

# Compte rendu outils libres côté client

Gautier Delpierre, Bastien Grosclaude

13 février 2022



# Table des matières

<b>1</b>	<b>Efficacité de l'environnement de travail</b>	<b>3</b>
1.1	Question 1 . . . . .	3
1.2	Question 2 . . . . .	3
1.3	Question 3 . . . . .	4
1.4	Question 4 . . . . .	5
1.5	Question 5 . . . . .	5
1.6	Question 6 . . . . .	6
1.7	Question 7 . . . . .	6
1.8	Question 8 . . . . .	7
1.9	Question 9 . . . . .	7
<b>2</b>	<b>SSH</b>	<b>8</b>
2.1	Question 1 . . . . .	8
2.2	Question 2 . . . . .	8
2.3	Question 3 . . . . .	8
2.4	Question 4 . . . . .	9
2.5	Question 5 . . . . .	9
2.6	Question 6 . . . . .	10
2.7	Question 7 . . . . .	10
<b>3</b>	<b>GIT</b>	<b>12</b>
3.1	Question 1 . . . . .	12
3.2	Question 2 . . . . .	12
3.3	Question 3 . . . . .	13

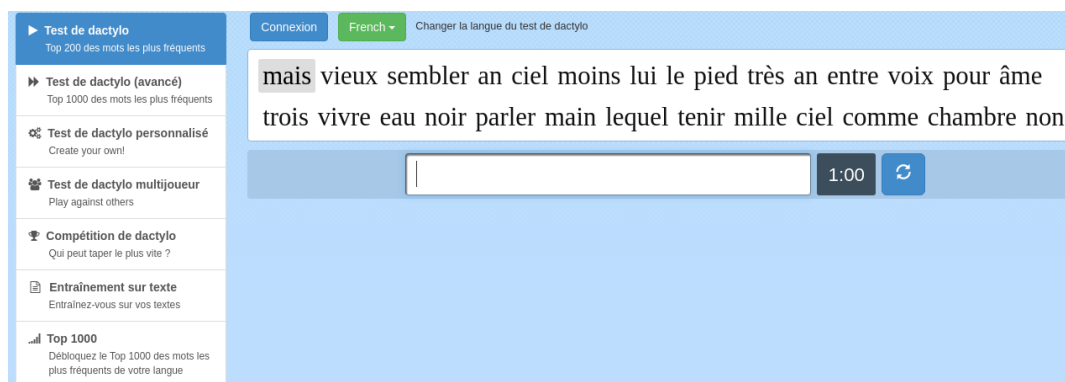
# 1 Efficacité de l'environnement de travail

## 1.1 Question 1

Problème	Raccourcis
page précédente	Alt + ->
page suivante	Alt + <-
Nouvel onglet	Ctrl + t
Fermer un onglet	Ctrl + w
Naviguer entre les onglets	Ctrl + tab
Reouvrir le dernier onglet fermer	Ctrl + maj + t
Rafraîchir	F5 ou ctrl + r
Renommer un fichier ou un dossier	F2
Naviguer a travers les fenêtres	Alt + tab
Naviguer entre les bureaux	Super + tab
Ferme terminale, bash, ssh, zsh, vagrant	Ctrl + d
Tty	Ctrl+alt+f1...f8
Ouvrir le menu	super
Menu de fenetre	Alt + espace

## 1.2 Question 2

Nous avons le site 10fastfingers.com pour s'améliorer à taper clavier. Nous avons choisis celui-la car nous pouvons faire des courses de rapidité entre nous.



Voici le résultat après quelque entraînement.

Résultat		Capture d'écran
<div>52 MPM</div> <div>(mots par minute)</div>		
Frappes	(262   6)	268
Précision	93.91%	
Mots corrects	47	
Mots incorrects	1	

### 1.3 Question 3

Pour paramétrer GNU Readline en mode Emacs il faut taper la commande suivante : **set -o emacs**

Pour paramétrer Emacs comme éditeur par défaut il faut taper la commande suivant : **update-alternatives --config editor**

Il faut ensuite choisir la ligne ou il y a écrit Emacs.

