Running a Git Server

on the AS/400

REMAIN
SOFTWARE

Agenda

- Introduction
- Git Basics
- Git Server Basic
  - GitHub
- Clone
- Push
- Pull or Fetch_and_Rebase
- Branching and Pull Requests
- Forking and Pull Request

REMAIN SOFTWARE

Introduction

Wim Jongman

CTO – Remain Software
TD/OMS and Gravity

All around techno freak
3D printing, IoT, Programming, DevOps, Web Graphics, Git fanboy.

Sloeber Committer, an IoT IDE

Eclipse Nebula Lead,
Committer for E4 Incubator.

In my free time I like to ..

Introduction

The Magic User Group Git repo:

https://github.com/magic-user

This content is here

https://github.com/magic-user/ug-meetings

REMAIN SOFTWARE

Introduction

Who are we

Why are we here

What is our skill level in

- Windows command line
- Unix command line
- Git

What are our goals

Introduction

What are my goals.

Introduce you to Git

Introduction

- Did You:
  - Install Git?
  - Install Putty ?
  - Create a GitHub account ?
  - Get your account on the magic-ug machine ?

Git Basics

Git Basics

- What is Git
- UUID’s
- The Three Amigos
  - Repositories
  - The Working Directory
  - The Index
- Commits

What is Git

Git Basics

Git Basics – What is Git

- Git is a Source Version Control System (VCS)

- What are the features of a VCS?
  - Notion of a repository
  - Check-in / Check-out
  - Keep a change history

Git Basics – What is Git

- Git is a Distributed (VCS)

A Distributed VCS is able to have a shared state across multiple remote machines.

How is this possible?

The UUID

Git Basics

REMAIN SOFTWARE

Git Basics – UUID’s

- Universal **Unique** ID
- 32 byte number (2^256)
- between 0 and

13.407.807.929.934.078.340.780.792.994.259.079.299.425.942.340.780.792.934.093.407.807.929.942.593.407.807.929.942.594.259.340.780.792.994.259.593.407.807.929.943.407.803.407.807.929.942.597.929.942.592.597.099.574.024.998.206 *

*** Approximately**

REMAIN SOFTWARE

Git Basics – UUID’s

- Why is this UUID important?
- Git is a distributed version control system
- Git uses a UUID as key for every commit
- Therefore:
  - If two UUID’s across distributed repositories are equal then it is the same commit.
  - If the same UUID exist in two repositories then it is the same repository.

