

Mind of tjiho

- [Lol](#)
- [Ciné 2021](#)
- [Super Frigo](#)
- [Commandes](#)
- [Git](#)
- [grub option pour vieux pc](#)
- [musique](#)
- [Snippets vscode](#)
- [Genre et tri](#)
- [psql](#)
- [kobo](#)
- [docker](#)

Lol

lol

```
wmic path softwareLicensingService get OA3xOriginalProductKey
```

```
“ KG2NM-3WRRC-VKX3W-H7Y72-MP2KC
```

Ciné 2021

- drunk
 - adieu les cons
 - balloon
 - mandibules
 - Si le vent tombe
 - Vers la bataille
 - Hospitalité
 - Il mio Corpo
 - annette
 - titane
 - drive my car
-

Simon:

- Harakiri (Seppuku)
 - nausicaa et la vallée du vent
-

Other:

- district 9
 - Jungle Cruise
 - Le Goût des autres
-

- house
- pentipoule

api ciné

<https://www.bdfci.info/>

cineserie.com

Super Frigo

Food recognition:

Datasets

- <https://www.kaggle.com/moltean/fruits>
- <https://www.tensorflow.org/datasets/catalog/food101>

Code

- <https://github.com/neuronest/food-recognition-azureml>


```
wpctl set-volume @DEFAULT_SINK@ .10- # decrease sound by 10%
```

Systemd

```
# Automatic start-up of systemd user instances  
loginctl enable-linger
```

GPU

```
sudo envycontrol -s hybrid  
switcherooctl launch cmd  
envycontrol -s integrated
```

Debian

```
sudo apt install linux-headers-$(uname -r)  
sudo apt install broadcom-sta-dkms  
sudo dpkg-reconfigure broadcom-sta-dkms
```

Git

Merge

- Merge: `git merge origin/architecture`
- Squash (on remplace tout par un seul commit): `git merge --squash origin/architecture`
- Rebase (un jolie merge): `git rebase origin/architecture`

grub option pour vieux pc

- pci=noms
- reboot=pci
- acpi=force
- apm=power_off

Dans /etc/default/grub

puis update-grub

<http://mylinuxexplore.blogspot.com/2011/11/solved-ubuntu-doesnt-shutdown-properly.html>

musique

- Dès Que Le Vent Soufflera (Renaud)
- San Francisco (forestier)
- j'ai demandé a la lune (indochine)
- le chène liège (cabre)
- Les Murs de poussière (cabrel)
- je veux (zaz)

Snippets vscode

fichier > préférences > configurer les extraits de code

vue-js

```
{
  // Place your snippets for vue here. Each snippet is defined under a snippet name and has a
  // prefix, body and
  // description. The prefix is what is used to trigger the snippet and the body will be
  // expanded and inserted. Possible variables are:
  // $1, $2 for tab stops, $0 for the final cursor position, and ${1:label}, ${2:another} for
  // placeholders. Placeholders with the
  // same ids are connected.
  // Example:
  // "Print to console": {
  //   "prefix": "log",
  //   "body": [
  //     "console.log('$1');",
  //     "$2"
  //   ],
  //   "description": "Log output to console"
  // }
  "component": {
    "prefix": "<template>",
    "body": [
      "<template></template>",
      "<script>",
      "import { Fragment } from \"vue-fragment\"",
      "import { bus } from \"../main.js\"",
      "export default {",
      "  props: {",
      "  },",
      "  data() {",
      "    return {}",
      "  },",
      "  mounted() {"
```

```
        "    bus.$on(\"navigation\", () => {",
        "    })",
    "" },",
    "" methods: {},",
    "" components: { Fragment },",
    ""}",
    ""</script>"
    []
    []}
    }
```

Genre et tri

- chansons corse
- chansons française
- chanson américaines
- folk
- country
- world
- jazz
- classique
- rock
- latin
- blue
- pop
- soundtrack
- country

Patricia Kaas -> chanson française, chant thomas fersen -> chanson française (style 2000) Pauline Croze -> chanson française style zaz/pop

psql

```
\l => list database
```

```
\c <db name>: connect to a certain database
```

```
\dt: list all tables in the current database using your search_path
```

```
\dt *.*: list all tables in the current database regardless your search_path
```

```
SELECT * FROM osm_poi_point LIMIT 1
```

```
SELECT name,name_en,subclass,mapping_key,osm_id,geometry FROM osm_poi_point WHERE name <> ''  
LIMIT 1;
```

kobo

- <https://github.com/Rain92/qt5-kobo-platform-plugin>
- <https://github.com/NiLuJe/FBIInk/tree/master>
- <https://github.com/olup/kobowriter>
- <https://github.com/koreader/koxtoolchain>
- <https://github.com/Rain92/UltimateMangaReader>
- <https://github.com/koreader/koxtoolchain>
- <https://www.mobileread.com/forums/showthread.php?t=336210>

docker

```
# Docker Cheatsheet

## Images

```bash
docker images # lister les images locales
docker pull nginx:alpine # télécharger une image
docker build -t monapp:v1 . # build depuis Dockerfile
docker build -t monapp:v1 -f Dev.dockerfile . # Dockerfile custom
docker rmi <image_id> # supprimer une image
docker image prune -a # virer les images non utilisées
```

## Containers

