System Admin

- Configure FTP
- Configure Postfix
- Configuring Multi-boot Filesystems
- Crontab
- Server Hostname
- Swap Allocation
- Synchronizing Time Using NTP
- Systemd Services
- Unattended Upgrades

Configure FTP

You can use `scp` instead, taking advantage of the added security and already-configured users on your system. It works a lot like `ssh`

Copy a file from a remote host to your local host -

```
scp -i ~/.ssh/some_key -P 22 username@123.123.123.12:/home/username/test .
```

If you still need or want FTP, you can follow the steps below to configure the FTP server and then connect with Filezilla.

Installing Very Secure FTP Daemon

I am using an Ubuntu 19.04 server in this guide, depending on your system your steps may vary slightly.

Assuming you have nothing installed, run `sudo apt-get update && sudo apt install vsftpd` to install vsftpd (Very Secure FTP Daemon). Navigate to the home directory of the user you wish to enable FTP access, and run the following.

```
# Login as your sudo user
sudo su USER

# Create FTP Directory
mkdir /home/USER/ftp
sudo chown nobody:nogroup /home/USER/ftp
sudo chmod a-w /home/USER/ftp
```

Create User FTP Directories

Create a directory where files can be uploaded, you can name this directory whatever you want. Give this directory permissions so you can upload files to it via FTP clients like FileZilla.

```
mkdir /home/USER/ftp/files
sudo chown USER:USER /home/USER/ftp/files
sudo chmod 777 /home/USER/ftp/files
```

Configure vsftpd Settings

If you have a firewall enabled, be sure you open the TCP ports 20, 21, 990, and 40000-50000 before you continue.

Add the following to `/etc/vsftpd.conf`

```
# FTP Initial Configuration Options
pasv_min_port=40000
pasv_max_port=50000
user_sub_token=$USER
local_root=/home/$USER/ftp
pasv_min_port=40000
pasv_max_port=40000
userlist_enable=YES
userlist_file=/etc/vsftpd.userlist
userlist_deny=NO
pasv_promiscuous=YES
allow_anon_ssl=NO
force_local_data_ssl=YES
force_local_logins_ssl=YES
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
require_ssl_reuse=NO
ssl_ciphers=HIGH
```

Run `echo "USER" | sudo tee -a /etc/vsftpd.userlist && sudo systemctl restart vsftpd` to add your user to the userlist file we configured above and restart the service

`tail -f /var/log/syslog` in another console to see a live feed of service logs when restarting instead of checking the status with `sudo systemctl status`

Change or modify the following values, when editing these files I like to comment out the default value, and create a separate value in an organized list with my custom settings. This is useful later should I want to refer back to the default value I can just search it up in my file, and keeps things organized so when I can pick things back up quickly. You can do this however you see fit. The result of my modified values within `vsftpd.conf` is below.

```
# Values Modified During FTP Setup
chroot_local_user=YES
write_enable=YES
ssl_enable=YES
```

Run `sudo systemctl restart vsftpd` to restart the service and test your connection using Filezilla.

Debugging FTP Connections

If you're having issues with your FTP connection, check on the service with the following commands
`sudo systemctl -l status vsftpd` `sudo tail -f /var/log/vsftpd.log`

To test FTP connections via commandline, run the following
`ftp -p IPADDRESS`

You cannot connect to FTP via commandline using this method if you have enabled SSL/TLS because your connection will *not* be encrypted under TLS. Use Filezilla or another encrypted connection method instead.

