Challenge One:

AWS IAM configuration:

To access the AWS do the following steps:

1. in the terminal write: pip3 install awscli

2. in the terminal write: aws configure

In order to configure use the **new_user_credential.csv** file and fill the steps as fig. 1. (this key is sent to you via email)

```
ehsan@eh-VB:~$ aws configure --profile invincible
AWS Access Key ID [None]: AKIAR2O6V5KDVJR6UIP6
AWS Secret Access Key [None]: 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
Default region name [None]: eu-central-1
Default output format [None]: text
ehsan@eh-VB:~$
```

Fig. 1. aws configuration

Furthermore, the aws_Key.pem is also needed, as pair key, in order to connect to EC2 via SSH. By using the key if the error 0644 (bad permission or key leak) is occurred the accessing mode to the key should be changed as in fig. 2.

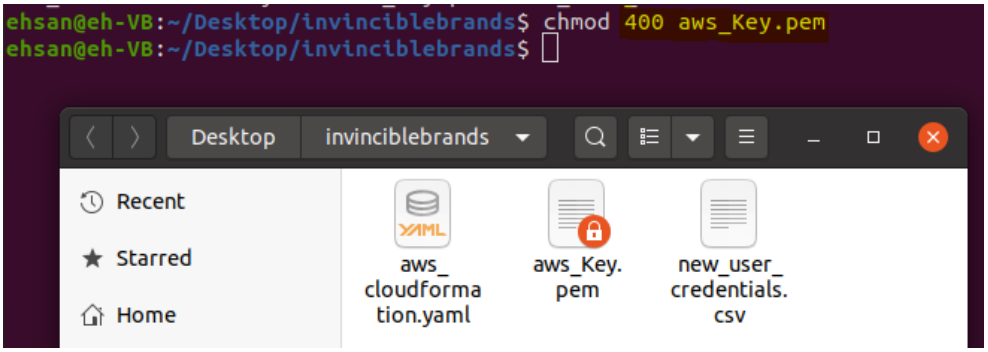

Fig. 2. Changing the pair kay mode.

After changing the mode the lock key should be appeared on the file.

To connect and create cloduformation stack the command in below can be used.

aws cloudformation create-stack --stack-name invincible-challenge --template-body file://$PWD/aws_cloudformation.yml --profile invincible --region eu-central-1 --parameters ParameterKey=KeyPairName,ParameterValue=TestKey ParameterKey=SubnetIDs,ParameterValue=SubnetID1\\,SubnetID2

finally, the stacks will be created as fig. 3.

