

DPU TEST DRIVE LAB BUILDOUT PROCESS

DRAFT

Contents

Doc version2

 Current Testing Gaps2

 Limitations2

Equipment Requirements.....2

 Networking2

 Compute.....3

 Core Infrastructure Setup Requirements.....4

Lab Infrastructure Deployment Instructions.....5

 Phase 1: Install ELK Stack5

 Phase 2: Base Setup and Tool Deployment.....5

 VMware Deployment Variables Setup7

VMware Deployment Cleanup Scripts 11

 Clean Script 1.....12

 Clean Script 2.....12

Phase 3: PSM Configuration Build Process12

 PSM Build Phases.....12

PSM Deployment Cleanup Scripts15

Doc version

0.1	Toby Makepeace	Initial Draft	3/3/2025
0.2	Don Zaino	Structure Edits	4/7/2025
0.3	DZ	PSM Build	4/13/2025
0.4	DZ	Add manage script	4/21/2025
0.5	DZ	Consolidation of tasks	
0.6	DZ	Consolidation of tasks	4/22/2025
0.7	DZ	New git repo link	4/23/2025
0.8	DZ	Mange script additions	4/24/2024

Current Testing Gaps

- **PSM concurrency:** Performance and stability under multiple simultaneous users have not been validated.
- **CX 10000 multi-user interaction:** Behavior and reliability with multiple concurrent users remain untested.

Limitations

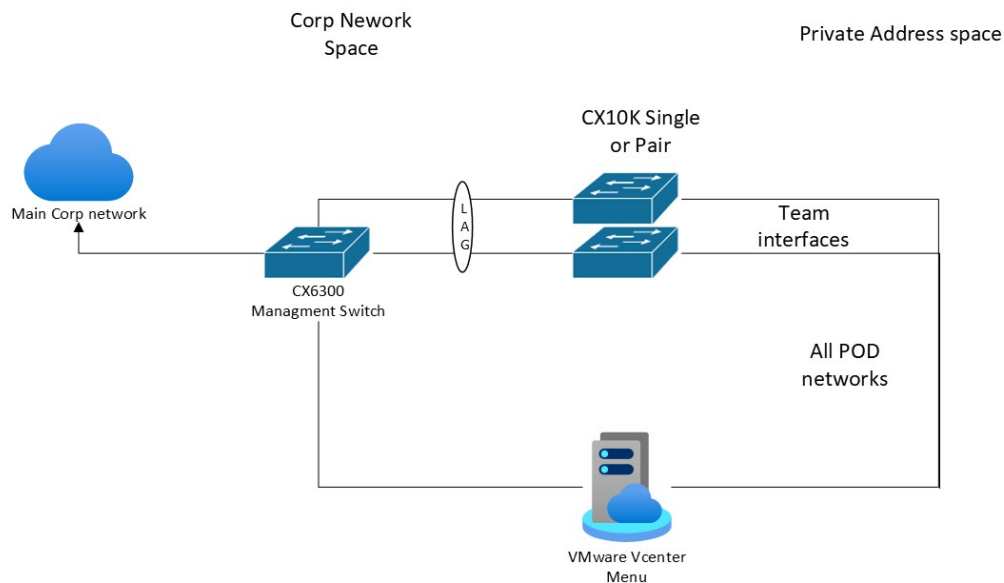
- **Shared environment:** A single user can affect the experience or stability for others during their session. Lab access is based on mutual trust.
- **Recovery process:** A full lab rebuild can be triggered if needed, with an expected recovery time of under 45 minutes.

Equipment Requirements

These are tested configurations proven in existing lab environments.

Networking

- **2 x Aruba CX 10000 Switches**
Core switching platform for data center fabric and DPU-based security services.
- **1 x Aruba CX 6300 or equivalent (1G copper switch)**
Provides Out-of-Band (OOB) connectivity for:
 - iLO management interfaces
 - VMware ESXi management interfaces
 - CX 10000 management ports
 - Service workloads



