Proxmox Lab: Game of Active Directory - Creating VM Templates

benheater.com/proxmox-lab-goad-creating-vm-templates

0xBEN August 26, 2024

In this module, we'll be taking steps to create some Windows Server 2016 and Windows Server 2019 templates using Packer for use in the Proxmox Game of Active Directory (GOAD) v3 lab

0xBEN
Aug 26, 2024 7 min read

ℹ️

This module is part of a larger project on setting up ***Game of Active Directory (GOAD) v3*** on Proxmox alongside our existing lab infrastructure. Click here to be taken back to the project landing page.

Previous Step

Proxmox Lab: Game of Active Directory - Environment Setup

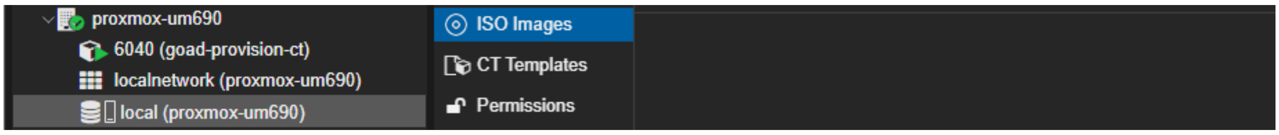
In this module, we'll be taking steps to set up the initial environment and prepare to deploy Game of Active Directory (GOAD) in our existing Proxmox environment.

Objectives for this Step

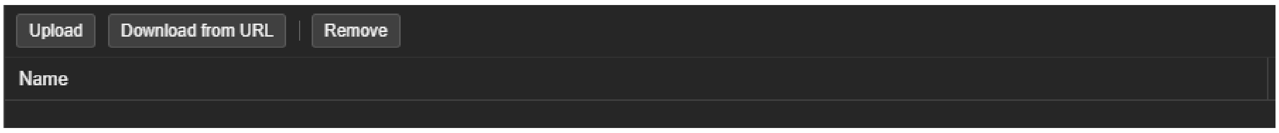
- Log into the provisioning Linux Container
- Create packer configurations
- Automate the creation of Windows ISOs to serve as templates in the environment

Prepare the ISOs

Download ISOs to PVE



Click on your PVE node > "local" > "ISO Images"

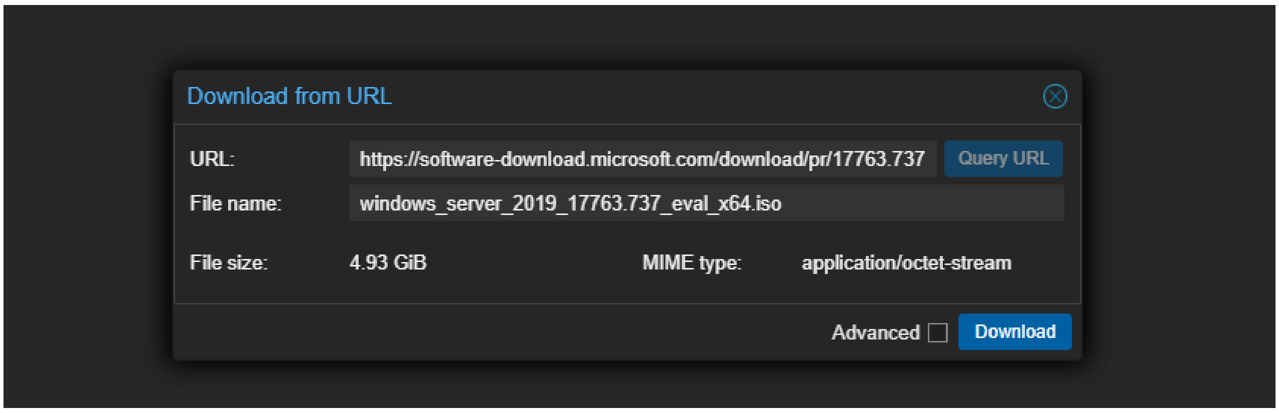


Click "Download from URL"

