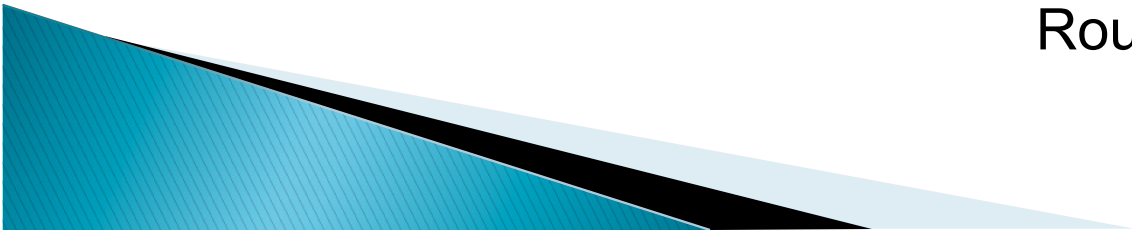




Gestión de Configuraciones con *Puppet*

Carlos Armas

Roundtrip Networks Corp.

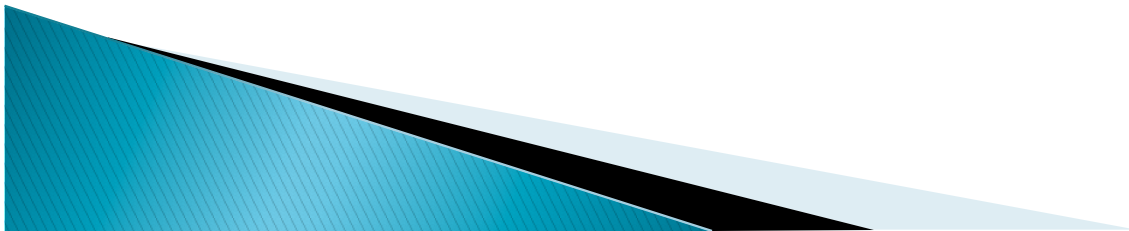
roundtrip networks

Por qué?

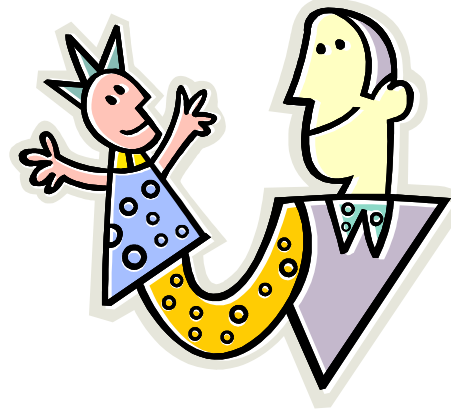
- ▶ Instalar y proveer sistemas rápido!
- ▶ Eliminar tareas repetitivas



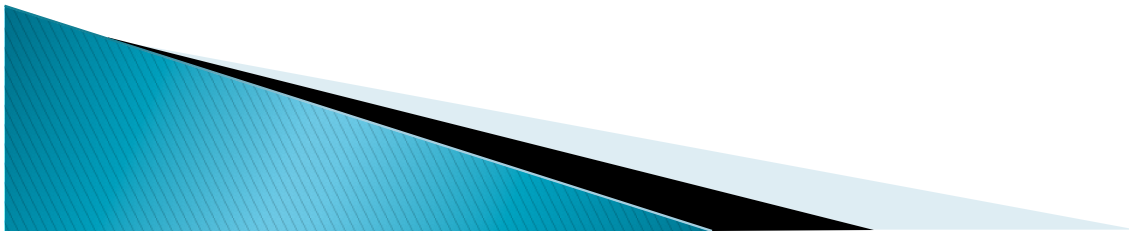
- ▶ La configuración de los sistemas tiende a la divergencia
- ▶ La complejidad de administración aumenta exponencialmente a medida que crece el número de sistemas a administrar



