南昌大学实验报告

姓名：Qing Liu

学号：6130116184

邮箱地址：1119637652@qq.com

专业班级：Class 164 of Computer Science and Technology

实验日期：SUN. April 28th, 2019

课程名称：Cloud Computing Technology Experiments

实验项目名称

Load Balancing

实验目的

- Understanding the concept of load balancing
- Monitor the utilization status of each VM and each host
- Moving VMs from hot spot to cold ones
- Complete this experiment using at least two computers, the more the better

实验基础

- **Hardware: Lenovo Ideapad 700 - 15ISK**
- **Software: Windows 10 HOME Edition, Vmware Workstation Pro, Ubuntu 16.04 LTS and Docker CE, Nginx**

实验步骤

Hints: My partners are Zhiyu Wang(6130116195) and Zhengbang Zeng(6130116182).

Start mongo-express web service

- Pull the necessary image from docker hub

```
$ docker pull mongo
$ docker pull mongo-express
```

- Create a brige network in docker

```
$ docker network create some-network
```

- Start a container of mongo

```
$ docker run --name some-mongo -d mongo:latest
```

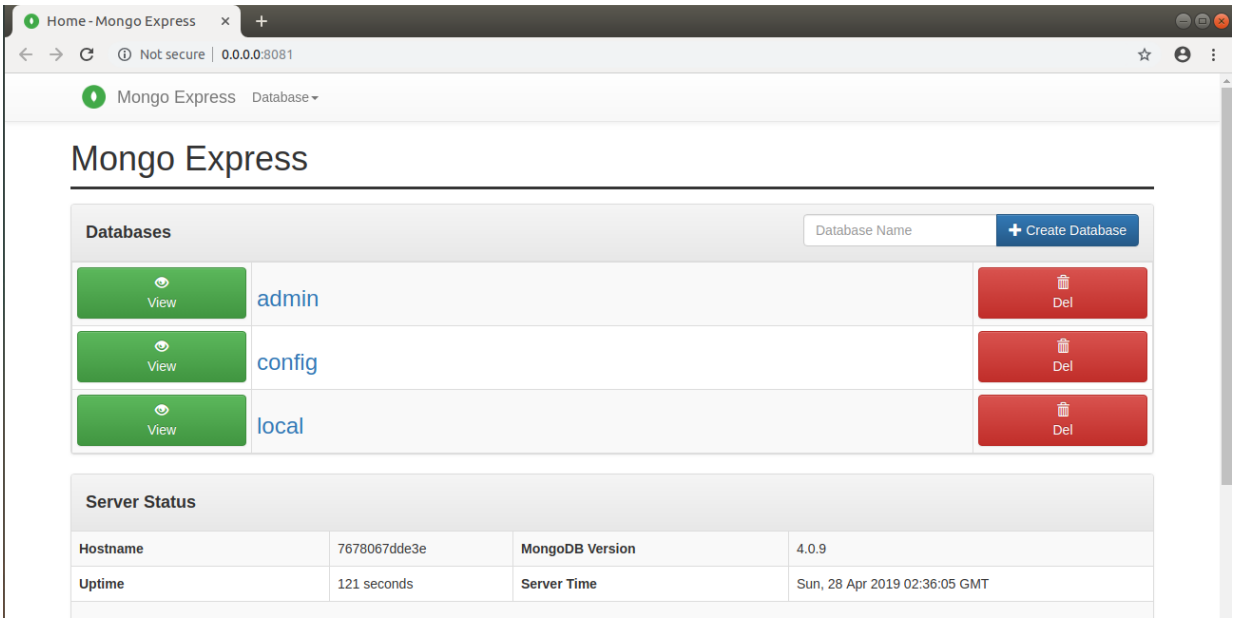
- Connect the mongo container to the network you create

```
$ docker network connect some-network some-monge
```

- Start a container of mongo-express

```
$ docker run --network some-network -e ME_CONFIG_MONGODB_SERVER=some-mongo \
-p 8081:8081 mongo-express
```

- Visit by browser



Test the web service with random data

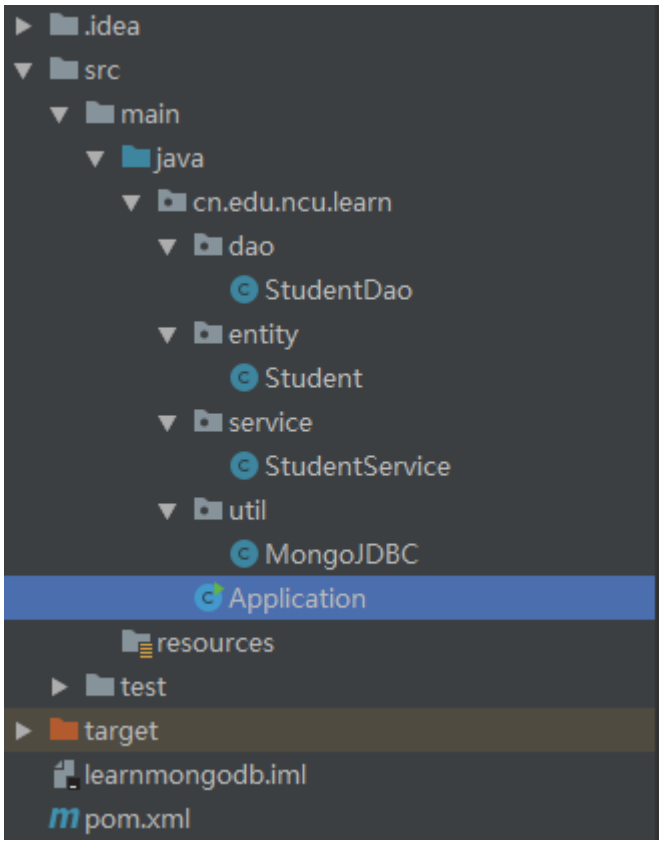
In this part, I wrote an simple application with java(it has already been pushed to github, and will deleted on June 7, 2019) to create a new database `school` and insert 100 documents in collection `student`. Every student document has 4 fields -- id, name, age and gender. Here are the steps:

- Run docker container by image mongo and set the port for vm

```
$ docker run --name some-mongo -p 27017:27017 -d mongo
```

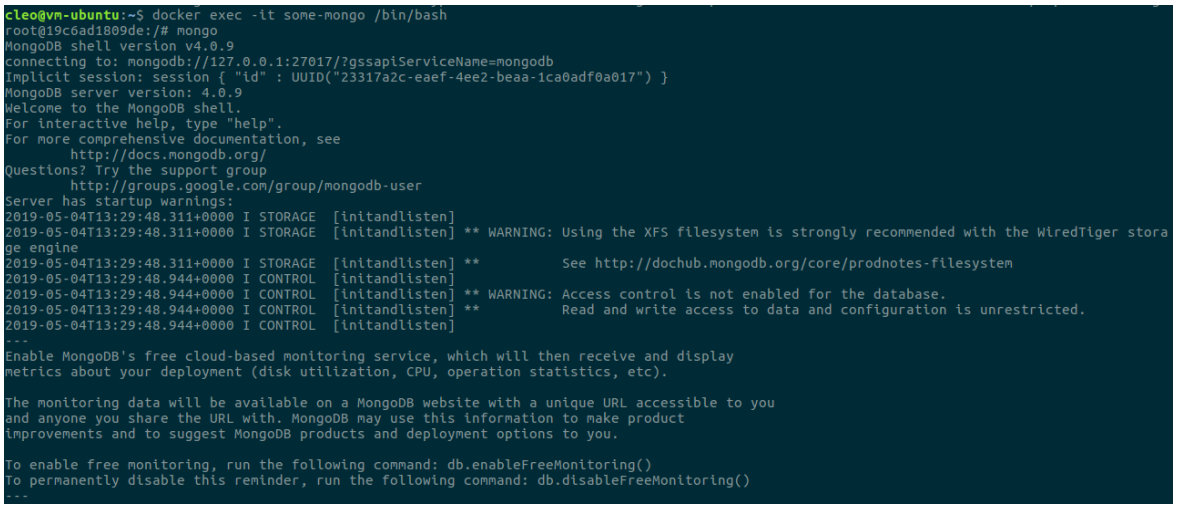
- Run the application

  Here are the structure of the application, as for the code, it was't showed.