The Three Amigos

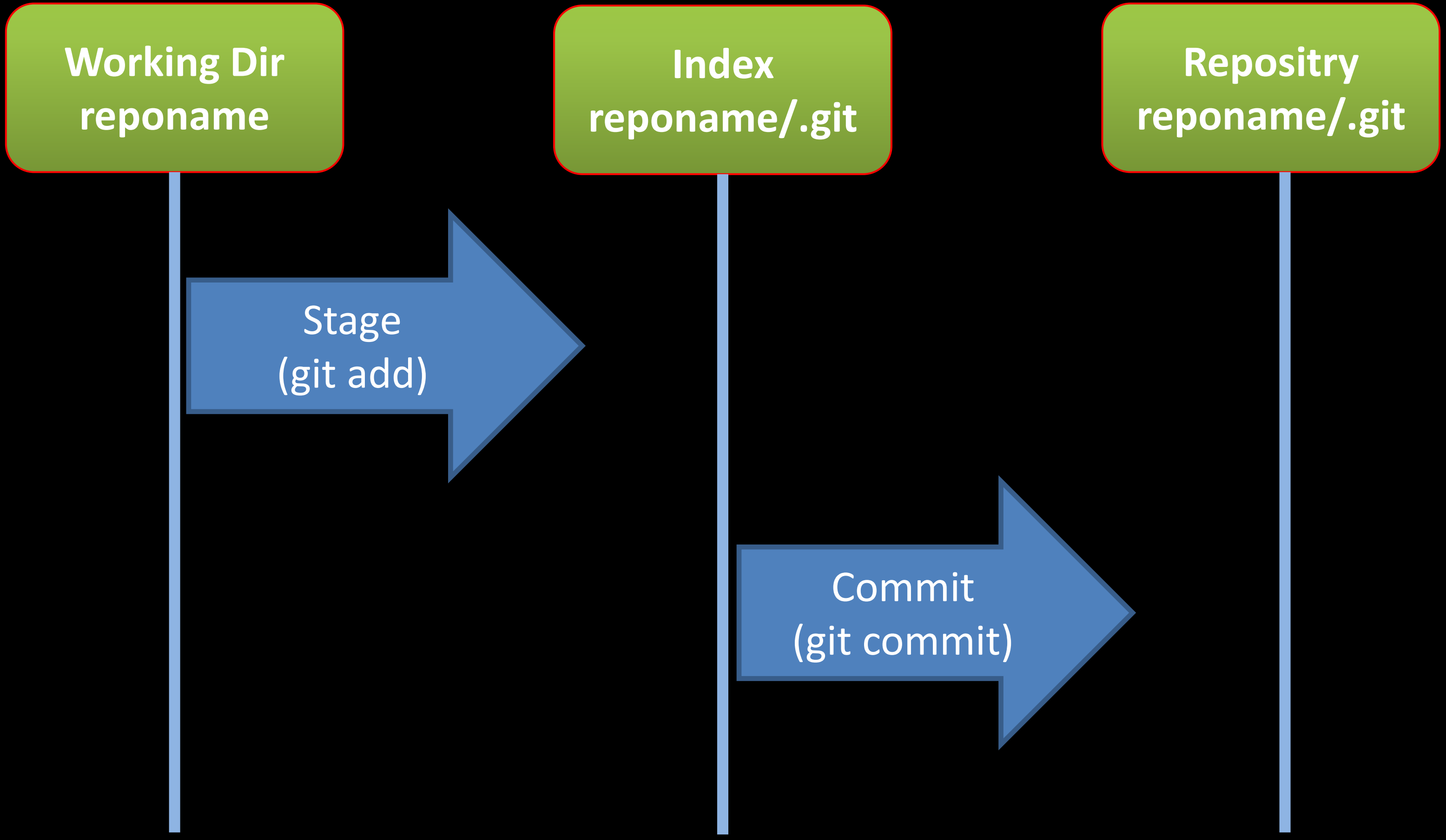
The Working Directory, the Index and the Repository

