## Table of contents

Introduction ......................................................................................................................................... 2
  Overview ........................................................................................................................................ 2
  Intended audience ............................................................................................................................ 2

Background ........................................................................................................................................ 2
  Mapping tendencies ......................................................................................................................... 2
  Framework scope ............................................................................................................................. 4

Example implementation ....................................................................................................................... 8
  Deployment framework solution blocks ............................................................................................... 8
  Network infrastructure services ........................................................................................................ 10
  Provisioning ................................................................................................................................... 11
  Configuration management ............................................................................................................. 13
  Deployment framework implementation ............................................................................................ 15

Summary .......................................................................................................................................... 17

For more information .......................................................................................................................... 18
Introduction

Overview

This document presents a technical survey of Linux deployment issue, best practices and tools. It includes recommendations for assessing the various requirements of a complete deployment framework, both from a strategic and tactical perspective. Through examination of background concepts and best practices, an example implementation of integrating these diverse offerings into a cohesive deployment framework is presented.

The primary focus of this document is on the HP ProLiant platform and the deployment of Red Hat Enterprise Linux. As possible, these boundary conditions are relaxed to include more of the Linux on ProLiant general ecosystem.

Intended audience

The primary audience for this document is junior or intermediate system administrators, although some general sections are applicable to a wider audience. Readers should be familiar with general computing paradigms and comfortable with a focus on Linux system administration and IT practices.

Background

Mapping tendencies

For customers that are new to Linux deployment on industry standard systems, like the HP ProLiant platform, it is often quite a daunting task to have both a strategic and tactical implementation. Several main factors contribute to the success of a deployment framework, including the culture of the support staff and the totality of solution space being targeted. For a given scale of the solution, the remaining consideration is how expedient, repeatable, and controlled the deployment framework must be over the target lifetime of the business services being managed.

Figure 1 shows that the problem space can be divided into at least two general continuums: the operating platform mix and the integration culture of the support staff.
As one criterion, an environment needs to assess both the current and planned mix of operating systems, and decide if the tendency is towards one or multiple platforms. If you determine that a single operating platform is your desired end state, this commonly yields a smaller set of tools and solutions to consider. If you determine that a mix of operating system is your desired end state, you will have many more deployment framework choices and you will have to rely on other criteria to narrow the choices.

As shown in Figure 1, the support staff likely already has some tendencies. If those are engrained in the culture, you should explore other options before crafting a deployment framework, yet favor those that mesh with existing tendencies. For staff whose experience and knowledge base tends towards tool-based methodologies, they tend to rely heavily on local integration of mix and match processes, optimizing the end-to-end approach for the local business goals. Those that rely on more encompassing solution-oriented offerings, strive for commonality across multiple target platforms with offerings that tend towards ease of use and the resulting operational efficiency.

Given the mapping between the operating platform and the integration culture of the support staff, two different styles of deployment offerings are commonly available. If coming from a more proprietary or commercial operating system platform, you usually must use the tools of the particular vendor or Linux distribution only. One of the most liberating facets of Linux, and Free and Open Source Software (FOSS) in general, is the amount and quality of choices available. With a fair amount of overlap, plus considerable depth and breadth available, it becomes a matter of policy or preference when looking at offerings.
The scale and the longevity of the business service being provided are important in the development of a deployment framework. For these criteria, there are often three differing strategic approaches for deployment processes:

- **Divergence** - for environments comprised of quick cycles and short life times of the resulting service. These processes focus heavily on efficient, automated installation processes with little concern for long-term configuration management and maintenance tasks, instead favoring a reinstall of the then latest approved operating environment. For this reason, divergence-based methodologies tend to rely upon native or vendor provided tools with as little integration as possible. Speed and agility from a starting point is the driving concern for selection of deployment framework components.

- **Congruence** - where the business service being offered is comprised of a very small set, but has longevity. Systems all go through a prescribed set of steps, typically starting at a known image-like snapshot. Even errant changes are captured and replayed along with the rework steps toward the current configuration, and all systems are directed to stay in synchronization. Since this process flow is all orchestrated, a combination of tool-based and solution oriented offerings are typically employed. Consistency through detection of new steps, usually via automation, is the driving concern for deployment frameworks utilizing this methodology.