- The result in `some-mongo`
  - We can use the bash to operate the MongoDB container.

```
$ docker exec -it some-mongo /bin/bash
$ mongo
```



```
cleo@vm-ubuntu:~$ docker exec -it some-mongo /bin/bash
root@19c6ad1809de:/# mongo
MongoDB shell version v4.0.9
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("23317a2c-eaef-4ee2-beaa-1ca0adf0a017") }
MongoDB server version: 4.0.9
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2019-05-04T13:29:48.311+0000 I STORAGE  [initandlisten]
2019-05-04T13:29:48.311+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger stora
ge engine
2019-05-04T13:29:48.311+0000 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2019-05-04T13:29:48.944+0000 I CONTROL  [initandlisten]
2019-05-04T13:29:48.944+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-05-04T13:29:48.944+0000 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2019-05-04T13:29:48.944+0000 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
```

- Show the databases

```
show dbs
```

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
school  0.000GB
```

- Change to the new database `school`

```
use school
```

```
> use school
switched to db school
```

- Query the documents in collection `student`

```
db.student.find()
```

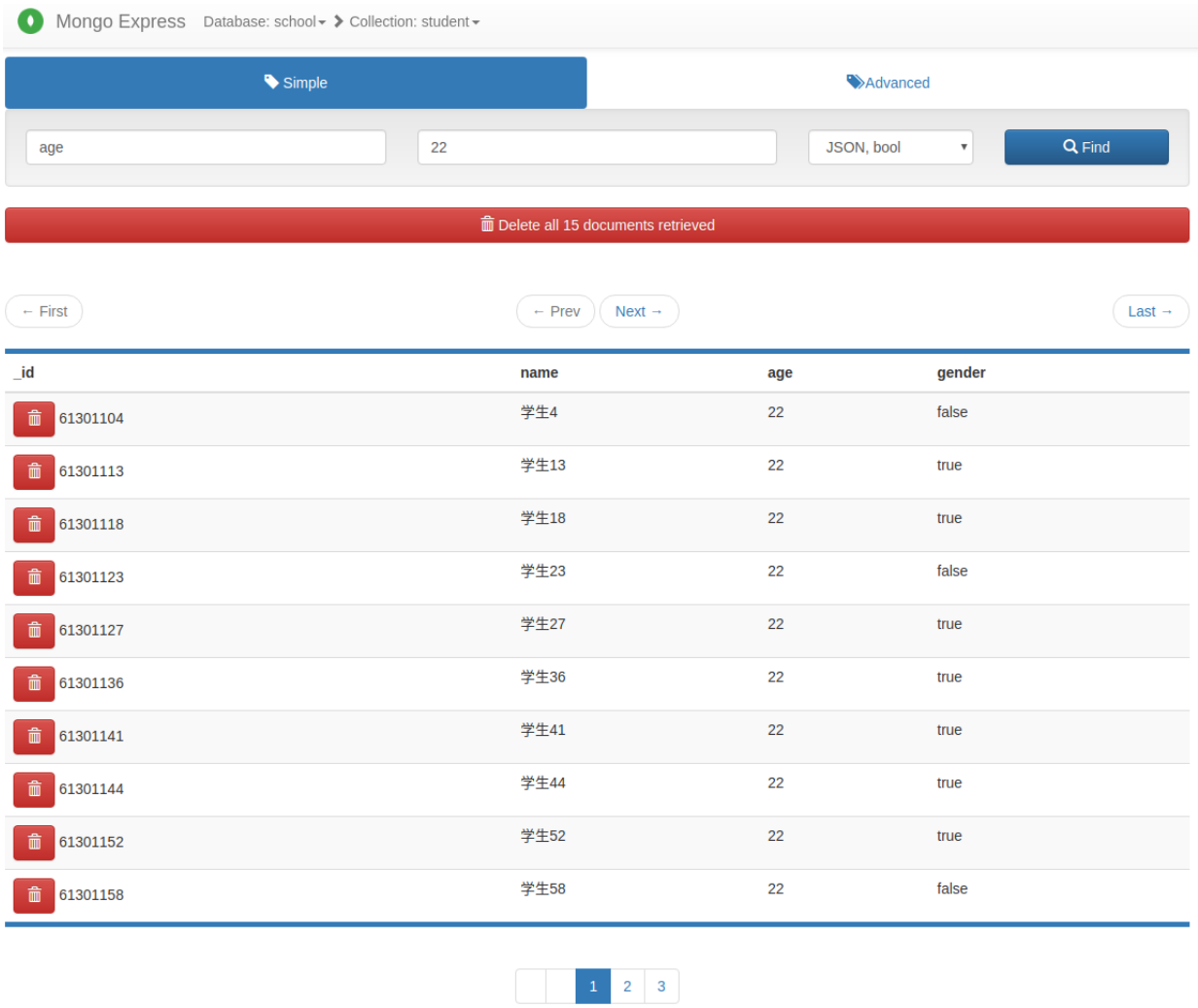
```
> db.student.find()
{ "_id" : 61301101, "name" : "学生1", "age" : 24, "gender" : true }
{ "_id" : 61301102, "name" : "学生2", "age" : 18, "gender" : true }
{ "_id" : 61301103, "name" : "学生3", "age" : 24, "gender" : false }
{ "_id" : 61301104, "name" : "学生4", "age" : 22, "gender" : false }
{ "_id" : 61301105, "name" : "学生5", "age" : 24, "gender" : false }
{ "_id" : 61301106, "name" : "学生6", "age" : 21, "gender" : true }
{ "_id" : 61301107, "name" : "学生7", "age" : 21, "gender" : false }
{ "_id" : 61301108, "name" : "学生8", "age" : 24, "gender" : true }
{ "_id" : 61301109, "name" : "学生9", "age" : 19, "gender" : true }
{ "_id" : 61301110, "name" : "学生10", "age" : 21, "gender" : true }
{ "_id" : 61301111, "name" : "学生11", "age" : 20, "gender" : false }
{ "_id" : 61301112, "name" : "学生12", "age" : 24, "gender" : false }
{ "_id" : 61301113, "name" : "学生13", "age" : 22, "gender" : true }
{ "_id" : 61301114, "name" : "学生14", "age" : 20, "gender" : true }
{ "_id" : 61301115, "name" : "学生15", "age" : 19, "gender" : true }
{ "_id" : 61301116, "name" : "学生16", "age" : 20, "gender" : false }
{ "_id" : 61301117, "name" : "学生17", "age" : 24, "gender" : false }
{ "_id" : 61301118, "name" : "学生18", "age" : 22, "gender" : true }
{ "_id" : 61301119, "name" : "学生19", "age" : 24, "gender" : false }
{ "_id" : 61301120, "name" : "学生20", "age" : 24, "gender" : true }
Type "it" for more
>
```

- Run the `mongo-express` connected to the some-mongo