Git Basics

REMAIN SOFTWARE

Git Basics – What is Git

- Change/Stage/Commit workflow



The Repository the

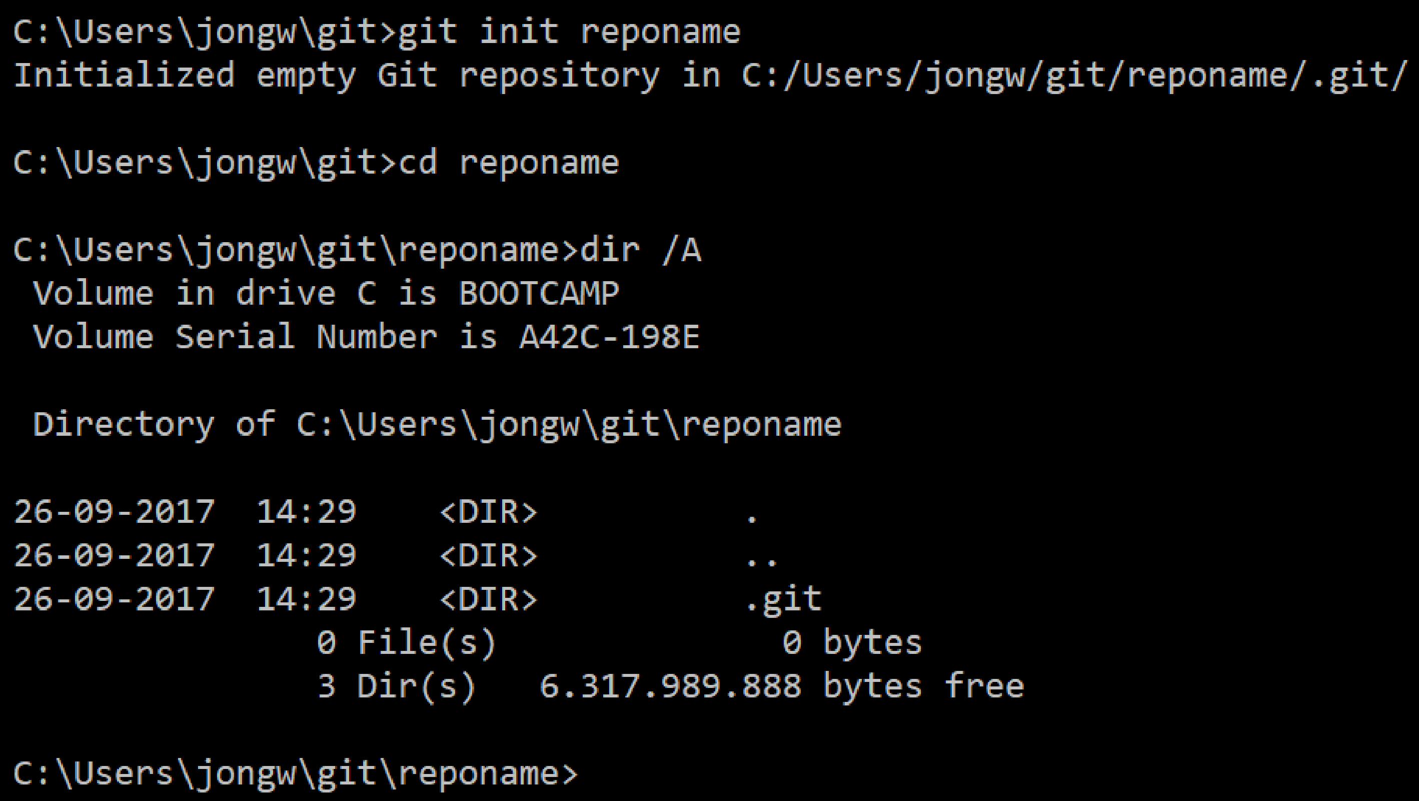
Git Basics

Git Basics – Repositories

- Central place of storage
- Can be created by:
  - Initializing
    - Create a new repository
  - Cloning
    - Copy from a server
  - Forking
    - Duplicate a repository on the server

Git Basics – Repositories

- Initializing a repository is easy!



```
C:\Users\jongw\git>git init reponame
Initialized empty Git repository in C:/Users/jongw/git/reponame/.git/

C:\Users\jongw\git>cd reponame

C:\Users\jongw\git\reponame>dir /A
 Volume in drive C is BOOTCAMP
 Volume Serial Number is A42C-198E

 Directory of C:\Users\jongw\git\reponame

26-09-2017  14:29    <DIR>          .
26-09-2017  14:29    <DIR>          ..
26-09-2017  14:29    <DIR>          .git
               0 File(s)              0 bytes
               3 Dir(s)   6.317.989.888 bytes free

C:\Users\jongw\git\reponame>
```

REMAIN SOFTWARE

Repository

Exercise

Git Basics – Repositories

- Install git
- Create a git directory in your home directory
- Create 5 repositories in this directory
- Use: **git init <*reponame*>**
- Examine the generated files

The Working Directory

Git Basics

Git Basics – The Working Directory

- Root of the repository directory
- This is where you create and edit files
  - Create files
  - Chang files
  - Remove files

