Hubs Cloud Community Edition

Private Server Installation and Configuration Guide

License

Table of Contents

Table of Figures

The Grid. A Digital Frontier.
I tried to picture clusters of information as they moved through the computer.
What did they look like? Ships? Motorcycles? Were the circuits like freeways?
I kept dreaming of a world I thought I'd never see.
And then, one day… I got in!

Kevin Flynn - Tron: Legacy

1 Introduction

Mozilla Hubs

Ever since I found out about Mozilla Hubs, I wanted to get my own server running. I don't mean I wanted an AWS-hosted version - I wanted to host it on my own server hardware, and allow users both within my organization, as well as external visitors, to be able to use it.

While there were some articles on self-hosting, there always seemed to be some sort of limitation, especially around sound support. Try as I might, I could never quite get it going, and what I did get working was not very stable.

On May 31st, 2024, everything changed. This is the day Mozilla ended support for Mozilla Hubs, and the non-profit Hubs Foundation was incorporated.

Hubs Foundation

The community came together to continue supporting and evolving the project(s), eventually releasing the Hubs Cloud Community Edition. This allowed users to deploy a full Hubs stack outside of AWS that was fully under their own control.

Like all good projects, Hubs is a complex infrastructure of services and applications, and (at least as of this writing) still contains some interesting hold-over characteristics from the AWS hosting days. As such, most folks will probably find it easier to deploy on a hosting provider. There are some good articles and instructions to follow on the Hubs Foundation website
https://hubsfoundation.org/
and gitlhub site
https://github.com/Hubs-Foundation/hubs-cloud/tree/master/community-edition.

I'm not most folks. I authored 4 patents on data privacy and security - I have a keen interest in having complete control of my own private server. I also do not want to have to worry about somebody's business decisions having an impact on me. I like having access to source code that I can pick apart, understand, and maintain myself. I like being the master of my own destiny.

DIY Metaverse - Hubs Cloud Community Edition Server

This document outlines how to take your own bare metal server and turn it into a full Hubs Cloud Community Edition server. It will show you how to deploy and manage;

- Fully-functional hypervisor as a private cloud
- Robust firewall that includes NAT, DNS, and DHCP services
- Private email server for access control
- Fully functional Hubs Cloud Community Edition server
- Reverse Proxy to allow public access (optional)

The main differences between this type of deployment versus external hosting provider versions are

1. **Control and Privacy.** Since this is running on your own hardware, anything you put on the server is completely under your control.
2. **Cost.** The deployment outlined herein does not *require* any subscriptions, registrations, or monthly fees. The only costs are for the actual server hardware, and support services such as internet and electricity.

Final Thoughts Before Proceeding

If you like the idea of controlling your own Hubs Server on your own hardware, this document will give you the blueprints and guidance to help make it a reality.

There is a lot of work to be done.

While I will try and explain the purpose of each step, there are a lot of concepts to bring together here including networking, emailing, domain management, certificate management, server management, and more. There are no scripts. There are few GUIs. This all runs on command-line Linux, not on Windows or MacOS (although I have been asked to port it to Windows and/or Windows Server - stay tuned!).

If I haven't scared you off yet, then let's get started on a journey of discovery as we build our own private metaverse together. Enjoy!

Terry

Disclaimer:
The instructions provided in this guide are for informational purposes only. While every effort has been made to ensure accuracy, the author is not responsible for any damage, loss of data, or other issues that may arise from following these instructions. Proceed at your own risk, and ensure you have adequate backups and recovery measures in place before making any changes to your systems.

2 Support Infrastructure Installation

2.1 Hardware

Let's start with the hardware.

Recommended Server Hardware

- Intel 64 or AMD64 with Intel VT/AMD-V CPU flag
- Memory, minimum 8 GB
- Fast and redundant storage, best results with SSD disks
- 2 x 1 Gbit Network Interface Card (NIC)

For this build, I bought a used Dell PowerEdge T410 Server;

- Intel(R) Xeon(R) CPU X5670 @ 2.93GHz (2 Sockets)
- 128GB RAM
- 16TB RAID 0 Storage (I used a bunch of 4TB disks, not SSDs)
- 2 x 1Gbit NIC

(Side note: I bought this as a used server for around C$900.00.)

Make sure that the BIOS is up-to-date, and configure the storage as a RAID 0 (one big disk) configuration (if you are using a RAID controller). Since I don't know your hardware, I'll leave these two steps as an exercise for the student. Good luck!

Recommended Workstation

While most people use Windows or Mac, I use a Linux (Debian) workstation. This guide is written almost exclusively for Linux. Where possible, I will show examples of using Windows workstation commands as well. If you want to use Windows or Mac as your workstation, you should have the following:

- **Up-to-date browser.** I have found that Google Chrome or Mozilla Firefox work best.
- **Telnet/SSH client.** This is built into MacOS. For Windows, you will need to download and install something like PuTTY. If you are on Windows 10 or later, then try the built-in Windows Terminal.
- **Network Configuration.** Understanding of how to manually configure your workstation IP address (NOT using DHCP). This is only needed for a short time until the firewall is installed, but is a necessary step.

Additional Requirements

Since we are going to be creating a local network, and connecting the machine to the internet, we will

need to use both NICs on the server. One will be the Local network (LAN), and the other will be the Wide-Area internet connection (WAN).

In order to have the workstation on the LAN, we will also need a local ethernet switch (any 5-8 port 1Gb will do). You will be able to replace this with a smart (e.g. WiFi) router at the end of this to allow for smart device access to the server, but configuration needs a wired connection through a dumb router in order to let the devices be on the same subnet to begin with.

Now plug the first server NIC into the ethernet switch, and the second server NIC into your ISP's router.

You will eventually attach your workstation's NIC to the ethernet switch but, for now, leave it connected to the internet via your ISP router, as we have some server software to download and install first.

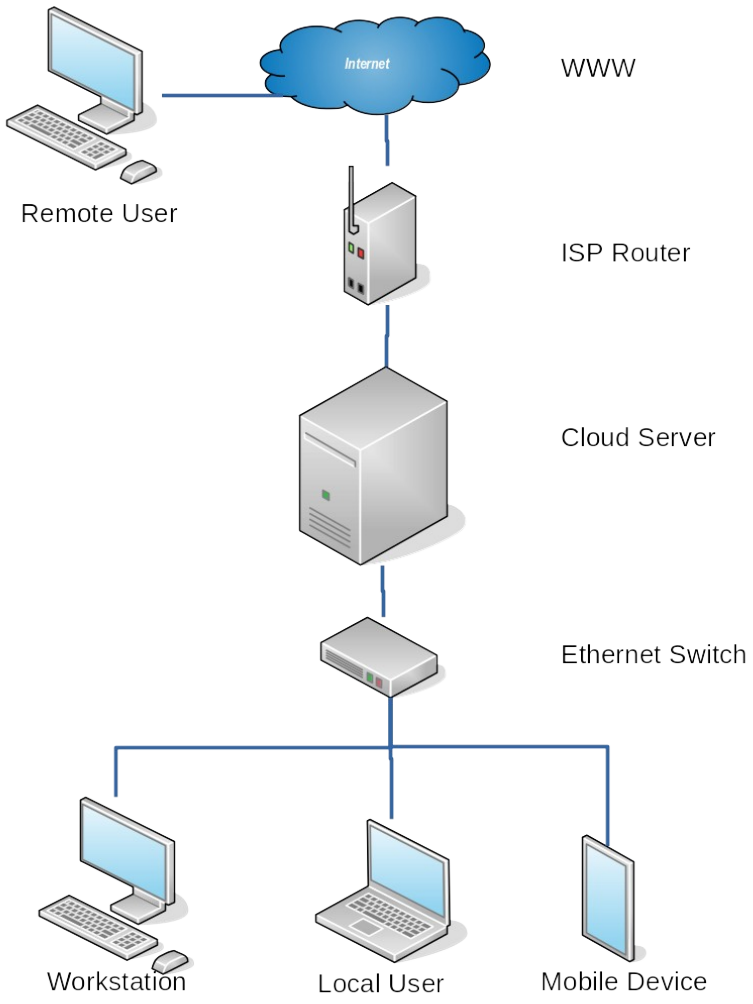


Figure 1: Network Topology

2.2 Required Software

There are a number of packages and servers that we need to download and install on our cloud. Navigate to each of the following in a browser, and download to your workstation (typically to a Downloads directory);

- **Proxmox**. This is our private cloud platform.
  https://www.proxmox.com/en/proxmox-virtual-environment/overview

- **Debian**. This is our guest Linux operating system. Hubs will be running on two Debian instances.
  https://cdimage.debian.org/debian-cd/current/amd64/iso-dvd/

- **OPNSense**. This is the firewall. It will also provide many of the networking services, including DHCP, DNS, and more. Select “amd64” for the architecture, and “dvd” for the image type. Then select the mirror location closest to you.

  Since OPNSense is a compressed image, you will need to decompress it back into an ISO file. On your workstation, you could try right-clicking the .bz2 file and selecting “Extract Here” in Linux. You may need to download and install a decompression tool for Windows or MacOS in order to decompress it.
  https://opnsense.org/download/

- **Citadel**. This is our local email server. It will provide authentication support without needing to register (and pay for!) a real domain.
  https://citadel.org

In all cases, you should check the integrity of the images by performing the recommended checksums. On Linux, this is done using the shasum command. For example:

```
$ shasum -a256 OPNsense-24.7-dvd-amd64.iso.bz2
4452df716417cac324bb06322fc4428870ac2a64fd6ae47675a421e8db0a18b5  OPNsense-24.7-dvd-amd64.iso.bz2
```

As you become comfortable with this infrastructure, you can start making substitutions. For example, you may be behind a real corporate firewall. If so, the local firewall becomes redundant. As well, you may already have a registered domain, and email service provider. You could then forego installing Citadel and use your existing email service instead. Even Proxmox can be replaced with a different hypervisor such as VMWare ESXi or others.

2.3 The Private Cloud (Hypervisor)

What is a hypervisor? A hypervisor (aka virtual machine monitor - VMM or virtualizer) is software that allows users to create and manage virtual machines running a variety of operating systems and supporting various hardware configurations.

A bare-metal hypervisor typically installs like an operating system onto server hardware. The user typically boots some installation media (e.g. a USB drive or DVD) and installs a bootable image to the server.

A computer on which a hypervisor runs one or more virtual machines is typically called a ***host*** machine, and each virtual machine is called a ***guest*** machine.

Proxmox Virtual Environment

While there are a number of Hypervisors available, fewer and fewer are available for free and based on open-source.

For our purposes, we will be installing Proxmox as our hypervisor.

You should have already downloaded the installer image for the latest Proxmox VE ISO Installer.

Step 1: Create Boot DVD

Call me old fashioned, but the best way to use this ISO is to burn it to a DVD. If you don't already have one, you can still find them on Amazon or at Best Buy or Canada Computers, for example.

On Linux, MacOS, or Windows, it is a simple right-click on the ISO file in your file manager to burn the image to a blank disk.

Step 1 (alternative): Create USB Stick

Alternatively, you can create a bootable USB drive using the Linux dd command, or a Windows tool for creating bootable USB drives. Again, left as an exercise for the student.

Step 2: Boot the Install Media

I would suggest the following installation options in the Proxmox installer:

- ***Install using Text UI.*** If you are trying to use graphical install, it may fail depending on the type of graphics card you have in the server. For example, the T410 fails loading Cairo for Matrox driver support.
- ***Select XFS File System.*** Installation may fail on unsquashfs of cdrom if ext4 selected.
- ***Manually configure the LAN NIC IP address***. For this example, I will assign the server the

following:

```
IP address: 192.168.100.2 / 24
Subnet mask: 255.255.255.0
Gateway: 192.168.100.1 (This will be the address of the firewall.
```

Why do we need to manually assign an IP address?

We have a chicken and egg scenario. If we are installing the firewall as a guest on the host server, then the host needs to boot first, and the firewall boots second. However, the firewall is providing the DHCP service that would normally dynamically assign the IP address to the host server. Chicken and egg.

So we manually assign the IP address for the host server to use at boot time, and then configure the firewall to reserve this address and assign it to the host so all other nodes on this subnet (192.168.100.x) can reference the host server by name via DNS.

Once the Proxmox installation is complete, the server should reboot, and ultimately display a console message stating that the server can be reached at a specific URL using a browser.

Why the 8006?

Proxmox management console listens on the LAN NIC at port 8006, instead of the traditional HTTP port 80, or HTTPS port 443. It still uses SSL, so the first time you connect to it, you will need to accept the self-signed certificate.

Step 3: Connect to the Proxmox Management Console

It is now time to unplug your workstation from your ISP Router, and plug it into your new private switch. On the workstation, you will need to configure the NIC to a manual address as well, as there is still no DHCP service running on the LAN yet. I use 192.168.100.100 for my workstation address, as this will be in the dynamically assigned DHCP address range (more on that in the firewall section). For now, assign the IP address of the workstation manually.

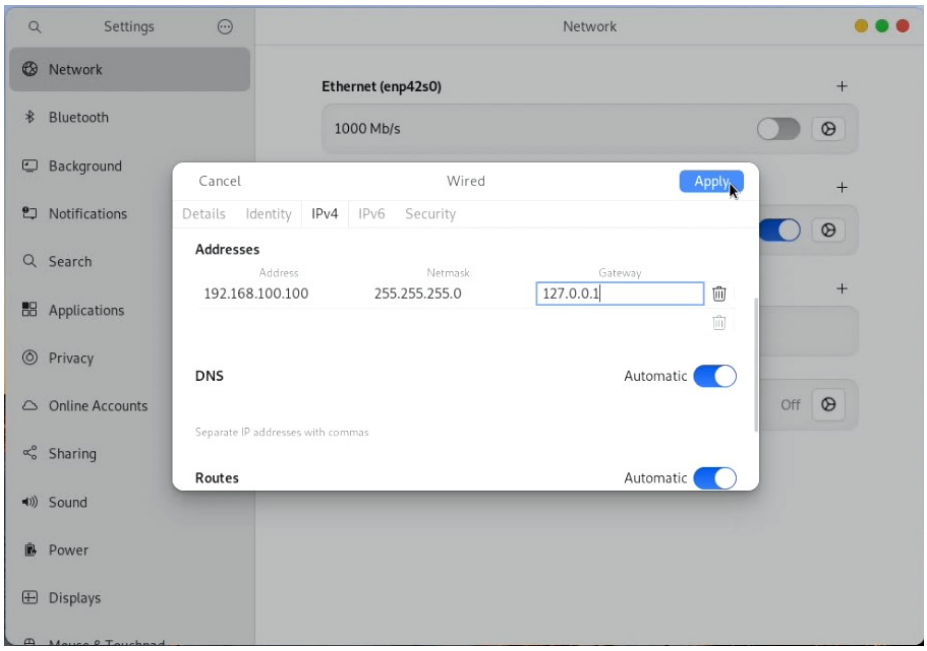


Figure 2: Debian Manual IP Assignment

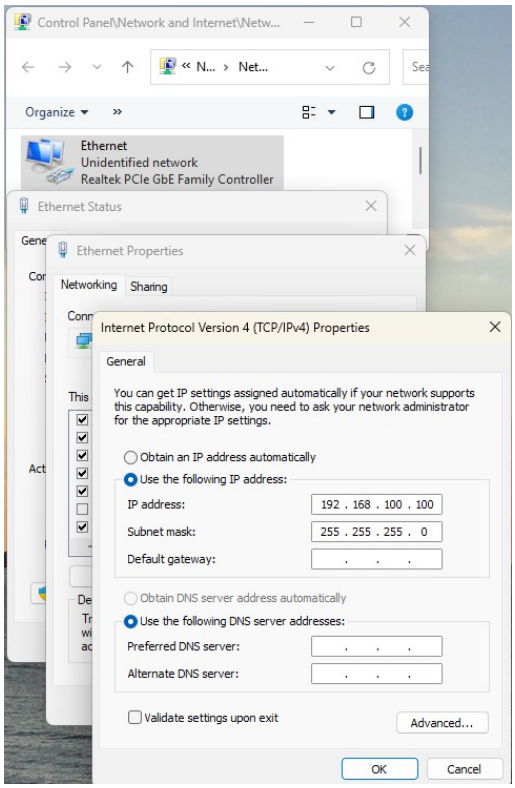


Figure 3: Windows Manual IP Assignment

If you are using Windows as your workstation, open the control panel and navigate to Network and Internet→Networks, select your card and right click to navigate to Ethernet Status→Properties→ IP4 Protocol. Then enter the static IP address and subnet mask, and save it. Similarly, MacOS allows you to manually assign an IP address.

Why are we manually assigning an IP address? This is temporary until we get the firewall running, which will provide the DHCP and DNS services. Once running, we can then automatically assign the IP address of the workstation via DHCP.

Once configured, you can open a browser session to the console. In our example, that URL would be https://192.168.100.2:8006. Log into Proxmox for the first time. The username should be “root”, and the password should be the one you specified at install time.

This is our private cloud.

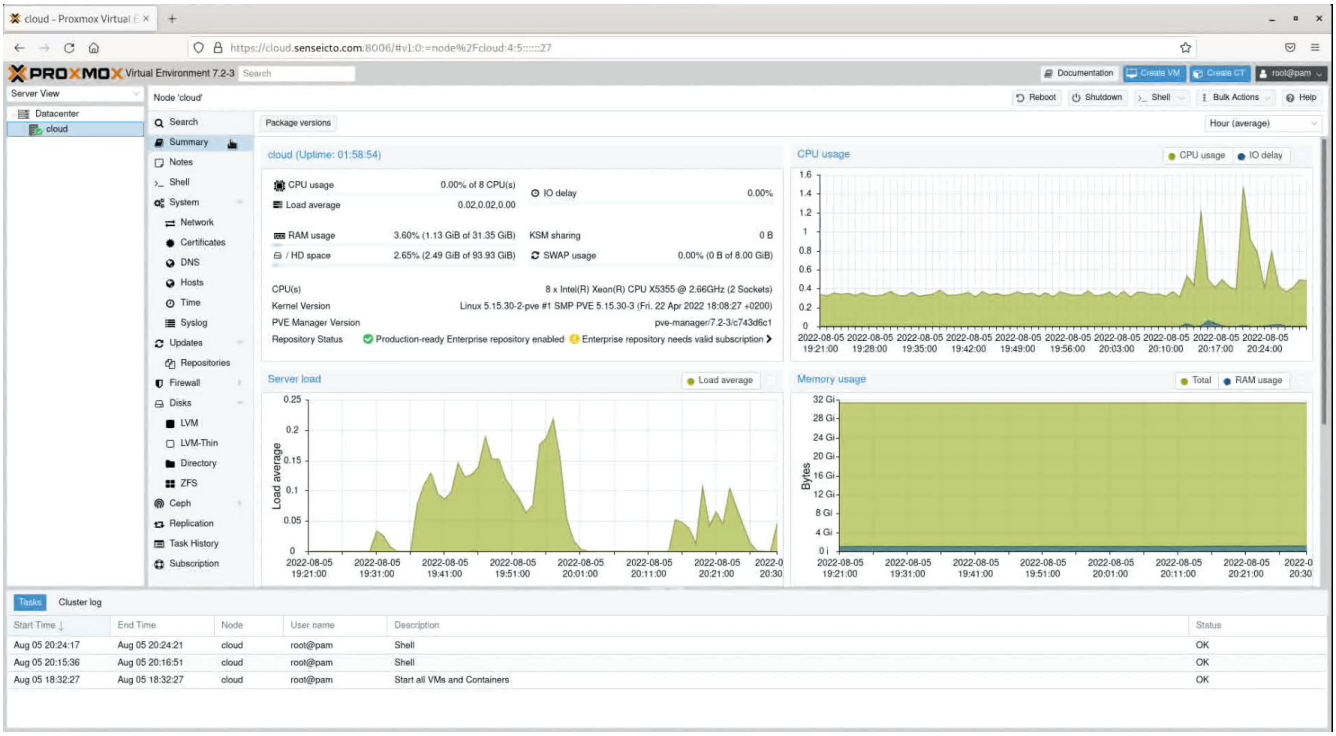


Figure 4: Proxmox Console

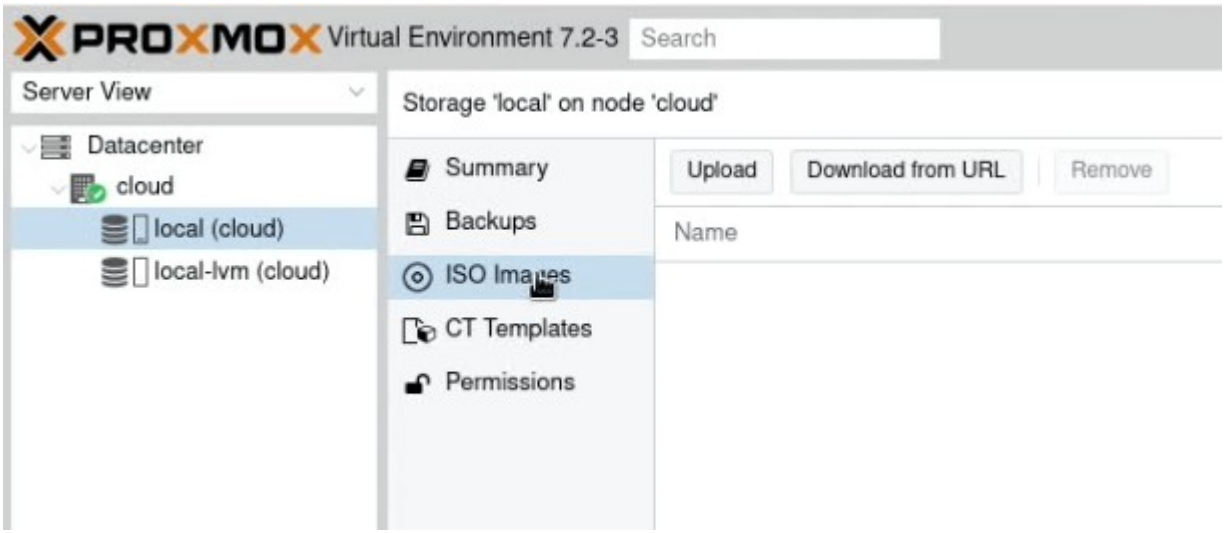


Figure 5: Proxmox ISO Uploading

On the Proxmox administration console;

- Navigate to the Host→Local (disk)
- Select the “ISO Images” option on the sidebar.
- Click the “Upload” button to open a file browser on your workstation.
- Navigate to the workstation download directory to which you downloaded and decompressed the Debian and OPNSense ISO files.
- Upload each ISO to the cloud server.

Why do we need to upload the ISO files? Proxmox allows you to create guest machines by running their installer as if you had booted a new PC with it. In order to boot that installer image, it needs access to the ISO file. Normally, this ISO would be burned to a CD or DVD, or used to create a bootable USB drive from which the new PC hardware could boot. Proxmox uses the ISO file directly, but needs it accessible on its own file system, so we upload them all.

2.4 Firewall

For this server, we use OPNSense as our firewall, which we downloaded in the previous step.

2.5 Create a Guest

In order to create a guest virtual machine, click the “Create VM” button on the top-right of the Proxmox console. This will open a wizard to guide you through the virtual machine creation. I recommend the following settings:

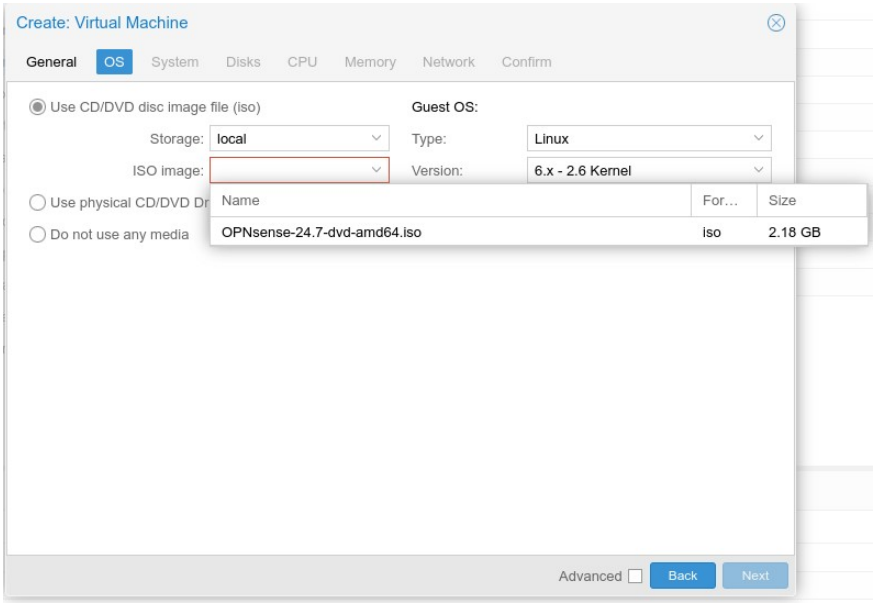
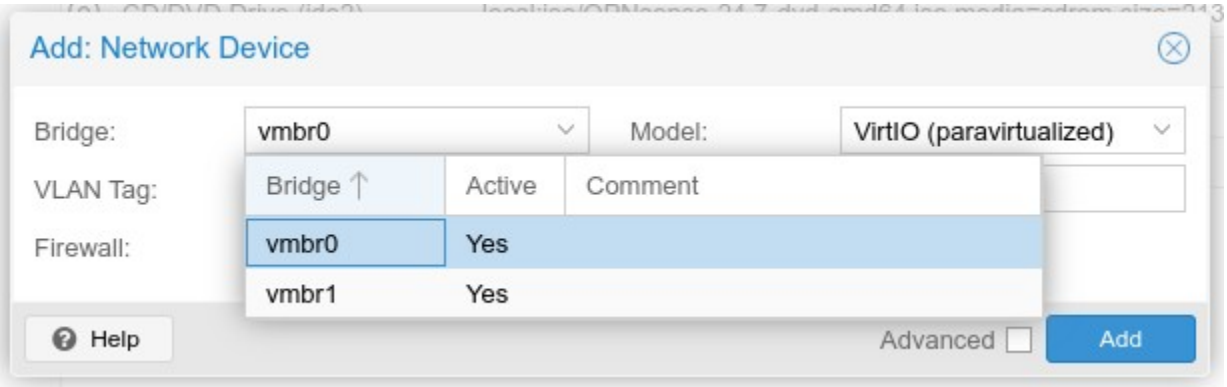


Figure 6: Create VM

ISO Image: OPNSense-xx.x-dvd-amd64.iso (from the drop-down list of ISOs you uploaded previously)
Memory: 4096 MiB
Processors: 4 Sockets, 4 Cores (total of 16)
Hard Disk: 50G

Leave everything else as default.

Once created, you still need to manually add the second NIC to the configuration, as the firewall needs to see both. Click on the “Hardware” tab of the guest’s sidebar, and then click the “Add” button to drop down the list of possible hardware additions, selecting “Network Device”. This should open a pop-up window in the browser. Select the other “bridge” (the one not already assigned to the guest), typically vmbr1.



2.6 Install OPNSense

Once a guest is created, you can boot the virtual machine by navigating to the virtual machine name on the left sidebar, then selecting “>_ Console” option on the virtual machine’s sidebar. If the machine is not running yet, it will then ask if you want to startup.

The first time you boot OPNSense, it will be running in “live” mode. At this point you are asked some configuration questions.

IP Addresses

Because we enabled two NICs, one for LAN and one for WAN, we need to now configure them. Ensure that the first NIC on the server machine is plugged into the local router, and that the second NIC is plugged into the ISP router (your internet connection).

Answer the following wizard questions;

- “Press any key to start the manual interface assignment:” <any key>
- “Do you want to configure LAGGS now?” <n>
- “Do you want to configure VLANs now?” <n>
- Enter the WAN interface name or ‘a’ for auto-detection: <Second NIC name>

As seen in the example, the second interface name for this server is “vnet1”.

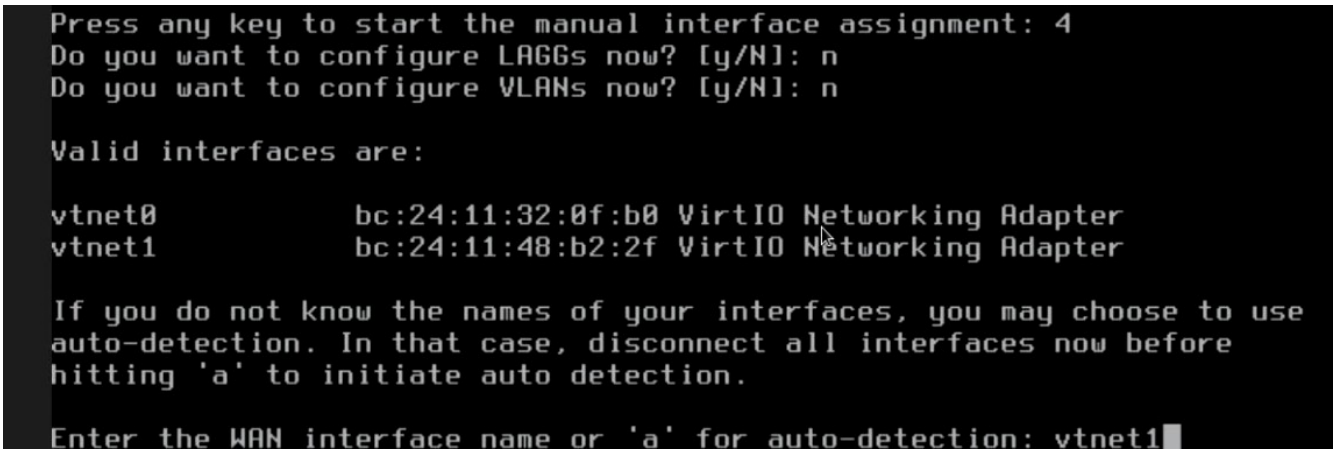


Figure 7: Specify WAN Interface

Then specify the LAN NIC:

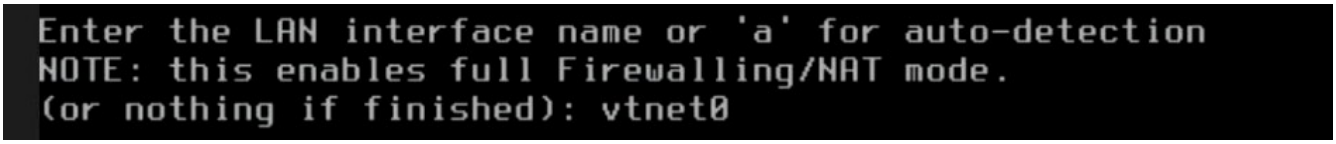


Figure 8: Specify LAN Interface

Once configured, the firewall will proceed to boot for the first time into “Live” mode. It should sense the IP address assigned by the ISP router to the WAN interface, and assign 192.168.1.1 to the LAN interface.

