

# Cluster Computing

By Levi Savage Lowe

April 12, 2022

Cluster computing is a relatively new form of computer science manipulation. To accomplish building a cluster computer, one must link two or more computers together by using a management operating system. VMWare and ProxMox are examples of this type of operating system (OS). By linking hardware together, one can easily create a new system that has the ability to compete with super computers. I knew that building a cluster would be a knowledgeable journey for me, while providing for future students of the Computer Science program. These students will have a powerful new piece of technology that can help them learn in the future.

When I first started this project, I was quite excited because this was actually something I had been looking forward to doing for a few years now. I had started mild research in cluster computing back in the second half of my third year in college. I was originally looking at the idea of clustering two raspberry pi units. Raspberry pis are small computers that are usually used to control small devices such as robots. These little computers are very limited with their capabilities which makes the idea of increasing their resources by clustering two of them a desirable idea. This is generally done by distributing multiple tasks across the different units allowing them to preform them at a fast speed. Even though this is where my research first began, clustering raspberry pis is completely different from clustering computers or servers.

With my new research, I found that specific OSs are needed in order to do proper clustering. As stated before, these OSs are specifically used to manage systems and also configure these computers to work together while using all of the resources available to complete a task.

At the start of my project, I found myself only looking at working on desktops to create an above average computer. This would have been done with old desktops that I was able to retrieve from Alamosa High School and Adams State University. However, these plans quickly changed after I noticed that Adams State's Computing Services (CS) was about to recycle six of its old servers. After a

few weeks of talking with CS, I was able to acquire these servers. Once I got the servers moved into the Turing lab, I discovered each server had an outstanding amount of storage available which combined to approximately 12 Terabytes (TB) and each server had at least 4 CPU cores within it.

Unfortunately, upon taking a look inside of the servers, I found that almost all of the Random-Access Memory (RAM) was removed. Of course, servers are unable to operate without any RAM, so I ended up pulling all of the RAM that I needed out of the old desktops that I had collected from Alamosa High School and Adams State. This enabled me to give the servers about 400 Gigabytes (GB) of RAM. With all of the hardware issues inside of the servers fixed, I was now able to move on and acquire the last of the hardware I needed: a network switch to give all of the servers internet without needing to find 12 Ethernet ports to connect them to. All I needed to do was plug the network switch into an available Ethernet port which would give it the



*Figure 1: All of the servers and the Kiosk plugged in and ready to run.*

ability to distribute an internet connection to each server. Network switches are configured to do this out of the box so no other configurations were needed, and the type of switch would not affect the

performance of the cluster. Next, I needed a monitor and desktop which would serve as a Kiosk as well as the essential computer accessories such as a mouse, keyboard, cords, etc.

Once I finally had all of the components plugged in and ready to go, I booted up the Kiosk. I immediately ran into an unforeseen problem: the desktop did not work at all. I quickly acquired another working desktop and was capable of getting it to boot. Finally, I turned on all of the servers and the desktop to find that they all powered on. With this first victory, I quickly ran into another issue: one of the servers would not display to the monitor.

This now required me to go through troubleshooting on that server. First, I tested both of the display ports on the server to make sure it wasn't an issue with the first port I chose to use; however, this provided no results. This pushed me to double check that the monitor was working correctly which turned out to be fine after using it with the other servers. Finally, I turned to the RAM that I had just installed inside of the server. I removed all of the cards leaving one inside to test. Upon turning on the server, I finally got a display on the monitor. This meant that the main problem was likely that one of the RAM cards that I installed was defective. I continued to add RAM into the server until one of them finally stopped allowing the server to display an image on the monitor.

With the bad RAM finally identified, I replaced it and found that all of the hardware now worked as it needed to. It was time for me to move on to installing the OS on each of the servers as well as the desktop. On the desktop, I chose the Zorin OS, and on each of the servers, I chose to run the ProxMox OS.

These two OSs were chosen based on how they function and how they can cooperate with one another. Both are different flavors of the Linux based OS which allows them to easily be able to remote into one another with secure shell (SSH). To be a bit more specific, Zorin is a user-friendly OS that uses Graphical User Interface (GUI) to use the computer similarly to other OSs such as Microsoft Windows 10. This differs from ProxMox which has no GUI but is instead a terminal that runs a virtual

machine (VM) manager that can be accessed through a web browser such as Chrome, FireFox, Microsoft Edge, etc.