Windows Server ISOs

Microsoft Official Links

Links last verified to be working as of Aug. 23, 2024



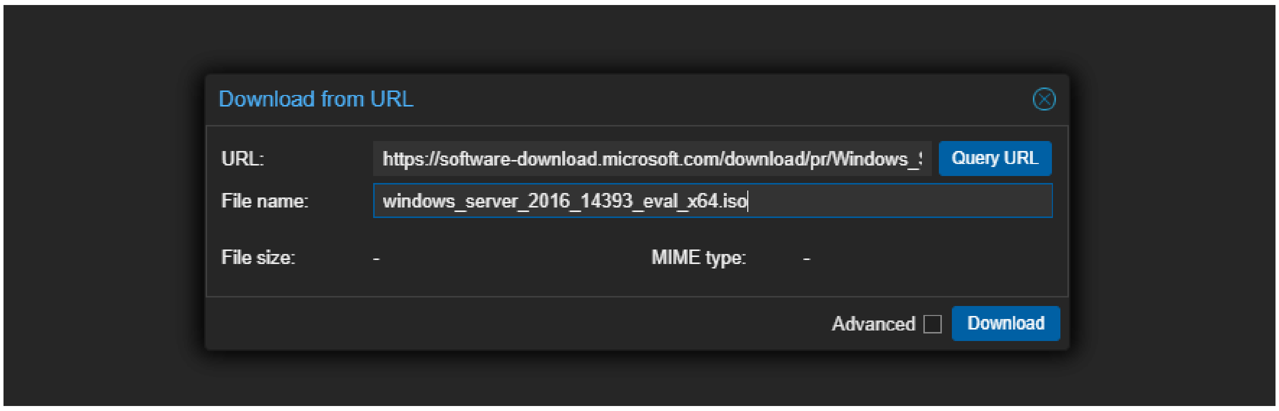
Download from URL

URL: https://software-download.microsoft.com/download/pr/17763.737

File name: windows_server_2019_17763.737_eval_x64.iso

File size: 4.93 GiB MIME type: application/octet-stream

Manually enter the file name, as clicking "Query URL" results in an extremely long file name. Click "Download"



Download from URL

URL: https://software-download.microsoft.com/download/pr/Windows_

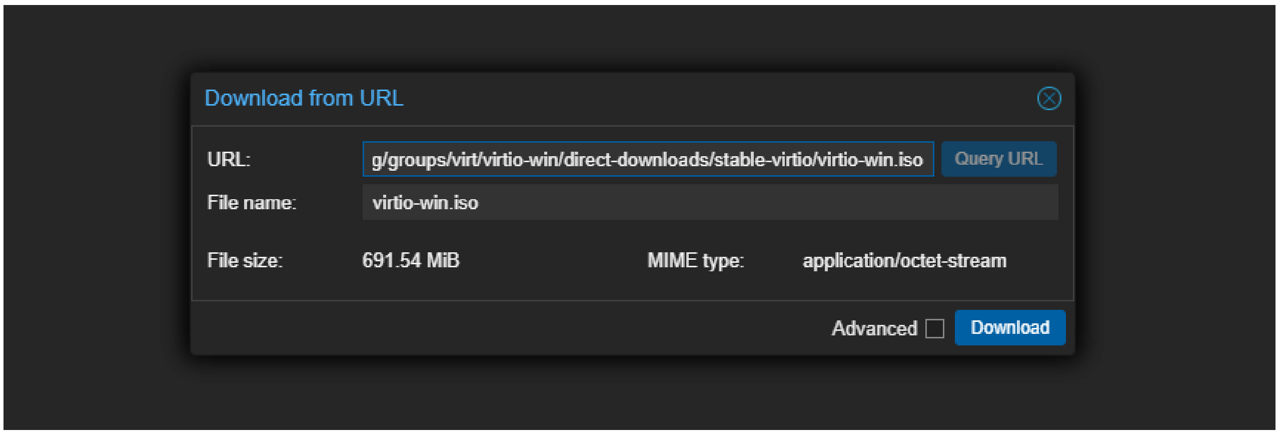
File name: windows_server_2016_14393_eval_x64.iso