```
$ docker network connect some-network some-monge
$ docker run --network some-network -e ME_CONFIG_MONGODB_SERVER=some-mongo \
-p 8081:8081 mongo-express
```

```
cleo@vm-ubuntu:~$ docker network connect some-network some-mongo
cleo@vm-ubuntu:~$ docker run --network some-network -e ME_CONFIG_MONGODB_SERVER=some-mongo -p 8081:8081 mongo-express
Waiting for some-mongo:27017...
Welcome to mongo-express
------------------------


Mongo Express server listening at http://0.0.0.0:8081
Server is open to allow connections from anyone (0.0.0.0)
basicAuth credentials are "admin:pass", it is recommended you change this in your config.js!
Database connected
Admin Database connected
```

- Visit `0.0.0.0:8081` with chrome, and select the school database, view the student collection and query with come condition like age=22



We can see that the address in chrome is `0.0.0.0:8081/db/school/student?key=age&value=22&type=J`. And we can check the ip address of the vm is `192.168.123.128` with command `ifconfig`.

Test the mongo service

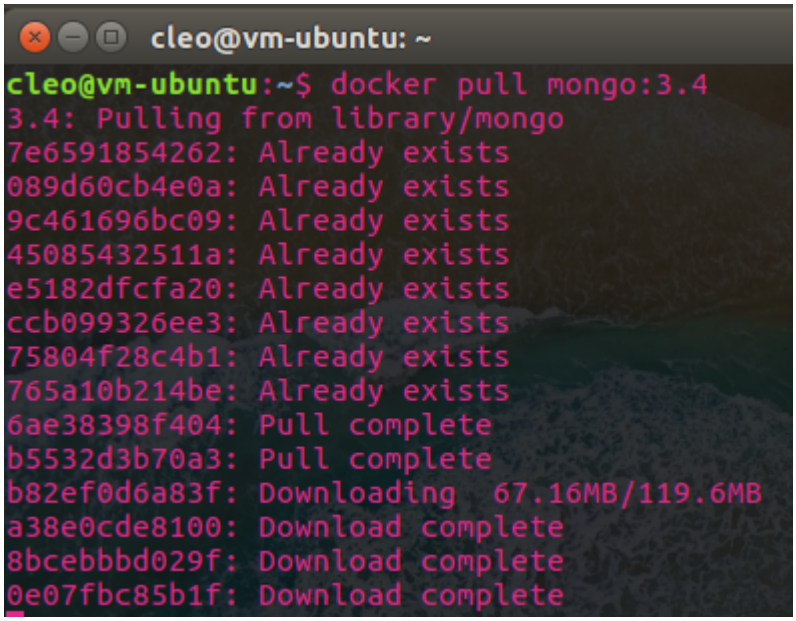
