PRACTICAL CS WORKSHOP #1

Qiaodan (Jordan) Zuo

Overview

- Workshop Series Introduction & Objectives
- Environment Setup (AWS EC2)
- Linux & Linux Utilities
- Git & GitHub
- Agile Development
- Python Development in Linux

Workshop Series Introduction

- 8~12 Workshops. Totally around 16~24 Hours
- Free & voluntary
- Tutorials for practical CS knowledge and tools
- Practice with real world examples
- Team up to finish industrial level projects

Workshop Series Topics

- CS Basic - Git, Agile, Linux etc.
- Advanced Python
- Data Structure & Algorithm
- Data Process, Cleaning, Visualization & Analysis
- Web Scraper & Database
- Other topics (optional): RabbitMQ, Kafka, Celery, Spark, etc.
- Career Advice (Resume, LinkedIn, Networking etc.)
- Capstone Projects

Objective

- Overcome the fear to Computer Science (coding, terminal etc.)
- Build foundation of Computer Science for further study
- Refine resume from a technical perspective (Keyword & Projects)
- Prepare for interview, summer projects & internship
- Broaden potential job opportunities

Environment Setup

- **Amazon Web Service (AWS):**
  - On-demand cloud computing platform by Amazon
- **Amazon Elastic Compute Cloud (EC2):**
  - A major part of AWS
  - Provide virtual machine rental service
- We will run our own Linux (Ubuntu) instance on AWS!

Environment Setup

- Create an AWS account:
  - AWS official site
  - Click for tutorial
- Launch Instance (*Ubuntu Server 18.04 LTS (HVM), SSD Volume Type*)
  - Go to instance site
  - Click **Launch Instance** -> Select **Ubuntu Server 18.04 LTS**
  - Select **t2.micro** (free for 12 months) -> **Review and Launch**
  - **Launch** -> **Create a new key pair** and give it a name
  - **Download Key Pair**
  - Be sure to remember the location of key pair file (***.pem***)

Environment Setup

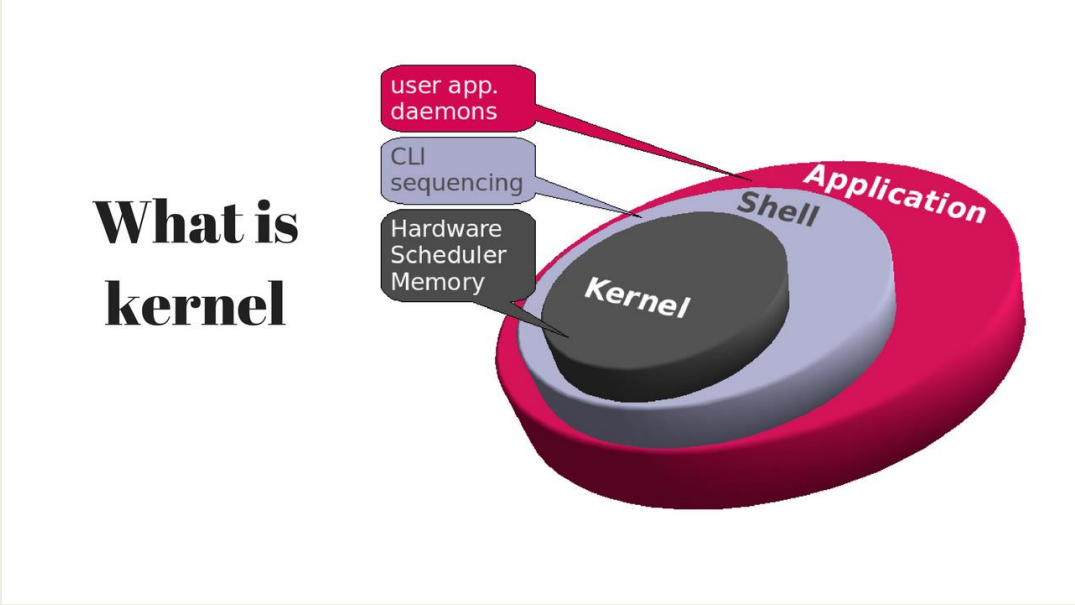
- Under **Instance** section, select **Instance** tab
- Find your instance and click **Connect** (the **Example** part is important)
- For Mac & Linux users Click for tutorial
  - Copy string from the 3rd point (start with `**chmod 400**`)
  - Copy string from **Example** (start with `**ssh -i** `), then run it in terminal
- For Windows users Click for tutorial
  - Download **putty.exe** and **puttygen.exe** from here
  - Open **puttgen.exe**, load the key pair file you got from EC2 (**.pem*)
  - Then, **Save Private Key**, be sure you know the location of it (you will have a **.ppk* file)
  - Back to **Example** from Instance page, copy string starting from `**ubuntu@**` all the way to the end
  - Open **putty.exe**, paste it under **Host Name**, at **Saved Session**, type "**ec2**", then click **Save**
  - On left side, find **Connect** -> **SSH** -> **Auth** -> **Browse**, find the **.ppk* file you saved before
  - Click **Open**

Time Out

- Now you just had your very first interaction with Cloud Computing & AWS
- Remember EC2 is customizable and scalable, you can extend it as you need
- Add **AWS** to your resume, you will have a better chance to be noticed by recruiters
- AWS is very popular among small/midsize companies
- Now, it's time for some Linux tutorial

Linux

- Linux is a family of Unix-like Operating System
- Linux vs Linux Kernel
- Linux Distributions (Distros)
  - Ubuntu (popular distro world wide)
  - Red Hat (popular commercial Linux)
  - Debian
  - Linux Mint
- Desktop Environment (GUI)
  - MATE
  - KDE
  - GNOME



