

# Version Control w/ GIT

PowerShell The Developer Way



# Mark Warneke

<http://aka.ms/mark>

---

Premier Field Engineer  
Business Application – EMEA  
Microsoft Services



Version Control General

GIT

Repository & Build

# CONTENT

What is Version Control

How to use Version Control

Why use version control

Basic usage of GIT and VSTS

# OBJECTIVE



A blue-tinted background image showing the lower legs and feet of several people standing on a stage. They are wearing dark trousers and sneakers. The image is slightly out of focus, emphasizing the text in the foreground.

# Version Control

# Version Control

## What is Version Control?

Function\_1  
.txt



Function\_2  
.txt



180101\_  
Function1  
.txt



180101\_01\_  
Function1  
.txt



180101\_01\_  
Function1  
\_FINAL  
.txt

"Hello World"

```
function {  
  "Hello World"  
}
```

```
function ($s) {  
  $s  
}
```

```
function ([string]$s) {  
  $s  
}
```

```
# My Function 1  
function ([string]$s) {  
  if(-Not Empty($s)) {  
    return $s  
  }  
}
```



# Version Control

## What is Version Control?



Create Workflows



Work w/ Versions



Code Together



Keep History



Automate Tasks

# Version Control

## Commit





# Version Control



## File itself:

Latest File Content

## Commit:

Changes/Differences  
From Latest Version

## Commit

# Mark  
"Hello World"

# Mark  
# TODO: function  
"Hello World"

# Mark  
# My function  
function (\$s) {  
 \$s  
}

# Mark  
# My function  
function ([string]\$s) {  
 return \$s  
}



initial



adds todo



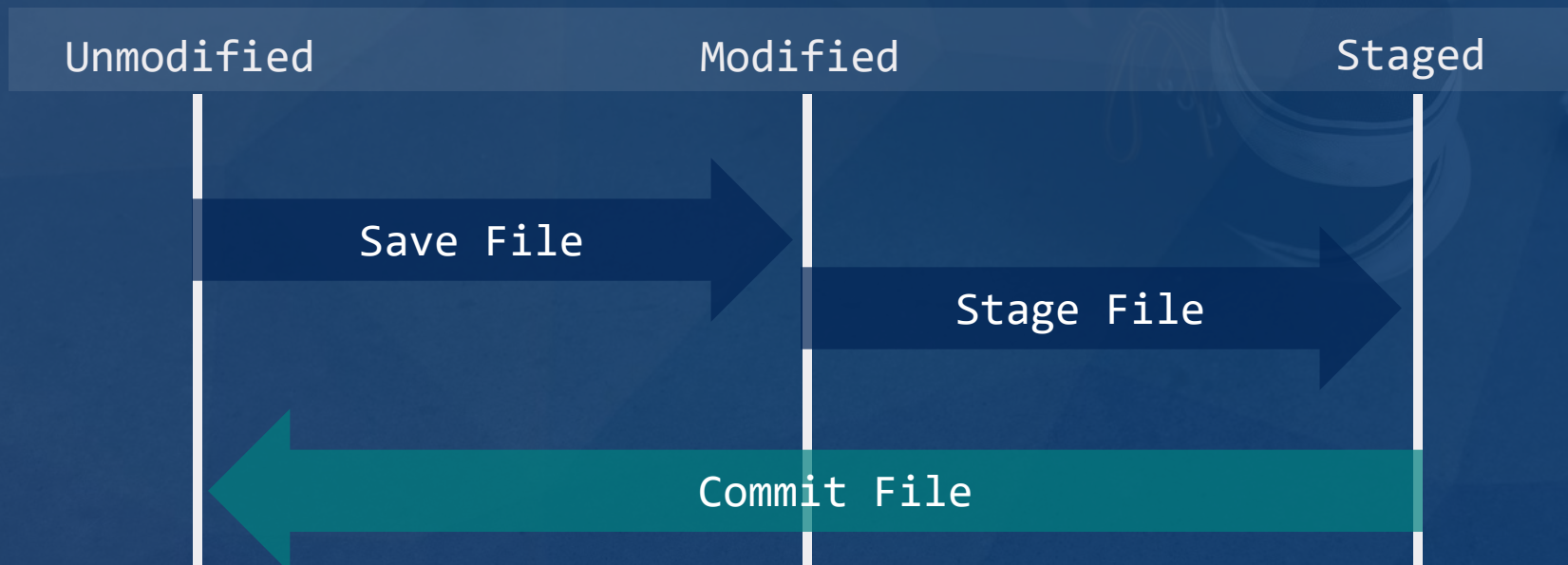
adds function



master

# Version Control

## Files and commits





# Version Control

---

## Commit



# Version Control

## Branch

Feature



# Version Control

## Why Use Version Control

Simultaneous  
Development

Quality

Workflow

History



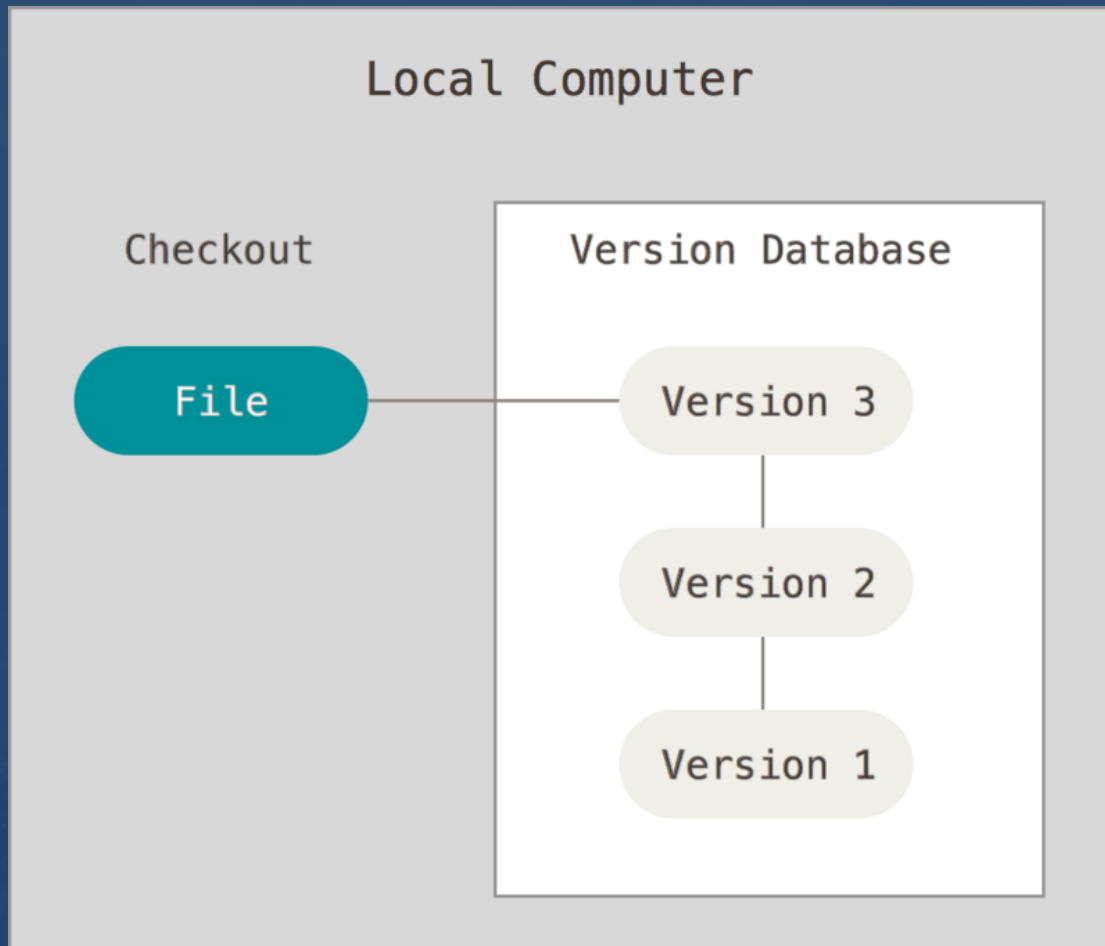
The background is a dark blue gradient with a faint, semi-transparent image of a desk. On the desk, there is a rotary telephone in the upper right, a pen, and a pencil. The word "Repository" is written in a large, white, sans-serif font, centered horizontally. A thin white horizontal line is positioned directly beneath the text.

