Unsorted

- New Page

New Page

Exit-on-error mode

Put this at the top of your bash script: `set -e`. If a command returns a nonzero status, the shell will exit.

Resources: https://unix.stackexchange.com/questions/97101/how-to-catch-an-error-in-a-linux-bash-script http://linuxcommand.org/lc3_man_pages/seth.html - man page

See files that would be unzipped without unzipping

```
unzip -v $FILE_NAME.zip
```

Unzip to a directory

```
unzip package.zip -d /opt
```

Resource: https://www.cyberciti.biz/faq/linux-howto-unzip-files-in-root-directory/

Color output in less

```
FILE=bla.txt
less -r $FILE
```

YAML templating

`template.yml`:

```
---
bobs:
  password: "${bobs_password}"
  username: "${bobs_username}"
  machines: "${bobs_machines}"
```

Fill in the template and create `final.yml`:

```
export sample="Hello Yaml!"
export bobs_machines="10.2.3.4"
export bobs_username="nats"
export bobs_password="password"
rm -f final.yml temp.yml
( echo "cat <<EOF >final.yml";
  cat template.yml;
  echo "EOF";
) >temp.yml
. temp.yml
cat final.yml
```

Resource: https://www.starkandwayne.com/blog/bashing-your-yaml/

Lint YAML

```
python3 -c 'import yaml, sys; yaml.safe_load(sys.stdin)' < file.yml
```

Lint JSON

```
cat file.json | jq
```

Code Style

Function declarations:

```
my_func() {


}
```

Variable declaration:

```
cool_var='great'
```

Source filenames should be lowercase with underscores to separate words, i.e. `the_best_script.sh`

Resources: https://google.github.io/styleguide/shell.xml##Variable_Names

https://dev.to/puritanic/shellscripting-functions-2696

https://stackoverflow.com/questions/673055/correct-bash-and-shell-script-variable-capitalization

---

$HOME vs ~ in scripts

Opt to use `$HOME` as `~` is expanded by the terminal into the `$HOME` variable. Subsequently `$HOME` is going to be a more robust option.

Resource: https://ryankubik.com/blog/tilde-vs-home

Global variables in functions

If you want to set something as a global variable and use it in a function, do the following:

```
export SOMETHING="variable"


great_function() {
    echo $SOMETHING
}
```

Parse an env file

`export $(egrep -v '^##' .env | xargs)`

The env file you want to parse should look something like this:

```
FOO='bar'
USERNAME='user'
```

Resource: https://gist.github.com/judy2k/7656bfe3b322d669ef75364a46327836

Send message to all users

```
echo "hi" | wall
```

Move file with rsync

```
rsync --partial --progress filename.txt user@ipaddress_or_hostname:~
```

Use SSH and rsync together

```
rsync -azvhe ssh --progress local_dir_to_copy user@192.168.1.2:remote_copy_of_dir
```

Remote to local:

```
rsync -azvhe ssh --progress user@192.168.1.2:~/dir_to_copy_to_local local_copy_of_remote_dir
```

Resource: https://www.tecmint.com/rsync-local-remote-file-synchronization-commands/

Sync two folders

Start with a dry run to ensure that it’s doing what you expect it to do:

```
rsync -rv --append-verify --dry-run folder_to_sync_from/ /home/ubuntu/folder_to_sync_to
```

Once you’ve confirmed it’s working as expected, simply remove the `--dry-run` to execute.

Resource: https://unix.stackexchange.com/questions/487646/copy-changed-new-files-only-to-a-different-directory

Transfer multiple remote directories

This worked well for my MacOS systems:

```
IP=192.168.1.56
DIRS=(/Users/user/dir1 /Users/user/dir2 /Users/user/.dotfiles /Users/user/dir3)

for DIR in "${DIRS[@]}"; do
    echo "Transferring remote directory ${DIR}, please wait."
    rsync -avzh --progress "user@${IP}:${DIR}" .
    echo "Transfer of remote directory ${DIR} complete!"
done
```

Resource: https://unix.stackexchange.com/questions/575156/using-rsync-to-back-up-home-folder-with-same-permissions-recursive

---

Break up VM into multiple 1 GB files

Need to transfer a VM? Try this.

```
split -b 1000m vm.ova vm.ova.split
```

Bring it back together

```
cat vm.ova.splita* > vm.ova
```

Kill netcat after 3 seconds

```
timeout 3 nc google.com 80; echo exit=$?
```

Resource: https://unix.stackexchange.com/questions/96817/how-do-i-get-the-nc-command-to-end-after-2-seconds

---

Test connectivity to service

Netcat:

```
HOSTNAME_OR_IP=system.mydomain.com
PORT=5432
# -z: report connectivity status only
# -w1: wait one second and then give up
nc -z -w1 $HOSTNAME_OR_IP $PORT
```

Telnet:

```
HOSTNAME_OR_IP=system.mydomain.com
PORT=5432
# -e X: set escape character to X
echo X | telnet -e X $HOSTNAME_OR_IP $PORT
```

Resources:

- https://webhostinggeeks.com/howto/how-to-check-the-connection-to-mysql-db-server/
- https://www.redhat.com/sysadmin/telnet-netcat-troubleshooting
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/

Determine if string exists in a file

```
if grep -q SomeString "$FILE"; then
  # SomeString was found
fi
```

Resource: https://stackoverflow.com/questions/11287861/how-to-check-if-a-file-contains-a-specific-string-using-bash

---

Delete files if they match a wildcard

If you have two files, `someFile` and `someFile2`, this will delete both of them:

```
if ls someFile* 1> /dev/null 2>&1; then
  rm someFile*
fi
```

Resource: https://stackoverflow.com/questions/12375722/how-do-i-test-in-one-line-if-command-output-contains-a-certain-string

---

Check if multiple binaries are installed

```
if command -v cowsay 2>/dev/null && ! command -v gshuf 2>/dev/null; then
  # do things if cowsay is installed but gshuf isn't
fi
```

Check if file is missing

```
if [[ ! -f "some/file.txt" ]]; then
  # do things if the file is not there
fi
```

One-liner to check if file exists

```
if [ ! -f /tmp/foo.txt ]; then echo "File not found!"; else echo "file found"; fi
```

Resource: https://unix.stackexchange.com/questions/474244/one-liner-to-check-for-file-exists

Create missing directory

```
[[ -d dir ]] || mkdir dir
```

Resource: https://stackoverflow.com/questions/18622907/only-mkdir-if-it-does-not-exist

---

Number of duplicate lines in a file#

```
FILE='/etc/passwd'
sort "${FILE}" | uniq -c
```

Resource: https://stackoverflow.com/questions/6712437/find-duplicate-lines-in-a-file-and-count-how-many-time-each-line-was-duplicated

---

Run command in another directory as another user

This particular example is used to run a command as root in another directory

```
sudo -u root /bin/bash -c 'pushd "$HOME/.some_directory/"; ls -lart; popd'
```

Resource: https://stackoverflow.com/questions/10382141/temporarily-change-current-working-directory-in-bash-to-run-a-command

Bypass post-install configurations

If you have to install things via apt and run into blue screens that require human interaction, do this to avoid it:

```
sudo DEBIAN_FRONTEND=noninteractive apt install -yq [packagename]
```

Resource: https://serverfault.com/questions/227190/how-do-i-ask-apt-get-to-skip-any-interactive-post-install-configuration-steps

Get count of and list of most frequently used bash commands

```
history | awk '{a[$2]++;next}END{for (i in a){print i " --> " a[i]}}' \
  | sort -nr -k3
```

Reinstall a package in debian-based system

```
sudo apt install --reinstall <package> -y
```

Unable to fetch some archives, maybe run apt-get update or try with –fix-missing

Run this if `apt update --fix-missing` isn't working for you:

```
rm -rf /var/lib/apt/lists/*; apt update
```

Alternatively, you can also try:

```
apt clean; apt update
```

Resource: https://stackoverflow.com/questions/38743951/unable-to-fetch-some-archives-maybe-run-apt-get-update-or-try-with-fix-missin

Save off auth logs for review

Useful to see who has logged into your system and when

```
last -F > /tmp/last
```

https://serverfault.com/questions/375091/getting-login-year-data-with-the-last-command-on-linux

“Incognito mode” for bash

Turn off history for a session.

```
export HISTFILE=
```

If you do want a record of your activity in another file, run this command instead:

```
export HISTFILE=0
```

This will create a file, 0, with your history for that session.