Things to know about Linux

- Linux distros share lots of commands with MacOS, since they are both Unix-like OSs
- More than 90% of servers in the world are running Linux
- 100% of top 500 supercomputers are running Linux
- Android is the most popular Linux-based OS
- Ubuntu is the most popular Linux-based desktop OS
- The author of Linux Kernel, Linus Torvalds, also created Git version control system

Job Description Example

"Linux Experience" in job description:

- Experience working under Unix-like Operating Systems (commands, tools and projects)

Fixed Income Desk Strategist / Quantitative Developer at **Morgan Stanley**

Skills Required

Bachelor's degree or higher in computer science, engineering, physics or related discipline
Commercial experience of Q/KDB
Commercial experience of HTML5, JavaScript, AngularJS
Knowledge of C++, Java or Scala would be advantageous

Time series analysis and machine learning experience would be advantageous
Experienced in both Linux and Windows environments

Linux Commands

- Cheatsheet
- Most common Linux commands:
  - *ls, pwd, cd, rm, rmdir, mkdir, cp, mv, cat, touch, head, tail, less, chmod, cal, date, whoami, df, du, free, which, ping, clear, sudo*
- Let's Practice!

Redirect, Append & Pipe

- >: Redirect output to some file
  - `***echo "HELLO" > hello.txt***` *hello.txt* file now should have content "Hello"
  - `***2>&1***` *redirect standard error(2) to standard output(1)*
  - `***command >/dev/null***` *discard standard output(1) from command*
  - `***command 2>/dev/null***` *discard standard error(2) from command*
  - `***command >/dev/null 2>&1***` *discard both standard output(1) & error(2)*
- >>: Append output to some file
  - `***echo "World" >> hello.txt***`
  - *hello.txt* file now should have content "Hello\nWorld"
- |: Take output from left side and use as input for command on right
  - `***cat hello.txt | grep "llo"***`
  - Command above will find the line has "llo", hence display "Hello"

find

- Recursively find files, which contains "file-name" in file name under current directory
  - `` `find . -name "*file-name*" ` ``
- Find files, which contains "abc-" in file name under */dev* directory (not its subdirectory)
  - `` `find /dev -maxdepth 1 -name "abc-*" ` ``
- Documentation

grep

- Display line with word ‘echo’ in file test.txt
  - `***grep 'echo' text.txt***`
- Display all line & line# with exact word “echo” in file a directory (recursively)
  - `***grep -rnw 'echo' text.txt***`
  - *-r: Recursively*
  - *-n: Line Number*
  - *-w: Exact word only (doesn’t match substring)*
- Documentation

Other Linux Utilities

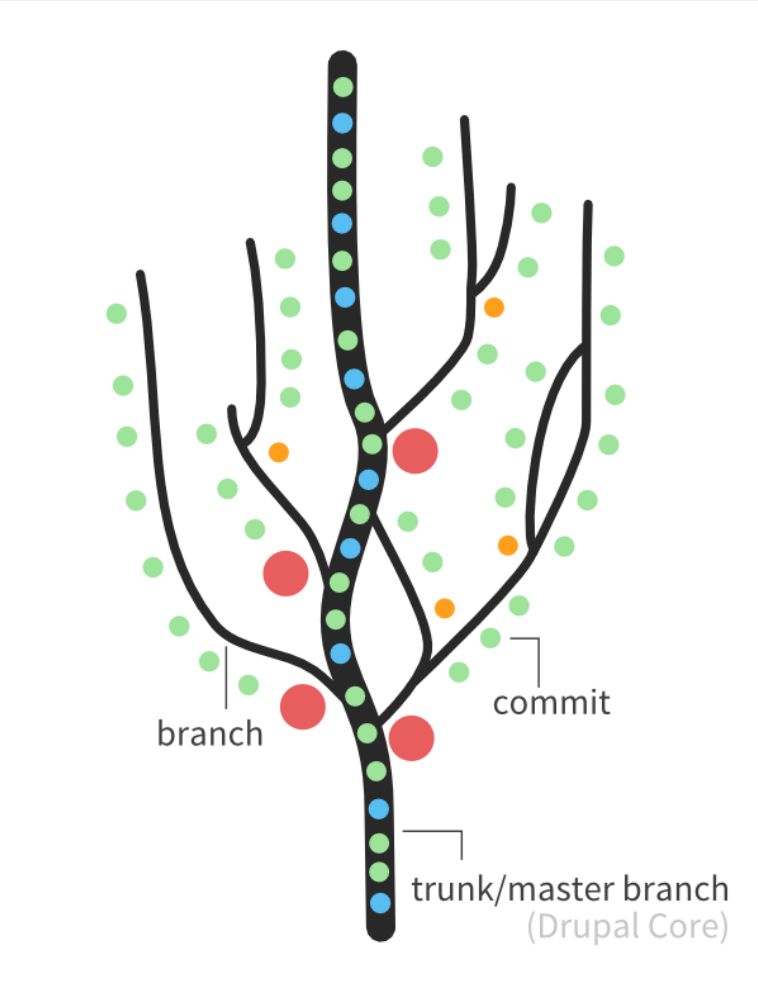
- Vi / Vim
- Tmux
- Bash/Zsh
- sed
- awk
- git
- ...

