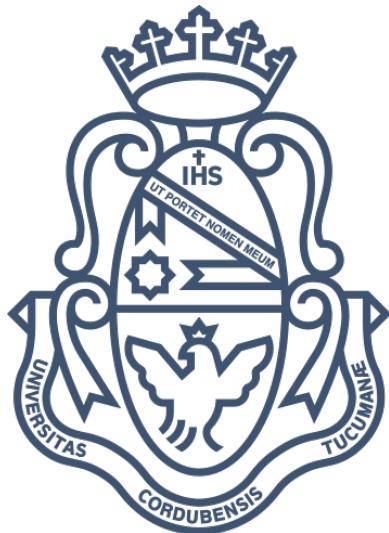


Universidad Nacional de Córdoba



5 de noviembre de 2016

Facultad de Ciencias Exactas, Físicas y Naturales

Infraestructura tecnológica virtual con automatización y orquestación.

Juan Pablo, Arese

(Email: juan.p.arese@gmail.com, Teléfono (03564) 15524759, Matrícula 36935267)

Werner Christian, Diers

(Email: wernerdiels@gmail.com, Teléfono (0351) 152353267, Matrícula 35109591)

Director de PI: Eschoyez, Maximiliano Andrés

Co-director de PI: Migliazzo, Oscar Andrés

Resumen

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

La cantidad de servicios y servidores necesarios en las organizaciones tiende a ser cada vez mayor. Cada día los despliegues son más complejos, es necesario trabajar con aplicaciones clusterizadas, múltiples *datacenters*, redes privadas, públicas y nubes híbridas, como también aplicaciones con distintas dependencias. Este tipo de tareas ya no pueden realizarse individualmente para cada máquina y menos aún en entornos escalables donde se pueden llegar a tener cientos de nodos. Para ello es necesario contar con herramientas profesionales para orquestar éstas tareas complejas, de manera sencilla.

Hablar de orquestación implica eficiencia en los recursos, tanto humanos como computacionales, y por ello implica hablar de virtualización, aprovisionamiento y *datacenters* dinámicos. En este sentido, virtualización puede aplicar a computadoras, sistemas operativos, dispositivos de almacenamiento, aplicaciones o redes, sin embargo la virtualización de servidores es el objetivo principal en este proyecto. Las limitaciones que imponen los servidores x86, que están diseñados para correr un solo sistema operativo a la vez, resultan en una gran ineficiencia pudiendo alcanzar entre el cinco y quince porciento de su capacidad. Ésto desafía a los administradores a utilizar software para simular la existencia de hardware y crear un sistema de cómputo virtual, permitiendo ejecutar más de una máquina virtual y múltiples sistemas operativos en un mismo servidor, mejorando el uso de los recursos disponibles y reduciendo costos.

Índice

1. Introducción	7
1.1. Descripción	7
1.2. Objetivos	8
1.3. Intereses personales	8
1.4. Intereses institucionales	8
1.5. Metodología	9
1.6. Requerimientos	9
1.6.1. Casos de uso	10
1.6.2. Diagrama de secuencia	11
1.7. Cronograma a seguir	13
1.8. Objetivo a alcanzar en cada etapa:	13
1.9. Gestión de riesgos	13
1.9.1. Identificación de riesgos	14
1.9.2. Análisis de riesgos	15
1.9.3. Planificación de riesgos	16
2. Marco teórico	19
2.1. Sistema operativo	19
2.1.1. CentOS	19
2.1.2. Debian	20
2.1.3. FreeBSD	21
2.1.4. Solaris	22
2.2. Virtualización	23
2.2.1. Tipos de virtualización	23
2.2.2. Herramientas de virtualización	25
2.3. Aprovisionamiento	27
2.3.1. Herramientas de aprovisionamiento	28
2.4. Orquestación	30
2.4.1. Herramientas de orquestación	31
2.5. Protocolo PXE	33
2.5.1. PXE APIs	34
2.5.2. Funcionamiento de PXE	34
3. Desarrollo	37
3.1. Elección de la plataforma	37
3.2. Arquitectura de desarrollo	38
3.3. KVM/Qemu	39
3.3.1. Arquitectura	39
3.3.2. Requerimientos de hardware	40
3.3.3. Limitaciones de KVM	41
3.3.4. Configuración de la red	42
3.3.5. Creación del sistema de virtualización KVM/Qemu	42
3.4. Cobbler	45
3.4.1. Tópicos generales de Cobbler	45
3.4.2. Sistema de aprovisionamiento y automatización Cobbler	51
3.4.3. Repositorio local separado de Cobbler	55

3.4.4. Importar imágenes ISO al servidor Cobbler	57
3.5. Puppet	57
3.5.1. Tópicos generales de Puppet	58
3.5.2. Creación del sistema de orquestación Puppet.	64
3.6. Windows 7	68
3.6.1. El lado de Windows	69
3.6.2. SAMBA	71
3.6.3. Creación de la imagen de instalación Windows 7	73
3.6.4. Capturar el equipo de referencia	77
3.6.5. El lado de Linux	93
3.7. Interfaz Web	95
3.8. Validación de las Herramientas	98
4. Conclusión	102
5. Trabajos Futuros	102
6. Bibliografía	103

Índice de tablas

1.	Importancia de los Requerimientos	10
2.	Riesgos del proyecto	15
3.	Matriz de importancia de los riesgos	15
4.	Implicación de los riesgos	16
5.	Planificación de riesgos	17
6.	Pasos automatización Windows 7	73
7.	Casos de prueba del usuario del sistema - Parte 1	99
8.	Casos de prueba del usuario del sistema - Parte 2	100
9.	Casos de prueba del usuario del sistema - Parte 3	101

Índice de figuras

1.	Caso de uso en un mismo servidor	11
2.	Servidor de máquinas virtuales separado	11
3.	Creación de VMs	12
4.	Modificar políticas de orquestación	12
5.	CentOS logo	19
6.	Debian logo	20
7.	FreeBSD logo	21
8.	Solaris logo	22
9.	Diagrama de virtualización completa	24
10.	Diagrama de para-virtualización	24
11.	Diagrama de virtualización a nivel de sistema operativo	25
12.	Docker logo	25
13.	KVM/Qemu logo	26
14.	OpenVZ logo	27
15.	VirtualBox logo	27
16.	Cobbler logo	28
17.	FAI logo	29
18.	Foreman logo	29
19.	Vagrant logo	30
20.	Ansible logo	31
21.	Chef logo	32
22.	Puppet logo	33
23.	APIs de PXE [7]	34
24.	Proceso de PXE [7]	36
25.	Arquitectura de desarrollo	39
26.	Arquitectura de desarrollo en un entorno mixto.	39
27.	Diagrama de arquitectura de KVM/Qemu	40
28.	Detalles de conexión	43
29.	Crear red virtual	44
30.	Modelado de Cobbler	45
31.	Cobbler Web	55
32.	Paso de archivos a Windows	75
33.	Paso de archivos a Windows	75

34. Paso de archivos a Windows	76
35. Modo auditoría	76
36. Ejecución sysprep	77
37. Carga WinPE.iso para extraer imagen	78
38. Prioridad de inicio en máquina virtual	78
39. Verificación de letras de unidad	79
40. ImageX capturando imagen	80
41. Windows System Image Manager	81
42. Idioma y teclado.	82
43. Idioma.	82
44. Firewall activado por defecto.	83
45. Discos.	83
46. Configuración discos. Disco 0.	84
47. Discos. Particionamiento.	84
48. Discos. Particionamiento.	85
49. Discos. Particionamiento.	85
50. Discos. Particionamiento.	86
51. Instalación del sistema operativo.	86
52. Instalación del sistema operativo. Elección de la versión.	87
53. Elección de partición de instalación.	87
54. Aceptación de las condiciones de privacidad y uso.	88
55. Clave de producto.	88
56. Activación del producto.	89
57. Hostname, organización, propietario y zona horaria.	89
58. Idioma, teclado, regional.	90
59. Organización, propietario y zona horaria.	90
60. Configuración de red.	91
61. Contraseña administrador.	91
62. Cuentas de usuario, administrador.	92
63. Cuentas de usuario, invitado.	92
64. Método get	95
65. Método post	96
66. Método run	96
67. Crear virtual_machine	97
68. Crear virtual_machine_parametrizada	97
69. Editar políticas/servicios	98
70. Estado de las máquinas virtuales	98

1. Introducción

Hoy en día no se concibe la implementación de un solo servidor con un servicio. Múltiples servicios virtuales dan paso a mejoras significativas en la producción, aplicando técnicas de balanceo de carga, servidores redundantes, etc. Por ello es necesario trabajar con aplicaciones clusterizadas, múltiples *datacenters*, redes privadas, públicas y nubes híbridas, como también aplicaciones con distintas dependencias. En sectores con cientos de nodos administrables, alrededor de quinientos o más, esta tarea debe realizarse por una persona que cuente con los conocimientos técnicos y los procesos de producción, como también los métodos para asegurar la estabilidad y continuidad del sistema. Para ello es necesario contar con herramientas profesionales para orquestar éstas tareas complejas de manera sencilla. En el área de cómputos, la orquestación trata de alinear los requerimientos de las organizaciones con las aplicaciones, datos e infraestructura. Define las políticas y los niveles de servicios a través de flujos de trabajo automatizados, aprovisionamiento y administración de cambios. Ésto crea una infraestructura que puede escalar basándose en las necesidades de cada aplicación.

La orquestación reduce el tiempo y el esfuerzo necesario para desplegar múltiples instancias de una sola aplicación. A medida que se requieren más recursos, las herramientas de automatización pueden ejecutar tareas que, previamente, sólo podían realizarse por múltiples administradores cada uno en su estación de trabajo física.

De la mano de la orquestación, se encuentra la virtualización de computadoras, sistemas operativos, dispositivos de almacenamiento, aplicaciones o redes, entre otros. Las limitaciones que imponen los servidores x86, que están diseñados para correr un solo sistema operativo a la vez, resultan en una gran ineficiencia, alcanzando entre el cinco y quince por-ciento de su capacidad. Ésto desafía a los administradores a utilizar software para simular la existencia de hardware y crear un sistema de cómputo virtual, permitiendo ejecutar más de una máquina virtual y múltiples sistemas operativos en un mismo servidor, mejorando la eficiencia del mismo y reduciendo costos.

Cloud Computing o nube computacional no es lo mismo que virtualización, internamente en esta arquitectura puede existir la virtualización. Describe el envío de recursos computacionales compartidos (software o datos) bajo demanda a través de Internet. Se esté o no en la nube, se puede comenzar virtualizando los servidores locales y luego moverlos a ella para obtener aún más agilidad. Así, la orquestación es crucial al momento de desplegar servicios en la nube porque éstos:

- Están pensados para escalar arbitraria y dinámicamente, sin requerir intervención humana directa para hacerlo.
- Forman parte de flujos de trabajo en varios dominios técnicos y de negocio.

La virtualización se destaca también apoyando la innovación a través del uso de entornos virtuales para desarrollar, practicar y aprender. Además, se puede establecer entornos únicos de software para el aprendizaje sin demandar el uso exclusivo de recursos de hardware.

1.1. Descripción

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 de forma au-

tomática múltiples máquinas virtuales con escasa interacción humana, posibilitando 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 de datos.

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.

1.2. Objetivos

Principal:

- Desarrollar un sistema que integre diferentes tecnologías para implementar las técnicas de orquestación, virtualización y configuración automática para facilitar la gestión de servidores virtuales y sus servicios asociados.

Secundarios:

- Instalar y utilizar sistemas operativos para servidor.
- Emplear herramientas de virtualización.
- Usar herramientas de aprovisionamiento.
- Utilizar herramientas de orquestación.
- Analizar protocolos para inicio a través de la red.

Antecedentes de Proyectos similares

- No hay antecedentes en este tema.

1.3. Intereses personales

La principal motivación en este Proyecto Integrador es realizar un proyecto que nos permita aplicar los conocimientos obtenidos a lo largo de la carrera de grado de Ingeniería en Computación. Abordamos temas como sistemas operativos, sistemas de archivos, protocolos, metodologías de desarrollo, redes de datos, programación en diferentes lenguajes, 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. Una posible implementación del proyecto es desarrollar un sistema de infraestructura con automatización y orquestación que permita disminuir la carga de trabajo de los administradores de éstas aulas y facilitar las tareas de mantenimiento de las aulas, cumpliendo con las políticas del área que las administra.

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 se denomina 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 quienes también se obtienen los requerimientos. Esto permite una buena retro-alimentación, con la cual se devuelven correcciones.

El sistema incluye desarrollo en los lenguaje de programación Python, Bash y el lenguaje descriptivo propio de la herramienta de orquestación. El servidor de aprovisionamiento será el encargado de atender las peticiones de los dispositivos para su instalación.

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 Co-director. Se separaron los requerimientos en funcionales y no funcionales.

Durante la etapa de obtención de los mismos, se les asignaron niveles de importancia, definiendo cuáles son más necesarios para el cumplimiento del objetivo del presente trabajo.

De este modo, se han definido cinco niveles de importancia, siendo el número 5 el que representa la mayor importancia, disminuyendo sucesivamente hasta el nivel 1, el menor nivel de importancia.

ID	Requerimiento	Importancia
RF01	La herramienta debe poder aprovisionar máquinas virtuales.	5
RF02	La herramienta debe poder aprovisionar distintos sistemas operativos.	5
RF03	La herramienta debe poder aprovisionar software y configuraciones a las máquinas.	5
RF04	La herramienta debe poder aprovisionar software y configuraciones utilizando plantillas.	3
RF05	La herramienta debe poder aprovisionar utilizando repositorios locales.	5
RF06	La herramienta debe poder actualizar los repositorios locales.	4
RF07	La herramienta debe poder setear políticas a las máquinas virtuales.	5
RF08	La herramienta debe ser escalable, integrar nuevas máquinas virtuales fácilmente.	5
RF09	La herramienta debe poseer una interfaz web que permita utilizar todas sus funcionalidades predeterminadas.	3
RF10	La herramienta debe permitir visualizar y modificar el estado de las máquinas (encendido o apagado) vía interfaz web.	3
RNF01	Las herramientas a integrar deben poseer licencias de uso libre.	5
RNF02	La herramienta debe estar implementada en una versión estable al momento de realizar el Proyecto Integrador.	4

Tabla 1: Importancia de los Requerimientos

1.6.1. Casos de uso

En las figuras 1 y 2, se ilustran los requerimientos más importantes.

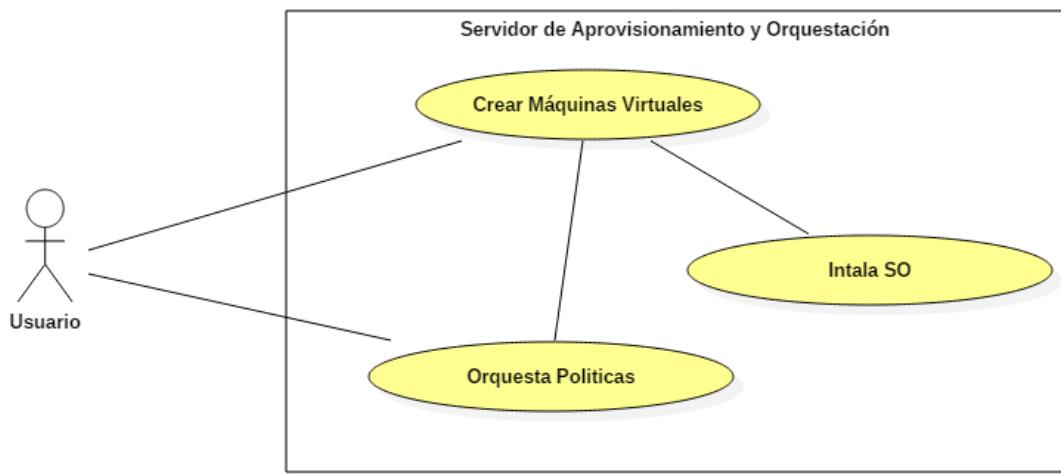


Figura 1: Caso de uso en un mismo servidor

También se puede dar el caso en el cual el servidor de máquinas virtuales se encuentre en otro servidor físico.

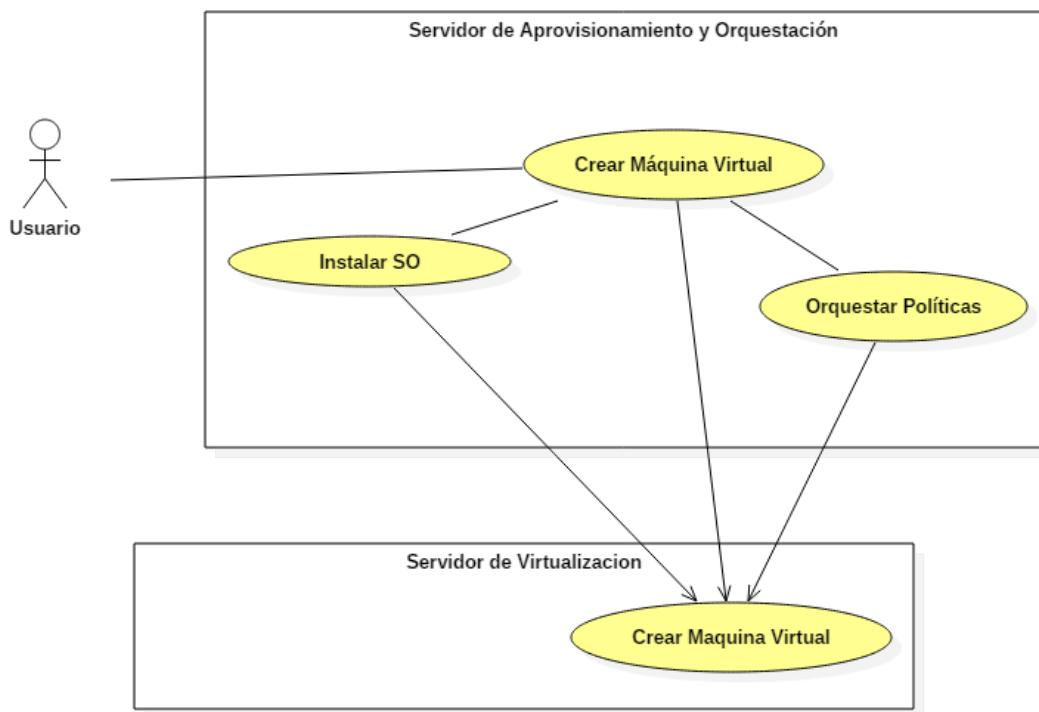


Figura 2: Servidor de máquinas virtuales separado

1.6.2. Diagrama de secuencia

En las figuras 3 y 4, se ilustran las secuencias de los requerimientos más importantes.

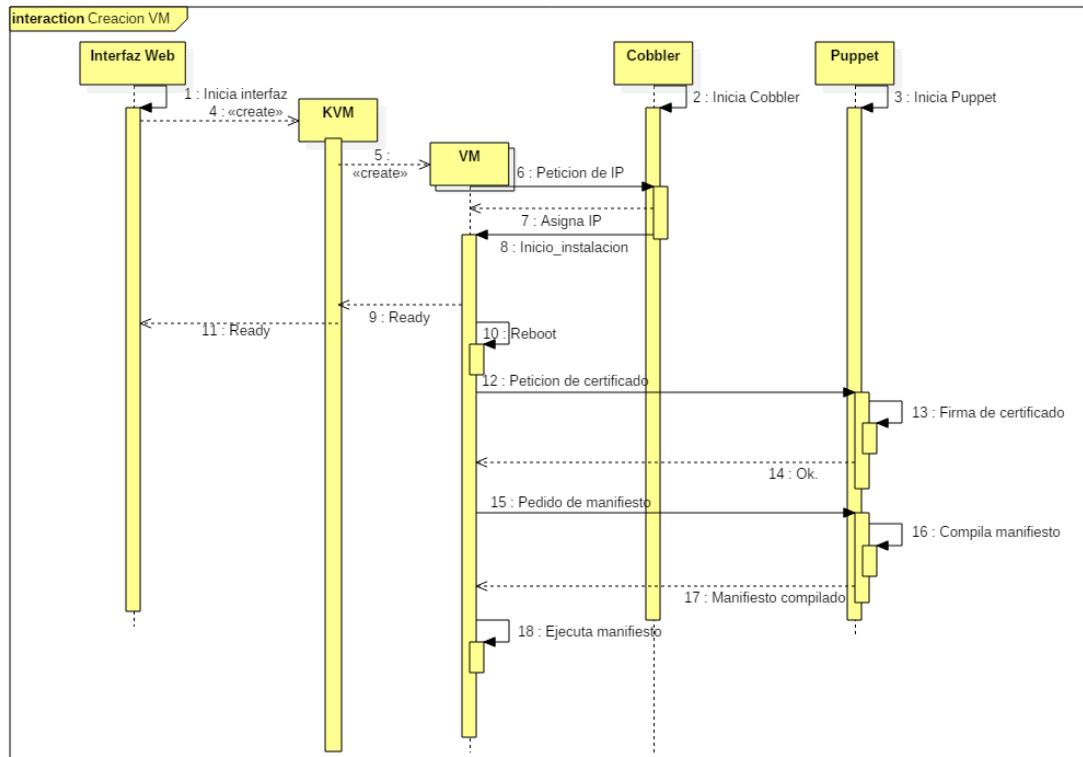


Figura 3: Creación de VMs

El siguiente diagrama ilustra la secuencia seguida al orquestar los diferentes nodos.

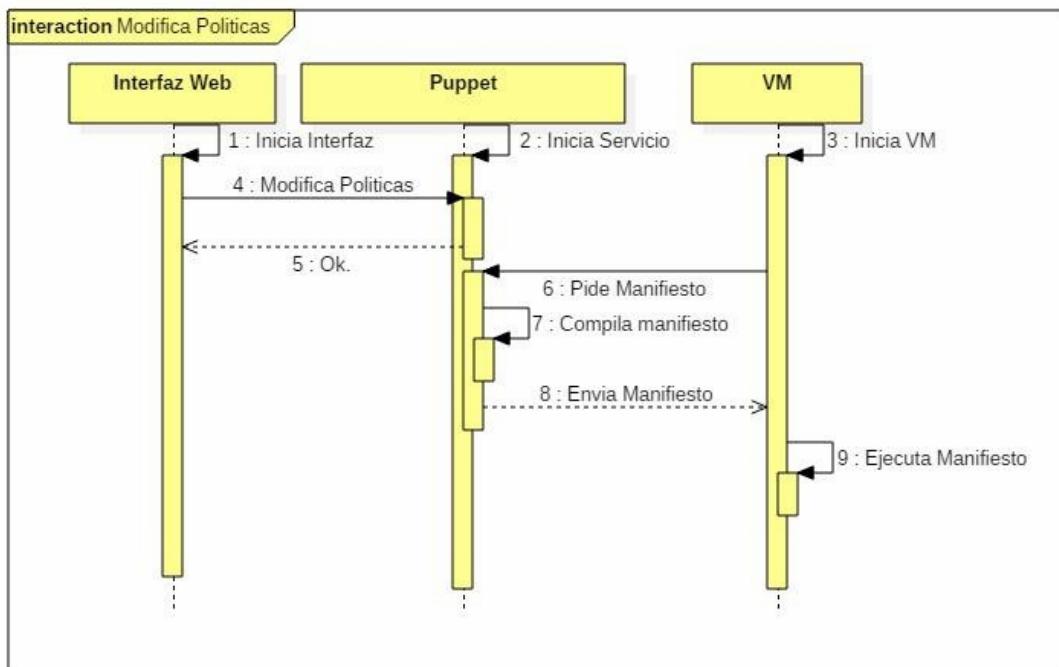


Figura 4: Modificar políticas de orquestación

1.7. Cronograma a seguir

El cronograma está dividido en siete etapas diferentes:

- Investigar y seleccionar el sistema operativo base.
- Investigar y seleccionar herramientas de virtualización.
- Investigar y seleccionar herramientas de aprovisionamiento.
- Investigar y seleccionar herramientas de orquestación.
- Desarrollo de un sistema que integre las herramientas seleccionadas para que trabajen en conjunto.
- Desarrollo de perfiles que permitan obtener un abanico de posibilidad para la creación de máquinas virtuales.
- Desarrollo de una interfaz web simple para una administración base del sistema.