Notes

Here's a working config file, with some comments on some extra settings I found on the manpages for vsftpd.conf

```
# *Example config file /etc/vsftpd.conf.bak
# *Don't forget to backup your default /etc/vsftpd.conf.bak
#
# Custom FTP configuration for basic server configuration
#
# These settings should be refined for security
# Firewall should be used and reflect the settings in this file
# For more security, use keys and disable password authentication
# +Restrict FTP access to a list of approved IP's with distributed keys



# FTP Custom Configuration Options

# Set chroot user options
chroot_local_user=YES
user_sub_token=$USER

# Set Directory FTP Will Default Into
local_root=/home/$USER/ftp
write_enable=YES
# If you can't write with Write_enable=YES, check directory permissions
# Create .../ftp/files and chmod 777 .../ftp/files
```

```
# Passive FTP Connection Settings
pasv_promiscuous=YES
pasv_min_port=40000
pasv_max_port=50000


# userlist_enable=YES tells vsftpd to read /etc/vsftpd.userlist
# /etc/vsftpd.userlist should contain one user per line
userlist_enable=YES
userlist_file=/etc/vsftpd.userlist
# Sets the userlist to be a whitelist or a blacklist
# userlist_deny=YES will deny FTP for any user on the list
userlist_deny=NO
# Enable logs for failed FTP connections due to userlist errors
# userlist_log=YES


# Enable dual logs fot vsftpd in /var/log/
# log/xferlog - stardard parsable log
# log/vsftpd.log - vsftpd formatted logs
dual_log_enable=YES


# This option specifies the location of the RSA certificate to use for SSL
# encrypted connections.
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
# Other Settings For SSL
ssl_enable=YES
allow_anon_ssl=NO
force_local_data_ssl=YES
force_local_logins_ssl=YES
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
require_ssl_reuse=NO
ssl_ciphers=HIGH


# Default vsftpd.conf Values
```

```
# READ THIS: This example file is NOT an exhaustive list of vsftpd options.
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.

# The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.

secure_chroot_dir=/var/run/vsftpd/empty
pam_service_name=vsftpd
connect_from_port_20=YES
use_localtime=YES
dirmessage_enable=YES
local_enable=YES
anonymous_enable=NO
listen_ipv6=YES
listen=NO
```

Modifying the values below during setup of TLS encryption caused vsftpd to crash on startup..

These values were obtained following this tutorial. Just noting this in case I missed something here, so I can revisit it later.

```
# Working values, establishes TLS connection via Filezilla FTP
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem

# Modified values from generating ssl cert that are crashing vsftpd
# rsa_cert_file=/etc/ssl/private/vsftpd.pem
# rsa_private_key_file=/etc/ssl/private/vsftpd.pem
```

Configure Postfix

> Postfix is a Mail Transfer Agent (MTA) that can act as an SMTP server or client to send or receive email. There are many reasons why you would want to configure Postfix to send email using Google Apps and Gmail. One reason is to avoid getting your mail flagged as spam if your current server's IP has been added to a blacklist.
>
> Linode Postfix Tutorial

Install postfix and mailutils -

```
sudo apt install postfix mailutils
```

Create Google App Token

When attempting to send mail from a new host, you may encounter errors with Google blocking or filtering your mail as spam. To prevent this, simply create a GMail account you wish to send the mail under, Activate 2FA on the new account, then Generate App Tokens to distribute to the hosts / apps you wish to send mail on your behalf. See below for further instructions once you have a GMail account created, and have generated an app password / token.

Postfix App Token Authentication

Once you have the app token, we'll need to add it to `/etc/postfix/sasl/sasl_passwd` - If this file doesn't already exist, create it and include the following lines, modified with your information

```
sudo echo "[smtp.gmail.com]:587 username@gmail.com:password" > /etc/postfix/sasl/sasl_passwd;
```

Instead of using the password you usually input when logging into the GMail account, add the app token generated after enabling 2FA following the links in the first step above. Below, we notify postfix that we've made these changes by running `sudo postmap /etc/postfix/sasl/sasl_passwd`. This will create a sasl_passwd.db file in the `/etc/postfix/sasl` directory.

