Hewlett Packard Enterprise

HPE Insight Control Server Provisioning

How to Create an OS Build Plan for Installing Docker EE on Ubuntu 14.04

Technical white paper

Contents

Summary

Insight Control server provisioning (ICsp) provides OS Build Plans, scripts, packages, and configuration files that are used to deploy operating systems, configure hardware, update firmware and perform scripted installations.

ICsp tasks, such as installing a server or updating firmware, are performed using OS Build Plans (OSBP). OS Build Plans are simply a collection of ordered steps and associated parameters that when placed together, in the proper order, can perform just about any action you require. Insight Control server provisioning comes ready to run, with sample build plans and build plan steps that are designed to work right out of the box. These sample build plans are very important, because they demonstrate the steps needed to perform the most common deployment-related operations.

Docker Enterprise Edition (Docker EE) is designed for enterprise development and IT teams who build, ship, and run business critical applications in production at scale. The Insight Control server provisioning appliances do not have an OSBP to perform scripted installation of Docker Enterprise Edition (EE). This document provides instructions and sample scripts to be used in conjunction with the existing OS Build Plans in ICsp to automate the provision of Docker ready servers on HPE ProLiant and Synergy Server.

Enhancing ICsp OS Build Plans to add Docker EE deployment automation has the following benefits:

- Accelerates Docker EE deployment time.
- Minimizes unintended user installation issues.
- Enables HPE to incorporate Docker installation best practices as part of the ICsp deployment plan.

Target audience: This document is intended for IT architects, IT administrators and system integrators. This document assumes that the reader is familiar with Insight Control server provisioning.

Docker Enterprise Edition overview

Docker EE is integrated, certified and supported to provide enterprises with the most secure container platform in the industry to modernize all applications. An application centric platform, Docker EE is designed to accelerate and secure the entire software supply chain, from development to production while running on any infrastructure.

Docker Enterprise Edition provides a container runtime, with integrated and multi-tenant orchestration, security and management in addition to an ecosystem of certified technologies. This gives enterprises an open container platform that ensures a simplified yet rich user experience. The new modular platform makes it easy to install, configure and upgrade Docker on certified infrastructure (operating systems and cloud providers).

Docker EE is a certified container platform for the CentOS Distribution, Red Hat Enterprise Linux (RHEL), Ubuntu, SUSE Linux Enterprise Server (SLES), Oracle Linux and Windows Server 2016, as well as cloud providers AWS and Azure. Docker and its partners provide cooperative support for certified containers and plugins so that customers can confidently use these products in production.

Docker EE is available in three tiers:

- Basic: The Docker platform for certified infrastructure, with support from Docker Inc. and Certified Containers and Plugins from Docker Store.
- Standard: Adds secure multi-tenancy with advanced image, container management, LDAP/AD user integration, secure software supply chain (Docker Datacenter).
- Advanced: Adds Docker Security Scanning and continuous vulnerability monitoring.

Docker EE secures the entire software supply chain from development to production, ensuring all applications are safer. Regardless of whether it is a traditional homegrown application, ISV application or micro services architecture, it will benefit from a modern security model with Docker EE.

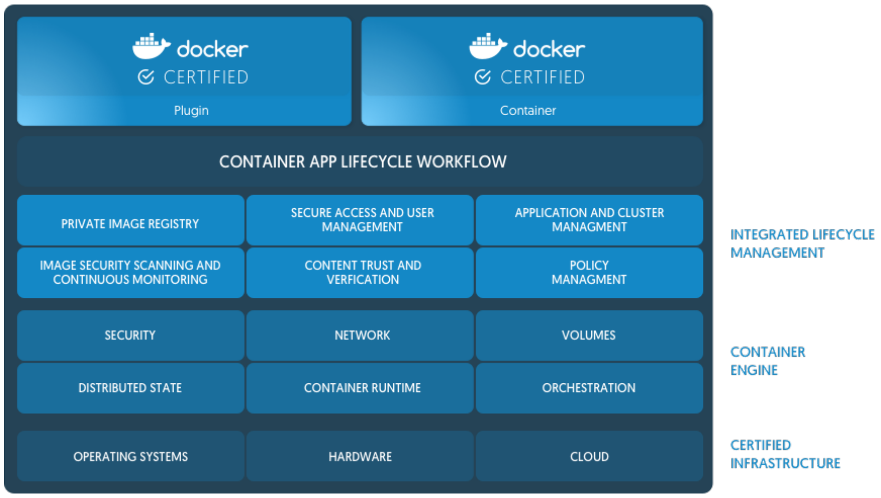


Figure 1. Docker Enterprise Edition

For more information on Docker EE refer to Docker Enterprise Edition.

HPE Insight Control server provisioning overview

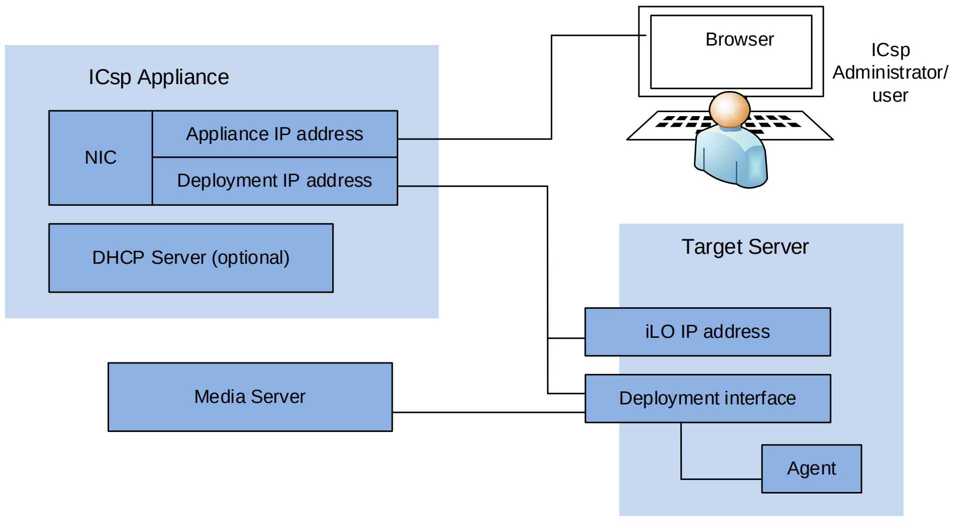


Figure 2. ICsp appliance architecture

The Insight Control (IC) server provisioning appliance is pre-packaged with OS Build Plans (OSBPs) that perform scripted installations. For detailed information on the Insight Control server provisioning appliance setup, see the HPE Insight Control server provisioning Installation Guide at http://h20565.www2.hpe.com/hpsc/doc/public/display?docId=c05305712.

The Media Server is a virtual appliance, separate from Insight Control server provisioning, that holds deployment software. OS Build Plans use the software on the Media Server to provision managed servers. Software (media) on the Media Server can include vendor-supplied OS distribution files, captured images, and firmware and driver updates such as HPE Service Packs for ProLiant (HPE SPP). Having a separate Media Server allows you to easily manage the disk space and contents of the Media Server as opposed to being constrained by the appliance. You can also tailor the network bandwidth requirements of the Media Server to the deployment load.