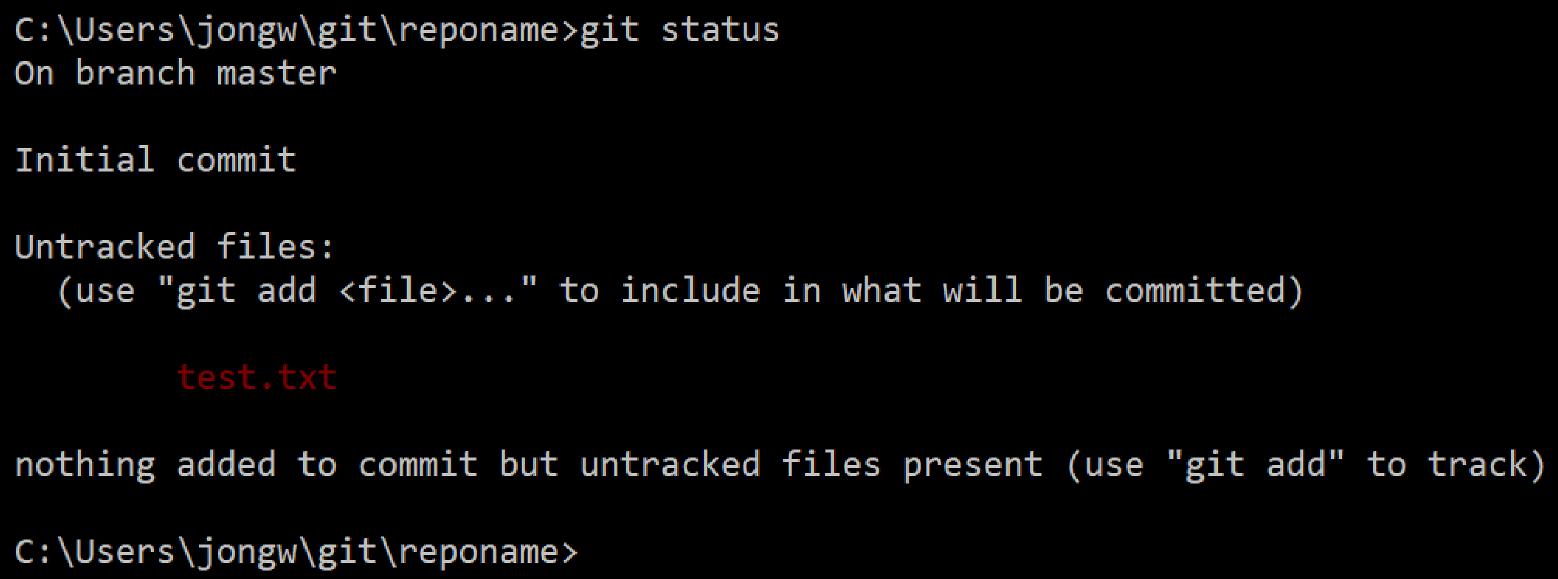
- IT IS NOT THE REPOSITORY
  - Repository sits somewhere in .git directory

Working Directory

Exercise

Git Basics – The Working Directory

- Create some files in a working directory
- Go to the command line and use: **git status**



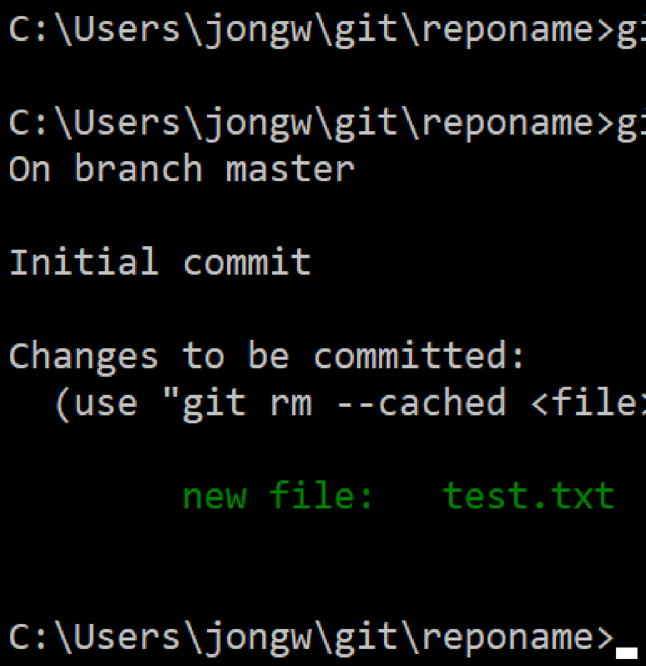
```
C:\Users\jongw\git\reponame>git status
On branch master

Initial commit

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

        test.txt

nothing added to commit but untracked files present (use "git add" to track)

C:\Users\jongw\git\reponame>
```

