

Instalación en Ubuntu Server 22.04 LTS

“ Los pasos que se describen han sido efectuados sobre a una instalación limpia del sistema operativo Ubuntu Server 22.04.2 LTS con una configuración de 2GB de ram y 2 procesadores 64bit. La elección del sistema operativo como la configuración de hardware puede variar de acuerdo a los requerimientos de su organización.

Puesta a punto del sistema operativo

Comenzaremos con la instalación del software necesario para el correcto funcionamiento de plataforma de acuerdo a los [requisitos que tiene Decidim](#).

Aquí un resumen de software y librerías que instalaremos:

- Paquetes varios
- PostgreSQL 14.6
- Ruby 3.0.5
- NodeJS 16.19.1
- Npm 8.19.3
- ImageMagick

Tenga en cuenta que la presente guía ha sido realizada con un usuario llamado “decidim”, con permisos de administrador, los pasos se realizan desde el home del usuario y la instalación de la plataforma se realiza sobre el directorio “/home/decidim/decidim-app”.

Para crear dicho usuario y asignarle los permisos requeridos, puede utilizar los siguientes comandos:

```
adduser decidim  
usermod -aG sudo decidim
```

Actualización y configuración SO

Es importante tener el sistema operativo actualizado, por lo que comenzaremos actualizando la lista de paquetes de nuestro repositorio y aplicaremos las actualizaciones disponibles, al final limpiaremos los paquetes innecesarios.

```
sudo apt update  
sudo apt upgrade  
sudo apt autoremove
```

Configura el timezone del servidor para prevenir problemas con la hora de tu servidor.

```
sudo dpkg-reconfigure tzdata
```

Aquí tendrás que seleccionar la zona horaria que se ajusta a tu ubicación geográfica

Instalación de paquetes

Comenzamos con la instalación de algunos paquetes necesarios:

```
sudo apt install -y autoconf bison build-essential libssl-dev libyaml-dev libreadline6-dev zlib1g-dev libncurses5-  
dev libffi-dev libgdbm-dev imagemagick libicu-dev
```

Instalación de Ruby

Continuaremos con la instalación de ruby usando el método [rbenv](#) el cuál nos permitirá manejar distintas versiones.

Ejecuta los siguientes comandos para instalar rbenv:

```
git clone https://github.com/rbenv/rbenv.git > ~/.rbenv  
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc  
echo 'eval "$(rbenv init -)"' >> ~/.bashrc  
source ~/.bashrc
```

Ahora deberías comprobar si tienes rbenv correctamente instalado, ejecutando el comando **type rbenv** debería darte esta respuesta:

```
decidim@decidim:~$ type rbenv
rbenv is a function
rbenv ()
{
    local command;
    command="${1:-}";
    if [ "$#" -gt 0 ]; then
        shift;
    fi;
    case "$command" in
        rehash | shell)
            eval "$(rbenv "sh-$command" "$@" )"
            ;;
        *)
            command rbenv "$command" "$@"
            ;;
    esac
}
```

Es momento de instalar Ruby, para lo cuál haremos uso de ruby-build para simplificar su instalación.

```
git clone https://github.com/rbenv/ruby-build.git > ~/.rbenv/plugins/ruby-build
```

Ahora deberías poder ejecutar el comando **rbenv install -l** que te listará todas las versiones de ruby disponibles.

```
decidim@decidim:~$ rbenv install -l
2.7.7
3.0.5
3.1.3
3.2.1
jruby-9.4.1.0
mruby-3.1.0
picoruby-3.0.0
truffleruby-22.3.1
truffleruby+graalvm-22.3.1
```

Usaremos la versión 3.0.5, por lo que ejecutaremos los siguientes comandos:

```
rbenv install 3.0.5  
rbenv global 3.0.5
```

Verificaremos que tenemos todo en orden ejecutando el comando ***ruby -v***:

```
decidim@decidim:~$ ruby -v  
ruby 3.0.5p211 (2022-11-24 revision ba5cf0f7c5) [x86_64-linux]
```

Si salió todo bien, ahora necesitaremos configurar Gems, el gestor de paquetes para Ruby.

Para configurar Gems ejecutamos los siguientes comandos:

```
echo "gem: --no-document" > ~/.gemrc  
gem install bundler
```

Podemos volver a comprobar que todo marcha bien ejecutando el comando ***gem env home***:

```
decidim@decidim:~$ gem env home  
/home/decidim/.rbenv/versions/3.0.5/lib/ruby/gems/3.0.0
```

Con estos pasos ya tenemos instalado ruby en nuestro servidor.

