EBook Gratis

APRENDIZAJE puppet

Free unaffiliated eBook created from
Stack Overflow contributors.

#puppet

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: puppet

It is an unofficial and free puppet ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official puppet.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con el títere

Observaciones

Esta sección proporciona una descripción general de qué es títere y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema importante dentro de títere y vincular a los temas relacionados. Dado que la Documentación para títere es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Versiones

Títere Open Source

Versión	Fecha de lanzamiento
4.7.0	2015-09-22
4.6.0	2015-08-10
4.5.0	2015-05-17
4.4.0	2016-03-16
4.3.0	2015-11-17
4.2.0	2015-06-24
4.1.0	2015-05-19
4.0.0	2015-04-15
3.8.0	2015-04-28
3.7.0	2014-09-04
3.6.0	2014-05-15
3.5.0	2014-04-03
3.4.0	2013-12-19
3.3.0	2013-09-12
3.2.0	2013-05-14

Versión	Fecha de lanzamiento
3.1.0	2013-02-04
3.0.0	2012-09-28
2.7.0	2011-06-17
2.6.0	2010-07-20
0.25.0	2009-09-05
0.24.0	2007-12-13
0.23.0	2007-01-20
0.22.0	2007-01-06
0.20.0	2006-10-18
0.19.0	2006-09-07
0.18.0	2006-06-14
0.17.0	2006-05-16
0.16.0	2006-04-21
0.15.0	2006-03-13
0.14.0	2006-03-06
0.13.0	2006-02-09
0.12.0	2006-01-26
0.11.0	2006-01-17
0.10.0	2006-01-09
0.9.3	2006-01-03
0.9.2	2005-11-22

Examples

¿Qué es el títere y por qué debería importarme?

Puppet es una solución de gestión de la configuración. Los usuarios describen el estado deseado de un servidor o software y la administración de la configuración logra este estado. Esto trae las

siguientes ventajas:

- Las configuraciones se pueden reproducir exactamente igual cada vez, tantas veces como sea necesario
- Las configuraciones para todo el software y los servidores se almacenan en una ubicación central. Esto hace que la copia de seguridad y el control de versiones de las configuraciones sean fácilmente alcanzables
- Los cambios en todos los servidores se propagan a través de toda la infraestructura en un par de minutos, sin tener que iniciar sesión en ninguna máquina directamente
- Todo se describe en el mismo idioma, lo que facilita la configuración del nuevo software.
- Los módulos son similares a las bibliotecas y permiten que las configuraciones se consoliden. Ya existen módulos para todos los paquetes de software principales, lo que hace que su instalación sea extremadamente fácil
- Los servidores pueden compartir información entre ellos, influyendo en la configuración de otros servidores. Por ejemplo, un nuevo servidor puede registrarse automáticamente con el equilibrador de carga y la solución de monitoreo

Puppet usa Ruby para describir el estado deseado de un servidor, llamado **nodo** . Lo hace con el uso de primitivas llamadas **tipos de recursos** . De forma predeterminada, cada 30 minutos, el **agente de** títere se autentica contra el **servidor de** títere. A continuación, envía una lista de propiedades de sí mismo llamados **hechos** . El servidor analiza los hechos y los archivos de configuración llamados **manifiestos** y compila el estado deseado para el nodo. A continuación, envía esa **configuración** al nodo, donde el agente la aplica.

Para dar una idea de lo poderoso que puede ser esto, aquí hay un par de ejemplos de complejidad creciente que muestran lo que el títere puede hacer por usted.

Ejemplo: Usuario

Este ejemplo crea el *nombre* de *usuario* en el nodo *myserver* y lo agrega a la *rueda de* grupo.

```
node 'myserver' {
    user { 'username':
      ensure => 'present',
      groups => ['wheel'],
    }
}
```

Este archivo que se almacenaría en el *servidor* Puppet es el *manifiesto* . El *tipo de recurso* en este ejemplo es *usuario* . Cada tipo de recurso tiene propiedades opcionales y requeridas. En este ejemplo, *asegúrese de que* sea necesario y que los *grupos* sean opcionales. Esta configuración específica solo se aplicaría a *myserver* . Puede aplicar configuraciones a todos los nodos colocándolos fuera de una definición de nodo.

Es posible tomar un par de definiciones de recursos y almacenarlas como **módulos** . Un módulo es similar a una biblioteca. Estos módulos se pueden compartir en línea y, por lo general, encontrará uno para cada paquete de software importante. La forma oficial de compartir módulos es a través de la forja de títeres: https://forge.puppet.com/

Ejemplo: Postgres

Este ejemplo instala un servidor postgres en el nodo *myserver* y crea una base de datos *db* , propiedad de *nombre* de *usuario* , identificada por *contraseña* . Lo hace utilizando el módulo postgresql.

```
node 'myserver' {
    class { 'postgresql::server': }

    postgresql::server::db { 'db':
        user     => 'username',
        password => 'password',
    }
}
```

En este caso *postgresql* es un módulo. El módulo en sí se encarga de identificar el sistema operativo, descargar e instalar el programa y luego configurarlo de acuerdo con el manifiesto. Este es un ejemplo básico, pero el módulo permite una gran personalización.

