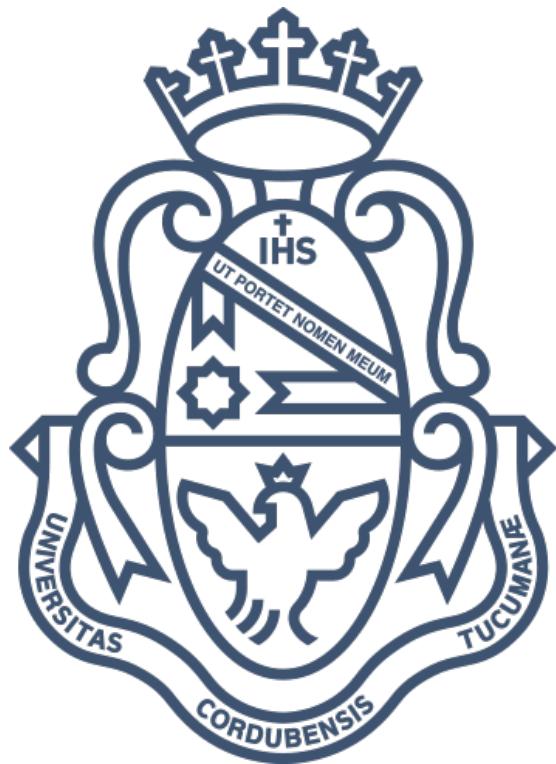


Universidad Nacional de Córdoba



April 27, 2016

Facultad de Ciencias Exactas, Físicas y Naturales

Infraestructura tecnológica virtual con automatización y orquestación.

Alumnos: Juan Arese, Werner Diers

Director de PI: Eschoyez, Maximiliano Andrés

Co-director de PI: Migliazzo, Oscar Andrés

Índice

| | |
|---|----------|
| I. Introducción | 6 |
| 1. Resumen del Proyecto Integrador | 6 |
| 1.1. Descripción | 6 |
| 1.2. Objetivos | 6 |
| 1.3. Intereses personales | 7 |
| 1.4. Intereses Institucionales | 7 |
| 1.5. Metodología | 7 |
| 1.6. Requerimientos | 8 |
| 1.7. Cronograma a seguir | 8 |
| 1.8. Objetivo a alcanzar en cada etapa: | 9 |
| II. Marco teórico | 9 |
| 2. Sistema operativo | 10 |
| 2.1. CentOS | 10 |
| 2.2. Debian | 11 |
| 2.3. FreeBSD | 12 |
| 2.4. Solaris | 13 |
| 3. Virtualización | 14 |
| 3.1. Tipos de virtualización | 14 |
| 3.1.1. Virtualización completa | 14 |
| 3.1.2. Para-virtualización | 15 |
| 3.1.3. Para-virtualización de drivers | 15 |
| 3.1.4. Virtualización a nivel del sistema operativo | 15 |
| 3.1.5. Emulación | 16 |
| 3.2. Herramientas de virtualización | 16 |
| 3.2.1. Docker | 17 |
| 3.2.2. KVM/Qemu | 17 |
| 3.2.3. OpenVZ | 18 |
| 3.2.4. VirtualBox | 18 |
| 4. Aprovisionamiento | 19 |
| 4.1. Herramientas de aprovisionamiento | 19 |
| 4.1.1. Cobbler | 20 |
| 4.1.2. Fully Automatic Installation (FAI) | 20 |
| 4.1.3. Foreman | 21 |
| 4.1.4. Vagrant | 21 |
| 5. Orquestación | 22 |
| 5.1. Herramientas de orquestación | 22 |
| 5.1.1. Ansible | 23 |
| 5.1.2. Chef | 23 |

| | |
|--|-----------|
| 5.1.3. Puppet | 24 |
| 6. Protocolo PXE | 25 |
| 6.0.4. PXE APIs | 25 |
| 6.0.5. Funcionamiento de PXE | 26 |
| 7. SAMBA | 28 |
| 7.1. Arquitectura SAMBA | 28 |
| III. Desarrollo | 29 |
| 8. Elección de la plataforma | 29 |
| 8.1. Arquitectura de desarrollo | 29 |
| 9. KVM/Qemu | 31 |
| 9.1. Arquitectura | 31 |
| 9.2. Requerimientos de hardware | 32 |
| 9.3. Limitaciones de KVM | 33 |
| 9.4. Configuración de la red | 34 |
| 9.5. Creacion del sistema de virtualizacion qemu-KVM | 34 |
| 9.5.1. Instalación de paquetes | 34 |
| 9.5.2. Creacion de una red NAT | 34 |
| 9.5.3. Creacion de VMs | 35 |
| 10.Cobbler | 37 |
| 10.1. Tópicos generales de Cobbler | 37 |
| 10.1.1. Modelado | 37 |
| 10.1.2. Distros | 38 |
| 10.1.3. Profiles | 38 |
| 10.1.4. Systems | 38 |
| 10.1.5. Images | 39 |
| 10.1.6. Repositorios | 39 |
| 10.1.7. Import | 40 |
| 10.1.8. Kickstarts | 41 |
| 10.1.9. Snippets | 43 |
| 10.1.10. Integración con Puppet | 44 |
| 10.1.11. Replicate | 45 |
| 10.2. Creacion del sistema de aprovisionamiento y automatizacion Cobbler | 46 |
| 10.2.1. Instalación | 46 |
| 10.2.2. Configuración de servicios | 47 |
| 10.2.3. Repositorio local separado de Cobbler | 50 |
| 10.2.4. Importar imágenes ISO al servidor Cobbler | 52 |
| 11.Puppet | 52 |
| 11.1. Tópicos generales de Puppet | 52 |
| 11.1.1. Module | 52 |
| 11.1.2. Resources | 53 |
| 11.1.3. Manifests | 56 |

| | |
|--|-----------|
| 11.1.4. Catálogos | 56 |
| 11.1.5. Classes | 57 |
| 11.1.6. Funciones | 57 |
| 11.1.7. Metaparámetros | 57 |
| 11.1.8. Definición de nodos | 58 |
| 11.1.9. Node group | 59 |
| 11.2. Creacion del sistema de orquestacion Puppet | 59 |
| 11.2.1. Requerimientos de sistema y chequeo de versión de sistema operativo | 59 |
| 11.2.2. Chequeo de la configuración de red | 60 |
| 11.2.3. Instalación de puppetserver | 60 |
| 11.2.4. Configuraciones para los servidores: | 60 |
| 11.2.5. Instalar el paquete puppet-agent | 61 |
| 11.2.6. Configuraciones para los agentes | 62 |
| 11.2.7. Ejecutar puppet | 63 |
| 12. Automatización de Windows 7 | 63 |
| 12.1. El lado de Windows | 64 |
| 12.1.1. Instalar WAIK | 65 |
| 12.1.2. WinPE | 65 |
| 12.1.3. Instalar y personalizar Windows 7 | 66 |
| 12.1.4. Generalizar el equipo de referencia para preparar la imagen | 68 |
| 12.1.5. Capturar el equipo de referencia | 70 |
| 12.1.6. Autounattend.xml | 72 |
| 12.1.7. Crear nuevos medios de instalación de Windows 7 para la imagen personalizada | 84 |
| 12.2. El lado de Linux | 84 |
| 12.2.1. Integración con Cobbler | 86 |

Lista de tablas

| | |
|---|----|
| 1. Pasos automatización Windows 7 | 64 |
|---|----|

Lista de figuras

| | |
|---|----|
| 1. CentOS logo | 10 |
| 2. Debian logo | 11 |
| 3. FreeBSD logo | 12 |
| 4. Solaris logo | 13 |
| 5. Diagrama de virtualización completa | 15 |
| 6. Diagrama de paravirtualización | 15 |
| 7. Diagrama de virtualización a nivel de SO | 16 |
| 8. Docker logo | 17 |
| 9. KVM/Qemu logo | 17 |
| 10. OpenVZ logo | 18 |
| 11. VirtualBox logo | 18 |
| 12. Cobbler logo | 20 |
| 13. FAI logo | 20 |

| | | |
|-----|---|----|
| 14. | Foreman logo | 21 |
| 15. | Vagrant logo | 21 |
| 16. | Ansible logo | 23 |
| 17. | Chef logo | 23 |
| 18. | Puppet logo | 24 |
| 19. | APIs de PXE | 26 |
| 20. | Proceso de PXE | 27 |
| 21. | Arquitectura de desarrollo | 30 |
| 22. | Arquitectura de desarrollo en un entorno mixto. | 31 |
| 23. | Diagrama de arquitectura de KVM/Qemu | 32 |
| 24. | Detalles de conexión | 35 |
| 25. | Crear red virtual | 36 |
| 26. | Modelado de Cobbler | 38 |
| 27. | Cobbler Web | 50 |
| 28. | Paso de archivos a Windows | 67 |
| 29. | Paso de archivos a Windows | 67 |
| 30. | Paso de archivos a Windows | 68 |
| 31. | Modo auditoría | 68 |
| 32. | Carga WinPE.iso para extraer imagen | 69 |
| 33. | Prioridad de inicio en máquina virtual | 69 |
| 34. | Ejecución sysprep | 70 |
| 35. | Verificación de letras de unidad | 71 |
| 36. | ImageX capturando imagen | 72 |
| 37. | Windows System Image Manager | 73 |
| 38. | Idioma y teclado. | 73 |
| 39. | Idioma | 74 |
| 40. | Firewall activado por defecto. | 74 |
| 41. | Discos. | 75 |
| 42. | Configuración discos. Disco 0. | 75 |
| 43. | Discos. Particionamiento. | 76 |
| 44. | Discos. Particionamiento. | 76 |
| 45. | Discos. Particionamiento. | 77 |
| 46. | Discos. Particionamiento. | 77 |
| 47. | Instalación del SO. | 78 |
| 48. | Instalación del SO. Elección de la versión. | 78 |
| 49. | Elección de partición de instalación. | 79 |
| 50. | Aceptación de las condiciones de privacidad y uso. | 79 |
| 51. | Clave de producto. | 80 |
| 52. | Activación del producto. | 80 |
| 53. | Hostname, organización, propietario y zona horaria. | 81 |
| 54. | Idioma, teclado, regional. | 81 |
| 55. | Organización, propietario y zona horaria. | 82 |
| 56. | Configuración de red. | 82 |
| 57. | Contraseña administrador. | 83 |
| 58. | Cuentas de usuario, administrador. | 83 |
| 59. | Cuentas de usuario, invitado. | 84 |

Capítulo I.

Introducción

1. Resumen del Proyecto Integrador

El Sistema desarrollado en este Proyecto Integrador, pretende facilitar algunas de las tareas que los administradores de los laboratorios realizan en las salas de informática. Las funcionalidades desarrolladas permiten a los administradores (o incluso a cualquier persona sin conocimiento técnico) crear máquinas virtuales con un solo click, dichas máquinas virtuales cuentan con todos los servicios y programas necesarios para los alumnos de las diferentes carreras o inclusive se pueden crear máquinas virtuales con otros perfiles, como un perfil para docentes. Además, permite orquestar las políticas a seguir de las diferentes máquinas de la red.

1.1. Descripción

El sistema de infraestructura virtual con automatización y orquestación, tiene como objetivo principal brindar una herramienta a los administradores de laboratorios que facilite la preparación y configuración de sus aulas de manera simple.

El sistema está dividido en dos partes. Una parte de la herramienta está destinada al despliegue en masa de máquinas virtuales. Permite al administrador, crear múltiples máquinas virtuales con escasa interacción humana, de forma automática. Pudiéndose especificar el sistema operativo deseado y componentes de hardware. Del mismo modo, en un laboratorio con máquinas físicas, es posible realizar el despliegue a través de la red.

La segunda parte está destinada a la administración de la configuración de las máquinas virtuales. El sistema permite aplicar cambios de configuración y políticas a un conjunto de máquinas como también a máquinas particulares. Esto evita que el administrador tenga que preparar cada estación de trabajo de a una por vez, disminuyendo la carga de trabajo y el tiempo requerido para llevar a cabo la tarea.

El sistema incluye desarrollo en el lenguaje de programación Python, bash y el lenguaje propio de Puppet, la herramienta utilizada para la orquestación. El servidor de aprovisionamiento, haciendo uso del protocolo PXE, será el encargado de atender las peticiones de los dispositivos para su instalación.

Este Proyecto Integrador servirá como base para futuros Proyectos.

1.2. Objetivos

Principal:

- Desarrollar un sistema para manejar automatismos administrados de políticas de una institución o empresa, para que el administrador pueda configurar las máquinas virtuales o físicas que se utilizan en la misma. Como caso particular se tomará un laboratorio informático de aprendizaje.

Secundarios:

- Estudiar Sistemas Operativos para servidor.
- Estudiar herramientas de virtualización.
- Estudiar herramientas de aprovisionamiento.
- Estudiar herramientas de administración de configuración.
- Analizar protocolos para inicio a través de la red como PXE.

Antecedentes de Proyectos similares

- No hay antecedentes en este tema.

1.3. Intereses personales

La principal motivación en este Proyecto Integrador es darle un final a la carrera de grado de Ingeniería en Computación realizando un proyecto que nos de más herramientas para nuestro futuro laboral. Abordamos temas como sistemas operativos, sistemas de archivos, protocolos, metodologías de desarrollo, redes de datos, programación en Python, Shell scripting, virtualización, diagramas UML.

1.4. Intereses Institucionales

La Facultad de Ciencias Exactas, Físicas y Naturales actualmente cuenta con alrededor de cinco aulas de informática, en las cuales se dictan materias de todos los ciclos y especialidades de ingeniería. La idea del Proyecto Integrador es desarrollar un sistema de infraestructura con automatización y orquestación que permita disminuir la carga de trabajo de los administradores de estas aulas y facilitar las tareas de mantenimiento de las mismas, cumpliendo con las políticas del área que administra las aulas.

1.5. Metodología

Para afrontar el Proyecto Integrador de la carrera se utilizó una metodología de desarrollo ágil de software basado en el desarrollo iterativo e incremental. El trabajo desarrollado en una unidad de tiempo es llamado una iteración, las cuales constan de un corto lapso de tiempo de entre una y tres semanas. Cada iteración se compone de un ciclo de vida que integra diversas etapas como planificación, definición de los requerimientos, investigación, diseño, codificación, pruebas y documentación. En cada iteración se agrega una nueva “funcionalidad” al sistema y a medida que avanzan los ciclos el sistema aumenta de tamaño, por esto lo llamamos incremental. Otra característica de la metodología ágil que se utilizó, es una comunicación fluida con el “cliente” que en este caso son los Directores del Proyecto Integrador, de los que también se obtienen los requerimientos. Esto permite una buena retroalimentación, con la cual se devuelven correcciones . No ser estrictos con la documentación es una característica que se tomó del desarrollo ágil, tener todo anotado luego facilita la generación del informe.

Lugar previsto para la realización:

- Laboratorio de Arquitectura de Computadoras, Facultad de Ciencias Exactas, Físicas y Naturales.

Requerimiento de Instrumental y Equipos:

- Computadora personal.

Inversión económica:

- Inversión provista por el alumno: ninguna
- Apoyo económico externo a la Facultad: ninguno.

1.6. Requerimientos

Los requerimientos del sistema se obtuvieron directamente desde el Director y el Codirector del Proyecto Integrador.

1. La herramienta debe poder aprovisionar distintos sistemas operativos:

CentOS

Ubuntu

Windows

2. La herramienta debe poder aprovisionar máquinas virtuales con o sin GUI .
3. La herramienta debe aprovisionar a través de la red.
4. La herramienta debe poder aprovisionar utilizando plantillas.
5. La herramienta debe poder aprovisionar utilizando repositorios locales.
6. La herramienta debe poder actualizar los repositorios locales.
7. La herramienta debe poder setear políticas a las máquinas virtuales, utilizando Puppet.
8. La herramienta debe ser escalable, integrar nuevas máquinas virtuales fácilmente.
9. La herramienta debe utilizar licencias de código abierto.
10. La herramienta debe estar implementada en la versión más actual al momento de realizar el Proyecto Integrador.

1.7. Cronograma a seguir

El cronograma está dividido en seis etapas diferentes:

- Familiarización con el sistema operativo GNU/Linux elegido.
- Investigación de diferentes herramientas de virtualización.
- Investigación de diferentes herramientas de aprovisionamiento.
- Pruebas de las herramientas seleccionadas.
- Pruebas de la herramienta de administración de configuración.
- Preparación y desarrollo del informe del trabajo final y cierre del mismo.

1.8. Objetivo a alcanzar en cada etapa:

- **Primera etapa:** conocer el sistema operativo GNU/Linux. La principal motivación, es la de informarse sobre la base donde se implementará el sistema desarrollado en este Proyecto Integrador.
- **Segunda etapa:** obtener los conocimientos suficientes para crear y administrar máquinas virtuales.
- **Tercera etapa:** conocer cuáles son las herramientas disponibles y utilizadas en ambientes de producción para el aprovisionamiento de máquinas, teniendo en cuenta que deben ser de código abierto, analizarlas y elegir la más apropiada para realizar el Proyecto Integrador.
- **Cuarta etapa:** implementar las herramientas seleccionadas en conjunto.
- **Quinta etapa:** realizar la implementación conjunta de todas las herramientas automatizando la instalación y realizar configuraciones pertinentes en los sistemas ya instalados, por medio de Puppet.
- **Sexta etapa:** desarrollo del informe del trabajo final

Capítulo II. Marco teórico

Previamente al abordaje del desarrollo, requerimos conocimientos teóricos que nos posibiliten comprender el entorno donde se ejecutarán las aplicaciones que componen el Sistema, las herramientas que darán soporte o permitirán cumplir con las funcionalidades previstas y las que se utilizarán para desarrollar. Algunas de las mismas fueron explícitamente solicitadas en los requerimientos. El resto de las herramientas, que están implícitas en los requerimientos, precisaron de investigación y pruebas para conocer si permiten cumplir estos requerimientos y además, si hay varias opciones, elegir la más conveniente. Podemos dividir esta sección del informe en áreas diferentes:

- Sistema operativo
- Virtualización
- Aprovisionamiento
- Orquestación
- Protocolo PXE
- SAMBA

La información contenida en esta sección se desprende de las investigaciones realizadas en cada una de las etapas del Proyecto Integrador.

2. Sistema operativo

2.1. CentOS



Figura 1: CentOS logo

CentOS es un distribución Linux empresarial, basada en Red Hat Enterprise Linux. CentOS concuerda con la política de distribución de Red Hat y apunta a ser binariamente compatible en su totalidad. Cada versión de esta distro tiene soporte por siete años en cuestiones de actualizaciones de mantenimiento y seguridad, lo que se traduce en un ambiente confiable, predecible, reproducible, de bajo mantenimiento y seguro.

La principal ventaja de esta distro es que se obtiene un conjunto estable de la mayoría de paquetes que por lo general sólo incluyen correcciones de errores. En su última versión, CentOS 7 solo está disponible para la arquitectura x86_64, y representa un gran cambio frente a versiones anteriores del sistema operativo, como la inclusión de systemd, Gnome 3, GRUB 2, y el sistema de archivos XFS. El entorno de escritorio KDE también forma parte de la oferta de CentOS 7.

Principales novedades de CentOS 7:

- Actualización del núcleo del sistema: Kernel 3.10.0.
- Soporte para Linux Containers.
- Inclusión de VMware Tools y controladores de gráficos 3D.
- OpenJDK-7 como JDK por defecto.
- Cambio a systemd.
- Cambio a firewalld y GRUB2 .
- XFS es el sistema de archivos por defecto y permite escalar la capacidad de almacenamiento del sistema hasta 500 terabytes. XFS es un sistema de archivos de 64 bits con journaling de alto rendimiento, y está especialmente indicado para discos grandes (superiores a 1 TB). No obstante y para necesidades menos exigentes se pueden emplear otros sistemas de archivos, como Ext4.
- iSCSI y FCoE (Fiber Channel over Ethernet) en el espacio del Kernel.
- Soporte para PPTv2 (Precision Time Protocol).
- Soporte para tarjetas Ethernet 40G.
- Soporte UEFI.

En cuanto a systemd, es el reemplazo de init como demonio para iniciar servicios, procesos y recursos del sistema. Systemd es la nueva forma predeterminada de iniciar los sistemas Linux, y ha sido adoptado por Red Hat, Debian y Ubuntu, entre otros. CentOS 7 es compatible con Microsoft Active Directory (y obviamente con Red Hat), por lo que puede trabajar con facilidad en entornos heterogéneos. CentOS 7 incluye PCP (Performance Co-Pilot), un conjunto de frameworks y servicios en tiempo real para supervisar y monitorizar el rendimiento del sistema.

2.2. Debian



Figura 2: Debian logo

Debian es una distribución libre, por completo manejada por la comunidad, no está basada en ninguna otra distribución y por el contrario gran parte de las distribuciones actuales están basadas en ella. Debian es famoso por filosofía de estabilidad ante todo, por eso mismo, no tiene un cronograma de lanzamiento de nuevas versiones. Estas se liberán cuando estén realmente listas. Es una distribución ampliamente utilizada.

En su última versión, Debian 8 Jessie, las siguientes arquitecturas tienen soporte: x32 (i386), x86-64 (amd64), Motorola/IBM PowerPC, MIPS, IBM S/390 y ARM.

Este release incluye el nuevo estándar sistema de inicio *systemd*.

Principales características de Jessie:

- Actualización de administrador de paquetes: apt 1.0.9.8.1
- Núcleo del sistema: Linux kernel 3.16
- Cambio a systemd.
- Entornos gráficos : Gnome 3.14, KDE 4.14,
- Los puertos para el kernel de FreeBSD (kfreebsd-amd64 y kfreebsd-i386), incluídos para versiones anteriores no son parte de esta versión.
- Soporte UEFI para amd64, i386 y arm64.

Esto permitiría aprovechar todas las ventajas de FreeBSD en el servidor, brindándole la opción de usar Debian a los usuarios del servidor y eliminando la necesidad de que se familiaricen con FreeBSD.

2.3. FreeBSD



Figura 3: FreeBSD logo

FreeBSD es un sistema operativo basado en BSD para arquitecturas Intel (x86 e Itanium), AMD64, AlphaTM y UltraSPARC.

FreeBSD viene con una excelente colección de herramientas de sistema como parte del sistema base. A pesar de esto, existen otras que no vienen incluidas y se necesitan instalar para utilizarlas. FreeBSD ofrece dos tecnologías complementarias para instalar software de terceros en el sistema: la Colección de puertos o ports de FreeBSD y los paquetes binarios. Los paquetes binarios son archivos simples que descargamos desde repositorios. Contienen una copia de los programas binarios precompilados de la aplicación y se pueden manipular con las herramientas de gestión de paquetes de FreeBSD: `pkg_add`, `pkg_delete`, `pkg_info`, etc. Por otro lado, existen ciertos pasos que se deben llevar a cabo para compilar un programa (descargar, desempaquetar, parchear, compilar e instalar). Los ficheros que conforman un port permiten que el sistema se encargue de todo esto, mediante un conjunto simple de órdenes. La colección de puertos para instalar se encuentra en `/usr/ports`.

FreeBSD proporciona compatibilidad binaria con muchos otros sistemas operativos tipo UNIX, como Linux. Esto es necesario, ya que muchos desarrolladores y compañías sólo desarrollan para Linux. La compatibilidad binaria permite a los usuarios utilizar en FreeBSD cerca del 90% de las aplicaciones desarrolladas para Linux sin que sea necesario realizar alguna modificación sobre la aplicación.

Otras características:

- Servicios multiusuario que permiten a mucha gente usar el sistema FreeBSD simultáneamente.
- Conexión de redes TCP/IP muy robusta, con soporte para estándares industriales.
- La protección de memoria que garantiza que las aplicaciones (o los usuarios) no se estorben los unos a los otros.
- Compatibilidad binaria con muchos programas nativos de Linux, SCO, SVR4, BSDI y NetBSD.
- Soporte para multiprocesamiento simétrico con múltiples CPUs.

2.4. Solaris



Figura 4: Solaris logo

Solaris es un sistema operativo de tipo Unix desarrollado por Sun Microsystems desde 1992 como sucesor de SunOS. Es un sistema certificado oficialmente como versión de Unix. Aunque Solaris fue desarrollado como software privado, la mayor parte de su código se ha liberado como proyecto de software libre denominado OpenSolaris. Solaris es famoso por su escalabilidad, especialmente en sistemas SPARC. Sun Solaris se ejecuta sobre la arquitectura SPARC en 32 y 64 bits, o sobre procesadores x86 (incluidos Intel y AMD). Sin embargo, en agosto de 2010, Oracle decidió interrumpir la publicación y distribución de OpenSolaris.

Solaris tiene una reputación de ser muy adecuado para el multiprocesamiento simétrico (SMP), soportando un gran número de CPUs. Históricamente Solaris ha estado firmemente integrado con la plataforma hardware de Sun, SPARC, con la cual fue diseñado y promocionado como un paquete combinado. Esto proporcionaba frecuentemente unos sistemas más fiables pero con un coste más elevado que el del hardware de PC.

A partir de su versión 10, Sun Microsystems ha promocionado Solaris con sus propias estaciones de trabajo y servidores de 64 bits basados en procesadores AMD Opteron e Intel Xeon, así como también en sistemas de 32 bits. Esta versión añadió soporte para paravirtualización cuando es utilizada como “sistema operativo invitado” en ambientes basados en Xen.

En su última versión, 11.3, sus principales características son:

- Incluye una nueva versión de OpenStack (Juno) con soporte para topologías de red adicionales y nuevos servicios.
- SNAT, soporte IPv6.
- Pools de almacenamiento.
- Aprovisionamiento de máquinas (bare metal provisioning) como servicio .
- Incluye soporte para desarrollo basado en la API REST utilizando el Demónio de Administración Remota (permite configuración remota de los sistemas Oracle usando Python, C y Java).
- Sistema de archivos ZFS.
- Solaris Containers

3. Virtualización

Virtualización es un término amplio para software ejecutándose, usualmente sistemas operativos, de manera concurrente y aislada de otros programas en el mismo sistema. Muchas de las implementaciones de virtualización utilizan un “hypervisor”, una capa de software que controla el hardware y provee sistemas operativos huéspedes con acceso a los dispositivos de hardware subyacentes. El hypervisor permite ejecutar múltiples sistemas operativos en el mismo sistema físico ofreciendo hardware virtualizado al sistema operativo huésped. Esta tecnología, provee un conjunto de herramientas para aumentar la flexibilidad y reducir los costos, los cuales son tópicos importantes en cualquier empresa o institución. En esencia, la virtualización incrementa la flexibilidad desacoplando un sistema operativo y los servicios y aplicaciones soportados por él, de una plataforma de hardware física específica, permitiendo el establecimiento de múltiples entornos virtuales sobre una plataforma de hardware compartida. Estos entornos pueden ser creados localmente o aprovisionados extremadamente. La virtualización se destaca también apoyando la innovación a través del uso de entornos virtuales para practicar y aprender. Un estudiante puede comenzar un curso o trabajo un entorno de sistema conocido, estándar y aislado del entorno de producción; si se produce algún tipo de daño solo afecta al sistema virtual. Además se puede establecer entornos únicos de software para el aprendizaje sin demandar el uso exclusivo de recursos de hardware. Aunque en comparación los costos de inversión para tener un número elevado de máquinas físicas son mucho mayores que el costo para invertir en un servidor con altos recursos para realizar la virtualización, se podría decir que la virtualización posee inconvenientes vinculados con sus exigentes requerimientos de hardware, en cuanto a capacidad de procesamiento y de memoria RAM y de almacenamiento. Otra desventaja es que del sistema de virtualización depende del sistema operativo anfitrión. Es decir, el anfitrión es el punto débil del sistema ya que se comparte por todos los sistemas virtualizados, si se rompe éste, se rompen todas las máquinas virtuales.

