

BIG DATA: Atelier2

Docker

Raouf ben abdelwahed

Images

Docker Images

- **Like blueprints for containers**

- Runtime environment

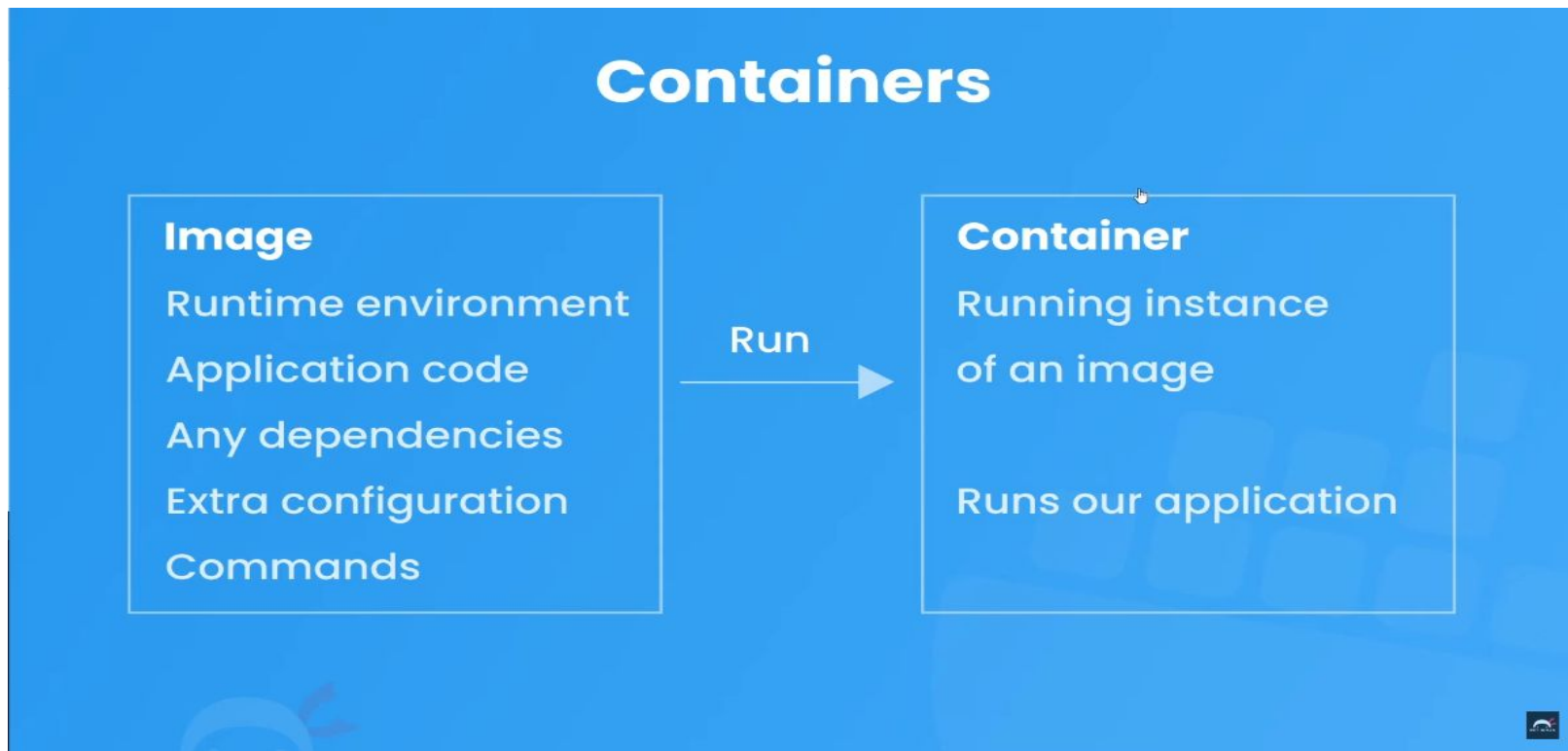
- Application code

- Any dependencies

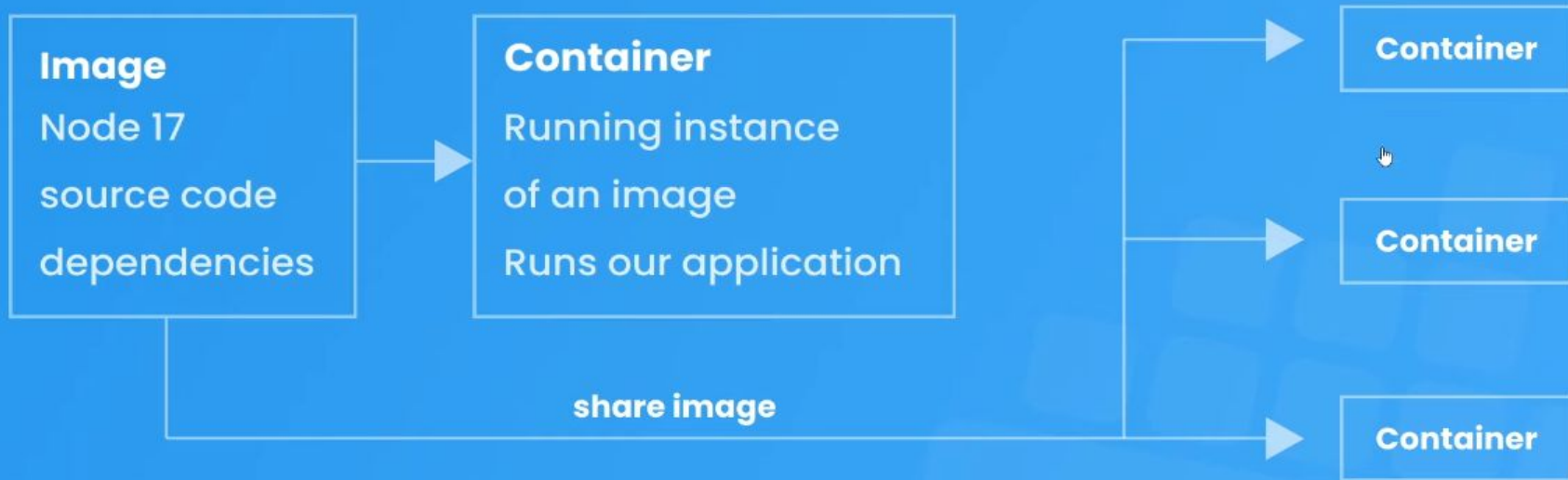
- Extra configuration (e.g. env variables)

- Commands

Conteneurs



Containers

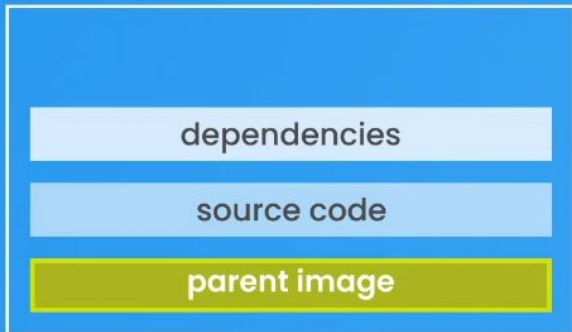


images ?

Docker Images

Images are made up from several "layers"

Image



Parent image:

Includes the OS & sometimes the runtime environment

Image Docker : dockerhub

hub.docker.com/search?type=image

docker hub Search for great content (e.g., mysql) Explore Pricing Sign In Sign Up

Docker Containers Plugins

Filters 1 - 25 of 8,769,085 available images. Suggested


Images


☐ Verified Publisher *Official Images Published By Docker*

☐ Official Images *Official Images Published By Docker*

Categories

- ☐ Analytics
- ☐ Application Frameworks
- ☐ Application Infrastructure
- ☐ Application Services
- ☐ Base Images
- ☐ Databases
- ☐ DevOps Tools

nginx  1B+ Downloads 10K+ Stars
Updated a month ago
Official build of Nginx.
Container Linux 386 ARM IBM Z ARM 64 mips64le x86-64 PowerPC 64 LE Application Infrastructure

alpine  1B+ Downloads 8.4K Stars
Updated 2 months ago
A minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in size!
