

Infraestructura tecnológica virtual con automatización y orquestación.


Arese, Juan Pablo - Diers, Werner Christian

Facultad de Ciencias Exactas, Físicas y Naturales - UNC

Marzo 2017



Organización de la Presentación


- ▶ Introducción
 - ▶ Objetivos
 - ▶ Arquitectura
 - ▶ Desarrollo del sistema
 - ▶ Servidor web
 - ▶ Servidor de virtualización
 - ▶ Servidor de aprovisionamiento
 - ▶ Servidor de orquestación
 - ▶ Conclusión
 - ▶ Trabajos futuros
 - ▶ Demostración
- 

Introducción



Introducción

Una infraestructura moderna implica:


- ▶ Costos
 - ▶ Rendimiento computacional
 - ▶ Aplicación de políticas
 - ▶ Configuraciones establecidas por cada entidad
 - ▶ Estandarización de los recursos y parámetros utilizados
 - ▶ Agilidad
- 

Introducción

¿Qué es **virtualización**?

Software ejecutándose, de forma **concurrente** y **aislada** de **otros procesos** en el mismo sistema.


Es la manera más eficaz de **reducir los costos** y **aumentar la agilidad** de cualquier organización.



Introducción

¿Qué es aprovisionamiento?

Aprovisionar es proveer o hacer que algo esté disponible. En el **contexto de esta presentación**, aprovisionar es el **conjunto de acciones** requeridas para preparar una máquina virtual para su uso básico.

- ▶ Disco
 - ▶ Memoria RAM
 - ▶ CPU
 - ▶ Sistema operativo
 - ▶ Servicios
 - ▶ Configuración
- 

Introducción

¿Qué es orquestación?

- ▶ Automatizar procesos y flujos de trabajo.
- ▶ Consistencia de la infraestructura.
- ▶ Infraestructura como código.
- ▶ Integrar servicios rápidamente.



Objetivos



Objetivos

Integrar diferentes herramientas con el fin de implementar técnicas de **orquestación**, virtualización, instalación y **configuración automática** para facilitar la **gestión de servidores** virtuales y sus servicios asociados.



Arquitectura




Arquitectura

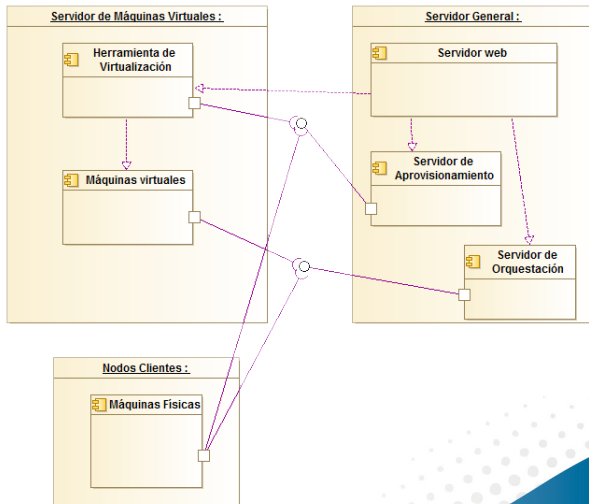
La arquitectura implementada es cliente - servidor.

En esta arquitectura, múltiples clientes **realizan peticiones** a los servidores, los cuales les **dan respuesta**.

