

# Setup and Install Docker in a Proxmox LXC Container

This guide is for Proxmox VE 6.4.4 or earlier and will NOT work well on version 7. See [this guide](#) for 7.

[You should check out my website at noted.lol!](#)

Containers are a lightweight alternative to fully virtualized machines (VMs). They use the kernel of the host system that they run on, instead of emulating a full operating system (OS). This means that containers can access resources on the host system directly.

The runtime costs for containers is low, usually negligible. However, there are some drawbacks that need be considered:

- Only Linux distributions can be run in Proxmox Containers. It is not possible to run other operating systems like, for example, FreeBSD or Microsoft Windows inside a container.
- For security reasons, access to host resources needs to be restricted. Therefore, containers run in their own separate namespaces. Additionally some syscalls (user space requests to the Linux kernel) are not allowed within containers.

Proxmox VE uses [Linux Containers \(LXC\)](#) as its underlying container technology. The “Proxmox Container Toolkit” (`pct`) simplifies the usage and management of LXC, by providing an interface that abstracts complex tasks.

Containers are tightly integrated with Proxmox VE. This means that they are aware of the cluster setup, and they can use the same network and storage resources as virtual machines. You can also use the Proxmox VE firewall, or manage containers using the HA framework.

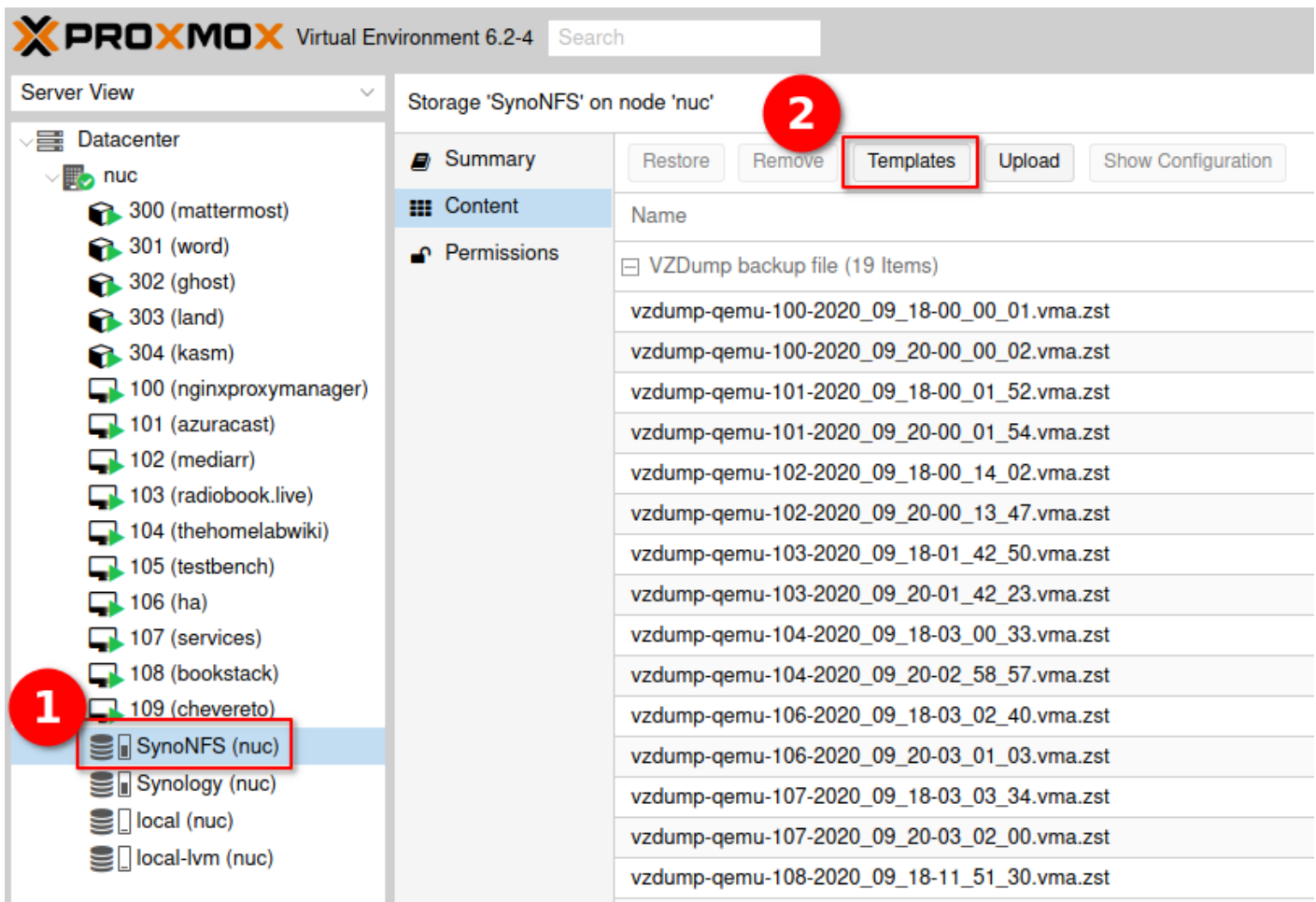
Our primary goal is to offer an environment that provides the benefits of using a VM, but without the additional overhead. This means that Proxmox Containers can be categorized as “System Containers”, rather than “Application Containers”.

