Task #5 – EC2

What to do

Sub-task 1 – allocate EC2 resources

1. Create Linux EC2 instance (choose any free-tier eligible AMI)
2. Configure security group for the EC2 instance so that:
    a. allow access over HTTP/HTTPS from anywhere
    b. allows SSH connection from your IP address only
    c. optional task – write a script which would update the security group based on your current IP address (very useful in case you don't have a static IP address)

    A script example for Windows:

    ```
    rem .\task-3-auth-win.bat <your-profile-name> <security-group-id> <region>

    for /f %%a in ('powershell Invoke-RestMethod api.ipify.org') do set
    PublicIP=%%a

    aws ec2 authorize-security-group-ingress --group-id %2 --protocol tcp --port
    22 --cidr %PublicIP%/32 --profile %1 --region %3
    ```

    A script example for Unix:

    ```
    #!/usr/bin/env bash

    # sh .\task-3-auth-unix.sh <your-profile-name> <security-group-id> <region>

    profile=$1

    groupId=$2

    region=$3

    dig @resolver1.opendns.com ANY myip.opendns.com +short | aws ec2 authorize-
    security-group-ingress --group-id $groupId --protocol tcp --port 22 --cidr
    $1/32 --profile $profile --region $region
    ```

    d. **known pitfall** – Windows might block connections with its firewall by default
3. Make sure Apache HTTP server is installed and running on the instance. Make sure that it starts whenever the instance boot/reboot.
4. Download the static web site created in module 3 on the instance.
5. Make sure you can view the site by accessing your EC2 instance over HTTP.

Sub-task 2 – automate EC2 configuration

1. Create a new EC2 instance based on any free-tier Linux AMI and assign it the S3 readonly IAM role from module 2.
2. Configure the new EC2 instance so that it does the following steps automatically upon startup (tip – use cloud init directives and user data):
    a. install Apache HTTP server
    b. download the static web site created in module 3 from S3
3. Ensure that you can access the static web site on the EC2 instance over HTTP.
4. Create a custom AMI based on the EC2 instance.
5. Delete the EC2 instance and create another one based on the custom AMI.

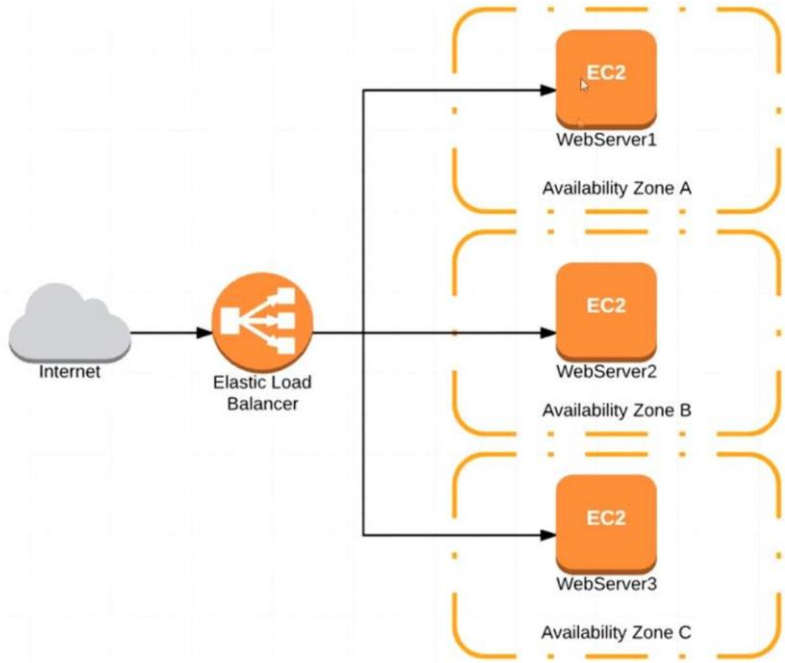
6. Make sure the web site is still available over HTTP.

Sub-task 3 – introducing EBS basics

1. Create EBS volume and attach it to the EC2 instance from the first sub-task.
2. Write any file to it and detach from the instance.
3. Attach it to the instance from the second sub-task and make sure the file is visible and accessible.

Sub-task 4 – create a load-balanced application

1. Create a simple project using your preferred language/frameworks/build tools. Feel free to customize it as you wish, but keep it simple and keep in mind the following points:
    a. any code must be hosted on EPAM's GitLab
    b. automate creation of deployable artifacts
    c. right now, there will be just one artifact – a very simple web-application (the details are below)
    d. throughout the subsequent modules, you will have to produce another artifact which will share some code with the web-application
    e. so, use something like Gradle multi-module project for Java or multi-package project for Python/NodeJS
2. In your project, create a simple web-application with one endpoint (UI page, REST API endpoint, or else) which would return the name of the region and AZ the application is running in (use this API).
3. Build your application and upload the resulting artifact (JAR, ZIP, TAR, or else) to S3.
4. Create another custom AMI based on one created in the sub-task 2:
    a. install a runtime for your web-application (Tomcat for Spring MVC, or JVM for Spring Boot, or NodeJS, or Python packages, or else)
    b. remove the Apache HTTP server in case your runtime of choice already provides an HTTP server
    c. download and deploy your web-application artifact from S3
5. Create an auto-scaling group which scales between 2-3 instances running the custom AMI. Scale out when CPU usage is more than 50%.
6. Attach an elastic load balancer to the auto-scaling group.



** Optional Task is not mandatory for completion this module but highly recommended, if you don't have a time to complete it - just skip it*

What should I remember?

1. **Once you create AWS Account -> Setup Multi-factor Authentication**
2. **Do NOT share your account**
3. **Do NOT commit your account Credentials into the Git**
4. **Terminate/Remove all created resources/services once you finishe Module**
5. **Please Do not forget to delete NAT Gateway if you used it.**
6. **Do NOT keep instance running if you don't use it**
7. **Carefully keep track of billing and working instances so you don't exceed limits**