Compute

- **VMware ESXi Host**

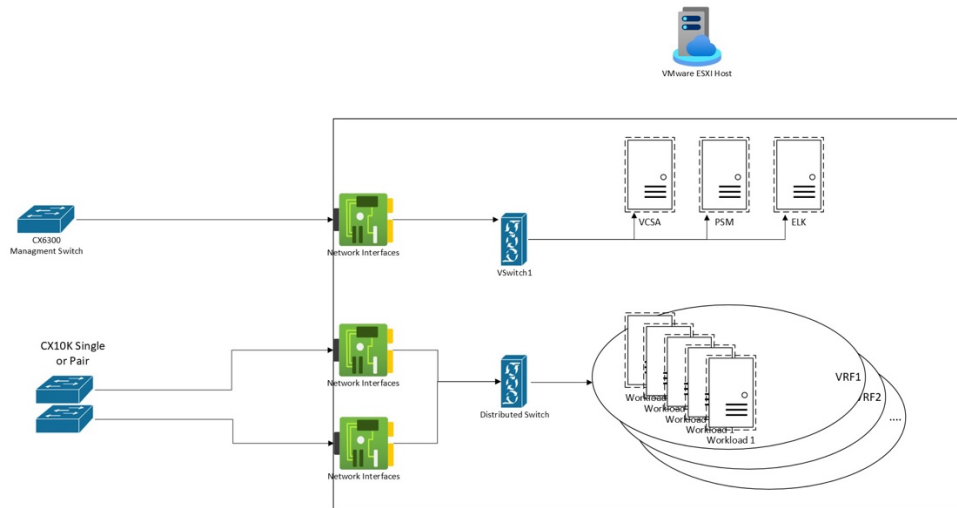
Host 1 – For lab workloads, virtual desktops, and toolsets:

- vCenter
- Pensando Services Manager (PSM)
- ELK Stack (Elasticsearch, Logstash, Kibana)

Recommended Specs:

- 64+ CPU cores
- 128 GB RAM
- 2 TB SSD/NVMe
- Networking: 4 x 1G, 2 x 10/25G

Will run on less compute resources but not recommended



Core Infrastructure Setup Requirements

The infrastructure deployment is performed in two phases using scripts available on GitHub.

Pre-Deployment Checklist

Before you begin, make sure your vCenter and ESXi hosts are fully operational with a standard setup. In vCenter, confirm the following:

- A Datacenter is defined.
- An accessible Datastore is available.
- A VM image is present and ready to be cloned.

You will also need an **Ubuntu workstation (20.04, 22.04, or 24.04)** with a static IP and sufficient CPU, RAM, and disk resources to support the installation of required applications.

A lightweight Linux VM image is required for cloning workloads during the lab setup. The recommended image is TinyVM, which is optimized for minimal resource consumption and quick deployment.

You can download the TinyVM image from either of the following sources:

- **From the Internal repository:**
/pensandotools/PSM_Test_Drive_Light/VMimage\$
- **Public GitHub repository:**
[TinyVM Download](#)

Note: The TinyVM must be configured as “**Other Linux 3.x or later (32-bit)**” in vSphere to ensure compatibility and power on.

Lab Infrastructure Deployment Instructions

Phase 1: Install ELK Stack

Installs the ELK stack on your Ubuntu VM

1. Deploy Ubuntu 20.04/22.04/24.04 VM with at least:
2 vCPU / 16 GB RAM / 96 GB Disk
2. From the terminal run the following command:

```
wget -O ELK_Install_Ubuntu_script.sh  
https://raw.githubusercontent.com/tdmakepeace/ELK\_Single\_script/refs/heads/main/ELK\_Install\_Ubuntu\_script.sh  
chmod +x ELK_Install_Ubuntu_script.sh  
./ELK_Install_Ubuntu_script.sh
```

3. Select **Option B** – installs dependencies (requires reboot)

Run script again # `./ELK_Install_Ubuntu_script.sh`

4. After reboot, run again with **Option E** – installs ELK, log forwarder, dashboards.

Choose the Branch Main (not *main) or the one that matches your CX10K Firmware version

Do you want to install Elastiflow license = Select No
Review and press enter

ELK services are now installed successfully.

Phase 2: Base Setup and Tool Deployment

Clones the GIT Test Drive Repo and installs Powershell and modules for your environment

The setup menu includes several options, but we will focus only on the those intended to be executed once during the initial configuration.

- E** - Setup the environmental variables. (directory, PSM details, Axis Details (if required))
- D** - Download and clone the Gilt repo for the test-drive.
- B** - Base setup of the Powershell environment.
- I** - Install the modules required in Powershell.
- R** - will allow you to review all the available tasks.