For instructions on setting up a Media Server, see the “HPE Insight Control server provisioning Installation Guide,” available at http://h20565.www2.hpe.com/hpsc/doc/public/display?docId=c05305712.

For further instructions on managing and administering an ICsp appliance, please refer to the “HPE Insight Control Server Provisioning Administrator Guide” available at http://h20566.www2.hpe.com/hpsc/doc/public/display?docId=c05303955.

For extensive information on ICsp OS Build Plans, please refer to the “HPE Insight Control Build Plans Reference Guide” available at http://h20566.www2.hpe.com/hpsc/doc/public/display?docId=c05305518.

This section describes the challenges this solution is built to address and the benefits of using the solution.

OS Build Plans

An OS Build Plan is a sequence of steps that execute in a specific order to perform a task on a target server. OS Build Plan steps are autonomous operations, such as `run script` or `install package`. OS Build Plans are typically used for provisioning operating systems, but can be used to automate most tasks. OS Build Plans use the software on the Media Server to provision managed servers.

Four types of steps are available: Run Script, Deploy Package, Deploy Configuration File, and Capture Configuration File.

Creating Build Plans

You can use existing OS Build Plans as templates for creating your own. Hewlett Packard Enterprise supplies OS Build Plans with the Insight Control server provisioning product that work out of the box, but they are also designed to be used as templates. The HPE-provided OS Build Plans in ICsp are read-only and may not be edited, but you can save a copy as a starting point to work from.

Custom attributes

Custom attributes substitute specific values into scripts, configuration files, and package paths when an OS Build Plan is run. This is useful for configuring installation processes, including network and server configuration. Custom attributes are typically used for overriding default values.

Deploy Configuration File

Configuration files are text files stored on the appliance that are used for text-based data such as unattended installation files or hardware configuration files. The `Deploy Configuration File` step takes the specified configuration file and writes it to a user-specified location on the target server. These steps are often followed by a `run script` step that makes use of the configuration file. You can use one of the many sample configurations provided by HPE or you can create your own.

Scripts

Scripts in Insight Control server provisioning are used to accomplish provisioning tasks. To run a single script, insert it as a step in an OS Build Plan. Alternatively, you can run multiple scripts using a series of steps within a Build Plan. Insight Control server provisioning supports a variety of script types, for example, bash shell (sh), C shell (csh), KornShell (ksh), Python, Windows batch file (.BAT) and Opsware Global File System (OGFS) scripts.

Supported Configurations

This ICsp OS Build Plan has been tested using ICsp 7.6 on ProLiant DL/BL Servers and Synergy against Ubuntu 14.04. The OSBP should work on any Proliant DL/BL Servers and Synergy Compute Modules that are supported by ICsp.

Location of required ICsp scripts

All the required scripts to create the OS Build Plan are available at https://github.com/HewlettPackard/ICsp-Docker-OSBP, under the **ubuntu** folder.

OS Build Plan for Docker Enterprise Edition installation on Ubuntu 14.04

Prerequisites

- ICsp is installed and configured to deploy Ubuntu 14.04 OS.
- If the target system is behind a proxy, the proxy hostname and port must be provided.

Register Servers and Create Custom Build Plan

1. Register each of the servers that you will be deploying in ICsp by providing the required iLO information:

Log in to the ICsp server and navigate to `Servers` → `Add server`

Populate the fields below:

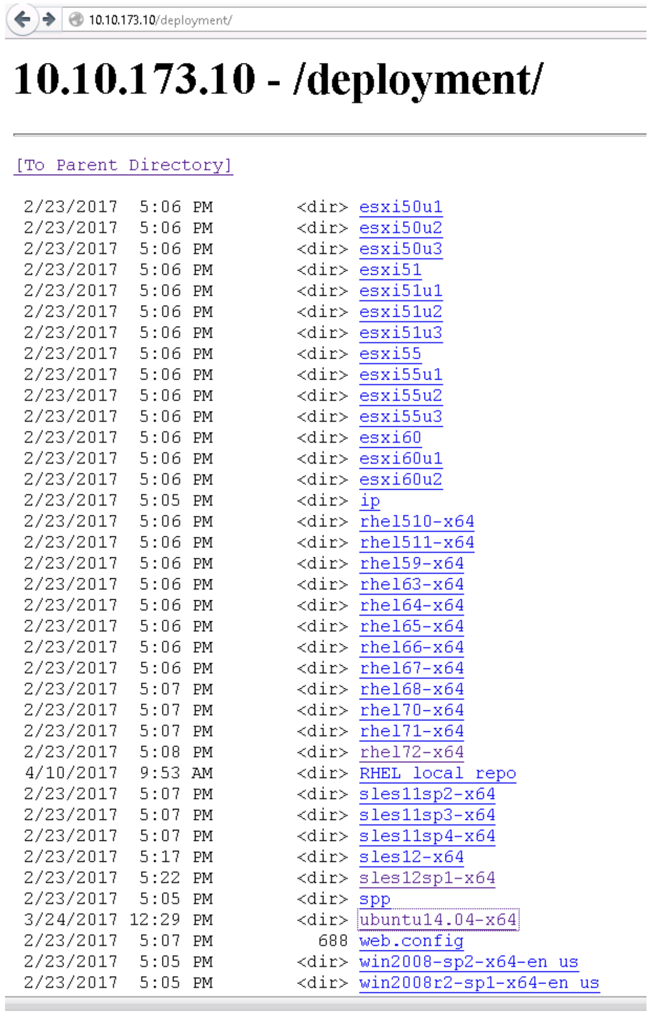
a. iLO IP Address

b. Username

c. Password

Click Add.

2. Log in to the server configured as your HPE ICsp media server.
3. Upload Ubuntu 14.04 to ICsp media server:
   a. Create a directory on the media server share for Ubuntu 14.04. Confirm these directories can be seen in a web browser by using the Media Server URL as shown in Fig. 3