Through Winter break, I had researched and decided that ProxMox and VMWare would be my best options as management OSs based on how they handle using multiple machines as well as their ability to manage virtual machines. After learning that these two OSs work almost exactly the same, the main difference being that VMWare needs to be purchased while ProxMox is free to use, I decided to use ProxMox.

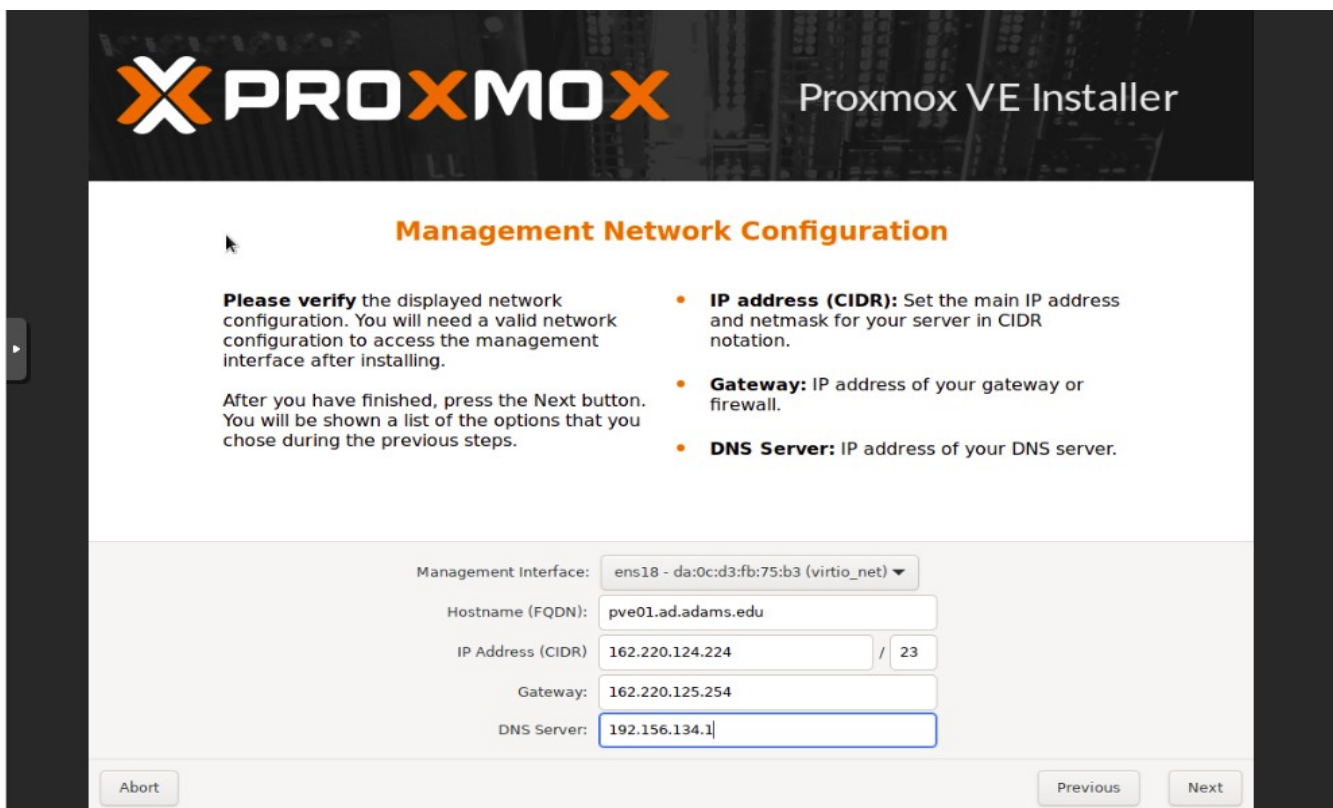
Soon after, I was stuck at installing ProxMox on the servers because I needed CS to set a static Internet Protocol (IP) address to each of the servers respectively. By giving CS the Media Access Control (MAC) addresses for the servers, they would set specific IP addresses to the servers. This will lock in the IP address keeping it from being used by any other computer on campus. If this was not done, then major issues would come up while configuring ProxMox. I contacted CS to have this configuration done within their network. After communicating with them for a week, I was informed that they already had a few IP addresses set aside for the Computer Science department. They then asked me to test these IP addresses to make sure they were currently available.

