

CSCI 5902 Adv. Cloud Architecting
Fall 2023
Instructor: Lu Yang

Modules 4 Adding a Compute Layer (Sections 1 - 3)
Sep 25, 2023

Housekeeping and feedback (1/3)



1. S3 Transfer Acceleration - multiple edge locations on the route?


- 1) As the data arrives at an edge location, it is routed to Amazon S3 over an optimized network path (<https://docs.aws.amazon.com/AmazonS3/latest/userguide/transfer-acceleration.html>)
- 2) AWS routing devices use special algorithms for real-time monitoring and calculation to find out quickly which route is optimal for a specific session at a certain moment. (<https://www.msp360.com/resources/blog/amazon-s3-transfer-acceleration-explained/>)
- 3) Where is the 'edge location' mentioned above? The nearest to the data source or the S3 bucket?

2. Why S3 bucket has to be globally unique?


S3 URI

 s3://lutestbucket000/index.html

Amazon Resource Name (ARN)

 arn:aws:s3:::lutestbucket000/index.html

Entity tag (Etag)

 9d6d80521775a4c7aaa59c05ee1deb4c

Object URL

 <https://lutestbucket000.s3.amazonaws.com/index.html>

- In S3, the **URI** is a resource identifier within the context of the S3 protocol. It is used for **API** access.
- An **ARN** is used **within** the AWS Cloud infrastructure to identify/access resources, while the URI is used externally identify/access these resources via APIs.
- **URL** is a unique web address that points to a **specific public bucket or object** within that bucket. This URL can be shared with anyone to grant read or write access to the bucket's contents, depending on the permissions set.

Housekeeping and feedback (2/3)



3. S3 prefix performance (https://www.reddit.com/r/aws/comments/lpjzex/please_eli5_how_s3_prefixes_speed_up_performance/?rdt=60179)
“3,500 PUT/COPY/POST/DELETE or 5,500 GET/HEAD requests per second per prefix in a bucket”

```
s3://awsexamplebucket/folderA/object-A1  
s3://awsexamplebucket/folderA/object-A2  
s3://awsexamplebucket/folderB/object-B1  
s3://awsexamplebucket/folderB/object-B2  
s3://awsexamplebucket/folderC/object-C1
```

You can GET 5.5K rps (requests per second) from each of the "folderA", "folderB", and "folderC".

So what you want to do is not to make paths long, just distinct from each other. Usually, you group similar/related content in "folders", just like you do with files on your computer. If files are uploaded by users, you can make "folders" by date, so each day is a separate path. Then you can make 5.5K GET rps for objects from each day (or hour, or second, depending on how small "folders" you make).

4. Difference of AWS Services and AWS Resources

An AWS resource is any entities that can be created or managed by an AWS service. Examples of resources include virtual machines, databases, and storage buckets.

AWS services, on the other hand, are the tools or platforms that you use to create, configure, and manage those resources. Examples of AWS services include EC2 (Elastic Compute Cloud), S3 (Simple Storage Service), and RDS (Relational Database Service). In short, resources are the instances you create and manage using AWS services.

Housekeeping items and feedback (3/3)



5. AWS Snow Family and AWS DataSync

(<https://cloud.in28minutes.com/aws-certification-moving-data-aws-on-prem-snowball-vs-snowmobile-vs-datasync>)

- If you have available connectivity between your location and AWS, your first try is [AWS DataSync](#)
- The Snow Family of devices should only be used when you have connectivity challenges with the internet/cloud

AWS Academy Cloud Architecting

Module 4: Adding a Compute Layer



Module overview



Sections

1. Architectural need
2. Adding compute with Amazon EC2
3. Choosing an AMI to launch an Amazon EC2 instance
4. Selecting an Amazon EC2 instance type
5. Using user data to configure an Amazon EC2 instance
6. Adding storage to an Amazon EC2 instance
7. Amazon EC2 pricing options
8. Amazon EC2 considerations

Labs

- Guided Lab: Introducing Amazon EFS
- Challenge Lab: Creating a Dynamic Website for the Café