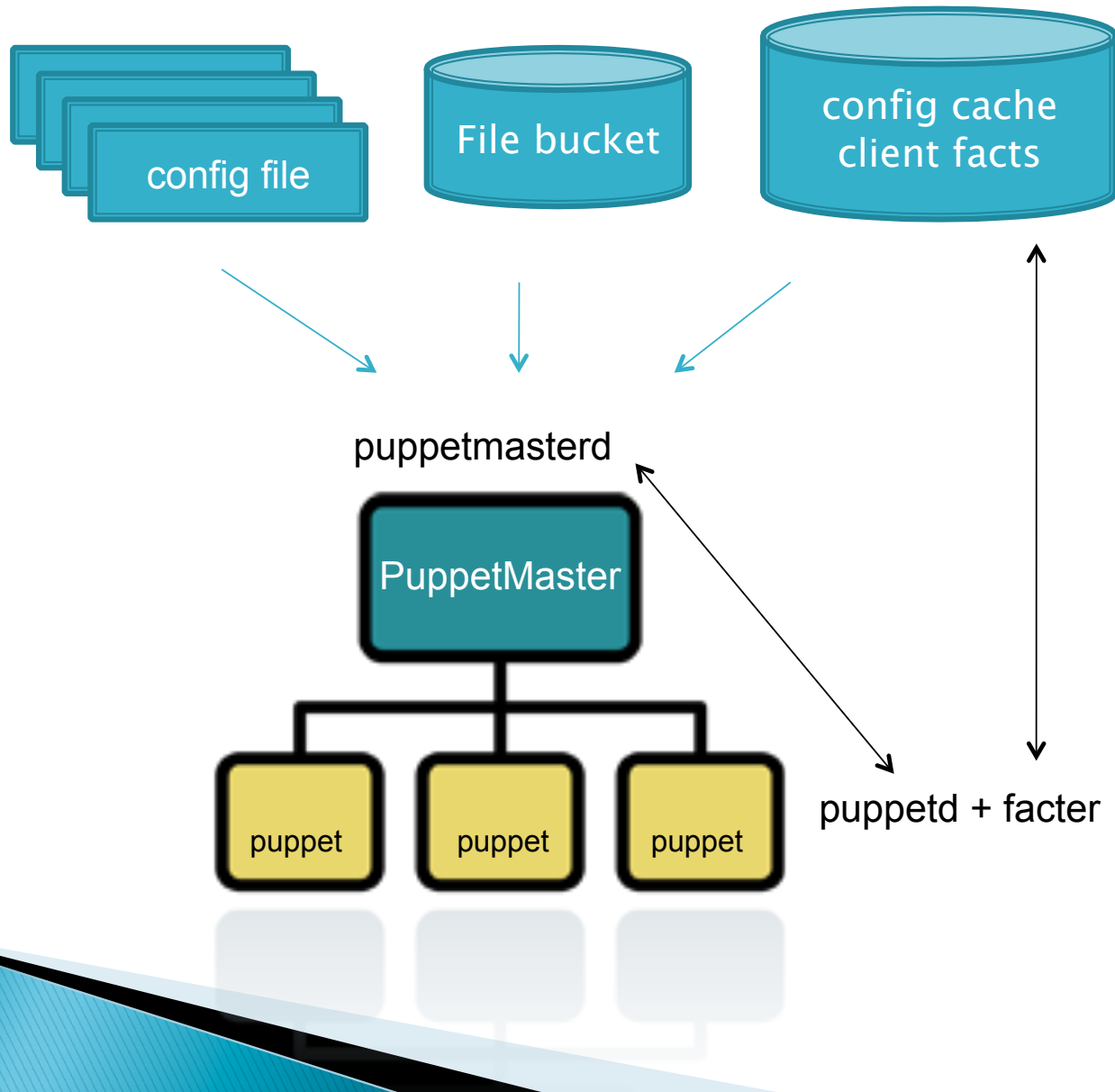
Puppet?



- Lenguaje de configuración desarrollado en *Ruby*
- Surge como alternativa de cfengine, adicionando nuevas funciones
- Fácil de extender, con comunidad de contribuyentes amplia
- Un servidor central (*puppetmaster* o “titiritero”) controla la configuración del resto de los “servidores titeres” (puppets)
- En los servidores “titeres”, el utilitario *facter* provee información detallada sobre la configuración y recursos locales



Componentes



Distribución común

Como instalar?

