AWS Module 6 - Compute

By lemasyma

Posted Feb 10, 2021 • Updated Oct 3, 2021 • 14 min read • 1 views

Lien de la note Hackmd

Section 1: Compute services overview

AWS compute services

- Amazon EC2:
  - resizable virtual machine
- Amazon EC2 auto-scaling:
  - define conditions to launch or terminate EC2 instances
- Amazon ECR:
  - store and retrieve Docker images
- Amazon ECS:
  - Container orchestration service that supports Docker
- VMWare Cloud on AWS:
  - hybrid cloud without custom hardware
- AWS Elastic Beanstalk:
  - run and manage web app
- AWS Lambda:
  - serverless compute solution
- Amazon EKS:
  - run managed kubernetes on AWS
- Amazon LightSail:
  - building app or website
- AWS Batch:
  - running batch job at any scale
- AWS Fargate:
  - run containers
- AWS Outpost:
  - run AWS services in your on-premises data center
- AWS Serverless Repository:

Categorizing compute services

Services	Key Concepts	Characteristics	Ease of Use
• Amazon EC2	• Infrastructure as a service (IaaS) • Instance-based • Virtual machines	• 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	• Write and deploy code that executes 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	• Spin up and execute 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	• 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

- 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 pattern ?
  - Which configuration settings wll 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

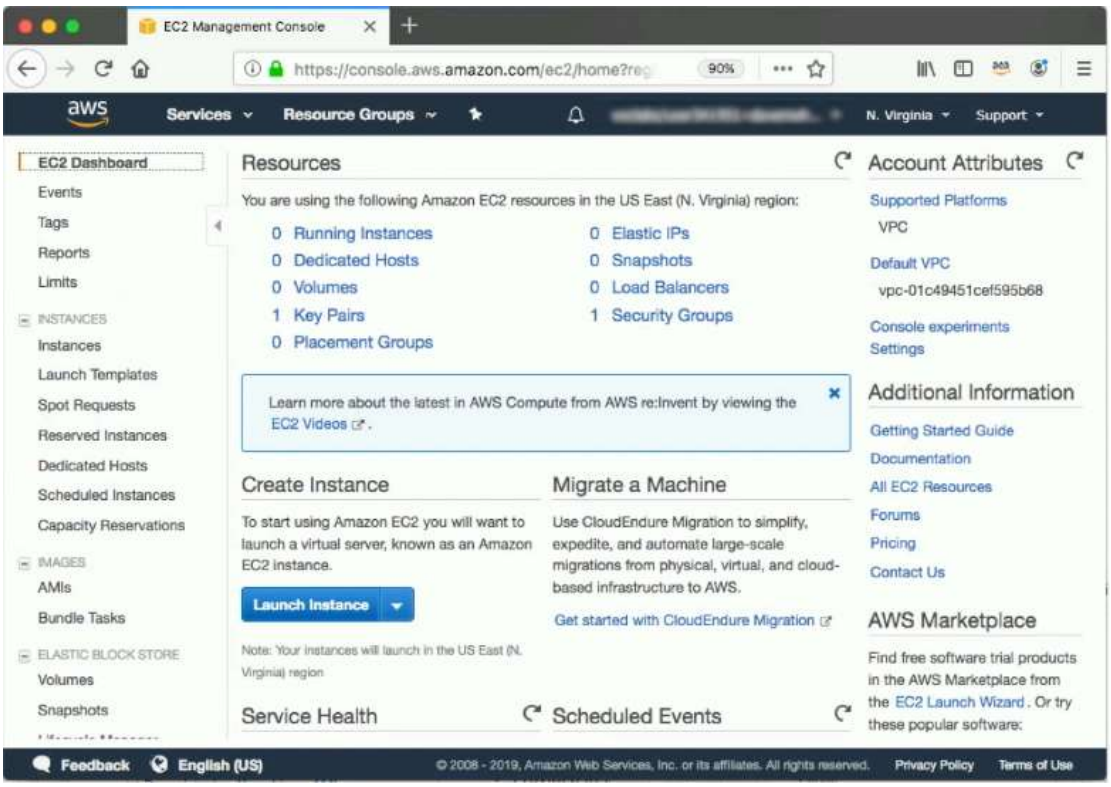
Amazon Elastic Compute Cloud (Amazon EC2)

Example uses of Amazon EC2 instances:

- App server
- web server
- Database server
- Game server
- Mail server
- Media server
- Catalog server
- File server
- Computing server
- Proxy server

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

Launching an amazon EC2 instance

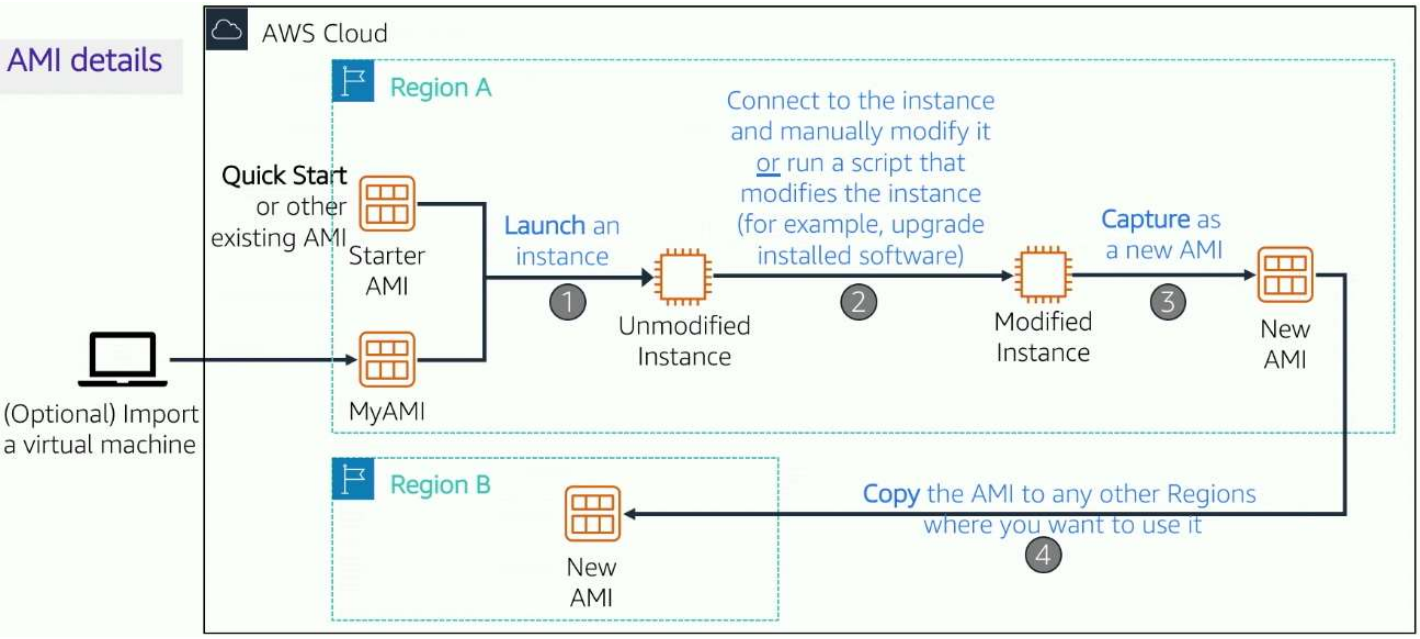


Nine key decisions when creating a EC2 instance.

