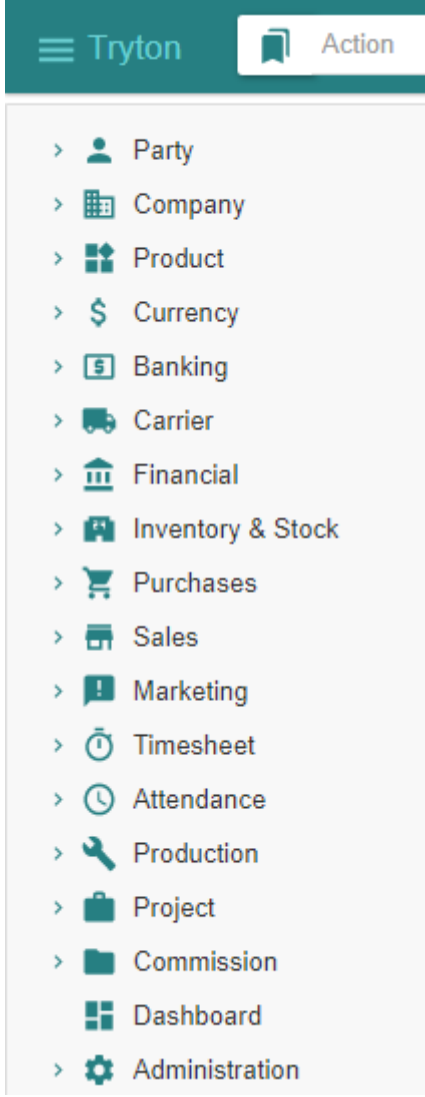
Tryton By Example

-

Getting started

-

Administration



Github - https://github.com/clixwise/tryton-by-example
Version of presentation : 6.0
Version of Tryton : 6.0
Verified for execution on : Windows 10 & Powershell 7

Author : Marc Rottiers

Foreword

This presentation aims to expedite the process of learning the basics of the TRYTON ERP. It rests on a personal initiative. The content does not replace official TRYTON documentation in any manner.

First-time end-users who want to explore this package should benefit the most. System administration aims at running a demonstration environment without further consideration for performance, security, etc. Production-grade system setup and usage will probably differ from exposed techniques that are meant to keep the explanations as concise as possible.

The material relates to TRYTON 6.0 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 given information.

Are described an *installation procedure* as well as some *use cases* by example. There are explanatory documents as well as accompanying database samples and ancillary scripts.

The author acknowledges documentation that he had the opportunity to analyse for the purpose of creating the present material. Special credit and thanks to @ced, @pokoli, @dave, @edbo who provide support on the https://discuss.tryton.org/ forum.

Feedback is appreciated. Please post on https://github.com/clixwise/tryton-by-example

Structure of material

Topics
Tryton 6.0 - Doc 00.01 - Installation & administration
Tryton 6.0 - Doc 05.01 - Basic functionality
Tryton 6.0 - Doc 10.01 - Purchase
Tryton 6.0 - Doc 15.01 - Sale
Tryton 6.0 - Doc 80.01 - Ancillaries

Structure

- Database snapshots
- Utilities
- Tryton 6.0 - Doc 00.01 - Installation & administration.pdf
- Tryton 6.0 - Doc 00.01 - Installation & administration.pptx
- Tryton 6.0 - Doc 05.01 - Basic functionality.pdf
- Tryton 6.0 - Doc 05.01 - Basic functionality.pptx
- Tryton 6.0 - Doc 10.01 - Purchase.pdf
- Tryton 6.0 - Doc 10.01 - Purchase.pptx
- Tryton 6.0 - Doc 10.01 - Purchase.xlsx
- Tryton 6.0 - Doc 15.01 - Sale.pdf
- Tryton 6.0 - Doc 15.01 - Sale.pptx
- Tryton 6.0 - Doc 15.01 - Sale.xlsx
- Tryton 6.0 - Doc 80.01 - Ancillaries.pdf
- Tryton 6.0 - Doc 80.01 - Ancillaries.pptx

> Database snapshots

Name

- Tryton 6.0 - Doc 00.01.tryt01-db-backup.tar
- Tryton 6.0 - Doc 05.01.tryt01-db-backup.tar
- Tryton 6.0 - Doc 10.01.tryt01-db-backup.tar
- Tryton 6.0 - Doc 15.01.tryt01-db-backup.tar

> Utilities

Name

- query.dbms_01.sql
- query.dbms_02.sql
- query.res_user.sql
- Tryton 6.0 - Doc 00.01 - Installation & administration.database.post01.backup.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.database.post01.restore.binary.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.database.post01.restore.character.createNot.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.database.post01.restore.character.createYes.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.database.tryt11.backup.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.database.tryt11.query.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.database.tryt11.restore.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.docker.post01.create.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.docker.post01.delete.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.docker.post01.start.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.docker.post01.status.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.docker.post01.stop.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.docker.status.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.create.permanent.docker.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.create.permanent.windows.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.create.volatile.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.delete.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.start.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.status.ps1
- Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.stop.ps1
- Tryton 6.0 - Doc 05.01 - Basic functionality.import.tryt11.countries.ps1
- Tryton 6.0 - Doc 05.01 - Basic functionality.import.tryt11.currencies.ps1

Slides

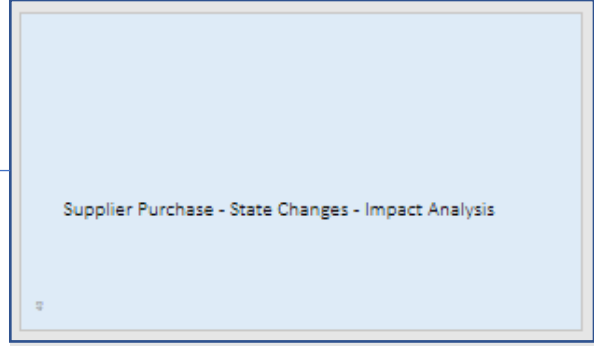
We use the following color conventions to separate the different (sub)sections in each presentation

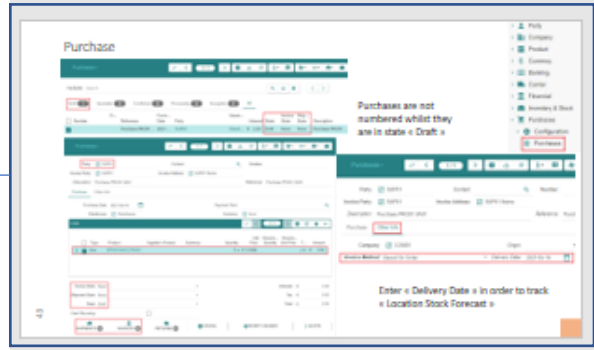
Level 1 section - Heading

Level 2 section - Heading

Level 3 section - Heading

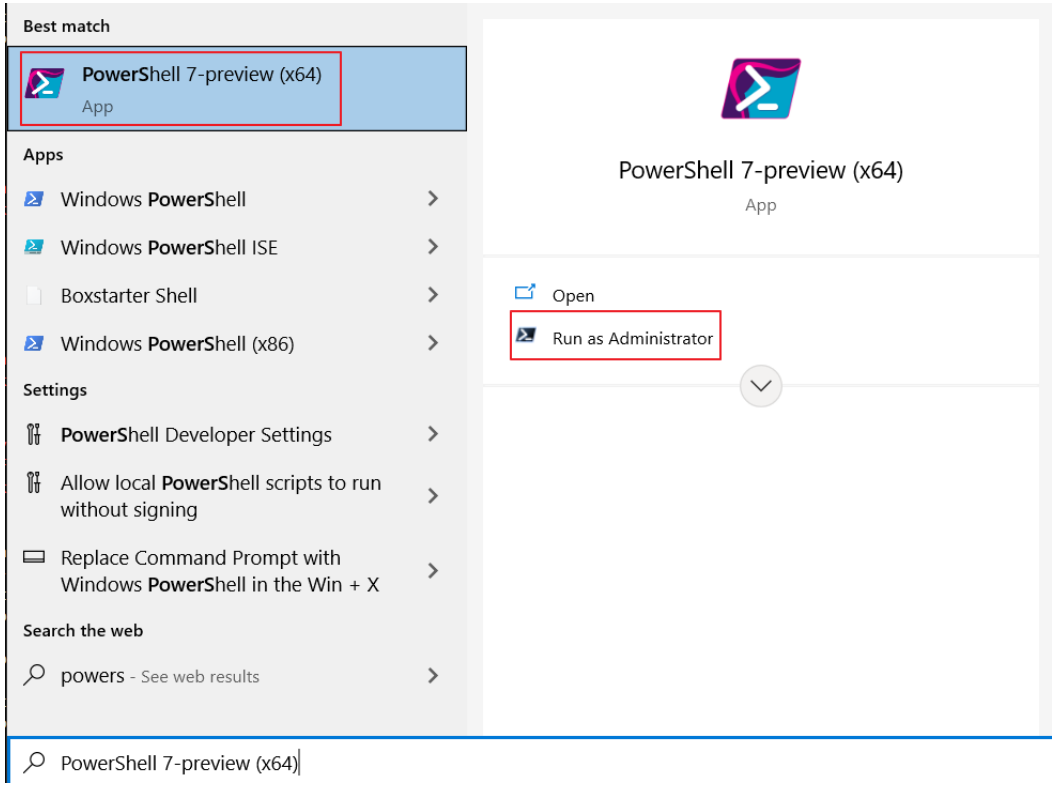
Level 4 section - Text





Scripts

- Powershell scripts « *.ps1 » are provided in the folder « Utilities » to support routine operations
- Powershell can be accessed as shown below
- Each script needs some tuning e.g. with respect to the database name, etc.
- Powershell scripts are executed as follows :
  - In their folder e.g. : ./"Tryton 6.0 - Doc 01.01 - Installation & administration.docker.status"
  - In File Explorer : « Run with Powershell »



Database snapshots

We have taken a snapshot at the end of each presentation section (Basic Functionality, Purchase, etc.)

File name	
Tryton 6.0 - Doc 00.01.tryt01-db-backup.tar	Database state at end of « 00.01 »
Tryton 6.0 - Doc 05.01.tryt01-db-backup.tar	Database state at end of « 05.01 »
Tryton 6.0 - Doc 10.01.tryt01-db-backup.tar	Database state at end of « 10.01 »
Tryton 6.0 - Doc 15.01.tryt01-db-backup.tar	Database state at end of « 15.01 »

Table of Contents

TOC

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

CONTAINERS

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 outside of container
Container Management	**How to manage containers**
Container Uninstallation	**How to uninstall containers**
Container Multi-versioning	**How to install containers from multiple image versions**
System Reboot	**How to proceed after system reboot**

CLIENT

User Interface	**Interface to TRYTON & PGADMIN**
PgAdmin4	Setting up & Exploring the pgadmin4 interface
Tryton	Logging & Logout

DATABASES

Operations	
Tryton & Postgres	Working with the database
Backup	
Tryton & Postgres	Backing up the database (UTF-8 compliant)
Restore	
Tryton & Postgres	Restoring the database (UTF-8 compliant)
Multi-database Container	
Tryton	Managing multiple databases in a Database Container

SUMMARY

Next	**Next topics**
Issues	**Documentation points still to be resolved**
References	**Links of interest**

Docker Installation

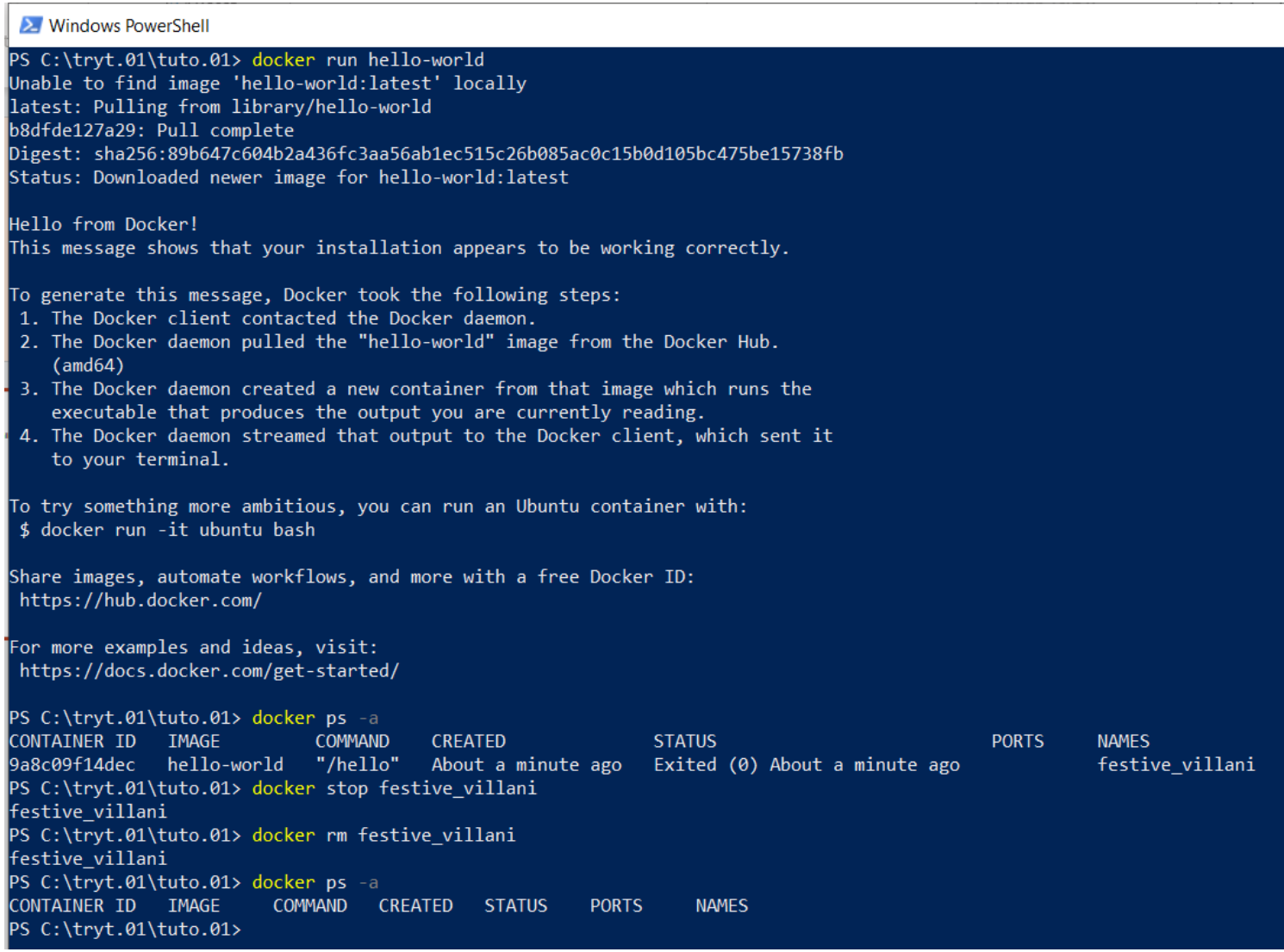
Installation

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 »