Resources: https://github.com/mubix/post-exploitation/wiki/Linux-Post-Exploitation-Command-List
https://www.maketecheasier.com/linux-command-line-history-incognito/

List NFS mounts

```
mount |grep nfs
```

When all users last logged in

```
lastlog
```

All previously logged in users

```
lastlog |grep -v "Never"
```

Resource: https://www.rebootuser.com/?p=1623

Determine if host is sharing file systems

If there are local mounts, you will be able to see who you're sharing with.

```
showmount -e localhost
```

Hosts with active connections to the system

```
netstat -pan |grep ESTABLISHED
```

Check if file has a certain number of lines

If file has 3 lines, tell us about it:

```
if [[ $(wc -l file) == *3* ]]; then
  echo "There are three lines in file"
fi
```

Resource: https://stackoverflow.com/questions/12375722/how-do-i-test-in-one-line-if-command-output-contains-a-certain-string

If wc provides an inaccurate count

Add a newline to the end of the file with:

```
printf "\n" >> file.txt
```

Resource: https://stackoverflow.com/questions/12616039/wc-command-of-mac-showing-one-less-result

Strace

General Example

```
strace /bin/ls
```

View output of a running process

```
PID=5
strace -p "${PID}" -e write
```

Resource: https://unix.stackexchange.com/questions/58550/how-to-view-the-output-of-a-running-process-in-another-bash-session

Filter on write functions

```
strace -e write /bin/ls
```

Objdump

Used to get the assembly output of a given binary

```
BIN="$(which ls)"
objdump -d "${BIN}" | tee binary_output.txt
```

Get size of all subfolders

```
du -sh *
```

Resource: https://www.macobserver.com/tips/mgg-answers/seeing-folders-size-terminal/##:~:text=From%20the%20Terminal%2C%20type%3A%20du,and%20folders%20with%20their%20sizes

List largest directories and files

```
du -hsx * | sort -rh | head -10
```

Resource: https://www.cyberciti.biz/faq/how-do-i-find-the-largest-filesdirectories-on-a-linuxunixbsd-filesystem/

Kerberos

List tickets in a keytab

```
ktutil -kt file.keytab list
```

Resources: https://kb.iu.edu/d/aumh##list https://community.pivotal.io/s/article/Kerberos-Cheat-Sheet

Create kerberos ticket

```
kinit <username> -k -t /where/to/store/keytab/nameofkeytab.keytab
```

Grep

Not matching

This example will find all jobs that don't contain `com`:

```
launchctl list | grep -v com
```

Multiple not matching

```
grep -v ".git\|.terraform"
```

Resource: https://stackoverflow.com/questions/13610642/using-grep-for-multiple-search-patterns

Regex Example

This will get the version of ruby from rvm: `rvm list | head -n 1 |grep -Po '\-(.*?)\s' | tail -c +2` The output from that command that it is parsing will look something like this: `ruby-2.5.1 [ x86_64 ]`

It will return `2.5.1`.

Resource: https://askubuntu.com/questions/89995/bash-remove-first-and-last-characters-from-a-string

Negative Match

Match Inversion

```
grep -Ev 'pattern1|pattern2|pattern3' file
```

Resource: https://stackoverflow.com/questions/10411616/grep-regex-not-containing-a-string

Egrep Regex and return capture group

This will read a file with a bunch of gmail addresses in it (among other things), and then print the unique email addresses found.

```
cat file.txt | grep gmail.com | \
  egrep -o '\w+@gmail.com' | uniq -u
```

Resource: https://stackoverflow.com/questions/18892670/can-not-extract-the-capture-group-with-neither-sed-nor-grep/18892742

Search for string in files and return file name

```
grep -lrnw "thing to search for" .
```

Resources:

- https://stackoverflow.com/questions/6637882/how-can-i-use-grep-to-show-just-filenames-on-linux
- https://stackoverflow.com/questions/3908156/grep-output-to-show-only-matching-file

Output only found files w/ grep

```
grep -rnlw "stringtofind" .
```

Resource: https://stackoverflow.com/questions/3908156/grep-output-to-show-only-matching-file

Search for string in specific file type

```
grep -i -r 'mystring' --include "*.yml"
```

Resource: https://community.home-assistant.io/t/searching-for-text-in-yaml-and-py-files/181948

Create self-signed cert

```
openssl req \
    -x509 \
    -nodes \
    -newkey rsa:4096 \
    -keyout server.key \
    -out server.crt \
    -days 3650 \
    -subj "/C=US/ST=Oregon/L=Portland/O=Company
Name/OU=Org/CN=www.example.com"Department/CN=*"
```

Resource: https://stackoverflow.com/questions/10175812/how-to-create-a-self-signed-certificate-with-openssl

---

Archivers

Extract command

This will extract several popular formats for you without needing to worry about the syntax for each:

```
extract() {
  if [[ -f "${1}" ]] ; then
      case "${1}" in
          *.tar.bz2)   tar xvjf   "${1}"    ;;
          *.tar.gz)    tar xvzf   "${1}"    ;;
          *.bz2)       bunzip2    "${1}"    ;;
          *.rar)       rar x      "${1}"    ;;
          *.gz)        gunzip     "${1}"    ;;
          *.tar)       tar xvf    "${1}"    ;;
          *.tbz2)      tar xvjf   "${1}"    ;;
          *.tgz)       tar xvzf   "${1}"    ;;
          *.zip)       unzip      "${1}"    ;;
          *.Z)         uncompress "${1}"  ;;
          *.7z)        7z x       "${1}"  ;;
          *)           echo "don't know how to extract ${1}..." ;;
      esac
  else
      echo "${1} is not a valid file!"
  fi
}
```

Extract tar contents to specific directory

```
tar -xf archive.tar -C /target/directory
```

Resource: https://askubuntu.com/questions/45349/how-to-extract-files-to-another-directory-using-tar-command

Create tar.gz with var

```
name="directory_name"
tar -czvf "$name".tar.gz $name
```

Create encrypted zip

```
zip -r unenc.zip original_file
openssl enc -in unenc.zip -aes-256-cbc -e > enc.zip
# enter password when prompted
```

Depending on the situation, you may consider destroying the original unencrypted file:

```
rm unenc.zip
```

Decrypt and unzip encrypted zip

```
openssl enc -in enc.zip -aes-256-cbc -d > unenc.zip
# enter password when prompted
unzip unenc.zip
```

Remove duplicates from a file

```
sort inputFile | uniq -u > outputfile
```

Remove last line from a file

```
head -n -1 foo.txt > temp.txt ; mv temp.txt foo.txt
```

Resource: https://stackoverflow.com/questions/4881930/remove-the-last-line-from-a-file-in-bash

---

Curl

Get status code

```
BLUE="\033[01;34m"
status=$(curl -s -o /dev/null -w "%{http_code}" http://target.com)
echo -e "${BLUE}$status${RESET}"
```

Resource: https://superuser.com/questions/272265/getting-curl-to-output-http-status-code

Follow redirect

```
curl -L http://google.com
```

Resource: https://stackabuse.com/follow-redirects-in-curl/

Show response headers for request

```
curl -i http://google.com
```

Resource: https://stackabuse.com/follow-redirects-in-curl/

Silent mode

Use this when you are leveraging curl in a script:

```
curl -s http://google.com
```

Upload a file

To upload a file to an endpoint at `/upload` with a form field, `fileUpload`, you can do the following:

```
curl http://target.com/upload -F "fileUpload=@test.txt" -vvv
```

Send a POST request with params

```
curl -X POST http://localhost:4999/target \
  -d "param1=value1&param2=value2"
```

Download a file

This will also ensure that if a proxy is in place, it will not be used to connect to target.com.

```
curl http://target.com/puppet-linux \
  -o puppet-linux \
  -vvvv \
  --noproxy target.com
```

Resource: https://stackoverflow.com/questions/800805/how-do-i-make-curl-ignore-the-proxy

Download a file to a certain folder

```
TER_VAR=1.1.9
OS=linux
ARCH=amd64
DL_URL="https://releases.hashicorp.com/terraform/${TER_VER}/terraform_${TER_VER}_${OS}_${ARCH}
.zip"

curl ${DL_URL}" -L --output "${HOME}/Downloads/terraform_${TER_VER}_linux_amd64.zip"
```

`-L` - in case a redirect is found `-o` - output path for download

Resource: https://askubuntu.com/questions/285976/download-zip-file-with-curl-command

Use proxy

Useful to proxy requests through something like Burp Suite.

```
curl -X GET http://google.com --proxy http://localhost:8080
```

Resource: https://forum.portswigger.net/thread/how-do-i-get-burpsuite-to-intercept-my-curl-x-get-requests-that-i-am-launching-from-command-line-bfb630dd

3000 millisecond timeout

```
curl -m 3 http://www.google.com
```

Resource: https://ec.haxx.se/usingcurl-timeouts.html

Check if a string is a timestamp

```
date -r "${STR}"
```

Example:

```
date -r 1530279830
```

Resource: https://medium.com/@amalmurali47/h1-702-ctf-web-challenge-write-up-53de31b2ddce

---

Kill processes

This particular example will kill all `python` and `go` processes:

```
pkill python go
```

Resource: https://www.techwalla.com/articles/how-to-kill-all-python-processes-in-ubuntu

---

Useful alternative to man pages

Explain Shell

Display Web Content in Terminal

```
curl http://www.cyberciti.biz/
lynx -dump www.cyberciti.biz
wget -O - http://www.cyberciti.biz
```

Resource: https://www.cyberciti.biz/faq/unix-linux-get-the-contents-of-a-webpage-in-a-terminal/

Open file, grep it, send output to command

This particular example will search for instances of the ENC string and send them to `eyaml decrypt -s`:

```
cat file.txt | grep ENC | xargs --null eyaml decrypt -s
```

Delete to end of line in terminal

Ctrl+k

Delete to beginning of line in terminal

Ctrl+u

Resource: https://askubuntu.com/questions/269046/bash-delete-from-cursor-till-end-of-line-with-a-keyboard-shortcut

Generate a self-signed cert

```
openssl req -new -newkey rsa:4096 -x509 -sha256 -days 365 \
  -nodes -out key.crt -keyout key.key
```

Run command repeatedly

Option #1

```
CMD='date'
while true; do
  "${CMD}"; sleep 2; \
done
```

Option #2

```
CMD='date'
watch "${CMD}"
```

Resource: https://www.howtoforge.com/linux-watch-command/

Sleep with timer

Set MIN to the desired number of minutes you want to `sleep` for:

```
MIN=1
for i in $(seq $(($MIN*60)) -1 1); do
  echo -n "$i, "; sleep 1; \
done
```

Resource: https://www.commandlinefu.com/commands/view/5886/countdown-clock

---

Base64 encode with no word wrap

Useful for long strings that you want to base64 encode.

```
echo "thing" | base64 -w 0
```

Resource: https://conf.splunk.com/files/2019/recordings/SEC2286.mp4

Base64 decode string

```
echo 'stringtodecode' | base64 -d
```

Resource: https://www.base64decode.net/linux-base64-decode

Test if variable is set

```
if [[ -z "${1}" ]]; then
    echo -e "No input provided, exiting.\n"
    exit 1
else
  echo "input is ${1}"
fi
```

