



# ADMINISTRACIÓN DE INFRAESTRUCTURAS Y SISTEMAS INFORMÁTICOS (AISI)

Grado en Ingeniería Informática

Grado en Ingeniería Informática

Roberto R. Expósito ([roberto.rey.exposito@udc.es](mailto: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 realización de esta práctica se justificará de la siguiente forma:
  - Documento en formato PDF que incluya **todas las capturas de pantalla** mostradas en las transparencias: **12, 16, 17**



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



**IMPORTANTE**



- **ENTREGA** a través de Moodle: **19/02 (15:00)**
- **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-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 HCL

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



```
source "amazon-ebs" "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-ebs.ubuntu"
  ]
}
```



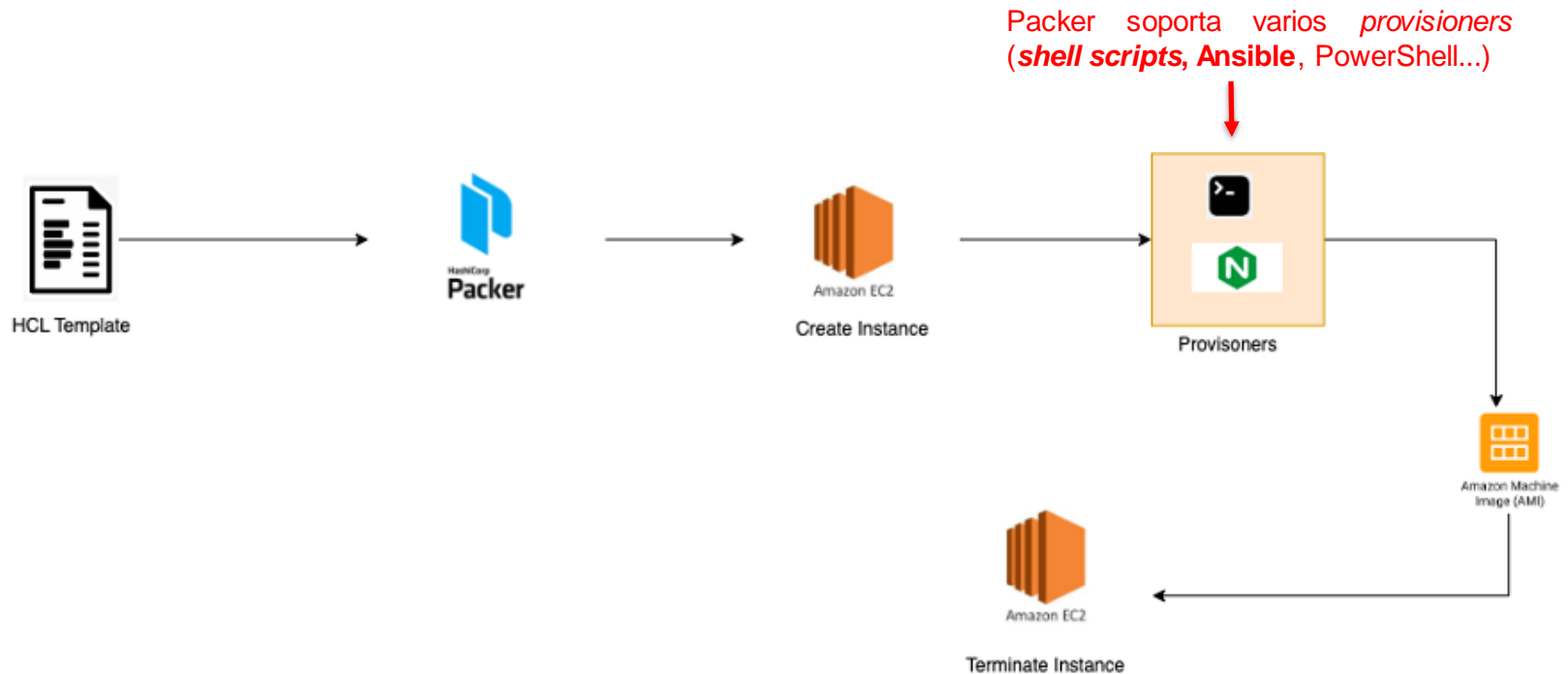
El soporte para plantillas en formato JSON se considera obsoleto y pronto dejará de recibir nuevas *features*



