**Manual de instalación y configuración de entorno de trabajo Plataforma de Gestión de Proyectos**

# ÍNDICE

[**Índice** 2](#_Toc27009)

[Precondiciones para realizar la instalación 4](#_Toc27010)

[Configurar servidor de PostgreSQL 4](#_Toc27011)

[Instalar dependencias del server de BD 4](#_Toc27012)

[Paso 1: Autenticarse a postgres y conectarse a la consola de pgsql 5](#_Toc27013)

[Paso 2: Ponerle contraseña al usuario “postgres” 5](#_Toc27014)

[Paso 3: Establecer las subredes que accederán al postgres usando md5 5](#_Toc27015)

[Paso 4: Configurar direcciones 5](#_Toc27016)

[Paso 5: Reiniciar el servidor del PostgreSQL 6](#_Toc27017)

[Optimización del PostgreSQL (opcional) 6](#_Toc27018)

[Paso 1: Cambiar algunas opciones básicas del archivo postgresql.conf 6](#_Toc27019)

[Paso:2 Reiniciar el servidor PostgreSQL 7](#_Toc27020)

[Instalación de Bases de datos sobre PostgreSQL 7](#_Toc27021)

[Paso1: Autenticarse a postgres, 7](#_Toc27022)

[Paso 2: Desconectar la consola del pgsql 7](#_Toc27023)

[Paso 3: Agregar el lenguaje PLR a la base de datos de la app 8](#_Toc27024)

[Paso 4: Solo si es necesario mantener los datos históricos: desde el usuario del postgres restaurar las bases de datos 8](#_Toc27025)

[Paso 5: Salir del usuario postgres Ctrl+D y reiniciamos el servicio 8](#_Toc27026)

[Servidor de aplicaciones con Apache 8](#_Toc27027)

[Pre-requisitos 8](#_Toc27028)

[Paso 1: Configurar repositorios del sistema operativo 8](#_Toc27029)

[Paso 2: Actualizar el sistema operativo con los los nuevos repositorios 9](#_Toc27030)

[Paso 3: Instalar dependencias para conectarse al servidor de base de datos 9](#_Toc27031)

[Paso 4: Instalar Apache y otras dependencias 9](#_Toc27032)

[Paso 5: Instalar Git 9](#_Toc27033)

[Paso 6: Instalar dependencias de la gema rmagick 10](#_Toc27034)

[Paso 7: Instalar RVM 10](#_Toc27035)

[Paso 8: Instalar ruby y ruby gems 10](#_Toc27036)

[Paso 9: Instalar ruby con rvm 10](#_Toc27037)

[Paso 10: Configurar Repositorio de gemas 11](#_Toc27038)

[Paso 11: Cambiar el gem sources del servidor 11](#_Toc27039)

[Paso 12: Instalar gemas bundler y passenger 11](#_Toc27040)

[Paso 14: Establecer el repo de gemas local 12](#_Toc27041)

[Paso 15: Instalar las gemas 12](#_Toc27042)

# PRECONDICIONES PARA REALIZAR LA INSTALACIÓN

1. Tener instalado en la computadora o servidor el SO Ubuntu 20.04 LTS. Aunque puede ser 18.04
2. Contar con un repositorio del Ubuntu 18.04 LTS de 32 ó 64 bit según lo requiera la arquitectura del sistema (opcional).
3. Paquetes necesarios, buscar en la carpeta adjunta.

*rvm-2.7.5*

ruby- 2.7.2 rubygems-3.3.5 Ficheros:

Fichero puma.rb

Fichero puma.service

1. Cumplir con el resto de los requisitos que se especificarán a continuación.

# Instalación del Sistema Operativo

### Paso 1: Configurar repositorios del sistema operativo

Ejecutar en terminal:

*$ sudo nano /etc/apt/sources.list*

Ejemplo de repositorios:

*deb http://ubuntu.uci.cu/ubuntu focal main restricted universe multiverse*

*deb http://ubuntu.uci.cu/ubuntu focal-security main restricted universe multiverse*

*deb http://ubuntu.uci.cu/ubuntu focal-updates main restricted universe multiverse*

*deb http://ubuntu.uci.cu/ubuntu focal-proposed main restricted universe multiverse*

*deb http://ubuntu.uci.cu/ubuntu focal-backports main restricted universe multiverse*

### Paso 2: Actualizar el sistema operativo con los nuevos repositorios

Ejecutar en terminal:

*$ sudo apt update*

*$ sudo apt upgrade*

### Paso 3: Instalar dependencias para conectarse al servidor de base de datos

Ejecutar en terminal:

