

UNIVERSIDAD DEL VALLE

ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

VIRTShell - Framework para Aprovisionamiento de Soluciones Virtuales

TESIS PRESENTADA POR CARLOS ALBERTO LLANO RODRÍGUEZ
PARA OBTENER EL GRADO DE MAESTRÍA EN INGENIERÍA CON ÉNFASIS
EN INGENIERÍA DE SISTEMAS

2015

Facultad de Ingeniería

Agradecimientos

- A Jhon Alexander Sanabria, profesor de la Facultad de Ingeniería de la Universidad del Valle, por su inmensa paciencia y apoyo a lo largo de todo el proyecto.
- A mis amigos colaboradores de la Universidad del Valle, por todo el apoyo y por creer siempre en este gran esfuerzo.
- A mis compañeros y amigos por sus palabras de aliento y constante apoyo.
- A mi familia sin ella, no hubiera podido alcanzar esta meta.
- A todas aquellas personas que de una u otra forma colaboraron en la realización del presente trabajo.

Índice general

Agradecimientos	2
1. Introducción	7
2. Aprovisionamiento de recursos virtuales	8
2.1. Alternativas de despliegue actuales	10
2.1.1. Nixes	10
2.1.2. SmartFrog	11
2.1.3. Radia	11
2.1.4. Cobbler	12
2.1.5. Amazon EC2	12
2.1.6. HP Utility Data Center	12
2.1.7. Oracle VM Templates	12
2.1.8. Chef	13
2.1.9. Puppet	13
2.1.10. Cfengine	13
2.1.11. Bcfg2	14
3. API REST	15
3.1. Formato de entrada y salida	15

3.2. Autenticación	16
3.2.1. Authentication Header	16
3.2.2. Solicitud canónica para firmar	18
3.2.3. Tiempo de sello	18
3.2.4. Ejemplos de autenticación	18
3.3. Codigos de error	19
3.4. Hosts	20
3.4.1. Ejemplos de peticiones HTTP	22
3.5. Files	27
3.5.1. Ejemplos de peticiones HTTP	28
3.6. Images	31
3.6.1. Ejemplos de peticiones HTTP	33
3.7. Packages	36
3.7.1. Ejemplos de peticiones HTTP	37
3.8. Properties	39
3.8.1. Ejemplos de peticiones HTTP	41
 4. Recomendaciones	 45
 Bibliografía	 46

Índice de figuras

Índice de cuadros

CAPÍTULO 1

Introducción

CAPÍTULO 2

Aprovisionamiento de recursos virtuales

La computación en la nube ha sido un punto importante de investigación en la industria recientemente. Esta puede ser descrita como una nueva clase de computación en la cual dinámicos y escalables recursos pueden ser provistos sobre internet. Para los usuarios esto es transparente y ellos solo pagan lo que usan de acuerdo a niveles de servicio establecidos con los proveedores de nubes.

Una de las principales características de la computación en la nube es la virtualización, la cual consiste en crear una versión virtual de un recurso tecnologico en lugar de usar una versión física. La virtualización se puede aplicar a computadoras, sistemas operativos, dispositivos de almacenamiento de información, aplicaciones o redes permitiendo que las empresas ejecuten más de un sistema virtual, además de múltiples sistemas operativos y aplicaciones, en un único servidor, de esta manera se logra economía de escala y una mayor eficiencia.

En la actualidad predominan dos tecnicas de virtualizacion, la primera tecnica se denomina virtualización de hardware y consiste en crear un hardware sintético el cual usan las maquinas virtuales como propio, la idea es virtualizar el sistema opera-

tivo completo el cual se ejecuta sobre un software llamado el hipervisor, su función es interactuar directamente con la CPU en el servidor físico, ofreciendo a cada uno de los servidores virtuales una total autonomía e independencia. Incluso pueden coexistir en una misma máquina distintos servidores virtuales funcionando con distintos sistemas operativos. Esta técnica es la mas desarrollada y hay diferentes clases que cada fabricante ha ido desarrollando y adaptando, como por ejemplo Xen, KVM, VMWare y VirtualBox.

La segunda tecnica es conocida como virtualización del sistema operativo. En esta técnica lo que se virtualiza es el sistema operativo completo el cual corre directamente virtual sobre la máquina física. En esta técnica las maquinas virtuales son llamadas contenedores, los cuales acceden por igual a todos los recursos del sistema. La ventaja es a su vez una desventaja: Todas las maquinas virtuales usan el mismo Kernel que el sistema operativo lo que reduce mucho los errores y multiplica el rendimiento, pero a su vez solo puede haber un mismo tipo de sistema operativo en los contenedores, no se puede mezclar Windows-Linux-Etc. Este sistema, también se acerca mucho a lo que seria una virtualización nativa.

Sin importar la tecnica de virtualización que se use, la instalación de una maquina virtual (o de un contenedor) requiere normalmente de la generación e instalación de una imagen y la instalación y configuración de paquetes de software. Estas tareas generalmente son realizadas por técnicos de los proveedores de la nube. Cuando un usuario de la nube solicita un nuevo servicio o mas capacidad de computo, el administrador selecciona la apropiada imagen para clonar e instalar en los nodos de la nube. Si no hay una imagen apropiada para los requerimientos del cliente se crea y configura una nueva que cumpla con la solicitud. Esta creación de una nueva imagen puede ser realizada modificando la imagen mas cercada de las ya existentes. En el momento de la creación optima de la imagen un administrador puede tener dificultades y preguntás como, cual es la mejor configuración?, cuales paquetes y sus dependencias deberían ser instaladas? y como encontrar una imagen que mejor llene las expectativas?.

Es por esta razón que los proveedores de la nube desean cada vez mas automatizar y simplificar este proceso porque la dependencia entre paquetes de software y la dificultad de mantenimiento agrega tiempo a la creación de las maquinas virtuales. En otras palabras los proveedores de nube quieren dar mas flexibilidad y agilidad a la hora de satisfacer los requerimientos de los usuarios finales.

