# Linux /Unix / Ubuntu Commands

| | |
|---|---|
| **date** | get date |
| **hostname** | get hostname |
| **cd** | get out of all directory to root directory |
| **vim** | to edit a single file |
| **touch** | create a new file |
| **tz** | get time zone |
| **df** | get disk info +used |
| **ls** | list directory |
| **free** | check free disk space +total used |
| **cd /** | go to root-> / |
| **info** | get info of some command more than the man (manuals) |
| **nano** | open a text editor |
| **df** | used to display the amount of available disk space for file systems on which the invoking user has appropriate read access |
| **du** | smiler to df but gives the info in different form |
| **ping** | check speed /get site access time  using pings |
| **cd ..** | go one step back |
| **rmdir test** | remove directory |
| **mkdir test** | make directory |
| **pwd** | get complete path |
| **locate** | get location with complete path of that file in the directory |
| **cat new.txt** | check the data / line in the file |
| **echo hello, my name is numan >> new.txt** | insert method to append data into the file with line break |
| **passwd** | change password |
| **—help** | get help to execute the command |
| **man** | get manual |
| **printf hello, my name is numan >> new.txt** | insert method to append data into the file no line break |
| **uptime** | get hoe long the system is working and user load |
| **w** | get quick summary of evert user logged into you computer |
| **history** | get history of terminal commands |
| **cp** | copy command to copy directory or file data to another |
| **last** | used to display the list of all the users logged in and out since the file /var/log/wtmp was created |
| **ps** | display current running processes |
| **chmod** | give permission to Owner/user/admin.  RWX (Read Write Execute) |
| **apt-get** | get upgrades + update |
| **Zip** | used to compress one or more files and store them in a new file with .zip extension. |
| **Unzip** | used to uncompress files. |
| **grep** | used to search for a text in the file or folder. |

## Directory Structure Ubuntu /linux

**/:** The directory called “root.” It is the starting point for the file system hierarchy. Note that this is not related to the root, or superuser, account

**/bin:** Binaries and other executable programs

**/etc:** System configuration files.

**/home:** Home directories.

**/opt:** Optional or third party software.

**/tmp:** Temporary space, typically cleared on reboot.

**/usr:** User related programs.

**/var:** Variable data, most notably log files.

# Linux /Unix / Ubuntu Advanced Commands

## FIND:

The Linux Find Command is one of the most important and frequently used command-line utilities in Unix-like operating systems. Find command is used to search and locate the list of files and directories based on conditions you specify for files that match the arguments.

**find . -name filename.txt**
Find all the files whose name is **filename.txt** in a current working directory.

**find /home -name filename.txt**
Find all the files under /home directory with name **filename.txt**

**find /home -iname filename.txt**
Find all the files whose name is filename.txt and contains both capital and small letters in /home directory.

**find / -type d -name filename**
Find all directories whose name is **filename** in / directory.

**find . -type f -name filename.php**
Find all php files whose name is **filename.php** in a current working directory.

**find . -type f -name “*.php”**
Find all php files  in a current working directory.

**find . -type f -perm  0777 -print**
Find all files whose permissions are 777

**find / -type f !perm  0777 -print**
Find all files without permissions 777

**find / -perm  2644**
Find all SGID bit files whose permissions set to 644

**find / -perm  1551**
Find all Sticky bit set files whose permissions are 551

**find / -perm  /u=s**
Find all SUID set files

**find / -perm  /g=s**
Find all SGID set files

**find / -perm  /u=r**
Find all read only files

**find / -perm  /a=x**
Find all executable set files

**find / -type f -perm 0777 -print -exec chmod 644 {} \;**
Find all 777 permissions files and change its permission to 644

**find / -type d -perm 0777 -print -exec chmod 644 {} \;**
Find all 777 permissions directories and change its permission to 644

**find . -type f -name  "filename.txt" -exec rm -f {} \;**
Find a single file  **filename.txt** and remove it

## grep [options] pattern [files]

The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression

**Options Description**

**-c** : This prints only a count of the lines that match a pattern

**-h :** Display the matched lines, but do not display the filenames.

**-i :** Ignores, case for matching

**-l :** Displays list of a filename only.

**-n :** Display the matched lines and their line numbers.

**-v :** This prints out all the lines that do not matches the pattern

**-e exp :** Specifies expression with this option. Can use multiple times.

**-f file :** Takes patterns from file, one per line.

**-E :** Treats pattern as an extended regular expression (ERE)

**-w :** Match whole word

**-o :** Print only the matched parts of a matching line,

with each such part on a separate output line.

## sed OPTIONS... [SCRIPT] [INPUTFILE...]

The SED command in UNIX stands for stream editor and it can perform lot's of functions on files like, searching, find and replace, insertion or deletion. Though most common use of SED commands in UNIX is for substitution or for find and replace. By using SED you can edit files even without opening it, which is much quicker way to find and replace something in file, than first opening that file in VI Editor and then changing it

**sed 's/testing/test/' new.txt**
Replace a word in the file all the occurrence

**sed 's/testing/test/2' new.txt**
Replace a word in the file first two occurrence

**sed 'nd' filename.txt**

To Delete a particular line say n

**sed '$d' filename.txt**

To Delete a last line

**sed 'x,yd' filename.txt**

To Delete line from range x to y

**sed 'nth,$d' filename.txt**

To Delete from nth to last line

**sed '/pattern/d' filename.txt**

To Delete pattern matching line

## <u>TAIL</u>

The tail command is a command-line utility for outputting the last part of files given to it via standard input. It writes results to standard output. By default tail returns the last ten lines of each file that it is given. It may also be used to follow a file in real-time and watch as new lines are written to it.

**-n :** shows number of lines

**-c :** shows number of bytes

**-q:** used to combine files

**-f :** see changes in the file

**tail /usr/share/dict/words /usr/share/dict/french**

Shows multiples files

**ls -t /etc | tail -n 5**

the output of the ls command is piped to tail

## AWK command

Command: **awk options 'selection _criteria {action }' input-file > output-file**

**-f program-file :** Reads the AWK program source from the file program-file, instead of from the first command line argument**.**

**-F fs :** Use fs for the input field separator

**awk '/manager/ {print}' employee.txt**

Print the lines which match with clthe given pattern.

**awk '{print $1,$4}' employee.txt**

Splitting a line into fields

## Built In Variables In Awk

Awk's built-in variables include the field variables—$1, $2, $3, and so on ($0 is the entire line) — that break a line of text into individual words or pieces called fields.

- **NR:** NR command keeps a current count of the number of input records. Remember that records are usually lines. Awk command performs the pattern/action statements once for each record in a file
- **NF:** NF command keeps a count of the number of fields within the current input record.
- **FS:** FS command contains the field separator character which is used to divide fields on the input line. The default is "white space", meaning space and tab characters. FS can be reassigned to another character (in BEGIN) to change the field separator.
- **RS:** RS command stores the current record separator character. Since, by default, an input line is the input record, the default record separator character is a newline.
- **OFS:** OFS command stores the output field separator, which separates the fields when Awk prints them. The default is a blank space. Whenever print has several parameters separated with commas, it will print the value of OFS in between each parameter.
- **ORS:** ORS command stores the output record separator, which separates the output lines when Awk prints them. The default is a newline character. print

automatically outputs the contents of ORS at the end of whatever it is given to print.

## Cut Command

The cut command in UNIX is a command for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by byte position, character and field.

**cut OPTION... [FILE]...**

**cut -b 1,2,3 state.txt**

List without range

**cut -b 1-3,5-7 state.txt**

List with range

**cut -b 1- state.txt**

1- indicate from 1st byte to end byte of a line

**cut -d "delimiter" -f (field number) file.txt**

Find the field name with delimiter and cut it

**cut --complement -d " " -f 1 state.txt**

As the name suggests it complements the output. This option can be used in the combination with other options either with -f or with -c.

**cut -d " " -f 1,2 state.txt --output-delimiter='%'**

By default the output delimiter is the same as input delimiter that we specify in the cut with -d option. To change the output delimiter use the option –output-delimiter=”delimiter”.

## Nano Command / Notepad

Nano is a user-friendly, simple and WYSIWYG(What You See Is What You Get) text editor, which improves the features and user-friendliness of UW Pico text editor. Unlike vim editor or any other command-line editor, it doesn’t have any mode. It has an easy GUI(Graphical User Interface) which allows users to interact directly with the text in spite of switching between the modes as in vim editor.

- **Ctrl+o =======>** To save a file
- **Ctrl+u =======>** used to paste the text
- **Ctrl+k =======>** used to cut
- **Ctrl+w =======>** search a word in a file
- **Ctrl+g =======>** get help
- **Ctrl+r =======>** read file
- **Ctrl+y =======>** previous page
- **Ctrl+c =======>** cur. Pos
- **Ctrl+j =======>** justify
- **Ctrl+w =======>** where is
- **Ctrl+v =======>** next page
- **Ctrl+u =======>** unCut text
- **Ctrl+t =======>** to spell

## VIM Command / Notepad

Vim, the popular text editor comes with a built-in feature to encrypt files with a password. Vim uses algorithms like Blowfish to encrypt files. It is faster and convenient than some other utilities available to do so.

vim has five modes (insert, visual, normal, command-line, select) but important are (insert, visual and normal).

**Insert Mode:**This mode allows you to enter a text into your document. You can enter insert mode by pressing “ **I** ”. This mode only allows you to enter text in your file.

**Command Mode:**This is default mode. If you’re in a different mode and want to go back to command mode, just hit the Escape key.

**Visual mode:**The visual mode permits the user to select the text as you would do with a mouse but using the keyboard instead of the mouse. Useful to copy several lines of text for example. The shortcut is: “**V**".

**Normal Mode:**By default, Vim starts in “normal” mode. Normal mode can be accessed from other modes by pressing Esc or <C-[>. In Normal mode key presses don’t work as one would expect.

**Select Mode:**Select-mode is commonly used in snippet plugins such as vim-snipmate.Mappings can be created that are unique to select-mode so as not to interfere with regular visual mode behaviour.

**Basic Commands of VIM**

- **:q quit the current file (if other tabs are open, they will remain) - if unsaved changes are there, you will get an error message**
- **:qa quit all**
- **:q! quit and ignore unsaved changes**
- **:w save changes**
- **:w filename provide a filename if it is a new file or if you want to save to another file**
- **:w! save changes even if file is read-only, provided user has appropriate permissions**
- **:wa save changes made to all the files opened**
- **:wq save changes and quit**
- **:wq! save changes even if file is read-only and quit**

- **:e refreshes the current buffer**
- **:e filename open file for editing**
- **:e# switch back to previous buffer**
- **:e#1 open first buffer**
- **:e#2 open second buffer and so on**
- **:bn open buffer next**
- **:bp open buffer previous**
- **:split filename open file for editing in new horizontal split screen, Vim adds a highlighted horizontal bar with filename for each file thus opened**
- **:vsplit filename open file for editing in new vertical split screen**
- **If filename is not given, the current one is used**
- **:tabe filename open file for editing in new tab instead of adding to buffers**
- **:tabn to go to next tab**
- **:tabp to go to previous tab**
- **:tabr to go to first tab (r for rewind)**
- **:tabl to go to last tab (l for last)**
- **:tabm 0 move current tab to beginning**
- **:tabm 2 move current tab to 3rd position from left**
- **:tabm move current tab to end**
- **:help [keyword] : perform a search of help documentation for whatever keyword you enter.**
- **:e [file] : Opens a file, where [file] is the name of the file you want opened**
- **Ctrl +f : one page forward.**
- **Ctrl+b : one page back.**
- **Gg : place the cursor start of the file.**
- **G : place the cursor end of the file.**
- **# : where # is the number of lines, this command takes you to the line specified.**
- **Yy : copies a line**
- **yw : copies a word**
- **Y$ : Copies from where your cursor is to the end of a line.**
- **v : Highlight one character at a time using arrow buttons or the h, k, j, l buttons**
- **p : Paste whatever has been copied to the unnamed register**
- **d : Deletes highlighted text**
- **dd : Deletes a line of text**
- **Dw : Deletes a word**

- **x : delete a single character**
- **u : undo the last operation.**
- **Ctrl +r : Redo the last operation.**
- **Ctrl + ws : Split windows horizontally**
- **Ctrl + wv : Split windows vertically**
- **Ctrl +ww : Switch between windows**
- **Ctrl + wq : Quit a window**
- **Ctrl + wh : Moves your cursor to the window to the left**
- **Ctrl + wl : Moves your cursor to the window to the right**
- **Ctrl + wj : Moves your cursor to the window below the one you're in**
- **Ctrl + wk : Moves your cursor to the window above the one you're in**

**Some additional commands**

- **tab sview /path/to/file:** To open a file in read only mode in a new tab,
- **vim -M filename:** opens the file in read only mode.
- **escape :** esc
- **quit :** q
- **write :** w

## Dockers

Docker is a set of platform as a service products that uses OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels.

## Docker's Containers

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an applications code, runtime, system tools, system libraries and settings.

Container images become containers at runtime and in the case of Docker containers images become containers when they run on Docker Engine. Available for both Linux and Windows-based applications, containerized software will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development.

## Purpose of Docker

It is used to automate the deployment of applications inside software containers and the automation of operating system level virtualization on Linux. It's lightweight then Containers and boots-up in seconds.

## Advantages of Docker

- **Docker produces an API for container management**

  Docker produces an API for container management in an image format and a chance to use a remote registry for sharing containers. This scheme serves both developers and system administrators with advantages for instance.

- **Fast application deployment**

Containers carry the minimal runtime requirements of the application, decreasing their size and enabling them to be deployed instantly.

- **Transferable across machines**

  An application and all its dependencies can be grouped into a separate container that is autonomous from the host version of Linux kernel, platform configuration, or deployment type. This container can be assigned to another machine that runs Docker and performed there without adaptability issues.

- **Version control and component retain**

  You can pursue succeeding versions of a container, inspect irregularities or go back to previous versions. Containers reuse segments from the preceding layers, which makes them remarkably light.

- **Sharing**

  You can use a distant repository to share your container with others. Red Hat provides a registry for this purpose, and it is also desirable to configure your own individual repository.

- **Light and minimal overhead**

  Docker images are typically very small, which promotes rapid delivery and reduces the time to deploy new application containers.

## Disadvantages of Docker

- **Containers don’t work at bare-metal rates**

  A bare metal environment is a computer system or network in which a virtual machine is installed directly on hardware rather than within the host operating system.