Options for this setup need to be followed one by one :

E - Setup the enviromental variables. (directory, PSM details, Axis Details)
D - Download and clone the Git repo for the testdrive.
B - Base setup of the powershell enviroment.
I - Install the modules required in powershell.

Other options:

C - Change the enviromental variables. (directory, PSM details, Axis Details)
G - Refesh the gitrepo

R - Read notes on using the VMware scripts manually.
P - Read notes on using the PSM scripts manually.

x - to exit

E or D or B or I
C or G or R or P

From the terminal of your ubuntu host run the following command:

```
wget -O TestDrive_Install_script.sh  
https://raw.githubusercontent.com/tdmakepeace/DPU\_TestDrive/refs/heads/main/TestDrive\_Install\_script.sh && chmod +x TestDrive_Install_script.sh && ./TestDrive_Install_script.sh
```

Step 1- Select Option E - Input your specific values (example in **BOLD**):

- (GITREPO) - https://github.com/tdmakepeace/DPU_TestDrive.git
- (Root Folder) - **Pensandotools**
- (path to directory) - **DPU_TestDrive**
- (IP of your PSM) - **10.9.20.84**
- (API Username of PSM) - **admin**
- (API Password of PSM) - **password**
- (AXIS PSM API Key) - **can be ignored unless setting up RBAC -hit enter**
- (AXIS Workgroup) - **can be ignored unless setting up RBAC- hit enter**
- (Review is this interface & IP you want to use) - **YES**
- (Review your inputs) - **YES**

*Log out and log back in to workstation

You can verify your inputs of option **E** at any time by running the following command
#cat [.bashrc.local](#)

Run script again # [./TestDrive_Install_script.sh](#)

Step 2 - Select Option D – Follow the prompts and select main branch #3

Step 3 - Select **Option B – Follow the prompts**

Step 4 - Select **Option I - PowerShell Prep**

(Open on 2nd terminal window)

In the 2nd terminal window - Enter the PowerShell prompt:
#pwsh

From the original terminal session copy and run the following commands into the new 2nd window:

```
cd /pensandotools/DPU_TestDrive/ESX/  
Install-Module -Name VMware.PowerCLI -Confirm:$false  
(select option (A))
```

```
cd "PowerCLI-Example-Scripts/Modules/VMware.vSphere.SsoAdmin"  
Import-Module ./VMware.vSphere.SsoAdmin.psd1  
cd ../../../../
```

Go back to your 1st terminal window – Hit enter. The scripts are now ready for the DPU TestDrive

VMware Deployment Variables Setup

This will walk you through setting up your environment variables for the VMware portion

Step 5 - From your Ubuntu terminal: in the Powershell prompt

#pwsh

From the directory `#!/pensandotools/DPU_TestDrive/ESX`

Run script # `./BuildVariables.ps1`

Input your specific values (example in **BOLD**):

- vCenter IP: **10.9.X.X**
- Admin login: **administrator@penlabs.net**
- Password: **password**
- ESXi host IP: **10.9.X.X**
- Clone VM name: **TinyVMDeploy** = *(name of vm you want cloned)*
- VMware disk = Datastore: **datastore-1**
- VMware domain = your DC Name: **pod10-vcsa**
- DVS Name: **Demo_dvs**

- Tag Categories for workloads: **Demo**
- Tag Categories for VRF: **VRFs**
- Number of VRFs: **3**
- Workloads per VRF: **3**

**Currently you cannot create a vm from a template from power-shell cli

You may verify with: `#more TestDrive.ps1`

You can now go back to your 1st terminal window and Hit enter

This will drop you into the manage script menu, if you need to access this menu at any time:

From your Ubuntu terminal:

`#./manage_scripts.sh`

Moving forward you will now manage all environments using [./manage_scripts.sh](#) from the `#!/pensandotools/DPU_TestDrive/accounts directory`.

This provides a single interface for managing the entire lab environment—centralized control for rebuilds, administration tasks, and ongoing operations.