File size: - MIME type: -

Repeat the process for Windows Server 2016

ISOs to Install Drivers on Windows Guests

Official Link to VirtIO Drivers for Windows


Download from URL

URL: g/groups/virt/virtio-win/direct-downloads/stable-virtio/virtio-win.iso

File name: virtio-win.iso

File size: 691.54 MiB

MIME type: application/octet-stream

In this case, you can click "Query URL" and then click "Download"

Build ISO for Unattended Install

These steps need to be done on your provisioning Linux Container

```
cd /root/GOAD/packer/proxmox/scripts/sysprep/
```

Bash

```
wget https://cloudbase.it/downloads/CloudbaseInitSetup_Stable_x64.msi
```

Bash

```
cd /root/GOAD/packer/proxmox/
```

Bash

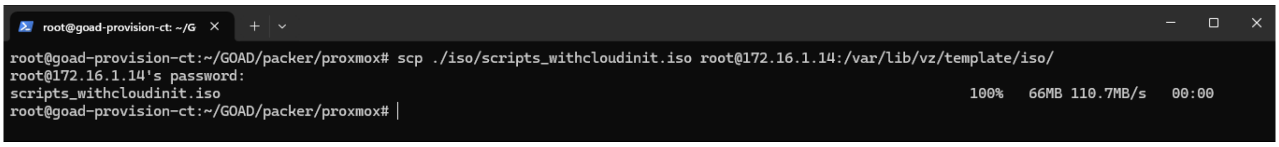
```
./build_proxmox_iso.sh
```

Bash

```
scp ./iso/scripts_withcloudinit.iso root@172.16.1.14:/var/lib/vz/template/iso/
```

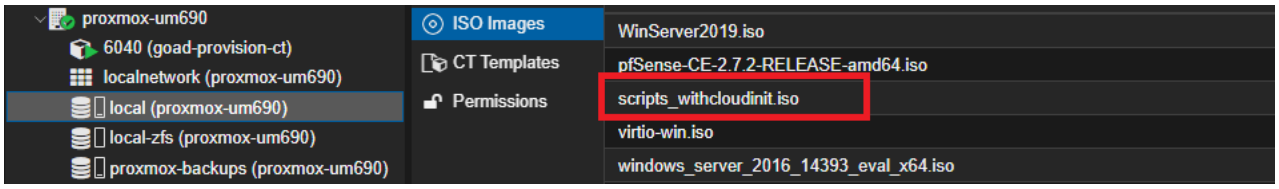
Bash

Run `scp` on the provisioning container to copy `scripts_withcloudinit.iso` to your PVE node's `local` ISO storage directory



```
root@goad-provision-ct:~/GOAD/packer/proxmox# scp ./iso/scripts_withcloudinit.iso root@172.16.1.14:/var/lib/vz/template/iso/
root@172.16.1.14's password:
scripts_withcloudinit.iso                                                                                   100%   66MB 110.7MB/s   00:00
root@goad-provision-ct:~/GOAD/packer/proxmox# |
```

File was copied via `scp`



```
WinServer2019.iso
pfSense-CE-2.7.2-RELEASE-amd64.iso
scripts_withcloudinit.iso
virtio-win.iso
windows_server_2016_14393_eval_x64.iso
```

Verifying we see it on the host as well

Stage Packer Environment

ℹ️

These steps need to be done on your provisioning Linux Container

⚠️

We're going to skip the step of creating the dedicated user as done by Mayfly, because in the guide, we end up giving the `infra_as_code@pve` user full admin anyway, so we can just use the `root` credentials

