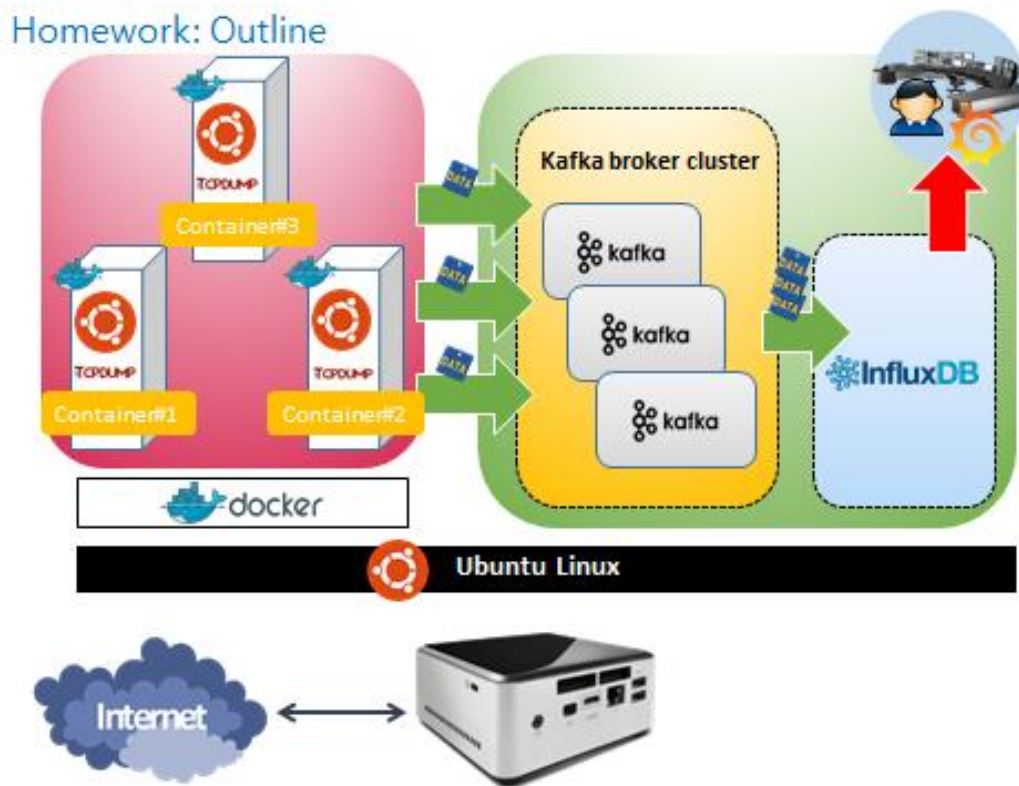


CSTL 2020 Final Project

Monitoring pings and Visualization



We are going to use InfluxDB as our database to save network packet information retrieved using tcpdump from 3 containers.

A packet is a group of bits that includes data plus control information.

Every data being transmitted through your computer's networks are packed in a network packet.

You will retrieve network packet information from the containers and save them on the database at your NUC.

You will use kafka to transmit parsed packet information to the NUC from your containers.

The network packets you are going to parse are ICMP packets.

They are the packets that are used for ping commands.

These are the following things you need to include in your report in this section:

Your ultimate goal is to check network connectivity by using ping commands

Keep in mind that it is possible you will need to install additional softwares.

Instructions to install every single software or tools that are necessary to complete this lab. is not presented and therefore, you will have to figure out how to install them if necessary.

It is recommended that you google them if you need them.

The materials you need to include in your report are highlighted yellow.

1. Check IP address (In the NUC and every container)

(Recommendation) Container image - Ubuntu 18.04.

Take a screenshot of the command you used to check the IP address and the output of the command. Include it in your report.

