



ADMINISTRACIÓN DE INFRAESTRUCTURAS Y SISTEMAS INFORMÁTICOS (AISI)

Grado en Ingeniería Informática

Grado en Ingeniería Informática

Roberto R. Expósito (roberto.rey.exposito@udc.es)

Jorge Veiga (jorge.veiga@udc.es)

PRÁCTICA 1

Packer



Objetivo

3

- El propósito de esta práctica es aprender a utilizar las opciones básicas de **Packer**, una herramienta laC de código abierto que permite automatizar la creación de **imágenes máquina** idénticas desde ficheros de código fuente para múltiples entornos virtuales
 - Packer soporta múltiples plataformas:
 - AWS, Azure, CloudStack, GCE, VirtualBox, Docker, Vagrant, VMware...



<https://packer.io>

Build Automated Machine Images



Justificación de la práctica

4

- La realización de esta práctica se justificará de la siguiente forma:
 - Documento en formato PDF que incluya las capturas de pantalla indicadas para demostrar la realización del **ejercicio 5**
 - **Debes incluir capturas similares a las mostradas en las transparencias: 13, 14**



Para ayudar a identificarlas, estas transparencias incluyen esta imagen en la parte superior derecha



IMPORTANTE



- **ENTREGA** a través de Moodle: **17/02 (15:30)**
- **ES OBLIGATORIO** usar la nomenclatura que se propone para nombrar los recursos y debe apreciarse sin confusión en las capturas aportadas
 - **NO RECORTES** las capturas de pantalla, **debe verse toda la información** que sea relevante para comprobar el trabajo realizado
- **NO** seguir estas normas **IMPLICA UNA CALIFICACIÓN “C”** en esta práctica



¿Qué es una **imagen máquina**?

5

- Se puede definir como una unidad estática que contiene un SO y *software* pre-instalados y que se puede utilizar para crear rápidamente nuevos entornos virtuales (en local, en la nube, ...)
- Los **formatos de imagen máquina** normalmente cambian para cada entorno virtual y/o plataforma en la nube
 - AMI para el servicio *cloud* EC2 de AWS
 - OVF/OVA para VirtualBox (también soportados por otros hipervisores)
 - VMX para VMware
 - Box para Vagrant
- Packer permite **automatizar** el proceso de creación de imágenes máquina y describir su contenido usando ficheros de configuración como **plantillas** mediante lenguajes declarativos
 - Packer soporta JSON y HCL
 - <https://developer.hashicorp.com/packer/docs/templates>



Plantillas de Packer

6

- Ejemplo de plantilla en formato JSON

Genera una imagen máquina
(una AMI) para instanciar VMs
en la nube del proveedor
AWS usando el servicio EC2



```
{
  "builders": [
    {
      "type": "amazon-ebs",
      "access_key": "...",
      "secret_key": "...",
      "region": "us-east-1",
      "source_ami": "ami-fce3c696",
      "instance_type": "t2.micro",
      "ssh_username": "ubuntu",
      "ami_name": "packer {{timestamp}}"
    }
  ],
  "provisioners": [
    {
      "type": "shell",
      "script": "setup_things.sh"
    }
  ]
}
```

- Packer también proporciona su propio lenguaje de configuración para describir una imagen máquina: *Hashicorp Configuration Language* (HCL)
 - <https://developer.hashicorp.com/packer/guides/hcl>



Ejercicio 1: Instalación de Packer

7

- Prerrequisitos
 - VirtualBox \geq 7.0.6
 - Vagrant \geq 2.3.4
- Instala [Packer](#) en tu equipo y ejecuta un comando básico de prueba

```
[rober@oceania ~]$ packer version
Packer v1.8.5
[rober@oceania ~]$
```

- Échale un vistazo a la terminología básica de la herramienta
 - <https://developer.hashicorp.com/packer/docs/terminology>
 - **¿Qué debes aprender?**
 - **Conceptos:**
 - *Builders, Commands, Provisioners, Post-processors, Templates*



Ejercicio 2: Crea una plantilla de Packer

8

- Basándote en este [ejemplo](#) de la documentación y usando como plantilla el fichero *template.pkr.hcl* proporcionado en el [repositorio de la práctica](#)
 - Crea una plantilla HCL para generar un **Vagrant box** para **VirtualBox** basado en **Ubuntu** con el software [Docker Engine](#) pre-instalado en el box
- Para ello realiza la siguiente configuración en la plantilla HCL:
 - Utiliza el *builder* de Packer: **vagrant**
 - Utiliza el *provider* de Vagrant: **virtualbox**
 - Utiliza como Vagrant box de base: **ubuntu/focal64**
 - Usa la misma versión del box de la práctica previa (parámetro *box_version*)
 - Para instalar Docker Engine en el box usa el *provisioner* **shell** de Packer
 - Configura la ejecución del *script* de instalación de Docker proporcionado en el repositorio (*provisioning/install-docker-ubuntu.sh*)
 - Curiosear el *script* para ver cómo se instala Docker en Ubuntu
- Valida e inspecciona tu plantilla con los comandos *validate* e *inspect*
 - <https://www.packer.io/docs/commands/validate>
 - <https://www.packer.io/docs/commands/inspect>