After further testing, these IP address proved to be available leaving me only needing to collect to the campus's network gateway as well as its primary and secondary domain name system (DNS) which CS gave to me right away. With having all of the network information that I needed, I started installing the Zorin OS to the Kiosk which did not have any problems installing, and I took a few steps to complete the configuration. Finally, I turned my attention to the first of the servers.

This first server was going to be the first of ProxMox managers. In turn, it would become the housing for the rest for the cluster. First, I needed to begin the installation process for the OS. The name pve01 was given to this specific server. Once the installation process was completed all I needed to do was type in '<https://162.220.124.224:8006>' to a search bar to access the web interface of the server. To

log into the server, I needed to use the root user since there were no other log in options available until I created them.

My first task when I opened pve01 was to check and make sure the summary mirrored the hardware that was available. Once everything checked out, I went through and made sure all of the software was up-to-date which took a little bit of time to get through. Once the updates were finally completed, it was time for me to move on to building templates.



**PROXMOX** Proxmox VE Installer

### Management Network Configuration

**Please verify** the displayed network configuration. You will need a valid network configuration to access the management interface after installing.

After you have finished, press the Next button. You will be shown a list of the options that you chose during the previous steps.

- **IP address (CIDR):** Set the main IP address and netmask for your server in CIDR notation.
- **Gateway:** IP address of your gateway or firewall.
- **DNS Server:** IP address of your DNS server.

Management Interface: ens18 - da:0c:d3:fb:75:b3 (virtio\_net) ▼

Hostname (FQDN): pve01.ad.adams.edu

IP Address (CIDR): 162.220.124.224 / 23

Gateway: 162.220.125.254

DNS Server: 192.156.134.1

Abort Previous Next

*Figure 2: Installation of ProxMox on server 01 and labeling it as pve01, setting its IP address, Gateway, and DNS.*

When building the templates, I first needed to download proper Optical Disc Image (ISO) files for the corresponding OSs that I wanted the cluster to have access to. The desired OSs were the following: Ubuntu, Ubuntu Server, Zorin, Zorin Lite, Microsoft Windows 10, and Microsoft Windows Hyper-V Server. There are two available paths that I could use to have ProxMox collect these files. One

was either uploading the file from the Kiosk local library or copy and paste the download Uniform Resource Locator (URL) for the file.

The easiest path was to import the download URL and have ProxMox download and configure everything itself. This process worked for Ubuntu, Ubuntu Server, Microsoft Windows 10, and Microsoft Windows Hyper-V Server. Unfortunately, I ran into some issues with Zorin and Zorin Lite. No matter which URL I chose, I was unable to get ProxMox to be able to download and configure these two OSs. I instead decided to take the route of downloading the ISO to the Kiosk local library and then uploading it the ProxMox by uploading it from said library.

I still had no luck in being able to get either of the Zorin OSs on the ProxMox database which left me to decide to abandon those OSs as options. Now with my chosen OSs downloaded into ProxMox's library, I was finally about to start creating either VMs or Containers (CT). First, I needed to figure out if VMs or CTs were what I wanted my templates to be made out of. Some of the key differences between these are the use of resources as well as live migration.

First, I will talk about what VMs and CTs are. These two branches are emulations of hardware and software of actual computers. Management OSs such as ProxMox use emulation to simulate all of the needs of an actual machine such as CPU, RAM, and Graphical Interface. VMs and CTs are just two different versions of this simulation. VMs are capable of completing live migration, which is crucial for servers and web pages. Live migration is having the entire emulation transferred to a whole other unit which is apart of the cluster without any downtime. As a simple example, we have a web server running on pve01 and this unit needs to both update and restart. One can simply tell the web server to migrate over to the unit pve02 without needing to shut anything down on the web server side. This allows a web page to continue running without shutting down, and users can still visit the page while pve01 updates and restarts.

Creating a VM and CT is fairly easy within ProxMox: all one needs to do is click on the Create VM or Create CT in the top right hand corner of the ProxMox web interface. One would then designate which OS they would use as well as how much CPU, RAM, and internal storage the emulation will use.

