tldr pages

Simplified and community-driven man pages

Generated on Mon Feb 1 23:33:04 2021

Common

7z

Un archiveur de fichiers avec un haut taux de compression.

Plus d'informations : https://www.7-zip.org/.

- Compresser un fichier ou un dossier :

```
7z a {{archive.7z}} {{chemin/vers/fichier_ou_dossier}}
```

- Chiffrer une archive existente (en incluant les en-têtes) :

```
7z a {{archive_chiffree.7z}} -p{{motdepasse}} -mhe=on
{{archive.7z}}
```

- Extraire un fichier 7z existant en conservant l'arborescence des fichiers :

```
7z x {{archive.7z}}
```

- Extraire une archive vers la destination donnée :

```
7z x {{archive.7z}} -o{{chemin/vers/destination}}
```

- Extraire une archive vers la sortie standard :

```
7z x {{archive.7z}} -so
```

- Archiver en utilisant un algorithme de compression particulier :

```
7z a -t{{zip|gzip|bzip2|tar}} {{archive.7z}} {{chemin/vers/
fichier_ou_dossier}}
```

- Lister les types d'archives disponibles :

```
7z i
```

- Lister le contenu d'une archive :

```
7z l {{archive.7z}}
```

apropos

Recherche dans les pages de manuel, par exemple pour trouver une nouvelle commande.

- Recherche par mot clé :

```
apropos {{expression_reguliere}}
```

- Recherche sans limiter la sortie à la largeur du terminal :

```
apropos -l {{expression_reguliere}}
```

- Recherche les pages qui contiennent toutes les expressions données (fonction ET)

```
apropos {{expression_reguliere_1}} -a
{{expression_reguliere_2}} -a {{expression_reguliere_3}}
```

atom

Un éditeur de texte multiplateforme proposant de nombreuses extensions.

Les extensions sont gérées par **`apm`**.

Plus d'informations : https://atom.io/.

- Ouvrir un fichier ou un dossier :

  ```
  atom {{chemin/vers/fichier_ou_dossier}}
  ```

- Ouvrir un fichier ou un dossier dans une nouvelle fenêtre :

  ```
  atom -n {{chemin/vers/fichier_ou_dossier}}
  ```

- Ouvrir un fichier ou un dossier dans une fenêtre existante :

  ```
  atom --add {{chemin/vers/fichier_ou_dossier}}
  ```

- Ouvrir atom en mode sans-échec (les extensions ne seront pas chargées) :

  ```
  atom --safe
  ```

- Empêcher atom de se lancer en arrière-plan, en le forçant à s'attacher au terminal :

  ```
  atom --foreground
  ```

awk

Langage de programmation polyvalent pour travailler sur des fichiers.

Plus d'informations : https://github.com/onetrueawk/awk.

- Affiche la cinquième colonne (ou le champ) dans un fichier qui utilise des espaces comme séparateur :

```
awk '{print $5}' {{nom_de_fichier}}
```

- Affiche la deuxième colonne dans des lignes contenant "quelque-chose" dans un fichier qui utilise des espaces comme séparateur :

```
awk '/{{quelque-chose}}/ {print $2}' {{nom_de_fichier}}
```

- Affiche la dernière colonne de chaque ligne d'un fichier en utilisant une virgule (au lieu des espaces) comme séparateur :

```
awk -F ',' '{print $NF}' {{nom_de_fichier}}
```

- Additionne les valeurs de la première colonne des lignes d'un fichier et affiche le total :

```
awk '{s+=$1} END {print s}' {{nom_de_fichier}}
```

- Additionne les valeurs de la première colonne des lignes d'un fichier et affiche ces valeurs puis affiche le total :

```
awk '{s+=$1; print $1} END {print "--------"; print s}'
{{nom_de_fichier}}
```

- Affiche une ligne sur trois en partant de la première ligne :

```
awk 'NR%3==1' {{nom_de_fichier}}
```

- Affiche les lignes dont la valeur de la colone 10 vaut la valeur recherchée :

```
awk '($10 == valeur)'
```

- Affiche les lignes dont la valeur de la colone 10 est comprise entre un min et un max :

```
awk '($10 >= valeur_min && $10 <= valeur_max)'
```

base32

Encode ou décode un fichier ou l'entrée standard vers ou depuis la base 32, et retourne le résultat à la sortie standard.

- Encode un fichier :

```
base32 {{fichier}}
```

- Décode un fichier :

```
base32 -d {{fichier}}
```

- Encode depuis stdin :

```
{{commande}} | base32
```

- Décode depuis stdin :

```
{{commande}} | base32 -d
```

base64

Encoder ou décoder un fichier ou l'entrée standard en utilisant le codage Base64 à destination de la sortie standard.

- Encoder un fichier :

```
base64 {{fichier}}
```

- Décoder un fichier :

```
base64 -d {{fichier}}
```

- Encoder depuis stdin :

```
{{une_commande}} | base64
```

- Décoder depuis stdin :

```
{{une_commande}} | base64 -d
```

basename

Retourne la portion ne contenant pas de dossiers d'un chemin complet.

- N'afficher que le nom du fichier depuis un chemin :

  ```
  basename {{path/to/file}}
  ```

- N'afficher que le nom du fichier depuis un chemin, en ôtant un préfixe donné :

  ```
  basename {{chemin/vers/fichier}} {{suffixe}}
  ```

bat

Affiche et concatène le contenu d'un ou plusieurs fichiers.

Un clon de **`cat`** avec mise en valeur de la syntaxe et integration avec Git.

Plus d'informations : https://github.com/sharkdp/bat.

- Affiche le contenu d'un fichier sur la sortie standard :

```
bat {{fichier}}
```

- Concatène le contenu de plusieurs fichiers vers le fichier de destination :

```
bat {{fichier1}} {{fichier2}} > {{fichier_de_destination}}
```

- Ajoute le contenu d'un ficher à la fin du fichier de destination :

```
bat {{fichier1}} {{fichier2}} >> {{fichier_de_destination}}
```

- Numérote toutes les lignes affichées :

```
bat -n {{fichier}}
```

- Affiche le contenu d'un fichier JSON sur la sortie standard avec mise en valeur de la syntaxe :

```
bat --language json {{fichier.json}}
```

- Affiche toutes les langages prises en charge :

```
bat --list-languages
```

borg

Outil de sauvegarde avec déduplication.

Crée des sauvegardes distantes ou locales qui peuvent être montées comme un système de fichiers.

Plus d'informations : https://borgbackup.readthedocs.io/en/stable/usage/general.html.

- Initialise un dépôt local :

```
borg init {{/chemin/vers/repertoire_du_depot}}
```

- Sauvegarde un répertoire dans le dépôt en créant une archive appelée "Lundi" :

```
borg create --progress {{/chemin/vers/repertoire_du_depot}}::
{{Lundi}} {{/chemin/vers/repertoire_source}}
```

- Liste toutes les archives d'un dépôt :

```
borg list {{/chemin/vers/repertoire_du_depot}}
```

- Extrait un répertoire donné de l'archive nommée "Lundi" à partir d'un dépôt distant tout en excluant tous les fichiers *.ext :

```
borg extract {{utilisateur}}@{{hote}}:{{/chemin/vers/
repertoire_du_depot}}::{{Lundi}} {{chemin/vers/
repertoire_destination}} --exclude '{{*.ext}}'
```

- Nettoie un dépôt en effaçant toutes les archives âgées de plus de 7 jours tout en affichant les changements :

```
borg prune --keep-within {{7d}} --list {{/chemin/vers/
repertoire_du_depot}}
```

- Monte un dépôt comme un système de fichiers FUSE :

```
borg mount {{/chemin/vers/repertoire_du_depot}}::{{Lundi}}
{{/chemin/vers/point_de_montage}}
```

- Affiche l'aide sur la création d'archives :

```
borg create --help
```

cat

Affiche et concatène le contenu d'un ou plusieurs fichiers.

- Affiche le contenu d'un fichier sur la sortie standard :

```
cat {{fichier}}
```

- Concatène le contenu de plusieurs fichiers vers le fichier de destination :

```
cat {{fichier1}} {{fichier2}} > {{fichier_de_destination}}
```

- Ajoute le contenu d'un ficher à la fin du fichier de destination :

```
cat {{fichier1}} {{fichier2}} >> {{fichier_de_destination}}
```

- Numérote toutes les lignes affichées :

```
cat -n {{fichier}}
```

cd

Modifier le répertoire de travail courant.

- Se déplacer vers le dossier donné :

  ```
  cd {{chemin/vers/dossier}}
  ```

- Se déplacer vers le répertoire personnel de l'utilisateur actuel :

  ```
  cd
  ```

- Remonter vers le parent du répertoire courant :

  ```
  cd ..
  ```

- Revenir au répertoire précédent :

  ```
  cd -
  ```

chmod

Modifie les droits d'accès d'un fichier ou d'un répertoire.

- Donne les droits d'e[x]écution à l'[u]tilisateur auquel le fichier appartient :

```
chmod u+x {{fichier}}
```

- Donne à l'utilisateur les droits de lecture (r) et d'écriture (w) sur un fichier/répertoire :

```
chmod u+rw {{fichier_ou_repertoire}}
```

- Enlève les droits d'exécution pour le [g]roupe :

```
chmod g-x {{fichier}}
```

- Donne à tous (a) les utilisateurs les droits de lecture et d'exécution :

```
chmod a+rx {{fichier}}
```

- Donne aux autres utilisateurs (qui sont dans un autre groupe) les mêmes droits que ceux du groupe propriétaire :

```
chmod o=g {{fichier}}
```

- Modifie les permissions recursivement en donnant aux membres du groupe et aux autres utilisateurs le droit d'écriture :

```
chmod -R g+w,o+w {{repertoire}}
```

chown

Modifie l'utilisateur et le groupe propriétaire des fichiers et dossiers.

- Modifie le propriétaire d'un fichier/dossier :

```
chown {{utilisateur}} {{chemin/vers/fichier_ou_dossier}}
```

- Modifie l'utilisateur et le groupe propriétaire d'un fichier/dossier :

```
chown {{utilisateur}}:{{groupe}} {{chemin/vers/
fichier_ou_dossier}}
```

- Modifie récursivement le propriétaire d'un dossier et de son contenu :

```
chown -R {{utilisateur}} {{chemin/vers/dossier}}
```

- Modifie le propriétaire d'un lien symbolique :

```
chown -h {{utilisateur}} {{chemin/vers/lien_symbolique}}
```

- Modifie the propriétaire d'un fichuer/dossier pour correspondre à un fichier de référence :

```
chown --reference={{chemin/vers/fichier_de_référence}}
{{chemin/vers/fichier_ou_dossier}}
```

clear

Efface l'écran du terminal.

- Effacer l'écran (identique à la séquence Contrôle-L sur une interface bash) :

  ```
  clear
  ```

convert

Outil de conversion d'image de Imagemagick.

Plus d'informations : https://imagemagick.org/script/convert.php.

- Convertir une image JPG en PNG :

```
convert {{image.jpg}} {{image.png}}
```

- Redimensionner une image à 50% de ses dimensions d'origine :

```
convert {{image.png}} -resize 50% {{image2.png}}
```

- Redimensionner une image en conservant son ratio hauteur/largeur initial pour une taille maximum de 640x480 :

```
convert {{image.png}} -resize 640x480 {{image2.png}}
```

- Coller plusieurs images horizontallement :

```
convert {{image1.png}} {{image2.png}} {{image3.png}} +append
{{image123.png}}
```

- Créer un gif à partir d'une série d'images avec un délai de 100ms entre chaque :

```
convert {{image1.png}} {{image2.png}} {{image3.png}} -delay
{{100}} {{animation.gif}}
```

- Créer une image avec un simple arrière-plan uni :

```
convert -size {{800x600}} "xc:{{#ff0000}}" {{image.png}}
```

cp

Copie des fichiers et des répertoires.

- Copier un fichier vers un autre emplacement :

```
cp {{chemin/vers/fichier_source.ext}} {{chemin/vers/
fichier_cible.ext}}
```

- Copier un fichier vers un autre répertoire en conservant le nom du fichier :

```
cp {{chemin/vers/fichier_source.ext}} {{chemin/vers/
répertoire_parent_cible}}
```

- Copier récursivement le contenu d'un répertoire vers un autre emplacement (si la destination existe, le répertoire est copié à l'intérieur) :

```
cp -R {{chemin/vers/répertoire_source}} {{chemin/vers/
répertoire_cible}}
```

- Copier un répertoire récursivement, en mode verbeux (affiche les fichiers au fur et à mesure de leur copie) :

```
cp -vR {{chemin/vers/répertoire_source}} {{chemin/vers/
répertoire_cible}}
```

- Copier les fichiers texte vers un autre emplacement, en mode interactif (demande confirmation avant d'écraser) :

```
cp -i {{*.txt}} {{chemin/vers/répertoire_cible}}
```

- Déréférencer les liens symboliques avant de copier :

```
cp -L {{link}} {{chemin/vers/répertoire_cible}}
```

- Copier en utilisant le chemin complet des fichiers source, en créant les répertoires intermédiaires manquants :

```
cp --parents {{source/chemin/vers/fichier}} {{chemin/vers/
fichier_cible}}
```

deluge

Client BitTorrent à base de ligne de commande.

Plus d'informations : https://deluge-torrent.org.

- Télécharge un torrent :

```
deluge {{url|magnet|chemin/vers/fichier}}
```

- Télécharge un torrent à l'aide d'un fichier de configuration particulier :

```
deluge -c {{chemin/vers/fichier_configuration}} {{url|magnet|
chemin/vers/fichier}}
```

- Télécharge un torrent et lance un interface usager particulier :

```
deluge -u {{gtk|web|console}} {{url|magnet|chemin/vers/
fichier}}
```

- Télécharge un torrent et enregistre les journaux dans un ficher :

```
deluge -l {{chemin/vers/fichier_journalisation}} {{url|
magnet|chemin/vers/fichier}}
```

deluged

Un processus démon pour le client BitTorrent Deluge.

Plus d'informations : https://deluge-torrent.org.

