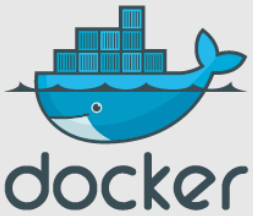


Présentation

Historique

CLI

Configuration

Format

Bonnes pratiques

Limitations

TP

Commandes utiles

Application Web

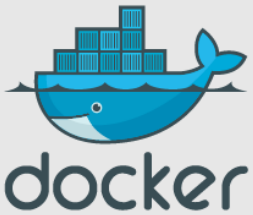
Mode debug

Mode production

Déploiement simple de conteneurs en production : Docker Compose

Cécile Cavet

5 juin 2018



Présentation

Présentation

Historique

CLI

Configuration

Format

Bonnes pratiques

Limitations

TP

Commandes utiles

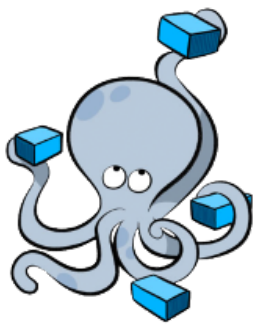
Application Web

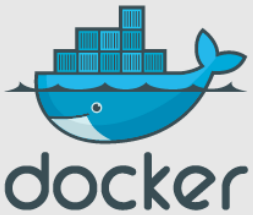
Mode debug

Mode production

But :

- Micro services reliés / applications distribuées en services.
- Applications en production, staging, development, testing.
- Déploiement sur 1 seul noeud ➡ multi-noeuds : Docker stack + Swarm, k8s, Mesos...





Présentation

Présentation

Historique

CLI

Configuration

Format

Bonnes pratiques

Limitations

TP

Commandes utiles

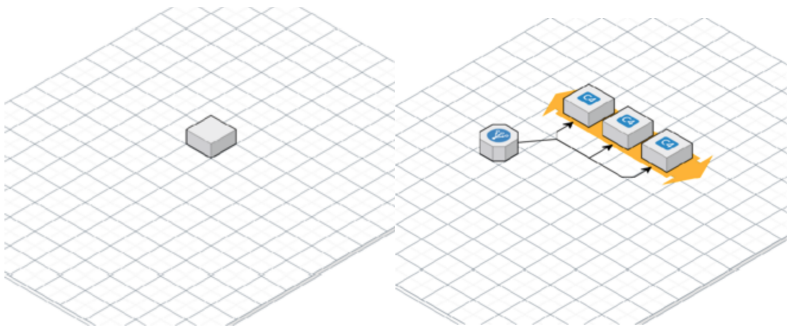
Application Web

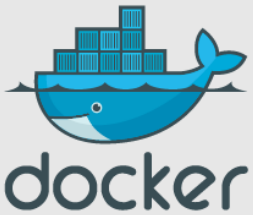
Mode debug

Mode production

Outil d'orchestration léger :

- Plusieurs instances de conteneurs.
- Préservation des volumes de données.
- Recrée seulement les conteneurs modifiés.



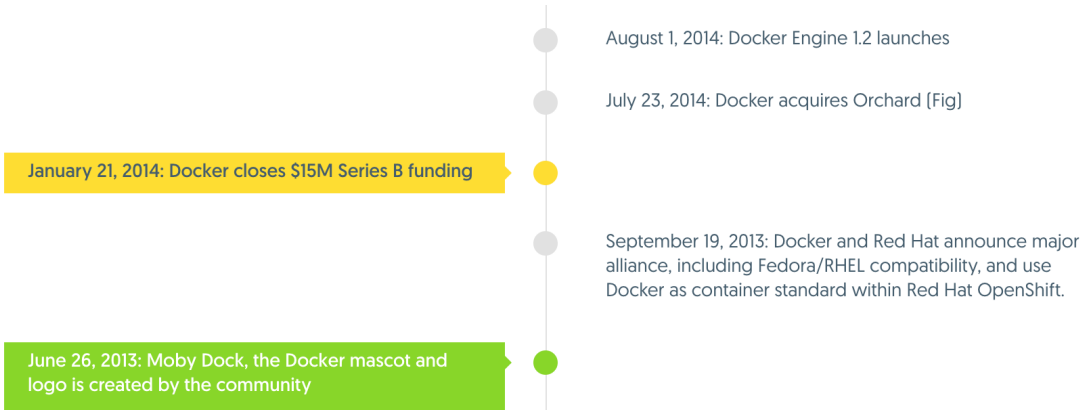


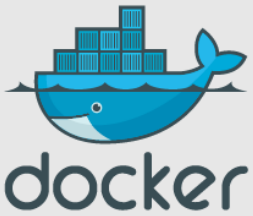
Historique

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

De Fig à Compose :

- Orchard : outil d'orchestration Fig (en Python) depuis 2013.
- Docker : achat d'Orchard en 2014.
- Fig ➡ Compose en 2015, compatible Swarm.
- Format : v2 en 2016 et v3 en 2017, compatible Docker stack.





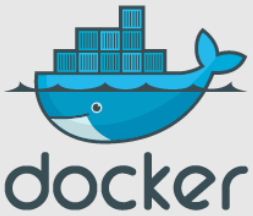
Description

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

CLI :

```
build     Build or rebuild services
help      Get help on a command
kill      Kill containers
logs      View output from containers
port      Print the public port for a port binding
ps        List containers
pull      Pulls service images
rm        Remove stopped containers
run       Run a one-off command
scale     Set number of containers for a service
start     Start services
stop      Stop services
restart   Restart services
up        Create and start containers
```

- Différences avec Docker :
  - Pas de manipulation des images (**save**, **search**, **images**, **import**, **export**, **tag**, **history**).
  - Pas de mode interactif (**attach**, **exec**, **run -i**, **login**, **wait**).



