Practical – 2

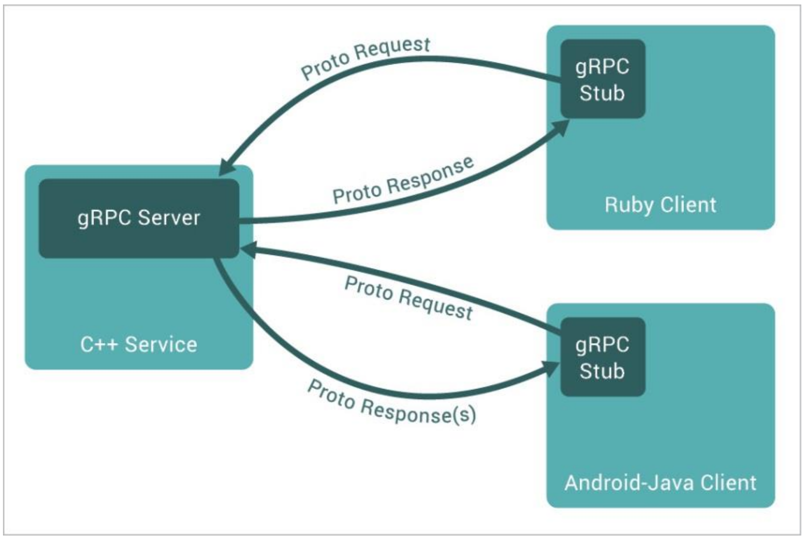
Aim: Build a Java-based service that exposes an API using the gRPC framework, and then writing a client to use the generated gRPC client-side stub.

Theory:

gRPC (gRPC Remote Procedure Calls[1]) is an open source remote procedure call (RPC) system initially developed at Google. It uses HTTP/2 for transport, Protocol Buffers as the interface description language, and provides features such as authentication, bidirectional streaming and flow control, blocking or nonblocking bindings, and cancellation and timeouts. It generates cross-platform client and server bindings for many languages. Most common usage scenarios include connecting services in microservices style architecture and connect mobile devices, browser clients to backend services

Working of gRPC:

- A client application is able to call methods directly on a server-side application present on other machines.
- Service is defined, methods are specified which can be further remotely called with their parameters and return types.
- On the other hand, the server runs a GRPC server to handle client calls.
- It uses protocol buffers as the Interface Definition Language to enable communication between two different systems used for describing the service interface and the structure of payload messages.
- HTTP/2 – GRPC is basically a protocol built on top of HTTP/2. HTTP/2 is used as a transport.
- Protobuf serialization – Messages that we serialize both for the request and response are encoded with protocol buffers.
- Clients open one long-lived connection to GRPC server.
- A new HTTP/2 stream for each RPC call.
- Allows Client-Side and Server-Side Streaming.
- Bidirectional Streaming.



Code:

GreetingService.proto

```
syntax = "proto3";
package com.example.grpc;

// Request payload
message HelloRequest {
  // Each message attribute is strongly typed.
  // You also must assign a "tag" number.
  // Each tag number is unique within the message.
  string name = 1;

  // This defines a strongly typed list of String
  repeated string hobbies = 2;

  // There are many more basics types, like Enum, Map
  // See https://developers.google.com/protocol-buffers/docs/proto3
  // for more information.
}

message HelloResponse {
  string greeting = 1;
}

// Defining a Service, a Service can have multiple RPC operations
service GreetingService {
  // Define a RPC operation
  rpc greeting(HelloRequest) returns (stream HelloResponse);
}
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example.grpc</groupId>
  <artifactId>grpc-hello-server</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>grpc-hello-server</name>
  <url>http://maven.apache.org</url>
  <dependencies>
  <dependency>
      <groupId>io.grpc</groupId>
      <artifactId>grpc-netty</artifactId>
      <version>1.7.0</version>
    </dependency>
    <dependency>
      <groupId>io.grpc</groupId>
      <artifactId>grpc-protobuf</artifactId>
      <version>1.7.0</version>
    </dependency>
    <dependency>
      <groupId>io.grpc</groupId>
      <artifactId>grpc-stub</artifactId>
      <version>1.7.0</version>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <extensions>
      <extension>
        <groupId>kr.motd.maven</groupId>
        <artifactId>os-maven-plugin</artifactId>
        <version>1.5.0.Final</version>
      </extension>
    </extensions>
    <plugins>
      <plugin>
        <groupId>org.xolstice.maven.plugins</groupId>
        <artifactId>protobuf-maven-plugin</artifactId>
        <version>0.5.0</version>
        <configuration>
          <protocArtifact>com.google.protobuf:protoc:3.4.0:exe:${os.detected.classifier}</protocArtifact>
          <pluginId>grpc-java</pluginId>
          <pluginArtifact>io.grpc:protoc-gen-grpc-java:1.7.0:exe:${os.detected.classifier}</pluginArtifact>
        </configuration>
        <executions>
          <execution>
            <goals>
              <goal>compile</goal>
              <goal>compile-custom</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

GreetingServiceImpl.java

```
package com.example.grpc;

import io.grpc.stub.StreamObserver;

public class GreetingServiceImpl extends GreetingServiceGrpc.GreetingServiceImplBase {
  @Override
  public void greeting(GreetingServiceOuterClass.HelloRequest request,
        StreamObserver<GreetingServiceOuterClass.HelloResponse> responseObserver) {
  // HelloRequest has toString auto-generated.
    System.out.println(request);

    // You must use a builder to construct a new Protobuffer object
    GreetingServiceOuterClass.HelloResponse response =
GreetingServiceOuterClass.HelloResponse.newBuilder()
      .setGreeting("Hello there, " + request.getName())
      .build();

    // Use responseObserver to send a single response back
    responseObserver.onNext(response);
    responseObserver.onNext(response);
    responseObserver.onNext(response);

    // When you are done, you must call onCompleted.
    responseObserver.onCompleted();
  }
}
```

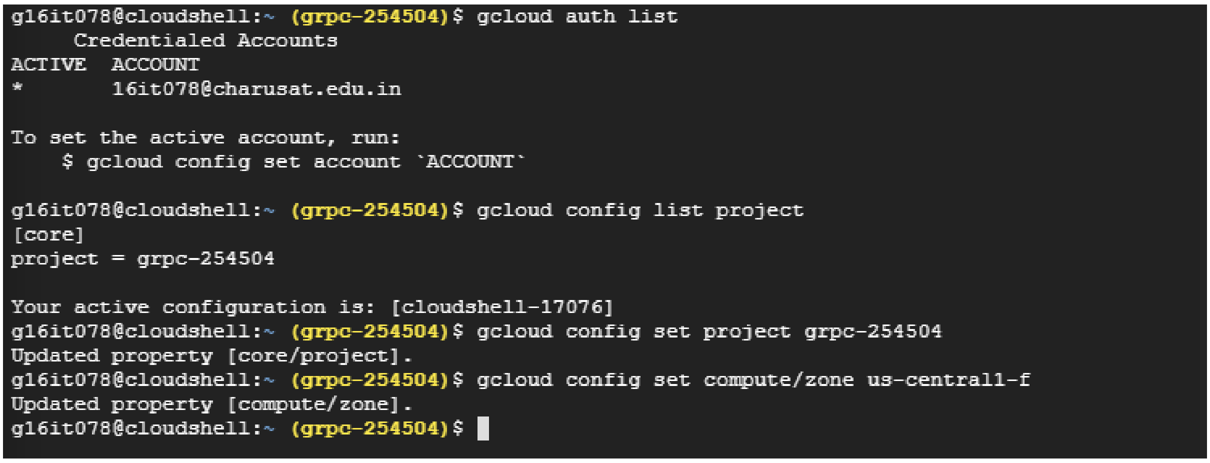
App.java

```
package com.example.grpc;

import io.grpc.*;

public class App
{
    public static void main( String[] args ) throws Exception
    {
      // Create a new server to listen on port 8080
      Server server = ServerBuilder.forPort(8080)
        .addService(new GreetingServiceImpl())
        .build();

      // Start the server
      server.start();