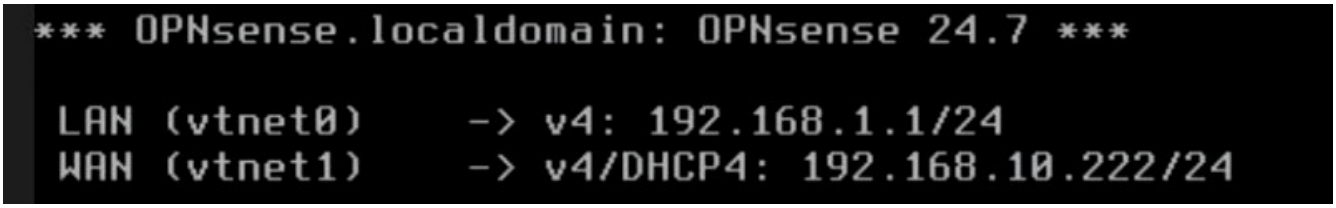


Figure 9: Network Assignments

We will need to change the LAN base address and DHCP settings later, in order to avoid conflicts with the ISP.

At this point, log in with the “installer” username and password “opnsense” which will start the installation wizard and use the previous answers for the network interfaces.

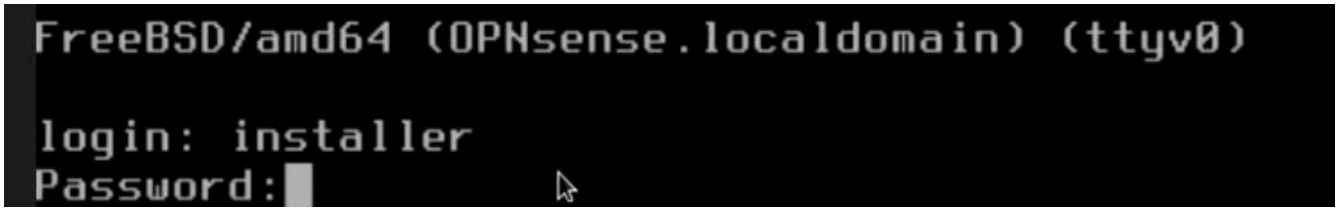


Figure 10: Installer Login

For most questions, the default answers should work.

Make sure to hit the spacebar to select the hard disk to install to - an asterisk should appear next to the disk.

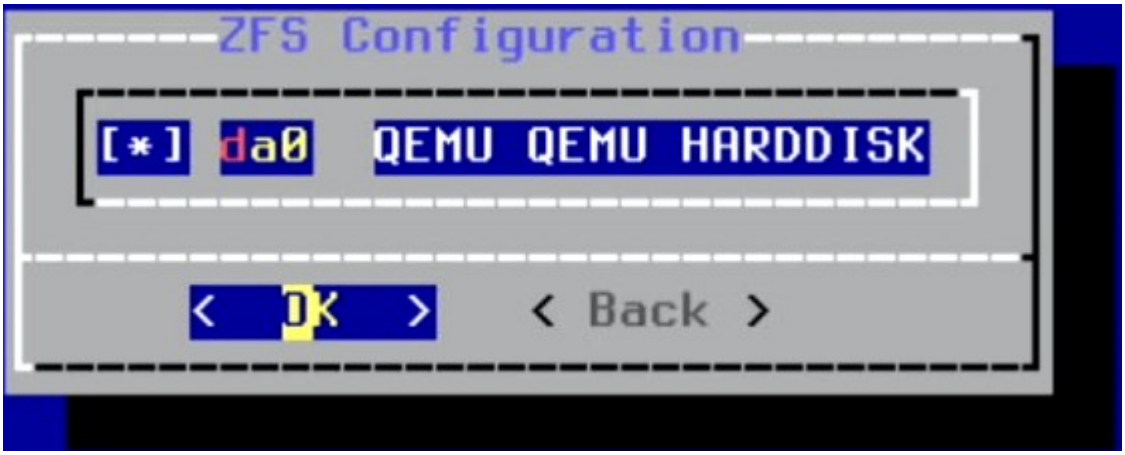


Figure 11: Specify Disk

Once complete, the firewall should reboot from the new installation, and let you log in. Use the root credentials to login this time. Then select option “2” in order to change the LAN IP address and setup the DHCP service pool.

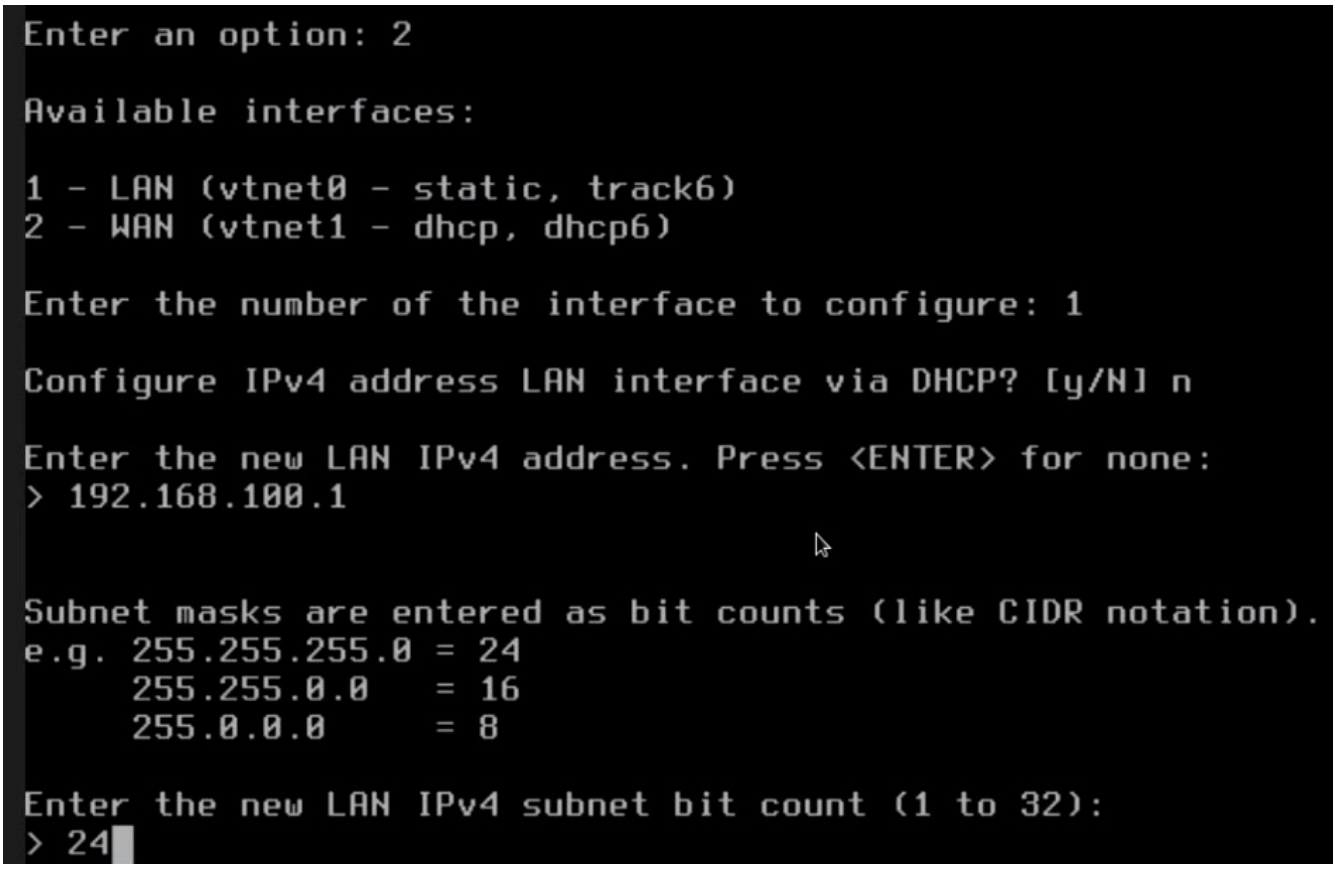


Figure 12: LAN IP Reset

- Make sure to use a 24-bit netmask (255.255.255.0).
- Do not configure IPv6 for now.
- Enable the DHCP Server on LAN, specifying the pool range from 200-254.

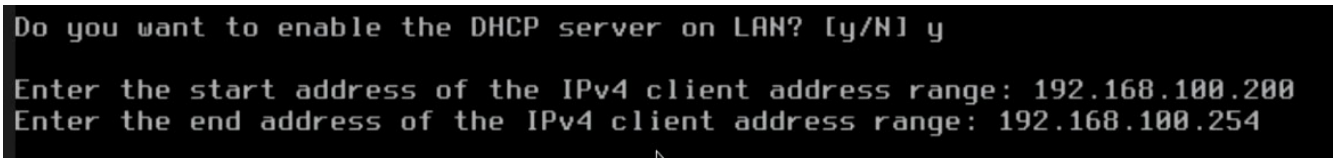


Figure 13: DHCP Pool Specification

Answer “n” for everything else, and let the server write the configuration. You should now be able to open another browser tab and log into the OPNSense management console.

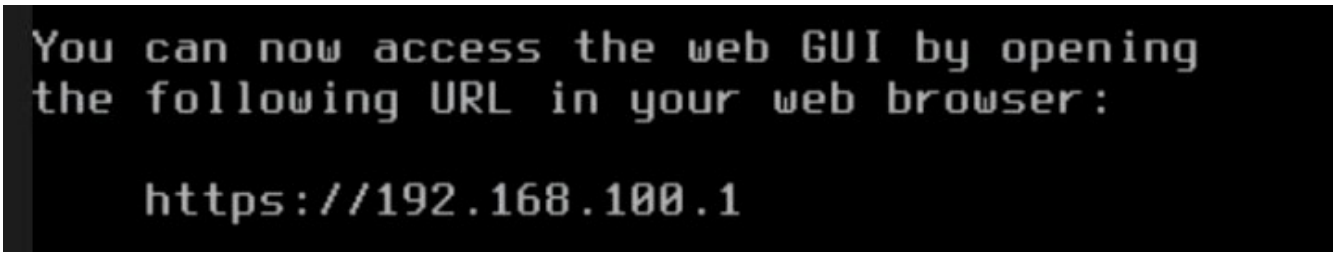


Figure 14: Firewall Console URL

Figure 15: OPNSense Login Screen

Finish the Installation

Open up a new browser tab and navigate to the URL specified above. This should bring you to a login screen, where you can login using the root credentials.

This first time you log in, the configuration wizard should start, letting you finish the configuration steps.

```
Hostname: firewall
Domain: <your preferred domain>
```

Note that the the name and domain that you specify will become the DNS entry for this firewall, allowing you to then access it via “firewall.<your domain>. In my case, this is “firewall.senseicto.com”

What Domain should I use? You do not need to actually register a domain. Since your guest machines and all devices on the LAN segment will be using this firewall as the Domain Name Server, you can specify whatever domain name you can imagine. The only time you need to register a domain is when

- You want to generate signed certificates to avoid the pesky “insecure access” messages from your browser, and/or

- You want to allow public access to your Hubs Cloud Community Edition server. (However, even this isn’t strictly necessary - we will discuss later.)

If you do register a domain, then use that domain name here as your preferred domain.

Update the Firewall

Navigate to System → Firmware → Updates, and select the Updates sub-tab to check the status of the firewall, applying any updates if available.

Final Configuration

Now that the firewall is running, we can start using it as a DHCP server and a DNS server.

First, let’s add the Cloud server to the DNS. Since it is not using DHCP, we need to add it as an override to the Unbound DNS service.

Navigate to Services → Unbound DNS → Overrides and click the “+” button on the far right. This should open a pop-up window that allows you to specify the Proxmox Cloud server name and IP address.

Use any name you want, but use the domain that you specified for the firewall. This is the base domain going forward for all guests that we create on this cloud.

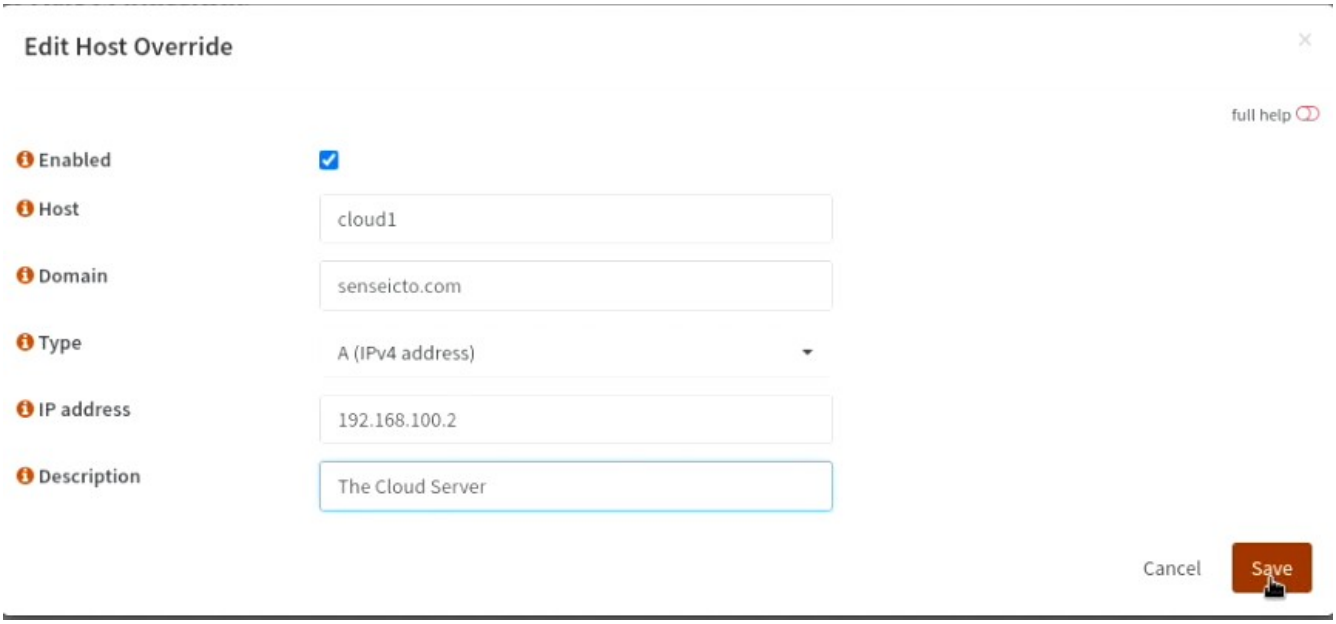


Figure 16: Add Cloud to DNS

Click the Save button, and then the Apply button on the bottom of the Overrides page.

You should also now be able to reset your workstation's network configuration to use DHCP. You should be assigned an address from the DHCP pool configured, between 192.168.100.200 - 192.168.100.254. Go back to the Workstation Network setting screen and change the settings to "Automatically obtain IP address…". Then restart your workstation - it should now be assigned an address from the pool, and should be using the DNS of the firewall for domain name resolution.

You should now be able to open a browser tab and navigate to the Proxmox Management Console via the name specified above, "https://cloud1.senseicto.com:8006" in this example.

Now that the Firewall is running, we want to make sure that it starts up every time the cloud restarts. This is done in the Proxmox console.

Navigate to Datacenter→Cloud1→Firewall and select the "Options" option on the Firewall guest configuration sidebar;

- Double click the "Start at boot" and select "Yes".
- Double click the "Start/Shutdown order" and make it "1"

This will ensure that the Firewall starts first, every time the Proxmox host reboots.

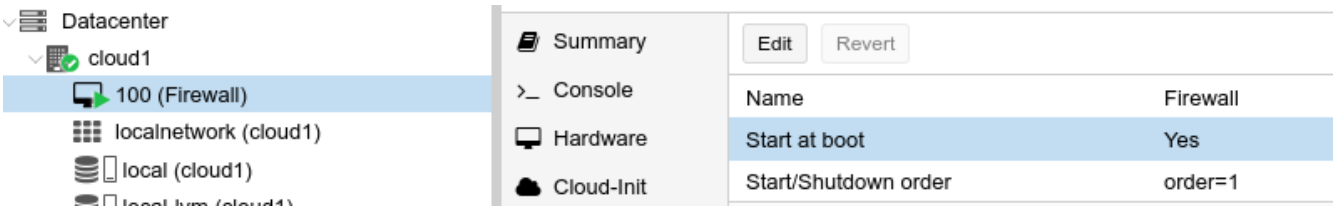


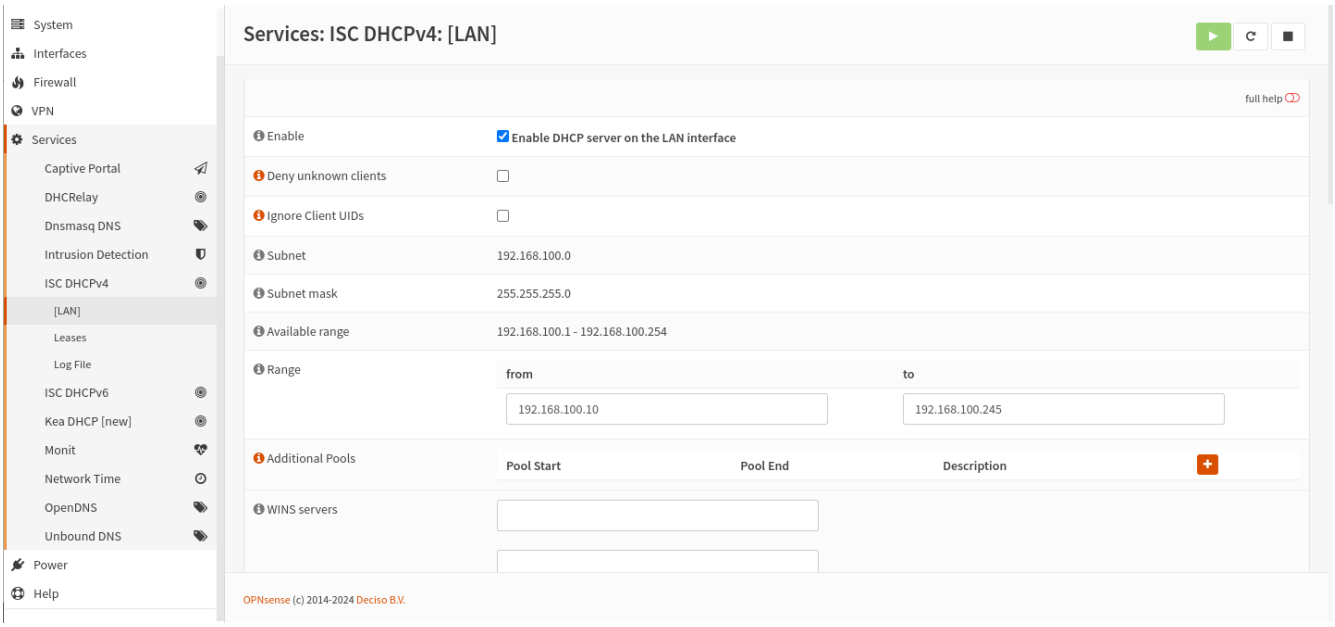
Figure 17: Configure Firewall Boot

Just like the Proxmox console, the firewall console should now be accessible via the domain name instead of an IP address. In this example, it is https://firewall.senseicto.com.

Configure the Firewall

Open up a browser and navigate to the OPNSense address we defined above. Log in using the administrator credentials that you specified for the firewall during the install. Now navigate to the DHCP configuration page at **Services→ISC DHCPv4→[LAN]**. You should see the DHCP configuration screen.

Reset the DHCP Range



Change the Range→from value, starting it at 200
i.e. 192.168.100.10 should be changed to 192.168.100.200 in the screen above.

Now scroll down to the Domain name, and enter the default domain name that you are using. Mine is “senseicto.com”.

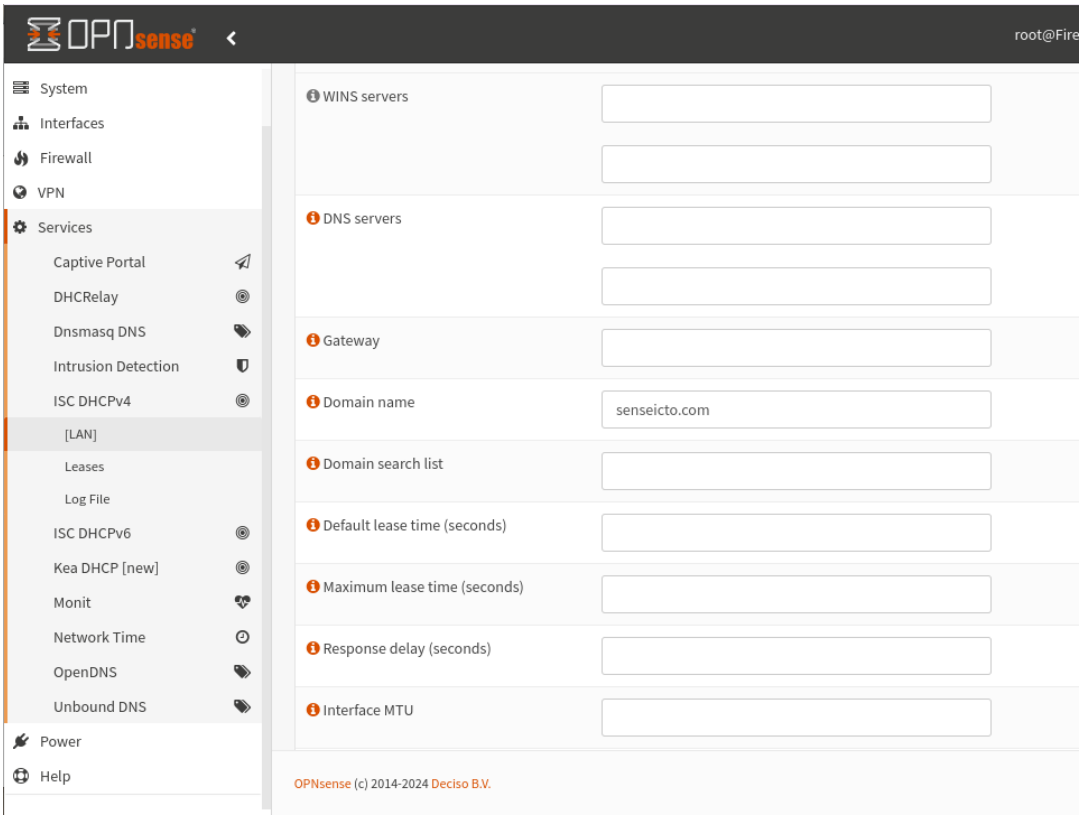


Figure 18: Default Domain Settings

Now scroll down a little further and click the “Save” button.

That’s it. We now have a working firewall that provides us with the DHCP and DNS services that we will need, amongst other.

Enable Static Mappings

Navigate to **Services → Unbound DNS → General** and enable the “Register DHCP Static Mappings” setting.

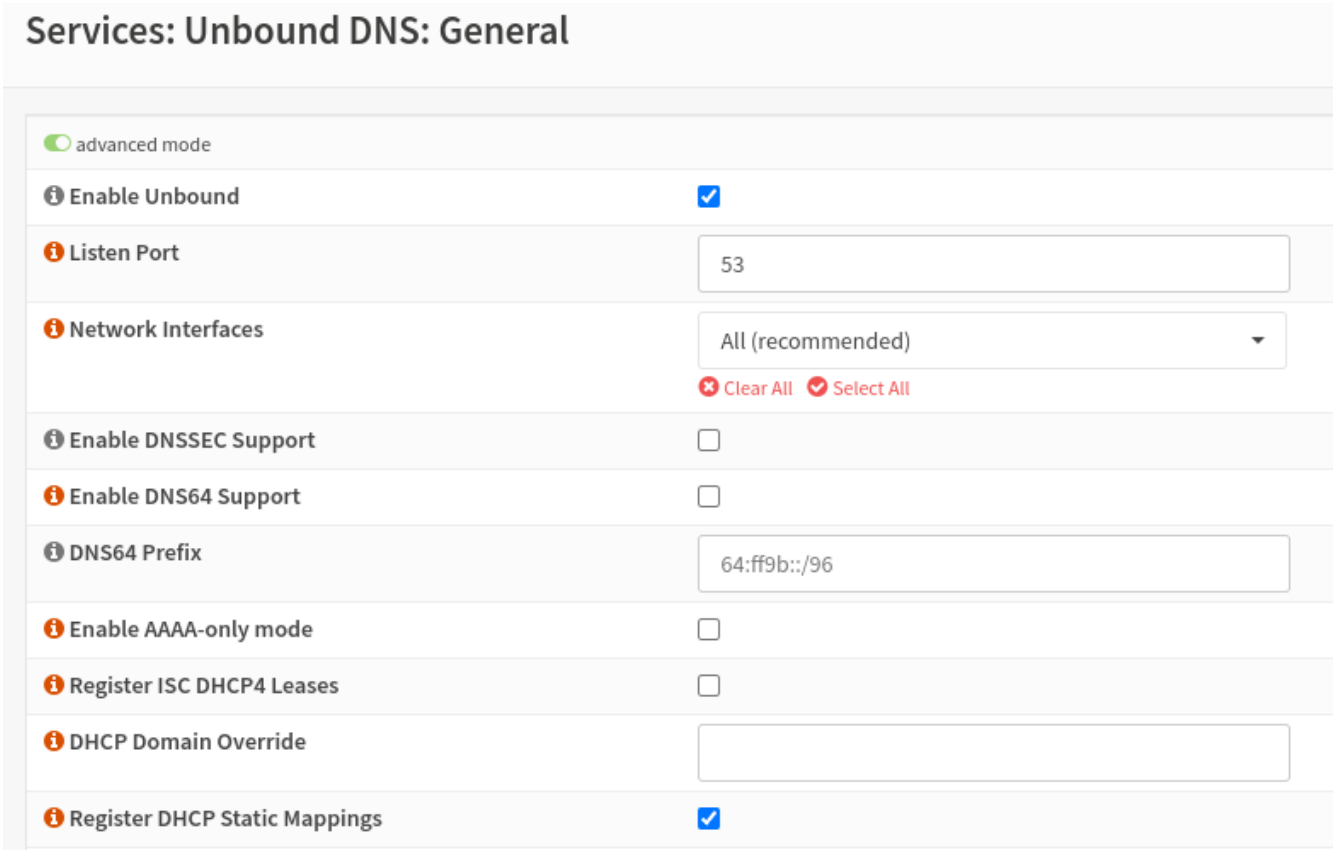


Figure 19: Register Static DHCP Mappings

Enabling DHCP Static Mapping Registrations in the DNS allows nodes on the network that use this firewall as a DNS to address other nodes via their hostname.domain rather than just their IP address. This is crucial for proper functioning of our server.

2.7 Mail Server

An important aspect of the Hubs Cloud Community Edition is authentication. By default, the server uses a generated “magic link” that is emailed to the user in order to let them authenticate themselves. No passwords to steal, or credentials to spoof.

How Does It Work? A “magic link” is a passwordless authentication strategy. It works by emailing a URL with a unique and time-limited embedded token to the user requesting access. If the user is properly registered, they will be sent an email with the link. When they click on the link, a new browser session is established to finalize the authentication. The original browser session will then automatically redirect to the Hubs room or service (e.g. Spoke, Admin console,…) that the user is trying to access.

In order to let our server work, we need to be able to email out these magic links. There was a time when we could have used a free email service such as Gmail in order to do this. However, email spammers ruined the party, forcing Google and other service providers to limit programmatic access to sending email.

Our workaround is to simply setup our own private email server. It will not be able to send real emails out to the public. However, all we need is the ability to get the magic link. We can setup the private email server to let the server send emails to registered users on the server, who can then use a web browser to login to the email server and retrieve their magic links. No real email delivery is required - it all stays self-contained on our private email server!

Using a Corporate Server. If your organization manages their own email hosting, they can probably provide credentials for the Hubs Cloud Community Edition server to use in order to send real magic link emails. Talk to your IT email administrator about this.

Public Access Authentication. If you intend to provide remote access to your Hubs server, you will need to be able to either;

1) Use your corporate Email Server as described above.

or

2) Have the user use an email address you control on the private email server, allowing you to retrieve the magic link email and click it on their behalf. Yes, this works. I will even show you how to let remote users log in to the email server and get their own magic link emails.

or

3) Register with an external email service provider who does allow you to programatically connect via SMTP to send emails. Note that you may still have issues with your ISP blocking SMTP traffic.

What we need is a simple, browser-based email server that we can install and manage without having

to register a domain. For our purposes, we will use Citadel (https://citadel.org), an easy-to-use, open source email server.

Citadel runs on an existing platform. We will be running it on a Linux Debian guest instance using the ISO that we downloaded previously.

2.7.1 Create a Debian Guest

Note: We will be creating several Debian guests for our server to use. As such, the other parts of this document will make reference to this section instead of repeating it.