The above was cited from the Proxmox VE website.

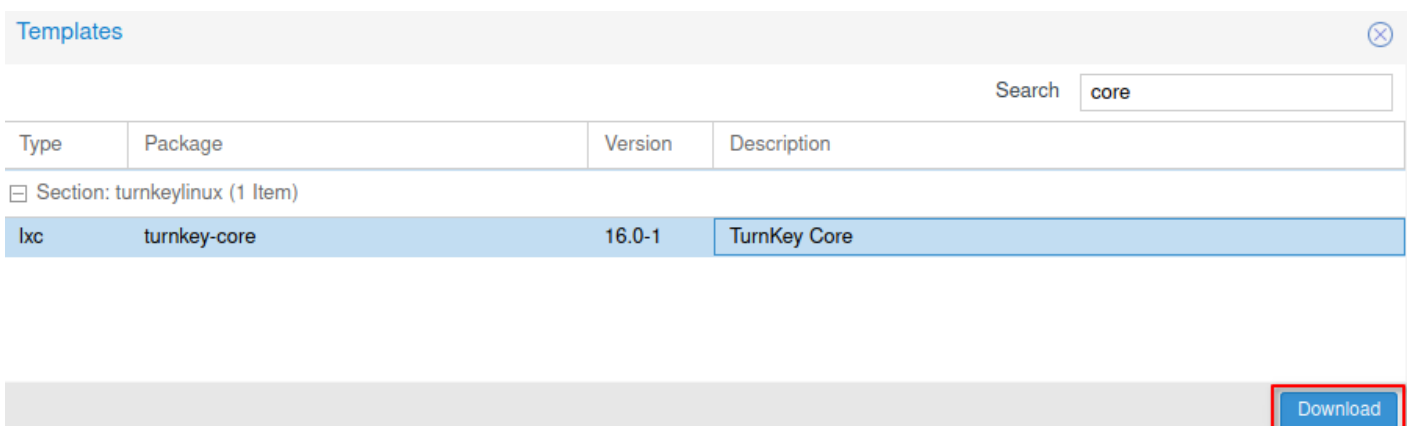
# Download the Turnkey Core template

To setup and install Docker in a Proxmox LXC Container, you will have to download the Turnkey Core template to your storage.