In this part, I opened three virtual machines in my physical machine and the ip address are from `192.168.206.128` to `192.168.206.130`. I run the java application to insert 100 documents of student two virtual machines, which had already run the mongo docker container and mongo-express docker container. The virtual machine whose ip address is `192.168.206.130` run as a nginx server.

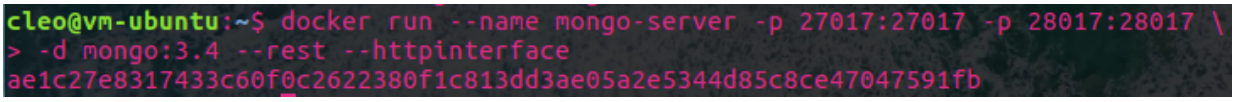
Run two docker mongo server and the version of mongo image is 3.4

The mongo 3.4 has the default restful api, so I send http requests to the api to finish this task.

- Pull the mongo 3.4 image and run it in both mongo server virtual machines

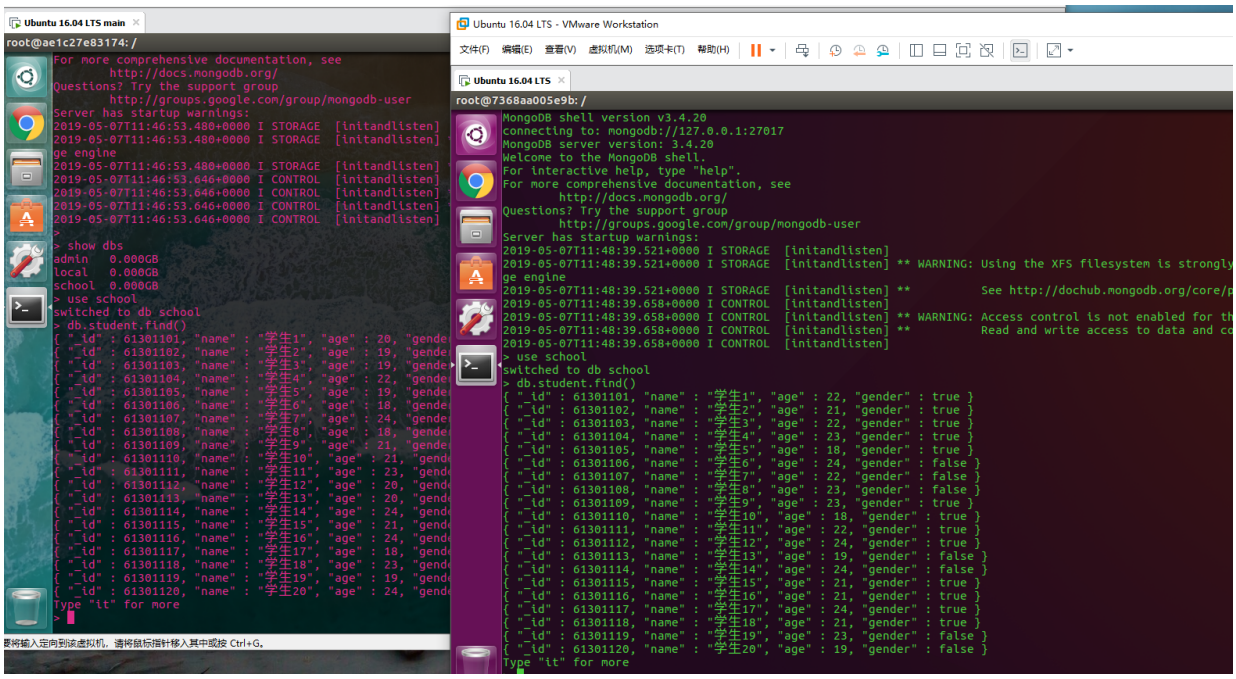
```
$ docker pull mongo:3.4
$ docker run --name mongo-server -p 27017:27017 -p 28017:28017 \
-d mongo:3.4 --rest --httpinterface
```





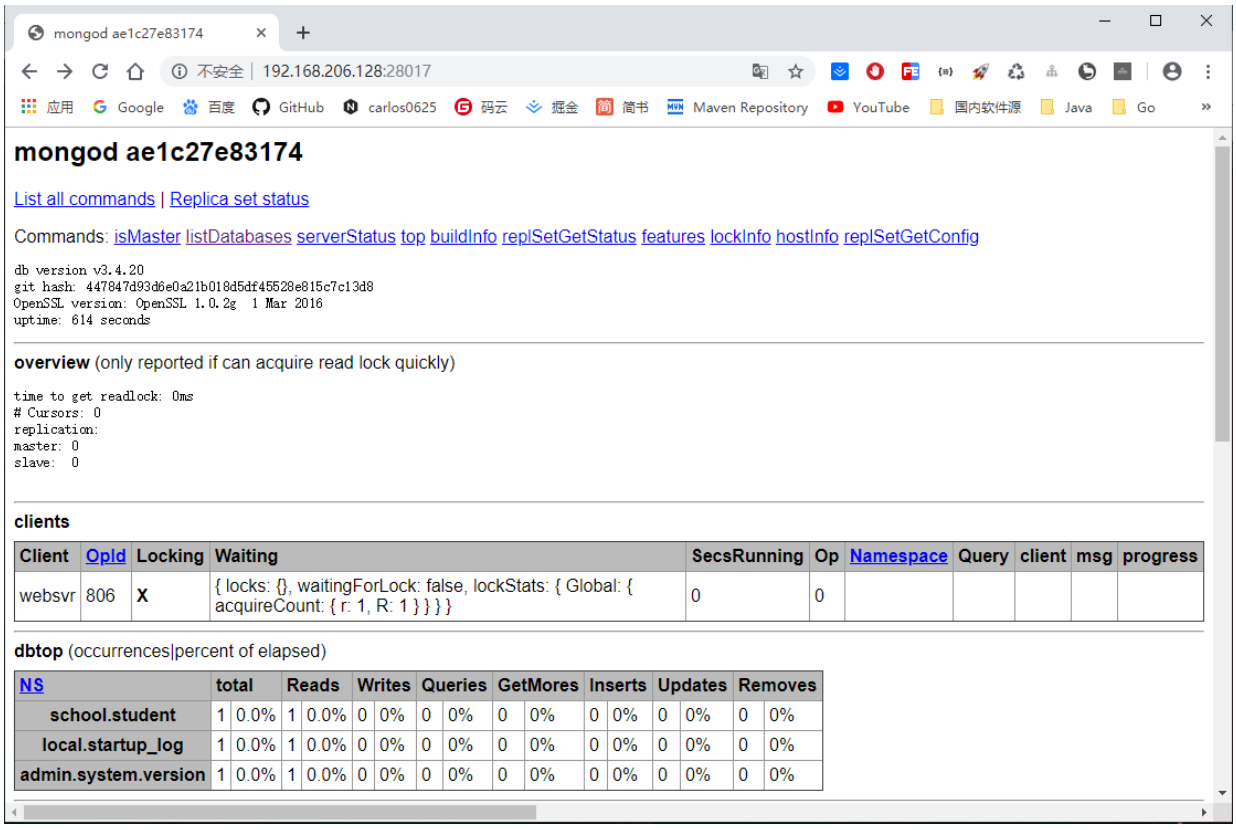
- Run the java application to insert data into two mongo server.

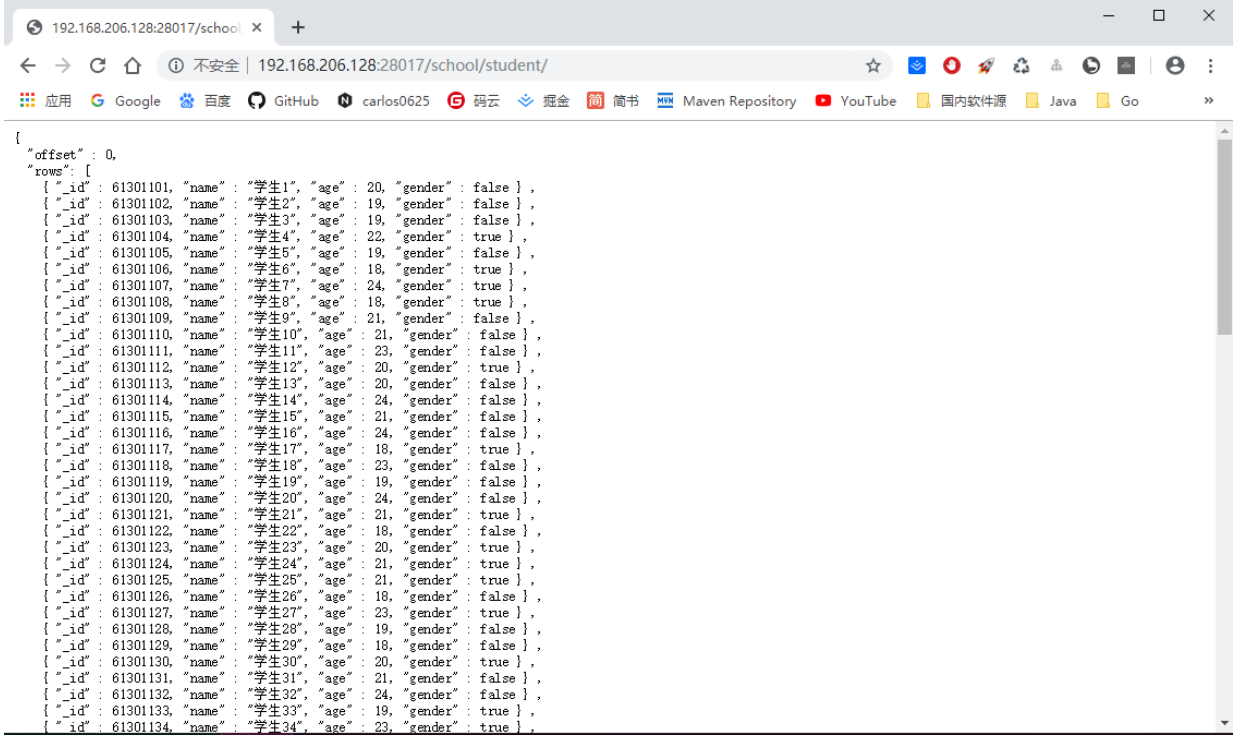
Check the data in mongo db in both of two virtual machines.



If you like, you can run the mongo-express container to connect to mongo-server container.

- Test the default restful api in mongo service





Build Nginx Server

- In the third virtual machine, pull the nginx image from docker hub and run a nginx container. The ip address of this server is `192.168.206.130`.