Description

Présentation
- Historique
- CLI
- Configuration
- Format
- Bonnes pratiques
- Limitations

TP
- Commandes utiles
- Application Web
- Mode debug
- Mode production

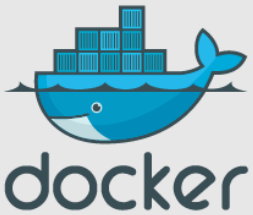
Fichier de configuration :

docker-compose.debug.yml

```yaml
version: "3"
services:
  webapp:
    build:
      context: ./webapp
      dockerfile: Dockerfile.debug
    image: webapp:debug
    container_name: webapp
    env_file:
      - webapp/ecole.env
    ports:
      - "$DJANGO_PORT:8000"
    volumes:
      - media_data:/webapp/media/uploads
      - static_data:/webapp/static

volumes:
  media_data:
  static_data:
```

- Format YAML : **docker-compose.yml**



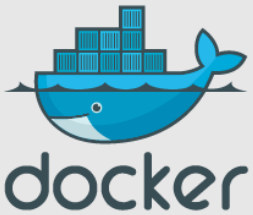
Description

Présentation
- Historique
- CLI
- Configuration
- Format
- Bonnes pratiques
- Limitations

TP
- Commandes utiles
- Application Web
- Mode debug
- Mode production

Fichier de configuration :

- Similitudes avec Docker :
  - exécution : mêmes options que **run** (**nom**, **port**, **volumes**, **réseau**, **environnement**...)
  - micro-services : **service**.
  - construction des images : **build**.
- Différences :
  - interdépendance des conteneurs : **depends_on**
    - découverte des services : nom des services (et alias réseau **container_name**) plutôt que IP/FQDN.
    - ordre d'instantaciation.



Format

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

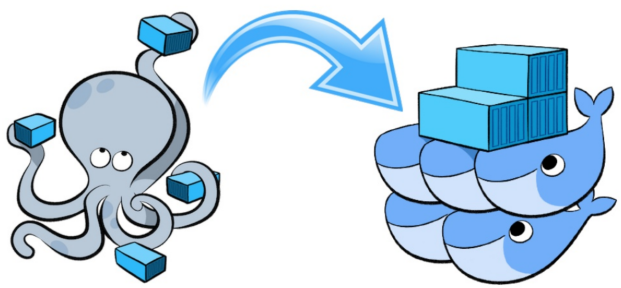
Version :

- En-tete **version** : syntaxe de Docker Compose.
- Version du format :
  - v1 : va être déprécié.
  - v2 : compatible Rancher.
    - New : **services**, **volumes** et **networks** dédiés, arguments spécifiques de **build**.
    - Obsolète : **links**.
  - v3 : même structure que v2, compatible Swarm.
    - New : **deploy**, **secrets**, **config**.
    - Obsolète : **volume_driver**, **volumes_from** et les propriétés étendues.

Format

Nouveautés de v3 : Docker stack + Swarm

- Déploiement : **deploy**
  - Rolling updates, contraintes de placement flexibles, limites des ressources...
- Secrets : **secrets** pour gérer certaines informations.
- Dev/CI/Prod : **override.yml** pour les développeurs.
- Bundle (expérimentale).



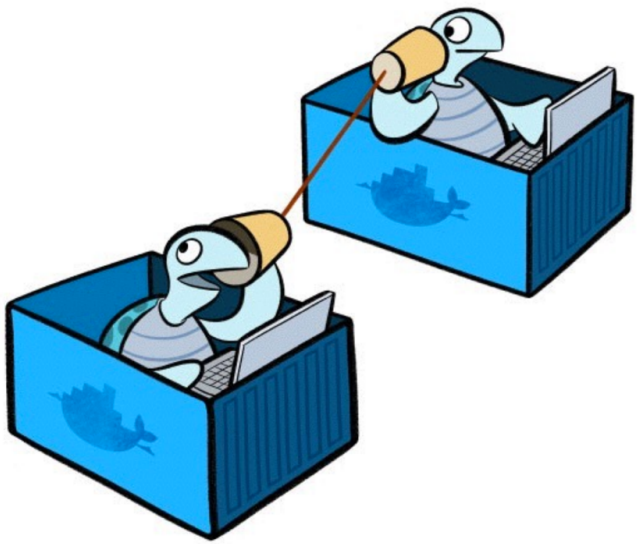
- **mode** – global | replicated
- **replicas**
- **placement** – constraints: node.id/hostname/role/labels, engine.labels
- **update_config** – parallelism, delay, failure_action, monitor, max_failure_ratio
- **resources** – cpu_shares, cpu_quota, cpuset, mem_limit, memswap_limit, mem_swappiness
- **restart_policy** – condition, delay, max_attempts, window
- **labels**

