API Example

The aim of this example is to demonstrate a three-tier web application that consumes APIs that uses a database that is in a private network in a VPC architecture. The web server is in the public network and has access to the database in the private network. Following are the steps to implement this example:

Create the VPC architecture

1. Create a VPC – 'choose VPC and more'
2. Provide a suitable name or leave it with the default – project
3. Provide a IPv4 CIDR block 10.0.0.0/16 (or any other)
4. Tenancy - Default
5. For this example, one Availability Zone (AZ) with one public and private subnet is sufficient
6. Choose NAT gateway '1 per AZ' - instances in the private network to have internet access to install packages / apps / patching etc.
7. We do not need a VPC end point since we are not using S3.

Routing tables

1. Verify that the routing table for the private network has local access (10.0.0.0/16) and to the NAT gateway (0.0.0.0/0)
2. Verify that the routing table for the public network has local access (10.0.0.0/16) and to the Internet Gateway

Network ACL

1. Leave it with the default settings

Bastion Host / Web server

1. Create an Ubuntu instance in the public network within the VPC that you created, let it have its own key pair and security group.
2. This instance can act as the Bastion host to access instances in the private network.
3. In the security group add SSH, TCP ports 8000 and 3000 with 'Anywhere' access.
4. TCP port 8000 is for API access, this can be removed after testing the APIs.

Database server

1. Create an Ubuntu instance in the private network within the VPC that you created, let it have its own key pair and security group.
2. You will access this instance from the Bastion host.
3. In the security group add SSH and TCP port 3306, both with source as the security group of the Bastion host.

Install MySQL

1. SSH / Putty into Bastion host
2. Copy DB server key – you can copy paste
3. Change permission of key to '400' – chmod 400 db_server_key.pem
4. SSH to DB Server –

```
ssh -i DBServer-Key.pem ubuntu@10.0.133.111
```

5. Use the following steps to install MySQL and to setup the database:

```
sudo apt update
sudo apt upgrade
sudo apt install mysql-server
sudo systemctl start mysql
sudo mysql -u root
```

6. Once you are in MySQL you can create a database and a user with access to this database:

```
mysql> CREATE DATABASE tododb;
GRANT ALL PRIVILEGES ON `tododb`.* TO 'elan'@'%';
mysql> FLUSH PRIVILEGES;
```

7. Next, access MySQL configuration file to change bind-address:

```
sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf
```

8. Change bind-address to private IP address of DB Server:

```
bind-address            = 10.0.133.111
mysqlx-bind-address     = 10.0.133.111
```

9. Restart MySQL server

```
sudo systemctl restart mysql
```

10. Exit from the DB Server

11. Now you should be able to access the DB Server from the Bastion host. First install `mysql-client` on the Bastion host

```
sudo apt update

sudo apt upgrade

sudo apt install mysql-client

mysql -u elan -h 10.0.133.111 -p
```

Now, we are ready to create the backend (Django) and then the frontend (React) for our simple example. Please follow the instructions in the following link:

`https://www.geeksforgeeks.org/integrating-django-with-reactjs-using-django-rest-framework/`

The instructions are mostly applicable straightaway, however a few a tweaking is to be done:

1. Check the current Node version

```
nodejs -v
```

2. If its below version 14, install Node version 14.0 or above

```
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -

sudo apt install nodejs

sudo apt install build-essential
```

3. In **Step 1** the command should be

```
mkdir django-react-project
```

4. In Step 4 you may have to install the 'venv', please check the error message and follow instructions.

5. In **Step 9**, before executing 'migration' edit the settings file to refer to our MySQL database server as follows:

```
vi backend/setting.py
```

```
DATABASES = {
    'default': {
        # 'ENGINE': 'django.db.backends.sqlite3',
        # 'NAME': BASE_DIR / 'db.sqlite3',

        'ENGINE'  : 'django.db.backends.mysql',
        'NAME'    : 'tododb',
        'USER'    : 'elan',
        'PASSWORD': 'gc1t#123',
        'HOST'    : '10.0.133.111',
        'PORT'    : '3306',
    }
}
```

I have commented the old settings and added the new settings.

6. In **Step 10,** before executing 'runserver', edit the settings file to change ALLOWED HOSTS:

ALLOWED_HOSTS = ['52.63.28.69']

This should be the public IP of your App server (Bastion Host), now you can execute the 'runserver' command.

```
python manage.py runserver 10.0.6.219:8000
```

Here we use the private IP of the App server (Bastion Host), port 8000 has been opened in the security group.

7. In rest of the instructions, make sure to use the appropriate IP addresses.