What is Massh?

Massh is a mass ssh'er that allows for parallel execution of commands on remote systems. Massh makes updating and managing hundreds or even thousands of systems possible. In addition to massively parallelized command execution massh can push files in parallel and push/run scripts in parallel.

Massh also comes with **Pingz**, a mass ping'er that can also do mass DNS lookups. Finally, Massh comes with **Ambit**, a string expander that allows for both pre-defining groups of hosts as well as passing arbitrary strings that represent host groupings. The combination of Massh and Ambit creates a powerful way to manage groups of systems as configurable units.

Why Massh?

Massh has existed in one form or another for about 8 years. Over that time the lack of a fast, feature rich, generally available mass ssh'er kept Massh alive. During this time Massh was influenced by internally developed mass ssh'ers in use at various companies of employment. The focus has always been on a fast mass ssh'er that facilitates managing an environment of services, not servers. Finally, Massh's clean, organized output sets it apart from a lot of the mass ssh'ers in general use today.

Massh Quick Start

Massh Menu

Massh has a menu option that makes Massh'ing easy for everyone:

```
zombiebite $ massh -M

Massh Menu
1) Massh a Command
2) Massh a File
3) Massh a Script
4) Help Text
Choice? : 1

Enter a filename, full file path, abitrary range or pre-defined
range containing the hosts that you want to Massh to.

HOSTS? : [1,2].a.tt

Enter the command that will be Massh'ed to your remote hosts.

COMMAND? : id bill

Would you like Full Output or Simple Verification
1) Full Output
2) Simple Verification

OUTPUT TYPE? : 1
Running...
2.a.tt : uid=502(bill) gid=502(bill) groups=502(bill),10(wheel),25(named),101(puppet),10000(ssh),10001(svn),48(apache)
1.a.tt : uid=502(bill) gid=502(bill) groups=502(bill),10(wheel),25(named),101(puppet),10000(ssh),10001(svn),48(apache)
```

For The Rest of Us

1) Massh depends on having ssh public/private keys on all the machines that you will be connecting to. Not only does this make using massh inherently safer but it makes it easier to get things done. Generating SSH Public/Private key pairs is beyond the scope of this document.

2) Once you have SSH keys setup on your target machines simply copy massh and ambit to /usr/local/bin or somewhere in your $PATH.

3) Using Massh's range option (-r) connect to two machines and run uptime:

```
zombiebite $ massh -r web1.a.tt:web2.a.tt -c uptime
web2.a.tt : Succeeded
web1.a.tt : Succeeded
```

Note that the two machine names are separated by a colon (:). A colon is the designated range separator. Also note that the output of the command was supressed and substituted with the success of failure of the command passed. Let's try this again with a few different options:

```
zombiebite $ massh -r web[1,2].a.tt -c uptime -o
web2.a.tt : 23:49:24 up 45 days,  8:46,  0 users,  load average: 0.00, 0.00, 0.00
web1.a.tt : 23:49:25 up 48 days,  5:58,  1 user,  load average: 0.00, 0.00, 0.00
```

In this example, since the hosts are consecutively named the range can be simplified. Command output is shown via the -o command line argument. The default is to suppress full output unless specified. This default can be overridden via $HOME/.massh.

For the final example the following hostnames are added to a files called hosts.txt in /tmp:

```
1.a.tt
2.a.tt
3.a.tt
4.a.tt
web1.a.tt
web2.a.tt
web3.a.tt
web4.a.tt
```

This file can now be referenced by massh:

```
zombiebite $ massh -f /tmp/hosts.txt -c uptime -o
web2.a.tt : 23:58:19 up 45 days,  8:55,  0 users,  load average: 0.03, 0.01, 0.00
2.a.tt    : 23:58:19 up 45 days,  8:55,  0 users,  load average: 0.03, 0.01, 0.00
4.a.tt    : 23:58:19 up 45 days,  8:55,  0 users,  load average: 0.03, 0.01, 0.00
web4.a.tt : 23:58:19 up 45 days,  8:55,  0 users,  load average: 0.03, 0.01, 0.00
3.a.tt    : 23:58:19 up 48 days,  6:07,  1 user,  load average: 0.00, 0.00, 0.00
web1.a.tt : 23:58:19 up 48 days,  6:07,  1 user,  load average: 0.00, 0.00, 0.00
web3.a.tt : 23:58:19 up 48 days,  6:07,  1 user,  load average: 0.00, 0.00, 0.00
1.a.tt    : 23:58:20 up 48 days,  6:07,  1 user,  load average: 0.00, 0.00, 0.00
```

Ideally you should be dangerous enough to stay busy for a while. Read on to become more dangerous.