*$ sudo apt install postgresql postgresql-contrib-12 postgresql-server-dev-12*

*$ sudo apt install build-essential libxslt1-dev libxml2 libxml2-dev curl apache2-dev libapr1-dev libaprutil1-dev g++ gcc autoconf automake bison libc6-dev libffi-dev libgdbm-dev libncurses5-dev libsqlite3-dev libtool libyaml-dev make pkg-config sqlite3 zlib1g-dev libgmp-dev libreadline-dev libssl-dev libcurl4 libcurl4-openssl-dev*

### Paso 4: Instalar Git

Ejecutar en terminal:

*$ sudo apt install git-core git-gui git-doc*

### Paso 5: Instalar dependencias generales

$ **sudo** apt-get install build-essential libmagick++-dev libxslt1-dev libxml2 libxml2-dev libcurl4openssl-dev libgmp3-dev freetds-dev git git-gui gitk

### Paso 6: Instalar dependencias para el plugin de gestión documental

On Debian use:

|  |  |
| --- | --- |
| |  | | --- | | $ **sudo** apt-get install libxapian-ruby1.9.1 xapian-omega libxapian-dev xpdf xpdf-utils antiword |   unzip\ catdoc libwpd-0.9-9 libwps-0.2-2 gzip unrtf catdvi djview djview3 uuid uuid-dev |

On Ubuntu use:

$ **sudo** apt-get install xapian-omega libxapian-dev xpdf antiword unzip catdoc libwpd-0.10-10 libwps-0.4-4 gzip unrtf catdvi djview djview3 uuid uuid-dev

On CentOS user:

|  |  |
| --- | --- |
| |  | | --- | | $ **sudo** yum install libxapian-ruby1.9.1 xapian-omega libxapian-dev xpdf antiword\ unzip catdoc |   libwpd-0.9-9 libwps-0.2-2 gzip unrtf catdvi djview djview3 uuid uuid-dev |

### Paso 7: Instalar dependencias necesarias para gema rmagick

*$ sudo apt install imagemagick libmagickcore-dev libmagickwand-dev*

# CONFIGURAR SERVIDOR DE POSTGRESQL

### Paso 1: Instalar dependencias del server de BD

$ **sudo** apt install postgresql-12 postgresql-contrib-12 r-base-core r-base r-base-dev postgresql-server-dev-12

### Paso 2: Autenticarse a postgres y conectarse a la consola de pgsql.

# su postgres

postgres$ psql

### Paso 3: Ponerle contraseña al usuario “postgres”.

postgres# ALTER ROLE postgres WITH PASSWORD 'postgres'; postgres# \q

postgres$ exit

***Nota: Cada una de las dos últimas líneas se pueden reemplazar por Ctrl + D.***

## Instalación de bases sobre posgresql

### Paso1: Autenticarse a postgres,

Conectarse a la consola de pgsql para crear usuarios, bases de datos asignar los permisos necesarios de los usuarios sobre las bases de datos.

# su postgres

postgres$ psql

postgres# CREATE USER redmine WITH PASSWORD 'led800lnx' CREATEDB;

postgres# CREATE DATABASE "redminedb" WITH OWNER = "redmine" ENCODING = 'UTF8';

postgres# CREATE DATABASE "redminedb\_dev" WITH OWNER = "redmine" ENCODING = 'UTF8';

postgres# CREATE DATABASE "redminedb\_test" WITH OWNER = "redmine" ENCODING = 'UTF8';

postgres# GRANT ALL ON DATABASE "redminedb" TO "redmine";

postgres# GRANT ALL ON DATABASE "redminedb\_dev" TO "redmine";

postgres# GRANT ALL ON DATABASE "redminedb\_test" TO "redmine";

### Paso 2: Desconectar la consola del pgsql.

postgres# \q

***Nota: La línea se puede reemplazar por Ctrl + D.***

### Paso 3: Solo si es necesario mantener los datos históricos: desde el usuario del postgres restaurar las bases de datos.