Packer Build Variables

```
cd /root/GOAD/packer/proxmox/
```

Bash

```
cp config.auto.pkrvars.hcl.template config.auto.pkrvars.hcl
```

Bash

```
nano config.auto.pkrvars.hcl
```

Bash

⚠️

Since my environment and your environment are not identical, please take care to ensure you're entering details ***tailored to your environment***

```
proxmox_url              = "https://172.16.1.14:8006/api2/json"
proxmox_username         = "root@pam"
proxmox_password         = "YOUR_ROOT_PASSWORD_HERE"
proxmox_skip_tls_verify = "true"
proxmox_node             = "proxmox-um690"
proxmox_pool             = "GOAD"
proxmox_iso_storage      = "local"
proxmox_vm_storage       = "local-lvm"
```

HCL

❗ **Specific changes to note:**

- `proxmox_url` — this is the IP address of my PVE node
- `root@pam` — because we did not create a service account
- `proxmox_password` — your root user's password on PVE (not the container)
- `proxmox_node` — the hostname of your PVE node
  You cand find this in the GUI under `Datacenter` view in the top-left
- `proxmox_pool` — using `GOAD` here, as this is the pool I created earlier
- `proxmox_vm_storage` — be sure you're using the correct storage name
  - For example, on my PVE node, I use `local-zfs`
  - I left it to `local-lvm` here, because most folks are using the default

Windows ISO Variables

Server 2019

```
cd /root/GOAD/packer/proxmox/
```

Bash

```
nano windows_server2019_proxmox_cloudinit.pkvars.hcl
```

Bash

```
winrm_username        = "vagrant"
winrm_password        = "vagrant"
vm_name               = "WinServer2019x64-cloudinit-qcow2"
template_description  = "Windows Server 2019 64-bit - build 17763.737.190906-2324
- template built with Packer - cloudinit - {{isotime \"2006-01-02 03:04:05\"}}"
iso_file              = "local:iso/windows_server_2019_17763.737_eval_x64.iso"
autounattend_iso      = "./iso/Autounattend_winserver2019_cloudinit.iso"
autounattend_checksum =
"sha256:811b24542a4a3c9787745ca1d303220396dc8ac4fb9b268c71ea72ff95267115"
vm_cpu_cores          = "2"
vm_memory             = "4096"
vm_disk_size          = "40G"
vm_sockets            = "1"
os                    = "win10"
vm_disk_format        = "qcow2"
```

HCL

❗ **Changes to My Variables File**

- `iso_file` — `"local:iso/windows_server_2019_17763.737_eval_x64.iso"`
  This is the name of file I gave the ISO when I downloaded it in an earlier step

- `vm_disk_format` — `"raw"`
    - Because my guest storage is `local-zfs` and not `local-lvm`, I need to use the `raw` storage format
    - I left it as `qcow2` in the example above, because that will suit most people

Server 2016

```
cd /root/GOAD/packer/proxmox/
```

Bash

```
nano windows_server2016_proxmox_cloudinit.pkvars.hcl
```

Bash

```
winrm_username = "vagrant"
winrm_password = "vagrant"
vm_name = "WinServer2016x64-cloudinit-qcow2"
template_description = "Windows Server 2016 64-bit - build 14393 - template built
with Packer - cloudinit - {{isotime \"2006-01-02 03:04:05\"}}"
iso_file = "local:iso/windows_server_2016_14393_eval_x64.iso"
autounattend_iso = "./iso/Autounattend_winserver2016_cloudinit.iso"
autounattend_checksum =
"sha256:848c1e8a6c5de5f3944cecdad8878250daf1d32c4858876c1d3be83bd7cfd610"
vm_cpu_cores = "2"
vm_memory = "4096"
vm_disk_size = "40G"
vm_sockets = "1"
os = "win10"
vm_disk_format = "qcow2"
```

HCL

❗ Changes to My Variables File

- `iso_file` — `"local:iso/windows_server_2016_14393_eval_x64.iso"`
  This is the name of file I gave the ISO when I downloaded it in an earlier step
- `vm_disk_format` — `"raw"`
    - Because my guest storage is `local-zfs` and not `local-lvm`, I need to use the `raw` storage format
    - I left it as `qcow2` in the example above, because that will suit most people