3.1. Tipos de virtualización

3.1.1. Virtualización completa

Consiste en la virtualización de paquetes y herramientas para correr de forma totalmente virtualizada, sin modificaciones, sistemas operativos huéspedes. Este modo cuenta con la ventaja de consolidar sistemas viejos en hardware nuevo, más eficiente y reducir el espacio físico y costos de operación relativos al consumo energético y refrigeración de estos sistemas menos eficientes. La virtualización completa ofrece, sin embargo, menor rendimiento de entrada/salida que instalaciones nativas (también llamadas “bare-metal” o “metal-pelado”) de sistemas operativos. Por ejemplo el software KVM, Xen, VMware Workstation o VirtualBox hacen uso de esta técnica. Cabe destacar que en el caso de KVM se requiere soporte de hardware para ejecutar la virtualización, ya sea con procesadores Intel o AMD.

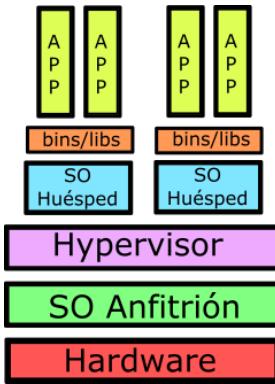


Figura 5: Diagrama de virtualización completa

3.1.2. Para-virtualización

Para-virtualización es una técnica de virtualización que implica ejecutar versiones modificadas de los sistemas operativos. El sistema operativo para-virtualizado es modificado para que se de cuenta de que está siendo virtualizado, ofreciendo una habilidad aumentada para la optimización, ya que el huésped está al tanto de su entorno. El rendimiento está generalmente muy cerca de la ejecución nativa de sistemas operativos no virtualizados. Por ejemplo, utilizan esta técnica KVM, XEN y VMware Server ESX.

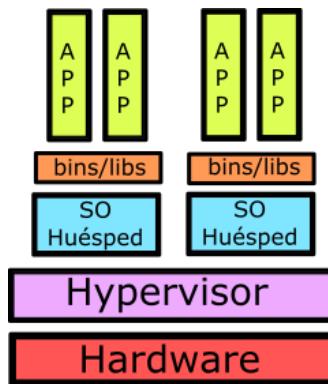


Figura 6: Diagrama de paravirtualización

3.1.3. Para-virtualización de drivers

La para-virtualización y la virtualización completa pueden ser combinadas para permitir a sistemas operativos no modificados recibir un rendimiento cercano de entrada/salida al de ejecución nativa, por medio de drivers para-virtualizados en sistemas operativos completamente virtualizados.

3.1.4. Virtualización a nivel del sistema operativo

También llamada virtualización basada en contenedores, esta técnica virtualiza un servidor físico a nivel del sistema operativo, permitiendo que múltiples servidores virtuales aislados y seguros se ejecuten sobre un solo servidor físico. Con la virtualización basada

en contenedores, no existe la sobrecarga asociada con tener a cada huésped ejecutando un sistema operativo completamente instalado. Este enfoque también puede mejorar el rendimiento porque hay un solo sistema operativo encargándose de los avisos de hardware. Una desventaja de la virtualización basada en contenedores, sin embargo, es que cada invitado debe utilizar el mismo sistema operativo que utiliza el host. Por ejemplo, Jaulas con Warden en FreeBSD, OpenVZ o Linux-Vserver usan esta técnica.

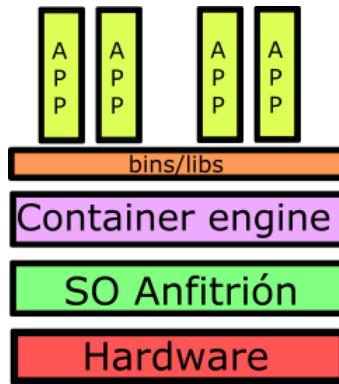


Figura 7: Diagrama de virtualización a nivel de SO

3.1.5. Emulación

Un emulador es hardware o software que permite a un sistema de computación comportarse como otro sistema. Generalmente, un emulador permite a un sistema correr software o utilizar dispositivos periféricos diseñados para el otro sistema. Por ejemplo Qemu es un emulador muy usual en ingeniería.

3.2. Herramientas de virtualización

Algunas de las herramientas más utilizadas para virtualizar son las siguientes, sin embargo como uno de los requisitos es utilizar herramientas de código abierto, no se indagó acerca de VMWare:

- Docker
- KVM/Qemu
- OpenVZ
- VirtualBox
- VMWare Workstation

3.2.1. Docker

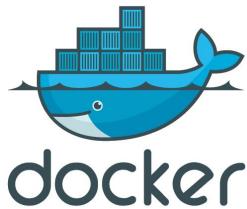


Figura 8: Docker logo

Docker es otra herramienta de virtualización para Linux basada en contenedores. La idea detrás de Docker es crear contenedores ligeros y portables para las aplicaciones de software que puedan ejecutarse en cualquier máquina con Docker instalado, independientemente del sistema operativo que la máquina tenga por debajo, facilitando así también los despliegues.

El contenedor Docker se puede desplegar en cualquier otro sistema (que soporte esta tecnología), con lo que se ahorra el tener que instalar en este nuevo entorno todas aquellas aplicaciones que normalmente se utilicen.

Un contenedor Docker no contiene todo un sistema completo, sino únicamente aquellas librerías, archivos y configuraciones necesarias para desplegar las funcionalidades que contenga. Asimismo Docker se encarga de la gestión del contenedor y de las aplicaciones que contenga.

Para obtener esta fluidez Docker extiende LXC (LinuX Containers), un sistema de virtualización ligero que permite crear múltiples sistemas totalmente aislados entre si sobre la misma máquina o sistema anfitrión. Y todo dado que no se emula un sistema operativo completo, sólo las librerías y sistemas de archivos necesarios para la utilización de las aplicaciones que se tengan instaladas en cada contenedor.

En las últimas versiones de Docker se ha introducido drivers de Docker y una librería llamada libcontainer, que ayuda a que Docker sea totalmente multiplataforma, teniendo compatibilidad con Windows y Mac OS X, además de Linux.

3.2.2. KVM/Qemu



Figura 9: KVM/Qemu logo

KVM (Kernel-based Virtual Machine), desarrollado por Red Hat Enterprise Linux, es una infraestructura de virtualización completa para el kernel de Linux que lo transforma en un hypervisor. Fue incorporado a la línea principal del kernel Linux en la versión 2.6.20.

KVM requiere un procesador con hardware que permita extensión para virtualización (Intel VT o AMD-V). También está disponible para instalarlo desde los “ports” de FreeBSD en la forma de módulos de kernel.

La para-virtualización tiene soporte para ciertos dispositivos en Linux, OpenBSD, FreeBSD y Windows (entre otros) utilizando la API VirtIO. Se tiene placa Ethernet, un controlador de entrada/salida de disco paravirtual, gráficos VGA.

Qemu puede utilizarse como emulador y como virtualizador. Cuando se utiliza como virtualizador, Qemu toma un rendimiento cercano al nativo. Para ello, debe ejecutarse bajo el hypervisor Xen o KVM. En conjunto KVM/Qemu, KVM es quien hace las veces de árbitro del acceso al CPU y memoria, y Qemu emula los recursos de hardware.

3.2.3. OpenVZ



Figura 10: OpenVZ logo

OpenVZ es una herramienta de virtualización para Linux basada en contenedores. OpenVZ crea múltiples contenedores aislados en un servidor físico y asegurando que las aplicaciones no entren en conflicto. Utiliza un kernel de Linux modificado y por consiguiente sólo puede correr Linux. Todos los contenedores comparten la misma arquitectura y versión del kernel. Cada contenedor se comporta como un servidor autónomo. Ya que OpenVZ emplea un modelo de kernel único, es tan escalable como kernel Linux 2.6, lo que significa que soporta hasta 64 CPUs y hasta 64 GiB de RAM. Un entorno virtual único se puede escalar hasta el equipo físico entero. También cuenta con migración en vivo que posibilita mover un contenedor de un servidor físico a otro sin apagar el contenedor. Un propietario (root) de un servidor físico OpenVZ (conocido como Nodo de Hardware) puede ver todos los procesos y archivos de los contenedores. Esto hace la administración masiva de escenarios posible: se puede ejecutar un simple script de intérprete de comandos que actualice todos (o sólo algunos seleccionados) los contenedores a la vez.

3.2.4. VirtualBox



Figura 11: VirtualBox logo

VirtualBox, desarrollado por Oracle, es un virtualizador completo de propósito general para hardware x86, orientado al uso para servidor, escritorio y embebido. Como software de código abierto, se puede utilizar bajo la licencia GNU General Public License 2 (GPL2).

AL día de la fecha, VirtualBox corre en Windows, Linux, Macintosh y Solaris y soporta una amplia variedad de sistemas operativos, entre ellos RHEL(7,6,5,4), Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris y OpenSolaris, OS/2, y OpenBSD.

4. Aprovisionamiento

En general, aprovisionamiento, significa proveer o hacer que algo esté disponible. El término es utilizado en un gran variedad de contextos en el área de Tecnologías de Información. En este Proyecto Integrador, el término hace referencia a lo siguiente: Aprovisionamiento es el conjunto de acciones para preparar una máquina virtual, con el sistema apropiado, datos y software, y dejarla lista para su operación.

- Tareas típicas que se tienen que llevar a cabo para que esto suceda son:
- Seleccionar el conjunto de hardware virtualizado (memoria RAM, disco, cantidad de procesadores asignados, placa de red, etc) para crear una máquina virtual bare-metal.
- Cargar el sistema operativo adecuado.
- Configurar el sistema (dirección IP, gateway, DNS, hostname, MAC, etc).
- Actualizar el sistema y aplicar parches.
- Cargar el conjunto de aplicaciones necesarias.
- Configurar el sistema para adaptarlo a las políticas definidas de la institución.

En resumen, el aprovisionamiento de máquinas virtuales se realiza basado en los recursos disponibles y en los requisitos específicos de cada máquina virtual, según sea la funcionalidad que se le vaya a dar.

4.1. Herramientas de aprovisionamiento

Algunas de las herramientas de código abierto más utilizadas para aprovisionar son las siguientes:

- Cobbler
- FAI
- Foreman
- Vagrant

4.1.1. Cobbler



Figura 12: Cobbler logo

Cobbler es un servidor Linux de aprovisionamiento que centraliza y simplifica el control de los servicios incluyendo DHCP, TFTP y DNS con el propósito de realizar instalaciones de sistemas operativos basadas en la red. Puede ser configurado para PXE, reinstalaciones y huéspedes virtualizados utilizando Xen, KVM o VMWare como también dispositivos físicos. Está dirigido especialmente a Red Hat Linux y sus derivados, pero es posible configurarlo para que inicie con PXE otras distribuciones de Linux como Debian, Ubuntu o Knoppix. Es una herramienta que se encuentra en creciente desarrollo y cada vez añade más soporte a distintas distros e incluso a FreeBSD y a futuro Windows. Actualmente cuenta con poco soporte para el sistema operativo de Microsoft.

Cuenta con un sistema integrado para administración de configuración pero también cuenta con soporte para la integrar la herramienta de administración de configuración Puppet. Cobbler se basa en el mecanismo de Kickstart y ofrece perfiles de instalación que pueden ser aplicados a una o muchas máquinas. La información de contenida en un plantilla kickstart puede ser modificada dinámicamente pasando variables (llamadas ksmeta) o utilizando snippets, donde se puede mantener el código común simplificando la lectura y minimizando el tamaño del archivo kickstart.

4.1.2. Fully Automatic Installation (FAI)



Figura 13: FAI logo

FAI es un sistema no interactivo para instalar, personalizar y administrar sistemas Linux y configuraciones de software en computadoras como también en máquinas virtuales y entornos chroot, desde pequeñas redes hasta una infraestructura grande escalable y clusters. Es una herramienta para la instalación totalmente automática de Debian y otras distros de Linux como Suse, Red Hat, Solaris, vía red, DVDs personalizados de instalación o en entornos chroot.

Algunas de las características más importantes:

- Instalar y actualizar Debian, Ubuntu, SUSE, Red Hat, etc.

- Despliegue centralizado y administración de configuración.
- Recuperació de desastre integrado.
- Fácil configuración de software RAID y LVM.
- Instalar máquinas virtuales usando KVM, Xen y VirtualBox.
- Control remoto vía SSH durante la instalación.

4.1.3. Foreman



Figura 14: Foreman logo

Foreman es una herramienta para el aprovisionamiento, configuración y monitorización de servidores físicos y virtuales. Puede aprovisionar máquinas bare-metal, virtualizadas y en la nube a través de instalaciones desatendidas por medio de DHCP, DNS, TFTP y PXE . Tiene una gran integración con software de administración de configuración como Puppet, Chef, Salt y otros por medio de plugins.

Algunas de sus características son:

- Descubrir, aprovisioinar y actualizar toda la infraestructura bare-metal.
- Crear y gestionar instancias entre nubes privadas y públicas.
- Agrupar hosts y dirigirlos en conjunto, sin importar la ubicación.
- Revenir cambios históricos para auditoría y resolución de problemas.

4.1.4. Vagrant



Figura 15: Vagrant logo

Vagrant provee entornos fáciles de reproducir, configurar construidos sobre tecnología industrial estándar. Vagrant es un software que crea y configura entornos de desarrollo virtuales aprovisionando sobre VirtualBox, VMware, KVM y contenedores Linux. Es una software que se encuentra una capa encima de estas herramientas. Luego herramientas de administración de configuración como Ansible, Chef, Salt, y Puppet pueden ser utilizadas para instalar y configurar automáticamente el software en la máquina.

5. Orquestación

La orquestación describe el alineamiento automatizado, la coordinación y la administración de complejos sistemas de computadoras, middleware y servicios. En este sentido, la orquestación se trata de alinear los requisitos de negocio con las aplicaciones, datos e infraestructura. Define las políticas y niveles de servicio a través de flujos de trabajo automatizados, aprovisionamiento y gestión de cambio. Esto crea una infraestructura alineada con la aplicación que puede ser escalada hacia arriba o abajo basándose en las necesidades de cada aplicación.

La orquestación también provee la gestión centralizada de los recursos. Por ejemplo, reduce el tiempo y esfuerzo para desplegar múltiples instancias de una sola aplicación. Cuando es necesario que se creen más instancias de diferentes aplicaciones, herramientas automatizadas pueden realizar tareas que, previamente, podían ser llevadas a cabo sólo por múltiples administradores.

Un escenario que se puede encontrar, por ejemplo, un administrador necesita desplegar una aplicación web, pero para hacerlo, primero debe crear el servidor de base de datos. Luego debe incluir en la base de datos todas las direcciones IP que pueden conectarse al servidor, y también agregar este nuevo servidor a la herramienta que lo monitorea, o abrir un puerto particular antes de proceder. Cada tarea expuesta puede ser automatizada, pero el conjunto de estas automatizaciones, junto con la coordinación secuencial de las mismas, realizada sin tener en cuenta el tipo de sistema operativo en el que corren, describen un proceso, en el cual actúa la orquestación.

No hay que confundir los términos automatización y orquestación. Éstos se podrían comparar con tarea y proceso.

La optimización de un proceso, por ejemplo, no se puede conseguir completamente por la automatización. A la automatización le concierne una tarea: ejecutar un servidor web, configurar un servidor web, detener un servicio. A la orquestación, sin embargo, le concierne la ejecución de un flujo de trabajo (si se quiere automatizado) de un proceso. Un proceso de aprovisionamiento lleva a cabo múltiples tareas e involucrar múltiples sistemas. El objetivo de la orquestación no es sólo ejecutar automáticamente un servidor, lo cual aumenta la velocidad en el proceso de despliegue y lleva las aplicaciones a producción más rápido. También permite una oportunidad para optimizar aquellos procesos para mejorar aún más la velocidad de despliegue.

Una de las maneras más simples de optimizar un proceso es eliminar los pasos repetitivos. Entonces, automatización trata acerca de codificar tareas y orquestación acerca de codificar procesos. Esta última toma ventaja de la automatización para reutilizar bloques básicos.

5.1. Herramientas de orquestación

Algunas de las herramientas de código abierto más utilizadas para orquestar son las siguientes:

- Ansible
- Chef
- Puppet

5.1.1. Ansible



Figura 16: Ansible logo

Es una plataforma para configurar y administrar computadoras. Combina instalación multi-nodo, ejecuciones de tareas ad hoc y administración de configuraciones.

Ansible distingue dos tipos: controladores y nodos. Primero, existe una única máquina de control donde la orquestación comienza. Los nodos son manejados desde esa máquina por el servicio OpenSSH. La máquina de control conoce a los nodos a través de un inventario. Esta herramienta usa una arquitectura sin agentes, es decir, los nodos no necesitan instalar ni ejecutar en segundo plano ningún proceso que se comunique con la máquina de control. Sin embargo, los nodos deben contar con Python ≥ 2.4 y las máquinas de control con Python 2.6.

Dispone de módulos que trabajan sobre JSON y la salida estándar puede ser escrita en cualquier lenguaje. Nativamente utiliza YAML para describir configuraciones de los sistemas.

Los sistemas operativos soportados en las máquinas de control son la mayoría de las distribuciones Linux y Unix (Red Hat, Debian, CentOS, OSX, y BSD) entre otros excepto Windows.

Ansible puede instalarse en ambientes virtualizados, nubes públicas y privadas, incluyendo VMWare, OpenStack, AWS, Eucalyptus, KVM y CloudStack.

5.1.2. Chef



Figura 17: Chef logo

Es una herramienta de automatización de infraestructura de sistemas o administración de configuraciones. Se enfoca en seguir un conjunto de pasos (llamados recetas) con el propósito de presentar un producto final ya listo para trabajar y/o probar.

Existen 2 tipos de versiones:

Chef Server está enfocado a ser el servidor central que permite suministrar a los diferentes nodos clientes con las diversas configuraciones necesarias, las cuales se mantienen alojadas en el servidor. El cliente sondea periódicamente al Chef Server para corroborar las últimas políticas y estado de la red, en caso que haya algún parámetro desactualizado, el cliente lo actualiza. Además ofrece balanceo de carga, escalabilidad, búsquedas rápidas entre otros.

Chef Solo es la versión de código abierto y reside localmente en el nodo, esto quiere decir que toda la información y recetas necesarias para configurar el nodo deben estar presentes en su disco duro. Esta herramienta utiliza Ruby-DLS para escribir las “recetas” (configuraciones de sistemas que describen como son manejadas las aplicaciones). Estas recetas, que pueden ser agrupadas para facilitar la administración, describen de forma secuencial una serie de recursos que deben estar en un estado particular.

Chef Server es soportado sobre RHEL/CentOS/Oracle Linux, y Ubuntu. Mientras que el soporte para los clients es AIX, RHEL/CentOS, FreeBSD, Mac OS X, Solaris (OS), Microsoft Windows, Ubuntu, ArchLinux, Debian, Fedora, y otros.

5.1.3. Puppet



Figura 18: Puppet logo

Es un sistema de orquestación que permite definir el estado de la infraestructura, forzando automáticamente que se llegue al estado correcto definido.

Puppet es una herramienta diseñada para administrar la configuración de sistemas similares a Unix y a Microsoft Windows de forma declarativa, es decir, se establece el estado requerido en vez como llegar al mismo.. El usuario describe los recursos del sistema y sus estados utilizando el lenguaje declarativo que proporciona Puppet. Esta información es almacenada en archivos denominados “manifiestos”. Puppet descubre la información del sistema a través de una utilidad llamada Facter, y compila los manifiestos en un catálogo específico del sistema que contiene los recursos y la dependencia de dichos recursos, estos catálogos son ejecutados en los sistemas de destino. La capa de abstracción de recursos permite a los administradores describir la configuración en términos de alto nivel, tales como usuarios, servicios y paquetes sin necesidad de especificar los comandos específicos del sistema operativo (como rpm, yum, apt).

Puppet funciona bajo la arquitectura cliente servidor donde un puppet master indica a sus agentes las configuraciones que deben aplicar. Además, los masters pueden aplicar manifiestos a sí mismos. Notar que hay dos etapas:

1. Compilar los catálogos
2. Aplicar los catálogos

Un catálogo es un archivo que describe los deseos de un estado de sistema para un nodo en particular. Enumera todos los recursos que necesitan ser administrados, así como las dependencias entre esos recursos.

En esta arquitectura, los nodos administrados corren la aplicación puppet-agent, usualmente en segundo plano y uno o más servidores corren la aplicación puppetmaster administrada por un servidor web (como Apache.) Periódicamente, los agentes piden al master el catálogo. El master, compila y corre el catálogo del nodo usando varias fuentes de información a las que tiene acceso. Una vez que recibe el catálogo, el agente chequea cada recurso descrito en él. Si encuentra algún recurso que no está en el estado deseado, se realizan los cambios necesarios para corregirlos. Luego de aplicar el catálogo, el agente envía un reporte al master.

Los sistemas soportados son Linux (Red Hat Enterprise y derivados, Debian, Ubuntu, Fedora), Unix (BSD, Mac OS X, Oracle Solaris, AIX) y Windows.

6. Protocolo PXE

El protocolo PXE (Preboot Execution Environment) es un estándar que les permite a computadoras, dentro de una red, que todavía no fueron cargadas con un sistema operativo, ser configuradas e iniciadas remotamente por un administrador. Utiliza una extensión de opciones del protocolo DHCP.

Las ventajas de utilizar PXE incluyen:

- La máquina cliente no necesariamente necesita un sistema operativo o un disco rígido.
- Como este protocolo es independiente del vendedor nuevos tipos de computadoras pueden ser añadidos a la red.

6.0.4. PXE APIs

Preboot Services API: Contiene muchas funciones de control e información.

Trivial File Transport Protocol (TFTP) API: Habilita la apertura y cierre de conexiones TFTP, la lectura desde una conexión TFTP y la escritura en otra.

User Datagram Protocol (UDP) API: Habilita la apertura y cierre de conexiones TFTP, la lectura desde una conexión UDP y la escritura en otra.

Universal Network Driver Interface (UNDI) API: Habilita el control básico de entrada/salida a través de la interfaz de red del cliente. Esto permite la utilización de protocolos universales de controladores para que el mismo controlador pueda ser usado en cualquier interfaz que soporte esta API.

El siguiente diagrama ilustra la relación entre los NBP y las APIs de PXE:

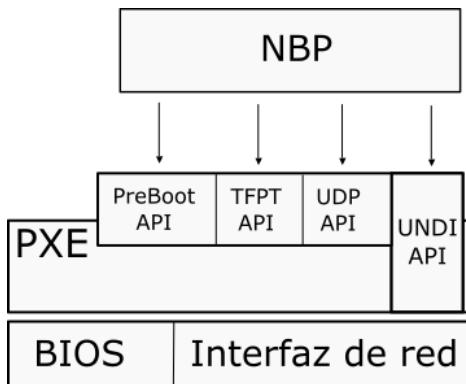


Figura 19: APIs de PXE

6.0.5. Funcionamiento de PXE

1. El cliente realiza un broadcast de un mensaje DHCPDISCOVER al puerto estándar DHCP (UDP 67). Un campo de opciones en este mensaje contiene:
 - a) Etiqueta para el identificador del cliente UUID.
 - b) Etiqueta para la versión de UNDI (Universal Network Device Interface) del cliente.
 - c) Etiqueta para la arquitectura del cliente.
 - d) La opción 60, Class ID, puesta a to “PXEClient:Arch:xxxxx:UNDI:yyyyy”.
2. El servidor DHCP responde enviando un mensaje DHCPOFFER al cliente en el puerto estándar DHCP (UDP 68). El mensaje contiene parámetros estándar DHCP:
 - a) Una dirección IP para el cliente.
 - b) Parámetros configurados por el administrador.
3. Del DHCPOFFER que es recibido, el cliente guarda:
 - a) La dirección IP.
 - b) Parámetros configurados por el administrador.
 - c) Una lista de Boot Servers del campo “Boot Server” en las etiquetas PXE del DHCPOFFER.
4. El cliente debe enviar una solicitud por la dirección del Boot Server al servidor y esperar por el acuse de recibo (acknowledgment – ACK).
5. El cliente selecciona y descubre un Boot Server. Este paquete puede ser enviado por broadcast al puerto 67. Este paquete es el mismo que el DHCPDISCOVER inicial en el paso uno, excepto que es codificado como DHCPREQUEST y ahora contiene lo siguiente:
 - a) La IP asignada al cliente por el servidor DHCP.
 - b) Una etiqueta con el identificador del cliente (UUID).
 - c) Una etiqueta con la versión de UNDI del cliente.

- d) Una etiqueta con la arquitectura del cliente.
- e) La opción 60, Class ID, puesta a to “PXEClient:Arch:xxxxx:UNDI:yyzzz”.
- f) El tipo de Boot Server en el campo de opciones de PXE.
6. El Boot Server envía un paquete DHCPACK unicast al cliente. Este ACK contiene:
- El nombre del archivo ejecutable.
 - Parámetros de configuración MTFTP.
 - Otras opciones necesarias para que el NBP pueda ser ejecutado.
7. El cliente descarga el archivo ejecutable utilizando TFTP (puerto 69). El archivo descargado y la ubicación del código descargado en memoria depende de la arquitectura de la CPU del cliente.
8. El cliente PXE determina si es necesaria la verificación de autenticidad del archivo descargado. Si se requiere se envía otro DHCPREQUEST preguntando por credenciales.
9. El cliente inicia la ejecución del código descargado.

El cliente PXE esperará por la información necesaria unos 60 segundos. La etapa DHCPDISCOVER puede repetirse hasta cuatro veces, con tiempos de espera de 4,8,16 y 32 segundos respectivamente. Si el cliente recibe la DHCPOFFER dentro de ese tiempo, se procederá con DHCPREQUEST. Si no, se detendrá con un error de PXE.