      // Server threads are running in the background.
      System.out.println("Server started");
      // Don't exit the main thread. Wait until server is terminated.
      server.awaitTermination();
    }
}
```

Client.java

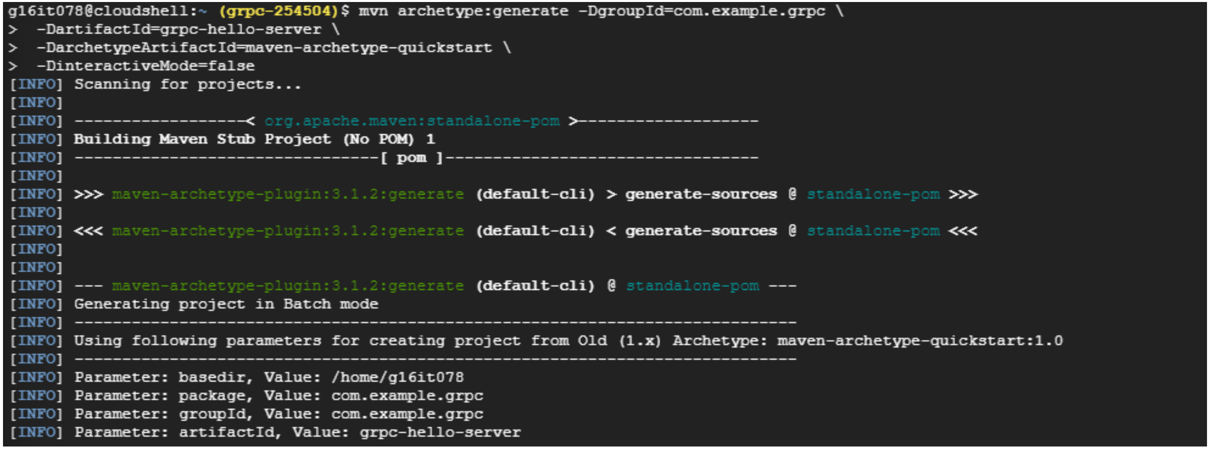
```
package com.example.grpc;
import io.grpc.*;

// New import
import io.grpc.stub.*;

public class Client
{
  public static void main( String[] args ) throws Exception
  {
    final ManagedChannel channel = ManagedChannelBuilder.forTarget("localhost:8080")
      .usePlaintext(true)
      .build();

    // Replace the previous synchronous code with asynchronous code.
    // This time use an async stub:
     GreetingServiceGrpc.GreetingServiceStub stub = GreetingServiceGrpc.newStub(channel);

    // Construct a request
    GreetingServiceOuterClass.HelloRequest request =
      GreetingServiceOuterClass.HelloRequest.newBuilder()
        .setName("Ray")
        .build();

    // Make an Asynchronous call. Listen to responses w/ StreamObserver
    stub.greeting(request, new StreamObserver<GreetingServiceOuterClass.HelloResponse>() {
      public void onNext(GreetingServiceOuterClass.HelloResponse response) {
        System.out.println(response);
      }
      public void onError(Throwable t) {
      }
      public void onCompleted() {
        // Typically you'll shutdown the channel somewhere else.
        // But for the purpose of the lab, we are only making a single
        // request. We'll shutdown as soon as this request is done.
        channel.shutdownNow();
      }
    });
  }
}
```

Output:



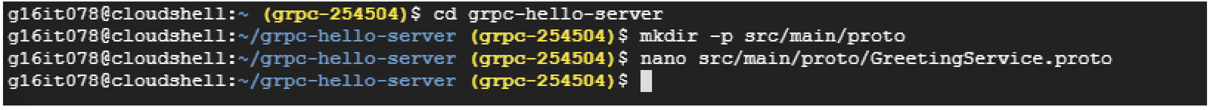
```
g16it078@cloudshell:~ (grpc-254504)$ gcloud auth list
     Credentialed Accounts
ACTIVE  ACCOUNT
*       16it078@charusat.edu.in

To set the active account, run:
    $ gcloud config set account `ACCOUNT`

g16it078@cloudshell:~ (grpc-254504)$ gcloud config list project
[core]
project = grpc-254504

Your active configuration is: [cloudshell-17076]
g16it078@cloudshell:~ (grpc-254504)$ gcloud config set project grpc-254504
Updated property [core/project].
g16it078@cloudshell:~ (grpc-254504)$ gcloud config set compute/zone us-central1-f
Updated property [compute/zone].
g16it078@cloudshell:~ (grpc-254504)$
```

Figure 1 Authenticate cloud shell

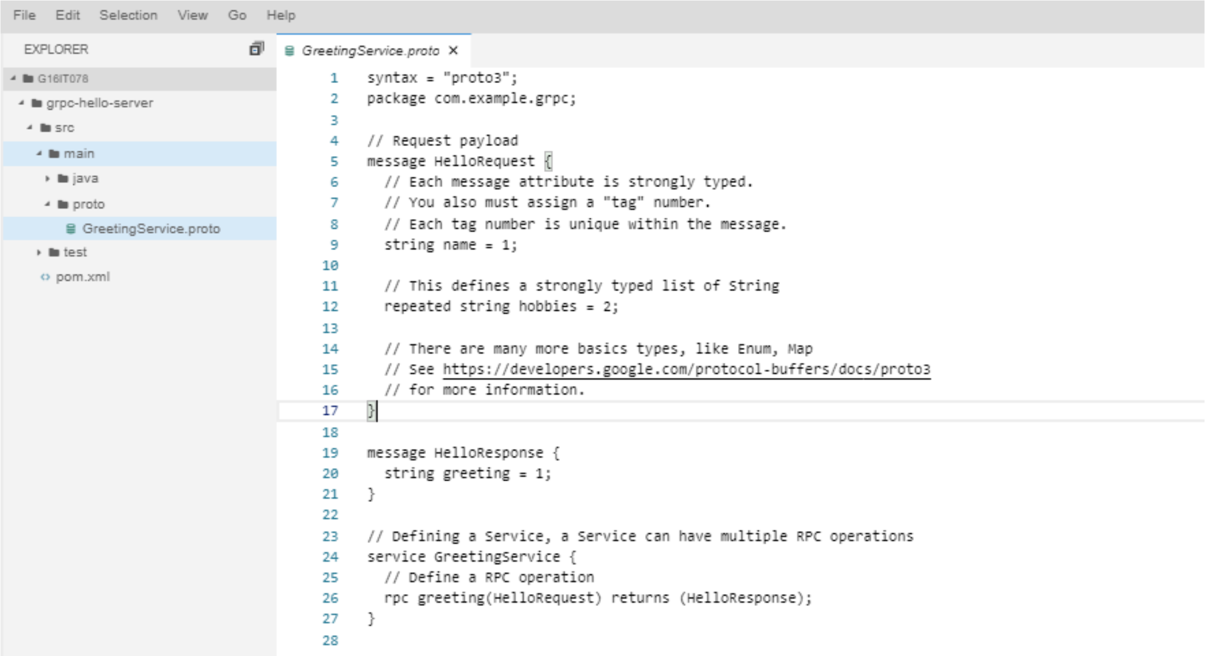


