

Administración centralizada con Puppet



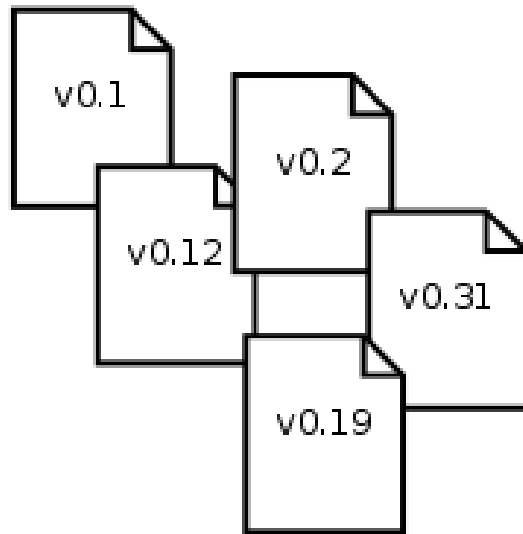
Puppet

Lucas Di Pentima
<lucas@di-pentima.com.ar>
LUGLi – Santa Fe

Temas a tratar

- Problemática a solucionar
 - Conceptos generales de Puppet
 - Ejemplos de aplicación
 - Temas para seguir estudiando
-
-

Problemática: Scripts personalizados



`/usr/local/sbin/chequear-servicio.sh`

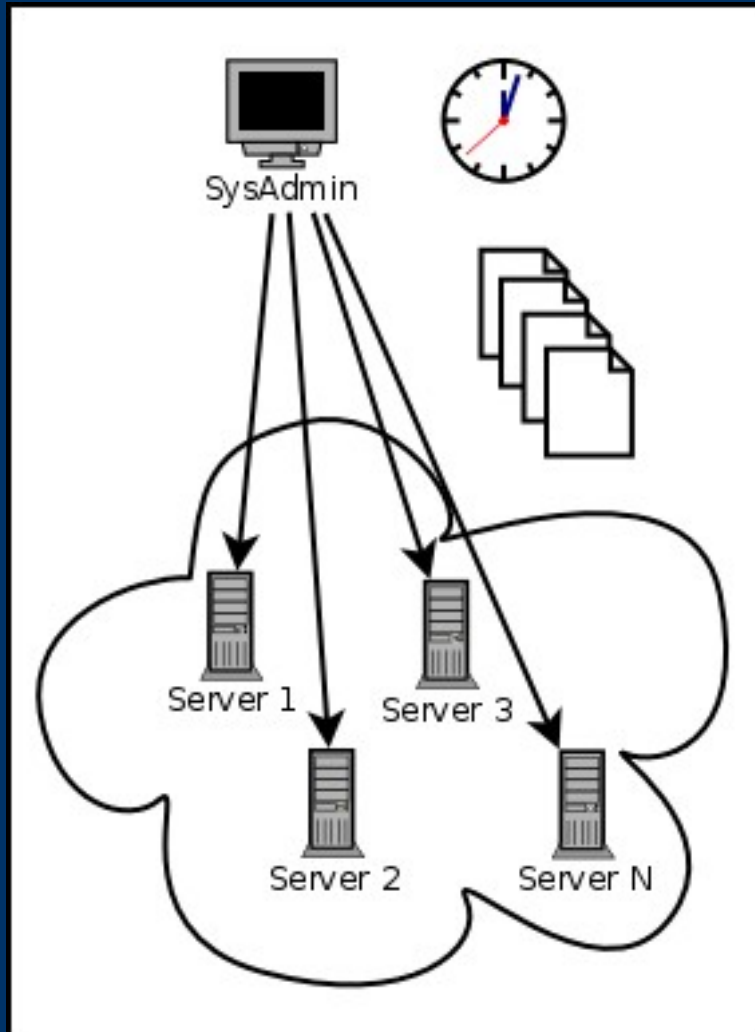