En Fedora y EPEL, como parte de la distribución:

```
# yum install puppet-server    <= puppetmaster
# yum install puppet           <= cliente
```

Paquetes disponibles para UNIX/Linux:

Ubuntu, Debian, SuSe, Mandriva, FreeBSD, otros...

(Visite: <http://reductivelabs.com/trac/puppet/wiki/DownloadingPuppet>)

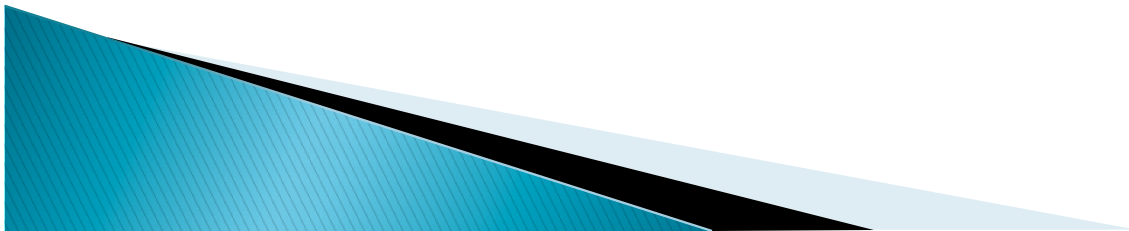
Configuración flexible:

/etc/puppet

ssl/	(SSL certificates)
manifests/site.pp	(config central)
definitions/	(patrones)
classes/	(clases)
nodes/	
users/	
(etc.)	

Elementos de configuración:

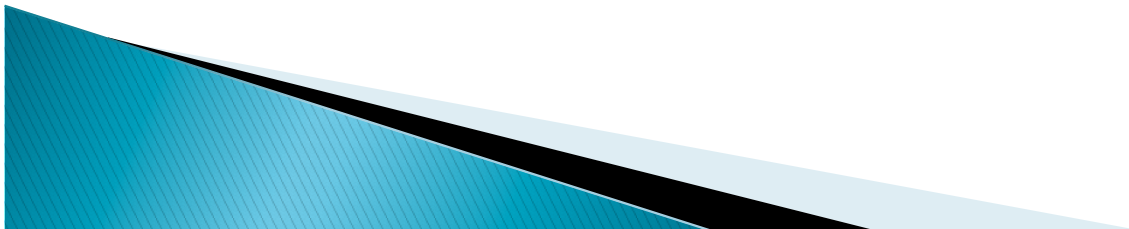
- ▶ Tipos
- ▶ Definiciones
- ▶ Modelos
- ▶ Clases
- ▶ Nodos...



Tipos



- ▶ elemento que Puppet sabe como configurar
 - Fichero (contenido, permisos, pertenencia)
 - Paquete (asegurar que esté instalado o no)
 - Servicio (habilitado, desahabilitado, corriendo, o detenido)
 - Biblioteca de tipos distribuidos con el paquete:
 - cron, exec, file, filebucket, group, host, mount, notify, package, service, user, zone, sshkey.....



Ejemplo: /etc/sudoers file



```
file { "/etc/sudoers":  
    ensure => file,  
    owner  => root,  
    group  => root,  
    mode   => 600,  
    source => "puppet://server/files/sudoers"  
}
```

Clases



► Colección de objetos (tipos):

```
class unix_users {  
  user { 'pandersen':  
    shell => '/bin/bash',  
    uid => '2010',  
    gid => "users",  
    home => '/home/pandersen',  
    managehome => true,  
    ensure => 'present'  
  }  
  user { 'tolick':  
    shell => '/bin/bash',  
    uid => '2013',  
    gid => "users",  
    home => '/home/tolick',  
    managehome => true,  
    ensure => 'present'  
  }  
}
```

Nodos



- ▶ Un bloque de configuración de un servidor cliente
- ▶ Puede contener clases, tipos
- ▶ Si existe un nodo “default”, se aplica a todos los clientes conocidos que no tienen una definición específica

```
node "otter.roundtripnetworks.com" {  
    include sudo_clase  
    include unix_users  
    include apache_clase  
}
```

```
node "default" {  
    include sudo_clase  
    include unix_users  
}
```

