



# Creando y automatizando tu propio entorno de Directorio Activo



**Jorge Escabias**

**Analista de ciberseguridad en Zerolynx**



**@MrSquid25**

**jescabias@zerolynx.com**

# Índice

## 1. Introducción

- ¿De qué vamos a hablar?
- Paradigma
- ¿Y si los formo...?
- Solución

## 2. Objetivo

- Pasos a seguir

## 3. Creación del entorno

- Herramientas necesarias
  - Hardware
  - Software
- Elección de despliegue
  - Creación manual
  - Packer
  - Vagrant
  - Packer + Vagrant

## 4. Automatización del entorno

- Terraform y Ansible
  - Terraform
  - Ansible

## 5. Despliegue final

- Resumiendo...
- Arquitectura

## 6. Demo

## 7. Conclusiones

- Ventajas vs Desventajas
- Posibles mejoras

## 8. Referencias

# Introducción



# Introducción

*¿De qué vamos a hablar?*



Fuente: <http://download.microsoft.com/documents/uk/pcit/1-PCIT-Empowering%20People%20centric%20IT.pptx>

Database Info	Node Info	Analysis
Sessions		19895
Relationships		5168619
ACLs		3118877
Azure Relationships		0
ON-PREM OBJECTS		
Users		150445
Groups		75786
Computers		63101
OUS		578
GPOs		2185
Domains		11




DETECTIONLAB

# Introducción

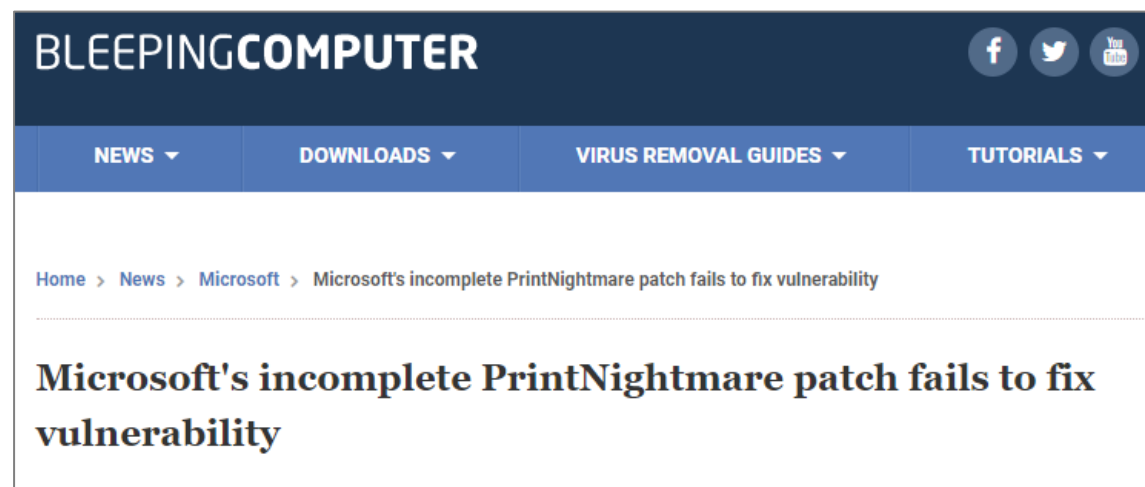
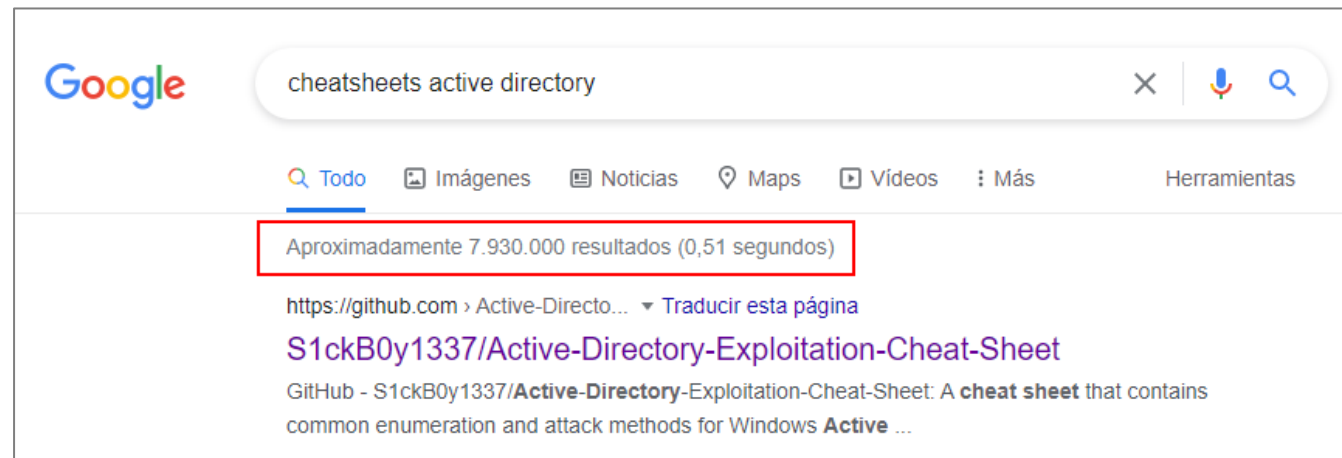
## Paradigma

### Security assessment: Unsecure Kerberos delegation

06/30/2021 • 2 minutes to read • 

#### What is Kerberos delegation?

Kerberos delegation is a delegation setting that allows applications to request end-user access credentials to access resources on behalf of the originating user.





# Introducción

¿Y si los formo...?

