Engineering for everyday world

AceEngineer

GIT

Development Document

8th November 2017

19-Dec-2014	01	New Issue					VA	-	-
DATE	**REV**	**DESCRIPTION**					**ORIG**	**CHK**	**APPR**
DOCUMENT CONTROL NO		**Project**	**Type**	**Area**	**Client**	**-**	**-**	**Sequence**	**Revision**
		0026	**PY**	**-**	**-**	**-**	**-**	**0001**	**01**

Revision History:

REV	DATE	DESCRIPTION	ORIG	CHK	APPR
01	9th Dec 2014	Manual for python coding	MP	VA	VA

Change Log

REV	SECTION	CHANGE DESCRIPTION

Document Holds

Hold	DESCRIPTION
HOLD 01	

CONTENTS

1 INTRODUCTION

Git is a version control system (VCS) for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.

- Distributed among all computers. Loss of files or entire project is difficult
- All changes and history is stored in every computer

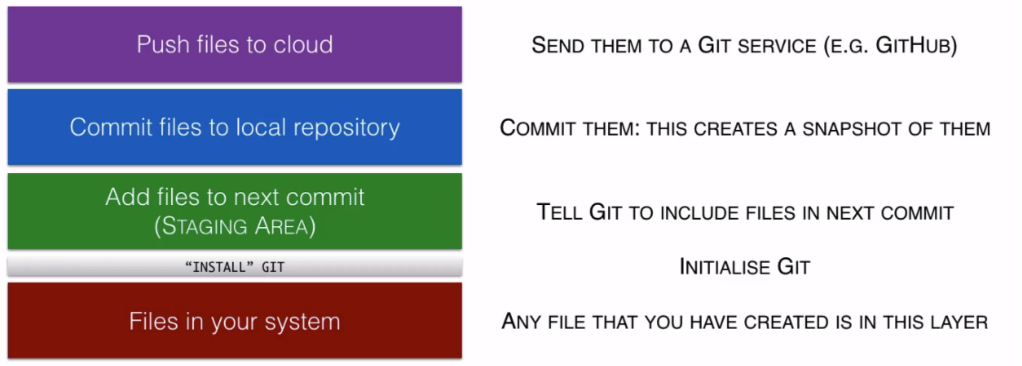
There are several tools that can be used to work in conjunction with git to make it easy to work in Git.

- Git Extension - Graphical interface. However, not all business requirements may be captured.
- Uses text editing tool (NotePad or BeyondCompare) to highlight key differences.

Git is a very important and popular tool used for code development process. There are several other tools available in the market such as TFS, Dropbox, OneDrive which handle everything.

2 BASICS

2.1 Understanding Git

- Git is set of layers
  - Filesystem
  - Initialize Git in a folder
  - Add files to the staging area (for next commit)
  - Commit files to local repository (creates a snapshot)
  - Host snapshots remotely (to a git service such as github, bitbucket etc.)
  - 
Push files to cloud
Send them to a Git service (e.g. GitHub)
Commit files to local repository
Commit them: this creates a snapshot of them
Add files to next commit (Staging Area)
Tell Git to include files in next commit
"INSTALL" GIT
Initialise Git
Files in your system
Any file that you have created is in this layer
- Each layer has a function
- Layers provide error control, redundancy and other features.
- Layers are physical (inside a hidden .git folder)
- Git calculates the difference between 2 versions and stores the difference.
- Git is used to move the files between each layer

2.2 Repository

Understanding repository

To get a summary of repository, branches, and status of remote and local branches, the following command is useful

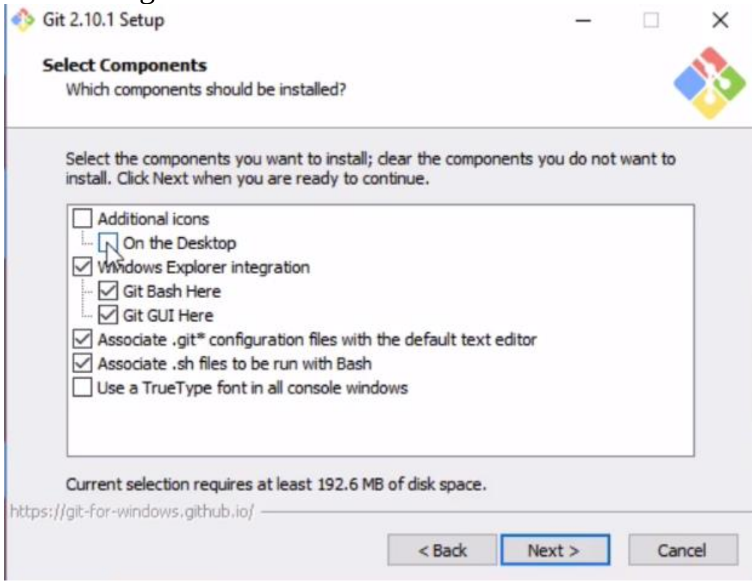
git remote show origin

2.3 References

https://realpython.com/python-git-github-intro/ A good introduction to the basics

3 SET UP

3.1 Git

- Download git:
    - https://git-scm.com/download/win
- Recommended settings are:
    - 

Select Components

Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [ ] Additional icons
    - [ ] On the Desktop
- [x] Windows Explorer integration
    - [x] Git Bash Here
    - [x] Git GUI Here
- [x] Associate .git* configuration files with the default text editor
- [x] Associate .sh files to be run with Bash
- [ ] Use a TrueType font in all console windows

Current selection requires at least 192.6 MB of disk space.

