Microsoft Git Solutions

Conditions and Terms of Use

Copyright and Trademarks

Microsoft Git Solutions

Agenda

Microsoft TFS/Git

Agenda

- Overview
- Git Tools & Clients
- Use Command Prompt
- Migrate TFS Skills to Git
- Set up Git
- Develop using Git
- Collaborate using Git
- Use Branches
- Resolve Conflicts
- View History and Tags
- Pull Request
- Branch Policies
- Merge or Rebase
- Git Permissions
- Git Network
- Repo Rename
- Move a Repo to another Team Project
- New Git Features in Azure DevOps
- Hands on Labs

Microsoft Git Solutions

Overview

Git

Git is a **distributed version control** system that was primarily designed for managing source code, but can be used to track any kind of files.

Git was **created in 2005 by Linus Torvalds** (of Linux kernel fame) with the goals of:

- Creating a truly distributed version control system that didn’t limit teams to a linear workflow
- Not sacrifice performance and ensuring data integrity
- Making it free to all (GNU General Public License 2)

Microsoft Git Support

Our client and server are standard Git implementations

Our client will work with any Git repository

- local, enterprise, GitHub, GitHub Enterprise, BitBucket, Subversion

Azure DevOps will work with any Git client

- Git command lines, XCode, Eclipse, IntelliJ, etc.

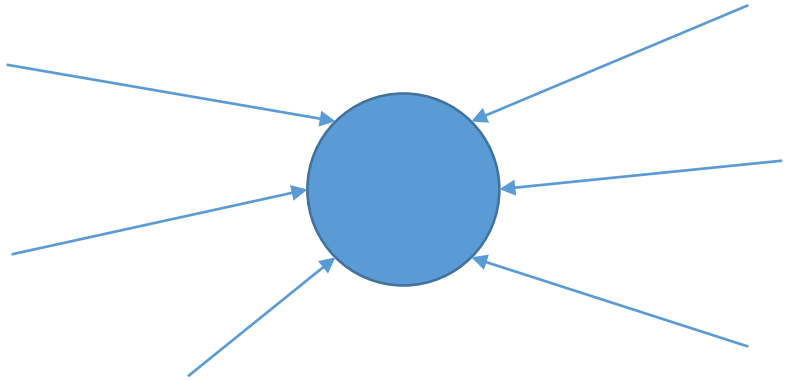
Our core principle is to provide a good experience for interoperable Git capabilities

Brian Harry's Blog Posts

- Introducing Git Support in TFS
  http://blogs.msdn.com/b/bharry/archive/2013/01/30/git-init-vs.aspx
- Microsoft Joins the Eclipse Foundation
  https://blogs.msdn.microsoft.com/bharry/2016/03/08/microsoft-joins-the-eclipse-foundation/

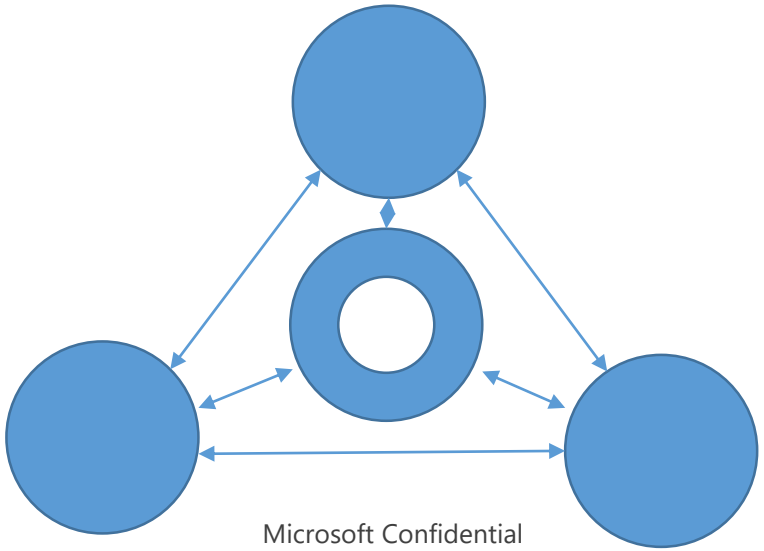
Centralized Version Control System

- Most version control systems are centralized version control systems (CVCS). In a CVCS, a single, shared repository is used to maintain an authoritative copy of the source code, and all changes are made against the central copy. Individuals have local copies of the files which they modify and send back to the server to share with others.
- Team Foundation Version Control (TFVC) is a centralized, server-based version control system.
  - TFVC enables two modes of operating against the mode uses Server Workspaces, where the server must be contacted before any developer can add, edit, rename, delete, or even diff their changes against the original.
  - The second (default) mode uses Local Workspaces, where the additional metadata on the developer's machine tracks changes to files, but the server must still be contacted to update files and submit changes. Local workspaces offer greater flexibility for working with source control while offline, and without good communication can lead to additional merging before changes can be checked in. NOTE: Metadata does NOT include complete file history.
- In addition to TFVC, there numerous other CVCSs – including systems such as Visual SourceSafe5, CVS6 and Subversion7.



Distributed Version Control System

- Git is an open source distributed version control system. Git enables developers to work in two modes
  - By setting up a central copy and working in a hub and spoke model
  - Or, by creating a true peer-to-peer model.
- The first mode uses a central repository. Once the repo is cloned by developers, the central repo need not be contacted to be able to commit and view full history from their local repo. Developers can work in complete isolation and disconnected from the central repo until they are ready to merge their changes with the central repo.
- In addition to Git, there are numerous other DVCSs – including Monotone and Mercurial.



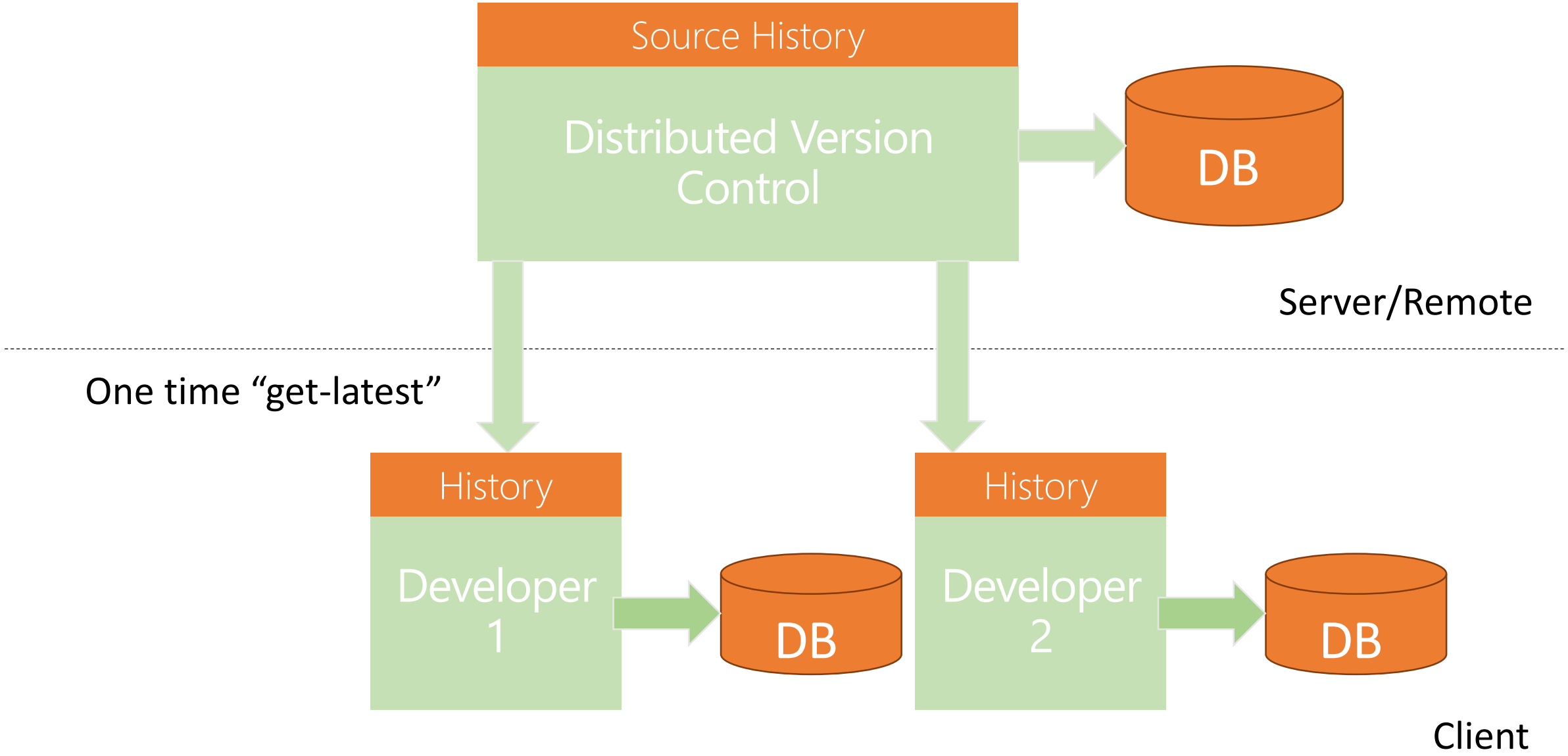
Central or Distributed

- Avoid binary artifacts when using a distributed repository!
  - As all history is cloned to every developer's machine.
  - For large/binary assets, such as videos, libraries and assemblies, DVCS is not the best option for storing binary content.
  - Git Large File Storage (LFS) as an improved way to integrate large binary files such as audio samples, datasets, graphics, and videos into your Git workflow.
    - Support for Git LFS is in TFS 2015 Update 2
  - Git LFS is a newer, open source extension that replaces large files with text pointers inside Git, while storing the file contents on a remote server like GitHub.com or GitHub Enterprise.

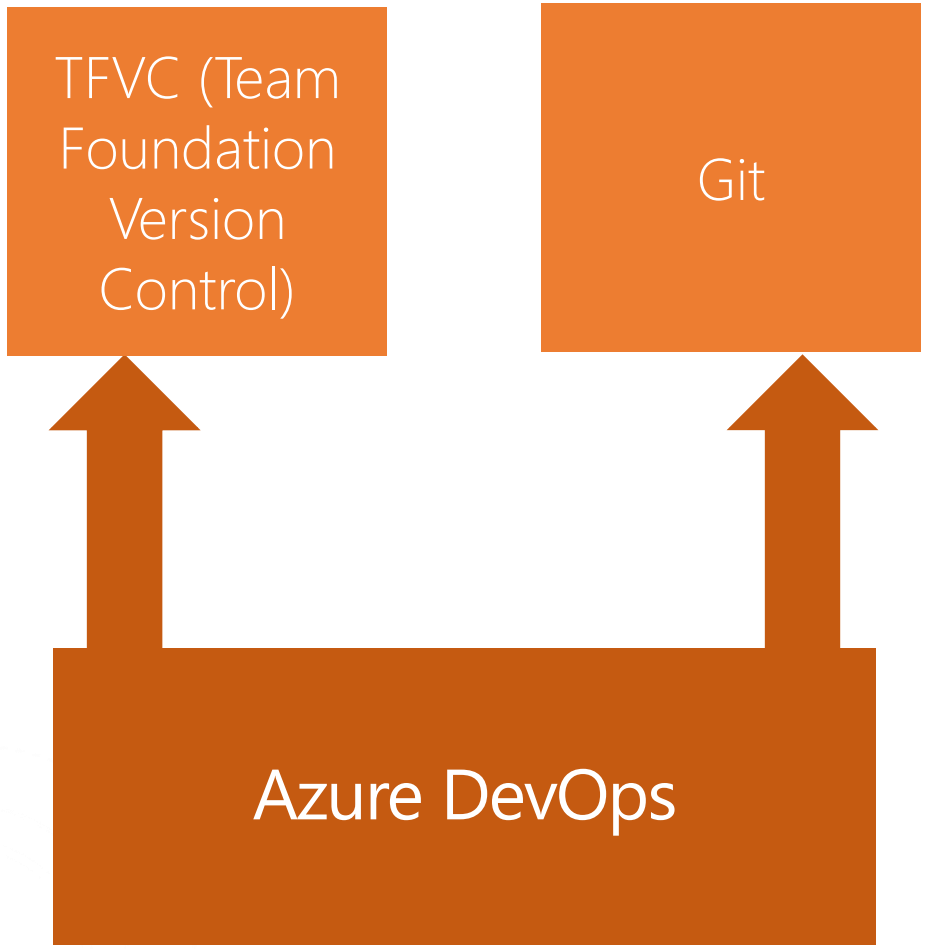
Modern source-control approaches

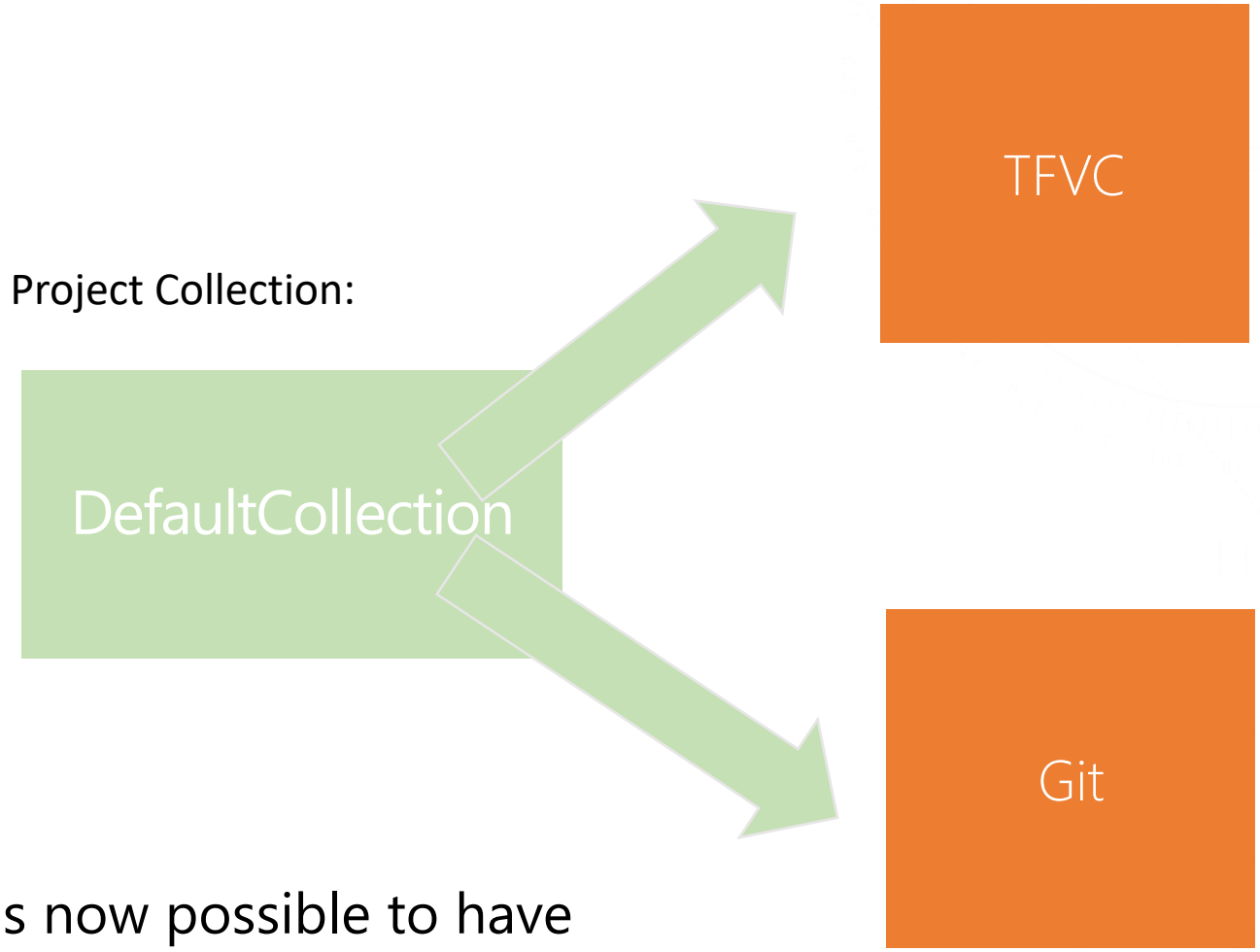
		Strengths	Best for
Centralized Version Control	Check-in Check-out	• Fine level permission control • Allows usage monitoring	• Large integrated codebases • Control and auditability over source code down to the file level
	Edit Commit	• Offline editing support • Easy to edit files outside Visual Studio or Eclipse	• Medium-sized integrated codebases • A balance of fine-grained control with reduced friction
Distributed Version Control (DVCS)		• Fast offline experience • Complete repository with portable history • Flexible advanced branching model	• Modular codebases • Integrating with open source • Highly distributed teams

Distributed Version Control Systems



Azure DevOps Server

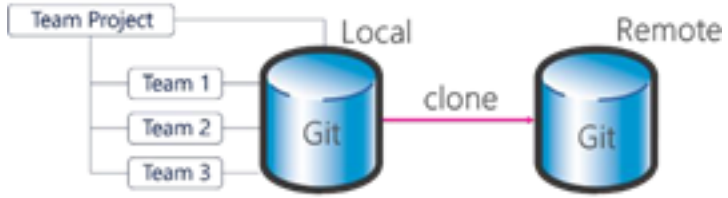




It is now possible to have
Both Git and TFVC repositories
within the **SAME TEAM PROJECT**..

Team Project

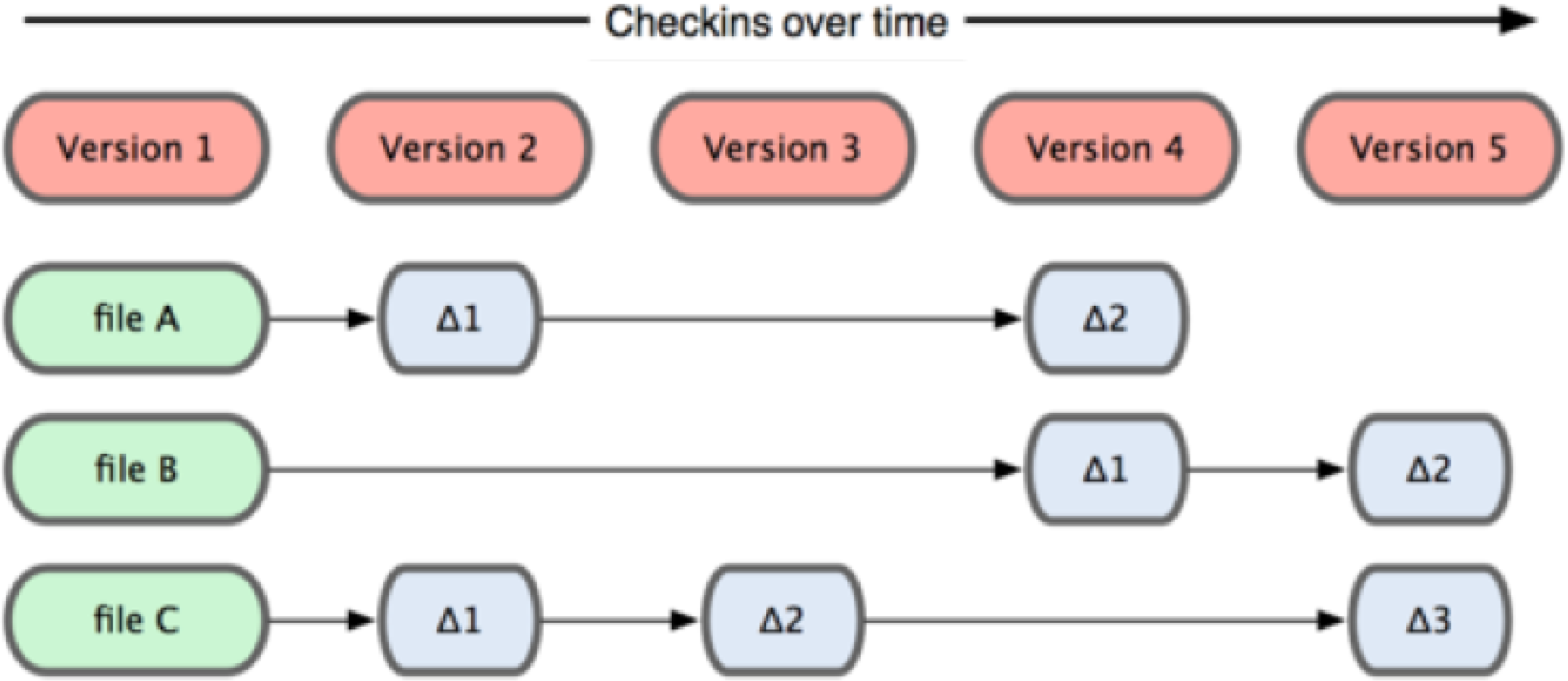
- When you select TFVC as your repository it is shared across other Team Projects. You can scale the repository from small to large codebases with millions of files per branch and large binary files and easily identify changes made by all developers.
- You do not need to create more Team Projects to get more Git repos. You can manage permissions for all Git repos in a single team project in one place – which is a good reason not to create separate Team Projects (TPs) for each repo.
- The decision whether to use a single or multiple repositories should be based on the shape and architecture of the product. Some considerations:
  - Code with dependencies may benefit from a single repository.
  - Multiple small and cohesive repositories can reduce cloning and storage costs, but complicates repo traceability and maintenance.
  - Your TFVC or Git repo is the boundary of atomicity.
    - You cannot commit to two repos atomically.
    - You cannot coordinate code between repos, complicating build/release/versioning processes.
  - Although not recommended, you could combine and embed multiple repos using Git Tools – Submodules. Separate Git repos will not be fetched with a default pull from the main repo, but all of the Submodules can be pulled down during cloning.



TFS vs Git Branching

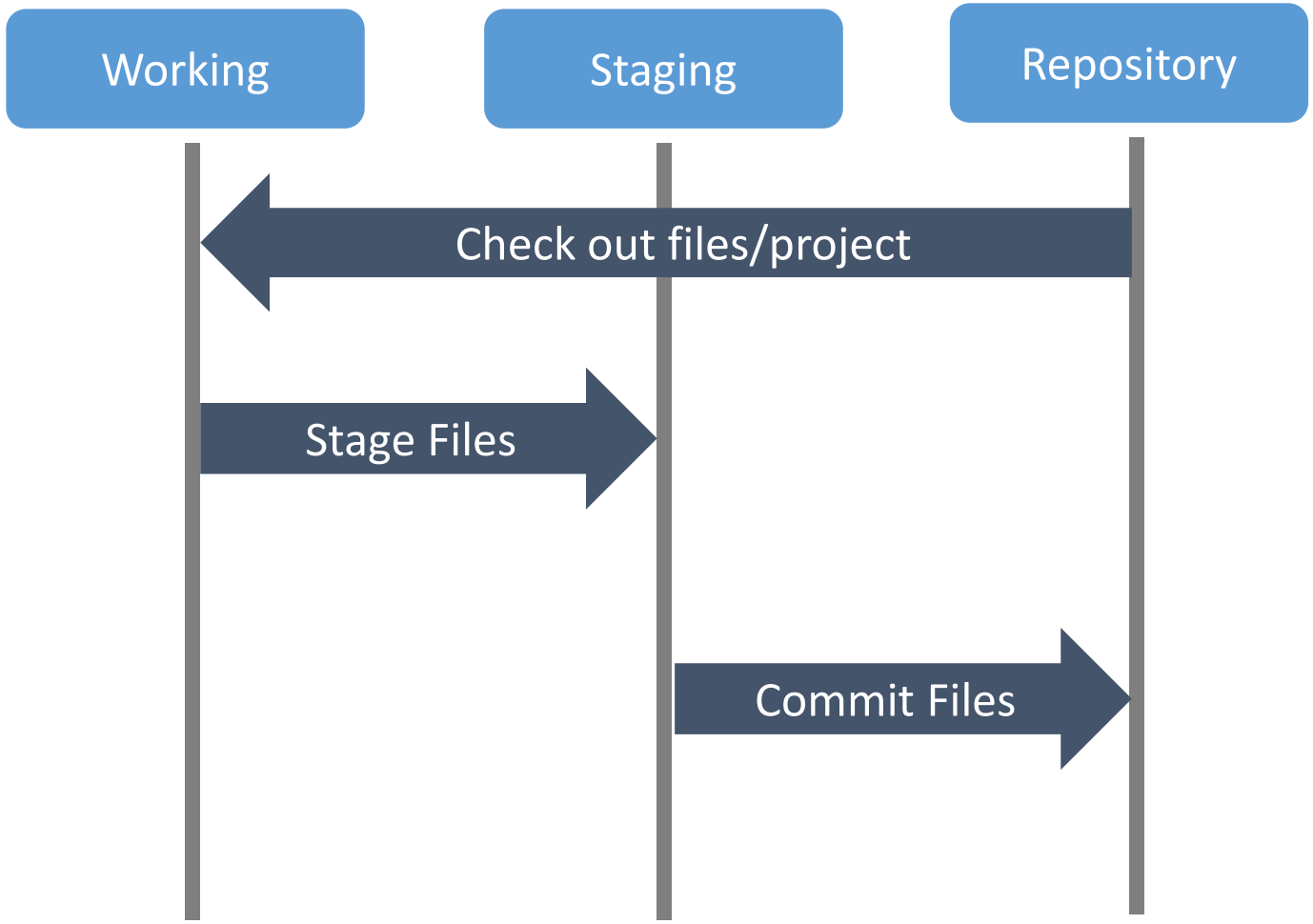
- You need to take a step back and examine how Git stores its data and how it does branching and merging.
  - Every user has a local repository, making status, commit, branch, merge and diff operations fast.
  - A branch is really nothing more than a pointer to a specific commit, and switching branches in Git will swap out the content of your working directory which is fundamentally different from the TFVC experience where your branches live in different folders.
  - Deleting any branch is just deleting the pointer, which helps keeping the branch list clean and bloat of repository minimal.