1.8. Objetivo a alcanzar en cada etapa:

- **Primera etapa:** Conocer diferentes sistemas operativos y seleccionar el más adecuado para el desarrollo de sistema.
- **Segunda etapa:** Conocer diferentes herramientas de virtualización y seleccionar la más adecuada.
- **Tercera etapa:** Conocer cuáles son las herramientas disponibles y utilizadas en ambientes de producción para el aprovisionamiento de máquinas virtuales, teniendo en cuenta que deben ser de uso libre, analizarlas y elegir la más apropiada para realizar el Proyecto Integrador.
- **Cuarta etapa:** Conocer las herramientas de uso libre de orquestación que se encuentran en el mercado seleccionar la adecuada.
- **Quinta etapa:** Implementar las herramientas seleccionadas en conjunto.
- **Sexta etapa:** Realizar la implementación conjunta de todas las herramientas automatizando la instalación y realizar configuraciones pertinentes en los sistemas ya instalados por medio de *scripts* y la herramienta de orquestación, obteniendo diferentes perfiles.
- **Séptima etapa:** Desarrollo de una interfaz web simple que permita crear máquinas virtuales y asignarles el perfil deseado de manera fácil y sin necesidad de conocimientos técnicos.

1.9. Gestión de riesgos

Para la gestión de riesgos de nuestro proyecto, introducimos un pequeño resumen basado en la bibliografía.

Los riesgos se pueden definir como una probabilidad de que una circunstancia adversa ocurra. La gestión de riesgos permite identificar los riesgos de un proyecto y crear planes para minimizar su efecto sobre el proyecto.

Dado la falta de experiencia de los desarrolladores del proyecto en esta temática, la gestión de riesgos es importante dado a las incertidumbres inherentes con las que se encuentran. Por ejemplo: requerimientos ambiguos, dificultades en las estimaciones de los tiempos y recursos, etc.

El proceso de gestión de riesgos comprende varias etapas:

1. **Identificación de riesgos:** Consiste en identificar los posibles riesgos para el proyecto, el producto y los negocios.
2. **Análisis de riesgos:** Consiste en evaluar las probabilidades y las consecuencias de los riesgos.
3. **Planificación de los riesgos:** Se crean planes para abordar los riesgos.
4. **Supervisión de riesgos:** Proceso continuo en el que se evalúa cada riesgo y se analiza si han cambiado sus probabilidades o efectos.

1.9.1. Identificación de riesgos

En esta etapa debemos descubrir los posibles riesgos del proyecto. Clasificaremos los riesgos con los siguientes tipos:

- **Riesgos de tecnología:** Derivan de las tecnologías de software o hardware utilizadas en el sistema que se está desarrollando.
- **Riesgos organizacionales:** Derivan del entorno organizacional donde el software se está desarrollando.
- **Riesgos de estimación:** Derivan de los estimados administrativos, de las características del sistema y los recursos requeridos para construir dicho sistema.

La tabla 2 muestra los riesgos encontrados.

ID	Riesgo	Tipo
RS01	Falta de infraestructura para llevar a cabo el proyecto	Organizacional
RS02	Incompatibilidad de funciones del sistema operativo con el hardware disponible	Tecnológicas
RS03	Dificultades en la interacción entre la aplicación y el sistema operativo	Tecnológicas
RS04	Diferencias de paquetes disponibles entre los sistemas operativos	Tecnológicas
RS05	Diferencias entre funciones disponibles entre los sistemas operativos	Tecnológicas
RS06	Disponibilidad de información sobre las herramientas	Tecnológicas
RS07	Miembro del grupo de desarrollo abandona el proyecto	Organizacional
RS08	Conflictos por el tiempo dedicado a otros trabajos/proyectos	Organizacional
RS09	Cambios de requerimientos que impliquen modificar lo ya realizado	Organizacional
RS10	Tiempos de aprendizaje o de desarrollo subestimados	Estimación
RS11	Compatibilidad de las herramientas con los distintos sistemas operativos	Tecnológicas

Tabla 2: Riesgos del proyecto

1.9.2. Análisis de riesgos

En esta etapa, consideramos cada riesgo por separado para analizar las probabilidades de que se concreten, además de la severidad del efecto que producen. Luego de este análisis, se evalúa la importancia de cada riesgo con el objetivo de decidir cuáles son los riesgos que deben tenerse en cuenta durante el desarrollo del proyecto. Esta importancia, depende de las probabilidades y del efecto asignado a cada riesgo. No existe una metodología definida para realizar este proceso, por lo que depende en gran medida de la opinión y de la experiencia previa de quien realice el análisis.

En nuestro caso, definimos tres niveles de probabilidad (alta, moderada, baja) y tres niveles de seriedad del efecto (grave, tolerable, insignificante) que producen los riesgos. Cada combinación de probabilidad y efecto se traduce en un nivel de importancia del requerimiento.

Probabilidad /Impacto	Insignificante	Tolerable	Grave
Baja	Baja	Baja	Alta
Moderada	Baja	Moderada	Alta
Alta	Moderada	Alta	Alta

Tabla 3: Matriz de importancia de los riesgos

ID	Riesgo	Probabilidad	Impacto	Importancia
RS01	Falta de infraestructura para llevar a cabo el proyecto	Baja	Tolerable	Baja
RS02	Incompatibilidad de funciones del sistema operativo con el hardware disponible	Baja	Grave	Alta
RS03	Dificultades en la interacción entre la aplicación y el sistema operativo	Baja	Grave	Alta
RS04	Diferencias de paquetes disponibles entre los sistemas operativos	Moderada	Tolerable	Moderada
RS05	Diferencias entre funciones disponibles entre los sistemas operativos	Baja	Tolerable	Baja
RS06	Disponibilidad de información sobre las herramientas	Moderada	Tolerable	Moderada
RS07	Miembro del grupo de desarrollo abandona el proyecto	Baja	Grave	Alta
RS08	Conflictos por el tiempo dedicado a otros trabajos/proyectos	Moderada	Tolerable	Moderada
RS09	Cambios de requerimientos que impliquen modificar lo ya realizado	Baja	Tolerable	Baja
RS10	Tiempos de aprendizaje o de desarrollo subestimados	Moderada	Tolerable	Moderada
RS11	Compatibilidad de las herramientas con los distintos sistemas operativos	Moderada	Alta	Alta

Tabla 4: Implicación de los riesgos

1.9.3. Planificación de riesgos

Una vez que analizamos los riesgos, podemos interpretar cuáles son los más importantes, para tenerlos en cuenta durante el resto del desarrollo del proyecto. Debemos plantear las estrategias que nos permitirán gestionar esos riesgos. Tampoco existe una metodología definida para llevar a cabo este paso.

A continuación describiremos estrategias para cada uno de los riesgos.

ID	Riesgo	Consecuencias	Solución	Mitigación
RS01	Falta de infraestructura para llevar a cabo el proyecto	Complicaciones para desarrollar, realizar pruebas y puestas en común entre los integrantes del proyecto y los directores	Encontrar una nueva infraestructura donde realizar el proyecto	Planificación del uso de la infraestructura
RS02	Incompatibilidad de funciones del sistema operativo con el hardware disponible	Dificultades para integrar varios sistemas operativos al desarrollo	Conseguir hardware compatible	Desarrollar la parte del proyecto que tenga que ver con la funcionalidad en conflicto en un hardware compatible
RS03	Dificultades en la interacción entre la aplicación y el sistema operativo	Dificultades para la integración entre las herramientas y los sistemas operativos	Utilizar Aplicaciones y herramientas desarrolladas para dicho sistema operativo	Utilizar Aplicaciones y sistemas operativos compatibles
RS04	Diferencias de paquetes disponibles entre los sistemas operativos	Mayor dificultad para que la aplicación soporte varios sistemas operativos	Eliminar del desarrollo la funcionalidad relacionada con el paquete	Buscar paquetes con funcionalidades similares y adaptar el desarrollo
RS05	Diferencias entre funciones disponibles entre los sistemas operativos	Mayor dificultad para que la aplicación soporte varios sistemas operativos	Si la funcionalidad es importante para el desarrollo, eliminar la compatibilidad con el sistema operativo o eliminar la función del proyecto	Buscar funcionalidades similares o hacer que el proyecto presente opciones diferentes para cada sistema operativo
RS06	Disponibilidad de información sobre las herramientas	Mayor tiempo de aprendizaje sobre las herramientas	Utilizar herramientas que dispongan de una documentación más completa	Analizar si la información que se brinda es suficiente para utilizarla en el proyecto
RS07	Miembro del grupo de desarrollo abandona el proyecto	Extensión de los tiempos de desarrollo, posibles cambios de requerimientos	Eliminar objetivos secundarios o requerimientos para acortar el tiempo de desarrollo	Revisar los requerimientos para limitarse a los esenciales
RS08	Conflictos por el tiempo dedicado a otros trabajos/proyectos	Extensión de los tiempos de desarrollo	Dedicación de horas semanales exclusivas al proyecto	Revisar los requerimientos para eliminar los pocos importantes
RS09	Cambios de requerimientos que impliquen modificar lo ya realizado	Extensión en los tiempos de desarrollo	Realizar correcciones en la planificación y en las metas para cumplir con los requerimientos sin un aumento de los tiempos de desarrollo	Analizar la necesidad de los nuevos requerimientos y enfocarse en los principales
RS10	Tiempos de aprendizaje o de desarrollo subestimados	Extensión en los tiempos de desarrollo.	Cambiar requerimientos para disminuir el tiempo requerido para completar el proyecto	Aumentar la dedicación de tiempo al proyecto
RS11	Compatibilidad de las herramientas con los distintos sistemas operativos	Impedimento en la continuación del proyecto	Cambiar las herramientas incompatibles	Utilización de herramientas específicas para cada sistema operativo

Tabla 5: Planificación de riesgos

2. Marco teórico

Previamente al abordaje del desarrollo, se requieren conocimientos teóricos que 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 ellas fueron 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. Se puede dividir esta sección del informe en áreas diferentes:

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

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

2.1. Sistema operativo

2.1.1. CentOS



Figura 5: 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 es binariamente compatible en su totalidad. Cada versión de esta distribución 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 distribución es que se obtiene un conjunto estable de paquetes que por lo general sólo incluyen correcciones de errores. CentOS 7 sólo 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 TB. 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 la adoptaron 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.

2.1.2. Debian



Figura 6: 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. éstas se liberan cuando estén realmente listas. Es una distribución ampliamente utilizada.

Debian 8 Jessie, tiene soporte para las siguientes arquitecturas: 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), incluidos para versiones anteriores, no son parte de esta versión.
- Soporte UEFI para amd64, i386 y arm64.

2.1.3. FreeBSD



Figura 7: FreeBSD logo

FreeBSD es un sistema operativo basado en Unix 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 pre-compilados de la aplicación y se pueden administrar 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 puerto 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 multi-procesamiento simétrico con múltiples CPUs.

2.1.4. Solaris



Figura 8: 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 se desarrolló 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 multi-procesamiento simétrico (SMP), soportando un gran número de CPUs. Sun desarrolló el sistema operativo Solaris poniendo énfasis en su integración con la plataforma de hardware SPARC. Ambos productos se ofrecían 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, como también en sistemas de 32 bits. Esta versión añadió soporte de paravirtualización cuando se utiliza 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

2.2. 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. Posibilita el establecimiento de múltiples entornos virtuales sobre una plataforma de hardware compartida. Estos entornos pueden crearse localmente o aprovisionados externamente. La virtualización se destaca también apoyando la innovación a través del uso de entornos virtuales para practicar y aprender. 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, memoria RAM y 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 entre todos los sistemas virtualizados. Si se rompe éste, se rompen todas las máquinas virtuales.

En resumen, una máquina virtual (*virtual machine* - VM) es un conjunto de software altamente aislado con un sistema operativo y aplicaciones adentro. Cada VM es completamente independiente. Poner múltiples VMs en una misma computadora habilita a varias aplicaciones y sistemas operativos correr en un solo servidor virtual o *host*. Una fina capa de software, llamado *hypervisor*, desacopla las máquinas virtuales del *host* y asigna dinámicamente recursos computacionales a cada VM.

2.2.1. Tipos de virtualización

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.

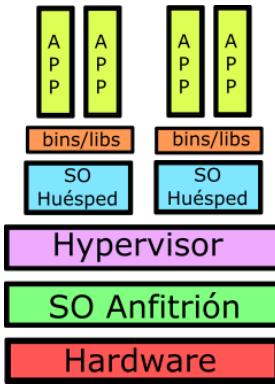


Figura 9: Diagrama de virtualización completa

Para-virtualización Para-virtualización es una técnica de virtualización que implica ejecutar versiones modificadas de los sistemas operativos. El sistema operativo para-virtualizado se encuentra modificado para que se de cuenta 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.

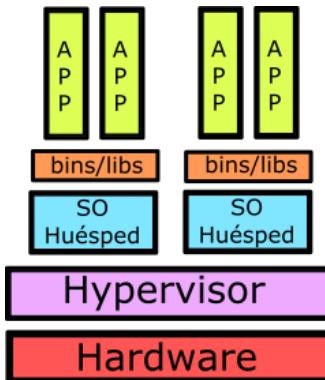


Figura 10: Diagrama de para-virtualización

Para-virtualización de drivers La para-virtualización y la virtualización completa pueden combinarse 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.

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 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 (en este caso la variante Debian/KFreeBSD puede correr en jaulas Warden sobre FreeBSD), OpenVZ, Docker o Linux-Vserver usan esta técnica.

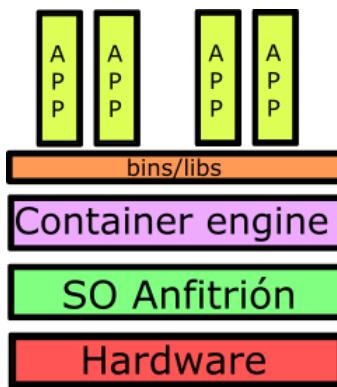


Figura 11: Diagrama de virtualización a nivel de sistema operativo

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. Como ejemplo de emulador el principal referente es Qemu.

2.2.2. Herramientas de virtualización

Algunas de las herramientas más utilizadas para virtualizar son las siguientes:

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

Como uno de los requisitos es utilizar herramientas de código abierto, no se indagó acerca de VMWare.

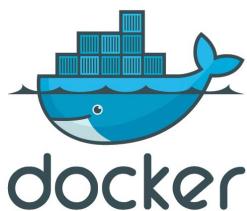


Figura 12: Docker logo

Docker Docker es una 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). Un contenedor Docker no incluye todo un sistema completo, sino únicamente aquellas librerías, archivos y configuraciones necesarias para desplegar las funcionalidades que contenga. Así mismo, 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 sí sobre la misma máquina o sistema anfitrión.

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, Mac OS X y FreeBSD además de Linux.



Figura 13: KVM/Qemu logo

KVM/Qemu 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 puertos 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.



Figura 14: OpenVZ logo

OpenVZ 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.



Figura 15: VirtualBox logo

VirtualBox 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).

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 FreeBSD.

2.3. 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 llevan a cabo para que ésto 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 servicios necesarios.
- 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.

2.3.1. Herramientas de aprovisionamiento

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

- Cobbler
- FAI
- Foreman
- Vagrant



Figura 16: Cobbler logo

Cobbler 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 configurarse 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 distribuciones 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 integrar la herramienta de administración de configuración Puppet. Cobbler se basa en el mecanismo de *kickstart* y ofrece perfiles de instalación que pueden aplicarse a una o muchas máquinas. La información contenida en una plantilla *kickstart* puede modificarse 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*.



Figura 17: FAI logo

Fully Automatic Installation (FAI) 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 distribuciones de Linux como Suse, Red Hat, Solaris, ya sea 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.
- Recuperación 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.



Figura 18: Foreman logo

Foreman 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, aprovisionar 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.
- Revisar cambios históricos para auditoría y resolución de problemas.



Figura 19: Vagrant logo

Vagrant 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 éstas herramientas. Luego herramientas de administración de configuración como Ansible, Chef, Salt, y Puppet pueden utilizarse para instalar y configurar automáticamente el software en la máquina.

2.4. 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 escala 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 realizan tareas que previamente podían llevarse a cabo sólo por múltiples administradores.

Un escenario que se puede encontrar es 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 automatizarse, pero el conjunto de éstas 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 simplemente 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 involucra 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.

2.4.1. Herramientas de orquestación

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

- Ansible
- Chef
- Puppet



Figura 20: Ansible logo

Ansible 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 de máquinas: controladores y nodos. Primero, existe una única máquina de control donde la orquestación comienza. Los nodos se manejan 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 escribirse 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.



Figura 21: Chef logo

Chef 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” (configuraciones de sistemas que describen cómo están manejadas las aplicaciones) necesarias para configurar el nodo deben estar presentes en su disco duro. Esta herramienta utiliza Ruby-DLS para escribir las recetas. Éstas, que pueden agruparse para facilitar la administración, describen de forma secuencial una serie de recursos que deben estar en un estado particular.

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



Figura 22: Puppet logo

Puppet 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 de cómo 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 se ejecutan 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 administración, 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 *puppet-master* 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. Entonces, si el agente encuentra algún recurso que no se encuentre en el estado deseado, realiza los cambios necesarios para corregirlo. 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.

2.5. Protocolo PXE

El protocolo PXE (*Preboot Execution Environment*) es un estándar que les permite a computadoras, dentro de una red, que todavía no tienen instalado un sistema operati-

vo, configurarse e iniciarse 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.
- Al ser independiente del fabricante, nuevos tipos de computadoras pueden añadirse a la red.

2.5.1. 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 usarse en cualquier interfaz que soporte esta API.

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

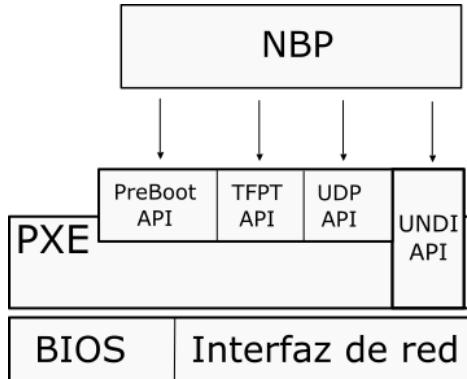


Figura 23: APIs de PXE [7]

2.5.2. Funcionamiento de PXE

En la figura 24 se puede observar el funcionamiento del protocolo 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 “PXEClient:Arch:xxxxx:UNDI:yyzzz”.
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 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 enviarse por *broadcast* al puerto 67. Este paquete es el mismo que el DHCPDISCOVER inicial en el paso uno, excepto que está 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 “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:
 - a) El nombre del archivo ejecutable.
 - b) Parámetros de configuración MTFTP.
 - c) Otras opciones necesarias para que el NBP pueda ejecutarse.
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.

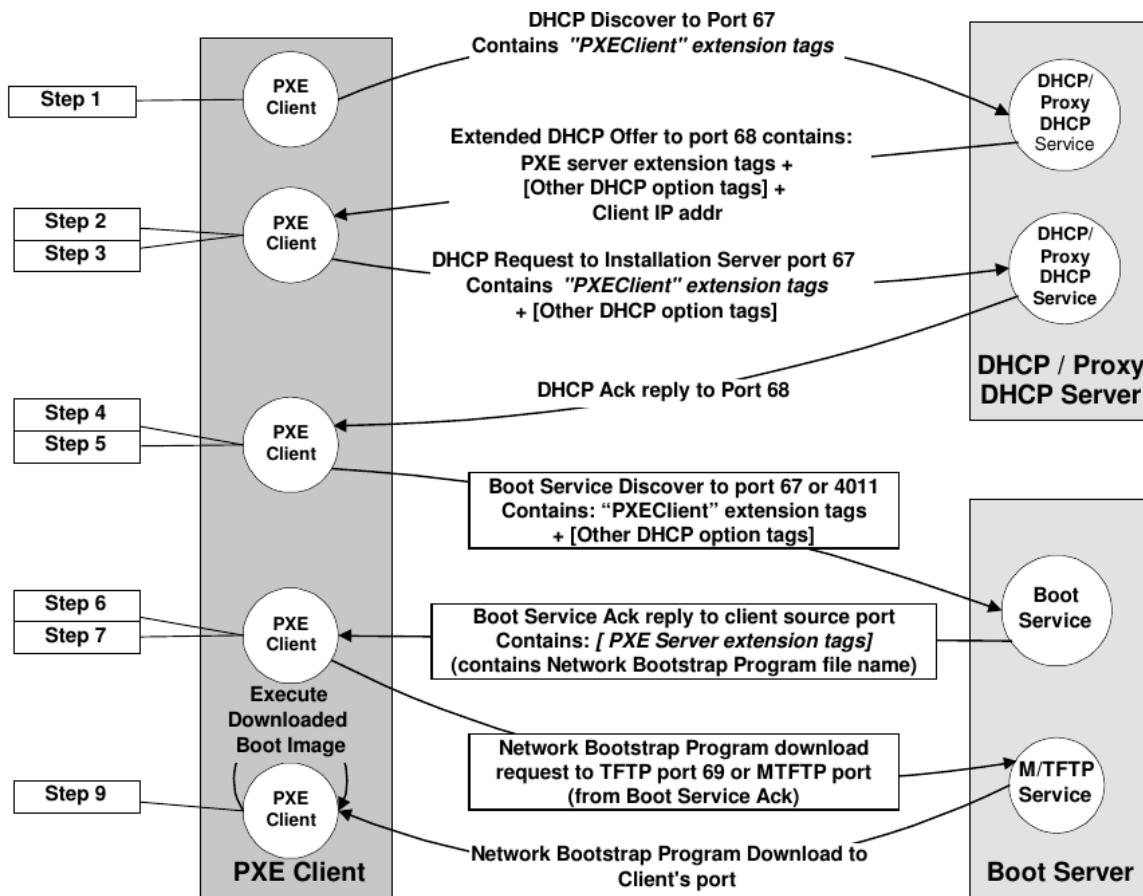


Figura 24: Proceso de PXE [7]

3. Desarrollo

Para facilitar la lectura y comprensión de las herramientas utilizadas, el presente informe no refleja las iteraciones del proyecto aún cuando éstas se implementaron acorde a la metodología empleada, realizando entregas iterativas.