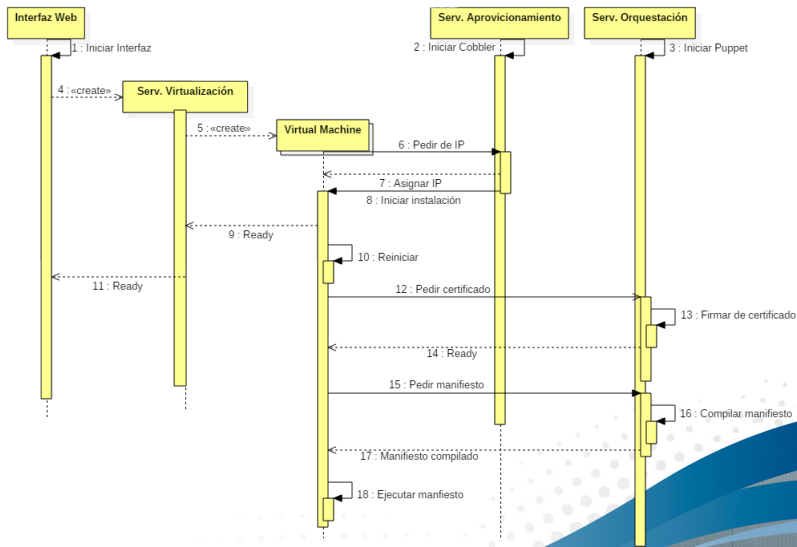
El sistema cuenta con 4 servidores principales, un servidor de máquinas virtuales, uno de aprovisionamiento, otro de orquestación y un servidor web.



Arquitectura



Arquitectura



Desarrollo



Servidor Web



Desarrollo - Servidor Web

Se utilizó la herramienta Python Bottle para la realización del servidor web, dado que es un WSGI (Web Server Gateway Interface) rápido, sencillo y ligero.

Los **GET** y **POST** son *decorators* que enlazan una pieza de código con una URL.



Desarrollo - Servidor Web

```
#Pagina que recibe los parametros para crear una VM con disco y memoria a eleccion
@get('/virtual_machine_parametrizada')
def creaVM_parametrizado():
    peticionhtml = open("/home/webs/Python/HTMLs/virtual_machine_parametrizada.html","r", o_
    return peticionhtml

#Toma los parametros ingresados para crear la VMs parametrizada
@post('/virtual_machine_parametrizada')
def do_creaVM_parametrizado():
    perfil = request.forms.get('boton1')
    ram = request.forms.get('ram')
    disco = request.forms.get('disco')
    #reviso que los parametros sean adecuados
    if str(perfil)=="None":
        return "'Seleccione un perfil\n'"
    if CreaVm_parametrizada(perfil,ram,disco)=="error":
        return"'Sólo se admiten valores numéricos en los parámetros RAM y disco\n'"
    return estadosVM()
```

```
threads = []
t1 = threading.Thread(target=CreaVm, args=(ncentos,"centos"))
t2 = threading.Thread(target=CreaVm, args=(nubuntu,"ubuntugui"))
t3 = threading.Thread(target=CreaVm, args=(nwindows, "windows"))
threads.append(t1)
threads.append(t2)
threads.append(t3)
```

Desarrollo - Servidor Web

← → ↻ ⓘ 192.168.122.1:8888/estados ☆ ⋮

Crear máquinas virtuales optimizadas.

Crear máquina virtual con parámetros.

Editar configuraciones de las máquinas virtuales.

Editar política de una máquina virtual.

Nombre de la VM	Estado	Acción	
ubuntugui1510	ejecutando	<input type="checkbox"/> Encender	<input type="checkbox"/> Apagar
centos366	apagado	<input type="checkbox"/> Encender	<input type="checkbox"/> Apagar
windows1652	apagado	<input type="checkbox"/> Encender	<input type="checkbox"/> Apagar

Ejecutar acción

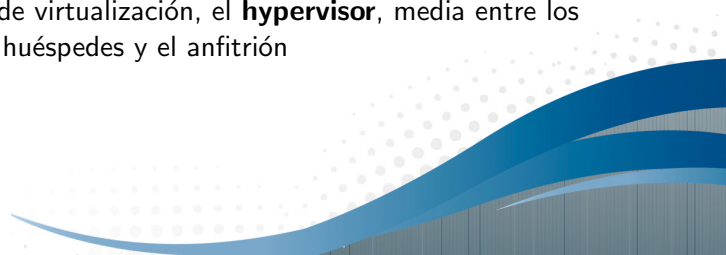
Virtualización



Desarrollo - Servidor de virtualización

Existen diferentes tipos de virtualización. Nosotros aplicamos **virtualización completa**.

