CONFIGURACIÓN INICIAL

MÁQUINA VIRTUAL

ACCESO A MÁQUINA

- Máquina virtual Ubuntu mediante Vmware
  - Usuario: curso
  - pwd: P@ssw0rd

LISTA DE APLICACIONES INSTALADAS

- Chrome
- Postman
- Robo3T
- Guake
- Visual Code Editor

LISTA DE PAQUETES INSTALADOS

- zsh & Oh-my-zsh
- git
- Docker CE y Docker Compose
- curl y wget
- nvm, node, avn
- chrome-gnome-shell

INSTALACIÓN DE APLICACIONES

INSTALACIÓN DE CHROME

- Desde Ubuntu 18 no es necesario ejecutar apt update después de añadir el repo :-)

```
wget -q -O - https://dl-ssl.google.com/linux/linux_signing_key
sudo add-apt-repository "deb http://dl.google.com/linux/chrome
sudo apt install google-chrome-stable
```

INSTALACIÓN DE POSTMAN

```
snap install postman
```

INSTALACIÓN DE ROBO3T (ROBOMONGO)

```
snap install robomongo
```

INSTALACIÓN DE GUAKE

```
sudo apt install guake
```

- La versión 3.0.5 tiene un bug, por lo que al hacer exit se queda colgado. Lo compilamos mejor

CONFIGURACIÓN DE GUAKE

- Cambiamos preferencias F1 para que se vea
- Cambiamos nivel de transparencia
- Lo configuramos como aplicación de inicio

CONFIGURACIÓN MODO NOCTURNO

- Ajustes -> Pantalla -> Se pueden ajustar los azules :-)

INSTALACIÓN DE VISUAL CODE EDITOR

- Instrucciones en la web:

```
curl https://packages.microsoft.com/keys/microsoft.asc | gpg
sudo mv microsoft.gpg /etc/apt/trusted.gpg.d/microsoft.gpg
sudo sh -c 'echo "deb [arch=amd64] https://packages.microsoft
sudo apt update
sudo apt install code # or code-insiders
```

- Peviamente instalamos curl (y wget ya que estamos):

```
sudo apt install curl wget
```

CONFIGURACIÓN GNOME-SHELL

- Mediante web https://extensions.gnome.org/
- Instalamos extensión *Integración con GNOME Shell*
- Instalamos conector:

```
sudo apt install chrome-gnome-shell
```

- Instalo aplicación Alternate Tab
  - Cambio de aplicación mediante *ALT+TAB* sin agrupar por aplicación

INSTALACIÓN DE PAQUETES

INSTALACIÓN DE DOCKER CE

- Ver documentación

```
sudo apt-get install apt-transport-https ca-certificates curl
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
sudo add-apt-repository "deb [arch=amd64] https://download.doc
sudo apt update
sudo apt install docker-ce
```

- Comprobamos que esté arrancado:

```
sudo systemctl status docker
```

CONFIGURACIÓN DOCKER CE

- Ejecutar el comando docker sin sudo:

```
sudo usermod -aG docker ${USER}
```

INSTALACIÓN DOCKER COMPOSE

- Ver documentación

```
sudo curl -L https://github.com/docker/compose/releases/downlo
sudo chmod +x /usr/local/bin/docker-compose
```

- Testear instalación:

```
$ docker-compose --version
docker-compose version 1.22.0, build 171
```

INSTALACIÓN DE ZSH

- Algunos prefieren fish
- Otros son fieles a bash
- Yo prefiero zsh:

```
sudo apt install zsh
chsh -s $(which zsh)
```

INSTALACIÓN DE OH-MY-ZSH

- Ver instrucciones y configuración

```
sh -c "$(curl -fsSL https://raw.github.com/robbyrussell/oh-my-
```

- Plugins añadidos: yarn npm node sudo web-search

INSTALACIÓN DE NODE.JS

TIPOS DE INSTALACIÓN

- Desde el sitio web de Node.js
- Mediante un gestor de versiones:
  - Varias versiones de Node.js en la misma máquina
  - Evitar tener que hacer sudo cuando instalemos paquetes de forma global
    - Los paquetes globales se instalan para un único usuario y version de node
    - Los paquetes globales sirven para cualquier proyecto

GESTORES DE VERSIONES DE NODE

- Los gestores de versiones más habituales son:
- nvm para Linux/Mac
- nvm-windows para Windows (sin relación con nvm)
- n para Linux/Mac