Click the “Create VM” button on the top-right of the Proxmox console. This will open a wizard to guide you through the virtual machine creation. I recommend the following settings:

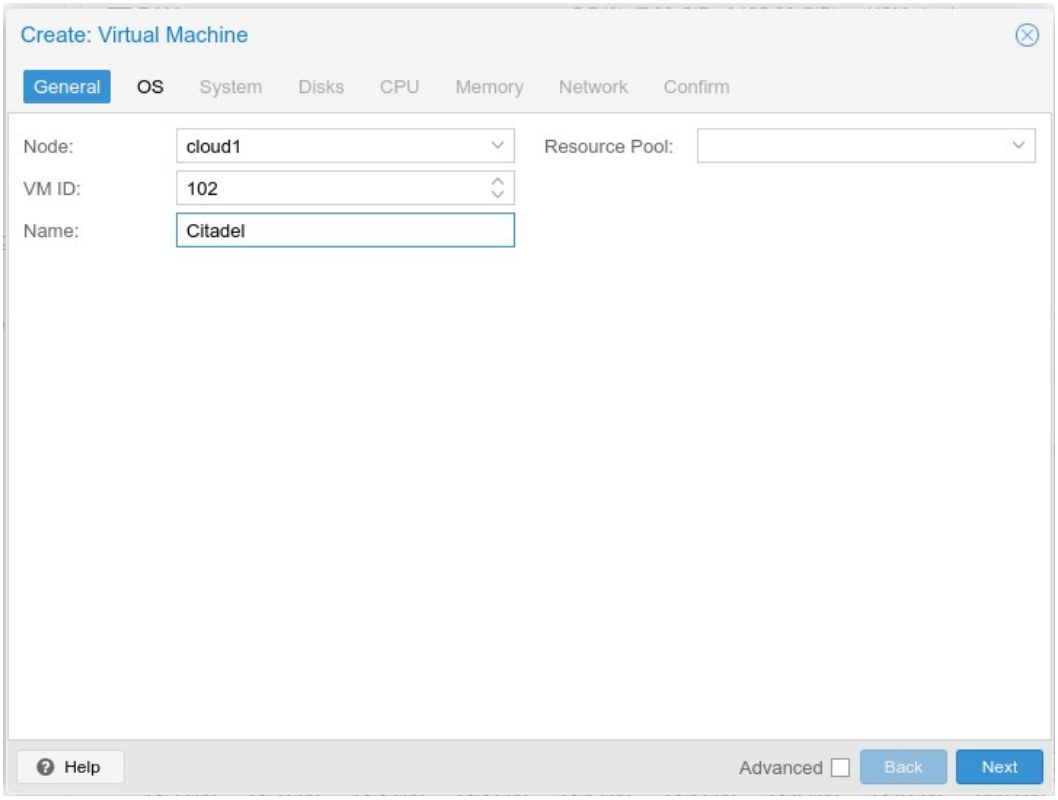


Figure 20: Proxmox Guest Name

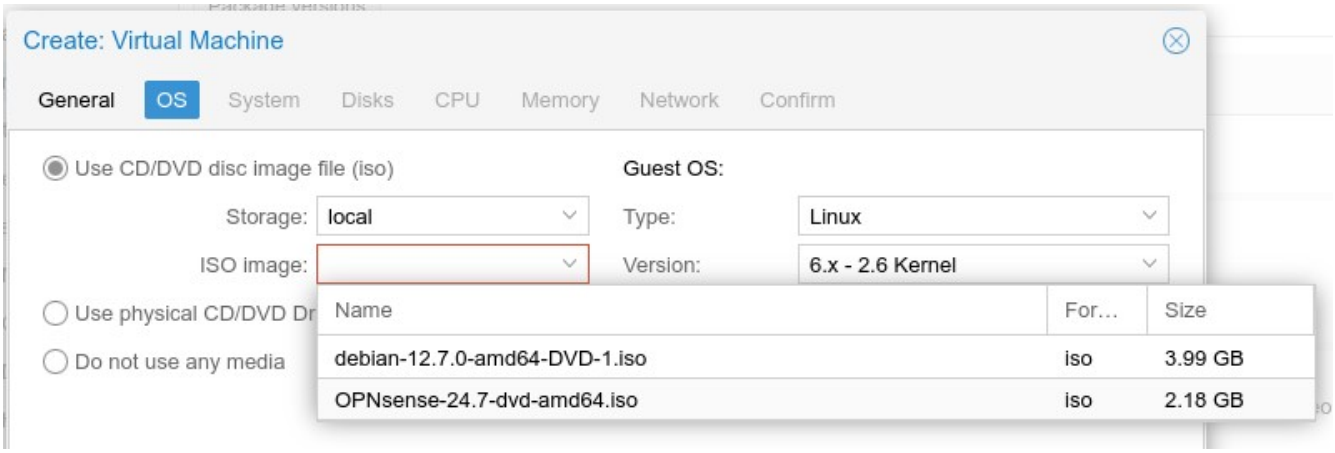


Figure 21: Debian Guest Creation

Name: Citadel
ISO Image: debian-xx.x.x-amd64-DVD-1.iso (from the drop-down list of ISOs)
Memory: 4096 MiB
Processors: 4 Sockets, 4 Cores (total of 16)
Hard Disk: 50G
Leave everything else as default.

Unlike the firewall, we do not need two NIC cards - the default vmbr0 is already connected to the LAN network segment, so we are good to go.

Before we start the Debian installation, we want to add it to the firewall's DHCP (Dynamic Host Configuration Protocol) server in order to have it assigned a proper network name, and static IP address.

Open up a browser and navigate to the OPNSense address we defined above. Log in using the administrator credentials that you specified for the firewall during the install. Now navigate to the DHCP configuration page at Services→ISC DHCPv4→[LAN]. You should see the DHCP configuration screen that we used to configure the DHCP server of the firewall. Scroll to the bottom, and click the “+” button in the DHCP Static Mappings section.

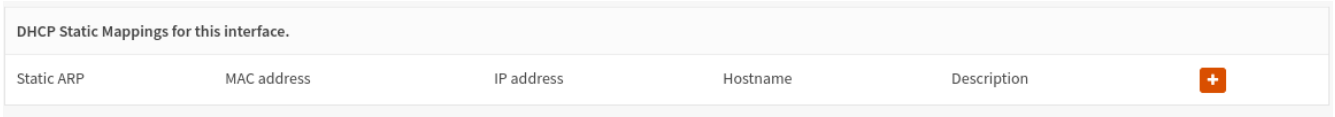


Figure 22: Add Static Mapping

A new window should now open, allowing us to add a static mapping for the Citadel email server.

Get the Guest MAC

We need to get the MAC address of the Debian guest that we are creating. In the Proxmox console window, we can click on the guest's Hardware sidebar option, and see the "virtio" value (MAC) for the Network Device.

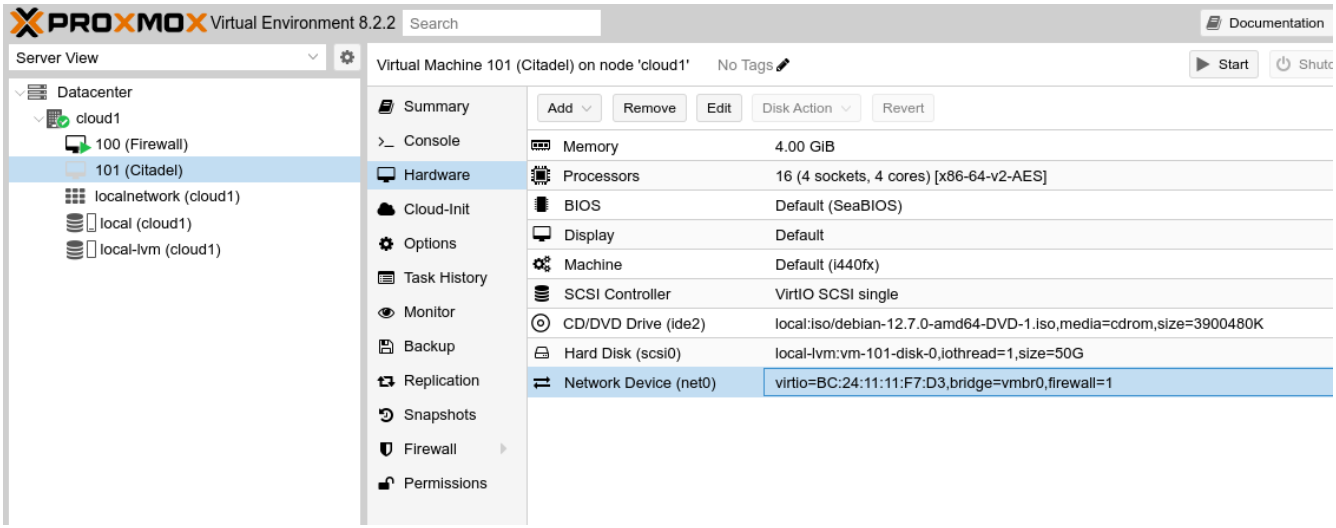


Figure 23: Guest MAC Address

Why Do We Need The MAC? Think of the MAC (Media Access Control) address as being the "actual" identity of the network device on the network. When the device first boots, if it is using DHCP to configure itself, the DHCP server maps this MAC address to an assigned IP address, and tells the device to use this assigned IP.

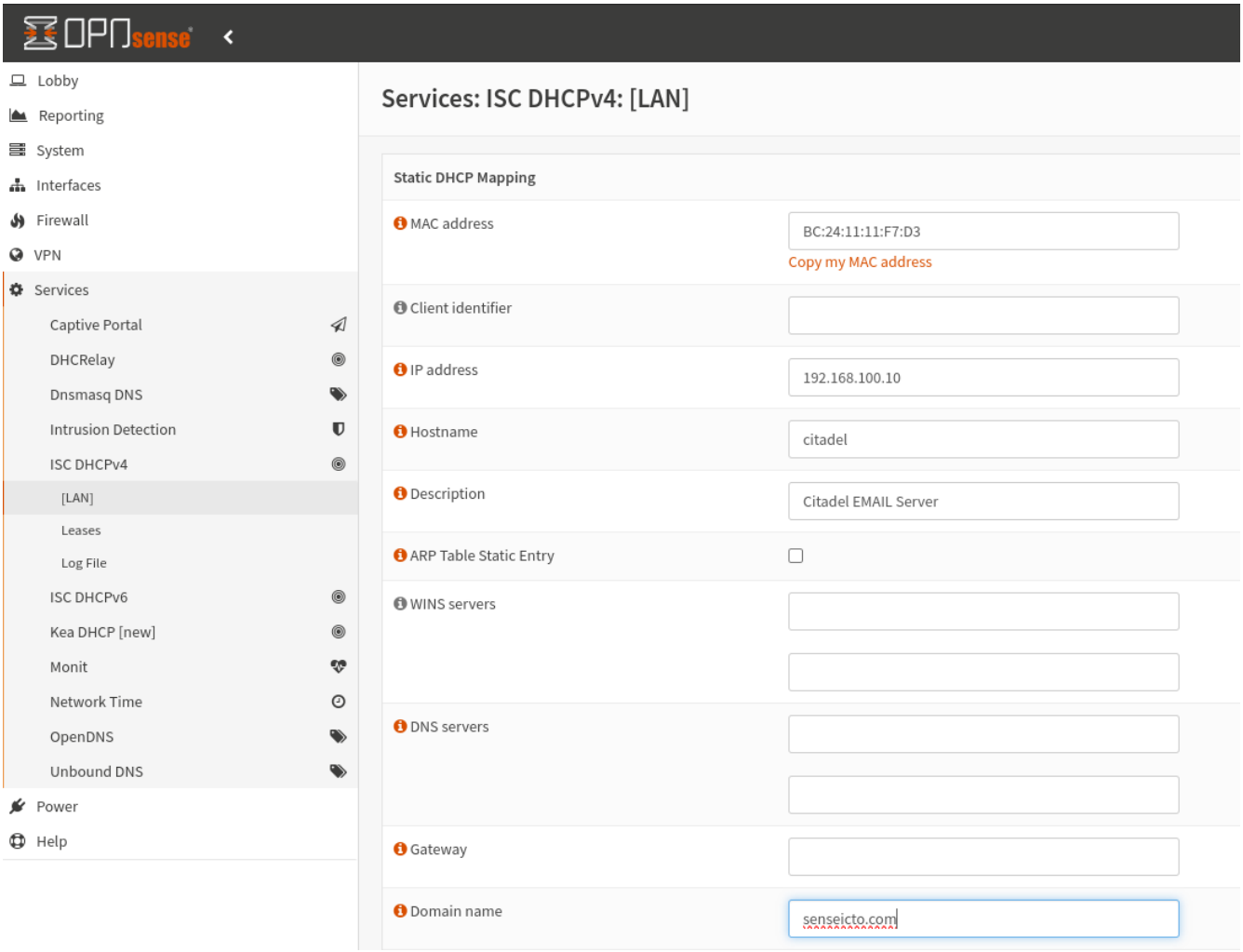
On top of this, the DNS server allows us to map a hostname and domain name to this IP address. This is how a network interface gets configured, and how these values are used to assign both an IP address, and a hostname, to a given device.

Take note of the virtio value. In this example, the Citadel NIC has a MAC of BC:24:11:11:F7:D3. This is the value we will use in assigning and identifying the server via DHCP and DNS.

In the firewall console, navigate to the static mapping screen (should already be open from above) and populate the fields. In this example, I used:

MAC Address: BC:24:11:11:F7:D3
IP Address: 192.168.100.10
Hostname: citadel
Description: Citadel EMAIL Server
Domain name: senseicto.com

Note: The IP Address, Hostname, Description should be unique to each instance. The above data is for illustrative purposes, using the Email server as the example.



Scroll to the bottom of the screen and click the “Save” button. It should return you to the DHCP configuration screen, listing the freshly minted static mapping at the bottom.

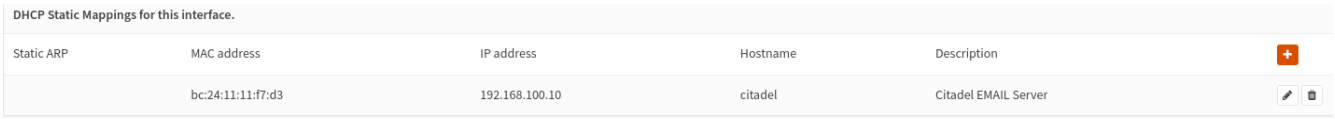


Figure 24: Citadel Static Mapping

We can now start the Debian installation for our Citadel server.

Start The Installer

Navigate to the new guest’s Console sidebar tab, and click the “Start Now” button.

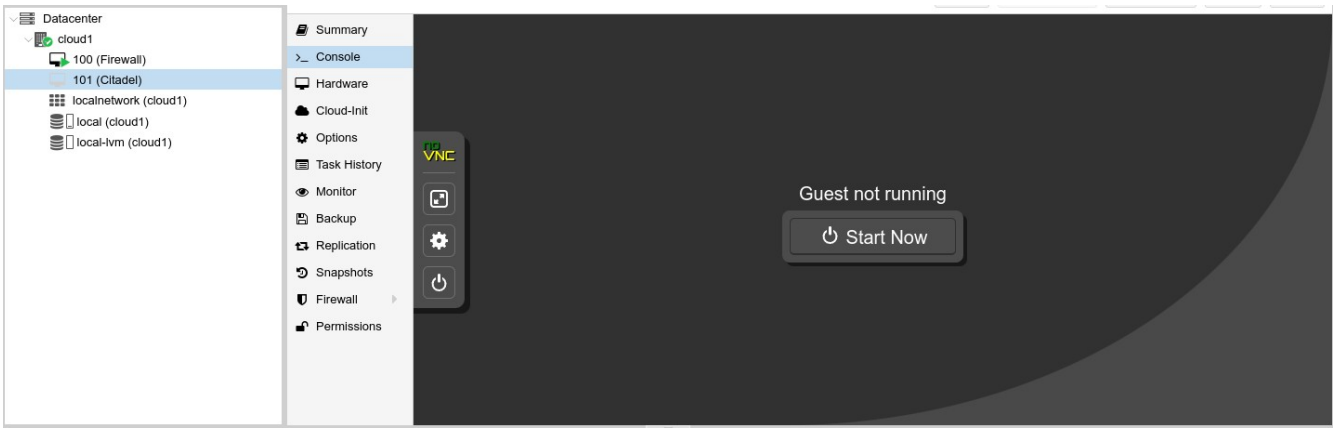


Figure 25: Starting a guest from console

You are now booting into the Debian installer. The first screen asks what type of install you want, and allows you to configure any additional OS-specific items. For our purposes, a default installation is all that we require.

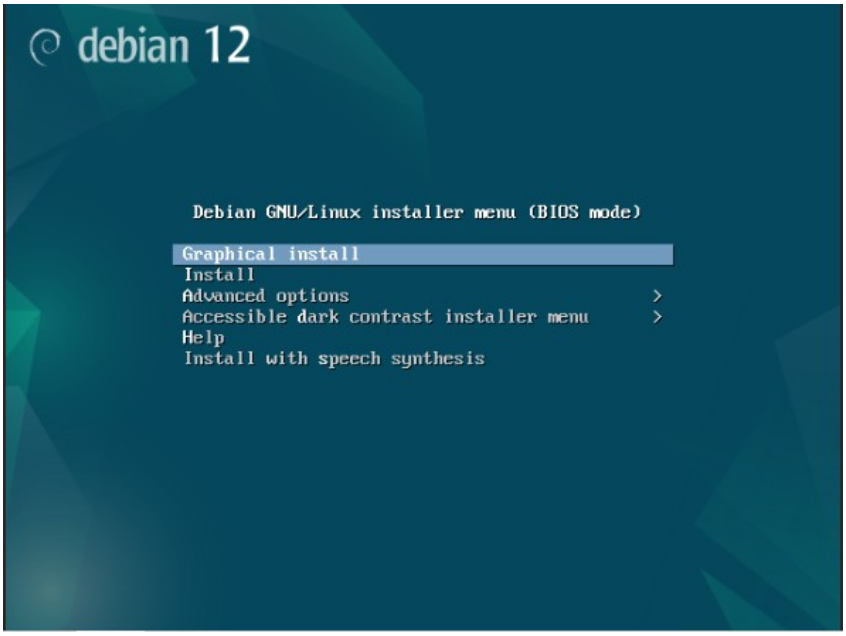


Figure 26: Debian Installer Menu

Select Language

While you are free to select any language, I would recommend English - English for now.

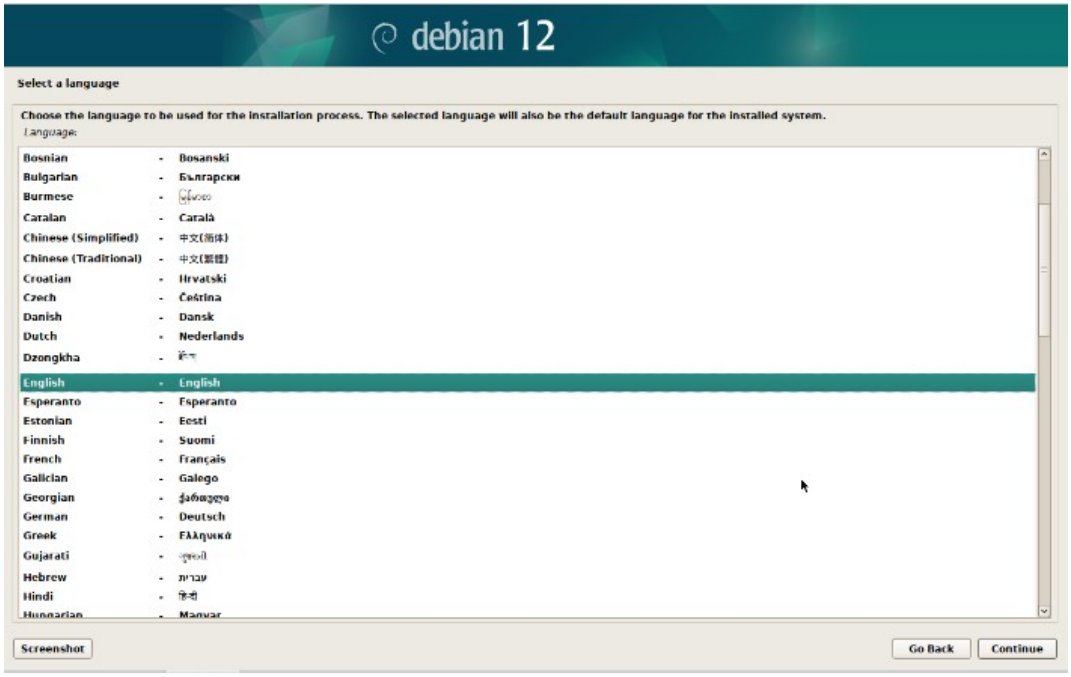


Figure 27: Select Language

Select Location

Again, while you are free to choose whatever country you want. I choose Canada.

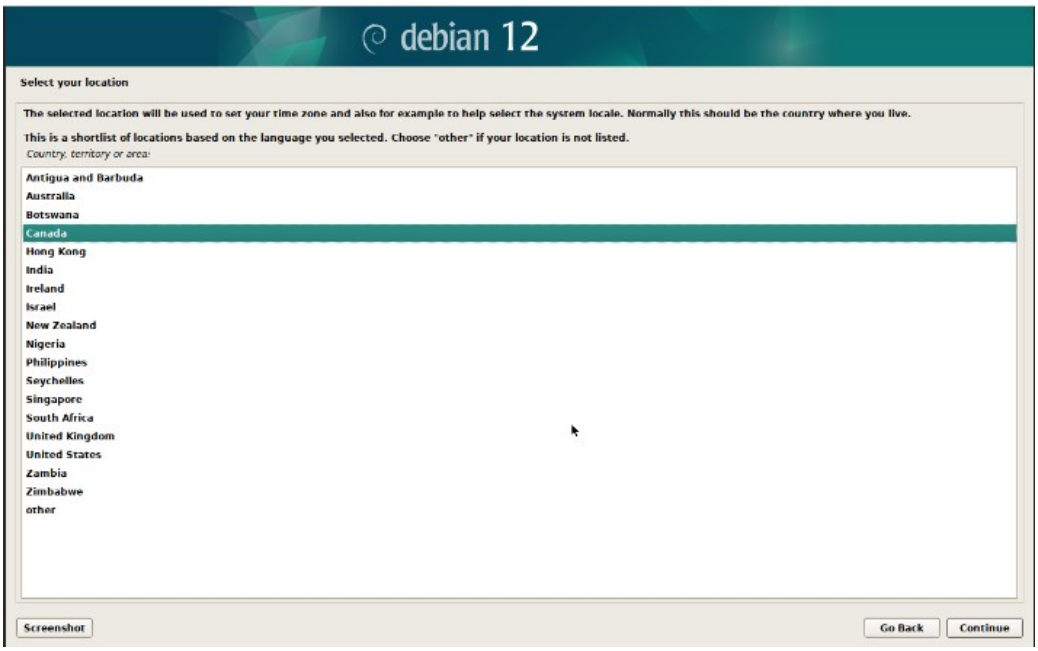


Figure 28: Debian Location Setting

Keyboard Selection

This one is tricky. You are using your own keyboard layout, so likely best to select it here.

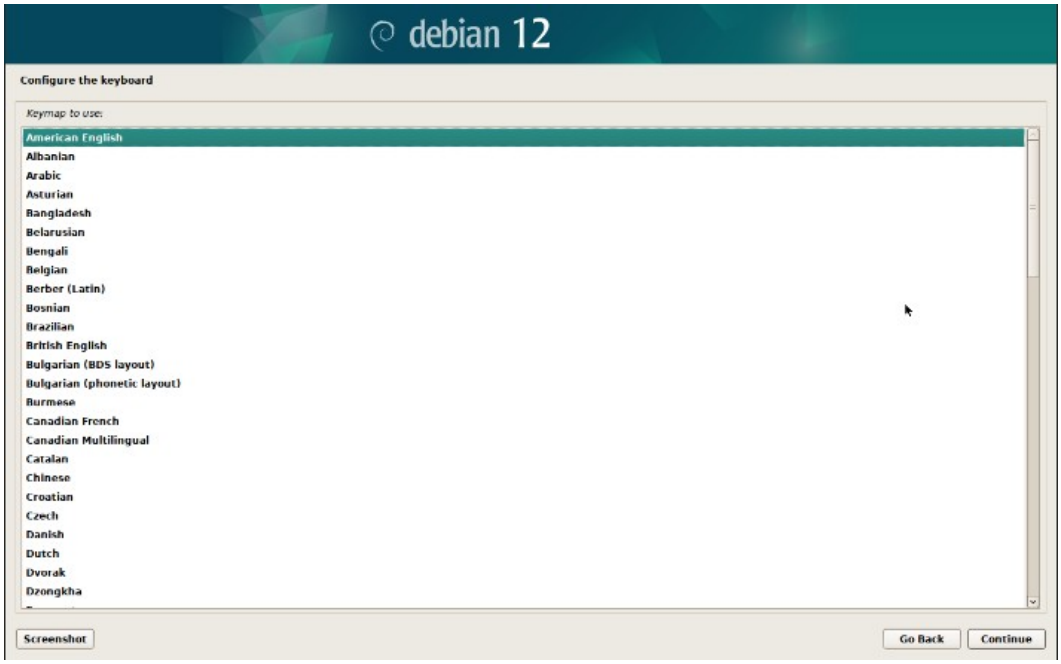


Figure 29: Debian Keyboard Layout

Network Hostname and Domain

If the firewall was configured properly with the DHCP static specification for this guest, the installer will retrieve this information from the DHCP server running in the firewall and populate the fields for you. In this example, the hostname “citadel” was retrieved, along with “senseicto.com” for the domain..

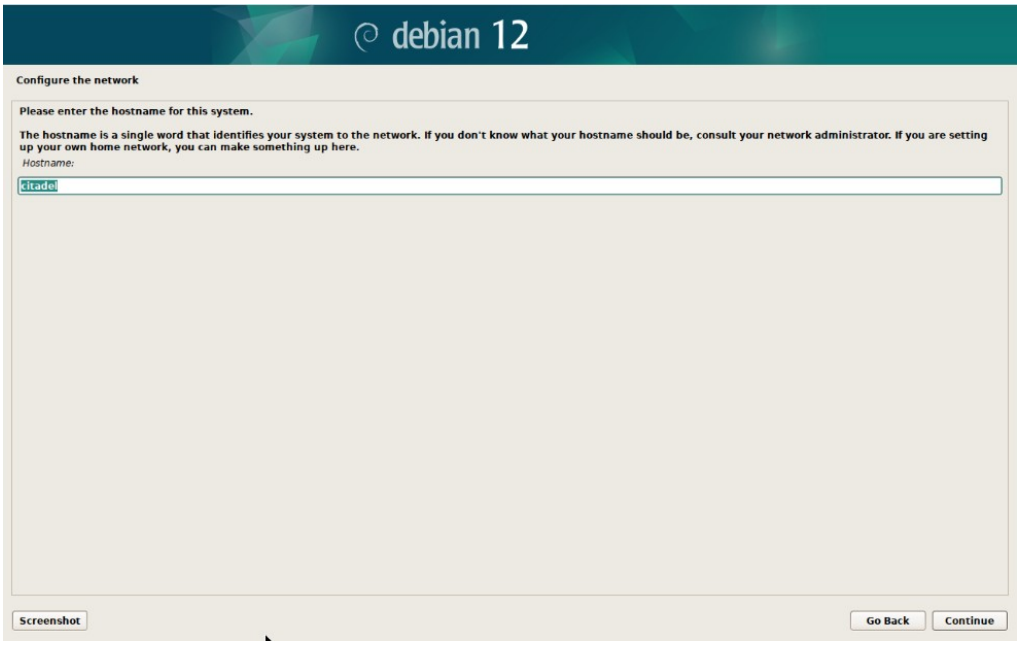


Figure 30: Network Hostname

Figure 31: Network Domain

Root Password

Specify the root password to be used as the operating system root user’s credentials.

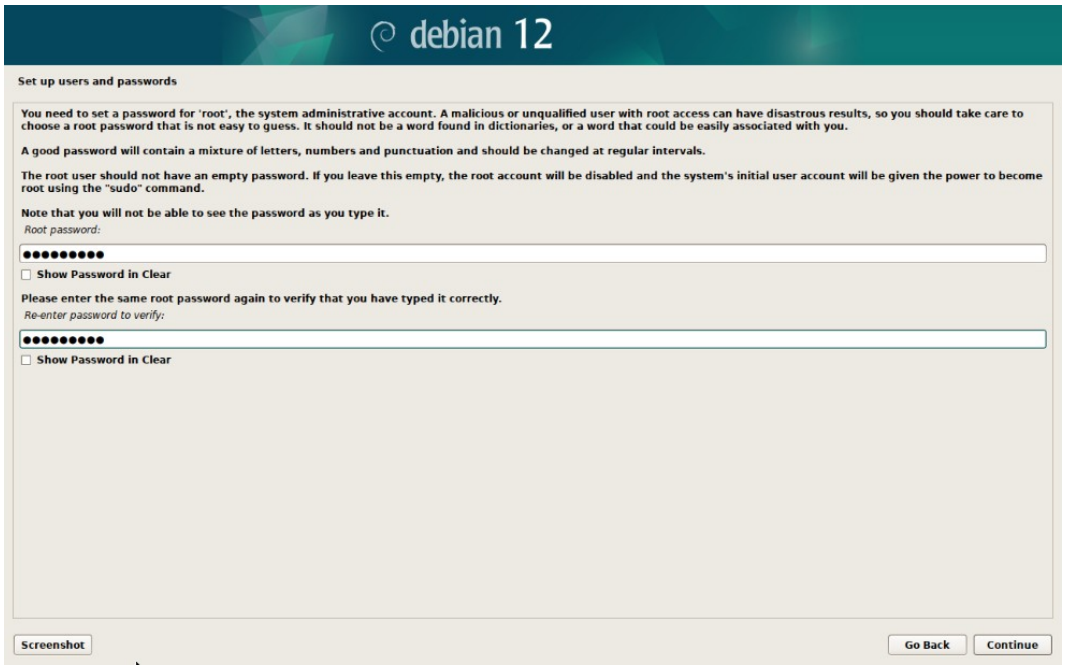


Figure 32: Root Password

Setup Users and Passwords

Create your own basic operating system account credentials. You will use this as your default login credentials, especially remotely via SSH.

Figure 33: Create User

Setup Clock

This configures the system clock. This is important as the system clock is used by the updater amongst other programs and services.

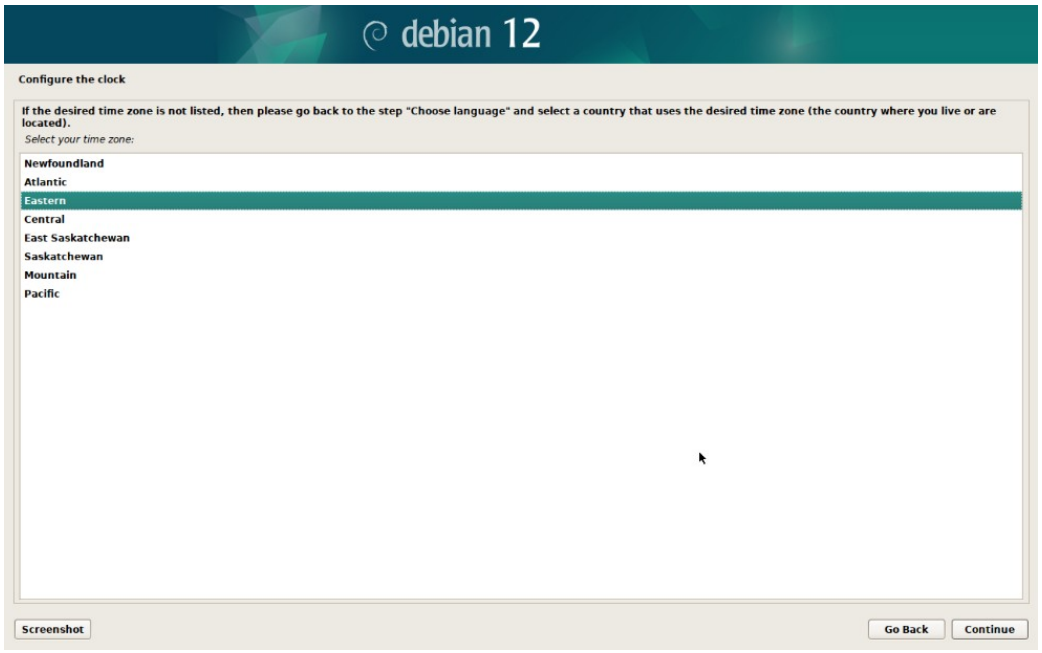


Figure 34: Setup Clock

Partition Disks

Select “Guided - use entire disk”.