**PENTESTER ACADEMY**

**Start Your Red Team Journey  
Get Certified**



**CRTP**      **CRTE**      **CARTP**      **PACES**



**SEC564**

**Red Team Exercises and  
Adversary Emulation**

Jorge Orchilles



**PENTEST HACKFEST** November 16–21  
Live Online 

**SANS**

**NEW COURSE**

**Evasion Techniques and  
Breaching Defenses**

**OSEP**



**OFFENSIVE security**



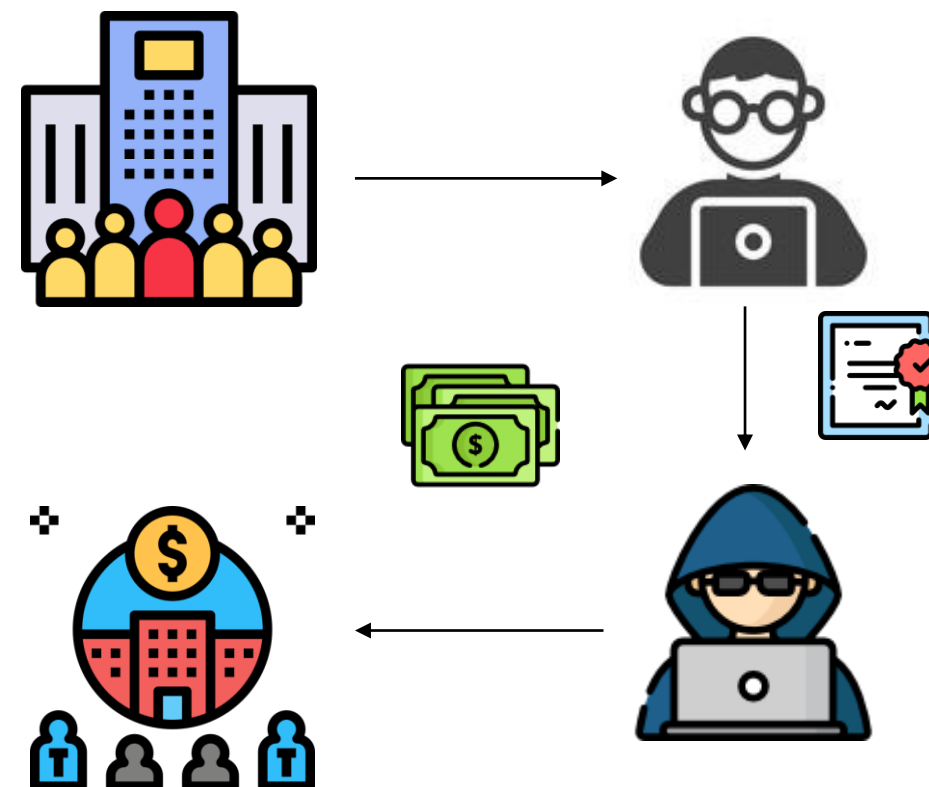
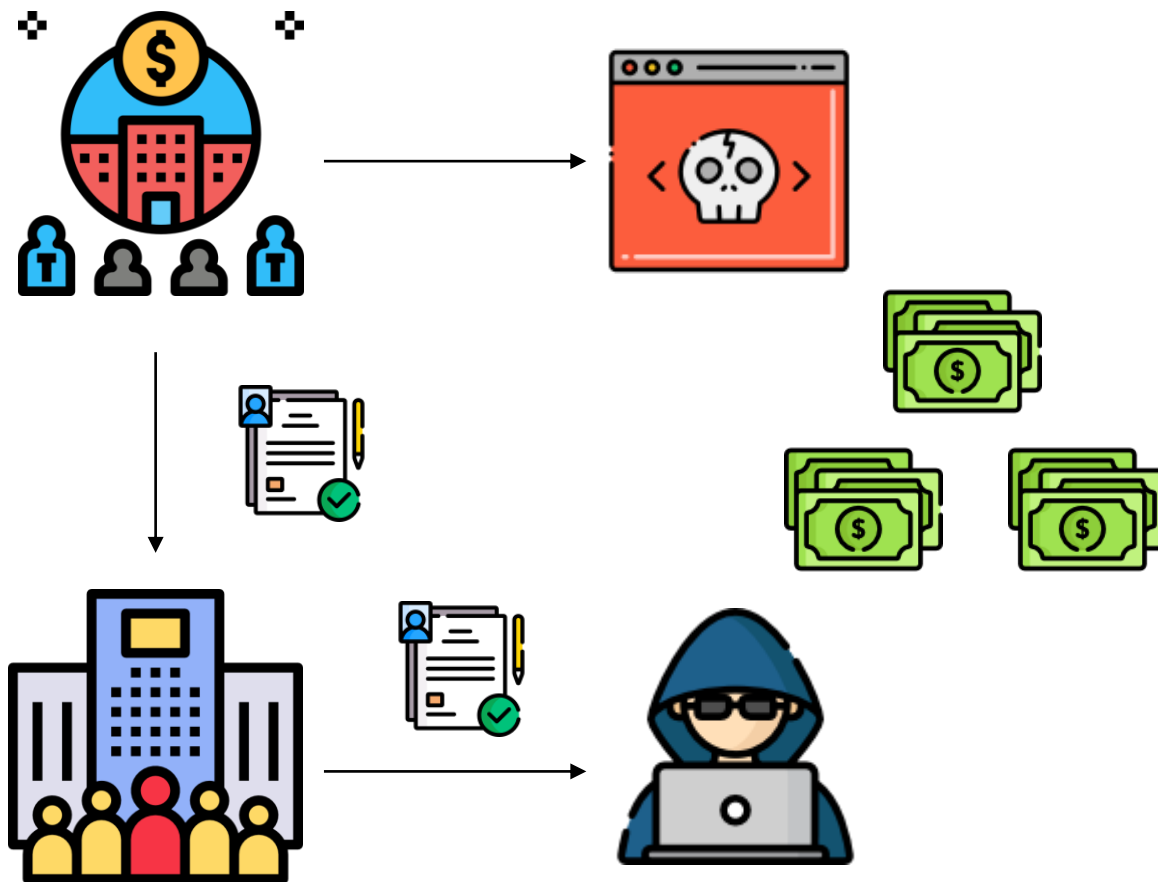
**SPECTEROPS**

Adversary Tactics:  
**Red Team  
Operations**



# Introducción

## *Solución*





# Objetivo



# Objetivo

1. Crear un entorno de pruebas donde poner en práctica el abuso de vulnerabilidades y fallos de configuración de entornos de Directorio Activo.
2. Tener un entorno propio (reducción de costes) donde practicar seguridad ofensiva y defensiva.
3. Disponer de un lugar para entender los fallos de configuración y las vulnerabilidades (ataque) como los eventos que se generan al intentar explotarlas (defensa).
4. Disponer de un lugar para probar soluciones o parches previo a un despliegue en entornos productivos.
5. Además, dicho entorno debe ser sencillo de manejar, desplegar y gestionar.
6. Debe ser fácilmente actualizable/mejorable y replicable.
7. Por último, debe ser un entorno autogestionado e independiente.



# Creación del entorno

# Creación del entorno

## *Herramientas necesarias: Hardware*

Para almacenar el entorno, podemos plantearnos varias opciones dentro del mercado:

Servidor interno



Azure



Mini PC



# Creación del entorno

## Herramientas necesarias: Hardware

La única opción que permite cumplir con todos los requisitos comentados en el apartado anterior es el Mini PC:

- No ocupa espacio.
- Es portátil.
- Potencia suficiente para ejecutar hasta 9 máquinas virtuales de manera concurrente.
- Su precio suele rondar los 400€.
- Dispone de dos slots de RAM hasta 16 GB, 1 slot para SSD M2 y 1 slot para SATA.
- Comprando a la baja, por 700~800€ tendríamos el Mini PC listo para albergar el entorno.



