***FireFighter work***

The first thing you will need to do is follow the documentation: -

1. Go to Jenkins dashboard and follow up the already scheduled content pull project if it is already started, like the automatic schedule one, if not you start it manually.
2. Once it is finished its task check the status.
3. Once it is failed you can follow up accordingly
4. Once it is succeeded or you can see it in the console log, you follow what it said at the end of the console with good messages.
5. Then go to the page what changes have been done in the link:- <https://github.com/pearca/qiactive/pull/5138/files>
6. Checkout every files added or modified during the build process
7. There are files that cannot be shown in the GitHub and you can check it out in the Terminal
   1. Got the repo directory first cd $PEARSONPATH
   2. You can checkout to the latest branch created by Jenkins and make sure you can access it. You can make sure using git fetch origin content-pull-QA-XX
   3. Switch the branch to the new branch created during the job’s build by like git checkout content-pull-QA-XX you can find this name in:-<https://github.com/pearca/qiactive/pulls>
   4. Then see the difference made as compare to the master using the command git difftool -x json-diff master:filename filename
   5. After you check the confusions you can comment what you observed like share that you are satisfied or happy with the changes and submit it.
   6. After that someone from the review team will approve the result.
8. Once it is approved we do the merge and push to the final branch.

**Merging the approved branch to the Master or any other branch**

**Example to branch follows:**

1. Block first anybody from going to do the merge as you are doing merge by going to the link <https://qa.qiactive.com/get-banana.html>
2. Make sure you don’t have other application running in your grail local server and make sure you close any previous local running specs or tests.
3. Go back to terminal and go to the virtual-env and activate it
   1. virtualenv qiactive-venv/
   2. source qiactive-venv/bin/activate
4. Go back to the repo link cd $PEARSONPATH
5. make sure you are on the master branch using git checkout master
6. make sure you have all the latest source from the master by using git pull
7. start merging it now safely using tools/git-safemerge origin/content-pull-xx
8. you will be asked to make sure say yes
9. you will be asked to commit with a message you can leave it without message using :wq
10. you push it then to the remote master using ./pre-push --content-pull
    1. there are times we use other methods instead of the content-pull, like untestable or assess-js
11. at the end it will ask you if you are sure to do the push to the master, you type master and hit enter
12. make sure to release your banana at the end

**make sure you running the virtualBox server**

**in the terminal make sure the grails and py is in the right direction and running correctly.**

1. Next comes is content-push-QA by going to the Jenkins using the link <http://10.25.97.40:8080/job/Content-Push-Master-to-QA/>
2. There are times that we can push data to the QA in step 9 as there could be a problem
   1. Error code 302 which is a problem the Jenkin jobs from logging in to the QA using the giving credentials. To avoid this, you login and make some changes like renew the terms policy or changes of password, which is very rare, but still can happen
   2. Error code 403 is the password or usename is not correct
   3. Error code 404 is the server is down or the site is not working
   4. Error code 502 is a sign of bad gateway
   5. Error code 500, server problem from pushing datas to the QA because of some criteria been violated. For example, subtest GUID conflict can happen, in which to same subtest having different subtest GUID. This is not allowed and the Jenkins can not override the file and does fail automatically. To avoid this,
      1. You should make sure you have same GUID for same subtest and make sure they are under different test group. You can also set the ID manually.
      2. You can upload it manually by login to the QA. First download the metadata from CIT by going to the specific subtest. Then you login 🡪 admin🡪test🡪go down and choose the upload file.
3. Then you do the deployment process <http://10.25.97.40:4000/fires/deploy-central-qa/>
4. The deployment to the CIT is like optional and you can do it more upon demand as it runs every day automatically <http://10.25.97.40:4000/fires/deploy-cit-qa/>
5. Then Check for all the items ready to deploys are on QA and move them to “ready to test”
6. When you are deploying, in step 10, you need to make group aware of your action in case someone is doing or working in it same repo and you can email them saying like: with subject :- QA Central Recycling in 20 Minutes and with a body :- Please let me know if you want me to wait.

Thanks .