Resource: https://stackoverflow.com/questions/3601515/how-to-check-if-a-variable-is-set-in-bash

---

Check if multiple parameters are set

```
if [[ $## -ne 4 ]]; then
   echo 'Required vars missing'
   return 1
fi
```

Resource: https://www.unix.com/shell-programming-and-scripting/268257-how-do-i-check-if-all-parameters-set-bash.html

---

Add ssh fingerprint to known_hosts

Useful if you need this functionality in a script.

```
IP=192.168.1.6
ssh-keyscan -H "${IP}" >> "${HOME}/.ssh/known_hosts"
```

Resource: https://www.techrepublic.com/article/how-to-easily-add-an-ssh-fingerprint-to-your-knownhosts-file-in-linux/

Working with input

Check if input is an ip address

```
if [[ -z $1 ]]; then
  echo "Usage: $0 <ip address>"
  exit
else
  if [[ $1 =~ ^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+$ ]]; then
    echo "Valid IP input, doing things with $1"
  else
    echo "Invalid IP input"
    exit
  fi
fi
```

Resource: https://stackoverflow.com/questions/13777387/check-for-ip-validity

Turn into function

```
check_input() {
  if [[ -z $1 ]]; then
    echo "Usage: $0 <ip address>"
    exit
  else
    if [[ $1 =~ ^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+$ ]]; then
      echo "Valid IP input, doing things with $1"
    else
      echo "Invalid IP input"
      exit
    fi
  fi
}
```

```
# Pass input from $1 to function
check_input $1
```

Resource: https://unix.stackexchange.com/questions/298706/how-to-pass-parameters-to-function-in-a-bash-script

Check input against multiple cases

```
if [[ "${1}" != "puppet" ]] && \
   [[ "${1}" != "chef" ]] || \
   [[ -z "${1}" ]]; then
  echo "Usage: $0 <CM target>"
  exit
fi
```

Resource: https://unix.stackexchange.com/questions/67898/using-the-not-equal-operator-for-string-comparison

Validate script is run as root

```
if [[ "${EUID}" -ne 0 ]]; then
  echo "Please run as root"
  exit
fi
```

Resource: https://stackoverflow.com/questions/18215973/how-to-check-if-running-as-root-in-a-bash-script/18216122##18216122

Multiline if with multiple conditions

```
if [[ -f "${CONTROL_NODE_FILES}/authorized_keys" ]] || \
   [[ -f "${MANAGED_NODE_FILES}/authorized_keys" ]] || \
   [[ -f "${MANAGED_NODE_FILES}/${KEY_NAME}" ]] || \
   [[ -f "${MANAGED_NODE_FILES}/${KEY_NAME}.pub" ]] || \
   [[ -f "${CONTROL_NODE_FILES}/${KEY_NAME}" ]] || \
```

```
    [[ -f "${CONTROL_NODE_FILES}/${KEY_NAME}.pub" ]]; then
         echo -e \
           "${YELLOW}
           Found existing SSH key pair, skipping the key pair generation step.
           ${RESET}"
fi
```

UFW

Show open ports

```
ufw status
```

Allow port

Open the firewall for port 46666:

```
ufw allow 46666/tcp
```

Allow service

This will work for OpenSSH:

```
ufw allow 'OpenSSH'
```

This is for Apache:

```
ufw allow 'Apache Full'
```

Disable UFW

```
ufw disable
```

Enable UFW

```
ufw enable
```

Resource: https://linuxize.com/post/how-to-setup-a-firewall-with-ufw-on-ubuntu-18-04/

Keep NC listener open

This will ensure that netcat doesn't hang when an incoming connection is established.

```
PORT_TO_LISTEN_ON=80
nc -lvk "${PORT_TO_LISTEN_ON}"
```

Resource: https://unix.stackexchange.com/questions/423407/how-can-i-keep-netcat-connection-open

Create random string

```
head /dev/urandom | tr -dc A-Za-z0-9 | head -c 13 ; echo ''
```

Example usage:

```
wget https://github.com/artyuum/Simple-PHP-Web-Shell/blob/master/index.php \
   -O "/var/www/html/$(head /dev/urandom | tr -dc A-Za-z0-9 | head -c 13 ; echo '').php"
```

Resource: https://unix.stackexchange.com/questions/230673/how-to-generate-a-random-string

Create random number

```
r1=$((1 + $RANDOM % 10))
echo $r1
```

Resource: https://stackoverflow.com/questions/1194882/how-to-generate-random-number-in-bash/1195035

Remove newline from end of string

```
tr -d '\n'
```

Resource: https://stackoverflow.com/questions/12524308/bash-strip-trailing-linebreak-from-output

Create three directories in tmp

```
mkdir /tmp/{d1,d2,d3}
```

Test cron job

```
# Add this to the user's crontab (crontab -e):
* * * * *   /usr/bin/env > /home/username/tmp/cron-env
# If root user:
* * * * * /usr/bin/env > /root/tmp/cron-env
# Create run-as-cron.sh
echo -e '##!/bin/bash\n/usr/bin/env -i $(cat /home/username/tmp/cron-env) "$@"' \
  > run-as-cron.sh
# Run the script to test it
run-as-cron /the/problematic/script --with arguments --and parameters
```

Resource: https://unix.stackexchange.com/questions/42715/how-can-i-make-cron-run-a-job-right-now-for-testing-debugging-without-changing

Create daily cron job

This will create a daily cronjob that restarts a systemd job on the system every day at 3:05 am.

```
create_daily_service_restart() {
  cron_job_loc='/etc/cron.daily/restart_service.sh'
  # Create and set permissions for cron job
  echo "$(which systemctl) restart service" > "${cron_job_loc}"
  chmod +x "${cron_job_loc}"
```

```
}
```

Resources:

- [when the job runs]https://www.cyberciti.biz/faq/linux-when-does-cron-daily-weekly-monthly-run/)
- original idea
- how to create a cronjob with a script

Enable logging to /var/log/cron

```
# Enable cron-specific log
sed -i 's/##cron/cron/' /etc/rsyslog.d/50-default.conf
```

Crontabs for another user

You need to be root to do this.

List crontabs for a user:

```
crontab -u $USERNAME -l
```

You can also edit a crontab by using `-e` in place of `-l`.

Copy file to location with random name

```
# This will remove dashes, newlines and spaces
name=$(date | md5sum | tr -d '-' | tr -d '\n' | tr -d ' ')
if cp -R orig "$name"; then
  echo "Created $name"
fi
```

Resources: https://www.howtogeek.com/howto/30184/10-ways-to-generate-a-random-password-from-the-command-line/ https://stackoverflow.com/questions/15801599/shell-script-copy-directory-passed-as-variable-to-another-directory-also-as-var

Copy multiple files to folder

```
cp {file.txt,file2.txt,file3.txt} dst_folder
```

Copy multiple folders to folder

```
cp -r {dir1,dir2,dir3} dst_folder
```

Show routes

```
netstat -nrl
```

Create route

This is useful if you’re having issues with VPN and something like Burp Suite.

```
sudo route add <target system> <local IP>
```

Netstat (without actually running it)

```
awk 'function hextodec(str,ret,n,i,k,c){
    ret = 0
    n = length(str)
    for (i = 1; i <= n; i++) {
        c = tolower(substr(str, i, 1))
        k = index("123456789abcdef", c)
        ret = ret * 16 + k
    }
    return ret
}
```

```
function getIP(str,ret){
    ret=hextodec(substr(str,index(str,":")-2,2));
    for (i=5; i>0; i-=2) {
        ret = ret"."hextodec(substr(str,i,2))
    }
    ret = ret":"hextodec(substr(str,index(str,":")+1,4))
    return ret
}
NR > 1 {{if(NR==2)print "Local - Remote";local=getIP($2);remote=getIP($3)}\
  {print local" - "remote}}' /proc/net/tcp
```

Resource: https://staaldraad.github.io/2017/12/20/netstat-without-netstat/

Show routing rules to reach a destination

```
ip route get <target>
```

For example, this will show the route that your system will take to get to `8.8.8.8` (Google's DNS Server):

```
ip route get 8.8.8.8
```

Note that you can also run this command on a mac after you install the `iproute2mac` package via: `brew install iproute2mac`

Resource: https://systemoverlord.com/2020/03/22/security-101-virtual-private-networks-vpns.html

---

AWK

Remove leading and trailing whitespace

```
trimmed_string=$(echo "no_trimmed_string" | xargs)
```

Another alternative:

```
awk '{$1=$1;print}'
```

Resource: https://unix.stackexchange.com/questions/102008/how-do-i-trim-leading-and-trailing-whitespace-from-each-line-of-some-output

Print everything but first column

This assumes you have text coming from before the `|`.

```
| awk '{$1=""; print $0}'
```

Resource: https://stackoverflow.com/questions/2961635/using-awk-to-print-all-columns-from-the-nth-to-the-last/2961994

Split on / and print first column

```
awk -F '/' '{print $0}'
```

Use in an alias

This particular example will get the IPv6 address for `eth0`. The trick is remembering to escape the `$2`:

```
alias getIP="ifconfig eth0 |grep inet6 |grep global | head -n1 | awk '{print \$2}'"
```

Resource: https://stackoverflow.com/questions/20111063/bash-alias-command-with-both-single-and-double-quotes

Get string after hyphen

```
echo "foo-123" | awk -F- '{print $NF}'
# Will output 123
```

Resource: https://unix.stackexchange.com/questions/193482/get-last-part-of-string-after-hyphen

---

Get size of current directory

```
du -hsx
```

Find

Find files with a specific string and open in VSCode

```
find . -iname '*.hcl' -exec grep -rnw "prod" {} \; -maxdepth 2 \
  | awk -F : '{print $1}' | xargs code
```

Find and lint all YAML files

```
find . -type f -iname '*.yaml' -o -iname '*.yml' -o -iname '*.yml.tpl' \
  | grep -v "submodule_folder\|.terraform" | xargs -I {} cat {} \
  | python3 -c 'import yaml, sys; yaml.safe_load(sys.stdin)'
```

The only issue with this solution is it won't list the files that it finds. If you would like this information as well, do the following instead:

```
files=$(find . -type f -iname '*.yaml' -o -iname '*.yml' -o -iname '*.yml.tpl' \
  | grep -v "submodule_folder\|.terraform"); \
  for file in $files; do \
    echo "Testing ${file}" && cat ${file} \
    | python3 -c 'import yaml, sys; yaml.safe_load(sys.stdin)'; done
```

Find and lint all JSON files

```
find . -type f -iname "*.json" -type f \
  | grep -v "submodule_folder\|.terraform" \
  | xargs -I{} jq . "{}" > /dev/null
```

If you want the names of each finding, you can adopt the work done for linting YAML above.

