

TP NOTE VIRTU

mon github:

<https://github.com/julian2bot/tetris-virtu>

1 Préparatifs sous Docker compose

QUESTION 1 - Pour quel services est-il nécessaire de construire l'image plutôt que de la télécharger telle quelle depuis hub docker?

les services qui doivent être construits sont le backend et frontend

QUESTION 2 - Quelles commandes permettent de construire chacune des images qui ne sont pas téléchargées?

backend

```
docker build -t backend .
```

frontend

```
docker build -t frontend .
```

QUESTION 3 - Le démarrage des services peut être compromis si d'autres processus de la machine virtuelle sont déjà en écoute sur certains ports. Quels sont les ports concernés?

les ports concernés sont : 8081

et 8080 / 6379 mais perso pas eu de problème avec.

2 Passage sur swarm

QUESTION 4 - Utiliser les commandes docker save et docker load pour disposer de copies des images backend et frontend sur toutes vos machines virtuelles

```
#save le front
docker save terramino-go-frontend:latest -o frontend.tar
# save le back
docker save terramino-go-backend:latest -o backend.tar
```

```
scp backend.tar frontend.tar o22302184@172.16.0.244:/home/user/  
scp -J o22302184@acces-tp.iut45.univ-orleans.fr o22302184@o22302184-244:~/tuerNguyen/terramino-go  
.terramino-go-frontend.tar
```

puis copier sur les autres vm (mais je suis pas sur les vm donc tkt ca fonctionne)

puis exacuteer pour chaque VM:

```
docker load < backend.tar  
docker load < frontend.tar
```

QUESTION 5 - Déployer la stack décrite ci-dessus

Executer la commande:

```
docker stack deploy -c docker-stack.yml terramino
```

Vérif du deploiement:

```
docker stack services terramino  
docker stack ps terramino
```

QUESTION 6 - À partir du fichier docker-compose.yml de terramino, créez un fichier docker-stack.yml et déployez-le pour obtenir la stack décrite ci-dessus.

`docker-stack.yml`

```
version: "3.8"  
  
services:  
  frontend:  
    image: terramino-frontend:latest  
    ports:  
      - "8081:8081"  
    deploy:  
      replicas: 3  
      restart_policy:  
        condition: on-failure  
  
  backend:  
    image: terramino-backend:latest
```

```

environment:
  REDIS_HOST: redis
  REDIS_PORT: 6379
  TERRAMINO_PORT: 8080
ports:
  - "8080:8080"
deploy:
  replicas: 2
  restart_policy:
    condition: on-failure

redis:
  image: redis:alpine
  volumes:
    - redis_data:/data
deploy:
  replicas: 1
  restart_policy:
    condition: on-failure

volumes:
  redis_data:

```

QUESTION 7 - Quelles précaution sont nécessaires pour assurer que tous les clients observent les mêmes données (la valeur du record)?

Pour que tous les clients voient la même valeur du record, tous les backends doivent utiliser le même service Redis unique avec un stockage persistant.

QUESTION 8 - Assurer les contraintes de la question précédente via le fichier docker-stack.yml.

nouveau docker-stack.yml

```

version: "3.8"

services:
  frontend:
    image: terramino-frontend:latest
    ports:
      - "8081:8081"
    deploy:
      replicas: 3           # 3 répliques du frontend comme demandé
      restart_policy:
        condition: on-failure

  backend:
    image: terramino-backend:latest

```

```

environment:
  REDIS_HOST: redis          # Tous les backends pointent vers le même
Redis
  REDIS_PORT: 6379
  TERRAMINO_PORT: 8080
ports:
  - "8080:8080"
deploy:
  replicas: 2                # 2 répliques du backend
  restart_policy:
    condition: on-failure

redis:
  image: redis:alpine
  volumes:
    - redis_data:/data       # Volume persistant pour conserver le record
  deploy:
    replicas: 1              # UNE seule réplique pour garantir l'unicité
des données
    restart_policy:
      condition: on-failure

volumes:
  redis_data:               # Volume partagé pour Redis

```

QUESTION 9 -

La robustesse est vérifiée en arrêtant successivement chaque conteneur avec :

```
docker service scale terramino_backend=0
```