`/usr/local/bin/chequear_servicio`

`/opt/bin/check_service.sh`

`/des/con/trol/total.sh`

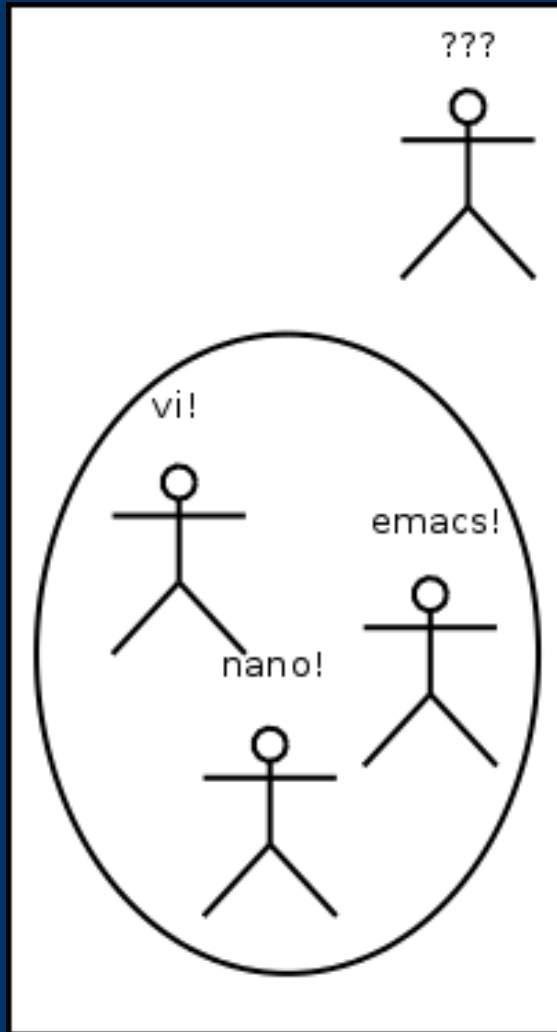
- Distintas versiones de lo mismo.
- Poca portabilidad entre instalaciones.

Problemática: Gestión manual del software



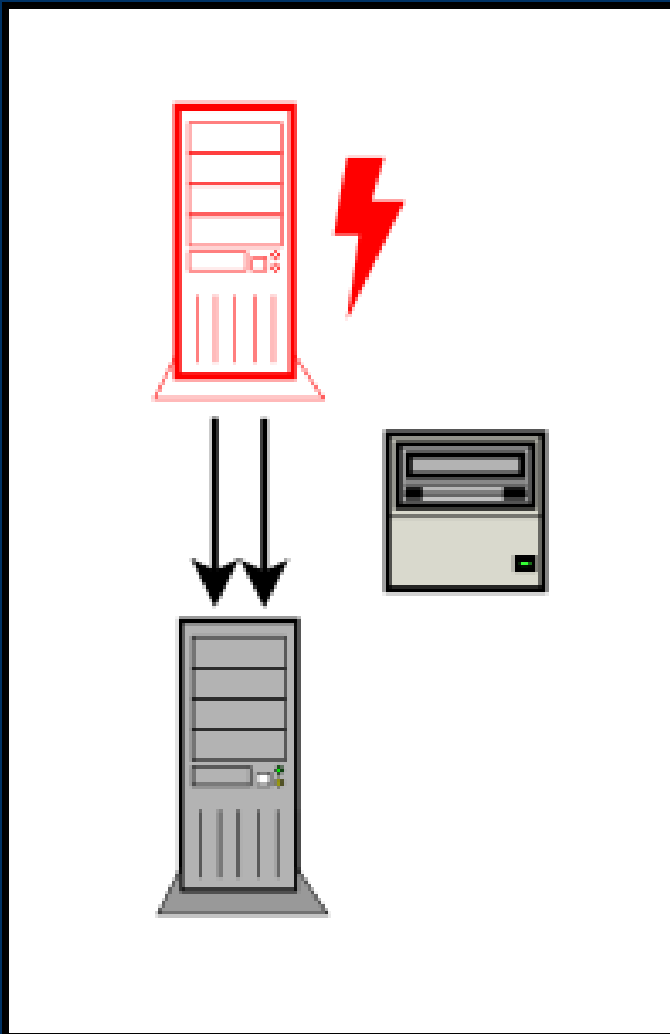
- Esfuerzo directamente proporcional al número de servidores.
- Tareas repetitivas que tienden a provocar errores.