  Containers utilise resources more efficiently than virtual machines. But containers are however directed to performance overhead due to overlay networking, interfacing within containers and the host system and so on. If you want 100% bare-metal performance, you want to apply bare metal, not containers.

- **The container ecosystem is split**

  But the core Docker platform is open source, some container products don't work with other ones.

- **Data storage is intricate (very complicated )**

  By design, all of the data inside a container leaves forever when it closes down except you save it somewhere else first. There are ways to store data tenaciously in Docker, such as Docker Data Capacities, but this is arguably a test that still has yet to be approached in a seamless manner.

- **Graphical applications do not operate well**

  Docker was created as a solution for deploying server applications that don't need a graphical interface. While there are some creative approaches that one can practice to run a GUI app inside a container, these solutions are solid at best.

- **Few applications do not benefit from Docker Containers**

  In common, the applications that are intended to work as a collection of thoughtful microservices attain the most from containers. Contrarily, Docker's one real benefit is that it can interpret application delivery by giving an easy packaging mechanism.

## Dockers Commands

- docker run hello world – For checking the number of images on your system
- docker images – For searching an image in the Docker Hub
- docker search <image> – For searching the image by its name
- docker run – Runs a command in a new container.

- docker start – Starts one or more stopped containers
- docker stop – Stops one or more running containers
- docker build – Builds an image form a Docker file
- docker pull – Pulls an image or a repository from a registry
- docker push – Pushes an image or a repository to a registry
- docker export – Exports a container's filesystem as a tar archive
- docker exec – Runs a command in a run-time container
- docker search – Searches the Docker Hub for images
- docker attach – Attaches to a running container
- docker commit – Creates a new image from a container's change
- docker ps – to see all images with status and port on which they are running more than images
- docker [OPTIONS] COMMAND

## Parameters

- --config string Location of client config files
- -c, --context string Name of the context to use to connect to the daemon (overrides DOCKER_HOST env var and default context set with "docker context use")
- -D, --debug Enable debug mode
- -H, --host list Daemon socket(s) to connect to
- -l, --log-level string Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
- --tls Use TLS; implied by --tlsverify
- --tlscacert string Trust certs signed only by this CA
- --tlscert string Path to TLS certificate file
- --tlskey string Path to TLS key file
- --tlsverify Use TLS and verify the remote
- -v, --version Print version information and quit

## Dockers Compose

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

## Dockers Compose Commands

- docker-compose [-f <arg>...] [options] [COMMAND] [ARGS...]
- docker-compose -h|--help
- docker-compose start
- docker-compose stop
- docker-compose pause
- docker-compose unpause
- docker-compose ps -list down
- docker-compose up - run
- docker-compose down -remove

## Parameters

| | |
|---|---|
| -f, --file FILE | Specify an alternate compose file |
| | (default: docker-compose.yml) |
| -p, --project-name NAME | Specify an alternate project name |
| | (default: directory name) |
| --verbose | Show more output |

- --log-level LEVEL Set log level (DEBUG, INFO, WARNING, ERROR, CRITICAL)
- --no-ansi Do not print ANSI control characters
- -v, --version Print version and exit
- -H, --host HOST Daemon socket to connect to
- 
- --tls Use TLS; implied by --tlsverify
- --tlscacert CA_PATH Trust certs signed only by this CA
- --tlscert CLIENT_CERT_PATH Path to TLS certificate file
- --tlskey TLS_KEY_PATH Path to TLS key file
- --tlsverify Use TLS and verify the remote
- --skip-hostname-check Don't check the daemon's hostname against the name specified in the client certificate
- --project-directory PATH Specify an alternate working directory(default: the path of the Compose file)
- --compatibility If set, Compose will attempt to convert keys in v3 files to their non-Swarm equivalent
- build Build or rebuild services
- bundle Generate a Docker bundle from the Compose file
- config Validate and view the Compose file
- create Create services
- down Stop and remove containers, networks, images, and volumes
- events Receive real time events from containers
- exec Execute a command in a running container
- help Get help on a command
- images List images
- kill Kill containers
- logs View output from containers
- pause Pause services
- port Print the public port for a port binding
- ps List containers
- pull Pull service images
- push Push service images
- restart Restart services
- rm Remove stopped containers
- run Run a one-off command
- scale Set number of containers for a service

| | |
|---|---|
| start | Start services |
| stop | Stop services |
| top | Display the running processes |
| unpause | Unpause services |
| up | Create and start containers |
| version | Show the Docker-Compose version information |

**Docker compose**
**: tool for defining & running multi-container docker applications**
**: use yaml files to configure application services (docker-compose.yml)**
**: can start all services with a single command : docker compose up**
**: can stop all services with a single command : docker compose down**
**: can scale up selected services when required**

**Step 1 : install docker compose**
**(already installed on windows and mac with docker)**
**docker-compose -v**

**Ways**

**Using PIP**
**pip install -U docker-compose**

**Step 2 : Create docker compose file at any location on your system**
**docker-compose.yml**

**Step 3 : Check the validity of file by command**
**docker-compose config**

**Step 4 : Run docker-compose.yml file by command**
**docker-compose up -d**

**Steps 5 : Bring down application by command**
**docker-compose down**

**TIPS**
**How to scale services**

**—scale**
**docker-compose up -d --scale database=4**

## Dockers Compose and File Difference

A Dockerfile is a simple text file that contains the commands a user could call to assemble an image whereas Docker Compose is a tool for defining and running multi-container Docker applications.

Docker Compose define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment. It get an app running in one command by just running docker-compose up.Docker compose uses the Dockerfile if one add the build command to your project’s docker-compose.yml. Your Docker workflow should be to build a suitable Dockerfile for each image you wish to create, then use compose to assemble the images using the build command.

**Docker Volume**

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers. While bind mounts are dependent on the directory structure of the host machine, volumes are completely managed by Docker.

Volumes have several advantages over bind mounts:

- Volumes are easier to back up or migrate than bind mounts.
- You can manage volumes using Docker CLI commands or the Docker API.
- Volumes work on both Linux and Windows containers.
- Volumes can be more safely shared among multiple containers.
- Volume drivers let you store volumes on remote hosts or cloud providers, to encrypt the contents of volumes, or to add other functionality.
- New volumes can have their content pre-populated by a container.

**Differences between -v and --mount behavior**

- As opposed to bind mounts, all options for volumes are available for both --mount and -v flags.
- When using volumes with services, only --mount is supported.
- docker volume create my-vol  **Create a volume**
- docker volume ls  **List volume**
- docker volume inspect my-vol  **Inspect a volume**
- docker volume rm my-vol  **Remove a volume**

Following are the auto executable file condition base in the end of the dockerfile

- **stop.sh ( a script that will stop all services )**
- **start.sh ( a script that will auto start all services )**

## Exposing Ports

Exposing ports is a way of documenting which ports are used, but does not actually map or open any ports. Exposing ports is optional. You publish ports using the --publish or --publish-all flag to docker run . This tells Docker which ports to open on the container's network interface.

**E.g :- EXPOSE 8000 (place it in the docker-compose / dockerfile)**

## Remove Container Commands

- docker container rm id/name … to remove container
- docker container rm -f id/name … to remove running container forcefully
- docker container rm id/name id/name id/name … to remove multiple containers

## Remove Single Image Commands

- docker rmi image name /id

**OR**

- docker image rm IMAGE ID … to remove image

## Start a Container from an Image Commands

- docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
- E.g :docker run --name test -it debian

## Advance Commands

### Conditional dependency based on healthcheck

In certain cases, it is necessary to define the order in which specific services are started (e.g. applications should start after the database has booted successfully). Therefore, Docker introduced health checks for containers, which can be used to constrain the startup order of the services. In the example below the service, *pgweb* depends on a healthy *db* service. Unfortunately, this feature was removed in version 3 of Docker Compose.

```
version: '3.0'

services:
 db:
   image: postgres:10
   environment:
     POSTGRES_USER: admin
     POSTGRES_PASSWORD: password
   healthcheck:
     test: ["CMD-SHELL", "pg_isready -U postgres"]
     interval: 30s
```

```
    timeout: 30s
    retries: 3


  pgweb:
    image: sosedoff/pgweb
    ports:
      - "8081:8081"
    environment:
      - DATABASE_URL=postgres://admin:password@db:5432/postgres?sslmode=disable
    depends_on:
      db:
        condition: service_healthy
```

## Network Security

Docker Compose provides functionalities for defining networks between individual services, whereas containers are only allowed to talk to services within the same network. The example below shows three services and two networks (i.e. *db-backend* and *frontend*). It prohibits communication between services *maildev* and *db.* Only *pgweb* and *db* are allowed to talk to each other since they are part of the same network. The results can be checked via accessing **http://localhost:8081** after *run.sh* has been executed.

```
version: '3'


services:
```

```yaml
maildev:
  image: djfarrelly/maildev:1.0.0-rc2
  ports:
    - "8082:80"
    - "25"
  networks:
    - frontend

pgweb:
  image: sosedoff/pgweb
  restart: on-failure
  ports:
    - "8081:8081"
  environment:
    - 
DATABASE_URL=postgres://admin:password@db:5432/postgres?sslmode=disable
  networks:
    - frontend
    - db-backend

db:
  image: postgres:10
  environment:
    POSTGRES_USER: admin
    POSTGRES_PASSWORD: password
  networks:
    - db-backend

networks:
 db-backend:
 Frontend:
```

## Database Initialisation

Sometimes it is handy to start with an initialised database (e.g. inserting dumped database exports). The official PostgreSQL image from Docker supports this feature.

```
version: '3'

services:
 pgweb:
   image: sosedoff/pgweb
   restart: on-failure
   ports:
     - "8081:8081"
   environment:
     -
DATABASE_URL=postgres://admin:password@db:5432/persons?sslmode=disable
 db:
   image: postgres:10
```

```
    ports:
      - "5432:5432"
    environment:
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: password
      POSTGRES_DB: persons
    volumes:
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
```

## Docker-in-Docker

A very handy but dangerous way to use Docker inside Docker containers is linking *docker.sock* from the host via volumes (see code below). This way the Docker installation of the host can be used inside the container. A scenario for this setup might be a CI framework (e.g. Jenkins) inside a Docker container, which should be able to build and push Docker images.

```
version: '3'

services:
 dind:
   image: docker
   volumes:
     - /var/run/docker.sock:/var/run/docker.sock
   entrypoint: ["docker", "run", "busybox", "echo", "\"hello from busybox\""]
```

# Bash Scripting

Anything you can run normally on the command line can be put into a script and it will do exactly the same thing. Similarly, anything you can put into a script can also be run normally on the command line and it will do exactly the same thing.

**nano myscript.sh :** write into the file
**./myscript.sh :** run .sh file
**ls -l myscript.sh :** check permissions of the file
**chmod 755 myscript.sh :** give executable permissions to the file **rwx**

**My .sh file :**

```
#!/bin/bash
echo Hello World!
ls
Pwd
```

**Output:**

```
Hello World!
myscript.sh
/Users/nomanirfan/desktop/bash
```

**Variables:**
**Special Variables**

- **$0 -** The name of the Bash script.
- **$1 - $9 -** The first 9 arguments to the Bash script. (As mentioned above.)

- **$# -** How many arguments were passed to the Bash script.
- **$@ -** All the arguments supplied to the Bash script.
- **$? -** The exit status of the most recently run process.
- **$$ -** The process ID of the current script.
- **$USER -** The username of the user running the script.
- **$HOSTNAME -** The hostname of the machine the script is running on.
- **$SECONDS -** The number of seconds since the script was started.
- **$RANDOM -** Returns a different random number each time is it referred to.
- **$LINENO -** Returns the current line number in the Bash script.

**Input + Output Printing**

```
echo Hello, who am I talking to?
read varname
echo It\'s nice to meet you $varname
```

**echo:** Display output
**read:** Takes input from command line

It's common in Linux to pipe a series of simple, single purpose commands together to create a larger solution tailored to our exact needs. The ability to do this is one of the real strengths of Linux. It turns out that we can easily accommodate this mechanism with our scripts also. By doing so we can create scripts that act as filters to modify data in specific ways for us.

Bash accommodates piping and redirection by way of special files. Each process gets its own set of files (one for STDIN, STDOUT and STDERR respectively) and they are linked when piping or redirection is invoked. Each process gets the following files:

- **STDIN** - /proc/<processID>/fd/0
- **STDOUT** - /proc/<processID>/fd/1
- **STDERR** - /proc/<processID>/fd/2

To make life more convenient the system creates some shortcuts for us:

- **STDIN** - /dev/stdin or /proc/self/fd/0
- **STDOUT** - /dev/stdout or /proc/self/fd/1

- **STDERR** - /dev/stderr or /proc/self/fd/2

**fd** in the paths above stands for file descriptor.

## Setting own variables

**variable=value**
There is no space before and after = sign.  We also leave off the $ sign from the beginning of the variable name when setting it. We can also assign path to variables and we can use that variable in different ways
For Example

```
#!/bin/bash

firstvariable=Hellow

secondvariable=World

echo $firstvariable $secondvariable

echo

dir=/Users/zainulabdeen/Desktop/LNMP_stack

ls $dir
```

The output of this file is
Hellow World and it also shows the list of the $dir directory

The variables only had to store a single word. When we want variables to store more complex values however, we need to make use of quotes. This is because under normal circumstances Bash uses a space to determine separate items.

- Single quotes will treat every character literally.

- Double quotes will allow you to do substitution (that is include variables within the setting of the value and it prints out the value of variable used in echo statement).
- We can also assign commands to the variables

**Exporting Variables**

We can also export variable with its value from one file to the other

**User Input/Output**

Example:-

```
echo  Your name?

read varname

echo Hellow $varname
```

Two commonly used options however are **-p** which allows you to specify a prompt and **-s** which makes the input silent. This can make it easy to ask for a username and password combination like the example below:

```
read -p ‘Username: ‘ uservar

read -sp ‘Password: ’ passvar

echo $uservar your password is $passvar
```

-s makes input non-show like a password input

## Arithmetic Operations

**Let**

let <arithmetic expression>

let a=5+4

echo $a  #9

let “a = 5 + 4” (If we use quotes we can use space between arithmetics)

echo $a  #9

let a++

let “a = 4 * 5“

## Expression

Expr is similar to let except instead of saving the result to a variable it instead prints the answer. Unlike let you don't need to enclose the expression in quotes. You also must have spaces between the items of the expression. It is also common to use expr within command substitution to save the output to a variable.

