Set up Configuration Files using opsim3_config

Configuration Files

All of the benchmark configuration files are maintained in opsim3_config on github. The primary branches correspond to the set of configuration files for a particular benchmark run (denoted by its name).

You MUST have setup configuration files on the machine where you intend to run OpSim and specify that in --config=~/someplace to execute opsim.py. Instead of thinking of sets of configuration files as subdirectories, think of them as "git commits"

Recommended

1. A machine with ssh keys set up for github to checkout opsim3_config, make and commit changes, push them to github.
2. A machine for simulation production
    a. A directory for each benchmark run is created with and OpSim installation in /lsst/config (a separate git clone and git checkout is done to set up each)
    b. A directory for checking out a read-only copy of the opsim3_conf repository is needed - the SHA of the commit for a particular set of config settings is specified. (e.g. ~/github/opsim3_config)

Setup Method 1: Batch execute all benchmark runs.

As part of the OpSim code installation each production machine has a copy of all the current baseline/tier1/neo config branches (e.g. in ops2:/lsst/config/). Each of these directories is a separate git clone of opsim3_config and has the branch indicated by the name of the subdirectory already checked out (a checked out branch is unique to the each subdirectory). This setup and is suitable for using to batch run a set of benchmark runs all at once without having to check out each branch and start the run in sequence.

If you want to create your own set of benchmark directories:

The github repository "opsim_production_tools" contains a script which generates all of the benchmark ("base" config) directories

```
# clone the repository
cd ~/github
git clone git@github.com/lsst-sims/opsim_production_tools.git
# then create set of directories, each of which is a git clone of all the base branches using config_maint
~/github/lsst-sims/opsim_production_tools/config_maint
# valid switches are
-c, --create
-u, --update
--config-loc # wherever you want it if you don't want it in /lsst/config (default)
```

Setup Method 2: How to set up your own configuration commits to experiment with settings.

In this method, a single git clone of opsim3_config (e.g. located in your home directory) is set up to manage creation of sets of configuration files that will be used for running simulations. One commit can be arranged such that it is a complete set of settings with a message describing it so it may be differentiated from other commits.

The essential actions are: check out an opsim3_config branch (optionally creating a new branch based on a benchmark branch); editing the settings in the files; committing them; pushing them to github (so they can be checked out on any production machine).

Create a new branch in github (can also do this directly in git):

```
# Create a new branch based off a benchmark branch on github.com
# In github switch to the branch you want as your base
# Create a new branch by typing kcook/my_new_branch_name in the "Switch branches/tags" dropdown box
# Check out the new branch from github to your config machine

Clone the repository:

# create a writeable clone of opsim3_config.git
# choose an appropriate location to work on the config files and cd there, e.g.
cd ~/github
# clone the config repository
git clone git@github.com/lsst-sims/opsim3_config.git
```

Now the process is to make edits, stage them and commit them (and note SHA1). From a production machine you can then check out these commits and start a simulation.

Switch to the branch you created in github:

```
# From your config machine
git branch -a # check all branches on remote machine
git branch # shows names of branches already on local
# checkout a branch from remote
git checkout -t origin/xxxxx # -t means track the branch on your local machine (same branch name)
# or git checkout xxxxxx to get into branch already on the local machine
git branch # check you are on the branch you think you are
```

Or you can create a new branch from within git:

```
git checkout -b mynew_test # creates a new branch from this point
```

Displaying the history (or log) of commits:

```
git log # reverse chronological order of commits
# SHA is unique to this repository; but first 7 chars are enough (short version)
# short commit (~70 chars) should be concise and meaningful
git log --pretty=short # multiline SHA and short message
git log --pretty=oneline # SHA and short message
git log --format="%\n" # formatted
git log -n 1 # limits the number of commits - most recent first
git log -n 1 --pretty=oneline
# man pages needs the command
man git-log
man git-commit
```