We also opt to use `> /dev/null` so that we only get files with errors.

Resources:

- run jq on a file without pipe
- only print errors

List all file extensions in CWD

```
find . -type f | perl -ne 'print $1 if m/\.([^.\/]+)$/' | sort -u
```

Resource: https://stackoverflow.com/questions/1842254/how-can-i-find-all-of-the-distinct-file-extensions-in-a-folder-hierarchy

Grep output of find and save to file

```
find . -iname "thing to find" -exec grep -rnw "term to search" {} \; | tee filename
```

As an example, we can search for all python files recursively from the current directory and then grep for the word “git” in those files:

```
find . -iname "*.py" -exec grep -rnw "git" {} \; | tee git
```

Resource: https://unix.stackexchange.com/questions/131535/recursive-grep-vs-find-type-f-exec-grep-which-is-more-efficient-faster

Search for content in tar.gz files

One way:

```
find . -iname "*.tar.gz" -exec zgrep "STRING" {} \;
```

Another way:

```
find . -name '*.tar.gz' -print0 | xargs -0 zgrep "STRING"
```

Resource: https://unix.stackexchange.com/questions/187742/how-do-i-grep-recursively-through-gz-files

Find and delete all hidden directories matching string

```
TARGET_DIR='.terragrunt-cache'
find . -iname "${TARGET_DIR}" -exec rm -rf {} +
```

Resources:

- https://unix.stackexchange.com/questions/187742/how-do-i-grep-recursively-through-gz-files
- https://www.cyberciti.biz/faq/unix-linux-centos-ubuntu-find-hidden-files-recursively/

Find and replace all instances of a string

This will find all go files recursively from the current directory and replace all instances of “toreplace” with “replaced”.

Linux:

```
find . -iname "*.go" | xargs sed -i 's/toreplace/replaced/g'
```

OS X with gnu-sed (installed via `brew install gnu-sed`):

```
find . -iname "*.go" | xargs gsed -i 's/toreplace/replaced/g'
```

Resource: https://stackoverflow.com/questions/1585170/how-to-find-and-replace-all-occurrences-of-a-string-recursively-in-a-directory-t

Find all bin files and create single file with strings output

```
find . -iname "*.bin" -exec strings "{}" \; | tee combined_strings_output.txt
```

Resource: https://superuser.com/questions/566198/linux-command-find-files-and-run-command-on-them

Find directories with a certain name

```
find . -type d -name "thingtosearchfor*" -print 2>/dev/null
```

Resource: https://askubuntu.com/questions/153144/how-can-i-recursively-search-for-directory-names-with-a-particular-string-where

Find a specific file

```
find /etc -name "passwd"
```

Find all bash_histories and send their contents to a file

```
find . -iname ".bash_history" -exec cat {} \; | tee ~/bash_histories.txt
```

Resource: https://stackoverflow.com/questions/864316/how-to-pipe-list-of-files-returned-by-find-command-to-cat-to-view-all-the-files

List all directories the current user has access to

```
find . -maxdepth 1 -type d -perm -u=rx
```

Run ls -lart on all dirs current user can access

This will also display the folder at the top of the output because we've used `+`.

```
find . -maxdepth 1 -type d -perm -u=rx -exec ls -lart {} + 2>/dev/null
```

Find all bash_history files the current user can read

```
find . -maxdepth 1 -type d -perm -u=rx -print0 \
  | xargs -0 -I{} find '{}' -readable -iname '.bash_history' 2>/dev/null
```

Find all .ssh directories that the current user has access to

Clean:

```
find . -maxdepth 1 -type d -perm -u=rx -print0 \
  | xargs -0 -I{} find '{}' -readable -type d -iname '.ssh' 2>/dev/null
```

Not quite as clean:

```
find . -maxdepth 1 -type d -perm -u=rx \
  | xargs ls -lart 2>/dev/null |grep .ssh | grep "^.r..r..r"
```

Breaking this down:

`find . -maxdepth 1 -type d -perm -u=rx` - find all directories the current user has access to

`| xargs ls -lart 2>/dev/null` - send the output of the find command to

`ls -lart` and suppress the Permission denied messages

`|grep .ssh | grep "^.r..r..r"` - output all files with ssh in the name with read permissions for all users

Show all occurrences of users running ssh

```
find . -name .bash_history -print -exec grep ssh {} \;
```

Find all .go files recursively

```
find . -name "*.go"
```

Find bins with SUID or SGID set

Added output to `/tmp/out.txt` and backgrounded in the event you're on a crappy shell.

```
for i in `locate -r "bin$"`; do
  find $i \( -perm -4000 -o -perm -2000 \) -type f 2>/dev/null; done \
  | tee /tmp/out.txt &
```

Resource: https://www.tecmint.com/how-to-find-files-with-suid-and-sgid-permissions-in-linux/

Search for secrets in config.xml files recursively

In this example, we are searching for patterns in config.xml files matching `<secret`, `<password`, or `<credential`:

```
find . -name "config.xml" -exec grep -iE "<secret|<password|<credential" {} + \
  | uniq -u
```

Find all csv files and delete them

```
find . -name "*.csv" -exec rm {} +
```

Delete all json files recursively

```
find . -name "*.json" -exec rm {} \;
```

Resource: https://www.cyberciti.biz/faq/linux-unix-how-to-find-and-remove-files/

Show all executable files

```
find . -maxdepth 1 -perm -111 -type f
```

Resource: https://stackoverflow.com/questions/7812324/finding-executable-files-using-ls-and-grep

Find world writeable directories

```
find /\(-perm -o w -perm -o x\) -type d 2>/dev/null
```

Resource: https://delta.navisec.io/privilege-escalation/

Remove any files in a directory

```
find "/path/to/dir" -type f -exec rm /path/to/dir/* {} \;
```

Resource: https://www.cyberciti.biz/faq/linux-unix-shell-check-if-directory-empty/

Find files not matching a pattern name

This will list all files in the current directory that do NOT have `.template.`, `.settings.` in their name.

```
find . -type f -not -name "*.template.*" -not -name "*.settings.*"
```

Resource: https://alvinalexander.com/linux-unix/linux-find-files-not-matching-filename-pattern-dont-match/

Look for unique ip addresses with find output

```
find . -type f -not -name "*.template.*" -not -name "*.settings.*" \
  | xargs grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" | sort -u
```

Resources:

- find ip addresses with grep
- getting unique ip addresses

Find files with multiple extensions

This particular example can be useful when looking for yaml files:

```
find . -name '*.yaml' -or -name '*.yml'
```

Resource: https://askubuntu.com/questions/1170231/how-to-use-multiple-criteria-for-find

Find all .js files and tar them

```
find . -iname '*.js' -print0 | xargs -0 tar -cf /tmp/js.tar
```

Resource: https://stackoverflow.com/questions/47493109/how-to-find-specific-file-types-and-tar-them

Find all .class and .jar files and tar them

```
mkdir /tmp/java_files && \
  find . -iname '*.jar' -or -name '*.class' -exec cp {} /tmp/java_files \;
tar -czvf /tmp/java_files.tar.gz /tmp/java_files
```

Resource:

- how to tar a directory

Decompile all .class and .jar files in a directory

```
wget https://www.benf.org/other/cfr/cfr-0.151.jar
for file in java_files/*; do
  java -Xmx2048m -jar cfr-0.151.jar $file \
  --outputdir output --caseinsensitivefs true; done
```

Resource:

- run command on all files in a directory

Find all js files and cp into a folder

```
find . -iname "*.js" -exec cp {} ./all_js \;
```

Find all yaml files

```
find . -iname '*.yaml' -o -iname "*.yml" -type f
```

`-o`: OR in `find`

Resources:

- https://unix.stackexchange.com/questions/102191/find-with-multiple-name-and-exec-executes-only-the-last-matches-of-nam
- explains the “OR” in find

Recursively search files for string

This will identify all files that have “letsencrypt” in them:

```
find . -type f -exec grep -l "letsencrypt" {} +
```

Resource: https://www.cyberciti.biz/faq/howto-recursively-search-all-files-for-words/

Exclude directories found with find

```
find . -type f -name "*_peaks.bed" ! -path "./tmp/*" ! -path "./scripts/*"
```

Resource: https://stackoverflow.com/questions/14132210/use-find-command-but-exclude-files-in-two-directories

Using +

The `+` appends each selected file name to the end of the command.

Resource: https://stackoverflow.com/questions/6085156/using-semicolon-vs-plus-with-exec-in-find

Determine if file has been modified within the last n minutes

```
minutes=2
find . -mmin "-${minutes}" -type f -print
```

Resource: https://stackoverflow.com/questions/28337961/find-out-if-file-has-been-modified-within-the-last-2-minutes

Find files that match multiple patterns

```
find Documents \( -name "*.py" -o -name "*.html" \)
```

Resource: https://stackoverflow.com/questions/1133698/using-find-to-locate-files-that-match-one-of-multiple-patterns

Great find tutorial

TomNomNom breaks down the find command and makes it very approachable with this tweet: https://twitter.com/TomNomNom/status/1269263730992439296

Add terminal contents to a file

This is very useful if you have a long command that you want to modify quickly.

