Overview

This tutorial will take you through the process of downloading the image of our Docker container, launching the image, and then running our client/server application locally. Docker is not compatible with mininet, so if you wish to run mininet simulations, follow the instructions for Building Application from Source.

Prerequisites

Make sure you have a working and updated version of Docker installed on your computer. It is recommended that you use Docker Desktop to view and run our container, the link to install Docker is provided here.

Also, we need to use some sort of terminal in order to load and launch the docker container. On MacOS/Linux simply use the terminal app, on Windows Powershell or WSL would work.

Getting our Image

We have provided two images for our container, one compatible with arm64 devices, the other with amd64. Make sure you use the correct image for your computer in order to be able to successfully run our program.

The images are located in a Google Drive folder, located here.

Simply download whichever image pertains to your computer.

Once you have downloaded the tar file, open a terminal window and locate the downloaded file.

The following command can be used to load the tar file into an image.

```
docker load -i pq_app_<arm/amd>_image.tar
```

To confirm the image was properly configured, run the command:

```
docker images
```

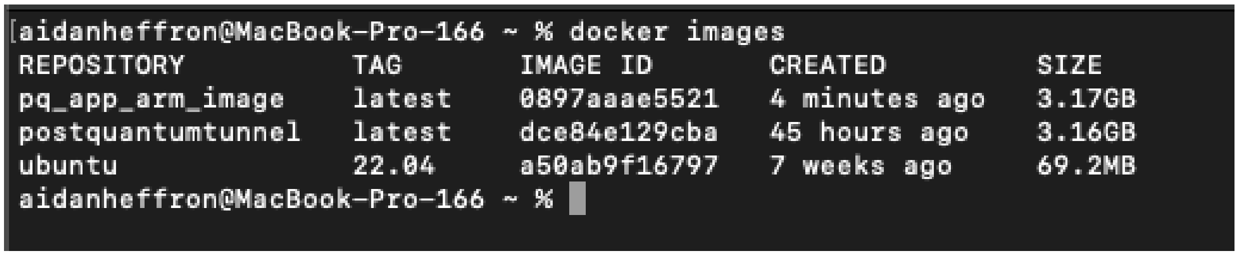
Listed should be an image with the name `pq_app_<arm/amd>_image`

Running the Image

To run the image as a container, we need to get the image ID of the application. Run the following command to print out your images and their IDs:

```
docker images
```

This will produce a result like the picture below:



```
[aidanheffron@MacBook-Pro-166 ~ % docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
pq_app_arm_image    latest    0897aaae5521   4 minutes ago    3.17GB
postquantumtunnel   latest    dce84e129cba   45 hours ago     3.16GB
ubuntu              22.04     a50ab9f16797   7 weeks ago      69.2MB
aidanheffron@MacBook-Pro-166 ~ %
```

Take note of the “Image ID” field for the pq_app_<arm/amd>_image, in my case the ID is: 0897aaae5521

Now, we will launch this image in interactive mode.

```
docker run -it --name pq_app_<arm/amd>_container <Image ID>
```

Using the Container

To use our program, we need to initiate some environment variables, luckily all of these variables are already defined in /root/.bashrc so all we need to do is source this file

```
source /root/.bashrc
```

****IMPORTANT** Every time the container is launched, you will need to source this file, otherwise environment variables will not be set and you will encounter errors!**

And voila! Everything is now set up to use our application. Project code can be found in

```
/root/quantumsafe/ProjectCode
```

To run the server and client:

****UPDATE AS OF APRIL 13th, 2024****

After reviewing some feedback with the user, there have been some quick updates to the source code, as such we need to pull these changes in server and client.

```
cd /root/quantumsafe/ProjectCode/server
git pull

cd /root/quantumsafe/ProjectCode/client
git pull
```

Now, with the updated changes, to run the server:

```
cd /root/quantumsafe/ProjectCode/server

go run *.go
```

This will take over your terminal, so to then run the client you will need to open a new tab in your terminal and get back into the container. To get the container ID of the running container, use:

```
docker ps
```

Copy the “Container ID” and use it in the following command:

```
docker exec -it <container ID> /bin/bash
```

Now, you should be inside the docker container in your second terminal. Make sure to source the .bashrc file before continuing!

```
source /root/.bashrc
```

