Specialist Diploma in Artificial Intelligence

AI Applications with Deep Learning

Deploying a Deep Learning Model

Sub topics

- Using Docker
- Tensorflow Server
- Integrating with a Client Application

Deploying a Deep Learning Model

USING DOCKER

Docker and Containerization

- In development, our application works on specific version of:
  - Operating Systems
  - Dynamically Linked Libraries
  - Dependencies (for interpreted languages like Python, Nodejs)

- When that code gets deployed onto production, it stops work due to version differences

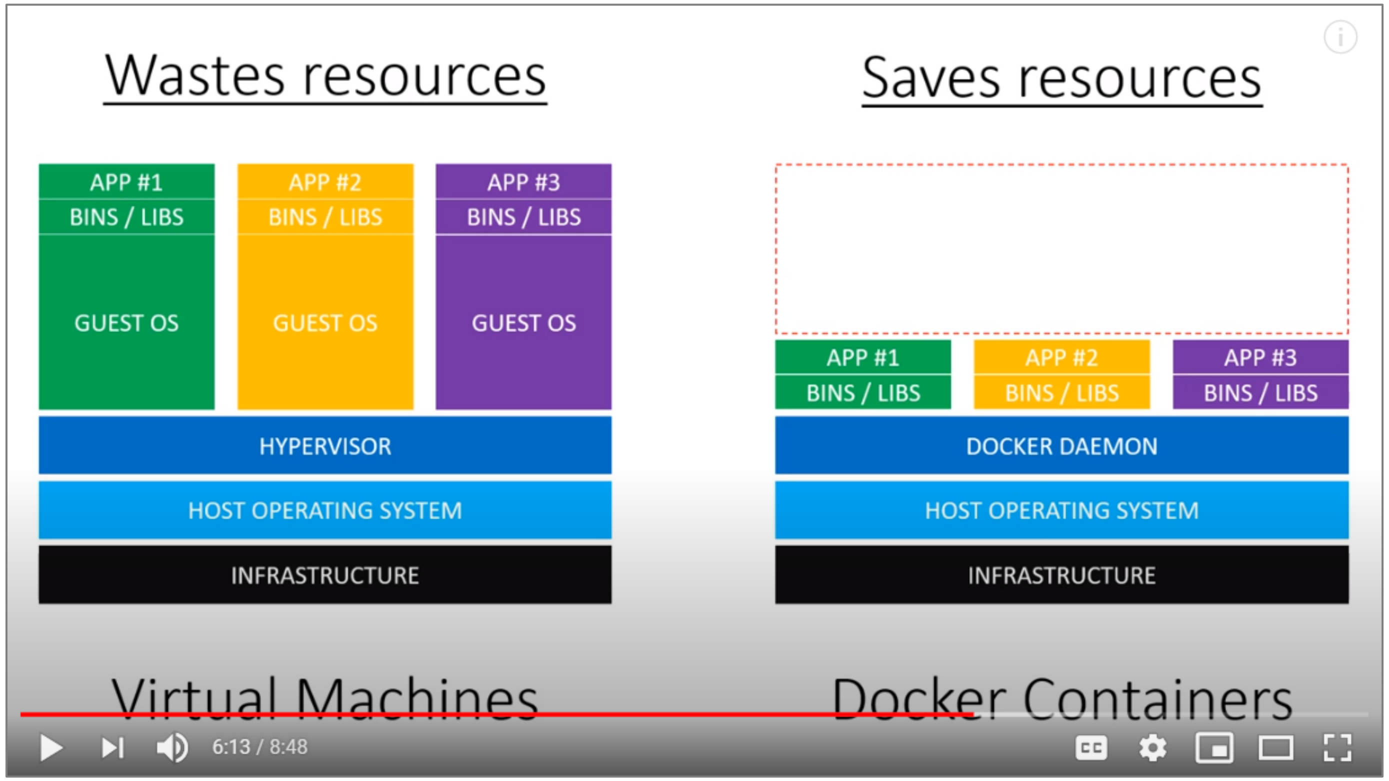
Docker and Containerization

- Docker solves this problem by keeping a thin layer of OS libraries, dependencies in a container

- One host machine can run multiple containers.

- Virtual Machines achieves a similar result, but have more overhead

Docker and Containerization

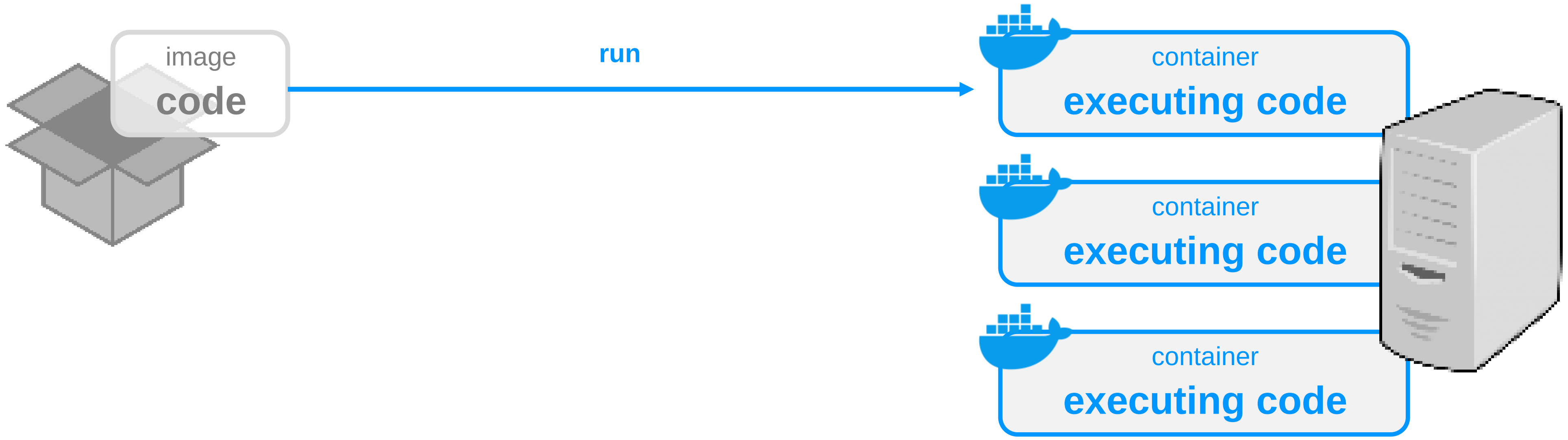


https://www.youtube.com/watch?v=TvnZTi_gaNc

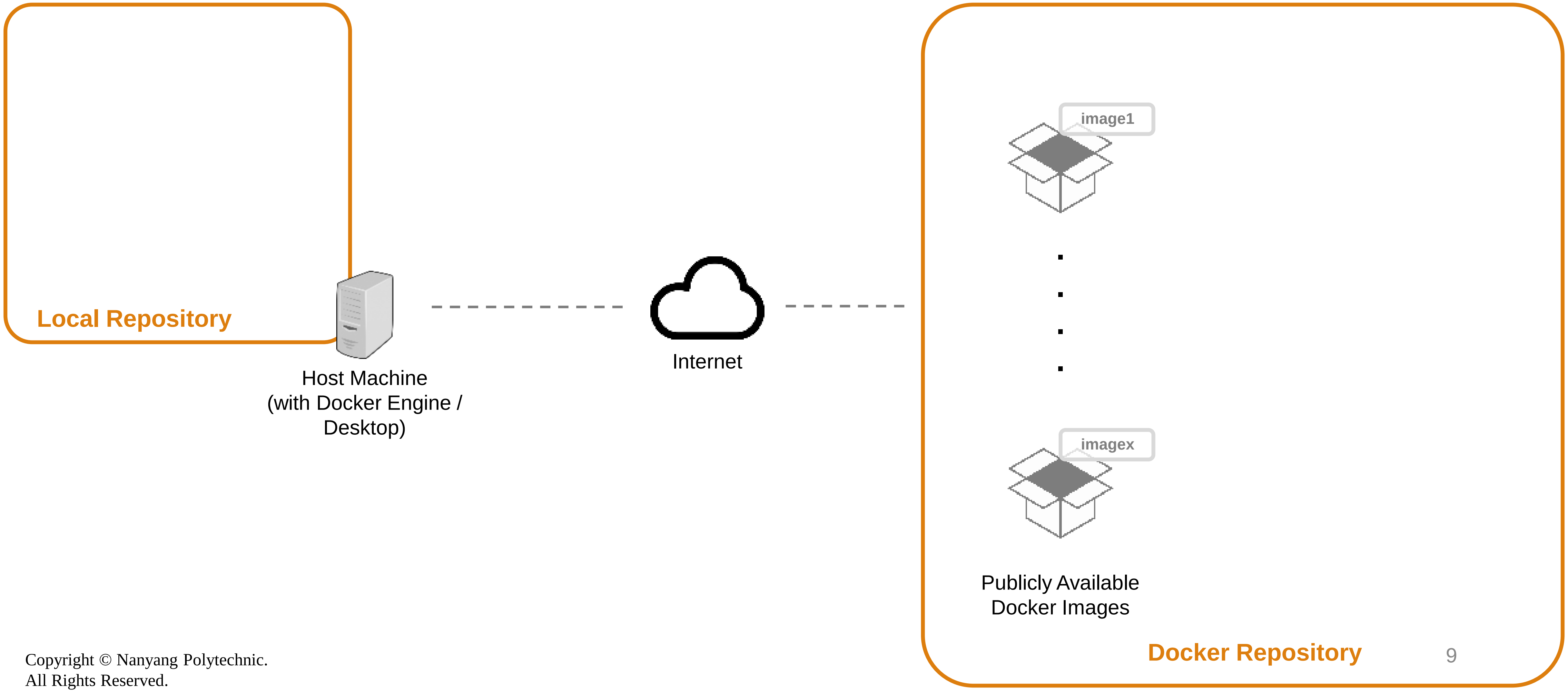
What is a Docker Image and Docker Container?