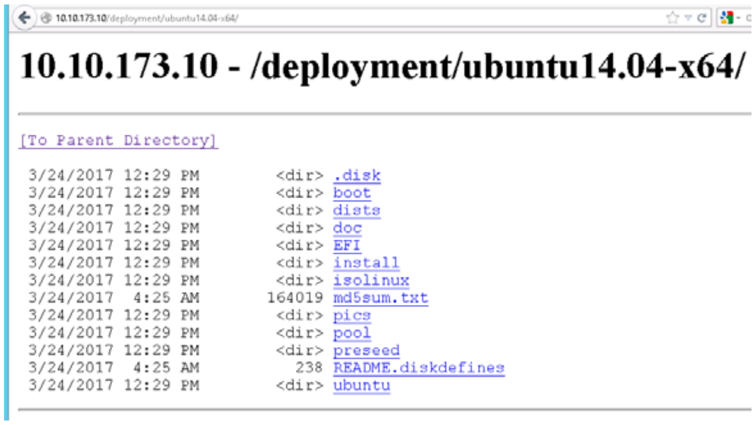


```
10.10.173.10/deployment/

10.10.173.10 - /deployment/

[To Parent Directory]

 2/23/2017  5:06 PM        <dir> esxi50u1
 2/23/2017  5:06 PM        <dir> esxi50u2
 2/23/2017  5:06 PM        <dir> esxi50u3
 2/23/2017  5:06 PM        <dir> esxi51
 2/23/2017  5:06 PM        <dir> esxi51u1
 2/23/2017  5:06 PM        <dir> esxi51u2
 2/23/2017  5:06 PM        <dir> esxi51u3
 2/23/2017  5:06 PM        <dir> esxi55
 2/23/2017  5:06 PM        <dir> esxi55u1
 2/23/2017  5:06 PM        <dir> esxi55u2
 2/23/2017  5:06 PM        <dir> esxi55u3
 2/23/2017  5:06 PM        <dir> esxi60
 2/23/2017  5:06 PM        <dir> esxi60u1
 2/23/2017  5:06 PM        <dir> esxi60u2
 2/23/2017  5:05 PM        <dir> ip
 2/23/2017  5:06 PM        <dir> rhel510-x64
 2/23/2017  5:06 PM        <dir> rhel511-x64
 2/23/2017  5:06 PM        <dir> rhel59-x64
 2/23/2017  5:06 PM        <dir> rhel63-x64
 2/23/2017  5:06 PM        <dir> rhel64-x64
 2/23/2017  5:06 PM        <dir> rhel65-x64
 2/23/2017  5:06 PM        <dir> rhel66-x64
 2/23/2017  5:06 PM        <dir> rhel67-x64
 2/23/2017  5:07 PM        <dir> rhel68-x64
 2/23/2017  5:07 PM        <dir> rhel70-x64
 2/23/2017  5:07 PM        <dir> rhel71-x64
 2/23/2017  5:08 PM        <dir> rhel72-x64
 4/10/2017  9:53 AM        <dir> RHEL local repo
 2/23/2017  5:07 PM        <dir> sles11sp2-x64
 2/23/2017  5:07 PM        <dir> sles11sp3-x64
 2/23/2017  5:07 PM        <dir> sles11sp4-x64
 2/23/2017  5:17 PM        <dir> sles12-x64
 2/23/2017  5:22 PM        <dir> sles12sp1-x64
 2/23/2017  5:05 PM        <dir> spp
 3/24/2017 12:29 PM        <dir> ubuntu14.04-x64
 2/23/2017  5:07 PM          688 web.config
 2/23/2017  5:05 PM        <dir> win2008-sp2-x64-en us
 2/23/2017  5:05 PM        <dir> win2008r2-sp1-x64-en us
```

Figure 3. ICsp Media Server with Ubuntu 14.04 media

Note: for more information about creating and using a Media Server, please check the official ICsp Installation Guide at: http://h20565.www2.hpe.com/hpsc/doc/public/display?docId=c05305712

b. Copy the Ubuntu 14.04 installation media to the Media server ubuntu14.04-x64 directory created in the previous step. Once the copy is complete, confirm that the full contents of the ISO are visible from a web browser at the URL http://<media_server>/<share_name>, as shown in Figure 4.



```
10.10.173.10 - /deployment/ubuntu14.04-x64/

[To Parent Directory]

 3/24/2017 12:29 PM        <dir> .disk
 3/24/2017 12:29 PM        <dir> boot
 3/24/2017 12:29 PM        <dir> dists
 3/24/2017 12:29 PM        <dir> doc
 3/24/2017 12:29 PM        <dir> EFI
 3/24/2017 12:29 PM        <dir> install
 3/24/2017 12:29 PM        <dir> isolinux
 3/24/2017  4:25 AM       164019 md5sum.txt
 3/24/2017 12:29 PM        <dir> pics
 3/24/2017 12:29 PM        <dir> pool
 3/24/2017 12:29 PM        <dir> preseed
 3/24/2017  4:25 AM          238 README.diskdefines
 3/24/2017 12:29 PM        <dir> ubuntu
```

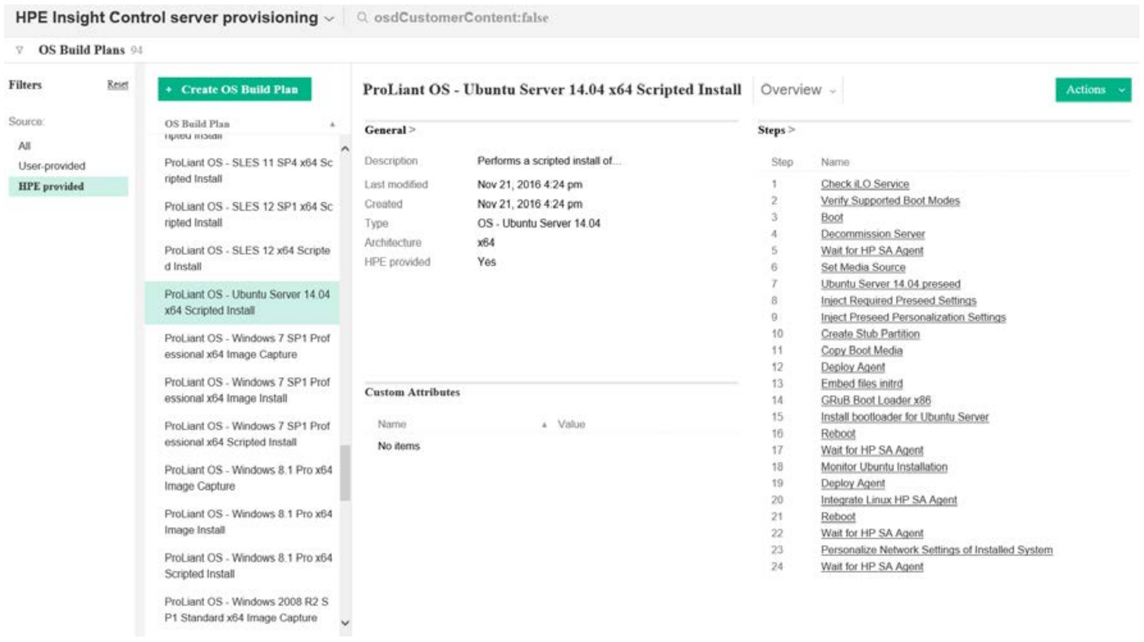
Figure 4. ICsp Media Server

4. Create a Custom Build Plan as follows:

   a. Select the ICsp provided "ProLiant OS - Ubuntu 14.04 x64 Scripted Install" OS Build Plan

   b. Save the OS Build Plan with a new name, for example "ProLiant OS - Ubuntu 14.04 x64 Scripted Install with Docker"

