



Práctica 3

Dado que los ejercicios que se presentaron en las unidades poseen una característica incremental, lo visto en esta práctica tomará como base lo anterior instalado y configurado. Para ello se instalará una aplicación para automatizar el proceso de Integración Continua y otra para llevar adelante la gestión de configuración de ambientes.

Se instalará el software de aprovisionamiento Puppet el cual está escrito en el lenguaje de programación Ruby. Puppet nos permitirá gestionar las configuraciones y las distintas dependencias de software que tenga nuestro ambiente de una manera explícita, declarando el estado en que queremos que ese encuentre nuestra máquina. Una vez declarado el estado deseado, Puppet se encargará de hacer lo necesario para hacer que esa máquina se encuentre en dicho estado. La principal ventaja de esta herramienta es la simplificación de la administración de un gran volumen de servidores, los cuales pueden estar bajo sistemas operativos Linux o Windows y poseer un detalle de dichos cambios.

Anteriormente hemos instalado el software Vagrant para la gestión de nuestro ambiente virtual. Vagrant soporta el aprovisionamiento de software a través de Puppet en su configuración pero no será nuestro objetivo, sino que lo que queremos es que puedan obtener el conocimiento para comprender el funcionamiento básico de esta herramienta sin la presencia de Vagrant, por lo cual el ejemplo se centrará en la configuración de Puppet sin la integración con Vagrant.

Puppet utiliza archivos de configuración llamados manifiestos (la extensión del archivo es **.pp**). Es bueno aclarar que el contenido de los manifiestos no es ejecutado en un orden secuencial sino que Puppet armará su propia lista de ejecución a partir de las dependencias establecidas.

La arquitectura de Puppet es de cliente/servidor, es decir un equipo actuará como maestro y uno o más agentes estarán instalados en los nodos que requieren el aprovisionamiento. No es lo recomendable pero a fines de realizar un ejemplo práctico, se instalará Puppet como maestro y agente en un mismo entorno. La comunicación que se realiza entre cliente-servidor o agente-maestro se realiza mediante encriptación a través de certificados SSL (el nodo maestro será la autoridad certificante). El tipo de comunicación será push hacia el maestro, es decir que los agentes se comunicarán automáticamente al servidor para buscar



novedades de actualización (por defecto esto ocurrirá cada 30 minutos), de esta manera al tener un nodo central las configuraciones que se realicen serán impactadas en los nodos agentes. Al establecerse la comunicación cada cierto periodo de tiempo es fundamental que el nodo maestro tenga un servicio de configuración automático de su huso horario para evitar errores en las sincronizaciones. Cuando el agente se comunica con el maestro le envía una serie de datos (facts) que se verifican con la configuración establecida como estado deseado para dicho nodo, en el caso de que no coincida el agente se encargará de alcanzar dicho estado, por ejemplo una actualización de un servicio, de manera automática a través de lo declarado en el manifiesto.

La manera de asignar los nodos agentes que poseerán conexión al nodo maestro se realiza mediante nombres de dominio. El ejemplo de este curso se remite a la configuración manual de los nombres de dominio, sin tener que configurar un servidor DNS pero recomendando que en un ambiente en donde se utilizará este tipo de herramientas se realice mediante un servidor DNS para facilitar la administración de los nodos.

Esta herramienta se provee de manera gratuita y de código abierto y también se puede encontrar una versión paga, Puppet Enterprise, la cual posee más funcionalidades.

En el directorio `utn-devops/hostConfigs/puppet` se encuentran todos los archivos que se usan en la práctica para la ejecución de Puppet, estos contienen nombres diversos ya que lo que se hace es transferirlos desde la máquina host a la máquina virtual (esto lo pueden ver en el archivo `Vagrantfile`) y luego se renombran a su correspondiente directorio mediante la ejecución del archivo `Vagrant.bootstrap.sh`. El archivo `site.pp` es el archivo de inicio de Puppet desde allí se declaran las políticas de los módulos ya creados que se desean aplicar a los distintos nodos. La estructura de directorio de cómo queda un módulo lo pueden ver ingresando a la máquina virtual y navegar el directorio `/etc/puppet/code/environments/production/modules` para verlos. Igualmente la estructura de un módulo es muy sencilla, sigue el siguiente esquema:

`/etc/puppet/code/environments/production/modules/NOMBREMODULO`

Directorio - files (contiene los archivos a transferir)

Directorio - manifests (contiene las clases que se crean, de haber al menos una que contenga el nombre `init.pp` para que Puppet reconozca el módulo)

Comencemos

El ejercicio consistirá en instalar Jenkins mediante Puppet, para ello se hará una declaración en el manifiesto (archivo de configuración) para que Puppet transfiera el archivo de configuración de nuestra aplicación, de esta manera podremos, si se quisiera, independizarnos de Vagrant y crear un repositorio de código para las configuraciones de nuestra aplicación (práctica recomendada)..