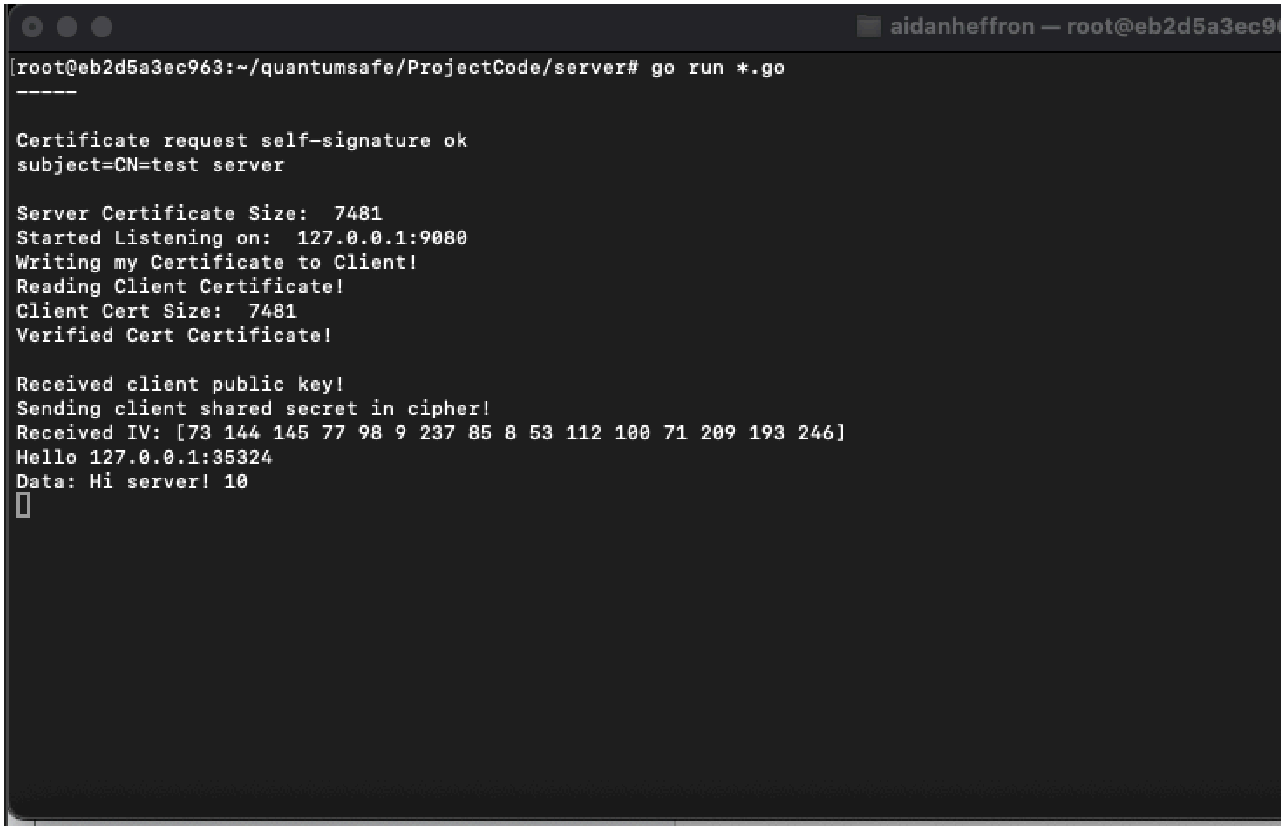
Then to run the client, use:

```
cd /root/quantumsafe/ProjectCode/client

```

```
go run *.go
```

For a more exhaustive list on how to run and interact with our program, look into the Running Our Program documentation

