

Tryton By Example

A tutorial explaining by example
Tryton installation & usage
for first time administrators & users
of the system

-

Administration

Github - [GitHub - mrmarcroottiers/tryton-101: Learn Tryton by example](https://github.com/mrmarcroottiers/tryton-101: Learn Tryton by example)

Version of documentation : 01.01 Dated 26-Apr-21

Version of Tryton : 5.8

Verified for execution on : Windows 10 & Powershell 7

Documentation : GNU GPL

Author : Marc Rottiers

Foreword

This tutorial aims to expedite the process of learning the basics of the TRYTON ERP. System administrators and end-users who want to explore this package should benefit. On the other hand, if you practise the system already, the present documentation will be of little value.

The tutorial rests on a personal initiative. Its content does not contain nor replace official TRYTON documentation in any manner. In fact, this material can be discarded once one feels sufficiently familiar with the system. One should then tailor the usage of the system to the specific administrative workflows in the company.

The material describes the *installation procedure* as well as some *use cases* by example. It leans on database samples and accompanying explicative documents.

The author acknowledges documentation that he had the opportunity to analyse for the purpose of creating the present material.

The tutorial is written based on running TRYTON 5.8 on Windows 10 Home. There is no warranty that the same results will or can be achieved using a different setup. In particular, the author cannot take responsibility for loss or corruption of data that would result from handling processes based on information in the present documentation. He will be grateful to receive any comments in order to improve quality. Post remarks on <https://github.com/mrmarcroottiers/tryton-101/issues>

Known Issues

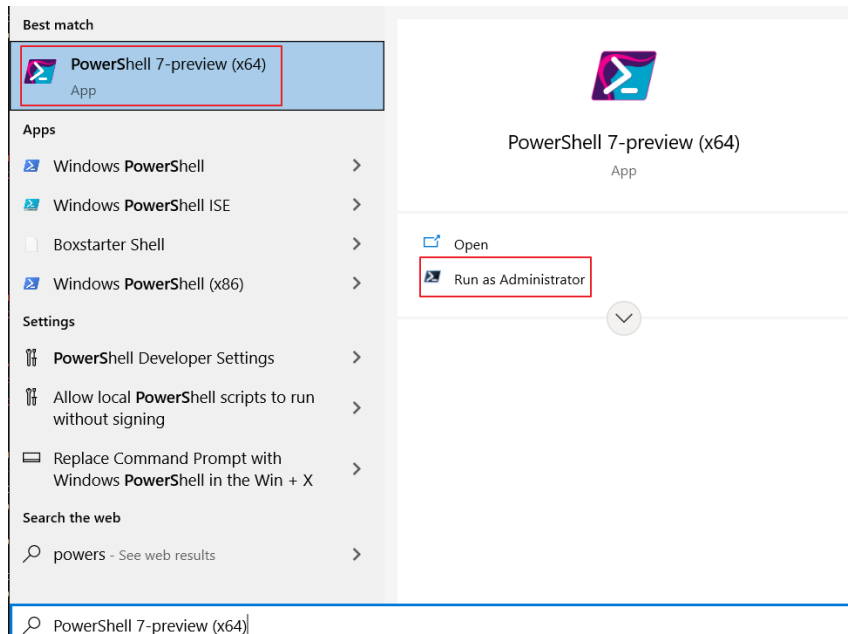
These are unresolved topics that relate to the present documentation, not to the system functioning.

Document	Subject
Tryton 5.8 - Doc 80.01 - Features	Attaching a document to an item causes an exception
Tryton 5.8 - Doc 15.05 - Sales	When a « Sale » evolves into state « Processing », a « Customer Shipment » is produced. Contrarily to a « Customer Invoice », the property « Reference » is not editable. Incidentally, « Reference » in « Customer Invoice » and « Customer Shipment » is not set from « Customer Sale »
Tryton 5.8 - Doc 10.05 - Purchases Tryton 5.8 - Doc 15.05 - Sales	Account Move Lines are generated when « Invoice » into « Validated » state Account Move Lines are not generated when « Invoice » into « Validated » state

Related files & documents

List of files related to some of the topics covered in this document

./"Tryton 5.8 - Doc 01.01 - Installation & administration.database.tryt20.backup"	Backup « tryt20 » database
./"Tryton 5.8 - Doc 01.01 - Installation & administration.database.tryt20.restore"	Restore « tryt20 » database
./"Tryton 5.8 - Doc 01.01 - Installation & administration.docker.tryt20.create"	Create « tryt20 » containers
./"Tryton 5.8 - Doc 01.01 - Installation & administration.docker.tryt20.delete"	Delete « tryt20 » containers
./"Tryton 5.8 - Doc 01.01 - Installation & administration.docker.tryt20.start"	Start « tryt20 » containers
./"Tryton 5.8 - Doc 01.01 - Installation & administration.docker.tryt20.stop"	Stop « tryt20 » containers



- The « *.ps1 » files above are Powershell scripts
- Powershell can be accessed as indicated
- The above scripts need some tuning with respect to the exact operation to be performed

Table of Contents

TOC

- In blue essential information relative to TRYTON
- Consider other sections if unfamiliar with Docker or Postgres

Docker Installation	Installing Docker on Windows
Container Installation	How to install containers
Tryton - « Permanent » Data	Installing Tryton with data residing on volume outside of container
Tryton - « Volatile » Data	Installing Tryton with data inside of container
Postgres - « Permanent » Data	Installing Postgres with data on volume
Docker Administration	How to administer containers
Commands	Commands for managing containers
Reboot	What when rebooting PC
Shell Execution	Executing commands inside a container
Container Uninstallation	How to uninstall containers
Database Backup	Database Backup
Tryton	Backing up a Tryton database (UTF-8)
Postgres	Backing up a Postgres database (UTF-8)
Database Restore	Database Restore
Tryton	Restoring a Tryton database
Postgres	Restoring a Postgres database
Database Operations	Operations about the respective databases
Tryton	Tryton
Postgres	Postgres
User Interface	User Interface
PgAdmin4	Setting up & Exploring the pgadmin4 interface
Tryton	Logging / Logout - Usage
References	Some interesting documentation



Docker Installation

Installation

Context

Windows 10 & Powershell 7

Remark

We do not use WSL2

Download

See <https://docs.docker.com/get-docker/>

Control

Run « docker run hello-world »

```
Windows PowerShell
PS C:\tryt.01\tuto.01> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
b8dfde127a29: Pull complete
Digest: sha256:89b647c604b2a436fc3aa56ab1ec515c26b085ac0c15b0d105bc475be15738fb
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

PS C:\tryt.01\tuto.01> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS              NAMES
9a8c09f14dec   hello-world    "/hello"                 About a minute ago   Exited (0) About a minute ago              festive_villani
PS C:\tryt.01\tuto.01> docker stop festive_villani
festive_villani
PS C:\tryt.01\tuto.01> docker rm festive_villani
festive_villani
PS C:\tryt.01\tuto.01> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS              NAMES
PS C:\tryt.01\tuto.01>
```

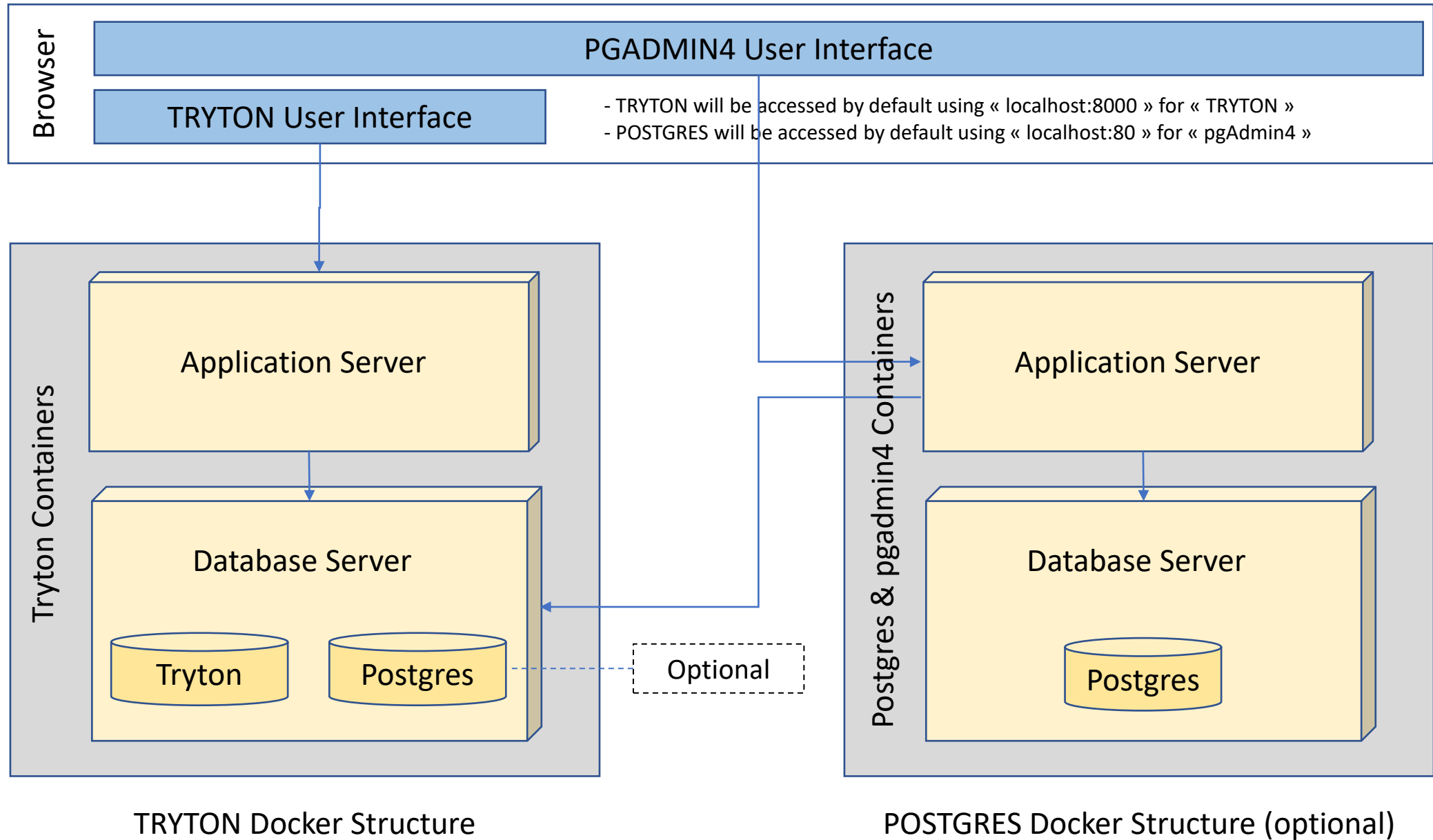
Container Installation

Motivation

We install two series of Docker containers :

- Series 1 is about the TRYTON ERP system : its server application, its database and file systems
- Series 2 is about managing POSTGRES databases and interfacing them using « pgAdmin4 »
- Series 1 is compulsory since it is about the TYTON ERP system : server & database
- Series 2 is optional as its purpose is to use « pgAdmin4 » for direct access to Postgres databases

Motivation



Docker and Databases

Installing TRYTON ERP Containers

TRYTON Docker can be installed in one of two ways with respect to its database and files container :

- In a permanent fashion. It means that the database and any « attachment » files will remain available when a Docker container is removed, accidentally or not.
- In a semi-permanent fashion. It means that if we delete the Docker container the TRYTON database will disappear together with any « attachment » files storing information alongside the database.

Installing POSTGRES Containers (Optional)

The same remark applies.