1. Select an AMI

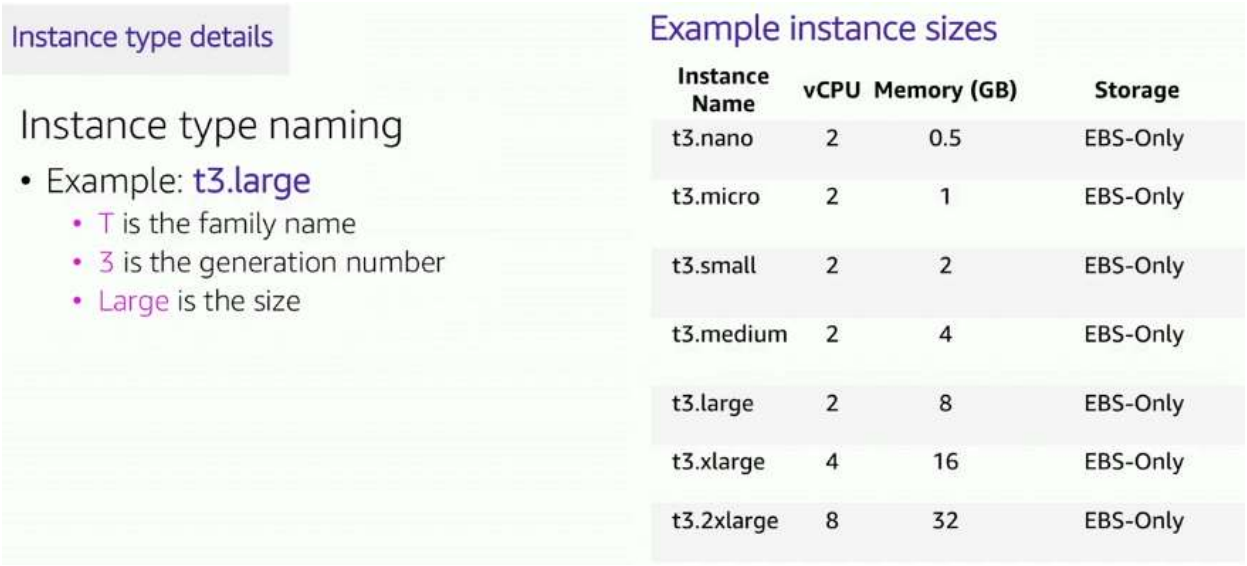
- Amazon Machine Image (AMI)
  - Is a template that is used to create an EC2 instance
  - Contains a *Windows* or *Linux* OS
  - Often has some software pre-installed
- AMI choices:
  - Quick Start
    - Linux and Windows AMIs provided by AWS
  - My AMIs
    - Any AMIs that you created

        - Pre-configured templates from third parties
    - Community AMIs
        - AMIs shared by others; use at you own risk

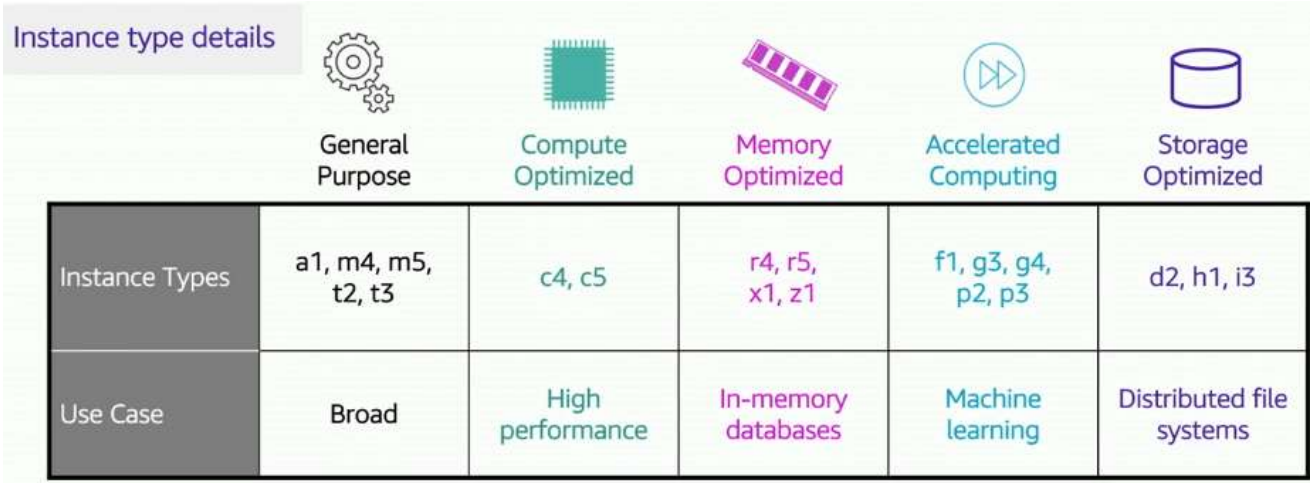


2. Select an instance type

- Consider you 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 computed
- Instance types offer *family, generation* and *size*



Based on use case



Networking features

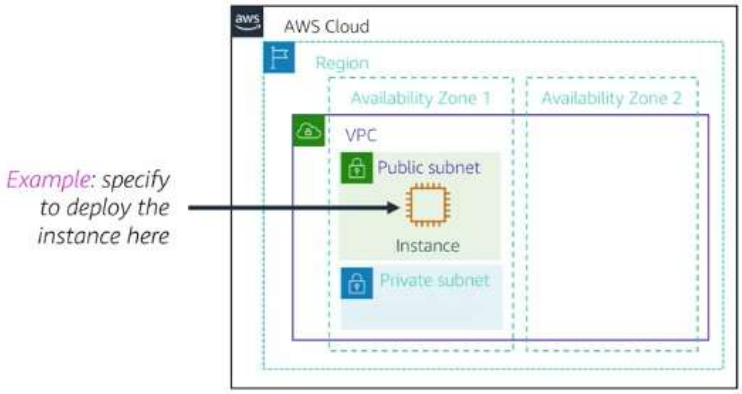
- The network bandwith (GBps) varies by instance type
- To maximize networking and bandwith performance of your instance type
  - If you have interdependent instances, launch them into a *cluster placement group*
  - Enable enhanced networking
- Enhanced networking types are supported on most instance types
- Enhanced networking types
  - Elastic Network Adapter (ENA): Supports network speeds of up to 100 Gpbds
  - Intel 82599 Virtual Function interface: Supports network speeds of up to 10 Gbps

Section 3: Amazon EC2 Part 2

3. Specify network settings

- Where should the instance be deployed ?