Single funtion scripts		
Axis accounts	-	a
Create / Delete / Update Expiry		
ESX single pod reset	-	b
Reset a single POD VM's		
PSM single pod reset	-	c
Reset a single POD PSM		
Single Password reset	-	d
After a full lab sessions		
Full ESX pods reset	-	e
Full PSM Reset	-	f
Recreate all default Passwords	-	g
Recreate all random Passwords	-	h
Combined Scripts		
Delete expired user and reset pod (interactive option, by email)	-	m
Delete expired user and reset pod (by pod numbers)	-	n
Initial Scripts		
Setup and build the first ESX config (networking/users/vrf/pods, etc - one off process)	-	o
Setup and build the first PSM config (networking/users/vrf/pods, etc - one off process)	-	p

From the script menu:

- Select **O** to build the VMware environment.

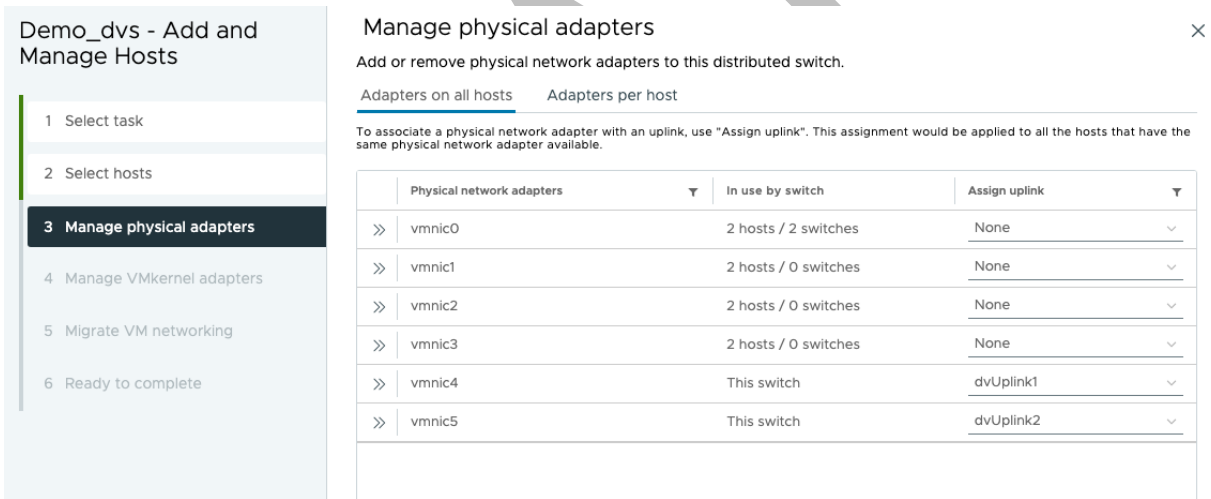
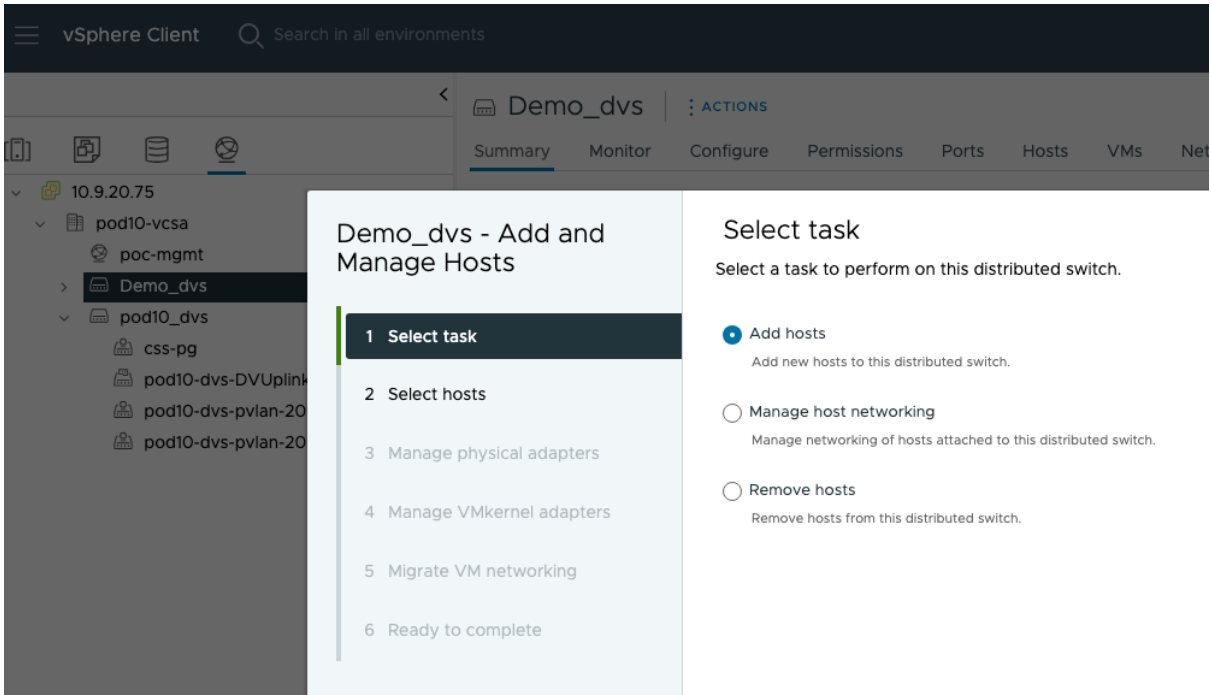
⚠ You may see error-like messages during the process—these are normal and indicate that the build is running in the background.

