

The slide has a dark grey header bar. On the left side of the header is a decorative icon consisting of three hexagons in orange, teal, and green. The main title 'Why version control?' is centered in the header. The slide content is organized into two main sections: 'Scenario 1:' and 'Has this ever happened to you?'. 'Scenario 1:' is preceded by a diamond-shaped bullet point. The list under 'Scenario 1:' includes the following items:

- Your program is working
- You change “just one thing”
- Your program breaks
- You change it back
- Your program is still broken--*why?*

'Has this ever happened to you?' is preceded by a diamond-shaped bullet point.

In the bottom right corner of the slide, there is a small graphic of three hexagons in light blue, teal, and light green, with the number '2' inside the bottom-right hexagon.



## Why version control? (part 2)

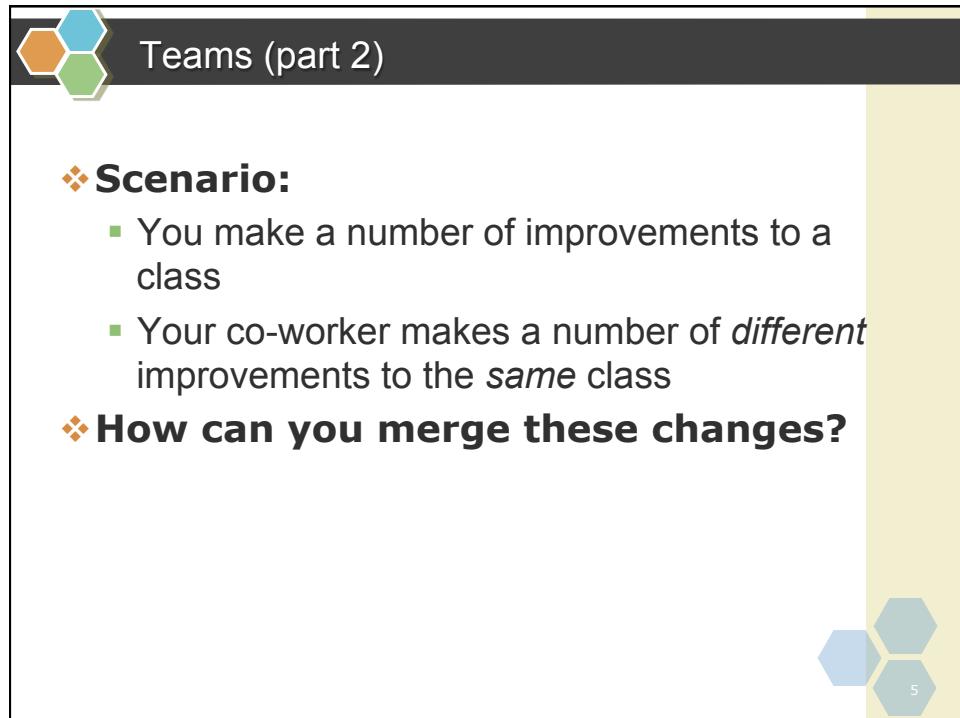
- ❖ **Your program worked well enough yesterday**
- ❖ **You made a lot of improvements last night...**
  - ...but you haven't gotten them to work yet
- ❖ **You need to turn in your program now**
- ❖ **Has this ever happened to you?**



## Version control for teams

- ❖ **Scenario:**
  - You change one part of a program--it works
  - Your co-worker changes another part--it works
  - You put them together--it doesn't work
  - Some change in one part must have broken something in the other part
  - What were all the changes?





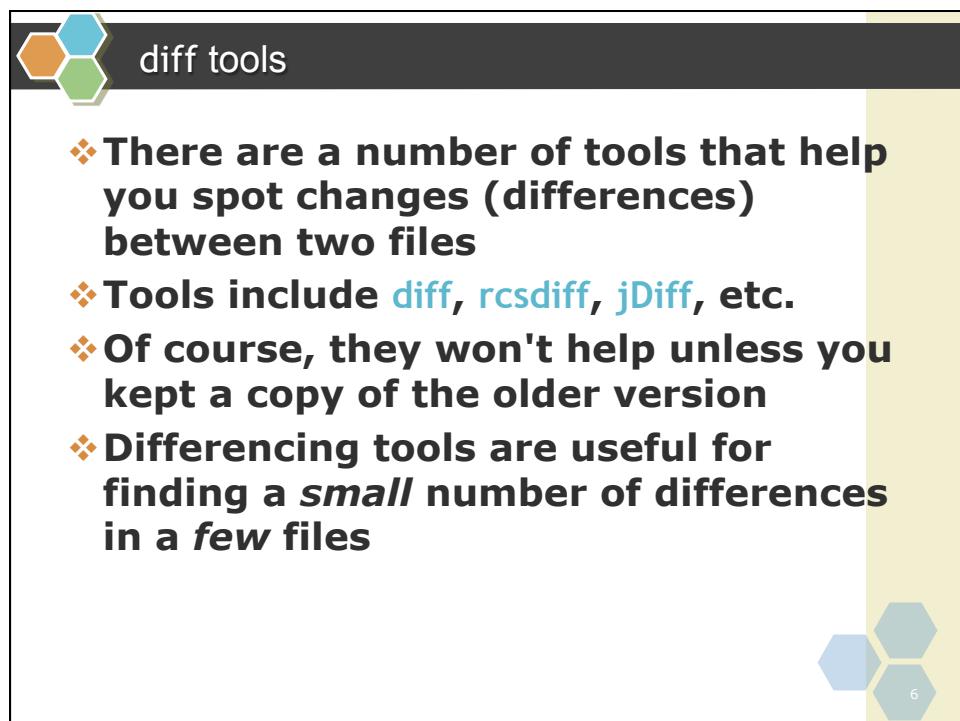
## Teams (part 2)

❖ **Scenario:**

- You make a number of improvements to a class
- Your co-worker makes a number of *different* improvements to the *same* class

❖ **How can you merge these changes?**

5



## diff tools

❖ **There are a number of tools that help you spot changes (differences) between two files**

❖ **Tools include `diff`, `rcsdiff`, `jDiff`, etc.**

❖ **Of course, they won't help unless you kept a copy of the older version**

❖ **Differencing tools are useful for finding a *small* number of differences in a *few* files**

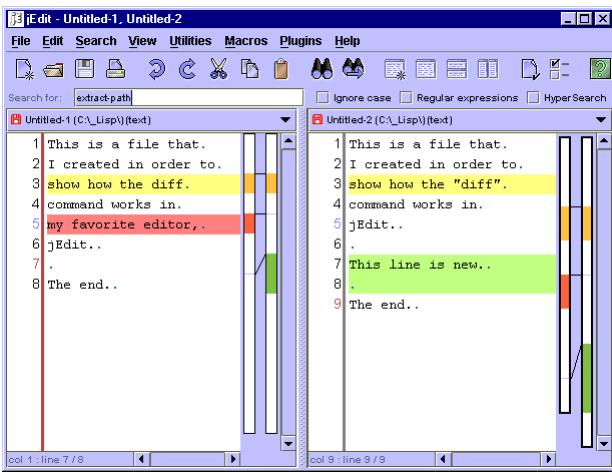
6

 jDiff

- ❖ **jDiff is a plugin for the jEdit editor**
- ❖ **Advantages:**
  - Everything is color coded
  - Uses synchronized scrolling
  - It's inside an editor--you can make changes directly
- ❖ **Disadvantages:**
  - Not stand-alone, but must be used within **jDiff**
  - Just a diff tool, not a complete solution

 7

 jDiff



 8



## Version control systems

❖ **A version control system (often called a source code control system) does these things:**

- Keeps multiple (older and newer) versions of everything (not just source code)
- Requests comments regarding every change
- Allows “check in” and “check out” of files so you know which files someone else is working on
- Displays differences between versions




## Vocabulary

❖ Repository (source control repository)

- A server that stores the files (documents)
- Keeps a change log

❖ Revision, Version

- Individual version (state) of a document that is a result of multiple changes

❖ Check-Out, Clone

- Retrieves a working copy of the files from a remote repository into a local directory
- It is possible to lock the files



 Vocabulary

- ❖ Change
  - A modification to a local file (document) that is under version control
- ❖ Change Set / Change List
  - A set of changes to multiple files that are going to be committed at the same time
- ❖ Commit, Check-In
  - Submits the changes made from the local working copy to the repository
  - Automatically creates a new version
  - Conflicts may occur!



