Assignment-1 RMI & Socket

Dhruvit Patel (CWID: 10404032)

Two types of mode for FTP:

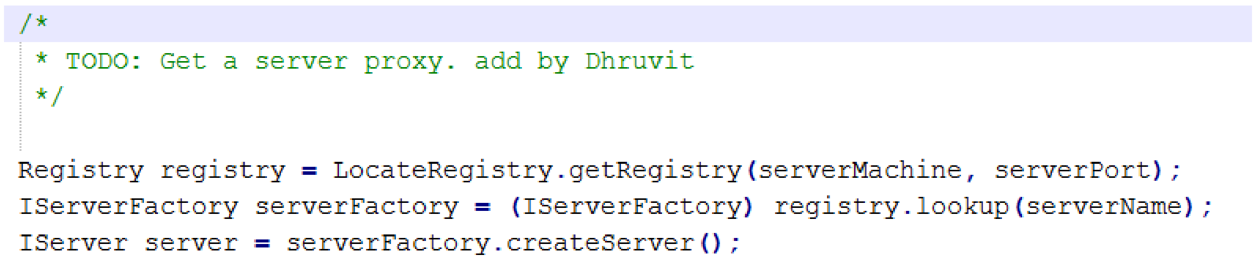
Active Mode:

In Active mode, Client acts like a server. Client create socket and Server connect to this socket for File transfer.

Passive Mode:

In Passive mode, Server create a socket and client connect to this socket. Client makes the server to work in passive mode.

Client Side

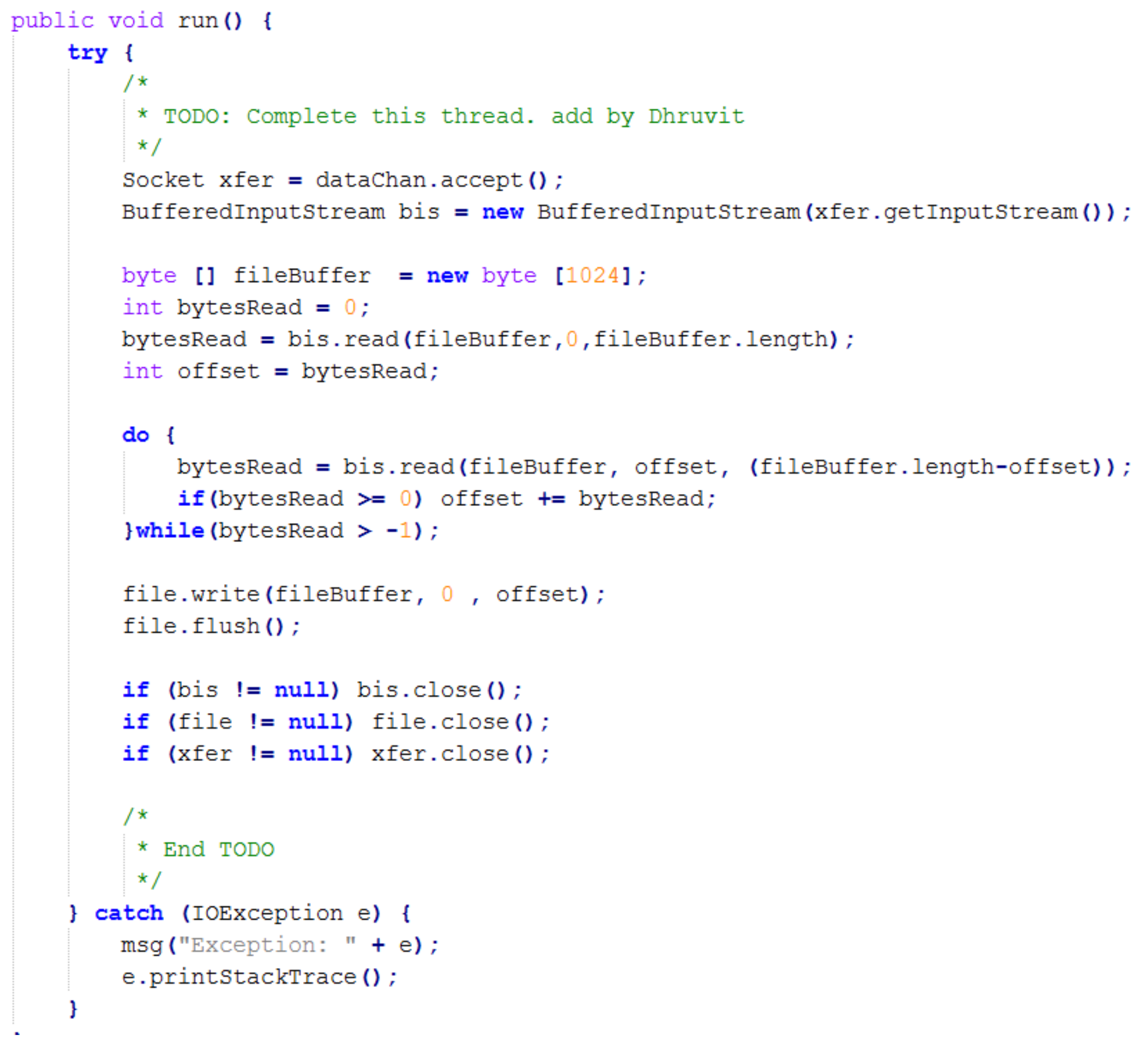


```
/*
 * TODO: Get a server proxy. add by Dhruvit
 */

Registry registry = LocateRegistry.getRegistry(serverMachine, serverPort);
IServerFactory serverFactory = (IServerFactory) registry.lookup(serverName);
IServer server = serverFactory.createServer();
```