Module objectives



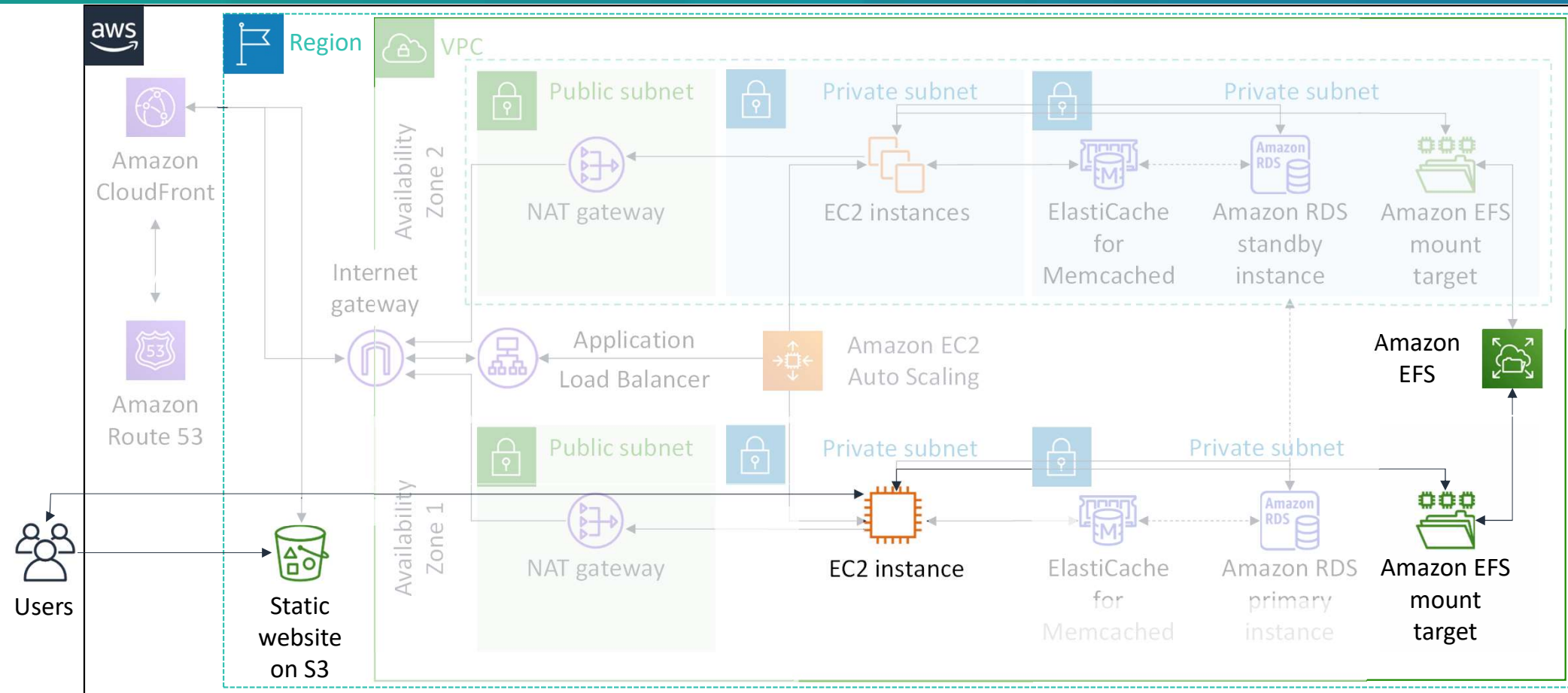
At the end of this module, you should be able to:

- Identify how Amazon Elastic Compute Cloud (Amazon EC2) can be used in an architecture
- Explain the value of using Amazon Machine Images (AMIs) to accelerate the creation and repeatability of infrastructure
- Differentiate between the EC2 instance types
- Recognize how to configure Amazon EC2 instances with user data
- Recognize storage solutions for Amazon EC2
- Describe EC2 pricing options
- Determine the placement group given an architectural consideration
- Launch an Amazon EC2 instance

Module 4: Adding a Compute Layer

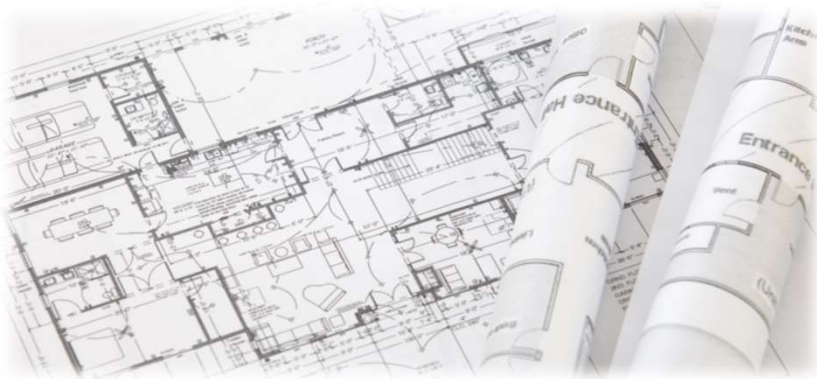
Section 1: Architectural need

Compute as part of a larger architecture



Café business requirement









The café wants the website to display more than static content and to provide dynamic capabilities. They want to introduce online ordering for customers, and enable café staff to view submitted orders.



Module 4: Adding a Compute Layer

Section 2: Adding compute with Amazon EC2

AWS runtime compute choices (1/3)

Virtual Machines (VMs)	Containers	Platform as a Service (PaaS)	Serverless	Specialized Solutions
 Amazon Elastic Compute Cloud (Amazon EC2)	 Amazon Elastic Container Service (Amazon ECS)	 AWS Elastic Beanstalk	 AWS Lambda	 AWS Outposts
 Amazon Lightsail			 AWS Fargate	 AWS Batch

Higher infrastructure control and customization

Faster application deployment











Fully managed services

Different compute services are available to meet the needs of different use cases.

This module will discuss Amazon EC2.