Instalación de PostgreSQL

Decidim utiliza PostgreSQL como base de datos sql por lo que instalaremos los paquetes necesarios en nuestro servidor:

```
sudo apt install -y postgresql libpq-dev
```

Podemos comprobar la instalación de Postgresql ejecutando el siguiente comando:

```
decidim@decidim:~$ psql -V  
psql (PostgreSQL) 14.6 (Ubuntu 14.6-0ubuntu0.22.10.1)
```

Instalación de NodeJS, NPM y YARN

Ahora instalaremos NodeJS lo cual haremos utilizando el gestor de versiones node ***nvm***. Una vez instalado nvm procedemos a instalar node con el comando ***nvm install <node-version>*** lo que instalará node y npm.

```
curl https://raw.githubusercontent.com/creationix/nvm/master/install.sh | bash
source ~/.profile
nvm install 16.19.1
```

Podemos comprobar la instalación de NodeJS y NPM ejecutando los siguientes comandos:

```
decidim@decidim:~$ node --version
v16.19.1

decidim@decidim:~$ npm --version
8.19.3
```

Una vez instalado node y npm es momento de proceder a instalar yarn que luego será utilizado por el generador de Decidim.

```
npm install --global yarn
```

Instalar Decidim

Generar la aplicación

Con el software necesario instalado podemos proceder a crear nuestra aplicación Decidim utilizando su generador.

Lo primero que haremos es instalar la Gema decidim:

```
gem install decidim
```

Luego ejecutaremos el generador para crear nuestra aplicación:

```
decidim decidim-app
```

En este punto tendremos creado un directorio en ~/decidim-app con el código de decidim.

Terminada la tarea del generador vamos a realizar algunas configuraciones para poner en funcionamiento nuestra aplicación Decidim.

Conexión a base de datos

Comenzaremos **creando un usuario para la conexión a la base de datos**. Para esto ejecutaremos el siguiente comando:

```
sudo -u postgres psql -c "CREATE USER decidim_app WITH SUPERUSER CREATEDB NOCREATEROLE PASSWORD 'Password1'"
```

Con este comando crearemos un usuario llamado `decidim_app` con la contraseña `Password1`.
Utiliza tu propio usuario y contraseña segura.

Configuración de Gemas

Una vez que contamos con las credenciales creadas debemos configurar Decidim para hacer uso de las mismas. Para almacenar esta información, que es sensible, haremos uso de un archivo de configuración YAML que ubicaremos fuera del código de la aplicación.

Para traducir este archivo al sistema config de Decidim, vamos a incluir la gema de Ruby "figaro" en nuestra aplicación que se encargará de ello.

En este momento configuraremos algunas gemas adicionales que también utilizaremos en nuestra instancia de Decidim: `whenever`, `passenger`, `delayed_job_active_record` y `daemons`.

Primero nos movemos al directorio donde se encuentra decidim:

```
cd ~/decidim-app
```

Seguido, editamos el archivo Gemfile:

```
nano Gemfile
```

Agregamos las siguientes líneas antes de la primera declaración `group`:

```
gem "figaro"  
gem "whenever", require: false
```

Ahora dentro del grupo `production` agregaremos las siguientes líneas:

```
gem "passenger"  
gem 'delayed_job_active_record'  
gem "daemons"
```

Quedando el grupo production de la siguiente manera:

```
group :production do
  gem "passenger"
  gem 'delayed_job_active_record'
  gem "daemons"
end
```

Guardamos el archivo y salimos. (Ctrl + x luego y)

El archivo Gemfile debería verse como éste:

```
decidim@decidim:~/decidim-app$ cat Gemfile
# frozen_string_literal: true

source "<https://rubygems.org>"

ruby RUBY_VERSION

gem "decidim", "0.27.2"
# gem "decidim-conferences", "0.27.2"
# gem "decidim-consultations", "0.27.2"
# gem "decidim-elections", "0.27.2"
# gem "decidim-initiatives", "0.27.2"
# gem "decidim-templates", "0.27.2"

gem "bootsnap", "~> 1.3"

gem "puma", ">= 5.0.0"

gem "faker", "~> 2.14"

gem "wicked_pdf", "~> 2.1"

gem "figaro"
gem "whenever", require: false

group :development, :test do
  gem "byebug", "~> 11.0", platform: :mri

  gem "brakeman"
```

```
gem "decidim-dev", "0.27.2"
end

group :development do
  gem "letter_opener_web", "~> 2.0"
  gem "listen", "~> 3.1"
  gem "spring", "~> 2.0"
  gem "spring-watcher-listen", "~> 2.0"
  gem "web-console", "~> 4.2"
end

group :production do
  gem "passenger"
  gem 'delayed_job_active_record'
  gem "daemons"
end
```