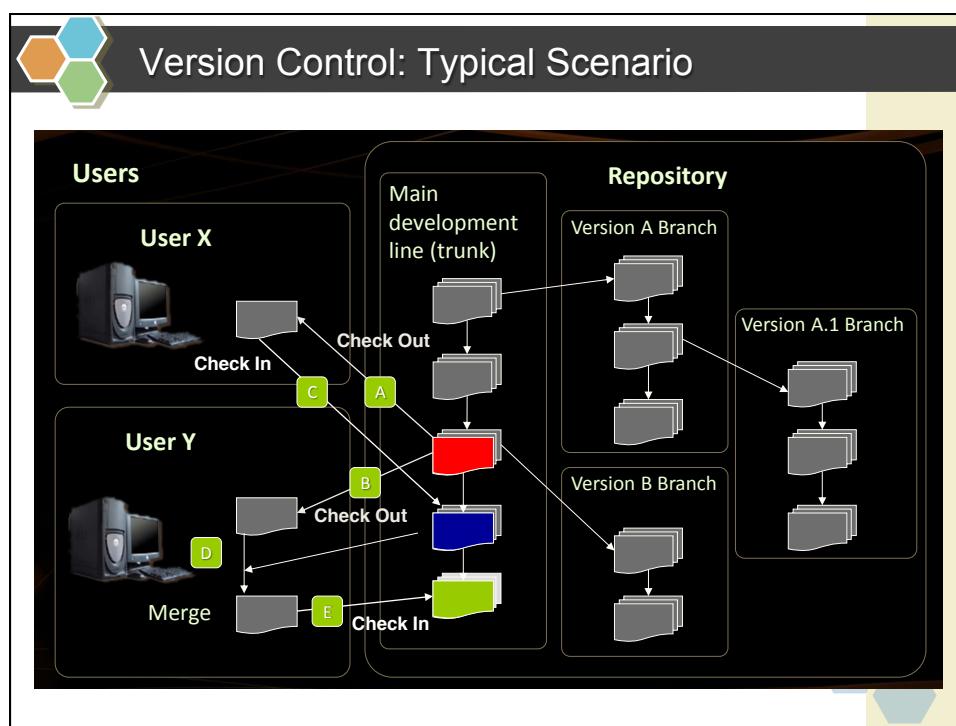
 Vocabulary

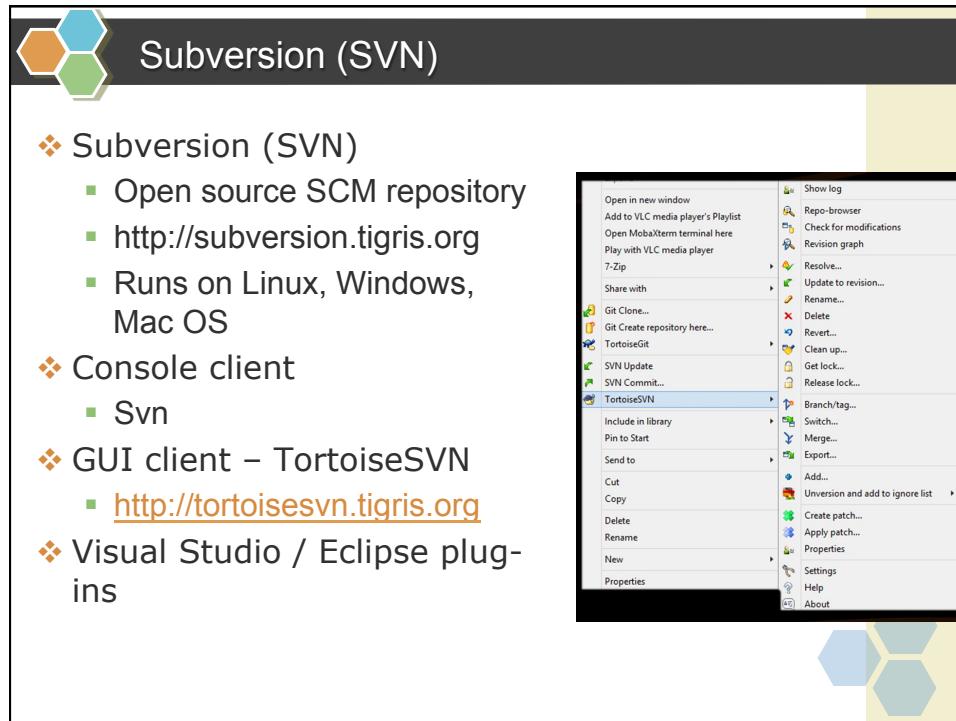
- ❖ Conflict
  - The simultaneous change to a certain file by multiple users
  - Can be solved automatically and manually
- ❖ Update, Get Latest Version, Fetch / Pull
  - Download the latest version of the files from the repository to a local working directory + merge conflicting files
- ❖ Undo Check-Out, Revert / Undo Changes
  - Cancels the local changes
  - Restores their state from the repository



## Vocabulary

- ❖ Merge
  - Combines the changes to a file changed locally and simultaneously in the repository
  - Can be automated in most cases
- ❖ Label / Tag
  - Labels mark with a name a group of files in a given version
  - For example a release
- ❖ Branch / Branching
  - Division of the repositories in a number of separate workflows

