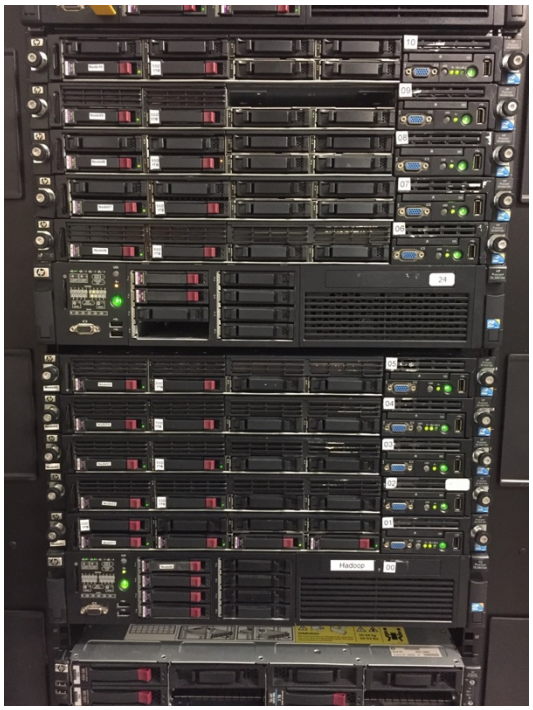
CSC570 Containerization for Big data

UNIVERSITY OF ILLINOIS SPRINGFIELD

Hands-On Exercise 3-1 Deploying Multi-Container Apps with Docker Compose

1 CSCluster at UIS



11-node Hadoop nodes

- CPU: Intel Xeon **4C/8T** per node
- Memory: **266 GB** in total
  - Master (head) node: 74GB RAM
  - 2nd Master node: 48GB RAM
  - 9 worker nodes: 16GB/node
- Storage: **10 TB SSD** in total
  - Master node: 400GB,
  - 10 worker nodes: 1TB SSD/node
- Hadoop: **Cloudera CDH 6.3**
  - HDFS, MapReduce
  - Spark 2, HBase, Hive
  - Pig, HUE, and etc.

VMs for Docker/K8S (You will use these VMs)

- CPU: Intel Xeon **8** core/node
- Memory: **16GB** RAM/node
- Storage: **100GB**

3-node Cassandra cluster (NoSQL-Column family DB)

- CPU: Intel Xeon **4C/8T** per node
- Memory: **48GB** RAM (16GB/node)
- Storage: **3TB SSD** (1TB/node)

PostgreSQL node (Relational DB)

- CPU: Intel Xeon **4C/8T** per node
- Memory: **18 GB RAM**
- Storage: **1TB SSD**

MongoDB node (NoSQL-Document DB)

- CPU: Intel Xeon **4C/8T** per node
- Memory: **16 GB RAM**
- Storage: **1TB SSD**

2 Accessing your VM for Docker and Kubernetes

2.1 Accessing campus network

If you are in UIS campus, you are fine. If you are not in UIS campus, you should install a Cisco VPN client software. The VPN client software gives remote users a secure and encrypted VPN (Virtual Private Network) connection to the UIS campus network. Please see below website, https://www.uis.edu/informationtechnologyservices/connect/vpn/

2.2 Accessing your VM using SSH

After you install the VPN client software and make a VPN connection to UIS network, you can access your VM using a terminal (Mac), PowerShell (Windows), or Putty (Windows).

- Check the IP address for your VM in the 'Course Information' under 'Modules' in Canvas

Type below command in your preferred SSH shell client:

```
ssh your-login@10.92.128.36
```

your-login: your UIS NetID (for example, **slee675** from slee675@uis.edu)
Initial password & IPs: See the Virtual Machine IPs page in the 'Course Information' under 'Modules' in Canvas.

After you logged in to your VM, you will see below prompt. The 'us2004lts' is a name of your VM and stands for 'Ubuntu Sever 20.04 LTS' that we are using as an OS.

```
your-login@us2004lts:~$
```

Example1: using terminal in Mac

```
LeeMBP15:~ sslee777$ ssh sslee777@10.92.128.36
The authenticity of host '10.92.128.36 (10.92.128.36)' can't be established.
ECDSA key fingerprint is SHA256:dXhKfHsYIXe/53hvU+HOK2V6fVrTbz/QxmUhpnPXpzA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.92.128.36' (ECDSA) to the list of known hosts.
sslee777@10.92.128.36's password: <== Use initial password until you change it
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-33-generic x86_64)
...

sslee777@us2004lts:~$
```

3 Deploying Multi-container Apps with Docker Compose

3.1 Overview

We learned how to create a custom image and run a container for PostgreSQL.

Now we want to run a web-based admin/devel tool, pgAdmin, for PostgreSQL and need to create another container for it. Docker compose make deploying multi-container apps easy. In this hands-on exerciswe, we will learn how to create a multi-container app – PostgreSQL and pgAdmin – with Docker compose. We will also figure out how to add another container for managing Docker environments, Portainer, as an assignment.