2. Install InfluxDB (In the NUC)

refer to: <https://computingforgeeks.com/install-influxdb-on-ubuntu-18-04-and-debian-9/>

```
$ sudo curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install influxdb
```

```
$ sudo systemctl enable --now influxdb
```

- check service status:

\$ systemctl status influxdb

Add the screenshot of the output of the command above in your report like the screenshot below.

```
● influxdb.service - InfluxDB is an open-source, distributed, time series database
   Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-05-11 05:42:02 UTC; 3 weeks 2 days ago
     Docs: https://docs.influxdata.com/influxdb/
    Main PID: 2032 (influxd)
      Tasks: 25 (limit: 4915)
    CGroup: /system.slice/influxdb.service
            └─2032 /usr/bin/influxd -config /etc/influxdb/influxdb.conf

Jun 04 00:09:51 bpf2 influxd[2032]: ts=2020-06-04T00:09:51.021961Z lvl=info msg="Snapshot for path written" log_id=0MhZvAv0000 engine=tsm
Jun 04 00:09:51 bpf2 influxd[2032]: ts=2020-06-04T00:09:51.022009Z lvl=info msg="Cache snapshot (end)" log_id=0MhZvAv0000 engine=tsml tra
Jun 04 00:12:06 bpf2 influxd[2032]: ts=2020-06-04T00:12:06.814875Z lvl=info msg="Retention policy deletion check (start)" log_id=0MhZvAv0
Jun 04 00:12:06 bpf2 influxd[2032]: ts=2020-06-04T00:12:06.816201Z lvl=info msg="Deleted shard group" log_id=0MhZvAv0000 service=retentio
Jun 04 00:12:06 bpf2 influxd[2032]: ts=2020-06-04T00:12:06.817681Z lvl=info msg="Deleted shard" log_id=0MhZvAv0000 service=retention trac
Jun 04 00:12:06 bpf2 influxd[2032]: ts=2020-06-04T00:12:06.818893Z lvl=info msg="Retention policy deletion check (end)" log_id=0MhZvAv000
Jun 04 00:42:06 bpf2 influxd[2032]: ts=2020-06-04T00:42:06.814897Z lvl=info msg="Retention policy deletion check (start)" log_id=0MhZvAv0
Jun 04 00:42:06 bpf2 influxd[2032]: ts=2020-06-04T00:42:06.815121Z lvl=info msg="Retention policy deletion check (end)" log_id=0MhZvAv000
Jun 04 01:12:06 bpf2 influxd[2032]: ts=2020-06-04T01:12:06.814822Z lvl=info msg="Retention policy deletion check (start)" log_id=0MhZvAv0
Jun 04 01:12:06 bpf2 influxd[2032]: ts=2020-06-04T01:12:06.815018Z lvl=info msg="Retention policy deletion check (end)" log_id=0MhZvAv000
lines 1-19/19 (END)
```

3. tcpdump (In all the containers)

refer to: <https://www.tcpdump.org/manpages/tcpdump.1.html>

tcpdump is a tool that allows you to monitor network packets that pass by a network interface.

\$ sudo tcpdump -i {interface_name}

will show you monitor results of a network interface.

It is up to you to figure out which network interface to monitor.

Add the screenshot of the tcpdump output on your ping commands from each container like the screenshot below.

Keep in mind that the src ip(source ip address) and dst ip(destination ip address) should be visible in your report.

You need to ping to each container from NUC.

(NUC) -> (container A),

(NUC) -> (Container B),

(NUC) -> (Container C)

```
11:02:15.612968 IP [redacted] > [redacted]: ICMP echo request, id 7725, seq 1, length 64
11:02:15.612990 IP [redacted] > [redacted]: ICMP echo reply, id 7725, seq 1, length 64
```