`Actions → Save as`



ripted Install
ProLiant OS - SLES 11 SP4 x64 Scripted Install
ProLiant OS - SLES 12 SP1 x64 Scripted Install
ProLiant OS - SLES 12 x64 Scripted Install
ProLiant OS - Ubuntu Server 14.04 x64 Scripted Install
ProLiant OS - Windows 7 SP1 Professional x64 Image Capture
ProLiant OS - Windows 7 SP1 Professional x64 Image Install
ProLiant OS - Windows 7 SP1 Professional x64 Scripted Install
ProLiant OS - Windows 8.1 Pro x64 Image Capture
ProLiant OS - Windows 8.1 Pro x64 Image Install
ProLiant OS - Windows 8.1 Pro x64 Scripted Install
ProLiant OS - Windows 2008 R2 SP1 Standard x64 Image Capture

ProLiant OS - Ubuntu Server 14.04 x64 Scripted Install

General >

Description	Performs a scripted install of...
Last modified	Nov 21, 2016 4:24 pm
Created	Nov 21, 2016 4:24 pm
Type	OS - Ubuntu Server 14.04
Architecture	x64
HPE provided	Yes

Custom Attributes

Name	Value
No items	

Steps >

Step	Name
1	Check iLO Service
2	Verify Supported Boot Modes
3	Boot
4	Decommission Server
5	Wait for HP SA Agent
6	Set Media Source
7	Ubuntu Server 14.04 preseed
8	Inject Required Preseed Settings
9	Inject Preseed Personalization Settings
10	Create Stub Partition
11	Copy Boot Media
12	Deploy Agent
13	Embed files initrd
14	GRuB Boot Loader x86
15	Install bootloader for Ubuntu Server
16	Reboot
17	Wait for HP SA Agent
18	Monitor Ubuntu Installation
19	Deploy Agent
20	Integrate Linux HP SA Agent
21	Reboot
22	Wait for HP SA Agent
23	Personalize Network Settings of Installed System
24	Wait for HP SA Agent

Figure 5. Ubuntu 14.04 OS Build Plan

5. Add attributes to the OS Build Plan using `Custom Attributes → Edit`

   Custom attributes are user-defined name/value pairs that are used as a form of variable substitution in scripts and other appliance functions. To create custom attributes, select `OS Build Plans → Custom Attributes → Edit → Create Custom Attribute`.

   For Docker installation, the following attributes are created:

a. **docker_repo (Mandatory):** Docker repository for Docker Enterprise Edition. This can be either the URL provided by Docker (external URL) or an internal URL accessible via http. When using an external URL, please note that if your license covers different Linux systems you will need to add the '`ubuntu`' folder. For instance if the URL you've been provided with is '`https://storebits.docker.com/ee/linux/sub-36xxxxx3-dcc3-4ae8-b36d-06xxxxxa9/`', the custom attribute needs to be '`https://storebits.docker.com/ee/linux/sub-36xxxxx3-dcc3-4ae8-b36d-06xxxxxa9/ubuntu`'. If your license only covers Ubuntu then the provided URL should be enough. In case of doubt, please navigate to the URL using a browser to discover the correct URL. Please make sure to specify the protocol (e.g. `https://`) when specifying the URL.

b. **docker_version (Optional):** version of Docker Enterprise Edition to be installed (i.e. `17.03`). If no version is specified then the latest found will be installed.

c. **internal_ubuntu_repo (Optional):** URL pointing to an internal Ubuntu repository. For systems where the internet access is restricted, an internal repository can be used instead. When this custom attribute is specified e.g. http://<repository server>/<path to your repo>, the Docker EE repository URL associated with the subscription will be skipped. Please make sure to specify the protocol (e.g. `http://`) when specifying the URL.

d. **nic_bonds (optional):** the OSBP has the option to create one or more NIC bonds to provide HA networking. This custom attribute defines the list of bonds we intend to create. Format is as follows:

```
<Bond name1>, <MAC address 1>, <MAC address 2>
<Bond name2>, <MAC address 3>, <MAC address 4>
...
```

For instance:

```
Bond0, 00:11:22:33:44:55, 00:11:22:33:44:66
Bond1, 00:11:22:33:44:77, 00:11:22:33:44:88
...
```

This custom attribute can have any number of NIC pairs, but can also be left empty if bonding is not required in the system. The IP address assigned to the bond will be static and will be taken from the first NIC. If the first NIC does not have an assigned IP, then the IP of the second NIC will be used instead.

e. **proxy_hostname (Optional):** Proxy hostname.

f. **proxy_port (Optional):** Proxy port.

g. **no_proxy (Optional):** Comma-separated list of IP addresses or server names where the proxy should not be used for.

6. Create a new Custom Configuration file that will include a modified preseed to replace the default one in the original OSBP. This new preseed file contains the following changes:

– Creation of a `docker` user.

– Fix for routing issues during provisioning

To create a Custom Configuration preseed File:

`Configuration Files → Select Ubuntu 14.04 preseed → Actions → Save As <<Ubuntu 14.04 preseed for Docker>>`

Edit the `<< Ubuntu 14.04 preseed for Docker>>` configuration file:

`Configuration Files → Select <<Ubuntu 14.04 preseed for Docker>> → Actions → Edit`

Download the Ubuntu preseed file named "`preseed_Ubuntu_14.04.cfg`" from GitHub, copy the contents of the file into the newly created Configuration File and select OK to save the configuration file. Replace the ICsp default Ubuntu Server 14.04 preseed configuration file in step 7 of the OS Build Plan with the newly created custom preseed configuration file.

Refer to Appendix A for the script if you have trouble accessing it on GitHub.

7. Create a configuration file for the `sources.list` file and add it at the end of the OS Build Plan (step 25).

   This step is required since the installation process replaces the contents of the file, setting the default repository as the ICsp media server IP, which is not ideal. As a workaround, follow the steps below:

   a. Download the script named "`sources.list`" from GitHub. Refer to Appendix B for the script if you have trouble accessing it on GitHub

   b. Create a new Configuration File in ICsp using

      `Configuration Files` → `Actions` → `Create configuration file`

   c. Copy the contents of the downloaded script and save.

   d. Add the configuration file to the end of your OS Build Plan (step 25), using:

      `OS BuildPlan` → `Actions` → `Edit` → `Add steps`

Select "`Deploy Configuration File`" and specify the newly created "`sources.list`" as the Configuration File. In the Install Path field, specify the path "`/etc/apt/sources.list`".

8. Add a script to install Docker at the very end of the OSBP, which includes:

   – Download and installation of Docker EE packages.

   – Configuration of `aufs` storage driver.

   – Enablement and start-up of the Docker service.

   – Display of Docker info and run of a "hello world" container with the `docker` user to test that Docker is properly installed.

   To do this, follow the steps below:

   a. Download the script named "`install_docker_on_Ubuntu_14.04.sh`" from GitHub to install Docker EE. Refer to Appendix C for the script if you have trouble accessing it on GitHub.

   b. Create a new Script in ICsp using

      `Scripts` → `Actions` → `Create Script`

   c. Copy the contents of the downloaded script and save.

   d. Add the script at the end of the OS Build Plan (step 26)

      `OS BuildPlan` → `Actions` → `Edit` → `Add steps`

9. Add a script to provide bonding between one or more pairs of NICs, defined in one of the custom attributes specified above. If the custom attribute is left blank or if the values are incorrect, the script will finish without errors but won't attempt to create any NIC teams.

   a. Download the script named "`bonding_on_Ubuntu_14.04.sh`" from GitHub to install NIC teaming. Refer to Appendix D for the script if you have trouble accessing it on GitHub.

   b. Choose `Scripts` → `Actions` → `Create Script`

   c. Copy the contents of downloaded script and save.

   d. Add the script at the end of the OS Build Plan (step 27)

The OSBP is now complete and ready to deploy Docker EE to a target server. To run the OSBP on the target server, select the server and click `Actions` → `Run OS Build Plans`. You should see the message stating that the job succeeded once the job OS Build Plan is successful.