**** Important:**

After initiating **option O**, you must manually assign physical uplinks to the distributed switch (DVS). It is recommended to use two interfaces - one connected to each CX 10000 switch for redundancy and traffic distribution.

Follow these steps to assign uplinks:

1. In the vSphere client, navigate to the **Distributed Switch**, then select **Add Hosts**.
2. Choose your **ESXi host**, then assign the appropriate physical NICs.
3. Accept the default settings to complete the process.



After this task is complete return to the terminal and hit enter an C to confirm the rest of the vSphere buildout

After successfully running Option **O**, this should have now created everything needed from the vSphere side.

Explanation of individual VMware Build Scripts for reference:

ESXI-testdrive-Build1

The ESXI-testdrive-Build1.ps1 script configures the foundational networking within the VMware environment.

ESXI-testdrive-Build2

The ESXI-testdrive-Build2.ps1 script configures the Virtual machines and other objects within the VMware environment.

The ESXI-testdrive-Build2.ps1 script **runs after**:

- ESXI-testdrive-Build1.ps1 has completed successfully
- Uplinks have been manually assigned to the Distributed Virtual Switch (DVS)

VMware Deployment Cleanup Scripts

From your Ubuntu terminal:

```
#./manage_scripts.sh
```

From the script menu:

- Select **b** to reset your ESXi pods

Note:

In some cases, the TESTDRIVE role may be removed, but associated VRF_userX accounts may persist.

You can manually delete these user accounts in vSphere by navigating to:
Administration → Users and Groups.

Explanation of individual VMware Cleanup Scripts for reference:

If there is ever a reason that it is needed to remove **& destroy** all configuration you can run the below scripts. These are in the *folder /pensandotools/DPU_TestDrive/ESX*

Clean script 1 deletes VMs

The ESXI-testdrive-clean1.ps1 script will **remove** the VM resource groups, workloads, and permissions, for the pods. It will take about 3 minutes per pod assigned.

Clean script 2 Deletes DVS, Resource pool

The ESXI-testdrive-clean2.ps1 script will remove the VM users, and distributed switching. It will take about 2 minutes to run.

Clean Script 1

```
# ./ESXI-testdrive-clean1.ps1
```

Clean Script 2

```
# ./ESXI-testdrive-clean2.ps1
```

Phase 3: PSM Configuration Build Process

This section outlines the tasks required to set up and integrate the Pensado Services Manager (PSM) with the CX 10000 switches for demonstration or customer environments.

You can download the PSM OVA image from the HPE Networking Support Portal:

[Download PSM OVA](#)

(Login required.)