Fuente: <https://www.gizcomputer.com/nuc7i3bnh-mini-pc-caracteristicas/>

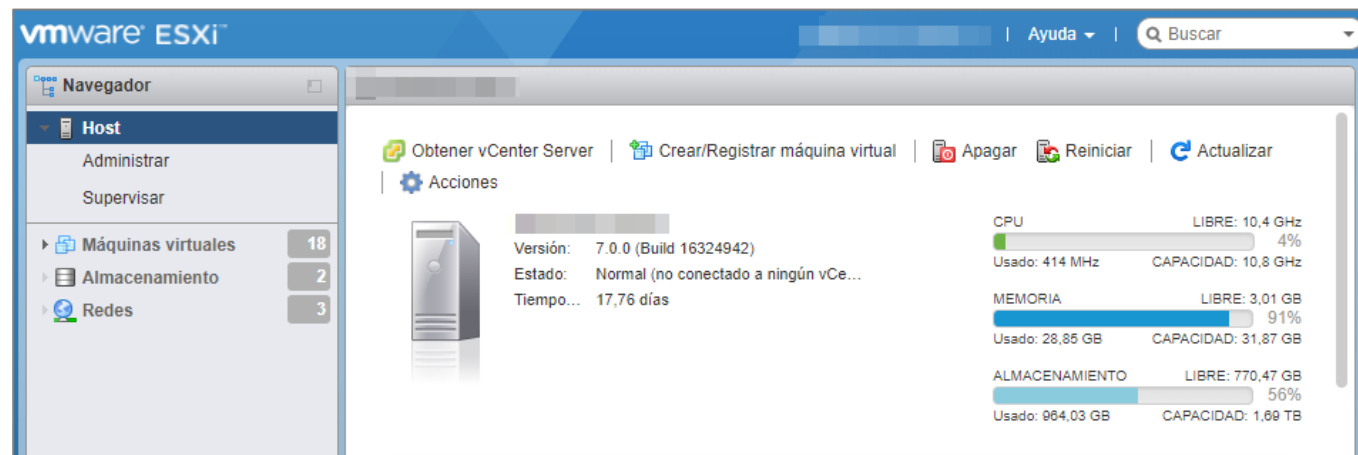
# Creación del entorno

## Herramientas necesarias: Software

En cuanto al apartado de software, la selección es mucho más sencilla. Es necesario disponer de un hipervisor para gestionar las máquinas virtuales del laboratorio y, como no, de las ISOs de los equipos a desplegar.

Las opciones elegidas son las siguientes:

- **VMWare ESXI** como hipervisor. Permite ser instalado como sistema operativo de un servidor físico y dispone de una versión gratuita con las funcionalidades ideales para gestionar entornos de prueba.
- Licencia de **VMWare Workstation**.
- **ISOs** de evaluación de Windows. Al ser versiones de evaluación, son gratuitas y funcionales durante 90 días (Windows 10) y 180 días (Windows Server).



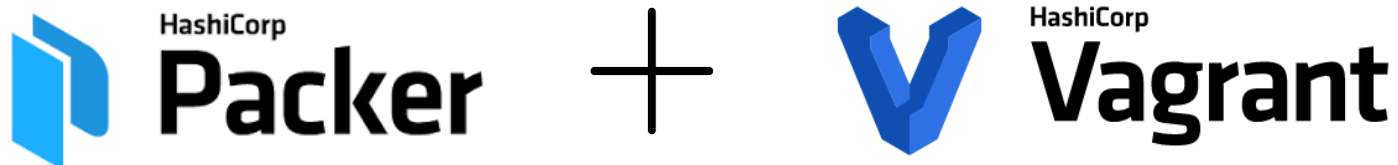


# Creación del entorno

## *Elección de despliegue*

Una vez tengamos montado el mini PC, el siguiente paso es elegir el proceso mediante el cual las máquinas virtuales (las plantillas), van a ser generadas. Para ello, podemos plantear dos procesos diferentes, aunque no excluyentes:

1. Crear el entorno de manera manual para, posteriormente, tomarlo como plantilla en la automatización del despliegue.
2. Usar herramientas como Packer y Vagrant para automatizar el proceso de creación de las plantillas.



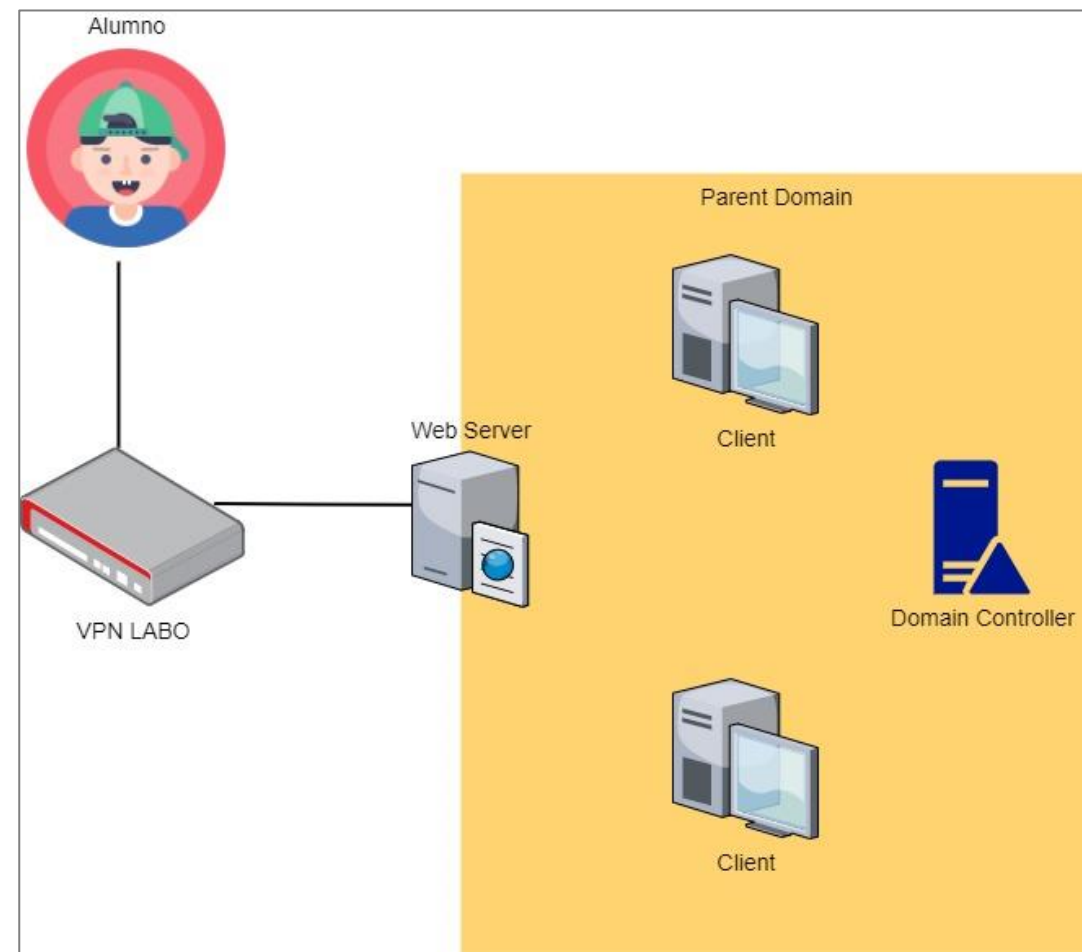
# Creación del entorno

## Elección de despliegue: Creación manual

La opción primera es la más sencilla, la creación del entorno de pruebas a mano. En otras palabras, desplegar manualmente todas las máquinas una por una como si de un entorno real se tratara.

