An astronaut in space with planets and stars

Description automatically generated

A yellow line drawing of a hand holding a megaphone

Description automatically generated

STEPS PODCAST

***Creating A Website with docker***

***Progressive Pull Org ®***

Contents

[Installed Software 3](#_Toc185945788)

[Introduction 3](#_Toc185945789)

[Git 3](#_Toc185945790)

[Check Git Version 4](#_Toc185945791)

[Source Tree 5](#_Toc185945792)

[NodeJS and NPM 5](#_Toc185945793)

[Using NPM to download jQuery 6](#_Toc185945794)

[GitHub Desktop 10](#_Toc185945795)

[Visual Studio Code 12](#_Toc185945796)

[Using the Debugger for JavaScript Application 12](#_Toc185945797)

[Docker Desktop 15](#_Toc185945798)

[Docker 16](#_Toc185945799)

[Introduction 16](#_Toc185945800)

[Docker File 16](#_Toc185945801)

[Docker Image 18](#_Toc185945802)

[Docker Container 18](#_Toc185945803)

[Docker Compose 19](#_Toc185945804)

[Create the Image 19](#_Toc185945805)

[Steps to Building Image 20](#_Toc185945806)

[Starting the Docker Container 21](#_Toc185945807)

[Steps to Starting the Docker Container 22](#_Toc185945808)

[Command Line 24](#_Toc185945809)

[File Structure 25](#_Toc185945810)

[Steps to Starting the Docker Compose 26](#_Toc185945811)

[RUN vs. CMD in Docker File 27](#_Toc185945812)

[Using the RUN Instruction 27](#_Toc185945813)

[Using the CMD Instruction 27](#_Toc185945814)

[Key Differences 27](#_Toc185945815)

[References 27](#_Toc185945816)

[Markdown 27](#_Toc185945817)

[Linux Commands 28](#_Toc185945818)

[GIT 28](#_Toc185945819)

[Source Tree 28](#_Toc185945820)

[NodeJS and NPM 28](#_Toc185945821)

[GitHub Desktop 28](#_Toc185945822)

[Visual Studio Code 28](#_Toc185945823)

[Docker Desktop 28](#_Toc185945824)

[Docker 28](#_Toc185945825)

# Installed Software

## Introduction

I’m **not going to walk** you through the installation process *for this software*. You can find plenty of tutorials on YouTube. Instead, I’ll provide you with a list of the software you need for your machine.

## Git

A screenshot of a computer

Description automatically generated

One essential software package to **install on your computer is Git—not GitHub or GitLab**. Git is a distributed version control system, extensively used in software development. Think of it as the backbone of collaborative coding. When you're working on a project with multiple people, Git tracks all the changes made to the code, who made them, and when. This allows multiple developers to work on the same project at the same time without interfering with each other's work.

While **Git, GitHub, and GitLab are all related** to version control and managing code, they **serve different purposes**. After installing Git on your computer, you can access it with command-line window. GitHub is a web-based platform that uses Git for version control. GitLab is a web-based DevOps lifecycle tool that provides a Git repository manager.

<https://git-scm.com/>

### Check Git Version

To verify if Git is successfully installed on your computer, you can **check the Git version number**. To open the Command Prompt Window, **click the Windows Start button**.

A screenshot of a computer

Description automatically generated

That will open the Start menu. At the top of the menu, there's a **search field where you can enter 'CMD'** to open the Command Prompt window.

A screenshot of a computer

Description automatically generated

To see the version number**, type 'git -- version'** in the Command Prompt window.

A screenshot of a computer program

Description automatically generated

## Source Tree

A screenshot of a computer

Description automatically generated

**SourceTree is a free Git client** developed by Atlassian. It provides a graphical user interface (GUI) for managing Git repositories, making it easier for both beginners and experienced developers to interact with Git without needing to use the command line

SourceTree simplifies how you interact with your Git repositories so you can focus on coding. Visualize and manage your repositories through SourceTree’s simple Git GUI.

<https://www.sourcetreeapp.com/>

## NodeJS and NPM

A screenshot of a computer

Description automatically generated

Another important software package to install is Node.js, an open-source, cross-platform runtime environment that **allows you to execute JavaScript code** outside of a web browser. It's built on Chrome's V8 JavaScript engine, which enables developers to use JavaScript for server-side scripting.