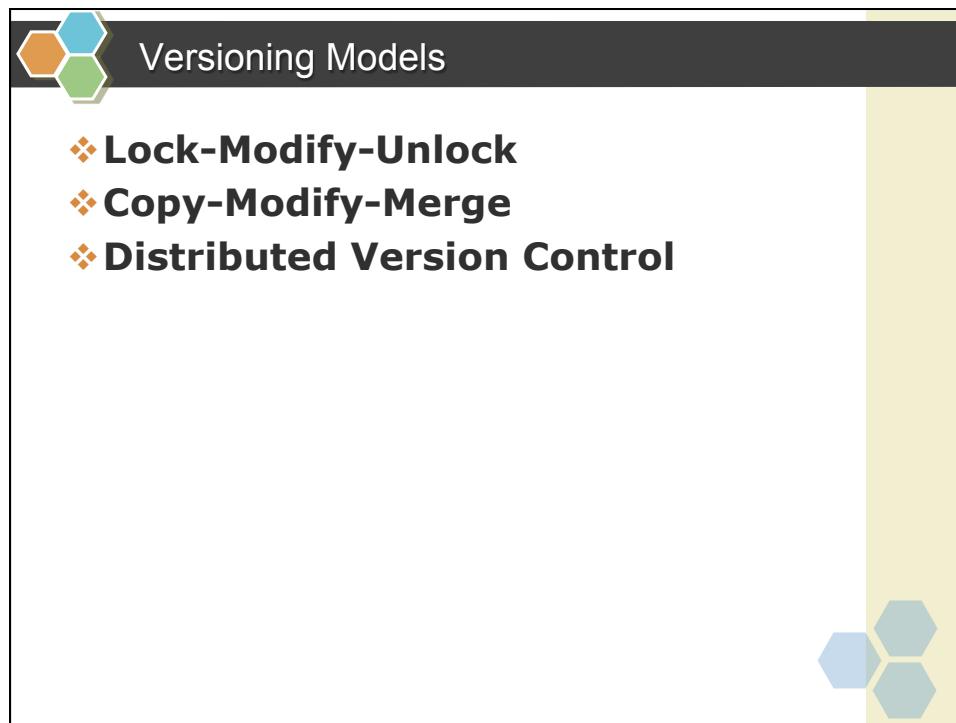




The screenshot shows a Windows desktop environment. In the center, there is a context menu open over a file or folder. The menu is titled "TortoiseSVN" and contains various options such as "Open in new window", "Share with", "Send to", "Cut", "Copy", "Delete", "Rename", "New", and "Properties". To the right of the menu, there is a small graphic element consisting of three hexagons in orange, blue, and green.

## Subversion (SVN)

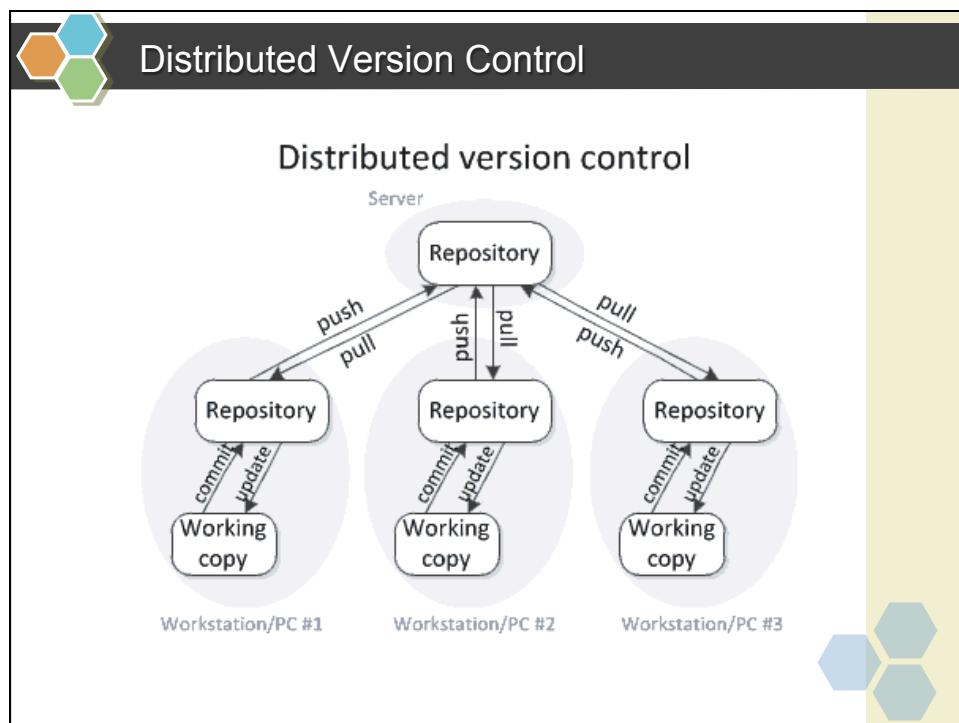
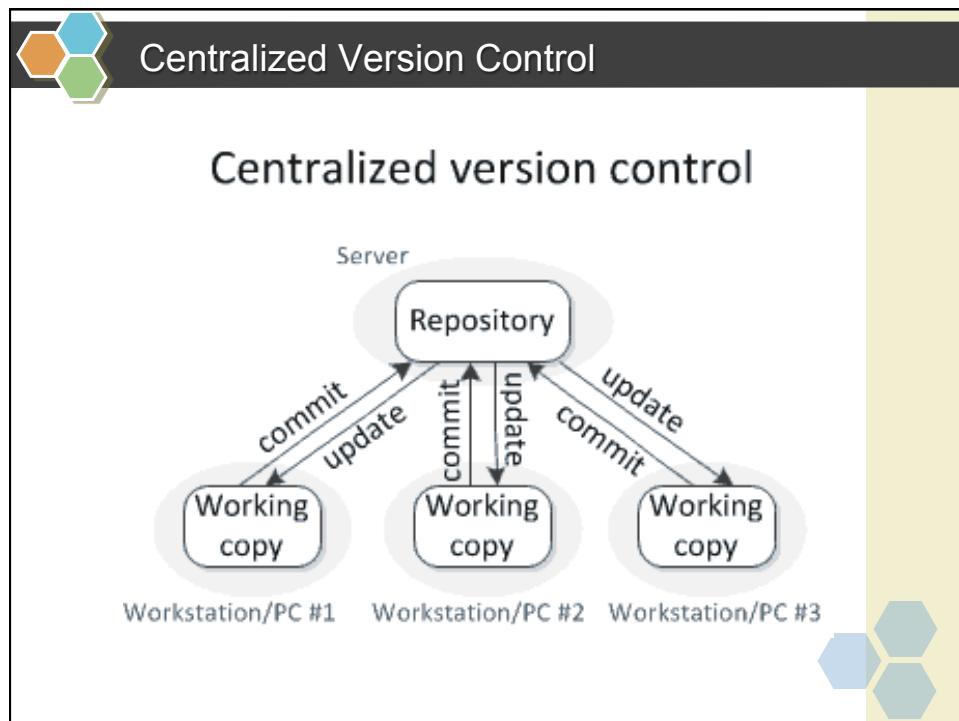
- ❖ Subversion (SVN)
  - Open source SCM repository
  - <http://subversion.tigris.org>
  - Runs on Linux, Windows, Mac OS
- ❖ Console client
  - Svn
- ❖ GUI client – TortoiseSVN
  - <http://tortois svn.tigris.org>
- ❖ Visual Studio / Eclipse plug-ins



The screenshot shows a Windows desktop environment. On the left, there is a list of versioning models. The items are listed in bold text with orange diamond icons to the left of them. The list includes "Lock-Modify-Unlock", "Copy-Modify-Merge", and "Distributed Version Control". To the right of the list, there is a vertical yellow bar and a small graphic element consisting of three hexagons in blue and grey.

## Versioning Models

- ❖ Lock-Modify-Unlock
- ❖ Copy-Modify-Merge
- ❖ Distributed Version Control



### The Lock-Modify-Unlock Model

Andy and Bobby check-out file A.

The check-out is done without locking. They just get a local copy.

The diagram illustrates the initial state of the Lock-Modify-Unlock model. At the top center is a cylinder labeled "Repository". Inside it, a green rectangular block is labeled "A". Two arrows point from this block to two separate computer workstations below. The workstation on the left is associated with a male user named "Andy" and has a green rectangle labeled "A" on its screen. The workstation on the right is associated with a female user named "Bobby" and also has a green rectangle labeled "A" on its screen. The word "Check-out" is written above each arrow pointing to the workstations.

### The Lock-Modify-Unlock Model

Andy locks file A and begins modifying it.

The diagram shows the transition to the second stage of the model. The central "Repository" cylinder still contains the "A" file. However, the local copy on Andy's workstation now features a yellow padlock icon next to the "A" label. The word "Lock" is written above the padlock. The local copy on Bobby's workstation remains unchanged, showing only the "A" label. The label "(Local Edit)" is placed near the Andy workstation's local copy.