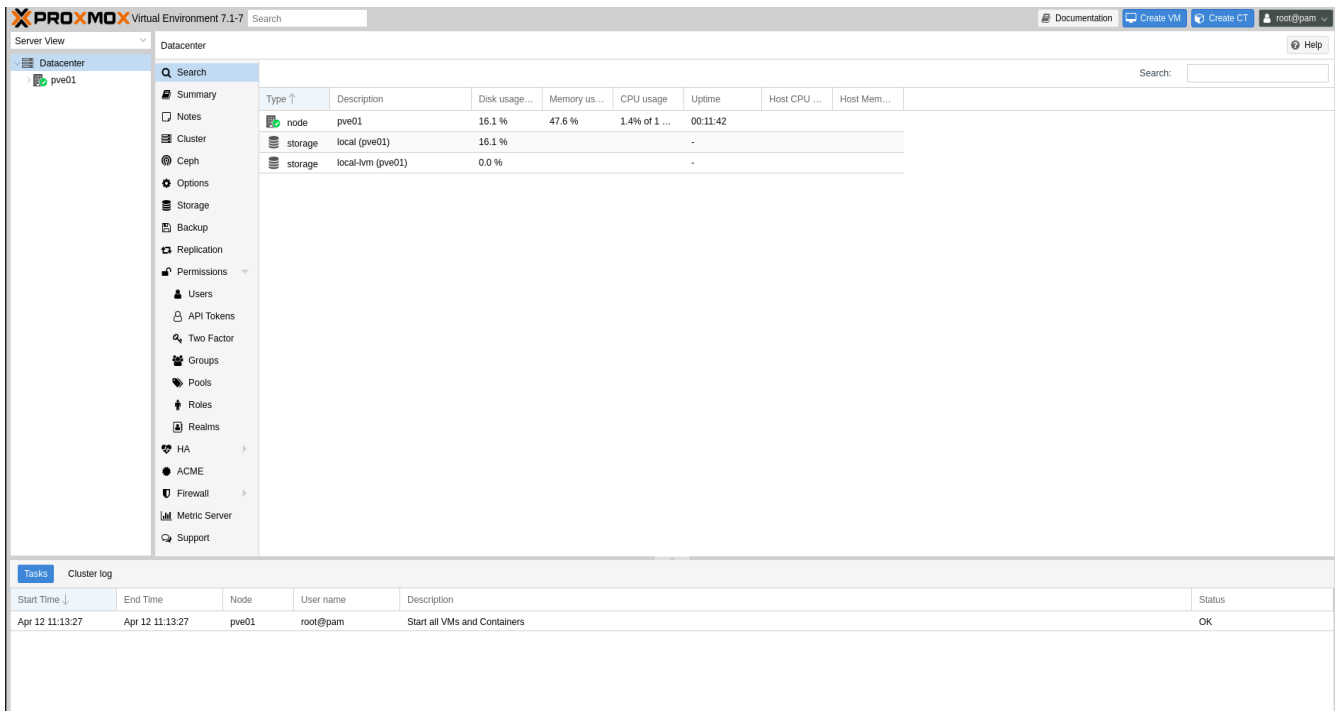


Figure 3: The web interface of ProxMox that is displayed once the URL <https://162.220.124.224:8006> is searched

The draw back to using VMs, though, is resource consumption. VMs require the exact same number of resources that are needed to normally run whatever OS it is being asked to simulate. An example is Microsoft Windows 10, which recommends 4 core CPU and at least 4 GBs of RAM. This is where CTs become beneficial because CTs are able to use fewer resources that are recommended for an OS and still be able to run it optimally. The drawback with CTs, though, is they are unable to live migrate like VMs.

Now with an understanding of what VMs and CTs are, I ended up deciding to make the templates out of VMs since there were plenty of resources in the servers that would be running the



cluster. I started out by creating an Ubuntu and an Ubuntu Server VM. After proper installation, I needed to do proper configuration within both of the VMs to make them work properly when being cloned from a template. My first step required me to remove all of the `ssh_host` files within the directories of the OS so that once the new templates were created they would generate new SSH IDs so one would be able to SSH into each OS if needed. If this was not done, it would create issues in the future because two host's having the same SSH ID would eventually create conflicts with communication.

The second step needed was truncating the Machine ID that was now associated with the OS. This was simply done with a terminal command. With these two steps completed, the two VMs were ready to be turned into templates that could be used over and over again. After completing the process of turning them into templates, I went through the template settings and cleaned them up a bit so no unnecessary resources were being used when they were cloned. I then moved onto creating the Microsoft Windows 10 and Microsoft Windows Hyper-V server templates.