Convention about password names

Everywhere a password is needed we give it the value « Password »

Commands

Most commands have been explored during the installation process.
We repeat them here for convenience

Container commands	
<code>docker run hello-world</code>	Check installation still operational
<code>docker ps -a</code>	List containers
<code>docker stop a_container_name</code>	Stop a container
<code>docker start a_container_name</code>	Start a container
<code>docker rm a_container_name</code>	Remove a stopped container
<code>docker logs a_container_name</code>	Logs
<code>docker system prune</code>	Remove stopped containers, unused networks and volumes

```
PS C:\Users\mrmar> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
8fa35e9a01ee   dpage/pgadmin4 "/entrypoint.sh"        9 days ago    Up 3 days    0.0.0.0:80->80/tcp, 443/tcp        dev-pgadmin
802400de87ff   postgres      "docker-entrypoint.s..." 9 days ago    Up 3 days    0.0.0.0:5432->5432/tcp            dev-postgres
09de820c6944   tryton/tryton  "/entrypoint.sh uwsg..." 12 days ago    Up 3 days    127.0.0.1:8000->8000/tcp          tryton
d0b1b1578223   postgres      "docker-entrypoint.s..." 12 days ago    Up 3 days    0.0.0.0:5433->5432/tcp            tryton-postgres
```

Commands

Volume commands	
<code>docker volume create a_volume_name</code>	Create a volume
<code>docker volume ls</code>	List the volumes
<code>docker volume rm a_volume_name</code>	Remove a volume
<code>docker volume prune</code>	Remove all unused volumes. « Unused » = not container referenced
<code>docker volume inspect a_volume_name</code>	Inspect a volume
Volume commands	
<code>docker network create a_network_name</code>	Create a network
<code>docker network ls</code>	List the networks
<code>docker network rm a_network_name</code>	Remove a network
<code>docker network prune</code>	Remove all unused networks. « Unused » = not container referenced
<code>docker network inspect a_network_name</code>	Inspect a network

```
PS C:\Users\mrmar> docker volume ls
DRIVER      VOLUME NAME
local       9d31bdf883f8072214686e8fdb261a
local       893c5a68307141a60653bbacf5ede7
local       9518b6e333b5dbbbb4321327b70047
local       tryton-data
local       tryton-database
```

```
PS C:\Users\mrmar> docker network ls
NETWORK ID    NAME        DRIVER    SCOPE
33778f0cf5fa  bridge     bridge    local
ceeb96fab0b4  host       host      local
b52953b3bf65  none       null      local
af20b36cba67  tryton     bridge    local
```

Commands

Bash Shell commands	
<code>docker container exec -it a_container_name /bin/bash</code>	Attach container Bash shell to the terminal
<code>docker container exec -it tryt20-postgres /bin/bash</code>	Example for above
<code>/usr/bin/pg_isready --dbname=tryt20</code>	<code>/var/run/postgresql:5432 - accepting connections</code>

Ancillary commands	
<code>docker exec tryt20-postgres sed -n 4p /var/lib/postgresql/data/pgdata/postmaster.pid</code>	Display postgres port

Tryton - « Permanent » Data

Motivation

The next slides demonstrate a way of setting up the Tryton environment whereby the database and the files are preserved when the corresponding TRYTON container is « removed ».

This solution is inspired from [<https://discuss.tryton.org/t/how-to-run-tryton-using-docker/3200>].

This method performs an additional « volume creation » to store (binary) data outside the database.

Also refer to :

<https://stackoverflow.com/questions/18496940/how-to-deal-with-persistent-storage-e-g-databases-in-docker>

<https://docs.docker.com/storage/volumes/>

Execute

Install :

1. A container « **tryt01-postgres** » from docker image « postgres » ;
« **-p 5434:** » can be changed (default : 5432)
2. A container « **tryt01** » from docker image « tryton/tryton » ;
« **-p 8001:** » can be changed (default : 8000)
3. Two volumes : « **tryt01-database** » & « **tryt01-datafile** »
4. One network : « **tryt01-network** »

A nameless container is used to initialize the TRYTON database.

The location where the volumes for the TRYTON database and the TRYTON files (binary attachments) are stored :

- Subfolder « **tryt01-database** » with respect to the directory (Get-Location) where the Powershell script executes
- Subfolder « **tryt01-datafile** » with respect to the directory (Get-Location) where the Powershell script executes

Some useful commands in case of retry

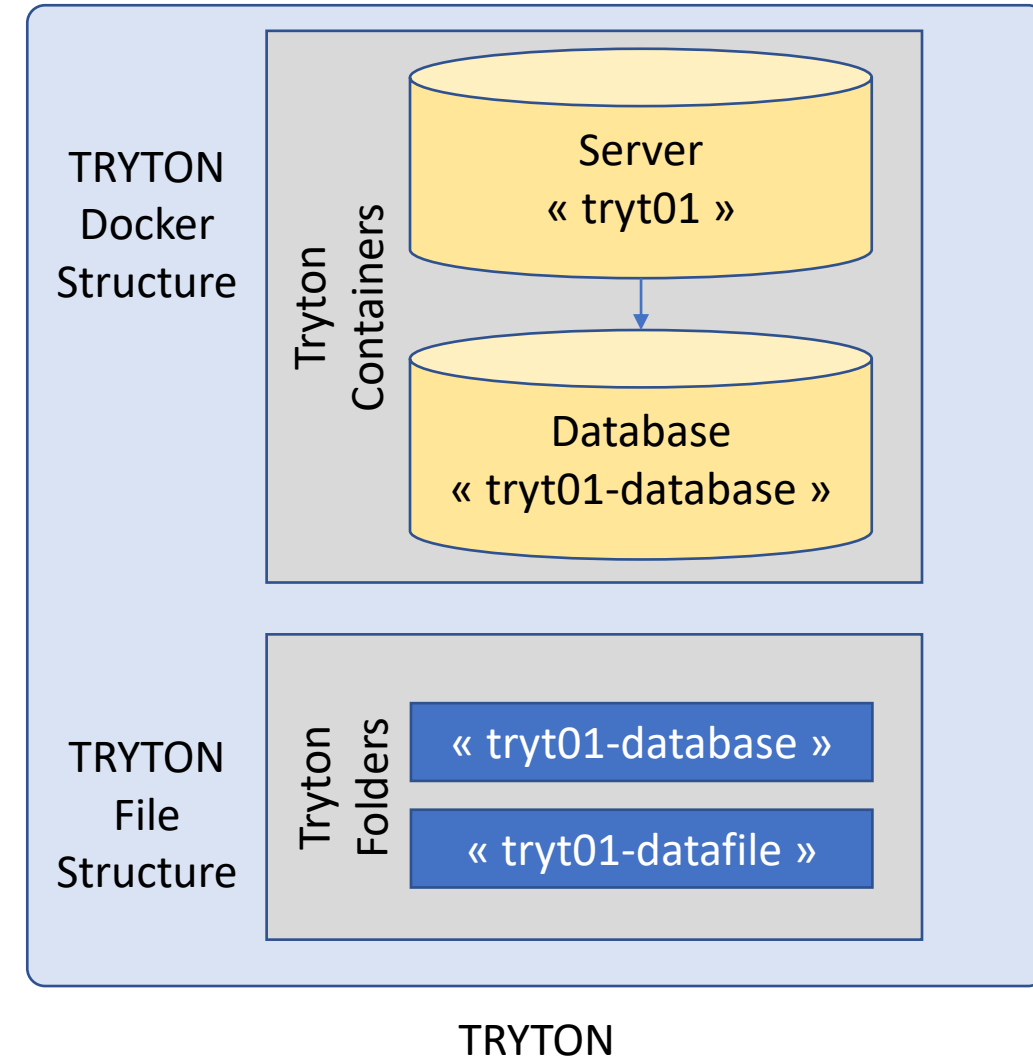
docker stop a_container_name

docker rm a_container_name

docker volume prune

docker network prune

dir env:



Execute - « tryton »

In this script, we use « tryton », the standard name.

```
docker pull tryton/tryton
docker network create tryton-network
$POSTGRES_PASSWORD="Password"
# Tryton database container – Create
$TRYTON_VOL_DB = (Get-Location).toString().replace("\","/").replace("C:/","c//") + "/"+"tryton-database"
docker volume create tryton-database
docker run --name tryton-postgres --env PGDATA=/var/lib/postgresql/data/pgdata --env POSTGRES_DB=tryton --env
POSTGRES_PASSWORD=${POSTGRES_PASSWORD} --volume ${TRYTON_VOL_DB}:/var/lib/postgresql/data --network tryton-network -p
5432:5432 --detach postgres
Start-Sleep -Seconds 20 # required to wait for postgres to properly connect
docker exec -tiu postgres tryton-postgres psql -c '\l+'
dir
# Tryton transient container to initialize the tryton database in its container
docker run --env DB_HOSTNAME=tryton-postgres --env DB_PASSWORD=${POSTGRES_PASSWORD} --network tryton-network --interactive --
tty --rm tryton/tryton trytond-admin -d tryton --all
docker exec -tiu postgres tryton-postgres psql -c '\l+'
# Tryton server container
$TRYTON_VOL_FI = (Get-Location).toString().replace("\","/").replace("C:/","c//") + "/"+"tryton-datafile"
docker volume create tryton-datafile
docker run --name tryton --env DB_HOSTNAME=tryton-postgres --env DB_PASSWORD=${POSTGRES_PASSWORD} --volume
${TRYTON_VOL_FI}:/var/lib/trytond/db --network tryton-network --publish 127.0.0.1:8000:8000 --detach tryton/tryton
dir
# Obtain Gateway address for usage in pgadmin4 - creating server
docker inspect tryton-postgres -f '{{json .NetworkSettings.Networks }}' # "Gateway":"172.18.0.1","IPAddress":"172.18.0.2"
```

Execute - « tryt01 »

In this script, we use « tryt01 » so that it is easy to create a separate « tryt02 » ensemble.

```
docker pull tryton/tryton
docker network create tryt01-network
$POSTGRES_PASSWORD="Password"
# Tryton database container – Create
$TRYTON_VOL_DB = (Get-Location).toString().replace("\","/").replace("C:/","c//") + "/"+"tryt01-database"
docker volume create tryt01-database
docker run --name tryt01-postgres --env PGDATA=/var/lib/postgresql/data/pgdata --env POSTGRES_DB=tryt01 --env
POSTGRES_PASSWORD=${POSTGRES_PASSWORD} --volume ${TRYTON_VOL_DB}:/var/lib/postgresql/data --network tryt01-network -p
5433:5432 --detach postgres
Start-Sleep -Seconds 20 # required to wait for postgres to properly connect
docker exec -tiu postgres tryt01-postgres psql -c '\|+'
dir
# Tryton transient container to initialize the tryton database in its container
docker run --env DB_HOSTNAME=tryt01-postgres --env DB_PASSWORD=${POSTGRES_PASSWORD} --network tryt01-network --interactive --
tty --rm tryton/tryton trytond-admin -d tryt01 --all
docker exec -tiu postgres tryt01-postgres psql -c '\|+'
# Tryton server container
$TRYTON_VOL_FI = (Get-Location).toString().replace("\","/").replace("C:/","c//") + "/"+"tryt01-datafile"
docker volume create tryt01-datafile
docker run --name tryt01 --env DB_HOSTNAME=tryt01-postgres --env DB_PASSWORD=${POSTGRES_PASSWORD} --volume
${TRYTON_VOL_FI}:/var/lib/trytond/db --network tryt01-network --publish 127.0.0.1:8001:8000 --detach tryton/tryton
dir
# Obtain Gateway address for usage in pgadmin4 - creating server
docker inspect tryt01-postgres -f '{{json .NetworkSettings.Networks }}' # "Gateway":"172.18.0.1","IPAddress":"172.18.0.2"
```

Tryton - « Volatile » Data

Motivation

The next slides demonstrate a way of setting up the Tryton environment whereby the database and the files are not preserved when the corresponding TRYTON container is « removed ».