### The Lock-Modify-Unlock Model

Bobby tries to lock the file too, but she can't.  
Bobby waits for Andy to finish and unlock the file.

The diagram illustrates the Lock-Modify-Unlock model. A central cylinder labeled "Repository" contains a green block labeled "A". A yellow padlock is attached to block "A". To the left, a man named "Andy" is shown working at a computer. To the right, a woman named "Bobby" is also working at a computer. A stopwatch icon with an arrow pointing to it indicates a "Wait" state. A curved arrow points from the stopwatch to the padlock, signifying that Bobby is waiting for Andy to release the lock. Both Andy and Bobby have green labels above them with their names.

### The Lock-Modify-Unlock Model (4)

Andy commits his changes and unlocks the file.

This diagram shows the progression of the Lock-Modify-Unlock model. It starts with the same setup as the previous diagram: a Repository cylinder with a block "A", a padlock, and two users, Andy and Bobby. In this stage, Andy is shown committing his changes, indicated by an arrow labeled "Commit" pointing to the Repository. The padlock is now removed from the block "A", allowing Bobby to proceed. The Repository cylinder now contains a green block labeled "Andy". Both Andy and Bobby are shown working at their computers again, with their respective labels above them.

### The Lock-Modify-Unlock Model (5)

Now Bobby can take the modified file and lock it.  
Bobby edits her local copy of the file.

Repository  
Andy

Lock

Andy

(Local Edit)

Bobby

### The Lock-Modify-Unlock Model (6)

Bobby finishes, commits her changes and unlocks the file.

Repository  
Andy Bobby

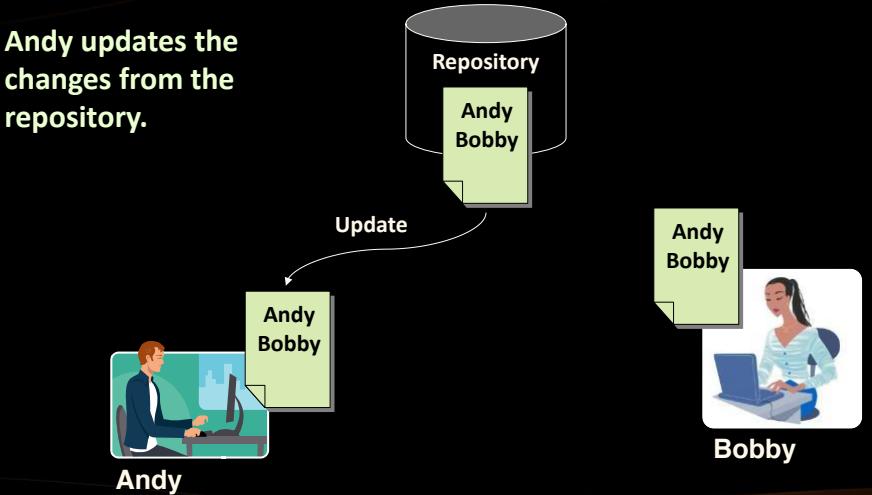
Commit

Andy

Bobby

 The Lock-Modify-Unlock Model (7)

Andy updates the changes from the repository.

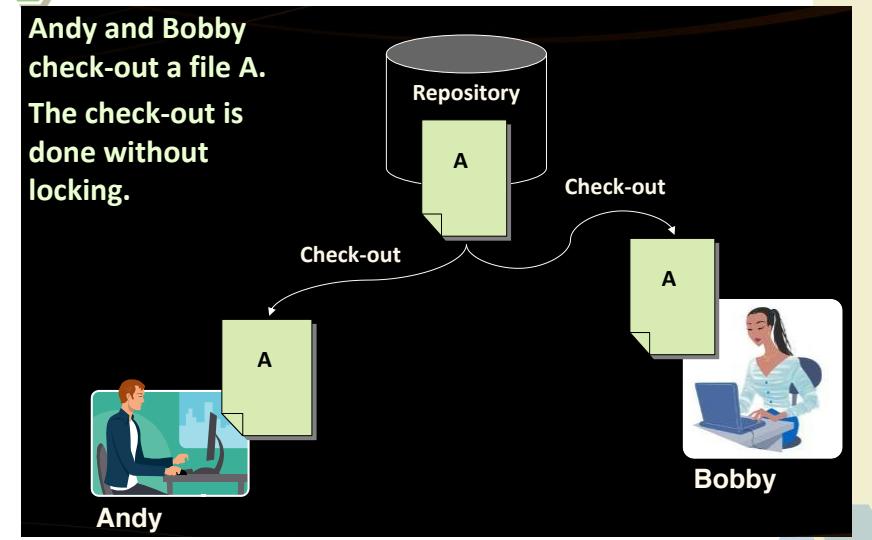


Andy

Bobby

 The Copy-Modify-Merge Model (1)

Andy and Bobby check-out a file A.  
The check-out is done without locking.



Andy

Bobby

### The Copy-Modify-Merge Model (2)

Both of them edit the local copies of the file (in the same time).

The diagram illustrates the Copy-Modify-Merge model at the second step. It features a central cylinder labeled "Repository" with a green rectangular block labeled "A" attached to its side. Two users, "Andy" and "Bobby", are shown working on their local copies. On the left, "Andy" is depicted at a desk with a computer monitor, with a green box labeled "Andy" above him and the text "(Local Edit)" below. On the right, "Bobby" is shown at a desk with a laptop, with a green box labeled "Bobby" above him and the text "(Local Edit)" below. Arrows point from each user's local copy towards the central repository.

### The Copy-Modify-Merge Model (3)

Bobby commits her changes to the repository.

The diagram illustrates the Copy-Modify-Merge model at the third step. It shows the "Repository" cylinder with a green rectangular block labeled "Bobby" attached to its side. A curved arrow labeled "Commit" points from "Bobby's" local copy towards the repository. "Andy" is still shown at his local workspace on the left, while "Bobby" is shown at her local workspace on the right, both with green boxes labeled "Bobby" above them and the text "(Local Edit)" below.

### The Copy-Modify-Merge Model (4)

Andy tries to commit his changes.  
A conflict occurs.

Commit  
**X**

(Local Conflict)