Most VC use Deltas

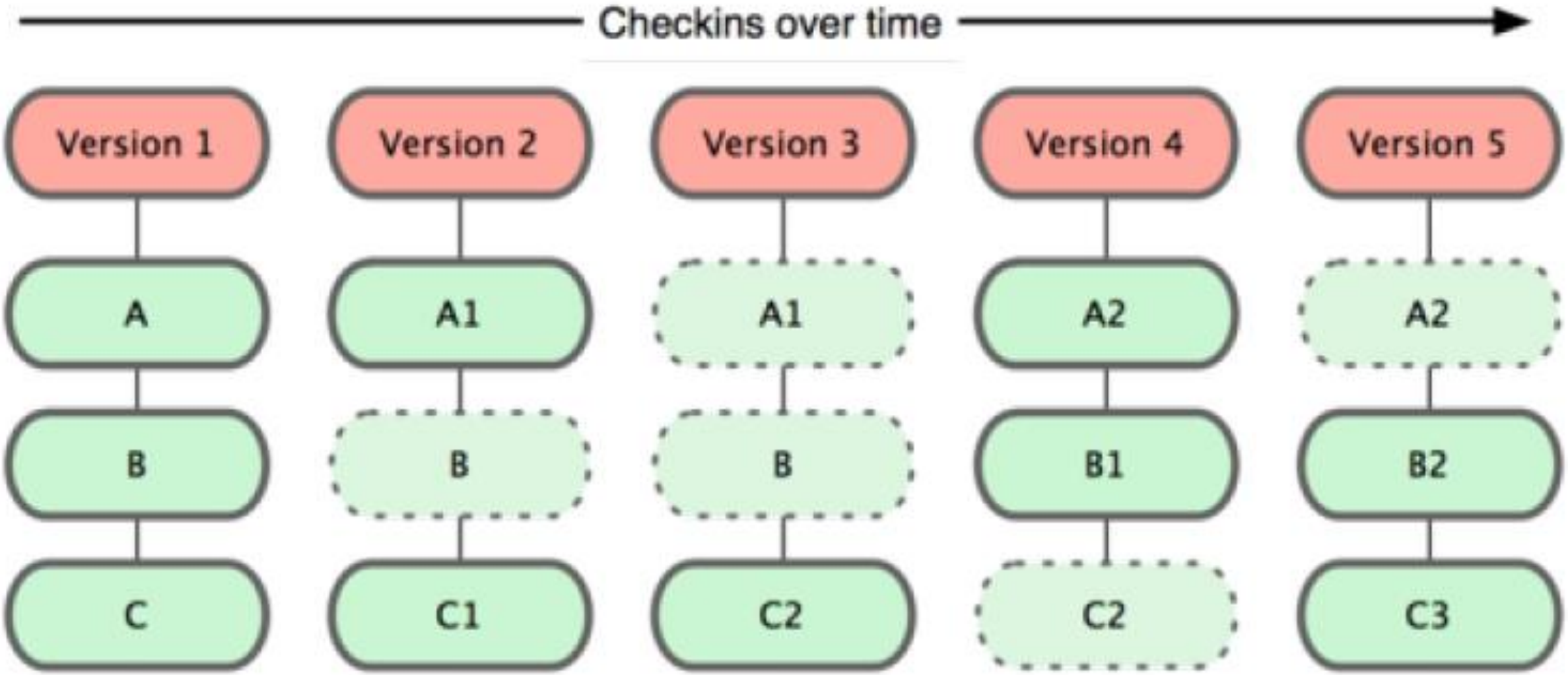


Local Operations

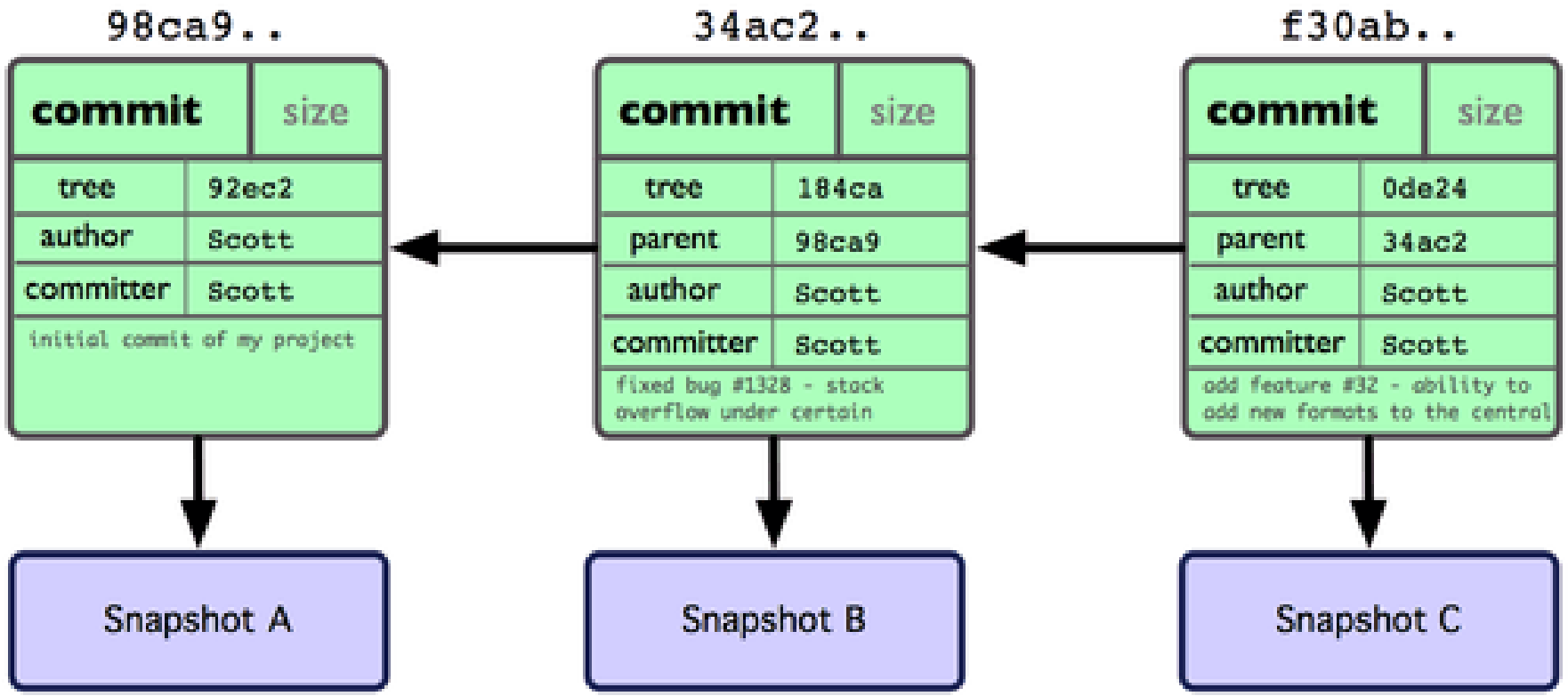
- Three directories
  - Working
  - Staging
  - Repository



