

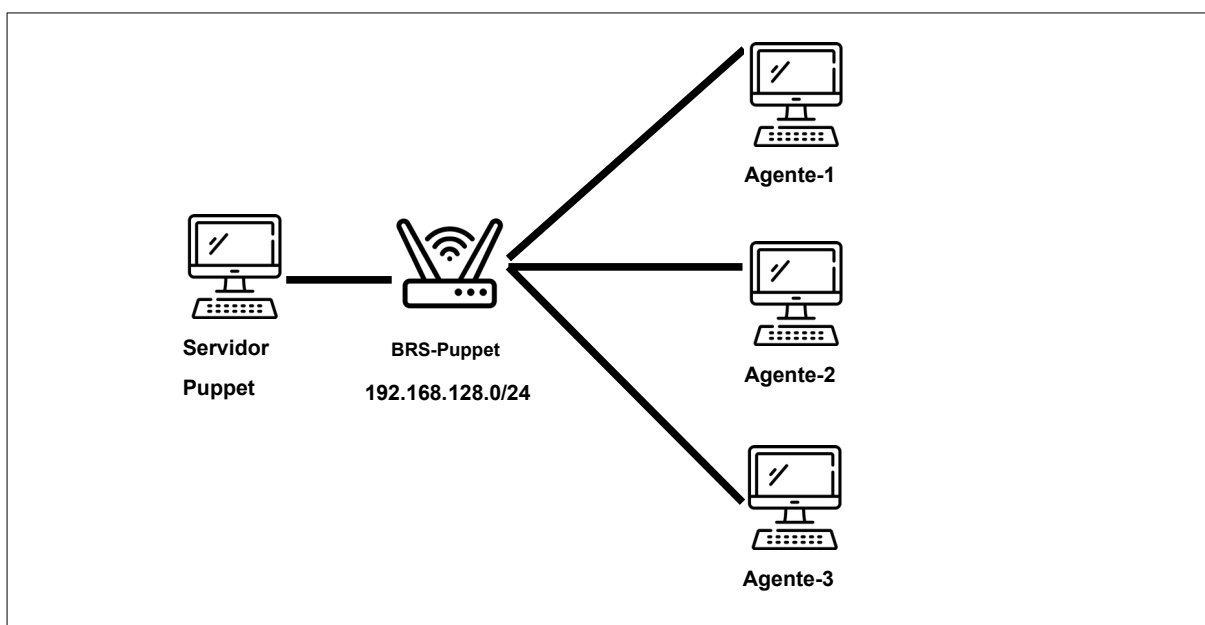
UT3. Guía de instalación del servidor y los agentes Puppet

Índice

0. Esquema del laboratorio
1. Configurar FQDN y /etc/hosts en ambas máquinas
2. Añadir el repositorio oficial de Puppet
3. Instalar y configurar el Servidor Puppet
4. Instalar y configurar el Agente Puppet
5. Registrar el Agente Puppet en el Servidor Puppet (certificados SSL)
6. Primer ejemplo de distribución de configuración (manifiesto LEMP)

0. Esquema del laboratorio

El laboratorio sigue el esquema que se puede apreciar en la siguiente imagen:



1. Configurar FQDN y /etc/hosts en ambas máquinas

La resolución de nombres es **clave** en Puppet; todo el sistema gira en torno a certificados con FQDN.

1.1. Cambiar el hostname completo (FQDN)

En **puppet-server**:

```
sudo hostnamectl set-hostname puppet-server.localdomain.lan
```

En **agent**:

```
sudo hostnamectl set-hostname agent.localdomain.lan
```

Comprueba en cada máquina:

```
hostname  
hostname -f
```

Deberías ver el FQDN completo.

1.2. Editar /etc/hosts en ambas máquinas

Edita el fichero:

```
sudo nano /etc/hosts
```

Añade (o adapta) las líneas:

```
192.168.128.100 puppet-server.localdomain.lan puppet-server  
192.168.128.150 agent.localdomain.lan agent
```

Nota didáctica: puedes pedir al alumnado que explique la diferencia entre hostname, FQDN y las entradas en /etc/hosts.

1.3. Verificar resolución

En **ambas** máquinas, prueba:

```
ping -c 3 puppet-server.localdomain.lan  
ping -c 3 agent.localdomain.lan
```

Si todo está bien, cada host debería resolver el FQDN del otro a la IP correspondiente.

2. Añadir el repositorio oficial de Puppet

PuppetLabs mantiene un repo específico para Debian 11 (Bullseye). Lo vamos a instalar en **las dos máquinas**.