Ahora actualizaremos nuestra aplicación para que incluya las gemas adicionales. Para ello ejecutamos el siguiente comando:

```
cd ~/decidim-app
bundle install
```

Archivo de configuración

En este punto crearemos el archivo de configuración donde incluiremos las credenciales de acceso a la base de datos (creadas más arriba) y una clave secreta que usara Decidim para encriptar la cookie y otros temas de seguridad.

Éste archivo servirá para cargar cualquier variable de entorno de Decidim. Puede consultar las mismas [aquí](#).

Para generar la clave secreta usaremos el siguiente comando:

```
cd ~/decidim-app
rake secret
```

Esto nos devolverá una cadena (hay que copiarla) que utilizaremos en nuestro archivo de configuración.


```
nano ~/decidim-app/config/application.yml
```

Agregamos las siguientes líneas:

```
DATABASE_URL: postgres://decidim_app:Password1@localhost/decidim_production
```

```
SECRET_KEY_BASE:
```

```
b882695289c20ac4ff02869a8d787f836421cf4bbf7fc461ac149712a2bf5a343437c319654668c111f8de4de782c5  
ddf06a5c741513c0328926c1a0f96979e0
```

Tenga en cuenta usar el nombre de la base de datos la credencial y el secret creados anteriormente.

Compilación de assets y generación de base de datos

Es momento de utilizar algunos comandos de rails para crear la base de datos, ejecutar las migraciones necesarias y compilar los assets necesarios para el funcionamiento de Decidim.

```
cd ~/decidim-app  
yarn  
bin/rails db:create RAILS_ENV=production  
bin/rails assets:precompile db:migrate RAILS_ENV=production
```

La salida del primer comando deberá ser similar a la siguiente:

```
decidim@decidim:~/decidim-app$ bin/rails db:create RAILS_ENV=production  
Created database 'decidim_production'
```

El segundo comando compilará los assets y ejecutará las migraciones necesarias para inicializar la base de datos que utilizará la aplicación.

Alta de usuario administrador

Para poder loguearnos por primera vez a nuestra aplicación y empezar a hacer uso de la misma debemos contar con un usuario administrador (sysadmin).

Tenga en cuenta que este usuario se diferencia de los usuarios administradores de las organizaciones que forman parte de Decidim. Puede conocer [más aquí](#).

Para crear el usuario entraremos en la consola Rails de nuestra aplicación Decidim y crearemos el primer usuario administrador.

```
bin/rails console -e production
```

Aparecerá un prompt con el siguiente mensaje:

```
Loading production environment (Rails 6.1.6)
irb(main):001:0>
```

Aquí debemos escribir las siguientes líneas presionando Enter al final de cada una.

```
email = "admin@admin.com"
password = "<contraseña segura>"
user = Decidim::System::Admin.new(email: email, password: password, password_confirmation: password)
user.save!
```

Escribe quit para salir.

Esta será la credencial “system admin” de acceso a tu aplicación Decidim. Usa tu correo y una contraseña segura.

Si todo ha ido bien hasta aquí, deberíamos contar con una instalación básica de Decidim.

Instalación y configuración de servidor web

Es momento de configurar un servidor web que gestione las peticiones de los usuarios. Para ello haremos uso de Nginx como servidor web.

Instalación Nginx

Podemos instalar nginx y habilitar las reglas de firewall con los siguientes comandos:

```
sudo apt -y install nginx
sudo ufw allow http
sudo ufw allow https
```

En caso de utilizar algún otro firewall deberá agregar las excepciones para los servicios http y https.