- Lance le démon Deluge :

```
deluged
```

- Lance le démon Deluge sur un port spécifique :

```
deluged -p {{port}}
```

- Lance le démon Deluge à l'aide d'un fichier de configuration spécifique :

```
deluged -c {{chemin/vers/fichier_configuration}}
```

- Lance le démon Deluge et enregistre les journaux dans un fichier :

```
deluged -l {{chemin/vers/fichier_journalisation}}
```

df

Montre un aperçu de l'utilisation de l'espace disque.

- Afficher tous les systèmes de fichiers et leur utilisation d'espace disque :

  ```
  df
  ```

- Afficher tous les systèmes de fichiers et leur utilisation d'espace disque dans un format plus facilement :

  ```
  df -h
  ```

- Afficher le système de fichiers et son utilisation d'espace disque rattaché au chemin donné :

  ```
  df {{chemin/vers/fichier_ou_dossier}}
  ```

- Afficher des statistiques sur le nombre d'inodes disponibles :

  ```
  df -i
  ```

docker-build

Construis une image à partir d'un Dockerfile.

Plus d'informations : https://docs.docker.com/engine/reference/commandline/build/.

- Construis une image Docker en utilisant le Dockerfile du répertoire courant :

```
docker build .
```

- Construis une image Docker à partir d'un Dockerfile situé à une URL précisée :

```
docker build {{github.com/creack/docker-firefox}}
```

- Construis une image Docker et l'étiquette :

```
docker build --tag {{nom:etiquette}} .
```

- N'utilise pas le cache lors de la construction de l'image :

```
docker build --no-cache --tag {{nom:etiquette}} .
```

- Construis une image Docker utilisant un Dockerfile spécifique :

```
docker build --file {{Dockerfile}} .
```

- Construis avec des variables personnalisées définies à la volée :

```
docker build --build-arg {{HTTP_PROXY=http://
10.20.30.2:1234}} --build-arg {{FTP_PROXY=http://
40.50.60.5:4567}} .
```

docker-compose

Exécute et gère des applications au travers de plusieurs conteneurs Docker.

Plus d'informations : https://docs.docker.com/compose/reference/overview/.

- Liste tous les conteneurs en cours d'exécution :

  ```
  docker-compose ps
  ```

- Crée et démarre en arrière-plan tous les conteneurs décrits dans le fichier `docker-compose.yml` du répertoire courant :

  ```
  docker-compose up -d
  ```

- Démarre tous les conteneurs après les avoir recréés si nécessaire :

  ```
  docker-compose up --build
  ```

- Démarre tous les conteneurs spécifiés dans un fichier compose alternatif :

  ```
  docker-compose --file {{chemin/vers/fichier}} up
  ```

- Arrête tous les conteneurs en cours d'exécution :

  ```
  docker-compose stop
  ```

- Arrête et supprime tous les conteneurs, réseaux, images et volumes :

  ```
  docker-compose down --rmi all --volumes
  ```

- Affiche et suit la journalisation de tous les conteneurs :

  ```
  docker-compose logs --follow
  ```

docker container

Gère les conteneurs Docker.

Plus d'informations : https://docs.docker.com/engine/reference/commandline/container/.

- Liste les conteneurs Dockers en cours d'exécution :

```
docker container ls
```

- Démarre un ou plusieurs conteneur arrêtés :

```
docker container start {{nom_conteneur_1}}
{{nom_conteneur_2}}
```

- Tue un ou plusieurs conteneurs en cours d'exécution :

```
docker container kill {{nom_conteneur}}
```

- Arrête un ou plusieurs conteneurs en cours d'exécution :

```
docker container stop {{nom_conteneur}}
```

- Mets en pause tous les processus d'un ou plusieurs conteneurs :

```
docker container pause {{nom_conteneur}}
```

- Affiche des informations détaillées sur un ou plusieurs conteneurs :

```
docker container inspect {{nom_conteneur}}
```

- Exporte le système de fichiers d'un conteneur sous forme d'archive Tar :

```
docker container export {{nom_conteneur}}
```

- Crée une nouvelle image à partir des changements d'un conteneur :

```
docker container commit {{nom_conteneur}}
```

docker

Gestion des conteneurs et des images Docker.

Plus d'informations : https://docs.docker.com/engine/reference/commandline/cli/.

- Liste les conteneurs Docker en cours d'exécution :

  ```
  docker ps
  ```

- Liste tous les conteneurs Docker (en cours d'exécution ou arrêtés) :

  ```
  docker ps -a
  ```

- Démarre un conteneur à partir d'une image, avec un nom personnalisé :

  ```
  docker run --name {{nom_conteneur}} {{image}}
  ```

- Démarre ou arrête un conteneur existant :

  ```
  docker {{start|stop}} {{nom_conteneur}}
  ```

- Télécharge une image depuis un registre Docker :

  ```
  docker pull {{image}}
  ```

- Ouvre un shell dans un conteneur déjà en cours d'exécution :

  ```
  docker exec -it {{nom_conteneur}} {{sh}}
  ```

- Supprime un conteneur arrêté :

  ```
  docker rm {{nom_conteneur}}
  ```

- Récupère et suit les journaux de message d'un conteneur :

  ```
  docker logs -f {{nom_conteneur}}
  ```

echo

Affiche les paramètres donnés dans la console.

- Affiche un message (les guillemets sont facultatifs) :

  ```
  echo "{{Hello World}}"
  ```

- Affiche un message avec des variables d'environment :

  ```
  echo "{{Ma variable PATH est $PATH}}"
  ```

- Affiche un message sans retour à la ligne :

  ```
  echo -n "{{Hello World}}"
  ```

- Ajoute un message à un fichier :

  ```
  echo "{{Hello World}}" >> {{fichier.txt}}
  ```

- Active l'interprétation des spécificateurs d'échappement :

  ```
  echo -e "{{Colonne 1\tColonne 2}}"
  ```

ftp

Outils permettant d'interragir avec un serveur avec le protocole FTP.

- Se connecter à un serveur FTP :

  ```
  ftp {{ftp.exemple.com}}
  ```

- Passer au mode de transfert binaire (medias, fichiers compressés, etc) :

  ```
  binary
  ```

- Transférer plusieurs fichiers sans demander de confirmation pour chaque :

  ```
  prompt off
  ```

- Télécharger plusieurs fichiers :

  ```
  mget {{*.png}}
  ```

- Uploader plusieurs fichiers :

  ```
  mput {{*.zip}}
  ```

- Supprimer plusieurs fichiers sur le serveur distant :

  ```
  mdelete {{*.txt}}
  ```

- Renommer un fichier sur le serveur distant :

  ```
  rename {{ancien_fichier}} {{nouveau_fichier}}
  ```

ghdl

Simulateur à source ouverte pour le langage VHDL.

Plus d'informations : http://ghdl.free.fr.

- Analyse un fichier de source VHDL et génère un fichier objet :

  ```
  ghdl -a {{fichier.vhdl}}
  ```

- Élabore un design (où `{{design}}` est le nom d'une unité de configuration, d'entité, ou d'architecture) :

  ```
  ghdl -e {{design}}
  ```

- Exécute un design élaboré :

  ```
  ghdl -r {{design}}
  ```

- Exécute un design élaboré et sauvegarde la sortie à un fichier de forme d'onde :

  ```
  ghdl -r {{design}} --wave={{sortie.ghw}}
  ```

- Vérifie le syntaxe d'un fichier de source VHDL :

  ```
  ghdl -s {{fichier.vhdl}}
  ```

- Affiche l'aide générale :

  ```
  ghdl --help
  ```

git add

Ajoute les fichiers modifiés à l'index.

Plus d'informations : https://git-scm.com/docs/git-add.

- Ajouter un fichier à l'index :

```
git add {{chemin/vers/fichier}}
```

- Ajouter tous les fichiers (suivis et non-suivis) :

```
git add -A
```

- Ajoute les modifications des fichiers déjà suivis :

```
git add -u
```

- Ajoute aussi les fichiers ignorés :

```
git add -f
```

- Ajoute des parties de fichiers interactivement :

```
git add -p
```

- Ajoute interactivement les parties d un fichier specifié :

```
git add -p {{chemin/vers/fichier}}
```

- Ajouter un fichier interactivement :

```
git add -i
```

git am

Appliaquer des fichiers de path. Utile lorsque l on recoit des comits par email.

Voir aussi **`git format-patch`**, pour generer des ficheirs de patch.

Plus d'informations : https://git-scm.com/docs/git-am.

- Appliquer un fichier de patch :

```
git am {{chemin/vers/fichier.patch}}
```

- Annuler l'application d un fichier de patch :

```
git am --abort
```

- Appliquer autant de fichier de correctif que possible, en enregistrant les morceaux échoués pour rejeter le fichier :

```
git am --reject {{chemin/vers/fichier.patch}}
```

git annex

Gérez les fichiers avec Git, sans archiver leur contenu.

Lorsqu'un fichier est annexé, son contenu est déplacé dans un stockage clé-valeur et un lien symbolique est créé qui pointe vers le contenu.

Plus d'informations : https://git-annex.branchable.com.

- Aide :

  ```
  git annex help
  ```

- Initialize le repo avec Git annex :

  ```
  git annex init
  ```

- Ajoute un fichier :

  ```
  git annex add {{chemin/vers/fichier_ou_repertoire}}
  ```

- Affiche le statut courrand d un fichier ou repertoire :

  ```
  git annex status {{chemin/vers/fichier_ou_repertoire}}
  ```

- Synchronise un repository local avec un distant :

  ```
  git annex {{distant}}
  ```

- Recupére un ficheir ou un repertoire :

  ```
  git annex get {{chemin/vers/fichier_ou_repertoire}}
  ```

git apply

Applique un correctif a un fichier et/ou a l index.

Plus d'informations : https://git-scm.com/docs/git-apply.

- Afficher les messages a propos des fichiers corrigés :

```
git apply --verbose {{chemin/vers/fichier}}
```

- Applique le correctif et ajoute les fichiers a l index :

```
git apply --index {{chemin/vers/fichier}}
```

- Applique un correctif depuis une source distante :

```
curl {{https://example.com/file.patch}} | git apply
```

- Affiche les differencs resultantes et applique le correctif :

```
git apply --stat --apply {{chemin/vers/fichier}}
```

- Applique le correctif en ordre inverse :

```
git apply --reverse {{chemin/vers/fichier}}
```

- Stocke le resultat du correctif dans l index sans modifier la branche courrante :

```
git apply --cache {{chemin/vers/fichier}}
```

git archive

Cree une archive de fichiers depuis un branche donée.

Plus d'informations : https://git-scm.com/docs/git-archive.

- Crée une archive `.tar` avec le contenu de la HEAD et l'affiche sur la sortie stanard :

  ```
  git archive --verbose HEAD
  ```

- Crée une archive `.zip` avec le contenu de la HEAD et l'affiche sur la sortie stanard :

  ```
  git archive --verbose --format=zip HEAD
  ```

- Pareil que ci-dessus mais ecrit dans l archive specifiée :

  ```
  git archive --verbose --output={{chemin/vers/fichier.zip}}
  HEAD
  ```

- Crée une archive depuis le dernier commit de la branche spécifiée :

  ```
  git archive --output={{chemin/vers/fichier.tar}}
  {{nom_de_branche}}
  ```

- Crée une archive avec le contenu d un repertoire donné :

  ```
  git archive --output={{chemin/vers/fichier.tar}} HEAD:
  {{chemin/vers/repertoire}}
  ```

- Ajoutez un chemin d'accès à chaque fichier pour l'archiver dans un répertoire spécifique :

  ```
  git archive --output={{chemin/vers/fichier.tar}} --
  prefix={{chemin/vers/cible}}/ HEAD
  ```

git bisect

Utiliser une recherche binaire pour trouver le commit qui a introduit un bug.

Git saute automatiquement d'avant en arrière dans le graphe de commit pour isoler le commit défectueux.

Plus d'informations : https://git-scm.com/docs/git-bisect.

- Démarrez une dissection sur une plage de commit délimitée par un bug connu et un commit propre connu (généralement plus ancien) :

  ```
  git bisect start {{bad_commit}} {{good_commit}}
  ```

- Pour chaque `git bisect` selectionné, le marquer comme "bad" ou "good" apres l'avoir testé pour le probléme :

  ```
  git bisect {{good|bad}}
  ```

- Apres que `git bisect` pointe vers le mauvais commit, terminer la dissection et retourner a la branche précedante :

  ```
  git bisect reset
  ```

- Sauter un commit lorrs de la dissection (e.g. celui qui echoue les tests pour une autre raison) :

  ```
  git bisect skip
  ```

git blame

Affiche le hash de commit et le dernier auteur de chaque ligne d un fichier.

Plus d'informations : https://git-scm.com/docs/git-blame.

- Affiche le hash de commit et le nom de l auteur en face de chaque ligne :

```
git blame {{file}}
```

- Affiche le hash de commit le nom et l email de l auteur en face de chaque ligne :

```
git blame -e {{file}}
```

git branch

Commande Git principale pour travailler avec des branches.

Plus d'informations : https://git-scm.com/docs/git-branch.

- Liste les branches locale, prefixe la branche courrante avec `*` :

```
git branch
```

- Liste toutes les branches (locale et distantes) :

```
git branch -a
```

- Affiche le nom de la branche courrante :

```
git branch --show-current
```

- Crée une nouvelle branche depuis le commit courrant :

```
git branch {{nom_de_branche}}
```

- Crée une nouvelle branche depuis un commit en particulier :

```
git branch {{nom_de_branche}} {{commit_hash}}
```

- Renommer une branche (ne pas se trouver sur la branche pour le faire) :

```
git branch -m {{ancien_nom_de_branche}}
{{nouveau_nom_de_branche}}
```

- Supprimer un branche locale (ne pas se trouver sur la branche pour le faire) :

```
git branch -d {{nom_de_branche}}
```

- Supprimer une branche distante :

```
git push {{nom_distant}} --delete {{nom_de_branche_distante}}
```

git bundle

Empaquetez des objets et des références dans une archive.

Plus d'informations : https://git-scm.com/docs/git-bundle.