Practice Makes Perfect

- Practice and repeat, then you can have **Linux** on your resume
- There are tons of other commands/tools in Linux, you are welcome to explore more

Git vs GitHub

- **Git** is a distributed version control system for tracking changes in source code during software development
- **GitHub** is a website that provides repository hosting service
- **Git** is a tool, **GitHub** is a website for projects that use Git

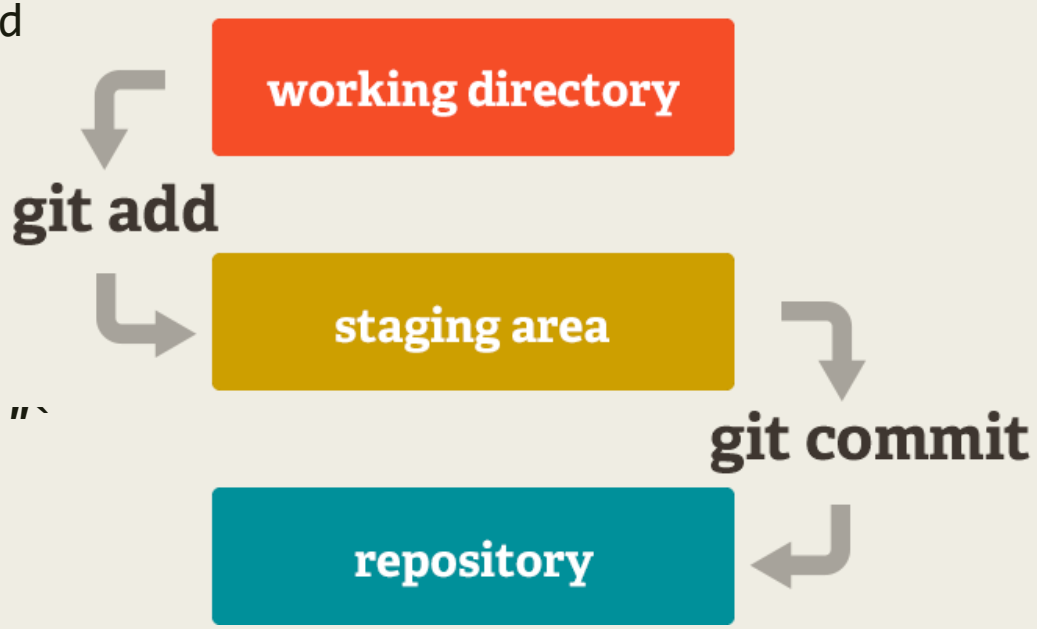


Git

- **Repository (Repo)**: A directory contains project source and git histories. Imagine a folder with Git histories and features.
- **Local**: All source and git history in your local machine
- **Remote**: Oppose to **Local**, source and git history hosted *remotely* by service like GitHub (like a cloud)
- **Branch**: A develop line that is independent to other
  - ***master*** branch: Main develop line by default (can be changed)
  - ***develop*** branch: A separate branch from master named “develop”

Git

- Working directory: You local code
- Staging area: Code that ready to be committed
- Commit: Ask git to record change
- Push: Update remote repository
- Workflow:
  - Add to staging area: `***git add .***`
  - Record change: `***git commit -m "add files"***`
  - Update remote repo: `***git push***`



Setup Git Credential

- Click for tutorial
- Create SSH key:
  - `ssh-keygen -t rsa -b 4096 -C your github email@xxx.com`
- Press Enter 3 times
- Start SSH agent and add SSH key
  - `eval $(ssh-agent -s)`
  - `ssh-add ~/.ssh/id_rsa`
- Set SSH key in GitHub
  - `cat ~/.ssh/id_rsa.pub` copy the output
  - Paste it here and give your key a name you like

```
ubuntu@ip-172-31-95-207:~$ ssh-keygen -t rsa -b 4096 -C "qzuo@fordham.edu"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa.
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:VjYXzkcRUziP2pgwOMHkcrdGVfcxAC4SyPaar9y8i7s qzuo@fordham.edu
The key's randomart image is:
+---[RSA 4096]----+
|    . .+.  .+oBBo|
|     + .+ .+ ooo+|
|    . o.o+*.+ .+.|
|       ++=++ .. .|
|      o S.oo =   |
|     o . .  + .  |
|      .          |
|    . +.         |
|     E+=o        |
+----[SHA256]-----+
ubuntu@ip-172-31-95-207:~$ eval $(ssh-agent -s)
Agent pid 1092
ubuntu@ip-172-31-95-207:~$ ssh-add ~/.ssh/id_rsa
Identity added: /home/ubuntu/.ssh/id_rsa (/home/ubuntu/.ssh/id_rsa)
```

```
ubuntu@ip-172-31-95-207:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCYTJyzB9I9PJZd0xX4WDrEE2q8afjBHMQm1s
+Bu5QiC2C0zh801wGP0UKIx9FDyWwm0LKkDepFDqo6gRqvoMbOAz9OfXjKVbC8H06IWmGXQ4TyPF7v
+6zkIIFEGVmeQU9r6ltsqdKWj3j/iGShU1SaEDM9izNnhTzU91N2A1XL2t7yXC6jGYlJWsGIPHUiEio9XwrfpCT7lkRNghSu34W
b7DRo+U0MF7YnoSIjd2XG/mYUpkzhvY6UGavz8euw4OLWN
+Ov3Syd1/XTXyqn15mMD0P6biBg7GXIlHT6wahj5UFgUxffrw4zMBxYeB94rm3AI7IThy2Wc3IjPbXEbwfk1fU31vjgKr22FXEc
9rHSI+bmaDroblbxSm1ch4hoQLVqcHK43i5Gee5B9O9o5WdIKrBCf8uDO8b+9PVz3joce0IT/NQ
+ApqUpFEUeW2gX7xkE2IhPqcSMmWygCBlD18J07wGv95pIzISloZSI/ub+hZQ8nC3Das
+CxQqSHrKnQbmNAuQm5W2H4JgkGyltIJ0hwMlzNUN2h9LHuzQmX2tARiCYK3zY/nOj+qjbM4DmgQVPJvas1FiH
+1wVmNC4aKge/gEx9ywmCdhjVdaQyDSvLoFPCioi09DyXwjLLfLT6KGBnFsAyNWzwXnKu0EIc8RsUJAqLrfjM/Q5kojkaKJYQ==
qzuo@fordham.edu
```