Node.js comes with NPM (Node Package Manager), the **standard package manager for Node.js.** The NPM webpage, npmjs.com, is the official site where you can browse packages, read documentation, and find general information about NPM. The site also offers resources for both free and Pro users, including various tools for JavaScript development.

<https://nodejs.org/en>

### Using NPM to download jQuery

I'll **demonstrate how to download a JavaScript** library using jQuery as an example. I've chosen jQuery because it's commonly used and suitable for demonstration purposes.

1. Open the work-space folder:
   * I was previously working in a directory called 'DIR' so to **change the directory to 'work-space',** do the following. **Select 'Open Folder'** and choose the desired folder.

A screenshot of a computer

Description automatically generated

1. Open a terminal window:
   * Launch a new terminal. By default, it **opens in PowerShell**.

A screenshot of a computer

Description automatically generated

* + If you prefer, you can **switch to Bash**, provided you have Git Bash installed.

A screenshot of a computer

Description automatically generated

1. Create a package.json file:
   * The package.json file will keep a **list of all the JavaScript libraries** you've installed.

A screenshot of a computer

Description automatically generated

* + Run **npm init** and follow the prompts.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

* + **You can accept the default values by pressing enter** than type 'yes'.

A screenshot of a computer

Description automatically generated

* + You can **open the package.json** file to view its contents.

1. Install jQuery:

A screenshot of a computer

Description automatically generated

* + Visit the npm manage site **and find the command for installing jQuery**.

A screenshot of a computer

Description automatically generated

* + Copy the ***npm I jquery*** command, paste it into the terminal, and run it to install the library.

A screenshot of a computer program

Description automatically generated

* + The library will **be added to the dependencies** in your package.json file.

1. Remove Unwanted Library:

A computer error message

Description automatically generated

* + If you mistakenly add the wrong files, **delete the unwanted entries** from your package.json file.

A close-up of a number

Description automatically generated

* + **Run npm install** to update the dependencies.

A screenshot of a computer

Description automatically generated

* + Perform a cleanup by **letting the package manager handle deletions**, then manually remove any leftover folders if necessary.

## GitHub Desktop

A screenshot of a computer

Description automatically generated

GitHub Desktop is a free, open-source application created to streamline your development workflow. It enables you to interact with **GitHub repositories** **using a graphical user interface** (GUI) rather than the command line or a web browser. You can decide which files to commit to your GitHub repository, and each commit will include the author's name and a commit message. If there are files you don't want to commit, you can add file path and file name to the .gitignore file.

<https://github.com/apps/desktop>

## Visual Studio Code

A screenshot of a computer

Description automatically generated

Visual Studio Code, commonly known as VS Code, is a free, lightweight, yet powerful source code editor developed by Microsoft. Its key features include **integrated debugging tools**, syntax highlighting to aid in reading and understanding code, intelligent code completion that suggests code based on context, reusable snippets of code, code refactoring tools to restructure code without changing its behavior, and built-in support for version control with Git.

<https://code.visualstudio.com/>

### Using the Debugger for JavaScript Application

Today, I'm going to show you **how to use the Visual Code JavaScript debugger**.

1. Open the index.js file:

A screenshot of a computer

Description automatically generated

**Open the index.js** by clicking on the file.

1. Set abreakpoint:

A screenshot of a computer

Description automatically generated

1. **Click on a line number** where you want to add a breakpoint.
2. When you **see a red dot** that indicates a breakpoint has been set.
3. Open Debugger Tab:

A screenshot of a computer

Description automatically generated

Click **the debugger symbol** on the left side menu.

1. Run and Debug Button:

A screenshot of a computer

Description automatically generated

**Click on the Run and Debug Button** to start the debugger.

A screenshot of a computer

Description automatically generated

**Select Node.js**, as it's commonly used to run JavaScript applications outside the browser.

1. Watch Expression and Step Over Button:

A screenshot of a computer

Description automatically generated

1. To add a watch, **go to the watch panel** and add the expressions you want to monitor.

A screenshot of a computer

Description automatically generated

1. Use **'Step Over'** to see how variables change over time.

A screenshot of a computer

Description automatically generated

1. To see the console outputs, **continue running the code**.
2. Using Terminal Window:

If you prefer to run this **through a terminal window, follow the steps below**:

A screenshot of a computer

Description automatically generated

* 1. Open **a terminal window**.