Instalación y configuración de Passenger

Ahora instalaremos **Passenger** que nos servirá de pasarela para ejecutar la aplicación Ruby.

```
# Install our PGP key and add HTTPS support for APT
sudo apt-get install -y dirmngr gnupg apt-transport-https ca-certificates curl

curl https://oss-binaries.phusionpassenger.com/auto-software-signing-gpg-key.txt | gpg --dearmor | sudo tee
/etc/apt/trusted.gpg.d/phusion.gpg >/dev/null

# Add our APT repository
sudo sh -c 'echo deb https://oss-binaries.phusionpassenger.com/apt/passenger jammy main >
/etc/apt/sources.list.d/passenger.list'
sudo apt-get update

# Install Passenger + Nginx module
sudo apt-get install -y libnginx-mod-http-passenger
```

Lo que sigue es activar Passenger con la siguiente línea:

```
if [ ! -f /etc/nginx/modules-enabled/50-mod-http-passenger.conf ]; then sudo ln -s /usr/share/nginx/modules-
available/mod-http-passenger.load /etc/nginx/modules-enabled/50-mod-http-passenger.conf ; fi
```

Podemos comprobar que se creó correctamente ejecutando el comando **ls**:

```
decidim@decidim:~/decidim-app$ sudo ls /etc/nginx/conf.d/mod-http-passenger.conf
/etc/nginx/conf.d/mod-http-passenger.conf
```

Reiniciamos el servidor Nginx

```
sudo service nginx restart
```

Ahora podemos ejecutar el comando ***passenger-config validate-install*** y comprobar que todo marcha bien.

```
decidim@decidim:~/decidim-app$ sudo /usr/bin/passenger-config validate-install
```

```
What would you like to validate?
```

```
Use <space> to select.
```

```
If the menu doesn't display correctly, press '!'
```

```
▶ ● Passenger itself
```

```
○ Apache
```

```
-----  
  
* Checking whether this Passenger install is in PATH... ✓
```

```
* Checking whether there are no other Passenger installations... ✓
```

```
Everything looks good. :-)
```

Es momento de editar la configuración de Passenger para indicar la ruta a la instalación de Ruby que efectuamos anteriormente. Para ello editamos el archivo `/etc/nginx/conf.d/mod-http-passenger.conf`

```
sudo nano /etc/nginx/conf.d/mod-http-passenger.conf
```

Aquí debemos asegurarnos que `passenger_ruby` apunta a nuestra carpeta ruby `.rbenv`.

```
### Begin automatically installed Phusion Passenger config snippet ###
```

```
passenger_root /usr/lib/ruby/vendor_ruby/phusion_passenger/locations.ini;
```

```
passenger_ruby /home/decidim/.rbenv/shims/ruby;
```

```
passenger_instance_registry_dir /var/run/passenger-instreg;
```

```
### End automatically installed Phusion Passenger config snippet ###
```

Para evitar un **posible error** futuro, relacionado el servicio Passenger y con el uid del usuario que se ejecuta (por defecto `www-data`), es necesario modificar los permisos del directorio `/home/decidim`, para realizar ello deberemos hacer

```
cd /home
```

```
sudo chmod g+x,o+x decidim/
```

Configuración host virtual en Nginx

Una vez instalados Nginx y Passenger, configuraremos nginx para que dirija las peticiones http(s) a nuestra copia de Decidim. Para ello, tenemos que crear un archivo de configuración de Nginx y configurar un nuevo host virtual con nuestro dominio (por ejemplo: my-decidim.org):

```
sudo nano /etc/nginx/sites-available/decidim
```

Copia el siguiente contenido dentro del archivo.

```
server {  
    listen 80;  
    listen [::]:80;  
  
    server_name my-decidim.org;  
    client_max_body_size 32M;  
  
    passenger_enabled on;  
    passenger_ruby /home/decidim/.rbenv/shims/ruby;  
  
    rails_env    production;  
    root        /home/decidim/decidim-app/public;  
}
```

Reemplaza **my-decidim.org** por el dominio que usarás para tu instancia de Decidim.

Tenga en cuenta que deberá realizar las acciones necesarias en su dns para que su dominio apunte correctamente al servidor donde se encuentra la instalación de Decidim.

Si tu aplicación funcionará detrás de un proxy reverso [consulta aquí](#).

Creamos un enlace simbólico para habilitar el nuevo sitio.

```
sudo ln -s /etc/nginx/sites-available/decidim /etc/nginx/sites-enabled/
```

Reinicia el servicio nginx:

```
sudo service nginx restart
```

Configuración SSL

Llegamos al momento en donde debemos generar un certificado ssl para el acceso a nuestro servidor web, ya que Decidim por defecto redirecciona a https.

