Module 6: Compute

AWS Academy Cloud Foundations

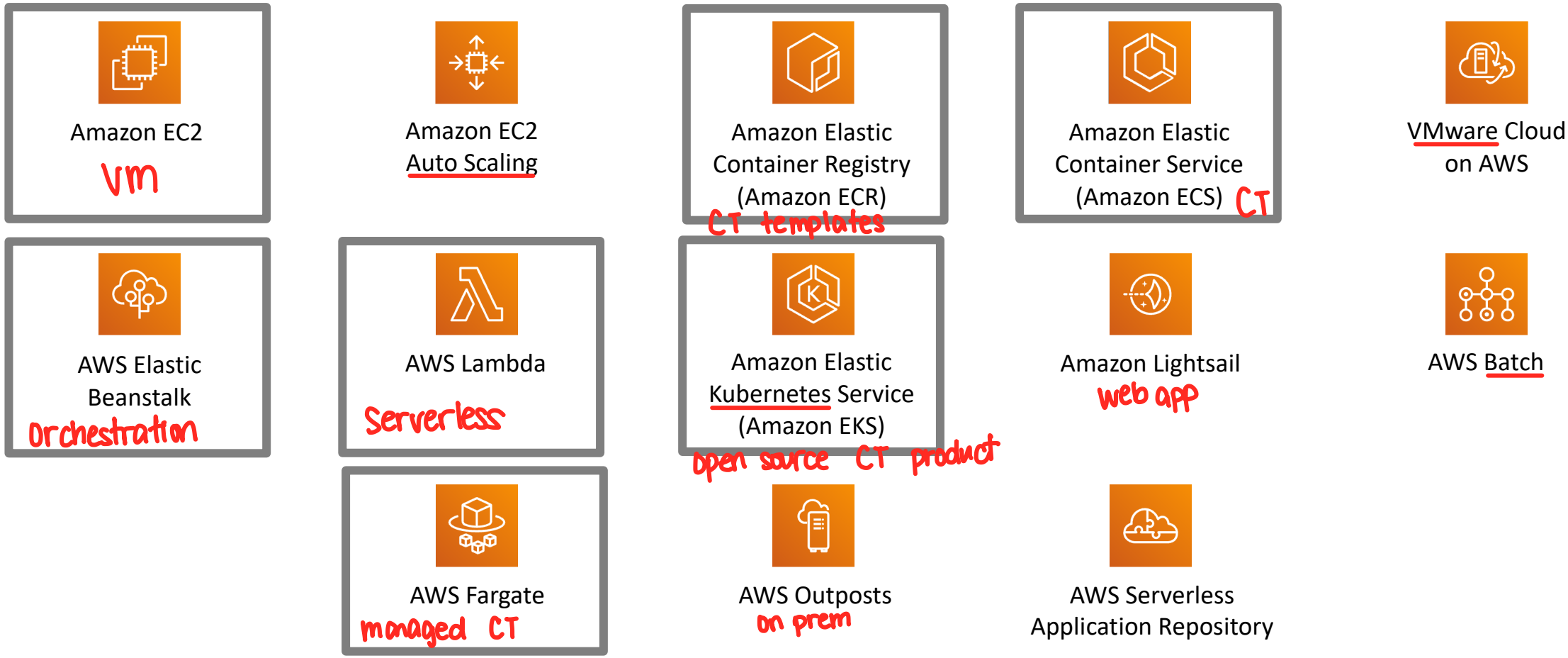
Section 1: Compute services overview

Module 6: Compute

AWS compute services

-> many services to support diff user needs

Amazon Web Services (AWS) offers many compute services. This module will discuss the highlighted services.



Categorizing compute services

services are "options" for you to choose from with each one having its pros and cons

Services	Key Concepts	Characteristics	Ease of Use
• Amazon EC2	• Infrastructure as a service (IaaS) • Instance-based • Virtual machines → control + flexibility	• Provision virtual machines that you can manage as you choose	A familiar concept to many IT professionals.
• AWS Lambda	• Serverless computing • Function-based • Low-cost → run code or scripts w/o managing a server	• Write and deploy code that runs on a schedule or that can be triggered by events • Use when possible (architect for the cloud)	A relatively new concept for many IT staff members, but easy to use after you learn how.
• Amazon ECS • Amazon EKS • AWS Fargate • Amazon ECR	• Container-based computing • Instance-based → lightweight & fast	• Spin up and run jobs more quickly	AWS Fargate reduces administrative overhead, but you can use options that give you more control.
• AWS Elastic Beanstalk	• Platform as a service (PaaS) • For web applications → orchestration ease in deployment	• Focus on your code (building your application) • Can easily tie into other services—databases, Domain Name System (DNS), etc.	Fast and easy to get started.

Choosing the optimal compute service

there are many services to choose from
So choose the option that best fits your needs

- The optimal compute service or services that you use will depend on your use case
- Some aspects to consider –
  - What is your application design?
  - What are your usage patterns?
  - Which configuration settings will you want to manage?
- Selecting the wrong compute solution for an architecture can lead to lower performance efficiency
  - A good starting place—Understand the available compute options

Section 2: Amazon EC2

Module 6: Compute

Amazon Elastic Compute Cloud (Amazon EC2)

VMs are the closest to traditional servers

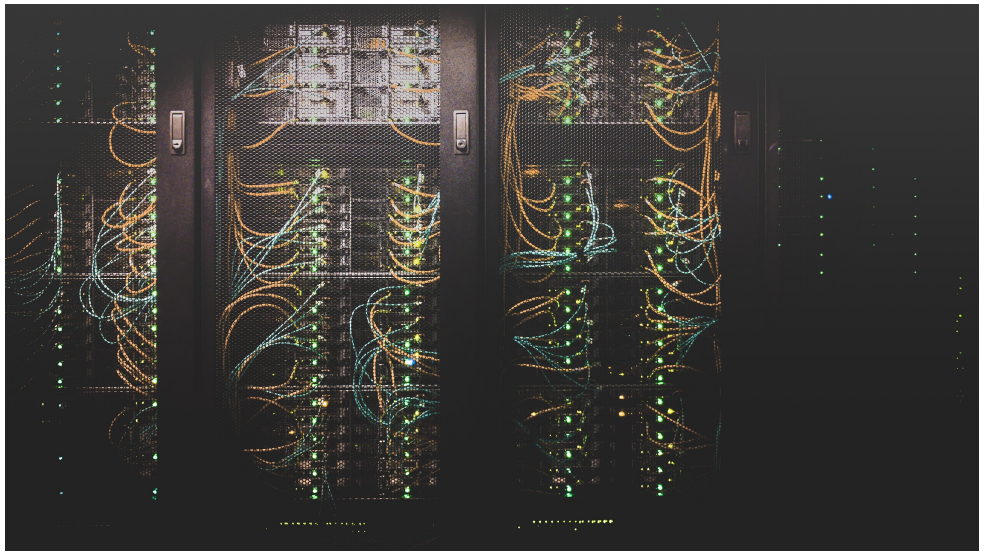
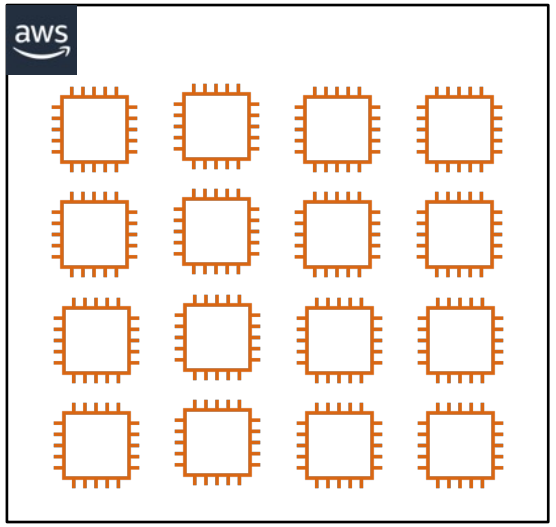