Docker Compose lets you

- Describe an entire app in a single declarative configuration file
- Deploy it with a single command
- Manage its entire lifecycle with a simple set of commands.
- Store and manage the configuration file in a version control system

Docker Compose

- Is an external Python binary that you have to install on a Docker host
- You define multi-container (microservices) apps in a YAML file
- Pass the YAML file to the docker-compose command line
- Compose deploys it via the Docker API

pgAdmin is the most popular and feature rich Open Source administration and development platform for PostgreSQL, the most advanced Open Source database in the world. See pgAdmin official site: https://www.pgadmin.org/

Portainer is a powerful, open-source management toolset that allows you to easily build, manage and maintain Docker environments. See Portainer official site: https://www.portainer.io/

Related Chapters in textbook

- Chapter 9: Deploying Apps with Docker Compose
- Chapter 8: Containerizing an app
- Chapter 7: Containers

3.2 Install Docker Compose

To install Docker compose, you just need to type below:
Ubuntu will install docker-compose and dependent packages.

```
sslee777@us2004lts:~$ sudo apt install docker-compose
[sudo] password for sslee777:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  python3-cached-property python3-docker python3-dockerpty python3-docopt
  python3-texttable python3-websocket
The following NEW packages will be installed:
  docker-compose python3-cached-property python3-docker python3-dockerpty
  python3-docopt python3-texttable python3-websocket
0 upgraded, 7 newly installed, 0 to remove and 21 not upgraded.
...
```

Let's check whether docker-compose command is working.

```
sslee777@us2004lts:~$ docker-compose version
docker-compose version 1.25.0, build unknown
docker-py version: 4.1.0
CPython version: 3.8.2
OpenSSL version: OpenSSL 1.1.1f  31 Mar 2020
sslee777@us2004lts:~$
```

3.3 Docker Compose file for PostgreSQL and pgAdmin

Docker compose uses YAML files to define multi-service applications. YAML is a subset of JSON, so you can also use JSON. The default name for the file is docker-compose.yml. Use the -f flag to specify custom filenames.

Let's start to create a directory for the app.

```
sslee777@us2004lts:~$ mkdir multiConApp
sslee777@us2004lts:~$ cd multiConApp/
sslee777@us2004lts:~/multiConApp$
```

Then create a file, docker-compose.yml, and copy and paste following compose file. To paste it in 'vi' editor, type "i" then paste it. To quit from vi, push "esc" and type, ":", then "wq"

```
sslee777@us2004lts:~/multiConApp$ vi docker-compose.yml
version: "3.7"
services:

  postgres:
    container_name: myPostgresCon
    image: postgres:12.1
    restart: always
    environment:
      POSTGRES_DB: myDB
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: mypassword
      PGDATA: /var/lib/postgresql/data
    volumes:
      - postgres-data:/var/lib/postgresql/data
    ports:
      - "5432:5432"
    networks:
      - pgnet

  pgadmin:
    image: dpage/pgadmin4
    restart: always
    environment:
      PGADMIN_DEFAULT_EMAIL: my-email@uis.edu
      PGADMIN_DEFAULT_PASSWORD: mypassword
      PGADMIN_LISTEN_PORT: 5050
    volumes:
      - pgadmin-data:/var/lib/pgadmin
    ports:
      - "5050:5050"
    networks:
      - pgnet
    links:
      - "postgres:pgsql-server"

networks:
    pgnet:
        driver: bridge

volumes:
  postgres-data:
  pgadmin-data:

sslee777@us2004lts:~/multiConApp$
```

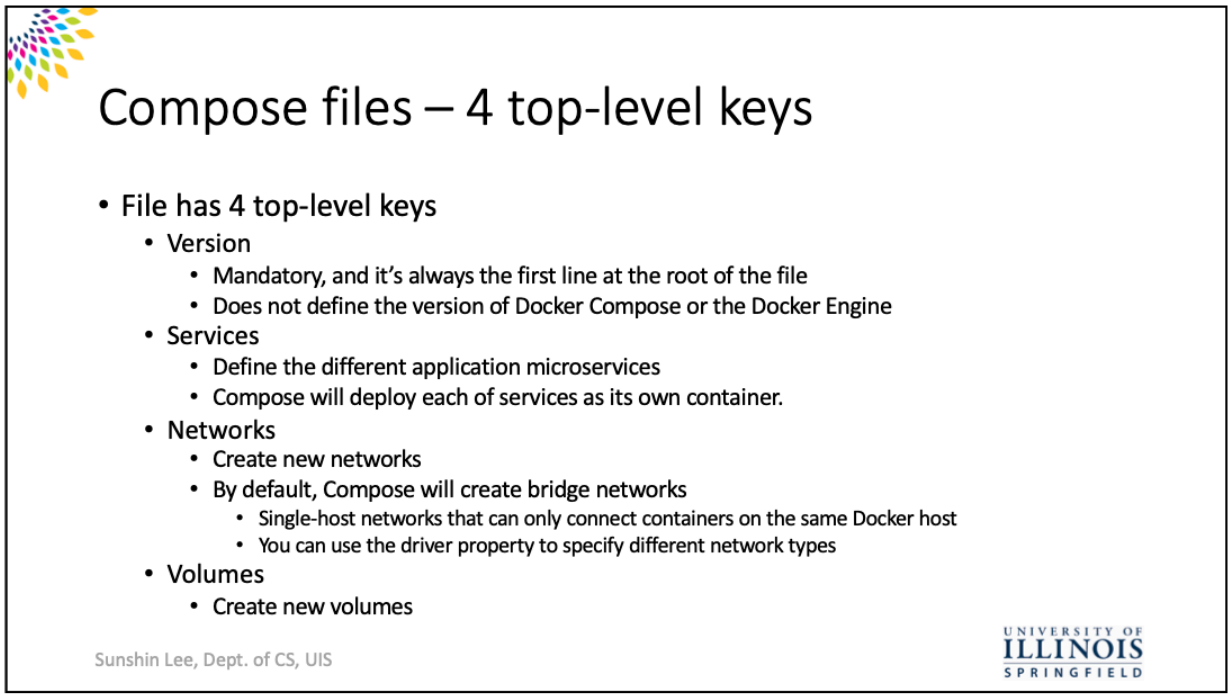
3.3.1 Top-level keys

In this compose file, we have 4 top-level keys (sections)