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

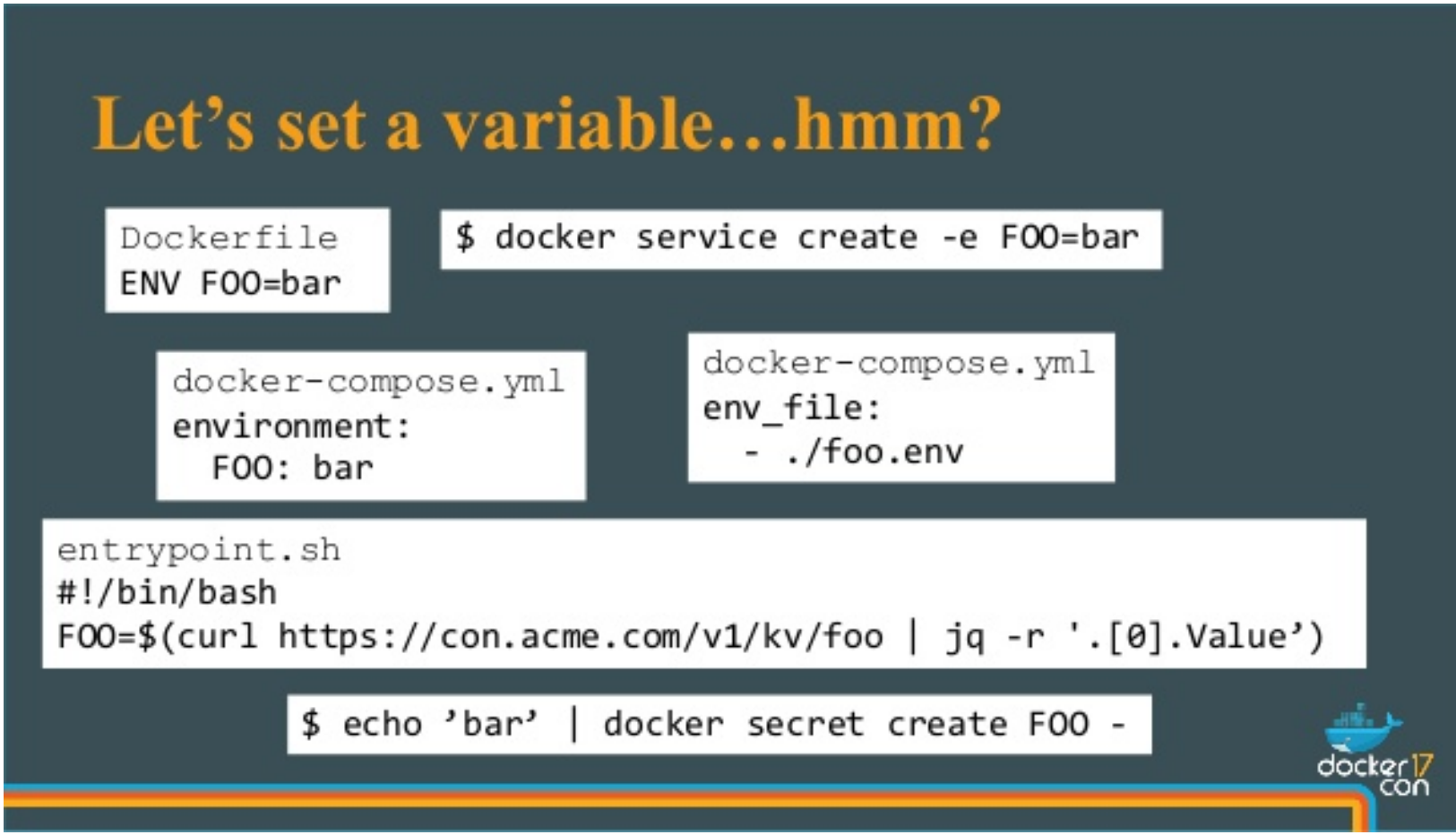
Bonnes pratiques

Quelques fonctionnalités :

- Variables d'environnement.
- Secrets.
- Fichier Compose pour Dev/Prod.

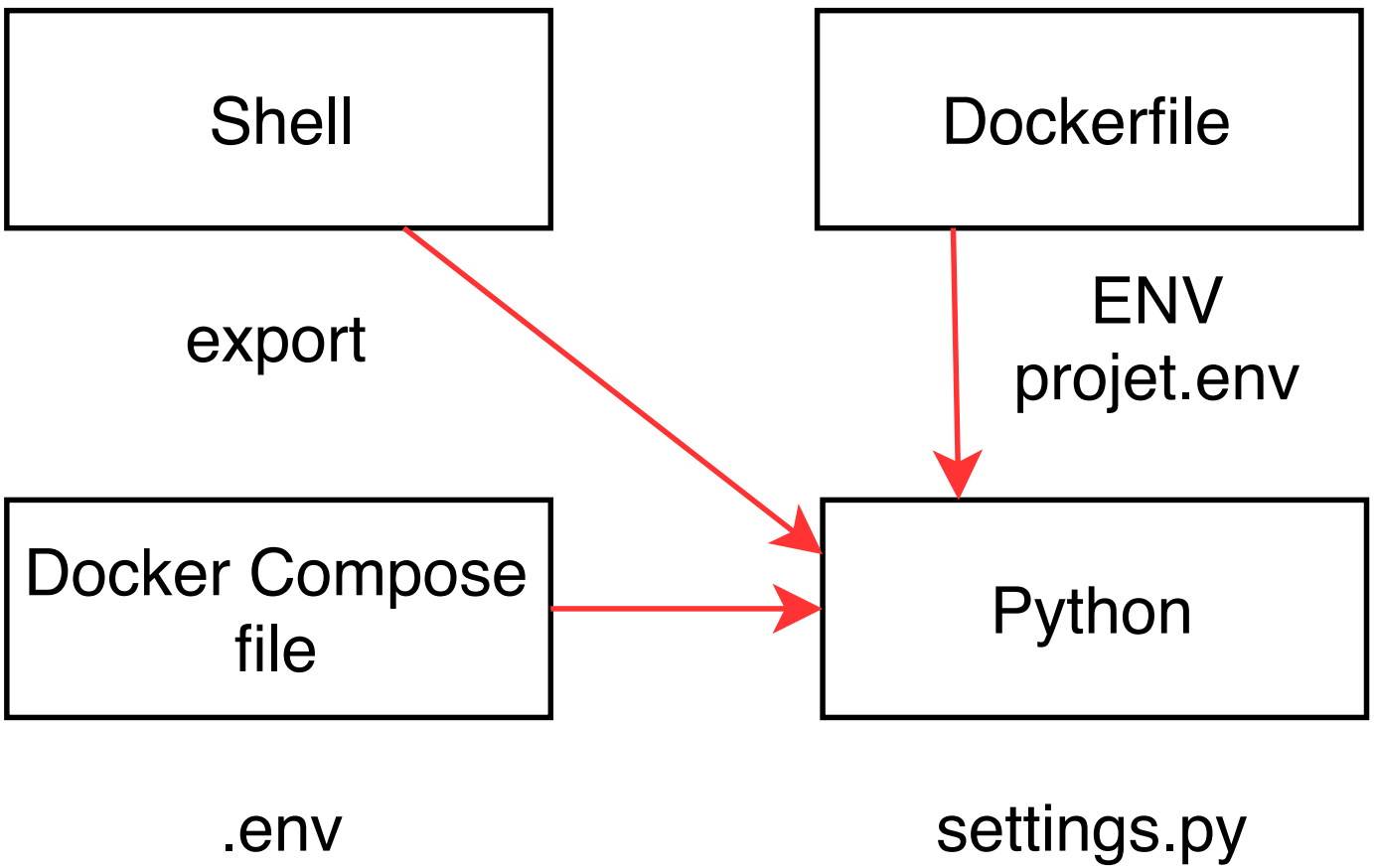


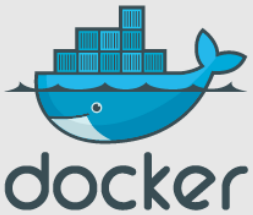
Variables d'environnement



Variables d'environnement

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production



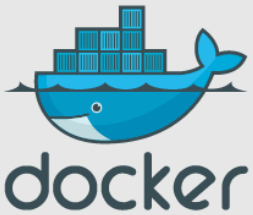


Variables d'environnement

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Docker :

- Shell :
  export DJANGO_SECRET=...
- Fichier : service.env
  DJANGO_SECRET=...
- Dockerfile :
  ENV DJANGO_SECRET {$DJANGO_SECRET}
- Python : os.getenv('DJANGO_SECRET')

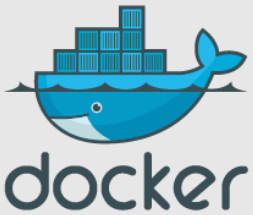


Variables d'environnement

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Docker Compose :

