Stacki Tutorial #3: Automating Installs with CSV Files

Introduction

Stacki is capable of performing fully unattended installs of server operating systems. To achieve this level of automation, Stacki takes advantage of several CSV-formatted files that tell the Stacki server about your backend servers, giving Stacki the information it needs to perform the installations while you go get lunch.

What you will learn

This short tutorial will focus on the CSV format that defines hosts Stacki will be installing.

Specifically, when done with this tutorial, you will be able to:

- Create a CSV file with the relevant information to perform an install.
- Load the CSV file into Stacki.
- Kick off the installation or re-installation of one or many servers.
- Tell Stacki about additional network cards.

While Stacki is capable of much more install customization than discussed here, and there is an entirely different method available to do quick installs, this tutorial will focus solely on CSV host definition and installation. This will keep it in line with the goal of the series – to offer bite-sized information that can be used today.

A more in-depth treatment of server installations is available on the Stacki Wiki for those who need more than this tutorial offers.

What Stacki Does For Installs

Stacki's purpose in life is to install machines accurately, repeatably, and quickly. This tutorial touches on the very heart of what Stacki is.

To achieve automated installation, Stacki uses Pallets. A Pallet is a parameterized collection of all the software necessary to spin up a given type of server, and a set of ordering instructions to inform Stacki about which steps to perform, in what order.

Stacki does much more than simply spin up an ISO install and wait for input. Parameterization eliminates the need for administrators to sit and answer installer prompts, while the software stack in the Pallet defines much more than one might expect. Passwordless ssh between the Stacki server and the backend servers is configured by default, for example. Stacki comes with a Pallet to fully install CentOS. Users who are interested in expanding the installation to include more functionality can look on the Wiki at the Extend Backend Nodes heading.

Prerequisites

This tutorial assumes the available environment includes:
An installed Stacki server that meets the requirements defined on the Wiki
At least one (physical or virtual) server connected to the network Stacki defines as “private”.
For the advanced section of this tutorial, the backend server should have a second NIC physically connected to the network Stacki defines as “public”.

The CSV File
Stacki requires certain information about each server to be installed. One way of giving Stacki that information is CSV files. Since we are focused on CSV-enabled installations, we’ll start by looking at a simple one in a spreadsheet tool:

Name	Appliance	IP	MAC	Interface	Rack	Rank	Subnet
backend-0-0	backend	10.1.1.155	08:00:27:49:B4:A7	eth0	0	2	private

This CSV file defines a single server to be installed. It contains the following fields:

- **Name** – the hostname to associate with this server.
- **Appliance** – the type of machine to install. By default, Stacki only has backend for Appliance type, though there can be several types in the system at once.
- **IP** – the IP address to assign the primary NIC – the one on the “private” network.
- **MAC** – the MAC address to associate the IP with. Note that Stacki uses the MAC address to determine which machines to install, so make certain this field is accurate.
- **Interface** – what Linux interface to assign this MAC to.
- **Rack** – the rack this server resides in.
- **Rank** – the location in the rack.
  - Note that Rack and Rank are for you to identify machines quickly if problems occur.
- **Subnet** – the Stacki subnet the NIC we just defined should be defined on. The first NIC should be on the private network, as that’s the network installations occur over.

Our CSV file

We are going to use a CSV formatted exactly like the one above, but with a couple of backends in it. Here is the file this tutorial will use:

This CSV defines two backend hosts. Note the difference in naming style – any valid hostname can be used in the “name” field.

Name	Appliance	IP	MAC	Interface	Rack	Rank	Subnet
backend-0-0	backend	10.1.1.253	08:00:27:6E:88:08	eth0	0	2	private
CocoaMaster357	backend	10.1.1.252	08:00:27:5F:12:1C	eth0	0	1	private

Load The Host File

Now that the hosts have been defined, this file needs to be loaded into Stacki so the server knows about them.
First, the CSV we've defined needs to be physically on the Stacki server. In Linux and iOS, scp is probably the easiest way to transfer the file. On Windows, a tool like WinSCP or Filezilla is probably easiest. However you get the file transferred, put it into root's home folder (/root).
Next, lets load the file into Stacki so it knows about the servers:
For the tutorial's purposes, let's look at the list of hosts on the Stacki server first:

```
[root@donrh7 ~]# stack list host
HOST         RACK RANK CPUS APPLIANCE DISTRIBUTION RUNACTION INSTALLACTION
donrh7:      0    0    1    frontend  default      os        install
backend-0-0: 0    0    1    backend   default      os        install
```

Notice that the Stacki server already has a machine named backend-0-0. That is not a problem for Stacki. If the information in the incoming spreadsheet is changed, the data will be updated. If it hasn't changed, Stacki will do nothing. For this tutorial, we will focus on the new server defined in the spreadsheet – CocoaMaster357 (for those curious about where that name came from, see this blog), as backend-0-0 is in the spreadsheets and system solely to show how multiple servers can be defined.

Now we're ready to load the host file into stacki. We accomplish this with the following command:

```
[root@donrh7 ~]# stack load hostfile file=stacki.cocoamaster.csv
/export/stack/spreadsheets/RCS/stacki.cocoamaster.csv,v  <--  /export/stack/spreadsheets/stac-
ki.cocoamaster.csv
initial revision: 1.1
done
/export/stack/spreadsheets/RCS/stacki.cocoamaster.csv,v  -->  /export/stack/spreadsheets/stac-
ki.cocoamaster.csv
revision 1.1 (locked)
done
[root@donrh7 ~]#
```

Note that Stacki maintains versions of hostfiles. While this tutorial will not go into uses of this versioning, it can be good knowledge to have.
The output tells us that revision 1.1 has been loaded, and locked.