- The version of the compose file: we are using version 3.7
- The services which will be built
- The networks which connect the different services
- All used volumes

```
version: "3.7"
services:
...
networks:
...
volumes:
...
```

Also see the lecture slides



If you want to know more about Docker compose file version, see
https://docs.docker.com/compose/compose-file/compose-versioning/

3.3.2 Services

In services section, we configure two services: postgres and pgadmin. Docker will create two containers for them.

```
services:
  postgres:
       ...

  pgadmin:
       ...
```

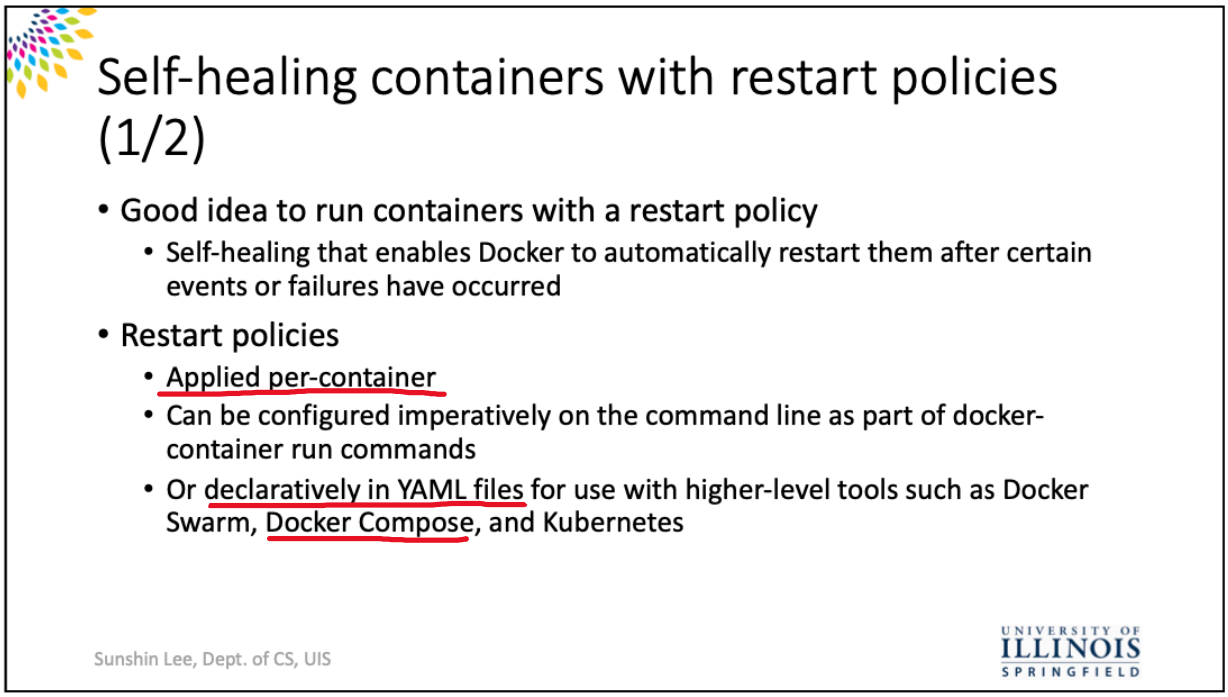
3.3.2.1 Services: PostgreSQL

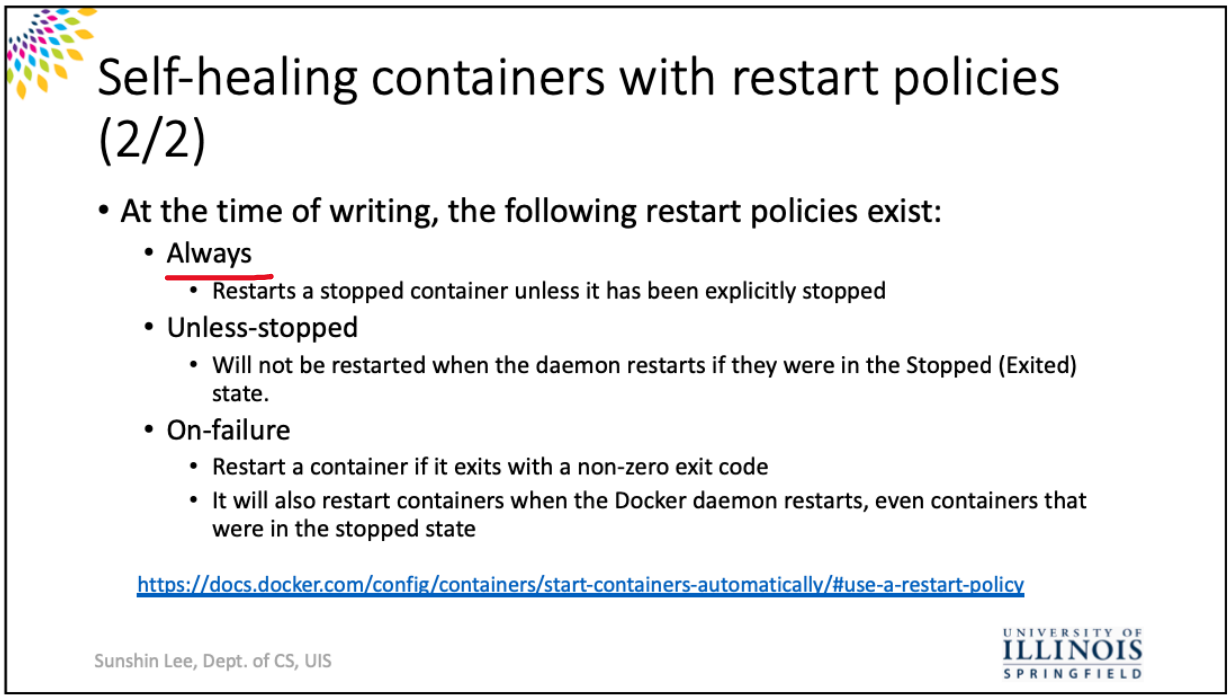
For postgreSQL, postgres service is configured.

```
postgres:
  container_name: myPostgresCon
  image: postgres:9.4
  restart: always
  environment:
    POSTGRES_DB: myDB
    POSTGRES_USER: admin
    POSTGRES_PASSWORD: mypassword
    PGDATA: /var/lib/postgresql/data
  volumes:
    - postgres-data:/var/lib/postgresql/data
  ports:
    - "5432:5432"
  networks:
    - pgnet
```