Existen varias soluciones que permiten la interacción con los diferentes ambientes de virtualización. Estas soluciones usan diferentes enfoques para realizar despliegues de software en las maquinas virtuales, que dan un rápido, controlado y automático despliegue de software, en todas las maquinas de una red físicas o virtualizadas, permitiendo mejorar los tiempos de instalación de nuevas funcionalidades de forma confiable y segura de la misma forma que ayudan a disminuir el tiempo y el costo de los despliegues de aplicaciones y servicios. Sin embargo no todas las soluciones son de código abierto, algunas son de desarrollo propietario, en donde solo ofrecen el API al público pero no el código de la solución como tal y manejan sus propias herramientas de virtualización.

2.1. Alternativas de despliegue actuales

En esta sección, se describirán las herramientas mas significativas que existen indicando las ventajas y desventajas de cada una.

2.1.1. Nixes

Nixes es una herramienta usada para instalar, mantener, controlar y monitorear aplicaciones en PlanetLab [1]. Nixes consiste de un conjunto de scripts bash, un archivo de configuración, y un repositorio web, y puede automaticamente instalar, actualizar y resolver dependencias solo de paquetes RPM.

Para sistemas de pequeña escala, Nixes es fácil de usar: los usuarios simplemente crean el archivo de configuración para cada aplicación y modifican los scripts a desplegar en los nodos. Pero para grandes y complejos sistemas, Nixes no es efectivo, porque el no provee un mecanismo automático de flujo de trabajo.

2.1.2. SmartFrog

SmartFrog (SF) es un framework para servicios de configuración, descripción, despliegue y administración del ciclo de vida. Consiste de un lenguaje declarativo, un motor que corre en los nodos remotos y ejecuta plantillas escritas en el lenguaje de SmartFrog y un modelo de componentes. El lenguaje soporta encapsulación (que es similar a las clases de python), herencia y composición que permite personalizar y combinar configuraciones. SmartFrog, permite enlaces estáticos y dinámicos entre componentes, que ayudan a soportar diferentes formas de conexión en tiempo de despliegue.

El modelo de componentes SF, administra el ciclo de vida a través de cinco estados: instalado, iniciado, terminado y fallido. Esto permite al motor del SmartFrog detectar fallas y reiniciar automáticamente re-despliegues de los componentes [2]. SmartFrog es desarrollado y mantenido por un equipo de investigación en los laboratorios de Hewlett-Packard en Bristol, Inglaterra, así como por el laboratorio Europeo de Hewlett-Packard y adicional con contribuciones de otros usuarios de SmartFrog y desarrolladores externos a HP. Se utiliza en la investigación de HP específicamente en la automatización de la infraestructura y automatización de servicios, además de ser utilizado en determinados productos de HP.

2.1.3. Radia

Herramienta de administración de cambios que utiliza un enfoque basado en modelos [3]. Para cada dispositivo administrado, el administrador define un estado deseado, el cual es mantenido como un modelo en un repositorio central. Nixes, usa seis maquinascálculosmodelos: paquete (configuración, instalación, entradas de registro, binarios, entre otras); mejores prácticas; dependencias de software (relaciones con otros componentes de software, sistemas operativos y hardware); infraestructura (servidores, almacenamiento y elementos de red); inventarios de software (software instalado actualmente) e interoperabilidad entre modelos de servicios administrados.

2.1.4. Cobbler

Cobbler es una plataforma que busca el rápido despliegue de servidores y en general computadores en una infra-estructura de red, se basa en el modelo de scripts y cuenta con una completa base de simples comandos, que permite hacer despliegues de manera rápida y con poca intervención humana. Cobbler al igual que SmartFrog es capaz de instalar máquinas físicas y máquinas virtuales. Cobbler, es una pequeña y ligera aplicación, que es extremadamente fácil de usar para pequeños o muy grandes despliegues. [?]

2.1.5. Amazon EC2

Amazon EC2 es un API propietario de Amazon y maneja un enfoque manual, que permite desplegar imágenes de máquinas virtuales conocidas como AMI (Amazon Machine Images) [?], que son las imágenes que se utilizan en Amazon para arrancar instancias. El concepto de las amis es similar a las máquinas virtuales de otros sistemas. Básicamente están compuestas de una serie ficheros de datos que conforman la imagen y luego un xml que especifica ciertos valores necesarios para que sea una imagen válida para Amazon que es el image.manifest.xml.

2.1.6. HP Utility Data Center

HP Utility Data Cente (UDC) es un producto comercial, que se centra en la administración automatizada de servidores de red, usando el concepto de "infraestructura programable ". Los elementos de hardware, como nodos de servidores, switches, firewalls y elementos de almacenamiento, son cableados en una infraestructura de configuración. El software de administración UDC permite configurar combinaciones de estos componentes en servidores virtuales usando cableados virtuales. [?]

2.1.7. Oracle VM Templates

Oracle VM Templates, es un producto comercial de la empresa Oracle, cuyo objetivo es realizar despliegues rapidos de aplicaciones Oracle y no-Oracle, con base

en imágenes de software pre-configuradas manualmente. Cuenta con una interfaz gráfica que permite crear y administrar servidores virtuales con facilidad. [?]]

2.1.8. Chef

Chef es una herramienta de gestión de la configuración escrito en Ruby y Erlang. Utiliza un lenguaje de dominio específico escrito también en Ruby para la escritura y configuración de recetas”. Estas recetas contienen los recursos que deben ser creados. Chef se puede integrar con plataformas basadas en la nube, como Rackspace, Internap, Amazon EC2, Cloud Platform Google, OpenStack, SoftLayer y Microsoft Azure. Chef contiene soluciones para sistemas de pequeña y gran escala. [4]

Es uno de los cuatro principales sistemas de gestión de la configuración en Linux, junto con Cfengine, Bcfg2 y Puppet.

