CPD Lab 2 - IaaS (EC2) and PaaS (Elastic Beanstalk)

Introduction

This lab covers the basic concepts about EC2 and Elastic Beanstalk. For IaaS (EC2), you will create two EC2 instances with WordPress pre-installed, with a load balancer. You will also establish an SSH connection for remote login and administration of an instance.
For PaaS (Elastic Beanstalk) you will create an Elastic Beanstalk environment and investigate Auto Scaling.
The lab completion would also highlight to you the differences between creating EC2 instances and using Elastic Beanstalk environment.

Tasks:

Task 1: Create an EC2 instance
Task 2: Connect to EC2 instance using SSH
Task 3: Create another EC2 instance
Task 4: Create a Load Balancer
Task 5: Exercising the Load Balancer
Task 6: Launch a sample application on Elastic Beanstalk
Task 7: Releasing the allocated resources

Task1: Create an EC2 instance

Follow the steps listed below. If further information is required then refer to 'Link 1'under 'Links' Section at the bottom of the page:

- Select EC2 from Services in the AWS console
- Make a note of the region as that is where this instance will be created
- Click 'Launch Instance'
- Step 1: 'Choose an Instance Type',
  - You are presented with a choice of Amazon Machine Images (AMI)
  - Type 'WordPress' to search
  - Go to 'AWS Marketplace' and select

  'WordPress Certified by Bitnami and Automatic' Linux/Unix, Ubuntu 16.04 | 64-bit (x86) Amazon Machine Image (AMI) | Updated: 1/9/20
  - Click continue on the dialog box
- Step 2: Choose an Instance Type
  - Select t2.micro 'Free tier eligible', click Next
- Step 3: Configure Instance Details – click Next
- Step 4: Add Storage – click Next
- Step 5: Add Tag – For 'Key' enter Name and for Value enter 'MyInstance1' and click Next

- Step 6: Configure Security Group
  - The default settings are okay for accessing the web server over http and https ports and for SSH access through port 22. Click 'Review and Launch'
- Step 7: Review Instance Launch – review and launch the instance by clicking 'Launch'
- In the dialog box that pops up: Select an existing key pair or create a new key pair
  - Choose 'Create a new key pair'
  - Key pair name: MyKeyPair
  - Click 'Download Key pair' which downloads the MyKeyPair.pem file. This file will be used for the SSH connection in 'Task 2'
  - Click Launch Instances
- Check the newly created instance under EC2 running instances

Task 2: Connect to EC2 instance using SSH

The instructions below are for Windows. For other operating systems refer to Link 2 under Links below. You can refer to connection instructions by selecting your EC2 instance, and selecting Actions->Connect
Download PuTTY from Link 3 under 'Links' below. The downloaded file from Task 1, MyKeyPair.pem needs to be converted to MyKeyPair.ppk to be usable by PuTTY to connect to EC2 instance.

Step 1: Converting .pem file to .ppk

- Start PuTTYgen
- Select RSA at the bottom of the displayed interface.
- Click Load and browse to the folder containing the .pem file. Change the file type to open from the default .ppk to 'All files'
- Select the .pem file downloaded in Task 1 and click 'Open'
- Click OK on the displayed message box PuTTYgen notice and click 'Save private key'
- Ignore the warning regarding passphrase by clicking 'Yes'
- Use the same name for the key as used for keypair and click Save
- With the key in the correct PuTTY format we should be able to connect to the instance crested in Task 1

Step 2: Connecting to the EC2 instance

- Start PuTTY
- Under 'Category' select 'Session'
- Enter bitnami@Public-IP-Adress-Of-Instance (this is user_name@public_dns_name)

- In the ‘Category’ pane, expand ‘Connection’, expand SSH, and then click ‘Auth’
- Click ‘Browse’ to select the .ppk file and click ‘Open’
- Click ‘Yes’ to the message box PuTTY Security Alert
- A remote connection should open
- This provides full control over the EC2 instance to execute commands, install software, shutdown, reboot etc.
- To learn about the operating system, enter $cat /etc/os-release
- Close the connection

