**TAM Bootcamp - Git**

Login to projects.eng.vmware.com

* Use your vmware AD credentials

-or-

Login to github.com

* Use your personal account credentials

**Getting Started**

On your laptop (this example is on the mac)

* Open a terminal session

Install homebrew (if you don't have it installed) on your mac

* <https://brew.sh/>

% brew update

% brew doctor

Install/upgrade git on your mac

* % git --version

Git version 2.24.1 (Apple Git-126)

* If you see this, you are running the apple version of git, not the official distribution
* You may even get the following nasty-gram from github

Hi @davemazur,

You recently used a password to access the repository at davemazur/velero\_dev\_guide with git using git/2.24.1 (Apple Git-126).

Basic authentication using a password to Git is deprecated and will soon no longer work. Visit <https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/> for more information around suggested workarounds and removal dates.

Thanks,  
The GitHub Team

* Next, install git

% brew install git

* Brew installs the official git distro in /usr/local/bin
* So far, so good. Now you have to change your path to use the official git distribution

% export PATH=/usr/local/bin:$PATH

* Confirm that the git version is correct

% git --version

Git version 2.30.0

* In order to upgrade git

% brew upgrade git

In a browser, go to projects.eng.vmware.com (or your github.com account)

* Go to Github.com/davemazur/tambootcamp-git
* Both have the same content

**Basic Commands**

Clone the sample project on github/gitlab

* Create and cd to the correct directory (best practice)
  + Cd /Users/dmazur/go/src/github.com/davemazur
* Git clone
  + Git clone <https://github.com/davemazur/tambootcamp-git.git>

% git clone <https://github.com/davemazur/tambootcamp-git.git>

Cloning into 'tambootcamp-git'...

remote: Enumerating objects: 6, done.

remote: Counting objects: 100% (6/6), done.

remote: Compressing objects: 100% (5/5), done.

remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0

Receiving objects: 100% (6/6), done.

* Cd into tambootcamp-git
* Look at .git directory
* Do a git status

% git status

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

**Modify/Checkin a file**

* Vi file
  + Make a change in the file

# file:  hello-world.go

# description:  my very first go program

package main

import "fmt"

func main() {

    fmt.Println("hello world")

}

**Git status**

% git status

On branch main

Your branch is up to date with 'origin/main'.

Changes not staged for commit:

  (use "git add <file>..." to update what will be committed)

  (use "git restore <file>..." to discard changes in working directory)

modified:   hello-world.go

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

**Git add**

* Git status

% git add hello-world.go

dmazur@dmazur-a01 tambootcamp-git % git status

On branch main

Your branch is up to date with 'origin/main'.

Changes to be committed:

  (use "git restore --staged <file>..." to unstage)

modified:   hello-world.go

**Git commit**

* Git status

% git commit -m "added notes to hello-world.go"

[main b7150cf] added notes to hello-world.go

 1 file changed, 3 insertions(+)

% git status

On branch main

Your branch is ahead of 'origin/main' by 1 commit.

  (use "git push" to publish your local commits)

nothing to commit, working tree clean

* *Combine "add" + "commit" into one command*
  + % git commit -a -m "new version"

**Git push**

% git push

Enumerating objects: 5, done.

Counting objects: 100% (5/5), done.

Delta compression using up to 8 threads

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 411 bytes | 411.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

To <https://github.com/davemazur/tambootcamp-git.git>

   818263e..b7150cf  main -> main

* Go over to git hub and see the changes

Graphical user interface, text, application, email

Description automatically generated

**Git restore**

* Accidentally delete a file in your local directory

% rm hello-world.go

% git restore hello-world.go

% ls

* Restore a previous version of a file from the master branch

% rm hello-world.go

% git restore --source main~2 hello-world.go

**Git log**

% git log

commit b7150cfac7fb1f216f5eaffc92ea6040f69bea6a (**HEAD -> main**, **origin/main**, **origin/HEAD**)

