Friday, December 3, 2021

Lab 4: Git Hooks for forcing local auto-testing

Configuration

By Default you can find git hooks in `.git/hooks`

```
git config --local core.hooksPath "$GIT_DIR/hooks"
```

Git Hooks

1. Start with the usual creation of a lab repo.

```
mkdir ./lab4
cd ./lab4
git init
```

2. Now create a `test.sh` file for the git commit

```
echo -e "#/bin/bash\necho \"test works\";" > ./test.sh
chmod 751 ./test.sh # u+rwx,g+rx,o+x
./test.sh || echo "debug test.sh"
git add ./test.sh
git commit -m "added a basic test script"
```

3. Find the sample template hooks and copy the `pre-commit` template located at...

   `.git/hooks/pre-commit.sample`

   ... and create a copy at `.git/hooks/pre-commit`:

```
ls -alp .git/hooks
cp -vf .git/hooks/pre-commit.sample .git/hooks/pre-commit
```

4. Edit the file `.git/hooks/pre-commit` and CAREFULLY add an if-then-else to run the `test.sh` script we already created.

4a. Find the 2 lines (should be around line 21-22 in the default sample):

```
# Redirect output to stderr.
exec 1>&2
```

4b. and replace those with the following (note the two lines are included in the replacement text to simplify changes)

```
forcecommittest=$(git config --bool hooks.forcecommittest)

# Redirect output to stderr.
exec 1>&2

# now run unit tests
if [ "$forcecommittest" != "true" ] ; then
   echo "skipping tests (hooks.forcecommittest disabled)" ;
else
   if [ -x ./tests.sh ] ; then
        ./test.sh || exit 1 ;
   else
        echo "skipping tests (put tests in ./test.sh)" ;
   fi
fi
```

5. Now enable tests with the new hook:

```
git config --local hooks.forcecommittest true
```

6. Now test that the new hook works on new commits:

```
echo "change to test." > ./testfile.txt
git add ./testfile.txt
git commit -m "test git hook"
```