**Project - 3**

**Bootstrapping An Amazon EC2 Instance Using User-data To Run A Python Web App**

**In this project, you will**

* **Create an AWS CF stack with an Amazon EC2 instance, a security group with inbound access, and an IAM instance profile.**
* **Install software packages on the EC2 instance's first launch by creating a user data asset.**
* **Configure the software packages after installation using a script downloaded by the user data.**
* **Deploy the application using user data**

**Services Used:**

* ***AWS CloudFormation***
* ***EC2 Instance***
* ***Security Group with Inbound Access***
* ***IAM Instance Profile***
* ***Python and user data***

Project Architecture Drawing

Diagram

Description automatically generated

**Project Description**

Bootstrapping an Amazon EC2 instance using user-data is a process of automatically configuring a virtual server on Amazon Web Services (AWS) and launching a Python web application on it. In this project, user-data is used to automate the configuration of the EC2 instance, including installing the necessary software and dependencies, downloading and configuring the web application, and starting the web server.

The user-data script is essentially a shell script that runs when the EC2 instance is launched. It can be used to automate many aspects of the EC2 instance, such as installing packages, configuring network settings, and running scripts.

To bootstrap the EC2 instance, the user-data script will first update the software packages and dependencies of the instance. Then, it will install Python and any required modules that the Python web application depends on. Once the dependencies are installed, the script will download the Python web application and place it in the appropriate directory.

The final step of the user-data script is to start the web server, which will serve the Python web application. This can be accomplished by running the appropriate command to start the web server, such as **python app.py**.

By using user-data to automate the configuration of the EC2 instance, this project can save time and effort, and allow for faster and more reliable deployments of Python web applications on AWS.

User Data Bootstrapped

1. **yum update -y**
2. **yum install -y httpd.x86\_64**
3. **systemctl start httpd. service**
4. **systemctl enable httpd. service**
5. **yum install python3-pip**
6. **yum install git -y**
7. **pip3 install virtualenv**
8. **echo Hello Everyone Updated Software Pakages Installed httpd Started httpd Installed python3-pip And VirtualEnv Installed Flask Installed Git for Cloning > /var/www/html/index.html**

**EC2 Instance Launched:**

**Graphical user interface, text, application

Description automatically generated**

* **Python Flask Webapp Code to Display Date & Time**

Text

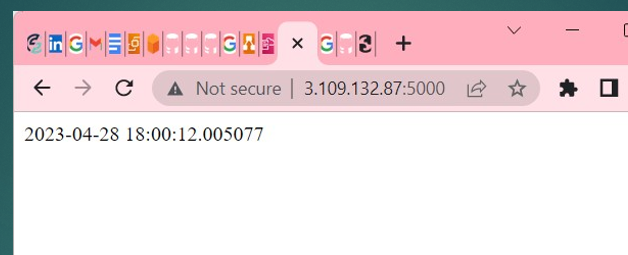
Description automatically generated

### . How to Install and Run

1. Connect EC2 Instance Thru EC2 Connect
2. mkdir app ~ To Create New App Directory
3. cd app ~ To Change to Created Directory
4. git clone < https://github.com/ipsa0926/flaskdatetime\_webapp.git> ~ To Download Flask App Code
5. cd flaskdatetime\_webapp ~ To Change to App Downloaded Directory
6. ls ~ List Out app.py & README.md Files
7. python3 -m virtualenv venv ~ To Create Virtual Environment
8. . venv/bin/activate ~ To Activate Virtual Environment
9. pip3 install Flask ~ To Install Flask
10. python3 app.py ~ To Deploy Flask Web App on Default Port No. 5000

**example:** [**http://3.109.132.87:5000**](http://3.109.132.87:5000)

**Deployed Flask Web App:**

****

### 

### Thanks, you