A computer screen with a red arrow pointing to a computer code

Description automatically generated

* 1. To run the program without the debugger type **node and file name.**

## Docker Desktop

A screenshot of a computer

Description automatically generated

Docker Desktop is a widely used application that simplifies the process of building, sharing, and **running containerized applications**. It offers an intuitive user interface and comprehensive command line interface (CLI) tools to manage Docker containers and images seamlessly. With Docker Desktop, developers can create, test, and debug containers on their local machines before deploying them to production environments. It supports both Windows and macOS operating systems, making it a versatile tool for developers across different platforms.

Key features include Kubernetes integration, which allows developers to run Kubernetes clusters locally, and robust support for various programming languages and frameworks. Docker Desktop is essential for streamlining development workflows and ensuring consistency across different stages of application deployment.

<https://www.docker.com/products/docker-desktop/>

# Docker

## Introduction

Docker is an open-source platform that allows developers to automate the deployment of **applications inside lightweight containers**. [These containers package all the dependencies and code your app needs to run, ensuring it works consistently across different environments](https://www.howtogeek.com/devops/what-does-docker-do-and-when-should-you-use-it/).

Here’s a quick breakdown of **Docker’s key features**:

* **Isolation**: Each container runs in its own isolated environment, like a virtual machine but without the overhead of running a full operating system.
* **Portability**: Containers can run on any system that supports Docker, making it easy to move applications between different environments.
* **Efficiency**: Containers share the host system’s kernel, which makes them more lightweight and faster to start compared to virtual machines.
* [**Consistency:** By packaging everything your application needs, Docker ensures that it runs the same way in development, testing, and production environments, reducing the “it works on my machine” problem](https://www.howtogeek.com/733522/docker-for-beginners-everything-you-need-to-know/).

## Docker File

A diagram of a paper with arrows

Description automatically generated

A Dockerfile is a text document that **contains a series of instructions** on how to build a Docker image. These instructions are executed step-by-step to create an image that can be used to run containers.

Here are some **key points about Dockerfiles**:

* **Base Image**: The FROM instruction specifies the base image to use for the Docker image.
* **Commands**: Instructions like RUN, CMD, and ENTRYPOINT define commands to be executed inside the container.
* **File Operations**: COPY and ADD are used to copy files and directories into the image.
* **Environment Variables**: ENV sets environment variables.
* **Ports**: EXPOSE specifies which ports the container will listen on.
* **Metadata**: LABEL adds metadata to the image.

Here’s a **simple example of a Dockerfile**:

# Use an official Python runtime as a parent image

FROM python:3.8-slim

# Set the working directory in the container

WORKDIR /app

# Copy the current directory contents into the container at /app

COPY . /app

# Install any needed packages specified in requirements.txt

RUN pip install --no-cache-dir -r requirements.txt

# Make port 80 available to the world outside this container

EXPOSE 80

# Define environment variable

ENV NAME World

# Run app.py when the container launches

CMD ["python", "app.py"]

[This Dockerfile **sets up a Python environment**, installs dependencies, and runs a Python application](https://docs.docker.com/reference/dockerfile/).

## Docker Image

**What is Docker Image?**

[Docker images](https://www.geeksforgeeks.org/what-is-docker-image/) are built using the[Dockerfile](https://www.geeksforgeeks.org/what-is-dockerfile/) which consists of a set of instructions that are required to containerize an application. A [Docker](https://www.techtarget.com/searchitoperations/definition/Docker) image is a file used to execute code in a Docker [container](https://www.techtarget.com/searchitoperations/definition/container-containerization-or-container-based-virtualization).Docker images act as a set of instructions to build a Docker container, such as a template. Docker images also act as the starting point when using Docker. An image is comparable to a snapshot in virtual machine ([VM](https://www.techtarget.com/searchitoperations/definition/virtual-machine-VM)) environments. Docker images have multiple layers, each originating from the previous layer but different. A docker image is a platform-independent image that can be built in the Windows environment and it can be pushed to the [docker hub](https://www.geeksforgeeks.org/what-is-docker-hub/)and pulled by others with different OS environments like Linux.