Type the following to open a file: CTRL + x CTRL + e

Resource: https://stackoverflow.com/questions/657130/fastest-ways-to-move-the-cursor-on-a-terminal-command-line

Get second column of file

```
cat output.csv | awk '{print $2}'
```

Make offline copy of a site

```
wget --mirror           \
     --convert-links    \
     --html-extension   \
     --wait=2           \
     -o log             \
     http://howisoldmybusiness.com
```

Resource: https://alvinalexander.com/linux-unix/how-to-make-offline-mirror-copy-website-with-wget/

Download entirety of a site

```
wget --random-wait -r -p -e robots=off -U mozilla http://www.example.com
```

Resource: https://stackoverflow.com/questions/11124292/why-does-wget-only-download-the-index-html-for-some-websites

Sed

Insert text after matched string

This will add `retries=2` after `[ssh_connection]` in ansible.cfg:

```
sed -i '/\[ssh_connection\]/a retries=2' ansible.cfg
```

Resource: https://unix.stackexchange.com/questions/121161/how-to-insert-text-after-a-certain-string-in-a-file

Remove last line of a file

```
sed -i '$ d' foo.txt
```

Resource: https://stackoverflow.com/questions/4881930/remove-the-last-line-from-a-file-in-bash

Remove lines from file

This particular example will remove lines 5-10 and 12 from `file.txt`:

```
sed -e '5,10d;12d' file.txt
```

Resource: https://stackoverflow.com/questions/2112469/delete-specific-line-numbers-from-a-text-file-using-sed

Find and replace line in file

BSD:

```
sed -i '' 's/<orig pattern>/<modified>/' file.txt
```

GNU:

```
sed -i 's/<orig pattern>/<modified>/' file.txt
```

BSD and GNU:

```
sed -i".orig" 's/<orig pattern>/<modified>/' file.txt
```

Note: This last one will create a `.orig` file that you'll probably want to delete later

Resources: https://stackoverflow.com/questions/7573368/in-place-edits-with-sed-on-os-x https://www.javaer101.com/en/article/13833496.html

Find and replace line assigned to variable in file

BSD:

```
new_str="so new"
sed -i '' "s/staticlineinfile/${new_str}/g" "file/to/change.conf"
```

GNU:

```
new_str="so new"
sed -i "s/staticlineinfile/${new_str}/g" "file/to/change.conf"
```

BSD and GNU:

```
new_str="so new"
sed -i".orig" "s/staticlineinfile/${new_str}/g" "file/to/change.conf"
```

Resources: https://stackoverflow.com/questions/4247068/sed-command-with-i-option-failing-on-mac-but-works-on-linux https://askubuntu.com/questions/76808/how-do-i-use-variables-in-a-sed-command

Add prefix to command output

```
yourcommand1 | sed  's/^/[prefix1]/'
```

For example, this will get all ip addresses associated with running containers, and add http:// to the front of each of them:

```
docker ps | awk '{ print $13 }' | grep -o -P "(.*:\d+)" | sed  's/^/http:\/\//'
```

Resource: https://stackoverflow.com/questions/42482477/add-prefix-in-bash-command-output

Remove nth line of a file

Remove the nth line of a file:

```
sed 'Nd' file
```

You have to use `-i` to modify the file.

For example, to remove the second line from somefile.txt and overwrite it:

```
sed -i '2d' somefile.txt
```

Resource: http://www.aodba.com/use-sed-command-delete-lines-linux/

Change shell with sed

This will change the shell from `/usr/sbin/nologin` to `/bin/bash` for the www-data user:

```
sed -i '/www-data/s/\/usr\/sbin\/nologin/\/bin\/bash/g' /etc/passwd
```

Resource: https://superuser.com/questions/558394/replacing-bin-bash-with-bin-false-in-etc-passwd-file

Get nth line of a file

```
sed 'NUMq;d' file
```

For example, to get the second line from somefile.txt:

```
sed '2q;d' somefile.txt
```

Resource: https://stackoverflow.com/questions/6022384/bash-tool-to-get-nth-line-from-a-file

Get range of lines from a file

This example will get lines 747-749 from `file`:

```
sed -n -e '747,749p' file
```

Resource: https://unix.stackexchange.com/questions/138398/how-to-get-lines-10-to-100-from-a-200-line-file-into-a-new-file

Insert text at beginning of every line of a file

This example inserts bla at the beginning of every line of filename:

```
sed 's/^/bla/' filename
```

Resource: https://stackoverflow.com/questions/4080526/using-sed-to-insert-text-at-the-beginning-of-each-line

Add comma to end of every line of a file but the last

```
sed '$!s/$/,/' file
```

Resource: https://stackoverflow.com/questions/35021524/how-can-i-add-a-comma-at-the-end-of-every-line-except-the-last-line/35021663

Avoid escaping slashes

If you have slashes in your replacement string, you can use `|` instead of `/` to avoid escaping those slashes. For example:

```
sed "s|regex|replace|" file.txt
```

Resource: https://stackoverflow.com/questions/40714970/escaping-forward-slashes-in-sed-command/40715028

Capture control group

This will grab the number (51265) from the string:

```
echo 'BLA: 51265 - BLA - BLA' | sed -r "s/.*([0-9]{5}).*/\1/"
```

Resource: https://stackoverflow.com/questions/11650940/sed-how-to-do-regex-groups-using-sed

Replace one character with two

```
echo "asdlksad ~ adlkajsd ~ 12345" | sed 's/~/~\n/g'
```

Resource: https://stackoverflow.com/questions/18365482/how-to-replace-one-character-with-two-characters-using-tr

Replace specific lines of file

Print output without changing file:

```
sed '9,11s/\/\///g' config.js
```

Make the changes to the file: BSD:

```
sed -i '' '9,11s/\/\///g' config.js
```

Resource: https://stackoverflow.com/questions/16202900/using-sed-between-specific-lines-only

---

Merge two folders

```
rsync -avh --progress /path/to/source/ /path/to/destination
```

For example, if you have a recipes folder in `/Volumes/stuff/recipes` that you want to sync with your local recipes folder:

```
rsync -avh --progress ~/recipes /Volumes/stuff
```

Resources: https://unix.stackexchange.com/questions/149965/how-to-copy-merge-two-directories https://unix.stackexchange.com/questions/149965/how-to-copy-merge-two-directories##:~:text=If%20you%20want%20to%20move,are%20on%20the%20same%20filesystem.

Conditional based on output of a command

```
if echo $(command_to_run) | grep -q "thing to find in output of command"
    then echo "Yay, we found things!"
    else "We didn't find the thing we wanted to find!"; exit 1
fi
```

Resource: https://stackoverflow.com/questions/3943854/grep-in-if-statement

Setuid

```
chmod u+s <filename>
```

Resource: https://www.liquidweb.com/kb/how-do-i-set-up-setuid-setgid-and-sticky-bits-on-linux/

Remove files without an extension

In this case, we'll `rm` every file that doesn't end with the `apk` file extension:

```
find . -type f ! -name "*.apk" -exec rm {} \;
```

Resource: https://stackoverflow.com/questions/41935440/how-to-remove-files-without-certain-extension

Assign output to variable

```
pid=$(adb shell ps -A | grep processname | awk -F ' ' '{print $2}')
```

Resource: https://www.cyberciti.biz/faq/unix-linux-bsd-appleosx-bash-assign-variable-command-output/

Get first two characters of a string

Use `head -c 2`

Example that will get the first two characters of the apk found in `/tmp`:

```
find /tmp -name "*.apk*" | head -c 2
```

Resource: https://stackoverflow.com/questions/1405611/how-to-extract-the-first-two-characters-of-a-string-in-shell-scripting

Break long command down

This is useful if you have a long command and want to be able to lay it out to analyze. Assign the parameter to a variable and add `\` for line breaks.

For example:

```
command="https://google.com/endpoint \
?fields=field1,field2 \
field3&field4=abc&field5=1234xyz"
curl $command
```

Resource: https://stackoverflow.com/questions/32341091/multiline-curl-command

Get today's date

```
today=`date '+DATE:%m%d%y' | cut -d: -f2`
```

Get the time

```
time=$(date '+TIME:%H%M%S' | cut -d: -f2)
```

Get the year

```
year=$(date | cut -d' ' -f 6)
```

Get the month and day

```
month_day=$(date '+DATE:%m%d' | cut -d: -f 2)
```

Create folder with date and time

```
mkdir `echo $(date +"%c") | tr -s ' ' | tr ' ' '_'`
```

Resource: https://stackoverflow.com/questions/13799789/expansion-of-variables-inside-single-quotes-in-a-command-in-bash

Check if today is a weekend

```
if [[ $(date +%u) -gt 5 ]]; then
    echo 'Sorry, you cannot run this program on the weekend.'
    exit
fi
```

Resource: https://stackoverflow.com/questions/3490032/how-to-check-if-today-is-a-weekend-in-bash

---

Kill process by path

```
pkill -9 -f /path/to/app

# Example:
pkill -9 -f /tmp/Linenum.sh
```

Kill all processes that are a parent of a zombie

```
# Don't do this.  Incredibly risky sledge hammer!
kill $(ps -A -ostat,ppid | awk '/[zZ]/ && !a[$2]++ {print $2}')
```

Systemd

Run script on startup

1. Create systemd service file called `<service name>.service` and set the path to where you plan to have the script on disk:

   ```
   [Unit]
   After=network.service

   [Service]
   ExecStart=/usr/local/bin/<name of script>.sh

   [Install]
   WantedBy=default.target
   ```