# Repository



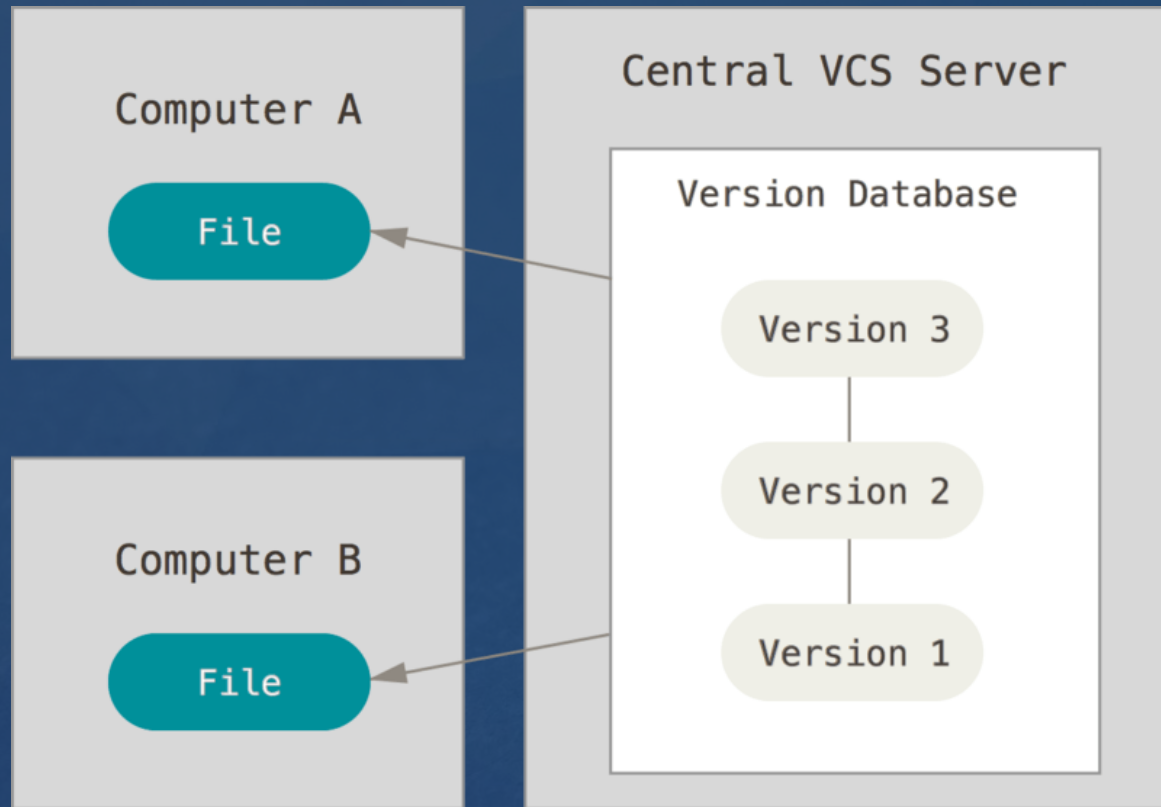
# Repository

## Local Version Control Systems



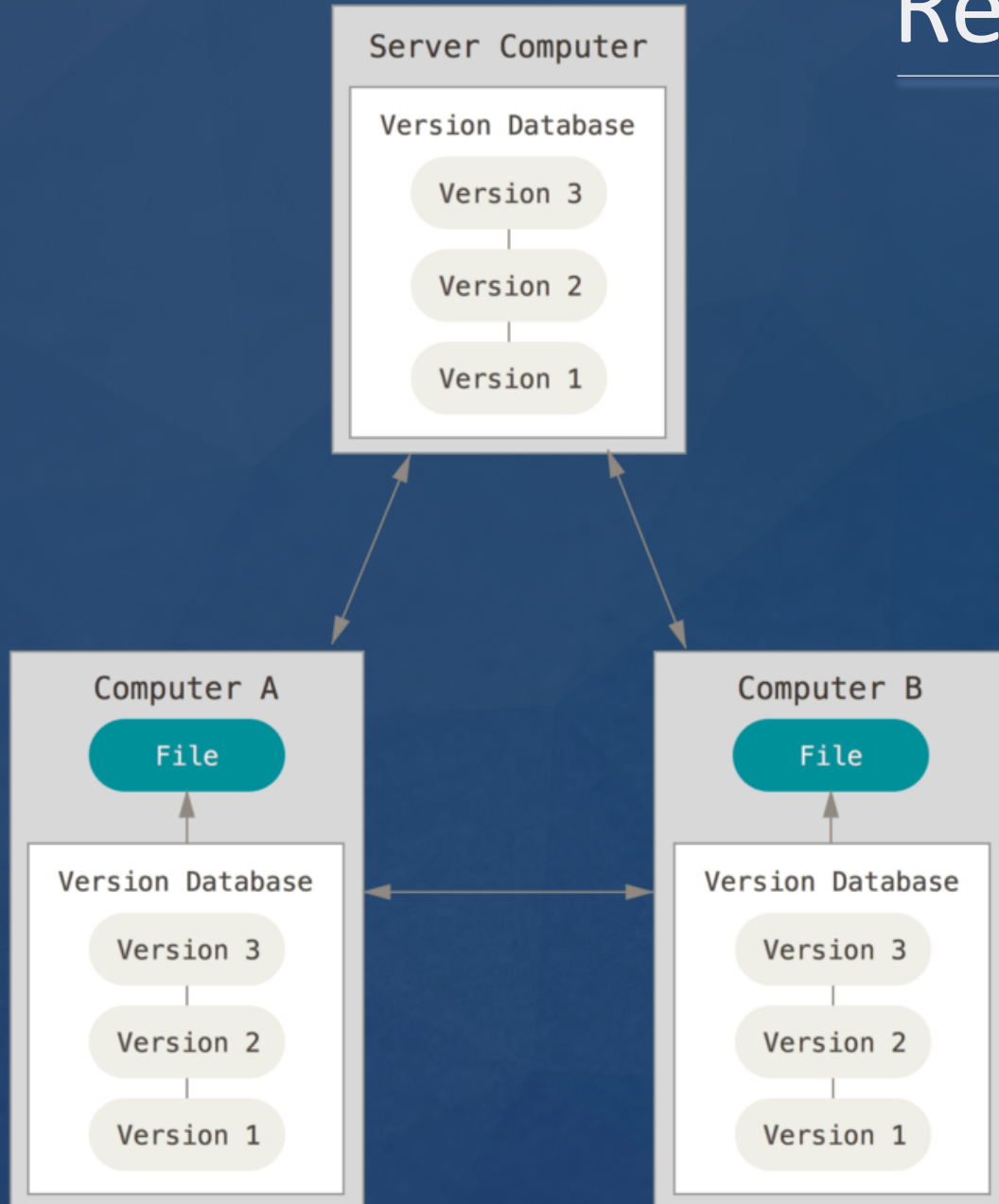
# Repository

## Centralized Version Control Systems



# Repository

## Distributed Version Control Systems





The background is a dark blue, semi-transparent overlay on a photograph of a desk. The desk features a white pen holder filled with various pens and pencils, a stack of papers, and a blue folder. The overall aesthetic is professional and clean.