- For each unique project, create a new git repository.
- A visual git interface program:
    - Tortoisegit
    - https://tortoisegit.org/

4 GIT GENERAL WORKING

Commits are only saved on desktop.
Push will push the code to the remote server

4.1 Summary

The are given below:
A summary of the most commonly used commands are given in table below:

Category	Command	Description
	git status	Status of files on branch. Displays files modified and created.
	git checkout	
	git pull	
	git merge	
Branch - Local		
	git branch -v	View all branches
	Git branch –d branchName	Delete branch with name “branchName”
	Git branch	

Table 4.1 – Commonly Used Commands

4.2 Usage

The use of git can be done in one of the following ways:

- Git-bash
    - git-bash is purported to contain more features as opposed to in traditional windows command prompt.
- Windows prompt
- GUI
    - GUI can be used for simple operations with some success.

Project workflow structure should be defined prior to implementing Git to help simplify the initial implementation.

4.3 Command Line Help

- To display help on a particular topic
    - Git help keyword (or) Git keyword –help.
    - An example is given below
        - Git help config (or)
        - Git config –help

4.4 Initial Set-up

4.4.1 Configuring GIT environment

The configuration will contain all the parameters required to configure the user parameters for working.

- Configuration : Will list all the configuration properties
    - *git config --list*
- Configure User Name and User Email
    - *git config --global user.name "Vamsee Achanta"*
    - *git config --global user.email Vamsee_Achanta@gmail.com*
- Config command can also be used to set the following:
    - GUI tool
    - Merge tool
    - Difference comparison tool
    - Etc.

4.4.2 Initialization of Local Repository

To initialize a local repository of Existing Code as a git repository:
git init
Files can be tracked. Saved and committed to control the version.

To share code with a team, you need an origin (i.e a remote repository) to commit, push and integrate/share the code.

4.4.3 Add Ignore Files

To ignore the temporary files. A common map of temporary files to

git ignore

- Git ignore file.
- .gitignore file is a text file with files that can be ignored.
- For example in Python, ignore
    - *.pyc

Creating a .gitignore File:
.gitignore file in windows is not allowed. Follow below steps:

- Create a empty text file
- Rename the empty text file to .gitignore using command prompt

https://stackoverflow.com/questions/10744305/how-to-create-gitignore-file

4.4.4 Add or Remove Remote Origin

Add a remote origin repository.
git remote add origin http://tfs2015.idc.com:8080/tfs/orcaFlex/_git/dataInput

To remove remote origin
git remote remove origin http://tfs2015.idc.com:8080/tfs/orcaFlex/_git/dataInput

To display origin locations:
git remote -v

Example output:
```
origin  http://tfs2015.oxy.com:8080/tfs/orcaFlex/_git/dataInput/ (fetch)
origin  http://tfs2015.oxy.com:8080/tfs/ orcaFlex/_git/ dataInput/ (push)
```

4.4.5 Cloning of Remote Repository

Git clone is used to create a local copy of an existing repository. Git cone command can be used to clone an existing remote repository in to local repository.
Git clone

-
- Internally, git clone first calls git init to create a new repository.
- It then copies the data from the existing repository, and checks out a new set of working files.

Remove Directory from Git Tracking:

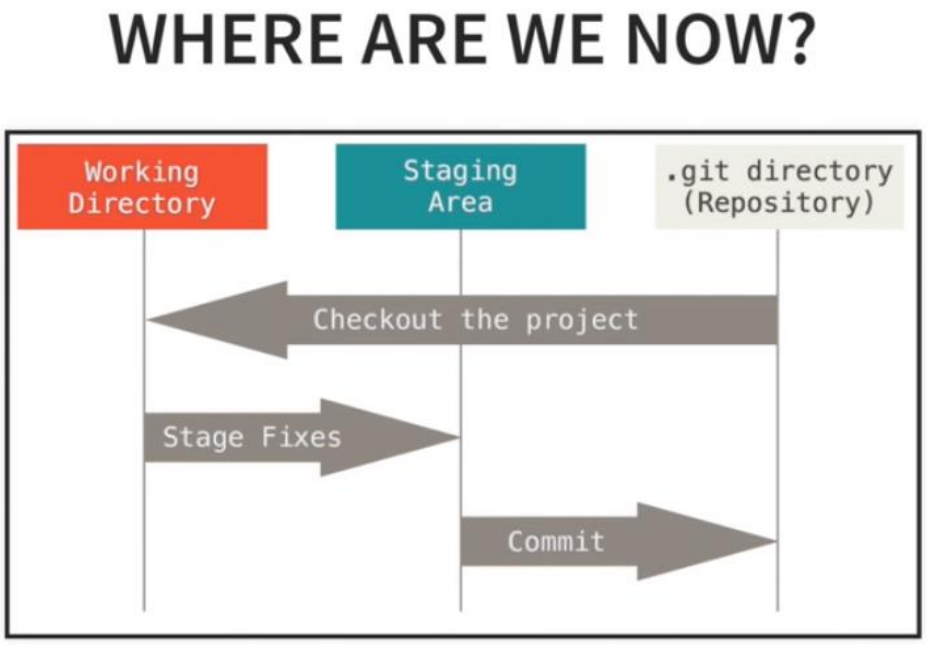
- If a directory should be removed from tracking, run below command to remove all related files:
    - *rm –rf .git*
    - This command removes the .git directory and associated files.