Example:

expr 5 + 4

expr “5 + 4”

expr 5+4

## Double Parentheses

In the section on Variables we saw that we could save the output of a command easily to a variable. It turns out that this mechanism is also able to do basic arithmetic for us if we tweak the syntax a little. We do so by using double brackets

Example:

```
a=$(( 4 + 5 ))

echo $a  #9


a=$((4+5))

echo $a  #9

b=$(( a + 3 ))

echo $b  #12

b=$(( $a + 3 ))

echo $b  #12


(( b++ ))

echo $b  #13


(( b += 3 ))

echo $b  #16
```

## Length of variable

```
${#variable}
```

Example:

```
a='Hello World'

echo ${#a} #11


b=4953

echo ${#b}  #4
```

**If Statements**

If a particular test is true, then perform a given set of actions. If it is not true then don't perform those actions. If follows the format below:

```
if [ <some test> ]

then

<commands>

fi
```

Anything between then and fi (if backwards) will be executed only if the test (between the square brackets) is true.

| Operator | Description |
|---|---|

| | |
|---|---|
| ! EXPRESSION | The EXPRESSION is false. |
| -n STRING | The length of STRING is greater than zero. |
| -z STRING | The lengh of STRING is zero (ie it is empty). |
| STRING1 = STRING2 | STRING1 is equal to STRING2 |
| STRING1 != STRING2 | STRING1 is not equal to STRING2 |
| INTEGER1 -eq INTEGER2 | INTEGER1 is numerically equal to INTEGER2 |
| INTEGER1 -gt INTEGER2 | INTEGER1 is numerically greater than INTEGER2 |
| INTEGER1 -lt INTEGER2 | INTEGER1 is numerically less than INTEGER2 |
| -d FILE | FILE exists and is a directory. |
| -e FILE | FILE exists. |
| -r FILE | FILE exists and the read permission is granted. |
| -s FILE | FILE exists and it's size is greater than zero (ie. it is not empty). |
| -w FILE | FILE exists and the write permission is granted. |
| -x FILE | FILE exists and the execute permission is granted. |

## Nested If statements

```
If [ $1 -gt 10 ]

then

        echo Thats a large number

        If (( $1 % 2 == 0 ))

        then

                echo And also an even number

        fi

fi
```

**If else with Break**

The break statement terminates the current loop and passes program control to the command that follows the terminated loop.

**Example:**

```
        i=0

        while [[ $i -lt 5 ]]

        do

          echo "Number: $i"

          ((i++))

          if [[ $i -eq 2 ]]; then

            break
```

```
      fi
    done
    echo 'All Done!'
```

**If else with Continue**

```
i=0
while [[ $i -lt 5 ]]; do
  ((i++))
  if [[ "$i" == '2' ]]; then
    continue
  fi
  echo "Number: $i"
done
```

**Switch Statements**

Example

```
Case $1 in
    start)
        echo starting
        ;;
    stop)
        echo stop
        ;;
    restart)
        echo restarting
        ;;
    *)
        echo Don\'t Know
esac
```

- If $1 is equal to 'start' then perform the subsequent actions. the ) signifies the end of the pattern.
- We identify the end of this set of statements with a double semi-colon ( ;; ). Following this is the next case to consider.
- Remember that the test for each case is a pattern. The ***** represents any number of any character. It is essentialy a catch all if for if none of the other cases match. It is not necessary but is often used.
- esac is case backwards and indicates we are at the end of the case statement. Any other statements after this will be executed normally.

## While Loop

**Example**:

```
counter=1
While [ $counter -le 5 ]
do
        echo $counter
        ((counter++))
    Done
```

## Until Loop

The until loop is fairly similar to the while loop. The difference is that it will execute the commands within it until the test becomes true.

Example

```
    counter=1
    until [ $counter -le 5 ]
    do
        echo $counter
        ((counter++))
    Done
```

## For Loop

Example

```
names='Stan Kyle Cartman'
for name in $names
do
    echo $name
done
```

## Functions

Functions can be written in two ways

1. function_name() {

        Command

    }

        function_name () { commands; }  => single line version

2. function function_name {

        Commands

    }

## Scope of variables

______A variable defined outside of function can be used within the function and the variable defines within function cxan be used outside of the function with the same value

But if we use word ‘local’ before variable then the scope of that variable remains within that function (eg local var1=’C’)

## String Manipulation

```
var1=”Hello,”

var2=” World”

var3=$var1$var2

echo “$var3”
```

**Concatenation one or more variable with literal strings:**

```
var1=”Hello, ”

var2=”${var1}World”

echo “$var2”
```

**Concatenating Strings with the += Operator**

```
var1=”Hello,”
var1+=”World”
echo “$var1”
```

**Task 1:**

**find /Users/nomanirfan/desktop/dir -type f -name "*.txt" -exec grep -H 'virus' {} \; -delete**

**Task2:**

```
find . -type f -name "*.php" -exec chmod 755 {} \;
  find . -type f -name "*.sh"  -delete
```

# Day 7 : Wednesday, 29 April 2020
# Bash Scripting

## Return and Catch exit codes

**Some exit codes list below**

- 1 - Catchall for general errors
- 2 - Misuse of shell builtins (according to Bash documentation)
- 126 - Command invoked cannot execute
- 127 - “command not found”
- 128 - Invalid argument to exit
- 128+n - Fatal error signal “n”
- 130 - Script terminated by Control-C
- 255* - Exit status out of range

## Reading and writing to files

We  can read file by using **cat** command and in order to write file cat command is also used echo and printf command also used to write in file but cat command is preferable

- cat filename  => To read content of file
- cat > filename  => To write content into file

## Pipes

In bash, a pipe is the | character with or without the & character. With the power of both characters combined we have the control operators for pipelines, | and |&. In pipes the output of command is input of the other command

## Array in Shell Scripting

An array is a systematic arrangement of the same type of data. But in Shell script Array is a variable which contains multiple values may be of same type or different type since by default in shell script everything is treated as a string. An array is zero-based ie indexing start with 0.

### How to Declare Array in Shell Scripting?

We can declare an array in a shell script in different ways.

1. **Indirect Declaration**

In Indirect declaration, We assigned a value in a particular index of Array Variable. No need to first declare.

```
ARRAYNAME[INDEXNR]=value
```

2. **Explicit Declaration**

In Explicit Declaration, First We declare array then assigned the values.

```
declare -a ARRAYNAME
```

3. **Compound Assignment**

In Compount Assignment, We declare array with a bunch of values. We can add other values later too.

```
ARRAYNAME=(value1 value2  .... valueN)
```

- [@] & [*] means All elements of Array.
- echo ${arr[@]}        echo ${arr[*]}
- echo ${arr[@]:0}      => Print all values starting from zero index
- echo ${arr[@]:1}      => Print all values starting from index number 1
- echo ${arr[0]:1}      => Only print zero index value
- echo ${arr[@]:1:4}    => Print all values strating from index 1 to index 4
- echo ${arr[0]:1:3}    => Print zero index value and it only prints the part of value starting from index 1 to index 3
- echo ${#arr[0]}       => print length of zero index in array
- echo ${#arr[@]}       => print count length of array
- arr[@] =>  All Array Elements.

  /Search_using_Regular_Expression/ : Search in Array

- Search Returns 1 if it found the pattern else it return zero. It does not alter the original array elements.
- echo ${arr[@]/*[aA]*/}

**To Search & Replace in Array**

//Search_using_Regular_Expression/Replace : Search & Replace

Search & Replace does not change in Original Value of Array Element. It just returned the new value. So we can store this value in same or different variable.

echo ${arr[@]//a/A}  => it does not change the riginal value it just show values in this pattern

**To delete Array Variable in Shell Script?**

To delete index-1 element

- unset arrayname[1] => To delete index 1 value
- unset arrayname      => To delete whole array

## SSH Protocol

**What is SSH**
SSH, or Secure Shell, is a remote administration protocol that allows users to control and modify their remote servers over the Internet. The service was created as a secure replacement for the unencrypted Telnet and uses cryptographic techniques to ensure that all communication to and from the remote server happens in an encrypted manner. It provides a mechanism for authenticating a remote user, transferring inputs from the client to the host, and relaying the output back to the client.

**How SSH works**
In mac and linux just open terminal and type the command. The SSH command consists of 3 distinct parts:
command:
**ssh {username}@{hostip}**

The SSH key command instructs our system that we want to open an encrypted Secure Shell Connection. **{username}** represents the account you want to access. For example, we may want to access the **root** user, which is basically synonymous for system administrator with complete rights to modify anything on the system. **{hostip}** refers to the computer you want to access. This can be an IP Address **(e.g.192.164.0.1)**

# Advance Nginx
# How to set Nginx Proxy

## What is a Reverse Proxy?

A standard proxy server works on behalf of clients, often by providing privacy or filtering content. A reverse proxy works on behalf of a server, intercepting traffic and routing it to a separate server.

There are several reasons you might want to install a reverse proxy. One of the main reasons is privacy.

If you have multiple servers, a reverse proxy can help balance loads between servers and improve performance. As a reverse proxy provides a single point of contact for clients, it can centralize logging and report across multiple servers.

Nginx can improve performance by serving static content quickly, and passing dynamic content requests to Apache servers.

## What is a Reverse Proxy?

A standard proxy server works on behalf of clients, often by providing privacy or filtering content. A reverse proxy works on behalf of a server, intercepting traffic and routing it to a separate server.

There are several reasons you might want to install a reverse proxy. One of the main reasons is privacy.

If you have multiple servers, a reverse proxy can help balance loads between servers and improve performance. As a reverse proxy provides a single point of contact for clients, it can centralize logging and report across multiple servers.

Nginx can improve performance by serving static content quickly, and passing dynamic content requests to Apache servers.

This guide will help you install and configure an Nginx reverse proxy on your system.

## Setting Up an Nginx Reverse Proxy

## Step 1: Install Nginx from Default Repositories

Open a terminal window and enter the following:

```
sudo apt-get update
```

Allow the package manager to finish refreshing the software lists, then enter the following:

```
sudo apt-get install nginx
```

### Step 2 (optional): Install Nginx from Official Repository

### Add Security Key

In a terminal window, enter the following:

```
sudo wget https://nginx.org/keys/nginx_signing.key
```

```
sudo apt-key add nginx_signing.key
```

**Open sources.list File for Editing**

In the terminal, enter the following:

```
sudo vi /etc/apt/sources.list
```

**Add Nginx Sources to Repository List**

Enter the following lines in the /etc/apt/sources.list file you just opened:

```
deb https://nginx.org/packages/mainline/debian/ <CODENAME> nginx
```

```
deb-src https://nginx.org/packages/mainline/debian/ <CODENAME> nginx
```

**Install Latest Release of Nginx**

To install the latest release of Nginx, use the commands:

```
sudo apt-get remove nginx-common
sudo apt-get update
```

```
sudo apt-get install nginx
```

**Step 3: Start Nginx and Configure to Launch on Reboot**

To start Nginx:

```
sudo systemctl start nginx
```

To enable Nginx:

```
sudo systemctl enable nginx
```

To check Nginx is running:

```
sudo systemctl status nginx
```

**Step 4: Unlink Default Configuration File**

In the terminal, enter the following:

```
sudo unlink /etc/nginx/sites-enabled/default
```

**Step 5: Create New Configuration File**

To create a new configuration file, enter:

```
cd /etc/nginx/sites-available/
```

```
sudo vi custom_server.conf
```

Replace custom_server with a name that’s meaningful to you. In the new file, enter:

```
server {

listen 80;

location / {

proxy_pass http://my_server;

}
}
```

**Step 6: Link and Activate Configuration File**

To activate the new Nginx file, enter:

```
ln -s /etc/nginx/sites-available/custom_server.conf
```

```
/etc/nginx/sites-enabled/custom_server.conf
```

**Step 7: Test and Restart Nginx**

To test Nginx:

```
sudo service nginx configtest
```

To restart Nginx:

```
sudo service nginx restart
```

**Proxy Buffers**

By default, Nginx buffers traffic for servers that it proxies for. Buffers improve server performance as a server response isn't sent until the client finishes sending a complete response.

To turn the buffer off, open the configuration file from Step 5. Under the location/section, add the following:

```
proxy_buffering off
```

**Request Headers**

Headers provide the server information about the requests made, or about the client.

Nginx redefines two of the header fields: host is configured for $proxy_host, and connection is configured for close.  If you use those headers, be sure to change the behavior in the configuration file.

If any header strings are empty, Nginx simply eliminates those fields.

To change the way Nginx handles heathers, use the following commands in your configuration file:

```
location / {

proxy_set_header Host $host;

}
```

This example tells Nginx to set host to the $host variable.

To prevent a header field from being passed to the proxied server, use an empty string as follows:

```
location / {
```

```
proxy_set_header header-variable "";

}
```

**Load Balancing**

You can use the configuration file to route traffic to several servers. To use this configuration, your configuration file will look similar to this example:

```
http   {

server   {

proxy_pass http://my_server

}

}
```

In other words, the HTTP configuration goes outside the server configuration from Step 5.

To create a name for a group of servers, Use the upstream command:

```
}

http   {

upstream server_group   {

server my.server1.com weight=3;

server my.server2.com;

}

server  {

location / {

proxy_pass http://server_group;

}

}

}
```

**Configure NGINX**

Now we need to configure NGINX to use SSL. First, create a new configuration snippet file with the command:

```
sudo nano /etc/nginx/snippets/self-signed.conf
```

In that new file, add the following contents:

```
ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
```

Save and close that file.

Next, create a second configuration snippet that points to our newly-generated SSL key and certificate. To do this, issue the command:

```
sudo nano /etc/nginx/snippets/ssl-params.conf
```

In that new file, add the following contents:

```
ssl_protocols TLSv1.2;

ssl_prefer_server_ciphers on;

ssl_dhparam /etc/ssl/certs/dhparam.pem;

ssl_ciphers
ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384;

ssl_ecdh_curve secp384r1; # Requires nginx >= 1.1.0
```

```
ssl_session_timeout  10m;

ssl_session_cache shared:SSL:10m;

ssl_session_tickets off; # Requires nginx >= 1.5.9

# ssl_stapling on; # Requires nginx >= 1.3.7

# ssl_stapling_verify on; # Requires nginx => 1.3.7

resolver 8.8.8.8 8.8.4.4 valid=300s;

resolver_timeout 5s;

add_header X-Frame-Options DENY;

add_header X-Content-Type-Options nosniff;
add_header X-XSS-Protection "1; mode=block";
```

Because we are using a self-signed certificate, we disable SSL stapling (a method for quickly and safely determining whether or not an SSL certificate is valid). If you're not using a self-signed certificate, remove the # symbols before the two lines. You can also change the resolver line to reflect your preferred DNS servers. Save and close that file.

