Microsoft

WorkshopPLUS: Azure DevOps Essentials – Git

Overview

Microsoft Services

Overview

- Intros
  - Name, Role
  - Experience with version control
  - Goals for the training
- Agenda
  - Intro to git
  - Setting up a repository
  - Using git
  - Branching and merging
  - Using git with Visual Studio

Module Overview

- Version Control Systems
- Trends
- Tools
- Git Services

Git

- Difference between Git and GitHub

❖ **Git** is a version control system, a tool to manage your source code history.

❖ **GitHub** is a hosting service for Git repositories.

History of Version Control

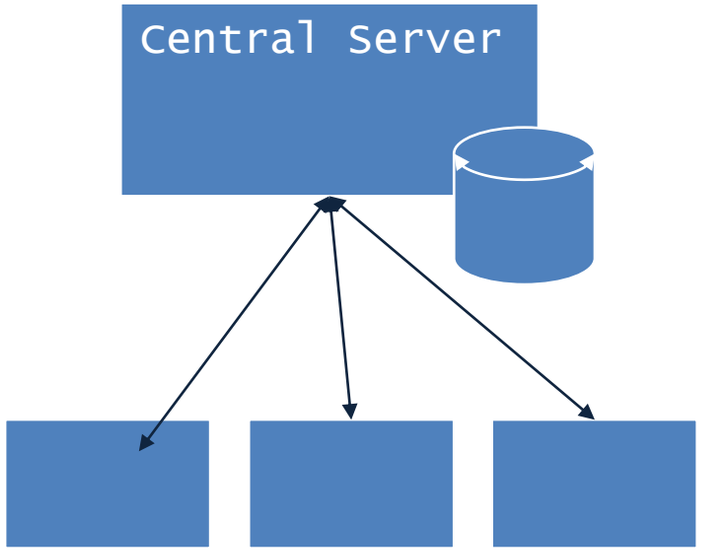
- Local
- Centralized (CVCS)
- Distributed (DVCS)

Centralized

- Shared repository is used to maintain an authoritative copy of the source code.
- Individuals have local copies of the files which they modify and send back to the server to share with others.

Microsoft Visual SourceSafe 6.0

TFS



Concurrent Versions System (CVS)

SUBVERSION

Distributed Version Control System

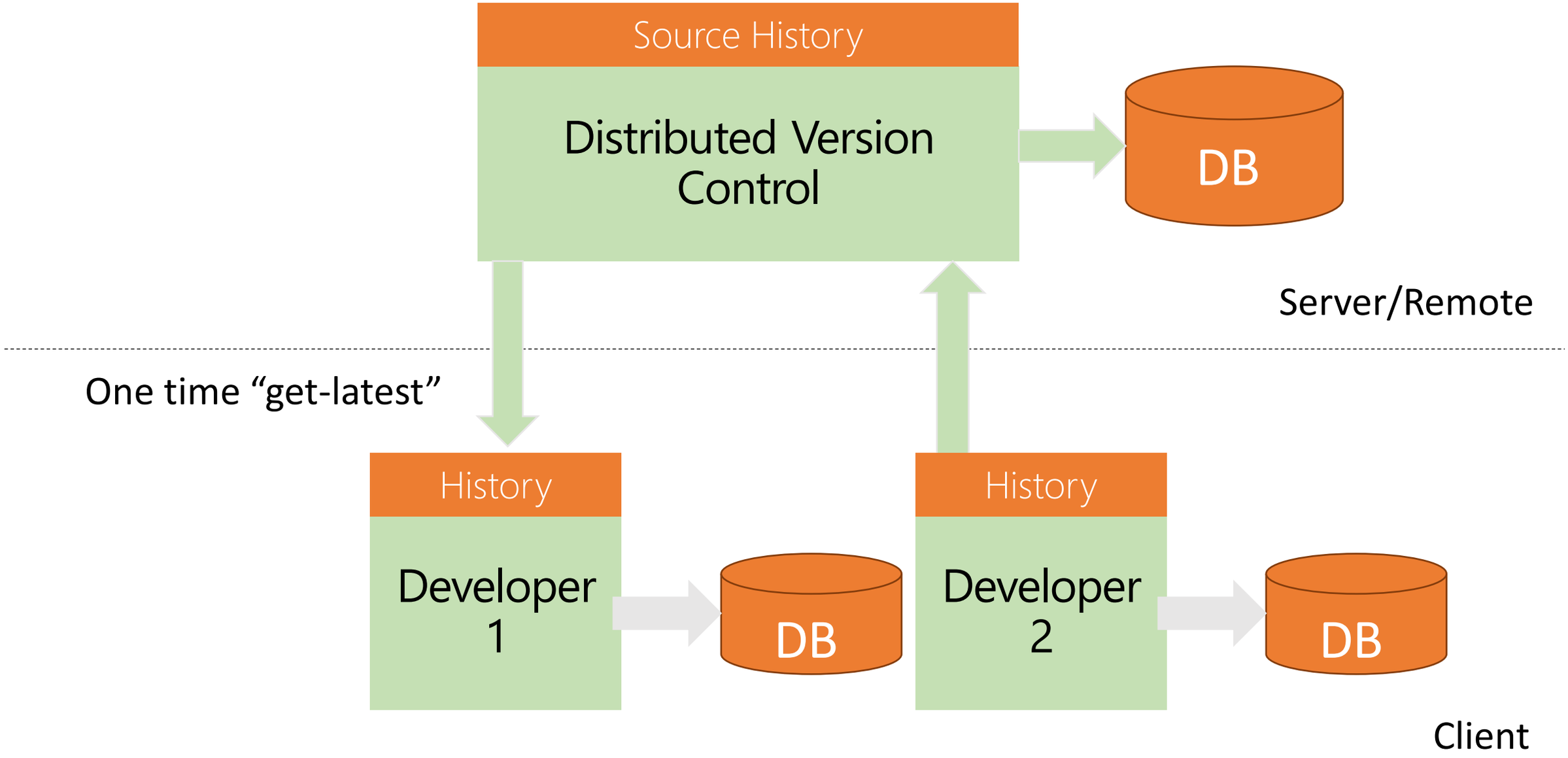
- Developers “clones” a copy of a repository and has the full history of the project locally.
- Developers can work offline and disconnected from the central repo until ready to commit changes.