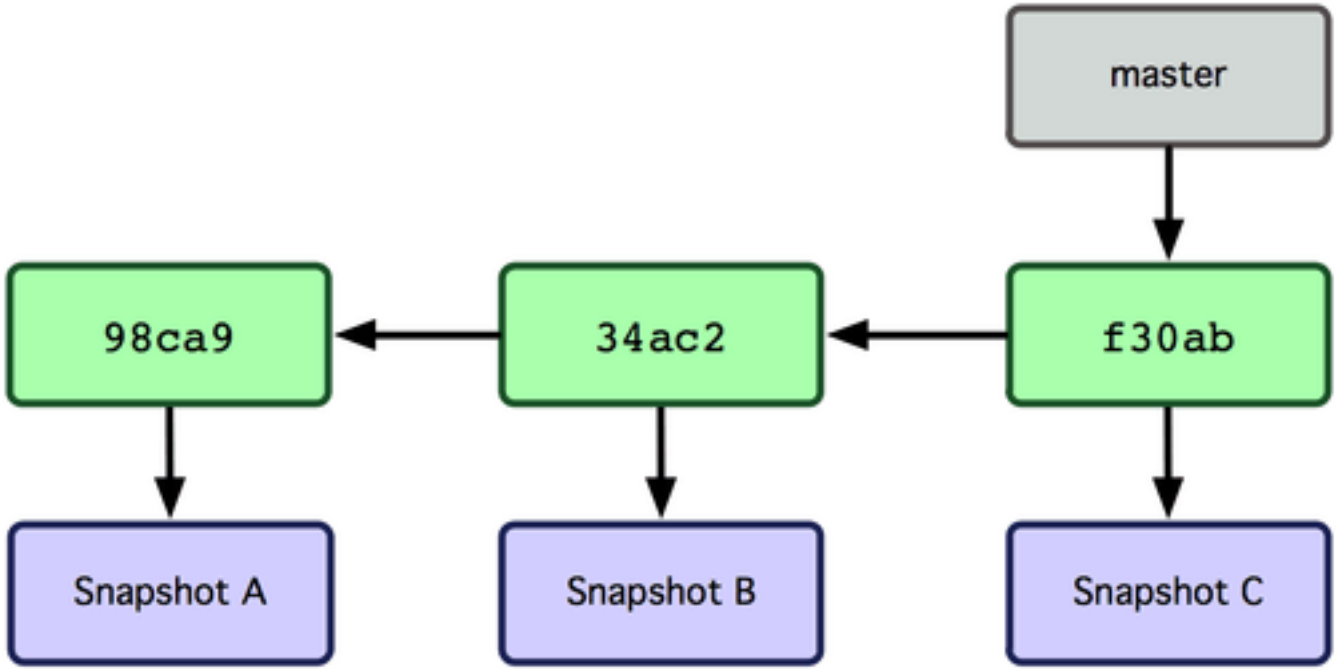
Git uses Snapshots



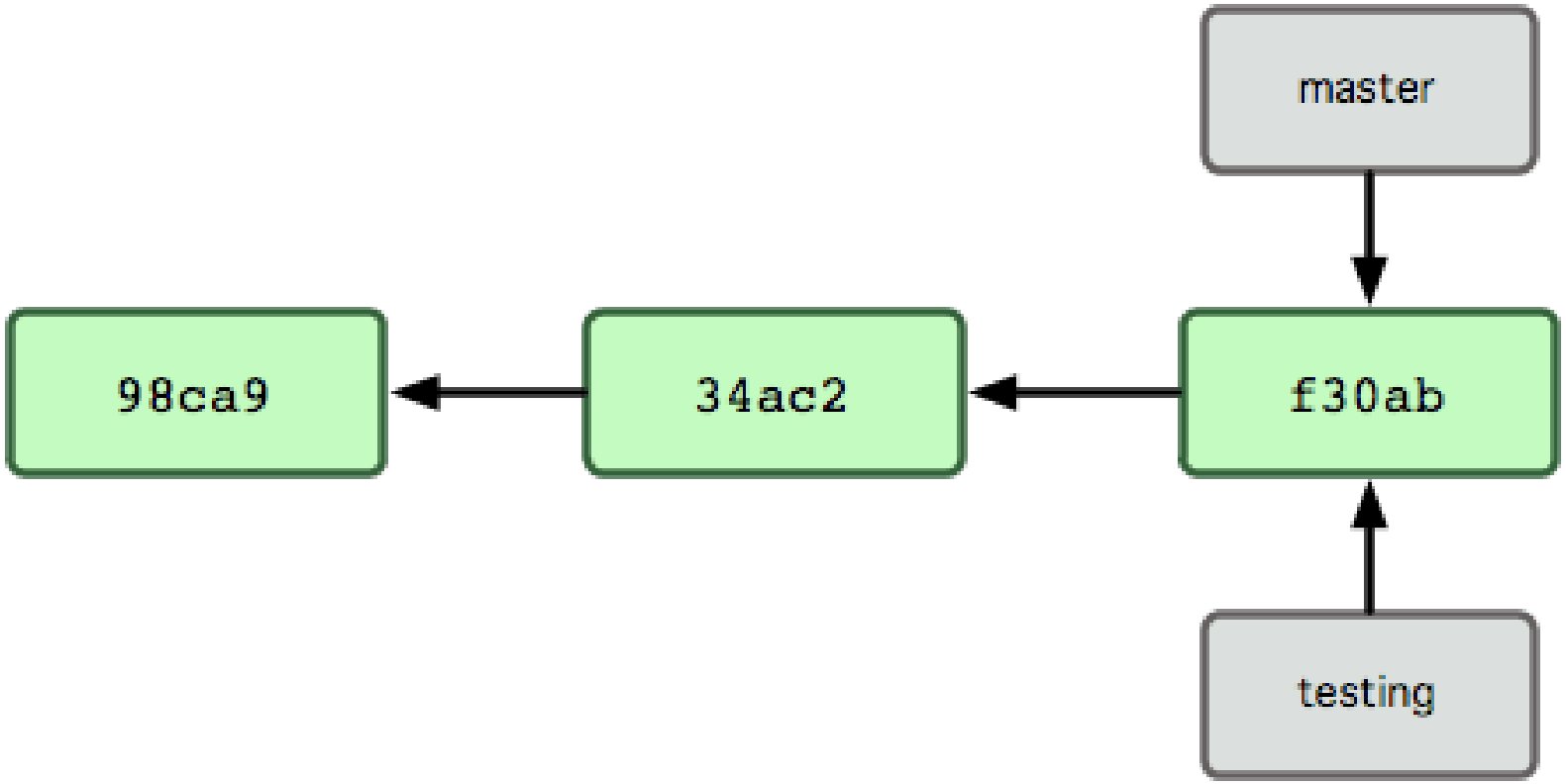
Make Changes and Commit



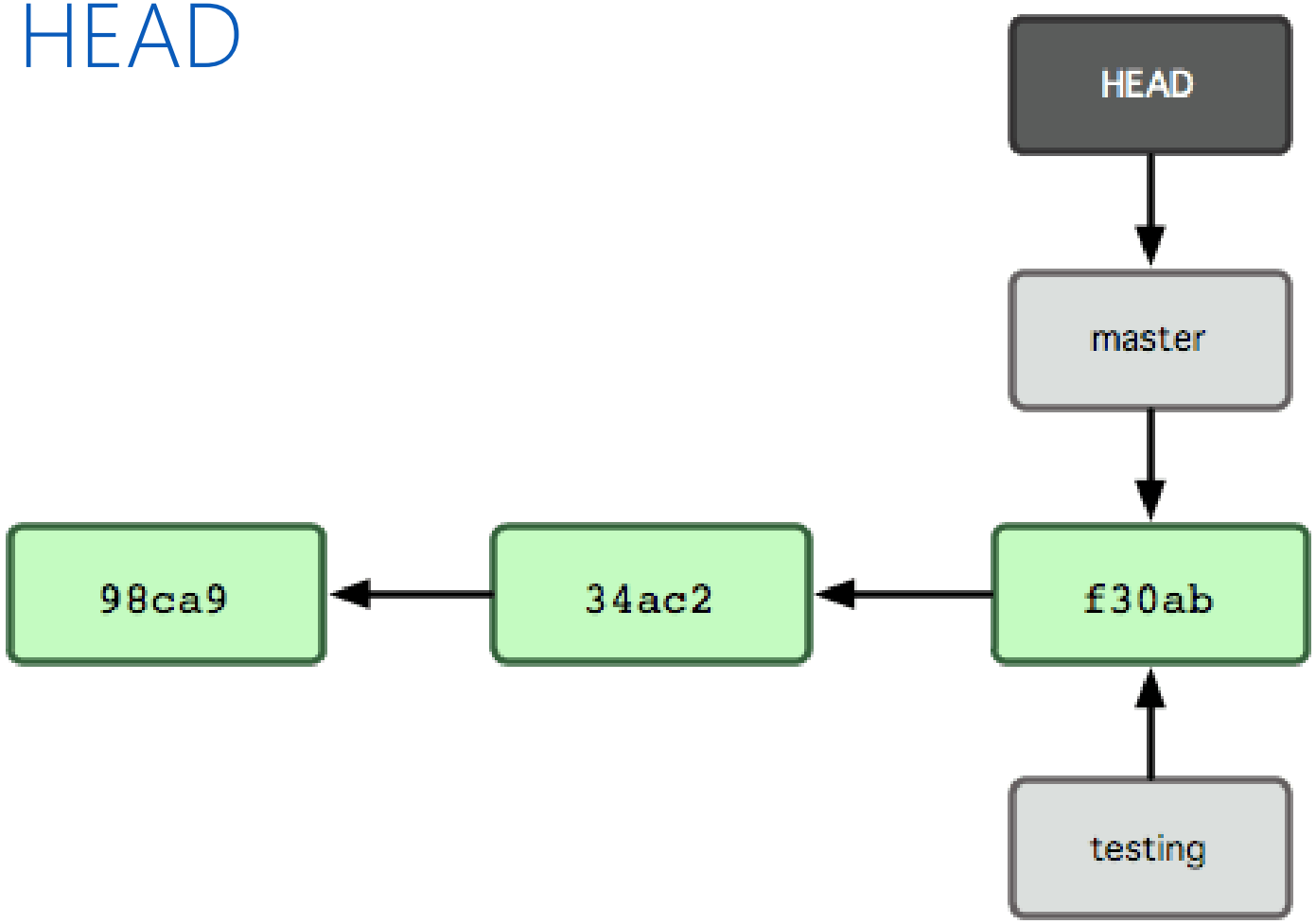
Branch



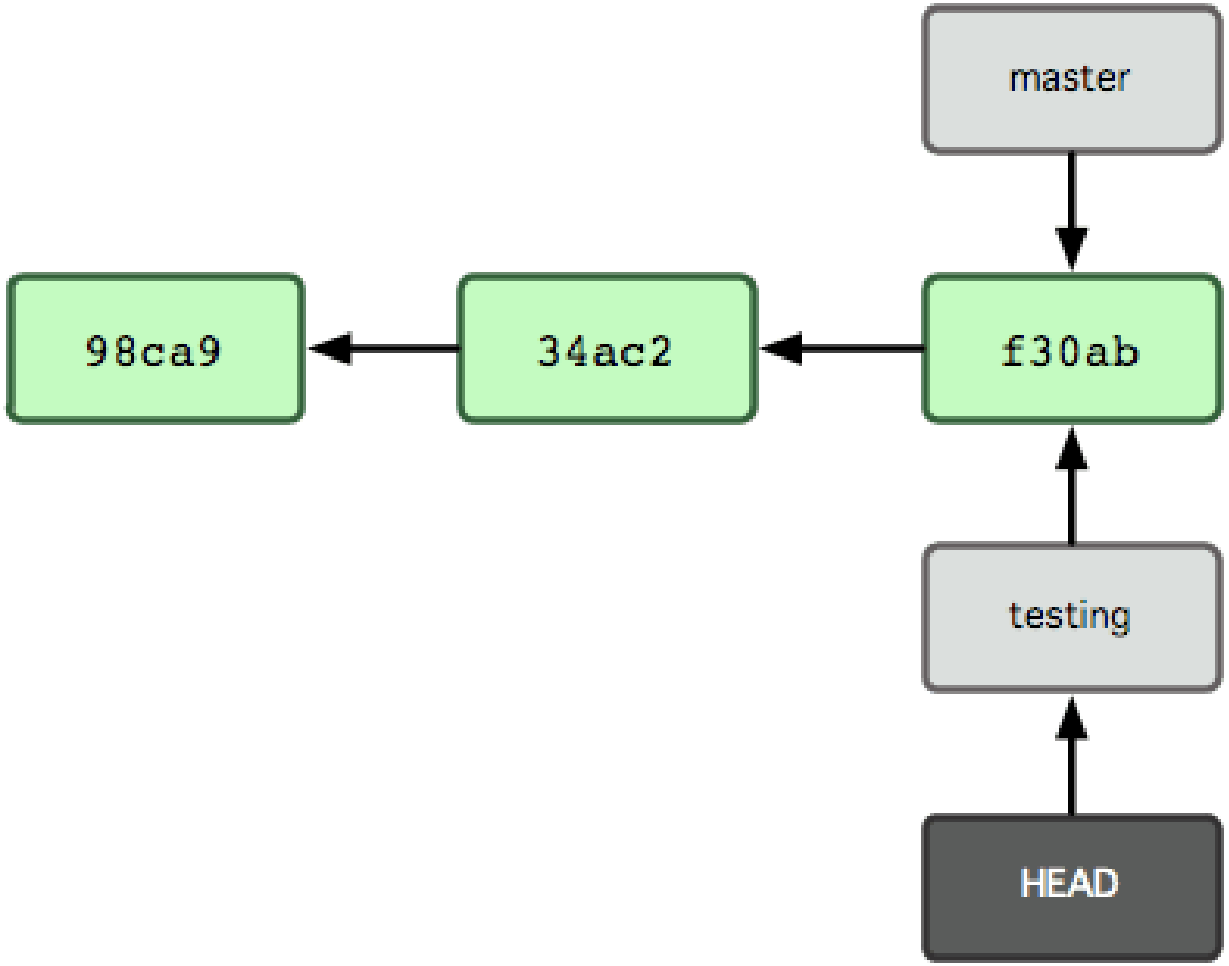
New Branch



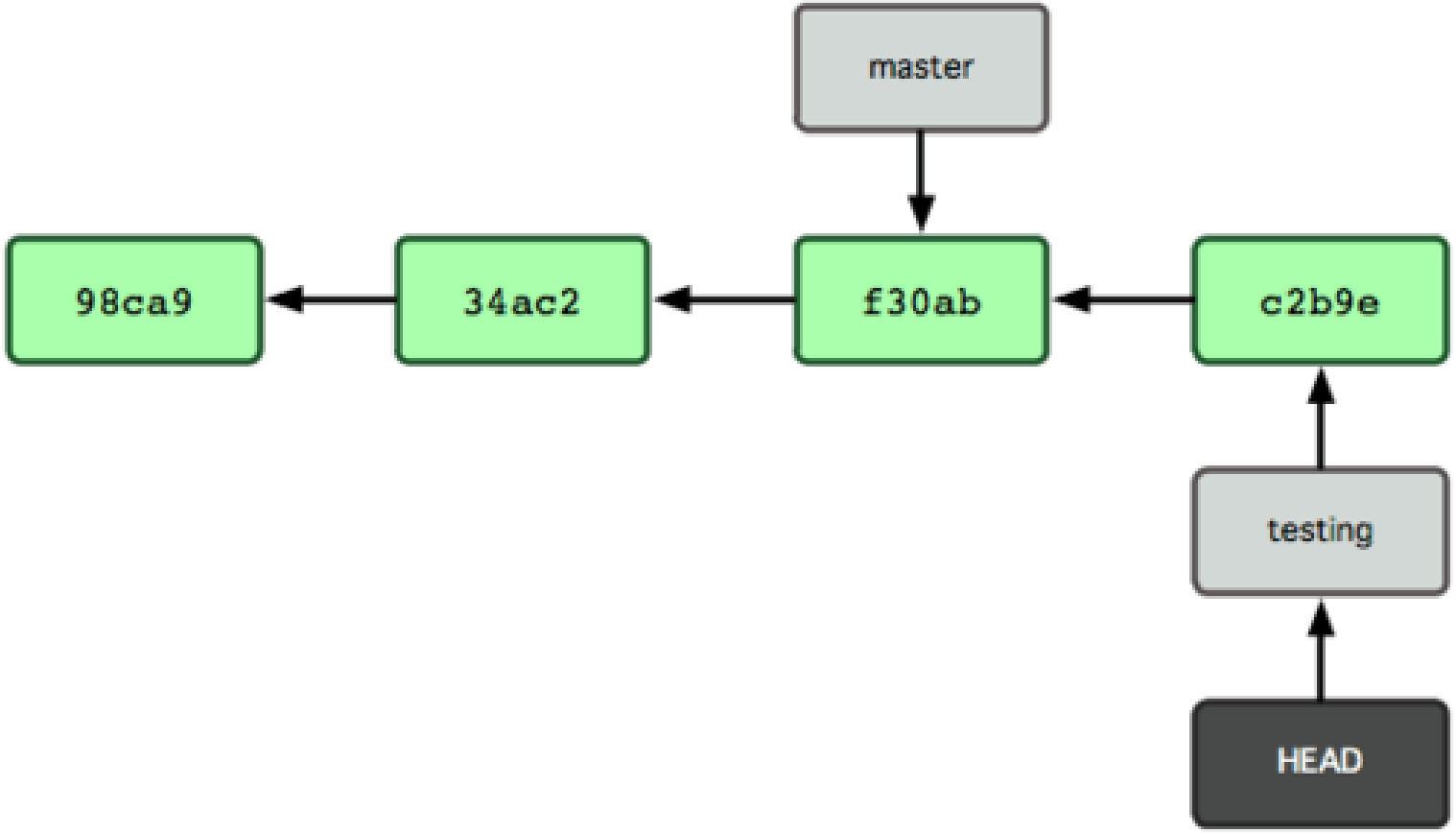
HEAD



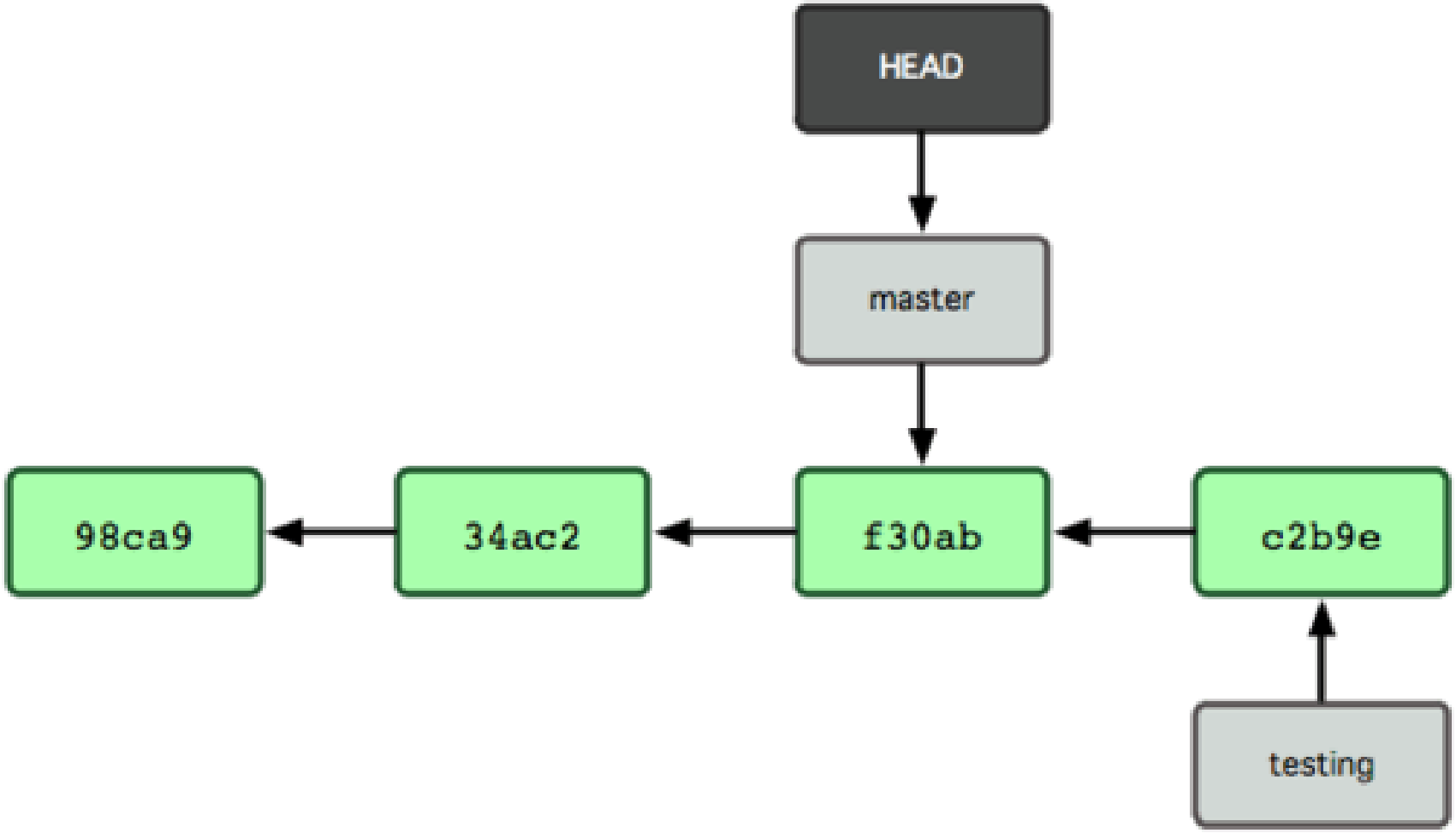
Context Switch – Checkout Testing



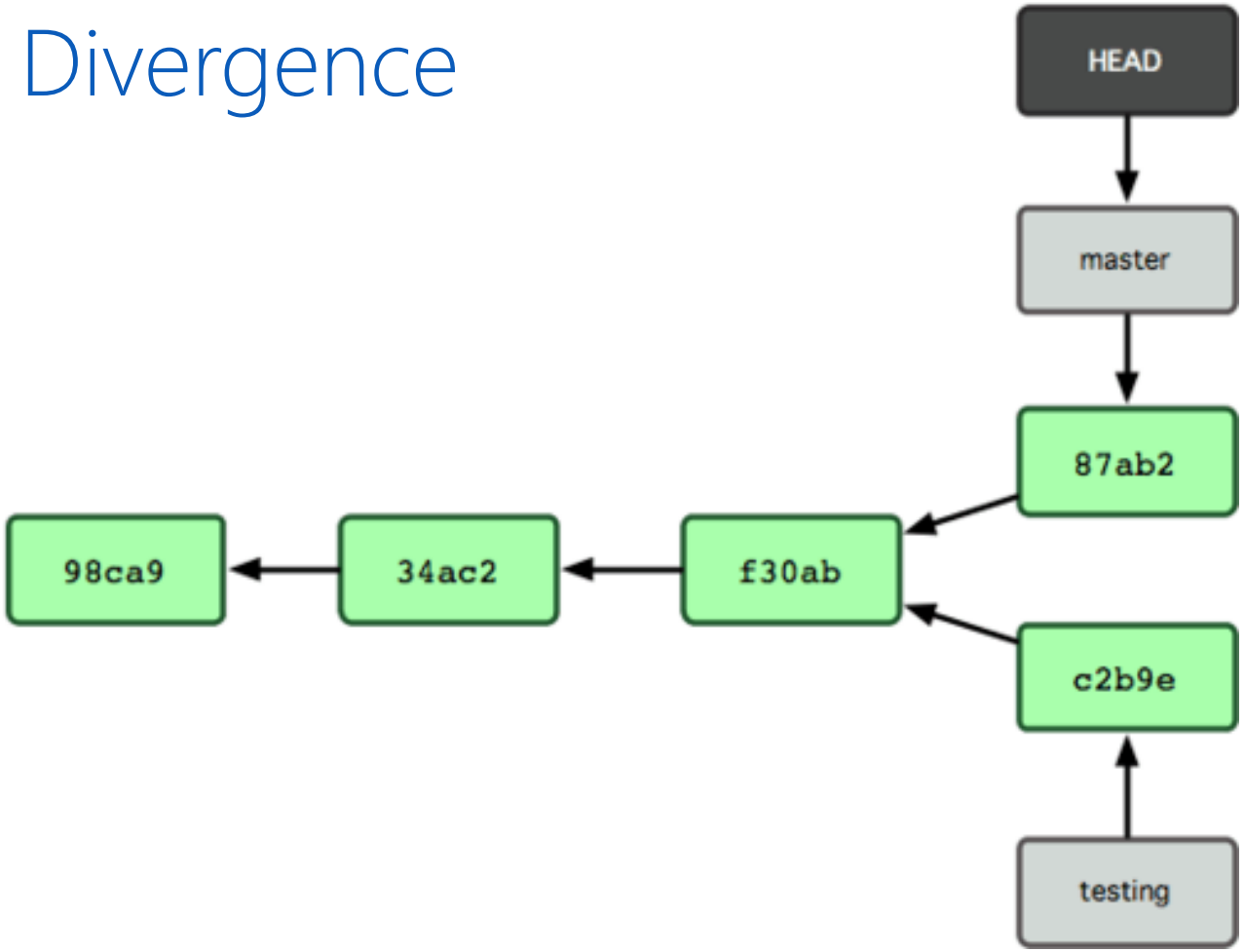
Moving Forward



Context Switch – Checkout Master



Divergence

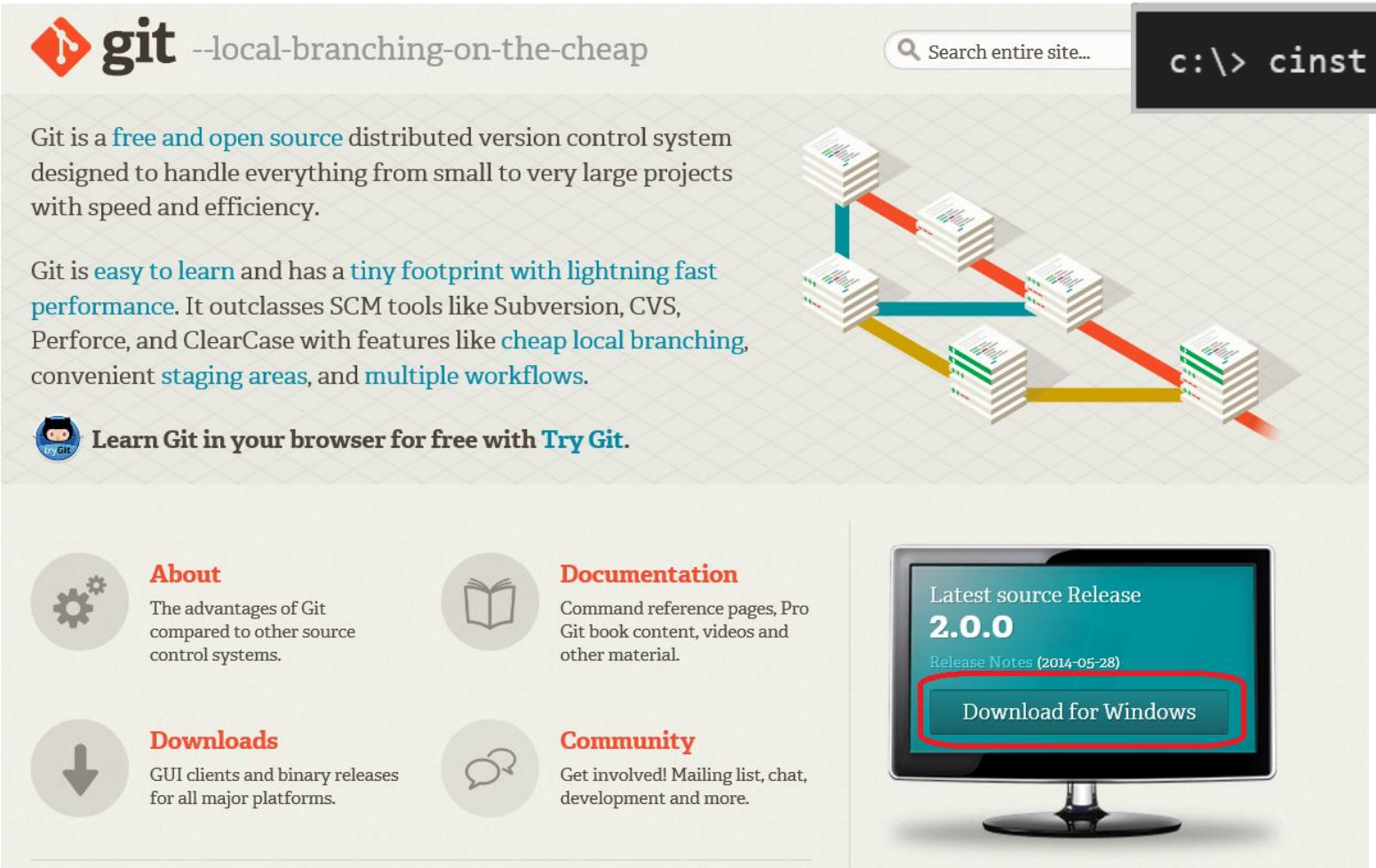


Microsoft Git Solutions

Git Tools & Clients

Get Git for windows (msysGit)

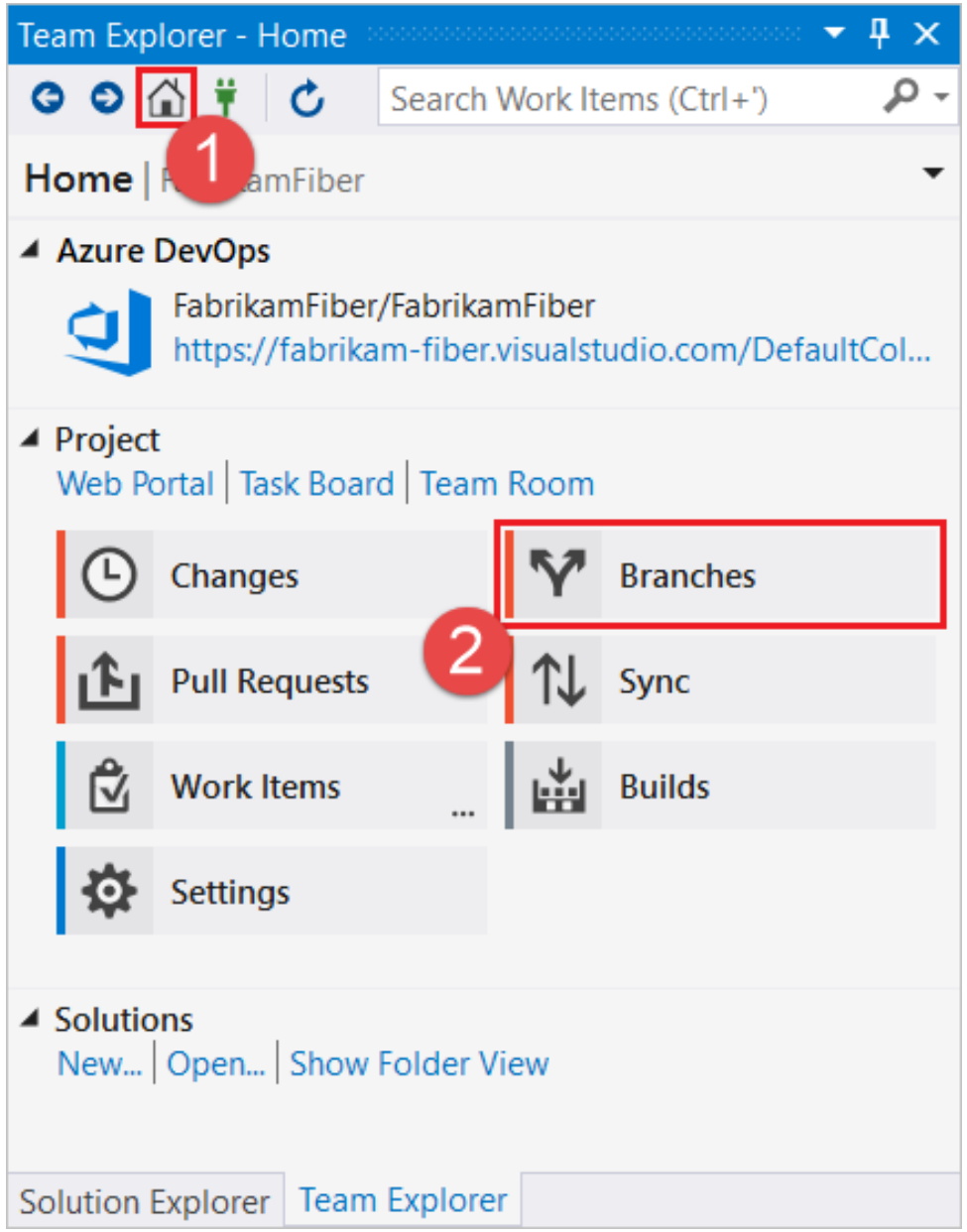
http://git-scm.com




```
c:\> cinst git.install
```

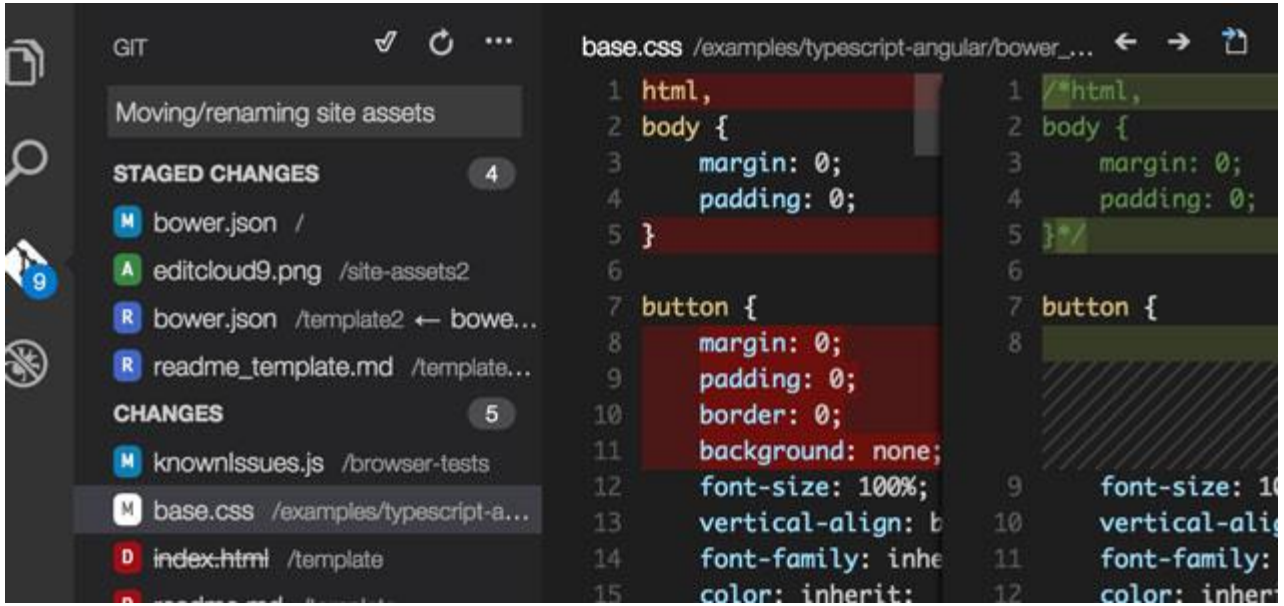
Git Integration in Visual Studio

Visual Studio



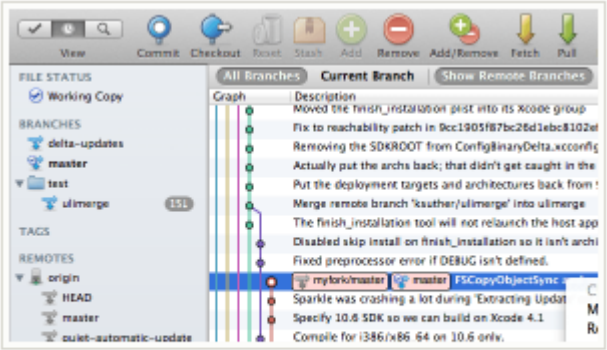
Git Integration in Visual Studio Code

- Visual Studio Code is free and available on your favorite platform — Linux, OS X, and Windows.
  - Code editing redefined, optimized for building & debugging modern web & cloud applications.
  - VS Code Integration – announced at //Build 2015
  - https://www.visualstudio.com/products/code-vs.aspx
  - https://code.visualstudio.com/docs/editor/versioncontrol

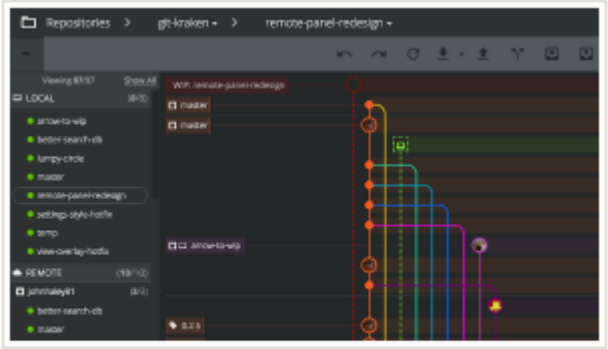


Git GUI Clients

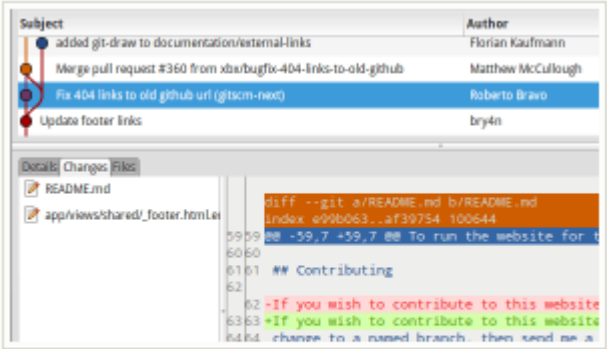
- There are multiple additional GUI Clients available
  - https://git-scm.com/downloads/guis



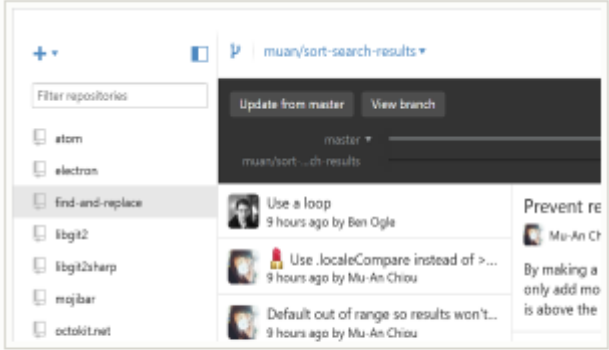
SourceTree
Platforms: Mac, Windows
Price: Free



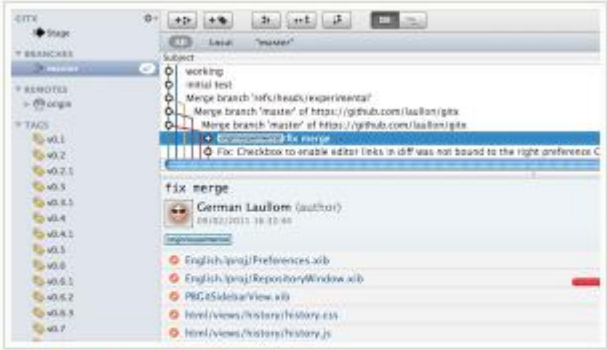
GitKraken
Platforms: Windows, Mac, Linux
Price: Free



gitg
Platforms: Linux
Price: Free



GitHub Desktop
Platforms: Windows, Mac
Price: Free



GitX-dev
Platforms: Mac
Price: Free

Microsoft Git Solutions

Command Prompt

Overview

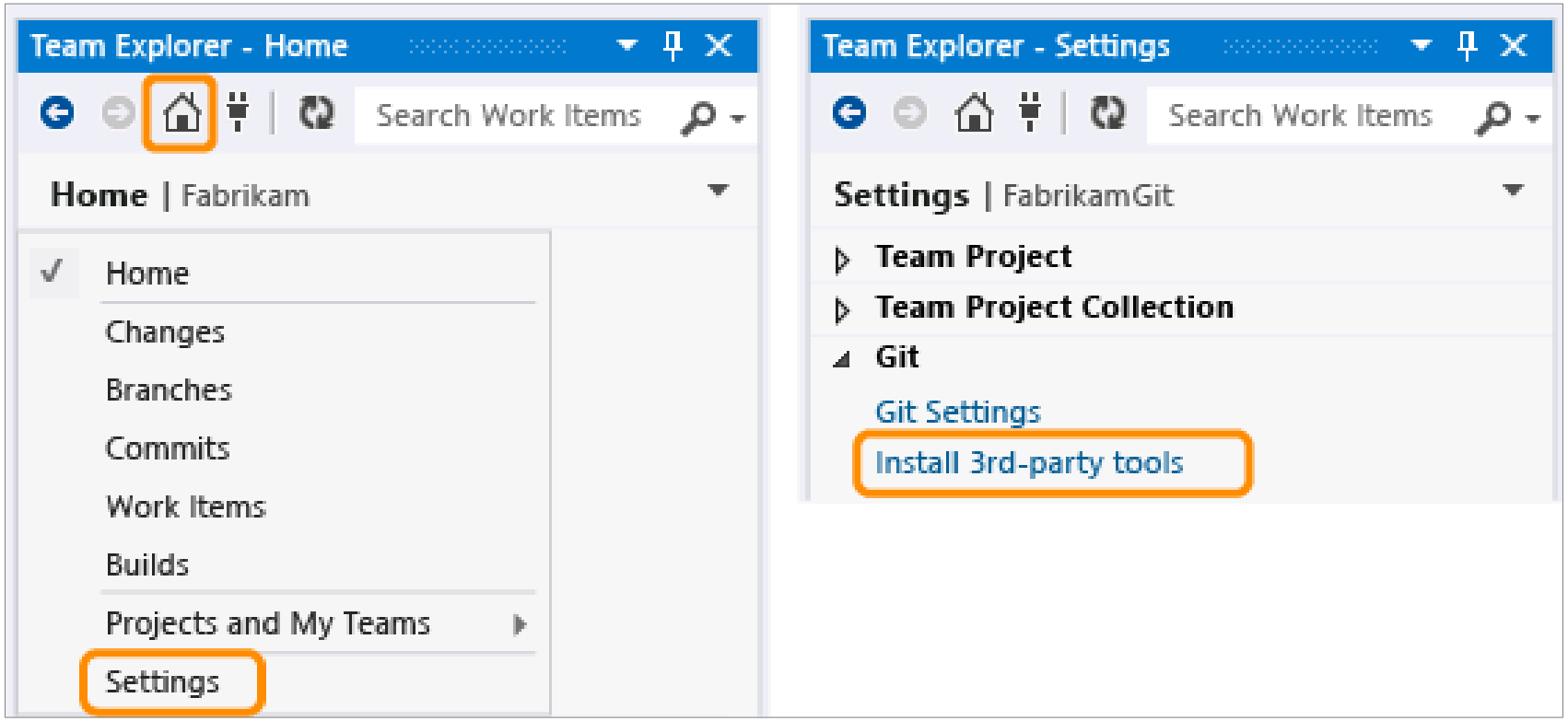
- Operations
- Tools
- Launch the Git Command Prompt

Common Operations

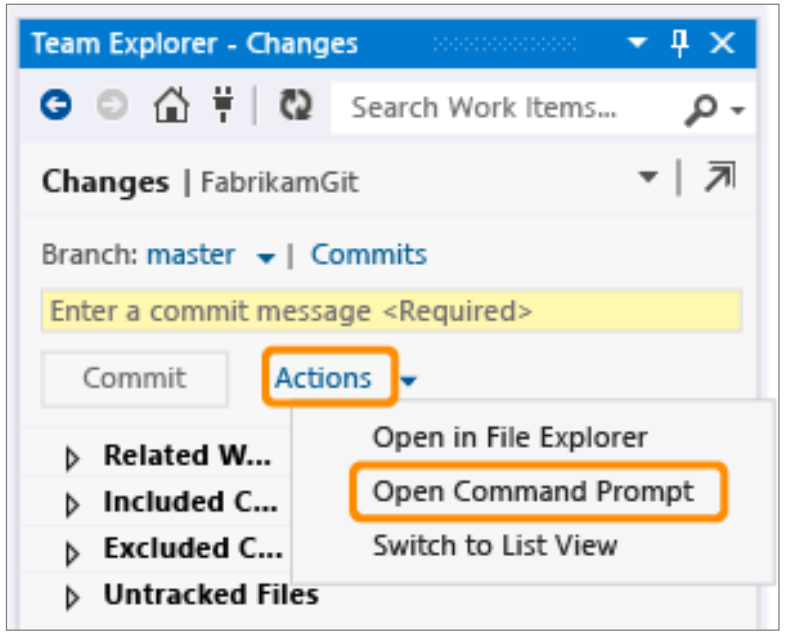
Operation	Command
Create (initialize) a local repository	git init
Clone a remote repository	git clone
Fetch and then pull changes from a remote repository	git fetch, git pull
Manage the set of repositories ("remotes") whose branches you track.	git remote
Stage and then Commit changes	git add, git commit
Undo a committed change	git revert, git reset
Branch and merge/rebase	git branch, git merge or git rebase
Push changes to a remote repository	git push

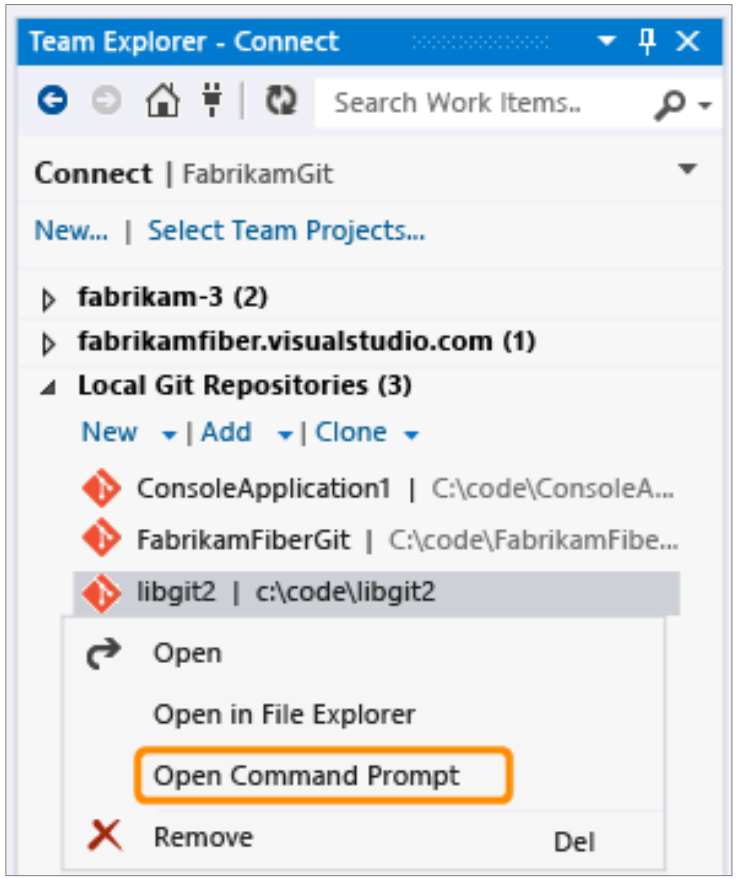
Tools

- Install third–party tools



Launch the Git Command Prompt





Demo: Command Prompt

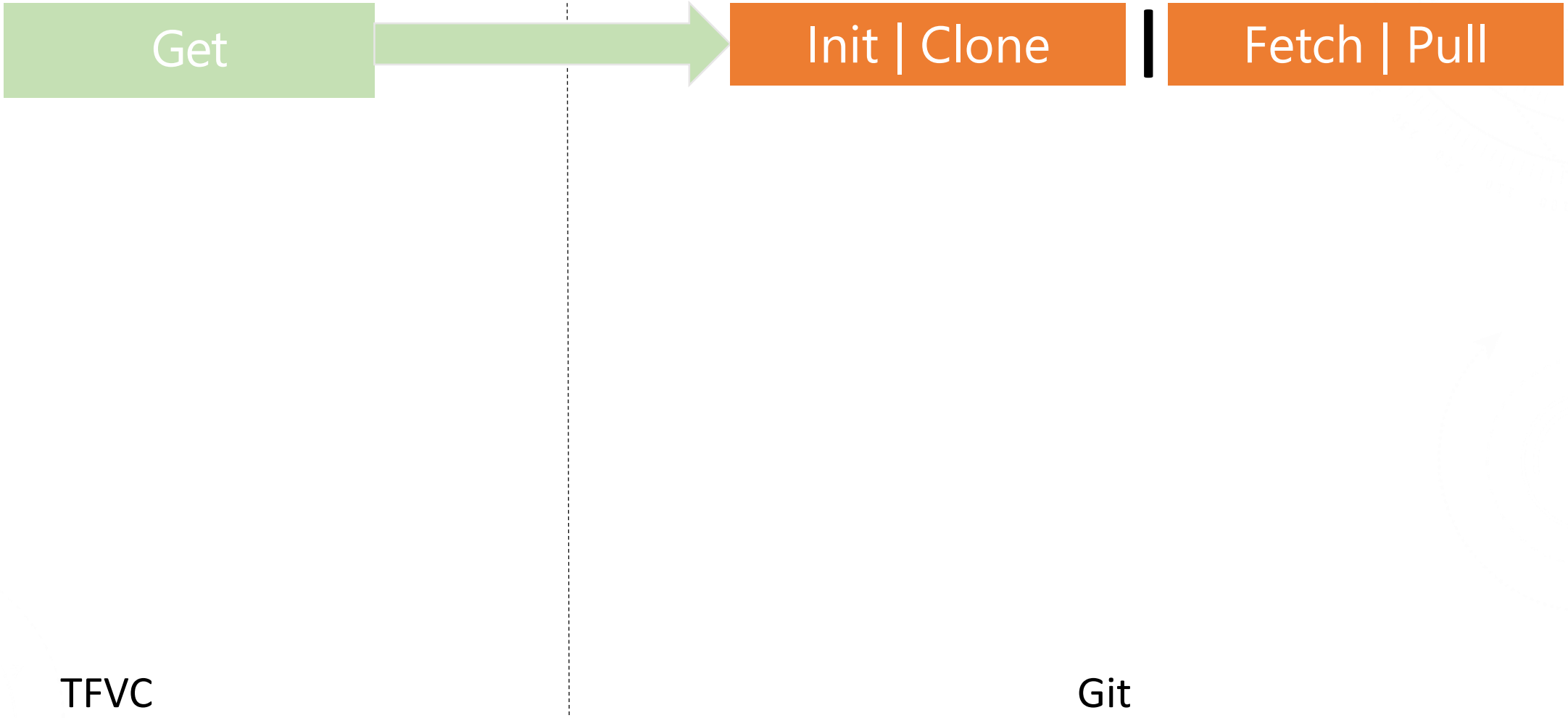
Microsoft Git Solutions

Migrating tfvc skills to git

command mapping

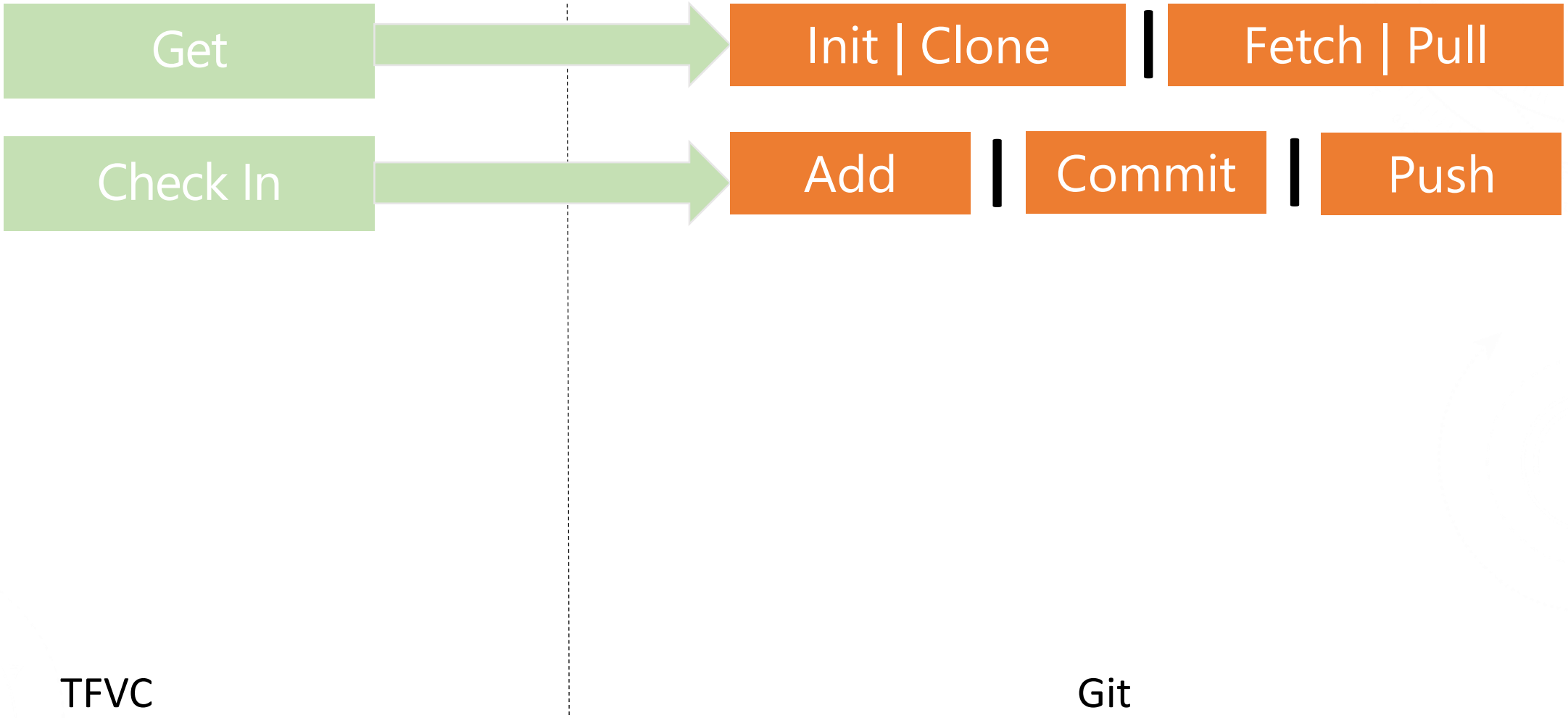
TFVC	Git
Get	
Check In	
Pending Changes	
Shelve	
Rollback	