Step 3: Getting EC2 Instance Logs

- Go to the EC2 Dashboard by searching for EC2 under services
- Select the EC2 instance
- Click on ‘Actions’ and select ‘Instance Settings->Get Instance Screenshot’. The displayed information could be helpful for troubleshooting in case the PuTTY connection does not work. Click ‘Close’
- Click on Actions and select ‘Instance Settings’->Get System Log
- These logs are also helpful to identify problems
- From within the logs determine the username and password as described in https://docs.bitnami.com/aws/how-to/get-started-wordpress-aws-marketplace-beginner/ and note it down

Task 3: Create another EC2 instance

Refer to Link 4 under ‘Links’ below

- There are various ways to create another instance with the same configuration as that of an existing EC2 instance
- Select the instance MyInstance1 and from under ‘Actions’, select ‘Launch More Like This’
- This would take you to Step 7: Review Instance Launch.
- Click ‘Edit Tags’ and change the instance name to MyInstance2
- Click ‘Review and Launch’
- Click Launch
- In the dialog box, select the existing KeyPair created in Task 1, and click ‘Launch Instances’
- Navigate to Running Instances under EC2
- You will now have two instances running

Task 4: Create a Load Balancer

Now that you have two EC2 instances, you will create a load balancer for the incoming traffic. A load balancer provides a route for the incoming traffic and can distribute the traffic load between the running EC2 instances.

Hands-on:

Follow the steps below. For further details refer to Link 5 under ‘Links’ below

On the EC2 dashboard, in the left navigation pane, Click ‘Load Balancers’

- Click on ‘Create Load Balancer’
- From the shown options, click ‘Create’ under ‘Network Load Balancer’
- Step 1: Configure Load Balancer
  - Specify Name as MyNLB
  - Scheme : Internet-facing
  - Under ‘Listeners’, leave it as default TCP, Port 80
  - VPC: Select the default VPC for your account
  - Availability Zones: Select all availability zones, check eu-west-2a, eu-west-2b and eu-west-2c
  - Click Next
- Step 2: Configure Security Setting
  Click Next
- Step 3: Configure Security Groups
  Click ‘Select an existing security group’
  Select the default Security Group
  Click Next
- Step 4: Configure Routing
  Name the ‘Target Group’ as MyTG
  Target Type: Instance
  Expand ‘Advanced health check settings’ and observe different settings. Change interval to 10 seconds
  Click Next
- Step 5: Register Targets
  Under instances, Select both of your instances created in Task 2 and 4, and Click ‘Add to Registered’
  Click Next
- Step 6: Review
- Click Create
- This will take some time to be created

Task 5: Exercising the Load Balancer

- From Load Balancers on EC2 dashboard, under 'Basic Configuration' get DNS Name of MyNLB and connect to it through a browser. This should open the WordPress site.
- You would now make a small change to one of the EC2 instance so that the two instances can be recognized based on different interface language while opening them in the browser
  - Connect to EC2 instance from Task 1 using its <public-ip>/wp-admin/ through the browser
  - Connect using the username/password from Task 1, Step3
  - From Settings, change the 'User Interface Language' to any other than default (English) as described in https://docs.bitnami.com/aws/how-to/get-started-wordpress-aws-marketplace-beginner/
- From under 'Load Balancer', under Descriptions enable 'Cross-Zone Load Balancing' which would deliver incoming traffic to the two zones (each has one EC2 instance)
- Connect to the public DNS of the load balancer
- Refresh the load balancer connection through the browser and you should notice connection to each of the two EC2 instances

Task 6: Launch a Sample Application on Elastic Beanstalk

Elastic Beanstalk reduces the management complexity to deploy and manage applications on AWS cloud without concerns about the infrastructure. Elastic beanstalk automatically handles all the details of capacity provisioning, load balancing, scaling and application health monitoring.
In this task we will make use of a sample application provided by AWS and create an Elastic Beanstalk Environment.
You should follow the steps listed below. Detailed steps are under Link 6.

