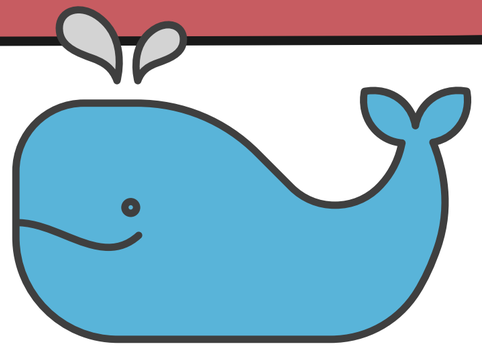
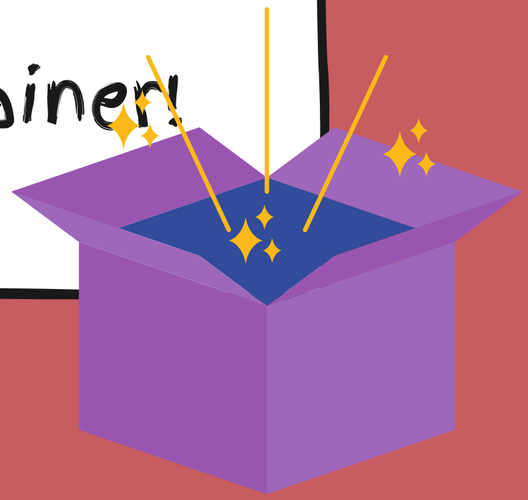


Docker Notes

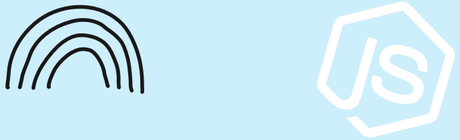
Everything you need to run your first Container!

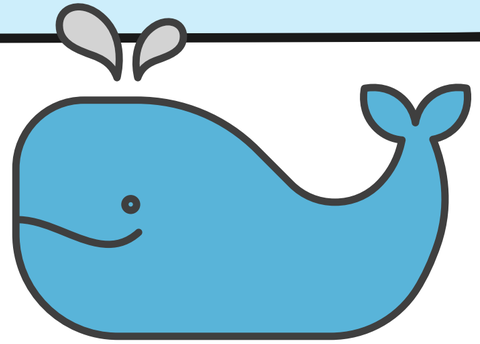


Let's know each other :)

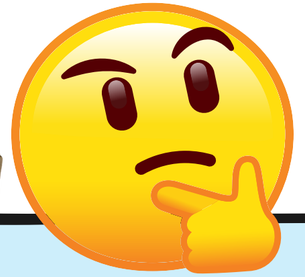
Hi, I'm Jatin Shharma.
So I teach students and professionals these interesting topics like:
Programming, Automation Testing and DevOps
If you too have the same interest feel free to connect with me on Linked In

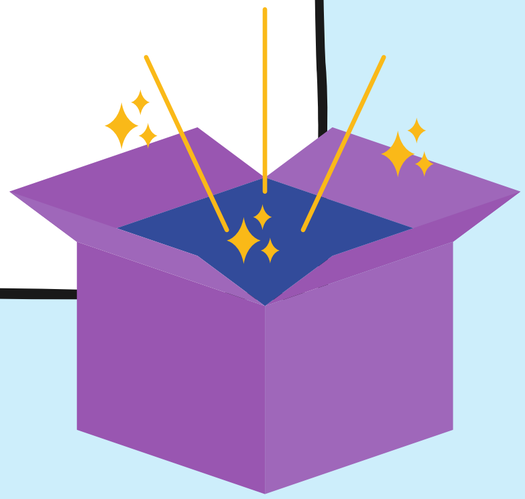
ah and my website: www.testautomationacademy.in