Here, Client searching for socket which is created by the server. Client look up into registry.

File Transfer Operations:

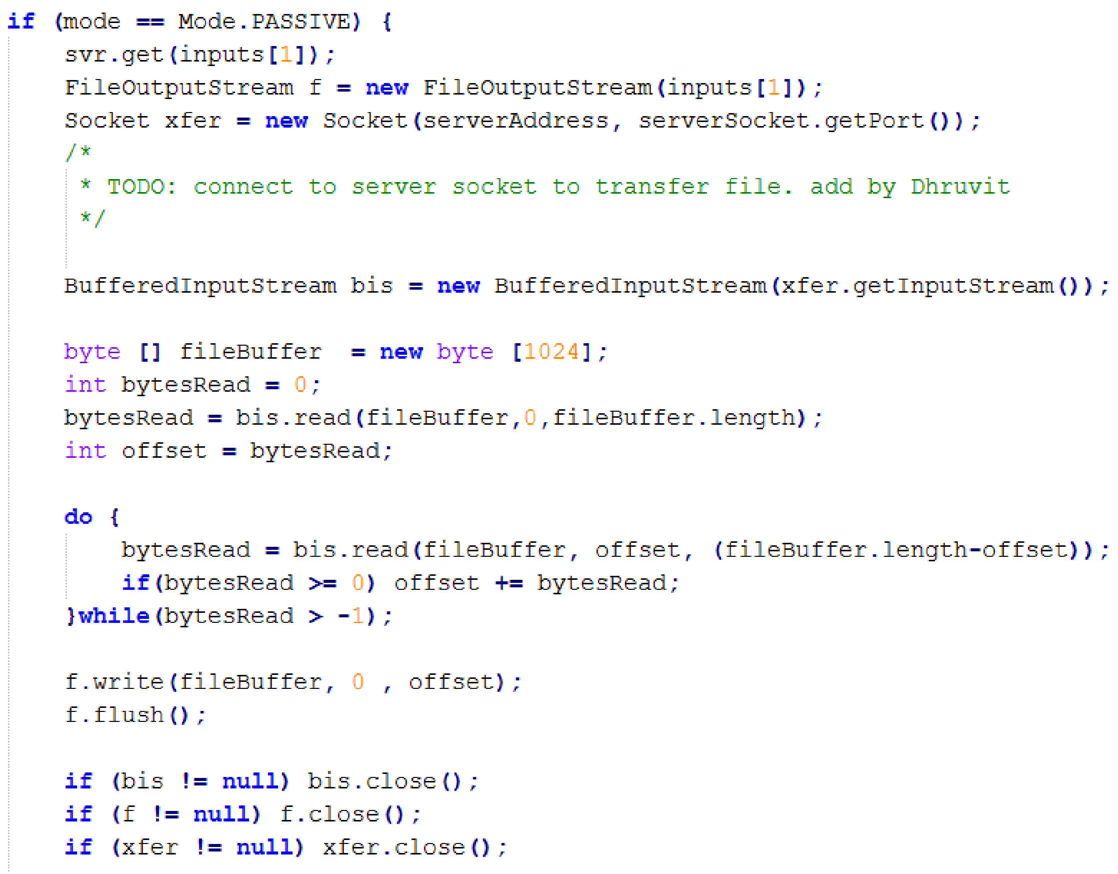
After binding with the server, client choose operation from client side terminal for the file transfer. Following is Get Thread for accepting connection from the server.



```java
public void run() {
    try {
        /*
         * TODO: Complete this thread. add by Dhruvit
         */
        Socket xfer = dataChan.accept();
        BufferedInputStream bis = new BufferedInputStream(xfer.getInputStream());

        byte [] fileBuffer  = new byte [1024];
        int bytesRead = 0;
        bytesRead = bis.read(fileBuffer,0,fileBuffer.length);
        int offset = bytesRead;

        do {
            bytesRead = bis.read(fileBuffer, offset, (fileBuffer.length-offset));
            if(bytesRead >= 0) offset += bytesRead;
        }while(bytesRead > -1);

        file.write(fileBuffer, 0 , offset);
        file.flush();

        if (bis != null) bis.close();
        if (file != null) file.close();
        if (xfer != null) xfer.close();

        /*
         * End TODO
         */
    } catch (IOException e) {
        msg("Exception: " + e);
        e.printStackTrace();
    }
```

