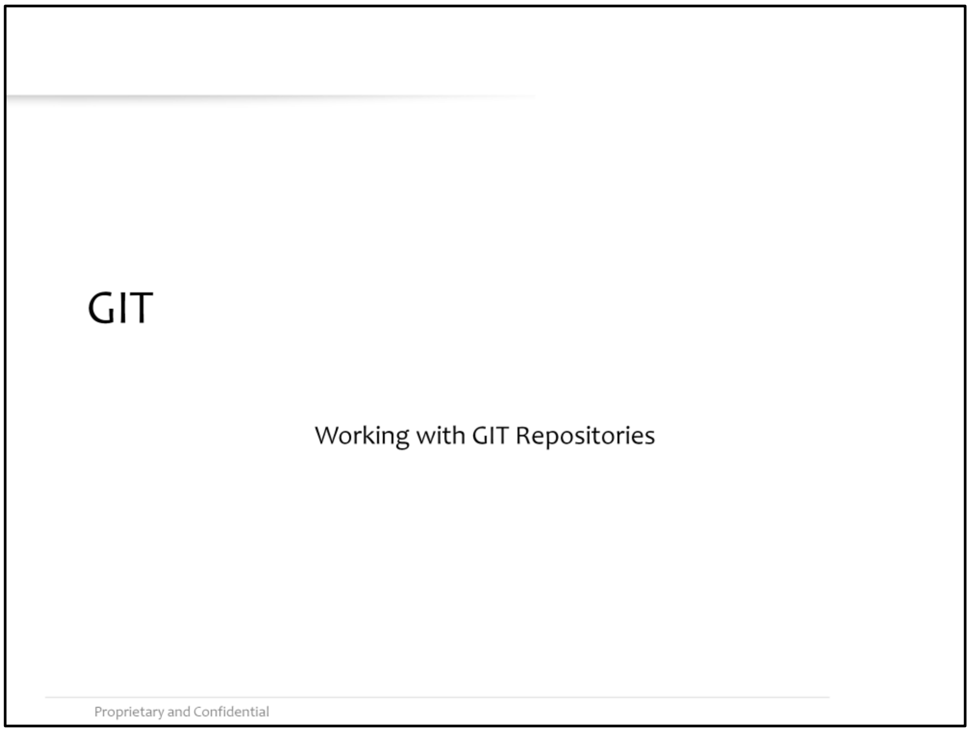
Instructor Notes:

Add instructor notes here.



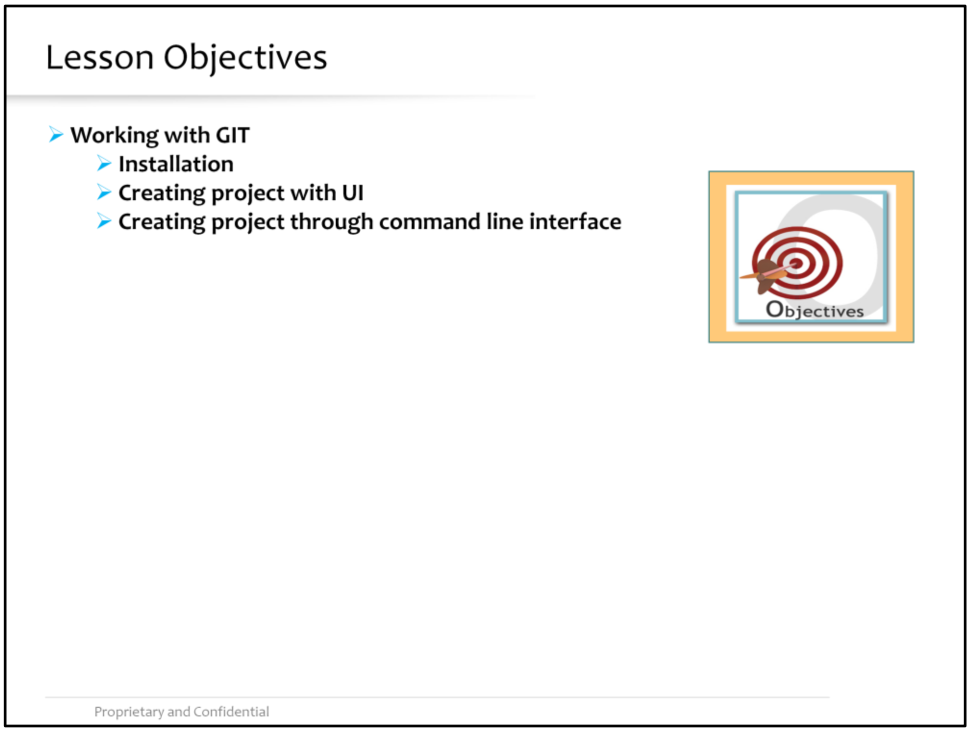
GIT

Working with GIT Repositories

Proprietary and Confidential

Instructor Notes:

Add instructor notes here.