```
g16it078@cloudshell:~ (grpc-254504)$ mvn archetype:generate -DgroupId=com.example.grpc \
>  -DartifactId=grpc-hello-server \
>  -DarchetypeArtifactId=maven-archetype-quickstart \
>  -DinteractiveMode=false
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------< org.apache.maven:standalone-pom >------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.1.2:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.1.2:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.1.2:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: basedir, Value: /home/g16it078
[INFO] Parameter: package, Value: com.example.grpc
[INFO] Parameter: groupId, Value: com.example.grpc
[INFO] Parameter: artifactId, Value: grpc-hello-server
```

Figure 2 Create gRPC Service

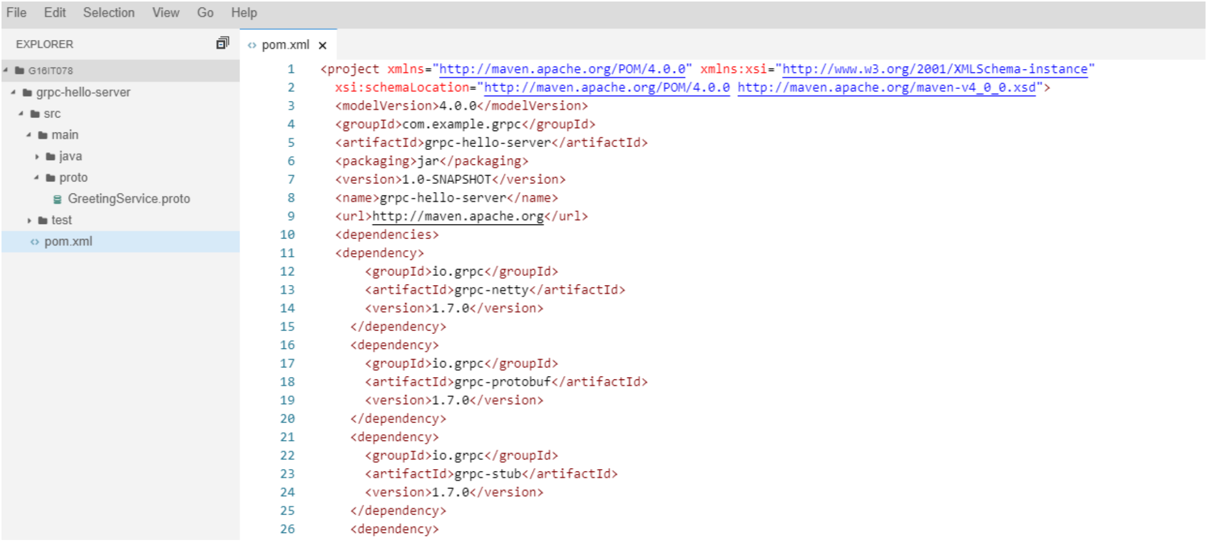


```
g16it078@cloudshell:~ (grpc-254504)$ cd grpc-hello-server
g16it078@cloudshell:~/grpc-hello-server (grpc-254504)$ mkdir -p src/main/proto
g16it078@cloudshell:~/grpc-hello-server (grpc-254504)$ nano src/main/proto/GreetingService.proto
g16it078@cloudshell:~/grpc-hello-server (grpc-254504)$
```



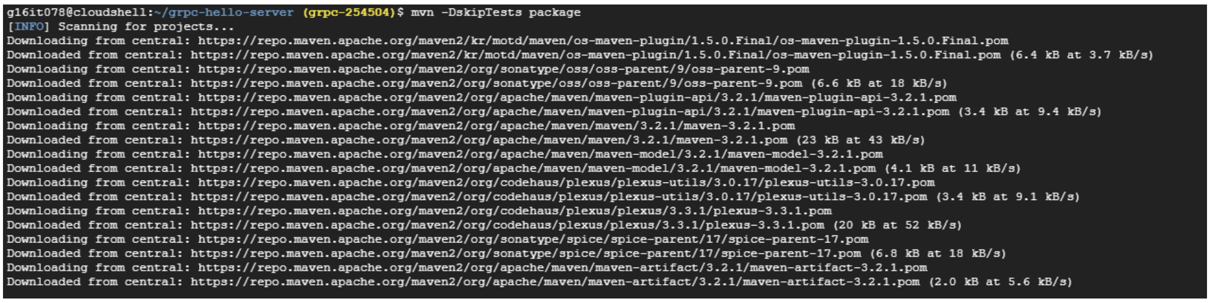
```
syntax = "proto3";
package com.example.grpc;

// Request payload
message HelloRequest {
  // Each message attribute is strongly typed.
  // You also must assign a "tag" number.
  // Each tag number is unique within the message.
  string name = 1;

  // This defines a strongly typed list of String
  repeated string hobbies = 2;

  // There are many more basics types, like Enum, Map
  // See https://developers.google.com/protocol-buffers/docs/proto3
  // for more information.
}

message HelloResponse {
  string greeting = 1;
}

// Defining a Service, a Service can have multiple RPC operations
service GreetingService {
  // Define a RPC operation
  rpc greeting(HelloRequest) returns (HelloResponse);
}
```

Figure 3 Create proto file

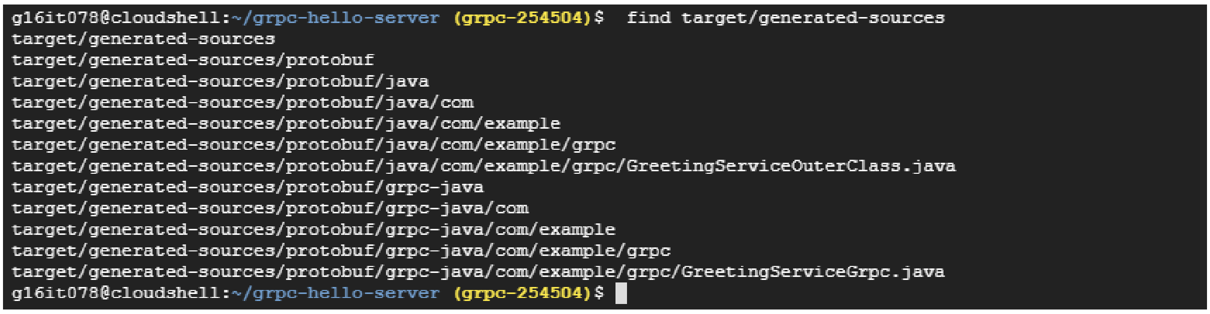


```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example.grpc</groupId>
  <artifactId>grpc-hello-server</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>grpc-hello-server</name>
  <url>http://maven.apache.org</url>
  <dependencies>
  <dependency>
      <groupId>io.grpc</groupId>
      <artifactId>grpc-netty</artifactId>
      <version>1.7.0</version>
    </dependency>
    <dependency>
      <groupId>io.grpc</groupId>
      <artifactId>grpc-protobuf</artifactId>
      <version>1.7.0</version>
    </dependency>
    <dependency>
      <groupId>io.grpc</groupId>
      <artifactId>grpc-stub</artifactId>
      <version>1.7.0</version>
    </dependency>
    <dependency>
```

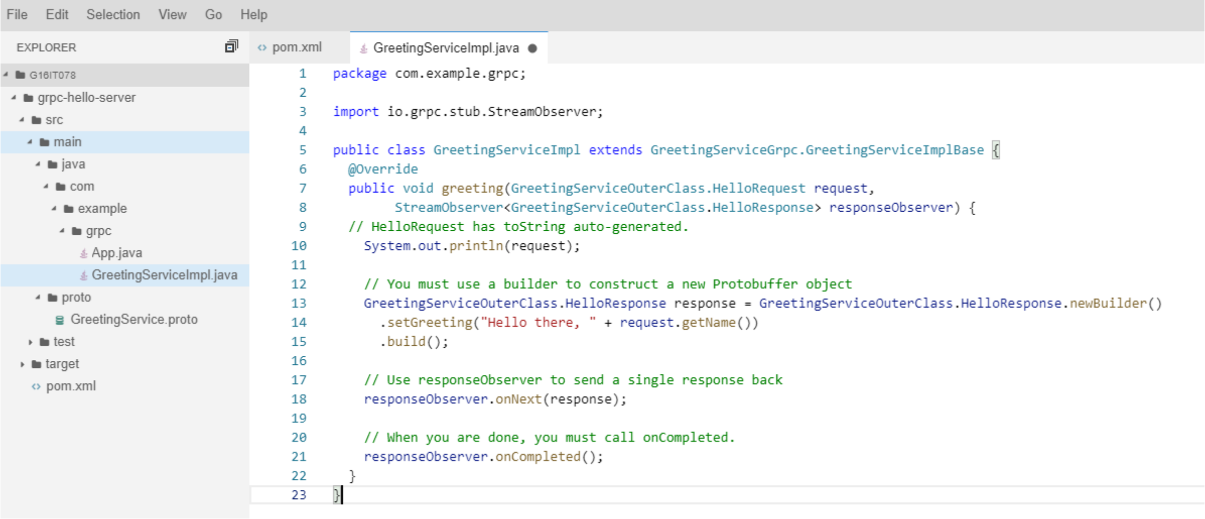
Figure 4 Include Dependencies in pom filr