En este caso, el entorno consta de cuatro máquinas virtuales; un DC, un servidor y dos equipos de usuario. Todos pertenecientes al mismo dominio: *ccn.local*.

Crear/Registrar máquina virtual   Consola   Encender   Apagar   Suspender   Actualizar				
<input type="checkbox"/>	Máquina virtual	Condición	Espacio utilizado	Sistema operativo invitado
<input type="checkbox"/>	WKSTN-01	✓ Normal	51,07 GB	Microsoft Windows 10 (64 bits)
<input type="checkbox"/>	WKSTN-02	✓ Normal	51,07 GB	Microsoft Windows 10 (64 bits)
<input type="checkbox"/>	DC-01	✓ Normal	93,87 GB	Microsoft Windows Server 2019 (64 bits)
<input type="checkbox"/>	SERVER-01	✓ Normal	94,05 GB	Microsoft Windows Server 2019 (64 bits)



# Creación del entorno

## Elección de despliegue: Packer

**Packer** es una herramienta de código abierto que permite crear imágenes de máquinas idénticas para múltiples plataformas a partir de una única configuración de origen. Este fichero de configuración suele estar escrito en formato JSON aunque, hace un mes, HashiCorp publicó la beta de HCL Packer, un lenguaje propietario para adaptarse al resto de la familia HashiCorp.

```
PS C:\Users\Jorge\Downloads\Packer> packer build -only=vmware-iso .\w2019.json
Warning: Warning when preparing build: "vmware-iso"

A checksum of 'none' was specified. Since ISO files are so big,
a checksum is highly recommended.

Warning: Warning when preparing build: "vmware-iso"

Your vmx data contains the following variable(s), which Packer normally sets
when it generates its own default vmx template. This may cause your build to
fail or behave unpredictably: scsi0.virtualDev

+-----+
+1;32mvmware-iso: output will be in this color.+[0m

+1;32m==> vmware-iso: Retrieving ISO+[0m
+1;32m==> vmware-iso: Trying https://software-download.microsoft.com/download/pr/17763.737.190906-2324.rs5_release_svc_refresh_SERVER_EVAL_x64FRE_en-US_1.iso+[0m
+1;32m==> vmware-iso: Trying https://software-download.microsoft.com/download/pr/17763.737.190906-2324.rs5_release_svc_refresh_SERVER_EVAL_x64FRE_en-US_1.iso+[0m
+1;32m==> vmware-iso: https://software-download.microsoft.com/download/pr/17763.737.190906-2324.rs5_release_svc_refresh_SERVER_EVAL_x64FRE_en-US_1.iso => C:\Users\
447.iso+[0m
+1;32m==> vmware-iso: Configuring output and export directories...+[0m
+1;32m==> vmware-iso: Creating floppy disk...+[0m
+0;32m vmware-iso: Copying files flatly from floppy_files+[0m
+0;32m vmware-iso: Copying file: .\answer_files\2019\Autounattend.xml+[0m
+0;32m vmware-iso: Copying file: .\scripts\2019\disable-screensaver.ps1+[0m
+0;32m vmware-iso: Copying file: .\scripts\2019\disable-winrm.ps1+[0m
+0;32m vmware-iso: Copying file: .\scripts\2019\enable-winrm.ps1+[0m
+0;32m vmware-iso: Copying file: .\scripts\2019\microsoft-updates.bat+[0m
+0;32m vmware-iso: Copying file: .\scripts\2019\unattend.xml+[0m
+0;32m vmware-iso: Copying file: .\scripts\2019\sysprep.bat+[0m
+0;32m vmware-iso: Copying file: .\scripts\2019\win-updates.ps1+[0m
+0;32m vmware-iso: Done copying files from floppy_files+[0m
+0;32m vmware-iso: Collecting paths from floppy_dirs+[0m
+0;32m vmware-iso: Resulting paths from floppy_dirs : []+[0m
+0;32m vmware-iso: Done copying paths from floppy_dirs+[0m
+1;32m==> vmware-iso: Creating required virtual machine disks+[0m
```

```
w10.json
1 {
2   "builders": [
3     {
4       "type": "vmware-iso",
5       "vm_name": "windows_10",
6       "communicator": "winrm",
7       "iso_url": "{{user `iso_url`}}",
8       "iso_checksum": "{{user `iso_checksum`}}",
9       "headless": false,
10      "boot_wait": "6m",
11      "boot_command": "",
12      "winrm_username": "vagrant",
13      "winrm_password": "vagrant",
14      "winrm_timeout": "4h",
15      "shutdown_timeout": "2h",
16      "shutdown_command": "a:/sysprep.bat",
17      "guest_os_type": "windows9-64",
18      "disk_size": "{{user `disk_size`}}",
19      "vnc_port_min": 5900,
20      "vnc_port_max": 5980,
21      "version": 11,
22      "floppy_files": [
23        "{{user `autounattend`}}",
24        ".\\scripts\\10\\fixnetwork.ps1",
25        ".\\scripts\\10\\disable-screensaver.ps1",
26        ".\\scripts\\10\\disable-winrm.ps1",
27        ".\\scripts\\10\\enable-winrm.ps1",
28        ".\\scripts\\10\\microsoft-updates.bat",
29        ".\\scripts\\10\\win-updates.ps1",
30        ".\\scripts\\10\\unattend.xml",
31        ".\\scripts\\10\\sysprep.bat"
32      ]
33    }
34  ]
35 }
```

Fuentes: <https://www.packer.io/intro>

# Creación del entorno





## Elección de despliegue: Vagrant

Por otro lado, **Vagrant** es una herramienta que permite construir y gestionar entornos de máquinas virtuales en un único flujo de trabajo. Originalmente se desarrolló para VirtualBox, pero, a día de hoy, está integrado con múltiples proveedores como VMWare, Amazon y Digital Ocean.