Running Massh

The first thing to do is run massh with no options:

```
zombiebite $ massh
Usage:
      massh [-f file | -r range] [-c cmd | -s script | -p file]
            [-o -D [-S -R] [-F -R]]
      -f        File containing hostnames or ranges
                the following directories:
                    * /usr/local/var/massh
                    * /Users/mmarscha/massh
                    * $CWD, full path, relative path
      -r        Arbitrary or file defined host groupings
      -c        Command to run. Quote commands to insure execution.
      -s        Script to push to all hosts and run.
      -p        File to push to all hosts.
      -o        Full, hostname tagged output
      -P        Number of parallel session. Default is 50
      -F        Output only hostnames where remote command failed.
      -S        Output only hostnames where remote command succeeded.
      -R        Regurgitate F,S output and feed back to massh -r
      -D        Debug
      -M        Massh Menu
      -O        SSH options in additinon to the following:
                 -o StrictHostKeyChecking=no -o LogLevel=QUIET
                 -o BatchMode=yes -o ConnectTimeout=5
Output:
Succeeded  - The command, push, push/run ran with no errors.
   Failed  - The connection, command, push, push/run failed.
  Skipped  - Host was listed in one of the following files:
                    * /usr/local/var/massh/hosts.down
                    * /Users/mmarscha/massh/hosts.down
Examples:
    massh -f /tmp/hosts.txt -c 'cat /etc/passwd | grep ^root:' -o
    massh -r dbservers -p ~/.hushlogin
    massh -r web.[1,5,[10..15]].google.com -c 'last | head -10' -o
    massh -r webservers:dbservers:appservers -c 'df -h' -o
```

Massh Files

Massh Files are files located the system wide directory: /usr/local/var/massh or in user specific directories: $HOME/massh that are prefixed with "hosts." "file." and/or "script." that Massh can make use of in various ways (outlined later in this document). Massh Files allow Systems Administrators to easily organize large scale command execution, configuration management, file syncronization and remote script pushing/running.

Ambit Ranges

Ambit is used by Massh for interpolating ranges passed to the (-r) Massh command line argument. Ambit take the string passed to the (-r) command line argument and compare it to files prefixed with "hosts." in /usr/local/var/massh and $HOME/massh. This means that a file with all target hosts call "hosts.all" in /usr/local/var/massh or $HOME/massh can be referenced simply by the name all on the command line:

```
zombiebite $ cat $HOME/massh/hosts.all
web1.a.tt
web2.a.tt
web3.a.tt
web4.a.tt
zombiebite $ massh -r all -c uptime -o
web2.a.tt : 01:56:35 up 45 days, 10:53,  0 users,  load average: 0.00, 0.00, 0.00
web4.a.tt : 01:56:35 up 45 days, 10:53,  0 users,  load average: 0.00, 0.00, 0.00
web1.a.tt : 01:56:35 up 48 days,  8:05,  1 user,  load average: 0.00, 0.00, 0.00
web3.a.tt : 01:56:35 up 48 days,  8:05,  1 user,  load average: 0.00, 0.00, 0.00
```

Ambit will also interpolate ranges on the command line or as entries in "hosts." files:

```
zombiebite $ massh -r web[1..4].a.tt -c uptime -o
web2.a.tt : 01:59:41 up 45 days, 10:56,  0 users,  load average: 0.00, 0.00, 0.00
web4.a.tt : 01:59:41 up 45 days, 10:56,  0 users,  load average: 0.00, 0.00, 0.00
web3.a.tt : 01:59:42 up 48 days,  8:08,  1 user,  load average: 0.00, 0.00, 0.00
web1.a.tt : 01:59:42 up 48 days,  8:08,  1 user,  load average: 0.00, 0.00, 0.00
zombiebite $ cat $HOME/massh/hosts.all
web[1..4].a.tt
zombiebite $ massh -r all -c uptime -o
web2.a.tt : 02:00:28 up 45 days, 10:57,  0 users,  load average: 0.06, 0.02, 0.00
web4.a.tt : 02:00:28 up 45 days, 10:57,  0 users,  load average: 0.06, 0.02, 0.00
web3.a.tt : 02:00:29 up 48 days,  8:09,  1 user,  load average: 0.00, 0.00, 0.00
web1.a.tt : 02:00:29 up 48 days,  8:09,  1 user,  load average: 0.00, 0.00, 0.00
```

Ambit can also interpolate multiple ranges, either file based or arbitrary strings at the same time, separated with a colon (:):

```
zombiebite $ cat $HOME/massh/hosts.all
web[1,2].a.tt
zombiebite $ massh -r all:web[3,4].a.tt -c uptime -o
web2.a.tt : 02:02:27 up 45 days, 10:59,  0 users,  load average: 0.01, 0.00, 0.00
web4.a.tt : 02:02:27 up 45 days, 10:59,  0 users,  load average: 0.01, 0.00, 0.00
web1.a.tt : 02:02:28 up 48 days,  8:11,  1 user,  load average: 0.00, 0.00, 0.00
web3.a.tt : 02:02:28 up 48 days,  8:11,  1 user,  load average: 0.00, 0.00, 0.00
```

Simple Output of Remote Commands

The default output type for Massh is to give a simple Succeeded or Failed for the command run with the (-c), on the hosts specified with either the -f or -r option:

```
zombiebite $ massh -r web[1..4].a.tt -c uptime
web2.a.tt : Succeeded
web4.a.tt : Succeeded
web3.a.tt : Succeeded
web1.a.tt : Succeeded
```