# Plantillas de Packer

7

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





# Ejercicio 1: Instalación de Packer

8

- Prerrequisitos de la práctica 0
  - VirtualBox  $\geq$  7.0.14
  - Vagrant  $\geq$  2.4.1
- Instala [Packer](#) (versión  $\geq$  1.10.1) y ejecuta un comando de prueba

```
[rober@oceania ~]$ packer version
Packer v1.10.1
[rober@oceania ~]$
```

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

```
[rober@oceania ~]$ packer plugins install github.com/hashicorp/vagrant
Installed plugin github.com/hashicorp/vagrant v1.1.2 in "/home/rober/.config/packer/plugins"
[rober@oceania ~]$
```





## 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, 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



- 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 de los comandos de Packer
    - La creación del box puede tardar 5-10 minutos, tened paciencia!
  - El fichero del box (`package.box`) se creará en la subcarpeta **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: xxx2324/jammy64** 
- Lista todos los boxes usando el comando `box list` de Vagrant

```
[rober@oceania ~]$ vagrant box list
bento/ubuntu-20.04 (virtualbox, 202309.09.0)
boxomatic/rocky-9 (virtualbox, 20230727.0.1)
debian/bookworm64 (virtualbox, 12.20230723.1)
debian/bookworm64 (virtualbox, 12.20231211.1)
debian/bullseye64 (virtualbox, 11.20221219.1)
generic/rocky8 (virtualbox, 3.6.14)
generic/rocky8 (virtualbox, 4.1.20)
generic/rocky8 (virtualbox, 4.2.16)
generic/rocky8 (virtualbox, 4.3.6, (amd64))
generic/rocky8 (virtualbox, 4.3.8, (amd64))
rre2324/jammy64 (virtualbox, 0)
ubuntu/jammy64 (virtualbox, 20230110.0.0)
[rober@oceania ~]$
```


Vagrant box añadido y  
correctamente nombrado →



Una vez añadido el box, 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: xxx2324-docker** 
  - 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 'rre2324/jammy64'...
==> default: Matching MAC address for NAT networking...
==> default: Setting the name of the VM: AISI-P1-rre2324-docker
==> default: Fixed port collision for 22 => 2222. Now on port 2201.
==> 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) => 2201 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
```



# Ejercicio 5: Despliega tu Vagrant box

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

Hostname correctamente configurado → `vagrant@rre2324-docker:~$ df -h`

Filesystem	Size	Used	Avail	Use%	Mounted on
tmpfs	198M	960K	197M	1%	/run
/dev/sda1	39G	2.1G	37G	6%	/
tmpfs	988M	0	988M	0%	/dev/shm
tmpfs	5.0M	0	5.0M	0%	/run/lock
vagrant	1.8T	332G	1.5T	19%	/vagrant
tmpfs	198M	4.0K	198M	1%	/run/user/1000


Comprobamos la carpeta sincronizada que se configura por defecto → `vagrant@rre2324-docker:~$ ls /vagrant/`

```
Vagrantfile  html  output-aisi  provisioning  template.pkr.hcl
vagrant@rre2324-docker:~$
```

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

```
vagrant@rre2324-docker:~$ docker run --rm hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:4bd78111b6914a99dbc560e6a20eab57ff6655aea4a80c50b0c5491968
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
  - 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 (*docker run*) que ejecuta el 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@re2324-docker:~$ 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
```

```
2f44b7a888fa: Pull complete
```

```
8b7dd3ed1dc3: Pull complete
```

```
35497dd96569: Pull complete
```

```
36664b6ce66b: Pull complete
```

```
2d455521f76c: Pull complete
```

```
dc9c4fdb83d6: Pull complete
```

```
8056d2bcf3b6: Pull complete
```

```
Digest: sha256:4c0fdaa8b6341bfdeca5f18f7837462c80cff90527ee35ef185571e1c327beac
```

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

```
e7d98383602b868b6c136a5f75c5c9984fb39f8da15f11d184ed43774cf3988a
```

```
vagrant@re2324-docker:~$
```

```
vagrant@re2324-docker:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
e7d98383602b	nginx	"/docker-entrypoint..."	6 seconds ago	Up 4 seconds	0.0.0.0:80->80/tcp, :::80->80/tcp	nginx

```
vagrant@re2324-docker:~$
```