- **Convergence** - for environments that focus on consistency of business services across various operating platform. The starting point, even operating platform, is of little concern and you can treat as a black box. Combinations of tool-based and solution-oriented offerings are typically included, and due to the business logic associated with the service configuration, custom or point tools are likely in use for the integration. The key tenet of convergence is to manage the business service configuration to behave as desired result.

Therefore, it is important to map an environment’s tendencies and assess future changes before attempting to develop a deployment framework. Considering the number of operating platforms, the integration culture, and the scaling tendencies will minimize any change management issues. Recognition of the desired attributes also filters out choices that are not compatible with current and future directions. Any quadrant, and mix of continuums, can have successful deployment framework instances.

**Framework scope**

In order to develop an effective deployment strategy, the next aspects under consideration are what key artifacts of the traditional IT infrastructure are included. It is important for your deployment strategy to determine which are in scope, as it really is the boundary layers between components that you are trying to manage with your deployment framework. Even if starting with a minimal set, you must anticipate how to manage more components over time, so that the deployment framework scales appropriately for your environment.

Figure 2 shows the portions of the overall IT model that you can visualize as a stacked layer representation.
**Hardware** - The foundation is the hardware platform. It can be either a physical system, a virtualization hypervisor, or even a virtual machine. This is the base element that any deployment framework must interact with. In other words, the component that you unpack from the shipping container that must either have remote, perhaps via network or baseboard management controller (BMC), or human hands-on access.

In the context of this document, the assumption for the hardware component layer is a physical HP ProLiant server. For the rack mount and deskside server configurations, like the DL and ML series, you can use traditional keyboard, mouse, and monitor plus the built-in media drive to interact with the server. In addition, you can decide to interact with the Integrated Lights-Out (iLO) form of BMC to completely manage the system. For the BL series blade servers, users most often elect to work at the iLO level or even at the enclosure Onboard Administrator (OA) which can connect to the various blade servers and their respective iLO.

Since the iLO and OA options offer the best options for later automation, one should include these in a deployment framework plan.

**NOTE:** As the iLO and OA interfaces provide physical presence-like access to the servers, you should setup the network interfaces on a secure management network. That allows you to implement secure access controls and restricted access.

Given this access, you must familiarize yourself with all the various ways to interact with these management interfaces. From any web browser, you can login and control the server, including mounting virtual media and controlling the server from a remote console. Additionally, command line interface access is available via SSH and the Virtual Serial Port providing most of the same functions.
You can access the iLO control through any number of means including HP Online Configuration utility, hponcfg, the System Management Architecture for Server Hardware (SMASH) command line protocol, and the Intelligent Platform Management Interface (IPMI).

New for the HP ProLiant Gen8 platform, is the embedded HP Intelligent Provisioning (IP) environment. As an extension to the previous capabilities of the iLO, IP offers tools and utilities to configure a system’s hardware resources, like storage array and BIOS or System ROM settings, to launch an installation of support operating systems, and to act a management or monitoring interface, even when the respective server is powered off.

**Firmware** - As a collection of devices, the hardware component will most certainly have corresponding firmware components. For your hardware, you can directly access and maintain or easily manage the corresponding firmware components from the operating system component. Research the manufacturer’s suggested methodology for managing firmware updates and understand the possible modes for flashing, or upgrading firmware. Due to the hardware specificity, firmware upgrades are often directly tied to either the platform vendor or the individual component vendor.

In the context of this document and given the HP ProLiant assumption, you can flash nearly all of the device firmware on-line from the operating system. Activation of the new firmware is either immediate with a momentary outage or reset, or requires an operating system reboot or power cycle to complete activation. For some firmware components, you need a quiescent device and HP recommends that you perform only an off-line mode of upgrade.

