eriq@swc-anderson-lt-2.local:/Users/eriq — ssh eriq@colostate.edu@login11.rc.colorado.edu — 229×57

```yaml
name: quick-test
channels:
  - bioconda
  - defaults
dependencies:
  - _libgcc_mutex=0.1=main
  - bcftools=1.9=ha228f0b_4
  - bzip2=1.0.8=h7b6447c_0
  - ca-certificates=2019.5.15=1
  - curl=7.65.3=hbc83047_0
  - htslib=1.9=ha228f0b_7
  - krb5=1.16.1=h173b8e3_7
  - libcurl=7.65.3=h20c2e04_0
  - libdeflate=1.0=h14c3975_1
  - libedit=3.1.20181209=hc058e9b_0
  - libgcc-ng=9.1.0=hdf63c60_0
  - libssh2=1.8.2=h1ba5d50_0
  - libstdcxx-ng=9.1.0=hdf63c60_0
  - ncurses=6.1=he6710b0_1
  - openssl=1.1.1c=h7b6447c_1
  - samtools=1.9=h10a08f8_12
  - tk=8.6.8=hbc83047_0
  - xz=5.2.4=h14c3975_4
  - zlib=1.2.11=h7b6447c_3
prefix: /home/eriq@colostate.edu/miniconda3/envs/quick-tes
t
```

1,2 All

```bash
# assign my GitHub username to the variable USER
USER=eriqande

# assign the current working directory to the variable RUNDIR
RUNDIR=$PWD

# make a new directory named whatever the user wanted for the output directory
mkdir -p $DD

# make variables to hold log and error file names
LOG=${PWD}/${RP}log
ERR=$LOG.stderr

# print the date/time when the process is starting
echo "STARTING at $(date)"

# make a clean slate. remove any files with the name
# of the error output file
rm -f $ERR

# cycle over the student GitHub names, and for each one *do*
# the commands that appear before the *done* keyword. Indenting
# is used to make it easier to read, but is not essential.
for L in $GHNAMES; do

    echo "Working on $L, starting at $(date)"  # print a progress line to stdout
    REPO=$GHP/${RP}$L     # combine variables into new variables that
    echo $REPO            # hold the URL for the repository to be
    DEST=$DD/$L           # cloned and the path where it should be cloned to

    # store the commands themselves into variables. Note the
    # use of double quotes.
    CLONE_IT="git clone ${REPO/github.com/$USER@github.com} $DEST"
    BRANCH_IT="git checkout -B $BRANCH"
    PUSH_IT="git push -u origin $BRANCH"


    # now, run those commands, chained together by exit-status-AND
    # operators (so it will stop if any one part fails), while
    # all the while appending error statements to the Error file. Run it
    # all within an "if" statement so you can deliver a report as to
    # whether the whole shebang succeeded or failed.
    if $CLONE_IT 2>> $ERR  && \
        cd $DEST && \
        $BRANCH_IT 2>> $ERR  && \
        $PUSH_IT 2>> $ERR  && \
        cd $RUNDIR   # at the very end make sure to return to the original working directory
    then
        echo "FULL SUCCESS $L"
    else
        echo "FAILURE SOMEWHERE WITHIN $L"
        cd $RUNDIR  # get back to the working directory from which the original command was run.
                    # so we are ready to handle the next student repo.
    fi
```

114,1 96%

```
(base) [~]--% ls
git-repos    quick-test-env.yml  test-envs-dir
miniconda3   README.mdwn         test-gh-classroom
(base) [~]--%
(base) [~]--%
```

```
(base) [bcftools]--% ls
ad-bias.so          fill-from-fasta.so  isecGT.so
af-dist.so          fill-tags.so        mendelian.so
check-ploidy.so     fixploidy.so        missing2ref.so
check-sparsity.so   fixref.so           prune.so
color-chrs.so       frameshifts.so      setGT.so
contrast.so         GTisec.so           split.so
counts.so           GTsubset.so         tag2tag.so
dosage.so           guess-ploidy.so     trio-switch-rate.so
fill-AN-AC.so       impute-info.so
(base) [bcftools]--%
```

[froggies] 0:data- 1:edit 2:big-long-window-name* "login11" 16:22 16-Feb-20