You should be able to login via SSH to the brand new system using the `docker` account and the password `ChangeMe123!`. You can then switch to root if required using the same password, but you won't be allowed to connect directly with `root` via SSH. It is highly recommended that you change both passwords as soon as you log in for the first time.

Note: the `docker` user is not part of the `sudoers` by default, so you won't be able to run privileged commands or to switch to `root` by using the `sudo` command. You should instead switch to `root` by using the `su` command (with either "`su -`" or "`su - root`") and then entering the `root` password.

Appendix A – Ubuntu 14.04 preseed file for Docker

```
# Copyright 2017 Hewlett Packard Enterprise Development LP Licensed under the
# Apache License, Version 2.0 (the "License"); you may not use this file except
# in compliance with the License. You may obtain a copy of the License at
# http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law
# or agreed to in writing, software distributed under the License is distributed
# on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
# express or implied. See the License for the specific language governing
# permissions and limitations under the License.

### Hardware
# If non-free firmware is needed for the network or other hardware, you can
# configure the installer to always try to load it, without prompting. Or
# change to false to disable asking.
d-i hw-detect/load_firmware boolean false

### Mirror settings
# Injected by "Inject Required Settings" with media server values
d-i mirror/country string manual
d-i mirror/protocol string
d-i mirror/http/hostname string
d-i mirror/http/directory string
d-i mirror/http/proxy string
d-i mirror/http/mirror select

### Live Installer source
#Gives the location of the live-installer on the network
d-i live-installer/net-image string

### Apt setup
# Uncomment this if you don't want to use a network mirror.
d-i apt-setup/use_mirror boolean false

# You can choose to install restricted and universe software, or to install
# software from the backports repository.
d-i apt-setup/restricted boolean false
d-i apt-setup/universe boolean false
d-i apt-setup/backports boolean false
# Select which update services to use; define the mirrors to be used.
# Values shown below are the normal defaults.
d-i apt-setup/services-select multiselect
d-i apt-setup/security_host string

### Localization
# Preseeding only locale sets language, country and locale.
d-i debian-installer/locale string @system_locale:en_US@

# Keyboard selection.
# Disable automatic (interactive) keymap detection.
d-i console-setup/ask_detect boolean false
d-i keyboard-configuration/layoutcode string us

### Clock and time zone setup
# Controls whether or not the hardware clock is set to UTC.
d-i clock-setup/utc boolean true

# You may set this to any valid setting for $TZ; see the contents of
# /usr/share/zoneinfo/ for valid values.
```

```
d-i time/zone string GMT-0

# Controls whether to use NTP to set the clock during the install
d-i clock-setup/ntp boolean false

### Network configuration
# netcfg will choose an interface that has link if possible. This makes it
# skip displaying a list if there is more than one interface.
# Injected by "Inject Required Settings" with value auto
d-i netcfg/choose_interface select

# Any hostname and domain names assigned from dhcp take precedence over
# values set here. However, setting the values still prevents the questions
# from being shown, even if values come from dhcp.
d-i netcfg/get_hostname string unassigned-hostname
d-i netcfg/get_domain string unassigned-domain

# Adding this line to fix route issues
d-i netcfg/no_default_route boolean true

### Partitioning
# Alternatively, you may specify a disk to partition. If the system has only
# one disk the installer will default to using that, but otherwise the device
# name must be given in traditional, non-devfs format (so e.g. /dev/hda or
# /dev/sda, and not e.g. /dev/discs/disc0/disc).
# For example, to use the first SCSI/SATA hard disk:
# Injected by "Inject Required Settings" with the path to bootdisk
d-i partman-auto/disk string
# In addition, you'll need to specify the method to use.
# The presently available methods are:
# - regular: use the usual partition types for your architecture
# - lvm:     use LVM to partition the disk
# - crypto:  use LVM within an encrypted partition
d-i partman-auto/method string regular

# You can choose one of the three predefined partitioning recipes:
# - atomic: all files in one partition
# - home:   separate /home partition
# - multi:  separate /home, /usr, /var, and /tmp partitions
d-i partman-auto/choose_recipe select atomic

# This makes partman automatically partition without confirmation, provided
# that you told it what to do using one of the methods above.
d-i partman-partitioning/confirm_write_new_label boolean true
d-i partman/choose_partition select finish
d-i partman/confirm boolean true
d-i partman/confirm_nooverwrite boolean true

### Package selection

# By default the installer requires that repositories be authenticated
# using a known gpg key. This setting can be used to disable that
# authentication. Warning: Insecure, not recommended.
d-i debian-installer/allow_unauthenticated string true
# The above is used to allow unauthenticated pakcages while using
# IIS web server, Its not madatory whilst using linux web server.

tasksel tasksel/first multiselect  ubuntu-server
tasksel tasksel/first select OpenSSH server
```

```
# Individual additional packages to install
d-i pkgsel/include string cifs-utils

# Policy for applying updates. May be "none" (no automatic updates),
# "unattended-upgrades" (install security updates automatically), or
# "landscape" (manage system with Landscape).

# Automatically download and install stable updates?
unattended-upgrades unattended-upgrades/enable_auto_updates boolean false

# Install language support packages.
 d-i pkgsel/install-language-support boolean true
 d-i pkgsel/ignore-incomplete-language-support boolean true

d-i pkgsel/update-policy select none

### Boot loader installation
# This is fairly safe to set, it makes grub install automatically to the MBR
# if no other operating system is detected on the machine.
d-i grub-installer/only_debian boolean true

# This one makes grub-installer install to the MBR if it also finds some other
# OS, which is less safe as it might not be able to boot that other OS.
d-i grub-installer/with_other_os boolean true

### Finishing up the installation
# The kernel image (meta) package to be installed; "none" can be used if no
# kernel is to be installed.
d-i base-installer/kernel/image string linux-image-3.13.0-32
# Avoid that last message about the install being complete.
d-i finish-install/reboot_in_progress note

### Account setup
# Skip creation of a root account (normal user account will be able to
# use sudo). The default is false; preseed this to true if you want to set
# a root password.
d-i passwd/root-login boolean true
# Alternatively, to skip creation of a normal user account.
d-i passwd/make-user boolean true

# Root password, either in clear text
# or encrypted using an MD5 hash.
d-i passwd/root-password-crypted password @encrypted_root_password:$1$GLfVx4iq$bYbwJ6oLYGJRph6HVVF7A0@

# Create docker user
d-i passwd/user-fullname string Docker
d-i passwd/username string docker
d-i passwd/user-password-crypted password @encrypted_root_password:$1$GLfVx4iq$bYbwJ6oLYGJRph6HVVF7A0@
d-i user-setup/allow-password-weak boolean true
```