A diferencia de Packer, Vagrant funciona con un único fichero de configuración llamado *Vagrantfile*. Su función principal es describir el tipo de máquina requerida para un proyecto, definir su configuración y aprovisionarla con los recursos que se desee. Como es obvio, es necesario pasar a Vagrant una plantilla de VM sobre la que pueda trabajar. Para ello, podemos generarlas con Packer, mediante el uso de Post-Processors o descargar las imágenes ya disponibles en <https://app.vagrantup.com/boxes/search>.

```
PS C:\Users\Jorge\Downloads\Vagrant> vagrant up W2019
==> vagrant: A new version of Vagrant is available: 2.2.19 (installed version: 2.2.18)!
==> vagrant: To upgrade visit: https://www.vagrantup.com/downloads.html

Bringing machine 'W2019' up with 'vmware_desktop' provider...
==> W2019: Cloning VMware VM: 'W2019'. This can take some time...
==> W2019: Verifying vmnet devices are healthy...
==> W2019: Preparing network adapters...
==> W2019: Starting the VMware VM...
==> W2019: Waiting for the VM to receive an address...
```

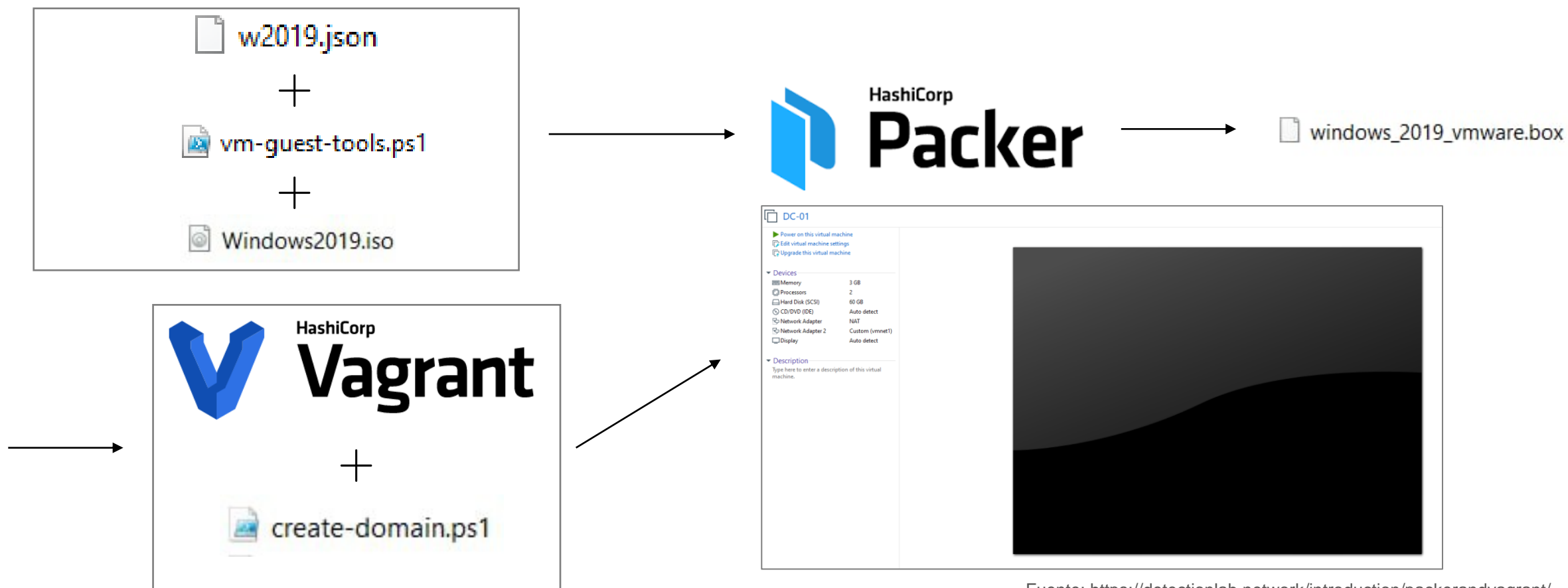
Discover Vagrant Boxes				
<input type="text" value="detectionlab"/>				
Provider <b>any</b> virtualbox vmware libvirt more		Sort by <b>Downloads</b> Recently Created Recently Updated		
	detectionlab/win2016 1.9	<span>virtualbox</span> <span>vmware_desktop</span>	Downloads 7,725	Released 2 months ago
	detectionlab/win10 1.8	<span>virtualbox</span> <span>vmware_desktop</span>	Downloads 4,405	Released about 1 year ago
	cyberdefenders/win10 1.2 A fork from Chris Long's DetectionLab with ELK stack instead of Splunk.	<span>virtualbox</span>	Downloads 206	Released about 1 year ago
	cyberdefenders/win2016 1.2 A fork from Chris Long's DetectionLab with ELK stack instead of Splunk.	<span>virtualbox</span>	Downloads 206	Released about 1 year ago

Fuentes: <https://www.vagrantup.com/intro>

# Creación del entorno

## *Elección de despliegue: Packer + Vagrant*

En resumen, el flujo de Packer con Vagrant sería el siguiente:



Fuente: <https://detectionlab.network/introduction/packerandvagrant/>

# Automatización del entorno





# Automatización del entorno

## *Terraform y Ansible*

Una vez que tenemos creadas todas las plantillas del laboratorio, el siguiente paso sería desplegarlo. Sin embargo, este despliegue requiere de ciertos puntos a tener en cuenta:

- Es necesaria una persona para iniciar este proceso.
- Ciertos comportamientos del entorno pueden requerir de una acción posterior al despliegue del laboratorio, como, por ejemplo, simular una sesión RDP entre dos equipos.
- El entorno requiere ser destruido cada cierto tiempo para garantizar una estabilidad y limpieza del ecosistema.

Para automatizar este proceso, Terraform y Ansible vienen a nuestro rescate.



**Nota:** La automatización se realiza desde una máquina Ubuntu sobre el mini PC.

# Automatización del entorno

## Terraform y Ansible: Terraform

**Terraform** es una herramienta de orquestación de código abierto que nos va a permitir definir nuestra infraestructura como código (*HashiCorp Configuration Language*). Posee integración con varios proveedores como Azure, AWS, Digital Ocean y, como no, VMWare vSphere.

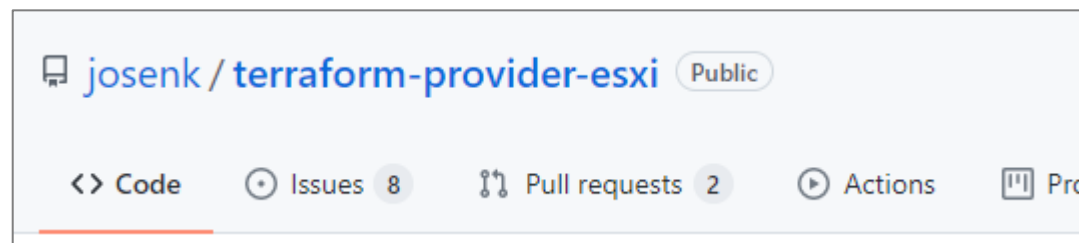
Gracias al proveedor de ESXi, podemos utilizar Terraform para clonar las máquinas virtuales creadas anteriormente, las despliegue y las arranque.

```
jordi@ubuntu:~/Desktop/ESXI$ terraform apply

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# esxi_guest.DC-01-Pruebas will be created
+ resource "esxi_guest" "DC-01-Pruebas" {
+   boot_disk_size      = (known after apply)
+   boot_disk_type      = "thin"
+   clone_from_vm       = "Backup-DC-01"
+   disk_store          = "datastore1"
+   guest_name          = "DC-01-Pruebas"
+   guest_shutdown_timeout = (known after apply)
+   guest_startup_timeout = (known after apply)
+   guestos              = "windows2019srv-64"
+   id                  = (known after apply)
+   ip_address           = (known after apply)
+   memsize              = (known after apply)
+   notes                = (known after apply)
+   numvcpus             = (known after apply)
+   ovf_properties_timer = (known after apply)
+   power                = "on"
+   resource_pool_name   = "/"
+   virthwver            = (known after apply)
}
```