4.5 Work Stages

There 3 work stages:

- Working Directory
- Staging Area
- .git Directory (Repository)

There are 2 repositories. Local Repository. Remote repository. Both of these repositories will go through the above stages.



Git pull origin master
Git push origin master

4.6 Removing Files

4.6.1 Remove single file

- Remove single file:
    - *git rm –cached <file>*
- This is the opposite command of git add

4.6.2 Stash

Git stash will save the changed files and allow to change branch. Further investigation is needed to understand how the files are saved and how they can be retrieved.

https://git-scm.com/book/en/v2/Git-Tools-Stashing-and-Cleaning Git Stash Changes

4.7 Security

4.7.1 SSH

Public keys -

Private keys -

Generating a key in git-bash

- Generate ssh key
    - Ssh-keygen
- Enter or leave passprase blank
- The public key saved location willb e provided:
    - `/c/Users/vamsee.achanta/.ssh/id_rsa.pub.`
- Copy and add the public ssh key in github.com
    - Edit profile
    - Add SSH and GPG keys
        - Add a new SSH key
- Add the origin again (remove old one if exsiting) using ssh link

4.7.2 SSL Verify

Utilize no verify flag for git.

4.8 Git Documentation

- Vim readme.md
    - To insert text, press I key
- Utilizes markdown language
    - Title
        - # REST API
    - Subtitles
        - ## Description
        - ## Installation Instructions
    - Code blocks
        - ‘’’
        - Code goes here
        - ‘’’
- Essentially showcase projects for other team members or job recruiters etc.

4.9 Git Scripts

Good Morning

I have put up a bash script which can be used to pull latest changes (master) for all local feature branches. Below is how to set it up (one time only):

1. Copy the "git-update-feature-branches" to your local folder, e.g. "c:\bash"
2. Add the following folders to the "Path" environment variable
   - your installed Git's bin folder. This is where it locates on my box:
"C:\Users\tranq\AppData\Local\Programs\Git\bin"
   - the bash script folder you've just copied the file to, e.g. "c:\bash"
3. Then, run the following command from your favorite command prompt to update your feature branches.
   git update-feature-branches

If you want to rename the bash script file, it should be in the format "git-xxx", without file extension and with "xxx" being anything you want. Then, you run git "xxx".

4.10 Merging

4.10.1 Manual Merge

If merging is not done regularly, a tedious manual merge is needed

If 2 parties are working,

4.10.2 Squash Merge

https://docs.microsoft.com/en-us/azure/devops/repos/git/merging-with-squash?view=azure-devops

4.10.3 Rebase

Git Rebase. Always rebase the local branch that is behind master.

Working in PyCharm:

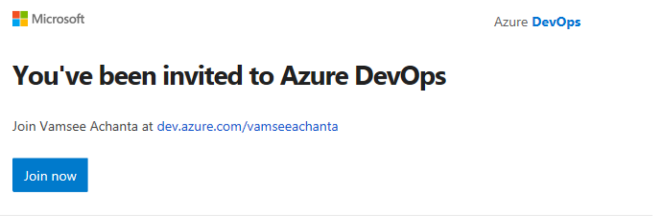
- Select the branch to rebase
- Click (do not select) on Master menu
- Click rebase current onto Selected (master).

5 AZURE

Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through a global network of Microsoft-managed data centers.

- The Azure devops repository owner will create a devops account
- The team members are invited y the owner

- Members then:
  - Invitation link come in the form of mail.
  - Clone the repository and start working