Container Linux PowerPC 64 LE ARM 64 ARM x86-64 IBM Z 386 riscv64 Featured Images Base Images

Installer Docker

- Pour vérifier l'installation: `docker --version`
- Windows: téléchargez et installez Docker Desktop à partir du site web officiel (<https://www.docker.com/products/docker-desktop>)
- Linux:

`sudo apt update`

`sudo apt install -y docker.io`

`sudo systemctl start docker`

`sudo systemctl enable docker`

Lancement de conteneurs

- dans le terminal, taper: `docker run hello-world`

La commande `docker run` est utilisée pour créer et exécuter un conteneur Docker à partir d'une image Docker spécifique. Elle permet de lancer une instance d'une image Docker en tant que processus *isolé et autonome*.

- Créez un premier conteneur à partir de l'image Ubuntu :

`docker run -it --name mon-premier-conteneur ubuntu`

- L'option `-it` permet une interaction en mode texte avec le conteneur.
- Le nom `mon-premier-conteneur` est attribué au conteneur.

Les principales commandes Docker

1. Commandes Docker de base

- `docker --version` : Vérification de l'installation.
- `docker pull` : Téléchargement d'images Docker.
- `docker images` : Liste des images disponibles.
- `docker rmi` : Supprimer une image Docker de votre système
- `docker ps` : Liste des conteneurs en cours d'exécution.
- `docker run` : Création et exécution d'un conteneur.