# GIT

# GIT

---

## Benefits of GIT

Simultaneous  
Development

Faster  
Releases

Built-in  
Integration

Strong  
Community

Git in Team

Pull  
Requests

Branch  
Policies

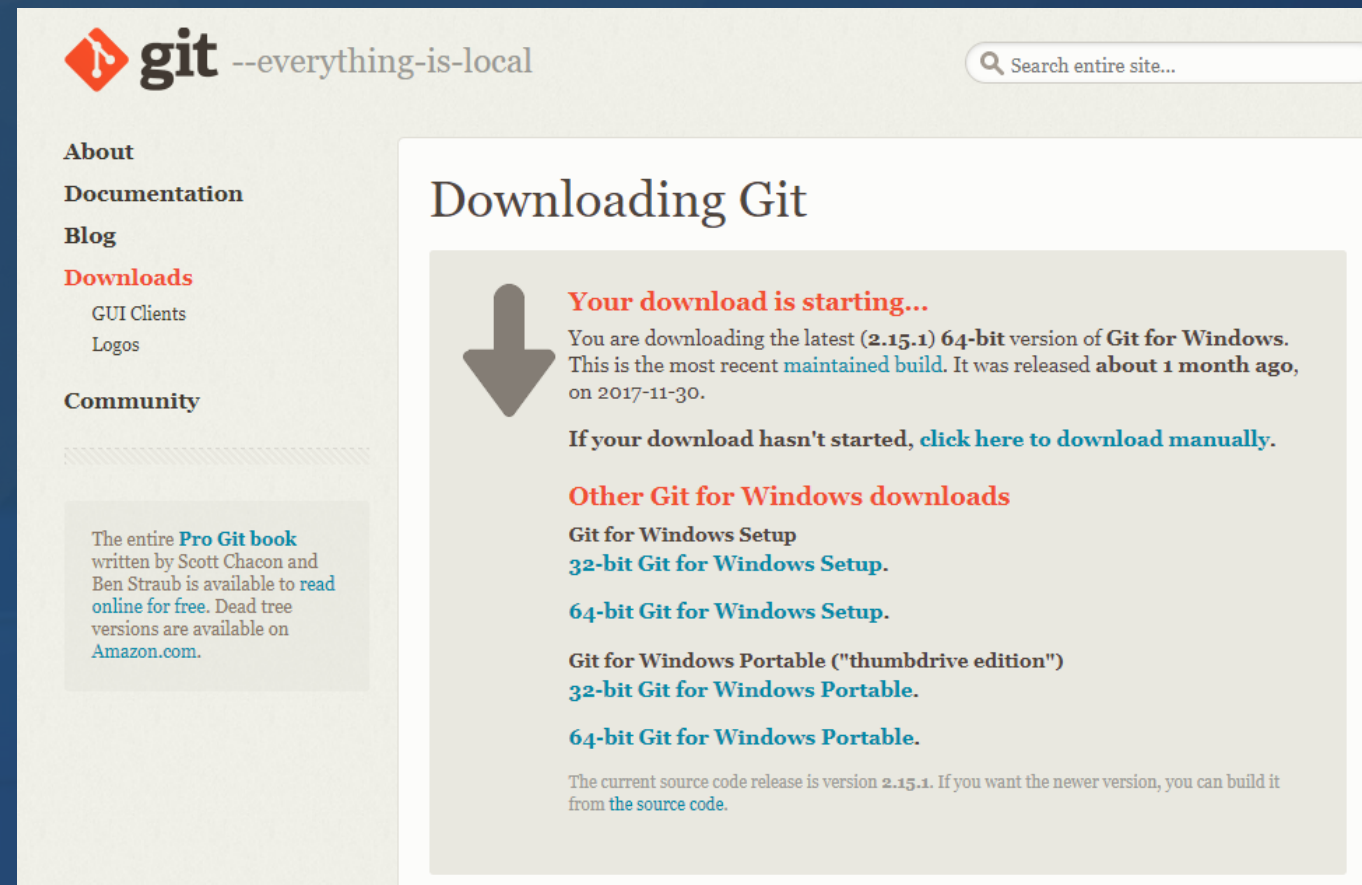
# GIT

## Install Git

<https://git-scm.com/download/win>

```
# add git executable to $env:PATH
```

```
Install-Module posh-git
```



The screenshot shows the Git website's download page for Windows. The header features the Git logo and the tagline "--everything-is-local", along with a search bar. A sidebar on the left contains links to "About", "Documentation", "Blog", "Downloads" (highlighted in red), and "Community". Under "Downloads", there are links for "GUI Clients" and "Logos". A text box in the sidebar promotes the "Pro Git book" by Scott Chacon and Ben Straub, available for free online or on Amazon. The main content area is titled "Downloading Git" and features a large downward arrow icon. The text states: "Your download is starting..." followed by details about the latest (2.15.1) 64-bit version of Git for Windows, noting it's the most recent maintained build released about a month ago. It provides a link to download manually if the automatic download hasn't started. Below this, a section titled "Other Git for Windows downloads" lists links for "Git for Windows Setup", "32-bit Git for Windows Setup", "64-bit Git for Windows Setup", "Git for Windows Portable ('thumbdrive edition')", "32-bit Git for Windows Portable", and "64-bit Git for Windows Portable". At the bottom, it mentions the current source code release is version 2.15.1 and provides a link to the source code.


**git** --everything-is-local

Search entire site...

**About**  
**Documentation**  
**Blog**  
**Downloads**  
GUI Clients  
Logos  
**Community**

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

## Downloading Git

 **Your download is starting...**

You are downloading the latest (**2.15.1**) **64-bit** version of **Git for Windows**. This is the most recent [maintained build](#). It was released **about 1 month ago**, on 2017-11-30.

If your download hasn't started, [click here to download manually](#).

### Other Git for Windows downloads

**Git for Windows Setup**  
[32-bit Git for Windows Setup](#).  
[64-bit Git for Windows Setup](#).

**Git for Windows Portable ("thumbdrive edition")**  
[32-bit Git for Windows Portable](#).  
[64-bit Git for Windows Portable](#).

The current source code release is version **2.15.1**. If you want the newer version, you can build it from [the source code](#).



# GIT

---

## Install Git Chocolatey

```
# install chocolatey
```

```
Set-ExecutionPolicy Bypass -Scope Process -Force
```

```
iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
```

```
# install git via chocolatey
```

```
choco install git
```

```
# Install posh git
```

```
PowerShellGet\Install-Module posh-git -Scope CurrentUser
```

# GIT

---

## Setup Git

```
git config --global user.name "John Doe"  
git config --global user.email johndoe@example.com
```

```
git config --list
```

<https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setu>

# GIT

## Commands: Get Started

```
mkdir \dev\tmp\git
```

```
cd \dev\tmp\git\
```

```
git init
```

```
posh-git ~ git [master]
C:\dev\tmp> mkdir git

Directory: C:\dev\tmp

Mode                LastWriteTime         Length Name
----                -
d-----            1/10/2018   8:37 AM         git

C:\dev\tmp> cd git
C:\dev\tmp\git> git init
Initialized empty Git repository in C:/dev/tmp/git/.git/
C:\dev\tmp\git [master]> get-childitem -Hidden

Directory: C:\dev\tmp\git

Mode                LastWriteTime         Length Name
----                -
d--h--              1/10/2018   8:37 AM         .git

C:\dev\tmp\git [master]>
```

