SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

git

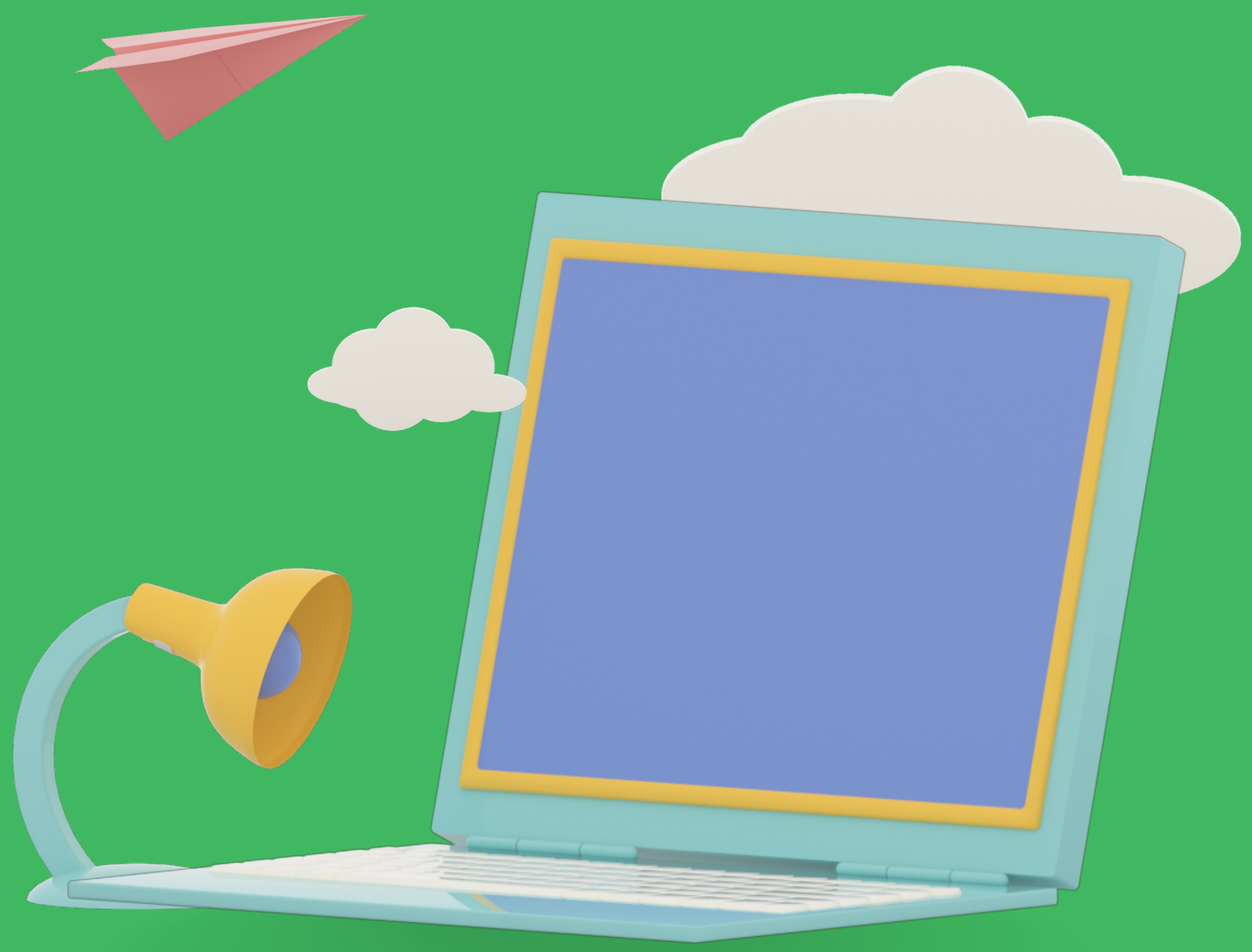
GIT

Gestion de versions décentralisée

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

Cahier Git

points à retenir



Bases de git

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

Pourquoi git :

- sauver le travail courant, revenir en arrière, tester du code, garder des archives, permettre de nettoyer son code,
- faire des versions pour les mises en prod, faciliter la maintenance, les changelogs, les suivis de correctifs,
- travailler à plusieurs sur un même projet, potentiellement sur des mêmes lignes de codes, avec des outils de fusion,
- grâce aux branches, travailler sur plusieurs versions d'un même projet : par fonctionnalité, de prod et de correctifs,
- rapidité des commandes et décentralisation.