myPostgresCon (Container) is a container name of the postgres service. It will run the postgres:9.4 image (from DockerHub) in a Docker container.

Restart policy can be defined in 'restart' subsection. As restart policy is applied per-container, postgres and pgadmin service have restart policies as always. See lecture slides below.





This Compose file contains the following environment variables:

- POSTGRES_DB: myDB
- POSTGRES_USER: admin
- POSTGRES_PASSWORD: mypassword

The PGDATA environment variable is used to configure the PostgreSQL server to store the data to /var/lib/postgresql/data directory of the container.

In volumes, all the contents of the /var/lib/postgresql/data directory will be saved permanently in the postgres-data volume.

Ports: The container port 5432 (right) is mapped to the Docker host port 5432 (left).

Networks: This service will use pgnet (postgresql network) network.

3.3.2.2 Services: pgAdmin

For pgAdmin, pgadmin service is configured

```
  pgadmin:
    image: dpage/pgadmin4
    restart: always
    environment:
      PGADMIN_DEFAULT_EMAIL: my-email@uis.edu
      PGADMIN_DEFAULT_PASSWORD: mypassword
      PGADMIN_LISTEN_PORT: 5050
    volumes:
      - pgadmin-data:/var/lib/pgadmin
    ports:
```

```
      - "5050:5050"
    networks:
      - pgnet
    links:
      - "postgres:pgsql-server"
```

pgadmin service will run the dpage/pgadmin4 image (from DockerHub) in another Docker container.

As restart policy is applied per-container, pgadmin service have a value ‘always’ as restart policy.

Following environment variables is configured:

- PGADMIN_DEFAULT_EMAIL: For login. Please use your e-mail, netID@uis.edu
- PGADMIN_DEFAULT_PASSWORD: mypassword.
- PGADMIN_LISTEN_PORT: the default value is 5050

The PGADMIN_LISTEN_PORT is used to set the pgAdmin port 5050 in the container.

Volumes: All the contents of the /var/lib/pgadmin directory will be saved permanently in the pgadmin-data volume.

Networks: This service will use pgnet (postgresql network) network.

Links: A hostname alias pgsql-server to the db container is created. So, you can access the PostgreSQL server using pgsql-server as the hostname (no IP address required).

3.3.3 Networks

By configuring networks here, Docker will create networks, e.g., pgnet in this exercise. It’s a bridge mode network. For detail, see https://docs.docker.com/network/bridge/

```
networks:
    pgnet:
        driver: bridge
```

3.3.4 Volumes

Docker will create new volumes, e.g., postgres-data for postgres and pgadmin-data for pgadmin, so each container can use them.

```
volumes:
  postgres-data:
  pgadmin-data:
```

3.4 Running PostgreSQL and pgAdmin

Now it's time to run PostgreSQL and pgAdmin using Docker compose file. You just need to run a command below. We assume you already have docker-compose.yml in the multiConApp directory. (Note: Don't forget to change your e-mail address of PGADMIN_DEFAULT_EMAIL environment variable.)

First three lines show that Docker is creating one network and two volumes. It then pulling the images, postgres and pgadmin, for creating their containers. Finally, it created a container, myPostgresCon and multiconapp_pgadmin_1. We named myPostgresCon in the compose file. However Docker named pgadmin, multiconapp_pgadmin_1, because we didn't named it.

```
sslee777@us2004lts:~/multiConApp$ docker-compose up -d
Creating network "multiconapp_pgnet" with driver "bridge"
Creating volume "multiconapp_postgres-data" with default driver
Creating volume "multiconapp_pgadmin-data" with default driver

Pulling postgres (postgres:9.4)...
9.4: Pulling from library/postgres
619014d83c02: Pull complete
7ec0fe6664f6: Pull complete
...
Digest: sha256:42a7a6a647a602efa9592edd1f56359800d079b93fa52c5d92244c58ac4a2ab9
Status: Downloaded newer image for postgres:9.4

Pulling pgadmin (dpage/pgadmin4:)...
latest: Pulling from dpage/pgadmin4
cbdbe7a5bc2a: Pull complete
26ebcd19a4e3: Pull complete
...
Digest: sha256:b1f00b8163cf5689e720099daa0cc8ee61777a71b8ee6088d28e0a2bb1984dea
Status: Downloaded newer image for dpage/pgadmin4:latest
Creating myPostgresCon ... done
Creating multiconapp_pgadmin_1 ... done
sslee777@us2004lts:~/multiConApp$
```

To verify ports, run command below.

```
sslee777@us2004lts:~/multiConApp$ sudo netstat -tlpn
[sudo] password for sslee777:
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      906/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      991/sshd: /usr/sbin
tcp        0      0 127.0.0.1:36125         0.0.0.0:*               LISTEN      961/containerd
tcp6       0      0 :::22                   :::*                    LISTEN      991/sshd: /usr/sbin
tcp6       0      0 :::5432                 :::*                    LISTEN      844753/docker-proxy
tcp6       0      0 :::5050                 :::*                    LISTEN      844892/docker-proxy
sslee777@us2004lts:~/multiConApp$
```

As you can see, the port 5050 and 5432 are opened by Docker.