- Should a *public IP address* be automatically assigned ?
  - To make it internet-accessible

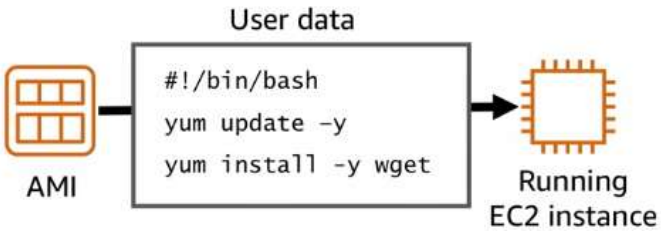


4. Attach IAM role (optional)

- Will software on the EC2 insrance need to interact with other AWS services ?
  - If yes, attach an appropriate IAM Role
- An AWS Identity and Access Management (IAM) role that is attache to an EC2 instance is kept in an **instance profile**
- You are *not* restricted to attaching a role only at instance launch
  - You can also attach a role to an instance that already exists

5. User data script (optional)

- Optionally specify a user data script at instance launch
- Use **user data** scripts to customize the runtime environment of your instance
  - Script executes the first time the instance starts
- Can be used strategically
  - Reduce the number of custom AMIs that you build and maintain



6. Specify storage

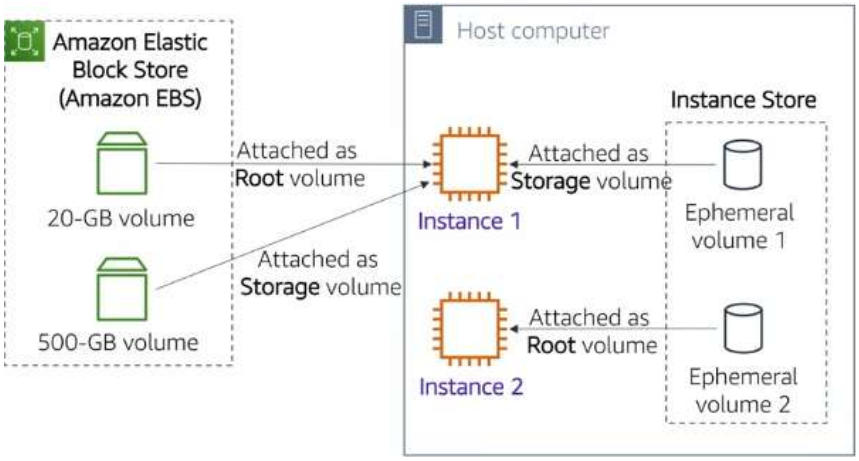
- Configure the *root volume*
  - Where the guest operating system is installed
- Attach *additional storage volumes* (optional)
  - AMI might already include more than one volume
- For each volume, specify:
  - The *size* of the disk (in GB)

        - Different types of SSDs and HDDs are available
    - If the volume will be deleted when the instance is terminated
    - If *encryption* should be used

Amazon EC2 storage options

- **Amazon Elastic Block Store (Amazon EBS)**
    - *Durable*, block-level storage volumes
    - You can stop the instance and start it again, and the data will still be there
- **Amazon Elastic Block Store**
    - Storage is provided on disls that are attached to the host computer where the EC2 instance is running
    - *If the instance stops, data stored here is deleted*
- Other options for storage (not for root volume)
    - Mount an **Amazon Elastic File System (Amazon EFS)** file system
    - Connect to **Amazon Simple Storage Service (Amazon S3)**

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 ?
        - The OS volume would survive
        - Any data stored on Amazon EBS would remain intact
        - Any data stored in ephemeral volume 1 would be lost
- **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)?
        - All data stored in ephemeral volume 2 would be lost, including the OS

Section 4: Amazon EC2 Part 3

A **tag** is a label that you can assign to an AWS resource

- Consists of a *key* and an optional *value*

- Tagging is how you can attach **metadata** to an EC2 instance
- Potential benefits from tagging - Filtering, automation, cost allocation and access control

8. Security group settings

A **security group** is a *set of firewall rules* that control traffic to the instance.

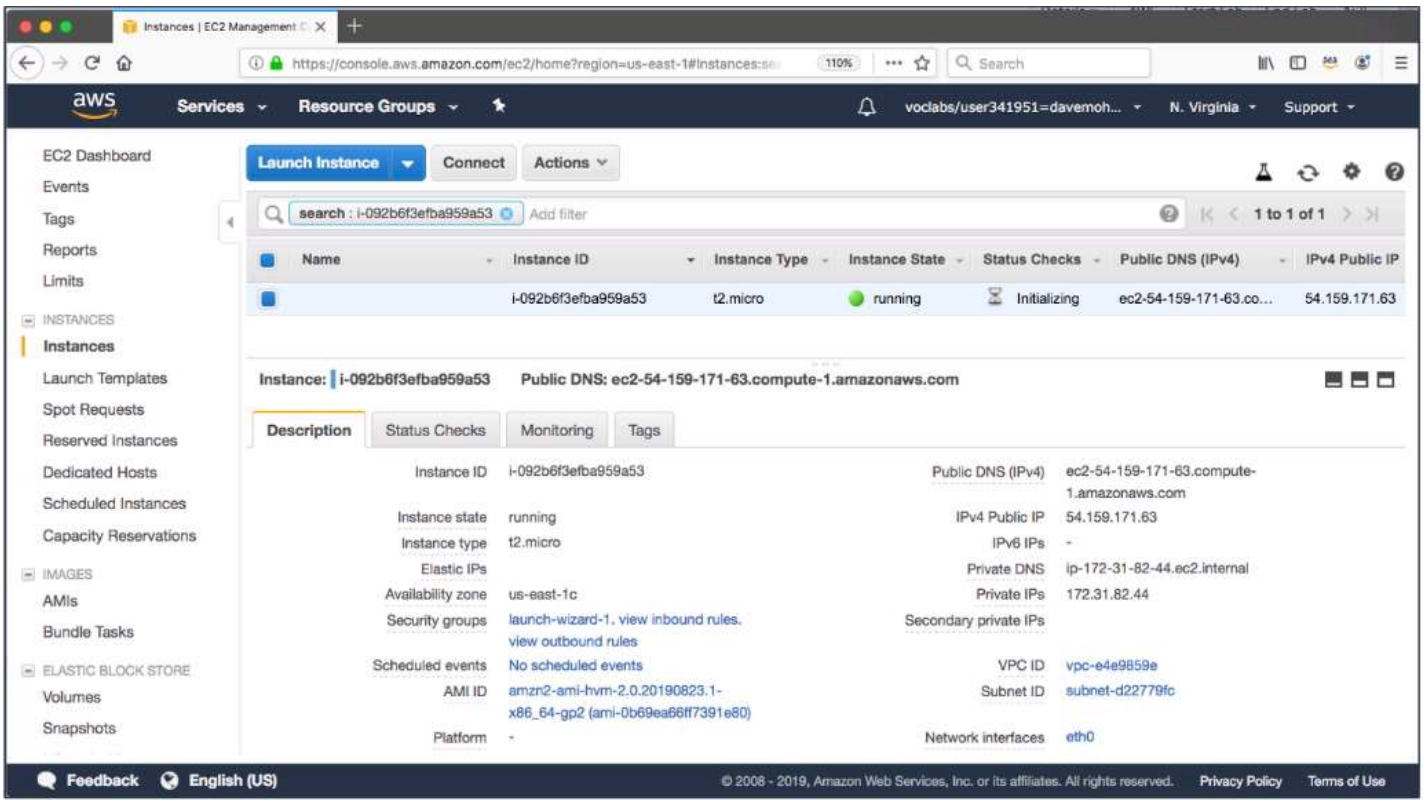
- It exsists *outside* of the instance's guest OS

Create **rules** that specify the **source** and which **ports** that network communications can use.