- Fichier : .env
  DJANGO_SECRET=...
- **docker-compose.yml** :
  - Docker :
    env_file:
    - webapp/service.env
  - Compose :
    environment:
    - DJANGO_SECRET=$DJANGO_SECRET
    - DJANGO_SECRET

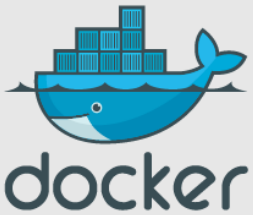


Secrets

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Docker Compose :

- Nouveau depuis le format v3.1 !
- Pour stocker :
  - Nom d'utilisateur et mot de passe.
  - Certificats TLS certificates et clés.
  - Clés SSH.
  - Données : nom de bases de données ou de serveurs internes.
  - Strings ou binaires génériques (jusqu'à 500 KB).
- Fichier secret ou Docker secret.



Fichier Compose en Dev/Prod

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Docker Compose :

- Fichiers multiples : option -f
  - **docker-compose.debug.yml**
  - **docker-compose.production.yml**
- Fichiers multiples v2 : override lut par défaut (remplace ou étend les valeurs).
  - **docker-compose.yml**
  - **docker-compose.override.yml**
- Étendre un fichier de base :

  extends:

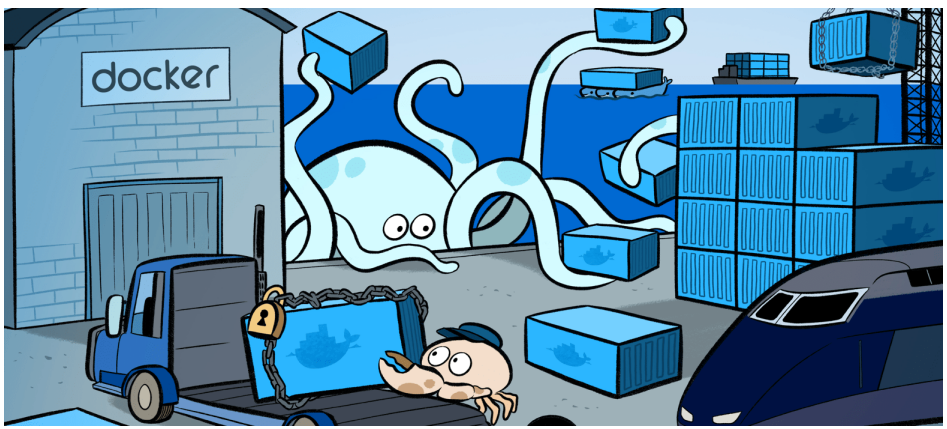
  file: common-services.yml

  service: webapp

Déploiement en production

Limitations de Docker Compose :

- Déploiement sur un noeud unique.
- Pas de rollback possible.
- Pas de monitoring continu de l'état des conteneurs.

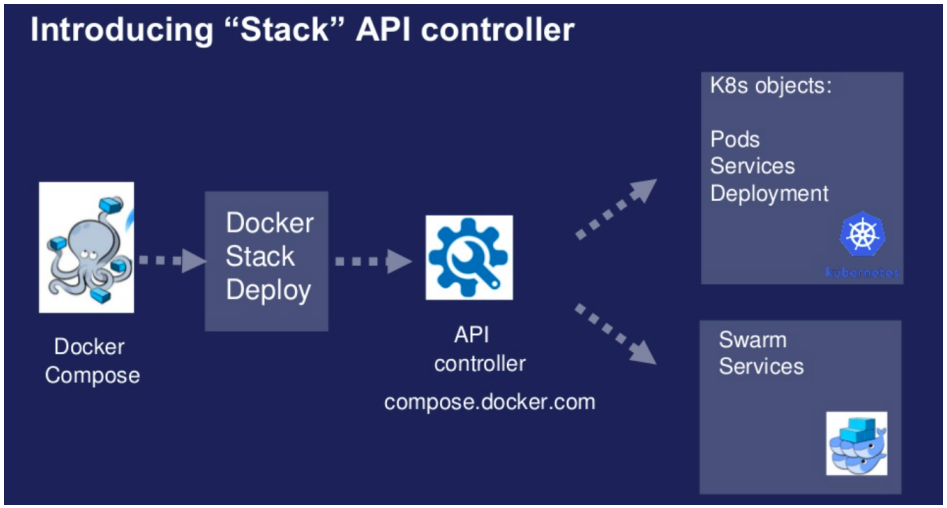


Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Déploiement en production

Orchestrateurs en production :

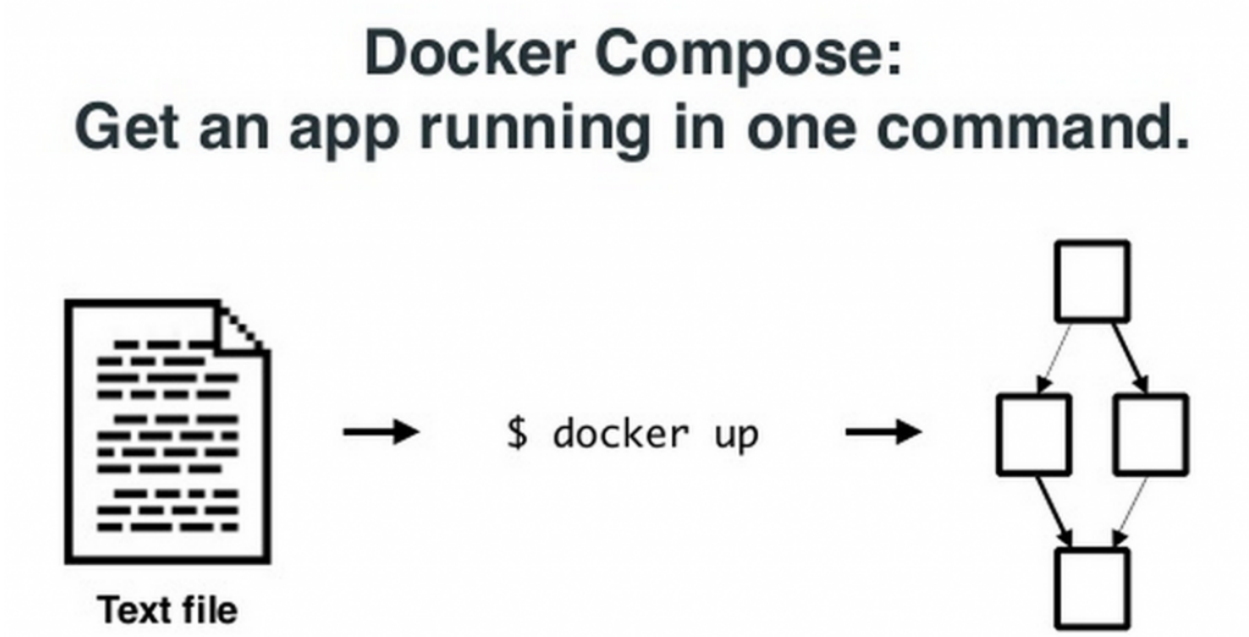
- Swarm : **docker-compose.yml** + stack deploy
- Rancher : **docker-compose.yml** ➡ **rancher-compose.yml**.
- k8s : **docker-compose.yml** ➡ kompose ➡ manifestes k8s.