En **puppet-server** y en **agent**:

```
cd /tmp
wget https://apt.puppet.com/puppet7-release-bullseye.deb

sudo dpkg -i puppet7-release-bullseye.deb
sudo apt update
```

Actividad sugerida: comprueba que se han añadido nuevas entradas de Puppet en `/etc/apt/sources.list.d/`.

3. Instalar y configurar el Servidor Puppet

3.1. Instalar el paquete puppetserver

En **puppet-server**:

```
sudo apt install puppetserver
```

Esto instalará Puppet Server y sus dependencias (OpenJDK, etc.).

3.2. Añadir Puppet al PATH

Los binarios se instalan en `/opt/puppetlabs/bin`. Vamos a asegurarnos de que están en el PATH del usuario:

```
source /etc/profile.d/puppet-agent.sh
echo $PATH
```

Opcionalmente, para dejarlo permanente en `~/.bashrc`:

```
echo 'export PATH=$PATH:/opt/puppetlabs/bin/' | tee -a ~/.bashrc
source ~/.bashrc
```

Comprueba:

```
which puppet
which puppetserver
```

Deberían apuntar a `/opt/puppetlabs/bin/puppet` y similares.

3.3. Ajustar memoria de Java para Puppet Server

Puppet Server corre sobre JVM. En un laboratorio con 2 GB de RAM, la guía sugiere asignar 1 GB a Puppet.

Edita:

```
sudo nano /etc/default/puppetserver
```

Busca la línea `JAVA_ARGS=` y pon, por ejemplo:

```
JAVA_ARGS="-Xms1g -Xmx1g"
```

- -Xms1g: memoria inicial de 1 GB.
- -Xmx1g: memoria máxima de 1 GB.

Guarda y cierra.

3.4. Activar y arrancar el servicio puppetserver

Recarga systemd (por si acaso):

```
sudo systemctl daemon-reload
```

Habilita y arranca Puppet Server:

```
sudo systemctl enable --now puppetserver
```

Comprueba el estado:

```
sudo systemctl status puppetserver
```

Debería estar en estado active (running).

3.5. (Opcional) Abrir el puerto en UFW

Si usas UFW en la máquina puppet-server:

```
sudo ufw allow from 192.168.128.0/24 to any proto tcp port 8140  
sudo ufw status
```

8140 es el puerto por defecto de Puppet Server para la comunicación con los agentes.

3.6. Configuración básica de Puppet Server

Vamos a usar el comando puppet config set para editar /etc/puppetlabs/puppet/puppet.conf sin tocar el fichero a mano.

En **puppet-server**:

```
puppet config set server puppet-server.localdomain.lan --section main  
puppet config set runinterval 1h --section main
```

```
puppet config set environment production --section server  
puppet config set dns_alt_names puppet-server,puppet-server.localdomain.lan --section server
```

- server: FQDN del Puppet Server.
- runinterval: cada cuánto se ejecuta el agente automáticamente (aquí, cada 1 hora).
- environment: entorno por defecto (production).
- dns_alt_names: nombres alternativos para el certificado SSL del server.

Revisa el resultado:

```
cat /etc/puppetlabs/puppet/puppet.conf
```

Reinicia Puppet Server para aplicar cambios:

```
sudo systemctl restart puppetserver
```

4. Instalar y configurar el Agente Puppet

4.1. Instalar puppet-agent

En la máquina **agent**:

```
sudo apt install puppet-agent
```

Esto instala el servicio puppet (el agent) y los binarios en /opt/puppetlabs/bin.

4.2. Arrancar y habilitar el servicio puppet

En **agent**:

```
sudo /opt/puppetlabs/bin/puppet resource service puppet ensure=running enable=true
```

Este comando usa Puppet para declararle a sí mismo que el servicio puppet debe estar en estado running y enable=true.

Comprueba con systemd:

```
sudo systemctl status puppet
```

4.3. Añadir Puppet al PATH del agente

Igual que en el servidor, en **agent**:

```
source /etc/profile.d/puppet-agent.sh  
echo $PATH
```

Opcional, dejarlo en ~/.bashrc:

```
echo 'export PATH=$PATH:/opt/puppetlabs/bin/' | tee -a ~/.bashrc  
source ~/.bashrc
```

Comprueba:

```
which puppet
```

4.4. Configurar el agente para que apunte al servidor

Primero, verifica que llega al Puppet Server por nombre:

```
ping -c 3 puppet-server.localdomain.lan
```

Después, ejecuta en **agent**:

```
puppet config set server puppet-server.localdomain.lan --section agent  
puppet config set ca_server puppet-server.localdomain.lan --section agent
```

- server: FQDN del Puppet Server.
- ca_server: FQDN del servidor que actúa como CA (normalmente el mismo Puppet Server).