We also need to generate the dhparam.pem file with the command:

```
sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
```

The above command will take some time.

The next step is to configure NGINX to be aware that we're going to be using SSL. Let's assume you have a server block for example.com in sites-available. Open that server block with the command:

```
sudo nano /etc/nginx/sites-available/example.com
```

In that file, edit it to reflect the following:

```
server {

    listen 443 ssl;

    listen [::]:443 ssl;

    include snippets/self-signed.conf;

    include snippets/ssl-params.conf;

    server_name example.com www.example.com;

    root /var/www/example.com/html;

    index index.html index.htm index.nginx-debian.html;
}
```

Below that add a new server block (to perform an HTTPS redirect) like so:

```
server {

    listen 80;

    listen [::]:80;

    server_name example.com www.example.com;

    return 302 https://$server_name$request_uri;
}
```

Finally, we need to link from sites-available to sites-enabled with the command:

```
ln -s /etc/nginx/sites-available/www.example.com /etc/nginx/sites-enabled/
```

**Check the profiles**

Using ufw, we can check our new SSL-enabled profiles with the command:

```
sudo ufw app list

sudo systemctl restart nginx
```

**Pass Parameters in Nginx index.php**

```
server {

listen 80 default_server;

root /var/www/project/public;
index index.php server_name localhost;

location / {

 try_files $uri $uri/ /index.php?$args;  #this line pass arguments

#try_files $uri $uri/ /index.php?$query_string;

}

location ~ \.php$ {

 fastcgi_pass unix:/var/run/php5-fpm.sock; include fastcgi_params;

 }

 }
```

## Error Logs

Logs are very useful to monitor activities of any application apart from providing you with valuable information while you troubleshoot it.

Like any other application, NGINX also records events like visitors to your site, issues it encountered and more to log files. This information enables you to take preemptive measures in case you notice some serious discrepancies in the log events.

**Default Log**

1. Access log
2. Error Log

**Custom Log**

```
http {
 log_format custom '$remote_addr - $remote_user [$time_local] '
             '"$request" $status $body_bytes_sent '
             '"$http_referer" "$http_user_agent" "$gzip_ratio"';

 server {
```

```
                gzip on;
                ...
                access_log /var/log/nginx/domain1.access.log custom;
                ...
           }
}
```

Enable Error:

```
http {
        ...
        error_log  /var/log/nginx/error_log  crit;
        ...
}
```

## Advance IPtables
## How to Log Packets

## Log All Dropped Input Packets

When things are not working as expected with your IPTables rules, you might want to log the IPTables dropped packets for troubleshooting purpose.
If you already have whole bunch of iptables firewall rules, add these at the bottom, which will log all the dropped input packets (incoming) to the /var/log/messages

```
iptables -N LOGGING
iptables -A INPUT -j LOGGING

iptables -A LOGGING -m limit --limit 2/min -j LOG --log-prefix "IPTables-Dropped: " --log-level 4

iptables -A LOGGING -j DROP
```

In the above example, it does the following:

- iptables -N LOGGING: Create a new chain called LOGGING
- iptables -A INPUT -j LOGGING: All the remaining incoming packets will jump to the LOGGING chain
- line#3: Log the incoming packets to syslog (/var/log/messages). This line is explained below in detail.
- iptables -A LOGGING -j DROP: Finally, drop all the packets that came to the LOGGING chain. i.e now it really drops the incoming packets.

In the line#3 above, it has the following options for logging the dropped packets:

- -m limit: This uses the limit matching module. Using this you can limit the logging using –limit option.
- –limit 2/min: This indicates the maximum average matching rate for logging. In this example, for the similar packets it will limit logging to 2 per minute. You can also specify 2/second, 2/minute, 2/hour, 2/day. This is helpful when you don’t want to clutter your log messages with repeated messages of the same dropped packets.
- -j LOG: This indicates that the target for this packet is LOG. i.e write to the log file.

- –log-prefix “IPTables-Dropped: ” You can specify any log prefix, which will be appended to the log messages that will be written to the /var/log/messages file
- –log-level 4 This is the standard syslog levels. 4 is warning. You can use number from the range 0 through 7. 0 is emergency and 7 is debug.

## Log All Dropped Outgoing Packets

```
iptables -N LOGGING
iptables -A OUTPUT -j LOGGING

iptables -A LOGGING -m limit --limit 2/min -j LOG --log-prefix "IPTables-Dropped: " --log-level 4

iptables -A LOGGING -j DROP
```

## Log All Dropped Packets (both Incoming and Outgoing)

```
iptables -N LOGGING
iptables -A INPUT -j LOGGING
iptables -A OUTPUT -j LOGGING

iptables -A LOGGING -m limit --limit 2/min -j LOG --log-prefix "IPTables-Dropped: " --log-level 4

iptables -A LOGGING -j DROP
```

## IPTable Modules

**Ip ranges**

The iprange module allows you to specify source and/or destination addresses as ranges. While the built-in “-s” and “-d” will take a netmask, this can work with an arbitrary range of addresses, for example: “iptables -I INPUT -m iprange -s 192.168.1.5-192.168.1.100 -j ACCEPT”, that isn't an address range that can be specified by “192.168.1.0/25” sort of syntax.

### Multiport

Like “-m tcp –dport 22”, but can match multiple ports: “iptables -I INPUT -m tcp -m multiport –source-ports 25,110,143 -j ACCEPT” would match, in one rule, e-mail related ports.

### State

If you don't know “state”, you should. It's a great way of making an extremely effective yet simple firewall for a system. “iptables -I INPUT -i ! lo -m state –state ESTABLISHED,RELATED -j ACCEPT” combined with a default deny policy allows any incoming traffic related to your outgoing connections, and denies everything else.

### Connrate

This includes a “–connrate min:max” and matches when a connection (managed by the conntrack module) is within the range. An optional “!” can appear before the rate to invert the sense of the match.

## How to take backup of iptables

iptables-save > /opt/iptables.backup

## How to restore iptables from backup file

iptables-restore < /opt/iptables.backup

## Prevent DoS (Denial of Service) Attack

- iptables -A INPUT -p tcp --dport 80 -m limit --limit 25/minute --limit-burst 100 -j ACCEPT
    - -m limit: This uses the limit iptables extension
    - –limit 25/minute: This limits only a maximum of 25 connections per minute. Change this value based on your specific requirement

    ○ –limit-burst 100: This value indicates that the limit/minute will be enforced only after the total number of connections have reached the limit-burst level.

## Port Forwarding

The following example routes all traffic that comes to the port 442 to 22. This means that the incoming ssh connection can come from both port 22 and 422.

- iptables -t nat -A PREROUTING -p tcp -d 192.168.102.37 --dport 422 -j DNAT --to 192.168.102.37:22

If you do the above, you also need to explicitly allow incoming connections on the port 422.

```
iptables -A INPUT -i eth0 -p tcp --dport 422 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 422 -m state --state ESTABLISHED -j ACCEPT
```

# Ping Flood Attach

Use curl to download and save the pingflood.sh script listed under Resources to the victim virtual machine.

```
# curl <url of script> -o pingflood.sh
```

Make the script executable.

```
# chmod +x pingflood.sh
```

Use the iptables-save command to save the current rule set.

```
# iptables-save > iptables.old
```

Run pingflood.sh as root and it will replace the current firewall.

You can print the current rule set to stdout with iptables -nvL.

Run the ping flood attack a second time monitoring traffic with iptraf. After 5 seconds stop the attack and note the number packets lost. You should see that almost all the packets were dropped by the firewall rules.

**Ping Flood Script**

```
# Flush current rules
iptables -F
iptables -X

# Create a custom chain
iptables -N syn_flood

# Send SYN packets to new chain
iptables -A INPUT -p tcp --syn -j syn_flood
iptables -A syn_flood -m limit --limit 1/s --limit-burst 3 -j RETURN
iptables -A syn_flood -j DROP

# Limit the incoming ICMP ping requests and log
iptables -A INPUT -p icmp -m limit --limit  1/s --limit-burst 1 -j ACCEPT
iptables -A INPUT -p icmp -m limit --limit 1/s --limit-burst 1 -j LOG --log-prefix
PING-DROP:
iptables -A INPUT -p icmp -j DROP
iptables -A OUTPUT -p icmp -j ACCEPT
```

**Debug the Assignment**

- Make sure both virtual machines are on the same network.
- The ping -f command must be run as root.
- Install iptraf on the victim.

## GIT Commands

- git config
- git init
- git clone
- git add
- git commit
- git diff
- git reset
- git status
- git rm
- git log
- git show
- git tag
- git branch
- git checkout
- git merge
- git remote
- git push
- git pull
- git stash

## SQL MASTER SLAVE REPLICATION PROCESS

After installation of the mysql follow the following steps:

```
vim /etc/mysql/mysql.conf.d/mysqld.cnf
```

On the Master node, scroll and locate the bind-address attribute as shown below.

```
bind-address  =127.0.0.1 172.20.0.1
```

Change the loopback address to match the IP address of the Master node.

```
bind-address  =10.128.0.28
```

Next, specify a value for the server-id attribute in the [mysqld] section. The number you choose should not match any other server-id number. Let’s assign the value 1.

```
server-id =1
```

At the very end of the configuration file, copy and paste the lines below.

```
log_bin = /var/log/mysql/mysql-bin.log

log_bin_index =/var/log/mysql/mysql-bin.log.index

relay_log = /var/log/mysql/mysql-relay-bin
```

```
relay_log_index = /var/log/mysql/mysql-relay-bin.index
```

```
systemctl restart mysqls
```

```
systemctl status mysql
```

```
mysql -u root -p
```

```
mysql> CREATE USER 'replication_user'@'10.0.2.6' IDENTIFIED BY
'replica_password';
```

```
mysql> GRANT REPLICATION SLAVE ON *.* TO 'replication_user '@'10.0.2.6';
```

```
mysql> SHOW MASTER STATUS\G
```

**SALVE**

Head out to the slave server and like we did with the Master server, open the MySQL configuration file.

```
$ sudo vim  /etc/mysql/mysql.conf.d/mysqld.cnf
```

just like the master server, proceed to edit the following lines.

```
bind-address            = 10.0.2.6
```

Again, paste the lines below at the very end of the configuration file.

```
log_bin = /var/log/mysql/mysql-bin.log

log_bin_index =/var/log/mysql/mysql-bin.log.index

relay_log = /var/log/mysql/mysql-relay-bin

relay_log_index = /var/log/mysql/mysql-relay-bin.index
```

Next restart MySQL server on slave node.

```
$ sudo systemctl restart mysql
```

Once done, save and exit the text editor

Next, log in to the MySQL shell as shown.

```
$ sudo mysql -u root -p
```

In this step, you will need to make some configuration that will allow the slave server to connect to the master server. But first, stop the slave threads as shown.

```
mysql> STOP SLAVE;
```

To allow the slave server to replicate the Master server, run the command.

```
mysql> CHANGE MASTER TO MASTER_HOST  ='10.0.2.6', MASTER_USER
='replication_user', MASTER_PASSWORD ='numan@1234', MASTER_LOG_FILE =
'mysql-bin.000002', MASTER_LOG_POS = 1643;
```

```
mysql> START SLAVE;
```

```
$ sudo mysql -u root -p
```

Let's create a test database. In this case, we will create a database called replication_db.

```
mysql> CREATE DATABASE replication_db;
```

Now, log in to your MySQL instance in the slave server.

```
$ sudo mysql -u root -p
```

Now list the databases using the query.

```
mysql> SHOW DATABASES;
```

**Steps By Step**

**Replication of PostgreSQL using PITR (Point In Time Recovery)**

**# create directory for archive logs**

```
sudo -H -u postgres mkdir /var/lib/postgresql/pg_log_archive
```

**# enable archive logging**

```
sudo nano /etc/postgresql/12/main/postgresql.conf
```

**# Changes to be made :-**

```
  wal_level = replica
  archive_mode = on # (change requires restart)
  archive_command = 'test ! -f /var/lib/

postgresql/pg_log_archive/%f && cp %p /var/lib/postgresql/pg_log_archive/%f'
```

**# restart cluster**

```
sudo systemctl restart postgresql@12-main
```

**# create database with some data**

```
sudo su - postgres
psql -c "create database test;"
psql test -c "
create table posts (
  id integer,
  title character varying(100),
  content text,
  published_at timestamp without time zone,
  type character varying(100)
);

insert into posts (id, title, content, published_at, type) values
(100, 'Intro to SQL', 'Epic SQL Content', '2018-01-01', 'SQL'),
(101, 'Intro to PostgreSQL', 'PostgreSQL is awesome!', now(), 'PostgreSQL');
"
```

**# archive the logs**

```
psql -c "select pg_switch_wal();" # pg_switch_xlog(); for versions < 10
```

**# backup database**

```
pg_basebackup -Ft -D /var/lib/postgresql/db_file_backup
```

**# stop DB and destroy data**

```
sudo systemctl stop postgresql@12-main
rm /var/lib/postgresql/12/main/* -r
ls /var/lib/postgresql/12/main/
```

**# restore**

```
tar xvf /var/lib/postgresql/db_file_backup/base.tar -C /var/lib/postgresql/12/main/
tar xvf /var/lib/postgresql/db_file_backup/pg_wal.tar -C
/var/lib/postgresql/12/main/pg_wal/
```

**# add recovery.conf**

```
nano /var/lib/postgresql/12/main/recovery.conf

  restore_command = 'cp /var/lib/postgresql/pg_log_archive/%f %p'
```

**# start DB**

```
sudo systemctl start postgresql@12-main
```

**# verify restore was successful**

```
psql test -c "select * from posts;"
```

**——————————— Do PITR to a Specific Time ———————————**

**# backup database and gzip**

```
pg_basebackup -Ft -X none -D - | gzip > /var/lib/postgresql/db_file_backup.tar.gz
```

**# wait**

```
psql test -c "insert into posts (id, title, content, type) values
(102, 'Intro to SQL Where Clause', 'Easy as pie!', 'SQL'),
(103, 'Intro to SQL Order Clause', 'What comes first?', 'SQL');"
```

**# archive the logs**

```
psql -c "select pg_switch_wal();" # pg_switch_xlog(); for versions < 10
```

**# stop DB and destroy data**

```
sudo systemctl stop postgresql@10-main
rm /var/lib/postgresql/12/main/* -r
ls /var/lib/postgresql/12/main/
```

**# restore**

```
tar xvfz /var/lib/postgresql/db_file_backup.tar.gz -C /var/lib/postgresql/12/main/
```

**# add recovery.conf**

```
nano /var/lib/postgresql/12/main/recovery.conf
```

**Make Changes**

```
  restore_command = 'cp /var/lib/postgresql/pg_log_archive/%f %p'
  recovery_target_time = '2018-02-22 15:20:00 EST'
```

**# start DB**

```
sudo systemctl start postgresql@10-main
```

**# verify restore was successful**

```
psql test -c "select * from posts;"
tail -n 100 /var/log/postgresql/postgresql-12-main.log
```

**# complete and enable database restore**

```
psql -c "select pg_wal_replay_resume();”
```

**# Logs Show the time and date with you placed in the file will show you the PITR Working on time**

## Rsync

**Rsync** (Remote Sync) is a most commonly **used** command for copying and synchronizing files and directories remotely as well as locally in Linux/Unix systems.

| | |
|---|---|
| ● -v, –verbose | Verbose output |
| ● -q, –quiet | suppress message output |
| ● -a, –archive | archive files and directory while synchronizing ( -a equal to following options -rlptgoD) |
| ● -r, –recursive | sync files and directories recursively |
| ● -b, –backup | take the backup during synchronization |
| ● -u, –update | don’t copy the files from source to destination if destination files are newer |
| ● -l, –links | copy symlinks as symlinks during the sync |
| ● -n, –dry-run | perform a trial run without synchronization |
| ● -e, –rsh=COMMAND | mention the remote shell to use in rsync |
| ● -z, –compress | compress file data during the transfer |
| ● -h, –human-readable | display the output numbers in a human-readable format |
| ● –progress | show the sync progress during transfer |

## CronTAB

The **crontab** is a list of **commands** that you want to run on a regular schedule, and also the name of the **command** used to manage that list. **Crontab** stands for “**cron** table, ” because it uses the job scheduler **cron** to execute tasks; **cron** itself is named after “chronos, ” the Greek word for time.