Photo by Taylor Vick on Unsplash

On-premises servers

Example uses of Amazon EC2 instances

- ✓ Application server
- ✓ Web server
- ✓ Database server
- ✓ Game server
- ✓ Mail server
- ✓ Media server
- ✓ Catalog server
- ✓ File server
- ✓ Computing server
- ✓ Proxy server



Amazon EC2 instances

Photo by panumas nikhomkhai from Pexels

Amazon EC2 overview

VMs have full control but also have more responsibilities

- **Amazon Elastic Compute Cloud (Amazon EC2)**
  - Provides virtual machines—referred to as EC2 instances—in the cloud.
  - Gives you *full control* over the guest operating system (Windows or Linux) on each instance.
- You can launch instances of any size into an Availability Zone anywhere in the world.
  - Launch instances from **Amazon Machine Images (AMIs)**.
  - Launch instances with a few clicks or a line of code, and they are ready in minutes.

    → speed & agility
- You can control traffic to and from instances.

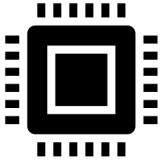
  → security groups

2. Select an instance type -> machine category

Choices made using the Launch Instance Wizard:

1. **AMI**
2. **Instance Type**
3. **Network settings**
4. **IAM role**
5. **User data**
6. **Storage options**
7. **Tags**
8. **Security group**
9. **Key pair**

- Consider your use case
  - How will the EC2 instance you create be used?
- The **instance type** that you choose determines –
  - Memory (RAM)
  - Processing power (CPU)
  - Disk space and disk type (Storage)
  - Network performance



- Instance type categories –
  - General purpose
  - Compute optimized
  - Memory optimized
  - Storage optimized
  - Accelerated computing

Categories depend on the service provider

- Instance types offer *family*, *generation*, and *size*

EC2 instance type naming and sizes -> refers to how much resources

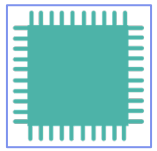
Instance type naming

- Example: **t3.large**
  - T is the family name
  - 3 is the generation number
  - Large is the size

Example instance sizes

Instance Name	vCPU	Memory (GB)	Storage
t3.nano	2	0.5	EBS-Only
t3.micro	2	1	EBS-Only
t3.small	2	2	EBS-Only
t3.medium	2	4	EBS-Only
t3.large	2	8	EBS-Only
t3.xlarge	4	16	EBS-Only
t3.2xlarge	8	32	EBS-Only

Select instance type: Based on use case → what your organization needs



	General Purpose	Compute Optimized	Memory Optimized	Accelerated Computing	Storage Optimized
Instance Types	a1, m4, m5, t2, t3	c4, c5	r4, r5, x1, z1	f1, g3, g4, p2, p3	d2, h1, i3
Use Case	Broad	High performance	In-memory databases	Machine learning	Distributed file systems

generic use-case
default choice

CPU-heavy
non-GPU compute

big data
in-memory db

floating point
linear algebra
graphics

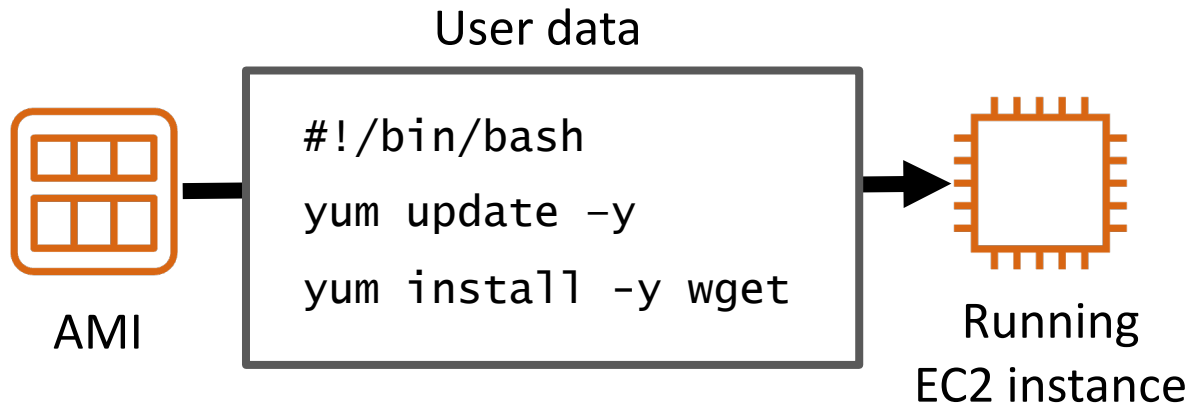
large storage
fast I/O or IOPS

5. User data script (optional)

you can run scripts on create such as download apache & git pull your web code

Choices made by using the Launch Instance Wizard:

1. **AMI**
2. **Instance Type**
3. **Network settings**
4. **IAM role**
5. **User data**
6. **Storage options**
7. **Tags**
8. **Security group**
9. **Key pair**



- Optionally specify a user data script at instance launch
- Use **user data** scripts to customize the runtime environment of your instance
  - Script runs the first time the instance starts

good for initial setup

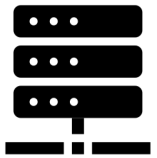
- Can be used strategically
  - For example, reduce the number of custom AMIs that you build and maintain

6. Specify storage → customize storage

Choices made by using the Launch Instance Wizard:

1. **AMI**
2. **Instance Type**
3. **Network settings**
4. **IAM role**
5. **User data**
6. **Storage options**
7. **Tags**
8. **Security group**
9. **Key pair**

- Configure the root volume → such as C:\
  - Where the guest operating system is installed
- Attach additional storage volumes (optional) → such a D:\
  - AMI might already include more than one volume
- For each volume, specify:
  - The **size** of the disk (in GB)
  - The **volume type**
    - Different types of solid state drives (SSDs) and hard disk drives (HDDs) are available
  - If the volume will be deleted when the instance is terminated
  - If **encryption** should be used
    - → user choice means user responsibility