An alternative is documented later. It performs an additional « volume creation » to alleviate the problem.

Execute (1/2)

Pull latest Tryton Docker image (note : occurs automatically when not available locally) :

```
docker pull tryton/tryton
```

```
PS C:\tryt.01\tuto.01> docker pull tryton/tryton
Using default tag: latest
latest: Pulling from tryton/tryton
45b42c59be33: Already exists
25e1e74e2827: Pull complete
ecd401ec74b3: Pull complete
7df2dc86d106: Pull complete
ed33375f55fe: Pull complete
73cb71ba644d: Pull complete
7933c55f7184: Pull complete
e3649bc3d410: Pull complete
Digest: sha256:e66cc2434ceff72b857923d81b366527e87430a3bd9a3869a33012d4c22a5eb6
Status: Downloaded newer image for tryton/tryton:latest
docker.io/tryton/tryton:latest
PS C:\tryt.01\tuto.01> █
```


Execute (2/2)

Obtain latest version of Tryton

```
docker pull tryton/tryton
```

Tryton database container + database initialization volatile container

```
docker run --name tryton-postgres -e POSTGRES_PASSWORD=Password -e POSTGRES_DB=tryton -d postgres # Start a PostgreSQL instance
```

```
docker run --link tryton-postgres:postgres -e DB_PASSWORD=Password -it tryton/tryton trytond-admin -d tryton --all # Define database tables
```

Tryton server containers : tryton & optionally tryton-cron for scheduled actions

```
docker run --name tryton -p 8000:8000 --link tryton-postgres:postgres -e DB_PASSWORD=Password -d tryton/tryton # Start a Tryton instance
```

```
docker run --name tryton-cron --link tryton-postgres:postgres -e DB_PASSWORD=Password -d tryton/tryton trytond-cron -d tryton # Start a cron instance
```

Obtain Gateway address for usage in pgadmin4 - creating server

```
docker inspect tryton-postgres -f "{{json .NetworkSettings.Networks }}" # "Gateway":"172.18.0.1","IPAddress":"172.18.0.2"
```

In blue, container and database variable names that can be chosen

In red, connection points whose external « p:xyz » can be adapted

Execute (3/3)

```
PS C:\tryt.01\tuto.01> docker run --name tryton-postgres -e POSTGRES_PASSWORD=Password -e POSTGRES_DB=tryton -d postgres
29eca77e8ac93d129146a1be0c32ee5c83e6e5ff74c34cfe6aabd21e82bc43cd
PS C:\tryt.01\tuto.01> docker run --link tryton-postgres:postgres -e DB_PASSWORD=Password -it tryton/tryton trytond-admin -d tryton --all
"admin" email for "tryton":
                        @gmail.com
"admin" password for "tryton":
"admin" password confirmation:
PS C:\tryt.01\tuto.01> docker run --name tryton -p 8000:8000 --link tryton-postgres:postgres -e DB_PASSWORD=Password -d tryton/tryton
5661a59a43f04f8d208949c41ea5314e48c5c69a93d8f905b8b948bb4c71b868
PS C:\tryt.01\tuto.01> docker run --name tryton-cron --link tryton-postgres:postgres -e DB_PASSWORD=Password -d tryton/tryton trytond-cron -d tryton
5a143ce0ee8fdc98254c7ac4f14b77ddaec0eaffee3695186ff8f0e21f8f65ac
PS C:\tryt.01\tuto.01> docker inspect tryton-postgres -f "{{json .NetworkSettings.Networks }}"
{"bridge":{"IPAMConfig":null,"Links":null,"Aliases":null,"NetworkID":"42fdbac4cc4198619ba42832b808f76c68790d1d293913f683607a7951a01d89","EndpointID":"95fddfb2a1387d1b8c416f9831312ef3637c4acabb3587970912baac86d74e9","Gateway":"172.17.0.1","IPAddress":"172.17.0.2","IPPrefixLen":16,"IPv6Gateway":"","GlobalIPv6Address":"","GlobalIPv6PrefixLen":0,"MacAddress":"02:42:ac:11:00:02","DriverOpts":null}}
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5a143ce0ee8f	tryton/tryton	"/entrypoint.sh tryt..."	37 seconds ago	Up 36 seconds	8000/tcp	tryton-cron
5661a59a43f0	tryton/tryton	"/entrypoint.sh uwsg..."	54 seconds ago	Up 52 seconds	0.0.0.0:8000->8000/tcp	tryton
235b0c7a014f	tryton/tryton	"/entrypoint.sh tryt..."	2 minutes ago	Exited (0) About a minute ago		zealous_noether
29eca77e8ac9	postgres	"docker-entrypoint.s..."	2 minutes ago	Up 2 minutes	5432/tcp	tryton-postgres

```
PS C:\tryt.01\tuto.01> docker rm zealous_noether
zealous_noether
PS C:\tryt.01\tuto.01> docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5a143ce0ee8f	tryton/tryton	"/entrypoint.sh tryt..."	About a minute ago	Up About a minute	8000/tcp	tryton-cron
5661a59a43f0	tryton/tryton	"/entrypoint.sh uwsg..."	About a minute ago	Up About a minute	0.0.0.0:8000->8000/tcp	tryton
29eca77e8ac9	postgres	"docker-entrypoint.s..."	3 minutes ago	Up 3 minutes	5432/tcp	tryton-postgres

```
PS C:\tryt.01\tuto.01>
```

Outcome : the « Gateway":"172.17.0.1" or the "IPAddress":"172.17.0.2" will be used later

Postgres - « Permanent » Data

Motivation

The next slides demonstrate how to set up an empty Postgres database. This allows to experiment with using PSQL commands upon a database inside a Docker container.

The script creates a POSTGRES database whose data rests « outside » the container. Even when the container is (accidentally) removed using « `docker rm my_postgres_container_name` », the database will persist.

See : [https://dev.to/shree_j/how-to-install-and-run-psql-using-docker-41j2]

It might also be helpful to show folks how to save their data even after running `docker rm <container_id>`. This is how I normally accomplish this:

First, create a local directory to hold the data:

```
mkdir -p /home/<your_user_id_here>/pgdata
```

Then start PostgreSQL using a volume mount so the container will store the data in this newly created local directory:

```
docker run \  
  -d \  
  --name postgresql-container \  
  -p 5432:5432 \  
  -e POSTGRES_PASSWORD=somePassword \  
  -v /home/<your_user_id_here>/pgdata:/var/lib/postgresql/data \  
  postgres
```

Using this method, you can be safe in knowing that even if you accidentally run `docker rm <container_id>` that you can restart PostgreSQL again and have all of your data just as you left it previously.

Execute (1/2)

Install :

- A container « **dev-postgres** » from image « postgres » ; « **-p 5432:** » can be changed
- A container « **dev-pgadmin** » from image « dpage/pgadmin4 » ; « **-p 80:** » can be changed

postgres

docker pull postgres

docker run -d --name **dev-postgres** -e POSTGRES_PASSWORD=**Password** -v \${HOME}/postgres-data:/var/lib/postgresql/data **-p 5432:5432** postgres

pgdamin4

docker pull dpage/pgadmin4

docker run **-p 80:80** -e 'PGADMIN_DEFAULT_EMAIL=x@gmail.com' -e 'PGADMIN_DEFAULT_PASSWORD=**Password**' --name **dev-pgadmin** -d dpage/pgadmin4

inspection

docker exec **dev-postgres** ls /var/lib/postgresql/data

docker exec -tiu postgres **dev-postgres** psql -c '\|+'

docker inspect **dev-postgres** -f "{{json .NetworkSettings.Networks }}"

docker inspect -f "{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}" **dev-postgres**

docker inspect -f '{{.Name}} - {{.NetworkSettings.IPAddress }}' \$(docker ps -aq)

```
PS C:\tryt.01\tuto.01> docker run -d --name dev-postgres -e POSTGRES_PASSWORD=Password -v ${HOME}/postgres-data:/var/lib/postgresql/data -p 5432:5432 postgres
9c43c4a91932619dea73bab39983556765a62dc130ce161ee9be2239f7c6d7
PS C:\tryt.01\tuto.01> docker run -p 80:80 -e 'PGADMIN_DEFAULT_EMAIL=mr.marc.rottiers@gmail.com' -e 'PGADMIN_DEFAULT_PASSWORD=Password' --name dev-pgadmin -d dpage/pgadmin4
5327218f27dce686534a5d794cba4620177a4bd433cc5f9ef344325a61fa3caa
PS C:\tryt.01\tuto.01> docker inspect dev-postgres -f "{{json .NetworkSettings.Networks }}"
{"bridge":{"IPAMConfig":null,"Links":null,"Aliases":null,"NetworkID":"42fdbac4cc4198619ba42832b808f76c68790d1d293913f683607a7951a01d89","EndpointID":"b358d4cd36c77af49b9d5001091381fa62b7a88830a056a5539160a615d17635","Gateway":"172.17.0.1","IPAddress":"172.17.0.5","IPPrefixLen":16,"IPv6Gateway":"","GlobalIPv6Address":"","GlobalIPv6PrefixLen":0,"MacAddress":"02:42:ac:11:00:05","DriverOpts":null}}
PS C:\tryt.01\tuto.01> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
5327218f27dc   dpage/pgadmin4 "/entrypoint.sh"        24 seconds ago Up 23 seconds 0.0.0.0:80->80/tcp, 443/tcp        dev-pgadmin
9c43c4a91932   postgres      "docker-entrypoint.s..." 40 seconds ago Up 39 seconds 0.0.0.0:5432->5432/tcp             dev-postgres
5a143e0ee8f    tryton/tryton  "/entrypoint.sh tryt..." 4 minutes ago  Up 4 minutes  8000/tcp                          tryton-cron
5661a59a43f0   tryton/tryton  "/entrypoint.sh uwsg..." 5 minutes ago  Up 5 minutes  0.0.0.0:8000->8000/tcp             tryton
29eca77e8ac9   postgres      "docker-entrypoint.s..." 6 minutes ago  Up 6 minutes  5432/tcp                          tryton-postgres
PS C:\tryt.01\tuto.01>
```

Execute (2/2)

Outcome :

- The « postgres-data » directory is created under my user name home directory and not inside a docker container.
- Hence the database persists even when the docker container is removed or when the docker application itself is eventually removed.
- The « Gateway » "172.17.0.1" or the « IPAddress » "172.17.0.5" (might vary) will be used later in « pgadmin4 » to define the database as part of a server.