command mapping

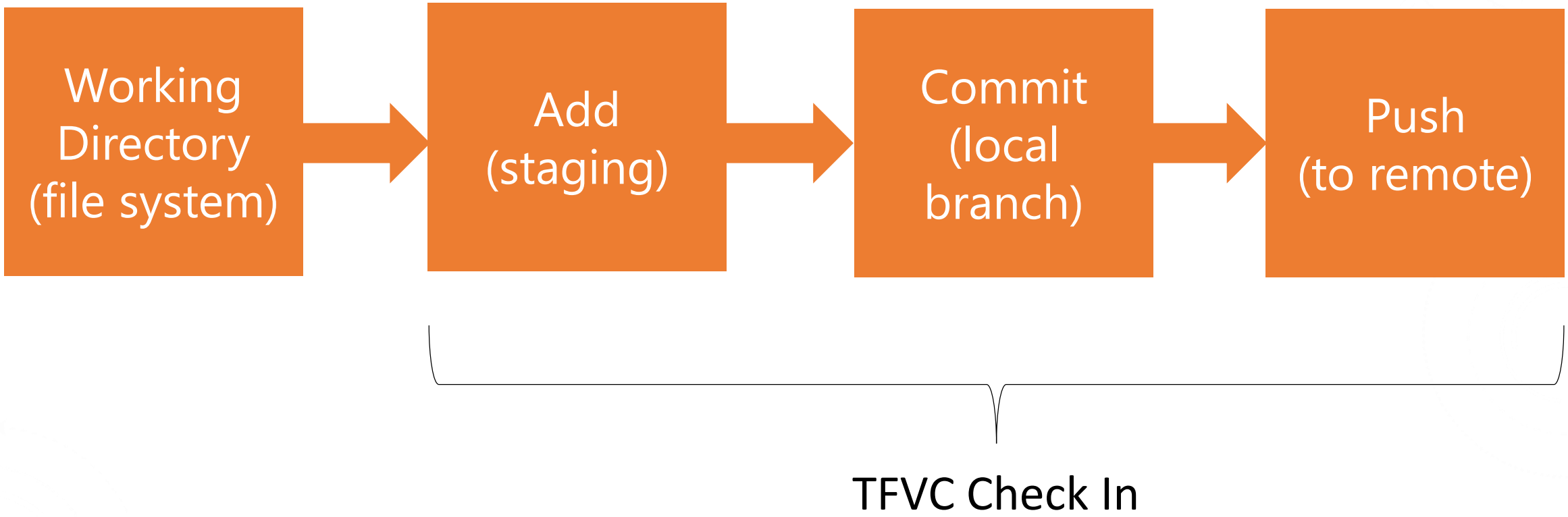


Demo: Get | Git Clone

command mapping

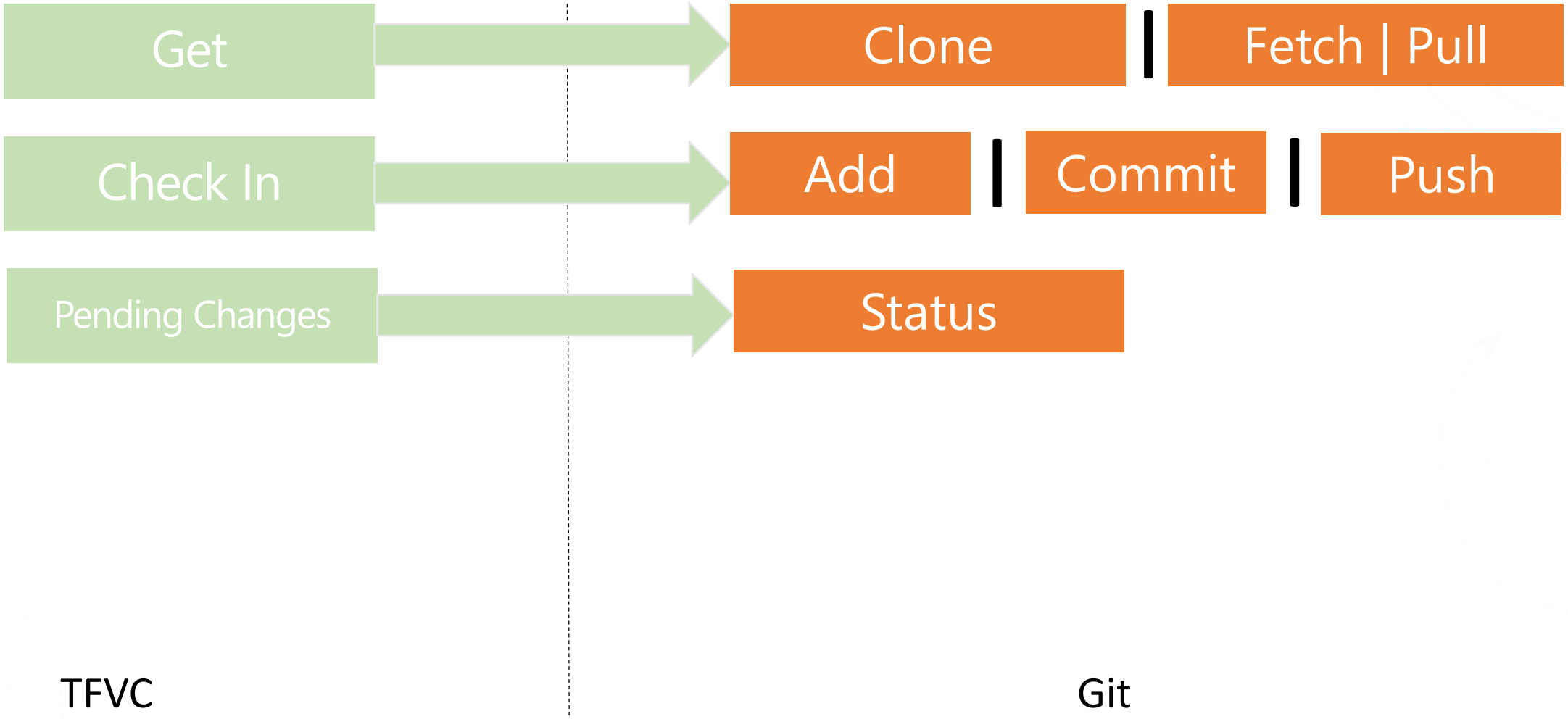


Git Staging lifecycle



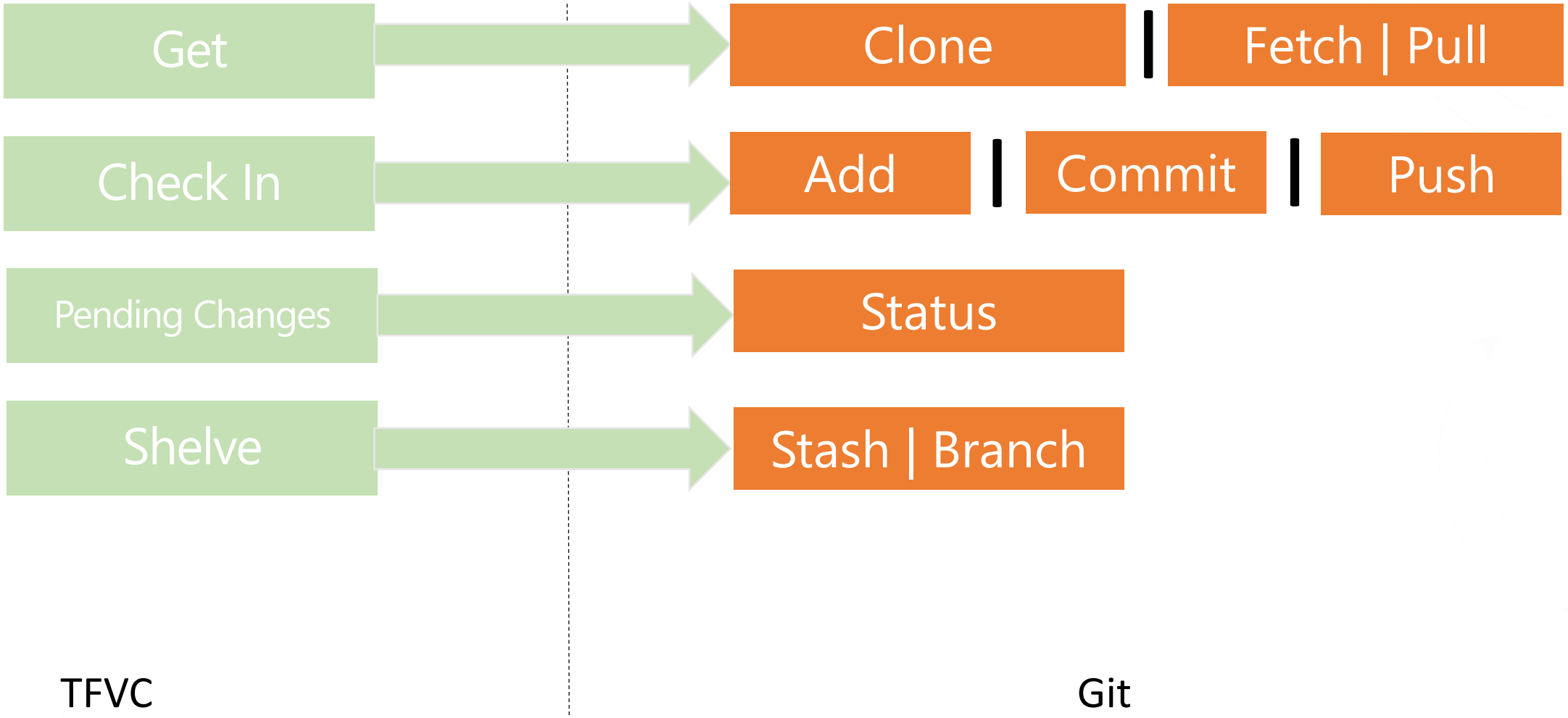
Demo: Check In | Git add; Git commit; Git push

command mapping



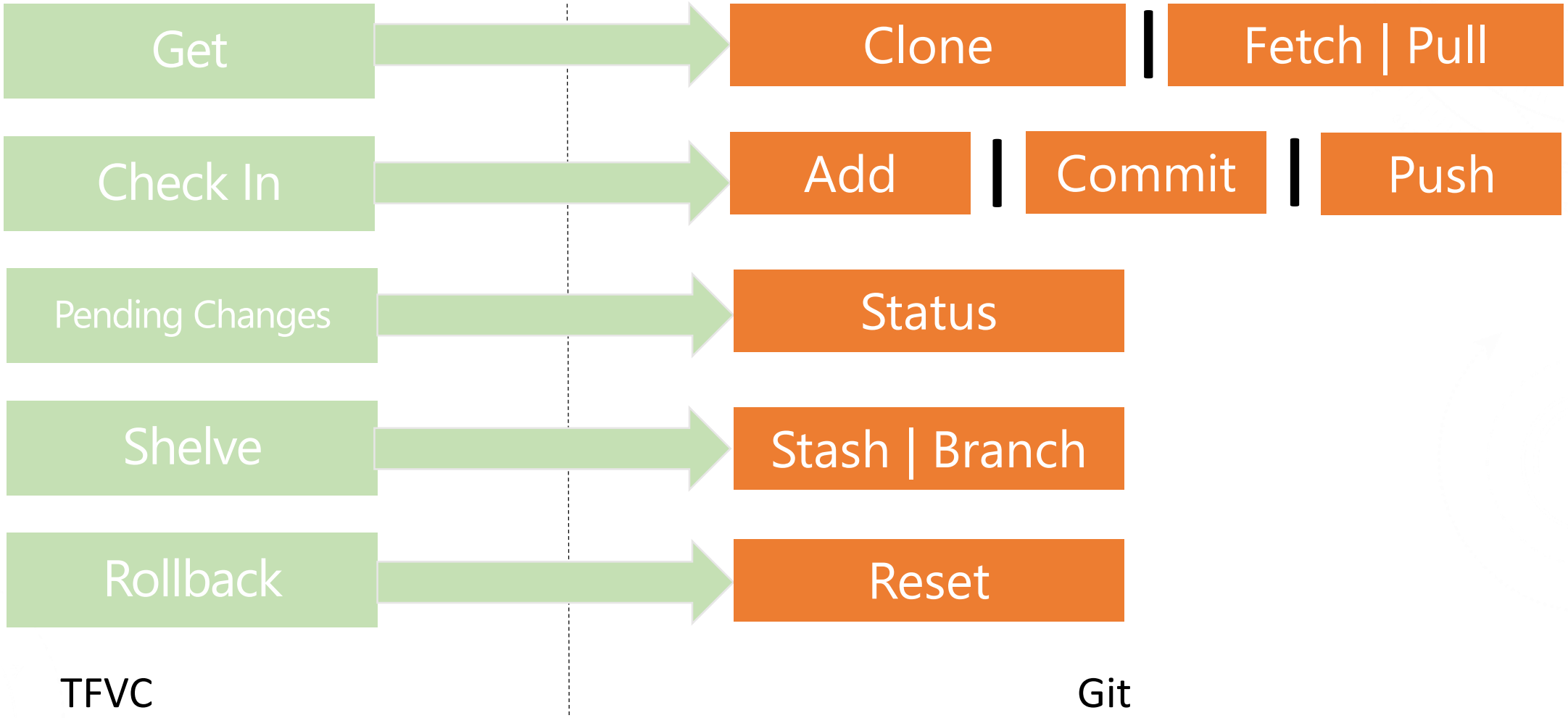
Demo: Pending Changes | Git status

command mapping



Demo: Pending Changes | Git stash

command mapping



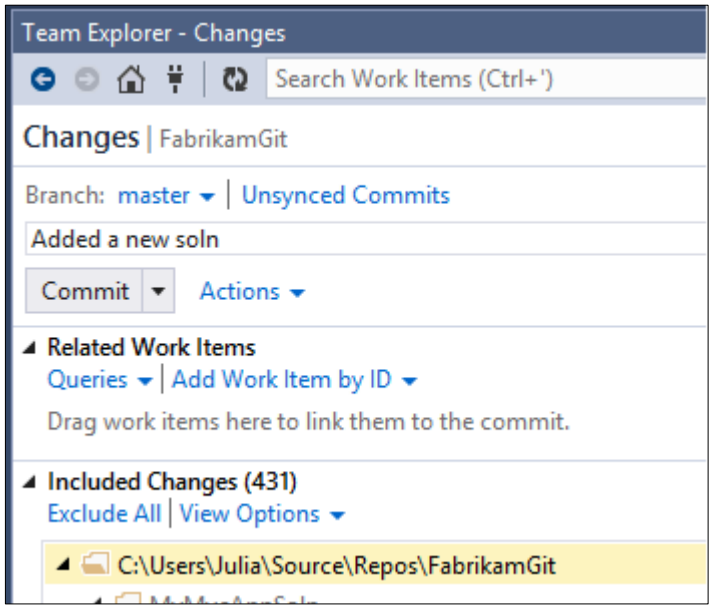
Demo: Git & visual studio

Microsoft Git Solutions

Using Git with Visual Studio

Use Visual Studio with Git

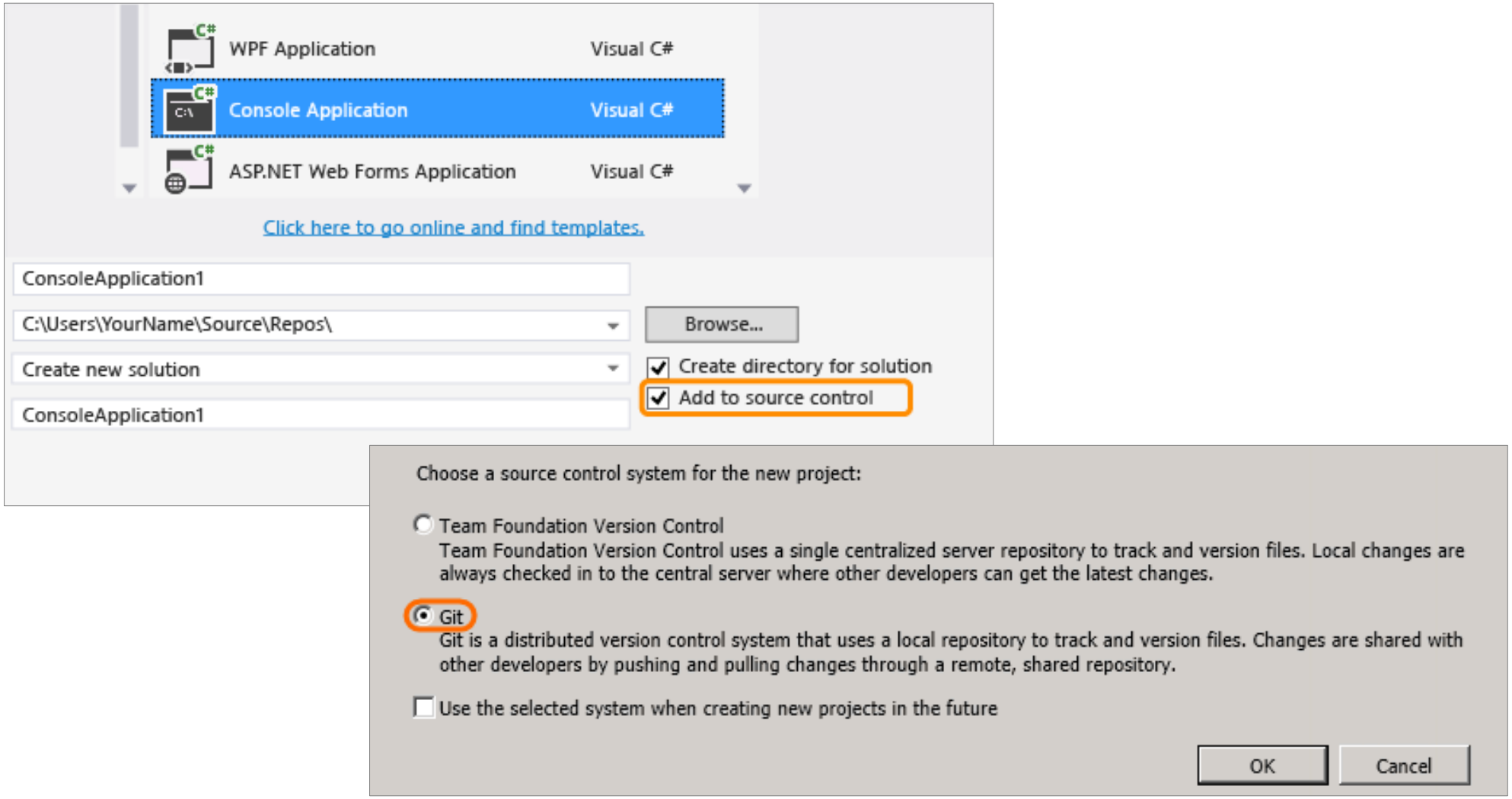
- Use Visual Studio and Git to collaborate with your team using:
  - Team Foundation Server (on-premises or in the cloud),
  - CodePlex,
  - Or a third-party service such as GitHub or Bitbucket



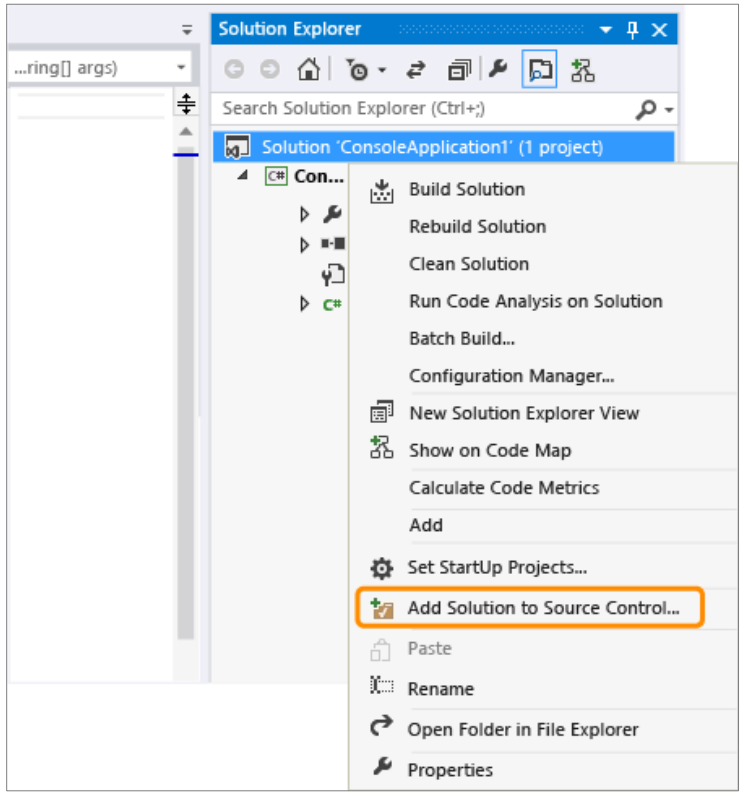
Local Repository

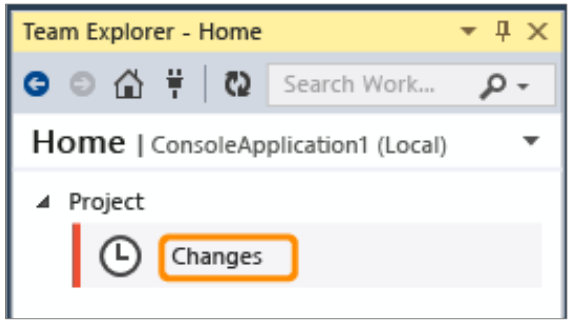
- Create a local repository on your development machine
- Code, commit, branch, and merge code
- Publish one or more branches from your local repository into a team project

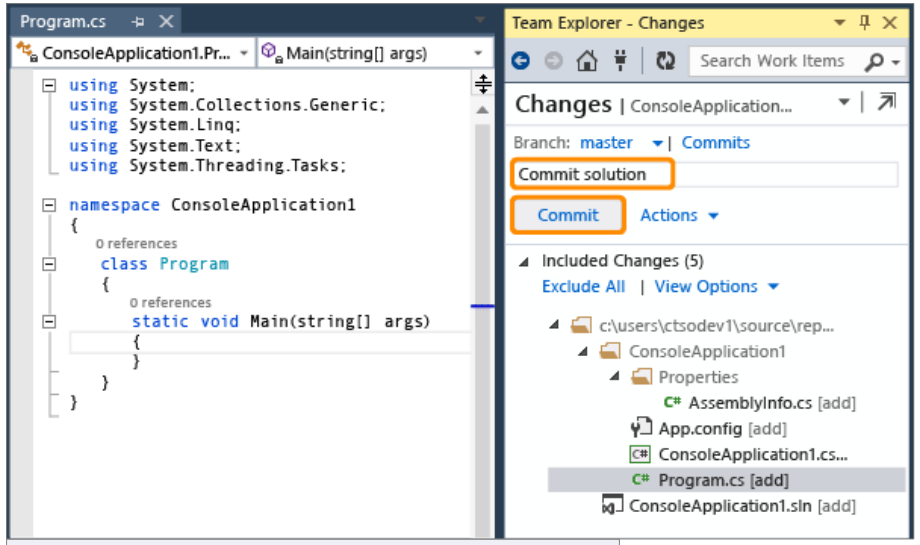
Create a New Solution under Local Git Version Control