Jenkins, es un software de automatización de compilación, de pruebas y de despliegue en ambientes, es decir que automatiza el pipeline de un producto de software. Es una herramienta gratuita y de código abierto, muy utilizada para la automatización de la práctica de Integración Continua y Deployment. Esta herramienta está basada en la gestión de tareas (Jobs) en los cuales se pueden especificar distintas características tales como su orden de ejecución, las condiciones necesarias para que ejecute y la tarea en sí misma. Por otro lado, Jenkins provee una interfaz gráfica que permite visualizar métricas de la salud del código del software que se está automatizando.

Importante: en esta unidad sólo se instalará Jenkins y algunos plugins, no habrá ningún tipo de configuración adicional y ejecución de tareas.

1. Abrir una terminal de comandos (presionar las teclas "Windows" + r, escribir "cmd" y luego hacer clic en "Ok"). Luego de cada comando debe presionar la tecla "Enter" para que se ejecute"
2. `cd UTN-DevOps/utn-devops`
3. `git checkout unidad-3-puppet`
4. `git pull`
5. `vagrant up --provision`
 - a. Este comando demora unos cuantos minutos. Si durante mucho tiempo no finaliza presione al mismo tiempo el nombre de las teclas que se encuentran entre comillas: "ctrl" + "c"
 - b. Vuelva a ejecutar el mismo comando: `vagrant up --provision`
6. `vagrant ssh`

Configuración de Puppet. Se enviará una petición a Puppet Master para que acepte las peticiones del agente que acabamos de instalar, recordemos que en este caso es el mismo equipo. El master realizará una serie de configuraciones para aceptar al agente que realizó la petición (análisis de certificados SSL). Este comando en otro tipo de configuración se debería ejecutar en el nodo que contiene

Centro de e-Learning SCEU UTN - BA.

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148

www.sceu.frba.utn.edu.ar/e-learning

solamente el Puppet agente.

