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
  - Interfaz web
  - Herramienta de virtualización
  - Herramienta de aprovisionamiento
  - Herramienta 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
- Concurrencia
- Aislamiento

Introducción

¿Qué es aprovisionamiento?

- Proveer o hacer que algo esté disponible
- Conjunto de acciones para preparar una máquina virtual
  - Disco
  - Memoria RAM
  - CPU
  - Sistema operativo
  - Servicios
  - Configuración

Introducción

¿Qué es orquestación?

- Automatizar procesos y flujos de trabajo
- Consistencia
- Infraestructura como código
- Integrar servicios

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.

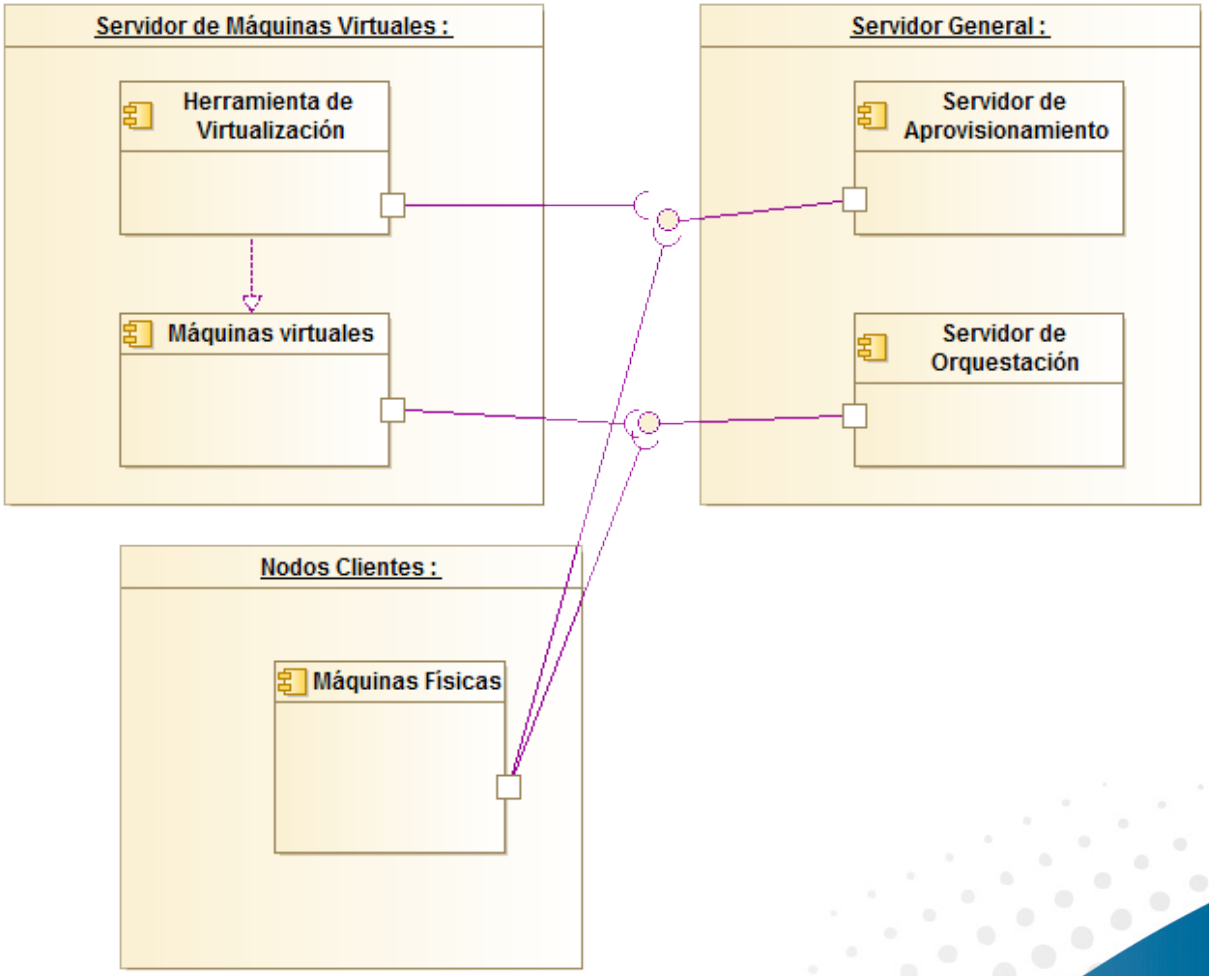
Arquitectura

Arquitectura

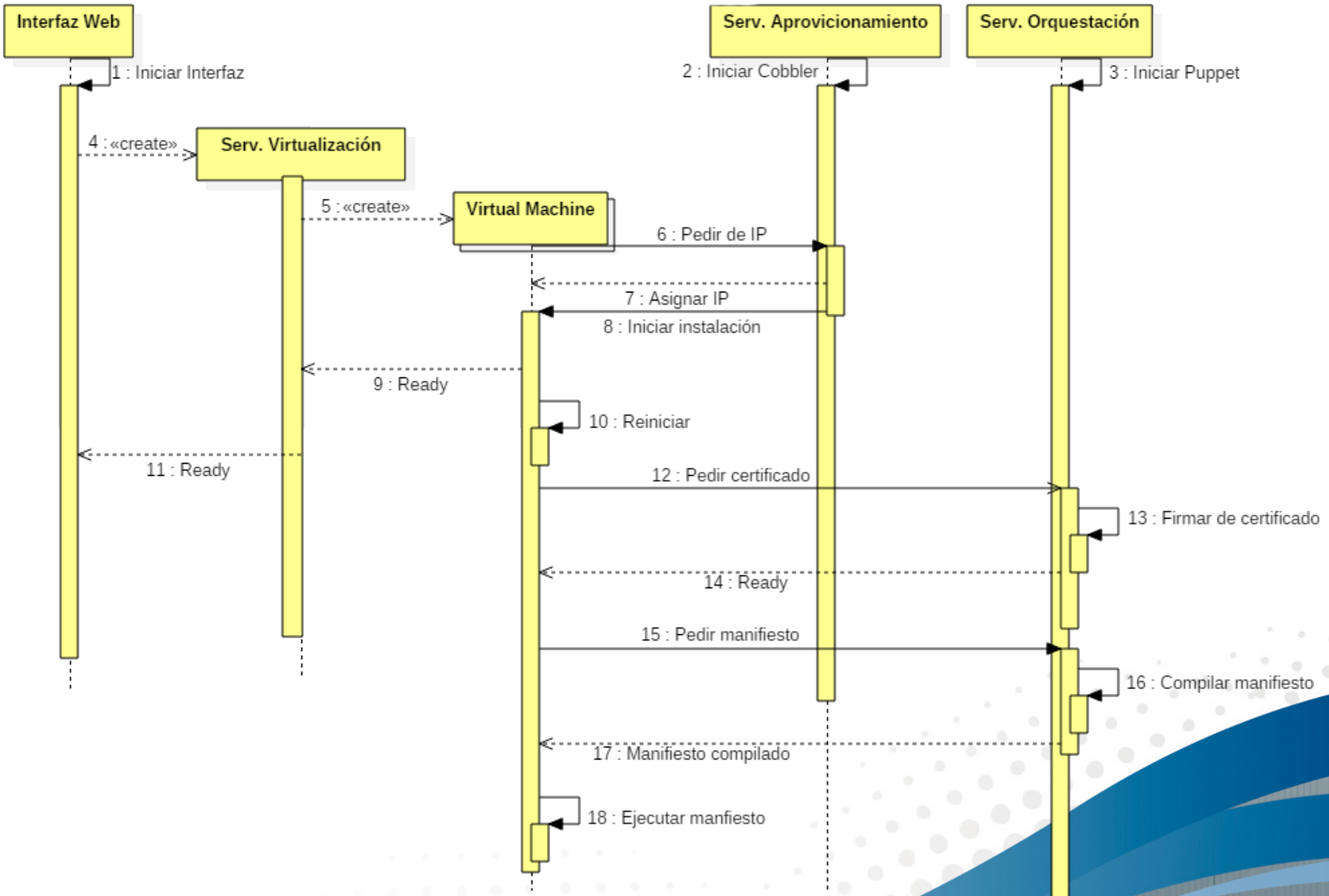
La arquitectura implementada es la de cliente - servidor.
Las tareas del servidor son las siguientes:

- Crear las máquinas virtuales
- Asignar direcciones IP por medio del protocolo DHCP
- Proveer a la máquina con el sistema operativo deseado y los parámetros de configuración establecidos
- Orquestar las políticas definidas para una máquina o un conjunto de máquinas

Arquitectura



Arquitectura



Desarrollo

Interfaz Web

Desarrollo - Interfaz Web

La herramienta utilizada para crear la interfaz web fue Python Bottle.

- Es un WSGI (Web Server Gateway Interface) rápido, sencillo y ligero
- Distribuído como un módulo único
- Su única dependencia es la Librería Estándar de Python
- Puede ejecutarse como un servidor web autónomo
- Plugins para bases de datos populares

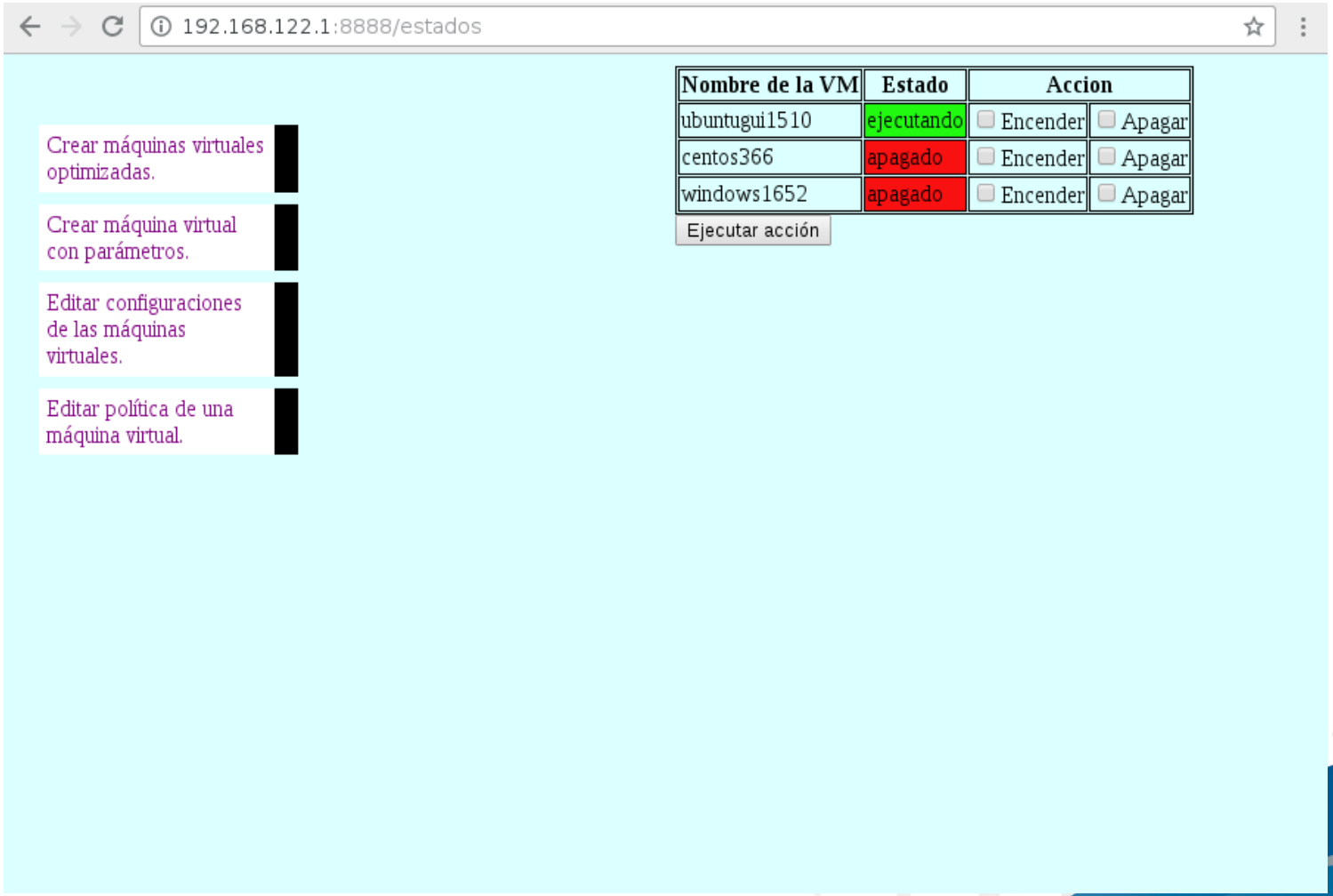