Run postmap, and restrict access to our new file containing this password

```
sudo postmap /etc/postfix/sasl/sasl_passwd;
sudo chown root:root /etc/postfix/sasl/sasl_passwd /etc/postfix/sasl/sasl_passwd.db;
```

```
sudo chmod 600 /etc/postfix/sasl/sasl_passwd /etc/postfix/sasl/sasl_passwd.db;
```

Configure Relay Server

Configure postfix to relay mail through GMail's server by making the below changes to `/etc/postfix/main.cf` -

```
# Change / modify this line..
relayhost = [smtp.gmail.com]:587

# Add these lines...
# Enable SASL authentication
smtp_sasl_auth_enable = yes
# Disallow methods that allow anonymous authentication
smtp_sasl_security_options = noanonymous
# Location of sasl_passwd
smtp_sasl_password_maps = hash:/etc/postfix/sasl/sasl_passwd
# Enable STARTTLS encryption
smtp_tls_security_level = encrypt
# Location of CA certificates
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
```

Send Mail

That's it! Now restart postfix with `sudo systemctl restart postfix` and test sending mail using any of the commands below -

```
echo "This email confirms that Postfix is working" | mail -s "Testing Posfix" emailuser@example.com
```

or..

```
sendmail emailaddress@gmail.com
FROM: admin@sub.domain.com
SUBJECT: Hi
Body test text
.
```

Mailer Daemon

To change the email the system sends security alerts to, modify the `/etc/aliases` file to use your email address for the `root` field below. If this isn't already in the file, add it, and run `sudo newaliases` to update the system with the new information.

```
# See man 5 aliases for format
postmaster:    root
root: someone@somedomain.com
```

Now to test that his works correctly, attempt to sudo somewhere on the system where you'll be required to enter a password, and botch it - all three times. You'll get an email from your server warning you of the security event! Missing a password on an attempt to sudo is a security event

Configuring Multi-boot Filesystems

When installing a fresh Linux Distribution, you might want to dual-boot, or even multi-boot, into different desktop environments. There are some pretty specific requirements we'll need to setup manually for our new partitions though, see below for details on the different partitions needed to setup an open ended multi-boot system alongside windows. This configuration will prompt for selection of OS on boot, and will allow for nearly any number of distributions to be tested alongside each other. These instructions vary slightly based on your specific scenario, so be sure to read and understand the need for each setting below.

Installation Media

When installing *any* operating system, we first need to create our installation media. Sometimes these are distributed as Installation CDs, but a simple USB can be turned into the same thing very easily.

Depending on your system, see the sections below.

Choosing a Distribution

Not sure what distribution to use, or searching for a legit ISO?

Distrowatch is your friend. They provide rankings, comment boards, forums, and (usually) working links to ISO downloads.

Linux ISO Tools

To burn an image using `umount` and `dd` on Linux, run the commands below.

```
lsblk
```

the command above will list all block storage devices connected to your system, find your device in the list, and take note of the name assigned to it within the `/dev/` directory. Usually, this name is `sd<?><?>` or `sd<?>` for the primary partition. Usually, we would want to write to the primary on the USB to ensure that we boot from our ISO.

```
sudo dd if=/home/user/Downloads/inputfile.iso of=/dev/sd<?><?> && sync
```

Windows ISO Image Writer

Universal USB Installer handles almost every scenario for most if not all distros. Head over to UUI's Download Page, grab the tool and see that your settings are adjusted for your needs.

Your Drive letters, distro release, and persistent file size may be different depending on the requirements you have for your media. To format a drive and prepare for writing -

Formatting Media - UUI

To create USB installation media -

USB Installation Media - UUI

To create a persistant USB to boot wherever you like, adjust the slider on the bottom of the widows labeled `Step 4:` to your desired size.

> When using this tool to create a persistent USB device, there is a temporary directory created in your C:\Users\shaun\AppData\Local\Temp\ directory! If there is not enough space on your drive, the process will not terminate itself, but it will not be able to complete. The temporary files tend to be named nsf9D50.tmp or similar, and will take up equal or slightly more space than is being written to your USB. So, if you create a persistent USB with 10GBs of storage ontop of writing a 2GB .iso, you can expect to need ~12GB on your C:\ drive in order for the process to complete successfully. Once completed, the temp files should be automatically deleted. If the process gets hung due to insufficient space on your drive, this may not happen and you may need to check your Temp directory to manually delete the files yourself.