Problemática: Configuraciones ad hoc



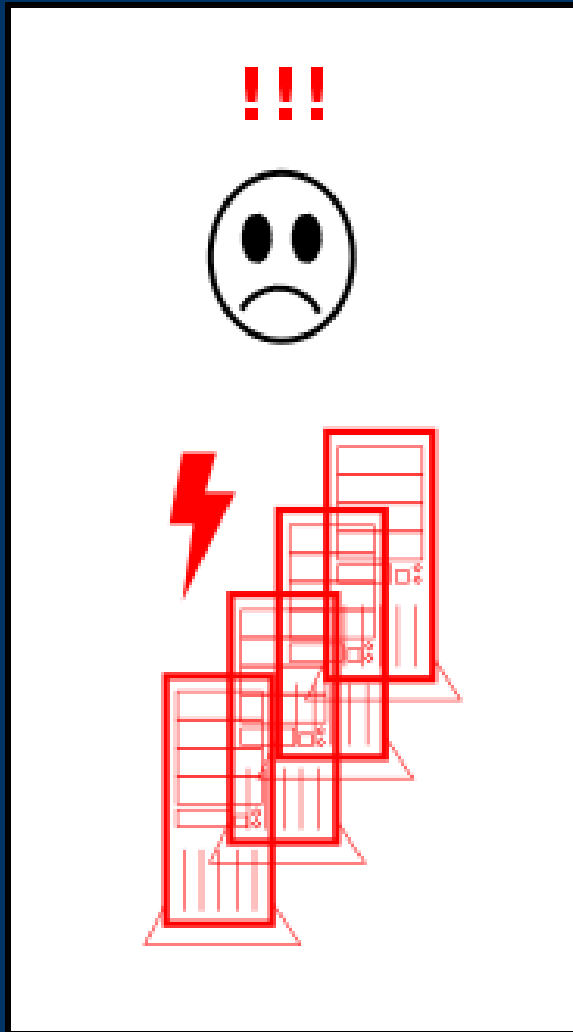
- Distintos criterios con el pasar del tiempo.
- Un estilo diferente por cada persona en el equipo.
- Barrera de entrada alta para los nuevos.

Problemática: Restauración “bare metal” de un servidor



- ¿Tenemos backup reciente del sistema operativo?
- Tiempos de recuperación y cortes de servicio excesivamente altos.

Problemática: Restauración “bare metal” de N servidores



- ¡OUCH!
- Llevar backups de sistemas de N servidores es muy tedioso.

Problemática: Documentación

- ¿Qué servicios están activos?
- ¿Donde están instalados esos servicios?
- ¿Que tan actualizada está la documentación?



Problemática: ¡Escalabilidad!