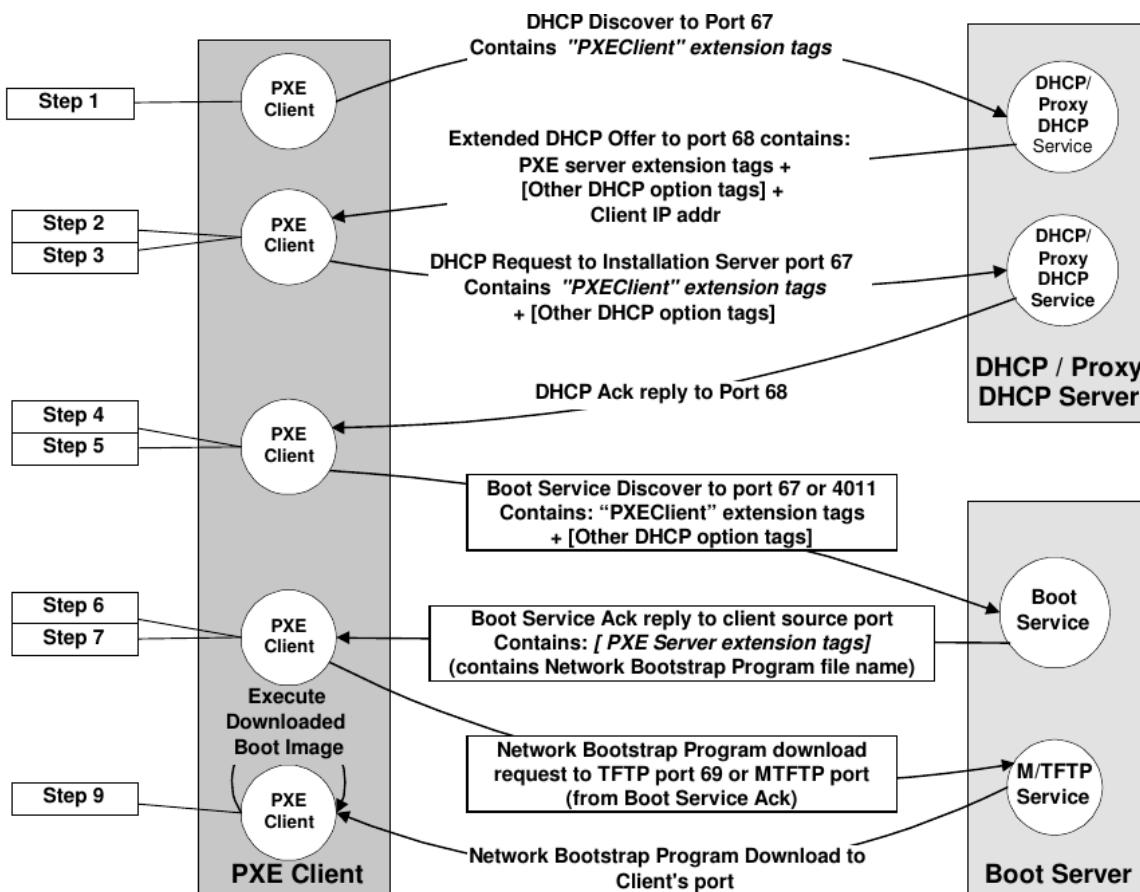


Figura 20: Proceso de PXE

7. SAMBA

SAMBA es un servidor SMB (Server Message Block) libre, desarrollado por Andrew Tridgell y que en la actualidad es mantenido por un grupo de personas de todo el mundo, como casi todos los proyectos distribuidos bajo la Licencia Pública General de GNU. SAMBA es capaz de ejecutarse en una gran cantidad de variantes Unix, como Linux, Solaris, SunOS, HP-UX, ULTRIX, Unix de Digital, SCO Open Server y AIX por nombrar tan sólo algunas. Con SAMBA podremos hacer que nuestro sistema Linux actúe como servidor SMB dentro de la red.

SAMBA es en sí un paquete muy complejo, que brinda a los usuarios Linux de un sin fin de posibilidades a la hora de interactuar con equipos Windows y Linux que estén coexistiendo en redes heterogéneas.

Los beneficios al instalar un servidor SAMBA en Linux son los siguientes:

- Compartir uno o más sistemas de archivos.
- Compartir impresoras, instaladas tanto en el servidor como en los clientes.
- SAMBA permite compartir entre máquinas Windows y Linux recursos.
- Siendo un recurso una carpeta o la impresora.

7.1. Arquitectura SAMBA

SAMBA está compuesta de un servidor y un cliente, así como de algunas herramientas que permiten realizar servicios prácticos o hacer un test de la configuración.

El servidor está compuesto de dos aplicaciones (llamadas demonios): smbd, núcleo del servidor, provee los servicios de autenticación y acceso a los recursos. nmbd, permite mostrar los servicios disponibles en SAMBA (visualización de los servidores SAMBA en la red, etc).

El cliente: smbclient es un cliente para Linux que provee una interfaz que permite la transferencia de archivos, el acceso a impresoras, etc.

smbtar: permite transferir de o hacia un archivo TAR bajo Linux.

testparm: comprueba la sintaxis del archivo smb.conf, el archivo de configuración de SAMBA.

El protocolo de comunicación que permite esta comunicación entre Windows y Linux se llama SMB (Server Message Block). Puesto a punto por Microsoft en 1987, retomando un concepto desarrollado por IBM en 1985 (NetBIOS), este protocolo se apoya sobre NetBEUI (así como sobre TCP/IP). El interés de TCP/IP proviene del hecho que es ampliamente adoptado. Por ello TCP/IP ya ha sido implementado en la mayoría de sistemas operativos (Unix, Linux, AmigaOS, MacOS, OS/2, etc) según el esquema siguiente:

- Aplicación
- SMB
- NetBios
- TCP/IP

- NetBeui
- IPX/SPX
- Controladores de red

Capítulo III.

Desarrollo

8. Elección de la plataforma

Una vez realizada la interiorización de las diferentes herramientas utilizadas en el mercado, se decidió unificar la plataforma para el desarrollo del Proyecto Integrador.

El sistema operativo base elegido fue CentOS 7, su última versión al día de la fecha. Esto es debido a que junto con su versión empresarial, es un sistema operativo muy robusto y confiable, orientado a servidores y uno de los más utilizado en el mercado. Además, la mayoría de las herramientas de virtualización, aprovisionamiento y orquestación son nativas de RedHat y por ende de CentOS. Finalmente se tuvo en cuenta la nueva estandarización del iniciador de sistemas, systemd, para esta versión del sistema operativo y para gran parte de las futuras entregas en diferentes distribuciones Linux, orientadas a servidor y también de escritorio.

Como herramienta de virtualización se optó por KVM, la cual fue, como se mencionó antes, desarrollada nativamente para esta distribución de Linux y que cuenta con una extensa documentación. A su vez, en el Laboratorio de Computación, existe un servidor que utiliza esta herramienta para servir a las terminales de ciertas aulas de informática. Consultando con los Directores del Proyecto, a profesionales en el tema se llegó a la conclusión que KVM es una herramienta estable y apta para un entorno de producción.

Para el aprovisionamiento de máquinas virtuales también por motivos de desarrollo nativo y recomendación de profesionales que la han utilizado se eligió a Cobbler, ya que es la más estable y fiable para entornos de producción. También cuenta con soporte integrado para orquestación con Puppet, herramienta que fue elegida por ser parte de los requerimientos del Proyecto.

8.1. Arquitectura de desarrollo

El siguiente esqueleto representa la arquitectura de desarrollo utilizada para todas las pruebas.

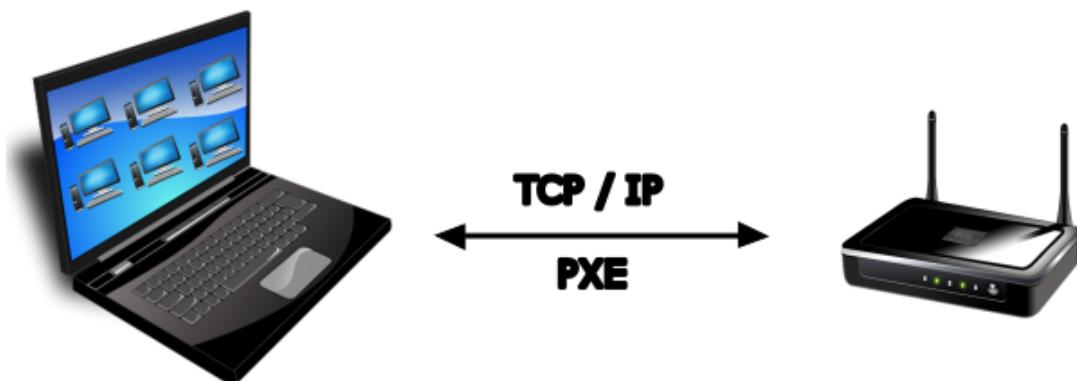


Figura 21: Arquitectura de desarrollo

Sin embargo, como también se puede aplicar para máquinas de escritorio o un entorno mixto, con equipos virtualizados y reales, se realizaron pruebas como se muestra en el siguiente esquema.

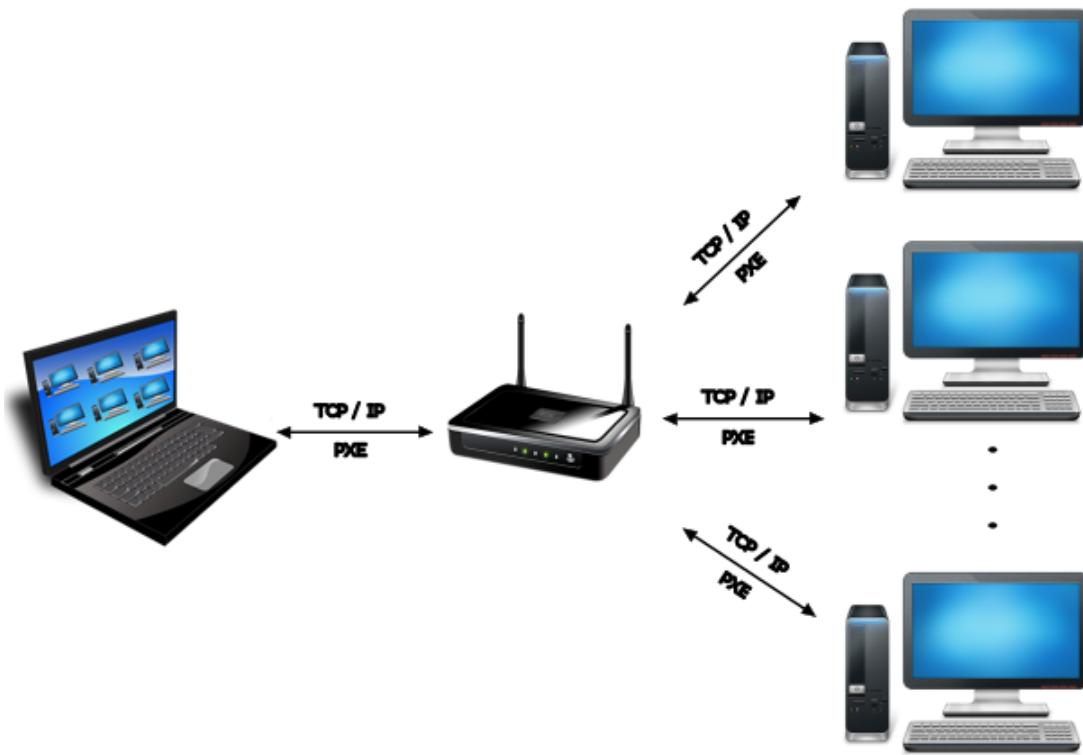


Figura 22: Arquitectura de desarrollo en un entorno mixto.

9. KVM/Qemu

9.1. Arquitectura

En este Proyecto se utiliza la técnica de virtualización completa. En particular, la arquitectura que utiliza KVM es la siguiente:

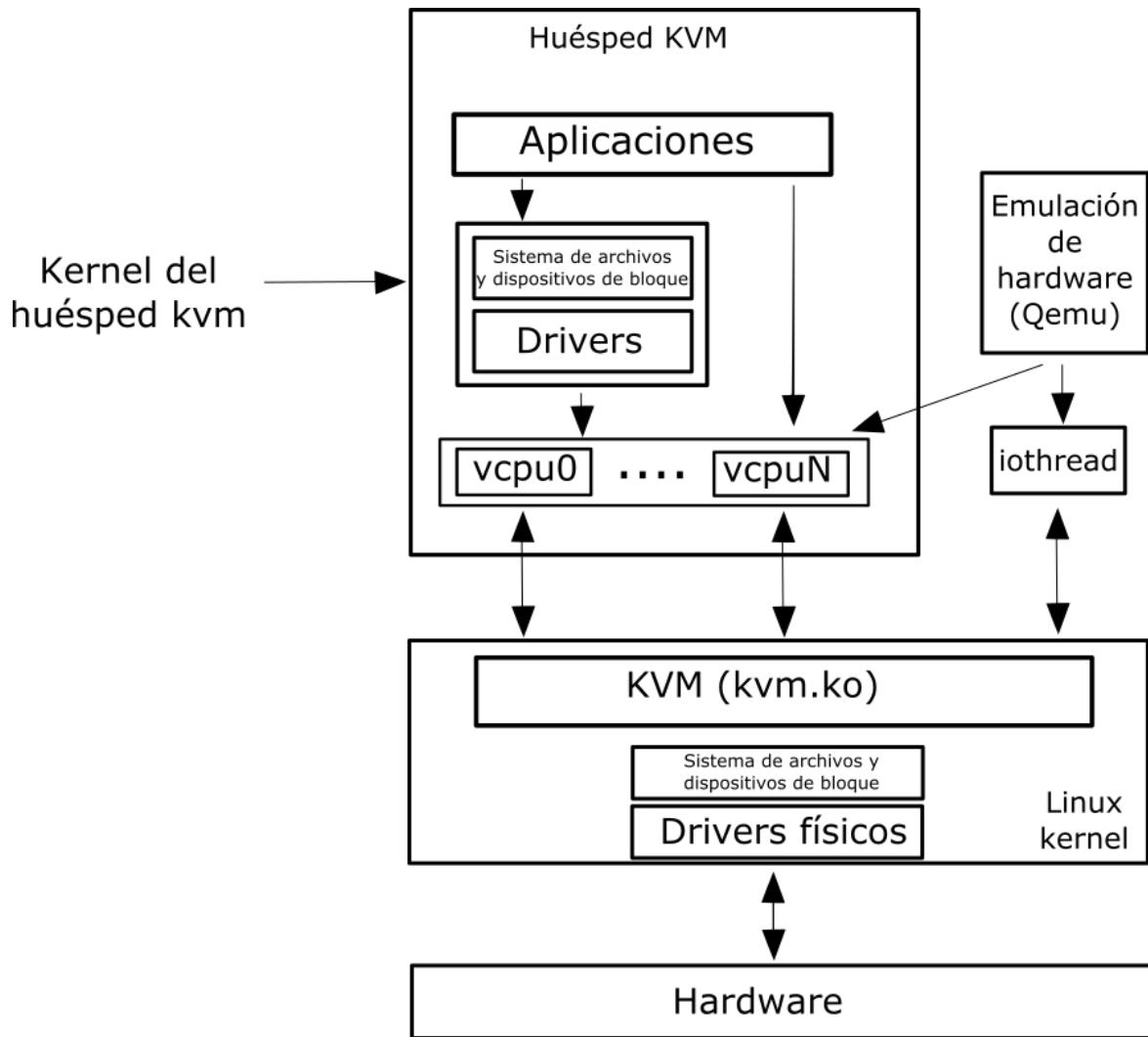


Figura 23: Diagrama de arquitectura de KVM/Qemu

9.2. Requerimientos de hardware

El hipervisor KVM requiere que el microprocesador cuente con VT-x para procesadores de Intel o con AMD -V para los propios de AMD. Para poder confirmar que un procesador cuenta con esto, en los sistemas basados en Linux, se debe ejecutar el siguiente comando:

```
grep -E 'svm|vmx' /proc/cpuinfo
```

La salida de este comando es una porción del archivo `/proc/cpuinfo` en el cual se detallan las diferentes flags que contiene el procesador, entre ellas, la `svm` (AMD) o `vmx` (Intel). En caso de no poseer esas flags, el procesador no soporta hiper-virtualización y la salida será vacía.

La siguiente, es la salida obtenida con un AMD Athlon(tm) II P360 Dual-Core Processor de 1,7GHz:

```
flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm 3dnowext
3dnow constant_tsc rep_good nopl nonstop_tsc extd_apicid pni monitor cx16 popcnt
lahf_lm cmp_legacy svm extapic cr8_legacy abm sse4a 3dnowprefetch osvw ibs
skinit wdt nodeid_msrs hw_pstate npt lbrv svm_lock nrip_save
```

Se debe asegurar que el módulo de KVM este cargado, para esto ejecutamos :

```
lsmod | grep kvm
```

La salida obtenida, nuevamente en la misma máquina que en el caso anterior es:

```
kvm_amd 60554 0 kvm 448375 1 kvm_amd
```

En caso de no estar cargados los módulos, se deben cargar manualmente, de la siguiente manera.

```
modprobe kvm_amd
```

9.3. Limitaciones de KVM

- El número máximo de CPUs por huésped es elevado (240 para RHE 7.1) por lo que no aplica en este trabajo.
- La virtualización anidada no está soportada.
- Sobre utilización de memoria es soportada por KVM utilizando el disco de swap.
- Sobre utilización de CPUs es soportada por KVM, se recomienda no utilizar más de diez CPUs virtuales por cada CPU física.
- Virtualización de dispositivos SCSI no está soportada. Virtualización de dispositivos IDE en KVM es limitada a cuatro por huésped.
- Soporta 32 slots para dispositivos PCI (paravirtualizados) y 8 de estos por cada slot (datos RHE7)
- La asignación de dispositivos referenciados a dispositivos físicos son de uso exclusivo a la VM .
- La migración y salvado, o restauración de la VM no está soportada mientras el dispositivo esté en uso.
- KVM no soporta kernels de real time.

9.4. Configuración de la red

KVM soporta las siguientes configuraciones de red para la virtualización:

- Redes virtuales usando NAT (Network Address Translation)
- Dispositivos físicos distribuidos usando la asignación de dispositivos PCI
- Redes puenteadas (bridge)

9.5. Creacion del sistema de virtualizacion qemu-KVM

9.5.1. Instalación de paquetes

Se ejecutan los siguientes comandos, el primero actualiza los paquetes actuales del sistema a la ultima version, mientras que el segundo instala los paquetes seleccionados, estos paquetes entre otras cosas proveen la herramienta qemu-KVM, una interfaz grafica como es el virt-manager para administrar y crear las maquinas virtuales, y un conjunto de comandos donde se destaca el comando virt-install que es el utilizado para crear las maquinas virtuales.

```
yum update
```

```
yum install -y kvm libvirt qemu-kvm virt-manager libvirt qemu-system-x86 qemu-img  
libvirt-python libvirt-client virt-install virt-viewer
```

Se añade el usuario que va a utilizar KVM al grupo KVM, en este caso el usuario es admin.

```
usermod -G kvm -a admin
```

9.5.2. Creacion de una red NAT

KVM soparta tres tipos de configuraciones de redes

- Redes virtuales usando NAT (Network Address Translation)
- Dispositivos físicos distribuidos usando la asignación de dispositivos PCI
- Redes puenteadas (bridge)

Se utiliza una red virtual NAT dado que (porque carajo usamos esta red????)

Aprovechamos la interfaz grafica del virt-manager para crear la red. Primero nos dirigimos a Editar>Detalles de la conexión. Se abre el siguiente menu.

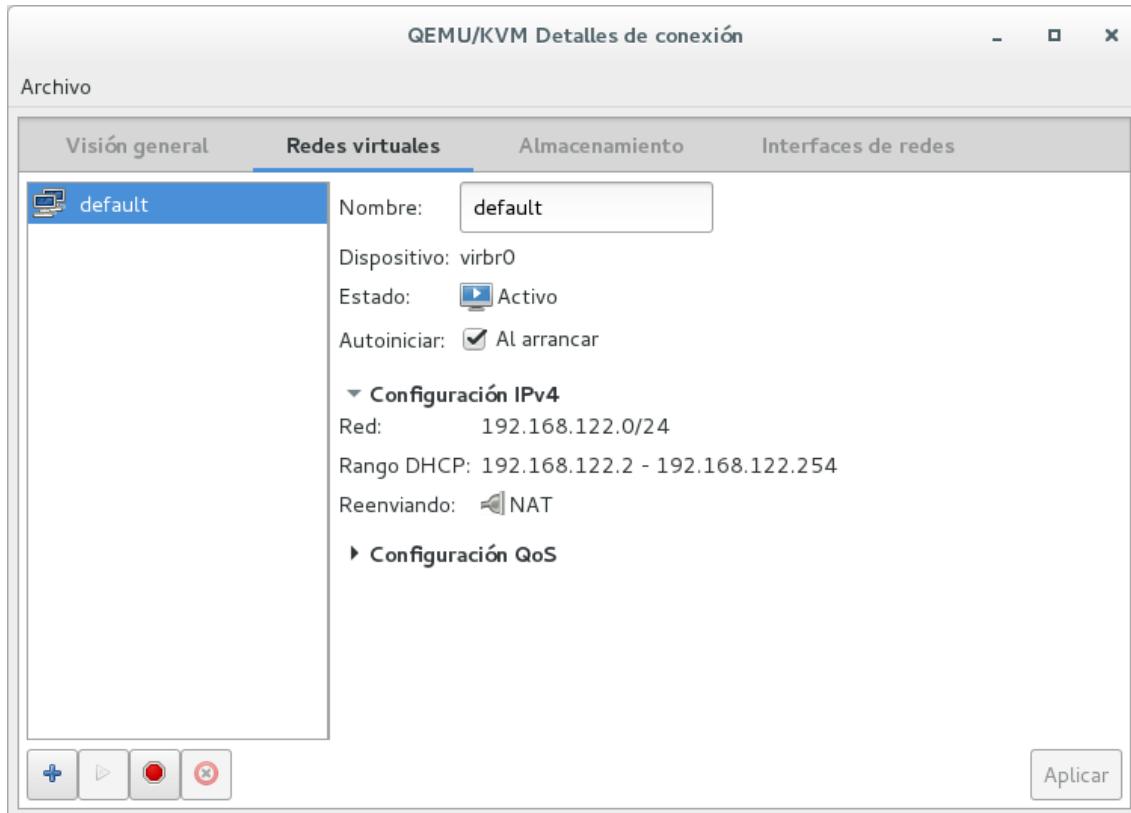


Figura 24: Detalles de conexión

Se añade una nueva red, clickeando en el signo mas.

La ventana que se abre, cuenta con 4 etapas. En la primera se selecciona el nombre de la red, puppet. Luego indicamos la red deseada (En este proyecto se uso 192.168.122.0/24) y se deshabilita el DHCP dado que utilizaremos un servidor para esto. En la tercer etapa esta la opcion de habilitar IPV6 y finalmente en la ultima etapa se indica que la red debe ser NAT y se elige el dispositivo al cual se reenvia.

9.5.3. Creacion de VMs

Las maquinas virtuales se pueden crear tanto sieguiendo al GUI como por linea de comandos, este ultimo tiene la ventaja de poder ser utilizada para scripting.

Entre la gran cantidad opciones del comando `virt-install` se destacan:

- **-connect:** Con esta opcion se indica que se trabajara con el emulador qemu
- **-virt-type:** Con esta opcion se indica que la virtualizacion la realizara KVM
- **-name:** Seleccionamos un nombre unico para la VM. Este nombre no es del host sino es el nombre para identificarlo dentro del campo de qemu-KVM
- **-ram** Indicamos la cantidad de memoria ram en MB que dispondra la VM
- **-disk:** Hubicamos el path absoluto donde se encontrara la imagen creada e indicamos el tamaño en GB de la misma.

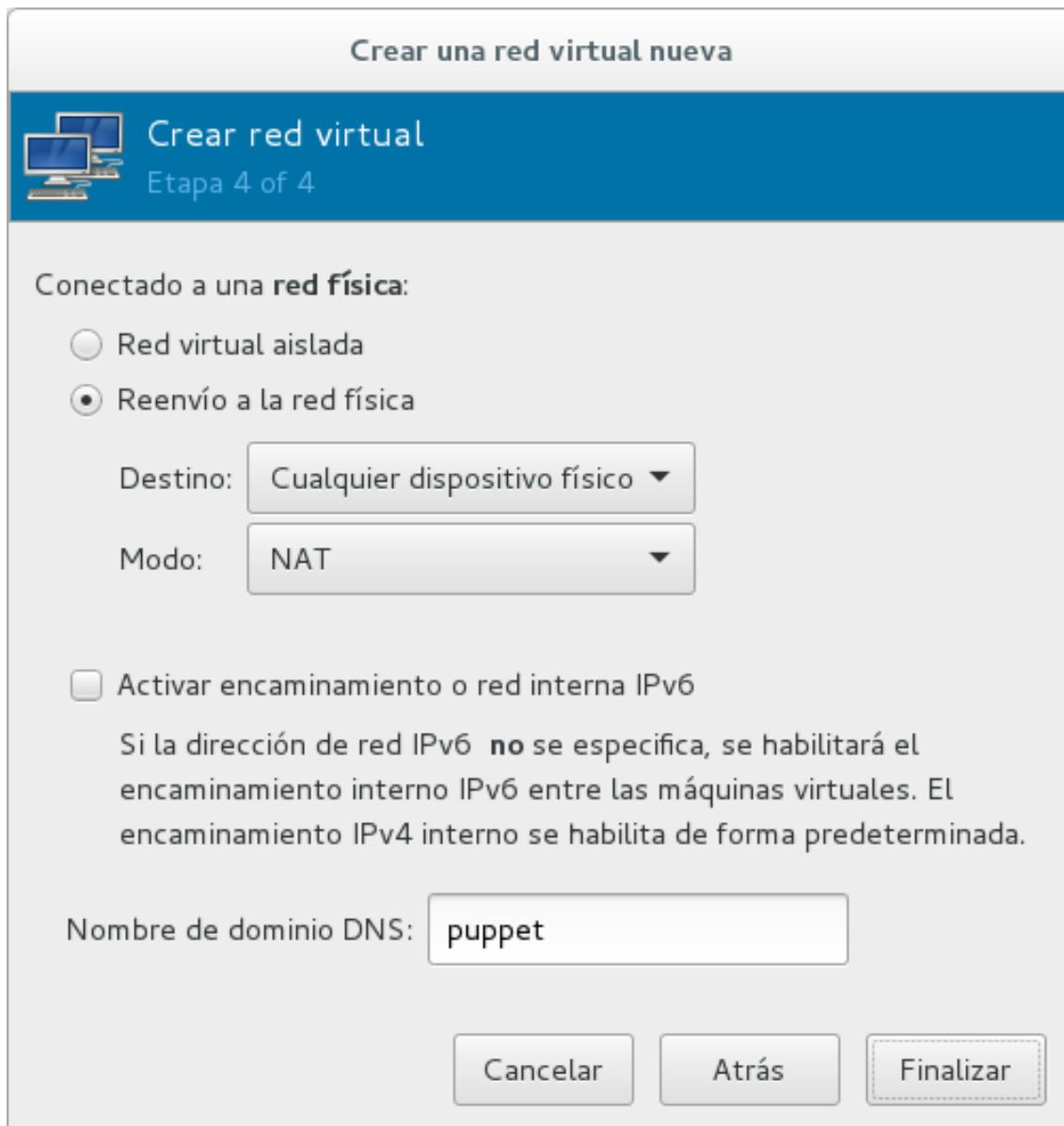


Figura 25: Crear red virtual

- **-network:** Si se trabaja con una red virtual, se le indica a la VM a cual debe conectarse
- **-pxe:** Esta opcion es muy importante dado que indica que el SO a instalar se obtiene via red.
- **-os-type y -os-variant:** Para aumentar el rendimiento se le indica a la VM que tipo de SO contendra (Linux, Windows, etc) y la variante del mismo (Entre los Linux puede ser: Centos, Debian, Ubuntu, etc.)

Un ejemplo utilizado en el desarollo del proyecto es:

```
virt-install --connect qemu:///system --virt-type kvm --name centos1025 --ram 1024 --disk path=/var/lib/libvirt/images/centos1025.qcow2,size=15 --network network=puppe --pxe --os-type linux --os-variant rhel7
```

10. Cobbler

El servidor cobbler debe contener los paquetes necesarios para poder instalar cada SO (CentOS, Ubuntu y Windows) Por lo cual debe contar con un espacio en disco minimo para soportar esto. Ademas, dado que el mismo servidor contiene el sistema de automatizacion y orquestacion, se estima que debe contar con un minimo de 30GB de disco.

