Server Documentation

Documentation for System Admins on how to give users access to the Raspberry Pi Club's Proxmox Server and anything else related.

1.0 Network Diagram

2.0 Ricochet Access□□

1. Generate and Add SSH Keys to Github (Refer to this doc)
2. Edit SSH Config File
   - Navigate to inside the .ssh folder on the user's system
   - Create a new file called "config" (unless it already exists) and edit it with Notepad
     - Remove any file extension from the file (e.g. .txt)
   - Enter the following and save the file:

```
Host [NAME]
    HostName raspberrypiclub.org
    User [USER]
    LocalForward 8006 192.168.1.18:8006
    Port 1666
```

3. Import the User's SSH Key (Refer to this doc)
4. Add a New Proxmox User (Refer to this doc)
5. Access the Proxmox Server (Refer to this doc)

3.0 Tailscale Setup □□

1. Install Tailscale and create an account
2. Create new Ubuntu Server VM on the Proxmox server *(Refer to Create a VM in How to: Proxmox)*
3. Create a Tailscale Connection (Refer to the doc)

4.0 Docker □□

Docker Swarm IP Table

Docker Swarm Nodes	IP Addresses
Honeybee	192.168.1.64
Bumblebee	192.168.1.65
Masonbee	192.168.1.66

1. Test whether or not the user can connect to each Docker swarm:
    - Click here to learn how to add new users to the Docker group

```
ssh PAT@192.168.1.64
ssh PAT@192.168.1.65
ssh PAT@192.168.1.66
```

5.0 Setting Up NFS □□

Creating and using NFS with Docker

Any stack file that you deploy will have at least one service that will need to store static data. This will become a problem when stand-alone methods are used for volume creation, because the volume will not magically transport to whatever node the container gets deployed to. Enter NFS share volumes to the rescue!!

Step One - Install NFS

Make sure that you have installed NFS on the clients/nodes (including manager nodes)

```
sudo apt install nfs-common
```

On the Ubuntu NAS (gump.lan or 10.10.1.33 is our BMS file server, as of this writing), install the server components as well

```
sudo apt install nfs-kernel-server
```

Step Two - Using NFS in A Stack

Making an NFS Export

Then, as an example, we might set up a share like this:

```
sudo mkdir /pools/pool1/mysharename
sudo chown nobody:nogroup /pools/pool1/mysharename
sudo nano /etc/exports
sudo systemctl restart nfs-kernel-server
```

The /etc/exports line to add our share and publish it would look something like this:

```
/pools/pool1/mysharename     *(rw,sync,no_subtree_check)
```

Add no_root_squash after no_subtree_check if the container wants to change ownership on files during runtime.

Test The Export

To check if the export worked, use these commands. If no errors are reported - it worked!

```
sudo mount 10.10.1.33:/pools/pool1/mysharename /mnt
sudo touch /mnt/testfile
sudo rm /mnt/testfile
sudo umount /mnt
```

Step Three - Usage in a Stackfile

Add the following to the stack.yml file in the volumes definition area:

```
letsencrypt:
  driver_opts:
    type: "nfs"
    o: "addr=10.10.1.33,rw,noatime,rsize=8192,wsize=8192,tcp,timeo=14,nfsvers=4"
    device: ":/pools/pool1/mysharename"
```

Deploy

Deploy the stack, this will create the volume and connect it as it deploys. Using Portainer, you can also browse the content.

6.0 Monitoring □□