An Introduction to Cloud Computing

with Python and Amazon Web Services (AWS)

Harry Adam 09/10/24

I. Introduction

Welcome and Overview

Hello All

- About myself
- Define cloud computing
- Purpose of the talk
  - Using Python to interact with AWS and automate cloud services

Why Cloud Computing?

The Benefits!

- Flexibility
    - No need for physical servers, scale up or down as needed
- Cost-effective
    - Pay-as-you-go pricing model
- High availability
    - AWS provides global infrastructure for reliability and low-latency

Why AWS?

- AWS is the leading cloud service provider
- Key services include
  - EC2 - compute
  - S3 - storage
  - RDS - database
  - Lambda - serverless computing
- AWS integrates well with Python with the **boto3** library - the AWS Python SDK

II. Getting Started with AWS

Setting Up an AWS Account

- Register - https://aws.amazon.com/account/
- AWS Free Tier - https://aws.amazon.com/free

Navigating the AWS Management Console

- Quick demo of AWS console
    - https://eu-west-2.console.aws.amazon.com/console/home?region=eu-west-2#
- Note: using eu-west-2 (London) region

Python and Boto3 Introduction

Getting Set Up

- Install the AWS Python SDK
  - pip install boto3
- Authenticate your machine with AWS
  - Create a user with access tokens
  - Get the AWS CLI - https://aws.amazon.com/cli/
    - aws configure
- Check authenticated with
  - aws sts get-caller-identity
- Demo

Python and Boto3 Introduction

Hello World

- Demo
  - hello_world.py

III. Key AWS Services with Python

S3 (Simple Storage Service)

-
- Object storage
- Demo
  - s3.py
    - Need unique bucket name!

- *Demo is in free tier*
- *Free Tier Limits:*
  - *5 GB of Standard Storage.*
  - *20,000 GET requests and 2,000 PUT requests.*
  - *15 GB of Data Transfer Out.*

EC2 (Elastic Compute Cloud)

- Amazon EC2
- Compute
- Demo
  - Will need to create a key pair for this
  - ec2.py

- *Demo is in free tier*
- *Free Tier Limits:*
  - *750 hours per month of a t2.micro or t3.micro instance.*
  - *30 GB of EBS storage.*
  - *15 GB of Data Transfer Out.*

Lambda (Serverless Functions)

-
- Serverless
- Demo
  - Will need to create a role for this
  - lambda.py

- *Demo is in free tier*
- *Free Tier Limits:*
  - *1 million requests per month.*
  - *400,000 GB-seconds of compute time per month.*

RDS (Relational Database Service)

- amazon RDS
- RDMS
- No demo of this!

- *Free tier limits*
  - *Instance Type: 750 hours per month for db.t2.micro or db.t3.micro instances.*
  - *Database Engines: Supports MySQL, PostgreSQL, MariaDB, Oracle (BYOL), and SQL Server (Express).*
  - *Storage: 30 GB total storage for data and backups.*
  - *Backups: 30 GB of backup storage included.*
  - *Data Transfer: Free for inbound; standard charges for outbound after limits.*
  - *Duration: Free for 12 months from account creation. Monitor usage via AWS Console to avoid charges.*

IV. Best Practices & Security

IAM (Identity and Access Management)

- Always use IAM to manage access to your AWS resources securely
- Avoid using the root account for API access
  - Create dedicated users and roles

Cost Management

- Be aware of costs when using services
  - Especially once out of free tier!
- Use tools like AWS Budget and Cost Explorer to monitor usage

Monitoring and Scaling

- CloudWatch
- CloudTrail

V. Conclusion and Q&A

Key Takeaways

- Cloud computing enables flexibility, cost effectiveness and scalability
- AWS provides a wide variety of services that can be automated using Python and boto3
    - S3 for storage
    - EC2 for virtual machines / compute
    - Lambda for serverless computing
    - RDS for databases
    - Plenty of more services for you to explore!

Next Steps!

And Any Questions?

Demos, challenges & solutions on:

https://github.com/husername1/aws-with-python-talk

- Questions?
- Challenges!
  - Warm up: Get the demos running on your machine
  - Challenge 1: Automate an S3 Backup System
    - Write a Python script that uploads files from your local system to an S3 bucket daily
    - Use versioning in S3 to manage changes to the files
    - Advanced: Implement a mechanism to automatically delete files that are older than 30 days
  - Challenge 2: Deploy a Web Application on EC2
    - Create a basic Flask or Django web app
    - Use boto3 to automate launching an EC2 instance and deploy the web app on it
    - Ensure the app is accessible via a public IP
  - Bonus challenge (no solution): Deploy a web application with database
    - Hint: Look at AWS Elastic Beanstalk, https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/python-quickstart.html
  - **Remember to delete everything on AWS once finished to save any unexpected charges!**