Microsoft

Azure DevOps

You've been invited to Azure DevOps

Join Vamsee Achanta at dev.azure.com/vamseeachanta

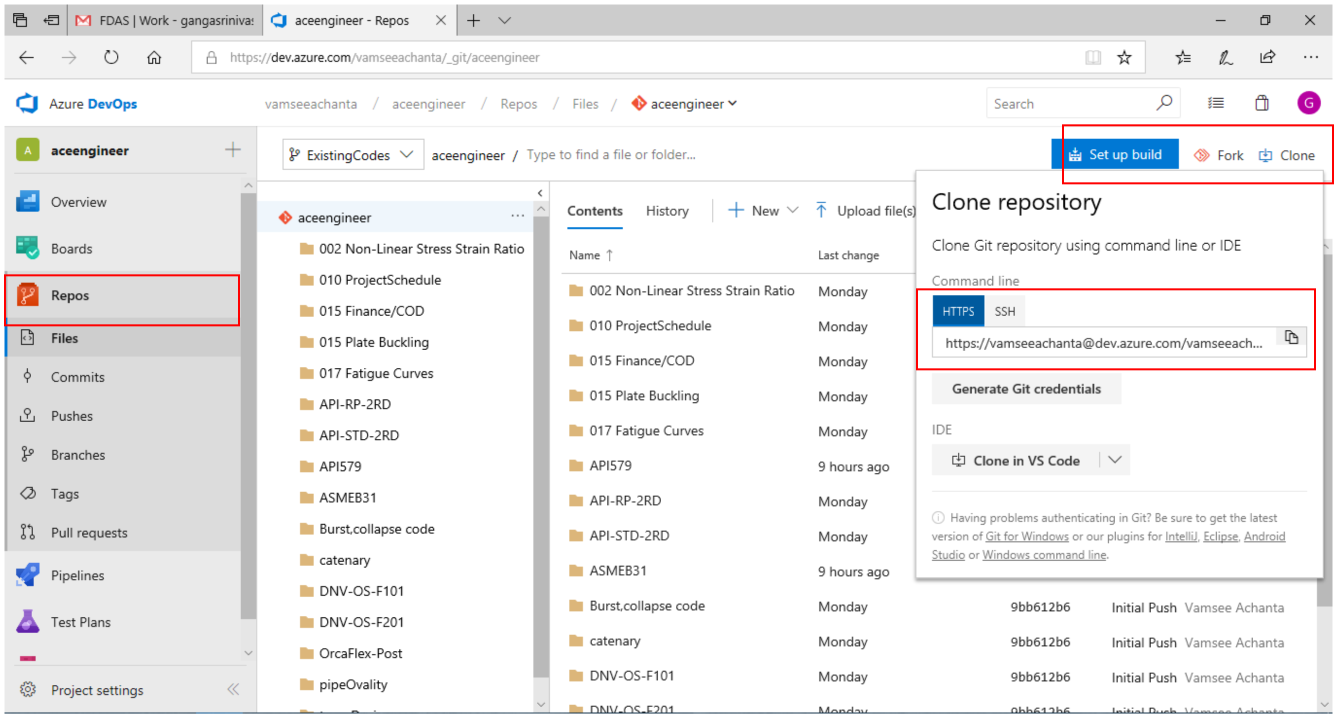
Join now

- Join the invitation by clicking on join option with using your account credentials.

5.1 Git Cloning

Git cloning instructions:

- Copy the repository link in git bash
  - $git clone https://vamseeachanta@dev.azure.com/vamseeachanta/aceengineer/_git/aceengineer



Name	Last change
002 Non-Linear Stress Strain Ratio	Monday
010 ProjectSchedule	Monday
015 Finance/COD	Monday
015 Plate Buckling	Monday
017 Fatigue Curves	Monday
API579	9 hours ago
API-RP-2RD	Monday
API-STD-2RD	Monday
ASMEB31	9 hours ago
Burst,collapse code	Monday 9bb612b6 Initial Push Vamsee Achanta
catenary	Monday 9bb612b6 Initial Push Vamsee Achanta
DNV-OS-F101	Monday 9bb612b6 Initial Push Vamsee Achanta

Clone repository

Clone Git repository using command line or IDE

Command line

HTTPS SSH

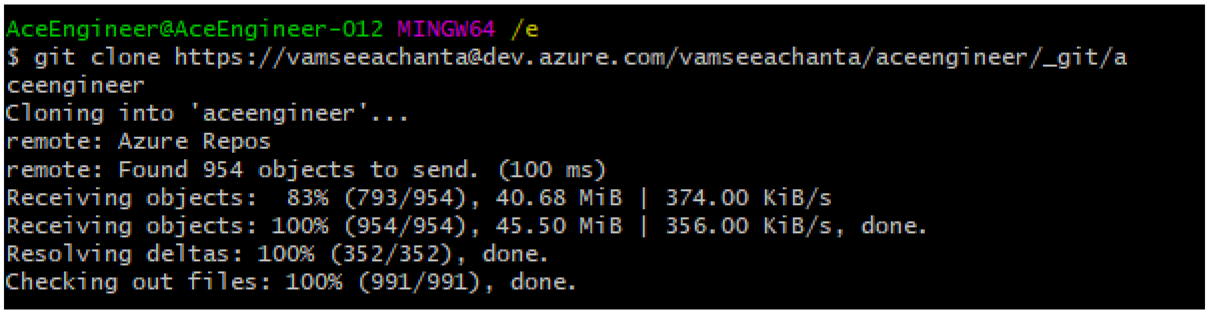
https://vamseeachanta@dev.azure.com/vamseeach...

Generate Git credentials

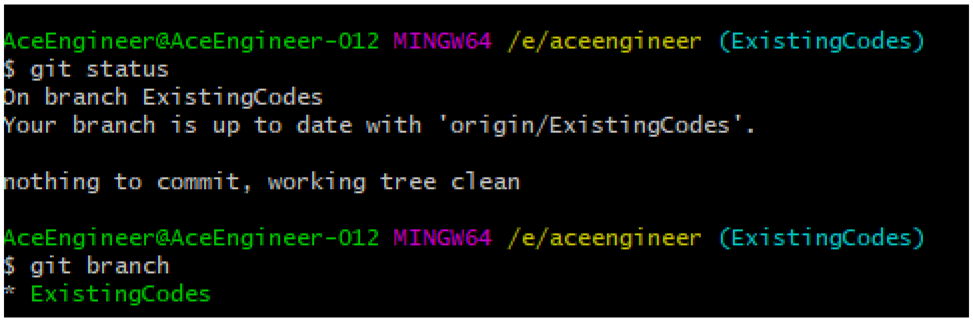
IDE

Clone in VS Code

Having problems authenticating in Git? Be sure to get the latest version of Git for Windows or our plugins for IntelliJ, Eclipse, Android Studio or Windows command line.



```
AceEngineer@AceEngineer-012 MINGW64 /e
$ git clone https://vamseeachanta@dev.azure.com/vamseeachanta/aceengineer/_git/a
ceengineer
Cloning into 'aceengineer'...
remote: Azure Repos
remote: Found 954 objects to send. (100 ms)
Receiving objects:  83% (793/954), 40.68 MiB | 374.00 KiB/s
Receiving objects: 100% (954/954), 45.50 MiB | 356.00 KiB/s, done.
Resolving deltas: 100% (352/352), done.
Checking out files: 100% (991/991), done.
```



