Table of Contents

[1 Introduction 2](#_Toc79400948)

[2 Schema model Project 2](#_Toc79400949)

[3 AWS Services - Used 2](#_Toc79400950)

[3.1 EC2 2](#_Toc79400951)

[3.2 Amazon S3 2](#_Toc79400952)

[3.3 CLOUD9 2](#_Toc79400953)

[3.4 DynamoDB 2](#_Toc79400954)

[3.5 Lambda 2](#_Toc79400955)

[3.6 NoSQL Workbench 2](#_Toc79400956)

[3.7 AirFlow 2](#_Toc79400957)

[4 Main Packages - Used 3](#_Toc79400958)

[4.1 Package: Boto3 3](#_Toc79400959)

[4.2 Package: flask\_dynamo 3](#_Toc79400960)

[4.3 HTML 3](#_Toc79400961)

[5 Files and Folders - overall 4](#_Toc79400962)

[6 How to upload from git 4](#_Toc79400963)

[7 link Git & YouTube 4](#_Toc79400964)

[8 Deploy Flask Environment 5](#_Toc79400965)

[8.1 How to Execute Python Flask File on AWS 5](#_Toc79400966)

[9 Documentation 7](#_Toc79400967)

[9.1 Deploy flask on AWS Cloud9 7](#_Toc79400968)

# Introduction

Aaaa

# Schema model Project

# AWS Services - Used

## EC2

## Amazon S3

AAA

## CLOUD9

## DynamoDB

DynamoDB is NoSQL database, as it’s fast, scalable, requires 0 maintenance, has a simple API, and is a joy to work with.

## Lambda

## NoSQL Workbench

## AirFlow

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/workbench.settingup.html

AAA

# Main Packages - Used

## Package: Boto3

**Why Using:**

Boto3 is the **name of the Python SDK for AWS**. It allows you to directly create, update, and delete AWS resources from your Python scripts.

You use the AWS SDK for Python (Boto3) to create, configure, and manage AWS services, such as Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3). The SDK provides an object-oriented API as well as low-level access to AWS services.

The SDK is composed of two key Python packages: Botocore (the library providing the low-level functionality shared between the Python SDK and the AWS CLI) and Boto3 (the package implementing the Python SDK itself).

**Documentation**:

* <https://boto3.amazonaws.com/v1/documentation/api/latest/guide/dynamodb.html>
* https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html

## Package: flask\_dynamo

**Why Using:**

**Documentation**:

## HTML

Use Bootstrup

https://getbootstrap.com/docs/5.0/getting-started/introduction/

Navbar - <https://getbootstrap.com/docs/5.0/components/navbar/>

# Files and Folders - overall

**File: credentials**

**Path: ~/.aws/credentials**

https://docs.aws.amazon.com/sdkref/latest/guide/file-location.html

**File : \_\_init\_\_.py**

<https://docs.python.org/3/reference/import.html#regular-packages>

Python defines two types of packages, regular packages and namespace packages. Regular packages are traditional packages as they existed in Python 3.2 and earlier. A regular package is typically implemented as a directory containing an \_\_init\_\_.py file. When a regular package is imported, this \_\_init\_\_.py file is implicitly executed, and the objects it defines are bound to names in the package’s namespace. The \_\_init\_\_.py file can contain the same Python code that any other module can contain, and Python will add some additional attributes to the module when it is imported.

# How to upload from git

# link Git & YouTube

aaaa

botocore.exceptions.ClientError: An error occurred (ExpiredTokenException) when calling the PutRecords operation: The security token included in the request is expired

# Deploy Flask Environment

## How to Execute Python Flask File on AWS

There are some ways to deploy Flask web application on web:

1. Deploying a Flask application to Elastic Beanstalk
2. Deploying a Flask application with EB2

We are working on DEBUG mode.

we select option #2 and we work at Debug mode.

In the main execute python file **- run.py** we config port: **8080** and **debug=True**

Figure - run.py file

*from market import app*

*if \_\_name\_\_=='\_\_main\_\_':*

*app.run(host='0.0.0.0', port=8080,debug=True)*

Setup to execute flask web application on debug mode:

1. In cloud9:

On terminal: Execute the lines below line by line :

cd eb-flask/

python3 -m venv virt (execute this line only once to create the environment : virt)

virt/bin/activate

pip install -r requirements.txt

python run.py

*ec2-user:~/environment/Revital $ cd eb-flask/*

*ec2-user:~/environment/Revital/eb-flask $ python3 -m venv virt*

*ec2-user:~/environment/Revital/eb-flask $ source virt/bin/activate*

*(virt) revital:~/environment/Revital/eb-flask $ python run.py*

*/home/ec2-user/environment/Revital/eb-flask/virt/lib64/python3.7/site-packages/flask\_sqlalchemy/\_\_init\_\_.py:873: FSADeprecationWarning: SQLALCHEMY\_TRACK\_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.*