**Command:**

```
1 2 3 4 5 /root/backup.sh
```

**Structure:**

```
* * * * * 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)
```

# Ansible

Ansible is an open-source software provisioning, configuration management, and application-deployment tool. It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows. It includes its own declarative language to describe system configuration

**Advantages of Ansible**

Very simple to set up and use: No special coding skills are necessary to use **Ansible's** playbooks (more on playbooks later). Powerful: **Ansible** lets you model even highly complex IT workflows. Flexible: You can orchestrate the entire application environment no matter where it's deployed.

An **Ansible playbook** is an organized unit of scripts that defines work for a server configuration managed by the automation tool **Ansible**. **Ansible** is a configuration management tool that automates the configuration of multiple servers by the use of **Ansible playbooks**.

**STEPS to write Ansible File**

1. Step 1 - Setup Ansible Playbook Project.
2. Step 2 - Generate Ansible Roles for the Directory Structure.
3. Step 3 - Setup hosts and site.yml.
4. Step 3 - Setup Common Roles.
5. Step 4 - Setup 'web' Roles.
6. Step 5 - Setup 'db' Roles.
7. Step 6 - Run the Ansible Playbook.
8. Step 7 - Testing.

## Server Monitoring Tools

## Nagios XI

Nagios XI is a centralized enterprise server, application, and network monitoring software. Powered by the Nagios Core 4 engine, users gain insight into server performance, network protocols, applications, and services. Nagios XI supports hundreds of third-party add-ons to allow the software to monitor common business applications. The platform has a customizable GUI so users can create the best layout and design for their IT team’s needs.

## CONFIGURATIONS

**Installation**

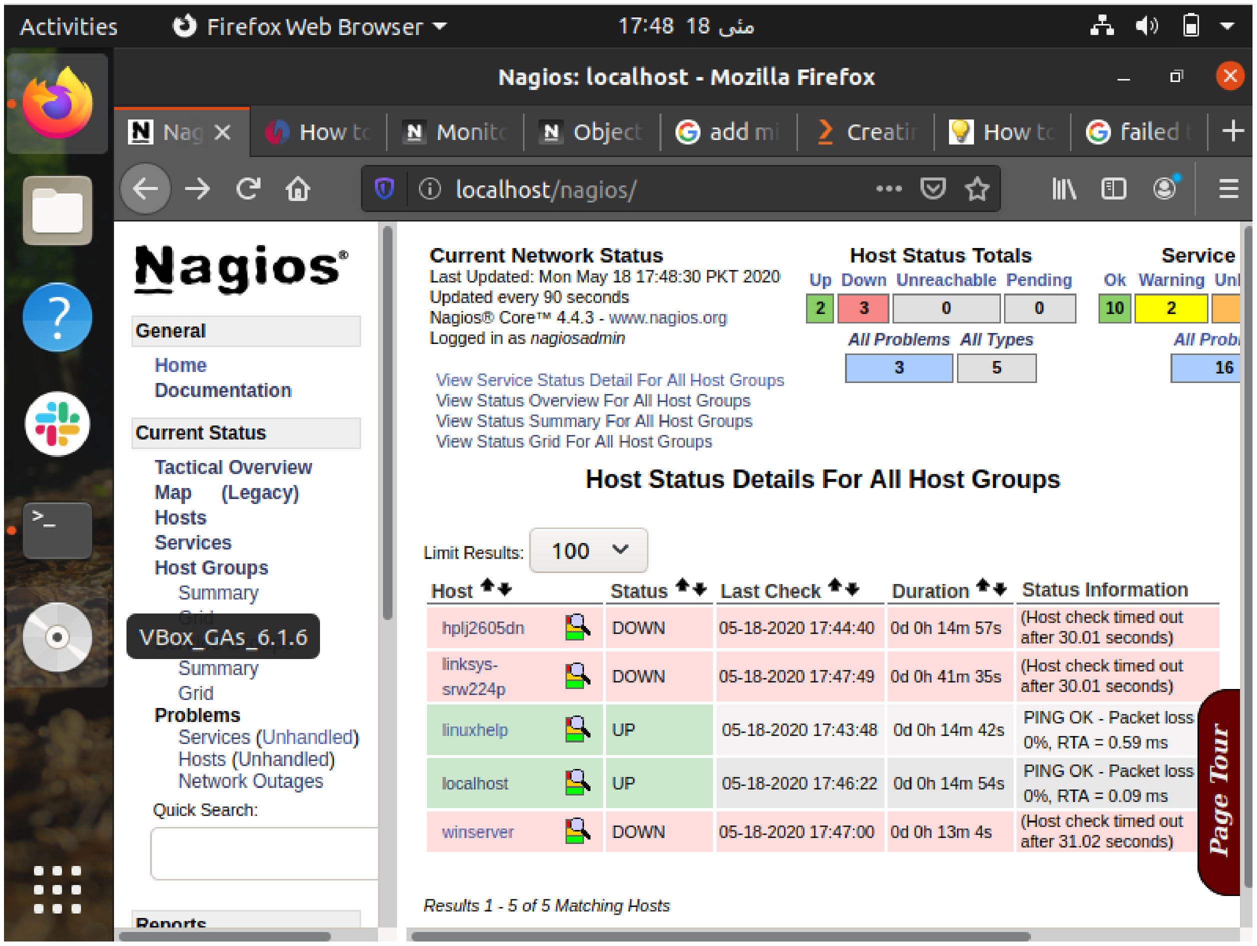
```
apt-get install wget build-essential apache2 php apache2-mod-php7.0 php-gd libgd-dev
sendmail -y
```

```
useradd nagios
```

```
groupadd nagcmd

usermod -a -G nagcmd nagios

usermod -a -G nagios,nagcmd www-data

wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.2.0.tar.gz

tar -xzvf nagios-4.2.0.tar.gz

cd nagios-4.2.0/

./configure --with-nagios-group=nagios --with-command-group=nagcmd

make all

make install

make install-commandmode/

make install-init

make install-config

cp -R contrib/eventhandlers/ /usr/local/nagios/libexec/

chown -R nagios:nagios /usr/local/nagios/libexec/eventhandlers
```

## **Installing Plugins**

```
wget https://nagios-plugins.org/download/nagios-plugins-2.1.2.tar.gz

tar -xzvf nagios-plugins-2.1.2.tar.gz

cd nagios-plugins-2.1.2/

./configure --with-nagios-user=nagios --with-nagios-group=nagios --with-openssl

make

Make install

vim /usr/local/nagios/etc/nagios.cfg

mkdir -p /usr/local/nagios/etc/servers

vim /usr/local/nagios/etc/objects/contacts.cfg

a2enmod rewrite

a2enmod cgi

htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

ln -s /etc/apache2/sites-available/nagios.conf /etc/apache2/sites-enabled/

vim /etc/systemd/system/nagios.service

systemctl start apache2
systemctl start nagios

systemctl enable nagios

vim /etc/init.d/nagios

chmod +x /etc/init.d/nagios

systemctl restart apache2

systemctl restart nagios
```



Current Network Status
Last Updated: Mon May 18 17:48:30 PKT 2020
Updated every 90 seconds
Nagios® Core™ 4.4.3 - www.nagios.org
Logged in as *nagiosadmin*

View Service Status Detail For All Host Groups
View Status Overview For All Host Groups
View Status Summary For All Host Groups
View Status Grid For All Host Groups

Host Status Totals

| Up | Down | Unreachable | Pending |
|---|---|---|---|
| 2 | 3 | 0 | 0 |

| All Problems | All Types |
|---|---|
| 3 | 5 |

Service

| Ok | Warning | Unl |
|---|---|---|
| 10 | 2 | |

| All Probl |
|---|
| 16 |

Host Status Details For All Host Groups

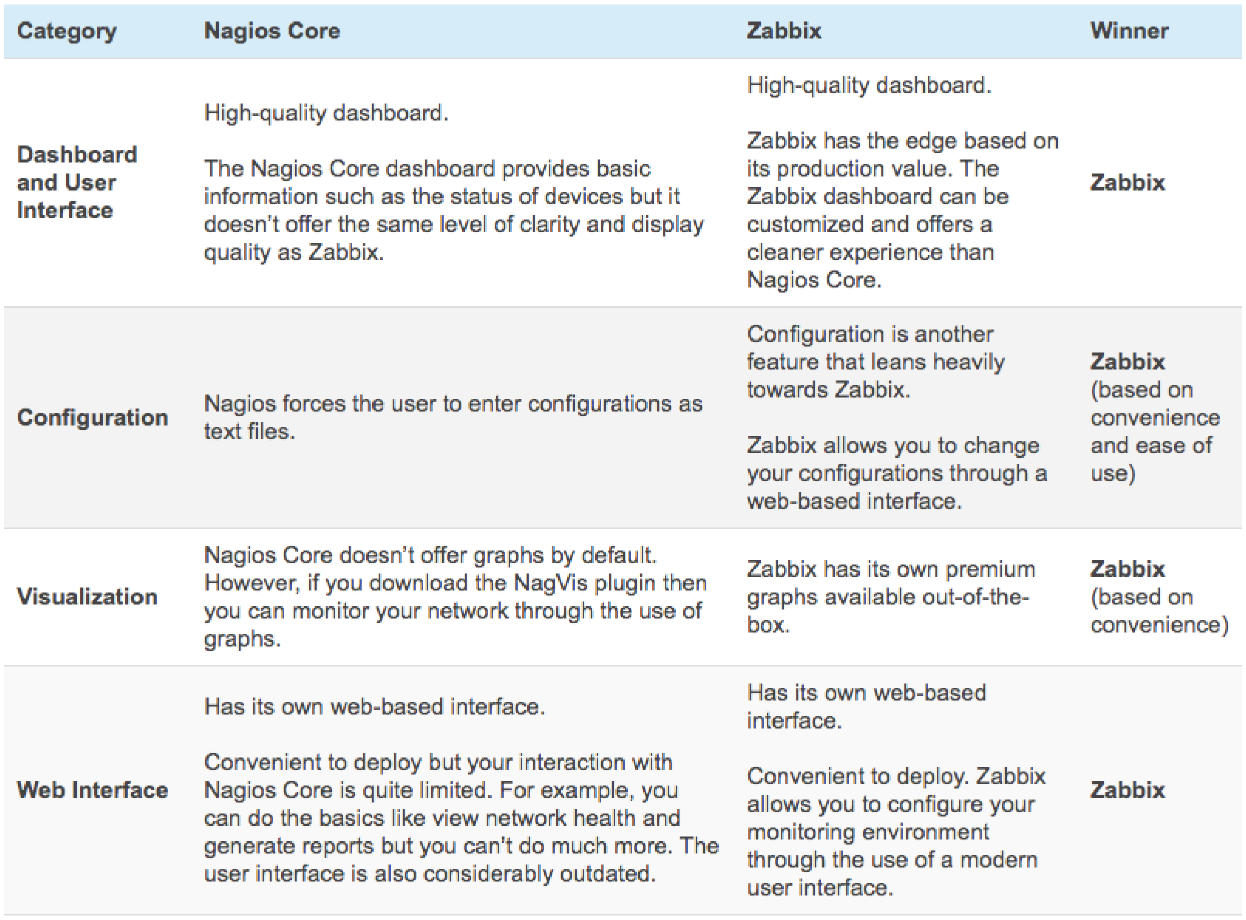
Limit Results: 100

| Host | Status | Last Check | Duration | Status Information |
|---|---|---|---|---|
| hplj2605dn | DOWN | 05-18-2020 17:44:40 | 0d 0h 14m 57s | (Host check timed out after 30.01 seconds) |
| linksys-srw224p | DOWN | 05-18-2020 17:47:49 | 0d 0h 41m 35s | (Host check timed out after 30.01 seconds) |
| linuxhelp | UP | 05-18-2020 17:43:48 | 0d 0h 14m 42s | PING OK - Packet loss 0%, RTA = 0.59 ms |
| localhost | UP | 05-18-2020 17:46:22 | 0d 0h 14m 54s | PING OK - Packet loss 0%, RTA = 0.09 ms |
| winserver | DOWN | 05-18-2020 17:47:00 | 0d 0h 13m 4s | (Host check timed out after 31.02 seconds) |