1. `sudo puppet agent -tv`
 - a. Cuando la terminal deje de mostrar actividad presionar las teclas en conjunto control y c. Es decir ctrl+c. Esto finalizará el comando anterior
 - b. Este comando se utiliza para probar las directivas que se configuraron para un agente de Puppet
2. `sudo puppet cert sign utn-devops.localhost`
 - a. Esta llamada se debería ejecutar en el master. Se utiliza para generar los certificados de seguridad con el nodo agente
3. `sudo puppet agent -tv`
 - a. Ejecutar una segunda llamada para que vuelva a tomar la configuración una vez firmado el certificado de seguridad con el nodo agente. Se instalará todo el servidor Jenkins con el software adicional que necesite para ejecutar el ciclo de Integración Continua. Todo lo instalado se realizará mediante Puppet, por lo cual para analizar qué se instaló y configuró se pide revisar el archivo que se encuentra en su computadora
“UTN-DevOps\utn-devops\hostConfigs\puppet\init_jenkins.pp” Esto puede demorar bastante tiempo. Esperar hasta que el comando libere la terminal.
 - b. Las directivas y la manera en que se encuentra estructurado un módulo de Puppet se pueden ver navegando el directorio “/etc/puppet/code/environments/production/modules/jenkins/” de la máquina virtual
 - i. `ls -l /etc/puppet/code/environments/production/modules/jenkins/`
 - ii. Al realizar ese comando se mostrarán dos directorios “files” y “manifests”. El primero contiene los archivos que se desean transferir desde el master a los nodos si se usara la directiva “source => 'puppet:///modules/NOMBREMODULO/ARCHIVO'”. El directorio manifest posee la declaración de la clase que contendrá los atributos de lo que se desee instalar y/o configurar
En el directorio del curso “utn-devops\hostConfigs\puppet” pueden ver los archivos de configuración de Puppet los cuales poseen comentarios dentro que explican cada directiva



```
Debug: Exec[config-app](provider=posix): Executing check 'test $(docker ps |grep apache)'
Debug: Executing 'test $(docker ps |grep apache)'
Debug: /Stage[main]/Docker_install/Exec[config-app]/onlyif: sh: 1: grep: not found
Debug: Executing '/usr/bin/dpkg-query -W --showformat='${(Status)} ${Package} ${Version}\\n' openjdk-8-jre'
Debug: Executing '/usr/bin/apt-get -q -y -o DPkg::Options::=--force-confold install openjdk-8-jre'

Notice: /Stage[main]/Jenkins/Package[openjdk-8-jre]/ensure: ensure changed 'purged' to 'present'
Debug: /Package[openjdk-8-jre]: The container Class[Jenkins] will propagate my refresh event
Debug: Exec[install_jenkins_key](provider=posix): Executing '/usr/bin/wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -'
Debug: Executing '/usr/bin/wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -'
Notice: /Stage[main]/Jenkins/Exec[install_jenkins_key]/returns: executed successfully
Debug: /Stage[main]/Jenkins/Exec[install_jenkins_key]: The container Class[Jenkins] will propagate my refresh event
Debug: Executing '/usr/bin/dpkg-query -W --showformat='${(Status)} ${Package} ${Version}\\n' jenkins'
Debug: Executing '/usr/bin/apt-get -q -y -o DPkg::Options::=--force-confold install jenkins'
Notice: /Stage[main]/Jenkins/Package[jenkins]/ensure: ensure changed 'purged' to 'present'
Debug: /Package[jenkins]: The container Class[Jenkins] will propagate my refresh event
Debug: Exec[replace_jenkins_port](provider=posix): Executing '/bin/sed -i -- 's/HTTP_PORT=8080/HTTP_PORT=8082/g' /etc/default/jenkins'
Debug: Executing '/bin/sed -i -- 's/HTTP_PORT=8080/HTTP_PORT=8082/g' /etc/default/jenkins'
Notice: /Stage[main]/Jenkins/Exec[replace_jenkins_port]/returns: executed successfully
[Info: /Stage[main]/Jenkins/Exec[replace_jenkins_port]: Scheduling refresh of Service[jenkins]
Debug: /Stage[main]/Jenkins/Exec[replace_jenkins_port]: The container Class[Jenkins] will propagate my refresh event
Debug: Exec[reload-systemctl](provider=posix): Executing '/bin/systemctl daemon-reload'
Debug: Executing '/bin/systemctl daemon-reload'
Notice: /Stage[main]/Jenkins/Exec[reload-systemctl]/returns: executed successfully
Debug: /Stage[main]/Jenkins/Exec[reload-systemctl]: The container Class[Jenkins] will propagate my refresh event
Debug: Exec[apt-get update](provider=posix): Executing '/usr/bin/apt-get update'
Debug: Executing '/usr/bin/apt-get update'
Notice: /Stage[main]/Jenkins/Exec[apt-get update]/returns: executed successfully
Debug: /Stage[main]/Jenkins/Exec[apt-get update]: The container Class[Jenkins] will propagate my refresh event
Debug: Executing '/bin/systemctl is-active jenkins'
Debug: Executing '/bin/systemctl is-enabled jenkins'
Debug: Executing '/bin/systemctl is-active jenkins'
Debug: Executing '/bin/systemctl restart jenkins'
Notice: /Stage[main]/Jenkins/Service[jenkins]: Triggered 'refresh' from 1 events
Debug: /Stage[main]/Jenkins/Service[jenkins]: The container Class[Jenkins] will propagate my refresh event
Debug: Class[Jenkins]: The container Stage[main] will propagate my refresh event
Debug: Executing '/usr/bin/dpkg-query -W --showformat='${(Status)} ${Package} ${Version}\\n' docker-compose'
Debug: Executing '/usr/bin/dpkg-query -W --showformat='${(Status)} ${Package} ${Version}\\n' docker-compose'
Debug: Executing '/usr/bin/apt-get -q -y -o DPkg::Options::=--force-confold install docker-compose'
Notice: /Stage[main]/Docker_install/Package[docker-compose]/ensure: ensure changed 'purged' to 'present'
Debug: /Package[docker-compose]: The container Class[Docker_install] will propagate my refresh event
Debug: Class[Docker_install]: The container Stage[main] will propagate my refresh event
Debug: Finishing transaction 21314780
Debug: Storing state
Debug: Stored state in 0.03 seconds
Notice: Finished catalog run in 433.58 seconds
Debug: Executing '/etc/puppet/etckeeper-commit-post'
```