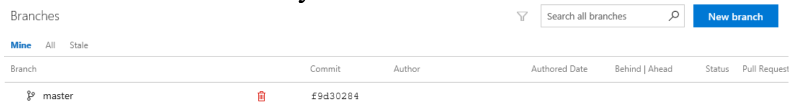
```
AceEngineer@AceEngineer-012 MINGW64 /e/aceengineer (ExistingCodes)
$ git status
On branch ExistingCodes
Your branch is up to date with 'origin/ExistingCodes'.

nothing to commit, working tree clean

AceEngineer@AceEngineer-012 MINGW64 /e/aceengineer (ExistingCodes)
$ git branch
* ExistingCodes
```

5.2 Typical WorkFlow – Code Development

The typical workflow followed in Azure Devops is given below:

- Pull the latest master code
- Start working on your local branch
    - Create a local branch (Never work in master on your computer)
        - *git branch <branchname>*
        - *git branch <ExistingCodes>*
        - or use GUI alternatively as below and create “New Branch”
        - 

Branch	Commit	Author	Authored Date	Behind | Ahead	Status	Pull Request
master	f9d30284					

    - Checkout the local branch
        - *Git branch (to view all the existing branches)*
        - *git checkout <branchname>*
        - *git checkout <ExistingCodes>*
    - Create files, edit codes per the project needs

When branch is ready for review (and/or approval):

- Push the branch to the origin
    - Commit the relevant files in following steps
    - Add the files:
        - *git add –A* (Add all files)
        - *git add ASMEB31\ASMEB31Sizing.py* (Add all files)

- Git commit -m "Initial Push"
- Push the files to the origin
  - *git push -u origin <branch>*
  - *git push -u origin <ExistingCodes>*
- Create a pull request
  - Go to Azure devops website
  - Create the pull request
  - *screenshot*
  - Copy the person who should be reviewing the code in the pull request.

5.3 Typical WorkFlow – Code Approval

TBA

6 TYPICAL WORKFLOWS

A typical workflow is given below:

- Code is being edited

6.1 WorkFlow – Existing Local Repository

Situation Description:

- Code is cloned from remote repository
- The code now exists in local drive.
- Git is to track all the local code changes
- After code update is complete, commit code to remote repository

Example Git workflow instructions are below:

- Create a branch
  - *Git branch calc-divide*
- Check out the branch
  - *Git checkout calc-divide*
- Update the code with all required changes
- Move Files : Add to staging area
  - *git add –A*
  - or add specific files using below command
    - *git add mycode.py*
- Commit changes to the local branch of git with comment in quotes.
  - *git commit –m "updated calculation method"*
- Move Files : Pushing branch changes to remote:

- *git push –u origin calc-divide*
- *git commit –a*
- TFS Instruction here: If TFS is the official system
  - Create a Pull Request to the team to merge branch into master by going to the browser.
  - *Git*
  - The team is now officially responsible to do the pull and merge into master
- Integrate the branch with master
  - *git checkout master*
  - *git pull origin master*
  - *git branch --merged*
  - *git merge calc-divide*
  - This will integrate the branch to the origin. The other person can see.

Move Files : Merge changes to master
  - *git push origin master*
  - Output will be as follows for a successful merge:
    - *Git branch –merged*
    - *Calc-divide*
    - **master*
- Delete local branch locally
  - *git branch –d calc-divide*
- To delete the remote branch
  - *Git push origin --delete calc-divide*
- To view the existing branch and associated comment:
  - *Git branch –v*

Need graphics for this.

6.2 WorkFlow – Existing Remote Repository

- Work on your changes and 'Commit' partial changes.
  - Note: The commit operation commits code locally and not to git.
- Finally, when all your commits are ready to be pushed:
  - Do a Pull to make sure if others worked on the same files as you committed.
  - If so, pull action will report conflicts
- Resolve the conflicts. As a developer you should resolve the conflicts before you push all your changes to git.
- Push all your changes to git.

Need graphics for this.

6.3 WorkFlow – Bug Fix Branch

Usecase Definition. When working on a branch, we may need to do bug fixes on existing master code. The recommended procedure is:

- Push previous working branch, branch1 to master (origin)
- Pull from master (origin)
- Create new branch, branch 2
- Do code changes and push branch 2 to master
- Change branch to branch1
- Pull the code
- Start working
- Submit branch 1 to master
    - May need to rebase code

Need graphics for this.

6.4 WorkFlow – Orphan Branch

Git Orphan Branch:
git checkout --orphan ExceptionFolder
git rm -rf .
To add fresh folder:
git add Exceptions/

https://stackoverflow.com/questions/15745045/how-do-i-resolve-git-saying-commit-your-changes-or-stash-them-before-you-can-me

https://stackoverflow.com/questions/1384325/in-git-is-there-a-simple-way-of-introducing-an-unrelated-branch-to-a-repository/4288660#4288660

6.5 WorkFlow – Most recent changes

Undo last commit and those changes locally: git reset --soft HEAD~1
Undo last commit and discard those changes: git reset --hard HEAD~1

6.6 Workflow - Commits

Undo all commits after target commit: git reset --hard (commit hash)

7 COMMON ERRORS

This section summarizes the common errors encountered and possible troubleshooting hints. If the error is not present in this section, Google for further help.