El proyecto se desarollo sin tener en cuenta SELinux ni Firewall, por lo cual ambos son desabilitados de la siguiente forma.

Editar el archivo /etc/sysconfig/selinux y setear:

```
SELINUX=disabled
```

En el caso del firewall ejecutar:

```
systemctl stop firewalld.service  
systemctl disabled firewalld.service  
systemctl mask firewalld.service  
systemctl status firewalld.service
```

10.1. Tópicos generales de Cobbler

10.1.1. Modelado

Cobbler utiliza objetos para definir la configuración de aprovisionamiento. A medida que se desciende por el árbol de objetos, las variables se sobre escriben y se añaden a la información definida en los objetos superiores.

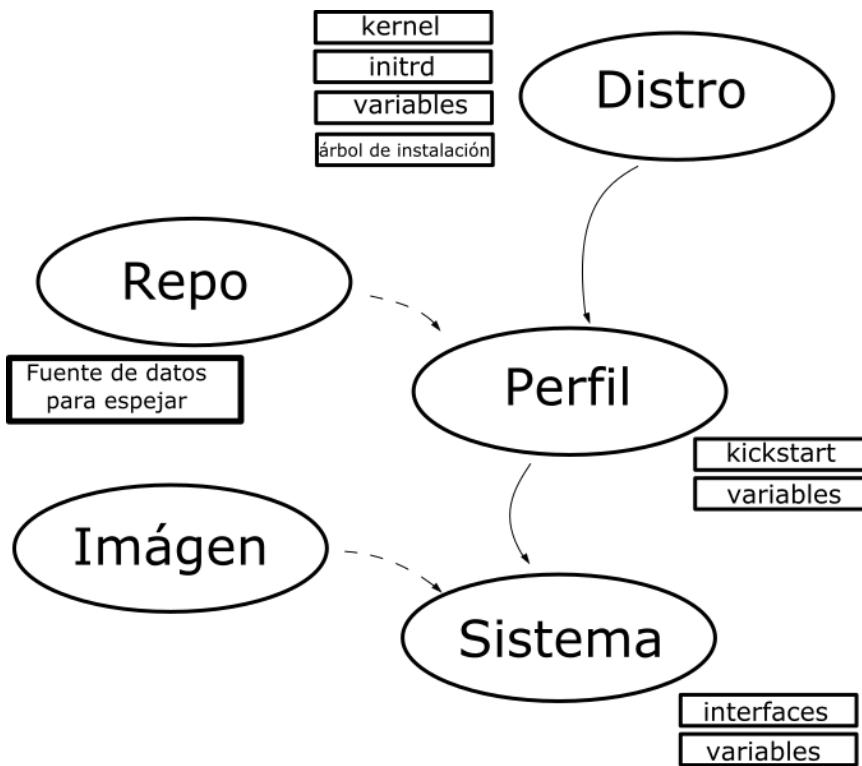


Figura 26: Modelado de Cobbler

10.1.2. Distros

Distribución que se desea instalar. Importar el contenido de la distro ayuda a disminuir el tiempo de instalación ya que no se utilizan fuentes de instalación externas. Generalmente es más fácil utilizar el comando `import` en vez de añadir la distribución manualmente.

10.1.3. Profiles

Un perfil o profile asocia una distribución a opciones especializadas adicionales, como puede ser un kickstart. Los perfiles son el núcleo del aprovisionamiento y debe existir al menos uno por cada distribución. Un perfil puede representar, por ejemplo, una configuración de web server o de escritorio.

10.1.4. Systems

Los grabaciones de sistemas mapean una pieza de hardware (o una máquina virtual) con el profile asignado a correr en ella. Esto puede verse como una forma de asignarle un rol a un sistema específico. Cuando se aprovisiona vía koan y PXE, no es necesario crearlos ya que son útiles cuando una personalización de un sistema específico es necesaria. Por ejemplo, personalizar la MAC, si hay un rol específico para una máquina dada, se debería crear una grabación del sistema para ésta.

10.1.5. Images

Cobbler puede bootear imágenes físicamente o virtualmente. Los despliegues de máquinas no basadas en imágenes son generalmente más fáciles para trabajar y llevan a una infraestructura más sustentable. La mayoría de las instalaciones de cobbler están directamente basadas en la distribución (kernel + initrd). La siguiente página documenta algunas cosas que no están basadas en kernel + initrd y muestra como instalarlas con cobbler y koan. Por ejemplo, trata la instalación de sistemas operativos Windows usando qemu/KVM:

<https://fedorahosted.org/cobbler/wiki/AllAboutImages>

<https://fedorahosted.org/cobbler/wiki/KoanWithIsos>

10.1.6. Repositorios

Espejar repositorios le permite a Cobbler espejar el árbol de instalación (`cobbler import`) y también paquetesopcionales. Si se espeja todo esto localmente en la red, las instalaciones y actualizaciones serán más rápidas (usualmente es válido realizar esto para largos setups en datacenters, laboratorios, etc). Si un profile tiene un repositorio dado, este repositorio puede ser automáticamente configurado durante el aprovisionamiento y los sistemas instalados podrán usarlo como espejo (`yum_post_install_mirror` debe estar habilitado). Si se especifica una lista de paquetes para `-rpm-list`, se puede espejar solo esa parte del repositorio, más sus dependencias. Por ejemplo, si se espeja FC6 Extras, para descargar Cobbler y Koan, ponemos `-rpm-list="cobbler koan"` y se saltea la parte de los paquetes de juegos. Esta función sólo funciona para repositorios http o ftp.

Los repositorios pueden ser creados del siguiente modo:

```
cobbler repo add --mirror=url --name=string [--rpmlist=list]
[--createrepo-flags=string] [--keep-updated=Y/N] [--priority=number] [--arch=string]
[--mirror-locally=Y/N] [--breed=yum|rsync|rhn]
```

Donde

mirror: Es la dirección del espejo yum. Puede ser una URL rsync://, una ubicación ssh o una ubicación http:// o ftp:// de un espejo. Direcciones del filesystem también funcionan. Esta dirección debe especificar un repositorio exacto a espejar, solo una arquitectura y una distribución.

name: Este nombre es el usado para guardar la ubicación del espejo.

rpmlist: Con esta opción se puede decidir espejar solo una parte de un repositorio (la lista de paquetes dados más dependencias). Por ejemplo : `--rpmlist="paquete_1 paquete_2"`. Esta opción sólo funciona con repositorios http:// y ftp:// para espejos de otros tipos esta opción será ignorada.

createrepo-flags: Especifica banderas opcionales para añadir a la herramienta `createrepo` la cual es llamada cuando se ejecuta `cobbler reposync` para el repositorio dado. Por defecto se tiene '`-c cache`'.

keep-updated: Especifica si el repositorio debería ser o no actualizado durante una ejecución normal de `cobbler reposync`. El repositorio puede seguir siendo actualizado por el nombre.

mirror-locally: Cuando se configura a N, especifica que este repositorio yum se utiliza para ser referenciado directamente por kickstarts y no para ser espejado localmente en el servidor cobbler. Solo espejos con URLs http:// y ftp:// son soportados cuando se utiliza --mirror-locally=N, no se puede usar URLs del filesystem.

priority: Especifica la prioridad del repositorio (menor número, mayor prioridad) que se aplica a máquinas instaladas usando los repositorios que tienen el plugin yum priorities instalado. Por defecto se tiene 99.

arch: Especifica la arquitectura que el repositorio debeía utilizar. Por defect se utiliza la arquitectura del servidor cobbler.

breed: Usualmente cobbler comprenderá este parámetro si no se entrega.

Para crear un repositorio local, por ejemplo para instalar Puppet en una instalación desde cero, y sin una conexión a Internet, primero es necesario tener los paquetes necesarios y sus dependencias, para ello se ejecuta:

```
sudo yum install --downloadonly --downloaddir=<directory> <package>
```

Donde se debe reemplazar <directory> por el directorio donde se descargará el paquete con sus dependencias y <package> por el puppet.

Una vez obtenidos, crear una carpeta con el nombre del repositorio en /var/www/cobbler/repo_mirror por ejemplo:

```
sudo mkdir /var/www/cobbler/repo_mirror/puppet
```

Luego es necesario añadirlo al servidor:

```
cobbler repo add --name=puppet --keep-updated=N --arch=x86_64 --mirror-locally=Y  
--breed=yum
```

Donde --name debe ser el mismo que el de la carpeta creada anteriormente. Acto seguido ejecutar:

```
createrepo /var/www/cobbler/repo_mirror/puppet
```

```
cobbler reposync
```

Para añadir este nuevo repositorio a un profile de instalación existente:

```
cobbler profile edit --name=centos7 --repos=puppet
```

Se muestra la información acerca del mismo con:

```
cobbler repo report --name=puppet
```

10.1.7. Import

El propósito de “cobbler import” es configurar un servidor de instalación por red para una o más distribuciones. Éste espeja contenido basado en un imágen DVD, un archivo ISO, un árbol en un filesystem, un espejo externo rsync o una ubicación SSH.

```
$ cobbler import --path=/path/to/distro -name=F12
```

Este ejemplo muestra los dos argumentos requeridos para import: --path y -name.

Luego de que import es ejecutado, cobbler tratará de detectar el tipo de distribución y automáticamente asignar kickstarts. Por defecto, proveerá el sistema borrando el disco duro, configurando eth0 para DHCP y utilizando la contraseña por defecto “cobbler”. Si esto no es deseado, editar los archivos kickstart en /var/lib/cobbler/kickstarts para hacer algo distinto o cambiar la configuración del kickstart después que cobbler cree el profile. El contenido espejado es guardado automáticamente en /var/www/cobbler/ks_mirror.

Ejemplos:

1. `cobbler import --path=rsync://mirrorserver.example.com/path/ --name=fedora --arch=x86`
2. `cobbler import --path=root@192.168.1.10:/stuff --name=bar`
3. `cobbler import --path=/mnt/dvd --name=baz --arch=x86_64`
4. `cobbler import --path=/path/to/stuff -name=glorp`
5. `cobbler import --path=/path/where/filer/is/mounted --name=anyname \ --available-`

Una vez importado, ejecutar “`cobbler list`” o “`cobbler report`” para ver que se ha añadido. Si se quiere forzar la utilización de una plantilla kickstart de cobbler para todos los profiles creados por un import, se puede pasar la opción `-kickstart` a import para saltar la auto detección del kickstart.

10.1.8. Kickstarts

Los kickstarts son archivos que indican cómo debe ser configurado el sistema operativo, el archivo contiene palabras claves, valores y en otros casos solo contienen la palabra clave que en sí misma es una configuración específica.

Algunas palabras clave (keywords) son opcionales, mientras que otras son necesarias para la instalación.

Keywords

- **autopart (optional)** : Creación automática de particiones, 1 GB o más para el directorio raíz (/), una partición de intercambio y una partición de arranque apropiada para la arquitectura . Uno o más de los tamaños de las particiones por defecto puede ser redefinido con la zona de directivas.
- **ignoredisk (optional)** : Hace que el instalador ignore los discos especificados.

La sintaxis es:

```
ignoredisk -drives=drive1,drive2,...
```

- **auth or authconfig (required)** :Establece las opciones de autenticación para el sistema. Es similar al comando authconfig , que se puede ejecutar después de la instalación . Por defecto, las contraseñas son encriptadas y no utilizan shadow .
- **bootloader (required)**: Especifica cómo se debe instalar el gestor de arranque.
- **clearpart (optional)** : Elimina las particiones del sistema, antes de la creación de nuevas particiones . Por defecto no se eliminan las particiones .

- **cmdline (optional)** : Realiza la instalación en un modo de línea de comandos completamente no interactivo. Cualquier solicitud por interacciones detendrá la instalación.

- **device (optional)** : El comando de dispositivo , indica al programa de instalación para instalar módulos adicionales , es en este formato :

```
device <type><moduleName> -opts=<options>
```

- **driverdisk (optional)** : Disquetes de controladores se pueden usar durante instalaciones kickstart.

- **firewall (optional)** : Esta opción corresponde a la pantalla de configuración de firewall en el programa de instalación.

- **firstboot (optional)** : Determinar si el agente de configuración se inicia la primera vez que se arranca el sistema . Si se activa, el paquete firstboot debe estar instalado. Si no se especifica, esta opción está desactivada por defecto.

- **halt (optional)** : Detiene el sistema después de que la instalación se ha completado con éxito . Esto es similar a una instalación manual, en donde Aanaconda muestra un mensaje y espera a que el usuario presione una tecla antes de reiniciar. Durante una instalación Kickstart, si no se especifica el método de terminación, la opción reboot se utiliza como predeterminado.

- **graphical (optional)** : Realice la instalación kickstart en modo gráfico . Este es el valor predeterminado .

- **install (optional)** : Le dice al sistema para instalar un sistema nuevo en lugar de actualizar un sistema existente. Este es el modo por defecto.

- **ignore disk (optional)** : Se utiliza para especificar los discos que Aanaconda no debe tocar durante la partición , el formato, y la limpieza . Este comando tiene un único argumento necesario , que toma una lista separada por comas de nombres de unidad de ignorar.

```
ignoredisk -drives=[disk1,disk2,...]
```

- **interactive (optional)** : Utiliza la información proporcionada en el archivo kickstart durante la instalación, pero permite la inspección y modificación de los valores dados . Se le presentará con cada pantalla del programa de instalación con los valores del archivo kickstart . Puede aceptar los valores haciendo clic en Siguiente o cambiar los valores y haga clic en Siguiente para continuar.

- **key (optional)**: Especifique una clave de instalación, que es necesaria para ayudar en la selección de paquetes e identificar su sistema con fines de apoyo. Este comando es Red Hat Enterprise Linux específico.

- **keyboard (required)** : Establece el tipo de teclado.

- **lang (required)** : Establece el idioma que desea utilizar durante la instalación y el idioma predeterminado para utilizar en el sistema instalado.

- **logvol (optional)** : Crea un Logical Voume con la sintaxis:

```
logvol <mntpoint> -vgname=<name> -size=<size> -name=<name><options>
```

- **logging (optional)** : Este comando controla el registro de errores de Aanaconda durante la instalación. No tiene ningún efecto en el sistema instalado.

- **monitor (optional)** : Si no se da el comando monitor, Aanaconda utilizará X para detectar automáticamente la configuración del monitor.
- **network (optional)** : Configura la información de red para el sistema. Si la instalación no requiere redes y la información de la red no se proporciona en el archivo kickstart, el programa de instalación asume que la instalación debe hacerse sobre eth0 a través de una dirección IP dinámica (BOOTP / DHCP), y configura el sistema final, instalado para determinar su dirección IP de forma dinámica.
- **part or partition (required for installs, ignored for upgrades)** : Crea una partición en el sistema.
- **poweroff (optional)** : Apaga el sistema luego de que la instalación se complete exitosamente.
- **raid (optional)** : Monta un sistema RAID.
- **reboot (optional)** : Reinicia el sistema después de una instalación exitosa.
- **repo (optional)** : Configura un repositorio adicional YUM que puede ser utilizado como fuente para la instalación de paquetes.
- **rootpw (required)** : Establece la contraseña de root.
- **selinux (optional)** : Establece el estado del SELinux en el sistema instalado.
- **services (optional)** : Modifica el conjunto predeterminado de servicios que se ejecutarán bajo el nivel de ejecución predeterminado.
- **shutdown (optional)** : Apaga el sistema después de una instalación exitosa.
- **text (optional)** : Realiza la instalación kickstart en modo texto. Las instalaciones Kickstart se ejecutan en modo gráfico por defecto.
- **timezone (required)** : Selecciona la zona horaria del sistema.
- **upgrade (optional)** : Indica que se realiza una actualización del sistema instalado.
- **user (optional)** : Crea usuario en el sistema.
- **vnc (optional)** : Permite que la instalación gráfica pueda ser vista de forma remota a través de VNC.
- **volgroup (optional)** : Crea logical volume group con la sintaxis:
`volgroup <name><partition><options>`
- **zerombr (optional)** : Si se especifica zerombr, y si es su único argumento, cualquier tabla de partición no válidas que se encuentran en los discos son inicializadas. Esto destruye todos los contenidos de discos con tablas de partición inválidas.

10.1.9. Snippets

Los snippets son una forma de reutilizar bloques de código entre kickstarts (también funcionan en otros tipos de archivos). Esto quiere decir que cada vez que el texto SNIPPET aparezca en un archivo kickstart será reemplazado por los contenidos en el archivo correspondiente dentro de `/var/lib/cobbler/snippets/`. Esto permite la reutilización de código en cada plantilla, aliviando también la lectura de las mismas.

Para utilizar un snippet, es necesario crear un archivo en el directorio `/var/lib/cobbler/snippets/nuevo_snippet` y, en un archivo kickstart, al momento de llamar a esta porción de código se utiliza:

```
$SNIPPET('nuevo_snippet')
```

Los snippets pueden ser guardados en subdirectorios para una mejor organización. El orden de precedencia será como sigue:

```
/var/lib/cobbler/snippets/$subdirectorio/$nombre_del_snippet
```

Para referenciarlo desde el archivo kickstart, ahora se tiene:

```
$SNIPPET('direcorio/nuevo_snippet')
```

Cobbler no reconoce caracteres que no estén en el alfabeto inglés, por este motivo se recomienda no utilizar caracteres especiales como `ñ` u otros que lleven tilde.

10.1.10. Integración con Puppet

Este ejemplo es relativamente avanzado, involucrando “`mgmt-classes`” de Cobbler para controlar diferentes tipos de configuración inicial. Pero si en cambio se opta por poner la mayor parte de la configuración inicial en Puppet en vez de aquí, entonces podría ser más simple.

Manter class mappings en cobbler Primero se debe asignar “`management classes`” a la distro, profile o system.

```
cobbler distro edit --name=distro1 --mgmt-classes="distro1"
cobbler profile add --name=webserver --distro=distro1 --mgmt-classes="webserver
likes_llamas" --kickstart=/etc/cobbler/my.ks
cobbler system edit --name=system --profile=webserver --mgmt-classes="orange"
--dns-name=system.example.org
```

Para Puppet el `--dns-name` (mostrado arriba) debe estar configurado porque esto es lo que Puppet estará enviando a Cobbler y es como encontrará el sistema. Puppet no tiene conocimiento sobre el nombre del sistema objeto en Cobbler. Para hacerlo de forma segura, probablemente se utilice FQDN aquí (lo cual es lo que se quiere si se utiliza cobbler para administrar DNS).

External Nodes Cobbler provee uno, así configura Puppet para usar `/usr/bin/cobbler-ext-nodes`:

```
[main]
external_nodes = /usr/bin/cobbler-ext-nodes
```

y también añadir lo siguiente al archivo de configuración:

```
node_terminus = exec
```

Ésto es un script simple que toma el información en la siguiente URL, la cual es una URL que siempre retorna un documento YAML en la forma que Puppet espera que sea retornado. Este archivo contiene todos los parámetros y clases que están para ser asignadas en el nodo en cuestión. Esta URL de Cobbler es: <http://cobbler/cblr/svc/op/puppet/hostname/foo> y esto retornará datos como:

```

--- classes:
  - distro1
  - webserver
  - likes_llamas
  - orange

parameters:
  tree:  'http://.../x86_64/tree'

```

Estos parámetros vienen de todo lo que Cobbler monitorea en “`--ks-meta`” (también es un parámetro). De este modo se puede fácilmente añadir parámetros como añadir clases y mantener todo organizado en un lugar. En caso de tener parámetros o clases globales para añadir, esto se puede hacer editando los siguientes campos en `/etc/cobbler/settings`:

```

mgmt_classes:  [ ]
mgmt_parameters:
  from_cobbler:  1

```

10.1.11. Replicate

Este comando descarga la configuración de un servidor Cobbler a otro. Sirve para tener implementaciones de High Availability, recuperación de desastres o para balanceo de carga.

```
cobbler replicate --master=master.example.org
```

Con los argumentos por defecto, solo la metadata de la distribución y del perfil es sincronizada. A continuación se muestra los argumentos que se le pueden pasar a Cobbler para que replique:

```

# cobbler replicate --help
Usage: cobbler [options]
Options: -h, --help show this help message and exit
--master=MASTER Cobbler server to replicate from.
--distros=PATTERN pattern of distros to replicate
--profiles=PATTERN pattern of profiles to replicate
--systems=PATTERN pattern of systems to replicate
--repos=PATTERN pattern of repos to replicate
--image=PATTERN pattern of images to replicate
--omit-data do not rsync data
--prune remove objects (of all types) not found on the master

```

Setup En cada servidor que será la réplica del master, instalar Cobbler normalmente y asegurarse que `/etc/cobbler/settings` y `/etc/cobbler/modules.conf` están configurados apropiadamente. Utilizar cobbler check para ver si existe algún error. El comando no modificará estos archivos.

Los archivos son transferidos por rsync (sobre ssh) o por scp, por lo que es necesario tener un agente ssh antes de utilizar el comando de réplica o si no, utilizar authorized_keys en el host remoto.

10.2. Creacion del sistema de aprovisionamiento y automatizacion Cobbler

10.2.1. Instalación

Primero y principal, Cobbler necesita Python, alguna versión superior a la 2.6. Además, requiere de un servidor DHCP, FTP, HTTP y Rsync y una serie de paquetes.

- createrepo
- httpd
- mkisofs mod_wsgi
- mod_ssl
- python-cheetah
- python-netaddr
- python-simplejson
- python-urlgrabber
- PyYAML
- rsync
- syslinux
- tftp-server
- yum-utils

La interfaz web de cobbler requiere Django.

Se añade el repositorio Epel de CentOS 7

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm  
rpm -Uvh epel-release-*
```

Se instalan los paquetes necesarios para cobbler, el servidor DHCP, FTP, HTTP, etc.

```
yum install cobbler cobbler-web dhcp pykickstart system-config-kickstart tftp  
httpd xinetd fence-agents-all -y
```

10.2.2. Configuración de servicios

Se deben configurar los servicios que requiere cobbler (DHCP, FTP, HTTP, etc)

Configurar FTP y activacion de RSYNCD

Editar /etc/xinetd.d/tftp modificando disable = yes por no. Luego ejecutar:

```
systemctl start rsyncd  
systemctl enable rsyncd
```

Configurar DHCP

Copiar el archivo de configuración de ejemplo:

```
cp /usr/share/doc/dhcp-4.2.5/dhcpd.conf.example /etc/dhcp/dhcpd.conf
```

Verificar en cada caso la versión de dhcp.

Luego editar /etc/dhcp/dhcpd.conf, la configuracion utilizada en este trabajo se adecua la red “puppet” creada.

```
# A slightly different configuration for an internal subnet.  
subnet 192.168.122.0 netmask 255.255.255.0 {  
    range 192.168.122.1 192.168.100.254;  
    option domain-name-servers puppet;  
    option domain-name "localdomain";  
    option routers 192.168.122.1;  
    option broadcast-address 192.168.122.255;  
    default-lease-time 600;  
    max-lease-time 7200; }
```

De la misma forma se configura el archivo /etc/cobbler/dhcp.template.

```
subnet 192.168.122.0 netmask 255.255.255.0 {  
    option routers 192.168.122.1;  
    option domain-name-servers 192.168.122.1;  
    option subnet-mask 255.255.255.0;  
    range dynamic-bootp 192.168.122.1 192.168.122.254;  
    default-lease-time 21600;  
    max-lease-time 43200;  
    next-server $next-server;
```

10.2 Creacion del sistema de aprovisionamiento y automatizacion Cobbler COBBLER

```
class "pxeclients" { match if substring(option vendor-class-identifier, 0, 9) = "PXEClient";  
if option pxe-system-type = 00:02 {  
filename "ia64/elilo.EFI";  
}  
} else if option pxe-system-type = 00:06 {  
filename "grub/grub-x86.EFI"; }  
else if option pxe-system-type = 00:07 {  
filename "grub/grub-x86_64.EFI"; }  
else { filename "pxelinux.0";  
}  
}
```

Añadir a /etc/hosts la dirección del servidor y su hostname (puppet). Luego configurar el parámetro ServerName en /etc/httpd/conf/httpd.conf con el nombre del host, en este caso será puppet.

ServerName puppet

Los servicios ya estan listos para funcionar, se los inicia e indica que se inicien automaticamente al prender la maquina

```
systemctl start httpd.service  
systemctl start dhcpcd.service  
systemctl start xinetd.service  
systemctl start cobblerd.service  
  
systemctl enable httpd.service  
systemctl enable dhcpcd.service  
systemctl enable xinetd.service  
systemctl enable cobblerd.service
```

Configuramos el servidor de cobbler, para ello se edito el archivo /etc/cobbler/settings.

La clave de los usuarios root de las VMs por defecto se decidió que sea “qwerty”. Se encripta esta clave con el comando

```
openssl passwd -1
```

Lo que dará algo similar a:

```
Password: Verifying - Password: $1$U.Svb2gw$MNHrAmG.axVHYQaQRySR5/
```

10.2 Creacion del sistema de aprovisionamiento y automatizacion Cobbler COBBLER

y se coloca en la sección correspondiente del archivo de configuración de cobbler

```
default_password_crypted:  "$1$U.Svb2gw$MNHrAmG.axVHYQaQRySR5/"
```

Dado que se desea utilizar un servidor DHCP en la misma máquina que contiene el servidor Cobbler, se modificar la sección “manage_dhcp: 0” para habilitar el administrador del DHCP.

```
manage_dhcp:  1
```

Configurar ahora la dirección IP del servidor Cobbler en las variables “server” y “next_server” colocando la interfaz virtual que conecta a la máquina física con las máquinas virtuales,virbr0:

```
next_server:  192.168.122.1  
server:  192.168.122.1
```

Para habilitar la interfaz web de Cobbler y configurar usuario y contraseña, se modificaron las siguientes líneas del archivo /etc/cobbler/modules.conf para que queden de este modo:

```
[authentication] module = authn_configfile  
[authorization] module = authz_allowall
```

El usuario y la contraseña para la interfaz web que por defecto tiene tanto a usuario como contraseña a “cobbler”, se utiliza correr el siguiente comando e ingresar la contraseña preferida dos veces:

```
htdigest /etc/cobbler/users.digest "Cobbler" admin
```

En el proyecto se utilizó admin como el usuario para ingresar a la interfaz web y la clave es “qwerty”

Se edita el archivo /etc/debmirror.conf comentando las siguientes líneas:

```
#@dists="sid";  
#@arches="i386";
```

Se reinician todos los servicios para asegurarse de que todos tomen su nueva configuración

10.2 Creacion del sistema de aprovisionamiento y automatizacion Cobbler COBBLER

The screenshot shows the Cobbler Web Interface in Mozilla Firefox. The URL is https://192.168.122.1/cobbler_web/ksfile/list. The page title is "Cobbler Kickstart Templates". On the left, there is a sidebar with navigation links: Configuration (Distros, Profiles, Systems, Repos, Images, Kickstart Templates, Snippets, Management Classes, Settings), Resources (Packages, Files), Actions (Import DVD, Sync, Reposync, Hardlink, Build ISO), and Cobbler (Distro, Events, Online Documentation, Online Help Chat). The main content area displays a table of kickstart files with columns for "File" (listing paths like /var/lib/cobbler/kickstarts/Centos_GUI.ks, /var/lib/cobbler/kickstarts/alumno.v2.ks, etc.) and "Actions" (each row has an "Edit" link). At the bottom of the table, it says "Cobbler 2.6.11".

Figura 27: Cobbler Web

```
systemctl restart httpd.service  
systemctl restart dhcpcd.service  
systemctl restart xinetd.service  
systemctl restart cobblerd.service
```

El próximo paso es descargar los “network boot loaders” con el comando `cobbler get-loaders` y se sincroniza el servidor de cobbler.