**BEST PRACTICE:** For HP ProLiant Gen8, firmware is available within the Service Pack for ProLiant (SPP) and is accompanied by the HP Smart Update Manager (HPSUM) utility to deliver, deploy, upgrade, and report on the platform’s firmware components. You can accomplish these functions for off-line scenarios from within the IP boot option for HP ProLiant Gen8 systems, or by using the SPP as a mounted application from the operating system for on-line firmware upgrades. In addition, you can choose to use the lower level firmware components in their Smart Component EXEcutable (scexe) packaging to upgrade specific device firmware on-line.

**Operating System** - For all but the simplest of services, it is customary to run a general purpose operating system on the server. Even black-box services that appear to be an appliance-like offering have this characteristic component. Almost all operating systems offer the ability to install on bare-metal, using some type of process developed by the provider.

In the context of this document, it is assumed that the user wants to run a general purpose Linux distribution on the HP ProLiant. For the remainder of this document, the Red Hat Enterprise Linux (RHEL) distribution will be the focus. Many of the same concepts apply to other Linux distributions, but for simplicity, all of the examples provided will have this particular RHEL focus. At this time, Red Hat provides the RHEL distribution via media based images. With the required subscription model, customers are also allowed to access any updates for the respective RHEL version directly from the Red Hat Network (RHN) to address issues and security fixes in the form of errata.

To further simplify the model and representation, it is assumed that the RHEL operating system contains the kernel, device driver, and minimal user space packages which comprise a minimally functional installation. HP and Red Hat work together continuously to for hardware enablement of each successive HP ProLiant generation of servers, ensuring the necessary kernel features are present for the core chipset and that minimally functional device drivers are present for certification. When schedules for releases of the hardware platform and the distribution cannot be reconciled, either
specific kernel errata are required or HP and Red Hat codevelop Driver Update Packages (DUP) that deliver updated device drivers necessary to enable the respective HP ProLiant generation server.

**BEST PRACTICE:** HP publishes minimum operating system versions for each ProLiant generation platform, noting any special requirement, such as specific kernel errata or driver updates. You must review any applicable HP Customer Advisory (CA) reports to understand possible required additional actions. Boot critical device driver updates are also made available as part of the HP ProLiant Gen8 IP offering, presented to the operating system on the Virtual Install Disc (VID). You must explicitly enable this function in order to present the VID to the operating system as a USB device. The format of these driver updates should integrate directly with the native Anaconda based RHEL installer.

**Agentry** - The category of agentry is focused on components, typically supplied by the hardware platform vendor, that interface directly, or through the operating system, to provide specific functionality or manageability with the hardware. These are delivered in custom formats or in operating system compatible packages.

In the context of this document, HP ProLiant Gen8 agentry is contained within the SPP and is accompanied by the HPSUM utility but can also be found in the Software Delivery Repository (SDR) which you can directly install with normal operating system utilities.

**Applications and Services** - For the application and service components, these are often under the explicit control and discretion of the business entity. These are either contained in the operating system distribution itself, delivered in other formats, or even locally developed by the business entity.

**Additional Deployment Considerations** - Providing a more complete representation of the aspects that should be considered for a deployment framework, several other realms must be included in the overall model. Ways to continually manage configurations, monitor the health of the components, and being able to cope with disaster recovery through backup and recovery are key areas to account for. A full treatment of these areas is well beyond the scope of this document, however representative examples will be provided. With all these consideration, the key attributes for the deployment framework to take into account and to provide include the client-side application, if any, plus the model for data transport and the version control of any changes applied. A brief description of each of the consideration follows:

- **Configuration Management** - Similar to the previous discussion of scale and process preferences, the means to manage the configuration of the deployments vary from very simple (divergence) to complex (convergence). In addition, local preferences, tendencies, or policies can dictate how configurations must be managed, whether applied ad-hoc by direct file edits or through a more formal change management process where changes are packaged into operating system compatible formats.

- **Monitoring** - Many different characteristics should be considered in the deployment framework. Often the monitoring is done across the following:
  - availability (basic binary quality of on/up or off/down)
  - performance (continuous, analog measure of given quantity’s value)
  - security (reporting and compliance to policy)
  - integrity (consistency of desired results versus actual results)

- **Backup/Recovery** - To address possible disaster recovery, a core discipline present in the environment is likely the backup and recovery processes. Depending upon the tendencies and scale discussed in previous sections, an environment can decide to have the deployment framework to completely recreate any business service, leaving only actual service data for
the backup and recovery process.