AWS runtime compute choices (2/3)



Virtual Machines (VMs)	Containers	Platform as a Service (PaaS)	Serverless	Specialized Solutions
 Amazon Elastic Compute Cloud (Amazon EC2)	 Amazon Elastic Container Service (Amazon ECS)	 AWS Elastic Beanstalk	 AWS Lambda	 AWS Outposts
 Amazon Lightsail			 AWS Fargate	 AWS Batch
<p>AWS offers Amazon EC2 and Lightsail for hosting applications. Amazon EC2 is a mix of multiple services and has its own individual features used to create a single architecture. Amazon EC2 instances are meant for small to complex architecture. Lightsail, on other hand, is an integrated product of services offered by AWS. Lightsail is better for small to medium scale workloads.</p>	<p>Amazon Elastic Container Service is a service that supports Docker containers and helps orchestrate scalable, containerized applications. With this, customers can access all the features of their applications via the cloud, including launching, stopping and querying features via API calls.</p>	<p>It is a solution that runs web applications and services that are developed in languages such as Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker. Organizations must simply upload the code, while Elastic Beanstalk handles provisioning, scaling, and load balancing</p>	<p>AWS Lambda is a service that allows users to run code without the need for managing servers or provisions. Lambda allows serverless computing without administration or any charges outside of the price for the time spent computing. Instead, it applies the precise amount of computing power to a ZIP file or container image to run code for any request or event.</p> <p>AWS Fargate is an engine for Amazon Compute that runs containers without the need for close server or cluster maintenance on the part of the customer. This means that organizations don't need to spend precious time building and managing the infrastructure behind their applications.</p>	<p>AWS Outpost enables customers to tap into AWS infrastructure to make use of services and operating models from any data center—or even on-premises servers. Batch computing is the execution of a series of programs ("jobs") on one or more computers without manual intervention. AWS Batch is a set of batch management capabilities that enables developers, scientists, and engineers to easily and efficiently run hundreds of thousands of batch computing jobs on AWS.</p>

AWS runtime compute choices (3/3)



- **Containers on AWS Overview:** <https://www.youtube.com/watch?v=AYAh6YDXuho&t=23s>

- **When is Elastic Beanstalk the Best Method for Managing Docker Containers on AWS?**

For businesses new to AWS or new to the containerization concept, just getting started with Docker, or developing new applications, Elastic Beanstalk may be the best approach to support Docker containers. Elastic Beanstalk offers a simple interface, allows Docker images to be pulled from public or private registries, and coordinates the deployment of multiple Docker containers to Amazon ECS clusters. Elastic Beanstalk gives you less control over application scaling and capacity but makes deploying Docker containers on AWS ever so straightforward.

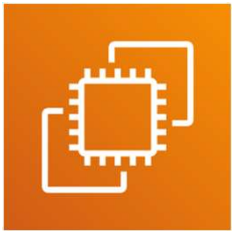
- **When is Elastic Container Service the Best Method for Managing Docker Containers on AWS?**

In comparison to Elastic Beanstalk, Elastic Container Service provides greater control over application architectures and orchestration of Docker containers. You specify the size and number of cluster nodes and determine if auto-scaling should be used.

Elastic Container Service uses tasks to launch Docker containers. A task includes the container definition, providing the ability to group containers in sets that launch together then terminate simultaneously. ECS provides significantly greater flexibility and customization in scheduling and CPU and memory utilization. In addition, ECS does not require special integration efforts to work with many other AWS services.

Elastic Container Service is appropriate when you need to run microservices that require integration with other AWS services, or use custom or managed schedulers to run batch workloads on EC2 On-Demand, Reserved, or Spot Instances. Businesses wanting to containerize legacy code and migrate it to AWS without needing to rewrite code should take the ECS option.

Applications or workflows comprised of loosely coupled, distributed services running on various platforms or accessing widely-distributed data source can also benefit by using Elastic Container Service. Reference: <https://www.missioncloud.com/blog/resource-docker-containers-on-aws-use-elastic-beanstalk-or-elastic-container-services>



Amazon Elastic
Compute Cloud
(Amazon EC2)

Amazon EC2 provides resizable compute capacity in the cloud.

- Provides virtual machines (servers)
- Provisions servers in minutes
- Can automatically scale capacity up or down as needed
- Enables you to pay only for the capacity that you use