Marc Rottiers > postgres-data >		
Name	Date modified	Type
base	07/03/2021 15:38	File folder
global	07/03/2021 15:38	File folder
pg_commit_ts	07/03/2021 15:38	File folder
pg_dynshmem	07/03/2021 15:38	File folder
pg_logical	07/03/2021 15:38	File folder
pg_multixact	07/03/2021 15:38	File folder
pg_notify	07/03/2021 15:38	File folder
pg_replslot	07/03/2021 15:38	File folder
pg_serial	07/03/2021 15:38	File folder
pg_snapshots	07/03/2021 15:38	File folder
pg_stat	07/03/2021 15:38	File folder
pg_stat_tmp	07/03/2021 15:38	File folder
pg_subtrans	07/03/2021 15:38	File folder
pg_tblspc	07/03/2021 15:38	File folder
pg_twophase	07/03/2021 15:38	File folder
pg_wal	07/03/2021 15:38	File folder
pg_xact	07/03/2021 15:38	File folder
pg_hba.conf	07/03/2021 15:38	CONF File
pg_ident.conf	07/03/2021 15:38	CONF File
PG_VERSION	07/03/2021 15:38	File
postgresql.auto.conf	07/03/2021 15:38	CONF File
postgresql.conf	07/03/2021 15:38	CONF File
postmaster.opts	07/03/2021 15:38	OPTS File
postmaster.pid	07/03/2021 15:38	PID File

Docker Administration

Commands

Commands

Most commands have been explored during the installation process.
We repeat them here for convenience

Container commands	
<code>docker run hello-world</code>	Check installation still operational
<code>docker ps -a</code>	List containers
<code>docker stop a_container_name</code>	Stop a container
<code>docker start a_container_name</code>	Start a container
<code>docker rm a_container_name</code>	Remove a stopped container
<code>docker logs a_container_name</code>	Logs
<code>docker system prune</code>	Remove stopped containers, unused networks and volumes

```
PS C:\Users\mrmar> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
8fa35e9a01ee   dpage/pgadmin4 "/entrypoint.sh"        9 days ago    Up 3 days    0.0.0.0:80->80/tcp, 443/tcp        dev-pgadmin
802400de87ff   postgres      "docker-entrypoint.s..." 9 days ago    Up 3 days    0.0.0.0:5432->5432/tcp            dev-postgres
09de820c6944   tryton/tryton  "/entrypoint.sh uwsg..." 12 days ago    Up 3 days    127.0.0.1:8000->8000/tcp          tryton
d0b1b1578223   postgres      "docker-entrypoint.s..." 12 days ago    Up 3 days    0.0.0.0:5433->5432/tcp            tryton-postgres
```

Commands

Volume commands	
<code>docker volume create a_volume_name</code>	Create a volume
<code>docker volume ls</code>	List the volumes
<code>docker volume rm a_volume_name</code>	Remove a volume
<code>docker volume prune</code>	Remove all unused volumes. « Unused » = container referenced
<code>docker volume inspect a_volume_name</code>	Inspect a volume
Volume commands	
<code>docker network create a_network_name</code>	Create a network
<code>docker network ls</code>	List the networks
<code>docker network rm a_network_name</code>	Remove a network
<code>docker network prune</code>	Remove all unused networks. « Unused » = not container referenced
<code>docker network inspect a_network_name</code>	Inspect a network

```
PS C:\Users\mrmar> docker volume ls
DRIVER      VOLUME NAME
local       9d31bdf883f8072214686e8fdb261a
local       893c5a68307141a60653bbacf5ede7
local       9518b6e333b5dbbbb4321327b70047
local       tryton-data
local       tryton-database
```

```
PS C:\Users\mrmar> docker network ls
NETWORK ID    NAME        DRIVER    SCOPE
33778f0cf5fa  bridge     bridge    local
ceeb96fab0b4  host       host      local
b52953b3bf65  none       null      local
af20b36cba67  tryton     bridge    local
```

Reboot

After Reboot

Each time the PC is rebooted, we need to execute the following commands in Powershell
docker ps -a # Control the status

```
PS C:\Users\mrmar> docker ps -a
```

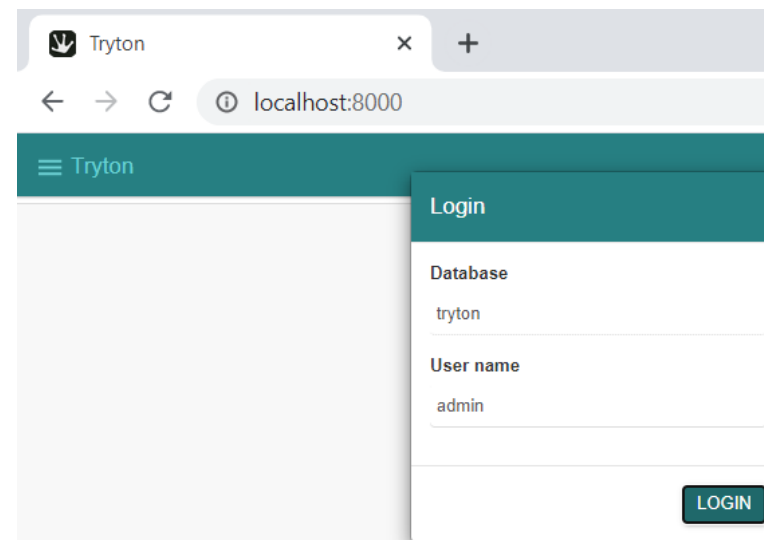
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
8fa35e9a01ee	dpage/pgadmin4	"/entrypoint.sh"	11 days ago	Exited (255) 10 minutes ago
802400de87ff	postgres	"docker-entrypoint.s..."	11 days ago	Exited (255) 10 minutes ago
09de820c6944	tryton/tryton	"/entrypoint.sh uwsg..."	2 weeks ago	Exited (255) 10 minutes ago
d0b1b1578223	postgres	"docker-entrypoint.s..."	2 weeks ago	Exited (255) 10 minutes ago
2657e25ff2af	tryton/tryton	"/entrypoint.sh tryt..."	2 weeks ago	Exited (14) 1 weeks ago

Start the containers

docker start tryton-postgres tryton tryton-cron

docker start dev-postgres dev-pgadmin

Above commands are unnecessary when the
PC is set to "Sleep"



Shell Execution

Execute

Executing commands and shells inside a Docker Container is sometimes necessary. Examples are :

- Executing « psql » inside the « Postgres » Docker Container
- Executing a Windows command like « ls », « mkdir », « cp between container and windows »
- etc.

Commands	
<code>docker exec tryton-postgres ls</code>	List content of directory
<code>docker exec tryton-postgres env</code>	List environment variables
<code>docker cp tryton-postgres:/dump.tar .</code>	Copy tar file from inside of container
<code>docker exec -it tryton-postgres /bin/bash</code>	Executes shell in interactive mode

Usefull documentation :

<https://martinheinz.dev/blog/3>

<https://phoenixnap.com/kb/docker-run-command-with-examples>

Execute

```
postmaster.pid
PS C:\Users\mrmar> docker exec -it dev-postgres echo "I'm inside the container"
I'm inside the container
```

```
see 'docker run --help'.
PS C:\Users\mrmar> docker container exec -it dev-postgres /bin/bash
root@802400de87ff:/# ls
bin boot dev docker-entrypoint-initdb.d docker-entrypoint.sh etc
root@802400de87ff:/#
```

```
Usage: docker exec [OPTIONS] CONTAINER COMMAND [ARG...]
```

Run a command in a running container

```
PS C:\Users\mrmar> docker exec tryton-postgres ls
```

```
bin
boot
dev
directory
docker-entrypoint-initdb.d
docker-entrypoint.sh
etc
home
lib
lib64
media
mnt
opt
postgres
proc
root
run
sbin
srv
sys
tmp
t.tar
usr
var
PS C:\Users\mrmar>
```

```
PS C:\Users\mrmar> docker exec tryton-postgres ls /var/lib/postgresql/data/pgdata
```

```
base
global
pg_commit_ts
pg_dynshmem
pg_hba.conf
pg_ident.conf
pg_logical
pg_multixact
pg_notify
pg_replslot
pg_serial
pg_snapshots
pg_stat
pg_stat_tmp
pg_subtrans
pg_tblspc
pg_twophase
PG_VERSION
pg_wal
pg_xact
postgresql.auto.conf
postgresql.conf
postmaster.opts
postmaster.pid
PS C:\Users\mrmar>
```

Execute

```
PS C:\Users\mrmar> docker exec tryton-postgres ls
```

```
bin
boot
dev
directory
docker-entrypoint-initdb.d
docker-entrypoint.sh
etc
FILE1.tar
home
lib
lib64
media
mnt
opt
postgres
proc
root
run
sbin
srv
sys
tmp
t.tar
usr
var
```

```
PS C:\Users\mrmar> docker cp tryton-postgres:/FILE1.tar ./FILE2.tar
```

```
PS C:\Users\mrmar> dir fi*
```

Directory: C:\Users\mrmar

Mode	LastWriteTime	Length	Name
----	-----	-----	----
d----	05/04/2021 16:16		FILE
-a---	05/04/2021 16:34	8704	FILE2.tar

```
PS C:\Users\mrmar> docker exec dev-postgres env
```

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/lib/postgresql/13/bin
HOSTNAME=802400de87ff
POSTGRES_PASSWORD=Password
GOSU_VERSION=1.12
LANG=en_US.utf8
PG_MAJOR=13
PG_VERSION=13.2-1.pgdg100+1
PGDATA=/var/lib/postgresql/data
HOME=/root
```

```
PS C:\Users\mrmar> docker exec tryton-postgres env
```

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/lib/postgresql/13/bin
HOSTNAME=d0b1b1578223
PGDATA=/var/lib/postgresql/data/pgdata
POSTGRES_DB=tryton
POSTGRES_PASSWORD=Password
GOSU_VERSION=1.12
LANG=en_US.utf8
PG_MAJOR=13
PG_VERSION=13.2-1.pgdg100+1
HOME=/root
```


Container Uninstallation

Deleting everything

docker ps -a

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
132372a60d4b	postgres	"docker-entrypoint.s..."	2 hours ago	Up 2 hours	0.0.0.0:5433->5432/tcp	tryton-postgres
4b6f9e99f6b4	tryton/tryton	"/entrypoint.sh uwsg..."	2 hours ago	Up 2 hours	127.0.0.1:8000->8000/tcp	tryton
5a2d77dfdc68	dpage/pgadmin4	"/entrypoint.sh"	5 hours ago	Up 2 hours	0.0.0.0:80->80/tcp, 443/tcp	dev-pgadmin
f22dca2038de	postgres	"docker-entrypoint.s..."	5 hours ago	Up 2 hours	0.0.0.0:5432->5432/tcp	dev-postgres

docker stop tryton-postgres tryton dev-pgadmin dev-postgres

tryton-postgres

tryton

dev-pgadmin

dev-postgres

docker rm tryton-postgres tryton dev-pgadmin dev-postgres

tryton-postgres

tryton

dev-pgadmin

dev-postgres

docker network prune

WARNING! This will remove all custom networks not used by at least one container.

Are you sure you want to continue? [y/N] y

Deleted Networks:

tryton

docker volume prune

WARNING! This will remove all local volumes not used by at least one container.

