Amazon EC2 (Elastic Compute Cloud)

Amazon Elastic Compute Cloud (Amazon EC2) is a web-based service that allows businesses to run application programs in the Amazon Web Services (AWS) public cloud. Amazon EC2 allows a developer to spin up virtual machines (VMs), which provide compute capacity for IT projects and cloud workloads that run with global AWS data centers.

An AWS user can increase or decrease instance capacity as needed within minutes using the Amazon EC2 web interface or an application programming interface (API). A developer can code an application to scale instances automatically with AWS Auto Scaling. A developer can also define an autoscaling policy and group to manage multiple instances at once.

How EC2 works

To begin using EC2, developers sign up for an account at Amazon's AWS website. They can then use the AWS Management Console, the AWS Command Line Tools (CLI), or AWS Software Developer Kits (SDKs) to manage EC2.

A developer then chooses EC2 from the AWS Services dashboard and 'launch instance' in the EC2 console. At this point, they select either an Amazon Machine

Image (AMI) template or create an AMI containing an operating system, application programs, and configuration settings. The AMI is then uploaded to the Amazon S3 and registered with Amazon EC2, creating an AMI identifier. Once this has been done, the subscriber can requisition virtual machines on an as-needed basis.

Data only remains on an EC2 instance while it is running, but a developer can use an Amazon Elastic Block Store volume for an extra level of durability and Amazon S3 for EC2 data backup.

VM Import/Export allows a developer to import on-premises virtual machine images to Amazon EC2, where they are turned into instances.

EC2 also offers Amazon CloudWatch which monitors Amazon cloud applications and resources, allowing users to set alarms, view graphs, and get statistics for AWS data; and AWS Marketplace, an online store where users can buy and sell software that runs on AWS.

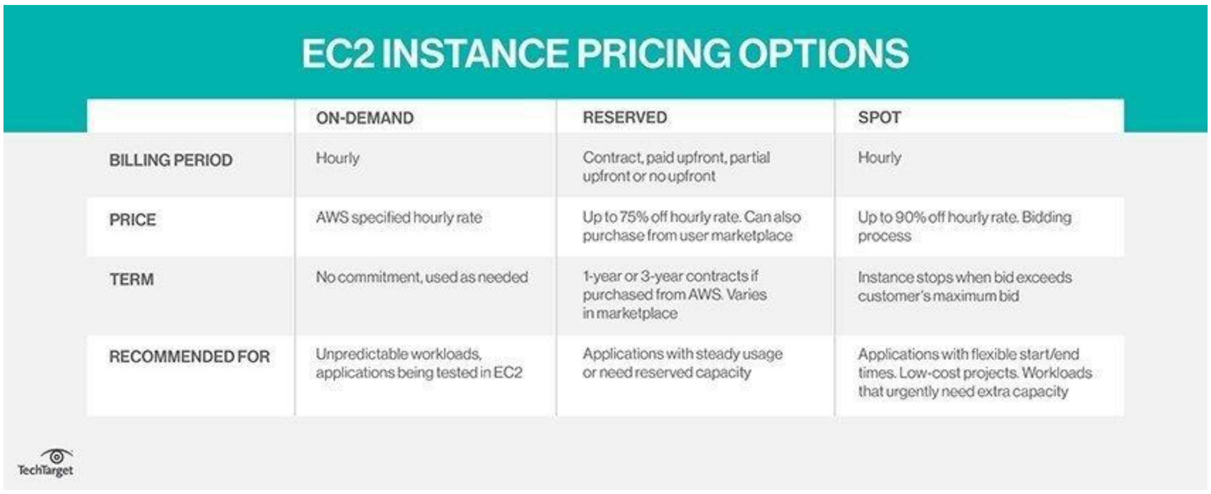
Amazon EC2 instance types

Instances allow developers to expand computing capabilities by 'renting' virtual machines rather than purchasing hardware. An EC2 instance is used to run applications on the Amazon Web Services infrastructure.

