Desplegament d'aplicacions web en servidors remots


Ara que ja sabem com configurar un servidor remot per a accedir a ell, veurem en este document com instal·lar el programari necessari per a desplegar aplicacions web en este servidor.

> **NOTA IMPORTANT**: convé que no instal·les res del que s'explica ací directament en el VPS, ja que es donen les nocions importants per a instal·lar diversos servidors web (Apache, Nginx), diverses bases de dades (MariaDB, MongoDB), però possiblement no vas a tindre totes eixes coses en el servidor. Limita't a instal·lar el que s'indica en els exercicis només, que serà el que utilitzem en el curs.

1. Passos principals per a desplegament d'aplicacions Node.js

En primer lloc veurem quins passos són els recomanables per a poder desplegar còmodament aplicacions Node.js en servidors remots.

1.1. Instal·lació d'Apache

Apache és, ara com ara, el servidor més utilitzat per a desenvolupament d'aplicacions web. La seua instal·lació i configuració varien lleugerament d'un sistema operatiu a un altre, especialment quant al nom i ubicació dels executables i dels fitxers de configuració. Ací veurem el cas que ens ocupa: instal·lar-ho en una distribució Debian per al nostre VPS.

Per a instal·lar Apache en Debian, escrivim este comando:

```
sudo apt-get install apache2
```

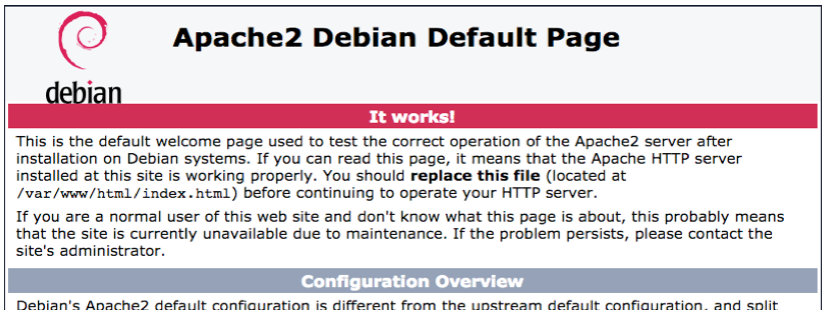
Automàticament, Apache quedarà instal·lat com a servici, i s'iniciarà amb el sistema operatiu. Podem iniciar, detindre o reiniciar el servidor amb estos comandos (l'opció `reload` és més lleugera i menys profunda que `restart`, i s'empra per a recarregar els arxius de configuració simplement):

```
sudo systemctl start apache2
sudo systemctl stop apache2
sudo systemctl restart apache2
sudo systemctl reload apache2
```

O també amb estos altres:

```
sudo /etc/init.d/apache2 start
sudo /etc/init.d/apache2 stop
sudo /etc/init.d/apache2 reload
sudo /etc/init.d/apache2 restart
```

Per defecte, Apache queda escoltant en el port 80, així que per a provar que està en marxa, podem accedir a la URL que ens van donar per al nostre VPS en registrar (per exemple, *vps-xxxxxxxx.vps.ovh.net*). Si hem contractat un nom de domini i l'hem associat a la IP del nostre VPS, també ho podem usar per a accedir a Apache. Veurem una pàgina de benvinguda similar a esta:



1.1.1. Configuració bàsica

La configuració d'Apache en Linux Debian es troba distribuïda en diversos arxius, tots ells situats en la carpeta `/etc/apache2`. Abans d'editar qualsevol d'estos arxius, convé fer una còpia de seguretat del seu estat original, per si els canvis deixen a l'arxiu amb errors, i Apache no pot iniciar. Per exemple:

```
sudo cp /etc/apache2/apache2.conf /etc/apache2/apache2.conf.original
```

El **arxiu principal de configuració** d'Apache, dins de la carpeta anterior, és `apache2.conf`. En ell existixen referències als altres arxius de configuració, que permeten gestionar ports, hosts virtuals i altres elements. Dins d'este arxiu principal en si, podem configurar coses com:

- La carpeta per defecte dels arxius de configuració (paràmetre `ServerRoot`)
- La carpeta arrel on se situaran els documents web (paràmetre `DocumentRoot`). Per defecte, el valor d'este últim element és `var/www/html`.
- Noms d'arxius per defecte a carregar si no s'especifica cap en la URL (paràmetre `DirectoryIndex`).
- Arxiu on s'emmagatzemaran els missatges d'error produïts (paràmetre `ErrorLog`, el valor del qual per defecte és `/var/log/apache2/error.log`)
- ... etc.

