



ADMINISTRACIÓN DE INFRAESTRUCTURAS Y SISTEMAS INFORMÁTICOS (AISI)

Grado en Ingeniería Informática

Grado en Ingeniería Informática

Roberto R. Expósito (roberto.rey.exposito@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, GCE, VirtualBox, Docker, Vagrant, VMware...



Build Automated Machine Images

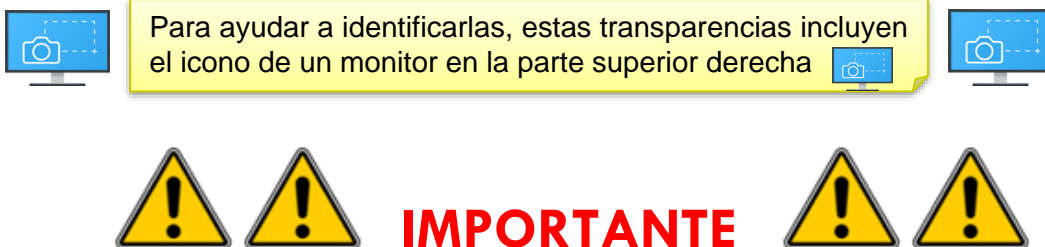
<https://www.packer.io>



Justificación de la práctica

4

- La **entrega** de la práctica consistirá en un **breve documento** en formato **PDF** que incluya las **todas capturas de pantalla** mostradas en las transparencias:
 - 12, 16, 17**



- En ocasiones, durante la práctica, se pide crear recursos con un nombre que empieza por un **prefijo** que contiene información del estudiante y del curso
- ES OBLIGATORIO** usar la siguiente nomenclatura para nombrar los recursos:
<iniciales del nombre y apellidos><curso>-<nombre del recurso>
 - Ejemplo: El alumno Roberto Rey Expósito, que hace la práctica en el curso 2024/2025, utilizará el siguiente prefijo: **rre2425**
- 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 la práctica



¿Qué es una imagen máquina?

5

- Se puede definir como una unidad estática que contiene un SO y *software* pre-instalado que se utiliza 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 Hashicorp Configuration Language (HCL)**
 - <https://developer.hashicorp.com/packer/docs/templates>



Plantillas de Packer

6

- Ejemplo de plantilla usando el lenguaje HCL

Genera una imagen máquina (una AMI en este caso) para crear VM en la nube del proveedor AWS usando el servicio EC2

```
source "amazon-ecs" "ubuntu" {
  ami_name      = "learn-packer-linux-aws"
  instance_type = "t2.micro"
  region        = "us-west-2"
  source_ami_filter {
    filters = {
      name      = "ubuntu/images/*ubuntu-jammy-22.04-amd64-server-*"
      root-device-type = "ebs"
      virtualization-type = "hvm"
    }
    most_recent = true
    owners      = ["099720109477"]
  }
  ssh_username = "ubuntu"
}

build {
  name      = "learn-packer"
  sources = [
    "source.amazon-ecs.ubuntu"
  ]
}
```