Contenedor  
en ejecución



# Ejercicio 6: Servidor web con Docker



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

```
vagrant@rre2324-docker:~$ 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:340px;border:2px solid #000;text-align: center;">
    <strong>
      <p></p>
      <h3><u>GEI AISI: 2023/2024</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>
```

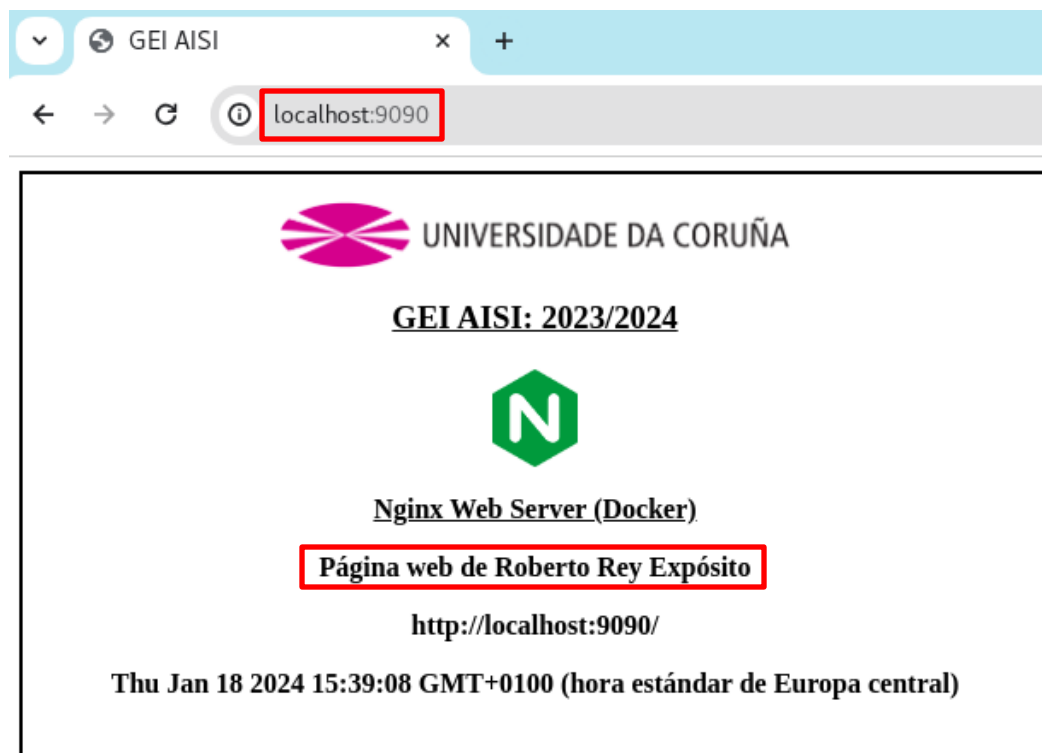




# Ejercicio 6: Servidor web con Docker



- 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@ire2324-docker:~$ 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
2024/01/18 10:35:19 [notice] 1#1: using the "epoll" event method
2024/01/18 10:35:19 [notice] 1#1: nginx/1.25.3
2024/01/18 10:35:19 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/01/18 10:35:19 [notice] 1#1: OS: Linux 5.15.0-57-generic
2024/01/18 10:35:19 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/01/18 10:35:19 [notice] 1#1: start worker processes
2024/01/18 10:35:19 [notice] 1#1: start worker process 29
2024/01/18 10:35:19 [notice] 1#1: start worker process 30
172.17.0.1 - - [18/Jan/2024:10:35:25 +0000] "GET / HTTP/1.1" 200 1008 "-" "curl/7.81.0" "-"
172.17.0.1 - - [18/Jan/2024:10:35:35 +0000] "GET / HTTP/1.1" 200 1008 "-" "curl/7.81.0" "-"
10.0.2.2 - - [18/Jan/2024:10:36:28 +0000] "GET / HTTP/1.1" 200 1008 "-" "curl/7.61.1" "-"
vagrant@ire2324-docker:~$
```

← 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](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/>