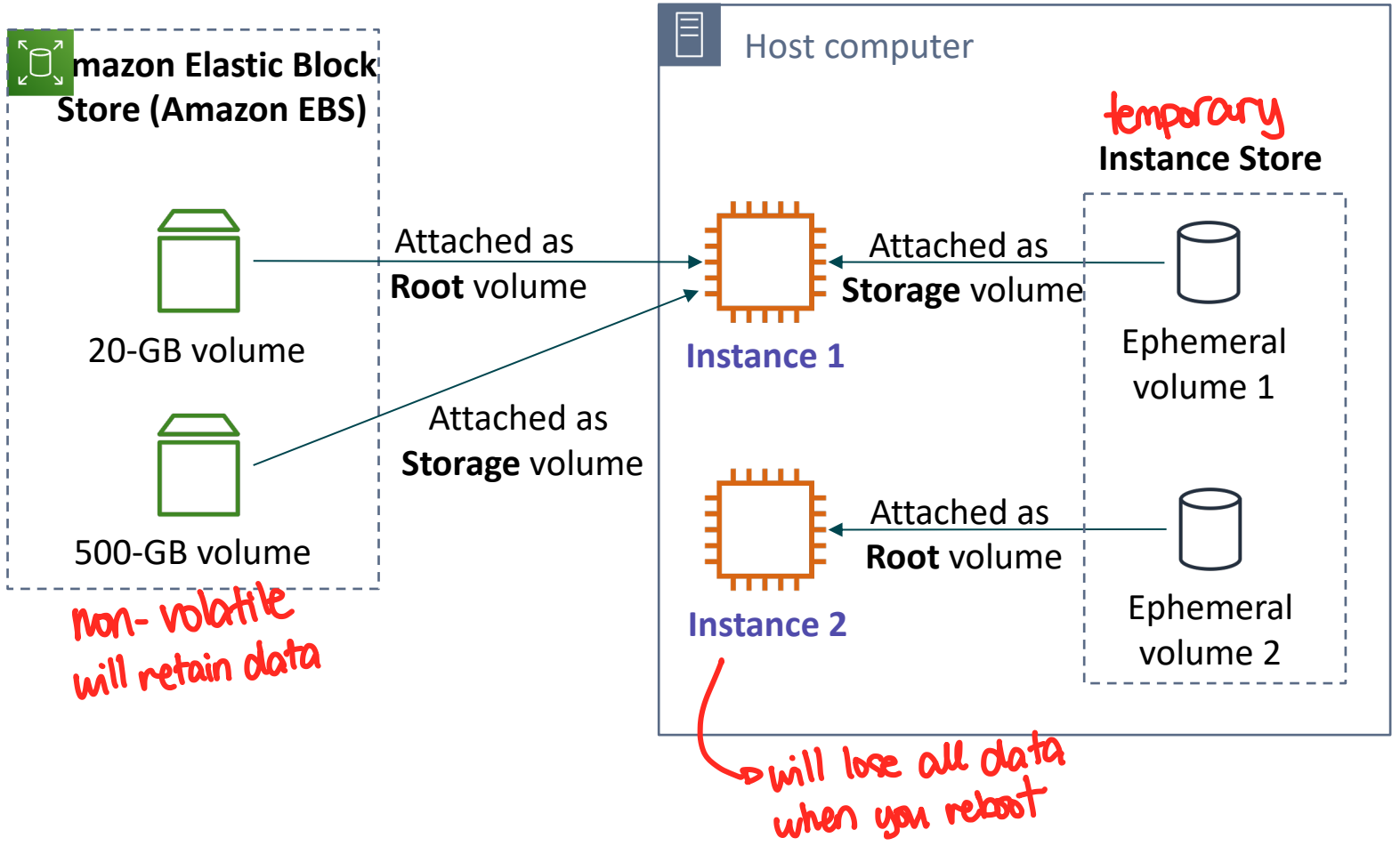


Amazon EC2 storage options

- **Amazon Elastic Block Store (Amazon EBS)** – → separate cost from your VM
  - Durable, block-level storage volumes.
  - You can stop the instance and start it again, and the data will still be there. → non-volatile but w/ payment
- **Amazon EC2 Instance Store** – → part of a VM w/ no added cost
  - Ephemeral storage is provided on disks that are attached to the host computer where the EC2 instance is running. → volatile and good for temporary instances
  - If the instance stops, data stored here is deleted.
- Other options for storage (not for the root volume) –
  - Mount an **Amazon Elastic File System (Amazon EFS)** file system. → network storage good for scaling & shared data
  - Connect to **Amazon Simple Storage Service (Amazon S3)**. → object storage for unstructured data

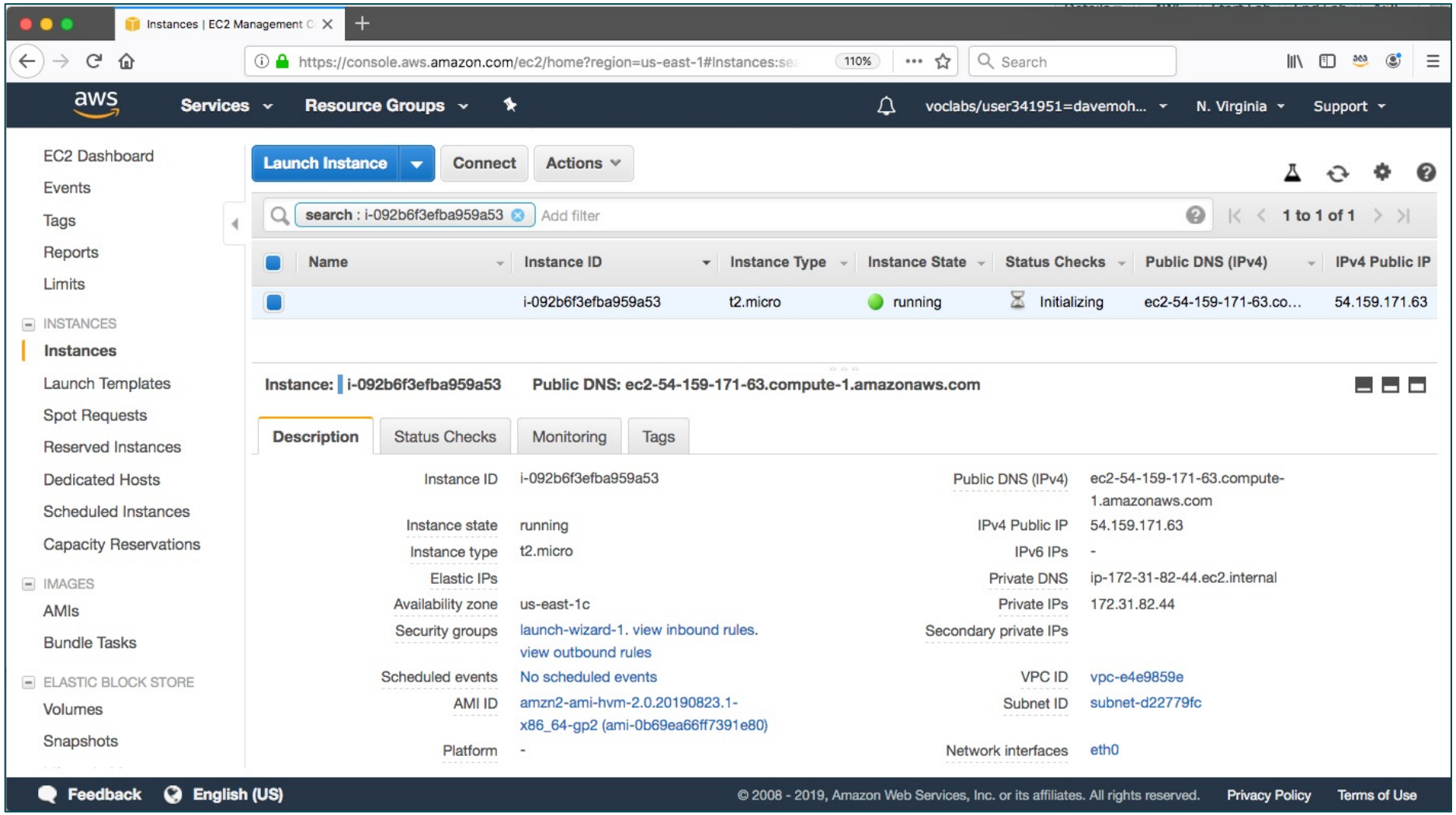
Example storage options

- **Instance 1** characteristics –
  - It has an **Amazon EBS** *root volume* type for the operating system.
  - What will happen if the instance is stopped and then started again?
- **Instance 2** characteristics –
  - It has an **Instance Store** *root volume* type for the operating system.
  - What will happen if the instance stops (because of user error or a system malfunction)?