Motivation

- The aim for this presentation is to explore the

Container Installation

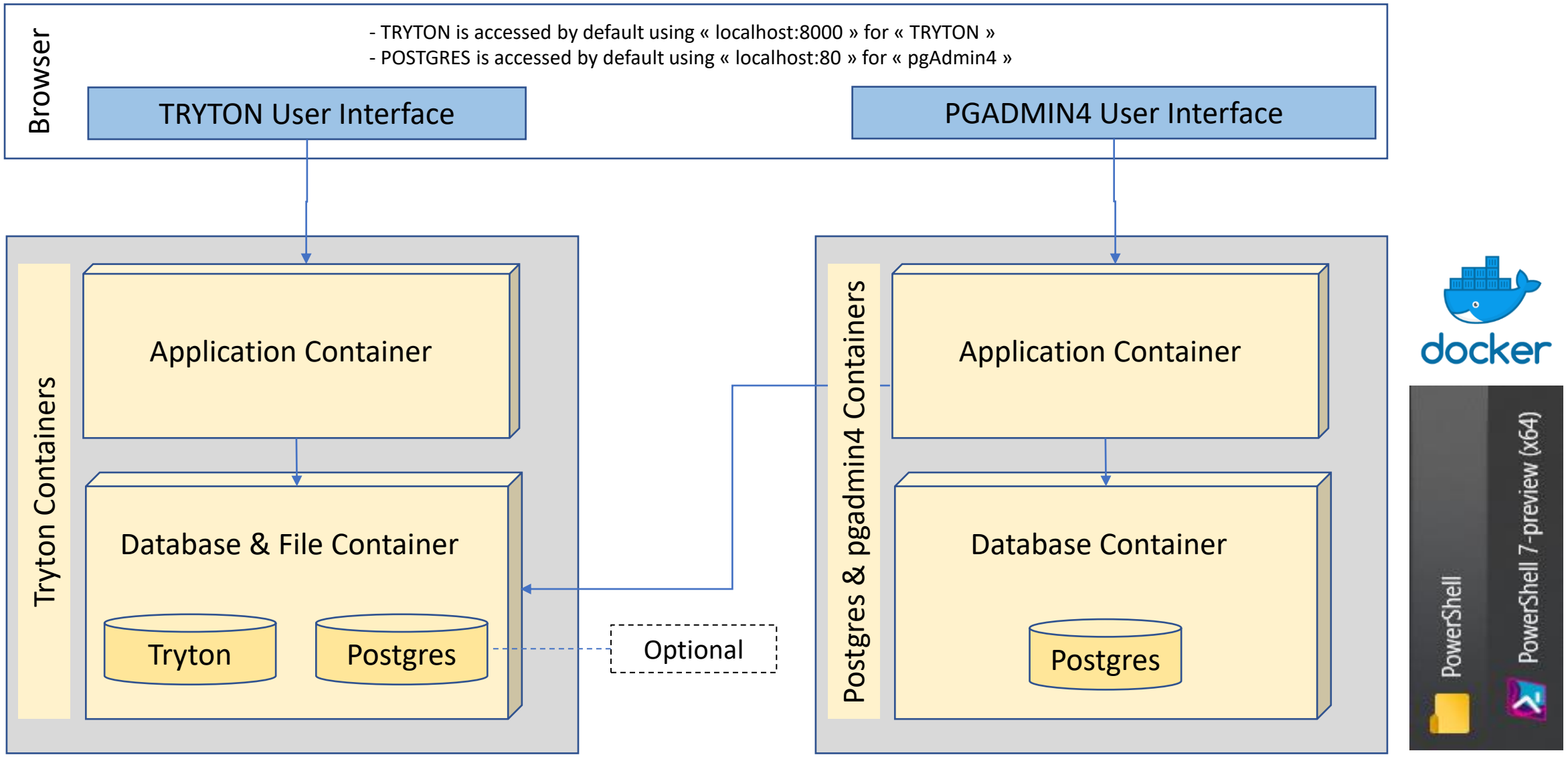
Motivation

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 helps you practice the TYTON system
- Series 2 is optional as it only helps understanding how TRYTON uses the underlying database

Structure



TRYTON Docker Structure

POSTGRES Docker Structure (optional)

Docker Image

- Docker Containers are installed from Docker Images
- Docker Images are « pulled » from a Docker Hub

- This presentation pertains to TRYTON 6.0
- So make sure to « pull » the correct image version by using : « docker pull tryton/tryton:6.0 »
- If you do « docker pull tryton/tryton », you pull the « latest » image version (tag)
- Verify using « docker image ls »

```
PS C:\Users\mrmar\docker.tryton.6.0> docker pull tryton/tryton:6.0
6.0: Pulling from tryton/tryton
f7ec5a41d630: Already exists
3e9c95f22a30: Pull complete
0dde2e82bead: Pull complete
af51f3e524f6: Pull complete
92e498c64da8: Pull complete
87d10cda06b2: Pull complete
4ccc30c1867a: Pull complete
22ae100fbf9d: Pull complete
Digest: sha256:be7c3815facfd538bc929085148adf39fb04b4894a1c0b57e73c
Status: Downloaded newer image for tryton/tryton:6.0
docker.io/tryton/tryton:6.0
```

```
PS C:\Users\mrmar\docker.tryton.6.0> docker image ls
REPOSITORY                  TAG       IMAGE ID       CREATED         SIZE
tryton/tryton               6.0       9e17644cc375   4 days ago      448MB
my-nginx                    latest    8a875bd3beb1   6 days ago      22.6MB
dpage/pgadmin4              latest    048c641d6e64   2 weeks ago     244MB
tryton/tryton               latest    77f3902ebc4c   3 weeks ago     460MB
```

Commands

https://docs.docker.com/reference/

Check installation	
docker run hello-world	Check installation is operational

Container commands	
docker ps -a	List containers
docker stop a_container_name	Stop a container
docker start a_container_name	Start a container
docker rm a_container_name	Remove a stopped container
docker logs a_container_name	Output the logs produced by the container
docker container prune	Remove stopped containers

```
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	
docker volume create a_volume_name	Create a volume
docker volume ls	List the volumes
docker volume rm a_volume_name	Remove a volume
docker volume prune	Remove all unreferenced volumes. « Unreferenced » = not referenced by container
docker volume rm	Remove all unused volumes. « Unused » = not used by container
docker volume inspect a_volume_name	Inspect a volume

Volume commands	
docker network create a_network_name	Create a network
docker network ls	List the networks
docker network rm a_network_name	Remove a network
docker network prune	Remove all unreferenced networks. « Unreferenced » = not referenced by container
docker network rm	Remove all unused networks. « Unused » = not used by container
docker network inspect a_network_name	Inspect a network

Commands

Other commands	
docker system prune	Remove all unused containers, networks, images (both dangling and unreferenced), and optionally, volumes.

```
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
```

Docker Containers and their Host Environment

Installing TRYTON ERP Containers

The TRYTON « Database & File Container » can be installed in one of two ways with respect to database and files permanence :

- In a « permanent » setup. It means that the database and any « attachment » files are maintained in the host environment i.e. in the Windows file system and not in the container itself. They will thus remain available when the corresponding Docker container is removed, accidentally or not.

- In a « semi-permanent » setup. 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

The same remark applies with respect to whether the database is stored in the container or in a mounted volume.

Convention about password names

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

Tryton - « Permanent » Data

Principle

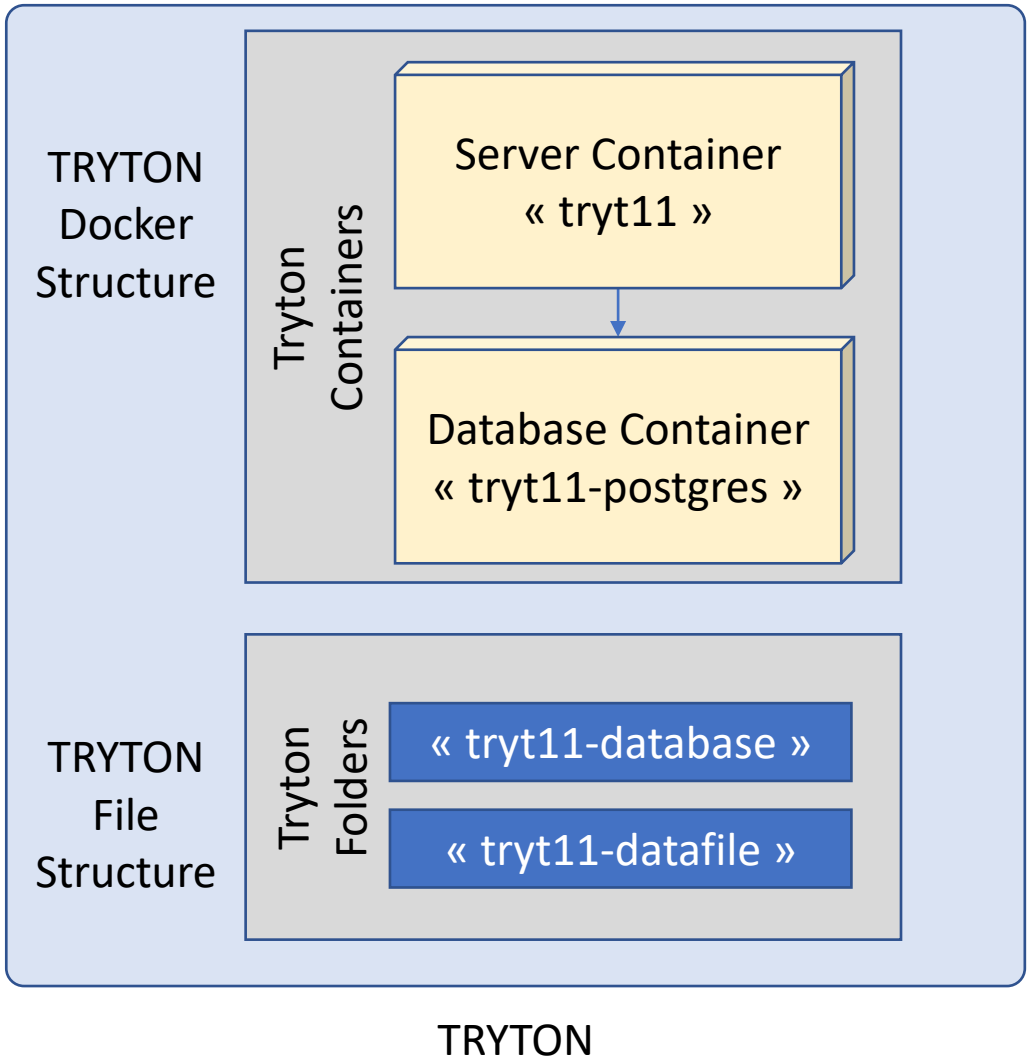
Install :

1. A container « **tryt11-postgres** » from docker image « postgres » ;
   « **-p 5434:** » can be changed (default : 5432)
2. A container « **tryt11** » from docker image « tryton/tryton » ;
   « **-p 8001:** » can be changed (default : 8000)
3. Two volumes : « **tryt11-database** » & « **tryt11-datafile** »
4. One network : « **tryt11-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 « **tryt11-database** » with respect to the directory (Get-Location) where the Powershell script executes
- Subfolder « **tryt11-datafile** » with respect to the directory (Get-Location) where the Powershell script executes



TRYTON

Refer to :

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

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

Scripts

Docker Powershell Commands - Run as ./"..."	
Tryton 6.0 - Doc 00.01 - Installation & administration.docker.status	Query status
Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.create.permanent.docker (*)	Create « tryt11 » containers Permanent data when containers removed Permanent data inside docker volume (faster)
Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.create.permanent.windows (*)	Create « tryt11 » containers Permanent data when containers removed Permanent data inside windows volume (slower)
Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.create.volatile	Create « tryt11 » containers Volatile data when containers removed
Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.start	Start « tryt11 » containers
Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.stop	Stop « tryt11 » containers
Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.status	Query status of « tryt11 » containers
Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.delete	Delete « tryt11 » containers

« Docker » database and file permanency is recommended for two reasons :

- Database handling by the application system is faster when stored on a Docker volume
- Accessing files on a Windows volume from inside a Docker container requires a cross-environment user authorisation setup

./"Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.create.permanent.docker"

