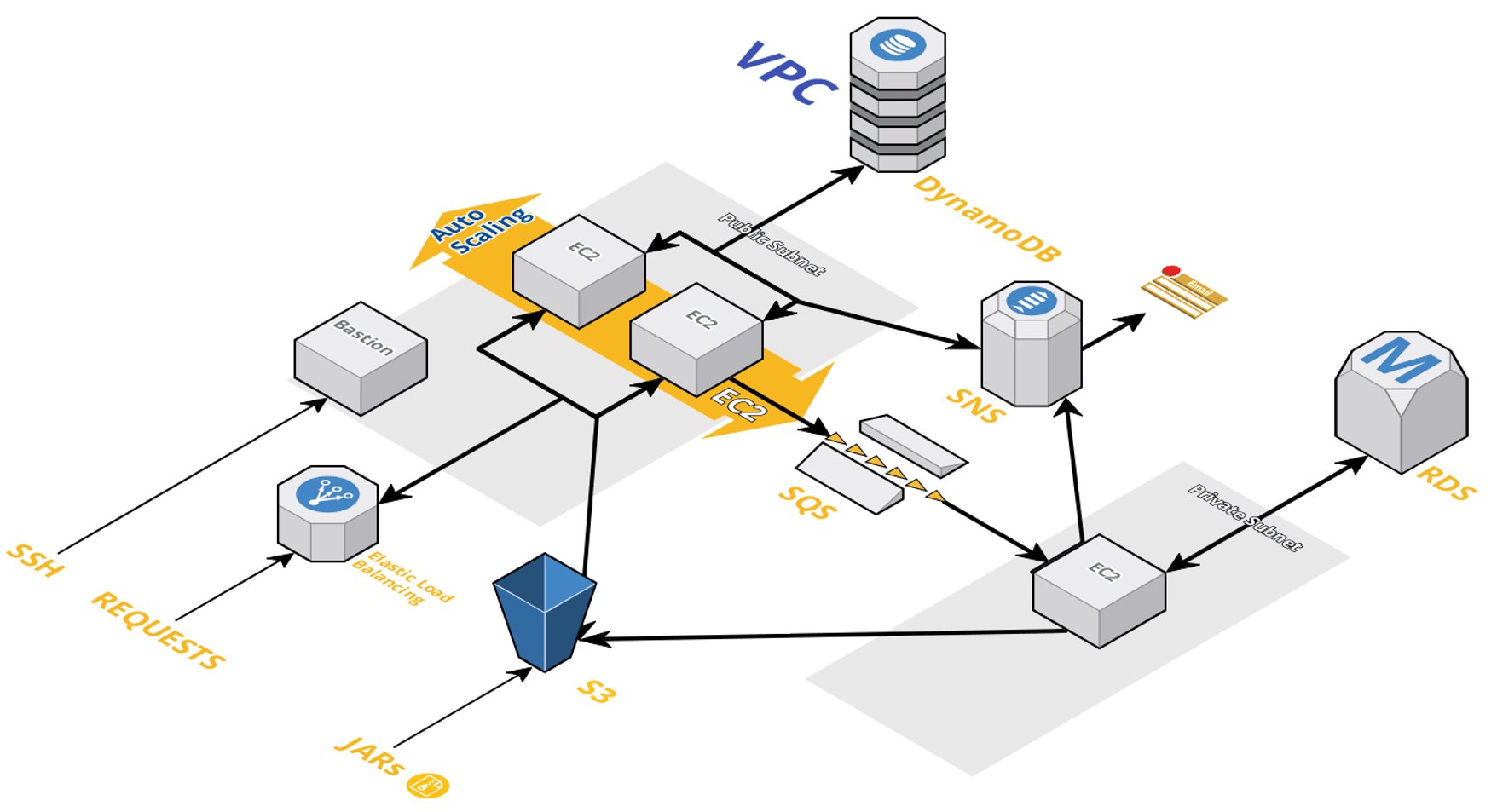
**Final Task requirements:**



General infrastructure diagram (with few discrepancies, exact infrastructure is described below)

* Create a **Terraform script** which will generate and deploy next infrastructure:
  + VPC with four subnets (two public, two private).
  + ASG with minimum two EC2 instances (one EC2 instance per each public subnet). EC2 instances in public subnets should have SSH/HTTP/HTTPS access from all IPs.
  + NAT EC2 instance (aka Bastion) in public subnet.
  + EC2 instance in one of the private subnets. Private EC2 should have SSH access from public subnets only. Implement ability to download software from the internet using created NAT EC2 instance.
  + All EC2 instances (except of NAT EC2) both public/private should have java8 installed during instance creation.
  + DynamoDB database with next parameters (EC2 instances should have permissions to access DynamoDB table):
    - TableName: ***edu-lohika-training-aws-dynamodb***
    - Field: UserName, type: String
  + Postgres RDS with next parameters (only private subnets should have access to the RDS instance) Please not that you need to create RDS instance in VPC (in private subnets).
    - DBName: ***EduLohikaTrainingAwsRds***
    - Port: ***5432***
    - User/password: ***rootuser/rootuser***
    - (you don’t need to define a schema; it will auto generating)
  + SNS topic with the next name – ***edu-lohika-training-aws-sns-topic***
  + SQS queue with the next name – ***edu-lohika-training-aws-sqs-queue***
  + LoadBalancer with 80 port targeting to public EC2 instances.
  + Health check for ELB: ***/health***
  + Run into public subnet:
    - calc-0.0.1-SNAPSHOT.jar
  + Run into private subnet:
    - persist3-0.0.1-SNAPSHOT.jar
    - set environment variable RDS\_HOST with correct RDS address
  + On your local machine, you need to have java 8.
  + On your local machine execute:
    - java -cp calc-client-1.0-SNAPSHOT-jar-with-dependencies.jar CalcClient <ELB’s DNS name>