- Empaquetez tout les objets et les refferences d'une branche spécifiée :

```
git bundle create {{chemin/vers/fichier.bundle}}
{{nom_de_branche}}
```

- Crée un empaquetage de tout les fichiers de toutes les branches :

```
git bundle create {{chemin/vers/fichier.bundle}} --all
```

- Crée un empaquetage des 5 derniers commits de la branche courrante :

```
git bundle create {{chemin/vers/fichier.bundle}} -{{5}}
{{HEAD}}
```

- Crée un empaquetage des 7 derniers jours :

```
git bundle create {{chemin/vers/fichier.bundle}} --
since={{7.days}} {{HEAD}}
```

- Verifie qu'un empaquetage est valide et peut étre appliquer à la branche courrante :

```
git bundle verify {{chemin/vers/fichier.bundle}}
```

- Affiche sur la sortie standard la liste des réfferences contenues dans un empaquetage :

```
git bundle unbundle {{chemin/vers/fichier.bundle}}
```

- Extraire une branche spécifique d'un fichier de bundle dans le référentiel actuel :

```
git pull {{chemin/vers/fichier.bundle}} {{nom_de_branche}}
```

git cat-file

Fournir des informations sur le contenu ou le type et la taille des objets du référentiel Git.

Plus d'informations : https://git-scm.com/docs/git-cat-file.

- Obtenir la taill [s] du commit HEAD en octets :

```
git cat-file -s HEAD
```

- Obtenir le type [t] (blob, tree, commit, tag) d'un objet Git spécifié :

```
git cat-file -t {{8c442dc3}}
```

- Affiche le contenu [p] d'un objet Git basé sur son type :

```
git cat-file -p {{HEAD~2}}
```

git check-ignore

Analyser et déboguer les fichiers ignorés / exclus (".gitignore") de Git.

Plus d'informations : https://git-scm.com/docs/git-check-ignore.

- Verifie qu'un fichier ou repertoire est ignoré :

```
git check-ignore {{chemin/vers/fichier_ou_repertoire}}
```

- Verifie que plusieurs fichiers ou repertoires sont ignorés :

```
git check-ignore {{chemin/vers/fichier}} {{chemin/vers/
repertoire}}
```

- Utilisez des chemins d'accès, un par ligne, de stdin :

```
git check-ignore --stdin < {{chemin/vers/fichier_annexe}}
```

- Ne pas vérifier l'index (utilisé pour déboguer pourquoi les chemins ont été suivis et non ignorés) :

```
git check-ignore --no-index {{chemin/vers/
fichiers_ou_repertoires}}
```

- Inclure les details pour chaque occurence dans le chemin :

```
git check-ignore --verbose {{chemin/vers/
fichiers_ou_repertoires}}
```

git checkout

Extraire une branche ou des chemins vers l'arborescence de travail.

Plus d'informations : https://git-scm.com/docs/git-checkout.

- Créer une branche et basculer dessus :

```
git checkout -b {{nom_de_branche}}
```

- Créer une branche depuis une reférence spçifiaue et basculer dessus (par exemple, branche locales/distantes, tag, commit) :

```
git checkout -b {{nom_de_branche}} {{reference}}
```

- Basculer sur une branche locale existante :

```
git checkout {{nom_de_branche}}
```

- Basculer sur la branche précedante :

```
git checkout -
```

- Basculer sur une brnche distante existante :

```
git checkout --track {{nom_distant}}/{{nom_de_branche}}
```

- Annule tout les changements dans le repertoire courrant (voir `git reset` pour plus de commandes d'annulation) :

```
git checkout .
```

- Annule tout les changements dans le fichier spécifié :

```
git checkout {{filename}}
```

- Remplace un fichier par sa version d'une autre branche :

```
git checkout {{nom_de_branche}} -- {{filename}}
```

git cherry-pick

Appliquer les modifications introduites par les commits existants à la branche actuelle.

Pour appliquer les changements a une autre branche, utiliser d'abord **`git checkout`** pour basculer sur la branche désirée.

Plus d'informations : https://git-scm.com/docs/git-cherry-pick.

- Applique un commit à la branche courrante :

  ```
  git cherry-pick {{commit}}
  ```

- Appliquer une plage de commits à la branche courrante (voir aussi `git rebase --onto`) :

  ```
  git cherry-pick {{start_commit}}~..{{end_commit}}
  ```

- Appliquer plusieurs commits non sequentiels à la branche courrante :

  ```
  git cherry-pick {{commit_1}} {{commit_2}}
  ```

- Appliquer les changements d'un commit a la branche courrante sans créer de commit :

  ```
  git cherry-pick -n {{commit}}
  ```

git cherry

Rechercher des commits qui n'ont pas encore été appliqués en amont.

Plus d'informations : https://git-scm.com/docs/git-cherry.

- Afficher les commits (et leurs messages) avec des commits équivalents en amont :

  ```
  git cherry -v
  ```

- Spécifiez une branche amont et une branche de rubrique différentes :

  ```
  git cherry {{origin}} {{topic}}
  ```

- Limiter les commits a ceux dans la limite donnée :

  ```
  git cherry {{origin}} {{topic}} {{base}}
  ```

git clean

Supprimer les fichiers non suivis du repertoire.

Plus d'informations : https://git-scm.com/docs/git-clean.

- Supprimer les fichiers non suivis par Git :

  ```
  git clean
  ```

- Supprimer les fichiers non suivis par Git de manière interactive :

  ```
  git clean -i
  ```

- Affiche les fichiers non suivis qui peuvent étre suprimmés :

  ```
  git clean --dry-run
  ```

- Nettoyage forcé des fichiers non suivis par Git :

  ```
  git clean -f
  ```

- Nettoyage forcé des repertoires non suivis par Git :

  ```
  git clean -fd
  ```

- Supprime tout les fichiers suivis, incluant ceux repertoriés par `.gitignore` et `.git/info/exclude` :

  ```
  git clean -x
  ```

git clone

Clone un dépôt existant.

Plus d'informations : https://git-scm.com/docs/git-clone.

- Clone un dépôt existant :

```
git clone {{location_du_depot_distant}}
```

- Clone un dépôt existant et ses sous-modules :

```
git clone --recursive {{location_du_depot_distant}}
```

- Clone un dépôt local :

```
git clone -l
```

- Clone silencieusement :

```
git clone -q
```

- Clone un dépôt existant en ne récupérant que les 10 commits les plus récents sur la branche par défaut (plus rapide) :

```
git clone --depth {{10}} {{location_du_depot_distant}}
```

git commit

Commit les fichers dans le repository.

Plus d'informations : https://git-scm.com/docs/git-commit.

- Commit les fichiers en stage dans le dépôt avec un message :

  ```
  git commit -m {{message}}
  ```

- Commit tous les fichiers modifiés avec un message :

  ```
  git commit -am {{message}}
  ```

- Mets à jour le dernier commit avec les modifications en stage :

  ```
  git commit --amend
  ```

- Commit seulement les fichiers spécifiés (qui sont déjà en stage) :

  ```
  git commit {{chemin/vers/mon/fichier1}} {{chemin/vers/mon/
  fichier2}}
  ```

git config

Gérer les options de configuration personnalisées pour les référentiels Git.

Ces configurations peuvent étre locales (pour le réferentiel courrant) ou globales (pour l utilisateur).

Plus d'informations : https://git-scm.com/docs/git-config.

- Liste les entrés de configurations locales (stockés dans `.git/config` du repertoire courrant) :

  ```
  git config --list --local
  ```

- Liste les entrés de conffigurations globales (stoqués dans `~/.gitconfig`) :

  ```
  git config --list --global
  ```

- Liste toutes les entrés de conffiguration, globales et locales :

  ```
  git config --list
  ```

- Recupére la valeure d'une entrée de configurations :

  ```
  git config alias.unstage
  ```

- Attribue la valeur d'une entree de configuration :

  ```
  git config --global alias.unstage "reset HEAD --"
  ```

- Restore la valeur d'une entree de configuration globale a sa valeur par défaut :

  ```
  git config --global --unset alias.unstage
  ```

- Édite le fichier de configuration du référentiel courrant dans l'éditeur par défaut :

  ```
  git config --edit
  ```

- Édite le fichier de configuration globale dans l'éditeur par défaut :

  ```
  git config --global --edit
  ```

git describe

Créer un nom unique et lisible pour un objet à partir d’une référence disponible.

Plus d'informations : https://git-scm.com/docs/git-describe.

- Créer un nom unique pour le commit courant (le nom contient le tag le plus recent, le nombre de commits additionnel, and le hash abrégé du commit) :

  ```
  git describe
  ```

- Créer un nom avec un hash de commit de 4 carctéres :

  ```
  git describe --abbrev={{4}}
  ```

- Générer un nom avec le chemin complet du tag :

  ```
  git describe --all
  ```

- Décrire un tag Git :

  ```
  git describe {{v1.0.0}}
  ```

- Créer un nom pour le dernier commit d'une branche donnée :

  ```
  git describe {{nom_de_branche}}
  ```

git diff

Afficher les changements sur les fichiers suivis.

Plus d'informations : https://git-scm.com/docs/git-diff.

- Afficher les changements sur les fichiers suvis :

```
git diff
```

- Afficher tout les changements sur les fichiers par rapport a la téte de branche :

```
git diff HEAD
```

- Afficher tout les changements sur les fichiers ajoutés mais pas encore commités :

```
git diff --staged
```

- Afficher les changements de tout les commits a partir d une date/heure donnée (expression de dates, ex : "1 week 2 days" ou une date ISO) :

```
git diff 'HEAD@{3 months|weeks|days|hours|seconds ago}'
```

- Afficher seulement les noms des fichiers modifiés depuis un commit donné :

```
git diff --name-only {{commit}}
```

- Afficher un resumé des creation de fichiers, renomages ou changements de droits depuis un commit :

```
git diff --summary {{commit}}
```

- Comparer un ficheir entre deux branches ou commits :

```
git diff {{branche_1}}..{{branche_2}} [--] {{chemin/vers/
fichier}}
```

- Comparer plusieurs fichiers de la branche courrante avec une autre branche :

```
git diff {{branche}}:{{chemin/vers/fichier2}} {{chemin/vers/
fichier}}
```

git difftool

Afficher les modifications apportées aux fichiers à l'aide d'outils de comparaison externes. Accepte les mêmes options et arguments que Git diff.

Plus d'informations : https://git-scm.com/docs/git-difftool.

- Lister les outils de coparaison disponnible :

```
git difftool --tool-help
```

- Configurer Meld comme outil de comparaison par defaut :

```
git config --global diff.tool "{{meld}}"
```

- Utiliser l'outil de comparaison par defaut pour afficher les fichiers modifiés :

```
git difftool --staged
```

- Utiliser un outil de comparaison spécifique (opendiff) pour afficher les changements depuis un commit :

```
git difftool --tool={{opendiff}} {{commit}}
```

git fetch

Cherche les objets et réferences depuis un registre distant.

Plus d'informations : https://git-scm.com/docs/git-fetch.

- Cherche les dernières modifications du référentiel amont distant par défaut (si configuré) :

  ```
  git fetch
  ```

- Cherche les nouvelles branches depuis un registre distant :

  ```
  git fetch {{nom_distant}}
  ```

- Cherche les nouvelles branches depuis tout les registres distant :

  ```
  git fetch --all
  ```

- Recherche egalement les tags depuis le registre courrant :

  ```
  git fetch --tags
  ```

- Supprime les références locales de branches ayant été supprimés du registre distant :

  ```
  git fetch --prune
  ```

git flow

Une colletion d'extentions Git pour procurer des opperation de registre supplementaires

Plus d'informations : https://github.com/nvie/gitflow.

- Initialiser dans un registre Git existant :

  ```
  git flow init
  ```

- Commencer le travail sur une fonctionnalité basé sur la branche `develop` :

  ```
  git flow feature start {{feature}}
  ```

- Terminer le travail sur une branche de fonctionnalité, le merger dans la branche `develop` puis suprimmer :

  ```
  git flow feature finish {{feature}}
  ```

- Publier une fonctionalité sur le serveur distant :

  ```
  git flow feature publish {{feature}}
  ```

- Recupérer un fonctionalité publiée par un autre utilisateur :

  ```
  git flow feature pull origin {{feature}}
  ```

git format-patch

Preparer des ficchiers de correctifs, utiles pour les envoyer par email.

Voir egalement **`git am`**, qui peut appliquer des fichiers de correctifs genérés.

Plus d'informations : https://git-scm.com/docs/git-format-patch.

-Créer un fichier de correctif `.patch` nommé automatiquement pour tout les commits non poussés :

```
git format-patch {{origin}}
```

- Crér un fichier correctif `.patch` pour les changements entre 2 révisions :

```
git format-patch {{revision_1}}..{{revision_2}}
```

- Créer un fichier correctif `.patch` pour les 3 derniers commits :

```
git format-patch -{{3}}
```

git fsck

Vérifier la validité et la connectivité des nœuds dans un réferentiel Git.

N'applique ucune modification. Voir **`git gc`** pour nettoyer.

Plus d'informations : https://git-scm.com/docs/git-fsck.

- Verifier le registre courrant :

  ```
  git fsck
  ```

- Lister tout les tags trouvés :

  ```
  git fsck --tags
  ```

- Lister tout les noeuds racine trouvés :

  ```
  git fsck --root
  ```

git gc

Optimise le registre local en nettoyant les fichiers inutiles.

Plus d'informations : https://git-scm.com/docs/git-gc.

- Optimise le registrey :

  ```
  git gc
  ```

- Optimise le registre plus agressivement, plus long :

  ```
  git gc --aggressive
  ```

- Afficher les objets a supprimer :

  ```
  git gc --no-prune
  ```

- Supprime tout les objets trouvés sans l'afficher sur la sortie standart :

  ```
  git gc --quiet
  ```

- Afficher le manuel :

  ```
  git gc --help
  ```

git-grep

Rechercher une occurence de texte nomport ou dans l historique d un repository.

Comprends la plus-part des arguments que le **`grep`** classique.

Plus d'informations : https://git-scm.com/docs/git-grep.