Initializes the Repository

Creates Hidden (.git) Folder

Local Database



# GIT

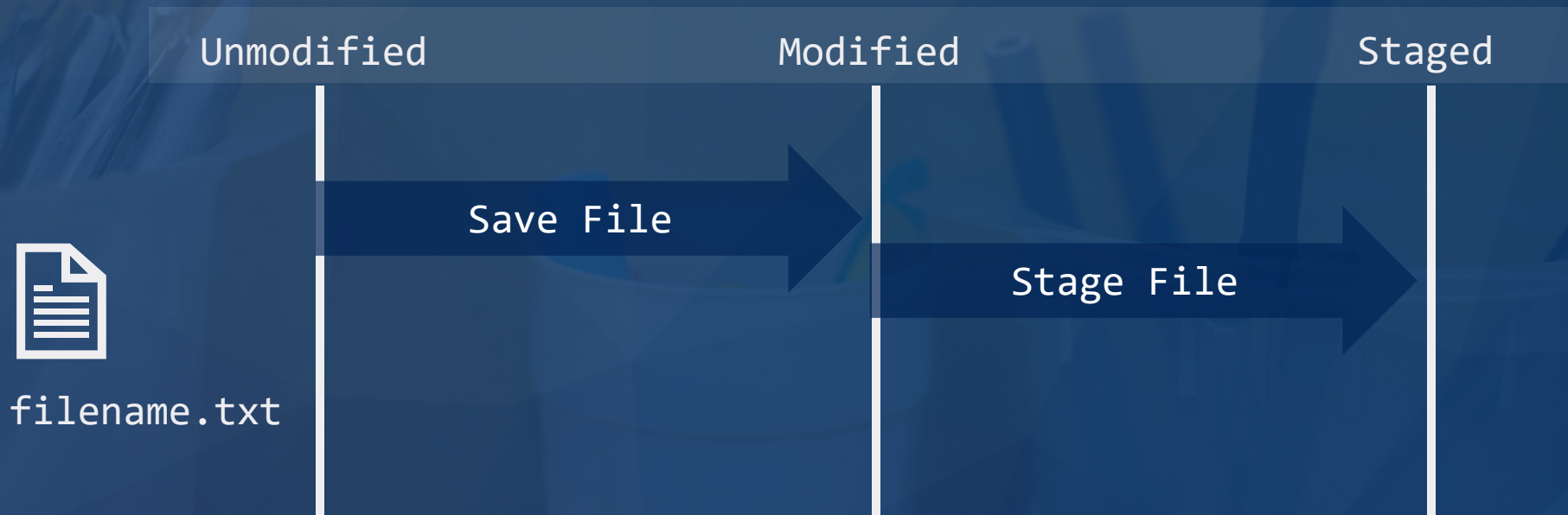
## Commands: Tracking Files

```
echo "Hello World" > filename.txt
```

Saves The File

```
git add ./filename.txt
```

Stages The File



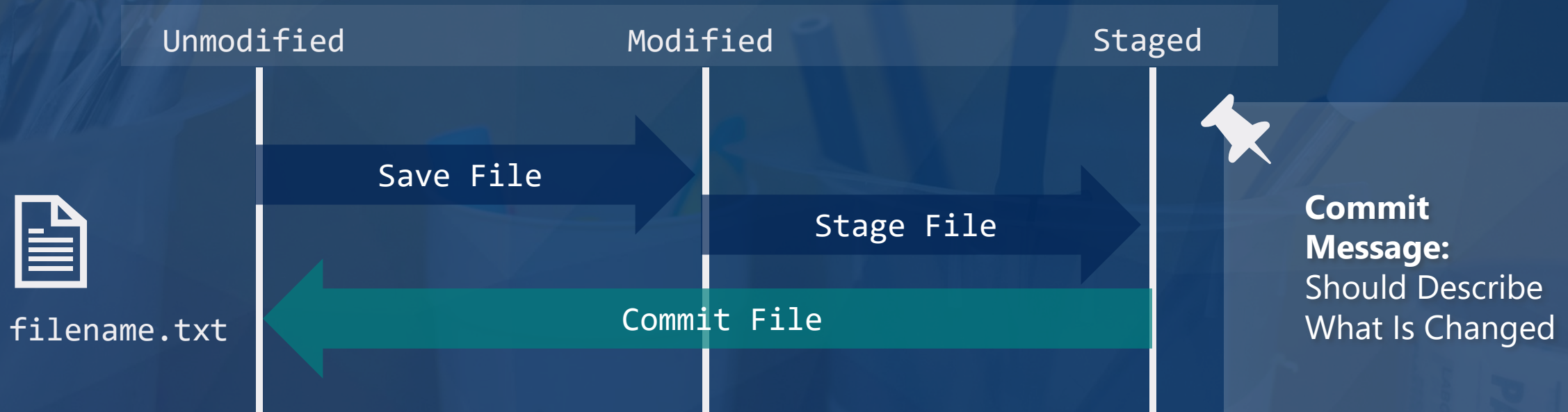
# GIT

## Commands: Commit

```
git commit -am "initial commit"
```

Commit All Staged Files

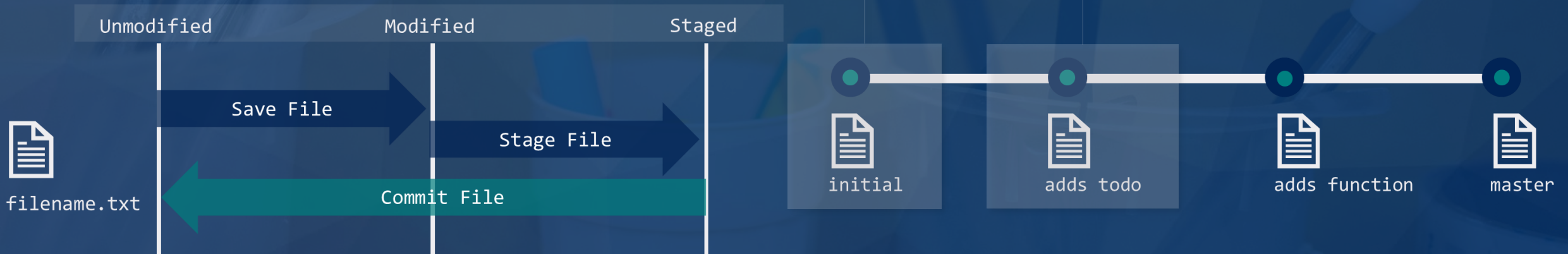
Commit Message



# GIT

## Commands: commit

```
# filename.txt already exists  
echo "//TODO" > filename.txt  
git add ./filename.txt  
git commit -am "adds todo"
```