OK, configuremos con definiciones...

defintions/account.pp:

```
define account(  
  $uid,  
  $gid,  
  $groups,  
  $comment = "User $name",  
  $home = "/home/$name",  
  $shell = "/bin/bash",  
  $ensure = present )  
{  
  user { $name:  
    uid => $uid,  
    gid => $gid,  
    password => '*',  
    groups => $groups,  
    comment => $comment,  
    home => $home,  
    shell => $shell,  
    managehome => true,  
    ensure => $ensure,  
  }  
}
```

users/carmas.pp:

```
@account { "carmas":  
  ensure => present,  
  uid => 3011,  
  gid => "admins",  
  groups => [ "root", "wheel" ],  
  comment => "Carlos Armas",  
  home => "/home/carmas",  
  shell => "/bin/bash",  
  require => Group["admins"]  
}
```



classes/unix_users.pp:

```
class unix_users::admins {  
  realize(  
    Account["carmas"]  
  )  
}
```

classes/baseclass.pp:

```
class baseclass {  
  include unix_users::admins  
  group { "admins":  
    gid => 3000,  
    ensure => present  
  }  
}
```

Un usuario/admin "carmas" en cada servidor....
(pueden ser 1000 servidores!)

nodes/default.pp:

```
node default:  
{  
  include baseclass  
}
```

nodes/mailserver.pp:

```
node "mailserver":  
{  
  include baseclass  
}
```

nodes/mailserver.pp:

```
node "mailserver":  
{  
  include baseclass  
}
```

Con definiciones, adicionar usuarios es fácil...

```
users/carmas.pp:
@account { "carmas":
  ensure => present,
  uid => 3011,
  gid => "admins",
  groups => [ "root", "wheel" ],
  comment => "Carlos Armas",
  home => "/home/carmas",
  shell => "/bin/bash",
  require => Group["admins"]
}
```

```
users/john.pp:
@account { "john":
  ensure => present,
  uid => 3012,
  gid => "admins",
  groups => [ "root", "wheel" ],
  comment => "Johnny Root",
  home => "/home/john",
  shell => "/bin/bash",
  require => Group["admins"]
}
```



```
classes/unix_users.pp:
class unix_users::admins {
  realize(
    Account["carmas"]
    Account["john"]
  )
}
```

```
classes/baseclass.pp:
class baseclass {
  include unix_users::admins
  group { "admins":
    gid => 3000,
    ensure => present
  }
}
```