Limitations :

- beaucoup de commandes, parfois complexes,
- quelques paramètres à gérer attentivement,
- solutions multiples pour une même procédure.

Origines :

- créé pour pour gérer les sources du noyau Linux,
- optimisé pour l'usage hors ligne, la décentralisation, la duplication, les merges, les performances.

install avec ubuntu server

- sudo apt-get install git
- git
  - => pas d'erreur
- Voir :

https://git-scm.com/download/linux
ou
https://git-scm.com/download

quelques bonnes pratiques

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

Organisation :

- Par équipe définir le fonctionnement en amont,
- Chaque intervenant de l'équipe utilise le même formalisme (contenu, commentaires),
- Utiliser des étiquettes (tag) pour identifier un lot de code spécifique,
- Livrer un fichier texte qui décrie le contenu et le contexte : release-note,
- Avoir des identifiants dans le workflow : notamment indiquer les références de la ou des fiches fonctionnelles dans le système de gestion de projet,

Nom de version : x.y.z

- x = majeur : versions importantes du projet, souvent avec une cassure entre les versions lorsque x change (usage, migration, nouvelle API, ...),
- y = mineur : versions courantes du projet, lors de livraison régulières avec ajout de fonctionnalités ou corrections, pas de changement majeurs d'usage ou d'interconnexion,
- z = micro : versions de maintenance avec des corrections mais sans nouveauté fonctionnelle.

éléments git

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

Architecture :

- décentralisé : un serveur correspond à un dépôt, plusieurs serveurs peuvent être associé à un seul projet,
- la connexion avec le dépôt est ponctuelle. Localement image de travail et copie du serveur.

Lexique et commandes :

- commit : lot de modifications,
- log : historique des évènements git (tout est tracé, localement et sur le serveur),
- repo/repository : local ou distant (sur le serveur) c'est le projet vu par git,
- remote : repo distant sur le serveur,
- local : repo local sur le poste de travail,
- branch : version spécifique et isolée des fichiers, regroupe des commits,

Lexique et commandes :

- merge : fusionner différents commits ou branches, si un même fichier modifié => conflit (géré ou non automatiquement),
- push : envoyer les éléments du repo local vers le repo remote,
- fetch : lire les changements sur le repo remote, sans modifier les fichiers locaux,
- pull : mettre à jour les fichiers locaux avec les changements du repo remote,
- master : branche de base du repo,
- cherry-pick : copie d'un élément,
- rebase : déplacer le point de rattachement de branche ou de commit,
- rebase interactif : réécriture des commits d'une branche (ordre, contenu, commentaires, ...).

Git commandes de base

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

- **clone** : copie d'un repo distant,
- **fetch** : commande de fetch : lire le remote,
- **fetch --prune** : commande de fetch et supprime les branches enlevées du remote,
- **pull** : commande de pull : récupérer depuis le remote,
- **push** : commande de push : envoyer vers le remote,
- **push origin :branch-to-remove** : supprime sur le repo distant la branche par son nom,
- **add** : sélection des éléments du prochain commit,
- **commit** : crée le commit depuis les éléments sélectionnés
- **commit -m 'message'** : ajoute un commentaire au commit,
- **commit --amend -m 'message'** : met à jour le dernier commit (et ajout si combiné avec add) et change le commentaire,
- **commit --amend --no-edit** : met à jour le dernier commit et conserve son commentaire,
- **status** : état du repo local et des fichiers modifiés,
- **diff** : affiche les changements courants du repo local,
- **tag -a v1.0.0** : crée une étiquette,
- **init** : création d'un repo,
- **commit --allow-empty -m 'Initial empty commit' :** création du commit de démarrage d'un repo,
- **remote add origin** : ajout d'un repo distant (RTFM pour les options)
- **branch -d feature-XXX** : supprimer une branche par son nom si possible,
- **branch -D feature-XXX** : supprime en forçant la branche,
- **reset --hard ZZZ** : attention perte de données, tout annuler et repartir de l'élément ZZZ (id de commit ou branche en général),

