Introduction and Overview of Cloud Computing

CSYE 6225: Network Structure & Cloud Computing
Northeastern University

Instructor: Raja Alomari,PhD

M3: Overview

Topics include:

- Introduction to Cloud Computing
- Cloud Deployment Models
- Cloud Service Models
- Major Cloud Providers
- Identity and Access Management (IAM)
- Cloud Security Best Practices
- Practical Hands-On Lab
- Future Trends in Cloud Computing

Cloud Computing

Overview, Models, and Key Concepts

Objectives:

- Understand Cloud Computing Concepts:
- Differentiate Cloud Service Models:
- Differentiate Cloud Service Models:
- Understanding Identity and Access Management (IAM):
- Implement Security Best Practices.
- Engage in Hands-On Cloud Lab Exercises:

What is Cloud Computing?

Definition: Delivery of computing services over the internet (public cloud) or a private network (private cloud) to offer faster innovation, flexible resources, and economies of scale.

Key Concepts:

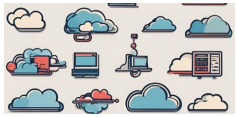
- **On-Demand Self-Service**
  Users can provision and manage computing resources as needed, without human intervention from the service provider.
- **Broad Network Access**
  Services are accessible over a network (internet or private network) from any device (e.g., laptops, phones, etc.) with standard protocols.

Resource Pooling
Resources (e.g., storage, processing) are pooled to serve multiple users, either within a public cloud environment or dedicated for a private cloud.

Rapid Elasticity
Resources can be scaled up or down quickly, automatically adjusting to meet fluctuating demands in both public and private cloud environments.

Measured Service
Cloud usage is metered and billed based on consumption in public clouds, while private clouds track usage for internal cost management or optimization.

Timeline: History and Evolution of Cloud Computing



1960s

Concept of Time-Sharing: Early concepts of shared computing resources emerged, allowing multiple users to access mainframe computers.

1999

salesforce.com: The first company to deliver enterprise applications over the internet, pioneering the Software as a Service (SaaS) model.

2002

Amazon Web Services (AWS) Launched.

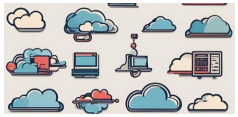
AWS began offering web-based services, marking the start of modern cloud computing.

2006

AWS EC2 & S3

AWS introduced Elastic Compute Cloud (EC2) and Simple Storage Service (S3), popularizing Infrastructure as a Service (IaaS).
AWS introduced Elastic Compute Cloud (EC2) and Simple Storage Service (S3), popularizing Infrastructure as a Service (IaaS).

Timeline: History and Evolution of Cloud Computing



2008

Google App Engine (GAE)

Google introduced its Platform as a Service (PaaS), allowing developers to build applications on Google's infrastructure.

2010

Microsoft Azure Launched

Microsoft entered the cloud space with Azure, further driving adoption of cloud services.

2011

IBM SmartCloud

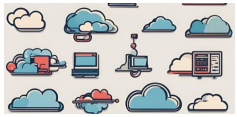
IBM launched SmartCloud to offer enterprise-grade cloud solutions, especially for private cloud and hybrid cloud models.

2013

Docker Popularizes Containers

Docker made container technology widely accessible, boosting cloud-native applications and microservices. Docker made container technology widely accessible, boosting cloud-native applications and microservices.

Timeline: History and Evolution of Cloud Computing



2019

Hybrid Cloud & Multi-Cloud

Companies increasingly adopt hybrid and multi-cloud strategies, mixing public and private clouds for flexibility and risk management.

2020s

Serverless Computing & Edge Computing

Continued innovations in serverless computing (e.g., AWS Lambda) and edge computing, extending cloud capabilities closer to users.

Benefits of Cloud Computing

Cost Efficiency: Pay-as-you-go pricing eliminates upfront hardware costs.

Scalability & Elasticity: Scale resources automatically to meet demand.

Global Accessibility: Access from anywhere with internet connectivity.

Security: Built-in security protocols, encryption, and access control.