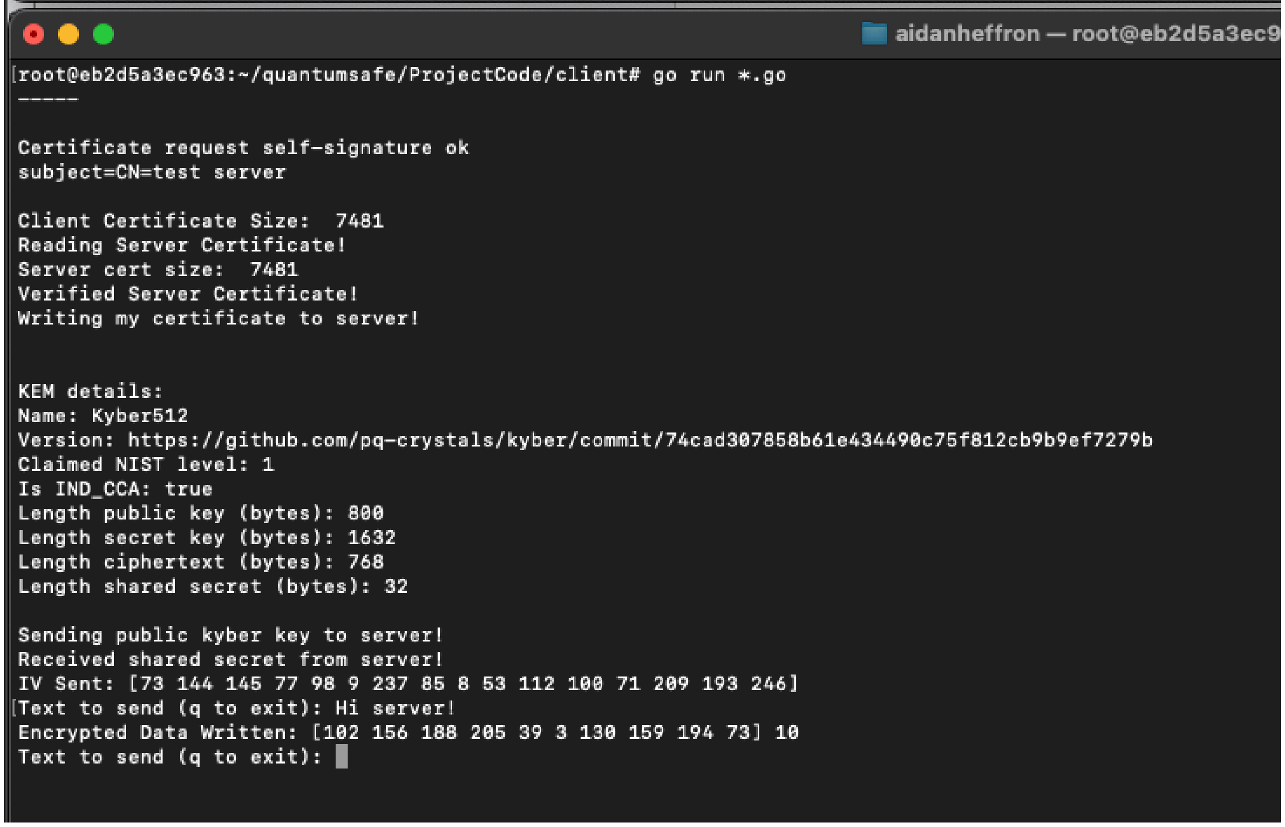


```
[root@eb2d5a3ec963:~/quantumsafe/ProjectCode/server# go run *.go
-----

Certificate request self-signature ok
subject=CN=test server

Server Certificate Size:  7481
Started Listening on:  127.0.0.1:9080
Writing my Certificate to Client!
Reading Client Certificate!
Client Cert Size:  7481
Verified Cert Certificate!

Received client public key!
Sending client shared secret in cipher!
Received IV: [73 144 145 77 98 9 237 85 8 53 112 100 71 209 193 246]
Hello 127.0.0.1:35324
Data: Hi server! 10
```



```
[root@eb2d5a3ec963:~/quantumsafe/ProjectCode/client# go run *.go
-----

Certificate request self-signature ok
subject=CN=test server

Client Certificate Size:  7481
Reading Server Certificate!
Server cert size:  7481
Verified Server Certificate!
Writing my certificate to server!


KEM details:
Name: Kyber512
Version: https://github.com/pq-crystals/kyber/commit/74cad307858b61e434490c75f812cb9b9ef7279b
Claimed NIST level: 1
Is IND_CCA: true
Length public key (bytes): 800
Length secret key (bytes): 1632
Length ciphertext (bytes): 768
Length shared secret (bytes): 32

Sending public kyber key to server!
Received shared secret from server!
IV Sent: [73 144 145 77 98 9 237 85 8 53 112 100 71 209 193 246]
[Text to send (q to exit): Hi server!
Encrypted Data Written: [102 156 188 205 39 3 130 159 194 73] 10
Text to send (q to exit):
```