Desarrollo - Interfaz Web

```python
#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", 0)
    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()
```

```python
    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 - Interfaz Web



Virtualización

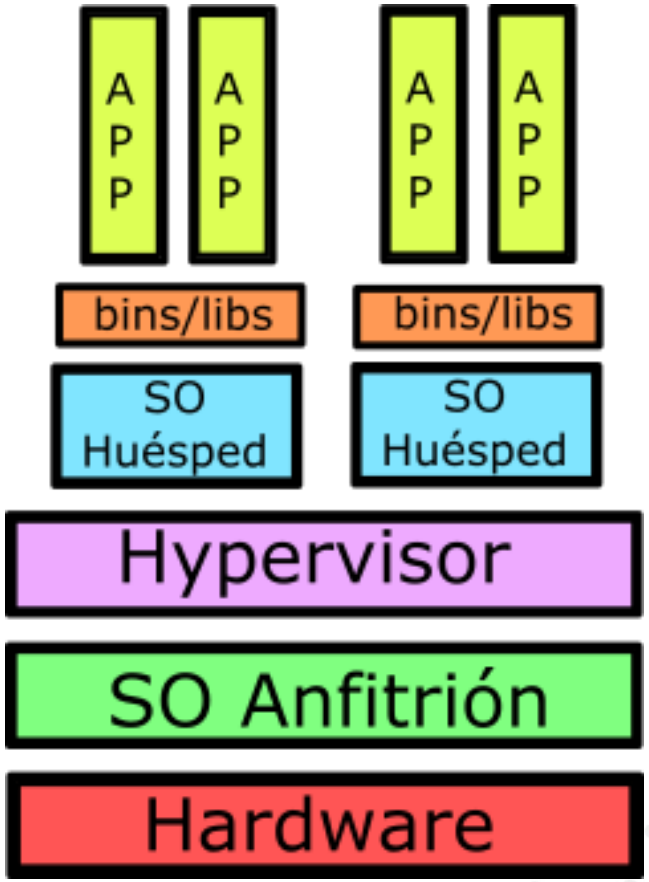
Desarrollo - Virtualización

La herramienta utilizada para virtualizar fue KVM/Qemu.
KVM utiliza virtualización completa:

- 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

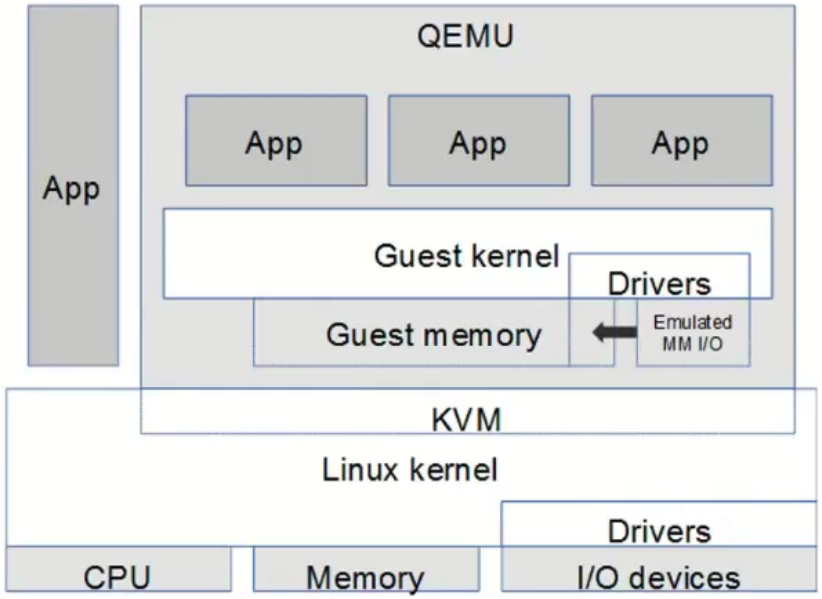
Desarrollo - Virtualización

Esquema de virtualización completa



Desarrollo - Virtualización

Arquitectura de KVM



Desarrollo - Virtualización

Ejemplo de creación de una máquina virtual

