STARTER

Onboarding guide

Table of Contents

Install Ubuntu desktop

1. Overview

What you'll learn:

In this tutorial, we will guide you through the steps required to install Ubuntu Desktop on your laptop.

What you'll need:

- A laptop or PC
- A flash drive of 12GB or above

2. Download an Ubuntu Image

You can download an Ubuntu image here.

3. Create a Bootable USB stick

To install Ubuntu Desktop, you need to write your downloaded ISO to a USB stick to create the installation media. This is not the same as copying the ISO, and requires some bespoke software.

For this tutorial, we'll use balenaEtcher, as it runs on Linux, Windows and Mac OS. Choose the version that corresponds to your current operating system, download and install the tool.

Select your downloaded ISO, choose your USB flash drive, and then click Flash! to install your image.

4. Boot from USB flash drive

Insert the USB flash drive into the laptop or PC you want to use to install Ubuntu and boot or restart the device. It should recognise the installation media automatically. If not, try holding F12 during startup and selecting the USB device from the system-specific boot menu.

You should now see the welcome screen inviting you to either try or install Ubuntu.

To proceed, click Install Ubuntu.

You will be asked to select your keyboard layout. Once you've chosen one, click Continue.

5. Installation Setup

Next, you will be prompted to choose between the Normal installation and Minimal installation options. Choose Normal installation. In Other options check both boxes and ensure you are able to remain connected to the internet throughout the installation.

6. Drive Management

This screen allows you to configure your installation. If you would like Ubuntu to be the only operating system on your device, select Erase disk and install Ubuntu.

7. Choose your Location

Select your location and timezone from the map screen and click Continue. This information will be detected automatically if you are connected to the internet.

8. Create Your Login Details

On this screen, you will be prompted to enter your name and the name of your computer as it will appear on the network. Finally, you will create a username and a strong password.

You can choose to log in automatically or require a password. If you are using your device whilst traveling, it's recommended to keep automatic login disabled.

9. Complete the Installation

Now sit back and enjoy the slideshow as Ubuntu installs in the background!

Once the installation has completed, you will be prompted to restart your machine. Click Restart Now.

When you restart, you will be prompted to remove your USB flash drive from the device. Once you've done this, press ENTER.

Main Tools

*All of the below tools can be downloaded using the App Center

1. Browser

Downloading Google Chrome

Open your terminal either by using the Ctrl+Alt+T keyboard shortcut or by clicking on the terminal icon. Use wget to download the latest Google Chrome .deb package:

```
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
```

Installing Google Chrome

Installing packages on Ubuntu requires administrative privileges. Running the following command as a user with sudo privileges to install Chrome .deb package on your system:

```
sudo apt install ./google-chrome-stable_current_amd64.deb
```

When prompted, enter your user password, and the installation will start.

At this point, you have Chrome installed on your Ubuntu system.

2. Slack

Installing Slack as a Deb Package

Visit the Slack for Linux download page and download the latest Slack .deb package.

Once the download is complete, double-click the file, and the Ubuntu Software Center will open. To start the installation click on the “Install” button. You may be prompted to enter your user password.

3. Skype

Installing Skype with apt

Skype is available from the official Microsoft Apt repositories. To install it, follow the steps below:

● Open your terminal and download the latest Skype .deb package using the following wget command:

```
wget https://go.skype.com/skypeforlinux-64.deb
```

● Once the download is complete, install Skype by running the following command as a user with sudo privileges :

```
sudo apt install ./skypeforlinux-64.deb
```

Development Tools

1. PHPStorm

Standalone installation

● Download the tarball .tar.gz.

● Extract the tarball to a directory that supports file execution. For example, to extract it to the recommended /opt directory, run the following command:

```
sudo tar -xzf PhpStorm-*.tar.gz -C /opt
```

Execute the PhpStorm.sh script from the extracted directory to run PhpStorm.