El **arxiu de gestió de ports** també es troba dins de la carpeta `/etc/apache2`. En concret, és l'arxiu `ports.conf`, i conté els ports que s'habiliten per a treballar amb Apache. Per defecte, hi ha dos ports

habilitats:

- El port 80 per a connexions HTTP
- El port 443 per a connexions HTTPS

Podem editar qualsevol d'estos ports per a fer que Apache escolte estes connexions per altres ports, i podem habilitar més que eixos dos ports, afegint més directives `Llisten` a les que ja hi ha.

> **Exercici 1:**
>
> Instal·la Apache en el teu VPS seguint les instruccions indicades en este apartat. Edita el fitxer `ports.conf` per a habilitar també els ports 8000 i 8080. Reinicia el servidor perquè incloga els canvis.

1.1.2. Definint hosts virtuals

Els hosts virtuals permeten que un solo ordenador puga allotjar múltiples dominis i pàgines web, de manera que una sola adreça IP pot respondre a diferents noms de domini. També un host virtual permet assignar un subdomini a una aplicació concreta, situada en una carpeta determinada. Per exemple, si el nostre domini és *daw.ovh*, podem definir un subdomini anomenat *proves.daw.ovh*, i col·locar la web en la carpeta */home/usuari/proves*, per exemple.

En la carpeta `/etc/apache2/sites-available` es tenen definits els hosts virtuals, i en la carpeta `/etc/apache2/sites-enable` es tenen enllaços simbòlics als hosts de la carpeta anterior que estan activats o habilitats actualment.

El servidor web Apache s'instal·la per defecte amb un host virtual ja definit. La configuració d'este host la podem trobar en `/etc/apache2/sites-available/000-default.conf`, i al·ludix a la carpeta de documents per defecte `/var/www/html`. Per defecte este lloc virtual està habilitat, per la qual cosa podem comprovar que existix un enllaç simbòlic a este fitxer en el directori `/etc/apache2/sites-enable`.

Suposem que volem configurar un *host* virtual per al subdomini *maycalle.ovh* (que ja tindrem registrat prèviament en OVH), de manera que atenga peticions en el port 80 i carregue la web que tenim emmagatzemada en `/home/usuari/librosweb`. En este cas, hem de crear un arxiu de configuració en `/etc/apache2/sites-available`. Podem copiar el que ja hi ha i donar-li un altre nom:

```
cd /etc/apache2/sites-available
cp 000-default.conf 001-librosweb.conf
```

Després, editem este nou arxiu `001-librosweb.conf` i ho configurem amb les dades del nom de domini o subdomini associat, i carpeta de la web. Afegim també un bloc `<Directory>` per a donar permisos d'accés a la carpeta en qüestió.

```
<VirtualHost *:80>
    ServerAdmin may.calle@gmail.com
    ServerName maycalle.ovh
    DocumentRoot "/home/debian/librosweb"
    DirectoryIndex index.js
    <Directory "/home/debian/librosweb">
       Require all granted
    </Directory>
</VirtualHost>
```

Finalment, hem d'habilitar este nou lloc, per al que usarem el comando `ln` per a crear un *àlies* de l'arxiu en la carpeta `/etc/apache2/sites-enabled`. El següent comando s'escriu en una sola línia:

```
sudo ln -s /etc/apache2/sites-available/001-librosweb.conf
/etc/apache2/sites-enabled
```

Per a deshabilitar el lloc simplement hem d'esborrar l'enllaç simbòlic de l'altra carpeta.

Després dels canvis realitzats, haurem de reiniciar el servidor perquè els incorpore:

```
sudo systemctl reload apache2
```

Veurem que en visitar `maycalle.ovh` en el navegador es mostra el contingut de l'arxiu *index.js* en lloc d'executar-se, això és perquè Apache, per defecte, no processa arxius *.js* a diferència dels arxius *.php*. Per tant, Apache està servint l'arxiu index.js com un arxiu estàtic. Una possible solució és configurar Apache per a actuar com un proxy invers que reenvie les sol·licituds al port on s'executa la teua aplicació Node.js o utilitzar **Phusion Passenger**, que veurem més endavant.