- Docker Image
  - A package containing all software required to run an application

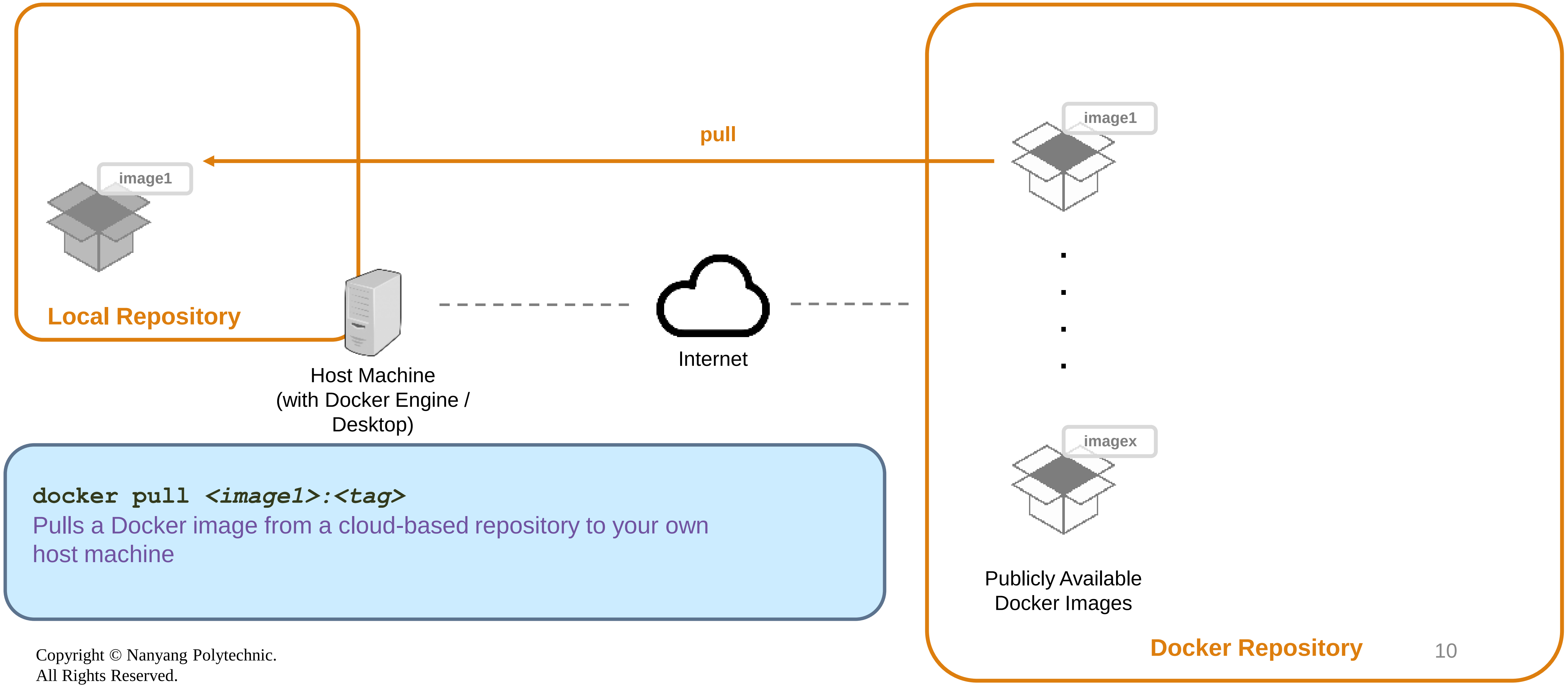
- Docker Container
  - A compartmentalized environment that the software contained is executing



Docker – At The Beginning


Local Repository
Host Machine (with Docker Engine / Desktop)
Internet
image1
imagex
Publicly Available Docker Images
Docker Repository

