

CryptoLab Instruction

January 29, 2023

1 General Description

CryptoLab is a system where students can practice cryptographic attacks and implement exchange policies. It consists of several machines running containerized (in Docker) apps. One machine that we call a **gateway** acts as a central access point for students. The other machines run **backend** server app (also Docker containerized) that do the actual work. Below is a diagram of the system's topology.

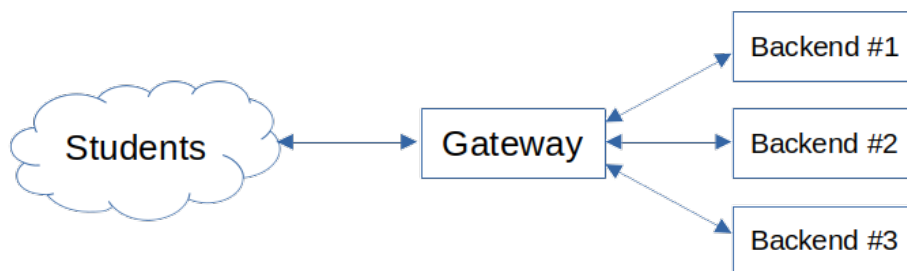


Figure 1: Scheme of the system.

Students have access to the gateway machine alone. Gateway is listening 80 port, and students send their requests using HTTP protocol. Gateway runs a reverse proxy that acts as a load balancer for the backend machines. When the gateway receives a student's request, it redirects the request to one of the backend machines, which process the request and send back some response. This is a normal system's workflow.

2 Requirements

1. We need 3-4 computers running some Linux (preferably Ubuntu or similar) distro with sudo rights.
2. All machines must be accessible by IP address or by some name.
3. One machine (the gateway) should be accessible by the name `cryptolab.csil.sfu.ca`.
4. All machines need Git, Docker, and SSH installed.

3 Installation

First and foremost we must decide what machines are going to act as the gateway and the backends. This one is easy: the gateway has address `cryptolab.csil.sfu.ca`, while the backend machines can have any address. From now on we will refer to the address of the first backend machine by `ADDRESS-1`, `ADDRESS-2` is the address of the second backend machine, `ADDRESS-3` is the address of the third, etc.

Setting up a backend Log in via SSH or by other means into the first backend machine, and follow the steps below.

STEP 1. Download the source code of the backend part of the system by executing this command in terminal

```
$ git clone https://csil-git1.cs.surrey.sfu.ca/oome1che/crypto-backend.git
```

If the above command produces an error that the project could not be found or you're not authorized to clone the project, email the instructor. Next, change the working directory by running

```
$ cd crypto-backend
```

STEP 2. Make `build.sh` and `run.sh` scripts executable by running

```
$ chmod 700 build.sh run.sh stop.sh
```

STEP 3. Build and containerize the backend's app

```
$ ./build.sh
```

This process takes a while. At the end (if everything is ok) you should see a message like this

```
Successfully built ac121ec2650f
Successfully tagged crypto/backend:latest
```

If you did get an error, investigate the error message, and fix the issue. At the time of writing this instruction, the building process went to completion with no problem.

STEP 4. Run the app

```
$ ./run.sh
```

STEP 5. Check that the app is running by calling

```
$ curl http://localhost:8888/heartbeat
```

You should get the output like this

```
{"cpu_count":4,"cpu_usage":12.5,"current-time":"2023-01-29 05:25:09",
"host":"","is-alive":1,"mem_available":1102172160,"mem_total":10368090112,
"mem_usage":89.4,"mem_used":8458248192,"server-id":"298c4",
"unix-epoch":1674969909.3859441}
```

This gives you some information about server the app is running at.

STEP 6. Repeat all the above steps for the other backend machines as well.

Setting up a gateway Log in into gateway machine via SSH or by other means, and follow the steps below.

