GIT important commands:

Cloning using git:

- git clone https://github.com/WilmingtonGroup/ComplianceWeek.git

Create the branch and dont push to master branch:

- git checkout -b compdev\_branch

- git push origin compdev\_branch

Branch will be seen in github.com

Push from local server to branch and not to dev(master)

git config --global credential.helper wincred

https://help.github.com/articles/caching-your-github-password-in-git/

Entering your username

git config --global user.name "Vanitha Sophia"

entering your email

git config --global user.email vanitha.sophia@cognizant.com

git status(red)

git add --all

git status(green)

git commit -m "hjg"

git push origin compdev\_branch

find the error log

tail -f staging.compliance.week.com

Step 1: Get a clone of GIT repository to local machine

git clone <download link from GITHUB>

Step 2: Create a branch in local machine

git checkout -b <branch name>

Step 3: Push the code to GIT

git push origin <branch name>

Step 4: SSH the server and create the branch

git fetch && git checkout <branch name>

Errors like:

1. error: Your local changes to the following files would be overwritten by checkout:

docroot/sites/staging.complianceweek.com/settings.php

Please, commit your changes or stash them before you can switch branches.

Aborting

* **git stash**
* **git status**

1. On branch staging

Your branch is behind 'origin/staging' by 22 commits, and can be fast- forwarded.

(use "git pull" to update your local branch)

Untracked files:

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

info.php

* **rm info.php**
* **git fetch && git checkout comp\_branch**

Create branch and commit changes (Important)

1. Branch should be created from master

Git checkout master

1. Pull the changes from git master to local master

Git pull origin master

1. Step again

Git pull origin master

1. Create the new branch

Git checkout –b branch\_name

1. Git add, git commit, git push
2. In ssh dev server,
3. Git fetch && git checkout <branch\_name>

For existing branches which doesn’t have the latest pull

1. Switch to ‘master’ local

Git checkout master

1. Pull the latest changes to local master

git pull origin master

1. Checkout to your branch

Git checkout <branchname>

1. git merge --no-ff master

DELETE option

1. Git fetch && git checkout <branch\_name>

Rebase issue to be solved:

1. If rebase removes your files
2. Delete the branch from local

Git checkout master

Git branch –D <branch\_name>

1. Git checkout <branch\_name>

Hardware details:

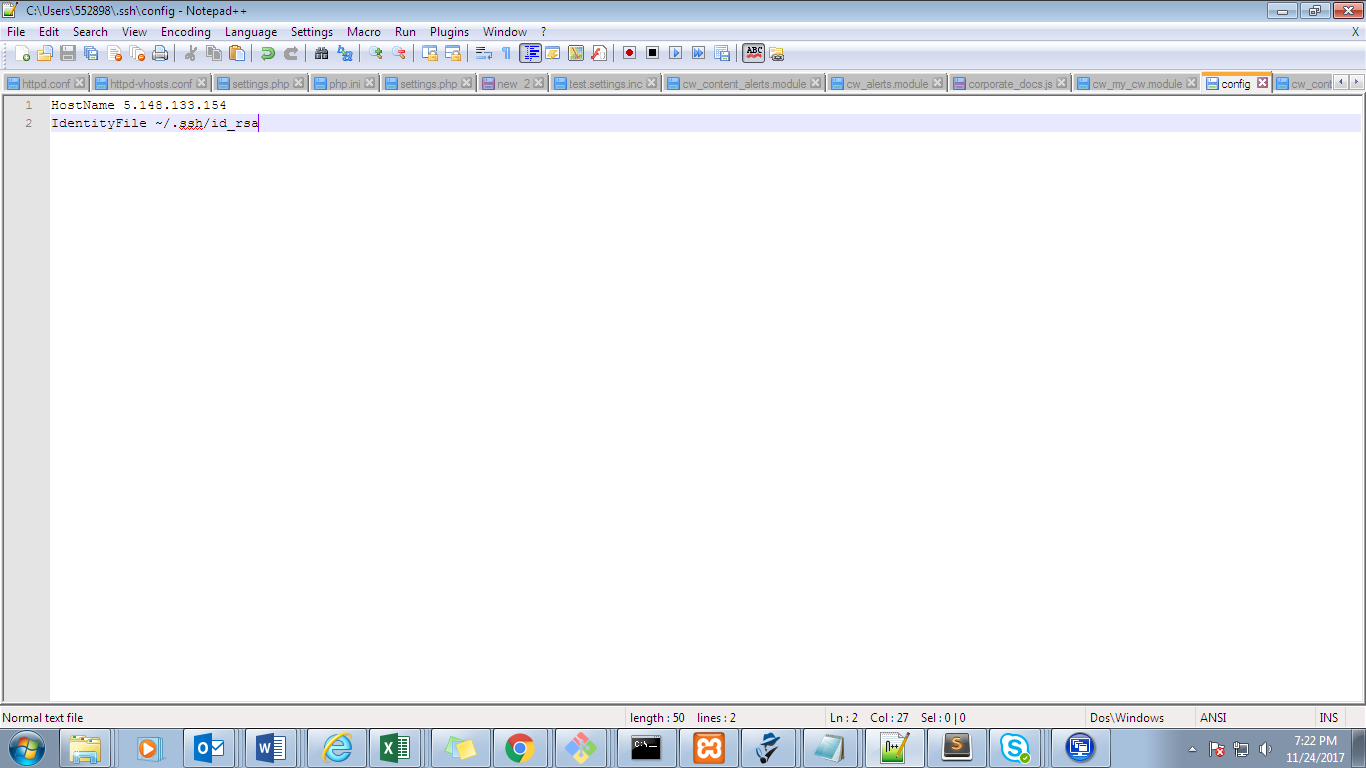
* Ubuntu 16.04
* They're actually all cloud based so they're configured from VM and there's 4 VM CPUs   
  Different RAM on the staging to the Live production servers   
  6G on staging and 18G on prod
* The main difficulty is that there are three nodes with a load balancer
* Each node has its own server, apache ssl, solr, memcache   
  and there's a load balancer that distributes hits
* It's running php fpm 5.6