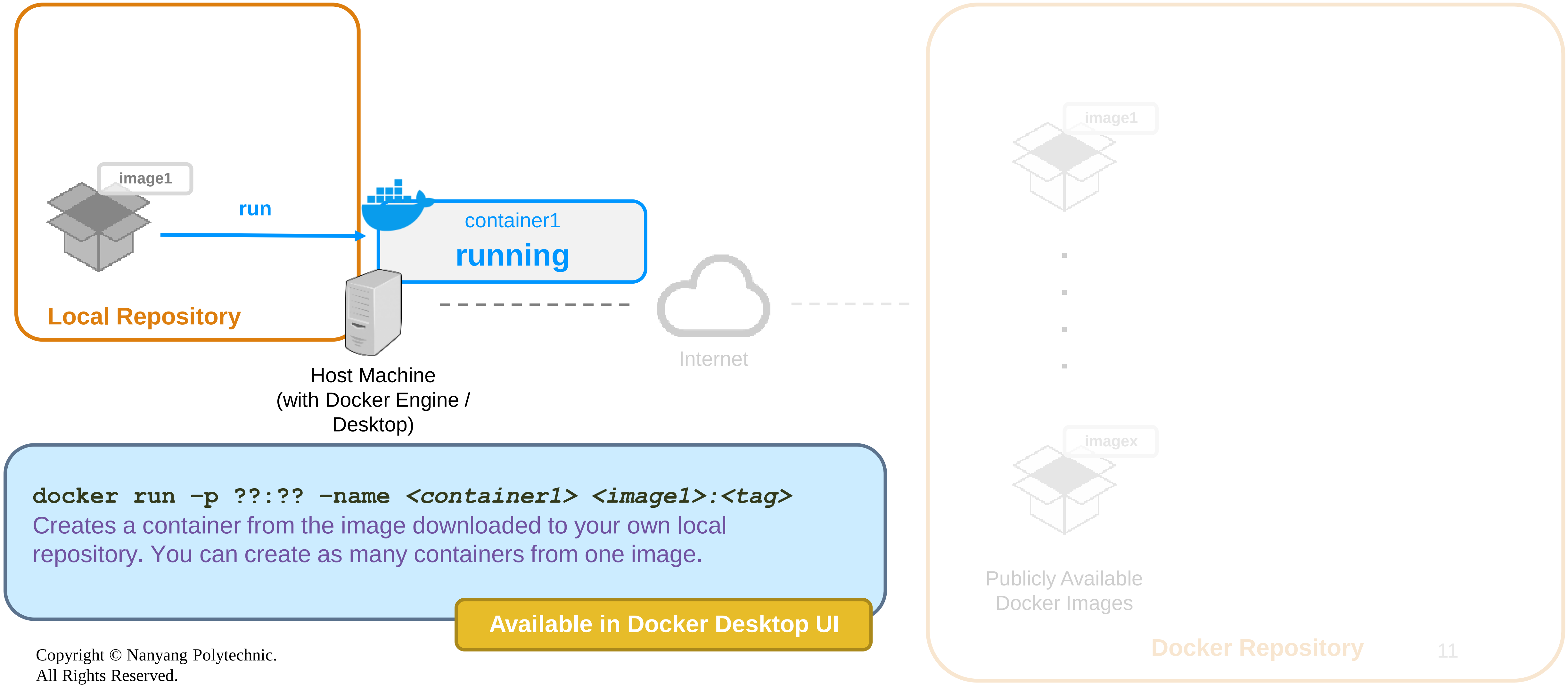
Docker Command - Pull



`docker pull <image1>:<tag>`

Pulls a Docker image from a cloud-based repository to your own host machine

Docker Command - Run

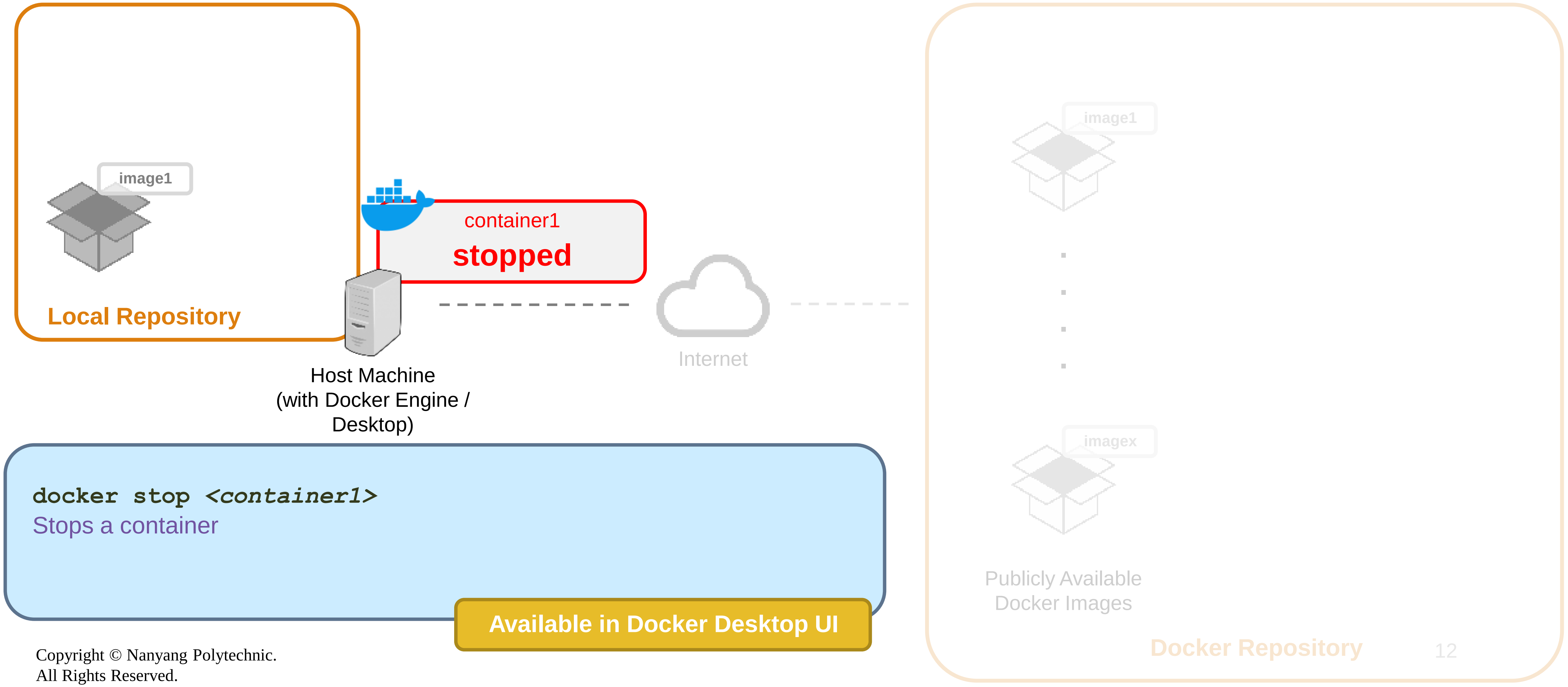


```
docker run -p ??:?? -name <container1> <image1>:<tag>
```

Creates a container from the image downloaded to your own local repository. You can create as many containers from one image.

Available in Docker Desktop UI

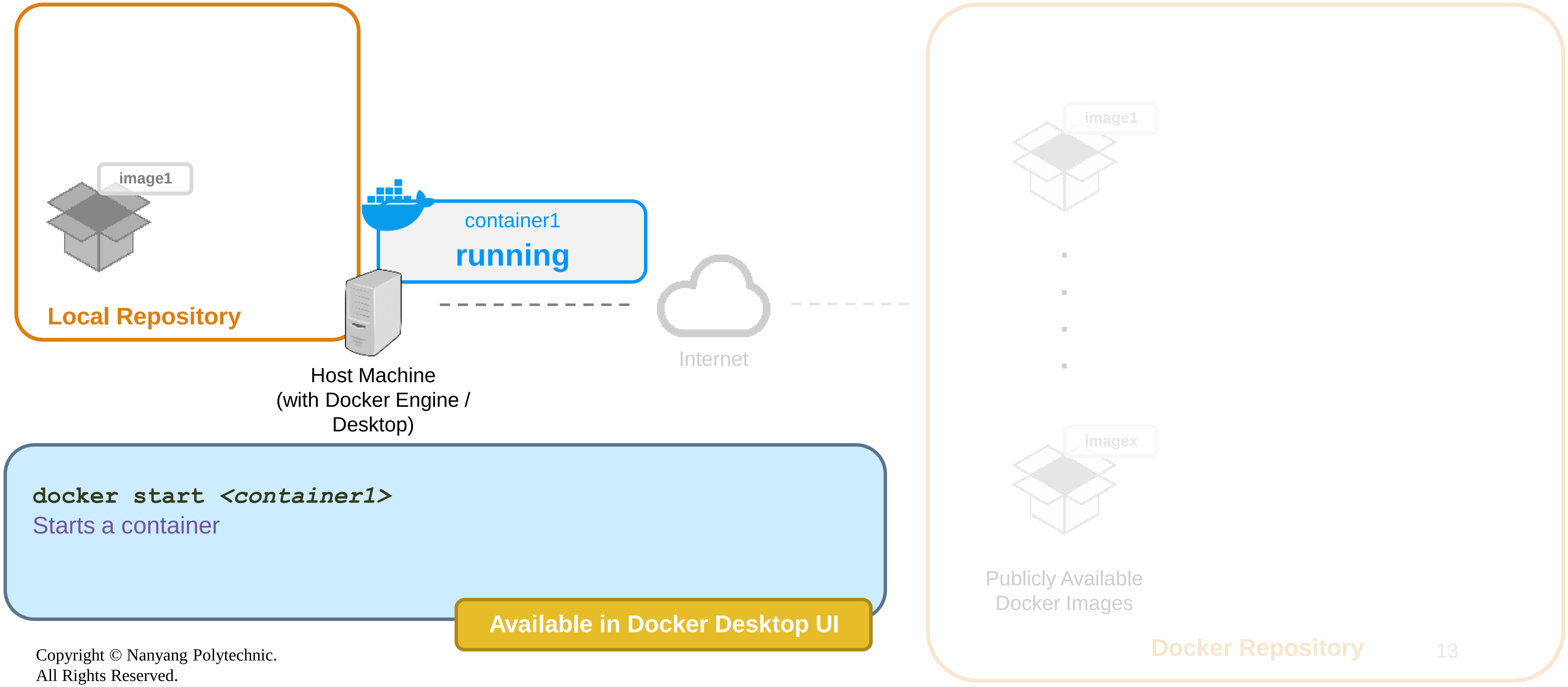
Docker Command - Stop



```
docker stop <container1>
```
Stops a container