4. Initiate kafka zookeeper and kafka broker servers (In NUC)

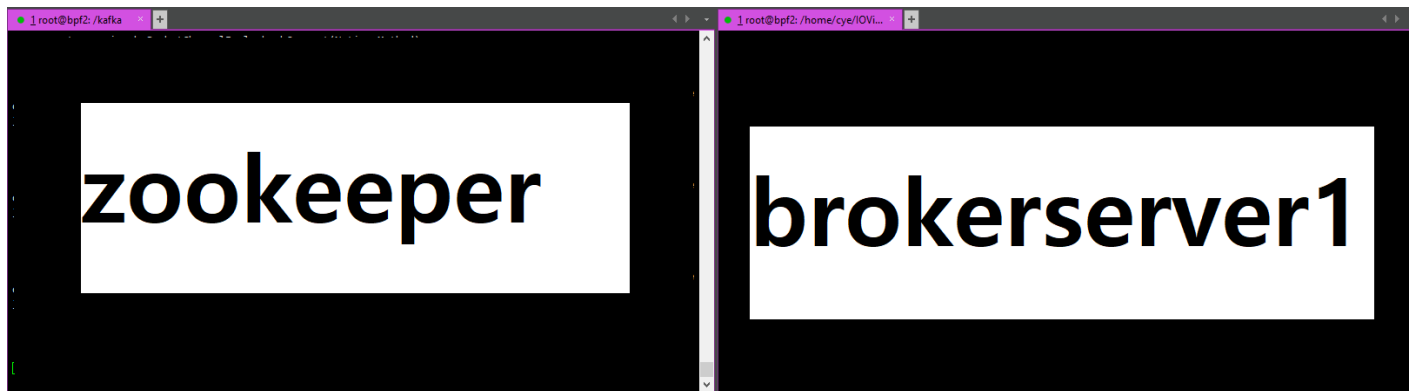
Your task first starts with downloading codes from the github repository we prepared.

https://github.com/gist-banana/cstl_2020

From the repository, you will find a directory, Kafka.

Your first task is to set up and initiate kafka zookeeper with a broker server.

Also, you need to add a screenshot of the zookeeper console, broker 1 console.



It does not matter which console program you use. As long as the zookeeper and a brokerserver are visible, it is okay.

5. Download kafka producer and kafka consumer (In all the containers and the NUC)

Download the repository again in your containers.

https://github.com/gist-banana/cstl_2020

It is necessary that you download this repository in every container and the NUC.

Add the screenshot of the github repository you downloaded.

You should move into the directory that you downloaded and output every content in the repository like below.

```
cye@bpf2:~/cstl_2020$ ls
consumer.py  producer.py  README.md
cye@bpf2:~/cstl_2020$
```

6. Edit the kafka producers (In all the containers)

You need to edit kafka consumer code inside the downloaded github repository.

You can edit interface name and bootstrap servers inside the code.

For kafka consumer and kafka producers, you may refer to: <https://pypi.org/project/kafka-python/>

If you input the IP address of the ping sender in the variable IP_ADDRESS, you will be able to monitor and send only ICMP packets via kafka.

After you edit the code, you need to copy and paste the entire code in your report.

7. Edit the kafka consumer (In the NUC)

You need to input bootstrap servers, topicName and your student name in the upper section of the code.

In the section below, you need to edit the code so you can save destination IP address, source IP address, and ICMP values (either request or reply).

Don't forget to put your name in the STUDENT_NAME variable.

Data saved in InfluxDB should be saved like the screenshot above.

After you edit the code, you need to copy and paste the entire code in your report.

8. Send ping to containers (In the NUC and containers)

We need three containers.

Let's refer to them as container A, container B, and container C.

First, activate kafka producers from all of your containers.

Second, activate the kafka consumer from your NUC.