Alternatively, you can check out the tools below if you have issues using Universal USB Installer.

win32diskimager

If you've written to a USB and need to recreate the media for any reason, you'll need to clear the contents on the USB before you can attempt to reformat and reimage the storage device. To do this, ROSA can be used to clear the USB by selecting the device and clicking clear. After doing this

you'll just need to format the drive using windows quick format tool via right click->format->FAT. Then, use win332diskimager to copy the new image to your device.

BIOS Configuration

When configuring a multi-boot system with specific partitions for different distributions, you'll need to enable the following settings within your BIOS -

- Disable Secure Boot
- Disable fast boot
- Set to UEFI mode only within boot options.

If you're unsure how to modify these settings, try running the setting in question through google along with the model of your motherboard. This will hopefully provide some more specific instructions on using the BIOS of your system.

Modifying these settings will allow us to create EFI files within a given EFI partition, created below, where the system defines the boot sequence for multiple operating systems. This allows us to leave our boot sequence open-ended, and easily append EFI system files to our partition / boot options during the installation of a new system. There are, unfortunately, a few discrepancies to how these steps will be performed - unless the system is to be configured *exactly* the same.

USB Boot / Install

Insert your USB installer you created above using ROSA Image Writer or `dd` command above, and reboot the system. Be sure to pay attention and press the required key to enter the BIOS during boot. For me, the key was delete or F2. Once in the BIOS, navigate to your boot sequence / options and there should be a list of connected storage devices, including all HDD, SSD, USBs, etc. Find your USB installer in the list and select it, this will boot into the installer for your distribution. The installer is usually found on the desktop as an executable application. These installers are usually usable systems, but be aware that there will be no persistent data between reboots until the installation is completed.

When selecting your installation media to boot from within BIOS, be sure to select the media that corresponds with how your system is configured to boot. In this example, the media should start similar to `UEFI: USB ....` If you were not using a UEFI configuration, simply select the same media without the `UEFI:` prefix.

Partitioning

Once booted into the USB created above, you will likely se an installation prompt. When given the choice, select 'custom installation or 'custom partition configuration option, and continue with the guide below.

Bootloader Partition

If you are already using Grub on an existing EFI partition, you won't need to create a new one. Skip this step, but make note of where this partition is, we will need it during installation.

This is the partition where we will create and store new bootloaders during installation of different distributions. You will not directly edit or view this partitions contents, but it is the backbone of the system-selection prompt (grub) that you will receive when booting after completing this configuration. There may be a need to step into this partition if you decide to customize your grub configuration, but we won't get into that here.

Size: 1GB (this is generous)
Type: FAT32 **Location:** Beginning of Space (Volume we are partitioning)
Mount: (Leave empty / blank) **Flags:** boot, efi (also called ESP or EFISYS)

You should always choose to install the bootloader on the same disk the EFI filesystem exists, whether your case required the creation of a new EFI volume or if you are installing alongside a previous one. Failing to do so can could cause issues during installation.

The only exception to this is when initially installing a Linux / Grub Bootloader - you will have to create a new EFI partition for the Grub Bootloader. Grub will pick up the windows partition automatically, but if it doesn't, you can always run `sup grub-update` to search for new EFI partitions or configurations and update your Grub Bootloader appropriately.

Root Partition

This partition will store the Linux system files for your distribution, and unless otherwise partitioned separately, your user's home directory and all of its content. This should be set according to both your distributions total installation size, and if you are not partitioning dedicated space - you should figure in any extra space your user(s) might require for new packages, updates, and applications. Running out of space is a lot worse than having too much, so try to be a little generous here.

Size: Adjust according to installed size of distribution we are using.
Type: exf4 (Logical)
Location: Beginning of Space (Volume we are partitioning)
Mount: `/` **Flags:** root

Swap Partition (Optional)

This is the space your system will use if you run out of memory. If you max out your RAM, this will prevent your system from freezing up. Be cautious of low RAM systems with little or no swap, the downfall to swap space is that once it is used it cannot be reallocated until the system reboots.