```
# virt-install \
      --connect qemu:///system \
      --name=centos-vm \
      --disk path=/var/lib/libvirt/images/centos-vm.qcow2,size=25 \
      --graphics spice \
      --vcpus=2 --ram=3072 \
      --network network=puppet, mac="52:54:00:d5:a1:76" --pxe \
      --os-type=linux \
      --os-variant=centos7
```

Aprovisionamiento

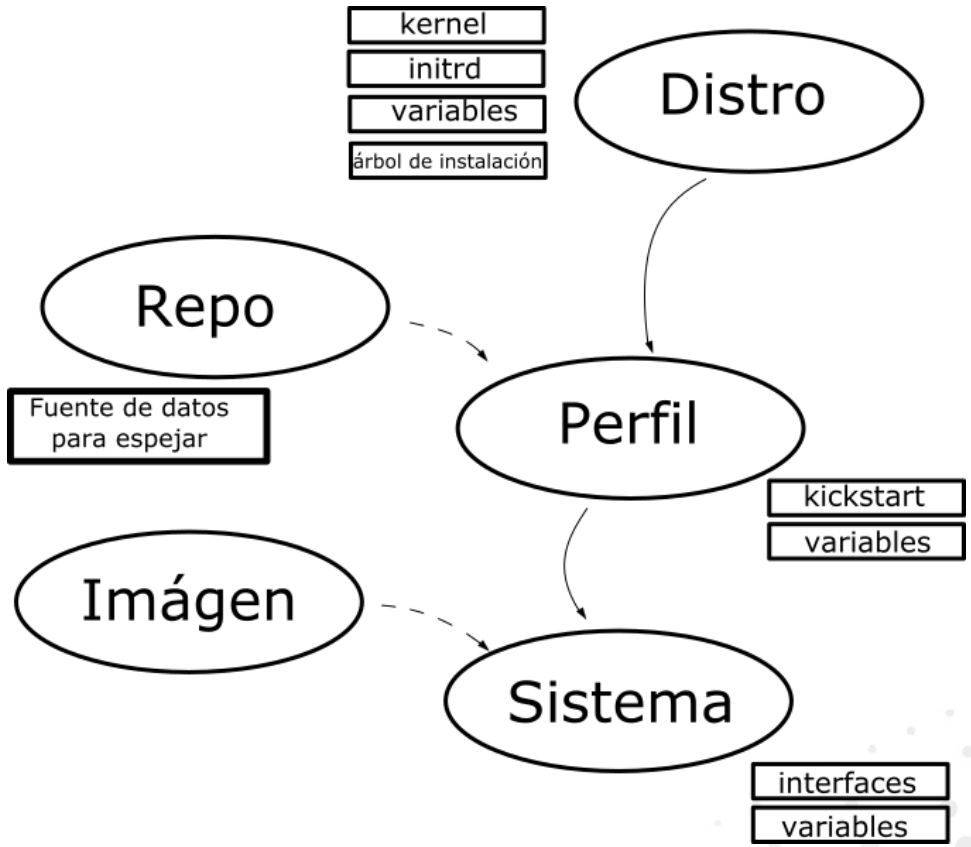
Desarrollo - Aprovisionamiento

La herramienta utilizada para el aprovisionamiento fue Cobbler. Utiliza una arquitectura cliente - servidor.

- El servidor debe ejecutarse en un sistema basado en Unix
- Centraliza y simplifica el control de servicios incluyendo PXE, DHCP, TFTP y DNS con propósito de realizar instalaciones basadas en red de sistemas operativos