2. Create the script to run on startup (call it whatever you want).
3. Move the files into place and set the permissions:

```
sudo mv <name of script>.sh /usr/local/bin/<name of script>.sh
sudo mv <service name>.service /etc/systemd/system/<service name>.service
sudo chmod 744 /usr/local/bin/<name of script>.sh
sudo chmod 664 /etc/systemd/system/<service name>.service
```

4. Enable the service:

```
sudo systemctl daemon-reload
sudo systemctl enable <service name>.service
```

Resource: https://linuxconfig.org/how-to-run-script-on-startup-on-ubuntu-20-04-focal-fossa-server-desktop

Show all enabled services

```
systemctl list-unit-files | grep enabled
```

Resource: https://askubuntu.com/questions/795226/how-to-list-all-enabled-services-from-systemctl

Get status of a service on the system

```
systemctl $SERVICE_NAME status
```

Get status of all user services on the system

```
XDG_RUNTIME_DIR="/run/user/$UID" systemctl --user status
```

Resource: https://unix.stackexchange.com/questions/615917/failed-to-get-d-bus-connection-connection-refused

Get status of a specific user service

```
XDG_RUNTIME_DIR="/run/user/$UID" systemctl --user status file.service
```

Set env var in systemd service

Create the service like so: `/etc/systemd/system/myservice.service`:

```
[Service]
Environment="MY_ENV=abc"
Environment="ANOTHER_ENV=zyx"
ExecStart=/opt/myservice -env=true
```

Run these commands to update and restart the service:

```
# Reload units
systemctl daemon-reload
# Start the service
systemctl start myservice
# Confirm everything works as expected
systemctl status myservice
```

Resource: https://serverfault.com/questions/413397/how-to-set-environment-variable-in-systemd-service

Tail -f systemd

```
journalctl --follow _SYSTEMD_UNIT=gunicorn.service
```

Resource: https://thierry.marianne.io/view/0ae42ee4d7e86763a0de1765b8ae4ff6

Show what's listening on a port

```
sudo lsof -i -P -n | grep LISTEN
```

Resource: https://www.cyberciti.biz/faq/unix-linux-check-if-port-is-in-use-command/

Exclude file from zip

This will exclude the `.git` folder, the `README.md` file and the `Makefile`:

```
zip -r output.zip . -x '*.git*' -x README.md -x Makefile
```

Resources:

- excluding files
- make a zip of everything in the current directory

Trim whitespace from a bash var

```
echo "   lol  " | xargs
```

Resource: https://stackoverflow.com/questions/369758/how-to-trim-whitespace-from-a-bash-variable

Get open ports

This is one option that has some nice and concise output:

```
sudo lsof -i -P -n |grep LISTEN
```

Here's another for newer versions of linux with a bit more info:

```
sudo ss -tulwn
```

Resource: https://www.cyberciti.biz/faq/unix-linux-check-if-port-is-in-use-command/

Handy Shortcuts

This will get the last run command:

```
!!
```

So for example, if you run this command:

```
ls -lart
```

and then run this command:

```
!! |grep bash_history
```

You will be running the equivalent of:

```
ls -lart |grep bash_history
```

This will get the second command run in a line of commands:

```
!:1
```

For example, if you run this command:

```
file $(ldd /bin/ls | grep libc.so | cut -d' ' -f3) -L
```

and then type this in:

```
!:1
```

you will get the following command:

```
$(ldd /bin/ls | grep libc.so | cut -d' ' -f3)
```

Resource:

- this will not be available for you unless you purchase the course, sorry

---

Make

Print something

```
@echo Things I want to say!
```

Resource: https://www.gnu.org/software/make/manual/html_node/Echoing.html

Variable declaration examples

Normal variable assignment:

```
BOOL_VALUE := false
@echo $(BOOL_VALUE)
```

Run bash command and assign output to `$(BUILD_BUCKET)`:

```
BUILD_BUCKET := $(shell terraform output build_bucket | tr -d \")
@echo $(BUILD_BUCKET)
```

Resources:

- how to run bash script
- how to assign bash script output to a variable to be used in a Makefile

Run several actions in a row

```
# Create phony target to avoid conflict with a file of the
# same name and to improve performance
.PHONY: all
all: build test

# Create phony target to avoid conflict with a file of the
# same name and to improve performance
.PHONY: build
build:
 go build

# Create phony target to avoid conflict with
# a file of the same name and to improve performance
.PHONY: test
test:
 go test -v
```

To run everything, simply run:

```
make
```

Resources:

- https://www.gnu.org/software/make/manual/html_node/Echoing.html
- running several actions in a row
- explanation of phony targets

Bash command substitution

```
clear_build_bucket:
    - bucket=my-bucket; \
    aws s3api delete-objects --bucket $${bucket} \
      --delete "$$(aws s3api list-object-versions \
      --bucket $${bucket} \
      --query='{Objects: Versions[].{Key:Key,VersionId:VersionId}}')"; \
    aws s3api delete-objects --bucket $${bucket} \
      --delete "$$(aws s3api list-object-versions \
        --bucket $${bucket} --query='{Objects:
DeleteMarkers[].{Key:Key,VersionId:VersionId}}')"
```

Resource:

- running several actions in a row

Use bash for loop

```
JSON_FILES := $(shell find . -type f -iname "*.json")
lint:
  for json in $(JSON_FILES); do echo "JSON file found: $${json}"; done
```

Export env var

This example is part of a `terragrunt` snippet in which the folder is the region associated with the `Makefile`, which is then used for the `AWS_DEFAULT_REGION` value.

```
REGION := $(shell ls | grep us)
export AWS_DEFAULT_REGION ?= $(REGION)
```

Resource: https://github.com/nzoschke/gofaas/blob/master/Makefile

Loop over array

```
SHELL=/bin/bash
TF_FILES = .terraform .terragrunt-cache .terraform.lock.hcl

.PHONY: destroy
destroy:
 - terragrunt run-all destroy --terragrunt-non-interactive -lock=false --terragrunt-source-update
 # Clear terragrunt and terraform files
 for file in $(TF_FILES); do find . -name $$file -prune -exec rm -rf {} \; ; done
```

Resource: https://stackoverflow.com/questions/52282549/for-each-on-target-of-makefile-variable

Run command and continue even w/ error

In this example, the find commands to clear the terragrunt cache will run even if the `run-all destroy` command has an error:

```
.PHONY: destroy
destroy:
    - terragrunt run-all destroy --terragrunt-non-interactive
    # Clear terragrunt cache
    find . -type d -name ".terragrunt-cache" -prune -exec rm -rf {} \;
    find . -type d -name ".terraform" -prune -exec rm -rf {} \;
```

Set default value

Observe $MYVAR in `Makefile`:

```
MYVAR ?= default

.PHONY: all

all:
    echo $(MYVAR)
```

Then:

```
$ make
echo default
default
$ export MYVAR=notDefault
$ make
echo notDefault
notDefault
```

Resource: https://stackoverflow.com/questions/53369903/use-environment-variable-if-set-otherwise-use-default-value-in-makefile

If then else

```
.PHONY: all
all:
ifeq ($(REGION),)
 echo "REGION not set, exiting!"
 exit 1
else
all: init plan apply
endif
```

Resources:

- original idea
- Provided great example

Run Makefile from another directory

```
make -C /path/to/makefile
```

For example, let's say there's a Makefile in `${FOLDER}`:

```
FOLDER=/opt/scripts/installer_of_things
make -C "${FOLDER}"
```

Resource: https://superuser.com/questions/370575/how-to-run-make-file-from-any-directory

---

Get unique values from bash array

```
echo "${ids[@]}" | tr ' ' '\n' | sort -u | tr '\n' ' '
```

Resource: https://stackoverflow.com/questions/13648410/how-can-i-get-unique-values-from-an-array-in-bash

Get public IP

Using `curl`:

```
# Will get your IPv4 address:
curl ifconfig.me
```

Using `wget`:

```
# Will get your IPv6 address (if applicable):
wget -O - ifconfig.co 2>/dev/null
```

Show hidden files with tree

```
tree -a
```

Expanded view of running processes

Run this to get a sense of processes and their parents

```
ps axjf
```

Download file to specific dir with wget

```
install_dir='/home/ubuntu/install_here'
mkdir $install_dir
wget <file to download> -P $install_dir
```

Resource: https://www.tecmint.com/wget-download-file-to-specific-directory/

Get newest file matching a string

This will find all files that start with `vncpass` in `/home/ubuntu/vnc` and get the most recent one:

```
ls -t /home/ubuntu/vnc/vncpass* | head -1
```

Resource: https://stackoverflow.com/questions/5885934/bash-function-to-find-newest-file-matching-pattern

Get all files except '.' and '../'

```
ls -A
```

Resource: https://stackoverflow.com/questions/22407480/command-to-list-all-files-except-dot-and-dot-dot

Convert cert to single line

```
awk 'NF {sub(/\r/, ""); printf "%s\\n",$0;}'  ca.pem
```

Resource: https://serverfault.com/questions/466683/can-an-ssl-certificate-be-on-a-single-line-in-a-file-no-line-breaks

Get certificate portion of pem file

```
openssl x509 -in cert.pem -outform der | base64 -w 64
```

For a full example, run this to generate a self-signed cert:

```
openssl req \
    -x509 \
    -nodes \
    -newkey rsa:2048 \
    -keyout server.key \
    -out server.crt \
    -days 3650 \
    -subj "/C=GB/ST=London/L=London/O=Global Security/OU=IT Department/CN=yourhostname.com"
```

and then run:

```
openssl x509 -in server.crt -outform der | base64 -w 64
```

Resource: https://stackoverflow.com/questions/40366409/get-just-the-certificate-portion-from-an-openssl-pem-file

Curl with client certificate

This will help you to avoid using the `-k` option if you're using a self signed certificate (which you probably shouldn't unless you're doing some testing or have another good reason).