2.1.9. Puppet

Puppet es una herramienta diseñada para administrar la configuración de sistemas similares a Unix y a Microsoft Windows de forma declarativa. 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. Puppet descubre la información del sistema a través de una utilidad llamada Facter, y compila los manifiestos en un catálogo específico del sistema que contiene los recursos y la dependencia de dichos recursos, estos catálogos son ejecutados en los sistemas de destino. [5]

2.1.10. Cfengine

Cfengine es un sistema basado en el lenguaje escrito por Mark Burgess, diseñado específicamente para probar y configurar software. Cfengine es como un lenguaje de muy alto nivel. La idea de Cfengine es crear un único archivo o conjunto de archivos de configuración que describen la configuración de cada host de la red. Cfengine se ejecuta en cada host, y analiza cada archivo (o archivos), que especifica una política

para la configuración del sistema; la configuración del host es verificada contra el modelo y, si es necesario, cualquier desviación de la configuración es corregida. [6]

2.1.11. Bcfg2

Bcfg2 está escrito en Python y permite gestionar la configuración de un gran número de ordenadores mediante un modelo de configuración central. Bcfg2 funciona con un modelo simple de configuración del sistema, modelando elementos intuitivos como paquetes, servicios y archivos de configuración (así como las dependencias entre ellos). Este modelo de configuración del sistema se utiliza para la verificación y validación, permitiendo una auditoría robusta de los sistemas desplegados. La especificación de la configuración de Bcfg2 está escrita utilizando un modelo XML declarativo. Toda la especificación puede ser validada utilizando los validadores de esquema XML ampliamente disponibles. [7]

CAPÍTULO 3

API REST

Este capítulo está destinado a los desarrolladores que deseen interactuar con el VirtShell API, para realizar aprovisionamientos automáticos desde cualquier plataforma de desarrollo. El VirtShell API es un API REST que provee acceso a los objetos en el VirtShell Server, esto incluye los hosts, imágenes, archivos, templates, aprovisionadores y usuarios. Por medio del API podrá crear ambientes, máquinas virtuales y contenedores personalizados, realizar configuraciones y administrar los recursos físicos y virtuales de manera programática.

3.1. Formato de entrada y salida

El API solo soporta el formato json para intercambio de información. Cualquier solicitud que no se encuentre en formato json resultará en un error con código 406 (Content Not Acceptable Error).

3.2. Autenticación

La autenticación es el proceso de demostrar la identidad al sistema. La identidad es un factor importante en las decisiones de control de acceso. Las solicitudes se conceden o deniegan en parte sobre la base de la identidad del solicitante.

El VirtShell API REST utiliza un esquema HTTP personalizado basado en una llave-HMAC (Hash Message Authentication Code) para la autenticación. Para autenticar una solicitud, primero se concatenan los elementos seleccionados de la solicitud para formar una cadena. A continuación, utiliza una clave secreta de acceso para calcular el HMAC de esa cadena. Informalmente, llamamos a este proceso la firma de la solicitud; y llamamos a la salida del algoritmo HMAC la firma; ya que simula las propiedades de seguridad de una firma real. Por último, se agrega esta firma como un parámetro de la petición, con la sintaxis descrita en esta sección.

Cuando el sistema recibe una solicitud fehaciente, se obtiene la clave secreta de acceso que dicen tener, y lo utiliza de la misma manera que se calcula una firma del mensaje que recibió. A continuación, compara la firma que se calcula con la firma presentada por el solicitante. Si las dos firmas coinciden, el sistema llega a la conclusión de que el solicitante debe tener acceso a la clave secreta de acceso, y por lo tanto actúa con la autoridad del principal al que se emitió la clave. Si las dos firmas no coinciden, la solicitud se descarta y el sistema responde con un mensaje de error.

Ejemplo de una petición autenticada:

```
GET /api/virtshell/appliance/{applianceId} HTTP/1.1
```

```
Host: host1.edu.co
```

```
Date: Fri, 01 Jul 2011 19:37:58 +0000
```

```
Authorization: 0PN5J17HBGZHT7JJ3X82:frJIUN8DYpKDtOLCwo//yllqDzg=
```

3.2.1. Authentication Header

El API REST utiliza el encabezado de autorización HTTP estándar para pasar información de autenticación. (El nombre de la cabecera estándar es insuficiente, ya que solo lleva la información de autenticación y no la de autorización). Bajo

el esquema de autenticación de VirtShell, el encabezado de autorización tiene la siguiente forma.

```
Authorization: JanuUserId:Signature
```

Los usuarios tendrán un ID de clave de acceso (VirtShell Access Key ID) y una clave secreta de acceso (VirtShell Secret Access Key) cuando se registran. Para la petición de autenticación, el elemento de VirtShell Access Key Id identifica la clave secreta que se utilizó para calcular la firma, y (indirectamente) el usuario que realiza la solicitud.

Para la firma de los elementos de la petición se usa el RFC 2104HMAC-SHA1, por lo que la parte de la firma de la cabecera autorización variará de una petición a otra. Si la solicitud de la firma calculada por el sistema coincide con la firma incluida en la solicitud, el solicitante habrá demostrado la posesión de la clave secreta de acceso. La solicitud será procesada bajo la identidad, y con la autoridad, de la promotora que se emitió la clave.

A continuación se muestra la pseudo-gramática que ilustra la construcción de la cabecera de la solicitud de autorización (\n significa el punto de código Unicode U +000A comúnmente llamado salto de línea).

```
Authorization = VirtShellUserId + ":" + Signature;
```

```
Signature = Base64( HMAC-SHA1( UTF-8-Encoding-Of( YourSecretAccessKeyID
```

```
StringToSign = HTTP-Verb + "\n" +
Host + "\n" +
Content-MD5 + "\n" +
Content-Type + "\n" +
Date + "\n" +
CanonicalizedResource;
```

```
CanonicalizedResource = <HTTP-Request-URI, from the protocol name up
```

HMAC-SHA1 es un algoritmo definido por la RFC 2104 (ver la RFC 2104 con llave Hashing para la autenticación de mensajes). El algoritmo toma como entrada

dos cadenas de bytes: una clave y un mensaje. Para la solicitud de autenticación, se utiliza la clave secreta (YourSecretAccessKeyID) como la clave, y la codificación UTF-8 del StringToSign como el mensaje. La salida de HMAC-SHA1 es también una cadena de bytes, llamado el resumen. El parámetro de la petición de la Firma se construye codificada en Base64.

3.2.2. Solicitud canónica para firmar

Cuando el sistema recibe una solicitud autenticada, compara la solicitud de firma calculada con la firma proporcionada en la solicitud de StringToSign. Por esta razón, se debe calcular la firma con el mismo método utilizado por VirtShell. A este proceso de poner una solicitud en una forma acordada para la firma se denomina "canonización".