- Specify the **port** number and the **protocol**, such as TCP, UDP or ICMP
- Specify the **source** that is allowed to use the rule

9. Identify the key pair

- At instance launch, you specify an existing key pair *or* create a new key pair
- A **key pair** consists of
  - A *public key* that AWS stores
  - A *private key* file that you store
- It enables secure connections to the instance
- For **Windows AMIs**
  - Use the private key to obtain the administrator password that you need to log in to your instance
- For **Linux AMIs**
  - Use the private key to use SSH to securely connect to your instance



Another option: Launch an EC2 instance with the AWS CLI

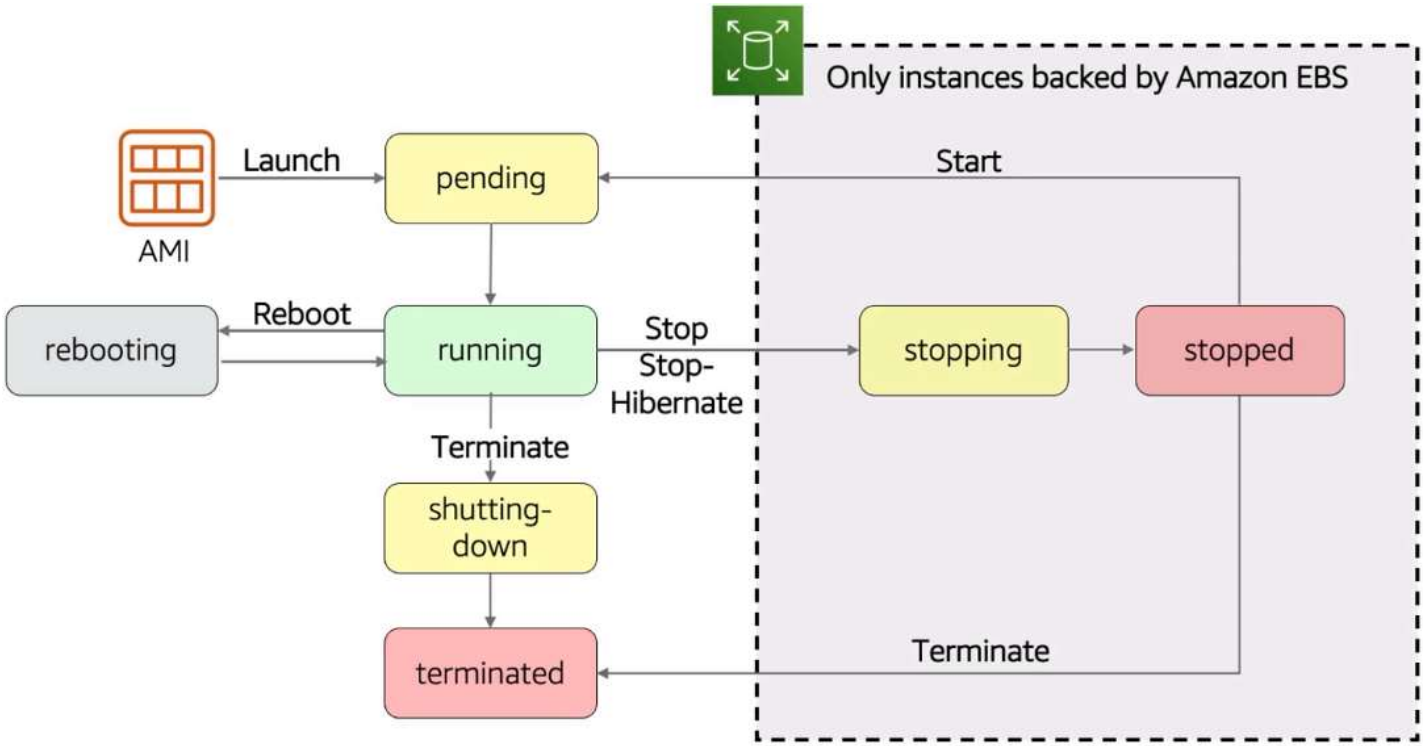
- EC2 instances can also be created programmatically

Shell

```
aws ec2 run-instances --image0id ami-1a2b3c4d --count 1 --instance-type c3.large \
--key-name MyKeyPair --security-groups MySecurityGroup --region us-east-1
```

This example shows how simple the command can be.

- This command assumes that the key pair and security group already exists
- More option could be specified



Consider using an Elastic IP address

- **Rebooting** an instanc will *not* change any IP addresses or DNS hostnames
- When an instance will *not* change any IP addresses or DNS hostnames
- When an instance is **stopped** and then **started** again
  - The *public* IPv4 address and *external* DNS hostname will change
  - The *private* IPv4 address and internal DNS hostname do *not* change
- If you require a persistent public IP address
  - Associate an *Elastic IP address* with the instance
- Elastic IP address characteristics
  - Can be associated with instances in the Region as needed
  - Remains allocated to your account until you choose to release it

EC2 instance metadata

> It is data about your instance

- While you are connected to the instance, you can view it
  - In a browser: `http://169.254.169.254/latest/meta-data/`
  - In a terminal window: `curl http://169.254.169.254/latest/meta-data/`
- Example retrievable values
  - Public IP address, private IP address, public hostname, instance ID, security groups, Region, Availability zone
  - Any user data specified at instance launch can also be accesse at: `http://169.254.169.254/latest/user-data/`
- It can be used to configure or manage a running instance

Amazon CloudWatch for monitoring

- Use **Amazon CloudWatch** to monitor EC2 instances
  - Provides near-real-time metrics
  - Provides charts in the Amazon EC2 console **Monitoring** tab
  - Maintains 15 months of historical data
- **Basic monitoring**
  - Default, no additional cost
  - Metric data sent to CloudWatch every 5 minutes
- **Detailed monitoring**
  - Fixed monthly rate for seven pre-selected metrics
  - Metric data delivered every 1 min