The Index

Git Basics

Git Basics – The Index

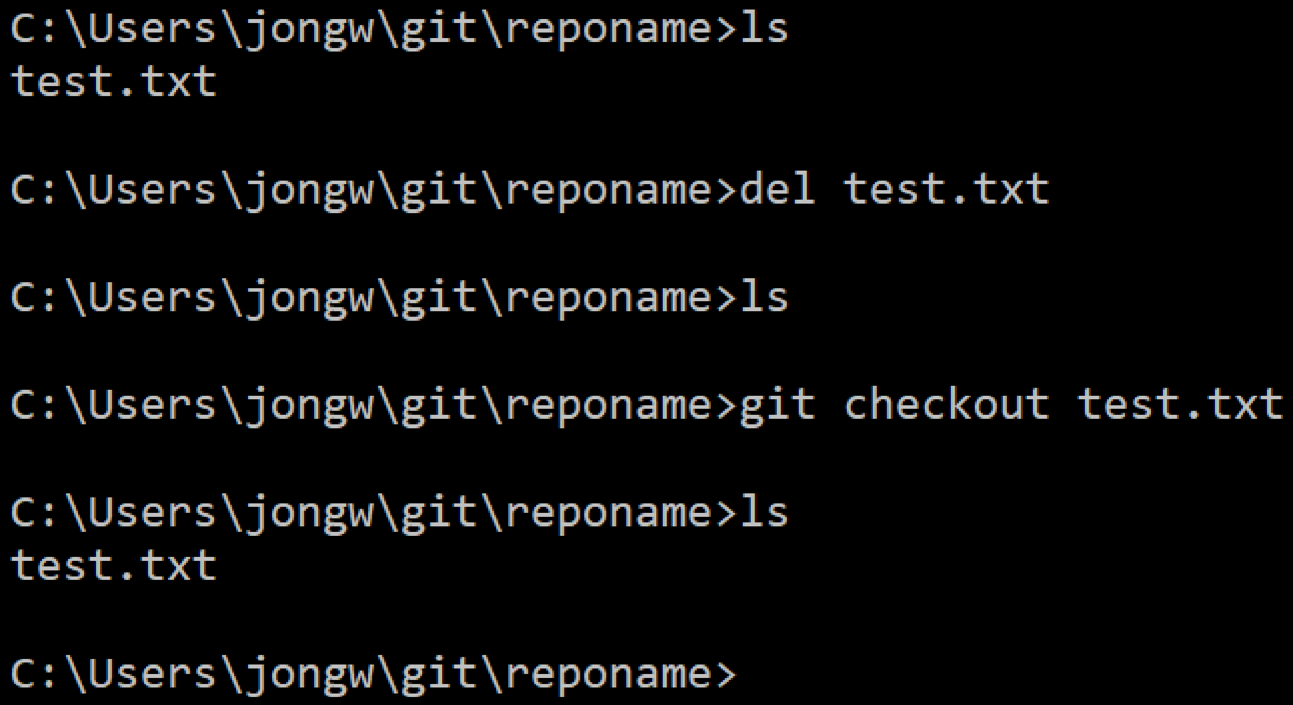
- Staging area to stage files
- Only staged files can be committed
- Use: **git add <*filename*|*>**



```
C:\Users\jongw\git\reponame>git add *

C:\Users\jongw\git\reponame>git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   test.txt


C:\Users\jongw\git\reponame>_
```

Git Basics – The Index

- To copy a file from the index to the working directory
- Use: **git checkout <*filename*>**



```
C:\Users\jongw\git\reponame>ls
test.txt

C:\Users\jongw\git\reponame>del test.txt

C:\Users\jongw\git\reponame>ls

C:\Users\jongw\git\reponame>git checkout test.txt

C:\Users\jongw\git\reponame>ls
test.txt

C:\Users\jongw\git\reponame>
```

REMAIN SOFTWARE

Index

Exercise

Git Basics – The Index

- Add some files to the index with **git add**
- Use: **git status** to view the result
- Remove a file (from the working directory)
- Restore it with: **git checkout <*filename*>**