*'SQLALCHEMY\_TRACK\_MODIFICATIONS adds significant overhead and '*

*\* Serving Flask app 'market' (lazy loading)*

*\* Environment: production*

*WARNING: This is a development server. Do not use it in a production deployment.*

*Use a production WSGI server instead.*

*\* Debug mode: on*

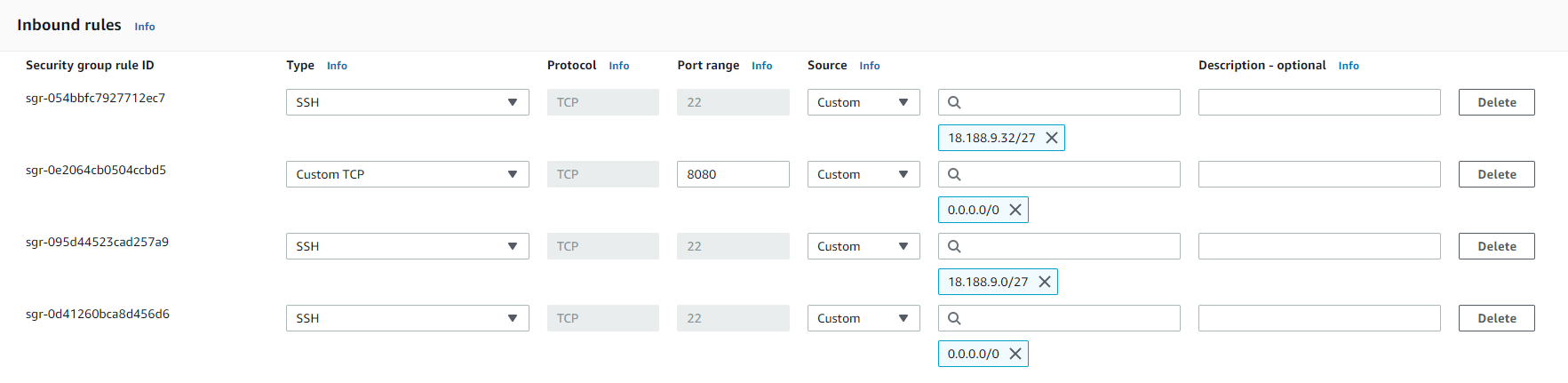
*\* Running on all addresses.*

*WARNING: This is a development server. Do not use it in a production deployment.*

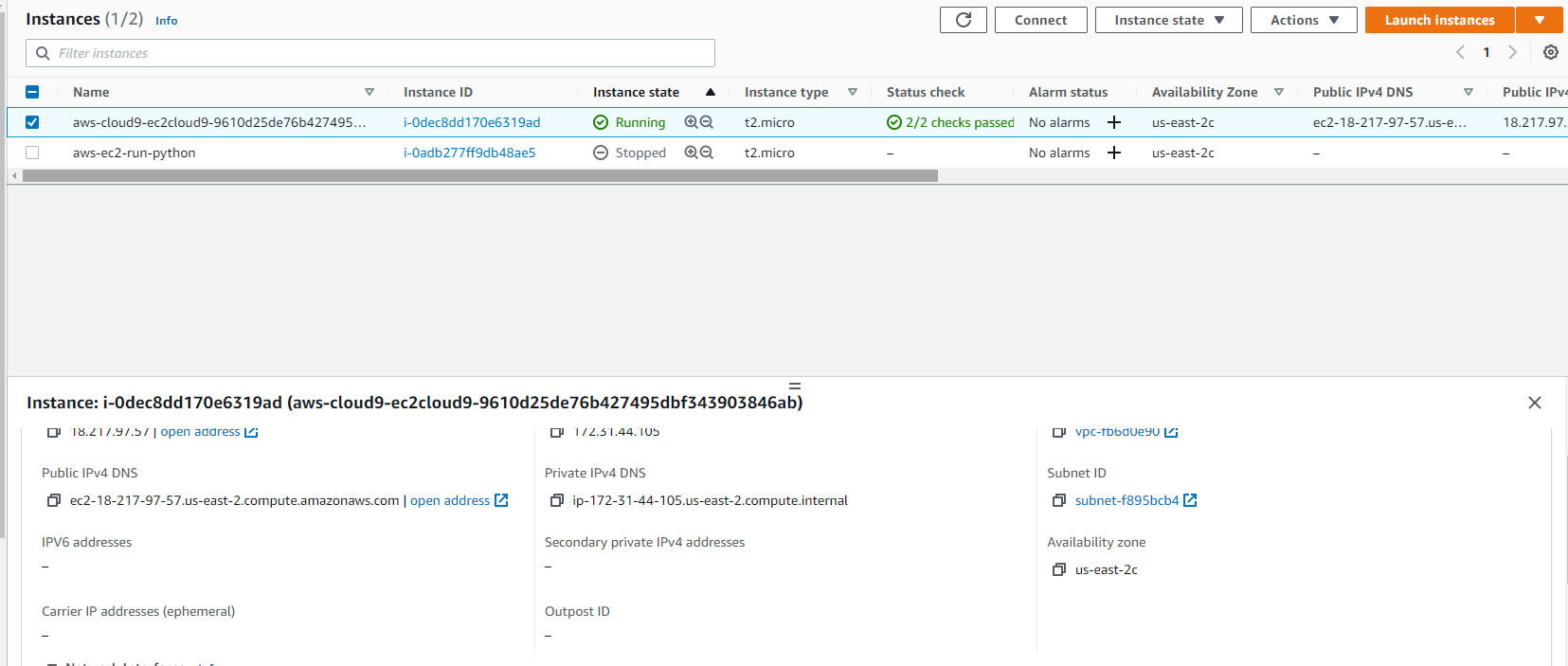
*\* Running on http://172.31.44.105:8080/ (Press CTRL+C to quit)*