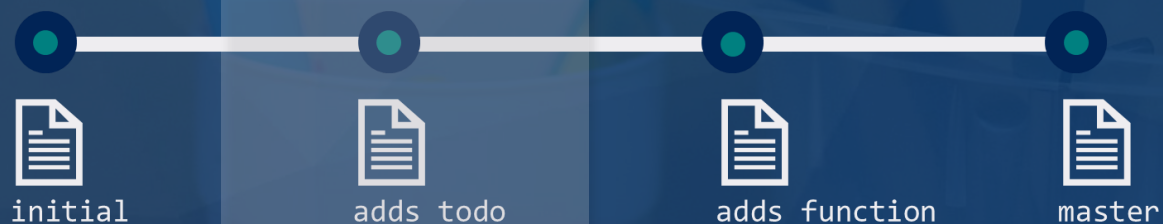




# GIT

## Commands: **Unmodifying**

```
git checkout -- ./filename.txt
```



# GIT

## Commands: Status

### git status

```
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

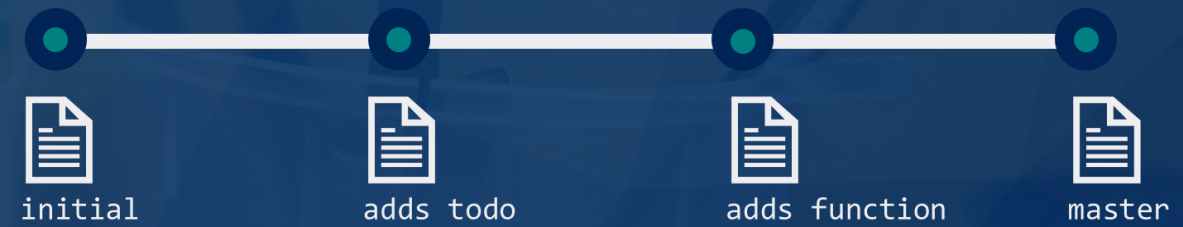
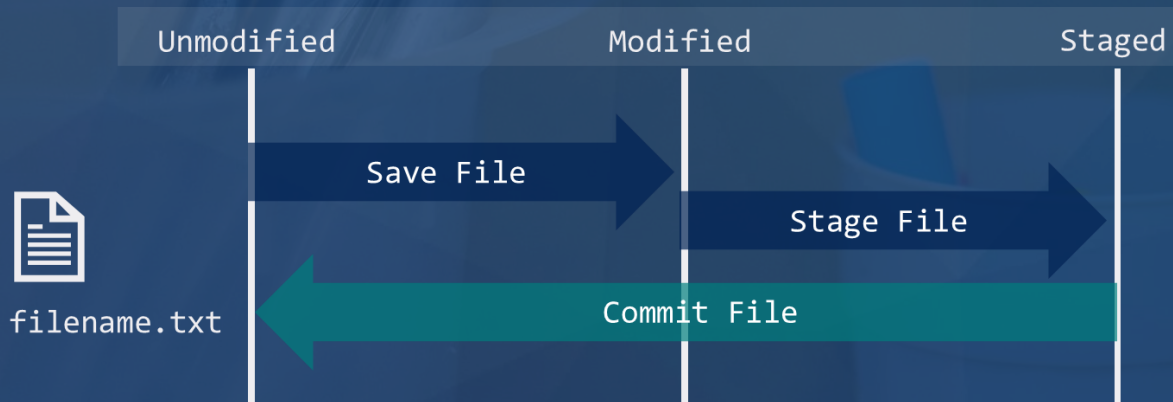
```
C:\dev\tmp\git [master]> echo "test" > filename.txt
C:\dev\tmp\git [master +1 ~0 -0 !]> git status
On branch master
```

No commits yet

Untracked files:  
(use "git add <file>..." to include in what will be committed)

filename.txt

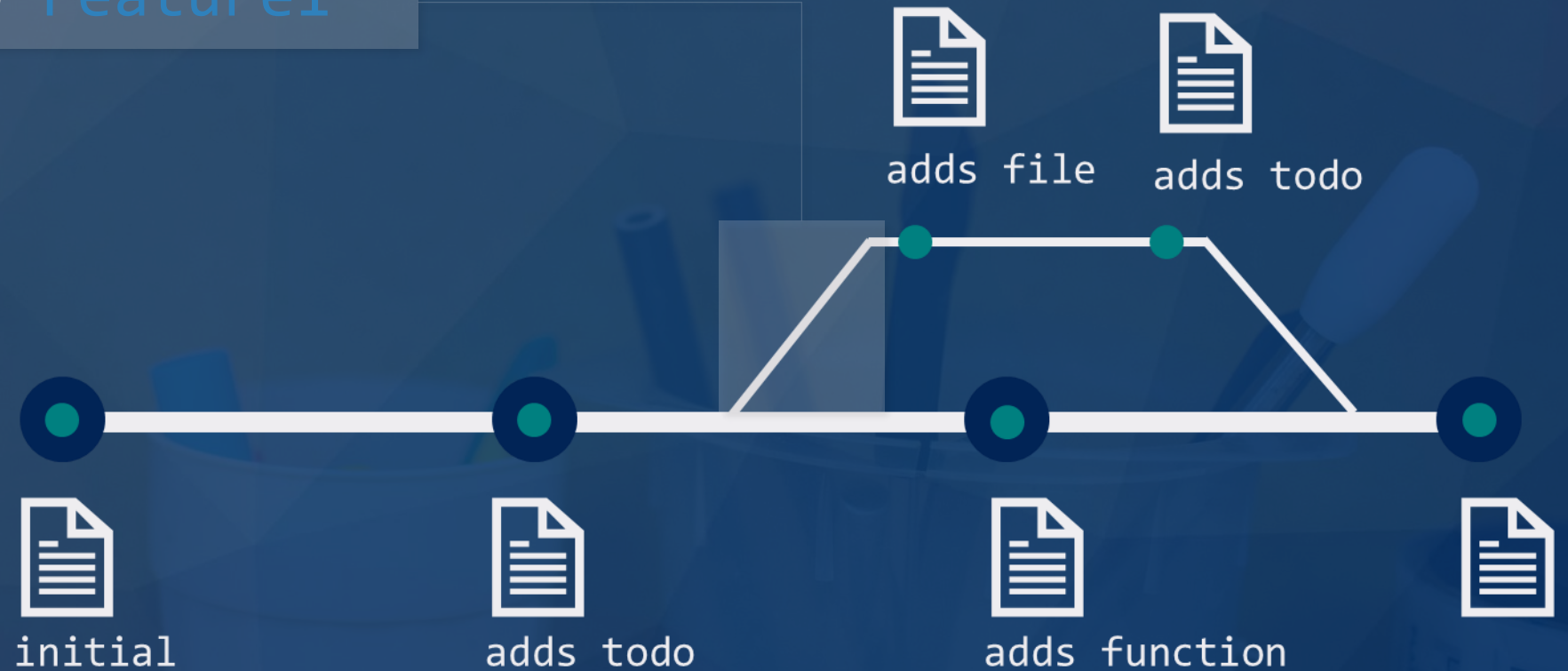
nothing added to commit but untracked files present (use "git add" to track)  
C:\dev\tmp\git [master +1 ~0 -0 !]>



# GIT

## Commands: Create Branch

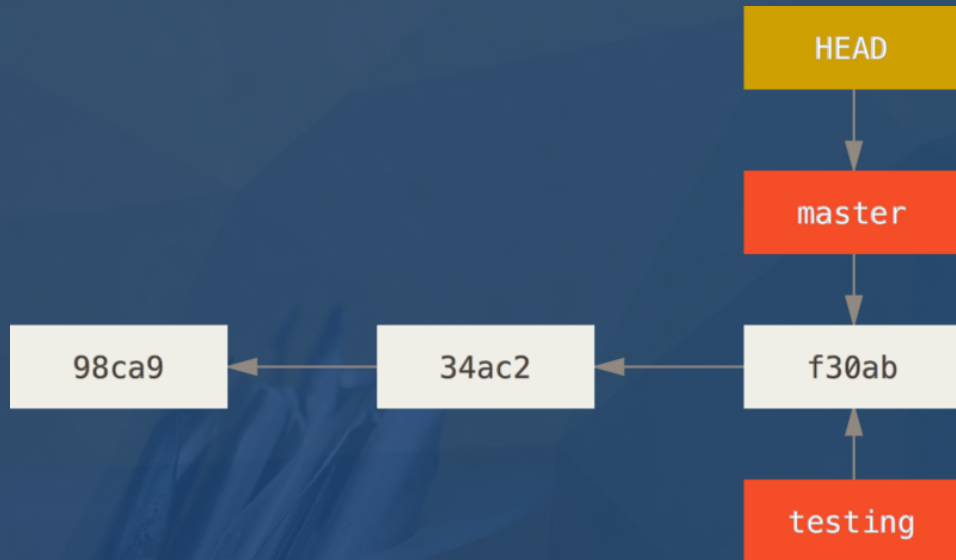
```
git checkout -b Feature1
```



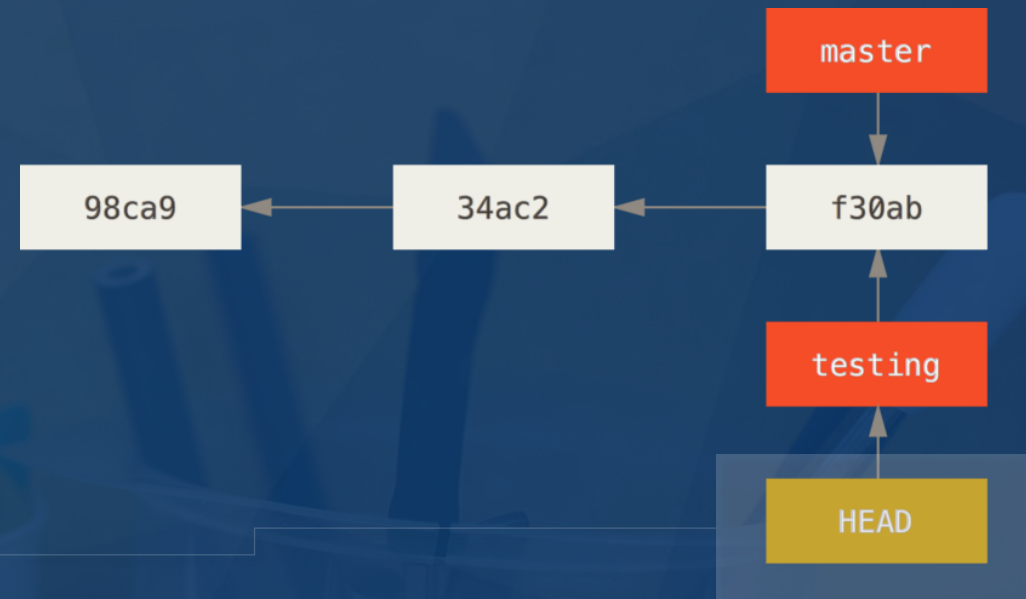


# Git Branching

## Commands: Switch Branch



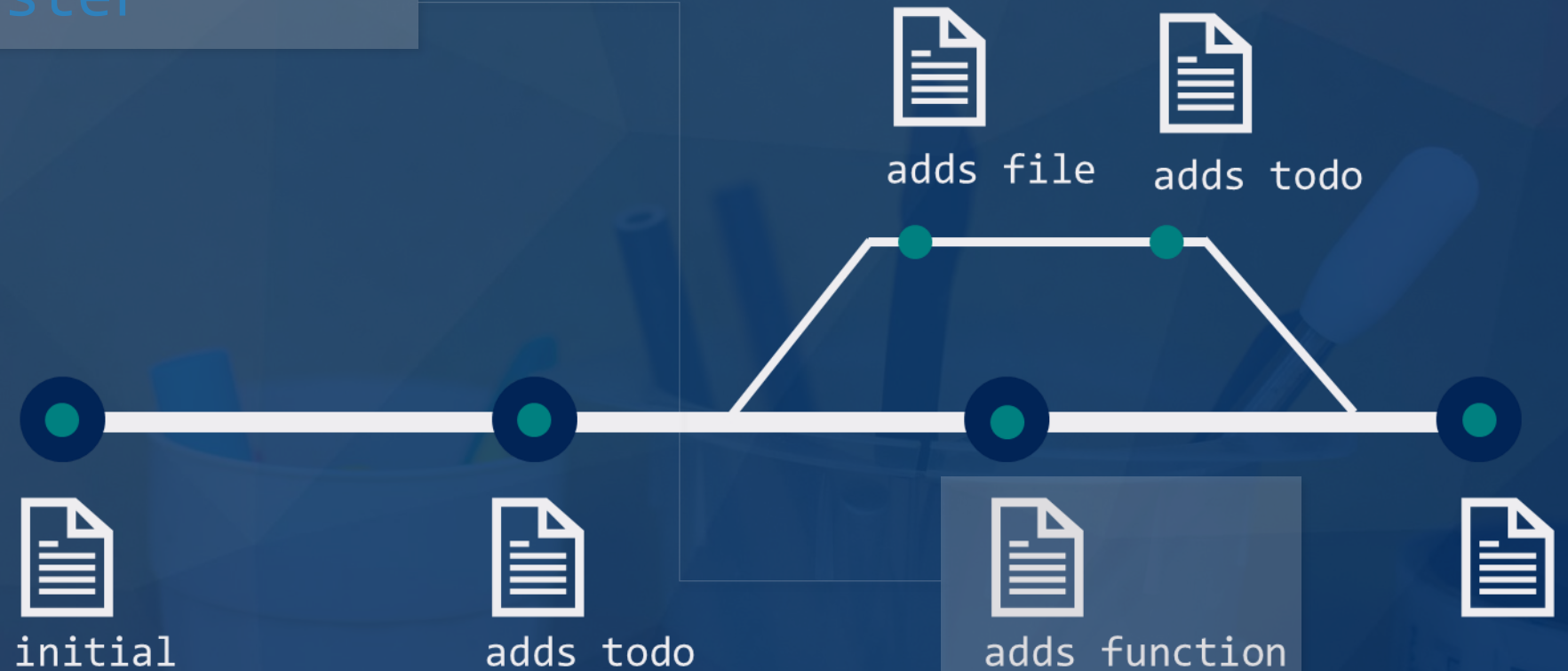
git **checkout** testing



# GIT

## Commands: Switch Branch

