Tech Team Process Documentation

12/4/15

Web Developer Process:

* View tickets in sprint board on Jira
* Move ticket to “In process” column
* Create branch of ticket you’re going to work on from master
* If there’s a need for a new proc, proc change, or any change to edmx;
  + Create a subtask for a Database Developer to make such changes/creations etc
  + When ready, make edmx changes on master and push to master
  + Merge master back into branch of ticket you’re working on
  + Add proc name into original ticket as a comment
* Always add the ticket number to every commit for that branch. Example: [KP-XXXX]
* Checkout the master branch and do any pulls as necessary
* Checkout your branch and merge master into your local branch
* Build the solution and run all Unit/Integration Tests
* Test the branch on your local box and make any bug fixes necessary
* Checkout master and merge your ticketed branch into master
* Build the solution
* Do a pull to make sure you have latest and push master to master
* Check Kudu to make sure the solution builds
* Test the ticket on the dev environment
* Move the ticket to the “To Test” column in the sprint board (Resolve the ticket) with instructions on how to test the ticket
* (Possibly)-Don’t delete the branch until the sprint has been pushed.

This process allows the developer to work on any ticket with clean code and have the ability to push or not push any ticket’s code with ease.

Additional Items:

* Sprint items need to be pushed to QA to be fully tested during the sprint
* Once Code Stop has been reached, no more new items will be pushed to QA
  + If any bug fixes are required, branches will be made from QA and not master
  + Merge fixed bugs into master and into QA
  + Never push master into QA, this allows other developers to continue to work and push edmx changes to master without ruining the release branch (QA)
* QA Database needs to be refreshed often
* Any Hotfix needs to be reviewed by Rodney before pushing
* Never push any new code on Staging or Test on Staging until the day of the release
* Push the QA branch to staging, and then to live. Don’t push master to Live.
* (Possibly)-Add proc name to specific ticket anytime there’s a change to that proc or added new proc.

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.

Create remotes:

* git remote add origin <https://github.com/klasresearch/KLAS.git>
* 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)
* 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)
* git fetch live-toolbox

First, create your local branches that map to the remote branches if you haven’t done that already:

* 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

To add the live-toolbox and live-research remotes if they haven’t already been added locally use:

* 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)
* git remote add live-research <http://kudu-live-research/live-research.git>

Once the code is checked in and it’s ready to be tested you’ll want to release it to the QA environment for testing. To do this simply merge master into QA and push to code to 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. MASTER SHOULD NOT BE MERGED INTO QA.

* developer makes fix on qa 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

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

Congrats you just did your hotfix.