postgres$ pg\_restore -h localhost -U redmine -d redminedb -O <

/directorio/al/proerp\_db.backup

### Paso 4: Salir del usuario postgres Ctrl+D y reiniciamos el servicio.

# invoke-rc.d postgresql restart

# INSTALACIÓN DE LAS DEPENDENCIAS PARA DESARROLLO

Ubicar el directorio con el paquete RVM Paso 1: Instalar RVM y Dependencias

### Paso 1 Copiar *rvm-1.29.12*.zip para el home del usuario y descompactar.

Se muestran dos variantes

$ tar --strip-components=1 -xzf *rvm-1.29.12.*tar.gz

$ unzip *rvm-1.29.12.*zip en caso de estar en .zip

### Paso 2 Entrar al directorio *rvm-1.29.12* y ejecutar

$ cd ~/rvm-1.29.12/

$ ./install --auto-dotfiles

### Paso 3 Arrancar el RVM (para que corra el ruby)

$ source /home/usuario/.rvm/scripts/rvm

|  |
| --- |
| Paso 4 Ir a las opciones de la terminal terminal > edit > profile preferences > Command > Run command as login shell. Paso 5 Cerrar terminal y volver a abrir. Comprobar instalación con rvm -v. $ rvm -v  Paso 6 Copiar contenido de carpeta rvm/archives del paquete de instalación para /home/usuario/.rvm/archives.  $ cp /directorio/al/rvm/archives/ /home/pc\_name/.rvm/archives/ Aca se deben copiar para el rvm los ficheros de  ruby-x.x.x.tar.gz rubygems-x.x.x.tgz  Por ejemplo:  ruby-2.7.5.tar.bz2 rubygems-3.3.5.tgz Paso 7 Instalar Ruby ejecutar en la terminal: $ sudo apt install ruby-full  $ rvm install 2.7.5 --rubygems 3.3.5 |

PROYECTOS

# GESTIÓN Y CONFIGURACIÓN DEL REPOSITORIO DE GEMAS

### Paso 1: Ubicar el directorio con el repositorio de gemas

Copiar el repositorio local de gemas(repo) para el *home* del usuario.

### Paso 2: Pasos para que el rvm reconozca las gemas

Especificando: global o default y dónde instalarlas. Comprobar listando las gemas reconocidas por rvm.

$ source ~/.rvm/scripts/rvm

$ rvm use *2.7.5*

$ rvm gemset use default

$ rvm gemset list

### Paso 3: Pasos para identificar las versiones instaladas de ruby y actualización del bundle

$ source ~/.rvm/scripts/rvm

$ rvm cleanup all

$ rvm reload

$ source ~/.rvm/scripts/rvm

$ bundle platform --ruby

$ bundle update  *#esto actualzia el bundle*

$ bundle update  *--ruby #actualzia la version de ruby*

### Paso 4: Opcional reemplazar las gemas de un gemset

Si desea reemplazar las gemas de un gemset determinado se debe eliminar las gemas:

$ rvm gemset empty

### Paso 5: Configurar Repositorio de gemas (no es necesario si tiene plena conexión a Internet)

Copiar repositorio de gemas **repo.tar.gz** para /home/user/

Ejecutar en terminal situado en /home/user para descomprimir:

*$ tar -xf repo.tar.gz*

### Paso 6: Cambiar el gem sources del servidor (no es necesario si tiene plena conexión a Internet)

Ejecutar en terminal:

Verificar los REPOS antes instalados, actualizar los mismos y comprobar

|  |
| --- |
| $ gem source // Listar REPO de gemas disponibles  $ gem source --remove  [https://rubygems.org /](https://rubygems.org/)  //eliminando REPO no deseado  -Caso de la UCI gem source --remove http://nexus.prod.uci.cu/repository/rubygems-all/  $ gem source --add file:/directorio/al/repo/ //adicionar el directorio de gemas ubicado dentro del repositorio local de gemas(repo)  -Caso de la UCI  gem source --add http://nexus.prod.uci.cu/repository/rubygems-all/    $ gem source //comprobar/ Listar REPO de gemas disponibles    -Configurar un mirror de la dirección del Gemfile (No se modifica la dirección original del Gemfile pero actualiza desde el mirror --solo para bundle install)  $ bundle config mirror.https://rubygems.org/ file:/home/usuario/repo  $ bundle config mirror.https://rubygems.org/ http://nexus.prod.uci.cu/repository/rubygems-all |

### Paso 7: Ubicarse en la dirección del directorio de gemas

Situarse en la dirección del directorio de gemas ubicado dentro del repositorio local de gemas(repo)

|  |
| --- |
| $ cd /directorio/al /repo/gems |

### Paso 8: Activar el RVM.

$ source ~/.rvm/scripts/rvm

### Paso 8: Instalar las gemas relacionadas a bundler

Instalar las gemas relacionadas a bundler (DEMORA mucho más si se hace como root). Para entorno de despliegue del producto instalar sin la documentación $ gem install --ri –Doc bundler (MUCHO más rápido)

$ gem install bundler

$ gem install builder

### Paso 9: Reindexar las gemas incorporadas

Ubicarse dentro del directorio repositorio de gemas(repo) y reindexar las gemas incorporadas.