Making changes to config files

```
vi survey/LSST.conf
git status # sheck that you edited; shows files that have changed
git diff survey/LSST.conf # checks between current state and last commit
git checkout survey/LSST.conf # resets the file to original from last commit
```

Stage changes

```
git add survey/LSST.conf # stages one file
git add . # will stages EVERYTHING so make sure this is what you want
git status # tells what you have done
git reset HEAD test.txt # can back out a file that was staged accidentally
git clean -f # will remove untracked files
git clean -d # will remove untracked directories
```

Commit changes

```
git commit -m "Making a one year run" #commits with a short message (no editor)
git log --pretty=oneline # displays a list of commits (represents a set of files)
```

Continue editing, staging, and committing.

Push all changes up to repository on github

```
git push origin tier1/run02 # for example
```

A Scenario and possible use cases

1) I need to adjust numbers in a previous commit (forgot to make a complete set of changes)

```
cd survey
vi Universal.conf
git add Universal.conf
git commit -n "Making requested visits 1 year WFD"
```

2) I meant to change LSST.conf to point to a different cloud file

```
cd survey

vi SiteCP.conf
```

edit cloud table name and seeing table names

```
git add SiteCP.conf
git commit -m "Changing Seeing and Clouds to 1 Year tables."
```

3) I want to adjust the minimum distance to the moon.

```
cd scheduler

vi Scheduler.conf
```

edit the parameter

```
git add Scheduler.conf
```

Now we have 3 different commits but only the last on has all the changes we want.

So we roll up all the commits using git rebase

Rebase destroys history - So this only on non-public branches

```
 git rebase -i # do interactively to do clever things
```

shows 3 commits

use interactive commands to manipulate

doing nothing - leaves things as is (three commits stays three commits)
doing :q leaves rebase without ANY changes made
edit "s" instead of "pick"
:wq - leaves rebase, gives log message editor, saves changes
top line (Line 1) is the short message - make a descriptive as possible as short as possible
can include detail in subsequent lines
:wq to save finally

```
 git log # shows that there is one commit in addition to the original checkout point
```

What if I want to get back to a SHA?

```
 git checkout 949b0ac
```

What if I want to create a new branch from this point in my commit history?

```
git checkout -b mynew_test
```

What if I want to include that last change after all?

```
git merge mareuter/test_conf
```

What if the changed I want to add back are in various commits?

```
git cherry-pick …. #will let you pick and choose what other changes you might want to add in by specifying a SHA
to reapply to this branch; sometime there are conflicts you may have to resolve
```

Setting Up A Basic Workflow to Run Simulations

This section describes a typical workflow of creating a place to change parameters and execute runs to explore a particular topic.

A suggested procedure is to use one machine (home) on which to clone opsim3_conf, make edits, commit them, and push them to github so they can be checked out on any other production machine. This way only one machine will have to be maintained with a read/write repository (so you can push commits to github) a set of ssh keys.

A full description of all configuration parameters is here: http://ops2.lsst.org/docs/current/configuration.html#configuration

```
 # Setup read/write repository with keys on one machine
 # check out all branches (or the ones you are interested in) and make new ones for new studies
```

On home laptop

```
 # chose an appropriate location to work on the config files and cd there, e.g.
 cd /Users/petry/github
 # clone the repository to this location
 git clone git@github.com:/lsst-sims/opsim3_config.git
 # make edits, stage them and commit them; and note SHA1
 <see above>
 # push commits to github
 git push origin tier1/run02 # (or whatever branch you are working on)
```

On production machine

```
# choose a location for the configuration files (could be your home dir or the production dir)
# clone the repository - this is a one-way version; you can't commit changes from here
git clone https://github.com/lsst-sims/opsim3_config.git
git checkout master
git checkout cc52e00 # checkout the commit (SHA) containing the configs that you want to run
git log -n 1 --pretty=oneline # make sure you have the commit you think you do by cross checking the message
```

Go to How to Run OpSim and MAF to start the simulation and run MAF.