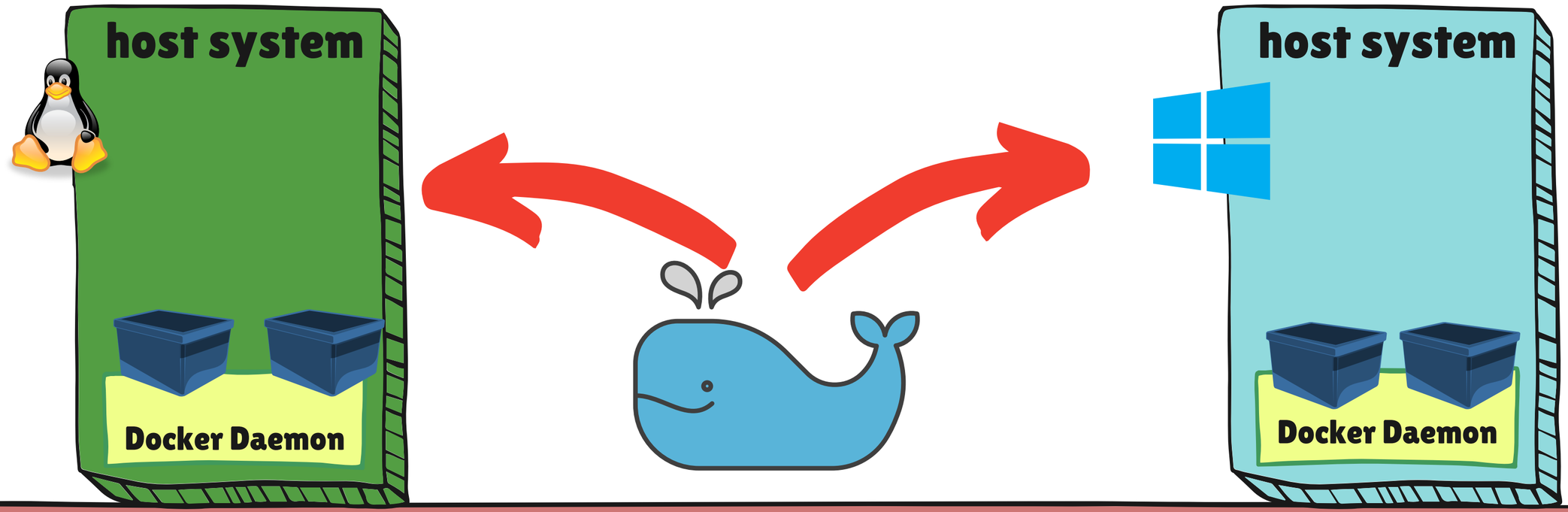
What is Docker





What is Docker

Docker is a platform for **developers** and **sysadmins** to **develop**, **ship**, and **run** applications by using containers.



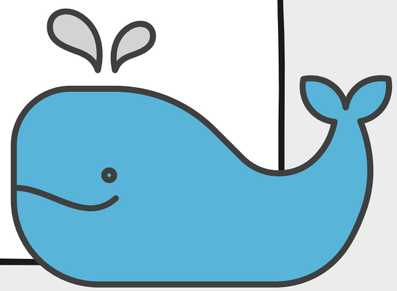
Docker aids the user to test and deploy the code into production Quickly!

VIMP

Why use Docker ?

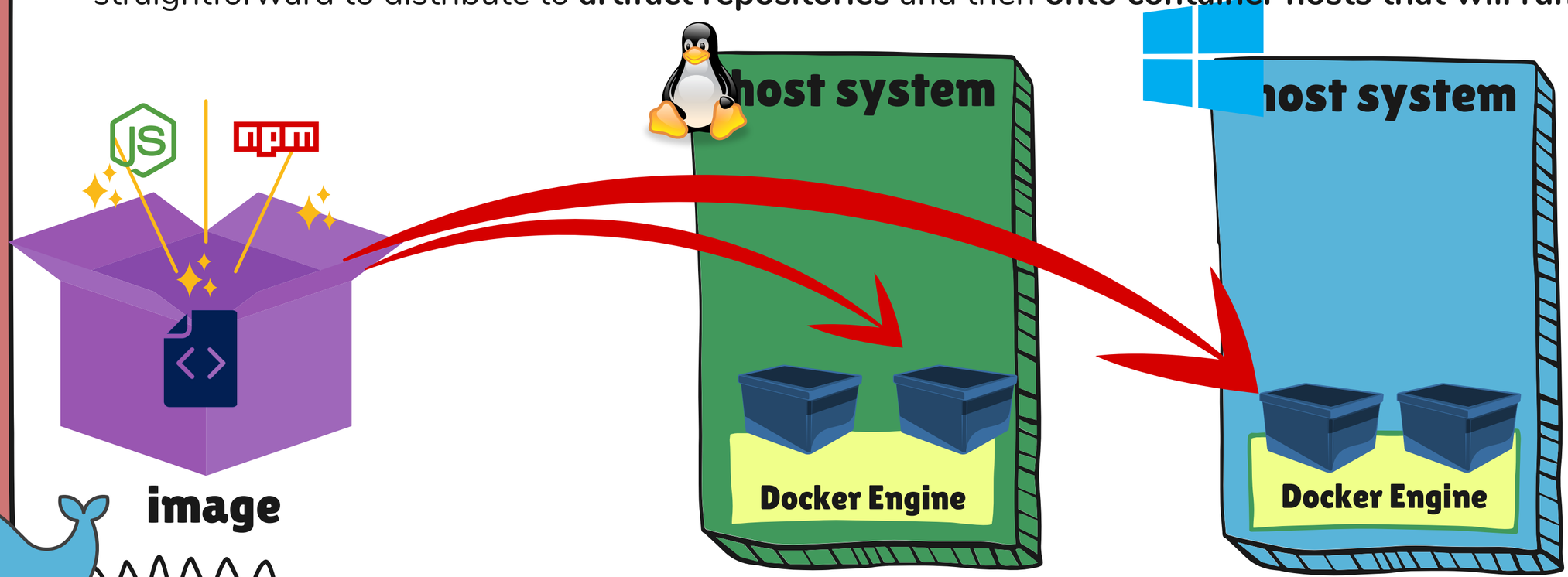
Docker helps in:

1. Application packaging 
2. Process Isolation and Management
3. Multitenancy 
4. Rapid Deployment 



What is Application Packaging?

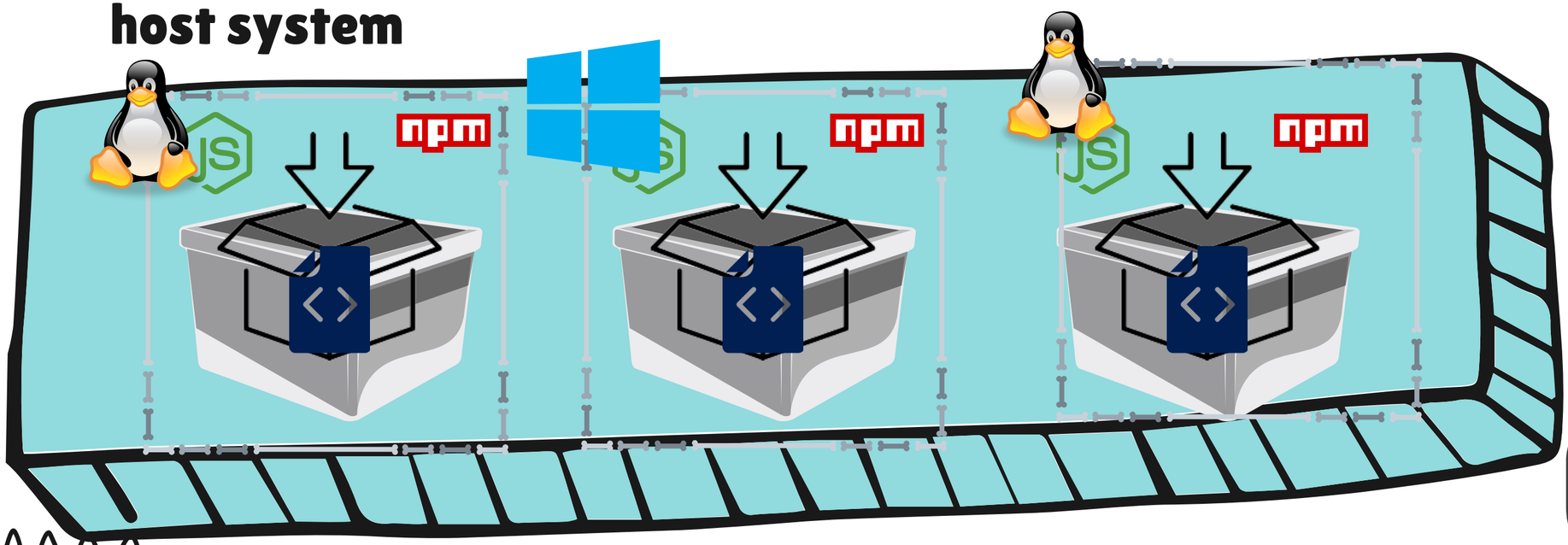
Docker helps you package applications and their **dependencies** into **portable application images** that are straightforward to distribute to **artifact repositories** and then **onto container hosts that will run them**