Como caso particular, en la implementación se tomará un laboratorio informático de aprendizaje. Las funcionalidades desarrolladas permiten orquestar las políticas a seguir de las diferentes máquinas de la red al mismo tiempo que posibilitan a los administradores (o incluso a cualquier persona sin conocimiento técnico) crear máquinas virtuales con pocos *clicks*, dichas máquinas virtuales cuentan con todos los servicios y programas necesarios para los alumnos de las diferentes carreras. También se pueden crear máquinas virtuales con otros perfiles, por ejemplo, un perfil para docentes.

Algunas de las limitaciones del proyecto se presentan a continuación:

- Tanto SELinux como el *firewall* están desactivados.
- Debido a lo anterior, la interfaz web no cuenta con autenticación.
- A partir de CentOS 7, quedó descontinuado el paquete necesario para dar soporte a repositorios locales de Ubuntu sobre una distribución CentOS. Entonces, se necesita contar con una conexión a Internet para el despliegue de Ubuntu.
- Ubuntu 16.04 LTS y Windows 10 no tienen soporte de la herramienta de despliegue utilizada. A su vez, Windows 10 tampoco cuenta con soporte de la herramienta de orquestación.

3.1. 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.

Selección del sistema operativo base. Para seleccionar el sistema operativo base, se tomaron en cuenta los siguientes parámetros:

- Buena penetración de mercado: De esta manera, se obtendrá una experiencia útil a futuro.
- Sistema operativo estable y vigente.
- Buena integración del sistema operativo con las herramientas a utilizar.

Analizando estos aspectos, se optó por utilizar como sistema operativo base a CentOS 7 dado que es la última versión de este sistema operativo, es uno de los sistemas con más penetración en el mercado informático y cuenta con el respaldo de Red Hat Enterprise ya que es su versión libre.

También, se tuvo en cuenta la inclusión de *systemd* y que una gran cantidad de herramientas, tanto de virtualización como aprovisionamiento y orquestación, están diseñadas para Red Hat Enterprise y por ende CentOS.

Selección de la herramienta de virtualización. Para seleccionar la herramienta de virtualización, se tomaron en cuenta los siguientes parámetros:

- Penetración y uso en el mercado.
- Documentación y utilización en el ámbito universitario.
- Integración con el sistema operativo.

Teniendo en cuenta los puntos enunciados antes, el conjunto KVM/Qemu es la herramienta más apta dado que posee una amplia documentación y escalabilidad; a su vez está desarrollada para Red Hat Enterprise y se utiliza en ambientes de producción. Además, se usa en el Laboratorio de Computación de la FCEFyN.

Selección de la herramienta de aprovisionamiento. Se tuvo en cuenta para seleccionar esta herramienta:

- Penetración y uso en el mercado.
- Documentación y soporte.
- Integración con el sistema operativo.

La herramienta seleccionada fue Cobbler porque se mantiene continuamente actualizada, tiene soporte de la comunidad, posee una buena documentación y, por sobre todo, tiene una excelente integración con el sistema operativo seleccionado y con la herramienta de virtualización. También tiene interesantes funcionalidades como la posibilidad de crear diferentes perfiles de instalación (requisito funcional) y la capacidad de funcionar con un servidor DHCP externo a la aplicación.

Selección de la herramienta de Orquestación. Esta selección se basó en una recomendación de los directores del proyecto. Se optó por Puppet, dado que es una de las herramientas con más penetración de mercado y por ende brinda una muy buena experiencia a futuro. Pero además, como todas las herramientas seleccionadas, Puppet cuenta con dos versiones, *Enterprise* y *Open Source*, entonces siguiendo el requerimiento no funcional RNF01 se escogió la versión *Open Source*.

3.2. Arquitectura de desarrollo

La figura 25 representa la arquitectura de desarrollo utilizada para todas las pruebas.

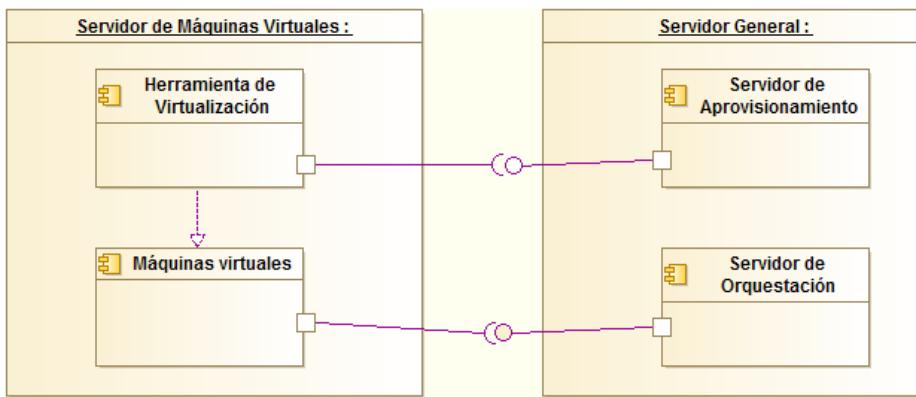


Figura 25: Arquitectura de desarrollo

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 la figura 26.

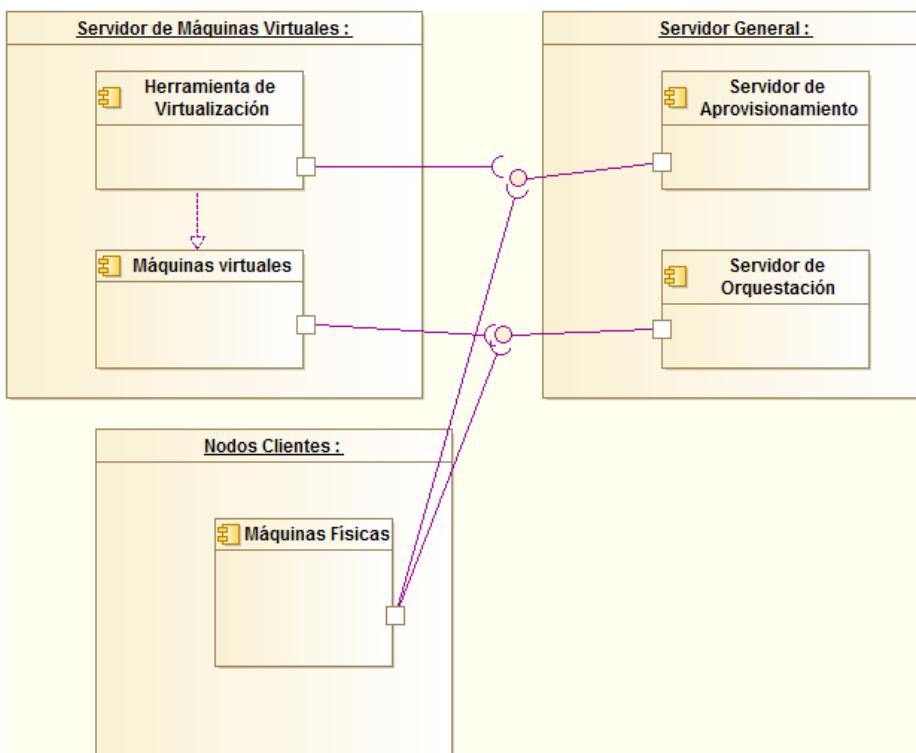


Figura 26: Arquitectura de desarrollo en un entorno mixto.

3.3. KVM/Qemu

3.3.1. Arquitectura

En este Proyecto se utiliza la técnica de virtualización completa. En particular, la arquitectura que utiliza KVM es la mostrada en la figura 27:

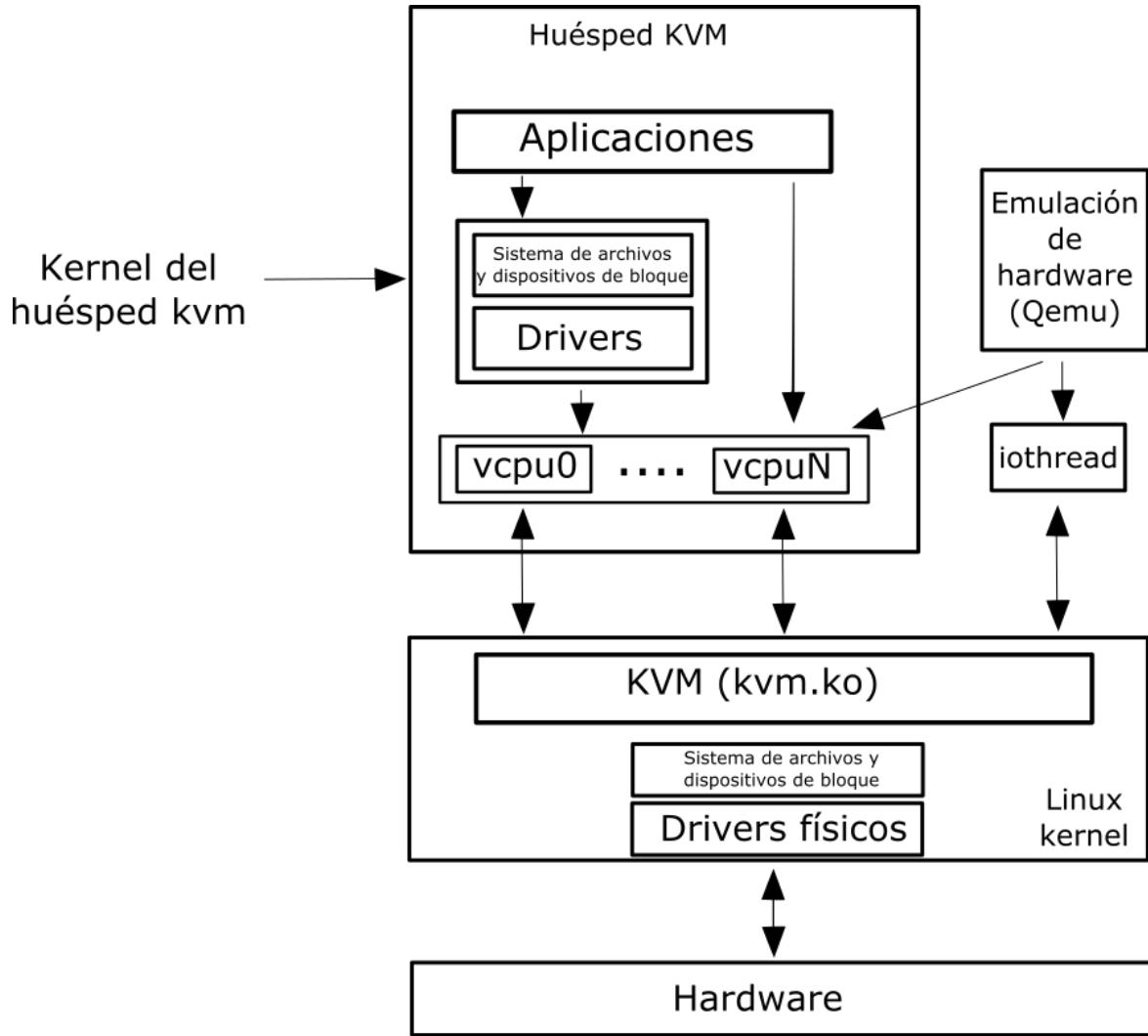


Figura 27: Diagrama de arquitectura de KVM/Qemu

3.3.2. Requerimientos de hardware

El *hypervisor* 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 banderas que contiene el procesador, entre ellas, la `svm` (AMD) o `vmx` (Intel). En caso de no poseer esas banderas, el procesador no soporta 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 dni monitor cx16 popcntlahf_lm cmp_legacy svm extapic
cr8_legacy abm sse4a 3dnowprefetch osvw ibs skinit wdt nodeid_msr
hw_pstate npt lbrv svm_lock nrip_save
```

Se debe asegurar que el módulo de KVM esté cargado, para ejecutar:

```
lsmod | grep kvm
```

La salida obtenida 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
```

3.3.3. Limitaciones de KVM

A medida que un servicio tecnológico se extiende aparecen formas de optimizar su implementación y su uso, buscando generalmente mayor eficiencia y rentabilidad. Una estrategia que existe para lograrlo es el *overcommit*. En términos de virtualización, el *overcommit* es el parámetro que indica qué cantidad de recursos virtuales se definen respecto los recursos de hardware que puede entregar un *host* físico. Hay dos parámetros que tienen una incidencia directa en el rendimiento de una aplicación: la memoria RAM y la CPU.

En el caso de la memoria RAM, hay *overcommit* si la suma de los GB de RAM de todas las VMs es mayor a los GB de RAM de que disponen los servidores físicos. Por lo tanto, en este caso, una VM no contaría con *n*GB de RAM real, sino que uso de dicho recurso que es compartido entre varias máquinas virtuales. Si la solución está bien diseñada, el *overcommit* es proporcional a la demanda de recursos y contempla picos de carga a nivel de plataforma, no tiene por qué generar conflictos. Además, permite reducir los costos gracias a esta especulación sobre la demanda.

En el caso del *overcommit* de procesador o CPU, dado que las capacidades de procesamiento que ofrecen actualmente los servidores físicos son muy altas, el *overcommit* sirve para aprovechar todas las prestaciones del *host*. Si bien la mayoría de los hypervisors permiten configurar muchos VCPUs (CPUs virtuales) por core físico, la realidad es que no es recomendable configurar más de 4 ó 5 VCPUs por core físico, ya que afectaría notablemente al rendimiento del servidor. Cuanto más *overcommit* tenga la plataforma más aumenta la posibilidad de cargas de CPU física cercanas al 100% que producirán solicitudes o tiempos de respuesta altos que inutilizarán la aplicación.

Para el *hypervisor* de KVM, las siguientes restricciones deben tenerse en cuenta al momento de planificar una solución:

- El número máximo de CPUs por huésped es grande (240 para RHE 7.1).
- La virtualización anidada no tiene soporte.
- El *overcommit* de memoria RAM tiene soporte utilizando el disco de *swap*.
- Para hacer *overcommit* de CPUs no se recomienda utilizar más de **10 VCPUs** por cada CPU física.
- La virtualización de dispositivos SCSI no está soportada.
- La virtualización de dispositivos IDE en KVM se limita a 4 por huésped.
- La asignación de dispositivos referenciados a dispositivos físicos es de uso exclusivo de la VM.
- La migración y guardado, o restauración de la VM, no tienen soporte mientras el dispositivo se encuentra en uso.
- KVM no soporta *kernels de real time*.



3.3.4. 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*).

3.3.5. Creación del sistema de virtualización KVM/Qemu

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

```
yum update
yum install -y kvm libvirt qemu-kvm virt-manager libvirt qemu-system-x86_64
qemu-img libvirt-python libvirt-client virt-install
```

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

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

Creación de una red NAT Aprovechando la interfaz gráfica virt-manager para crear la red, primero ir a:

1. Editar
2. Detalles de la conexión.
3. Redes virtuales.

Se abre el menú de la figura 28.

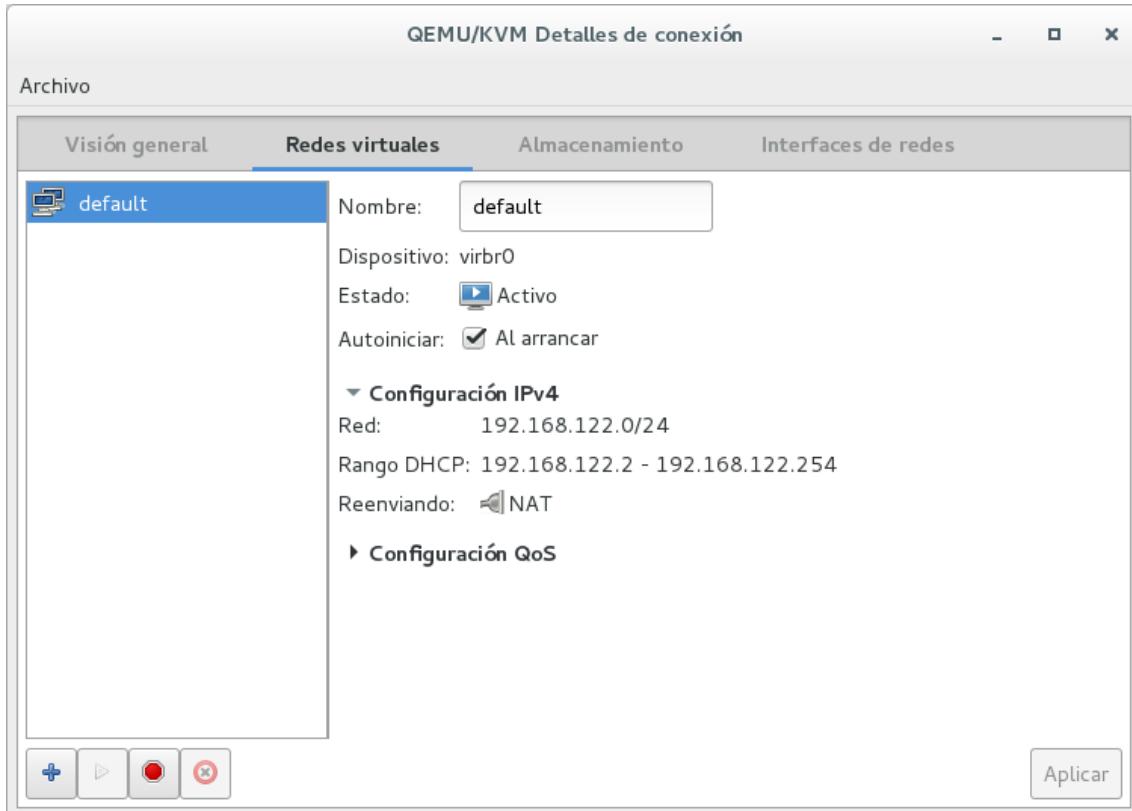


Figura 28: Detalles de conexión

Se añade una nueva red, haciendo *click* en el signo +. La ventana que aparece cuenta con cuatro etapas:

1. Se selecciona el nombre de la red, *puppet*.
2. Luego, se indica la red deseada (en este proyecto se usó 192.168.122.0/24) y se deshabilita el DHCP dado que se utilizará un servidor para esto.
3. Se encuentra la opción de habilitar IPv6.
4. Finalmente, se indica que la red debe ser NAT y se elige el dispositivo al cual se reenvía. Ver figura 29.

Creación de VMs Las máquinas virtuales se pueden crear tanto siguiendo la GUI como por línea de comandos. Este último método tiene la ventaja de poder utilizarse para *scripting*.

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

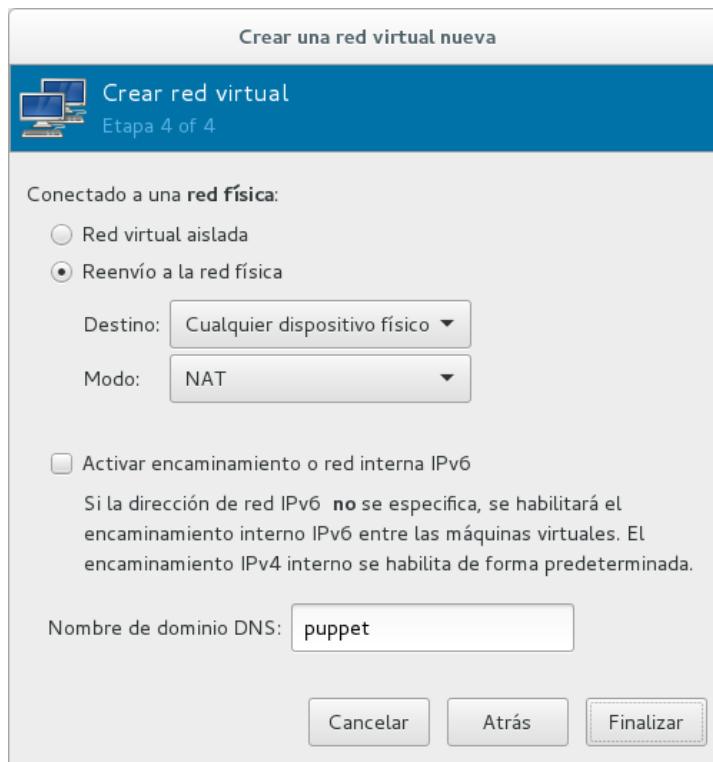


Figura 29: Crear red virtual

- **-connect:** Con esta opción se indica que se trabajará con el emulador qemu.
- **-virt-type:** Con esta opción se indica que la virtualización la realizará KVM.
- **-name:** Con esta opción se indica un nombre único para la VM. Este nombre no es del sistema operativo sino es el nombre para identificarlo dentro del campo de qemu-KVM.
- **-ram** Con esta opción se indica la cantidad de memoria RAM en MB que dispondrá la VM.
- **-disk:** Con esta opción se indica el *path* absoluto donde se encontrará la imagen creada y el tamaño en GB de la misma.
- **-network:** Si se trabaja con una red virtual de datos, con esta opción se indica a la VM a cuál red debe conectarse.
- **-pxe:** Esta opción es muy importante porque indica que el sistema operativo a instalar se obtiene a través de la red de datos.
- **-os-type y -os-variant:** Con esta opción se le indica a la VM qué tipo de sistema operativo contendrá (Linux, Windows, etc) y la variante del mismo (Centos, Debian, Ubuntu, Windows 8, Windows 7, etc.) para aumentar el rendimiento.

