USENIX Association

# Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta

Atlanta, Georgia, USA
October 10–14, 2000

# VA SystemImager

**Brian Elliott Finley <brian@valinux.com>**

## Abstract

Linux use in corporations and research organizations has been growing at an amazing rate. It is often used on large numbers of identical systems serving as Internet server farms or high performance computing clusters. Without the help of specialized tools, the time and effort required to install and maintain large numbers of machines grows almost linearly as new systems are added. This creates the demand for a tool with the ability to automate the installation of new systems and maintain the software, configuration, and content of those systems on an ongoing basis.

System administrators at large sites will often develop tools for automating the deployment and update of their own systems, but these tools are often very inflexible and are only designed to address the specific needs of one particular set of systems. Therefore these tools are often re−created by system administrators at site after site, not being able to capitalize on the work of their neighbors. The need was perceived for a tool that could provide this functionality at many different sites with different configurations. This required that the tool be easy to install, simple and straightforward to use, and that it be designed in an open and extensible manner to accommodate future changes and site specific customizations.

This paper describes the resultant tool, VA SystemImager. It is Open Source software and is designed in a very modular manner. Great pains were taken to ensure that it would be flexible and could easily be modified to accommodate new hardware, software, and site specific configuration needs in future iterations. VA SystemImager is written mostly in Perl and makes use of rsync(1), syslinux(2), and pxelinux(2). It also required the creation of a customized miniature Linux distribution for the installation media. This paper will also discuss some of the differences between VA SystemImager and the KickStart network installation tool from RedHat. KickStart is the tool most often compared to VA SystemImager. Although there are a handful of other tools available, none of them offer the flexibility and ease of use of VA SystemImager.

## Extended Abstract

**Design Goals**

− Images should be pulled from an already running system.
− Completely unattended installs were a must.
− The unattended install system had to be able to repartition the destination drive(s).
− One of the main design goals was ease of use. This had to be a tool that could be used
by a system administrator that didn’t necessarily understand how it worked.
− It was also necessary for it to install easily and quickly so that it could be useful right away without a lot of site specific customization.
− Images should be stored as normal files in appropriately named directories as opposed to "dd" style block level images of physical disks.
− Not everyone who needs to install a lot of Linux systems uses the same distribution, so it had to be independent of any and all packaging systems (such as RPM).
− It should be able to store multiple images, for different types of systems and for revision control, and provide a mechanism for unattended install clients to know which image to install.
− Once a client was installed, it should be able to update itself to a new or updated image.
− It should easily accommodate different distributions.
− It should have a command line interface designed such that it could later be wrapped with a GUI.
**Some VA SystemImager terms and commands that will be referred to in this abstract:**

− autoinstall client − A machine on to which Linux is to be installed using the VA SystemImager automated process.
− autoinstall media − The media that is used to boot an autoinstall client in order to begin the autoinstall process. This media can be a floppy, a CDROM, the network, or the local hard drive of the autoinstall client.
− updateclient − A command that is executed on client systems allowing them to be updated or synchronized to a new or updated image.
− imageserver − The machine that will hold the images.
− master client − The machine that has been manually

installed and configured the way you want your image to look.
− getimage − This command is run from the imageserver to pull a system image from a master client.
− prepareclient − This command is run on the master client immediately prior to running getimage on the imageserver.
− makedhcpserver − Used to create the /etc/dhcpd.conf file. DHCP can be used to assign IP addresses to autoinstall clients.
− makedhcpstatic − Used to re−write the /etc/dhcpd.conf file, adding static entries based on the IP addresses already handed out to all of the autoinstall clients.
− autoinstall script − A unique autoinstall script is created for each image and is used by the autoinstall client as part of the autoinstall process. The names of these scripts begin with the image name and end in .master. For example: "my_webserver_image_v1.master"
− addclients − Creates a soft link to the master autoinstall script with the name of each host that will receive that image. It also allows you to populate the /etc/hosts file with sequential host names and IP addresses.

### Resultant Architecture

VA SystemImager began as a series of utilities written in bash. Minimal system requirements were considered a top priority. As VA SystemImager matured, and the utilities became internally more complex, it became clear that bash was falling short of the need. Perl was chosen to pick up the yoke and has allowed for cleaner, more advanced code. It was determined that Perl is installed as part of most "base" Linux installs and therefore was a reasonable choice from a minimal requirements perspective.

The architecture was designed to be open to future modification at every level. The protocol used for transferring files during installs and updates is currently rsync(1). But the modular code will easily allow for a drop−in replacement using mftp(5) or other appropriate file transfer utilities. All file transfer mechanisms are implemented in a "pull" fashion, which is generally considered to be superior to a "push". Using a "pull" mechanism, it is much easier to monitor the state of the receiving system prior to and during the file transfers.