## 1.4 Question 4

Oui notre history contient des informations sensibles. Pour y remédier, il faut soit supprimer les lignes de notre history grâce à la commande **history -d <line number>** ou alors lorsque l'on rentre une information sensible mettre un espace avant pour que la ligne n'apparaisse pas dans l'historique.

L'employer utilise zsh, l'historique ce trouve donc dans : `~/ .zshrc_ history`

Pour éviter que les commandes courtes polluent notre historique on peut les ignorer en ajoutant au fichier `.bashrc` la ligne :

```
HISTIGNORE='cd* :ls* :pwd'
```

## 1.5 Question 5

**Fonction mkcd à rentrer dans .bashrc :**

```
function mkcd {
last=$(eval "echo \$$# ")
if [ ! -n "$last" ]; then
    echo "Entrer a directory name"
elif [ -d $last ]; then
    echo "\"$last' already exists"
else
    mkdir @$@ && cd $last
fi
}
```

**Fonction gitemergency à rentrer dans .bashrc :**

```
function gitemergency() {
    Git add.
    Git commit -a -m "$1"
    Git push
}
```

## 1.6 Question 6

```
Ajouter ce script dans /etc/bash_completion
_backup()
{
    local cur prev opts

    cur="\${COMP_WORDS[COMP_CWORD]}"
    prev="\${COMP_WORDS[COMP_CWORD-1]}"
    local files=("\${cur}*" )

    case \${COMP_CWORD} in
        1) opts='getent passwd cut -d : -f1';;
        2) opts="now tonight tomorrow";;
        3) opts="\${files[@]}";;
        *);;
    esac

    COMPREPLY=(
        \$(compgen -W "\${opts}" -- \${cur}) )
    return 0
}

complete -o nospace -F _backup backup
```

## 1.7 Question 7

Dans un premier temps il faut clone le dépôt oh-my-zsh.  
Ensuite il faut éditer le thème actuellement utilisé sur zsh de la façon suivante :

```
PROMPT="%{?:%{$fg_bold[green]%}→ :%{$fg_bold[red]%}→ )"
PROMPT+= ' %{$fg[cyan]%}%c%{$reset_color%} $(git_prompt_info) $(vagrant_prompt_info) '

ZSH_THEME_GIT_PROMPT_PREFIX="%{$fg_bold[blue]%}git:(%{$fg[red]%})"
ZSH_THEME_GIT_PROMPT_SUFFIX="%{$reset_color%} "
ZSH_THEME_GIT_PROMPT_DIRTY="%{$fg[blue]%}) %{$fg[yellow]%}x"
ZSH_THEME_GIT_PROMPT_CLEAN="%{$fg[blue]%})"

ZSH_THEME_VAGRANT_PROMPT_PREFIX="%{$fg_bold[blue]%}["
ZSH_THEME_VAGRANT_PROMPT_SUFFIX="%{$fg_bold[blue]%}%{$reset_color%} "
ZSH_THEME_VAGRANT_PROMPT_RUNNING="%{$fg_no_bold[green]%}●"
ZSH_THEME_VAGRANT_PROMPT_POWEROFF="%{$fg_no_bold[red]%}●"
ZSH_THEME_VAGRANT_PROMPT_SUSPENDED="%{$fg_no_bold[yellow]%}●"
ZSH_THEME_VAGRANT_PROMPT_NOT_CREATED="%{$fg_no_bold[white]%}○"
```

## 1.8 Question 8

Voici la fonction permettant d'éteindre ou de démarrer docker :

```
docker() {  
  dockerstatus=$(sudo service --status-all | grep docker | cut -d" " -f3)  
  if [[ $dockerstatus == "+" ]]  
    then  
      sudo systemctl stop docker  
  else  
      sudo systemctl start docker  
  fi  
}  
zle -N apacheXmariadb
```

## 1.9 Question 9

Nous avons téléchargé et essayé qterminal, terminal et GNOME terminale. Mais nous avons choisis d'utiliser qterminal car il y a des onglets de terminal, la séparations des terminaux, la transparence, et on peut choisir nos raccourcis clavier.