Docker Engine consist of server which a program which called as docker daemon

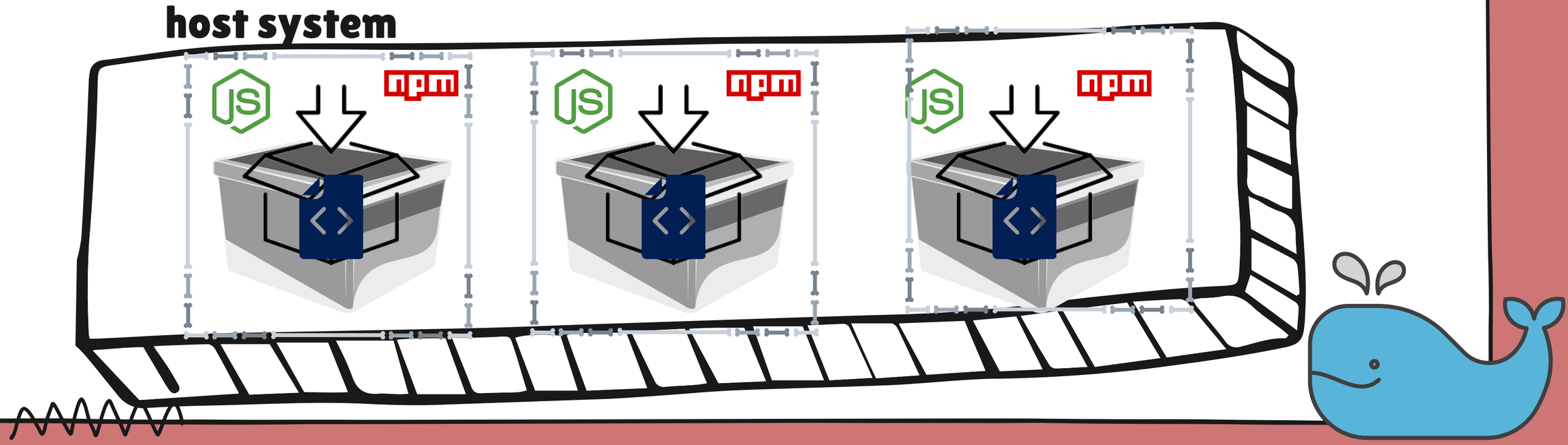
What is Process Isolation in Docker?

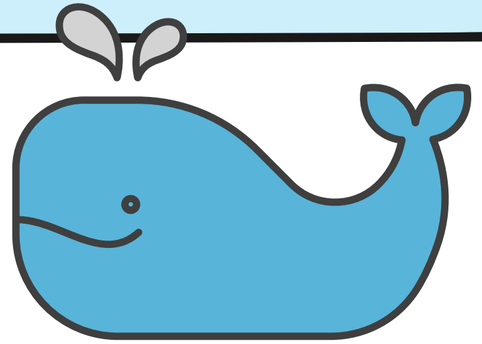
Docker's container engine and command line tool make it simple to retrieve application images and start **isolated instances of each application process.**



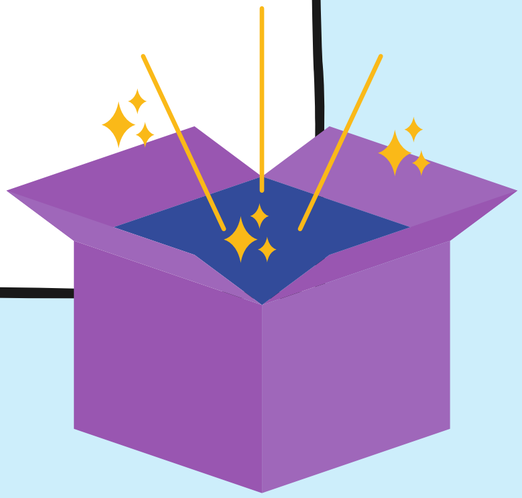
⚠️ Key terms to Remember

Docker’s container engine and command line tool make it simple to retrieve application images and start **isolated instances of each application process.**