Example implementation

With the background concepts previously covered, you should now have a good basic understanding of what a comprehensive deployment framework must provide. All that remains is a practical implementation, providing coverage of these aspects. Given the focus of this document, deployment of HP ProLiant with RHEL, one approach is to utilize as many solution-based offerings from the vendors as possible. In that vein, you can start from the service component downward to leverage the highest level tools available to provide the deployment framework.

Starting from bare metal installation, provision everything needed to provide and maintain the service in a consistent, repeatable fashion. Given this approach, a good starting point is the Red Hat Network (RHN) itself, since all systems already contain a subscription and have access. With this interface, you can ensure that entitled systems can access updates, additional software, and various other channels of components. However, this only addresses a small portion of a system's lifecycle rather than the larger needs for a deployment framework to be satisfied.

Deployment framework solution blocks

The next logical step is to use the Red Hat Network Satellite server. This brings an RHN instance inside your infrastructure, plus adds both monitoring, configuration management, and provisioning aspects, thus covering many deployment framework requirements. Figure 3 shows a simplified block diagram of this solution.
Based upon these native building blocks already present, it is fairly straightforward to implement a robust, flexible, and scalable deployment framework. Further, each of these can be extended through the inherent modularity and interfaces, to flesh out the remaining aspects.

**NOTE:** At this point, much of the Red Hat Network Satellite server’s software stack is really FOSS, courtesy of Red Hat principles and practices. With that knowledge, it is rather straightforward to assemble the corresponding pieces with some integration steps and end up with a fully functional equivalent. As such, much of the remaining information applies irregardless of the starting point.

In the next few sections, each of the larger blocks of functionality is described, refined, and integrated with other offerings to complete the deployment framework. While the solution for this example deployment framework was derived in a top-down fashion, it is desirable to examine it from a bottom-up approach as these offerings integrate and build upon each other.
Network infrastructure services

Starting with the foundational element, the network infrastructure services, you must attend to these very basic services. Even if the network infrastructure services are already provided and no additional work on your part is required, you should understand them as they offer great impact to the overall deployment strategy. Comprehensive documentation exists to guide you through installation, setup, and configuration of each of these services, so only a brief mention is made of the required functionality and integration aspects needed for each of these services.

The network infrastructure can be broken up into a handful of core services as shown in Figure 4.

![Figure 4: Deployment framework solution – network infrastructure services](image)

The following elaborates on each service, noting particular aspects to address for the deployment framework. Any best practices are noted as some services may also be augmented by inclusion of other tools and by the integration with these services peers.

**Network Time Protocol (NTP)** - Using either a directed or broadcast based implementation for NTP yields a coherent source of time across all configured systems. Given some care, you can easily scale this service.

**BEST PRACTICE:** You can easily couple NTP into the DHCP service as you can provide the parameter for a source along with other information to systems requesting an IP address.

**Dynamic Host Configuration Protocol (DHCP)** - When systems are connected to the network, systems can either obtain an IP address dynamically or be configured statically. With DHCP, newly connected systems can receive connectivity and service configuration information.

**BEST PRACTICE:** You can easily configure DHCP to provide IP reservations, and thus give the appearance of static IP addresses. This recommended approach still retains the benefits of minimal client side configuration.
Domain Name Service (DNS) - In order to associate memorable hostnames with IP addresses, DNS provides a way for humans and systems to perform this forward and reverse translation.

BEST PRACTICE: It is not uncommon for DNS and DHCP to be more closely coupled, allowing a client system to suggest a hostname and allowing DNS and DHCP to provide an IP address and the associated mapping function. This is known as Dynamic DNS (DDNS).

Preboot Execution Environment (PXE) - As a companion service to DHCP, PXE allows systems to obtain a network bootable image. Setup and usage of this service is often overlooked until much later. This service provides a way to automate installations and other tasks with a simple boot cycle, all without the use of physical installation media. It is heavily leveraged for a complete deployment framework. Review the next-server parameter to provide the PXE-DHCP coupling.

