Linux Commands

- CronJobs
- File and Folder
- Process
- Disk Management
- Install Docker on Ubuntu
- Enable IPv6 on Ubuntu
- Find Unidentified Web Service on a Server
- Calculate Size of a Folder
- Git Pull & Merge from Forked Repository
- Running a Web Server with PHP
- Zip Files and Folders
- Unzip .tar.gz File

CronJobs

Command Format

```
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

Multiple CronJobs for the Same User

Steps to Set Multiple Cron Jobs for the Same User

1. Edit the User's Crontab File:

```
crontab -e
```

2. Add Multiple Cron Jobs: Simply add each job on a new line. For example:

```
# Run script1.sh every hour
0 * * * * /path/to/script1.sh

# Run script2.sh every day at 2:30 AM
30 2 * * * /path/to/script2.sh

# Run script3.sh every Monday at 8:00 AM
0 8 * * 1 /path/to/script3.sh
```

Get All CronJobs from Each Users

```
getent passwd | cut -d: -f1 | while read user; do echo "Cron jobs for user: $user"; sudo crontab -l -u "$user" 2>/dev/null; done
```

File and Folder

Show file contents from tail with specific line number limit:

```
tail -n 200 <file>
```

Show file contents from head with specific line number limit:

```
head -n 200 <file>
```

Show only modified time of a file:

```
stat -c %y filename
```

Human-readable format:

```
stat -c %y filename | cut -d'.' -f1
```

File and directory size with sorting:

```
du -ah <folder-name>/ | sort -h
```

Process

Sort process by memory usage

```
ps aux --sort=-%mem | head -n 30
```

Sort process by CPU usage

```
ps aux --sort=-%cpu | head -n 30
```

Disk Management

Disk device list

```
lsblk
```

Show disk detail:

```
lsblk -o NAME,UUID,SIZE,TYPE,MOUNTPOINT
```

Detect the disk

```
sudo fdisk -l /dev/<device-name>
```

Create a filesystem / format partition

```
sudo mkfs.ext4 /dev/<device-name>
```

Resize Filesystem (if already partitioned)

```
sudo resize2fs /dev/sdb1
```

Display storage size

```
df -h
```

Mount disk

```
mount /dev/<device-name> /mnt/<path>
```

Unmount disk

```
umount /mnt/<path>
```

Persist the mount on reboot

```
nano /etc/fstab
```

```
/dev/<device-name> /mnt/<path> ext4 defaults 0 2
```

Install Docker on Ubuntu

To install Docker on Ubuntu, you can follow these steps:

Step 1: Update Package Index

First, update the package index to ensure you install the latest versions of Docker and its dependencies:

```
sudo apt update
```

Step 2: Install Dependencies

Install the packages necessary to allow apt to use a repository over HTTPS:

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

Step 3: Add Docker's Official GPG Key

Add Docker's official GPG key to your system:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Step 4: Add Docker Repository

Add the Docker repository to your APT sources:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

Step 5: Install Docker Engine

Update the package index again, and install Docker:

```
sudo apt update
sudo apt install docker-ce
```

Step 6: Verify Docker Installation

Check that Docker Engine is installed correctly by running the `hello-world` container:

```
sudo docker run hello-world
```

This command downloads a test image and runs it in a container. If Docker is set up correctly, you should see a message confirming that Docker is working.

Step 7: Manage Docker as a Non-Root User (Optional)

If you want to run Docker commands without using `sudo`, add your user to the `docker` group:

```
sudo usermod -aG docker ${USER}
```

Log out and back in, or run `newgrp docker`, to activate the changes.

Step 8: Start and Automate Docker

Start Docker and enable it to start on boot:

```
sudo systemctl start docker
sudo systemctl enable docker
```

Step 9: Verify Docker Version (Optional)

To verify the installed Docker version, you can use:

```
docker --version
```

That's it! Docker should now be installed and ready to use on your Ubuntu system.

Enable IPv6 on Ubuntu

Activating IPv6 on your system typically involves ensuring that your network infrastructure and configuration support IPv6 connectivity. Here's a general guide on how to activate and configure IPv6 on Linux, which should help you enable IPv6 connectivity for your Docker environment:

Enable IPv6 on Linux

1. **Check Current IPv6 Support**

First, verify if IPv6 is already enabled on your Linux system:

```
cat /proc/sys/net/ipv6/conf/all/disable_ipv6
```

If the output is `0`, IPv6 is enabled. If it's `1`, IPv6 is disabled.

2. **Enable IPv6 Temporarily**

To enable IPv6 temporarily (until next reboot):

```
sudo sysctl -w net.ipv6.conf.all.disable_ipv6=0
```

3. **Enable IPv6 Permanently**

To enable IPv6 permanently, edit the `/etc/sysctl.conf` file:

```
sudo nano /etc/sysctl.conf
```

Add or uncomment the following line to enable IPv6:

```
net.ipv6.conf.all.disable_ipv6 = 0
```

Save and close the file. Then apply the changes:

```
sudo sysctl -p
```

4. **Check IPv6 Configuration**

Verify that IPv6 has been enabled:

```
ip a
```

You should see IPv6 addresses listed along with IPv4 addresses for your network interfaces.

Docker IPv6 Configuration

1. **Docker Daemon Configuration**

Edit Docker daemon configuration (`/etc/docker/daemon.json` or `/etc/default/docker`) to enable IPv6 support:

```
{
  "ipv6": true,
  "fixed-cidr-v6": "<IPv6 subnet>",
  "default-address-pools":
  [
    {"base":"<IPv6 subnet>", "size":64}
  ]
}
```

Replace `<IPv6 subnet>` with your desired IPv6 subnet allocation. For example:

- `2001:db8:abcd::/64`

2. **Restart Docker**

After making changes to Docker daemon configuration, restart the Docker service:

```
sudo systemctl restart docker
```

3. **Verify Docker IPv6 Support**

Check Docker network settings to confirm IPv6 support:

```
docker network inspect bridge
```

Ensure that IPv6 configuration (`"EnableIPv6": true`) is properly set.

Final Steps

- **Test Connectivity:** Verify IPv6 connectivity from your Docker containers using tools like `ping6` or `traceroute6`.
- **Adjust Firewall Rules:** Ensure that firewall rules (e.g., `ufw`, `iptables`) allow IPv6 traffic, especially if you're using firewall rules that might block certain types of traffic.

By following these steps, you should be able to activate and configure IPv6 on your Linux system and Docker environment, enabling IPv6 connectivity for applications and services running within Docker containers. Adjust configurations as per your specific network setup and security requirements.

Find Unidentified Web Service on a Server

If you're on a server console and need to update the SSL certificate but don't know which web server service is running, here's a systematic approach you can follow:

Step-by-Step Approach:

1. **Identify the Web Server Service**:
   - Check if there are any web server processes running. You can do this by listing active processes:

     ```
     ps aux | grep -i 'apache\|http\|nginx\|lighttpd\|tomcat'
     ```

     This command searches for common web server processes (`apache`, `httpd`, `nginx`, `lighttpd`, `tomcat`). It will show you if any of these services are currently running.
2. **Check Listening Ports**:
   - Determine if there are any services listening on standard web ports (80 for HTTP, 443 for HTTPS):

     ```
     sudo netstat -tuln | grep -E ':80|:443'
     ```

     This command will list all services listening on ports 80 (HTTP) and 443 (HTTPS). Note the process name or PID associated with these ports.
3. **Inspect Running Services**:
   - Look for configuration files or directories that indicate the presence of a web server. Common locations include:
     - `/etc/apache2/` (for Apache HTTP Server)
     - `/etc/nginx/` (for Nginx)
     - `/etc/httpd/` or `/etc/httpd/conf/` (for Apache HTTP Server on some distributions)
     - `/etc/lighttpd/` (for Lighttpd)
     - `/opt/tomcat/` (for Apache Tomcat)
4. **Examine SSL/TLS Configuration Files**:
   - Once you've identified the web server service, check its configuration files to confirm where SSL certificates are stored and configured:
     - For Apache HTTP Server, look in `httpd.conf` or `ssl.conf`.
     - For Nginx, check `nginx.conf` and any included configuration files in `/etc/nginx/sites-available/`.
     - Adjust the paths based on your specific server setup.

5. **Update SSL Certificates**:
   - Copy your SSL certificates (`ca_bundle.crt`, `certificate.crt`, `private.key`) to the appropriate directory based on your findings from the configuration files.
   - Update the SSL configuration to point to the new certificates.
6. **Restart the Web Server**:
   - After updating the SSL configuration and placing the certificates, restart the web server service to apply changes:

     ```
     sudo systemctl restart apache2     # For Apache HTTP Server
     sudo systemctl restart nginx       # For Nginx
     ```

7. **Verify SSL Installation**:
   - Use online tools like SSL Labs (https://www.ssllabs.com/ssltest/) or command-line tools (`openssl` commands) to verify that your SSL certificate installation is correct and secure.

By following these steps, you should be able to identify the web server service running on your server console, update the SSL certificates accordingly, and ensure that your website or application is secured with the updated SSL/TLS configuration.

Calculate Size of a Folder