Commits

Git Basics

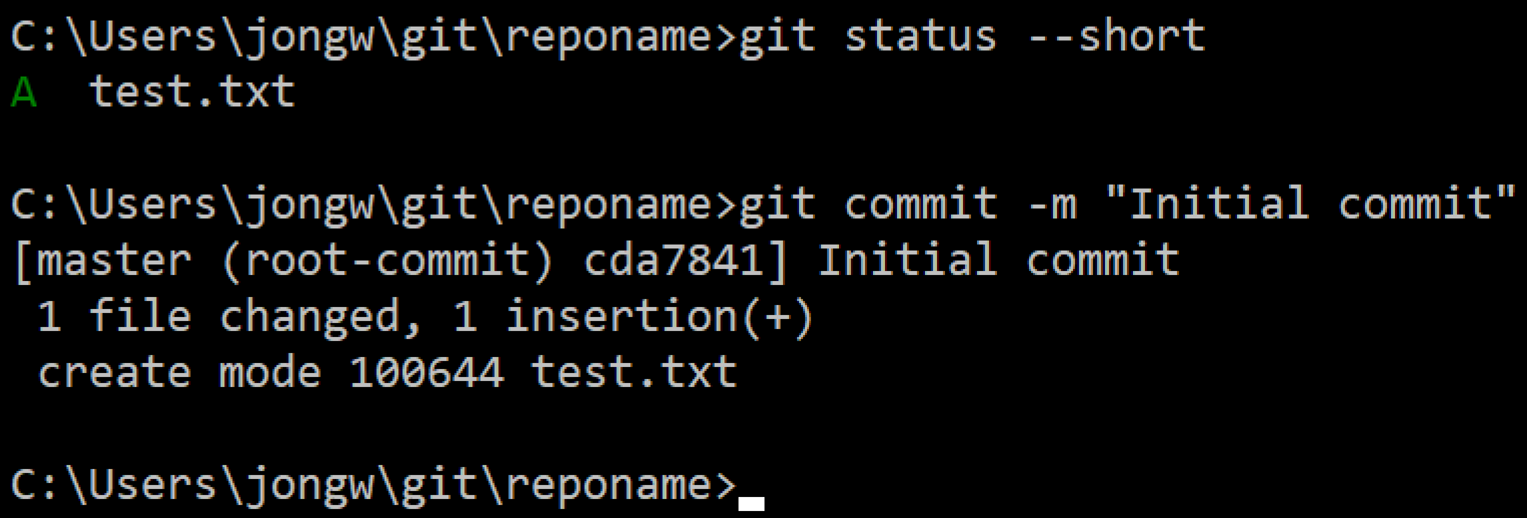
Git Basics – Commits

- Commits move files from the index to the repository.
- Only staged (index) files can be committed
- Use: **git status** to see what will be committed
- To commit
  Use: **git commit –m “*commit message*”**

REMAIN SOFTWARE

Git Basics – Commits

- To commit
  Use: **git commit –m “*commit message*”**



```
C:\Users\jongw\git\reponame>git status --short
A  test.txt

C:\Users\jongw\git\reponame>git commit -m "Initial commit"
[master (root-commit) cda7841] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt

C:\Users\jongw\git\reponame>
```

REMAIN SOFTWARE

Git Basics – Commits

- Use: **git status**

```
C:\Users\jongw\git\reponame>git status
On branch master
nothing to commit, working directory clean
```

- Use: **git log**

```
C:\Users\jongw\git\reponame>git log
commit cda784165aa84e25dffe29fe65da318fdf2137ce
Author: Wim Jongman <wim.jongman@remainsoftware.com>
Date:   Tue Sep 26 16:44:43 2017 +0200

    Initial commit
```

REMAIN SOFTWARE

Commits

Exercise

Git Basics – Commits

- Commit the files you’ve created
- Use **git status** and **git log** to view the results

Recap

Git Basics

Git Basics – Recap

// Create a repository
git init <*reponame*>

// Create files in the repo directory and stage
git add <*filename*|*>

// Commit files to the repo
git commit –m "*message*"

// Copy a file from the index back to the Working Directory
git checkout <*filename*>

// Show the status of the index and the Working Directory
git status

// Show the commit log
git log

// To get Help
git help <*command*>

REMAIN SOFTWARE

Git Server Basics

Git Server Basics

A git server:

is used to centralize commits from collaborators into a central repository;

enables users to copy (fork) the repositories to their own area on the server;

enables users to download (clone) the repositories to their laptops and upload (push) the changes back into the server;

GitHub

REMAIN SOFTWARE

GitHub

- Creating a GitHub account.
- Creating a Repository
- Cloning the Repository
- Pushing changes

GitHub

Exercise

GitHub

- Create a GitHub account
- Create a Repository (initialize with README)

Cloning

GitHub

GitHub - Cloning

Cloning is the act of downloading a repository from the Git Server.