BEST PRACTICES:
- As part of HP hardware enablement with Linux distributions like RHEL, sometimes driver updates are required. You can use these driver updates during installation from the media with the linux dd option. However, for newer version of driver updates (DUP), you can also chain load them directly during PXE-based installations, which is highly encouraged for your deployment framework.

- For HP ProLiant Gen8, the Service Pack for ProLiant (SPP) collection product is introduced. Containing a comprehensive, tested set of firmware, drivers, and agents, plus the HP System Update Manager (HPSUM), you can download SPP and set it up as an PXE target itself. In this context, the most valuable portion is the firmware maintenance, since you can integrate the drivers with the native installer (via DUPs), and manage the agents more easily via other means.

NOTE: This approach does require one extra boot cycle to install firmware upgrades, if needed, but this type of automation targets a completely managed infrastructure.

In summary, there are four major network infrastructure services that are fundamental to a deployment framework. You can group them to provide a firm foundation, and with augmentation also address the hardware-firmware layer with the SPP inclusion into PXE, and address the hardware-operating system integration with the DUP inclusion into PXE.

Provisioning

Figure 5 shows that the provisioning portion is provided by a set of tools that works together harmoniously.
Figure 5: Deployment framework solution – provisioning

The Cobbler toolset provides a way to collect installation media into network repositories, offering standardized ways to present these to target systems. Many different Linux distributions are supported.

**BEST PRACTICES:**

- Relying upon both DHCP and PXE from the network infrastructure services, this toolset allows systems to easily perform a network boot and installation process. In this way, you can easily automate the startup logistics of installations.

- Using Cobbler also implies the notion of the repository functionality. Plan what kind of storage paradigm to use, whether direct or network attached, and if the repository can be scaled, both in terms of storage growth over time plus the ability to replicate this to other geographical sites as needed. Before the system installation process itself can become completely unattended, the profile functionality must be present. Utilizing kickstart as an answer file technology, all of the typical interactive install questions and choices can be pre-answered. Many different kickstart profiles may be established, to match the desired services being deployed. With the coupling of Cobbler and kickstart, you can manage them in a mix and match layered offering or simply as templates to reuse. In any case, the key issue is to completely describe the installation to remove user interaction for the sake of consistent, repeatable installations.
With both Cobbler and kickstart in place, the RHEL installer, **anaconda**, can now perform the install completely without intervention. The anaconda toolset thus provides a scripted installation paradigm, to allow hardware abstraction, yet delivering a correct by construction result.

Based upon the described best practices so far, it is reasonable to provision a service platform, in an automated fashion that has a known set of firmware versions, the necessary device drivers integrated, and an operating system profile that can be completed pre-determined.

**Configuration management**

The configuration management portion is provided by the RHN Satellite server, based upon the spacewalk toolset, as shown in Figure 6.

---

**Figure 6**: Deployment framework – configuration management

The key functionality provided by this block is to manage the post-install life cycle of the target systems. By providing operating system updates directly from Red Hat, you can maintain your systems easily with regard to security updates, bug fixes, and other enhancement requests. These offerings are visible within the repository and are typically preconfigured as channels, which the target systems subscribe to. A simple bootstrap process allows the system to access the content that the respective subscription or entitlement allows.
BEST PRACTICES:

- You must develop and adopt a security policy which includes proactive patch management strategies. Once these are in place, the use of the configuration management functionality simply becomes an instrument of implementation.
- Due to the rich capabilities of the previously describe provisioning block, you can make the bootstrap process an integral part of the profile, simply embed the steps into the post section of a kickstart file.

In addition to offering only operating system content, this toolset also allows an easy way to import content into other channels. Using either the web interface or direct command-line access, you can create new channels and content included. The content can either be static, or a channel can actively poll for new content and include it automatically.