SSH keys / Add new

Title

EC2

Key

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQCYTJyzB9I9PJZd0xX4WDrEE2q8afjBHMQm1s+Bu5QiC2C0zh801wGP0UKI
x9FDyWwm0LKkDepFDqo6gRqvoMbOAz9OfXjKVbC8H06IWmGXQ4TyPF7v+6zkIIFEGVmeQU9r6ltsqdKWj3j/iGShU
1SaEDM9izNnhTzU91N2A1XL2t7yXC6jGYlJWsGIPHUiEio9XwrfpCT7lkRNghSu34Wb7DRo+U0MF7YnoSIjd2XG/mYU
pkzhvY6UGavz8euw4OLWN+Ov3Syd1/XTXyqn15mMD0P6biBg7GXIlHT6wahj5UFgUxffrw4zMBxYeB94rm3AI7IThy2
Wc3IjPbXEbwfk1fU31vjgKr22FXEc9rHSI+bmaDroblbxSm1ch4hoQLVqcHK43i5Gee5B9O9o5WdIKrBCf8uDO8b+9PVz
3joce0IT/NQ+ApqUpFEUeW2gX7xkE2IhPqcSMmWygCBlD18J07wGv95pIzISloZSI/ub+hZQ8nC3Das+CxQqSHrKnQ
bmNAuQm5W2H4JgkGyltIJ0hwMlzNUN2h9LHuzQmX2tARiCYK3zY/nOj+qjbM4DmgQVPJvas1FiH+1wVmNC4aKge/
gEx9ywmCdhjVdaQyDSvLoFPCioi09DyXwjLLfLT6KGBnFsAyNWzwXnKu0EIc8RsUJAqLrfjM/Q5kojkaKJYQ==
qzuo@fordham.edu

Add SSH key

Git Scenario #1 - Standalone

- Say you create a repository on GitHub and you want to work on this repo on you own
- Workflow:
  - Find SSH/HTTP remote URL for this repo on GitHub
    - **git@github.com:your_username/repo_name.git**
  - Clone to local:
    - `**git clone** git@github.com:your_username/repo_name.git`
  - Add your information (always do this when you have a new repo):
    - `***git config --local user.name "your_github_username"***`
    - `***git config --local user.email your_github_email@xxx.com***`
  - Now you can use `***add, commit, push***` as long as you are working under this directory

```
ubuntu@ip-172-31-95-207:~$ cat ~/.ssh/id_rsa.pub
ubuntu@ip-172-31-95-207:~$ git clone git@github.com:qz-fordham/project-zero.git
Cloning into 'project-zero'...
The authenticity of host 'github.com (192.30.253.113)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,192.30.253.113' (RSA) to the list of known hosts.
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 11 (delta 2), reused 7 (delta 1), pack-reused 0
Receiving objects: 100% (11/11), done.
Resolving deltas: 100% (2/2), done.
ubuntu@ip-172-31-95-207:~$ cd project-zero/
ubuntu@ip-172-31-95-207:~/project-zero$ git config --local user.email "qzuo@fordham.edu"
ubuntu@ip-172-31-95-207:~/project-zero$ git config --loca user.name "qz-fordham"
```

```
ubuntu@ip-172-31-95-207:~/project-zero$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
ubuntu@ip-172-31-95-207:~/project-zero$ touch test_file
ubuntu@ip-172-31-95-207:~/project-zero$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

        test_file

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

```
ubuntu@ip-172-31-95-207:~/project-zero$ git add .
ubuntu@ip-172-31-95-207:~/project-zero$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   test_file

ubuntu@ip-172-31-95-207:~/project-zero$ git commit -m "add test file"
[master 42e53e0] add test file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test_file
ubuntu@ip-172-31-95-207:~/project-zero$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

```
ubuntu@ip-172-31-95-207:~/project-zero$ git push
Counting objects: 3, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 303 bytes | 303.00 KiB/s, done.
Total 3 (delta 0), reused 1 (delta 0)
To github.com:qz-fordham/project-zero.git
   8a5d923..42e53e0  master -> master
```

Git Scenario #2 - Collaboration (part 1)

- Say you and your teammates are working on same repo (create by other), you want to make some change but not affect master branch before you are ready
- Workflow:
  - ***Fork*** the repo you want to work on from repo page
    - Fork means that you are making a copy of a repo in your account
  - Repeat **clone** & **config** setup in scenario #1

Git Scenario #2 - Collaboration (part 2)

- Make a new branch and checkout:
  - `***git checkout -b feature-branch-name***`
  - Make your code change
  - *Repeat* `***add, commit, push***` step in scenario #1 (push to new branch, not ***master***)
  - Setup remote to original repo: `***git remote add upstream https://original_repo_url.git***`
  - Switch to forked master branch: `***git checkout master***`
  - Make forked master sync with remote (upstream) master: `***git pull upstream master***`
    - Forked Master is up to date with remote upstream master
  - Switch back to feature branch: `***git checkout feature-branch-name***`
  - Merge forked master in your feature branch: `***git merge master***`
  - Go to the repo website and make a ***Pull Request (PR)***
  - Wait for other team member to review your code
  - Once your PR is approved, you can merge the PR and close it