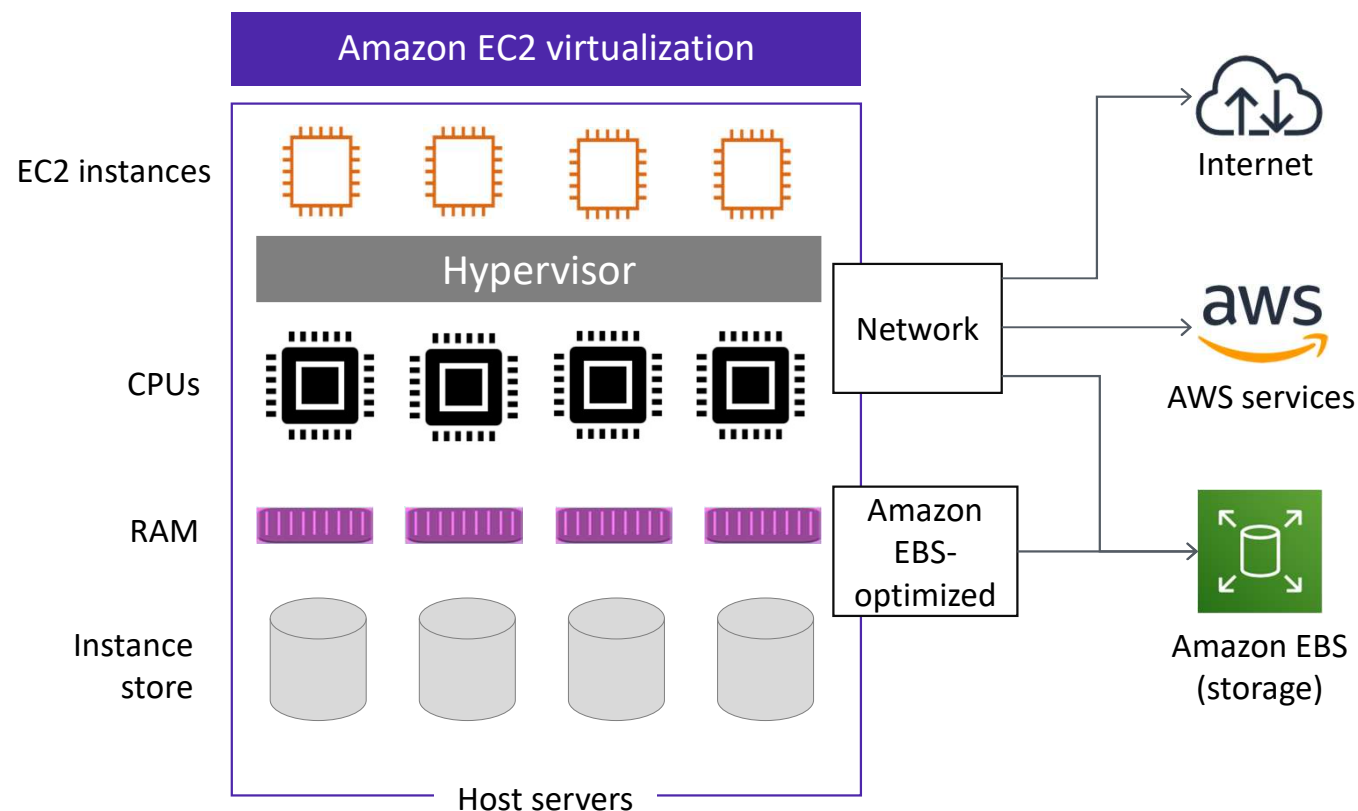
Why is it called **Elastic Compute Cloud**?

- **Elastic** because you can easily increase or decrease the number of servers you run to support an application automatically. You can also increase or decrease the size of existing servers
- **Compute** because most users run servers to host running applications or process data, which require compute resources. These resources include processing power (CPU) and memory (RAM)
- **Cloud** because the EC2 instances that you run are hosted in the cloud

EC2 instances

An EC2 instance is a **virtual machine** that runs on a physical host.

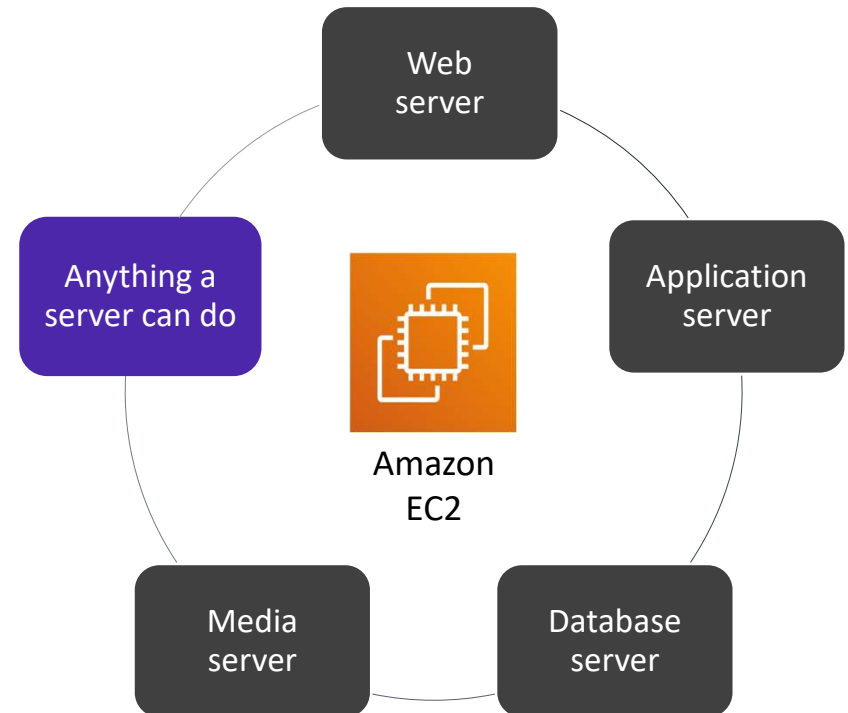
- You can choose different configurations of CPU and memory capacity
- Supports different storage options
 - Instance store
 - Amazon Elastic Block Store (Amazon EBS)
- Provides network connectivity