7.1 Error 1: Not a git repository (or any of the parent directories): .git

Solution: The command has to be entered in the directory of the repository. The error is complaining that your current directory isn't a git repo
- Are you in the right directory? Does typing ls show the right files?
- Have you initialized the repository yet? Typed git init? (See git-init documentation)

8 REMOTE REPOSITORIES

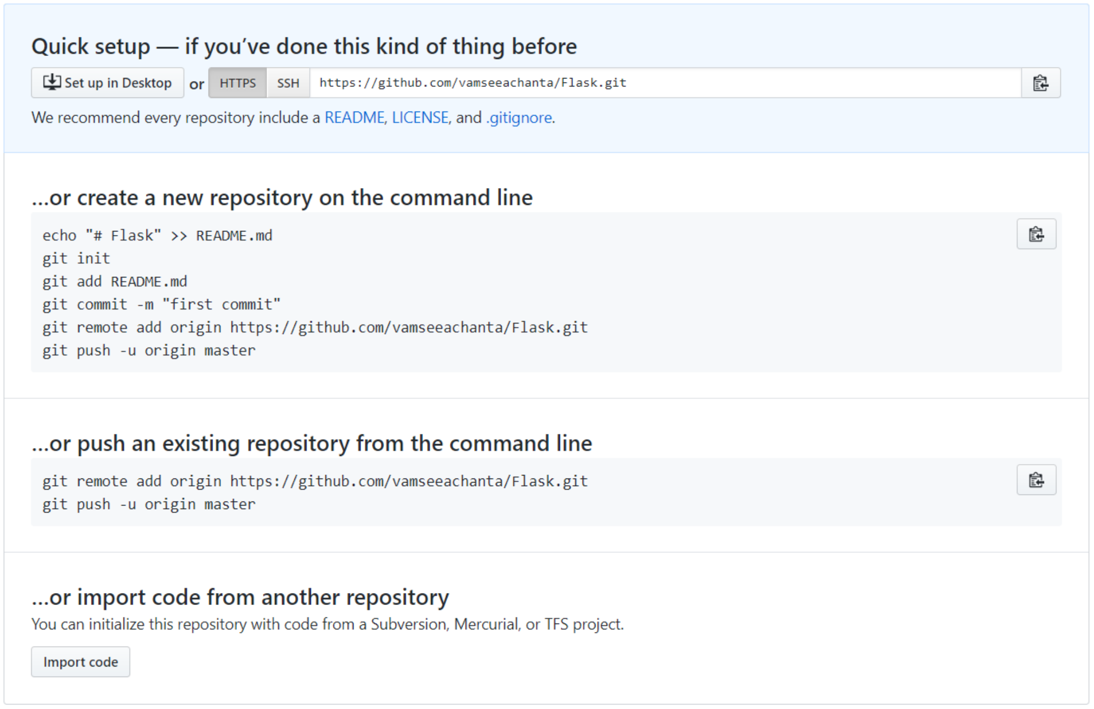
8.1 Github

The key features are:

- A free to use tool for public repos.
- A paid service for private repos

Creating a new repository using a project name of choice

Quick Setup Info:
See below



Quick setup — if you've done this kind of thing before

```
https://github.com/vamseeachanta/Flask.git
```

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# Flask" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/vamseeachanta/Flask.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/vamseeachanta/Flask.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

git remote add origin https://github.com/vamseeachanta/Flask.git
git push --set-upstream origin master
git push -u origin master

9 GIT EXTENSIONS – GUI

Using GUIs make Git easier. However, at some point, situations may arise when git is certainly required.

9.1 References

git-extensions-documentation.pdf
https://www.youtube.com/watch?v=TlZXSkJGKF8

10 SUBPROJECT

Subproject is defined a project within a project. This needs to be developed further to help save headaches.

Clone Subdirectory
Method 1 : To work with only a subdirectory of a repository, use sparseCheckout, [4]. See commands below:

```
$ mkdir pcl-examples
$ cd pcl-examples                                   #make a directory we want to copy folders to
$ git init                                  #initialize the empty local repo
$ git remote add origin -f https://github.com/PointCloudLibrary/pcl.git     #add the remote origin
$ git config core.sparsecheckout true               #very crucial. this is where we tell git we are checking out specifics
$ echo "examples/*" >> .git/info/sparse-checkout" #recursively checkout examples folder
$ git pull --depth=2 origin master                  #go only 2 depths down the examples directory
```

Method 2 :

git clone project/ subproject/
cd subproject
git filter-branch --prune-empty --subdirectory-filter dirB HEAD
Note that to do a subsequent pull, add --*depth 0* flag to the command.