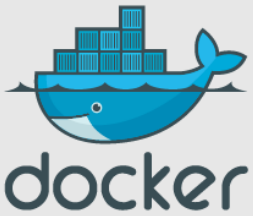


TP

Présentation
- Historique
- CLI
- Configuration
- Format
- Bonnes pratiques
- Limitations

TP
- Commandes utiles
- Application Web
- Mode debug
- Mode production

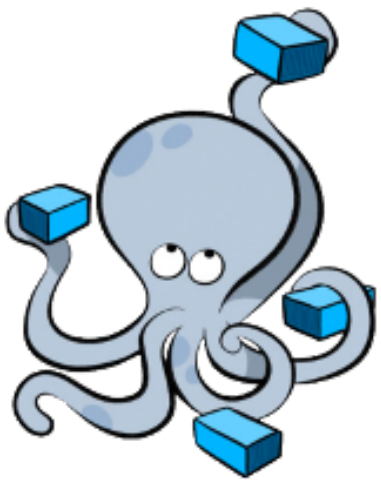




Commandes utiles

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

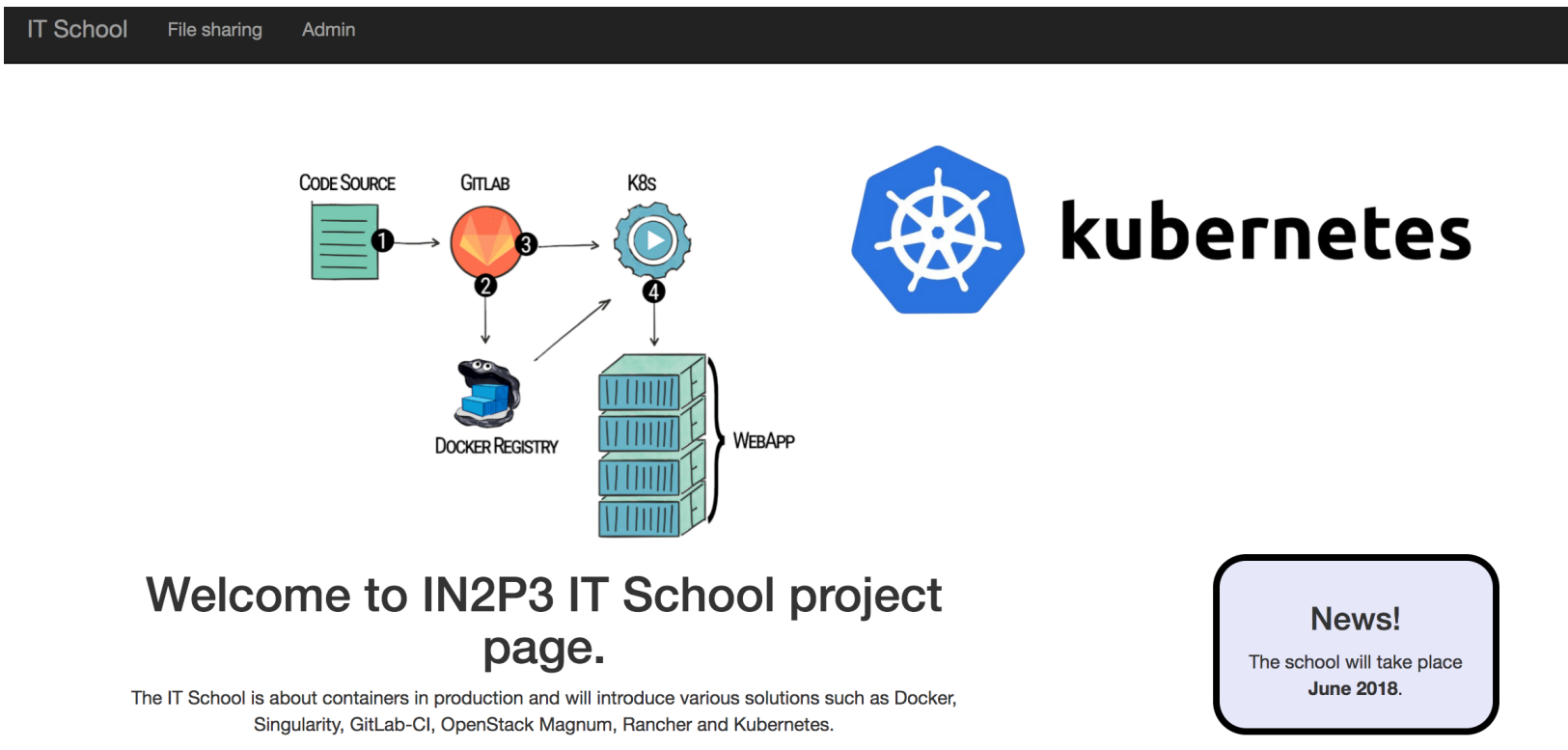

```
$ docker-compose --log-level DEBUG up -d
$ docker-compose up -d --scale service=3
$ docker-compose --compatibility up -d
$ docker-compose down --volumes
$ docker-compose ps
$ docker-compose logs --tail 10 --follow
```

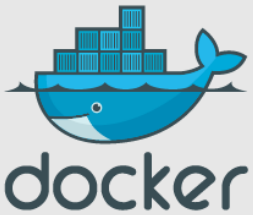


Application Web avec l'environnement Django

Présentation
- Historique
- CLI
- Configuration
- Format
- Bonnes pratiques
- Limitations

TP
- Commandes utiles
- **Application Web**
- Mode debug
- Mode production



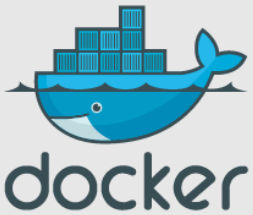


Application Web avec l'environnement Django

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Micro services :

- Django + uWSGI.
- Postgres.
- Nginx.



