Lab Homework 1: The Lab Environment

# Learning Objectives

In this homework, you will learn how to:

* Connect to the ITELL vlab or setup your computer to run the labs.
* Use Docker to manage the containers which make up the database systems in this course
* Connect to SQL Server.

# Overview

In this course we will explore several database systems:

* SQL Server (By Microsoft),
* Hadoop (By Cloudera)
* Apache Cassandra
* MongoDB
* Redis, and
* Kafka (By Confluent)

How can we install and configure all these systems? The short answer is, thanks to Docker and Git we won’t have to! Using Git, we will access the configuration files necessary to deploy each of these database systems as a series of Docker containers. This approach provides a consistent setup for each student; one that is easy to use and can be re-created easily. It’s the ultimate learner’s playground for if you mess things up you can always rebuild the environment back to its initial state with a simple command. This approach is known as Infrastructure as Code among the DevOps community, and we’re merely applying it to this course to make it easier for you to explore these technologies in a safe and consistent manner. Enjoy!

## Where will you run the containers?

You have two options when it comes to running the database system containers. Run them on your computer or ours (the ITELL vLab).

### Your Computer

If you want to run them on **your computer**, you will need:

* Windows 10 Pro or Mac OSX
* A Computer with 8 GB RAM Minimum, (16 GB Recommended)
* An Installation of Git <https://git-scm.com/downloads>
* An Installation of Docker Community Edition: <https://store.docker.com/search?type=edition&offering=community>
* On Windows you will need SQL Server Management Studio:  
  <https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms>   
  On MacOS you will need SQL Operations Studio: <https://github.com/Microsoft/sqlopsstudio/releases>

### Our Computer

You will receive a Windows 10 virtual machine in the ITELL vLab with the required software installed for you. You will have to access our computer remotely, which is explained in the next section.

# Part 1: The ITELL vLab

**NOTE: If you are running the labs from your own personal computer, you can skip this part. However, most of you will be using the ITELL vLab.**

In this part you will become familiar with the ITELL (vLab). The ITELL vLab provides a logged in user with a means to access other computers remotely. Unlike remote lab, these computers are configured for very specific tasks, usually associated with a course. Because it would be wasteful to setup a physical computer for each student, ITELL vLab uses virtual machines (VM) which are virtual computers running on top of a physical computer. All the virtual machines running on the host computer share its resources of CPU, RAM and Disk.

Follow these instructions to connect to the ITELL vLab: <https://answers.syr.edu/x/coENB>

Once you are connected, follow these instructions to navigate to the Virtual Machine used in this course, and Power It On, and access the Console: <https://answers.syr.edu/x/bQL-Aw>

For this class, the Virtual Machine (VM) we will use runs the Windows 10 operating system. The previous instructions explained how to power on your virtual machine and access the console (a.k.a. Screen / Keyboard and Mouse) for this virtual machine.

By now, you're probably at a login window and wondering: How do I log on? Type:

Username: **LocalAdmin**   
Password: **SU44orange!**

This will log you in to your virtual machine and bring you to the desktop. Here's where you'll stay to work on the homework.

NOTE: When you are done with the lab it's a good idea to power off your virtual machine. That way the host computers will reclaim resources for other students and their lab/homework activities! The instructions to power off are here: <https://answers.syr.edu/x/bQL-Aw>

IMPORTANT: Practice accessing the ITELL environment, powering your VM on and off, opening the console and logging on. You will need to do this a few times each week so its important to get the process down

# Part 2: Getting the Lab Containers and Supporting Files

With the pre-requisites out of the way, let's cover how to get the lab containers and supporting files. Everything you need is stored in a git source code management repository hosted on GitHub. This section walks you through the basics of accessing your These are stored in a git repository, which is a form of version control for code. This allows changes to be made and labs to be updated while you're doing the them! That way if there's a mistake or improvement the change can be made, and all the students can easily get the corresponding updates.

## Cloning the Labs and Datasets

The git repositories for the course labs and datasets can be found on the GitHub website at <https://github.com/mafudge/adv-db-labs> and <https://github.com/mafudge/datasets> respectively. Let's use Git to copy them to your computer, which is called a *clone* in git nomenclature.

1. Open the PowerShell prompt on Windows or the Terminal on MacOS.
2. First let’s clone the labs. From the command line, type:  
   git clone https://github.com/mafudge/adv-db-labs.git  
   press ENTER to submit the command. It will take a few seconds to copy the code locally.
3. Next, let’s clone the datasets. From the command line, type:  
   git clone https://github.com/mafudge/datasets.git  
   press ENTER to submit the command. It will take a few seconds to copy the code locally. You always need to press ENTER to submit the command, so we will leave that out from now on.
4. You only need to clone the repositories once! Both now live on your computer in folders called adv-db-labs and datasets. So how do you know it worked? Or that you’re in the right place? At the command prompt,
   1. On windows type:   
      dir
   2. On OSX type: (NOTE: that a lowercase “L”)  
      ls -l

If you are in the correct folder you should see two folders named adv-db-labs and datasets respectively.

## Accessing the Repository folder

Cloning is a one-time deal, but you'll always need change the working directory to the repository folder.

1. Open the PowerShell prompt on Windows or the Terminal on MacOS.
2. Type:  
   cd adv-dv-labs  
   to change into the repository folder.   
   You should notice the command prompt has changed to reflect the current folder.
3. You can verify you're in the git repository by typing the following at the command line:  
   git status   
   It should say something like on branch master to let you know you're working in a folder which is being tracked by git.