3.2.3. Tiempo de sello

Un sello de tiempo válido (utilizando el HTTP header Date) es obligatorio para solicitudes autenticadas. Por otra parte, el tiempo del sello enviado por un usuario que se encuentra incluido en una solicitud autenticada debe estar dentro de los 15 minutos de la hora del sistema cuando se recibe la solicitud. En caso contrario, la solicitud fallará con el código de estado de error RequestTimeTooSkewed. La intención de estas restricciones es limitar la posibilidad de que solicitudes interceptadas pueden ser reproducidos por un adversario. Para una mayor protección contra las escuchas, se debe utilizar el transporte HTTPS para solicitudes autenticadas.

3.2.4. Ejemplos de autenticación

Parametro	Valor
VirtShellUserId	13010f3e-3f46-4889-b989-592ce8fb30c6
VirtShellSecretAccessKey	c991f519-bed0-4dab-9165-6d9f722dc845
Base64:	Yzk5MWY1MTktYmVkMC00ZGFjLTkxNjUtNmQ5ZjcyMmRjODQ1

Ejemplo de un objeto con GET

Este es un ejemplo que consulta por un enviroment dado su identificador.

Request	StringToSign
GET /api/virtshell/enviroment/45 HTTP/1.1 Host: host1.edu.co Date: Tue, 27 Mar 2007 19:36:42 +0000 Authorization: 13010f3e-3f46-4889-b989-592ce8fb30c6: Yzk5MWY1MmVkMC00ZGFiLTtNmQ5ZjcyMmRjODQ1	GET\n host1.edu.co\n \n \n Tue, 27 Mar 2007 19:36:42 +0000 /api/virtshell/enviroment/45

Ejemplo de un objeto con DELETE

Este ejemplo remueve un usuario.

Request	StringToSign
DELETE /api/virtshell/user/9876 HTTP/1.1 Host: host1.edu.co Date: Tue, 27 Mar 2007 21:20:27 +0000 Authorization: 13010f3e-3f46-4889-b989-592ce8fb30c6: Yzk5MWY1MmVkMC00ZGFiLTtNmQ5ZjcyMmRjODQ1	DELETE\n host1.edu.co\n \n \n Tue, 27 Mar 2007 21:20:27 +0000 /api/virtshell/user/9876

3.3. Codigos de error

Aqui se presenta una lista de codigos de error que pueden resultar de una petición al API en cualquier recurso.

- **400 Bad Request** La solicitud no pudo ser procesada con éxito porque el URI no era válido. El cuerpo de la respuesta contendrá una razón del fracaso de la petición. Esta respuesta indica error permanente.
- **403 Forbidden** La solicitud no pudo ser procesada con éxito porque la identidad del usuario no tiene acceso suficiente para procesar la solicitud. Esta respuesta indica error permanente.
- **406 Content Not Acceptable** Un recurso genera este error de acuerdo al tipo de cabeceras enviadas en la petición. Esta respuesta indica un error permanente e indica un formato de salida no soportado. La respuesta de este tipo de error

no contiene un contenido debido a la inhabilidad del servidor para generar una respuesta en el formato solicitado.

- **404 Not Found** La solicitud no pudo ser procesada con éxito porque la solicitud no era válida. Lo más probable es que no se encontrara la url. Esta respuesta indica error permanente.
- **500 Server Error** La solicitud no pudo ser procesada debido a que el servidor encontró una condición inesperada que le impidió cumplir con la petición.
- **501 Not Implemented** La solicitud no se pudo completar porque el servidor o bien no reconoce el método de petición o el recurso solicitado no existe.

Los errores que no sean de código 406 (Content Not Acceptable) contienen una respuesta en formato json, que contiene un breve mensaje explicado el error con más detalle. Por ejemplo, una consulta POST /virtshell/api/v1/hosts, con un cuerpo vacío, daría lugar a la siguiente respuesta:

```
1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json
3
4 {"error": "Missing input for create instance"}
```

3.4. Hosts

Representan las máquinas físicas; un host es un anfitrión de máquinas virtuales o contenedores. Los métodos soportados son:

Acción	Metodo HTTP	Solicitud HTTP	Descripción
get	GET	/hosts/id	Gets one host by ID.
list	GET	/hosts	Retrieves the list of hosts.
create	POST	/hosts/	Inserts a new host configuration.
delete	DELETE	/hosts/id	Deletes an existing host.
update	PUT	/hosts/id	Updates an existing host.

Representación del recurso de un host:

```
1 {  
2   "uuid": string,  
3   "name": string,  
4   "os": string,  
5   "memory": string,  
6   "capacity": string,  
7   "enabled": string,  
8   "type": string,  
9   "local_ipv4": string,  
10  "local_ipv6": string,  
11  "public_ipv4": string,  
12  "public_ipv6": string,  
13  "instances": [ instance_resource ],  
14  "created": [ "at": number, "by": number ]  
15 }
```

Ejemplo:

```
1 {  
2   "uuid": "ab8076c0-db91-11e2-82ce-0002a5d5c51b",  
3   "name": "host-01-pdn",
```

```
4  "os": "Ubuntu_12.04_3.5.0-23.x86_64",
5  "memory": "16GB",
6  "capacity": "120GB",
7  "enabled": "true|false",
8  "type": "StorageOptimized|GeneralPurpose|HighPerformance",
9  "local_ipv4": "15.54.88.19",
10 "local_ipv6": "ff06:0:0:0:0:0:0:c3",
11 "public_ipv4": "10.54.88.19",
12 "public_ipv6": "yt06:0:0:0:0:0:0:c3",
13 "instances": [
14     ... instances resource is here
15 ],
16 "created": ["at": "timestamp", "by": 1234]
17 }
```