Ejercicio 2: Crea una plantilla de Packer

- **¿Qué debes aprender?**
 - **Comandos:**
 - *validate, inspect, build*
 - **Opciones de configuración de los *builders*:**
 - *communicator*
 - **Opciones de configuración del *builder* Vagrant:**
 - *source_path, provider, box_version*
 - **Opciones de configuración de los *provisioners*:**
 - *only, timeout*
 - **Opciones de configuración del *provisioner* shell:**
 - *script, inline*



Ejercicio 3: Crea tu Vagrant *box*

10

- Usa tu plantilla para crear un Vagrant *box* usando el comando [*build*](#)
 - Tal y como se indica [*aquí*](#), es útil establecer la variable `PACKER_LOG=1` para incrementar el nivel de verbosidad que produce el comando *build*
 - La creación del *box* puede tardar 5-10 minutos, paciencia!
 - El fichero que representa el *box* (*package.box*) se creará en una subcarpeta con nombre ***output-aisi***
- Añade el *box* que acabas de crear con Packer a tu entorno local usando el comando *box add* de Vagrant (usa el parámetro `--name`)
 - **Debes nombrar el *box* siguiendo el formato: xxx-aisi2223/focal64** 
- Lista todos los *boxes* de tu *host* usando el comando *box list* de Vagrant
 - En este punto ya puedes eliminar la subcarpeta *output-aisi* si lo deseas


Vagrant *box* añadido y
correctamente nombrado



```
[rober@oceania ~]$ vagrant box list
boxomatic/alpine-3.16 (virtualbox, 20220821.0.1)
generic/rocky8        (virtualbox, 3.6.14)
generic/rocky8        (virtualbox, 4.1.0)
hashicorp/bionic64    (virtualbox, 1.0.282)
rre-aisi2223/focal64  (virtualbox, 0)
ubuntu/focal64        (virtualbox, 20220517.0.0)
ubuntu/focal64        (virtualbox, 20220905.0.0)
```



Ejercicio 4: Despliega tu Vagrant box

- Edita el *Vagrantfile* disponible en el repositorio para desplegar una VM
 - Configura el *hostname* de la VM
 - **Debes nombrar tu VM siguiendo el formato: xxx-aisi2223-docker** 
 - Utiliza el *box* que has creado previamente
 - Configura la redirección del puerto 8080 de tu *host* al puerto 80 de la VM
- Despliega una VM usando el comando *up* de Vagrant

```
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'rre-aisi2223/focal64'...
==> default: Matching MAC address for NAT networking...
==> default: Setting the name of the VM: AISI-P1-rre-aisi2223-docker
==> default: Fixed port collision for 22 => 2222. Now on port 2200.
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
default: Adapter 1: nat
==> default: Forwarding ports...
default: 80 (guest) => 8080 (host) (adapter 1)
default: 22 (guest) => 2200 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
default: SSH address: 127.0.0.1:2200
default: SSH username: vagrant
default: SSH auth method: private key
]
```



Ejercicio 4: Despliega tu Vagrant box

- Conéctate por *ssh* a la VM y comprueba la carpeta sincronizada

Hostname correctamente configurado →

```
vagrant@rre-aisi2223-docker ~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            975M   0    975M   0% /dev
tmpfs           199M  976K   198M   1% /run
/dev/sda1       39G   2.3G   37G    6% /
tmpfs           992M   0    992M   0% /dev/shm
tmpfs           5.0M   0    5.0M   0% /run/lock
tmpfs           992M   0    992M   0% /sys/fs/cgroup
/dev/loop0      64M    64M    0 100% /snap/core20/1623
/dev/loop1      47M    47M    0 100% /snap/snapd/16292
/dev/loop2      68M    68M    0 100% /snap/lxd/22753
vagrant         1.8T  477G   1.4T   26% /vagrant
tmpfs           199M   0    199M   0% /run/user/1000
vagrant@rre-aisi2223-docker:~$ ls /vagrant/
Vagrantfile  html  provisioning  template.pkr.hcl
vagrant@rre-aisi2223-docker:~$
```

Comprobamos la carpeta sincronizada que se configura por defecto →

- Ejecuta un contenedor Docker de prueba para comprobar la instalación
- *docker run --rm hello-world*

```
vagrant@rre-aisi2223-docker:~$ docker run --rm hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:aa0cc8055b82dc2509bed2e19b275c8f463506616377219d9642221ab53cf9fe
Status: Downloaded newer image for hello-world:latest
```

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub. (amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.