INSTALACIÓN DE NVM

- Ver instrucciones

```
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.3
```

CONFIGURACIÓN DE NVM

- Añado en el .zshrc para que se ejecute:

```
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads
```

INSTALACIÓN NODE MEDIANTE NVM

- Ver versiones instaladas:

```
nvm ls
```

- Instalar una versión de node (hasta ultimo patch):

```
nvm install 8.11
```

- Instala la última versión lts:

```
nvm install node --lts
```

- Instala la última versión disponible:

```
nvm install node
```

SELECCIONAR VERSIÓN DE NODE

- Usar una versión en particular:

```
nvm use 8.11.3
```

- Usar una versión en particular al abrir shell:

```
nvm alias default 8.11.3
```

SELECCIONAR VERSIÓN DE NODE POR PROYECTO

- A mano:
  - La versión se define en un fichero *.nvmrc*
  - Se ejecuta el comando `nvm use` para que lea la versión
- Automáticamente:
  - Modificando el fichero *.zshrc*
  - Mediante un paquete adicional: **avn**

INSTALACIÓN DE AVN

- avn sirve para *detectar la versión de node de cada proyecto* y cambiar el ejecutable que se utiliza
- avn se instala mediante el gestor de paquetes

```
$ npm install -g avn avn-nvm
$ avn setup
```

- Es un poco lento, no del gusto de todos

CONFIGURACIÓN AVN

- Creamos fichero *.node-versión* dentro de nuestro proyecto

```
cd proyecto
echo v10 >.node-version
```

- Cada vez que entremos en el directorio se activará la versión de node correspondiente
  - ¡Debe estar instalada!

LINTER PARA NODE.JS

- Eslint analiza el código y nos muestra errores.
- Eslint avisa de muchos errores, pero solo arregla algunos
- Utilizaremos prettier
  - Formateador de código
  - Se combina con eslint y arregla más errores

INSTALACIÓN Y CONFIGURACIÓN DE GIT

```
sudo apt install git
git config --global user.name "Your Name"
git config --global user.email "youremail@domain.com"
git config --global core.editor "vim"
```

- Comprobamos:

```
$ git config --list
```

CONFIGURACIÓN GITHUB

- Crear cuenta en GitHub si no tienes
- Copiar clave pública a GitHub si usas ssh

EDITOR DE CÓDIGO

VISUAL CODE EDITOR

- Utilizaremos Visual Code Editor
- Es un producto open source de Microsoft realizado mediante node.js (electron)
- Tiene un buen debugger para node.js

VISTAZO GENERAL

- Emmet ya viene por defecto
- Visor markdown por defecto
- Integración con GitHub
- Debug integrado
- Gestor de extensiones integrado
  - Extensiones online

EDICIÓN

- Archivo, Preferencias, Configuración:

```
"editor.tabSize": 2
```

JEST

Jest: autocompletado para tests mediante Jest

CHROME DEBUGGER

- Debug mediante Visual Studio de código web

SINCRONIZAR CONFIGURACIÓN ENTRE EQUIPOS

- Instalamos settings sync para sincronizar configuración entre equipos

AUTOCOMPLETADO

- Autocompletado de path:
  - Utilizamos plugin Path Autocomplete
  - Para que funcione en markdown, hay que cambiar la configuración (Archivo, Preferencias, Configuración)

```
"path-autocomplete.triggerOutsideStrings": true
```

LINTER

- Utilizaremos eslint
- Tendremos que instalar posteriormente el propio linter y establecer su configuración
  - En cada proyecto en particular

FORMATEADOR DE CÓDIGO

- prettier

MARKDOWN LINTER

- markdownlint
- Cambio la configuración del linter en mi proyecto mediante *.markdownlist.json*

```
{
  "MD012": false
}
```

MARKDOWN

- No es necesario, Visual Studio tiene un visor de Markdown integrado
- Markdown All in One

SERVIDOR WEB

- Se instala y se pulsa luego en el botón Go Live (parte inferior).
- Live Server

EXPRESSSNIPPET

- Para autocompletado con Express
- Se disparan con *app*

PROMISE SNIPPET

- Para autocompletado de promesas
- Se disparan con *promise*

DOCKER

- Para gestionar contenedores e imágenes de docker

MOCHA

- La librería más popular para tests con Node.js
- Snippets para Mocha
- Workspace para tests

INTRODUCCIÓN A NODE.JS

- Vamos a ver un vistazo general a node.js antes de entrar en proyectos