```
docker ps # containers running
docker ps -a # tous les containers
docker run -d --name web nginx # run en background
docker run -it ubuntu bash # run interactif
docker run -d -p 8080:80 nginx # port mapping host:container
docker run -d -v /host/path:/container/path nginx # bind mount
docker run -d -v myvolume:/data nginx # named volume
docker run --rm nginx # auto-delete après stop
docker run -e NODE_ENV=prod node # variable d'env
docker run --env-file .env node # fichier d'env

docker exec -it <container> bash # shell dans container running
docker logs -f <container> # logs en live
docker stop <container> # SIGTERM graceful
docker kill <container> # SIGKILL brutal
docker rm <container> # supprimer
docker rm -f <container> # stop + rm
```

## Volumes & Networks

```
docker volume ls
docker volume create mydata
docker volume inspect mydata
docker volume rm mydata

docker network ls
docker network create mynet
docker network connect mynet <container>
docker run -d --network mynet nginx # attach au run
```

## Docker Compose

```
docker compose up -d # démarrer la stack
docker compose down # stop + rm containers
docker compose down -v # + supprime les volumes
docker compose logs -f # logs de tous les services
docker compose ps # état des services
docker compose exec web bash # exec dans un service
docker compose build # rebuild les images
docker compose pull # pull les nouvelles images
docker compose up -d --build # rebuild + restart
```

## Debugging

```
docker inspect <container> # JSON complet du container
docker stats # CPU/RAM/IO en live
docker top <container> # processus dans le container
docker diff <container> # fichiers modifiés vs image
docker cp <container>:/path ./local # copier fichier depuis container
docker history <image> # layers de l'image
```

## Cleanup

```
docker system df # espace disque utilisé
docker system prune # containers stoppés + images dangling
```

```
docker system prune -a --volumes # TOUT nettoyer (attention)
```

# Dockerfile essentials

```
FROM node:20-alpine
WORKDIR /app
COPY package*.json ./
RUN npm ci --only=production
COPY . .
EXPOSE 3000
USER node
CMD ["node", "server.js"]
```

# Compose essentials

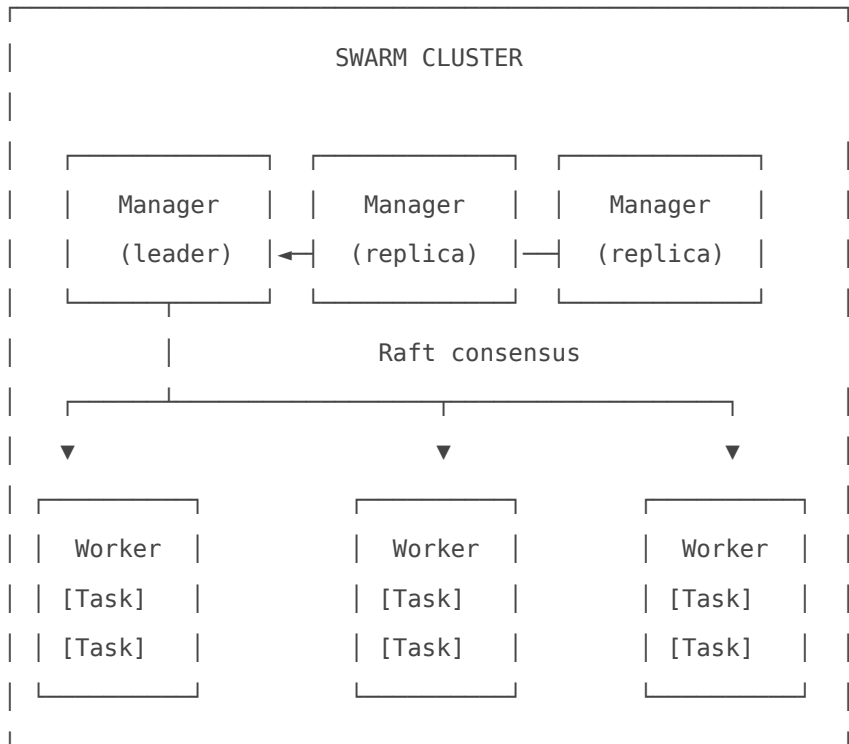
```
services:
 web:
 build: .
 ports:
 - "3000:3000"
 environment:
 - NODE_ENV=production
 volumes:
 - ./src:/app/src
 depends_on:
 - db
 restart: unless-stopped

 db:
 image: postgres:16-alpine
 volumes:
 - pgdata:/var/lib/postgresql/data
 environment:
 POSTGRES_PASSWORD: secret

volumes:
 pgdata:
```

# Docker Swarm Cheatsheet

## Concepts clés



Stack = groupe de services (1 docker-compose.yml = 1 stack)

Service = définition d'une app (image + config + replicas)

Task = instance d'un service (= 1 container qui tourne)

## Initialisation & Nodes