La herramienta utilizada fue KVM/Qemu.

- ▶ El sistema operativo huésped **desconoce** que está en un entorno virtual
 - ▶ El hardware se encuentra virtualizado por el sistema operativo anfitrión
 - ▶ La capa de virtualización, el **hypervisor**, media entre los sistemas huéspedes y el anfitrión
- 

Desarrollo - Servidor de virtualización

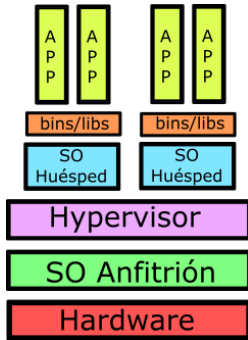


Figure 1 : Virtualización completa

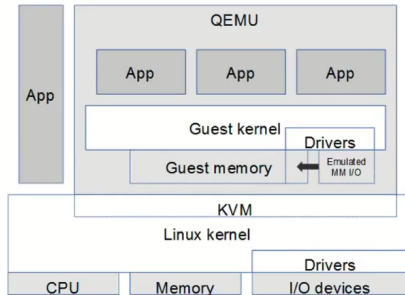


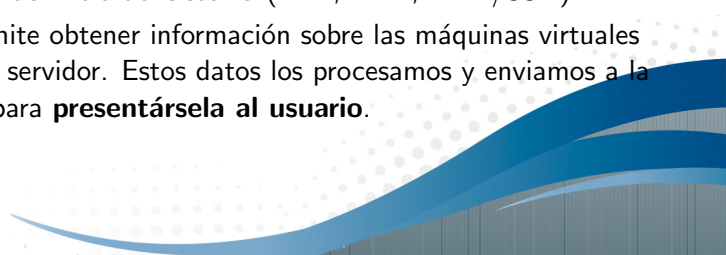
Figure 2 : Arquitectura KVM/Qemu

Desarrollo - Servidor de virtualización

Virsh es la librería que controla al servidor de virtualización. Su funcionalidad es administrar la creación de las máquinas virtuales, personalizando diferentes aspectos como:

- ▶ Nombre
- ▶ Estado
- ▶ Aceleración por hardware
- ▶ MAC Address
- ▶ Prioridad de inicio del sistema (PXE, DVD, HDD/SSD)

También permite obtener información sobre las máquinas virtuales alojadas en el servidor. Estos datos los procesamos y enviamos a la interfaz web para **presentársela al usuario**.



Aprovisionamiento




Desarrollo - Servidor de aprovisionamiento

El centro del servidor de aprovisionamiento es Cobbler. Además de este, encontramos servidores PXE, DHCP, TFTP, SAMBA y DNS.

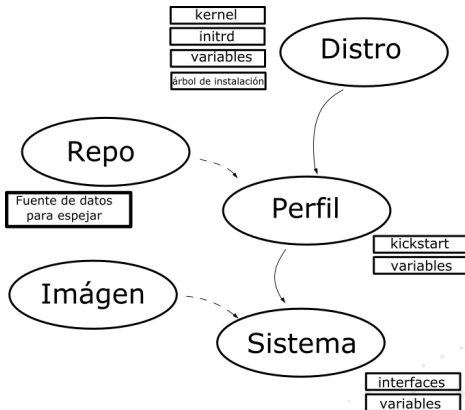
El servidor Cobbler se **basa en objetos** para definir la instalación y configuración deseada en cada caso.

Los objetos se ordenan en una **jerarquía vertical**, donde el objeto inferior, contiene a los superiores.