```
git checkout master
```



# GIT

## Commands: Switch Branch

```
git checkout master
```

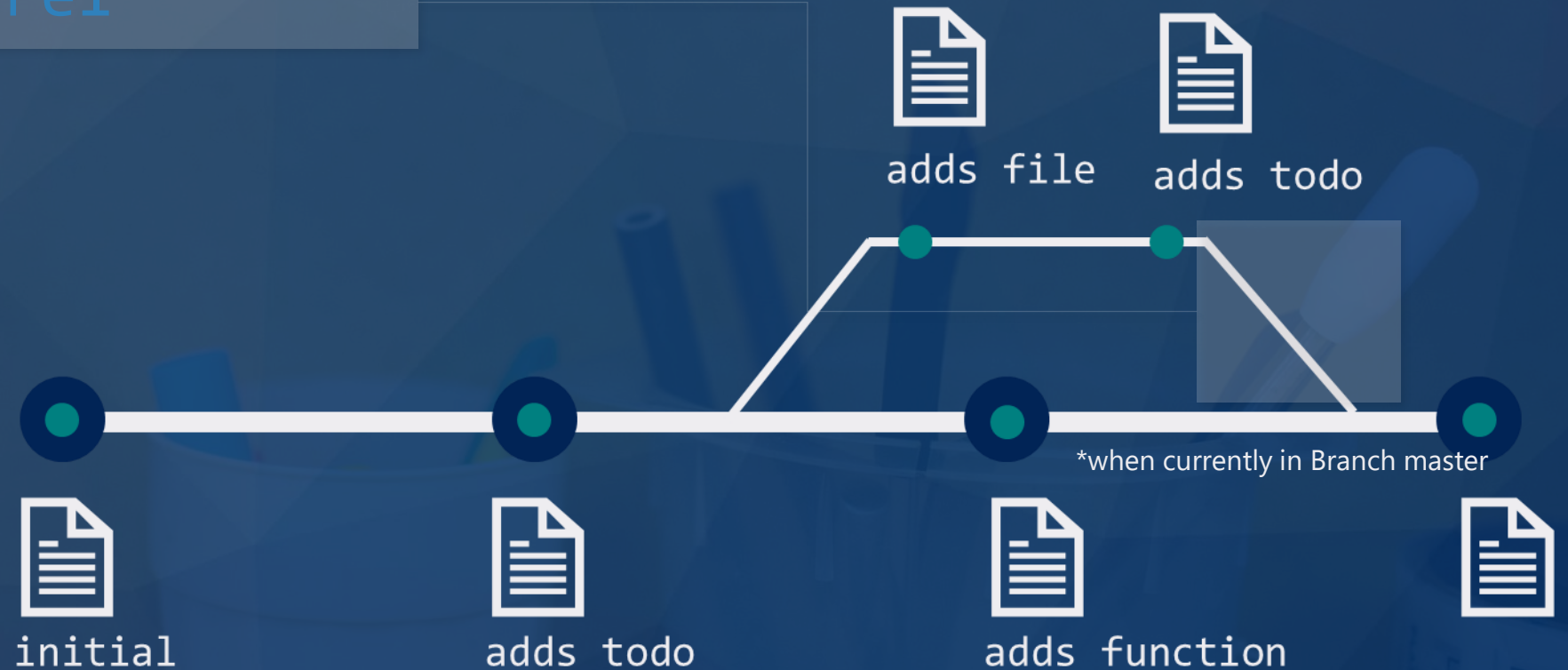




# GIT

## Commands: Merge Branch

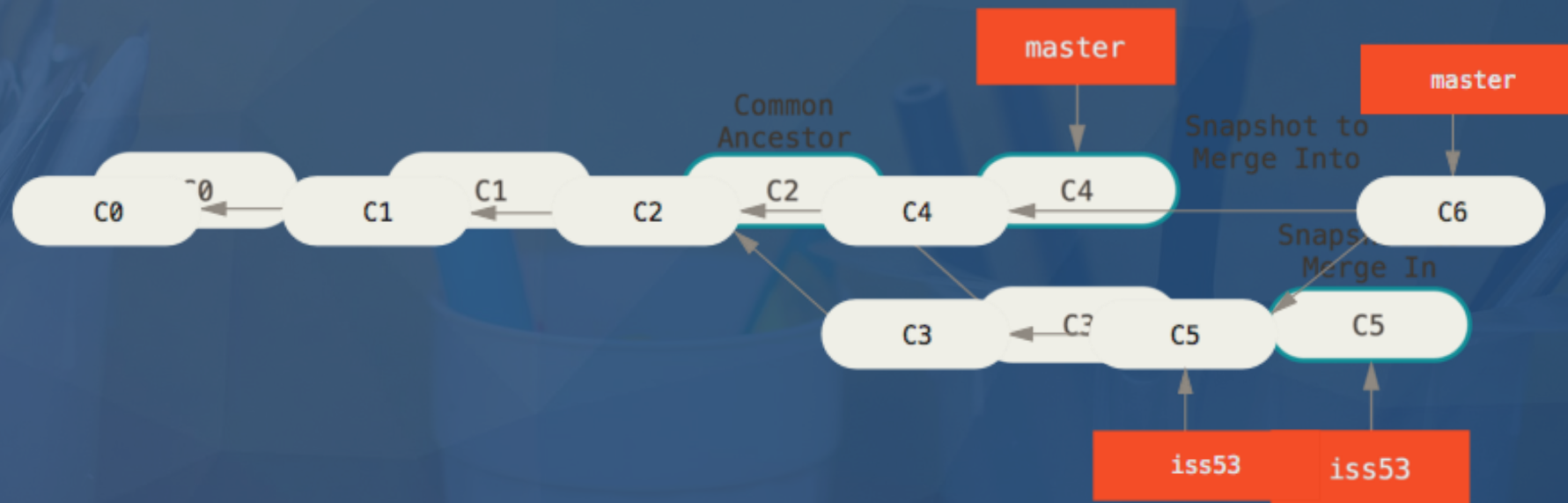
\* `git merge Feature1`



# GIT

## Commands: Merge Branch

\* `git merge iis53`



# GIT

## Commands: Merge Conflicts

`git merge iis53`

Auto-merging index.html

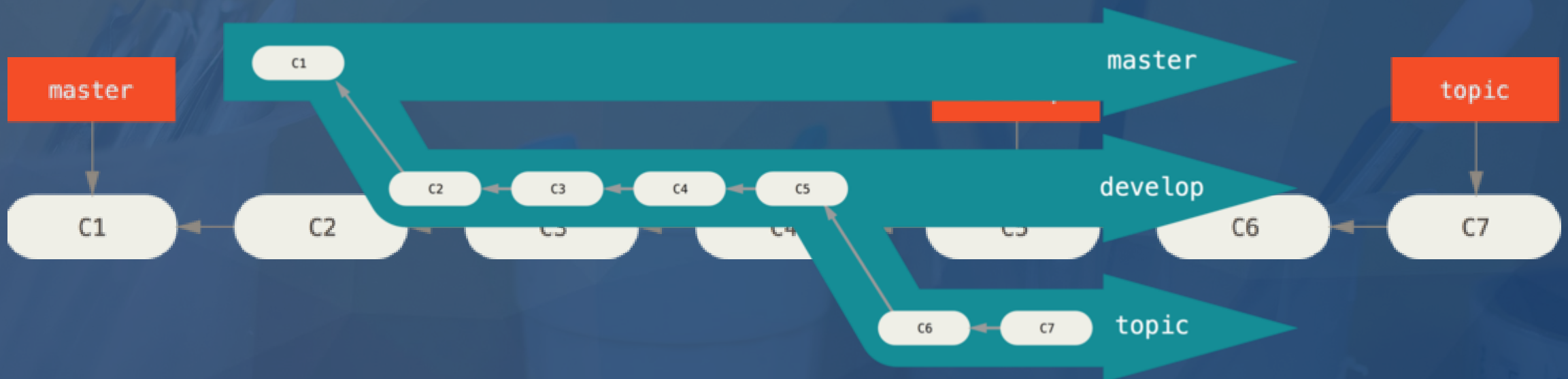
CONFLICT (content): Merge conflict in index.html

Automatic merge failed; fix conflicts and then commit the result.