Amazon EC2 provides different instance types, sizes and pricing structures designed for different computing and budgetary needs. In addition to general purpose instances, Amazon EC2 offers an instance type for compute, memory, accelerated computing, and storage-optimized workloads. AWS limits how many instances a user can run in a region at a time, depending on the type of instance. Each instance type comes with different size options corresponding to the CPU, memory and storage needs of each enterprise.

Cost

On-Demand instances allow a developer to create resources as needed and to pay for them by the hour. Reserved instances (RIs) provide a price discount in exchange for one and three-year contract commitments -- a developer can also opt for a convertible RI, which allows for the flexibility to change the instance type, operating system or tenancy. There's also an option to purchase a second-hand RI from the Amazon EC2 reserved instances marketplace. A developer can also submit a bid for spare Amazon EC2 capacity, called Spot instances, for a workload that has a flexible start and end time. If a business needs dedicated physical server space, a developer can opt for EC2 dedicated hosts, which charge hourly and let the business use existing server-bound software licenses, including Windows Server and SQL Server.



EC2 INSTANCE PRICING OPTIONS

	ON-DEMAND	RESERVED	SPOT
BILLING PERIOD	Hourly	Contract, paid upfront, partial upfront or no upfront	Hourly
PRICE	AWS specified hourly rate	Up to 75% off hourly rate. Can also purchase from user marketplace	Up to 90% off hourly rate. Bidding process
TERM	No commitment, used as needed	1-year or 3-year contracts if purchased from AWS. Varies in marketplace	Instance stops when bid exceeds customer's maximum bid
RECOMMENDED FOR	Unpredictable workloads, applications being tested in EC2	Applications with steady usage or need reserved capacity	Applications with flexible start/end times. Low-cost projects. Workloads that urgently need extra capacity

TechTarget

A breakdown of Amazon EC2 instances and their associated prices.

Benefits

Getting started with EC2 is easy, and because EC2 is controlled by APIs developers can commission any number of server instances at the same time to quickly increase or decrease capacity. EC2 allows for complete control of instances which makes operation as simple as if the machine were in-house.

The flexibility of multiple instance types, operating systems, and software packages and the fact that EC2 is integrated with most AWS Services -- S3, Relational Database Service (RDS), Virtual Private Cloud (VPC) -- makes it a secure solution for computing, query processing, and cloud storage.

Challenges

Resource utilization -- developers must manage the number of instances they have to avoid costly large, long-running instances.