```
docker pull tryton/tryton:6.0

docker network create tryt11-network
docker volume create tryt11-database
docker volume create tryt11-datafile
$POSTGRES_PASSWORD ="Password"

# tryt11 database container
docker run --name tryt11-postgres --env PGDATA=/var/lib/postgresql/data/pgdata --env POSTGRES_DB=tryt11 --env
POSTGRES_PASSWORD=${POSTGRES_PASSWORD} --mount source=tryt11-database,target=/var/lib/postgresql/data --network tryt11-network
-p 5443:5432 --detach postgres
Start-Sleep -Seconds 30 # Replace by detecting database is 'up'
# docker exec -tiu postgres tryt11-postgres psql -c '\l+'

# tryt11 transient container to initialize the tryt11 database in its container
docker run --env DB_HOSTNAME=tryt11-postgres --env DB_PASSWORD=${POSTGRES_PASSWORD} --network tryt11-network --interactive --tty
--rm tryton/tryton trytond-admin -d tryt11 --all

# tryt11 server containers : tryt11 & optionally tryt11-cron for scheduled actions
docker run --name tryt11 --env DB_HOSTNAME=tryt11-postgres --env DB_PASSWORD=${POSTGRES_PASSWORD} --mount source=tryt11-
datafile,target=/var/lib/trytond/db --network tryt11-network -p 8011:8000 --detach tryton/tryton
```

Refer to :
[https://discuss.tryton.org/t/how-to-run-tryton-using-docker/3200] with special credits to David Harper

./"Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.create.permanent.windows"

```
docker pull tryton/tryton:6.0

docker network create tryt11-network
$POSTGRES_PASSWORD ="Password"
# Tryton database container – Create
$TRYTON_VOL_DB = (Get-Location).tostring().replace("\","/").replace("C:/","c//") + "/"+"tryt11-database"
# docker volume create tryt11-database - for future use
docker run --name tryt11-postgres --env PGDATA=/var/lib/postgresql/data/pgdata --env POSTGRES_DB=tryt11 --env
POSTGRES_PASSWORD=${POSTGRES_PASSWORD} --volume ${TRYTON_VOL_DB}:/var/lib/postgresql/data --network tryt11-network -p 5443:5432
--detach postgres
Start-Sleep -Seconds 20 # required to wait for postgres to properly connect
docker exec -tiu postgres tryt11-postgres psql -c '\l+'
dir
# Tryton transient container to initialize the tryton database in its container
docker run --env DB_HOSTNAME=tryt11-postgres --env DB_PASSWORD=${POSTGRES_PASSWORD} --network tryt11-network --interactive --tty --
rm tryton/tryton:6.0 trytond-admin -d tryt11 --all
docker exec -tiu postgres tryt11-postgres psql -c '\l+'
# Tryton server container
$TRYTON_VOL_FI = (Get-Location).tostring().replace("\","/").replace("C:/","c//") + "/"+"tryt11-datafile"
# docker volume create tryt11-datafile - for future use
docker run --name tryt11 --env DB_HOSTNAME=tryt11-postgres --env DB_PASSWORD=${POSTGRES_PASSWORD} --volume
${TRYTON_VOL_FI}:/var/lib/trytond/db --network tryt11-network --publish 127.0.0.1:8011:8000 --detach tryton/tryton:6.0
dir
# Obtain Gateway address for usage in pgadmin4 - creating server
docker inspect tryt11-postgres -f "{{json .NetworkSettings.Networks }}" # "Gateway":"172.18.0.1","IPAddress":"172.18.0.2"
```

Tryton - « Volatile » Data

./"Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.create.volatile"

```
# Obtain version 6.0 of Tryton
docker pull tryton/tryton:6.0

# Tryton database container + database initialization volatile container
docker run --name tryt11-postgres -e POSTGRES_PASSWORD=Password -e POSTGRES_DB=tryt11 -p 5443:5432 -d postgres # Start a PostgreSQL instance
docker run --link tryt11-postgres:postgres -e DB_PASSWORD=Password -it tryton/tryton:6.0 trytond-admin -d tryt11 --all # Define database tables
# Tryton server containers : tryton & optionally tryton-cron for scheduled actions
docker run --name tryt11 -p 8011:8000 --link tryt11-postgres:postgres -e DB_PASSWORD=Password -d tryton/tryton:6.0 # Start a Tryton instance
docker run --name tryt11-cron --link tryt11-postgres:postgres -e DB_PASSWORD=Password -d tryton/tryton:6.0 trytond-cron -d tryt11 # Start a cron
instance

# Obtain Gateway address for usage in pgadmin4 - creating server
docker inspect tryt11-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 choosen
In red, connection points whose external « p:xyz » can be adapted

Note about « docker inspect tryt11-postgres » :
the « Gateway":"172.17.0.1" or the "IPAddress":"172.17.0.2" will be used in « pgAdmin4 » to set up the server
Note about « pgAdmin4 » login (see later) values defined in the script :
- User : « x@gmail.com
- Password : « Password »

```
PS C:\Users\mrmar\docker.tryton.6.0> docker pull tryton/tryton:6.0
6.0: Pulling from tryton/tryton
Digest: sha256:be7c3815facfd538bc929085148adf39fb04b4894a1c0b57e73c4f5dc5dab6ea
Status: Image is up to date for tryton/tryton:6.0
docker.io/tryton/tryton:6.0
PS C:\Users\mrmar\docker.tryton.6.0> docker run --name tryt11-postgres -e POSTGRES_PASSWORD=Password -e POSTGRES_DB=tryt11 -p 5443:5432 -d postgres
30eedf401b967fe9316369df1ad8ba34e17aa9ada8cb08013e1a392ac3b3def2
PS C:\Users\mrmar\docker.tryton.6.0> docker run --link tryt11-postgres:postgres -e DB_PASSWORD=Password --rm -it tryton/tryton:6.0 trytond-admin -d try
"admin" email for "tryt11": x@g.c
"admin" password for "tryt11":
"admin" password confirmation:
PS C:\Users\mrmar\docker.tryton.6.0> docker run --name tryt11 -p 8011:8000 --link tryt11-postgres:postgres -e DB_PASSWORD=Password -d tryton/tryton:6.0
4129f9e3cba8e0ee0b4ae98bb1bcd17feb657943fa30ab71f3bb39067b25f36a
PS C:\Users\mrmar\docker.tryton.6.0> docker inspect tryt11-postgres -f "{{json .NetworkSettings.Networks }}"
{"bridge":{"IPAMConfig":null,"Links":null,"Aliases":null,"NetworkID":"e687284abe301936a529d5be904e98128e93b1461cd8b2e6c956e6cba40d25f6","EndpointID":"5
9bd","Gateway":"172.17.0.1","IPAddress":"172.17.0.2","IPPrefixLen":16,"IPv6Gateway":"","GlobalIPv6Address":"","GlobalIPv6PrefixLen":0,"MacAddress":"02:
PS C:\Users\mrmar\docker.tryton.6.0> docker exec -tiu postgres tryt11-postgres psql -c '\l+'
                                                                    List of databases
   Name    |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges   |  Size   | Tablespace |                Description
-----------+----------+----------+------------+------------+-----------------------+---------+------------+--------------------------------------------
 postgres  | postgres | UTF8     | en_US.utf8 | en_US.utf8 |                       | 7877 kB | pg_default | default administrative connection database
 template0 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +| 7729 kB | pg_default | unmodifiable empty database
           |          |          |            |            | postgres=CTc/postgres |         |            |
 template1 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +| 7729 kB | pg_default | default template for new databases
           |          |          |            |            | postgres=CTc/postgres |         |            |
 tryt11    | postgres | UTF8     | en_US.utf8 | en_US.utf8 |                       | 14 MB   | pg_default |
(4 rows)

PS C:\Users\mrmar\docker.tryton.6.0> docker ps -a
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS                      PORTS                      NAMES
4129f9e3cba8   tryton/tryton:6.0    "/entrypoint.sh uwsg…"   44 seconds ago   Up 42 seconds               0.0.0.0:8011->8000/tcp     tryt11
30eedf401b96   postgres             "docker-entrypoint.s…"   2 minutes ago    Up 2 minutes                0.0.0.0:5443->5432/tcp     tryt11-postgres
```

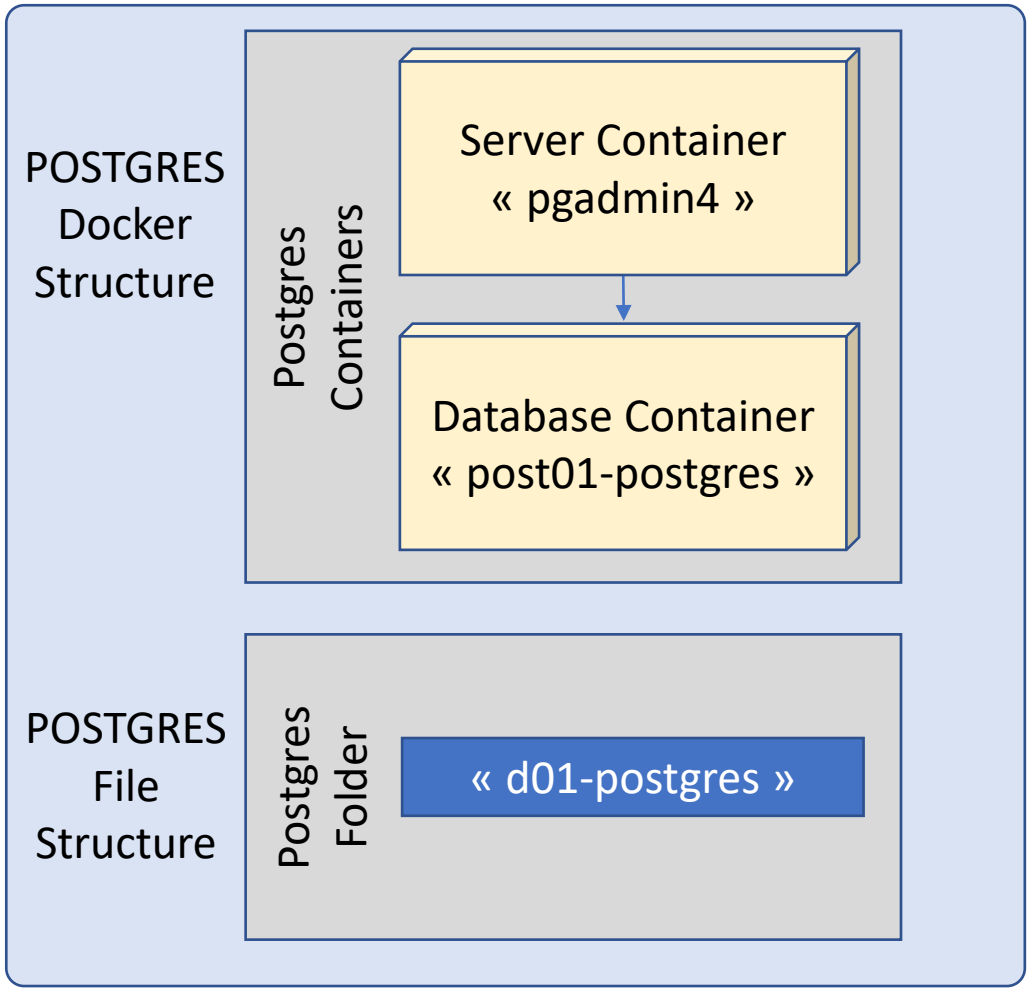
Postgres - « Permanent » Data

Principle

Install :

1. A container « **post01-postgres** » from docker image « postgres » ;
   « **-p 5433:** » can be changed (default : 5432)
2. A container « **post01-pgadmin** » from docker image « dpage/pgadmin4 » ;
   « **-p 81:** » can be changed (default : 80)

The POSTGRES database is stored in subfolder « **post01-postgres** » with respect to the directory ${HOME} == USER



POSTGRES

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

Scripts

Database Powershell Command - Run as ./"..."	
Tryton 6.0 - Doc 00.01 - Installation & administration.docker.post01.create	Create Postgres & Pgadmin Containers
Tryton 6.0 - Doc 00.01 - Installation & administration.docker.post01.delete	Delete Postgres & Pgadmin Containers & Database
Tryton 6.0 - Doc 00.01 - Installation & administration.docker.post01.start	Start Postgres & Pgadmin Containers
Tryton 6.0 - Doc 00.01 - Installation & administration.docker.post01.stop	Stop Postgres & Pgadmin Containers
Tryton 6.0 - Doc 00.01 - Installation & administration.docker.post01.status	Query status of Postgres & Pgadmin Containers

./"Tryton 6.0 - Doc 00.01 - Installation & administration.docker.post01.create"

Install :

- A container « **post01-postgres** » from image « postgres » ; « **-p 5433:** » can be changed
- A container « **post01-pgadmin** » from image « dpage/pgadmin4 » ; « **-p 81:** » can be changed

```
# postgres
docker pull postgres
docker run -d --name post01-postgres -e POSTGRES_PASSWORD=Password -v ${HOME}/postgres-data/:/var/lib/postgresql/data -p 5433:5432 postgres
# pgdamin4
docker pull dpage/pgadmin4
docker run -p 81:80 -e 'PGADMIN_DEFAULT_EMAIL=x@gmail.com' -e 'PGADMIN_DEFAULT_PASSWORD=Password' --name post01-pgadmin -d dpage/pgadmin4
# inspection
docker exec post01-postgres ls /var/lib/postgresql/data
docker exec -tiu postgres post01-postgres psql -c '\l+'
docker inspect post01-postgres -f "{{json .NetworkSettings.Networks }}"
docker inspect -f "{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}" post01-postgres
docker inspect -f '{{.Name}} - {{.NetworkSettings.IPAddress }}' $(docker ps -aq)
```