## Docker Container

**What is Docker Container?**

A **Docker container image is a lightweight, standalone, executable package** of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Think of a **container as another form of**[**virtualization**](https://www.sdxcentral.com/networking/sdn/definitions/whats-network-virtualization/ease-of-use-virtualization/). VMs, also just one form of virtualization, allow a piece of hardware to host multiple operating systems as software. VMs are added to the host machine so that the hardware power can be shared among different users and appear as separate servers or machines. Containers virtualize the OS, splitting it into virtualized compartments to run container applications.

Docker Daemon is responsible for the assembling and running of code as well as the distribution of the finalized containers. **Docker Daemon takes the commands a developer enters the Docker client terminal and executes them**.

This **approach allows pieces of code** **to be put into smaller, easily transportable pieces** that can run anywhere Linux or Windows is running. It’s a way to make applications even more distributed and strip them down into specific functions.

## Docker Compose

Docker Compose is a tool that simplifies the process of **defining and running multi-containerDocker applications**.

Here’s a quick overview of its **key features**:

1. **Configuration with YAML**: You define your application’s services, networks, and volumes in a single YAML file. This makes it easy to manage and the version controls your configurations.
2. **Single Command Operations**: With a single command (docker-compose up), you can start all the services defined in your configuration file. Similarly, you can stop, rebuild, and manage the lifecycle of your application with simple commands.
3. **Multi-Environment Support**: Docker Compose works in various environments, including development, testing, staging, and production. [This flexibility makes it a powerful tool for continuous integration (CI) workflows](https://docs.docker.com/compose/).
4. [**Service Management:** You can view the status of running services, stream log outputs, and run one-off commands on services, which helps in debugging and monitoring](https://docs.docker.com/compose/).

For example, if you’re developing a web application that requires a web server, a database, and a caching service, **you can define all these components** in a docker-compose.yml file and manage them together.

## Create the Image

In the README file on GitHub, there is a section titled Usage Instructions where I explain how to **Build the Docker image**. Below, I will detail the Docker command used to build the image.

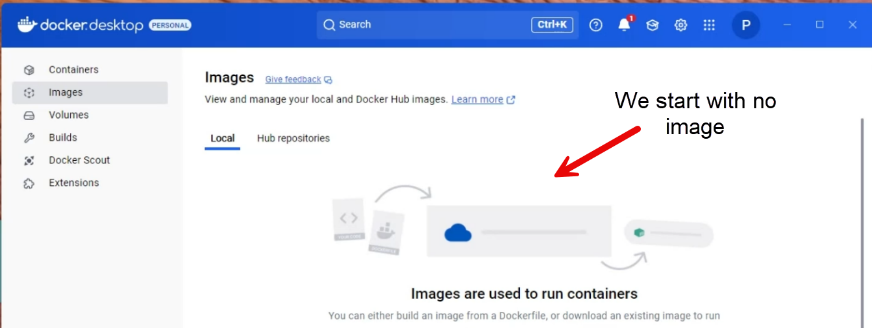
The command **docker build -t node-docker-app .**is used to build a Docker image from a Dockerfile. Here’s a breakdown of what each part of the command does:

* **docker build**: This is the Docker command to build an image from a Dockerfile.
* **-t node-docker-app**: The **-t** flag allows you to tag the image with a name*.* In this case, the image is tagged as **node-docker-app**.
* **.** : The period at the end specifies the build context, which is the current directory. Docker will look for a Dockerfile in this directory to build the image.

[**In summary**, this command tells Docker to create an image named node-docker-app using the Dockerfile located in the current directory](https://www.digitalocean.com/community/tutorials/how-to-build-a-node-js-application-with-docker).

### Steps to Building Image

In Docker Desktop, click the Images Menu to view all the images. **We start with no image**.