- Rechercher une occrence dans les fichiers suivis :

```
git grep {{chaine_recherché}}
```

- Rechercher une occrence dans les fichiers suivis d'appres un pattern de fichiers :

```
git grep {{chaine_recherché}} -- {{file_glob_pattern}}
```

- Rechercher une occrence dans les fichiers suivis et les sous-modules :

```
git grep --recurse-submodules {{chaine_recherché}}
```

- Rechercher une occurence à partir d un point spécifique dans l'historique :

```
git grep {{chaine_recherché}} {{HEAD~2}}
```

- Rechercher une occurence dans toutes les branches :

```
git grep {{chaine_recherché}} $(git rev-list --all)
```

git help

Afficher le manuel de Git.

Plus d'informations : https://git-scm.com/docs/git-help.

- Afficher le manuel d'une sous commande :

```
git help {{subcommand}}
```

- Méme chose dans un navigateur :

```
git help --web {{subcommand}}
```

- Afficher la liste des sous commandes disponnibles :

```
git help --all
```

- Lister les manuels disponnibles :

```
git help --guide
```

- Lister toutes les variables de configuration disponnibles :

```
git help --config
```

git ignore

Génerer le fichier .gitignore depuis des templates prédéfinis.

Plus d'informations : https://docs.gitignore.io/install/command-line.

- Lister les templates disponnibles :

  ```
  git ignore list
  ```

- Genénerer un template .gitignore :

  ```
  git ignore {{item_a,item_b,item_n}}
  ```

git-imerge

Génerer un merge ou un rebase entre deux branches de maniére incrémentale.

Les conflits entre les branches sont suivis en paires de commits individuels, pour simplifier la résolution des conflits.

Plus d'informations : https://github.com/mhagger/git-imerge.

- Démarrer un i-merge rebase (se placer dans la branche a rebaser d'abord) :

  ```
  git imerge rebase {{branche_sur_laquelle_rebaser}}
  ```

- Démarrer imerge merge (se placer dans la branche depuis laquelle merger d'abord) :

  ```
  git imerge merge {{branche_a_merger}}
  ```

- Afficher le diagramme ASCII du merge ou rebase en cours :

  ```
  git imerge diagram
  ```

- Continuer l opperation après une résolution de conflit (d'abord `git add` les fichiers en conflits) :

  ```
  git imerge continue --no-edit
  ```

- Terminer l'opperation i-merge après la resolution de tout les conflits :

  ```
  git imerge finish
  ```

- Annuler l'opperation et retourner à la branche précédante :

  ```
  git-imerge remove && git checkout {{previous_branch}}
  ```

git init

Initialise un nouveau registre Git.

Plus d'informations : https://git-scm.com/docs/git-init.

- Initialise un nouveau registre Git local :

  ```
  git init
  ```

- Initialiser un référentiel barebones, adapté à une utilisation distante via ssh :

  ```
  git init --bare
  ```

git instaweb

Outil pour le lancemend d'un serveur gitweb.

Plus d'informations : https://git-scm.com/docs/git-instaweb.

- Démare un serveur gitweb depuis le repository courrant :

```
git instaweb --start
```

- Écoute uniquement sur le port localhost :

```
git instaweb --start --local
```

- Écoute sur un port spécifique :

```
git instaweb --start --port {{1234}}
```

- Utiliser un daemon http spécifique :

```
git instaweb --start --httpd {{lighttpd|apache2|mongoose|
plackup|webrick}}
```

- Lancer en méme temps qu un navigateur web :

```
git instaweb --start --browser
```

- Stoppe le serveur :

```
git instaweb --stop
```

- Redémarre le serveur :

```
git instaweb --restart
```

git lfs

Travailler dans un registre Git avec des fichiers volumineux.

Plus d'informations : https://git-lfs.github.com.

- Initialise le Git LFS :

  ```
  git lfs install
  ```

- Suivre des fichiers correspondant à un pattern :

  ```
  git lfs track '{{*.bin}}'
  ```

- Changer l'URL du point de terminaison Git LFS (utile si le serveur LFS est séparé du serveur Git) :

  ```
  git config -f .lfsconfig lfs.url {{lfs_endpoint_url}}
  ```

- Lister les pattern de fichiers suivis :

  ```
  git lfs track
  ```

- Lister les fichiers suivis ayant été commité :

  ```
  git lfs ls-files
  ```

- Pousser tout les objets LFS vers le serveur distant :

  ```
  git lfs push --all {{nom_distant}} {{nom_de_branche}}
  ```

- Chercher tout les objets LFS :

  ```
  git lfs fetch
  ```

- Charger tout les objets LFS :

  ```
  git lfs checkout
  ```

git log

Afficher un historique de commits.

Plus d'informations : https://git-scm.com/docs/git-log.

- Afficher la séquence de commits à partir de l'actuel, dans l'ordre chronologique inverse du dépôt Git dans le répertoire de travail actuel :

```
git log
```

- Afficher l'historique de fichiers ou repertoires en particulier :

```
git log -p {{chemin/vers/fichier_ou_repertoire}}
```

- Afficher la liste des fichiers modifiés pour chaque commit :

```
git log --stat
```

- Afficher un graphique des commits dans la branche actuelle en utilisant uniquement la première ligne de chaque message de commit :

```
git log --oneline --graph
```

- Afficher un graphique de tout les commits, tags et branches dans le dépot entier :

```
git log --oneline --decorate --all --graph
```

- Afficher uniqueent les commits dont le message contient la chaine (non sensible à la case) :

```
git log -i --grep {{chaine_recherché}}
```

- Afficher les N derniers commits d'un utilisateur :

```
git log -n {{number}} --author={{author}}
```

- Afficher les commits entre deux dates :

```
git log --before={{date}} --after={{date}}
```

git ls-remote

Commande Git pour répertorier les références dans un dépot distant en fonction du nom ou de l'URL.

Si aucun nom ou URL n'est donné, alors la branche amont configurée sera utilisée, ou l'origine distante si la première n'est pas configurée.

Plus d'informations : https://git-scm.com/docs/git-ls-remote.

- Afficher les références du dépot configuré par défaut :

```
git ls-remote
```

- Afficher uniquement les références HEAD du dépot configuré par défaut :

```
git ls-remote --heads
```

- Afficher uniquement les tags du dépot configuré par défaut :

```
git ls-remote --tags
```

- Afficher les références du dépot précisé :

```
git ls-remote {{repositiory-url}}
```

- Afficher les références du dépot précisé filtrés par un pattern :

```
git ls-remote {{repositiory-name}} "{{pattern}}"
```

git ls-tree

Lister le contenu d'un arbre.

Plus d'informations : https://git-scm.com/docs/git-ls-tree.

- Lister le contenu de l'arbre dans la branche :

  ```
  git ls-tree {{nom_de_branche}}
  ```

- Lister le contenu de l'arbre dans le commit, recursif avec les sous-arbres :

  ```
  git ls-tree -r {{commit_hash}}
  ```

- Lister uniquement les moms de fichiers de l'arbre dans un commit :

  ```
  git ls-tree --name-only {{commit_hash}}
  ```

git merge

Merge branches.

Plus d'informations : https://git-scm.com/docs/git-merge.

- Merge une branche dans votre branche courrante :

  ```
  git merge {{nom_de_branche}}
  ```

- Editer le message de merge :

  ```
  git merge -e {{nom_de_branche}}
  ```

- Merge une branche et créer un commit de merge :

  ```
  git merge --no-ff {{nom_de_branche}}
  ```

- Annuler un merge en cas de conflit :

  ```
  git merge --abort
  ```

- Continuer un merge apreés une résolution de conflit :

  ```
  git merge --continue
  ```

git mergetool

Executer un utilitaire de différences pour resoudre les conflits de merge.

Plus d'informations : https://git-scm.com/docs/git-mergetool.

- Démarrer l'outil de différences par défaut :

```
git mergetool
```

- Lister les outils de différences valides :

```
git mergetool --tool-help
```

- Démarrer l'outil de différences en précisant son nom :

```
git mergetool --tool {{tool_name}}
```

- Démarer l'outil de différerences sans dialogues :

```
git mergetool --no-prompt
```

- Utiliser explicitement l'outil de différences graphique (voir la variable de config `merge.guitool`) :

```
git mergetool --gui
```

- tiliser explicitement l'outil de différences clasique (voir la variable de config `merge.tool`) :

```
git mergetool --no-gui
```

git mv

Déplacer ou renomer des fichiers inscrits dans l'index.

Plus d'informations : https://git-scm.com/docs/git-mv.

- Déplace les fichiers dans l'index Git, valide au prochain commit :

  ```
  git mv {{chemin/vers/fichier}} {{new/path/to/file}}
  ```

- Renome un fichier et met a jour l'index, valide au prochain commit :

  ```
  git mv {{filename}} {{new_filename}}
  ```

- Force l'écrasement d'un fichier :

  ```
  git mv --force {{file}} {{cible}}
  ```

git notes

Ajoute ou inspecte des notes d'objets.

Plus d'informations : https://git-scm.com/docs/git-notes.

- Lister toutes les notes et leurs objets rattachés :

  ```
  git notes list
  ```

- Lister toutes les notes attaches a un objet donné :

  ```
  git notes list [{{object}}]
  ```

- Afficher les notes attachés a un objet donné :

  ```
  git notes show [{{object}}]
  ```

- Ajoute une note à un objet donné :

  ```
  git notes append {{object}}
  ```

- Ajoute une note à un objet donné, en spécifiant le message :

  ```
  git notes append --message="{{message_text}}"
  ```

- Edite une note éxistante :

  ```
  git notes edit [{{object}}]
  ```

- Copy la note d'un objet vers un autre :

  ```
  git notes copy {{source_object}} {{objet_cible}}
  ```

- Supprime toutes les notes d'un objet spécifié :

  ```
  git notes remove {{object}}
  ```

git pr

Récupére les pull-request GitHub localement.

- Récupére une pull-request spécifique :

  ```
  git pr {{pr_number}}
  ```

- Récupére une pull-request d un dépot spécifique :

  ```
  git pr {{pr_number}} {{distant}}
  ```

- Récupére une pull-request depuis sont url :

  ```
  git pr {{url}}
  ```

- Nettoie les branches de pull-request terminés :

  ```
  git pr clean
  ```

git pull

Récupère une branche depuis le serveur distant et la fusionne dans la branche local.

Plus d'informations : https://git-scm.com/docs/git-pull.

- Télécharge les changements depuis le serveur distant par défaut et fusionne les :

  ```
  git pull
  ```

- Télécharge les changements depuis le serveur distant par défaut et applique les changements locaux par dessus :

  ```
  git pull --rebase
  ```

- Télécharge les changements depuis un serveur et une branche distante, puis fusionne les dans HEAD :

  ```
  git pull {{nom_distant}} {{branche}}
  ```

git push

Pousse les commits vers un dépôt distant.

Plus d'informations : https://git-scm.com/docs/git-push.

- Envoie les changements locaux dans la branche courante vers sa contrepartie distante :

  ```
  git push
  ```

- Envoie les changements locaux d'une branche spécifique vers sa contrepartie distante :

  ```
  git push {{nom_distant}} {{local_branch}}
  ```

- Publie la branche courante vers un dépôt distant, crée le nom de la branche distante :

  ```
  git push {{nom_distant}} -u {{branche_distante}}
  ```

- Envoi les changements locaux sur toutes les branches locales vers leur contrepartie sur le dépôt distant :

  ```
  git push --all {{nom_distant}}
  ```

- Supprime une branche dans un dépôt distant :

  ```
  git push {{nom_distant}} --delete {{branche_distante}}
  ```

- Supprime les branches distantes qui n'ont pas de contrepartie locale :

  ```
  git push --prune {{nom_distant}}
  ```

- Publie les tags qui ne sont pas sur le dépôt distant :

  ```
  git push --tags
  ```

git rebase

Rejoue les commits d'une branche par dessus une autre.

Communément utilisé pour dupliquer les commits d'une branche dans une autre, en créant de nouveaux commits dans la branche de destination.

Plus d'informations : https://git-scm.com/docs/git-rebase.

- Rejouer les commits de la branche courrante sur la branche master :

  ```
  git rebase {{master}}
  ```

- Rejouer les comits interactivement, ce qui permet aux commits d'éte re-arrangés, exclus, combimés ou modifiés :

  ```
  git rebase -i {{branche_de_base_ou_commit}}
  ```

- Continuer le re-jeu des commits apres la resolution d'un conflit :

  ```
  git rebase --continue
  ```

- Continuer le re-jeu des commits en sautant la résolution d'un conflit :

  ```
  git rebase --skip
  ```

- Annule l'operation (ex : en cas de conflict) :

  ```
  git rebase --abort
  ```

- Déplacez une partie de la branche actuelle sur une nouvelle base, fournissant l'ancienne base à partir de laquelle commencer :

  ```
  git rebase --onto {{new_base}} {{old_base}}
  ```

- Rejoue les 5 derniers commits, ce qui permet aux commits d'éte re-arrangés, exclus, combimés ou modifiés :

  ```
  git rebase -i {{HEAD~5}}
  ```

- Resoudre automatiquement les conflits en precisant la version a conserver (`theirs` signifie la version des fichiers a privilégier) :

  ```
  git rebase -X theirs {{master}}
  ```

git reflog

Affiche un log des changements locaux comme HEAD, tags et branches.

Plus d'informations : https://git-scm.com/docs/git-reflog.

- Afficher le reflog de HEAD :

```
git reflog
```

- Affiche le reflog d'une branche spécifique :

```
git reflog {{nom_de_branche}}
```

- Affiche les 5 dérniéres entrés dans le reflog :

```
git reflog -n {{5}}
```

git remote

Organisation des dépots suivis ("remotes").

Plus d'informations : https://git-scm.com/docs/git-remote.

- Affiche les dépots existants, leur nom et url :

  ```
  git remote -v
  ```

- Affiche les informations a propos d'un dépot :

  ```
  git remote show {{nom_distant}}
  ```

- Ajoute un dépot :

  ```
  git remote add {{nom_distant}} {{url_distant}}
  ```

- Change l'url d'un dépot (ajouter `--add` pour conserver l'url existante) :

  ```
  git remote set-url {{nom_distant}} {{new_url}}
  ```

- Suprime un dṕot :

  ```
  git remote remove {{nom_distant}}
  ```

- Renome un dépot :

  ```
  git remote rename {{old_name}} {{new_name}}
  ```

git repack

Empaqueter les objets décompressés dans un dépot Git.