Build the Templates

Packer Build File

```
cd /root/GOAD/packer/proxmox
```

Bash

```
nano packer.json.pkr.hcl
```

Bash

packer.json.pkr.hcl (click to expand)

```
packer {
  required_plugins {
    proxmox = {
      version = ">= 1.1.2"
      source  = "github.com/hashicorp/proxmox"
    }
  }
}

source "proxmox-iso" "windows" {
  additional_iso_files {
    device           = "sata3"
    iso_checksum     = "${var.autounattend_checksum}"
    iso_storage_pool = "local"
    iso_url          = "${var.autounattend_iso}"
    unmount          = true
  }
  additional_iso_files {
    device   = "sata4"
    iso_file = "local:iso/virtio-win.iso"
    unmount  = true
  }
  additional_iso_files {
    device   = "sata5"
    iso_file = "local:iso/scripts_withcloudinit.iso"
    unmount  = true
  }
  cloud_init              = true
  cloud_init_storage_pool = "${var.proxmox_vm_storage}"
  communicator            = "winrm"
  cores                   = "${var.vm_cpu_cores}"
  disks {
    disk_size         = "${var.vm_disk_size}"
    format            = "${var.vm_disk_format}"
    storage_pool      = "${var.proxmox_vm_storage}"
    type              = "sata"
  }
  insecure_skip_tls_verify = "${var.proxmox_skip_tls_verify}"
  iso_file                 = "${var.iso_file}"
  memory                   = "${var.vm_memory}"
  network_adapters {
    bridge = "vmbr1"
    model  = "virtio"
    vlan_tag = "10"
  }
  node                 = "${var.proxmox_node}"
  os                   = "${var.os}"
  password             = "${var.proxmox_password}"
  pool                 = "${var.proxmox_pool}"
  proxmox_url          = "${var.proxmox_url}"
  sockets              = "${var.vm_sockets}"
  template_description = "${var.template_description}"
  template_name        = "${var.vm_name}"
  username             = "${var.proxmox_username}"
  vm_name              = "${var.vm_name}"
  winrm_insecure       = true
  winrm_no_proxy       = true
  winrm_password       = "${var.winrm_password}"
  winrm_timeout        = "120m"
  winrm_use_ssl        = true
  winrm_username       = "${var.winrm_username}"
  task_timeout         = "40m"
}

build {
  sources = ["source.proxmox-iso.windows"]

  provisioner "powershell" {
    elevated_password = "vagrant"
    elevated_user     = "vagrant"
    scripts           = ["${path.root}/scripts/sysprep/cloudbase-init.ps1"]
  }

  provisioner "powershell" {
    elevated_password = "vagrant"
    elevated_user     = "vagrant"
    pause_before      = "1m0s"
    scripts           = ["${path.root}/scripts/sysprep/cloudbase-init-p2.ps1"]
  }

}
```

JSON

❗ Changes Made to the Configuration

```
cloud_init_storage_pool = "${var.proxmox_vm_storage}"
```

JSON

- `${var.proxmox_iso_storage}` has been changed to `${var.proxmox_vm_storage}`
  - Because `proxmox_iso_storage` is set in the packer variables as `local`
  - And, `local` does not support content type of disk image

```
  network_adapters {
    bridge = "vmbr1"
    model  = "virtio"
    vlan_tag = "10"
  }
```

JSON

- `bridge = "vmbr3"` has been changed to `bridge = "vmbr1"` to match the setup from the original Proxmox lab
  - See the network diagram at the main landing page

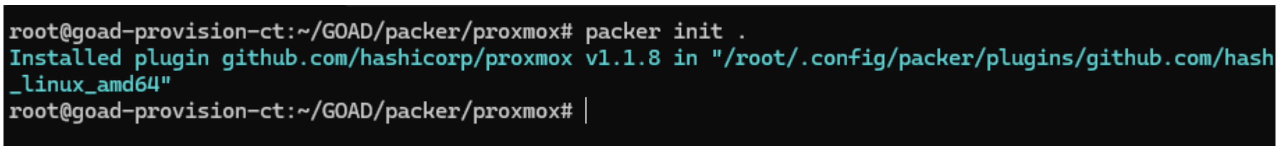
Build Server 2019 Template

```
cd /root/GOAD/packer/proxmox
```