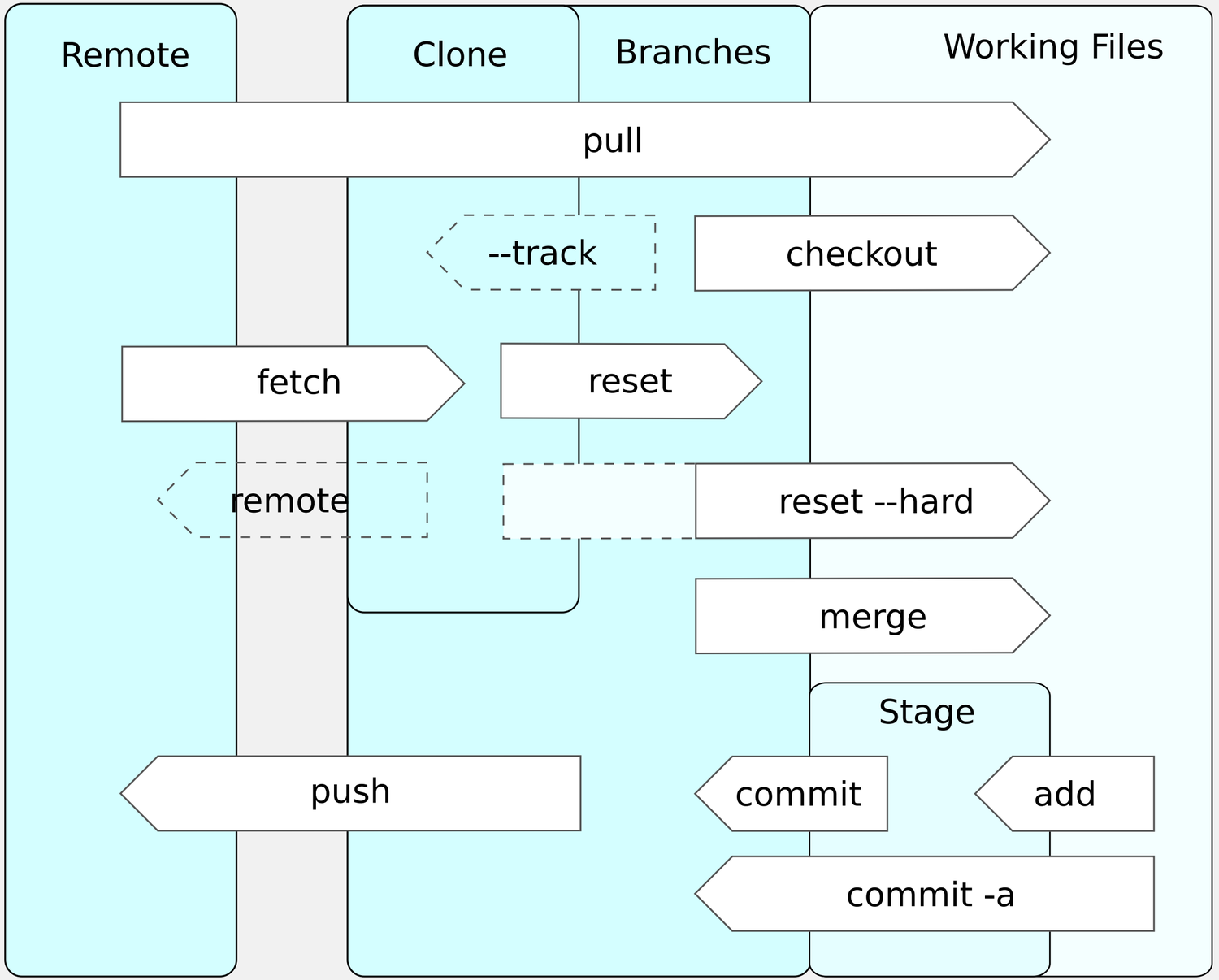
Git commandes de base

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

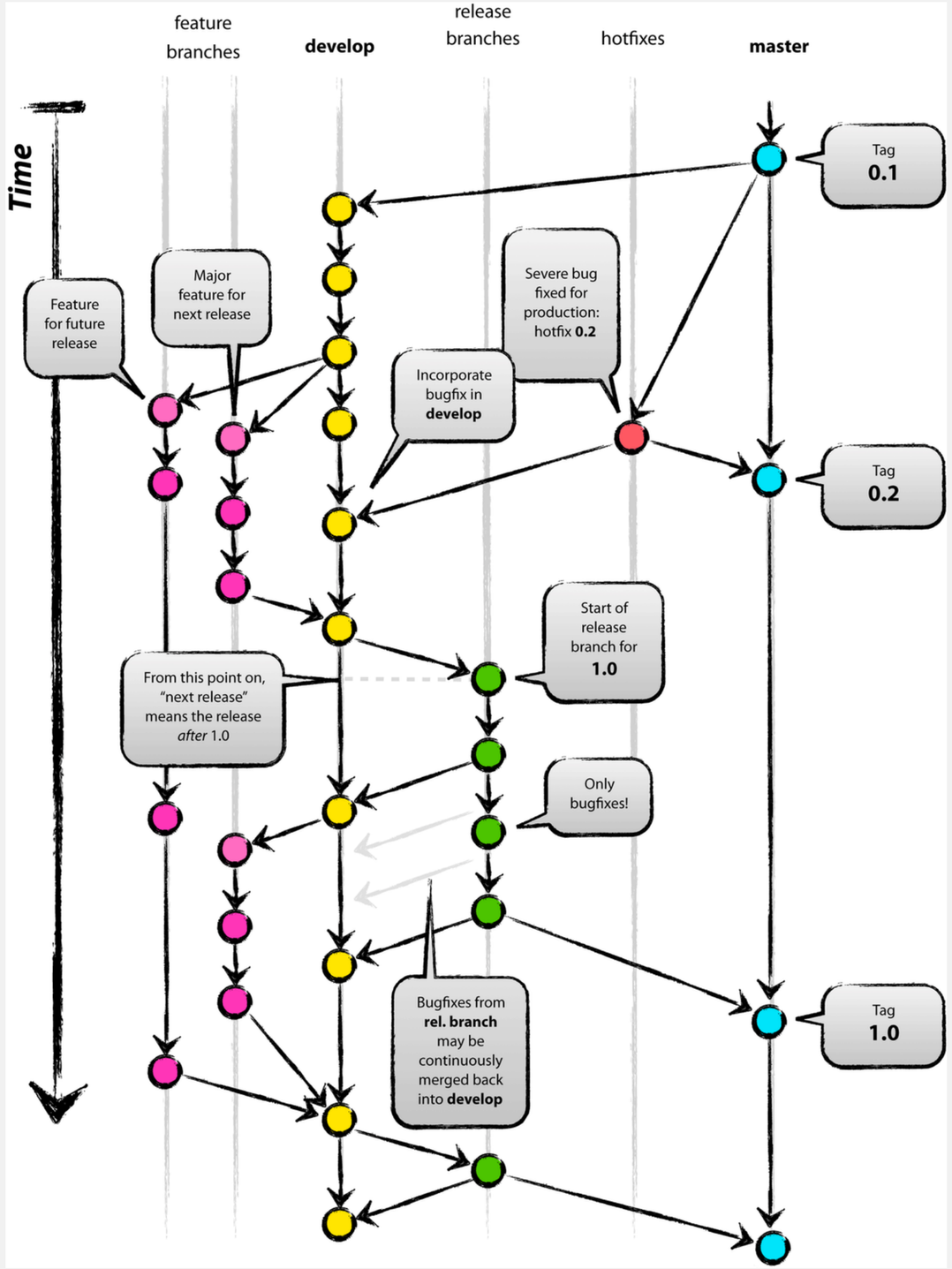
- **checkout master :** bascule sur la branche master,
- **checkout -b release-1.0** : crée la branche de release pour la version 1.0, d'après le commit courant et y va,
- **log** : affiche l'historique courant (branches, commits)
- **show $SHA** : affiche les détails de l'élément
- **stash** : met de côté des modifications locales que l'on ne souhaite pas associer à un commit,
  - **stash list** : affiche les éléments stockés,
  - **stash pop** : sort les derniers éléments stockés,
  - **stash clear** : efface les éléments
- **rebase -i develop** : déplace la branche courante sur le dernier commit de 'develop', et traite interactivement les commits de la branch courante (à voir par l'exemple),
- **merge dev-feature-1** : fusionne la branche nommée avec la branche courante (ici devrait être develop),
- **merge --no-ff release-1.0** : fusionne la branche nommée avec la branche courante (ici devrait être master),
- **reflog :** affiche les éléments internes git, retrace tout l'historique local, et permet de revenir à un état perdu,
- **alias** : définir des raccourcis, comme :
  - ci = commit,
  - cp = cherry-pick,
  - rb = rebase,
  - co = checkout,
  - st = status,
  - br = branch,
  - lg = log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(bold yellow)%d%C(reset)' --all

