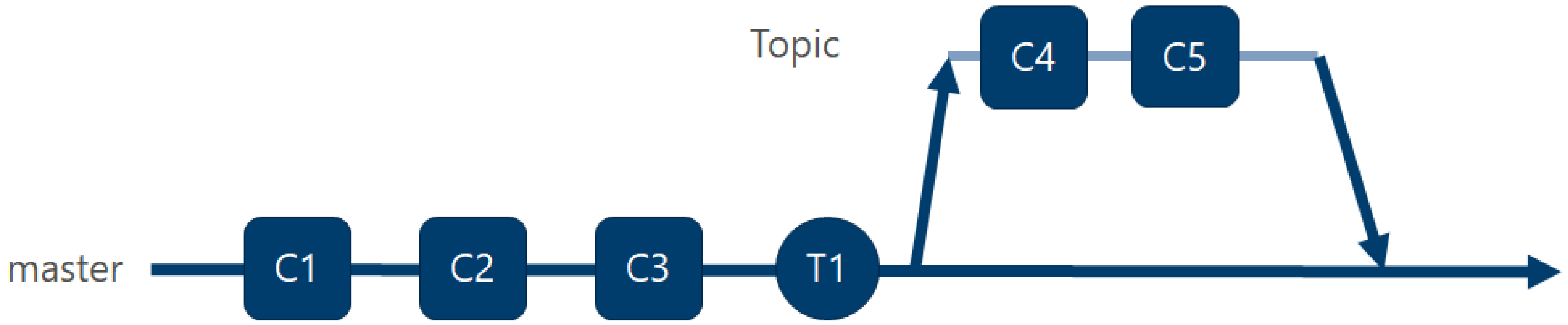
- Branching in Git
- Create a Branch
- Merge a Branch
- Push a Branch
- Resolve Content Conflicts
- Commit the Merge

Branching in Git

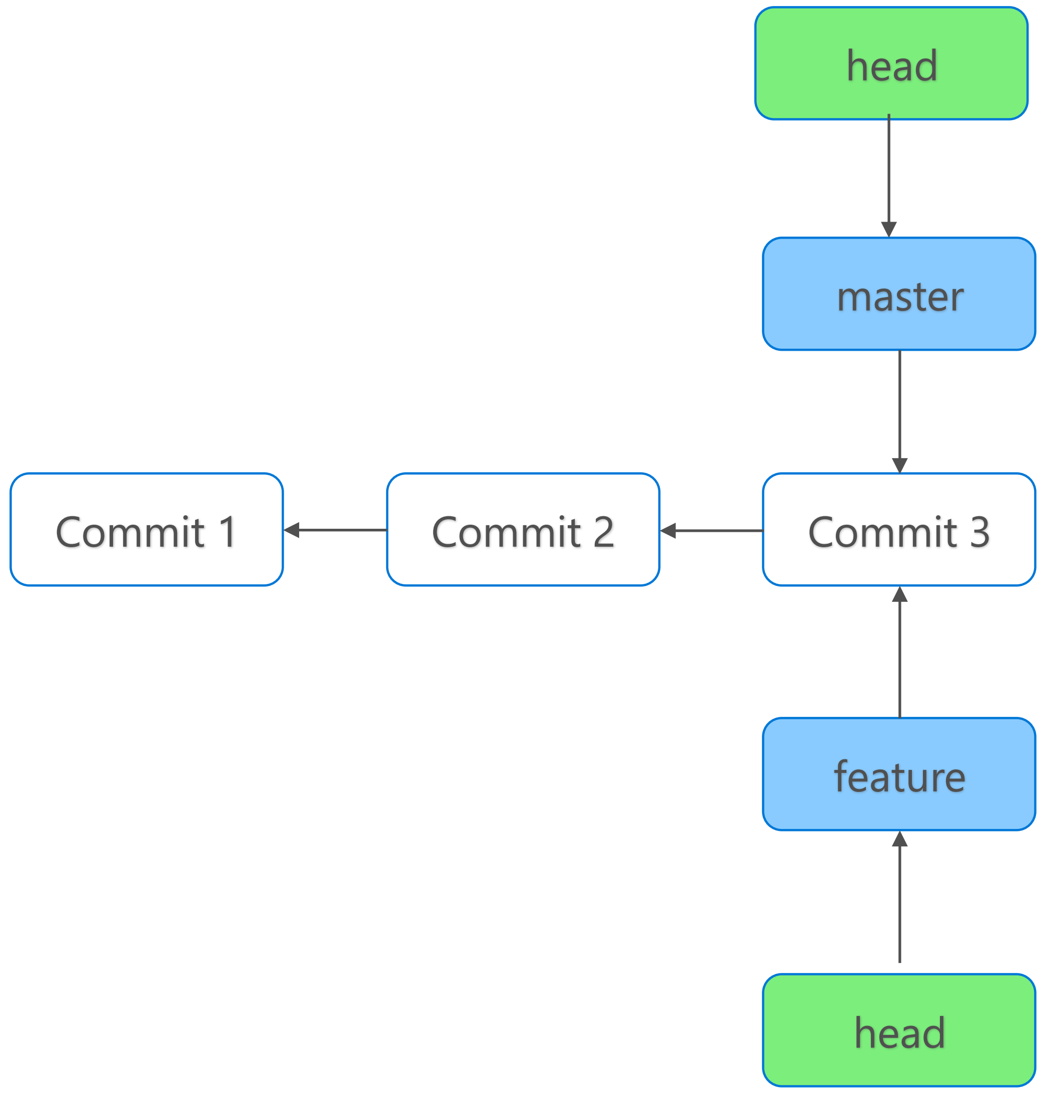
- Switch contexts, suspend work, and isolate risk
- In Git
  - the value of branching is higher
  - the complexity and cost lower, and
  - development teams are encouraged to branch (often)
- Some people create a "topic" branch for each task they perform
  - When satisfied with the work, they merge it back into the master branch
  - You have the option to publish the branch into a remote repository (such as a Git team project) to collaborate with others.