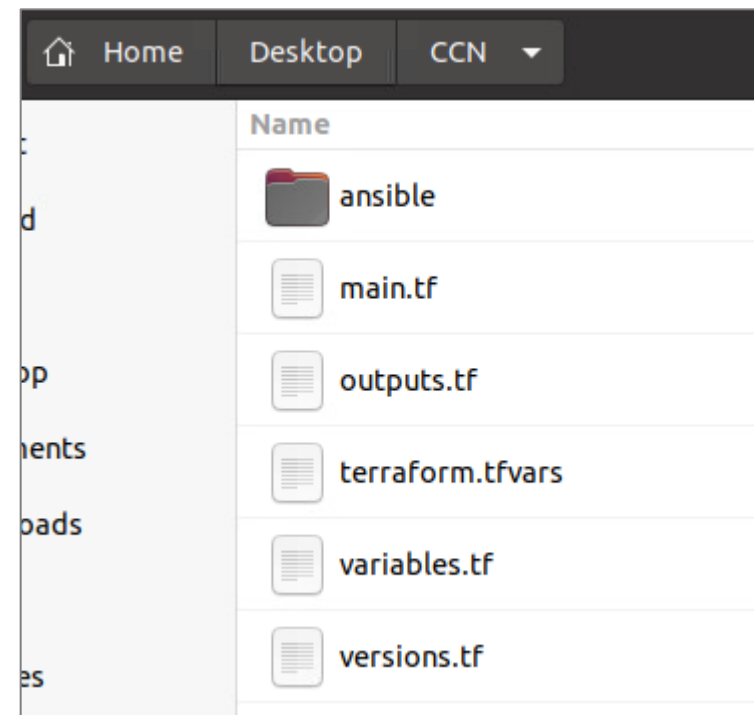
Fuente: <https://www.terraform.io/>

# Automatización del entorno

## *Terraform y Ansible: Terraform*

La estructura típica de archivos para cada módulo de Terraform es la siguiente:

- **Main.tf** - Fichero con la configuración principal del módulo. Máquinas a desplegar, característica de cada máquina, orden de despliegue...
- **Variables.tf** - Fichero con la definición de las variables del módulo. Usuario y contraseña del ESXI, redes, puerto de conexión...
- **Output.tf** - Fichero donde se define la información que pintará Terraform al terminar de ejecutar el despliegue. Por ejemplo, pintar por pantalla las IPs de cada máquina.
- **Versions.tf** - Fichero donde se definen las versiones de Terraform y los plugins a usar.
- **Terraform.tfvars** - Permite definir algunas variables no declaradas en Variables.tf.



Fuente: <https://learn.hashicorp.com/tutorials/terraform/module-create?in=terraform/modules>

# Automatización del entorno

## *Terraform y Ansible: Ansible*

En esta fase del despliegue, nuestro Directorio Activo se encuentra funcionando correctamente, pero ¿y si queremos simular actividad de usuarios? Es ahí donde entra Ansible.

**Ansible** es un motor open source que automatiza los procesos para preparar la infraestructura, gestionar la configuración, implementar las aplicaciones y organizar los sistemas. Su funcionamiento es sencillo, mediante SSH o WinRM, Ansible se conecta a los equipos e inserta pequeños programas denominados módulos, los cuales permiten llevar a cabo tareas de automatización en la plataforma. Ansible utiliza plantillas en formato YAML y permite incluir módulos para automatizar tareas. De hecho, para entornos Windows, dispone de su propio módulo desarrollado en PowerShell.



```
1 ---
2
3
4 - name: Fijar IP
5   win_shell: "If (-not(get-netipaddress | where {$_.IPAddress -eq '10.10.1.45'}))
6     {$adapter = (get-netipaddress | where {$_.IpAddress -like '169.*'}).InterfaceAlias; New-
      NetIPAddress -InterfaceAlias $adapter -AddressFamily IPv4 -IPAddress 10.10.1.45 -
      PrefixLength 24 -DefaultGateway 10.10.1.1 } Else { Write-Host 'IP Address Already
      Created.' }"
```

**Nota:** Aunque Ansible es **muy** potente, nosotros en este esquema solo lo hemos utilizado para simular un entorno vivo.

Fuente: <https://www.redhat.com/es/topics/automation/learning-ansible-tutorial>

# Automatización del entorno

## Terraform y Ansible: Ansible

**Ansible** funciona en base a *playbooks*. Como su nombre indica, el *playbook* contiene toda la información necesaria para aplicar los despliegues sobre las máquinas objetivo. A su vez, es necesario facilitarte un inventario, donde se indica la IP de los equipos y el usuario y contraseña a utilizar por Ansible para conectarse.

```
jordi@ubuntu:~/Desktop/CCN/ansible$ ansible-playbook ccnlab.yml -i inventory
[WARNING]: Ansible is being run in a world writable directory (/home/jordi/Desktop/CCN/ansible)
https://docs.ansible.com/ansible/devel/reference_appendices/config.html#cfg-in-world-
[WARNING]: Unable to parse /home/jordi/Desktop/CCN/ansible/inventory as an inventory
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
[WARNING]: While constructing a mapping from /home/jordi/Desktop/CCN/ansible/roles/WK
[WARNING]: While constructing a mapping from /home/jordi/Desktop/CCN/ansible/roles/WK
[WARNING]: While constructing a mapping from /home/jordi/Desktop/CCN/ansible/roles/WK
[WARNING]: Could not match supplied host pattern, ignoring: DC-01

PLAY [DC-01] *****
```

```
ccnlab.yml
~/Desktop/CCN/ansible

1 ---
2 - hosts: DC-01
3   roles:
4     - DC-01
5   tags: DC-01
6
7 - hosts: WKSTN-01
8   roles:
9     - WKSTN-01
10  tags: WKSTN-01
11
12 - hosts: WKSTN-02
13   roles:
14     - WKSTN-02
15   tags: WKSTN-02
16
17 - hosts: WEB-01
18   roles:
19     - WEB-01
20   tags: WEB-01
21
```

Fuente: [https://docs.ansible.com/ansible/latest/user\\_guide/](https://docs.ansible.com/ansible/latest/user_guide/)

# Despliegue final





# Despliegue final

## Resumiendo...

Recapitulemos.

1. Hemos creado las máquinas que formarán parte de nuestro laboratorio a mano.
2. Hemos almacenado dichas OVAs en el Mini PC.
3. Usando Terraform, se clonan las máquinas creadas para desplegar el laboratorio de manera automática.
4. Usando Ansible, se despliegan una serie de procesos para simular un entorno vivo y con movimiento.
5. Quedaría crear una tarea programada que destruya el entorno (Terraform destroy) todas las noches y vuelva a desplegarlo y aprovisionarlo (Terraform deploy + Ansible playbook) en la máquina Ubuntu.