No te preocupes por la sintaxis del comando *docker* de esta transparencia y las siguientes. Aprenderás a utilizar Docker en la próxima práctica



Ejercicio 5: Servidor web con Docker



- Personaliza la página web que deberás mostrar en el servidor web Nginx que desplegaremos a continuación en un contenedor Docker
 - Abre el fichero *index.html* con un editor de texto en tu equipo para incluir tu nombre y apellidos
 - **Debes modificar únicamente la línea 19** 
- Ejecuta el contenedor Docker que despliega el servidor web Nginx y comprueba su estado (*docker ps*)
 - `docker run --rm -d --name nginx-aisi -p 80:80 -v /vagrant/html:/usr/share/nginx/html nginx`

```
vagrant@rre-aisi2223-docker:~$ docker run --rm -d --name nginx-aisi -p 80:80 -v /vagrant/html:/usr/share/nginx/html nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
8740c948ffd4: Pull complete
d2c0556a17c5: Pull complete
c8b9881f2c6a: Pull complete
693c3ffa8f43: Pull complete
8316c5e80e6d: Pull complete
b2fe3577faa4: Pull complete
Digest: sha256:b8f2383a95879e1ae064940d9a200f67a6c79e710ed82ac42263397367e7cc4e
Status: Downloaded newer image for nginx:latest
64092aade8c5312bd21560020ea0384f2de554990890157588fdd2423b7ccaf6
vagrant@rre-aisi2223-docker:~$
vagrant@rre-aisi2223-docker:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
64092aade8c5	nginx	"/docker-entrypoint..."	8 seconds ago	Up 5 seconds	0.0.0.0:80->80/tcp, :::80->80/tcp	nginx-aisi

Contenedor
en ejecución

Nombre del
contenedor



Ejercicio 5: Servidor web con Docker



- Accede al servidor web **desde la VM** usando el comando *curl*

```
vagrant@rre-aisi2223-docker:~$ curl localhost
<html>
<head>
  <meta charset= "utf-8">
  <title>GEI AISI: Test Page</title>
  <script type="text/javascript">
    function getURL() {
      document.write("URL: " + window.location.href);
    }
    function getTIME() {
      document.getElementById("current_date").innerHTML = Date();
    }
  </script>
</head>
<body>
  <div style="width:600px;height:200px;border:2px solid #000;text-align: center;">
    <strong><br>
    <u>GEI AISI: 2022/2023</u>
    <p><u>Nginx Web Server (Docker)</u></p>
    <p>Página web de Roberto Rey Expósito</p>
    <p><script>getURL();</script></p>
    <p><div id="current_date"><script>getTIME();</script></p>
  </strong>
</div>
</body>
</html>
vagrant@rre-aisi2223-docker:~$
```

- Accede al servidor web **desde el navegador de tu host**



¿Por qué debes acceder al puerto 8080 y no al 80? ¿Funciona el acceso si accedes desde la VM con *curl* al puerto 8080?



Ejercicio 5: Servidor web con Docker

- Obtén los *logs* del contenedor
 - *docker logs nginx-aisi*

```
vagrant@re-aisi2223-docker:~$ docker logs rre-aisi2223-nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
2023/01/13 10:37:30 [notice] 1#1: using the "epoll" event method
2023/01/13 10:37:30 [notice] 1#1: nginx/1.23.3
2023/01/13 10:37:30 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/01/13 10:37:30 [notice] 1#1: OS: Linux 5.4.0-125-generic
2023/01/13 10:37:30 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/01/13 10:37:30 [notice] 1#1: start worker processes
2023/01/13 10:37:30 [notice] 1#1: start worker process 28
2023/01/13 10:37:30 [notice] 1#1: start worker process 29
/docker-entrypoint.sh: Configuration complete; ready for start up
172.17.0.1 - - [13/Jan/2023:10:42:22 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/7.68.0" "-"
2023/01/13 10:43:23 [error] 29#29: *2 open() "/usr/share/nginx/html/favicon.ico" failed (2: No s
:8080", referer: "http://localhost:8080/"
10.0.2.2 - - [13/Jan/2023:10:43:23 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://localhost:
10.0.2.2 - - [13/Jan/2023:10:43:28 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0 (X11; Linux
10.0.2.2 - - [13/Jan/2023:10:43:30 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (X11; Linux x8
10.0.2.2 - - [13/Jan/2023:10:43:30 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (X11; Linux x8
10.0.2.2 - - [13/Jan/2023:10:43:31 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (X11; Linux x8
172.17.0.1 - - [13/Jan/2023:10:43:34 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/7.68.0" "-"
vagrant@re-aisi2223-docker:~$
```

- Por último, detén el contenedor
 - *docker stop nginx-aisi*



Referencias

16

- Documentación sobre plantillas HCL
 - https://developer.hashicorp.com/packer/docs/templates/hcl_templates
- *Builder Vagrant*
 - <https://developer.hashicorp.com/packer/plugins/builders/vagrant>
- *Provisioner shell*
 - <https://developer.hashicorp.com/packer/docs/provisioners/shell>
- Otra documentación interesante
 - <https://developer.hashicorp.com/packer/docs/commands>
 - <https://developer.hashicorp.com/packer/docs/builders>
 - <https://developer.hashicorp.com/packer/docs/communicators>
 - <https://developer.hashicorp.com/packer/docs/provisioners>