Bash

```
packer init .
```

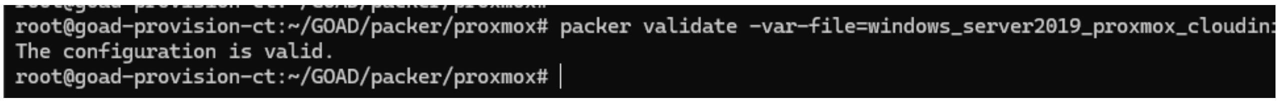
Bash



```
root@goad-provision-ct:~/GOAD/packer/proxmox# packer init .
Installed plugin github.com/hashicorp/proxmox v1.1.8 in "/root/.config/packer/plugins/github.com/hash
_linux_amd64"
root@goad-provision-ct:~/GOAD/packer/proxmox# |
```

```
packer validate -var-file=windows_server2019_proxmox_cloudinit.pkvars.hcl .
```

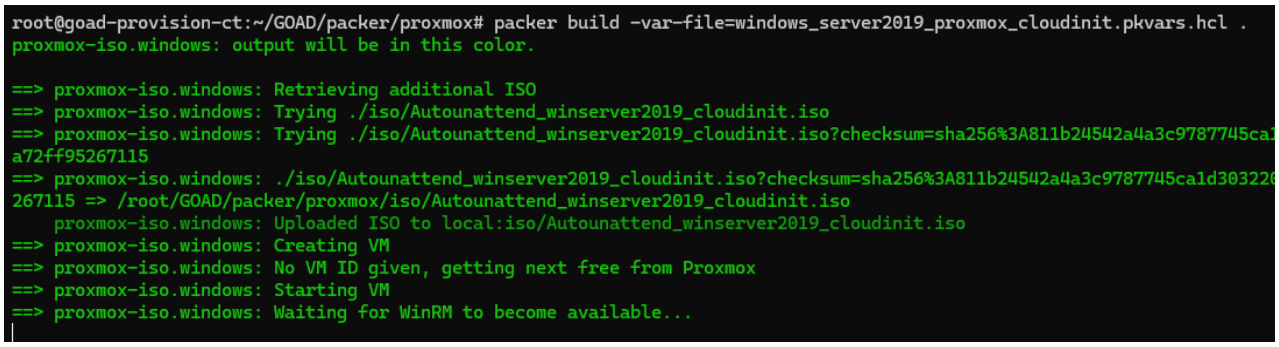
Bash



```
root@goad-provision-ct:~/GOAD/packer/proxmox# packer validate -var-file=windows_server2019_proxmox_cloudin
The configuration is valid.
root@goad-provision-ct:~/GOAD/packer/proxmox# |
```

```
packer build -var-file=windows_server2019_proxmox_cloudinit.pkvars.hcl .
```

Bash



```
root@goad-provision-ct:~/GOAD/packer/proxmox# packer build -var-file=windows_server2019_proxmox_cloudinit.pkvars.hcl .
proxmox-iso.windows: output will be in this color.

==> proxmox-iso.windows: Retrieving additional ISO
==> proxmox-iso.windows: Trying ./iso/Autounattend_winserver2019_cloudinit.iso
==> proxmox-iso.windows: Trying ./iso/Autounattend_winserver2019_cloudinit.iso?checksum=sha256%3A811b24542a4a3c9787745ca1
a72ff95267115
==> proxmox-iso.windows: ./iso/Autounattend_winserver2019_cloudinit.iso?checksum=sha256%3A811b24542a4a3c9787745ca1d30322
267115 => /root/GOAD/packer/proxmox/iso/Autounattend_winserver2019_cloudinit.iso
    proxmox-iso.windows: Uploaded ISO to local:iso/Autounattend_winserver2019_cloudinit.iso
==> proxmox-iso.windows: Creating VM
==> proxmox-iso.windows: No VM ID given, getting next free from Proxmox
==> proxmox-iso.windows: Starting VM
==> proxmox-iso.windows: Waiting for WinRM to become available...
|
```