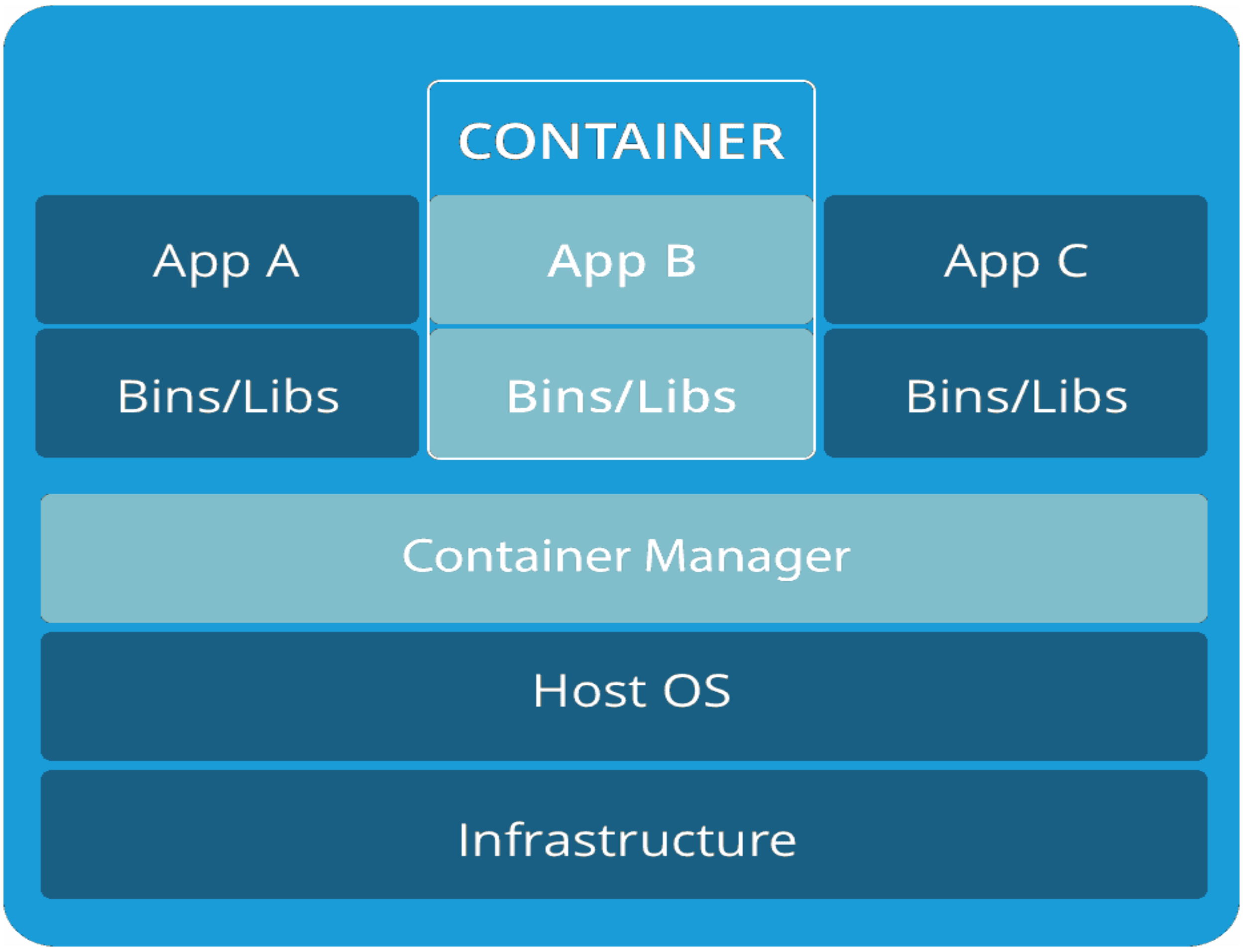
Security -- developers must make sure that public facing instances are running securely.

Deploying at scale -- running a multitude of instances can result in cluttered environments that are difficult to manage.

Management of AMI lifecycle -- developers often begin by using default Amazon Machine Images. As computing needs change, custom configurations will likely be required.

Ongoing maintenance -- Amazon EC2 instances are virtual machines that run in Amazon's cloud. However, they ultimately run on physical hardware which can fail. AWS alerts developers when an instance must be moved due to hardware maintenance. This requires ongoing monitoring.

How Does Container Work



Container Architecture
A container requires an operating system, supporting programs and libraries, and system resources to run a specific program. When working inside a container, you can create a template of an environment you need. The container essentially runs a snapshot of the system at a particular time, providing consistency in the behavior of an app.

The container shares the host's kernel to run all the individual apps within the container. The only elements that each container requires are bins, libraries, and other runtime components.

Pros Of Container

- Containers can be as small as 10MB and you can easily limit their memory and CPU usage. So, they are lightweight.
- Since they are small in size, they can boot up faster and can be quickly scaled too.
- Containers are exemplary when it comes to **Continous Integration and Continous Deployment** (CI/CD) implementation.

Read this blog to know about what is Kubernetes Pod which is an important component of Kubernetes.
Cons Of Container