Run the following command to see how the ports are mapped

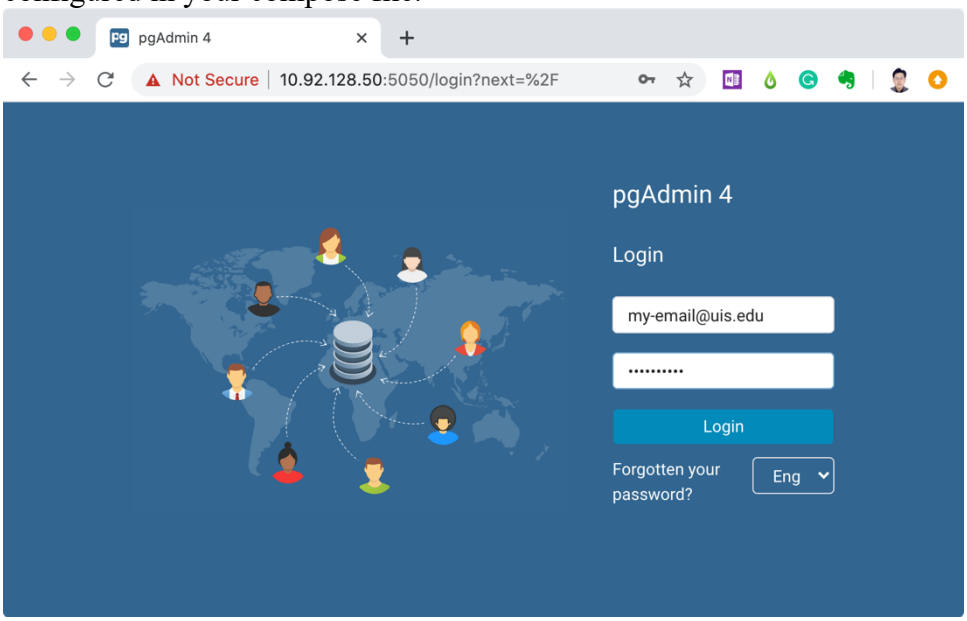
```
sslee777@us2004lts:~/multiConApp$ docker-compose ps
        Name                       Command              State                    Ports
-------------------------------------------------------------------------------------------------------
multiconapp_pgadmin_1   /entrypoint.sh                  Up      443/tcp, 0.0.0.0:5050->5050/tcp, 80/tcp
myPostgresCon           docker-entrypoint.sh postgres   Up      0.0.0.0:5432->5432/tcp

sslee777@us2004lts:~/multiConApp$
```

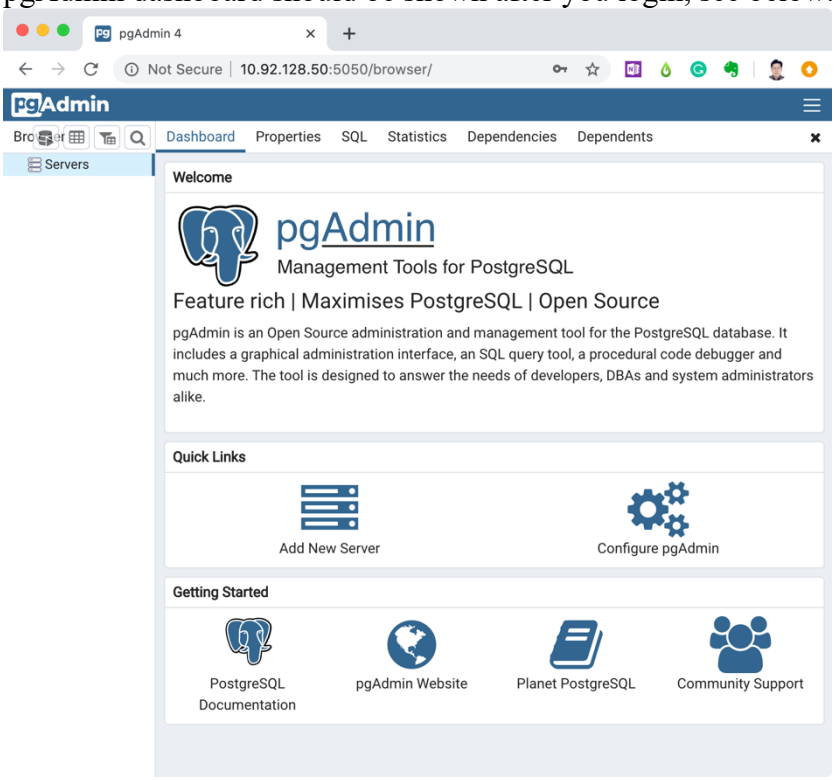
For the pgadmin service, Docker host port 5050 is mapped to the container TCP port 5050.
For the postgresql service, Docker host port 5432 is mapped to the container TCP port 5432.