1. Click on your storage then click on the templates button.

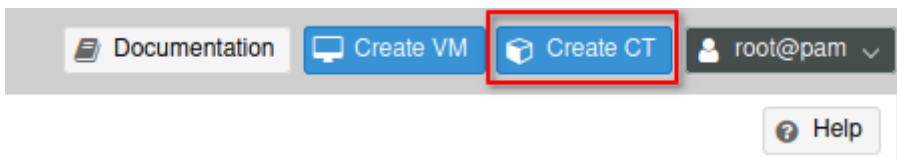


2. Search for the core template and click download.



Type	Package	Version	Description
Section: turnkeylinux (1 Item)			
lxc	turnkey-core	16.0-1	TurnKey Core

3. When it's done you can click the "Create CT" button in the upper right corner.



4. Set your hostname and password and be sure to UNCHECK "Unprivileged container" or this will not work. Now move on to the Template tab.

The screenshot shows the 'Create: LXC Container' form in the 'General' tab. The form has several input fields: 'Node' (set to 'nuc'), 'CT ID' (set to '110'), 'Hostname' (set to 'docker'), 'Resource Pool' (empty), 'Password' (masked with dots), 'Confirm password' (masked with dots), and 'SSH public key' (empty). There is a checkbox labeled 'Unprivileged container' which is unchecked and highlighted with a red rectangular box. A 'Load SSH Key File' button is located below the SSH public key field. The form has tabs for 'General', 'Template', 'Root Disk', 'CPU', 'Memory', 'Network', 'DNS', and 'Confirm'. A close button (X) is in the top right corner.The screenshot shows the bottom navigation bar of the form. It includes a 'Help' button with a question mark icon, an 'Advanced' checkbox which is checked, and 'Back' and 'Next' buttons.

5. Select the storage where you downloaded the Turnkey Core template then choose the template.

The screenshot shows the 'Create: LXC Container' form in the 'Template' tab. The form has two input fields: 'Storage' (set to 'SynoNFS') and 'Template' (set to 'debian-10-turnkey-core\_16.0-1\_i'). Both fields are highlighted with a blue rectangular box. The form has tabs for 'General', 'Template', 'Root Disk', 'CPU', 'Memory', 'Network', 'DNS', and 'Confirm'. A close button (X) is in the top right corner.

6. In the Storage tab choose where you want the container to use storage. For the disk size I recommend at-least 20gb of space. This depends on what you plan on doing with the container. For example if you plan on downloading a bunch of files, you may want a larger size disk.

Create: LXC Container ⓧ

General Template **Root Disk** CPU Memory Network DNS Confirm

Storage: SynoNFS

Disk size (GiB): 20

---

Enable quota: ☐ ACLs: Default

Mount options: Skip replication: ☐

7. On the CPU tab, I usually put 4 cores but this is overkill for a container. This depends on the processor you have in your system.

Create: LXC Container ⓧ

General Template Root Disk **CPU** Memory Network DNS Confirm

Cores: 4

---

CPU limit: unlimited CPU units: 1024

8. On the Memory tab I usually allocate 2gb for containers.

Create: LXC Container ⓧ

General Template Root Disk CPU **Memory** Network DNS Confirm

Memory (MiB): 1908

Swap (MiB): 1000

9. On the Network tab I let the DHCP host assign the IP automatically.

Create: LXC Container

GeneralTemplateRoot DiskCPUMemoryNetworkDNSConfirm

Name:eth0

MAC address:auto

Bridge:vmbr0

VLAN Tag:no VLAN

Rate limit (MB/s):unlimited

Firewall:☒

IPv4: ☐ Static ☒ DHCP

IPv4/CIDR:

Gateway (IPv4):

IPv6: ☐ Static ☒ DHCP ☐ SLAAC

IPv6/CIDR:

Gateway (IPv6):

10. On the DNS tab I just use my gateway IP.

Create: LXC Container

GeneralTemplateRoot DiskCPUMemoryNetworkDNSConfirm

DNS domain:192.168.1.1

DNS servers:192.168.1.1

11. Confirm your settings and make sure you DO NOT tick "Start after created". Click finish and let the container create.