**How to check that everything working as expected.**

\* Deploy infrastructure to AWS (`terraform apply`)

\* Go to AWS Management console and manually create e-mail subscribtion to newly created SNS topic.

\* Confirm subscription

\* Run client application on your local machine. Console output should looks like (`name` will differ):

```

ODL1610003:OneDrive\_1\_1-27-2021 rgederin$ java -cp calc-client-1.0-SNAPSHOT-jawith-dependencies.jar CalcClient load-balancer-519852589.us-west-2.elb.amazonaws.com

Request: name=1786K, 65 + 41

Response: result=106, datetime=2021-02-05T11:55:51.202, ip=52.89.145.154

Request: name=1786K, 89 + 89

Response: result=178, datetime=2021-02-05T11:55:53.624, ip=52.12.64.107

Request: name=1786K, 87 + 89

Response: result=176, datetime=2021-02-05T11:55:55.900, ip=52.89.145.154

Request: name=1786K, 71 + 15

Response: result=86, datetime=2021-02-05T11:55:58.400, ip=52.89.145.154

Request: name=1786K, 61 + 2

Response: result=63, datetime=2021-02-05T11:56:00.689, ip=52.12.64.107

```

\* After starting the application you will recieve periodically e-mail notifications from applications running on public and private EC2s

```

AWS Notifications <no-reply@sns.amazonaws.com>

From calc: 1786K

```

```

AWS Notifications <no-reply@sns.amazonaws.com>

From persist: 20

```

\* Go to EC2 dashboard and terminate one of the instance from public subnet (not NAT EC2!). After this calculation responses should come only from one working EC2 insatnce. Everything else should be the same:

```

Request: name=AZSAS, 48 + 11

Response: result=59, datetime=2021-02-05T11:59:46.922, ip=52.12.64.107

Request: name=AZSAS, 24 + 68

Response: result=92, datetime=2021-02-05T11:59:49.400, ip=52.12.64.107

Request: name=AZSAS, 57 + 87

Response: result=144, datetime=2021-02-05T11:59:51.899, ip=52.12.64.107

Request: name=AZSAS, 12 + 14

Response: result=26, datetime=2021-02-05T11:59:54.208, ip=52.12.64.107

Request: name=AZSAS, 32 + 77

Response: result=109, datetime=2021-02-05T11:59:56.692, ip=52.12.64.107

```

\* After some time ASG will recreate EC2 instance and it should automatically load balanced by ELB (without stopping client application)

```

Request: name=SH6SW, 14 + 53

Response: result=67, datetime=2021-02-05T12:03:43.147, ip=52.12.64.107

Request: name=SH6SW, 88 + 39

Response: result=127, datetime=2021-02-05T12:03:46.840, ip=34.213.185.228

Request: name=SH6SW, 46 + 48

Response: result=94, datetime=2021-02-05T12:03:49.489, ip=34.213.185.228

Request: name=SH6SW, 22 + 65

Response: result=87, datetime=2021-02-05T12:03:51.828, ip=52.12.64.107

Request: name=SH6SW, 76 + 5

Response: result=81, datetime=2021-02-05T12:03:54.131, ip=34.213.185.228

```

\* SSH to one of the public EC2 instances and check DynamoDB table using AWS CLI commands:

Output should be like:

```{

"Count": 8,

"Items": [

{

"UserName": {

"S": "SH6SW"

}

},

{

"UserName": {

"S": "5Y5NN"

}

},

{

"UserName": {

"S": "M5QJD"

}

},

{

"UserName": {

"S": "7J0GN"

}

},

{

"UserName": {

"S": "TXREG"

}

},

{

"UserName": {

"S": "1786K"

}

},

{

"UserName": {

"S": "AZSAS"

}

},

{

"UserName": {

"S": "1UYVM"

}

}

],

"ScannedCount": 8,

"ConsumedCapacity": null

}

[ec2-user@ip-10-0-1-118 ~]$

```

\* SSH from public EC2 to private EC2 (using its private IP) and check entries in RDS table (table name is "LOGS"):

Output should be like:

```

Password for user rootuser:

line

----------------------------------------------------

91+12

13+36

35+46

90+31

65+41

89+89

87+89

69+50

78+52

6+31

48+79

65+47

27+2

47+93

66+94

89+30

96+53

86+35

24+30

62+72

6+33

67+50

0+0

56+40

--More--

```