Amazon EC2 console view of a running EC2 instance

You can use the Web GUI



Another option: Launch an EC2 instance with the AWS Command Line Interface → you can also use scripts

- EC2 instances can also be created programmatically.

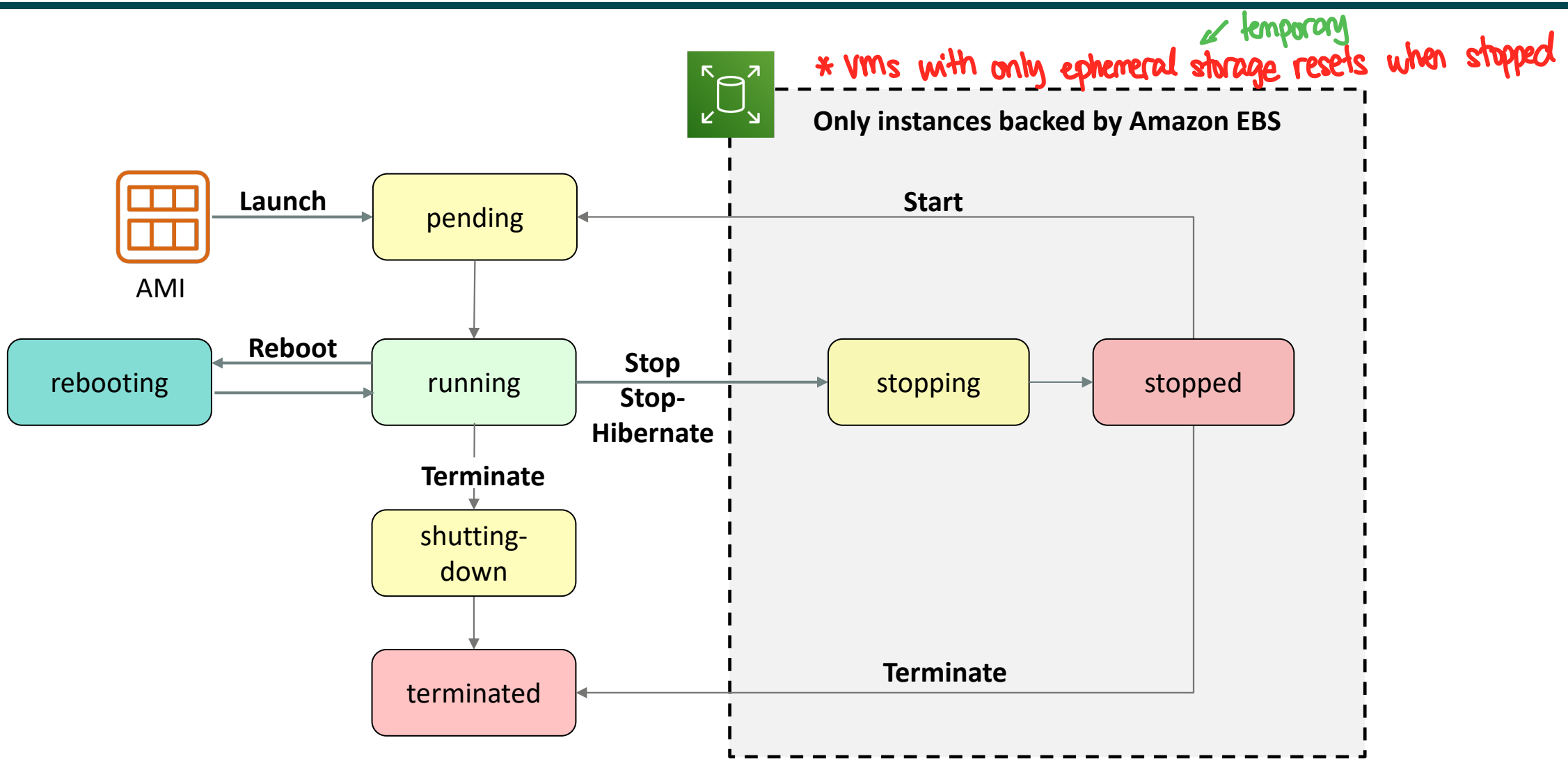
AWS Command Line Interface (AWS CLI)

- This example shows how simple the command can be.
  - This command assumes that the key pair and security group already exist.
  - More options could be specified. See the AWS CLI Command Reference for details.

Example command:

```
aws ec2 run-instances \
--image-id ami-1a2b3c4d \
--count 1 \
--instance-type c3.large \
--key-name MyKeyPair \
--security-groups MySecurityGroup \
--region us-east-1
```

Amazon EC2 instance lifecycle → machine states



Activity: Check your understanding

1. Between Amazon EC2 or Amazon RDS, which provides a managed service? What does *managed service* mean?
   - **ANSWER:** Amazon RDS provides a managed service. Amazon RDS handles provisioning, installation and patching, automated backups, restoring snapshots from points in time, high availability, and monitoring.
2. Name at least one advantage of deploying Microsoft SQL Server on Amazon EC2 instead of Amazon RDS.
   - **ANSWER:** Amazon EC2 offers complete control over every configuration, the OS, and the software stack.
3. What advantage does the Quick Start provide over a manual installation on Amazon EC2?
   - **ANSWER:** The Quick Start is a reference architecture with proven best practices built into the design.
4. Which deployment option offers the best approach for all use cases?
   - **ANSWER:** Neither. The correct deployment option depends on your specific needs.
5. Which approach costs more: using Amazon EC2 or using Amazon RDS?
   - **ANSWER:** It depends. Managing the database deployment on Amazon EC2 requires more customer oversight and time. If time is your priority, then Amazon RDS might be less expensive. If you have in-house expertise, Amazon EC2 might be more cost-effective.

Section 3: Amazon EC2 cost optimization

Module 6: Compute

Amazon EC2 pricing models → many options for different use-cases

On-Demand Instances

- Pay by the hour
- No long-term commitments.
- Eligible for the AWS Free Tier.

- stop guessing capacity but very expensive

Dedicated Hosts

- A physical server with EC2 instance capacity fully dedicated to your use.

- for compliance; rent the whole host server

Dedicated Instances

- Instances that run in a VPC on hardware that is dedicated to a single customer.