```
g16it078@cloudshell:~/grpc-hello-server (grpc-254504)$ mvn -DskipTests package
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/kr/motd/maven/os-maven-plugin/1.5.0.Final/os-maven-plugin-1.5.0.Final.pom
Downloaded from central: https://repo.maven.apache.org/maven2/kr/motd/maven/os-maven-plugin/1.5.0.Final/os-maven-plugin-1.5.0.Final.pom (6.4 kB at 3.7 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/sonatype/oss/oss-parent/9/oss-parent-9.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/oss/oss-parent/9/oss-parent-9.pom (6.6 kB at 18 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-plugin-api/3.2.1/maven-plugin-api-3.2.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-plugin-api/3.2.1/maven-plugin-api-3.2.1.pom (3.4 kB at 9.4 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven/3.2.1/maven-3.2.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven/3.2.1/maven-3.2.1.pom (23 kB at 43 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-model/3.2.1/maven-model-3.2.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-model/3.2.1/maven-model-3.2.1.pom (4.1 kB at 11 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0.17/plexus-utils-3.0.17.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0.17/plexus-utils-3.0.17.pom (3.4 kB at 9.1 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus/3.3.1/plexus-3.3.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus/3.3.1/plexus-3.3.1.pom (20 kB at 52 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/sonatype/spice/spice-parent/17/spice-parent-17.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/spice/spice-parent/17/spice-parent-17.pom (6.8 kB at 18 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-artifact/3.2.1/maven-artifact-3.2.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-artifact/3.2.1/maven-artifact-3.2.1.pom (2.0 kB at 5.6 kB/s)
```

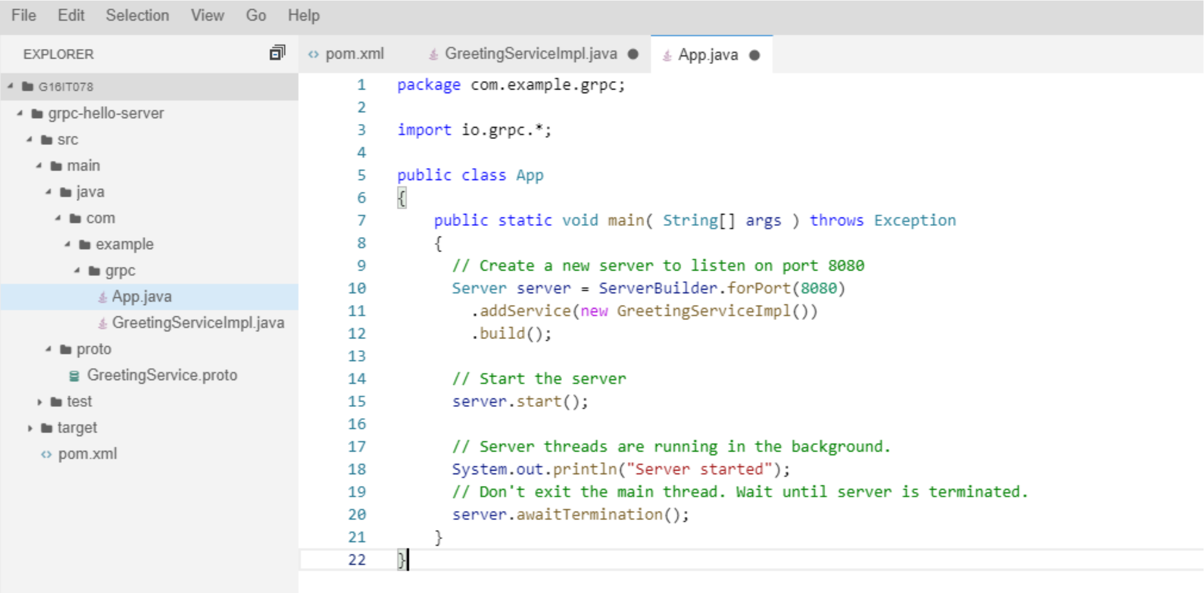


```
g16it078@cloudshell:~/grpc-hello-server (grpc-254504)$  find target/generated-sources
target/generated-sources
target/generated-sources/protobuf
target/generated-sources/protobuf/java
target/generated-sources/protobuf/java/com
target/generated-sources/protobuf/java/com/example
target/generated-sources/protobuf/java/com/example/grpc
target/generated-sources/protobuf/java/com/example/grpc/GreetingServiceOuterClass.java
target/generated-sources/protobuf/grpc-java
target/generated-sources/protobuf/grpc-java/com
target/generated-sources/protobuf/grpc-java/com/example
target/generated-sources/protobuf/grpc-java/com/example/grpc
target/generated-sources/protobuf/grpc-java/com/example/grpc/GreetingServiceGrpc.java
g16it078@cloudshell:~/grpc-hello-server (grpc-254504)$
```

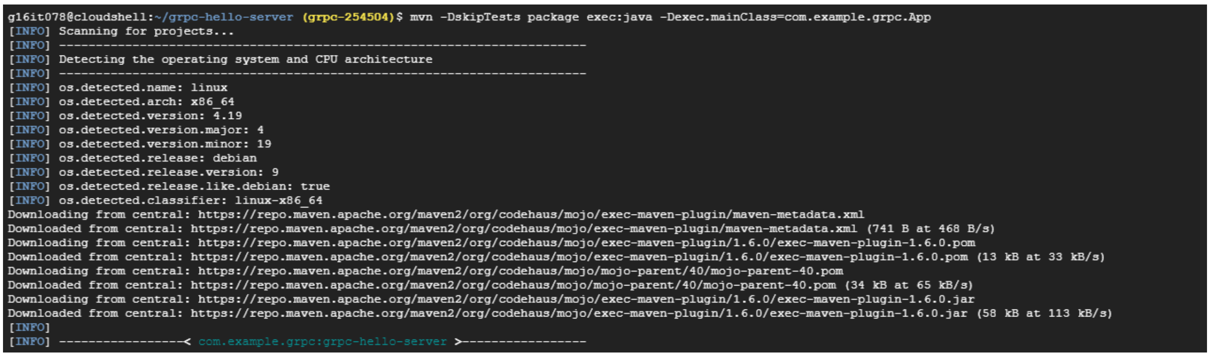


```java
package com.example.grpc;

import io.grpc.stub.StreamObserver;

public class GreetingServiceImpl extends GreetingServiceGrpc.GreetingServiceImplBase {
  @Override
  public void greeting(GreetingServiceOuterClass.HelloRequest request,
        StreamObserver<GreetingServiceOuterClass.HelloResponse> responseObserver) {
  // HelloRequest has toString auto-generated.
    System.out.println(request);

    // You must use a builder to construct a new Protobuffer object
    GreetingServiceOuterClass.HelloResponse response = GreetingServiceOuterClass.HelloResponse.newBuilder()
      .setGreeting("Hello there, " + request.getName())
      .build();

    // Use responseObserver to send a single response back
    responseObserver.onNext(response);

    // When you are done, you must call onCompleted.
    responseObserver.onCompleted();
  }
}
```

Figure 5 Create Service file

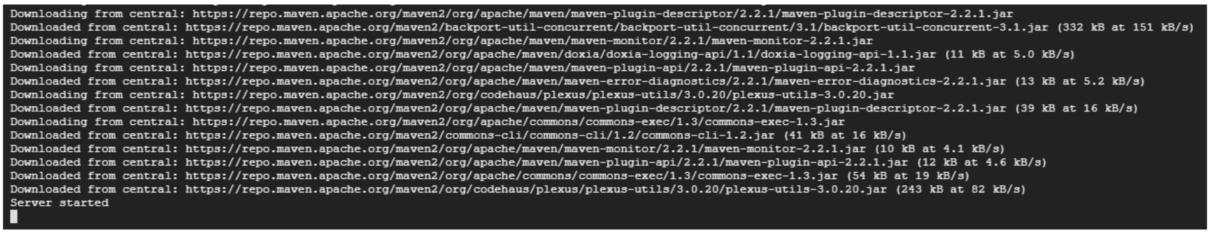


```java
package com.example.grpc;

import io.grpc.*;

public class App
{
    public static void main( String[] args ) throws Exception
    {
      // Create a new server to listen on port 8080
      Server server = ServerBuilder.forPort(8080)
        .addService(new GreetingServiceImpl())
        .build();