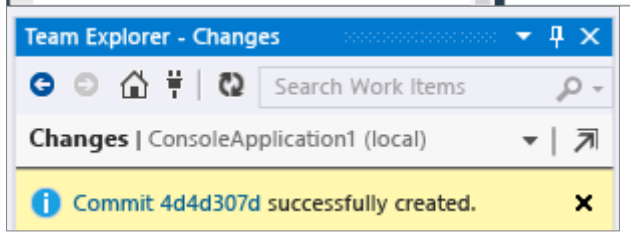


Add an Existing Solution under Local Git Version Control

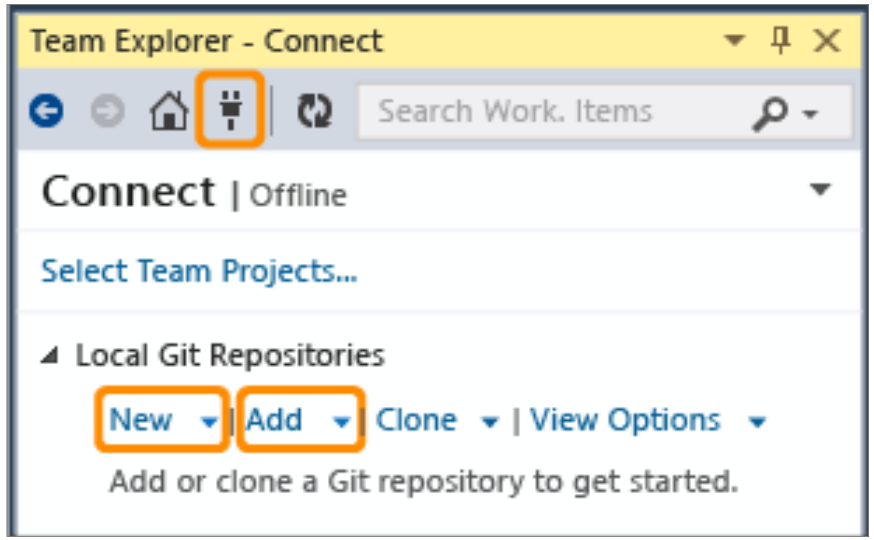


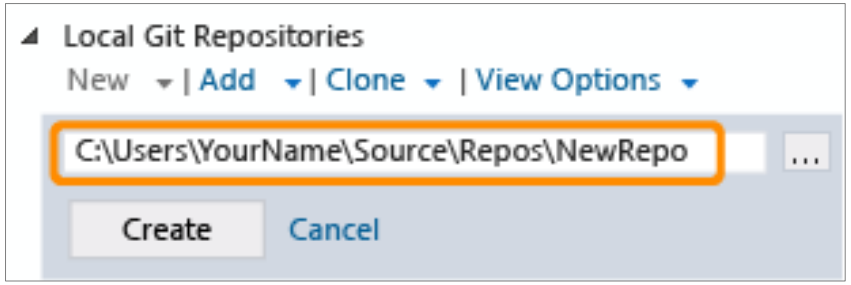




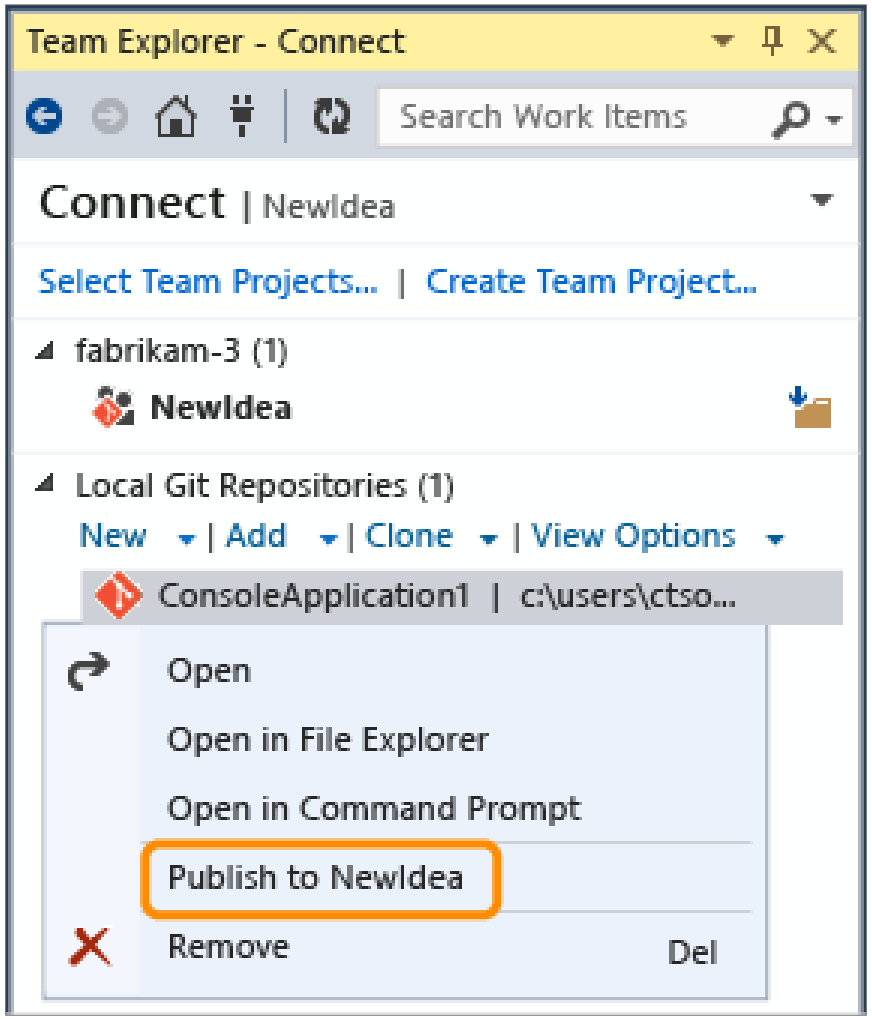


Create or Add a Local Repository





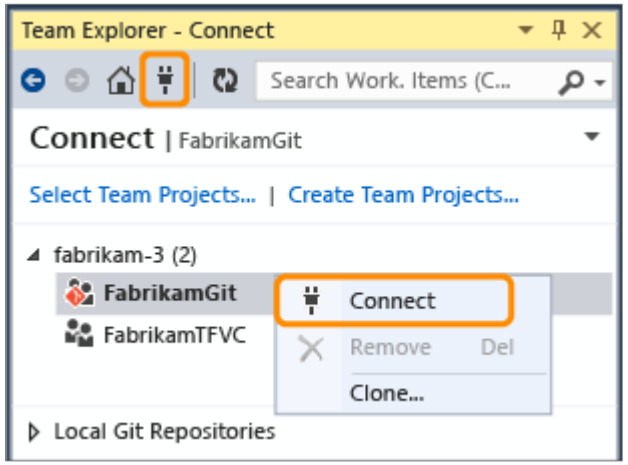
Publish Local Repository into TFS

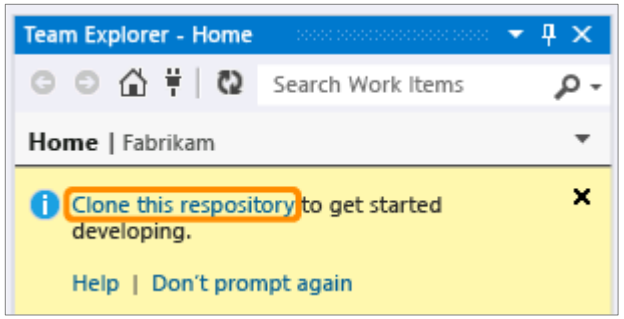


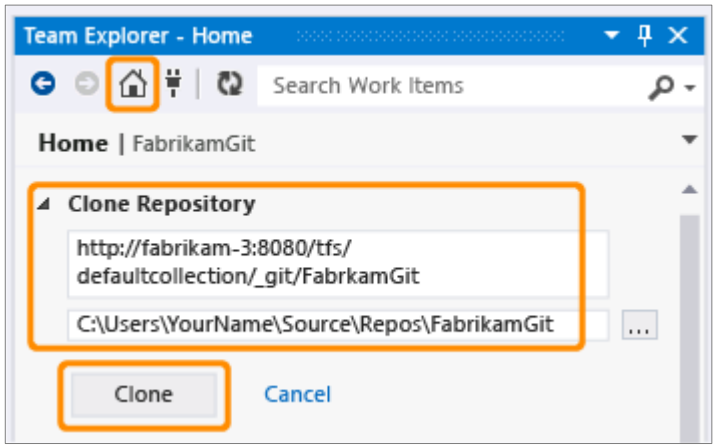
Remote Repository

- Use Visual Studio and Git to collaborate on:
  - Azure DevOps (on-premises or in the cloud)
  - GitHub
  - Or on a third-party service such as GitHub or Bitbucket

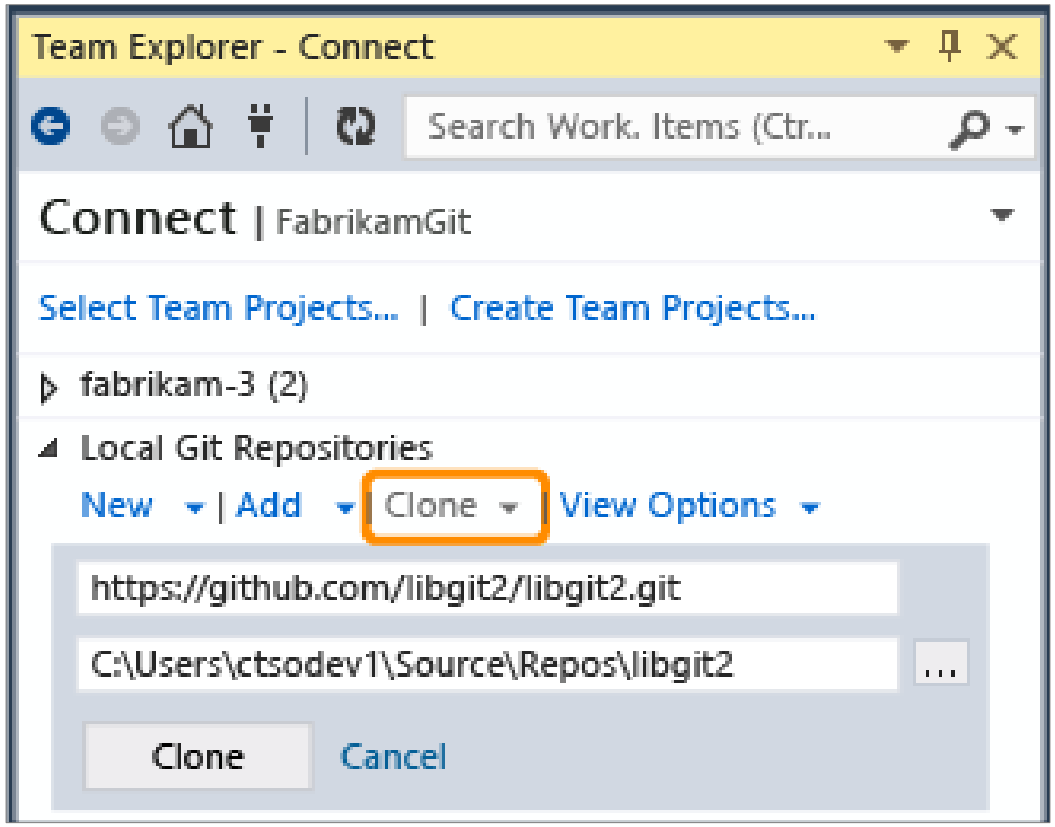
Open and Clone a Git Team Project





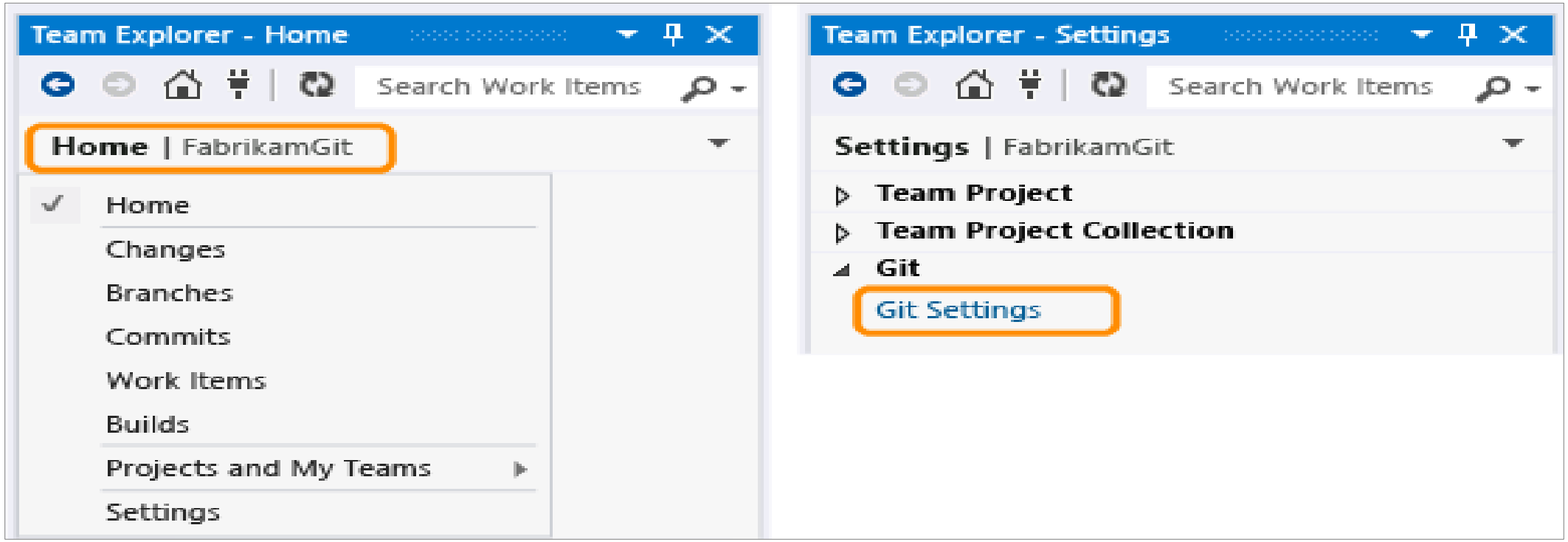


Clone a Remote Git Repository from a Third-party Service



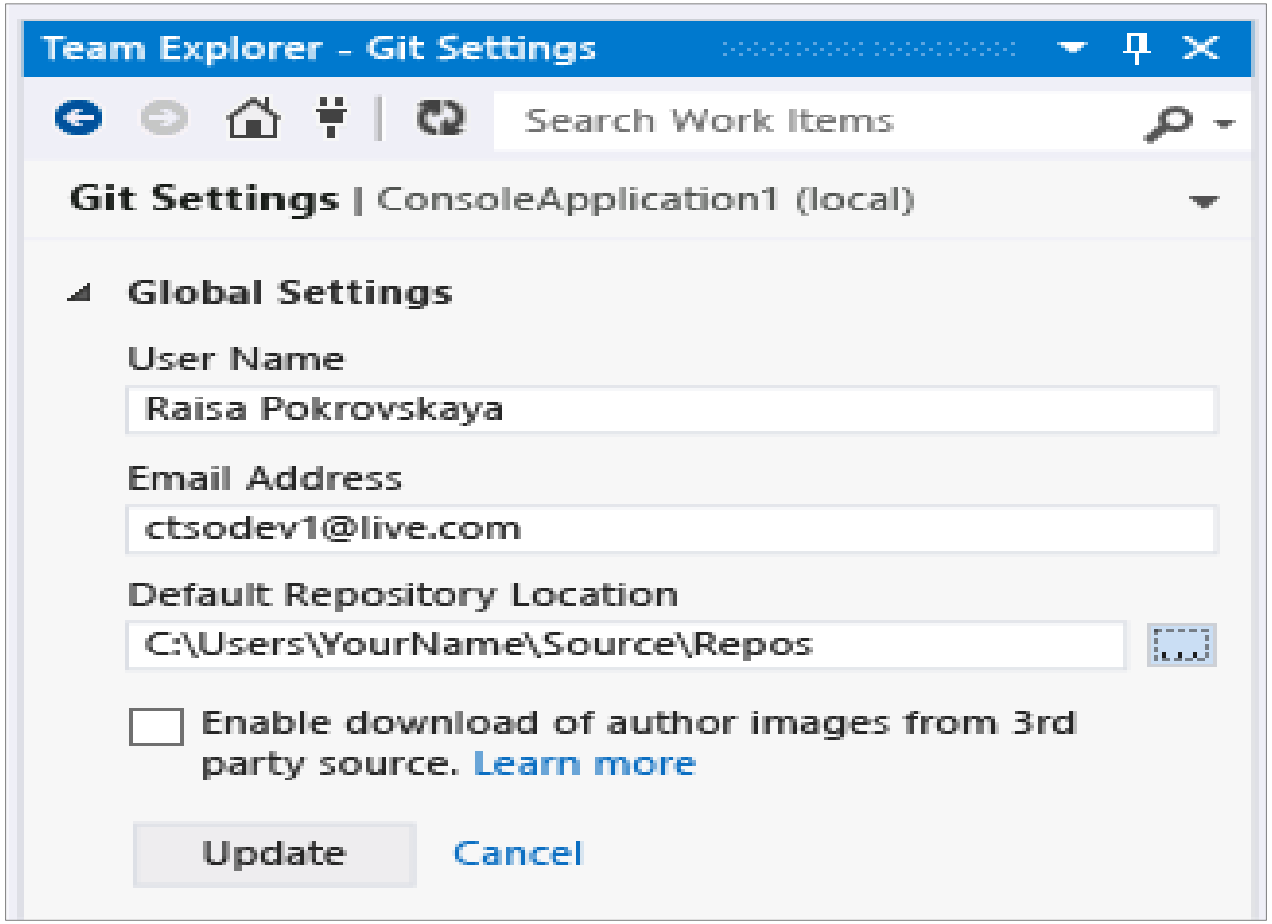
Customize Git Settings

- Open the Git Settings page



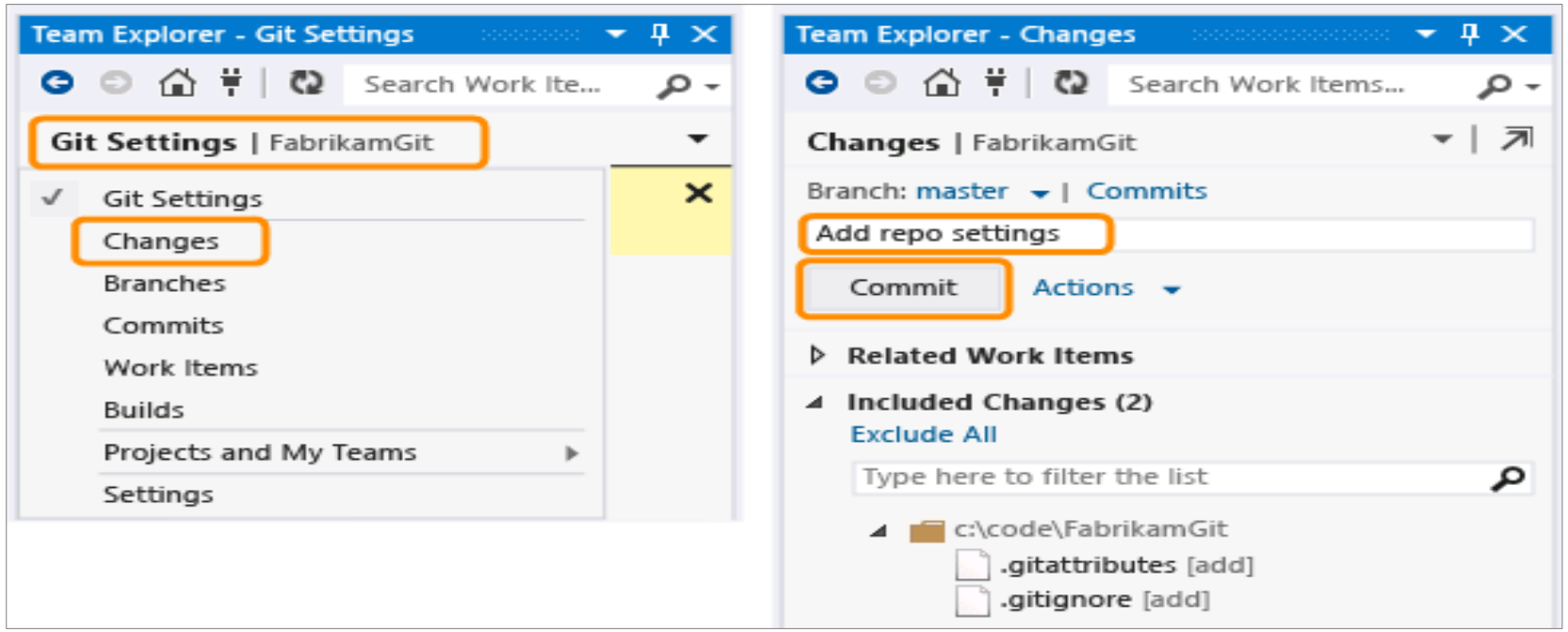
Global Settings

- Control aspects of how Git functions for the current user



Repository Settings

- Control how Git functions in each individual local repository



Demo: Getting Started

Demo: Set up Git

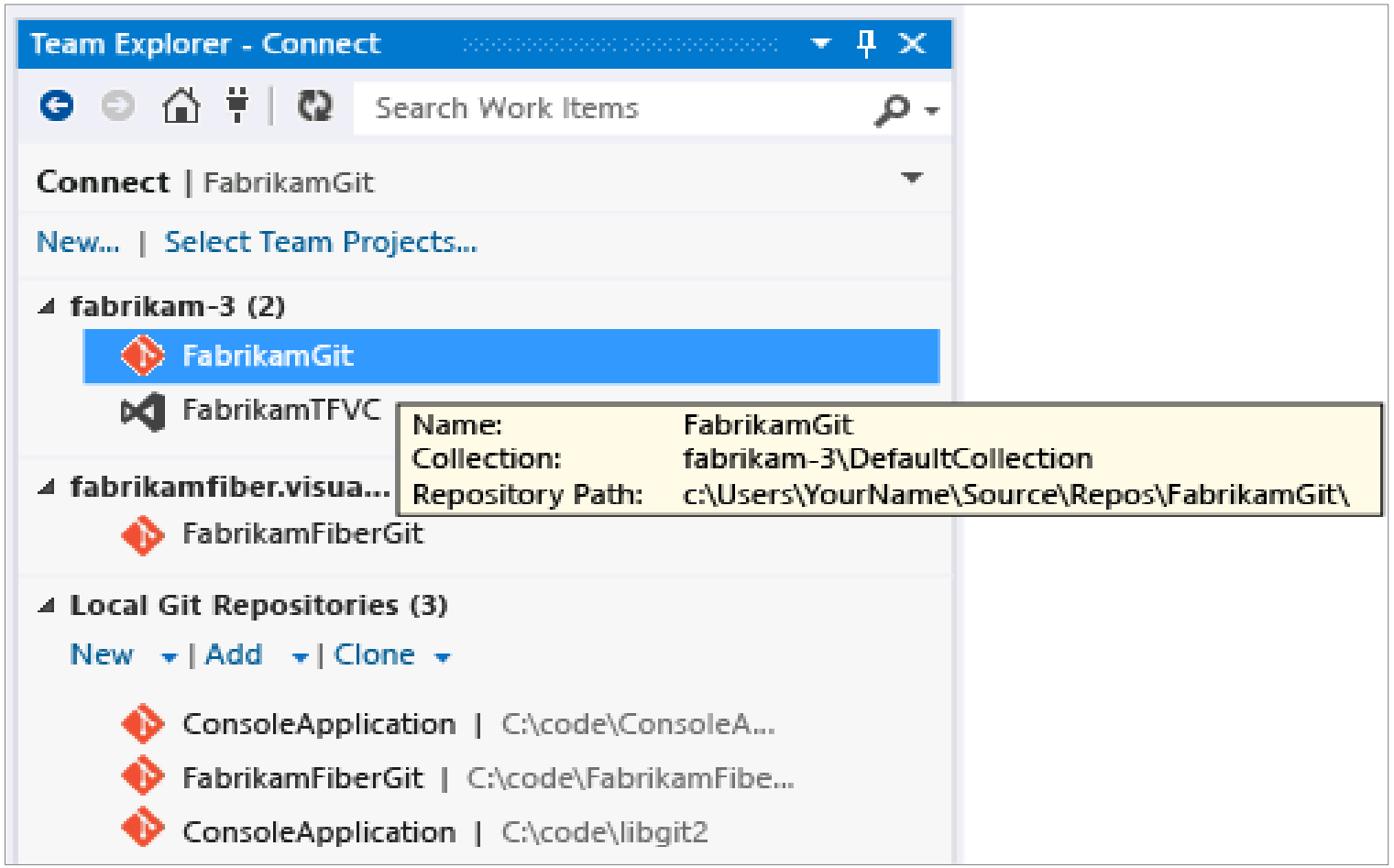
Microsoft Git Solutions

Develop using Git

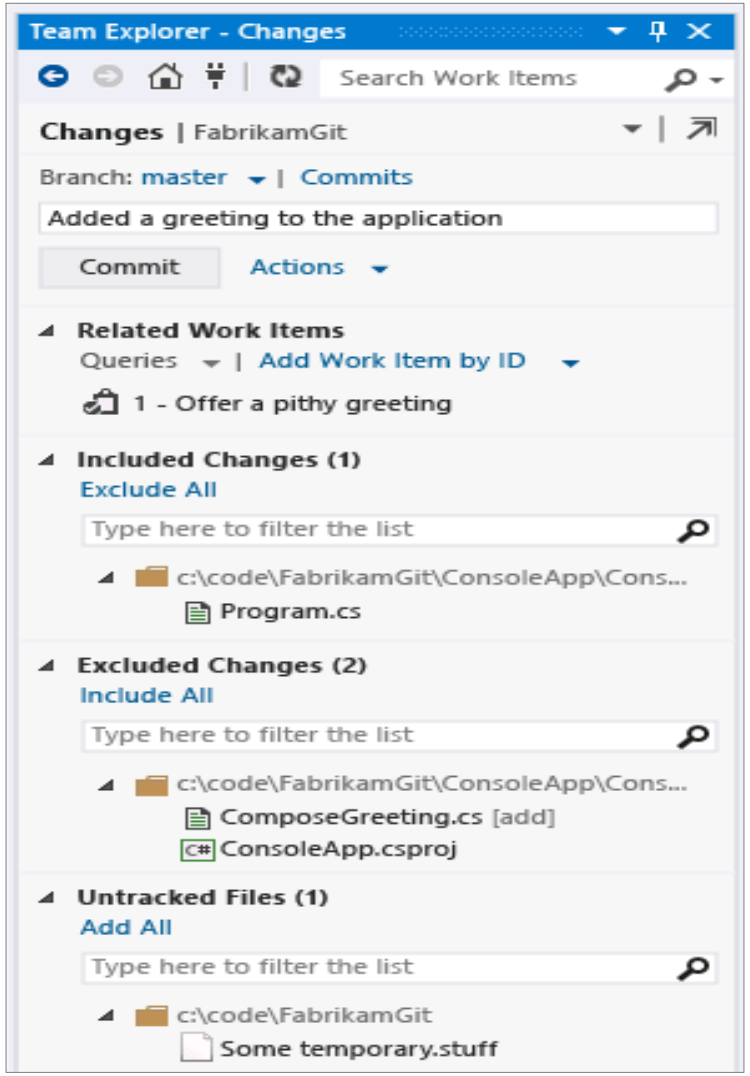
Overview

- Connect
- Manage and Commit
- Browse
- Use .gitignore
- Customize .gitignore

Connect

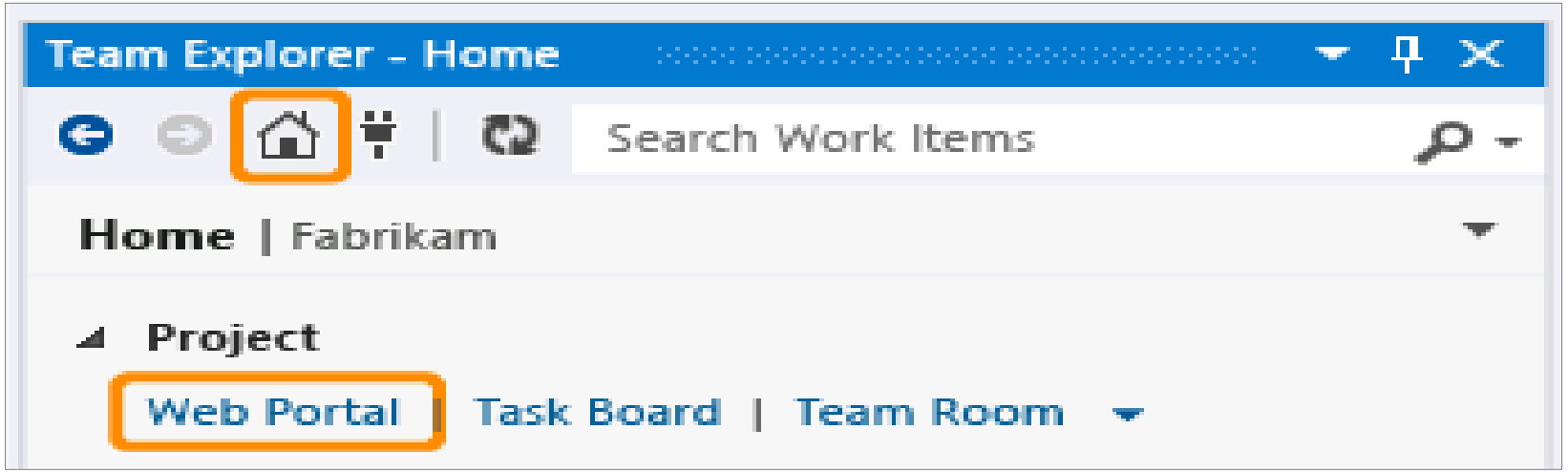

Team Explorer - Connect
Search Work Items
Connect | FabrikamGit
New... | Select Team Projects...
fabrikam-3 (2)
FabrikamGit
FabrikamTFVC
Name: FabrikamGit
Collection: fabrikam-3\DefaultCollection
Repository Path: c:\Users\YourName\Source\Repos\FabrikamGit\
fabrikamfiber.visua...
FabrikamFiberGit
Local Git Repositories (3)
New | Add | Clone
ConsoleApplication | C:\code\ConsoleA...
FabrikamFiberGit | C:\code\FabrikamFibe...
ConsoleApplication | C:\code\libgit2