BitKeeper
Scalable Version Control

mercurial

darcs

Distributed Version Control Systems



Modern Source-Control Approaches

	Strengths	Best for	Disadvantages
Centralized Version Control (CVCS)	• Fine level permission control • Allows usage monitoring • Easy setup	• Large integrated codebases, long history, many binary files • Control and auditability down to the file level	• Single point of failure • Remote commits are slow • Merging can be difficult • Offline is a challenge. Committing / viewing history requires repo access
Distributed Version Control (DVCS)	• Fast offline experience. Complete repository with portable history • Pull Requests model for reviewing code • Branching and merging is easy and extremely fast	• Modular codebases • Open source projects • Highly distributed teams • On the go teams, everything can be done without Internet except push/pull	• Many large binary files could impact performance • Long history 100k+ refs could take a lot of time and disk space, performance hit • Learning curve to adopt

Microsoft Git Support

Azure DevOps git repos will work with any Git client

- Git command lines, XCode, Eclipse, IntelliJ Git support

Our core principle is about providing a good and interoperable Git capability

Windows and DevOps code bases are on Git

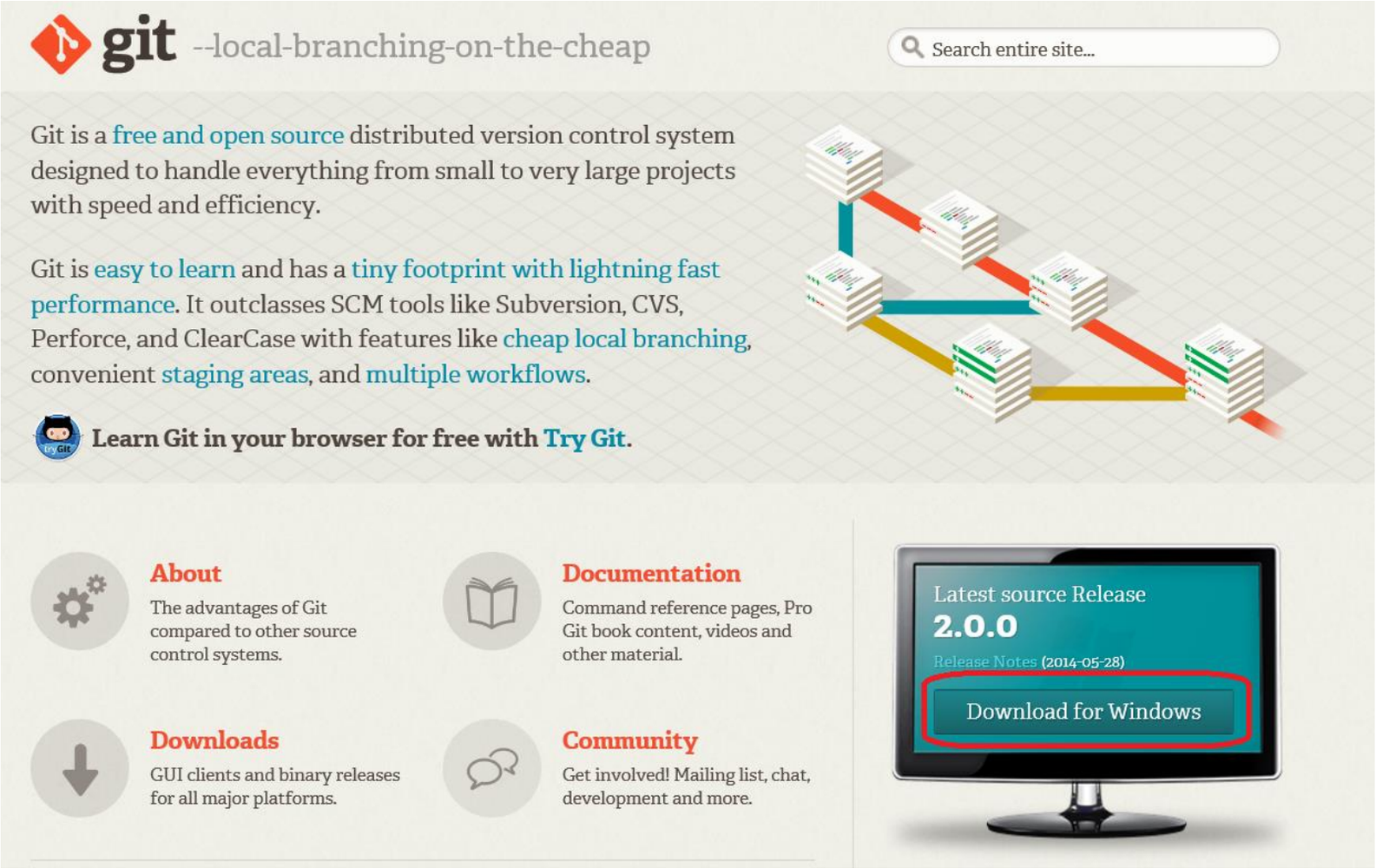
Lesson 4: Git Tools

Git for Windows

Git Clients

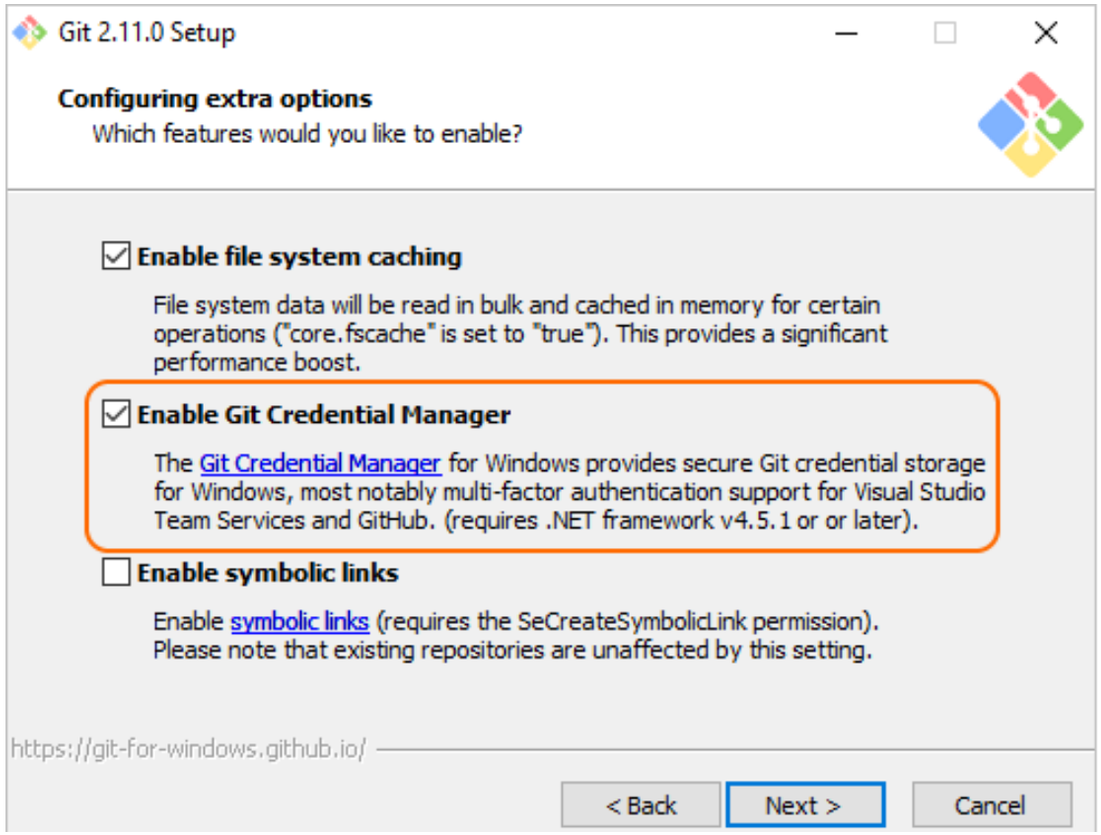
Git for Windows

- http://git-scm.com
- http://gitforwindows.org



Git Credential Manager

Git Credential Managers simplify authentication with your Azure DevOps Services



Git Clients

- https://git-scm.com/download/gui/windows