Lesson Objectives

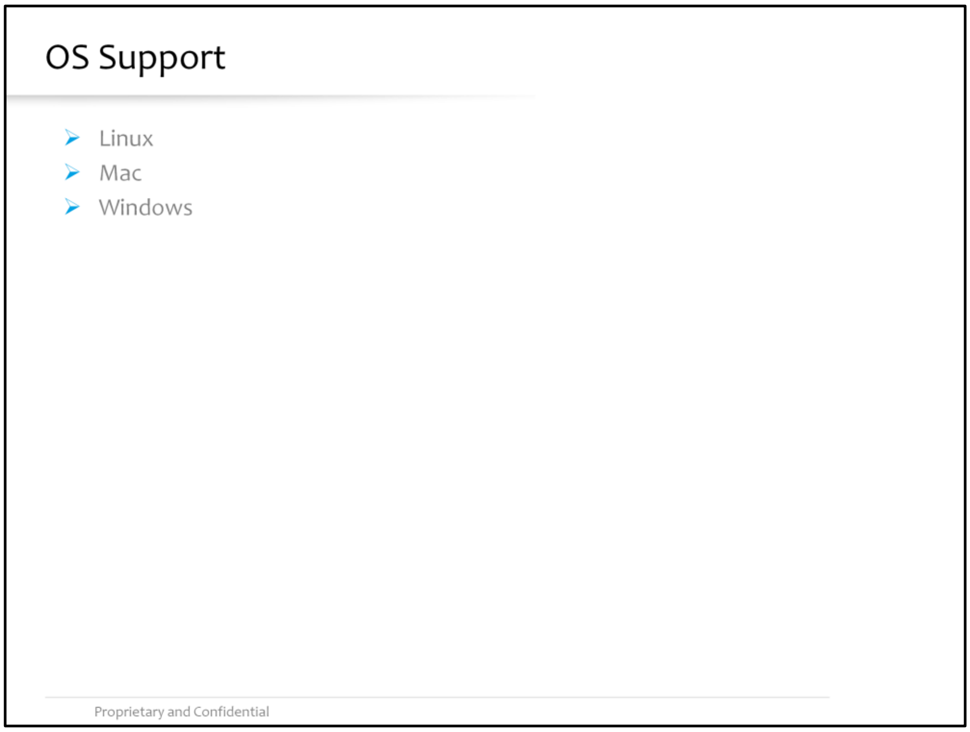
- Working with GIT
  - Installation
  - Creating project with UI
  - Creating project through command line interface

Objectives

Proprietary and Confidential

Instructor Notes:

Add instructor notes here.



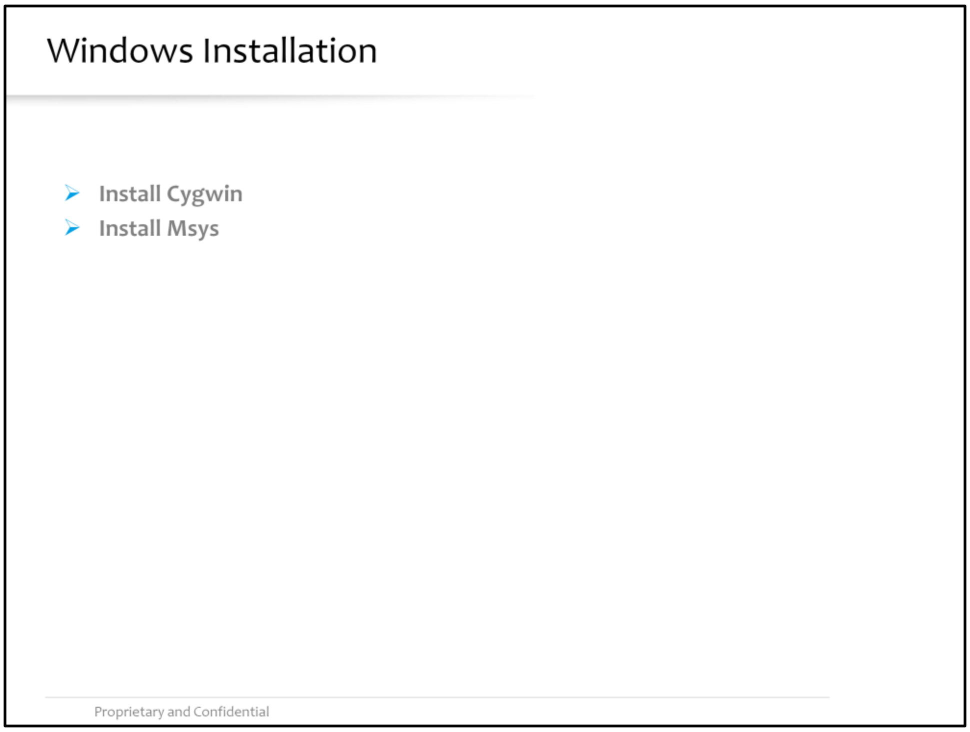
OS Support

- Linux
- Mac
- Windows

Proprietary and Confidential

Ref : A 40-byte hex representation of a SHA1 or a name that denotes a particular object. They may be stored in a file under $GIT_DIR/refs/ directory, or in the $GIT_DIR/packed-refs file
Reachable: All of the ancestors of a given commit are said to be "reachable" from that commit. More generally, one object is reachable from another if we can reach the one from the other by a chain that follows tags to whatever they tag, commits to their parents or trees, and trees to the trees or blobs that they contain.
Porcelain: Cute name for programs and program suites depending on core git, presenting a high level access to core git. Porcelains expose more of a SCM interface than the plumbing
Alternate object database: Via the alternates mechanism, a repository can inherit part of its object database from another object database, which is called "alternate

Instructor Notes:

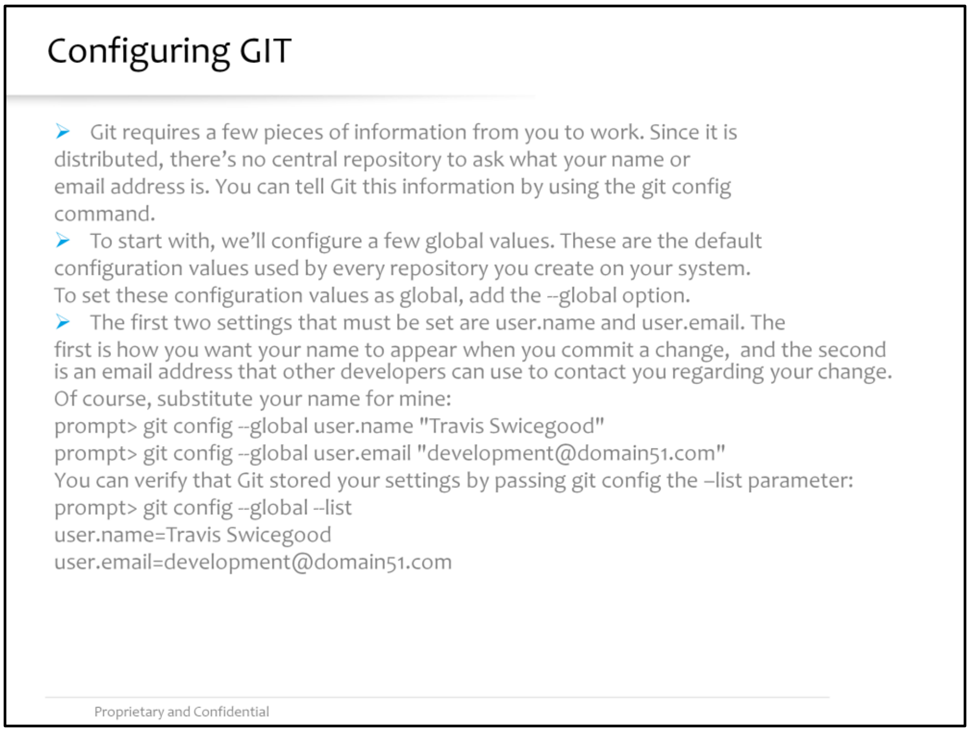


Windows Installation

- Install Cygwin
- Install Msys

Proprietary and Confidential

Instructor Notes:



Configuring GIT

- Git requires a few pieces of information from you to work. Since it is distributed, there's no central repository to ask what your name or email address is. You can tell Git this information by using the git config command.
- To start with, we'll configure a few global values. These are the default configuration values used by every repository you create on your system. To set these configuration values as global, add the --global option.
- The first two settings that must be set are user.name and user.email. The first is how you want your name to appear when you commit a change, and the second is an email address that other developers can use to contact you regarding your change. Of course, substitute your name for mine:

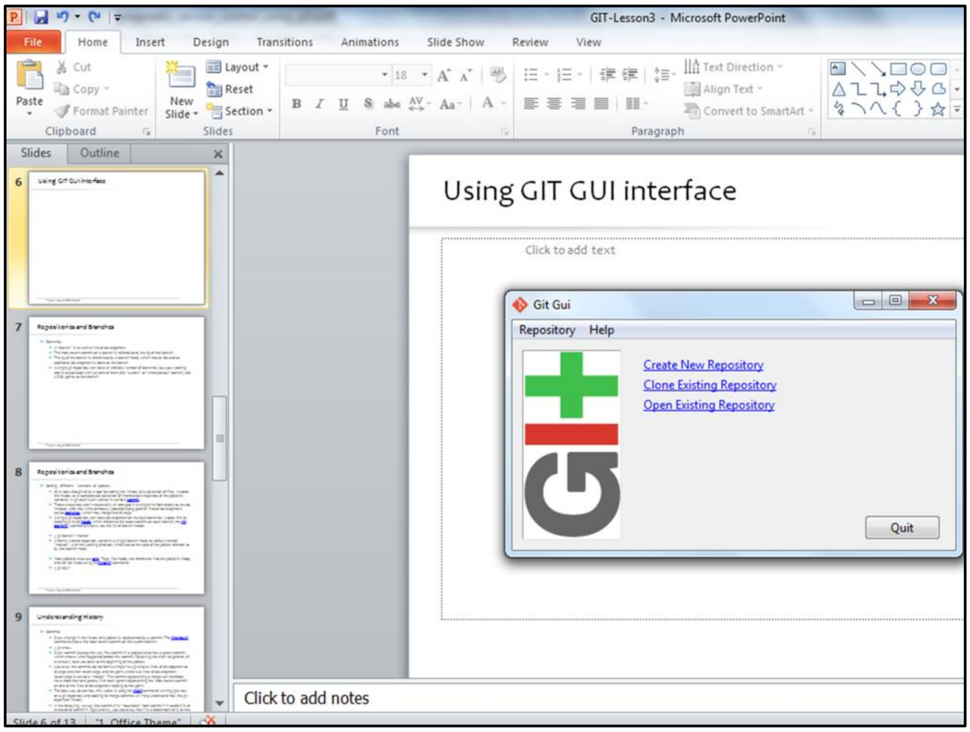
```
prompt> git config --global user.name "Travis Swicegood"
prompt> git config --global user.email "development@domain51.com"
```

You can verify that Git stored your settings by passing git config the –list parameter:

```
prompt> git config --global --list
user.name=Travis Swicegood
user.email=development@domain51.com
```

Proprietary and Confidential

Instructor Notes:


GIT-Lesson3 - Microsoft PowerPoint

Using GIT GUI interface

Click to add text

Git Gui

Repository Help

Create New Repository

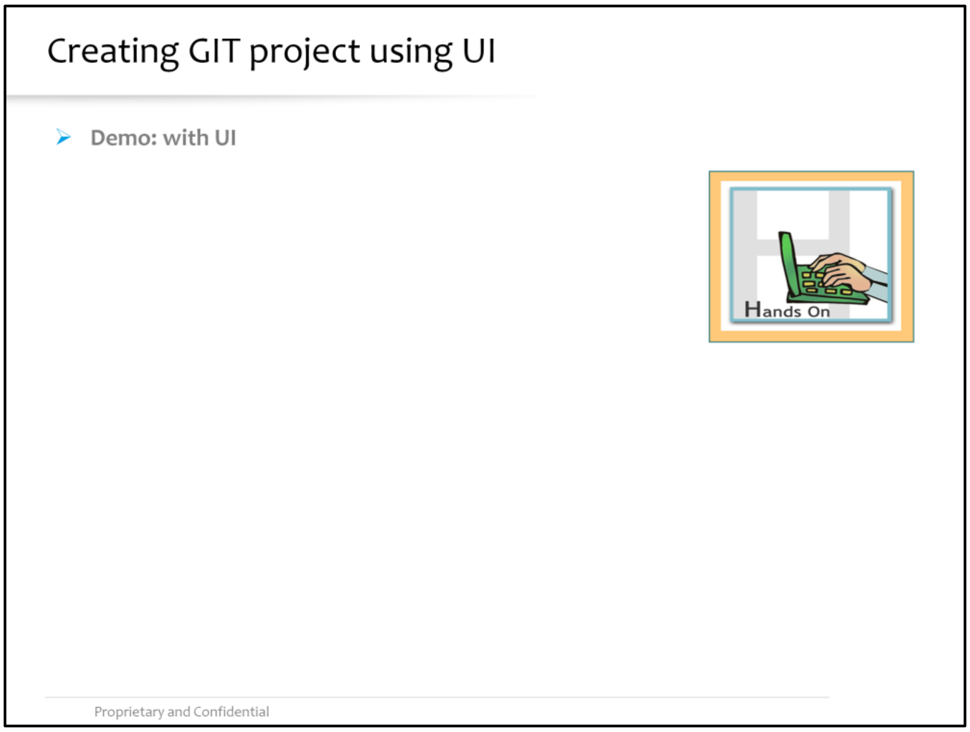
Clone Existing Repository

Open Existing Repository

Quit

Click to add notes

Instructor Notes:

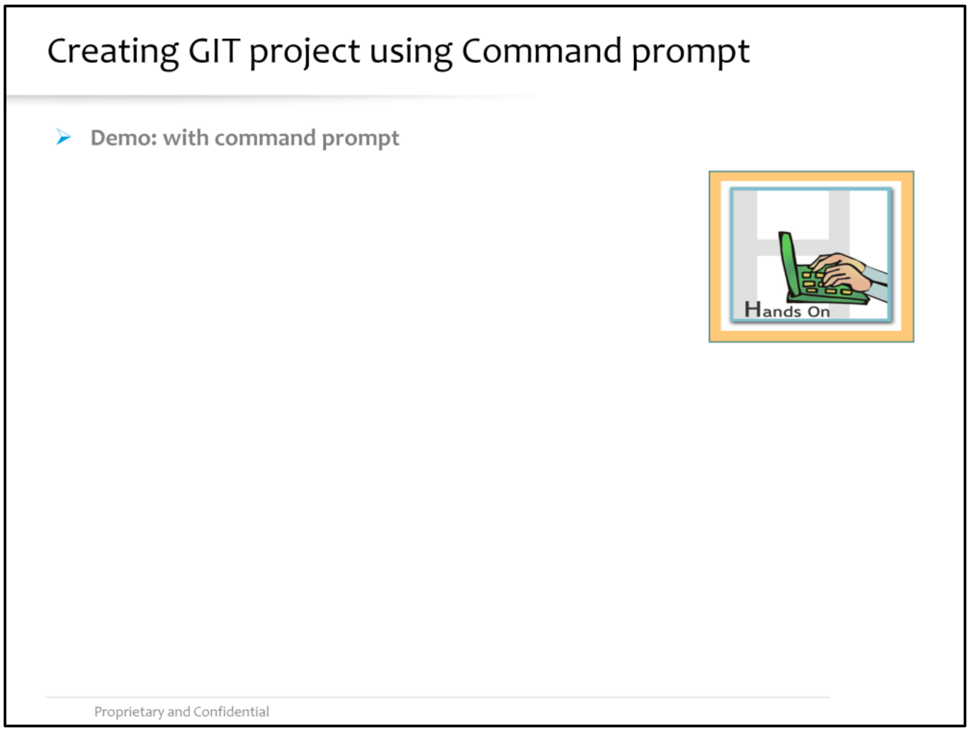


Creating GIT project using UI

- Demo: with UI

Proprietary and Confidential

Instructor Notes:

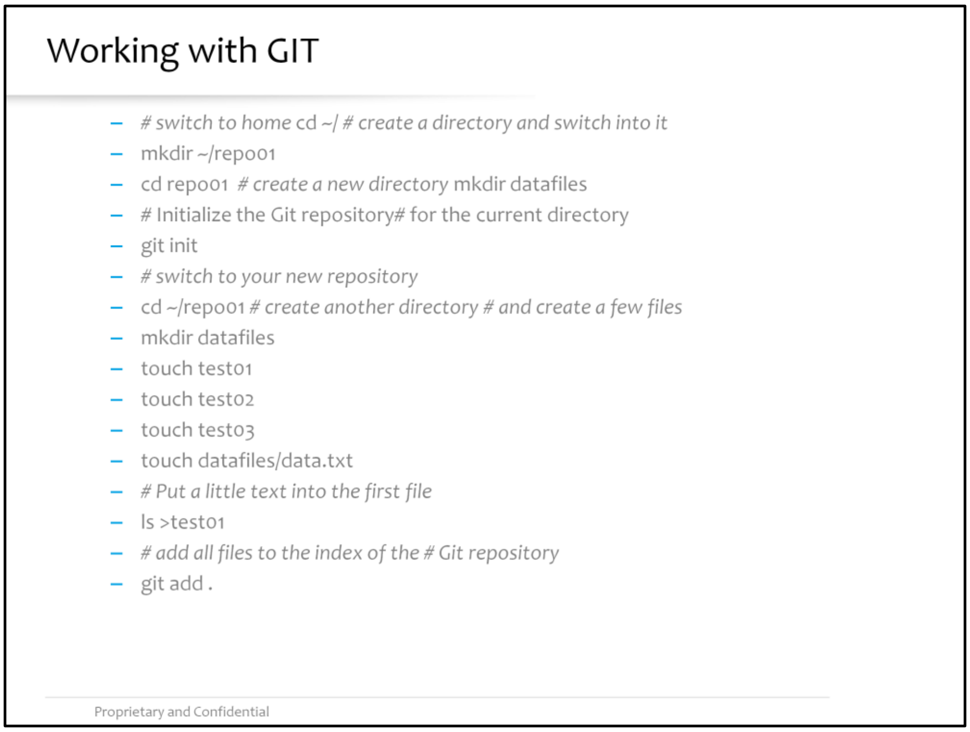


Creating GIT project using Command prompt

- Demo: with command prompt

Proprietary and Confidential

Instructor Notes:

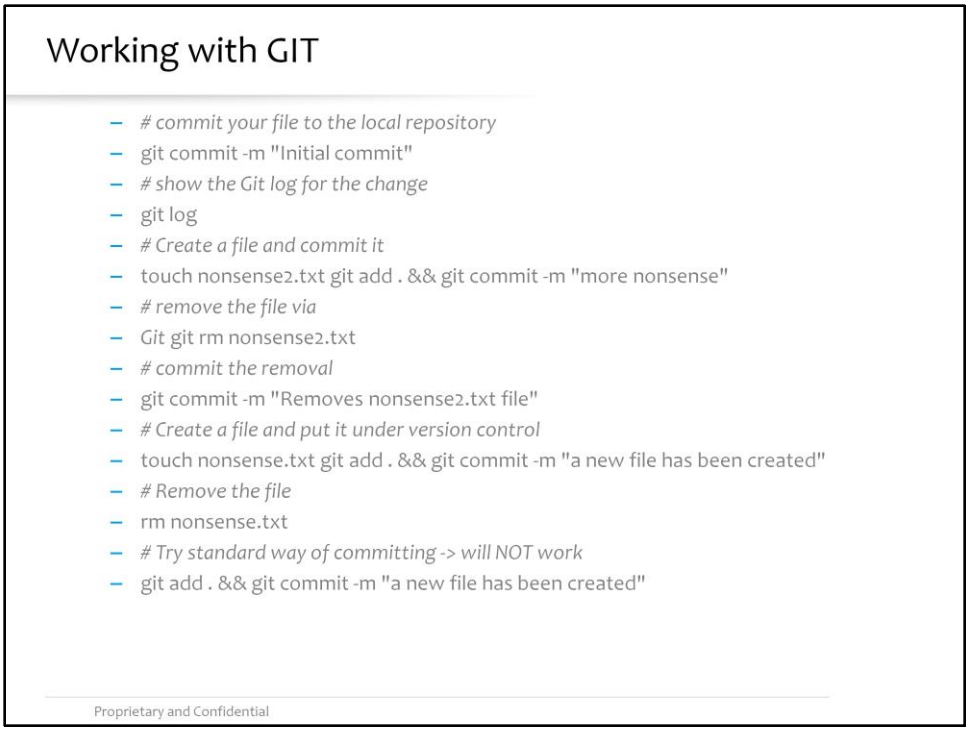


Working with GIT

- *# switch to home cd ~/ # create a directory and switch into it*
- mkdir ~/repo01
- cd repo01 *# create a new directory* mkdir datafiles
- # Initialize the Git repository# for the current directory
- git init
- *# switch to your new repository*
- cd ~/repo01 *# create another directory # and create a few files*
- mkdir datafiles
- touch test01
- touch test02
- touch test03
- touch datafiles/data.txt
- *# Put a little text into the first file*
- ls >test01
- *# add all files to the index of the # Git repository*
- git add .

Proprietary and Confidential

Instructor Notes:

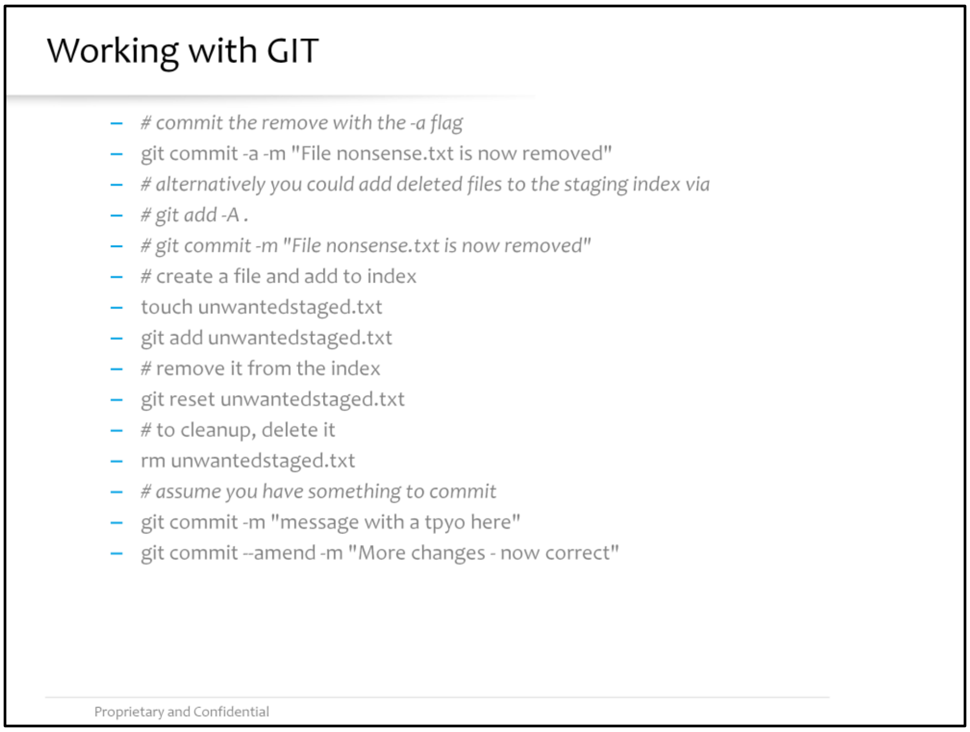


Working with GIT

- *# commit your file to the local repository*
- git commit -m "Initial commit"
- *# show the Git log for the change*
- git log
- *# Create a file and commit it*
- touch nonsense2.txt git add . && git commit -m "more nonsense"
- *# remove the file via*
- Git git rm nonsense2.txt
- *# commit the removal*
- git commit -m "Removes nonsense2.txt file"
- *# Create a file and put it under version control*
- touch nonsense.txt git add . && git commit -m "a new file has been created"
- *# Remove the file*
- rm nonsense.txt
- *# Try standard way of committing -> will NOT work*
- git add . && git commit -m "a new file has been created"