> **Exercici 2:**
>
> Per a realitzar este exercici seguirem estos passos:
>
> 1. Descarrega este exemple de projecte Node.js que hem utilitzat en alguna sessió anterior. Conté un codi simple per a provar l'autenticació basada en sessions amb Express (sense base de dades). Descomprimeix-ho en una carpeta
> 2. Puja-ho a un repositori teu en GitHub. Per a això:
>   - Crea un repositori nou amb el nom que vulgues (per exemple *ProvaSessionsExpress*), i afig un fitxer *README*
>   - Clona el repositori en el teu ordinador, i còpia dins el codi del projecte descomprimit del pas 1.
>   - Afig un fitxer `.gitignore` que incloga la carpeta `node_modules`
>   - Fes un *Commit* i un *Push* per a pujar els canvis a GitHub

3. Obri sessió en el teu VPS i descarrega el projecte en la teua carpeta d'usuari (utilitza *git clone* per a clonar el repositori la primera vegada, i *git pull* si fas canvis les següents vegades).
4. Afig un *virtual host* en Apache (carpeta */etc/apache2/sites-available*) perquè accedisca a eixa carpeta pel port 8080:

```
<VirtualHost *:8080>
   ServerAdmin el_teu_correu@gmail.com
   ServerName vps-xxxxxx-vps.ovh.net
   DocumentRoot "/home/debian/ProvaSessionsExpress"
   DirectoryIndex index.js
   <Directory "/home/debian/ProvaSessionsExpress">
      Require all granted
   </Directory>
</VirtualHost>
```

5. Recorda afegir també l'enllaç simbòlic en la carpeta *sites-enabled* Reinicia el servidor i prova de carregar el fitxer *index.js* (es mostrarà directament el codi font)

1.2. Node.js

A l'hora d'instal·lar Node.js, podem emprar el gestor de versions *NVM* o bé la distribució corresponent de Node per al nostre sistema operatiu. En este document s'explica detalladament com instal·lar l'una o l'altra opció.

> **Exercici 3:**
>
> Instal·la Node.js (l'última versió LTS disponible) en la teua VPS. És preferible que ho instal·les des de la web oficial, sense usar NVM, seguint estes instruccions per a Linux.

1.3. Enllaçar Node amb Apache a través del servidor Passenger

És habitual que en un servidor remot coexistisquen diferents servidors web, com per exemple Node i Apache, o Apache i Nginx. En este cas, no és molt recomanable haver d'accedir a les aplicacions Node per un port diferent al port per defecte, llevat que siguen aplicacions de prova o per a ús particular, però tampoc podem deixar escoltant a Node pel port 80 si ja hi ha un altre servidor ocupant eixe port... llevat que comuniquem tots dos servidors.

Passenger és un servidor d'aplicacions que permet treballar amb aplicacions desenvolupades en diversos frameworks, com per exemple Ruby o Node. Monitora les aplicacions per a tornar a alçar-les si cauen, i oferix alguns avantatges addicionals, com el balanceig de càrrega. A més, disposa de mòduls per a integrar-se amb els servidors Apache o Nginx, de manera que, utilitzant els mateixos ports, podem fer que estos servidors "passen" a Passenger les aplicacions que indiquem en els corresponents arxius de configuració.

1.3.1. Instal·lació de Passenger

Instal·lem Passenger i el mòdul per a Apache a través del repositori oficial. Cada comando *sudo* a continuació ha d'introduir-se en una sola línia independent de la resta, en el mateix orde.

```
sudo apt-get install -y dirmngr gnupg

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80
--recv-keys 561F9B9CAC40B2F7

sudo apt-get install -y apt-transport-https ca-certificates

sudo sh -c 'echo deb https://oss-binaries.phusionpassenger.com/
apt/passenger bullseye main > /etc/apt/sources.list.d/passenger.list'

sudo apt-get update

sudo apt-get install -y libapache2-mod-passenger
```

Habilitem el mòdul Passenger per a Apache, i reiniciem el servidor

```
sudo a2enmod passenger
sudo systemctl restart apache2
```

Verificar la instal·lació abans de continuar.

```
sudo /usr/bin/passenger-config validate-install
```

1.3.2. Configuració d'un *host virtual* amb Passenger

Afegir un nou arxiu de configuració de *virtual host* per a la nostra aplicació Node:

```
sudo nano /etc/apache2/sites-available/appnode.conf
```

Editar l'arxiu per a deixar-lo amb esta aparença:

```
<VirtualHost *:80>

   ServerName el_teu_domini.com

   DocumentRoot /home/usuari/web_node
   PassengerAppRoot /home/usuari/web_node

   PassengerAppType node
   PassengerStartupFile app.js

   <Directory /home/usuari/web_node>
      Allow from all
      Options -MultiViews
      Require all granted
   </Directory>

</VirtualHost>
```

Habilitar el nou *virtual host*, i reiniciar Apache per a acceptar els canvis:

```
sudo ln -s /etc/apache2/sites-available/app.node.conf
/etc/apache2/sites-enabled

sudo systemctl reload apache2
```

Ara la nostra nova web ja ha d'estar visible i operativa. Podem també comprovar-ho amb el comando `curl`, a més del navegador:

```
curl http://www.tu_dominio.com
```

> **NOTA**: òbviament, tant el domini de `ServerName` com les carpetes indicades en `DocumentRoot`, `PassengerAppRoot` i `Directory` les adaptarem a les que requerisca la nostra aplicació.