- Original repo is from *git@github.com:**fordham-msqf-official**/project-zero.git*
- Now forked to *git@github.com:**qz-fordham**/project-zero.git*

```
ubuntu@ip-172-31-95-207:~$ git clone git@github.com:qz-fordham/project-zero.git
Cloning into 'project-zero'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 13 (delta 3), reused 8 (delta 1), pack-reused 0
Receiving objects: 100% (13/13), done.
Resolving deltas: 100% (3/3), done.
ubuntu@ip-172-31-95-207:~$ git config --local user.email "qzuo@fordham.edu"
fatal: --local can only be used inside a git repository
ubuntu@ip-172-31-95-207:~$ cd project-zero/
ubuntu@ip-172-31-95-207:~/project-zero$ git config --local user.email "qzuo@fordham.edu"
ubuntu@ip-172-31-95-207:~/project-zero$ git config --local user.name "qz-fordham"
```

```
ubuntu@ip-172-31-95-207:~/project-zero$ git checkout -b develop
Switched to a new branch 'develop'
ubuntu@ip-172-31-95-207:~/project-zero$ touch test_file
ubuntu@ip-172-31-95-207:~/project-zero$ git add .
ubuntu@ip-172-31-95-207:~/project-zero$ git commit -m "add test_file to develop branch in forked repo"
On branch develop
nothing to commit, working tree clean
ubuntu@ip-172-31-95-207:~/project-zero$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
ubuntu@ip-172-31-95-207:~/project-zero$ git pull
Already up to date.
ubuntu@ip-172-31-95-207:~/project-zero$ git checkout develop
Switched to branch 'develop'
ubuntu@ip-172-31-95-207:~/project-zero$ git merge master
Already up to date.
ubuntu@ip-172-31-95-207:~/project-zero$ git push origin develop
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:      https://github.com/qz-fordham/project-zero/pull/new/develop
remote:
To github.com:qz-fordham/project-zero.git
 * [new branch]      develop -> develop
```

fordham-msqf-official / project-zero

Watch 0 | Star 0 | Fork 1

Code | Issues 0 | Pull requests 0 | Projects 1 | Wiki | Insights

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base repository: fordham-msqf-official/project-... | base: master | head repository: qz-fordham/project-zero | compare: develop

✓ Able to merge. These branches can be automatically merged.

Create pull request Discuss and review the changes in this comparison with others.

Your part is finished temporarily

- Now, your teammates will review your code change and justify if they are good enough to be merged in main develop line (**master branch**)
- If your code is not good enough, you can keep working on your local code and repeat `**add, commit, checkout master, pull, checkout feature-branch, merge master, push**` until your PR is approved by peers
- To demonstrate review process, let’s switch to the GitHub account of the original repo owner (***fordham-msqf***)

fordham-msqf-official / project-zero

Watch 0 | Star 0 | Fork 1

Code | Issues 0 | Pull requests 1 | Projects 1 | Wiki | Insights | Settings

Develop #1

Edit

Open qz-fordham wants to merge 3 commits into `fordham-msqf-official:master` from `qz-fordham:develop`

Conversation 0 | Commits 3 | Checks 0 | Files changed 1

Changes from all commits | File filter... | Jump to... | +0 −0

Diff settings | Review changes

0 [redacted] test_file

No changes.

Write | Preview

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

- Comment
  Submit general feedback without explicit approval.
- Approve
  Submit feedback and approve merging these changes.
- Request changes
  Submit feedback that must be addressed before merging.

Submit review

Add more commits by pushing to the **develop** branch on **qz-fordham/project-zero**.

Changes approved

1 approving review by reviewers with write access. Learn more.

Show all reviewers

Continuous integration has not been set up

Several apps are available to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request

You can also open this in GitHub Desktop or view command line instructions.

fordham-msqf-official / project-zero

Watch 0 | Star 0 | Fork 1

Code | Issues 0 | Pull requests 0 | Projects 1 | Wiki | Insights | Settings

Tutorials of practical Computer Science knowledge & tools for Quants

Edit

Manage topics

8 commits | 1 branch | 0 releases | 1 contributor | MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

qz-fordham and fordham-msqf Develop (#1) ...

Latest commit c0054b6 19 seconds ago

```
* test

* rm test

* add test file
```

LICENSE	Initial commit	3 days ago
README.md	Initial commit	3 days ago
test_file	Develop (#1)	19 seconds ago

When you made a mistake

git reset & git rm –cached

- To undo git add:
  - *`**git reset .** `*
- To unto git commit:
  - *Un-stage changes `**git reset --soft HEAD~1**`*
  - *Un-stage & discard changes `**git reset --hard HEAD~1**`*
- Remove file/directory in remote repo but keep it in local:
  - *`**git rm --cached file_name**`, then commit*
  - *`**git rm -r --cached directory_name**`, then commit*

Solve conflict & management

git diff & git log

- Compare your local un-added change
  - `*git diff file_name*`
- Compare change since last commit
  - `*git diff*`
- Compare two branches
  - `*git diff branch_1…branch_2*`
- Compare file from two branches
  - `*git diff branch_1 branch_2 file_name*`
- Check git log, show one-line version of log, graph and branches
  - `*git log --graph --oneline --decorate*`

Switch branches in between work

git stash

- If your current branch has changes and you don’t want to commit yet, but you need to switch to other branches, use `***git stash***` to store you change in a **Stack**(*LIFO*) like structure
  - *Stash un-committed code* `***git stash***`
  - *List all stashes* `***git stash list***`
  - *Show difference between stash and local* `***git stash show***`
- When you done with your work in other branch and you want to keep working on previously stashed code
  - *Recover latest stash in current branch* `***git stash pop***`
  - *Same as above* `***git stash pop stash@{0}***`
  - Recover the 3rd most recent stash `***git stash pop stash@{2}***`

.git & .gitignore

- **.git** is a hidden directory under your local repo directory, it contains all git tracking information, including config, commit etc. If you delete .git, your local repo will no longer be tracked by git
- **.gitignore** is a hidden file under your local repo directory, it excludes files & dirs, which you don’t want to commit to remote repo. It’s good to have and always remember to put **.gitignore** in itself.