Opérations git

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat



Workflow git



SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

config git

Config générale :

- more ~/.gitconfig
  - [user]
    - name = Sebastien GRIVOLAT
    - email = email@example.com
    - username = sgr-fr
  - [alias]
    - cp = cherry-pick
  - …
- more ~/.gitignore_global
  - .idea/
  - ansible/
  - !ansible/.gitkeep
  - uploads
  - …

Config du projet (= du repo) :

- more .git/config
  - [core]
    - repositoryformatversion = 0
    - logallrefupdates = true
  - [remote "origin"]
    - url = git@github.com:sgr-fr/XXX.git
    - fetch = +refs/heads/*:refs/remotes/origin/*
  - [branch "master"]
    - remote = origin
    - merge = refs/heads/master
    - rebase = true
  - …
- more .gitignore
  - build
  - …

bonnes pratiques / usage avancé

- **commit atomique** :
  - un seul sujet,
  - un périmètre technique restreint,
  - s'appuyer sur bon découpage des fiches fonctionnelles,
  - si plusieurs lib impactées = plusieurs commits.
- **faire de la relecture** :
  - validation croisée des livrables,
  - règles de codage (algorithmie),
  - règles d'écriture (syntaxe).
- **bien découper** :
  - 1 fiche fonctionnelle (issue) = 1 merge request (branch de feature) = 1 à n commits
- **avoir une checklist de merge** :
  - relire le contenu,
  - lancer tests fonctionnels,
  - vérifier les titres,
  - vérifier les commentaires,
  - vérifier les traductions,
  - amender le release notes,
  - supprimer le code mort,
  - vérifier la mise à jour de la doc,
  - lancer les outils automatiques (cs-fixer, testsunit, ...),
  - rebaser la branche sur la tête de develop,
  - créer une merge request dans l'outil d'intégration (gitlab, github, ...)
  - ...

bonnes pratiques / usage avancé

- **utiliser des textes percutants** :
  - concis et précis : titres (nom de branch),
  - commentaires de commit,
  - release notes, …
  - Mettre en place des règles de nommage.
  - Horodater lorsque c'est utile.
- **exemple de règle de titre** :
  - débute par un verbe à l'impératif,
  - en anglais,
  - max 50 caractères,
  - quelques mots précis en plus du verbe
- tailles :
  - limiter le stockage dans le repo
  - que des sources, pas de binaires
  - plus facile d'ajouter que de supprimer des ressources lourdes
- **exemple de règle de message** :
  - laisser une ligne vide sous le titre,
  - donner des détails en complément du titre,
  - couper les lignes à 72 caractères,
  - lister et tracer les références ou éléments externes éventuels,
  - mettre à la fin les références des tickets concernés dans l'outil de suivi ou de gestion de fiches.

serveur git pour le repo remote

- sur un serveur minimaliste :
  - accès ssh/sftp
  - répertoire local (au serveur)
    - mkdir my-repo
    - cd my-repo
    - git --bare init
  - sur le client (poste de dev)
    - git remote add origin ssh://USER@SERVER:22/DIRECTORY

- en saas :
  - chez un fournisseur
  - auto-hebergé
  - ...