Note about « docker inspect post01-postgres » :
the « Gateway":"172.17.0.1" or the "IPAddress":"172.17.0.2" (e.g.) will be used in « pgAdmin4 » to set up the server
Note about « pgAdmin4 » login (see later) values defined in the script :
- User : « x@gmail.com
- Password : « Password »

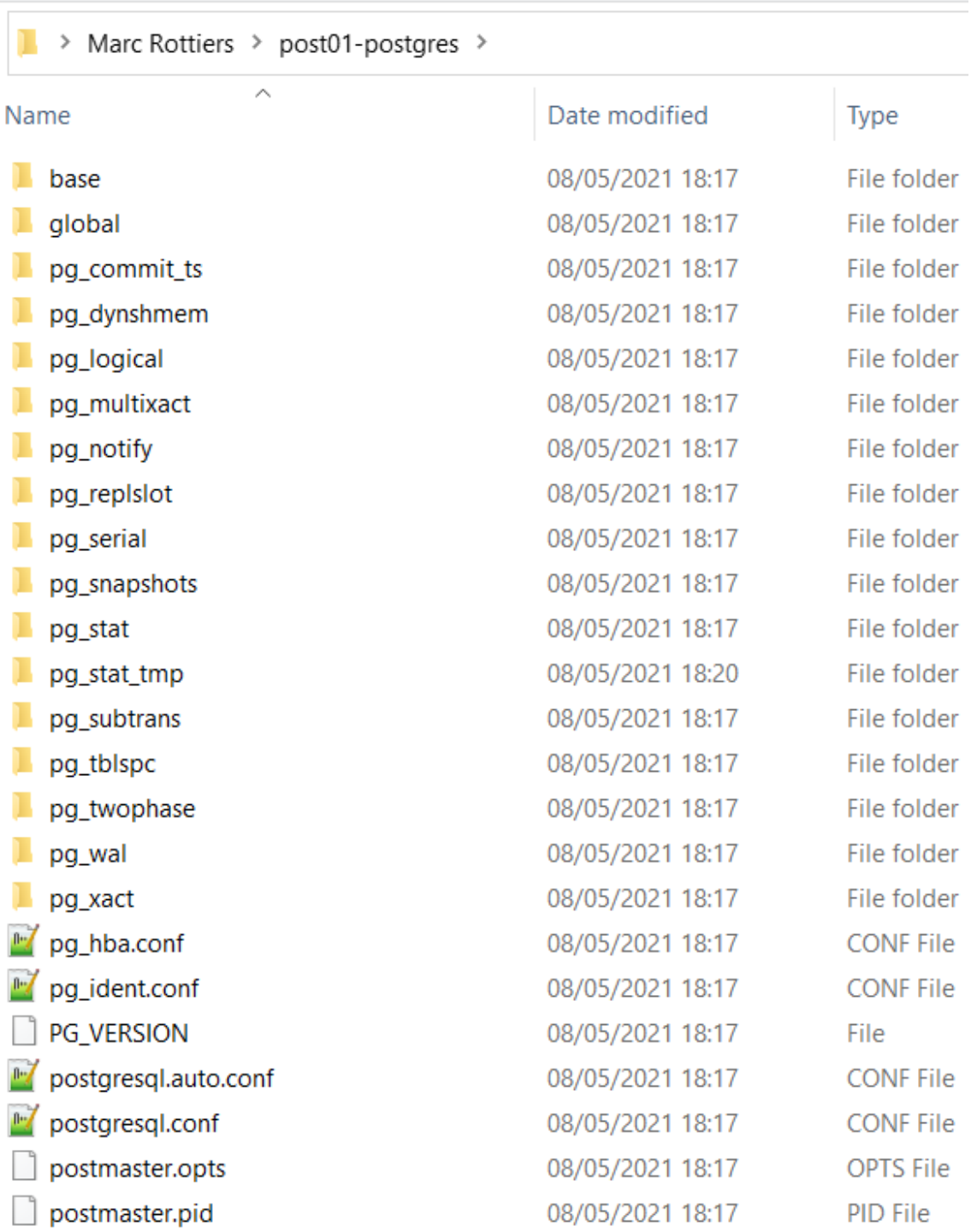
./"Tryton 6.0 - Doc 00.01 - Installation & administration.docker.post01.create"

```
PS C:\Users\mrmar\docker.tryton.6.0> ./"Tryton 6.0 - Doc 00.01 - Installation & administration.docker.post01.create"
-----------------------------------------
1. Create Postgres Container 'post01-postgres'
-----------------------------------------
Using default tag: latest
latest: Pulling from library/postgres
Digest: sha256:61d5d8ef6cb4e2035f053f26b6b455c201a809354084cc8426b6904b8dd35602
Status: Image is up to date for postgres:latest
docker.io/library/postgres:latest
23da037c8b30f17d721269cc35c46efef66c4af7c630a3afc204cad09b09c9e5
-----------------------------------------
2. Create Postgres Container 'post01-pgadmin'
-----------------------------------------
Using default tag: latest
latest: Pulling from dpage/pgadmin4
Digest: sha256:38617bc122e547dcfe3adaba52143f583343928b3700ada6feb9dcf6d13e0ca6
Status: Image is up to date for dpage/pgadmin4:latest
docker.io/dpage/pgadmin4:latest
bbcf2e9704bc1bd69e7fb2f9dce2e74fab5fd3e696d29051ca28e4a9988b5ebc
----------
3. Inspect
----------
```

```
   Name    |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges   |  Size   | Tablespace |                Description
-----------+----------+----------+------------+------------+-----------------------+---------+------------+---------------------------
 postgres  | postgres | UTF8     | en_US.utf8 | en_US.utf8 |                       | 7877 kB | pg_default | default administrative conn
 template0 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +| 7729 kB | pg_default | unmodifiable empty database
           |          |          |            |            | postgres=CTc/postgres |         |            |
 template1 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +| 7729 kB | pg_default | default template for new da
           |          |          |            |            | postgres=CTc/postgres |         |            |
(3 rows)

{"bridge":{"IPAMConfig":null,"Links":null,"Aliases":null,"NetworkID":"e687284abe301936a529d5be904e98128e93b1461cd8b2e6c956e6cba40d25f6'
432","Gateway":"172.17.0.1","IPAddress":"172.17.0.4","IPPrefixLen":16,"IPv6Gateway":"","GlobalIPv6Address":"","GlobalIPv6PrefixLen":0,'
```

Postgres in Host

The « post01-postgres » directory is created under my user name home directory and not inside a docker container.



Container Management

Tryton Containers : Start - Stop - Status

./"Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.stop"

```
Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope CurrentUser
#
Write-Host "------------------"
Write-Host "1. Stop containers"
Write-Host "------------------"
docker stop tryt11-postgres tryt11 tryt11-cron
docker ps -a
#
Write-Host "-------"
Write-Host "2. Done"
Write-Host "-------"
Pause
```

./"Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.start"

```
# tryt11
Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope CurrentUser
#
Write-Host "------------------"
Write-Host "1. Stop containers"
Write-Host "------------------"
docker stop tryt11-postgres tryt11 tryt11-cron
#
Write-Host "-------------------"
Write-Host "2. Start containers"
Write-Host "-------------------"
docker start tryt11-postgres tryt11 tryt11-cron
#
Write-Host "----------------"
Write-Host "3. Docker Status"
Write-Host "----------------"
docker ps -a
Start-Sleep -Seconds 20 # Replace by detecting database is 'up'
docker exec -tiu postgres tryt11-postgres psql -c '\l+'
```

./"Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.status"

```
Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope CurrentUser

# Step 1 : state
Write-Host "---------"
Write-Host "1. Status"
Write-Host "---------"
docker ps -a
docker volume ls
docker network ls
docker inspect tryt11-postgres -f "{{.Name}} - {{json .NetworkSettings.Networks}}"
Start-Sleep -Seconds 20 # Replace by detecting database is 'up'
docker exec -tiu postgres tryt11-postgres psql -c '\l+'

# Step 2 : done
Write-Host "-------"
Write-Host "2. Done"
Write-Host "-------"
```

Container Uninstallation

Motivation

- This section explains how to delete installed containers should they be reinstalled
- If the installation proceeds according to plan, the section can be skipped for later

Tryton

./"Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.delete"

```
Write-Host "---------"
Write-Host "1. Status"
Write-Host "---------"
docker ps -a
docker network ls
docker volume ls
dir
#
Write-Host "--------------------"
Write-Host "2. Delete all tryt11"
Write-Host "--------------------"
docker stop tryt11-postgres tryt11 tryt11-cron
docker rm tryt11-postgres tryt11 tryt11-cron
docker network rm tryt11-network
docker volume rm tryt11-database tryt11-datafile
Remove-Item -Recurse -Force tryt11-database
Remove-Item -Recurse -Force tryt11-datafile
#
Write-Host "---------"
Write-Host "3. Status"
Write-Host "---------"
docker ps -a
docker network ls
docker volume ls
dir
#
Write-Host "-------"
Write-Host "4. Done"
Write-Host "-------"
```

Postgres

./"Tryton 6.0 - Doc 00.01 - Installation & administration.docker.post01.delete"

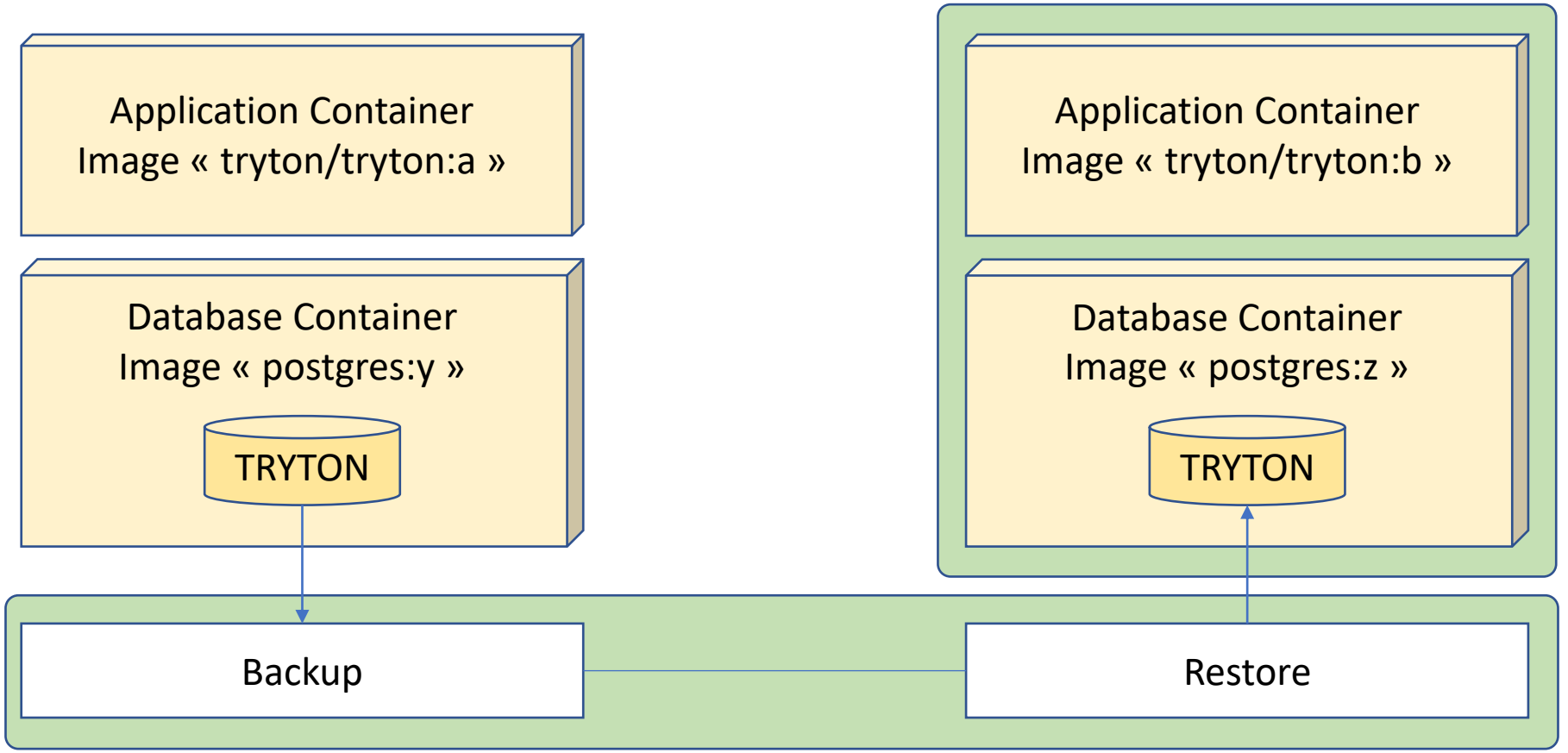
```
Write-Host "---------"
Write-Host "1. Status"
Write-Host "---------"
docker ps -a
docker network ls
docker volume ls
dir
#
Write-Host "-------------------------------------------"
Write-Host "2. Delete all post01-postgres & post01-pgadmin"
Write-Host "-------------------------------------------"
docker stop post01-postgres post01-pgadmin
docker rm post01-postgres post01-pgadmin
Remove-Item -Recurse -Force ${HOME}/post01-postgres # HOME == USER
#
Write-Host "---------"
Write-Host "3. Status"
Write-Host "---------"
docker ps -a
docker network ls
docker volume ls
dir
#
Write-Host "-------"
Write-Host "4. Done"
Write-Host "-------"
```

Container multi-versioning

Motivation

- Installing containers originating from different TRYTON image versions allow integration for bug correction
- Generally, the data base format does not change between major TRYTON versions.

