White Paper

Data Centre Series
Manifest Based Installs with JET

Table of Contents

Executive Summary

The JumpStart Enterprise Toolkit (JET) was released by Sun Microsystems™ (now Oracle™) to allow the quick and easy set up of the JumpStart automated build system.

While JET allows single systems to be installed quickly and with a good degree of flexibility, setting up – and more importantly maintaining – standard build environments, is a task left up to the installer.

By layering a manifest based installer on top of the JET core, installers can define standard composite components and then quickly reference those composite components from their installation selections. In essence, allowing standards to be developed, deployed and then maintained.

This white paper will discuss how to use an existing JET infrastructure to deploy a manifest based installation methodology, to allow consistent builds across an estate of platforms, without having to spend considerable effort duplicating configurations across each installation template.

Using this approach, time can be spent setting up standard build components and testing them, rather than having to inspect each new system install to make sure the latest version of the build standard has been applied.

Introduction

The Sun JumpStart® technology was introduced as part of Solaris 2, allowing the network deployment of the Solaris operating system, removing the need for physical media (at that time, tape or CD) to be present on each system.

The basic premise is that an installation server is created on the network, and the machine that needs to be installed (the client), requests basic network information, followed by a number of steps where it sets up it's local disks, and then obtains the requisite operating system image to be installed from the server. Once the process has completed, the client system will be ready for use.

The basic JumpStart technology concentrated on the operating system installation, providing a simple 'hook' to allow customisation of the installed system through a 'finish' script. The JumpStart Enterprise Toolkit takes the simple 'finish' script mechanism, and provides a more intuitive and manageable interface to JumpStart; providing front end scripts to simplify the set up of clients and pre-written 'finish' scripts to perform the usual post-installation setup work.

The JumpStart Enterprise Toolkit is itself feature limited – it provides basic building blocks for Solaris operating system installation and configuration, and then relies on 'add-on' modules to enhance it's capabilities. By keeping the toolkit simple, maximum flexibility is still preserved; the downside, is that an installer might have to replicate configuration information in many places.

To scale the JumpStart Enterprise Toolkit, some kind of 'composite' component functionality needs to be introduced. When an installer wants to build a particular host, it would be easier to just have to select 'Java Application Server', or 'Corporate Firewall' than individually select each of the constituent packages, files and scripts that these 'composites' represent.

This functionality can be achieved through a variety of means – this paper discusses the approach of using a manifest based installation technique, where composite components are defined by a plain text manifest, and then each individual host can then just reference one or more manifests.

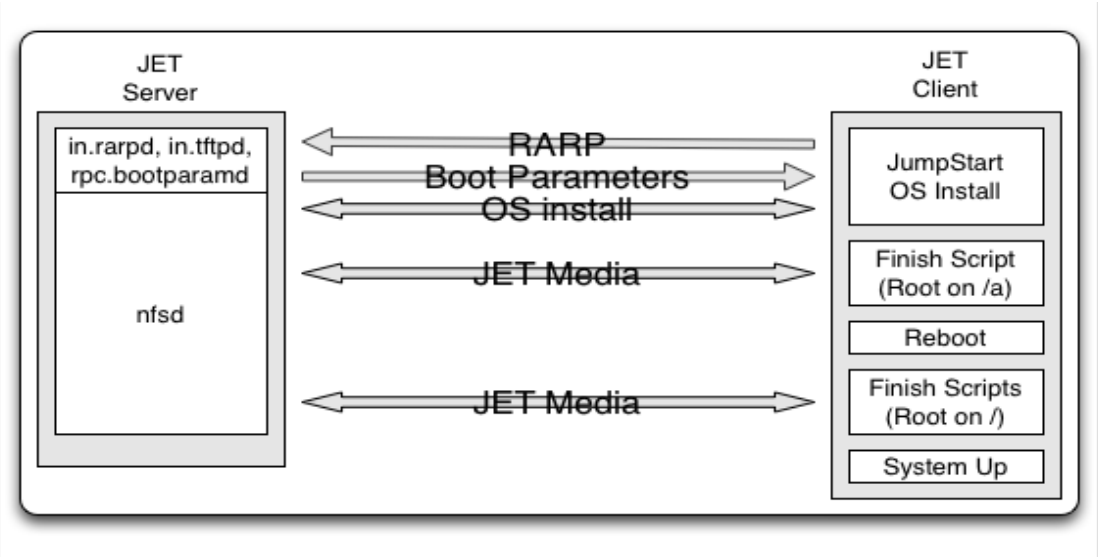
It is assumed that you already have a working JET solution; if not, then go and get that set up first!