1. Then follow what is next as per the info in <http://10.25.97.40:4000/guides/firefighter-guide/>
2. During the time of sending and receiving i18n we follow the instruction given in <https://github.com/pearca/qiactive/wiki/Translation-Management-System-(TMS)-Automatic-Tool>
3. There are times deploying war files to QA does not work. Is that case we can do it manually. That is we go to the folder location where the war file exist , workspace/web/ChooseShare/target That way you will find the war file and you download it. Then you open cyberduck and copy it manually by dragging the downloaded war file to webapp folder of the server side opened in the cyberduck. In the cyberduck app you may have many servers listed, you need to open the exact one and go to qiactive/tomcat/webapp location and there where you drop your war file. After that you go back to terminal and go to the server and restart it then the war file deployed manually will be able to extract successfully.

1. When you see all the building and testing are gone well and successfully start sending emails to team with all the process completed. This is to be done most of the time twice. So you can update them one in mid-day and second at the end of the day. The subject can be like :- FF MD status and The body can look like :-

If I am asked to build from someone after they made a change

1. If they made change in Central I build the ChooShare <http://10.25.97.40:8080/job/ChooseShare-QA/>
2. If they made a change in Assess I build the push-master-QA <http://10.25.97.40:8080/job/Content-Push-Master-to-QA/>
3. If they made with objective C related staffs – run the ios projects the names with App----

Under the content pull all the sub-content can be found like in <https://github.com/pearca/qiactive/pull/5138> this will be repeated at the end of the day with current or latest updates.

While content pull is running you can follow in the build box server to follow what the Jasmine is doing during the test. Using the link 10.25.97.40 . Refer to the spread sheet link to get the full link and credential info. You can follow the steps in <http://10.25.97.40:4000/fires/mac-build-box/>

Once it is done as well, you can go to the <https://cit.qa.qiactive.com/choose/test/edit/151> to check what is made or if everything is going well.

Errors could happen: -

1. There are errors that can happen because of some scripts we write in the command shell and not being able to use it or access it. Or if the request reply with a null…in this case we can disable the export, like wms5 symbol span
2. In content-push-release there could be user credentials conflicts, which will send an error with 403 conflicts.
3. A project can be failed also if another job is going through.
4. There are times the grails do not start easily; you can try to start it locally by going to the choose-share directory like
   1. cd $PEARSONPATH/web/choose-share
   2. then grails run-app
   3. or you can check separately for the grails by grails or grails –version
   4. if this does not work, you will need to go to the .bash\_profile if it is added to the PATH root.

During the production we will need go through first the approval to do that

1. I need a link like <https://pearson.service-now.com/support/mystoredetail.do?sysparm_document_key=sc_cat_item,a80b77af5de06100839b759a791d7c7d&locationId>=
2. After I get approved I will be able to sign in to the ls link <http://ls.ic.ncs.com/>
3. and then go to the Conan job scheduler using the link <http://releng.ic.ncs.com/conan/requests/job?rt=209151>
   1. when you made a the ticket for the productions the page, scheduled page should look like 

whenever I made a change to my local codes and I try to pull content from the master, I may get code diff message and if did not make any major change and do not want to stop me from pulling my content, I can first go to the terminal and do thses commandes, git reset --hard origin/master.

Running a test locally without the server or running the whole app

1. If I want to run single test, I just have to go to the specific html runnerSpec and run it. Example cd $PEARSONPATH/jasmine-tests/SpecRunner-td-1.html
2. If I want to run the whole test in the master repo I go to the location cd $PEARSONPATH/tools/allJasmine

**How to raise ticket to deployment of Q-i 2.9  to**[**beta1.qiactive.com**](http://beta1.qiactive.com/)

1. You should first prepare a pull request for the database for all the springs not deployed since the last deployment. Prepare it like <https://github.com/pearca/qiactive/pull/5259/>
2. Place a folders full of these database data to the SFTP server so that other teams can access it. <https://q-interactive-sftp.pearson.com/>
3. Then you raise a ticket for that to run the scripts. In the giving link <https://pearsonstsprod.service-now.com/main/newchange.do>

**About release-2.9**

Today we have to create the new release branch called release 2.9 and we will ingest the i18n files to that release as we want to deploy the release into BETA1. Breaking up the central files.

