Install Guide for frigate and Home Assistant on Proxmox 7.3

Preamble

This is my best recollection of how I completed the installation of Frigate and home assistant on Proxmox. Unfortunately, there are many steps to the install, so please take care to follow them carefully and be wary that I may have documented some steps poorly.

Some of the following material I have created, but most of it is copied and pasted and edited from a variety of sources that I believe to be reliable. Use at your own risk but I have tried to be as accurate and clear as possible for the modestly familiar user.

Before you start, here are some references that might be helpful:

- Running Frigate in Docker in Proxmox LXC with remote nas Share (cifs) · Discussion #1111 · blakeblackshear/frigate (github.com)
- Get started with the M.2 or Mini PCIe Accelerator | Coral
- https://github.com/blakeblackshear/frigate/discussions/1111#discussioncomment-2085971
- Google Coral USB + Frigate + PROXMOX - Third party integrations - Home Assistant Community (home-assistant.io)

Install Proxmox 7.x

I recommend running proxmox on a decent and recent intel platform. For the base machine, I have had good luck using a refurbished Dell 7060 with an 8th generation i7 processor and 32Gb of memory. For frigate's video storage I use an external NAS (Netgear ReadyNAS 314). I use 2 differently keyed Coral modules plugged into (with adapters purchased on Amazon) the NVMe m2 slot and wifi slot on the motherboard of the 7060.

1. Download the latest Proxmox 7.x ISO
2. Burn image to a usb stick using balenaetcher
3. Go into the base machine's BIOS and turn off secure boot
4. Install ProxMox
   Connect the bootable USB drive to the computer where you want to install Proxmox and boot it up. When the welcome screen comes up, select **Install Proxmox VE**, press Enter, and **agree** to the license agreement.
5. If you have several hard drives installed on that machine, select the drive that you want to install Proxmox. Click on **Next** and then set up your **Location** and **Time Zone**. On the next screen, set up a password to login to Proxmox and enter your email address. The email address is used for Proxmox to send alert notifications about backup failures and other events.
6. Lastly, you need to set up the **Network Configuration**. So, enter a **Hostname**, for example, pve.local or proxmox.local or something similar. Then, set up a static IP address (e.g., 10.0.0.100) so the Proxmox machine always uses the same IP address from the network. Click on **Next** and give it a minute for the installation to complete.
7. You are now done with the proxmox server and can continue with a browser on another machine. Login as root.
8. Open the browser UI at 10.0.0.xx:8006, select the proxmox server and navigate to Updates, then "Refresh", then "Upgrade". Its normal that some will fail as you don't have the enterprise license
   When the terminal window opens, help the process along by answering "Y"
9. Enable proxmox iommu feature. iommu – allows passthru of pcie stuff
   a. Follow directions here: Pci passthrough - Proxmox VE
   b. Don't forget to update grub after changing /etc/default/grub, update-grub
10. The pve-headers must be installed on the proxmox host before installing the coral drivers
    a. open: nano /etc/apt/sources.list
    b. add this line to the file (refer to Package Repositories - Proxmox VE):

       deb http://download.proxmox.com/debian/pve bullseye pve-no-subscription

       i. now save the file and run: apt-get update
       ii. run: apt install pve-headers-$(uname -r)
       iii. you should see that the headers are found and installed
11. Install dkms for the coral install to work– see this reference: How to build a kernel module with DKMS on Linux (xmodulo.com)

a. No need for “sudo” since you are root
b. Run: *apt-get install dkms --reinstall*

12. Install coral pcie drivers – install them into the proxmox host, not a vm, watch the command output carefully to be sure the install worked.
    a. Get started with the M.2 or Mini PCIe Accelerator | Coral
    b. Be sure to use this command to force a reinstall:

        sudo apt-get install gasket-dkms libedgetpu1-std --reinstall

        i.
    c. See: M2 EdgeTPU (Coral AI) Problem | Proxmox Support Forum
    d. Reboot!
13. Verify the coral drivers are working with:
    a. ls /dev/apex*

        you should get something like this: /dev/apex_0, if not, then stop here and figure out what is not working. I had to repeat the dkms install for some reason before I could get it to work. Also, don’t forget to reboot!