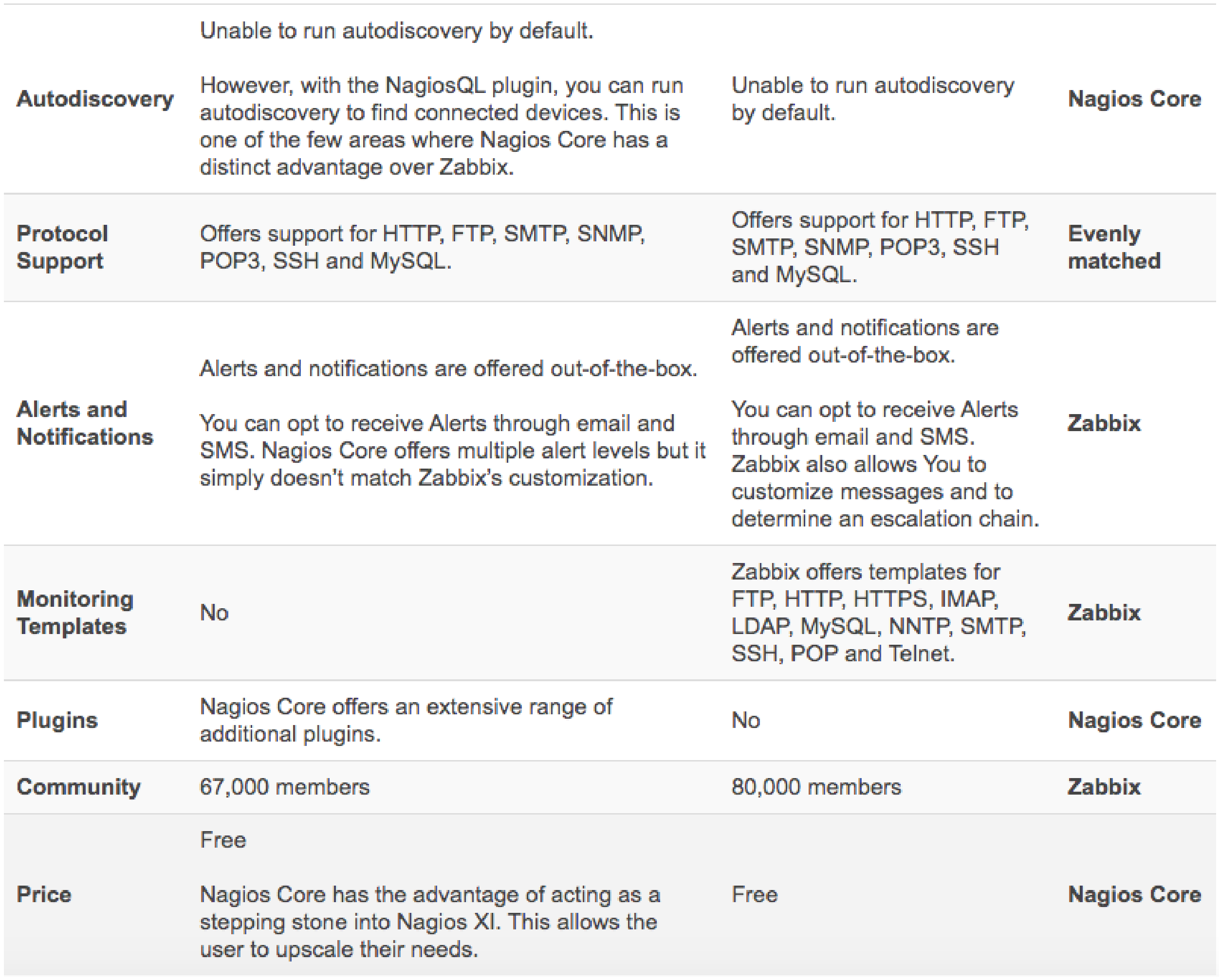
*Results 1 - 5 of 5 Matching Hosts*

## <u>COMPARISON NAGIOS VS ZABBIX</u>

After comparing the two it is clear that Zabbix is the winner. While Nagios Core has the basics in place to run effective network monitoring it simply doesn't have the experience and configurability that Zabbix does. Zabbix is a free network monitor that performs like a product situated in the very top price bracket.



| Category | Nagios Core | Zabbix | Winner |
|---|---|---|---|
| **Dashboard and User Interface** | High-quality dashboard.<br><br>The Nagios Core dashboard provides basic information such as the status of devices but it doesn't offer the same level of clarity and display quality as Zabbix. | High-quality dashboard.<br><br>Zabbix has the edge based on its production value. The Zabbix dashboard can be customized and offers a cleaner experience than Nagios Core. | **Zabbix** |
| **Configuration** | Nagios forces the user to enter configurations as text files. | Configuration is another feature that leans heavily towards Zabbix.<br><br>Zabbix allows you to change your configurations through a web-based interface. | **Zabbix** (based on convenience and ease of use) |
| **Visualization** | Nagios Core doesn't offer graphs by default. However, if you download the NagVis plugin then you can monitor your network through the use of graphs. | Zabbix has its own premium graphs available out-of-the-box. | **Zabbix** (based on convenience) |
| **Web Interface** | Has its own web-based interface.<br><br>Convenient to deploy but your interaction with Nagios Core is quite limited. For example, you can do the basics like view network health and generate reports but you can't do much more. The user interface is also considerably outdated. | Has its own web-based interface.<br><br>Convenient to deploy. Zabbix allows you to configure your monitoring environment through the use of a modern user interface. | **Zabbix** |



| | | | |
|---|---|---|---|
| **Autodiscovery** | Unable to run autodiscovery by default.<br><br>However, with the NagiosQL plugin, you can run autodiscovery to find connected devices. This is one of the few areas where Nagios Core has a distinct advantage over Zabbix. | Unable to run autodiscovery by default. | **Nagios Core** |
| **Protocol Support** | Offers support for HTTP, FTP, SMTP, SNMP, POP3, SSH and MySQL. | Offers support for HTTP, FTP, SMTP, SNMP, POP3, SSH and MySQL. | **Evenly matched** |
| **Alerts and Notifications** | Alerts and notifications are offered out-of-the-box.<br><br>You can opt to receive Alerts through email and SMS. Nagios Core offers multiple alert levels but it simply doesn't match Zabbix's customization. | Alerts and notifications are offered out-of-the-box.<br><br>You can opt to receive Alerts through email and SMS. Zabbix also allows You to customize messages and to determine an escalation chain. | **Zabbix** |
| **Monitoring Templates** | No | Zabbix offers templates for FTP, HTTP, HTTPS, IMAP, LDAP, MySQL, NNTP, SMTP, SSH, POP and Telnet. | **Zabbix** |
| **Plugins** | Nagios Core offers an extensive range of additional plugins. | No | **Nagios Core** |
| **Community** | 67,000 members | 80,000 members | **Zabbix** |
| **Price** | Free<br><br>Nagios Core has the advantage of acting as a stepping stone into Nagios XI. This allows the user to upscale their needs. | Free | **Nagios Core** |

## HA Proxy (Load Balancer and Monitoring)

HAProxy is installed with RightScale load balancer ServerTemplates. Load-balancer servers are also known as front-end servers. Generally, their purpose is to direct users to available application servers. A load-balancer server may have only the load balancer application (HAProxy) installed or, in rare cases, it may be an application server in addition to a load balancer, which is not a recommended configuration.

# HA Proxy installation

Now start the setup. SSH to your HAProxy server as a privileged user and install HAProxy using the following commands.

```
sudo add-apt-repository ppa:vbernat/haproxy-1.8

sudo apt-get update

sudo apt-get install haproxy
```

Now edit haproxy default configuration file /etc/haproxy/haproxy.cfg and start configuration.

```
sudo vi /etc/haproxy/haproxy.cfg
```

**Now configure file by replacing the following given code**

```
global

        log /dev/log    local0

        log /dev/log    local1 notice

        chroot /var/lib/haproxy

        stats socket /run/haproxy/admin.sock mode 660 level admin

        stats timeout 30s

        user haproxy

        group haproxy
        daemon


        # Default SSL material locations

        ca-base /etc/ssl/certs

        crt-base /etc/ssl/private


        # Default ciphers to use on SSL-enabled listening sockets.

        # For more information, see ciphers(1SSL). This list is from:

        #  https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/

        ssl-default-bind-ciphers
ECDH+AESGCM:DH+AESGCM:ECDH+AES256::RSA+AES:RSA+3DES:!aNULL:!MD5:!DSS

        ssl-default-bind-options no-sslv3



defaults

        log     global

        mode    http

        option  httplog

        option  dontlognull

     timeout connect 5000

     timeout client  50000

     timeout server  50000

        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http

        errorfile 408 /etc/haproxy/errors/408.http

        errorfile 500 /etc/haproxy/errors/500.http

        errorfile 502 /etc/haproxy/errors/502.http

        errorfile 503 /etc/haproxy/errors/503.http

        errorfile 504 /etc/haproxy/errors/504.http



frontend Local_Server

    Bind *:80

    mode http

    default_backend My_Web_Servers



backend My_Web_Servers

    mode http

    balance roundrobin      # round robin algo used to load server /website link for understanding it
(https://www.guru99.com/round-robin-scheduling-example.html)

    option forwardfor

    http-request set-header X-Forwarded-Port %[dst_port]

    http-request add-header X-Forwarded-Proto https if { ssl_fc }

    option httpchk HEAD / HTTP/1.1rnHost:localhost

    server server1.com  192.168.56.102:80

    server server2.com  192.168.56.102:80
```

```
listen stats *:1936

    stats enable

    stats hide-version

    stats refresh 30s

    stats show-node

    stats auth username:password

    stats uri  /haproxy?stats
```

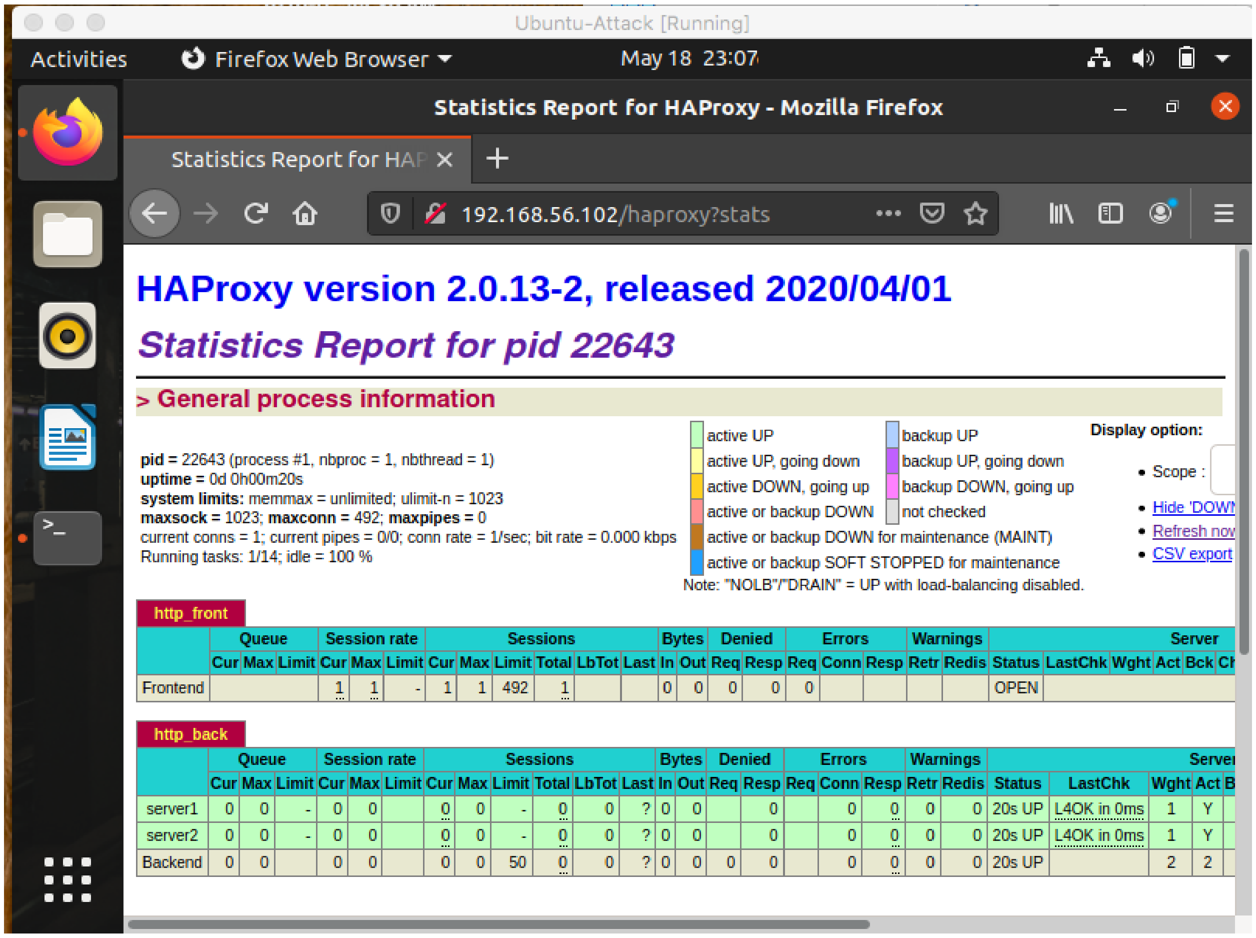
Now you have made all necessary changes in your HAProxy server. Now verify configuration file before restarting service using the following command.

```
haproxy -c -f /etc/haproxy/haproxy.cfg

sudo service haproxy restart
```

Free port 80 uisng : sudo kill -9 ‘ sudo lsof -t -i:80 ‘

**Help taken through :** [https://www.youtube.com/watch?v=TkiGgUkn_PI](https://www.youtube.com/watch?v=TkiGgUkn_PI)

## Verifications


Ubuntu-Attack [Running]
Activities
Firefox Web Browser
May 18 23:07
Statistics Report for HAProxy - Mozilla Firefox
Statistics Report for HAP
192.168.56.102/haproxy?stats

HAProxy version 2.0.13-2, released 2020/04/01

*Statistics Report for pid 22643*

> General process information

**pid** = 22643 (process #1, nbproc = 1, nbthread = 1)
**uptime** = 0d 0h00m20s
**system limits:** memmax = unlimited; ulimit-n = 1023
**maxsock** = 1023; **maxconn** = 492; **maxpipes** = 0
current conns = 1; current pipes = 0/0; conn rate = 1/sec; bit rate = 0.000 kbps
Running tasks: 1/14; idle = 100 %

| | |
|---|---|
| 🟩 active UP | 🟦 backup UP |
| 🟨 active UP, going down | 🟪 backup UP, going down |
| 🟧 active DOWN, going up | 🟪 backup DOWN, going up |
| 🟥 active or backup DOWN | ⬜ not checked |
| 🟫 active or backup DOWN for maintenance (MAINT) | |
| 🟦 active or backup SOFT STOPPED for maintenance | |