Get the client certificate (`client.pem`):

```
echo quit | openssl s_client -showcerts -servername yourhostname.com \
  -connect yourhostname.com:443 > cacert.pem
```

Use it in your curl command:

```
curl --cacert cacert.pem -X POST https://yoursite.com/endpoint -d 'data=hi'
```

Resources:

- get the cacert and use it
- https://serverfault.com/questions/845766/* cert generation

Prevent curl output from being cut off

If curl output is cut off, put the variable in double quotes.

```
GOOGLE=$(curl -s -i https://google.com)
echo $GOOGLE ## fail
echo "$GOOGLE" ## success
```

Resource: https://stackoverflow.com/questions/53892741/missing-output-when-storing-curl-output-as-bash-variable

Diff output of two commands

This will diff the output of a find command with a bunch of not statements against a find command that pipes it output to grep, which in turn uses a perl regex to try an find files that match `a bunch of numbers.json`.

```
diff <(find . -type f -not -name "*.template.*" \
  -not -name "*.settings.*") <(find . -type f -name "*.json*" \
  | grep -P '.*?\d+.json')
```

Resources:

- diff
- using grep with find

Use json for POST body

```
curl -k -s -i -X POST --compressed \
    -H "Content-Type: application/json" \
    -d @input.json \
    https://target.com/endpoint
```

Diff two directories

```
diff -qr directory-1 directory-2
```

If you want to exclude criteria from the output, you can modify this example:

```
diff -qr goutils goproject | grep -v -E "(*utils.go|*utils_test.go|.git)"
```

Resource: https://www.tecmint.com/compare-find-difference-between-two-directories-in-linux/

Diff files and dirs - show diff and unique content

```
diff -x '.git' -bur directory-1/ directory-2/ | grep -- '---\|Only'
```

`-x`: exclude `-b`: ignore whitespace `-u`: unified context (3 lines before and after) `-r`: recursive

Resources:

- https://stackoverflow.com/questions/2019857/diff-files-present-in-two-different-directories
- how to use grep with —

Get PID's without ps

```
find /proc -mindepth 2 -maxdepth 2 -name exe -exec ls -lh {} \; 2>/dev/null
```

Resource: https://unix.stackexchange.com/questions/115927/find-the-pids-of-all-threads-of-a-process-without-ps-or-pidof

---

Find PID of process using port

```
sudo ss -lptn 'sport = :80'
```

Resource https://unix.stackexchange.com/questions/106561/finding-the-pid-of-the-process-using-a-specific-port

---

Netstat without netstat

Copy pasta ftw:

```
awk 'function hextodec(str,ret,n,i,k,c){
    ret = 0
    n = length(str)
    for (i = 1; i <= n; i++) {
        c = tolower(substr(str, i, 1))
        k = index("123456789abcdef", c)
        ret = ret * 16 + k
    }
    return ret
}
function getIP(str,ret){
    ret=hextodec(substr(str,index(str,":")-2,2));
    for (i=5; i>0; i-=2) {
        ret = ret"."hextodec(substr(str,i,2))
    }
    ret = ret":"hextodec(substr(str,index(str,":")+1,4))
    return ret
}
NR > 1 {{if(NR==2)print "Local - Remote";local=getIP($2);
remote=getIP($3)}{print local" - "remote}}' /proc/net/tcp
```

Resources: https://staaldraad.github.io/2017/12/20/netstat-without-netstat/

Show lines that match in two files

```
grep -Ff file1.txt file2.txt
```

Resource: https://unix.stackexchange.com/questions/125155/compare-two-files-for-matching-lines-and-store-positive-results

Show files that have part of a string in them

```
grep 'string_to_look_for' . -R 2>/dev/null
```

Resource: https://superuser.com/questions/614526/finding-files-which-contain-a-certain-string-using-find-1-and-grep-1/614538

Cosmetic Color Output

This is an example of how you might consider different colors for output in a bash script:

```
RED="\033[01;31m"      # Issues/Errors
GREEN="\033[01;32m"    # Success
BLUE="\033[01;34m"     # Heading
YELLOW='\033[0;33m'    # Informational
RESET="\033[00m"       # Normal

echo -e "${BLUE}Running apt-get update, please wait...${RESET}"
apt-get update -y
```

Resource: https://github.com/TH3xACE/OFFPORT_KILLER/blob/master/OFFPORT_KILLER.sh

CLI input

Boolean Flag Example

`sample.sh`:

```
FORCE=false

while getopts ':f' option; do
    case "${option}" in
        'f') FORCE=true ;;
    esac
done
shift $(( OPTIND - 1 ))

main() {
    if [[ $FORCE = true ]]; then
        echo "Option f was supplied on the command line"
    else
```

```
        echo "Option f was not supplied on the command line"
    fi
}

main
```

Example Usage:

```
bash sample.sh -f
```

Resources:

- good cli example
- another good cli example
- true/false settings

Get FQDN

```
hostname -f
```

Resource: https://www.tecmint.com/hostname-command-examples-for-linux/##:~:text=To%20view%20the%20name%20of,the%20FQDNs%20of%20the%20machine.&text=To%20display%20the%20alias%20name,%2C%20use%20the%20%2Da%20flag.

Create file with multiple lines

```
# set for the example below
BLA='hi'

cat <<EOT >> greetings.txt
line 1
"${BLA}"
"$(whoami)"
line 4
EOT
```

Resource: https://unix.stackexchange.com/questions/77277/how-to-append-multiple-lines-to-a-file

Run command on every line of a file

This example will resolve a list of hosts to their ip addresses:

```
getent hosts $(<host_list.txt)
dig + short $(<host_list.txt)
```

Resource: https://stackoverflow.com/questions/13939038/how-do-you-run-a-command-for-each-line-of-a-file

Show long line of ps

If there's a process that you can't read, you can run this command to see the whole thing:

```
ps -ef | more
```

Resource: https://unix.stackexchange.com/questions/229541/view-full-commands-in-ps-output

Copy multiple files

```
cp {file1,file2} dest
```

Resource: https://stackoverflow.com/questions/24206349/copy-multiple-files-from-one-directory-to-another-from-linux-shell

Get active network interface

This script will work on Mac OS or Linux:

```
ifconfig | awk '/UP/{print $1}' | grep 'eth0:\|en0:' | sed 's/://'
```

Resources:

- https://www.2daygeek.com/
- gave me the idea about using `awk` to parse the output
- grep OR
- remove the `:` from the output

Check if script is running on a Mac

```
if [[ `uname` == 'Darwin' ]]; then
  echo "I am running on MacOS"
else
  echo "I am running on something."
fi
```

Use placeholder with xargs

If instead of the echo you had a long command that you wanted to pass to sed, this is how you would do it:

```
echo 'REPLACED' | xargs -I {} sed -i "s/REPLACE_ME/{}/" file.txt
```

The `{}` serves as the placeholder for what we're piping in.

Resource: https://www.shogan.co.uk/how-tos/using-placeholder-templates-with-xargs-in-the-pipeline/

Replace IP Address in file

```
pub_ip=$(dig @resolver1.opendns.com A myip.opendns.com +short -4)
sed -i -r "s/(\b[0-9]{1,3}\.){3}[0-9]{1,3}\b/$pub_ip/" file.txt
```

Resource: https://stackoverflow.com/questions/5277156/using-sed-to-search-and-replace-an-ip-address-in-a-file

Find IP addresses in file

This will find invalid IP addresses if those are present:

```
grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" file.txt
```

To match only valid ip addresses:

```
# <!-- markdownlint-disable-next-line -->
grep -oE "(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)" file.txt
```

Resources: https://unix.stackexchange.com/questions/296596/how-to-check-if-any-ip-address-is-present-in-a-file-using-shell-scripting match only valid IP addresses

Check if string matches regex

```
[[ $date =~ ^regex$ ]] && echo "matched" || echo "did not match"
```

Resource: https://stackoverflow.com/questions/21112707/check-if-a-string-matches-a-regex-in-bash-script/21112809

Check if variable is empty

```
if [[ -z "$var" ]]
then
      echo "\$var is empty"
else
      echo "\$var is NOT empty"
fi
```

For example:

```
if [[ -z $(aws s3 ls |grep -i tf-remote-state) ]]; then
echo 'Remote state not configured! Configuring...';
else echo 'Remote state already configured, moving on...'; fi
```

Resource: https://www.cyberciti.biz/faq/unix-linux-bash-script-check-if-variable-is-empty/

Check if env var is set

```
if [[ -z "${DEPLOY_ENV}" ]]; then
  MY_SCRIPT_VARIABLE="Some default value because DEPLOY_ENV is undefined"
else
  MY_SCRIPT_VARIABLE="${DEPLOY_ENV}"
fi
```

Wait for process to finish execution

This particular example will loop until ansible is finished running. If it is not running at all, it will skip the loop entirely.

```
while /usr/bin/pgrep ansible >/dev/null; do
        echo "ansible is running"
        sleep 1
done
echo 'ansible is not running!'
```

Resource: https://stackoverflow.com/questions/38666191/while-loop-in-bash-that-uses-pgrep-to-check-if-service-exists

Regex capture group

```
users=(bob jane clyde)
regex=".*>\s(.*)\s<.*"

for user in "${users[@]}"; do
  line=$(grep "$user favorite food" userdata.txt)
  if [[ $line =~ $regex ]]; then
    echo "$user has a favorite food and it is: ${BASH_REMATCH[1]}"
  fi
done
```

Resource: https://stackoverflow.com/questions/1891797/capturing-groups-from-a-grep-regex

Alias an export

```
alias sroot="export SROOT="\$PWD""
alias drumit="cd \$SROOT/abc/def/drumit"
```