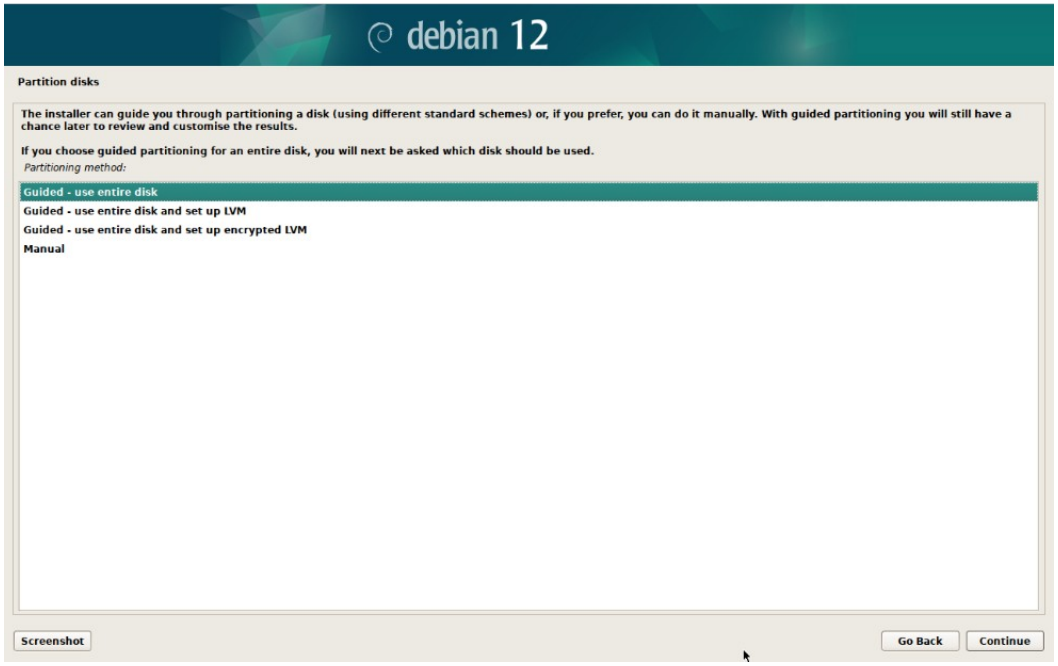


Figure 35: Partition Disks

Select the disk listed. This should match the disk specified when we created the guest in ProxMox.

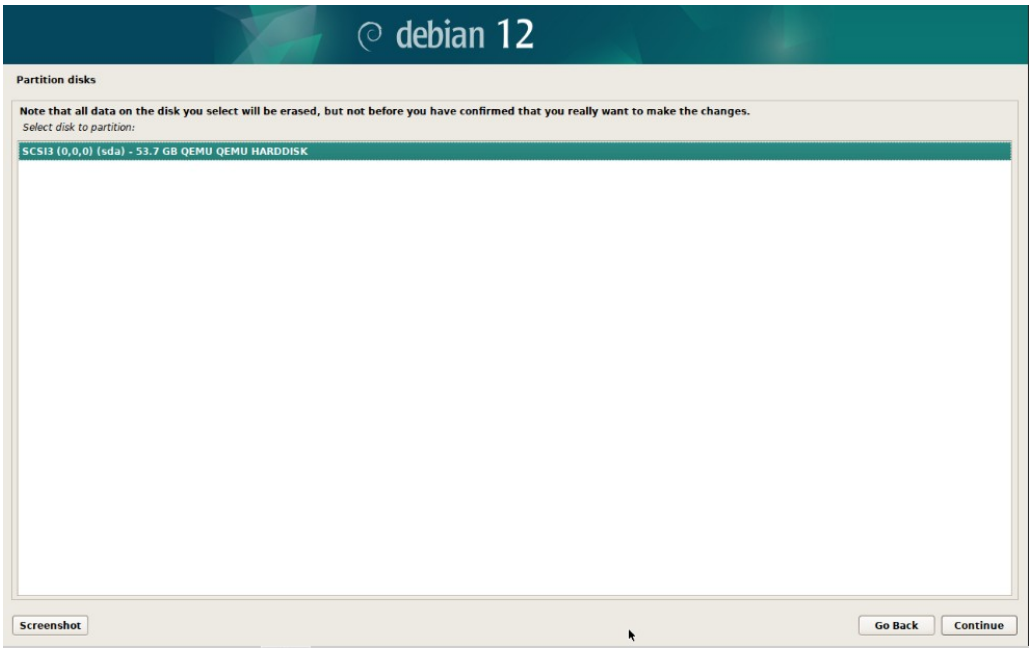


Figure 36: Disk Selection

Place all of the files in one partition.

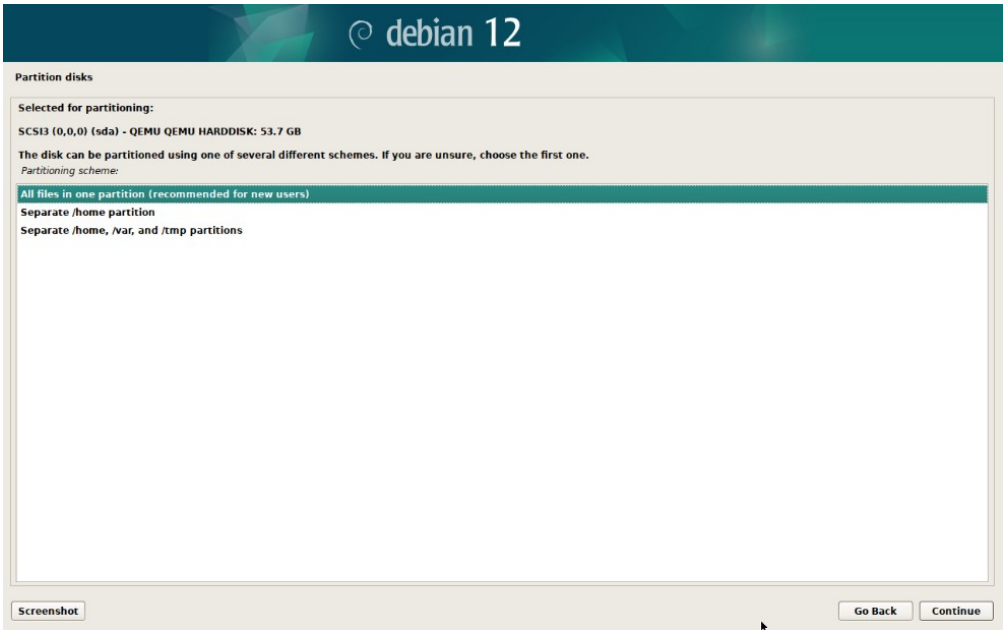


Figure 37: File Placement

Finish Partitioning

Finish the partitioning and write the changes to the virtual disk.

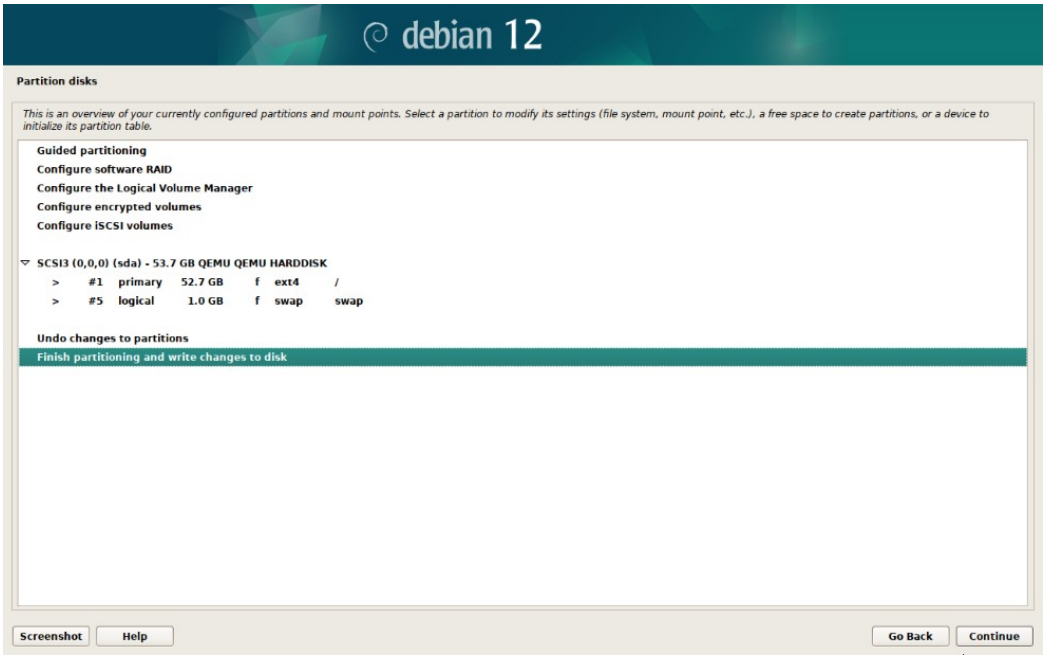


Figure 38: Finish Partitioning

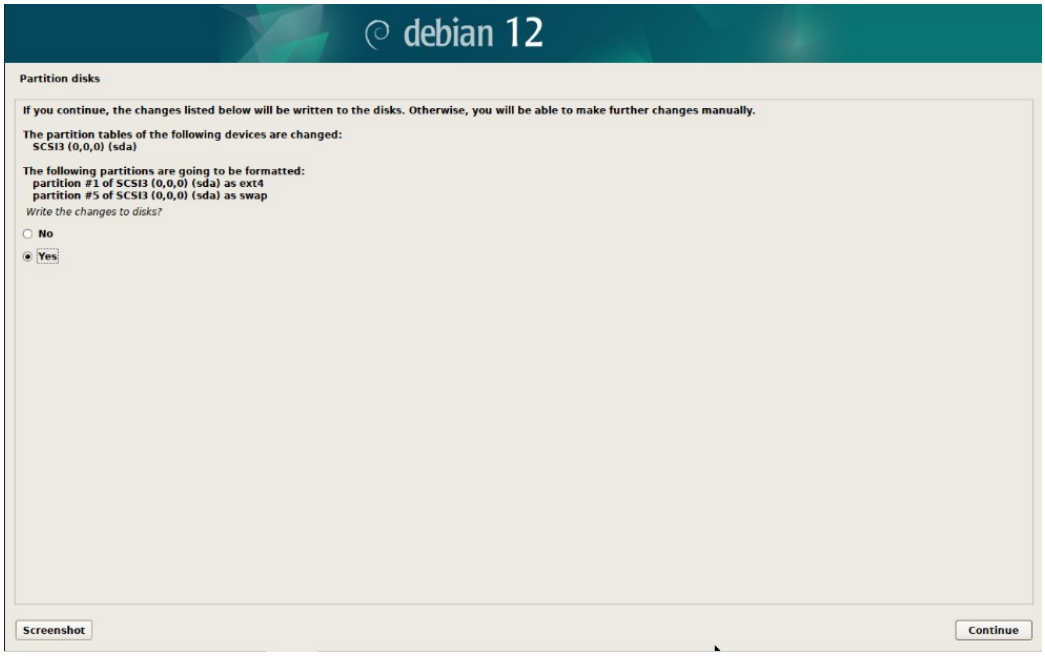


Figure 39: Confirm Writing Changes

At this point, the installer will start the base system installation.

Configure Package Manager

The Debian installer will retrieve installation files from a variety of location, including more local disks, as well as network repositories. In this case, we do not need to scan more media, as we will be letting the installer automatically retrieve the most recent versions of files from the online repositories.

Leave the default selection of “No”.

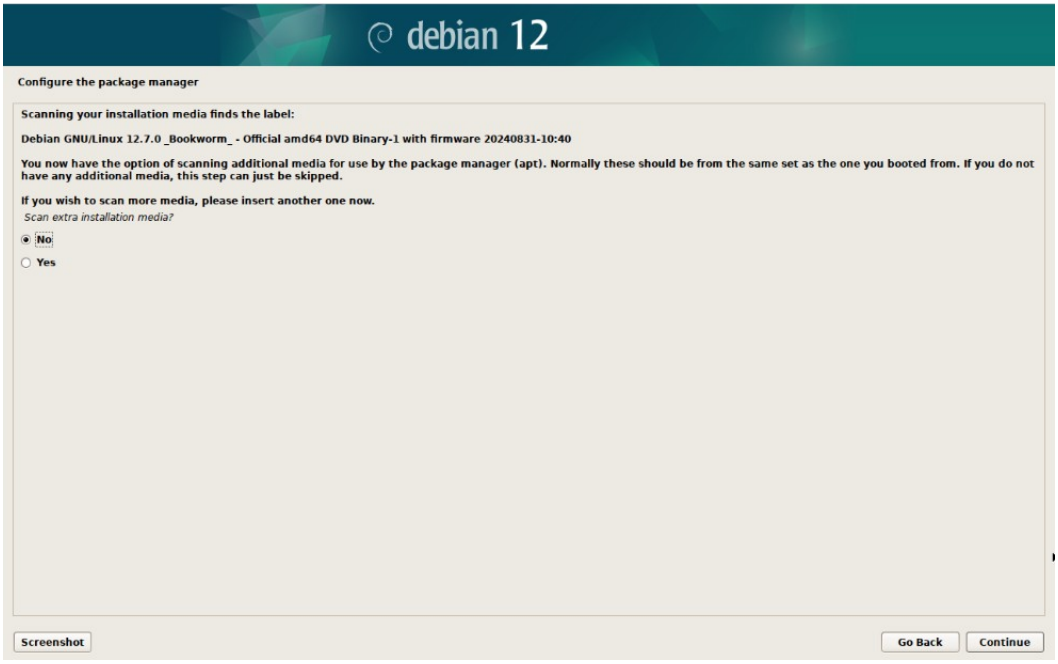


Figure 40: Scan Extra Media

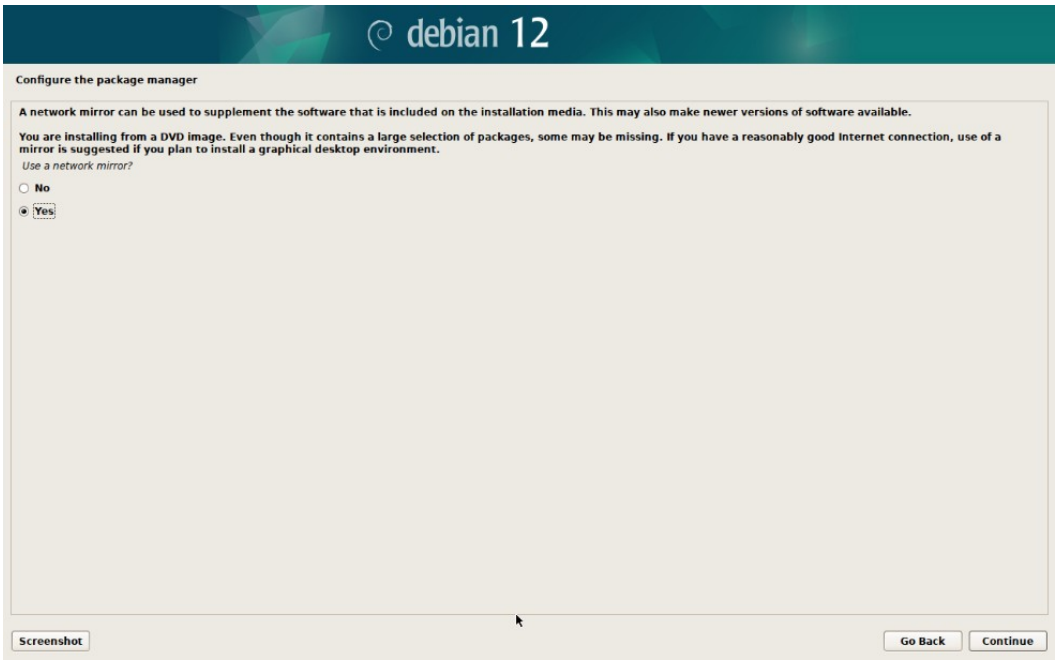


Figure 41: Use Network Mirror

As described earlier, we will let the installer retrieve the most up-to-date files from the internet repositories. Select “Yes” to access the network mirror hosting the files online.

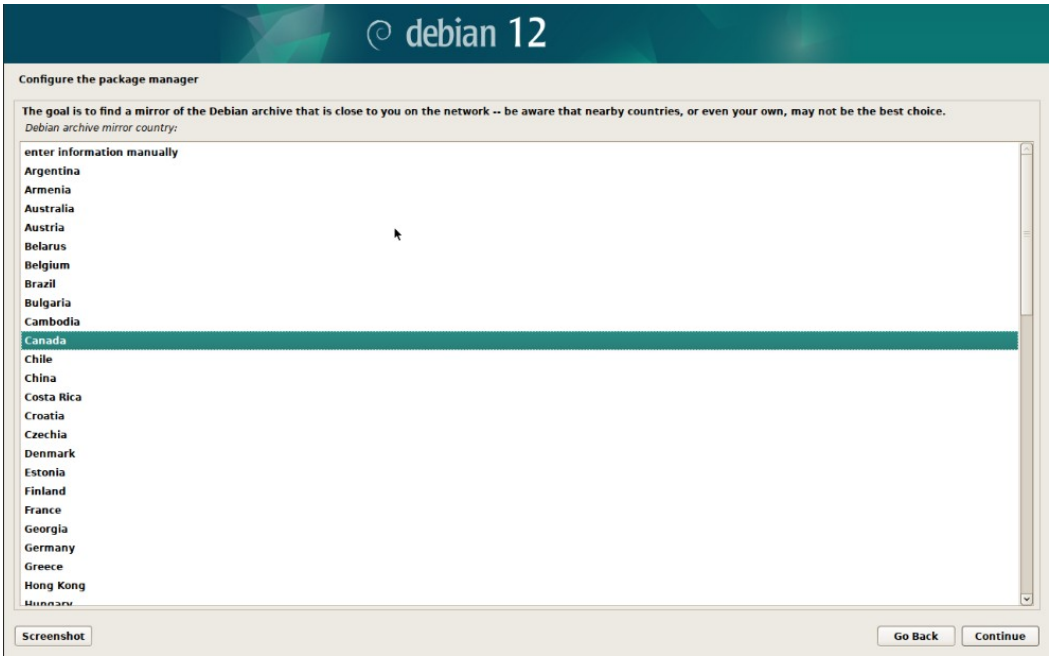


Figure 42: Select Mirror Country

Select the country closest to your location.

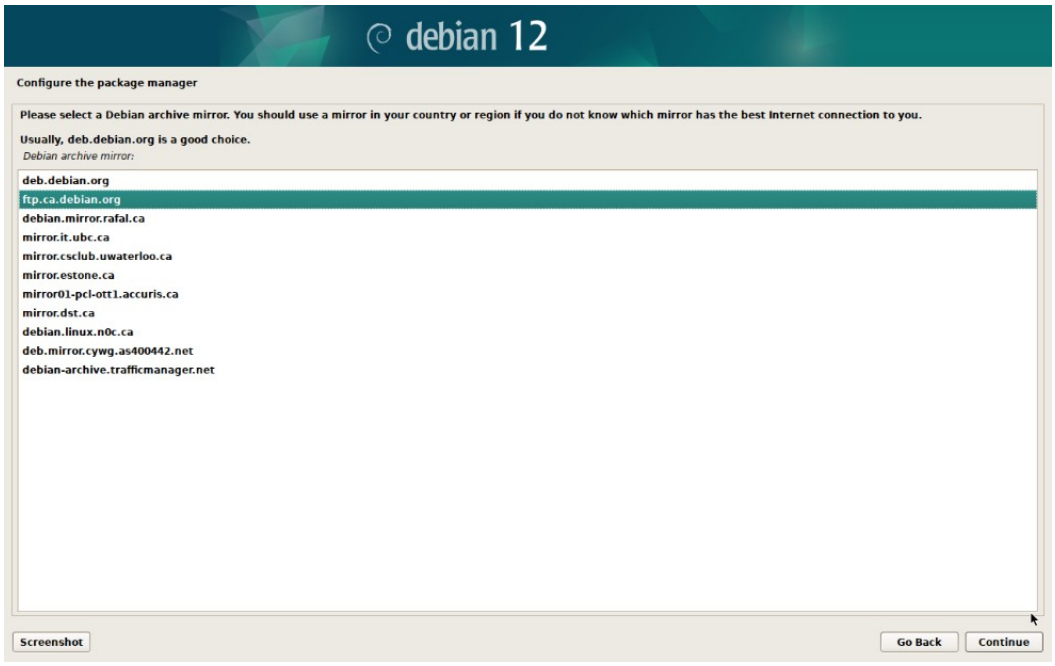


Figure 43: Mirror Selection

Select the mirror in your selected country.

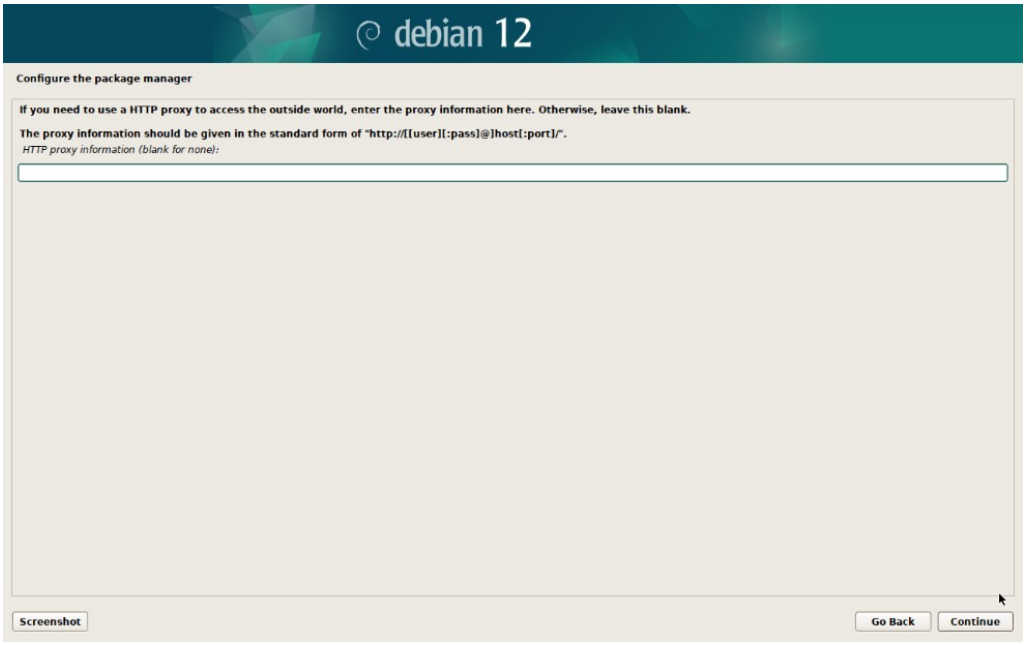


Figure 44: HTTP Proxy Specification

Assuming you are installing where your server is directly connected to your ISP router, there is no need to specify an HTTP proxy - simply hit Continue. However, if you are standing this server up

behind a corporate firewall and network, you may need to specify an HTTP proxy in order to access the internet. This may be specified in your browser settings, or your network configurations.

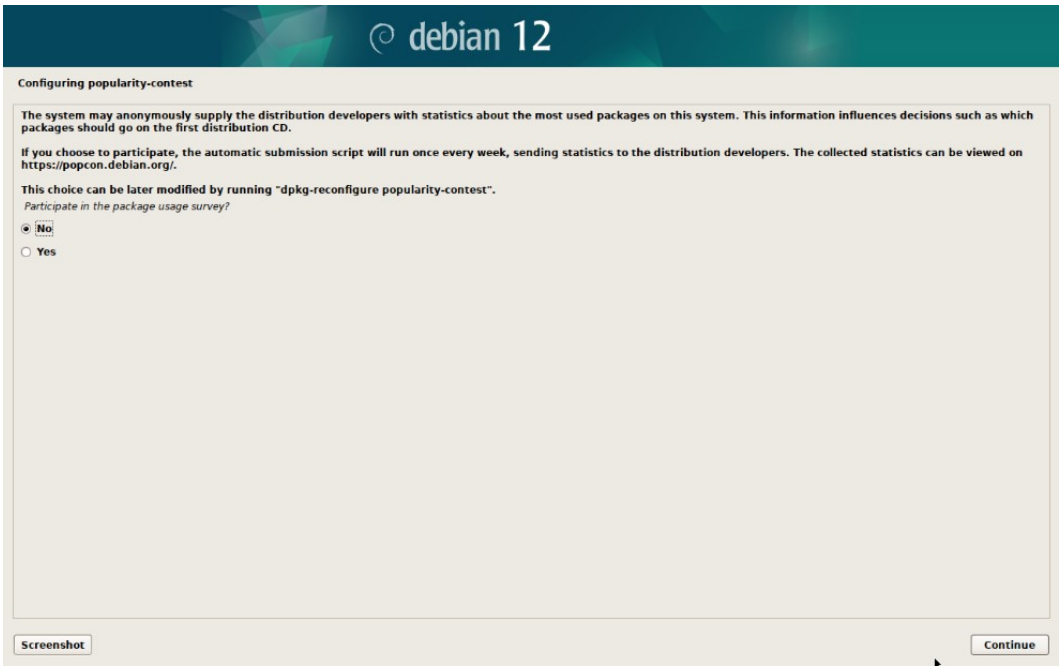


Figure 45: Popularity Contest Configuration

One option during installation is allowing the operating system to anonymously provide usage statistics to the Debian developers. This is not dissimilar to what Windows, MacOS, and other operating systems do. However, the main difference is that Debian allows you to opt-out entirely, unlike Windows that will always provide a base amount of information to Microsoft - they call it “Required Data”.

I like my privacy - just say “No”.

Software Selection

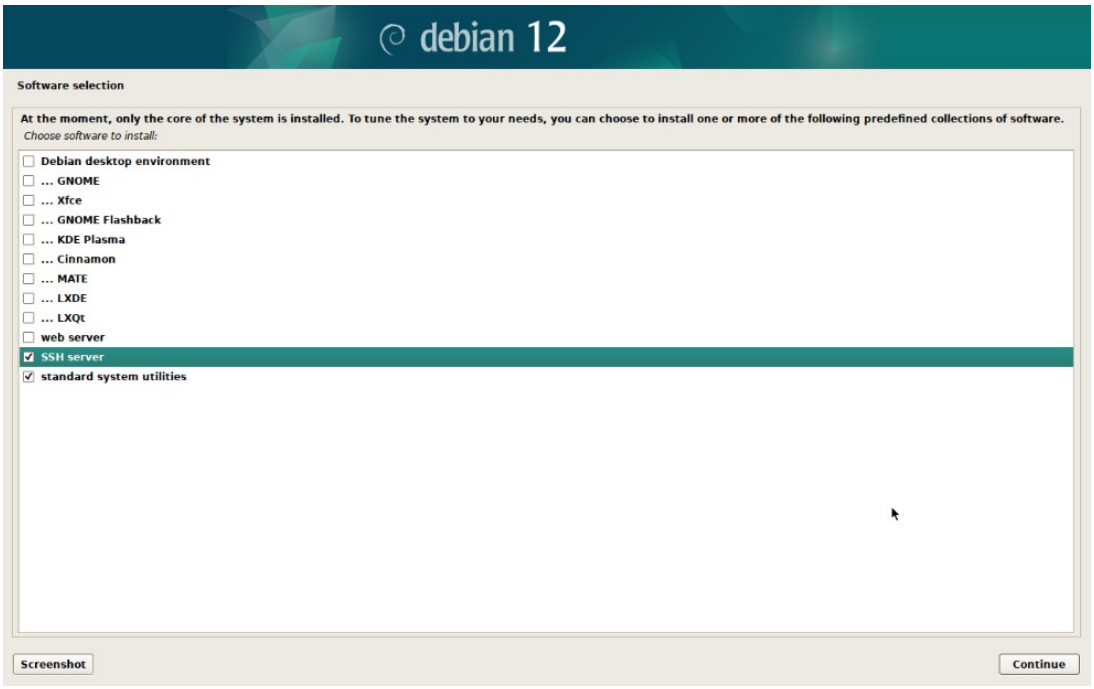


Figure 46: Software Selection

We are at the most important stage - the actual system software options screen.

The majority of the options relate to the type of desktop that you would like to run. Unlike Windows or MacOS, Linux has a broad range of desktop styles and themes to choose from. Some, like GNOME, are even configurable to let your Debian workstation look and feel like a Windows or MacOS desktop!

While we will revisit this screen later on in the server build when establishing the Hubs Installer Workstation, the majority of our Debian instances will be running without a graphical desktop. Because they are just servers, there is no need to incur the overhead of a full GUI desktop.

For now, simply select

- SSH Server - this will allow us to log into the instance remotely.
- standard system utilities - we will be using these as well.

GRUB Boot Loader

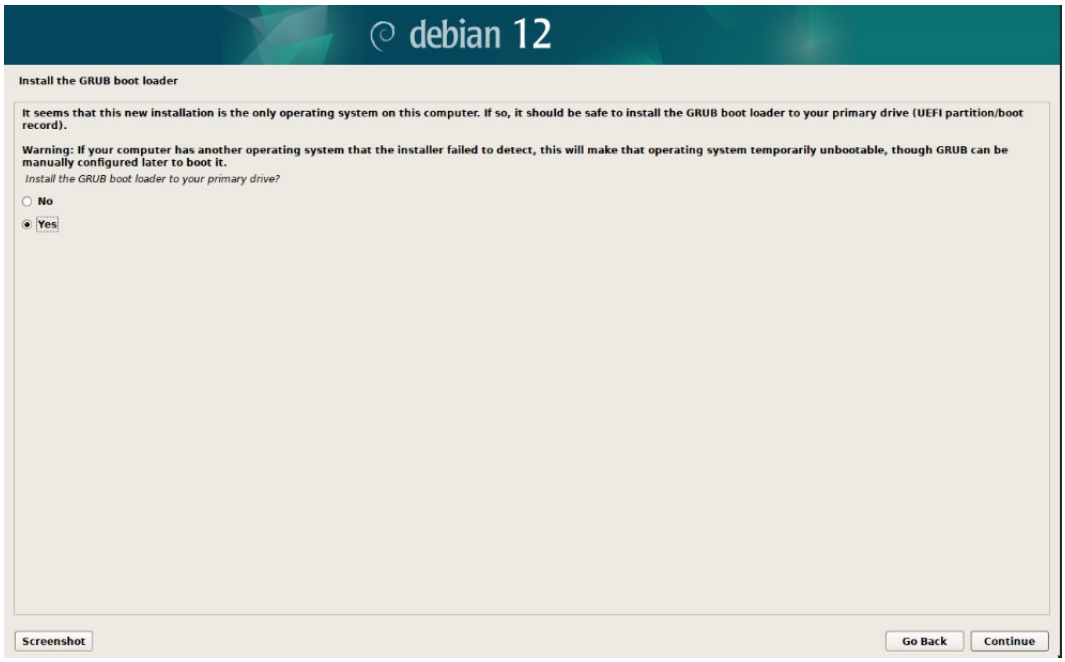


Figure 47: Install GRUB Boot Loader

Allow the installer to configure the virtual hard disk to boot Debian. Select “Yes”.