Method 3 : Submodules can be used. See [5].
https://git-scm.com/book/en/v2/Git-Tools-Submodules

11 PROJECT DELIVERY

11.1 Microservices

https://smartbear.com/learn/api-design/what-are-microservices/ Microservices
https://opensource.com/resources/what-are-microservices

11.2 CI/CD

https://www.atlassian.com/continuous-delivery/ci-vs-ci-vs-cd

12 REFERENCES

No.	Description	Comment
[1]	https://www.youtube.com/watch?v=HVsySz-h9r4	A very good Git intro video
[2]	https://stackoverflow.com/questions/10744305/how-to-create-gitignore-file	Add .gitignore file
[3]	http://nvie.com/posts/a-successful-git-branching-model/	Branch Features and Production.
[4]	https://stackoverflow.com/questions/600079/how-do-i-clone-a-subdirectory-only-of-a-git-repository http://scriptedonachip.com/git-sparse-checkout	sparseCheckout
[5]	https://git-scm.com/book/en/v2/Git-Tools-Submodules	submodules
[6]	https://github.com/github/training-kit/blob/master/downloads/github-git-cheat-sheet.md	Github git cheatsheet

APPENDIX 1.0 – INSTALLATION

Check Installation and version
Git –version

If version shows up, this means that git is successfully installed.

https://git-scm.com/download/win

APPENDIX 2.0 – FILES TO IGNORE

Language	**File extension**	
Python	*.pyc	
TFS (Team Foundation server)	*.md	

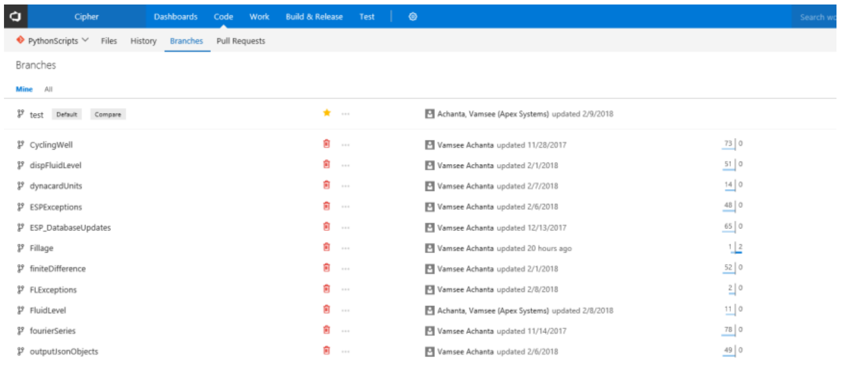
To access previous logs: git log

To exit the log file, type q and press enter.

APPENDIX 3.0 –TFS

3.1 Comparing Branches

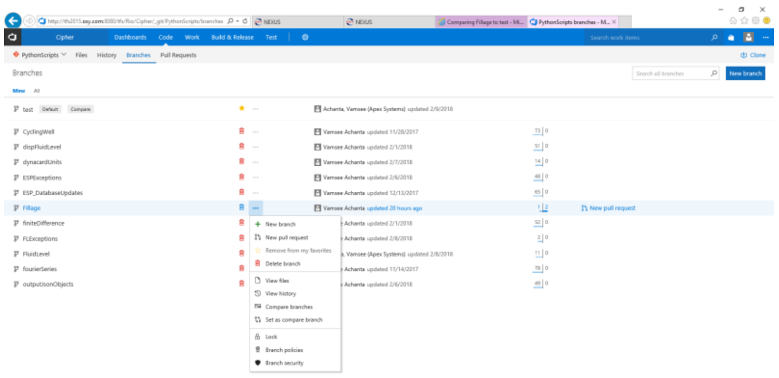
- Comparing a branch code to another branch. An example screenshot is shown below to compare the “Fillage” branch with “test/stage” below
- Click on Branches to view all branches:



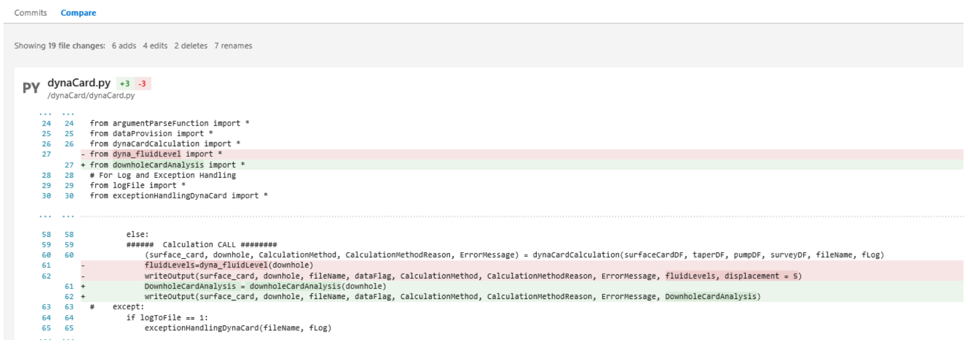
test	Achanta, Vamsee (Apex Systems) updated 2/9/2018	
CyclingWell	Vamsee Achanta updated 11/28/2017	73 | 0
dispFluidLevel	Vamsee Achanta updated 2/1/2018	51 | 0
dynacardUnits	Vamsee Achanta updated 2/7/2018	14 | 0
ESPExceptions	Vamsee Achanta updated 2/6/2018	48 | 0
ESP_DatabaseUpdates	Vamsee Achanta updated 12/13/2017	65 | 0
Fillage	Vamsee Achanta updated 20 hours ago	1 | 2
finiteDifference	Vamsee Achanta updated 2/1/2018	52 | 0
FLExceptions	Vamsee Achanta updated 2/8/2018	2 | 0
FluidLevel	Achanta, Vamsee (Apex Systems) updated 2/8/2018	11 | 0
fourierSeries	Vamsee Achanta updated 11/14/2017	78 | 0
outputJsonObjects	Vamsee Achanta updated 2/6/2018	49 | 0

-
- The default branch to compare is “test”

- On interested branch to compare, click on "More Actions" to expose options shown below.