Amazon EC2 use cases

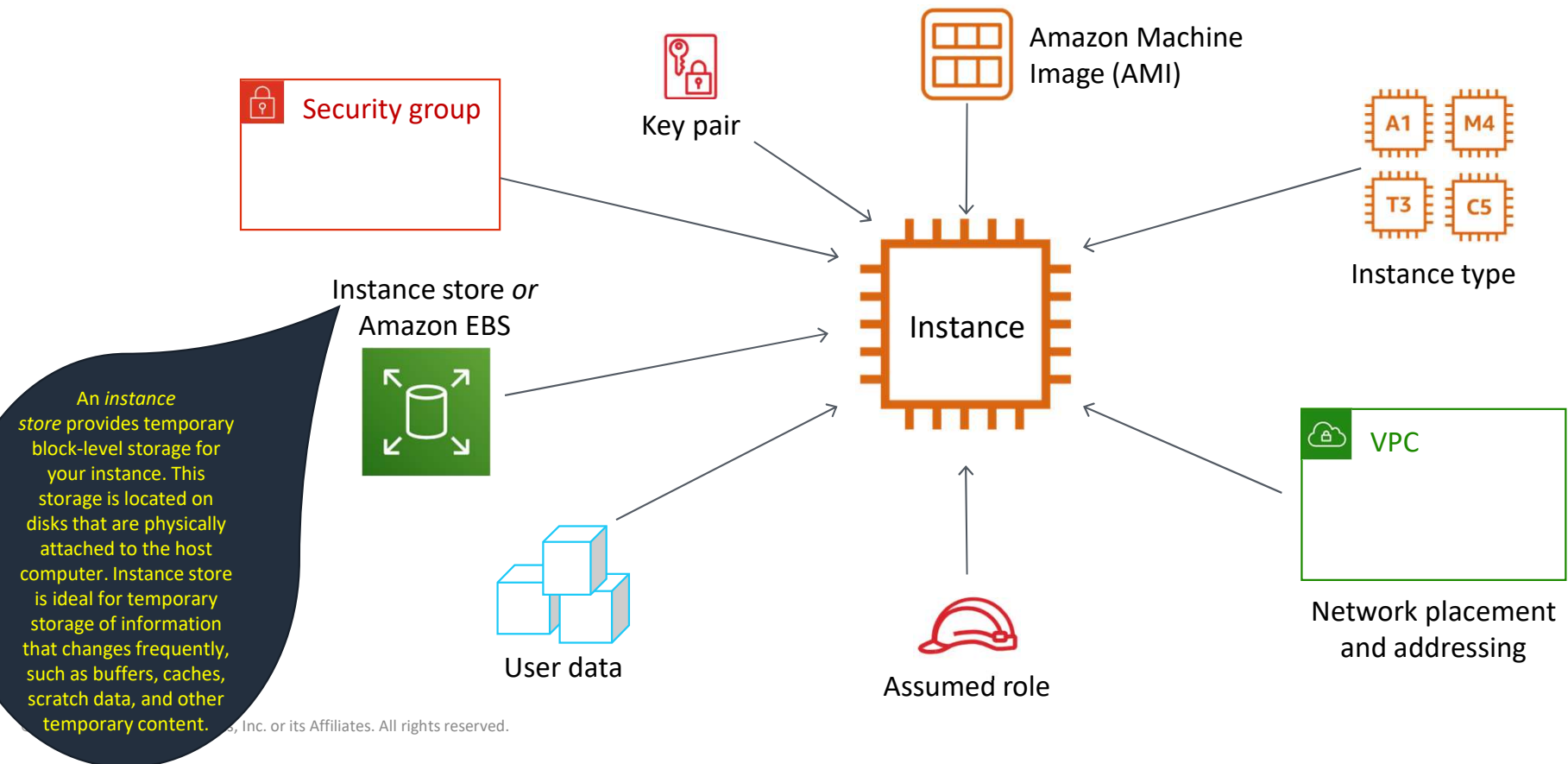
Use Amazon EC2 when you need:

- Complete control of your computing resources, including *operating system* and *processor type*
- Options for optimizing your compute costs –
 - *On-Demand Instances*, *Reserved Instances*, and *Spot Instances*
 - *Savings Plans*
- Ability to run any type of workload, for example –
 - Simple websites
 - Enterprise applications
 - High performance computing (HPC) applications
- EC2 capabilities
 - Renting virtual machines (EC2)
 - Storing data on virtual drives (EBS)
 - Distributing load across machines (ELB)
 - Scaling the services using an auto-scaling group (ASG)



Provisioning an EC2 instance

Essential instance launch configuration parameters



Section 2 key takeaways



- Amazon EC2 enables you to run Microsoft Windows and Linux **virtual machines** in the cloud.
- You can use an EC2 instance when you need **complete control of your computing resources** and want to **run any type of workload**.
- When you launch an EC2 instance, you must choose an **AMI** and an **instance type**. Launching an instance involves specifying configuration parameters, including **network**, **security**, **storage**, and **user data** settings.