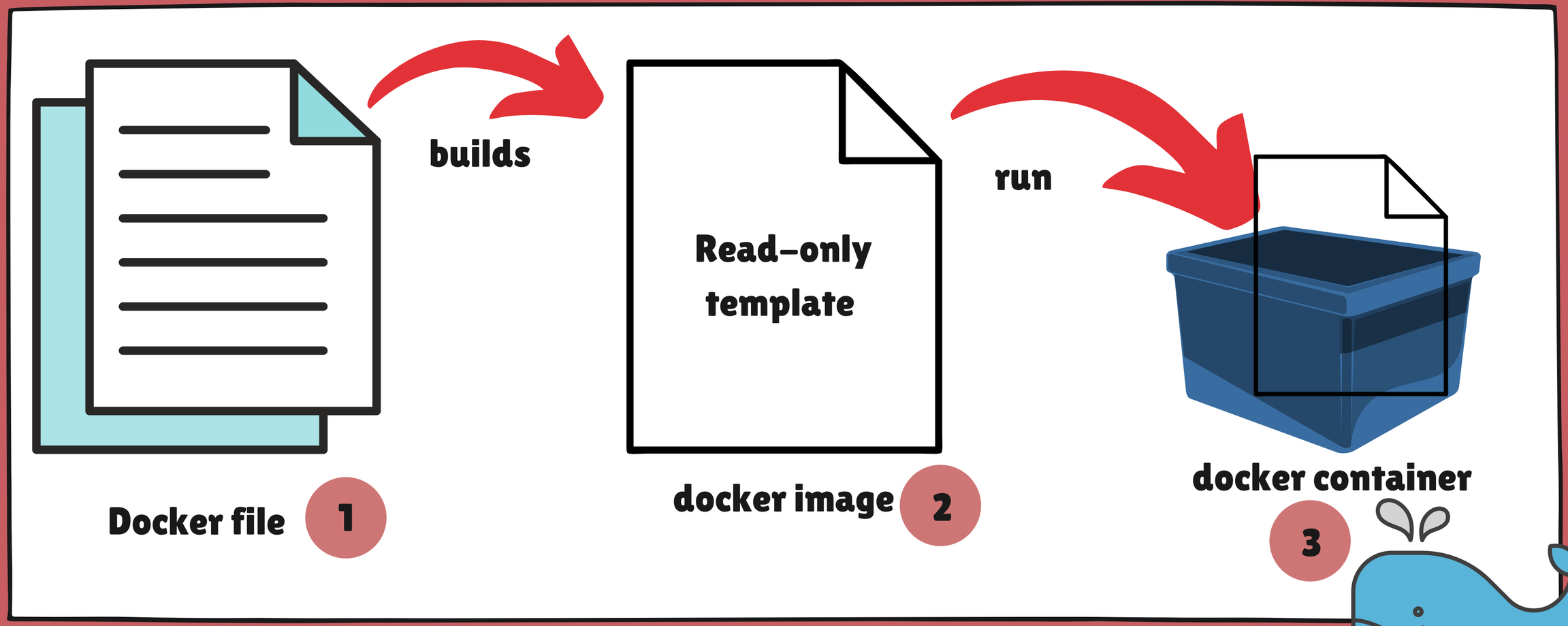




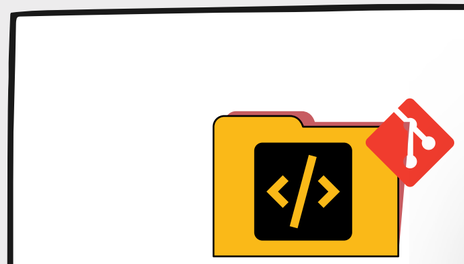
Workflow to Containerize any application



Workflow to Containerize any application



Workflow to Containerize any application



dockerfile

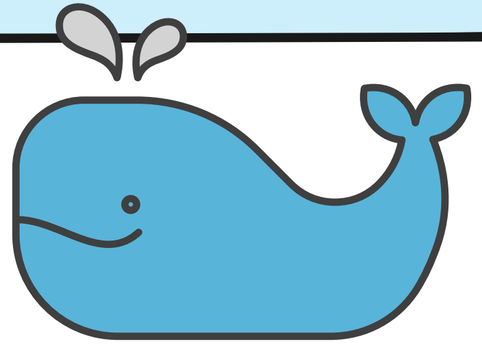
1

```
FROM node:16.13.2-alpine
RUN mkdir /app
WORKDIR /app
COPY package.json /app
RUN npm install
COPY . /app
CMD ["npm","start"]
```

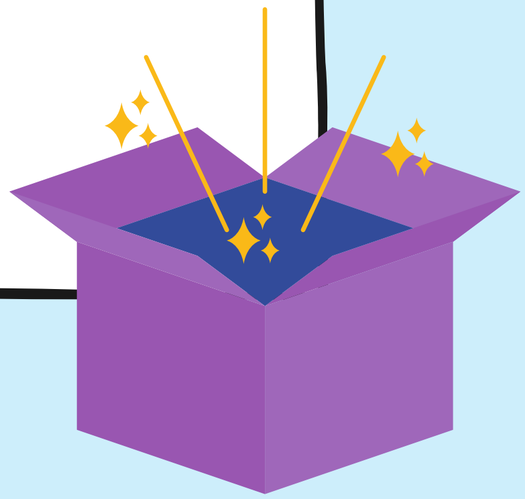
2 docker build

3 docker run

Note: Here we are creating a dockerfile for nodejs application!