-
- Click on "Compare Branches" to compare the changes
- Example screens showing the comparisons are given below:



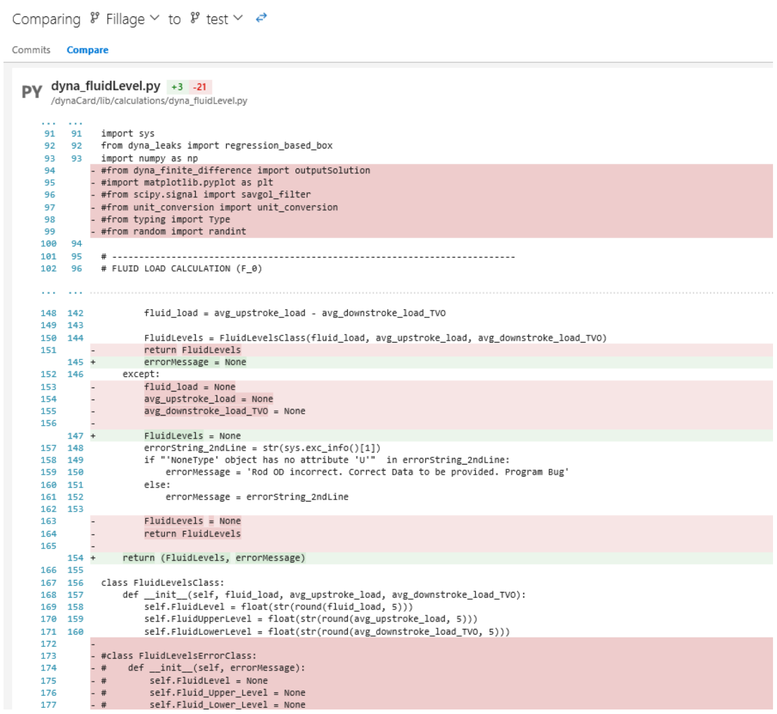
Showing 19 file changes: 6 adds 4 edits 2 deletes 7 renames

PY dynaCard.py +3 -3
/dynaCard/dynaCard.py

```
  from argumentParseFunction import *
  from dataProvision import *
  from dynaCardCalculation import *
- from dyna_fluidLevel import *
+ from downholeCardAnalysis import *
  # For Log and Exception Handling
  from logFile import *
  from exceptionHandlingDynaCard import *

          else:
          ######  Calculation CALL ########
              (surface_card, downhole, CalculationMethod, CalculationMethodReason, ErrorMessage) = dynaCardCalculation(surfaceCardDF, taperDF, pumpDF, surveyDF, fileName, fLog)
-             fluidLevels=dyna_fluidLevel(downhole)
-             writeOutput(surface_card, downhole, fileName, dataFlag, CalculationMethod, CalculationMethodReason, ErrorMessage, fluidLevels, displacement = 5)
+             DownholeCardAnalysis = downholeCardAnalysis(downhole)
+             writeOutput(surface_card, downhole, fileName, dataFlag, CalculationMethod, CalculationMethodReason, ErrorMessage, DownholeCardAnalysis)
  #    except:
          if logToFile == 1:
              exceptionHandlingDynaCard(fileName, fLog)
```

-


Comparing Fillage to test

Commits Compare

PY dyna_fluidLevel.py +3 -21
/dynaCard/lib/calculations/dyna_fluidLevel.py

```
   import sys
   from dyna_leaks import regression_based_box
   import numpy as np
 - #from dyna_finite_difference import outputSolution
 - #import matplotlib.pyplot as plt
 - #from scipy.signal import savgol_filter
 - #from unit_conversion import unit_conversion
 - #from typing import Type
 - #from random import randint

   # --------------------------------------------------------------------------
   # FLUID LOAD CALCULATION (F_0)

...

           fluid_load = avg_upstroke_load - avg_downstroke_load_TVO

           FluidLevels = FluidLevelsClass(fluid_load, avg_upstroke_load, avg_downstroke_load_TVO)
 -         return FluidLevels
 +         errorMessage = None
       except:
 -         fluid_load = None
 -         avg_upstroke_load = None
 -         avg_downstroke_load_TVO = None
 -
 +         FluidLevels = None
           errorString_2ndLine = str(sys.exc_info()[1])
           if "'NoneType' object has no attribute 'U'"  in errorString_2ndLine:
               errorMessage = 'Rod OD incorrect. Correct Data to be provided. Program Bug'
           else:
               errorMessage = errorString_2ndLine

 -         FluidLevels = None
 -         return FluidLevels
 -
 +     return (FluidLevels, errorMessage)

   class FluidLevelsClass:
       def __init__(self, fluid_load, avg_upstroke_load, avg_downstroke_load_TVO):
           self.FluidLevel = float(str(round(fluid_load, 5)))
           self.FluidUpperLevel = float(str(round(avg_upstroke_load, 5)))
           self.FluidLowerLevel = float(str(round(avg_downstroke_load_TVO, 5)))
 -
 - #class FluidLevelsErrorClass:
 - #    def __init__(self, errorMessage):
 - #        self.FluidLevel = None
 - #        self.Fluid_Upper_Level = None
 - #        self.Fluid_Lower_Level = None
```

•