More about Git & GitHub

- Git Documentation
- GitHub collaboration control

Team Project #1 – Git & GitHub

- Create your own GitHub account
- Team up with at least 2 others (recommend 3-5 in total)
- Download team-project-1.py from project zero
- 1 team member create a repository with “team-project-1.py”
- Once repo is created, proceed to Team Project #2
- Don’t work on project #1 before setup is done for project #2
- Setup your local working directory with Git
- Finish task described in that file with steps in “Scenario #2”

Practice Makes Perfect

- Team up and practice, then you can have **Git** on your resume
- Something nice to have (things companies will like!):
  - A highly active GitHub account with meaningful code/projects
  - A personal GitHub page to show case your projects
  - Keep using Git for your personal projects

Agile Development

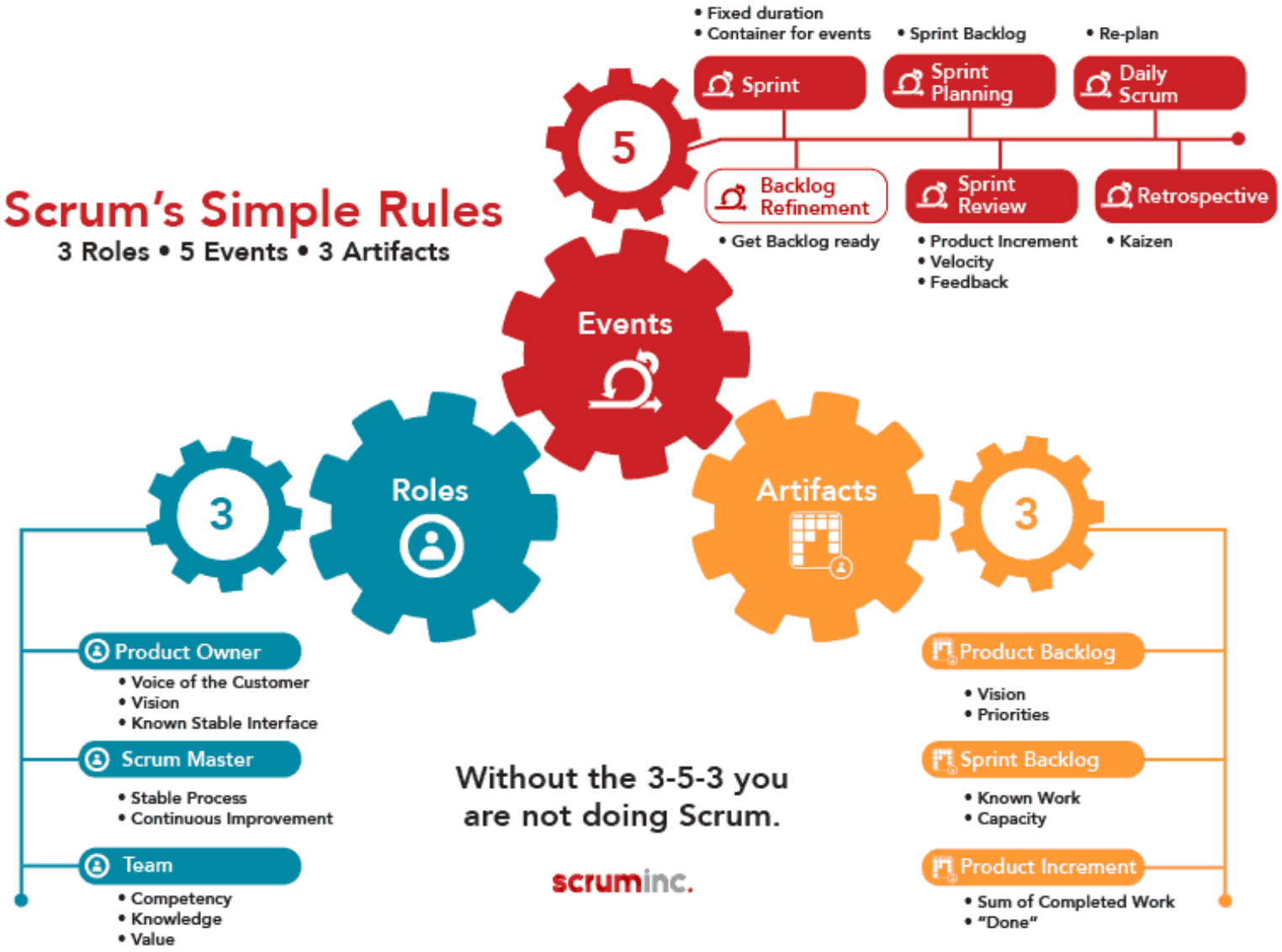
- “Agile Software Development is **an approach to software development** under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s).”

  --- From Wikipedia

- **Scrum**: An Agile framework (practice), widely used in industry
- **Kanban**: Another main stream Agile framework
- Silicon Valley Scrum scene

Scrum's Simple Rules

3 Roles • 5 Events • 3 Artifacts



Without the 3-5-3 you are not doing Scrum.

scruminc.