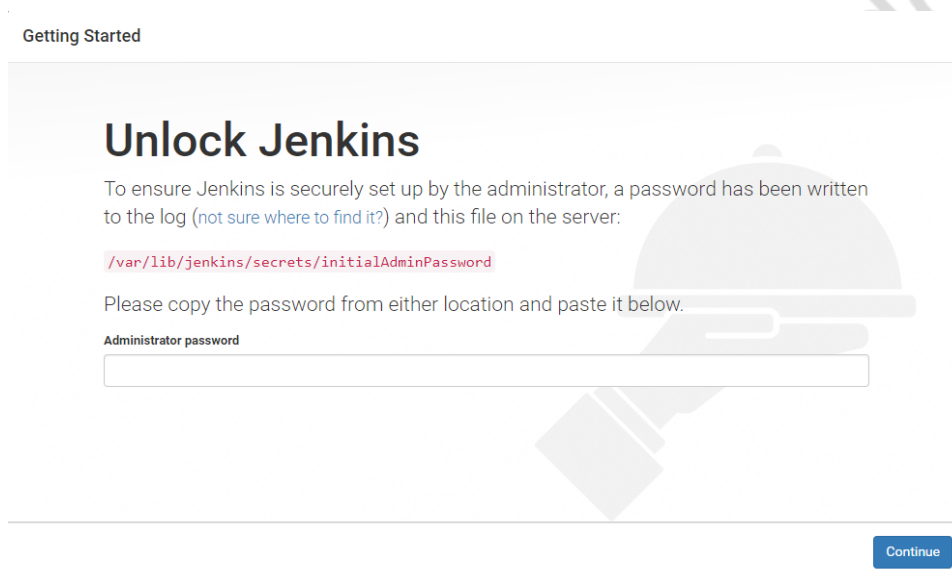
- c. Hay algunas situaciones aleatorias la ejecución falla. Si se visualiza un error similar al siguiente ejecutarlo de nuevo

```
Debug: Executing '/usr/bin/apt-get -q -y -o DPkg::Options::=--force-confold install jenkins'
Error: Execution of '/usr/bin/apt-get -q -y -o DPkg::Options::=--force-confold install jenkins' returned 100: E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?
Errors: /Stage[main]/Docker_install/Package[docker-compose]/ensure: change from purged to present failed: Execution of '/usr/bin/apt-get -q -y -o DPkg::Options::=--force-confold install jenkins' returned 100
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?
Notice: /Stage[main]/Jenkins/File[etc/default/jenkins]: Dependency Package[jenkins] has failures: true
Warning: /Stage[main]/Jenkins/File[etc/default/jenkins]: Skipping because of failed dependencies
Notice: /Stage[main]/Jenkins/File[etc/init.d/jenkins]: Dependency Package[jenkins] has failures: true
Warning: /Stage[main]/Jenkins/File[etc/init.d/jenkins]: Skipping because of failed dependencies
Notice: /Stage[main]/Jenkins/Service[jenkins]: Dependency Package[jenkins] has failures: true
Warning: /Stage[main]/Jenkins/Service[jenkins]: Skipping because of failed dependencies
Debug: Class[Jenkins]: The container Stage[main] will propagate my refresh event
Debug: Executing '/usr/bin/dpkg-query -W --showformat='${(Status)} ${Package} ${Version}\\n' docker-compose'
Debug: Executing '/usr/bin/dpkg-query -W --showformat='${(Status)} ${Package} ${Version}\\n' docker-compose'
Error: Execution of '/usr/bin/apt-get -q -y -o DPkg::Options::=--force-confold install docker-compose' returned 100: E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?
Errors: /Stage[main]/Docker_install/Package[docker-compose]/ensure: change from purged to present failed: Execution of '/usr/bin/apt-get -q -y -o DPkg::Options::=--force-confold install docker-compose' returned 100: E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?
Debug: Class[Docker_install]: The container Stage[main] will propagate my refresh event
Debug: Finishing transaction 25709000
Debug: Storing state
[Info: Creating state file /var/lib/puppet/state/state.yaml]
Debug: Stored state in 0.44 seconds
Notice: Finished catalog run in 315.90 seconds
Debug: Executing '/etc/puppet/etckeeper-commit-post'
Debug: Closing connection for https://utn-devops.localhost:8140
Debug: Creating new connection for https://utn-devops.localhost:8140
Debug: Starting connection for https://utn-devops.localhost:8140
Debug: Caching connection for https://utn-devops.localhost:8140
Debug: Closing connection for https://utn-devops.localhost:8140
```

Para visualizar la instalación de Puppet, abrir el archivo Vagrant.bootstrap.sh. En ese archivo se puede observar que se define la instalación de Puppet como maestro y agente en el mismo equipo. No es lo recomendado y se confeccionó de esta manera ya que estos son ejemplos para mostrar el uso y configuraciones básicas de distintas herramientas de software.