Build Server 2016 Template

```
cd /root/GOAD/packer/proxmox
```

Bash

```
packer init .
```

Bash

```
packer validate -var-file=windows_server2016_proxmox_cloudinit.pkvars.hcl .
```

Bash

```
packer build -var-file=windows_server2016_proxmox_cloudinit.pkvars.hcl .
```

Bash

Troubleshooting Build Errors

Server 2016 — Exit Code 259

💡

Run the build again with logging output by using some runtime environment variables `PACKER_LOG` and `PACKER_LOG_PATH`

```
PACKER_LOG=1 PACKER_LOG_PATH=server_2016_packer.log packer build -var-file=windows_server2016_proxmox_cloudinit.pkvars.hcl .
```

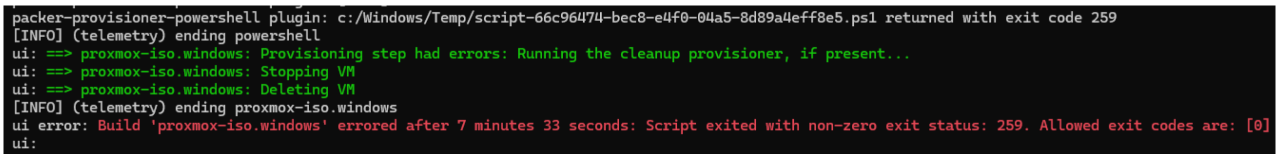
Bash

Example showing re-running of `packer build` command with log output enabled, logs to `./packer.log`

```
less -R ./server_2016_packer.log
```

Bash

Read the build log file after another failure ...



```
packer-provisioner-powershell plugin: c:/Windows/Temp/script-66c96474-bec8-e4f0-04a5-8d89a4eff8e5.ps1 returned with exit code 259
[INFO] (telemetry) ending powershell
ui: ==> proxmox-iso.windows: Provisioning step had errors: Running the cleanup provisioner, if present...
ui: ==> proxmox-iso.windows: Stopping VM
ui: ==> proxmox-iso.windows: Deleting VM
[INFO] (telemetry) ending proxmox-iso.windows
ui error: Build 'proxmox-iso.windows' errored after 7 minutes 33 seconds: Script exited with non-zero exit status: 259. Allowed exit codes are: [0]
ui:
```

We can see more detailed error output about what happened before Packer stopped. The script it tried to run exited with error code `259`.

💡

Exit code `259` indicates a status of `No more data available` and is not technically an error. We can tell `packer` to ignore this error and continue with the build.

This is likely an exit code from the `sysprep` command at the end of `/root/GOAD/packer/proxmox/scripts/sysprep/cloudbase-init-p2.ps1`

```
cd /root/GOAD/packer/proxmox
```

Bash

```
nano packer.json.pkr.hck
```

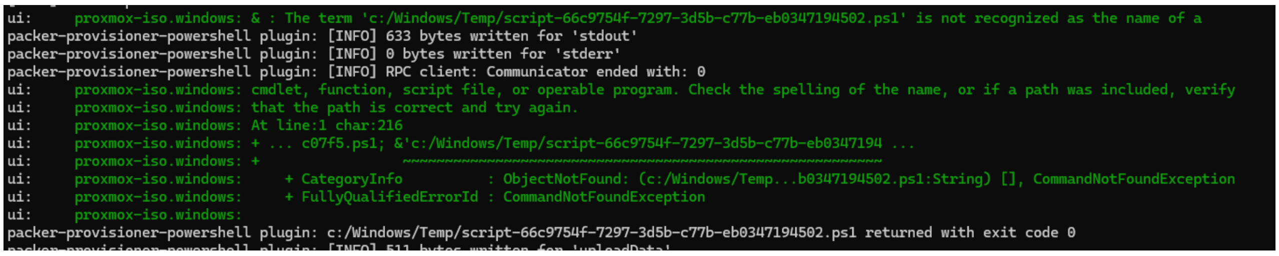
Bash

```
                              provisioner "powershell" {
                              elevated_password = "vagrant"
                              elevated_user     = "vagrant"
                               pause_before      = "1m0s"
scripts            = ["${path.root}/scripts/sysprep/cloudbase-init-p2.ps1"]
                              valid_exit_codes  = [0, 259]
                                                        }
```