GET Operation:

Here, Client listen for connection from server, when connection is accepted, client create input stream for getting file from the server.



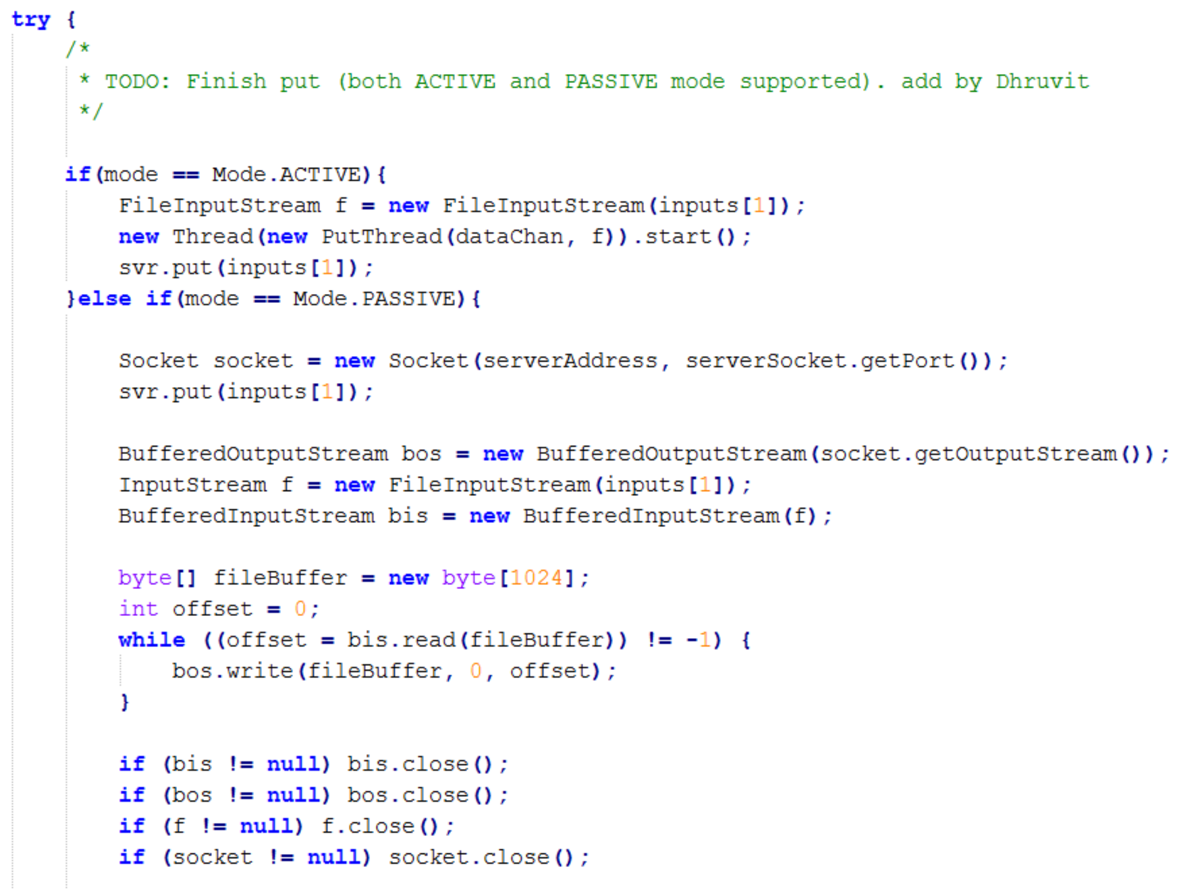
```
if (mode == Mode.PASSIVE) {
    svr.get(inputs[1]);
    FileOutputStream f = new FileOutputStream(inputs[1]);
    Socket xfer = new Socket(serverAddress, serverSocket.getPort());
    /*
     * TODO: connect to server socket to transfer file. add by Dhruvit
     */

    BufferedInputStream bis = new BufferedInputStream(xfer.getInputStream());

    byte [] fileBuffer  = new byte [1024];
    int bytesRead = 0;
    bytesRead = bis.read(fileBuffer,0,fileBuffer.length);
    int offset = bytesRead;

    do {
        bytesRead = bis.read(fileBuffer, offset, (fileBuffer.length-offset));
        if(bytesRead >= 0) offset += bytesRead;
    }while(bytesRead > -1);

    f.write(fileBuffer, 0 , offset);
    f.flush();

    if (bis != null) bis.close();
    if (f != null) f.close();
    if (xfer != null) xfer.close();
```