2. Sublime

Install Sublime Text 3 via the official apt repository:

● Open terminal via Ctrl+Alt+T or by searching for “Terminal” from desktop app launcher. When it opens, run command to install the key:

```
wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key
add -
```

● Then add the apt repository via command:

```
echo "deb https://download.sublimetext.com/ apt/stable/" | sudo tee
/etc/apt/sources.list.d/sublime-text.list
```

● Finally check updates and install sublime-text by running commands:

```
sudo apt-get update
```

```
sudo apt-get install sublime-text
```

3. MySQL Workbench

Install MySQL

Step 1: Update/Upgrade Package Repository

```
sudo apt-get update
```

Step 2: Install MySQL

```
sudo apt-get install mysql-server
```

Step 3: Check if MySQL was successfully installed by running

```
mysql --version
```

Step 4: Secure your MySQL user account with password authentication by running the included security script:

```
sudo mysql_secure_installation utility
```

Enable UFW (uncomplicated firewall)

To enable UFW on your system, run:

```
sudo ufw enable
```

To enable MySQL in UFW run:

```
sudo ufw allow mysql
```

Start MySQL service

To start MySQL service run:

```
sudo systemctl start mysql
```

Install MySQL Workbench

Download configuration file from the apt repository: https://dev.mysql.com/downloads/repo/apt/

Check the downloaded configuration file in the particular directory. To do that, navigate into the Downloads using the following command and list files.

```
cd Downloads
```

```
ls
```

Use the following command to add MySQL repository URLs in the apt sources list so that you can install the software on your Ubuntu 20.04 system.

```
sudo apt install ./mysql-apt-config_0.8.22-1_all.deb
```

Terminal Tools

1. Terminator

Terminator is an alternative terminal for Linux that comes with a little additional features and functionality that you won't find in the default terminal application.

To get terminator installed on your system enter the following commands in the terminal (SHORTCUT: Ctrl+Alt+T);

```
sudo add-apt-repository ppa:gnome-terminator
sudo apt-get update
sudo apt-get install terminator
```

2. Vim

Vim is an advanced and highly configurable text editor built to enable efficient text editing from the command line interface. You can install it by running the following commands from the terminal:

```
sudo apt update
sudo apt-get install vim
```

3. Customize Terminal

The bashrc file in Linux is a configuration file containing configurations related to the system’s terminal.

```
cd ~
vim .bashrc
```

By adding the following code to the .bashrc file, we show the current checked out git branch

Comment these lines:

```
#if [ "$color_prompt" = yes ]; then
#
PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[0
1;34>
#else
#    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
#fi
#unset color_prompt force_color_prompt
```

And add these:

```
parse_git_branch() {
 git branch 2> /dev/null | sed -e '/^[^*]/d' -e 's/* \(.*\)/(\1)/'
}
if [ "$color_prompt" = yes ]; then

PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[0
1;34m\]\w\[\033[01;31m\] $(parse_git_branch)\[\033[00m\]\$ '
else
 PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w$(parse_git_branch)\$ '
fi
```

4. Git

Git is the world's most popular distributed version control system used by many open-source and commercial projects. It allows you to collaborate on projects with your fellow developers, keep track of your code changes, revert to previous stages, create branches , and more.

Install git by running the following commands:

```
sudo apt update
sudo apt install git
```

One of the first things you need to do after installing Git is to configure your git username and email address. Git associates your identity with every commit you make.

To set your global commit name and email address run the following commands:

```
git config --global user.name "Your Name"
git config --global user.email "youremail@yourdomain.com"
```

5. Docker

Docker is an open-source containerization platform that allows you to quickly build, test, and deploy applications as portable containers that can run virtually anywhere. A container represents a runtime for a single application and includes everything the software needs to run.

Install Docker

Step 1: Update the packages index and install the dependencies necessary to add a new HTTPS repository:

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

Step 2: Download the Docker GPG key and place it in a specific directory:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg
```

Step 3: Add the Docker APT repository to your system, referencing the key:

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

Or(more secure way):

```
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
```

Step 4: Now that the Docker repository is enabled, you can install any Docker version that is available in the repositories. To install the latest version of Docker, run the commands below:

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io
```

Step 5: Once the installation is completed, the Docker service will start automatically. You can verify it by typing:

```
sudo systemctl status docker
```

6. Docker Compose

Docker Compose is a tool that allows you to run multi-container application environments based on definitions set in a YAML file. It uses service definitions to build fully customizable environments with multiple containers that can share networks and data volumes.