Anatomy of an JET Solaris Install

Using the JET software to install Solaris is broken down into a number of simple steps:

- Install JET software
- Copy Solaris media
- Use ‘make_template’ to create an initial configuration for the new system
- Use ‘make_client’ to setup the build environment for the new system
- Start the build on the new system, normally by using ‘boot net – install’

The installation process for a JET deployment is as follows:



For the purposes of this paper, we are interested in the work that happens after the OS has been installed – the ‘finish’ scripts marked in the diagram above – both before and after the first (and subsequent) reboot(s).

The core SUNWjet package doesn’t give you any access to running your own scripts within the build. The design of JET deliberately provided this functionality in a distinct module called ‘custom’; the intention being that it would be then easier for installers to take the ‘custom’ module as a basis for producing their own modules, which could then implement standard builds, or other internally produced software.

Producing customer specific modules still takes time and effort – not least to debug and keep updated. In many cases, JET is just used for OS delivery and simple configuration; all subsequent customisation is still done by hand.

A Manifest Based Approach

In many cases, installers already have an arsenal of scripts they use to configure systems; whether for PCI compliance, firewall hardening or just simple application readiness. Instead of having to go to the effort of updating each JET host template with all the specific scripts, it would be easier to specify that this host is a 'Corporate Firewall' or 'Java Application Server', and then have a definition of what a 'Corporate Firewall' installation actually means.

This is the concept we are calling a 'manifest' –

a list of the wagons forming a freight train

In this instance, a manifest is a simple flat file, describing activities to be performed. Manifests can reference (include) other manifests, allowing compound specifications to be built up, and more importantly, components to be re-used between manifests.

For example, a 'Corporate Firewall' manifest might include the following:

- Install the 'SUNWipfr' and 'SUNWipfu' firewall packages
- Install a basic rule set for IPF and activate
- Verify that root logins are disabled (ssh/telnet)
- Enable routing

If we were then to look at the 'Java Application Server', we might also decide that disabling root logins is part of our security standard, so actually, this is a common component that should be applied to all hosts.

In this case, we could create a new manifest 'Core Security' which contained the 'standard' hardening, disabling direct root logins, switching off services that are never used; and then just get 'Corporate Firewall' to include the 'Core Security' manifest, removing the duplication of work.

Removing this duplication is also important from the point of view of maintenance. If your security standard is updated to include other settings, as an installer, you now just have to go to the 'Core Security' manifest and make those changes. Any hosts that are subsequently built with manifests that include the 'Core Security' manifest (indirectly or directly) will then be built to your updated standard – without updates having to be made to the individual JET templates.

Using Manifests With JET

The first step in adding a manifest capability to an existing JET server is to download and install the module, developed by Maui Systems Ltd.

The module is available from ‘http://downloads.maui.co.uk/release/JetMFI.pkg.bz2’

Once downloaded, install it onto your JET server:

```
root@jet# bunzip JetMFI.pkg.bz2
root@jet# pkgadd -d JetMFI.pkg JetMFI
Processing package instance <JetMFI> from </tmp>

JET Manifest Install(common) 20140125-6700

## Executing checkinstall script.
Using </opt/SUNWjet/Products> as the package base directory.
## Processing package information.
## Processing system information.
## Verifying package dependencies.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.

This package contains scripts which will be executed with super-user
permission during the process of installing this package.

Do you want to continue with the installation of <JetMFI> [y,n,?] y

Installing JET Manifest Install as <JetMFI>

## Installing part 1 of 1.
/opt/SUNWjet/Products/mfi/check_client
/opt/SUNWjet/Products/mfi/copy_media
/opt/SUNWjet/Products/mfi/install
/opt/SUNWjet/Products/mfi/make_client
/opt/SUNWjet/Products/mfi/mfi
/opt/SUNWjet/Products/mfi/mfi.conf
/opt/SUNWjet/Products/mfi/mfi.conf.solarisai
/opt/SUNWjet/Products/mfi/mfi_functions
/opt/SUNWjet/Products/mfi/postinstall
[ verifying class <none> ]
## Executing postinstall script.

Creating sample template......

Adding product configuration information for
        + base_config
        + custom
        + mfi
Client template created in /opt/SUNWjet/Templates
Installation of <JetMFI> was successful.
```

Overview of JET boot levels

On the server, you create a template using the 'make_template' command; by running the 'make_client' command, this sets up the build environment.