GUI Clients

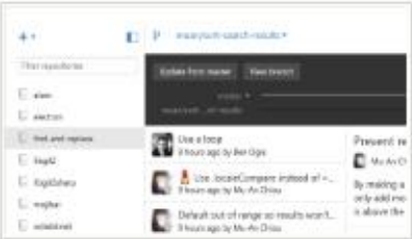
Git comes with built-in GUI tools for committing (git-gui) and browsing (gitk), but there are several third-party tools for users looking for platform-specific experience.

If you want to add another GUI tool to this list, just follow the instructions.

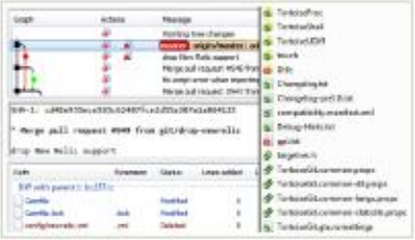
All | Windows | Mac | Linux | Android | iOS

22 Windows GUIs are shown below ↓

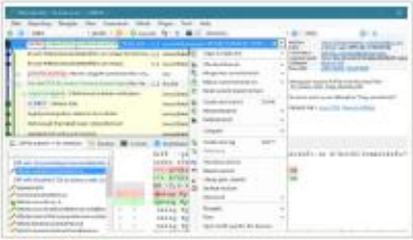
SourceTree
Platforms: Mac, Windows
Price: Free
License: Proprietary



GitHub Desktop
Platforms: Mac, Windows
Price: Free
License: MIT

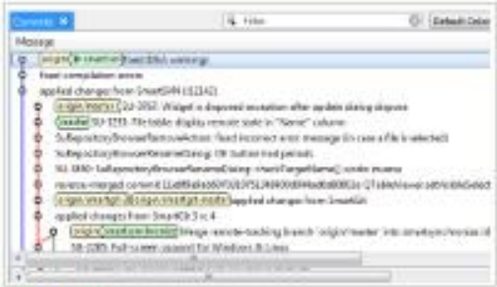


TortoiseGit
Platforms: Windows
Price: Free
License: GNU GPL



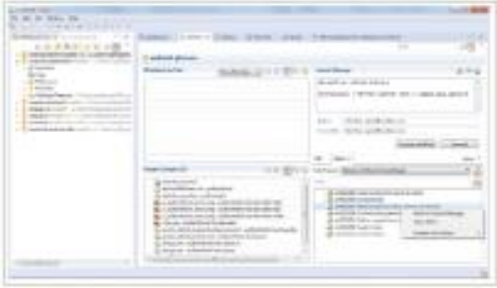
Git Extensions
Platforms: Linux, Mac, Windows
Price: Free
License: GNU GPL