Pour désinstaller les autres terminaux nous avons utilisé la commande :

**apt remove**

## 2 SSH

### 2.1 Question 1

Pour se connecter a la vagrant srv sans utiliser la commande `vagrant ssh` il faut utiliser la commande **`ssh alice@srv.local`** et ensuite rentrer le mot de passe **1234**.

On peut remarquer que l'historique est vide, car c'est l'historique du vagrant.

### 2.2 Question 2

Dans un premier temps on créer une clé privée et publique via la commande **`ssh-keygen`**. On peut les retrouver dans le dossier `.ssh` .

Ensuite pour nous connecter en ssh sans mot de passe on doit déposer la clé publique sur le compte `alice@cli` via la commande :

```
ssh-copy-id -i ~/.ssh/id_rsa alice@srv.local
```

Il suffit ensuite de taper la commande **`ssh alice@srv.local`**. Pour déposer manuellement la clé sur `bob@cli` on doit d'abord sur notre machine copier notre clé publique :

```
cp ~/.ssh/id_rsa.pub
```

Et ensuite sur le vagrant ajouté notre clé dans le fichier `authorized_keys` :

```
cat /vagrant/id_rsa.pub >> ~/.ssh/authorized_keys
```

Une passphrase permet de débloquent la clé privée en local et n'est pas dans le bash history.

Pour éviter de la retaper à chaque fois il faut faire la commande suivante :

```
ssh-add ~/.ssh/id_rsa
```

### 2.3 Question 3

Pour ajouter la clé publique du serveur manuellement, il faut utiliser la commande suivante :

```
ssh-keyscan -H srv.local > ~/.ssh/known_hosts
```

Pour nous connecter sur le compte **`bob@cli`** en tapant **`ssh bc`** il faut créer un fichier de configuration dans `.ssh` et le configurer de la façon suivante :

```
Host bc
```

```
  Hostname 192.168.57.2
```

```
  User bob
```



Pour se connecter en stfp au compte `alice@cli` il faut utiliser la commande :

**sftp alice@cli.local**

Pour copier un fichier il faut utiliser la commande **get** par exemple :

**get /etc/apt/sources.list**

Et pour récupérer un fichier il faut utiliser la commande **put** par exemple :

**put test**

Pour se connecter en sshfs au compte `alice@cli` il faut dans un premier temps créer le fichier `/tmp/alicecli` puis utiliser la commande :

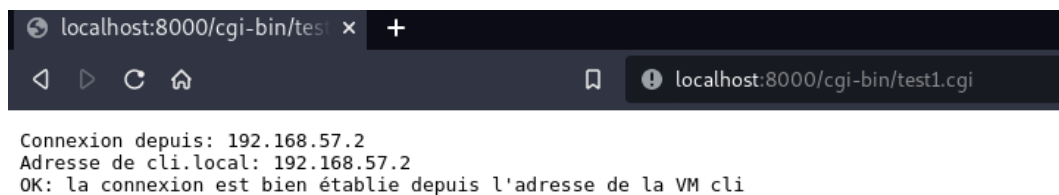
**sshfs alice@cli.local :/home /tmp/alicecli**

## 2.4 Question 4

Pour créer un tunnel ssh entre notre machine et `srv` en passant par `cli` il faut utiliser la commande suivante sur notre machine :

**ssh -L 8000 :srv.local :80 bob@cli.local**

Puis il faut aller sur `localhost :8000/cgi-bin/test1.cgi` pour vérifier si le tunnel fonctionne.