3.5 Accessing PostgreSQL DB from pgAdmin

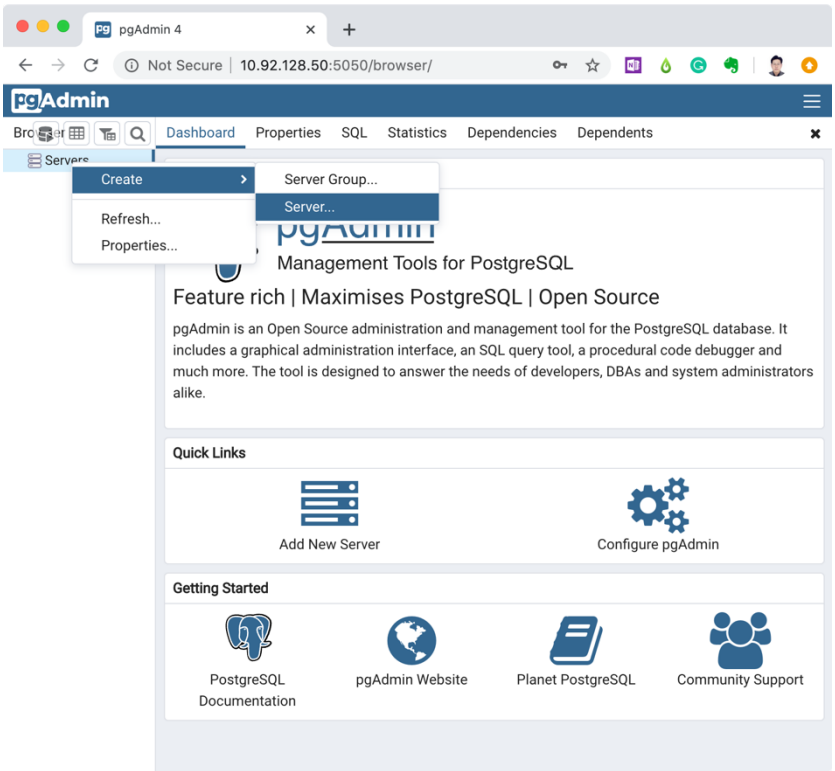
Visit you VM host with 5050 port, e.g. http://10.92.128.50:5050/
You should see the pgAdmin 4 login page. Login with your email and password that you configured in your compose file.



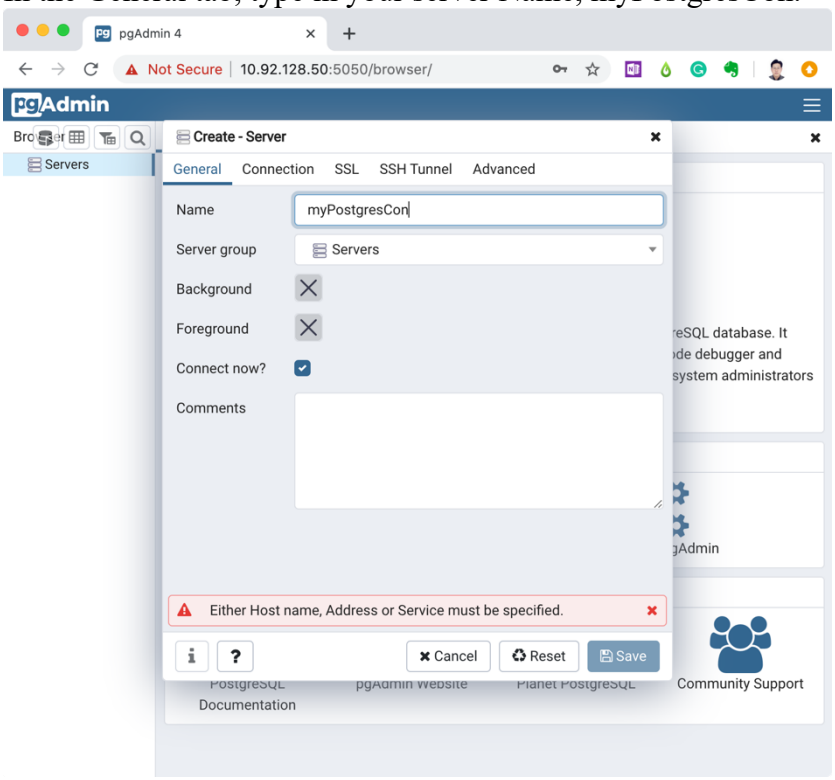
pgAdmin dashboard should be shown after you login, see below.



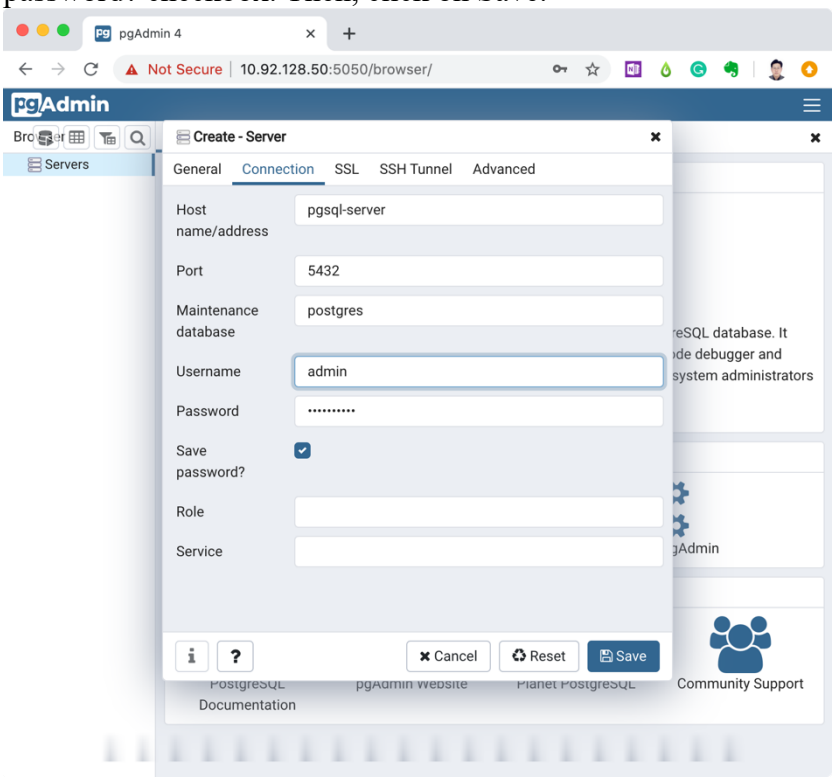
To add the PostgreSQL server running as a Docker container, right click on Servers, and then go to Create > Server.



