Assignment 11

Due: Wed 29.01.2020; 18:00h (1 Week)

Goals

This assignment trains you in...

- Docker
- Kubernetes

Task 1: Dockerfile Difficulty: Easy

In the past tutorials, you programmed a lots of web applications, e.g. MIMUC Twitch. It is the time to deploy your application.

Your tasks are:

- Write a Dockerfile to encapsulate the MIMUC Twitch, and letting your application running inside a Docker container.
- Build Docker image for the MIMUC Twitch, specify the image name as mimuc-twitch, and tag the version as 0.0.1
- Run the application using `docker run`

Hint: Use Dockerfile reference https://docs.docker.com/engine/reference/builder/.

Task 2: Kubernetes Difficulty: Easy

You have built the MIMUC Twitch image. Now it is the time to orchestrating it in Kubernetes. Your tasks are:

- Write a Kubernetes deployment configuration file, to allow the MIMUC Twitch run with 5 replications
- Add a load balancer to the MIMUC Twitch pods

Hint: Use Kubernetes reference to learn how to write a deployment configuration file: https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#writing-a-deployment-spec

Task 3: Turning “Stateful” to “Stateless” for Scalable Application

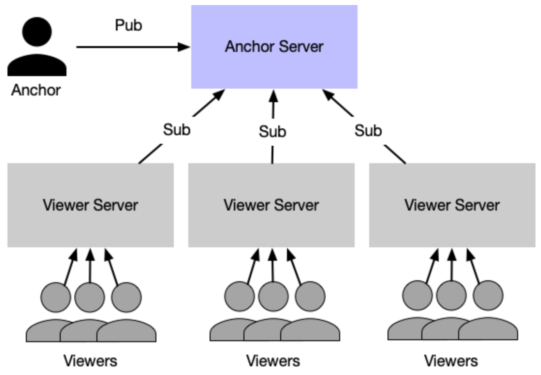
Difficulty: Very Hard

If you access the anchor page and starts broadcasting, then access the viewer page multiple times. Then you will observe sometimes the stream may not working. This is because the load balancer forwards anchor and viewer on to different pods. In this case, our MIMUC Twitch is a monolithic “stateful” application.

To fix the issue and turning our application to a “stateless” application, your tasks are:

1. Separate the MIMUC Twitch server to two different servers: Anchor server and viewer server. In the anchor server, it opens an API to allow anchors to stream and send their data to the serve; In a viewer server, it subscribes to the anchor server and receive the stream data from anchor server then replicates the data and send out to more viewers.
2. Separate the Dockerfile into two different Dockerfiles, one for anchor app and one for viewer app, then build corresponding two Docker images.
3. Rewrite your Kubernetes Deployment config file into two different pods config files: an anchor pod and a viewer pod. The anchor pod has 1 replica and viewer pod has 3 replicas.
4. Write two separate Kubernetes Load Balancer Service files that one for anchor pod and one for viewer pods.

With the above tasks, the viewer pods are stateless and can be load balanced properly. The application architecture should look like the following figure:



Think and answer the following question about the architecture design:

1. Does the anchor pod a “stateless” application?
2. Why an anchor server does not need to replicate but viewer server replicates so many?

Hint: Think about the number of anchors and the number of viewers and server’s pressure.

Submission

Please turn in your solution as ZIP file via Uni2Work. You can form groups of up to three people.

We encourage you to sign up for Slack! All you need is a CIP account and an email address that ends in “@cip.ifi.lmu.de”. Ask us if you don’t know how to get them.

If you have questions or comments before the submission, please contact one of the tutors. They are on Slack @Aleksa and @Andre, remember that they also want to enjoy their weekends ☺

It also makes sense to ask the question in our #omm-ws1920 channel. Maybe fellow students can help or benefit from the answers, too!