Application Web avec l'environnement Django

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

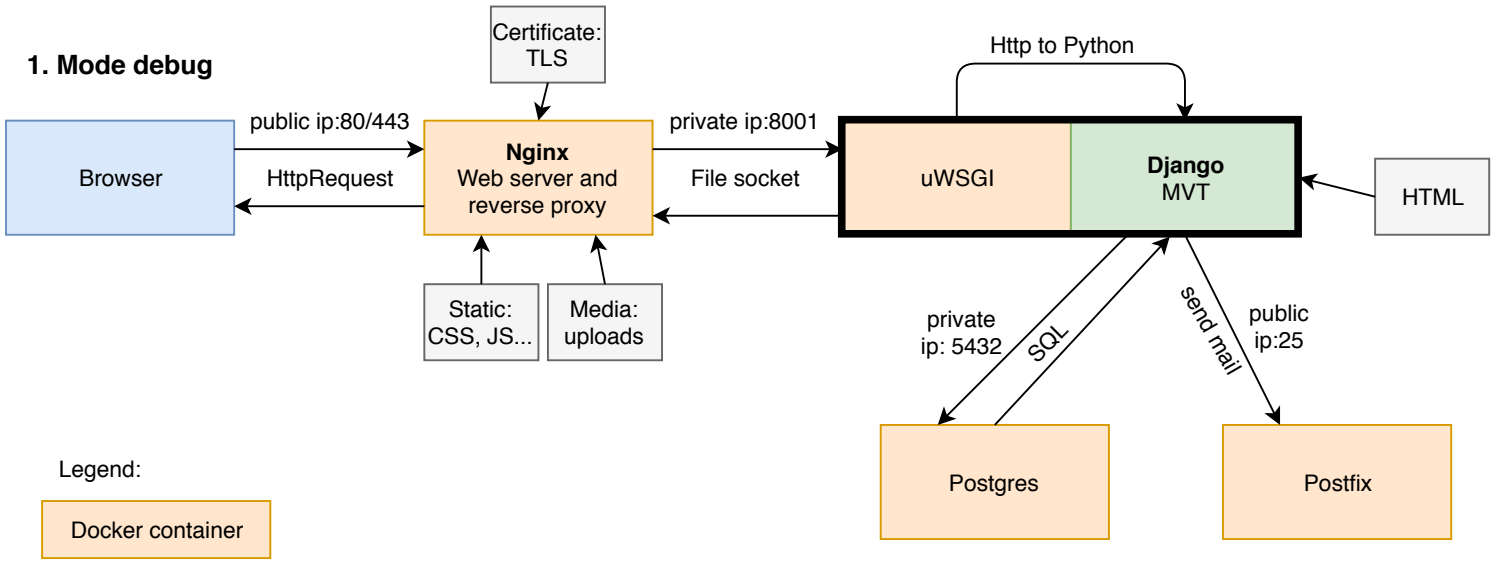
Ecosystème Docker :

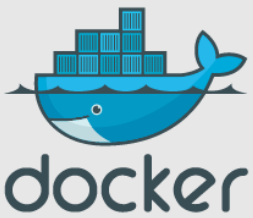
- Outils : Docker, Docker Compose.
- Images : DockerHub, Registre GitLab, Build.
- Réseau : user-defined bridges (résolution DNS automatique entre les conteneurs).
- Volumes : volume (espaces isolés sur l'hôte).

Mode debug

Présentation
- Historique
- CLI
- Configuration
- Format
- Bonnes pratiques
- Limitations

TP
- Commandes utiles
- Application Web
- **Mode debug**
- Mode production





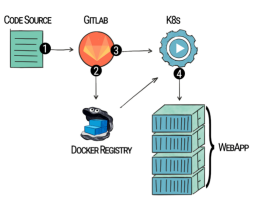
Mode debug

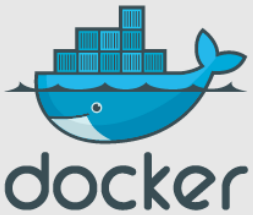
Présentation
- Historique
- CLI
- Configuration
- Format
- Bonnes pratiques
- Limitations

TP
- Commandes utiles
- Application Web
- Mode debug
- Mode production

Application Web :

- Image : Dockerfile et/ou Registre GitLab.
- Environment : Python 3.6.
- Bibliothèques : Django 1.11 (LTS), Bootstrap 3 (CMS).



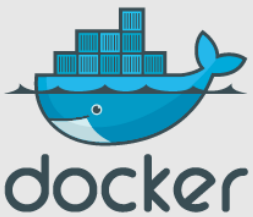


Mode debug

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Fichiers :

```
Docker_06_ComposeAdvanced/
|--README.md
|--.env
|--docker-compose.debug.yml
|--docker-compose.debugv3.yml
|--docker-compose.overridetemplate.yml
|__webapp
   |--Dockerfile.debug
   |--ecole.env
   |--ecole
   |--...
```



Etape #1

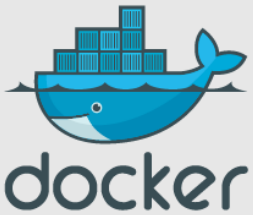
Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Instructions :

Récupérer le code de l'application webapp.

- Terminal :

```
$ git clone \
  https://gitlab.in2p3.fr/MaitresNageurs/\
  EnBarque.git
$ cd EnBarque/Docker_06_ComposeAdvanced/webapp
```



Etape #1 suite : optionnelle

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

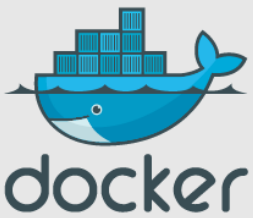
Instructions :

Installer les dépendances de l'application webapp dans un environnement local ou virtuel (venv).

- Terminal en local[1] :

```
$ pip install -r requirements.txt
$ export DJANGO_SECRET=1234
$ export DJANGO_HOSTIP=localhost
```

[1] MV cloud : $ export DJANGO_HOSTIP=public_ip



Etape #1 suite : optionnelle

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Instructions :

Lancer webapp en mode **debug** sans conteneur et vérifier son bon fonctionnement.