Git important commands

1. Create the file in .ssh/config

HostName 5.148.133.154

IdentityFile ~/.ssh/id\_rsa

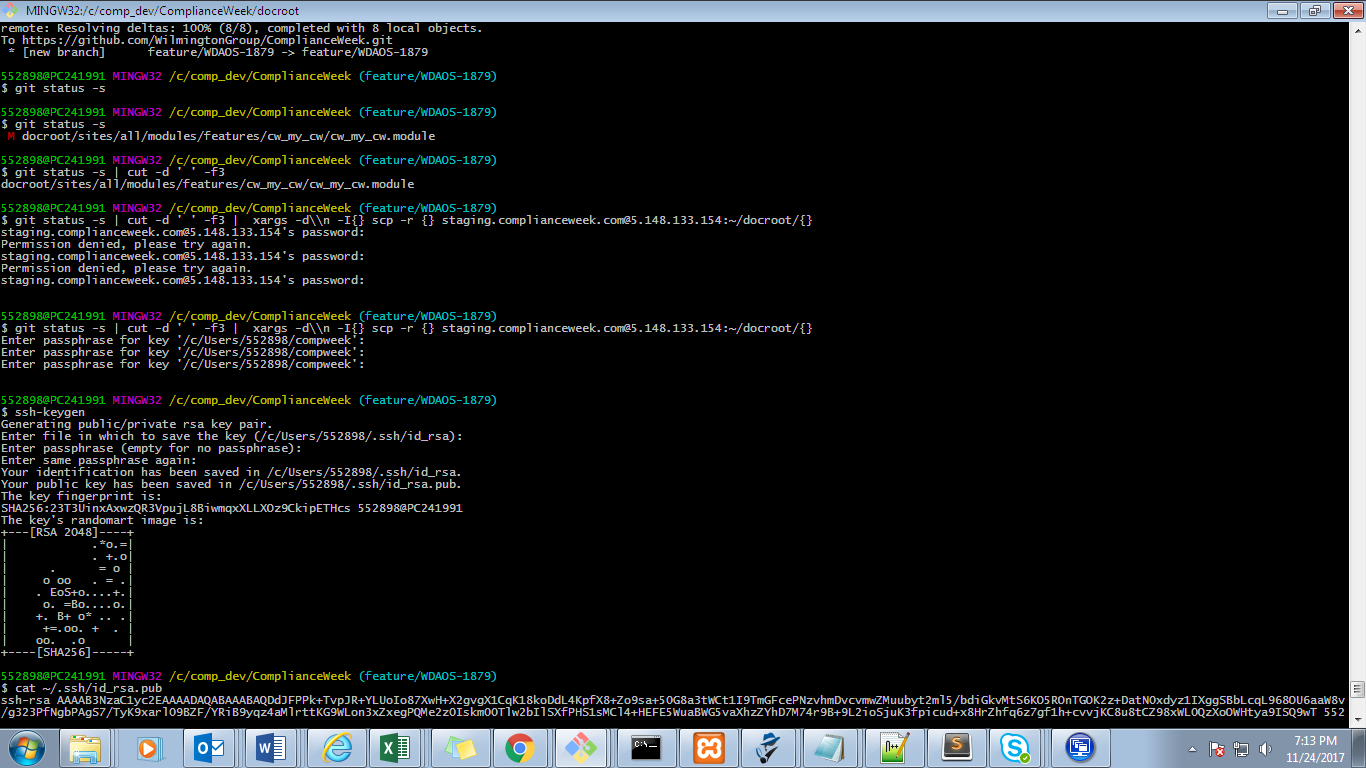


git status -s | cut -d ' ' -f3 | xargs -d\\n -I{} scp -r {} [staging.complianceweek.com@5.148.133.154:~/docroot/{}](mailto:staging.complianceweek.com@5.148.133.154:~/docroot/%7b%7d)

git status -s | cut -d ' ' -f3 | xargs -d\\n -I{} scp -r {} [eventsone@5.148.133.154:~/docroot/{}](mailto:eventsone@5.148.133.154:~/docroot/%7b%7d)

git status -s | cut -d ' ' -f3 | xargs -d\\n -I{} scp -r {}  [dev-evo8.hsj.co.uk@5.148.133.154:~/docroot/{}](mailto:eventsone@5.148.133.154:~/docroot/%7b%7d)