When I first tried booting either of these ISO files, I was able to begin running their installation, however, I quickly ran into problems. Since ProxMox is not a normal BIOS, the Windows Installation Media seemed to be missing certain drivers that would allow it to detect the virtual hard drive that I was using for these VMs. With a little research on the ProxMox wiki, I was able to find these missing drivers in an ISO file that ProxMox had developed for creating Microsoft Windows VMs. After creating a new CD Drive for the Microsoft Windows VMs, I was able to plug in the `windows_drivers.iso`. By digging through this file inside of the Windows Installation Media, I was able to get virtual hard drives to be detected, however, this revealed a new issue. The Windows Installation Media was unable to write to these types of drivers.

This issue is where I hit a dead end and was unable to get the Windows VMs to work properly. I decided to abandon these ISO files until I could come back to them at a later point and figure out why

the hard drives could not be used. Now that I had the Ubuntu templates created, it was time for me to create users that could use the cluster and eventually invite others. For the time being, I decided to only create three new users. Dr. Shafee, Professor Sellman, and myself. I started with myself as a guinea pig so I could make sure that I got the basics down on my own account first before trying to create anything new for anyone else.

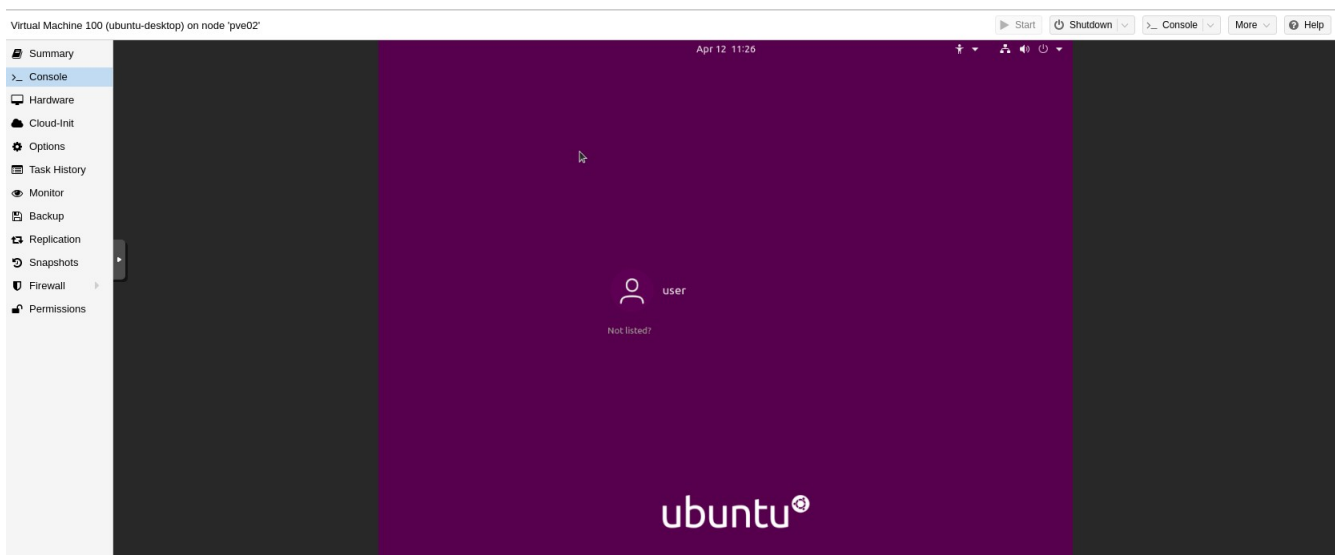


Figure 4: A virtual machine of Ubuntu that is being managed by Proxmox.

First, I needed to create a user for myself within the Proxmox GUI which was as simple as going to the user page and adding a user. Sadly though, this actually did not add a user into the actual database of Proxmox. Instead, I needed to enter the Proxmox shell and add myself as a user through the user-directory file. With this done, I was finally a user that could log in with Proxmox, but I was a user with the lowest set of permissions which only allowed me to view the cluster and not initiate any commands. To fix this issue, I needed to create a new group known as 'Admins' which gave any user that was a member of the group Administrative privileges. The same was done to create Dr. Shafee's and Professor Sellman's user accounts.