- good for isolation

Reserved Instances

- Full, partial, or no upfront payment for instance you reserve.
- Discount on hourly charge for that instance.
- 1-year or 3-year term. → long term w/ discounts

Scheduled Reserved Instances

- Purchase a capacity reservation that is always available on a recurring schedule you specify.
- 1-year term.

- reservation but w/ schedules
- if you don't need 24/7

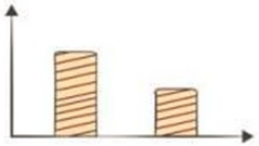
Spot Instances

- Instances run as long as they are available and your bid is above the Spot Instance price.
- They can be interrupted by AWS with a 2-minute notification.
- Interruption options include terminated, stopped or hibernated.
- Prices can be significantly less expensive compared to On-Demand Instances
- Good choice when you have flexibility in when your applications can run.

- if you need high compute with the lowest cost and you are not in a hurry

Per second billing available for On-Demand Instances, Reserved Instances, and Spot Instances that run Amazon Linux or Ubuntu.

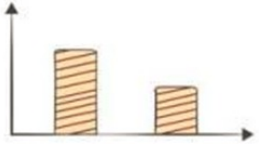
Amazon EC2 pricing models: Benefits

On-Demand Instances	Spot Instances	Reserved Instances	Dedicated Hosts
• Low cost and flexibility - good for short term - for auto scaling	• Large scale, dynamic workload - time insensitive - lowest cost	• Predictability ensures compute capacity is available when needed - long term - good for constant use or baseline resource needs	• Save money on licensing costs • Help meet compliance and regulatory requirements - rent the whole phy server - no "noisy neighbor" since whole server is dedicated

Amazon EC2 pricing models: Use cases → sample use-cases

Spiky Workloads



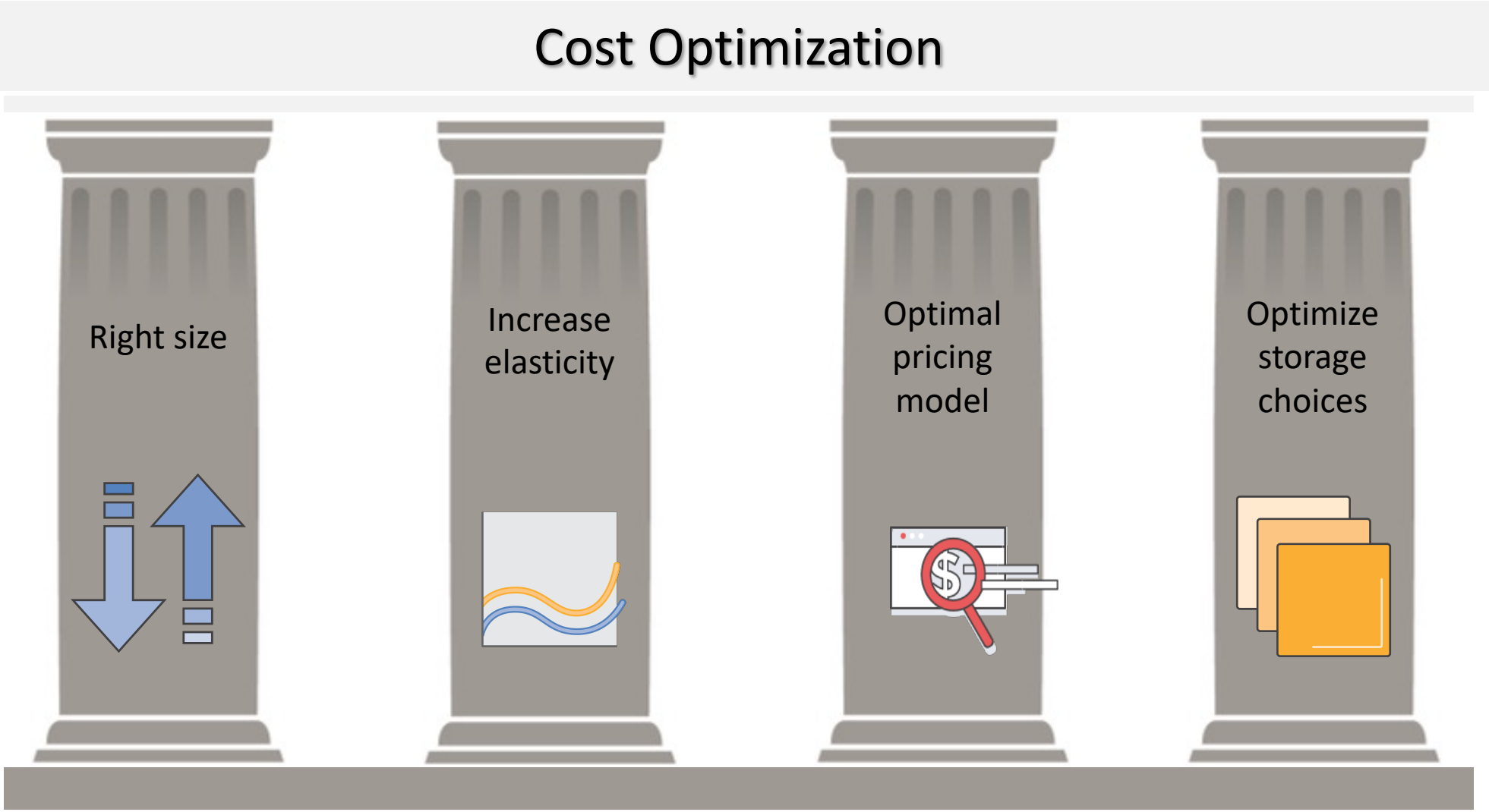
Time-Insensitive Workloads

Steady-State Workloads

Highly Sensitive Workloads

On-Demand Instances	Spot Instances	Reserved Instances	Dedicated Hosts
• Short-term, spiky, or unpredictable workloads • Application development or testing	• Applications with flexible start and end times • Applications only feasible at very low compute prices • Users with urgent computing needs for large amounts of additional capacity	• Steady state or predictable usage workloads • Applications that require reserved capacity, including disaster recovery • Users able to make upfront payments to reduce total computing costs even further	• Bring your own license (BYOL) • Compliance and regulatory restrictions • Usage and licensing tracking • Control instance placement

The four pillars of cost optimization -> practices to help reduce cost

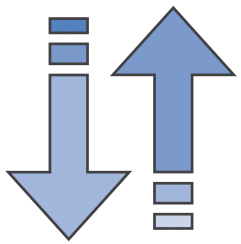


Pillar 1: Right size

determine your baseline by monitoring & observing
right size means resources are not wasted and the cost is manageable

Pillars:

1. Right size
2. Increase elasticity
3. Optimal pricing model
4. Optimize storage choices



✓Provision instances to match the need

- CPU, memory, storage, and network throughput
- Select appropriate instance types for your use

✓Use Amazon CloudWatch metrics (monitoring)

- How idle are instances? When?
- Downsize instances

✓Best practice: Right size, then reserve

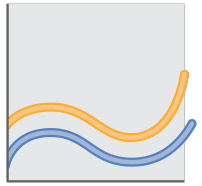
↳ since RI is cheaper

Pillar 2: Increase elasticity

scaling will support the changes or sudden resource spikes

Pillars:

1. Right-Size
2. Increase Elasticity
3. Optimal pricing model
4. Optimize storage choices



- ✓**Stop** or **hibernate** Amazon EBS-backed instances that are not actively in use
  - Example: non-production development or test instances
- ✓Use **automatic scaling** to match needs based on usage
  - Automated and time-based elasticity

Pillar 3: Optimal pricing model

-> you can use multiple pricing models at the same time

Pillars:

1. Right-Size
2. Increase Elasticity
3. Optimal pricing model
4. Optimize storage choices

✓Leverage the right pricing model for your use case

- Consider your usage patterns

✓Optimize and *combine* purchase types

✓Examples:

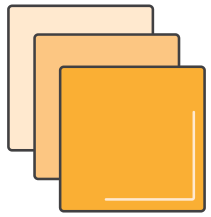
- Use **On-Demand Instance** and **Spot Instances** for variable workloads
- Use **Reserved Instances** for predictable workloads

✓Consider serverless solutions (AWS Lambda)

Pillar 4: Optimize storage choices → storage is not cheap

Pillars:

1. Right-Size
2. Increase Elasticity
3. Optimal pricing model
4. Optimize storage choices



- ✓ Reduce costs while maintaining storage performance and availability
- ✓ Resize EBS volumes
- ✓ Change EBS volume types
  - ✓ Can you meet performance requirements with less expensive storage?
  - ✓ Example: **Amazon EBS Throughput Optimized HDD (st1)** storage typically costs half as much as the default **General Purpose SSD (gp2)** storage option.
- ✓ Delete EBS snapshots that are no longer needed
- ✓ Identify the most appropriate destination for specific types of data
  - ✓ Does the application need the instance to reside on Amazon EBS?
  - ✓ Amazon S3 storage options with lifecycle policies can reduce costs

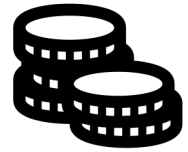
Measure, monitor, and improve -> its a continuous process

- Cost optimization is an ongoing process.

- Recommendations –
  - Define and enforce **cost allocation tagging**.

  - Define metrics, set targets, and review regularly.

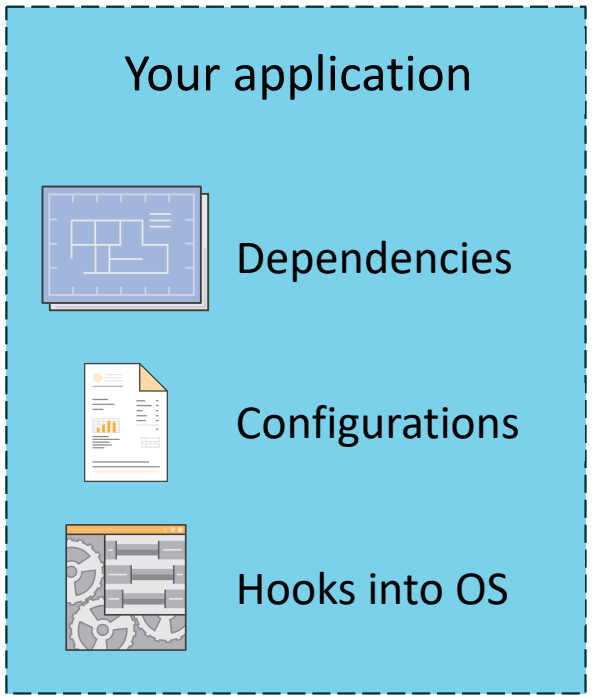
  - Encourage teams to **architect for cost**.
  - Assign the responsibility of optimization to an individual or to a team.



Container basics

- **Containers** are a method of operating system virtualization.

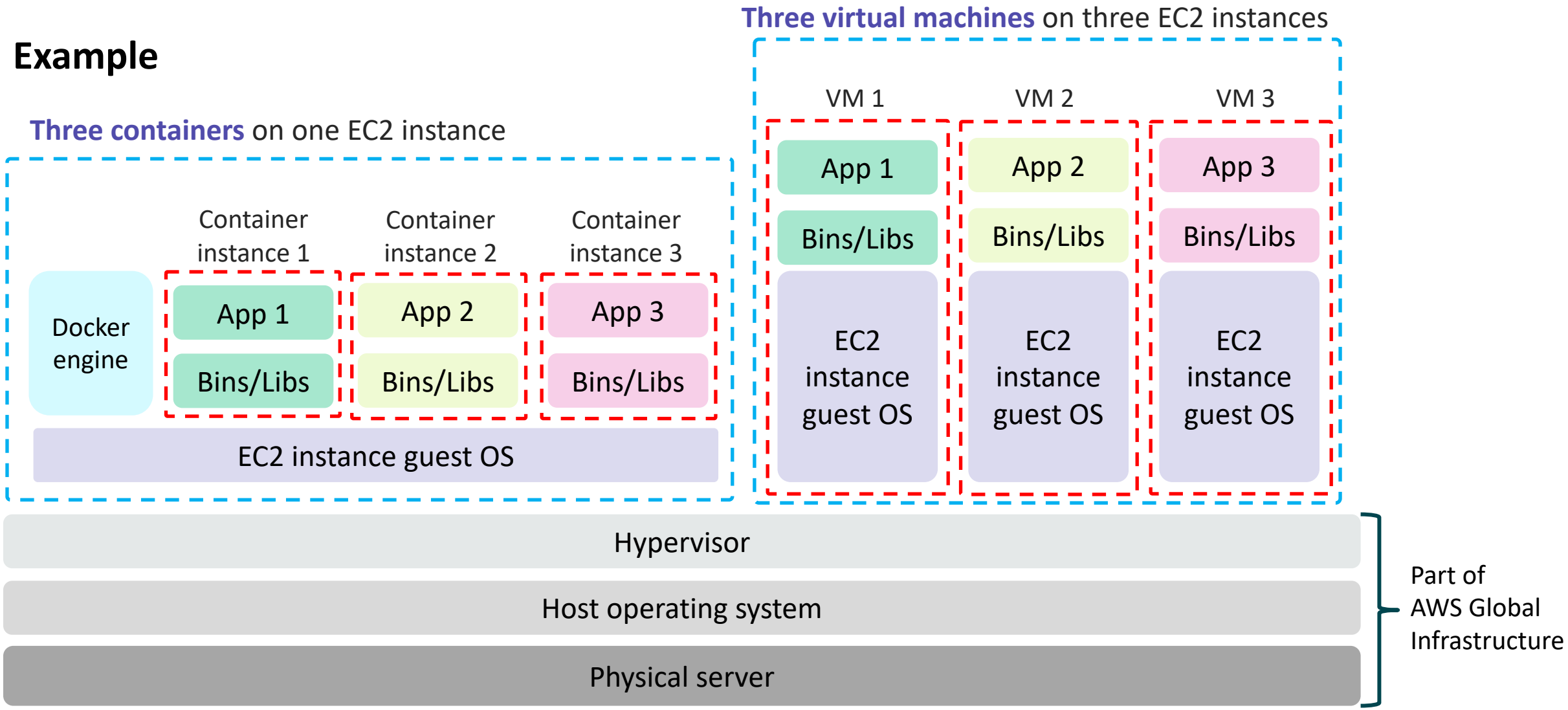
- Benefits –
  - Repeatable.
  - Self-contained environments.
  - Software runs the same in different environments.
    - Developer's laptop, test, production.
  - Faster to launch and stop or terminate than virtual machines