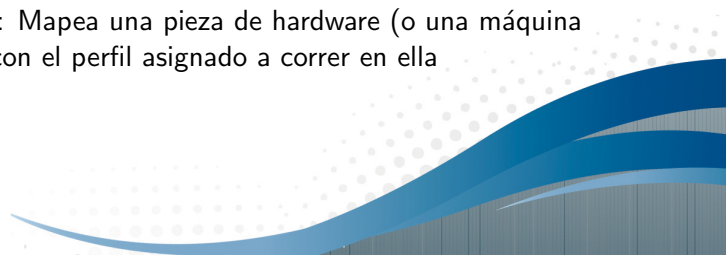


Desarrollo - Servidor de aprovisionamiento

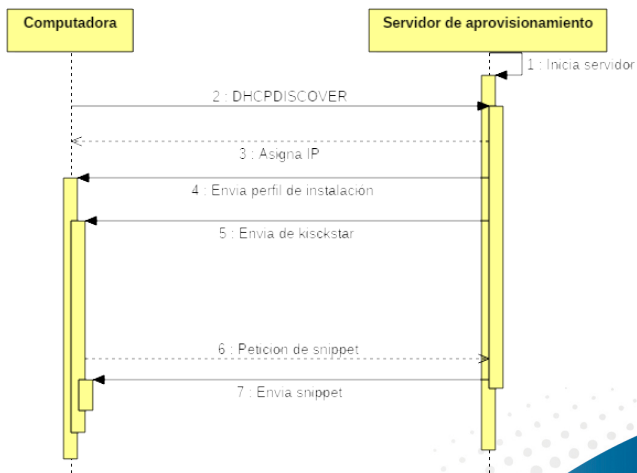
Modelado de Cobbler



Desarrollo - Servidor de aprovisionamiento

- ▶ **Distro:** Distribución que se desea instalar
 - ▶ **Repo:** Repositorio, sitio centralizado donde se almacena y mantiene información digital
 - ▶ **Perfil:** Asocia una distribución a opciones especializadas adicionales, como puede ser un archivo de configuración
 - ▶ **Imágen:** Copia del estado de un sistema computacional, guardado en un archivo o disco
 - ▶ **Sistema:** Mapea una pieza de hardware (o una máquina virtual) con el perfil asignado a correr en ella
- 

Desarrollo - Servidor de aprovisionamiento




Orquestación



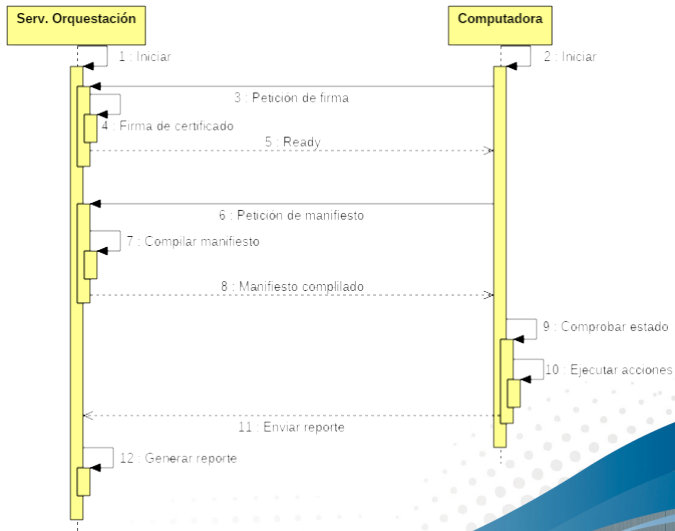
Desarrollo - Servidor de orquestación

El servidor usado para orquestar fue Puppet.