$ gem generate\_index

***Nota: Cada vez que se incorporen nuevas gemas hay que repetir el reindexado***

# INSTALACION DEL ENTORNO DE TRABAJO

### Paso 1: Ubicarse en el directorio.

$ cd /home/proerp

### Paso 2: Copiar el código de la app para la pc Generar el fichero secret Token

Generar el fichero secret Token solo si no existe ya dentro de la carpeta inicializers de la carpeta config dentro del directorio que contiene el código de la aplicación.

### **Paso 3: Copiar el código de la aplicación**

Suponiendo que la carpeta del código sea /var/www/ y configurar como sigue:

Copiar para /var/www/proerp/config los ficheros **configuration.yml** y **database.yml** y personalizarlo con las configuraciones de la organización, o modificar los que contienen ‘example’ en el nombre.

Ejecutar en terminal:

*$ cd /var/www/proerp*

*$ sudo chown user:user -R /var/www/proerp*

*$ sudo chmod 755 -R /var/www/proerp*

### Paso 4: Configurar el database.yml

Ubicar el fichero database.yml dentro de la carpeta config dentro del directorio que contiene código de la aplicación**.**

|  |
| --- |
| $ nano /directorio/al/codigo\_app/config/database.yml |

Ejemplo: Configuración del database.yml

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **production**: adapter: postgresql database: redminedb host: localhost username: redmine password: led800lnx encoding: utf8 pool: 5

**development:**

adapter: postgresql database: redminedb\_dev host: localhost username: redmine password: led800lnx encoding: utf8 pool: 5

**test:**

adapter: postgresql

database: redminedb\_test host: localhost username: redmine password: led800lnx encoding: utf8 pool: 5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# PASOS PARA CREAR LA LLAMADA CADA VEZ QUE SE REINICIE EL SERVIDOR

### Paso 1: Verificar rvm:

Instalar rvm 1.29.12 para que el fichero pumactl se ejecute correctamente

Descompactar el rvm*-*1.29.12

Entrar al directorio *rvm-1.29.12* y ejecutar

$ ./install --auto-dotfiles

### Paso 2: Crear fichero puma.rb en el config/ de la aplicación proerp.

#workers Integer(ENV['WEB\_CONCURRENCY'] || 2)

#threads\_count = Integer(ENV['RAILS\_MAX\_THREADS'] || 5)

#threads threads\_count, threads\_count #preload\_app!

#rackup DefaultRackup

#port ENV['PORT'] || 3000

#environment ENV['RACK\_ENV'] || 'development'

#on\_worker\_boot do

# Worker specific setup for Rails 4.1+

# See: https://devcenter.heroku.com/articles/deploying-rails-applications-with-the-puma-webserver#on-worker-boot

# ActiveRecord::Base.establish\_connection #end

# Change to match your CPU core count workers Integer(ENV['WEB\_CONCURRENCY'] || 2)

# Min and Max threads per worker threads 1, 6

app\_dir = File.expand\_path("../..", \_\_FILE\_\_) shared\_dir = "#{app\_dir}/shared"

# Default to production

rails\_env = ENV['RAILS\_ENV'] || "production"

#daemonize true environment rails\_env

# Set up socket location

bind "unix://#{shared\_dir}/sockets/puma.sock"

# Logging

stdout\_redirect "#{shared\_dir}/log/puma.stdout.log", "#{shared\_dir}/log/puma.stderr.log", true

# Set master PID and state locations pidfile "#{shared\_dir}/pids/puma.pid" state\_path "#{shared\_dir}/pids/puma.state" activate\_control\_app

on\_worker\_boot do require "active\_record"

ActiveRecord::Base.connection.disconnect! rescue ActiveRecord::ConnectionNotEstablished

ActiveRecord::Base.establish\_connection(YAML.load\_file("#{app\_dir}/config/database.yml")

[rails\_env])

end

### Paso 3: Ejecutar la aplicación

$ bundle exec puma -e production -C config/puma.rb

### Paso 4: Copiar y editar el fichero puma.service en el directorio /etc/systemd/system

Se deben editar en el fichero los siguientes elementos:

* ruta para el wrapper;
* ruta de la aplicacion;
* ruta para la configuracion del puma; - usuario para ejecutar el puma.