Disaster Recovery: Automatic backup and recovery options ensure business continuity and data integrity in case of failures.

Flexibility & Innovation: Easily adopt new technologies and services, fostering innovation and allowing quick adaptation to changing business needs.

Automatic Updates: Cloud providers handle software updates and patches, reducing the burden on IT teams and ensuring access to the latest features and security enhancements.

Resource Optimization: Efficiently allocate and use resources, reducing waste and improving operational efficiency.

Environmentally Friendly: Shared resources in cloud data centers lead to more efficient energy use compared to traditional on-premises infrastructure.

Public vs Private Cloud

Aspect	Public Cloud	Private Cloud
Ownership	Owned and operated by third-party providers (e.g., AWS, Azure)	Owned and managed by the organization or a third-party provider exclusively for one organization
Resource Sharing	Shared infrastructure among multiple organizations (multi-tenant)	Dedicated infrastructure for a single organization (single-tenant)
Cost Model	Pay-as-you-go; Operational Expenses (OpEx)	Higher OpEx and potential CapEx for on-premises setups
Scalability	Virtually unlimited scalability; resources are highly elastic	Scalable, but limited by the physical infrastructure available or the contract with the hosting provider

Public vs Private Cloud

Aspect	Public Cloud	Private Cloud
Maintenance	Managed entirely by the cloud provider	Managed by the organization or outsourced to a third party
Security	Standardized security features (e.g., encryption, firewalls)	Enhanced security as infrastructure is dedicated to one organization, customizable for compliance needs
Deployment	Off-premises; accessible over the internet	Can be on-premises or hosted by a third party, but dedicated to one organization
Accessibility	Accessible from anywhere via the internet	Accessible via private network or VPN for enhanced security
Control	Limited control over underlying infrastructure	Full control over infrastructure, including hardware and software customization

Public vs Private Cloud

Aspect	Public Cloud	Private Cloud
Compliance	May face challenges with stringent data regulations	Easier to meet strict compliance requirements (e.g., HIPAA, GDPR) due to dedicated resources
Upgrades & Updates	Automatic upgrades handled by the provider	Managed by the organization, can control timing of updates
Use Cases	Ideal for businesses with variable workloads and cost-efficiency needs	Best for organizations with high security, regulatory, or performance requirements
Performance	Performance may vary based on network traffic and shared resources	More consistent performance due to dedicated resources

Cloud Computing: Broad Categories

Compute

Definition: Provides virtualized computing resources over the cloud.

Examples: Virtual Machines (VMs), Containers, Serverless Functions (e.g., AWS Lambda, Azure Functions).

Storage

Definition: Offers scalable and accessible data storage solutions.

Examples: Object Storage (e.g., AWS S3, Azure Blob Storage), Block Storage (e.g., AWS EBS, Azure Disk Storage), File Storage (e.g., AWS EFS, Azure Files).

Cloud Computing: Broad Categories

Networking

Definition: Manages and connects various cloud resources and networks.

Examples: Virtual Private Cloud (VPC), Load Balancers, Content Delivery Networks (CDN) (e.g., AWS CloudFront, Azure CDN), VPNs.

Security

Definition: Protects cloud infrastructure and data from threats and vulnerabilities.

Examples: Identity and Access Management (IAM), Encryption Services, Security Information and Event Management (SIEM), Firewalls.

Cloud Computing: Ongoing and Future Advances

Serverless Computing Run code without provisioning or managing servers, automatically scaling based on demand.
Examples: AWS Lambda, Azure Functions, Google Cloud Functions.

Artificial Intelligence (AI) / Machine Learning (ML)
Provides platforms and services for developing, training, and deploying AI/ML models.
Examples: AWS SageMaker, Google AI Platform, Azure Machine Learning.

Internet of Things (IoT) Integration
Definition: Connects and manages a network of physical devices and sensors.
Examples: AWS IoT Core, Azure IoT Hub, Google Cloud IoT.