Un ejemplo utilizado en el desarrollo 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=puppet --pxe --os-type linux --os-variant rhel7
```

3.4. Cobbler

El servidor Cobbler debe contener los paquetes necesarios para poder instalar cada sistema operativo (CentOS, Ubuntu y Windows). Por lo cual, debe contar con un espacio en disco mínimo para soportar esto. Además, dado que el mismo servidor contiene el sistema de automatización y orquestación, se estima que debe contar con un mínimo de 30GB de disco.

El proyecto se desarrolló sin tener en cuenta SELinux ni *firewall*, por lo cual, ambos se deshabilitan de la siguiente forma.

Editar el archivo `/etc/sysconfig/selinux` y configurar:

```
SELINUX=disabled
```

En el caso del *firewall* ejecutar:

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

3.4.1. Tópicos generales de Cobbler

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.

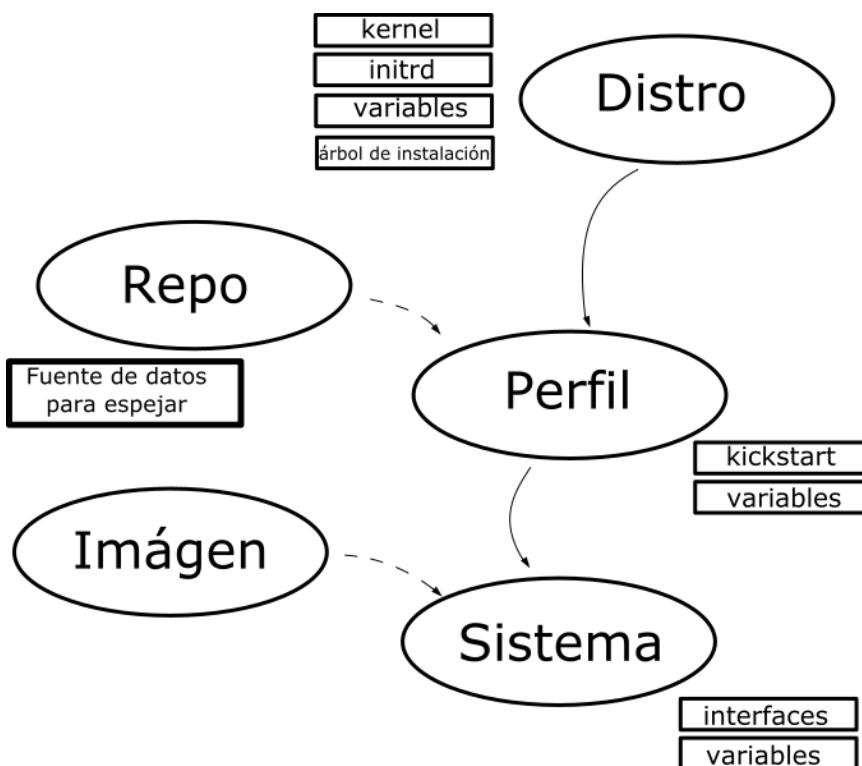


Figura 30: Modelado de Cobbler

Distros Distribución que se desea instalar. Importar el contenido de la distribución 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.

Profiles Un *profile* o perfil 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 servidor web o de escritorio.

Systems Las 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 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 la misma.

Images Cobbler puede arrancar 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 cómo instalarlas con Cobbler. Por ejemplo, trata la instalación de sistemas operativos Windows usando qemu/KVM:

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

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

Repositorios Espejar repositorios le permite a Cobbler espejar el árbol de instalación (`cobbler import`) y también paquetes opcionales. Si se espeja todo esto localmente en la red, las instalaciones y actualizaciones serán más rápidas (usualmente es válido realizar esto para grandes configuraciones en *datacenters*, laboratorios, etc). Si un *profile* tiene un repositorio dado, este repositorio puede configurarse automáticamente 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. Esta función sólo funciona para repositorios http o ftp.

Los repositorios pueden crearse del siguiente modo:

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

Donde se tiene:

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 se usa para guardar la ubicación del espejo.

rpm-list: Con esta opción se puede decidir espejar solo una parte de un repositorio (la lista de paquetes dados más dependencias). Por ejemplo: `--rpm-list="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 se llama cuando se ejecuta `cobbler reposync` para el repositorio dado. Por defecto se tiene '`-c cache`'.

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

mirror-locally: Cuando se configura a N, especifica que este repositorio yum se utiliza para referenciarse directamente por *kickstarts* y no para espejarse localmente en el servidor Cobbler. Sólo 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 debería utilizar. Por defecto se utiliza la arquitectura del servidor Cobbler.

breed: Especifica la versión de la distribución. Usualmente Cobbler detectará este parámetro si no se especifica.

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
```

Import El propósito de Cobbler import es configurar un servidor de instalación por red para una o más distribuciones. Espeja contenido basado en una imagen 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 se ejecuta *import*, 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 se guarda automáticamente en */var/www/cobbler/ks_mirror*.

Ejemplos:

1.

```
cobbler import --path=rsync://mirrorserver.example.com/path/
--name=centos --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
\ z
--available-as=nfs://nfs.example.org:/where/mounted/
```

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*.

Kickstarts Los *kickstarts* son archivos que indican cómo debe configurarse el sistema operativo. Un archivo *kickstart* contiene palabras claves, valores y en otros casos, sólo contiene 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 (opcional)**: 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 pueden re-definirse.
- **auth or authconfig (requerida)**: Establece las opciones de autentificación para el sistema. Es similar al comando *authconfig*, que se puede ejecutar después de la instalación. Por defecto, las contraseñas se encriptan y no utilizan shadow.
- **bootloader (requerida)**: Especifica cómo se debe instalar el gestor de arranque.
- **clearpart (opcional)**: Elimina las particiones del sistema, antes de la creación de nuevas particiones. Por defecto no se eliminan las particiones.
- **cmdline (opcional)**: Realiza la instalación en un modo de línea de comandos completamente no interactivo. Cualquier solicitud por interacciones detendrá la instalación.
- **firewall (opcional)**: Esta opción corresponde a la pantalla de configuración de *firewall* en el programa de instalación.
- **firstboot (opcional)**: 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 (opcional)**: Detiene el sistema después de que la instalación se ha completado con éxito. Esto es similar a una instalación manual, en donde Anaconda muestra un mensaje y espera a que el usuario presione una tecla antes de reiniciar. Durante una instalación con *kickstart*, si no se especifica el método de terminación, la opción *reboot* se utiliza como predeterminado.
- **graphical (opcional)**: Realiza la instalación *kickstart* en modo gráfico. Este es el valor predeterminado.
- **install (opcional)**: Indica instalar un sistema nuevo en lugar de actualizar un sistema existente. Este es el modo por defecto.
- **interactive (opcional)**: 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*.
- **keyboard (requerida)**: Establece la distribución del teclado.
- **lang (requerida)**: Establece el idioma que desea utilizar durante la instalación y el idioma predeterminado para utilizar en el sistema instalado.
- **network (opcional)**: 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 (requerida)**: Crea una partición en el sistema.
- **poweroff (opcional)**: Apaga el sistema luego de que la instalación se complete exitosamente.

- **raid (opcional):** Monta un sistema RAID.
- **reboot (opcional):** Reinicia el sistema después de una instalación exitosa.
- **repo (opcional):** Configura un repositorio adicional YUM que puede utilizarse como fuente para la instalación de paquetes.
- **rootpw (requerida):** Establece la contraseña de *root*.
- **selinux (opcional):** Establece el estado del SELinux en el sistema instalado.
- **shutdown (opcional):** Apaga el sistema después de una instalación exitosa.
- **text (opcional):** Realiza la instalación *kickstart* en modo texto. Las instalaciones con *kickstart* se ejecutan en modo gráfico por defecto.
- **timezone (requerida):** Selecciona la zona horaria del sistema.
- **upgrade (opcional):** Indica que se realiza una actualización del sistema instalado.
- **user (opcional):** Crea un usuario en el sistema.
- **zerombr (opcional):** 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.

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 apareza en un archivo *kickstart* se reemplaza por los contenidos en el archivo correspondiente dentro de /var/lib/cobbler/snippets/. Esto permite la re-utilización de código en cada plantilla, aliviando también la lectura de ellas.

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

```
$SNIPPER('nuevo_snippet')
```

Los *snippets* pueden guardarse en subdirectorios para una mejor organización. El orden de precedencia es el siguiente:

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

Para referenciarlo desde el archivo *kickstart*, ahora se tiene:

```
$SNIPPER('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.

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, se sincroniza solo la *meta-data* de la distribución y del perfil. 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
```

3.4.2. Sistema de aprovisionamiento y automatización Cobbler

Instalación Primero y principal, Cobbler necesita Python >= 2.6. Además, requiere de un servidor DHCP, FTP, HTTP, 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
httpd xinetd fence-agents-all -y
```

Configuración de servicios Se deben configurar los servicios que requiere Cobbler:

Configurar FTP y activación 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 configuración utilizada en este trabajo se adecúa 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;
```

```

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 están listos para funcionar, se los inicia y configura para iniciar automáticamente al arrancar la máquina.

```

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

```

Se debe configurar el servidor de Cobbler, para ello se editó el archivo /etc/cobbler/settings.

Se decidió que la clave del usuario *root* de las VMs por defecto sea “qwerty”. Se encripta esta clave con el comando:

```
openssl passwd -1
```

Lo que dará una salida similar a:

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

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 modifica la sección ~~manage_dhcp: 0~~ para habilitar el administrador de 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"~~. En el proyecto se utilizó "admin" como el usuario para ingresar a la interfaz web y la clave "qwerty". Correr el siguiente comando e ingresar la contraseña preferida dos veces:

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

File	Actions
/var/lib/cobbler/kickstarts/Centos_GUI.ks	Edit
/var/lib/cobbler/kickstarts/alumno_v2.ks	Edit
/var/lib/cobbler/kickstarts/alumno_v3 con gui.ks	Edit
/var/lib/cobbler/kickstarts/alumno_v3.ks	Edit
/var/lib/cobbler/kickstarts/default.ks	Edit
/var/lib/cobbler/kickstarts/esxi4-ks.cfg	Edit
/var/lib/cobbler/kickstarts/esxi5-ks.cfg	Edit
/var/lib/cobbler/kickstarts/gu_sample.ks	Edit
/var/lib/cobbler/kickstarts/legacy.ks	Edit
/var/lib/cobbler/kickstarts/xerescus.ks	Edit
/var/lib/cobbler/kickstarts/sample.ks	Edit
/var/lib/cobbler/kickstarts/sample.seed	Edit
/var/lib/cobbler/kickstarts/sample_autoyast.xml	Edit
/var/lib/cobbler/kickstarts/sample_end.ks	Edit
/var/lib/cobbler/kickstarts/sample_esx4.ks	Edit
/var/lib/cobbler/kickstarts/sample_esx4.ks	Edit
/var/lib/cobbler/kickstarts/sample_esx5.ks	Edit
/var/lib/cobbler/kickstarts/sample_old.seed	Edit
/var/lib/cobbler/kickstarts/ubuntu.seed	Edit
/var/lib/cobbler/kickstarts/ubuntu_gui.seed	Edit

Figura 31: Cobbler Web

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

```
systemctl restart httpd.service
systemctl restart dhcpd.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 sincronizar el servidor Cobbler:

```
cobbler get-loaders
systemctl restart cobblerd.service
cobbler cobbler sync
```

3.4.3. 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 divido como se desee, por ejemplo, por sistema operativo, por distribución, por arquitectura, etc.

Se recomienda utilizar como base el directorio `/var/ftp/pub` dado que se utilizará FTP como 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
anónimos.
gpgcheck=0
enabled=1
priority=1
```

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

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
```

3.4.4. Importar imágenes ISO al servidor Cobbler

De esta forma, se puede generar una distribución que se utilice para instalar los sistemas operativos sin necesidad de acceder a Internet, de forma fácil y rápida.

Primero crear un directorio y luego montar el archivo ISO:

```
mkdir /mnt/centos
```

```
mount -t iso9660 -o loop,ro /directorio/isos/CentOS-7-x86_64-DVD-1503-01. /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
```

3.5. 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 dato 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 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.

3.5.1. Tópicos generales de Puppet

Module Un módulo o *module*, es un conjunto de código de Puppet empaquetado 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 módulos 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 están 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
```

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) sólo el estado final deseado, y depende de proveedores integrados para lidiar 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 sigue 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 tipo de recursos, se puede tener un paquete y un servicio, ambos con el mismo título, pero no dos servicios con ese título.
- **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 aclarar 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 aplicarse antes o después de otro, se puede indicar una relación entre ellos. Incluso se puede indicar **que** cambios en un recurso causen que otro recurso se refresque.

- **Cambios, eventos y reportes:** Si Puppet realiza cambios en un recurso, registra esos cambios como eventos. Esos eventos aparecen 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 referencia 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. 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 inequívocamente 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 una 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 el comportamiento del recurso.

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 sobrescribir 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 características 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 período (horas, minutos, meses, etc). Mientras que 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 entradas 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 administrarse individualmente para cada usuario.
- **Host:** Instalar y manejar entradas de *hosts*. Para muchos sistemas, esas entradas deben estar sólo en /etc/hosts, pero algunos sistemas operativos tienen diferentes soluciones.
- **Mount:** Maneja de *filesystems* montados. Puede 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 parámetro source para poner el archivo adecuado.
- **Resources:** Este es un *meta-tipo* que puede controlar otro tipo de recursos. Cualquier *meta-parámetro* 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.

Manifests Un *manifest* o manifiesto, 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 *parser* no retornará nada si no hay errores. En caso de que se detecte un error, debe corregirse 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 y notificar los cambios que Puppet habría realizado si hubiera sido ejecutado sin --noop.

```
puppet apply --noop
```

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

Los catálogos son documentos estáticos que contienen recursos y relaciones.

En la arquitectura estándar maestro-agente, los nodos solicitan los catálogos al Puppet Server, quien los compila cuando son solicitados. Los agentes mantienen en caché sus más recientes catálogos. Si al pedir el catálogo el *master* falla al compilarlo, los agentes reutilizan su catálogo cacheado.

Clases 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 ejecute, parseará este manifiesto y guardará la definición de clase. Luego, ésta puede declararse para aplicarla en los nodos de la infraestructura. En Puppet las clases son *singleton*, lo que quiere decir que una clase puede declararse sólo una vez en un nodo dado. Del siguiente modo se declara una clase:

```
include módulo::clase
```

Aquí, **módulo** le indica a Puppet dónde 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.

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 usarse 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.

Meta-parámetros Los meta-parámetros 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 al deseo del estado del recurso.

Los meta-parámetros hacen cosas como agregar *meta-data* a un recurso (*alias*, *tag*), poner límites cuando el recurso debe sincronizarse (*require*, *schedule*, etc.), evita que Puppet realice cambios (*noop*), y cambia la verborragia del *log* (*loglevel*).

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, sólo el primer 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 **sin** 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 coincide o no puede encontrarse, el nodo `default` es utilizado.

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

Coincidencias Un nodo dado sólo obtendrá los contenidos de *una* definición de nodo, incluso si dos declaraciones de nodo coinciden 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 utiliza ésta.
2. Si hay una expresión regular que coincide con el nombre del nodo, entonces utiliza ésta.
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***.

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 la herramienta 'facter tool'.

3.5.2. Creación del sistema de orquestación Puppet.

Puppet usualmente corre bajo la arquitectura cliente-servidor pero además, puede correr en una arquitectura auto-contenida. La decisión determina qué paquetes serán instalados y qué 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.

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 *puppet server* es un recurso intensivo y debe instalarse 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.

Verificación

Chequeo de la configuración de red En un despliegue agente-maestro se debe preparar la red para el tráfico generado por Puppet.

- **Firewall:** En el desarrollo de este proyecto se trabajó sin *firewall*, por lo tanto éste debe estar desactivado.

- **Resolución de nombres:** Cada nodo debe tener un nombre único.

Para cumplir con este último apartado, se decidió editar el archivo `/etc/hosts` de cada máquina 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 también tiene la necesidad de conocer cuál dirección IP corresponde a cada máquina, se realizó un pequeño código en Python que es capaz de obtener la dirección IP de una máquina en particular revisando el archivo `/var/lib/dhcpd/dhcpd.leases`. Este archivo, es utilizado por el servidor DHCP para saber a qué máquina corresponde cada dirección IP.

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

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

No se debe iniciar el servicio aún.

Como se mencionó anteriormente, *puppetserver* tiene ciertos requerimientos de hardware donde se destaca los 2GB de memoria RAM. Pero ésto no es exactamente así, lo que realmente sucede es que esa es la cantidad de memoria “recomendada” para que funcione correctamente un sistema en producción. Mientras exista ese hardware, *puppet* se “apropiará” de él.

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

```
# 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"
```

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 usarse 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 firmarse 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
```

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:

```
# rpm -Uvh https://yum.puppetlabs.com/puppetlabs-release-pcl-el-7.noarch.
# yum install puppet-agent
```

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

```
wget https://apt.puppetlabs.com/puppetlabs-release-pcl-wheezy.deb
sudo dpkg -i puppetlabs-release-pcl-wheezy.deb
sudo apt-get update
sudo apt-get install puppet-agent
```

Windows 7: Descargar el paquete instalador desde <https://downloads.puppetlabs.com/windows/puppet-agent-4.1.0-x64.msi>

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

La instalación puede ser gráfica o automática. Para hacerlo de forma automática, ingresar en el cuadro “Ejecutar”:

```
msiexec /qn /norestart /i puppet-agent-<VERSION>-x64.msi
PUPPET_MASTER_SERVER=puppet
PUPPET_AGENT_ACCOUNT_USER=Administrador
```

```
PUPPET_AGENT_ACCOUNT_PASSWORD=qwerty
PUPPET_AGENT_ACCOUNT_DOMAIN=example.com
```

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 la cual permite indicar qué usuario es el que ejecutará los manifiestos de Puppet 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* del servidor es "puppet". Si se nombró de otra forma al servidor, se debe editar esto.

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 (producción, desarrollo, etc). De cualquier forma, el maestro puede configurarse para ignorar esta configuración.

Comportamiento de la ejecución:

- **noop**: Si está habilitado el agente no realizará 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.
- **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 treinta 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 #Esta variable la resuelve el snippet para insertar
el hostname de la máquina.

server = puppet
runinterval = 2m
[agent]
classfile = $vardir/classes.txt
localconfig = $vardir/localconfig

```

Ejecutar Puppet Para iniciar el servicio de Puppet una vez configurado correctamente, basta con iniciarla a través de `systemctl`:

```

sudo systemctl start puppetserver
sudo systemctl enable puppetserver

```

En el caso de los clientes, existe una gran variedad de comandos para iniciarlos, puesto que son diferentes los sistemas operativos donde funcionan.

Sobre `Centos` es similar:

```

sudo systemctl start puppet
sudo systemctl enable puppet

```

Sobre `Ubuntu` se debe ejecutar:

```

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

```

Finalmente, en `Windows` se puede hacer por consola o por interfaz gráfica.

Cualquiera sea el sistema operativo que ejecute el agente de Puppet, lo primero que hará es comunicarse con su servidor. Esto podrá hacerlo buscando la dirección IP del servidor “`puppet`” (como se lo llamó en la configuración) en el archivo `hosts`, teniendo en cuenta que su ubicación difiere en los sistemas Linux y Windows.

Se genera un certificado SSL y el servidor puede firmarlo automáticamente, como en la configuración de este proyecto, o de forma manual.

Para corroborar la creación y firmado del servicio:

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

3.6. Windows 7

La instalación automatizada de Windows 7 requiere prepararse sobre un sistema Windows 7. Sin embargo, su despliegue se realiza desde el servidor Cobbler, sobre Linux.

3.6.1. El lado de Windows

Para automatizar la instalación de Windows 7, utilizando el método de imagen estándar personalizada, se necesita 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. En esta máquina es necesario contar con cualquier versión de Windows 7 que tenga las herramientas de WAIK (*Windows Automated Installation Kit*).
- **Máquina de referencia:** Con este término, Microsoft se refiere al equipo sobre el que se instalará el sistema operativo del cual se extraerá la imagen personalizada con las configuraciones y programas deseados.
- **Windows Automated Installation Kit:** Es un conjunto de herramientas y documentación compatible con la configuración y la implementación de los sistemas operativos Windows. Mediante WAIK, se puede automatizar las instalaciones de Windows, capturar imágenes de Windows con ImageX, 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.

Fases de pre-instalación de Windows En esta sección se presenta una serie de conceptos entre los que se incluye terminología, tecnologías principales y métodos de implementación.

Windows SIM (System Image Manager) Es la herramienta que permite crear archivos de respuesta para la instalación desatendida. Se debe usar Windows SIM en el equipo del técnico y, a continuación, transferir el archivo de respuesta para la instalación desatendida al equipo maestro antes de crear la imagen de instalación de Windows. Esta herramienta se instala junto con WAIK.

Archivo de respuesta Es un archivo XML en el que se escriben las respuestas para una serie de cuadros de diálogo de interfaz gráfica de usuario. 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. El archivo de respuesta del programa de instalación de Windows suele llamarse `Unattend.xml` o `Autounattend.xml`. Para crear y modificar este archivo de respuesta, se utiliza Windows SIM.

Contenido del archivo de respuesta

La configuración de un archivo de respuesta se organiza en dos secciones: componentes y paquetes.

- **Componentes:** La sección de componentes de un archivo de respuesta contiene todas las opciones de componentes que se aplican durante la instalación de Windows. Los componentes se organizan en varias fases de configuración: `windowsPE`, `offlineServicing`, `generalize`, `specialize`, `auditSystem`, `auditUser` y `oobeSystem`. Cada

fase de configuración representa una etapa diferente en la instalación de Windows. Las configuraciones se pueden aplicar durante una o más fases. Si se puede aplicar una configuración a más de una fase, se puede elegir la fase en la que se aplicará la configuración.

- **Paquetes:** Microsoft usa los paquetes para distribuir actualizaciones de software, *Service Packs* y paquetes de idioma. Los paquetes también pueden incluir características de Windows. En este proyecto no se ha utilizado esta sección.

Entorno de pre-instalación de Windows (Windows PE) WinPE o *Windows Pre-installation Environment* es un sistema operativo mínimo designado para preparar una computadora para la instalación de Windows. Puede utilizarse 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.

ImageX Es una herramienta de línea de comandos de Microsoft que permite capturar, modificar y aplicar imágenes de disco basadas en archivos para una implementación rápida. ImageX copia archivos de imagen de Windows (.wim) a una red.

En particular, las opciones más importantes al ejecutar ImageX son las siguientes:

- **/capture d:** 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"** Es una etiqueta para el archivo imagen creado.
- **/verify** Indica al programa que se desea verificar la consistencia de la imagen creada.

Sysprep Es la herramienta que prepara una imagen de Windows para las **imágenes** de disco, la comprobación del sistema o la entrega a un cliente. Sysprep se puede usar para quitar datos específicos del sistema desde una imagen de Windows, como es el caso del identificador de seguridad (SID) y la información única relativa al hardware de la máquina. Después de quitar la información del sistema única de una imagen, puede capturar dicha imagen de Windows y usarla para implementarla en varios sistemas. Además, Sysprep puede configurar la imagen de Windows para que ~~se~~ arranque en el modo auditoría. El modo auditoría permite probar la integridad del sistema e instalar aplicaciones y controladores de dispositivos adicionales. Sysprep también se usa para configurar Windows de forma que arranque con la "Bienvenida de Windows" la próxima vez que se inicie el sistema.

En particular, las opciones más importantes al ejecutar Sysprep son las siguientes:

- **/generalize:** Prepara a la instalación de Windows para crear una imagen. Si esta opción se especifica, toda la información única del sistema se quita. Por ejemplo, el SID, cualquier punto de restauración que exista, y todos los *logs* se eliminan. La próxima vez que la computadora inicia, la fase de configuración especial se ejecuta y se asigna un nuevo SID. El reloj de activación de Windows se reinicia si es que no

ha sido reiniciado 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 OOB^E 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 se selecciona, se 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 se procesa 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.

Startnet.cmd *Startnet.cmd* es un *script* que de forma predeterminada inicia *wpeinit.exe*, cuando se carga en memoria el entorno de pre-instalación, que precisamente sirve para instalar dispositivos *Plug and Play*, procesar configuraciones de *Autounattend.xml* y cargar recursos de red. Es necesario para montar SAMBA.

Oscdimg *Oscdimg* es una herramienta de línea de comandos que sirve para crear un archivo de imagen (.iso) de una versión personalizada de 32 bits ó 64 bits de Windows PE.

Las opciones aquí utilizadas 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:\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 dado.
- 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 la opción -nt.

3.6.2. SAMBA

SAMBA es un servidor SMB (*Server Message Block*) libre, desarrollado por Andrew Tridgell y que en la actualidad *está mantenido por 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 se puede hacer que el sistema Linux utilizado actúe como servidor SMB dentro de la red.

SAMBA es en sí un paquete muy complejo, que brinda a los usuarios Linux de un gran abanico 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.

Arquitectura SAMBA SAMBA está compuesto 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 autentificación y acceso a los recursos.
- nmbd: Permite mostrar los servicios disponibles en SAMBA (visualización de los servidores SAMBA en la red, etc).

El cliente está compuesto por:

- smbclient : 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, MacOS, OS/2, etc) según el esquema siguiente:

- Aplicación
- SMB
- NetBios
- TCP/IP
- NetBeui
- IPX/SPX
- Controladores de red

3.6.3. Creación de la imagen de instalación Windows 7

La tabla 6 contiene una columna para cada equipo que interviene en el proceso. Los pasos de la columna en la máquina técnica se realizan en ese equipo. Los pasos de la columna del equipo de referencia se realizan en el equipo donde se crea la imagen personalizada.

Paso	Máquina técnica	Máquina referencia
1	Instalar WAIK	
2	Crear un disco de inicio del Entorno de Pre-instalación de Windows (WinPE)	
3		Instalar y personalizar Windows 7
4		Generalizar el equipo de referencia con Sysprep preparando 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 6: Pasos automatización Windows 7

Instalar WAIK Desde la página oficial de Microsoft descargar el archivo KB3AIK_EN.iso. Grabarlo en un DVD o montarlo en una unidad virtual para instalarlo. El archivo se puede descargar del siguiente enlace:

<https://www.microsoft.com/es-ar/download/details.aspx?id=5753>

WinPE Para crear una imagen ISO de WinPE se necesita ejecutar *Deployment Tools Command Prompt* como administrador. Como se utilizó una versión de 64 bits se especifica amd64, si no, i386.

Copiar los archivos necesarios:

```
copype amd64 d:\directorio\winpe
```

Una vez copiados los archivos, montar el archivo WIM obtenido:

```
imageX /mountrw d:\directorio\winpe\winpe.wim 1 d:\directorio\winpe\mount
```

Para conectar con el servidor Cobbler, se necesita asegurar que se incluye una llamada a wpeinit en el script Startnet.cmd personalizado. Para ello, utilizar un editor de texto y modificar a Startnet.cmd como se muestra a continuación, reemplazando los valores correspondientes.