There are other methods available for doing automatic installs, such as RedHat's KickStart which installs systems based on a list of pre−defined packages. But package based installs are very limiting in that they generally don't have an automated way for dealing with non−packaged files. If you re−compile your kernel, add a piece of non−packaged software, or modify certain configuration files, you are usually required to do some sort of scripting or programming to deal with these "special cases".

In order keep imaging simple, VA SystemImager uses images that are based on a working installed system. We call this system a "master client". Just get one of your machines working exactly the way you want and pull it's image to the imageserver with the "getimage" command. You can re−compile your kernel, install custom software, and do any configuration file tweaking you like. VA SystemImager will get it all.

Now that you have your master client configured, we need to run the "prepareclient" command. prepareclient will collect the partition information from your disks and put it in the /etc/partitionschemes directory. A file will be created in this directory for each of your disks and will contain that disks partition information. prepareclient will also creates an rsync(1) configuration file (/etc/rsyncd.conf) and starts rsync in server mode (rsync −−daemon). This allows the imageserver to pull the image from the client, but will not cause the rsync daemon to be restarted after the master client is rebooted. This helps avoid security concerns of sharing a master client's root filesystem via rsync. rsync has the ability to use OpenSSH(6) as an alternate shell and plans are in place to modify VA SystemImager to run all operations over OpenSSH(6) for security purposes.

On the imageserver we now run the getimage command. Here's an example: "getimage −master−client=192.168.1.1 −image=my_webserver_image_v1" getimage contacts the master client and requests it's /etc/mtab file. This file contains the list of mounted filesystems and the devices on which they are mounted. It pulls out the mount points for the filesystems that are unsupported and creates an exclusion list. Currently supported filesystems are ext2 and reiserfs. Unsupported filesystems are things like proc, devpts, iso9660, etc. getimage then pulls the master client's entire system image, excluding the filesystems in the exclusion list. The files are pulled by connecting to the rsync(1) daemon running on the master client. All the files from the client will be copied over, recreating the filesystem and directory hierarchy in the image directory.

getimage can also be used to update an existing image. By simply specifying an existing image name, you are asking getimage to update that image to match the files on you master client. In this case, only the files that are different will be copied over. Files that exist in the old image but not on the master client will be deleted, and files that exist in both places but have changed will be updated. This is one way to keep an image updated when new security patches or other system updates come out. However, the recommended method is to never overwrite a known working image, so that you have a form a revision control. This is not true revision control, where individual file revisions are tracked on a line by line basis. It is more of a revision control on an image by image basis. This form of revision control also ties in to the updateclient command which will be discussed later. By default, all images are stored in the parent directory of "/var/spool/systemimager/images/" in a directory that bears the image name. For example: "/var/spool/systemimager/images/my_webserver_image_v1/".

After getimage has pulled the files to the image directory on the imageserver it creates a customized autoinstall script. The master script in this case would be named "my_webserver_image_v1.master". All autoinstall scripts are placed in the "/tftpboot/systemimager/" directory. The disk partitioning information left behind by the prepareclient command is used to add the necessary commands to re−partition the disk(s) on the autoinstall clients. Filesystem information taken from the /etc/fstab file in the image (Ie.: "/var/spool/systemimager/images/my_webserver_image_v1/etc/fstab") and is used to determine the appropriate filesystem creation commands and to determine mount points for the autoinstall process. Based on command line options passed to getimage or questions it has asked, certain networking information is added to the autoinstall script. This information is added in variable form as the autoinstall client will later determine the values for things such as it's hostname and IP address.

When running getimage interactively, it will prompt you to run the addclients command. addclients will ask you for the series of hostnames that you will be installing by combining a base host name and a number range. For example, if your base host name is "www", and your number range is from "1" to "3", then the resultant host names would be "www1, www2, www3". It will then prompt you to choose the image that will be installed to these hosts and will create soft links for each hostname that point to the master script for that image. For example: "www3.sh −> web_server_image_v1.master". If the image is updated and you choose to allow getimage to also update the master autoinstall script, then each of the associated soft links therefore point to the new master script. If individual host configuration is necessary, the soft link for that host can be removed and replaced with a copy of the master script that can then be customized for that host. This customization is a manual process and is up to the administrator of the system. addclients will then prompt you for the IP address information for these hosts and will re−write the imageserver's /etc/hosts file accordingly and copy this file to /tftpboot/systemimager/hosts. The latter file is used during the autoinstall process if the clients are using DHCP to obtain their IP addresses.