*\* Restarting with stat*

1. Open EC2 service
2. Go to the running instance - aws-cloud9-ec2cloud9
3. Go to Security Group of the running instance
4. Click on - aws-cloud9-ec2cloud9 security group
5. Click on tab – inbound rules
6. Add new rule: custom TCP with port 8080



1. Copy - Public IPv4 DNS from the instance

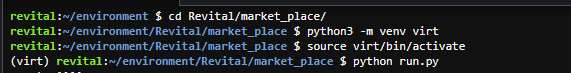


1. Concatenate the port to the: < Public IPv4 DNS >:8080
2. Test it in chrome browser:

For example:

<http://ec2-18-217-97-57.us-east-2.compute.amazonaws.com:8080>

<https://learn-to-code.workshop.aws/persisting_data/dynamodb/step-1.html>



# Documentation

## Deploy flask on AWS Cloud9

* <https://www.youtube.com/watch?v=bUEYe6AqlXE>

To install flask-dynamo, run: pip install flask-dynamo

* AWS\_ACCESS\_KEY\_ID (*your Amazon access key ID*)
* AWS\_SECRET\_ACCESS\_KEY (*your Amazon secret access key*)

<https://flask-dynamo.readthedocs.io/en/latest/quickstart.html#working-with-tables>

https://learn-to-code.workshop.aws/persisting\_data/dynamodb/step-1.html

troubleshooting:

error: RuntimeError: Working outside of application context.

https://flask.palletsprojects.com/en/2.0.x/appcontext/

<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/GettingStarted.CreateApp.html>

EROR:

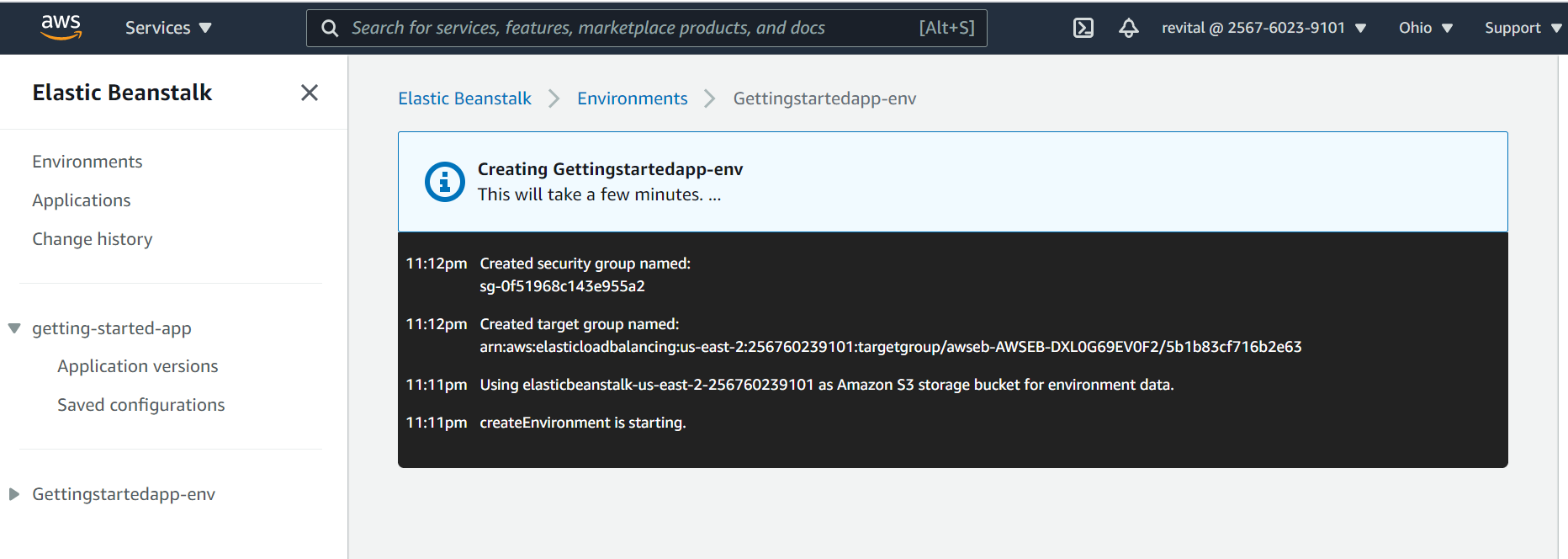
<https://aws.amazon.com/premiumsupport/knowledge-center/security-token-expired/>

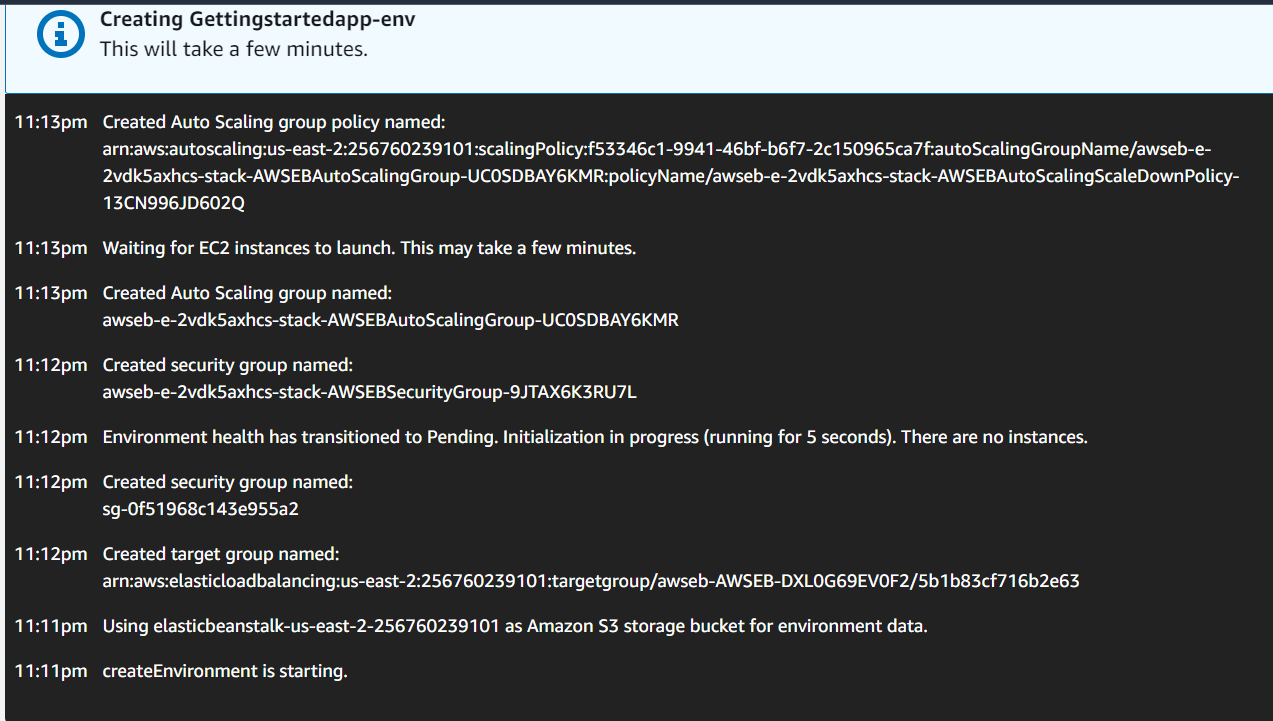
<https://dev.to/li_chastina/auto-refresh-aws-tokens-using-iam-role-and-boto3-2cjf>