In passive mode, Client gets the output stream created by the server for downloading file.

Creating the file input stream for transfer and acceptance of socket

Put Operation:

The active and passive modes for put operation work similar to the get operation as describe previously.



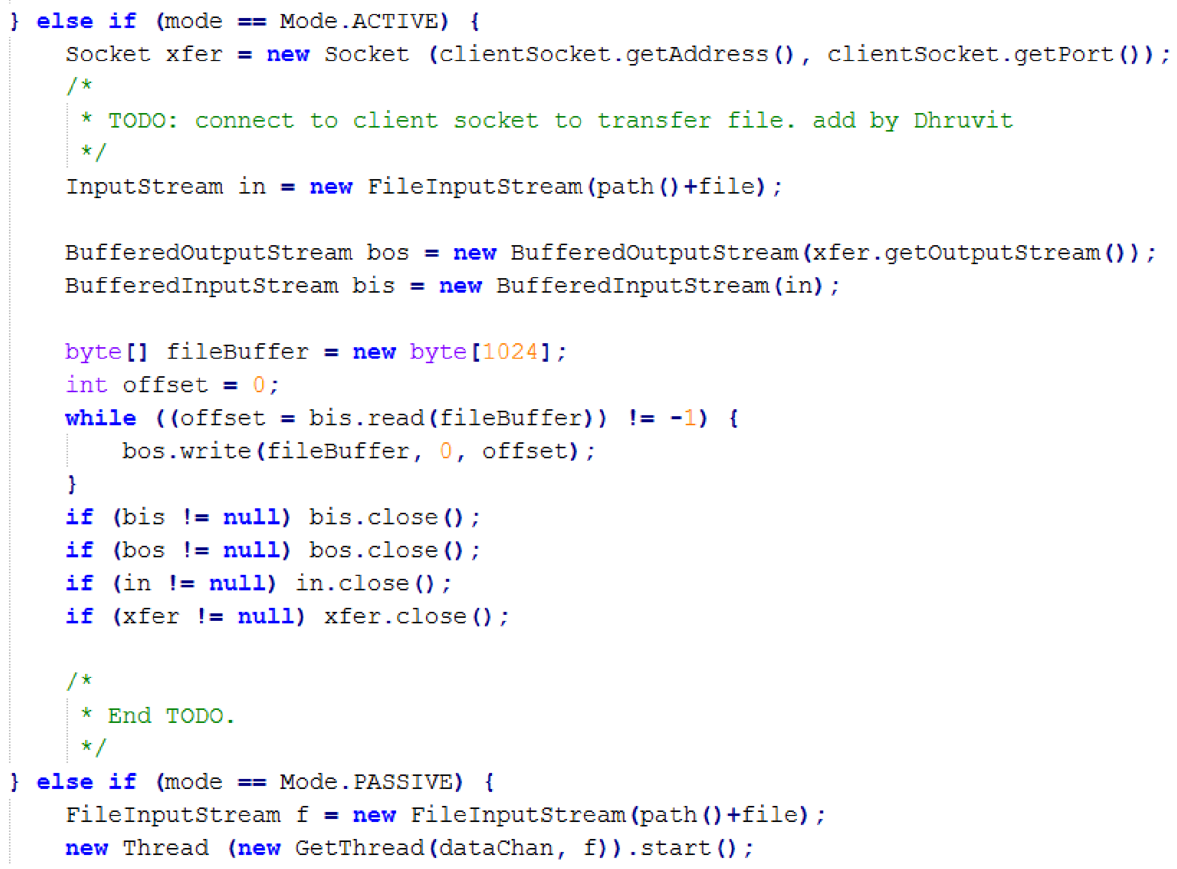
```
try {
    /*
     * TODO: Finish put (both ACTIVE and PASSIVE mode supported). add by Dhruvit
     */

    if(mode == Mode.ACTIVE){
        FileInputStream f = new FileInputStream(inputs[1]);
        new Thread(new PutThread(dataChan, f)).start();
        svr.put(inputs[1]);
    }else if(mode == Mode.PASSIVE){

        Socket socket = new Socket(serverAddress, serverSocket.getPort());
        svr.put(inputs[1]);

        BufferedOutputStream bos = new BufferedOutputStream(socket.getOutputStream());
        InputStream f = new FileInputStream(inputs[1]);
        BufferedInputStream bis = new BufferedInputStream(f);

        byte[] fileBuffer = new byte[1024];
        int offset = 0;
        while ((offset = bis.read(fileBuffer)) != -1) {
            bos.write(fileBuffer, 0, offset);
        }

        if (bis != null) bis.close();
        if (bos != null) bos.close();
        if (f != null) f.close();
        if (socket != null) socket.close();
```