> **NOTA**: les propietats `DocumentRoot` i `PassengerAppRoot` normalment apuntaran a la mateixa carpeta, on es trobe l'arxiu principal de la nostra aplicació Node.js. Si estem utilitzant algun framework especial, com Nest.js, on no hi ha un arxiu principal específic (en este cas, l'aplicació es llança amb *npm run* i alguna opció predefinida en l'arxiu *package.json*), llavors hem de construir l'aplicació (en el cas de Nest.js, usem `npm run build`), i les propietats anteriors de l'arxiu de configuració d'Apache han d'apuntar a la subcarpeta `dist` dins del projecte, sent l'arxiu principal (propietat `PassengerStartupFile`) l'arxiu `main.js`, dins d'esta carpeta.

Estos passos estan trets de la web oficial. També allí es pot obtindre com instal·lar Passenger de manera autònoma (*standalone*), i també integrat amb Nginx.

Exercici 4:

Completa els següents passos:

1. Instal·la Passenger seguint els passos indicats en este apartat
2. Modifica el *virtual host* que has creat abans en l'*Exercici 2* per a configurar-lo amb Passenger:

```
<VirtualHost *:8080>
   ServerAdmin el_teu_correu@gmail.com
   ServerName vps-xxxxxx-vps.ovh.net
   DocumentRoot "/home/debian/ProvaSessionsExpress"
   PassengerAppRoot "/home/debian/ProvaSessionsExpress"
   PassengerAppType node
   PassengerStartupFile index.js
   <Directory "/home/debian/ProvaSessionsExpress">
      Allow from all
      Options -MultiViews
      Require all granted
   </Directory>
</VirtualHost>
```

3. Reinicia Apache
4. Executa *npm install* en la teua carpeta */home/debian/ProvaSessionsExpress* perquè s'instal·len els paquets necessaris.
5. Prova d'accedir de nou a la web (ruta arrel) i ara ja hauria de veure's tot correctament.

2. Altres instal·lacions addicionals o alternatives

Com a alternatives o complements als passos seguits en l'apartat anterior, ací indicarem altres opcions que podem tindre en compte a l'hora de ser instal·lades en el VPS.

2.1. Nginx

Quan un sent parlar de servidors web, possiblement Apache és el primer que li ve a la ment. I no és casualitat, ja que entorn del 30 o 35% del trànsit web se servix actualment amb este servidor. No obstant això, en els últims anys han sorgit alternatives a este, i una de les quals ha arribat amb més força és Nginx. Actualment servix més del 20% de les webs existents.

Nginx és un servidor HTTP lleuger, d'origen rus. Inicialment es va crear per a donar suport a una web que rebia entorn de 500 milions de visites diàries, i actualment també s'empra com a servidor base d'altres llocs web coneguts, com Reddit o Dropbox.

Entre les característiques que podríem citar de Nginx, les més rellevants són:

- Està basat en un **model de connexions asíncron**. És a dir, en lloc de col·lapsar la memòria amb centenars de processos que gestionen els centenars de connexions simultànies d'una web, Nginx posa en marxa un procés per nucli de processador, i tots ells gestionen alhora milers de connexions. Això permet una càrrega molt de menor de treball per a la CPU, i menys consum de memòria.
- És **open source**, i funciona en diferents plataformes, encara que per al seu ús en producció s'aconsellen distribucions Linux, com Ubuntu, Debian o CentOS, entre altres.
- Està basat en **mòduls**, un potent i eficaç sistema de plugins que permet dotar de funcionalitats addicionals al servidor, si es requerix. El propi nucli de Nginx ja ve amb una gran varietat de mòduls incorporats, però també podem descarregar altres mòduls de tercers i instal·lar-los. En este sentit, té una filosofia similar a Node.js, per exemple.

Amb tot l'anterior, podem concloure que Nginx és una mescla d'eficiència, velocitat i potència, i per això cada dia és una alternativa més sòlida a Apache.