Branching strategies - Topic branch

- A topic is a bug or feature that a developer is working on until it's ready to be merged into a main branch.
- Not uncommon to create and delete multiple topic branches during a single day
- Why use topic branches?
  - Topic branches allow you to context-switch quickly and completely.
  - Easier to identify history and changes during code review.
  - Allows for merge when ready, regardless of the order in which they were created or worked on.
  - Feature development takes place in a dedicated branch instead of the master branch.
  - Enables multiple developers to work on a feature without disturbing master.
  - Master branch should be pristine; Leverage Pull Request.



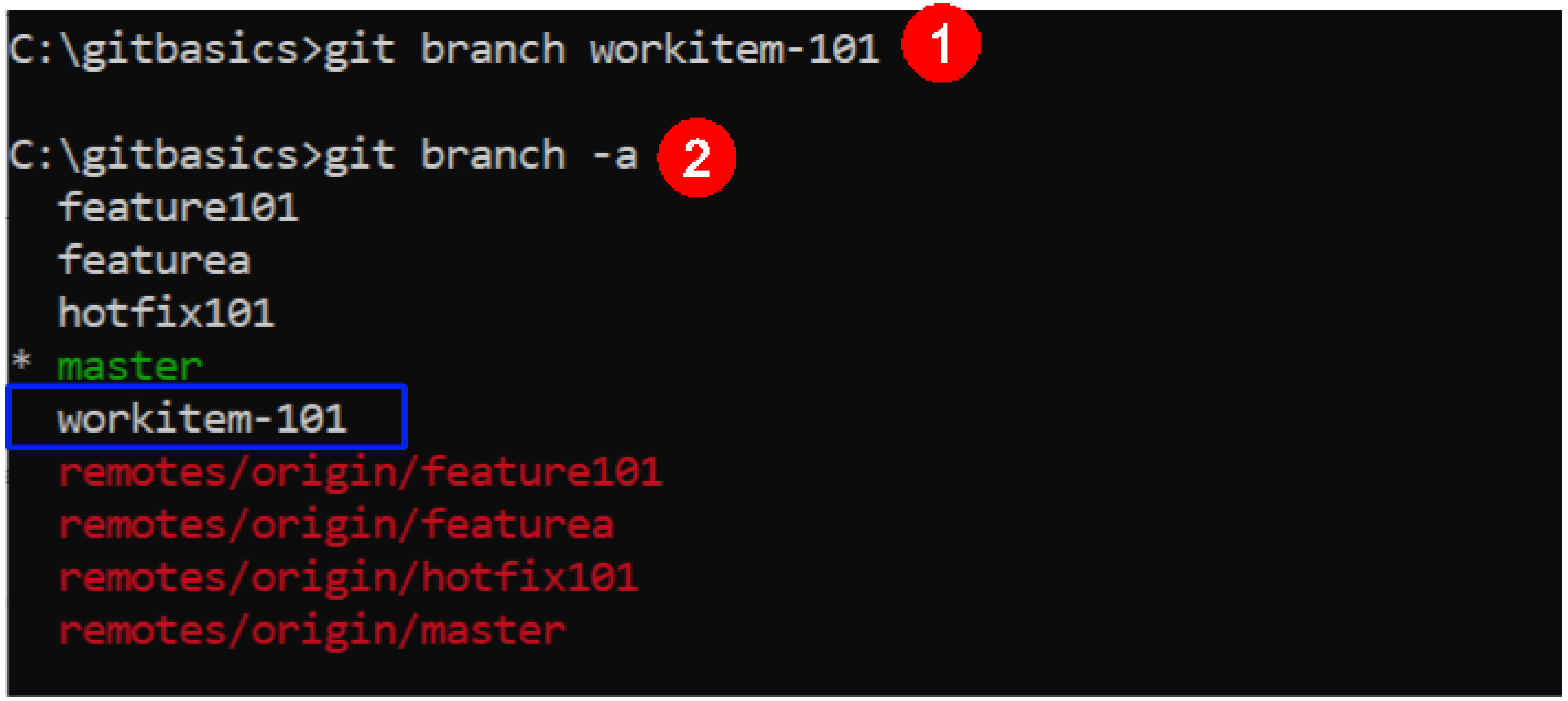
Branches are just pointers to commits



Create Branch – Local

`git branch <new-branch>`

Creates a new branch on your local repo



```
C:\gitbasics>git branch workitem-101

C:\gitbasics>git branch -a
  feature101
  featurea
  hotfix101
* master
  workitem-101
  remotes/origin/feature101
  remotes/origin/featurea
  remotes/origin/hotfix101
  remotes/origin/master
```

Create Branch – Local

```
git checkout -b <new-branch>
```

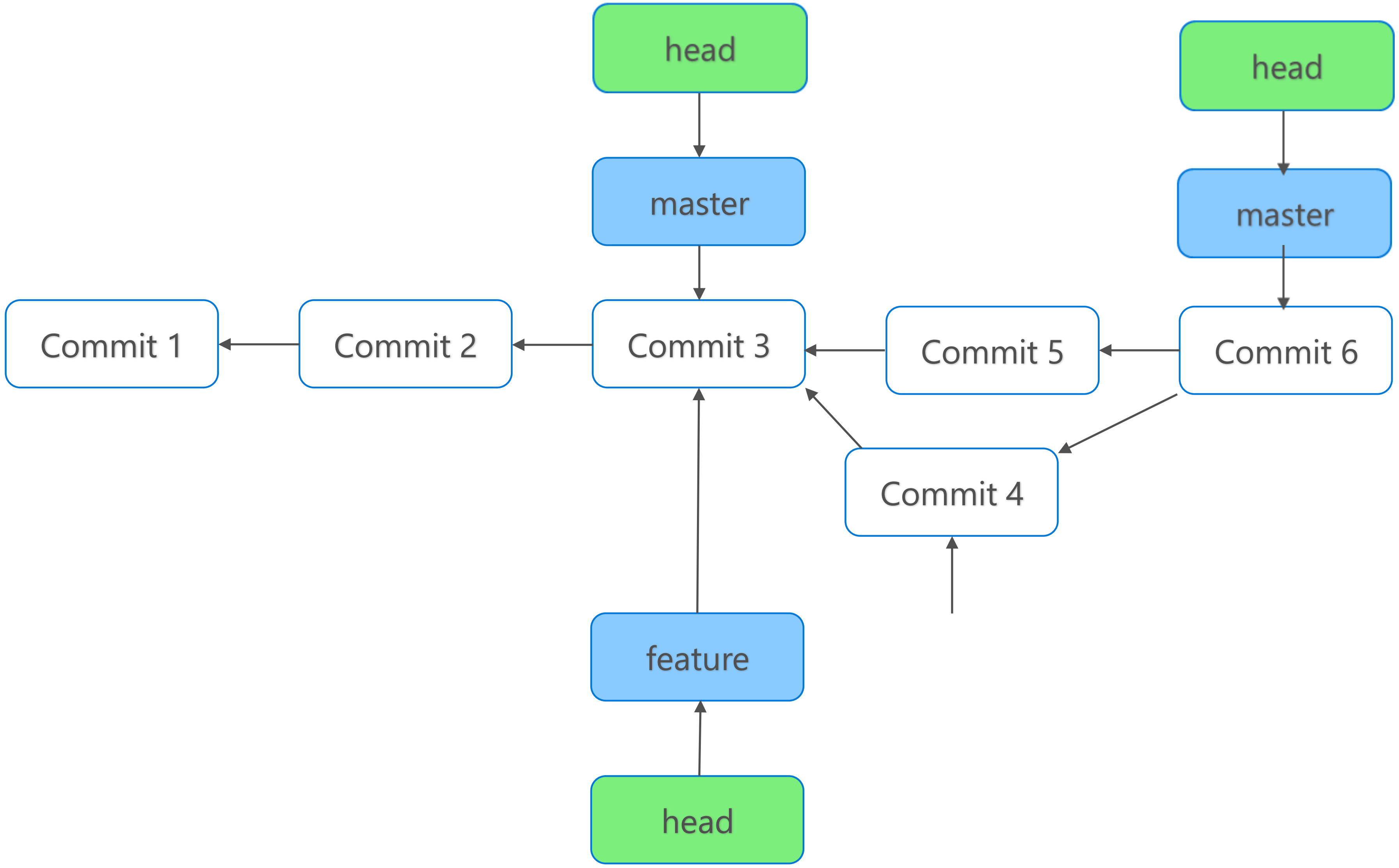
Creates a new branch on your local repo and checks it out

```
git checkout -b abranch
```