3.4.1. Ejemplos de peticiones HTTP

3.4.1.1. Crear un nuevo host - POST /virtshell/api/v1/hosts

```
1 curl -sv -X POST \
2   -H 'accept: application/json' \
3   -H 'X-VirtShell-Authorization: UserId:Signature' \
4   -d '{"name": "host-01-pdn",
5       "os": "Ubuntu_12.04_3.5.0-23.x86_64",
6       "memory": "16GB",
7       "capacity": "120GB",
8       "enabled": "true",
9       "type": "GeneralPurpose",
10      "local_ipv4": "15.54.88.19",
11      "local_ipv6": "ff06:0:0:0:0:0:0:c3",
12      "public_ipv4": "10.54.88.19",
13      "public_ipv6": "yt06:0:0:0:0:0:0:c3"}' \
```

```
14 'http://localhost:8080/virtshell/api/v1/hosts'
```

Respuesta:

```
1 HTTP/1.1 200 OK
2 Content-Type: application/json
3 { "create": "success" }
```

3.4.1.2. Obtener un host- GET /virtshell/api/v1/hosts/:id

```
1 curl -sv -H 'accept: application/json'
2     -H 'X-VirtShell-Authorization: UserId:Signature' \
3     'http://localhost:8080/api/virtshell/v1/hosts?id=
      ab8076c0-db91-11e2-82ce-0002a5d5c51b'
```

Respuesta:

```
1 HTTP/1.1 200 OK
2 Content-Type: application/json
3 {
4   "uuid": "ab8076c0-db91-11e2-82ce-0002a5d5c51b",
5   "name": "host-01-pdn",
6   "os": "Ubuntu_12.04_3.5.0-23.x86_64",
7   "memory": "16GB",
8   "capacity": "120GB",
9   "enabled": "true",
10  "type": "StorageOptimized",
11  "local_ipv4": "15.54.88.19",
```

```
12  "local_ipv6": "ff06:0:0:0:0:0:0:c3",
13  "public_ipv4": "10.54.88.19",
14  "public_ipv6": "yt06:0:0:0:0:0:0:c3",
15  "instances": [
16    {
17      "name": "name1",
18      "id": "72C05559-0590-4DA6-BE56-28AB36CB669C"
19    },
20    {
21      "name": "name2",
22      "id": "17173587-C4E9-4369-9C43-FCBF5E075973"
23    }
24  ],
25  "created": ["at": "20130625105211", "by": 10]
26 }
```

3.4.1.3. Obtener todos los host - GET /virtshell/api/v1/hosts

```
1 curl -sv -H 'accept: application/json'
2   -H 'X-VirtShell-Authorization: UserId:Signature' \
3   'http://localhost:8080/api/virtshell/v1/hosts'
```

Respuesta:

```
1 HTTP/1.1 200 OK
2 Content-Type: application/json
3 {
4   "hosts": [
5     {
6       "uuid": "ab8076c0-db91-11e2-82ce-0002a5d5c51b",
```



```
7     "name": "host-01-pdn",
8     "os": "Ubuntu_12.04_3.5.0-23.x86_64",
9     "memory": "16GB",
10    "capacity": "120GB",
11    "enabled": "true",
12    "type" : "StorageOptimized",
13    "local_ipv4": "15.54.88.19",
14    "local_ipv6": "ff06:0:0:0:0:0:0:c3",
15    "public_ipv4": "10.54.88.19",
16    "public_ipv6": "yt06:0:0:0:0:0:0:c3",
17    "instances": [
18        {
19            "name": "name1",
20            "id": "72C05559-0590-4DA6-BE56-28AB36CB669C"
21        },
22        {
23            "name": "name2",
24            "id": "17173587-C4E9-4369-9C43-FCBF5E075973"
25        }
26    ],
27    "created": ["at": "20130625105211", "by": 10]
28 },
29 {
30     "uuid": "ab8076c0-db91-11e2-82ce-0002a5d5c51b",
31     "name": "host-01-pdn",
32     "os": "Ubuntu_12.04_3.5.0-23.x86_64",
33     "memory": "16GB",
34     "capacity": "120GB",
35     "enabled": "true",
36     "type" : "GeneralPurpose",
37     "local_ipv4": "15.54.88.19",
38     "local_ipv6": "ff06:0:0:0:0:0:0:c3",
```

```
39     "public_ipv4": "10.54.88.19",
40     "public_ipv6": "yt06:0:0:0:0:0:0:c3",
41     "instances": [
42         {
43             "name": "name3",
44             "id": "DE11CC9A-482F-4033-A7F8-503EE449DD0A"
45         },
46         {
47             "name": "name4",
48             "id": "17173587-C4E9-4369-9C43-FCBF5E075973"
49         },
50     ],
51     "created": ["at": "20130625105211", "by": 10]
52 }
53 ]
54 }
```

3.4.1.4. Actualizar un host - PUT /virtshell/api/v1/hosts/:id

```
1 curl -sv -X PUT \
2     -H 'accept: application/json' \
3     -H 'X-VirtShell-Authorization: UserId:Signature' \
4     -d '{"memory": "24GB",
5         "capacity": "750GB"}' \
6     'http://localhost:8080/api/virtshell/v1/hosts?id=
       ab8076c0-db91-11e2-82ce-0002a5d5c51b'
```

Respuesta:

```
1 HTTP/1.1 200 OK
```

```
2 Content-Type: application/json
3
4 { "update": "success" }
```

3.4.1.5. Eliminar un host - DELETE /virtshell/api/v1/hosts/:id

```
1 curl -sv -X DELETE \
2     -H 'accept: application/json' \
3     -H 'X-VirtShell-Authorization: UserId:Signature' \
4     'http://localhost:8080/api/virtshell/v1/hosts?id=
    ab8076c0-db91-11e2-82ce-0002a5d5c51b'
```

Respuesta:

```
1 HTTP/1.1 200 OK
2 Content-Type: application/json
3 ''
4 ''json
5 { "delete": "success" }
```

3.5. Files

Representan toda clase de archivos que se requieran para crear o aprovisionar máquinas virtuales o contenedores. Los métodos soportados son:

Acción	Metodo HTTP	Solicitud HTTP	Descripción
get	GET	/files/id	Gets one file by ID.
create	POST	/files/	upload a new file.
delete	DELETE	/files/id	Deletes an existing file.
update	PUT	/images/id	Updates an existing file.