Are you sure you want to continue? [y/N] y

Deleted Volumes:

0c512afd78d327b43b99e350972e65c25fd4d3ae9be0cfdc362f08298254505f

tryton-database

tryton-data

Total reclaimed space: 55.68MB

Database Backup

Motivation

- Explaining how to perform database dump and restore for a database inside a container
- Ensuring that these operations preserve UTF8 encoding

Tryton

Tryton - Backup

Note :

- The “tryton-postgres” container contains two databases : “postgres” and “tryton”
- The “dev-postgres” container (if installed) contains one database : “postgres”

Generate backup inside of docker container (choose one of three formats)

```
docker exec tryton-postgres pg_dump -C -c -U postgres -O -f tryton-db-backup.sql tryton
```

```
docker exec tryton-postgres pg_dump -Fc -U postgres -O -f tryton-db-backup.bak tryton
```

```
docker exec tryton-postgres pg_dump -Ft -U postgres -O -f tryton-db-backup.tar tryton
```

Copy backup to outside of docker container

```
docker exec tryton-postgres ls -l
```

```
docker cp tryton-postgres:/tryton-db-backup.sql tryton-db-backup.sql
```

```
docker cp tryton-postgres:/tryton-db-backup.bak tryton-db-backup.bak
```

```
docker cp tryton-postgres:/tryton-db-backup.tar tryton-db-backup.tar
```

```
ls -l
```

In the examples above :

- “tryton-postgres” : name of docker container for tryton database
- “tryton” : name of tryton database

Refer to section “Postgres Backup” hereafter for more details

Postgres

Documentation

List of options	http://manpages.ubuntu.com/manpages/trusty/man1/pg_dump.1.html
	<code>pg_dump --help</code>
How to	http://postgresguide.com/utilities/backup-restore.html
	https://simkimsia.com/how-to-restore-database-dumps-for-postgres-in-docker-container/
	https://stackoverflow.com/questions/24718706/backup-restore-a-dockerized-postgresql-database

« pg_dump » - Redirection incorrectly working

Note :

- The “tryton-postgres” container contains two databases : “postgres” and “tryton”
- The “dev-postgres” container (if installed) contains one database : “postgres”

```
docker exec tryton-postgres pg_dump -C -c -U postgres -O postgres > postgres-db-backup.sql
```

```
docker exec tryton-postgres pg_dump -Fc -U postgres -O postgres > postgres-db-backup.bak
```

```
docker exec tryton-postgres pg_dump -Ft -U postgres -O postgres > postgres-db-backup.tar
```

- Above file content redirections generate incorrect results
- File assignment must be used (see hereafter)

« pg_dump » - Character mode

Generate backup inside of docker container

```
docker exec tryton-postgres pg_dump -C -c -U postgres -O -f postgres-db-backup.sql postgres
```

Copy backup to outside of docker container

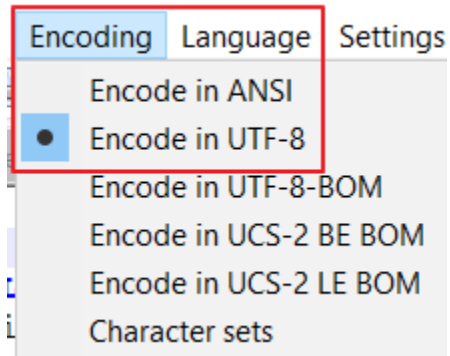
```
docker exec tryton-postgres ls -l
```

```
docker cp tryton-postgres:/postgres-db-backup.sql postgres-db-backup.sql
```

```
ls -l
```

- Generates « drop » and create » « postgres » database commands
- Dumps clear text in « correct » UTF8 (the encoding is taken from the database setting)
- The file is created inside docker container
- If the container is removed, the file is also removed unless it is saved on a persistent volume that is mounted on the container. Alternatively, the file can be copied outside container for preservation.

```
18
19 DROP DATABASE postgres;
20
21 -- Name: postgres; Type: DATABASE; S
22
23
24 CREATE DATABASE postgres WITH TEMPLA
25
26
27 \connect postgres
```



```
COPY public.person (personid, lastname) FROM stdin;
1 Tom B. Eréchsénçan
\.
```

« pg_dump » - Binary Mode

```
docker exec tryton-postgres pg_dump -Fc -U postgres -O -f postgres-db-backup.bak postgres
```

```
docker exec tryton-postgres pg_dump -Ft -U postgres -O -f postgres-db-backup.tar postgres
```

```
PGDMP|SOH|SO|NUL|EOT|BS|SOH|SOH|SOH|NUL|NUL|NUL|NUL|
NUL|RS|NUL|NUL|NUL|13.2 (Debian 13.2-1.pgdg10
NUL|BS|NUL|NUL|NUL|ENCODING|NUL|BS|NUL|NUL|NUL|ENC
SOH|SOH|NUL|NUL|NUL|SOH|SOH|NUL|NUL|NUL|SOH|SOH|NU
VT|NUL|NUL|NUL|NUL|NUL|NUL|NUL|NUL|SOH|NUL|NUL|NUL|
```

- « Fc » saves in a binary file
- « Ft » saves in a « tar » file. It is the most condensed version

```
PS C:\Users\mrmar> docker exec tryton-postgres pg_dump -Ft -U postgres -O -f postgres-db-backup.tar postgres
PS C:\Users\mrmar> docker exec tryton-postgres ls -l
total 96
drwxr-xr-x  2 root root 4096 Mar 11 00:00 bin
drwxr-xr-x  2 root root 4096 Jan 30 17:37 boot
drwxr-xr-x  5 root root  340 Apr  4 10:00 dev
drwx----- 2 root root 4096 Mar 28 18:19 directory
drwxr-xr-x  2 root root 4096 Mar 12 15:25 docker-entrypoint-initdb.d
lrwxrwxrwx  1 root root   34 Mar 12 15:26 docker-entrypoint.sh -> usr/local/bin/docker-entrypoint.sh
drwxr-xr-x  1 root root 4096 Mar 25 14:31 etc
drwxr-xr-x  2 root root 4096 Jan 30 17:37 home
drwxr-xr-x  1 root root 4096 Mar 12 15:25 lib
drwxr-xr-x  2 root root 4096 Mar 11 00:00 lib64
drwxr-xr-x  2 root root 4096 Mar 11 00:00 media
drwxr-xr-x  2 root root 4096 Mar 11 00:00 mnt
drwxr-xr-x  2 root root 4096 Mar 11 00:00 opt
-rw-r--r--  1 root root 1940 Mar 29 06:37 postgres
-rw-r--r--  1 root root 1151 Apr  6 15:27 postgres-db-backup.sql
-rw-r--r--  1 root root 8704 Apr  6 16:48 postgres-db-backup.tar
dr-xr-xr-x 219 root root   0 Apr  4 10:00 proc
```

Database Restore

Tryton

« pg_restore » - Binary Mode - Example in Context

Use « pg_restore » utility, not « psql »

- Before restoring the TRYTON database using the following script, log out the system if you happen to be using it.
- See section hereafter about TRYTON Login / Logout

```
# Step 1.1 : dump tryton
docker exec tryton-postgres pg_dump -Ft -U postgres -O -f tryton-db-backup.tar tryton
# Step 1.2 : export outside container (optional ; specifically use if later import in another container)
docker cp tryton-postgres:/tryton-db-backup.tar tryton-db-backup.tar
# Step 2 : docker stop/start containers
docker stop tryton-postgres tryton
docker start tryton-postgres tryton
docker ps -a
# Step 3 : drop and create tryton-copy
docker exec tryton-postgres dropdb -f -U postgres tryton-copy
docker exec tryton-postgres createdb -U postgres -T template0 tryton-copy
# Step 4.1 : import inside container (optional ; function of step 1.2 above)
docker cp tryton-db-backup.tar tryton-postgres:/tryton-db-backup.tar
# Step 4.2 : restore tryton-copy from tryton
docker exec -i tryton-postgres pg_restore -Ft -U postgres -d tryton-copy -v ./tryton-db-backup.tar
```

Errors

- We have documented hereafter three error situations we have encountered
- We have succeeded in eliminating all of them by tuning the « restore » script
- Please report if any error case reappears

Error #1

- It is possible to restore a backup database to its previous state whilst keeping its name unchanged e.g. « tryton »
- The restore can occur into the same container (hence overwriting the database) or into a different one
- If the error below occurs on login after restore, re-login a second time

Application Error

```
Traceback (most recent call last):
  File "/usr/local/lib/python37/dist-packages/trytond/transactionpy", line 227, in commit
    self.connection.commit()
psycopg2InterfaceError: connection already closed

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python37/dist-packages/trytond/wsgi.py", line 116, in dispatch_request
    return endpoint(request, **requestview_args)
  File "/usr/local/lib/python37/dist-packages/trytond/protocols/dispatcher.py", line 48, in rpc
    request, database_name, *request_rpc_params)
  File "/usr/local/lib/python37/dist-packages/trytond/protocols/dispatcher.py", line 63, in login
    database_name, user, parameters, context=context)
  File "/usr/local/lib/python37/dist-packages/trytond/security.py", line 42, in login
    raise
  File "/usr/local/lib/python37/dist-packages/trytond/transactionpy", line 136, in __exit__
    self.stop(type is None)
  File "/usr/local/lib/python37/dist-packages/trytond/transactionpy", line 147, in stop
    self.commit()
  File "/usr/local/lib/python37/dist-packages/trytond/transactionpy", line 229, in commit
    self.rollback()
  File "/usr/local/lib/python37/dist-packages/trytond/transactionpy", line 247, in rollback
    self.connection.rollback()
psycopg2InterfaceError: connection already closed
```


Error #2

- It is not possible to restore a backup database whilst changing its name e.g. from « tryt20 » to « tryt21 »
- The error occurs whether restoring into the same or a different container as the original one
- After login, the error below persists when clicking « Currencies » (e.g.) in the Menu Tree

Application Error

1

```
Traceback (most recent call last):
  File "/usr/local/lib/python37/dist-packages/trytond/wsgipy", line 116, in dispatch_request
    return endpoint(request, **requestview_args)
  File "/usr/local/lib/python37/dist-packages/trytond/protocols/dispatcherpy", line 48, in rpc
    request, database_name, *requestrpc_params)
  File "/usr/local/lib/python37/dist-packages/trytond/wsgipy", line 83, in auth_required
    return wrapped(*args, **kwargs)
  File "/usr/local/lib/python37/dist-packages/trytond/protocols/wrapperspy", line 131, in wrapper
    return func(request, pool, *args, **kwargs)
  File "/usr/local/lib/python37/dist-packages/trytond/protocols/dispatcherpy", line 181, in _dispatch
    result = rpcresult(meth(*c_args, **c_kwargs))
  File "/usr/local/lib/python37/dist-packages/trytond/ir/actionpy", line 252, in get_keyword
    for value in Actionget_action_values(type_, action_ids):
  File "/usr/local/lib/python37/dist-packages/trytond/ir/actionpy", line 119, in get_action_values
    return Actionread(action_ids, columns)
  File "/usr/local/lib/python37/dist-packages/trytond/model/modelsqlpy", line 859, in read
    getter_results = fieldget(ids, cls, field_list, values=result)
  File "/usr/local/lib/python37/dist-packages/trytond/model/fields/functionpy", line 106, in get
    return dict((name, call(name)) for name in names)
  File "/usr/local/lib/python37/dist-packages/trytond/model/fields/functionpy", line 106, in <genexpr>
    return dict((name, call(name)) for name in names)
  File "/usr/local/lib/python37/dist-packages/trytond/model/fields/functionpy", line 99, in call
    return method(records, name)
  File "/usr/local/lib/python37/dist-packages/trytond/ir/actionpy", line 863, in get_pyson
    getattr(poolget(windowres_model), '_order', 'null'))
  File "/usr/local/lib/python37/dist-packages/trytond/poolpy", line 187, in get
    return self_pool[selfdatabase_name][type][name]
KeyError: 'currencycurrency'
```