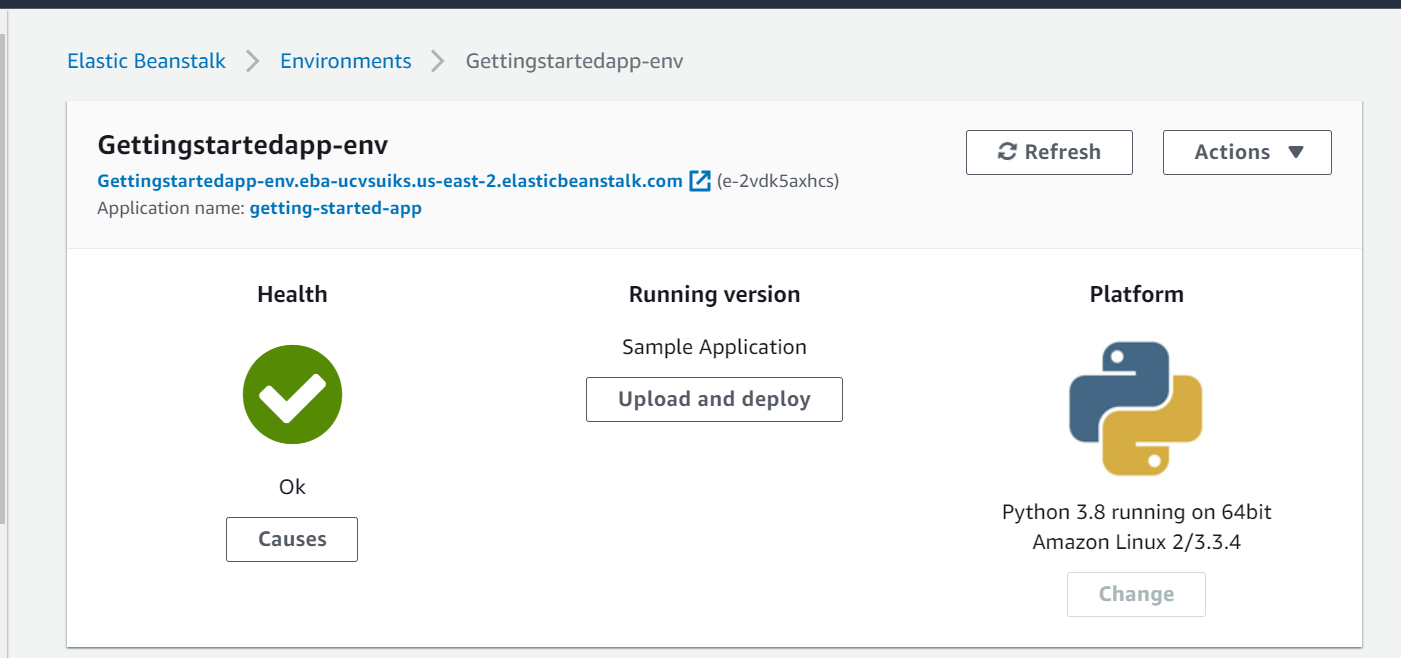
<https://developer.amazon.com/docs/login-with-amazon/retrieve-token-other-platforms-cbl-docs.html>

error:

A secret key is required to use CSRF.

