UMD DATA605 - Big Data Systems

AWS Overview

Instructor: Dr. GP Saggese - gsaggese@umd.edu**
TAs: Krishna Pratardan Taduri, kptaduri@umd.edu Prahar
Kaushikbhai Modi, pmodi08@umd.edu
v1.1

SCIENCE
ACADEMY

UMD DATA605 - Big Data Systems

Dr. GP Saggese gsaggese@umd.edu

AWS: Resources

- Some basic info in the slides
- Many tutorials on-line
- Mastery
  - Amazon Web Services in Action 3rd Edition

Amazon Web Services (AWS)

- **AWS is a platform offering complete cloud solutions**
  - Computing (e.g., EC2)
  - Storing (e.g., S3)
  - Networking
- **It offers different levels of abstractions**
  - IAAS, PAAS, SAAS
  - From virtual hardware to applications, e.g.,
    - Host web sites
    - Run enterprise software
    - Run machine learning applications
- **Services can be controlled in different ways**
  - A web interface (aka "console")
  - CLI: **aws** command
  - Programmatically
    - Through language libraries, SDK (e.g., Python **boto3**)

SCIENCE
ACADEMY

AWS as Business

- Services charge a pay-per-use pricing model
- Data centers are distributed globally
  - US, Europe, Asia, South America
- 500 new services and features every year
- Insanely profitable
  - 91B/year in revenue (2023)
  - Grows 42% year-over-year
  - Controls 30% of cloud business

1998   2017

Types of Cloud Computing

- Cloud computing enables a shared pool of configurable computing resources
    - E.g., servers, storage, networks, applications, services to be used:
    - From everywhere
    - Conveniently
    - On-demand
- Clouds can be:
    - *Public*: open to use by general public (e.g., AWS)
    - *Private*: virtualize and share IT infrastructure within a single organization (e.g., government)
    - *Hybrid*: a mixture of public and private clouds

AWS vs Google Cloud vs Microsoft Azure

- **Similarities:**
  - Worldwide infrastructure
  - Provide IAAS (computing, networking, storage)
    - AWS EC2 / Google Compute Engine / Azure VMs
    - AWS S3 / Google Cloud Storage / Azure Blob storage
  - Pay-as-you-go pricing model
- **Differences:**
  - AWS
    - Market leader, most mature, and powerful
    - (ab)uses a lot of open source technologies
  - Azure provides Microsoft stack in the cloud
  - Google seems more focused on cloud-native applications rather than migrating local applications to the cloud

From On-premise to AWS

- **Aka "Cloud-transformation"**
- **Move a medium-sized e-commerce site from on-premise to the cloud**
- **Architecture**
  - **Web-server**: handle requests from customers
  - **DB**: store product information and orders
  - **Static content** (e.g., JPEG image)
    - Delivered over a content delivery network (CDN)
    - Reduce load on company services
  - **Dynamic content** (e.g., HTML pages)
    - E.g., products and prices
    - Delivered by web server

From On-premise to AWS

- **Step 1: Move to cloud**
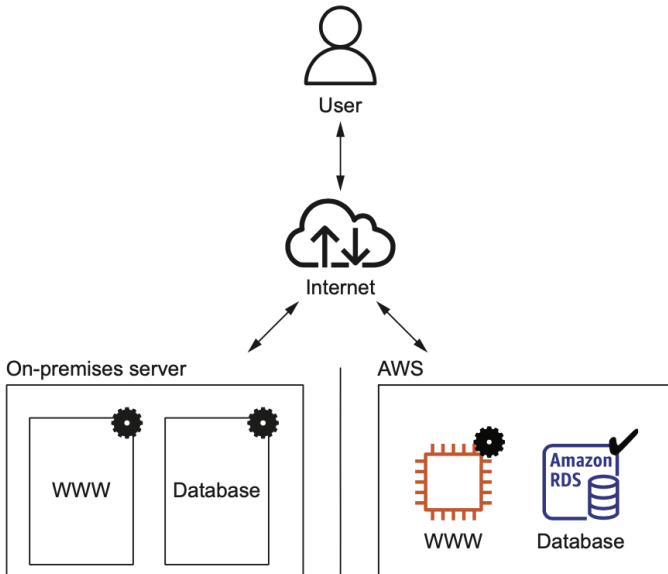
Optimizing Costs

- **On-demand instances**
  - Maximum flexibility, no restrictions
  - You start and stop VMs when you want
  - Pay by hour
- **EC2 / Compute saving plans**
  - 1 yr vs 3 yrs
  - Commit to a certain number of hours
  - Payment options: all, partial, no upfront
  - Discount: up to 3x cheaper than on-demand price
  - Useful for dev servers
- **Capacity reservation**
  - Get machines even in peak hours
- **Spot instances**
  - Bid for unused capacity
  - Price based on supply / demand
  - Discount: up to 10x cheaper than on-demand price
  - Useful for running asynchronous tasks