Tenga en cuenta que no es necesario conocer SQL o cómo instalar realmente un servidor Postgres para hacerlo. Los módulos oficiales están bien mantenidos y proporcionan una configuración de base sana y segura.

También es posible utilizar *hechos* en manifiestos. Los hechos actúan como variables.

Ejemplo: Condiciones que utilizan hechos.

Este ejemplo utiliza el módulo rsyslog para configurar rsyslog en todas las máquinas que no sean Windows.

```
if $osfamily != 'windows' {
  class { 'rsyslog::client': }
}
```

$ osfamilia es un hecho. Estos hechos se recogen cada vez que corre el agente títere. Tenga en cuenta que debido a que esta definición está fuera de la definición de un nodo, se aplica a todos los nodos. Sin embargo, *rsyslog :: client* solo se ejecutará en nodos que no ejecuten Windows.

Como el títere usa rubí, los elementos programáticos como los flujos de control y las variables se pueden usar en manifiestos.

Con la adición de **PuppetDB** puede compartir información entre varios nodos. Esto permite que un nodo influya en la configuración en un nodo diferente. Los ejemplos clásicos incluyen balanceadores de carga o soluciones de monitoreo.

Ejemplo: registro de un host con monitoreo usando recursos exportados

Este ejemplo crea un **recurso exportado** en un nodo y luego importa ese recurso en el servidor de monitoreo, agregando el host al monitoreo. Está utilizando el módulo títere Icinga2.

```
@@icinga2::object::host { $::fqdn:
```

```
  display_name     => $::fqdn,
  ipv4_address     => $::ipaddress_eth0,
}

node 'icinga2' {
    Icinga2::Object::Host <<| |>> { }
}
```

@@ icinga2 :: object :: host crea un objeto de definición de host. Esto se crea por cada nodo que ejecuta este código. El *@@ lo* marca como un *recurso exportado* . Por lo general, los nodos no comparten información en títeres. Los recursos exportados permiten hacer eso.

Tenga en cuenta que todos los valores de propiedad en la definición de host son hechos. Esto significa que serán diferentes para cada nodo que lo ejecute.

Finalmente, el recurso exportado se *importa* por el nodo *icinga2* . El módulo Icinga2 es responsable de asegurarse de que se creen y se vuelvan a cargar los archivos de configuración correctos.

¿Es para ti?

Si realiza implementaciones, configura sus aplicaciones en múltiples servidores y es necesario que inicie sesión en sus servidores y realice algunos cambios en la infraestructura, las aplicaciones, los requisitos previos, etc., entonces, definitivamente, Puppet puede ayudarlo.

Excepto todo esto, si maneja una gran infraestructura y desea una administración centralizada, también puede echar un vistazo.

Antes de iniciar

Antes de que decida trabajar en títere, hay algunas cosas que debe saber.

1. trabajo títere tanto en la arquitectura cliente-servidor (ampliamente utilizada) como en una sola máquina (especialmente para fines de prueba)
2. Puppet Master solo se puede configurar en una máquina Linux (máquina maestra / nodo), las ventanas solo se pueden usar como cliente (máquina administrada / nodo)
3. Si está configurando el maestro, debe tener en cuenta el uso de Linux y los comandos básicos.
4. Títere proporciona su propio lenguaje de configuración que se parece a json

Documentacion oficial

Puppet proporciona documentación oficial tanto para versiones de código abierto como para versiones empresariales. puedes encontrarlo aqui

Instalación

Requisitos del sistema

Sin embargo, el servicio maestro de Puppet requiere bastante recursos y debe instalarse en un servidor dedicado robusto.

- Como mínimo, su servidor maestro Puppet debe tener dos núcleos de procesador y al menos 1 GB de RAM.
- Para servir cómodamente al menos 1,000 nodos, debe tener 2-4 núcleos de procesador y al menos 4 GB de RAM.

Compruebe su configuración de red:

En una implementación de agente / maestro, debe preparar su red para el tráfico de Puppet.

- **Cortafuegos:** el servidor maestro Puppet debe permitir las conexiones entrantes en el puerto 8140, y los nodos del agente deben poder conectarse al maestro en ese puerto.
- **Resolución de nombres:** cada nodo debe tener un nombre de host único. Los DNS hacia adelante y hacia atrás deben estar configurados correctamente.

  Nota: El nombre de host predeterminado de Puppet master es puppet. Sus nodos de agente pueden estar listos antes si este nombre de host se resuelve en su maestro Puppet.

  El tiempo se debe establecer con precisión en el servidor maestro de Puppet que actuará como autoridad de certificación. Probablemente debería utilizar NTP.

Instalación de Puppet Server

Puppet proporciona paquetes oficiales que instalan Puppet Server 2.4 y todos sus requisitos previos en las siguientes plataformas.