**Command used to upload the files in the server and can be committed at the end of the all valid files added.**



SSH key without passphrase

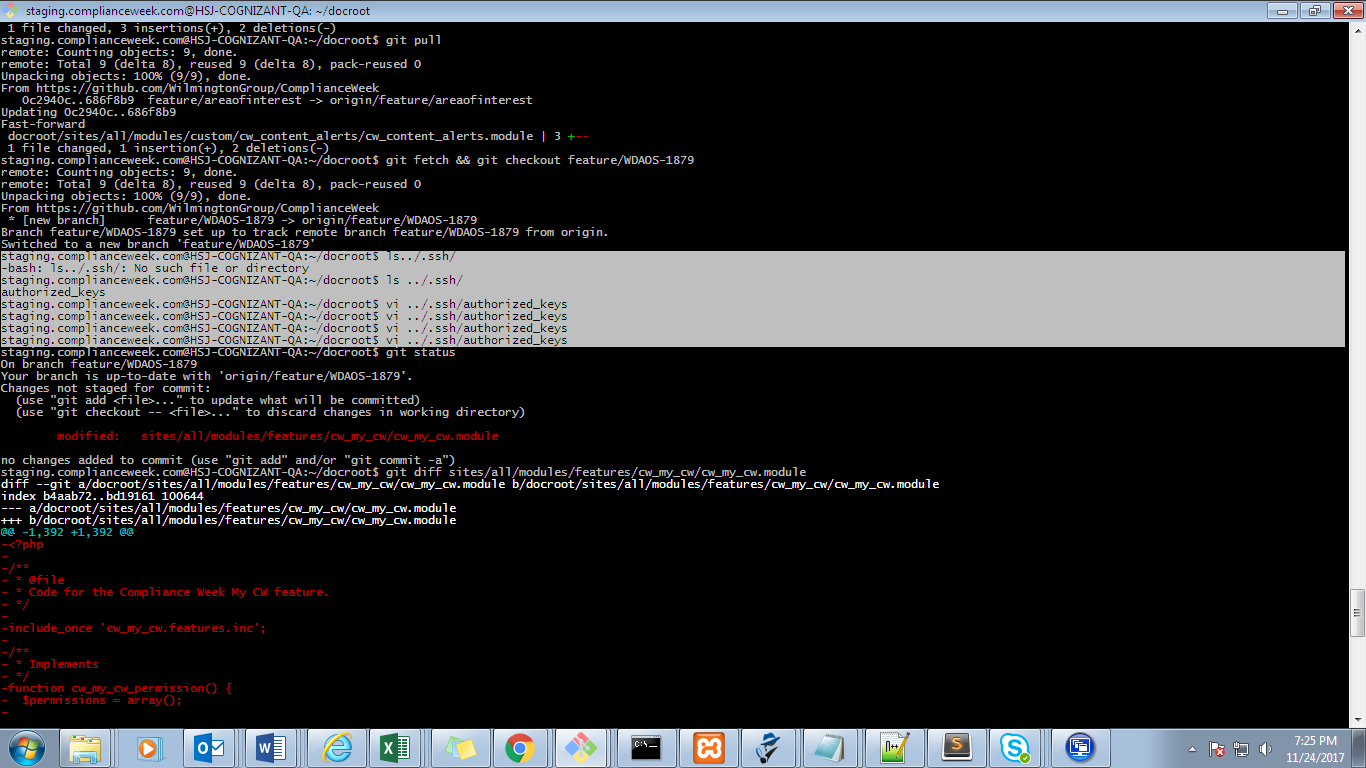
ssh-keygen

* **Generating the key**

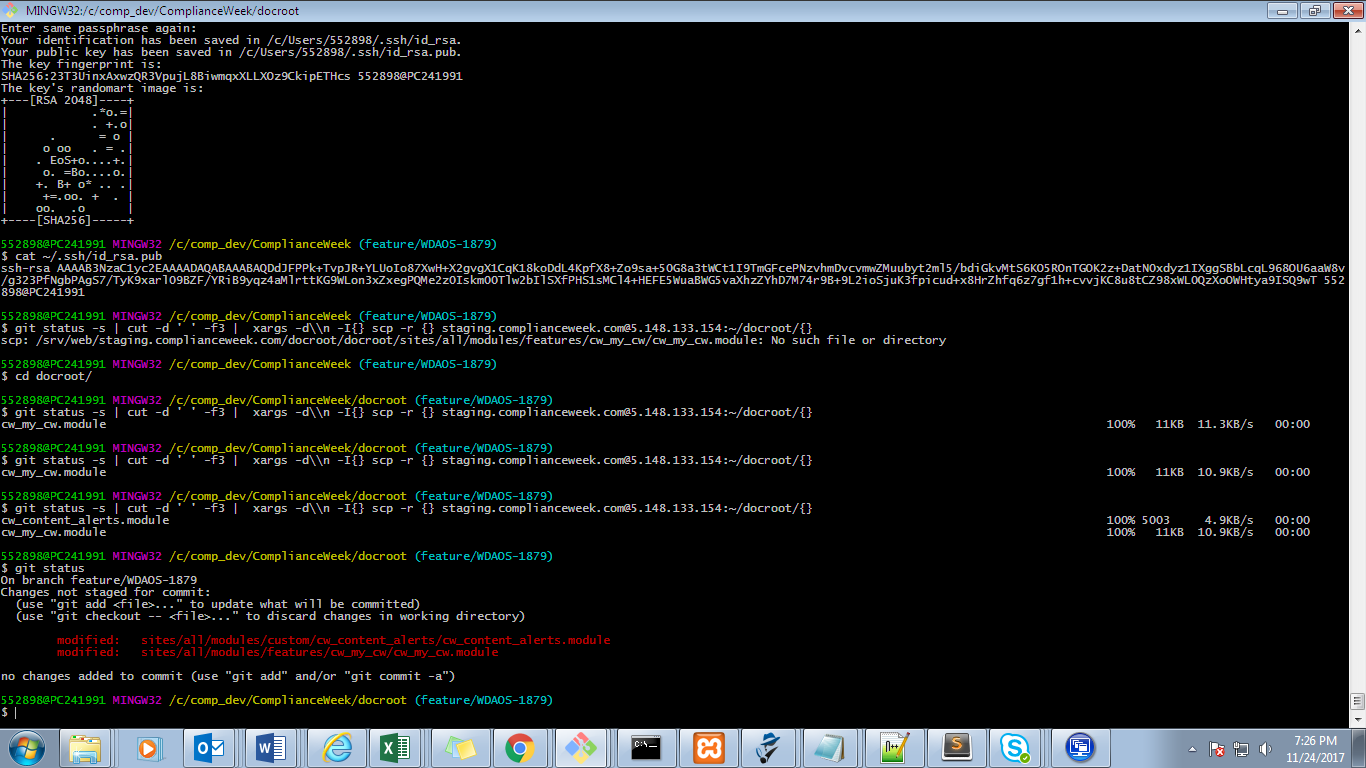
**Viewing the generated file after ssh-keygen**

cat ~/.ssh/id\_rsa.pub

* **Display the generated key**
* **Copy this key .ssh in the server**



**After adding the ssh keys in the server**



**To check whether uploaded in the server**

**Use git diff <filename>**

git diff sites/all/modules/features/cw\_my\_cw/cw\_my\_cw.module

git config --global core.autocrlf false

Applying patch:

1. Add the patch file in contrib module folder:

Eg: globalredirect

1. Navigate to that folder

Note: If the branch gets re-routed to master branch, make sure to delete .git folder from that branch

Rm –rf .git

1. Apply the patch using the command:

patch -p1 < 1438584-30\_globalredirect\_active\_path.patch

**GIT REBASE:**

GIT LOCAL:

- git checkout master

- git pull

- git checkout feature/haplo-amends-v3

- git pull origin feature/haplo-amends-v3

- git rebase master

- git push --force origin feature/haplo-amends-v3

GIT SERVER:

- git fetch && git checkout feature/haplo-amends-v3

- git pull

- merge message(ctrl x n no)

- git checkout master

- git pull

- git branch -D feature/haplo-amends-v3

- git checkout feature/haplo-amends-v3

- git pull

Compare master & other branches

<https://github.com/WilmingtonGroup/ComplianceWeek/compare/master...feature/WDAOS-2082-db-logs-v1>

**Copying one folder to another**

cp -rf staging-value.hsj.co.uk dev-value.hsj.co.uk

**Editing the file using nano**

1. nano settings.php
2. Searching for the string

Ctrl w

Search : staging

1. Save

Ctrl o

Enter

1. Exit

Ctrl x

**Setting the virtual host in server**

ssh cognizant@5.148.133.153

sudo su - go to root

cd /etc/apache2/sites-available

**Git config**

git config credential.helper store

git push <https://github.com/WilmingtonGroup/evo.git>

Username: <type your username>

Password: <type your password>

Masquerade module

* Adding primary key

function masquerade\_schema() {

return array(

'masquerade' => array(

'description' => 'Each masquerading user has their session recorded into the masquerade table. Each record represents a masquerading user.',

'fields' => array(

**// This modification is made to the contrib code. It ensures a primary key for the table**

**// It doesn't matter if the code is later overwritten by a contrib update so we won't make a patch for it**

**'masquerade\_id' => array(**

**'description' => 'The current session for this masquerading user corresponding to their {sessions}.sid.',**

**'type' => 'serial',**

**'unsigned' => TRUE,**

**'not null' => TRUE,**

**),**

'sid' => array(

'description' => 'The current session for this masquerading user corresponding to their {sessions}.sid.',

'type' => 'varchar',

'length' => '64',

'not null' => TRUE,

'default' => ''),

'uid\_from' => array(

'description' => 'The {users}.uid corresponding to a session.',

'type' => 'int',

'not null' => TRUE,

'default' => 0,

'disp-width' => '10'),

'uid\_as' => array(

'description' => 'The {users}.uid this session is masquerading as.',

'type' => 'int',

'not null' => TRUE,

'default' => 0,

'disp-width' => '10'),

),

**'primary key' => array('masquerade\_id'),**

'indexes' => array(

'sid' => array('sid', 'uid\_from'),

'sid\_2' => array('sid', 'uid\_as'),

),

),

'masquerade\_users' => array(

'description' => 'Per-user permission table granting permissions to switch as a specific user.',

'fields' => array(

'uid\_from' => array(

'description' => 'The {users}.uid that can masquerade as {masquerade\_users}.uid\_to.',

'type' => 'int',

'not null' => TRUE,

'default' => 0,

'disp-width' => 10,

),

'uid\_to' => array(

'description' => 'The {users}.uid that {masquerade\_users}.uid\_from can masquerade as.',

'type' => 'int',

'not null' => TRUE,

'default' => 0,

'disp-width' => 10,

),

),

'primary key' => array('uid\_from', 'uid\_to'),

),

);

}

Git Delete branch

- Delete locally

git branch -D backtop\_patch

- Delete remotely

git push origin --delete backtop\_patch