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

**Display option:**

- Scope :
- Hide 'DOWN
- Refresh now
- CSV export

**http_front**

| | Queue | | | Session rate | | | Sessions | | | | | | Bytes | | Denied | | Errors | | | Warnings | | Server | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | Last | In | Out | Req | Resp | Req | Conn | Resp | Retr | Redis | Status | LastChk | Wght | Act | Bck | Ch |
| Frontend | | | | 1 | 1 | - | 1 | 1 | 492 | 1 | | | 0 | 0 | 0 | 0 | 0 | | | | | OPEN | | | | | |

**http_back**

| | Queue | | | Session rate | | | Sessions | | | | | | Bytes | | Denied | | Errors | | | Warnings | | Server | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | Last | In | Out | Req | Resp | Req | Conn | Resp | Retr | Redis | Status | LastChk | Wght | Act | B |
| server1 | 0 | 0 | - | 0 | 0 | | 0 | 0 | - | 0 | 0 | ? | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 | 20s UP | L4OK in 0ms | 1 | Y | |
| server2 | 0 | 0 | - | 0 | 0 | | 0 | 0 | - | 0 | 0 | ? | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 | 20s UP | L4OK in 0ms | 1 | Y | |
| Backend | 0 | 0 | | 0 | 0 | | 0 | 0 | 50 | 0 | 0 | ? | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 20s UP | | 2 | 2 | |

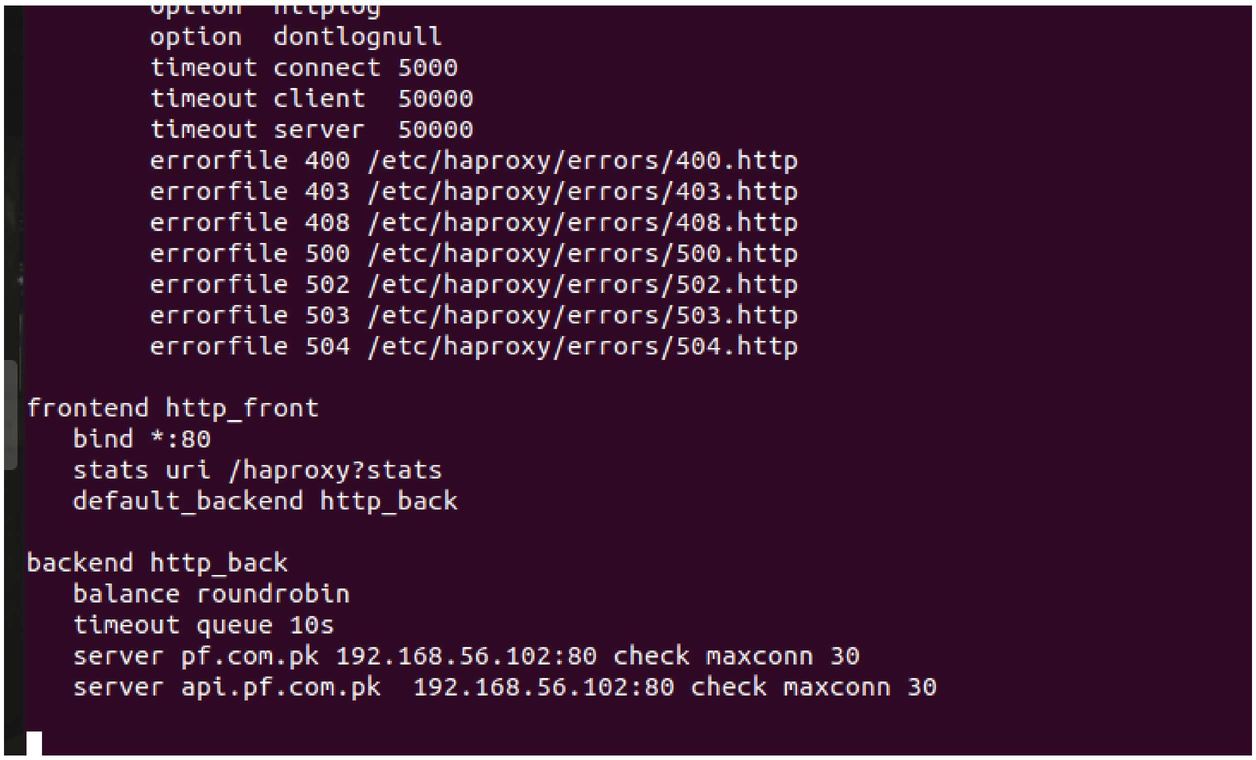
## HAProxy Questions :

- We have an application running on nginx port 443, how could we use HAProxy with it, and which ports will be used?

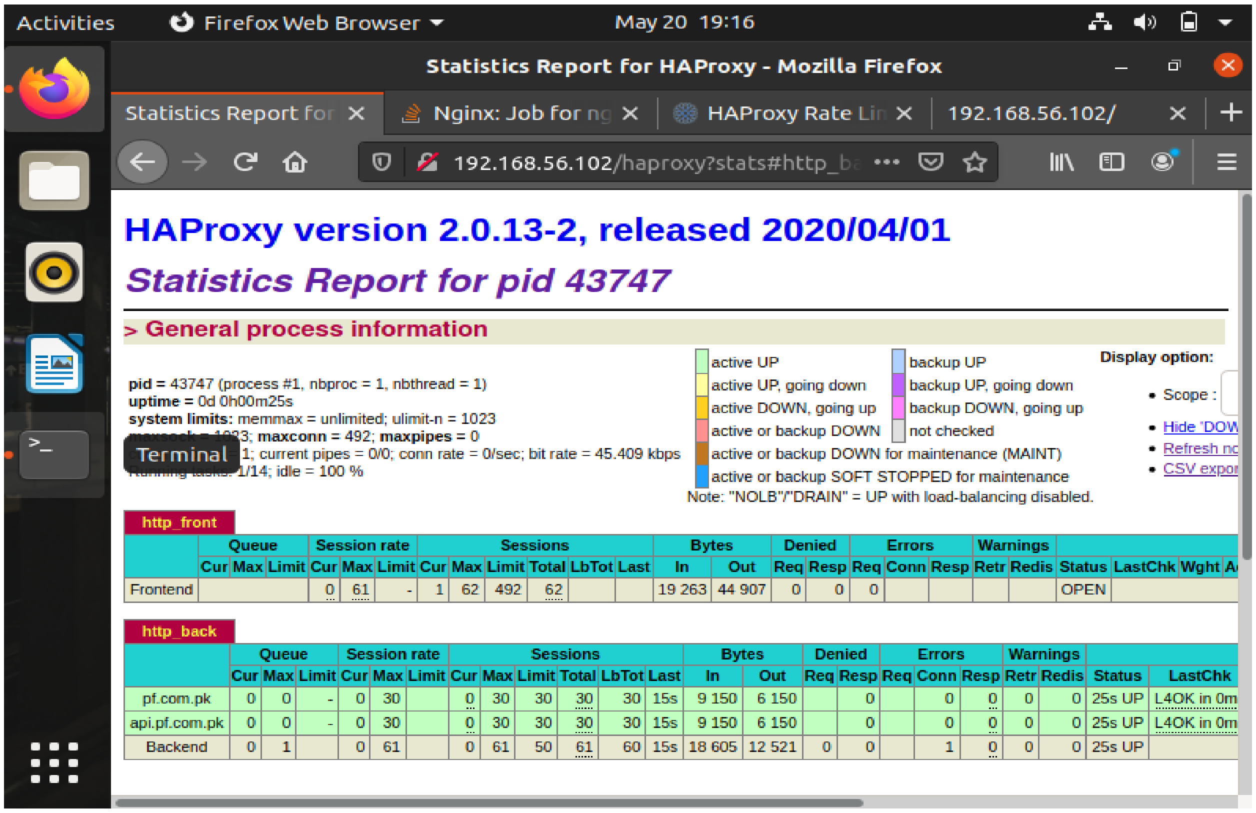
- How can we limit the user requests from HAProxy, so that DDOS doesn't land on the application? Proxy server should stop it at its end?
- Can we host multiple application servers for load balancing as well?

**We have an application running on nginx port 80 (default) , how could we use HAProxy with it, and which ports will be used?**

If i uses the ssl port 443, then it will use the same, i tried to use the port 443 it gave me error as there were no ssl certificates in it. As i put the port 80 it worked on 80



```
        option  dontlognull
        timeout connect 5000
        timeout client  50000
        timeout server  50000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http

frontend http_front
   bind *:80
   stats uri /haproxy?stats
   default_backend http_back

backend http_back
   balance roundrobin
   timeout queue 10s
   server pf.com.pk 192.168.56.102:80 check maxconn 30
   server api.pf.com.pk  192.168.56.102:80 check maxconn 30
```



**HAProxy version 2.0.13-2, released 2020/04/01**

***Statistics Report for pid 43747***

**> General process information**

**pid =** 43747 (process #1, nbproc = 1, nbthread = 1)
**uptime =** 0d 0h00m25s
**system limits:** memmax = unlimited; ulimit-n = 1023
**maxsock =** 1023; **maxconn** = 492; **maxpipes** = 0
[illegible] 1; current pipes = 0/0; conn rate = 0/sec; bit rate = 45.409 kbps
Running tasks: 1/14; idle = 100 %

| | |
|---|---|
| 🟩 active UP | 🟦 backup UP |
| 🟨 active UP, going down | 🟪 backup UP, going down |
| 🟧 active DOWN, going up | 🟪 backup DOWN, going up |
| 🟥 active or backup DOWN | ⬜ not checked |
| 🟫 active or backup DOWN for maintenance (MAINT) | |
| 🟦 active or backup SOFT STOPPED for maintenance | |

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

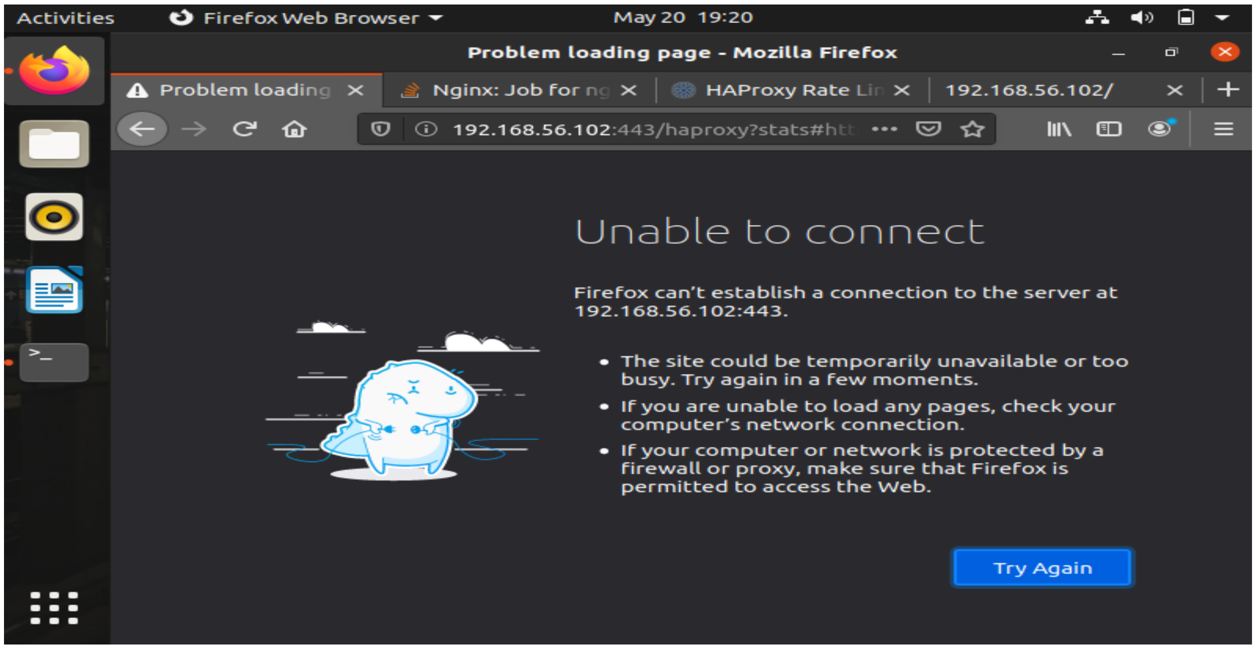
**Display option:**
- Scope :
- Hide 'DOW
- Refresh no
- CSV expor

**http_front**

| | Queue | | | Session rate | | | Sessions | | | | | | Bytes | | Denied | | Errors | | | Warnings | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | Last | In | Out | Req | Resp | Req | Conn | Resp | Retr | Redis | Status | LastChk | Wght | A |
| Frontend | | | | 0 | 61 | - | 1 | 62 | 492 | 62 | | | 19 263 | 44 907 | 0 | 0 | 0 | | | | | OPEN | | | |

**http_back**

| | Queue | | | Session rate | | | Sessions | | | | | | Bytes | | Denied | | Errors | | | Warnings | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | Last | In | Out | Req | Resp | Req | Conn | Resp | Retr | Redis | Status | LastChk |
| pf.com.pk | 0 | 0 | - | 0 | 30 | | 0 | 30 | 30 | 30 | 30 | 15s | 9 150 | 6 150 | | 0 | | 0 | 0 | 0 | 0 | 25s UP | L4OK in 0m |
| api.pf.com.pk | 0 | 0 | - | 0 | 30 | | 0 | 30 | 30 | 30 | 30 | 15s | 9 150 | 6 150 | | 0 | | 0 | 0 | 0 | 0 | 25s UP | L4OK in 0m |
| Backend | 0 | 1 | | 0 | 61 | | 0 | 61 | 50 | 61 | 60 | 15s | 18 605 | 12 521 | 0 | 0 | | 1 | 0 | 0 | 0 | 25s UP | |