```
<<<<<< HEAD:index.html
<div id="footer">contact :
email.support@github.com</div>
=====
<div id="footer">
    please contact us at support@github.com
</div>
>>>>>> iis53:index.html
```



# Branching Workflow



# GIT

---

## Commands: Clone Repository

```
git clone https://tfs... my-repo
```

```
Cloning into 'my-repo'...
```

```
remote: Reusing existing pack: 1857, done.
```

```
remote: Total 1857 (delta 0), reused 0 (delta 0)
```

```
Receiving objects: 100% (1857/1857), 374.35 KiB | 268.00 KiB/s,  
done.
```

```
Resolving deltas: 100% (772/772), done.
```

```
Checking connectivity... done.
```

```
cd my-repo
```

# GIT

---

## Commands: Add Repository

```
git remote add origin https://tfs...
```

```
git remote -v
```



# GIT

---

## Commands: Pull Repository

```
git pull origin
```

# GIT

---

## Commands: Push Repository

```
git push origin master
```

# GIT

---

## Commands: See Changes

git **diff**

```
diff --git a/Change.md b/Change.md
```

```
index 8ebb991..643e24f 100644
```

```
--- a/Change.md
```

```
+++ b/Change.md
```

```
@@ -65,7 +65,8 @@ This is a text that is in both version which  
gets
```

```
-merged in.
```

```
+merged in. Also, split your changes into comprehensive chunks  
if your patch is
```

```
+longer than a dozen lines.
```



# GIT

---

## Commands: Commit History

### git log

```
commit ca82a6dff817ec66f44342007202690a93763949
Author: Mark Warneke <mark.Warneke@microsoft.com>
Date:   Mon Mar 17 21:52:11 2018 -0700
```

changed the version number

```
commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Mark Warneke <mark.Warneke@microsoft.com>
Date:   Sat Mar 15 16:40:33 2018 -0700
```

removed unnecessary test

```
commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Mark Warneke <mark.Warneke@microsoft.com>
Date:   Sat Mar 15 10:31:28 2018 -0700
```

first commit

# GIT

---

## Commands: Tag

git tag

v0.1

v1.3

# GIT

---

## Ignore Files

### .gitignore