```
notepad.exe d:\directorio\winpe\mount\Windows\System32\startnet.cmd
```

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

Una vez modificados los valores del *script*, desmontar *winpe.wim*. Antes que nada, cerrar el explorador de Windows y cualquier programa que esté haciendo uso del directorio anterior, ya que en caso contrario dará error.

```
imagex /unmount d:\directorio\winpe\mount /commit
```

Copiar entonces *winpe.wim* al directorio *winpe\ISO\sources* renombrándolo como *boot.wim*:

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

Por último, crear una imagen ISO del nuevo WinPE.

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

Instalar y personalizar Windows 7 Una máquina de referencia tiene una instalación personalizada de Windows que se planea replicar en una o más computadoras. La instalación personalizada se puede crear utilizando un DVD o archivo ISO de Windows.

Este punto conviene realizarlo manualmente y de la forma usual, es decir, instalar manualmente el sistema operativo en la VM. 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 se reinicia e ingresa automáticamente en una cuenta temporal en modo administrador. Permanecerá en este estado por más que se reinicie el equipo, hasta que el comando *Sysprep* se ejecute luego de realizar todas las actualizaciones, instalación de los controladores de dispositivo y las aplicaciones deseadas.

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

Para transferir los programas a instalar, alojados en el servidor Cobbler en */windows/PROGRAMAS/*, desde la máquina de referencia ir a:

- Equipo.
 - Conectar a unidad de red.

Ingresar la dirección IP del servidor con el siguiente formato, como se muestra en las **figuras 32 y 33**:

\IP_del_servidor_Cobbler\entrada_SAMBA

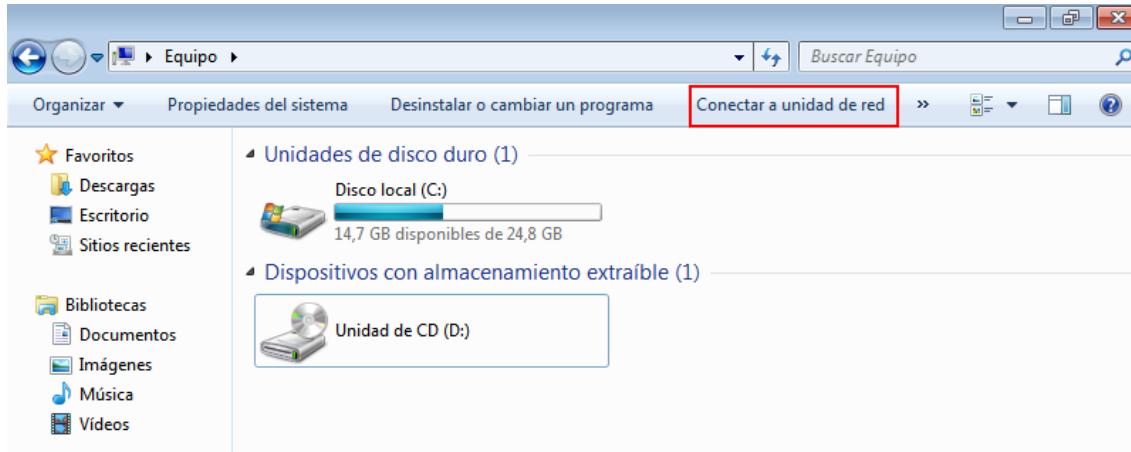


Figura 32: Paso de archivos a Windows

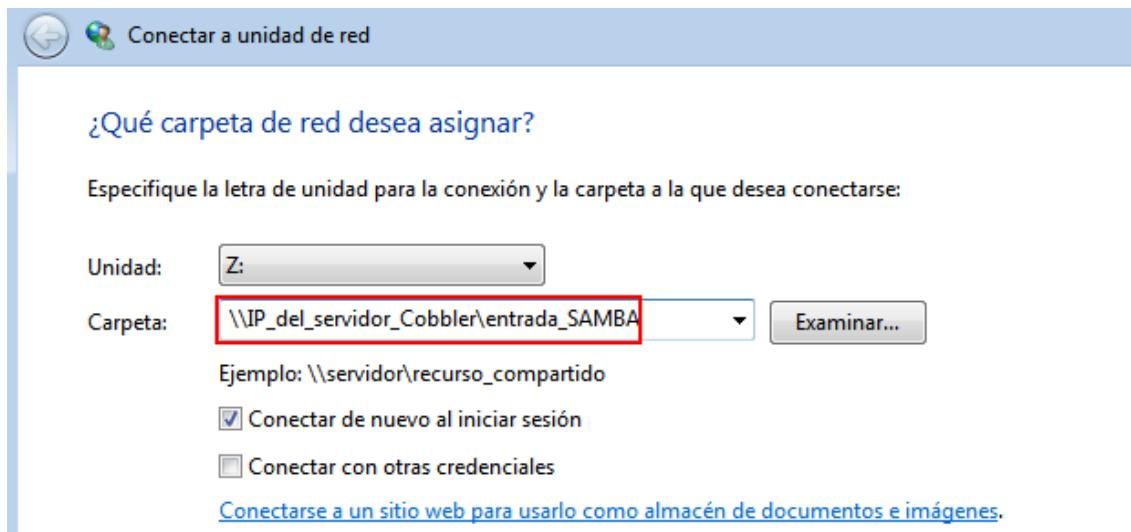


Figura 33: Paso de archivos a Windows

En la **figura 34** se muestra la ruta /windows/PROGRAMAS/ del servidor Cobbler, visto desde la máquina Windows.

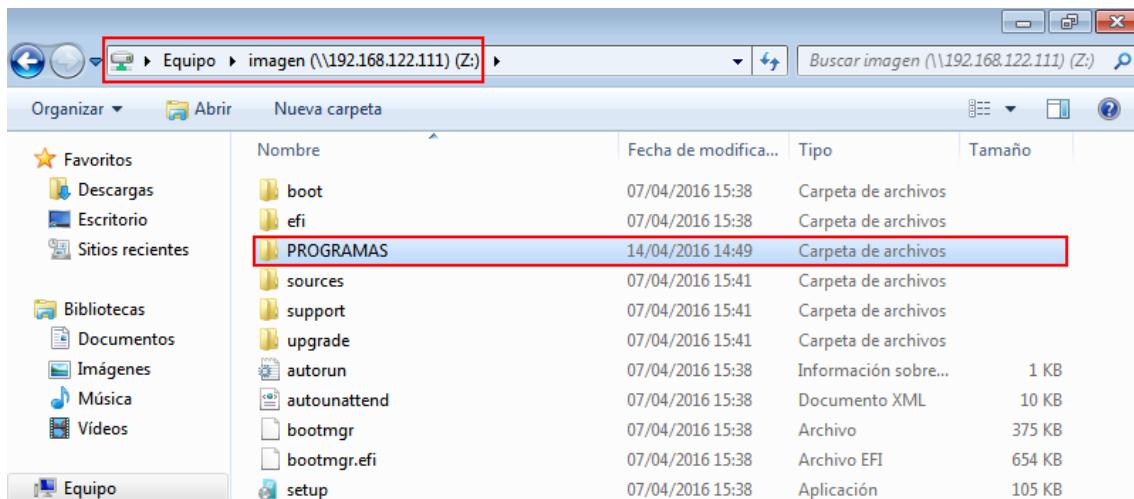


Figura 34: Paso de archivos a Windows

Además de instalar las aplicaciones necesarias, para que el agente Puppet pueda encontrar al nodo servidor, es necesario añadir una entrada al archivo *hosts*.

Desde la terminal, ejecutar como administrador:

```
echo IP_del_servidor_Cobbler puppet >> c:\Windows\System32\drivers\etc\hosts
```

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" al instalarla en cada equipo. Para esto se hace uso de la herramienta Sysprep.

De forma gráfica, cuando inicia el sistema operativo en modo auditoría, Windows 7 ejecuta Sysprep de forma automática. La figura 35 muestra el cuadro de diálogo.

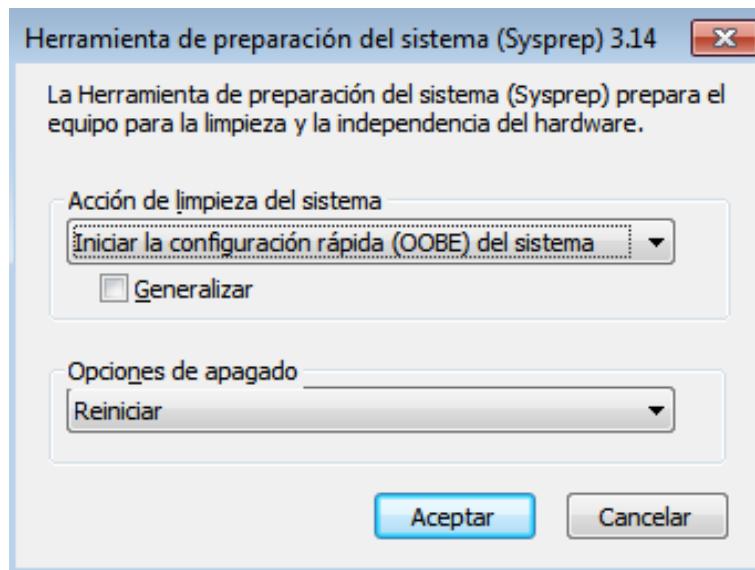


Figura 35: Modo auditoría

- En la sección Acción de limpieza del sistema, seleccionar Iniciar la configuración rápida (OOBE) del sistema.

- Activar la casilla Generalizar.
- En la sección Opciones de apagado, seleccionar Apagar.
- Pulsar Aceptar para ejecutar Sysprep y apagar el equipo.

Particularmente, se decidió realizar la generalización del sistema por consola. Entonces, ejecutar por línea de comando:

```
cd ../../..
cd C:\Windows\System32\Sysprep\
sysprep.exe /generalize /oobe /shutdown
```

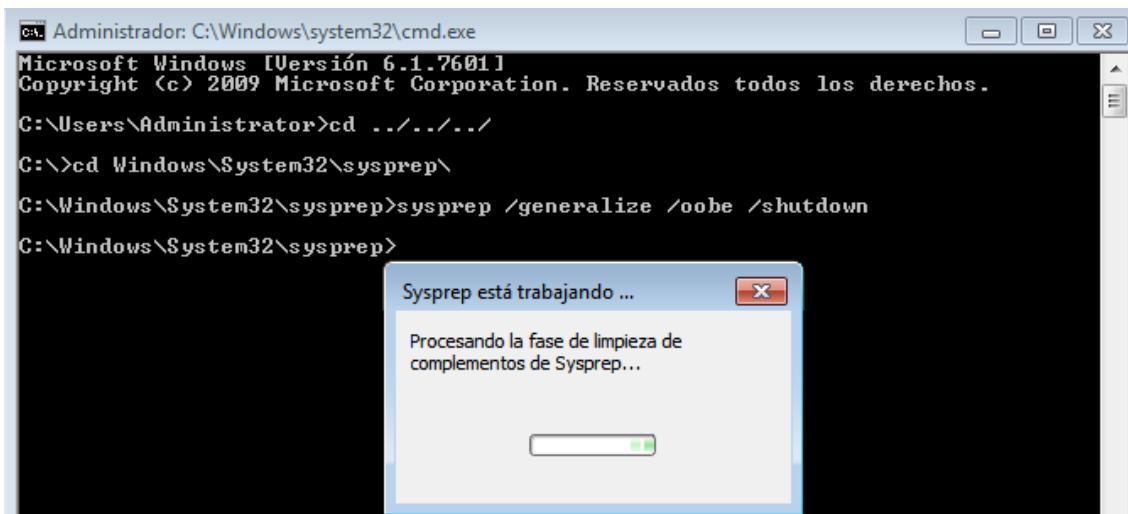


Figura 36: Ejecución sysprep

A esta altura del proceso, se tiene instalado Windows 7 en el equipo de referencia y preparado para capturar una imagen del sistema personalizado.

3.6.4. Capturar el equipo de referencia

Se utiliza la herramienta ImageX, incluida en el WinPE creado, para capturar la imagen. Ésta se guarda en el directorio compartido SAMBA.

Conectar en la máquina de referencia el WinPE y configurar el orden de arranque como muestran las [figuras 37 y 38](#).

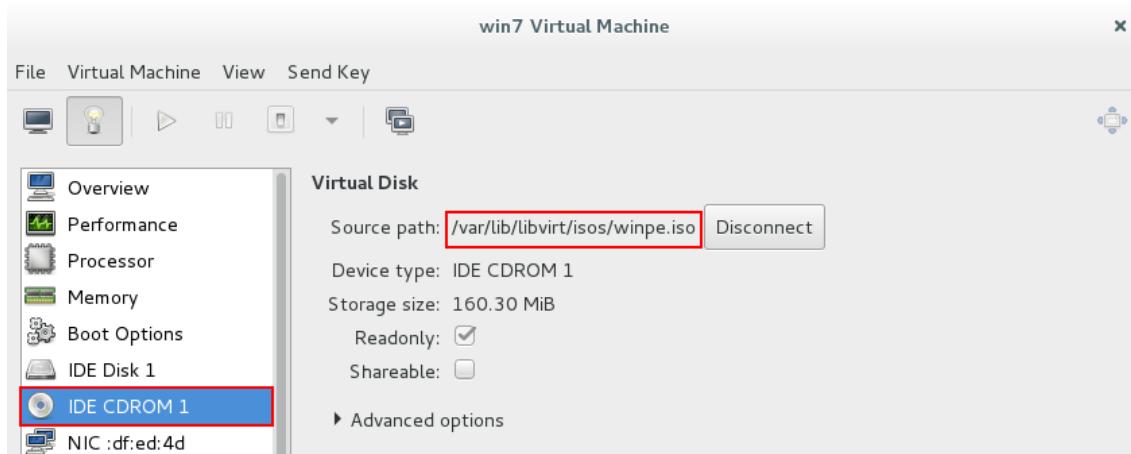


Figura 37: Carga WinPE.iso para extraer imagen

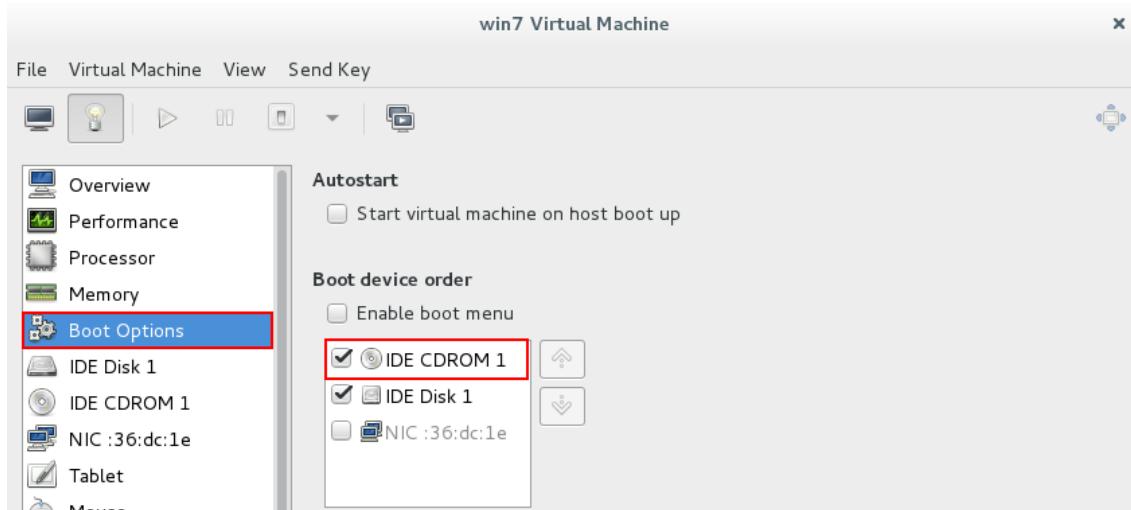
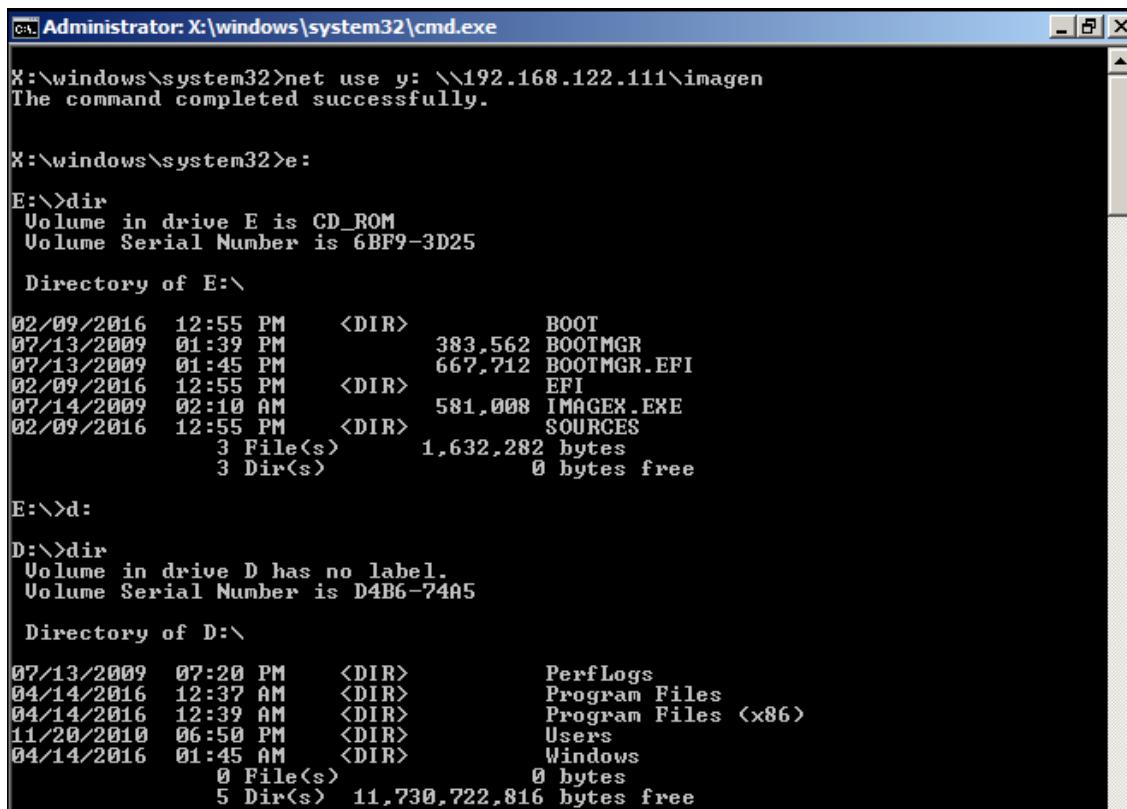


Figura 38: Prioridad de inicio en máquina virtual

Una vez encendida la VM, se carga el disco WinPE y retorna una consola. Verificar que la máquina conecta al servidor SAMBA y corroborar que se tiene acceso a la ubicación compartida:

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

Es necesario corroborar la letra asignada al disco WinPE, que usualmente es `e:` y verificar la letra de la partición donde está instalado Windows personalizado, la cual es usualmente `d:`.