Andy

Bobby

### The Copy-Modify-Merge Model (5)

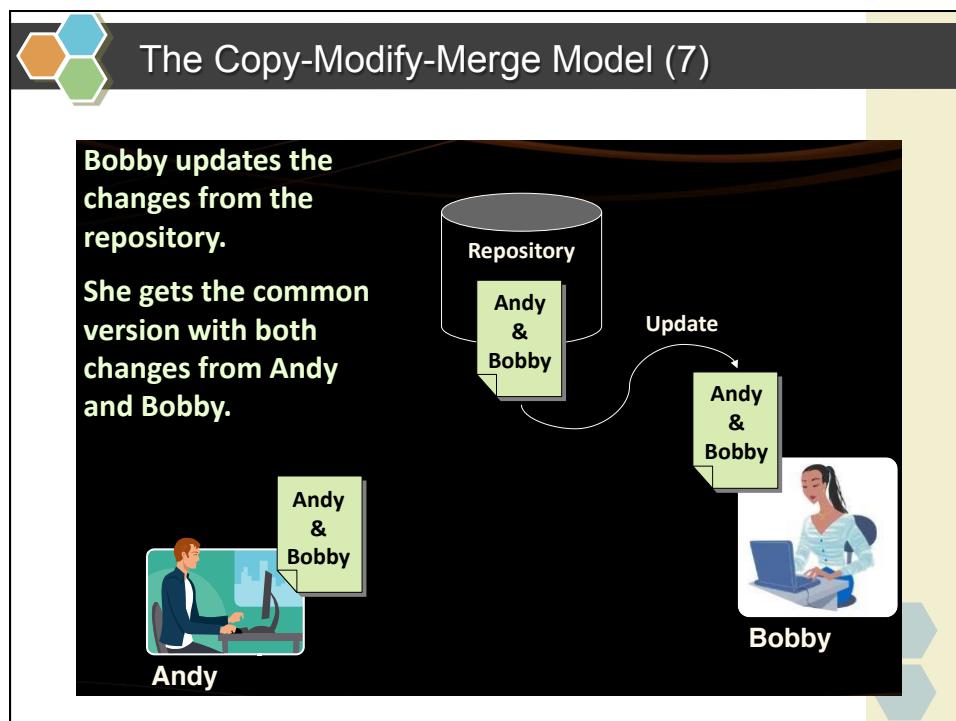
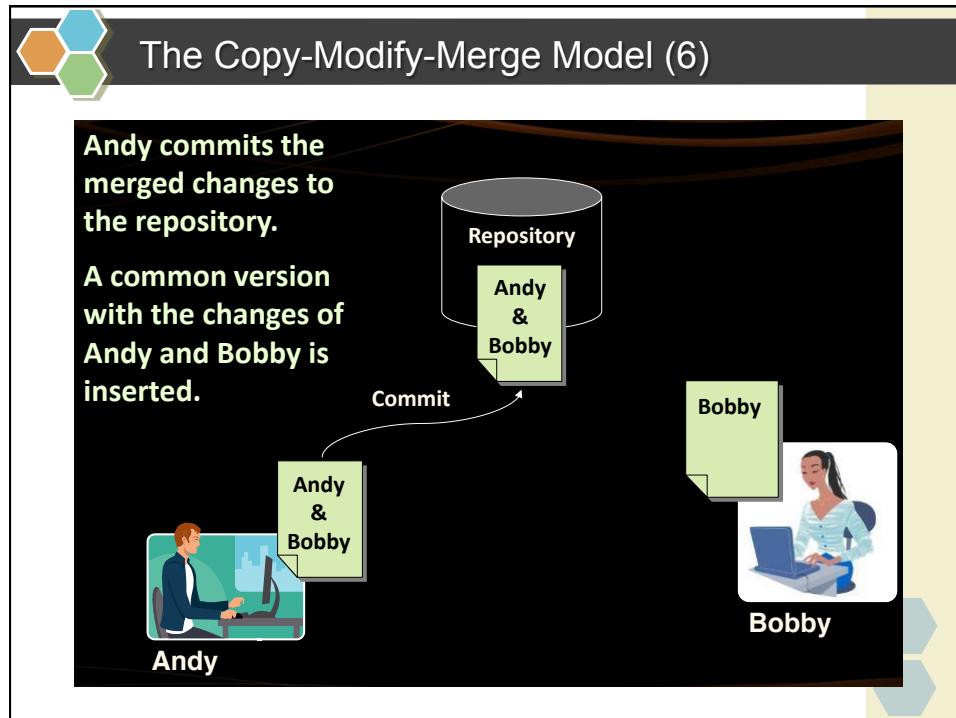
Andy updates his changes with the ones from the repository.  
The changes merge into his local copy.  
A merge conflict can occur.

Update  
(with merge)

(Local Merge)

Andy & Bobby

Bobby



## Distributed Version Control (1)

**Andy and Bobby** clone the remote repository locally.  
They both have the same files in their local repositories.

```

graph TD
    RR[Remote Repository Server] -- Clone --> LRAndy[Local Repository Andy]
    RR -- Clone --> LRBobby[Local Repository Bobby]
    subgraph LR
        LRAndy --- A1[A]
        LRBobby --- A2[A]
    end
    subgraph LR
        LRAndy --- Andy[Andy]
        LRBobby --- Bobby[Bobby]
    end

```

The diagram illustrates the initial state of distributed version control. At the top is a cylinder labeled "Remote Repository (Server)" containing a green hexagon labeled "A". Two arrows labeled "Clone" point down to two separate cylinders labeled "Local Repository (Andy)" and "Local Repository (Bobby)". Each local repository cylinder contains its own green hexagon labeled "A". Below each cylinder is a small illustration of a person at a computer: "Andy" on the left and "Bobby" on the right.

## Distributed Version Control (2)

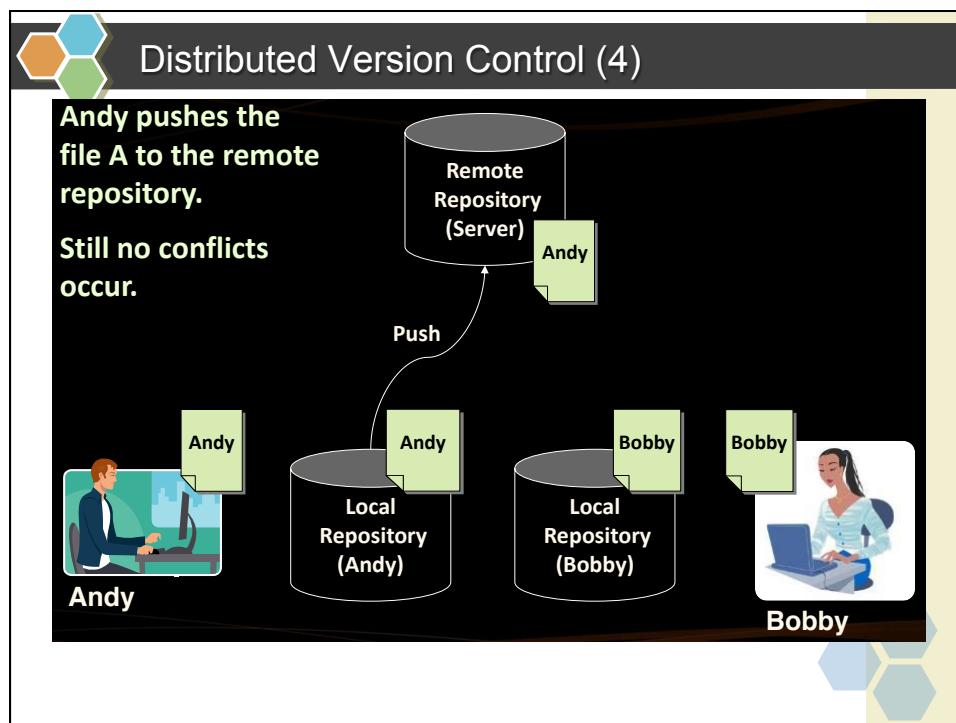
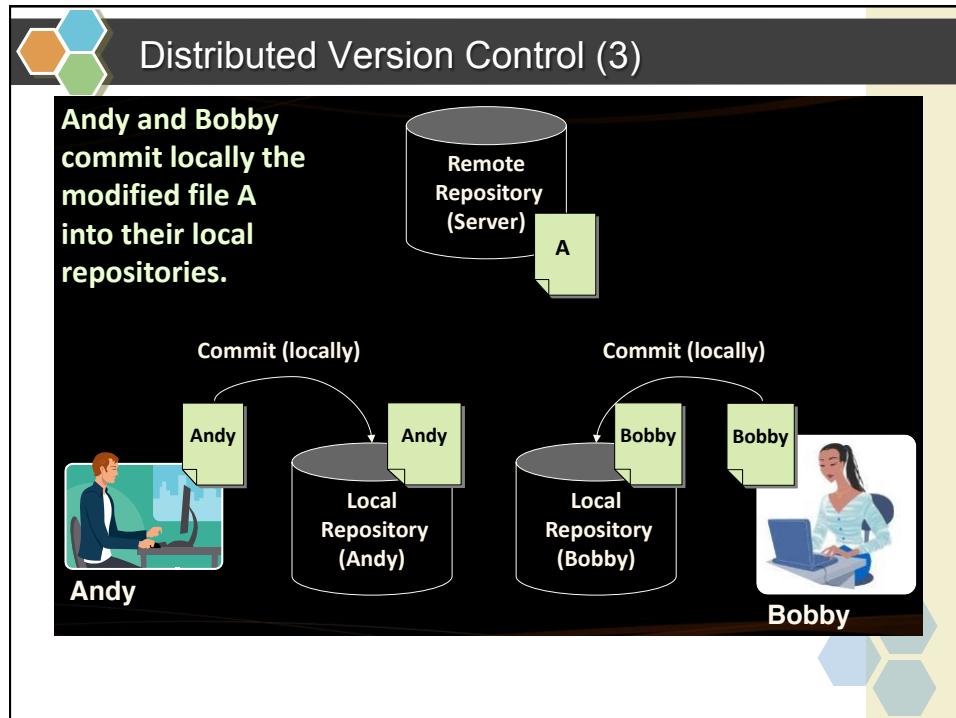
**Andy and Bobby** work locally on a certain file A.

```

graph TD
    RR[Remote Repository Server] -- Clone --> LRAndy[Local Repository Andy]
    RR -- Clone --> LRBobby[Local Repository Bobby]
    subgraph LR
        LRAndy --- A1[A]
        LRBobby --- A2[A]
    end
    subgraph LR
        LRAndy --- Andy[Andy]
        LRBobby --- Bobby[Bobby]
    end
    subgraph LR
        A1 --- LEAndy[(Local Edit) Andy]
        A2 --- LEBobby[(Local Edit) Bobby]
    end