```
systemctl restart cobblerd.service  
Sincronizar cobbler cobbler sync
```

10.2.3. Repositorio local separado de Cobbler

En el caso que se desee crear un repositorio local que no dependa del servidor Cobbler, se debe primero instalar los servicios necesarios para el funcionamiento del servicio de repositorios, estos servicios son `vsftpd` que asegura una conexión segura y `createrepo` que es el encargado de generar las bases de datos necesarias para el correcto funcionamiento del repositorio.

```
sudo yum install -y createrepo vsftpd lftp
```

Una vez hecho ésto, se tiene que crear el árbol de directorios adecuado de acuerdo a la aplicación. Éste puede estar dividido como se desee, por ejemplo, por sistema operativo, por distribución, por arquitectura, etc.

10.2 Creacion del sistema de aprovisionamiento y automatizacion Cobbler COBBLER

Se recomienda utilizar como base el directorio `/var/ftp/pub` dado que se utilizará este sistema de transmisión de archivos.

Una vez creado el sistema de archivos, se guardan los correspondientes paquetes rpm en los lugares adecuados, acordes a cómo se haya creado el árbol de archivos, y se ejecuta el siguiente comando:

```
createrepo -v /var/ftp/pub/nombre_repo
```

Es necesario entonces configurar el demonio vsftpd editando el archivo `/etc/vsftpd/vsftpd.conf`. Hay diversas configuraciones posibles, pero hay dos puntos importantes que deben existir:

1. `anonymous_enable=YES` #Indica que se puede acceder vía ftp de manera anónima.
2. `anon_root=/var/ftp/pub` #Indica la raíz del directorio al cual se puede acceder de manera anónima.

Por último, iniciar el servicio:

```
systemctl start vsftpd
```

En el lado del cliente es necesario informar del nuevo repositorio. Para ésto, crear un archivo en `/etc/yum.repos.d/nombredelrepo.repo` con el siguiente contenido:

```
[nombredelrepo]
name=nombredelrepo
comment ="Repositorio local para proyecto integrador"
baseurl=ftp://IP_servidor/nombre_del_sistema_de_archivos #Notar que el path
es absoluto, partiendo desde el path permitido para los usuarios anonimos.
gpgcheck=0
enabled=1
priority=1
```

Los paquetes `yum-priorities` e `yum-utils` deben estar instalados en el cliente, para poder utilizar sus funcionalidaes (dar prioridades a los repositorios y habilitarlos facilmente.)

Para utilizar este repositorio primero hay que habilitarlo:

```
yum-config-manager --enable proyectointegrador
```

Luego es necesario actualizar base de datos de repositorios:

```
yum makecache
```

Es posible utilizar sólo el repositorio recién creado y excluir los demás, para ello:

```
yum-config-manager --disable * --enable proyectointegrador
```

10.2.4. Importar imágenes ISO al servidor Cobbler

De esta forma, se puede generar una distribucion que se utilizara para instalar los SO sin necesidad de acceder a internet de una forma facil y rapida.

Se utiliza el comando mount. Primero crear un directorio y luego montar el archivo ISO:

```
mkdir /mnt/centos
```

```
mount -t iso9660 -o loop,ro /path/to/isos/CentOS-7-x86_64-DVD-1503-01.iso /mnt/centos
```

Luego ejecutar:

```
cobbler import --name=centos7 --arch=x86_64 --path=/mnt/centos
```

Ésto creará una copia local en el servidor, dando lugar a un nuevo objeto “distro” y “profile”. Los cuales se pueden verificar con:

```
cobbler distro list
```

```
cobbler profile list
```

11. Puppet

El agente y el servidor se comunican vía HTTPS con verificación de cliente. El nodo maestro (servidor) provee una interfaz HTTPS con varios extremos disponibles. Cuando se pide o envía cualquier cosa al servidor, el agente hace un pedido HTTPS o a uno de esos extremos.

Client-verified HTTPS quiere decir que cada maestro o agente tiene un identificador por certificado SSL y examinan los certificados de sus contrapartes para decidir si permite un intercambio de información. Puppet incluye un constructor de certificado de autorización para administrar los certificados. Los agentes puede pedir automáticamente los certificados vía la API HTTP del maestro. El administrador del nodo maestro puede usar el comando `puppet cert` para inspeccionar los pedidos y firmar nuevos certificados; los agentes pueden entonces descargar los certificados firmados.

11.1. Tópicos generales de Puppet

11.1.1. Module

Un módulo o module, es un conjunto de código de Puppet enpaquetado junto con los otros archivos y datos, que se necesita administrar sobre algún aspecto del sistema. Consiste en

una estructura predefinida de directorios que ayudan a Puppet a encontrar los contenidos del módulo.

Existe un repositorio público (The Puppet Forge) donde se pueden encontrar módulos hechos por la comunidad y también mantenidos por Puppet Labs. Estos modulos se pueden instalar en un servidor-puppet para utilizarlos.

Para ver los módulos instalados se puede ejecutar:

```
puppet module list
```

Los módulos son auto-contenidos y separados. Su estructura de archivo le da a Puppet una forma consistente de localizar cualquier clase, plantillas, plugins y binarios requeridos para satisfacer la funcionalidad del módulo.

Todos los módulos accesibles por el puppet master están localizados en los directorios especificados por la variable 'modulepath' en el archivo de configuración de Puppet. Para encontrar esta variable en cualquier sistema con Puppet, se puede ejecutar:

```
puppet agent --configprint modulepath
```

11.1.2. Resources

Cada recurso o resource, describe algún aspecto de un sistema y su estado, como por ejemplo, un servicio que debería estar ejecutándose o un paquete que se quiere instalado. El bloque de código que describe un recurso se llama declaración de recurso (resource declaration). Estas declaraciones de recurso están escritas en código Puppet, un DLS (Domain Specific Language) construido en Ruby. El DLS de Puppet es un lenguaje declarativo en vez de imperativo. Esto quiere decir que en vez de definir un proceso o un conjunto de comandos, el código de Puppet describe (o declara) solo el estado final deseado, y depende de proveedores integrados para lidar con la implementación.

```
puppet resource tool -> puppet resource <type> <name>
```

Puppet incluye una variedad de tipos de recursos integrados, que permiten administrar varios aspectos de un sistema. Algunos de los tipos de recursos claves que generalmente se encuentran en un sistema son los siguientes:

- **user:** Un usuario
- **group:** Un grupo de usuario
- **file:** Un archivo específico
- **package:** Un paquete de software
- **service:** Un servicio corriendo
- **cron:** Un trabajo programado de cron
- **exec:** Un comando externo
- **host:** Un host

Una declaración de recurso seguirá un patrón como el de abajo:

```
tipo {'título':  
    atributo => 'valor',  
}
```

- **Título:** Es un string que identifica un recurso para el compilador de Puppet.

El título no tiene que coincidir con lo que va a administrar en el sistema, pero a menudo se desea eso.

Los títulos deben ser únicos por tipos de recursos, se puede tener un paquete y un servicios ambos con el mismo título, pero no dos servicios con ese título.

- **Atributos:** Los atributos describen el estado deseado para un recurso; cada atributo maneja algún aspecto del recurso.

Cada tipo de recurso tiene su propio juego de atributos. Muchos tipos de recursos tienen atributos claves y una gran cantidad de opcionales.

Todos los atributos declarados deben tener un valor; el tipo de dato del valor depende de los que acepte el atributo.

- **Comportamiento:** Una declaración de recurso agrega un recurso al catálogo y le dice a Puppet que administre el estado del recurso. Cuando Puppet aplica el catálogo compilado, lo que hará es:

- Leer el estado actual del recurso en el sistema objetivo.
- Comprar el estado actual con el deseado
- Si es necesario, realizar cambios para llevar el estado actual al deseado.

- **Recursos no administrados:** Si el catálogo no contiene un recurso, implica que Puppet ya no lo administra, pero no que lo “elimina”, si se desea eliminarlo, se debe aclararlo en su estado deseado.

- `ensure => absent`

- **Singularidad:** Puppet no permite que se declare un mismo recurso dos veces. Esto prevee conflictos de valores. Si múltiples clases requieren el mismo recurso se puede usar una clase o un recurso virtual para añadirlo al catálogo en múltiples lugares sin duplicar.

- **Relaciones y orden:** Por defecto, Puppet aplica los recursos sin seguir el orden en que fueron escritos. Esto, se puede desactivar con la opción de ordenado. Sin embargo, si un recurso debe ser aplicado antes o después de otro, se puede indicar una relación entre ellos. Incluso se puede indicar que cambios en un recurso causen que otro se refresque.

- **Cambios, eventos y reportes:** Si Puppet realiza cambios, en un recurso, registra esos cambios como eventos. Esos eventos aparecerán en el *log* y en el reporte de ejecución de puppet.

- **Independencia de alcance:** Los recursos no están sujetos a los alcances. Un recurso, en cualquier ámbito, se puede referenciar desde cualquier otro ámbito.

- **Atributos especiales de los recursos.**

- **Name/Namevar:** Define un recurso en el sistema objetivo. Por ejemplo, el *name* de un servicio o paquete es el nombre por el cual las herramientas de paquetes o servicios lo reconocen o en el caso de un archivo, su namevar es el path. Esto es diferente al título, el cual identifica un recurso para el compilador de Puppet. Sin embargo, ellos a veces tienen el mismo valor. La separación de

nombre y título permite administrar un recurso que mantiene su título, pero que tiene diferente nombre en diferentes plataformas. Por ejemplo, un servicio *ntp* en sistemas Red Hat tiene por nombre *ntpd* y en sistemas debian *ntp* .

- **Ensure:** Esto generalmente maneja el aspecto más importante de un recurso en el sistema objetivo. Indica si el archivo existe, si el servicio está corriendo o parado, si el paquete está instalado, etc.

Tipos de recursos Todos los tipos tienen un atributo especial llamado namevar. Este es el atributo usado para identificar univocamente un recurso en el sistema de destino. Si no se especifica un valor para el namevar, este valor es tomado por defecto según el título del recurso.

Ejemplo:

```
file { '/etc/passwd':
  owner => root,
  group => root,
  mode => 644
}
```

En este código, */etc/passwd* es el título del recurso *file*, otros códigos de Puppet pueden hacer referencia al recurso como *File['/etc/passwd']* para declarar relación. Porque el path es el *namevar* para el tipo *file* y si no se le provee un valor, toma uno por defecto que es */etc/passwd*.

Atributos: A veces llamados parámetros, determinan el estado deseado para un recurso. Cualquiera de ellos modifica directamente el sistema (internamente, las llamadas “propiedades”) o afectan cómo el recurso se comporta.

Proveedores (providers): Implementan el mismo tipo de recursos en diferentes tipos de sistemas, ellos suelen hacer esto llamando a comandos externos. Aunque Puppet seleccionará automáticamente un proveedor apropiado por defecto, se lo puede sobreescibir con el atributo *provider*. Por ejemplo, el recurso *package* de sistemas Red Hat tiene por defecto YUM como *provider*, pero se puede especificar *provider => gem* para instalar librerías de Ruby con *gem*.

Características (features): Son habilidades que algunos proveedores pueden no soportar. Generalmente una característica corresponderá con algunos valores permitidos por un recurso de un atributo, por ejemplo, si un paquete soporta la característica *purgeable*, se puede especificar *ensure => purged* para borrar los archivos de configuración instalados por el paquete.

Algunas de las referencias de tipo más importantes son las explicadas a continuación:

- **Cron:** Instalar y manejar trabajos Cron. Todo Cron creado por Puppet requiere un comando y al menos un atributo de un periodo (horas, minutos, meses, etc). Mientras el nombre del Cron no es parte del trabajo actual, el nombre es almacenado en un comentario comenzando con *#Puppet Name:.* Ese comentario es usado para coincidir entadas crontab creadas por Puppet con un recurso Cron.

- **Exec:** Ejecuta comandos externos. Cualquier comando en un recurso Exec debe poder correr múltiples veces sin causar daños.
- **File:** El manejo de archivos incluye contenido, dueño, y permisos. El tipo archivo puede manejar archivos, directorios y enlaces simbólicos.
- **Group:** Manejo de grupos. En muchas plataformas esto sólo puede crear grupos. La membresía de los grupos debe ser administrada por cada usuario individual .
- **Host:** Instalar y manejar entradas de hosts. Para muchos sistemas, esas entradas deben estar solo en /etc/hosts, pero algunos SO tienen diferentes soluciones.
- **Mount:** Maneja de filesystems montados, incluyendo agregar la información de montaje a la tabla de montaje. El comportamiento actual depende de el valor de el parámetro *ensure*.
- **Notify:** Envío de un mensaje arbitrario a el log del agente en tiempo de ejecución.
- **Package:** Manejo de paquetes. Hay una bifurcación básica en los paquetes soportados correctamente: Algunos tipos de paquetes como yum y apt pueden recuperar sus propios archivos de paquetes, mientras que otros no pueden. Para esos paquetes, se puede usar el parametro *source* para poner el archivo adecuado.
- **Resources:** Este es un metatipo que puede controlar otro tipo de recursos. Cualquier metaparámetro especificado aquí sera pasado a los recursos generados, por lo que puede purgar recursos no administrados.
- **Service:** Controla servicios en ejecución. El soporte de este recurso varía ampliamente según el concepto de servicio de la plataforma.
- **User:** Administración de usuarios.

11.1.3. Manifiests

Un manifiesto o manifest, es un archivo de texto que contiene código Puppet y posee la extensión .pp. Para comprobar la sintaxis de un manifiesto se puede utilizar:

```
puppet parser validate <manifiesto.pp>
```

El parseador no retornará nada si no hay errores, en caso de que se detecte un error debe ser corregido antes de continuar. Si se trata de aplicar un manifiesto que no ha sido declarado, no cambiará nada en el sistema. Para ésto se debe crear un .pp que contenga un sentencia:

```
include módulo::clase
```

Antes de aplicar cambios en el sistema, se puede utilizar la bandera -noop para compilar el catálogo (catálogo) y notificar los cambios que Puppet habría realizado si hubiera sido ejecutado sin -noop.

```
puppet apply --noop
```

11.1.4. Catálogos

Los manifiestos de Puppet pueden usar lógica condicional para describir muchas configuraciones de nodos como una. Antes de configurar un nodo, Puppet compila los manifiestos en un catálogo, el cual solo es válido para un único nodo y no contiene lógica ambigua.

Los catálogos son documentos estáticos los cuales contienen recursos y relaciones.

En la arquitectura estándar maestro/agente, los nodos solicitan los catálogos al Puppet Server, el cual los compila cuando son solicitados. Los agentes mantienen en caché sus más recientes catálogos, si al pedir el catálogo, el master falla al compilarlo, ellos reusaran su catálogo cacheado.

11.1.5. Classes

Una clase es un bloque de código Puppet con nombre. Una clase administrará generalmente un conjunto de recursos relacionados a una función simple o un componente del sistema. Las clases usualmente contienen otras clases; este anidamiento provee una forma estructurada de juntar funciones de clases diferentes como componentes de soluciones más grandes. Para utilizar una clase, se necesita definirla escribiendo una definición de clase y guardándola en un archivo manifiesto. Cuando Puppet se ejecuta, parseará este manifiesto y guardará la definición de clase; luego ésta puede ser declarada para aplicarla en los nodos de la infraestructura. En Puppet las clases son singleton, lo que quiere decir que una clase puede ser declarada sólo una vez en un nodo dado. Cuando se declara una clase:

```
include módulo::clase
```

módulo le indica a Puppet donde encontrar esa clase. Sin embargo, para la clase principal de un módulo, además de llevar el mismo nombre que el módulo mismo, Puppet reconoce el nombre especial del archivo '`init.pp`' como el manifiesto que contendrá la clase principal de un módulo.

11.1.6. Funciones

Hay dos tipos de funciones en Puppet, statements (declaraciones) y rvalues. Las statements no retornan argumentos, son utilizadas para hacer trabajos independientes como importar. Rvalues retornan valores y pueden ser usadas solo en un statement requiriendo un valor, como una asignación o una declaración case.

Las funciones se ejecutan en el Puppet master, no se ejecutan en el agente. Por lo tanto sólo tienen acceso a los comandos y datos disponibles en el nodo maestro.

11.1.7. Metaparámetros

Los metaparámetros son atributos que trabajan con cualquier tipo de recurso, incluido los tipos personalizados y los tipos definidos.

En general, ellos afectan el comportamiento de Puppet en preferencia a el deseo del estado del recurso.

Los metaparámetros hacen cosas como agregar metadata a un recurso (alias, tag), poner límites cuando el recurso debe ser sincronizado (require, schedule, etc.), evita que Puppet realice cambios (noop), y cambia la verborrea del log (loglevel).

11.1.8. Definición de nodos

Una definición o declaración de nodo es un bloque de código Puppet que sólo será incluido en catálogos de nodos que coincidan. Esta característica permite asignar configuraciones específicas a nodos específicos.

Las declaraciones de nodos sólo coinciden con los nombres de los nodos. Por defecto, el nombre de un nodo es su *certname*.

```
# node 'www1.example.com' { include common include apache include squid }
node 'db1.example.com' { include common include mysql }
```

Ubicación Las definiciones de los nodos deben estar en el manifiesto principal. Éste puede ser un archivo o un directorio conteniendo muchos archivos.

```
# /etc/puppetlabs/code/environments/production/manifests/site.pp

node 'www1.example.com' {
  include common
  include apache
  include squid
}

node 'db1.example.com' {
  include common
  include mysql
}
```

En este ejemplo, solo el primero nodo obtendrá las clases *apache* y *squid* mientras que el segundo tendrá *mysql*. Ambos recibirán la clase *common*.

Nombramiento Una declaración de nodo debe realizarse según:

- Un string entre comillas conteniendo sólo letras, números, guiones bajos, guiones medios y puntos.
- Expresiones regulares.
- La palabra *default*, sin comillas.

Se pueden utilizar listas de nombres separados por comas para crear grupos de nodos son una sola declaración de nodo:

```
node 'www1.example.com', 'www2.example.com', 'www3.example.com' {
  include common
  include apache, squid
}
```

El nodo default Si ninguna declaración de nodo es macheada o no puede ser encontrada, el nodo *default* será utilizado.

Expresiones regulares Pueden ser utilizadas como nombres de nodo. Este es otro método para escribir una sola definición de nodo que coincide con múltiples nodos.

Coïncidencias Un nodo dado sólo obtendrá los contenidos de una definición de nodo, incluso si dos declaraciones de nodo pueden coincidir con el nombre del mismo. Puppet realizará las verificaciones para decidir cual definición utilizar:

1. Si hay una definición de nodo que contenga el nombre exacto del nodo, entonces utilizará esta.
2. Si hay una expresión regular que coincide con el nombre del nodo, entonces utilizará esta.
3. Si el nombre de nodo es del tipo FQDN (Fully Qualified Domain Name), Puppet cortará el grupo final y comenzará nuevamente desde el punto uno.
4. Puppet utilizará el nodo default.

11.1.9. Node group

Los grupos de nodos o node groups permiten segmentar todos los nodos de la infraestructura en grupos separados configurables basados en la información colectada por 'facter tool'.

11.2. Creacion del sistema de orquestacion Puppet.

Puppet usualmente corre bajo la arquitectura cliente-servidor, pero además, puede correr en una arquitectura autocontenido. La decisión determina que paquetes serán instalados y que configuraciones extra necesarias se harán.

Se toma la opción de utilizar la arquitectura cliente-servidor. Se debe completar la instalación y configuración de todos los puppet servers antes de instalar cualquier agente. El servidor necesariamente debe correr en un sistema basado en Unix.

11.2.1. Requerimientos de sistema y chequeo de versión de sistema operativo

- **Hardware:** El agente Puppet no tiene requerimientos particulares de hardware y corre prácticamente en cualquier computadora, sin embargo, el servidor es un recurso intensivo y debe ser instalado en un servidor robusto y dedicado. Como mínimo, el servidor debe tener dos procesadores y al menos 2GB de RAM, para administrar eficientemente mil nodos, debe poseer entre 2 y 4 procesadores y 4GB de RAM.
- **Sistemas operativos soportados:** Hay una gran variedad de distribuciones Linux que soportan puppet, entre ellas destaca la utilizada para la realización del Proyecto, CentOS 7.
- **Ruby:** Se soportan varias versiones de Ruby, pero se recomienda el uso de las versiones 2.1.x.
- **Librerías obligatorias:** Facter 2.4.3 o posterior, Hiera 2.0.0 o posterior, json gem cualquier versión moderna, rgen gem 0.6.6 o posterior.
- **Librerías opcionales:** msgpack gem es requerido si se utiliza msgpack racionalización.

11.2.2. Chequeo de la configuración de red

En un agent/master deployment se debe preparar la red para el tráfico de puppet.

- Firewall: En el desarollo de este proyecto se trabajo sin firewall, por lo tanto este debe estar desactivado.
- Resolución de nombres: Cada nodo debe tener un nombre único.

Para cumplir con este ultimo apartado, se decidió editar el archivo `/etc/hosts` de cada maquina virtual creada utilizando cobbler para añadir el host del puppetserver que corresponde a la primer dirección IP de la red virtual creada por KVM. Así mismo, como el servidor puppet tambien tiene la necesidad de conocer a que dirección IP corresponde a cada maquina, se realizo un pequeño dodigo de python que es capaz de obtener la dirección IP de una maquina en particular revisando el archivo `/var/lib/dhcpd/dhcpd.leases`. Este archivo, es utilizado por el servidor DHCP para saber a que maquina corresponde cada dirección IP.

11.2.3. Instalación de puppetserver

Primero se debe instalar puppetserver. Para ello habilitar los paquetes de los repositorios de Puppet Labs:

```
sudo rpm -ivh https://yum.puppetlabs.com/puppetlabs-release-pc1-el-7.noarch.rpm
```

```
yum install puppetserver
```

No se debe iniciar el servicio aún, dado que los no se ha configurado correctamente.

Como se menciono, puppetserver tiene ciertos requerimientos de HW donde se destaca los 2GB de memoria RAM pero esto no es exactamente así, lo que realmente sucede es que esa es la cantidad de memoria “recomendada” para que funcione correctamente un sistema en produccion y puppet mientras exista ese HW se “apropiara” de el.

Si lo que se desea es experimentar con un pequeño servidor puppet y unos pocos clientes se puede configurar para que utlice menor cantidad de memoria, esto se hace editando el archivo `/etc/sysconfig/puppetserver` y modificando la siguiente linea:

```
# Modify this if you'd like to change the memory allocation, enable JMX, etc
```

```
JAVA_ARGS="-Xms2g -Xmx2g"
```

Si se desea por ejemplo, utilizar 1536MB se debe reemplazar 2g por 1536m:

```
JAVA_ARGS="-Xms1536m -Xmx1536m"
```

11.2.4. Configuraciones para los servidores:

Básicas:

- `dns_alt_names`: Una lista de los hostnames de los servidores permitidos para usar cuando actúan como “masters environment”.
- `path`: Indica la ubicación del entorno.

- basemodulepath: Una lista de las ubicaciones que contienen módulos que pueden ser usados en todos los entornos.
- manifest: El principal punto de entrada para compilar los catálogos. Por defecto es “site.pp”.
- reports: Controlador de reportes que se usa.

Configuraciones de CA:

- ca: Si actúa como un autoridad de certificación.
- ca_ttl: Indica por cuánto tiempo son válidos los certificados.
- autosign: Indica si los certificados deben ser firmados de forma automática o manual.

El archivo de configuración utilizado en la realización del proyecto es el siguiente:

[master]

```
vardir = /opt/puppetlabs/server/data/puppetserver
logdir = /var/log/puppetlabs/puppetserver
rundir = /var/run/puppetlabs/puppetserver
pidfile = /var/run/puppetlabs/puppetserver/puppetserver.pid
codedir = /etc/puppetlabs/code
autosign = true
runinterval = 2m
```

11.2.5. Instalar el paquete puppet-agent

Según sea el sistema operativo sobre el que se instalará, se tiene:

Centos y derivados: Del mismo modo que para el servidor, es necesario añadir el repositorio y luego instalar:

```
sudo rpm -Uvh https://yum.puppetlabs.com/puppetlabs-release-pc1-el-7.noarch.rpm
```

```
sudo yum install puppet-agent
```

Ubuntu y derivados: Habilitar el repositorio según sea el caso como el siguiente ejemplo:

```
wget https://apt.puppetlabs.com/puppetlabs-release-pc1-wheezy.deb
```

```
sudo dpkg -i puppetlabs-release-pc1-wheezy.deb
```

```
sudo apt-get update
```

```
sudo apt-get install puppet-agent
```

Windows: Descargar el paquete instalador desde https://downloads.puppetlabs.com/windows/?_ga=2

- Arquitectura x64: Se recomienda el uso de puppet-agent-<VERSION>-x64.msi o puppet-agent-<VERSION>-x86.msi.
- Arquitectura x86: Debe utilizar puppet-agent-<VERSION>-x86.msi.

La instalación puede ser gráfica o automática. Para la última, desde el cuadro “Ejecutar”: msiexec /qn /norestart /i puppet-agent-<VERSION>-x64.msi PUPPET_MASTER_SERVER=puppet

Este tipo de instalación por comando tiene la ventaja de proveer una mayor cantidad de opciones a editar que en la instalación gráfica. La opción que se destaca es PUPPET_AGENT_ACCOUNT_USER el cual permite indicar qué usuario es que ejecutara los manifiestos de puppet agent en windows. Esta es una opción importante, porque el usuario debe poseer permisos de administrador para poder llevar a los recursos al estado deseado.

No se debe iniciar el servicio aún.

Por defecto, el valor del hostname de servidor es puppet, si se nombró de otra forma a la máquina servidora del puppet server, se debe editar esto.

11.2.6. Configuraciones para los agentes

Básicas:

- server: El nombre del nodo maestro al cual se le pedirán los manifiestos. Por defecto es puppet
- certname: Nombre con el cual el nodo agente pide el certificado y se presenta al servidor.
- environment: Indica el entorno solicitado cuando se contacta al maestro. De cualquier forma, el maestro puede configurarse para ignorar esta configuración.

Comportamiento de la ejecución:

- noop: Si está habilitado, el agente no reaizará ningún trabajo, en cambio mirará qué cambios tendría que realizar y lo reporta al servidor.
- priority: Permite asignar el valor “nice” para evitar que otras aplicaciones de la CPU no mueran por inanición mientras se aplican los catálogos.
- report: Indica si se deben enviar reportes, por defecto es “true” .
- tags: Limita a los agentes a correr recursos con ciertas etiquetas.
- usecacheonfailure: Se utiliza para tomar el último buen catálogo si el master no posee uno bueno.
- prerun_command y postrun_command: Comandos que se desean correr de cada lado de puppet

Comportamiento del servicio:

- runinterval: Indica cada cuánto tiempo el agente se contacta con el servidor para pedirle los manifiestos. Por defecto es 30 minutos.
- waitforcert: Indica al agente que persista si no puede obtener su certificado. Por defecto está habilitado.

El archivo de configuración utilizado en la realización del proyecto es el siguiente:

```
[main]
```

```
logdir = /var/log/puppet
```

```
rundir = /var/run/puppet
ssldir = $vardir/ssl
certname = $HOSTNAME #Aqui nos valemos de una variable que el mismo Snippet
es capaz de resolver para insertar el hostname de la maquina.
server = puppet
runinterval = 2m
[agent]
classfile = $vardir/classes.txt
localconfig = $vardir/localconfig
```

11.2.7. Ejecutar puppet

Para iniciar el servicio de puppet una vez configurado correctamente basta con iniciar lo a travez de systemctl

```
sudo systemctl start puppetserver
sudo systemctl enable puppetserver
```

En el caso de los clientes, tenemos variedad de comandos para iniciar los agentes de puppet puesto que son diferentes los SO donde funcionan.

En el caso de Centos es similar:

```
sudo systemctl start puppet
sudo systemctl enable puppet
```

En el caso de Ubuntu se debe ejecutar:

```
sudo /opt/puppetlabs/bin/puppet resource service puppet ensure=running enable=true
```

Finalmente en windows, puede hacer por consola o por interfaz grafica.

Cualquier sea el SO que ejecute el agente de puppet, lo primero que hara es comunicarse con su servidor, esto podra hacerlo buscando al direccion IP del servidor “puppet” como se lo llamo en el archivo de configuracion en el archivo hosts que su hubicacion difiere en los sistemas Linux y los Windows.

Se genera un certificado sll y el servidor puede firmarlo automaticamente, como en la configuracion de este proyecto o de forma manual.

Para corroborar la creación y firmado del servicio:

```
sudo puppet cert list --all
```

12. Automatización de Windows 7

La instalación automatizada de Windows 7 requiere su preparación sobre un sistema Windows 7 pero su despliegue se realiza desde el servidor Cobbler, bajo Linux.

12.1. El lado de Windows

Para automatizar la instalación de Windows 7, utilizando el método de imagen estándar personalizada, se realiza en siete pasos principales. Para ello es necesario contar con:

- **Máquina técnica:** Con este término, Microsoft se refiere a un sistema Windows utilizado para correr las herramientas y otras operaciones que forman parte de la automatización automatizada. En esta máquina es necesario contar con cualquier versión de Windows 7 que cuente con las herramientas de WAIK (Windows Automated Installation Kit).
- **Máquina de referencia:** Con este término, Microsoft se refiere al equipo sobre el cual se instalará el sistema operativo a partir del cual se extraerá la imagen personalizada con las configuraciones y programas deseados.
- **Windows Automated Installation Kit:** El kit de instalación automatizada de Windows es un conjunto de herramientas y documentación compatible con la configuración y la implementación de los sistemas operativos Windows. Mediante el WAIK, puede automatizar las instalaciones de Windows, capturar imágenes de Windows con ImageX, configurar y modificar imágenes usando Administración y mantenimiento de imágenes de implementación (DISM), crear imágenes de Windows PE, entre otras características.
- **Imagen de instalación de Windows 7:** Ya sea un DVD o un archivo ISO que provea la fuente de instalación del sistema operativo.

La siguiente tabla contiene una columna para cada equipo. Los pasos de la columna en la máquina técnica se realizan en su ese equipo. Los pasos de la columna del equipo de referencia se realizan en el equipo en el que se cree la imagen personalizada.

| Paso | Máquina técnica | Máquina referencia |
|------|--|---|
| 1 | Instalar WAIK | |
| 2 | Crear un disco de inicio del Entorno de Preinstalación de Windows (WinPE) | |
| 3 | | Instalar y personalizar Windows 7 |
| 4 | | Generalizar el equipo de referencia para preparar la imagen para la duplicación |
| 5 | | Capturar el equipo de referencia a un archivo de imagen mediante ImageX |
| 6 | Crear archivo Autounattend.xml | |
| 7 | Crear nuevos medios de instalación de Windows 7 para la imagen personalizada | |

Tabla 1: Pasos automatización Windows 7

12.1.1. Instalar WAIK

Desde la página oficial de Microsoft descargar el archivo KB3AIK_EN.iso, grabarlo en un DVD o montarlo en una unidad virtual e instalarlo.

12.1.2. WinPE

WinPE o Windows Preinstallation Environment es un Sistema operativo mínimo diseñado para preparar una computadora para la instalación de Windows. Puede ser utilizado para iniciar una computadora sin sistema operativo, para particionar y formatear discos rígidos, para copiar imágenes de discos y para iniciar la instalación de Windows desde una ubicación compartida en la red. WinPE se carga directamente en memoria y permite utilizar herramientas como ImageX para capturar, modificar y crear imágenes de instalación basadas en archivos.

Para crear una imagen ISO de WinPE es necesario ejecutar Deployment Tools Command Prompt como administrador.

Se debe copiar a una ubicación elegida los archivos necesarios. Como se ha utilizado una versión de 64 bits se especifica amd64, si no i386.

```
copype amd64 d:\path-del-directorio\winpe
```

Montar el archivo .WIM con:

```
imagex /mountrw d:\path-del-directorio\winpe\winpe.wim 1 d:\path-del-directorio\winpe
```

Cuando se carga en memoria el entorno de preinstalación, es ejecutado un script llamado Startnet.cmd el cual de forma predeterminada inicia Wpeinit.exe, que precisamente sirve para instalar dispositivos Plug and Play, procesar configuraciones de Autounattend.xml y cargar recursos de red. Se debe asegurar que se incluye una llamada a wpeinit en el script Startnet.cmd personalizado:

```
d:\ path-del-directorio\winpe\mount\Windows\System32\startnet.cmd
```

Utilizar un editor de texto y dejarlo como se muestra a continuación, reemplazando los valores correspondientes:

```
wpeinit
echo Conectando con servidor...
net use y: \\IP_del_servidor_Cobbler\entrada_SAMBA
y:
echo Preparando instalacion...
setup.exe /unattend:Autounattend.
```

Una vez modificado los valores del script, desmontar winpe.wim. Para ello es necesario cerrar el explorador de Windows y cualquier programa que esté haciendo uso de directorio anterior, caso contrario dará error. Luego:

```
imagex /unmount d:\path-del-directorio\winpe\mount /commit
```

Se copia entonces winpe.wim al directorio renombrándolo como boot.wim:

```
copy d:\path-del-directorio\winpe\winpe.wim d:\path-del-directorio\winpe\ISO\sources\boot.wim
```

Por último es necesario crear una imagen ISO de WinPE.

```
oscdimg -nt -m -h -bd:d:\path-del-directorio\winpe\etfsboot.com d:\path-del-directorio\winpe\ISO\winpe_cobbler_amd64.iso
```

Las opciones especificadas son las siguientes:

-nt: Permite nombres largos de archivo que sean compatibles con Windows NT 3.51.

-m: Pasa por alto el límite de tamaño máximo de una imagen.

-h: Incluye todos los directorios y archivos ocultos bajo el d:\path-del-directorio\ para esta imagen.

-b: Esta opción se utiliza para especificar el archivo que se escribirá en el sector de arranque del disco. No debe existir espacio alguno entre esta opción y el valor pasado.

-u2: Genera una imagen que sólo incluye el sistema de archivos UDF. Los sistemas que no sean capaces de leer UDF, sólo verán un archivo de texto predeterminado donde se alerta al usuario de que esa imagen sólo está disponible en equipos compatibles con UDF. Esta opción no puede combinarse con las opciones -nt.

12.1.3. Instalar y personalizar Windows 7

Una máquina de referencia tiene una instalación personalizada de Windows que se planea replicar en uno o más computadoras. Se puede crear utilizando un DVD o archivo ISO de Windows.

Este punto conviene realizarlo manualmente y de la forma usual. Sin embargo, cuando en el proceso de instalación, se llega a la pantalla de bienvenida de Windows (diálogo de creación de usuario y hostname) se debe presionar *Ctrl + Shift + F3* para ingresar en modo auditoría. Ahora la máquina debería reiniciarse e ingresar automáticamente en una cuenta temporal en modo administrador. Esto permanecerá así por más que se reinicie el equipo hasta que el comando Sysprep sea ejecutado luego de realizar todas las actualizaciones, los controladores de dispositivo e instalación de las aplicaciones.

Cada vez que sea reiniciado el equipo estando en este modo, aparecerá un cuadro de diálogo del modo auditoría. Simplemente ignorarlo hasta terminar de preparar la imagen.

Para pasarle los programas a instalar, alojados en el servidor en */windows/PROGRAMAS/*, desde la máquina de referencia ir a *Equipo → Conectar a unidad de red* e ingresar la dirección IP del servidor con el formato:

```
\IP_del_servidor_Cobbler\entrada_SAMBA
```

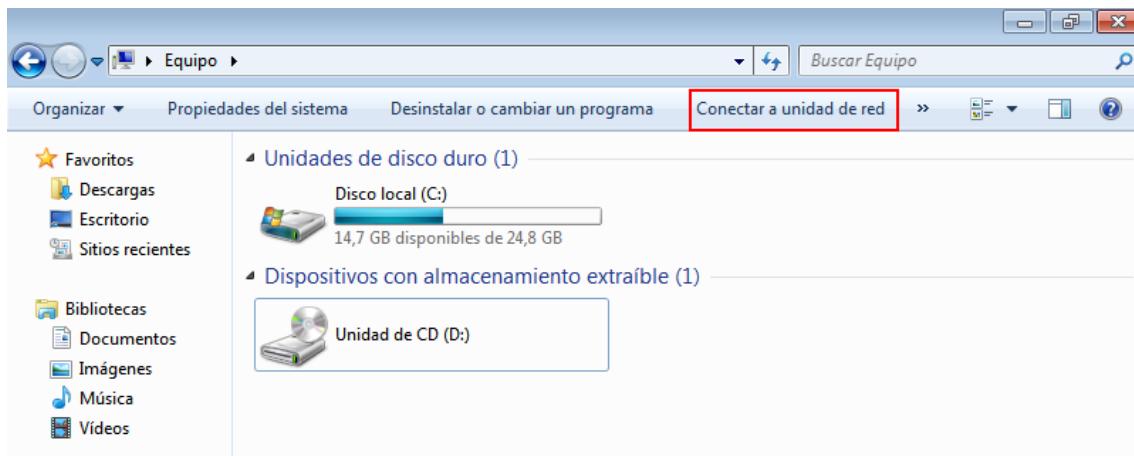


Figura 28: Paso de archivos a Windows

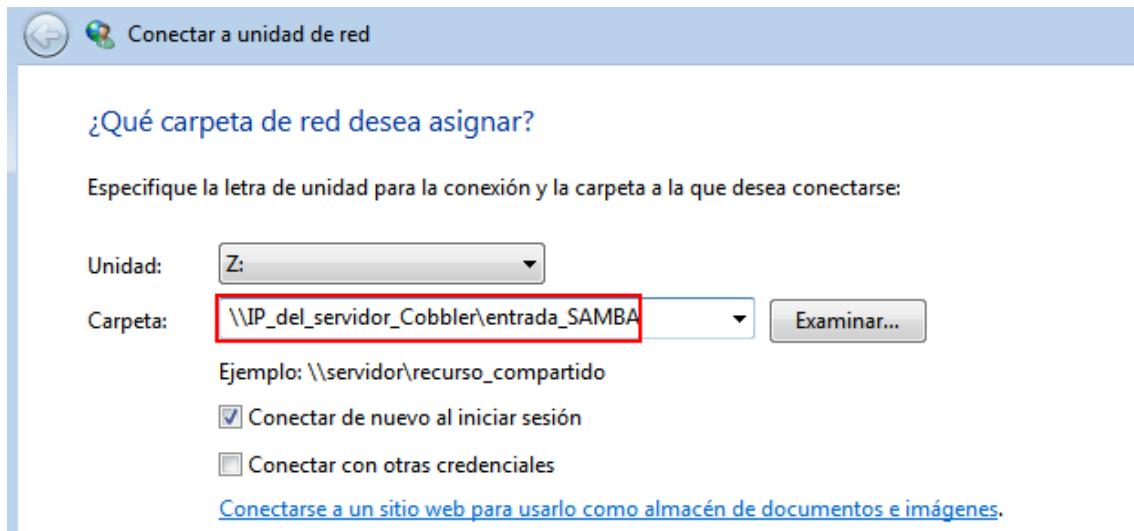


Figura 29: Paso de archivos a Windows

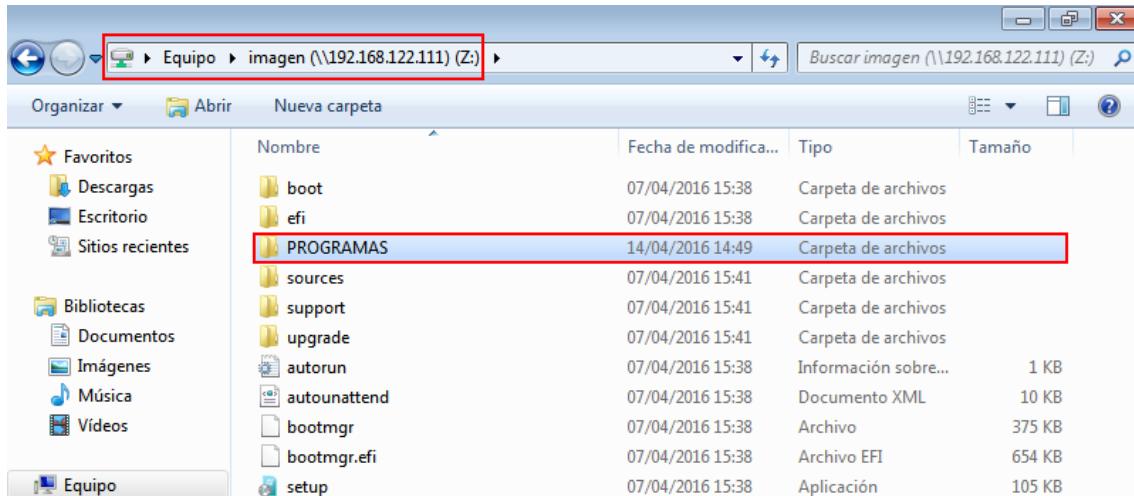


Figura 30: Paso de archivos a Windows

12.1.4. Generalizar el equipo de referencia para preparar la imagen

En este paso, se generaliza la imagen y se prepara para su inicio en Bienvenida de Windows después de haberla instalado en cada equipo. Al generalizar la imagen, se elimina de ella la información que depende de hardware, se restablece el temporizador de activación y se limpia Windows 7 para que se pueda duplicar la imagen en otros equipos.

Para esto se hace uso de la herramienta Sysprep, la cual limpia las configuraciones específicas de usuario y del hardware.

De forma gráfica, cuando inicia el sistema operativo en modo auditoría, Windows 7 ejecuta Sysprep de forma automática.

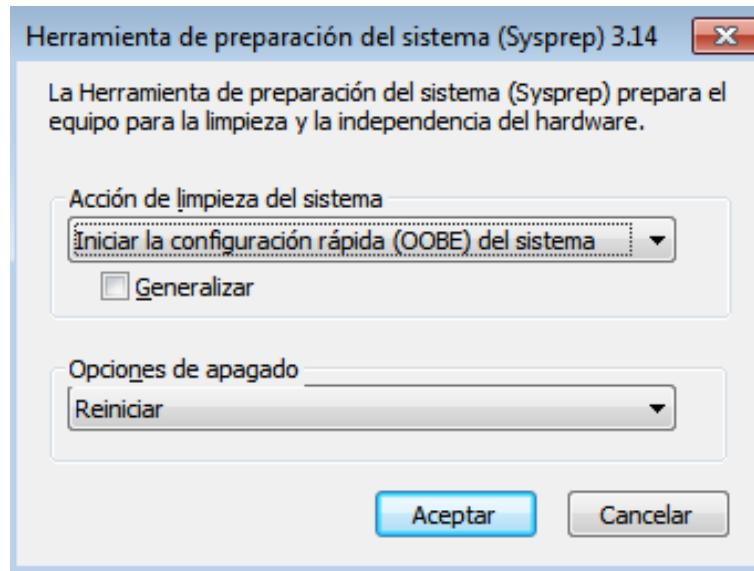


Figura 31: Modo auditoría

- En la lista Acción de limpieza del sistema, seleccionar Iniciar la configuración rápida (OOBE) del sistema.
- Activar la casilla Generalizar.
- En la lista Opciones de apagado, seleccionar Apagar.
- Aceptar para ejecutar Sysprep y apagar el equipo.

En este caso particular se decidió realizar la generalización del sistema por consola. Para ello es necesario cargar un WinPE.iso sin modificaciones.

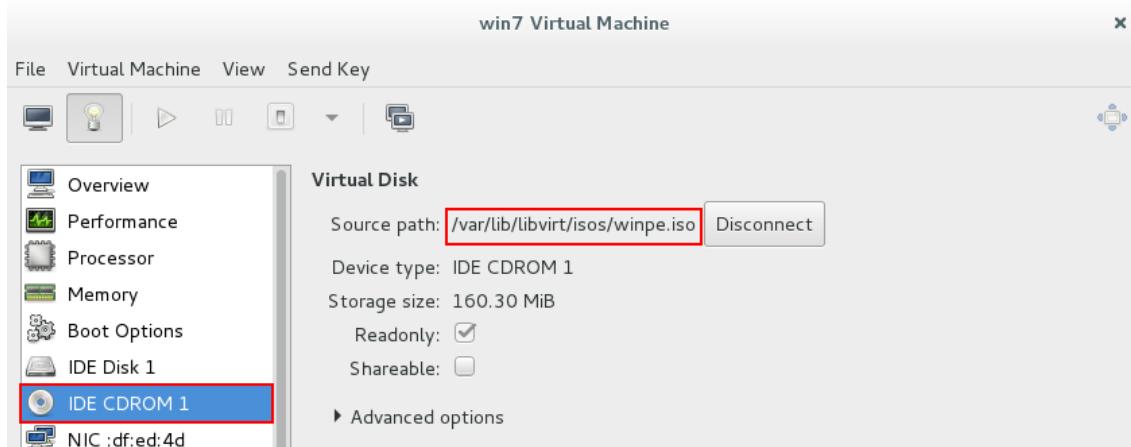


Figura 32: Carga WinPE.iso para extraer imagen

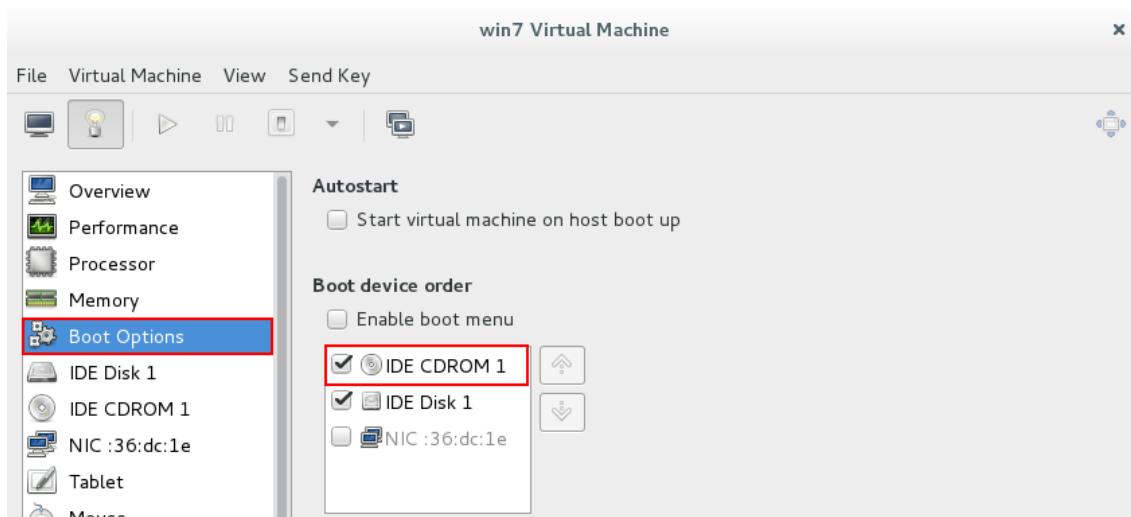


Figura 33: Prioridad de inicio en máquina virtual

Desde la consola, ejecutar por línea de comando desde C:\Windows\System32\Sysprep\:
`sysprep /generalize /oobe /shutdown`

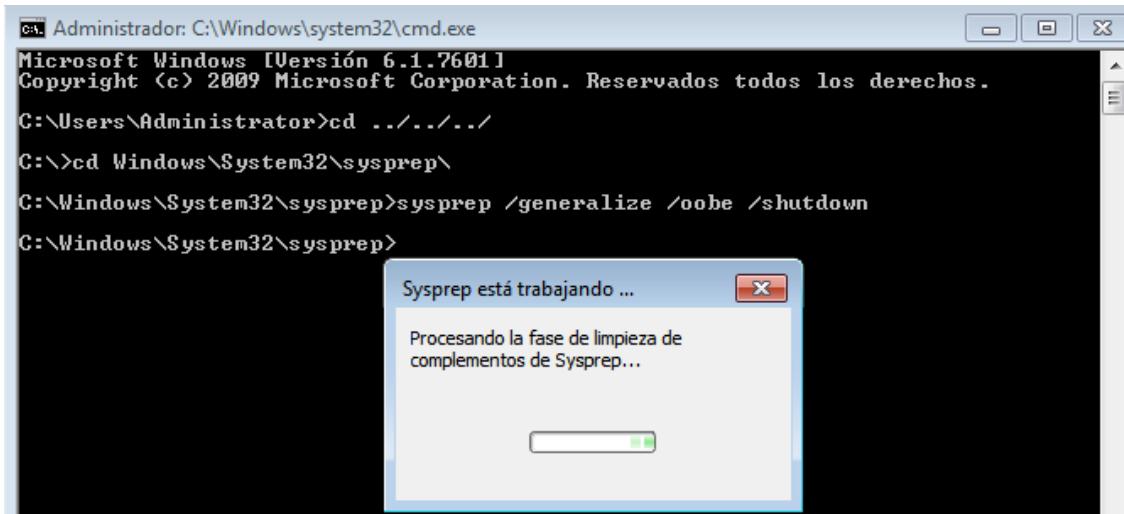


Figura 34: Ejecución sysprep

Donde las opciones más importantes son:

/generalize: Prepara a la instalación de Windows para crear una imagen. Si esta opción es especificada, toda la información única del sistema es removida. El SID (ID de Seguridad) se resetea, cualquier punto de restauración que exista es quitado y todos los logs son eliminados. La próxima vez que la computadora inicia, la fase de configuración especial es ejecutada y se asigna un nuevo SID y el reloj de activación de Windows se resetea si no ha sido reseteado ya tres veces, esto es, este proceso no puede repetirse más de tres veces por instalación de Windows. En ese caso, se debe comenzar con una nueva instalación desde cero.

/oobe: El término OOBE significa Out Of Box Experience y es la experiencia que un usuario tiene cuando se prepara para usar por primera vez un producto. Cuando esta opción es seleccionada, reinicia el equipo en modo Bienvenida de Windows. Este modo permite a los usuarios personalizar el sistema operativo, crear cuentas de usuario, cambiar el hostname y demás. Cualquier configuración pasada al oobeSystem por medio de un archivo de respuestas de instalación es procesada inmediatamente antes que la Bienvenida de Windows inicie.

/shutdown: Apaga la máquina luego que el comando sysprep finaliza.

/unattend:answerfile: Aplica las configuraciones dentro de un archivo de respuestas de instalación sobre el sistema operativo durante una instalación desatendida; answerfile especifica el path y el nombre del archivo que contiene las respuestas. Esta opción se puede ignorar, ya que como se verá más adelante basta con agregar este archivo en el directorio principal de instalación para que sea tomado por el instalador de Windows.

A esta altura del proceso, ha instalado Windows 7 en el equipo de referencia y está listo para capturar una imagen del mismo.

12.1.5. Capturar el equipo de referencia

En este paso se capture una imagen de la máquina de referencia utilizando la herramienta ImageX, que está en el WinPE anteriormente creado. La imagen será guardada en el

directorio compartido SAMBA. Se carga nuevamente el disco WinPE y se obtiene una consola.

Verificar que la máquina ve al servidor que tiene SAMBA configurado y corroborar que se tiene acceso a la ubicación compartida:

```
net use y: \\IP_del_servidor_Cobbler\entrada_SAMBA
y:
dir
```

Verificar la letra asignada al disco WinPE, usualmente es *e*: y verificar la letra de la partición donde está instalado Windows ya personalizado, usualmente *c*:

```
X:\windows\system32>net use y: \\192.168.122.111\imagen
The command completed successfully.

