# SAS Viya 3.5 (or 9.4) Platform with R, Python Client Containers and Postgres Backend

This is very easy to set up, but it is a good idea to understand what is going on. Docker Compose provides a mechanism to start and stop related containers in one cohesive set of instructions. I have not included any autoscaling as it is not needed for this example. If you want to do this, it is better to switch to a Kubernetes style platform. I use Docker Compose here because it is an easy way to make this happen.

The tradeoffs contemplated working on this platform is balancing size with setup complexity. Some of the size constraints just can’t be overlooked. The Viya 3.5 image is nearly 20GB.

## Prerequisites:

There are a number of local disk volumes that must be mapped to the containers.

There are a few parameters that are used in the *docker-compose.yml* for setting volumes. You are also required to create the base path. Now postgres does not like to have its data volume messed with, so it is good if you can put it on a non-cloud drive (not OneDrive or iCloud for example). All of the other directories used here will flow from this so this is truly the base level for the data directories. This also includes the *cas* folders that Viya uses, as well as the *platform\_shared* folder that is a user folder shared across all volumes for your use as a programmer.

~~$~~ export PLPATH = /Users/samuelcroker

On your first build, run the build command. It may work simple with docker-compose up -d but this is a little more controlled.

~~$~~ docker-compose build

If you ever need to regbuild any of the services in the cluster, you can use this technique. For example, if you needed to rebulild the python/jupyter container you could run

~~$~~ docker-compose build an\_python

This would rebuild the python container while leaving the rest in place.

# Running and Re-Running the Cluster

Once you have run the initial build, and all of the build scripts finish without error you can start the cluster by typing

$ export PLPATH = ///base path

$ docker-compose up -d

Loading the DVD Rental Database into Postgres

First, attach a terminal to the postgres-sas container.

~~$~~ docker exec -it postgres-sas /bin/bash

Now you can work directly on postgres. This may be the only time you ever do this so don’t let it get to you.

postgres-# \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 +

| | | | | postgres=CTc/postgres

template1 | postgres | UTF8 | en\_US.utf8 | en\_US.utf8 | =c/postgres +

| | | | | postgres=CTc/postgres

(3 rows)

postgres-# exit;

Use \q to quit.

postgres=# exit;

postgres@xxx:/$ createdb dvdrental

postgres@xxx:/$ psql

psql (13.1 (Debian 13.1-1.pgdg100+1))

Type "help" for help.

postgres=# \l

List of databases

Name | Owner | Encoding | Collate | Ctype | Access privileges

-----------+----------+----------+------------+------------+-----------------------

dvdrental | postgres | UTF8 | en\_US.utf8 | en\_US.utf8 |

postgres | postgres | UTF8 | en\_US.utf8 | en\_US.utf8 |

template0 | postgres | UTF8 | en\_US.utf8 | en\_US.utf8 | =c/postgres +

| | | | | postgres=CTc/postgres

template1 | postgres | UTF8 | en\_US.utf8 | en\_US.utf8 | =c/postgres +

| | | | | postgres=CTc/postgres

(4 rows)

postgres=# \c dvdrental

You are now connected to database "dvdrental" as user "postgres".

dvdrental=# \dt

Did not find any relations.

dvdrental=# exit;

postgres@xxx:/$ pg\_restore -U postgres -d dvdrental dvdrental.tar

postgres@xxx:/$ psql

psql (13.1 (Debian 13.1-1.pgdg100+1))

Type "help" for help.

postgres=# \l

List of databases

Name | Owner | Encoding | Collate | Ctype | Access privileges

-----------+----------+----------+------------+------------+-----------------------

dvdrental | postgres | UTF8 | en\_US.utf8 | en\_US.utf8 |

postgres | postgres | UTF8 | en\_US.utf8 | en\_US.utf8 |

template0 | postgres | UTF8 | en\_US.utf8 | en\_US.utf8 | =c/postgres +

| | | | | postgres=CTc/postgres

template1 | postgres | UTF8 | en\_US.utf8 | en\_US.utf8 | =c/postgres +

| | | | | postgres=CTc/postgres

(4 rows)

postgres=# \c dvdrental

You are now connected to database "dvdrental" as user "postgres".

dvdrental=# \dt

List of relations

Schema | Name | Type | Owner

--------+---------------+-------+----------

public | actor | table | postgres

public | address | table | postgres

public | category | table | postgres

public | city | table | postgres

public | country | table | postgres

public | customer | table | postgres

public | film | table | postgres

public | film\_actor | table | postgres

public | film\_category | table | postgres

public | inventory | table | postgres

public | language | table | postgres

public | payment | table | postgres

public | rental | table | postgres

public | staff | table | postgres

public | store | table | postgres

(15 rows)

dvdrental=# select \* from actor limit 2;

actor\_id | first\_name | last\_name | last\_update

----------+------------+-----------+------------------------

1 | Penelope | Guiness | 2013-05-26 14:47:57.62

2 | Nick | Wahlberg | 2013-05-26 14:47:57.62

(2 rows)

dvdrental=# exit;