```
sudo du -h -d 1 /<path>
```

Git Pull & Merge from Forked Repository

Checkout

From your project repository, check out a new branch and test the changes.

```
git fetch -u https://git.introvesia.com/ahmjw90/landing-page main:ahmjw90-main
git checkout ahmjw90-main
```

Merge

Merge the changes and update on Gitea.

```
git checkout main
git merge --no-ff ahmjw90-main
git push origin main
```

Running a Web Server with PHP

If you want to run a PHP server directly without Docker, you can use PHP's built-in web server, which is convenient for development purposes. Here’s how you can set it up to run on port 8080:

Running PHP Server Locally

1. **Navigate to Your PHP Application Directory**:
   Open a terminal or command prompt and navigate to the directory where your PHP files are located. For example:

   ```
   cd /path/to/your/php-app
   ```

2. **Start PHP Built-In Web Server**:
   Use the `php` command with the `-S` flag to start the PHP built-in web server:

   ```
   php -S localhost:8080
   ```

   This command starts a web server locally on port 8080, serving files from the current directory (`/path/to/your/php-app`).
3. **Access Your PHP Application**:
   Once the server is running, open your web browser and go to:

   ```
   http://localhost:8080
   ```

   You should see your PHP application running.
4. **Develop and Test**:
   You can now develop and test your PHP application locally. Any changes you make to PHP files will be immediately reflected in the running server.

Additional Notes:

- **PHP Built-In Server**: This is suitable for development purposes but may not be as robust or feature-rich as a full web server like Apache or Nginx in a production environment.
- **Routing**: PHP's built-in server supports PHP scripts directly and basic routing. For more complex routing or features, consider using a framework like Laravel, Symfony, or a micro-framework like Slim or Lumen.

- **Stopping the Server**: To stop the PHP built-in server, go back to your terminal or command prompt and press `Ctrl + C`.

This method allows you to run PHP applications locally without Docker, leveraging PHP's built-in capabilities for development and testing. Adjust paths and configurations based on your specific project setup and requirements.

Zip Files and Folders

To compress an entire folder into a .zip file, you can use several methods depending on your operating system. Here's how to do it using both command-line interfaces (CLI) and graphical user interfaces (GUI):

Command Line (CLI) Method

Using `zip` Command (Linux/MacOS/Windows with Git Bash or similar)

1. **Navigate to the Directory Containing the Folder**: Open a terminal or command prompt and navigate (`cd`) to the directory where the folder you want to compress is located.
2. **Run the Zip Command**: Use the `zip` command to create a .zip archive of the folder. For example, to compress a folder named `myfolder` into a single archive named `myfolder.zip`:

   ```
   zip -r myfolder.zip myfolder
   ```

   - `-r`: Recursively includes all files and subdirectories within `myfolder`.
   - `myfolder.zip`: Name of the output zip file.
   - `myfolder`: Name of the folder to compress.

   Replace `myfolder` with the actual name of your folder.
3. **Verify the Zip File**: Once the command completes, you should see `myfolder.zip` in the current directory.

Using `tar` Command (Linux/MacOS)

Alternatively, you can use `tar` combined with `gzip` for compression:

```
tar -czvf myfolder.tar.gz myfolder
```

- `-c`: Create a new archive.
- `-z`: Compress the archive using gzip.
- `-v`: Verbose mode (optional, shows progress).
- `-f myfolder.tar.gz`: Name of the output tar.gz file.
- `myfolder`: Name of the folder to compress.

This command creates a compressed `myfolder.tar.gz` archive of `myfolder`.

Graphical User Interface (GUI) Methods

Windows

1. **Navigate to the Folder**: Navigate to the folder you want to compress using Windows File Explorer.
2. **Select Files and Folders**: Select the folder (and its contents) you want to compress.
3. **Right-click and Select "Send to" > "Compressed (zipped) folder"**: This action creates a .zip file with the same name as the selected folder in the same directory.

macOS

1. **Navigate to the Folder**: Navigate to the folder you want to compress using Finder.
2. **Select Files and Folders**: Select the folder (and its contents) you want to compress.
3. **Right-click and Select "Compress [folder name]"**: This action creates a .zip file with the same name as the selected folder in the same directory.

Python Script (Cross-platform)

If you prefer using Python, you can create a script to compress a folder programmatically:

```
import zipfile
import os


def zip_folder(folder_path, output_path):
    with zipfile.ZipFile(output_path, 'w', zipfile.ZIP_DEFLATED) as zipf:
        for root, dirs, files in os.walk(folder_path):
            for file in files:
                zipf.write(os.path.join(root, file), os.path.relpath(os.path.join(root, file), os.path.join(folder_path, '..')))


if __name__ == '__main__':
    folder_to_zip = '/path/to/your/folder'
    output_zip_file = 'myfolder.zip'
    zip_folder(folder_to_zip, output_zip_file)
```

- Replace `/path/to/your/folder` with the path to the folder you want to compress.
- `myfolder.zip` will be the name of the output zip file.

Notes:

- Adjust paths and file names according to your specific requirements.
- Command-line tools (`zip`, `tar`) provide more flexibility and are suitable for automation or integration into scripts.
- GUI methods are convenient for one-off tasks and may vary slightly based on your operating system version.

Using these methods, you can easily compress entire folders into .zip archives based on your preferred environment and tools.

Unzip .tar.gz File

To unzip `.tar.gz` files, you typically use the `tar` command-line utility, which can handle both tar archives and gzip compression. Here's how you can unzip a `.tar.gz` file:

Unzip `.tar.gz` File

1. **Navigate to the Directory Containing the File**: Open a terminal or command prompt and navigate (`cd`) to the directory where your `.tar.gz` file is located.
2. **Run the Tar Command**: Use the `tar` command with the `-xzf` options to extract the contents of the `.tar.gz` file:

   ```
   tar -xzf yourfile.tar.gz
   ```

   - `-x`: Extract files from an archive.
   - `-z`: Filter the archive through gzip to decompress it.
   - `-f yourfile.tar.gz`: Specify the filename of the `.tar.gz` file you want to extract.

   Replace `yourfile.tar.gz` with the actual name of your `.tar.gz` file.
3. **Extract to a Specific Directory (Optional)**: You can specify a target directory where you want the contents to be extracted. For example, to extract into a directory named `myfolder`:

   ```
   tar -xzf yourfile.tar.gz -C myfolder
   ```

   - `-C myfolder`: Extracts the contents into the `myfolder` directory.

Additional Notes:

- **Preserving Permissions**: Use `-p` option with `tar` (`tar -xzf yourfile.tar.gz -p`) to preserve file permissions and ownership while extracting.
- **List Contents**: You can list the contents of a `.tar.gz` file without extracting it using `-t` option (`tar -tf yourfile.tar.gz`).
- **Handling Errors**: If encountering errors, ensure the `.tar.gz` file is correctly located and accessible, and verify permissions if necessary.

By using the `tar` command with appropriate options (`-xzf`), you can effectively unzip and extract contents from `.tar.gz` files in Unix-like systems (Linux, macOS, etc.). Adjust commands as needed based on your specific file names and directory structures.