Available in Docker Desktop UI

Docker Command - Start

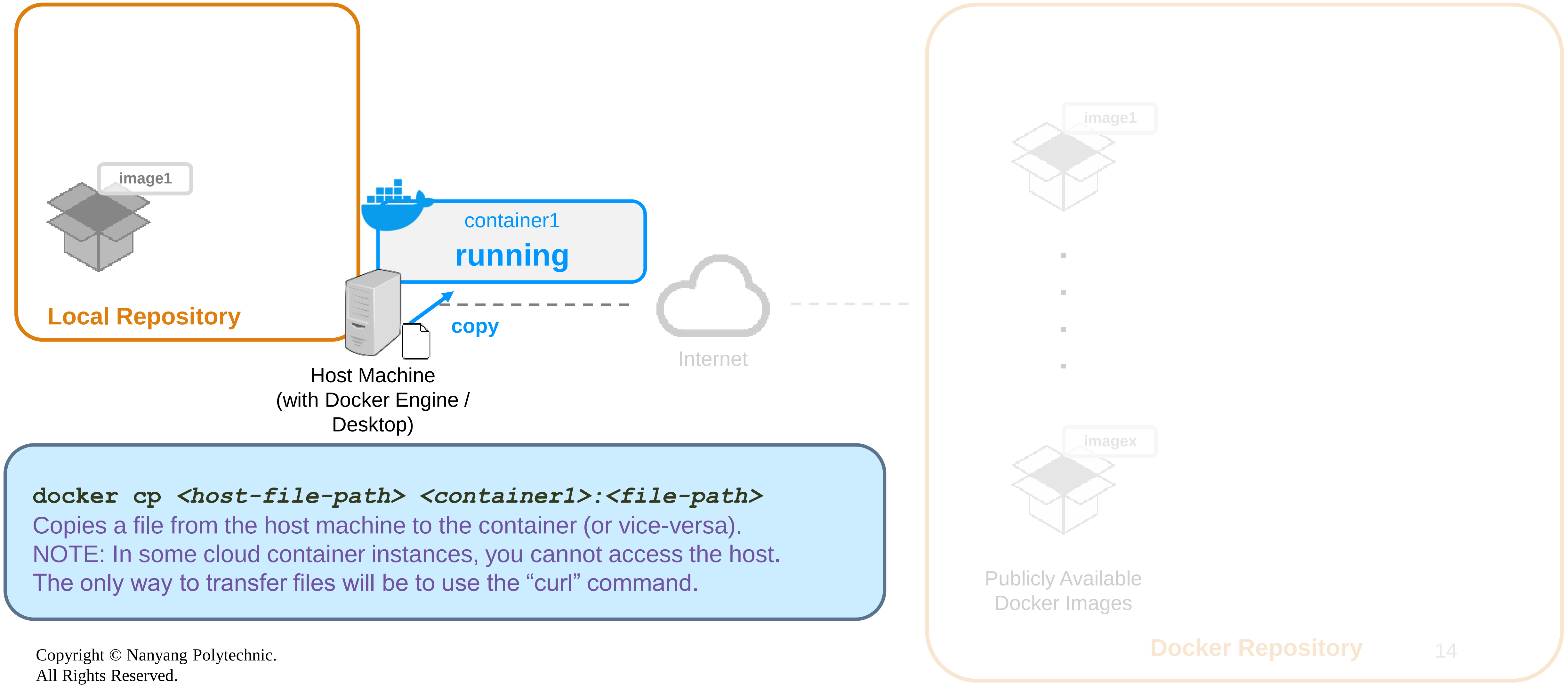


```
docker start <container1>
```

Starts a container

Available in Docker Desktop UI

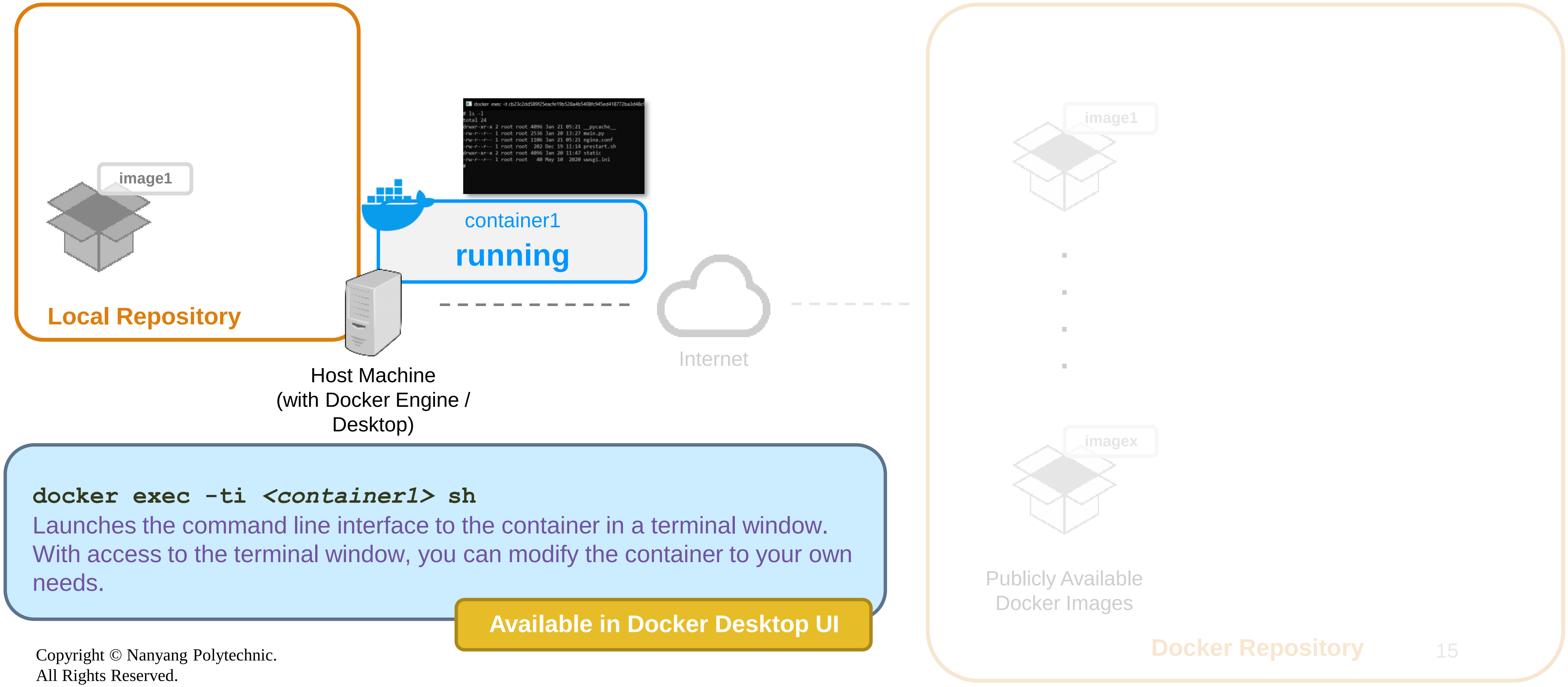
Docker Command - Copy



```
docker cp <host-file-path> <container1>:<file-path>
```

Copies a file from the host machine to the container (or vice-versa).
NOTE: In some cloud container instances, you cannot access the host. The only way to transfer files will be to use the “curl” command.

Docker Command - Exec



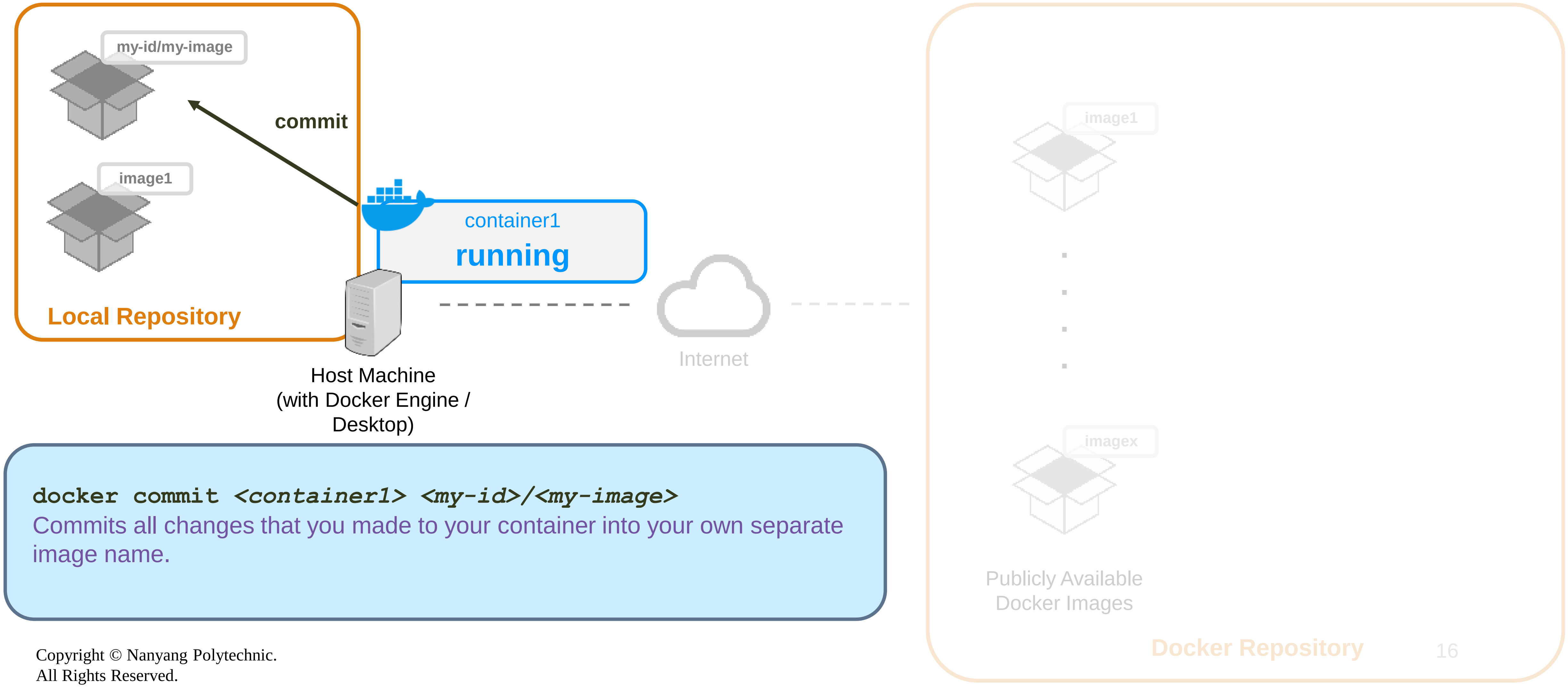
```
docker exec -it cb23c2dd589f25eacfe19b528a4b5408fc945ed418772ba3d48c
# ls -l
total 24
drwxr-xr-x 2 root root 4096 Jan 21 05:21 __pycache__
-rw-r--r-- 1 root root 2536 Jan 20 13:27 main.py
-rw-r--r-- 1 root root 1106 Jan 21 05:21 nginx.conf
-rw-r--r-- 1 root root  202 Dec 19 11:14 prestart.sh
drwxr-xr-x 2 root root 4096 Jan 20 11:47 static
-rw-r--r-- 1 root root   40 May 10  2020 uwsgi.ini
#
```

`docker exec -ti <container1> sh`

Launches the command line interface to the container in a terminal window. With access to the terminal window, you can modify the container to your own needs.

