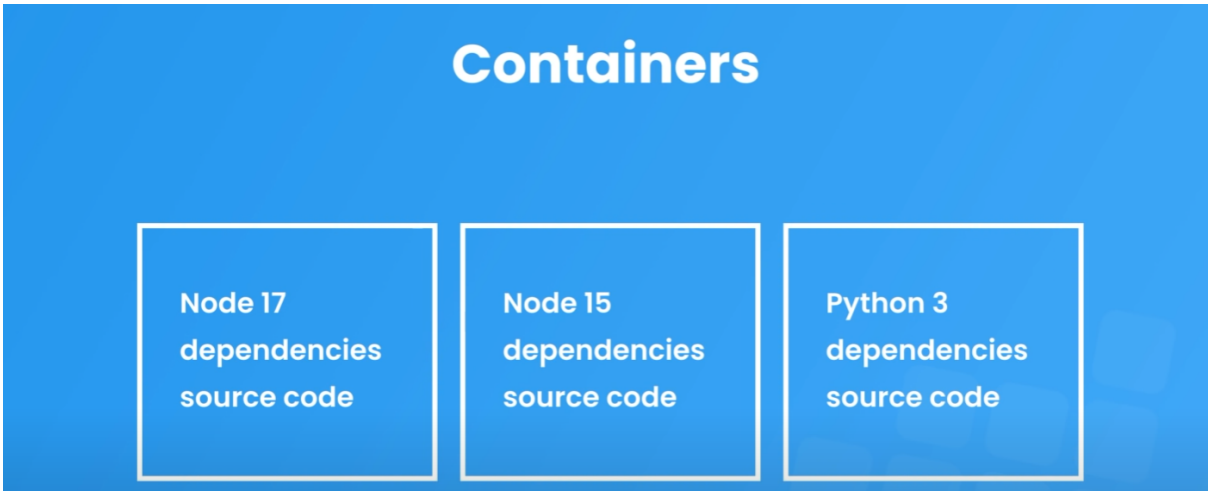
6. Docker

#1 What is Docker?

- Different applications will require → different configurations, dependencies, and libraries.
- Uses containers to run applications in isolate environment.
  - Containers → everything that our application requires in order to run.
  - Doesn’t matter what configurations we have, the application can run in isolation.
- Predictable, consistent, and isolated environment.
- Just need Docker on other systems to manage containers.



Basic idea behind Docker Containers

- Improve deployment process → pulled onto production server : all the configuration is already setup.
- But, why can't we use Virtual Machines instead of Docker? The idea seems to resembles the same!
  - **Virtual Machines** → has full operating system and typically slower!
  - **Containers** → Share the host's operating system and typically quicker!
    - Less memory - lightweight
  - Not that one is better than the other, there might be several cases where we might want to use VMs instead of containers.

#2 Images & Containers

- **Images**
  - Read-Only
  - Blueprint for containers
  - Include every single thing that our application needs to run.
    - Runtime environment, code, dependencies, additional configuration, and commands.
  - We can share the image to allow to form the container to run the application.
- When we run an image → Container is formed.
  - Container is the running instance of the image.

    - Isolated process → Like running in a box which is isolate from other processes in the computer.
- Images are made up from several *layers.*

  → We start with inclusion of Parent Image : OS and Runtime environment.

  → Source Code, Dependencies, etc.

  → Docker Hub

  → By specifying tags, we can determine the version of image along with Linux distribution.

    → Example : Alpine distribution → Lightweight

  → Doesn't have to be a node image, it could be Python or Ruby, anything.

```
docker pull node
```

#3 Docker File

- Whenever we start with docker, we will be starting with a parent image and then, we will continue to add on different layers.
- Create our own image → Docker File
    - Fundamentally, a set of instructions to develop an image → different layers that needs to be included in the image.
- Normally, if we have to run an application we will do so by installing all the dependencies and then, executing *node app*
- But, when I do so, it is going to use the node version that is installed locally on the computer and this could be different for everyone, causing issues.
- We need to run our application inside an isolated container, which has its own version of node running inside of it.
- **Docker File**
    - Each line represents layer in final image.

```
FROM node:17-alpine

// This is going to pull the image from local system - if we have downloaded it or
// from the docker hub if we haven't.
```

```
WORKDIR /app

// add a work directory where all commands will be run

COPY . /app -> COPY . .

// copy all of our soure code into the image -> first dot is the relative path of the
// directory

// We don't directly copy to the root : we copy to /app -> which is a specific directory
// for our code.

RUN npm install

// asks docker to run a command while the image is being built
// Run is executed during the build time -> when the image is being built

EXPOSE 4000

// Which Port the container should Expose. Also tells the port that is going to be
// used by the application. -> Kind of optional.
// Alternatively, we can do this by cmd : port mapping

CMD ["node", "app.js"]

// Run the application in container
// Runs the instance
// Write the commands in double quotes
```

- To run the image
  - *docker build -t my_app .*
  - T flag allows us to give a tag and the dot at the end defines where the Docker file exists.