```
$ docker pull nginx
$ docker run --name some-nginx -p 80:80 -d nginx
```

- Enter the nginx docker container with `bash`

```
$ docker exec -it some-nginx /bin/bash
```

We can check the global configuration for the nginx

```
$ cat /etc/nginx/nginx.conf

user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}
```

```
http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request"
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;
}
```

Copy the content of the file and save in file `nginx.conf` in vm's file `/home/$USER/docker.conf/nginx.conf`, and configure a 1:8 distribution ratio for the previous two Mongo servers.

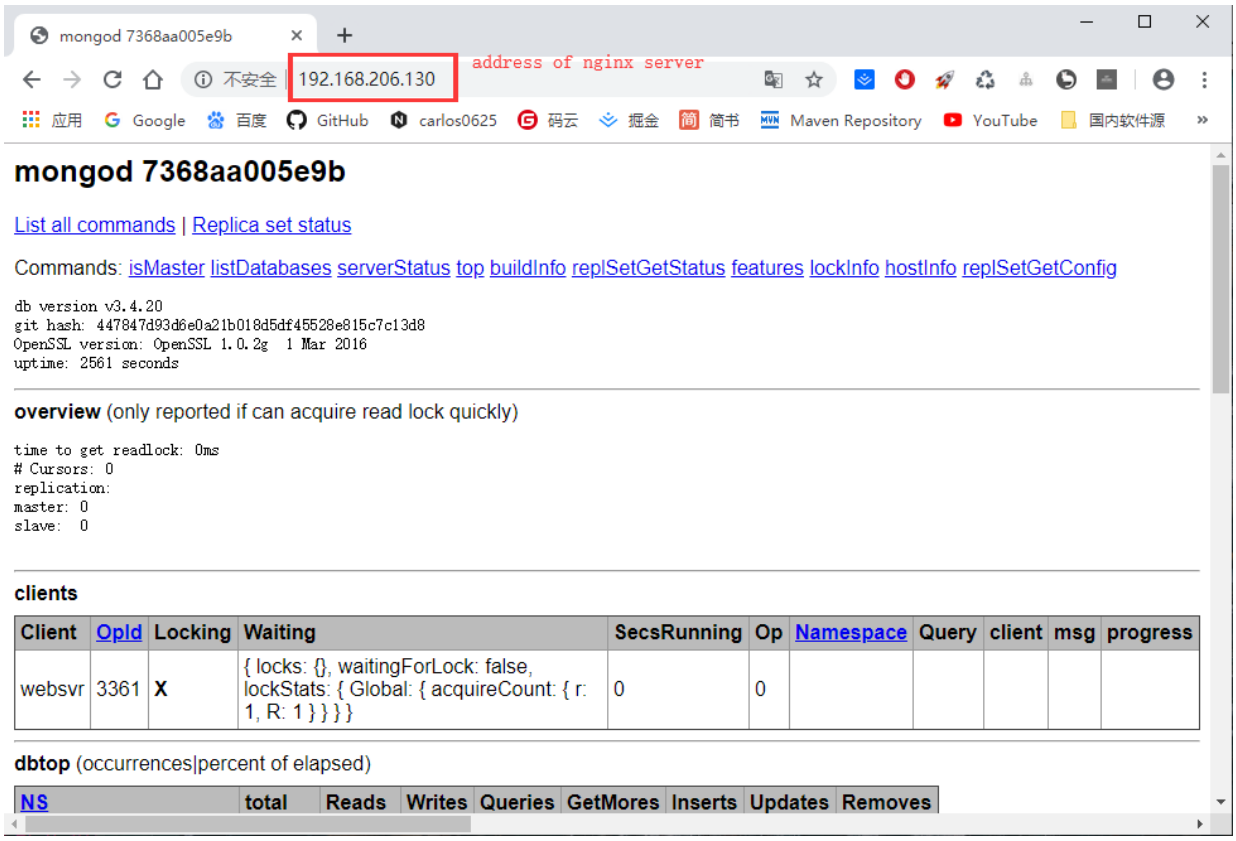
```


user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request"
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;
```

```
    keepalive_timeout  65;

    #gzip  on;
    upstream mongo_servers{
      server 192.168.206.128:28017 weight=1;
      server 192.168.206.129:28017 weight=8;
    }

    server {
      listen 80;
      server_name localhost;

      location / {
        proxy_pass http://mongo_servers;
        #root html;
        #index index.html index.htm;
      }
    }

    include /etc/nginx/conf.d/*.conf;
}
```

- Remove the old docker container of nginx, and start a new one with this command:

```
$ docker container rm some-nginx
$ docker run \
  --name my-nginx \
  -p 80:80 \
  -v /home/$USER/docker.conf/nginx.conf:/etc/nginx/nginx.conf \
  -d nginx
```

- Test for the nginx server. Open the chrome installed on my Windows and input the ip address `192.168.206.130` , it will show like this picture:



It means we build the nginx server successfully. The http request was send to `192.168.206.128:28017` or `192.168.206.129:28017`.

Generate the http request to the nginx server

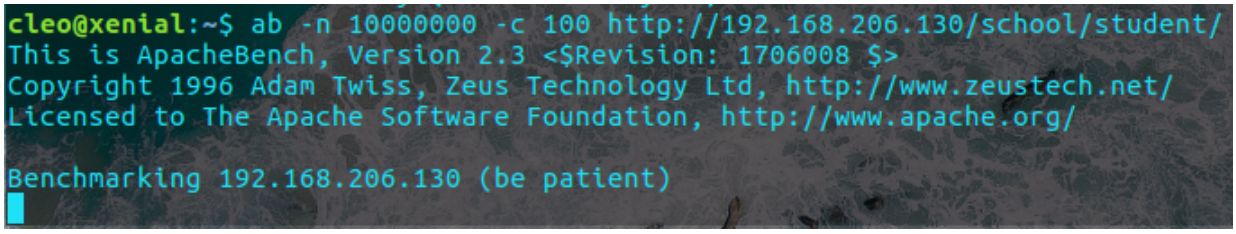
I use the **Apache AB** to send the http requests to nginx server.

- Install Apache utils