Plus d'informations : https://git-scm.com/docs/git-repack.

- Empaqueter les objets décompressés dans le dépot courrant :

```
git repack
```

- Egalement suprime les objets redondants après empaquetage :

```
git repack -d
```

git request-pull

Générer une requête demandant au projet en amont d'inclure les modifications dans son arborescence.

Plus d'informations : https://git-scm.com/docs/git-request-pull.

- Produire une requête résumant les changements entre la version v1.1 et le master :

```
git request-pull {{v1.1}} {{https://example.com/project}}
{{master}}
```

- Produire une requête résumant les changements entre la version v1.1 sur la branche master et la branche locale foo :

```
git request-pull {{v0.1}} {{https://example.com/project}}
{{master:foo}}
```

git reset

Enlève des commits ou des changements en réinitialisant la tête Git à l'état spécifié.

Si un chemin est passé en paramètre, Git reset fonctionne comme «unstage».

Si un hash de commit est passé en paramètre, Git reset annule les commits jusqu'à ce dernier.

Plus d'informations : https://git-scm.com/docs/git-reset.

- Tout enlever de la zone de stage :

```
git reset
```

- Enlever des fichiers spécifiques de la zone de stage :

```
git reset {{chemin/vers/fichier(s)}}
```

- Enlever une portion d'un fichier de la zone de stage :

```
git reset -p {{chemin/vers/fichier}}
```

- Annuler le dernier commit, mais garder les chagements éffectués dans votre système de fichier :

```
git reset HEAD~
```

- Défaire les deux derniers commits, et ajouter leur changements à l'index adding their changes to the index (dans la zone de stage) :

```
git reset --soft HEAD~2
```

- Enlever tout les changements qui n'ont pas été commit, qu'ils soient dans la zone de stage ou non (pour enlever seulement les changements de la zone de stage, utiliser `git checkout`) :

```
git reset --hard
```

- Réinitialiser le dépôt à un commit spécifique en retirant tout les changements (ceci inclus les changements dans des commits entre la tête et le commit spécifié!) :

```
git reset --hard {{commit}}
```

git restore

Restaurez les fichiers de l'arborescence de travail. Nécessite la version 2.23+ de Git.

Voir aussi **`git checkout`**.

Plus d'informations : https://git-scm.com/docs/git-restore.

- Restaurer un fichier supprimé à partir du contenu du commit actuel (HEAD) :

```
git restore {{chemin/vers/fichier}}
```

- Restaurer un fichier a la version d'un commit spécifié :

```
git restore --source {{commit}} {{chemin/vers/fichier}}
```

- Annulez toutes les modifications non validées des fichiers suivis, en revenant au HEAD :

```
git restore .
```

git rev-list

Liste les révisionts (commits) dans l'ordre chronologique inverse.

Plus d'informations : https://git-scm.com/docs/git-rev-list.

- Lister tout les commits dans la branche courrante :

```
git rev-list {{HEAD}}
```

- Lister tout les commits plus récents qu'une date spécifique, sur une branche spécifique :

```
git rev-list --since={{'2019-12-01 00:00:00'}} {{master}}
```

- Lister tout les commits de merge depuis un commit spécifique :

```
git rev-list --merges {{commit}}
```

git rev-parse

Afficher les métadonnées liées à des révisions spécifiques.

Plus d'informations : https://git-scm.com/docs/git-rev-parse.

- Afficher le hash de commit de la branche courrante :

  ```
  git rev-parse {{nom_de_branche}}
  ```

- Affiche le nom de la branche courrante :

  ```
  git rev-parse --abbrev-ref {{HEAD}}
  ```

- Obtenir le chamin absolu du repertoire racine :

  ```
  git rev-parse --show-toplevel
  ```

git revert

Créer un nouveau commit qui efface les changements du précedant.

Plus d'informations : https://git-scm.com/docs/git-revert.

- Crée un commit qui annule les changements du dernier commit :

  ```
  git revert {{@}}
  ```

- Crée un commit qui annule les changements des 5 dernier commit :

  ```
  git revert HEAD~{{4}}
  ```

- Crée un commit qui annule les changements de plusieurs commit :

  ```
  git revert {{master~5..master~2}}
  ```

- Ne pas créer de nouveau commit, remplacer uniquement dans l'arbre courrant :

  ```
  git revert -n {{0c01a9..9a1743}}
  ```

git rm

Supprimer des fichiers de l'index, du dépot et du systéme de fichiers.

Plus d'informations : https://git-scm.com/docs/git-rm.

- Supprimer un fichiers de l'index, du dépot et du systéme de fichiers :

```
git rm {{file}}
```

- Supprimer un répertoire de l'index, du dépot et du systéme de fichiers :

```
git rm -r {{directory}}
```

- Supprimer un fichiers de l'index, du dépot mais le conserve sur le systéme de fichiers :

```
git rm --cached {{file}}
```

git send-email

Envoyer une collection de correctifs par email.

Les correctifs peuvent être spécifiés sous forme de fichiers, de directions ou de liste de révisions.

Plus d'informations : https://git-scm.com/docs/git-send-email.

- Envoyer le dernier commit de la branche courrante :

  ```
  git send-email -1
  ```

- envoyer un commit spécifiaue :

  ```
  git send-email -1 {{commit}}
  ```

- envoyer de multiples commits de la branche courrante (ici : 10) :

  ```
  git send-email {{-10}}
  ```

- Envoyez un e-mail de présentation de la série de correctifs :

  ```
  git send-email -{{number of commits}} --compose
  ```

- Consultez et modifiez l'e-mail de chaque patch que vous êtes sur le point d'envoyer :

  ```
  git send-email -{{number of commits}} --annotate
  ```

git shortlog

Récapitule la sortie de **`git log`**.

Plus d'informations : https://git-scm.com/docs/git-shortlog.

- Afficher un résumé de tous les commits effectués, regroupés par ordre alphabétique par nom d'auteur :

  ```
  git shortlog
  ```

- Afficher un résumé de tous les commits effectués, regroupés par le nombre de commits effectués :

  ```
  git shortlog -n
  ```

- Afficher un résumé de tous les commits effectués, regroupés par le nom et l'email de l'utilisateur :

  ```
  git shortlog -c
  ```

- ffficher un résumé des 5 derniers commits effectués :

  ```
  git shortlog HEAD~{{5}}..HEAD
  ```

- Afficher tout les utilisateurs, emails et le nombre de commits dans la branche :

  ```
  git shortlog -sne
  ```

- Afficher tout les utilisateurs, emails et le nombre de commits dans toutes les branches :

  ```
  git shortlog -sne --all
  ```

git show-branch

Affiche les branches et leurs commits.

Plus d'informations : https://git-scm.com/docs/git-show-branch.

- Affiche un resumé du dernier commit dans la branche :

```
git show-branch {{nom_de_branche}}|ref|commit}}
```

- Comparer des commits avec plusieurs commits ou branches :

```
git show-branch {{nom_de_branche}}|ref|commit}}
```

- Comparer toutes les branches distantes :

```
git show-branch --remotes
```

- Comparer le branche locale avec la branche distante :

```
git show-branch --all
```

- Lister les derniers commits sur toutes les branches :

```
git show-branch --all --list
```

- Compareer une branche spécifique a la branche courante :

```
git show-branch --current {{commit|branch_name|ref}}
```

- Afficher le nom du commit au lieu du nom relatif :

```
git show-branch --sha1-name --current {{current|branch_name|
ref}}
```

- Continuez l'affichage d'un certain nombre de commits au-delà de l'ancêtre commun :

```
git show-branch --more {{5}} {{commit|branch_name|ref}}
{{commit|branch_name|ref}} {{...}}
```

git show-ref

commande Git pour lister les références.

Plus d'informations : https://git-scm.com/docs/git-show-ref.

- Affiche toutes les références dans le dépot :

  ```
  git show-ref
  ```

- Affiche seulement les références des têtes de branches :

  ```
  git show-ref --heads
  ```

- Affiche seulement les références de tags :

  ```
  git show-ref --tags
  ```

- Verifier l'existence d'une référence :

  ```
  git show-ref --verify {{chemin/vers/reference}}
  ```

git show

Affiche différents types d'objets Git (commits, tags, etc.).

Plus d'informations : https://git-scm.com/docs/git-show.

- Afficher des informations sur le dernier commit (hachage, message, modifications et autres métadonnées) :

  ```
  git show
  ```

- Affiche les informations du dernier commit :

  ```
  git show {{commit}}
  ```

- Affiche les informations associés au tag spécifié :

  ```
  git show {{etiquette}}
  ```

- Affiche les informations a propos du 3éme commit en partant du sommet de la branche :

  ```
  git show {{branche}}~{{3}}
  ```

- Afficher le message d'un commit sur une seule ligne, en supprimant la sortie diff :

  ```
  git show --oneline -s {{commit}}
  ```

- Affiche uniquement la liste des fichiers changés dans un commit :

  ```
  git show --stat {{commit}}
  ```

- Afficher le contenu d'un fichier tel qu'il était à une révision donnée (par exemple, branche, tag ou commit) :

  ```
  git show {{revision}}:{{chemin/vers/fichier}}
  ```

git sizer

Calcule diverses métriques de taille de dépot Git et vous alerte de tout ce qui pourrait causer des problèmes ou des inconvénients.

Plus d'informations : https://github.com/github/git-sizer.

- Signaler uniquement les statistiques dont le niveau de préoccupation est supérieur à 0 :

  ```
  git sizer
  ```

- Signaler toutes les statistiques :

  ```
  git sizer -v
  ```

- Afficher les options additionelles :

  ```
  git sizer -h
  ```

git stage

Ajouter le contenu du fichier à la zone de préparation.

Synonym of **`git add`**.

Plus d'informations : https://git-scm.com/docs/git-stage.

- Ajouter un fichier à l'index :

```
git stage {{chemin/vers/fichier}}
```

- Ajoute tout les fichiers à l'index (suivis et non suivis) :

```
git stage -A
```

- Ajout uniquement des fichiers déja suivis :

```
git stage -u
```

- Ajout egalement des fichiers ignorés :

```
git stage -f
```

- Ajout des fichiers par parties, interactivement :

```
git stage -p
```

- Ajout d'un fichier par parties, interactivement :

```
git stage -p {{chemin/vers/fichier}}
```

- Ajout d'un fichier, interactivement :

```
git stage -i
```

git stash

Stocker les modifications Git locales dans une zone temporaire.

Plus d'informations : https://git-scm.com/docs/git-stash.

- Stocker les changements courrants, sauf les fichiers non suivis :

  ```
  git stash [push -m {{nom_de_stash_optionel}}]
  ```

- Stocker les changements courrants, incluant le sfichiers non suivis :

  ```
  git stash -u
  ```

- Stocker les parties d'un fichier interactivement :

  ```
  git stash -p
  ```

- Lister tout les stash (saffiche leurs noms, les branches relatives et messages) :

  ```
  git stash list
  ```

- Applique un stash (par défaut, le dernier, nommé stash@{0}) :

  ```
  git stash apply {{nom_de_stash_ou_de_commit_optionel}}
  ```

- Applique un stash (par défaut le dernier, stash@{0}), et le suprimmer de la liste des stash si il n'y a pas de conflits :

  ```
  git stash pop {{nom_de_stash_optionel}}
  ```

- Suprime un stash (par défaut le dernier, stash@{0}) :

  ```
  git stash drop {{nom_de_stash_optionel}}
  ```

- Suprime tout les stash :

  ```
  git stash clear
  ```

git status

Affiche les changements sur les fichiers dans la branche courrante.

Affiche les fichiers édités, deplacés, renomés, ajoutés, suprimés par rapport a la référence de la branche courrante.

Plus d'informations : https://git-scm.com/docs/git-status.

- Affiche les fichiers mofifiés qui n ont pas encore été ajoutés pour le commit :

  ```
  git status
  ```

- Affiche les fichiers mofifiés (version courte) :

  ```
  git status -s
  ```

- Affiche les fichiers mofifiés, sans tenir des comptes des fichiers non suivis :

  ```
  git status --untracked-files=no
  ```

- Affiche les fichiers mofifiés (version courte) avec les informations de branche :

  ```
  git status -sb
  ```

git submodule

Inspecter, metre a jour et manager des sous modules.

Plus d'informations : https://git-scm.com/docs/git-submodule.

- Installer un submodule depuis le dépot courrant :

```
git submodule update --init --recursive
```

- Ajout d'un dépot Git en tant que sous module :

```
git submodule add {{repository_url}}
```

- Ajout d'un dépot Git en tant que sous module a répertoire donné :

```
git submodule add {{repository_url}} {{chemin/vers/
repertoire}}
```

- Metre à jour tout les sous modules à leurs derniers commit :

```
git submodule foreach git pull
```

git subtree

Outil pour gérer les dépendances de projet en tant que sous-projets.

Plus d'informations : https://manpages.debian.org/testing/git-man/git-subtree.1.en.html.

- Ajout d'un dépot Git en tant que sous arbre :

```
git subtree add --prefix={{chemin/vers/repertoire/}} --squash
{{repository_url}} {{master}}
```

- Metre a jour le sous arbre avec son dernier commit :

```
git subtree pull --prefix={{chemin/vers/repertoire/}}
{{repository_url}} {{master}}
```

- Merge le dépot d'un sous arbre dans la branche master :

```
git subtree merge --prefix={{chemin/vers/repertoire/}} --
squash {{repository_url}} {{master}}
```

- Pousser les commits vers le dépot d'un sous arbre :

```
git subtree push --prefix={{chemin/vers/repertoire/}}
{{repository_url}} {{master}}
```

- Extraire un nouvel historique de projet de l'historique d'un sous-arbre :

```
git subtree split --prefix={{chemin/vers/repertoire/}}
{{repository_url}} -b {{nom_de_branche}}
```

git svn

Opération bidirectionnelle entre un référentiel Subversion et Git.