BEST PRACTICES:

- As mentioned previously, new for HP ProLiant Gen8, the SPP collection product is being introduced. In addition to the media-based images, the pre-packaged content for Linux distributions is also made available int Software Delivery Repository (SDR). You can access the desired content easily, either directly from the main web site or from a locally mirrored copy, to include as a channel within the configuration management offering.
- Based upon local policy, you might want to only include locally-certified versions of content. For both the errata and for content like the SPP, you can accomplish this by utilizing test channels for evaluation, then later importing snapshots of these test channels into normal parent channels that are automatically being included and accessed.

Some other key attributes of the configuration management solution are the ability to directly manage target systems update schedules, provide system or organizational groupings as a means of access control, and the ability to manage distinct configuration file changes as needed to target systems.

BEST PRACTICE: One of the lesser known aspects of this solution is the well document Application Programming Interface (API) access. Utilization of this facility by experienced administrators can yield impressive automation efficiencies across a wide range of needs.

To summarize, in addition to the benefits and coverage of your deployment framework with the network infrastructure services and provisioning components, you now have the ability to manage the post-installation lifecycle of the operating system, applications, and agentry needed. As a bonus, this integrates the necessary content into a repository that can be accessed during installation processes as well.

Monitoring: Fortunately, the solution-based offering being described also provides monitoring functionality. In conjuction with the available Smart Management Add-On, you can setup many low-impact probes for any specific system or groups of systems. You can also setup thresholds and notification, via email or pager notices, upon reaching or exceeding such levels. Inclusion of these Add-On products also yields more functionality in both the configuration and provisioning functionality blocks.

BEST PRACTICE: You can integrate other HP products, such as Node Manager and Systems Insight Manager with the monitoring functionality. You can configure them both to monitor specific HP ProLiant functionality and you can integrate them using the RHN API.
**Backup / Recovery** - Conspicuous by its absence, you might wonder how to address this discipline given the deployment framework solution provided so far. But some of the core attributes are built-in to the solution. Usage of this implementation yields an infrastructure that can easily redeploy a service hosting platform to a known state. This can be a faster and more prudent approach than trying to capture all the entropy of the core platform over time through traditional backup and restore processes.

**NOTE:** This deployment framework implementation does not address the actual data of the service component. You must carefully plan a service to both backup this content and be able to restore the data in a timely manner.

Deployment framework implementation

Figure 7 shows the completed deployment framework.
In respect to the background material and concept, this implementation definitely focuses on the Linux operating platform but is clearly mid-range with regard to preferences for solution-oriented versus tool-based. While using vendor supplied offerings, the underpinning technology is mostly FOSS toolsets. Depending upon the configuration management paradigms used in the portion, you can cater to either divergence or convergence disciplines. As described, this implementation also works across the entire deployment component space, providing a scalable architecture. Sufficient details and references are provided to guide you through those tasks from implementation to developing a coherent local instance of this deployment framework.
Summary

The first part of this document gives you a relevant background to map the current landscape of possibilities and the tendencies of the organization. It also gives you an overall representation of the aspects needed by a deployment framework. These are provided to perform a self-assessment and survey approach.

The second part of this document presents an example architecture using the background information to provide such an implementation. Any number of other possibilities exists, but the solution outlined provides both a target to emulate and an example of how to utilize many tools to coherently provide such a deployment framework. Many other tools and combinations exist, but these are beyond the scope of this paper.
For more information

For additional information, refer to the resources listed below.

<table>
<thead>
<tr>
<th>Resource description</th>
<th>Web address</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP ProLiant Gen8</td>
<td><a href="http://www.hp.com/go/proliantgen8">www.hp.com/go/proliantgen8</a></td>
</tr>
<tr>
<td>Linux on HP ProLiant servers</td>
<td><a href="http://www.hp.com/go/proliantlinux">http://www.hp.com/go/proliantlinux</a></td>
</tr>
<tr>
<td>iLO</td>
<td><a href="http://www.hp.com/go/ilo">http://www.hp.com/go/ilo</a></td>
</tr>
<tr>
<td>Red Hat</td>
<td><a href="http://www.redhat.com">http://www.redhat.com</a></td>
</tr>
</tbody>
</table>