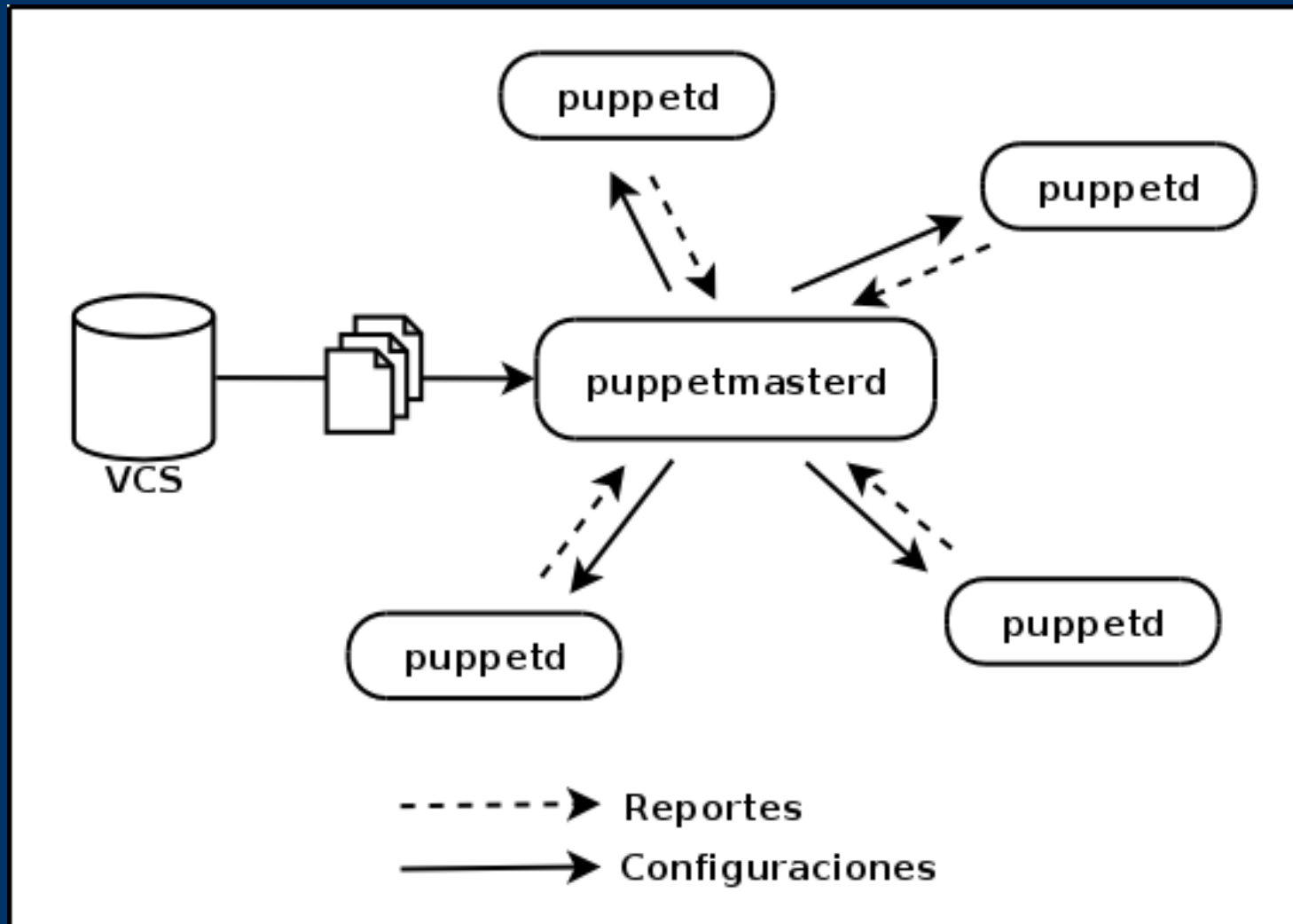
A mayor cantidad de servidores, mayor es la carga de su mantenimiento.
¡Esto no es negocio!



Puppet: Componentes

- Lenguaje declarativo.
 - Servidor.
 - Cliente.
 - Biblioteca de abstracción de recursos.
-
-

Puppet: Esquema de funcionamiento



Puppet: Abstracción de recursos

- Independencia de la implementación.
 - “providers”
 - Pensamiento en alto nivel.
 - “package”
 - “user” ...
 - Recursos relacionables entre si.
 - “subscribe” / “notify”
 - “require” / “before”
-
-

Puppet: Lenguaje declarativo

- Se usa para describir el estado de un servidor.
- Idempotencia: se obtiene el mismo resultado sin importar las veces que se aplique la configuración.



Puppet: Recursos (tipos)



- Usuario (user)
- Archivo (file)
- Paquete de software (package)
- Servicio (service)
- Tarea periódica (cron)
- ...

