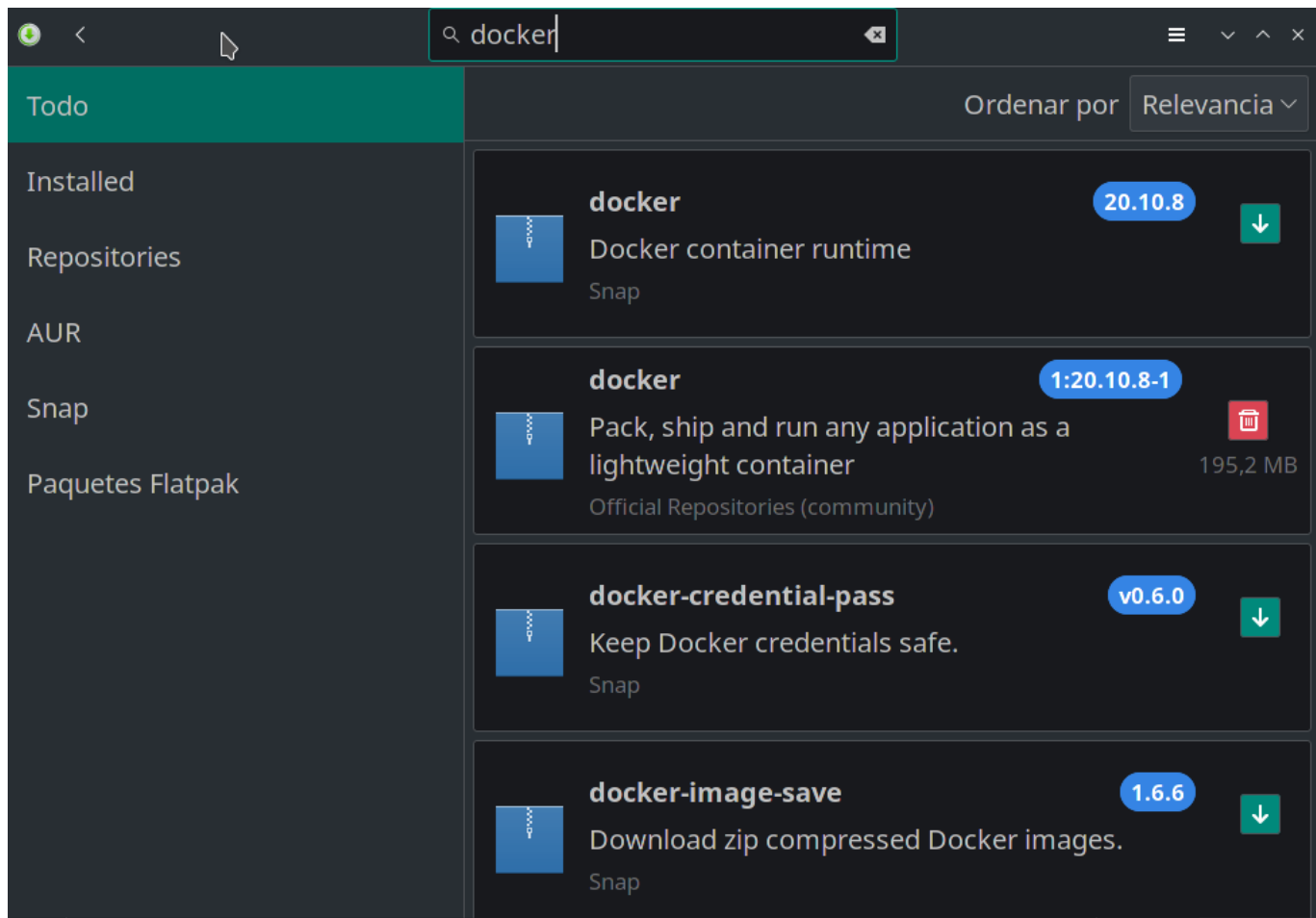


U2. Práctica de Docker

## INSTALACION DE DOCKER (en Manjaro)

Buscamos e instalamos docker desde el repositorio oficial de software de Manjaro:



Habilitamos docker en systemctl para que se haga un enlace simbolico que provoque que docker se inicie con el sistema (y reiniciamos)

```
> sudo systemctl enable docker
[sudo] password for jasnah:
> sudo systemctl status docker

● docker.service – Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; vendor preset: enabled)
   Active: active (running) since Thu 2021-10-07 09:54:11 CEST; 1min 45s ago
   TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
   Main PID: 750 (dockerd)
```

**U2. Practica de Docker****0. Crear, lanzar, parar un nuevo contenedor a partir de una imagen**

Ejemplos de ejecución para comprobar el funcionamiento:

- Ejemplo 1:

```
# docker run --name u1 -it ubuntu:17.10 /bin/bash
```

```
> docker run --name u1 -it ubuntu:17.10 /bin/bash
Unable to find image 'ubuntu:17.10' locally
17.10: Pulling from library/ubuntu
4ccdce43d1e0: Pull complete
c95f13c88d92: Pull complete
82656eee95ad: Pull complete
78ff727be57a: Pull complete
448bb314afa5: Pull complete
Digest: sha256:3b811ac794645dfaa47408f4333ac6e433858
Status: Downloaded newer image for ubuntu:17.10
root@444124b10dd0:/#
```

Hemos arrancado un contenedor usando la imagen de Ubuntu 17.10, en modo interactivo, con nombre "u1", y se nos abre una sesión de bash para interactuar.

Ejecutamos exit para salir del contenedor.

- Ejemplo 2:

```
# docker run -it -p 22 -p 8069:8069 -v/opt/Odoo:/home/Odoo --name odoo_11_dev ubuntu:16.04
```

```
> docker run -it -p 22 -p 8069:8069 -v/opt/Odoo:/home/Odoo --name odoo_11_dev ubuntu:16.04
Unable to find image 'ubuntu:16.04' locally
16.04: Pulling from library/ubuntu
58690f9b18fc: Pull complete
b51569e7c507: Pull complete
da8ef40b9eca: Pull complete
fb15d46c38dc: Pull complete
Digest: sha256:454054f5bbd571b088db25b662099c6c7b3f0cb78536a2077d54adc48f00cd68
Status: Downloaded newer image for ubuntu:16.04
root@0910ae84ef3e:/#
```

Hemos creado un contenedor con las siguientes características:

- El puerto 8069 abierto para acceder desde el anfitrión
- El puerto 22 abierto
- Una terminal interactiva
- Un directorio compartido entre el host (Docker) y el anfitrión (SO)

Es importante indicar la versión de la imagen descargada, por eso al final hemos indicado `ubuntu:16.04`

El resultado es una consola interactiva del SO, con las características especificadas, en el cual podremos ejecutar órdenes.

**U2. Practica de Docker**

- Ejemplo 3:

```
# docker run -d -e POSTGRES_USER=odoo -e POSTGRES_PASSWORD=odoo -e POSTGRES_DB=postgres --name db postgres:10
```

```
> docker run -d -e POSTGRES_USER=odoo -e POSTGRES_PASSWORD=odoo -e POSTGRES_DB=postgres --name db postgres:10
Unable to find image 'postgres:10' locally
10: Pulling from library/postgres
36d925ed8e30: Pull complete
c8b0099cd1a0: Pull complete
0bf2a548fff2: Pull complete
36e6fa05828b: Pull complete
000ab621c968: Pull complete
6855b4b63497: Pull complete
9201c39bd813: Pull complete
de19e180543b: Pull complete
b25ce2d23845: Pull complete
9a1621c579ee: Pull complete
ef3f0fd3ca11: Pull complete
f5442c57fe0c: Pull complete
9c8174e61b6c: Pull complete
efcc6054b54e: Pull complete
Digest: sha256:f99462f9cd5dfd4e0a6812799f00b362ecfa9ff579a5629ea59110473db25184
Status: Downloaded newer image for postgres:10
42535af5ab3681bedab4fd75d95cb76305286914d2cf239653bcd04cb7920e25
```

```
# docker run -p 8069:8069 --name odoo --link db:db -t odoo
```

```
> docker run -p 8069:8069 --name odoo --link db:db -t odoo
Unable to find image 'odoo:latest' locally
latest: Pulling from library/odoo
07aded7c29c6: Pull complete
4202cc480152: Pull complete
6f5f9940244e: Pull complete
0e07d3bb02ae: Pull complete
0ecf74ff7616: Pull complete
d12ad090856f: Pull complete
0cf68f36b396: Pull complete
36578e51156a: Pull complete
06b5b4a2b197: Pull complete
Digest: sha256:155d4ac2e63711ec81bdec2da293dada04473a3daa04419a19b5d54d33fc0fa8
Status: Downloaded newer image for odoo:latest
2021-10-07 11:27:39,489 1 INFO ? odoo: Odoo version 14.0-20211006
2021-10-07 11:27:39,489 1 INFO ? odoo: Using configuration file at /etc/odoo/odoo.conf
2021-10-07 11:27:39,489 1 INFO ? odoo: addons paths: ['/usr/lib/python3/dist-packages/odoo/addons', '/var/lib/odoo/addons/14.0', '/mnt/extra-addons']
2021-10-07 11:27:39,489 1 INFO ? odoo: database: odoo@172.17.0.2:5432
2021-10-07 11:27:39,620 1 INFO ? odoo.addons.base.models.ir_actions_report: Will use the Wkhtmltopdf binary at /usr/local/bin/wkhtmltopdf
2021-10-07 11:27:39,697 1 INFO ? odoo.service.server: HTTP service (werkzeug) running on 6258e0ff8c5d:8069
```