Plus d'informations : https://git-scm.com/docs/git-svn.

- Cloner un dépot SVN :

```
git svn clone {{https://example.com/subversion_repo}} {{local_dir}}
```

- Cloner un dépot SVN a partir d'une révision donnée :

```
git svn clone -r{{1234}}:HEAD {{https://svn.example.net/subversion/repo}} {{local_dir}}
```

- Metre a jourr le clone local a partir du depot SVN distant :

```
git svn rebase
```

- Chercher les changements distants dans le dépot SVN dans les appliquer sur le HEAD :

```
git svn fetch
```

- Commiter sur le SVN :

```
git svn dcommit
```

git switch

Basculez entre les branches Git. Nécessite la version 2.23+ de Git.

Voir egalement **`git checkout`**.

Plus d'informations : https://git-scm.com/docs/git-switch.

- Baculer sur une branche existante :

  ```
  git switch {{nom_de_branche}}
  ```

- Créer une nouvelle branche et basculer dessus :

  ```
  git switch --create {{nom_de_branche}}
  ```

- Créer une nouvelle branche en partant d'un commit donné et basculer dessus :

  ```
  git switch --create {{nom_de_branche}} {{commit}}
  ```

- Basculer sur la branche précedante :

  ```
  git switch -
  ```

- Basculer vers une branche et mettre à jour tous les sous-modules pour qu'ils correspondent :

  ```
  git switch --recurse-submodules {{nom_de_branche}}
  ```

- Basculer vers une branche et fusionner automatiquement la branche actuelle et toutes les modifications non validées dedans :

  ```
  git switch --merge {{nom_de_branche}}
  ```

git tag

Créer, lister, verifier et suprimer des tags.

Un tag est une reférence statique vers un commit.

Plus d'informations : https://git-scm.com/docs/git-tag.

- Lister tout les tags :

```
git tag
```

- Créer un tag avec le nom donné pointant vers le commit actuel :

```
git tag {{nom_d_etiquette}}
```

- Créer un tag avec le nom donné pointant vers un commit spécifié :

```
git tag {{nom_d_etiquette}} {{commit}}
```

- Créer un tag annoté avec le message spécifié :

```
git tag {{nom_d_etiquette}} -m {{message_d_etiquette}}
```

- Suprimer le tag avec le nom spécifié :

```
git tag -d {{nom_d_etiquette}}
```

- Metre a jour les tags depuis l'origine :

```
git fetch --tags
```

- Liste toutes les tags dont les ancêtres incluent un commit donné :

```
git tag --contains {{commit}}
```

git update-index

Commande Git pour manipuler l'index.

Plus d'informations : https://git-scm.com/docs/git-update-index.

- Prétendre qu'un fichier modifié est inchangé (`git status` ne l'affichera pas comme modifié) :

  ```
  git update-index --skip-worktree {{chemin/vers/
  fichier_modifié}}
  ```

git update-ref

Commande Git pour créer, mettre à jour et supprimer des références Git.

Plus d'informations : https://git-scm.com/docs/git-update-ref.

- Supprimer une référence, utile pour la réinitialisation du premier commit :

```
git update-ref -d {{HEAD}}
```

- Mettre a jour une référence avec un message :

```
git update-ref -m {{message}} {{HEAD}} {{4e95e05}}
```

git worktree

Gérez plusieurs arborescences de travail attachées au même dépot.

Plus d'informations : https://git-scm.com/docs/git-worktree.

- Créer un nouveau sous arbre avec la branche spécifiée extraite dedans :

  ```
  git worktree add {{chemin/vers/repertoire}} {{branche}}
  ```

- Créer un nouveau sous arbre branche extraite dedans :

  ```
  git worktree add {{chemin/vers/repertoire}} -b
  {{nouvelle_branche}}
  ```

- Répertoriez tous les sous arbres attachés à ce dépot :

  ```
  git worktree list
  ```

- Suprime les sous arbres (apres avoir suprimé les repertoires de travail) :

  ```
  git worktree prune
  ```

git

Système de gestion de versions décentralisé.

Plus d'informations : https://git-scm.com/.

- Obtenir la version de Git :

  ```
  git --version
  ```

- Afficher l'aide générale :

  ```
  git --help
  ```

- Afficher l'aide sur une commande Git :

  ```
  git help {{command}}
  ```

- Exécuter une commande Git :

  ```
  git {{command}}
  ```

grep

Recherche des motifs dans un texte.

Supporte des motifs simples et des expressions regulières.

- Recherche une chaîne de caractères précise :

```
grep {{chaîne_recherchée}} {{chemin/vers/fichier}}
```

- Recherche en ignorant la casse :

```
grep -i {{chaîne_recherchée}} {{chemin/vers/fichier}}
```

- Recherche récursivement (en ignorant les fichiers non-texte) dans le dossier courant une chaîne de caractères précise :

```
grep -RI {{chaîne_recherchée}} .
```

- Utilise des expressions rationnelles étendues (supporte `?`, `+`, `{}`, `( )` et `|`) :

```
grep -E {{^regex$}} {{chemin/vers/fichier}}
```

- Affiche 3 lignes de [C]ontexte, avant ([B]efore), ou [A]près chaque concordance :

```
grep -{{C|B|A}} 3 {{chaîne_recherchée}} {{chemin/vers/
fichier}}
```

- Affiche le nom du fichier avec la ligne correspondante pour chaque concordance :

```
grep -Hn {{chaîne_recherchée}} {{chemin/vers/fichier}}
```

- Utilise l'entrée standard au lieu d'un fichier :

```
cat {{chemin/vers/fichier}} | grep {{chaîne_recherchée}}
```

- Inverse le résultat pour exclure des chaînes de caractères spécifiques :

```
grep -v {{chaîne_recherchée}}
```

hping

Outil en ligne de commande permettant d'assembler ou analyser des paquets TCP/IP.

Inspirer par la commande **`ping`**.

Plus d'informations : http://www.hping.org.

- Ping localhost via TCP :

```
hping3 {{localhost}}
```

- Ping une adresse IP, via TCP, sur un port spécifique :

```
hping3 -p {{80}} -S {{192.168.1.1}}
```

- Ping une adresse IP, via UDP, sur le port 80 :

```
hping3 --udp -p {{80}} -S {{192.168.1.1}}
```

- Scanner un ensemble de ports TCP, sur une adresse IP spécifique :

```
hping3 --scan {{80,3000,9000}} -S {{192.168.1.1}}
```

- Effectuer un test de montée en charge sur le port 80 :

```
hping3 --flood -p {{80}} -S {{192.168.1.1}}
```

ifconfig

Configurateur des interfaces réseau.

- Affiche les paramètres de réseau d'un adaptateur ethernet :

  ```
  ifconfig {{eth0}}
  ```

- Affiche les détails de toutes les interfaces, y compris les interfaces désactivées :

  ```
  ifconfig -a
  ```

- Désactive l'interface eth0 :

  ```
  ifconfig {{eth0}} down
  ```

- Active l'interface eth0 :

  ```
  ifconfig {{eth0}} up
  ```

- Assigne une addresse IP à l'interface eth0 :

  ```
  ifconfig {{eth0}} {{addresse_ip}}
  ```

install

Copie des fichiers et mets à jour leurs attributs.

Copie des fichiers (souvent des exécutables) dans un répertoire système comme **`/usr/local/bin`** et leur donne les permissions et propriétaires appropriés.

- Copie des fichiers vers une destination :

```
install {{chemin/fichier/source}} {{chemin/repertoire/destination}}
```

- Copie des fichiers vers une destination en mettant à jour leur propriétaire :

```
install -o {{utilisateur}} {{chemin/fichier/source}} {{chemin/repertoire/destination}}
```

- Copie des fichiers vers une destination en mettant à jour leur groupe d'appartenance :

```
install -g {{utilisateur}} {{chemin/fichier/source}} {{chemin/repertoire/destination}}
```

- Copie des fichiers vers une destination en mettant à jour leur mode :

```
install -m {{+x}} {{chemin/fichier/source}} {{chemin/repertoire/destination}}
```

- Copie des fichiers en mettant à jour leur dates d'accès et de modification à partir de leurs sources respectives :

```
install -p {{chemin/fichier/source}} {{chemin/repertoire/destination}}
```

jekyll

Générateur de site statique simple et convenable aux blogs.

Plus d'informations : https://jekyllrb.com.

- Génère un serveur de développement qui tourne au http://localhost:4000/ :

  ```
  jekyll serve
  ```

- Active la régénération incrémentale :

  ```
  jekyll serve --incremental
  ```

- Active la sortie verbeuse :

  ```
  jekyll serve --verbose
  ```

- Génère le répertoire actuel dans `./_site` :

  ```
  jekyll build
  ```

- Nettoie le site (c.-à.-d. supprime la sortie du site et le répertoire `cache`) sans compiler :

  ```
  jekyll clean
  ```

jest

Une plateforme de test JavaScript avec zero configuration.

Plus d'informations : https://jestjs.io.

- Exécuter tous les tests disponibles :

  ```
  jest
  ```

- Exécuter les suites de test des fichiers dont les chemins correspondent aux expressions régulières indiquées :

  ```
  jest {{fichier_test1}} {{chemin/vers/fichier_test2.js}}
  ```

- Exécuter les tests dont les noms correspondent aux expressions régulières indiquées :

  ```
  jest --testNamePattern {{nom_test}}
  ```

- Exécuter les suites de test associées à un fichier source donné :

  ```
  jest --findRelatedTests {{chemin/vers/fichier_source.js}}
  ```

- Exécuter les suites de test associées à tous les fichiers non commités :

  ```
  jest --onlyChanged
  ```

- Surveiller les changements sur les fichiers et réexécuter les tests associés :

  ```
  jest --watch
  ```

- Montrer l'aide :

  ```
  jest --help
  ```

kosmorro

Calcule les éphémérides et les événements pour une date donnée, à un emplacement donné sur Terre.

Plus d'informations : http://kosmorro.space.

- Obtenir les éphémérides pour Paris (France) :

```
kosmorro --latitude={{48.7996}} --longitude={{2.3511}}
```

- Obtenir les éphémérides pour Paris (France), sur le fuseau horaire UTC+2 :

```
kosmorro --latitude={{48.7996}} --longitude={{2.3511}} --
timezone={{2}}
```

- Obtenir les éphémérides du 9 juin 2020 pour Paris (France) :

```
kosmorro --latitude={{48.7996}} --longitude={{2.3511}} --
date={{2020-06-09}}
```

- Générer un fichier PDF (TeXLive doit être installé) :

```
kosmorro --format={{pdf}} --output={{chemin/vers/le/
fichier.pdf}}
```

ln

Crée des liens vers des fichiers et répertoires.

- Crée un lien symbolique vers un fichier ou un répertoire :

```
ln -s {{chemin/vers/fichier_ou_repertoire}} {{chemin/vers/
lien_symbolique}}
```

- Modifie la cible d'un lien symbolique existant :

```
ln -sf {{chemin/vers/nouveau_fichier}} {{chemin/vers/
lien_symbolique}}
```

- Crée un lien dur vers un fichier :

```
ln {{chemin/vers/fichier}} {{chemin/vers/lien_dur}}
```

ls

Liste le contenu d'un répertoire.

- Liste les fichiers, un par ligne :

  ```
  ls -1
  ```

- Liste tous les fichiers, ainsi que les fichiers cachés :

  ```
  ls -a
  ```

- Liste tous les fichiers avec un format détaillé (permissions, propriétaire, taille et date de modification) :

  ```
  ls -la
  ```

- Liste les fichiers avec un format détaillé en utilisant des prefixes d'unités (ko, Mo, Go) :

  ```
  ls -lh
  ```

- Liste les fichiers avec un format détaillé en triant par taille décroissante :

  ```
  ls -lS
  ```

- Liste avec un format détaillé tous les fichiers en triant par date de modification (les plus anciennes en premier) :

  ```
  ls -ltr
  ```

lua

Un langage de programmation puissant, léger, et convenable aux

> systèmes embarqués.

Plus d'informations : https://www.lua.org.

- Démarre une session de commandes intéractive Lua :

  ```
  lua
  ```

- Exécute un script Lua :

  ```
  lua {{nom_du_script.lua}} {{--arguments-facultatifs}}
  ```

- Exécute une expression Lua :

  ```
  lua -e '{{print( "Hello World" )}}
  ```

matlab

Environnement de calcul numérique créé par MathWorks.

Plus d'informations : https://fr.mathworks.com/help/matlab/.

- Lance MATLAB sans afficher l'écran de démarrage :

  ```
  matlab -nosplash
  ```

- Exécute une instruction MATLAB :

  ```
  matlab -r "{{instruction_matlab}}"
  ```

- Exécute un script MATLAB :

  ```
  matlab -r "run({{chemin/vers/script.m}})"
  ```

mc

Midnight Commander, gestionnaire de fichiers à base de console.

La navigation dans les répertoires se fait à l'aide des touches directionnelles ou la souris, ou bien en tapant des commandes dans la console.

Plus d'informations : https://midnight-commander.org

- Démarre `mc` :

  ```
  mc
  ```

- Démarre `mc` en mode noir et blanc :

  ```
  mc -b
  ```

mkdir

Crée un répertoire.

- Crée un répertoire dans le répertoire actuel ou dans un chemin donné :

  ```
  mkdir {{répertoire}}
  ```

- Crée des répertoires récursivement (utile pour créer des répertoires imbriqués) :

  ```
  mkdir -p {{chemin/vers/répertoire}}
  ```

nano

Éditeur de texte simple et convivial. C'est un clone libre et amélioré de Pico.

Plus d'informations : https://nano-editor.org.

- Ouvre un fichier :

```
nano {{chemin/vers/fichier}}
```

- Ouvre un fichier en positionnant le curseur à une rangée et colonne donnée :

```
nano +{{ligne}},{{colonne}} {{chemin/vers/fichier}}
```

- Active le défilement fluide :

```
nano -S {{fichier}}
```

- Indente les nouvelles lignes à la même indentation que les lignes précédentes :

```
nano -i {{fichier}}
```

- Avant la modification, sauvegarde le fichier actuel sous le format `{{nom_du_fichier_actuel}}~` :

```
nano -B {{fichier}}
```

pip

Gestionnaire des paquets pour Python.