### What’s in the Repository?

The git repository is organized by folders. Each folder contains the setup and configuration information for a database system. The following folders are present:

* Cassandra – The Apache Cassandra noSQL column store database <http://cassandra.apache.org/>
* Hadoop – The Cloudera’s Quickstart Hadoop distribution <https://www.cloudera.com/downloads/quickstart_vms.html>
* Kafka – Confluent’s Kafka Platform with KSQL <https://www.confluent.io/>
* MongoDB - MongoDB, A noSQL document oriented database <https://www.mongodb.com/>
* Mssql – Microsoft SQL Server’s Latest version running on Linux<https://hub.docker.com/r/microsoft/mssql-server-linux/>
* Redis – Redis, a noSQL key-value data store<https://redis.io/>
* Other – other setups which might be used in the future

# Part 3: Common Docker Commands

In this final part, we will explore the common docker commands we will use to manage the database environments.

## Setup

1. Open the PowerShell prompt on Windows or the Terminal on MacOS.
2. Type:  
   cd adv-dv-labs  
   to change into the repository folder.
3. In this example, we will use the mssql database, so we need to change the working directory to that folder, type:  
   cd mssql  
   if you are in the correct folder, your command prompt should have mssql in it.

We are now in the folder containing the configuration data for the Microsoft SQL Server environment. The specifics of each environment will be covered in other labs. Each folder contains a docker-compose.yml file which defines the dependencies (networks, storage and software) specific to each environment. What’s important at this juncture is that the commands you use to manage any environment are the same.

## Docker Compose Commands

Here’s a quick summary of the common docker-compose commands we will use in the course:

|  |  |
| --- | --- |
| **Command** | **What it does** |
| docker-compose up -d | Creates and runs the environment specified in the docker-compose.yml file. The first time you do this it might take some time as your system will need to download the images associated with the containers. |
| docker-compose ps | This command will show you what services are running as part of the environment. It is a useful command to check the status of your environment to see if its up or down. |
| docker-compose stop | This command stops the running containers, but keeps the changes made in them. This is crucial if you want to shut down an environment but not lose any data or changes you’ve made. You can restart the containers with docker-compose start or docker-compose restart, or even docker-compose up -d |
| docker-compose down | This command stops and removes all running containers and removes any dependencies. It does a complete tear-down of the environment and will reset the container data back to its initial state. Only do this when you are done using the environment, or desire a reset of the environment’s data. To completely reset an environment, type:  docker-compose down -v –rmi all |

## Walk-Through the Commands

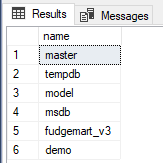
Let’s walk-through the docker commands connect to SQL Server, and create our databases:

1. Bring up the environment, type:   
   docker-compose up -d
2. It might take some time for the process to complete the first time as images need to be downloaded. When it’s done, check what’s running, type:   
   docker-compose ps

You should see the **mssql** service running on port **1433**

1. Let’s connect to SQL server open **SQL Server Management Studio** (on Windows) or **SQL Server Operations Studio** (if you’re on a Mac). From this point forward we will call this your SQL Client. Use the SQL Client to connect to SQL server with:
   1. Server name: **localhost**
   2. Authentication: **SQL Server Authentication**
   3. Login: **sa**
   4. Password: **SU2Orange!**
2. After you’ve connected, open a new query window by pressing CTRL+n.
3. Create the fudgemart and demo databases from the query window, type:

CREATE DATABASE fudgemart\_v3  
GO  
CREATE DATABASE demo  
GO  
SELECT name FROM sys.databases

1. Press the Run or Execute button (A green triangle in the toolbar) to execute the query on the server. In the output you should see both databases were created:  
   
2. Next you need to run the script to create the tables and data for **fudgemart\_v3**.
   1. Open a new query window, by pressing CTRL+n:
   2. Get the SQL Script at the following url: <https://raw.githubusercontent.com/mafudge/datasets/master/fudgemart/mssql.sql>
   3. Press CTRL+a to select the script on the webpage.
   4. Press CTRL+c to copy the script
   5. Switch back to the query window and press CTRL+v to paste the script.
   6. Execute the script. It will take a bit to complete the script.
3. Check to make sure the **fudgemart\_v3** database has tables and those tables have data. This is left as an exercise for the reader. When you are done, exit your SQL Client.
4. From the command-line, type:  
   docker-compose stop

To stop the mssql service.

1. Verify it has stopped. Type:

docker-compose ps  
Notice how the state no longer says **Up** but instead says **Exit 0**

1. As an exercise, start the environment back up and reconnect your SQL Client. Are your databases still there? They should be!
2. This next part is going to demonstrate that the environment can be reset back to the initial state whenever this is required. Type the following:  
   docker-compose down -v --rmi all  
   This will destroy all artifacts dependent on the environment. Now repeat steps 1-9 of this lab so that you have the necessary environment created.

# Summary

Throughout the video coursework your professor will provide live demos of using the technology of this course. You are encouraged to follow along with these demos. You can find the instructions for how to setup the environment in the **Setup** section of your weekly homework.

# Exercises

1. What would be the command to bring up the redis environment? How is the command different from the mssql environment? How is it the same?
2. Where is the specific configuration information about each environment stored?
3. Explain the difference between stopping an environment and bringing it down. Elaborate with use-cases for each.
4. What happens when you bring up an environment that is already up?
5. What was the most difficult aspect of this lab? What changes could be made to make it less difficult?

# Turning it in

Take your copy and paste each of the solutions to the exercises into the submission template file included with this assignment. Make sure your name and SU email are at the top and turn in your work through the course learning management system.