Roles in Scrum

- **Product Owner**
  - Represent product’s stakeholders and voice of the customers
  - Responsible for product backlog and deliverables
  - Bridge between team & customers; 1 PO per Scrum team
- **Scrum Master**
  - Help a Scrum team to reduce distractions (meetings/calls)
  - Ensure and guide team to follow Scrum processes
- **Development Team**
  - Engineers who bring actual value to deliverable products
  - A Scrum team normally has 3-9 developers

Scrum Workflow

Sprint	•Basic unit of Scrum development cycle •2-3 weeks/Sprint
Sprint Planning	•Team discussion for planning the upcoming Sprint. •2-4 hours/Sprint •Clear scope and detail so that team agrees to goal and understand tasks
Backlog refinement	•Clear uncertainties to get prepared for Sprint Planning. •2-4 hours/Sprint
Daily Scrum (stand up)	•Daily review of individual progress for current Sprint. •5-20 mins/day
Sprint Review	•Review complete/un-complete work for finished Sprint. •2-4 hours/Sprint
Retrospective	•Reflect on finished Sprint. More/Neutral/Less (Good/Ok/Bad). •Help team improve collaboration and efficiency continuously •1-2 hours/Sprint

Scrum Artifacts

- **Project Backlog**
  - A priority list of product expected tasks
- **Sprint Backlog**
  - Tasks development team must address during the next Sprint
- **Product Increment**
  - Potential deliverable items/features
  - PI should be completed/tested by the end of a Sprint

Scrum Board

- GitHub project board – GitHub native board
  - Customizable and free, very nice tools for team projects
- JIRA – Proprietary development tracking board
  - Powerful but not free, widely used in industry
- Other Scrum Board

J.P.Morgan Dashboards Projects Issues Boards Portfolio Create Search

As part of our Continued Service Resiliency efforts, All JIRA Services will be failed over to Carlstadt on coming weekend. Services will remain unavailable from 25th Jan (21:30 EST) to 26th Jan (11:30 AM EST). For any queries, please reach out to Service Desk

ICP Virtual Machine Service

VMS Sprint 19.02

3 days remaining Complete Sprint | Board

QUICK FILTERS: Japan | Audit | Prod V1 | Boilermaker | Tech_Debt | BELV-Softlaunch | Retro Action Items | Reported Externally | Only My Issues | Recently Updated

To Do

ICPVMS-1471
PROD - BKUP nic supplied and it shouldn't be from Magister to VMS
VMS Prod Support 1

ICPVMS-1509
EPV AIM: move to the new EPV vault.
VMS Prod Support 3

ICPVMS-1525
Confirm VMS monitoring process

ICPVMS-1353
DevOps/BMS - Retag cluster to have ST_IDENTIFIER=MULTI_TENA
VMS PSF 0

ICPVMS-1505
ITRC Break SSH Encryption needs sha1 removed.
VMS Risk / Breaks 1

In Progress

ICPVMS-163
Able to search by mac on the network interfaces admin page
VMS Technical Debt 2

ICPVMS-1347
Ensure BMS nightly sync works with Duplicate Data Centers
VMS PSF 3

ICPVMS-1473
Requirement for Test VMs
VMS Non-Prod Support 1

ICPVMS-1370
VMS PSF - DIT Story Tracker
VMS PSF 5

ICPVMS-1514
DIT: MDM returning 500 Server Errors
VMS Non-Prod Support 1

ICPVMS-1520
PROD: Different clusters have

In Dev

ICPVMS-1349
Add the new Tenant Field name to VM Create
VMS PSF 5

ICPVMS-1351
Send Tenant constraint to the TBN Placement call
VMS PSF 5

ICPVMS-1382
Hyperlink to cluster parent in read-only VM admin
VMS Technical Debt 1

In DIT

ICPVMS-1523
Map the ALL tenant type to MULTI_TENANT
VMS PSF 1

ICPVMS-1352
Update AZ API to include Tenant information
VMS PSF 5

ICPVMS-1249
Updated Validation check for Sophia Builds
VMS Technical Debt 2

Done

~~ICPVMS-1429~~
Fix SSAP Scan Flags
VMS Risk / Breaks 2

~~ICPVMS-1348~~
wps/create API changes for disks.
VMS BUGS 3

~~ICPVMS-125~~
As 'Q', I want to automatically delete VMs that fail during build.
VMS CleanUp 5

~~ICPVMS-140~~
As 'Q' I want to fix deleteVM_workflow code to initialize task_id before use
VMS CleanUp

~~ICPVMS-921~~
MVP - Repave - Create new WF
VMS Repave

~~ICPVMS-1527~~

Virtual Machine Service / ICPVMS-1352

Update AZ API to include Tenant information

Edit | Comment | Assign | More | Open | In Progress | Workflow | Export

Details

Type:	Story
Status:	IN SIT
Priority:	Medium
Resolution:	Unresolved
Affects Version/s:	None
Fix Version/s:	None
Labels:	VmsApiV4
Story Points:	5
Epic Link:	VMS PSF
Sprint:	VMS Sprint 19.01, VMS Sprint 19.02

Description

Update AZ API (AZ GET) to include Tenant

- Return tenant field in the list
- decouple tenant from service

Currently we return service: ["Gold", "WEBCO"], It should be:

Service:["gold"]
tenant: ["WEBCO"]

Sample JSON:
{ "platform" : ["ICP-V2"], "service" : ["DEDICATED", "DEDICATED.CLUSTER", "NONPROD", "PROD"], "network_zone" : ["ESF"], “tenant” : [“ALL”, “PSF”] }

,
"data_center" : "10030",
"links" : [

People

Assignee:
QIAODAN ZUO

Reporter:
GAURAV CHOUDHARY

Votes:
0 Vote for this issue

Watchers:
3 Stop watching this issue

Dates

Created:
29/Nov/18 2:47 PM

Updated:
6 hours ago

Development

2 branches
Updated Yesterday
25 commits
Latest Yesterday
3 pull requests MERGED
Updated Yesterday

Open Source Project Management Examples

- VUE.js – Frontend Framework
- VS Code – Text Editor/IDE
- TensorFlow – Machine Learning Framework

Agile Master!

- To have **Agile/Scrum** on your resume, you need practice Scrum workflow with your team. Don’t be Lazy!
- Things to remember:
  - Scrum roles
  - Scrum artifacts
  - Scrum workflow

Team Project #2 – GitHub Project Management

1. Repo owner in *Proj #1* should create project board from GitHub for your team repo.
2. Create 3 **issues** in team repo, one for each question in Proj #1. Include label, detail, AC, point, assignee, milestone.
3. **Product owner**: The repo owner.