Comprueba el puppet.conf del agente:

```
cat /etc/puppetlabs/puppet/puppet.conf
```

Reinicia el servicio puppet:

```
sudo systemctl restart puppet  
sudo systemctl status puppet
```

5. Registrar el Agente Puppet en el Servidor Puppet (certificados SSL)

Puppet usa certificados TLS para autenticar servidor y agentes. El flujo es:

1. El **agente** genera un CSR hacia el servidor.
2. El **servidor** firma el certificado.
3. El **agente** descarga el certificado firmado y se establece la confianza.

5.1. Lanzar el bootstrap del agente

En **agent**:

```
puppet ssl bootstrap
```

Este comando:

- Genera claves y CSR.
- Contacta con el Puppet Server.
- Queda “a la espera” de que el certificado sea firmado.

5.2. Firmar el certificado del agente en el servidor

En **puppet-server**, lista las peticiones pendientes:

```
puppetserver ca list --all
```

Deberías ver algo como agent.localdomain.lan en estado Requested.

Firma el certificado:

```
puppetserver ca sign --certname agent.localdomain.lan
```

Comprueba todos los certificados:

```
puppetserver ca list --all
```

Ahora deberías ver dos entradas principales:

- El certificado del propio Puppet Server.
- El certificado del agente agent.localdomain.lan.

5.3. Verificar en el agente

Vuelve a **agent**. El comando puppet ssl bootstrap debería completar con mensajes del tipo:

Notice: Completed SSL initialization

Si quieres forzar una primera ejecución:

```
puppet agent -t
```

6. Primer ejemplo de distribución de configuración (manifiesto LEMP)

Para que tu alumnado vea el valor de Puppet, vamos a crear un **módulo sencillo** que instala un stack LEMP (Nginx + MariaDB + PHP-FPM) en el agente y genera una página HTML de bienvenida. Esto sigue la estructura de la guía original.

Todo esto se hace en **puppet-server**.

6.1. Movernos al entorno de producción

```
cd /etc/puppetlabs/code/environments/production/
```

6.2. Crear la estructura del módulo lemp

```
sudo mkdir -p modules/lemp/{manifests,files}
cd modules/lemp
```

- manifests/: ficheros .pp con código Puppet.
- files/: ficheros estáticos que podrías distribuir (plantillas, config, etc.).

6.3. Crear el manifiesto principal init.pp

Edita:

```
sudo nano manifests/init.pp
```

Contenido de ejemplo:

```
class lemp {
  Package { ensure => 'installed' }

  $lemppackages = [ 'nginx', 'mariadb-server', 'php-fpm' ]
  package { $lemppackages: }
```

```
Service { ensure => 'running', enable => true }

$lempsvc = [ 'nginx', 'mariadb', 'php7.4-fpm' ]
service { $lempsvc: }

file { '/var/www/html/index.html':
  ensure => file,
  content => "<h1><center>Welcome to Nginx - Managed by Puppet</center></h1>",
  mode    => '0644',
}
}
```

Puntos didácticos a comentar:

- Definición de clase lemp.
- Uso de **resource defaults** (Package { ... }, Service { ... }).
- Uso de **arrays** (\$lempackages, \$lempsvc) para gestionar varios paquetes/servicios.
- Recurso file para distribuir un contenido fijo (aquí, una página HTML).

Valida el manifiesto:

```
sudo puppet parser validate manifests/init.pp
```

Si no hay salida, la sintaxis es correcta.

6.4. Asociar el módulo lemp al nodo agente

Volvemos al directorio de producción:

```
cd /etc/puppetlabs/code/environments/production/
```

Edita el fichero de nodos (usa site.pp; la guía usa site(s).pp, ajusta a tu gusto, pero sé coherente):

```
sudo nano manifests/site.pp
```

Contenido:

```
node 'agent.localdomain.lan' {
  include lemp
}
```

Valida:

```
sudo puppet parser validate manifests/site.pp
```

Actividad: prueba a definir un nodo default que se aplique a cualquier host que no coincida con un FQDN concreto.

6.5. Aplicar el manifiesto desde el agente

En **agent**, fuerza una ejecución:

puppet agent -t

Si todo está bien, Puppet:

- Instalará nginx, mariadb-server y php-fpm.
- Asegurará que nginx, mariadb y php7.4-fpm estén arrancados y habilitados.
- Creará /var/www/html/index.html con el contenido indicado.

Comprueba servicios:

```
sudo systemctl status nginx  
sudo systemctl status mariadb  
sudo systemctl status php7.4-fpm
```

Y desde tu navegador, accede a:

<http://192.168.128.150/>

Deberías ver la página “Welcome to Nginx - Managed by Puppet”.