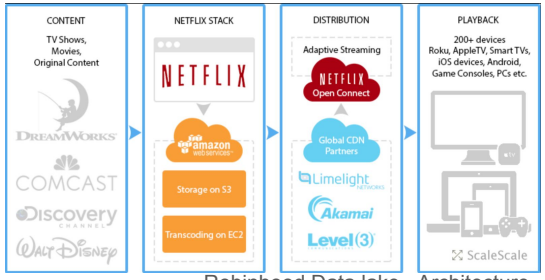
Real-World Examples

Netflix:

260M members.

100K AWS servers

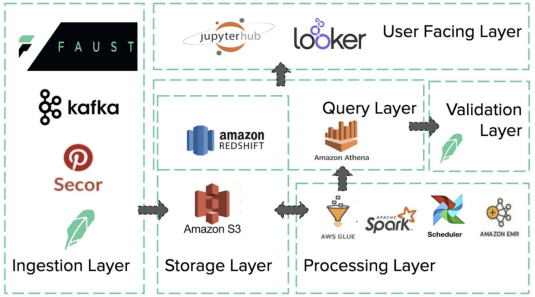
500B metrics collected daily.



Robinhood Data lake - Architecture

Real-World Examples

Robinhood Data lake: Architecture



Deployment Models

Public Cloud: Resources owned and operated by third-party cloud providers.

Private Cloud: Cloud infrastructure operated solely for a single organization.

Hybrid Cloud: Combination of public and private clouds with data portability.

Public Cloud

Key Features:

- Shared infrastructure
- Cost-efficient for small to large-scale operations
- Less control over infrastructure

Use Cases: Startups, global companies, SaaS products.

Providers: AWS, Microsoft Azure, Google Cloud.

Pros & Cons:

- Pros: No hardware costs, quick deployment, broad accessibility.
- Cons: Limited customization, potential security concerns.

Deployment Models

Private Cloud

Key Features:

- Dedicated hardware for one organization
- Full control over security, compliance, and customization
- Can be managed on-prem or via a third party

Use Cases: Financial institutions, healthcare, government sectors.

Example Providers: VMware, Proxmox, Nutanix, IBM Private Cloud.

Pros & Cons:

- Pros: High control, customizable, enhanced security.
- Cons: Higher cost, complex maintenance.

Hybrid Cloud

Key Features:

- Combines public and private clouds
- Seamless movement of workloads between environments
- Flexibility to keep sensitive data in private while scaling with public cloud

Use Cases: Multi-national corporations, Disaster recovery, Regulatory compliance.

Pros & Cons Table:

- Pros: Best of both worlds, improved flexibility, data portability.
- Cons: Complex setup, potential compatibility challenges.

Service Models: IaaS, PaaS, SaaS

IaaS (Infrastructure as a Service):

- Includes: Virtual machines, storage, networks, and operating systems.
- User Control: Hardware resources and operating system.

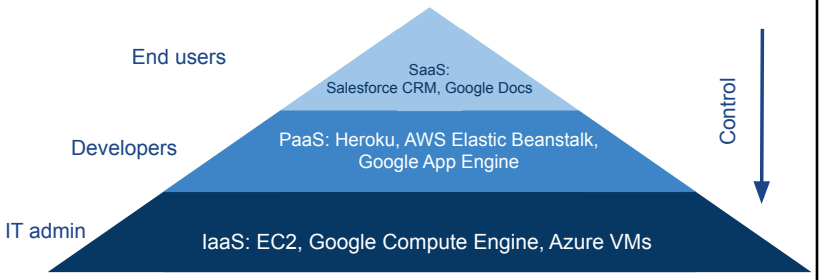
PaaS (Platform as a Service):

- Includes: Middleware, development tools, databases, and application hosting.
- User Control: Applications and data.

SaaS (Software as a Service):

- Includes: Applications and end-user services.
- User Control: Only the application settings and user data.

Service Models: IaaS, PaaS, SaaS



Infrastructure as a Service (IaaS)

Key Features:

- Virtual machines, storage, and networking
- Flexible and scalable
- Requires users to manage software and middleware

Use Cases:

- Web hosting
- Disaster recovery
- Testing and development environments

Examples: AWS EC2, Google Compute Engine, Azure Virtual Machines.