Install Docker in LXC Container

1. Watch this for reference:
   Docker in Proxmox V7 LXC with Turnkey Core - Lower Resources by 80% Compared to VMs - YouTube 183
2. Download Turnkey-core template
    1. From console, run “pveam update” to refresh the list of templates
    2. Go to proxmox storage (“local (pve)”) and go to “CT Templates”
    3. Search for “Core”, select “turnkey-core” and hit “download”
    4. I used 16.1.1 (“Debian-10-turnkey-core_16.1-1_amd64.tar.gz”)
    5. Click “CT Templates” again and verify it shows up in your list
3. Create LXC container

    1. Click “Create CT”
    2. General tab
        1. Hostname: “docker”
        2. Set password and note it for later user
        3. **Uncheck “unpriviledged container” (make it privileged)**
           (I think the container needs to be “priviledged” for the USB/PCI to pass through correctly.)
    3. Template tab
        1. Storage: local
        2. Template: choose the turnkey-core template
    4. Disk tab
        1. Select the rootfs and set to 128G (or whatever you want)
        2. Select the nas storage e.g., for me, mp0, and I set path to /media/hubfiles and size to 1000Gb. You can do this later if you want.
    5. CPU tab
        1. Cores: 4 (or whatever you want)
    6. Memory tab:
        1. Memory: 8196, swap: 2048 (or whatever you want)
    7. Network tab:
        1. IPv4: DHCP (I later assigned a fixed IP using my ASUS router)
    8. Accept all other defaults
    9. Click finish to initialize

4. Configure the new Container
    - Options->Features
        1. Enable “keyctl”
        2. Enable “Nesting”
        3. Enable cifs
    - Now install the Debian Turnkey Core
        - Start the container
          Open console for the container, and login with root + password noted earlier
          Skip first 2 prompts, then click “Install” to install security updates, Once finished,

CTRL+C to get out
Update & upgrade Debian:

```
apt update
apt upgrade
```

5. Install Docker in the LXC Container
   - Review this for reference: Install Docker Engine on Debian | Docker Documentation 113
   - And review this: Setup and Install Dock... | The Homelab Wiki 45

   - Run the below commands and refer to this: Install Docker Engine on Debian | Docker Documentation 113
     (Remove "sudo" from commands since you are already logged in as root)

```
apt-get install ca-certificates curl gnupg lsb-release

mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

- Install docker engine.

```
apt-get update

apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

- Verify Docker is running:

```
systemctl status docker

docker run hello-world
```

- Set docker's time zone

remove the symbolic link file (/etc/localtime):

rm -rf /etc/localtime

Identify the timezone you want to configure and create the symbolic link for it, for instance, I would like to set America/Chicago timezone:

ln -s /usr/share/zoneinfo/America/Chicago /etc/localtime

Now verify it by:

date

- Install Portainer (to simplify management of docker)
    1. Review this page here: Installing Docker and ... | The Homelab Wiki 44
    2. Run this code to install portainer on port 9000 and 8000

       docker run -d --name="portainer" --restart on-failure -p 9000:9000 -p 8000:8000 -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:latest

    3. Confirm container's IP
       ip addr
    4. Wait a few moments for portainer to start then open your browser and log in to portainer:
       http://ipaddress:9000
    5. Log in as "admin" and set the password