      // Start the server
      server.start();

      // Server threads are running in the background.
      System.out.println("Server started");
      // Don't exit the main thread. Wait until server is terminated.
      server.awaitTermination();
    }
}
```



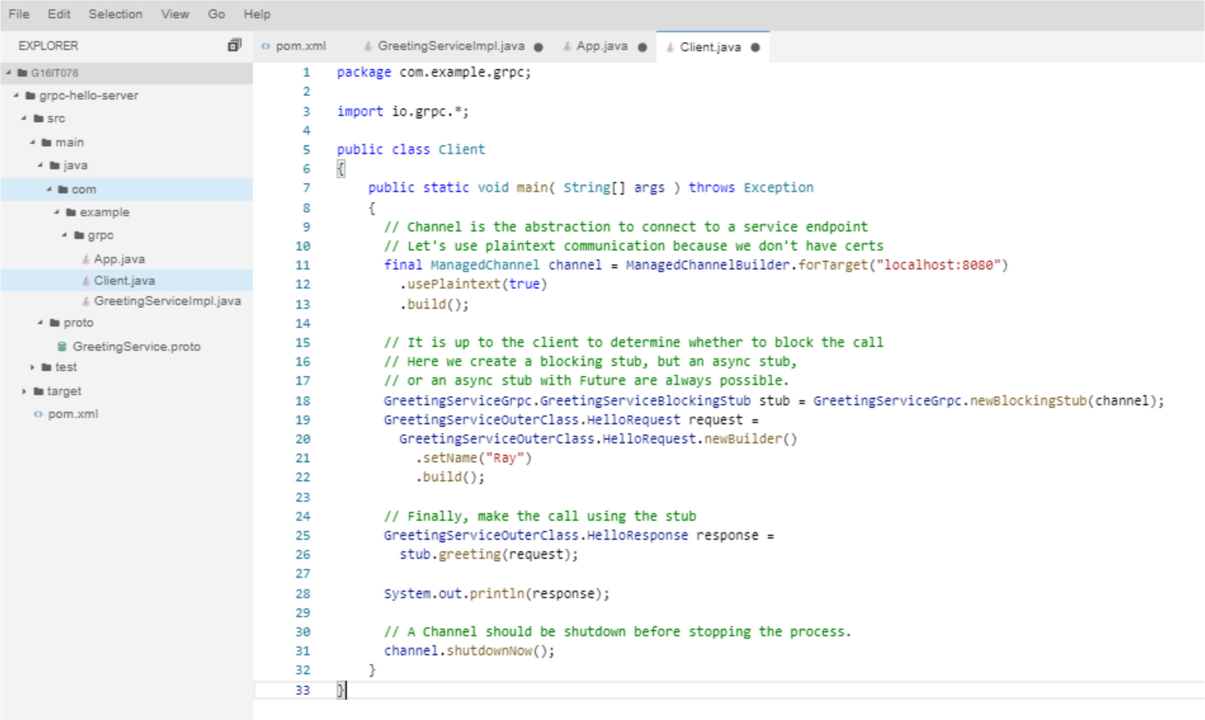
```
g16it078@cloudshell:~/grpc-hello-server (grpc-254504)$ mvn -DskipTests package exec:java -Dexec.mainClass=com.example.grpc.App
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Detecting the operating system and CPU architecture
[INFO] ------------------------------------------------------------------------
[INFO] os.detected.name: linux
[INFO] os.detected.arch: x86_64
[INFO] os.detected.version: 4.19
[INFO] os.detected.version.major: 4
[INFO] os.detected.version.minor: 19
[INFO] os.detected.release: debian
[INFO] os.detected.release.version: 9
[INFO] os.detected.release.like.debian: true
[INFO] os.detected.classifier: linux-x86_64
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/mojo/exec-maven-plugin/maven-metadata.xml
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/mojo/exec-maven-plugin/maven-metadata.xml (741 B at 468 B/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/mojo/exec-maven-plugin/1.6.0/exec-maven-plugin-1.6.0.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/mojo/exec-maven-plugin/1.6.0/exec-maven-plugin-1.6.0.pom (13 kB at 33 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/mojo/mojo-parent/40/mojo-parent-40.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/mojo/mojo-parent/40/mojo-parent-40.pom (34 kB at 65 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/mojo/exec-maven-plugin/1.6.0/exec-maven-plugin-1.6.0.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/mojo/exec-maven-plugin/1.6.0/exec-maven-plugin-1.6.0.jar (58 kB at 113 kB/s)
[INFO]
[INFO] ----------------< com.example.grpc:grpc-hello-server >-----------------
```

Figure 6 Start Server



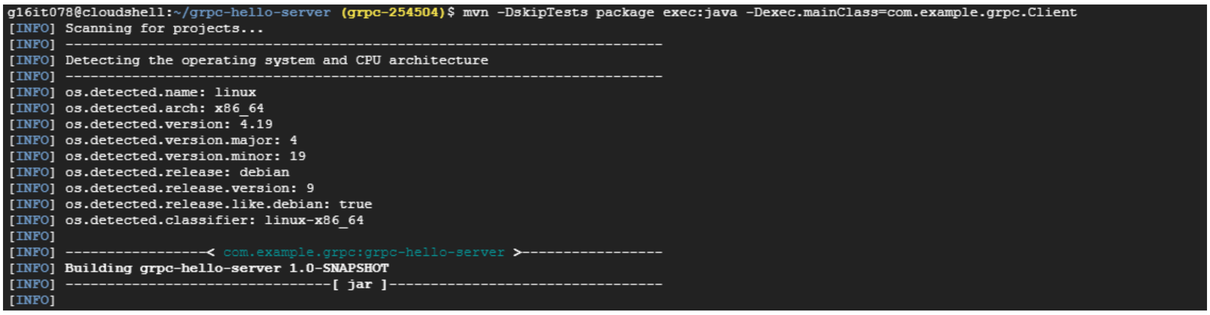
```
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-plugin-descriptor/2.2.1/maven-plugin-descriptor-2.2.1.jar
Downloaded from central: https://repo.maven.apache.org/maven2/backport-util-concurrent/backport-util-concurrent/3.1/backport-util-concurrent-3.1.jar (332 kB at 151 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-monitor/2.2.1/maven-monitor-2.2.1.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-logging-api/1.1/doxia-logging-api-1.1.jar (11 kB at 5.0 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-plugin-api/2.2.1/maven-plugin-api-2.2.1.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-error-diagnostics/2.2.1/maven-error-diagnostics-2.2.1.jar (13 kB at 5.2 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0.20/plexus-utils-3.0.20.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-plugin-descriptor/2.2.1/maven-plugin-descriptor-2.2.1.jar (39 kB at 16 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-exec/1.3/commons-exec-1.3.jar
Downloaded from central: https://repo.maven.apache.org/maven2/commons-cli/commons-cli/1.2/commons-cli-1.2.jar (41 kB at 16 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-monitor/2.2.1/maven-monitor-2.2.1.jar (10 kB at 4.1 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-plugin-api/2.2.1/maven-plugin-api-2.2.1.jar (12 kB at 4.6 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-exec/1.3/commons-exec-1.3.jar (54 kB at 19 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0.20/plexus-utils-3.0.20.jar (243 kB at 82 kB/s)
Server started
```

Figure 7 Server Started successfully

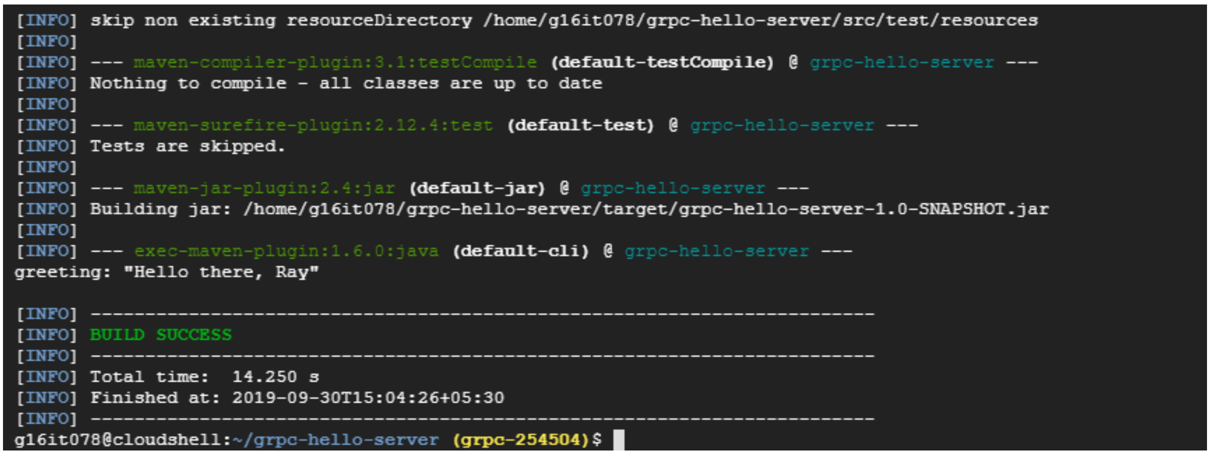


