

Contents

Cheatsheet Linux, SSH, Nginx	1
1. Linux : commandes de survie	1
Se déplacer et regarder	1
Fichiers	1
Droits et utilisateurs	2
Processus, réseau, paquets	2
2. SSH	2
Paire de clés	3
~/.ssh/config	3
Durcir un serveur (réflexes)	3
3. Nginx : reverse proxy	3
Configuration type (nginx/taskflow.conf)	4
Tester et recharger : dans le conteneur proxy	4
Certificat auto-signé avec openssl	5
Lire les logs pour diagnostiquer	5
Encadré : pièges fréquents	6

Cheatsheet Linux, SSH, Nginx

1. Linux : commandes de survie

Se déplacer et regarder

Commande	Effet
pwd	Où suis-je ?
ls -la	Liste détaillée, fichiers cachés inclus
cd /chemin · cd .. · cd ~ · cd -	Aller à... / parent / maison / dossier précédent
cat f · less f · head -n 20 f · tail -n 50 f	Afficher tout / paginé / début / fin
tail -f f	Suivre un fichier en direct (Ctrl+C pour quitter)
grep -rn "motif" dossier/	Chercher du texte (-i ignore la casse, -v inverse)
find . -name "*.yaml"	Chercher des fichiers
nano f	Éditeur simple (Ctrl+O enregistrer, Ctrl+X quitter)
man cmd · cmd --help	Aide

Fichiers

Commande	Effet
mkdir -p a/b/c	Crée des dossiers (et parents)
cp -r src dest · mv src dest	Copier / déplacer-renommer
rm f · rm -r dossier	Supprimer (pas de corbeille)

Commande	Effet
<code>touch f</code>	Crée un fichier vide
<code>ln -s cible lien</code>	Lien symbolique
<code>tar czf archive.tgz dossier/ · tar xzf archive.tgz</code>	Archiver / extraire
<code>df -h · du -sh dossier</code>	Espace disque libre / taille d'un dossier

Droits et utilisateurs

Commande	Effet
<code>ls -l : -rw-r--r-- 1 deploy deploy</code>	Type+droits (propriétaire, groupe, autres), propriétaire, groupe
<code>chmod 600 .env</code>	Lecture/écriture pour le propriétaire seul (secrets)
<code>chmod +x script.sh</code>	Rend exécutable
<code>chown user:groupe f</code>	Change le propriétaire
<code>sudo cmd</code>	Exécute en administrateur
<code>whoami · id</code>	Qui suis-je, mes groupes

Droits : r=4, w=2, x=1 ; 755 = rwxr-xr-x, 644 = rw-r--r--, 600 = rw-----.

Processus, réseau, paquets

Commande	Effet
<code>ps aux grep nginx · top (ou htop)</code>	Processus
<code>kill PID · kill -9 PID</code>	Arrêter / forcer
<code>ss -ltnp</code>	Ports en écoute et programme associé
<code>curl -I https://site · curl -fsS url</code>	En-têtes / requête, échec si erreur HTTP
<code>ping -c 3 hote</code>	Joint-on la machine ?
<code>sudo apt update && sudo apt install -y nginx</code>	Installer un paquet (Debian/Ubuntu)
<code>sudo systemctl</code>	Gérer un service (systemd)
<code>status\ start\ stop\ restart nginx</code>	
<code>journalctl -u nginx -f</code>	Logs d'un service systemd
<code>echo \$VAR · export VAR=valeur · env</code>	Variables d'environnement

Redirections : `cmd > f` (écrase) · `cmd >> f` (ajoute) · `cmd 2>&1` (erreurs aussi) · `cmd1 | cmd2` (tube). Enchaîner : `a && b` (b si a réussit). Tab complète, ↑ rappelle l'historique, Ctrl+R cherche dans l'historique, Ctrl+C interrompt.

2. SSH

Se connecter à une machine distante, en chiffré. Sur un **vrai serveur** (VPS), on s'authentifie **par clé**, jamais par mot de passe.

Dans nos TP, le « serveur » est votre poste : pas de SSH nécessaire. Ces gestes sont ceux que vous ferez en entreprise.

Paire de clés

Commande	Effet
ssh-keygen -t ed25519 -C "commentaire"	Crée ~/.ssh/id_ed25519 (privée , secrète) et .pub (publique , partageable)
ssh-copy-id user@serveur	Installe votre clé publique dans ~/.ssh/authorized_keys du serveur
ssh user@serveur · ssh -p 2222 -i ~/.ssh/cle user@serveur	Connexion (port / clé précisés)
ssh user@serveur 'docker ps'	Exécute une commande à distance
scp fichier user@serveur:/chemin/ ·	Copie vers le serveur
scp -r dossier user@serveur:/chemin/	
rsync -avz dossier/ user@serveur:/chemin/	Synchronise (ne transfère que les différences)
ssh-add ~/.ssh/cle	Charge la clé dans l'agent (passphrase saisie une fois)

Principe : le serveur détient votre clé **publique** (le cadenas), vous gardez la **privée** (la clé). Perdre la privée = perdre l'accès ; la divulguer = donner l'accès à quelqu'un d'autre.

~/.ssh/config

```
Host mon-serveur
  HostName 203.0.113.10
  User deploy
  Port 22
  IdentityFile ~/.ssh/id_ed25519
  IdentitiesOnly yes
```

Ensuite : ssh mon-serveur, scp f mon-serveur:/tmp/. Droits obligatoires : chmod 700 ~/.ssh ; chmod 600 ~/.ssh/id_ed25519 ~/.ssh/config ~/.ssh/authorized_keys.