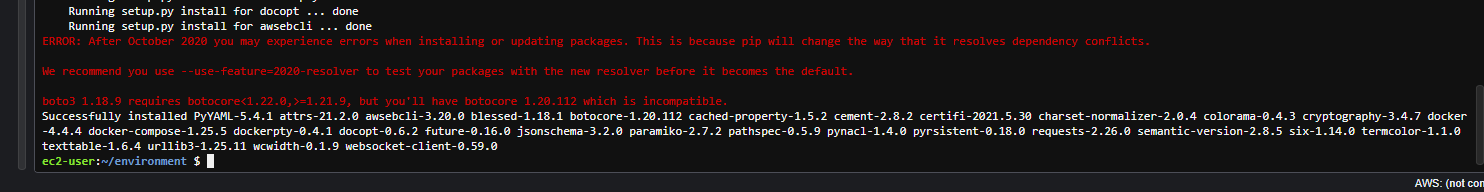


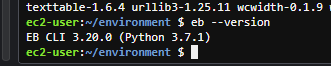


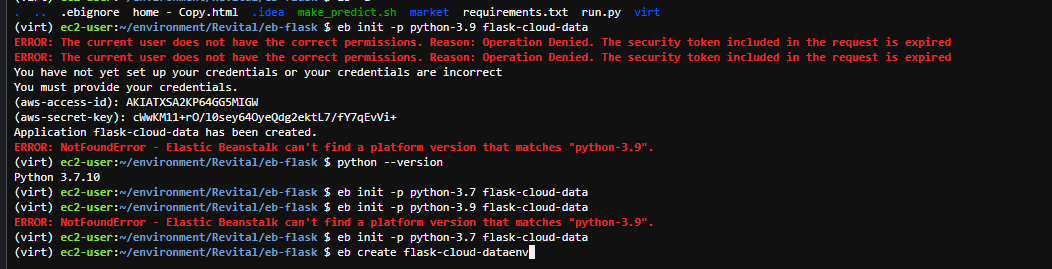


Use pip to install the EB CLI.

pip install awsebcli --upgrade --user





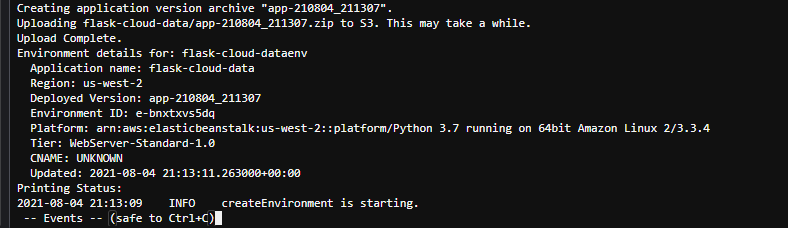


vim mode: https://www.keycdn.com/blog/vim-commands

eb init -p python-3.7 flask-cloud-data

eb create flask-cloud-dataenv

he take everything that is local and deploy



Troubleshooting:

<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli-troubleshooting.html>

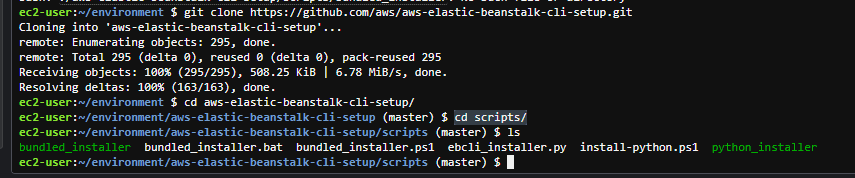
<https://www.youtube.com/watch?v=J9na_nTJYM8>

<https://github.com/aws/aws-elastic-beanstalk-cli-setup>

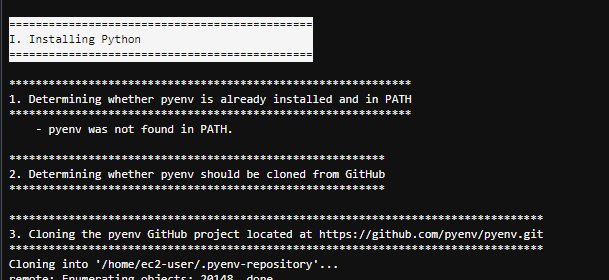
git clone <https://github.com/aws/aws-elastic-beanstalk-cli-setup.git>