PSM Build Phases

Task 1: PSM Deployment

- Begin by deploying one to three PSM instances via the OVA template on your existing vSphere environment.
- Assign appropriate compute and network resources recommended in the sizing guide.
- Configure basic network settings for each node (management IP, gateway, DNS).
- Complete the bootstrap process for each PSM node after initial power-on.
- For detailed steps on bootstrap and OVA deployment on ESXi, refer to the **PSM User Guide**:

[PSM Install Guide](#)

Task 2: CX 10000 Registration

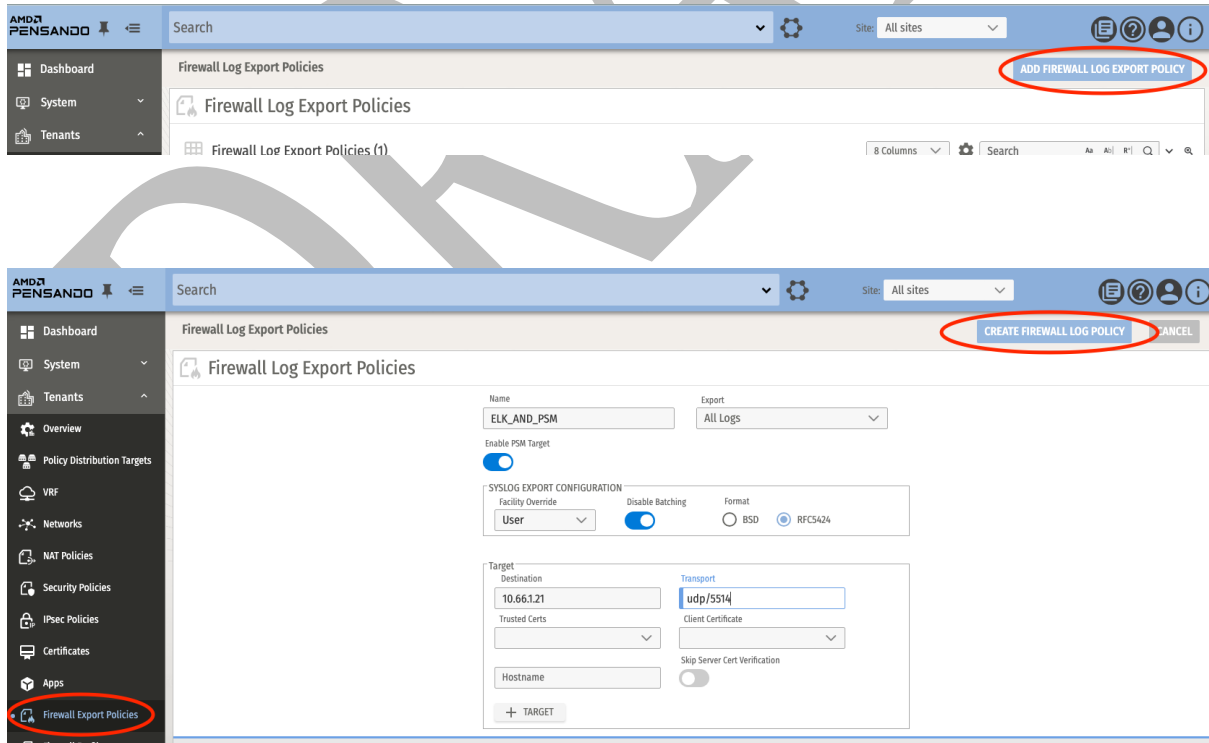
- Register the CX 10000 switches with the PSM instance.
- Ensure secure communication between devices and PSM is established.
- Refer to guide below for reference

Register switches with PSM

Task 3: PSM Configure global options for Firewall Export Policies

From the PSM GUI, Under the Tenants section go to the firewall export polices section:

1. Select Firewall Export policies from the menu.
2. Select ADD firewall export policy
3. You need to provide a “Name” for the policy, set export to All Logs, Enable PSM Target slider, Disable Batching, select format RFC5424, input your destination (IP of destination collector) and the Protocol /Port number required (udp/5514) * see figure below for **example**.



Task 4: Script deployment for PSM configuration

After deploying and bootstrapping the PSM nodes, the next step is to configure them using the provided automation scripts. These steps outline how to use the provided setup and cleanup scripts for managing PSM pods.

This phase assumes:


- CX switches are already configured, reachable and registered with PSM instance.
- PSM nodes are accessible and fully initialized.
- You have command-line access to your Ubuntu-based jump host.

Within the `/pensandotools/DPU_Test_Drive/PSM/PythonScripts` folder, you'll find the individual Python scripts used for provisioning and cleanup. These were bundled and installed from the GIT clone during the initial infrastructure set up. These are invoked by the higher-level shell scripts to automate the entire lifecycle of pod creation and teardown.