```

The diagram shows the state after local edits. The "Remote Repository (Server)" still has a green hexagon "A". The "Local Repository (Andy)" cylinder now contains a green hexagon "A" with a yellow corner labeled "(Local Edit) Andy". The "Local Repository (Bobby)" cylinder also contains a green hexagon "A" with a yellow corner labeled "(Local Edit) Bobby". The illustrations of "Andy" and "Bobby" are shown working on their respective computers.



## Distributed Version Control (5)

**Bobby tries to push her changes.**

**A versioning conflict occurs.**

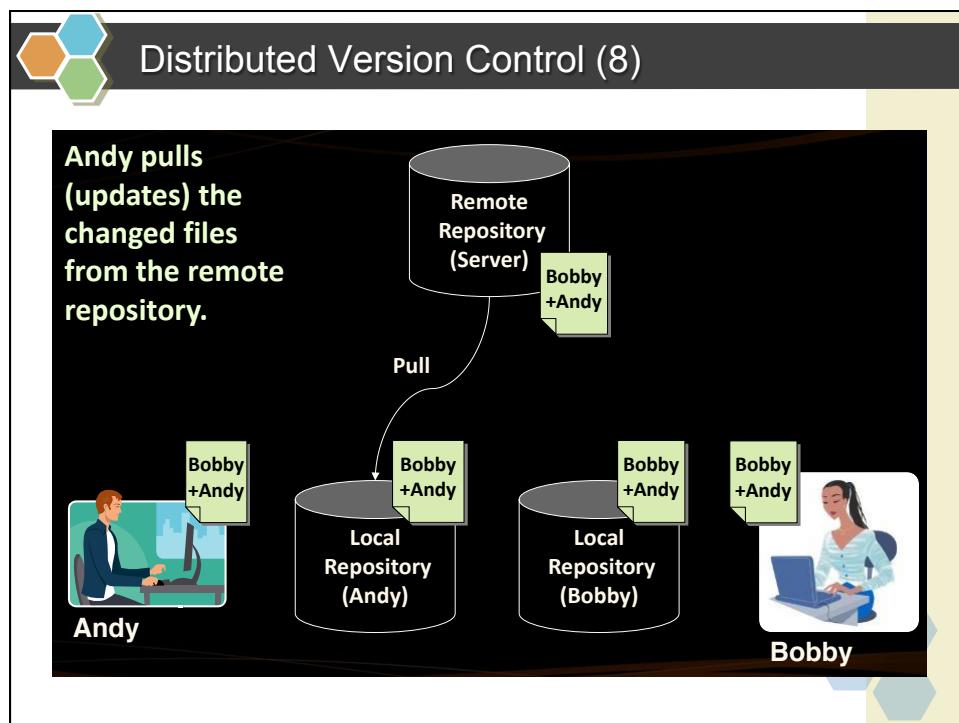
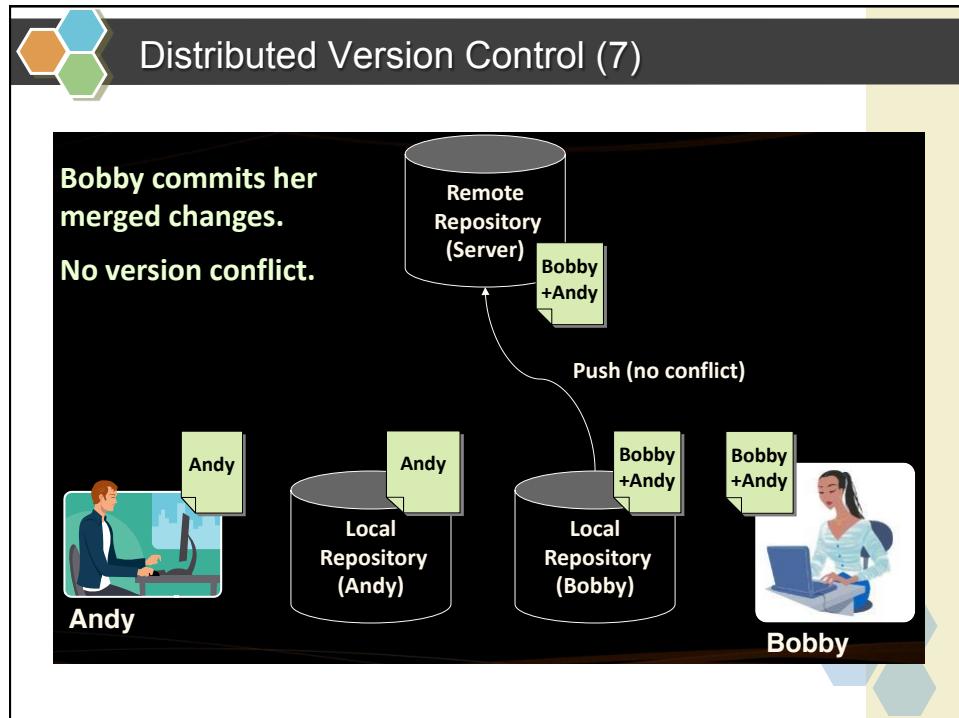
The diagram shows a central **Remote Repository (Server)** connected to two local repositories: **Local Repository (Andy)** and **Local Repository (Bobby)**. A user icon labeled **Andy** is associated with the Local Repository (Andy). A user icon labeled **Bobby** is associated with the Local Repository (Bobby). A red 'X' marks a path from the Local Repository (Bobby) to the Remote Repository, labeled **Push (conflict)**, indicating that Bobby's attempt to push changes to the central repository failed due to a conflict.