Platform as a Service (PaaS)

Key Features:

- Development platform including infrastructure, middleware, and tools
- Managed databases, OS, runtime environments
- Simplifies application development

Use Cases:

- Web and mobile app development
- API integration
- Microservices architecture

Examples: AWS Elastic Beanstalk, Google App Engine, Heroku.

Platform as a Service (SaaS)

Software as a Service (SaaS)

- Key Features:
  - Software accessible via web browser
  - Managed entirely by the provider
  - No need to install or maintain software on individual devices
- Use Cases:
  - CRM software (Salesforce)
  - Office productivity (Google Workspace, Microsoft 365)
  - Cloud storage (Dropbox, OneDrive)
- Examples: Slack, Zoom, Salesforce, Google Workspace.
- Visual: SaaS architecture where users access applications over the internet.

Managed vs Unmanaged Services

Managed Services	Unmanaged Services
Cloud provider handles most tasks like infrastructure management, updates, patches, and security.	The user or organization takes care of configurations, updates, patches, and security on the provided infrastructure.

Managed Services

Definition: Cloud provider manages the infrastructure, application stack, security, backups, updates, and monitoring.

Benefits:

- Minimal user intervention.
- Focus on core business instead of infrastructure management.
- Pre-configured solutions with automatic scaling, backups, and failover.
- Examples:
  - AWS Lambda: Fully serverless, auto-scaled event-driven code execution.
  - AWS RDS: Managed database service (automatic backups, patching, updates, etc.).
  - Azure App Services: Automatically manages app infrastructure, scaling, and security.

Unmanaged Services

Definition: The cloud provider offers the infrastructure or raw computing resources, but the user is responsible for handling configuration, updates, security, and backups.

Benefits:

- Greater control over the infrastructure and software.
- Ability to customize and configure environments exactly as needed.
- Suitable for specialized applications where control over the system stack is critical.
- Examples:
  - AWS EC2: Virtual servers where users configure the OS, storage, networking, and security.
  - AWS S3: Object storage service, users manage security settings, versioning, and lifecycle policies.
  - Azure Virtual Machines: User manages OS installation, patching, and scaling.

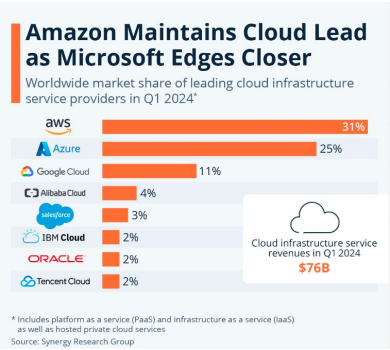
Managed vs Unmanaged Services

Aspect	Managed Services	Unmanaged Services
Control	Limited (focuses on application or usage layer)	Full control over infrastructure and software stack
Responsibilities	Provider handles infrastructure, updates, and security	User handles OS, updates, security, backups
Scalability	Automatic, often seamless (e.g., Lambda auto-scaling)	Requires user to configure scaling (e.g., EC2 auto-scaling)
Customization	Limited to application settings	Full customization of software stack and environment
Security Management	Cloud provider manages security patches and updates	Users must manage OS-level security and patches
Examples	AWS RDS, Lambda, Azure App Services	AWS EC2, S3, Azure VMs

Major Public Cloud Providers

Key Providers:

- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud Platform (GCP)
- IBM Cloud and Oracle Cloud
- Alibaba Cloud
- Oracle Cloud
- Rackspace