**U2. Practica de Docker****ODOO SOBRE DOCKER**

Vamos a la pagina oficial de repositorios de Docker y nos registramos [<https://hub.docker.com>], aquí podremos guardar nuestras propias imagenes para poder acceder a ellas desde cualquier sitio.

```
$docker login -u alfiefe10  
(alfiefe10 en este caso es mi nombre de usuario)
```

generamos un archivo docker-compose.yml con la configuracion que queremos:

```
version: '3'  
services:  
  odoo:  
    image: odoo:13  
    restart: always  
    ports:  
      - "8069:8069"  
    links:  
      - db  
    volumes:  
      - ./extra-addons:/mnt/extra-addons  
  db:  
    image: postgres:12  
    restart: always  
    environment:  
      - POSTGRES_USER=odoo  
      - POSTGRES_PASSWORD=odoo  
      - POSTGRES_DB=postgres  
      - PGDATA=/var/lib/postgresql/pgdata
```

hacemos el siguiente comando para que Docker componga un contenedor con las configuraciones guardadas:

```
$ docker-compose up -d
```

**U2. Practica de Docker**

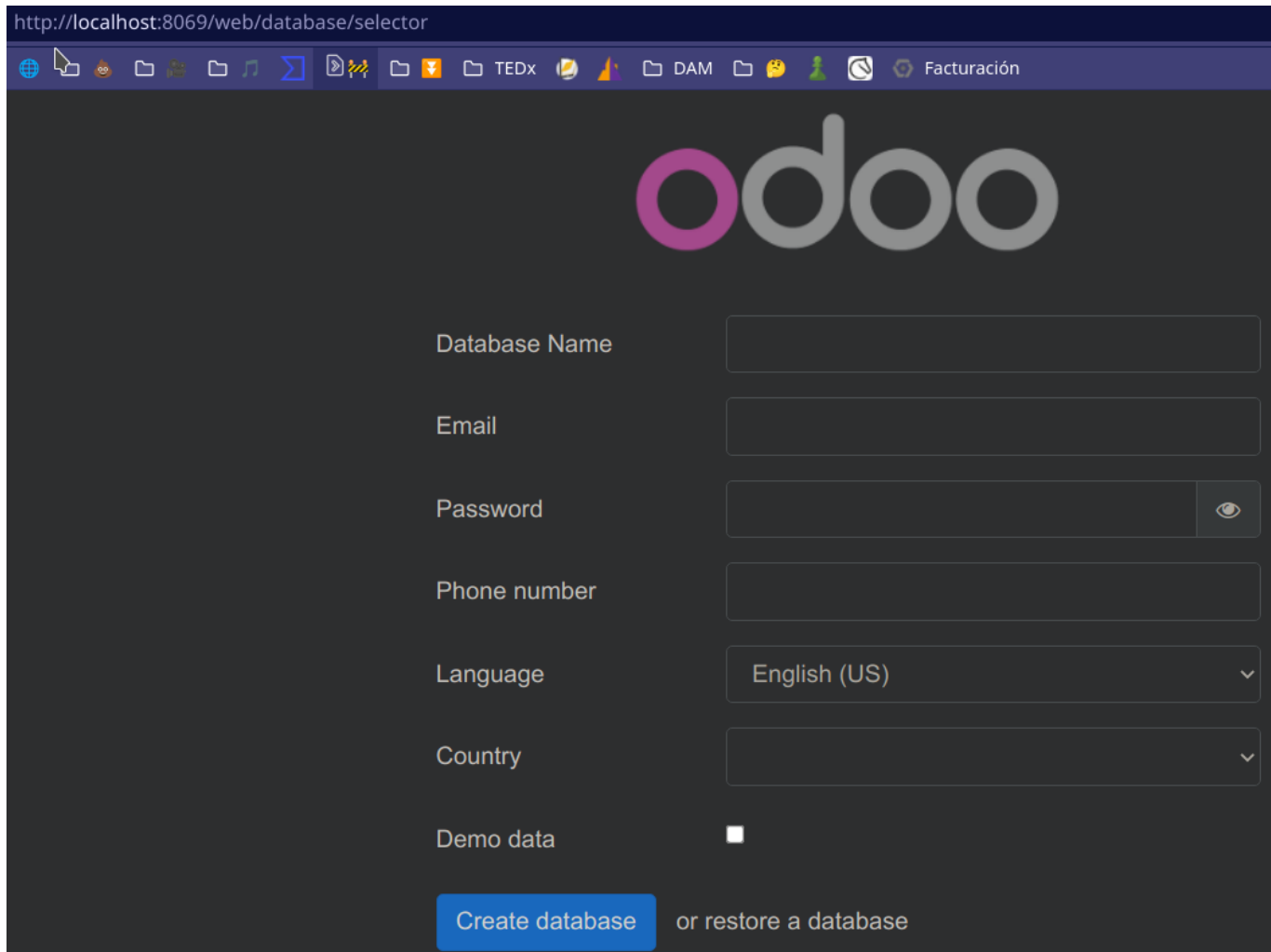
```
> docker-compose up -d

Creating network "docker_default" with the default driver
Pulling db (postgres:12)...
12: Pulling from library/postgres
7d63c13d9b9b: Pull complete
cad0f9d5f5fe: Pull complete
ff74a7a559cb: Pull complete
c43dfd845683: Pull complete
e554331369f5: Pull complete
d25d54a3ac3a: Pull complete
bbc6df00588c: Pull complete
d4deb2e86480: Pull complete
9314f456fc35: Pull complete
1fe77c8f09ee: Pull complete
f07a06e458de: Pull complete
690eab9aace6: Pull complete
f9e2797e33ce: Pull complete
Digest: sha256:5dee94aed72d1549f945e63195ebe720a1472df170f8d2a8807ea02c8ecf6aa2
Status: Downloaded newer image for postgres:12
Pulling odoo (odoo:13)...
13: Pulling from library/odoo
b380bbd43752: Pull complete
45015eb54192: Pull complete
5e8865fe2854: Pull complete
788061f0dbd6: Pull complete
98206d1f603f: Pull complete
e9473fd62cbf: Pull complete
2bb905b54884: Pull complete
bf88a9dcc63d: Pull complete
d6f38b1d63e5: Pull complete
Digest: sha256:b5f2b7391c4d5b6130bef56779aa315b9561017f5e6aa410649b4ae99bd9775c
Status: Downloaded newer image for odoo:13
Creating docker_db_1 ... done
Creating docker_odoo_1 ... done
```