2.1.1. Instal·lació i posada en marxa

Vegem com descarregar, instal·lar i posar en marxa una instància bàsica de Nginx. Els primers passos són similars a qualsevol altra ferramenta sota sistemes Linux: actualitzem repositoris i instal·lem el paquet:

```
sudo apt update
sudo apt install nginx
```

> **NOTA**: si ja tenim una instància d'un altre servidor (per exemple, Apache), corrent en el port 80, és convenient que detinguem el servici temporalment, perquè Nginx puga completar la seua instal·lació i posada en marxa en este port 80. Més endavant veurem com podem canviar el port i fer que cada servidor tinga el seu, si és necessari.

Per defecte, Nginx utilitza la mateixa carpeta de documents web que Apache (`/var/www/html`), per la qual cosa, si tenim Apache ja instal·lat, Nginx utilitzarà esta pàgina d'inici, i deixarà la seua canviada de nom en la mateixa carpeta, en un segon pla. Si Nginx és el nostre primer servidor, llavors en connectar al port 80 des del nostre domini (per exemple, *vps112233.ovh.net* o similar), veurem la pàgina de benvinguda de Nginx.

L'arxiu de configuració de Nginx és `/etc/nginx/nginx.conf` . Internament redirigix o inclou altres arxius de configuració addicionals, i de fet, gran part de la configuració que necessitem la farem en eixos arxius.

2.1.2. Afegir hosts virtuals

La mecànica de Nginx per a habilitar llocs web és similar a la utilitzada per Apache: existix una carpeta `/etc/nginx/sites-available` on definir arxius de configuració per als diferents hosts virtuals que tinguem (a més de la pròpia carpeta `/var/www/html`), i després existix una carpeta paral·lela `/etc/nginx/sites-enabled` amb enllaços simbòlics als llocs de la carpeta anterior que vulguem tindre actius a cada moment.

Per a afegir un nou host virtual en Nginx, n'hi ha prou amb afegir un nou arxiu a la carpeta `/etc/nginx/sites-available`. Ja existix un arxiu `default` que apunta a la carpeta per defecte `/var/www/html`, i ho podem utilitzar per a crear un duplicat, per exemple, `la_meua_web.com`, amb una aparença com esta:

```
server {
   listen 80;
   listen [::]:80;

   root /home/usuari/proves;
   index index.html index.htm;

   server_name la_meua_web.com;

   location / {
      try_files $uri $uri/ =404;
   }
}
```

- La directiva `listen` s'empra per a indicar el port pel qual escoltar. La segona directiva `listen` amb la sintaxi `[::]` s'empra per a direccions IPv6.
- La directiva `root` indica la carpeta arrel dels documents d'este host virtual
- La directiva `index` conté les pàgines que es carregaran quan s'accedisca a la carpeta de la web sense indicar cap. En este cas es buscarà primer la pàgina `index.html`, i si no es troba, es buscarà `index.htm` en segon lloc. També podríem incloure ací `index.php`, per exemple, si treballem amb PHP.
- La directiva `server_name` és el nom associat a l'host virtual (normalment, un àlies o subdomini)
- Els grups `location` s'empren per a configuracions particulars d'unes certes rutes. En este cas, s'emet un codi 404 en el cas que l'arxiu sol·licitat no es trobe.

Una vegada tenim el lloc configurat, per a fer-lo visible hem de crear un enllaç simbòlic d'este arxiu en la carpeta `sites-enabled`, amb el següent comando (en una sola línia):

```
sudo ln -s /etc/nginx/sites-available/la_meua_web.com
/etc/nginx/sites-enabled
```

2.1.3. Redirecció de peticions

En alguns tipus d'aplicacions, com per exemple aplicacions SPA (*Single Page Applications*), o fetes amb frameworks com a Angular, pot resultar interessant redirigir qualsevol URL que sol·licite el client a un únic recurs (per exemple, a la pàgina `index.html` en l'arrel de l'aplicació). Per a això, podem crear un arxiu de configuració (còpia de `default.conf`, com en el cas anterior), com este. La línia amb la instrucció `rewrite` s'encarrega de la redirecció pròpiament dita.

```
server {
   listen 80;
   server_name la_meua_web.com;

   root /home/debian/proves;
   rewrite ^\/.+$ /index.html last;

   location / {
 }
 ...
```

2.1.4. L'arxiu *favicon.ico* i els errors en el *log*

En alguns casos, en carregar una URL es tendix a buscar l'arxiu per defecte "favicon.ico", i si no es troba ens podem trobar amb missatge d'error en el navegador, o amb el corresponent missatge de log en l'arxiu d'errors. Per a evitar això, podem afegir esta directiva a la configuració del nostre host virtual:

```
location = /favicon.ico {
   return 204;
   access_log off;
   log_not_found off;
}
```

D'altra banda, davant qualsevol error en l'execució de Nginx, es generaran els corresponents missatges en l'arxiu de logs, situat en `/var/log/nginx/error.log`. Podem consultar-ho si alguna cosa va malament per a tindre una mica més d'informació sobre l'error. Si és degut a un error de sintaxi en algun fitxer de configuració, podem tindre alguna pista d'en quin arxiu i línia està l'error.