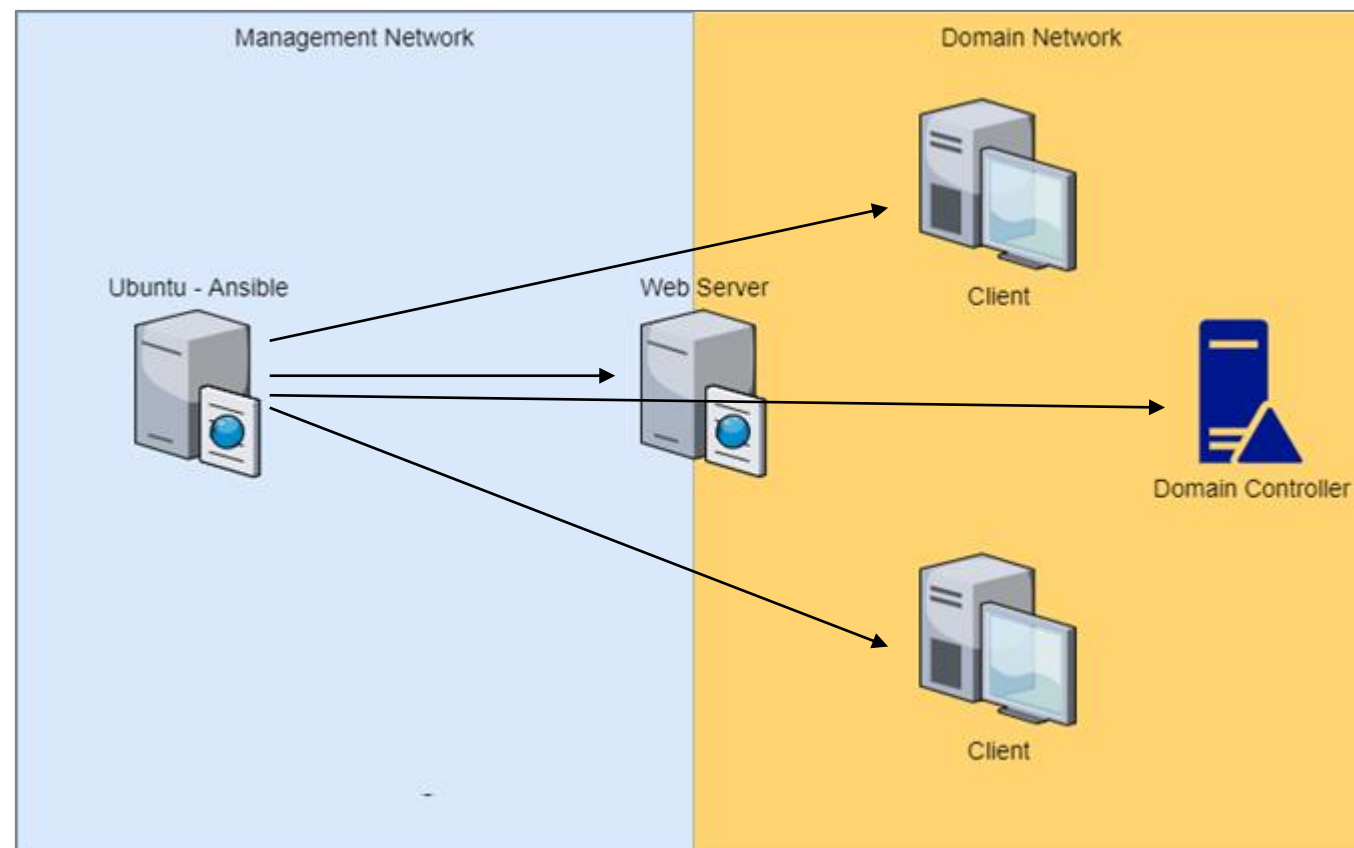


# Despliegue final

## Arquitectura

A nivel de arquitectura, dentro del mini PC, tenemos el siguiente esquema de máquinas:

<input type="checkbox"/>	Máquina virtual	▼	Sistema operativo invitado ▲
<input type="checkbox"/>	WKSTN-01		Microsoft Windows 10 (64 bits)
<input type="checkbox"/>	WKSTN-02		Microsoft Windows 10 (64 bits)
<input type="checkbox"/>	DC-01		Microsoft Windows Server 2019 (64 bits)
<input type="checkbox"/>	SERVER-01		Microsoft Windows Server 2019 (64 bits)
<input type="checkbox"/>	Ubuntu-Ansible		Ubuntu Linux (64 bits)



# Demo



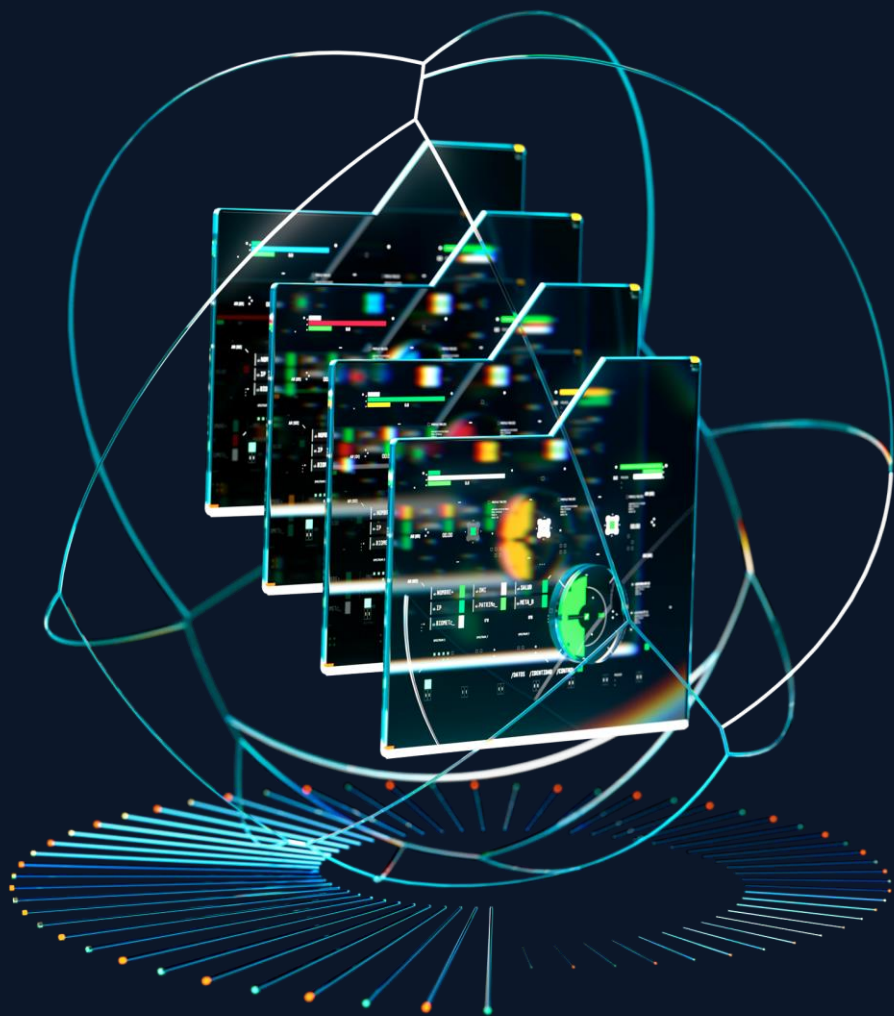
# Demo

Ficheros: [https://anonfiles.com/Fd40mfY6u4/Taller\\_Labo\\_AD\\_7z](https://anonfiles.com/Fd40mfY6u4/Taller_Labo_AD_7z)