Aquí se presentan distintas alternativas, la que detallamos a continuación describe obtener un certificado gratuito por medio de [Let's Encrypt](#). Usted puede utilizar certificados de su CA de confianza.

Para instalar un certificado Let's Encrypt utilizaremos una herramienta llamada [certbot](#) que además de generar el certificado hará la tarea de actualizarlo automáticamente.

Aquí dejamos un resumen de los [pasos indicados en la web de cerbot](#) para instalarlo en nuestro servidor.

```
sudo snap install core; sudo snap refresh core
sudo snap install --classic certbot
sudo ln -s /snap/bin/certbot /usr/bin/certbot
sudo certbot --nginx
```

El último comando ingresado nos pedirá que ingresemos una dirección de correo donde se nos notificará si existe algún problema con la renovación automática del certificado. Además, nos pedirá que aceptemos los términos y condiciones y por último que seleccionemos el dominio sobre el que queremos generar el certificado.

En este punto debería estar en condiciones de acceder a su instancia de Decidim ingresando por un navegador a su dominio por ej: **<https://my-decidim.org>**

Configuraciones adicionales y primer acceso

Configuraciones de envío de correos

Antes de continuar con el acceso y administración de la plataforma es buen momento para que configuremos el servicio de envíos de correos de Decidim para no tener problemas con las validaciones de los usuarios que se vayan a crear en la plataforma, ya que Decidim envía un correo con un link de activación al momento de crear usuarios.

Para enviar los correos electrónicos tendremos que añadir un procesador a Ruby on Rails que se encargue de ello. Para esto debemos configurar la gema ***delayed_jobs*** configurada previamente.

Para ello saremos los siguientes comandos:

```
cd ~/decidim-app
echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf && sudo sysctl -p

bundle install
bin/rails generate delayed_job:active_record
bin/rake db:migrate
```

Iniciaremos el proceso `delayed_job` con el siguiente comando:

```
RAILS_ENV=production bin/delayed_job restart
```

Ahora debemos asegurarnos de que *delayed_job* se inicia al reiniciar el sistema. Para ello podemos crear un script personalizado en el crontab de nuestro sistema.

```
cd ~/decidim-app
nano config/delayed_job_cron.sh
```

Puedes usar este script que se encargará de iniciar el proceso.

```
#!/bin/bash

export PATH="$HOME/.rbenv/bin:$PATH"
eval "$(rbenv init -)"
APP_PATH="$HOME/decidim-app"

if ! [ -s $APP_PATH/tmp/pids/delayed_job.pid ]; then
  RAILS_ENV=production $APP_PATH/bin/delayed_job start
fi
```

Le daremos permiso de ejecución a nuestro script.

```
chmod +x config/delayed_job_cron.sh
```

Por último lo agregaremos a nuestro crontab.

```
crontab -e
```

Abierto nuestro crontab agregamos una línea al final del mismo

```
*/5 * * * * /home/decidim/decidim-app/config/delayed_job_cron.sh
```

Finalmente reiniciamos Passenger con el comando:

```
sudo passenger-config restart-app ~/decidim-app
```

Configuración de cron jobs

Decidim necesita ejecutar algunas tareas programadas para su funcionamiento. Para programar las mismas utilizaremos la gema Whenever.

Necesitaremos crear el archivo cron schedule que la gema utilizará para saber cuándo ejecutar la tarea en segundo plano. Basta con crear un nuevo archivo en config/schedule.rb con este contenido:

```
nano ~/decidim-app/config/schedule.rb
```

```
env :PATH, ENV['PATH']

every :day, at: '12:00am' do
  rake "decidim:delete_download_your_data_files"
end

every :day, at: '12:01am' do
  rake "decidim:metrics:all"
end

every :day, at: '12:05am' do
  rake "decidim:open_data:export"
end

every :day, at: '12:06am' do
  rake "decidim_meetings:clean_registration_forms"
end

every :day, at: '12:07am' do
  rake "decidim:reminders:all"
end
```



```
every :day, at: '07:00am' do
  rake "decidim:mailers:notifications_digest_daily"
end

every :saturday, at: '7:00 am' do
  rake "decidim:mailers:notifications_digest_weekly"
end
```

Podemos comprobar la sintaxis y que se está leyendo correctamente nuestro archivo schedule con el siguiente comando:

```
cd ~/decidim-app
bundle exec whenever
```