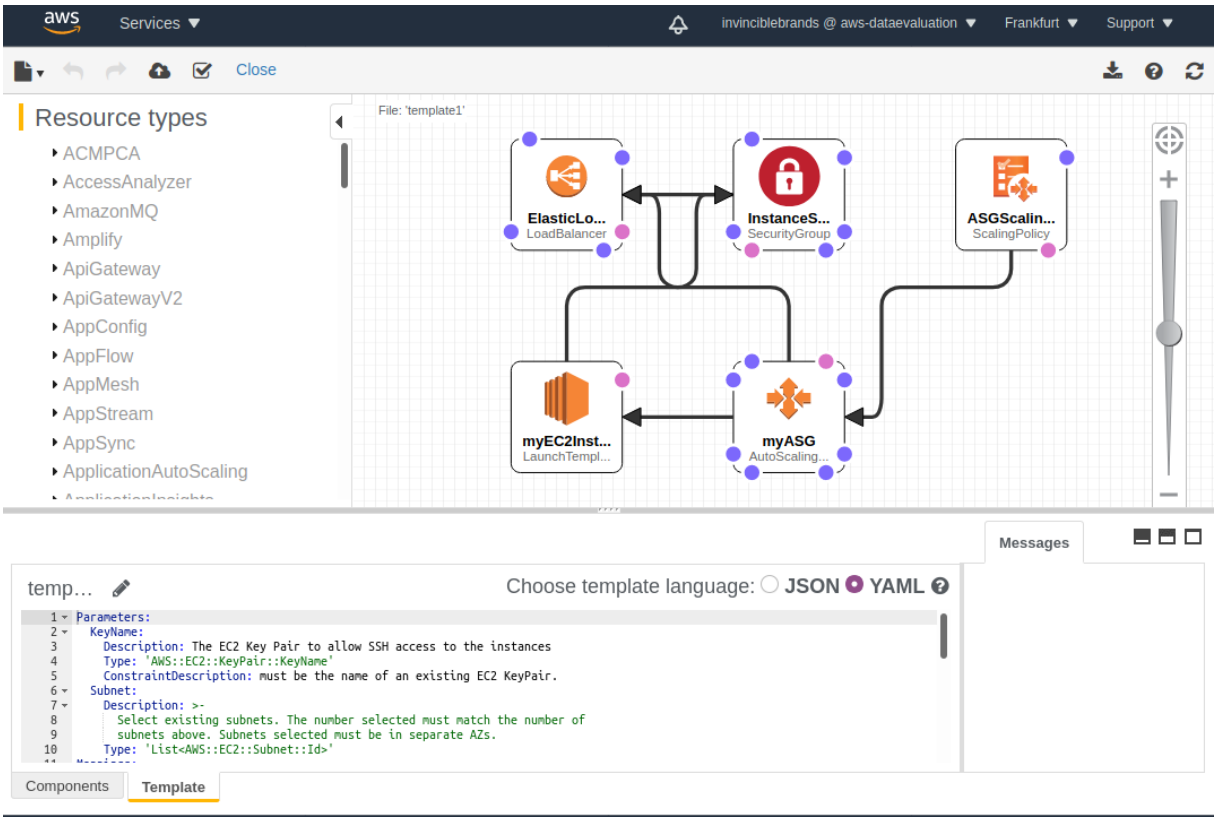

Fig. 3. Stacks created on aws.

To avoid being charge instead of using the aws cloudformation stack creation only we can use stack design environment to check whether the code is valid and also the overall picture of the stacks as fig. 3.

Challenge Two:

To cover the traffic spike the best approach that come to mind is using EC2 fleet. The Spot Fleet attempts to launch the number of Spot Instances and On-Demand Instances to meet the target capacity that you specified in the Spot Fleet request. The request for Spot Instances is fulfilled if there is available capacity and the maximum price which is specified in the request exceeds the current Spot price. The Spot Fleet also attempts to maintain its target capacity fleet if the Spot Instances are interrupted.

Moreover, it is also possible to set the maximum amount per hour base on willingness to pay for the fleet, and Spot Fleet launches instances until it reaches the maximum amount. When the maximum amount that customer is willing to pay is reached, the fleet stops launching instances even if it hasn't met the target capacity.

A Spot Instance pool is a set of unused EC2 instances with the same instance type (for example, m5.large), operating system, Availability Zone, and network platform. When customer make a Spot Fleet request, he/she can include multiple launch specifications, that vary by instance type, AMI, Availability Zone, or subnet. The Spot Fleet selects the Spot Instance pools that are used to fulfill the request, based on the launch specifications included in customers Spot Fleet request, and the configuration of the Spot Fleet request. The Spot Instances come from the selected pools. All the instances can be called via a single API. Using spot fleet with load balancer and autoscaling group can manage the traffic spikes.

Challenge Three:

ElastiCache is a service on AWS to manage and response to the applications request as fast as possible. ElastiCache has two methods, cache hit and cache miss. The Cache hit is when an application send a request and it comes from the elasticache itself. The cache miss is when the application send its request and it does not receive and response, to be precious there is no key for that data in the cache. Therefore, the data is read from RDS and then stored in elasticache. By this approach in the next time the response for this specific request will be cache hit instead if cache miss.