Automaticamente se ha iniciado la descarga de los contenedores docker necesarios, y se iniciado la base de datos, por lo que podremos acceder a nuestra instancia de Odoo desde localhost:8069 o la IP de la maquina y el puerto que hemos indicado.

Una vez en ella tocara crear nuestra base de datos, para la que deberemos elegir email, clave, lenguaje e idioma.

**U2. Práctica de Docker**

A screenshot of a web browser showing the Odoo database selector interface. The browser's address bar displays 'http://localhost:8069/web/database/selector'. The page has a dark theme with the Odoo logo at the top center. Below the logo, there are several input fields: 'Database Name', 'Email', 'Password' (with a toggle icon), 'Phone number', 'Language' (set to 'English (US)'), and 'Country'. At the bottom, there is a checkbox for 'Demo data' and a blue button labeled 'Create database' followed by the text 'or restore a database'. The browser's taskbar at the top shows various icons, including a folder named 'DAM' and a window titled 'Facturación'.

## Gestionar Docker

Existen diferentes herramientas para gestionar docker en un entorno mas amigable, por ejemplo Portainer CE.

“Portainer CE” en lugar de ser una aplicación propiamente dicha, se aloja en un contenedor Docker. En primer lugar crearemos un volumen en el que almacenar la informacion, lo haremos con:

```
$ docker volume create portainer_data
```

Una vez creado el volumen, lanzaremos el contenedor con todo lo necesario para que funcione.

**U2. Práctica de Docker**

“Portainer CE” se gestiona mediante una interfaz web, en este ejemplo mapeamos el servicio web a los puertos del anfitrión 8000 (para TCP si actúa como servidor de otros agentes de Portainer CE) y 9000 (para acceder a la interfaz web).

```
docker run -d -p 8000:8000 -p 9000:9000 --name=portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce
```

```
> docker run -d -p 8000:8000 -p 9000:9000 --name=portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce
Unable to find image 'portainer/portainer-ce:latest' locally
latest: Pulling from portainer/portainer-ce
7721cab3d696: Pull complete
0645e7e2a110: Pull complete
6329543ecfce: Pull complete
Digest: sha256:76ff22486bcd3713631b5f317efcb69e707a122fe96ffcc0589cf2d3e8e6b890
Status: Downloaded newer image for portainer/portainer-ce:latest
5000b884a8974653d7d2b1cac7f7aeddca9f9ab435743cee1eabca03bd810b4
```

Después de esto, podremos acceder a la interfaz de Portainer accediendo a <http://localhost:9000>.

El primer acceso nos solicitará que creamos una clave para el usuario “admin”, con al menos 8 caracteres.