Install Docker Compose

Step 1: To make sure we obtain the most updated stable version of Docker Compose, we'll download this software from its official Github repository. First, confirm the latest version available in their releases page. At the time of this writing, the most current stable version is v2.19.0. The following command will download the 2.19.0 release and save the executable file at /usr/local/bin/docker-compose, which will make this software globally accessible as docker-compose:

```
sudo curl -L
"https://github.com/docker/compose/releases/download/v2.19.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Step 2: Next, set the correct permissions so that the docker-compose command is executable:

```
sudo chmod +x /usr/local/bin/docker-compose
```

Step 3: To verify that the installation was successful, you can run:

```
docker-compose --version
```

7. SSH Keys

Secure Shell (SSH) is a network protocol for creating a secure connection between a client and a server. With SSH, you can run commands on remote machines, create tunnels, forward ports, and more.

To generate a new 4096 bits SSH key pair with your email address as a comment, run:

```
ssh-keygen -t rsa -b 4096 -C "your_email@domain.com"
```

Starter Project Setup

1. Development environment

Clone the git repository

First of all create a folder using your terminal by running the following commands:

```
cd ~
mkdir development
cd  development
mkdir repositories
cd repositories
```

Now we have created the ~/development/repositories folder where all of our projects will reside. Next we clone the Development Environment repository (**infrastructure/dev_env**):

```
git clone https://github.com/NikolovskiRatko/starter-architecture.git
starter-architecture
```

Configure the environment variables

Now that we have cloned the Docker Compose project, we shall take some time to properly configure the environment variables. A good start would be to copy the .env.sample file, to a file named .env and edit it accordingly. In the dev_env folder in the terminal:

```
cp .env.sample .env
```

The first environment variables to set are the *DOCUMENT_ROOT* and *STARTER_NODE_ROOT* which is essentially where the starter project will reside. Given we are going to use the same folder structure for convenience, you should only change the system user to your system user.

HINT: The ***pwd*** command stands for print working directory. It is one of the most basic and frequently used commands in Linux. When invoked the command prints the complete path of the current working directory.

Most of the remaining variables should stay the same, as they resemble the proper folder. The only variable that is set to a custom port is the *HOST_MACHINE_MYSQL_PORT* which is set to map a different port since MySQL will run locally on our system in order to use MySQL Workbench.

Create empty folders

Stention need to be created manually. In the dev_env folder run:

```
mkdir data
mkdir logs
```

Build and start the docker containers

In the dev_env folder from the terminal first run:

```
docker-compose build
```

This might take a few minutes, after it is done run:

```
docker-compose up
```

At this stage the docker containers should be running successfully and are ready to be used to build and serve the Starter web application.

Test docker containers

In order to check if the docker containers are running properly, run:

```
docker ps
```

This command is used to list the running containers. Also you may open a bash connection to the app docker container by running:

```
sudo docker exec -it app /bin/bash
```

Useful commands

In the process of testing the dev environment here are some useful commands:

To stop all running docker containers

```
docker stop $(docker ps -a -q)
```

To clear all cache from the broken docker-compose version

```
docker system prune -a
```

2. Clone web application

Clone the git repository

Make sure that the terminal is in the repositories folder that we previously created:

```
cd ~
cd development/repositories
```

Next we clone the Web Application repository:

```
git clone https://github.com/NikolovskiRatko/starter-architecture.git
starter-architecture
```

Configure the environment variables

Now that we have cloned the Web Application project, we shall take some time to properly configure the environment variables. A good start would be to copy the .env.example file, to a file named .env and edit it accordingly. In the starter folder in the terminal:

```
cp .env.example .env
```

Create a MySQL Workbench configuration

Open MySQL Workbench and create a new server connection with the following parameters:

Hostname: 127.0.0.1

Port: 33069 (or the port set in the .env file)

User: root

Password: password

After creating it, connect to it and create a new database named ***starter***.

Set proper permissions

In the ***starter*** folder run the following commands to set the local machines permissions:

```
sudo chown -R www-data. . && sudo setfacl -R -m u:$USER:rwx .
```

3. Build the web application

Install PHP dependencies

In the terminal we open a bash connection to the app docker container

```
sudo docker exec -it app /bin/bash
```

In the opened terminal session in the docker container now we can run the following series of commands:

```
composer install && php artisan config:clear && php artisan view:clear && php
artisan route:clear && composer dump-autoload && php artisan cache:clear &&
php artisan vue-i18:generate
```

Install Javascript dependencies

In the terminal we open a bash connection to the node docker container

```
sudo docker exec -it node /bin/bash
```

In the opened terminal session in the docker container now we can run the following series of commands:

```
yarn && yarn watch
```

Keep this tab in the terminal open so that you can track the javascript build when changes are made in the code editor.

Run database migrations and seeds

For the database migrations we run the following commands in the app docker container:

```
php artisan migrate:fresh && php artisan db:seed
```

Edit hosts file

To edit the local systems host file in the terminal run:

```
sudo vim /etc/hosts
```

Add the following line:

```
127.0.0.1    starter.test
```

Test in browser

Finally the Starter web application can be tested in the browser. Open Chrome and visit the following URL:

http://starter.test/

Starter Development Workflow

1. Folder structure

Front-end

All the frontend dependencies (resources/assets)

Sass code (resources/assets/sass)

Vue code (resources/assets/vue)

Main Vue instance file (resources/assets/vue/app.ts)

Global JS settings (resources/assets/vue/bootstrap.js)

All encompassing App component (resources/assets/vue/App.vue)

Back-end

Routes for api (routes)

Code per feature (app/Applications)

2. Git Workflow

Main branch structure

Development branch with tools and code used for developer testing (dev)

QA branch deployed on QA server instance for in house testing (qa)

Staging branch deployed on staging server instance for client testing (staging)

Prod branch for production value code (prod)

Create a new branch for each feature, task or bug

Naming convention example: git checkout -b task/21465-some-tasks-name-here

Merging

When ready locally and developer tested merge to development branch (dev), when QA approved merge to staging.

3. Task Workflow

Front-end (VueJs Components)

Add the route (resources/assets/vue/router/index.ts)

Create the view that will be the wrapper for the component (resources/assets/vue/views)

Create and add all the components for that feature (resources/assets/vue/features)

Define your variables, try to use types (resources/assets/vue/typings)

For reusable components use (resources/assets/vue/components)

Add constructor to component where initial values of variables are set

Back-end (Laravel API)

Basics of 2 layered architecture

API routes

Business logic layer

Data access layer

Providers

Dependency injection

Combined

Localization: Static translations are input in associative arrays (resources/lang/en) and run 'php artisan vue-i18:generate'

In VueJs the component is resources/assets/vue/components/Translation/Locales.vue

4. Debugging Workflow

Front-end debugging

Vue developers tool for VueJs components

Conventional browser debugging with breakpoints

Back-end debugging

Configure Xdebug in PHPStorm locally

```
sudo apt-get install php7.4-xdebug
```

1. **Navigate to /etc/php/*version*/mods-available/**
2. **sudo nano xdebug.ini and add following lines**

[xdebug]

xdebug.remote_enable=1

xdebug.remote_autostart=1

xdebug.remote_port=9000

xdebug.idekey=PHPSTORM

3. **Test if xdebug is loaded successfully using php -v**
4. **Go in PHPStorm and press CTRL+ALT+S to open settings**

-Navigate to Languages & Frameworks

-Navigate to PHP

-At CLI Interpreter press the three dots and click + on top left corner and add from /bin/php

Using Xdebug with PHPStorm

Personal Portfolio website

1. Extend the Starter codebase

…

2. Integrate HTML/CSS layout

…

3. Add CMS functionality in Admin Panel

…

Starter Server Provisioning

1. Ansible Project

Clone the git repository

First of all cd in the terminal to the repositories folder previously created:

```
cd ~
cd development
cd repositories
```

Now that we are in the correct folder where all our projects reside, we clone the Ansible Environment Provisioning repository (**infrastructure/host**):

```
git clone https://github.com/NikolovskiRatko/starter-architecture.git
starter-architecture
```

Create a

First of al