- From the new « TRYTON;POSTGRES » image(s) create a « b;z » version of the existing « a;y » environment
- Backup and restore the database : see sections hereafter
- Because each environnement exposes its own TCP IP ports, they can also seamlessly coexist

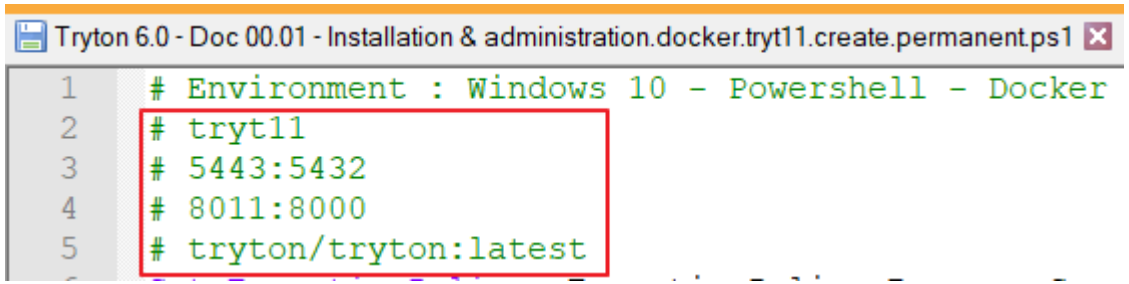


Installing containers built from « tryton/tryton » images with different « tags »

- Docker shines thanks to its capability to isolate containers from one another, especially when such containers are generated from an « image » == « application » having different « tags » == « versions ». F.i. :
- Image for version « 5.0 » or « 6.0 » : « tryton/tryton:5.0 » or « tryton/tryton:6:0 »
- Image for version « latest » : « tryton/tryton:latest » or simply « tryton/tryton »

- In order to generate containers according to the « image+tag » they are derived from, adapt the parameters identified in the choosen « create » utility script. Just perform a « replace all » of :
  - « tryt11 » : Container, database « names »
  - « 5443 » : the database port on the host system side & « 8011 » : the tryton port on the host system side
  - For simplicity, if the name is « 11 », we added « 11 » to default ports « 5432 » and « 8000 »
  - « tryton/tryton:latest » : change if necessary

- Finally adapt some environment variables and volume locations according to taste