Representación del recurso de un archivo:

```
1 {  
2   "uuid": "ab8076c0-db91-11e2-82ce-0002a5d5c51b",  
3   "name": "file_name.extension",  
4   "folder_name" : "folder_name",  
5   "download_url": "https://<host>:<port>/api/virtshell/v1  
   /files/folder_name/file.txt",  
6   "created": ["at": "timestamp", "by": user_id]  
7 }
```

Ejemplo:

```
1 {  
2   "uuid": "ab8076c0-db91-11e2-82ce-0002a5d5c51b",  
3   "name": "ubuntu_seed_14-04.tex",  
4   "folder_name" : "ubuntu_seeds",  
5   "download_url": "https://<host>:<port>/api/virtshell/v1  
   /files/ubuntu_seeds/ubuntu_seed_14-04.tex",  
6   "created": ["at": "20130625105211", "by": 10]  
7 }
```

3.5.1. Ejemplos de peticiones HTTP

3.5.1.1. Subir un nuevo archivo - POST /virtshell/api/v1/images

```
1 curl -X POST \  
2   -H 'accept: application/json' \  
3   -H 'X-VirtShell-Authorization: UserId:Signature' \  
4   -H "Content-Type: multipart/form-data" \  
5   -F "file_data=@/path/to/file/seed_file.txt;filename=  
   seed_file_ubuntu-14_04.txt" \  
6   -F "folder_name=ubuntu_seeds" \  
7
```

```
7 'http://<host>:<port>/api/virtshell/v1/files'
```

Respuesta:

```
1 HTTP/1.1 200 OK
2 Content-Type: application/json
3 {
4   "create": "success",
5   "location": "http://<host>:<port>/api/virtshell/v1/
6               files/ubuntu_seeds/seed_file_ubuntu-14_04.txt"
7 }
```

3.5.1.2. Obtener un archivo - GET /virtshell/api/v1/files/:id

Para descargar un archivo, primero recibirá la url apropiada que viene en la metadata provista por la url. Luego podrá descargarlo usando la url.

```
1 curl -sv -H 'accept: application/json'
2     -H 'X-VirtShell-Authorization: UserId:Signature' \
3     'http://<host>:<port>/api/virtshell/v1/files/?id=
4         ab8076c0-db91-11e2-82ce-0002a5d5c51b'
```

Respuesta:

```
1 HTTP/1.1 200 OK
2 Content-Type: application/json
3 {
4   "uuid": "ab8076c0-db91-11e2-82ce-0002a5d5c51b",
5   "name": "file_name.extension",
6 }
```

```
6  "folder_name" : "folder_name",
7  "download_url": "http://<host>:<port>/api/virtshell/v1/
    files/ubuntu_seeds/seed_file_ubuntu-14_04.txt",
8  "created":["at":"timestamp", "by":user_id]
9  }
```

3.5.1.3. Actualizar un archivo - PUT /virtshell/api/v1/files/:id

```
1  curl -sv -X PUT \
2    -H 'accept: application/json' \
3    -H 'X-VirtShell-Authorization: UserId:Signature' \
4    -H "Content-Type: multipart/form-data" \
5    -F "file_data=@/path/to/file/seed_file.txt;filename=
    seed_file_ubuntu-14_04_v2.txt" \
6    'http://localhost:8080/api/virtshell/v1/file?id=
    8de7b824-d7d1-4265-a3a6-5b46cc9b8ed5 '
```

Respuesta:

```
1  HTTP/1.1 200 OK
2  Content-Type: application/json
3
4  { "update": "success" }
```

3.5.1.4. Eliminar un archivo - DELETE /virtshell/api/v1/files/:id

```
1  curl -sv -X DELETE \
2    -H 'accept: application/json' \
3    -H 'X-VirtShell-Authorization: UserId:Signature' \
```

```
4 'http://localhost:8080/api/virtshell/v1/files?id=
  ab8076c0-db91-11e2-82ce-0002a5d5c51b'
```

Respuesta:

```
1 HTTP/1.1 200 OK
2 Content-Type: application/json
3 ''
4 '''json
5 { "delete": "success" }
```

3.6. Images

Representan imagenes de máquinas virtuales o contenedores. Los metodos soportados son:

Acción	Metodo HTTP	Solicitud HTTP	Descripción
get	GET	/images/id	Gets one image by ID.
list	GET	/images	Retrieves the list of images.
create	POST	/images/	Inserts a new image.
delete	DELETE	/images/id	Deletes an existing image.

Representación del recurso de una imagen:

```
1 {
2   "id": "kj5436c0-dc94-13tg-82ce-9992b5d5c51b",
3   "name": "ubuntu_server_14.04.2_amd64",
4   "type": "iso|container",
```

```
5  "os": "ubuntu",
6  "release": "trusty",
7  "version": "14.04.2",
8  "variant": "server|desktop",
9  "arch": "i386|amd64",
10 "timezone": "America/Bogota",
11 "ssh_key": "----- BEGIN PUBLIC KEY -----and a valid
    key here",
12 "preseed_url": "https://<host>:<port>/api/virtshell/v1/
    files/seeds/seed_ubuntu14-04.txt",
13 "download_url": "https://<host>:<port>/api/virtshell/v1/
    /files/images/3514296#file-lxc-ubuntu",
14 "created": ["at": "20130625105211", "by": 10],
15 "details": "...."
16 }
```

Ejemplo:

```
1  {
2    "id": "kj5436c0-dc94-13tg-82ce-9992b5d5c51b",
3    "name": "ubuntu_server_14.04.2_amd64",
4    "type": "iso",
5    "os": "ubuntu",
6    "release": "trusty",
7    "version": "14.04.2",
8    "variant": "server",
9    "arch": "amd64",
10   "timezone": "America/Bogota",
11   "preseed_url": "https://<host>:<port>/api/virtshell/v1/
    files/seeds/seed_ubuntu14-04.txt",
12   "created": ["at": "20150625105211", "by": 10]
13 }
```