=

```
git branch abranch
git checkout abranch
```

Merge Commit



```
git checkout –b feature
git commit
git checkout master
git commit
git merge
```

Branching – Common Options

git branch	Lists all the branches in your local repository
git branch <name>	Creates a new branch called <name>
git branch –d <name>	Deletes the branch with the specified name
git branch –D <name>	Force deletes the branch, even if uncommitted changes
git branch -a	Lists all the remote branches

Merging – Common Options

`git merge <branch>`	Merges commits from the requested branch to the branch you are currently on
`git merge <branch> --no-ff`	Merge commit even when fast-forward would be possible
`git merge <branch> --squash`	Combines all the commits into a new single commit on the branch being merged to
`git merge –-abort`	If conflict occurs, you can abort the merge with this command

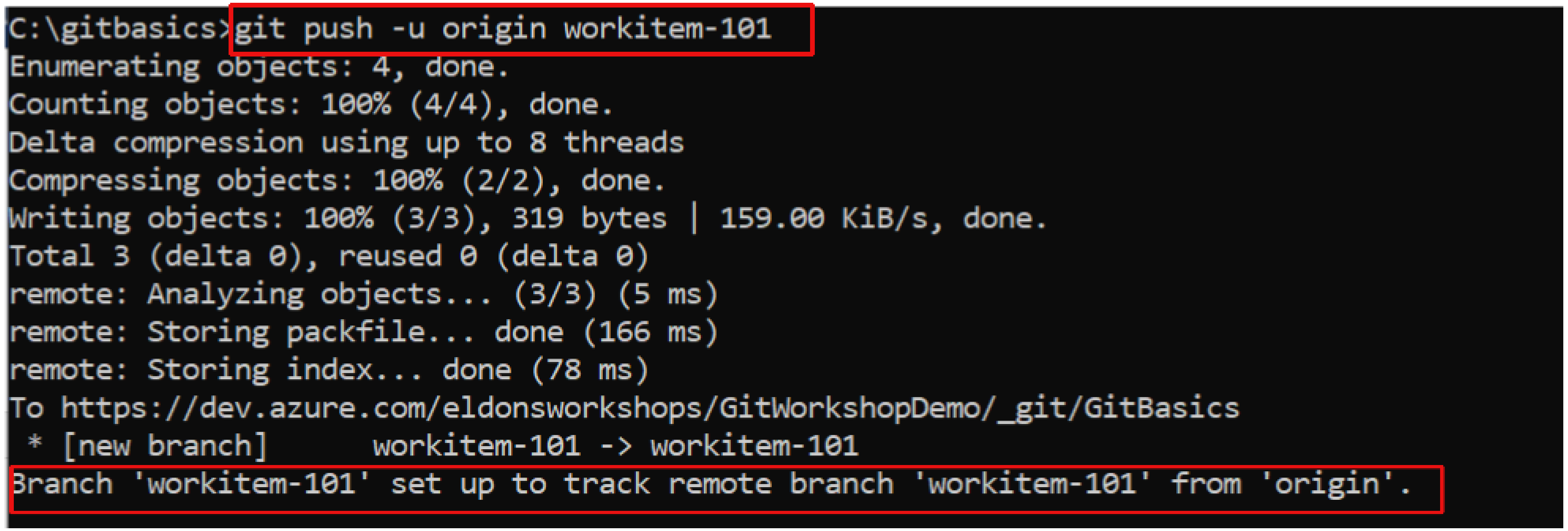
Push branch

git push origin <new-branch>

Pushes your new branch to the remote repo

git push -u origin<new-branch>

Pushes the branch and sets up tracking



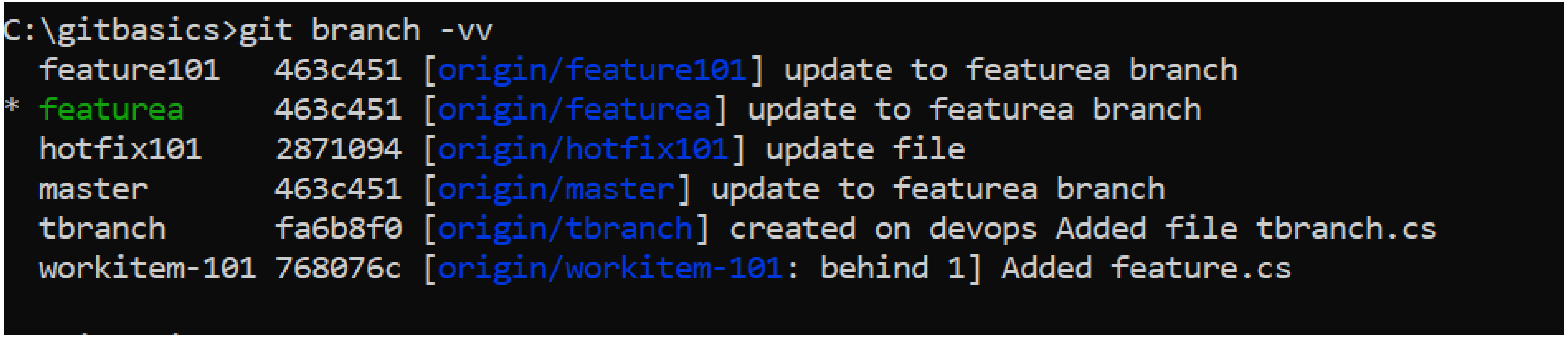
```
C:\gitbasics>git push -u origin workitem-101
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 319 bytes | 159.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: Analyzing objects... (3/3) (5 ms)
remote: Storing packfile... done (166 ms)
remote: Storing index... done (78 ms)
To https://dev.azure.com/eldonsworkshops/GitWorkshopDemo/_git/GitBasics
 * [new branch]      workitem-101 -> workitem-101
Branch 'workitem-101' set up to track remote branch 'workitem-101' from 'origin'.
```

Tracking Branches

Tracking branches are local branches that have a direct relationship to a remote branch

git branch -vv

Shows list of tracking branches you have setup



```