Module 4: Adding a Compute Layer

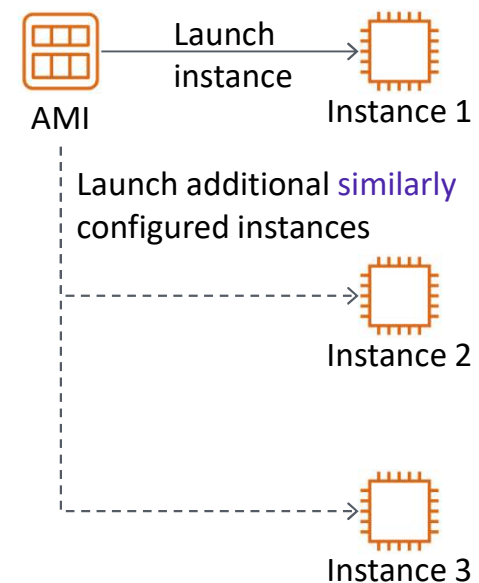
Section 3: Choosing an AMI to launch an EC2 instance

Amazon Machine Image (AMI)

An **AMI** provides the information that is needed to launch an instance, including:

- A **template** for the root volume
 - Contains the guest operating system (OS) and perhaps other installed software
- **Launch permissions**
 - Control which AWS accounts can access the AMI
- **Block device mappings**
 - Specifies any storage volumes to attach to the instance

Create multiple instances from the same AMI



AMI benefits

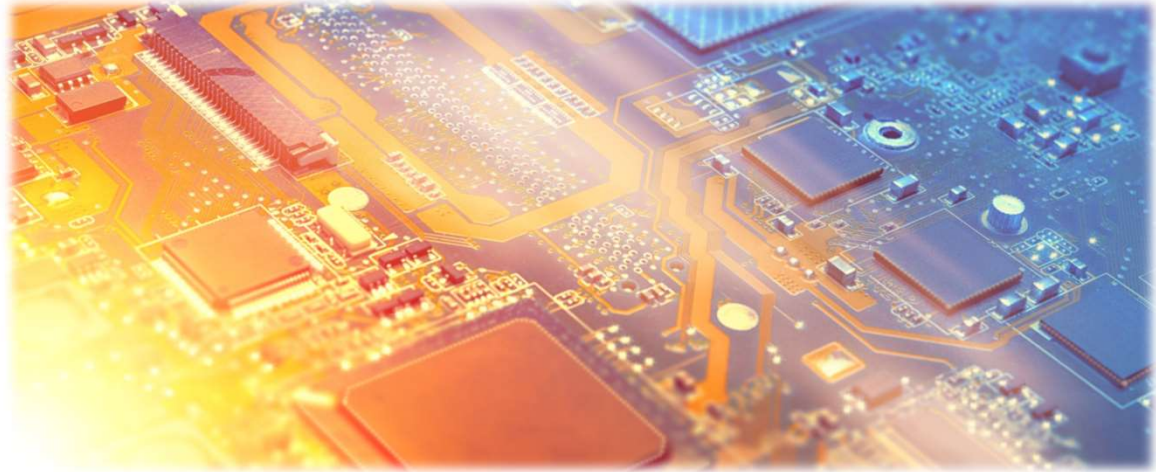
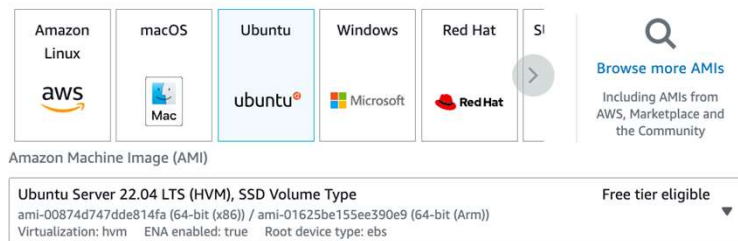


- **Repeatability**
 - An AMI can be used repeatedly to launch instances with efficiency and precision
- **Reusability**
 - Instances launched from the same AMI are identically configured
- **Recoverability**
 - You can create an AMI from a configured instance as a restorable backup
 - You can replace a failed instance by launching a new instance from the same AMI

Choosing an AMI

Choose an AMI based on:

- Region
- Operating system
 - Microsoft Windows, Linux, or MacOS
- Storage type of the root device
- Architecture
- Virtualization type
 - ParaVirtual (PV)
 - Hardware Virtual Machine (HVM)



AMI sources:

- **Quick Start** – *Linux and Microsoft Windows AMIs that are provided by AWS.*
- **My AMIs** – *Any AMIs that you create.*
- **AWS Marketplace** – *Pre-configured templates from third parties.*
- **Community AMIs** – *AMIs shared by others. Use at your own risk.*