3.6.1. Ejemplos de peticiones HTTP

3.6.1.1. Crear una nueva imagen - POST /virtshell/api/v1/images

```
1 curl -sv -X PUT \  
2   -H 'accept: application/json' \  
3   -H "Content-Type: text/plain" \  
4   -H 'X-VirtShell-Authorization: UserId:Signature' \  
5   -d '{"name": "ubuntu_server_14.04.2_amd64",  
6       "type": "iso",  
7       "os": "ubuntu",  
8       "release": "trusty",  
9       "version": "14.04.2",  
10      "variant": "server",  
11      "arch": "amd64",  
12      "timezone": "America/Bogota",  
13      "key": "/home/callanor/.ssh/id_rsa.pub",  
14      "preseed_url": "https://<host>:<port>/api/virtshell/  
15      v1/files/seeds/seed_ubuntu14-04.txt"}', \  
    'http://localhost:8080/api/virtshell/v1/image'
```

Respuesta:

```
1 HTTP/1.1 201 OK  
2 Content-Type: application/json  
3 { "create": "success" }
```

3.6.1.2. Obtener una imagen - GET /virtshell/api/v1/images/:id

```
1 curl -sv -H 'accept: application/json'  
2   -H 'X-VirtShell-Authorization: UserId:Signature' \  
3   -d ''
```

```
3 'http://localhost:8080/api/virtshell/v1/images?id=
  ab8076c0-db91-11e2-82ce-0002a5d5c51b'
```

Respuesta:

```
1 HTTP/1.1 200 OK
2 Content-Type: application/json
3 {
4   "id": "kj5436c0-dc94-13tg-82ce-9992b5d5c51b",
5   "name": "ubuntu_server_14.04.2_amd64",
6   "type": "iso",
7   "os": "ubuntu",
8   "release": "trusty",
9   "version": "14.04.2",
10  "variant": "server",
11  "arch": "amd64",
12  "timezone": "America/Bogota",
13  "preseed_url": "https://<host>:<port>/api/virtshell/v1/
    files/seeds/seed_ubuntu_14_04.txt",
14  "created": ["at": "20130625105211", "by": 10]
15 }
```

3.6.1.3. Obtener todas las imagenes - GET /virtshell/api/v1/images

```
1 curl -sv -H 'accept: application/json'
2   -H 'X-VirtShell-Authorization: UserId:Signature' \
3   'http://localhost:8080/api/virtshell/v1/images'
```

Respuesta:

```
1 HTTP/1.1 200 OK
2 Content-Type: application/json
3 {
4   "images": [
5     {
6       "id": "b180ef2c-e798-4a8f-b23f-aaac2fb8f7e8",
7       "name": "ubuntu_server_14.04.2_amd64",
8       "type": "iso",
9       "os": "ubuntu",
10      "release": "trusty",
11      "version": "14.04.2",
12      "variant": "server",
13      "arch": "amd64",
14      "timezone": "America/Bogota",
15      "preseed_file": "/home/callanor/seed_file.txt",
16      "created": ["at": "20130625105211", "by": 10]
17    },
18    {
19      "id": "ca326181-bc84-4edb-bfc5-843037e7195e",
20      "name": "centos_server",
21      "type": "container",
22      "os": "centos",
23      "version": "7",
24      "arch": "x86_64",
25      "download_url": "https://<host>:<port>/api/
        virtshell/v1/files/images/3514296#file-lxc-centos
        "
26      "created": ["at": "20140625105211", "by": 12]
27    }
28  ]
29 }
```

3.6.1.4. Eliminar una imagen - DELETE /virtshell/api/v1/images/:id

```
1 curl -sv -X DELETE \  
2   -H 'accept: application/json' \  
3   -H 'X-VirtShell-Authorization: UserId:Signature' \  
4   'http://<host>:<port>/api/virtshell/v1/images?id=  
    ab8076c0-db91-11e2-82ce-0002a5d5c51b'
```

Respuesta:

```
1 HTTP/1.1 200 OK  
2 Content-Type: application/json  
3 ''  
4 ''json  
5 { "delete": "success" }
```

3.7. Packages

Representan paquetes de software que se ejecutan en las máquinas virtuales o contenedores. Los métodos soportados son:

Acción	Metodo HTTP	Solicitud HTTP	Descripción
install	POST	/install_packages/	Install one or more packages.
upgrade	POST	/upgrade_packages/	Upgrade one or more packages.
remove	POST	/remove_packages/	Remove one or more packages.

Representación del recurso de un paquete:

```
1 {
2   "packages": [
3     {"name": "package_name1"},
4     {"name": "package_name2"}
5   ],
6   "hosts": [
7     {"name": "Host_", "range": "[1-3]"},
8     {"name": "database_001"}
9   ],
10  "tags": [
11    {"name": "db"},
12    {"name": "web"}
13  ]
14 }
```

Ejemplo:

```
1 {
2   "packages": [
3     {"name": "git"},
4     {"name": "nginx"}
5   ],
6   "hosts": [
7     {"name": "Host_", "range": "[1-3]"}
8   ]
9 }
```

3.7.1. Ejemplos de peticiones HTTP

3.7.1.1. Instalar uno o mas paquetes - POST /virtshell/api/v1/install_packages

```
1 curl -sv -X PUT \
2   -H 'accept: application/json' \
```

```
3 -H "Content-Type: text/plain" \  
4 -H 'X-VirtShell-Authorization: UserId:Signature' \  
5 -d '{ "packages": [{"name": "git"}, {"name": "nginx"}],  
6     "hosts": [{"name": "WebServer_", "range": "[1-3]"  
7         }]}', \  
    'http://localhost:8080/api/virtshell/v1/  
    install_packages '
```

Respuesta:

```
1 HTTP/1.1 202 Accepted  
2 Content-Type: application/json  
3 { "install_package": "accepted" }
```

3.7.1.2. Actualizar uno o mas paquetes - POST /virtshell/api/v1/upgrade_packages

```
1 curl -sv -X PUT \  
2 -H 'accept: application/json' \  
3 -H "Content-Type: text/plain" \  
4 -H 'X-VirtShell-Authorization: UserId:Signature' \  
5 -d '{ "packages": [{"name": "git"}, {"name": "nginx"},  
6     {"name": "mc"}],  
7     "hosts": [{"name": "WebServer_", "range": "[1-3]"  
        }]}', \  
    'http://localhost:8080/api/virtshell/v1/  
    upgrade_packages '
```

Respuesta:

```

1 HTTP/1.1 202 Accepted
2 Content-Type: application/json
3 { "install_package": "accepted" }