Pass: **Taller\_J0rnadas\_2021\_AD\_Automatizand0!!**



# Conclusiones



# Conclusiones

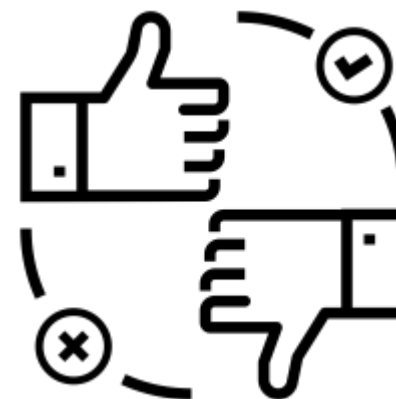
## *Ventajas vs Desventajas*

### Ventajas

- La versatilidad del mini PC nos tener entornos “todo en uno”.
- La existencia de herramientas como Ansible, Terraform, Packer y Vagrant nos habilita para crear estos entornos de manera rápida y sencilla.
- El coste de mantenimiento es inexistente.
- Disposición de laboratorios efímeros.
- Es posible crear entornos híbridos (Azure + Mini PC + Servidor Interno).
- Perfecto para entornos de pruebas y laboratorios internos.

### Desventajas

- Es necesario disponer de un mantenimiento continuo de las plantillas.
- La escalabilidad es limitada (recursos).
- Complejidad de uso de Ansible, Terraform, Packer y Vagrant.
- Necesidad de conocimiento para la generación de los entornos.
- VMWare ESXI en su versión gratuita no trae todas las funcionalidades necesarias.
- Coste ligeramente elevado, pero amortizable en un periodo de tiempo corto.





# Conclusiones

## Posibles mejoras

1. Integración de un ciclo CI/CD.



2. Creación de diferentes entornos de Directorio Activo según personas objetivo.



PRINCIPIANTE



INTERMEDIO



AVANZADO



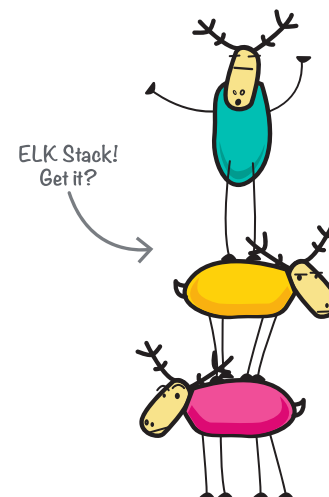
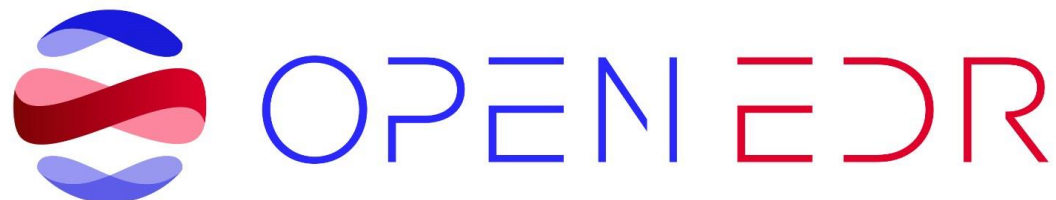
EXPERTO

3. Adición de elementos para Threat Hunting (Sysmon, WEF, ELK).



Sysmon

4. Inclusión de elementos de seguridad como OpenEDR.



**E** Elasticsearch

**L** Logstash

**K** Kibana

# Referencias



# Referencias

- ESXI -> <https://www.vmware.com/es/products/esxi-and-esx.html>
- Documentación ESXi -> <https://docs.vmware.com/es/VMware-vSphere/index.html>
- ISOs de evaluación de Windows -> <https://www.microsoft.com/es-es/evalcenter/>
- Introducción a Packer -> <https://www.packer.io/intro>
- VMWare Builder Packer -> <https://www.packer.io/docs/builders/vmware/iso>
- Communicators -> <https://www.packer.io/docs/communicators>
- Builders -> <https://www.packer.io/docs/builders/vsphere>
- Datasources -> <https://www.packer.io/docs/datasources>
- Provisioners -> <https://www.packer.io/docs/provisioners/windows-shell>
- Post-Processors -> <https://www.packer.io/docs/post-processors/vagrant/vagrant>
- Plantillas Packer Windows -> <https://github.com/StefanScherer/packer-windows>
- Introducción a Vagrant -> <https://www.vagrantup.com/intro>
- VMWare Provider Vagrant -> <https://www.vagrantup.com/docs/providers/vmware>
- Boxes Vagrant -> <https://app.vagrantup.com/boxes/search>
- Plantillas ejemplo Vagrant -> <https://github.com/clong/DetectionLab/tree/master/Vagrant>
- Packer vs Vagrant -> <https://detectionlab.network/introduction/packerandvagrant/>
- VagrantFile para ESXI -> <https://github.com/josenk/vagrant-vmware-esxi>

# Referencias

- Introducción a Terraform -> <https://www.terraform.io/>
- Proveedor de Terraform para ESXI -> <https://github.com/josenk/terraform-provider-esxi>
- Ficheros de Terraform -> <https://learn.hashicorp.com/tutorials/terraform/module-create?in=terraform/modules>
- Introducción a Ansible -> <https://www.redhat.com/es/topics/automation/learning-ansible-tutorial>
- Documentación Ansible -> [https://docs.ansible.com/ansible/latest/user\\_guide/index.html#getting-started](https://docs.ansible.com/ansible/latest/user_guide/index.html#getting-started)
- Ansible y Windows -> <https://www.redhat.com/en/topics/automation/automate-microsoft-windows-with-ansible>
- Ejemplo laboratorio AD con Vagrant y Ansible -> <https://github.com/jckhmr/adlab>
- Ejemplo laboratorio AD con Vagrant y Ansible II -> <https://github.com/Orange-Cyberdefense/GOAD>
- Ejemplo laboratorio en ESXI con Terraform y Ansible -> <https://github.com/clong/DetectionLab/tree/master/ESXi>
- OpenEDR -> <https://github.com/ComodoSecurity/openedr>



# MUCHAS GRACIAS

**#XVJORNADASCCNCERT**