Plus d'informations : https://pip.pypa.io.

- Installe un paquet :

  ```
  pip install {{paquet}}
  ```

- Installe une version particulière d'un paquet :

  ```
  pip install {{paquet}}=={{version}}
  ```

- Met à jour un paquet :

  ```
  pip install -U {{paquet}}
  ```

- Désinstalle un paquet :

  ```
  pip uninstall {{paquet}}
  ```

- Sauvegarde une liste des paquets installés :

  ```
  pip freeze > {{requirements.txt}}
  ```

- Installe des paquets à partir d'un fichier :

  ```
  pip install -r {{requirements.txt}}
  ```

- Affiche les informations d'un paquet installé :

  ```
  pip show {{paquet}}
  ```

pip3

Gestionnaire des paquets pour Python.

Plus d'informations : https://pip.pypa.io.

- Recherche un paquet :

```
pip3 search {{paquet}}
```

- Installe un paquet :

```
pip3 install {{paquet}}
```

- Installe une version particulière d'un paquet :

```
pip3 install {{paquet}}=={{version}}
```

- Met à jour un paquet :

```
pip3 install --upgrade {{paquet}}
```

- Désinstalle un paquet :

```
pip3 uninstall {{paquet}}
```

- Sauvegarde une liste des paquets installés :

```
pip3 freeze > {{requirements.txt}}
```

- Installe des paquets à partir d'un fichier :

```
pip3 install --requirements {{requirements.txt}}
```

- Affiche les informations d'un paquet installé :

```
pip3 show {{paquet}}
```

pipenv

Workflow de développement simple et unifié pour Python.

Gère les paquets et l'environnement virtuel d'un projet.

Plus d'informations : https://pypi.org/project/pipenv.

- Crée un nouveau projet :

  ```
  pipenv
  ```

- Crée un nouveau projet à l'aide de Python 3 :

  ```
  pipenv --three
  ```

- Installe un paquet :

  ```
  pipenv install {{paquet}}
  ```

- Installe toutes les dépendances d'un projet :

  ```
  pipenv install
  ```

- Installe toutes les dépendances d'un projet (y compris les paquets de développement) :

  ```
  pipenv install --dev
  ```

- Désinstalle un paquet :

  ```
  pipenv uninstall {{paquet}}
  ```

- Démarre une session de commandes dans l'environnement virtuel :

  ```
  pipenv shell
  ```

- Génère un `requirements.txt` (une liste de dépendances) pour un projet :

  ```
  pipenv lock --requirements
  ```

pwd

Affiche le nom du répertoire actuel.

- Affiche le répertoire actuel :

  ```
  pwd
  ```

- Affiche le répertoire actuel tout en traduisant les liens symboliques (c.-à-d. afficher le répertoire « physique ») :

  ```
  pwd -P
  ```

r

Interpréteur pour la langue R.

Plus d'informations : https://www.r-project.org.

- Démarre une session de commande R (REPL) :

  ```
  R
  ```

- Vérifie la version de R :

  ```
  R --version
  ```

- Exécute un fichier :

  ```
  R -f {{fichier.R}}
  ```

rsync

Transférer des fichiers vers ou depuis un hôte distant (pas entre deux hôtes distants).

Peut transférer un ou plusieurs fichiers correspondant à un motif.

- Transférer un fichier local vers un serveur distant :

```
rsync {{chemin/vers/fichier_local}} {{hote_distant}}:
{{chemin/vers/dossier_distant}}
```

- Transférer un fichier d'un serveur distant vers l'hôte local :

```
rsync {{hote_distant}}:{{chemin/vers/fichier_distant}}
{{chemin/vers/dossier_local}}
```

- Transférer un fichier sous forme d'[a]rchive (pour conserver les attributs) et compressé ([z]ippé), en mode [v]erbeux, lisible par l'[h]umain et afficher la [p]rogression du transfert :

```
rsync -azvhP {{chemin/vers/fichier_local}} {{hote_distant}}:
{{chemin/vers/dossier_distant}}
```

- Transférer un dossier et tous ses sous-dossiers d'un hôte distant vers l'hôte local :

```
rsync -r {{hote_distant}}:{{chemin/vers/dossier_distant}}
{{chemin/vers/dossier_local}}
```

- Transférer le contenu d'un dossier (mais pas le dossier lui-même) d'un hôte distant vers un hôte local :

```
rsync -r {{hote_distant}}:{{chemin/vers/dossier_distant}}/
{{chemin/vers/dossier_local}}
```

- Transférer un dossier [r]écursivement, dans une [a]rchive pour conserver les attributs, en résolvant les [l]iens symboliques, et ignorant les fichiers déjà transférés sa[u]f si plus récents :

```
rsync -rauL {{hote_distant}}:{{chemin/vers/fichier_distant}}
{{chemin/vers/dossier_local}}
```

- Transférer un fichier par SSH et effacer les fichiers de l'hôte local qui n'existent pas sur l'hôte distant :

```
rsync -e ssh --delete {{hote_distant}}:{{chemin/vers/
fichier_distant}} {{chemin/vers/fichier_local}}
```

- Transférer un fichier par SSH et afficher l'avancement global du transfert :

```
rsync -e ssh --info=progress2 {{hote_distant}}:{{chemin/vers/
fichier_distant}} {{chemin/vers/fichier_local}}
```

ssh

Secure Shell est un protocole utilisé pour se connecter de façon sécurisée à des systèmes distants.

On peut l'utiliser pour se connecter ou exécuter des commandes sur un serveur distant.

- Se connecter à un serveur distant :

```
ssh {{utilisateur}}@{{hote_distant}}
```

- Se connecter à un serveur distant en utilisant une identité spécifique (clé privée) :

```
ssh -i {{chemin/vers/fichier_clef}} {{utilisateur}}
@{{hote_distant}}
```

- Se connecter à un serveur distant en utilisant un port spécifique :

```
ssh {{utilisateur}}@{{hote_distant}} -p {{2222}}
```

- Exécuter une commande sur un serveur distant :

```
ssh {{hote_distant}} {{commande -avec -options}}
```

- Tunnel SSH : Transfert par port dynamique (le SOCKS proxy se trouve sur localhost:9999) :

```
ssh -D {{9999}} -C {{utilisateur}}@{{hote_distant}}
```

- Tunnel SSH : Transfère un port spécifique (localhost:9999 vers example.org:80) en désactivant l'allocation de pseudo-[t]ty et l'exécution de commandes distantes :

```
ssh -L {{9999}}:{{exemple.org}}:{{80}} -N -T {{utilisateur}}
@{{hote_distant}}
```

- Saut SSH : Se connecter sur un serveur distant à travers une machine de rebond (plusieurs machines de rebond peuvent être définies en les séparant par des virgules) :

```
ssh -J {{utilisateur}}@{{hote_de_rebond}} {{utilisateur}}
@{{hote_distant}}
```

- Transfert d'agent : Transfère les informations d'authentification vers la machine distante (voir `man ssh_config` pour les options disponibles) :

```
ssh -A {{utilisateur}}@{{hote_distant}}
```

tar

Utilitaire d'archivage.

Souvent combiné avec une méthode de compression, telle que gzip ou bzip.

Plus d'informations : https://www.gnu.org/software/tar.

- Créer une archive à partir de fichiers :

```
tar cf {{cible.tar}} {{fichier1 fichier2 fichier3}}
```

- Créer une archive gzip :

```
tar czf {{cible.tar.gz}} {{fichier1 fichier2 fichier3}}
```

- Extraie une archive (compressée) dans le dossier courant :

```
tar xf {{source.tar[.gz|.bz2|.xz]}}
```

- Extraie une archive dans un dossier cible :

```
tar xf {{source.tar}} -C {{dossier}}
```

- Créer une archive compressée, en utilisant le suffixe de l'archive pour déterminer le programme de compression :

```
tar caf {{cible.tar.xz}} {{fichier1 fichier2 fichier3}}
```

- Lister le contenu d'une archive tar :

```
tar tvf {{source.tar}}
```

- Extraire les fichiers correspondant au motif :

```
tar xf {{source.tar}} --wildcards "{{*.html}}"
```

tmux

Multiplexeur de terminaux. Permet plusieurs sessions avec fenêtres, panneaux, et plus encore.

Plus d'informations : https://github.com/tmux/tmux.

- Démarrer une nouvelle session :

```
tmux
```

- Démarrer une nouvelle session nommée :

```
tmux new-session -s {{nom}}
```

- Lister les sessions existantes :

```
tmux ls
```

- S'attacher à la session utilisée la plus récemment :

```
tmux attach-session
```

- S'attacher à une session nommée :

```
tmux attach-session -t {{nom}}
```

- Se détacher de la session actuelle (avec le préfixe Ctrl-B) :

```
Ctrl-B d
```

- Détruire une session nommée :

```
tmux kill-session -t {{nom}}
```

- Détruire la session actuelle (avec le préfixe Ctrl-B) :

```
Ctrl-B :kill-session<Enter>
```

tput

Accède et modifie les paramètres du terminal.

- Déplace le curseur à un endroit donné sur l'écran :

  ```
  tput cup {{coordonnée_y}} {{coordonnée_x}}
  ```

- Règle la couleur de l'avant-plan (af) ou de l'arrière-plan (ab) :

  ```
  tput {{setaf|setab}} {{code_de_couleur_ANSI}}
  ```

- Affiche le numéro de colonnes, de rangées, ou de couleurs :

  ```
  tput {{cols|lines|colors}}
  ```

- Active la sonnerie du terminal :

  ```
  tput bel
  ```

- Réinitialise tous les attributs du terminal :

  ```
  tput sgr0
  ```

- Active ou désactive le retour automatique à la ligne :

  ```
  tput {{smam|rmam}}
  ```

vim

Vim (Vi IMproved), un editeur de texte en ligne de commandes, fournit plusieurs modes pour differentes manipulations de texte.

Pressez **`i`** pour passer en mode édition. **`<Esc>`** revient au mode normal, qui ne permet pas l insertion de code.

Plus d'informations : https://www.vim.org.

- Ouvrir un fichier :

  ```
  vim {{chemin/vers/fichier}}
  ```

- consulter le manuel utilisateur :

  ```
  :help<Enter>
  ```

- Sauvegarder et fermer :

  ```
  :wq<Enter>
  ```

- Ouvrir un fichier a une ligne specifiée :

  ```
  vim +{{numero_ligne}} {{chemin/vers/fichier}}
  ```

- Annuler la derniere operation :

  ```
  u
  ```

- Rechercher un pattern dans un fichier (appuyez `n`/`N` pour aller a la prochaine/ précedante occurence) :

  ```
  /{{pattern_recherche}}<Enter>
  ```

- Effectuer une substition par éxpression reguliere dans tout le fichier :

  ```
  :%s/{{pattern}}/{{replacement}}/g<Enter>
  ```

- Afficher les numeros de ligne :

  ```
  :set nu<Enter>
  ```

WC

Compte les lignes, les mots ou les octets.

- Compte les lignes d'un fichier :

```
wc -l {{file}}
```

- Compte les mots d'un fichier :

```
wc -w {{file}}
```

- Compte les caractères (octets) d'un fichier :

```
wc -c {{file}}
```

- Compte les caractères d'un fichier (en prenant en compte l'ensemble des caractères multi-octets) :

```
wc -m {{file}}
```

which

Localise un programme dans le chemin de l'utilisateur.

- Fouille la variable d'environnement « PATH » et affiche l'endroit des programmes exécutables correspondantes à la requete :

  ```
  which {{exécutable}}
  ```

- Affiche toutes les exécutables correspondantes à la requete, si il y en a plus qu'un :

  ```
  which -a {{exécutable}}
  ```

z

Recherche les répertoires les plus utilisés et permet une navigation rapide à l'aide de chaine de caractères ou de regex.

Plus d'informations : https://github.com/rupa/z.

- Aller dans un répertoire qui contient "foo" dans son nom :

```
z {{foo}}
```

- Aller dans un répertoire qui contient "foo" et "bar' dans son nom :

```
z {{foo}} {{bar}}
```

- Aller dans le répertoire le mieux classé parmi ceux qui contiennent "foo" dans leurs noms :

```
z -r {{foo}}
```

- Aller dans le répertoire accédé le plus récemment parmi ceux qui contiennent "foo" dans leurs noms :

```
z -t {{foo}}
```

- Lis l'ensemble des répertoires dans la base de données `z` qui contiennent "foo" dans leurs noms :

```
z -l {{foo}}
```

- Supprime le répertoire courant de la base de données de `z` :

```
z -x .
```

zip

Package et compresse (archive) les fichiers en un fichier zip.

- Package et compresse [r]écursivement un répertoire et son contenu :

```
zip -r {{archive.zip}} {{chemin/du/répertoire}}
```

- E[x]clure des fichiers de l'archive :

```
zip -r {{archive.zip}} {{chemin/vers/le/répertoire}} -x
{{chemin/des/fichiers/exclus}}
```

- Archive un répertoire et son contenu avec le plus haut niveau [9] de compression :

```
zip -r -{{9}} {{archive.zip}} {{chemin/du/répertoire}}
```

- Package et compresse plusieurs répertoires et fichiers :

```
zip -r {{archive.zip}} {{chemin/du/répertoire1 chemin/du/
répertoire2 chemin/du/fichier}}
```

- Créé une archive chifrée (l'utilisateur sera sollicité pour saisir le mot de passe) :

```
zip -e -r {{archive.zip}} {{chemin/du/répertoire}}
```

- Ajoute des fichiers à une archive existante :

```
zip {{archive.zip}} {{chemin/du/fichier}}
```

- Supprime des fichiers d'une archive existante :

```
zip -d {{archive.zip}} "{{foo/*.tmp}}"
```

- Archive un répertoire et son contenu en plusieurs fichiers zip [s]cindés (ex : des fichiers de 3 Go) :

```
zip -r -s {{3g}} {{archive.zip}} {{chemin/du/répertoire}}
```

zless

Lire des fichiers compréssés.