The unattended install portion is flexible and can work with most any hardware available. It is also easily modified to work with new or special hardware. A miniature Linux distribution is used for the boot media for "autoinstalls" (unattended installs). It consists of a customized kernel and an initial ram disk. The same kernel and initial ram disk (initrd.gz) can be used to boot off floppy disks, CDROMs, the network, or a running system's local hard drive. The commands "makeautoinstalldiskette" and "makeautoinstallcd" make use of the syslinux(2) utility to create floppies and CDROMs that will boot the VA SystemImager kernel and initial ram disk. pxelinux(2), which is a sister tool to syslinux(2), allows the same kernel and initial ram disk to boot PXE capable machines off the network. A configuration file is needed by syslinux(2) and by pxelinux(2), but VA SystemImager handles this for you and the two tools are able to use the same configuration file.

The autoinstall client is a miniature Linux distribution that has been customized to contain the specific commands and utilities necessary to perform autoinstalls to clients. The kernel is compiled to contain all the necessary drivers for a majority of systems. Custom kernels can be compiled to match special configurations. To use a custom compiled kernel, simply copy it to /tftpboot/kernel. All of the autoinstall media is created from /tftpboot/kernel and /tftpboot/initrd.gz. syslinux is used to load the initial ram disk and to boot the kernel when using an autoinstall diskette or an autoinstall CD. pxelinux is used to load the initial ram disk and to boot the kernel when using network booting.

Once the kernel has booted, it mounts the initial ram disk as it's root filesystem. It then executes an initialization script that has been customized to do VA SystemImager specific things. This script will use DHCP to get the autoinstall client's IP address information. It makes the assumption that the DHCP server is the imageserver and contacts it to request the utilities that would not fit in the initial ram disk. It copies these utilities to another ram disk that is mounted as /tmp1. It then requests a hosts file from the imageserver (the one in /tftpboot/systemimager) and parses this file to find it's IP address in order to determine it's hostname. Finally it requests an autoinstall script from the imageserver based on this hostname and executes it. The autoinstall script is image specific. This is how a client determines which image it will receive.

The most common way to assign IP addresses to the autoinstall clients is DHCP. To easify the configuration of the DHCP configuration file (/etc/dhcpd.conf) VA SystemImager includes a utility called makedhcpserver. makedhcpserver will prompt you for all the necessary information to create a DHCP configuration file that is appropriate for VA SystemImager. It is also possible to continue to use DHCP to assign static IP addresses to your clients after installation. If you choose to do so, simply run the makedhcpstatic command. It will rewrite your /etc/dhcpd.conf file on the imageserver to contain static entries for each of your hosts.

Alternately, hostname, imageserver, and networking information can be put in a configuration file on a floppy diskette. When the autoinstall client boots, it will look for this file on the floppy and use the provided values instead of determining them dynamically. This will work with any of the autoinstall media. The configuration file can even be put on the autoinstall floppy itself! The format of this configuration file is simply VARIABLE=value for all the appropriate variables. The name of this file must be local.cfg and it must exist on the root of the floppy. The floppy can be formatted with either ext2 or fat. An example local.cfg file can be found with the documentation files which are installed in /usr/doc.

Sometimes you will want to update an image on your imageserver. There are a couple of ways to do this. The first way is do directly edit the files in the image directory. The best way to do this is to chroot into the image directory. Once you have done the chroot, you can work with the image as if it were actually a running machine. You can even install packages with RPM, for example. The second way is to run the getimage command again, specifying a master client that has been modified in the desired way. Only the files that have changed will be pulled across. Files that have been deleted on the master client will also be deleted in the image. You are also given the option to update the master autoinstall script for the image or to leave it alone. The advantages of this method are that you can verify that your new configuration works on the master client, and that the master autoinstall script is updated.

Once a system has been autoinstalled, the updateclient command can be used to update a client system to match a new or updated image on the imageserver. Let's say that you've installed your companies 300 web servers and a security patch comes out the next day. You simply update the image on the imageserver and run updateclient on each of your 300 web servers. Only the modified files are pulled over, and your entire site is patched! It is recommended that you create an entirely new image with a new version number so that you have a form of revision control. This way, if you find out that the patch you applied hosed your entire web farm, you simply do an updateclient back to the know working image!

By incorporating some modifications sent in by A.L. Lambert, using the "updateclient" command with the −autoinstall option will copy the autoinstall kernel and initial ram disk to the local hard drive of the client. It will then re−write the /etc/lilo.conf file to include an appropriate entry for the new kernel and initial ram disk and specify this new kernel as the default using the "−D" option. The next time the client system is booted, it will load the VA SystemImager kernel and initial ram disk, which will begin the autoinstall process! This means that you can re−install any running Linux machine without having to have someone feed the machine a floppy or CD, and without having to reconfigure the BIOS to boot off the network (which can be quite squirrelly with some BIOSes).