Instance store-backed versus Amazon EBS-backed AMI



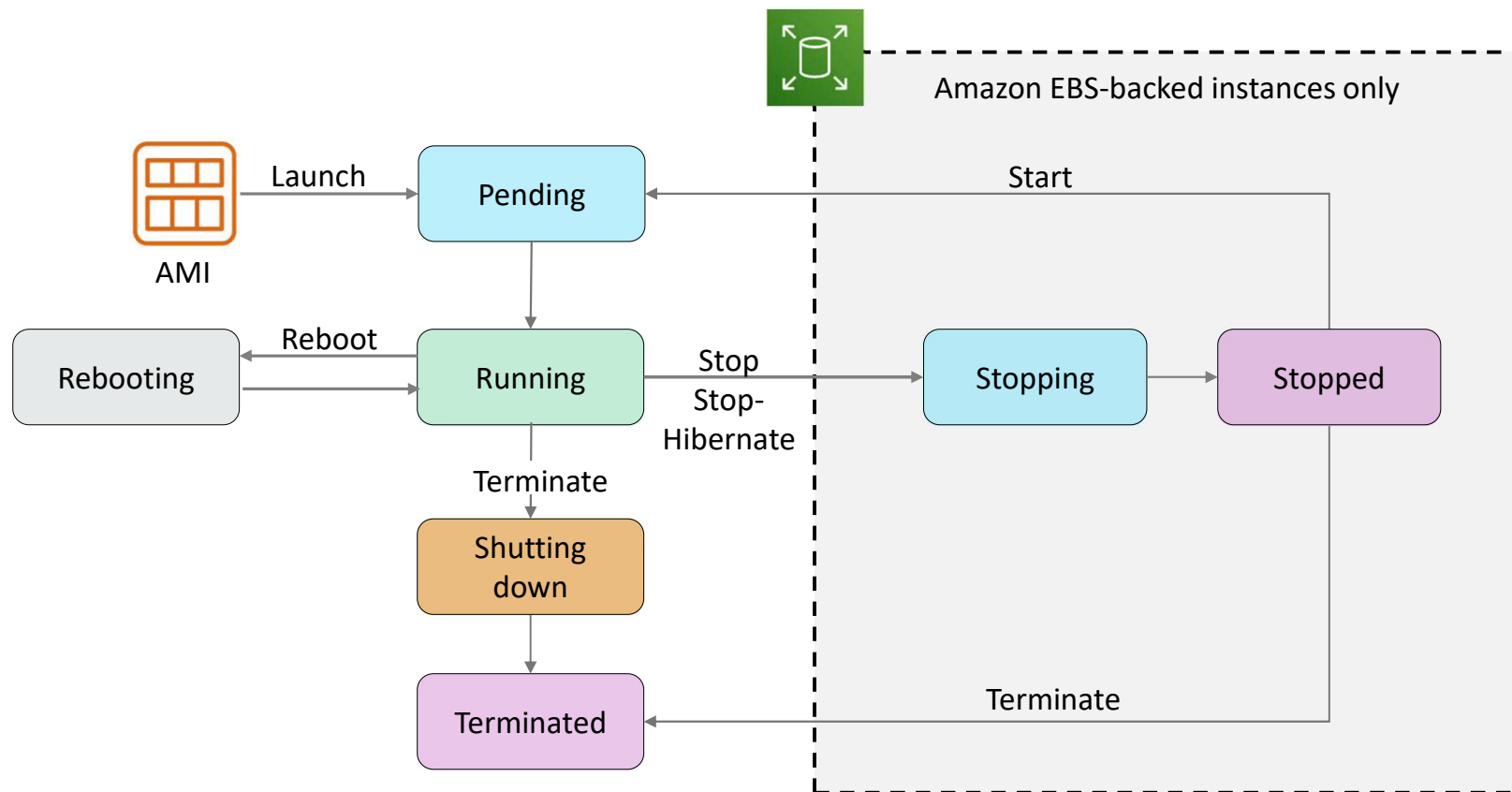
Characteristic	Amazon EBS-Backed Instance	Instance Store-Backed Instance
Boot time for the instance	Boots faster	Takes longer to boot
Maximum size of root device	16 TiB	10 GiB
Ability to stop the instance	Can stop the instance	Can't stop the instance, only reboot or terminate it
Ability to change the instance type	Can change the instance type by stopping instance	Can't change the instance type because the instance can't be stopped
Instance charges	You are charged for instance usage, EBS volume usage, and storing your AMI as an EBS snapshot	You are charged for instance usage and storing your AMI in Amazon S3

the root device for an instance launched from the AMI is an Amazon EBS volume

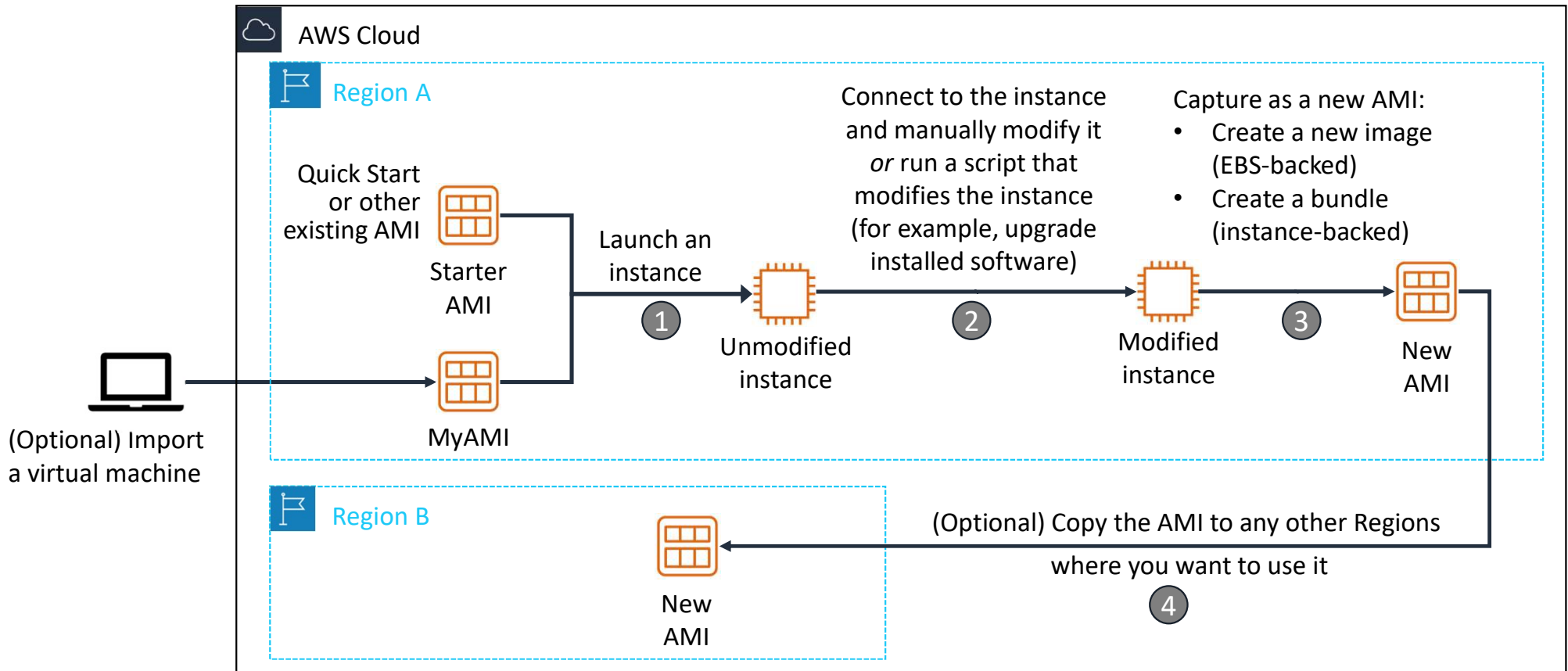
the root device for an instance launched from the AMI is an instance store volume created from a template stored in Amazon S3