   Scrum Master: Voluntarily.

   Development Team: All rest members.
4. Normally Product Owner & Scrum Master is not involved in development work, but for this project, both should
5. Perform Scrum workflow as a team.

   All 3 issues should be done within 1 sprint.
6. Perform:

   Planning

   Backlog Refine

   Daily stand up

   Review

   Retrospective
7. **Acceptable Condition (AC)**: All 3 issues are closed and related code works are done

Python Development in Linux

- Pip
- Virtual Environment
- Text Editor
- IDE (Jupyter Notebook, PyCharm)

Pip

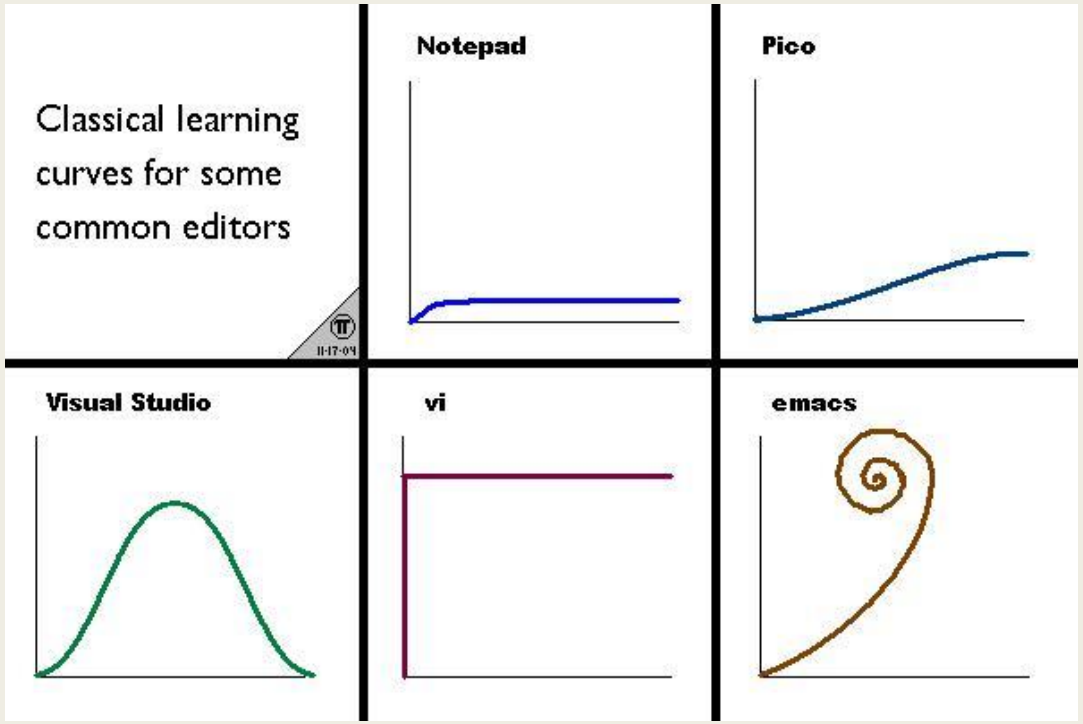
- Package management system for Python
- Install package using pip (you might need `***sudo***` before following commands to give pip superuser permission)
  - `***pip install package-name***`
  - `***pip3 install package-name***` *install package for Python3, when both Python2 and Python3 exist in your OS*

Virtual Environment

- A tool that helps to keep dependencies required by different projects separated by creating isolated python virtual environments for them.
  - Isolation
  - Portable
- ***Anaconda*** (*Recommend*): Powerful package manager & virtual environment
- ***Virtualenv***: Light-weight virtual environment for Python

Text Editor

- Vi/Vim: Ancient editor people can't give up
  - *Vi comes with all Linux distros*
  - *Extremely efficient in terms of editing*
  - *Vim is highly customizable*
  - *Most popular editors/IDEs have embedded "Vim Mode", which allows you to use Vim key bindings*
- VS Code: Microsoft opensource project
- Atom: Highly customizable modern text editor with beautiful GUI
- Sublime Text: Popular light-weight editor



Integrated Development Environment (IDE)

- **Text Editor**
  - Use text editor for small projects
- **Jupyter Notebook**
  - Interactive Python IDE
  - Nice tool for learning, taking notes, research and visualization
- **PyCharm/Intellij**
  - Industrial level IDE from **JetBrains** used by most professional
  - IDEs are good at **testing**, **tracking** and **handling dependencies**
  - Intellij is majorly used by Java developers, but it has plug-ins that support Python related functionalities
  - PyCharm is Python native IDE
  - Student can request for free version from JetBrains

My Python Development Setup

- Student at Fordham
  - *Ubuntu, Jupyter Notebook, Vim, Sublime Text, pip*
- Front-desk Developer at Hedge Fund
  - *Red Hat, Jupyter Notebook, Vim, PyCharm, virtualenv*
- Backend Developer at Bank
  - *Windows, Red Hat, Intellij, Vim, conda, pip*
- Personal Project
  - *Ubuntu, Jupyter Notebook, Vim, PyCharm, conda, pip*

Todo List

- Setup your Linux Environment with EC2
- Team up and set up your Git & GitHub repository
- Team lead should start a project board
- Work on 2 projects together and perform Scrum workflow
- Setup Pip, Conda for further development
- Practice

What we learned today?

- Amazon Web Service (AWS) EC2
- Linux & Utilities
- GitHub & Git
- Agile Development & Scrum
- Python Development tools (Pip)

Ready to Go!

- Review what we learned today and keep practicing with your team
- Update your resume once you feel confident
- **Practice! Practice! Practice!**
- Setup your development environment and understand knowledge is essential for our further workshops

Questions?