Size: 2GB-Preference (Ideally 50-100% of system RAM)
Type: linuxswap (Logical) **Location:** Beginning of Space (Volume we are partitioning)
Mount: (Leave empty / blank)

Home Partition (Optional)

This is optional. I would recommend having a separate storage device (Massive HDDs are getting cheaper..) to mount your home directories in, so if you ever need to reinstall the root directory of your distributions you'll be able to do so without having to worry about backing up or losing data.

I would not advise taking the gamble, you will probably need to reinstall at some point - and it's good insurance to have.

Size: Preference
Type: exf4 (Primary) **Location:** Beginning of Space (Volume we are partitioning)
Mount: `/home`

Installing

Now all we need to do is specify where to install the bootloader. This is easy since we just created that partition above, the EFI Partition. Select the partition from the dropdown and click 'continue' or 'install' at the bottom corner. After this is complete, you'll just need to reboot and witness the grub! From now on, you'll have an option of which system to boot into when starting your PC.

Adding New Systems

During installation of additional systems, we have two requirements, selecting a location for booting the system, and selecting a location for configuring the root filesystem.

Create a new `/root` (and `/home`, if you choose) partition(s), then select the EFI partition we created above for the bootloader install location (For me, this was sdb1 - the first partition of `/dev/sdb`) The basic requirements of both can be seen below

Bootloader

Size: 1GB (this is generous)
Type: FAT32 **Location:** Beginning of Space (Volume we are partitioning)
Mount: (Leave empty / blank) **Flags:** boot, efi

Root

Size: Adjust according to installed size of distribution we are using.
Type: exf4 (Logical) **Location:** Beginning of Space (Volume we are partitioning)
Mount: `/` **Flags:** root

Grub Issues

If you're having issues with **system options not appearing in grub**, be sure to load into a previous system and run `sudo update-grub` - this command will search for new entries in the EFI partition and automatically add them to your grub configuration / system prompt. You can manually step through the EFI partition using the grub command-line to bail yourself out, but this shouldn't be needed as returning to an already configured system and running this command will pick up all new systems for next reboot.

```
grub rescue> set prefix=(hd0,1)/boot/grub
grub rescue> set root=(hd0,1)
grub rescue> insmod normal
grub rescue> normal
grub rescue> insmod linux
grub rescue> linux /boot/vmlinuz-3.13.0-29-generic root=/dev/sda1
grub rescue> initrd /boot/initrd.img-3.13.0-29-generic
grub rescue> boot
```

```
sudo apt install efibootmgr


sudo modprobe efivars


sudo efibootmgr


sudo efibootmgr -b 4 -B
```

```
test -d /sys/firmware/efi && echo UEFI || echo BIOS
sudo blkid
sudo parted -l
cat /proc/cmdline
```

Rescue Grub

Manjaro Install Forum Guide

Crontab

Using `crontab` to schedule tasks for the system to perform is fairly straight forward, once you get familiar with the syntax used within the configuration. Run `crontab -e` to open the file for editing, and modify to your needs using the examples below

Tell crontab where to send email alerts to by adding the following lines to any `crontab`

```
MAILTO=someuser@somedomain.com
MAILFROM=someuser@somedomain.com
```

Alternatively, to silence all emails, just provide no email with `MAILTO=''`.

```
# Schedule our system to run the test.sh script once a day -
0 0 * * * /path/to/test.sh

# Syntax used for time -
* * * * * command to be executed
- - - - -
| | | | |
| | | | ----- Day of week (0 - 7) (Sunday=0 or 7)
| | | ------- Month (1 - 12)
| | --------- Day of month (1 - 31)
| ----------- Hour (0 - 23)
------------- Minute (0 - 59)

# Operators used in scheduling -
(*) : This operator specifies all possible values for a field. For example, an asterisk in the hour time field would be equivalent to every hour or an asterisk in the month field would be equivalent to every month.
(,) : This operator specifies a list of values, for example: “1,5,10,15,20, 25”.
(-) : This operator specifies a range of values, for example: “5-15” days , which is equivalent to typing “5,6,7,8,9,....,13,14,15” using the comma operator.
(/) : This operator specifies a step value, for example: “0-23/” can be used in the hours field to specify command execution every other hour. Steps are also permitted after an asterisk, so if you want to say every two hours, just use */2.
```

