Release process using GIT

In the past we’ve released straight from master but there’s been a need to stop this process as QA would like to have a their own environment to test a release candidate without having to worry about code check-ins or data changing in the middle of their test cases. The following describes doing a release from a QA release candidate instead of releasing from master.

Get the code and create local repositories:

* git remote add origin <https://github.com/klasresearch/KLAS.git> (This is stored on the Internet)
* git fetch origin
* git remote add live-research [http://kudu-live-research/live-research.git](http://www.google.com/url?q=http%3A%2F%2Fkudu-live-research%2Flive-research.git&sa=D&sntz=1&usg=AFQjCNHL3dpduy-HzHz6LPkKN4LFmFz7VA) (This is stored on a KLAS box called Ash – trying bring up “http://kudu-live-research/” in a browser)
* git fetch live-research
* git remote add live-toolbox [http://kudu-live-toolbox/live-toolbox.git](http://www.google.com/url?q=http%3A%2F%2Fkudu-live-toolbox%2Flive-toolbox.git&sa=D&sntz=1&usg=AFQjCNEIkx65Ac84o0SKGyfFL5CoJ0RQvw) (This is stored on a KLAS box called Ash – trying bring up “http://kudu-live-toolbox/” in a browser)
* git fetch live-toolbox

Next, create your local branches that map to the remote branches:

* git checkout -b master origin/master
* git checkout -b qa origin/qa
* git checkout -b staging origin/staging
* git checkout -b live-toolbox live-toolbox/live
* git checkout -b live-research live-research/live

Next, make code changes to MASTER and push:

* git pull
* git status
* make code changes
* git push

Once the code is checked in (pushed) you’ll want to release it to the QA environment for testing. To do this simply merge master into QA and push the code to the remote repository.

* git checkout qa
* git pull
* git merge master
* git push

If QA finds an issues that needs to be address before the release the developer should checkout QA make the fix and then merge that fix into master. At this point MASTER SHOULD NOT BE MERGED INTO QA.

* developer makes fix on qa branch and pushes it to qa

Merge qa into master

* git checkout master
* git pull
* git merge qa
* git push

If the code looks good go ahead and push it to Staging. The following commands will push both research and toolbox to staging. Kudu is not set up to do them individually.

* git checkout staging
* git pull
* git merge qa
* git push

If the code looks good on staging go ahead and push live. The following commands will push toolbox to live:

* git checkout live-toolbox
* git pull
* git merge qa ( merges in the tested release candidate)
* git push

How to do a hotfix.

The following commands will checkout the code on toolbox live so you can make your hotfix change.

* git checkout live-toolbox
* git pull ( Be sure to pull so you’re sure you have the latest committed code on live-toolbox )

Make you changes and push your code to do the release:

* git commit –m “my hotfix code for ticket xxxxx”
* git push

Merge your hotfix changes into master *(You might just want to cherry pick the changes you made in release into master to avoid merge conflicts. Bear in mind if you cherry pick you might overwrite recent changes in the master for the same files that were changed in release.)*

* git checkout master
* git pull
* git merge live-toolbox
* git push