Available in Docker Desktop UI

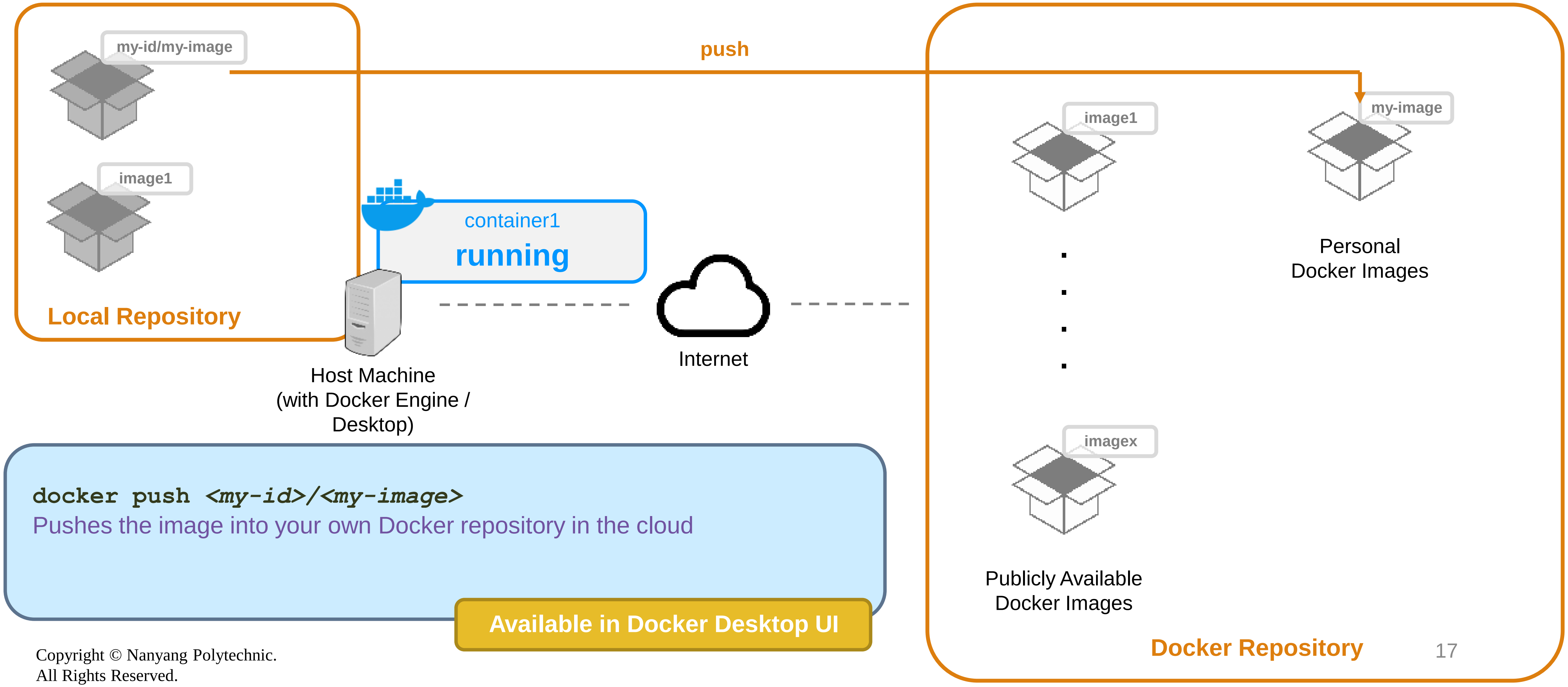
Docker Command - Commit



```
docker commit <container1> <my-id>/<my-image>
```

Commits all changes that you made to your container into your own separate image name.

Docker Command - Push



```
docker push <my-id>/<my-image>
```

Pushes the image into your own Docker repository in the cloud

Available in Docker Desktop UI

Docker References

- Docker Command Line Reference:
  - https://docs.docker.com/engine/reference/commandline/cli/

- Dockerfile Reference:
  - https://docs.docker.com/engine/reference/builder/

Pros and Cons

- Pros of Docker:

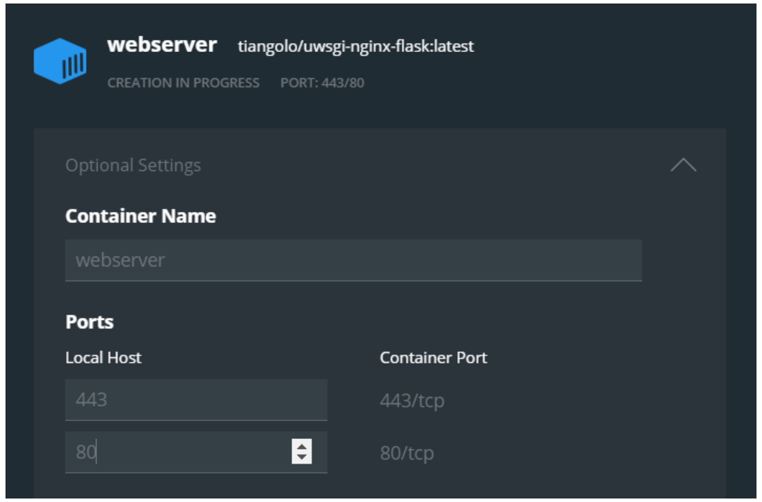
– Portable:	Once tested, you can deploy the image anywhere quickly
– Performance:	Small footprint with very low overhead over the host operating system compared to virtual machines.
– Isolation:	Multiple dockers on the same machine are isolated from each other. Your software runs predictably even if other Docker instances are updated.
– Agility:	Fast to create and start-up
– Scalable:	Automatically launch/take-down docker instances across large clusters of machines

Pros and Cons

- Cons of Docker:
  - Backup? Require backup and restore strategies if the Docker instance contains modified data.
  - No GUI: Graphical applications cannot run in a Docker environment
  - Security? Security needs to be assessed and monitored. If data you want to store is valuable, Docker may not be for you.
  - Usability? The default Docker project is entirely command-line driven.

Publishing Port-to-Port Mapping

- In order for the outside world to communicate with the Docker instance:
  - Expose a specific external port to the internal Docker port
  - Specify the mappings when creating the Docker container
  - Example:
    `docker create --name <name> -p 80:80 <image-name>`