Appendix B – Ubuntu default sources.list file

```
#deb cdrom:[Ubuntu-Server 14.04 LTS _Trusty Tahr_ - Release amd64 (20140416.2)]/ trusty main restricted

# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.
deb http://us.archive.ubuntu.com/ubuntu/ trusty main restricted
deb-src http://us.archive.ubuntu.com/ubuntu/ trusty main restricted

## Major bug fix updates produced after the final release of the
## distribution.
deb http://us.archive.ubuntu.com/ubuntu/ trusty-updates main restricted
deb-src http://us.archive.ubuntu.com/ubuntu/ trusty-updates main restricted

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team. Also, please note that software in universe WILL NOT receive any
## review or updates from the Ubuntu security team.
deb http://us.archive.ubuntu.com/ubuntu/ trusty universe
deb-src http://us.archive.ubuntu.com/ubuntu/ trusty universe
deb http://us.archive.ubuntu.com/ubuntu/ trusty-updates universe
deb-src http://us.archive.ubuntu.com/ubuntu/ trusty-updates universe

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team, and may not be under a free licence. Please satisfy yourself as to
## your rights to use the software. Also, please note that software in
## multiverse WILL NOT receive any review or updates from the Ubuntu
## security team.
deb http://us.archive.ubuntu.com/ubuntu/ trusty multiverse
deb-src http://us.archive.ubuntu.com/ubuntu/ trusty multiverse
deb http://us.archive.ubuntu.com/ubuntu/ trusty-updates multiverse
deb-src http://us.archive.ubuntu.com/ubuntu/ trusty-updates multiverse

## N.B. software from this repository may not have been tested as
## extensively as that contained in the main release, although it includes
## newer versions of some applications which may provide useful features.
## Also, please note that software in backports WILL NOT receive any review
## or updates from the Ubuntu security team.
deb http://us.archive.ubuntu.com/ubuntu/ trusty-backports main restricted universe multiverse
deb-src http://us.archive.ubuntu.com/ubuntu/ trusty-backports main restricted universe multiverse

deb http://security.ubuntu.com/ubuntu trusty-security main restricted
deb-src http://security.ubuntu.com/ubuntu trusty-security main restricted
deb http://security.ubuntu.com/ubuntu trusty-security universe
deb-src http://security.ubuntu.com/ubuntu trusty-security universe
deb http://security.ubuntu.com/ubuntu trusty-security multiverse
deb-src http://security.ubuntu.com/ubuntu trusty-security multiverse

## Uncomment the following two lines to add software from Canonical's
## 'partner' repository.
## This software is not part of Ubuntu, but is offered by Canonical and the
## respective vendors as a service to Ubuntu users.
# deb http://archive.canonical.com/ubuntu trusty partner
# deb-src http://archive.canonical.com/ubuntu trusty partner

## Uncomment the following two lines to add software from Ubuntu's
## 'extras' repository.
## This software is not part of Ubuntu, but is offered by third-party
## developers who want to ship their latest software.
# deb http://extras.ubuntu.com/ubuntu trusty main
# deb-src http://extras.ubuntu.com/ubuntu trusty main`
```