Server Side code:

Get thread in server side, getting connection request from client and upload file in server.

```java
public void run () {
    /*
     * TODO: Process a client request to transfer a file. add by Dhruvit
     */

    try {
    Socket socket = dataChan.accept();

    BufferedOutputStream bos = new BufferedOutputStream(socket.getOutputStream());
    BufferedInputStream bis = new BufferedInputStream(file);

    byte[] fileBuffer = new byte[1024];
    int offset = 0;
    while ((offset = bis.read(fileBuffer)) != -1) {
        bos.write(fileBuffer, 0, offset);
    }
    if (bis != null) bis.close();
    if (bos != null) bos.close();
    if (file != null) file.close();
    if (socket != null) socket.close();

    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Get operation in server side for both active and passive mode.



```
} else if (mode == Mode.ACTIVE) {
    Socket xfer = new Socket (clientSocket.getAddress(), clientSocket.getPort());
    /*
     * TODO: connect to client socket to transfer file. add by Dhruvit
     */
    InputStream in = new FileInputStream(path()+file);

    BufferedOutputStream bos = new BufferedOutputStream(xfer.getOutputStream());
    BufferedInputStream bis = new BufferedInputStream(in);

    byte[] fileBuffer = new byte[1024];
    int offset = 0;
    while ((offset = bis.read(fileBuffer)) != -1) {
        bos.write(fileBuffer, 0, offset);
    }
    if (bis != null) bis.close();
    if (bos != null) bos.close();
    if (in != null) in.close();
    if (xfer != null) xfer.close();

    /*
     * End TODO.
     */
} else if (mode == Mode.PASSIVE) {
    FileInputStream f = new FileInputStream(path()+file);
    new Thread (new GetThread(dataChan, f)).start();
```

Put operation for active and passive mode on server side

```
/*
 * TODO: Finish put (both ACTIVE and PASSIVE). add by Dhruvit
 */

try {
    if(mode == Mode.ACTIVE){

        Socket xfer = new Socket(clientSocket.getAddress(),clientSocket.getPort());
        BufferedInputStream bis = new BufferedInputStream(xfer.getInputStream());
        FileOutputStream f = new FileOutputStream(path()+file);

        int offset = 0;
        byte[] fileBuffer = new byte[1024];
        while ((offset = bis.read(fileBuffer)) != -1) {
            f.write(fileBuffer, 0, offset);
        }

        if (bis != null) bis.close();
        if (f != null) f.close();
        if (xfer != null) xfer.close();

    }else if(mode == Mode.PASSIVE){

        FileOutputStream f = new FileOutputStream(path() + file);
        new Thread(new PutThread(dataChan, f)).start();