You can manage all environments using `./manage_scripts.sh` from the `#!/pensandotools/DPU_TestDrive/accounts` directory.

From the script menu:

- Select **P** to build the PSM environment.

 You may see error-like messages during the process—these are normal and indicate that the build is running in the background.

Explanation of individual PSM Scripts for reference:

`build.sh` ->

takes the contents of the CSV files and builds the JSON and TXT payloads. It then runs the json files against the PSM to build the objects. if the objects exist they will return 404 or 414 errors.

`clean.sh` ->

takes the contents of the CSV files and builds the JSON and TXT payloads. It then runs the text files against the PSM to delete the objects. if the objects do not exist they will return 404 or 414 errors.

`rebuild.sh` ->

takes the tasks of both the clean and build scripts and runs them to reset. all the pods.

If there is ever a reason that it is needed to remove & Destroy all PSM configuration you can run the [clean.sh](#) located in the *folder /pensandotools/ DPU_Test_Drive /PSM*

2. (Optional) Edit scripts

Example: If needed, modify login credentials in:

```
/pensandotools/ DPU_Test_Drive /PSM/PythonScripts /logindetails.py
```

Tip: If vi doesn't work, install Vim with: `#sudo apt-get install vim`

Update the IPs, usernames, and passwords to match your lab setup.

If any file needs changes, you can edit it directly in place. Just update the necessary parameters in that specific file. This applies to any script or config file that was installed to match your lab setup.

3. (Optional) Configure input variables via CSV

Go to the CSV_example directory and review the .csv files. These define parameters for your pod build—VRFs, workloads, policies, etc.

Default CSVs are lab-ready with minimal edits. If needed:

- Modify the CSV files manually.
- Upload them to the Ubuntu server running the Python scripts at:

```
/pensandotools/DPU_TestDrive/PSM/CSV_example
```

PSM Deployment Cleanup Scripts

From your Ubuntu terminal:

```
#./manage_scripts.sh
```

From the script menu:

- Select **f** to reset your PSM environment

Explanation of additional individual PSM Scripts for reference:

Each shell script calls multiple Python modules to complete tasks like:

- VRF creation
- Workload provisioning
- Role assignment
- Policy creation

You can reuse these scripts to reset or reconfigure pods as needed.

Virtual environment

```
#Make sure your ubuntu host has latest packages
sudo apt-get update
```

```
#Enter the Virtual Environment
python3 -m venv .venv
. .venv/bin/activate
pip install -U pip
pip install -r requirements.txt
```

and / or depending on release

```
python3 -m venv .venv
. .venv/bin/activate
pip install -U pip
python -m pip install requests
```

```
*#deactivate gets you out of your VENV
```

PREP Scripts

```
clean.sh
python3 networks.py ../CSV_example/networks.csv
python3 role.py ../CSV_example/role.csv
python3 user.py ../CSV_example/user.csv
python3 vrf.py ../CSV_example/vrf.csv
python3 policy_FIRST.py ../CSV_example/policy_FIRST.csv
python3 policy_NEW.py ../CSV_example/policy_NEW.csv
python3 workloadgroup.py ../CSV_example/workloadgroup.csv
```

IMPORT Scripts

This will Convert the CSVs to JSON

```
python3 importpolicy.py policy_FIRST_csv.json
python3 importpolicy.py policy_NEW_csv.json
python3 importvrf.py vrf_csv.json
python3 importnetworks.py networks_csv.json
python3 importuser.py user_csv.json
python3 importrole.py role_csv.json
python3 importrolebinding.py rolebinding_csv.json
python3 importworkloadgroup.py workloadgroup_csv.json
```

DELETE Scripts (if needed)

```
python3 deleterolebinding.py rolebinding_list.txt
python3 deleterole.py role_list.txt
python3 deleteuser.py user_list.txt
python3 deletenetworks.py networks_list.txt
python3 deletevrf.py vrf_list.txt
python3 deletepolicy.py policy_FIRST_list.txt
python3 deletepolicy.py policy_NEW_list.txt
python3 deleteworkloadgroup.py workloadgroup_list.txt
clean.sh
```