- Now setup cifs share with these commands:
    - Refer to: Expand Your Proxmox Storage: A Guide to Adding CIFS (SMB) Storage to Proxmox (engineerworkshop.com) and then this for the commands: Proxmox: Mounting CIFS Shares in Containers • Thushan Fernando
    - In the proxmox host:
        - Check to be sure you can see the shares: $ pvesm scan cifs nas_ip_address
        - You should get a listing of the shares
    - While the docker container is shutdown, in the proxmox host, add the mount to the host and then share with the container, e.g., 103, and the mnt, e.g., hubfiles:
        - Either from the gui add the volume or try this command:

          pvesm add cifs hubfiles --server 10.0.0.220 --share incoming [--username admin] [--password whatever] [--path /mnt/pve/hubfiles]

        - For some reason adding the share to the container from the gui does not work so I used this command:

```
pct set 103 -mp0 /mnt/pve/hubfiles,mp=/media/hubfiles
```

Almost there……Before starting the container, modify the container .conf file as follows (I lifted this and slightly changed it from the frigate forum, so thanks to the contributor):

Via the proxmox host shell, go to `/etc/pve/lxc` and edit the container file via `nano 10x.conf` (choose right number of LXC container.) Adjust as needed to match your mount point, coral devices, memory, etc.

(I have little knowledge for how this file is interpreted but for me it works 😊)

```
arch: amd64
cores: 4
features: mount=nfs;cifs,nesting=1
hostname: docker
memory: 8196
mp0: hubfiles:100/vm-100-disk-0.raw,mp=/media/hubfiles,backup=1,size=1000G
net0: name=eth0,bridge=vmbr0,firewall=1,hwaddr=0A:87:64:D6:C1:80,ip=dhcp,type=veth
ostype: debian
rootfs: local-lvm:vm-100-disk-0,size=128G
swap: 2048
lxc.cgroup2.devices.allow: c 226:0 rwm
lxc.cgroup2.devices.allow: c 226:128 rwm
lxc.cgroup2.devices.allow: c 29:0 rwm
lxc.cgroup2.devices.allow: c 189:* rwm
lxc.cgroup2.devices.allow: c 120:* rwm
lxc.cgroup2.devices.allow: a
lxc.mount.entry: /dev/dri/renderD128 dev/dri/renderD128 none bind,optional,create=file 0, 0
lxc.mount.entry: /dev/bus/usb/002/ dev/bus/usb/002/ none bind,optional,create=dir 0, 0
lxc.mount.entry: /dev/dri dev/dri none bind,optional,create=dir
lxc.mount.entry: /dev/fb0 dev/fb0 none bind,optional,create=file
lxc.mount.entry: /dev/apex_0 dev/apex_0 none bind,optional,create=file 0, 0
lxc.mount.entry: /dev/apex_1 dev/apex_1 none bind,optional,create=file 0, 0
lxc.apparmor.profile: unconfined
lxc.cap.drop:
lxc.mount.auto: cgroup:rw
```

Install Frigate in Docker using Portainer

- Remember frigate won't start if there is no MQTT server to connect to. In my case I run MQTT in homeassistant so I had to complete the homeassistant install before launching frigate.
- You'll need to preload a frigate.yml in the appropriate folder before launching frigate the first time or it will not start. Look to the frigate doc's on how to create the frigate.yml.
- In the docker container create a folder to hold the database, e.g., /root/frigate
- Create a new stack in portainer and paste in this compose file as a starting point, modify as needed:

```
services:
  frigate:
    container_name: frigate
    privileged: true # this may not be necessary for all setups
    restart: unless-stopped
    image: blakeblackshear/frigate:stable

    shm_size: "256mb" # update for your cameras based on calculation

    devices:
      - /dev/bus/usb:/dev/bus/usb # passes the USB Coral, needs to be modified for other versions
      - /dev/apex_0:/dev/apex_0 # passes a PCIe Coral
      - /dev/apex_1:/dev/apex_1 # passes a PCIe Coral
      - /dev/dri/renderD128 # for intel hwaccel, needs to be updated for your hardware
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /media/hubfiles/frigate/frigate.yml:/config/config.yml:ro  # modify to match your setup
      - /media/hubfiles/frigate:/media/frigate                # modify to match your setup
      - /root/frigate:/db                             # modify to match your setup
      - type: tmpfs # Optional: 1GB of memory, reduces SSD/SD Card wear
        target: /tmp/cache
        tmpfs:
          size: 1000000000
    ports:
      - "5000:5000"
    environment:
      FRIGATE_RTSP_PASSWORD: "password"  #modify to whatever if using rtsp
      TZ: America/Chicago
```

