Due: 11/10/2018 by midnight

PROJECT 2: EMULATING PUB/SUB DISTRIBUTED SYSTEM USING DOCKER CONTAINERS

1. Problem statement: Publish/Subscribe (or pub/sub for short) is a popular indirect communication system. Pub/sub systems disseminates events to multiple recipients (called subscribers) through an intermediary. Examples of successful pub/sub include Twitter and "Bloomberg terminal"-like financial systems. In this project, we will emulate a pub/sub system using the lite weight virtualization or container technology in Docker technology.

2. Learning outcomes:

1. Implement a centralized pub/sub distributed application.
2. Implement a distributed pub/sub distributed application using all nodes managing subscribers and events.
3. Implement a distributed rendezvous based efficient pub/sub system.
4. Deploy a Docker node and a set of interacting Docker images to emulate a pub/sub application.

3. Preparation before lab

1. Download and deploy Docker: "Docker is a computer program that performs operating-system-level virtualization, also known as "containerization".
2. Read and understand the pub/sub model described in section 6.3.
3. You can work in groups of two or by yourself. Choose the right partner who compliments your abilities so that you can learn from each other through the process.

4. Problem Description

You are required to implement a pub/sub system in stages. The figure below gives a high level view of the pub/sub distributed systems model. It shows a set of subscribers who are interested in notification of events that a set of publishers publish. Figure 1 shows a centralized system with a simple interface subscribe (), publish (), and notify ().

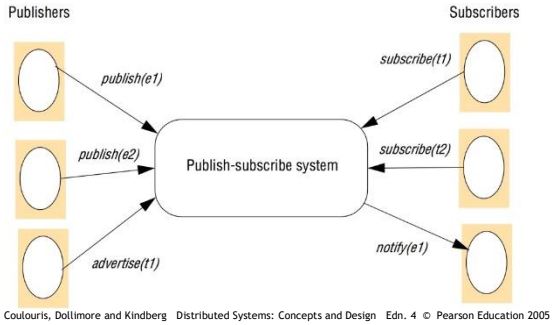

Figure 1: Pub/Sub Distributed System Model from Section 6.3 Coulouris Text

Read the applications of Pub/Sub, the programming model, implementation issues (centralized vs distributed), pages 241-253.

Phase 1: (25%) Download Docker, install it and learn how it works. Create a web interface where you can input ("TextBox") a small program in any language of your choice, and a command ("button") that will load the program in the Docker and execute it. The result of execution is display on the web output area.

Phase 2: (30%) Now that you are familiar with Docker, implement a centralized version of the pub/sub application. You will need to implement the functions described in the Figure 1, the event generator and subscription generator for fully testing the application.

Phase 3: (25%) Next step is to implement the distributed version of the pub/sub as described by rendezvous-based routing described in Figure 6.12 of Coulouris text.

Good design: (10%) Discuss the choice of technologies and UI details in your report. Comment you code appropriately. Provide a detailed design representation like figure 1, customized to your design and technologies you used.

Directory structure, documentation and readme: (10%) Provide a readme file that provides details of how to run deploy and run your code. Feel free to include screen shots of a working application.

5. Project Implementation Details and Steps:

1. **Building systems using Docker**:
   Containerization is a significant concept in modern systems development. This project gives you a hands-on experience. Building systems using Docker will provide skills that will be useful well beyond this project.

2. **Working with web interface to invoking Docker:**
   This is an essential software development skill. Implementing the web stack using Docker backend is useful for prototyping systems as well as building production systems. See https://codecon.bloomberg.com/

3. **Building a distributed pub/sub system***:*
   You can build a distributed pub/sub using multiple physical servers, multiple virtual machine spun off on a powerful physical server. In this case, we use lightweight virtualization to emulate the pub/sub system. In fact, with the experience you gain by this multiple communicating Docker virtual machines you should be able to implement any distributed system!

4. **Study, understand and design the system depicted in Figure 1:**
   Review the functions and the working of the pub/sub system described in Figure 1. Design the architecture, data structures, and functions needed to implement the system: first the centralized and then the distributed. Also decide the topics for subscriptions. Choose topics of interest to you. There are at least two issues that may need special programming capability: Generation of subscriptions (random) and generation of events (again random). Also consider a mechanism for displaying the notifications on the web interface. Make the web interface simple and easy to use. Plan the layout well.

5. **Design, implement the pub/sub system in phases:**
   You are free to use any language to implement the system; that is the beauty of the interface-implementation pattern and the Docker-based containerization!
   Implement the system in phases and save each version. Do not overwrite. We want to see incremental development steps. This also lets to plan your time so that you do not wait till the last minute to mash something together to satisfy the requirements. Also please DO NOT implement all phases at one time and submit one package. We want to observe the incremental development. We will take off point if we do not see the evidence for incremental development. This is a requirement.

6. **Deploy the integrated system and test it:**
   The various components listed above should have been deployed and tested individually. In this step you will run the entire integrated system. You can create a good test data that will completely test all the components of your system.

6. Project Deliverables:

- A standard directory structure for the three versions (including Phase 1's just Docker) of the software developed.
- A readme files documenting all the details from APIs used to how to run the application and any special features. Don't forget to include screen shots of the application running.
- You may also submit other media such as a video of your application working (optional).

7. Submission Details:

Create a compressed deployable distribution of your project. Use the steps given in the document for project 1. Submit it online on the CSE server timberlake. You should include all the details about the technology requirements to deploy and use your software and also a readme file providing clear instructions as to how to deploy and use your software.

8. First steps:

There is a lot of new stuff you will have to learn and do. You will definitely need the whole month.
Choose the design and technology for boxes and write up an abstract of 100 words with a suitable title. Submit it on timberlake, the CSE server before 10/21/2018 midnight.

9. References:

1. Pub/sub: https://cloud.google.com/pubsub/architecture
2. Docker Containerization: https://www.docker.com/
3. If you find any interesting web resource please share it with me. I will update this document.

10. Good luck, start your work right now.