JSON

Adds in `valid_exit_codes = [0, 259]` to tell packer to allow `259` as a non-error exit code

Server 2019 — Command Not Found

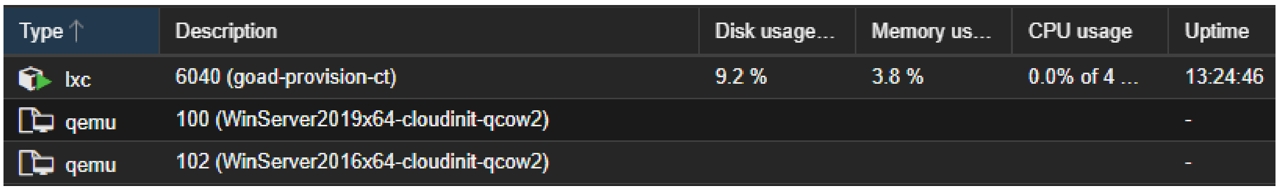


```
ui:      proxmox-iso.windows: & : The term 'c:/Windows/Temp/script-66c9754f-7297-3d5b-c77b-eb0347194502.ps1' is not recognized as the name of a
packer-provisioner-powershell plugin: [INFO] 633 bytes written for 'stdout'
packer-provisioner-powershell plugin: [INFO] 0 bytes written for 'stderr'
packer-provisioner-powershell plugin: [INFO] RPC client: Communicator ended with: 0
ui:      proxmox-iso.windows: cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify
ui:      proxmox-iso.windows: that the path is correct and try again.
ui:      proxmox-iso.windows: At line:1 char:216
ui:      proxmox-iso.windows: + ... c07f5.ps1; &'c:/Windows/Temp/script-66c9754f-7297-3d5b-c77b-eb0347194 ...
ui:      proxmox-iso.windows: +                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ui:      proxmox-iso.windows:     + CategoryInfo          : ObjectNotFound: (c:/Windows/Temp...b0347194502.ps1:String) [], CommandNotFoundException
ui:      proxmox-iso.windows:     + FullyQualifiedErrorId : CommandNotFoundException
ui:      proxmox-iso.windows:
packer-provisioner-powershell plugin: c:/Windows/Temp/script-66c9754f-7297-3d5b-c77b-eb0347194502.ps1 returned with exit code 0
packer-provisioner-powershell plugin: [INFO] 511 bytes written for 'uploadData'
```

You might see several of these command not found errors in the output, best answer I can come up with is that the attempt to copy the script files over WinRM did not complete before `packer` tried to execute them.

Cancel the build with `CTRL + C` and try again until no errors are present.

Current State of the Lab



Type ↑	Description	Disk usage...	Memory us...	CPU usage	Uptime
lxc	6040 (goad-provision-ct)	9.2 %	3.8 %	0.0% of 4 ...	13:24:46
qemu	100 (WinServer2019x64-cloudinit-qcow2)				-
qemu	102 (WinServer2016x64-cloudinit-qcow2)				-

You should now have your provisioning CT and two server templates created by Packer

Next Step

Proxmox Lab: Game of Active Directory - Installing the Lab

In this module, we'll be taking steps to provision the entire Proxmox Game of Active Directory lab environment using Terraform