```

Administrator: X:\windows\system32\cmd.exe

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 39: Verificación de letras de unidad

Desde la VM que corre WinPE, para capturar la imagen ejecutar:

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

Una vez finalizado el proceso de copiado se muestra el porcentaje de finalización y el tiempo requerido para completar el proceso.

```
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 40: ImageX capturando imagen

Autounattend.xml Para generar el archivo de respuestas, una vez instalado el WAIK, se debe abrir en modo Administrador el programa *Windows System Image Manager* desde el menú Inicio.

Seleccionar:

- Windows Image
 - Select a Windows image or catalog file.

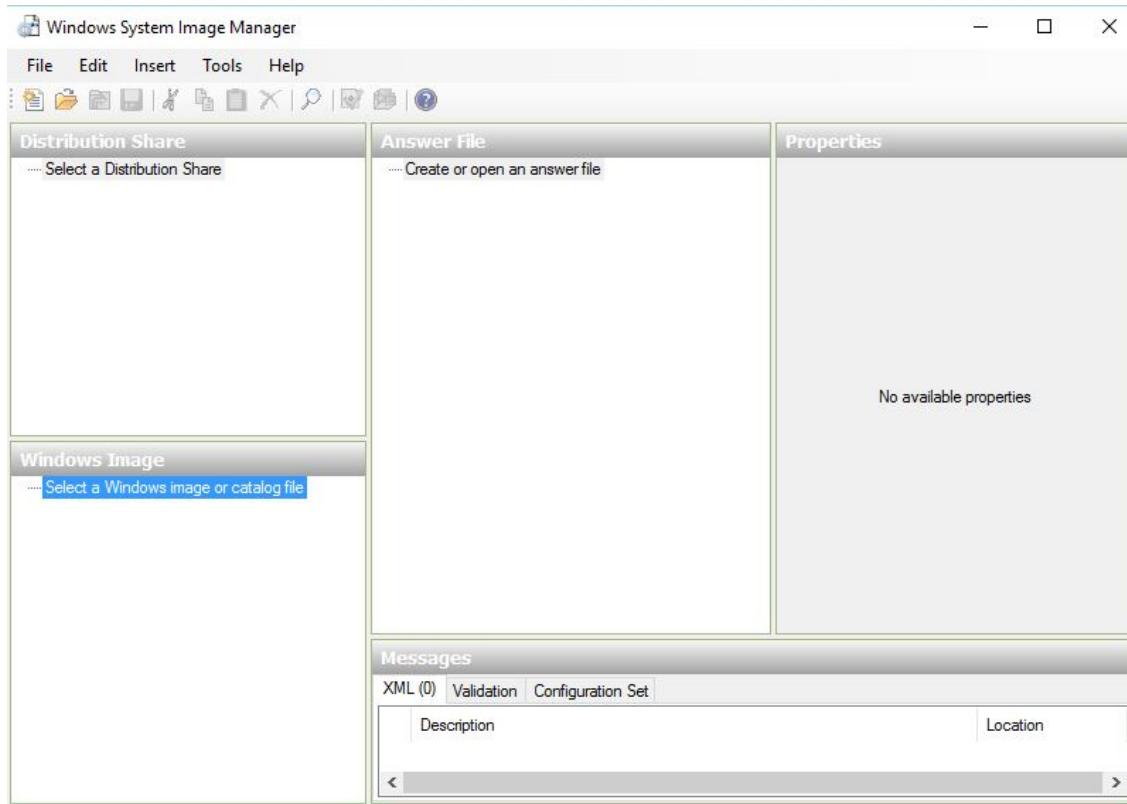


Figura 41: Windows System Image Manager

Se debe buscar en el directorio de la imagen del sistema operativo ya descomprimida, un archivo .clg. En particular, este archivo es `install_Windows_7_ULTIMATE.clg`. Luego, crear el archivo de respuestas de instalación:

- Answer File
 - Create or open a new answer file
 - ▷ Create a new answer file

A partir de la figura 42 y hasta la 63, se muestra una configuración personalizada completa utilizando Windows AIK.

En las figuras 42 y 43, se ha configurado la entrada de texto, interfaz gráfica y ayuda del sistema en idioma español.

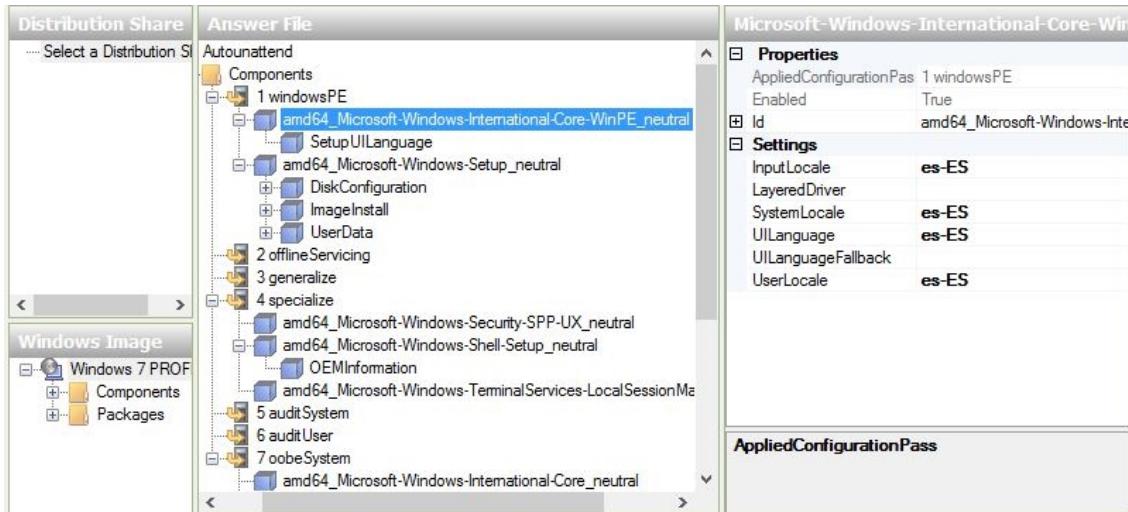


Figura 42: Idioma y teclado.

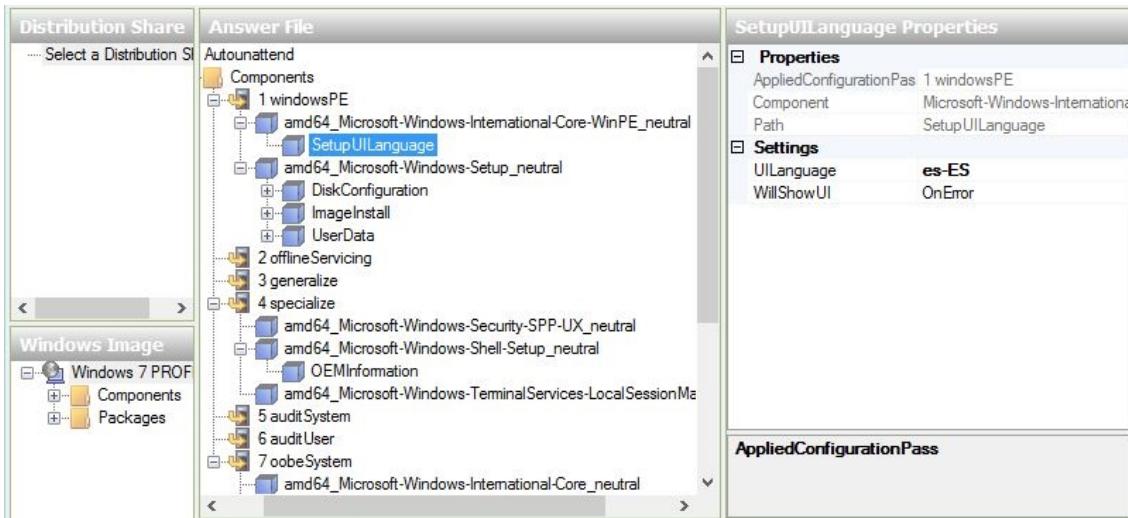


Figura 43: Idioma.

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

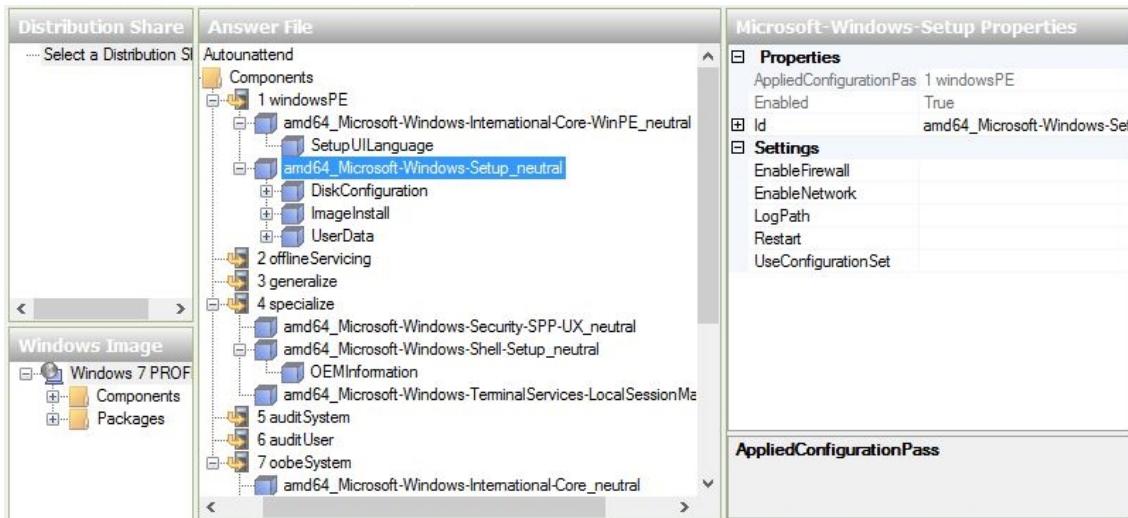


Figura 44: Firewall activado por defecto.

En las figuras 45 a 51, se han configurado los discos para que existan dos particiones. Una donde se instalará el MBR (*Master Boot Record*) y otra donde se copiarán los archivos del sistema operativo.

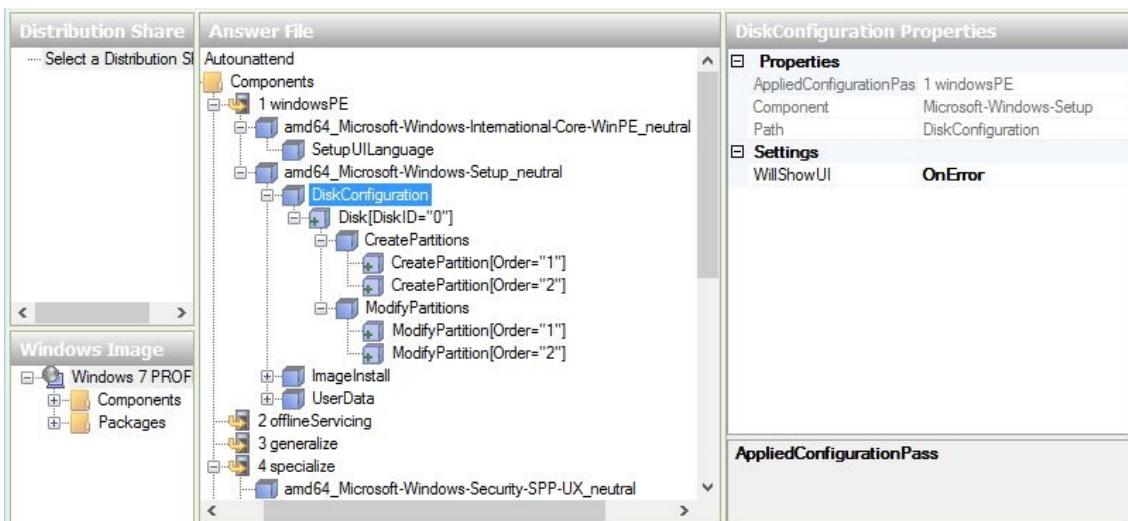


Figura 45: Discos.

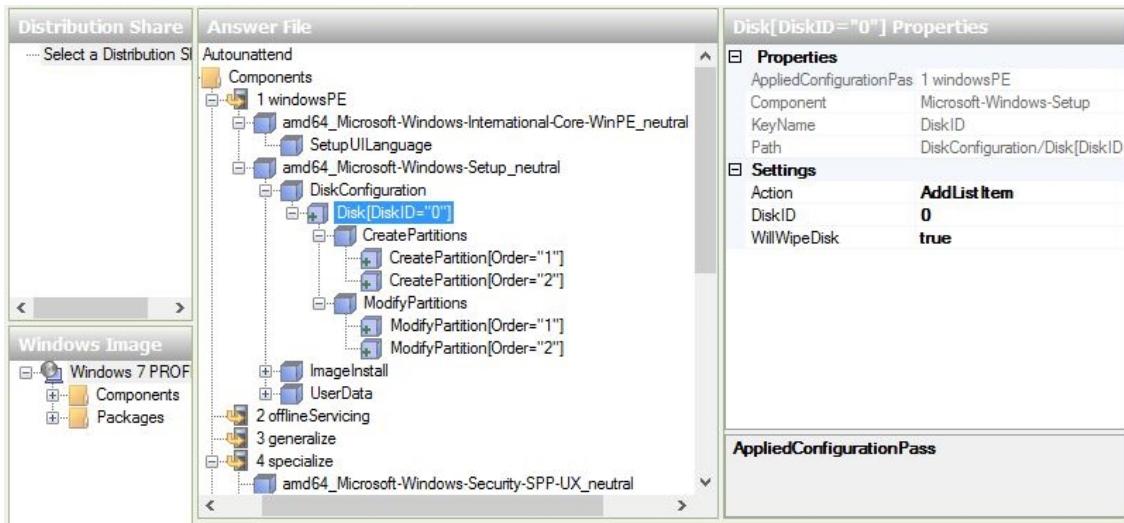


Figura 46: Configuración discos. Disco 0.

Partición de 300 MB, donde será instalado el MBR.

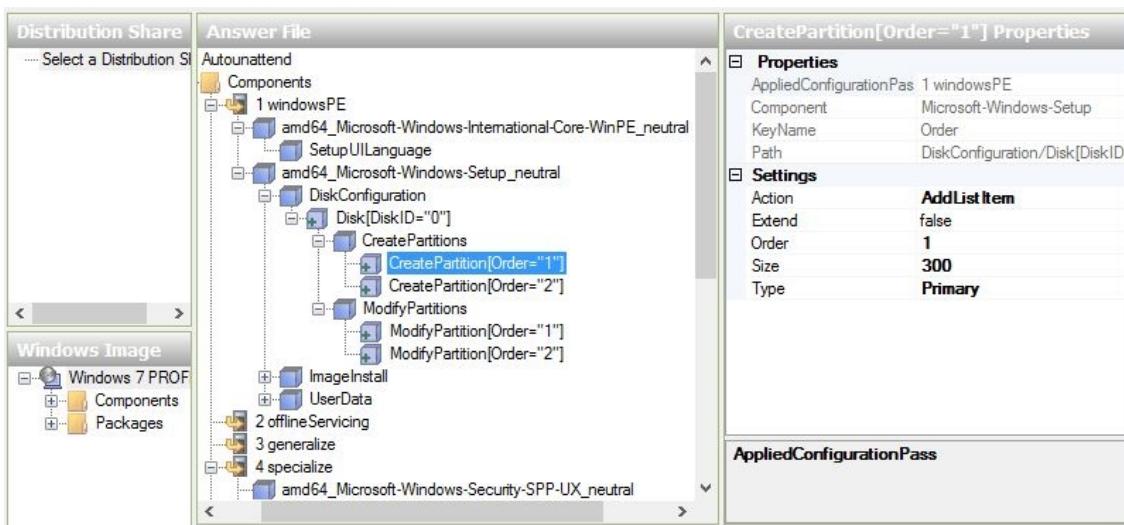


Figura 47: Discos. Particionamiento.

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

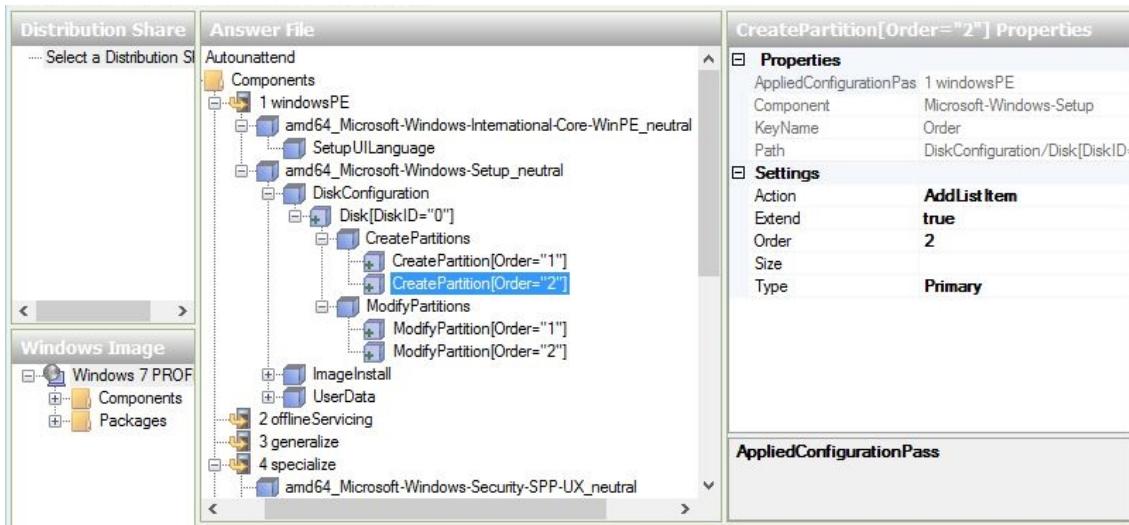


Figura 48: Discos. Particionamiento.

Formato de la partición.

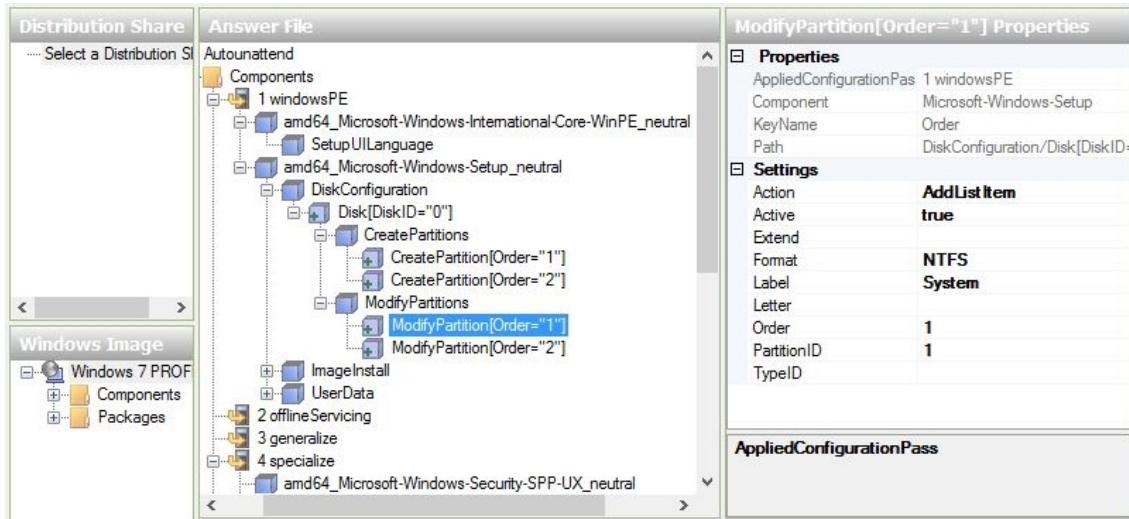


Figura 49: Discos. Particionamiento.

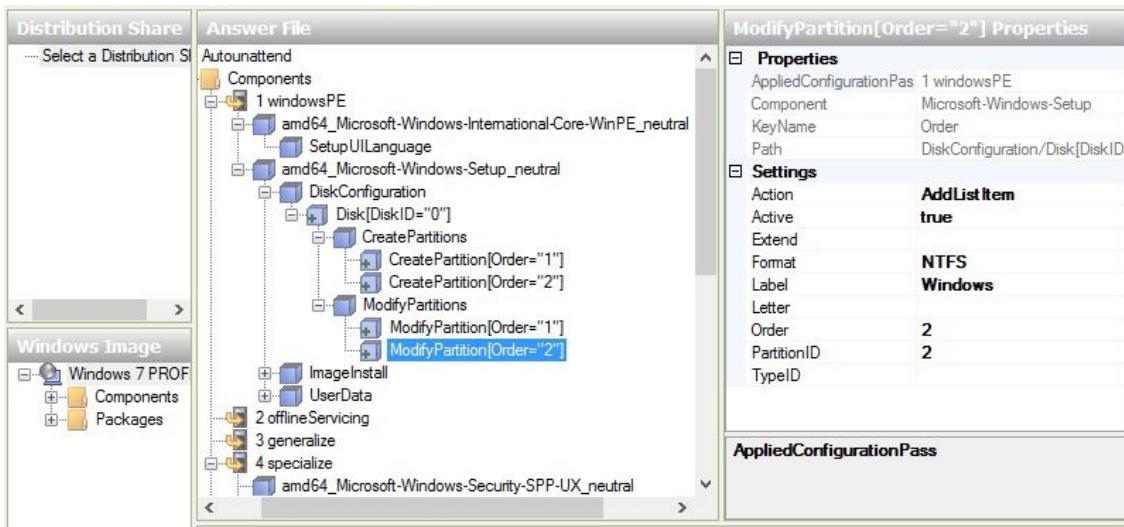


Figura 50: Discos. Particionamiento.

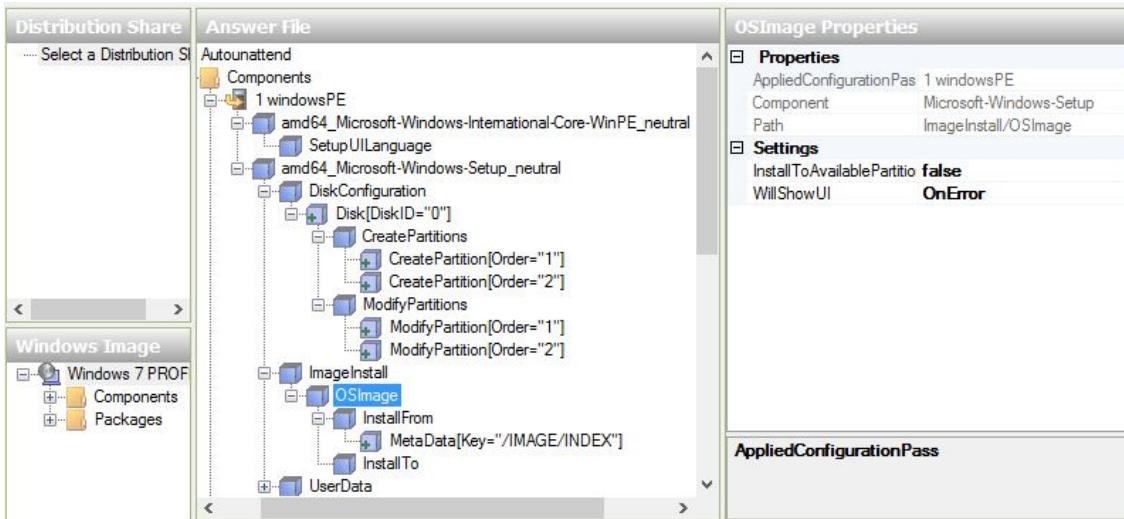


Figura 51: Instalación del sistema operativo.

En el caso de contar con unapc imagen de instalación con múltiples versiones como Windows Home, Basic, Professional y Ultimate y/o con ambas arquitecturas (i386 y amd64), se debe seleccionar cuál de opción instalar. La manera más fácil es conocer la posición, en la lista de opciones de instalación, de la versión que se quiere instalar. Seleccionarla como se muestra en la figura 52, donde se observa que en este caso es la primera posición.

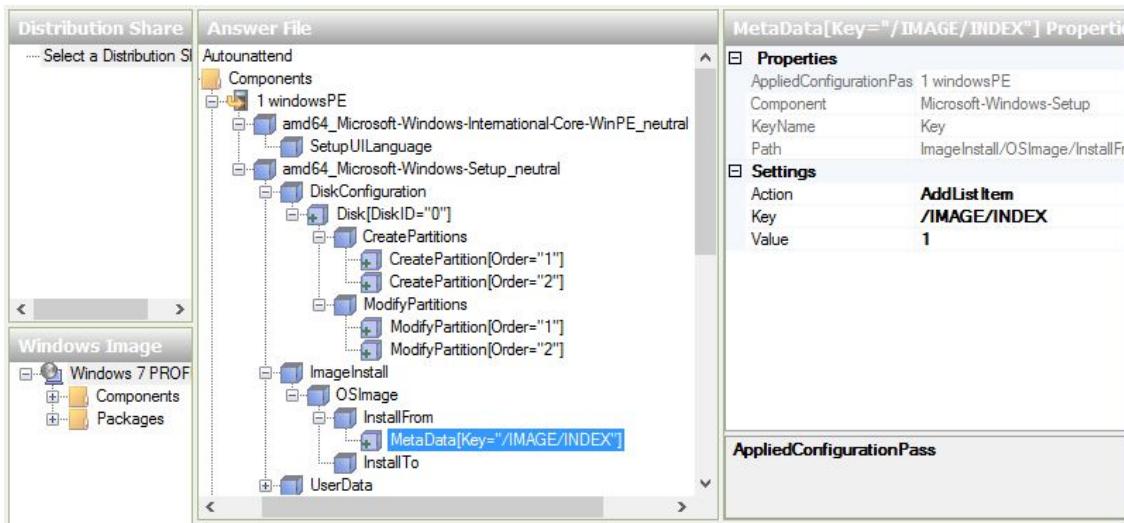


Figura 52: Instalación del sistema operativo. Elección de la versión.

Selección de la partición de instalación de Windows.