Resource: https://stackoverflow.com/questions/7747673/how-to-alias-an-export-in-bash

Remove an alias

```
unalias gs
```

Resource: https://askubuntu.com/questions/325368/how-do-i-remove-an-alias

Comma separated string to array

```
string="bla1,bla2"
IFS=', ' read -r -a blas <<< "$string"
echo ${blas[0]}
```

Resource: https://stackoverflow.com/questions/10586153/how-to-split-a-string-into-an-array-in-bash

Append value to array

```
ARRAY=()
## Append foo to $ARRAY:
ARRAY+=('foo')
## Append bar to $ARRAY:
ARRAY+=('bar')
```

Resource: https://stackoverflow.com/questions/1951506/add-a-new-element-to-an-array-without-specifying-the-index-in-bash

Pass array as value to function

```
f(){
  b=$1
  shift
  a=("$@")

  for i in "${a[@]}"
  do
    echo $i
  done
  echo $b
}

a=(value1 value2 value3)
b=STANDALONESTRING

f "$b" "${a[@]}"
```

Resource: https://stackoverflow.com/questions/16461656/how-to-pass-array-as-an-argument-to-a-function-in-bash

Debug bash script

Run this command to get debug output from a bash script:

```
bash -x name_of_script.sh
```

Resource: https://tldp.org/LDP/Bash-Beginners-Guide/html/sect_02_03.html

Exit bash script

```
exit 0  ## Exits the script completely.  $? contains 0 (success).
exit 1  ## Exits the script completely.  $? contains 1 (failure).
```

Resource: https://stackoverflow.com/questions/4419952/difference-between-return-and-exit-in-bash-functions

URL Encode a string

This is a lifesaver when working with `curl`.

```
urlencode() {

    old_lc_collate=$LC_COLLATE
    LC_COLLATE=C

    local length="${##1}"
    for (( i = 0; i < length; i++ )); do
        local c="${1:$i:1}"
        case $c in
            [a-zA-Z0-9.~_-]) printf '%s' "$c" ;;
            *) printf '%%%02X' "'$c" ;;
        esac
    done

    LC_COLLATE=$old_lc_collate
}
urlencode "a string yay"
```

Resource: https://gist.github.com/cdown/1163649

Get Public IP address and Geolocation

```
curl -s https://ipapi.co/$(curl -s ifconfig.me)/json | jq
```

Resources: https://ipapi.co/##api https://www.tecmint.com/find-linux-server-geographic-location/

Multiline string to a file

Everything in between the `EOM` delimiters is the string that will go into the file (`trust_policy.json` in this case).

```
cat > trust_policy.json <<- EOM
{
  "Version":"2012-10-17",
  "Statement":[
   {
      "Effect":"Allow",
      "Principal":{
      "Service":"ec2.amazonaws.com"
    },
    "Action":"sts:AssumeRole"
    }
  ]
}
EOM
```

Resource: https://stackoverflow.com/questions/23929235/multi-line-string-with-extra-space-preserved-indentation

---

Set default value for var

`test.sh`:

```
FOO=${1:-bar}
DOG=${2:-yes}

echo ${FOO}
echo ${DOG}
```

If you run it this way, the output will be will be reflected:

```
bash test.sh
```

```
## Output:
bar
yes
```

If you run it with arguments, the output will change:

```
bash test.sh chicken no

## Output:
chicken
no
```

Resource: https://stackoverflow.com/questions/16319720/bash-command-line-arguments-replacing-defaults-for-variables/16836338

View all arguments submitted via CLI

```
echo $@
```

Resource: https://linuxconfig.org/how-do-i-print-all-arguments-submitted-on-a-command-line-from-a-bash-script

Check spelling in files

```
aspell check --mode=markdown --lang=en README.md
```

Resource: https://www.manuel-strehl.de/check_markdown_spelling_with_aspell

Run bash commands in script after chroot

```
chroot /home/mayank/chroot/codebase /bin/bash <<"EOT"
cd /tmp/so
ls -l
```

```
echo $$
EOT
```

Delete files with specified extension from directory

```
DIR=~/Downloads
EXT=jar

ls "${DIR}" | grep "${EXT}" | xargs -I {} rm "${DIR}/{}"
```

Resource: https://stackoverflow.com/questions/51305706/shell-script-that-does-chroot-and-execute-commands-in-chroot/51312156

HEREDOC with bash variables

```
kubectl exec -it -f helm_file.yaml -- chroot /mnt /bin/bash <<EOF
./kubeletmein generate
./kubectl --kubeconfig ./kubeconfig.yaml get pods -n ${TARGET_NAMESPACE}
EOF
```

Resource: https://unix.stackexchange.com/questions/138418/passing-a-variable-to-a-bash-script-that-uses-eof-and-considers-the-variable-a

---

Bind non priv user to a priv port

```
setcap 'cap_net_bind_service=+ep' /path/to/program
```

Resource: https://stackoverflow.com/questions/413807/is-there-a-way-for-non-root-processes-to-bind-to-privileged-ports-on-linux

---

Test Web Socket

```
npm install -g wscat
TARGET="wss://ws-feed.gdax.com"
wscat -c "${TARGET}"
```

Resource: https://stackoverflow.com/questions/47860689/how-to-read-websocket-response-in-linux-shell

---

Convert string to lowercase

A few options:

```
# The strip() at the end removes any trailing newlines
echo "ABC" | python3 -c "print(open(0).read().lower().strip())"
echo "ABC" | sed 's/./\L&/g'
export a="ABC" | echo "${a,,}"
```

Resource: https://stackoverflow.com/questions/2264428/how-to-convert-a-string-to-lower-case-in-bash

String comparison

```
#!/bin/bash

read -p "Enter first string: " VAR1
read -p "Enter second string: " VAR2

if [[ "$VAR1" == "$VAR2" ]]; then
    echo "Strings are equal."
else
    echo "Strings are not equal."
fi
```

Resource: https://linuxize.com/post/how-to-compare-strings-in-bash/

Get architecture string

```
dpkg --print-architecture
```

Resource: https://unix.stackexchange.com/questions/180726/easy-command-line-method-to-determine-specific-arm-architecture-string

Reinstall package in /bin

If you accidentally delete a package in `/bin`, this is how you get it back.

```
MISSING_PACKAGE=coreutils ## change this based on your situation
cd && apt-get download "${MISSING_PACKAGE}"
dpkg -i *.deb
```

Resource: https://askubuntu.com/questions/906674/accidentally-removed-bin-how-do-i-restore-it

Get filename and extension from path

```
filepath='/path/to/file.txt'
extension="${filepath###*.}"
filename="$(basename ${filepath})"
```

Resources: https://stackoverflow.com/questions/965053/extract-filename-and-extension-in-bash
https://www.cyberciti.biz/faq/bash-get-filename-from-given-path-on-linux-or-unix/

Fix centos8 dnf

```
dnf -y --disablerepo '*' --enablerepo=extras swap centos-linux-repos
```

Resource: https://stackoverflow.com/questions/70963985/error-failed-to-download-metadata-for-repo-appstream-cannot-prepare-internal

Print functions in script

```
## Get all functions into newline separated string
fstr="$(declare -F \
    | awk '{print $NF}' \
    | sort \
    | egrep -v '^_')"

## This entry also covers converting a multiline string
## to an array:
SAVEIFS=$IFS   ## Save current IFS (Internal Field Separator)
IFS=$'\n'      ## Change IFS to newline char
funcs=($fstr)  ## split `fstr` into an array by the same name
IFS=$SAVEIFS   ## Restore original IFS

echo "This file has the following functions:"
for f in "${funcs[@]}"; do
  echo "${f}"
done
```

Resources:

- https://unix.stackexchange.com/questions/260627/how-do-you-list-all-functions-and-aliases-in-a-specific-script
- https://stackoverflow.com/questions/24628076/convert-multiline-string-to-array

Upgrade ubuntu install from LTS

Change the following line in `/etc/update-manager/release-upgrades` from:

```
Prompt=lts
```

to:

```
Prompt=normal
```

and run the following to kick the process off:

```
sudo do-release-upgrade
```

Resource: https://www.omgubuntu.co.uk/2020/10/how-to-upgrade-to-ubuntu-20-10-from-ubuntu-20-04

Get substring after delimiter

This example will get everything after the first hyphen:

```
var=one-two-three
cut -d '-' -f2- <<< $var # outputs: two-three
var=four-five
cut -d '-' -f2- <<< $var # outputs: five
```

Resource: https://linuxhint.com/bash-substring-after-character/

Get OS and Arch

```
export OS="$(uname | python3 -c 'print(open(0).read().lower().strip())')"
export ARCH="$(uname -a | awk '{ print $NF }')"
```

Set Static IP on Ubuntu

Open as root `/etc/netplan/50-cloud-init.yaml` (or whatever file is present in `/etc/netplan`). Replace the contents of the file with this yaml after updating the various IP addresses for your needs:

```
---
network:
  ethernets:
    eth0:
      dhcp4: false
      addresses: [192.168.20.124/24]
      routes:
        - to: 0.0.0.0/0
          via: 192.168.20.1
```

```
          metric: 100
      nameservers:
        addresses:
          - 1.1.1.1
          - 8.8.8.8
          - 192.168.20.1
  version: 2
```

Apply it:

```
netplan --debug apply
```

The `--debug` flag provides verbose output - feel free to omit it.

Add the following to `/etc/cloud/cloud.cfg.d/99-disable-network-config.cfg`:

```
network: {config: disabled}
```

Resource: https://linuxconfig.org/how-to-configure-static-ip-address-on-ubuntu-22-04-jammy-jellyfish-desktop-server

---

Get age of system

```
sudo dumpe2fs /dev/sda1 | grep 'Filesystem created:'
```

Resource: https://askubuntu.com/questions/1352/how-can-i-tell-what-date-ubuntu-was-installed