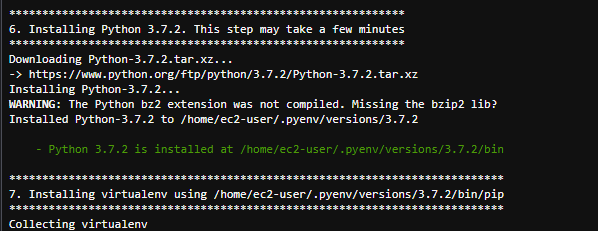
cd aws-elastic-beanstalk-cli-setup/

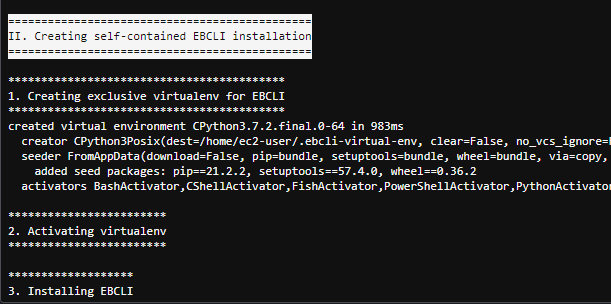
cd scripts/

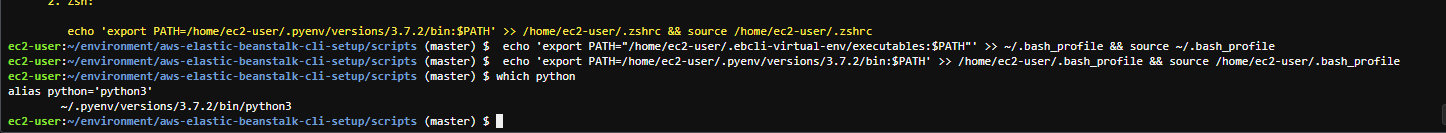


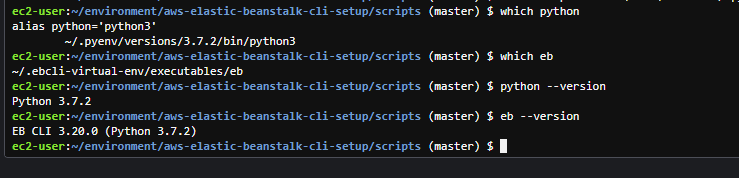
./bundled\_installer





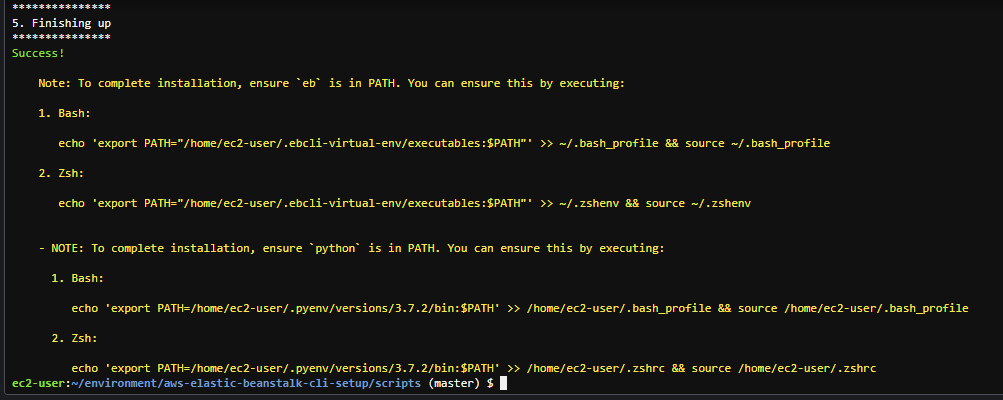