System 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.tryton.6.0> docker ps -a
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS                          PORTS     NAMES
bbcf2e9704bc   dpage/pgadmin4     "/entrypoint.sh"         10 minutes ago   Exited (0) About a minute ago             post01-pgadmin
23da037c8b30   postgres           "docker-entrypoint.s…"   10 minutes ago   Exited (0) About a minute ago             post01-postgres
8a8df058d5d0   tryton/tryton:6.0  "/entrypoint.sh tryt…"   3 hours ago      Exited (137) 58 seconds ago               tryt11-cron
65b4a6fc580a   tryton/tryton:6.0  "/entrypoint.sh uwsg…"   3 hours ago      Exited (30) About a minute ago            tryt11
23d18e90f17e   postgres           "docker-entrypoint.s…"   3 hours ago      Exited (0) About a minute ago             tryt11-postgres
```

Start the containers
docker start tryt11-postgres tryt11 tryt11-cron
docker start post01-postgres post01-pgadmin

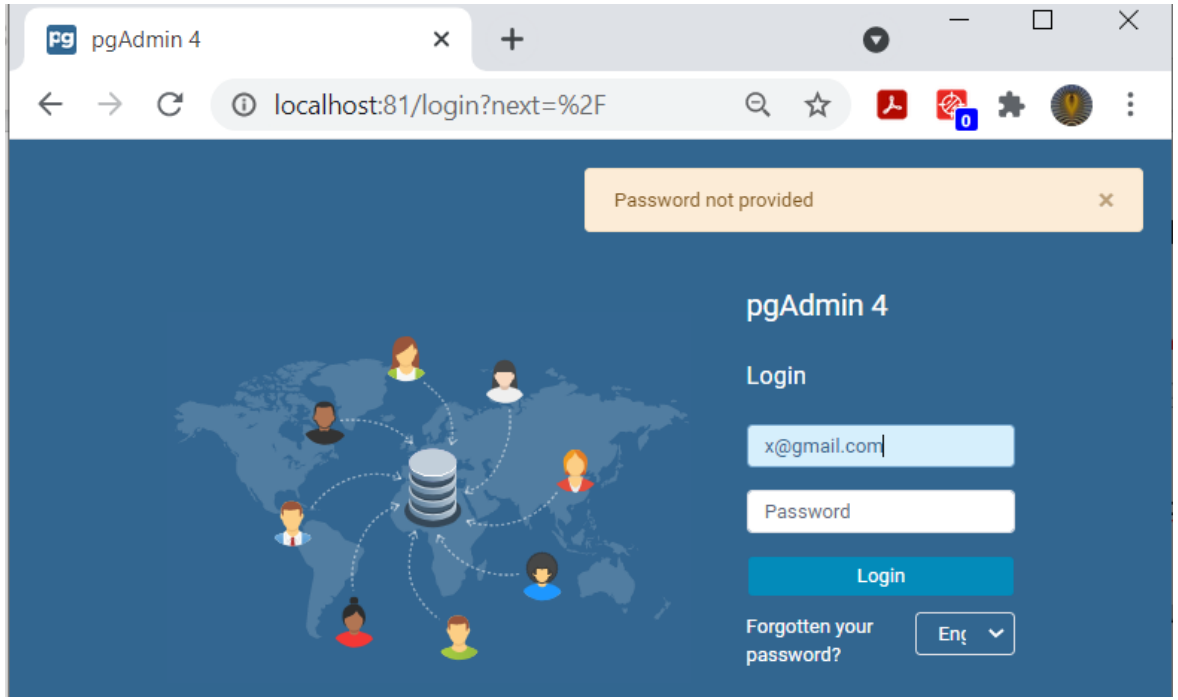
Above commands are unnecessary when the
PC is set to “Sleep”

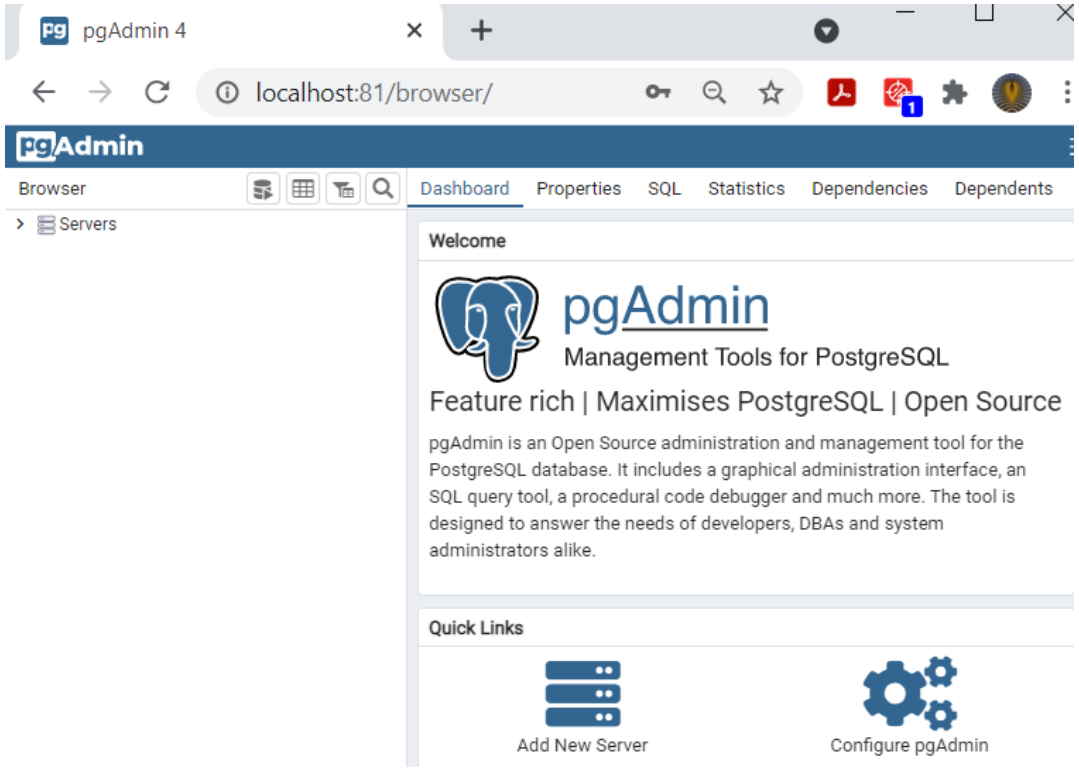
```
PS C:\Users\mrmar\docker.tryton.6.0> docker ps -a
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS                          NAMES
bbcf2e9704bc   dpage/pgadmin4     "/entrypoint.sh"         5 minutes ago    Up 5 minutes    443/tcp, 0.0.0.0:81->80/tcp    post01-pgadmin
23da037c8b30   postgres           "docker-entrypoint.s…"   5 minutes ago    Up 5 minutes    0.0.0.0:5433->5432/tcp         post01-postgres
8a8df058d5d0   tryton/tryton:6.0  "/entrypoint.sh tryt…"   3 hours ago      Up 3 hours      8000/tcp                       tryt11-cron
65b4a6fc580a   tryton/tryton:6.0  "/entrypoint.sh uwsg…"   3 hours ago      Up 3 hours      0.0.0.0:8011->8000/tcp         tryt11
23d18e90f17e   postgres           "docker-entrypoint.s…"   3 hours ago      Up 3 hours      0.0.0.0:5443->5432/tcp         tryt11-postgres
```

User Interface

PgAdmin4

pgAdmin4

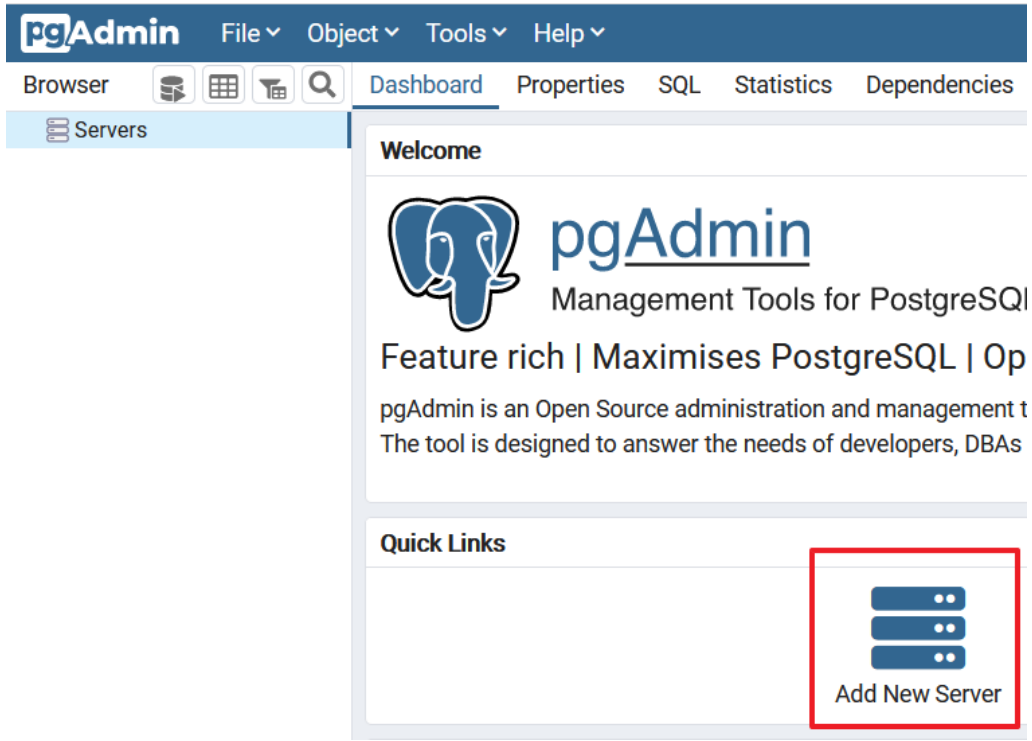


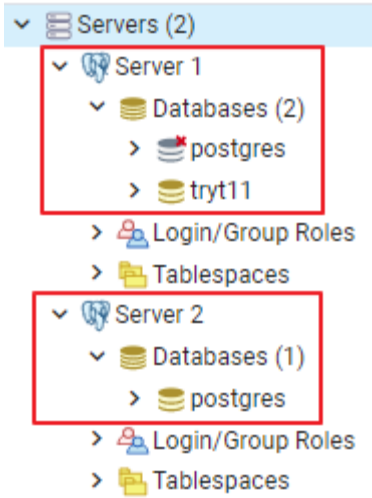


Note :

- User : « x@gmail.com » is the address used in the script
- Password : « Password » is the password defined in the script

Create servers to connect to the databases

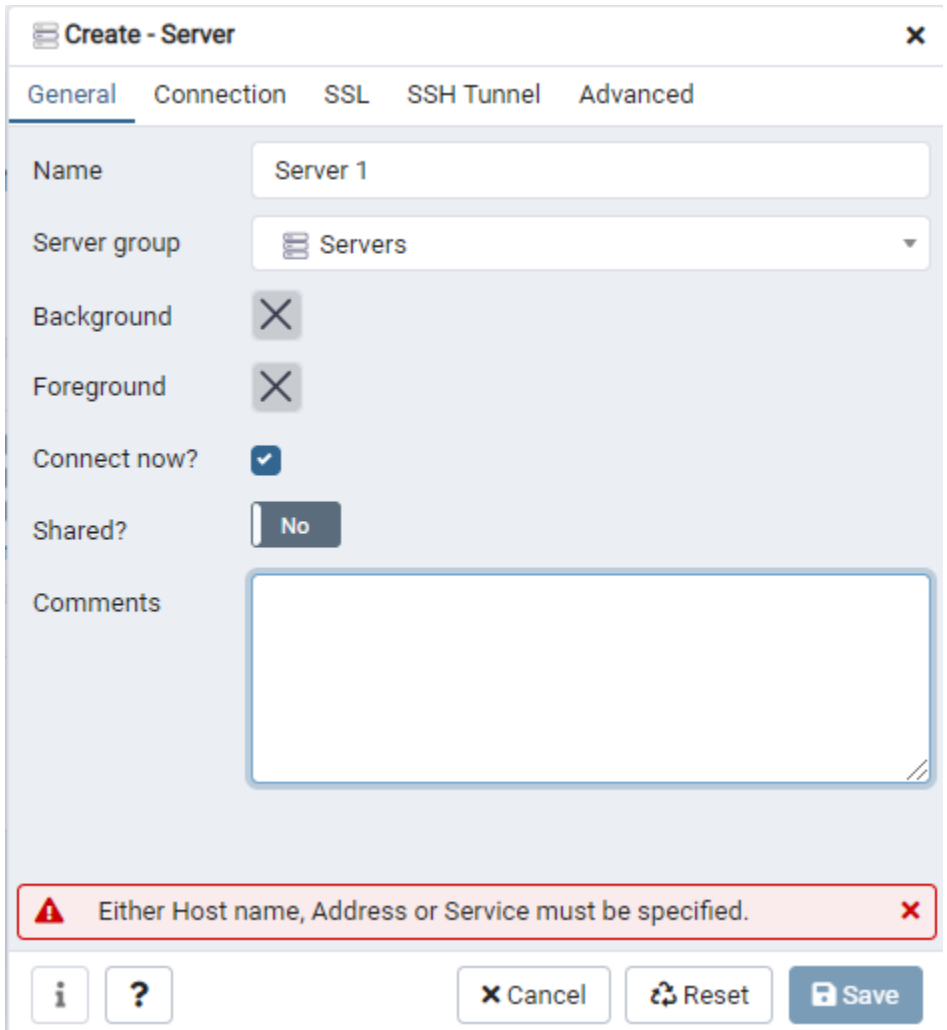


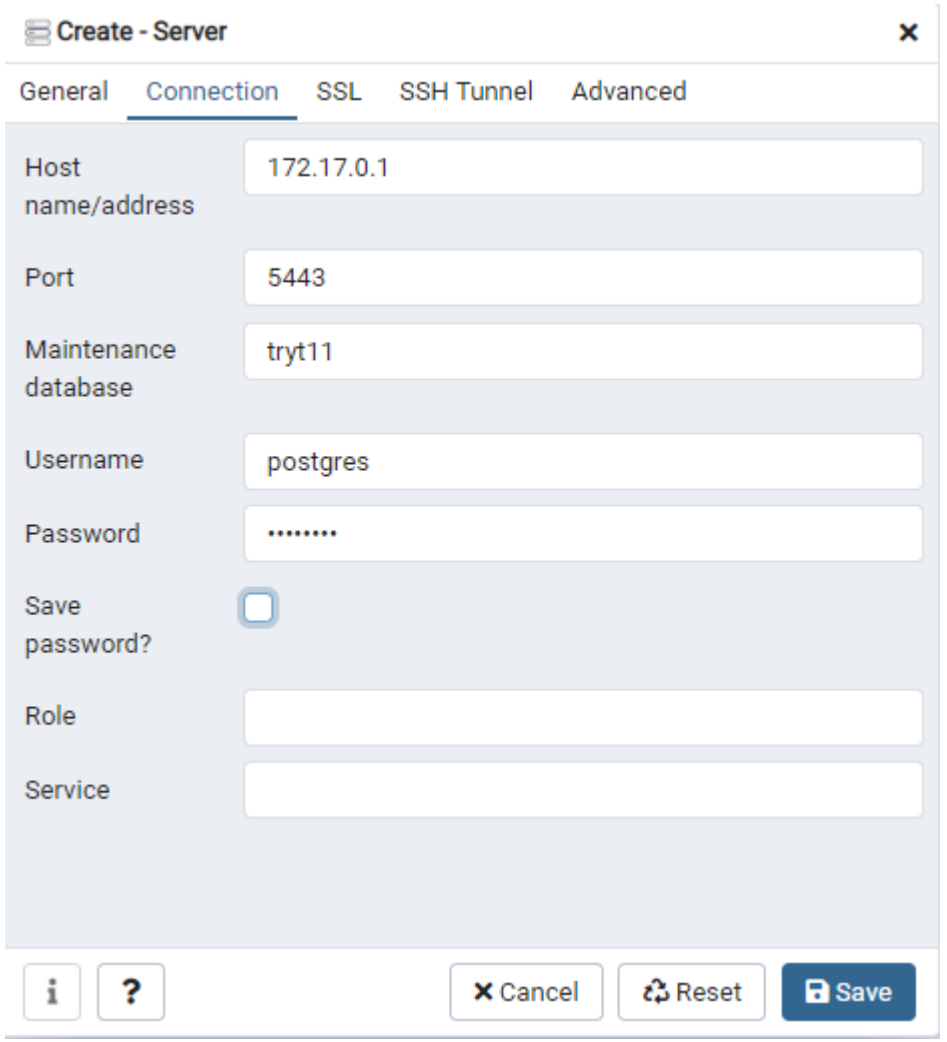


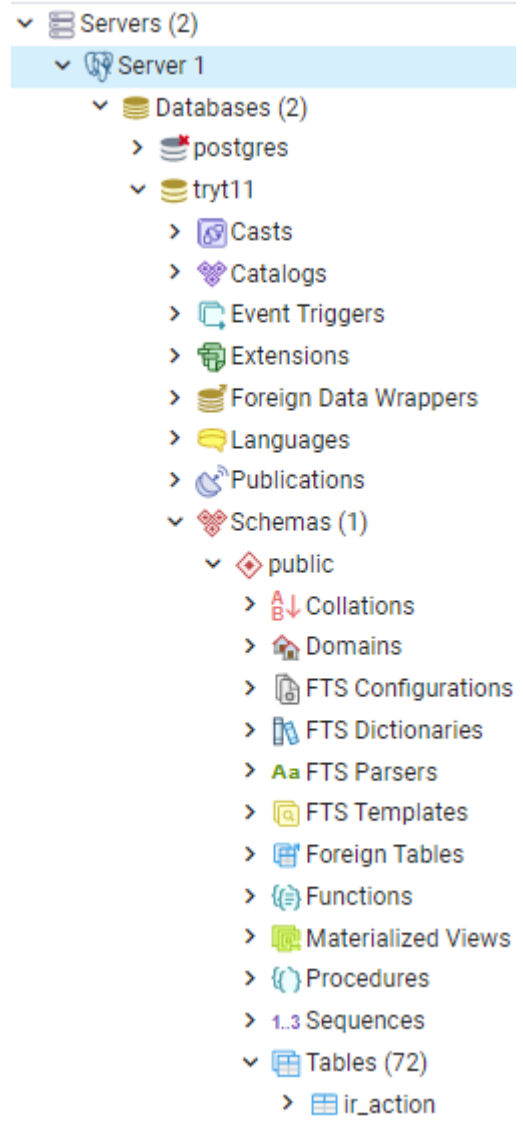
We define hereafter the following server(s) :

- Server 1 : for access to the « Tryton » Database Container
- Server 2 : if we created the optional « Postgres » Database Container

Server 1 - Connect to « tryt11 » database in « tryt11-postgres » container

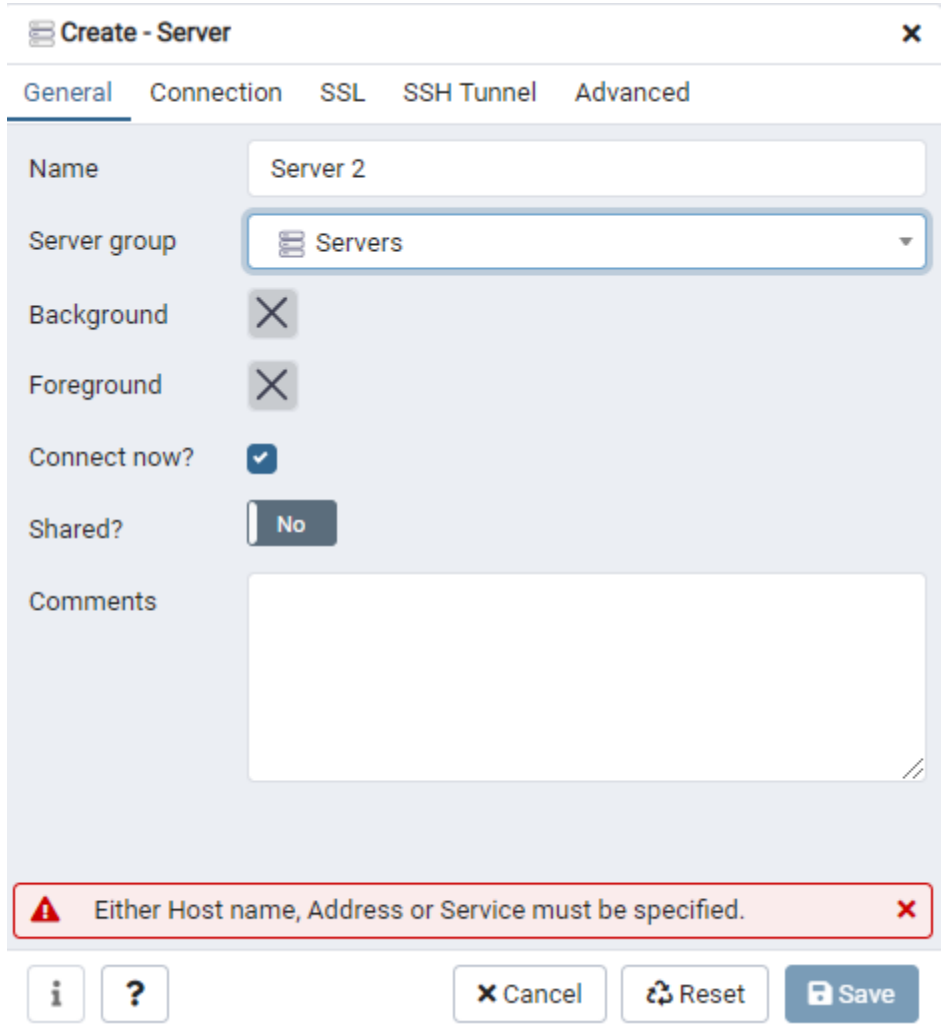


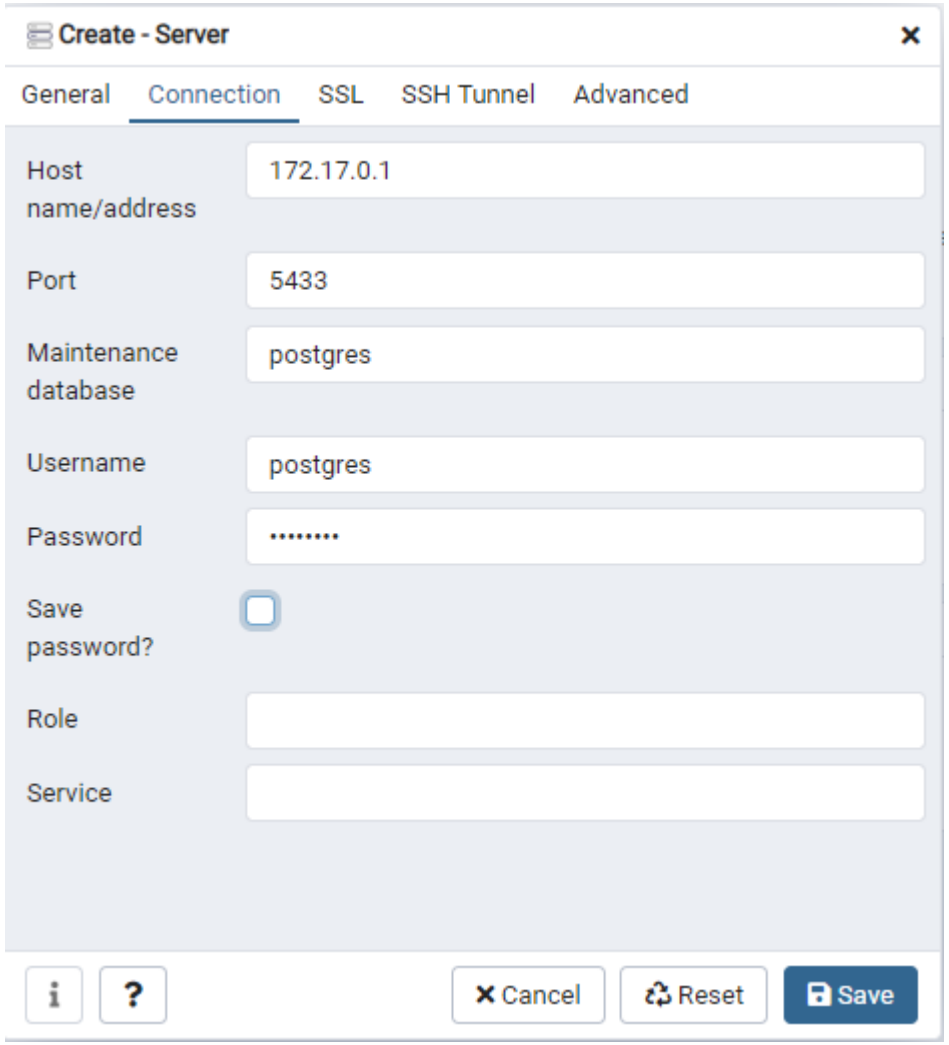


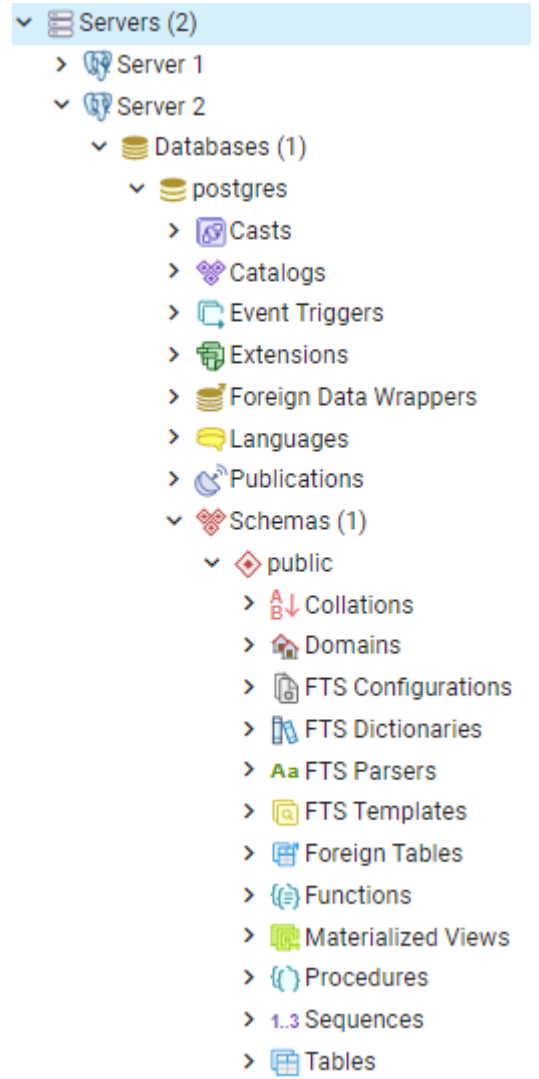