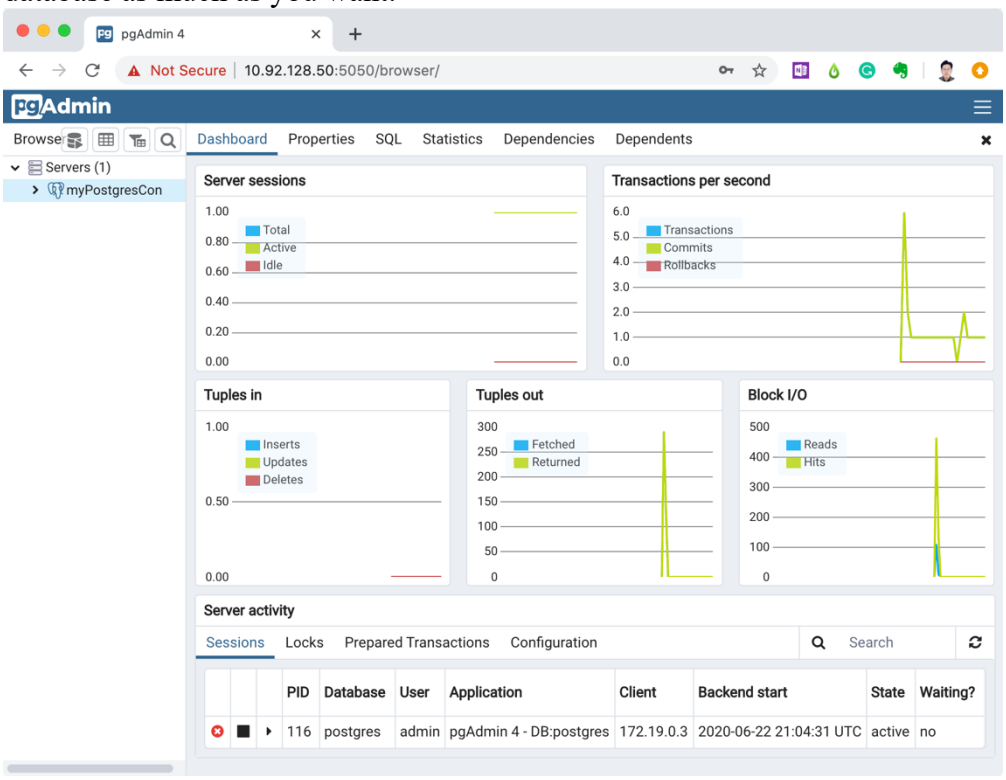
In the General tab, type in your server Name, myPostgresCon.



Go to the Connection tab and type in pgsql-server as Host name/address, 5432 as Port, postgres as Maintenance database, admin as Username, mypassword as Password and check Save password? checkbox. Then, click on Save.



pgAdmin 4 should be connected to your PostgreSQL database. Now, you should see dashboard if you click myPostgresCon under Servers (left). Now, you can work with your PostgreSQL database as much as you want.



3.6 Stopping PostgreSQL and pgAdmin

Stopping multi-container apps using Docker compose is quite easy.

```
sslee777@us2004lts:~/multiConApp$ docker-compose down
Stopping multiconapp_pgadmin_1 ... done
Stopping myPostgresCon         ... done
Removing multiconapp_pgadmin_1 ... done
Removing myPostgresCon         ... done
Removing network multiconapp_pgnet
sslee777@us2004lts:~/multiConApp$
```

The services should be stopped.

We may want to remove data/volumes for the app/services. List volumes first.

```
sslee777@us2004lts:~/multiConApp$ docker volume ls
DRIVER              VOLUME NAME
...
local               multiconapp_pgadmin-data
local               multiconapp_postgres-data
sslee777@us2004lts:~/multiConApp$
```

After you check the volumes, remove them.

```
sslee777@us2004lts:~/multiConApp$ docker volume rm multiconapp_pgadmin-data
multiconapp_postgres-data

multiconapp_pgadmin-data
multiconapp_postgres-data
sslee777@us2004lts:~/multiConApp$
```

