pssh HOWTO

Brent Chun

Table of Contents

1. Installation and Setup

To install the software, become `root` on your machine and do the following (on RedHat systems):

```
# rpm -ivh pssh-0.2.3-1.i386.rpm
Preparing...                ########################################### [100%]
   1:pssh                   ########################################### [100%]
```

By default, the software installs itself in `/usr/localbin` and `/usr/local/lib`. Thus, you'll next want to modify your `PATH` if needed:

```
# export PATH=$PATH:/usr/local/bin
```

2. Preliminaries

All four programs will print their usage and give an example if no arguments are given. For example, with `pssh`:

```
# pssh
Usage: pssh [OPTIONS] -h hosts.txt prog [arg0] ..

  -h --hosts   hosts file (each line "host[:port] [user]")
  -l --user    username (OPTIONAL)
  -p --par     max number of parallel threads (OPTIONAL)
  -o --outdir  output directory for stdout files (OPTIONAL)
  -t --timeout timeout in seconds to do ssh to a host (OPTIONAL)
  -v --verbose turn on warning and diagnostic messages (OPTIONAL)
  -O --options SSH options (OPTIONAL)

Example: pssh -h ips.txt -l irb2 -o /tmp/foo uptime
```

And for `pscp`:

```
# pscp
Usage: pscp [OPTIONS] -h hosts.txt local remote

  -h --hosts     hosts file (each line "host[:port] [login]")
  -r --recursive recusively copy directories (OPTIONAL)
  -l --user      username (OPTIONAL)
  -p --par       max number of parallel threads (OPTIONAL)
  -t --timeout   timeout in seconds to do scp to a host (OPTIONAL)
  -O --options   SSH options (OPTIONAL)

Example: pscp -h hosts.txt -l irb2 foo.txt /home/irb2/foo.txt
```

Note that before using any of these tools, you *will need to start ssh-agent*! This can be done as follows (substitute `zsh` with your particular shell).

```
# ssh-agent zsh
# ssh-add
Enter passphrase for /x/bnc/.ssh/identity:
```

3. Examples

3.1. pssh

The following example runs `hostname` on three machines (IPs or hostnames) specified in the file `ips.txt` using login `irb2` and saves the output in `/tmp/foo`.

```
# cat ips.txt
128.112.152.122
18.31.0.190
128.232.103.201

# pssh -h ips.txt -l irb2 -o /tmp/foo hostname
Success on 128.112.152.122:22
Success on 18.31.0.190:22
Success on 128.232.103.201:22

# ls /tmp/foo
128.112.152.122  128.232.103.201  18.31.0.190

# cat /tmp/foo/*
planetlab-1.cs.princeton.edu
planetlab1.xeno.cl.cam.ac.uk
planetlab1.lcs.mit.edu
```

By default, `pssh` uses at most 32 ssh processes in parallel to ssh to the various nodes. (This is somewhat important if you're controlling hundreds or thousands of machines.) By default, it also uses a timeout of one minute to ssh to a node and obtain a result. For ssh commands that take longer than this (e.g., `sleep 61`), the -t option can be used. Note that `pssh` and `pnuke` have a default timeout of one minute. `pscp` and `prsync` have no default timeout, but one can be specified using the -t option.

3.2. pscp

Here's an example of using `pscp` to copy files in parallel to a set of machines.

```
# pscp -h ips.txt -l irb2 /etc/hosts /tmp/hosts
Success on 128.112.152.122:22
Success on 18.31.0.190:22
Success on 128.232.103.201:22
```

Using the -r option will perform a recursive copy for copying entire directories.

3.3. pnuke

The `pnuke` command is useful when you want to kill a bunch of processes on a set of machines. For example, suppose you've got a bunch of `java` processes running on three nodes that you'd like to nuke (let's use the three machines from the `pssh` example). Here you would do the following:

```
# pnuke -h ips.txt -l irb2 java
Success on 128.112.152.122:22
```

```
Success on 18.31.0.190:22
Success on 128.232.103.201:22
```

The result of the above is to send `kill -9` to all processes owned by `irb2` with the string java in their name (as reported by `ps -ef`).

4. Environment Variables

All four programs take similar sets of options. All of these options can be set using the following environment variables:

```
PSSH_HOSTS
PSSH_USER
PSSH_PAR
PSSH_OUTDIR
PSSH_VERBOSE
PSSH_OPTIONS
```

Here are some example settings:

```
# export PSSH_HOSTS="/x/bnc/ips.txt"
# export PSSH_USER="irb2"
# export PSSH_PAR="32"
# export PSSH_OUTDIR="/tmp/bar"
# export PSSH_VERBOSE="0"
# export PSSH_OPTIONS="UserKnownHostsFile /tmp/known_hosts"
```

Using the above settings, the examples can be executed succinctly as:

```
# pssh hostname
Success on 128.112.152.122:22
Success on 18.31.0.190:22
Success on 128.232.103.201:22

# ls /tmp/bar
128.112.152.122  128.232.103.201  18.31.0.190

# cat /tmp/bar/*
planetlab-1.cs.princeton.edu
planetlab1.xeno.cl.cam.ac.uk
planetlab1.lcs.mit.edu

# pscp /etc/hosts /tmp/hosts
Success on 128.112.152.122:22
Success on 18.31.0.190:22
Success on 128.232.103.201:22

# pnuke java
Success on 128.112.152.122:22
Success on 18.31.0.190:22
Success on 128.232.103.201:22
```

5. Feedback

Send me email if you're having problems, find bugs, or have any random comments: Brent Chun (http://www.theether.org) (bnc at theether.org). Thus far, I've primarily been testing this software on PlanetLab (http://www.planet-lab.org).