The reason for doing this is that full output of a command can often times be way more than intended. As a default behavior it seems less obnoxious to see a "Succeeded" or "Failed" than 500 lines of returned output from each host because the following was run:

```
massh -r web[1..4].a.tt -c dmesg
```

Full Output of Remote Commands

Of course a simple "Succeeded" or "Failed" does not suffice and thus there is the (-o) option for full output. The same command above with full output returns:

```
zombiebite $ massh -r web[1..4].a.tt -c uptime -o
web2.a.tt : 00:31:22 up 45 days,  9:28,  0 users,  load average: 0.00, 0.00, 0.00
web4.a.tt : 00:31:22 up 45 days,  9:28,  0 users,  load average: 0.00, 0.00, 0.00
web1.a.tt : 00:31:23 up 48 days,  6:40,  1 user,  load average: 0.00, 0.00, 0.00
web3.a.tt : 00:31:23 up 48 days,  6:40,  1 user,  load average: 0.00, 0.00, 0.00
```

Pushing Files

Massh can be used to push a file specified with the (-p) option to the hosts specified with the (-r) or (-f) option:

```
zombiebite $ massh -r [1,2].a.tt -p tmp/hosts.down
2.a.tt : Succeeded
1.a.tt : Succeeded
```

Using Massh Files, pushing can be done in a more organized way. All files located in /usr/local/var/massh or $HOME/massh that are prefixed with "file." can be referenced via the file's suffix. For instance the file $HOME/massh/file.httpd.conf can be pushed with the following command:

```
zombiebite $ massh -r all -p httpd.conf
web2.a.tt : Succeeded
web1.a.tt : Succeeded
```

Pushing and Running Scripts

The first versions of Massh were actually created to push and run scripts to remote machines. This remains a very nice feature and is particularly useful when a single, over piped, run-on command just will not do. This option (-s) can be used with and without full output. The (-s) option will **push** a script (prefixed with 'script.'), **run** the script and then **removes** the script from the remote host. Be sure to set the script's mode so that it can be executed. Here's an simple script that writes a date and time string to a file as well as echo's it to standard output:

```
#!/bin/bash

# Name: time
# Author: Michael Marschall
# Modified: 2008.04.25

# I use this to check massh's script pushing and executing.
date +%Y%m%d%H%M > time.txt
date +%Y%m%d%H%M
```

Running it results in:

```
zombiebite $ massh -r [1,2].a.tt -s ~/massh/script.time -o
2.a.tt : 201008090131
1.a.tt : 201008090131
```

And here we verify that it wrote output to the file time.txt:

```
zombiebite $ massh -r [1,2].a.tt -c 'cat time.txt' -o
2.a.tt : 201008090127
1.a.tt : 201008090127
```

As with pushing files, Massh Files located in /usr/local/var/massh or $HOME/massh that are prefixed with "script." can be referenced by their suffix on the command line:

```
zombiebite $ massh -r web[1,2].a.tt -s time -o
web2.a.tt : 201008090252
web1.a.tt : 201008090252
```

hosts.down

The Massh File hosts.down allows for the arbitrary skipping of hosts by Massh. Massh will check /usr/local/var/massh/hosts.down and $HOME/massh/hosts.down to see if a host should be skipped by a massh run for whatever reason. In my current position we us pingz to check the hosts in /usr/local/var/massh/hosts.all (which represents the master list of hosts for which my team is responsible) and populate /usr/local/var/massh/hosts.down with the hosts that fail their ping check. This happens every

10 minutes. The populating of hosts.down can and will be expanded to monitoring check results in addition to ping checking.

Success, Failure, hosts.results and Regurgitate

The command line arguments (-S | -R) allow for the gathering of hosts that a command either succeeded or failed on. If (-S) is used then all the hosts that the command succeeded on will be printed to standard output and will be added to $HOME/massh/hosts.results. Additionally the date, time and full command will be added to $HOME/massh/hosts.results:

```
zombiebite $ massh -r [1,2].a.tt -c 'hostname | grep 1' -S
1.a.tt
zombiebite $ cat hosts.results
# 2010-08-09 00:19:42 massh -r [1,2].a.tt -c hostname | grep 1 -S
1.a.tt
```

Capturing the hosts that either succeeded or failed a Massh run can be very useful. When pushing code to hundreds of machines it is very useful to have a ready list of the host that failed the push. In addition to having the hosts list to standard output and captured in hosts.results, the Regurgitate option allows for the hosts to be outputted in an ambit friendly format - which allows for the feeding of Massh output back into Massh for a two step operation.

For a quick example, let's say I have a file on one of two web servers. I do not know which server has the file so I will check to see where the file is and upon finding it, use the -S and -R option to cat this file only on the server that has it. This will be done in one command:

```
zombiebite $ massh -r `massh -r web[1,2].a.tt -c 'test -f himom' -S -R` -c 'cat himom' -o
web2.a.tt : UrDuh Best Mommy
```

Syslog Logging

Massh will try to use logger to log to syslog for each run of Massh. Here is an example of such logging:

```
Aug  9 00:42:27 bur-asomon01 massh[4982]: USER=mmarscha COMMAND=massh -r sk11adui -s time
```