With the users finally created, it was time for me to create the cluster itself. This whole time I had only been working on one of the servers and configuring it to be the central unit to pull all of the

servers together into one branching unit. My next step was to power on all of the other servers, install ProxMox on them, and configure them as needed. One of the main reasons I chose to use ProxMox was its ability to easily cluster with all ProxMox OSs. Once all of the servers had been completely configured to be pve02 – pve06, I went back to pve01 and created a new cluster group named pve-cluster.

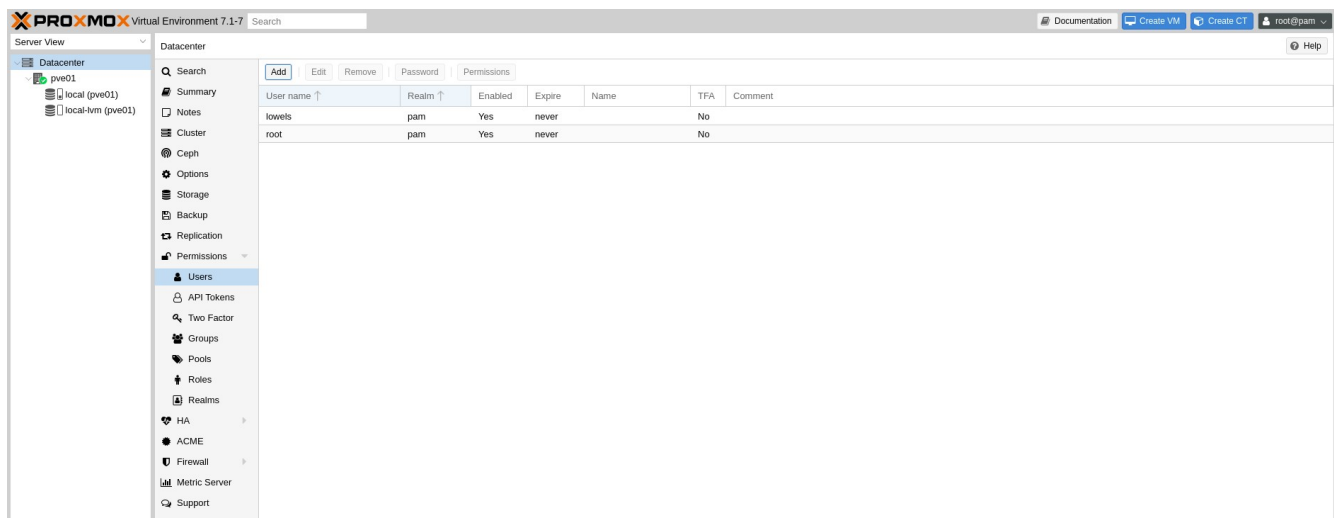


Figure 5: The creation of a user within the ProxMox GUI

From this point, all I needed to do was pull all of the cluster information into pve02 – pve06 to become part of pve-cluster. Within this cluster, I could now see all of the resources that are currently available. Pve-cluster has 72 CPU cores, 404.49 GBs of RAM, and 11.82 TBs of internal storage. The cluster was a success until just recently. Unfortunately, pve01 stopped responding to the cluster on the morning of April 12, 2022.

What exactly does this mean for the cluster though? Sadly, all of the ISO files, templates that were built have been lost as well as the user information that had been entered into pve01. This means these three components need be re-downloaded and reconstruction. Thankfully, pve01 was the only server that had almost no storage or RAM, leaving pve-cluster with 64 Cores, 392 GBs of RAM, and 11.74 TBs of storage. Not all is bad though, since this event occurred early on in the clusters life not

much data was lost. This shows an important path that I feel the cluster needs in order to stay reliable as well as effective. This path is that the Computer Science department should purchase a type of shared storage for the cluster to use. This way pve02 – pve06 will all have access to the same data as well as be able to automatically rescue all data that one server has and send it to another before it is too late and the data is lost. What I am specifically talking about is known as high availability.