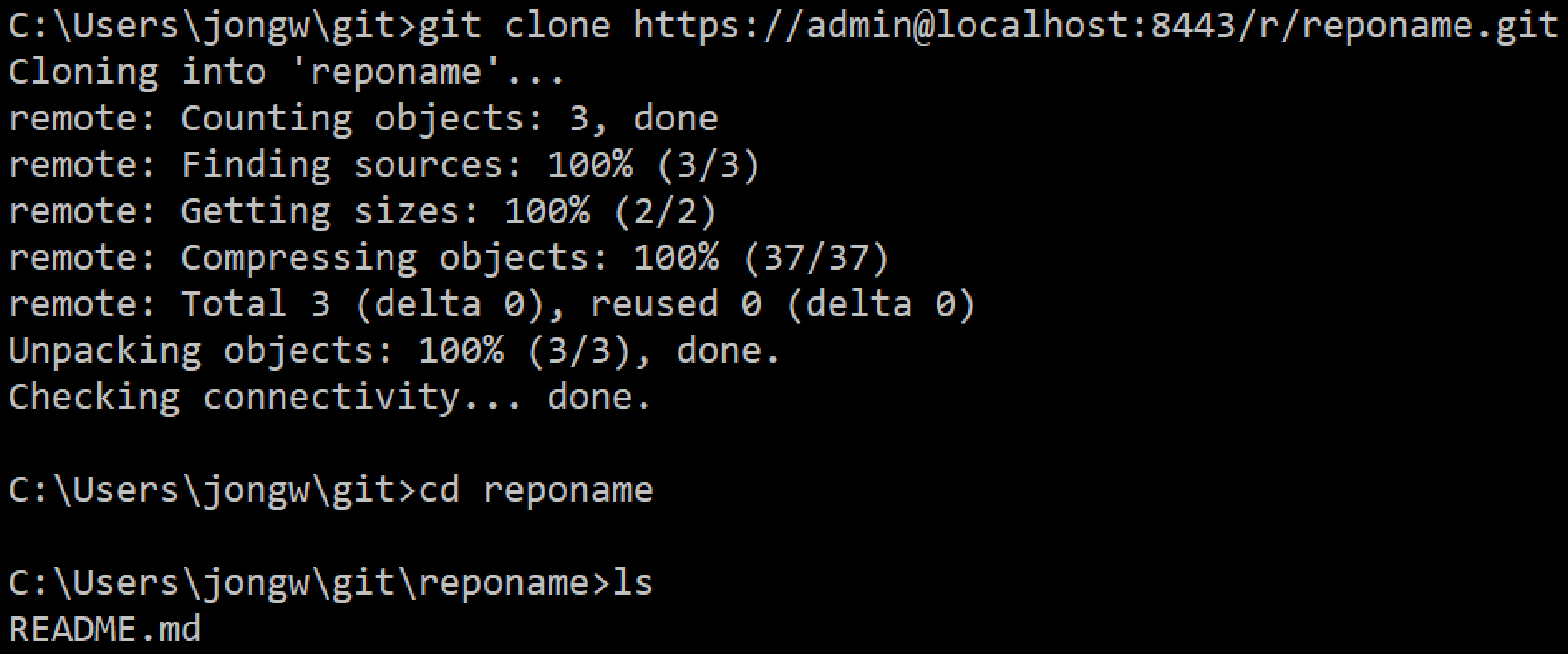
Cloning is done from your git directory.

The cloning process will automatically create the repository directory.

The cloning process will associate the remote repository with the cloned (local) repository (**origin**).

GitHub - Cloning

- Open the command line
- Go to your git directory
- Use: **git clone <*url*>**



```
C:\Users\jongw\git>git clone https://admin@localhost:8443/r/reponame.git
Cloning into 'reponame'...
remote: Counting objects: 3, done
remote: Finding sources: 100% (3/3)
remote: Getting sizes: 100% (2/2)
remote: Compressing objects: 100% (37/37)
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
Checking connectivity... done.

C:\Users\jongw\git>cd reponame

C:\Users\jongw\git\reponame>ls
README.md
```

Cloning

Exercise

REMAIN SOFTWARE

GitHub - Cloning

- Find the GitHub repository URL
- Open your command line
- Go to your git directory
- Use: **git clone <*url*>**
- Change into that directory
- Change or add a file
- Stage and commit that change