Haremos el inicio de una configuración básica de la aplicación Jenkins aplicando su clave a la interfaz web

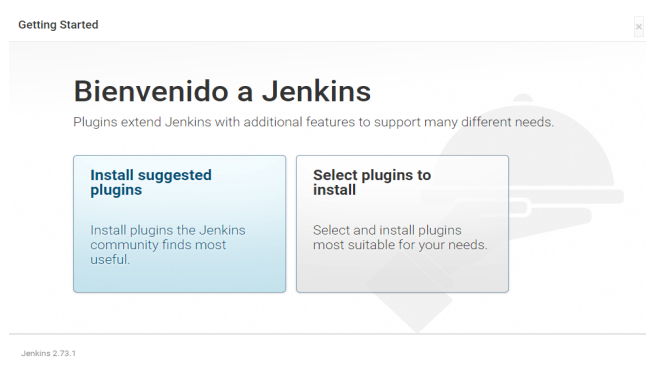
1. Ingresar en un navegador web a: <http://127.0.0.1:8082> para ver la aplicación de Jenkins
 - a. Al ser la primera que se ingresa nos pedirá una clave. Para esto volvemos a la terminal de comandos



2. `sudo cat /var/lib/jenkins/secrets/initialAdminPassword`
 - a. El resultado de este comando lo copiaremos y lo pondremos en el campo que nos pide la clave y presionaremos el botón “Continue”. Será similar al siguiente resaltado en color:

```
root@utn-devops:/home/ubuntu# cat /var/lib/jenkins/secrets/initialAdminPassword
8a6d317313114eb89315af80da87940f
root@utn-devops:/home/ubuntu#
```

3. Hacer clic en “Install suggested plugins”. Esto instalará varios paquetes que se utilizan comúnmente. La instalación puede demorar unos minutos





Getting Started

Getting Started

<input checked="" type="checkbox"/> Timestampers	<input checked="" type="checkbox"/> Credentials Binding Plugin	<input checked="" type="checkbox"/> Email Extension	<input checked="" type="checkbox"/> Build Timeout	<div>Email Extension</div> <div>** Docker Pipeline</div> <div>** Pipeline: Stage Tags Metadata</div> <div>** Pipeline: Declarative Agent API</div> <div>** Pipeline: Basic Steps</div> <div>** Pipeline: Declarative</div> <div>Pipeline</div> <div>Pipeline: Stage View</div> <div>** GitHub API</div> <div>Git</div>
<input checked="" type="checkbox"/> Pipeline	<input checked="" type="checkbox"/> Pipeline: Stage View Plugin	<input checked="" type="checkbox"/> Ant Plugin	<input checked="" type="checkbox"/> GitHub Branch Source	
<input type="checkbox"/> SSH Slaves	<input checked="" type="checkbox"/> Pipeline: GitHub Groovy Libraries	<input type="checkbox"/> Matrix Authorization Strategy	<input type="checkbox"/> Mailer Plugin	
<input checked="" type="checkbox"/> OWASP Markup Formatter Plugin	<input checked="" type="checkbox"/> Gradle Plugin	<input type="checkbox"/> PAM Authentication	<input checked="" type="checkbox"/> Git	
<input checked="" type="checkbox"/> Workspace Cleanup Plugin	<input type="checkbox"/> Subversion	<input type="checkbox"/> LDAP	<input checked="" type="checkbox"/> Folders Plugin	

** - required dependency

Jenkins 2.89.4

4. Crear el primer usuario administrador. Datos a respetar en el formulario:
 - a. Usuario: admin
 - b. Clave: utndevopsAl finalizar hacer clic "Save and Finish"

Getting Started

Create First Admin User

Usuario:	<input type="text" value="admin"/>
Contraseña:	<input type="password" value="*****"/>
Confirma la contraseña:	<input type="password" value="*****"/>
Nombre completo:	<input type="text" value="Admin"/>
Dirección de email:	<input type="text" value="test@test.com"/>

Jenkins 2.107.3

Continue as admin

Save and Finish

5. Hacer clic en "Save and Finish"

Centro de e-Learning SCEU UTN - BA.

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148

www.sceu.frba.utn.edu.ar/e-learning



Getting Started

Instance Configuration

Jenkins URL:

<http://127.0.0.1:8082>

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is not saved yet and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.121.3

[Not now](#)

[Save and Finish](#)

6. En la terminal de comandos de la máquina virtual ejecutar: exit

Con esto último ya dejamos instalado Jenkins. En la próxima unidad se realizará la configuración del mismo.

Si no se utilizará más la máquina virtual apagarla con el siguiente comando: vagrant halt