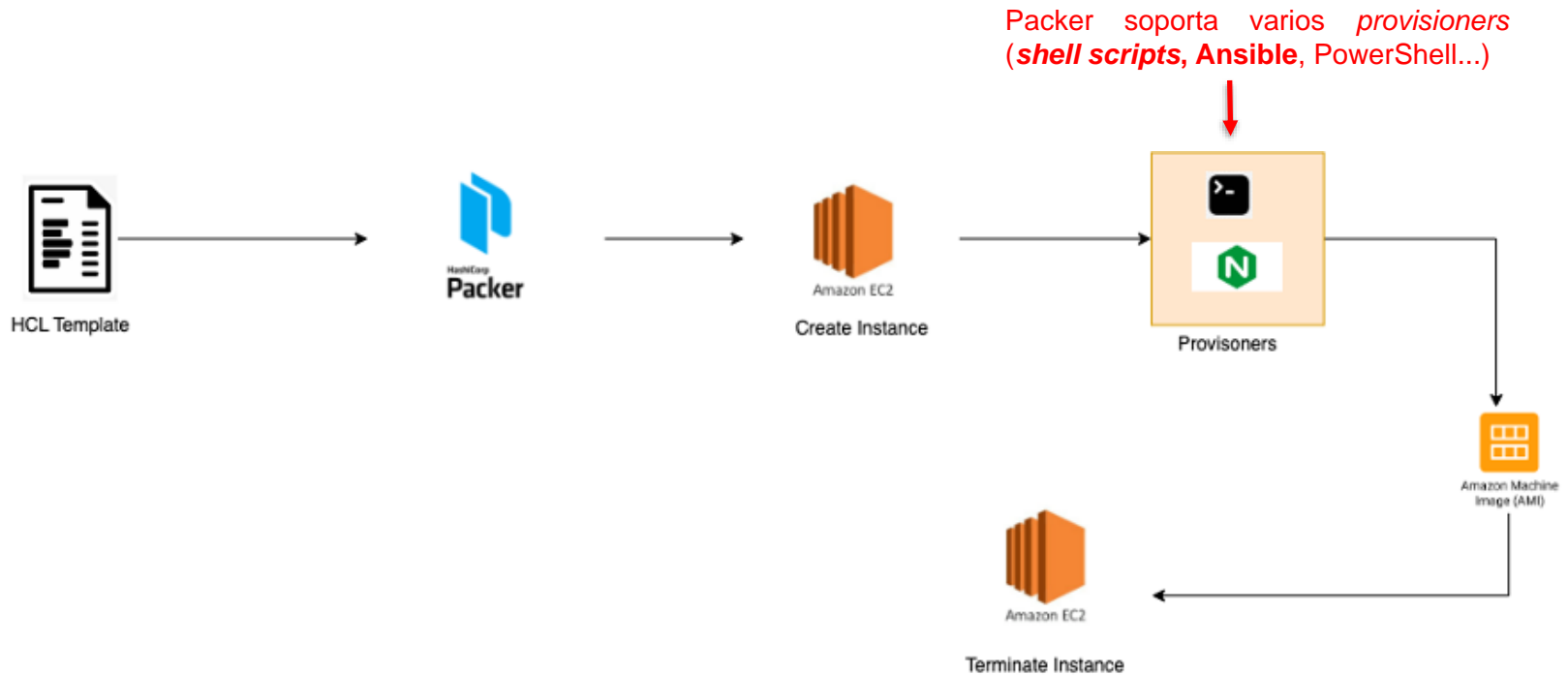
El soporte para plantillas en [formato JSON](#) se considera obsoleto y pronto dejará de recibir nuevas *features*



Plantillas de Packer

7

- Flujo de trabajo de creación de una AMI para AWS EC2





Ejercicio 1: Instalación de Packer

8

- Prerrequisitos de la práctica 0
 - VirtualBox \geq 7.1.6
 - Vagrant \geq 2.4.3
- Instala Packer (versión \geq 1.11.2) y ejecuta un comando de prueba

```
rober@mordor:~$ packer version
Packer v1.11.2
rober@mordor:~$
```

- Instala el *plugin* de Packer que integra el soporte para Vagrant
 - <https://developer.hashicorp.com/packer/integrations/hashicorp/vagrant>

```
rober@mordor:~$ packer plugins install github.com/hashicorp/vagrant
Installed plugin github.com/hashicorp/vagrant v1.1.5 in "/home/rober/.config/packer/plugins_linux_amd64"
rober@mordor:~$
```




Ejercicio 2: Primeros pasos con Packer

- Explora los comandos de CLI más relevantes de Packer
 - <https://developer.hashicorp.com/packer/docs/commands>
- Échale un vistazo a la terminología básica de la herramienta
 - <https://developer.hashicorp.com/packer/docs/terminology>
- **¿Qué debes aprender?**
 - **Comandos:**
 - *init, plugins, build, fix, validate, inspect*
 - **Conceptos:**
 - *Builders, Commands, Provisioners, Post-processors, Templates*