Once the client starts to build, the OS installs and then JumpStart calls the generic JET 'finish' script, which is how JET hooks into JumpStart. The finish environment at this point is slightly unique, in that the newly installed OS is actually mounted on /a (referenced by $ROOTDIR), rather than '/'. If you want scripts to run at this level, take this in to consideration. For copying files etc, the MFI module handles this for you.

After the 'finish' phase, JET reboots the system. The system will then reboot off it's newly installed operating system and root (/) is back to normal. At this point JET looks to see if any other work needs to be done. If any module has requested work that needs a reboot, these are numbered '1', '2', '3' etc, After all the items that need to run at reboot 1 have been processed, JET reboots the system and starts with reboot '2' jobs, if there are any.

If there are no more jobs to run at numbered levels, JET then looks for items that are listed at boot level 'n'; this is a special level that signifies that no reboot is required after the jobs are run. Once through the jobs at boot level 'n', the NFS mounts to the JumpStart server are cleared, and finally the boot level 'z' is run.

Boot levels 'n' and 'z' came about for specific reasons – a quick explanation of these might help you get to grips with boot levels.

'n' was introduced as there could be an indeterminate number of reboots requested by modules. For example a system with cluster installed would take 2 reboots, where as one without would only require 1. If you wanted to attach disk mirrors, you didn't really want to do that when the system was about to reboot, so you traditionally attach mirrors at boot 'n'.

'z' was introduced for scenarios where you need to change the primary IP address of the host – it builds on one network range, but then disconnects and will be connected to another network afterwards. As JET uses NFS, the primary IP address can't be changed until after the JumpStart server is disconnected, which is now boot level 'z'.

Create a Simple Manifest

Once installed, the package will create the base directory '/opt/SUNWjet/Manifests'; all manifests defined in client templates will be relative to this directory.

In your favourite editor, create the file '/opt/SUNWjet/Manifests/simple' with the following contents:

```
T  set        base_config_dns_domain example.com
```

```
f  overwrite  simple/issue /etc/issue
1  append     simple/issue2 /etc/issue
1  run        simple/myscript
f  pkgadd     SUNWipfr SUNWipfu
```

The above manifest instructs the MFI module to:

- ‘make_client’:
  - When make_client is run, set the template value base_config_dns_domain to ‘example.com’
- ‘finish phase’:
  - overwrite the contents of /etc/issue with those in ‘simple/issue’
  - add packages SUNWipfr and SUNWipfu
- boot ‘1’:
  - Append the contents of simple/issue2 onto /etc/issue
  - Run the script ‘simple/myscript’

Adding MFI to a JET Template

With a sample manifest built, add the ‘mfi’ settings to your JET template. Either create a new template using:

```
root@jet# make_template server_name base_config mfi
```

or add to an existing template:

```
root@jet# make_template -f -T server_name server_name mfi
```

Once the mfi module is added to the template, edit the template and find the line that starts:

```
mfi_manifests=""
```

and update this to:

```
mfi_manifests="simple"
```

Now you can run ‘make_client’ to set up the build for the server, including the manifest. As we haven’t created the scripts and files to be copied across, the make_client command should generate an error telling us things are missing.

Working with Manifests

Manifests have a simple syntax and potentially lots of whitespace to allow readability and comments. Some examples are in the standard JET template.

Comment lines start with a '#' character.

Boot Level	Description
T	Run when setting up build with 'make_client'
P	Manipulate the Solaris profile (Solaris 10 and lower)
f	Run during 'finish' phase; root on /a ($ROOTDIR)
1-9	Run after reboot 'x'
n	Run after all automatic JET reboots have completed
z	Run after 'n' and after NFS mounts removed

Verb	Description
set	Updated template variable (name value)
overwrite	Replace contents of file (src dest)
append	Append contents of file (src dest)
run	Run script (src)
chown	Change ownership of a file (target)
chgrp	Change group of a file (target)
chmod	Change permissions of a file (target)
mkdir	Create a directory (target)
pkgadd	Add SVR4 package (pkg)
pkgrm	Remove SVR4 package (pkg)
patchadd	Add Sun patch (patch-id)
include	Include another manifest at this point

When using JetMFI in conjunction with JetAI to deliver Solaris 11 installations, you can also use the following:

Verb	Description
ipsrepo	Add an IPS repository (publisher URL) *S11 only
ipsadd	Add a package from an IPS repository (URI) *S11 only
ipsrm	Remove an IPS package (URI) *S11 only

When manipulating Solaris 10 profiles, you can also use the following verbs:

Verb	Description
clear	Clear existing profile contents
update	Change an entry in an existing profile (name value)
add	Add a line to the end of the profile
remove	Remove a line from the profile (name)

All source files listed in manifests are looked for in 2 places. Firstly, the /opt/SUNWjet/Clients/<clientname> directory, and if not found there, they are searched for in the /opt/SUNWjet/Clients directory.