- avec l'outil : gitea
  - https://docs.gitea.io/fr-fr/
- avec l'outil de gestion de projet tuleap :
  - https://docs.tuleap.org/user-guide/intro.html

- dans le cloud sur github
  - https://github.com/
- dans le cloud sur gitlab :
  - https://gitlab.com/

tips linux

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

- **Ctrl-R**
  - recherche de commande en arrière dans l'historique
- **history**
  - afficher toutes les commandes déjà exécutées
- **wget localhost:XXX / curl localhost:XXX**
  - pour vérifier si mon serveur web est actif, fonctionne avec une url
- **cat > MON_FICHIER.TXT**
  - écrire un fichier, fin avec Ctrl-D
- **nano MON_FICHIER.TXT ou vi MON_FICHIER.TXT**
  - éditer rapidement mon fichier sans éditeur type IDE disponible
- **ssh -p XX user@machine**
  - se connecter à distance ou à une VM depuis l'espace de travail habituel (yes pour accepter le certificat)
  - permet aussi de partager des fichiers depuis une machine distante (par exemple avec filezilla ou vs codium)
- **pwd**
  - connaître le répertoire courant
- **sudo netstat -tulpa**
  - voir les connexions réseau et les ports ouverts
- **lsof -p 42**
  - voir les fichiers ouverts par le processus ID 42
- **df :**
  - place disponible sur les partitions
  - df -h
- **du :**
  - usage disque pour une ressource
  - du -h
- **sort :**
  - trier l'entrée
  - du -h | sort -h
- **wc :**
  - compter des mots
  - ls | wc -l

tips linux

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

- **grep XXX ou grep -v ZZZ**
  - chercher XXX
  - chercher tout sauf ZZZ
  - sudo netstat -tulpa | grep http | grep -v firefox
- **git commit --help**
  - avoir l'aide en ligne sur une commande
- **sh ou bash** dans un conteneur (via exec) :
  - **ping** : vers les autres conteneurs, avec leur nom
  - **ip a** : adresse du conteneur
  - **ps aux** : lister les processus en cours dans le conteneur
  - **netstat -tulpa** : ports exposés par le conteneur
  - **/var/log ou ailleurs** : accéder à des logs internes non exposés (par volume)
  - **changements internes** de conf ou de contenu : attention aux volumes en read-only, les modifs sont perdues au redémarrage du conteneur, donc pour tester en live uniquement
- **sudo dmesg** :
  - affiche les traces de la machine linux
- **reboot**
  - redémarre la machine
  - sudo reboot
  - (dans le doute…)
- **shutdown**
  - arrête la machine
  - à éviter à distance sans accès physique
  - sudo shutdown -h now
- **rmdir FOLDER**
  - supprimer un répertoire vide
- **rm -rf FOLDER**
  - supprimer sans confirmation un répertoire et tout son contenu
  - ATTENTION IRRÉVERSIBLE RISQUE DE PERTE DE DONNÉES

Ressources

- Doc officielle :
  - https://git-scm.com/book/fr/v2
- L'exemple de workflow présenté en détail :
  - français : https://www.occitech.dev/posts/2014/12/un-modele-de-branches-git-efficace/
  - original : http://nvie.com/posts/a-successful-git-branching-model/
- Des règles pour écrire les messages et titres de commit :
  - http://chris.beams.io/posts/git-commit/
- Des ressources diverses :
  - http://rogerdudler.github.io/git-guide/index.fr.html
  - https://www.atlassian.com/fr/git/tutorials
  - https://www.w3schools.com/git/default.asp?remote=github

Merci