C:\gitbasics>git branch -vv
  feature101   463c451 [origin/feature101] update to featurea branch
* featurea     463c451 [origin/featurea] update to featurea branch
  hotfix101    2871094 [origin/hotfix101] update file
  master       463c451 [origin/master] update to featurea branch
  tbranch      fa6b8f0 [origin/tbranch] created on devops Added file tbranch.cs
  workitem-101 768076c [origin/workitem-101: behind 1] Added feature.cs
```

Merge Conflicts – How to identify

Git fails during the merge

```
git merge new_branch_to_merge_later
Auto-merging merge.txt
CONFLICT (content): Merge conflict in merge.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Git fails during the merge

```
error: Entry '<fileName>' would be overwritten by merge. Cannot merge. (Changes in staging area)
```

Resolve Merge Conflicts

Git marks merge conflicts in the file with dividers

```
<<<<<<< HEAD
this is some content to mess with content to append
=======
totally different content to merge later
>>>>>>> new_branch_to_merge_later
```

Resolve Merge Conflicts

Fix the conflict and remove conflict dividers

```
this is some content cleaned up with content to append
and it is now complete
totally different content to merge later
```

Commit Merge Conflicts

Add and commit the final merged file

```
git add merge.txt
Git commit –m “Conflict resolved, ready to merge”
```

Merge Conflicts Abort

Exit merge process and return the branch to the state before the merge began.

```
git merge --abort
```

Branching Best Practices

- Keep changes small, isolated, and merge often
- Delete unwanted branches
- Pull before you Push

Demo: Branching

Branching and Merging

Lab: Branching

Exercise 1: Managing Branches in Visual Studio

Exercise 2: Managing Remote Branches using Azure DevOps

Microsoft