Pushing

GitHub

GitHub - Pushing

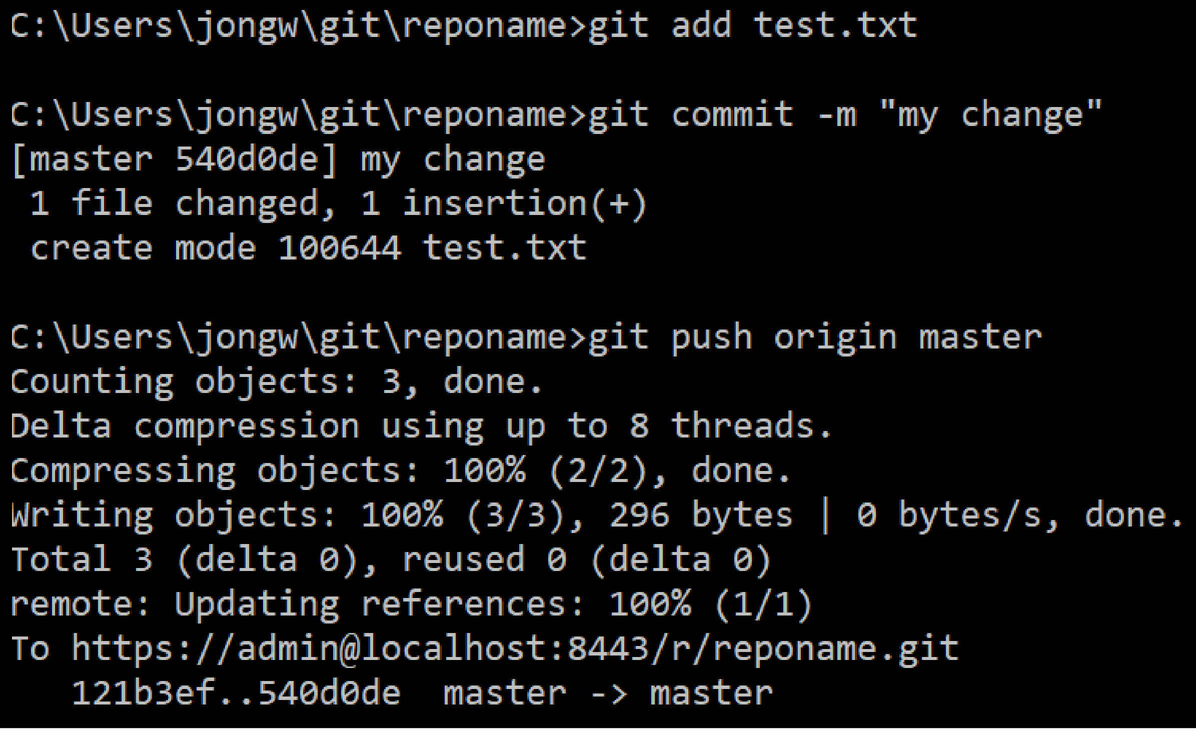
Pushing is the act of uploading committed changes into the remote repository.

GitHub - Pushing

- Open the command line
- Go to your git directory
- Change, stage and commit files

GitHub - Pushing

- Use: **git push origin master**
- **origin**: The remote repository
- **master**: The current branch



```
C:\Users\jongw\git\reponame>git add test.txt

C:\Users\jongw\git\reponame>git commit -m "my change"
[master 540d0de] my change
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt

C:\Users\jongw\git\reponame>git push origin master
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 296 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: Updating references: 100% (1/1)
To https://admin@localhost:8443/r/reponame.git
   121b3ef..540d0de  master -> master
```

Pushing

Exercise

REMAIN SOFTWARE

GitHub - Pushing

- Open your repository directory
- Change, Stage and Commit one or more files
- Push your changes: **git push origin master**
- Check your GitHub Repository

Pull or Fetch_and_Rebase

Merging Changes from Others

GitHub - Pulling

Pulling is a multi action function.

1. Fetch the changes from the remote
2. Merge the changes in the working directory
3. Commit a new merge commit

GitHub – Fetch And Rebase

Fetch only fetches the changes from the remote repository. You then have to:

Rebase: puts our commits on top of the remote commits.

OR

Merge: merges the changes and creates a new commit with two parents.

GitHub - Pulling

- Change a file directly on GitHub
- Open the command line
- Go to your git directory
- Use: git pull to get the changes from the server
- Use git log to see what happened

GitHub – Fetch And Rebase

- Change a file directly on GitHub
- Open the command line
- Go to your git directory
- Use: git fetch to get the changes from the server
- Use git status to see what happened
- Use git rebase to put your commits at the end
- Push the changes with git push

Branching

Demo

Forking and Pull Requests

Demo

REMAIN SOFTWARE

Thank You!!

REMAIN SOFTWARE