Manage and Commit



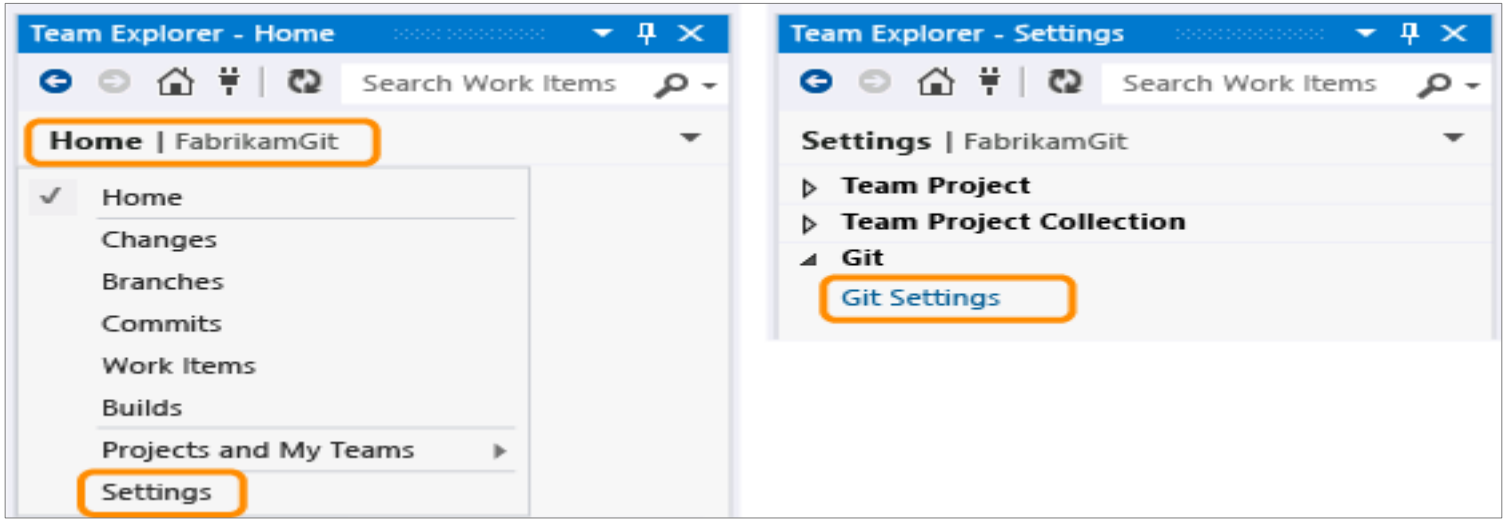
Browse

- Browse your local Git repository from File Explorer or Git Command Prompt
- Browse remote repositories from web browser (Keyboard: **Ctrl+0**, **A**)



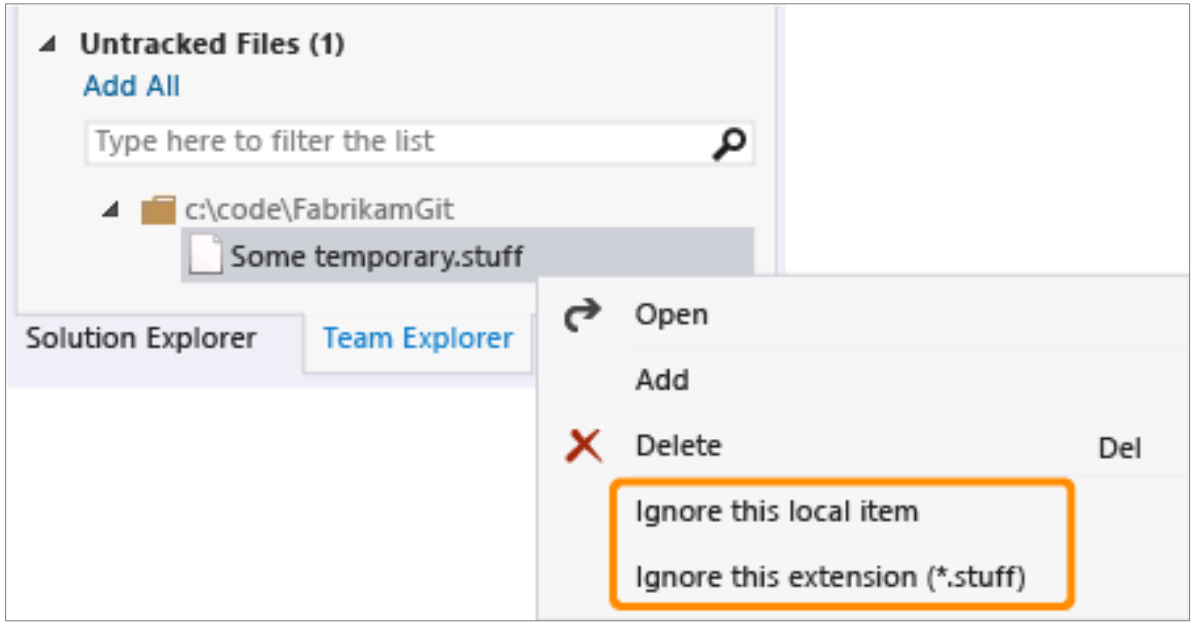
Use .gitignore

- Avoid file clutter in your work and in your repository
  - Example: Locally compiled binary files
- Ignored files do not appear in the Changes page in Team Explorer
- Ignored files are neither committed nor pushed



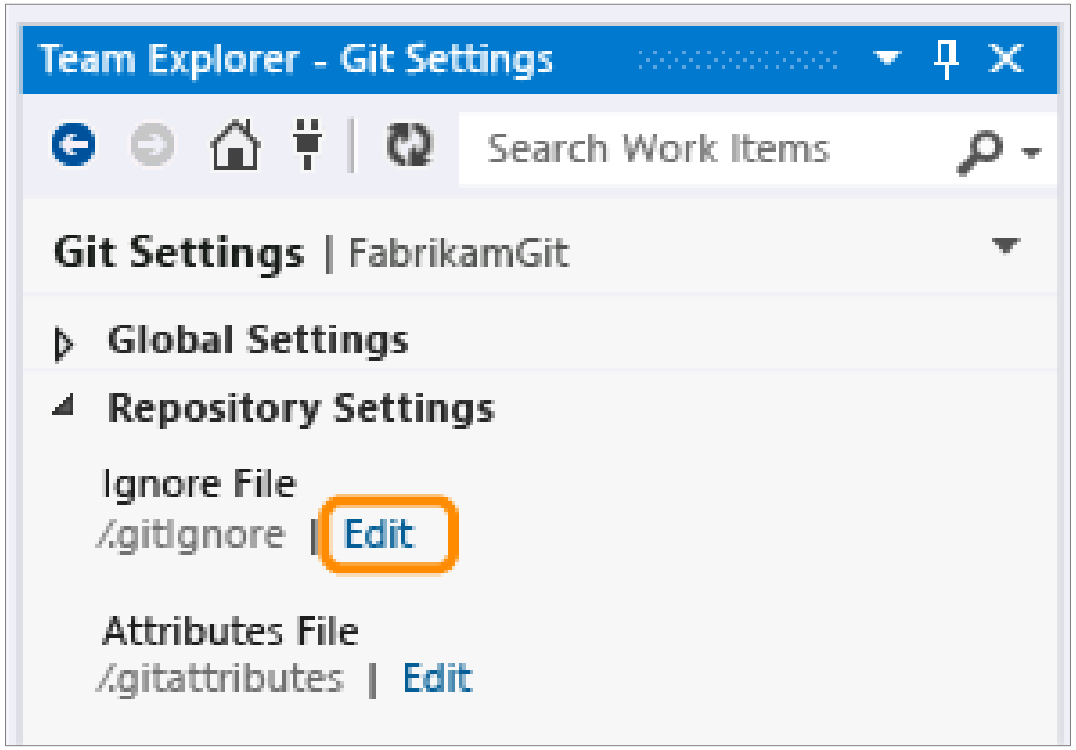
Customize .gitignore

- Ignore a file, a type of file, or even a folder from Changes



Customize .gitignore

- Edit .gitignore file from Git Settings



Demo: Develop using Git

Microsoft Git Solutions

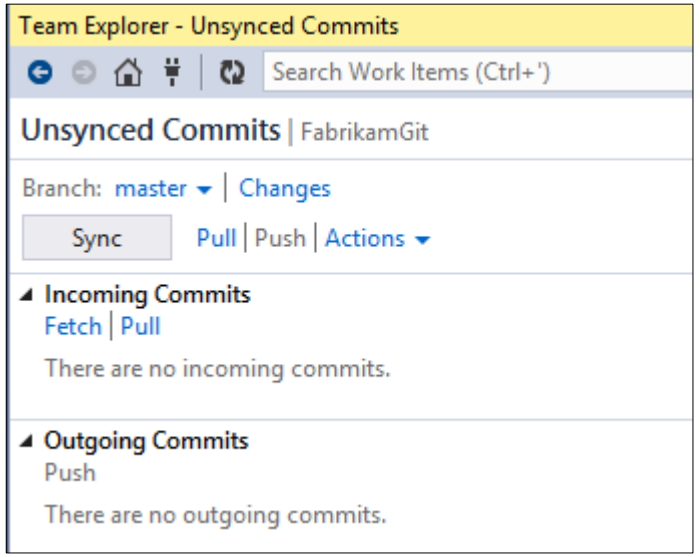
Collaborate using Git

Overview

- Fetch
- Pull
- Push
- Alerts
- Multiple Repositories

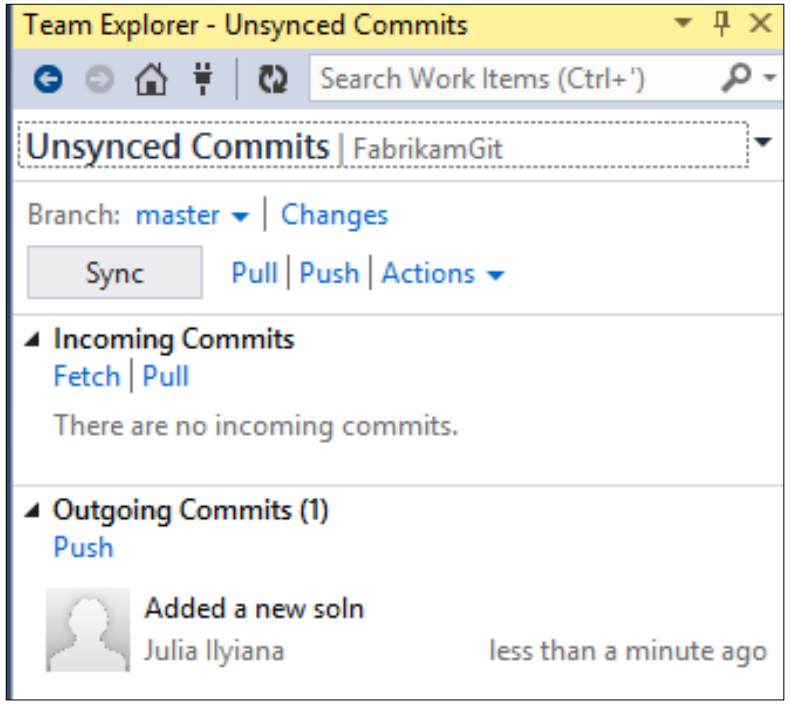
Fetch

- Fetch from Commits page
- Fetch commits from your team project before you pull
- Fetch before you can get a copy of a published branch
- After you fetch a commit, you can open its context menu and choose View Commit Details



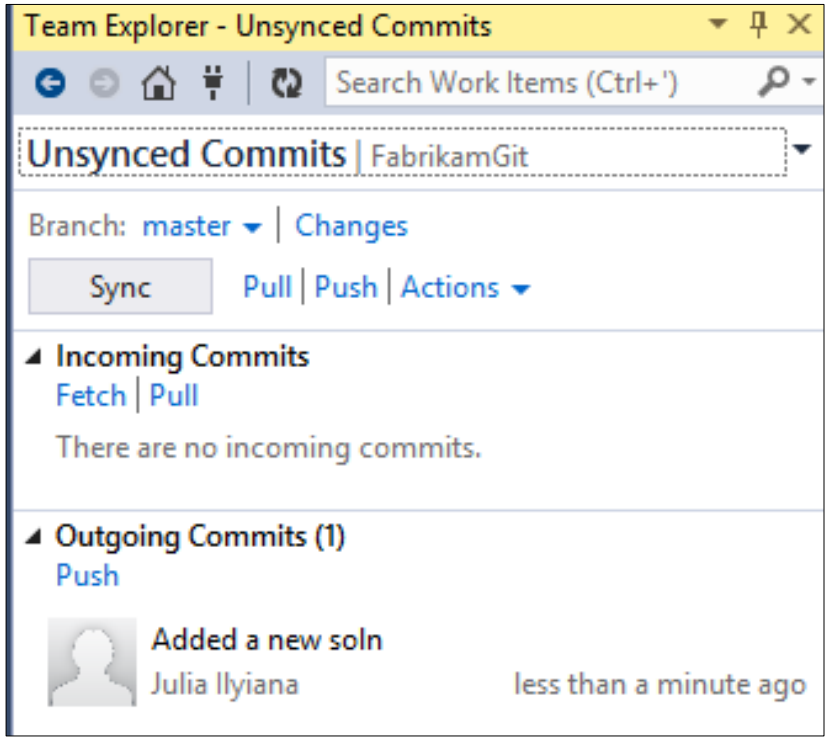
Pull

- Integrate changes from your team into your local repository
- Pull any commits pushed by your team



