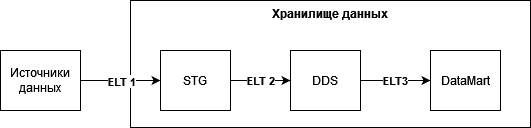
# Тестовое задание

Необходимо реализовать загрузку данных в корпоративное хранилище в слой DDS, для моделирования слоя DDS использовать методологию Data Vault 2.0. Скрипт загрузки разработать на python. Данные загружать в БД PostgreSQL. (Если нет своей тестовой БД, инструкция по получению БД в Приложении 1).

Слои хранилища представлены на схеме:



Задача:

1. Спроектировать два слоя: STG и DDS;
2. Разработать потоки перекладки и трансформации данных: ELT 1 и ELT 2.

# Тестовый источник

Как источник данных необходимо использовать REST-сервис <https://jsonplaceholder.typicode.com/posts/>. В качестве бизнес-ключей использовать поля Id и userId.

# Результат выполнения задания:

1. DDL скрипты по созданию таблиц в слоях STG и DDS;
2. python скрипты для потоков ELT 1 и ELT 2.

Приложение 1

# Варианты работы с БД PostgreSQL

## **Онлайн БД PostgreSQL**

Можно воспользоваться сервисом ElephantSQL. Для работы с данными до 20МБ бесплатный инстанс БД.

<https://www.elephantsql.com/plans.html>

## **Локальная БД PostgreSQL**

Инструкции по разворачиванию БД локально.

* **Docker Desktop (Windows and OSX):** [www.docker.com/products/ docker-desktop](http://www.docker.com/products/docker-desktop)
* **Docker CE (Linux):** <https://docs.docker.com/get-docker/>

**Install Docker [Optional]**

If you’re intending to install PostgreSQL directly on your development machine, or you already have a version running somewhere that you can use, then you can skip this section if you like. However, if like me you don’t like “faffing” around installing large apps on your local machine, then Docker is a great option for you (although paradoxically, Docker is quite a large application as of itself!)

**What Is Docker?**

Docker is a containerization platform that enables you to

* Package *your* apps as images and allow others to download and run them as containers (on Docker).
* Obtain other developer or software vendor “images” (from a repository), and run them as containers on your machine (so long as you’ve installed Docker).

The core concept of a Docker image is that they are self-contained, meaning that the image has everything it needs for it to run, avoiding complex installations, locating and installing third-party support libraries, etc. It ultimately avoids the “it works on my machine” argument.

There is a little bit of a learning curve to it (not much though), and once you master the basics, it can save you so much time and effort, that as a developer, I can’t recommend it highly enough.

**Docker Desktop vs. Docker CE**

Confusingly (for me at least), if you’re running Windows or OSX, you need to install something called *Docker Desktop*. If, however, you’re a Linux person, then you should install *Docker Community Edition* or CE. There are probably torturously pedantic reasons for this, which I’m not aware of, nor would I be interested in learning about, so all you really need to know is where to get them!

* **Docker Desktop Here:** [www.docker.com/products/docker-desktop](http://www.docker.com/products/docker-desktop)
* **Docker CE Here:** <https://docs.docker.com/get-docker/>

Before you can download and install Docker Desktop, you need to sign up for a Docker Hub account; this is a free sign-up so nothing really to worry about. It also comes in useful if you want to upload your own images to the Docker Hub for distribution.

**Warning!** at the time of writing, Docker Desktop can only be installed “directly” on Windows 10 professional. however, if you’re running Windows 10 home, you can work around this by using something called Windows Subsystem for linux (WSl).



as I’ve said before, I’m not going to go into detail on how to do this as the Docker guys have provided great instructions for this here:

[https://docs.docker.com/docker-for-windows/install-windows- home/.](https://docs.docker.com/docker-for-windows/install-windows-home/)

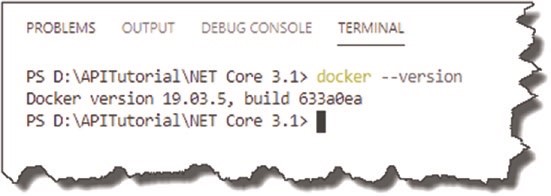
Docker Desktop installation is super simple; for Docker CE you will need to refer to the install instructions for your specific distro – again, however, it’s straightforward.

**Post-installation Check**

Irrespective of which flavor of Docker you install, post-installation, open a command line, and type

docker --version

You should get something like the following.