## Distributed Version Control (6)

**Bobby merges the her local files with the files from the remote repository.**

**Conflicts are locally resolved.**

The diagram shows a central **Remote Repository (Server)** connected to two local repositories: **Local Repository (Andy)** and **Local Repository (Bobby)**. A user icon labeled **Andy** is associated with the Local Repository (Andy). A user icon labeled **Bobby** is associated with the Local Repository (Bobby). A curved arrow labeled **Pull (Fetch + Merge)** points from the Remote Repository to the Local Repository (Bobby). The Local Repository (Bobby) contains a file labeled **Bobby + Andy**, indicating that Bobby has merged her local changes with the latest version from the remote repository. A blue hexagonal icon is visible at the bottom right of the slide.



## What is Git?



- ❖ Git
  - Distributed source-control system
  - Work with local and remote repositories
  - Git bash – command line interface for Git
  - Free, open-source
  - Has Windows version (msysGit)
    - <http://msysgit.github.io>
    - <https://www.atlassian.com/git/tutorials/setting-up-a-repository>

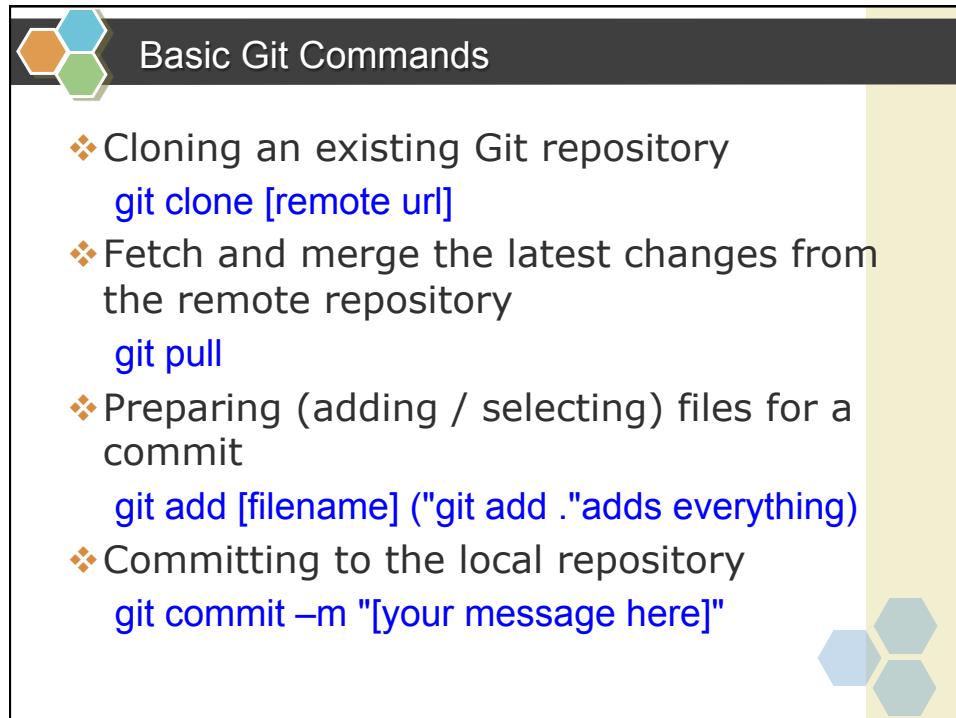


## Installing Git



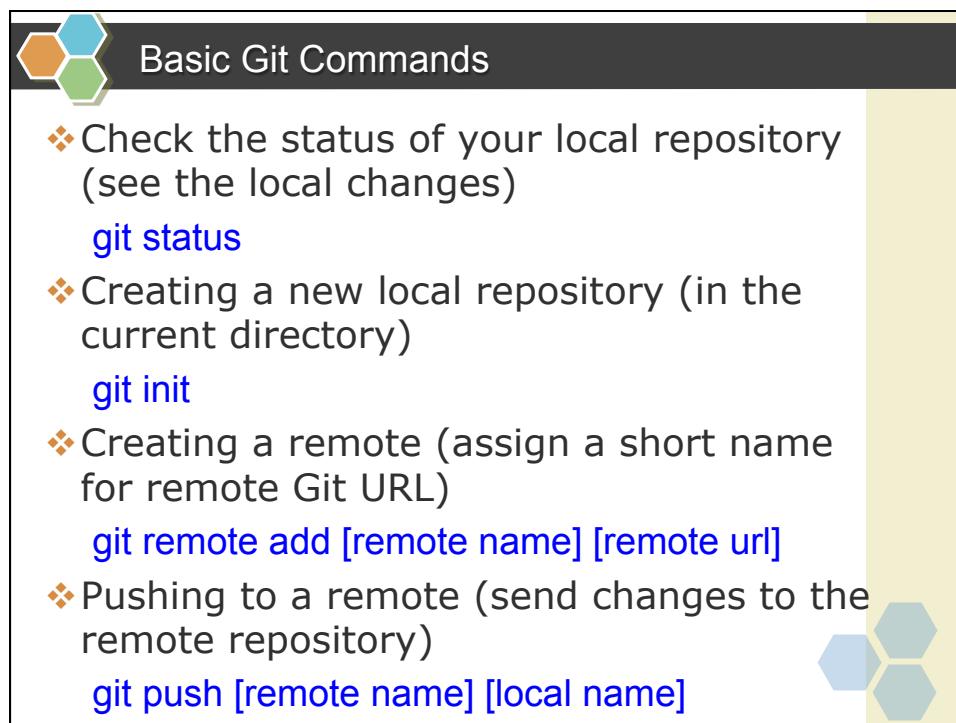
- ❖ msysGit Installation on Windows
  - Download Git for Windows from:  
<http://msysgit.github.io>
  - “Next, Next, Next” does the trick
  - Options to select (they should be selected by default)
    - “Use Git Bash only”
    - “Checkout Windows-style, commit Unix-style endings”
- ❖ Git installation on Linux:  
`sudo apt-get install git`