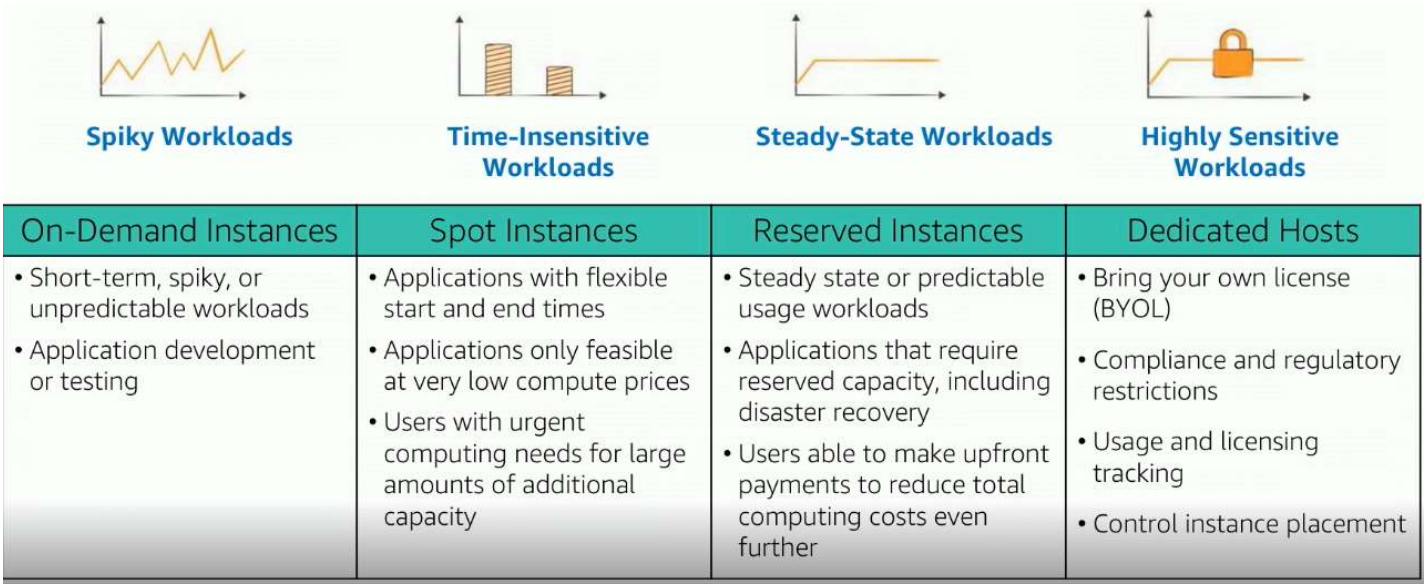
Section 5: Amazon EC2 Cost Optimization

Amazon EC2 pricing models

- **On-Demand Instances**
  - Pay by the hour
  - No long-term commitments
  - Elligible for the AWS Free Tier
- **Dedicated Hosts**
  - A physical server with EC2 instance capacity fully dedicated to your use
- **Dedicated instances**
  - Instances that run in a VPC on a hardware that is dedicated to a single customer
- **Reserverd Instances**
  - Full, partial, or no upfront payment for instance you reserve
  - Discount on hourly charge for that instance
  - 1-year or 3-year term
- **Scheduled Reserverd Instances**
  - Purchase a capacity reservation that is always available on a recurring schedule you specify
  - 1-year term
- **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

On-Demand Instances	Spot Instances	Reserved Instances	Dedicated
Low cost and flexibility	Large scale, dynamic workload	Predictability ensures compute capacity is available when needed	Save mone

Use cases



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 4 pillars of cost optimization

Pillar 1: Right size

- Provision instances to match the need
  - CPU, memory, storage and network throughput
  - Selct appropriate *instance types* for your use
- Use Amazon CloudWatch metrics
  - How idle are instances? When

- Best practice: right size, then reserve

Pillar 2: Increase elasticity

- **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 base on usage
  - Automated and time-based elasticity

Pillar 3: Optimal pricing model

- 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

- Reduce cost while maintaining storage performance and availability
- Resixe EBS volumes
- Changes EBS volumes 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 app need the instance to reside on Amazon EBS ?
  - Amazon S3 storage options with lifecycle policies can reduce costs

Measure, monitor and improve

- 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

Section 6: Container services

> **Containers** are a method of *operating system virtualization*

Benefits:

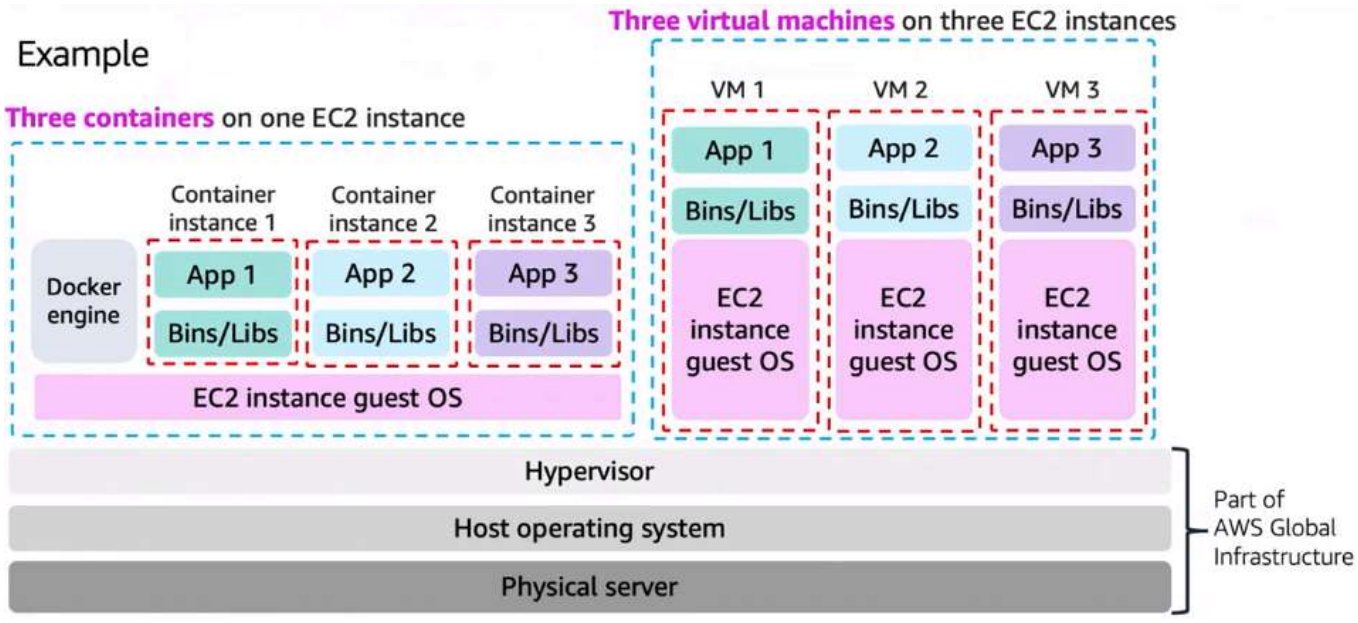
- Repeatable
- Self-contained environments
- Software runs the same in different environments
  - Developer's laptop, test, prod
- Faster to launch and stop or terminate than virtual machines

What is Docker ?

> **Docker** is a software platform that enables you to build, test, and deploy app quickly.

- You run containers on Docker
  - Containers are created from a template called an *image*
- A **container** has everything a software app needs to run

Containers vs VMs



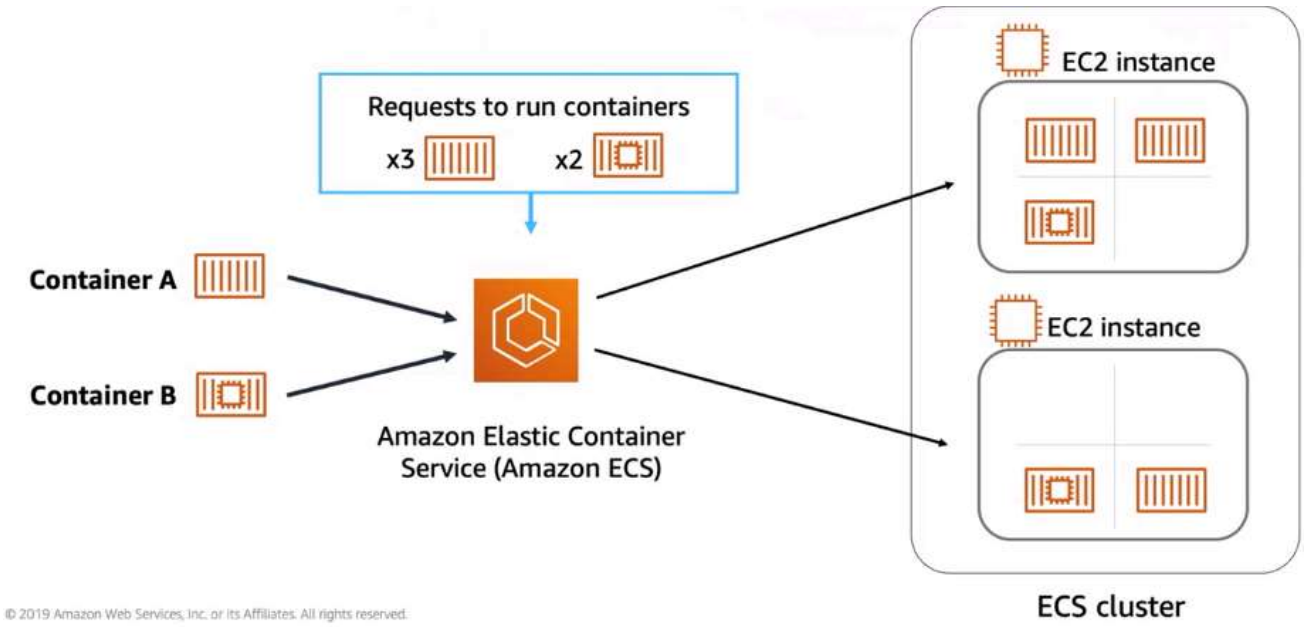
Amazon Elastic Container Service (Amazon ECS)

