ICD/TID are in the process of migrating to using GIT for SCM. This document outlines a few guidelines for the GIT master repos for projects.

1. Decide on a name for your project.
   - The name has to be unique to your project; that is, we should not have two projects with the same name.
   - For example, `autosave` is a good name for the project containing the `autosave` module from the collaboration.
   - Avoid spaces and other special characters in your name.
2. Use this name for GIT, `cram`, `eco`, JIRA and all the other package management tools in use.
3. The master git repos themselves are located in this subtree - `/afs/slac/g/cd/swe/git/repos`.
   1. There are subfolders for package types; for example, all master repos for IOC modules are located here - `/afs/slac/g/cd/swe/git/repos/epics/modules`.
   2. For example, the `autosave` module's master repo will be here - `/afs/slac/g/cd/swe/git/repos/epics/modules/autosave.git`
   3. If unsure of the location or the name, please ask one of the functional leads.
4. Create a bare repo called *`projectname`*`.git` in the appropriate location.
   1. For example, to create the master repo for the `autosave` module, you could

      ```
      cd /afs/slac/g/cd/swe/git/repos/epics/modules/
      git init --bare autosave.git
      ```

   2. For example, to create the master repo for the `MatlabSupport` IOC, you could

      ```
      cd /afs/slac/g/cd/swe/git/repos/epics/iocTop
      git init --bare MatlabSupport.git
      ```

5. You need to tell `eco` that this is a GIT module. Edit `${TOOLS}/eco_modulelist/modulelist.txt` and add an entry for your module.

   ```
   [mshankar@lcls-dev2 ~]$ cd ${TOOLS}/eco_modulelist/
   [mshankar@lcls-dev2 eco_modulelist]$ vim modulelist.txt
   [mshankar@lcls-dev2 eco_modulelist]$ git add modulelist.txt
   [mshankar@lcls-dev2 eco_modulelist]$ git commit -m "Added location for IOC MatlabSupport"
   [master 9957771] Added location for IOC MatlabSupport
    1 file changed, 2 insertions(+), 2 deletions(-)
   [mshankar@lcls-dev2 eco_modulelist]$ git push
   Counting objects: 5, done.
   ...
   To /afs/slac.stanford.edu/g/cd/swe/git/repos/eco_modulelist.git
      5872c50..9957771  master -> master
   [mshankar@lcls-dev2 eco_modulelist]$
   ```

6. That's it! You can now use `eco` to clone this repo into your workspace and work with it. Here are some useful commands to get you started.
   1. For example, to checkout MatlabSupport into your workspace, you can

      ```
      [mshankar@lcls-dev2 workspace]$ cd workspace
      [mshankar@lcls-dev2 workspace]$ eco
      Enter name of module/package to checkout: MatlabSupport
      Enter name of tag or [RETURN] to use HEAD>
      Using MAIN_TRUNK. The name of the directory will be MAIN_TRUNK.
      MatlabSupport is a git package. Cloning the repository at /afs/slac/g/cd/swe/git/repos/epics/iocTop/MatlabSupport.git
      Cloning into 'MAIN_TRUNK'...
      done.
      Enter full path for EPICS_SITE_TOP or [RETURN] to use "/afs/slac/g/lcls/epics">
      ...
      [mshankar@lcls-dev2 workspace]$
      [mshankar@lcls-dev2 workspace]$ cd MatlabSupport/MAIN_TRUNK/
      [mshankar@lcls-dev2 MAIN_TRUNK]$
      ```

   2. To get the current status of the repo, use `git status`. GIT's status messages are very helpful.

      ```
      [mshankar@lcls-dev2 MAIN_TRUNK]$ git status
      # On branch master
      # Changes not staged for commit:
      #   (use "git add ..." to update what will be committed)
      #   (use "git checkout -- ..." to discard changes in working directory)
      #
      #       modified:   Makefile
      #       modified:   configure/RELEASE
      #
      no changes added to commit (use "git add" and/or "git commit -a")
      [mshankar@lcls-dev2 MAIN_TRUNK]$
      ```

   3. GIT has a staging area, so to add a change or a new file to a repo, you should first use `git add`. You can use `git add --update` to add all untracked files.