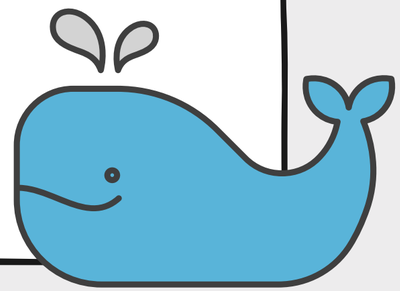
Basic Docker Commands



VIMP

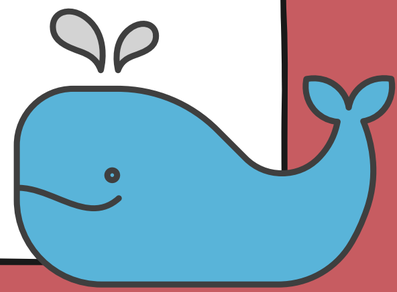
⚠️ Basic Docker Commands

1. docker **build : The docker build command builds Docker images from a Dockerfile**
2. docker **pull : Pull an image or a repository from a registry**
3. docker **push: Push an image or a repository to a registry**
4. docker **ps : returns an overview of all running containers.**
5. docker **images : Shows the list of images**



Basic Docker Commands

6. docker **system prune: Remove unused data**
7. docker kill **: Kill one or more running containers**
8. docker restart**: Restart one or more containers**
9. docker rm **: Remove one or more containers**
10. docker rmi **: Remove one or more images**

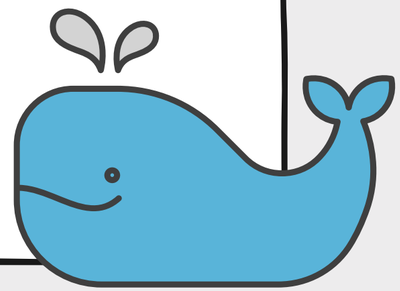


⚠️ Basic Docker Commands

11 docker **logs: Fetch the logs of a container**

12 docker run **:Run a command in a new container**

13 docker login**: Log in to a Docker registry**



Let's ~~package~~ containerize our application

Assignment:

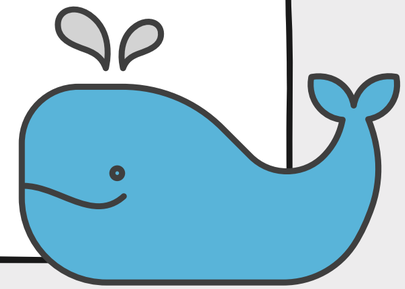
Your company has decided to adopt the devops process for your current project and they wish to change the way they are currently deploying the application on the server.

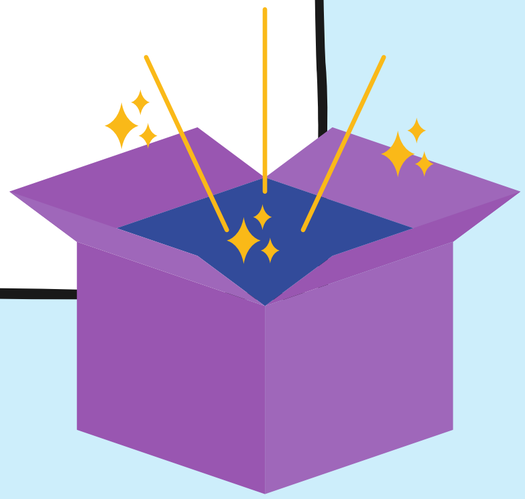
In past, the deployment was a very complex process because sometimes the developer system config and server config were mismatched. Hence it used to take a lot of time to deploy the application to PROD and find if the correct version has been deployed.

Instead of pulling the code from the remote repository, they want to deploy the docker image of the application.

Your Job is to containerize the existing application. Push the image to docker hub and deploy it into the cloud server.

PS make sure the image size is optimum.





Docker File Instruction List

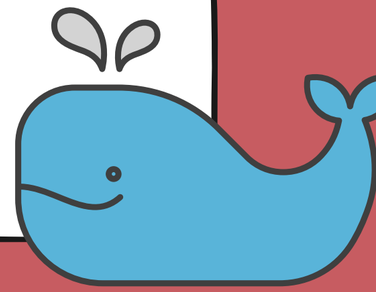
VIMP

Step1: Create Docker File

Docker File is text file that consist of all the instructions that are required a docker image.

Each instruction creates a docker layer in the image!

FROM	ADD	USER	ONBUILD
RUN	COPY	VOLUME	STOPSIGNAL
CMD	EXPOSE	WORKDIR	ENTRYPOINT
LABEL	ENV	ARG	HEALTHCHECK



INSTRUCTION LIST

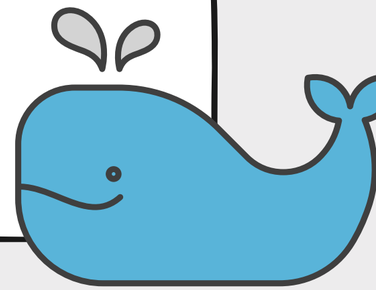
FROM

FROM is used to begin a new build stage. This sets the base image for subsequent instructions.