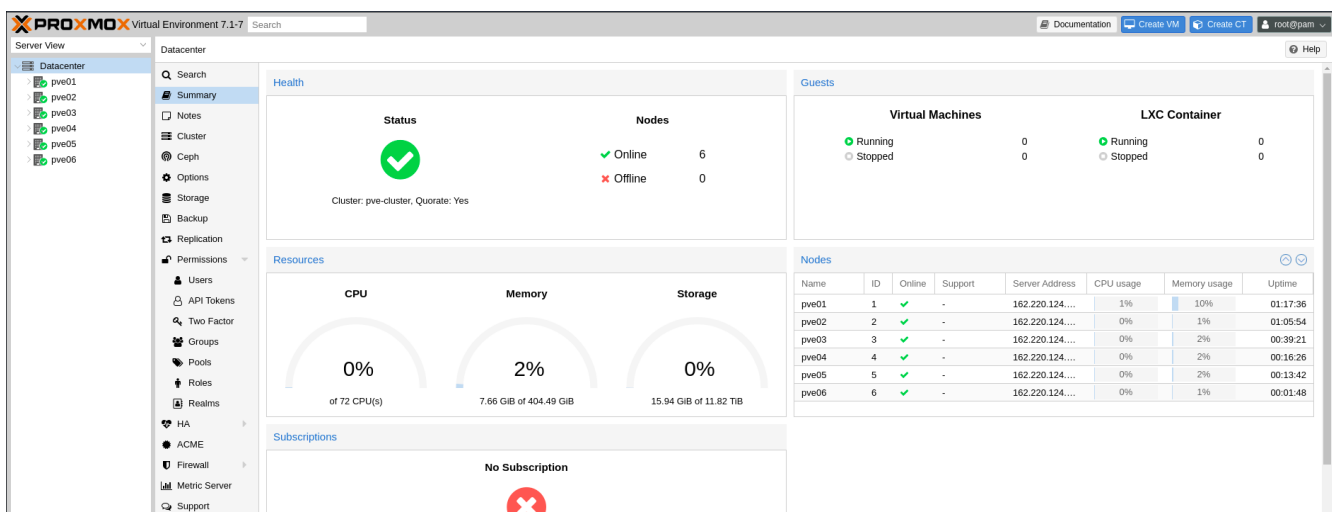


Figure 6: Summary of pve-cluster and with all available resources.

High availability at its most basic form works as follows: one has several VMs running web servers on pve01 as well as a few templates to build new web servers. With high availability active on a cluster, if for any reason pve01 ever stops working, all of the VMs and templates will immediately migrate over to pve02 or any other available server that can hold them. This helps save data as well as keep VMs running without any unforeseen issues.

In the end, I decided to fully recreate the cluster in order to get pve01 to respond once again. This meant that I needed to install new OSs into each of the servers. This took me an entire day to complete; however, I feel it was well worth it since it brought pve01 back into the cluster. To hopefully avoid anything like this happening again, I set pve-cluster to back itself up every Sunday at 0100. This will keep the cluster safe and secure for the future, however, having a shared storage that these backups

can be sent to would be recommended in the future because they are currently being saved to the local storage of the cluster.

When the cluster was completed, I was able to see that my predictions for the amount of resources where correct and after creating about 10 VMs with 12 CPUs, 40 GBs of RAM, 64 GBs of integrated storage. All of these VMs worked optimally as if they were real computers without any long loading times.

With the pve-cluster running fully once again, one can easily run many VMs or CTs seamlessly. This grouping of servers can provide students with many different experiences that will allow them to learn proper cybersecurity or how an unstable program or software can affect a computer. In the future, students or faculty could even add onto the cluster with more servers or even desktops. This will of course give the cluster even more resources to pull from making its possibilities even greater. In all, I am quite pleased with my end result and even though some issues where unable to be resolved, the end product is still the targeted goal, and I also learned what the pve-cluster needs to succeed in the future.

## References

Hasan, Mehedi. "Everything You Need To Know About Linux Zorin OS." ubuntuipit, <https://www.ubuntuipit.com/everything-you-need-to-know-about-linux-zorin-os/>. Accessed February 25, 2022.

LaCroix, Jay. "Proxmox Full Course." YouTube, uploaded by LearnLinuxTV, 7 Dec 2021, <https://www.youtube.com/playlist?list=PLT98CRl2KxKHnlbYhtABg6cF50bYa8Ulo>.

"Main Page." Proxmox VE, . 24 Jan 2022, 08:30 UTC. 13 Apr 2022, 20:09  
<[https://pve.proxmox.com/mediawiki/index.php?title=Main\\_Page&oldid=11292](https://pve.proxmox.com/mediawiki/index.php?title=Main_Page&oldid=11292)>.