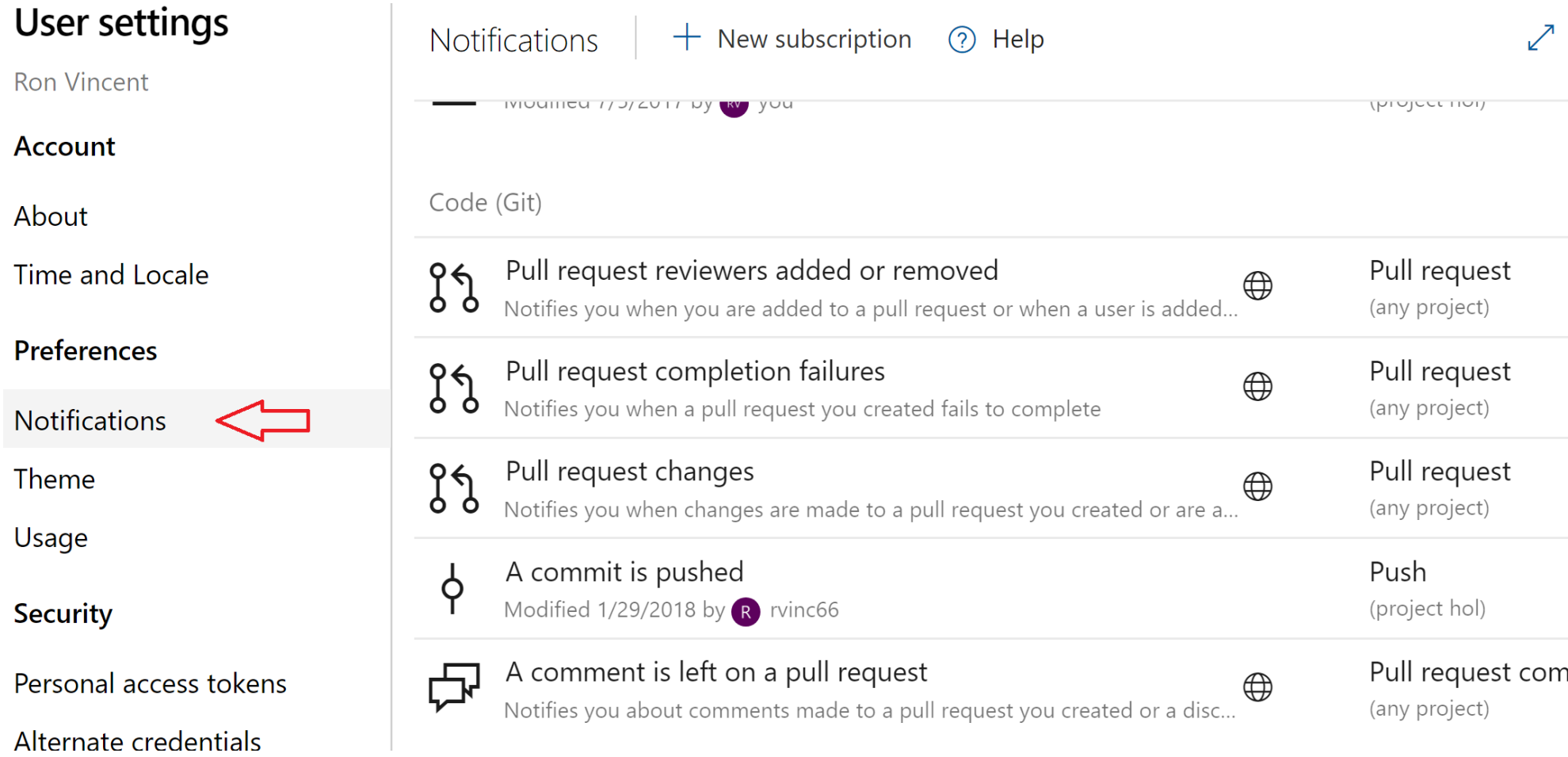
Push

- Push changes to commit them to the team's remote Git repository



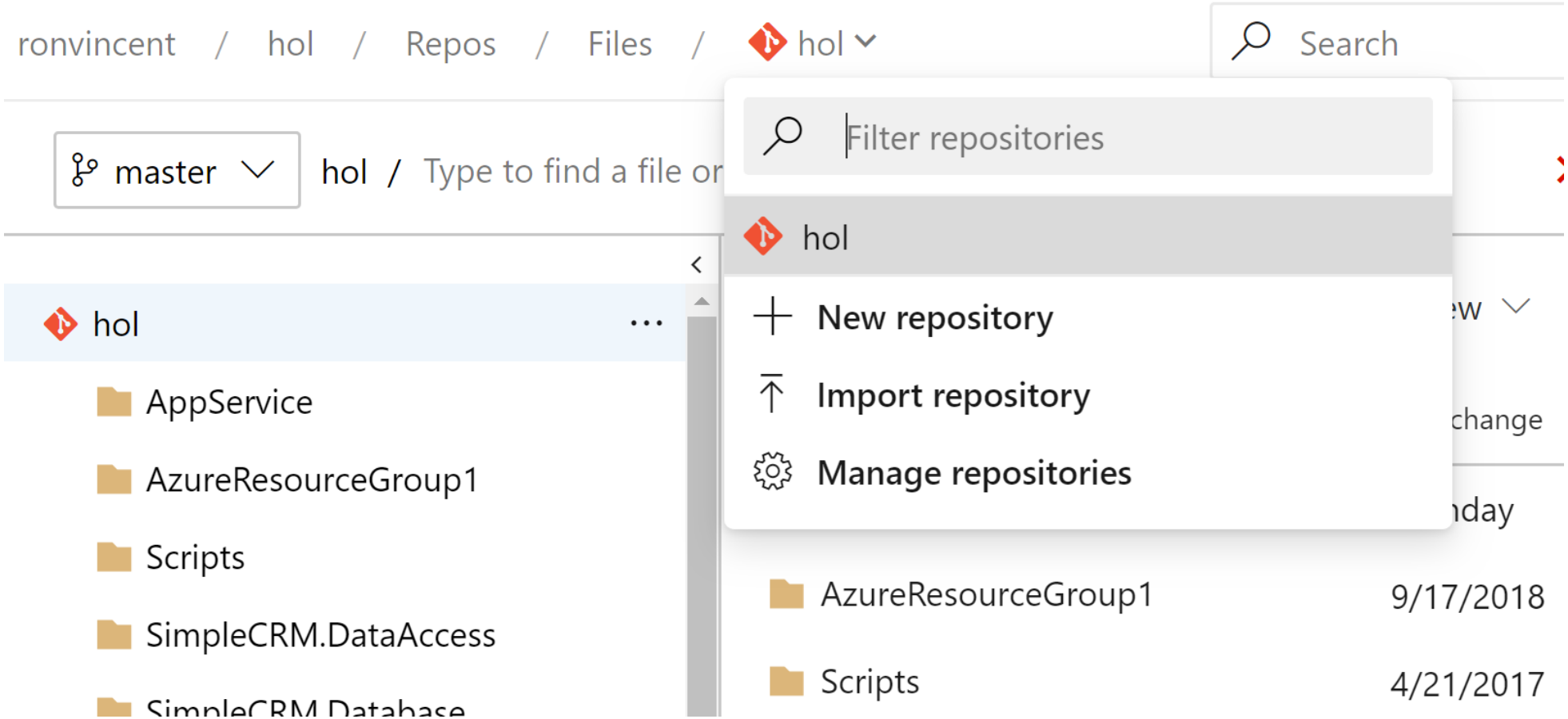
Alerts

- Subscribe to receive email alerts



Multiple Repositories

- Add additional repositories to your Git team project from web portal



Demo: Collaborate using Git

Microsoft Git Solutions

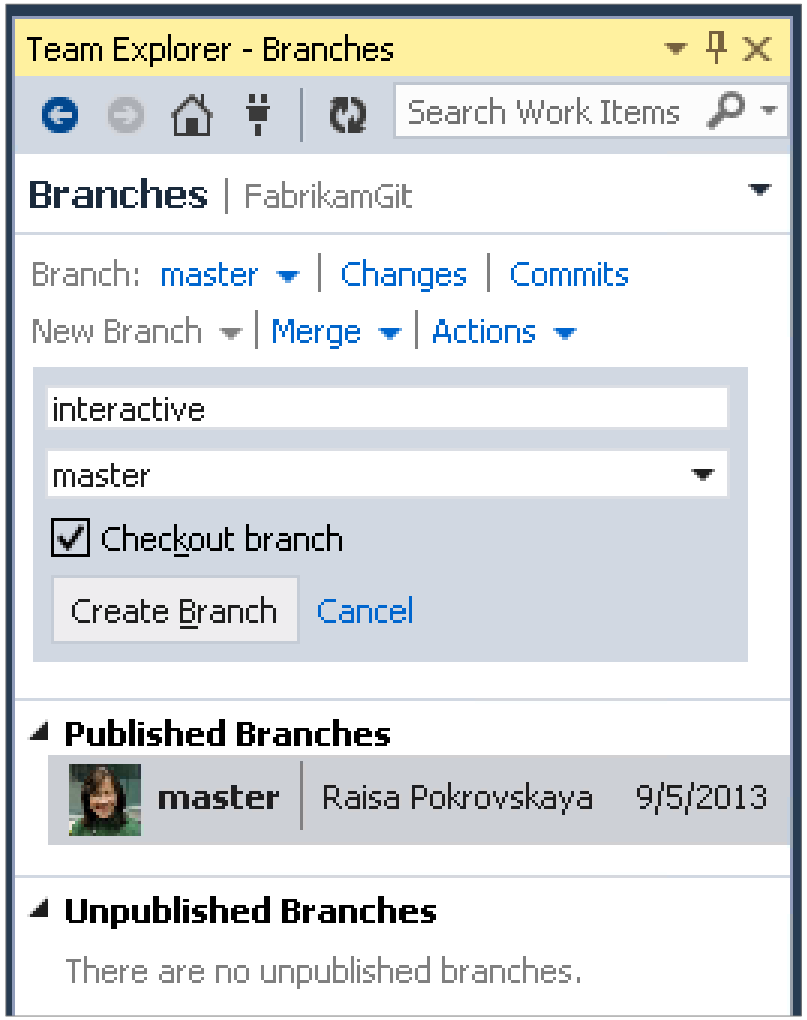
Use Branches

Overview

- Create a Branch
- Merge a Branch
- Publish a Branch

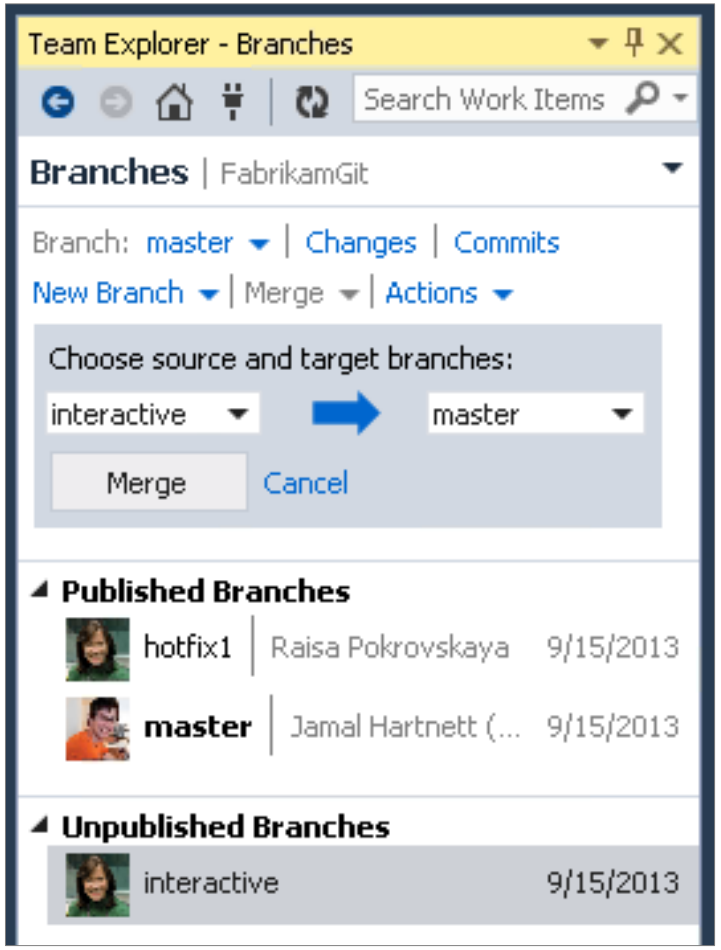
Create a Branch

- Create a branch from the branches page



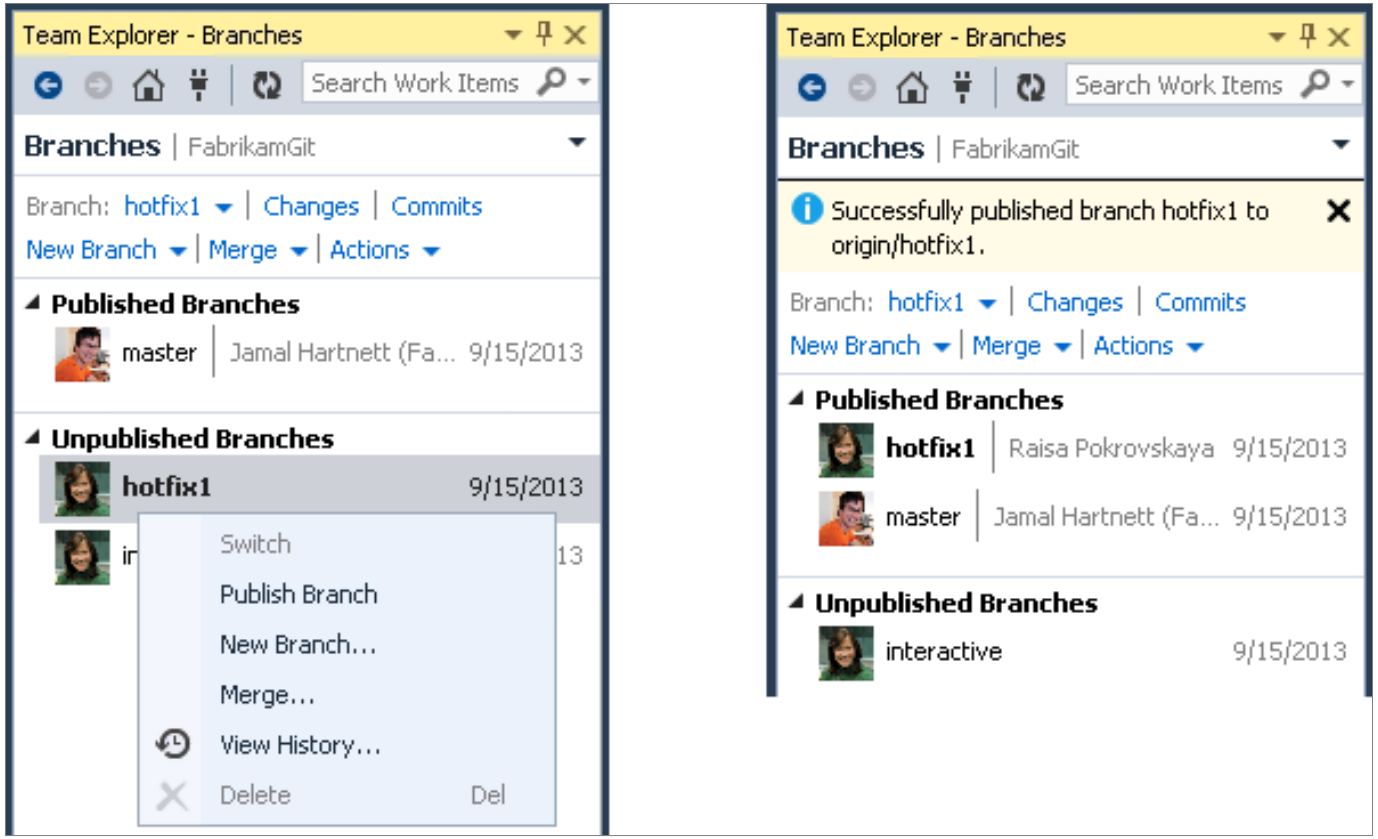
Merge a Branch

- Merge the work you have done in one branch into another branch



Publish a Branch

- Collaborate or preserve the work you have done on a branch by publishing it



Demo: Use Branches

Microsoft Git Solutions

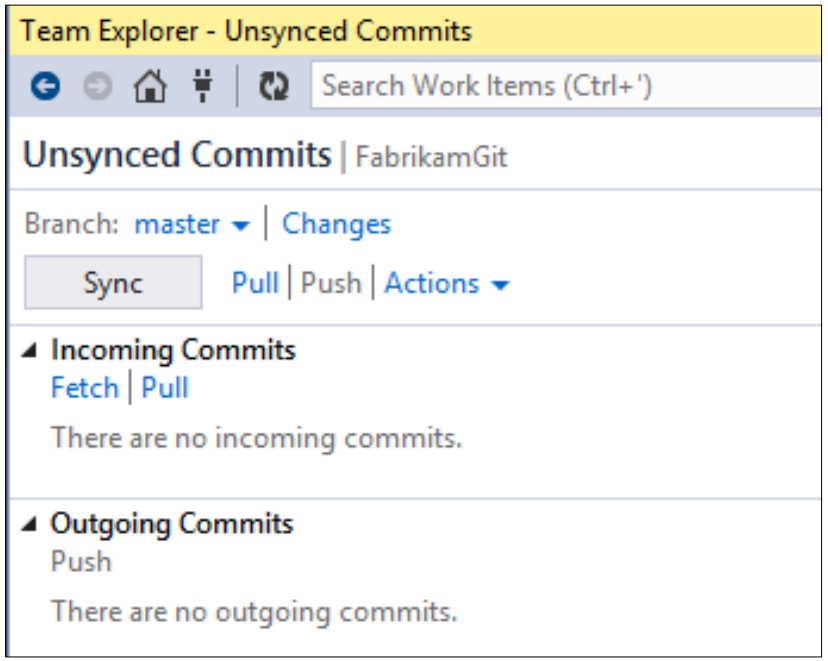
Resolve Conflicts

Overview

- Examples of Conflicts (Pull)
- Resolve Content Conflicts
- Commit the Merge

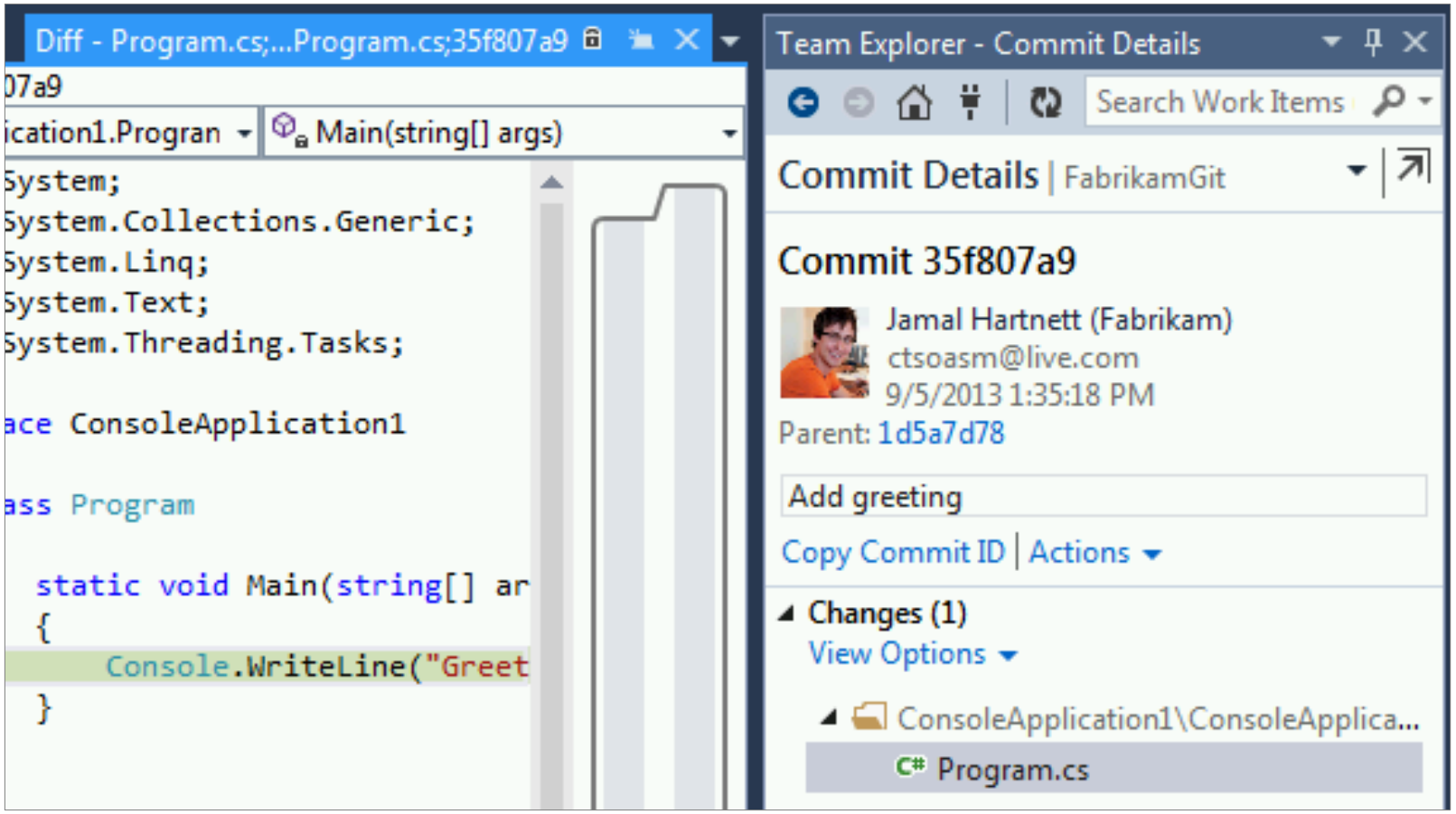
Examples of Conflicts—Pull

- Raisa wants to push a change to a file
- If any commits have been pushed since her last pull, she must pull them down before she can push her commit



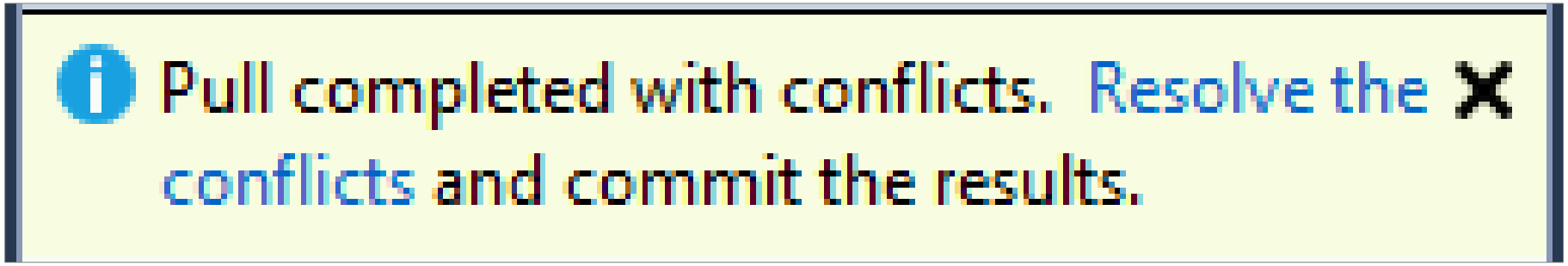
Examples of Conflicts—Pull (continued)

- She can view details about the incoming commit from Jamal and see that he has modified the same line of code that she modified



Examples of Conflicts—Pull (continued)

- When she tries to pull, Visual Studio shows her the conflict



Resolve Content Conflicts

Merge - vctmp2272_...Program.eb1d3e41.cs*

Accept Merge

1 Conflicts (0 Remaining)

Source: ConsoleApplication1\ConsoleApplicat...

ConsoleApplicatio | Main(string[] args

```
lain(string[] args)

riteLine("Greetings, planet.");
```

100 %

Target: ConsoleApplication1\ConsoleApplicati...

ConsoleApplicatio | Main(string[] args

```
ain(string[] args)

riteLine("Hello world!");
```

100 %

Result: ConsoleApplication1\ConsoleApplication1\Program.cs

ConsoleApplication1.Program | Main(string[] args)

```
:ic void Main(string[] args)

Console.WriteLine("Hello world!"); //greet user
```

Team Explorer - Resolve Conflicts

Search Work

Resolve Conflicts | FabrikamGit

Commit Merge | Undo Merge

Conflicts (1)

Name	Path
Program.cs [both modified]	Console...

Merge

Conflicting content changes have been made.
Compare Files

Edited on **Remote** | Diff | Take Remote
Edited on **Local** | Diff | Keep Local

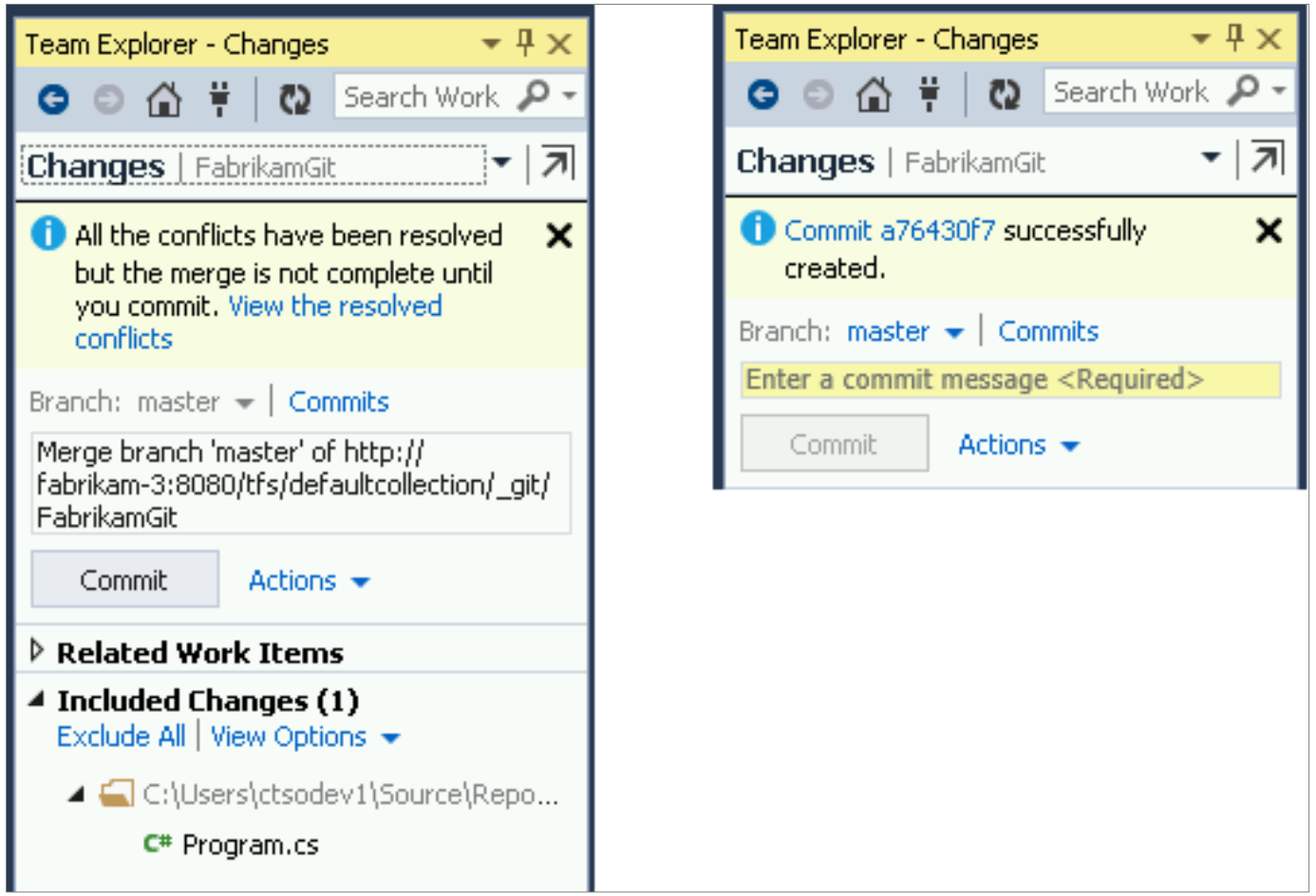
Resolved
There are no resolved conflicts

Commit the Merge

- You can commit the merge if you are ready
- Or, if you cannot resolve all the conflicts, you can undo all your resolutions

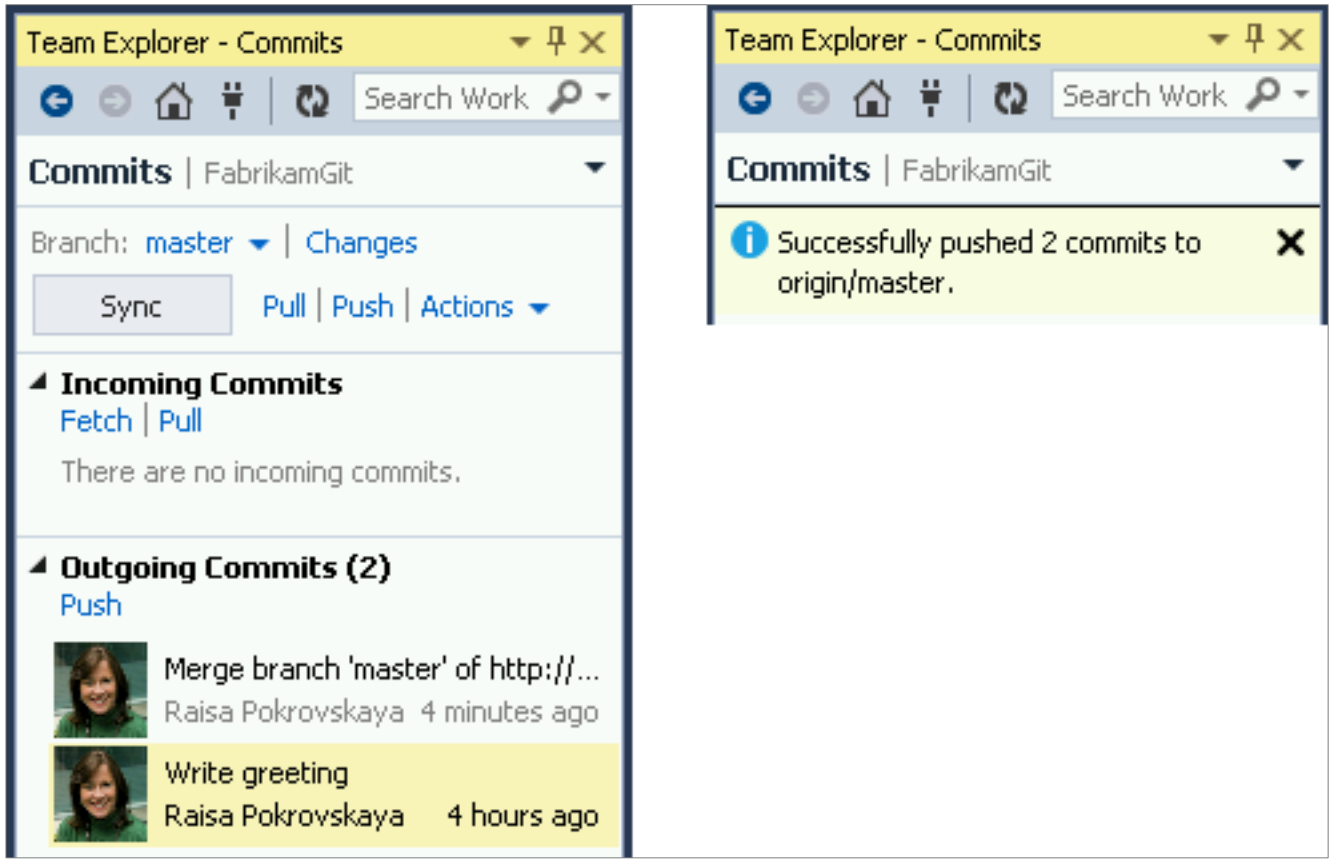
Commit the Merge (continued)

- Commit the merge



Commit the Merge (continued)

- Push changes into the remote repository



Demo: Resolving Conflicts

Microsoft Git Solutions

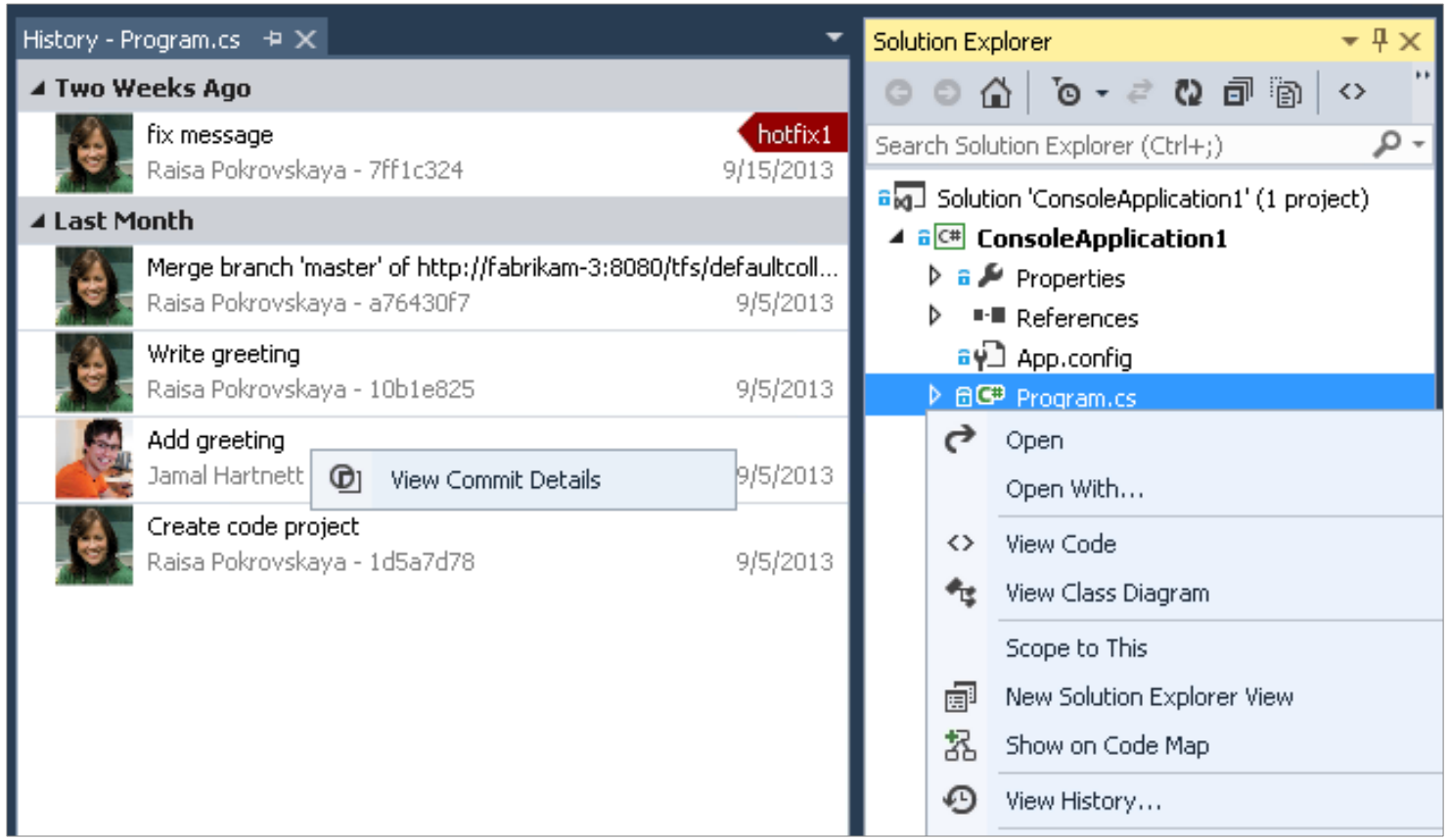
View History and Tags

Overview

- View Historical Data in Visual Studio
- View Historical Data in Web Browser
- Tags

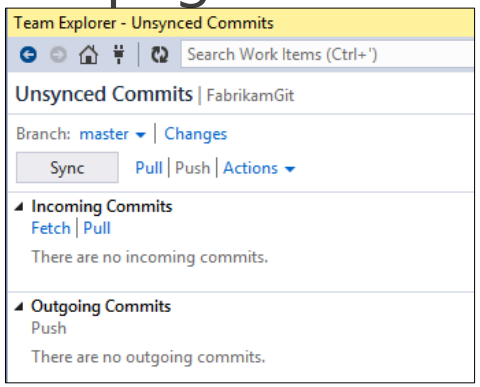
View Historical Data in Visual Studio

- File History

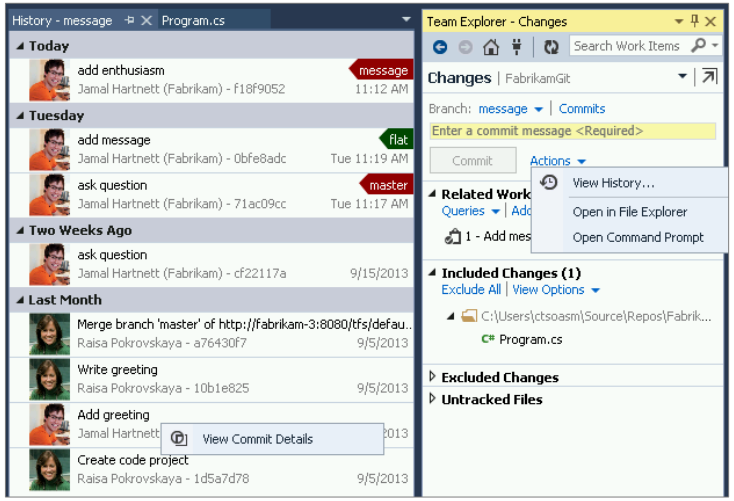


View Historical Data in Visual Studio

- Branch History
  - Go to the Unsynced Commits page and then fetch the latest changes