Third, send ping following the order below:

1. container A -> container B: 10 pings
2. container C -> container B: 20 pings
3. NUC -> container C: 40 pings

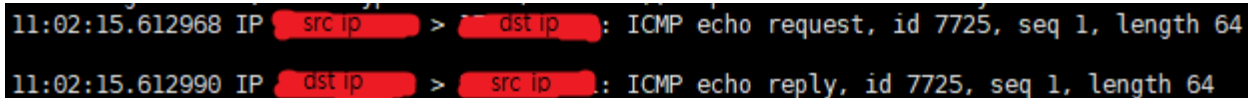
Keep in mind that you can control the number of times ping is sent by:

```
$ sudo ping {destination_IP} -c {number_of_pings}
```

For example, if I want to send ping 10 times to dst_IP:

```
# sudo ping {dst_IP} -c 10
```

In your container, if you activate kafka producer you will see:



```
11:02:15.612968 IP [redacted] > [redacted]: ICMP echo request, id 7725, seq 1, length 64
11:02:15.612990 IP [redacted] > [redacted]: ICMP echo reply, id 7725, seq 1, length 64
```

You have to take a screenshot of the printed output above and put them in your report.

You must include 3 screenshots in total.

9. Save Transmitted Data in the Database (In the NUC)

If you have edited your kafka consumer code properly, you will have following value saved in your database:

time	dst_IP	src_IP	student	value
1591175227227888489			choeyoungeun	request
1591175227265213855			choeyoungeun	reply
1591175228216055094			choeyoungeun	request
1591175228222200745			choeyoungeun	reply
1591175229184821142			choeyoungeun	request
1591175229190761717			choeyoungeun	reply
1591175230210398854			choeyoungeun	request
1591175230216378919			choeyoungeun	reply
1591175231230887427			choeyoungeun	request
1591175231237141073			choeyoungeun	reply
1591175232255551343			choeyoungeun	request
1591175232261745385			choeyoungeun	reply
1591175233279710683			choeyoungeun	request
1591175233285809814			choeyoungeun	reply
1591175234305039113			choeyoungeun	request
1591175234311284545			choeyoungeun	reply
1591175235327853835			choeyoungeun	request
1591175235335912339			choeyoungeun	reply
1591175236349952291			choeyoungeun	request
1591175236355921105			choeyoungeun	reply
1591175283751480837			choeyoungeun	request
1591175283761165892			choeyoungeun	reply
1591175284674693380			choeyoungeun	request
1591175284682868529			choeyoungeun	reply
1591175285697997104			choeyoungeun	request
1591175285705428606			choeyoungeun	reply
1591175286721506383			choeyoungeun	request
1591175286730215805			choeyoungeun	reply
1591175287748666447			choeyoungeun	request
1591175287753197428			choeyoungeun	reply
1591175288768565710			choeyoungeun	request
1591175288774486755			choeyoungeun	reply
1591175289792382052			choeyoungeun	request
1591175289798334467			choeyoungeun	reply
1591175290816588753			choeyoungeun	request
1591175290822749333			choeyoungeun	reply
1591175291841457610			choeyoungeun	request
1591175292027035378			choeyoungeun	reply
1591175292863391654			choeyoungeun	request
1591175292870753543			choeyoungeun	reply
1591175293886532672			choeyoungeun	request
1591175293892688912			choeyoungeun	reply
1591175294913312501			choeyoungeun	request
1591175294920699798			choeyoungeun	reply
1591175295936412579			choeyoungeun	request
1591175295943945449			choeyoungeun	reply
1591175296958122693			choeyoungeun	request
1591175296965773303			choeyoungeun	reply
1591175297981772574			choeyoungeun	request
1591175297988487498			choeyoungeun	reply

You can check out the contents in your DB by using the CLI of InfluxDB.

For InfluxDB CLI, you may refer to:

<https://docs.influxdata.com/influxdb/v1.8/introduction/get-started/>

You need to include the screenshot of every content in your DB.

If multiple screenshots are needed to show every content, you may do so.