```
$ sudo apt-get udpate
$ sudo apt-get install apache2-utils
```

- Use Apache AB to test the api

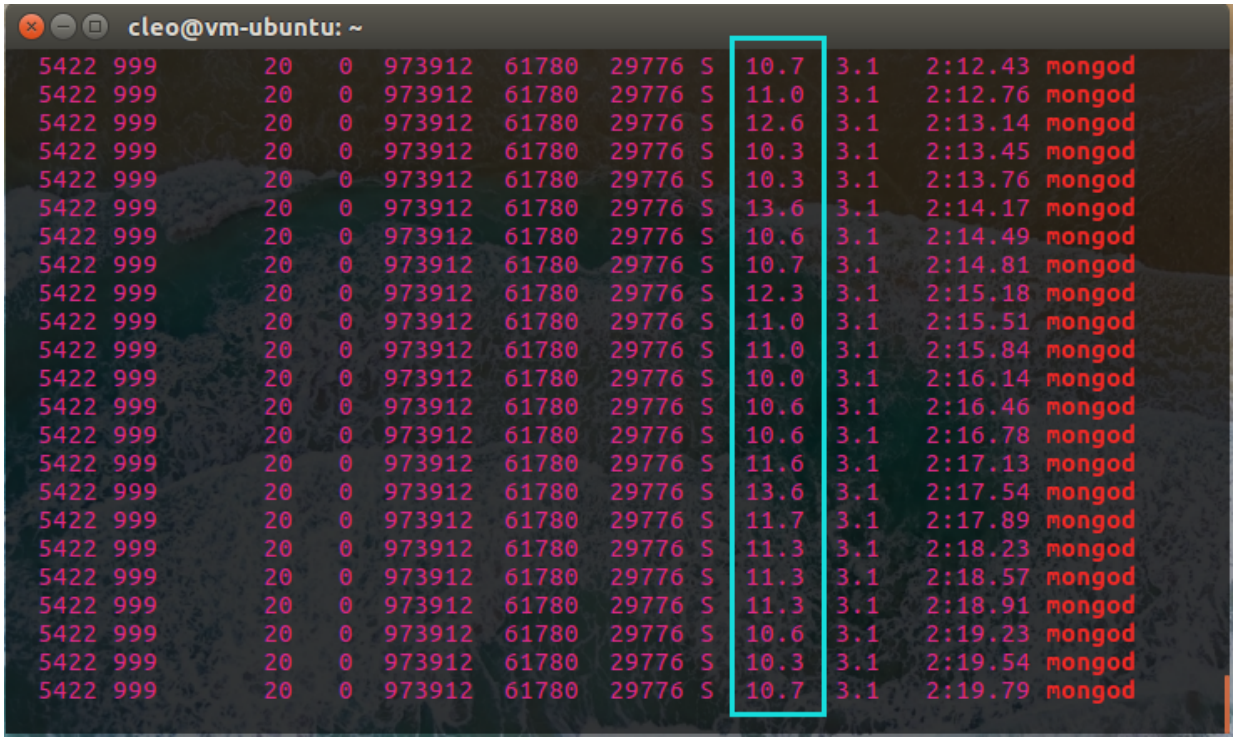
```
$ ab -n 10000000 -c 100 http://192.168.206.130/school/student/
```



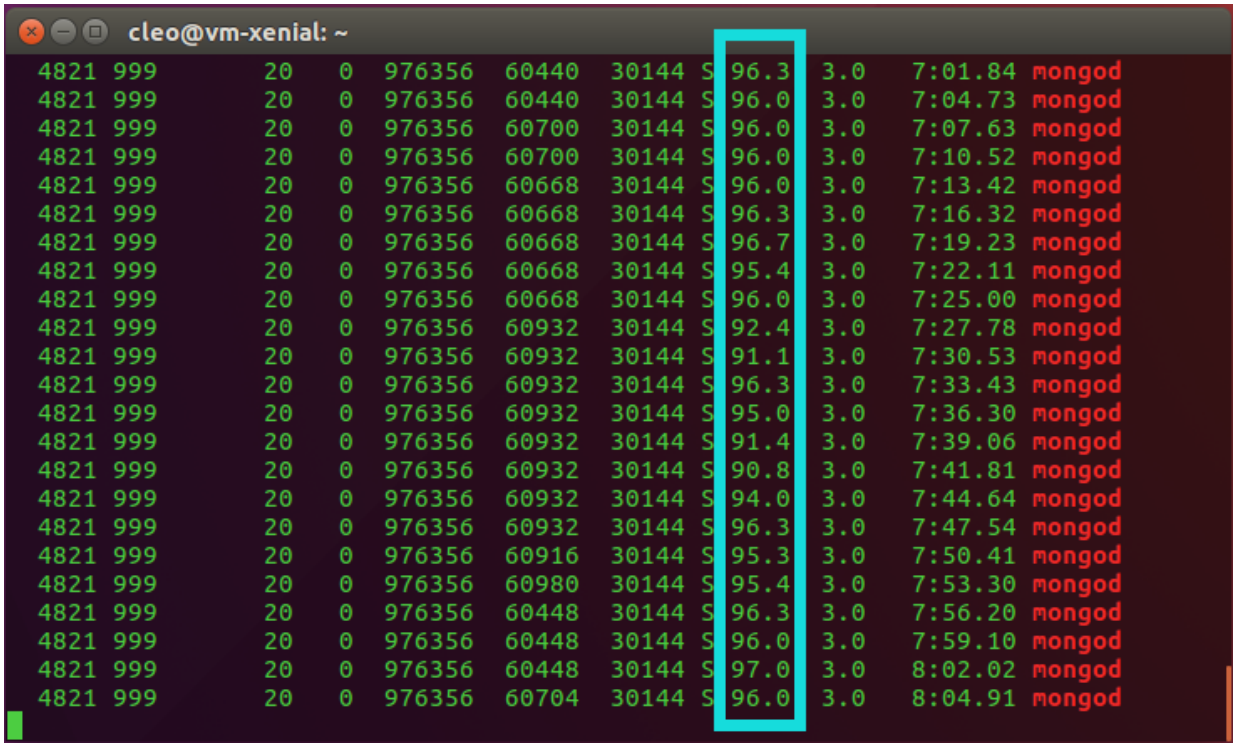
- check the cpu utilization of the mongo service in `192.168.206.128` and `192.168.206.129`

```
$ top | grep mongod
```

`192.168.206.128` :



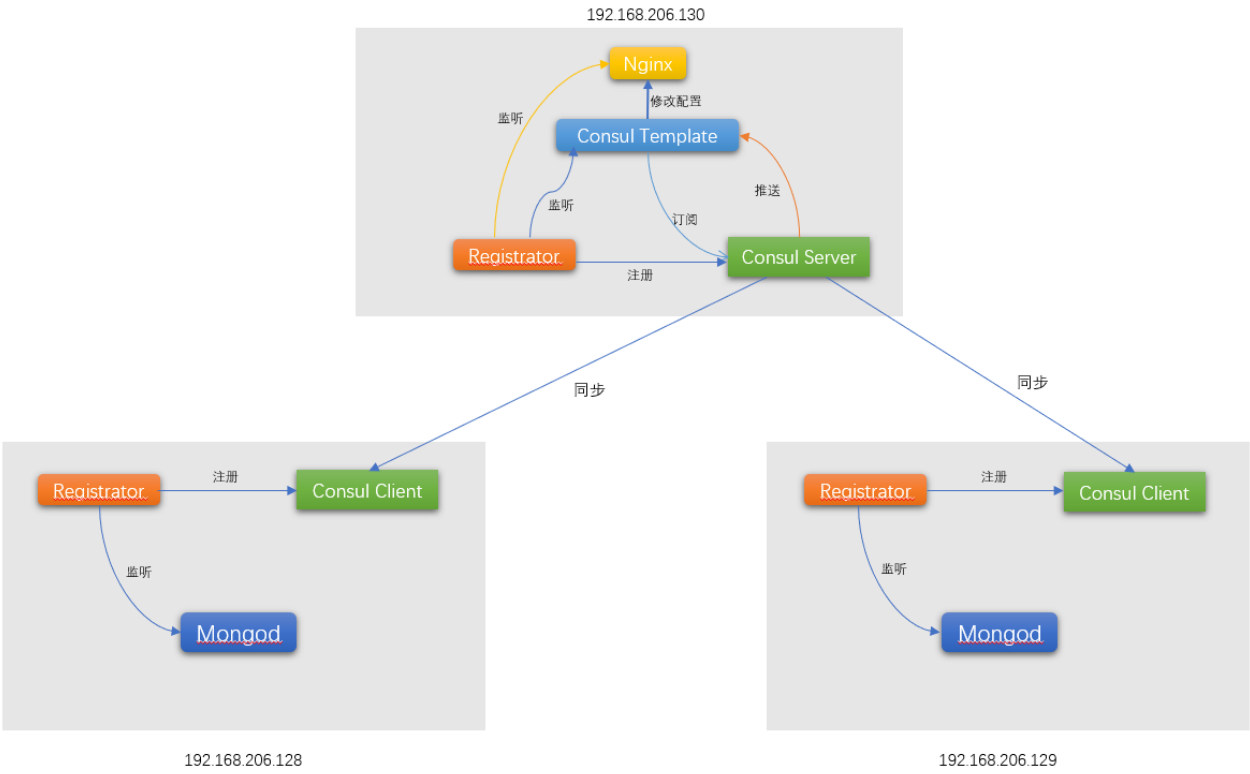
`192.168.206.129` :



The cpu utilization of mongo server 2 is higher than 80%. So, it is a hotspot.

Load balance

I do it by using Consul+Nginx+Consul-Template. This is my architecture of load balance stratigy:



Steps to build the load balance

1. Consul Server(`192.168.206.130`)

- Install docker-compose