STEP 1. First, we need to make sure that the gateway “sees” the backend machines. Recall that we use `ADDRESS-1`, `ADDRESS-2`, ... to denote an address of the corresponding backend machine. Run in terminal

```
$ ping <ADDRESS-1>
```

For example, if the first backend machine has address 10.10.10.10, then run

```
$ ping 10.10.10.10
```

and you should get something like

```
PING 10.10.10.10 (10.10.10.10) 56(84) bytes of data.
64 bytes from 10.10.10.10: icmp_seq=1 ttl=119 time=7.52 ms
64 bytes from 10.10.10.10: icmp_seq=2 ttl=119 time=6.63 ms
64 bytes from 10.10.10.10: icmp_seq=3 ttl=119 time=7.04 ms
64 bytes from 10.10.10.10: icmp_seq=4 ttl=119 time=6.96 ms
```

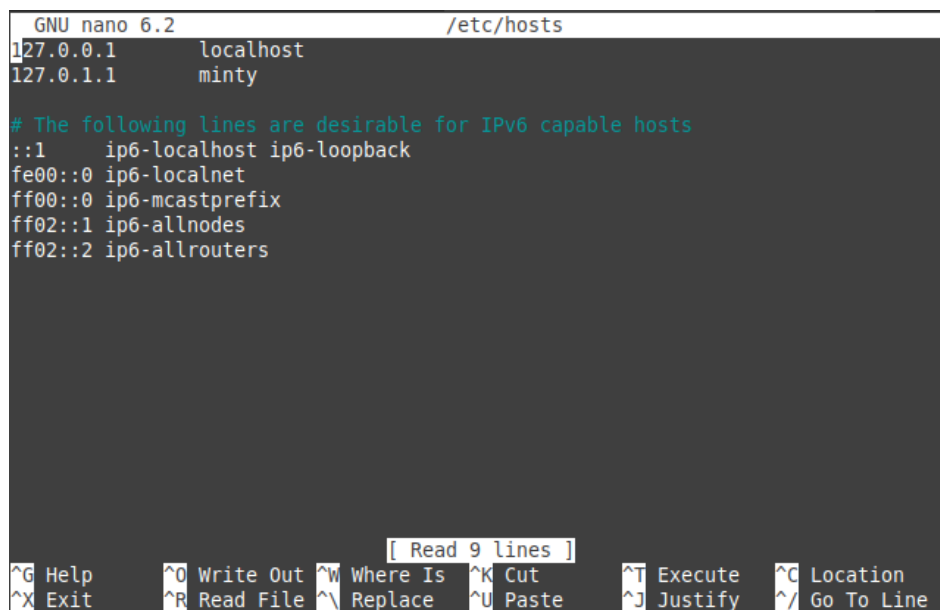
If you do not see similar output, this signals that the gateway does not know how to reach the first backend machine. In that case email IT support to check the issue, since gateway and backend machines are expected to be reachable from each other.

Run the same `ping` test for the other backend machines as well. All backend machines must be reachable from the gateway.

STEP 2. Next, we are going to add aliases for backends. That is open `/etc/hosts` file with `nano` editor by running (see Figure 2 for an example what a `/etc/hosts` looks like in the `nano` editor)

```
$ sudo nano /etc/hosts
```

enter user password, and you will see something similar to



```
GNU nano 6.2 /etc/hosts
127.0.0.1    localhost
127.0.1.1    minty

# The following lines are desirable for IPv6 capable hosts
::1          ip6-localhost ip6-loopback
fe00::0      ip6-localnet
ff00::0      ip6-mcastprefix
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
```

Figure 2: An example of what `/etc/hosts` looks like in `nano` editor.

Move cursor all the way down, and enter the following lines (the number of lines depends on the number of backend machines you have)

```
<ADDRESS-1> crypto-01
<ADDRESS-2> crypto-02
<ADDRESS-3> crypto-03
```

The above lines will assign the first backend machine the name `crypto-01`, the second backend machine will be reachable by the name `crypto-02`, and finally the third backend machine will receive name `crypto-03`. If you have four (or more) backends, you can enter them here as well.

Next, press `Ctrl+X`. You will be asked whether you want to save the buffer. Press `Y` (for Yes), and then hit `Enter` to finally save the file. Verify that the changes were saved by running

```
$ cat /etc/hosts
```

and you should see the newly added lines in the output of the above command.

As an example, suppose we have 3 backend machines with addresses 10.10.10.10, 10.10.10.20, and 10.10.10.30. Then we could assign them the aliases by changing the `/etc/hosts` to look like in the Figure 3.

After saving the file, and running the `cat /etc/hosts` command, you should see the lines similar to the ones below

```
GNU nano 6.2 /etc/hosts *
127.0.0.1 localhost
127.0.1.1 minty

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

# We added the following lines
10.10.10.10 crypto-01
10.10.10.20 crypto-02
10.10.10.30 crypto-03

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^_ Go To Line
```

Figure 3: `/etc/hosts` file after we have added aliases for the backend machines.

```
127.0.0.1 localhost
127.0.1.1 minty

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

# We added the following lines
10.10.10.10 crypto-01
10.10.10.20 crypto-02
10.10.10.30 crypto-03
```

STEP 3. Now we need to verify that the aliases work. Run

```
$ ping crypto-01
```

and check whether the gateway machine is able to send pings to the first backend machine called `crypto-01`. Run the same ping test for the other aliases.