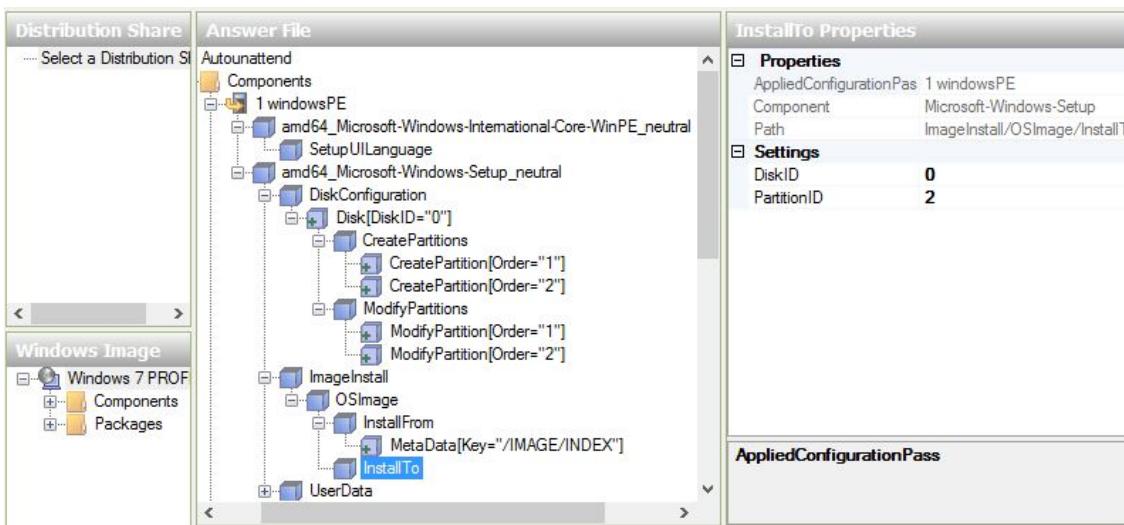


Figura 53: Elección de partición de instalación.

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

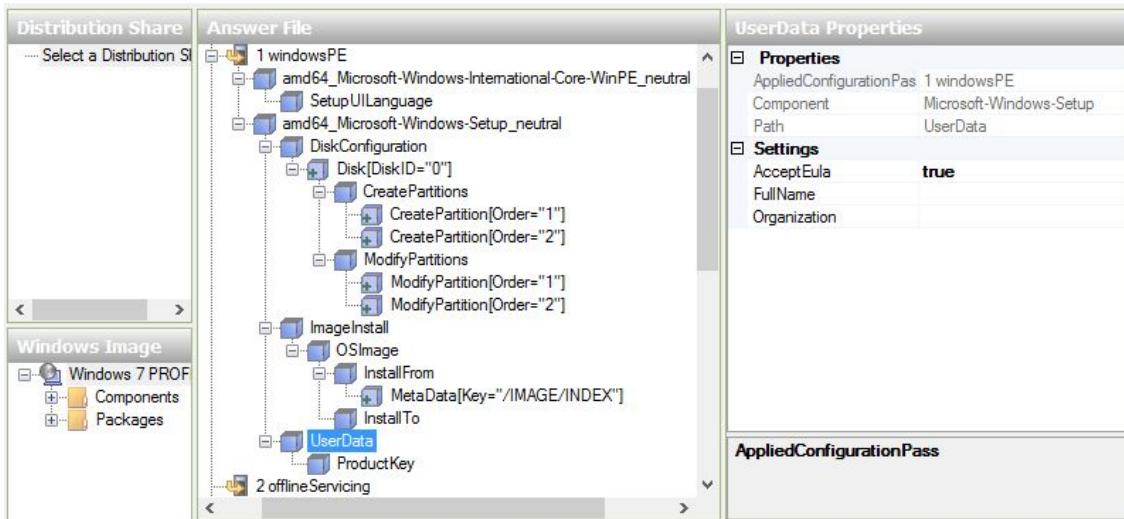


Figura 54: Aceptación de las condiciones de privacidad y uso.

Para poder realizar instalaciones desatendidas, la clave de producto debe ser del tipo de Volumen, esto es, una clave para realizar instalaciones en masa. A modo de ejemplo y para pruebas, Microsoft entrega una serie de claves para ésto. La clave de producto utilizada en este caso se obtuvo de <https://technet.microsoft.com/en-us/library/jj612867.aspx> y es **FJ82H-XT6CR-J8D7P-XQJJ2-GPDD4**.

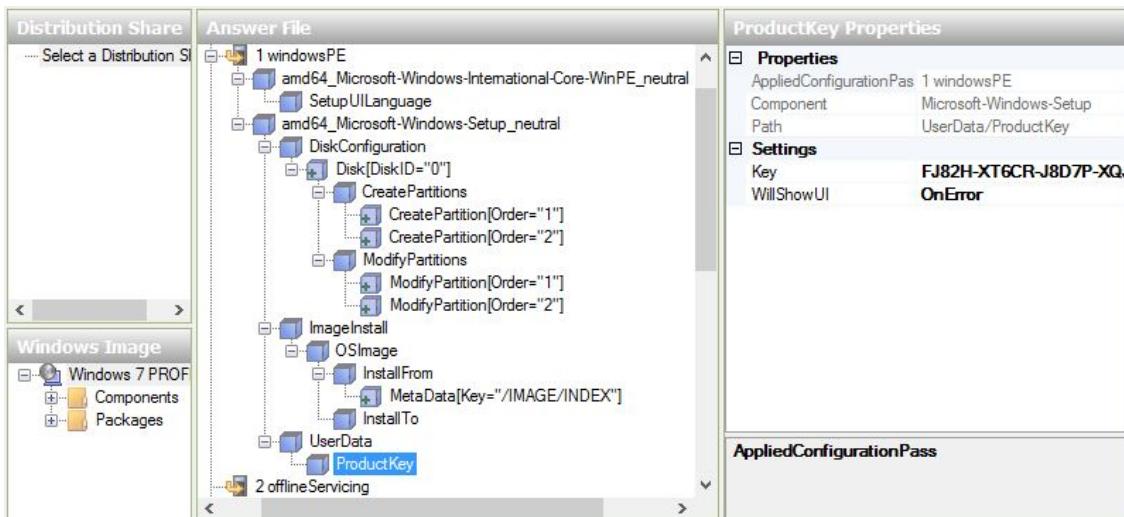


Figura 55: Clave de producto.

Saltar activación del producto.

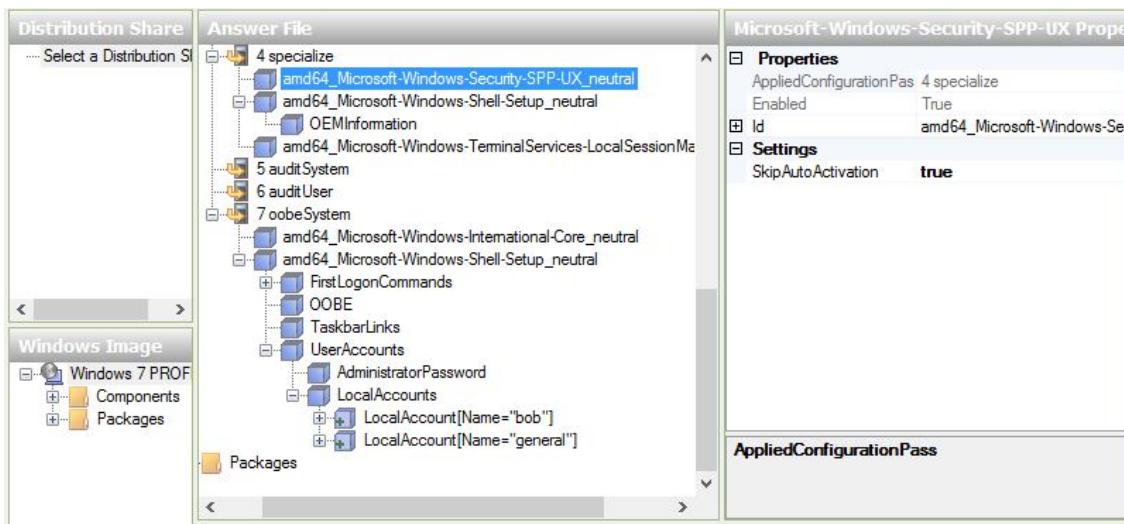


Figura 56: Activación del producto.

Uno de los problemas que se presentan al momento de realizar una instalación en masa, es asignar un *hostname* único a cada máquina que se instala con el mismo archivo Autounattend.xml. Para solucionarlo, 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.

La única manera válida para copiar todos los cambios y preferencias que se han hecho al perfil de usuario por defecto, es añadir el campo CopyProfile con el valor true. Esto es, cada vez que se crea un nuevo usuario, se le aplicarán las configuraciones hechas al personalizar la imagen.

Lo anterior se muestra en la figura 57.

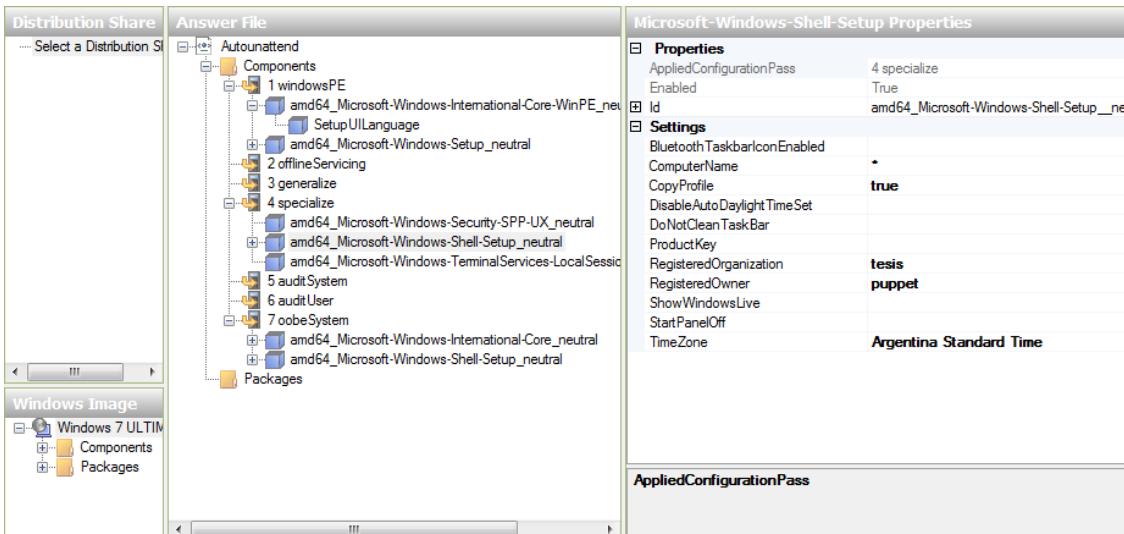


Figura 57: Hostname, organización, propietario y zona horaria.

Como se indicó anteriormente, algunas configuraciones se pueden ingresar en diferentes etapas. Entonces, se debe ingresar nuevamente la configuración deseada.

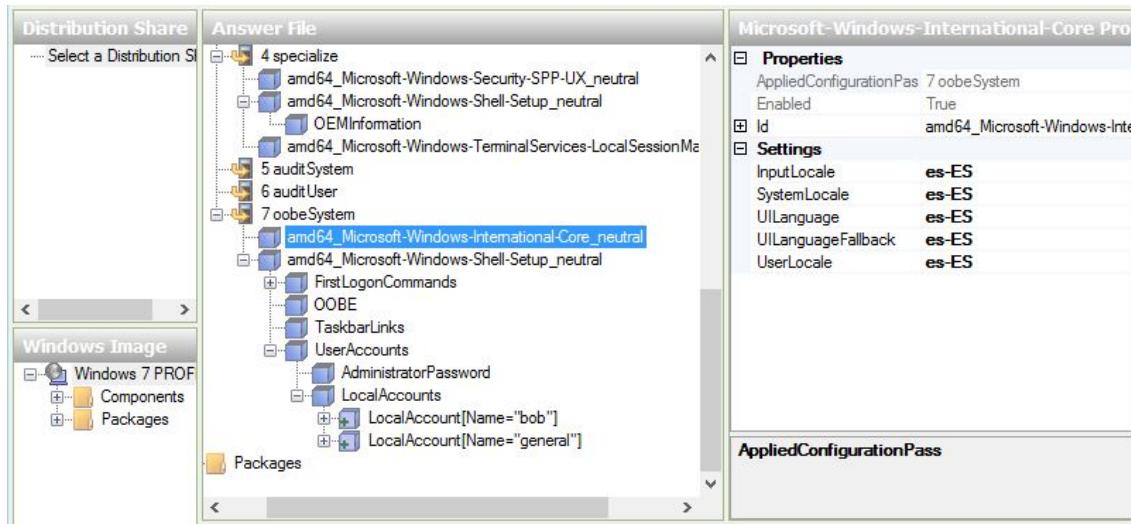


Figura 58: Idioma, teclado, regional.

Se debe ingresar nuevamente la configuración deseada.

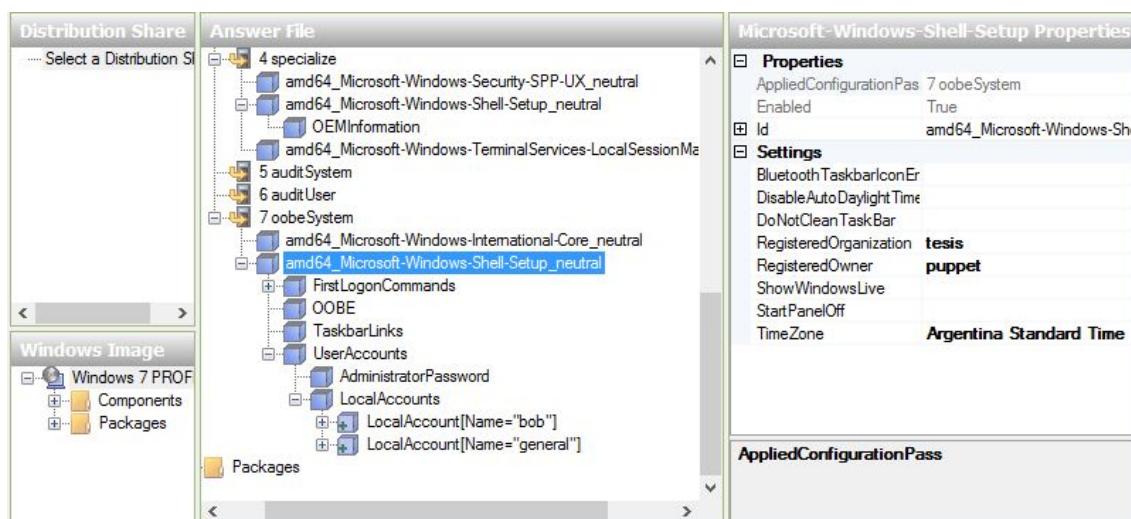


Figura 59: Organización, propietario y zona horaria.

Selección de la configuración rápida de red.

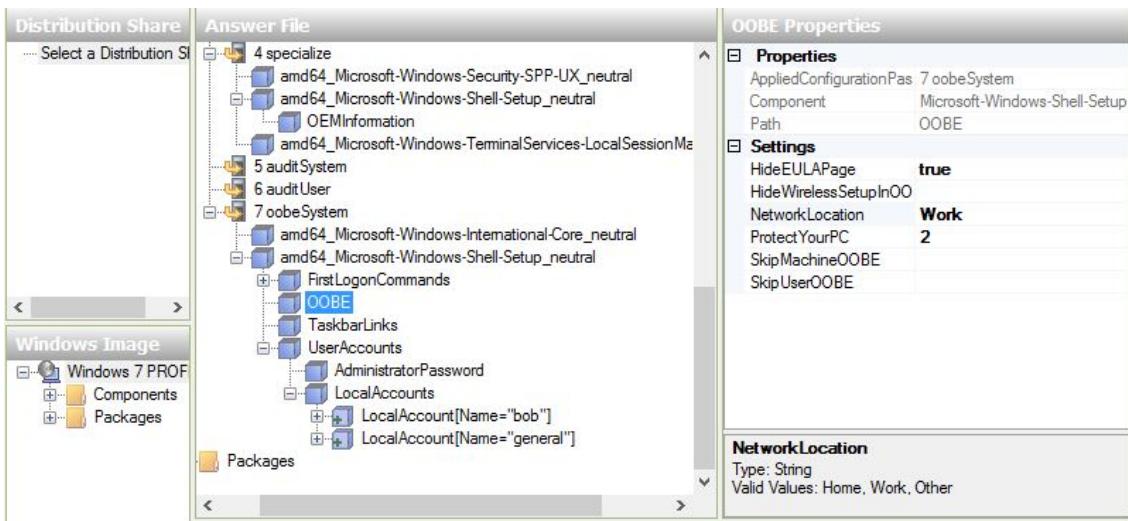


Figura 60: Configuración de red.

Configuración de contraseñas de administrador.

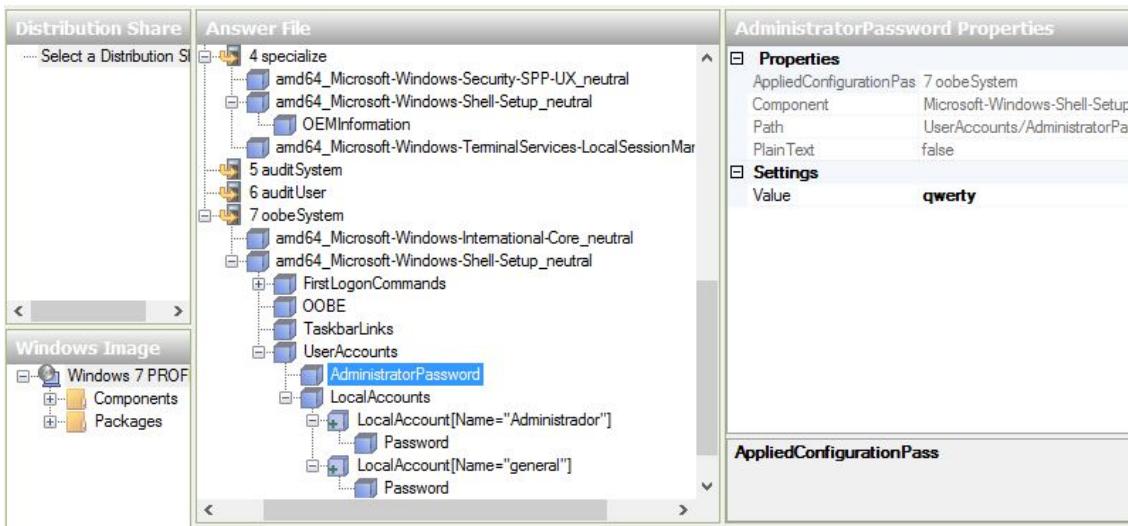


Figura 61: Contraseña administrador.

Las **figuras 62 y 63** muestran la creación de cuentas de usuario administrador e invitado.

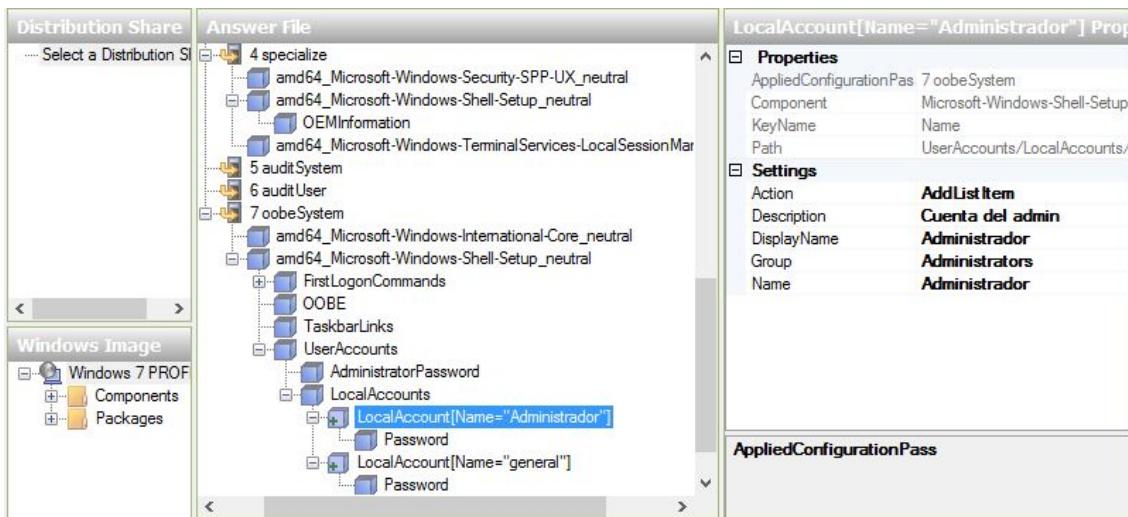


Figura 62: Cuentas de usuario, administrador.

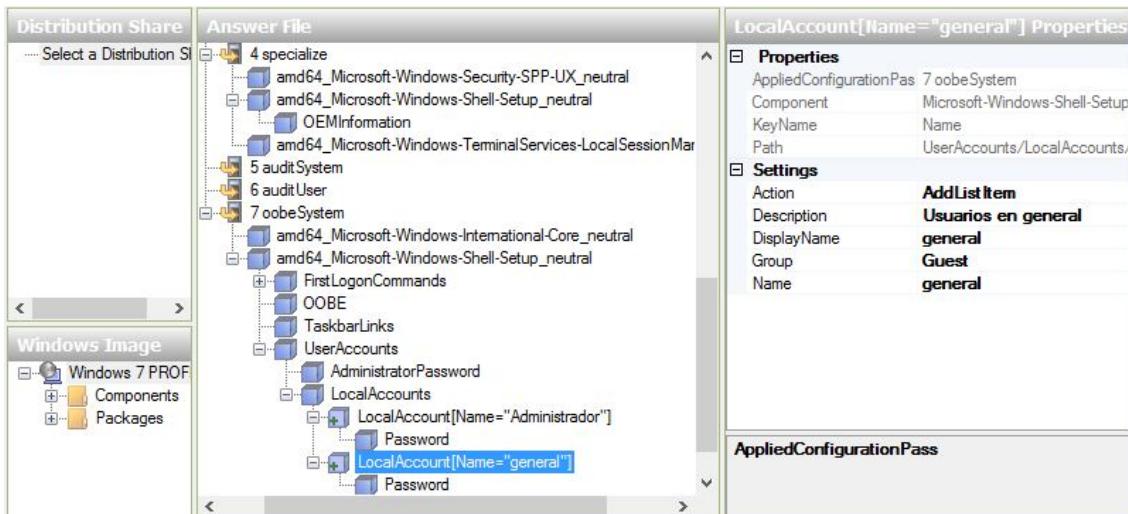


Figura 63: Cuentas de usuario, invitado.

Crear nuevos medios de instalación de Windows 7 para la imagen personalizada
Para unir todo lo expuesto en un archivo ISO, en la máquina técnica realizar las siguientes operaciones:

- Descomprimir todo el contenido de la imagen de instalación sin modificar de Windows 7 en \directorio-instalacion\.
- En el sub-directorio *Sources* (\directorio-instalacion\Sources\), sustituir la imagen *install.wim*, por la imagen capturada *imagenPersonalizada.wim*, renombrándola como *install.wim*.
- En el directorio principal de donde se extrajo la imagen, copiar el archivo de respuestas de instalación *Autounattend.xml*.

- Por último, generar el archivo ISO. Abrir *Deployment Tools Command Prompt* como administrador, y ejecutar:

```
o oscdimg -m -h -u2 -bd:\directorio-instalacion\boot\etfsboot.com  
d:\directorio-instalacion\ d:\path-directorio-destino\Win7-personalizado
```

En este punto se tiene una imagen de instalación personalizada y automatizada lista para instalar 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.

3.6.5. 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 directorio /var/lib/cobbler/isos, 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
```

La zona de intercambio entre ambos sistemas operativos se logra utilizando SAMBA. Por ello es necesario crear un directorio que se utilizará como zona de intercambio entre Linux y Windows:

```
sudo mkdir /windows/
```

En este directorio copiar el contenido de la imagen de instalación personalizada de Windows.

```
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
```

Instalación de 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

```

Luego de modificar la configuración reiniciar el servicio smb.

```
sudo systemctl restart smb
```

Para corroborar que se pueden compartir archivos, si se hace desde otra máquina Linux, es necesario tener los siguientes paquetes instalados:

```
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/
```

Para corroborar desde otra máquina Windows, los pasos se explican en la sección 3.6.3 en las figuras 32 a 34.

Deberían aparecer ya los archivos compartidos en esa ubicación.

Integración con Cobbler Para integrar la imagen de instalación Windows al servidor Cobbler se debe añadir una entrada distro y una entrada de profile a Cobbler, sincronizarlo y reiniciar el servicio.

```
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
```

3.7. Interfaz Web

Se decidió realizar la interfaz web en Python, dado que el paquete Bottle provee un servidor web fácil de utilizar y fue recomendado por el director del proyecto. Instalar Bottle es muy simple, simplemente se realiza a través del siguiente comando:

```
sudo easy-install bottle
```

Se crearon dos ejemplos muy simples que luego se utilizaron en la aplicación para recolectar parámetros utilizados para crear las máquinas virtuales.

El siguiente ejemplo pide el ingreso de tres números que representan la cantidad de máquinas virtuales a crear. El primero corresponde a CentOS, el segundo a Ubuntu y el último a Windows. Luego, utilizará estos parámetros para crear la cantidad requerida de máquinas virtuales en cada sistema operativo.

Con las siguientes funciones se definió el modo en que procederá el servidor Bottle cuando recibe peticiones get y post enviadas a la URL /virtual_machine. En la figura 64 se ve el decorador get, el cual indica que la función definida a continuación se ejecutará cuando una solicitud get apuntada a la URL /virtual_machine llegue al servidor. La función devuelve el HTML de un formulario al navegador, que el usuario completará para indicar la cantidad de máquinas virtuales que desea crear con cada sistema operativo. Los datos ingresados en el formulario se enviarán al servidor con el método post.

```
@get('/virtual_machine') #Pagina principal, la cual pide los datos necesarios para crear la VM.
def creaVM():
    peticionhtml = open("/home/juan/Tesis/Python/HTMLs/S0_estandar.html","r", 0)
    return peticionhtml
```

Figura 64: Método get

En la figura 65, se **tiene** el decorador post. Luego del decorador se encuentra la función que se ejecutará cuando una solicitud post apuntada a la URL /virtual_machine llegue al servidor. Este caso se dará, por ejemplo, cuando el usuario complete el formulario devuelto en la función de la figura 64. Esta función tomará los parámetros presentes en el formulario y realizará el proceso para crear las máquinas virtuales deseadas. Además, la función devuelve el HTML que indica que se están instalando las máquinas virtuales.

```
#Toma los parametros ingresados e indica si se creo o no la VM con exito.
@post('/virtual_machine')
def do_creaVM():
    #Control de error que indica si se colocaron parametros invalidos.
    try:
        ncentos = int(request.forms.get('ncentos'))
        nubuntu = int(request.forms.get('nubuntu'))
        nwindows = int(request.forms.get('nwindows'))
    except ValueError:
        frace = '''Por favor, ingrese solo el NUMERO de la cantidad de maquinas que desea crear \n'''
        peticionhtml = commands.getoutput("cat /home/juan/Tesis/Python/HTMLs/virtual_machine.html")
        return frace + peticionhtml

    CreaVm(ncentos, "centos")
    CreaVm(nubuntu,"ubuntugui")
    CreaVm(nwindows,"windows")

return estadosVM()
```

Figura 65: Método post

La última línea que se coloca en el código, será la que permita iniciar el servidor web de Bottle, es decir, la función run. Recibirá cada petición a la dirección y puerto definidos hasta que se detenga la ejecución de la aplicación.

```
run(host='localhost', port=8080, debug=True)
```

Figura 66: Método run

Si se declara la función run como en la figura 66, solo se puede acceder al servidor web Bottle desde la misma máquina que lo ejecuta (*localhost*) y utilizando el puerto 8080. Para permitir el acceso al servidor desde otra máquina, se debe reemplazar *localhost* por la IP deseada o por 0.0.0.0 para habilitar toda la red.

Finalmente, se ve que **esta** activada la opción de *debug*. Ésta permite ver, a través de la página web, el retorno de los errores que retorna Python. Una vez puesto en producción el sistema, esta opción debe desactivarse.

Las **figuras 67** 70 ilustran la sencilla interfaz web.

a

Servidor de maquinas virtuales - Mozilla Firefox

Servidor de maquinas virtuales | Servidor de maquinas virtuales | Servidor de maquinas virtuales | Estado de las maquinas v... | +

localhost:8080/virtual_machine

Indique la cantidad de VMs de con cada SO a crear.

Crear virtual machine con parametros.

Editar configuraciones de las maquinas virtuales.

Ver estado actual de las maquinas virtuales.

CentOS:

Ubuntu:

Windows:

Crear

The screenshot shows a Mozilla Firefox window with four tabs open. The active tab is titled 'Servidor de maquinas virtuales' and has the URL 'localhost:8080/virtual_machine'. The page content is a form for creating virtual machines. It asks for the number of VMs for three operating systems: CentOS, Ubuntu, and Windows. There is also a sidebar with links for creating unparametrized machines, editing configurations, and viewing current states.

Figura 67: Crear virtual_machine

Servidor de maquinas virtuales - Mozilla Firefox

Servidor de maquinas virtuales | Servidor de maquinas virtuales | Servidor de maquinas virtuales | Estado de las maquinas v... | +

localhost:8080/virtual_machine_parametrizada

Indique las caracteristicas deseadas.

Crear virtual machine sin parametros.

Editar configuraciones de las maquinas virtuales.

Ver estado actual de las maquinas virtuales.

Sistema Operativo: CentOS Windows Ubuntu

Memoria Ram:

Espacio de Disco:

Crear

The screenshot shows a Mozilla Firefox window with four tabs open. The active tab is titled 'Servidor de maquinas virtuales' and has the URL 'localhost:8080/virtual_machine_parametrizada'. The page content is a form for creating a parametrized virtual machine. It asks for system requirements: OS (CentOS, Windows, Ubuntu), RAM, and Disk Space. There is also a sidebar with links for creating unparametrized machines, editing configurations, and viewing current states.

Figura 68: Crear virtual_machine_parametrizada

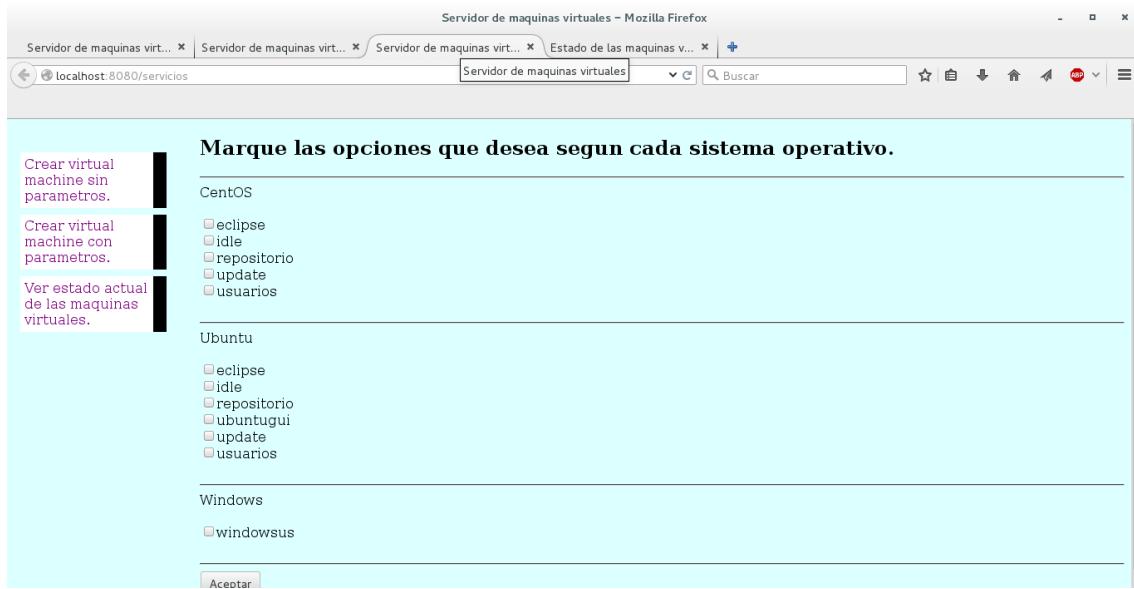


Figura 69: Editar políticas/servicios

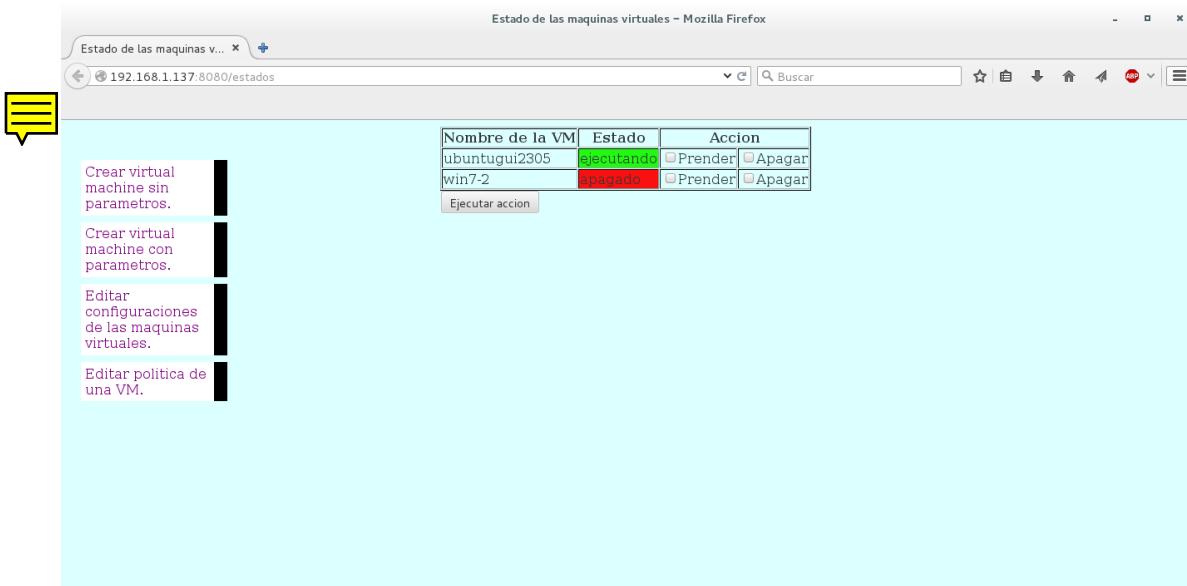


Figura 70: Estado de las máquinas virtuales

3.8. Validación de las Herramientas

Se realizó una etapa de pruebas para validar el funcionamiento del sistema en conjunto. Las tablas 7,8 y 9 muestran los casos de pruebas seleccionados.

ID	Requerimiento asociado	Casos de prueba	Pasos de la prueba	Datos de la prueba	Resultado esperado	Prioridad	Ejecutante	Resultado obtenido
PU_01	RF01 RF02 RF03 RF04 RF05 RF09	Crear máquina virtual con cada perfil	1- Ingresar a la web http://localhost/máquinas virtuales/ 2- Introducir X cantidad de máquinas a crear	CentOS = 1 Ubuntu = 1 Windows = 1	Correcta Creación de 3 máquinas virtuales, cada una con un diferente sistema operativo	Alta	Usuario	OK
PU_02	RF01 RF02 RF03 RF04 RF05 RF08 RF09	Crear varias máquinas virtuales con el mismo perfil	1- Ingresar a la web http://localhost/máquinas virtuales/ 2- Introducir X>1 cantidad de máquinas a crear	CentOS = 0 Ubuntu = 3 Windows = 0 CentOS (OS en mayúsculas)	Correcta Creación de 3 máquinas virtuales, cada una con el sistema operativo Ubuntu14	Alta	Usuario	OK
PU_03	RF09	Tipo de dato incorrecto en la creación de VM	1- Ingresar a la web http://localhost/máquinas virtuales/ 2- Introducir un string en la cantidad de máquinas a crear	CentOS = hola Ubuntu = 1 Windows = 1	Mensaje indicando que se deben utilizar números	Baja	Usuario	OK
PU_04	RF09	Ausencia de dato	1- Ingresar a la web http://localhost/máquinas virtuales/ 2- No introducir datos y poner a crear	CentOS = Ubuntu = 1 Windows = 1	Mensaje indicando que se deben utilizar números como datos	Baja	Usuario	OK

Tabla 7: Casos de prueba del usuario del sistema - Parte 1

PU_05	RF09	Navegar entre pestañas	1- Ingresar a la web <u>http://localhost:8080/</u> virtual_machine 2- Ingresar a cada pestaña	- Navegación correcta entre pestañas	Media	Usuario	OK	
PU_06	RF01 RF02 RF03 RF04 RF05 RF09	Crear máquinas virtuales parametrizadas	1- Ingresar a la web <u>http://localhost:8080/</u> virtual_machine_y 2- Introducir los datos necesarios	sistema operativo = Ubuntu RAM=2048 Disco = 25	Correcta creación de la VM con Ubuntu y las características pedidas	Media	Usuario	OK
PU_07	RF09	Características insuficientes	1- Ingresar a la web <u>http://localhost:8080/</u> virtual_machine_parametrizada_2 2- Introducir los datos necesarios	sistema operativo = Ubuntu RAM = 312 Disco = 5	Correcta creación de la VM con Ubuntu y las características mínimas (1024MB RAM y 15GB de disco para los sistemas GNU/Linux y 25GB de disco en windows)	baja	Usuario	OK
PU_08	RF09 RF10	Consultar estado de las VMs	1- Ingresar a la web <u>http://localhost:8080/</u> estados	-	Correcta visualización del estado de las VM	Media	Usuario	OK

Tabla 8: Casos de prueba del usuario del sistema - Parte 2

PU_09	RF09 RF10	Modificar estado de una VM	1- Ingresar a la web http://1localhost: 8080/ estados 2- Introducir los datos necesarios	Correcto encendido y apagado de la VM	Media	Usuario	OK	
PU_10	RF06 RF07 RF09	Añadir políticas de las VM	1- Ingresar a la web http://1localhost: servicios 2- Introducir los datos necesarios	Marcar las siguientes opciones en la sección Centos: Eclipse idle repositorio update usuarios	Adecuación correcta del estado de la máquina virtual Centos de acuerdo a lo pedido	Alta	Usuario	OK
PU_11	RF09	Ingreso Interfaz web desde máquinas en red	1- Ingresar a la web http://192.168.1.137. 8080/servicios	-	Correcto ingreso a la interfaz web desde una pc en la red	baja	Usuario	OK

Tabla 9: Casos de prueba del usuario del sistema - Parte 3

4. Conclusión

 La idea de este proyecto surge luego de finalizar la Práctica Profesional Supervisada, donde se tuvo una primera inserción a temas relacionados con servidores y recursos de red, implementando un sistema de monitorización de estos dispositivos.

Se comenzó este proyecto con escasos conocimientos acerca de virtualización, orquestación, programación en HTML, programación en Python, etc. Sin embargo, para lograr un desarrollo exitoso no es imprescindible conocer profundamente los aspectos técnicos. Para lograrlo se debe saber cómo enfrentarse al proceso, usar una metodología eficiente, tener una excelente comunicación con el cliente, contar con un buen equipo de trabajo, saber enfrentarse a temas desconocidos y ser proactivo.

El análisis formal del problema para la obtención de los requerimientos y los riesgos del proyecto, es algo que también debe destacarse. Esta es una fase imprescindible para poder llevar a cabo las estimaciones pertinentes a los tiempos de desarrollo e investigación de cualquier proyecto. En muchas ocasiones se cuenta con diferentes herramientas para llevar a cabo una misma tarea. El análisis de cada una de ellas y su elección, utilizando factores de decisión ponderados, es fundamental para el trabajo como ingeniero. De aquí también se puede hacer notar que la utilización de soluciones de código abierto no siempre permiten estar en la "cresta de la ola" tecnológica y muchas veces es necesario contar con software privativo para ~~sacar~~ la máxima producción.

Este Proyecto Integrador ha consolidado buena parte de los conocimientos ~~adquiridos~~ y, por su puesto, ha añadido muchos otros. La posibilidad de poner en práctica e integrar material de diversas materias ha sido beneficioso para el ejercicio de nuestra profesión en el ámbito académico y laboral.

5. Trabajos Futuros

Como en todo proyecto, existen factores que limitan su alcance. En este proyecto, la principal limitación fue el tiempo. Es por esta limitación que existen mejoras que se le pueden aplicar en un futuro, dando lugar a un Proyecto Integrador o Práctica Profesional Supervisada de otro alumno, por ejemplo.

Los siguientes son algunos de los trabajos futuros que se podrían llevar a cabo.

- **Protección:** Valoración de la probabilidad de que el sistema pueda resistir intrusiones accidentales o premeditadas. Para mejorar esta probabilidad se propone:
 - Modificar el sistema para que funcione con *firewall* y SELinux.
 - Incluir validación por usuario en la interfaz web.
 - Incluir un *log* de cambios al sistema que permita saber quién y qué cambio realizó.
- **Tolerancia a errores:** La tolerancia a errores refleja hasta qué punto el sistema se diseñó para evitar y tolerar errores. En las aplicaciones desarrolladas se introdujeron porciones de código que las protegen del mal funcionamiento. Sin embargo, esto está lejos de la perfección y muchos errores quedan sin reconocimiento, por lo cual, incluir más secciones dedicadas a subsanar errores es una interesante mejora.

- Las pruebas realizadas sobre las herramientas, y las pruebas de los resultados finales de las aplicaciones no fueron ejecutadas sobre equipos servidores dado que no se contaba con acceso a ellos. éstas pruebas se efectuaron sobre los equipos personales de escritorio y notebooks, quedando como tema pendiente la implementación de estos pasos en un servidor de alto rendimiento para aumentar el volumen de nodos administrados.

6. Bibliografía

[1] KVM

- Documentación Oficial de KVM de Red Hat Enterprise
- <http://www.linux-kvm.org/page/Documents>
- <http://linux.die.net/man/1/virt-install>
- <http://linux.die.net/man/1/virsh>

[2] Cobbler

- <http://cobbler.github.io/manuals/2.6.0/>
- <https://fedorahosted.org/cobbler/wiki/UserDocs>

[3] Kickstart

- [https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linu](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux)

[4] Puppet

- <https://docs.puppet.com/puppet/>
- <https://docs.puppet.com/windows/>

[5] Python-Bottle

- <http://bottlepy.org/docs/dev/index.html>
- <http://docs.python.org/dev/library/>

[6] Windows 7

- <https://technet.microsoft.com/es-es/library/dd744547%28v=ws.10%29.aspx>
- <https://technet.microsoft.com/es-ar/library/dd744263%28v=ws.10%29.aspx>
- <https://technet.microsoft.com/es-ar/library/ee523217%28v=ws.10%29.aspx>
- <https://technet.microsoft.com/es-es/library/dd744393%28v=ws.10%29.aspx>
- <https://technet.microsoft.com/es-es/library/dd744393%28v=ws.10%29.aspx>
- <https://technet.microsoft.com/es-ar/library/gg241188%28v=ws.10%29.aspx>
- <https://technet.microsoft.com/es-es/library/dd744317%28v=ws.10%29.aspx>
- <https://support.microsoft.com/es-ar/kb/973289>
- <http://www.letifer.org/2014/03/26/cobbler-and-windows/>

[7] PXE

- <ftp://download.intel.com/design/archives/wfm/downloads/pxespec.pdf>

[8] Ingeniería de Software

- <http://www.slideshare.net/noriver/desarrollo-iterativo-e-incremental>
- <http://www.javiergarzas.com/metodologias-agiles>
- http://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software
- http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-ii/m_gestionRiesgos.pdf
- http://www.dlsiis.fi.upm.es/docto_lsiis/Trabajos20052006/Gasca.pdf
<http://www.leobravo.cl/programas/recursos/Aprendiendo-uml-en-24-horas>

[9] Python

- <http://campusurico.utalca.cl/~rgarrido/recursos/python2.pdf>
- <http://stackoverflow.com/questions/3730964/python-script-execute-command-terminal>
- <http://www.cyberciti.biz/faq/python-execute-unix-linux-command-example/>
- <http://docs.python.org/dev/library/subprocess.html#module-subprocess>

[10] SAMBA

- https://es.wikipedia.org/wiki/Samba_%28programa%29
- <https://www.samba.org/>
- <http://www.ite.educacion.es/formacion/materiales/85/cd/linux/m4/servi>
- <http://www.techbrown.com/samba-server-configuration-centos-7-rhel-7.s>

[11] Ubuntu

- <https://wiki.debian.org/AutomatedInstallation>
- <https://help.ubuntu.com/14.04/installation-guide/amd64/index.html>
- <http://www.sanitarium.net/golug/netboot.html>
- <https://help.ubuntu.com/12.04/installation-guide/example-preseed.txt>