Puppet: Recursos: user

- home
 - uid
 - gid
 - shell
 - comment
-
-

Puppet: Recursos: file

- checksum
 - content
 - ensure (absent/present , file/directory)
 - user, group
 - source
 - mode
-
-

Puppet: Recursos: package

- ensure (absent, purged, present/installed)



Puppet: Recursos: service

- ensure (running/stopped)
- enable (true/false)



Puppet: Recursos: cron

- command
 - user
 - hour
 - minute
 - month
 - weekday
-
-

Puppet: Clases

```
class unix {  
  file {  
    "/etc/passwd":  
      owner => "root",  
      group => "root",  
      mode => 644;  
    "/etc/shadow":  
      owner => "root",  
      group => "root",  
      mode => 440;  
  }  
}
```

- Estructura que engloba recursos individuales
- Utilizable en casos únicos por host (ej: un servicio)
- Soporta jerarquía de herencia

Puppet: Definiciones

```
define hourlycronjob ($script) {
  $path = "/opt/bin/$script"
  file { $path:
    source => puppet://..$script,
    user => "root",
    group => "root",
    mode => 750
  }
  cron { $script:
    command => $path,
    minute => 0
  }
}

hourlycronjob { "checkpuppet":
  script => "chequear-puppet.sh"
}
```

- Como las clases, pero con argumentos.
- Utilizable en casos parametrizables por host (ej: virtualhosts en apache)

Puppet: Nodos

```
import "classes/*"

node default {
  include unix
}

node webserver1 inherits default {
  include apache
}

node webserver2 inherits default {
  include apache
}
```

- Como las clases, pero se aplican a los hosts.
- Se utilizan para asignar clases y definiciones a los clientes.
- Soportan herencia.

Puppet: Otras características

- Selectores
 - If/then
 - Case
 - Funciones
 - Variables / Arrays
 - Facts
-
-

Ejemplos: Configurar vim como editor preferido

```
class vim {  
  
  package { "vim":  
    ensure => present,  
  }  
  
  exec { "update-alternatives -set editor /usr/bin/vim.basic":  
    path => "/bin:/usr/bin",  
    unless => "ls -al /etc/alternatives/editor | grep  
vim.basic",  
  }  
}
```

Ejemplos: Deshabilitar login de root del sshd

```
class norootssh {  
  
    package { "openssh-server":  
        ensure => present  
    }  
  
    service { "ssh":  
        ensure => running  
    }  
  
    exec { "perl -pi -e 's/^PermitRootLogin yes$/PermitRootLogin  
no/' /etc/ssh/sshd_config:  
        path => "/bin:/usr/bin:/sbin:/usr/sbin",  
        unless => "grep 'PermitRootLogin no' /etc/ssh/sshd_config",  
        notify => service[ssh]  
    }  
}
```

Ejemplos: Virtualhosts de Apache

```
define vhost ( $domain, $admin, $docroot, $enable = true ) {
  $vhpath = "/etc/apache/conf.d/$domain.conf"
  file { $vhpath:
    notify => Service[apache],
    require => File[$docroot],
    ensure => $enable ? {
      true => present,
      false => absent
    },
    content => template("vhost.erb"),
    owner => root, group => root, mode => 640,
  }
  file { $docroot:
    ensure => $enable ? {
      true => directory,
      false => absent
    }
  }
}
}
```

Ejemplos: vhost.erb

```
<VirtualHost *>  
  ServerAdmin <%= admin %>  
  DocumentRoot <%= docroot %>  
  ServerName <%= domain %>  
</VirtualHost>
```

¡Esto no termina acá!

- Reportes
- Gráficas
- Extensibilidad
 - Funciones
 - Providers
 - Facts



Enlaces de interés

- <http://www.linux-mag.com/id/4141/>
 - <http://reductivelabs.com/trac/puppet/wiki/DocumentationStart>
 - <http://people.redhat.com/dlutter/puppet-app.html>
-
-

Administración centralizada con Puppet



Puppet

Lucas Di Pentima
<lucas@di-pentima.com.ar>
LUGLi – Santa Fe

¡GRACIAS!