Syntax: FROM <image> [AS <name>]

Remember:

- **It's all ways the First instruction in DockerFile**
- **FROM can be written multiple times to create images in a DockerFile.**
- **name is the build stage in the Dockerfile!**



INSTRUCTION LIST

Run

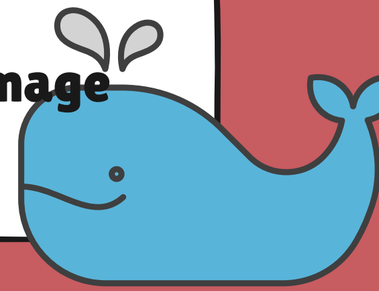
RUN instruction is used to execute a command inside a docker image!

Syntax: RUN [COMMAND]

Example: RUN yum update

Remember:

- **You can execute shell linux command or app command directly inside an image with the help of Run instruction.**
- **RUN instruction is used to executes any commands on top of the current image and this will create a new layer**



INSTRUCTION LIST

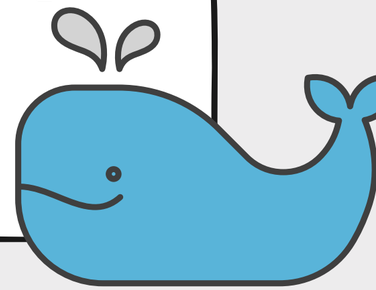
CMD

FROM is used to begin a new build stage. This sets the base image for subsequent instructions.

Syntax: FROM <image> [AS <name>]

```
cmd java SampleJava.java
```

cmd [“java ”,” SampleJava.java”]

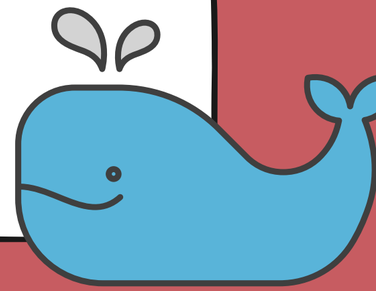


INSTRUCTION LIST

CMD

Remember:

- It's all ways the First instruction in DockerFile
- FROM can be written multiple times to create images in a DockerFile.
- name is the build stage in the Dockerfile!



INSTRUCTION LIST

LABEL

If you wish to add extra information or metadata information to the image you can use LABEL instruction

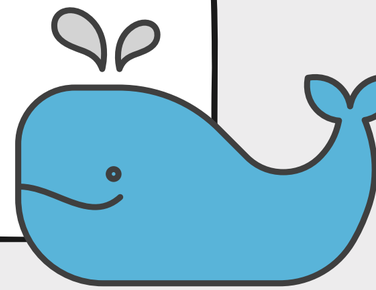
Syntax: LABEL <key>=<value>

Remember:

A LABEL is a key–value pair.

Example:

label “application_environment”=”development”



INSTRUCTION LIST

ADD

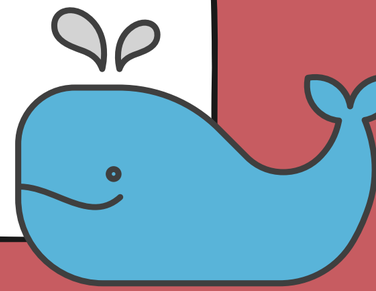
ADD copies new directories, files, and URLs of remote files from src.
The is a path where the source is copied in the destination container.

Syntax: ADD [--chown=<user>:<group>] <src> .. <dest>

Example:

Shell Form - ADD src dest

add /root/testData /data/



INSTRUCTION LIST

COPY

- The COPY copies new directories and files from source.
- COPY also adds those new directories and files to the container's filesystem at the path <DEST>

Syntax: COPY [--chown=<user>:<group>] <src> .. <dest>

Example

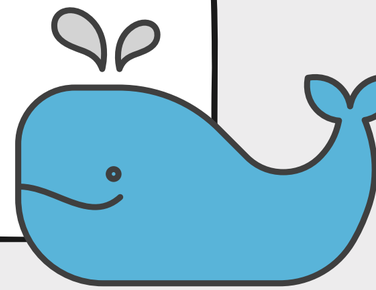
Shell Form

copy src dest

copy /root/testData /data/

Executable Form

copy ["src","dest"]



INSTRUCTION LIST

EXPOSE

EXPOSE sends information to Docker that the container listens on the specified network ports at runtime.

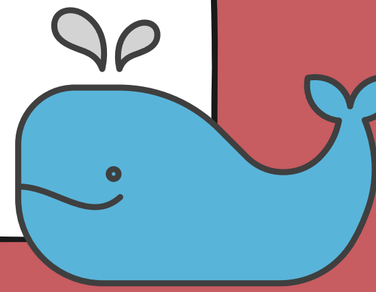
Syntax: EXPOSE <port>[PORT/<PROTOCOL>

Remember:

Docker uses this information to interconnect containers using links and to set up port redirection on docker host system.