## 2.5 Question 5

Maintenant on veut se connecter au serveur web de la machine `srv` depuis `cli` sans que les données transitent en clair dans le réseau. Pour ça il faut dans un premier temps se connecter en ssh sur `srv`. Puis dans un second temps se connecter à la machine `cli` depuis `srv` via la commande :

**ssh -R 9000 :localhost :80 bob@cli.local**

Puis on vérifie qu'on arrive à se connecter au serveur web via la commande :

```
wget -nv -O - http://localhost:9000/cgi-bin/test2.cgi
```

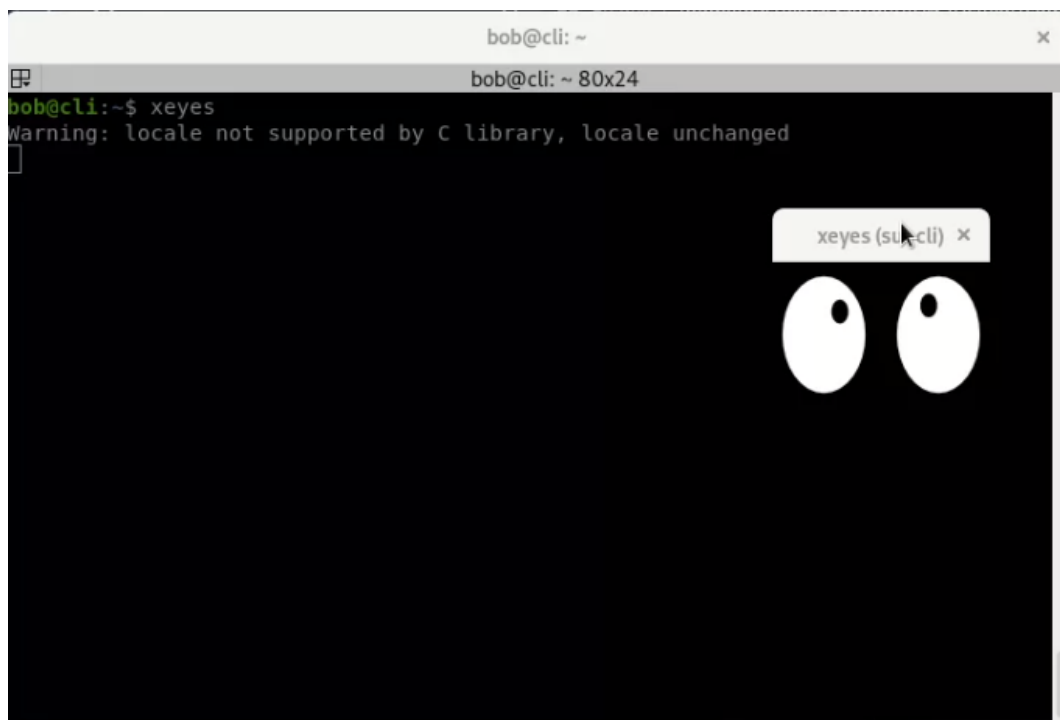
```
bob@cli:~$ wget -nv -O - http://localhost:9000/cgi-bin/test2.cgi
Connexion depuis: ::1
OK: la connexion est bien etablie depuis une adresse locale
2022-02-11 17:11:01 URL:http://localhost:9000/cgi-bin/test2.cgi [82] -> "-" [1]
```

## 2.6 Question 6

Après avoir téléchargé les paquets sur la machine cli, il faut se connecter en activant le forwarding grâce à la commande :

```
ssh -X bob@cli.local
```

Puis il suffit juste de lancer l'application.



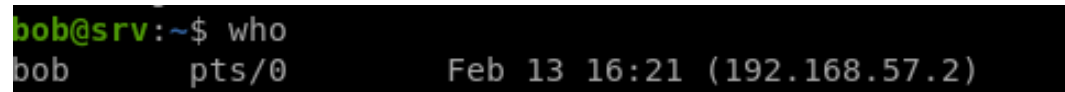
## 2.7 Question 7

Configuration du .ssh/config pour utiliser proxyjump de manière à rebondir automatiquement sur cli lorsqu'on se connect à srv :

Host srv.local

ProxyJump bob@cli.local :22

On peut ensuite vérifier avec quelle adresse on c'est connecté avec la commande who.