2. Cycle de vie des conteneurs

- `docker start` et `docker stop` : Démarrage et arrêt des conteneurs.
- `docker restart` : Redémarrage des conteneurs.
- `docker rm` : Suppression des conteneurs. l'option `-f` force la suppression.
- `docker logs` : Consultation des journaux des conteneurs.

exercice

- créer un conteneur monconteneur1 avec l'image ubuntu.
- créer un conteneur monconteneur2 avec la même image.
- supprimer le premier conteneur
- démarrer le deuxième conteneur en mode interactif (-it) et y exécuter une session shell.
- À l'intérieur du conteneur, exécutez des commandes pour afficher la version d'Ubuntu et le contenu du répertoire /.
- quitter la session shell du conteneur.
- afficher la liste des conteneurs en cours d'exécution.
- afficher la liste de tous les conteneurs.
- Utilisez les commandes `docker stop` et `docker rm` pour arrêter et supprimer le conteneur
- Supprimer l'image ubuntu de votre système.

Image personnalisée: Dockerfile

Dans le terminal: créer un dossier **tic** dans lequel on crée un dossier **mon-site-web**

1. accéder à ce dossier et y créer un fichier HTML **index.html**:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset='utf-8'>
  <meta http-equiv='X-UA-Compatible' content='IE=edge'>
  <title>Page Title</title>
  <meta name='viewport' content='width=device-width, initial-scale=1'>
</head>
<body>
  <p>ceci est un test de page web avec Docker</p>
</body>
</html>
```

2. revenir au dossier **tic** et y créer un fichier nommé **Dockerfile** avec le contenu suivant:

Utilisez une image de base (par exemple, Ubuntu)

FROM ubuntu

Exécutez une commande d'installation

RUN apt-get update && apt-get install -y nginx

Copiez un fichier local dans l'image

COPY mon-site-web /var/www/html/

Exposez le port 80 pour le trafic web

EXPOSE 80

Commande à exécuter lorsque le conteneur démarre

CMD ["nginx", "-g", "daemon off;"]

- construire l'image

`docker build -t mon-site-web .`

remarque: le point (.) dans cette commande signifie que le Dockerfile se trouve dans le répertoire actuel où vous exécutez la commande, et Docker devrait utiliser ce Dockerfile pour construire l'image.

- créer et exécuter le conteneur

`docker run -d -p 8080:80 --name mon-site-web-conteneur mon-site-web`

- L'option `-t` permet d'attribuer un nom à l'image.
 - L'option `-d` permet de détacher le conteneur.
 - L'option `-p` relie le port 8080 de l'hôte au port 80 du conteneur.
 - Le nom `mon-site-web-conteneur` est attribué au conteneur.
-
- ouvrir un navigateur et d'accéder à `http://localhost:8080` pour voir le site web fonctionner.

Création d'une image personnalisée: un conteneur Python

1. créer un dossier mon-site-python et y accéder:

```
mkdir mon-site-python
```

```
cd mon-site-python
```

2. Créez un fichier Python nommé `app.py` contenant une application web simple avec Flask:

```
from flask import Flask
```

```
app = Flask(__name__)
```

```
@app.route('/')
```

```
def hello_world():
```

```
    return 'Hello, Docker with Python!'
```

```
if __name__ == '__main__':
```

```
    app.run(debug=True, host='0.0.0.0')
```

3. Créez un fichier `requirements.txt` :

`Flask==2.0.2`

`Werkzeug==2.0.2`

4. Créez un Dockerfile pour l'application Python `Dockerfile` dans le même répertoire :

Utilisez une image Python

FROM python:3.9-slim-buster

Répertoire de travail dans le conteneur

WORKDIR /app

Copiez le fichier des exigences et installez les dépendances

COPY requirements.txt requirements.txt

RUN pip install -r requirements.txt

Copiez l'application Python dans le conteneur

COPY . .

Commande pour exécuter l'application Python

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

6. Construisez l'image à partir du Dockerfile (assurez-vous d'être dans le répertoire `mon-site-python` où se trouvent les fichiers) :

```
docker build -t mon-site-python .
```

remarque: le point (.) dans cette commande signifie que le Dockerfile se trouve dans le répertoire actuel où vous exécutez la commande, et Docker devrait utiliser ce Dockerfile pour construire l'image "mon-site-python".

7. Exécutez un conteneur basé sur cette image, en exposant le port 5000 du conteneur :

```
docker run -d -p 8082:5000 --name mon-site-python-conteneur mon-site-python
```

8. Accédez à l'application Python depuis un navigateur en visitant <http://localhost:8082>. Vous devriez voir le message "Hello, Docker with Python!".