> A highly scalable, fast, **container management service**.

- Key benefit
  - Ocherstartes the running of Docker 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

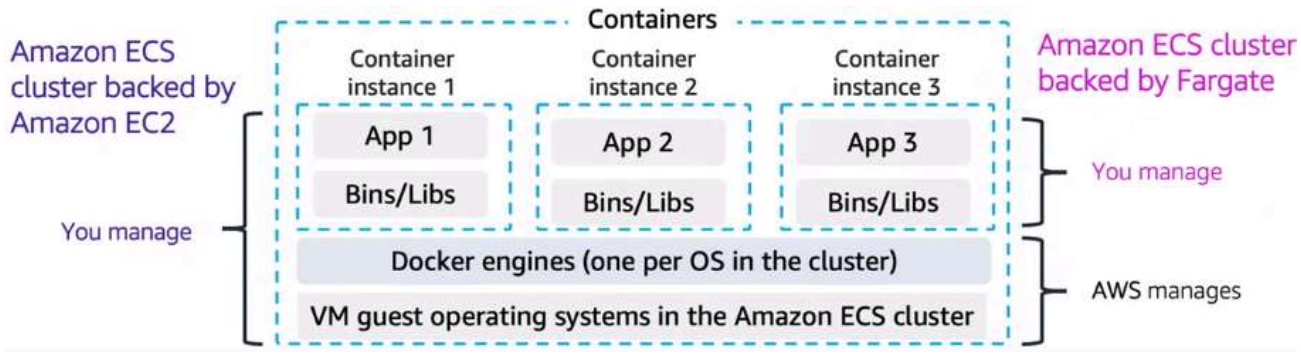
Amazon ECS orchestrates containers



Amazon ECS cluster options

Do you want to manage the Amazon ECS cluster that runs the containers ?

- Yes: create an **Amazon ECS cluster backed by Amazon EC2**
  - Provides more granular control over infrastructure
- No: create an **Amazon ECS cluster back by AWS Fargate**
  - Easier to maintain, focus on your app



What is Kubernetes ?

- Kubernetes is open source software for containers orchestration

- The same toolset can be used on premises and in the cloud
- Complements Docker
  - Docker enables you to run mutliple containers on a single OS host
  - Kubernetes **orchestrates** mutliple Docker hosts (nodes)
- Automates
  - Container provisioning
  - Networking
  - Load distribution
  - Scaling

Amazon Elastic Kubernetes Service (Amazon EKS)

- EKS
  - Enables you to run Kubernetes on AWS
  - Certified Kubernetes conformant
  - Supports Linux and Windows containers
  - Compatible with Kubernetes community tools and add-ons
- Use Amazon EKS to
  - Manage clusters of Amazon EC2 instances
  - Run containers that ar ochestrated by Kubernetes on those instances

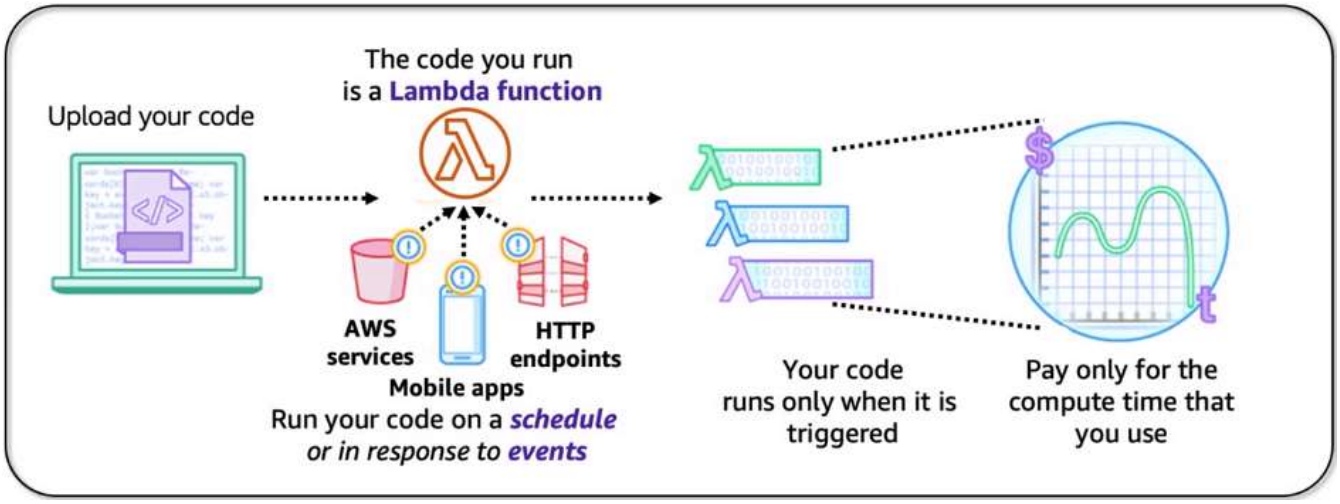
Amazon Elastic Container Registry (Amazon ECR)

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

- Supports
  - Team collab
  - Acces control
  - Third party integration
- Possible to use with Amazon EKS