Best Practices

Manifests are located in /opt/SUNWjet/Manifests; how you organise manifests below that is a standard your own organisation can specify. The current best practices on this are:

- /opt/SUNWjet/Manifests/services/<service name>
    - for example file_server, firewall etc.
- /opt/SUNWjet/Manifests/hosts/<hostname>
    - then use the 'include' verb to specify which services this host is to be configured with

As the search path for source files looks for client specific versions of files before more generic ones, locate your scripts and files in a sub-directory of /opt/SUNWjet/Clients, rather than client specific. If a particular client needs to

override the standard, you can populate the client specific /opt/SUNWjet/Clients/<client name> directory with that customised file.

- /opt/SUNWjet/Clients/builds/<service>/scripts
    - Location for service specific scripts
- /opt/SUNWjet/Clients/builds/<service>/files
    - Location for service specific files

Your service manifests would then contain items such as:

```
#
# DNS Corporate Settings
#
T  set        base_config_dns_domain example.com
T  set        base_config_dns_nameservers 10.0.0.1
f  overwrite  builds/corporate-dns/nsswitch.conf /etc/nsswitch.conf
```

And the host manifests:

```
#
# Manifest for 'my new server'
#
include services/corporate-dns
include services/standard-security
include services/java-application-server
```

Next Steps

By defining appropriate services and file structures, you can quickly segregate files and scripts belonging to a particular service into a directory structure that helps annotate where they are used.

JetMFI is compatible with SPARC and X86 installs, and when used with JetAI, it can also be used for deploying to Solaris 11 based systems through JET too.

Solution Benefits

This solution shows how to build and use a manifest based installation technique with JET, which will help segregate files and scripts for particular services, to allow easier deployment, management and maintenance.

As pre-existing scripts and files can be quickly incorporated into the manifest based approach, building standard and consistent operating system deployments is very much within grasp; without the effort of managing lots of individual JET templates.

With the capability of delivering to Solaris 10 and lower, along with Solaris 11 through JetAI, scripts and files can be deployed right across the Solaris estate.

OS Delivery	JumpStart	JET	JET+JetAI	JetAI+JetMFI
Solaris 2.2-2.6	✓	✓	✓	✓
Solaris 7,8,9,10	✓	✓	✓	✓
Solaris 11			✓	✓

JET, JetAI and JetMFI are not architecture specific; you can use either x86 or SPARC as a server platform, and a JET server can build both architectures of client.

Summary

JET is an incredibly powerful toolkit for deploying Solaris and is ideally suited to installing single hosts, or simple base installations of Solaris.

Manually setting complex builds through the supplied ‘custom’ module is tedious and difficult to manage when the number of hosts increases. By specifying composite collections of files and scripts into a ‘manifest’, the JetMFI addition to JET allows standard builds to be defined, managed and very easy to deploy.

As with all tools to deploy standard builds, the ‘standard’ still needs to be defined, built and tested. While the JetMFI module can’t help with the business side of the definition, it can help with the physical definition (in the form of manifests). Building and testing still need to be performed, but as JetMFI allows scripts and files to be applied (through JET) after the initial JumpStart finish phase, scripts can be written an debugged on standard Solaris systems before being added to JetMFI.

References

'Automating Solaris™ Installations, A Custom JumpStart™ Guide', Paul Anthony Kasper and Alan L. McClellan; Prentice Hall

'Oracle JumpStart Enterprise Toolkit',
http://www.oracle.com/technetwork/systems/jet-toolkit/index.html

About the Author

Marty Lee is a Technical Director for Maui Systems Ltd and was one of the original authors of the JumpStart Enterprise Toolkit while working as a subject matter expert for Sun. Still active in the delivery of Solaris installation services to customers, he also continues to be active in the field of high availability solutions for customers and delivering technical architecture consultancy.

Disclaimer

Manifest Based Installs with JET
January 2014
Author: Marty Lee
Maui Systems Ltd
1 Ramsay Cottages
Carlops
Scottish Borders
United Kingdom
EH26 9NF

t: +44 845 869 2661
w: maui-systems.co.uk