AWS Overview

Founded: 2006

Services: EC2, S3, RDS, Lambda, CloudFront, Route53

Global Reach: 24 Regions, 76 Availability Zones

Strengths: Market leader, comprehensive services, enterprise-ready

Google Cloud Platform (GCP) Overview

Founded: 2008

Services: Compute Engine, BigQuery, Cloud Storage, Kubernetes Engine

Global Reach: 20 regions, 61 zones

Strengths: Machine learning and AI tools, Big Data analytics

Microsoft Azure Overview

Founded: 2010

Services: Virtual Machines, Azure SQL, App Services, Azure Active Directory

Global Reach: 60+ regions worldwide

Strengths: Integration with Microsoft ecosystem, hybrid cloud solutions

Microsoft Azure Overview

Founded: 2010

Services: Virtual Machines, Azure SQL, App Services, Azure Active Directory

Global Reach: 60+ regions worldwide

Strengths: Integration with Microsoft ecosystem, hybrid cloud solutions

Identity and Access Management (IAM)

Identity and Access Management (IAM)

IAM Users

Definition: Represents an individual or system that needs to interact with AWS resources.

- Can be assigned specific permissions.
- Users can create access keys for API access.

Best Practices: Use strong password policies, assign least-privileged permissions.

IAM Roles

Definition: Roles provide permissions that can be assumed temporarily by users or services.

Use Cases:

- AWS Lambda needs to access an S3 bucket.
- Cross-account access: A user in one AWS account accesses resources in another account.

Identity and Access Management (IAM)

IAM Policies

Definition: Documents that define the actions, resources, and conditions under which actions are allowed or denied.

Structure:

- Version
- Statement
- Effect (allow/deny)
- Action and Resource

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::my-bucket/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:us-west-2:1[redacted]:instance/i-0123456789abcdef0"
        }
    ]
}
```


Identity and Access Management (IAM)

Types of AWS Policies

Identity-based Policies: Attached directly to an IAM user, group, or role

Resource-based Policies: Attached to AWS resources (e.g., S3 buckets, KMS keys).

Permission Boundaries: Defines the maximum permissions an IAM role or user can have.

Service Control Policies (SCPs): Applied to AWS Organizations, affecting all accounts in the organizational unit (OU).

Session Policies: Temporary policies used when users assume roles or create federated sessions.

Identity and Access Management (IAM)

Sample customer-managed policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::my-bucket/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:us-west-2:1[redacted]2:instance/i-0123456789abcdef0"
        }
    ]
}
```


Identity and Access Management (IAM)

AWS Managed Policies

- Created and maintained by AWS for common use cases.
- Simplifies permissions management by providing ready-to-use policies for typical job functions or tasks.
- +ve:
  - Predefined by AWS, reducing administrative burden.
  - Automatically updated by AWS to reflect changes in services.
  - Ideal for broad and general use cases like granting basic EC2, S3, or RDS permissions.
- Limitations:
  - Cannot be customized.
  - May provide more or fewer permissions than needed.
- Examples:
  - AdministratorAccess, AmazonS3ReadOnlyAccess, AmazonDynamoDBFullAccess.

Identity and Access Management (IAM)

Customer Managed Policies

- Created, owned, and maintained by the customer (i.e., AWS user).
- Tailored to specific organizational requirements or use cases.
- +ve:
  - Fully customizable based on your specific needs.
  - Can define granular permissions for individual users, roles, or groups.
  - Versioning allows you to track changes over time and revert if needed.
- Limitations:
  - Requires regular maintenance and updates.
  - More effort required to design, implement, and ensure security compliance.
- Examples:
  - A custom policy that allows users to manage a specific EC2 instance and S3 bucket.
  - A policy that permits access to a particular Lambda function but denies others.

Identity and Access Management (IAM): Policies

Aspect	AWS Managed Policies	Customer Managed Policies
Creation	Created and maintained by AWS	Created and maintained by the customer
Customization	Not customizable	Fully customizable
Maintenance	AWS automatically updates to include new features	Customer must manually update to add new permissions
Usage	Ideal for common, broad use cases	Best for specific, fine-grained access requirements
Examples	AdministratorAccess, AmazonS3ReadOnlyAccess	Custom policy allowing read-only S3 and write to EC2

Identity and Access Management (IAM)

Access Keys

Credentials consisting of an (1) access key ID and (2) secret access key to make programmatic calls to AWS services.