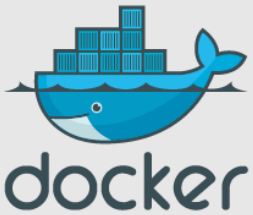
- Terminal en local[2] :

```
$ python manage.py makemigrations && \
  python manage.py migrate
$ python manage.py runserver
$ curl localhost:8000
```

- Navigateur : http://localhost:8000

[2] MV cloud :

$ python manage.py runserver public_ip:open_port



Etape #2 : Docker

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

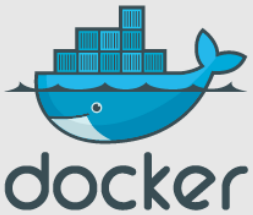
Instructions :

Construire l'image de l'application webapp en mode **debug**, exécuter l'application dans un conteneur Docker et ajouter un administrateur.

- Terminal en local[3] :

```
$ docker build -t webapp:debug \
  -f Dockerfile.debug .
$ docker run -d -p 8000:8000 \
  --env-file ecole.env  \
  --name webapp --rm webapp:debug
```

[3] MV cloud : ecole.env ➡ DJANGO_HOSTIP=public_ip



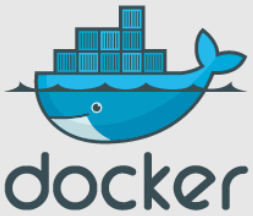
Etape #2 suite

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

- Terminal :

```
$ docker exec -it webapp \
  python manage.py createsuperuser \
  --username admin --email admin@gmail.com
$ docker stop webapp && docker rm webapp
```

- Navigateur : `http://localhost:8000`



Etape #3 : Docker Compose

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

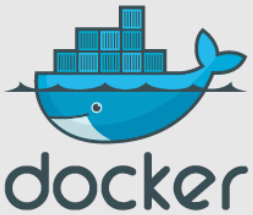
Instructions :

Lancer l'application webapp en mode **debug** avec Docker Compose en format v2.2. Vérifier le bon fonctionnement de l'application et ajouter un administrateur.

- Terminal en local[4] :

```
$ cd EnBarque/Docker_06_ComposeAdvanced
$ docker-compose -f docker-compose.debug.yml \
  config
$ ln -s docker-compose.debug.yml \
  docker-compose.yml
$ docker-compose up -d
```

[4] MV cloud : .env ➡ DJANGO_HOSTIP=public_ip



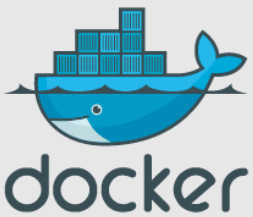
Etape #3 suite

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

- Terminal :

```
$ docker-compose ps
$ docker-compose logs
$ docker-compose exec webapp env
$ docker-compose exec webapp \
  python manage.py createsuperuser \
  --username admin --email admin@gmail.com
```

- Navigateur : `http://localhost:8000`



Etape #4 : variables d'environnement

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

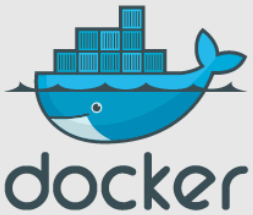
Instructions :

Changer le port de l'hôte dans le fichier .env :
8000 ➡ 80.

- Terminal :

```
$ docker-compose up -d
$ docker-compose ps
```

- Navigateur : `http://localhost:80`



Etape #5 : Dev/Prod

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

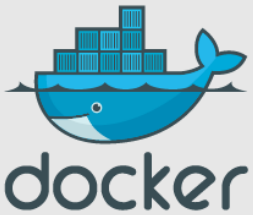
Instructions :

Utiliser le fichier Compose **override** afin d'avoir un cas spécifique de développement.

- Terminal :

```
$ mv docker-compose.overridetemplate.yml \
  docker-compose.override.yml
$ docker-compose config
$ docker-compose up -d
$ touch dev/test.txt && docker-compose exec  \
  webapp ls /webapp/media/uploads
```

- Navigateur : `http://localhost:8080`



Etape #6 : format v3

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Instructions :

Utiliser le fichier Compose v3 afin de tester les nouvelles fonctionnalités du format v3. Déployer en mode non Swarm afin d'utiliser la partie deploy avec Compose (fonctionne seulement avec les options **resources**, **replicas** et **restart_policy**).

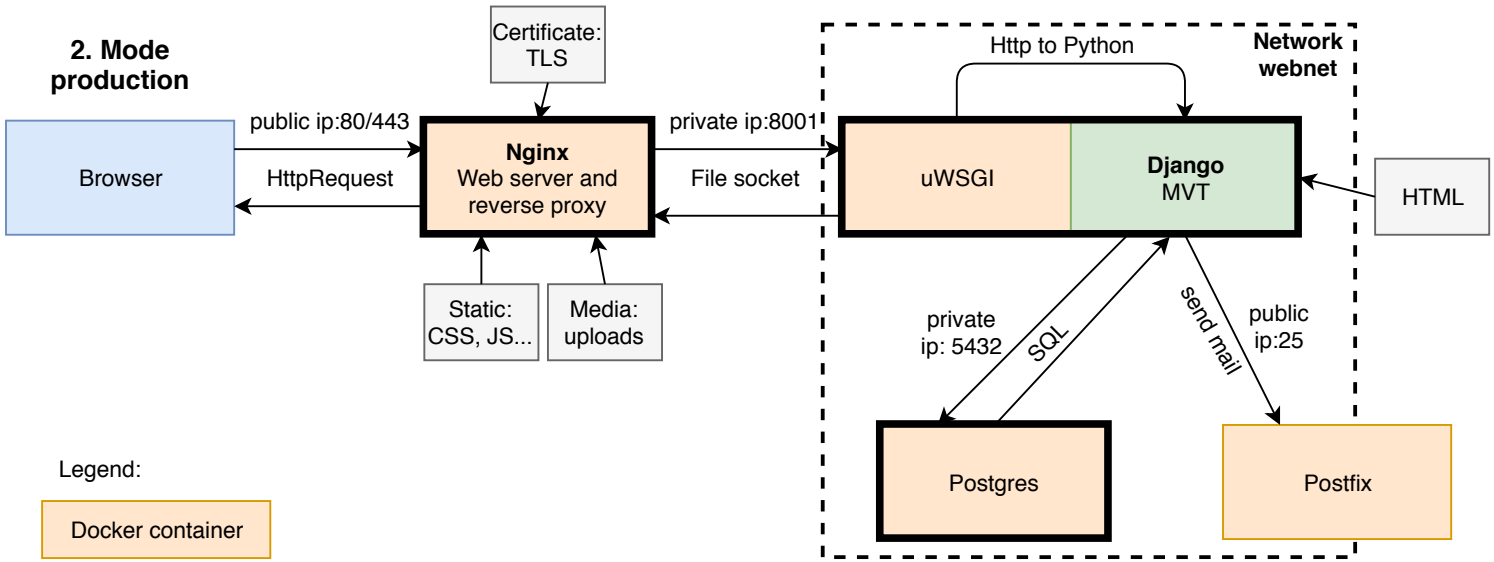
- Terminal :