Debería mostrarnos algo como esto:

```
decidim@decidim:~/decidim-app$ bundle exec whenever
PATH=/home/decidim/.rbenv/versions/3.0.5/lib/ruby/gems/3.0.0/bin:/home/decidim/.rbenv/versions/3.0.5/bin:/home/decidim/.rbenv/libexec:/home/decidim/.rbenv/plugins/ruby-build/bin:/home/decidim/.nvm/versions/node/v16.19.1/bin:/home/decidim/.rbenv/shims:/home/decidim/.rbenv/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

0 0 * * * /bin/bash -l -c 'cd /home/decidim/decidim-app && RAILS_ENV=production bundle exec rake decidim:delete_download_your_data_files --silent'

1 0 * * * /bin/bash -l -c 'cd /home/decidim/decidim-app && RAILS_ENV=production bundle exec rake decidim:metrics:all --silent'

2 0 * * * /bin/bash -l -c 'cd /home/decidim/decidim-app && RAILS_ENV=production bundle exec rake decidim_initiatives:check_validating --silent'

3 0 * * * /bin/bash -l -c 'cd /home/decidim/decidim-app && RAILS_ENV=production bundle exec rake decidim_initiatives:check_published --silent'

4 0 * * * /bin/bash -l -c 'cd /home/decidim/decidim-app && RAILS_ENV=production bundle exec rake decidim_initiatives:notify_progress --silent'

5 0 * * * /bin/bash -l -c 'cd /home/decidim/decidim-app && RAILS_ENV=production bundle exec rake decidim:open_data:export --silent'
```

```
6 0 * * * /bin/bash -l -c 'cd /home/decidim/decidim-app && RAILS_ENV=production bundle exec rake decidim_meetings:clean_registration_forms --silent'
```

```
7 0 * * * /bin/bash -l -c 'cd /home/decidim/decidim-app && RAILS_ENV=production bundle exec rake decidim:reminders:all --silent'
```

```
0 7 * * * /bin/bash -l -c 'cd /home/decidim/decidim-app && RAILS_ENV=production bundle exec rake decidim:mailers:notifications_digest_daily --silent'
```

```
0 7 * * 6 /bin/bash -l -c 'cd /home/decidim/decidim-app && RAILS_ENV=production bundle exec rake decidim:mailers:notifications_digest_weekly --silent'
```

```
## [message] Above is your schedule file converted to cron syntax; your crontab file was not updated.  
## [message] Run `whenever --help` for more options.
```

Ahora sólo es cuestión de añadirlo al crontab del servidor, para lo que hay que ejecutar:

```
bundle exec whenever --update-crontab
```

Podemos comprobar nuestro crontab con ***crontab -l***

Por último reiniciamos passenger

```
sudo passenger-config restart-app ~/decidim-app
```

Log de Decidim

Decidim cuenta con un log ubicado en el directorio donde se encuentra la aplicación. Puedes utilizar el comando tail para verificar si hay algún problema en su funcionamiento.