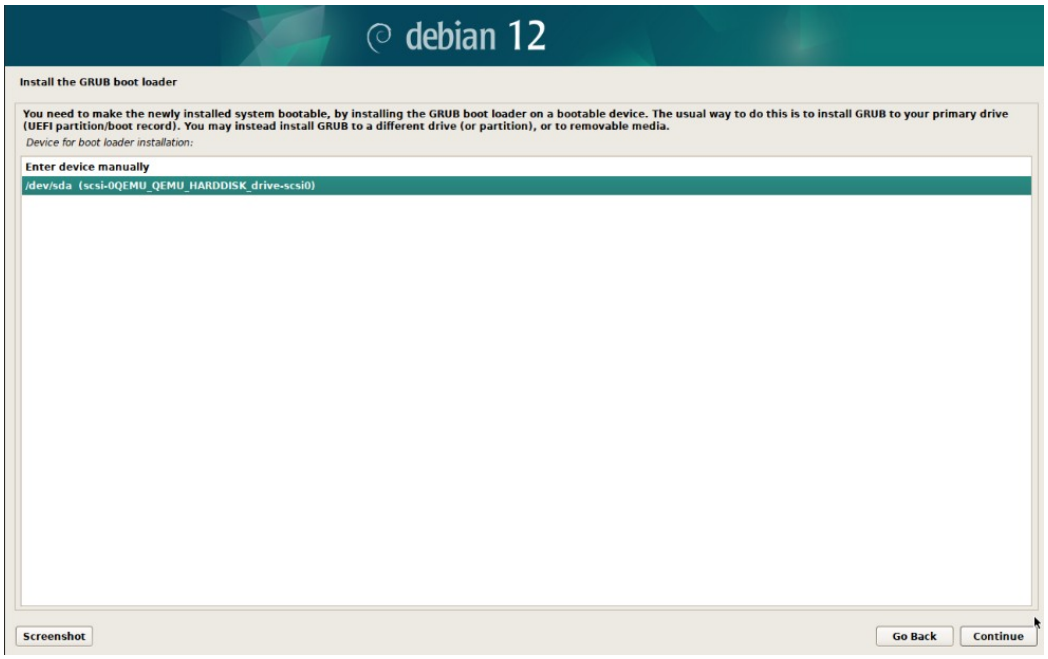


Figure 48: Specify Boot Disk

Select the disk (there should be only one) to which the boot loader should be installed.

You can now finish the installation and let it reboot.

Edit Sources.list

In order to keep updates clean, edit the sources.list file. In the guest console window, log into the guest as root, and edit the file:

```
# nano /etc/apt/sources.list
```

Add a # in front of the top line in order to comment it out.

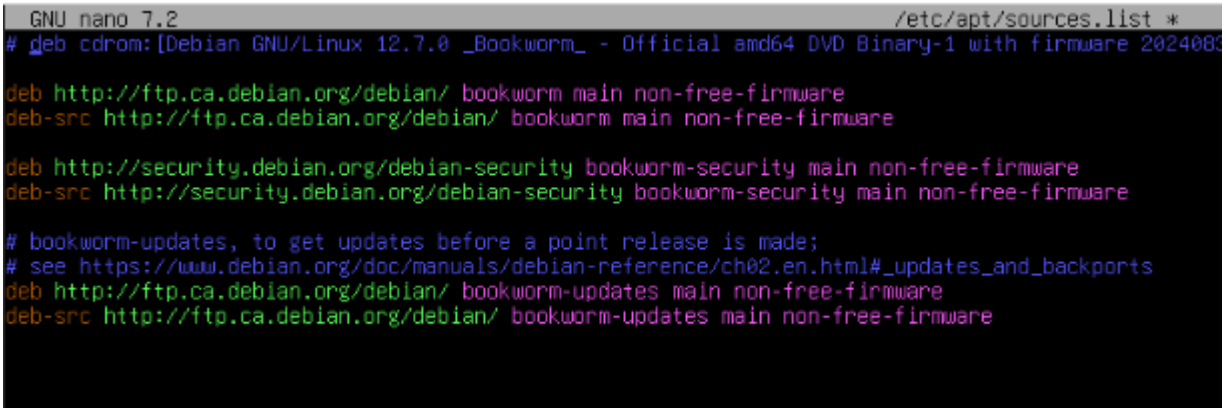


Figure 49: Edit Sources.list

Save the file by pressing <ctrl>x. Now update the repository settings by typing:

```
# apt-get update && apt-get upgrade
```

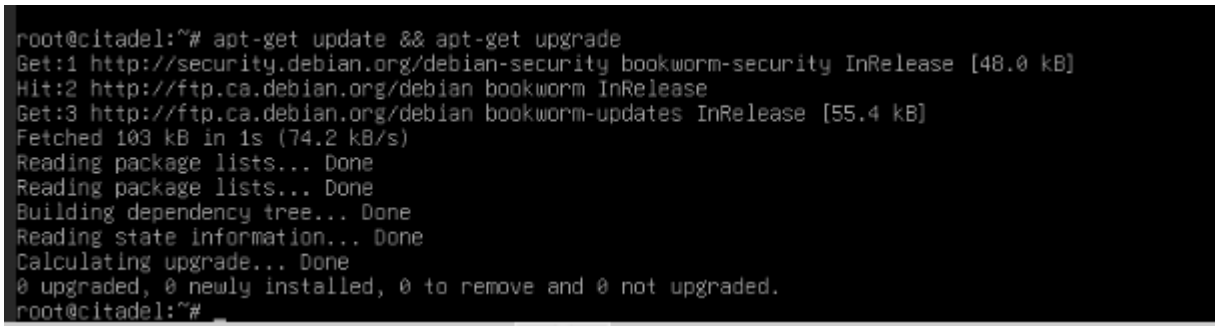


Figure 50: Update Upgrade

Set Startup Order

On the Proxmox administrative screen for the guest, set the “Start at boot” and the “Start/Shutdown order” setting in the options section.

```
Start at boot: Yes
Start/Shutdown order: 10
```

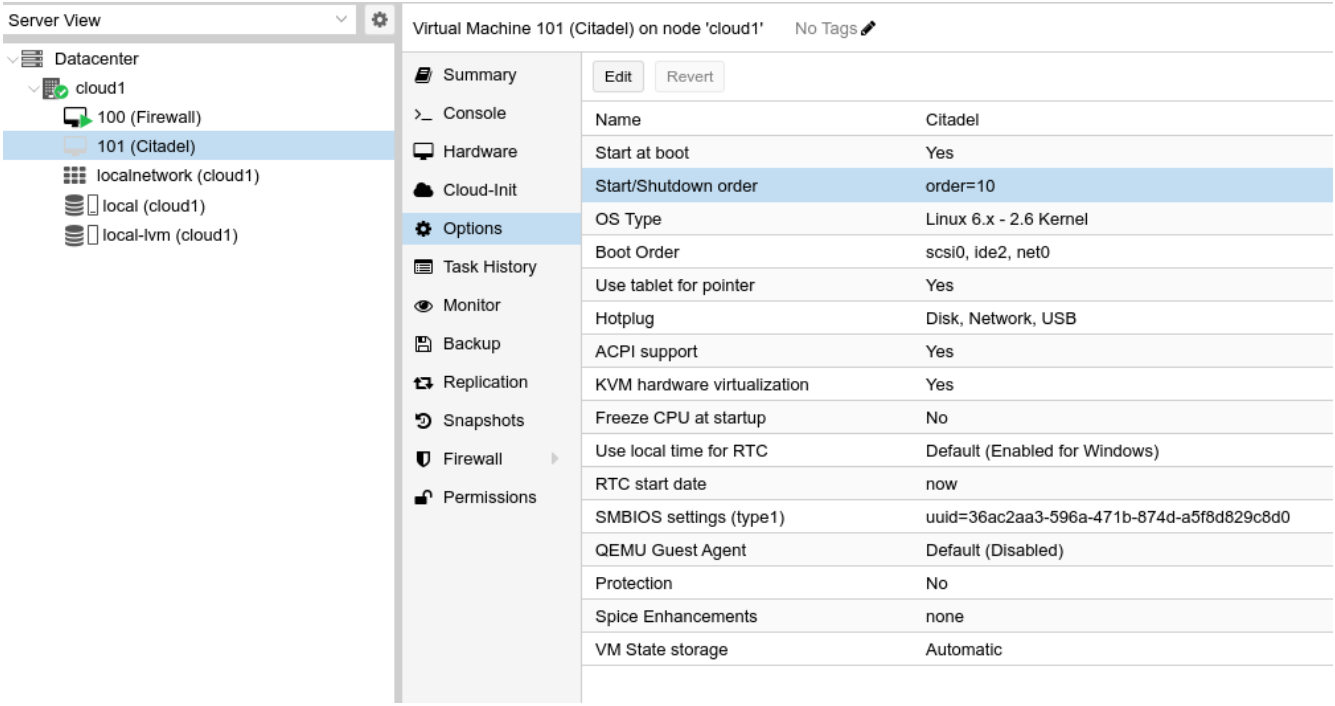


Figure 51: Guest Startup/Shutdown Order

2.7.2 Install Citadel Email Server

Having created a base Debian guest instance on the cloud, we can now log into it and start the installation. The easiest way to do this is via a SSH connection to the guest from your workstation.

SSH Linux and Windows:

Open a Terminal window (Linux) or Command Prompt (Windows). This should bring you to a command prompt. Use the SSH command to connect to the remote host. In this example, my username is “terry”, and the host name is “citadel.senseicto.com”.

```
$ ssh terry@citadel.senseicto.com
```

SSH (Secure SHell): The command asks for <username>@<hostname.domain>. The username is the username specified when you first installed the Debian guest. The hostname and domain are what you specified for this guest in the OPN Firewall.

Note that SSH is different from Telnet, in that it uses an encrypted connection as opposed to clear text.

This may give you a warning the first time, asking if you want to add the new host to the list of locally known hosts. Say yes.

```
The authenticity of host 'citadel.senseicto.com (192.168.100.10)' can't be established.
ECDSA key fingerprint is SHA256:AW+us1vlfbXElazXXxeCwhnOdtAA77cIBUNV5+PHw1Y.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'citadel.senseicto.com,192.168.100.10' (ECDSA) to the list of known hosts.
```

You will now be prompted for the password on the remote host.

```
terry@citadel.senseicto.com's password:
Linux citadel 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

You should now have a new command prompt that identifies your connection. You are now logged into the Debian guest that you created above.

Install Citadel

Change to root account

```
$ su - root
```

SU Command: The Super User command allows unprivileged accounts to temporarily run a privileged command. In this case, the “- root” instructs su to run a new shell as the root account. You will be asked for the root user’s password.

Now run the Citadel installer script

```
# wget -q -O - https://easyinstall.citadel.org/install | bash
```

Figure 52: Citadel Install

Review the license and accept if appropriate.

- Do you accept the terms of this license? … **Y**
- Do you want Easy Install to attempt to install your OS dependencies? **Y**
- Accept the OS updates
- Citadel administrator username - Leave as default (**admin**)
- Administrator password - change to preferred password
- Citadel User ID - Leave default (**root**)
- Listening address for the Citadel server - leave as default (*)
- Server port number - Leave default (**504**)
- Authentication method to use - Leave default (**0** Self Contained authentication)
- What HTTP port do you want to use for WebCit? **80**
- What HTTPS port do you want to use for WebCit? **443**
- All finished! You are ready to log in.

Administration Console

Open a browser window and navigate to https://<citadel hostname>.<domain name>. In this example, it would be https://citadel.senseicto.com.

Figure 53: Citadel Login

Login using the username and password you just entered during the Citadel installation.

Site wide Configuration

Navigate to Administration → Edit site-wide configuration
Change the Fully qualified domain name to your email domain. In this example, it is senseicto.com.

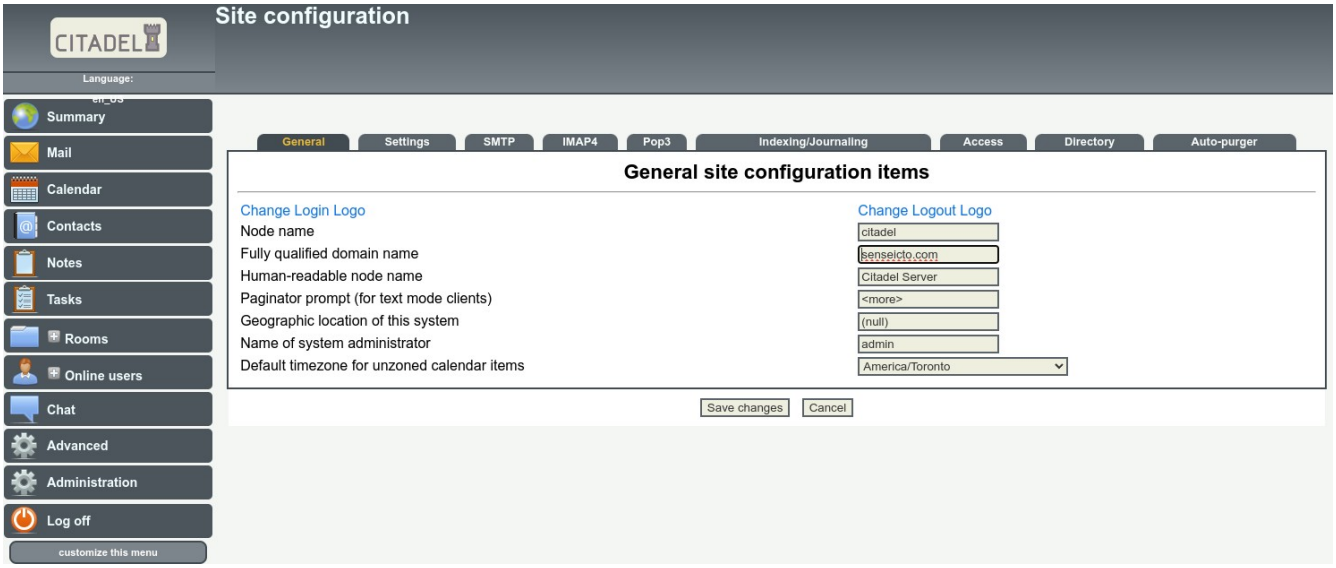


Figure 54: Citadel Domain Name

Open the SMTP tab and check the StartTLS box, then click the Save changes button.

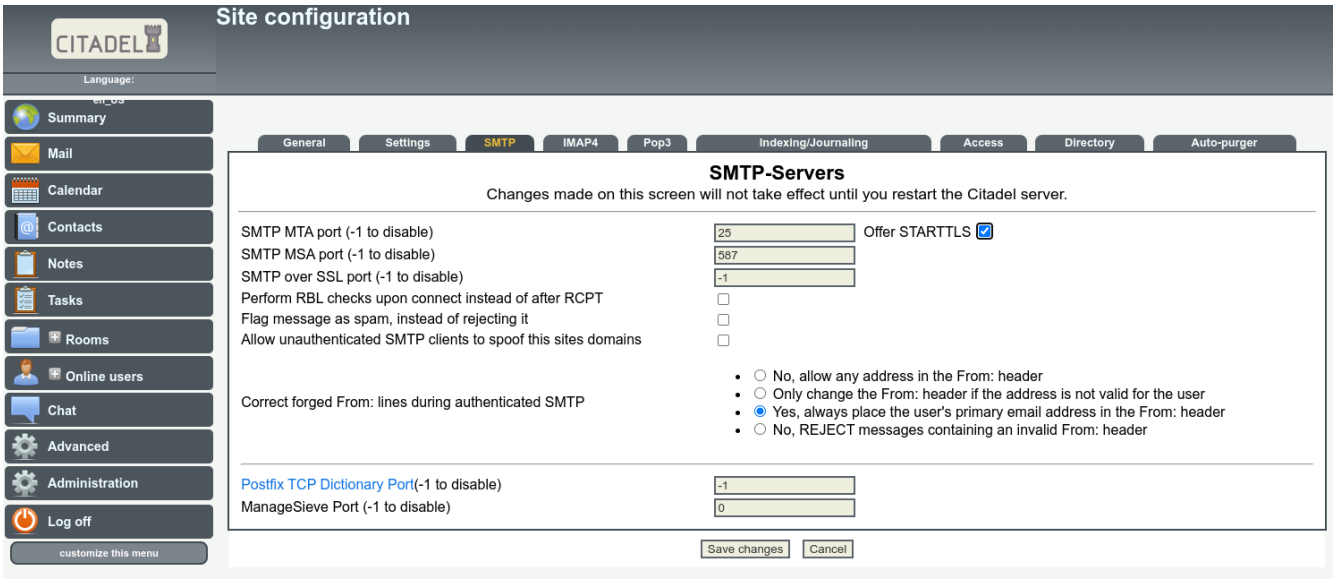


Figure 55: Citadel Configuration SMTP

Create Email Addresses

Create an email address for the Hubs Server to use for login authentication. Navigate to Administration→Add, change, delete user accounts

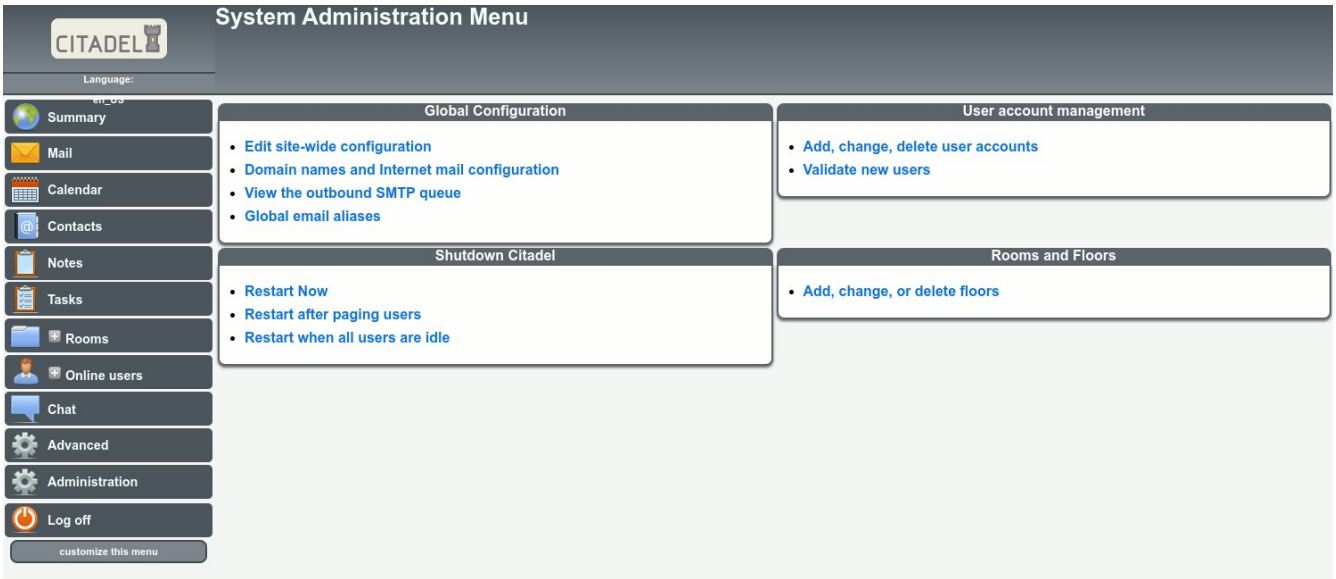


Figure 56: Citadel Admin Menu

Enter a new username - e.g. “hubsadmin”

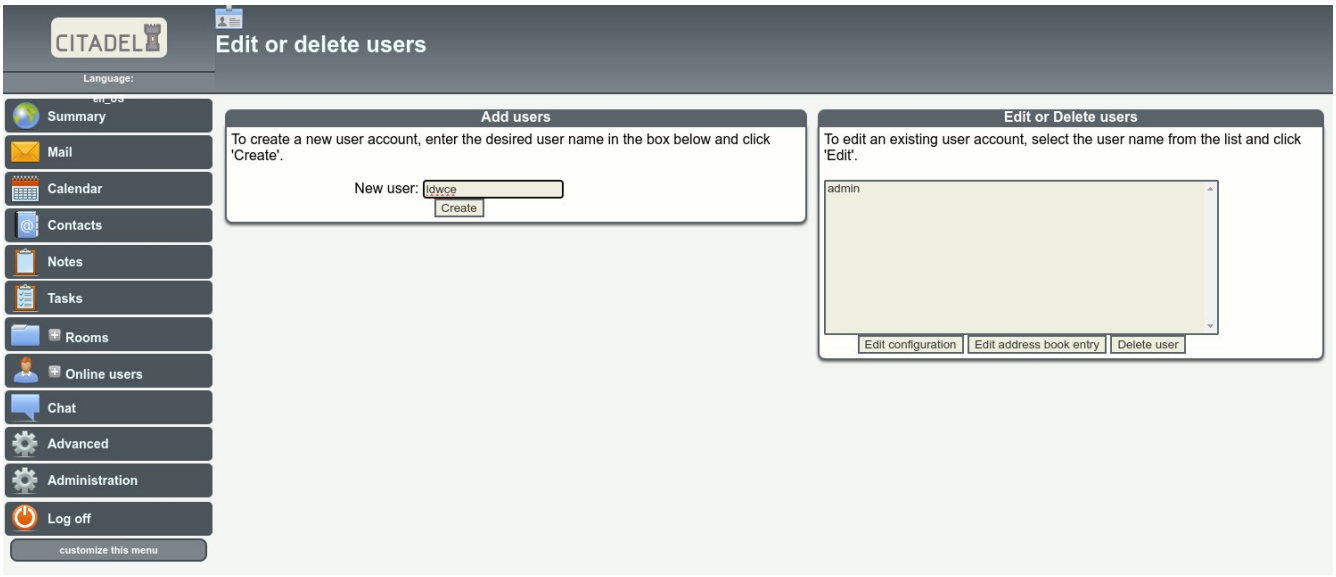


Figure 57: Citadel Add Hubs Admin User

Now edit the new user profile, specifying the actual email address

Figure 58: Citadel User Profile

Now add another user account for your use as a user of Hubs CE.

Email Test

Now try and send an email from the Hubs account to the personal account on the same server.

Logout of the Admin account, and log in as the Hubs account (e.g. hubsadmin).

Navigate to Mail→Write mail, and create a new message to the user account created above (e.g. terry). Click Send message,

Now log out, and log back in as the user (e.g. terry). Navigate to Mail and confirm receipt of the new mail message.

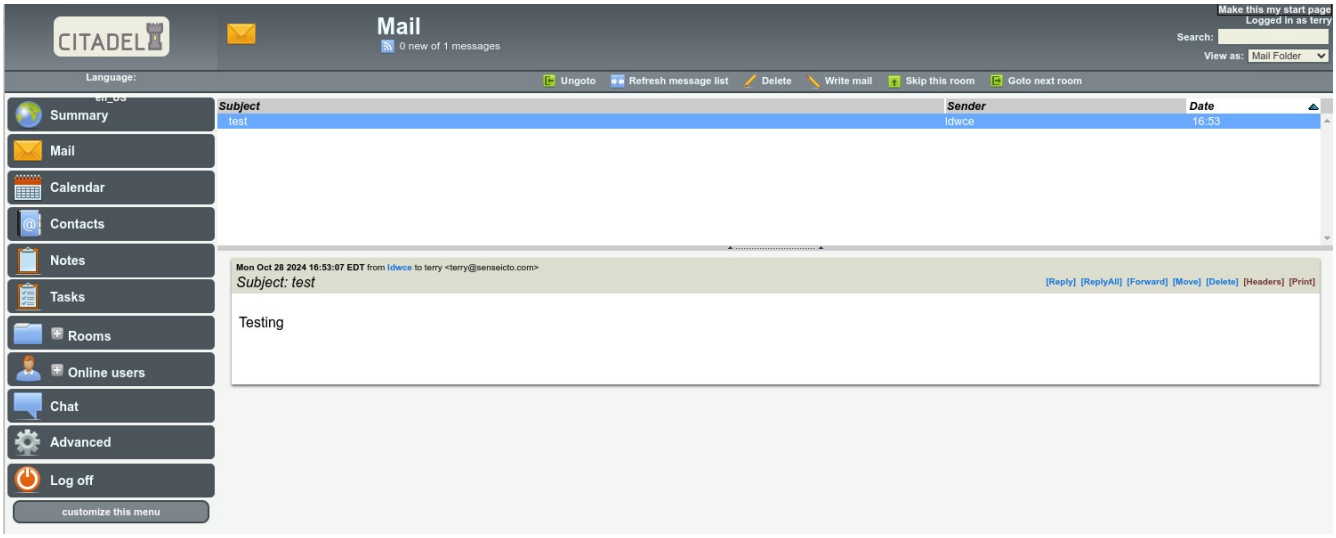


Figure 59: Citadel New Email

We now have a working email server to allow for local authentication without having to register an actual domain.

Change Certificates (OPTIONAL)

If you are using a registered domain, you can generate signed SSL certificates for the server. For example, you can manually generate Let's Encrypt certificates for free. You can then upload them to the Citadel server, and install them.

Generate Certificates
I will leave it as an exercise for the student to generate either self-signed certificates, or authenticated certificates such as Let's Encrypt using the certbot command on the Citadel server.

Install Certificates
Assuming you have created or copied certificates for the server to the local disk, you can create a symbolic link to them in order to make updating them easier. Let's Encrypt certificates, for example, only last 3 months.

Assuming you generated Let's Encrypt certificates and stored them in /etc/letsencrypt directory, then create the links using the root account (remember? $ su - root, which will change your prompt to "#")

Assuming you copied your private key and certificates files to the directory “/etc/letsencrypt”, then use the following symbolic link commands to identify them to Citadel;