**When you look at the Dockerfile**, you'll see that the first line starts with 'FROM', which pulls the Alpine Node image. The second instruction sets the working directory. Next, we copy the package.json file into the working directory and run the npm install command to download the dependencies to the Node Modules Folder. Then, we copy all the code into the working directory and expose port 3000. Finally, the last command runs node app.js, which starts the web server and launches the Express program.

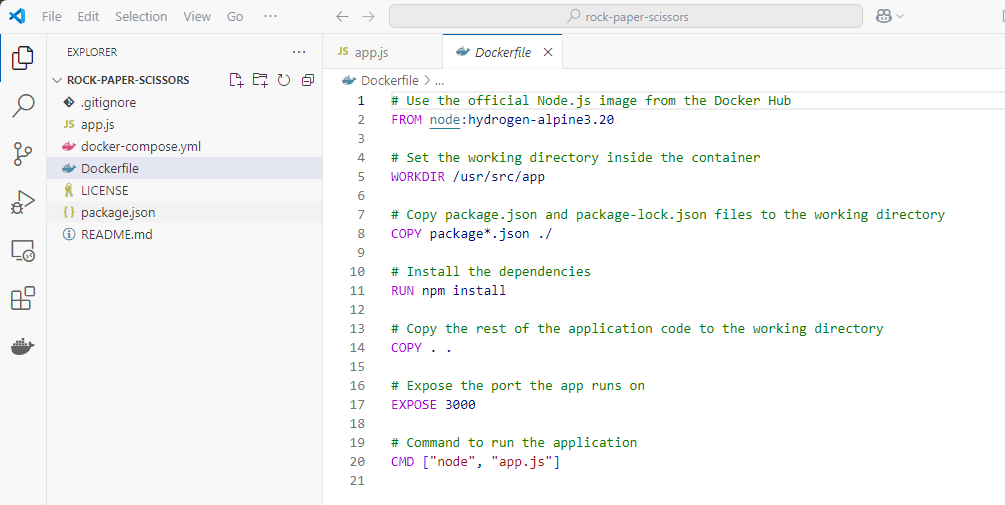


Figure 7.1

Now we must **open the terminal window** so we can run the docker command.

A screenshot of a computer

Description automatically generated

Next, **run the following Docker command**: docker build -t node-docker-app .

A red arrow pointing to a list

Description automatically generated

Now in the Docker Desktop, **click on the Images Menu** to see the list of images. In Docker Desktop, when you can **click on the image name (node-docker-app)** you can view the image layers and hierarchy used to build the base image.

A screenshot of a computer

Description automatically generated

You can **see the image’s 15 layers and the hierarchy**, which is based on the Linux distribution that forms the base image.

A screenshot of a computer

Description automatically generated

## Starting the Docker Container

In the README file on GitHub, there is a section titled Usage Instructions where I explain **How to start the Docker Container**. Below, I will detail the Docker command used to start the container.

The command **docker run -p 3000:3000 node-docker-app** is used to run a Docker container from an image. Here’s a breakdown of what each part of the command does:

* **docker run**: This command is used to create and start a new container from a specified image.
* **-p 3000:3000**: This flag maps port 3000 on your host machine to port 3000 in the container. This means that any traffic to port 3000 on your host will be forwarded to port 3000 in the container.
* **node-docker-app**: This is the name of the Docker image from which the container will be created.

### Steps to Starting the Docker Container

[The docker run command **starts a container from the node-docker-app image** and maps port 3000 on your local machine to port 3000 in the container, allowing you to access the application running inside the container via [http://localhost: 3000](http://localhost:3000)](file://C:\Users\cocon\Desktop\New%20folder\Resources\Steps_PodCast\CREATING_A_WEBSITE_WITH_DOCKER\The%20docker%20run%20command%20starts%20a%20container%20from%20the node-docker-app image%20and%20maps%20port%203000%20on%20your%20local%20machine%20to%20port%203000%20in%20the%20container,%20allowing%20you%20to%20access%20the%20application%20running%20inside%20the%20container%20via http:\localhost:3000).

A screenshot of a computer

Description automatically generated

Now we must **open the terminal window** so we can run the docker command.