Error #3

- We have experienced the following error whilst restoring a database under the same name in its original container
- The error disappeared when :
 - We performed a stop/start operation on the docker TRYTON containers
 - And subsequently restored the database

```
Traceback (most recent call last):
  File "/usr/local/lib/python37/dist-packages/trytond/wsgipy", line 116, in dispatch_request
    return endpoint(request, **requestview_args)
  File "/usr/local/lib/python37/dist-packages/trytond/protocols/dispatcher.py", line 48, in rpc
    request, database_name, *requesttrpc_params)
  File "/usr/local/lib/python37/dist-packages/trytond/protocols/dispatcher.py", line 63, in login
    database_name, user, parameters, context=context)
  File "/usr/local/lib/python37/dist-packages/trytond/security.py", line 50, in login
    session = user_id, Sessionnew()
  File "/usr/local/lib/python37/dist-packages/trytond/ir/session.py", line 47, in new
    session, = clscreate([values])
  File "/usr/local/lib/python37/dist-packages/trytond/ir/session.py", line 145, in create
    return super(Session, cls).create(vlist)
  File "/usr/local/lib/python37/dist-packages/trytond/model/modelsqlpy", line 159, in wrapper
    return func(cls, *args, **kwargs)
  File "/usr/local/lib/python37/dist-packages/trytond/model/modelsqlpy", line 671, in create
    cls__check_domain_rule(new_ids, 'create')
  File "/usr/local/lib/python37/dist-packages/trytond/model/modelsqlpy", line 1220, in __check_domain_rule
    domain = Ruledomain_get(cls__name__, mode=mode)
  File "/usr/local/lib/python37/dist-packages/trytond/ir/rulepy", line 246, in domain_get
    clause, clause_global = clsget(model_name, mode=mode)
  File "/usr/local/lib/python37/dist-packages/trytond/ir/rulepy", line 211, in get
    decoder = PYSONDecoder(cls_get_context())
  File "/usr/local/lib/python37/dist-packages/trytond/modules/company/irpy", line 77, in __get_context
    if useremployee:
  File "/usr/local/lib/python37/dist-packages/trytond/model/fields/fieldpy", line 344, in __get__
    return inst__getattn__(selfname)
  File "/usr/local/lib/python37/dist-packages/trytond/model/modelstorage.py", line 1586, in __getattn__
    read_data = selfread(list(indexkeys()), list(fieldskeys()))
  File "/usr/local/lib/python37/dist-packages/trytond/modules/stock/respy", line 44, in read
    result = super().read(ids, fields_names)
  File "/usr/local/lib/python37/dist-packages/trytond/modules/company/respy", line 125, in read
    result = super(User, cls).read(ids, fields_names)
  File "/usr/local/lib/python37/dist-packages/trytond/res/userpy", line 354, in read
    result = super(User, cls).read(ids, fields_names)
  File "/usr/local/lib/python37/dist-packages/trytond/model/modelsqlpy", line 781, in read
    order_by=history_order, limit=history_limit))
  File "/usr/local/lib/python37/dist-packages/trytond/backend/postgresql/database.py", line 69, in execute
    cursor.execute(self, sql, args)
psycopg2ProgrammingError: column awarehouse does not exist
LINE 1: SELECT "a"."name" AS "name", "a"."warehous" AS "warehouse",
          ^
```

Postgres

« psql » - Character Mode - Example in Context

- Use « psql » utility, not « pg_restore »
- Use the file variant « *.sql » when restoring to a different data base name
- It does not contain database « drop & create » commands

Step 1.1 : dump **tryton**

```
docker exec tryton-postgres pg_dump -C -c -U postgres -O -f postgres-db-backup.sql postgres
```

Step 1.2 : export outside container (optional ; specifically use if later import in another container)

```
docker cp tryton-postgres:/postgres-db-backup.sql postgres-db-backup.sql
```

Step 2 : drop and create **tryton-copy**

```
docker exec tryton-postgres dropdb -f -U postgres postgres-copy
```

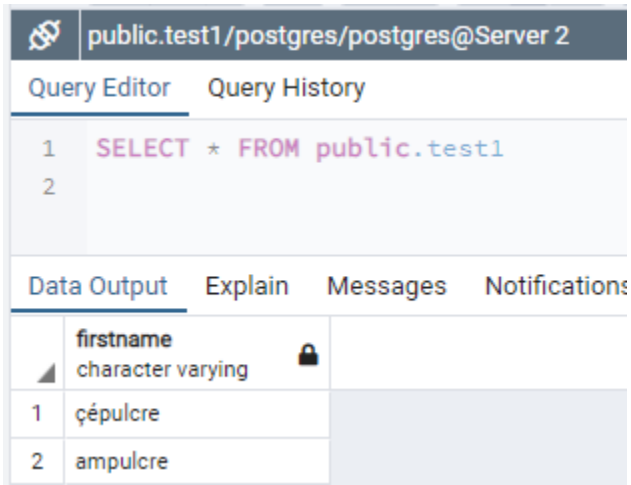
```
docker exec tryton-postgres createdb -U postgres -T template0 postgres-copy
```

Step 3.2 : import inside container (optional ; function of step 1.2 above)

```
docker cp postgres-db-backup.sql tryton-postgres:/postgres-db-backup.sql
```

Step 3.2 : restore **postgres-copy** from **postgres**

```
docker exec -i tryton-postgres psql -U postgres -f postgres-db-backup.sql postgres-copy
```



public.test1/postgres/postgres@Server 2	
Query Editor Query History	
1	SELECT * FROM public.test1
2	
Data Output Explain Messages Notifications	
	firstname
	character varying
1	çépulcre
2	ampulcre

« pg_restore » - Binary Mode - Example in Context

Use « pg_restore » utility, not « psql »

- Before restoring the TRYTON database using the following script, log out the system if you happen to be using it.
- See section hereafter about TRYTON Login / Logout

```
# Step 1.1 : dump tryton
docker exec tryton-postgres pg_dump -Ft -U postgres -O -f postgres-db-backup.tar postgres
# Step 1.2 : export outside container (optional ; specifically use if later import in another container)
docker cp tryton-postgres:/postgres-db-backup.tar postgres-db-backup.tar
# Step 2 : drop and create tryton-copy
docker exec tryton-postgres dropdb -f -U postgres postgres-copy
docker exec tryton-postgres createdb -U postgres -T template0 postgres-copy
# Step 3.2 : import inside container (optional ; function of step 1.2 above)
docker cp postgres-db-backup.tar tryton-postgres:/postgres-db-backup.tar
# Step 3.2 : restore postgres-copy from postgres
docker exec -i tryton-postgres pg_restore -Ft -U postgres -d postgres-copy -v ./postgres-db-backup.tar
```

```
PS C:\Users\mrmar> docker exec tryton-postgres dropdb -f -U postgres postgres-copy
PS C:\Users\mrmar> docker exec tryton-postgres createdb -U postgres -T template0 postgres-copy
PS C:\Users\mrmar> docker exec -i tryton-postgres pg_restore -Ft -U postgres -d postgres-copy -v ./postgres-db-backup.tar
pg_restore: connecting to database for restore
pg_restore: creating TABLE "public.test"
pg_restore: creating TABLE "public.test1"
pg_restore: processing data for table "public.test"
pg_restore: processing data for table "public.test1"
```

Database Operations

Tryton

List tables of database « tryton »

Exec in Powershell format (exec.ps1)

```
Write-Host "Database Tryton - Schema 'Public' - All Tables - Number of Rows"
Write-Host "init"
docker cp inpu.sql tryton-postgres:/inpu.sql
docker exec -it tryton-postgres psql -d tryton -U postgres -P pager=off -f inpu.sql -o outp.txt
docker cp tryton-postgres:/outp.txt outp.txt
Write-Host "fini"
```

SQL statements (inpu.sql)

```
-- Data base schema 'public' - All tables - Number of rows
with tbl as (
    SELECT table_schema, table_name
    FROM information_schema.tables
    where table_name not like 'pg_%' and table_schema in ('public')
)
SELECT table_schema, table_name,
(
    xpath( '/row/c/text()', query_to_xml( format('select count(*) as c from %I.%I', table_schema, table_name), false, true, '' ))
)[1]::text::int as rows_nmbr
from tbl
ORDER BY 3 DESC, 2; /* ORDER BY 3 DESC; */ /* ORDER BY 2; */
```


List tables of database « tryton »

File Explorer: Marc Rottiers > tryt01

Name	Date modified	Type
exec.ps1	21/02/2021 10:54	Windows PowerSh...
inpu.sql	21/02/2021 10:44	SQL File
outp.txt	21/02/2021 10:55	TXT File

Terminal Output:

```
1 Write-Host "Database Tryton - Schema 'Public' - All Tables - Number of Rows"
2 Write-Host "init"
3 docker cp inpu.sql tryton-postgres:/inpu.sql
4 docker exec -it tryton-postgres psql -d tryton -U postgres -P pager=off -f inpu.sql | Out-File -Encoding utf8 outp.txt
5 Write-Host "fini"
```

SQL File Content (inpu.sql):

```
1 -- Data base schema 'public' - All tables - Number of rows
2
3
4 with tbl as (
5     SELECT table_schema,
6            table_name
7     FROM information_schema.tables
8     where table_name not like 'pg_%' and table_schema in ('public')
9 )
10 select table_schema,
11        table_name,
12        (
13            xpath(
14                '/row/c/text()',
15                query_to_xml(
16                    format('select count(*) as c from %I.%I', table_schema, table_name), false, true, ''
17                )
18            )
19        )[1]::text::int as rows_nmbr
20 from tbl
21 ORDER BY 3 DESC; /* ORDER BY 3 DESC; */ /* ORDER BY 2; */
```

Table List Output (outp.txt):

table_schema	table_name	rows_nmbr
public	ir_translation	2439
public	ir_model_field	1965
public	ir_model_data	1464
public	ir_action	643
public	ir_action_keyword	274
public	ir_ui_menu	182
public	ir_action_act_window	177
public	party_address_format	169
public	ir_rule	155
public	ir_ui_menu-res_group	151
public	ir_rule_group	147
public	ir_action_wizard	135
public	ir_model_button	121
public	ir_lang	102
public	ir_cache	69
public	ir_ui_icon	50
public	account account type template	49
public	account account type template	38
public	account account type template	36
public	account account type template	25
public	account account type template	23
public	account account type template	16
public	account account type template	16
public	account account type template	16

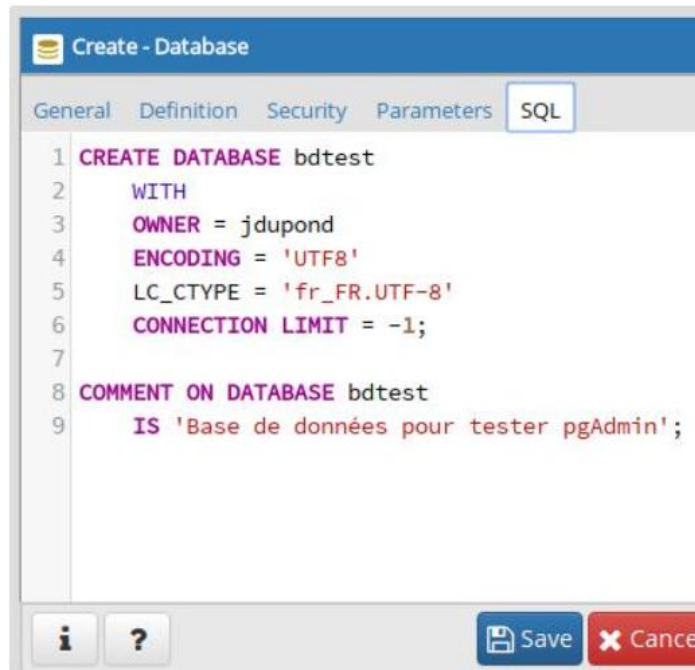
Postgres

Database creation

```
PS C:\Users\mrmar> docker exec -tiu postgres dev-postgres psql -c '\l+'
List of databases

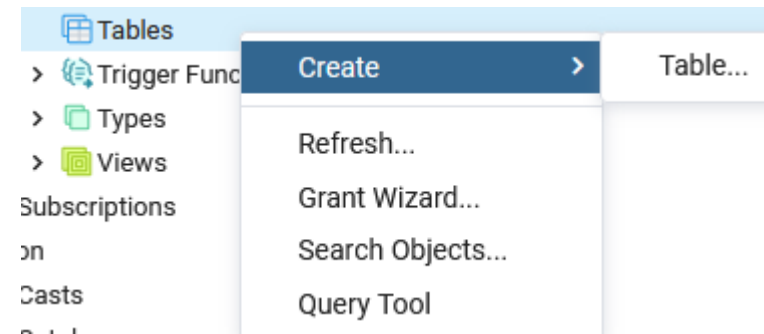
```

