Assignment 3: Virtualization using Docker

This assignment has two parts. In the first part, you will containerize your services and deploy them using Docker Compose. Then, in the second assignment, you will setup a Kubernetes cluster and then deploy your containers there.

Assignment 3.1: Container Virtualization

Background:

Docker[1] containers help to wrap up a piece of software in a complete file system that contains everything it needs to run code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in. Containers have similar resource isolation and allocation benefits as virtual machines, but a different architectural approach allows them to be much more portable and efficient.
Docker runs on a variety of Linux distributions; Docker does not run natively on OS X and Microsoft Windows. Installing Docker on OS X and Microsoft Windows platforms is done via a virtual machine.

Assignment:

In this assignment, you continue to develop your multi-user URL shortener service from assignments 1 and 2. This time, you will containerize your services using Docker, and then write a Docker Compose[2] file to manage all of your containers.

In addition, you should make sure that your URL shortener is persistent across reboots; so make sure that whatever service you are using to store data writes its contents to a file, and then use a Docker volume to make this persistent.

A good tutorial to start with Docker would be this one, but there are a lot of other resources available because Docker and Docker Compose are very widely used. Just remember to cite any you are using, as per the important notes in the bottom of this document.

For this half of the assignment, make sure that you pay attention to the following:

---

[1] http://sewiki.iai.uni-bonn.de/teaching/lectures/ise/2008/assignment_i
[2] https://docs.docker.com/compose/

1. Ensure that your containers are as efficient as possible. Pay attention to the ordering of your commands and try to include the minimum number of dependencies and source files.
2. Remember that your services must be accessible from the outside of your machine, not just from within the Docker container.

As usual, you can earn bonus points for anything else you implement but is not covered here. The same limitations apply as in the first assignment (i.e., you must implement the specification correctly to earn the points and you must mention/describe your bonus in your demo/report).

Implementation

We imagine that using Docker and Docker Compose for this assignment could be practical. If you are on Linux, you can typically download it through your distribution's package managers. On Windows or macOS, download Docker Desktop[3], which includes all the command-line tools as well.

For testing your services, we recommend the same tools as in the previous assignment.

Grading

Your grade for part 3.1 of the assignment is based on the following:

1. *Code* – Your implementation and configuration files will account for half of your grade of part 3.1. There will be a **demo** for this assignment as well, which you must show during the lab sessions *before* the submission deadline (like before). Highlight the following during your demo:
    a. Show to us that you deploy your services through Docker and that they both still work (no need to go over all paths, just a few examples of both the shortener and authentication service).
    b. Show that your service is now persistent across restarts of the services. Your services' data should also survive a `docker compose down` (i.e., destruction of the containers themselves).
    c. Describe anything you did to minimize the size/build efficiency of the containers.
    d. **[Bonus points]** Highlight any other aspect of your design that you think is relevant / you are proud of, most notably any bonus things you did.
2. *Report* – The other part will be graded based on your report. Like before, it doesn't have to be a full scientific report with introduction, background, etc., but it should provide us with an overview of what you implemented. Specifically, for this part:
    a. Describe your implementation (max 1 page). You shouldn't describe your Dockerfile step-by-step, but instead focus on the more key design decisions you made, such as how you made your

[3] https://www.docker.com/products/docker-desktop/

services persistent or what you did to ensure the compactness of your containers. Also include any other part you want to highlight.

b. Anything you did for the bonus part of assignment 3.1. While there is no page limit to this part, try to keep it succinct to shorten grading times.

Note, though, that your report should also include assignment 3.2, and any common aspects (see the section after assignment 3.2).

Tips

1. You might have already implemented (large parts of) this assignment as bonus in the previous assignments. If so, try to see if you can add some nice new bonus features here.
2. To keep things simple, you may assume that whatever implementation you use (e.g., an external database, the filesystem, etc) does not cause race conflicts if multiple services are accessing it at the same time.
3. A nice bonus for this part might be to implement an nginx proxy[4] to make your services available under a single port, if you haven't done so already.

[4] https://www.nginx.com/

Assignment 3.2: Container Orchestration

Background:

For the second part of the assignment, we will properly deploy your containers on a Kubernetes[5] cluster, which is a system for automated deployment and management of containers across multiple machines (nodes).

The main advantage to deploy containers on a system such as Kubernetes is to be able to harness multiple resources for your services, while controlling everything from a centralized access point. This plays particularly well to microservice architectures, where we can easily host parts of our code on different machines, and independently scale them to deal with different loads. Kubernetes can be used for all of this.

Assignment:

This part of the assignment requires you to first deploy a Kubernetes cluster on a few virtual machines provided by the University of Amsterdam. Specifically, they are hosted on the OpenLab[6] infrastructure, which is provided by the MNS research group[7].

You will be provided with 3 VMs per group, which are running Debian (this is nearly the same as Ubuntu, but this might matter in some cases – installing Docker is one of them, since Debian uses different keys). It makes sense to prepare one of them to be used in the *control plane*, which means that it will be used to manage the other two VMs, which you can setup as *worker nodes.* These two will be the VMs that actually run your containers.