webserver tiangolo/uwsgi-nginx-flask:latest
CREATION IN PROGRESS PORT: 443/80

Optional Settings

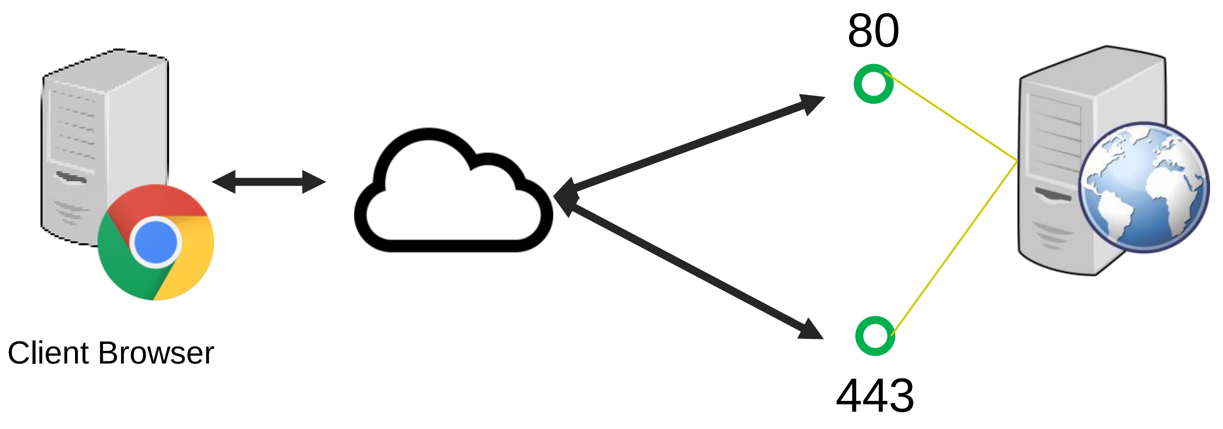
Container Name

webserver

Ports

Local Host	Container Port
443	443/tcp
80	80/tcp

Publishing Port-to-Port Mapping

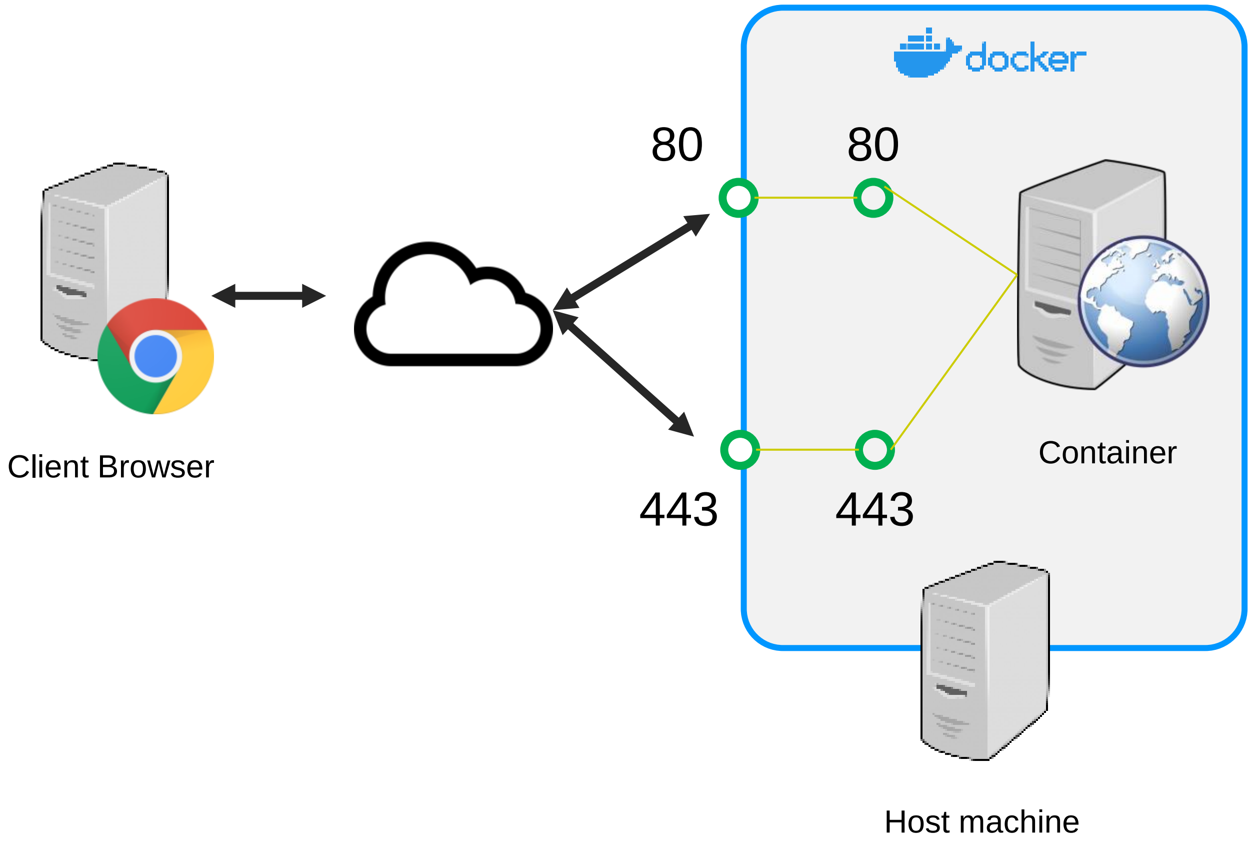


Servers respond to traffic at specific ports.

Example: Web servers normally listen for incoming requests at port 80 (for non-secure) / 443 (for secure)

All other ports are closed off to the outside world for security reasons.

Publishing Port-to-Port Mapping



In a Docker container, however, all incoming connections from the outside world is sealed off, unless ports are specifically opened.

You can expose or publish ports through Docker into the container.

These published ports can use a different number from the actual port in your container server.

Deploying a Deep Learning Model

TENSORFLOW MODEL SERVER

What is a Tensorflow “Server”

- Tensorflow “Server”:
  - A server that serves Tensorflow models
  - When set up, it listens for:
    - gRPC requests over port 8500
    - RESTful API requests over port 8501

Tensorflow “Server” as a Pre-Built Docker Image

- Google’s Tensorflow Model Server is available as a Docker image at hub.docker.com called “tensorflow/serving”

- You can pull the image directly this way:
  - docker pull tensorflow/serving

- Subsequently, use Docker Desktop to create a new container with the downloaded image.

Uploading Your Saved Model

- Tensorflow Model Server accepts only “.savedmodel” files

- To export a .savedmodel from Keras:
  - model.save(“mymodel.savedmodel”)

Copying Your Model into the Docker Container

- Tensorflow Model Server requires the models to be stored in the following structure:

```
models\
    model\
        1\
            saved_model.pb
            variables\
                variables.data-00000-of-...
                variables.index
```

The model name – you can give it any name, but we will use “model” as default

Each Tensorflow model can have multiple versions. You must have 1 level of folders to indicate the version

The saved_model.pb and the variables files are saved in the version folder

Accessing Your Model from Postman

- To perform prediction:

```
// Predict using the latest model version
POST http://localhost:8501/v1/models/model:predict


// Predict using a specific model version
POST http://localhost:8501/v1/models/model/versions/1:predict
```

The model name as determined by your folder name

The version number as determined by your folder name

Accessing Your Model from Postman

- Input Data:

```
{ "instances":
  [
     [ input 1 tensor ],
     [ input 2 tensor ],
     ...
  ]
}
```

- Output Predictions

```
{ “predictions":
  [
     [ prediction 1 tensor ],
     [ prediction 2 tensor ],
     ...
  ]
}
```

Important Note:

- To use Nvidia GPUs:
  - Your host OS must be a Linux operating system
  - You must install Nvidia Docker
    https://github.com/NVIDIA/nvidia-docker
  - Install the “tensorflow/serving:latest-gpu” Docker image
    https://hub.docker.com/r/tensorflow/serving

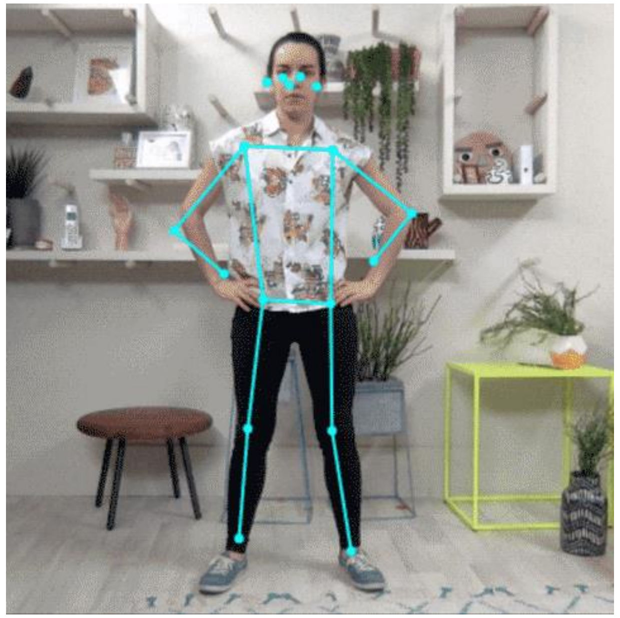
Deploying a Deep Learning Model

INTEGRATING WITH A CLIENT APPLICATION

Pose Estimation with Tensorflow Lite

- We use Javascript to enable the Webcam and connect it to PoseNet

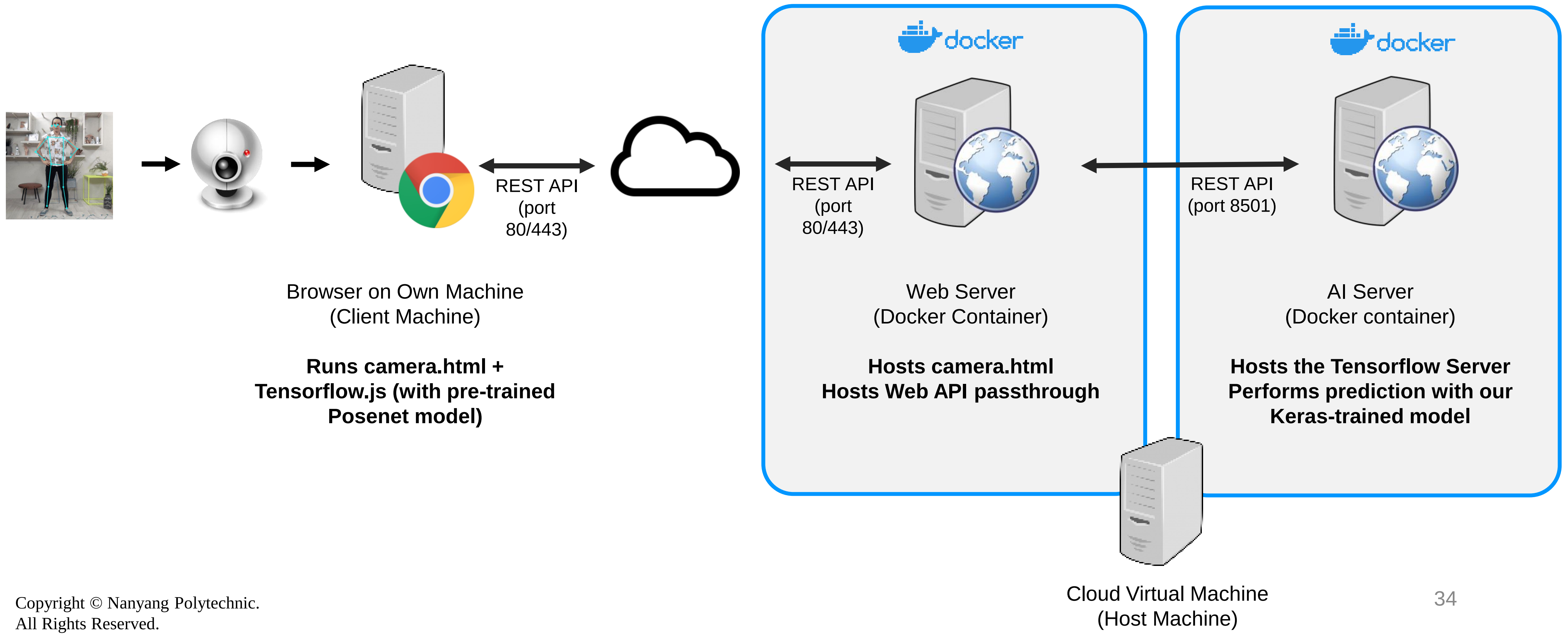
- PoseNet is a pre-trained Convolutional Neural Network model that extracts key joint positions from a photograph of a person.