A screenshot of a computer

Description automatically generated

Next, **run the following Docker command**:docker run -p 3000:3000 node-docker-app

A screen shot of a computer

Description automatically generated

The **console log message** indicates that the application server has started, and the application is now running at [http://localhost: 3000](http://localhost:%203000).

A screen shot of a computer

Description automatically generated

If you **open your browser** and enter the URL [http://localhost: 3000](http://localhost:%203000), you'll see a webpage that says 'Hello, Docker!’

A screenshot of a computer

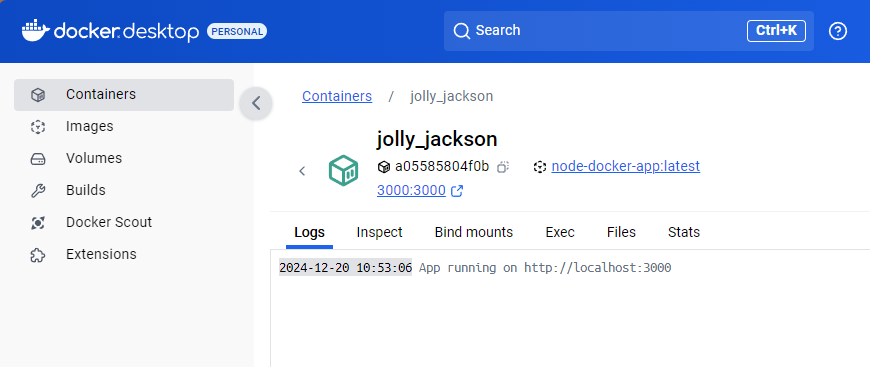
Description automatically generated

In Docker Desktop, **click the Containers Menu** to view all the containers.

A screenshot of a computer

Description automatically generated

In Docker Desktop, clicking on the container name **(jolly\_jackson)** allows you to view the file structure, log files, and access a command line to run Linux commands. The container name is generated by the daemon engine.



When **you open the app.js file**, you'll notice the first three lines are JavaScript Constants defining the Express JavaScript library and port number. The first method sends messages to the browser. The second part, under the 'listening' method, logs that the application is running and on which port.

A screenshot of a computer program

Description automatically generated

## Command Line

Accessing the command line is **excellent for debugging your container** while it’s running.

The **Linux commands I entered** the terminal perform the following actions:

The **pwd command** in Linux stands for "print working directory." It displays the current directory you're working on. When you type pwd and press Enter in your terminal, it will output the full path of your current directory.

The **ls -la command** in Linux lists all files and directories in the current directory, including hidden files. It provides detailed information about each file and directory, such as permissions, number of links, owner, group, size, and the last modification date. Here's a breakdown of what each part does:

* **ls**: Lists files and directories.
* **-l**: Uses a long listing format to provide detailed information about each file and directory.
* **-a**: Includes hidden files and directories in the listing.

A screenshot of a computer

Description automatically generated

## File Structure

Another excellent debugging tool is the **ability to inspect the file structure**. You can navigate to the working directory, which is the /user/src/app folder, to check the node\_modules folder. This will show what was installed when the npm install command was run.

A screenshot of a computer

Description automatically generated

## Steps to Starting the Docker Compose

**Introduction**

Docker Compose is a tool that simplifies the process of defining and running multi-container Docker applications. It **uses a docker-compose.yml** file to configure your application's services. With just a single command, you can create and start all the services defined in this configuration.

**Ensure Docker and Docker Compose are Installed**: Make sure you have both Docker and Docker Compose installed on your machine. You can check by running **docker --version** and **docker-compose --version** in your terminal. ***Ensure that the Docker Desktop is running***.