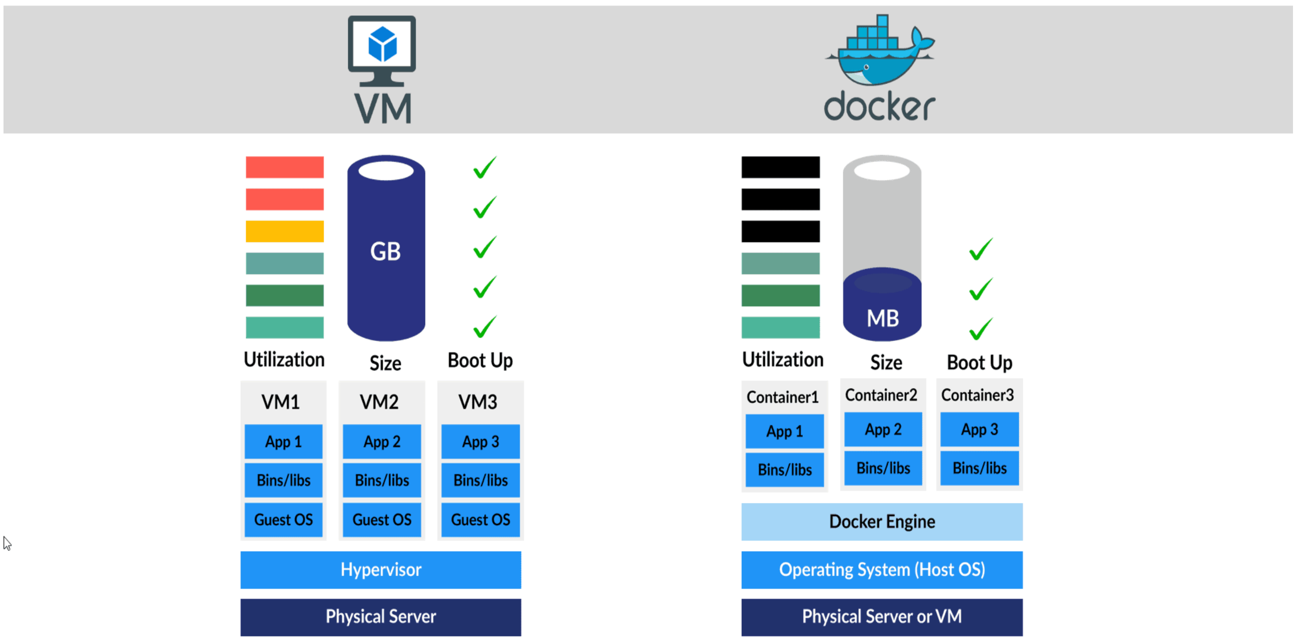
- Since the containers run on host OS, it has a dependency on the host underlying host Operating System.
- Containers cannot all by themselves cannot provide security at a commendable level.
- When the container is deleted if the data inside the container is lost. You will have to add Data Volumes in order to store the data.

Popular Container Providers:

- **Docker**
- **Rocket – rkt**
- **Linux containers – LXC**
- **CRI-O**
- **containerd**

- **Standard:** Docker created the industry standard for containers, so they could be portable anywhere
- **Lightweight:** Containers share the machine's OS system kernel and therefore do not require an OS per application, driving higher server efficiencies and reducing server and licensing costs

- **Secure:** Applications are safer in containers and Docker provides the strongest default isolation capabilities in the industry



	Docker	Virtual Machines (VMs)
Boot-Time	Boots in a few seconds.	It takes a few minutes for VMs to boot.
Runs on	Dockers make use of the execution engine.	VMs make use of the hypervisor.
Memory Efficiency	No space is needed to virtualize, hence less memory.	Requires entire OS to be loaded before starting the surface, so less efficient.
Isolation	Prone to adversities as no provisions for isolation systems.	Interference possibility is minimum because of the efficient isolation mechanism.
Deployment	Deploying is easy as only a single image, containerized can be used across all platforms.	Deployment is comparatively lengthy as separate instances are responsible for execution.
Usage	Docker has a complex usage mechanism consisting of both third party and docker managed tools.	Tools are easy to use and simpler to work with.

Docker solves problems like: missing or incorrect application dependencies such as libraries, interpreters, code/binaries, users; Example: running a Python or Java application with the right interpreter/VM or an 'legacy' third party application that relies on an old glibc

Developers benefit from Docker, yes. But when you think of the process of going into production, Docker is a real game changer:

- standardised packaging of software including all dependencies (docker images)
- standardised configuration (environment variables)
- standardised monitoring (container's stdout)
- standardised scalability (more containers)
- standardised error-handling (well, sort of, sometimes restarting a container helps)
- explicit distinction between stateless (app container) and stateful parts (volumes, db containers, ...)

Finally the handover from Dev to Ops doesn't have to be a pain anymore.