## Create: LXC Container



General

Template

Root Disk

CPU

Memory

Network

DNS

Confirm

Key ↑	Value
cores	4
hostname	docker
memory	1908
nameserver	192.168.1.1
net0	bridge=vmbbr0,name=eth0,ip=dhcp,ip6=dhcp,firewall=1
nodename	nuc
ostemplate	SynoNFS:vztmpl/debian-10-turnkey-core_16.0-1_amd64.tar.gz
pool	
rootfs	SynoNFS:20
searchdomain	192.168.1.1
swap	1000
vmid	110

☐ Start after createdAdvanced ☒

Back

Finish

12. Once done, you will see "TASK OK". Click the X in the upper corner and proceed to the next step.

## Task viewer: CT 110 - Create



Output

Status

Stop

```
Allocating group tables: 0/160 done
Writing inode tables: 0/160 done
Creating journal (32768 blocks): done
Multiple mount protection is enabled with update interval 5 seconds.
Writing superblocks and filesystem accounting information: 0/160 done

extracting archive '/mnt/pve/SynoNFS/template/cache/debian-10-turnkey-core_16.0-1_amd64.tar.gz'
Total bytes read: 506183680 (483MiB, 153MiB/s)
Detected container architecture: amd64
Creating SSH host key 'ssh_host_ed25519_key' - this may take some time ...
done: SHA256:oR+Ar4v7TNeonHo2Z6eOXVsFQL9bUSddx+271RfJqbl root@docker
Creating SSH host key 'ssh_host_dsa_key' - this may take some time ...
done: SHA256:6AmQfqBueqZrsQOhe/Nle5oULaD23bmDPO61Ds69jwc root@docker
Creating SSH host key 'ssh_host_ecdsa_key' - this may take some time ...
done: SHA256:AFZh2X+6b04ZKtieiz2bc///TsD8IY3isHyV7QQMxpY root@docker
Creating SSH host key 'ssh_host_rsa_key' - this may take some time ...
done: SHA256:oX9YalKvYHAWpAXAuunl87D9BnDgyOyus5WIGRm6TaM root@docker
TASK OK
```

13. Select the container and then click on Options then double click on Features.

PROXMOX Virtual Environment 6.2-4 Search

Server View

Datacenter

- nuc
  - 110 (docker)
  - 300 (mattermost)
  - 301 (word)
  - 302 (ghost)
  - 303 (land)
  - 304 (kasm)
  - 100 (nginxproxymanager)
  - 101 (azuracast)
  - 102 (mediarr)
  - 103 (radiobook.live)
  - 104 (thelabwiki)
  - 105 (testbench)

Container 110 (docker) on node 'nuc'

Summary Console Resources Network DNS Options Task History Backup Replication Snapshots

Edit Revert

Start at boot	No
Start/Shutdown order	order=any
OS Type	debian
Architecture	amd64
/dev/console	Enabled
TTY count	2
Console mode	tty
Protection	No
Unprivileged container	No
Features	none