A terminal window with a black background. The prompt is 'bob@srv:~\$'. The command 'who' has been entered. The output is 'bob pts/0 Feb 13 16:21 (192.168.57.2)'.

```
bob@srv:~$ who
bob pts/0 Feb 13 16:21 (192.168.57.2)
```

Configuration du .ssh/config pour utiliser proxycommand :

Host srv.local

ProxyCommand ssh cli -W %h :%p

## 3 GIT

### 3.1 Question 1

Pour initialiser un projet il faut ce mettre dans un répertoire et taper la commande **git init**.

Lorsqu'on démarre et arrête la configuration Vagrant, un fichier `.vagrant` apparaît. Pour l'ignorer il faut créer un fichier `.gitignore` a la racine du répertoire et y ajouter `.vagrant` dedans.

Pour ajouter les fichier a la staging area il faut utiliser la commande **git add .** et ensuite utiliser la commande **git commit -m "*commentaire*"** pour commit les fichiers sur le repository local.

Pour vérifier notre commit à fonctionner il faut utiliser la commande **git log**.

```
commit 8ed1f1bb1bc7462fc0440a10dcc2d15f95d276ae
Author: plubzzzz <bastien.grosclaude@orange.fr>
Date: Thu Jan 13 09:53:23 2022 +0100

    dossier vagrant
```

### 3.2 Question 2

Pour créer une nouvelle branche il faut utiliser la commande **git checkout -b patrick**.

Pour effectuer 2 commits distincts il faut utiliser la commande **git add -p**.

```

bastien@debianbastien:~/git1$ git add -p
diff --git a/Vagrantfile b/Vagrantfile
index e88caa0..162058b 100644
--- a/Vagrantfile
+++ b/Vagrantfile
@@ -14,7 +14,7 @@ Vagrant.configure("2") do |config|
  useradd --shell /bin/bash --create-home carol || true
  echo alice:1234 | chpasswd
  echo bob:azerty | chpasswd
-  echo patrick:azerty123 | chpasswd
+  echo patrick:azerty123 | chpasswd
  echo carol:secret | chpasswd
  echo Enabling password auth
  sed -i 's/PasswordAuthentication no/PasswordAuthentication yes/' /etc/ssh/sshd_config
(1/2) Stage this hunk [y,n,q,a,d,j,J,g,/,e,?]? y
@@ -33,8 +33,8 @@ Vagrant.configure("2") do |config|
  b.vm.hostname = "srv"

  b.vm.provision "shell", inline: <<-SHELL
-  apt-get -y install apache2 ruby ph
-  a2enmod cgi ph
+  apt-get -y install apache2 ruby php
+  a2enmod cgi php
  systemctl restart apache2.service
  cp /vagrant/srv/test1.cgi /usr/lib/cgi-bin/test1.cgi
  cp /vagrant/srv/test2.cgi /usr/lib/cgi-bin/test2.cgi
(2/2) Stage this hunk [y,n,q,a,d,K,g,/,e,?]? n

bastien@debianbastien:~/git1$ git commit -m "patrick"
[patrick e950b29] patrick
1 file changed, 3 insertions(+), 2 deletions(-)
bastien@debianbastien:~/git1$ git add .
bastien@debianbastien:~/git1$ git commit -m "php"
[patrick 43e1d5b] php
1 file changed, 2 insertions(+), 2 deletions(-)

```

Le working directory de la branche master est vide.

Pour intégrer les modification dans main il faut utiliser la commande **git merge patrick**.

Oui la branche que nous avons créée existe encore mais on peut la supprimer car nous avons intégré les modification. Pour supprimer la branche il faut utiliser la commande **git branch -D patrick**.

### 3.3 Question 3

Lorsqu'on essaye de merger la branche forward-new-port il y un conflit qui empêche le merge.

```

bastien@debianbastien:~/git1$ git merge forward-new-port
Fusion automatique de Vagrantfile
CONFLICT (contenu) : Conflit de fusion dans Vagrantfile
La fusion automatique a échoué ; réglez les conflits et validez le résultat.

```

Il faut donc modifier le fichier à la main pour le résoudre.