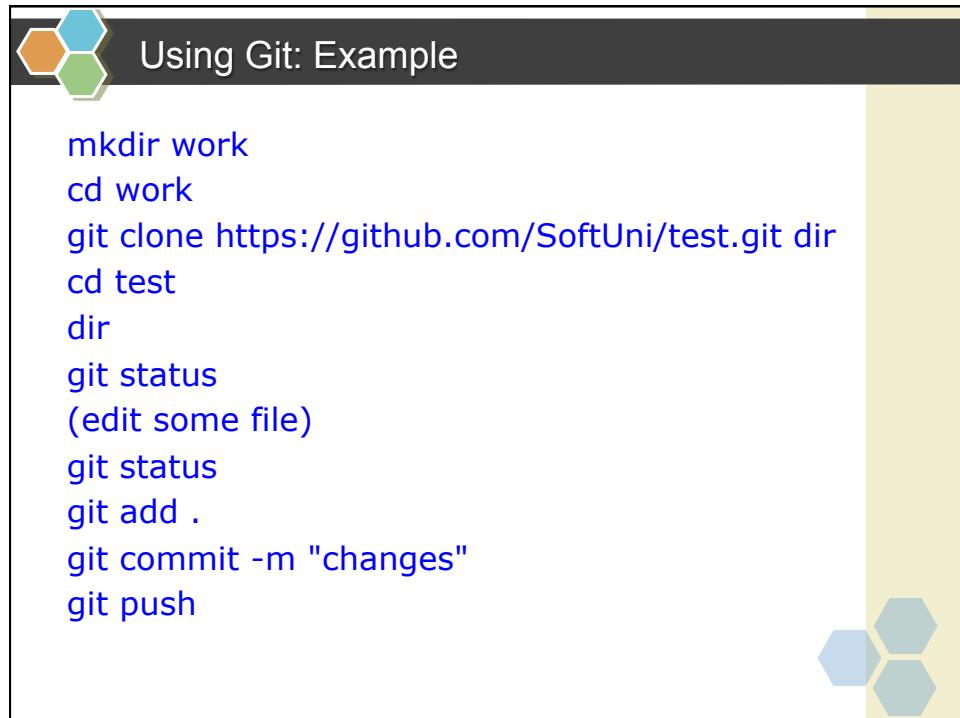
Basic Git Commands

- ❖ Cloning an existing Git repository  
`git clone [remote url]`
- ❖ Fetch and merge the latest changes from the remote repository  
`git pull`
- ❖ Preparing (adding / selecting) files for a commit  
`git add [filename] ("git add ." adds everything)`
- ❖ Committing to the local repository  
`git commit -m "[your message here]"`



Basic Git Commands

- ❖ Check the status of your local repository (see the local changes)  
`git status`
- ❖ Creating a new local repository (in the current directory)  
`git init`
- ❖ Creating a remote (assign a short name for remote Git URL)  
`git remote add [remote name] [remote url]`
- ❖ Pushing to a remote (send changes to the remote repository)  
`git push [remote name] [local name]`



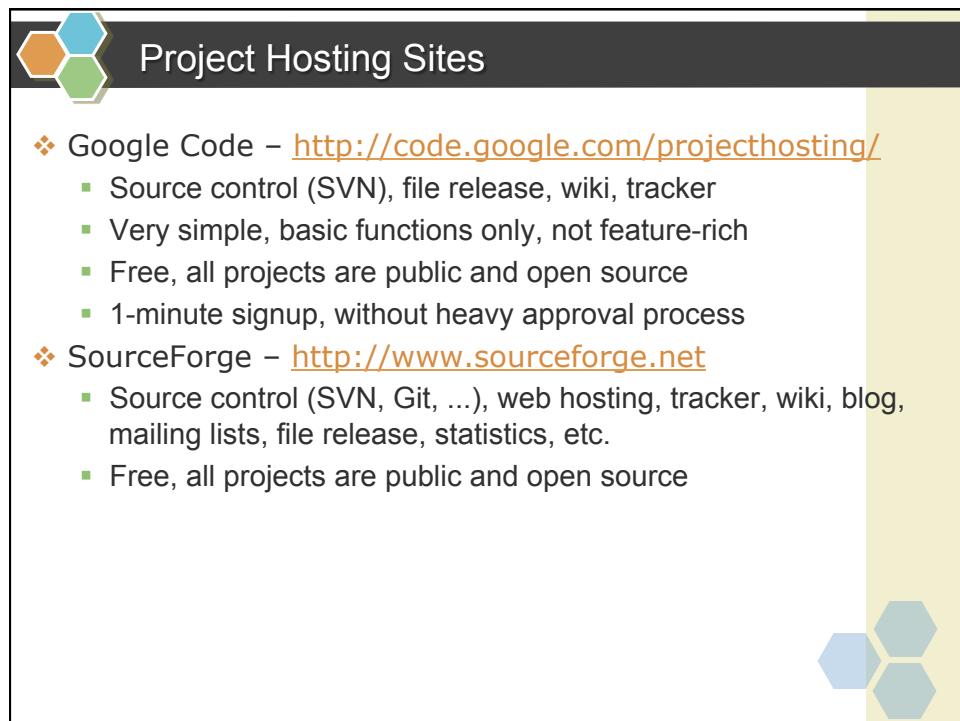
## Using Git: Example

```
mkdir work
cd work
git clone https://github.com/SoftUni/test.git dir
cd test
dir
git status
(edit some file)
git status
git add .
git commit -m "changes"
git push
```



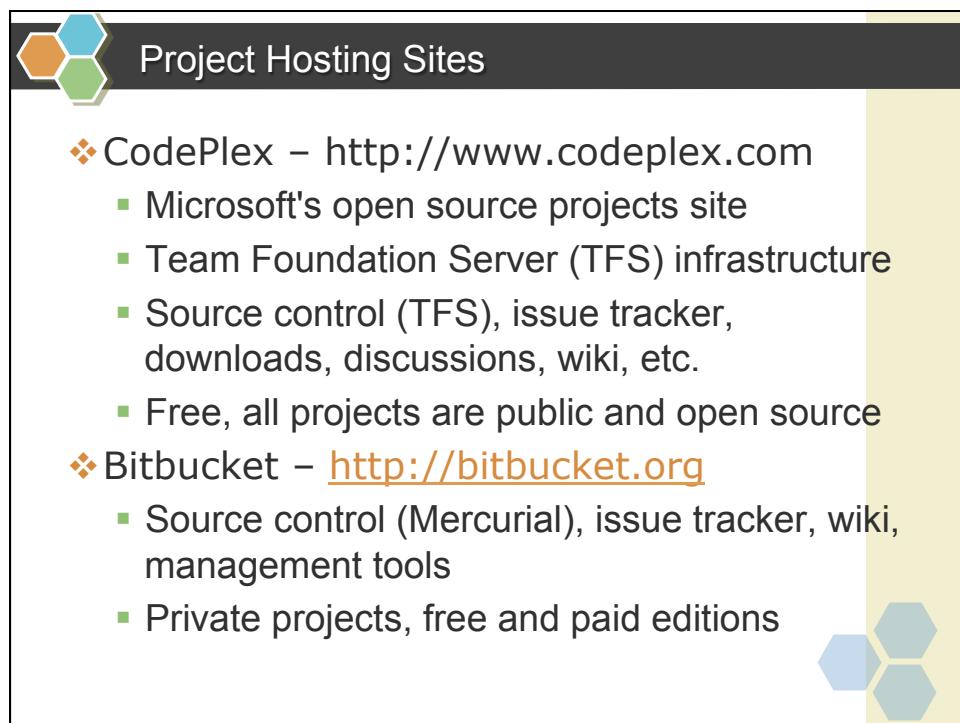
## Project Hosting Sites

- ❖ GitHub – <https://github.com>
  - The #1 project hosting site in the world
  - Free for open-source projects
  - Paid plans for private projects
- ❖ GitHub provides own Windows client
  - GitHub for Windows
  - http://windows.github.com
  - Dramatically simplifies Git
  - For beginners only



## Project Hosting Sites

- ❖ Google Code – <http://code.google.com/projecthosting/>
  - Source control (SVN), file release, wiki, tracker
  - Very simple, basic functions only, not feature-rich
  - Free, all projects are public and open source
  - 1-minute signup, without heavy approval process
- ❖ SourceForge – <http://www.sourceforge.net>
  - Source control (SVN, Git, ...), web hosting, tracker, wiki, blog, mailing lists, file release, statistics, etc.
  - Free, all projects are public and open source



## Project Hosting Sites

- ❖ CodePlex – <http://www.codeplex.com>
  - Microsoft's open source projects site
  - Team Foundation Server (TFS) infrastructure
  - Source control (TFS), issue tracker, downloads, discussions, wiki, etc.
  - Free, all projects are public and open source
- ❖ Bitbucket – <http://bitbucket.org>
  - Source control (Mercurial), issue tracker, wiki, management tools
  - Private projects, free and paid editions