Name	Owner	Encoding	Collate	Ctype	Access privileges
postgres	postgres	UTF8	en_US.utf8	en_US.utf8	
template0	postgres	UTF8	en_US.utf8	en_US.utf8	=c/postgres +
template1	postgres	UTF8	en_US.utf8	en_US.utf8	postgres=CTc/postgres +



```
CREATE DATABASE test1
WITH
OWNER = postgres
ENCODING = 'UTF8'
LC_COLLATE = 'en_US.UTF-8'
LC_CTYPE = 'en_US.UTF-8'
TABLESPACE = pg_default
CONNECTION LIMIT = -1;
```

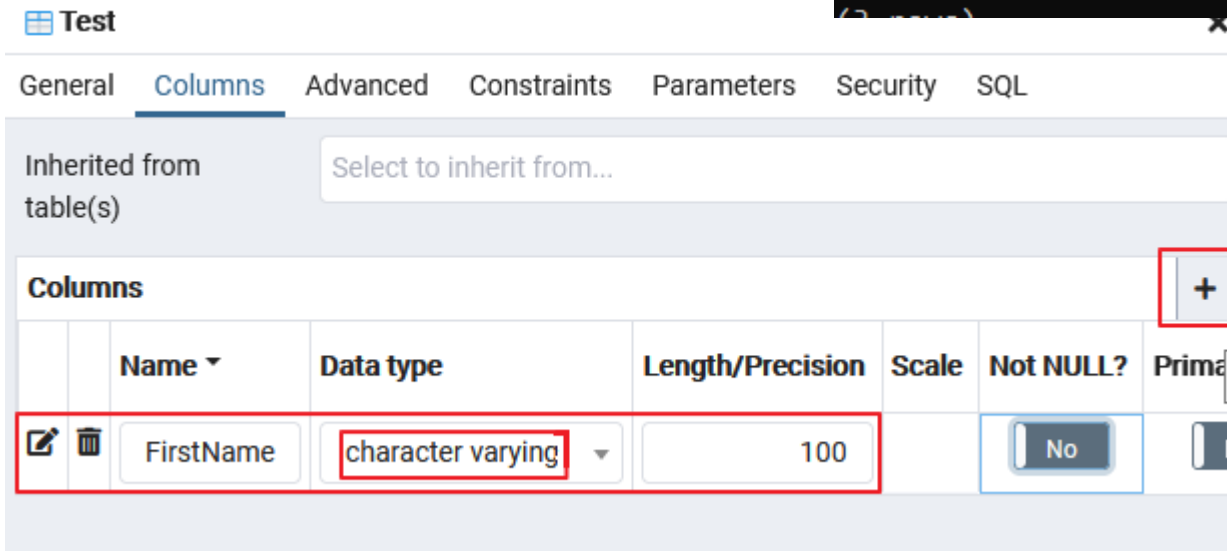
Table creation



```
PS C:\Users\mrmar> docker exec -tiu postgres dev-postgres psql -c '\l+'
List of databases

```

Name	Owner	Encoding	Collate	Ctype	Access privileges
postgres	postgres	UTF8	en_US.utf8	en_US.utf8	
template0	postgres	UTF8	en_US.utf8	en_US.utf8	=c/postgres
template1	postgres	UTF8	en_US.utf8	en_US.utf8	=c/postgres



Populate a sample UTF8 table

```
CREATE TABLE person(  
    personID int,  
    firstname varchar(255)  
);
```

```
INSERT INTO person (personID, firstname)  
VALUES (1, 'Tom B. Erichsençé');
```

```
DELETE FROM person  
WHERE firstname NOT IN  
( 'ampulcre', 'çépulcre' );
```

```
INSERT INTO person (personID, firstname)  
VALUES (2, 'ampulcrê');
```

```
SELECT * from person;
```

postgres/postgres@Server 2 ▾

Query Editor Query History

```
create table Person(  
    PersonID int,  
    LastName varchar(255)  
);  
  
INSERT INTO Person (PersonID, LastName)  
VALUES (1, 'Tom B. Erichsen');
```

Query Editor Query History

```
1 DELETE FROM test1  
2 WHERE firstname NOT IN ('ampulcre','çépulcre') ;  
3  
4 INSERT INTO test1 (firstname)  
5 VALUES ('ampulcre');  
6  
7 Select * from test1;
```

Data Output Explain Messages Notifications

	firstname character varying	
1	çépulcre	
2	ampulcre	

« pgadmin4 » view on Server Databases

The screenshot displays the pgAdmin 4 web interface. On the left, the 'Browser' pane shows a tree view of database objects. 'Server 2' is selected, and its 'postgres' database is expanded. The 'public' schema is also expanded, showing various object types. The 'Tables (1)' folder is highlighted, and the 'person' table is visible below it. On the right, the 'Query Editor' pane shows a SQL script with two statements: a 'CREATE TABLE' statement for 'Person' and an 'INSERT INTO' statement. The script is as follows:

```
1 create table Person(  
2     PersonID int,  
3     LastName varchar(255)  
4 );  
5  
6 INSERT INTO Person (PersonID, LastName)  
7 VALUES (1, 'Tom B. Erichsen');
```

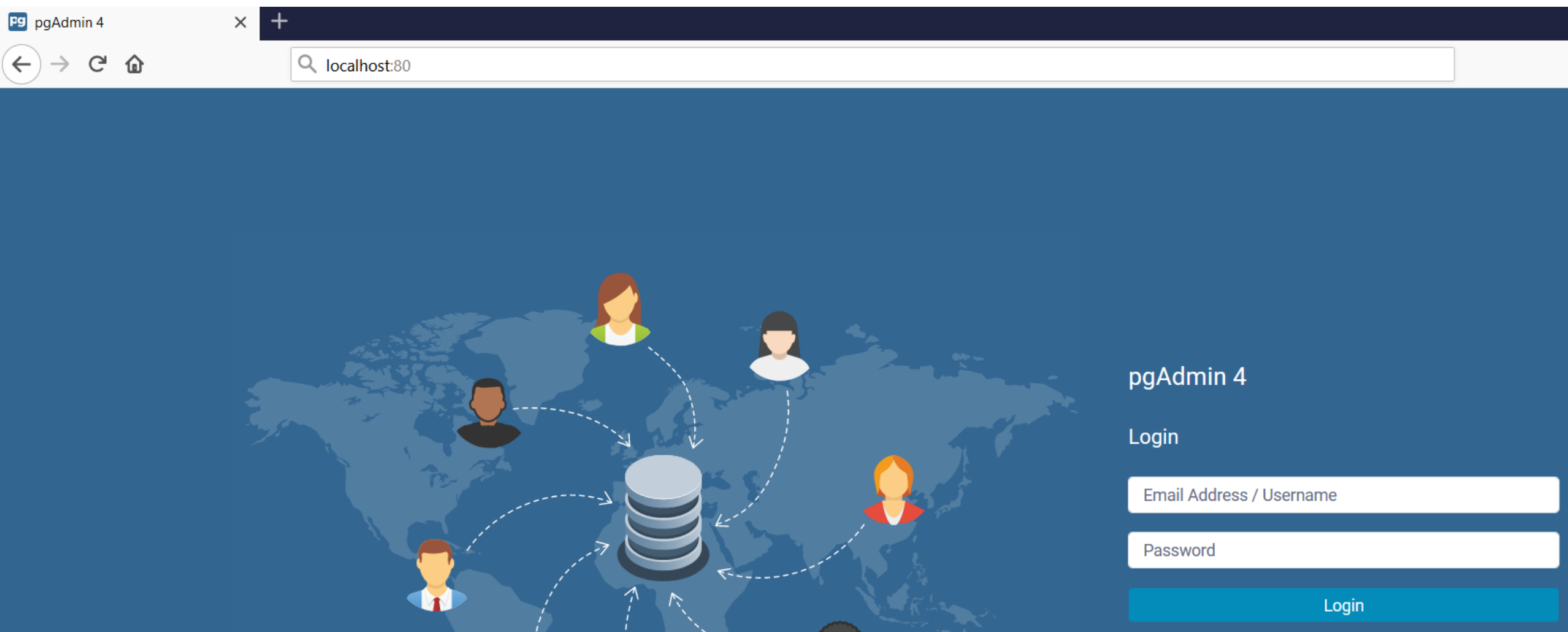
Below the query editor, the 'Messages' tab shows the execution results:

```
INSERT 0 1  
  
Query returned successfully in 78 msec.
```

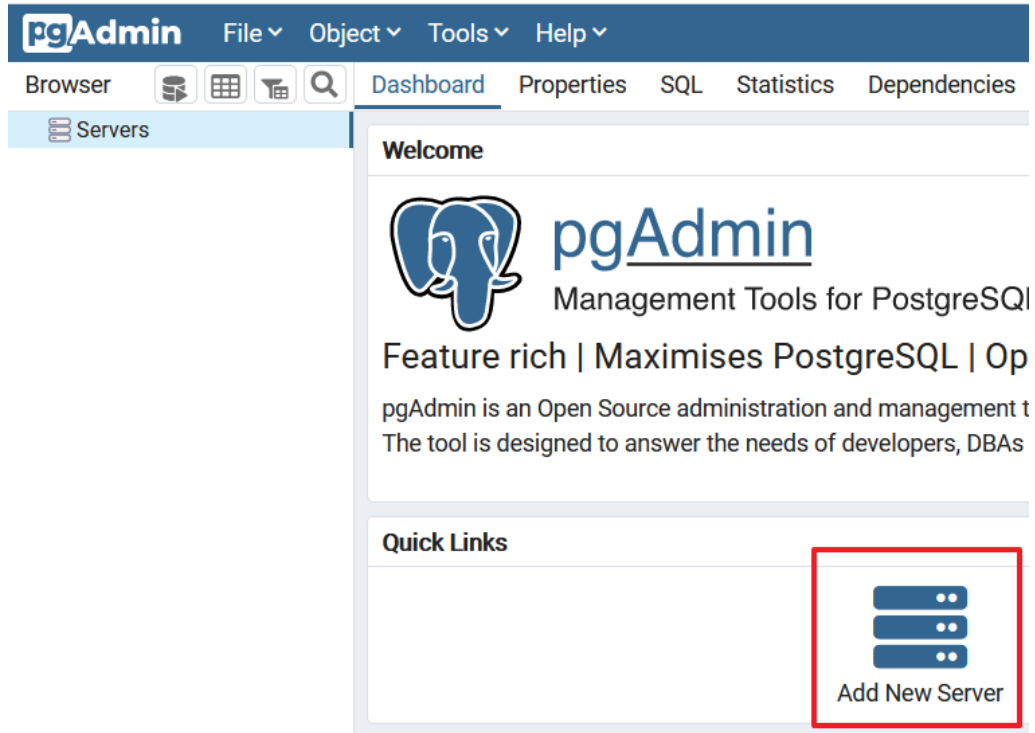
User Interface

PgAdmin4

pgAdmin4



Create servers to connect to the databases



```
PS C:\tryt.01\tryton\tryton.create.backup.restore> docker run --name tryton-postgres --env PGDATA=/var/lib/postgresql/data/pgdata --env POSTGRES_DB=tryton --env POSTGRES_PASSWORD=${env:POSTGRES_PASSWORD} --mount source=tryton-database,target=/var/lib/postgresql/data --network tryton -p 5433:5432 --detach postgres
```

```
PS C:\tryt.01\tryton\tryton.create.backup.restore> docker inspect tryton-postgres -f "{{json .NetworkSettings.Networks }}"
{"tryton":{"IPAMConfig":null,"Links":null,"Aliases":["132372a60d4b"],"NetworkID":"c762438d99a763276487b1cf5abfa8b40fd9d13bcf091f3fe1be54fd57273550","EndpointID":"69be5cb1492545d2fbad8caef953f18c9dd8c2794eb9b41a3a761f8e09ae0f3c","Gateway":"172.18.0.1","IPAddress":"172.18.0.2","IPPrefixLen":16,"IPv6Gateway":"","GlobalIPv6Address":"","GlobalIPv6PrefixLen":0,"MacAddress":"02:42:ac:12:00:02","DriverOpts":null}}
```

75

Create - Server

×

General

Connection

SSL

SSH Tunnel

Advanced

Host name/address

172.17.0.2

Port

5432

Maintenance database

tryton

Username

postgres

Password

●●●●●●●●

Save password?

