Mod 1: Intro to AWS

Learning Objectives: → Benefits of AWS
→ On-demand delivery vs Cloud deployments
→ Pay-as-you-go pricing model

What is a Client-Server model?

Client ⇄ Server → Services like Amazon Elastic Cloud Compute (EC2)
↳ Web Browser
↳ Desktop Application

What is Cloud Computing?

- On demand delivery
- Over the internet
- Pay-as-you-go pricing

Benefits of Cloud Computing

- Variable expenses (pay as you go)
- No guessing capacity
- No maintainance of data centers
- Economies of scale
- Global reach
- rapid speed & agility

3 different deployment models

- Cloud-Based Deployment
  - run all parts of application in cloud
  - migrate existing applications → cloud
  - Design + Build new applications in the Cloud
  - Can build low level or high-level infrastructure
- On premises Deployment
  - Deploy resources using virtualization & resource management tools
  - increase resource utilization w/ application management & virtualization technologies
  - aka "private cloud" deployment
- Hybrid Deployment
  - Connect cloud-based resources ⇄ On-premises infrastructure
  - integrate cloud-based resources w/ legacy IT applications

Mod 2: Compute in the Cloud

Learning Objectives:
- EC2 → Benefits, instance types, Billing options, Auto scaling
- Elastic Load Balancing → Benefits, Examples
- AWS Compute Options
- SWS vs SAS

Amazon Elastic Compute Cloud

- EC2 → Secure, resizable compute capacity in Cloud as EC2 instances
- Avoid overhead of local servers
  - hardware costs, wait time, installation, configurations
- Virtualization technology w/ EC2
  - isolate VMs
  - Share resources } aka Multitenancy
  - Keep data secure
  - launch instances in minutes
  - pay only for what you use
- Vertical Scaling
  - Allocate more memory & cpu when needed
- Control Network requests for server

① Launch
- Select configs (op sys, Application Server, Applications; instance type & hardware config)
- Specify Security Settings
  ↳ to control network traffic

② Connect
- Connect to instance
  - multiple ways
  - exchange data
- Users connect by logging in

③ Use
- run commands
  → add software
  → add/move files
  → more...

Amazon EC2 Instance Types

- Optimized for different tasks
  ↳ Consider computing, memory, & storage
  ↳ Each type grouped under instance family
- General purpose instances
  → provide balance of computing, memory, storage & network resources
  → Applications → application servers, gaming servers, backend servers, databases
  → Use if resource needs for Compute, memory, & Network are similar
- Compute Optimized instances
  → Ideal for compute-bound applications / high-performance processes
  → Can use for batch processing (many transactions)
  → high-performance web servers, compute-intensive app servers, dedicated gaming servers

- Memory Optimized instances
  - → Fast performance memory intensive tasks → processing large datasets
  - → processing large amounts of data
- Accelerated Computing instances
  - → good for → floating-point number calculations, graphics processing, data-pattern matching
  - → use hardware accelerators or co-processors to perform functions efficiently
    - → More efficient than software running on CPU
    - → Hardware accelerator: component that expedites data-processing
- Storage Optimized instances
  - → good for work needing → high performance w/ locally stored data, high sequential read/write speeds
  - → Input/Output operations per second (IOPS):
    - ↳ metric for performance of storage device

Amazon EC2 Pricing

- Multiple purchase options

On-demand
- good for short term, irregular workloads
- no upfront costs or min costs
- only pay for what you use

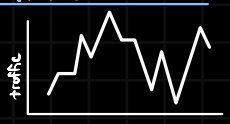
Reserved Instances
- Predictable usage (specific workloads)
- 1 or 3 yr terms w/ 3 payment options → ① upfront, ② Partial upfront, ③ No upfront

Spot Instances
- Flexible start & end times, or interruptions
- Spare computing EC2
  - ↳ Amazon can always reclaim EC2, so work can be interrupted

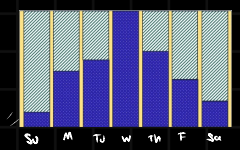
Savings Plan
- lower prices w/ commitment to consistent usage of 1 or 3 yr term
- Usage beyond commitment is charged on-demand rate

Dedicated Hosts
- Physical servers fully dedicated to your use

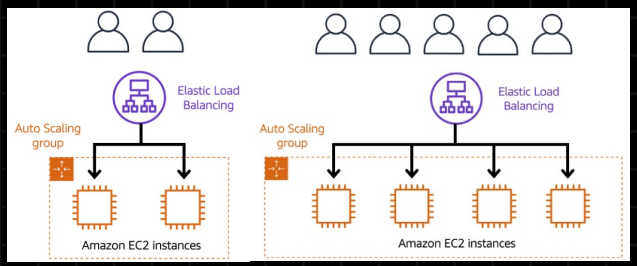
Scaling Amazon EC2



→ Can't handle peak hours
or
You waste resources at non-peak hours



- Scalability: beginning w/ only resources you need & designing architecture respond to demand by scaling in or out
  - ↳ Only pay for usage
  - ↳ Amazon provides this service
- Amazon EC2 Auto Scaling automatically adds & removes EC2 instances in response to demand
  - Dynamic Scaling: responds to changing demand
  - Predictive Scaling: sets schedule based on predicted demand
  - → Can use together for more speed !!!
  - Application Availability
  - Scaling up
    - more power to machines that are running
  - vs
  - Scaling Out
    - Creating more instances
  - → right amt of power for each process
- Auto Scaling Group
  - → Minimum Capacity: launched at creation
  - → Desired Capacity: optional, default capacity
  - → Maximum Capacity: Max instances that can be used



Directing traffic w/

Elastic Load Balancing

- AWS service that distributes incoming application traffic across multiple resources
- Load Balancer
  - Single point of contact for all web traffic to Auto Scaling Group
    - As EC2 instances are added/removed
      - → requests route to load balancer
      - → requests spread across multiple resources that will handle them
    - No single instance carries bulk of workload
- Auto-Scaling & Load-Balancing are SEPERATE services
  - ↳ Work together → high performance, availability

Ex: Coffee shop employee directs customers to which register they go to (ELB)

More registers are opened as traffic is increased and more customers (Auto-Scaling)

Each register is (EC2 instance)

Messaging & Queuing

- Monolithic Applications & Microservices
  - application components communicate w/ each other
  - Monolithic Application - App w/ tightly knit components
    - ↪ if one component fails, entire app may fail
  - Micro services Approach - components in app are loosely coupled → Simple Queue Service (SQS)
    - ↪ Maintains app availability when component fails → Simple Notification Service (SNS)
    - ↪ Uses services & components
- Amazon Simple Queue Service (SQS)
  - message queue service
  - Send, store, recieve messages btw software components