10. Visualization of InfluxDB using Grafana (In the NUC)

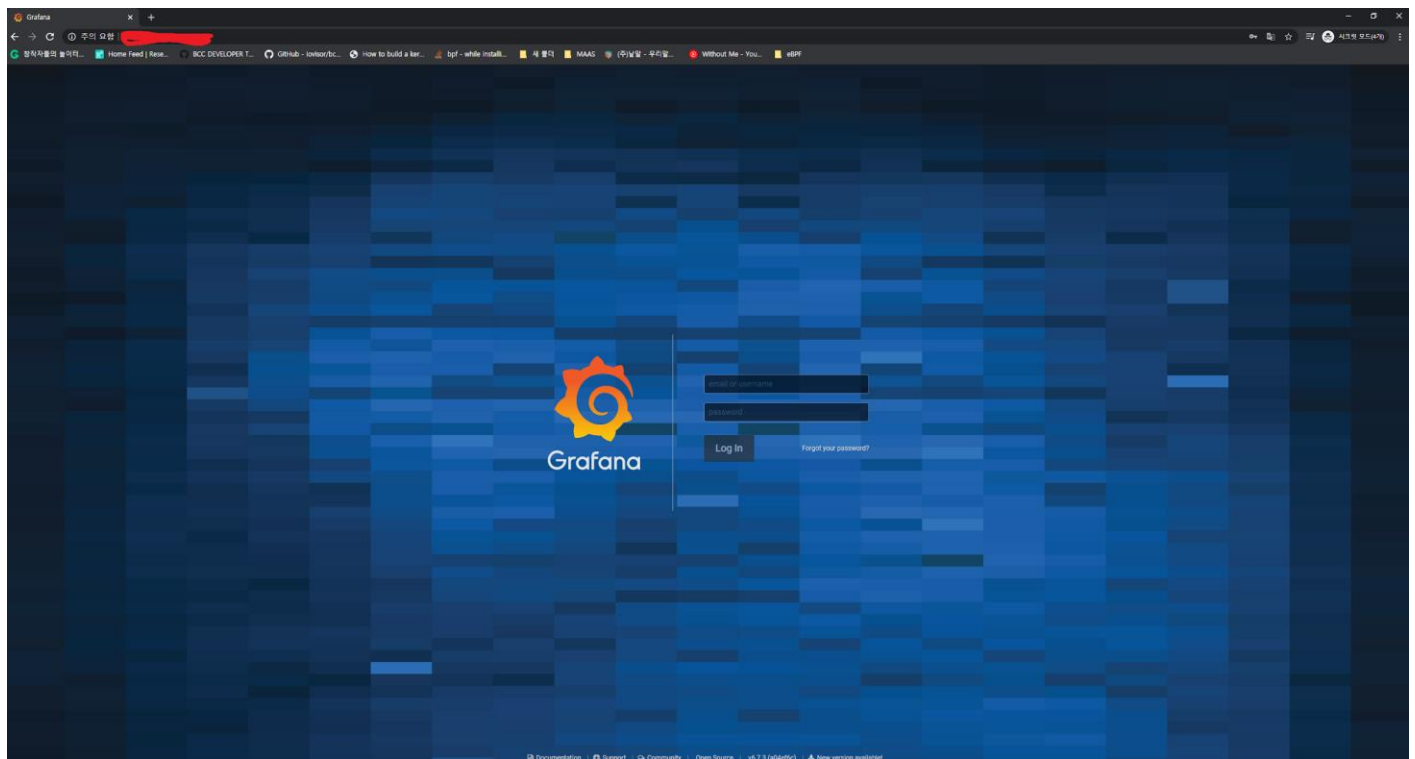
First, you need to install Grafana.

You may refer to: <https://grafana.com/docs/grafana/latest/installation/debian/>

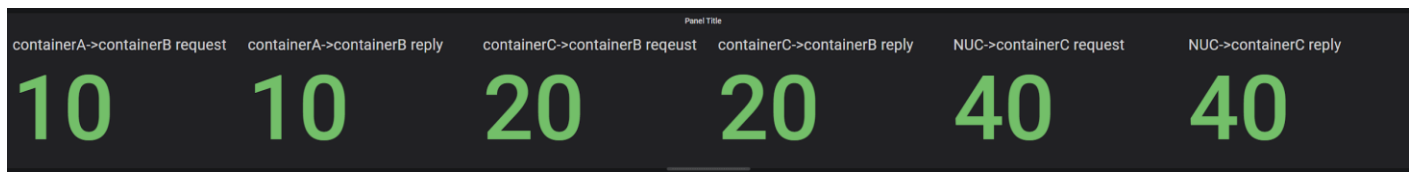
Then, you need to access your Grafana dashboard.