**Summary of Steps**

1) Install the VA SystemImager software on the machine chosen to be the imageserver and configure the machine to be a DHCP server using the "makedhcpserver" command.
2) Choose the "master client" and make any desired changes to this system including recompiling the kernel, installing software, and tweaking configuration files.
3) Install the VA SystemImager client software on the client, which prepares it for having it's image pulled.
4) Issue the "getimage" command from the imageserver, specifying the master client and the name to assign to the resultant image.
5) Create autoinstall media for floppy installs using the "makeautoinstalldiskette" command, or for CD installs using the "makeautoinstallcd" command. Network boot clients should be told to boot off the network.
6) Boot the autoinstall client off the chosen media and watch it autoinstall!

**VA SystemImager is available for download from:**

**http://systemimager.sourceforge.net/**

## Future Improvements

VA SystemImager is under active development. Many new features are being added by the core developers and by end users. Some of the more notable future improvements are:

− Support for software RAID on autoinstall clients.
− A multicast install utility is being developed that will allow an unlimited number of autoinstall clients to be installed in parallel.
− See http://systemimager.sourceforge.net/TODO for others.

## Acknowledgements

VA SystemImager was conceived and developed by Brian Finley. It’s initial implementation was known as pterodactyl and was used for software and password updates to Solaris boxes of varying hardware and OS versions across a nationwide enterprise network. Over time it evolved into the Linux specific autoinstall and update tool that it is today. Many of the design decisions for VA SystemImager were based on perceived shortcomings in other automated install tools for systems such as Solaris, RedHat Linux, and Windows.

Other people who have contributed code or documentation that has been incorporated (alphabetical order):
− Susan Coghlan <smc@acl.lanl.gov>
− Paonia Ezrine <paonia@home.welcomehome.org>
− Michael Jennings <mej@valinux.com>
− Ari Jort <ajort@valinux.com>
− Ian McLeod <ian@valinux.com>
− Michael P. McLeod <mmcleod@bcm.tmc.edu>
− Michael R. Nolta <mrnolta@princeton.edu>
− Laurence Sherzer <lsherzer@gate.net>
− Wesley Smith <wessmith@engr.sgi.com>

Other people who have contributed ideas and suggestions (alphabetical order):
− Ted Arden <ted@valinux.com>
− Tanmoy Bhattacharya <tanmoy@lanl.gov>
− Susan Coghlan <smc@acl.lanl.gov>
− Steven Duchene <sad@valinux.com>
− Stephen Greene <sgreene@valinux.com>
− Bartosz Ilkowski <barbi@danforthcenter.org>
− Ari Jort <ajort@valinux.com>
− Brian Luethke <luethke@msr.epm.ornl.gov>
− Chip Salzenberg <chip@valinux.com>
− Robert Saft <zardoz@valinux.com>

## References

(1) rsync −− remote synchronization/update protocol. rsync is used to transfer files from the imageserver to an autoinstall client or a client using the "updateclient" command.

rsync was written by Andrew Tridgell <tridge@samba.org> and Paul Mackerras <Paul.Mackerras@cs.anu.edu.au>.

http://rsync.samba.org/

(2) syslinux −− syslinux is a boot loader for the Linux operating system. It is used with the autoinstall diskette.

syslinux was written by H. Peter Anvin.

ftp://ftp.us.kernel.org/pub/linux/utils/boot/syslinux/

(3) pxelinux −− pxelinux is a network boot loader for the Linux operating system. It is used for autoinstalling over the network.

pxelinux was written by H. Peter Anvin and is part of the syslinux package listed above.

(4) tftp−hpa −− tftp−hpa is a tftp server that has been modified to accept certain options, or "non−options" required for the broken PXE protocol.

tftp−hpa was written by H. Peter Anvin.

http://www.kernel.org/pub/software/network/tftp/

(5) mftp −− mftp is a multicast ftp client that is currently being written by Ian McLeod <ian@valinux.com>. It is based on the multicast libraries being written by Roland Dreier <roland@valinux.com>.

(6) OpenSSH −− Ssh (Secure Shell) a program for logging into a remote machine and for executing commands in a remote machine. It is intended to replace rlogin and rsh, and provide secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel.

OpenSSH is OpenBSD’s rework of the last free version of SSH, bringing it up to date in terms of security and features, as well as removing all patented algorithms to separate libraries (OpenSSL).