Red Hat Enterprise Linux

- Enterprise Linux 6
- Enterprise Linux 7

Debian

- Debian 7 (Wheezy)
- Debian 8 (Jessie)

Ubuntu

- Ubuntu 12.04 (preciso)
- Ubuntu 14.04 (Trusty)

- Ubuntu 15.10 (astuto)
- Ubuntu 16.04 (Xenial)

Habilitar los repositorios del paquete Puppet.

Enterprise Linux 7

```
sudo rpm -Uvh https://yum.puppetlabs.com/puppetlabs-release-pc1-el-7.noarch.rpm
```

Para otras versiones mira aquí.

Instalación de títere maestro

```
yum install puppetserver
```

o

```
apt-get install puppetserver
```

> Puppet Server está configurado para utilizar 2 GB de RAM de forma predeterminada. Para cambiar mira aqui

Inicia el servicio Puppet Server:

```
systemctl start puppetserver
```

o

```
service puppetserver start
```

Lea Empezando con el títere en línea: https://riptutorial.com/es/puppet/topic/2871/empezando-con-el-titere

Capítulo 2: Agente

Sintaxis

1. agente títere [--nombre_de_cargo] [-D | --daemonize | --no-daemonize] [-d | --debug] [--códigos de salida detallados] [--digest DIGEST] [--disable [MESSAGE]] [--enable] [--fingerprint] [-h | --help] [-l | --logdest syslog | eventlog | FILE | console] [--masterport PORT] [--noop] [-o | - -un tiempo] [-t | --test] [-v | --verbose] [-V | --version] [-w | --waitforcert SECONDS]

Examples

¿Qué es?

El agente títere es un servicio que se ejecuta en los servidores. Una vez que se inicia el servicio, el agente se activará en segundo plano cada 30 minutos (de forma predeterminada).

El agente tiene 2 usos principales:

- Enviar datos del servidor al títere maestro
- Recibe el catálogo del maestro títere y aplícalo.

Desencadenar

Por defecto, el agente se activa cada 30 minutos. Este valor de intervalo se puede cambiar desde el archivo `puppet.conf` .

- Linux- `/etc/puppet/puppet.conf`
- Windows - `%PROGRAMDATA%\PuppetLabs\puppet\etc\puppet.conf`

Establezca el `runinterval` de `runinterval` en el intervalo deseado.

```
runinterval=xxx
```

El agente se puede activar manualmente con el comando:

```
puppet agent –t
```

Salida verbosa

A veces es útil obtener más salida en la ejecución del agente títere.

Es muy útil para la depuración.

Ejecute el agente títere con parámetros `verbose` y de `debug` :

- `debug` : habilite la depuración completa.
- `verbose` - Activa el informe detallado.

```
puppet agent -t --verbose --debug
```

Explotación florestal

Puppet agnet registra mensajes. Puedes ver estos registros aquí:

Linux - `/var/log/puppet/puppet.log`

Windows : vea el `Event Viewer` (Panel de control → Sistema y seguridad → Herramientas administrativas → Visor de eventos)

Lea Agente en línea: https://riptutorial.com/es/puppet/topic/7234/agente

Capítulo 3: Manejo de montaje NFS

Introducción

NFS es la forma más común de compartir discos entre computadoras en Linux. Permite al usuario en una computadora cliente acceder a los archivos a través de una red de manera similar al acceso al almacenamiento local. Aquí vemos cómo configurar Puppet para administrar el montaje y el servicio de unidades NFS.

Parámetros

Parámetro	Detalles
nombre	La ruta al directorio local en el que se debe montar la unidad remota.
dispositivo	La dirección del servidor remoto y la ruta del directorio en el servidor remoto, separados por :
atboot	Si esta unidad debe montarse durante el arranque. La habilitación hace que las unidades estén disponibles antes, pero puede causar un retraso en el arranque en caso de problemas de red o de montaje.
pasar	El orden Fsck es decirle a fsck qué orden para verificar los sistemas de archivos, si se configura en "0" el sistema de archivos se ignora. Por lo general, las unidades NFS no necesitan ser verificadas en los clientes, por lo que "0" es una opción adecuada.

Observaciones

- El directorio de destino de montaje debe existir en el cliente.
- Montar documentación de tipo de recurso.
- Descripción de fstab y detalles de la opción

Examples

Montaje de una unidad NFS remota

```
mount { '/path/to/local/folder':
  ensure  => 'mounted',
  atboot  => false,
  device  => 'server-ip-or-domain:/path/to/server/folder',
  fstype  => 'nfs',
  options => 'defaults',
  pass    => 0,
}
```

Lea Manejo de montaje NFS en línea: https://riptutorial.com/es/puppet/topic/8044/manejo-de-montaje-nfs

Creditos

S. No	Capítulos	Contributors
1	Empezando con el títere	Ankit Katiyar, Community, Matthieu FAURE, Mor Paz, mzhaase, Peter Souter, Quill
2	Agente	Oz Bar-Shalom
3	Manejo de montaje NFS	Amir Ali Akbari