Hands-on
Step1: Create an example application

- On the Console search for 'Elastic Beanstalk'
- Click the 'Get Started' Button
- This opens 'Create a web app'
- Provide an application name 'MyWebApp'
- Select Platform as Python (under preconfigured) and for 'Application code' select 'Sample Application'
- Click 'Create Application'

- The environment creation will be visible on the AWS console and takes few minutes to complete
- The application version named Sample Application is created and the code is deployed to Mywebapp-env
- If the Elastic Beanstalk environment dashboard is not open already you can navigate to Elastic Beanstalk under 'Services' and click Mywebapp-env
- You can navigate to the web application URL shown which opens your web application
- Elastic Beanstalk creates the following resources: EC2 instance, Security group (firewall), S3 bucket (storage), CloudWatch alarms, CloudFormation stack, Domain name
- View the Elastic Beanstalk environment console
    - Open the Elastic Beanstalk by clicking the Elastic Beanstalk tab on left-top corner of the screen
    - Click on Mywebapp-env which opens up high level information about your environment, that is URL, health status, name of currently deployed application version, five recent events, and platform version

Step 2: Deploy a new version of your application

There might be a need during application lifetime to deploy a newer version of your application

- The application version Sample Application was used in Step 1 to create the environment
- In order to upload a new version, download python-v1.zip from https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/samples/python-v1.zip
- Navigate to Mywebapp-env and click 'Upload and Deploy'
- Click 'Choose File' and select python-v1.zip downloaded above, and click 'Deploy'
- Navigate to 'Application Versions' under the application Mywebapp
- The two versions of the application will be visible

Step 3: Configure the environment for Auto Scaling

You will configure a load balanced automatically scaling environment with two and four Amazon EC2 instances in the Auto Scaling group. An additional EC2 instance will be created by Elastic Beanstalk. The two instances are associated with the environment load balancer improving the application's responsiveness and availability

- Navigate to Mywebapp-env
- Choose 'Configuration' from the left panel
- Click 'Modify' button in front of 'Capacity'

- Under 'Auto Scaling Group', change 'Environment Type' to 'Load Balanced'
- For 'Instances' change **Min** from 1 to 2 and **Max** to 4
- At the right bottom of the page 'Modify Capacity' click 'Apply'
- Click 'Confirm' on next page displaying warning

Step 4: Triggering Auto Scaling

Terminate one of the running instances and observe that another instance gets created and replaces the terminated instance automatically

- In the navigation pane on the left, select 'Health'
- The number of instances under 'Enhanced Health Overview' is 2, that is, the minimum number of instances in the Auto Scaling group as set in Step 3
- Select one of the instances, and under 'Instance Actions', click 'Terminate'
- You should see an instance being created automatically to replace it (the min instances set as 2)

Step 5: Clean up

The environment is to be terminated and the application needs to be deleted

- Terminate the environment
    - Open the environment dashboard by choosing Mywebapp, and then choosing Mywebapp-env
    - Choose 'Actions', and then choose 'Terminate Environment'
    - Confirm that you want to terminate Mywebapp-env by typing the environment name, and then choose 'Terminate'
- Delete the Mywebapp application
    - Open the main Elastic Beanstalk dashboard by choosing Elastic Beanstalk in the upper left of the environment dashboard
    - On the Elastic Beanstalk applications page, choose 'Actions' for the Mywebapp application, and then choose Delete application
    - Confirm that you want to delete Mywebapp by typing the application name, and then choose 'Delete'

Task 6: Releasing the allocated resources

Make sure that you have released all the resources created in Tasks above:

- Terminated EC2 instances created in Task 1 and Task 3
- Deleted the load balancer created in Task 4
- Deleted the Elastic Beanstalk application and environment in Task 6

Links

The links below are for your reference only in case further information is required to help complete the 'Tasks' above:

1. Getting Started with Amazon EC2 Linux Instances:
   https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

2. Connecting to your Linux instance
   https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html
   https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html

3. Installing PuTTY:
   https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
   putty.zip contains the binary files that can be downloaded and used instead of running the msi installer.

4. Launching an Instance Using Parameters from an Existing Instance
   https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/launch-more-like-this.html

5. Application Load Balancer
   https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html

6. AWS Elastic Beanstalk
   a. https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/GettingStarted.html
   b. https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html
   c. https://docs.aws.amazon.com/elastic-beanstalk/index.html