Technician's Guide:

Resource & Auth Server Requirements:

- **OpenJDK**'s Java 21 downloaded into your home directory on the Pitt CS linux cluster.
  - **OpenJDK** is required as Oracle Java has a bug with Bouncy Castle that prevents the server's from performing their cryptographic operations.

```
# After SSH'ing into the CS linux cluster:
wget
https://download.java.net/java/GA/jdk21.0.1/415e3f918a1f4062a0074a2794853d
0d/12/GPL/openjdk-21.0.1_linux-x64_bin.tar.gz
tar -xzvf openjdk-21.0.1_linux-x64_bin.tar.gz
rm -rf openjdk-21.0.1_linux-x64_bin.tar.gz
```

Client Requirements:

- Node.js installed on your local machine.
  - Download Node.js https://nodejs.org/en/download/.

Auth Server

Downloading the Auth Server:

- SSH into the Pitt machine that you would like to run this on.
- Download the latest release of the Auth Server:
  - Note: A GitHub Personal Access Token (PAT) is required since this project repository is private. If you need to create one, view the instructions https://docs.github.com/en/github/authenticating-to-github/keeping-your-account-and-data-secure/creating-a-personal-access-token.

```
authServerAssetID=$(curl -L -H "Authorization: Bearer <Github PAT>" \
-s https://api.github.com/repos/2241-cs1653/cs1653-project-
securecoders/releases/latest \
| /afs/pitt.edu/home/m/a/mab650/public/jq \
'.assets[] | select(.name == "AuthServer.zip") | .id')

wget --auth-no-challenge --header='Accept:application/octet-stream' \
'https://<GitHub PAT>:@api.github.com/repos/2241-cs1653/cs1653-project-
securecoders
/releases/assets/$authServerAssetID' \
-O AuthServer.zip

unzip AuthServer.zip -d authServer
```

Preparing to run the Auth Server

- First `cd` into the `authServer` directory and generate a key pair for the auth server using `generate_key_pair`.
    - `generate_key_pair` takes the following arguments:
        - `-pu/--pub-key-path`: The path to the directory where you want to store the public key. This must be a public directory so that the resource server can access it.
        - `-pv/--priv-key-path`: (optional): The path to the directory where you want to store the private key. If a directory is not provided then it will be stored in the same directory that this script is running in.
        - `-n/--name`: The name of the key pair. This will be used to name the files that are generated.
    - Additionally you can run `./generate_key_pair -h/--help` for more help on the usage of this script.

```
# example
cd authServer
./generate_key_pair -pu ~/public -pv ~/private -n authServerKeyPair
```

Running the Auth Server

```
/path/to/your/java20Installation/bin/java -jar \
Auth-0.0.1-SNAPSHOT.jar <port to run on> <path to public key> <path to
private key>
```

Resource Server

Downloading the Resource Server:

- SSH into the Pitt machine that you would like to run this on.
- Download the latest release of the Resource Server:
  - Note: A GitHub Personal Access Token is required since this project repository is private. If you need to create one, view the instructions: https://docs.github.com/en/github/authenticating-to-github/keeping-your-account-and-data-secure/creating-a-personal-access-token

```
resourceServerID=$(curl -L -H "Authorization: Bearer <Github PAT>" \
-s https://api.github.com/repos/2241-cs1653/cs1653-project-
securecoders/releases/latest \
| /afs/pitt.edu/home/m/a/mab650/public/jq \
'.assets[] | select(.name == "ResourceServer.zip") | .id')

wget --auth-no-challenge --header='Accept:application/octet-stream' \
'https://<GitHub PAT>:@api.github.com/repos/2241-cs1653/cs1653-project-
securecoders
/releases/assets/$resourceServerID' \
-O ResourceServer.zip

unzip ResourceServer.zip -d resourceServer
```

Preparing to run the Resource Server

- First `cd` into the `resourceServer` directory and generate a key pair for the resource server using `generate_key_pair`.
  - `generate_key_pair` takes the following arguments:
    - `-pu/--pub-key-path`: The path to the directory where you want to store the public key. This must be a public directory so that the resource server can access it.
    - `-pv/--priv-key-path`: (optional): The path to the directory where you want to store the private key. If a directory is not provided then it will be stored in the same directory that this script is running in.
    - `-n/--name`: The name of the key pair. This will be used to name the files that are generated.
  - Additionally you can run `./generate_key_pair -h/--help` for more help on the usage of this script.

```
# example
cd resourceServer
./generate_key_pair -pu ~/public -pv ~/private -n resourceServerKeyPair
```

- Next, obtain the public key of the auth server using `get_auth_server_pub_key`
  - `get_auth_server_pub_key` takes the following arguments:

        - `-p/--pub-key-path`: The path to the auth servers public key.
        - `-sp/--store-path`: The path to the directory where you want to store the auth servers public key.
    - Additionally you can run `./get_auth_server_pub_key -h/--help` for more help on the usage of this script.

```
# example
./get_auth_server_pub_key -p ~/public/authServerKeyPair.pub -sp .
```

Running the Resource Server

```
/path/to/your/java20Installation/bin/java -jar \
ResourceServer-0.0.1-SNAPSHOT.jar \
<port> <path to public key> <path to private key> <path to auth server
public key>
```

Client

- Navigate to `https://github.com/2241-cs1653/cs1653-project-securecoders/releases/latest` in your browser.
- Download `Client.zip`
- Extract this file into a directory that you would like to run the client from.
- Open a terminal in the directory that you extracted the client to.
- Obtain the public key of the resource server

Obtain the public key of the resource server (Mac/Linux)

- The public key of the resource server can be obtained using `get_resource_server_pub_key`
  - `get_resource_server_pub_key` takes the following arguments:
    - `-p/--pub-key-path`: The path to the resource servers public key.
    - `-sp/--store-path`: The path to the directory where you want to store the resource servers public key.
    - `-u/--username`: Your Pitt username to build the scp command.
    - `-m/--machine-name`: (optional): The name of the machine to scp from. If not provided then it will default to `ritchie`.
  - Additionally you can run `./get_resource_server_pub_key -h/--help` for more help on the usage of this script.

```
# example:
./get_resource_server_pub_key -p ~/public/resourceServerKeyPair.pub -sp . -u mab650
```

Obtain the public key of the resource server (Windows)

- The public key of the resource server can be obtained using `Get-ResourceServerPublicKey.ps1`
  - `Get-ResourceServerPublicKey.ps1` takes the following arguments:
    - `-pubKeyPath`: The path to the resource servers public key.
    - `-storePath`: The path to the directory where you want to store the resource servers public key.
    - `-userName`: Your Pitt username to build the scp command.
    - `-machineName`: (optional): The name of the machine to scp from. If not provided then it will default to `ritchie`.
  - Additionally you can run `Get-Help .\Get-ResourceServerPublicKey.ps1` for more help on the usage of this script.

```
# example:
.\Get-ResourceServerPublicKey.ps1 `
-pubKeyPath ~/public/resourceServerKeyPair.pub -storePath . -userName mab650
```

Create an SSH tunnel for the resource and auth servers

- For each server, open a new terminal and run the following command:
    - `-N` is not necessary but it will prevent the terminal from opening a shell on the remote machine.

```
ssh -L -N <localPort>:<remoteAddress>:<remotePort> <username>@<server>

# example, if I'm running one of these servers on ritchie.cs.pitt.edu:
ssh -L -N 8080:localhost:8080 mab650@ritchie.cs.pitt.edu
```

Running the Client

```
npm run prod
```

- Consult the User's manual on where to place the urls for the auth and resource servers and the path to the resource server public key.