After the above command has loaded the host file, Stacki is now aware of the new server. We can see that by executing stack list host again.

```
[root@donrh7 ~]# stack list host
HOST          RACK RANK CPUS APPLIANCE DISTRIBUTION RUNACTION INSTALLACTION
donrh7:         0    0    1  frontend  default      os        install
backend-0-0:    0    0    1  backend   default      os        install
cocoamaster357: 0    1    1  backend   default      os        install

[root@donrh7 ~]#
```

Notice here that cocoamaster357 has been added to the list of hosts. It is not installed yet, but Stacki has it in the database, and is ready to install it. Except we have to tell Stacki to do so. It will not automatically assume you want to install a server – because installs are destructive. That is a design view that permeates Stacki – things that are destructive are not done without administrators explicitly telling Stacki to perform the task. This approach was selected to protect systems from accidental harm.

Next, tell Stacki that it should install cocoamaster357:

```
[root@donrh7 ~]# stack set host boot cocoamaster357 action=install
[root@donrh7 ~]# stack set host attr cocoamaster357 attr=nukedisks value=true
[root@donrh7 ~]#
```

The first command is the only mandatory one – it tells Stacki to go ahead and install cocamaster357 the next time it boots. When the MAC we defined next boots on the private network, Stacki will go ahead and install it.

The second command is optional, but recommended for new installs. It tells Stacki to wipe all of the disks on cocoamaster357. By default, an install only reformats partitions that it is installing software on, this command tells Stacki to wipe all of the partitions.

The “server name” field in the above commands is very explicit by hostname for this tutorial. If you are installing multiple servers, you need not list each one, see the “Naming of Hosts/Host Groups” callout in the first tutorial.

Reinstalling

If a machine’s software ever becomes unstable, or a failure of disk or addition of hardware make it need a reinstall, these commands can also be used for a reinstall. Just like the initial installation, the next time the server boots it will reinstall after the set host boot command has been run again.

Performing the Installation

That is it. Cocoamaster357 is set up for installation, and will install on next boot. If you are following along to perform an install, go ahead and boot your target server(s), and they'll start installing. Should something unforeseen happen during the install – a hardware failure or network issues, for example, Stacki will re-attempt the install on successive boots. If the installation completes successfully, Stacki will automatically change the boot action to be "os" which means "run the locally installed os", bringing your server online at the end of the installation.

Multiple NICs

Many organizations want to define NICs on multiple networks, so that the private connection is used for installation, but a public connection can be used to communicate with the rest of the network. There are many things Stacki can do with networking, and we will cover them in other tutorials, but for this example, we will simply add a NIC on the public network to the server. First we need to add some information to our hosts CSV file:

Name	Appliance	IP	MAC	Interface	Rack	Rank	Subnet
backend-0-0	backend	10.1.1.253	08:00:27:6E:88:08	eth0	0	2	private
CocoaMaster357	backend	10.1.1.252	08:00:27:5F:12:1C	eth0	0	1	private
CocoaMaster357		192.168.0.218	08:00:27:98:26:8C	eth1			public

Here we have added a line underneath the server we're modifying. That line defines a new NIC on an existing server. Note that the Name field is the same, and the machine-specific information (Appliance type, rack, and rank) are not supplied. Otherwise, we fill in the IP, MAC, Interface, and Subnet fields. That's it. This can be done for as many servers as have multiple NICS in them, and since we're just starting, we'll reinstall to get them live.

Recall that the same command can be used to reinstall as to install, so next we will instruct Stacki to reinstall this server when next it boots onto the private network:

```
[root@donrh7 ~]# stack set host boot cocoamaster357 action=install
[root@donrh7 ~]#
```

That does it. A reboot will reinstall the server, and both NICs will be configured and come up.

Recall that password-less ssh is configured by default when Stacki server installs a server. We'll use that to check on the IP configuration of our reinstalled server:

```
[root@donrh7 ~]# ssh cocoamaster357
Last login: Wed Jul 22 15:28:50 2015
[root@cocoamaster357 ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.1.252  netmask 255.255.255.0  broadcast 10.1.1.255
        inet6 fe80::a00:27ff:fe5f:121c  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:5f:12:1c  txqueuelen 1000  (Ethernet)
        RX packets 62  bytes 10273 (10.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 84  bytes 9436 (9.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.218  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::a00:27ff:fe98:268c  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:98:26:8c  txqueuelen 1000  (Ethernet)
        RX packets 174  bytes 16219 (15.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 888 (888.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
[root@cocoamaster357 ~]#
```

Summary

This tutorial showed how to set up a CSV file that defines one or more servers, how to load the CSV file, and perform an install. In addition to these items, the tutorial showed how to wipe the partitions on the server not being installed, and how to add a NIC to the system via the CSV file.

Adding servers to an environment with Stacki is quick and effective, with no need to answer install questions or fill in the missing information from a golden image. The servers installed with Stacki have passwordless ssh, host firewalls, and all of the software necessary for a base Linux installation, all out-of-the-box.