2.2. Definir *hosts* virtuals des de Node.js

En la majoria d'exemples que hem vist o veurem realitzarem una única aplicació Node que posarem en marxa per si mateixa. Però és possible que en un mateix servidor tinguem coexistint diverses aplicacions, cadascuna d'elles associada a un subdomini diferent. Si hem configurat Passenger amb Apache podem definir un *host virtual* per a cada aplicació, amb el seu domini associat, que apunt a cada carpeta.

No obstant això, de manera alternativa, podem gestionar totes les aplicacions i *hosts virtuals* des de Node.js d'una altra forma. Per a començar, pot ser bona idea definir una carpeta on situar tots els projectes (en

subcarpetas independents). Podem crear una carpeta anomenada "*ProjectesNode*" dins de la nostra carpeta d'usuari, amb una subcarpeta per a cada projecte.

Suposem que definim un primer projecte anomenat "*Principal*" en la nostra carpeta de "*ProjectesNode*". Instal·lem Express en esta carpeta, i deixem un arxiu `app.js` com este:

```
const express = require('express');

let app = express();

app.get('/', (req, res) => {
 res.send('Benvingut a la web principal');
});

module.exports.app = app;
```

De la mateixa manera, podem crear un altre projecte, anomenat "*Proves*", per exemple, en la nostra carpeta de "*ProjectesNode*". Instal·lem novament Express i deixem l'arxiu `app.js` així:

```
const express = require('express');

let app = express();

app.get('/', (req, res) => {
 res.send('Benvingut a la web de proves');
});

module.exports.app = app;
```

Ara crearem una tercera carpeta anomenada "*Manager*", en la mateixa carpeta de "*ProjectesNode*". Instal·lem també Express, i una llibreria addicional anomenada `vhost`, que s'encarrega d'associar hosts virtuals amb aplicacions Express. L'arxiu `app.js` quedarà com seguix:

```
const express = require('express');
const vhost = require('vhost');

let app = express();

app.use(vhost('nachoib.ovh',
   require(__dirname + '/../Principal/app').app));
app.use(vhost('proves.nachoib.ovh',
   require(__dirname + '/../Proves/app').app));
app.listen(3000);
```

En este últim cas, el que fem és associar el nom *nachoib.ovh* amb l'aplicació "*Principal*", i el subdomini *proves.nachoib.ovh* amb el projecte de "*Proves*". Si posem en marxa esta última aplicació, podem accedir a les dos webs per separat amb les corresponents URLS:

- *http://nachoib.ovh:3000*
- *http://proves.nachoib.ovh:3000*

Alternativament, podem *mapejar* Passenger amb esta última aplicació *Manager* únicament, i que siga ella la que s'encarregue de redirigir a l'una o l'altra app en funció del domini. Encara que, en este últim cas, potser és més còmode configurar *hosts* virtuals independents en Apache amb Passenger.

2.3. Automatitzar la posada en marxa del servidor Node amb *forever*

Com a alternativa a l'ús de *Passenger* amb Apache, podem optar per no instal·lar Apache i gestionar l'arrancada de les aplicacions Node amb la ferramenta *forever* perquè s'inicien de manera automàtica en iniciar el sistema. Per a això, seguim estos passos (suposant que els fem com a usuari *root*, en cas contrari haurem d'anteposar el comando `sudo`). Prendrem com a exemple l'aplicació *Manager* que hem fet en l'apartat anterior.

1. Instal·lem, si no ho tenim ja, el mòdul `forever` del repositori de NPM. Este mòdul permet rearrancar de manera automàtica una aplicació Node, fins i tot quan es produïx un error que la fa finalitzar.

```
npm install forever -g
```

2. Creem un script en la carpeta `/etc/init.d`, per exemple:

```
nano /etc/init.d/servidor-node
```

3. Editem l'arxiu amb este contingut (posant la carpeta on estarà el projecte Node amb el Manager, si no coincidix amb la que s'indica a continuació):

```
#!/bin/sh
FOREVER="/usr/bin/forever"
APP_PATH="/home/usuario/ProyectosNode/Manager/app.js"
USER="usuario"
su - $USER -c "$FOREVER start $APP_PATH"
```

> **Nota:** Recorda que has de canviar l'usuari i la ruta del projecte d'acord amb les teues necessitats. En el VPS de OVH l'usuari és "debian".