STEP 4. The hardest part is done. Now run the following commands to clone the source code of the gateway into the current folder (if the link does not work, talk to the instructor)

```
$ git clone https://csil-git1.cs.surrey.sfu.ca/oomeiche/crypto-gateway.git
$ cd crypto-gateway
```

Next, we are going to edit gateway's configuration file. Run

```
$ nano nginx.conf
```

to open `nginx.conf` file in the `nano` editor (see Figure 4 for an example). Observe the four lines `server crypto-0*:8888` of directives at the very top of the file. They tell the gateway app where to look for the backend machines.

Comment out unnecessary or add more lines like that if you have more than 4 backend machines, and you created aliases for them in the previous steps. To comment out, simply add `#` a.k.a. the sharp symbol somewhere in front of the line, e.g.

```

GNU nano 6.2                               nginx.conf *
upstream backend {
    server crypto-01:8888;
    server crypto-02:8888;
    server crypto-03:8888;
    server crypto-04:8888;
}

server {
    listen 80;
    include      /etc/nginx/mime.types;
    default_type application/octet-stream;

    sendfile on;
    tcp_nopush on;

    set $upstream http://backend;

    location /myip {
        resolver 127.0.0.11 valid=30s;
        default_type text/plain;
        return 200 "My IP: $remote_addr\n";
    }

    location / {
        ##### Block non-VPN accesses #####
        include conf.d/access.rules;
        #####
        proxy_pass $upstream;
    }
}

```

[^]G Help [^]O Write Out [^]W Where Is [^]K Cut [^]T Execute [^]C Location
[^]X Exit [^]R Read File [^]\ Replace [^]U Paste [^]J Justify [^]/ Go To Line

Figure 4: nginx.conf file.

```

server crypto-01:8888;
server crypto-02:8888;
server crypto-03:8888;
# server crypto-04:8888;

```

comments out the last line. The number and aliases of the backend machines must match the number and aliases that you added into the `/etc/hosts` file in STEP 2. Press `Ctrl+X`, and save the file.

STEP 5. Now build/containerize the project by running

```

$ chmod 700 build.sh run.sh run-it.sh stop.sh
$ ./build.sh

```

You will be asked if you want to stop all running containers. Enter `y` (for Yes), and it will start compiling the gateway app for you. It should not take too long to see the following output at the end

```

Successfully tagged crypto/gateway:latest

```

If the gateway wasn't successfully packed into a container, then you will need to investigate what causes the trouble, since there could be many reasons. At the moment of writing this instruction the process goes smoothly with no issue (tested on several computers!).

STEP 6. Now all is needed is to run the gateway app by executing

```

$ ./run.sh

```

Verify that the gateway app is up and running by running the `curl` command

```
$ curl http://localhost/myip
```

and you should get

```
My IP: 127.0.0.1
```

If you did not get the above output, this signals that the gateway app didn't start up. You can check the problem by running the app in interactive mode by calling

```
$ ./run-it.sh
```

and seeing what error message is produced. Hopefully, this will guide in solving the problem.

STEP 7. Next, we need to run integration test. Connect to SFU VPN. Open your local browser and navigate to <http://cryptolab.csil.sfu.ca/heartbeat>. You should get an output like this ¹

```
{
  "cpu_count": 4,
  "cpu_usage": 26.8,
  "current-time": "2023-01-29T05:25:50",
  "host": "",
  "is-alive": 1,
  "mem_available": 1109323776,
  "mem_total": 10368090112,
  "mem_usage": 89.3,
  "mem_used": 8455544832,
  "server-id": "298c4",
  "unix-epoch": 1674969950.1613266
}
```

Every time you request the above page, gateway sends request to a different backend. The backend's ID is displayed in the `server-id` field. Refresh the page several times, and count how many different `server-id`'s you get. The total number of different `server-id`'s should be the same as the number of backend servers.

Also check the `current-time` field of all the backends. It must be roughly the same every time you refresh the page. If you note that some backend's time is different from the others, fix that.

¹ Different browsers render JSON format in different ways, so you may get somewhat different output, but you should see a lot of information regardless how its rendered.