```
# ignore all .a files
*.a

# but do track lib.a, even though you're ignoring .a files above
!lib.a

# only ignore the TODO file in the current directory, not subdir/TODO
/TODO

# ignore all files in the build/ directory
build/

# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt

# ignore all .pdf files in the doc/ directory and any of its subdirectories
doc/**/*.pdf
```



# Repository

---

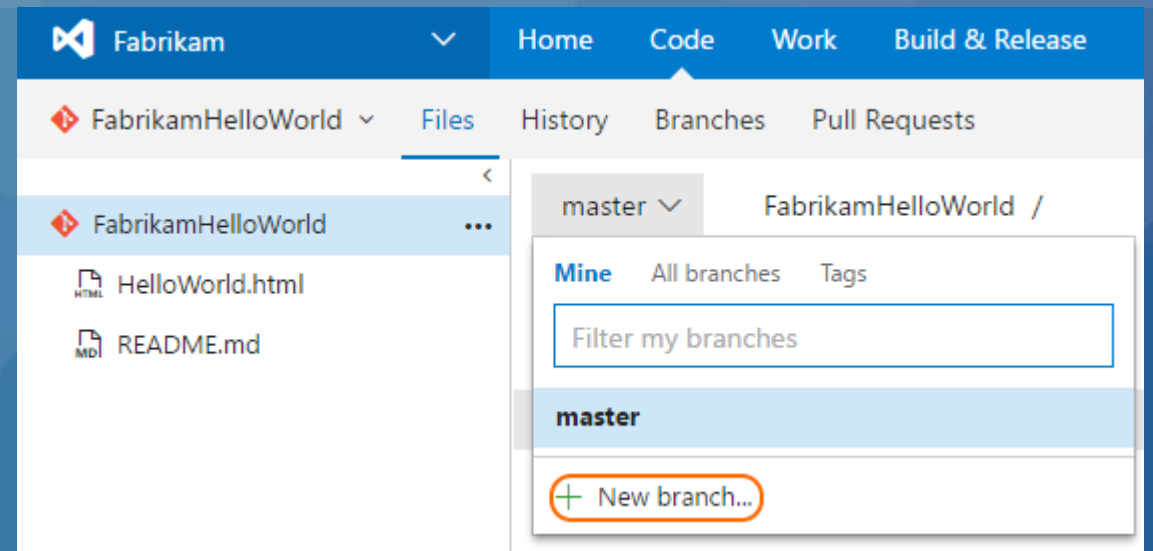
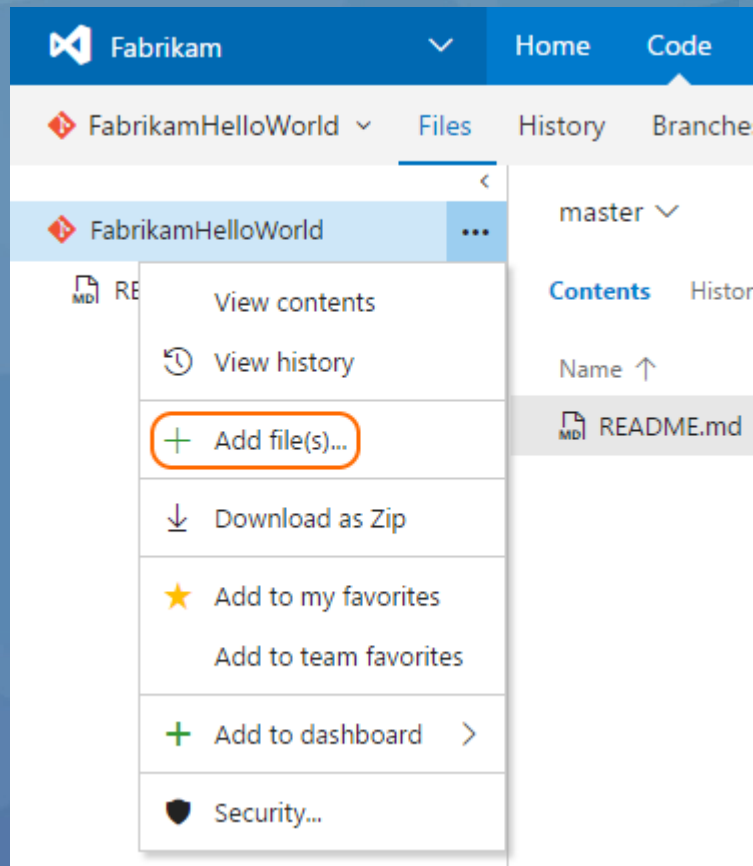
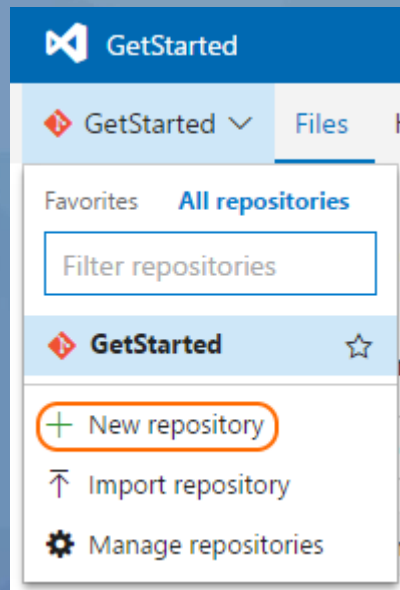
# Repository

## Objectives Repository

- Create TFS Repository
- Set up a Continuous Integration (CI) Trigger
- Execute CI
- Tasks for Builds Based on Branches Built
- Keep Code Quality High

# Repository

## Create a Repository TFS

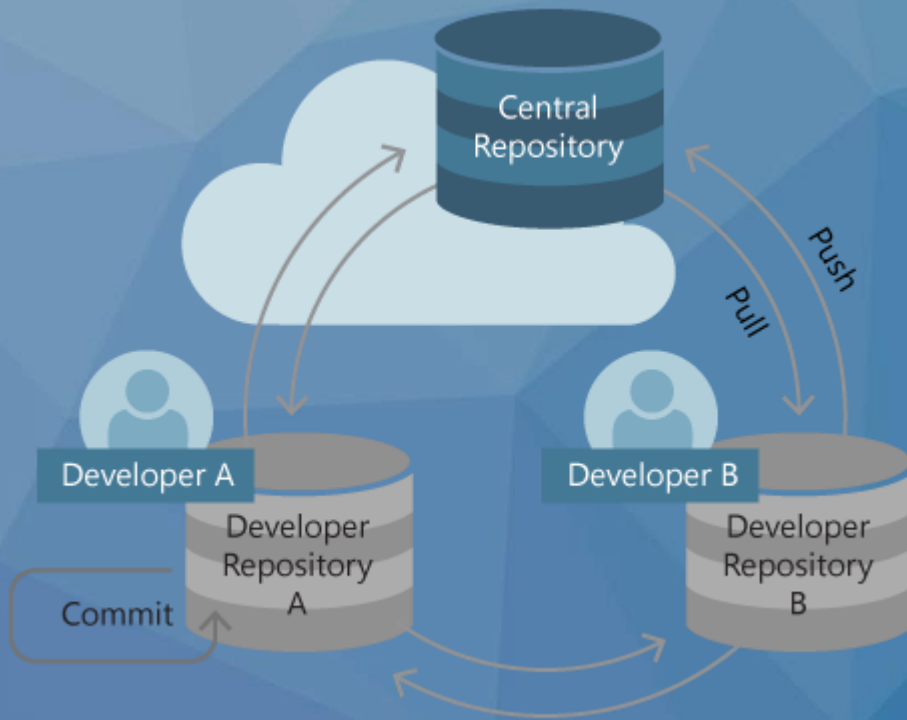


You updated [update-greeting](#) just now — [Create a pull request](#) (highlighted with an orange circle)



# Repository

## Add a Repository



Local:

```
> git init  
> git add --all  
> git commit -m "Initial commit"  
> git remote add origin https://...
```

Remote:

```
> git clone https://tfs...
```

# Repository

## Setup Trigger

## Build and Release

- Build Definition
- - Triggers

🏠 ... > DevOpsDemos-ASP.NET Core (.NET Framework)-CI

Tasks Variables **Triggers** Options Retention History


### Continuous Integration

Build every change to matching branches

Disable this trigger

☒ Enabled

#### Repositories

 dotnetcore-sample ^

Build when any branch changes

☐ Batch changes while a build is in progress

#### Branch Filters

Type	Branch specification	
Include	master	🗑
Include	features/*	🗑

+ Add

#### Path Filters

+ Add

# Repository

## Add Tasks Publish

Publish Build Artifacts ⓘ

[Link settings](#) [Remove](#)

Version 1.\* ▼

Display name \*

Path to Publish \* ⓘ  
 ...

Artifact Name \* ⓘ  

drop

Artifact Type \* ⓘ  

Server ▼

Control Options ^  

☒ Enabled

☐ Continue on error

Timeout \* ⓘ

Run this task ⓘ  

Custom conditions ▼

Custom condition ⓘ





[AKA.MS/MARK/PSGIT/VOICE](https://aka.ms/mark/psgit/voice)

Mark Warneke

[mark.warneke@microsoft.com](mailto:mark.warneke@microsoft.com)



@mark\_mit\_k\_  
aka.ms/mark

# Resources

- <https://www.visualstudio.com/learn/what-is-version-control/>
- <https://docs.microsoft.com/en-us/vsts/git/tutorial/gitworkflow>
- <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

NEXT: scripts

# Scripts

CommandLine

-----

```
Set-Location -LiteralPath 'C:\dev\tmp\git'
```

```
git init
```

```
Get-ChildItem -Hidden
```

```
echo "test" > filename.txt
```

```
git status
```

```
git add .
```

```
git commit -am "initial commit"
```

```
echo "test2" > filename.txt
```

```
git diff
```

```
cat .\filename.txt
```

```
git log
```

```
git checkout -b "develop"
```

```
git status
```

```
echo "test branch" > branch.txt
```

```
git add .
```

```
git commit -am 'initial branch'
```

```
git status
```

```
Get-ChildItem
```

```
git checkout master
```

```
Get-ChildItem
```

```
git status
```

```
git merge develop
```

```
git status
```

```
Get-ChildItem
```

```
git log
```