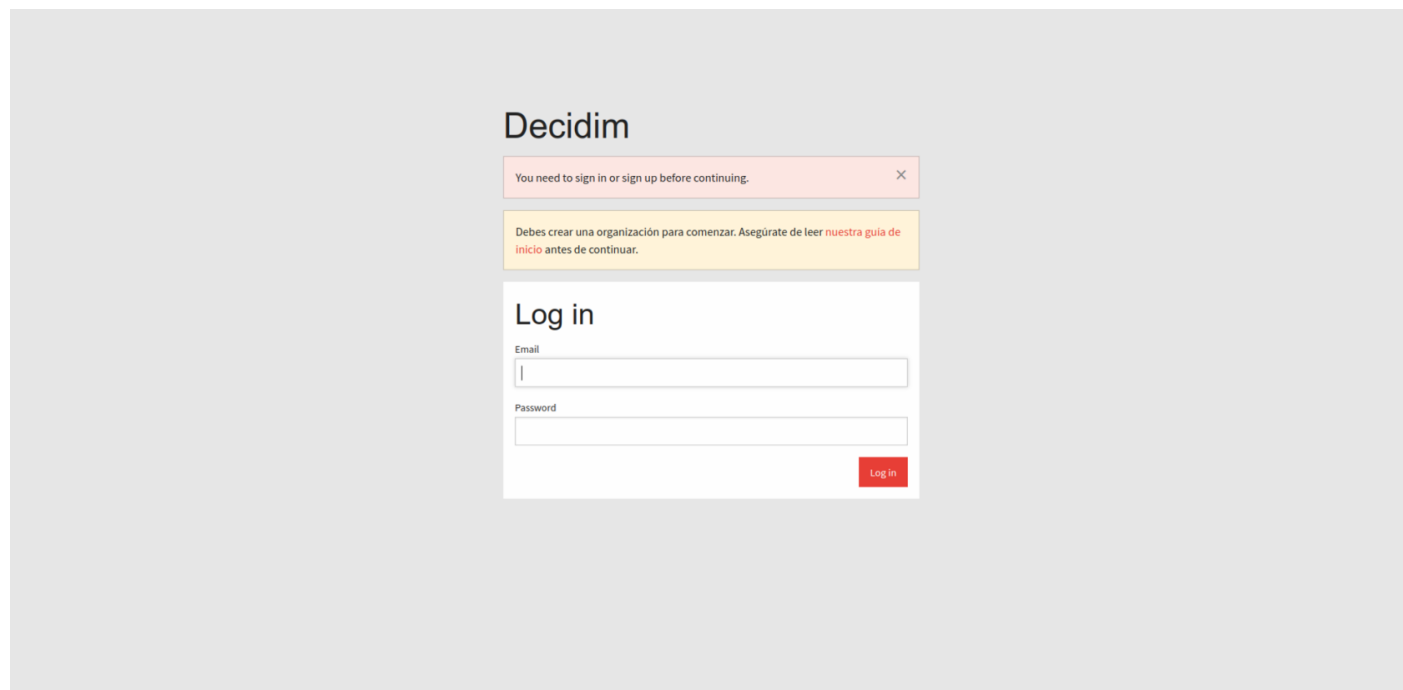
```
tail ~/decidim-app/log/production.log -f
```

Ingresa a Decidim por primera vez

Antes de continuar con el siguiente paso es recomendable que consultes esta sección para prevenir problemas al crear la organización.

Ya con nuestra aplicación funcionando podremos acceder por medio de un navegador web ingresando nuestra dirección de dominio. Por ejemplo, my-decidim.org

Por defecto el sistema nos redireccionará a la pantalla de login del administrador de sistema: `/system/admins/sign_in`



Aquí deberemos ingresar los datos de la credencial (system admin) creada en el proceso de instalación de la plataforma.

Una vez ingresado a la plataforma Decidim, lo primero que veremos es un formulario para dar de alta nuestra organización.

Decidim

Dashboard

Organizations

Admins

OAuth applications

admin@admin.com Logout

New organization

Sesión iniciada con éxito.

You must create an organization to get started. Make sure you read our [getting started guide](#) before proceeding.

Name*

There's an error in this field.

Reference prefix*

The reference prefix is used to uniquely identify resources across all organization

Host*

Secondary hosts

Enter each one of them in a new line

Organization admin name*

Organization admin email*

Organization locales

Locale	Enabled	Default?
Català	<input type="checkbox"/>	<input type="radio"/>

Aquí es importante completar con los datos de su organización y aunque en un futuro se podrán modificar es preferible completar con los datos finales.

Algo importante para tener en cuenta antes de guardar los cambios, es completar la configuración del servicio de envíos de correos SMTP. Para ello deberán hacer clic en “Show advanced settings” y completar con los datos del servicio SMTP que van a utilizar.

Una vez que completamos todos los campos, estaremos en condiciones de guardar presionando “Create organization & invite admin”. Aquí lo que hará Decidim será crear una nueva organización con los datos ingresados y enviará un correo a la dirección indicada en el campo “Organization admin email” con una invitación para administrar dicha organización, es por ello que resulta importante configurar el servicio SMTP de lo contrario este correo no será enviado y el administrador no podrá ingresar a la plataforma.

Con esto terminamos con la guía de instalación de Decidim. Esperamos que le haya sido útil y que tenga su instancia de Decidim corriendo correctamente.

¡Felicitaciones! Podemos decir que tenemos nuestra instancia de Decidim funcionando y lista para servir en producción.

Tenga en cuenta que en la presente guía no se han desarrollado temas que son importantes para la puesta a producción de su aplicación, como lo son los esquemas de backups de base de datos y archivos, la configuración de DNS y el servicio SMTP; No siendo estos menos importantes.