Appendix C – Docker Enterprise Edition installation script for Ubuntu 14.04

```
#!/bin/bash

# Copyright 2017 Hewlett Packard Enterprise Development LP Licensed under the
# Apache License, Version 2.0 (the "License"); you may not use this file except
# in compliance with the License. You may obtain a copy of the License at
# http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law
# or agreed to in writing, software distributed under the License is distributed
# on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
# express or implied. See the License for the specific language governing
# permissions and limitations under the License.

## Logging functions

# 0 - no output, 1 - error|warn messages, 2 - info|error|warn messages, 3 - debug|info|error|warn messages
LOG_FILE_LEVEL=${LOG_FILE_LEVEL:-3}
LOG_STDIO_LEVEL=${LOG_STDIO_LEVEL:-2}
LOG_FILE_DRIVER=${LOG_FILE_DRIVER:-1}
LOG_STDIO_DRIVER=${LOG_STDIO_DRIVER:-1}

SCRIPT_LOG=$HOME/bootstrap.log
touch $SCRIPT_LOG

function EMIT_FILE_DRIVER() {
    if [ 1 -eq $LOG_FILE_DRIVER ]; then
        echo -e $2 >> $SCRIPT_LOG
    fi
}

function EMIT_STDIO_DRIVER() {
    if [ 1 -eq $LOG_STDIO_DRIVER ]; then
        echo -e $2
    fi
}

function EMIT_LOG(){
    # 0 - no output, 1 - error|warn messages, 2 - info|error|warn messages, 3 - debug|info|error|warn messages
    case $1 in
        ERROR|WARN) [ $LOG_STDIO_LEVEL -gt 0 ] && EMIT_STDIO_DRIVER "$@" ;;
        INFO)  [ $LOG_STDIO_LEVEL -gt 1 ] && EMIT_STDIO_DRIVER "$@" ;;
        DEBUG) [ $LOG_STDIO_LEVEL -gt 2 ] && EMIT_STDIO_DRIVER "$@" ;;
    esac
    case $1 in
        ERROR|WARN) [ $LOG_FILE_LEVEL -gt 0 ] && EMIT_FILE_DRIVER "$@" ;;
        INFO)  [ $LOG_FILE_LEVEL -gt 1 ] && EMIT_FILE_DRIVER "$@" ;;
        DEBUG) [ $LOG_FILE_LEVEL -gt 2 ] && EMIT_FILE_DRIVER "$@" ;;
    esac
    r=$? #mute the error level for loglevel checks
}

function SCRIPTENTRY(){
    local ln="${BASH_LINENO[0]}"
    timeAndDate=`date`
    script_name=`basename "$0"`
    script_name="${script_name%.*}"
    EMIT_LOG DEBUG "[$timeAndDate] [DEBUG] [$ln] > $script_name $FUNCNAME"
}
```

```
function SCRIPTEXIT(){
    local ln="${BASH_LINENO[0]}"
    script_name=`basename "$0"`
    script_name="${script_name%.*}"
    EMIT_LOG DEBUG "[$timeAndDate] [DEBUG] [$ln] < $script_name $FUNCNAME"
}

function ENTRY(){
    local cfn="${FUNCNAME[1]}"
    local ln="${BASH_LINENO[0]}"
    timeAndDate=`date`
    EMIT_LOG DEBUG "[$timeAndDate] [DEBUG] [$ln] > $cfn $FUNCNAME"
}

function EXIT(){
    local cfn="${FUNCNAME[1]}"
    local ln="${BASH_LINENO[0]}"
    timeAndDate=`date`
    EMIT_LOG DEBUG "[$timeAndDate] [DEBUG] [$ln] < $cfn $FUNCNAME"
}


function INFO(){
    local function_name="${FUNCNAME[1]}"
    local msg="$1"
    local ln="${BASH_LINENO[0]}"
    timeAndDate=`date`
    EMIT_LOG INFO "[$timeAndDate] [INFO] [$ln] $msg"
}


function DEBUG(){
    local function_name="${FUNCNAME[1]}"
    local msg="$1"
    local ln="${BASH_LINENO[0]}"
    timeAndDate=`date`
    EMIT_LOG DEBUG "[$timeAndDate] [DEBUG] [$ln] $msg"
}

function ERROR(){
    local function_name="${FUNCNAME[1]}"
    local msg="$1"
    local ln="${BASH_LINENO[0]}"
    timeAndDate=`date`
    EMIT_LOG ERROR "[$timeAndDate] [ERROR] [$ln] $msg"
}

function WARN(){
    local function_name="${FUNCNAME[1]}"
    local msg="$1"
    local ln="${BASH_LINENO[0]}"
    timeAndDate=`date`
    EMIT_LOG WARN "[$timeAndDate] [WARN] [$ln] $msg"
}

#
#
## Variables
docker_config_file='/etc/default/docker'
docker_user=docker
```

```
docker_repo="@docker_repo@"
docker_version="@docker_version@"
internal_ubuntu_repo="@internal_ubuntu_repo@"

#
#
## Proxy settings
export proxy_hostname=@proxy_hostname@
export proxy_port=@proxy_port@
export http_proxy=http://@proxy_hostname@:@proxy_port@
export https_proxy=https://@proxy_hostname@:@proxy_port@
export no_proxy=@no_proxy@

#
#
## Script functions
function install_docker() {
    INFO 'Updating the system...'
    # Cleans up apt-get output and avoid some harmless errors
    export DEBIAN_FRONTEND=noninteractive
    apt-get -o Dpkg::Options::="--force-confold" -qq -y update

    INFO 'Installing required packages and setting up the repository...'
    apt-get -o Dpkg::Options::="--force-confold" -qq -y install apt-transport-https curl software-properties-
common > /dev/null
    [ $? -ne 0 ] && ERROR 'There was a problem installing the required packages!' && exit 1
    curl -fsSL ${docker_repo}/gpg | sudo apt-key add -
    [ $? -ne 0 ] && ERROR 'There was a problem retrieving the GPG key!' && exit 1
    if [ ${#docker_version} -gt 0 ]; then
        add-apt-repository "deb [arch=amd64] ${docker_repo} $(lsb_release -cs) stable-${docker_version}"
    else
        add-apt-repository "deb [arch=amd64] ${docker_repo} $(lsb_release -cs) stable"
    fi
    [ $? -ne 0 ] && ERROR 'There was a problem setting the Docker repository!' && exit 1

    INFO 'Installing Docker...'
    apt-get -o Dpkg::Options::="--force-confold" -qq -y update && \
    apt-get -o Dpkg::Options::="--force-confold" -qq -y install docker-ee > /dev/null
    [ $? -ne 0 ] && ERROR 'There was a problem installing Docker EE!' && exit 1

    ## Adding the proxy configuration to the docker daemon env vars
    if [ ! -z "${http_proxy}" ] || [ ! -z "${https_proxy}" ]; then
        INFO 'Adding proxy settings to daemon configuration...'
        [ ! -z "${http_proxy}" ]  && echo -e "export http_proxy=${http_proxy}" >> ${docker_config_file}
        [ ! -z "${https_proxy}" ] && echo -e "export https_proxy=${https_proxy}" >> ${docker_config_file}
        [ ! -z "${no_proxy}" ]    && echo -e "export no_proxy=${no_proxy}" >> ${docker_config_file}
    fi
}

function config_storage() {
    docker_conf="/etc/init/docker.conf"
    tmp_conf="/tmp/docker.conf"
    if [ $(grep aufs /proc/filesystems | wc -l) -eq 0 ]; then
        INFO 'Installing aufs packages...'
        apt-get -o Dpkg::Options::="--force-confold" -qq -y install linux-image-extra-$(uname -r) linux-image-
extra-virtual > /dev/null
    fi
    # Configure Docker daemon
    INFO 'Configuring the Docker daemon'
    awk 'NR==1,/DOCKER_OPTS=/{sub(/DOCKER_OPTS=/, "DOCKER_OPTS=\"--storage-driver=aufs\"")} 1' ${docker_conf} >
${tmp_conf}
    cp ${tmp_conf} ${docker_conf}
    [ $(grep 'DOCKER_OPTS="--storage-driver=aufs"' /etc/init/docker.conf | wc -l) -ne 1 ] && ERROR 'There was a
problem configuring the aufs storage!' && exit 1

}

function enable_and_start() {
    INFO 'Enabling and restarting service...'
    service docker restart
    [ $? -ne 0 ] && ERROR 'There was a problem enabling or starting the docker service!' && exit 1
    # Display Docker info
    INFO 'Checking Docker info...'
    docker info
    [ $(docker info | grep 'Storage Driver: aufs' | wc -l) -eq 1 ] && INFO 'Docker info appears to be as
expected. Storage configured correctly!'
    /usr/sbin/usermod -aG docker docker
    docker run hello-world
}

function configure_internal_ubuntu_repo() {
    INFO "Creating internal repo using provided URL: ${internal_ubuntu_repo}"
    mv /etc/apt/sources.list /etc/apt/sources.list.old
    echo "deb ${internal_ubuntu_repo} trusty main" > /etc/apt/sources.list
}

#
#
## Main
if [ ${#internal_ubuntu_repo} -gt 0 ]; then
    configure_internal_ubuntu_repo
fi
install_docker
config_storage
enable_and_start
exit 0
```