Note: until home assistant and thus MQTT are running, frigate will not run

Install Home Assistant in a VM

Look at this for reference:
Install proxmox and virtualize home assistant | JuanMTech

1. From the proxmox browser, Create new VM
    1. General: name the host and uncheck start-on-boot
    2. Under OS, choose “Do not use any media”
    3. Under System, for BIOS choose “OVMF (UEFI)”, Machine is q35, and for EFI storage choose “local-lvm”. Also ensure you uncheck “pre-enroll keys”.
    4. Under Disks, leave as is
    5. CPU: set to 4 cores or whatever
    6. Memory: set to 4098 max and 2048 min
    7. Network: take defaults

2. Modify the just created VM
    1. On the browser for the just created VM, go to the hardware tab, find the “Hard Disk (scsi0)”, then click “Detach”, then “Remove”.
    2. Get URL for the homeassistant image for proxmox KVM/ProxMox here: Alternative - Home Assistant 39 , copy the download URL to the .qcow2 file
        - On 11/11/2022: https://github.com/home-assistant/operating-system/releases/download/9.3/haos_ova-9.3.qcow2.xz
    3. Open the command prompt for proxmox (not for the VM itself). Run the below commands to get the homeassistant .qcow2 file.
        - wget https://github.com/home-assistant/operating-system/releases/download/9.3/haos_ova-9.3.qcow2.xz
    4. Decompress the file (From the same command line):

        xz -d -v (yourfilename).qcow2.xz

    5. Import the file into the VM, e.g. 102, (From the same proxmox host command line):
        qm importdisk 102 ./haos_ova-9.3.qcow2 local-lvm --format qcow2

    6. From ProxMox UI, go back into VM to attach this new disk to your VM
        Double click on the unused disk
        Check SSD emulation, and add the disk
    7. For the new vm, select “options” and update VM boot options
        Set SCSI drive to first and enable it, disable other boot options
    8. Start VM
    9. Open the VM console – you’ll note that the boot might fail because secure boot is turned on in the bios.
        1. If so then, in the VM’s console, type “exit”. This should bring you to the BIOS config page. Navigate to the secure bios setting and turn it off.
    10. Let the start up process proceed. Note the ip address for the homeassistant in the VM’s console. Open a browser at the IP:8123 and you should be good to go.

11. Follow the onboarding directions at the home assistant install web page

Install Home assistant Frigate add on and integration

Once home assistant has start:

1. From add on store, add this repository: https://github.com/blakeblackshear/frigate-hass-addons
2. Add the “Frigate NVR Proxy” add-on
3. Configure the add on (Enter the IP address and port of frigate) eg 192.168.1.210:5000
4. Set to show in sidebar
5. Start Frigate
6. Confirm you can see frigate in the side bar and it works

Celebrate!

Updating individual pieces

Promox Host

This update can be tricky if the kernal gets updated. If so, then the coral drivers need to be rebuilt into the kernel.

Refer to this site: How to: Update/Upgrade Proxmox VE 7.x (PVE 7.x) > Blog-D without Nonsense (dannyda.com)