So, some example for various schedules -

```
# To run a command once a day at midnight
0 0 * * * /path/to/unixcommand

# To run /path/to/command every five minutes, every day, enter:
5 0 * * * /path/to/command

# To run /path/to/command five minutes after midnight, every day, enter:
5 0 * * * /path/to/command

# Run /path/to/script.sh at 2:15pm on the first of every month, enter:
15 14 1 * * /path/to/script.sh

# Run /scripts/phpscript.php at 10 pm on weekdays, enter:
0 22 * * 1-5 /scripts/phpscript.php

# Run /root/scripts/perl/perlscript.pl at 23 minutes after midnight, 2am, 4am ..., everyday, enter:
23 0-23/2 * * * /root/scripts/perl/perlscript.pl

# Run /path/to/unixcommand at 5 after 4 every Sunday, enter:
5 4 * * sun /path/to/unixcommand
```

Alternative, more readable but less customizable syntax for scheduling common times -

```
@reboot□□Run once, at startup.
@yearly□□Run once a year, “0 0 1 1 *”.
@annually□(same as @yearly)
@monthly□Run once a month, “0 0 1 * *”.
@weekly□□Run once a week, “0 0 * * 0”.
@daily□□Run once a day, “0 0 * * *”.
@midnight□(same as @daily)
@hourly□□Run once an hour, “0 * * * *”.
```

Useful crontab commands

```
# Edit crontab configuration
crontab -e

# List crontab jobs
crontab -l
```

```
# Check status of cron
sudo systemctl status cron
sudo journalctl -u cron
sudo journalctl -u cron | grep backup-script.sh

# Cron logs
cat /var/log/cron
tail -f /var/log/cron
grep "my-script.sh"
tail -f /var/log/cron

# Backup cron
crontab -l > /nas01/backup/cron/users.root.bakup
crontab -u userName -l > /nas01/backup/cron/users.userName.bakup
```

Much of this and more information was found at CyberCiti

Server Hostname

Renaming An Ubuntu Linux Host

Renaming a host on Ubuntu is simple, just need to make some very small changes to both `/etc/hosts` and `/etc/hostname`. See the comments within the files below for more information. Once these changes are made, simply reboot the host and the changes will be applied.

```
# '/etc/hosts' should contain a line similar to the below
127.0.0.1 localhost
# Change it to the following to name the host 'alvin'
127.0.0.1 alvin
```

Similarly, the `/etc/hostname` file will contain *just* the name of the host. So, if we actually wanted to name our host 'alvin', we would change its content to reflect that.

```
alvin
```

> Don't forget to reboot the host to apply the changes. Also, if you are hosting any content or running applications, be sure to save your data and stop the processes if necessary in order to avoid creating issues.

Swap Allocation

Creating swap memory for your host could prevent system or services from crashing when under heavy loads. To do this, run the following commands.

Creating Swap Files

```
# To createa 512MiB swap file -
sudo dd if=/dev/zero of=/swapfile bs=1M count=512 status=progress

# To create a 1GiB swap file -
sudo dd if=/dev/zero of=/swapfile bs=1GB count=1

# To create a 10GiB swap file -
sudo dd if=/dev/zero of=/swapfile bs=1GB count=10
```

Enabling Swap

After creating the swap file of the desired size, in the desired directory, we'll need to set some permissions and prepare our file to be used for swap space -

```
# Set permissions
sudo chmod 600 /swapfile

# Format file to be used for swap allocation
sudo mkswap /swapfile

# Tell our system to mount this file for swap usage
sudo swapon /swapfile
```

Adding Default Swap Entry - fstab

In short, an fstab entry for mounting the swap partition we created above -

```
# <device> <dir> <type> <options> <dump> <fsck>
/swapfile none swap defaults 0 0
```

Add this line to your `/etc/fstab` to mount and use this partition for swap automatically on system reboots.