14. Tick Nesting then click ok.

Edit: Features

keyctl: ☐ unprivileged only

Nesting: ☒

NFS: ☐

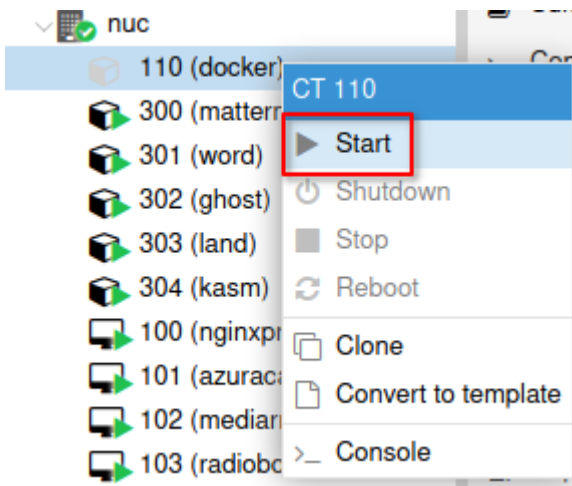
CIFS: ☐

FUSE: ☐

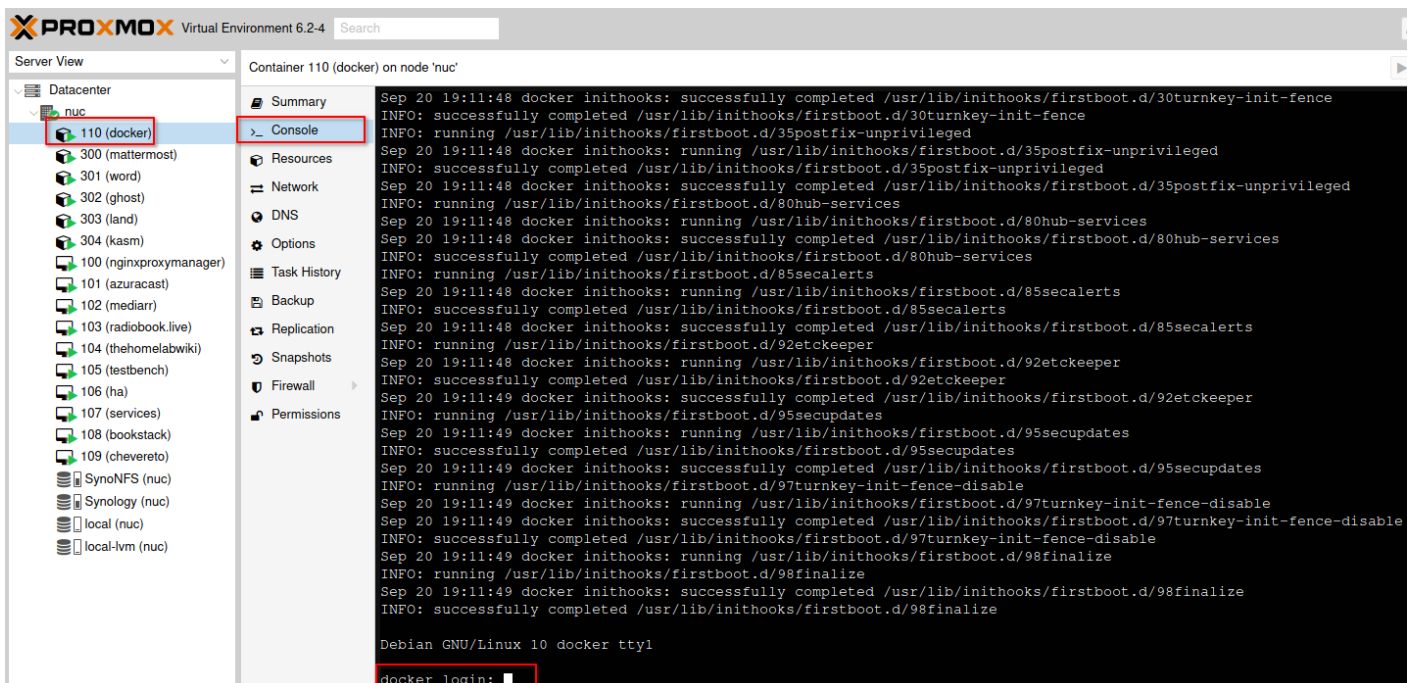
Create Device Nodes: ☐ Experimental

OK Reset

15. Now you can start your container and view the console.



This will start the container and run the install script then assign an IP to your container. Once done you will need to login using root and the password you chose when you created the container in step 4.