**docker –version**

**docker-compose --version**

**Navigate to Your Project Directory**: Open your terminal and navigate to the directory **containing your** **docker-compose.yml** file using the cd command.

cd path/to/your/project

A screenshot of a computer

Description automatically generated

**Start Docker Compose**: Run the docker-compose up to start Docker Compose and bring **up the services defined in your docker-compose.yml** file:

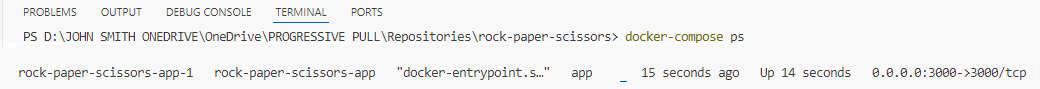
docker-compose up

If you want to run it in **detached mode** (in the background), use the -d flag:

docker-compose up -d

**Check the Status**: You can check the status of your services by running docker-compose ps:

docker-compose ps



**Stop Docker Compose**: To **stop and remove the containers, networks, and volumes** defined in your docker-compose.yml file, use docker-compose down:

docker-compose down

**Note**: You can stop Docker Compose from the Bash Terminal using **Ctrl + C**.

## RUN vs. CMD in Docker File

### Using the RUN Instruction

The RUN instruction allows you to execute commands during the image build process. This is typically used to install software packages or perform setup tasks. The commands specified in the RUN instruction are executed in a new layer on top of the current image.

In this example:

* The RUN **npm install** command installs the Node.js dependencies specified in the package.json file.

### Using the CMD Instruction

The CMD instruction specifies the default command to run when a container starts. Unlike the RUN instruction, which is executed during the build process, the CMD instruction is executed when the container is launched.

In this example:

* The **CMD ["node", "app.js"]** command specifies that node app.js should be run when the container starts, which will start the Node.js application.

### Key Differences

* **RUN**: Executes commands at build time and creates a new image layer. Used for installing software and setting up the environment.
* **CMD**: Specifies the command to run when a container is started. There can be only one CMD instruction in a Dockerfile. If multiple CMD instructions are specified, only the last one will be used.

## References

### Markdown

* [Markdown Guide](https://www.markdownguide.org/)
* [Markdown Cheat Sheet](https://commonmark.org/help/)
* [Dillinger Editor](https://dillinger.io/)
* [The Only Markdown Crash Course You Will Ever Need - Web Dev Simplified - YouTube](https://www.youtube.com/watch?v=_PPWWRV6gbA&t=60s)

### Linux Commands

* [Phoenixnap Linux Commands Cheat Sheet](https://phoenixnap.com/kb/linux-commands-cheat-sheet)
* [Linux Cheat Sheet - Loggly](https://www.loggly.com/wp-content/uploads/2015/05/Linux-Cheat-Sheet-Sponsored-By-Loggly.pdf)
* [Linux Commands - Digital Ocean](https://www.digitalocean.com/community/tutorials/linux-commands)

### GIT

* [Git Official Website](https://git-scm.com/)
* [Git Tutorial for Beginners: Learn Git in 1 Hour - Programmer with Mosh - YouTube](https://www.youtube.com/watch?v=8JJ101D3knE)
* [How Git Works: Explained in 4 Minutes - ByteByteGo - YouTube](https://www.youtube.com/watch?v=e9lnsKot_SQ)
* [GitHub Git Ignore References](https://github.com/github/gitignore)

### Source Tree

* [SourceTree Official Website](https://www.sourcetreeapp.com/)
* [Learn Git, Sourcetree & BitBucket Tutorial: A Crash Course for Beginners - Cameron McKenzie - YouTube](https://www.youtube.com/watch?v=x_3NA-_Rjwo)

### NodeJS and NPM

* [NodeJS Official Website](https://nodejs.org/en)
* [NPM Official Website](https://www.npmjs.com/)
* [NPM Documents](https://docs.npmjs.com/)

### GitHub Desktop

* [GitHub Desktop Official Webpage](https://github.com/apps/desktop)
* [Git, GitHub, & GitHub Desktop for beginners - Code Coder - YouTube](https://www.youtube.com/watch?v=8Dd7KRpKeaE)

### Visual Studio Code

* [Microsoft Visual Studio Code Official Website](https://code.visualstudio.com/)

### Docker Desktop

* [Docker Desktop Official Website](https://www.docker.com/products/docker-desktop/)

### Docker

* [Docker Manuals](https://docs.docker.com/manuals/)
* [Docker Tutorial - Geeks For Geeks](https://www.geeksforgeeks.org/docker-tutorial/)
* [Docker Workshop](https://docs.docker.com/get-started/workshop/)
* [Docker Introduction](https://docs.docker.com/get-started/introduction/)