4. Fem el script executable:

```
chmod +x /etc/init.d/servidor-node
```

5. Escrivim este comando per a fer els canvis permanents en iniciar (és a dir, que amb cada inici del sistema s'arrancarà el manager):

```
update-rc.d /etc/init.d/servidor-node defaults
```

6. Després d'això, hem de posar en marxa el servidor (`/etc/init.d/servidor-node`) o reiniciar el servidor VPS perquè accepte els canvis introduïts.

2.4. Servidors de bases de dades

Mostrarem els passos per a instal·lar un servidor de base de dades de la nostra elecció. Veurem els casos de MariaDB i MongoDB.

2.4.1. MariaDB

Per a instal·lar MariaDB en el nostre servidor VPS, accedim per SSH com a usuari root i escrivim el següent comando:

```
apt update
apt install mariadb-server
```

> **NOTA**: si volem instal·lar MySQL en lloc de MariaDB, els passos a seguir són els mateixos, però en el comando anterior canviarem el paquet `mariadb-server` per `mysql-server` .

Després de la instal·lació, podem executar amb permisos de *root* el comando:

```
mysql_secure_installation
```

per a assegurar alguns aspectes que queden poc segurs en la instal·lació per defecte, com la contrasenya de root o l'accés remot o anònim.

- Podem establir la contrasenya de root en el primer pas d'este assistent
- En el següent pas (Remove anonymous users), convé respondre que sí (I)
- En el tercer pas (Disallow root login remotely), podem triar, encara que potser convé triar que sí (I)
- El següent pas (Remove test database and access to it), també podem respondre que sí (I)
- Finalment, recarreguem la taula de privilegis dels usuaris (I) i ja està llest.

Una vegada finalitzada esta configuració, per a iniciar, detindre o reiniciar el servidor, escriurem estos comandos, respectivament:

/etc/init.d/mysql start /etc/init.d/mysql stop /etc/init.d/mysql restart

2.4.2. MongoDB

Per a instal·lar MongoDB en VPS, el propi és instal·lar-ho com a servici. Els passos a seguir sempre serà millor consultar-los en la pàgina oficial. En el nostre cas, pots seguir estos passos: Instal·lació de l'Edició Community de MongoDB en Debian 11.

Després d'això, ja podem llançar i detindre Mongo amb el servici:

```
/etc/init.d/mongodb start
/etc/init.d/mongodb stop
/etc/init.d/mongodb restart
```

Habilitar connexions externes

Podríem utilitzar un client per a connectar al nostre servidor remot de MongoDB. Però abans hem d'habilitar MongoDB per a acceptar connexions remotes. Per a això, editem l'arxiu `/etc/mongodb.conf` i comentem esta línia:

```
#bind_ip = 127.0.0.1
```

Reiniciem el servici de MongoDB, i després iniciem el nostre client MongoDB (per exemple, el *plugin* de Visual Studio Code, o l'aplicació *Compass* oficial de Mongo) i creem una nova connexió al nostre servidor (per exemple, *vps112233.ovh.net*) pel port que siga. Recordem que el port per defecte és 27017, però es pot modificar amb la directiva `port` en l'arxiu de configuració anterior:

```
port = 27017
```

2.5. Instal·lar PHP i altres funcionalitats

Finalment, instal·larem els mòduls de PHP per a poder treballar tant des d'Apache com des de Nginx, si fora el cas. També instal·larem l'aplicació *phpMyAdmin* per a gestionar les bases de dades MariaDB de manera remota amb esta aplicació web.

2.5.1. Instal·lació de PHP

Explicarem a continuació com instal·lar PHP de manera global, i com configurar-la després per al cas concret de Nginx (la configuració amb Apache és immediata).

En primer lloc, hem d'executar els següents comandos per a descarregar PHP del repositori corresponent:

```
sudo apt install programari-properties-common
sudo add-apt-repository ppa:ondrej/php
sudo apt update
```

Després, instal·lem tant `php7.4` com `php7.4-fpm`, que és la ferramenta que utilitza Nginx per a tractament de pàgines PHP.

```
sudo apt install php7.4
sudo apt install php7.4-fpm
```

Podem consultar la versió instal·lada amb estos comandos, respectivament:

```
php -v
php-fpm7.4 -v
```

De manera similar a com hem instal·lat FPM, podem instal·lar altres extensions que puguen ser necessàries en un moment donat, fent el corresponent `sudo apt install php7.4-XXXX`, sent XXXX l'extensió en si (per exemple, `php7.4-common`, o `php7.4-mysql`). Encara que en principi no són necessàries per a la instal·lació de base.