```
PS C:\Users\mrmar\docker.tryton.6.0> docker inspect tryt11-postgres -f "{{json .NetworkSettings.Networks }}"
{"bridge":{"IPAMConfig":null,"Links":null,"Aliases":null,"NetworkID":"e687284abe301936a529d5be904e98128e93b1
9bd","Gateway":"172.17.0.1","IPAddress":"172.17.0.2","IPPrefixLen":16,"IPv6Gateway":"","GlobalIPv6Address":"
```

Server 2 - Connect to « postgres » database in « post01-postgres » container







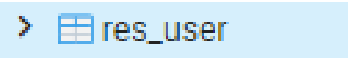

```
PS C:\Users\mrmar\docker.tryton.6.0> docker inspect d01-postgres -f "{{json .NetworkSettings.Networks }}"
{"bridge":{"IPAMConfig":null,"Links":null,"Aliases":null,"NetworkID":"e687284abe301936a529d5be904e98128e93b1
820","Gateway":"172.17.0.1","IPAddress":"172.17.0.4","IPPrefixLen":16,"IPv6Gateway":"","GlobalIPv6Address":"
```

« Tryton » - Initial Database State

Only « ir » & « res » tables are installed

- Tables (72)
  - 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_avatar
  - ir_avatar_cache
  - 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_device
  - res_user_login_attempt
  - res_user_warning

« Tryton » - Initial Database State



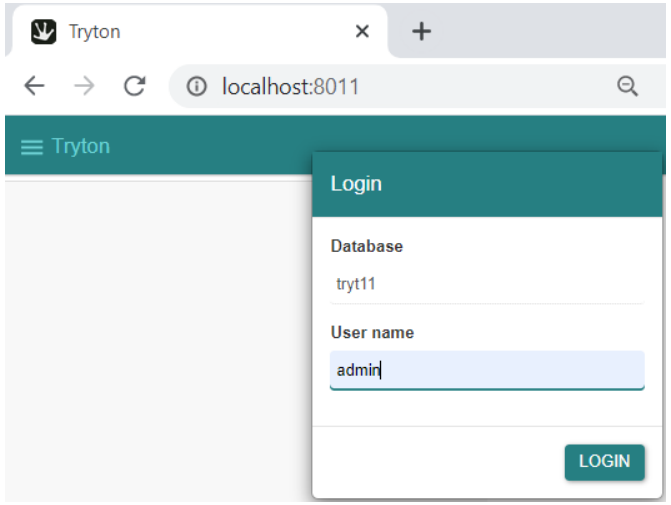
Data Output | Explain | Messages | Notifications

	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...

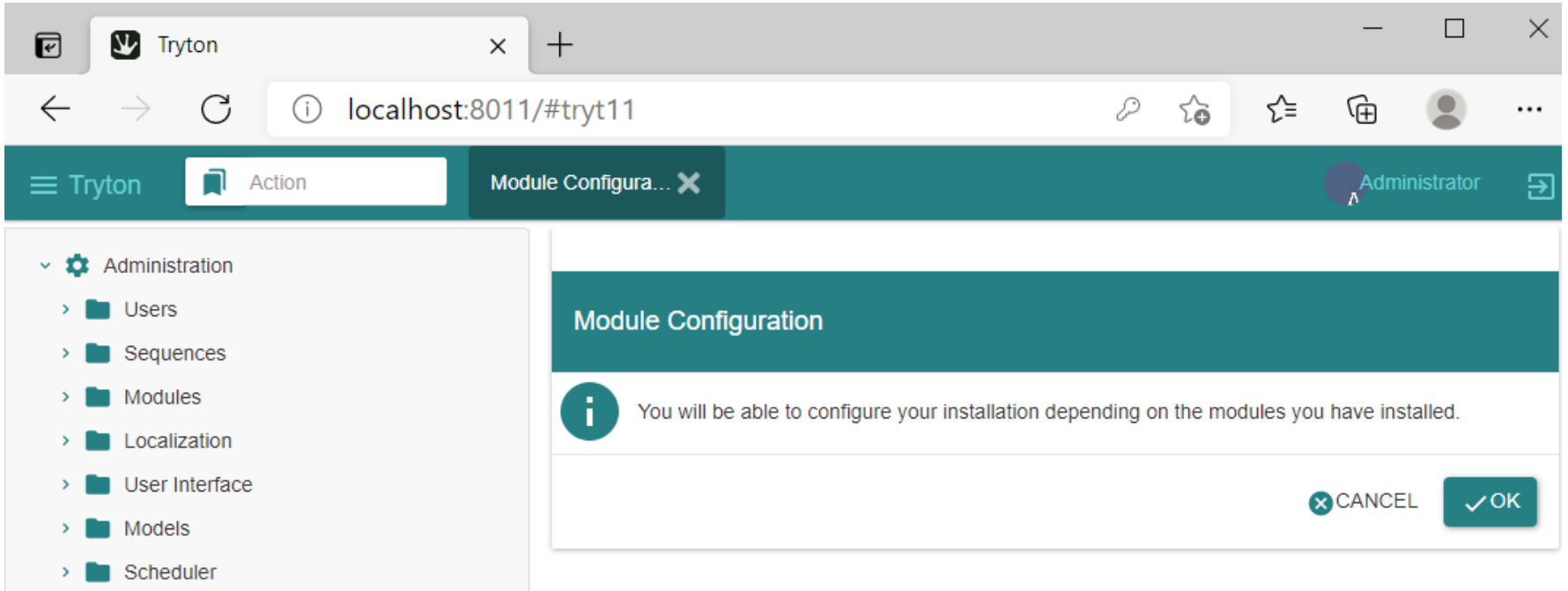
Tryton

Login / Logout

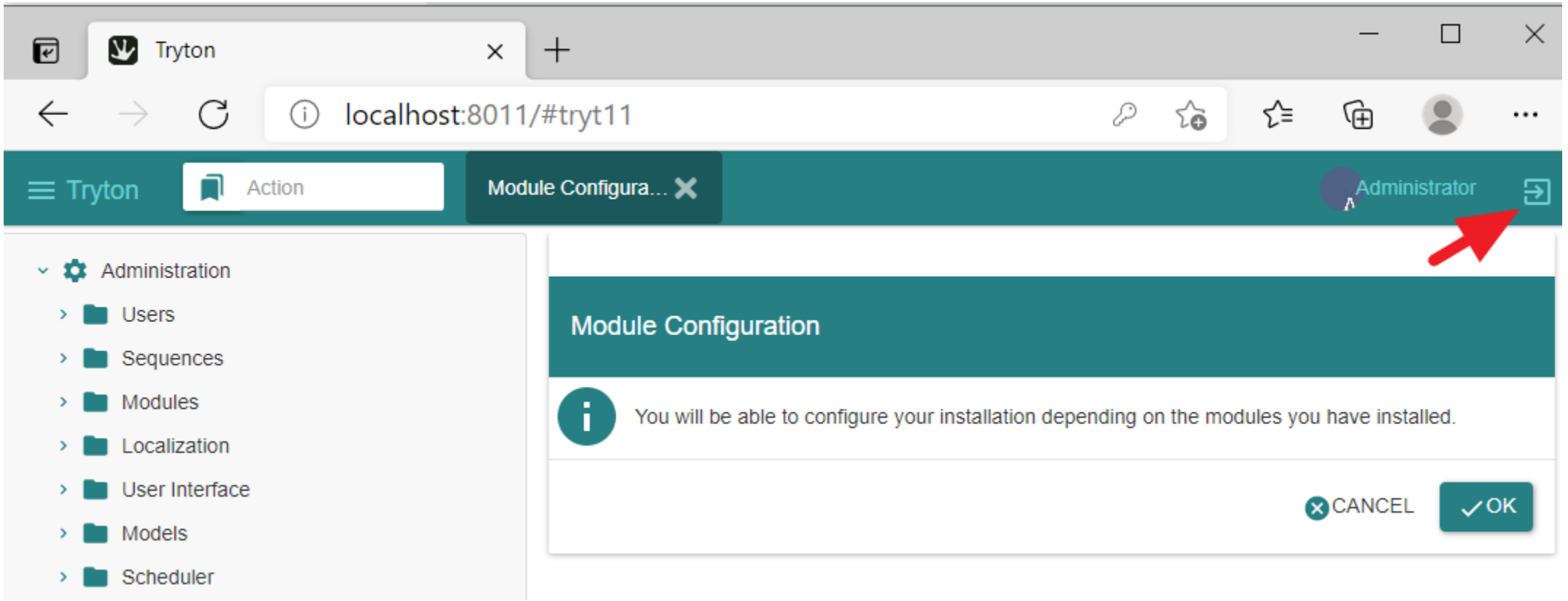
Tryton Login



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



Tryton Logout



It is compulsory to « logout » prior to performing database backup / restore operations (cache usage)

Database Operations

Tryton

./"Tryton 6.0 - Doc 00.01 - Installation & administration.database.tryt11.query"

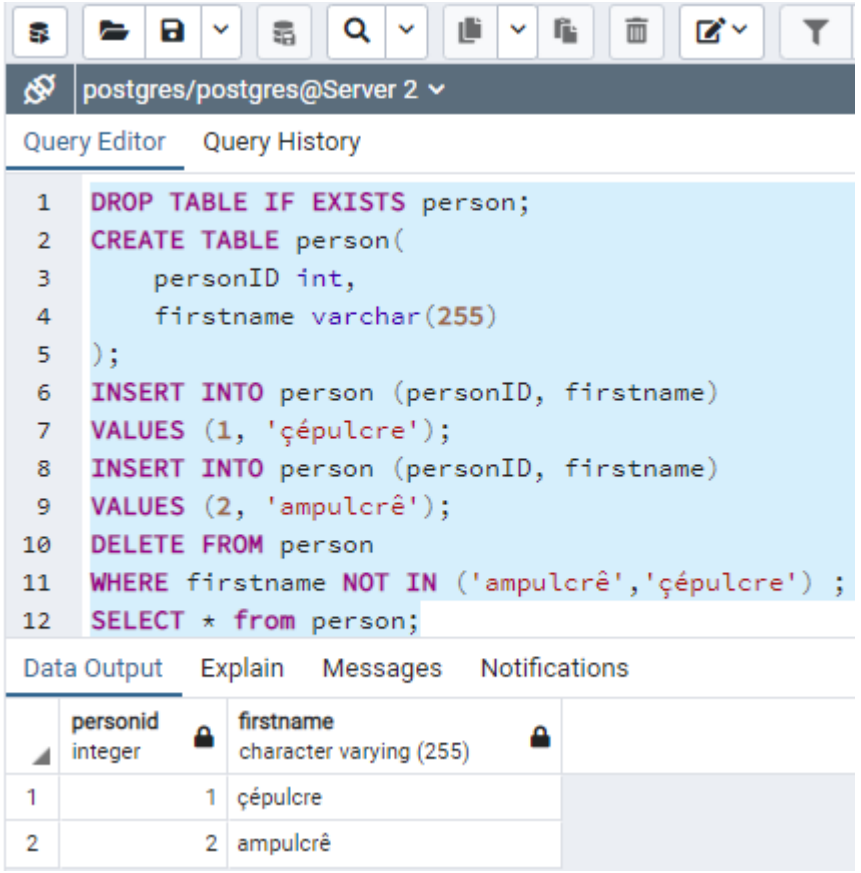
```
# container : tryt11-postgres
# database  : tryt11

Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope CurrentUser
#
Write-Host "---------"
Write-Host "1. Select"
Write-Host "---------"
docker cp query.dbms_01.sql tryt11-postgres:/inpu.sql
#
Write-Host "---------"
Write-Host "2. Access"
Write-Host "---------"
docker exec -it tryt11-postgres psql -d tryt11 -U postgres -P pager=off -f inpu.sql -o outp.txt
docker cp tryt11-postgres:/outp.txt query.dbms_02.txt
```

*.sql	
query.dbms_01.sql	list of tables with row count
query.dbms_02.sql	list of pk-fk table relationships
query.res_user.sql	list of 'res_user' table rows (example)

Postgres

Populate a sample UTF8 table to verify backup/restore correctness

```
DROP TABLE IF EXISTS person;
CREATE TABLE person(
   personID int,
   firstname varchar(255)
);
INSERT INTO person (personID, firstname)
VALUES (1, 'çépulcre');
INSERT INTO person (personID, firstname)
VALUES (2, 'ampulcrê');
DELETE FROM person
WHERE firstname NOT IN ('ampulcrê','çépulcre') ;
SELECT * from person;
```



Scripts

Scripts

TRYTON **Database Powershell Commands - Run as ./"..."**	
Tryton 6.0 - Doc 00.01 - Installation & administration.database.tryt11.backup	Backup
Tryton 6.0 - Doc 00.01 - Installation & administration.database.tryt11.restore	Restore
Tryton 6.0 - Doc 00.01 - Installation & administration.database.tryt11.query	Query tables

POSTGRES **Database Powershell Commands - Run as ./"..."**	
Tryton 6.0 - Doc 00.01 - Installation & administration.database.post01.backup	Backup (3 modes)
Tryton 6.0 - Doc 00.01 - Installation & administration.database.post01.restore.*	Restore
Tryton 6.0 - Doc 00.01 - Installation & administration.database.post01.query	Query tables

Database Backup

Motivation

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

Documentation

List of options	http://manpages.ubuntu.com/manpages/trusty/man1/pg_dump.1.html
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
Help	pg_dump --help

Tryton

./"Tryton 6.0 - Doc 00.01 - Installation & administration.database.tryt11.backup"

```
Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope CurrentUser
#
Write-Host "---------"
Write-Host "1. Status"
Write-Host "---------"
docker exec -tiu postgres tryt11-postgres psql -c '\l+'

# Step 2 : dump tryt11
Write-Host "-------"
Write-Host "2. Dump"
Write-Host "-------"
docker exec tryt11-postgres pg_dump -Ft -U postgres -O -f tryt11-db-backup.tar tryt11

# Step 3 : export outside container (optional ; specifically use if later import in another container)
Write-Host "--------------------------------"
Write-Host "3. Export 'tar' outside container"
Write-Host "--------------------------------"
docker cp tryt11-postgres:/tryt11-db-backup.tar tryt11-db-backup.tar

#
Write-Host "-------"
Write-Host "4. Done"
Write-Host "-------"
Pause
```

Postgres

Postgres - Backup - Redirection - Incorrect result

Note :

- The “tryt11-postgres” container contains two databases : “postgres” and “tryt01”
- The “post01-postgres” container (if installed) contains one database : “postgres”

docker exec **post01-postgres** pg_dump -C -c -U postgres -O **postgres** > post01-db-backup.sql
docker exec **post01-postgres** pg_dump -Fc -U postgres -O **postgres** > post01-db-backup.bak
docker exec **post01-postgres** pg_dump -Ft -U postgres -O **postgres** > post01-db-backup.tar

- Above file content redirections generate incorrect results with respect to UTF-8 characters
- File assignment must be used (see hereafter)

./"Tryton 6.0 - Doc 00.01 - Installation & administration.database.post01.backup"

```
#
Write-Host "-------"
Write-Host "2. Dump"
Write-Host "-------"
docker exec post01-postgres pg_dump -C -c -U postgres -O -f post01-db-backup.createYes.sql postgres # includes database create commands
docker exec post01-postgres pg_dump -c    -U postgres -O -f post01-db-backup.createNot.sql postgres # coes not include such commands
docker exec post01-postgres pg_dump -Fc   -U postgres -O -f post01-db-backup.bak postgres
docker exec post01-postgres pg_dump -Ft   -U postgres -O -f post01-db-backup.tar postgres
docker exec post01-postgres ls -l

#
Write-Host "--------------------------------"
Write-Host "3. Export 'tar' outside container"
Write-Host "--------------------------------"
docker cp post01-postgres:/post01-db-backup.createYes.sql post01-db-backup.createYes.sql
docker cp post01-postgres:/post01-db-backup.createNot.sql post01-db-backup.createNot.sql
docker cp post01-postgres:/post01-db-backup.bak post01-db-backup.bak
docker cp post01-postgres:/post01-db-backup.tar post01-db-backup.tar
```

Database Restore

Tryton

./"Tryton 6.0 - Doc 00.01 - Installation & administration.database.tryt11.restore"

Log out the system if you happen to be signed in

```
# Step 1 : docker stop/start containers
Write-Host "-------------------------------"
Write-Host "1. Docker stop/start containers"
Write-Host "-------------------------------"
docker stop tryt11-postgres tryt11
docker start tryt11-postgres tryt11

# Step 3 : drop and create tryt11-copy
Write-Host "-------------------------------"
Write-Host "3. Drop and create tryt11-copy"
Write-Host "-------------------------------"
docker exec tryt11-postgres dropdb -f -U postgres tryt11-copy
docker exec tryt11-postgres createdb -U postgres -T template0 tryt11-copy

# Step 4.1 : import inside container (optional ; function of step 1.2 above)
Write-Host "----------------------------"
Write-Host "4.1. Import inside container"
Write-Host "----------------------------"
docker cp tryt11-db-backup.tar tryt11-postgres:/tryt11-db-backup.tar

# Step 4.2 : restore tryt11-copy from tryt11
Write-Host "-----------------------------------"
Write-Host "4.2. Restore tryt11-copy from tryt11"
Write-Host "-----------------------------------"
docker exec -i tryt11-postgres pg_restore -Ft -U postgres -d tryt11-copy -v ./tryt11-db-backup.tar
```

Postgres

./"Tryton 6.0 - Doc 00.01 - Installation & administration.database.post01.restore.binary"

```
# Step 1 : docker stop/start containers
Write-Host "------------------------------"
Write-Host "1. Docker stop/start containers"
Write-Host "------------------------------"
docker stop post01-postgres post01
docker start post01-postgres post01

# Step 3 : drop and create post01-copy
Write-Host "------------------------------"
Write-Host "3. Drop and create post01-copy"
Write-Host "------------------------------"
docker exec post01-postgres dropdb -f -U postgres post01-copy
docker exec post01-postgres createdb -U postgres -T template0 post01-copy

# Step 4.1 : import inside container (optional ; function of step 1.2 above)
Write-Host "---------------------------"
Write-Host "4.1. Import inside container"
Write-Host "---------------------------"
docker cp post01-db-backup.tar post01-postgres:/post01-db-backup.tar

# Step 4.2 : restore post01-copy from post01
Write-Host "-----------------------------------"
Write-Host "4.2. Restore post01-copy from post01"
Write-Host "-----------------------------------"
docker exec -i post01-postgres pg_restore -Ft -U postgres -d post01-copy -v ./post01-db-backup.tar
```

./"Tryton 6.0 - Doc 00.01 - Installation & administration.database.post01.restore.character.createNot"

The « sql » file does not contain database drop & create statements

```
# Step 1 : docker stop/start containers
Write-Host "------------------------------"
Write-Host "1. Docker stop/start containers"
Write-Host "------------------------------"
docker stop post01-postgres post01-pgadmin
docker start post01-postgres post01-pgadmin

# Step 3 : drop and create post01-copy
Write-Host "------------------------------"
Write-Host "3. Drop and create post01-copy"
Write-Host "------------------------------"
docker exec post01-postgres dropdb -f -U postgres post01-copy
docker exec post01-postgres createdb -U postgres -T template0 post01-copy

# Step 4.1 : import inside container (optional ; function of step 1.2 above)
Write-Host "---------------------------"
Write-Host "4.1. Import inside container"
Write-Host "---------------------------"
docker cp post01-db-backup.createNot.sql post01-postgres:/post01-db-backup.createNot.sql

# Step 4.2 : restore post01-copy from post01
Write-Host "-----------------------------------"
Write-Host "4.2. Restore post01-copy from post01"
Write-Host "-----------------------------------"
docker exec -i post01-postgres psql -U postgres -d post01-copy -f post01-db-backup.createNot.sql
```

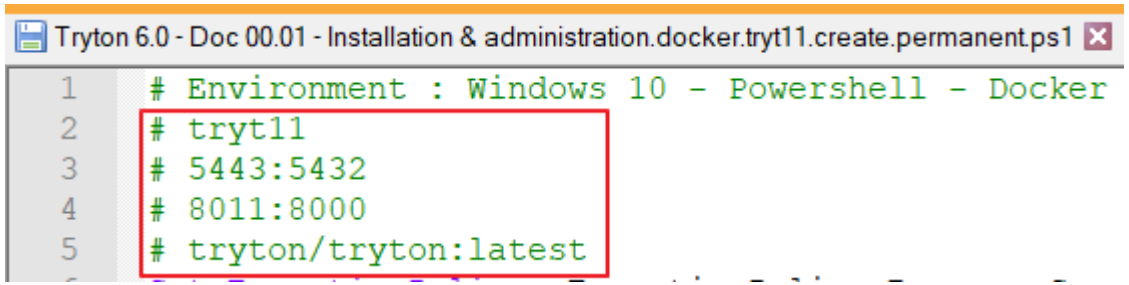
./"Tryton 6.0 - Doc 00.01 - Installation & administration.database.post01.restore.character.createYes"

The « sql » file contains database drop & create statements

```
# Step 1 : docker stop/start containers
Write-Host "------------------------------"
Write-Host "1. Docker stop/start containers"
Write-Host "------------------------------"
docker stop post01-postgres post01-pgadmin
docker start post01-postgres post01-pgadmin

# Step 2 : state
Write-Host "---------"
Write-Host "2. Status"
Write-Host "---------"
docker ps -a
Start-Sleep -Seconds 20 # Replace by detecting database is 'up'
docker exec -tiu postgres post01-postgres psql -c '\l+'

# Step 4.1 : import inside container (optional ; function of step 1.2 above)
Write-Host "---------------------------"
Write-Host "4.1. Import inside container"
Write-Host "---------------------------"
docker cp post01-db-backup.createYes.sql post01-postgres:/post01-db-backup.createYes.sql

# Step 4.2 : restore post01-copy from post01
Write-Host "-------------------------------------------------------------------------------------------------"
Write-Host "4.2. Restore post01-copy from post01 [!!! DROP & CREATE inside 'post01-db-backup.createYes.sql']"
Write-Host "-------------------------------------------------------------------------------------------------"
docker exec -i post01-postgres psql -U postgres -f post01-db-backup.createYes.sql
```

Multi-database Container

« Tryton Database Server » Container



- « Tryton 6.0 - Doc 00.01 - Installation & administration.docker.tryt11.create.volatile or permanent »

- The database container hosts a database server accessible with default port 5432:5432 (adaptable).
- Initially, the database server is setup in the script to manage one database.

- When performing database backup / restore operations it is prudent to :
  - Log out of TRYTON client
  - Double-check in the script whether the database and files need to be deleted or preserved

Backup a Tryton Database & Restore it to another name

Tryton 6.0 - Doc 00.01 - Installation & administration.database.tryt11.backup.ps1

```
#
# tryt11
# tryt11-postgres
# tryt11-db-backup.tar
#
```

- ./"Tryton 6.0 - Doc 00.01 - Installation & administration.database.tryt11.backup“

- Use this script to backup the database initially created in the « Tryton Database Server »
- In doing so, you have a « fresh » database that can be restored to another name inside the same container

Tryton 6.0 - Doc 00.01 - Installation & administration.database.tryt11.restore.ps1

```
# Change replace : tryt11
# Database is    : tryt11-copy
```

- ./"Tryton 6.0 - Doc 00.01 - Installation & administration.database.tryt11.restore“

- Use this script to restore a database backup in the « Tryton Database Server » container under another name
- From this moment onwards, when login into TRYTON, a choice between two databases will be proposed

Next

Using Tryton

This follow-up documents explain how to use Tryton

Topics
Tryton 6.0 - Doc 00.01 - Installation & administration
Tryton 6.0 - Doc 05.01 - Basic functionality
Tryton 6.0 - Doc 10.01 - Purchase
Tryton 6.0 - Doc 15.01 - Sale
Tryton 6.0 - Doc 80.01 - Ancillaries

Issues

Known Issues

These are unresolved topics that relate to the presentation, not to the functioning of the system.

Document	Subject
Tryton 6.0 - Doc 00.01 - Installation	Waiting on Postgres DB to be ready

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/]