After you successfully access your Grafana dashboard webpage, you need to include a screenshot of the login page like the picture below.

You must take a screenshot in a way that your IP address is visible.



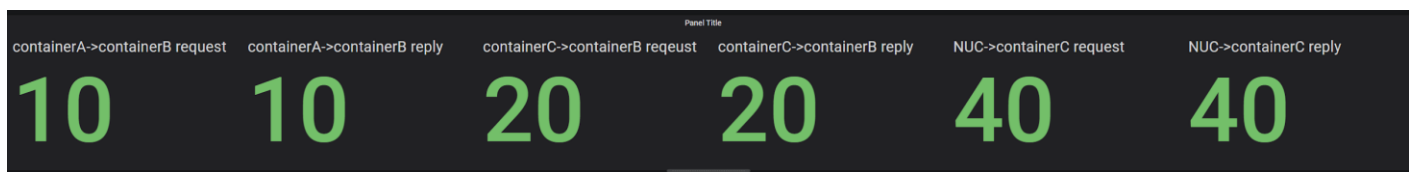
Afterwards, you need to make a dashboard like the picture below.



For the visualization like above, you can choose 'stat' from the visualization menu :



When the whole process is completed, you must include a screenshot of your dashboard like the picture below in your report.



You must also copy and paste the queries you have used for the dashboard.

Or you can just include a screenshot of the queries you used for the dashboard.

(Every single one of them must be included)

11. Build your container of kafka producer via Dockerfile

The container used in step 1 is made into your own container image using Dockerfile.

You may refer to: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

Finally, Dockerfile should contain the command to run the python file to run the producer.

You can write Dockerfile freely as long as the Kafka producer works properly.

You can also change the contents of the existing producer's python file for writing the Dockerfile.

However, when creating a container image through Dockerfile and deploying a container with that image, it must properly communicate with the Kafka broker running on NUC.

Commands for building Dockerfile can be found at the following site.

https://docs.docker.com/engine/reference/commandline/image_build/

The container successfully deployed can be checked as below.

```
ubuntu@cstl-test:~/cstl2020_answer$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
geumseongyoong/kafka-producer	0.9	d9abe5bcc887	2 hours ago	531MB
geumseongyoong/kafka-producer	0.4	40d9fc5b2ba9	7 hours ago	531MB
ubuntu	18.04	c3c304cb4f22	6 weeks ago	64.2MB


```
ubuntu@cstl-test:~/cstl2020_answer$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
d3584f4cd76d	geumseongyoong/kafka-producer:0.9	"/bin/sh -c 'sh host_'"	2 hours ago	Exited (1) 2 hours ago	
cstl5					
7b275e0da512	ubuntu:18.04	"/bin/bash"	14 hours ago	Exited (1) 8 hours ago	
cstltest					

Finally, please submit your Dockerfile via dropbox with your report and python files.

(Option) Upload your container to Docker hub

The image created in step 11 can be uploaded to the Docker hub.


You may refer to: <https://docs.docker.com/docker-hub/>

However, you do not need to use the Docker hub for this homework.


It doesn't matter if you just skip it.


After logging in to the Docker hub and creating a repository, the results are shown below.

[General](#)[Tags](#)[Builds](#)[Timeline](#)[Collaborators](#)[Webhooks](#)[Settings](#)




/ kafka-producer

This repository does not have a description 

 Last pushed: 2 hours ago

Docker commands

To push a new tag to this repository,

```
docker push  /kafka-producer:tagname
```

Tags

This repository contains 5 tag(s).

0.9		 2 hours ago
-----	---	---

If you upload your own image to the Docker hub, you can use it in any environment.

This is useful when deploying Pods from Kubernetes.