```java
package com.example.grpc;

import io.grpc.*;

public class Client
{
    public static void main( String[] args ) throws Exception
    {
      // Channel is the abstraction to connect to a service endpoint
      // Let's use plaintext communication because we don't have certs
      final ManagedChannel channel = ManagedChannelBuilder.forTarget("localhost:8080")
        .usePlaintext(true)
        .build();

      // It is up to the client to determine whether to block the call
      // Here we create a blocking stub, but an async stub,
      // or an async stub with Future are always possible.
      GreetingServiceGrpc.GreetingServiceBlockingStub stub = GreetingServiceGrpc.newBlockingStub(channel);
      GreetingServiceOuterClass.HelloRequest request =
        GreetingServiceOuterClass.HelloRequest.newBuilder()
          .setName("Ray")
          .build();

      // Finally, make the call using the stub
      GreetingServiceOuterClass.HelloResponse response =
        stub.greeting(request);

      System.out.println(response);

      // A Channel should be shutdown before stopping the process.
      channel.shutdownNow();
    }
}
```

Figure 8 Create Client App

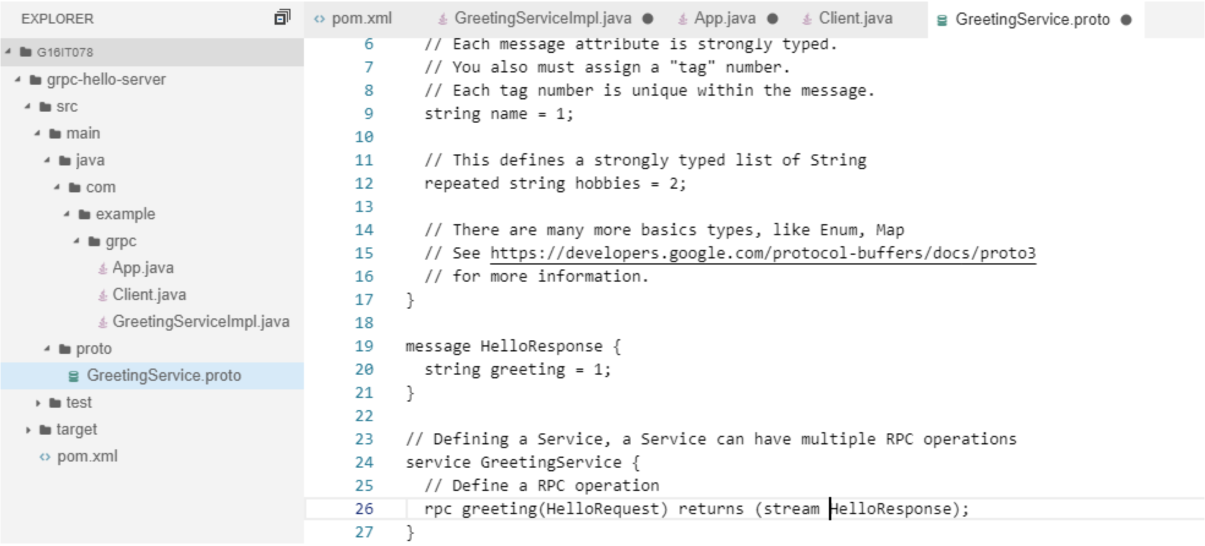


```
g16it078@cloudshell:~/grpc-hello-server (grpc-254504)$ mvn -DskipTests package exec:java -Dexec.mainClass=com.example.grpc.Client
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Detecting the operating system and CPU architecture
[INFO] ------------------------------------------------------------------------
[INFO] os.detected.name: linux
[INFO] os.detected.arch: x86_64
[INFO] os.detected.version: 4.19
[INFO] os.detected.version.major: 4
[INFO] os.detected.version.minor: 19
[INFO] os.detected.release: debian
[INFO] os.detected.release.version: 9
[INFO] os.detected.release.like.debian: true
[INFO] os.detected.classifier: linux-x86_64
[INFO]
[INFO] -----------------< com.example.grpc:grpc-hello-server >------------------
[INFO] Building grpc-hello-server 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
```

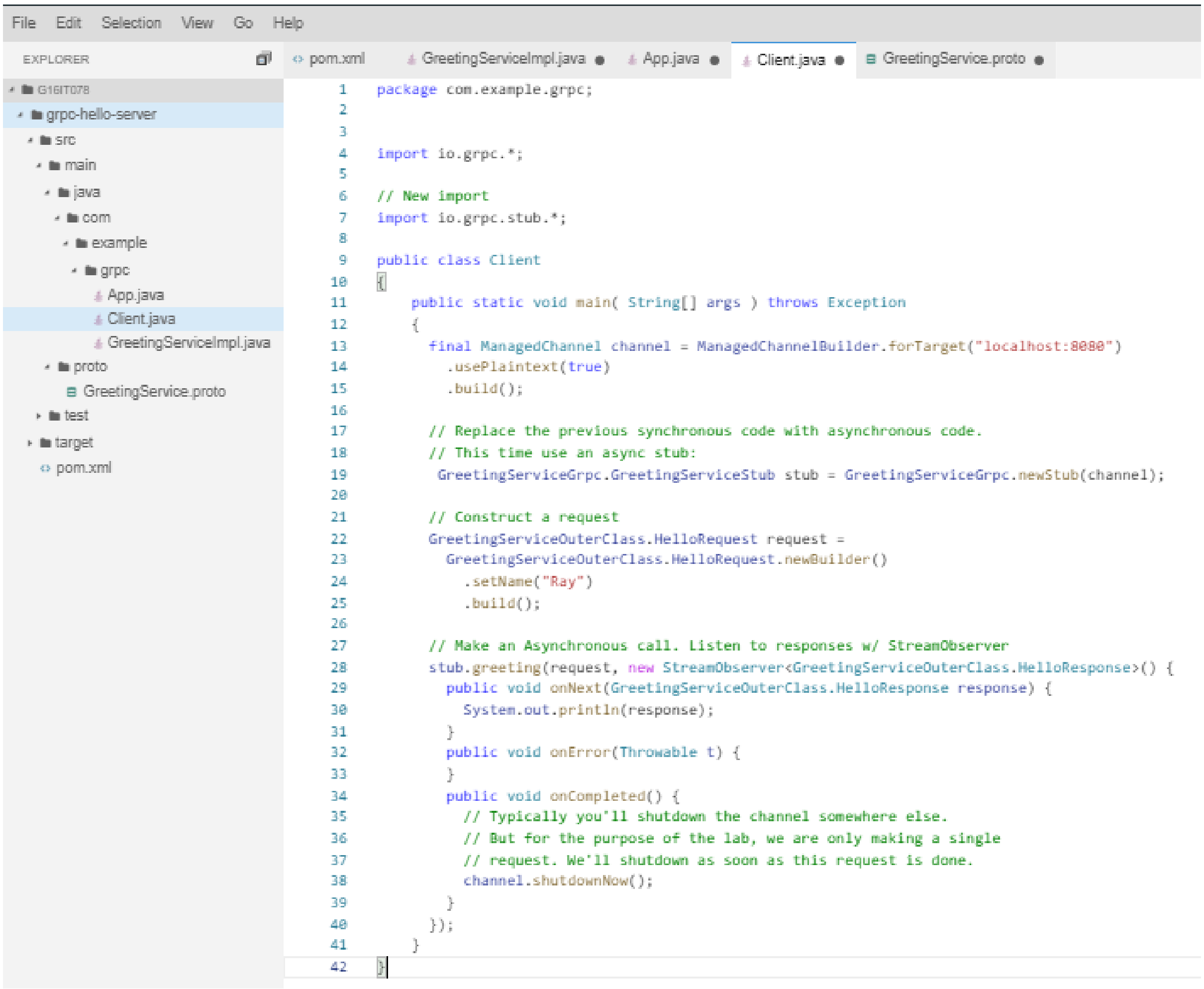