GitKraken
Platforms: Linux, Mac, Windows
Price: Free for non-commercial use
License: Proprietary



SmartGit
Platforms: Linux, Mac, Windows
Price: $79/user / Free for non-commercial use
License: Proprietary

Tower
Platforms: Mac, Windows
Price: $79/user (Free 30 day trial)
License: Proprietary



GitEye
Platforms: Linux, Mac, Windows
Price: Free
License: Proprietary

Git CLI

git [subcommand] -h	Get help on the usage for git or a specific sub-command
git <subcommand>	Execute a git sub-command

Init – Creating new repositories

git init	Convert current directory into a repository
git init <directory>	Create empty repo is the target directory

Clone – Downloading existing repositories

```
git clone
https://s.com/repo.git
```

Create a new local folder named “repo” and copy the entire history of the repository from the server to the repo folder

```
git clone
https://s.com/repo.git
my-repo
```

Create a new local folder named “my-repo” and copy the entire history of the repository from the server to the “my-repo” folder

Status - Example

```
C:\gitbasics>echo Feature A >>newcode.cs

C:\gitbasics>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:   newcode.cs

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

C:\gitbasics>
```

Stage – Common Options

git add <file>	Stage all changes in <file> for next commit
git add <directory>	Stage changes in <directory> for next commit
git add .	Stage all changes in current directory for next commit

Stage – Unstaging

git rm --cached <file>	Unstage changes in <file>
git checkout -- <file>	Revert any changes made locally

Commit – Common Options

git commit	Will launch text editor, enter commit message, saved, close editor and commit will complete
git commit -m “message”	Immediately creates commit with the message

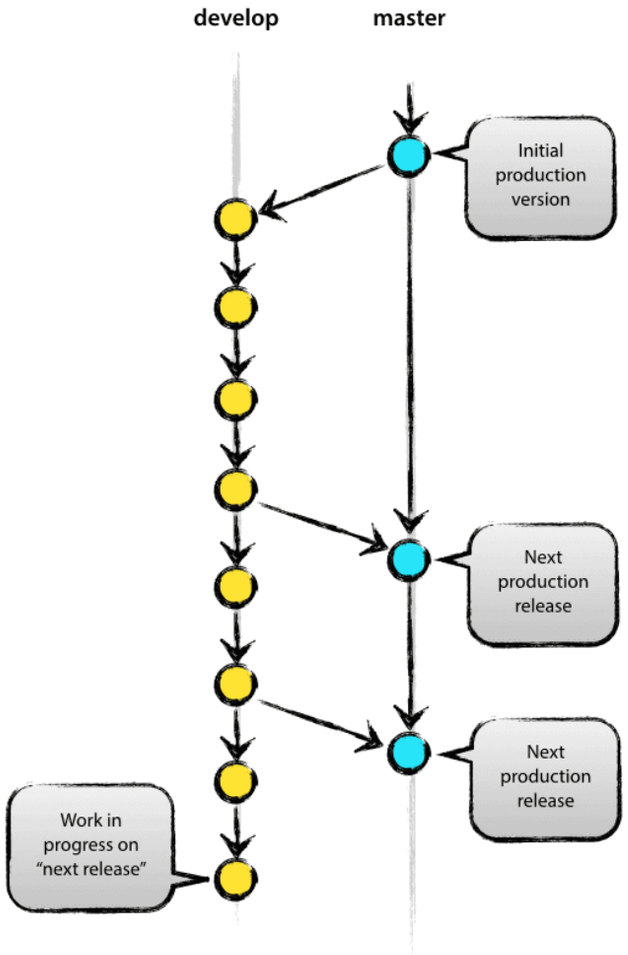
Branching – Basic Commands

Creating a branch

1- git branch <name-of-your-branch>
2- git checkout –b <name-of-your-branch>

Remove a branch

1- git branch -d <name-of-your-branch>
2- git checkout –D <name-of-your-branch>
3- git push origin –d <name-of-your-branch>



Module Summary

- Git is the most popular version control system today
- Git is an open source project and supported by many companies and individuals
- Wide support for Git in various OS's and many dev tools
- Amazon.com: Pro Git eBook : Chacon, Scott, Ben Straub: Kindle Store

Lab: Getting Started

Exercise 1: Install Git
Exercise 2: Stage and Commit changes
Exercise 3: Commit changes without staging