16. Now we have to go through the Turnkey setup. Skip the API entry.



Initialize Hub services

1) TurnKey Backup and Migration: saves changes to files, databases and package management to encrypted storage which servers can be automatically restored from.  
<https://www.turnkeylinux.org/tklbam>

2) TurnKey Domain Management and Dynamic DNS:  
<https://www.turnkeylinux.org/dns>

You can start using these services immediately if you initialize now. Or you can do this manually later (e.g., from the command line / Webmin)

API Key: (see <https://hub.turnkeylinux.org/profile>)

17. Skip the notifications.

System Notifications and Critical Security Alerts

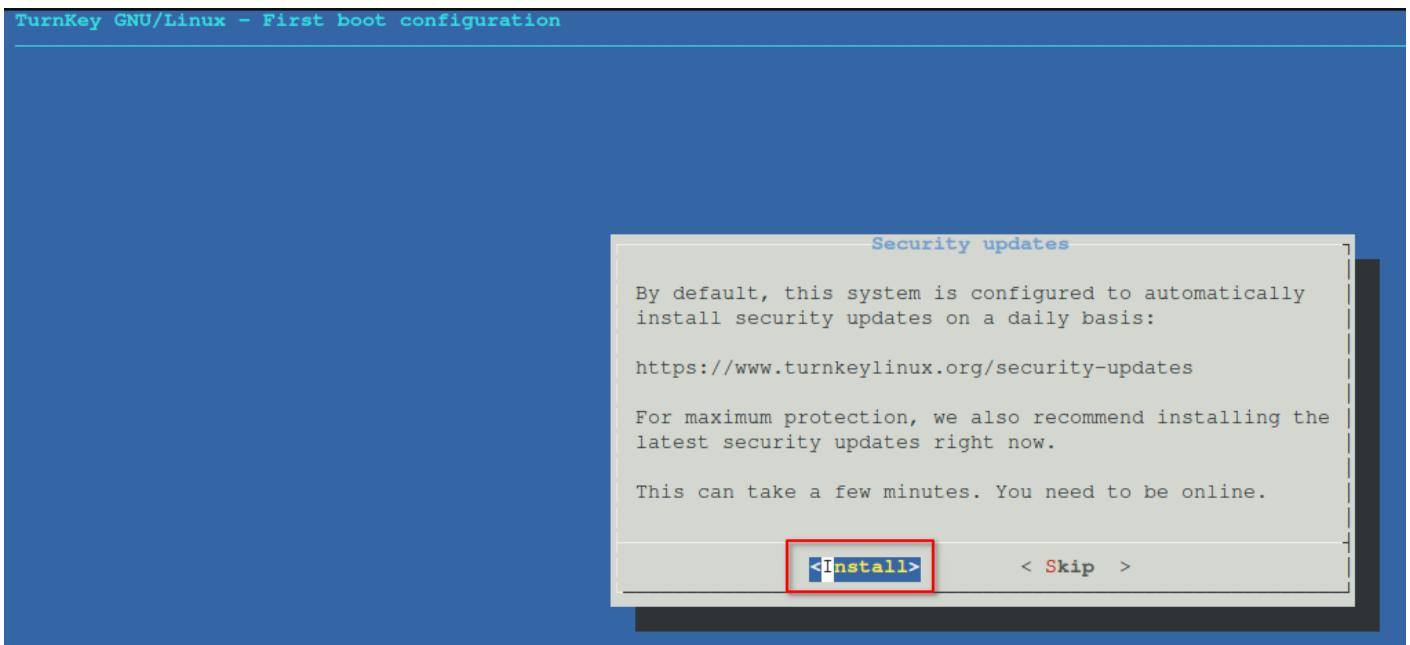
Enable local system notifications (root@localhost) to be forwarded to your regular inbox. Notifications include security updates and system messages.

You will also be subscribed to receive critical security and bug alerts through a low-traffic Security and News announcements newsletter. You can unsubscribe at any time.