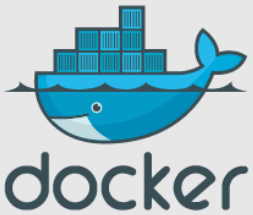
```
$ ln -s docker-compose.debugv3.yml \
   docker-compose.yml
$ docker-compose --compatibility up -d
$ docker stats
$ docker-compose down
```

Mode production

Présentation
- Historique
- CLI
- Configuration
- Format
- Bonnes pratiques
- Limitations

TP
- Commandes utiles
- Application Web
- Mode debug
- **Mode production**





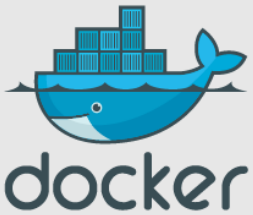
Mode production

Présentation
- Historique
- CLI
- Configuration
- Format
- Bonnes pratiques
- Limitations

TP
- Commandes utiles
- Application Web
- Mode debug
- Mode production

Base de données : PostgreSQL

- Image : DockerHub.
- Variables d'environnement : gérées par Compose.

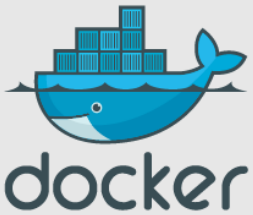


Mode production

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Serveur Web : Nginx

- Image : DockerHub.
- Configuration : mysite.conf.
- Protocoles : http (pour https, utiliser Let's Encrypt sur l'hôte).

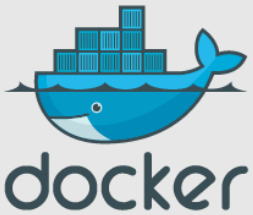


Mode production

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Passerelle de serveur Web : uWSGI

- Avec l'application webapp et connecté avec Nginx par socket web via un volume partagé.
- Configuration : uwsgi.ini

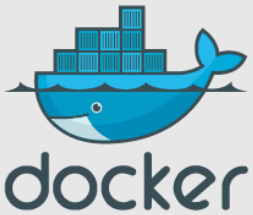


Mode production

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Fichiers :

```
Docker_06_ComposeAdvanced/
|--README.md
|--.env
|--docker-compose.production.yml
|--nginx
   |--mysite.conf
|__webapp
   |--Dockerfile.production
   |--wsgi
      |--uwsgi.ini
   |--...
```



Etape #1 : Docker Compose

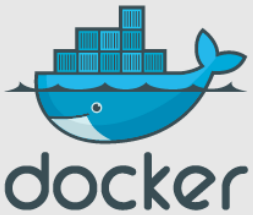
Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Instructions :

Changer les variables d'environnement pour passer en mode **production**.

- Terminal :

```
$ cat .env
DJANGO_PORT=80
DJANGO_HOSTIP=localhost
DJANGO_DB_USER=postgres
DJANGO_DB_PWD=postgres
DJANGO_DB_NAME=webapp
```



Etape #1 suite

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

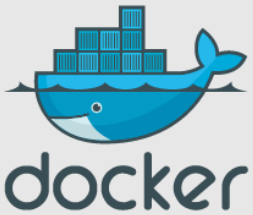
Instructions :

Lancer l'application webapp en mode **production** ainsi qu'une base de données et qu'un serveur Web utilisant le protocole HTTP avec Docker Compose en format v2.1.
Note : **depends_on** n'attend pas que les dépendances soient prêtes (seulement dans l'état démarrées).

- Terminal :

```
$ ln -s docker-compose.production.yml \
  docker-compose.yml
$ docker-compose up -d
```

- Navigateur : `http://127.0.0.1`



Etape #1 suite

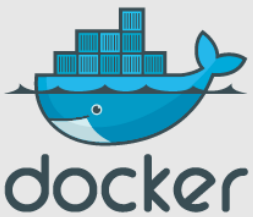
Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Instructions :

Vérifier le bon fonctionnement de l'application.

- Terminal :

```
$ docker-compose exec webapp \
  python manage.py createsuperuser \
  --username admin --email admin@gmail.fr
$ docker-compose ps
$ docker-compose logs
$ docker-compose exec webapp env
```



Etape #1 suite

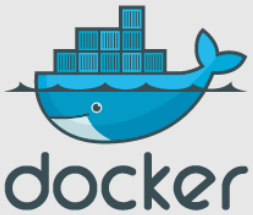
Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Instructions :

Faire un dump de la base de données.

- Terminal :

```
$ docker-compose exec db \
  pg_dumpall -c -U postgres > \
  dump_`date +%d-%m-%Y"_"%H_%M_%S`.sql
```



Etape #2 : passage à l'échelle

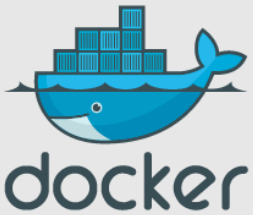
Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Instructions :

Utiliser la fonctionnalité de passage à l'échelle (**scale**) pour lancer plusieurs instances de webapp. Note : il faut commenter **container_name**.

- Terminal :

```
$ docker-compose up -d --scale webapp=3
$ docker-compose ps
```



Pour aller plus loin : Docker Swarm

Présentation
Historique
CLI
Configuration
Format
Bonnes pratiques
Limitations
TP
Commandes utiles
Application Web
Mode debug
Mode production

Instructions :

Utiliser Docker Stack et le fichier Compose pour lancer un déploiement sur un master Docker Swarm.

- Terminal :

```
$ docker swarm init
$ docker info
$ docker swarm node ls
$ docker stack deploy -c \
  docker-compose.debug.yml django
$ docker swarm leave --force
```