Andrew Dass

Deployment 4 - Deploying a Flask App to Amazon Elastic Beanstalk

Amazon Elastic Beanstalk is an efficient way to deploy self-made applications such as websites. The advantage of using Amazon Elastic Beanstalk is Amazon runs the infrastructure or other programs needed to generate and run the domain or url that is provided for your website application [1]. This makes the user responsible only to configure other software programs together and upload their application onto Elastic Beanstalk.

This documentation will explain how to set up a website with Amazon Elastic Beanstalk with the following technologies:

- Python IDE (* This tutorial will use Visual Studio Code)
  - Write, edit, and save your application code locally
  - Importing packages to run functions
    - Python
    - Pip
    - Flask
  - Setting up a Python Virtual Environment
    - Purpose of Virtual Environment
    - Installing needed Packages on Virtual Environment
      - Pip
      - Flask
    - Why is it necessary to reinstall the same packages
- GitHub
  - Create a Repository
  - Place code onto the Repository
  - Generate a Token
- Linux Terminal or Windows Command Prompt
  - Upload code to Github repository
  - Connect to EC2
- Amazon AWS Services:
  - **Make an AWS Amazon Account**
  - EC2 - Elastic Compute Cloud
    - Install Git
    - Using and Installing Jenkins on the EC2
  - IAM - Identity and Access Management
  - ELB - Elastic Beanstalk

Procedure:

1. When making a website, it is recommended to download an Integrated Development Environment or for short IDE. Once again this tutorial will use Visual Studio Code. To download Visual Studio Code refer to this link: https://code.visualstudio.com/Download [2]

   Benefits of using an IDE such as Visual Studio Code:
   a. Any IDE is where you edit, save, and deploy websites from scripts you have saved in this program quickly and efficiently
   b. Visual Studio Code also gives access to every terminal from your computer to locate your files, including your application files you created to deploy your website which is necessary

**Remember, when uploading a website always make sure you do not plagiarize from any other sources*

2. To make a website with Python, you must need a framework which is software needed to run a website. In this tutorial, we will use the Flask framework.

3. To install Flask, you also need to download the latest version of Python and then Pip beforehand locally on your machine. You must download Python, then Pip, then install Flask. Refer to the guides below to download python and pip:

   To Download Python: https://www.python.org/downloads/ [3]
   To Download Pip: https://pypi.org/project/pip/ [4]

   You can download these programs through the Windows Command Prompt or Linux Terminal. Once you downloaded Python and Pip locally on your machine, then use this command to download flask:

   ```
   pip install -U Flask
   ```
   Refer here for more details if you had difficulties installing flask:
   https://pypi.org/project/Flask/ [5]

4. After installing flask, you now have the packages needed to make a website using Python and the flask framework. The next step is to create a directory where you keep all of your Flask application files for one website. A directory can be a folder that stores all your website files. For example, you can make a folder on your desktop to save all of your website files. Your python file containing the flask framework is saved as a .py file.

5. It's good to always save one application's files in separate folders because when you deploy your application to a website such as Amazon Elastic Beanstalk, you also need to download a virtual environment. The packages needed to run the website that was

downloaded onto your machine earlier stays only on your desktop or laptop, but this issue can be resolved by using a virtual environment. You can download and may have to redownload the same packages again such as pip and flask after making a virtual environment but you can import it with your application onto a Amazon Elastic Beanstalk to make it function.

6. To create a virtual environment, cd into the directory where you placed the folder containing your website files. Run the following commands on Windows:
    a. py -m venv “virtual env name” - Creates the environment. “virtual env name” is the name you give your environment
    b. “virtual env name”/Scripts/activate - After creating it, run this command to activate it. If successful, you should see the name in parenthesis in the command prompt or terminal

7. Now you can now make a website using Flask. Refer to https://pypi.org/project/Flask/ [5] for a tutorial on how to create a website.

8. After finishing your website, login or make a Github account:
   https://github.com/login [6]

   Afterwards, create a repository and upload your files into that repository. Using the terminal to upload files is recommended since the terminal automatically uploads files instantly and does not have a 100 file limit upload as opposed to the upload button on Github.

9. To use Amazon’s services, including Amazon Elastic Beanstalk, you must make an account on their website. If you do not, then you cannot complete this project. To make an account refer to this link below:
   https://aws.amazon.com/?nc2=h_lg [7]
   On their website, click on “Create an AWS Account” and follow the instructions to making an account

10. After making an AWS account, go to the Identity and Access Management Section or IAM service. The IAM service gives permissions to what services a person can use. This is good when you want to do a project with others but give them limited permissions to work with the needed resources to complete a project. Once again, some services and many options on AWS are not free and you do not want to accumulate many unknown charges onto your account.

11. To create an IAM user do the following:
    - Select Users
    - Make a User
    - Add user to group
    - Give it a name

- Access type - Programmatic Access
- For filter policies, enable “AdministratorAccess-AWSElasticBeanstalk”.

After doing so, save your access key and secret access key. You won’t be able to go back to it without resetting it, so save these two keys in a document. These keys are needed later to be integrated in Jenkins.

12. After making a IAM user, then go to the AWS Elastic Beanstalk Service and create a new web server environment with the following modifications:
    a. Fill out application name, following naming directions
    b. Select Python as platform
    c. Select Sample application

13. The Elastic Beanstalk automatically makes one EC2 instance when the program is running. However, you must make another EC2 and install the following:
    a. Jenkins - Software needed to deploy the website
    b. Install Jenkins plugins:
        i. AWSEB Deployment Plugin
        ii. CloudBees Credentials Plugin
        iii. Upgrade or change to Java 11. Perform the following steps in the Amazon EC2:
            1. systemctl stop jenkins
            2. yum install java-11
            3. update-alternatives --config java
            4. By reading the options, select the number that shows Java 11
            5. systemctl start jenkins
            6. systemctl status jenkins (to show it is running)
            7. java -version

    c. Git - Read files from Github repo

14. In Jenkins, create a new Freestyle project and give it a name. Select Git and then do the following modifications
    a. Repository Url: Copy and paste the url where you uploaded your flask files
    b. Credentials:
        i. Add Jenkins -> AWS Credentials ->
            1. Access Key ID
            2. Secret Access Key
        ii. Make sure Branch specifier is main or the name of the branch that is on Github with the flask files
    c. Scroll down to Build, insert AWS Credentials and Region. Once again, add the credentials, the aws-region of where the ec2 is and validate it
    d. Enter the application name and environment name in the boxes below and validate it as well

   e. Under packaging, place a dot in “Root Object (File/Directory)”
   f. Under versioning, write python-01${BUILD_ID}

15. Now run the build in Jenkins. It takes a minute or two minutes to run and a green check mark shows the website is successful

16. Go back to Amazon Beanstalk where you deployed your ec2 with the same application and environment name. You should be able to see your website on Amazon EC2 when you click the environment.

Sources:
What is AWS Elastic Beanstalk? - AWS Elastic Beanstalk (amazon.com) [1]

https://code.visualstudio.com/Download [2]

https://www.python.org/downloads/ [3]

https://pypi.org/project/pip/ [4]

https://pypi.org/project/Flask/ [5]

https://github.com/login [6]

https://aws.amazon.com/?nc2=h_lg [7]