```
sudo apt-get install docker-compose
```

- Edit the `docker-compose.yml`

```
version: '2'
services:
  load_balancer:
    image: liberalman/nginx-consul-template:latest
    hostname: lb
    links:
      - consul_server_master:consul
    ports:
      - "80:80"

  consul_server_master:
    image: consul:latest
    hostname: consul_server_master
    ports:
      - "8300:8300"
      - "8301:8301"
```

```
      - "8302:8302"
      - "8400:8400"
      - "8500:8500"
      - "8600:8600"
    command: consul agent -server -bootstrap-expect 1 -advertise 192.168.206

  registrator:
    image: gliderlabs/registrator:latest
    hostname: registrator
    links:
      - consul_server_master:consul
    volumes:
      - "/var/run/docker.sock:/tmp/docker.sock"
    command: -ip 192.168.206.130 consul://192.168.206.130:8500
```

- Run multiple containers application

```
docker-compose up -d
```

- Visit `http://192.168.206.130:8500` to check the node informations and list of service register

2. Consul Client(`192.168.206.128` and `192.168.206.129`)

- Install docker-compose

```
sudo apt-get install docker-compose
```

- Edit docker-compose.yml

```
version '2'
services:
  consul_client_01:
    image: consul:latest
    ports:
      - "8300:8300"
      - "8301:8301"
      - "8301:8301/udp"
      - "8302:8302"
      - "8302:8302/udp"
      - "8400:8400"
      - "8500:8500"
      - "8600:8600"
    command: consul agent -retry-join 192.168.206.130 -advertise 192.168.206