```
# ln -sfv /etc/letsencrypt/privkey1.pem /usr/local/citadel/keys/citadel.key
# ln -sfv /etc/letsencrypt/fullchain1.pem /usr/local/citadel/keys/citadel.cer
```

And that’s it! We now have everything we need to host our Hubs Cloud Community Edition locally.

3 Hubs Cloud Community Edition Installation

References

Hubs Community Edition Source -
https://github.com/Hubs-Foundation/hubs-cloud/tree/master/community-edition

Overview

Hubs Cloud Community Edition automates most of the complex deployment process using Kubernetes. It orchestrates the installation from an installer node, creating the kubernetes-based deployment across all of the participating nodes reviewed below.

What is Kubernetes (K8s)? Developed by Google and released as open source in 2014, k8s automates the deployment, scaling, and management of containerized applications. It orchestrates containers that make up an application into logical units for easy management and discovery.

Kubernetes automates operational tasks of container management and includes built-in commands for deploying applications, rolling out changes, scaling up and down to fit changing needs, monitoring, and more.

In order to install Hubs Cloud Community Edition on our server, we need to create 3 Debian guests:

- **Installer Workstation**: This is a Debian workstation instance (has a desktop GUI) that will run the Kubespray installer.

- **Hubs Master Node**: This is the Kubernetes control plane server.

Master nodes host the K8s control plane components. The master node holds configuration and state data used to maintain the desired state. The control plane maintains communication with the worker nodes in order to schedule containers efficiently.

- **Hubs Worker node**: This is where the Hubs Cloud Community Edition services run.

Worker nodes are so-called because they run pods. Pods are usually single instances of an application. Containers run inside pods, and pods run on a node. Each cluster always contains at least one worker node.

Following the Debian guest creation notes for the Citadel server, create three new Debian guests using the following configuration details.

3.1 Create the Guests

3.1.1 Installer Workstation

In Proxmox Cloud

Name: InstallWorkstation
ISO Image: debian-xx.x.x-amd64-DVD-1.iso (from the drop-down list of ISOs)
Memory: 4096 MiB
Processors: 4 Sockets, 4 Cores (total of 16 VCPUs)
Hard Disk: 50G

In OPNSense Firewall

In the firewall, create the DHCP static entry as follows:
MAC Address: <as created by ProxMox>
IP Address: 192.168.100.21
Hostname: hubsinstaller
Description: Hubs Cloud Installation Workstation
Domain name: <your domain>
where <your domain> is whatever you decided earlier
for this example I am using *senseicto.com*

Leave everything else as default.

In Debian Installer

During the Debian installation process, specify that this is using a GUI desktop (Gnome) and enable SSH Server (for remote logins).

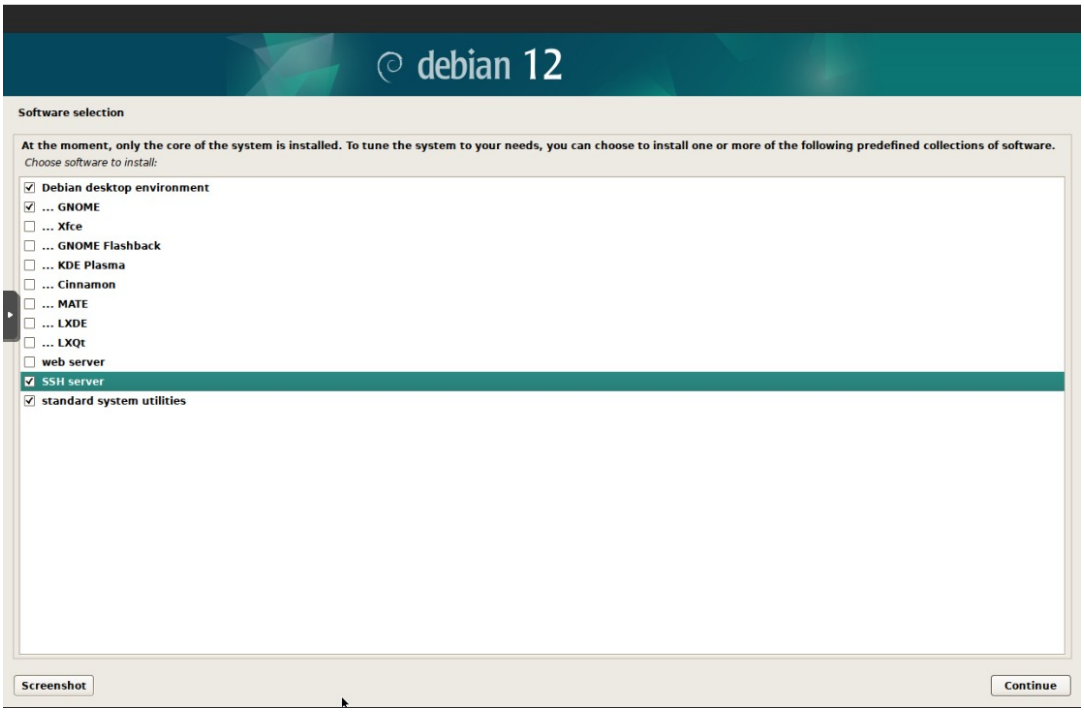


Figure 60: Workstation Software Selection

3.1.2 Hubs Master

In Proxmox Cloud

Name: HubsMaster
ISO Image: debian-xx.x.x-amd64-DVD-1.iso (from the drop-down list of ISOs)
Memory: 32768 MiB
Processors: 4 Sockets, 6 Cores (total of 24 VCPUs)
Hard Disk: 200G

In the Options tab, set **Start at boot** to Yes, and **Start/Shutdown order** to 20

In OPNSense Firewall

In the firewall, create the DHCP static entry as follows:
MAC Address: <as created by ProxMox>
IP Address: 192.168.100.30
Hostname: hubsmaster
Description: Hubs Cloud Master
Domain name: <your domain>
where <your domain> is whatever you decided earlier - for this example I am using *senseicto.com*

Leave everything else as default.

In Debian Installer

Make sure to *enable* SSH Server during software selection, and *disable* any desktop environment (i.e. we are making a headless server).

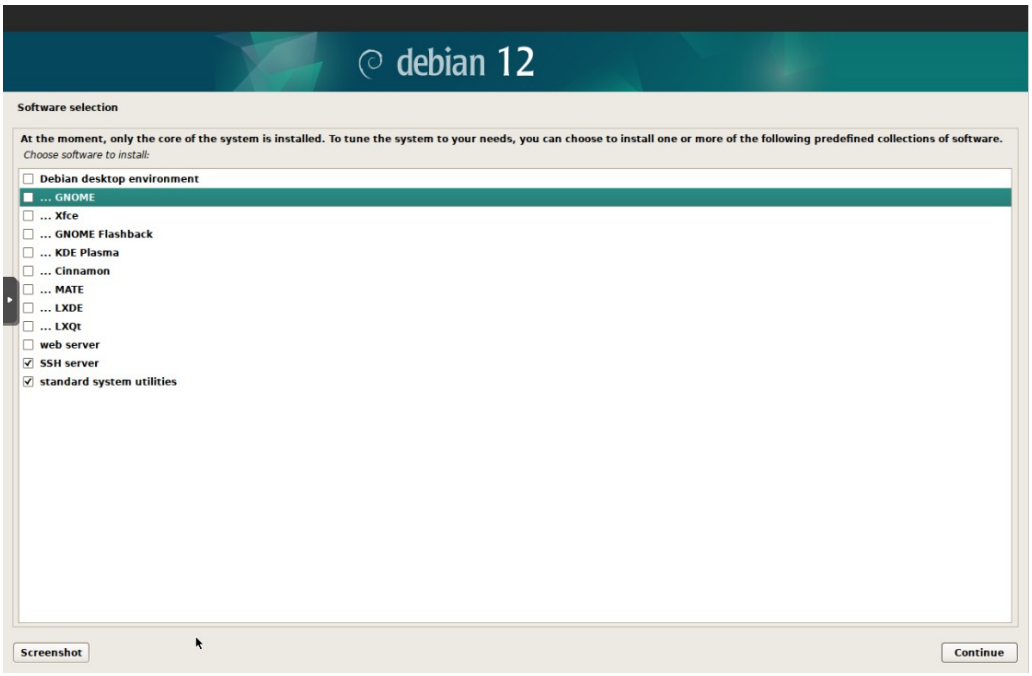


Figure 61: Hubs Nodes Software Selection

What is a "Headless Server"? A headless server usually refers to a server that does not provide a GUI user interface - IE no graphics card required. It is meant to run applications and services that provide their own GUI interfaces. By not having to manage a desktop, a lot of system resources are saved, including Memory, CPU, I/O, Disk and more.

3.1.3 Hubs Worker

In Proxmox Cloud

Name: HubsWorker
ISO Image: debian-xx.x.x-amd64-DVD-1.iso (from the drop-down list of ISOs)
Memory: 32768 MiB
Processors: 4 Sockets, 6 Cores (total of 24 VCPUs)
Hard Disk: 200G

In the Options tab, set **Start at boot** to Yes, and **Start/Shutdown order** to 21

In OPNSense Firewall

In the firewall, create the DHCP static entry as follows:
MAC Address: <as created by ProxMox>
IP Address: 192.168.100.31
Hostname: hubsworker
Description: Hubs Cloud Worker
Domain name: <your domain>
where <your domain> is whatever you decided earlier
for this example I am using *senseicto.com*

Leave everything else as default.

In Debian Installer

Make sure to *enable* SSH Server during software selection, and *disable* any desktop environment (i.e. headless server).

3.2 Install Kubernetes

On All Nodes (Install, Master, Worker)

make sure to do this step on all three of the guests.

Login as root. You can either use the shell for the guest via the Proxmox console, or log in from your workstation using SSH (as we did above). Then change roles to the 'root' user using the su command.

```
$ su - root
#
```

Need to add two lines to /etc/ssh/sshd_config
On all three nodes, run the following as root user. We will use the nano command to edit the sshd_config file and add the two lines to it. If you need help on using nano editor, simply Google it - there is lots of good info on using nano. (I avoid the vi editor as it is a more advanced interface that can be confusing for someone new to console-based editors.)

```
# nano /etc/ssh/sshd_config

PubkeyAcceptedKeyTypes +ssh-rsa
HostKeyAlgorithms +ssh-rsa
```

Pre-Requisites

Install sudo on all nodes. Using the apt-get command, we can install applications and packages that we need over the internet.

```
# apt-get update && apt-get upgrade && apt-get install -y sudo git python3-pip
```

What is Apt-get? Linux variants like Ubuntu and Debian provide an online packaging system that allows users to download and install programs, utilities and more. It is also used for maintaining and updating the actual operating system. In this case, we are installing the git access manager, as well as the python language support.

Ensure user added to /etc/sudoers file

```
# nano /etc/sudoers


...
# User privilege specification
root    ALL=(ALL:ALL) ALL
terry    ALL=(ALL:ALL) ALL
...
```

Replace “terry” with your local user name.

Why do we need SUDO (Super User Do)? Rather than being logged in as root, the sudo command allows unprivileged accounts to temporarily elevate their privileges, typically for one command. We will be using this throughout the installation and configuration of the servers.

Exit Root

You can now exit root and perform everything else as the default user

```
# exit
$
```

Disable Swap on all nodes

Kubernetes performs best when swap is disabled. Disable it with the following commands:

```
$ sudo swapoff -a
$ sudo sed -i '/ swap / s/^/#/' /etc/fstab
```

Generate and Distribute SSH Keypairs

On the *Hubs Installer workstation*, generate key pairs

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/terry/.ssh/id_rsa):
Created directory '/home/terry/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/terry/.ssh/id_rsa
Your public key has been saved in /home/terry/.ssh/id_rsa.pub
```

Now copy to the master and worker servers
ssh <username>@<hostname.domain>, e.g.

```
$ ssh-copy-id terry@hubsmaster.senseicto.com
$ ssh-copy-id terry@hubsworker.senseicto.com
```

and test the logins
ssh '<username>@<hostname.domain>, e.g.

```
$ ssh 'terry@hubsmaster.senseicto.com'
$ exit

$ ssh 'terry@hubsworker.senseicto.com'
$ exit
```

You should be logged directly into the hubs master and worker nodes after executing the ssh commands above (noted by the command prompt). Be sure to 'exit' back to the installer node after verifying that you can log in without a password to these nodes.

Why are we configuring SSH to not use passwords? The Kubernetes installation that we will be performing requires the installer node to be able to execute commands on the target master and worker nodes. As such, it uses passwordless SSH to do so.

On Hubsinstaller, get kubespray

```
$ cd ~/Documents
$ git clone https://github.com/kubernetes-sigs/kubespray.git
$ cd kubespray
```

NOTE: If you cannot find directory 'Documents', then you have not logged into the installer node graphically yet. Go to the console tab for the Hubsinstaller

workstation guest in the Proxmox administration screen and log in. This will create the user's directories and complete the installation of Debian on the guest.

Create an inventory document

```
$ cp -rfp inventory/sample inventory/hubscluster
```

Now edit the document to match the above servers

```
$ nano inventory/hubscluster/inventory.ini
```

```
[all]
hubsmaster ansible_host=192.168.100.30 etcd_member_name=etcd1
ansible_user=terry
hubsworker   ansible_host=192.168.100.31 etcd_member_name=
ansible_user=terry

[kube_control_plane]
hubsmaster

[etcd]
hubsmaster

[kube_node]
hubsworker


[calico_rr]

[k8s_cluster:children]
kube_control_plane
kube_node
calico_rr
```

```
$ sudo pip3 install -r requirements.txt --break-system-packages
```

NOTE: This will take a while. Kick back and relax, monitoring the progress as it goes in case any errors show up.

Test it:

```
$ ansible --version
```

```
ansible [core 2.16.13]
  config file = /home/terry/Documents/kubespray/ansible.cfg
  configured module search path = ['/home/terry/Documents/kubespray/library']
  ansible python module location =
/usr/local/lib/python3.11/dist-packages/ansible
  ansible collection location =
/home/terry/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/local/bin/ansible
  python version = 3.11.2 (main, Aug 26 2024, 07:20:54) [GCC 12.2.0]
(/usr/bin/python3)
  jinja version = 3.1.4
  libyaml = True
```

Install Kubernetes Cluster

Run the playbook to install across the nodes.

```
$ ansible-playbook -i inventory/hubscluster/inventory.ini --become --
user=<username> --become-user=root cluster.yml --extra-vars
"ansible_sudo_pass=<password>"
```

replacing <username> with your username (e.g. terry) and <password> with the account password.

Again, sit back and relax, monitoring the progress.

Test - From *hubsmaster* node

```
$ sudo kubectl cluster-info
[sudo] password for terry:
Kubernetes control plane is running at https://127.0.0.1:6443
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

```
$ sudo kubectl get nodes
NAME         STATUS   ROLES           AGE     VERSION
hubsmaster   Ready    control-plane   8m12s   v1.31.1
hubsworker   Ready    <none>          7m19s   v1.31.1
```

```
$ sudo kubectl get endpoints -n kube-system
NAME      ENDPOINTS                                                     AGE
coredns   10.233.87.193:53,10.233.91.193:53,10.233.87.193:53 + 3 more...   56m
```

3.3 Install Hubs CE on the Cluster

Reference https://github.com/Hubs-Foundation/hubs-cloud/tree/master/community-edition for new installation steps.

Prerequisites

On the Hubs Master (control plane) server

```
$ sudo apt-get update && sudo apt-get upgrade -y
$ sudo apt-get install -y nodejs npm wget curl
$ sudo npm install pem-jwk -g
```

Step I: Configure DNS

Create 4 placeholder records
Should be using your default domain as used above (e.g. senseicto.com)

We need to create the following records in the firewall DNS via the OPNSense console.

```
Host Name: {keep blank}
Domain: hubs.<your domain>
Record Type: A (IPv4 address)
IP Address: 0.0.0.0
Description: Main HUBS Server

Host Name: assets
Domain: hubs.<your domain>
Record Type: A
IP Address: 0.0.0.0
Description: HUBS Assets Hosting

Host Name: stream
Domain: hubs.<your domain>
Record Type: A
IP Address: 0.0.0.0
Description: HUBS Streaming Server

Host Name: cors
Domain: hubs.<your domain>
Record Type: A
IP Address: 0.0.0.0
Description: HUBS CORS Service
```

where <your domain> is replaced with the default domain you specified earlier. In this example, it is senseicto.com.

Open a browser session to the OPNSense server and navigate to Services→Unbound DNS→Overrides and in the Host Overrides tab add a new override by clicking the first “+” button. This should open a new Overrides dialog. Then simply add the above 4 new records, one at a time.
For now, the 0.0.0.0 addresses are placeholders - they will be replaced shortly with real IP addresses.

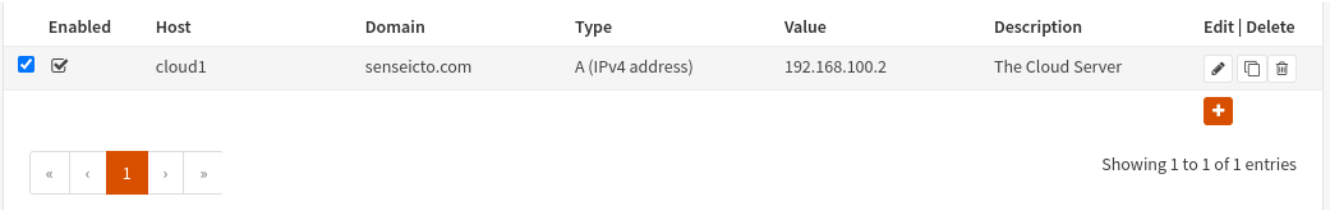


Figure 62: Host Overrides

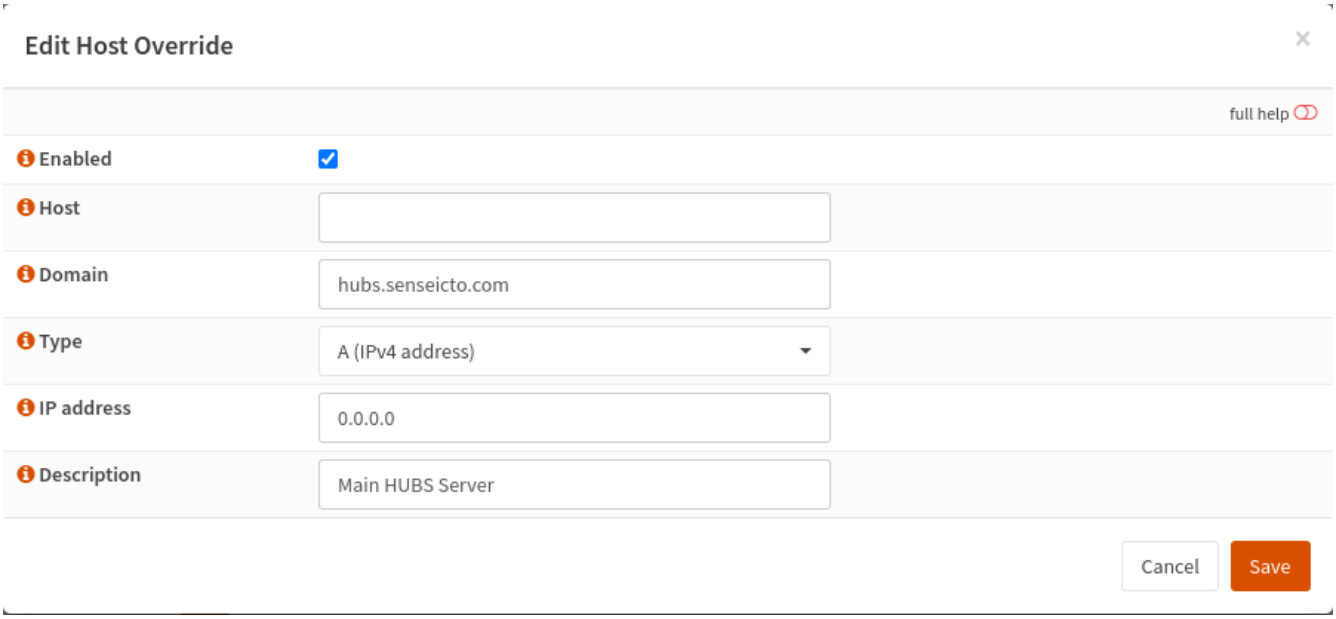


Figure 63: New Host Override

When complete, the host overrides should look like this;

Enabled	Host	Domain	Type	Value	Description	Edit | Delete
☑ ☑	cloud1	senseicto.com	A (IPv4 address)	192.168.100.2	The Cloud Server	
☐ ☑		hubs.senseicto.com	A (IPv4 address)	0.0.0.0	Main HUBS Server	
☐ ☑	assets	hubs.senseicto.com	A (IPv4 address)	0.0.0.0	HUBS Assets Hosting	
☐ ☑	stream	hubs.senseicto.com	A (IPv4 address)	0.0.0.0	HUBS Streaming Server	
☐ ☑	cors	hubs.senseicto.com	A (IPv4 address)	0.0.0.0	HUBS CORS Service	

Figure 64: New Host Overrides

Once they are all added, be sure to press the “Apply” button at the bottom of the screen.

Why is the Domain now “hubs.senseicto.com”? We are creating a new *subdomain* for the server. The base server will now be accessed via the name “hubs.senseicto.com”, and it will reference other services in the Hubs server via this subdomain, such as stream.hubs.senseicto.com. This helps contain all of the services together logically in the domain namespace, and is great if you are hosting Hubs using an existing corporate domain. Using a subdomain will help avoid naming conflicts with other servers using the base domain.

3.4 Download, Configure, and Deploy Community Edition

On the hubsmaster node (the Control plane), create a Documents directory;

```
$ mkdir ~/Documents
$ cd ~/Documents
```

Clone the GitHub repository using the command line in VSCode.

```
$ git clone https://github.com/Hubs-Foundation/hubs-cloud.git
```

Move into the community-edition directory.

```
$ cd hubs-cloud/community-edition
```

Install the module dependencies:

```
$ npm ci
```

Now we can edit the configuration values for our Hubs Cloud CE Server ny editing the configuration file.

```
$ nano input-values.yaml
```

HUB_DOMAIN	“hubs.<your domain>”
ADM_EMAIL	"hubsadmin@<your domain>"
Namespace	“localhubs”
Container_Dockerhub_Username	“mozillareality”
Container_Tag	“stable-latest”
DB_USER	“postgres”
DB_PASS	“hXsuwOoH3HSXhjfK”
DB_NAME	“retdb”
DB_HOST	“pgbouncer”
DB_HOST_T	“pgbouncer-t”
PGRST_DB_URI	“postgres://$DB_USER:$DB_PASS@pgbouncer/$DB_NAME”
PSQL	“postgres://$DB_USER:$DB_PASS@pgbouncer/$DB_NAME”
SMTP_SERVER	“citadel.<your domain>”
SMTP_PORT	“25”
SMTP_USER	“hubsadmin”
SMTP_PASS	“<hubsdomain email password>”
NODE_COOKIE	“dUDMBUkWeitqfvYa”
GUARDIAN_KEY	“xGtecfHhfaZTxKcW”
PHX_KEY	“ZAiohVUjBsrjuFMD”
SKETCHFAB_API_KEY	<your Sketchfab key>
TENOR_API_KEY	<your Tenor API Key>
PERSISTENT_VOLUME_SIZE	100Gi
OVERRIDE_RETICULUM_IMAGE:	"hubsfoundation/ret:$Container_Tag"
OVERRIDE_POSTGREST_IMAGE:	"No"
OVERRIDE_POSTGRES_IMAGE:	"No"
OVERRIDE_PGBOUNCER_IMAGE:	"No"
OVERRIDE_HUBS_IMAGE:	"hubsfoundation/hubs:$Container_Tag"
OVERRIDE_SPOKE_IMAGE:	"hubsfoundation/spoke:$Container_Tag"
OVERRIDE_NEARSPARK_IMAGE:	"No"
OVERRIDE_PHOTOMNEMONIC_IMAGE:	"No"
OVERRIDE_DIALOG_IMAGE:	"No"
OVERRIDE_COTURN_IMAGE:	"No"
OVERRIDE_HAPROXY_IMAGE:	"No"

In the original file, you should see a number of “changeMe” values, for example;

- **<your domain> should be changed.** In the example, this would be senseicto.com.
  E.g. HUB_DOMAIN: "hubs.senseicto.com"
- **Change your namespace**.
  E.g. Namespace: "localhubs"
- **Change the database password from the default**.
  E.g. DB_PASS: "hXsuwOoH3HSXhjfK"
- **Change the SMTP Server settings** to your server. e.g.
  - SMTP_SERVER: "citadel.senseicto.com"
  - SMTP_PORT: "25"
  - SMTP_USER: "hubsadmin@senseicto.com"
  - SMTP_PASS: "<whatever your Citadel hubsadmin email password is>"
- **Change the keys** to something 16 characters long, lower and upper case between a-Z
  - NODE_COOKIE: "dUDMBUkWeitqfvYa"
  - GUARDIAN_KEY: "xGtecfHhfaZTxKcW"
  - PHX_KEY: "ZAiohVUjBsrjuFMD"
- **If you have a Sketchfab key**, insert it here, otherwise leave the “?”
  SKETCHFAB_API_KEY: "?"
- **If you have a Tenor API key**, insert it here, otherwise leave it the “?”
  TENOR_API_KEY: "?"
- **Make the persistent volume 100 Gigabytes in size**
  PERSISTENT_VOLUME_SIZE: "100Gi"

Leave everything else default valued for now.

Run npm run gen-hcce followed by sudo kubectl apply -f hcce.yaml

```
$ npm run gen-hcce
> script@1.0.0 gen-hcce
> node generate_script/index.js

hcce.yaml file generated successfully.
```

Now apply the configuration

```
$ sudo kubectl apply -f hcce.yaml
```

Expose the services

Run kubectl -n <hcce_namespace> get svc lb to find your load balancer's external ip, eg;

```
$ sudo kubectl -n localhubs get svc lb
```

Note that the EXTERNAL-IP address is listed as <pending>. This is one of those hold-over items from running on a cloud like AWS that we need to now address ourselves. We need a load balancer.

As a result, we need to install more infrastructure at this point, namely a load balancer and an Ingress Controller.

3.5 Install MetalLB Load Balancer on Kubernetes Cluster

Why Do We Need a Load Balancer? When Hubs was deployed on a Kubernetes Cluster on AWS, the platform automatically provided a network load balancer. As a result, we need to provide our own. In this case, we will use MetalLB.

3.5.1 References

https://computingforgeeks.com/deploy-metallb-load-balancer-on-kubernetes/

3.5.2 Pre-Requisites

From the HubsMaster (control plane) server

```
$ sudo apt-get update
$ sudo apt-get install wget curl -y
```

Get the latest MetalLB release tag:

```
$ MetalLB_RTAG=$(curl -s
https://api.github.com/repos/metallb/metallb/releases/latest|grep tag_name|cut -d
'"' -f 4|sed 's/v//')
```

To check release tag use echo command:

```
$ echo $MetalLB_RTAG
0.14.8
```

Create directory where manifests will be downloaded to.

```
$ mkdir ~/Documents/metallb
$ cd ~/Documents/metallb
```

Download MetalLB installation manifest:

```
$ wget https://raw.githubusercontent.com/metallb/metallb/v$MetalLB_RTAG/config/
manifests/metallb-native.yaml
```

Below are the components in the manifest file:

- The metallb-system/controller deployment – Cluster-wide controller that handles IP address assignments.
- The metallb-system/speaker daemonset – Component that speaks the protocol(s) of your choice to make the services reachable.
- Service accounts for both controller and speaker, along with *RBAC permissions* that the needed by the components to function.

Install MetalLB Load Balancer on Kubernetes cluster

```
$ sudo kubectl apply -f metallb-native.yaml
namespace/metallb-system created
customresourcedefinition.apiextensions.k8s.io/addresspools.metallb.io created
customresourcedefinition.apiextensions.k8s.io/bfdprofiles.metallb.io created
customresourcedefinition.apiextensions.k8s.io/bgpadvertisements.metallb.io
created
customresourcedefinition.apiextensions.k8s.io/bgppeers.metallb.io created
customresourcedefinition.apiextensions.k8s.io/communities.metallb.io created
customresourcedefinition.apiextensions.k8s.io/ipaddresspools.metallb.io created
customresourcedefinition.apiextensions.k8s.io/l2advertisements.metallb.io created
serviceaccount/controller created
serviceaccount/speaker created
role.rbac.authorization.k8s.io/controller created
role.rbac.authorization.k8s.io/pod-lister created
clusterrole.rbac.authorization.k8s.io/metallb-system:controller created
clusterrole.rbac.authorization.k8s.io/metallb-system:speaker created
rolebinding.rbac.authorization.k8s.io/controller created
rolebinding.rbac.authorization.k8s.io/pod-lister created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:controller created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:speaker created
configmap/metallb-excludel2 created
secret/webhook-server-cert created
service/webhook-service created
deployment.apps/controller created
daemonset.apps/speaker created
validatingwebhookconfiguration.admissionregistration.k8s.io/metallb-webhook-
configuration created
```

...and wait...

```
$ sudo watch kubectl get all -n metallb-system
```

...and wait...

```
$ sudo kubectl get pods -n metallb-system --watch
```

The screen will automatically update. Wait until all services have a Ready state of “1/1”.

```
NAME                          READY   STATUS    RESTARTS   AGE
controller-8694df9d9b-n4qdk   1/1     Running   0          72s
speaker-4st74                 1/1     Running   0          72s
speaker-vms48                 1/1     Running   0          72s
```

Create Address Pool

MetalLB needs a pool of IP addresses to assign to the services when it gets such requests. We have to instruct MetalLB to do so via the IPAddressPool CR.
In the firewall, likeliest place to grab this is the DHCP pool…
For us, for example, this is 192.168.100.100 - 192.168.100.120

```
$ nano ~/Documents/metallb/ipaddress_pools.yaml
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: production
  namespace: metallb-system
spec:
  addresses:
  - 192.168.100.100-192.168.100.120
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: l2-advert
  namespace: metallb-system
```

Apply the configuration

```
$ sudo kubectl apply -f  ~/Documents/metallb/ipaddress_pools.yaml
ipaddresspool.metallb.io/production created
l2advertisement.metallb.io/l2-advert created
```

List created IP Address Pools and Advertisements:

```
$ sudo kubectl get ipaddresspools.metallb.io -n metallb-system
NAME         AUTO ASSIGN   AVOID BUGGY IPS   ADDRESSES
production   true          false             ["192.168.100.100-192.168.100.120"]
```

```
$ sudo kubectl get l2advertisements.metallb.io -n metallb-system
NAME        IPADDRESSPOOLS   IPADDRESSPOOL SELECTORS   INTERFACES
l2-advert
```

Get more details using describe kubectl command option:

```
$ sudo kubectl describe ipaddresspools.metallb.io production -n metallb-system
Name:         production
```

```
Namespace:    metallb-system
Labels:       <none>
Annotations:  <none>
API Version:  metallb.io/v1beta1
Kind:         IPAddressPool
Metadata:
  Creation Timestamp:  2024-08-24T21:44:15Z
  Generation:          1
  Resource Version:    13388
  UID:                 87810025-1b7a-4633-95be-c31b2335309e
Spec:
  Addresses:
    192.168.100.100-192.168.100.120
  Auto Assign:       true
  Avoid Buggy I Ps:  false
Events:              <none>
```

IMPORTANT NOTE:

By default, the IP addresses are assigned from the pool sequentially, setting up a race condition (?!). To ensure that the services get specific IP addresses for DNS configuration, add an IP address specifier to the manifest file:

```
$ nano metallb-native.yaml
...
spec:
  loadBalancerIP: 192.168.100.100
...
```

3.6 Install Nginx Ingress Controller on Kubernetes

3.6.1 References

https://computingforgeeks.com/deploy-nginx-ingress-controller-on-kubernetes-using-helm-chart/

Openlens Prometheus support - https://docs.nginx.com/nginx-ingress-controller/logging-and-monitoring/prometheus/

Why Do We Need and Ingress Controller? An Ingress in Kubernetes exposes HTTP and HTTPS routes from outside the cluster to services running within the cluster. All the traffic routing is controlled by rules defined on the Ingress resource.

An Ingress controller is what fulfills the Ingress, usually with a load balancer.

There are lots of installation options. The one we want is the Bare-metal method which applies to Kubernetes clusters deployed on Debian locally.

3.6.2 Prerequisites

```
$ mkdir ~/Documents/ingress
$ cd ~/Documents/ingress
```

```
$ controller_tag=$(curl -s https://api.github.com/repos/kubernetes/ingress-nginx/releases/latest | grep tag_name | cut -d '"' -f 4)
```

```
$ echo $controller_tag
helm-chart-4.11.3
```

```
$ wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/${controller_tag}/deploy/static/provider/baremetal/deploy.yaml
```

```
$ mv deploy.yaml nginx-ingress-controller-deploy.yaml
```

3.6.3 Install

Apply Nginx ingress controller manifest deployment file:

```
$ sudo kubectl apply -f nginx-ingress-controller-deploy.yaml
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
serviceaccount/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx created
```

```
role.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
configmap/ingress-nginx-controller created
service/ingress-nginx-controller created
service/ingress-nginx-controller-admission created
deployment.apps/ingress-nginx-controller created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
ingressclass.networking.k8s.io/nginx created
networkpolicy.networking.k8s.io/ingress-nginx-admission created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-
admission created
```

Run the following command to check if the ingress controller pods have started:

```
$ sudo kubectl get pods -n ingress-nginx -l app.kubernetes.io/name=ingress-nginx
--watch
NAME                                        READY   STATUS      RESTARTS   AGE
ingress-nginx-admission-create-qwjmb        0/1     Completed   0          3m19s
ingress-nginx-admission-patch-b99sk         0/1     Completed   0          3m19s
ingress-nginx-controller-855dc99c4-kl5q6    1/1     Running     0          3m19s
```

Once the ingress controller pods are running with a Ready status of 1/1, you can cancel the command typing Ctrl+C.

3.7 Finishing the Hubs Cloud CE Installation

Run the svc_lb listing again, taking note of the External-IP address