So we will need to do today :-

1. Create a new release called release-2.9
2. We will need to ingest the internalization staff called i18n
3. We will then merge it with the master which contains all the releases including previous releases.

Ingestion process we need to first to validate our input json scripts using text editor.

**To ingest the i18 files**

1. Download the files and validate their formats and
2. Make sure you are getting .strings, .properties, and .json and sometimes .gsp
3. Arrange the .strings files you downloaded and delete the .txt extentions
4. You need validate their format using different editors,
   1. For .strings use plutil -lint filename.strings in the terminal,
   2. For .json do it in the editors like sublime or other editors
      1. Using sublime
         1. To validate hit command + shift +p
         2. To get the invalide number command + p
            1. Then :row:culumn
5. Create two branch for the tms from the release-2.9 branch
   1. Tms-asses-xx (which contains the json and .strings files)
   2. Tms-central-xx (which contains the .properties files and .gsp if exist)
6. Make sure your tms/cli is compiled using the scripts <https://github.com/pearca/tms/tree/master/cli>
7. Then follow the documentations written in the <https://github.com/pearca/qiactive/wiki/Translation-Management-System-(TMS)-Automatic-Tool>
8. Make sure you are in the root of tms/cli of the respective branch
9. In one terminal tab you need to be in either branches you created
10. Do the ./importAssessData.shand you will be asked to give the file to be transformed like and you can type like messages\_da.json **for each languages you want to make changes** andunder the **Tms-asses-xx** created (xx is mostly dates ddmmyyyy)
11. Then you will be asked to give the file and Enter the tms json file for that specific language and you drag and drop the file with specific language
12. The do again step 10 and 11 with region in it like example for **messages\_da-DK.json** the da represent the language and the DK represent the region.
13. And again you drag and drop the same file as in 11.
14. Then you copy the strings files to the assess by going to the folder manually /**Volumes/workspace/qiactive/ipad/Give** and find the respective languages and regions and find if there is any other folder with a region in it and change the string file extension as well. And delete the language name added.
15. Create a PR for this branch and
16. Change your branch to other branch of central in the second Terminal **Tms-central-xx**
17. The same thing for the properties files also, we run the script .**/importCentralData**.shandtype **messages\_xx.properties** and drag and drop for the respective languageunderthe **tms/cli**
18. Then you delete the old ones and replace them with the new once by modifying their name, deleting the language or region letters.
19. Lastly you PR your branches to be merged in to the main branch – release-2.9

**Content-pull-2.9**

We need to create a new content-pull called content-pull-2.9 in our Jenkins to extract new content been made in CIT for the tests. But we need to be specific what kind of tests are we planning to include in the new release. So we will get all the tests that are going to be included in this new release. The other test will still be managed by the previous content-pull-QA of master. In general

1. we are having two content-pull both for release-2.9 and master.
2. We also will need to merge the release-2.9 to master sometimes in a day.
3. To that we need to first make sure our local release-2.9 and master is fully updated with their remote branch
4. Then I do the safemerge from release-2.9 to master

**How to push contents to Beta1 and Other env other than QA**

1. **ssh** [**qiactive@10.25.97.40**](mailto:qiactive@10.25.97.40) then you will be asked for the password, use find it in this link <https://docs.google.com/spreadsheets/d/1EkyWvzz6I2oEXn3WBBTWUP7a4OnFFjTVs2I-nbO6XIg>
2. then type **sudo su hosting** you will be asked same password as before and use it
3. then fid the location of the data using like **cd /Library/WebServer/Documents/build-product/Content-Push-Release2.5-releaseQA-to-QA/** fix the last job-name with the job you want to transfer
4. cd to the job number like **cd 34**
5. find what is inside it using the ls and then compress it into tar file using the command **tar -zxvf “BuildProduct-15-10-26-10-52.tbz”** make sure the tbz file is the exact and current file
6. then push it into the exact location like Beta1 or TEST or other using a command like **./deployTarsToExtEnv Beta1**
7. you will be asked username and password for that and use jwilsdon and password is with Jon for the password
8. After that it will do the rest of the job itself
9. Make sure you have the exact name for the desired environment and you double check it by looking at the **includedTests.manifest** file