Proprietary and Confidential

Instructor Notes:

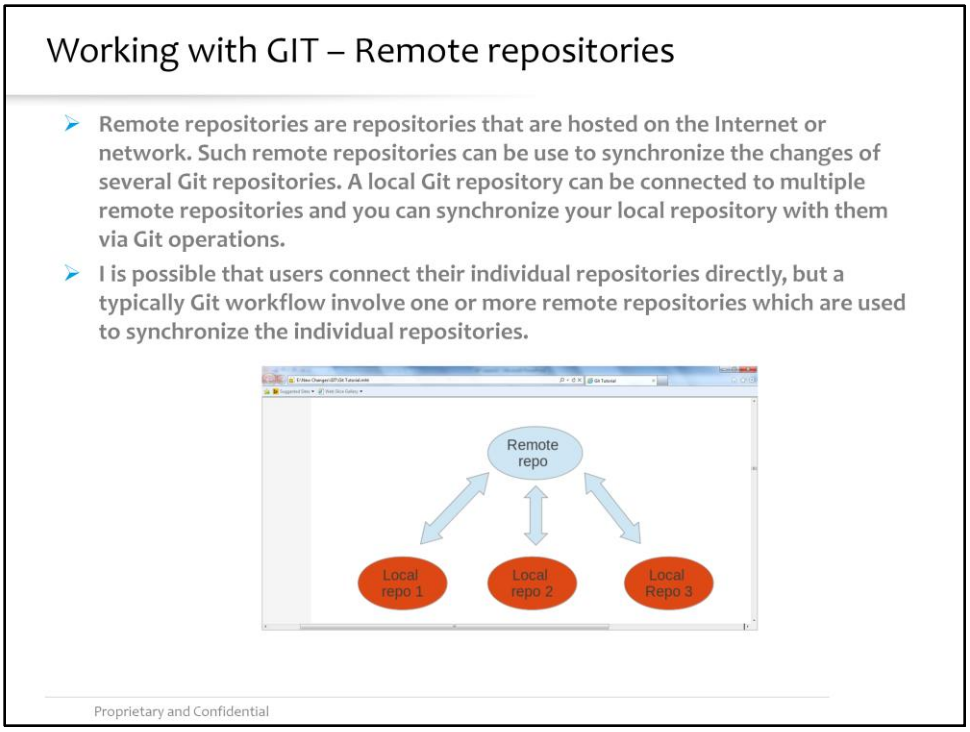


Working with GIT

- *# commit the remove with the -a flag*
- git commit -a -m "File nonsense.txt is now removed"
- *# alternatively you could add deleted files to the staging index via*
- *# git add -A .*
- *# git commit -m "File nonsense.txt is now removed"*
- # create a file and add to index
- touch unwantedstaged.txt
- git add unwantedstaged.txt
- # remove it from the index
- git reset unwantedstaged.txt
- # to cleanup, delete it
- rm unwantedstaged.txt
- *# assume you have something to commit*
- git commit -m "message with a tpyo here"
- git commit --amend -m "More changes - now correct"

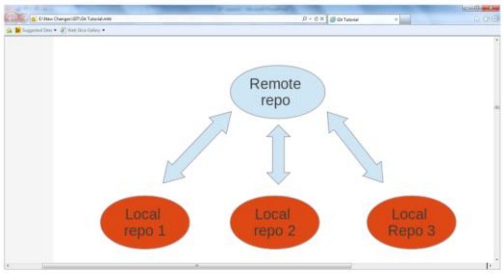
Proprietary and Confidential

Instructor Notes:



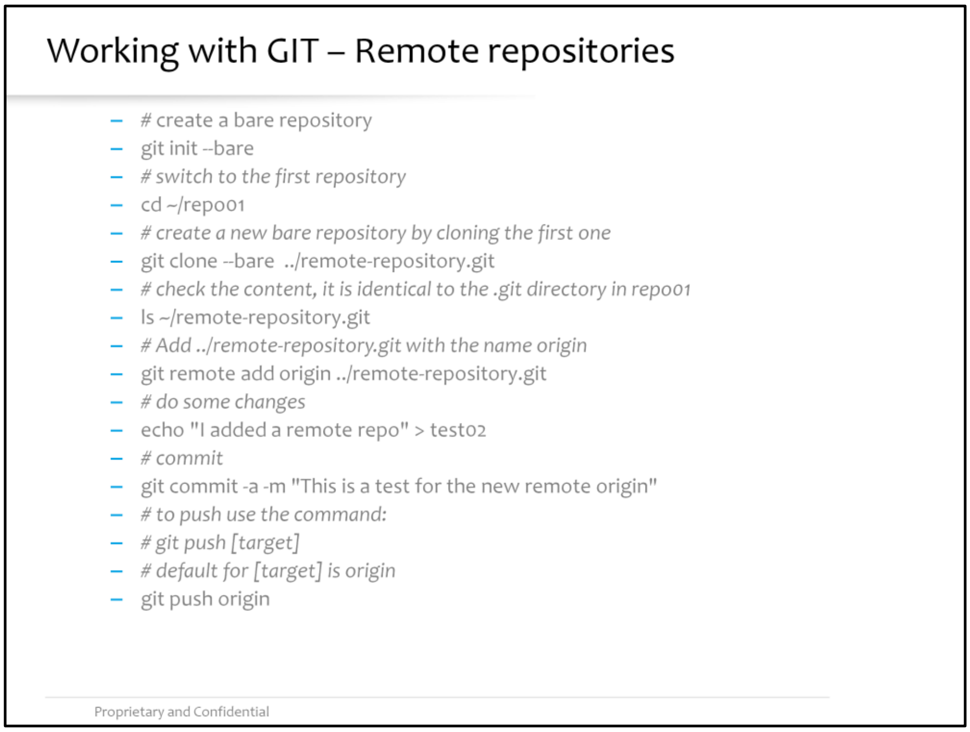
Working with GIT – Remote repositories

- Remote repositories are repositories that are hosted on the Internet or network. Such remote repositories can be use to synchronize the changes of several Git repositories. A local Git repository can be connected to multiple remote repositories and you can synchronize your local repository with them via Git operations.
- I is possible that users connect their individual repositories directly, but a typically Git workflow involve one or more remote repositories which are used to synchronize the individual repositories.



Proprietary and Confidential

Instructor Notes:



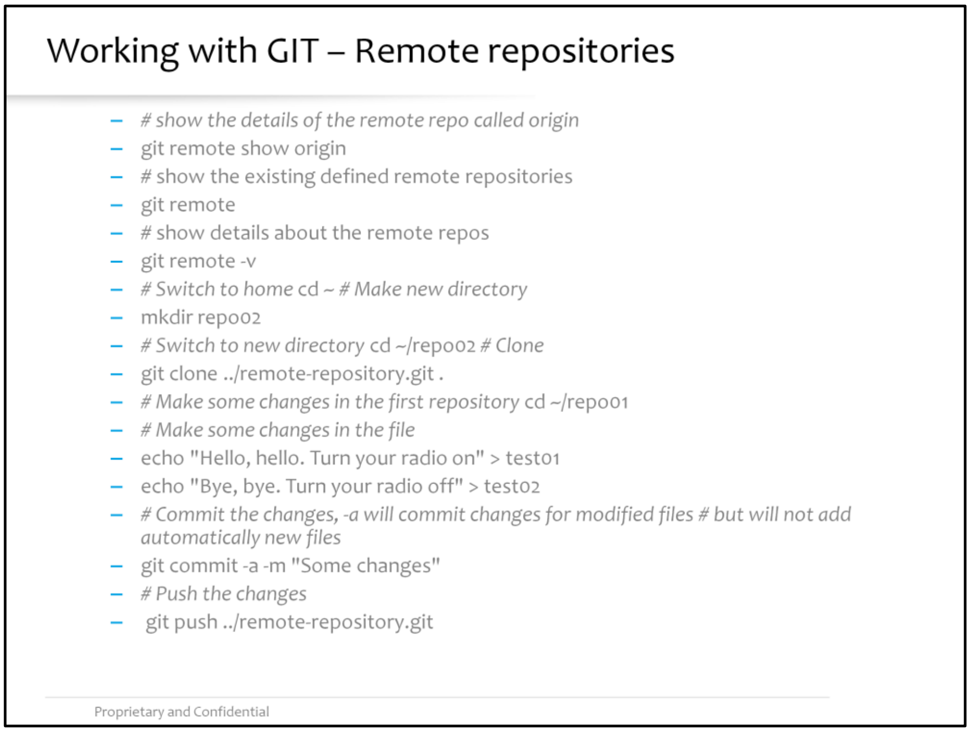
Working with GIT – Remote repositories

- # create a bare repository
- git init --bare
- # switch to the first repository
- cd ~/repo01
- # create a new bare repository by cloning the first one
- git clone --bare ../remote-repository.git
- # check the content, it is identical to the .git directory in repo01
- ls ~/remote-repository.git
- # Add ../remote-repository.git with the name origin
- git remote add origin ../remote-repository.git
- # do some changes
- echo "I added a remote repo" > test02
- # commit
- git commit -a -m "This is a test for the new remote origin"
- # to push use the command:
- # git push [target]
- # default for [target] is origin
- git push origin

Proprietary and Confidential

You can always connect to a remote repository if you know its URL and if you have access to it. If you clone (copy) a repository from another repository, a connection to this original repository is automatically created under the name *origin*. You can use this name to get and retrieve data from the remote repository.

Instructor Notes:

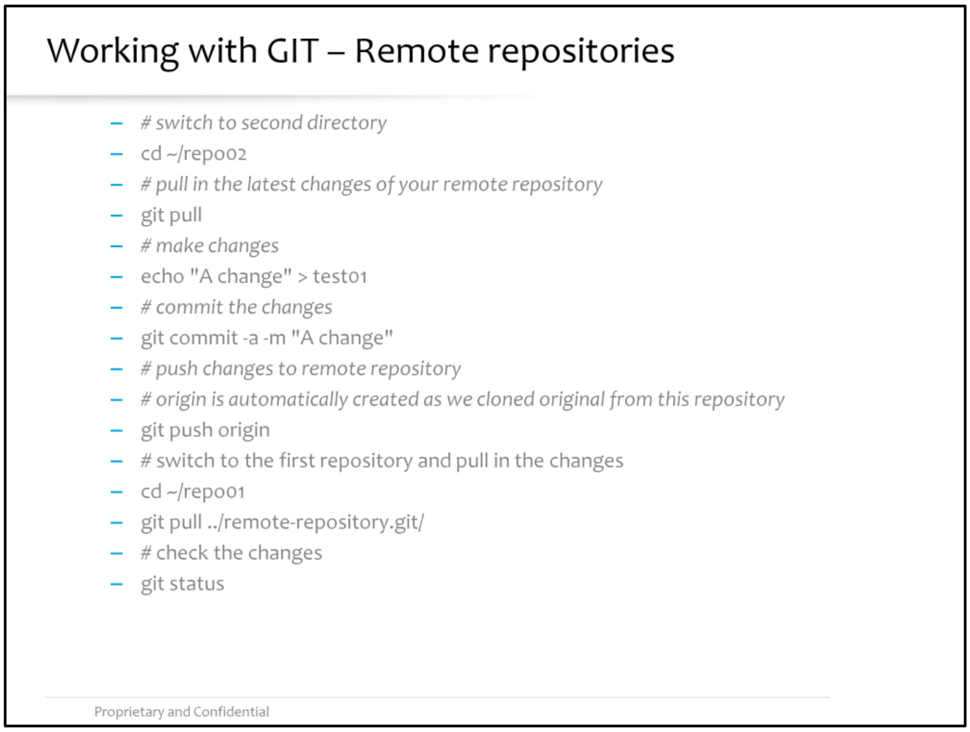


Working with GIT – Remote repositories

- *# show the details of the remote repo called origin*
- git remote show origin
- # show the existing defined remote repositories
- git remote
- # show details about the remote repos
- git remote -v
- *# Switch to home cd ~ # Make new directory*
- mkdir repo02
- *# Switch to new directory cd ~/repo02 # Clone*
- git clone ../remote-repository.git .
- *# Make some changes in the first repository cd ~/repo01*
- *# Make some changes in the file*
- echo "Hello, hello. Turn your radio on" > test01
- echo "Bye, bye. Turn your radio off" > test02
- *# Commit the changes, -a will commit changes for modified files # but will not add automatically new files*
- git commit -a -m "Some changes"
- *# Push the changes*
- git push ../remote-repository.git

Proprietary and Confidential

Instructor Notes:



Working with GIT – Remote repositories

- *# switch to second directory*
- cd ~/repo02
- *# pull in the latest changes of your remote repository*
- git pull
- *# make changes*
- echo "A change" > test01
- *# commit the changes*
- git commit -a -m "A change"
- *# push changes to remote repository*
- *# origin is automatically created as we cloned original from this repository*
- git push origin
- # switch to the first repository and pull in the changes
- cd ~/repo01
- git pull ../remote-repository.git/
- # check the changes
- git status

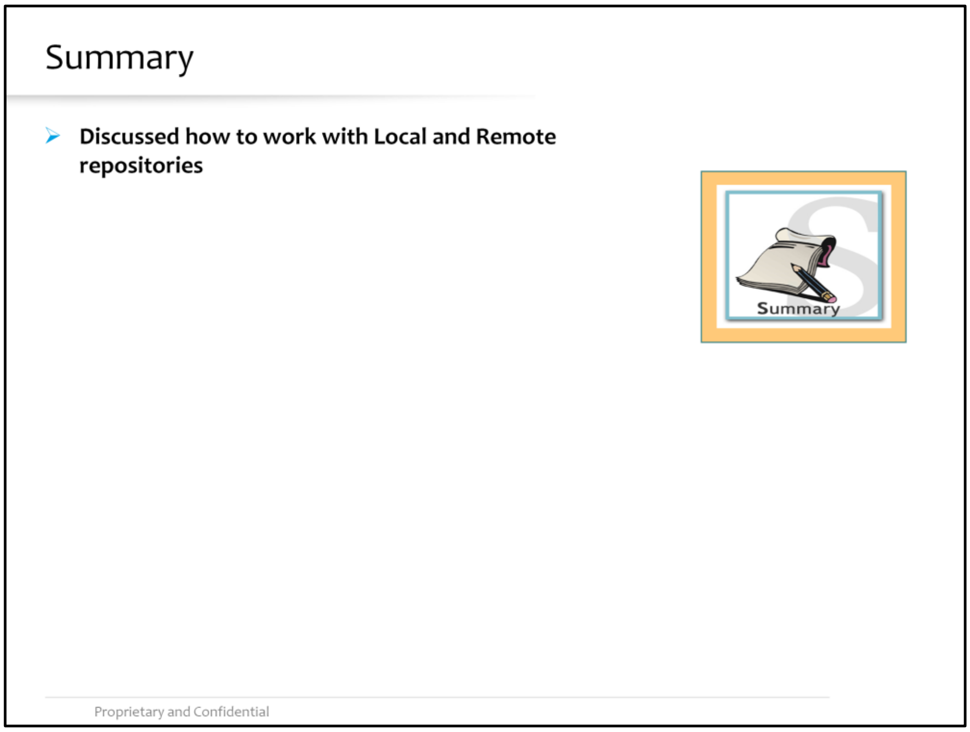
Proprietary and Confidential

Cloning remote repositories
Git support remote operations with other Git repositories. For communication with these repositories Git supports several transport types; the native protocol for Git is also called git.
The following will clone an existing repository via the Git protocol.
# switch to a new directory mkdir ~/online cd ~/online *# clone online repository* git clone git@github.com:vogella/gitbook.git
Alternatively you could clone the same repository via the http protocol.
# The following will clone via HTTP git clone http://vogella@github.com/vogella/gitbook.git

Remote operations via http and a proxy
It is possible to use the HTTP protocol to clone Git repositories. This is especially helpful, if your firewall blocks everything except http.
Git also provides support for http access via a proxy server. The following Git command could, for example, clone a repository via http and a proxy. You can either set the proxy variable in general for all applications or set it only for Git.
This example uses environment variables.
# Linux export http_proxy=http://proxy:8080 *# On Windows # Set http_proxy=http://proxy:8080* git clone http://dev.eclipse.org/git/org.eclipse.jface/org.eclipse.jface.snippets.git *# Push back to the origin using http* git push origin
This example uses the Git config settings.
// Set proxy for git globally git config --global http.proxy http://proxy:8080 // To check the proxy settings git config --get http.proxy // Just in case you need to you can also revoke the proxy settings git config --global --unset http.proxy

Instructor Notes:

Add instructor notes here.



Summary

- Discussed how to work with Local and Remote repositories

Proprietary and Confidential

Add the notes here.