- login to Proxmox VE web gui, Navigate to Datacenter -> node/cluster Name -> Updates, click on "Refresh" then click on ">_ Upgrade"
- If the kernel changes, then follow these steps to reinstall the coral drivers:
    1. First see if the coral drivers are installed with "ls /dev/apex*". If apex_0 and/or apex_1 exist then you are done - there is no need to reinstall the coral drivers.
    2. The pve-headers must be installed on the proxmox host before installing the coral drivers:
        a. open: *nano /etc/apt/sources.list*
        b. add this line to the file (refer to Package Repositories - Proxmox VE):

            deb http://download.proxmox.com/debian/pve bullseye pve-no-subscription

        c. now save the file and run: *apt-get update*
        d. run: *apt install pve-headers-$(uname -r)*
        e. you should see that the headers are found and installed
    3. Re-install dkms– see this reference: How to build a kernel module with DKMS on Linux (xmodulo.com)
        a. No need for "sudo" since you are root
        b. Run: *apt-get install dkms --reinstall*
    4. Re-install the coral pcie drivers into the proxmox host, not a vm. Watch the command output carefully to be sure the install worked.
        a. Refer to : Get started with the M.2 or Mini PCIe Accelerator | Coral.
        b. Be sure to use this command to force a coral reinstall:

            sudo apt-get install gasket-dkms libedgetpu1-std --reinstall

        c. If problems the see: M2 EdgeTPU (Coral AI) Problem | Proxmox Support Forum
        d. Reboot!

    5. Verify the coral drivers are working with: *ls /dev/apex**
        - you should get something like this: /dev/apex_0
        - if not, then stop here and figure out what is not working. I had to repeat the dkms install for some reason before I could get it to work. Also, don't forget to reboot!

Portainer

Updating the portainer version should be relatively easy and not upset anything

1. Log into docker using the console on promox for the docker instance, user=root.
2. Referring to: Upgrading on Docker Standalone - Portainer Documentation follow these steps:
    a. *docker stop portainer*
    b. *docker rm portainer*
    c. *docker pull portainer/portainer-ce:latest*
    d. *docker run -d -p 8000:8000 -p 9443:9443 -p 9000:9000 --name=portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:latest*
3. At this point, portainer should be running and the newest version of Portainer will now be deployed on your system, using the persistent data from the previous version, and will also upgrade the Portainer database to the new version.
4. Done

Docker

Updating is as easy as portainer, follow these steps:

1. Log into docker using the console on promox for the docker instance, user=root.
2. Enter these commands:
    a. *apt-get update*
    b. *apt upgrade*
    c. if a text input screen pops up during the update process, select “no configuration change”
3. check that docker is running with “*systemctl status docker*”

Individual containers

To update a container with a new “latest” build (e.g., frigate, wyze-bridge), simply go to the container’s “stack”, and at the top of the page select the “editor” tab and then down at the bottom select “Update the stack”. In the resulting pop-up, select “re-pull image and redeploy”

Note, however, that for frigate 12.x, The compose file needs a modification to pull the code from a new , non-docker, location: *ghcr.io/blakeblackshear/frigate:stable*

My docker compose file that I use to create the stack for frigate 12.x looks like this:

```
services:
 frigate:
   container_name: frigate
   privileged: true # this may not be necessary for all setups
   restart: unless-stopped
   image: ghcr.io/blakeblackshear/frigate:stable
```

```
shm_size: "1024mb" # update for your cameras based on calculation

devices:
  - /dev/bus/usb:/dev/bus/usb # passes the USB Coral, needs to be modified for other versions
  - /dev/apex_0:/dev/apex_0 # passes a PCIe Coral
  - /dev/apex_1:/dev/apex_1 # passes a PCIe Coral
  - /dev/dri/renderD128 # for intel hwaccel, needs to be updated for your hardware
volumes:
  - /etc/localtime:/etc/localtime:ro
  - /media/hubfiles/frigate/frigate.yml:/config/config.yml  # modify to match your setup
  - /media/hubfiles/frigate:/media/frigate                  # modify to match your setup
  - /root/frigate:/db                                       # modify to match your setup
  - type: tmpfs # Optional: 1GB of memory, reduces SSD/SD Card wear
    target: /tmp/cache
    tmpfs:
      size: 1000000000
ports:
  - "5000:5000"
environment:
  FRIGATE_RTSP_PASSWORD: "password"  #modify to whatever if using rtsp
  TZ: America/Chicago
```