- Parcourrir une archive compréssé avec `less` :

  ```
  zless {{fichier.txt.gz}}
  ```

zola

Un générateur de site statique à partir d'un unique binaire sans dépendances.

Plus d'informations : https://www.getzola.org/documentation/getting-started/cli-usage/.

- Créé la structure du répertoire utilisé par Zola dans un répertoire donné :

  ```
  zola init {{mon_site}}
  ```

- Cronstuit la totalité du site dans le répertoire `public` (si le répertoire existe, il est supprimé) :

  ```
  zola build
  ```

- Construit la totalité du site dans un répertoire différent :

  ```
  zola build --output-dir {{chemin/du/répertoire_de_sortie/}}
  ```

- Contruit et met à disposition le site à partir d'un serveur local (l'adresse par défaut est `127.0.0.1:1111`) :

  ```
  zola serve
  ```

- Construit l'ensemble des pages comme la commande build, sans écrire le résultat sur le disque :

  ```
  zola check
  ```

zopflipng

Utilitaire de compression d'images PNG.

Plus d'informations : https://github.com/google/zopfli.

- Optimise une imahe PNG :

```
zopflipng {{entrée.png}} {{sortie.png}}
```

- Optimise plusieurs images PNG et sauvegarde avec prefix donné :

```
zopflipng --prefix={{prefix}} {{image1.png}} {{image2.png}}
{{image3.png}}
```

zoxide

Garde une trace des répertoires les plus utilisés.

Utilise un algorithme de classement pour identifier le meilleur résultat.

Plus d'informations : https://github.com/ajeetdsouza/zoxide.

- Aller au répertoire avec le meilleur classement qui contient "foo" dans son nom :

```
zoxide query {{foo}}
```

- Aller au répertoire avec le meilleur classement qui contient "foo" et "bar" dans son nom :

```
zoxide query {{foo}} {{bar}}
```

- Démarre une recherche de répertoire interactive (requires `fzf`) :

```
zoxide query --interactive
```

- Ajoute un répertoire ou incrémente son classement :

```
zoxide add {{chemin/du/répertoire}}
```

- Supprime un répertoire de la base de donnée de `zoxide` :

```
zoxide remove {{chemin/du/répertoire}}
```

- Génére la configuration du shell pour la mise en place des alias de commandes (`z`, `za`, `zi`, `zq`, `zr`) :

```
zoxide init {{bash|fish|zsh}}
```

Linux

apt-add-repository

Gère la définition des dépôts apt.

- Ajout d'un nouveau dépôt :

  ```
  apt-add-repository {{repository_spec}}
  ```

- Suppression d'un dépôt :

  ```
  apt-add-repository --remove {{repository_spec}}
  ```

- Mise à jour du cache des paquets après ajout d'un dépôt :

  ```
  apt-add-repository --update {{repository_spec}}
  ```

- Activation pour les paquets source :

  ```
  apt-add-repository --enable-source {{repository_spec}}
  ```

apt-cache

Outil de recherche de paquets Debian et Ubuntu.

- Recherche un paquet dans vos sources actuelles :

  ```
  apt-cache search {{query}}
  ```

- Affiche des informations sur un paquet :

  ```
  apt-cache show {{package}}
  ```

- Indique si un paquet est installé et à jour :

  ```
  apt-cache policy {{package}}
  ```

- Affiche les dépendances d'un paquet :

  ```
  apt-cache depends {{package}}
  ```

- Affiche les paquets qui dépendent d'un paquet particulier :

  ```
  apt-cache rdepends {{package}}
  ```

apt-file

Recherche de fichiers dans les paquets apt, y compris ceux qui ne sont pas encore installés.

- Mise à jour la base de données des métadonnées :

  ```
  sudo apt update
  ```

- Recherche des paquets qui contiennent le fichier ou le chemin d'accès spécifié :

  ```
  apt-file search {{part/of/filename}}
  ```

- Énumère le contenu d'un paquet spécifique :

  ```
  apt-file list {{package_name}}
  ```

apt-get

Utilitaire de gestion des paquets Debian et Ubuntu.

Recherche des paquets en utilisant **`apt-cache`**.

- Mise à jour de la liste des paquets et des versions disponibles (il est recommandé de l'exécuter avant les autres commandes `apt-get`) :

  ```
  apt-get update
  ```

- Installation d'un paquet, ou mise à jour avec la dernière version disponible :

  ```
  apt-get install {{package}}
  ```

- Suppression d'un paquet :

  ```
  apt-get remove {{package}}
  ```

- Suppression d'un paquet et de ses fichiers de configuration :

  ```
  apt-get purge {{package}}
  ```

- Mise à jour de tous les paquets installés vers les dernières versions disponibles :

  ```
  apt-get upgrade
  ```

- Nettoyage du dépôt local - supprime les fichiers de paquets (`.deb`) des téléchargements interrompus qui ne peuvent plus être téléchargés:

  ```
  apt-get autoclean
  ```

- Suppression de tous les paquets qui ne sont plus nécessaires :

  ```
  apt-get autoremove
  ```

- Mise à jour des paquets installés (comme la commande `upgrade`), mais avec suppression des paquets obsolètes et installation des paquets supplémentaires pour répondre aux nouvelles dépendances :

  ```
  apt-get dist-upgrade
  ```

apt-mark

Utilitaire permettant de modifier l'état des paquets installés.

- Marquer un paquet comme étant automatiquement installé :

  ```
  sudo apt-mark auto {{package_name}}
  ```

- Maintenir un paquet à sa version actuelle et empêcher les mises à jour :

  ```
  sudo apt-mark hold {{package_name}}
  ```

- Permettre une nouvelle mise à jour d'un paquet :

  ```
  sudo apt-mark unhold {{package_name}}
  ```

- Afficher les paquets installés manuellement :

  ```
  apt-mark showmanual
  ```

- Afficher les paquets détenus qui ne sont pas mis à jour :

  ```
  apt-mark showhold
  ```

apt

Utilitaire de gestion des paquets pour les distributions basées sur Debian.

Remplacement recommandé pour apt-get lorsqu'il est utilisé de manière interactive dans les versions 16.04 et ultérieures d'Ubuntu.

- Mettre à jour la liste des paquets et des versions disponibles (il est recommandé de l'exécuter avant les autres commandes `apt`) :

  ```
  sudo apt update
  ```

- Recherche d'un paquet donné :

  ```
  apt search {{package}}
  ```

- Afficher les informations pour un paquet :

  ```
  apt show {{package}}
  ```

- Installer un paquet, ou le mettre à jour avec la dernière version disponible :

  ```
  sudo apt install {{package}}
  ```

- Supprimer un paquet (utiliser `purge` à la place supprime également ses fichiers de configuration) :

  ```
  sudo apt remove {{package}}
  ```

- Mettre à jour tous les paquets installés vers les dernières versions disponibles :

  ```
  sudo apt upgrade
  ```

- Lister tous les paquets :

  ```
  apt list
  ```

- Lister les paquets installés :

  ```
  apt list --installed
  ```

flameshot

Outil de capture d'écran avec une interface graphique.

Ajoute du texte, des formes, des couleurs et envoie à imgur.

Plus d'informations : https://flameshot.js.org.

- Lancez Flameshot en mode graphique :

```
flameshot launcher
```

- Prenez une capture d'écran en cliquant et en faisant glisser :

```
flameshot gui
```

- Prenez une capture d'écran en plein écran :

```
flameshot full
```

- Définissez le chemin de sauvegarde :

```
flameshot full --path {{path/to/directory}}
```

- Retardez la capture d'écran de N millisecondes et la sortie dans le presse-papiers :

```
flameshot full --delay {{2000}} --clipboard
```

iwctl

Un outil de ligne de commande pour gèrer iwd.

Plus d'informations : https://iwd.wiki.kernel.org/gettingstarted.

- Lancer le mode interactif, dans ce mode vous pouvez entrer les commandes directement, avec de l'autocompletion :

  ```
  iwctl
  ```

- Avoir l'aide générale :

  ```
  iwctl --help
  ```

- Afficher vos stations wifi :

  ```
  iwctl station list
  ```

- Lancer la recherche de réseaux avec une station :

  ```
  iwctl station {{station}} scan
  ```

- Afficher les réseaux trouvés par une station :

  ```
  iwctl station {{station}} get-networks
  ```

- Se connecter à un réseau avec une station, si des informations de connexion sont nécessaires elles seront démandées :

  ```
  iwctl station {{station}} connect {{network_name}}
  ```

pacman

Outil de gestion de paquets sur Arch Linux.

Plus d'informations : https://man.archlinux.org/man/pacman.8.

- Synchronise et mets à jour tous les paquets :

```
pacman -Syu
```

- Installe un nouveau paquet :

```
pacman -S {{nom_paquet}}
```

- Efface un paquet et ses dépendances :

```
pacman -Rs {{nom_paquet}}
```

- Recherche dans la base de données des paquets une expression régulière ou mot clé :

```
pacman -Ss "{{terme_recherche}}"
```

- Liste les paquets installés et leurs versions :

```
pacman -Q
```

- Liste seulement les paquets installés explicitement et leurs versions :

```
pacman -Qe
```

- Trouve à quel paquet un certain fichier appartient :

```
pacman -Qo {{fichier}}
```

- Vide le cache des paquets pour libérer de l'espace :

```
pacman -Scc
```

useradd

Crée un nouvel utilisateur.

- Crée un nouvel utilisateur :

  ```
  useradd {{nom}}
  ```

- Crée un nouvel utilisateur avec un dossier home par défaut :

  ```
  useradd --create-home {{nom}}
  ```

- Crée un nouvel utilisateur avec le shell spécifié :

  ```
  useradd --shell {{/chemin/vers/shell}} {{nom}}
  ```

- Crée un nouvel utilisateur qui appartient aux groupes supplémentaires (attention à l'omission des espaces) :

  ```
  useradd --groups {{groupe1,groupe2}} {{nom}}
  ```

- Crée un nouvel utilisateur sans un dossier home :

  ```
  useradd --no-create-home --system {{nom}}
  ```

userdel

Efface un utilisateur.

- Efface un utilisateur et son dossier home :

```
userdel -r {{nom}}
```

usermod

Modifie un compte utilisateur.

- Change le nom d'un utilisateur :

  ```
  usermod -l {{nouveau_nom}} {{utilisateur}}
  ```

- Ajoute l'utilisateur à des groupes supplementaires (attention à l'omission d'espaces) :

  ```
  usermod -a -G {{groupe1,groupe2}} {{utilisateur}}
  ```

- Crée un nouveau dossier home pour un utilisateur et déplace ses fichiers vers celui-ci :

  ```
  usermod -m -d {{/chemin/vers/home}} {{utilisateur}}
  ```

xbacklight

Outil pour ajuster la luminosité du rétroéclairage en utilisant l'extension RandR.

Plus d'informations : https://gitlab.freedesktop.org/xorg/app/xbacklight.

- Obtient le niveau de luminosité de l'écran actuel comme un pourcentage :

  ```
  xbacklight
  ```

- Régle la luminosité de l'écran à 40% :

  ```
  xbacklight -set {{40}}
  ```

- Augmente la luminosité de l'écran actuel de 25% :

  ```
  xbacklight -inc {{25}}
  ```

- Diminue la luminosité de l'écran actuel de 75% :

  ```
  xbacklight -dec {{75}}
  ```

- Augment la luminosité de l'écran à 100%, étendu sur 60 secondes (valeur donnée en ms), en 60 pas :

  ```
  xbacklight -set {{100}} -time {{60000}} -steps {{60}}
  ```

xclip

Outil de manipulation de presse-papiers X11, semblable à **`xsel`**.

Gère les selections primaires et secondaires de X, en plus du presse-papier système (**`Ctrl + C`**/**`Ctrl + V`**).

- Copie la sortie d'une commande vers la zone de sélection primaire de X11 (presse-papiers) :

```
echo 123 | xclip
```

- Copie la sortie d'une commande vers une zone de sélection de X11 donnée :

```
echo 123 | xclip -selection {{primary|secondary|clipboard}}
```

- Copie le contenu d'un fichier vers le presse-papiers système, avec la notation courte :

```
echo 123 | xclip -sel clip
```

- Copie le contenu d'un fichier vers le presse-papiers système :

```
xclip -sel clip {{fichier_entrée.txt}}
```

- Copie le contenu d'une image PNG vers le presse-papiers système (peut être collé dans d'autres programmes correctement) :

```
xclip -sel clip -t image/png {{fichier_entrée.png}}
```

- Colle le contenu de la zone de selection de X11 à la console :

```
xclip -o
```

- Colle le contenu du presse-papier système à la console :

```
xclip -o -sel clip
```

- Colle le contenu du presse-papier système à un fichier :

```
xclip -o -sel clip > {{fichier_sortie.txt}}
```

yay

Yet Another Yogurt : Un outil pour Arch Linux pour construire et installer des paquets depuis le Arch User Repository.

À regarder : **pacman**.

- Recherche interactivement et installe des paquets depuis les dépôts et l'AUR :

  ```
  yay {{nom_paquet|terme_recherche}}
  ```

- Synchronise et mets à jour tous les paquets depuis les dépôts et l'AUR :

  ```
  yay
  ```

- Synchronise et mets à jour seulement les paquets de l'AUR :

  ```
  yay -Sua
  ```

- Installe un nouveu paquet depuis les dépôts et l'AUR :

  ```
  yay -S {{nom_paquet}}
  ```

- Recherche dans la base de données de paquets un mot clé depuis les dépôts et l'AUR :

  ```
  yay -Ss {{mot_clé}}
  ```

- Montre des statistiques sur les paquets installés et la santé du système :

  ```
  yay -Ps
  ```

yes

Envoie un message à répétition en sortie console.

Cette commande est souvent utilisée pour éviter de devoir accepter des opérations successives (par exemple des installations via la commande **`apt-get`**).

- Envoyer « message » à répétition :

  ```
  yes {{message}}
  ```

- Envoyer « y » à répétition :

  ```
  yes
  ```

- Répondre « oui » à toutes les questions posées par la commande `apt-get` :

  ```
  yes | sudo apt-get install {{program}}
  ```