For more information, see Mounting Default Filesystems or ArchWiki - Fstab

Verifying Swap Configuration

To check available system swap space, run `free -h` to see output similar to the below -

```
root@host:~# free -h
               total        used        free      shared  buff/cache   available
Mem:           983Mi       260Mi        62Mi       0.0Ki       660Mi       560Mi
Swap:          1.0Gi        15Mi       1.0Gi
```

Alternatively, we could run 'sudo swapon --show' to see the below output

```
sudo swapon --show
NAME      TYPE  SIZE  USED PRIO
/swapfile file 1024M 15.8M   -2
```

Swappiness Values

The default swappiness value is set to 60, but to check, change, or verify your system swappiness, see the commands below

```
# Check system swappiness setting
cat /proc/sys/vm/swappiness
60

# Set a new swappiness value
sudo sysctl vm.swappiness=10
# Check the setting was applied
cat /proc/sys/vm/swappiness
10
```

Swappiness Persistance

Upon setting a custom swappiness value and rebooting, your custom configuration will be lost. Edit `/etc/sysctl.conf` to contain the line below to ensure this value is kept between system reboots

```
vm.swappiness=10
```

Removing Swap Files

First, turn off the swap file -

```
sudo swapoff -v /swapfile
```

Remove the swap file entry from your `/etc/fstab` if you previously created one. If present, remove the line similar to the below

```
/swapfile swap swap defaults 0 0
```

Last, delete the swap file using `rm` -

```
sudo rm /swapfile
```

Synchronizing Time Using NTP

> Check out NTP-Pool for a list of pools available to different regions.

Configuration

Network Time Protocol (NTP) allows us to easily synchronize our servers with the indicated NTP host. The settings stored in `/etc/systemd/timesyncd.conf` allow us to specify which NTP server we would prefer to sync with, as well as which server(s) to use should our preferred option fail for whatever reason.

```
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.
#
# Entries in this file show the compile time defaults.
# You can change settings by editing this file.
# Defaults can be restored by simply deleting this file.
#
# See timesyncd.conf(5) for details.

[Time]
#NTP=
#FallbackNTP=ntp.ubuntu.com
#RootDistanceMaxSec=5
#PollIntervalMinSec=32
#PollIntervalMaxSec=2048
```

The configuration above is an example of the default settings, which are commented out since these same default settings are assumed by the Ubuntu system. If you want to change them, just remove the comment and modify their values. Below, we have modified the settings to use various servers based on our preferences.

```
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.
#
# Entries in this file show the compile time defaults.
# You can change settings by editing this file.
# Defaults can be restored by simply deleting this file.
#
# See timesyncd.conf(5) for details.

[Time]
NTP=0.north-america.pool.ntp.org 1.north-america.pool.ntp.org
FallbackNTP=ntp.ubuntu.com 0.arch.pool.ntp.org

#RootDistanceMaxSec=5
#PollIntervalMinSec=32
#PollIntervalMaxSec=2048
```

Above, we tell systemd that we would prefer to connect to NTP servers in the following order

1. 0.north-america.pool.ntp.org
2. 1.north-america.pool.ntp.org
3. ntp.ubuntu.com
4. 0.arch.pool.ntp.org

Synchronization

If your server is for any reason out of sync, this could cause various issues down the line with services or applications. To correct this, simply synchronize with your configured NTP servers by running `sudo timedatectl set-ntp true && timedatectl status` - These commands will synchronize and then print the status of your NTP connection. Be sure to verify the information is correct, and double-check by running `date` within your bash terminal.

Systemd Services

To define our own service with `systemd`, we need to create a `daemon.service` file. This is easily done within a few quick lines using vim, and should only take a few minutes.