SCIENCE
ACADEMY

Programming the Infrastructure

- On AWS *everything* can be controlled via an API over HTTPS
  - Start a VM
  - Create 1TB of storage
  - Start Hadoop cluster
- Jeff Bezos 2002 API mandate
  - An email worth $100B/yr
  - HackerNews
  - An eye-witness from Google about it
- You can use:
  - AWS console
  - HTTP requests to the API
  - CLI (Command Line Interface)
    - Call AWS API from your terminal
  - SDK (Software Development Kit)
    - Call AWS API from your code
  - CloudFormation: templates that describe the state of the infrastructure and are translated into API calls

1. All teams will henceforth expose their data and functionality through service interfaces.

Infrastructure-As-Code

- **Use high-level programming language to control IT systems**
- You can apply software development to infra
  - Code repository
  - Automated tests
  - Continuous integration
- **DevOps (SRE in Google parlance)**
  - Mix devs and ops in the same team
  - Use software to bring development and operations closer together
  - Switch roles to experience each others' pain
    - Devs -> responsible for operational tasks (e.g., being on-call)
    - Ops -> involved in software development, making system easier to operate
  - Foster communication and collaboration

Infrastructure-as-Code: Advantages

- **Save time**
  - Reusing scripts or ready-to-use blueprints
  - Automating tasks done regularly
  - Copy-paste vs click-click-click
- **Fewer mistakes**
  - Push button flow
- **Consistency of actions**
  - Multiple deploys per day
- **Deployment pipeline**
  - Commit changes to source code in the repo
  - Application is built from source code
  - Automatic tests (e.g., integration tests)
  - Build testing environment
  - Run acceptance tests in isolation
  - Changes are propagated to production
  - Monitor production
- **The script is a detailed documentation**
  - It explains what and how, but not why (not a design document)

SCIENCE
ACADEMY

Create User Account

- Do not to use AWS root account for all development
- Create a new user
  - IAM = Identity and Access Management
- Access key ID + secret access key
- Access control
  - Enable programmatic access
  - Enable console access
  - Limit what an user can do through policies

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name*   mycli

+ Add another user

User name of the new user is mycli.

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. Learn more

Select AWS credential type*   ☑ **Access key - Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

Check Programmatic access to generate an access key

☐ **Password - AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

AWS Command Line Interface (CLI)

- Provide unified interface to all AWS services
- Output is in JSON format > **apt-get install awscli**
- Authenticate > **aws configure**
  - AWS access key ID
    - E.g., **AKIAIRUR3YLPOSVD7ZCA**
  - AWS secret access key
    - E.g., **SSKIng7jkA...enqBj7**
  - Default region name
    - E.g., **us-east-1**
- Execute command > **aws <service> <action> −key value**

Software Development Kit (SDK)

- SDK is a library that calls into AWS API from your favorite programming language
  - E.g., Python, Go, Ruby, C++, JavaScript
- **Pros:** it handles
  - Authentication
  - Retry on error
  - HTTPS communication
  - XML / JSON de-/serialization
- **Cons**
  - Imperative approach
  - Need to deal with dependencies

```python
import boto3

# Get the service resource.
dynamodb = boto3.resource('dynamodb')

# Create the DynamoDB table.
table = dynamodb.create_table(
    TableName='users',
    KeySchema=[
        {
            'AttributeName': 'username',
            'KeyType': 'HASH'
        },
        {
            'AttributeName': 'last_name',
            'KeyType': 'RANGE'
```

SCIENCE
ACADEMY

AWS CloudFormation

- **Use templates (e.g., JSON, YAML) to describe infrastructure**
- **Declarative vs imperative approach**
  - Declare the system, rather than listing the steps to build the system
- **AWS Stacks processes CloudFormation templates**
- **Pros**
  - Consistent way to describe the infrastructure
    - People implement same thing in a different way
  - Handle dependencies
  - Customizable
    - Inject custom parameters to customize templates
  - Testable
    - Create infra from a template, test, and shut down
  - Updatable
    - If you update the template, Stacks automatically applies changes to the infrastructure
  - It serves as documentation
    - Use it as code in source control

SCIENCE
ACADEMY

Securing Your System

- **Always install software updates**
  - New security vulnerabilities are found and fixed every day
- **Restrict access to AWS account**
  - Multiple persons and scripts should use different AWS account
  - Principle of "least privilege": grant only permissions needed to perform a job
- **Restrict network traffic**
  - Leave open only the ports that you strictly need
    - E.g., 80 for HTTP traffic, 443 for HTTPS
  - Close everything else
  - Encrypt traffic and data
- **Create a private network**
  - Subnets that are not reachable from the Internet

VPC

Public Subnet
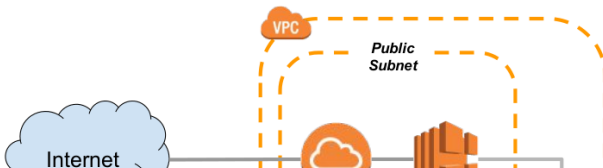
Internet

AWS Shared-responsibility Principle

- AWS is responsible for:
  - Protect network through monitoring Internet access
    - E.g., prevent DDoS attacks
  - Ensuring physical security of data centers
  - Decommissioning storage devices after end of life
- You are responsible for:
  - Restrict access using IAM
  - Encrypting network traffic (e.g., HTTPS)
  - Configuring firewall for VPN
  - Encrypting data
  - Updating OS (kernel, libs) and software
- More info here

SCIENCE
ACADEMY