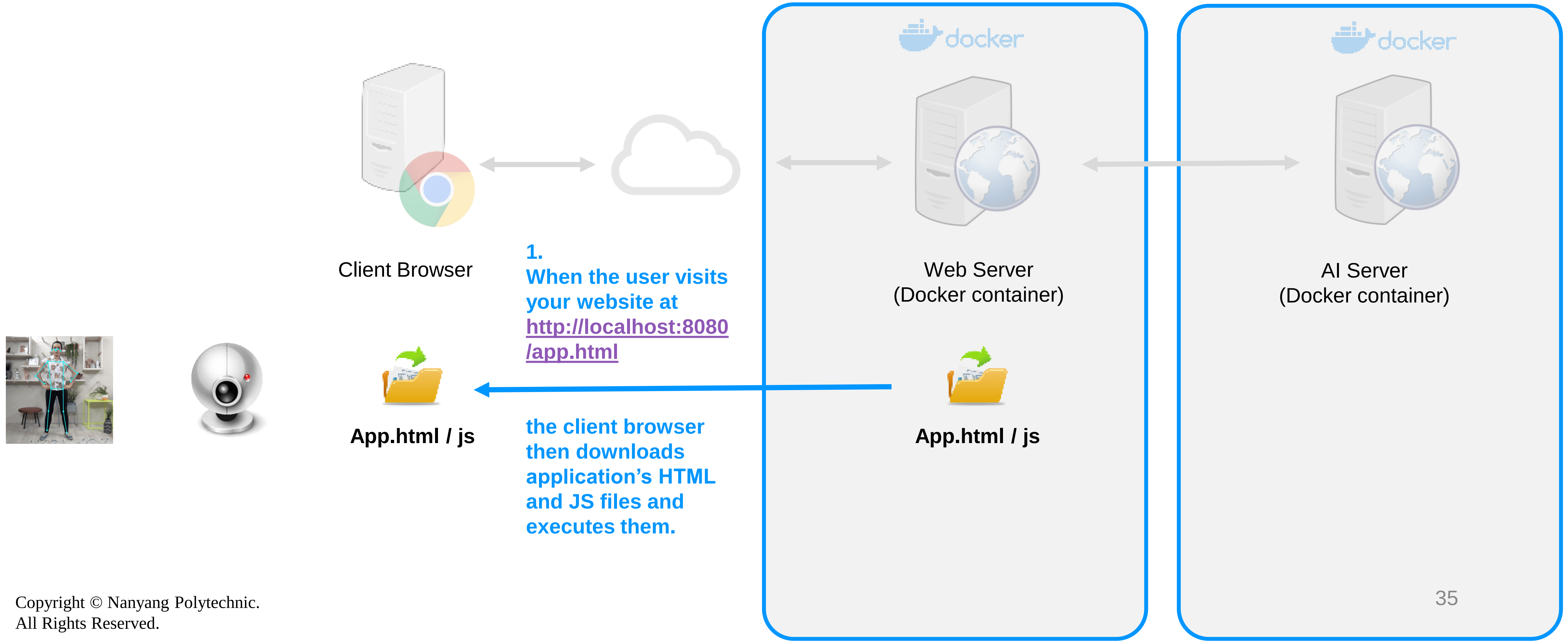
https://www.tensorflow.org/lite/models/pose_estimation/overview

https://arxiv.org/pdf/1803.08225.pdf

Solution Architecture Overview



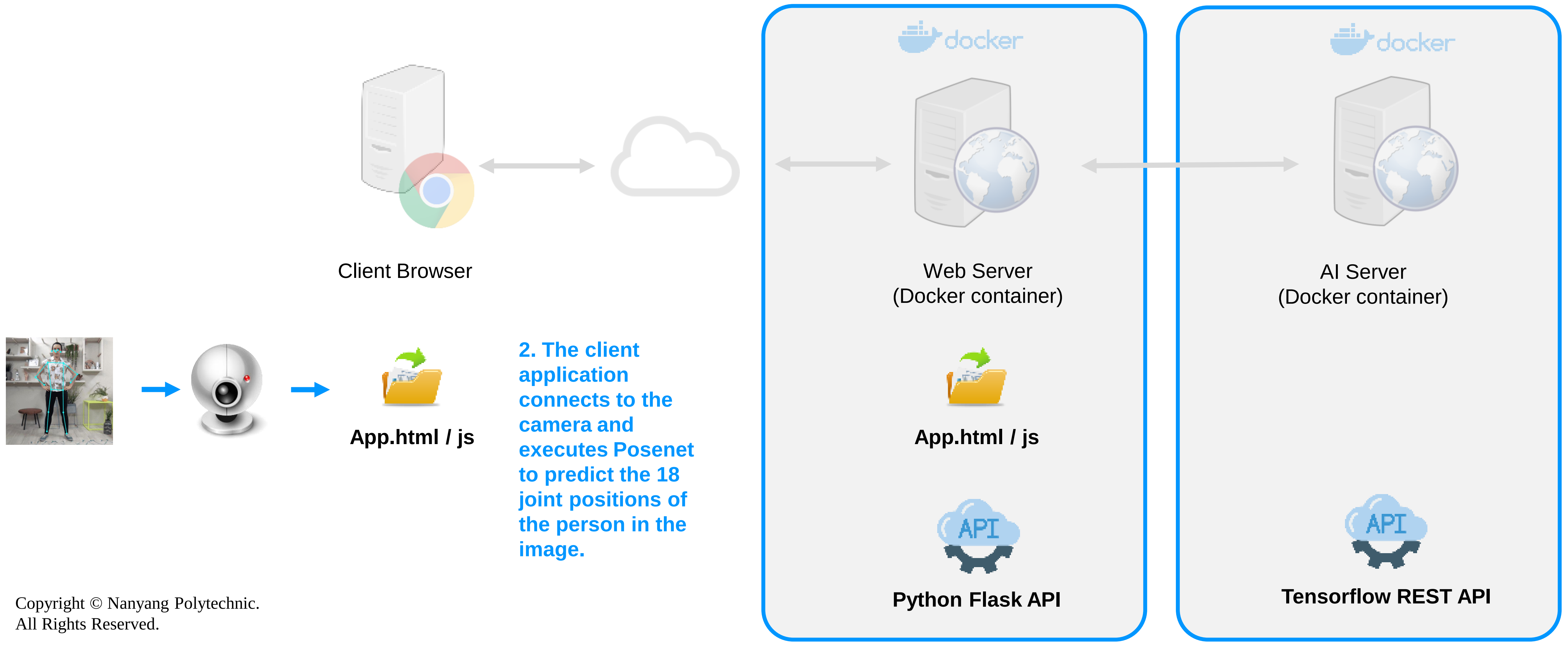
Solution Architecture Overview



1.
When the user visits your website at http://localhost:8080/app.html

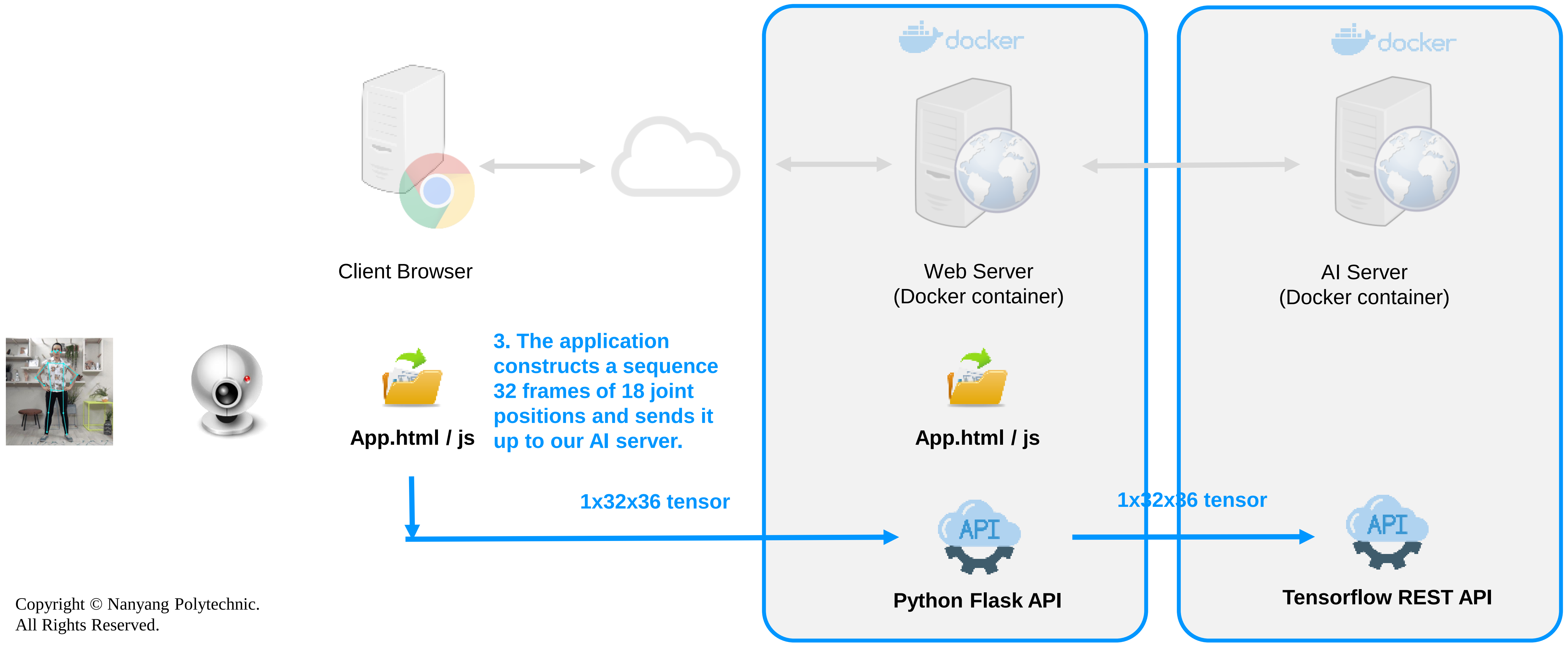
the client browser then downloads application's HTML and JS files and executes them.