Containers versus virtual machines

VM = HW virtualization
CT = OS virtualization

Example



Amazon Elastic Container Service (Amazon ECS) → manage CTs

- Amazon Elastic Container Service (**Amazon ECS**) –
  - A highly scalable, fast, container management service

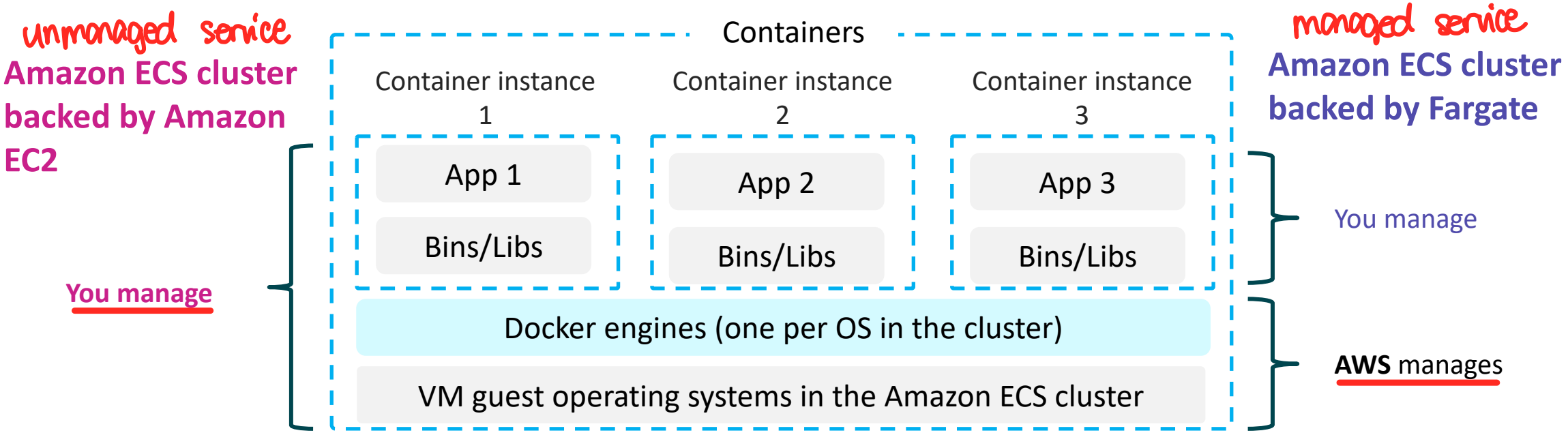
Amazon Elastic Container Service

- Key benefits –
  - Orchestrates the running of Docker containers
  - Maintains and scales the fleet of nodes that run your containers
  - Removes the complexity of standing up the infrastructure

- Integrated with features that are familiar to Amazon EC2 service users –
  - Elastic Load Balancing
  - Amazon EC2 security groups
  - Amazon EBS volumes
  - IAM roles

Amazon ECS cluster options

- **Key question**: Do ***you*** want to manage the Amazon ECS cluster that runs the containers?
  - If **yes**, create an **Amazon ECS cluster backed by Amazon EC2** (provides more granular control over infrastructure)
  - If no, create an Amazon ECS cluster backed by AWS Fargate (easier to maintain, focus on your applications)



Amazon Elastic Kubernetes Service (Amazon EKS)

well known & open source way to manage CT

- Amazon Elastic Kubernetes Service (**Amazon EKS**)
  - Enables you to run Kubernetes on AWS
  - Certified Kubernetes conformant (supports easy migration)
  - Supports Linux and Windows containers
  - Compatible with Kubernetes community tools and supports popular Kubernetes add-ons

Amazon Elastic Kubernetes Service

- Use Amazon EKS to –
  - Manage clusters of Amazon EC2 compute instances
  - Run containers that are orchestrated by Kubernetes on those instances

Amazon Elastic Container Registry (Amazon ECR)

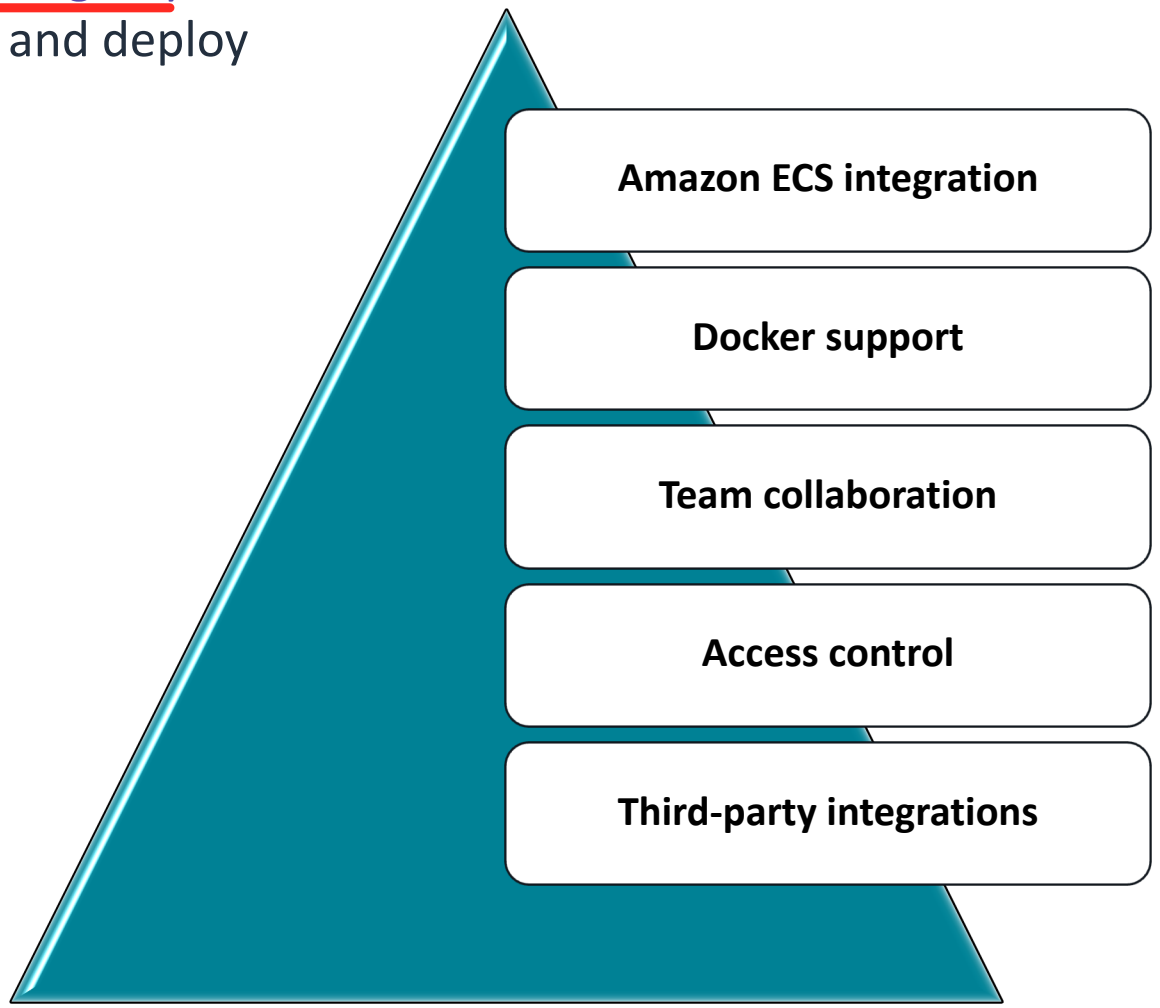
→ repo of CT images

Amazon ECR is a fully managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images.