Example:

expose 80 443



INSTRUCTION LIST

ENV

ENV is a key–value pair. It sets the <KEY> , an environment variable, to the value <VALUE>.

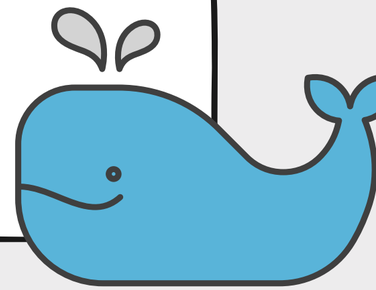
These variables will be set during the image build also available after the container is launched.

Syntax: EXPOSE <port>[PORT/<PROTOCOL>

Remember:

Examples:

env app_dir /data/



INSTRUCTION LIST

USER

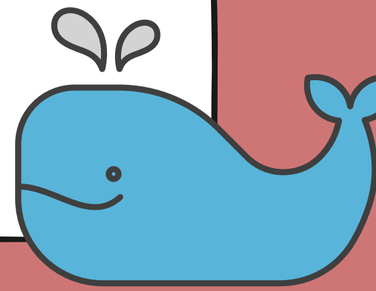
USER assigns user name and user group while running the image. It also assigns user name and user group for the RUN, CMD, and ENTRYPOINT instructions.

Syntax: USER <USER>:<GROUP>

Example :user webadmin

Remember:

USER instruction is used to set the username,group name, UID and GID for running subsequent commands. Else root user will be used.



INSTRUCTION LIST

VOLUME

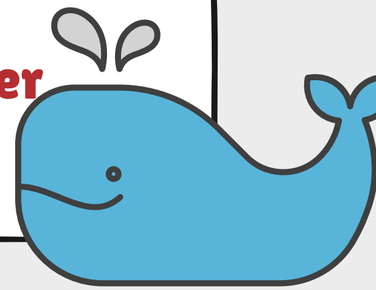
VOLUME creates a mount point with a specific name.

Syntax: VOLUME ["/data"]

Example volume /data

Remember:

VOLUME instruction is used to create or mount a volume to the docker container from the docker host filesystem.



INSTRUCTION LIST

WORKDIR

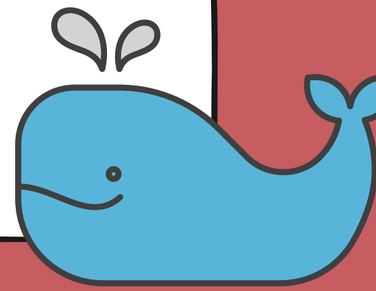
WORKDIR sets the directory for RUN, CMD, ENTRYPOINT, COPY, and ADD

Syntax: WORKDIR /path/to/workdir

Example: workdir /app/

Remember:

WORKDIR instruction is used to set the working directory.



INSTRUCTION LIST

ARG

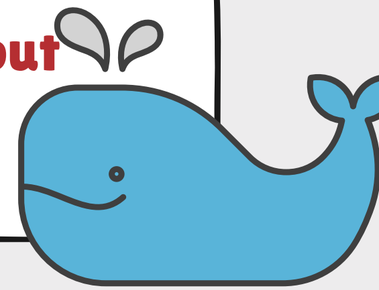
ARG defines the variables that are passed by the user to the builder at the build-time.

Syntax: ARG <name>=<default>

Example: arg tmp_name mycustom_image

Remember:

ARG instruction is also used to set environment variables with key and value, but this variables will set only during the image build not on the container

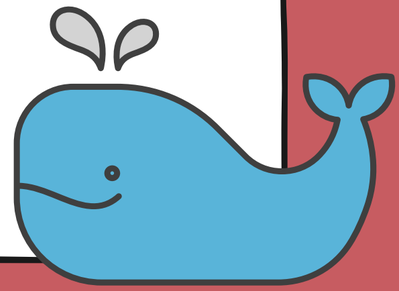


VIMP

How to Optimize a Docker File

Remember:

1. Always use **Alpine**
2. Using **Dockerignore**
3. Use **Multi Stage Build**
4. Keep **Application Data outside the Container!**
5. Reduce Number of **Layers created byDockerFile**
6. Build **Custom Base Images**
6. Only Install **Necessary Packages**



Practice Assignment

1. Run Jenkins on a DO in a dockerized container and add volume to the container

2. Run SONARQUBE on a DO in a dockerized container and add volume to the container