```
[INFO] skip non existing resourceDirectory /home/g16it078/grpc-hello-server/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ grpc-hello-server ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ grpc-hello-server ---
[INFO] Tests are skipped.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ grpc-hello-server ---
[INFO] Building jar: /home/g16it078/grpc-hello-server/target/grpc-hello-server-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ grpc-hello-server ---
greeting: "Hello there, Ray"

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  14.250 s
[INFO] Finished at: 2019-09-30T15:04:26+05:30
[INFO] ------------------------------------------------------------------------
g16it078@cloudshell:~/grpc-hello-server (grpc-254504)$
```

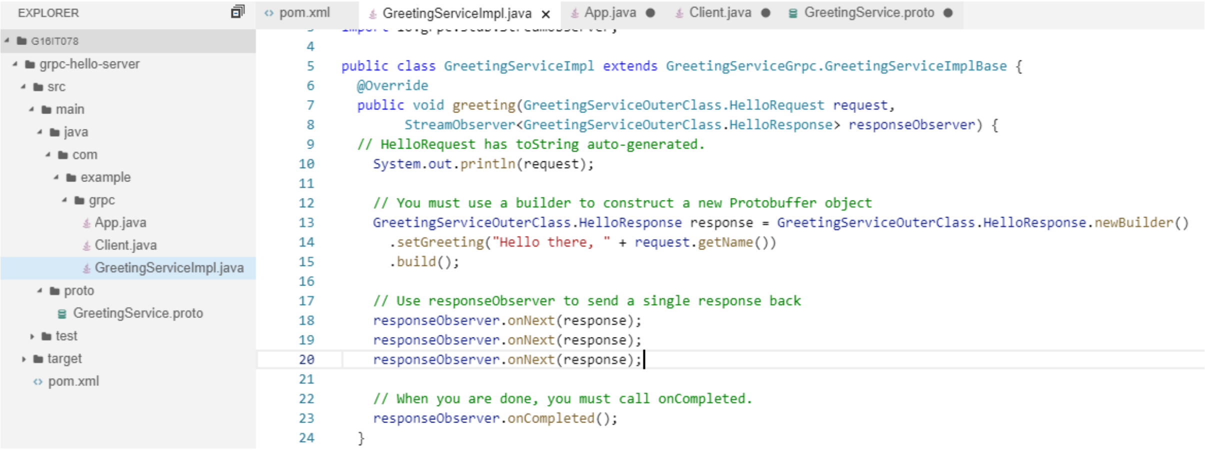
Figure 9 Output of application



```
  // Each message attribute is strongly typed.
  // You also must assign a "tag" number.
  // Each tag number is unique within the message.
  string name = 1;

  // This defines a strongly typed list of String
  repeated string hobbies = 2;

  // There are many more basics types, like Enum, Map
  // See https://developers.google.com/protocol-buffers/docs/proto3
  // for more information.
}

message HelloResponse {
  string greeting = 1;
}

// Defining a Service, a Service can have multiple RPC operations
service GreetingService {
  // Define a RPC operation
  rpc greeting(HelloRequest) returns (stream HelloResponse);
}
```

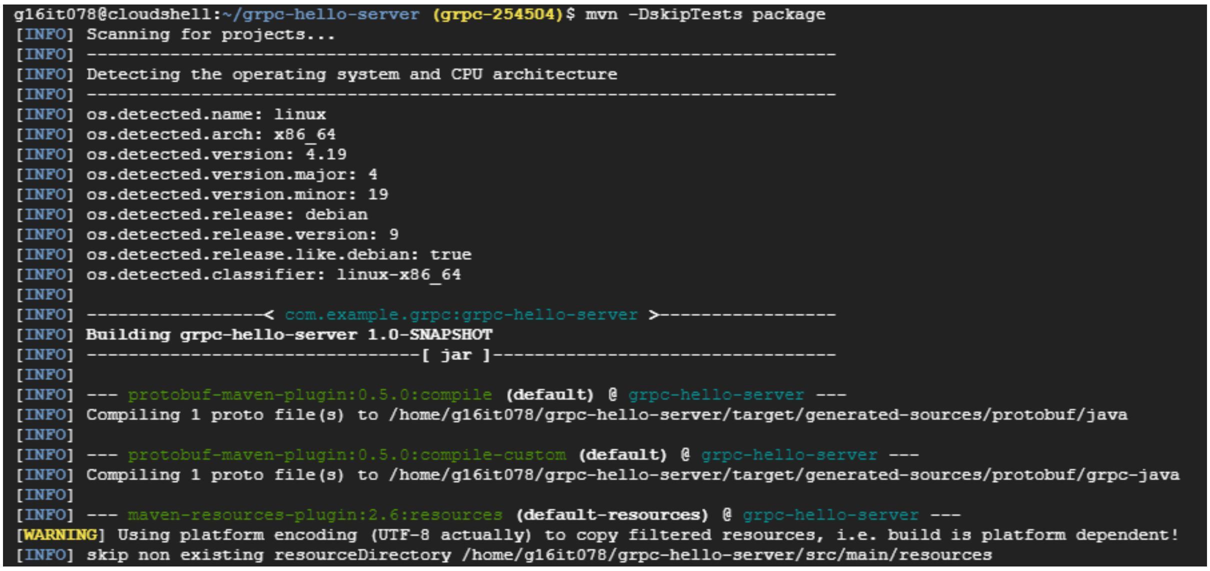


```
package com.example.grpc;


import io.grpc.*;

// New import
import io.grpc.stub.*;

public class Client
{
    public static void main( String[] args ) throws Exception
    {
      final ManagedChannel channel = ManagedChannelBuilder.forTarget("localhost:8080")
        .usePlaintext(true)
        .build();

      // Replace the previous synchronous code with asynchronous code.
      // This time use an async stub:
       GreetingServiceGrpc.GreetingServiceStub stub = GreetingServiceGrpc.newStub(channel);

      // Construct a request
      GreetingServiceOuterClass.HelloRequest request =
        GreetingServiceOuterClass.HelloRequest.newBuilder()
          .setName("Ray")
          .build();

      // Make an Asynchronous call. Listen to responses w/ StreamObserver
      stub.greeting(request, new StreamObserver<GreetingServiceOuterClass.HelloResponse>() {
        public void onNext(GreetingServiceOuterClass.HelloResponse response) {
          System.out.println(response);
        }
        public void onError(Throwable t) {
        }
        public void onCompleted() {
          // Typically you'll shutdown the channel somewhere else.
          // But for the purpose of the lab, we are only making a single
          // request. We'll shutdown as soon as this request is done.
          channel.shutdownNow();
        }
      });
    }
}
```

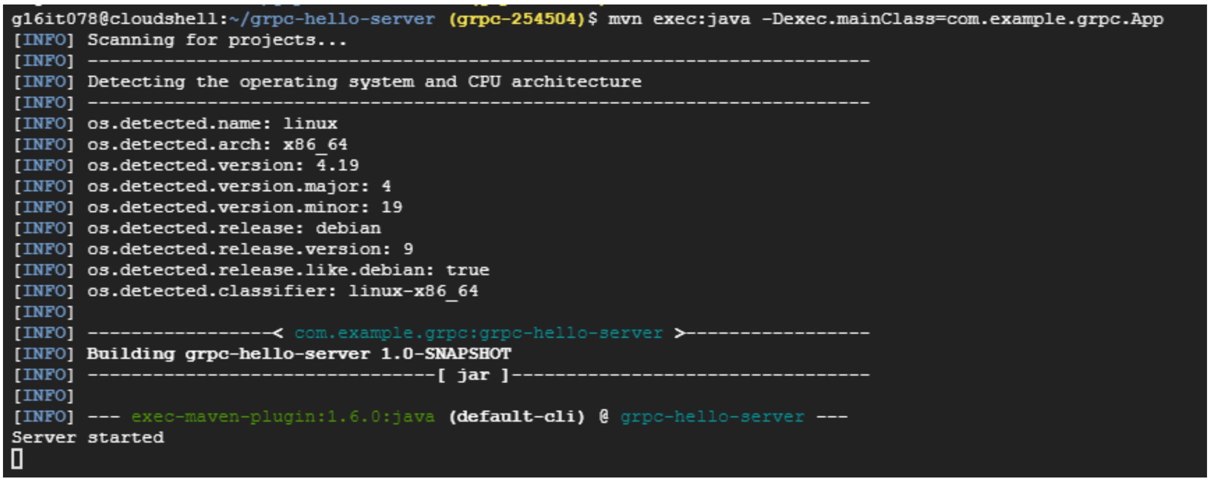