Amazon Elastic Container Registry

Image

Registry



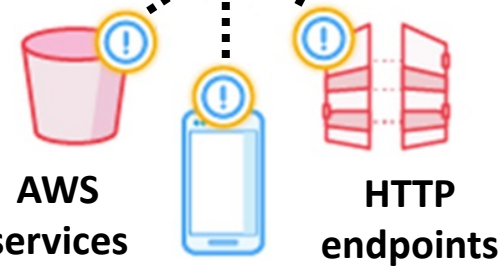
AWS Lambda: Run code without servers

→ serverless just means that you do not need to manage the server

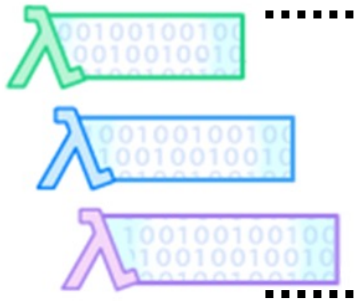
AWS Lambda is a **serverless** compute service.

Upload your code

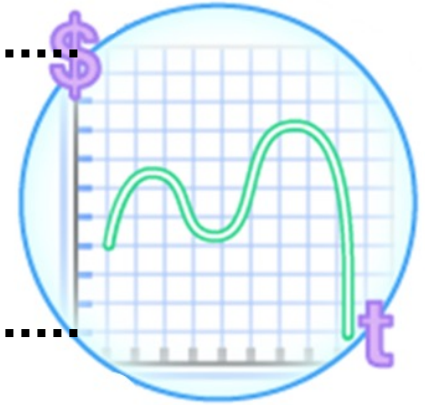
The code you run is a **Lambda function**



Run your code on a ***schedule*** *or in response to* ***events***



Your code runs only when it is triggered



Pay only for the compute time that you use

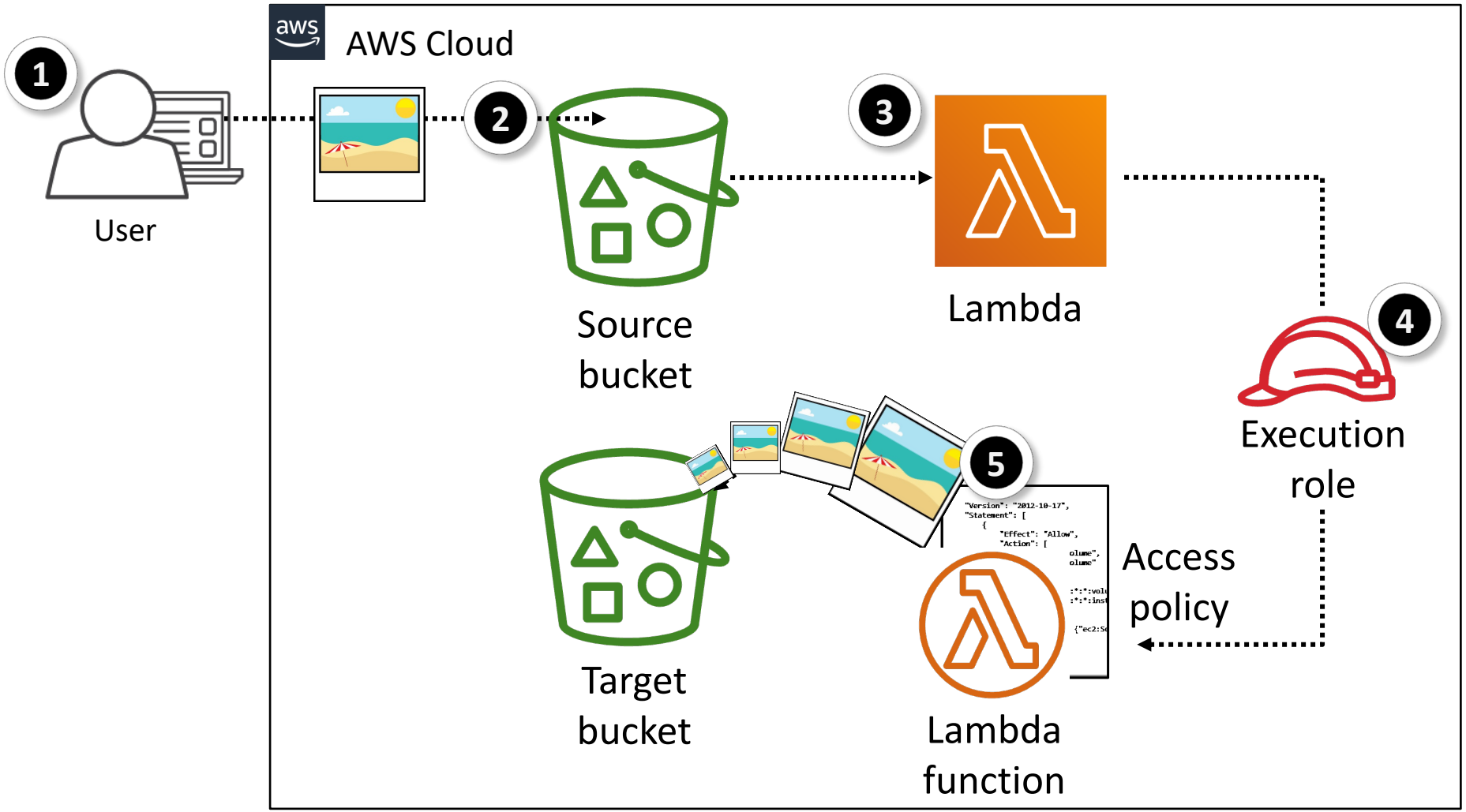
Benefits of Lambda

AWS Lambda

- It supports multiple programming languages
- Completely automated administration
- Built-in fault tolerance
- It supports the orchestration of multiple functions
- Pay-per-use pricing

Event-based Lambda function example: Create thumbnail images

→ call Lambda as scripts for processing



AWS Elastic Beanstalk

→ PaaS orchestration to simplify deployment for developer productivity

AWS Elastic Beanstalk

- An easy way to get **web applications** up and running
- A **managed service** that automatically handles –
  - Infrastructure provisioning and configuration
  - Deployment
  - Load balancing
  - Automatic scaling
  - Health monitoring
  - Analysis and debugging
  - Logging
- No additional charge for Elastic Beanstalk
  - Pay only for the underlying resources that are used

AWS Elastic Beanstalk deployments

- It supports web applications written for common platforms
  - **Java**, **.NET**, **PHP**, **Node.js**, **Python**, **Ruby**, **Go**, and **Docker**
- You upload your code
  - Elastic Beanstalk automatically handles the deployment
  - Deploys on servers such as Apache, NGINX, Passenger, Puma, and Microsoft Internet Information Services (IIS)