Best Practices:

- Rotate access keys regularly.
- Avoid embedding keys directly into code.
- Use IAM roles for temporary credentials where possible.

SSH Keys

SSH keys are used to authenticate access to AWS resources such as EC2 instances.

Best Practices:

- Use key pairs to securely access instances.
- Rotate keys and manage access tightly.
- Store private keys securely.

Identity and Access Management (IAM)

Trust Relationships

Defines which entities (principals) are allowed to assume a specific IAM role.

Use Cases:

- Cross-account access: Role in Account A can be assumed by users in Account B.
- AWS services like Lambda assuming roles to access resources.

Trusted entities

Entities that can assume this role under specified conditions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```


AWS CLI

```
$ aws ec2 describe-instances

$ aws ec2 start-instances --instance-ids i-1348636c

$ aws sns publish --topic-arn arn:aws:sns:us-east-1:
Failure"

$ aws sqs receive-message --queue-url https://queue.
```


Cloud Computing overview: Recap

What is AWS CLI?

- A command-line tool to manage AWS services.
- Allows interaction with AWS APIs through commands.
- Supports automation of AWS tasks.

Why Use AWS CLI?

- Faster than using the AWS Management Console for repetitive tasks.
- Scriptability and automation.
- Easily integrated with CI/CD pipelines.

What is CI/CD?

CI (Continuous Integration):

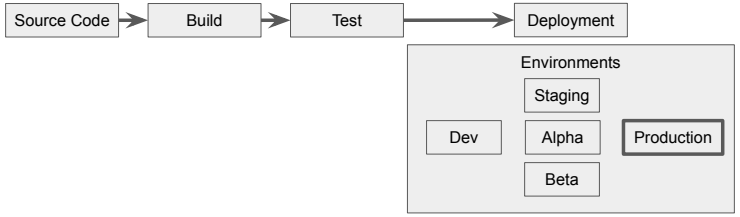
- Practice of merging code changes frequently into a shared repository.
- Each change triggers automated builds and tests, ensuring that new code integrates smoothly.

CD (Continuous Delivery/Continuous Deployment):

- Continuous Delivery: Automates the release process up to the staging environment, allowing teams to deploy manually to production.
- Continuous Deployment: Fully automates the release process, pushing code changes to production automatically once they pass tests.

GITLAB CI/CD, circleci, Travis CI, GitHub Actions, Jenkins

Typical Development/Deployment Cycle



Examples: CI/CD

```yaml
name: CI
on: [push, pull_request]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Set up Node.js
        uses: actions/setup-node@v2
      - run: npm install
      - run: npm test
```

```yaml
name: Deploy to Production
on:
  push:
    branches:
      - main
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Deploy to AWS
        run: aws s3 sync ./build s3://my-production-bucket
```


Back to AWS CLI

Installation of AWS CLI

- Installation Steps:
  - For Windows, macOS, Linux:
  - Full guide: AWS CLI Installation page.

You can also use an EC2 instance on your AWS account for practice.

Configuring AWS CLI:

- Set up access with aws configure
- Setup User Access Key or/and Role.
- Full guide: Setup Your Local environment credentials.

Basic AWS CLI Commands: S3 and IAM

S3 (S3 URI) with CLI:

- `aws help`
- `aws s3 ls`
- `aws s3 cp myfile.txt s3://mybucket/`
- `aws s3 cp s3://mybucket/myfile.txt .`

IAM with CLI:

- `aws iam create-user --user-name NewUser`
- `aws iam add-user-to-group --user-name NewUser --group-name Admins`
- `aws iam attach-role-policy --role-name MyRole --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess`

Basic AWS CLI Commands: EC2

EC2 with CLI:

- `aws ec2 run-instances --image-id ami-0abcdef12345 --instance-type t2.micro --key-name MyKeyPair`
- `aws ec2 describe-instances`
- `aws ec2 start-instances --instance-ids i-0abcdef12345`
- `aws ec2 stop-instances --instance-ids i-0abcdef12345`

Automated creation of an instance:

- `#!/bin/bash`
- `INSTANCE_ID=$(aws ec2 run-instances --image-id ami-0abcdef12345 --instance-type t2.micro --key-name MyKeyPair --query 'Instances[0].InstanceId' --output text)`
- `echo "Launched EC2 Instance with ID: $INSTANCE_ID"`

Basic AWS CLI Commands: Filtering

Filtering EC2 Instances by State:

- `aws ec2 describe-instances --filters "Name=instance-state-name,Values=running"`

Get the instance IDs of all running instances

- `aws ec2 describe-instances --query 'Reservations[*].Instances[*].InstanceId' --output text`

Basic AWS CLI Commands: IaaC

Deploy a CloudFormation stack:

- `aws cloudformation create-stack --stack-name MyStack --template-body file://template.yaml`

Custom Policy:

- `aws iam create-policy --policy-name EC2StartStopPolicy --policy-document file://ec2-policy.json`

Basic AWS CLI Commands: IaaC/Policy/Doc

Deploy a CloudFormation stack:

- `aws cloudformation create-stack --stack-name MyStack --template-body file://template.yaml`

Custom Policy:

- `aws iam create-policy --policy-name EC2StartStopPolicy --policy-document file://ec2-policy.json`

All AWS CLI Commands (Check latest Version):
https://awscli.amazonaws.com/v2/documentation/api/latest/reference/index.html

Overview Recap

Cloud Computing overview: Recap

- Cloud computing provides flexibility, scalability, and cost-efficiency.
- Public, private, and hybrid models offer varying levels of control and cost benefits.
- IaaS, PaaS, and SaaS provide different levels of abstraction from infrastructure management.
- IAM is crucial for securing access to cloud resources.

Cloud Computing overview: Recap

Cloud Computing Best Practices

- Security: Implement strong IAM policies, rotate access keys, and use MFA (Multi-Factor Authentication).
- Cost Management: Use cost monitoring tools to avoid over-provisioning resources.
- Scalability: Leverage autoscaling features to dynamically adjust resource allocation.
- Data Backup and Disaster Recovery: Ensure data is backed up and implement failover strategies for high availability.

Cloud Computing overview: Recap

Common Cloud Computing Pitfalls

- Over-Provisioning: Paying for more resources than needed due to poor cost management.
- Security Misconfigurations: Publicly exposing S3 buckets or incorrect IAM permissions.
- Vendor Lock-In: Becoming too reliant on one cloud provider, making it difficult to migrate to another.

Cloud Computing overview: Recap

Industry Trends and Future of Cloud Computing

- **Serverless Architecture:** Growing use of serverless platforms (AWS Lambda, Azure Functions).
- **AI and Machine Learning:** Cloud providers integrating AI/ML platforms (AWS SageMaker, Google AI).
- **Edge Computing:** Processing data closer to where it's generated (IoT integration).
- **Multi-Cloud Strategies:** Increasing use of multiple cloud providers to avoid vendor lock-in.

Lab: Hands-on

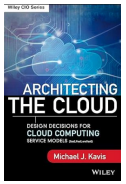
Objective: Create and configure IAM users, roles, policies.

Suggestions:

- Create an IAM user with programmatic access.
- Define a policy that grants read access to an S3 bucket.
- Explain an existing AWS managed policy.
- Create a role and assign it to an AWS Lambda function.
- Test cross-account access using a trust relationship.

Recommended Readings

1. "Cloud Computing: Concepts, Technology & Architecture" by Thomas Erl.
2. "Architecting the Cloud" by Michael J. Kavis.



Cloud Computing

Module 2 Conclusion

- Cloud Computing Provides Flexibility and Cost Efficiency:
- Deployment Models Vary Based on Control and Privacy Needs
- Service Models Offer Different Levels of Abstraction
- IAM is Essential for Cloud Security
- Security Best Practices Mitigate Risks:
- Cloud Providers Offer Diverse Services:
- Hands-On Experience is Critical for Mastery
- Cloud Computing is Evolving Rapidly
- Effective Cloud Management Requires Strategic Planning