Pel que fa a **Nginx**, ens faltaria habilitar PHP en ell. Este pas no és tan automàtic i senzill com en Apache, ja que Nginx no disposa de processament PHP natiu, i requerix d'uns passos a seguir, una vegada hem instal·lat prèviament el FPM (*FastCGI Process Manager*).

1. Obrim l'arxiu `/etc/php/7.x/fpm/php.ini` (sent la x la versió que tinguem instal·lada de PHP). Busquem la línia de configuració `cgi.fix_pathinfo`, i la deixem així, guardant l'arxiu en finalitzar:

```
cgi.fix_pathinfo=0
```

2. També pot resultar convenient, per problemes amb permisos, fer que s'atenguen les peticions PHP per TCP, i no a través d'un arxiu de socket especial que té php-fpm. Per a això, editem l'arxiu `/etc/php/7.x/fpm/pool.d/www.conf` i comentem esta línia i afegim la que va a continuació:

```
;listen = /run/php/php7.0-fpm.sock
listen = 127.0.0.1:9000
```

3. Reiniciem el processador PHP perquè accepte els nous canvis (substituïm novament la x per la versió que tinguem de PHP):

```
sudo systemctl restart php7.x-fpm
```

4. Indiquem a Nginx que les sol·licituds a pàgines PHP les passe al processador que hem instal·lat. Per a això, obrim l'arxiu de l'host virtual que vulguem configurar (per exemple, `/etc/nginx/sites-available/default`, o qualsevol altre que hàgem creat) i fem el següent:

- En primer lloc, afegir l'arxiu index.php com un dels possibles arxius d'inici:

```
index index.php index.html ...
```

- Descomentamos el següent bloc de directives, i el deixem com seguix (substituïm, de nou, la x per la nostra versió de PHP):

```
location ~ \.php$ {
 include /etc/nginx/fastcgi_params;
 fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
 fastcgi_pass 127.0.0.1:9000;
}
```

Després d'estos passos, reiniciem Nginx i provem de carregar algun contingut PHP bàsic, com l'anomenada a `phpinfo()`.

2.5.2. Administrar MySQL/MariaDB des de *phpMyAdmin*

La forma més senzilla de gestionar les nostres bases de dades MySQL (o MariaDB) potser és a través de l'aplicació *phpMyAdmin*, el client web per excel·lència per a gestionar les bases de dades d'este tipus. S'instal·la amb el següent comando (en mode *root* o amb permisos de superusuario):

```
apt-get install phpmyadmin
```

Durant la instal·lació, ens demanarà les següents dades:

- Si volem configurar algun servidor automàticament per a phpMyAdmin. Ens dona l'opció d'Apache (no la de Nginx), així que podem triar Apache perquè ho configure automàticament.
- Ens indicarà que necessita tindre una base de dades per defecte a la qual accedir,i que si crea la base de dades per defecte. Podem contestar que sí.
- Finalment, ens demanarà el password per a accedir a phpMyAdmin des del navegador, i que el confirmem després.

Després dels passos anteriors, ja podem accedir a phpMyAdmin amb el nom del nostre domini i el port pel qual estiga escoltant Apache. Per exemple:

```
http://vps112233.ovh.net/phpmyadmin
```

En el cas de voler utilitzar phpMyAdmin amb **Nginx**, podem fer un enllaç simbòlic de la carpeta d'instal·lació de phpMyAdmin (`/usr/local/phpmyadmin`) a la carpeta de la nostra aplicació web. Per exemple, si volem accedir des de la carpeta general d'aplicacions de Nginx:

```
ln -s /usr/share/phpmyadmin /var/www/html
```

Després, editem l'arxiu de configuració d'eixa carpeta (en el nostre exemple, seria l'arxiu `/etc/nginx/sites-available/default` , i afegim la configuració per a PHP, si no la té. També convindria deixar com a globals la ruta arrel de l'aplicació (*root*) i els arxius d'inici per defecte. Podria quedar una cosa així:

```
server {
   listen 80;
   server_name localhost;

   root /var/www/html;
   index index.html index.php;

   location / {
   }

   location ~ \.php$ {
   include /etc/nginx/fastcgi_params;
   fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
   fastcgi_pass 127.0.0.1:9000;
 }
 ...
```