Ejercicio 3: Crea una plantilla HCL

10

- Basándote en este [ejemplo](#) de la documentación de Packer 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 Jammy Jellyfish** (v22.04 LTS) con el software **Docker Engine** pre-instalado
- 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/jammy64](#)
 - Usa la última versión disponible del box (parámetro *box_version* de la plantilla)
 - 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 de la práctica (*provisioning/install-docker-ubuntu.sh*)
 - Curiosear el *script* para ver cómo instalar Docker en Ubuntu (basado en esta [guía](#))
- Valida e inspecciona tu plantilla con los comandos *validate* e *inspect*
 - <https://developer.hashicorp.com/packer/docs/commands/validate>
 - <https://developer.hashicorp.com/packer/docs/commands/inspect>



Ejercicio 3: Crea una plantilla HCL

11

- **¿Qué debes aprender?**
 - **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 4: Crea tu Vagrant *box*



12

- Usa tu plantilla para crear un Vagrant *box* usando el comando [*build*](#)
 - Tal y como se indica [*aquí*](#), es muy útil establecer la variable `PACKER_LOG=1` para incrementar el nivel de verbosidad de los comandos de Packer
 - El proceso de creación del *box* puede tardar 5-10 minutos, tened paciencia!
 - El fichero del nuevo *box* (***package.box***) se creará en la subcarpeta ***output-aisi***
- Añade el *box* que acabas de crear a tu entorno local de Vagrant usando el comando `vagrant box add` (usa el parámetro `--name`)
 - **Debes nombrar tu *box* siguiendo el formato: X/jammy64**
- Lista todos los *boxes* usando el comando `box list` de Vagrant



Recuerda que debes sustituir la "X" por tu **prefijo**

Vagrant *box* añadido y correctamente nombrado →


```
rober@mordor:~$ vagrant box list
boxomatic/alpine-3.16      (virtualbox, 20240704.0.1, (amd64))
debian/bookworm64         (virtualbox, 12.20241217.1, (amd64))
generic/rocky8             (libvirt, 4.3.12, (amd64))
generic/rocky8            (virtualbox, 4.3.12, (amd64))
rre2425/jammy64           (virtualbox, 0)
rrey/jammy64-gui          (virtualbox, 20240306, (amd64))
rrey/omv7                 (virtualbox, 1.0, (amd64))
rrey/rocky8-lustre-clients (virtualbox, 1.0, (amd64))
rrey/rocky8-lustre-servers (libvirt, 1.0, (amd64))
rrey/rocky8-lustre-servers (virtualbox, 1.0, (amd64))
ubuntu/jammy64            (virtualbox, 20240207.0.0)
rober@mordor:~$
```



Una vez añadido el *box* a Vagrant, si quieres ya puedes eliminar la carpeta *output-aisi*



Ejercicio 5: Despliega tu Vagrant box

- Edita el *Vagrantfile* disponible en el repositorio de la práctica para desplegar una VM usando tu *box*
 - Configura el *hostname* de la VM
 - **Debes nombrar tu VM siguiendo el formato: X-nginx** 
 - Utiliza el *box* que has creado previamente
 - Configura la redirección del puerto **9090** de tu *host* al puerto **80** de la VM
- Despliega la VM usando Vagrant

```
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'rre2425/jammy64'...
==> default: Matching MAC address for NAT networking...
==> default: Setting the name of the VM: AISI-P1-rre2425-nginx
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
default: Adapter 1: nat
==> default: Forwarding ports...
default: 80 (guest) => 9090 (host) (adapter 1)
default: 22 (guest) => 2222 (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: 127.0.0.1:2222
default: SSH username: vagrant
default: SSH auth method: private key
```



Ejercicio 5: Despliega tu Vagrant box

- Conéctate por *ssh* a la VM y comprueba el *hostname*:

Hostname correctamente
configurado →

```
vagrant@rre2425-nginx:~$ df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
tmpfs	197M	968K	196M	1%	/run
/dev/sda1	39G	2.2G	37G	6%	/
tmpfs	982M	0	982M	0%	/dev/shm
tmpfs	5.0M	0	5.0M	0%	/run/lock
vagrant	183G	51G	132G	28%	/vagrant
tmpfs	197M	4.0K	197M	1%	/run/user/1000

Comprobamos la carpeta
sincronizada que se
configura por defecto →

```
vagrant@rre2425-nginx:~$
```

```
vagrant@rre2425-nginx:~$ ls /vagrant/
```


```
Vagrantfile  html  output-aisi  provisioning  template.pkr.hcl
```

```
vagrant@rre2425-nginx:~$
```

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

```
vagrant@rre2425-nginx:~$ docker run --rm hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:1b7a37f2a0e26e55ba2916e0c53bfb60d9bd43e390e31aacd25cb3581
Status: Downloaded newer image for hello-world:latest
```

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




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



Ejercicio 6: Servidor web con Docker



15

- Personaliza la página que deberás mostrar en el servidor web Nginx que desplegaremos a continuación en un contenedor Docker
 - En el directorio *html*, abre el fichero *index.html* con un **editor de texto en tu equipo** para incluir tu nombre y apellidos
 - **Debes modificar únicamente la variable *name* (línea 7)** 
- Crea un contenedor Docker que ejecuta un servidor web Nginx y comprueba su estado (*docker ps*)
 - `docker run --rm -d --name nginx -p 80:80 -v /vagrant/html:/usr/share/nginx/html nginx`

```
vagrant@rre2425-nginx:~$ docker run --rm -d --name nginx -p 80:80 -v /vagrant/html:/usr/share/nginx/html nginx
```

```
Unable to find image 'nginx:latest' locally
```

```
latest: Pulling from library/nginx
```

```
af302e5c37e9: Pull complete
```

```
207b812743af: Pull complete
```

```
841e383b441e: Pull complete
```

```
0256c04a8d84: Pull complete
```

```
38e992d287c5: Pull complete
```

```
9e9aab598f58: Pull complete
```

```
4de87b37f4ad: Pull complete
```

```
Digest: sha256:0a399eb16751829e1af26fea27b20c3ec28d7ab1fb72182879dcae1cca21206a
```

```
Status: Downloaded newer image for nginx:latest
```

```
a2068094713671119b4c96fb52dfc2c73e4709f7fa4485a0cf1768874ef53726
```

```
vagrant@rre2425-nginx:~$
```

```
vagrant@rre2425-nginx:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
a20680947136	nginx	"/docker-entrypoint..."	4 seconds ago	Up 3 seconds	0.0.0.0:80->80/tcp, :::80->80/tcp	nginx

```
vagrant@rre2425-nginx:~$
```