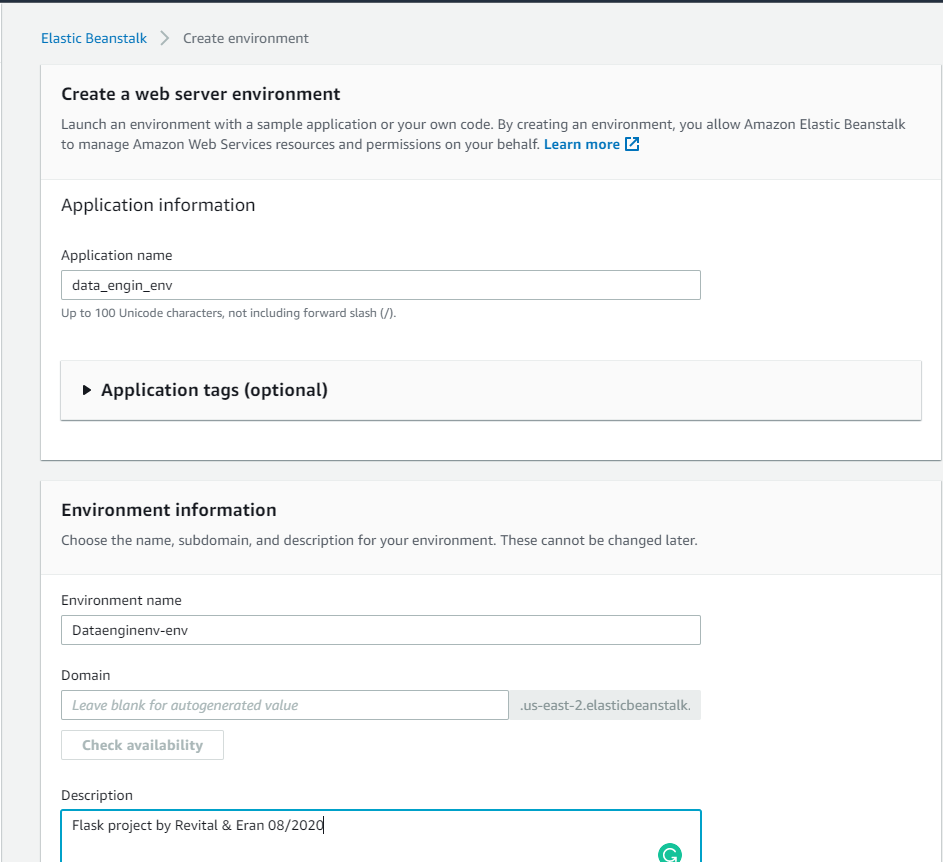


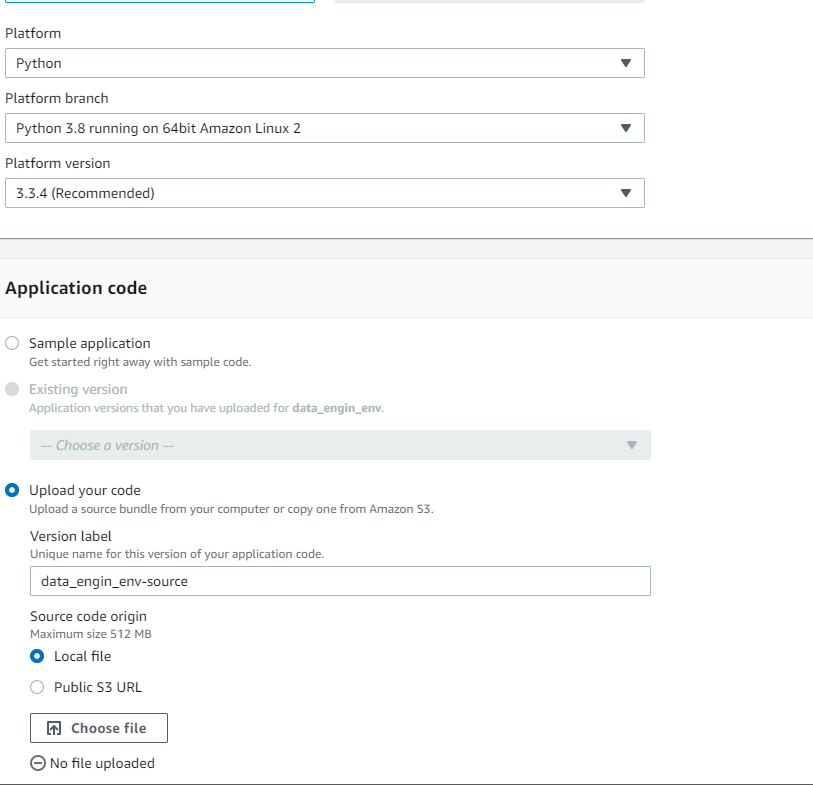
which eb

python --version

eb --version

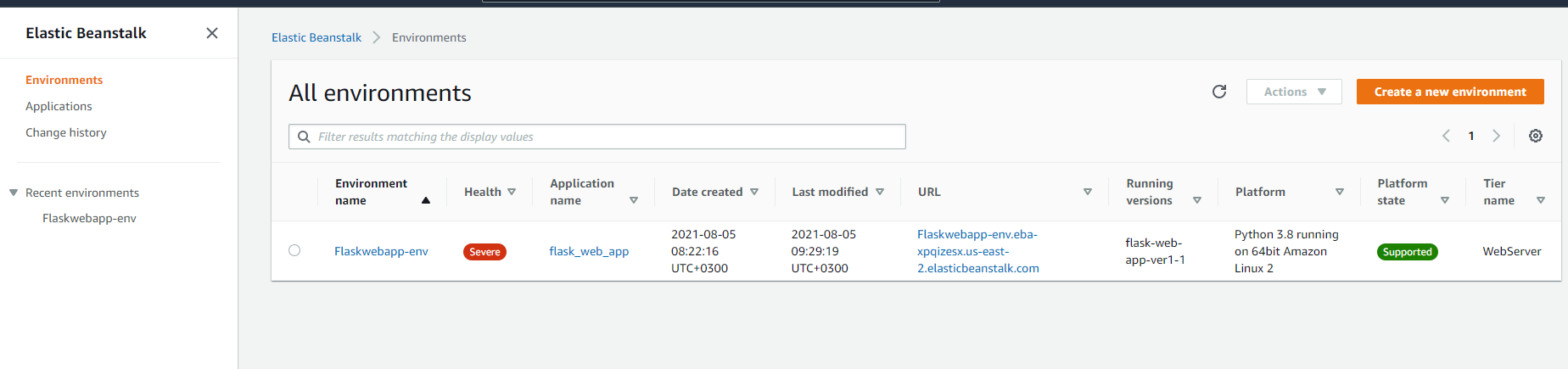






Application name: data\_engin\_env

Environment name: Dataenginenv-env



<http://flaskwebapp-env.eba-xpqizesx.us-east-2.elasticbeanstalk.com/>