```java

public class GreetingServiceImpl extends GreetingServiceGrpc.GreetingServiceImplBase {
  @Override
  public void greeting(GreetingServiceOuterClass.HelloRequest request,
        StreamObserver<GreetingServiceOuterClass.HelloResponse> responseObserver) {
  // HelloRequest has toString auto-generated.
    System.out.println(request);

    // You must use a builder to construct a new Protobuffer object
    GreetingServiceOuterClass.HelloResponse response = GreetingServiceOuterClass.HelloResponse.newBuilder()
      .setGreeting("Hello there, " + request.getName())
      .build();

    // Use responseObserver to send a single response back
    responseObserver.onNext(response);
    responseObserver.onNext(response);
    responseObserver.onNext(response);

    // When you are done, you must call onCompleted.
    responseObserver.onCompleted();
  }
```

Figure 10 Edit sevice file for different output

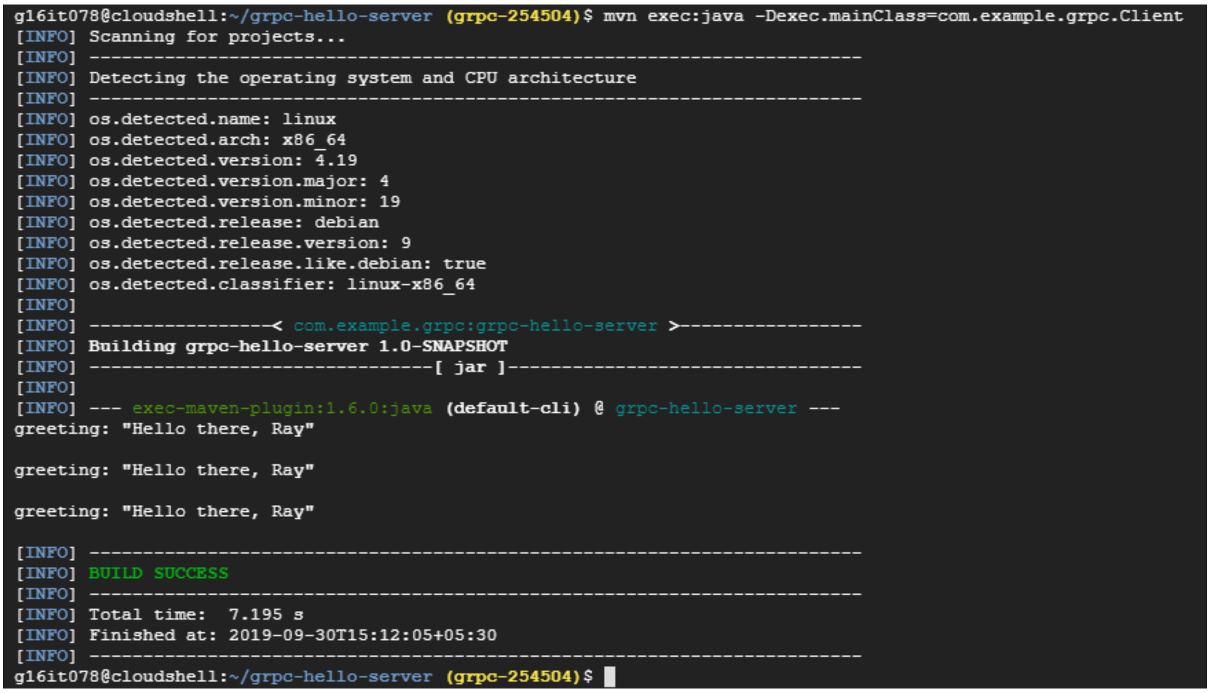


```
g16it078@cloudshell:~/grpc-hello-server (grpc-254504)$ mvn -DskipTests package
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Detecting the operating system and CPU architecture
[INFO] ------------------------------------------------------------------------
[INFO] os.detected.name: linux
[INFO] os.detected.arch: x86_64
[INFO] os.detected.version: 4.19
[INFO] os.detected.version.major: 4
[INFO] os.detected.version.minor: 19
[INFO] os.detected.release: debian
[INFO] os.detected.release.version: 9
[INFO] os.detected.release.like.debian: true
[INFO] os.detected.classifier: linux-x86_64
[INFO]
[INFO] ----------------< com.example.grpc:grpc-hello-server >----------------
[INFO] Building grpc-hello-server 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- protobuf-maven-plugin:0.5.0:compile (default) @ grpc-hello-server ---
[INFO] Compiling 1 proto file(s) to /home/g16it078/grpc-hello-server/target/generated-sources/protobuf/java
[INFO]
[INFO] --- protobuf-maven-plugin:0.5.0:compile-custom (default) @ grpc-hello-server ---
[INFO] Compiling 1 proto file(s) to /home/g16it078/grpc-hello-server/target/generated-sources/protobuf/grpc-java
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ grpc-hello-server ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/g16it078/grpc-hello-server/src/main/resources
```



```
g16it078@cloudshell:~/grpc-hello-server (grpc-254504)$ mvn exec:java -Dexec.mainClass=com.example.grpc.App
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Detecting the operating system and CPU architecture
[INFO] ------------------------------------------------------------------------
[INFO] os.detected.name: linux
[INFO] os.detected.arch: x86_64
[INFO] os.detected.version: 4.19
[INFO] os.detected.version.major: 4
[INFO] os.detected.version.minor: 19
[INFO] os.detected.release: debian
[INFO] os.detected.release.version: 9
[INFO] os.detected.release.like.debian: true
[INFO] os.detected.classifier: linux-x86_64
[INFO]
[INFO] ----------------< com.example.grpc:grpc-hello-server >----------------
[INFO] Building grpc-hello-server 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ grpc-hello-server ---
Server started
```

Figure 11 restart sever



```
g16it078@cloudshell:~/grpc-hello-server (grpc-254504)$ mvn exec:java -Dexec.mainClass=com.example.grpc.Client
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Detecting the operating system and CPU architecture
[INFO] ------------------------------------------------------------------------
[INFO] os.detected.name: linux
[INFO] os.detected.arch: x86_64
[INFO] os.detected.version: 4.19
[INFO] os.detected.version.major: 4
[INFO] os.detected.version.minor: 19
[INFO] os.detected.release: debian
[INFO] os.detected.release.version: 9
[INFO] os.detected.release.like.debian: true
[INFO] os.detected.classifier: linux-x86_64
[INFO]
[INFO] ----------------< com.example.grpc:grpc-hello-server >-----------------
[INFO] Building grpc-hello-server 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ grpc-hello-server ---
greeting: "Hello there, Ray"

greeting: "Hello there, Ray"

greeting: "Hello there, Ray"

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  7.195 s
[INFO] Finished at: 2019-09-30T15:12:05+05:30
[INFO] ------------------------------------------------------------------------
g16it078@cloudshell:~/grpc-hello-server (grpc-254504)$
```

Figure 12 Final Output

Conclusion:

In this practical we have studied about gRPC framework and understood and implemented on Google Cloud and and java.