```
$ sudo kubectl -n localhubs get svc lb
NAME   TYPE           CLUSTER-IP      EXTERNAL-IP
lb     LoadBalancer   10.233.33.196   192.168.100.100
```

Back in the firewall console, edit the Firewall A-Records overrides, changing 0.0.0.0 to the External-IP listed from the command above. For example;

	Enabled	Host	Domain	Type	Value	Description
☑	☑	cloud1	senseicto.com	A (IPv4 address)	192.168.100.2	The Cloud Server
☐	☑		hubs.senseicto.com	A (IPv4 address)	192.168.100.100	Main HUBS Server
☐	☑	assets	hubs.senseicto.com	A (IPv4 address)	192.168.100.100	HUBS Assets Hosting
☐	☑	stream	hubs.senseicto.com	A (IPv4 address)	192.168.100.100	HUBS Streaming Server
☐	☑	cors	hubs.senseicto.com	A (IPv4 address)	192.168.100.100	HUBS CORS Service

Figure 65: Final Host Overrides

4 Certificates

As promised, the server is now running, BUT we still have a certificate issue. If you try and navigate to your base server, you will get a security warning. If you accept the self-signed certificates, you only get a blank screen. That's because your browser is also not accepting the self-signed certificates of the other Hubs servers that you have running for CORS, ASSETS, and STREAM.

In order to run this with the self-signed certificates, ***every browser that tries to connect*** will need to navigate to EACH server, accepting the self-signed certificates. Then try navigating back to the base server and you should see a login screen. This means navigating (in order) to the 4 servers in the overrides. In the example, this would be;

- https://hubs.senseicto.com
- https://assets.hubs.senseicto.com
- https://stream.hubs.senseicto.com
- https://cors.hubs.senseicto.com
- https://stream.hubs.senseicto.com:4443

5 First Login

Navigate back the original https://hubs.senseicto.com.

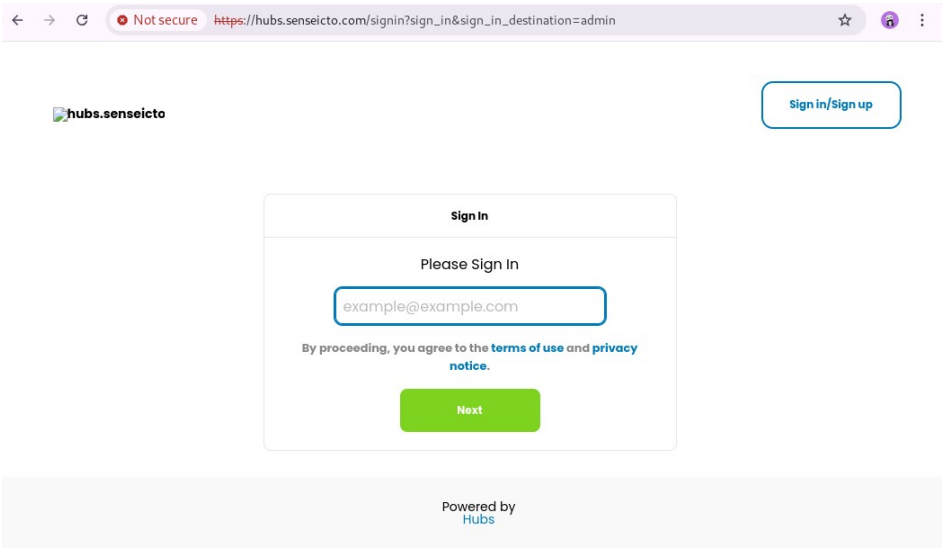


Figure 66: Hubs First Login

Type in the administration email that you specified in the configuration YAML file (e.g. hubsadmin@senseicto.com). You will then be asked to wait for the login link to be used.

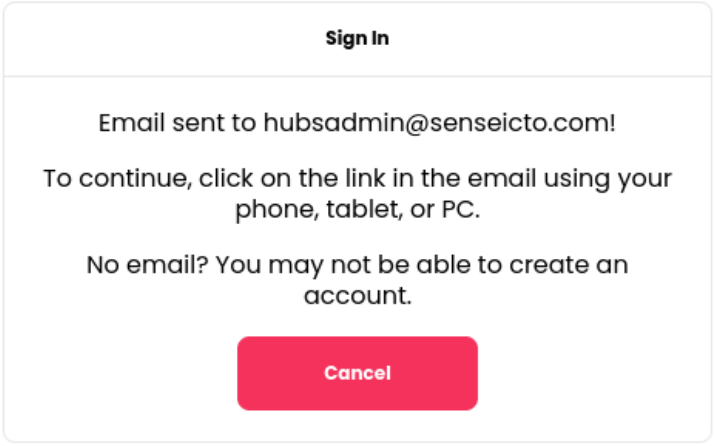


Figure 67: Wait for Login

Open the Citadel application and log in as the hubsadmin account you specified - e.g. hubsadmin@senseitcto.com. Then navigate to the Mail tab, and open the magic link email.

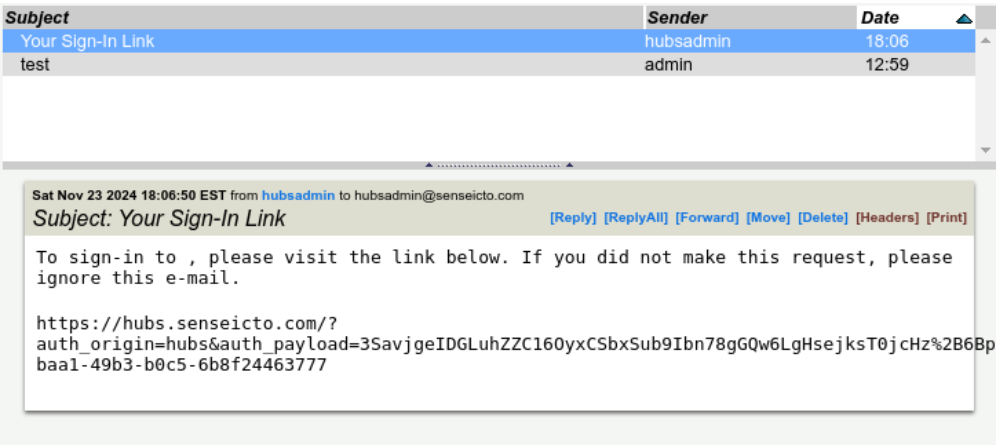


Figure 68: Hubs Magic Link Email

Copy and paste the link to a fresh browser tab. This should provide login verification.

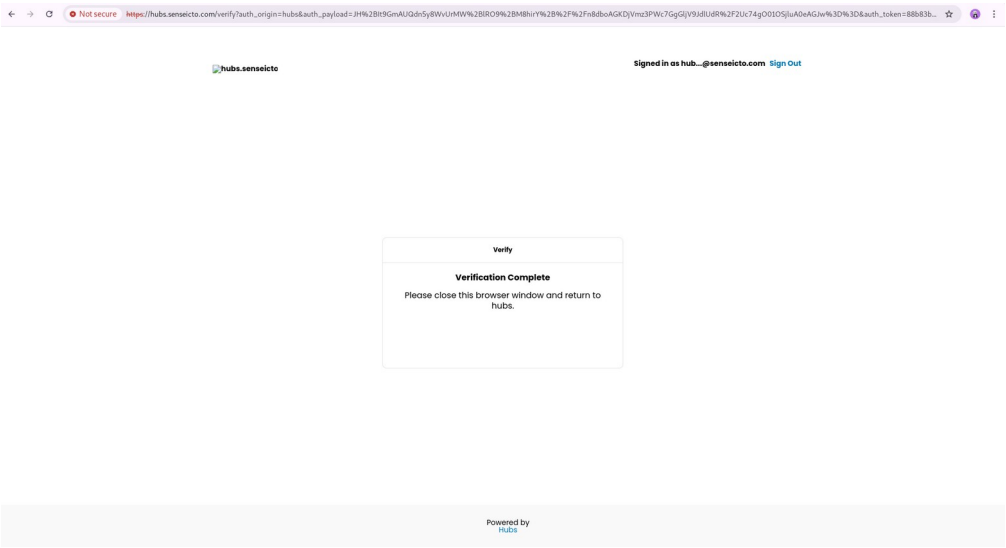


Figure 69: Hubs Login Verification

You can then close this tab, and go back to the original login tab. It should now display the front user screen.

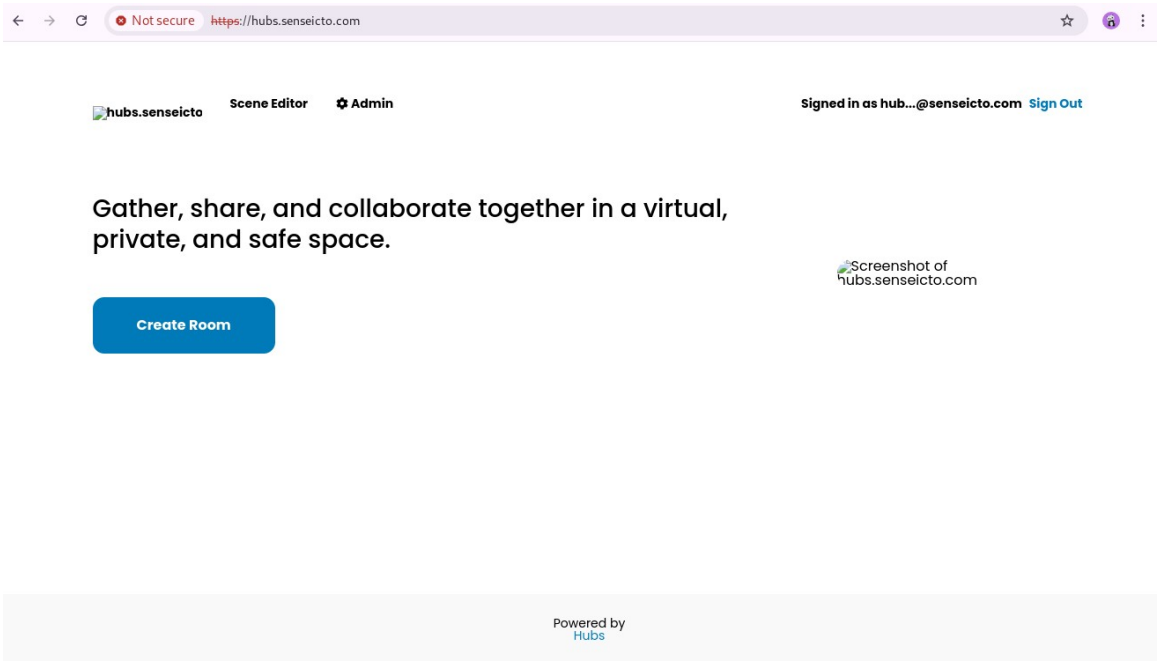


Figure 70: Hubs Welcome Screen

From here, you can create a room (although you are not configured to do so yet, so no useful room will be created!), go to the administrator's console, or go to the scene editor (aka Spoke).

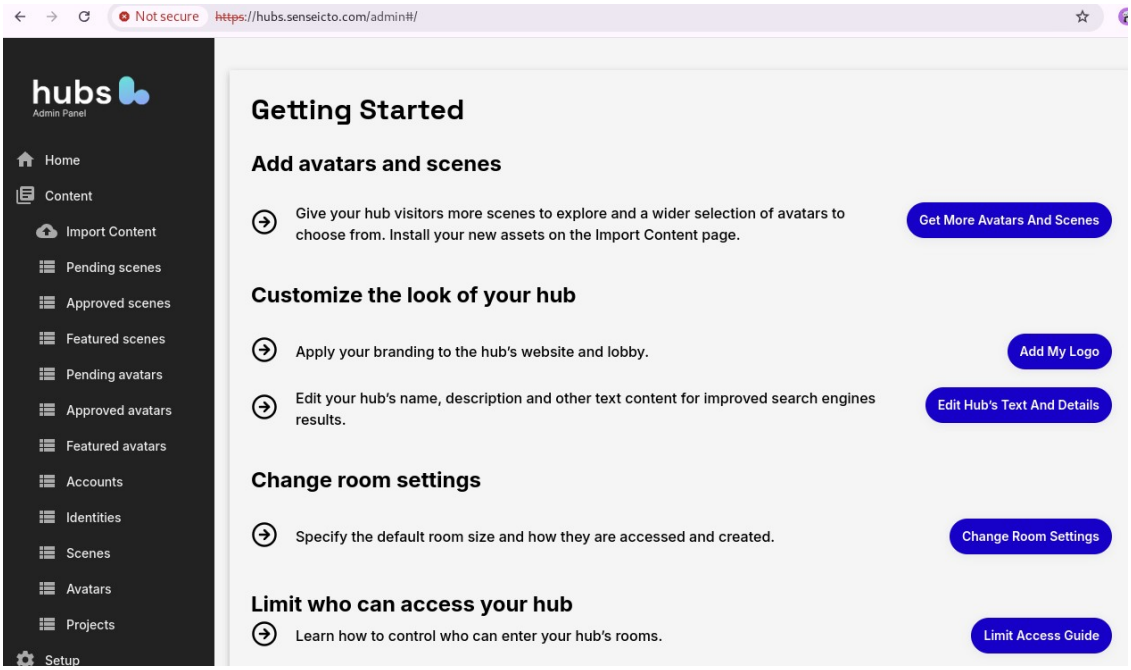


Figure 71: Hubs Administration Console

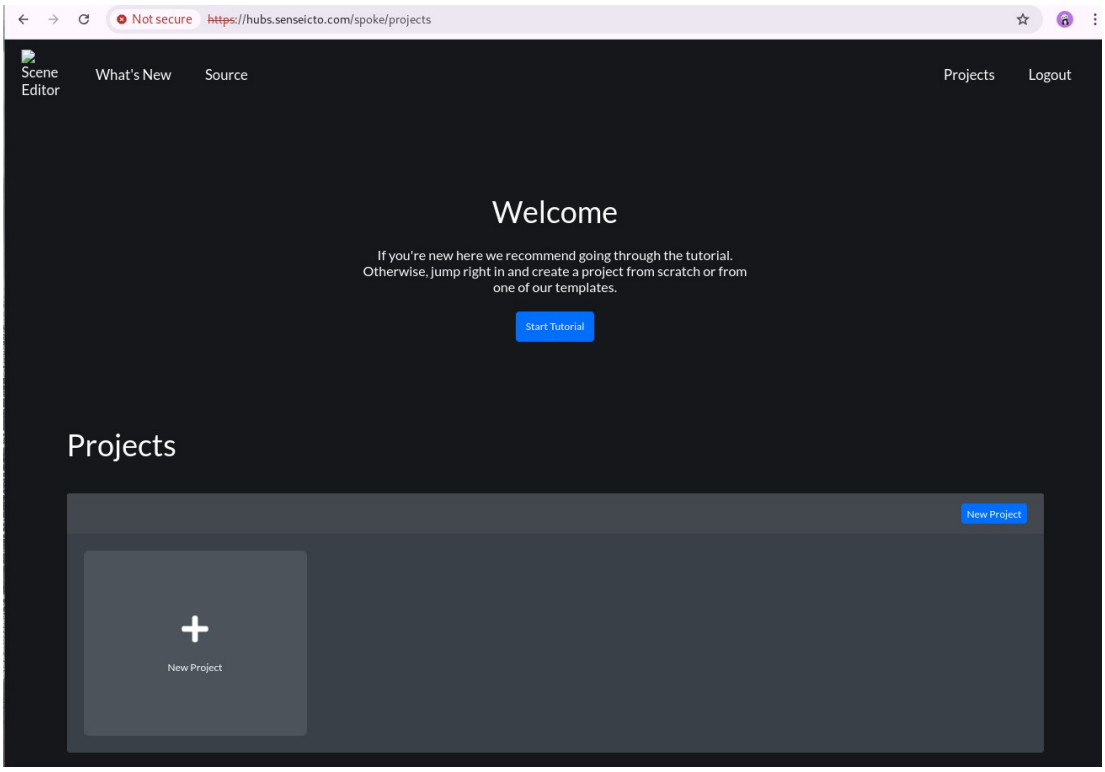


Figure 72: Hubs Scene Editor

5.1 Network Configuration

Another one of the legacy issues of AWS that still exists is the use of the public IP address of the server, specifically when trying to use STUN and WebRTC audio support in order to let users talk to each other in the rooms. Because STUN and ICE are hard-wired to use the public IP address of the server (or, in this case, the Public IP address of the ISP Router), we need to configure the firewall to perform NAT of the external (public) IP to the internal address of the actual Kubernetes worker node on the internal network if we want any users to be able to talk to other users, share video, and more.

By doing this, any browser client behind the firewall will have traffic destined for the public IP address redirected to the local IP address of the COTURN server running on the HubsWorker server.

What is NAT and Why do we need it? NAT, or Network Address Translation, allows the firewall / router to substitute one IP address for another. In this way, a client may request a specific IP address, but actually be routed to a different IP address.

Since the Hubs Cloud CE is hardwired to use its public IP address, we need to have the firewall reroute all traffic destined for the public address back to the local IP address of the Kubernetes worker node.

What is STUN and ICE and TURN and How Does Hubs Use Them? WebRTC is an open-source technology that lets browsers and devices communicate with one another. They do so using peer-to-peer (P2P) connections. In Hubs, this means that every user in the room is establishing a network connection with every other user in the room. If there are three users in the room, then each user is establishing two connections - one to each of the other users. This is one of the reasons why there is a connection limit on the number of concurrent users in a room.

WebRTC is the protocol that enables the uses to share audio, video such as webcams, and more while they are in the room together. It relies on other protocols and services including ICE, STUN and TURN in order to establish the P2P connections. One of the running pods is a TURN server called coturn, for example.

On OPNSense

Open a web browser session to the OPNSense console, and log in as the administrator. Then navigate to Firewall → Settings → Advanced and configure the following:

Network Address Translation

- Reflection for port forwards [check]
- Reflection for 1:1 [check]
- Automatic outbound NAT for Reflection [check]

Press the Save button at bottom of screen.

Create the rule.
First, check your public IP address by opening a browser window and navigating to https://ipinfo.io/ip.

Why Am I Going To ipinfo.io/ip? This is the service that Hubs is using to determine the “public” IP address that your server is located at on the internet. It will try and use this IP address for the clients to connect to the WebRTC services. We are going to redirect these calls to a local (private) IP address instead.

Navigate to Firewall → NAT → Port Forward and add a rule, setting the following;

```
Interface: WAN
Address Family: IPv4
Protocol: Any (*)
Source: Any (*)
Source Port Range: Any (*)
Destination: <public IP> / 24
Destination port range: Any (*)
Redirect target IP: <Kubernetes Worker1 node IP>
Redirect target port: Any (*)
Description: WebRTC Internal Redirect
```

- For the <public IP> value, replace it with the public IP address from ipinfo.io/ip.
- For <Kubernetes Worker1 node IP>, this is the assigned IP address for the worker node. In this example, 192.168.100.31.

Click the Save button at the bottom of the screen.

Click the “Apply changes” button at the top-right of the screen.

You should now have a Port Forward Rule.

			Source		Destination		NAT		
	Interface	Proto	Address	Ports	Address	Ports	IP	Ports	Description
!	LAN	TCP	*	*	LAN address	80, 443	*	*	Anti-Lockout Rule
⟷	WAN		*	*	129.222.167.85/24	*	192.168.100.31	*	

▶ Enabled rule ! No redirect ⟷ Linked rule
▶ Disabled rule ! Disabled no redirect ⟷ Disabled linked rule

Figure 73: Port Forward Public IP Address

6 Creating Your First Room

CONGRATULATIONS! If you have made it this far, you should have a working server!

Let's recap. We now have;

- A working private cloud infrastructure
- A Firewall
- A Hubs Cloud CE Server
- An email server

This is a locally-hosted Hubs server that user's can now share on your local network.

Let's now put it to the test by creating and hosting our first room.

The Hubs Foundation team have archived a number of room, avatar, and other assets in GitHub. Open a browser and navigate to
https://github.com/Hubs-Foundation/hubs-blender-files/tree/main/demo-server
to see what is there.

As a first room, I have always been impressed with the SB22 Conference Room.

- Navigate (or select) https://github.com/Hubs-Foundation/hubs-blender-files/blob/main/demo-server/hubs_scenes_export/Conference_Room_SB22_by_Hubs_Team_id_Qpg584G.glb and click the "Raw" download button in order to download the GLB file.
- On the main Hubs Welcome screen, select the "Scene Editor" to open the Spoke editor.
- Click the New Project button.
  - Click the "Import From Blender" blue button at the top right side of the window.
  - Scene Name: Conference Room SB22
  - Your Attribution: (leave blank for now)
  - Click the Select scene model… button and select the GLB file that you just downloaded to your workstation.
  - Select any image (for now) as the "Scene Thumbnail". (It will not let you publish without it, but you can change it later).
  - Click the Publish button.

You can now create a live room. Click on the freshly created scene, and then click the blue "Open Scene" button.

Click the "Create a room…" button.

In order to test the audio WebRTC configuration, open a different browser (e.g. Chrome and Firefox), or open a different Browser Profile, and copy and paste the URL of the new room. Enter the room on both browser windows. After allowing audio, you should hear an echo - it works!

If you, instead, hear nothing, then it’s likely the firewall setting for the NAT rule is incorrect. Go back and check that it is correct.

7 Troubleshooting Notes

7.1 Connecting Clients Using Self-Signed Certificates

You will need to accept all self-signed certificates in each of your browser profiles before actually connecting to a room. If the browser profile that you are using has not already accepted all of these certificates, you will likely just get a “white screen of death”. If you open the browser console inspector, you will likely see the URL of the Hubs services that you have forgotten to connect to manually in order to accept the certificates.

Using the browser profile that you want authenticated, navigate to each of the following and accept the certificates:

- https://<yourdomain>
- https://assets.<yourdomain>
- https://cors.<yourdomain>
- https://stream.<yourdomain>
- wss://stream.<yourdomain>:4443 (or https://stream.<yourdomain>:4443)

Then navigate to your room.

7.2 No Shared Audio

This is likely a firewall issue.

As mentioned above, in order for the WebRTC shared audio to work, the firewall needs to redirect all traffic requests for the server’s public IP address back to the private IP address. This is done via a Network Address Translation (NAT) rule.

Double check that the firewall rule is

- **using the IP address of the HubsWorker guest. In this example, the IP address is 192.168.100.31 (You specified it at the start of the configuration when the hubs master, worker, and workstation guests were created in the cloud.)**
- using “any” protocol
- specifies “any” source
- specifies port range from “any” to “any”
- **specifies the Destination as “Single host or Network, with a value of the public IP address returned from browsing to ipinfo.io/ip**
- specifies Destination port range from “any” to “any”
- specifies Redirect target IP as “Single host or Network” wit ha value of the worker node IP address from the DHCP assignment
- specifies the Redirect target port as “Any” (it display “other”).

The bullets highlighted in bold are typically the issue. Pay close attention to the IP address returned by browsing to ipinfo.io/ip as it will change periodically based on your ISP.

Save and accept the changes.

8 Publicly Accessible Server

Now that the server is running and properly configured, it is available to anyone on the LAN side of the network, i.e. inside the firewall. Referring back to Figure 1, If you connect your user's workstations to the ethernet switch, they will automatically be assigned an IP address on the internal LAN segment, and can use the Hubs servers, share rooms, etc.

For most, this is good enough. Good job! You can now go to the Hubs Foundation documentation and start configuring your own private server, designing scenes, and most everything else that you wanted to do with Hubs Cloud CE.

We can, however, take it a few steps further, namely allowing public internet access to our Hubs rooms and services. A great use case is using a room for a meeting, hosting a webinar, or any other reason you may need remote access to your Hubs instance.

In order to allow public access, we need to do the following;

1. Install and configure a Proxy Server. In this case, we will be using the Apache Traffic Server (https://trafficserver.apache.org/) as a reverse proxy.
2. Add inbound port forwarding rules to the Firewall.
3. (Optional, but strongly recommended) Register a domain name.
4. (Optional, but strongly recommended) Use a static public IP address, or configure a Dynamic DNS (DDNS) rule for the server.

Check With Your Internet Service Provider.

Before proceeding, ensure that your ISP supports allowing inbound connections to your servers. Some ISPs use a networking scheme called Carrier-Grade Network Address Translation (CGNAT). This means that you are actually sharing the public IP address with other customers of your ISP, so there is no way to allow for inbound server connections.

As well, your ISP may block remote access to certain ports. A good example is Port 25 for SMTP mail delivery. While we can overcome some of these blocked ports (including Port 25), the ISP may block them all.

Finally, the ISP or their equipment, such as your modem that you are using, may not support inbound connection routing, commonly known as port forwarding.

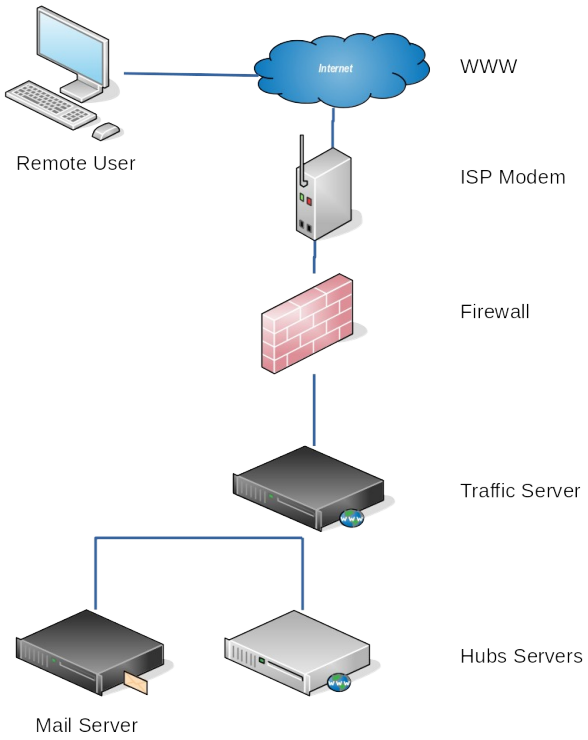


Figure 74: Server Connections

As we can see, in order for a public user to get access to the Hubs servers, communications needs to go through several layers;

- The ISP Modem/Router, which is assigned the public address. It needs to port forward to the Firewall.
- The Firewall, which needs to port forward to the Traffic Server.
- The Traffic Server, which proxies the Hubs Servers and the Mail server.

This is where the network configurations can appear complicated, but it is actually quite straightforward.

Since we are dealing with what, from the user's perspective, appears to be web servers, we will start by generating some certificates in order to properly secure those connections. If we wanted to, we could continue to use the self-signed certificates from the servers, but that would be complicated, messy, and hard for the public users.

8.1 Certificate Generation

Generated vs. Self-Signed Certificates. A *generated certificate* is issued by a trusted Certificate Authority (CA) and provides validation of the entity's identity. It provides a trust chain that links your certificate to a trusted "signing" certificate that the user's browser already trusts.

A *self-signed certificate*, on the other hand, can be created by you independently. These are typically used for internal or testing purposes but lacks the trust chain, which is why browsers warn of "self-signed certificates". You can manually add these certificates to be trusted by your browser, as we have already done above, but for public use it can make it messy and complicated for the end-users.

If you want to use a validated certificate to avoid the "self-signed certificate" issues, especially for public access via the Proxy Server, you can generate certificates for the domain that you own.

Note: This section assumes that you have registered your own domain with a registrar. Common issuers to look at include Google Domains, Namecheap, GoDaddy and others.

While there are a number of commercial and non-commercial certificate generators available, we will be using Let's Encrypt for our servers. Alternatively, you are free to use any generated certificates (even your own self-signed certificates) in place of Let's Encrypt generated ones.

Create a new certificate directory

```
$ mkdir -p ~/Documents/Certificates
$ cd ~/Documents/Certificates
```

On the Hubs Master node, install certbot in order to generate Let's Encrypt certificates

```
$ sudo apt-get install -y certbot
```

On the Hubs Master node, generate a wildcard cert using

```
$ certbot certonly --manual --preferred-challenges dns -d '<default domain>' -d
'*.<default domain>'
```

where <default domain> is replaced with the domain specified in the DNS. For this example;

```
$ certbot certonly --manual --preferred-challenges dns -d 'hubs.senseicto.com' -d
'*.hubs.senseicto.com' --work-dir ~/Documents/Certificates --config-dir
~/Documents/Certificates --logs-dir ~/Documents/Certificates
```

Follow the instructions, creating the "proof" text record in your registrar's DNS settings.

Note: When using Let's Encrypt to generate a "wildcard" certificate, I have found the first try may fail. Simply re-run the certbot command again, replace the generated token in the DNS entry, and wait for the update to propagate through the global DNS. You can do this by DIGing for the TXT entry, for example:

$ dig @8.8.8.8 -t txt _acme-challenge.hubs.senseicto.com

matching the returned token with the new one you just replaced in the DNS TXT entry.

Certificates and Kubernetes Secrets: Certificates play a crucial role in securing communication between services in Kubernetes, including encrypting traffic using TLS/SSL. Kubernetes Secrets provide a secure method to store key materials and make them accessible to the workloads that require them.

List the secrets

```
$ sudo kubectl -n localhubs get secrets
```

Delete the original secret:

```
$ sudo kubectl -n localhubs delete secret cert-hcce
```

Create a replacement:

```
$ sudo kubectl -n localhubs create secret generic cert-hcce --from-
file=tls.crt=/home/terry/Documents/Certificates/archive/hubs.senseicto.com/
fullchain1.pem
--from-file=tls.key=/home/terry/Documents/Certificates/archive/hubs.senseicto.com
/privkey1.pem --type=kubernetes.io/tls
```

Replace the directory names with ones appropriate for your environment.

Now reboot the Hubs Master and Worker nodes by logging in to each and typing

```
$ sudo reboot
```

Once they have both completely rebooted, open a browser tab and clear the previous history and data for your hubs site. Then navigate to your hubs site URL, e.g. https://hubs.senseicto.com

With properly signed certificates, users will no longer need to accept certificates for any of the default domain URLs. These certificates will also be used by the Proxy Server, as described in the following section.

Remote Mail Access

If you intend to let external users authenticate themselves, of if you intend to access the Hubs CE servers remotely, then you will need to provide proxy access to the Citadel Mail Server as well.

With Citadel, all protocols which are capable of encryption, such as IMAP and SMTP and XMPP, and even HTTPS in WebCit, will look for TLS keys and certificates in the following locations:

- /usr/local/citadel/keys/citadel.key - The private key for this server
- /usr/local/citadel/keys/citadel.csr - A certificate signing request
- /usr/local/citadel/keys/citadel.cer - The signed certificate (full chain with any intermediate certificates appended)

Generate an appropriate Let's Encrypt certificates (Certificate and Private Key) using certbot for your Citadel server as described above, but do so on the Citadel host.

Delete the original self-signed certificates.