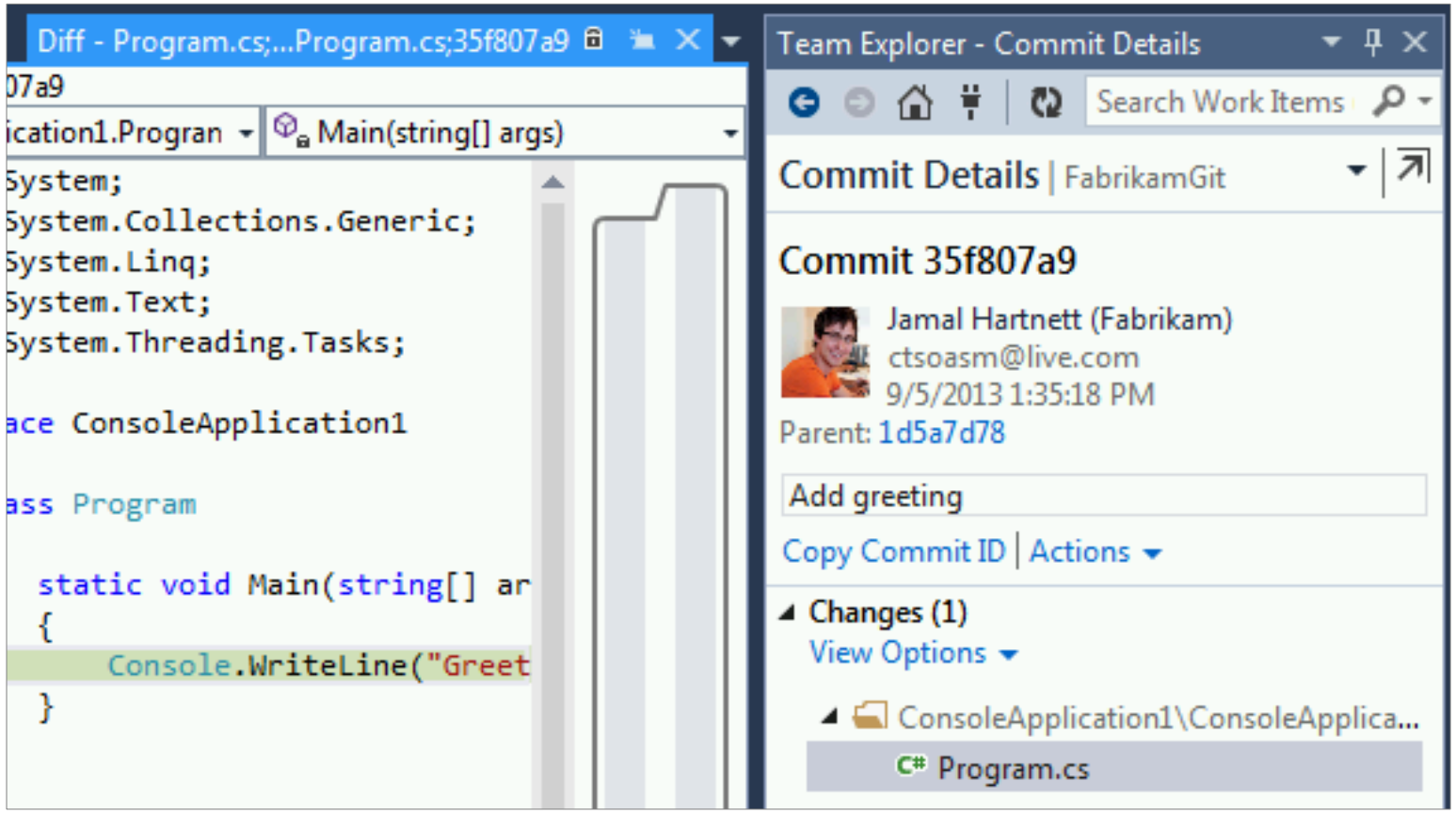


  - View the history



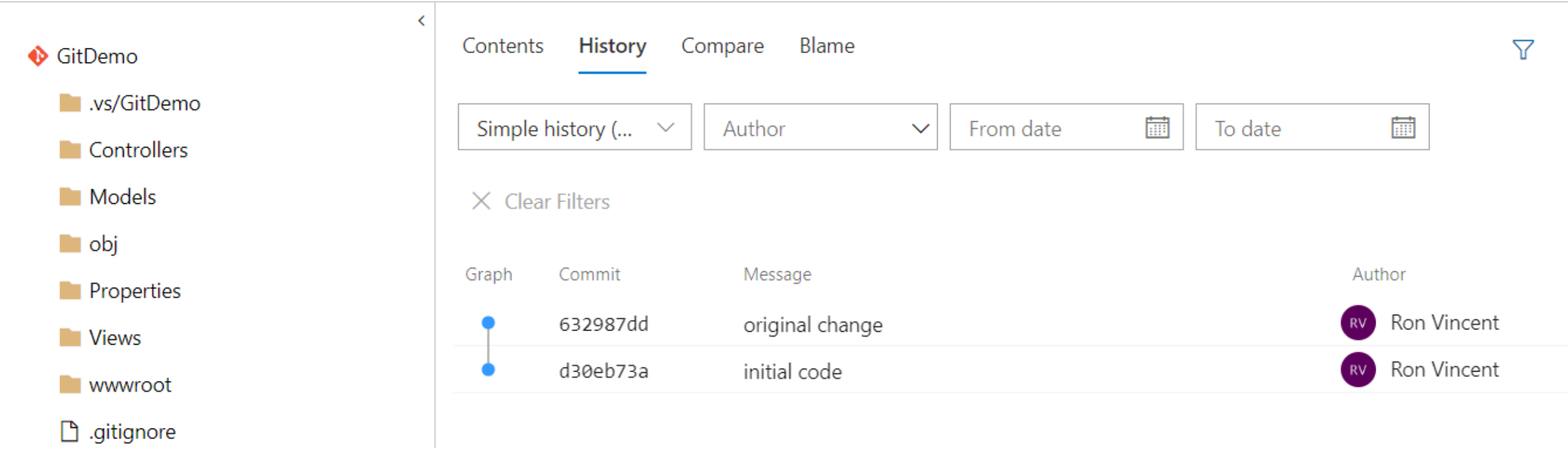
View Historical Data in Visual Studio

- Commit Details



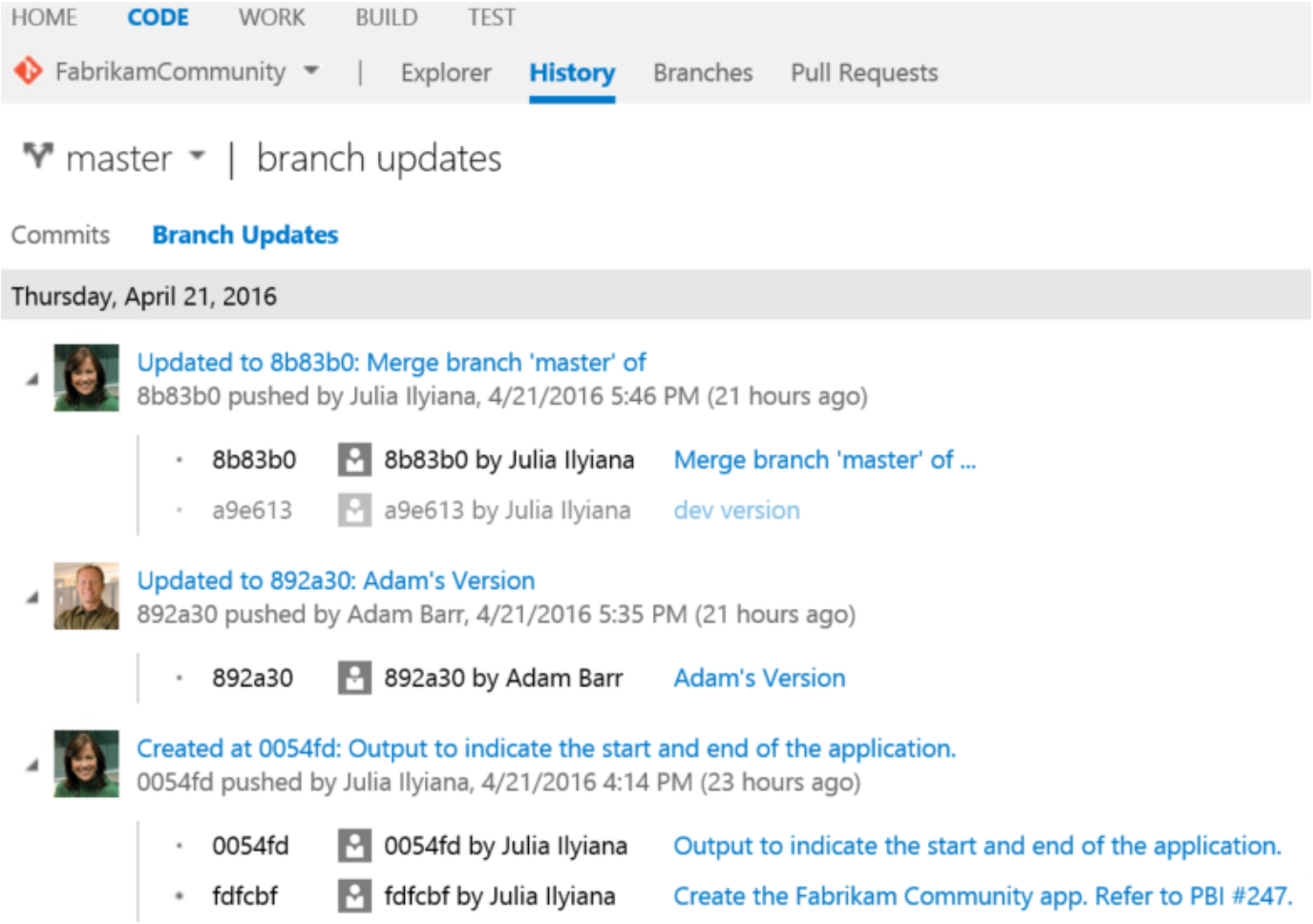
View Historical Data in Web Browser

- File History



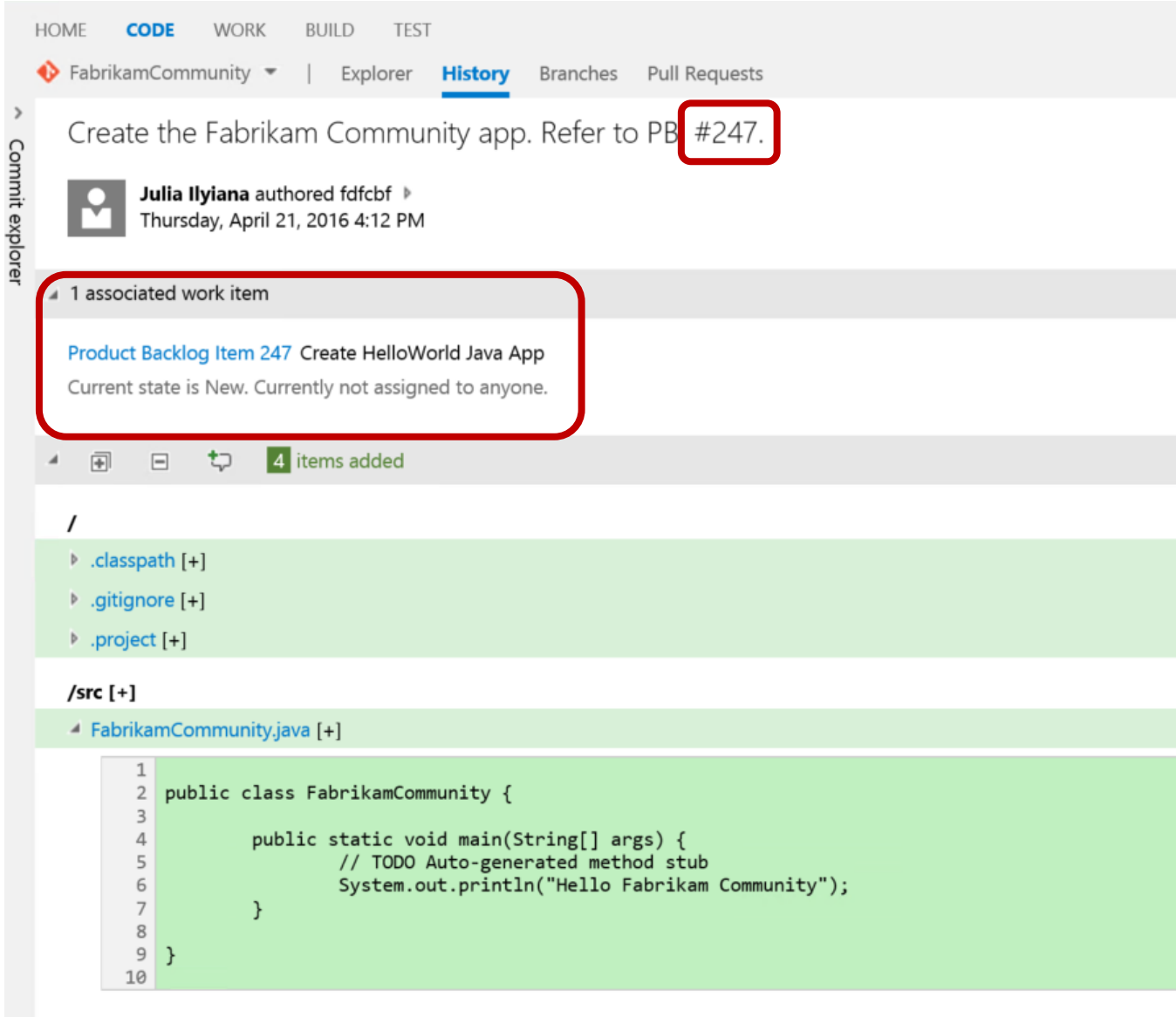
View Historical Data in Web Browser (continued)

- Branch History



View Historical Data in Web Browser (continued)

- Commit Details



Tags

- Pointer to a specific commit
- Use tags to label a certain point in time in your codebase
- Apply tags from the command prompt

```
git tag -a v1.0 -m "MVP"
git push origin v1.0
```

Demo: View History and Tags

Microsoft Git Solutions

Pull request

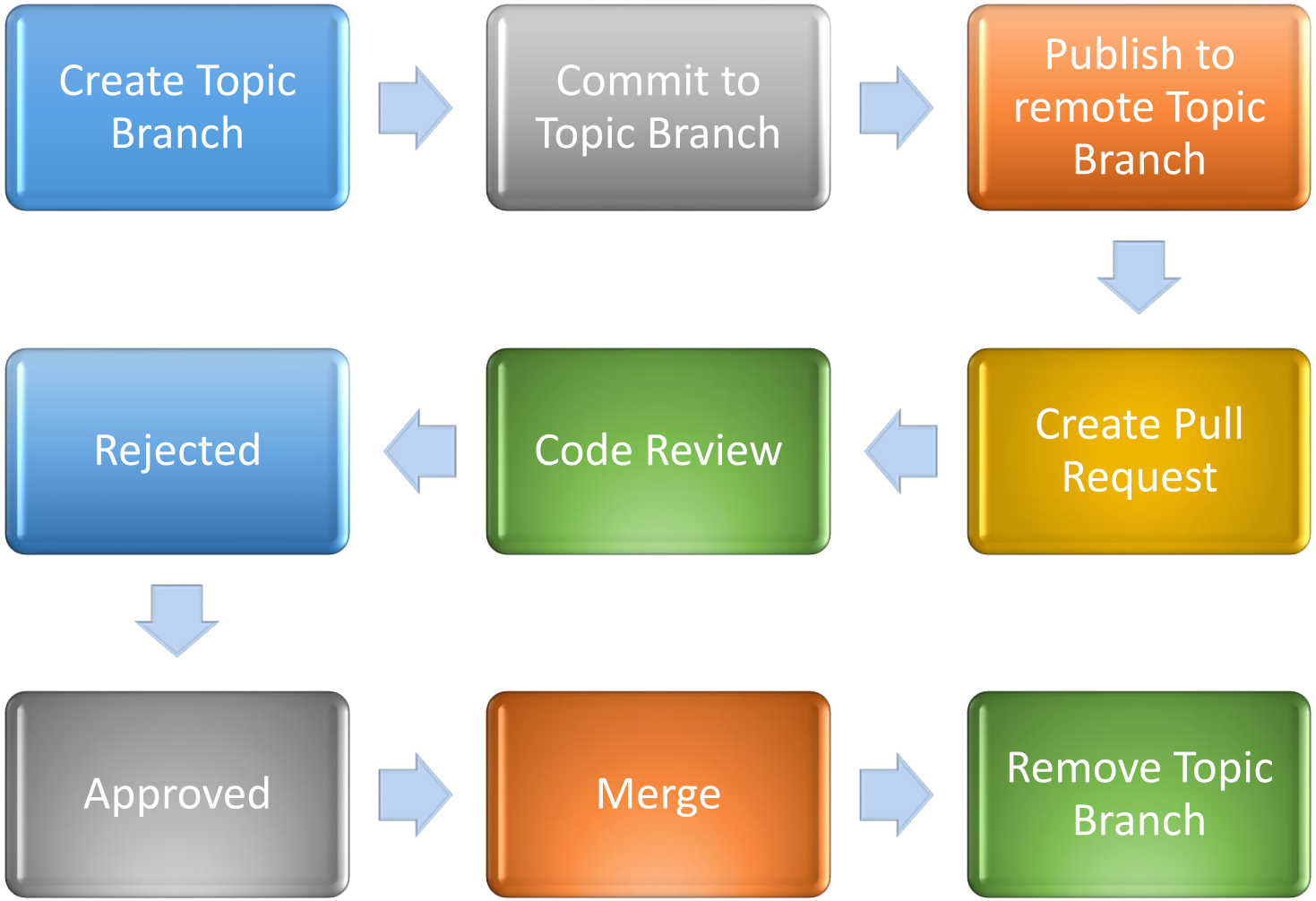
Overview

- Pull Request
- Azure DevOps Implementation
  - Workflow
  - Notifications

Pull Request

- Code Review
- Devs work on Topic Branches
- Merge to Target when ready
- Requires approval from a reviewer
  - Team or Individual

Pull Request Workflow



TFS Notifications

- Pull Requested created – Team
- Pull Request created or updated (Repo)
- Pull Request created on updated (Team Project)
- Pull request updated (Participant)

Demo: Pull Request

Microsoft Git Solutions

Branch Policies

Branch Policies

- Branch policies help teams protect their important branches.
  - Ensure no broken builds
  - Developer code reviews
- Policies work in conjunction with pull requests
- Developers publish their code to a topic branch
- Branch policies are evaluated on the serve
- Code meeting the policy will be merged when the pull request is complete
- Exempt from policy enforcement permission granted to a user or group can skip the policy requirement

Branch Policies Continued

- Wildcard characters are supported when configuring required code reviewers
  - Single asterisks (*) are supported
  - Examples
    - *.sql will match all files with the .sql extension
    - /ConsoleApplication/* will match all files under the folder named ConsoleApplication
    - /.gitattributes will match the .gitattributes file in the root of the repo
    - */.gitignore will match any .gitignore file in the repo
- Policies are not case-sensitive at this time
- You can add the users to a group, and then add the group as a reviewer where any member of the group can then approve on behalf of the group to meet the policy requirement

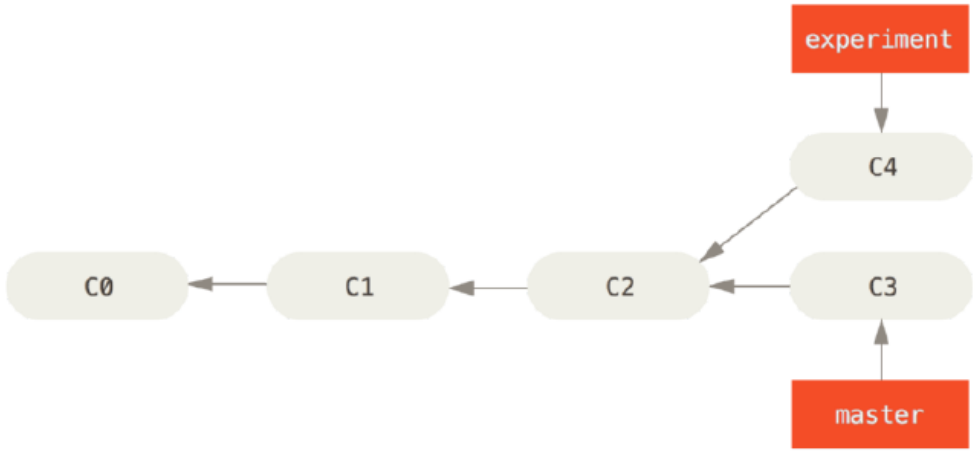
Demo: Branch Policies

Microsoft Git Solutions

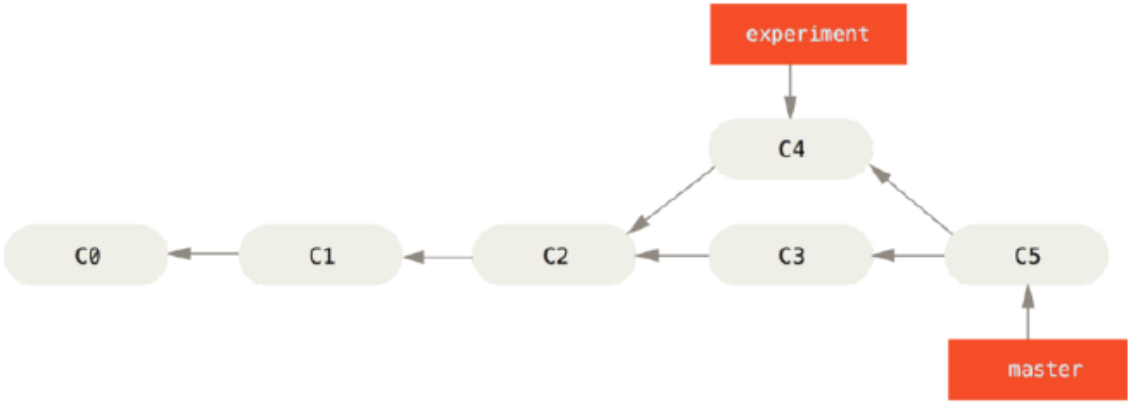
Merge or Rebase

Rebase or Merge

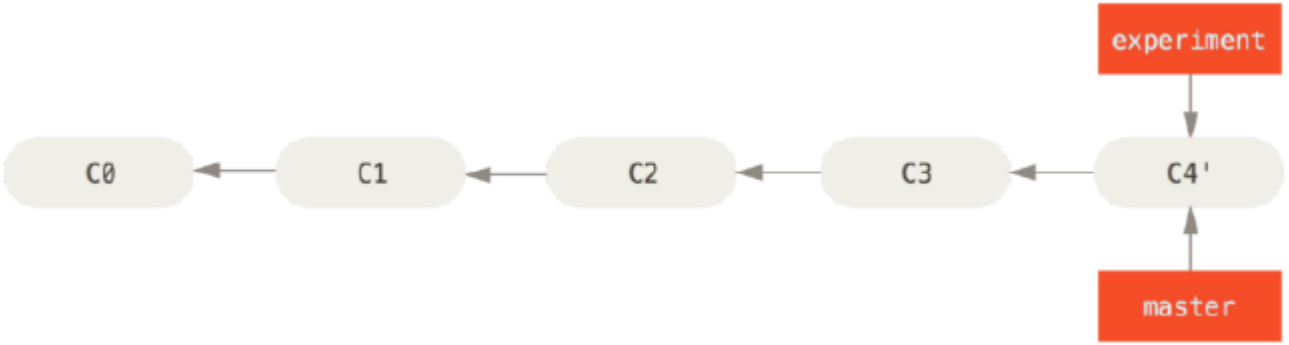
- We created a branch and made changes, meanwhile, changes have been made on master.



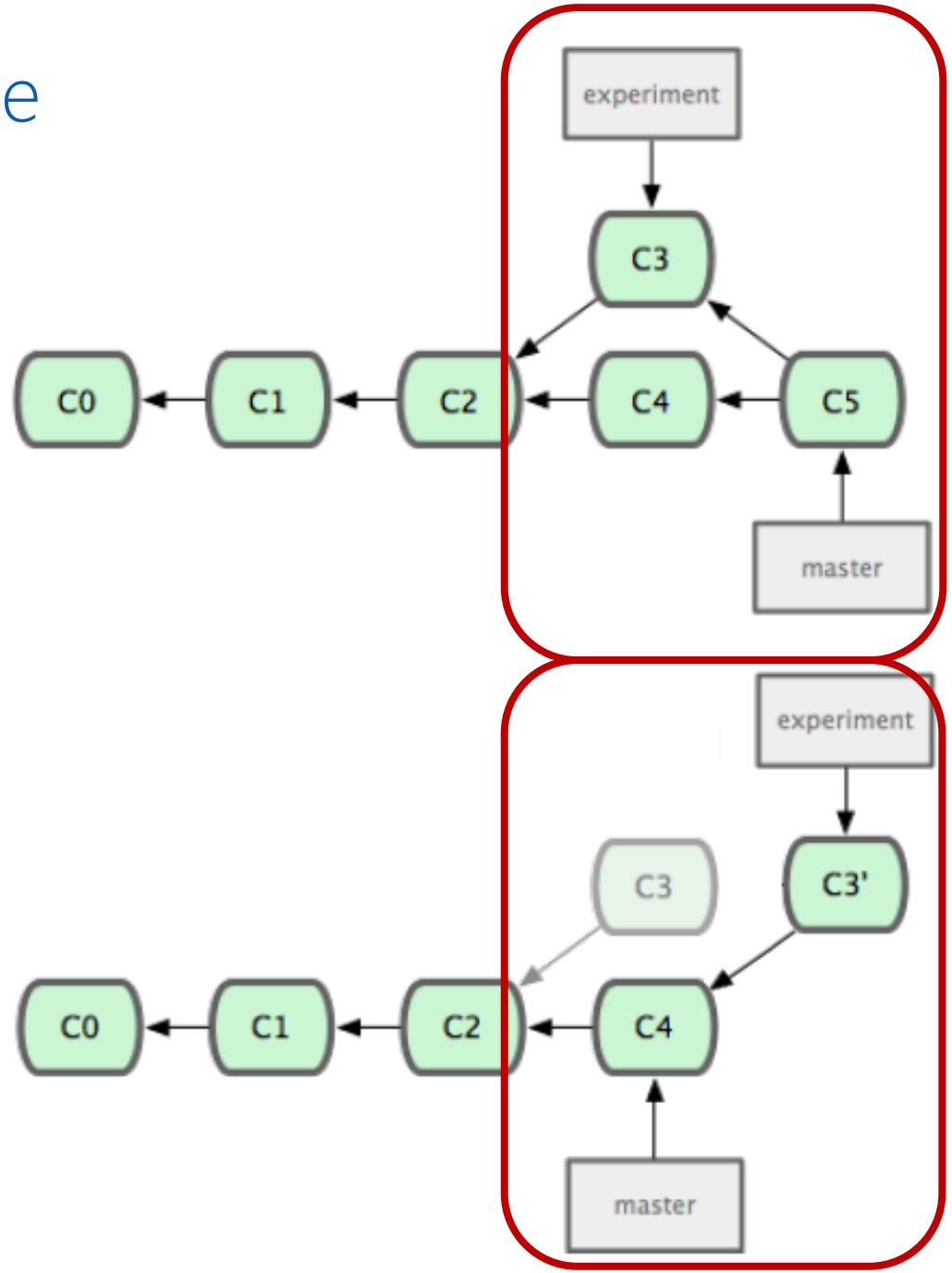
- Merge takes all the changes in one branch and merges them into another branch in one commit.



- Rebase says I want the point at which I branched to move to a new starting point.



Merge vs. Rebase



Demo: Rebase

Microsoft Git Solutions

Move Content Between TFS & Git

Microsoft Git Solutions

Git Permissions

Git Permissions

Permission name	TFSSecurity Action	TFSSecurity Namespace	Description	Contributors	Build Administrators	Project Administrators (Note 1)
Administer (Note 2)	Administer	GitRepositories	Can rename the repository, add additional repositories, verify the database, and set permissions for the branch. Users who have this permission can delete the repository if they also have the Force permission. At the branch level, can set permissions for the branch and delete the branch.			✓
Branch Creation	CreateBranch	GitRepositories	Can publish branches in the repository. Lack of this permission does not limit users from creating branches in their local repository; it merely prevents them from publishing local branches to the server. When a user creates a new branch on the server, they have Administer, Contribute, and Force permissions for that branch by default.	✓	✓	✓
Contribute	GenericContribute	GitRepositories	Can push their changes to the repository. At the branch level, can push their changes to the branch.	✓	✓	✓
Note Management	ManageNote	GitRepositories	Can push and edit Git notes to the repository. They can also remove notes from items if they have the Force permission. See this topic for more details on notes.	✓	✓	✓
Read	GenericRead	GitRepositories	Can clone, fetch, pull, and explore the contents of the repository, but cannot push any changes they make to the repository.	✓	✓	✓
Rewrite and destroy history (force push)	ForcePush	GitRepositories	Can force an update, which can overwrite or discard commits from any user. Deleting commits changes the history. Without this permission, users cannot discard their own changes. Rewrite and destroy history is also required to delete a branch. Because Rewrite and destroy history enables users to change the history or remove a commit from history, users who have this permission can delete a change and its history from the server. They can also modify the commit history of the server repository. At the branch level, can rewrite and destroy history of the branch.			
Tag Creation	CreateTag	GitRepositories	Can push tags to the repository, and can also edit or remove tags from items if they have the Force permission.	✓	✓	✓

Git Permissions Continued

Permission name	TFSSecurity Action	TFSSecurity Namespace	Description	Contributors	Build Administrators	Project Administrators
Read	GenericRead	GitRepositories	Can clone, fetch, pull, and explore the contents of the repository, but cannot push any changes they make to the repository.	✓	✓	✓
Rewrite and destroy history (force push)	ForcePush	GitRepositories	Can force an update, which can overwrite or discard commits from any user. Deleting commits changes the history. Without this permission, users cannot discard their own changes. Rewrite and destroy history is also			
			Because Rewrite and destroy history enables users to change the history or remove a commit from history, users who have this permission can delete a change and its history from the server. They can also modify the commit history of the			
			At the branch level, can rewrite and destroy history of the branch.			
Tag Creation	CreateTag	GitRepositories	Can push tags to the repository, and can also edit or remove tags from items if they have the Force permission.	✓	✓	✓

Microsoft Git Solutions

Git Network

Git Network

- Visual Studio uses libgit2.
- An endpoint for a clone, fetch, push, or pull operation in Git is called a “remote.”
- Visual Studio’s Git support contains support for remotes using the following protocols:
  - HTTP (i.e. https://github.com/libgit2/TestGitRepository, or https://yourname.visualstudio.com/DefaultCollection/_git/YourRepo)
  - File paths (i.e. F:\myrepo.git, or \\servername\sharename\myrepo.git)
  - git:// URIs
  - SSH based remote support is on the roadmap
- Authentication with HTTP endpoints
  - Basic authentication
  - Integrated Windows authentication
  - Team Foundation Service federated authentication
  - Personal access tokens
    - Personal access tokens can be used instead of a password to allow applications outside the browser access to the resources stored in your account.

Microsoft Git Solutions

Move a Repo to another Team Project

Summary

- Overview
- Git Tools & Clients
- Use Command Prompt
- Migrate TFS Skills to Git
- Set up Git
- Develop using Git
- Collaborate using Git
- Use Branches
- Resolve Conflicts
- View History and Tags
- Pull Request
- Branch Policies
- Merge or Rebase
- Git Permissions
- Git Network
- Repo Rename
- Git Features new to TFS 2015 Update 2
- Hands on Labs

Lab: Microsoft Git Solutions

Microsoft