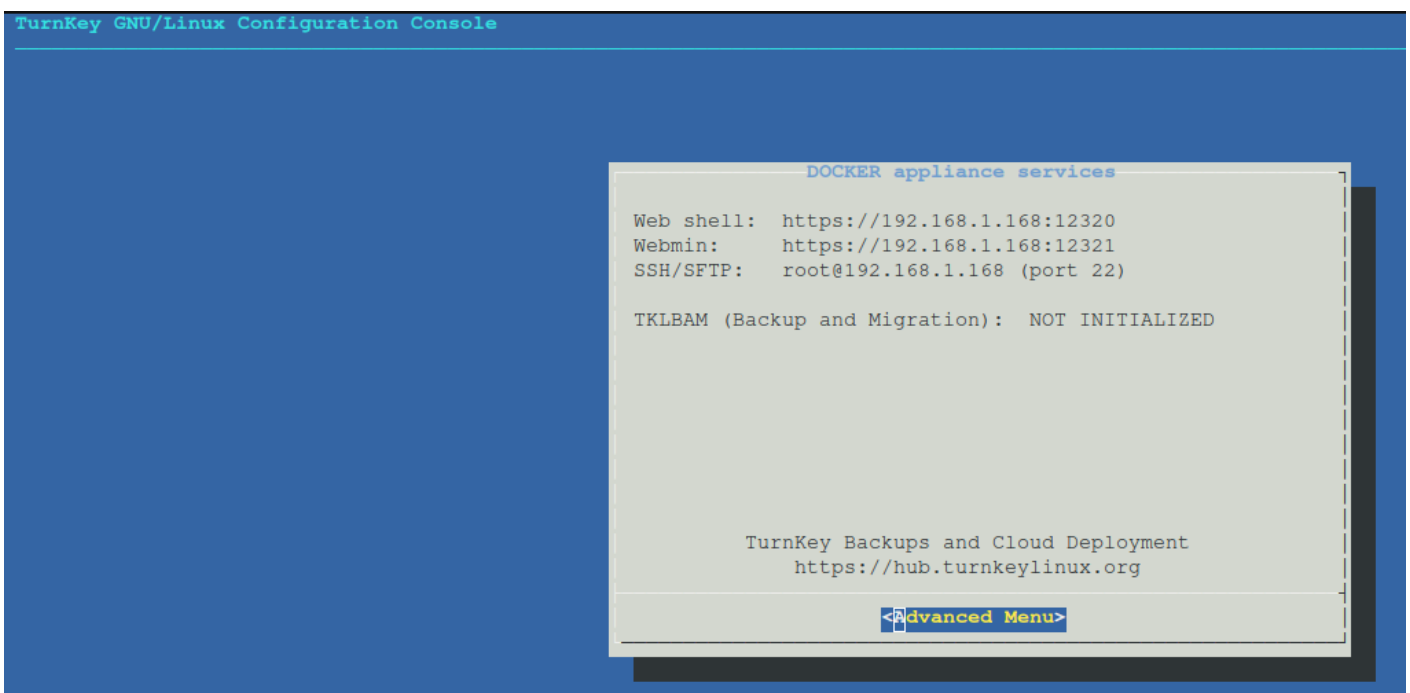
<https://www.turnkeylinux.org/security-alerts>

Email:

18. Install security updates.



This will take you to the final screen and will show you what the access IP and ports are for the appliance services. It's important to make a note of these incase you need them for future use.



19. Press CTRL+C to clear the console and now you can update the system by typing the commands below. Since we only use root in Proxmox, you do not need to use sudo for any commands through the console.

```
apt-get update
```

then

```
apt-get upgrade
```

# Install Docker on your Promox Container

Run the following commands one at a time.

```
apt install docker.io
```

```
systemctl enable docker
```

```
systemctl start docker
```

Now make sure Docker is running

```
systemctl status docker
```

## Install Portainer

This is completely optional but I wanted to add this in here just in-case you wanted a Docker UI. Run the following command to install Portainer 2.0

```
docker run -d \  
--name="portainer" \  
--restart on-failure \  
-p 9000:9000 \  
-v /var/run/docker.sock:/var/run/docker.sock \  
-v portainer_data:/data \  
portainer/portainer-ce
```

Once completed, visit the IP of your container and port 9000. Example: 192.168.1.168:9000.

If you forgot the IP of your container, open the console and type

```
ip addr
```

Press enter and find your container IP. It's usually towards the top.

Did you find this helpful? [Subscribe to me on Youtube](#) for more content!