A tutorial on how to setup these nodes is available on the assignment page in Canvas.

After you have deployed your cluster, you should deploy your services on your own cluster. To do so, create deployments[8] for each of your services, and, for the purpose of this assignment, make sure that at least one of your services has 3 replicas running. Then, create services[9] for your... services to make them externally accessible. Think about the best kind of service to use for your system.

Implementation

---

[5] https://kubernetes.io/
[6] https://mns-research.nl/open-lab/
[7] https://mns-research.nl
[8] https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
[9] https://kubernetes.io/docs/concepts/services-networking/service/

Once again, using Kubernetes for this half of the assignment might be advisable. The tutorial document contains the exact tools you will need (`kubeadm`, `kubectl`, `kubelet`) and how to install them. Also use the tools of previous parts/assignments to test your containers/services.

Grading

Your grade for assignment 3.2 is based on the following:

1. *Code* – Your implementation and configuration files will account for 50% of your grade for assignment 3.2. This is graded during the same **demo** as for assignment 3.1, so be sure to show in addition:
    a. That your Kubernetes cluster is up-and-running.
    b. That your services deploy on your cluster and work when accessed from your *local* machine (to show that your services are publicly reachable).
    c. Show that you have three replicas of a particular service, and that all of them offer a consistent view of the database.
    d. **[Bonus points]** Highlight any other aspect of your design that you think is relevant / you are proud of, most notably any bonus things you did.
2. *Report* – For assignment 3.2, the other 50% will be given based on another report part. You should include the following in the same report document as for 3.1:
    a. Describe your experience (max ½ page). This part of the assignment does not have a lot of design associated with it, so instead describe your experience with working with Kubernetes and whether there were any major challenges you encountered. You can, however, discuss some implementation too if you deem it relevant.
    b. Give answer to the following question (max ½ page):
    For your replicated service, Kubernetes has to make decisions which service to use for which request. Based on observations only*, try to deduce Kubernetes' algorithm for load balancing, and provide a few examples that prove or aid your deductions.*
    c. Anything you did for the bonus part of assignment 3.1. While there is no page limit to this part, try to keep it succinct to shorten grading times.

In the following section, there will also be a few generic points you will be graded on for this assignment. Be sure to read those carefully as well!

Tips

1. There is every chance that some part of your cluster deployment will not work; it is a complicated system, after all. If so, please try to use the

internet to find a solution yourself. Typical errors relate to using a wrong/forgetting to apply the network plugin[10].

Common parts:

In addition to parts 3.1 and 3.2, there is also a little part that is graded in common.

Specifically, your grade will depend a little bit on the quality of your presentation style and report quality, like in previous assignments:

1. For your presentation style, this is mostly based on whether you have made efforts to make the presentation go smoothly and whether you stay within time (10 minutes). Also note that we may deduct points if members of your groups are missing on presentation day (unless you have a good reason and let us know beforehand).
2. For the language in your report, this is mostly based on whether what you wrote is easily understandable and relevant. We might also deduct a few points if your language is too colloquial (it is a scientific environment, after all).

Additionally, don't forget to add a small table to your report that details who did what. What you write here won't influence your grade, but we will look at this if you encounter problems with cooperation in your group. Please let us know if you are unhappy about your group's teamwork!

Submission
Your work should be submitted in Canvas, under Assignment 3. Specifically, bundle your code and your report (as PDF) in a tarball or zipfile called *<group number>_web_service_3.tar.gz* or *<group number>_web_service_3.zip*, respectively. Concretely, your archive must contain the following:

1. Your working**(!!!)** source files
2. A *README.md* that explains how to get your code up-and-running. Specifically, tell us the commands to use to deploy your service. **A**lso explain which files are used for assignment 3.1, and which for 3.2 (or which files are used for both)

In addition to your archive, also please submit your report **separately** (i.e., submit two files). This way, we can use Canvas' built-in tools to annotate your report for easier feedback.

Note that, before you submit, you should have already joined a group in the People tab, or else we won't grade your submission. If you did, however, then only one member in your group has to submit, and Canvas will count it as a submission for everyone.

---

[10] https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/network-plugins/

Note: Please do not just change groups after you submitted assignment 2! Canvas is actually really annoying in this, so please prevent chaos and let us know if you need to do so for whatever reason before you actually do so.

Important Notes:

- TAs, and the rest of the team, are happy to help you! However, they will only do so **during working hours**.
- TAs have extensive experience in Python, but they can only provide **best-effort support** if you decide to use another language than Python, or another library than listed here.
- Using code written by someone else is not a sin; however, **you must provide a source in your README.md / comments!!!** You are also expected to provide a detailled explanation of how the copied code works, to show that you understand it. **Failure to do so will be considered plagiarism!**
    - Don't hesitate to ask a TA for advice if you are unsure about this or a particular piece of code.
- Code written by ChatGPT (or similar AI assistants) does not count as your own work. And even if you provide a source, you will not get a passing grade using these tools.