GIT

Working with GIT Repositories

Lesson Objectives

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

OS Support

- Linux
- Mac
- Windows

Windows Installation

- Install Cygwin
- Install Msys

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
```

GIT-Lesson3 - Microsoft PowerPoint

File | Home | Insert | Design | Transitions | Animations | Slide Show | Review | View

Paste | Cut | Copy | Format Painter | Clipboard
New Slide | Layout | Reset | Section | Slides
Font
Text Direction | Align Text | Convert to SmartArt | Paragraph

Slides | Outline

6 Using GIT GUI interface

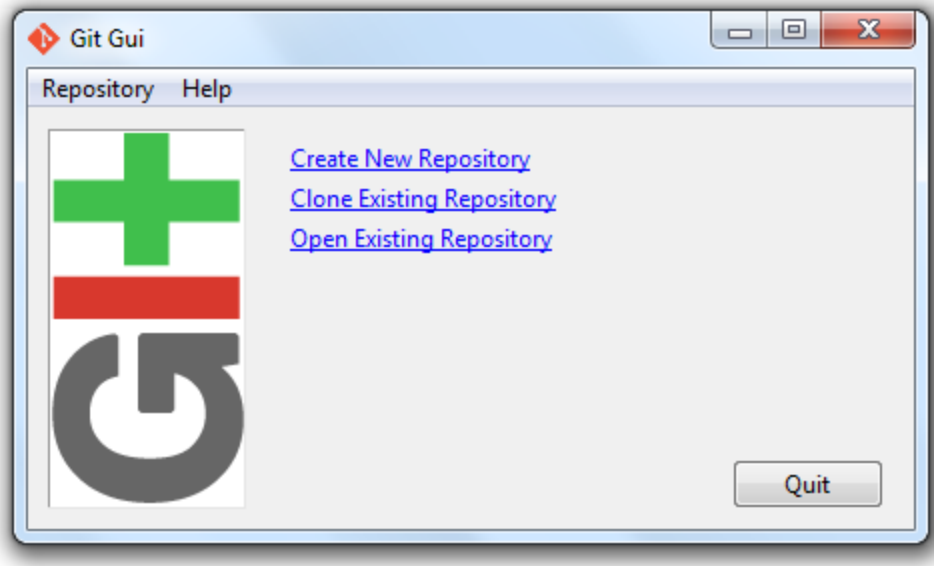
7 Repositories and Branches

8 Repositories and Branches

9 Understanding History

Using GIT GUI interface

Click to add text



Click to add notes

Slide 6 of 13 | "1_Office Theme"

Creating GIT project using UI

- Demo: with UI

Creating GIT project using Command prompt

- Demo: with command prompt

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 .

Working with GIT

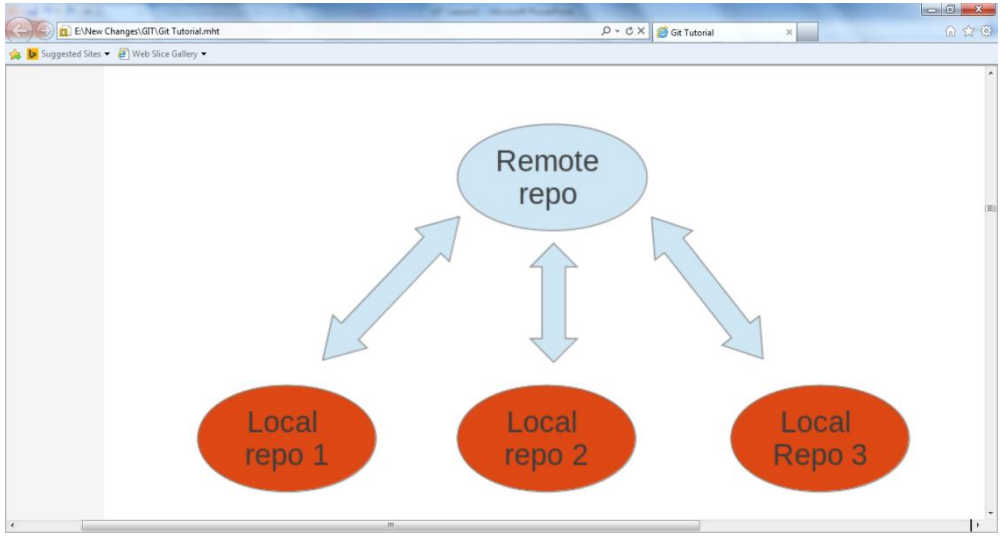
- *# commit your file to the local repository*
- git commit -m "Initial commit"
- *# show the Git log for the change*
- git log
- *# Create a file and commit it*
- touch nonsense2.txt git add . && git commit -m "more nonsense"
- *# remove the file via*
- *Git* git rm nonsense2.txt
- *# commit the removal*
- 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"

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"

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.



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

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

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

Summary

- **Discussed how to work with Local and Remote repositories**