Section 7: Introduction to AWS Lambda

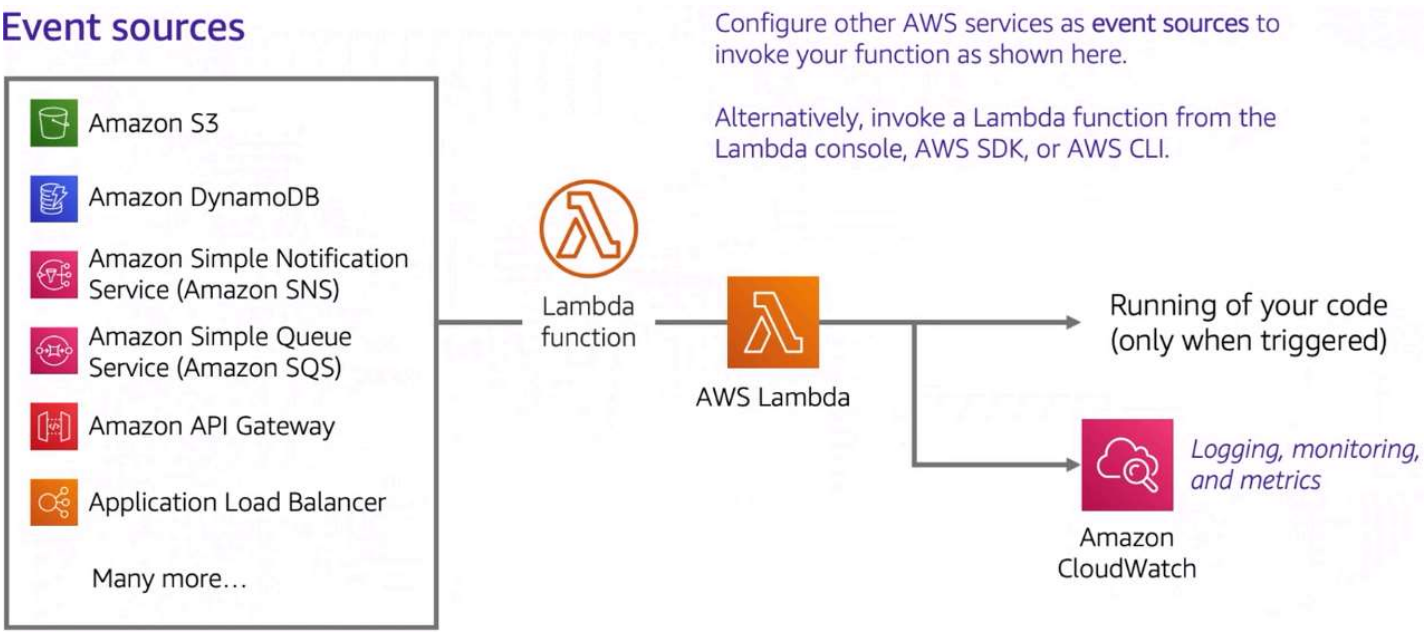
AWS Lambda is a **serverless** compute service.