Solution Architecture Overview



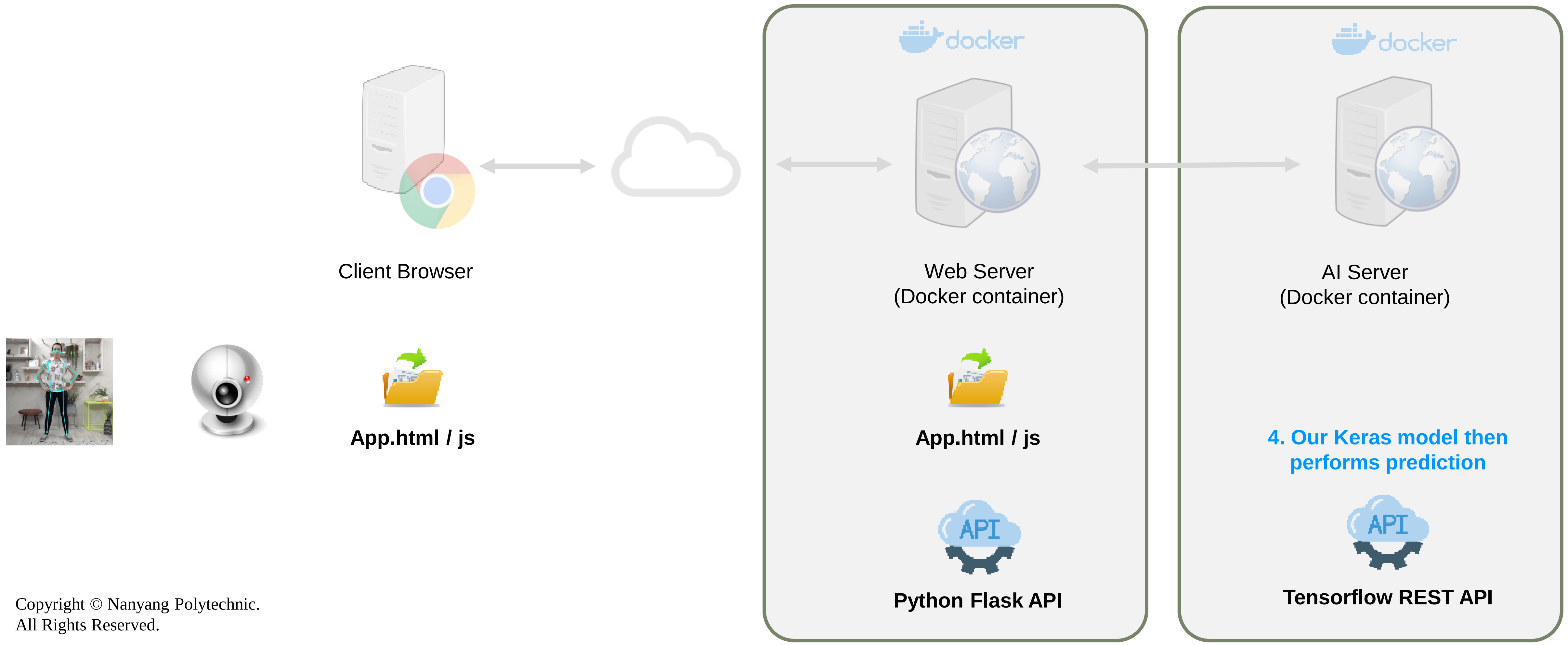
2. The client application connects to the camera and executes Posenet to predict the 18 joint positions of the person in the image.

Solution Architecture Overview

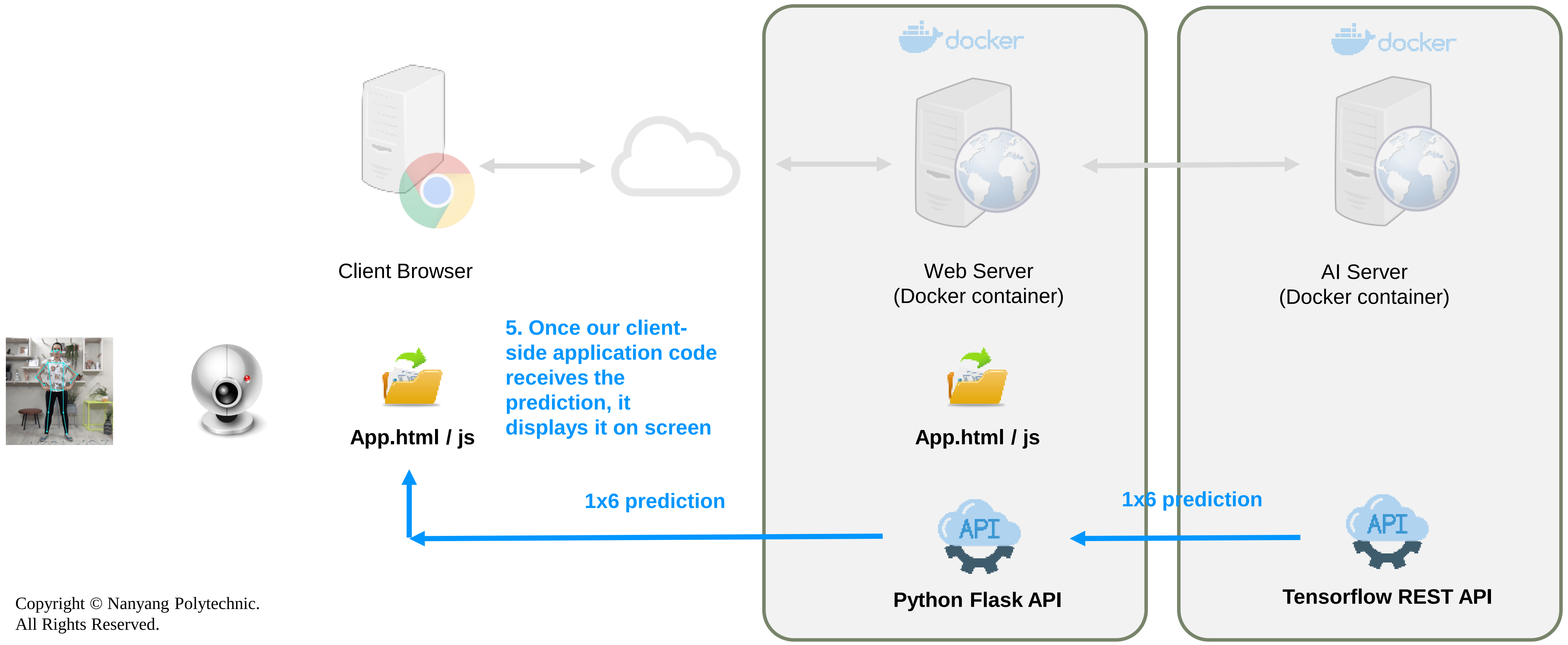


3. The application constructs a sequence 32 frames of 18 joint positions and sends it up to our AI server.

Solution Architecture Overview



Solution Architecture Overview



5. Once our client-side application code receives the prediction, it displays it on screen

Deploying Docker Images to the Cloud

- Once Docker containers are built, you can publish it as an image into a Docker repository.

- We will show an example of how you can pull your own image back down and run a container from it

- Various cloud providers (AWS, Google, Microsoft) offer Docker services to quickly launch Docker containers. Refer to their documentation for further instructions.

Sub topics

- Using Docker
- Tensorflow Server
- Integrating with a Client Application