***Se debe tomar el fichero ejemplo en la carpeta adjunta a este manual “systemdpuma-unit”***

[Unit]

Description=Puma web server

After=syslog.target

After=network.target

[Service]

# Foreground process (do not use --daemon in ExecStart or config.rb)

Type=simple

# Preferably configure a non-privileged user

User= usuario

# Specify the path to your puma application root

WorkingDirectory=/home/usuario/proerp

# The commands to start/stop/reload Puma

ExecStart=/home/usuario/.rvm/bin/proerp\_pumactl -F config/puma.rb start

ExecStop=/home/usuario/.rvm/bin/proerp\_pumactl -F config/puma.rb stop

ExecReload=/home/usuario/.rvm/bin/proerp\_pumactl -F config/puma.rb phased-restart

TimeoutSec=15

Restart=always

[Install]

WantedBy=multi-user.target

### Paso 5: Desde la Aplicación, crear un wrapper para levantar la aplicación:

$ rvm wrapper 2.7.2@default proerp pumactl

Esto genera un fichero wrapper en la dirección /home/usuario/.rvm/bin/proerp\_pumactl

### Paso 6: Después de instalar o hacer cambios en fichero puma.service ejecutar:

$ sudo systemctl daemon-reload

$ sudo systemctl stop puma.service

$ sudo systemctl enable puma.service

$ sudo systemctl start puma.service

$ sudo systemctl status puma.service

$ sudo /etc/init.d/nginx restart

### Paso 7: Cargar tipos de datos y funciones de la aplicación

Para desarrollo

$ sudo invoke-rc.d postgresql restart

$ rake db:migrate

$ rake redmine:plugins:migrate

Para producción

$ rake db:migrate RAILS\_ENV=production

$ rake redmine:plugins:migrate RAILS\_ENV=production

### Paso 8: Confirmar que la app está corriendo

$ rails server

# MONTAJE DEL ENTORNO DE DESARROLLO

### Paso 1: Instalar paquete del git:

$ sudo apt install git-svn git-core git-doc gitg gitk meld

### Paso 2: Instalar IDE de desarrollo

2.1 Instalar dependencias del IDE de desarrollo

$ sudo apt install openjdk-8-jdk openjdk-8-doc

Copiar el IDE para desarrollo en la carpeta deseada. Recomendable home. Ubicarse desde la consola dentro de la carpeta IDE de desarrollo e instalar. Desde bin correr:

$ ./rubymine.sh

### Paso 3: Abrir el IDE de desarrollo en su menú principal

**Configurar IDE desarrollo**

Ubicarse en: File/Default settings/Languaje &Frameworks/Ruby SDK and Gems/ y habilitar

Ruby 2.7.5

Ubicarse en: Run/Edit configuration/ radio button “Ruby other SDK” seleccionar Ruby 2.7.5

### Paso 4: Instalar gemas

Ubicarse en el directorio que contiene el código de la aplicación e Instalar gemas

|  |
| --- |
| $ cd /directorio/al/codigo\_app/  $ source ~/.rvm/scripts/rvm  $ bundle install  *## Para resolver problema del debug ## Si ocurre*  *$ gem install debase --pre* |

*$ gem install ruby-debug-ide*

***Nota: este paso demora unos minutos***

### Paso 5: Adicionar las gemas necesarias para el ngnix y el puma en la carpeta de las gemas de la aplicación proerp (ya están en el directorio /repo/gems ).

### Paso 6: Correr los migrates

Ejecutar en terminal:

*$ rake db:migrate*

*$ rake redmine:plugins:migrate*

# MONTAJE DE APLICACIÓN PARA DESPLIEGUE NGINX

Pasos para la instalación y configuración de la aplicación web ligera nginx con la gema unicorn para el uso de proxy inverso para la publicación de Ruby on Rails**.**

### Paso 1: Instalar nginx.

$ sudo apt-get install nginx

### Paso 2: Ir al directorio de la aplicación y crear las siguientes carpetas.

$ mkdir -p shared/pids shared/sockets shared/log

Paso 3: Adicionar al final del fichero /etc/nginx/sitesavailable/default el siguiente texto**:**

$ sudo gedit /etc/nginx/sites-available/default

***Nota: en amarillo se debe especificar según el nombre real del home y de la carpeta donde está la aplicación.***

Lo que está en verde es la direccion URL por la cual se publicará la aplicación.

upstream proerp {

# Path to PUMA SOCK file, as defined previously server unix:/home/usuario/proerp/shared/sockets/puma.sock fail\_timeout=0;

}

server { listen 81; server\_name localhost;

# Application root, as defined previously

root /home/usuario/proerp/public;

try\_files $uri/index.html $uri @proerp;

location @proerp { proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for; proxy\_set\_header Host $http\_host; proxy\_redirect off;

proxy\_pass http://proerp;

}

error\_page 500 502 503 504 /500.html; client\_max\_body\_size 4G;

keepalive\_timeout 10;

**}**