☒

Role

Service

i

?

✕ Cancel

↺ Reset

💾 Save

76

Server 2

✕

General

Connection

SSL

SSH Tunnel

Advanced

Host name/address

172.17.0.5

Port

5432

Maintenance database

postgres

Username

postgres

Role

Service

i

?

✕ Cancel

↺ Reset

💾 Save

Create servers to connect to the databases - Stability of IP Address

It might be necessary to « adjust » the IP address from time to time in « pgAdmin4 »

Connect to Server

Please enter the password for the user 'postgres' to connect the server - "Server 1"

Password

☐ Save Password

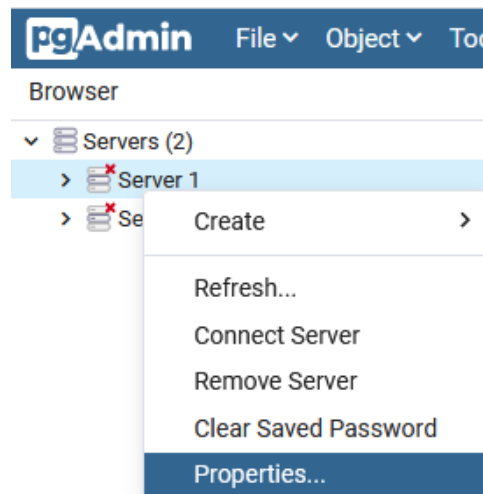
FATAL: database "tryton" does not exist

```
Select Windows PowerShell

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\mrmar> docker inspect tryton-postgres -f "{{json .NetworkSettings.Networks }}" #
{"bridge":{"IPAMConfig":null,"Links":null,"Aliases":null,"NetworkID":"580d205d51c6a151f208bf4
4d05996abdcbb2","EndpointID":"736b1bb8b5da4334b0074479028798f84d53cdd975fe0a34eaf57246310c29
IPAddress":"172.17.0.4","IPPrefixLen":16,"IPv6Gateway":"","GlobalIPv6Address":"","GlobalIPv6
2:42:ac:11:00:04","DriverOpts":null}}
PS C:\Users\mrmar>
```



Server 1

General Connection SSL SSH Tunnel Advanced

Host name/address

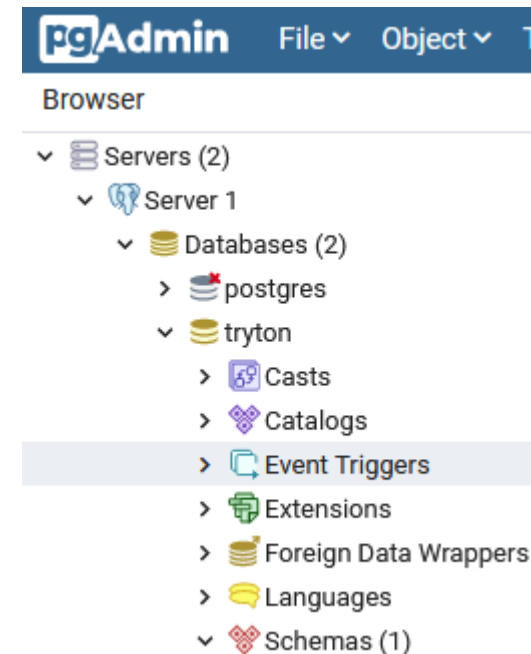
Port

Maintenance database

Username

Role

Service



« Tryton » - Database Result

Only « ir » & « res » tables are installed

- tryton
 - > Casts
 - > Catalogs
 - > Event Triggers
 - > Extensions
 - > Foreign Data Wrappers
 - > Languages
 - > Schemas (1)
 - public
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Procedures
 - > Sequences
 - Tables (69)

- Tables (69)
 - ir_action
 - ir_action-res_group
 - ir_action_act_window
 - ir_action_act_window_domain
 - ir_action_act_window_view
 - ir_action_keyword
 - ir_action_report
 - ir_action_url
 - ir_action_wizard
 - ir_attachment
 - ir_cache
 - ir_calendar_day
 - ir_calendar_month
 - ir_configuration
 - ir_cron
 - ir_email
 - ir_email_address
 - ir_email_template
 - ir_email_template-ir_action_report
 - ir_export
 - ir_export-res_group
 - ir_export-write-res_group
 - ir_export_line
 - ir_lang
 - ir_message
 - ir_model
 - ir_model_access
 - ir_model_button
 - ir_model_button-button_reset
 - ir_model_button-res_group
 - ir_model_button_click
 - ir_model_button_rule
 - ir_model_data
 - ir_model_field
 - ir_model_field_access
 - ir_module
 - ir_module_config_wizard_item
 - ir_module_dependency
 - ir_note
 - ir_note_read
 - ir_queue
 - ir_rule
 - ir_rule_group
 - ir_rule_group-res_group
 - ir_sequence
 - ir_sequence_strict
 - ir_sequence_type
 - ir_sequence_type-res_group
 - ir_session
 - ir_session_wizard
 - ir_translation
 - ir_trigger
 - ir_trigger__history
 - ir_trigger_log
 - ir_ui_icon
 - ir_ui_menu
 - ir_ui_menu-res_group
 - ir_ui_menu_favorite
 - ir_ui_view
 - ir_ui_view_search
 - ir_ui_view_tree_state
 - ir_ui_view_tree_width
 - res_group
 - res_user
 - res_user-ir_action
 - res_user-res_group
 - res_user_application
 - res_user_login_attempt
 - res_user_warning

« Tryton » - Database Result

>  res_user

Data Output		Explain	Messages	Notifications																		
	id [PK] integer		name character varying		active boolean		login character varying		password character varying		create_date timestamp without time zone		create_uid integer		email character varying		language integer		menu integer		password_hash character varying	
1		0	Root		false		root		[null]		2021-03-07 14:58:18.230466			0	[null]		[null]		2		[null]	
2		1	Administrator		true		admin		[null]		2021-03-07 14:58:17.920657			0	@gmai...		[null]		2		\$2b\$12\$7RE0EAyOog8...	

« Postgres » - Result


- ▼ Servers (2)
 - ▼ Server 1
 - ▼ Databases (2)
 - > postgres
 - > tryton
 - > Login/Group Roles
 - > Tablespaces
 - ▼ Server 2
 - ▼ Databases (1)
 - > postgres
 - > Login/Group Roles
 - > Tablespaces


- ▼ postgres
 - > Casts
 - > Catalogs
 - > Event Triggers
 - > Extensions
 - > Foreign Data Wrappers
 - > Languages
 - ▼ Schemas (1)
 - ▼ public
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Procedures
 - > 1..3 Sequences
 - > Tables
 - > Trigger Functions
 - > Types
 - > Views

Tryton

Login / Logout

Login into Tryton

 localhost:8000|

 Tryton - localhost:8000

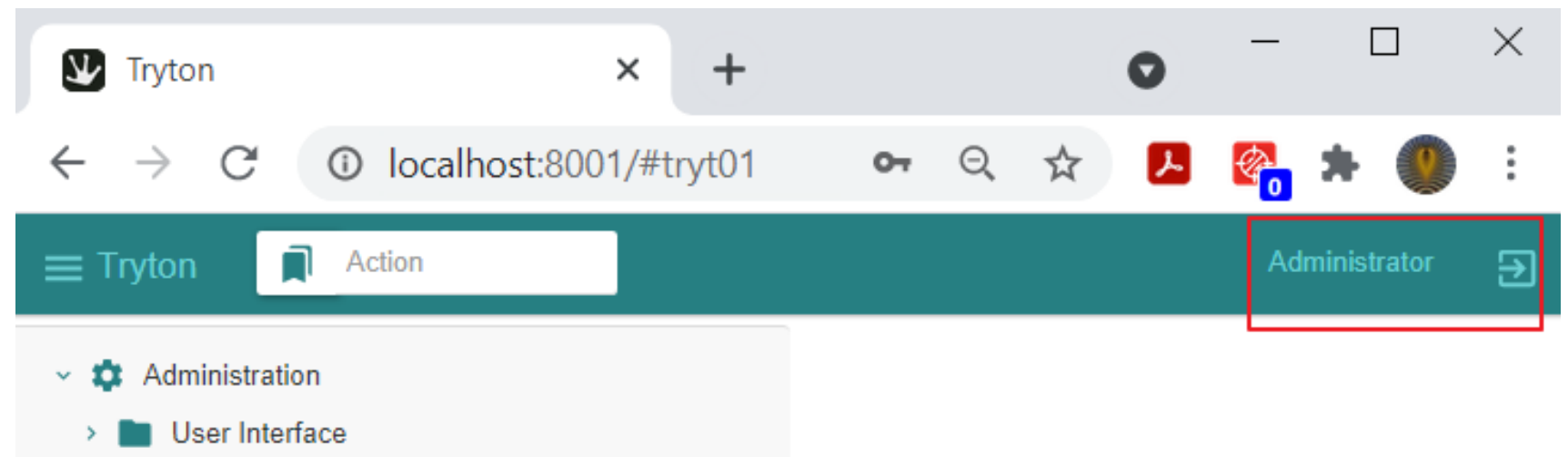
Login

Database
tryton

User name
admin

LOGIN

In this introductory document, all operations are performed with user « admin » and password « Password » at login.



Usage

Using Tryton

The follow-up document « Tryton 5.8 - Doc 05.01 - Basic Entities » explains how to use Tryton

References

Various sources of documentation

Documentation Latest

[<https://docs.tryton.org/en/latest>]

Docker Installation

<https://hub.docker.com/r/tryton/tryton/>

Classic Installation

[<https://blog.lordvan.com/blog/tryton-setup-config/>]

[<https://www.akarei.cz/tryton/>]

Administration Manual

[<https://readthedocs.org/projects/tryton-administration-manual/downloads/pdf/latest/>]

[https://tryton-administration-manual.readthedocs.io/_/downloads/en/latest/pdf/]

List of Modules

[<https://discuss.tryton.org/t/list-of-modules-and-what-they-do/2675/7>]

Stock

[<https://groups.google.com/g/tryton/c/H4ZqsJq37M8/m/W1TaVWu0AQAJ>]

[<https://docs.tryton.org/en/latest/stock.html#index-stock>]

Various sources of documentation

Trytond Documentation

[<https://readthedocs.org/projects/trytond/downloads/pdf/latest/>]

[<https://trytond.readthedocs.io/en/latest/>]

[<https://tryton.readthedocs.io/en/latest/>]

[<http://hg.tryton.org/readthedocs/>]

[<https://docs.readthedocs.io/en/latest/subprojects.html>]

[https://docs.readthedocs.io/en/latest/alternate_domains.html]

Other sources

Github

[<https://github.com/tryton>]

Downloads

[<https://downloads.tryton.org/>]