- ▶ Tiene soporte para la administración de **múltiples plataformas**.
 - ▶ Los recursos del sistema y sus estados se configuran utilizando un **lenguaje declarativo propio**.
 - ▶ Estas configuraciones se denominan **manifiestos**.
 - ▶ Las comunicaciones se realizan bajo **HTTPS**.
- 

Desarrollo - Servidor de orquestación

Ciclo de orquestación



Desarrollo - Servidor de orquestación

Estructura de los módulos de Puppet

```
[root@puppet Modulos Puppet]# tree -C
.
├── eclipse
│   ├── files
│   │   ├── eclipse.desktop.centos
│   │   └── eclipse.desktop.ubuntu
│   └── manifests
│       └── init.pp
├── httpd
│   └── manifests
│       ├── httpd.pp
│       └── init.pp
├── idle
│   └── manifests
│       └── init.pp
├── MySQL
│   └── manifests
│       ├── init.pp
│       └── mysql.pp
├── nfs
│   └── manifests
│       ├── client.pp
│       ├── init.pp
│       └── server.pp
```

Desarrollo - Servidor de orquestación

Ejemplo lenguaje declarativo de Puppet

```
class usuarios(  
  $usuario = "alumno"  
)  
{  
  if $osfamily == "Windows"  
  {  
    user { 'creo_usuario':  
      name => $usuario,  
      ensure => present,  
      groups => ['Usuarios'],  
      managehome => true,  
      password => 'alumno',  
    }  
  }  
  else  
  {  
    user { 'creo_usuario':  
      name => $usuario,  
      ensure => 'present',  
      password => '$!$t059HC0X$N/J0Km9dJQEGmwSnjDrW0/',  
      password_max_age => '99999',  
      password_min_age => '0',  
      allowdupe => 'false',  
      expiry => 'absent',  
      home => '/home/${usuario}'  
    }  
  
    group { 'grupo_usuario':  
      name => $usuario,  
      ensure => 'present',  
      allowdupe => 'false',  
      members => $usuario,  
    }  
  }  
}
```


Desarrollo - Servidor de orquestación

Ejemplo lenguaje declarativo de Puppet

```
class mysql(  
  $password = "4/Ulz4PFF0wu21EqxrXUrbvYFZNfc0r/4vQ"  
)  
{  
  package { 'paquete_mysql':  
    ensure => installed,  
    name   => 'mysql',  
  }  
  
  package { 'paquete_mysql-server':  
    ensure => installed,  
    name   => 'mysql-community-server',  
  }  
  
  service {'servicio_mysql':  
    ensure => running,  
    name   => 'mysqld',  
    require => Package['paquete_mysql-server'],  
    require => Package['paquete_mysql'],  
  }  
  
  exec{'set_clave_root_mysql' :  
    command => "mysqladmin -u root password ${$password}",  
    cwd     => '/',  
    require => Service['servicio_mysql'],  
  }  
}
```

Conclusiones

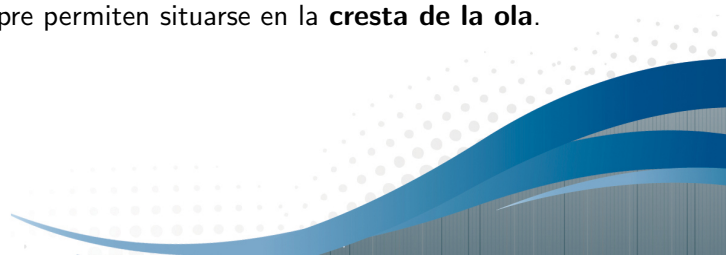


Conclusiones

El sistema obtenido, es un **sistema modularizado**. Cada tarea es realizada por un servidor de forma independiente al resto.

Se implementó un diseño que permite **escalabilidad y mejoras a futuro**.

Si bien se encontró una solución basada en **herramientas libres**, estas no siempre permiten situarse en la **cresta de la ola**.



Trabajos Futuros



Trabajos Futuros

- ▶ Protección:
 - ▶ Modificar el sistema para que funcione con **firewall** y **SELinux**.
 - ▶ Incluir **autenticación** por usuario en la interfaz web.
 - ▶ Incluir un **log de cambios** al sistema que permita saber quién y qué cambio realizó.
- ▶ Migración: Poder realizar la **migración en vivo** de máquinas virtuales.



Video demostración



Preguntas



Muchas Gracias!