**While deploying to PROD**

1. The name of the environment is also Production. Make sure you have same contents by going to the directory where this war is stored as was deployed to the BETA1 and go to the config.chosenConfiguration and change the content name of releaseQA to release..
2. Edit config.chosenConfiguration from releaseQA to release. To that:
   1. Vi config.chosenConfiguration
   2. To go to the edit mode press **i** and then edit it.
   3. After finishing press **esc** and then, save and quit using **ZZ**
   4. More follow <https://www.washington.edu/computing/unix/vi.html>
3. Then repeat step 6

**Resolving a conflicts**

1. Go to sourceTree and find out where are the conflicts and unstaged files
2. Do make an adjustment in there (sourceTree) either by the discard hunk or the right click and resolve conflict options

OR

1. Open the file in any other editor and make the changes and save it
2. Now you go back to terminal and commit the changes using the commands **git commit –av** or **git commit –m “commentMessage”**
3. Then go back to normal push form like **./pre-push –content-pull**

**Adding new alias in your bash**

1. Add any alias you want example alias gp=’git pull’
2. After you add, go back to the terminal and make sure you put it correctly using the command **alias | grep git**
3. You will see how you add your alias
4. You can make change in the terminal as well using the command **vi .bash\_profile** and adjust it in the terminal
5. Then do **source .bash\_profile**
6. You all set then

**How to make your script or file executable**

* 1. **Chmod +x ‘filename’**

**Creating PR for your branch**

1. Make a branch out of the base branch you want make **git checkout –b branch\_name**
2. Make any changes to the files inside that created branch
3. Save and add and commit like normaly you would do
   1. **Git add that\_file\_name**
   2. **Git commit –m “commentMessage”**
4. Push it to the remote using **git push –set-upstream origin That\_branch\_name** this is for the first time but later if you want just push, **git push origin That\_branch\_name**
5. You do not need to update that branch whenever you want to merge to the main branch like master or release-2.9.

**Reading server problem in their logs**

1. Get the splunk log information in <https://github.com/pearca/qiactive/wiki/Choose-Share#splunk-logs>
2. And follow the information and login to the next site and paste the code for that specific server command
3. <http://icdupaiammon03.ic.ncs.com:8000/en-US/app/search/search?earliest=1481140818.249&latest=1481142028.25&q=search%20sourcetype%3D%22eclick_cat_tomcat%22&display.page.search.mode=smart&display.prefs.events.offset=20&sid=1481143767.4016972>

**Deploying to environments through Conan**

when we plan to update out website servers of the deferent servers unlike the common QA, we do it through the Conan jobs website. But we need to first build our jobs in the Jenkins dashboard. This will make to create the war files and transport them to the nexus website. That way Conan can find them and update the respective website server. The common jobs we need to run for that website updating are:

1. **ChooseShare-deploy**
2. **SupportTabContent-deploy**
3. **ChooseShare-Static-Asset-Deploy**
4. **ChooseShare-DD**
5. We also need to make sure we run ChooseShare-DD after all the previous steps
6. We need to make sure we are deploying the right branch; we can do this by going to configuration of that job
7. We also need to make sure the version is the same in all the jobs. In our case we make the version to be 2.9-rc3,
8. we usually increment that version whenever we plan to deploy. succeed
9. make sure all the jobs are deployed to the nexus by going to the link <http://repo1.ic.ncs.com/nexus/#nexus-search;quick~qiactive>
10. Make sure you send the RTT number you will get from the Conan job in slack and
11. Send an email to the qi-interactive team to tell that server is down for the time.