The charges associated with AMIs do not depend on whether they are public or private. Whether it is public or private does not matter, it depends on the AMI's storage category for the root device.

Amazon EC2 instance lifecycle



Creating a new AMI



EC2 Image Builder



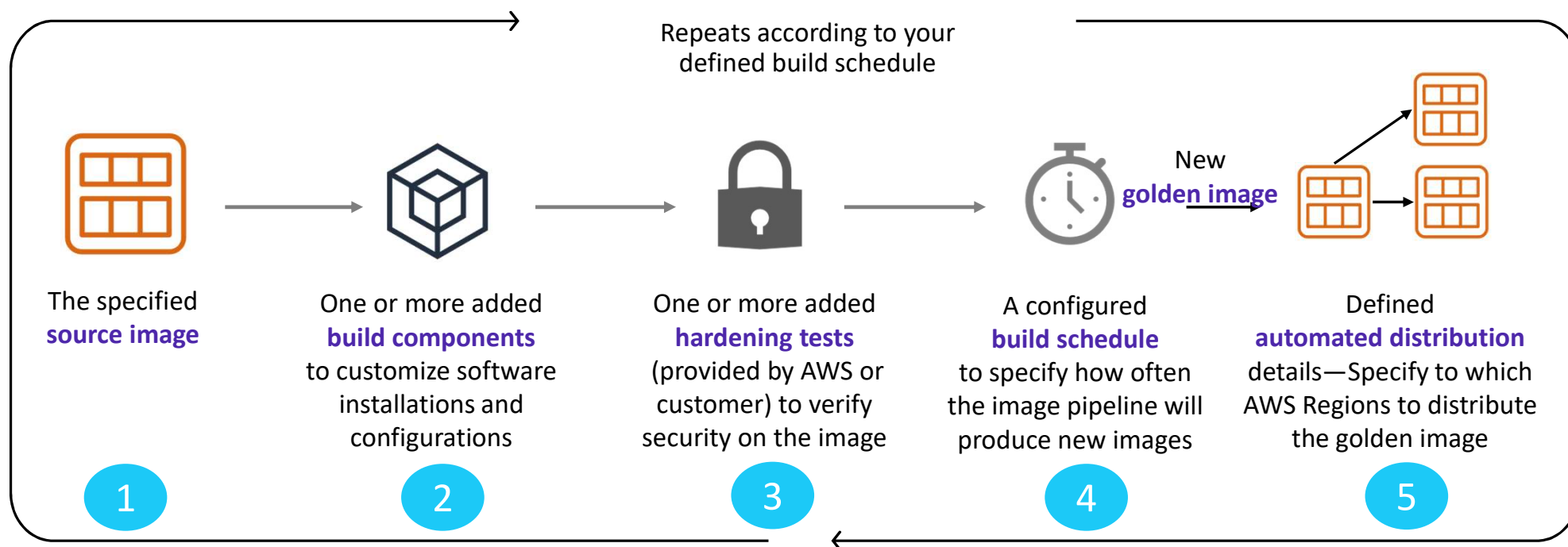
EC2 Image Builder

EC2 Image Builder automates the **creation**, **management**, and **deployment** of up-to-date and compliant **golden VM images**.

- Provides a graphical interface to create image-building pipelines
- Creates and maintains **Amazon EC2 AMIs** and on-premises VM images
- Produces secure, validated, and up-to-date images
- Enforces version control

How EC2 Image Builder works

An EC2 Image Builder **image pipeline**



Section 3 key takeaways



- An **AMI** provides the information that is needed to launch an EC2 instance
- For best performance, use an AMI with **HVM virtualization type**
- Only an instance launched from an Amazon EBS-backed AMI **can be stopped and started**
- An AMI is available in a **Region**