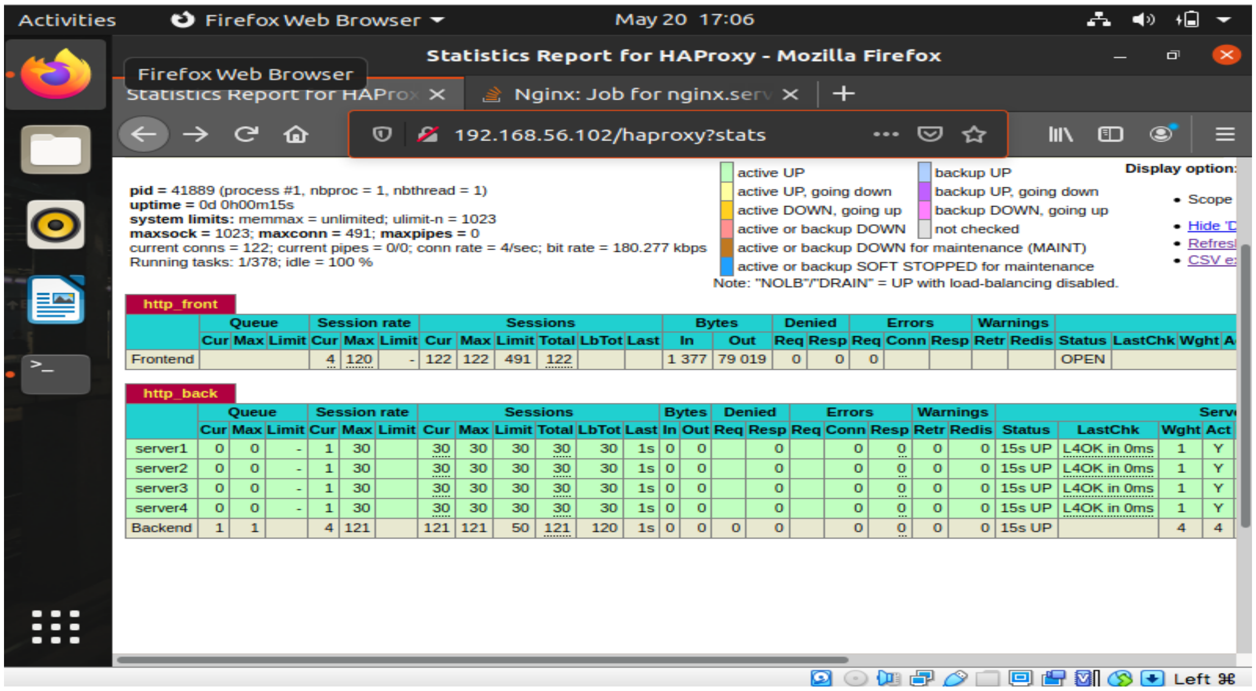
Unable to connect

Firefox can't establish a connection to the server at 192.168.56.102:443.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

**How can we limit the user requests from HAProxy, so that DDOS doesn't land on the application? Proxy server should stop it at its end?**

As it is clear from the above given there was no limit in it due to which there could be any type of attack on it so i use maxconn which is used to limit backend users as well as timeout queue that may help us to make it run faster as it will be available. Following are the given screenshots:


Activities
Firefox Web Browser
May 20 17:06
Statistics Report for HAProxy - Mozilla Firefox
Firefox Web Browser
Statistics Report for HAProx
Nginx: Job for nginx.serv
192.168.56.102/haproxy?stats
Left ⌘

**pid** = 41889 (process #1, nbproc = 1, nbthread = 1)
**uptime** = 0d 0h00m15s
**system limits:** memmax = unlimited; ulimit-n = 1023
**maxsock** = 1023; **maxconn** = 491; **maxpipes** = 0
current conns = 122; current pipes = 0/0; conn rate = 4/sec; bit rate = 180.277 kbps
Running tasks: 1/378; idle = 100 %

| | |
|---|---|
| 🟩 active UP | 🟦 backup UP |
| 🟨 active UP, going down | 🟪 backup UP, going down |
| 🟧 active DOWN, going up | 🟪 backup DOWN, going up |
| 🟥 active or backup DOWN | ⬜ not checked |
| 🟫 active or backup DOWN for maintenance (MAINT) | |
| 🟦 active or backup SOFT STOPPED for maintenance | |

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

**Display option:**

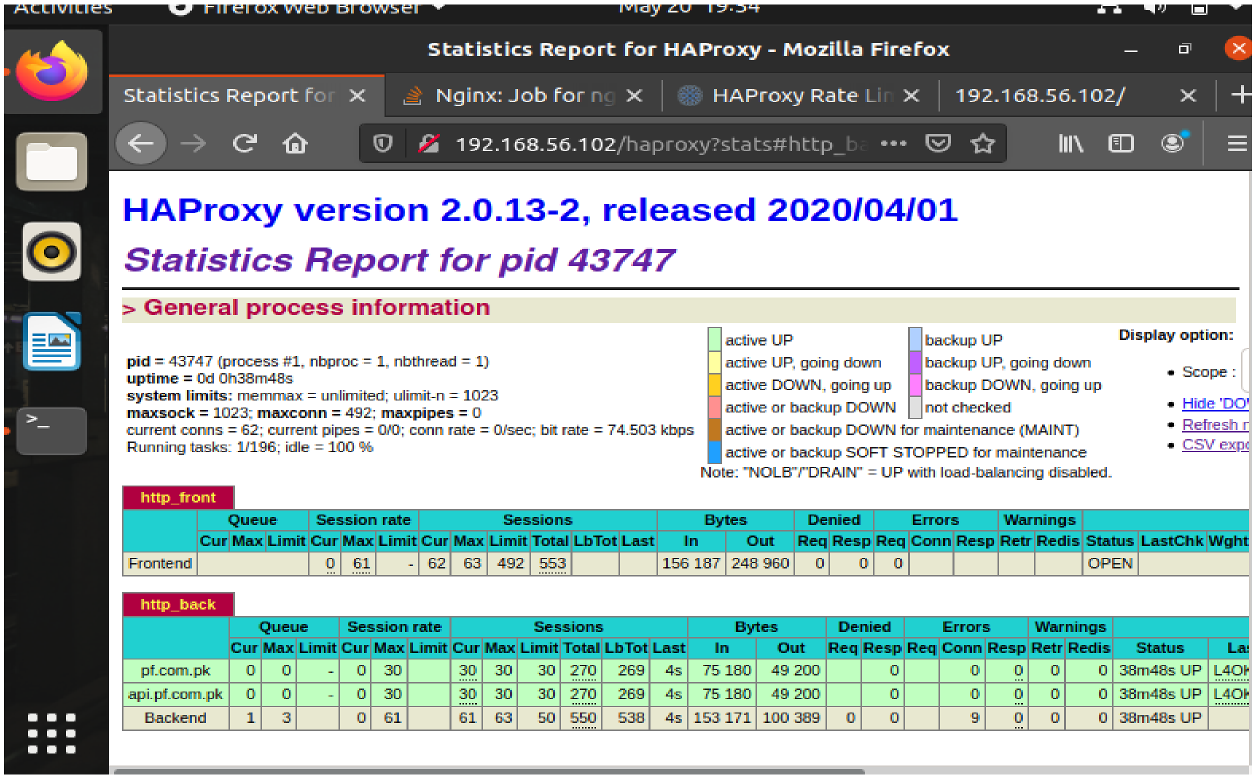
- Scope
- Hide 'D
- Refresl
- CSV e

**http_front**

| | Queue | | | Session rate | | | Sessions | | | | | | Bytes | | Denied | | Errors | | | Warnings | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | Last | In | Out | Req | Resp | Req | Conn | Resp | Retr | Redis | Status | LastChk | Wght | A |
| Frontend | | | | 4 | 120 | - | 122 | 122 | 491 | 122 | | | 1 377 | 79 019 | 0 | 0 | 0 | | | | | OPEN | | | |

**http_back**

| | Queue | | | Session rate | | | Sessions | | | | | | Bytes | | Denied | | Errors | | | Warnings | | Serv | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | Last | In | Out | Req | Resp | Req | Conn | Resp | Retr | Redis | Status | LastChk | Wght | Act |
| server1 | 0 | 0 | - | 1 | 30 | | 30 | 30 | 30 | 30 | 30 | 1s | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 | 15s UP | L4OK in 0ms | 1 | Y |
| server2 | 0 | 0 | - | 1 | 30 | | 30 | 30 | 30 | 30 | 30 | 1s | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 | 15s UP | L4OK in 0ms | 1 | Y |
| server3 | 0 | 0 | - | 1 | 30 | | 30 | 30 | 30 | 30 | 30 | 1s | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 | 15s UP | L4OK in 0ms | 1 | Y |
| server4 | 0 | 0 | - | 1 | 30 | | 30 | 30 | 30 | 30 | 30 | 1s | 0 | 0 | | 0 | | 0 | 0 | 0 | 0 | 15s UP | L4OK in 0ms | 1 | Y |
| Backend | 1 | 1 | | 4 | 121 | | 121 | 121 | 50 | 121 | 120 | 1s | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 15s UP | | 4 | 4 |



HAProxy version 2.0.13-2, released 2020/04/01

*Statistics Report for pid 43747*

> General process information

**pid** = 43747 (process #1, nbproc = 1, nbthread = 1)
**uptime** = 0d 0h38m48s
**system limits:** memmax = unlimited; ulimit-n = 1023
**maxsock** = 1023; **maxconn** = 492; **maxpipes** = 0
current conns = 62; current pipes = 0/0; conn rate = 0/sec; bit rate = 74.503 kbps
Running tasks: 1/196; idle = 100 %

| | | | |
|---|---|---|---|
| 🟩 | active UP | 🟦 | backup UP |
| 🟨 | active UP, going down | 🟪 | backup UP, going down |
| 🟧 | active DOWN, going up | 🟪 | backup DOWN, going up |
| 🟥 | active or backup DOWN | ⬜ | not checked |
| 🟫 | active or backup DOWN for maintenance (MAINT) | | |
| 🟦 | active or backup SOFT STOPPED for maintenance | | |

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

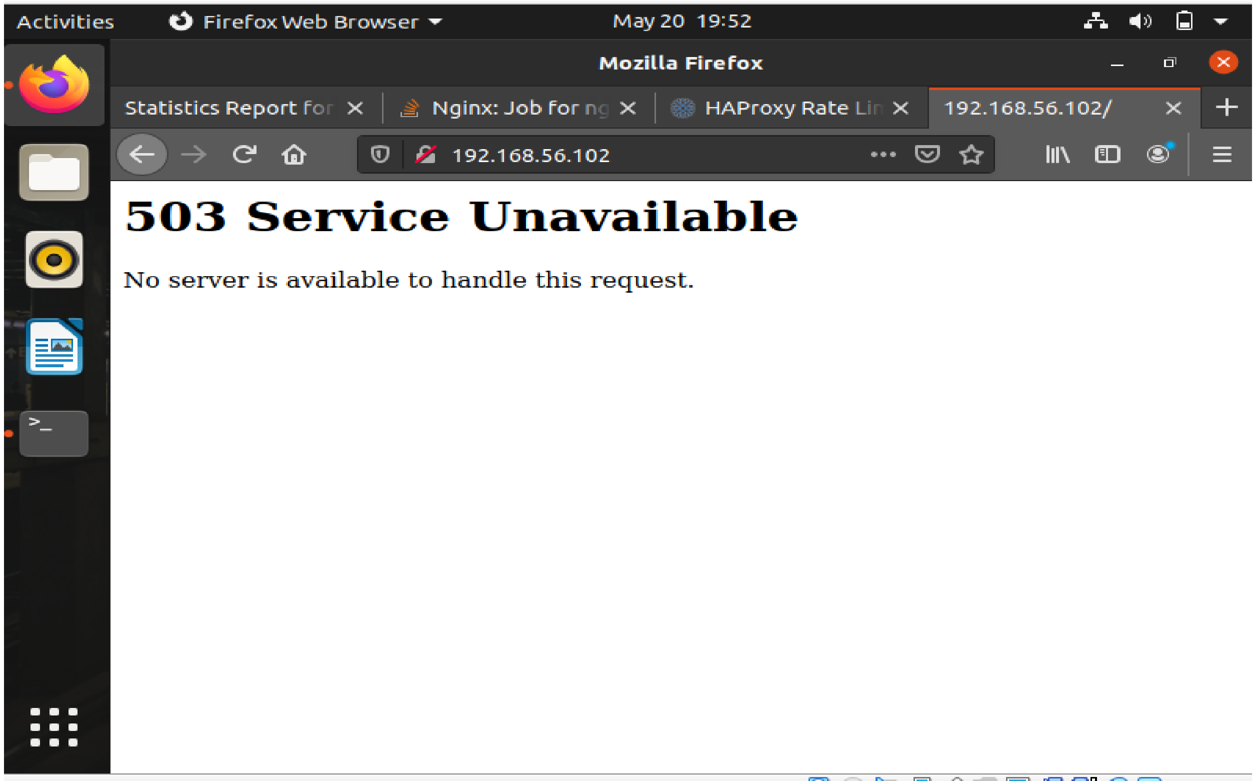
**Display option:**
- Scope :
- Hide 'DO
- Refresh r
- CSV expo

http_front

| | Queue | | | Session rate | | | Sessions | | | | | | Bytes | | Denied | | Errors | | | Warnings | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | Last | In | Out | Req | Resp | Req | Conn | Resp | Retr | Redis | Status | LastChk | Wght |
| Frontend | | | | 0 | 61 | - | 62 | 63 | 492 | 553 | | | 156 187 | 248 960 | 0 | 0 | 0 | | | | | OPEN | | |

http_back

| | Queue | | | Session rate | | | Sessions | | | | | | Bytes | | Denied | | Errors | | | Warnings | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | Last | In | Out | Req | Resp | Req | Conn | Resp | Retr | Redis | Status | La |
| pf.com.pk | 0 | 0 | - | 0 | 30 | | 30 | 30 | 30 | 270 | 269 | 4s | 75 180 | 49 200 | | 0 | | 0 | 0 | 0 | 0 | 38m48s UP | L4OK |
| api.pf.com.pk | 0 | 0 | - | 0 | 30 | | 30 | 30 | 30 | 270 | 269 | 4s | 75 180 | 49 200 | | 0 | | 0 | 0 | 0 | 0 | 38m48s UP | L4OK |
| Backend | 1 | 3 | | 0 | 61 | | 61 | 63 | 50 | 550 | 538 | 4s | 153 171 | 100 389 | 0 | 0 | | 9 | 0 | 0 | 0 | 38m48s UP | |



# 503 Service Unavailable

No server is available to handle this request.

**Can we host multiple application servers for load balancing as well?**

Yes we can host multiple servers on HA proxy for load balancing as in the above given configuration