    }
} catch (IOException e) {
    // TODO: handle exception. add by Dhruvit
    e.printStackTrace();
}
```

Testing:

I tested following test cases client server communication and I also demonstrated in video.

- Print the current working directory.
- Listing the contents of the remote directory.
- Upload and download files.

Following procedure for running in local machine:

- Run ftpd.sh in the local machine

  jar -xf ftpd.jar ftpd.sh
- Run ftp.sh in the local machine.

  jar -xf ftp.jar ftp.sh

Following procedure for running in remote machine

Created Amazon Ec2 instance. Connecting this instance with SSH command, primary key and Public DNS of Ec2 instance.

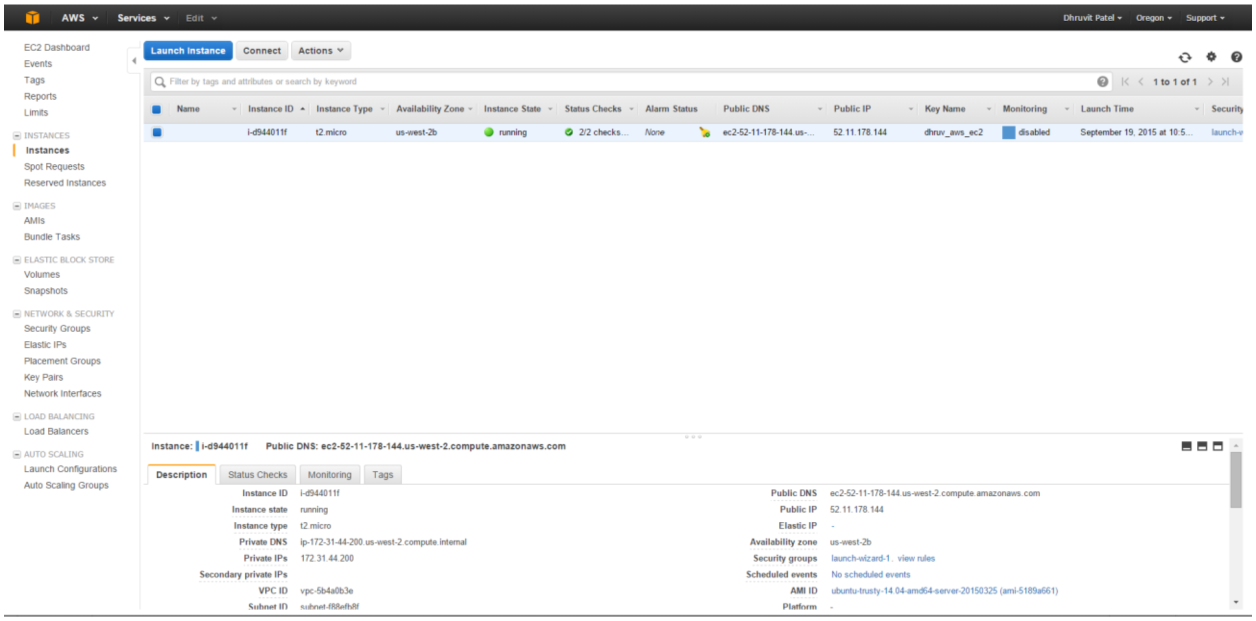
For login in to Ec2 Instance.

```
        sudo ssh -i ~/AWS_EC2/dhruv_aws_ec2.pem ubuntu@ec2-52-11-178-144.us-west
2.compute.amazonaws.com
```

Put the file from local machine to Ec2

```
        sudo scp -i ~/AWS_EC2/dhruv_aws_ec2.pem ftpd.jar ubuntu@ec2-52-11-178-144.us-west-
2.compute.amazonaws.com:~/tmp/cs549/ftp-test/ftpd.jar
```

Attached file is for Amazon Ec2 Instance.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key Name	Monitoring	Launch Time	Security
	i-d944011f	t2.micro	us-west-2b	running	2/2 checks...	None	ec2-52-11-178-144.us-...	52.11.178.144	dhruv_aws_ec2	disabled	September 19, 2015 at 10:5...	launch-v

Instance: i-d944011f Public DNS: ec2-52-11-178-144.us-west-2.compute.amazonaws.com

Instance ID	i-d944011f	Public DNS	ec2-52-11-178-144.us-west-2.compute.amazonaws.com
Instance state	running	Public IP	52.11.178.144
Instance type	t2.micro	Elastic IP	-
Private DNS	ip-172-31-44-200.us-west-2.compute.internal	Availability zone	us-west-2b
Private IPs	172.31.44.200	Security groups	launch-wizard-1 . view rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-5b4a0b3e	AMI ID	ubuntu-trusty-14.04-amd64-server-20150325 (ami-5189a661)
Subnet ID	subnet-f88efb8f	Platform	-