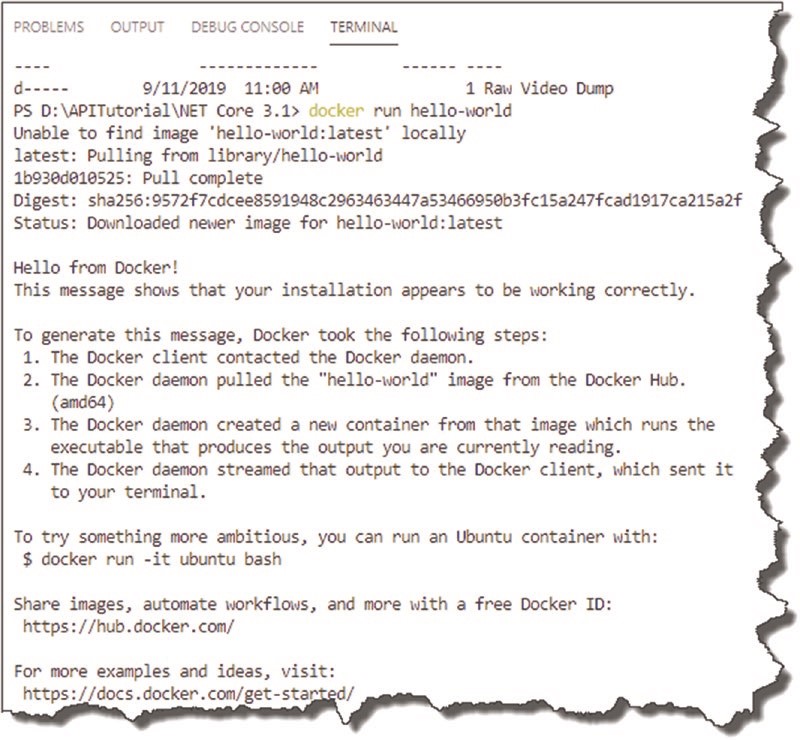


***Figure 2-8.*** *Check Docker version*

ChapTer 2 SeTTIng Up YOUr DeVelOpMenT enVIrOnMenT To further test that it is fully working, type

docker run hello-world

If this is the first time you’ve run this, Docker will go to the Docker Hub, pull down the hello-world image, and run it; you should see something like this.



***Figure 2-9.*** *Hello World Docker image download and run*

We don’t need to go into too much more detail about what’s happening here (although the output generated by hello-world does a pretty good job); suffice to say that Docker is set up and ready to go. I’ll cover more on Docker as we move through the tutorial.

**Install PostgreSQL**

If you don’t want to use Docker and want to install PostgreSQL directly on your development machine (or on another server, virtual machine, etc.), then you’ll need to follow the install steps for your OS. As mentioned previously, I won’t be detailing those steps in detail here as the PostgreSQL guys have done a great job of that already here: [www.postgresql.org/download/](http://www.postgresql.org/download/).

ChapTer 2 SeTTIng Up YOUr DeVelOpMenT enVIrOnMenT

**Warning!** I’ve spent many hours getting postgreSQl up and running on a linux box and connecting in from another machine. now this is due largely to the fact I’m not particularly great with linux, and so those of you that are adept with linux would undoubtedly have less trouble.



For me though, struggling with the nuances of installing a DB detract from the act of coding, which is what I *really* want to be doing. hence the reason why I *strongly* suggest the use of Docker.

native Windows and OSX installations of postgreSQl are (as usual) much easier.

**PostgreSQL Database**

Before moving on to writing our DB Context, I first want to make sure we have an instance of PostgreSQL up and running and configured correctly.

**Using Docker**

Now, I’m going to use Docker to run my instance of PostgreSQL on my development machine, so if you’ve chosen that approach too (or you want to see how easy it is to spin up an instance), read on. If you’ve already got a PostgreSQL instance running, you can skip to the Connecting with DBeaver section.

Ensuring you have Docker installed and running (see Chapter 2) at a command prompt; simply type

docker run --name some-postgres -e POSTGRES\_PASSWORD=mysecretpassword -p 5432:5432 -d postgres

**Note** this is all on one line.

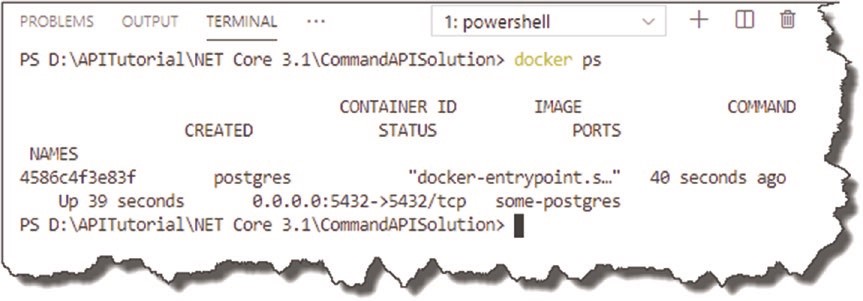
Assuming you have Docker installed and it’s running (I don’t like having Docker Desktop run automatically at startup, so I manually start it when needed), you should see the following.



***Figure 7-2.*** *PostgreSQL Image Downloaded and Running*

If this is the first time you’ve run this command, you’ll see that Docker is “Unable to find image” locally, so it pulls one down from Docker Hub. Typing

docker ps should show you the number of running containers.



***Figure 7-3.*** *The Docker PS Command*

persisting Our Data

Here, you can see that we have one, which should be our PostgreSQL instance.