```

3.7.1.3. Remover uno o mas paquetes - POST /virtshell/api/v1/remove_packages

```

1 curl -sv -X PUT \
2   -H 'accept: application/json' \
3   -H "Content-Type: text/plain" \
4   -H 'X-VirtShell-Authorization: UserId:Signature' \
5   -d '{ "packages": [{"name": "apache2"}],
6         "hosts": [{"name": "WebServer_", "range": "[1-3]"
7         }]}', \
8   'http://localhost:8080/api/virtshell/v1/
9   remove_packages '

```

Respuesta:

```

1 HTTP/1.1 202 Accepted
2 Content-Type: application/json
3 { "install_package": "accepted" }

```

3.8. Properties

Representan propiedades de configuración de las máquinas virtuales o contenedores. Los metodos soportados son:

Acción	Metodo HTTP	Solicitud HTTP	Descripción
get	GET	/properties/	Install one or more packages.

Representación del recurso de un paquete:

```
1 {
2   "properties": [
3     {"name": "propertie_name1"},
4     {"name": "propertie_name2"}
5   ],
6   "hosts": [
7     {"name": "Host_", "range": "[1-3]"},
8     {"name": "database_001"}
9   ],
10  "tags": [
11    {"name": "db"},
12    {"name": "web"}
13  ]
14 }
```

Ejemplo:

```
1 {
2   "properties": [
3     {"name": "memory"},
4     {"name": "cpu"}
5   ],
6   "hosts": [
7     {"name": "Host_", "range": "[1-3]"}
8   ]
9 }
```


3.8.1. Ejemplos de peticiones HTTP

3.8.1.1. Obtener una o mas propiedades de una unica instancia - POST /virtshell/api/v1/properties

```
1 curl -sv -X GET \  
2   -H 'accept: application/json' \  
3   -H "Content-Type: text/plain" \  
4   -H 'X-VirtShell-Authorization: UserId:Signature' \  
5   -d '{ "properties": [{"name": "memory"}, {"name": "cpu"  
6       },  
7       "hosts": [{"name": "WebServer"}]}' \  
   'http://localhost:8080/api/virtshell/v1/properties'
```

Respuesta:

```
1 HTTP/1.1 202 OK  
2 Content-Type: application/json  
3 {  
4   "id": "kj5436c0-dc94-13tg-82ce-9992b5d5c51b",  
5   "name": "Database001",  
6   "memory": 1024  
7 }
```

3.8.1.2. Obtener una o mas propiedades de una o mas instancias por tag - POST /virtshell/api/v1/properties

```
1 curl -sv -X GET \  
2   -H 'accept: application/json' \  
3   -H "Content-Type: text/plain" \  
4   -H 'X-VirtShell-Authorization: UserId:Signature' \  
   'http://localhost:8080/api/virtshell/v1/properties'
```

```
5 -d '{ "properties": [{"name": "memory"}, {"name": "cpu"  
    }],  
6     "tag": [{"name": "web"}]}' \  
7 'http://localhost:8080/api/virtshell/v1/properties'
```

Respuesta:

```
1 HTTP/1.1 202 OK  
2 Content-Type: application/json  
3 {  
4   properties: [  
5     {  
6       "id": "kj5436c0-dc94-13tg-82ce-9992b5d5c51b",  
7       "name": "WebServerPhp001",  
8       "memory": 1024,  
9       "cpu": 2  
10    },  
11    {  
12      "id": "591b3828-7aaf-4833-a94c-ad0df44d59a4",  
13      "name": "WebServerPhp002",  
14      "memory": 1024,  
15      "cpu": 1  
16    }  
17  ]  
18 }
```

3.8.1.3. Obtener una o mas propiedades de una o mas instancias usando como prefijo un rango - POST /virtshell/api/v1/properties

```
1 curl -sv -X GET \  

```

```
2 -H 'accept: application/json' \  
3 -H "Content-Type: text/plain" \  
4 -H 'X-VirtShell-Authorization: UserId:Signature' \  
5 -d '{ "properties": [{"name": "memory"}, {"name": "cpu"  
    }],  
6     {"name": "Database00", "range": "[1-3]"]}]' \  
7 'http://localhost:8080/api/virtshell/v1/properties'
```

Respuesta:

```
1 HTTP/1.1 202 OK  
2 Content-Type: application/json  
3 {  
4   properties: [  
5     {  
6       "id": "kj5436c0-dc94-13tg-82ce-9992b5d5c51b",  
7       "name": "Database001",  
8       "memory": 4024,  
9       "cpu": 2  
10    },  
11    {  
12      "id": "591b3828-7aaf-4833-a94c-ad0df44d59a4",  
13      "name": "Database002",  
14      "memory": 4024,  
15      "cpu": 1  
16    },  
17    {  
18      "id": "f7c81039-5c88-423b-8b0d-c124483d586b",  
19      "name": "Database003",  
20      "memory": 4024,  
21      "cpu": 3
```

```
22     }  
23   ]  
24 }
```

CAPÍTULO 4

Recomendaciones

- Diseñar e implementar políticas de seguridad para la administración de los archivos.
- Implementar una interfaz web que permita administrar los ambientes y máquinas virtuales.
- Implementar los agentes de monitoreo de recursos.
- Implementar algún mecanismo de seguridad que permita revisar las tramas que llegan y salen de las máquinas virtuales y los hosts.
- Realizar un plan de pruebas funcionales para los ambientes que se aprovisionan.

Bibliografía

Bibliografía

- [1] EECS Department, Northwestern University, <http://www.aqualab.cs.northwestern.edu/projects/149-nixes-tool-set>, 2013.
- [2] SmartFrog, <http://smartfrog.org/display/sf/SmartFrog+Home>, 2009.
- [3] SmartFrog, <https://radia.accelerite.com/solutions>, 2015.
- [4] Chef, <https://docs.chef.io/>, 2015.
- [5] Puppet, <http://projects.puppetlabs.com/projects/puppet>, 2015.
- [6] Cfengine, <https://www.gnu.org/software/cfengine/>, 2001.
- [7] Bdfg2, <http://bcfg2.org/>, 2015.