Contenedor
en ejecución



Ejercicio 6: Servidor web con Docker



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

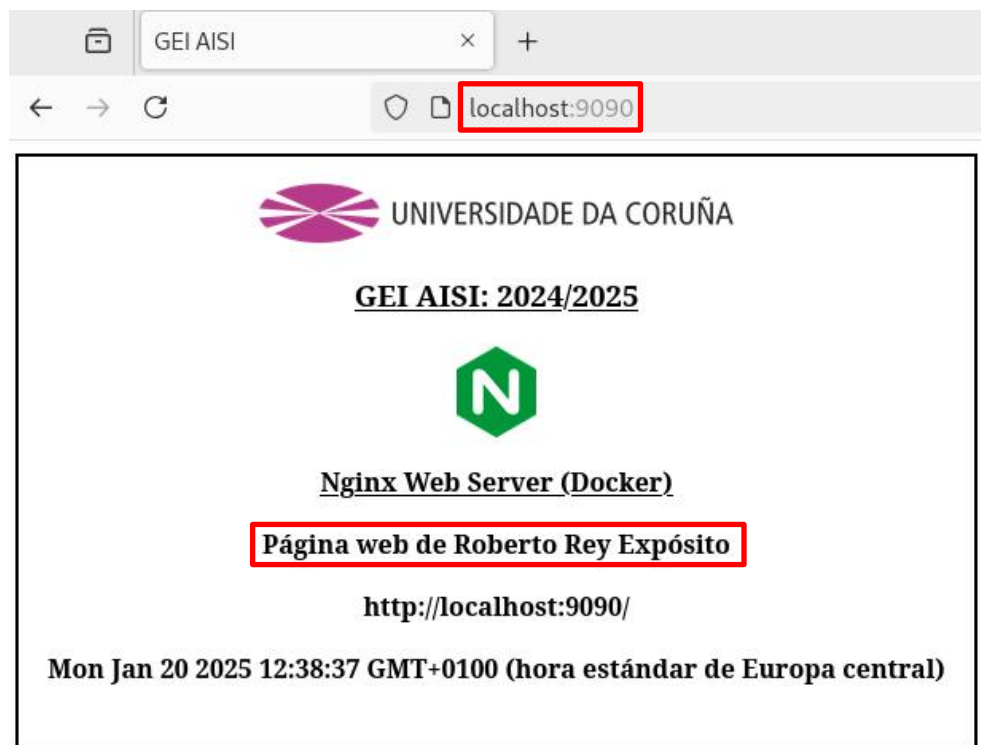
```
vagrant@rre2425-nginx:~$ curl http://localhost
<html>
<head>
  <meta charset="UTF-8">
  <title>GEI AISI</title>
  <script type="text/javascript">
    function getName() {
      var name = "Roberto Rey Expósito";
      document.getElementById("myName").innerHTML = name;
    }
    function getURL() {
      document.write(window.location.href);
    }
    function getTime() {
      document.getElementById("current_date").innerHTML = Date();
    }
  </script>
</head>
<body onload="getName()">
  <div style="width:600px;height:370px;border:2px solid #000;text-align: center;">
    <strong>
      <p></p>
      <h3><u>GEI AISI: 2024/2025</u></h3>
      <p>
      <p><u>Nginx Web Server (Docker)</u></p>
      <p>Página web de <span id="myName"></span></p>
      <p><script>getURL();</script></p>
      <p><div id="current_date"><script>getTime();</script></p>
    </strong>
  </div>
</body>
</html>
vagrant@rre2425-nginx:~$
```




Ejercicio 6: Servidor web con Docker



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



¿Por qué debes acceder al puerto 9090 y no al 80? ¿Funciona el acceso si accedes desde la VM con *curl* al puerto 9090?



Ejercicio 6: Servidor web con Docker

- Obtén los *logs* del contenedor

- *docker logs nginx*

```
vagrant@rre2425-nginx:~$ docker logs 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: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/01/20 11:41:27 [notice] 1#1: using the "epoll" event method
2025/01/20 11:41:27 [notice] 1#1: nginx/1.27.3
2025/01/20 11:41:27 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/01/20 11:41:27 [notice] 1#1: OS: Linux 5.15.0-100-generic
2025/01/20 11:41:27 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/01/20 11:41:27 [notice] 1#1: start worker processes
2025/01/20 11:41:27 [notice] 1#1: start worker process 30
2025/01/20 11:41:27 [notice] 1#1: start worker process 31
172.17.0.1 - - [20/Jan/2025:11:41:40 +0000] "GET / HTTP/1.1" 200 1008 "-" "curl/7.81.0" "-"
10.0.2.2 - - [20/Jan/2025:11:41:45 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0)"
vagrant@rre2425-nginx:~$
```

Peticiones GET recibidas
por Nginx

- Por último, detén el contenedor

- *docker stop nginx*



Referencias

19

- 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>
- Instalación de Docker Engine en Ubuntu
 - <https://docs.docker.com/engine/install/ubuntu/>