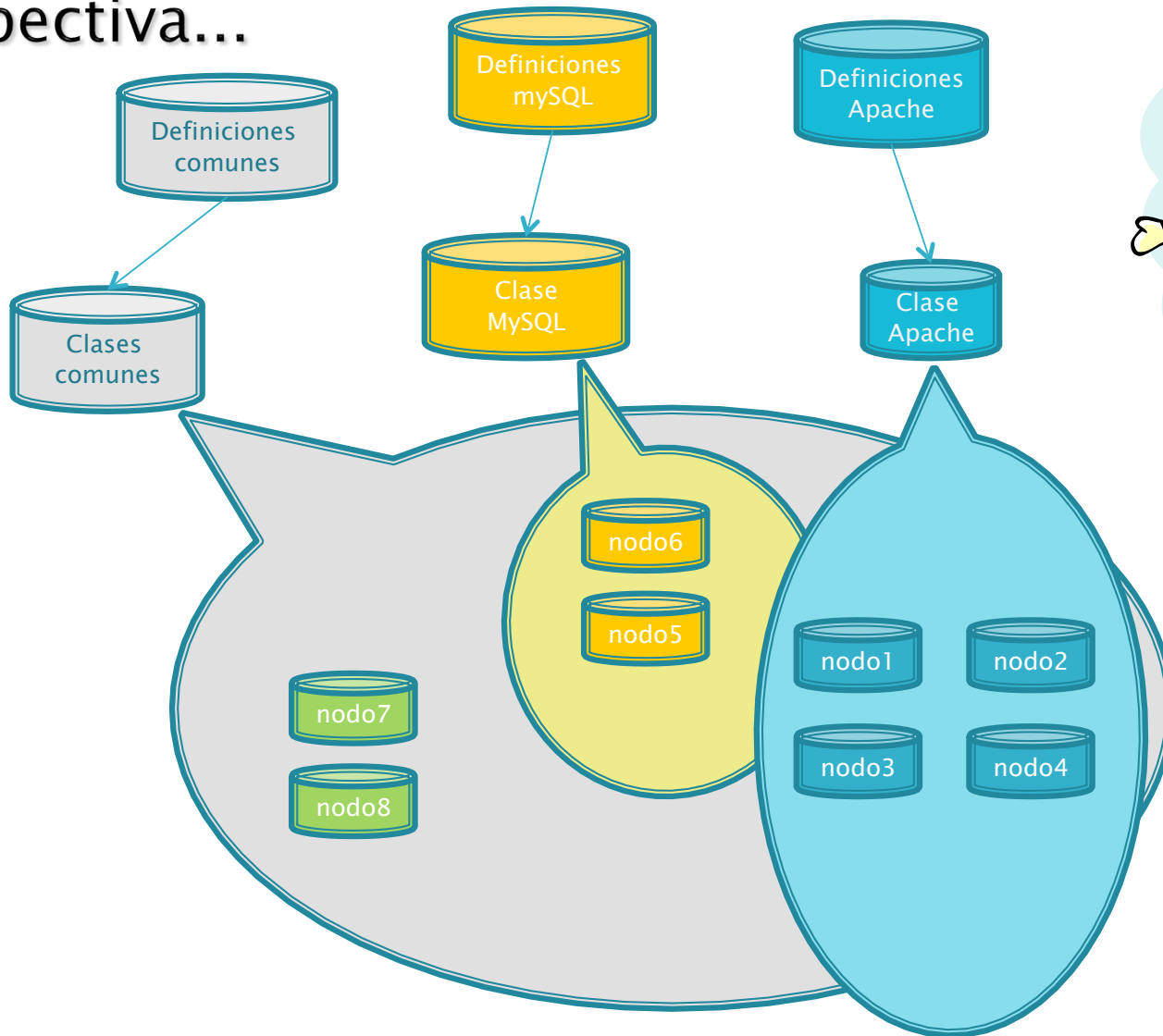
Ahora, adicionamos "john" en cada servidor (pueden ser 1000 servidores!) con solo un cambio en 2 files

```
nodes/default.pp:
node default:
{
  include baseclass
}
```

```
nodes/mailserver.pp:
node "mailserver":
{
  include baseclass
}
```

```
nodes/mailserver.pp:
node "mailserver":
{
  include baseclass
}
```

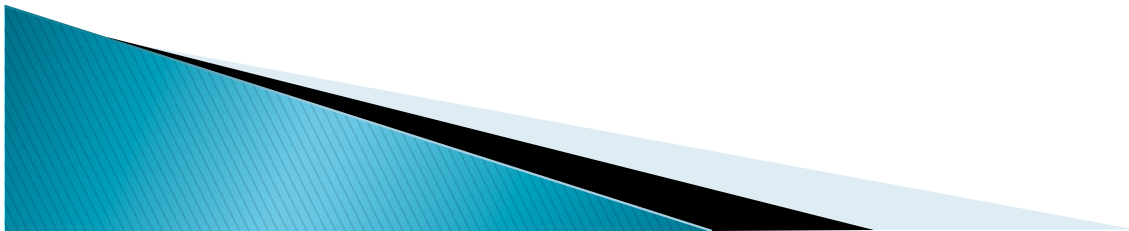
Perspectiva...



Contribuciones y ayuda



- ▶ La documentación es bastante informal
- ▶ Comunidad activa, muchos templates y configuraciones disponibles:
 - <http://reductivelabs.com/trac/puppet/wiki/ManagedByPuppet>
 - <http://reductivelabs.com/trac/puppet/wiki/Recipes>



Referencias

- *Puppet software*: <http://reductivelabs.com/trac/puppet>
- Book: “Pulling Strings with Puppet” – James Turnbull
- Some presentations:
<http://reductivelabs.com/trac/puppet/wiki/PuppetPresentations>