4 Submit

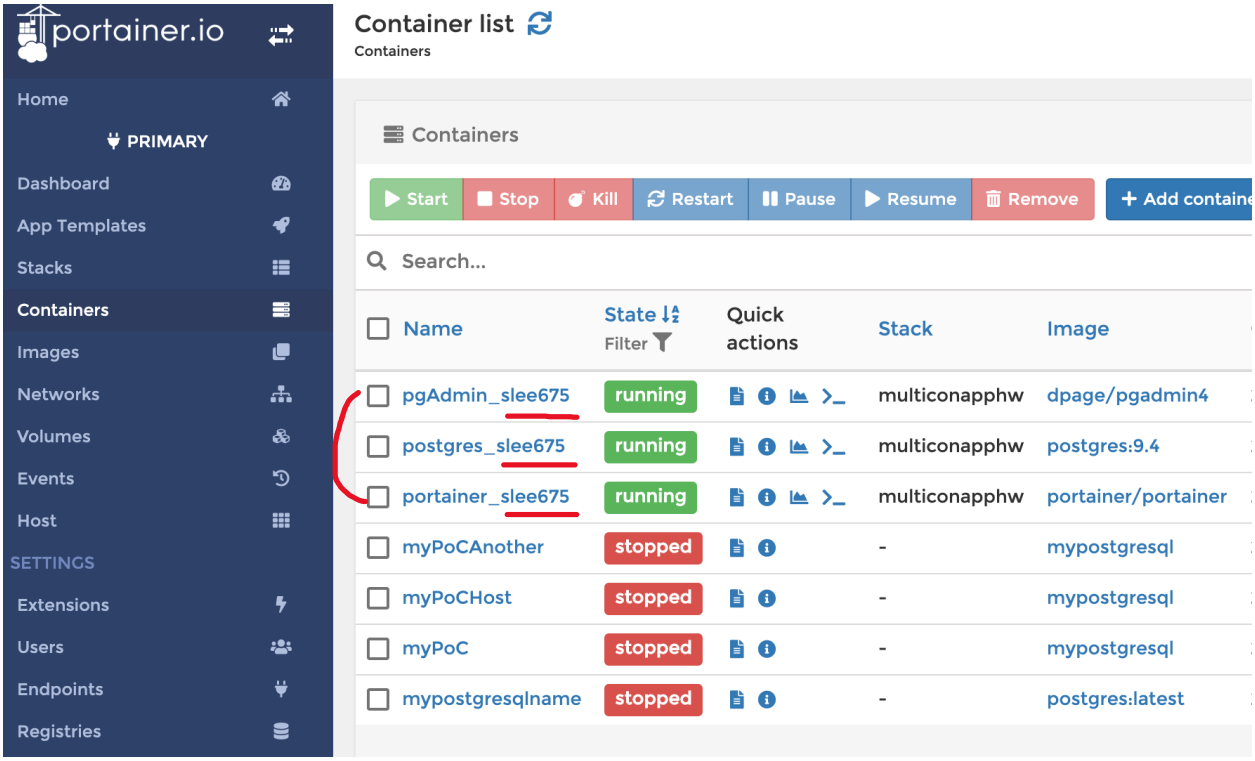
Submit a word document (docx, doc, or PDF) to Canvas.
Assignments

- Run your code/scripts, take screenshots, and explain it.
    a. Section 3.2 Install Docker compose
    b. Section 3.3 Create your Docker compose file for PostgreSQL and pgAdmin
        i. Don't forget to change e-mail and passwords in the file
    c. Section 3.4 Running PostgreSQL and pgAdmin
    d. Section 3.5 Accessing PostgreSQL DB from pgAdmin
        i. Dashboard screenshot only
- Add Portainer in your app, multiConAppHW (PostgreSQL + pgAdmin + Portainer).
    a. Portainer is a powerful web-based Docker management tool. See https://www.portainer.io/
    b. Create a new compose file
        i. Create a new directory, multiConAppHW
        ii. Under the directory, create a new compose file, docker-compose.yml, based on the file for the multiConApp in Section 3
        iii. Important: Name your containers/services and add your NetID
            1. Use 'container_name:' in the compose file
            2. For example,
                a. container_name: postgres_slee675
                b. container_name: pgAdmin_slee675
                c. container_name: portainer_slee675
    c. Add a service for Portainer in the compose file
        i. Don't forget to add container name, image, restart policy, environment, network, volume, and/or etc. for the service as you need.
        ii. Don't forget to name your Portainer container, see above
    d. Add top-level network(s) and/or volume(s) for the service as you need
    e. Take screenshot(s) of the compose file, running the app, listing containers using 'docker container ls', and Portainer showing three containers for the three services (PostgreSQL + pgAdmin + Portainer), see attached screenshot.

(Hint: Read articles in Reference)

If you have any problems or questions regarding this exercise, post messages in the 'Discussions' in Canvas. Posting exact codes or links to an article that have exact codes are not allowed.

A screenshot of Portainer showing containers. Three containers are running and their names are ended with my NetID, slee675.



5 Reference (Note: Links may not work in PDF, try to copy&paste it)

1. Set up a PostgreSQL server and pgAdmin with Docker, https://linuxhint.com/postgresql_docker/
2. Easy PostGreSQL 12 and pgAdmin 4 Setup with Docker, https://info.crunchydata.com/blog/easy-postgresql-12-and-pgadmin-4-setup-with-docker
3. Postgresql & PgAdmin powered by compose, https://github.com/khezen/compose-postgres
4. Use postgresql and pgAdmin in docker, https://gist.github.com/tingwei628/8584ddefc5d8e85f73566d5ab96bdc84
5. pgAdmin in Docker: Provisioning connections and passwords, https://technology.amis.nl/2020/01/02/pgadmin-in-docker-provision-connections-and-passwords/
6. Why you should use Docker Compose, https://www.enterprisedb.com/postgres-tutorials/why-you-should-use-docker-compose
7. Compose file version 3 reference, https://docs.docker.com/compose/compose-file/
8. The definitive Guide to Docker compose, https://gabrieltanner.org/blog/docker-compose
9. pgAdmin Official site: https://www.pgadmin.org/
10. pgAdmin Container Deployment Doc.: https://www.pgadmin.org/docs/pgadmin4/development/container_deployment.html
11. Containers: Portainer Review, https://medium.com/better-programming/portainer-review-382575dabb76