      ```
      [mshankar@lcls-dev2 MAIN_TRUNK]$ git add Makefile configure/RELEASE
      [mshankar@lcls-dev2 MAIN_TRUNK]$
      ```

   4. The staging area in GIT is one of the big changes from CVS that stumps many people. It is also incredibly useful. For example, the staging area let's you
      - Use `git add -p` to sift through all your changes and *stage* individual pieces for the next commit.
      - Then, use `git commit` to commit this set of changes.
      - Then, start over again with another `git add -p` to pick and choose the next set of changes.
      - Lather, rinse, repeat till all the changes have been commited.

5. To commit a set of changes from the staging area to your local cloned repo, use `git commit`

```
[mshankar@lcls-dev2 MAIN_TRUNK]$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD ..." to unstage)
#
#       modified:   Makefile
#       modified:   configure/RELEASE
#
[mshankar@lcls-dev2 MAIN_TRUNK]$ git commit -m "Added support for LinuxRT"
[master 507c271] Added support for LinuxRT
 2 files changed, 4 insertions(+)
[mshankar@lcls-dev2 MAIN_TRUNK]$ 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 directory clean
[mshankar@lcls-dev2 MAIN_TRUNK]$
```

6. So far, your changes have only been committed to the local repo. To push your changes to the master repo, use `git push`.

```
[mshankar@lcls-dev2 MAIN_TRUNK]$ git push
Counting objects: 9, done.
Delta compression using up to 24 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 481 bytes, done.
Total 5 (delta 4), reused 0 (delta 0)
To /afs/slac/g/cd/swe/git/repos/epics/iocTop/MatlabSupport.git
   a0e4130..507c271  master -> master
[mshankar@lcls-dev2 MAIN_TRUNK]$
```

7. When you are pushing for the very first time, you may need to do a `git push origin master`.
8. When you are ready to release, create a tag using `git tag`. You will have to push tags separately using `git push --tags`.

```
[mshankar@lcls-dev2 MAIN_TRUNK]$ git tag V_3_2_1
[mshankar@lcls-dev2 MAIN_TRUNK]$ git push --tags
Total 0 (delta 0), reused 0 (delta 0)
To /afs/slac/g/cd/swe/git/repos/epics/iocTop/MatlabSupport.git
 * [new tag]         V_3_2_1 -> V_3_2_1
[mshankar@lcls-dev2 MAIN_TRUNK]$
```

9. To checkout a tagged release, use `eco` again.

```
[mshankar@lcls-dev2 MAIN_TRUNK]$ cd ../..
[mshankar@lcls-dev2 workspace]$ eco
Enter name of module/package to checkout: MatlabSupport
Enter name of tag or [RETURN] to use HEAD>V_3_2_1
Using V_3_2_1. The name of the directory will be V_3_2_1.
MatlabSupport is a git package. Cloning the repository at /afs/slac/g/cd/swe/git/repos/epics/iocTop/MatlabSupport.git
['git', 'clone', '--recursive', '/afs/slac/g/cd/swe/git/repos/epics/iocTop/MatlabSupport.git', 'V_3_2_1']
Cloning into 'V_3_2_1'...
done.
['git', 'checkout', 'V_3_2_1']
Note: checking out 'V_3_2_1'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b new_branch_name

HEAD is now at 507c271... Added support for LinuxRT
Enter full path for EPICS_SITE_TOP or [RETURN] to use "/afs/slac/g/lcls/epics">
...
[mshankar@lcls-dev2 workspace]$ cd MatlabSupport/V_3_2_1/
[mshankar@lcls-dev2 V_3_2_1]$
```

10. Now run `make` and `cram push` and so on...