Author: Dave Mazur <davemazur@comcast.net>

Date:   Tue Jan 26 13:37:57 2021 -0500

    added notes to hello-world.go

commit 818263e5f2395b9adbc29f98ad4f83c754b7baff

Author: davemazur <davemazur@comcast.net>

Date:   Tue Jan 26 13:29:06 2021 -0500

    Create hello-world.go

commit fb2bab0aa08152172aa6aecb37afa1028405b7e0

Author: davemazur <davemazur@comcast.net>

Date:   Tue Jan 26 13:25:27 2021 -0500

    Initial commit

**Git help**

% git help git

% git help -a

* List of git commands

% git commit --help | more

**Advanced Topics**

**Git fork**

* Go to the github website

Graphical user interface, application, website

Description automatically generated

* Search for the "powershell" repository
* In the upper right hand corner select "fork"
* Select where you want to fork the repository

Graphical user interface, application, Teams

Description automatically generated

* Look at your repositories to view the new fork
* You can now clone from your forked repository

**Git pull**

* Two developers working in the same code base
* Developer 1 adds a file to a repository
  + Use the "Add File"->Create new file in the github web screen
  + Create test-pull
    - Add a comment to the file
  + Look at repository in github

Graphical user interface, text, application

Description automatically generated

* Developer 2 does a pull from your laptop to get the latest changes
  + % Git pull
* You should have pulled down the file that dev 1 created.
* Best practice
  + Pull every morning
  + On very heavily used repositories, pull multiple times during the day

**Branching**

* When you create a branch, you create an identical copy of the project at that point in time

**Remote Branching**

* You would use this for a project team...all team members work in this branch

* Navigate to your repository in github web
* Select the dropdown under the main branch

A picture containing graphical user interface

Description automatically generated

* Create a branch called "version-1"
* Now the number of branches changed to (2) and the default branch is "version-1"
  + You are automatically switched to the new branch
  + Any changes you make to the files in the repository will be applied to this branch

Graphical user interface, application, Teams

Description automatically generated with medium confidence

* Go to your local repository and
  + % git fetch
    - This updates the remote branch

% git branch -a

  main

\* testing

  remotes/origin/HEAD -> origin/main

  remotes/origin/main

  remotes/origin/version-1

**Local Branching**

* This is a branch that only "you" can see. It only exists on your local machine
* Git branch test-local-branch
* Git branch
  + Note \*main is the current context
* Git switch test-local-branch
  + Note \*test-local-branch is the current context
* *Note: Git checkout -b <name> will create the branch and switch contexts*
* Git status
* Make changes
* Push

Compare branches

* % git diff <local-branch> <origin/remote-branch>

**Merge changes with another developer**

* Dev 1 makes changes to hello.go in the remote repository in github web
* Dev 2 makes changes to hello.go in the local branch testing
* Dev 2 tries to checkin the changes
  + % git commit -a -m "checkin lines 3/4"
  + % git push

% git push

To <https://github.com/davemazur/kronos-git.git>

 ! [rejected]        testing -> testing (non-fast-forward)

error: failed to push some refs to '<https://github.com/davemazur/kronos-git.git>'

hint: Updates were rejected because the tip of your current branch is behind

hint: its remote counterpart. Integrate the remote changes (e.g.

hint: 'git pull ...') before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details

* Git status
* % git pull

* % vi hello.go

% more hello.go

<<<<<<< HEAD

#line 3

#line 4

                # description:  my very first go program

=======

# add line 1

# add line 2

# description:  my very first go program

>>>>>>> e7141045aae5d31c0b3db9ecff8eb777b1d09e50

                package main

                import "fmt"

                func main() {

                    fmt.Println("hello world")

}

* Remove lines you do not want
* Git add hello.go
* Git commit -m "fix merge"
* Git push
* Git status
* Look at file in git repository
  + You should see the latest changes

**Git Ignore**

* % cd to a repository of your choice
* Create a file you do not want to checkin
  + % vi my-secret-file
* Git status to show the file as "U" - untracked

Untracked files:

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

my-secret-file

* Edit a .gitignore file and add "my-secret-file"
* No need to source the .gitignore
* Git status to show the file is no longer recognized
* This .gitignore works in the current repository
* You can also create a "global" .gitignore
  + % cd $HOME
  + % vi .gitignore\_global
    - Add your rules
  + Add the global gitignore to your config (.gitprofile)
    - % git config --global core.excludesfile ~/.gitignore\_global
  + More .gitprofile

**End**