- Cobbler utiliza objetos para definir la configuración de aprovisionamiento:

Desarrollo - Aprovisionamiento

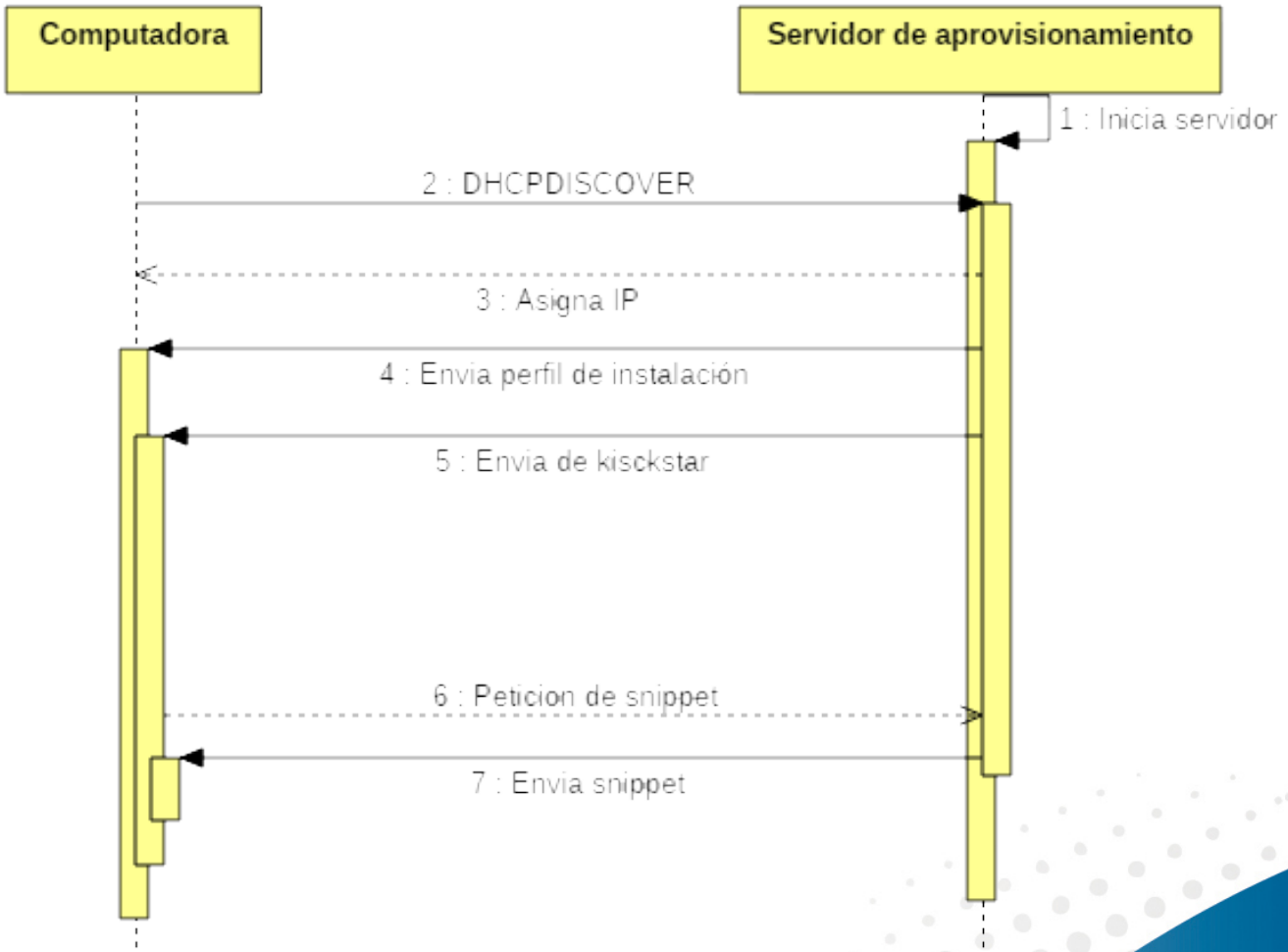
Modelado de Cobbler



Desarrollo - 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 - Aprovisionamiento



Orquestación

Desarrollo - Orquestación

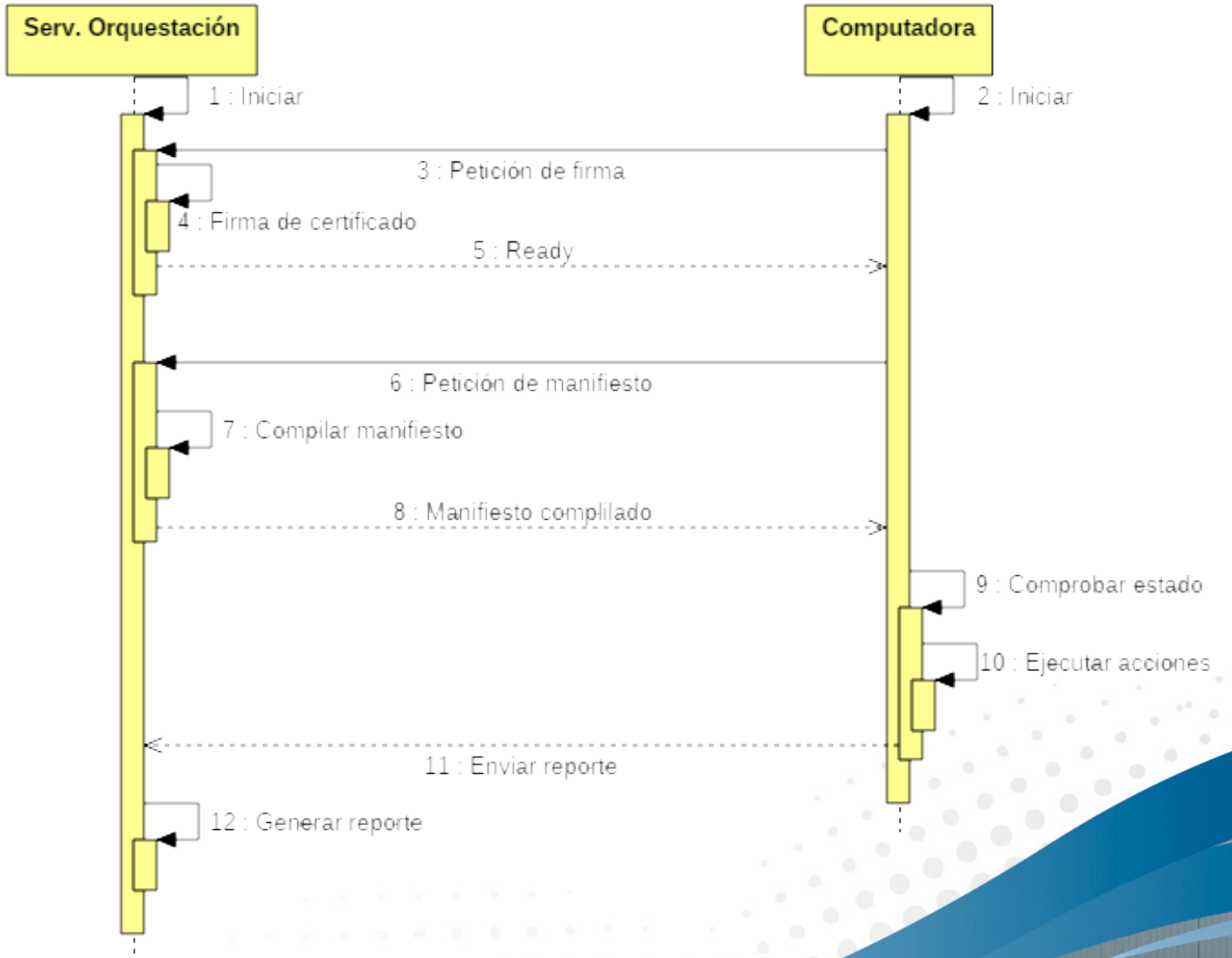
La herramienta utilizada para orquestar fue Puppet.

Utiliza una arquitectura cliente - servidor.

- El servidor (nodo maestro) debe ejecutarse en un sistema basado en Unix
- Los clientes (agentes) soportan múltiples plataformas
- Posee su propio DSL (Domain Specific Language)