```
$ sudo su root
# rm /usr/local/citadel/keys/citadel.key
# rm /usr/local/citadel/keys/citadel.cer
```

Create symbolic links in place of the placeholder files at the locations described above. Note that there is no signing request, so no link required. Using our example;

```
# ln -sfv /home/terry/Documents/Certificates/privkey1.pem
/usr/local/citadel/keys/citadel.key
# ln -sfv /home/terry/Documents/Certificates/fullchain1.pem
/usr/local/citadel/keys/citadel.cer
```

Reboot the Citadel Server

```
# reboot
```

Note: If you are using another certificate generation provider that involves a signing request, then create a corresponding link for the signing request as well.

8.2 Reverse Proxy

In order to allow public internet users to access the Hubs server, we need to be able to accept the incoming connection requests, and route them to the appropriate servers and services. As you have seen with the Kubernetes as well as the firewall DHCP and DNS configurations, we need to be able to uniquely address the Hubs Master, Worker, and the Kubernetes pods containing the services such as cors, assets and stream services.

The problem is this - we only have one IP address, and at least 5 different servers that we need to access. On the LAN side of the firewall (the private network), that's easy - they each have a unique IP address and DNS entry. On the WAN side (the public internet), we only have the one public IP address.

The solution is to use a reverse proxy.

What is a reverse proxy? Simply put, a reverse proxy can accept connection requests on one public IP address, and then reroute those connections to the appropriate servers and services on the internal network based on the requested URL.

By placing a reverse proxy at the front door of your firewall, we can let it manage the public connection requests to our Hubs server.

8.2.1 Create the Cloud Guest

As mentioned at the start of the Citadel server installation, refer to that section for details on creating a guest on the Proxmox host. Use the following details specific to the Apache Traffic Server guest;

In Proxmox Cloud

Name: TrafficServer
ISO Image: debian-xx.x.x-amd64-DVD-1.iso (from the drop-down list of ISOs)
Memory: 8192 MiB
Processors: 4 Sockets, 4 Cores (total of 16 VCPUs)
Hard Disk: 50G

In OPNSense Firewall

In the firewall, create the DHCP static entry as follows:
MAC Address: <as created by ProxMox>
IP Address: 192.168.100.40
Hostname: trafficserver
Description: Apache Reverse Proxy
Domain name: <your domain>

where <your domain> is whatever you decided earlier - for this example I am using *senseicto.com*

Leave everything else as default.

8.2.2 Install the Traffic Server Pre-Requisites

Java Installation

As Apache Traffic Server uses Java, we will be installing Amazon's Corretto OpenJDK.

To use the Corretto Apt repositories on Debian, we need to first import the Corretto public key and then add the repository to the system list by using the following commands:

```
# apt-get install -y gnupg  software-properties-common

# wget -O- https://apt.corretto.aws/corretto.key | apt-key add

# echo 'deb https://apt.corretto.aws stable main' >>
/etc/apt/sources.list.d/corretto-stable.list

# apt-get update
```

Now we can go ahead and install Java Corretto v21

```
# apt-get install -y java-21-amazon-corretto-jdk
```

Check the version that is installed.

```
# java -version
openjdk version "21.0.5" 2024-10-15 LTS
OpenJDK Runtime Environment Corretto-21.0.5.11.1 (build 21.0.5+11-LTS)
OpenJDK 64-Bit Server VM Corretto-21.0.5.11.1 (build 21.0.5+11-LTS, mixed mode,
sharing)
```

Build Pre-Requisites

Install all of the libraries and utilities required to build Apache Traffic Server.

```
# apt-get install -y  git cmake ninja-build zlib1g zlib1g-dev pkg-config libtool
tcl tcl-dev expat libpcre3 libpcre3-dev libcap-dev flex hwloc libhwloc-dev lua5.4
libncurses-dev curl libcurl4-openssl-dev g++ libssl-dev libxml2-dev dh-autoreconf
libmodule-install-perl luajit libbrotli-dev liblzma-dev libluajit-5.1-dev
libunwind-dev libpcre2-dev
```

8.2.3 Traffic Server Installation

References

https://docs.trafficserver.apache.org/en/latest/getting-started/index.en.html#installing-from-source-code

Installation Steps

Step 1: Clone the repository (you may skip this if you have downloaded an archive of the source code to build a specific official release of Traffic Server instead of the HEAD from source control):

```
$ mkdir -p ~/Documents
$ cd ~/Documents
$ git clone https://git-wip-us.apache.org/repos/asf/trafficserver.git
```

Step 2: Change to the cloned (or unarchived) directory:

```
$ cd trafficserver/
```

Step 3: Configure the source tree:

```
$ cmake -B build -DCMAKE_INSTALL_PREFIX=/opt/ts
```

Step 4: Build Traffic Server with the generated build, and test the results:

```
$ cmake --build build
$ cmake --build build -t test
```

This returned 59% tests passed, 44 tests failed out of 107 for me.

Step 5: Install Traffic Server to the configured location:

```
$ su - root
# cd /home/<accountname>/Documents/trafficserver
# cmake --install build
```

where <accountname> is the login credential you used to login to Debian.

Optionally, you may find it prudent to run the regression tests on your newly installed Traffic Server:

```
# cd /opt/ts
# bin/traffic_server -R 1
```

This returned REGRESSION_TEST DONE: PASSED

8.2.4 Reverse Proxy Configuration

Copy the Certificates

First, we need to copy the certificates from the Hubs Master that were generated by Let's Encrypt earlier.

Why Does the Reverse Proxy Need the Certificates? As described in an earlier note, the reverse proxy is what users will be connecting to publicly. It then redirects the connection to the actual Hubs server (or CORS, STREAM and ASSET servers) as required. In order to properly connect via HTTPS, the reverse proxy needs to be able to present those certificates as if it was the actual server that it is proxying.

The easiest way to do this is by using SCP.

What is "scp"? SCP, or Secure Copy Protocol, which provides file transfer between two hosts using Secure Shell (SSH). Since we already have SSH set up on our nodes, we can use SCP to move files between them.

On the Traffic Server node, create a directory in which to store the certificates

```
$ mkdir -p ~/Documents/Certificates/<base domain>
```

In this example, this would be

```
$ mkdir -p ~/Documents/Certificates/hubs.senseicto.com
```

On the Hubs Master node, copy the certificates to the Trafficserver node.

```
$ cd ~/Documents/Certificates/archive
$ scp -r <base domain> <user>@trafficserver.<base
domain>:/home/<user>/Documents/Certificates/
```

In this case, replace the <base domain> with your base domain, and the <user> and <base domain> with the OS user that you created on the trafficserver. In my example;

```
$ scp -r hubs.senseicto.com
terry@trafficserver.senseicto.com:/home/terry/Documents/Certificates/
```

Check that the files made it over to the trafficserver

```
$ ls ~/Documents/Certificates/hubs.senseicto.com/
fullchain1.pem  privkey1.pem
```

If you generated certificates for the mail server, then perform the same steps on the Citadel host,

replacing <base domain> with <citadel host fqdn>. In this example, it would be citadel.senseicto.com.

What is “fqdn”? It stands for Fully Qualified Domain Name, and represents the complete <hostname>.<domain name> that is assigned by DHCP in our case.

Create the Reverse Proxy Mappings

Log into the Traffic Server. Change to root, and navigate to the configuration files directory

```
$ su - root
# cd /opt/ts/etc/trafficserver
```

Edit the remap.config file

```
$ nano remap.config
```

Scroll to the bottom (using arrow keys - google nano if you need to understand how to edit with it). Add lines for remapping the Email server. In our example, this looks like

```
#
# Citadel Mail Server
#
map https://citadel.senseicto.com https://citadel.senseicto.com
reverse_map https://citadel.senseicto.com https://citadel.senseicto.com
```

What is Remapping? In order for the proxy server to allow connections from a single listener to a plurality of servers / domains / urls, it needs to understand what URL to look for inbound, and where to route the requests to internally. In the example above, we are telling Traffic Server how to accept requests destined for the email server publicly, and reroute it to the actual citadel server behind the firewall.

We are also telling Traffic Server that responses from the Citadel email server should be mapped back to the original requested URL.

Why does it look like we are mapping the same thing to itself? In our case, we are able to cheat a little. Normally, the mapping rule would map the external URL to an internal URL. In our case, because we control both the Global DNS defining the external domain, as well as the firewall’s DNS internally, we can point the same domain to two different IP addresses.

Let’s say the Traffic Server is getting public traffic at 10.0.0.5. So we can create a global DNS A record with our registrar that points citadel.senseicto.com to 10.0.0.5. We have the actual citadel server located at 192.168.100.10. We already created a DNS A record on our internal firewall by virtue of the DHCP configuration we did when we installed Citadel. So when an external user does a lookup (nslookup) on citadel.senseicto.com, they get back 10.0.0.5 from the global DNS. On the Traffic Server, the same lookup will return our internal address from the firewall DNS server;

```
# nslookup citadel.senseicto.com
Server:    192.168.100.1
Address:   192.168.100.1#53
Name: citadel.senseicto.com
Address: 192.168.100.10
```

This way, the external user never needs the actual internal IP address of the Citadel server. It sees the Traffic Server, thinking that it is the Citadel server. The Traffic Server, in turn, sees the actual IP address of the Citadel server, and makes the connection to it on behalf of the external user. That's a proxy server.

While editing remap.config, add all of the Hubs Community Edition servers. In our example, this looks like

```
#
# Hubs Community Edition
#
map https://hubs.senseicto.com/ https://hubs.senseicto.com/
reverse_map https://hubs.senseicto.com/ https://hubs.senseicto.com/
map http://hubs.senseicto.com/ http://hubs.senseicto.com/
reverse_map http://hubs.senseicto.com/ http://hubs.senseicto.com/
#
# wss
#
map wss://hubs.senseicto.com/ wss://hubs.senseicto.com/
reverse_map wss://hubs.senseicto.com/ wss://hubs.senseicto.com/
map wss://hubs.senseicto.com:4443/ wss://hubs.senseicto.com:4443/
reverse_map wss://hubs.senseicto.com:4443/ wss://hubs.senseicto.com:4443/
#
# Stream wss
#
map wss://stream.hubs.senseicto.com/ wss://stream.hubs.senseicto.com/
reverse_map wss://stream.hubs.senseicto.com/ wss://stream.hubs.senseicto.com/
map wss://stream.hubs.senseicto.com:4443/ wss://stream.hubs.senseicto.com:4443/
reverse_map wss://stream.hubs.senseicto.com:4443/ wss://stream.hubs.senseicto.com:4443/
#
# Assets wss
#
map wss://assets.hubs.senseicto.com/ wss://assets.hubs.senseicto.com/
reverse_map wss://assets.hubs.senseicto.com/ wss://assets.hubs.senseicto.com/
map wss://assets.hubs.senseicto.com:4443/ wss://assets.hubs.senseicto.com:4443/
reverse_map wss://assets.hubs.senseicto.com:4443/ wss://assets.hubs.senseicto.com:4443/
#
# CORS wss
#
```

```
map wss://cors.hubs.senseicto.com/ wss://cors.hubs.senseicto.com/
reverse_map wss://cors.hubs.senseicto.com/ wss://cors.hubs.senseicto.com/
map wss://cors.hubs.senseicto.com:4443/ wss://cors.hubs.senseicto.com:4443/
reverse_map wss://cors.hubs.senseicto.com:4443/ wss://cors.hubs.senseicto.com:4443/
#
# STUN wss
#
map wss://129.222.136.74/ wss://hubsworker.sensecto.com/
reverse_map wss://hubsworker.sensecto.com/ wss://129.222.136.74/
map wss://129.222.136.74:4443/ wss://hubsworker.sensecto.com:4443/
reverse_map wss://hubsworker.sensecto.com:4443/ wss://129.222.136.74:4443/
#
# Assets
#
map http://assets.hubs.senseicto.com/ http://assets.hubs.senseicto.com/
reverse_map http://assets.hubs.senseicto.com/ http://assets.hubs.senseicto.com/
map https://assets.hubs.senseicto.com/ https://assets.hubs.senseicto.com/
reverse_map https://assets.hubs.senseicto.com/ https://assets.hubs.senseicto.com/
#
# Stream
#
map http://stream.hubs.senseicto.com/ http://stream.hubs.senseicto.com/
reverse_map http://stream.hubs.senseicto.com/ http://stream.hubs.senseicto.com/
map https://stream.hubs.senseicto.com/ https://stream.hubs.senseicto.com/
reverse_map https://stream.hubs.senseicto.com/ https://stream.hubs.senseicto.com/
#
# Cors
#
map http://cors.hubs.senseicto.com/ http://cors.hubs.senseicto.com/
reverse_map http://cors.hubs.senseicto.com/ http://cors.hubs.senseicto.com/
map https://cors.hubs.senseicto.com/ https://cors.hubs.senseicto.com/
reverse_map https://cors.hubs.senseicto.com/ https://cors.hubs.senseicto.com/
```

Why are we mapping 129.222.136.74 to hubsworker.sensecto.com in this example?
Recall that the WebRTC configuration is calling https://ipinfo.io/ip in order to identify the public IP address of the server. It is used, as-is, by the client to reach the COTURN server. This appears to be one of the “hang-over” hard-wired configurations from the AWS hosting days. So 129.222.136.74 is what my call to ipinfo returned at the time of this writing.

In your case, you need to get the public IP address of your server. You can do this at a command prompt on the Hubs Worker;

```
$  curl https://ipinfo.io/ip
129.222.167.85
```

It should return the IP address to use in this mapping.

As for the Hubs Worker, replace the hubsworker.senseicto.com with your DNS entry for the Hubs Worker node that you specified on the firewall DHCP configuration.

Note: If your ISP changes your public IP address, you will need to edit this mapping to reflect the change and restart the Traffic Server. You will also need to make similar changes to the other public IP references in the firewall for NAT.

Configure the Traffic Server SSL Settings

We have given the Traffic Server rules with which to connect requested URLs by public users to internal servers. But that's only half the story. Since the Traffic Server is masquerading as our Citadel email server as well as the Hubs CE servers, it needs to present the SSL Certificates in order to make secure connections between itself and the client browsers.

Navigate to the root of the traffic server installation

```
$ su - root
# cd /opt/ts
```

Create SSL Directories

```
# mkdir -p /opt/ts/etc/ssl/certs
# mkdir -p /opt/ts/etc/ssl/keys
```

Create symbolic links to the certificates and keys previously copied

```
# ln -s /home/terry/Documents/Certificates/citadel.senseicto.com/privkey1.pem
/opt/ts/etc/ssl/keys/citkey.pem


# ln -s
/home/terry/Documents/Certificates/citadel.senseicto.com/fullchain1.pem
/opt/ts/etc/ssl/certs/citcert.pem


# ln -s /home/terry/Documents/Certificates/hubs.senseicto.com/privkey1.pem
/opt/ts/etc/ssl/keys/hubskey.pem


# ln -s /home/terry/Documents/Certificates/hubs.senseicto.com/fullchain1.pem
/opt/ts/etc/ssl/certs/hubscert.pem
```

Why are we using symbolic links? In this case, we are using symbolic links for two reasons. The first is naming. Traffic server wants to see unique filenames for each certificate/key pair. Instead of copying and renaming, we can use a link to create the naming relationship between them.

The other reason is for ongoing management and maintenance. Since the Let's Encrypt certificates only last 90 days, we will need to regenerate them and copy

```
them again to the traffic server. By using a link, we only need to copy the fresh
files over the stale ones. The link will automatically point to the new files,
and the traffic server will reference them automatically.
```

ssl_multicert.config

Edit the ssl_multicert.config file and append the certificate references that we jut created.

```
# nano /opt/ts/etc/trafficserver/ssl_multicert.config
ssl_cert_name=hubscert.pem ssl_key_name=hubskey.pem
ssl_cert_name=citcert.pem ssl_key_name=citkey.pem
```

records.yaml

We need to add configuration information to the records.yaml file. Because this is a yaml file, be aware of the leading whitespaces.

Edit the records.yaml file as follows;

```
# nano /opt/ts/etc/trafficserver/records.yaml
```

Around line 37, change the following (in bold)

```
   diags:
      debug:
        enabled: 3
        tags: http|dns|https
```

Around line 144, change the following (in bold)

```
    server_ports: 80 443:ssl 4443:ssl
```

Around line 188, add the following (in bold)

```
   res_track_memory: 0

# https://docs.trafficserver.apache.org/en/latest/admin-guide/files/
records.yaml.en.html#ssl-termination
  ssl:
    server:
      private_key:
        path: etc/ssl/keys/
      cert:
        path: etc/ssl/certs/
```

Around line 224, change the following (in bold)

```
    pristine_host_hdr: 1
```

Save the file and exit (ctrl-x).
Important Note: It appears the permissions to the certificates and keys needs to be 755. As the traffic server OS user, run chmod, e.g.

```
$ chmod -R 755 /home/terry
```

Configuring as a service

Navigate to the trafficserver source directory

```
# cd <sourcedirectory>/rc
```

In our example, this is

```
# cd /home/terry/Documents/trafficserver/rc
```

Make and install the service

```
# cmake -DCMAKE_INSTALL_PREFIX=/opt/ts
```

Copy the trafficserver script to /etc/init.d directory from the original source directory where you built it. In our example, this is

```
# cp trafficserver /etc/init.d
```

Configure as a service

```
# cd /etc/init.d
# chmod +x trafficserver
# update-rc.d trafficserver defaults
# systemctl daemon-reload
```

Restart the Traffic Server
service trafficserver start

Check the status

```
# service trafficserver status
● trafficserver.service - LSB: Startup/shutdown script for the Apache Traffic Server
     Loaded: loaded (/etc/init.d/trafficserver; generated)
     Active: active (running) since Wed 2024-12-18 18:36:48 EST; 2s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 9122 ExecStart=/etc/init.d/trafficserver start (code=exited, status=0/SUCCESS)
      Tasks: 34 (limit: 9471)
     Memory: 79.0M
```

```
        CPU: 675ms
     CGroup: /system.slice/trafficserver.service
             ├─9132 /opt/ts/bin/traffic_server
             └─9134 traffic_crashlog --syslog --wait --host Linux-x86_64 --user nobody

Dec 18 18:36:47 trafficserver systemd[1]: Starting trafficserver.service - LSB: Startup/shutdown
script for the Apache Traffic Server...
Dec 18 18:36:48 trafficserver trafficserver[9122]: Starting Apache Traffic Server: trafficserver.
Dec 18 18:36:48 trafficserver systemd[1]: Started trafficserver.service - LSB: Startup/shutdown script
for the Apache Traffic Server.
Dec 18 18:36:49 trafficserver traffic_server[9132]: NOTE: --- traffic_server Starting ---
Dec 18 18:36:49 trafficserver traffic_server[9132]: NOTE: traffic_server Version: Apache Traffic
Server - traffic_server - 10.1.0 - (build # 0 on Dec  4 2024 at 17:32:57)
```

The Traffic Server will now startup at boot time should you ever restart the server.

8.3 Firewall Rules

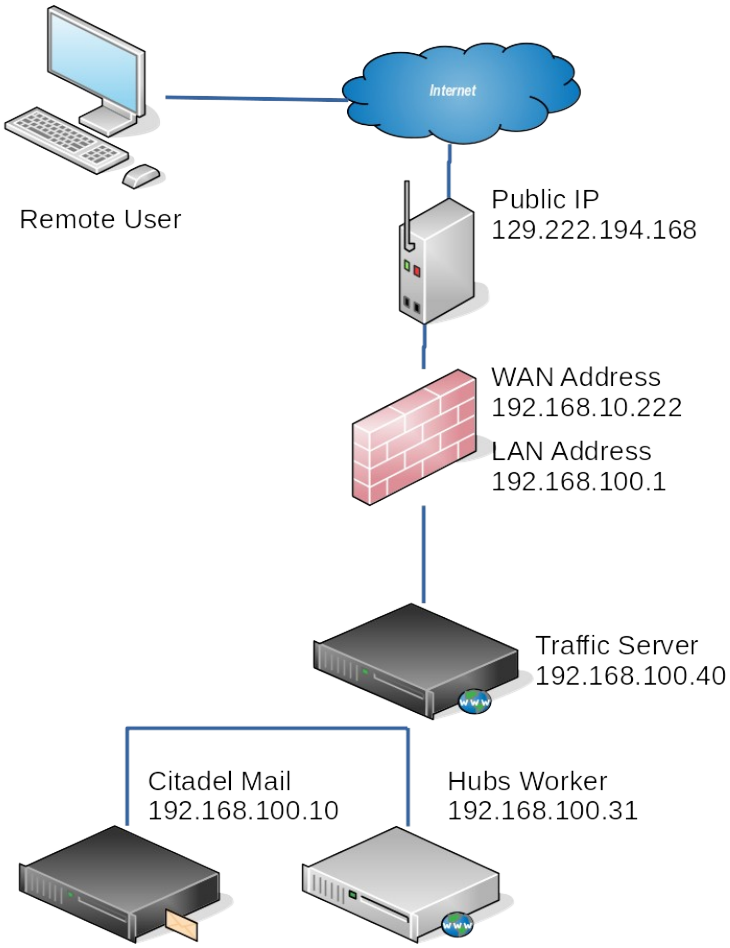


Figure 75: Server Network Addresses

In order to allow public users (outside your firewall) access to the Hubs server, we need to forward internet traffic to the hubsworker node. Recall that in our configuration, traffic from public users must traverse the ISP Router, followed by the Firewall, then the Traffic Server, and ultimately the HubsWorker and Citadel email servers.

At this juncture, we have a working Hubs server, and traffic server / reverse proxy. Next up the chain is the firewall, where we will need to accept any traffic and forward the network traffic it to either the Hubs Worker node, or the traffic server, based on the requested port - i.e. port forwarding.

What is Port Forwarding? When a connection is made between two computers, both machines are identified by their IP address, and the servers/services that they are hosting are identified by a network port (between 0-65535). Port forwarding allows external devices to access services on a private network by redirecting requests from a specific port on a public IP address (in this case our ISP router and firewall) to a specific port and device (our traffic server and Hubs servers) on a private IP address.

We have already created a Network Address Translation rule in order to let users inside the firewall share their audio while in a room - redirecting WebRTC connection traffic addressed for the public IP address to the private address of the Hubsworker node. We now need to create more rules in order to forward some of the requests destined for the firewall and redirect them to the traffic server or the Hubsworker servers. We determine the forward destination based on the port being requested.

Forward all HTTPS traffic to the traffic server

Navigate to Firewall → NAT → Port Forward and add a rule, setting the following;

```
Interface: WAN
Address Family: IPv4
Protocol: TCP/UDP
Source: Any (*)
Source Port Range: Any (*)
Destination: <WAN Address> / 24
Destination port range: HTTPS
Redirect target IP: <Traffic Server>
Redirect target port: HTTPS
Description: HTTPS forward to Traffic Server
```

- For the <WAN Address> value, replace it with the WAN Address of the firewall, in this example it is 192.168.10.222.
- For <Traffic Server>, this is the assigned IP address for the Traffic Server. In this example it is 192.168.100.40.

Forward all port 4443 traffic to the traffic server

Navigate to Firewall → NAT → Port Forward and add a rule, setting the following;

```
Interface: WAN
Address Family: IPv4
Protocol: TCP/UDP
Source: Any (*)
Source Port Range: Any (*)
```

```
Destination: <WAN Address> / 24
Destination port range: Other (from: 4443 to: 4443)
Redirect target IP: <Traffic Server>
Redirect target port: Other (4443)
Description: HTTPS forward to Traffic Server
```

- For the <WAN Address> value, replace it with the WAN Address of the firewall, in this example it is 192.168.10.222.
- For <Traffic Server>, this is the assigned IP address for the Traffic Server. In this example it is 192.168.100.40.

Forward all STUN requests to the Hubsworker

Navigate to Firewall → NAT → Port Forward and add a rule, setting the following;

```
Interface: WAN
Address Family: IPv4
Protocol: TCP/UDP
Source: Any (*)
Source Port Range: Any (*)
Destination: <WAN Address> / 24
Destination port range: STUN
Redirect target IP: <Hubs Worker Node>
Redirect target port: STUN
Description: STUN Forwarding
```

- For the <WAN Address> value, replace it with the WAN Address of the firewall, in this example it is 192.168.10.222.
- For <Hubs Worker Node>, this is assigned IP address for the Hubsworker. In this example it is 192.168.100.31.

Forward all port 5349 requests to the Hubsworker

Navigate to Firewall → NAT → Port Forward and add a rule, setting the following;

```
Interface: WAN
Address Family: IPv4
Protocol: TCP/UDP
Source: Any (*)
Source Port Range: Any (*)
```

```
Destination: <WAN Address> / 24
Destination port range: Other (from: 5349 to: 5349)
Redirect target IP: <Hubs Worker Node>
Redirect target port: Other (5349)
Description: STUN Forwarding
```

- For the <WAN Address> value, replace it with the WAN Address of the firewall, in this example it is 192.168.10.222.
- For <Hubs Worker Node>, this is assigned IP address for the Hubsworker. In this example it is 192.168.100.31.

			Source		Destination		NAT		
	Interface	Proto	Address	Ports	Address	Ports	IP	Ports	Description
!	LAN	TCP	*	*	LAN address	80, 443	*	*	Anti-Lockout Rule
	WAN		*	*	129.222.167.85/24	*	192.168.100.31	*	WebRTC Internal Redirect
	WAN	TCP/UDP	*	*	192.168.10.222/24	443 (HTTPS)	192.168.100.40	443 (HTTPS)	HTTPS forward to TrafficServer
	WAN	TCP/UDP	*	*	192.168.10.222/24	4443	192.168.100.40	4443	HTTPS forward to TrafficServer
	WAN	TCP/UDP	*	*	192.168.10.222/24	3478 (STUN)	192.168.100.31	3478 (STUN)	STUN Forwarding
	WAN	TCP/UDP	*	*	192.168.10.222/24	5349	192.168.100.31	5349	STUN Forwarding

Figure 76: All Firewall NAT Rules

8.4 ISP Port Forwarding

This section will just be descriptive, as each ISP provides different levels and mechanisms for NAT and port forwarding. You may want to, Google NAT configuring for your ISP, or simply call your ISP's tech support line directly. Similarly, each registrar provides different mechanisms for adding global DNS entries (A records) for domains that you manage. It is another student exercise to figure out how to forward with your ISP and registrar.

Essentially, we need to do two things

- Add Global DNS entries for the Hubs servers and services to point to the public IP address of the ISP Router, and
- Add NAT rules to the ISP Router to redirect the traffic to the firewall.

8.4.1 Adding to the Global DNS

When we created the 4 entries in the firewall for DHCP assignment of addresses for the Hubs Servers, the firewall automatically added A records for them to the internal DNS server.

What we need to do now is add public DNS A records for these servers that point to our public IP address of the ISP Router (the one returned by the call to https://ipinfo.io/ip.

Typical DNS A record creation is done via a registrar's customer portal, using the following characteristics;

- **Type**: A Record
- **Host**: <hostname>
- **Value**: <IP Address>
- **TTL**: Automatic

The <hostname> is not the FQDN - it is the name preprended to the domain name. The IP address is the value returned when public users lookup the <hostname>.<domain name>.

In our example, we need 4 A records;

Hubs

- **Type**: A Record
- **Host**: hubs
- **Value**: 129.222.167.85
- **TTL**: Automatic

assets.hubs

- **Type**: A Record
- **Host**: assets.hubs

- **Value**: 129.222.167.85
- **TTL**: Automatic

stream.hubs

- **Type**: A Record
- **Host**: stream.hubs
- **Value**: 129.222.167.85
- **TTL**: Automatic

cors.hubs

- **Type**: A Record
- **Host**: cors.hubs
- **Value**: 129.222.167.85
- **TTL**: Automatic

This will create the four entries mapping the server FQDNs to the public IP address of the ISP Router. For example, stream.hubs.senseicto.com is mapped to 129.222.167.85.

8.4.2 Non-DNS Alternative

Ok. There is a way to get around using a registrar and registering a domain. You can have each public client workstation add to the above mappings to their individual workstation hosts files. On linux, for example, it is located at /etc/hosts. So an example would look like

```
127.0.0.1        localhost
129.222.167.85   hubs.senseicto.com
129.222.167.85   stream.hubs.senseicto.com
129.222.167.85   assets.hubs.senseicto.com
129.222.167.85   cors.hubs.senseicto.com
```

This way, we are bypassing the public DNS by having every client user map the names to the public IP address themselves. The drawbacks, however, include;

- All public internet users would need to add these to their respective hosts file
- When (not if!) your public IP address changes, all public users would have to make corresponding changes to their hosts files.
- If you decide to change your domain name, all public users would have to make corresponding changes to their hosts files.

This is the sneaky bypass I mentioned at the start in order to avoid paying for registration of a domain. It is messy, complicated, and it works.

8.4.3 Configuring NAT on the ISP Router

Your ISP Router, if it allows configuration, can likely be managed either via an app on your mobile phone, or via a management portal hosted on the ISP router itself.

For example, on a WiFi router from my ISP, I use an app on my mobile phone.

- I navigate to the WiFi management icon, then select the “View WiFi equipment” option.
- From here, I can select the “Advanced settings”, giving me access to a variety of configuration options, including a DMZ, DNS Server settings, LAN and WAN settings, as well as the key one of interest - the Port forwarding.

In my case, I can setup a port forward rule similar to what we did in the firewall above.

- It starts with the device that I should forward to. In this case, the cloud server should already be connected to the ISP router on the WAN interface.
- I can then select “Manual Setup”, as my router has predefined rules for a variety of game port forwarding definitions.
- I can then select the protocol - in my case TCP/UDP.
- I can then select either an individual port number of a port range.

Since we have pointed all of the Hubs FQDNs to the ISP Router public IP address, we can just focus on the ports that we want to forward to the firewall. If we are not hosting anything else, then we can likely set this to a wildcard value (e.g. *) to forward all inbound network traffic to the firewall, and let it sort everything as per the rules we defined for it.

We can also specify rules for specific ports in place of a single wildcard, in cases where the ISP Router doesn’t allow that kind of port forwarding, or we want to use the ISP Router as a request filter. We already know that the traffic that we will be forwarding is on the following ports and protocols

- 80 / TCP
- 443 / TCP
- 4443 / TCP/UDP
- 3478 / TCP/UDP
- 5349 / TCP/UDP

Go ahead and create a port forwarding rule for each of these ports in your router.

When complete, you may need to reboot your ISP router for the rules to take effect.

Once complete, you can test your configuration with a mobile phone. Simply turn off the WiFi access so your phone is only using Mobile data, and then open a browser on the phone to your server, in this example it is https://hubs.senseicto.com. You should see the welcome screen of your server.

9 Wrapping Up

That’s it!

If you are reading this, I hope you enjoyed the journey and have successfully set up your own private metaverse!

I would love to hear about how it went for you. You can reach me on Discord in the Hubs channels.

Figure 77: The Conference Room

Thanks for reading!

Terry