Appendix D – Bonding script for Ubuntu 14.04

```
#!/bin/bash

# Copyright 2017 Hewlett Packard Enterprise Development LP Licensed under the
# Apache License, Version 2.0 (the "License"); you may not use this file except
# in compliance with the License. You may obtain a copy of the License at
# http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law
# or agreed to in writing, software distributed under the License is distributed
# on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
# express or implied. See the License for the specific language governing
# permissions and limitations under the License.

function create_bond() {
    INTERFACES="/etc/network/interfaces"
    TEMP_INTERFACES="/tmp/temp_interfaces"

    # Save original
    cp ${INTERFACES} ${INTERFACES}.orig

    # Get interface names
    int_name1=$(echo `ip link | grep -i -B 1 "${mac1}" | head -1 | cut -d':' -f2`)
    int_name2=$(echo `ip link | grep -i -B 1 "${mac2}" | head -1 | cut -d':' -f2`)

    # Find out how these interfaces boot (none, static, dhcp)
    bootproto1=$(grep "$int_name1" ${INTERFACES} | grep inet | cut -d' ' -f4)
    bootproto2=$(grep "$int_name2" ${INTERFACES} | grep inet | cut -d' ' -f4)

    # Collect and check the configured IP addressed
    ip1=$(echo `ip a | grep -i -A 2 ${int_name1} | grep "inet " | awk -F' ' '{ print $2 }' | cut -d'/' -f1`)
    ip2=$(echo `ip a | grep -i -A 2 ${int_name2} | grep "inet " | awk -F' ' '{ print $2 }' | cut -d'/' -f1`)
    [ ${#ip1} -eq 0 ] && [ ${#ip2} -eq 0 ] && echo "No IP was configured in any of the provided NICs. Exiting."
&& exit 0

    ## Create new interfaces file

    # The following code will start building a new temp interfaces file without the 2 NICs that are going to be
bonded
    [ -f ${TEMP_INTERFACES} ] && rm ${TEMP_INTERFACES}
    found=0
    while IFS='' read -r line || [[ -n "${line}" ]]; do
        if [ `echo ${line} | grep -E "(iface ${int_name1}|iface ${int_name2})" | wc -l` -eq 1 ]; then
            found=1
        elif [[ $found -eq 1 && `echo $line | grep iface | wc -l` -eq 0 ]]; then
            found=1
            [ `echo ${line} | grep -i "netmask" | wc -l` -eq 1 ] && netmask=`echo ${line} | cut -d' ' -f2`
            [ `echo ${line} | grep -i "gateway" | wc -l` -eq 1 ] && gateway=`echo ${line} | cut -d' ' -f2`
            [ `echo ${line} | grep -i "gw "     | wc -l` -eq 1 ] && gateway=`echo ${line} | awk -F'gw' '{ print
$2 }' | awk '{ print $1}'`
        else
            found=0
            echo "${line}" >> ${TEMP_INTERFACES}
        fi
    done < ${INTERFACES}

    # Adding now the two NICs and the bond information
    echo "" >> ${TEMP_INTERFACES}
    echo "auto ${int_name1}" >> ${TEMP_INTERFACES}
    echo "iface ${int_name1} inet manual" >> ${TEMP_INTERFACES}
    echo "bond-master ${bond_name}" >> ${TEMP_INTERFACES}
    echo "bond-primary ${int_name1}" >> ${TEMP_INTERFACES}
    echo "" >> ${TEMP_INTERFACES}
    echo "auto ${int_name2}" >> ${TEMP_INTERFACES}
    echo "iface ${int_name2} inet manual" >> ${TEMP_INTERFACES}
    echo "bond-master ${bond_name}" >> ${TEMP_INTERFACES}
    echo "" >> ${TEMP_INTERFACES}
    echo "auto ${bond_name}" >> ${TEMP_INTERFACES}
    echo "iface ${bond_name} inet static" >> ${TEMP_INTERFACES}
    if [[ ! -z "${ip1}" ]]; then
        echo "address ${ip1}" >> ${TEMP_INTERFACES}
        echo "netmask ${netmask}" >> ${TEMP_INTERFACES}
        [ ${#gateway} -gt 0 ] && echo "gateway ${gateway}" >> ${TEMP_INTERFACES}
    elif [[ ! -z "${ip2}" ]]; then
        echo "address ${ip2}" >> ${TEMP_INTERFACES}
        echo "netmask ${netmask}" >> ${TEMP_INTERFACES}
        [ ${#gateway} -gt 0 ] && echo "gateway ${gateway}" >> ${TEMP_INTERFACES}
    fi
    echo "bond-mode active-backup" >> ${TEMP_INTERFACES}
    echo "bond-miimon 100" >> ${TEMP_INTERFACES}
    echo "bond-slaves none" >> ${TEMP_INTERFACES}

    # Update the /etc/network/interfaces file
    cp ${TEMP_INTERFACES} ${INTERFACES}

    # "Restart" the network
    # The usual service networking restart would not work so we have to be creative here
    # (https://bugs.launchpad.net/ubuntu/+source/ifupdown/+bug/1301015)
    ip addr flush ${int_name1}
    ip addr flush ${int_name2}
    ifdown --exclude=lo -a && ifup --exclude=lo -a
}

function validate_macs() {
    if [ `ip link | grep -i "$1" | wc -l` -ne 1 ]; then
        echo "ERROR : MAC address $1 not found, skipping this bond"
        skip_bond=1
    fi
}

function initialize() {
    skip_bond=0
}

#
## Main

# Cleans up apt-get output and avoid some harmless errors
export DEBIAN_FRONTEND=noninteractive

# Read the interfaces into an array
readarray ifaces_list < <(echo "@nic_bonds@")

# Set the proxy and install ifenslave package
export proxy_hostname=@proxy_hostname@
export proxy_port=@proxy_port@
export http_proxy=http://@proxy_hostname@:@proxy_port@
export https_proxy=https://@proxy_hostname@:@proxy_port@
export no_proxy=@no_proxy@
```

```
apt-get -o Dpkg::Options::="--force-confold" -qq -y update && apt-get install -o Dpkg::Options::="--force-
confold" -qq -y ifenslave > /dev/null

# Permanently load bonding module
[ $(grep bonding /etc/modules | wc -l) -eq 0 ] && echo "bonding" >> /etc/modules

# Go through the list of bonds to be created
for t in "${ifaces_list[@]}"; do
    initialize
    bond_name=$(echo ${t} | cut -d',' -f1)
    mac1=$(echo ${t} | cut -d',' -f2)
    mac2=$(echo ${t} | cut -d',' -f3)
    validate_macs ${mac1} ${mac2}
    [ ${skip_bond} -eq 0 ] && create_bond
done

exit 0
```

Appendix E – Troubleshooting

If the ICsp Job fails, please check the following for more information:

- Check the job log by clicking the Logs link in the Jobs details and check for the step that failed.
- Check the console on the target server for errors during the installation.
- Make sure that the DHCP server on the deployment network is reachable. Refer to HPE Insight Control Server Provisioning Installation Guide.
- Make sure you are using the amended preseed file provided either on GitHub or Appendix A

If the Docker Installation fails:

- If you are behind a proxy, make sure that the proxy host and port are set and that the values are correct.
- Check the Docker URL in the custom attributes. You can test it in a browser to make sure it is correct and reachable.

If the NICs bond is not created:

- Check that the MAC addresses provided in the custom attributes match the ones in the server.
- Please make sure that at least one of your interfaces gets assigned an IP address during installation.
- Please note that the deployment network cannot be teamed. Only secondary NICs can be teamed.

Known issues with Ubuntu files with long names:

- Extracting the Ubuntu ISO under Windows (in the Media Server) might cause problems as Windows doesn't handle long names properly. One solution is to extract the ISO on a Linux server and then copy the files back to the Windows Media Server.

Other known errors:

- If you encounter an error "`RTNETLINK answers: File exists`" during the bonding step, this is safe to ignore. It usually happens when the network route is conflicted. You can troubleshoot further by checking the routing table (`route -n`).

Resources and additional links

HPE Servers
hpe.com/servers

HPE Storage
hpe.com/storage

HPE Networking
hpe.com/networking

HPE Insight Control Server provisioning
https://www.hpe.com/us/en/software/servers-insight-control.html

HPE Insight Control Server Provisioning documentation
http://h17007.www1.hpe.com/us/en/enterprise/servers/solutions/info-library/index.aspx?cat=insightmanagement&subcat=ic

Docker EE
https://docs.docker.com/enterprise/

To help us improve our documents, please provide feedback at hpe.com/contact/feedback.

Sign up for updates