```
docker swarm init # init swarm (devient manager)
docker swarm init --advertise-addr 192.168.1.10 # si plusieurs IPs

docker swarm join-token worker # affiche token pour ajouter worker
docker swarm join-token manager # token pour ajouter manager
docker swarm join --token <token> <manager-ip>:2377 # rejoindre le swarm
```

```

docker node ls # lister les nodes
docker node inspect <node> # détails d'un node
docker node promote <node> # worker → manager
docker node demote <node> # manager → worker
docker node update --availability drain <node> # vider un node (maintenance)
docker node update --availability active <node> # remettre en service
docker node rm <node> # retirer un node

docker swarm leave # quitter le swarm (depuis worker)
docker swarm leave --force # quitter (depuis manager)

```

# Services

```

Créer un service
docker service create --name web nginx:alpine
docker service create --name web -p 8080:80 nginx # port mapping
docker service create --name web --replicas 3 nginx # 3 instances
docker service create --name api -e NODE_ENV=prod node # env var
docker service create --name db --mount type=volume,src=dbdata,dst=/var/lib/mysql mysql

Gérer les services
docker service ls # lister les services
docker service ps <service> # voir les tasks d'un service
docker service logs -f <service> # logs (toutes les tasks)
docker service inspect <service> # config complète JSON

Scaling
docker service scale web=5 # scale à 5 replicas
docker service scale web=5 api=3 # scale plusieurs services

Update
docker service update --image nginx:1.25 web # rolling update
docker service update --replicas 10 web # changer replicas
docker service update --env-add DEBUG=1 web # ajouter env var
docker service update --env-rm DEBUG web # supprimer env var
docker service update --rollback web # rollback dernière update

Supprimer

```

```
docker service rm web
```

```
supprimer le service
```

# Stacks (déploiement via Compose)

```
docker stack deploy -c docker-compose.yml myapp # déployer une stack
docker stack ls # lister les stacks
docker stack services myapp # services de la stack
docker stack ps myapp # tasks de la stack
docker stack rm myapp # supprimer la stack
```

# Compose pour Swarm (v3+)

```
version: "3.8"

services:
 web:
 image: nginx:alpine
 ports:
 - "80:80"
 deploy:
 mode: replicated # ou 'global' (1 par node)
 replicas: 3
 update_config:
 parallelism: 1 # 1 task à la fois
 delay: 10s # attendre entre chaque
 failure_action: rollback
 order: start-first # new avant de kill old
 rollback_config:
 parallelism: 1
 delay: 10s
 restart_policy:
 condition: on-failure
 delay: 5s
 max_attempts: 3
 resources:
 limits:
```

```
 cpus: '0.5'
 memory: 256M
 reservations:
 cpus: '0.25'
 memory: 128M
 placement:
 constraints:
 - node.role == worker
 - node.labels.region == eu-west
 networks:
 - frontend
```

#### api:

```
 image: myapi:latest
 deploy:
 replicas: 2
 secrets:
 - db_password
 configs:
 - source: api_config
 target: /app/config.json
 networks:
 - frontend
 - backend
```

#### db:

```
 image: postgres:16-alpine
 deploy:
 mode: replicated
 replicas: 1
 placement:
 constraints:
 - node.role == manager # ou un node spécifique
 volumes:
 - pgdata:/var/lib/postgresql/data
 networks:
 - backend
```

```
networks:
```

```
frontend:
 driver: overlay
backend:
 driver: overlay
 internal: true # pas d'accès externe

volumes:
 pgdata:

secrets:
 db_password:
 external: true # créé manuellement avant

configs:
 api_config:
 file: ./config.json
```

## Secrets & Configs

```
Secrets (données sensibles, chiffrées)
echo "monsecret" | docker secret create db_password -
docker secret create ssl_cert ./cert.pem # depuis fichier
docker secret ls
docker secret inspect db_password
docker secret rm db_password

Configs (fichiers de config non sensibles)
docker config create nginx_conf ./nginx.conf
docker config ls
docker config inspect nginx_conf
docker config rm nginx_conf

Utilisation dans service
docker service create --name db \
 --secret db_password \
 --config source=my_config,target=/etc/app/config.json \
 postgres
```

# Réseau

```
docker network create -d overlay mynetwork # réseau overlay pour swarm
docker network create -d overlay --internal private_net # pas de sortie externe
docker network ls
docker network inspect mynetwork

Le routing mesh expose automatiquement les ports sur TOUS les nodes
Même si le container tourne sur node2, tu peux y accéder via node1:port
```

# Debugging

```
docker service ps <service> --no-trunc # voir erreurs complètes
docker service logs <service> 2>&1 | grep error
docker inspect <task_id> # inspecter une task spécifique
docker node ps <node> # tasks sur un node
docker events # events en temps réel
```

# Commandes utiles au quotidien

```
Forcer re-déploiement (pull nouvelle image si même tag)
docker service update --force web

Update avec nouvelle image
docker service update --image myapp:v2 --update-parallelism 2 --update-delay 30s web

Voir où tournent les tasks
docker service ps web --format "{{.Node}}: {{.CurrentState}}"

Drainer un node avant maintenance
docker node update --availability drain node-03

... maintenance ...
docker node update --availability active node-03
```