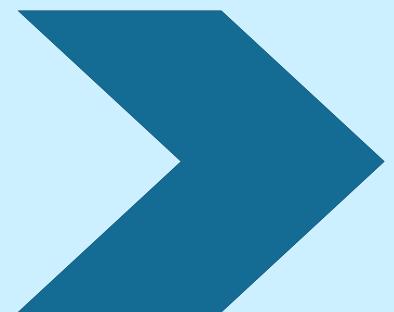


AWS EC2:

EC2 is NOT a virtual machine. It's more than that....



Kaustubh Nichit
kaustubhnichit09@gmail.com



FIRST, WHAT IS EC2?

EC2 = Elastic Compute Cloud

- **Virtual servers** in the cloud that you can launch in minutes.
- You choose the **OS, CPU/RAM, storage, network & security** - AWS handles the rest.



It's a virtual computer in the cloud, letting you install software, host websites, run databases, or perform complex tasks without managing physical hardware.



LAUNCHING AN EC2 INSTANCE

STEP 1 - YOU CHOOSE AN AMI

AMI = Pre-configured OS template (like **Ubuntu, Windows, Amazon Linux, custom images**).

It defines which OS your EC2 will run.

STEP 2 - CHOOSE INSTANCE TYPE

Instance type controls:

- CPU
- RAM
- Network bandwidth
- EBS bandwidth

Examples:

- t2.micro → small apps
- m5.large → general purpose
- c5 → compute-heavy
- r5 → memory-heavy



LAUNCHING AN EC2 INSTANCE

STEP 3 - ATTACH STORAGE (EBS VOLUME)

EBS = Elastic Block Storage

Behaves like a virtual **hard disk**.

You choose:

- **Size** (e.g., 20GB, 50GB)
- **Type** (gp3, io2, etc.)

Your EC2 boots from this disk.

STEP 4 - NETWORKING SETUP

Your EC2 lives inside a **VPC** (Virtual Private Cloud).

AWS assigns:

- **IP address**
- **Subnet**
- **Routing rules**

This decides if the instance:

- Can access the **internet**
- Is **isolated** internally
- Is part of a **private/public** subnet



LAUNCHING AN EC2 INSTANCE

STEP 5 - APPLY SECURITY GROUPS

Security Group = virtual firewall.

You define:

- Allowed **inbound** traffic (e.g., SSH 22, HTTP 80)
- Allowed **outbound** traffic

Secure by **default**: all traffic is **blocked** until allowed.

STEP 6: IAM ROLE (OPTIONAL BUT VERY POWERFUL)

Attach a role if EC2 needs access to:

- S3
- DynamoDB
- CloudWatch
- Secrets Manager

IAM Role = temporary, secure permissions without using access keys.

This prevents credential leakage.



BEFORE EC2: EVEN VIRTUALIZATION HAD LIMITATIONS

Earlier, companies bought **physical servers** and used a **hypervisor** to split them into **multiple VMs**.

This improved utilization...

But companies still had to manage:

- Hardware failures
- Disk replacements
- Firmware upgrades
- Hypervisor patching
- OS upgrades
- Cooling & power
- Network switches & firewalls
- On-site engineers for maintenance, etc

Example:

You partition a big server into 5 VMs →

But YOU still manage everything below the **VM layer**.

Operational burden = **VERY** high.



EC2 CHANGES THE ENTIRE INFRASTRUCTURE MODEL

With EC2, AWS manages:

- Physical servers
- Host patching
- Hardware failures
- Power & cooling
- Network backbone
- Data center security



You only manage:

- OS (if not using managed services)
- Application
- Configuration
- Access controls

This is the REAL value of cloud computing.



LAUNCHING AN EC2 INSTANCE

STEP 7 - KEY PAIR (FOR LOGIN)

For Linux EC2: use .pem file to SSH.

For Windows EC2: decrypt admin password & RDP.

Key = your login identity.

STEP 8 - BEHIND-THE-SCENES

When you click “Launch Instance” AWS:

- Allocates hardware
- Loads the AMI
- Mounts EBS volume
- Attaches network interfaces
- Applies security groups
- Boots OS
- Ensures health
- Makes instance available

All in seconds.



COMMON MISTAKES SECTION

MISTAKE 1: WRONG INSTANCE TYPE = WASTED MONEY

Choosing bigger instances than needed → unnecessary billing.

Right-sizing = essential.

MISTAKE 2: USING EC2 FOR EVERYTHING

EC2 ≠ always best choice.

Sometimes better options:

- Lambda
- ECS Fargate
- DynamoDB
- API Gateway

EC2 is amazing, but not always required.



COMMON MISTAKES SECTION

MISTAKE 3: STORING SECRETS ON EC2

NEVER store **passwords/API keys** inside EC2.

Use:

- Secrets Manager
- SSM Parameter Store

again, depends on security groups!



WHAT IF YOUR EC2 INSTANCE FAILS?

AWS may try to **restart** or **move** your EC2 instance if something goes wrong...

BUT that's **not enough** to keep your application always **available**.

If you want your application to stay **online** even when one EC2 instance **crashes**, you should use:

- **Auto Scaling** → automatically create new EC2 instances when needed
- **Load Balancer** → sends traffic to healthy instances
- **Multiple Availability Zones** → run your EC2 in more than one data center

This way, if one instance or one data center has a problem, your application **still keeps running**.



**IF YOU UNDERSTAND EC2
DEEPLY,
THE REST OF AWS BECOMES
10* EASIER.**



Kaustubh Nichit
kaustubhnichit09@gmail.com