Redis is one of the most populate key-value data store. the Elasticcahce redis has the below features:

- Multi availability zones with auto-failover feature.
- Read replicas to scale reads, having more reads, as well as having high availability.
- Data durability using AOF persistence. (even if the cache is stopped and then restarted the data in the cache before stopping can be still available.)
- Backup and restore the Redis clusters.

However, the self managing Redis is challenging and expensive and that is why elasticache comes into the picture.

Session storage is a method used for web application such as PHP. The data is stored in web page until the user is on that page. As soon as the page is closed the data will be also lost. Some of the benefits of session storages:

- The data will not be sent to data base.
- Session storage data has the larger capacity to hold.

Amazon ElastiCache for Redis is a web service that makes it easy to deploy, operate, and scale an in-memory cache in the cloud.

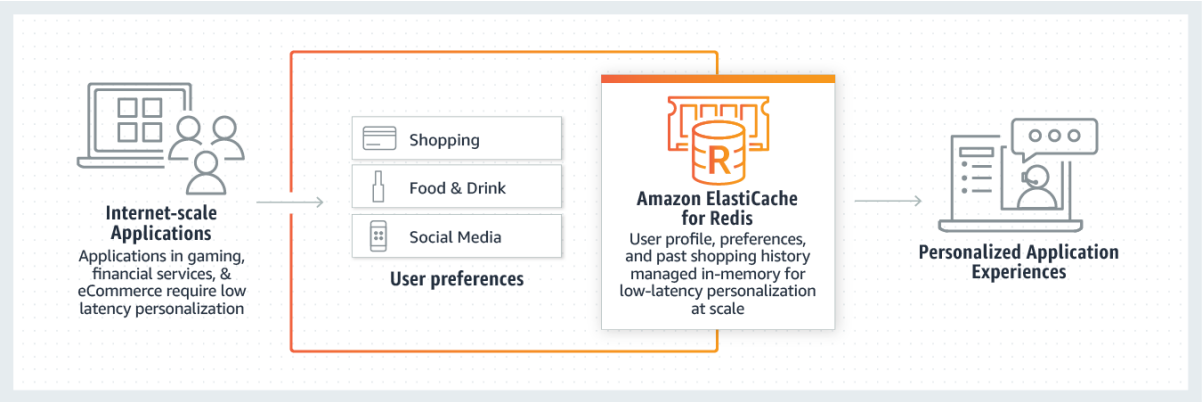

Fig. 4. Elasticache for Redis and Session store

Imagine you want to store the contents of a shopping cart in session state. The default provider stores the user's session in memory on the web server that received the request.

Using the default provider, the ELB must send every request from a specific user to the same web server. This is known as sticky sessions and greatly limits the elasticity. First, the ELB cannot distribute traffic evenly, often sending a disproportionate amount of traffic to one server. Second, Auto Scaling cannot terminate web servers without losing some user's session state. By moving the session state to

a central location, all the web servers can share a single copy of session state. This allows the ELB to send requests to any web server, better distributing load across all the web servers. In addition, Auto Scaling can terminate individual web servers without losing session state information. There are numerous providers available that allow multiple web servers to share session state. One option is use the DynamoDB Session State. This approach is expensive since the DynamoDB is charge per each transaction. Another option is to store session state in an ElastiCache cluster. Amazon ElastiCache for Redis is highly suited as a session store to manage session information such as user authentication tokens, session state, and more. Simply use ElastiCache for Redis as a fast key-value store to manage the session information. ElastiCache can be used to share PHP session information with multiple web servers and eliminate the dependency on ELB stick sessions. When you create an ElastiCache Redis cluster you can specify how many nodes you want and what subnets to place them on. After your cache is created, you can take the master node address and use it in the PHP session handler configuration. Your data will be distributed by Redis across all its nodes. When AWS needs to perform maintenance on Redis, it will migrate the master node to make sure Redis is always available.