  registrator:
    image: gliderlabs/registrator:latest
    volumes:
```

```
      - "/var/run/docker.sock:/tmp/docker.sock"
    command: -ip 192.168.206.128 consul://192.168.206.128:8500
```

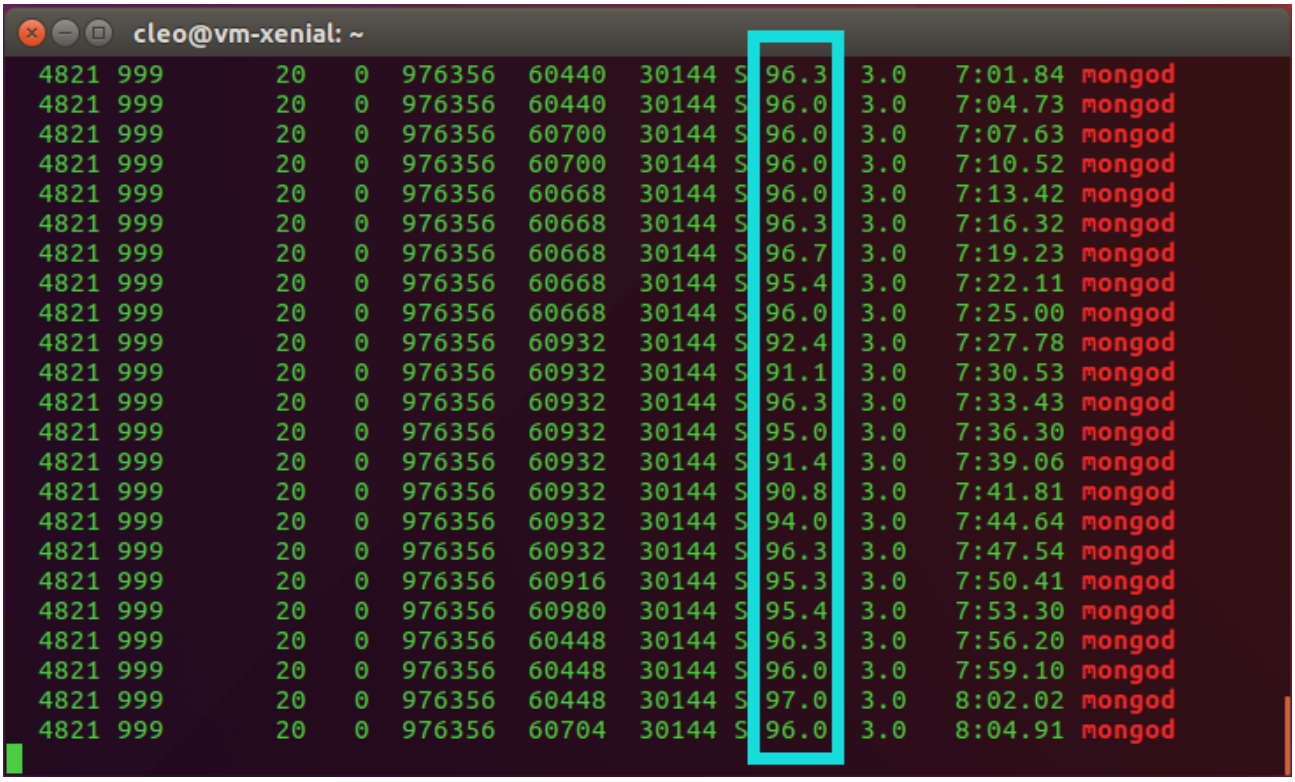
- Define the service and health check script

  This is part that I can not fugure out. So I stoped it. I have already do research for it for a long time.

实验数据或结果

The hotspot

The screen shot of cpu utilization in mongo service in `192.168.206.129`:



The result of load balance

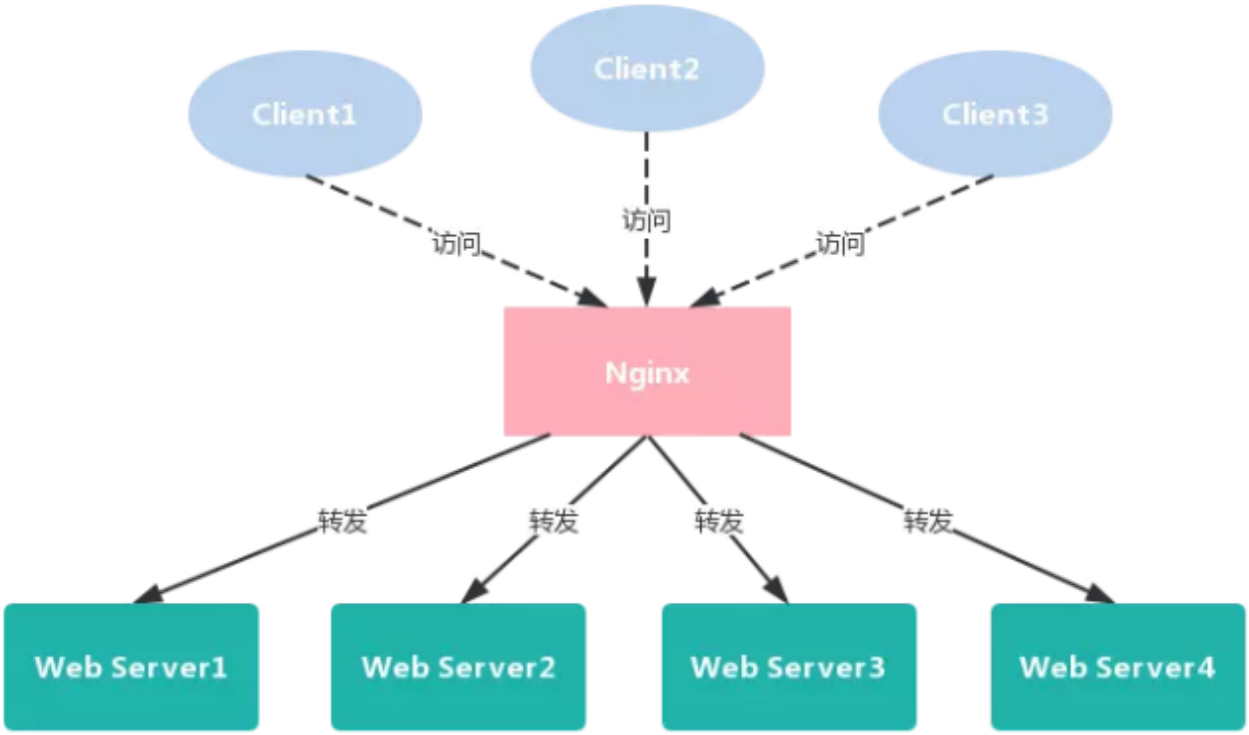
I'm sorry. I cannot figure it out.

实验思考

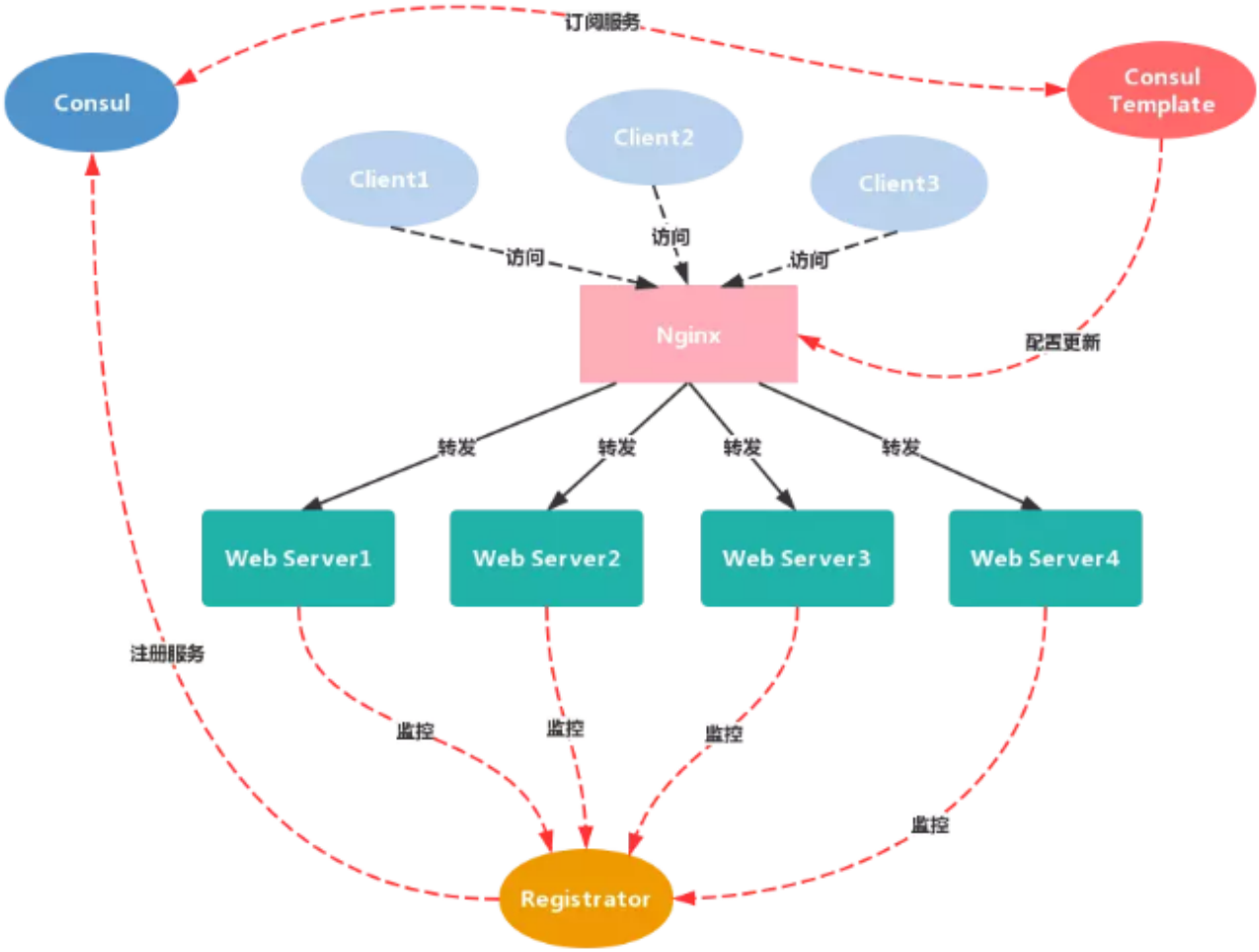
At present, I have four ways to do load balancing. The first, and simplest, is to use nginx for request distribution and shell script to dynamically detect its CPU usage on the Mongo server, closing the container directly when it exceeds 80%. In this way, requests will be sent to other servers without downtime. Second, you don't use off-the-shelf servers like nginx. Write two scripts, one for proxying, distributing requests, and detecting resource detection reports from application servers, and decide which server to send according to the reports. The other is used on the application server to detect CPU usage and report to the distributor. Thrid, use LVS + keepalived + shell. Use shell to collect real server load, CPU Idle to the scheduler regularly. Using the health check method of MISC_CHECK provided by

keepalived, write scripts to check the load, CPU Idle, swap information of each real server. When all of them do not exceed the critical value, return 0 (to join the cluster on behalf of the real server), and return 1 (to remove the cluster on behalf of the real server) when one of the options exceeds the critical value. The fourth is to use Consul + Nginx + Consul-Template.

This is the load balance using Consul+Nginx+Consul-Template in web service architecture. The traditional load balance's architecture looks like this picture:



What we have to do is auto-load balance:



参考资料

- https://www.cnblogs.com/wang-meng/p/5861174.html?tdsourcetag=s_pctim_aiomsg
- https://blog.csdn.net/boling_cavalry/article/details/78168085
- https://hub.docker.com/_/mongo
- https://hub.docker.com/_/mongo-express
- https://docs.docker.com/network/bridge/
- https://www.jianshu.com/p/fa41434d444a