Benefits of Lambda

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

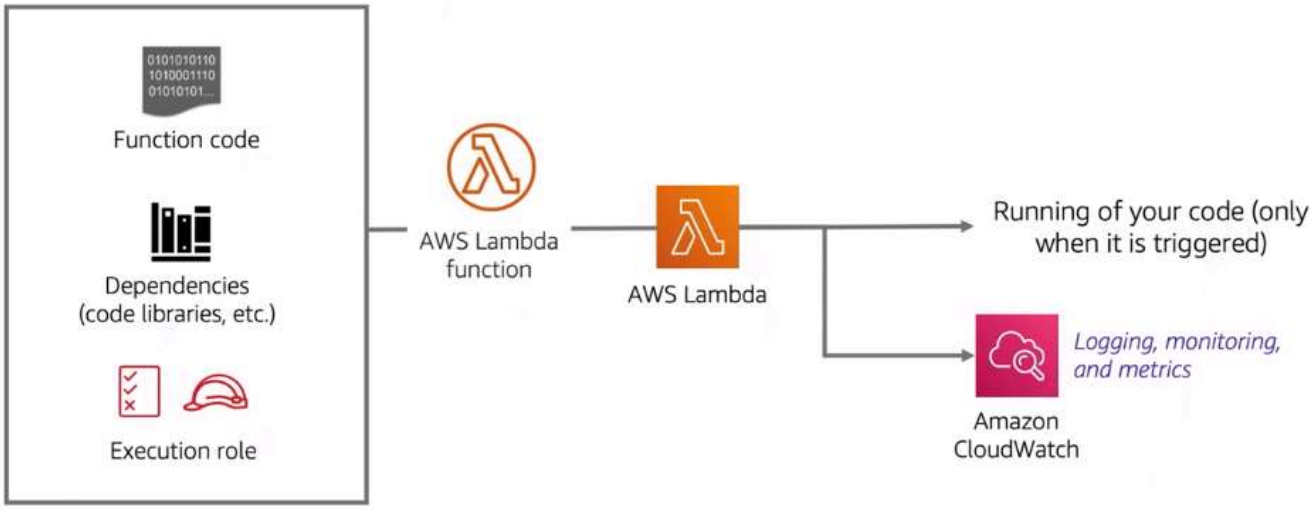
AWS Lambda event sources



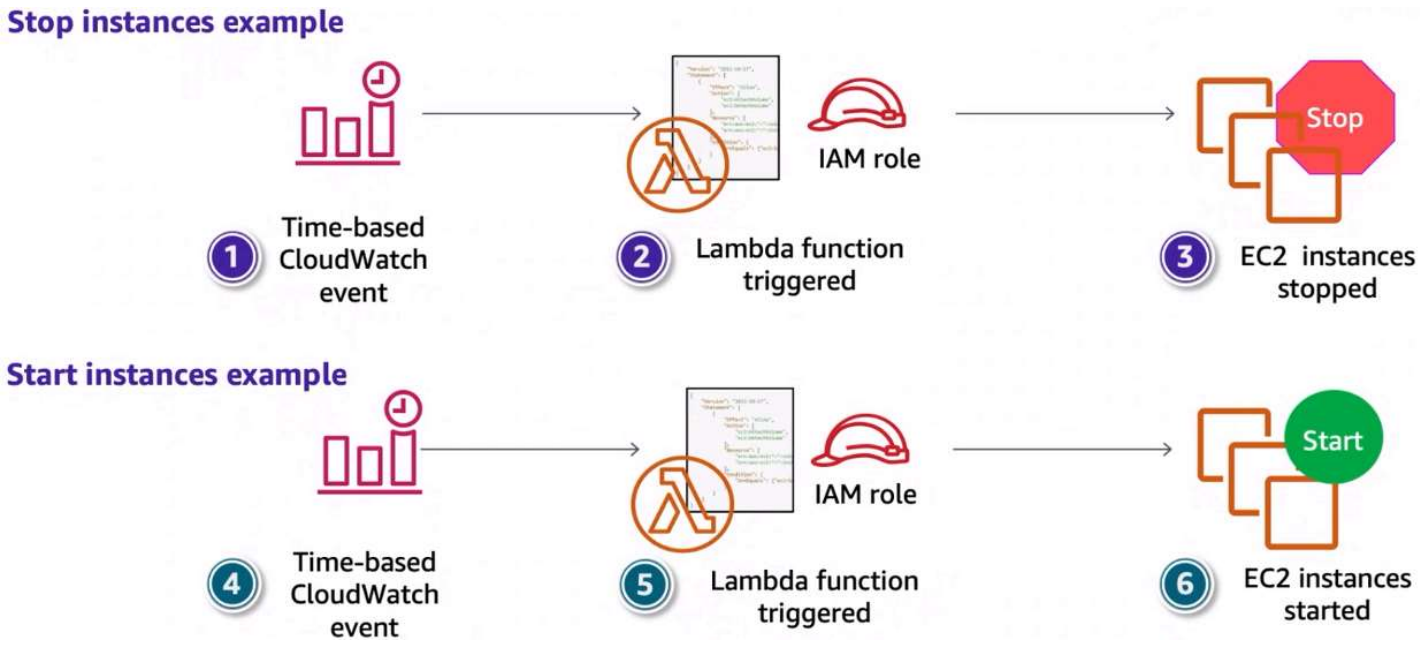
AWS Lambda function configuration

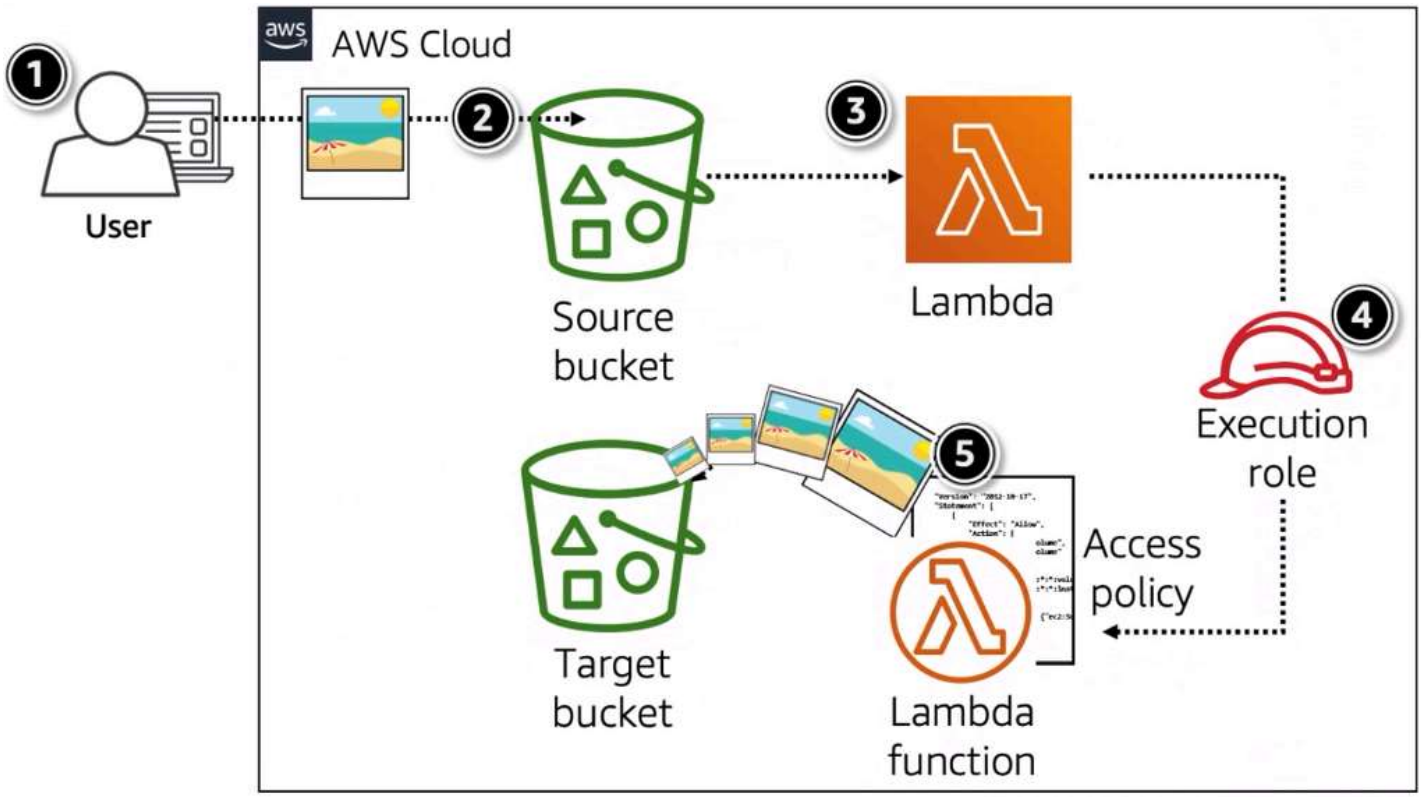
- Create lambda function: give a name
- Runtime environment

- Node.js
- Execution role to grant IAM permission to the function to interact with other services
- Configure the function
  - adding a trigger
- Add function code
- Specify the memory in megabytes (up to 3008MGB)
- Specify env variable



Schedule-based Lambda function example: start and stop EC2 instances





AWS Lambda limits

Soft limits per Region

- Concurrent executions = 1,000
- Function and layer storage = 75GB

Hard limits for individual function:

- Max function memory alloc = 3,008 MB
- Function tiemout = 15 min
- Deployement package size = 250 MB unzipped, including layers

Section 8: Introduction to AWS Elastic Beanstalk

AWS Elastic Beanstalk

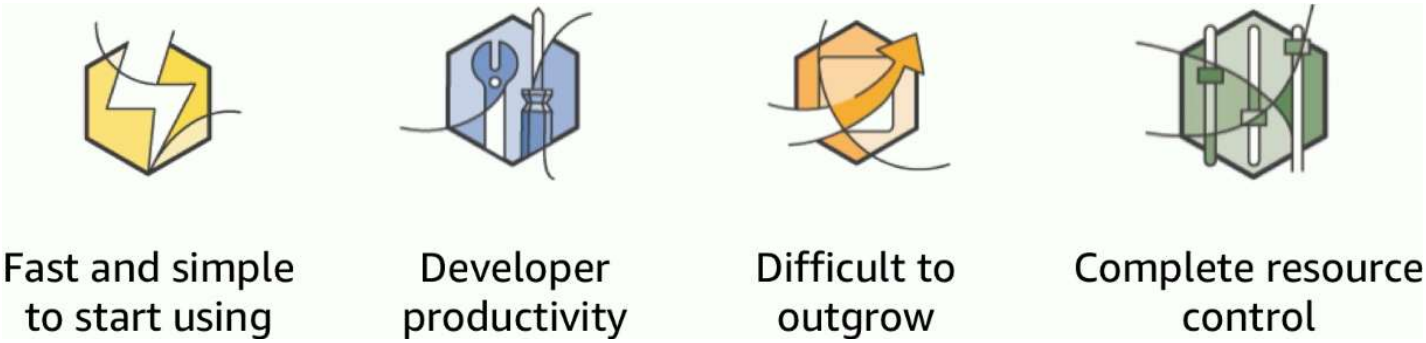
- An easy way to get *web app* up and running
- A *managed service* that automatically handles
  - Infra provisionning and config
  - Deployement
  - Load balancing
  - Automatic scaling
  - Health monitoring
  - Analysis and debugging
  - Logging

- Pay only for the underlying ressources that are used

AWS Elastic Beanstalk deployements

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

Benefits of Elastic Beanstalk



Wrap-up

Which AWS service helps developers quickly deploy resources which can make use of different programming languages, such as .Net and Java ?

1. AWS CloudFormation
2. AWS SQS
3. AWS Elastic Beanstalk
4. Amazon Elastic Compute Cloud (Amazon EC2)

▶ Answer

tronc commun S8 , AWS

tronc commun AWS S8

Further Reading

Feb 8, 2021

Feb 8, 2021

Feb 9, 2021

Lien de la note Hackmd Introduction Intro to cloud computing Advantages of cloud…

Lien de la note Hackmd Section 1: Fundamentals of pricing AWS pricing mode…

Lien de la note Hackmd Section 1: AWS Global Infrastructure The AWS Global…

OLDER
AWS Module 5 - Networking and Content Delivery

NEWER
AWS TD 2 - Module 5 and 6