X:\windows\system32>e:
E:>dir
Volume in drive E is CD_ROM
Volume Serial Number is 6BF9-3D25

Directory of E:\

02/09/2016  12:55 PM    <DIR>          BOOT
07/13/2009  01:39 PM        383,562  BOOTMGR
07/13/2009  01:45 PM        667,712  BOOTMGR.EFI
02/09/2016  12:55 PM    <DIR>          EFI
07/14/2009  02:10 AM        581,008  IMAGEX.EXE
02/09/2016  12:55 PM    <DIR>          SOURCES
                           3 File(s)     1,632,282 bytes
                           3 Dir(s)           0 bytes free

E:>d:
D:>dir
Volume in drive D has no label.
Volume Serial Number is D4B6-74A5

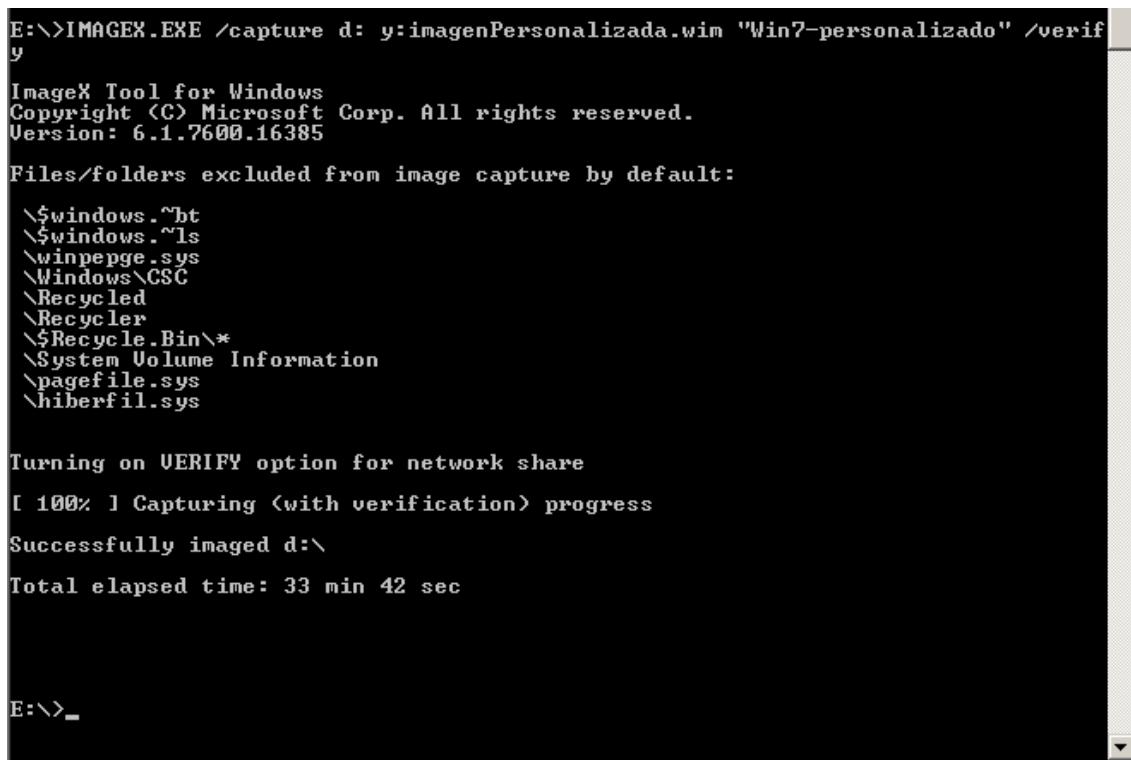
Directory of D:\

07/13/2009  07:20 PM    <DIR>          PerfLogs
04/14/2016  12:37 AM    <DIR>          Program Files
04/14/2016  12:39 AM    <DIR>          Program Files (x86)
11/20/2010  06:50 PM    <DIR>          Users
04/14/2016  01:45 AM    <DIR>          Windows
                           0 File(s)           0 bytes
                           5 Dir(s)   11,730,722,816 bytes free
```

Figura 35: Verificación de letras de unidad

Desde la unidad que contiene a WinPE:

```
e:\imageX.exe /capture d y:\windows\imagenPersonalizada.wim "Win7-Personalizado"
/verify
```



```

E:>IMAGEX.EXE /capture d: y:imagenPersonalizada.wim "Win7-personalizado" /verify
y

ImageX Tool for Windows
Copyright (C) Microsoft Corp. All rights reserved.
Version: 6.1.7600.16385

Files/folders excluded from image capture by default:
\$windows.\$bt
\$windows.\$ls
\winpege.sys
\Windows\CSC
\Recycled
\Recycler
\$Recycle.Bin\*
\System Volume Information
\pagefile.sys
\hiberfil.sys

Turning on VERIFY option for network share
[ 100% ] Capturing (with verification) progress
Successfully imaged d:\
Total elapsed time: 33 min 42 sec

E:>_

```

Figura 36: ImageX capturando imagen

Donde se tiene:

/capture c: Indica al programa que se quiere capturar la imagen ubicada en esa unidad de partición.

y:\windows\imagenPersonalizada.wim Es el path completo donde se guardará la imagen en SAMBA.

"Win7-Personalizado" Es una etiqueta para el archivo imagen creado.

/verify Indica al programa que se desea verificar la consistencia de la imagen creada.

12.1.6. Autounattend.xml

El archivo *Autounattend.xml* es donde se guardan los valores correspondientes a los diálogos de instalación del sistema operativo elegido. Ítems como creación de cuentas de usuario, idioma, configuraciones regionales, configuraciones de red, particionado de discos, etc. son los que se automatizarán. Para generararlo, una vez instalado el WAIK, se debe abrir el modo Administrador el programa Windows System Image Manager desde el menú Inicio.

Bajo la sección *Windows Image* seleccionar *Select a Windows image or catalog file*. En este punto se debe buscar en el directorio de la imagen del sistema operativo ya descomprimida, un archivo *.clg* que en este caso particular es *install_Windows 7 ULTIMATE.clg*. Acto seguido crear el archivo de respuestas de instalación *Create a New Answer File*.

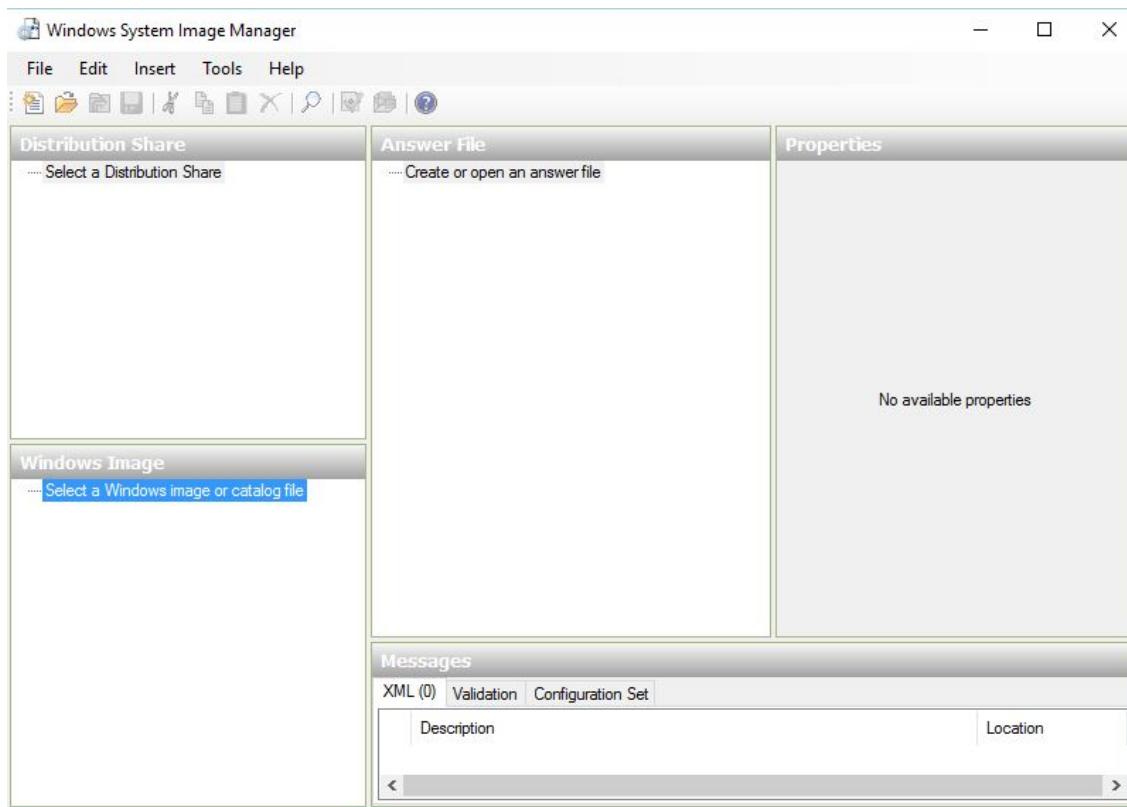


Figura 37: Windows System Image Manager

Se muestra a continuación una configuración personalizada completa utilizando Windows AIK.

Se ha configurado la entrada de texto, interfaz gráfica y ayuda del sistema en idioma español.

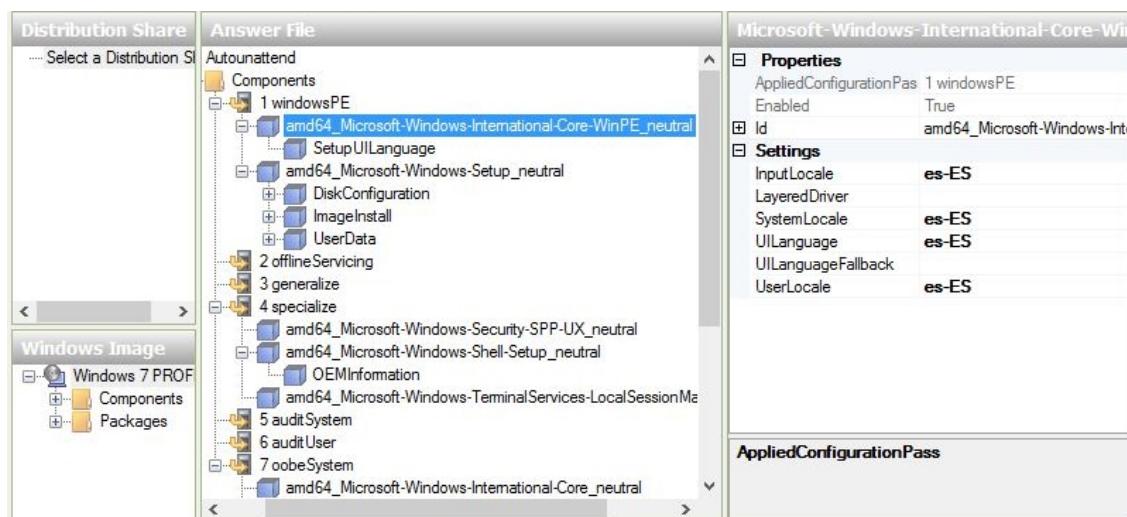


Figura 38: Idioma y teclado.

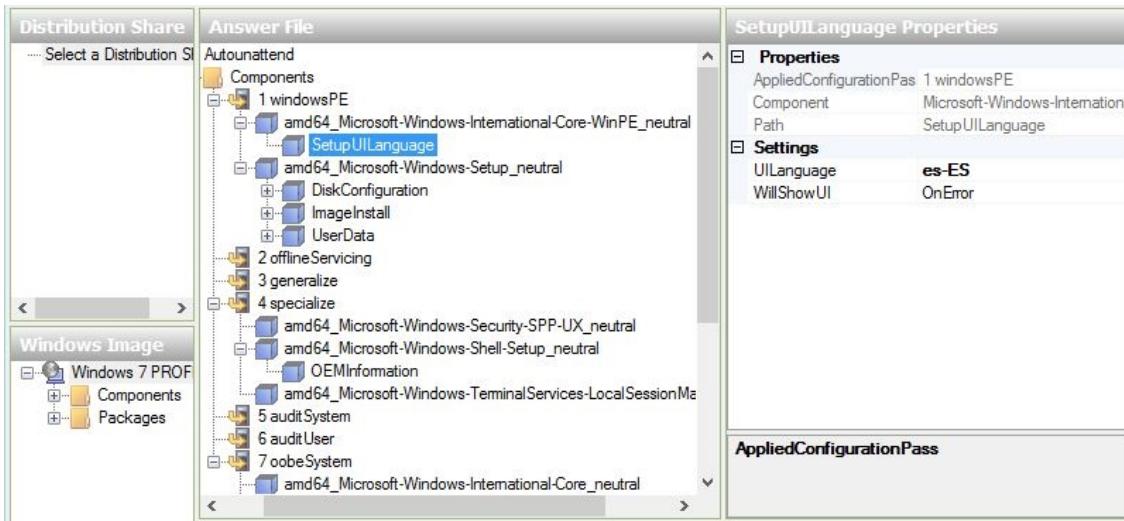


Figura 39: Idioma .

Se ha configurado el firewall para que esté habilitado, valor por defecto.

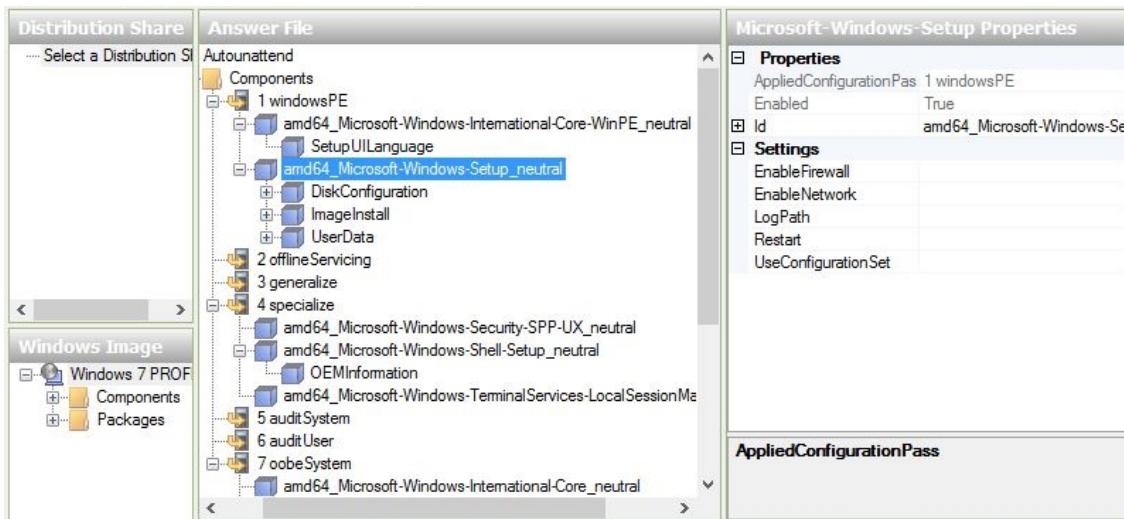


Figura 40: Firewall activado por defecto.

Se han configurado los discos para que existan dos particiones. Una donde se instalará el MBR y otra donde se copiarán los archivos del sistema operativo.

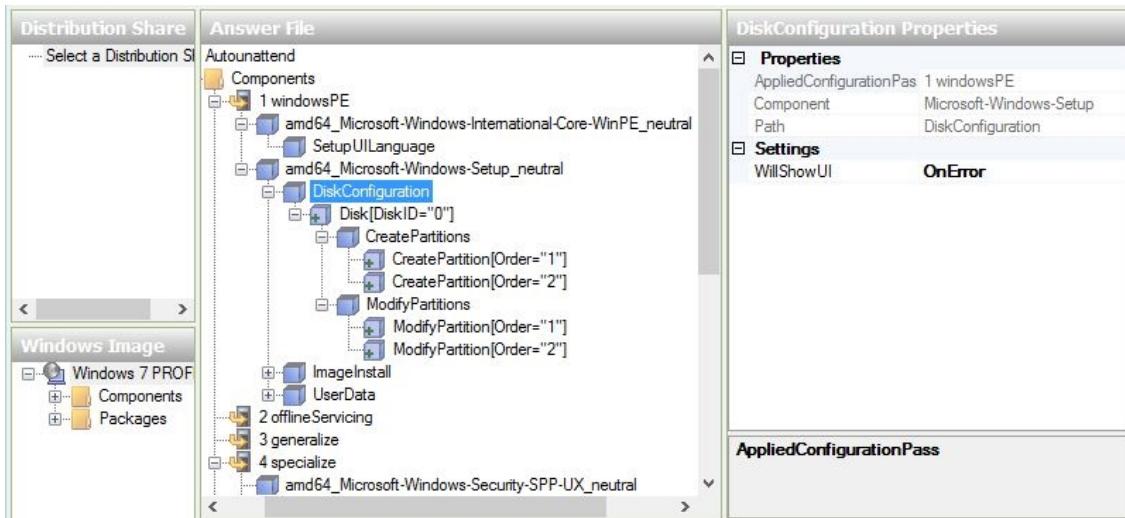


Figura 41: Discos.

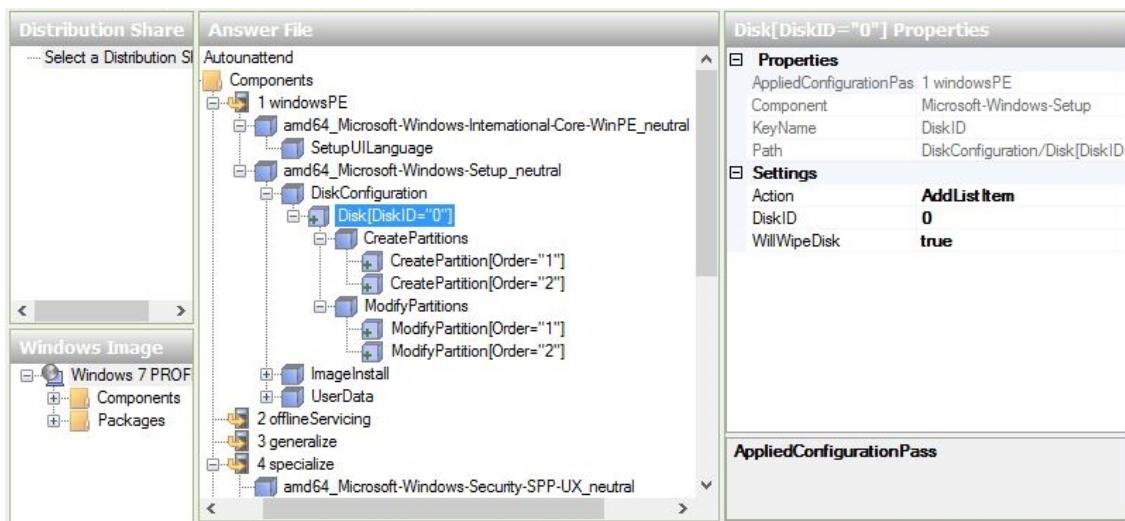


Figura 42: Configuración discos. Disco 0.

Partición de 300 MB, donde será instalado el MBR.

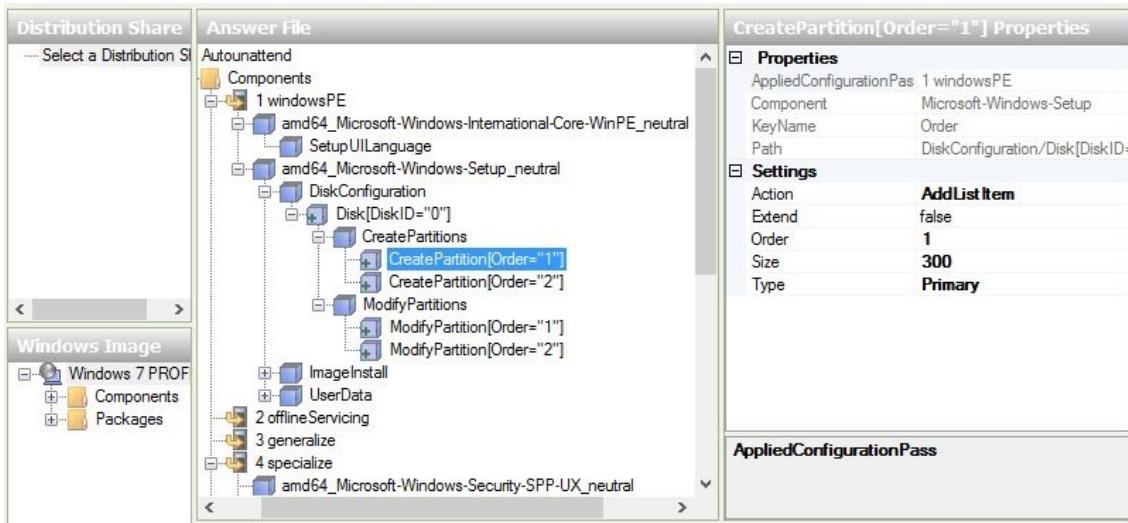


Figura 43: Discos. Particionamiento.

Partición que utilizará el espacio restante del disco, donde se copiarán los archivos de Windows.

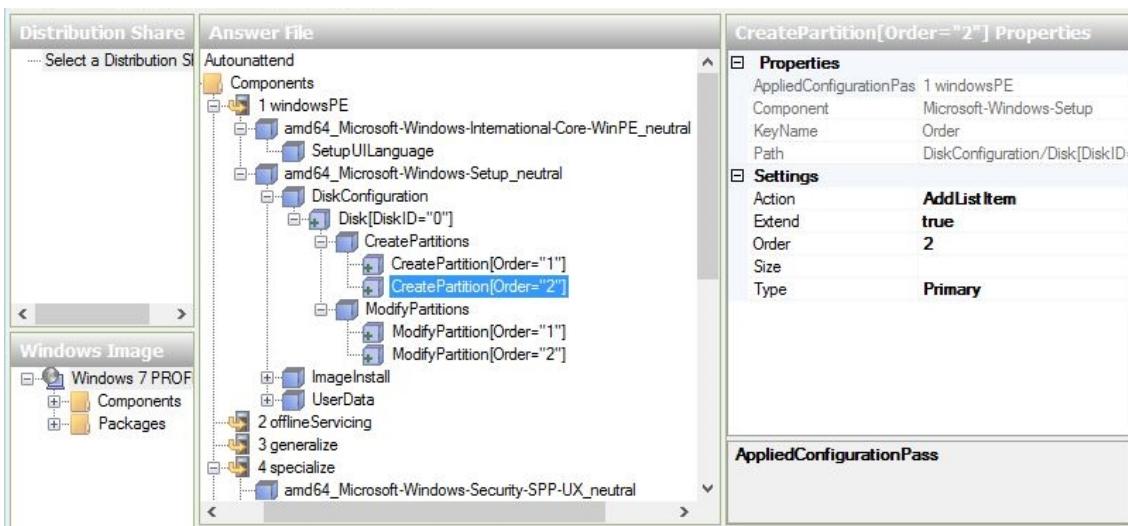


Figura 44: Discos. Particionamiento.

Formato de la partición.

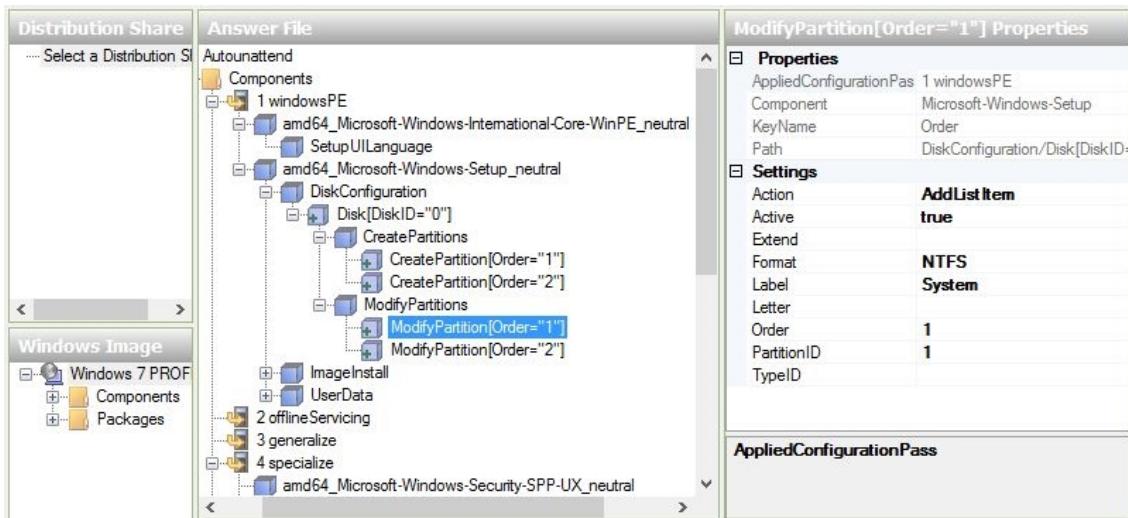


Figura 45: Discos. Particionamiento.

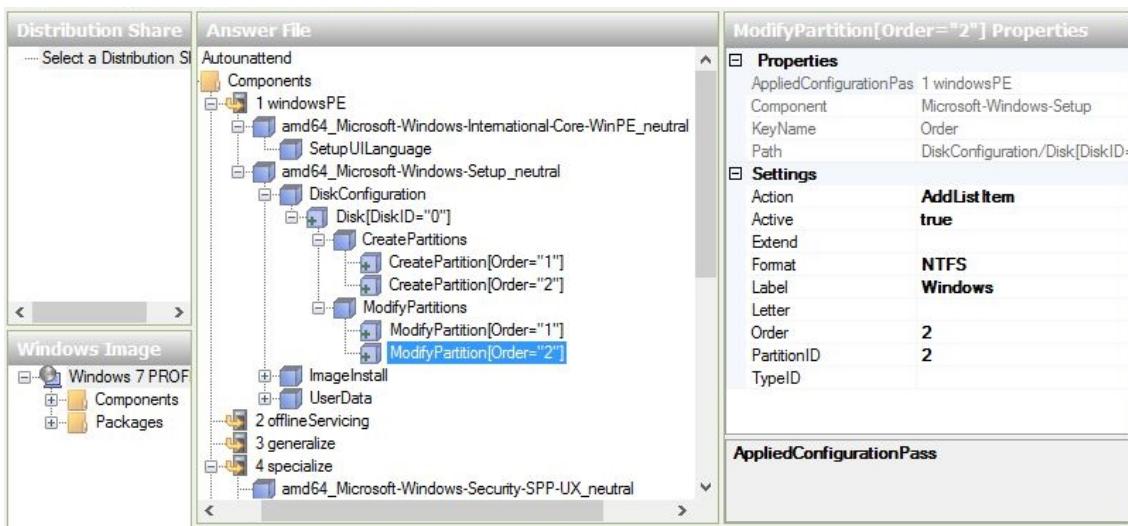


Figura 46: Discos. Particionamiento.

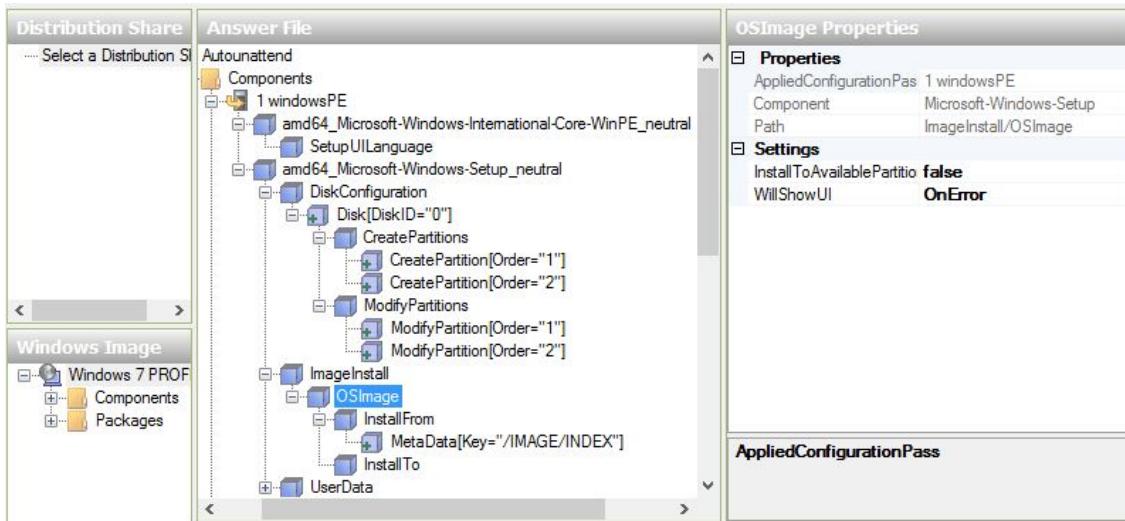


Figura 47: Instalación del SO.

En el caso de contar con una imagen de instalación multiversión, Windows Home, Basic, Professional y Ultimate y/o con ambas arquitecturas, i386 y amd64, es necesario seleccionar cuál de ellas se desea instalar. La opción más fácil y efectiva es contar la posición de la versión que se quiere instalar y seleccionarla como se muestra a continuación, donde se ve que en este caso es la primera opción.

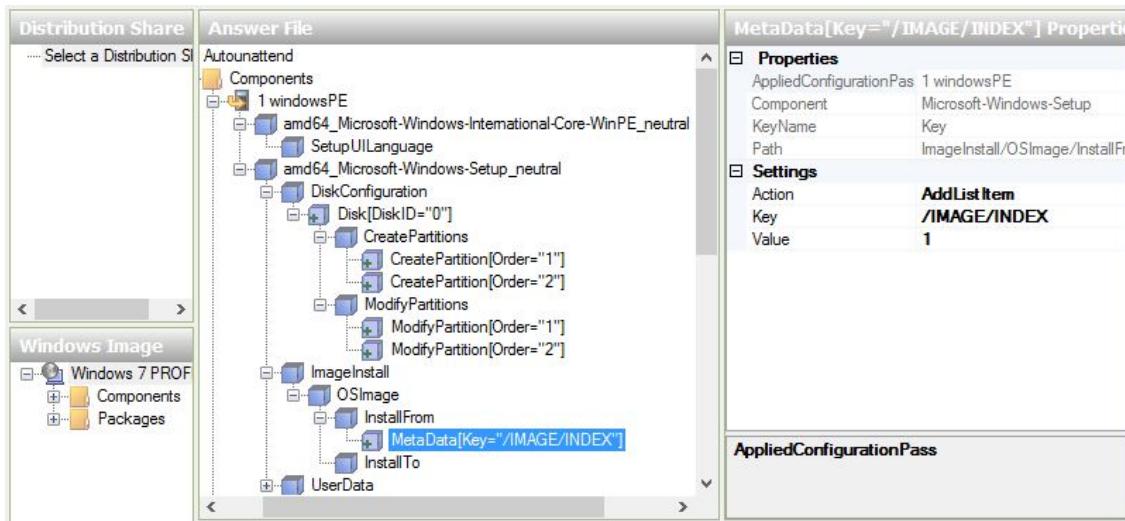


Figura 48: Instalación del SO. Elección de la versión.

Selección de la partición de instalación de Windows.

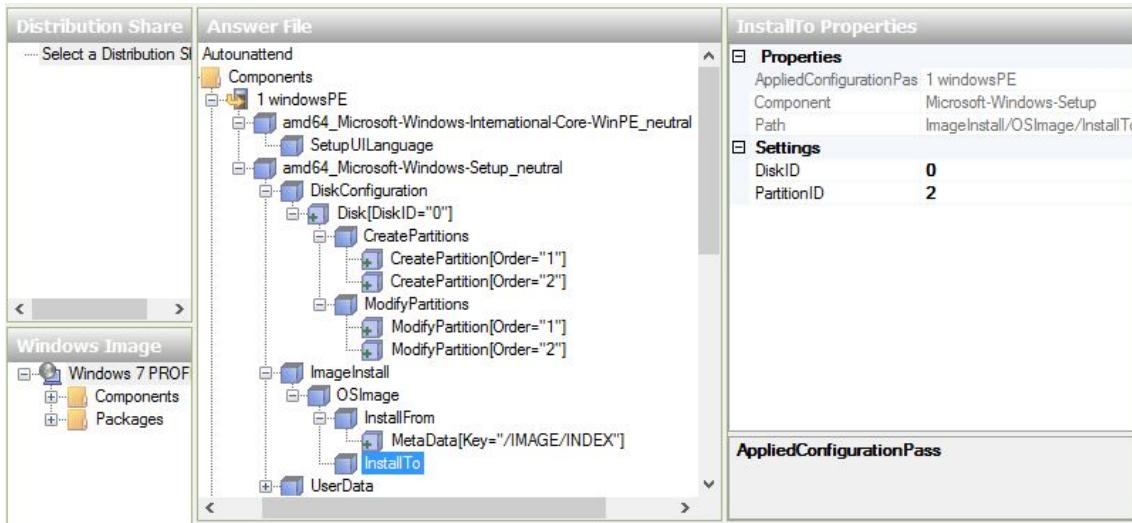


Figura 49: Elección de partición de instalación.

Aceptación de las condiciones de privacidad y uso del sistema operativo.

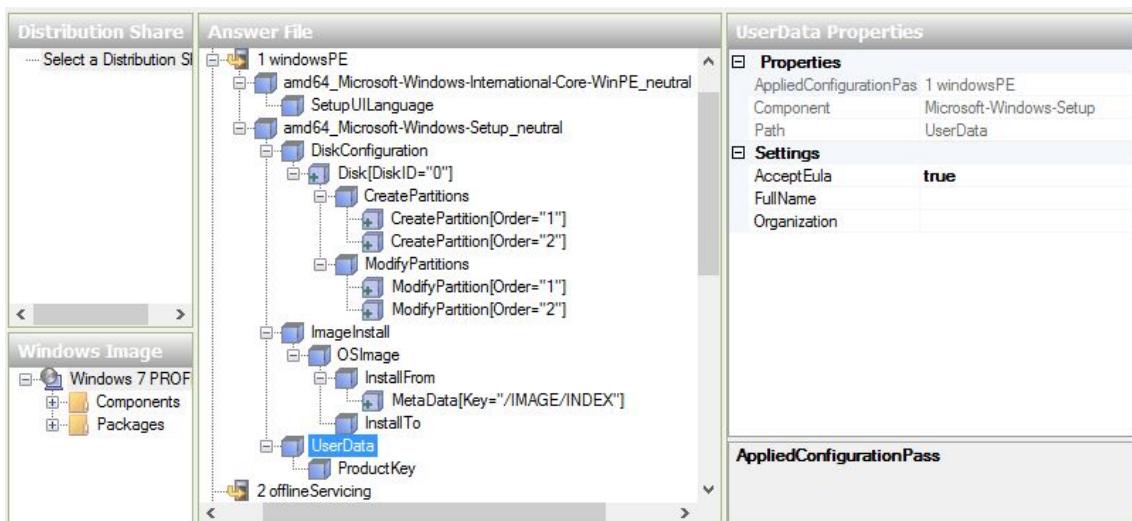


Figura 50: Aceptación de las condiciones de privacidad y uso.

La clave de producto, para poder realizar instalaciones desatendidas debe ser del tipo de Volúmen, esto es, una clave para realizar instalación en masa. A modo de ejemplo y para pruebas, Microsoft entrega una serie de claves para ésto: <https://technet.microsoft.com/en-us/library/jj612867.aspx>

Entonces, la clave de producto utilizada en este caso es: **FJ82H-XT6CR-J8D7P-XQJJ2-GPDD4**.

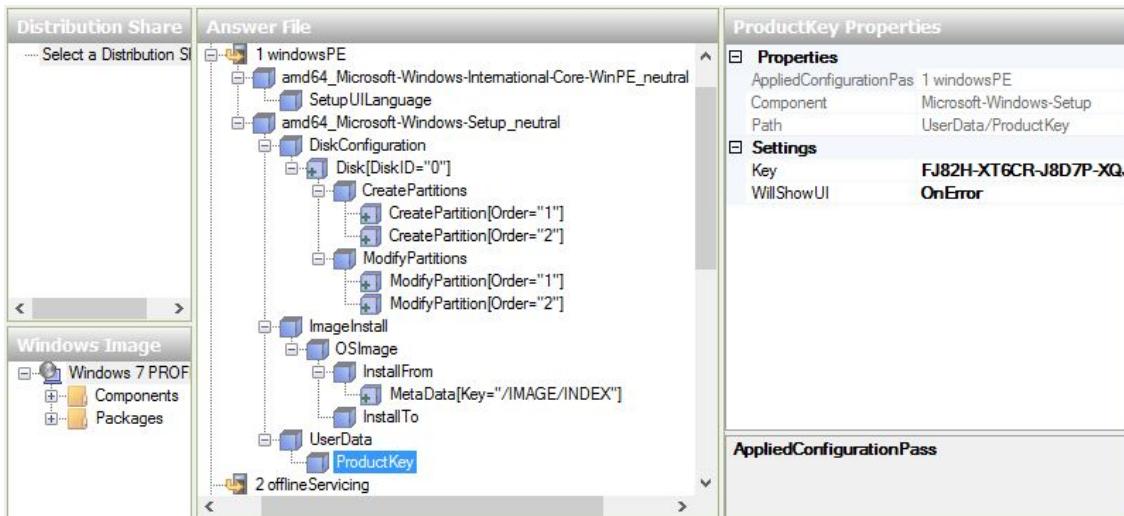


Figura 51: Clave de producto.

Saltar activación del producto.

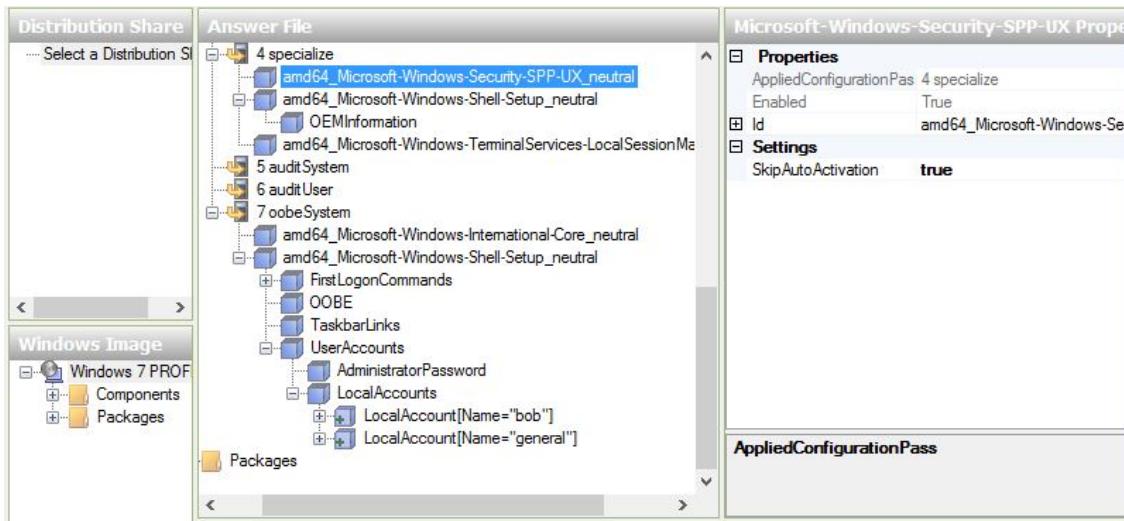


Figura 52: Activación del producto.

Para solucionar el problema de asignar un hostname único a cada máquina que sea instalada con este *Autounattend.xml* se dió con la siguiente solución. Utilizando un string representativo en el campo RegisteredOwner y un asterisco en el campo ComputerName, se obtiene un hostname pseudo-aleatorio, ya que estará compuesto por máximo ocho caracteres de RegisteredOwner y/o RegisteredOrganization concatenado con caracteres aleatorios. El campo ComputerName es un string con un tamaño máximo de quince caracteres.

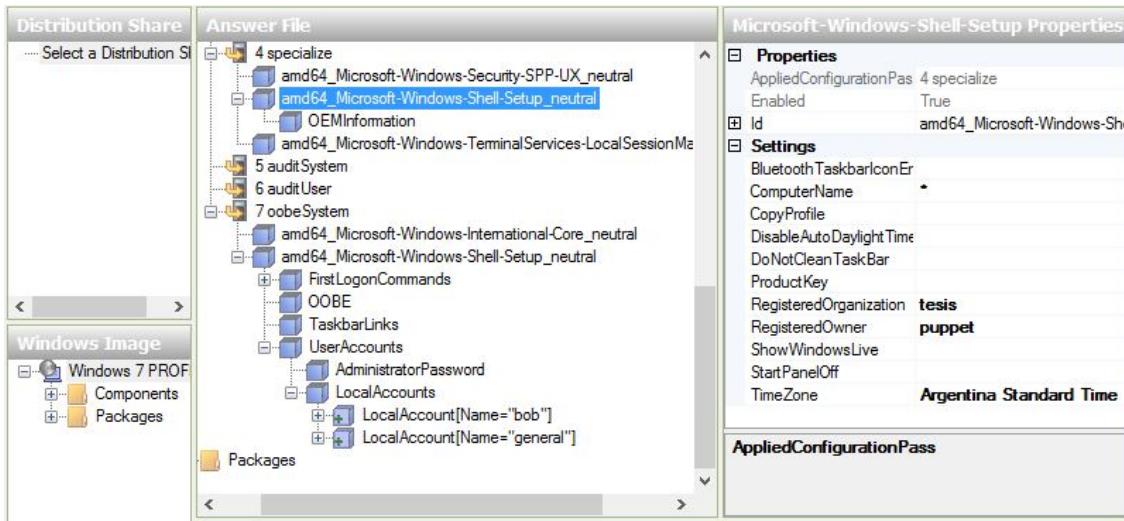


Figura 53: Hostname, organización, propietario y zona horaria.

Se debe ingresar nuevamente la configuración deseada.

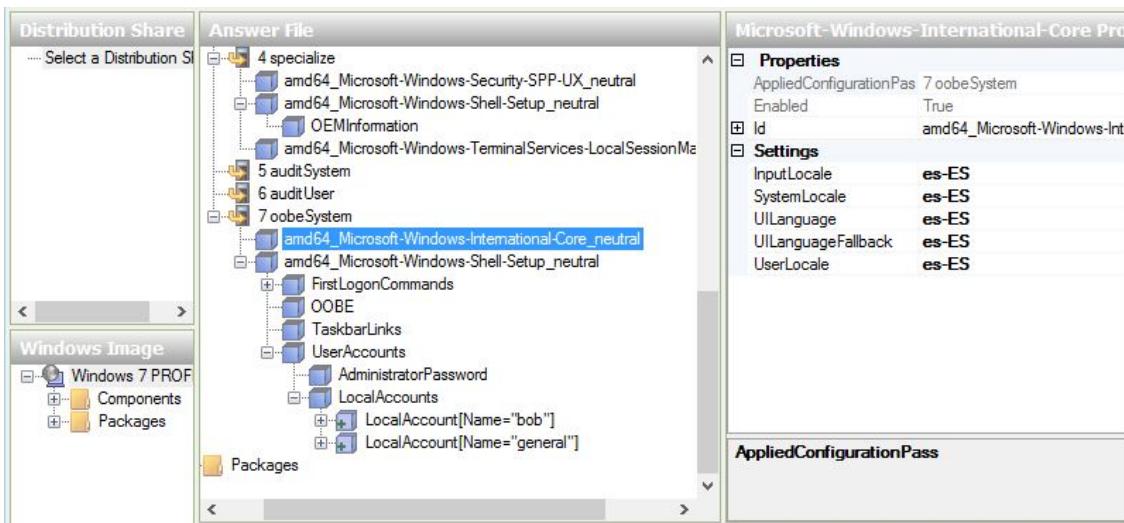


Figura 54: Idioma, teclado, regional.

Se debe ingresar nuevamente la configuración deseada.

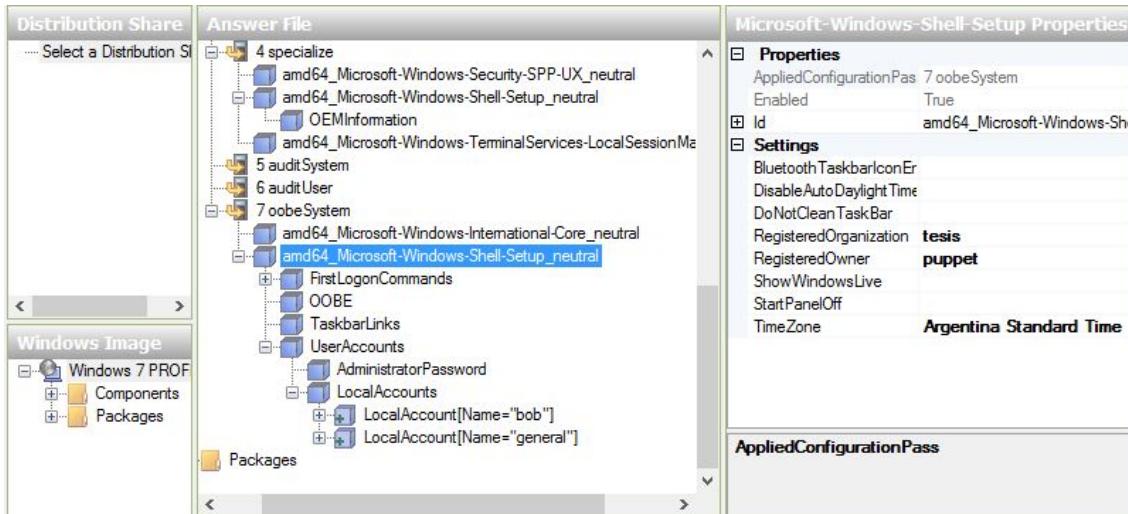


Figura 55: Organización, propietario y zona horaria.

Selección de la configuración rápida de red.

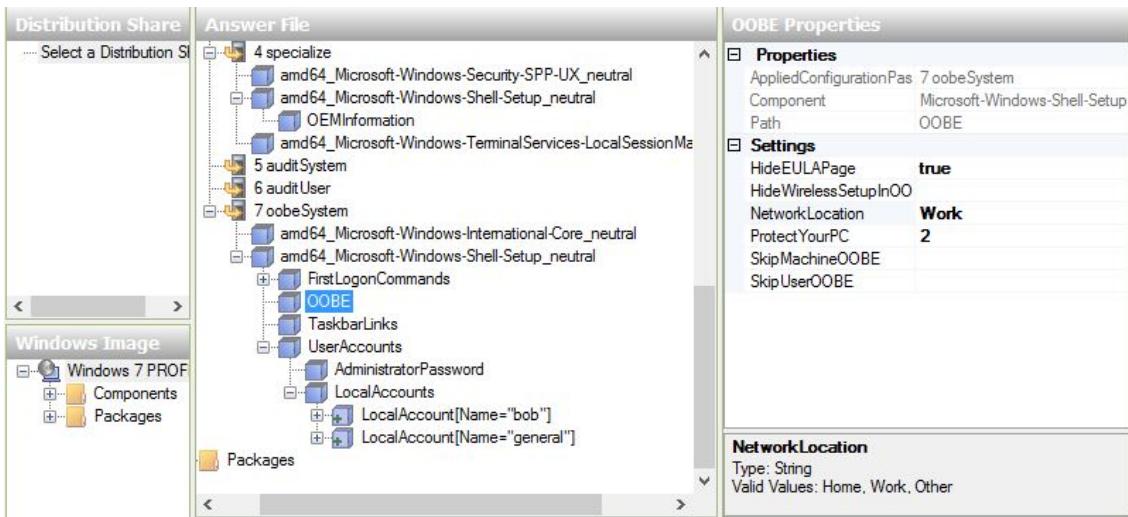


Figura 56: Configuración de red.

Configuración de contraseñas de administrador.

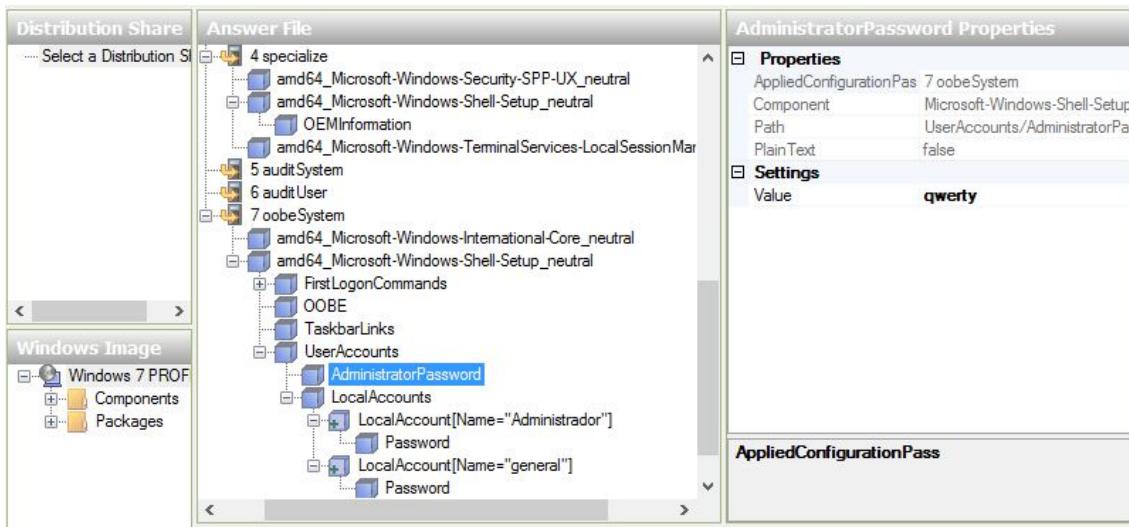


Figura 57: Contraseña administrador.

Creación de cuentas de usuario.

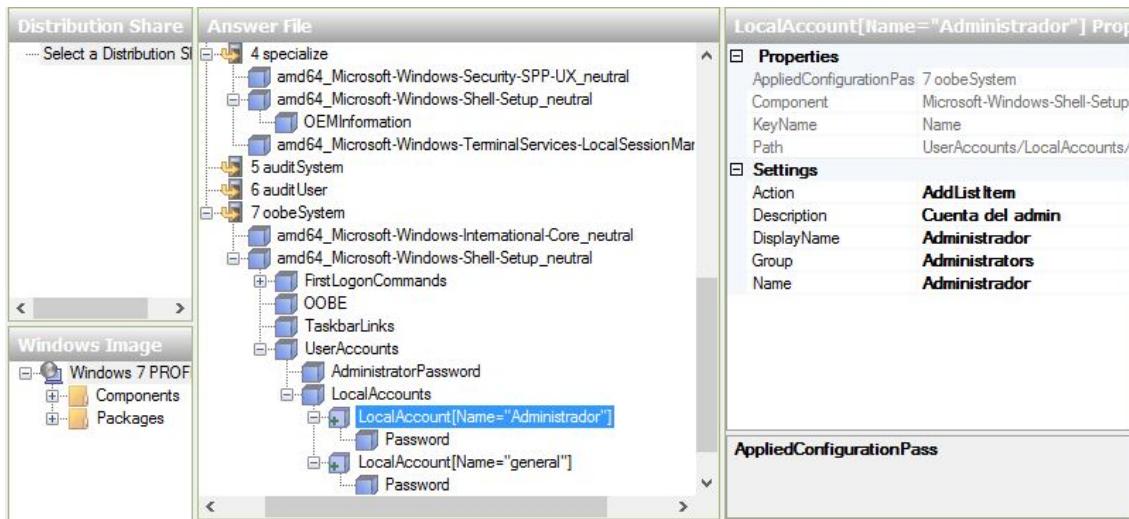


Figura 58: Cuentas de usuario, administrador.

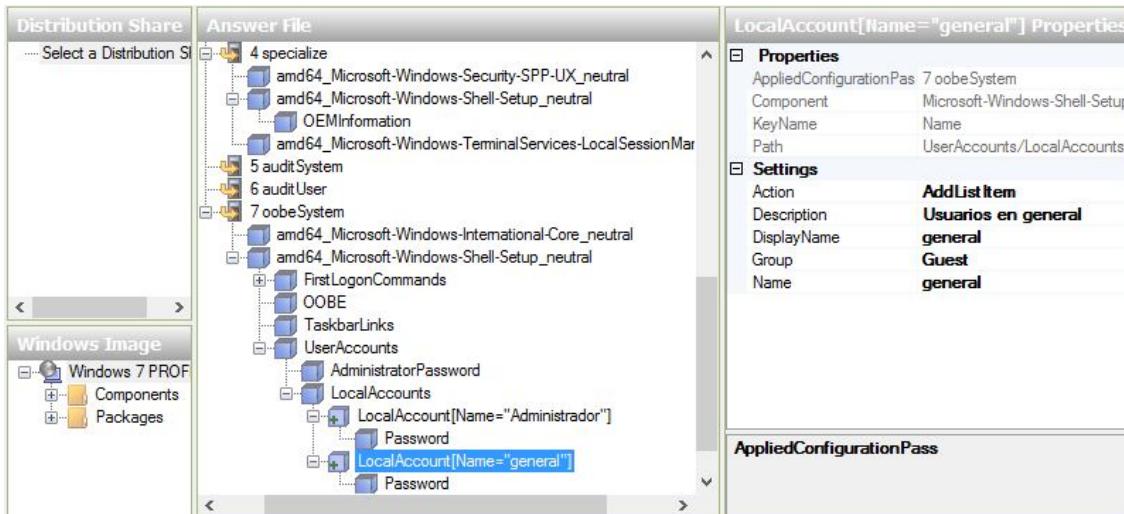


Figura 59: Cuentas de usuario, invitado.

12.1.7. Crear nuevos medios de instalación de Windows 7 para la imagen personalizada

En la máquina técnica realizar las siguientes operaciones:

- Descomprimir todo el contenido de la imagen de instalación sin modificar de Windows 7 en \path-diretorio-instalacion\.
- En el sub-directorio Sources (\path-diretorio-instalacion\Sources\), substituir la imagen install.wim, por la recientemente imagen capturada imagenPersonalizada.wim, renombrándola como install.wim.
- En el directorio principal de donde ha sido extraída la imagen, copiar el archivo de respuestas de instalación Autounattend.xml.
- Abrir Deployment Tools Command Prompt como administrador, y ejecutar:
 - oscdimg -m -h -u2 -bd:\path-diretorio-instalacion\boot\etfsboot.com d:\path-diretorio-instalacion\ d:\path-diretorio-destino\Win7-personalizado.iso

En este punto se tiene una imagen de instalación personalizada y automatizada lista para ser instalada en cualquier equipo. A continuación, se verá la preparación necesaria del servidor Linux para poder instalar y configurar SAMBA y la integración de Windows 7 con Cobbler para el posterior despliegue a través de la red.

12.2. El lado de Linux

Lo primero que se debe hacer es copiar todos los archivos necesarios, anteriormente generados en Windows, al servidor de despliegue, Cobbler.

En el servidor crear el siguiente directorio, donde se colocarán los archivos winpe_cobbler_amd64.iso y Win7-personalizado.iso:

```
sudo mkdir /var/lib/cobbler/isos
```

Darle los permisos necesarios:

```
sudo chmod -R 777 /var/lib/cobbler/isos/*.iso
```

También es necesario crear un directorio que se utilizará como zona de intercambio entre Linux y Windows:

```
sudo mkdir /windows/
```

En este directorio se debe copiar el contenido de la imagen de instalación de Windows personalizada, para ello:

```
sudo mkdir /mnt/windows
```

```
sudo mount -o loop /root/isos/WIN7_X64.iso /mnt/windows
```

```
sudo cp -rf /mnt/windows /windows
```

```
sudo umount /mnt/windows
```

La zona de intercambio entre ambos sistemas operativos se logra utilizando SAMBA.

Para instalar SAMBA en CentOS 7 ejecutar:

```
sudo yum install -y SAMBA SAMBA-client SAMBA-common SAMBA-winbind
```

Una vez finalizada la instalación agregar una ubicación compartida en el archivo de configuración */etc/SAMBA/smb.conf* como se muestra a continuación:

```
[global]
workgroup = PXESERVER
server string = SAMBA Server Version %v
log file = /var/log/SAMBA/log.%m
max log size = 50
idmap config * : backend = tdb
cups options = raw
netbios name = pxe
map to guest = bad user
dns proxy = no
public = yes
## Para instalaciones multiples no bloquear el kernel
kernel oplocks = no
nt acl support = no
security = user
guest account = nobody
```

```
[imagen]
```

```
comment = Windows 7
```

```

path = /windows
read only = no
browseable = yes
public = yes
printable = no
guest ok = yes
oplocks = no
level2 oplocks = no
locking = no

```

Habiendo modificado la configuración reiniciar el servicio:

```
sudo systemctl restart smb
```

Corroborar que se pueden compartir archivos, si se hace desde otra pc Linux, es necesario tener los siguientes paquetes:

```
sudo yum install SAMBA SAMBA-client SAMBA-common cifs-utils
```

Luego ejecutar:

```

sudo smbclient -L IP_del_servidor_Cobbler
sudo mkdir /media/SAMBA
sudo mount // IP_del_servidor_Cobbler /imagen/media/SAMBA/
sudo ls /media/SAMBA/

```

Deberían aparecer ya los archivos compartidos en esa ubicación.

12.2.1. Integración con Cobbler

Para integrar la imagen de instalación Windows al servidor Cobbler se debe añadir una entrada distro y una entrada de perfil a Cobbler.

```

sudo cobbler distro add --name=windows7-x86_64 --kernel=/usr/share/syslinux/memdisk
--initrd=/var/lib/cobbler/isos/winpe_cobbler_amd64.iso --kopts="raw iso" --arch=x86_64
--breed=windows

```

```

sudo cobbler profile add --name=windows7-x86_64 --distro=windows7-x86_64
sudo cobbler sync
sudo systemctl restart cobblerd

```