Durcir un serveur (réflexes)

Dans /etc/ssh/sshd_config : PasswordAuthentication no · PermitRootLogin no · utilisateur non-root avec sudo · pare-feu (sudo ufw allow OpenSSH && sudo ufw allow 80,443/tcp && sudo ufw enable). Après modification : sudo sshd -t (test) puis recharger sshd. **Gardez une session ouverte** pendant que vous testez la nouvelle.

3. Nginx : reverse proxy

Un **reverse proxy** reçoit les requêtes des visiteurs (un seul point d'entrée, HTTPS) et les transmet aux applications internes (front, api).

navigateur --HTTPS 443--> [Nginx proxy] --HTTP--> front:80 --/api--> api:3000

Configuration type (nginx/taskflow.conf)

```
server {
    listen 80;
    server_name taskflow.localhost;
    return 301 https://$host$request_uri;          # HTTP -> HTTPS
}

server {
    listen 443 ssl;
    server_name taskflow.localhost;

    ssl_certificate      /etc/nginx/certs/taskflow.crt;
    ssl_certificate_key  /etc/nginx/certs/taskflow.key;

    location / {
        proxy_pass http://front:80;                # nom du service Compose
        proxy_set_header Host                      $host;
        proxy_set_header X-Real-IP                 $remote_addr;
        proxy_set_header X-Forwarded-For          $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto       $scheme;
    }
}
```

Directive	Rôle
server { }	Un site virtuel
listen / server_name	Port écouté / nom de domaine servi
location /chemin { }	Règles pour une partie des URL
proxy_pass	Où envoyer la requête
proxy_set_header	En-têtes transmis à l'application (vraie IP, protocole d'origine)
return 301 ...	Redirection
ssl_certificate / ssl_certificate_key	Certificat et clé privée

Tester et recharger : dans le conteneur proxy

Commande	Effet
docker compose exec proxy nginx -t	Teste la configuration (toujours avant de recharger)
docker compose exec proxy nginx -s reload	Recharge sans coupure
docker compose logs -f proxy	Logs d'accès et d'erreur (ils sortent sur stdout/stderr)
docker compose restart proxy	Redémarrage complet si besoin

Sur un serveur classique (sans conteneur) : `sudo nginx -t` puis `sudo systemctl reload nginx` (ou `sudo service nginx reload` sans systemd) ; fichiers dans

/etc/nginx/sites-available/ (activés par lien dans sites-enabled/) ; logs dans /var/log/nginx/access.log et error.log.

Certificat auto-signé avec openssl

```
mkdir -p certs
openssl req -x509 -nodes -newkey rsa:2048 -days 365 \
  -keyout certs/taskflow.key -out certs/taskflow.crt \
  -subj "/CN=taskflow.localhost" \
  -addext "subjectAltName=DNS:taskflow.localhost"
chmod 600 certs/taskflow.key
openssl x509 -in certs/taskflow.crt -noout -subject -dates      # vérifier
```

Option	Sens
req -x509	Produit directement un certificat (pas une demande à une autorité)
-nodes	Clé non chiffrée (sinon Nginx demanderait une passphrase au démarrage)
-newkey rsa:2048	Crée une nouvelle clé
-days 365	Durée de validité
-subj / subjectAltName	Nom du site ; les navigateurs exigent le SAN

Le navigateur affiche un avertissement : **normal**, personne d'officiel n'a signé ce certificat. Sur un vrai serveur avec un nom de domaine public : **Let's Encrypt** (certbot) délivre un certificat gratuit reconnu et le renouvelle seul.

Tester sans navigateur :

```
curl -k https://taskflow.localhost/health
curl -k --resolve taskflow.localhost:443:127.0.0.1 https://taskflow.localhost/
curl -I http://taskflow.localhost
```

(Ports modifiés dans .env ? Ajoutez-les : https://taskflow.localhost:8443, --resolve taskflow.localhost:8443:127.0.0.1.)

Lire les logs pour diagnostiquer

Code / message	Signification probable
502 Bad Gateway	Nginx fonctionne mais l'application derrière est arrêtée/injoignable : docker compose ps, puis logs du service
504 Gateway Timeout	L'application met trop de temps
404	Mauvais location, ou fichier absent
connect() failed (111: Connection refused) while connecting to upstream	Mauvais nom/port dans proxy_pass, ou service pas démarré
host not found in upstream "front"	Le service front n'existe pas sur ce réseau
	Compose

Code / message	Signification probable
cannot load certificate	Chemin du certificat faux ou volume non monté

Méthode : symptôme > logs proxy > docker compose ps > logs du service en cause > cause.

Encadré : pièges fréquents

- `rm -rf` n'a ni corbeille ni confirmation : relisez le chemin, jamais `rm -rf /` ou `rm -rf ~`.
- **Permissions SSH** : une clé privée lisible par d'autres (`chmod 644`) est **refusée** par ssh (« UNPROTECTED PRIVATE KEY FILE »). `chmod 600`.
- Confondre clé **publique** et **privée** : seule la `.pub` se partage ; ne la collez jamais dans un chat ou un repo.
- Se couper l'accès : désactiver le mot de passe SSH **avant** d'avoir testé la clé. Gardez une session ouverte.
- **Nginx : `nginx -t` avant reload**. Une faute de frappe et un redémarrage brutal coupe le site.
- Point-virgule ; oublié en fin de directive Nginx : erreur de syntaxe.
- `proxy_pass http://localhost:3000` dans un conteneur : `localhost` est le conteneur Nginx lui-même ; utilisez le nom du service.
- Fichier de config modifié sur l'hôte mais pas rechargé (`nginx -s reload`) ou pas monté dans le conteneur.
- Un fichier `.env` ou une clé en droits 644 : `chmod 600`.
- Port 80/443 occupé : voir `setup-poste.md` (variables `HTTP_PORT` / `HTTPS_PORT`).