For more refer the following link <https://github.com/pearca/qiactive/wiki/Deploy-a-release-with-Nexus-and-CONAN>

**Good practices in Jenkins**

1. Our main job is to
   1. Deploy War files to website servers
   2. Push contents (Tar files) to website servers, this might need some manual work in addition of the Jenkins job
   3. SupportTabContent-local and deploy
   4. Deploy new App version
2. Run the dbmaintain-DEV every Monday morning
3. Run the dbmaintain-INT every Monday morning
4. Run the suportContentTab every time you see new content is being added

**when we schedule Conan**

1. the application name is QIACTIVE\_TOMCATAPP
2. version name is whatever version we gave to out Jenkins jobs. In our case now 2.9.0-rc3
3. the environment will be the name of website server environment you plan to update

**how to build your App locally for a specific branch or Server**

1. You go to the branch in your terminal ( make sure you have the latest contents)
2. And run the python command ./**buildContent.py dev**
3. Make sure you know the right configuration for the right branch like **dev** or **relleaseQA** or **release**
4. Try to run the server by going to the chooseShare directory and run the app using the command **grails run-app**. You can do this before step 2 as well.
5. And then go to your project rex link <http://localhost/give/homeUI_en.html>
6. And open the **AdminAssess** tab and you will see a new page which represents the app
7. You do your test then. In the project rex page
8. You can also go to the local page **localhost:8080/chooseShare**

**When merging branches, we need to run different command to run the specific files like**

1. if any file in give-www js files are changed

Then **--assess-js**

1. if any change in web/choose-share

Then **--chair**

1. If any change in web/choose

Then **--cit**

1. 4. And if combination

Then **--assess-js --chair --cit**

**How to revert commit from a branch**

1. search to the commit number that is the SHA on the far right of that commit list
2. then copy it and go to the terminal and go to the specific branch you want to revert
3. then you type **git revert SHA**
4. then you will be asked to make a commit message, but you can leave it blank by saying **:q**
5. then you need to push it using **git push origin yourbranchname**

**how to clean all the cache and dependencies?**

1. Go to the project directory
   1. **We/choose**
   2. **Web/chooseShare**
2. Run the command **grails clean-all && grails refresh-dependencies && grails compile**

**Deleting files under a given directory**

* 1. **find generated -mindepth 1 –delete**
  2. **find . –name “dirname” –d** for folder
  3. **find . –name “filename” –f** for files

**cloning commit done to another branch**

1. find the commit number from the site under the closed pull request button and find the exact commit and go inside by clicking it and go to the commit tab
2. you will see the number to the right and copy it
3. got back to your terminal and go to the right branch you wanted to clone to.
4. Right the command **git cherry-pick “commit\_number\_you\_got”**
5. Then do **git push –set-upstream origin branchname** ( as you are doing this in a new branch created branch for pull request)

**When deploying products to the final server products**

You check the sql

You check the all the tests one by one

Add liscese

**Configuring the Build Box**

If the build box starts to act weird, we need to check out external hard drive using the disc utility to check its health and go from it. If it looks mess we need to at most format it and restart our workspace from scratch. But we need

1. Clone the repo toolbox first using git clone <https://github.com/pearca/toolbox.git>
2. Run the documentation job <http://10.25.97.40:8080/job/_DOCUMENTATION_/>
3. MacPro-Cleanup <http://10.25.97.40:8080/job/MacPro-Cleanup/>

The in the external drive in the bbox as well, we need to

1. Go to the toolbox and run **npm install**
2. Go to the build dir and run **npm install**
3. Go to the util dir and run **npm install**

**How to update and run specs locally of a given branch**

1. Get the new and updated contents from the remote branch **git pull**
2. Run **grails clean**
3. Run **grails run-app**
4. Then in the root pearsonpath path run **./buildcontent dev noTar**
5. The you are ready to see all the updated current specs locally

**Some changes being made in the jobs build and configurations**

1. Content-Pull-QA is pointing to CIT-INT2 instead of CIT-QA
2. Content-Pull-QA is pulling contents to all tests
3. CIT is being deployed to CIT-INT2 instead if CIT-QA
4. Job Choose-QA is disabled until last release