First, we need to locate the binary for the command we want to be executed as a service. This is just good to have on-hand when defining a new service. Check where exactly your binary is using `which <command>`, seen below

```
which hexo
/home/hexouser/.nvm/versions/node/v20.9.9/bin/hexo
```

Now we know exactly where the binary that we execute is when we run the `hexo` command, and we will use it within the `hexo.service` file we create below, so be sure to have it handy.

> To create a user service, place the `hexo.service` file within the `$HOME/.config/systemd/hexouser/` directory. This will allow the user to manage the service without sudo by running `systemd --user start name.service`

Create a service file like the one below for hexo by running `sudo vim /etc/systemd/system/hexo.service`. If you are defining a service for something else, just rename this file accordingly.

```
[Unit]
Description=Personal hexo blog service
After=network.target

[Service]
Type=simple
# Another Type: forking
User=hexouser
WorkingDirectory=/home/hexouser/hexosite
ExecStart=/home/hexouser/.nvm/versions/node/v29.9.9/bin/hexo server --cwd /home/hexoroot/hexosite
ExecStop=/bin/kill -TERM $MAINPID
ExecReload=/bin/kill -HUP $MAINPID
Restart=on-failure
# Other restart options: always, on-abort, etc

# The install section is needed to use
```

```
# `systemctl enable` to start on boot
# For a user service that you want to enable
# and start automatically, use `default.target`
# For system level services, use `multi-user.target`
[Install]
WantedBy=multi-user.target
WantedBy=graphical.target
```

When making changes to a service, you need to run `sudo systemctl daemon-reload` between edits to apply your changes before restarting your service. Once the above file is created within `/etc/systemd/system/hexo.service` we can start our hexo blog using systemd by running the usual commands

```
# Start your new service
sudo systemctl start hexo.service
# Enable your service to start automatically on reboot or crashing
sudo systemctl enable hexo.service
# Check on your service
sudo systemctl status hexo.service
```

We can even check on our logs using `journalctl`

```
sudo journalctl -u hexo
journalctl --user-unit hexo
```

Unattended Upgrades

To configure linux hosts to automatically install updates and upgrades, add or edit the following lines in `/etc/apt/apt.conf.d/50unattended-upgrades`. Feel free to change the settings as you see fit.

```
Unattended-Upgrade::Mail "user@example.com";
Unattended-Upgrade::MailOnlyOnError "true";
Unattended-Upgrade::Remove-Unused-Kernel-Packages "true";
Unattended-Upgrade::Remove-Unused-Dependencies "true";
Unattended-Upgrade::Automatic-Reboot "true";
Unattended-Upgrade::Automatic-Reboot-Time "02:38";
```

At the top of `/etc/apt/apt.conf.d/50unattended-upgrades`, you'll notice the block below, be sure to follow my comments and make the changes needed

```
Unattended-Upgrade::Allowed-Origins {
        "${distro_id}:${distro_codename}";
"${distro_id}:${distro_codename}-security";
// Extended Security Maintenance; doesn't necessarily exist for
// every release and this system may not have it installed, but if
// available, the policy for updates is such that unattended-upgrades
// should also install from here by default.
"${distro_id}ESM:${distro_codename}";
"${distro_id}:${distro_codename}-updates"; // <-- Uncomment this line.
//"${distro_id}:${distro_codename}-proposed";
//"${distro_id}:${distro_codename}-backports";
};
```

Add the following lines to `sudo vim /etc/apt/apt.conf.d/20auto-upgrades`.

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Unattended-Upgrade "1";
# Add these two lines...
APT::Periodic::Download-Upgradeable-Packages "1";
APT::Periodic::AutocleanInterval "7";
```

Test that you can run a dy-run update using unattended-upgrades -

```
sudo unattended-upgrades --dry-run --debug
```

Also, check the logs for unattended-upgrades below

```
less /var/log/unattended-upgrades/unattended-upgrades.log
```