- El usuario describe los recursos del sistema y sus estados utilizando un lenguaje declarativo
- El nodo maestro 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

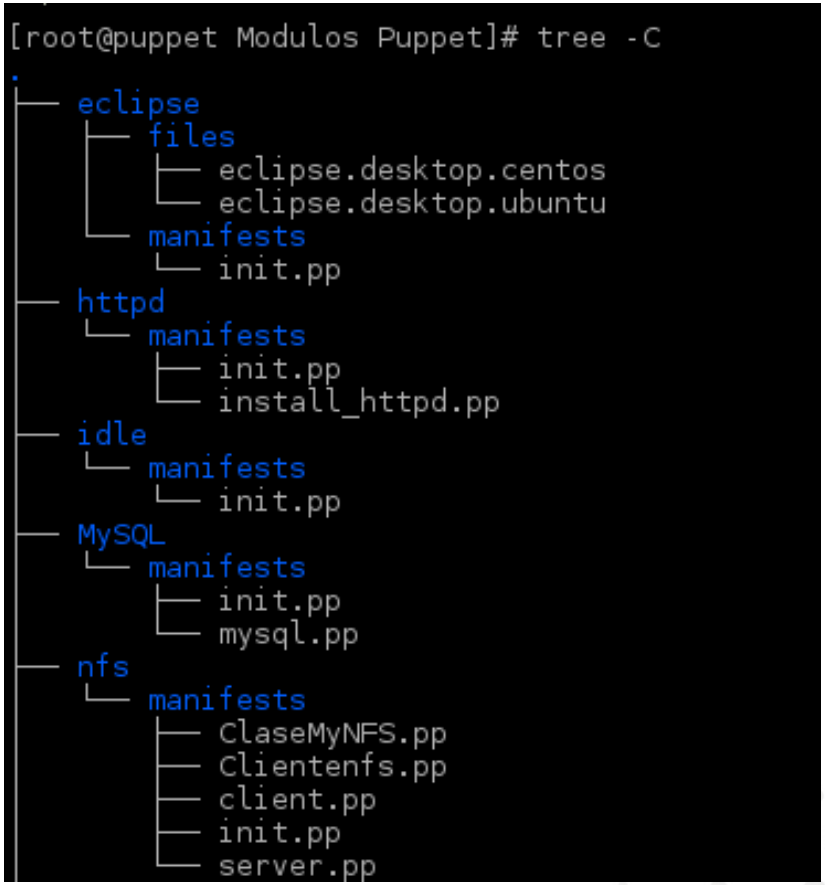
Desarrollo - Orquestación

Ciclo de orquestación



Desarrollo - Orquestación

Estructura de los módulos de Puppet



Desarrollo - Orquestación

Ejemplo lenguaje declarativo de Puppet

```
class myqsl(
    $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'],
  }
```

Desarrollo - 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   => '$1$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,
            require => User['creo_usuario'],
        }
```

Conclusiones

Conclusiones

- Elección del entorno realizado utilizando factores de decisión ponderados
- Solución modularizada
- Permite la escalabilidad necesaria
- Soluciones de código abierto no siempre permiten estar en la "cresta de la ola"
- El sistema final cumple con los requerimientos

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 el traslado de hosts virtuales entre las diferentes máquinas físicas

Video demostración

Video demostración

- Creación de las máquinas virtuales
- Aprovisionamiento de las máquinas con el sistema operativo deseado
- Orquestar las políticas definidas para una máquina o un conjunto de máquinas

Preguntas

Muchas Gracias!