Why AMI is needed, what are the advantages, where have you used it ?

What is an AMI (Amazon Machine Image)?

An **Amazon Machine Image (AMI)** is a **pre-configured template** that contains everything needed to **launch an EC2 instance**, including:

- The **OS** (e.g., Amazon Linux, Ubuntu, Windows)
- Application **server**, libraries, and dependencies
- Pre-installed **software** or custom configurations
- Any **user-defined settings** or startup scripts

Why is AMI Needed?

Imagine you're setting up a server with:

- Ubuntu 22.04
- Apache web server
- Python 3.11
- A Django application
- Custom firewall rules

Once you've done all that, you don't want to repeat the setup every time you launch a server. So, you **create an AMI** of this configured instance.

Now, every time you launch an EC2 from this AMI, **everything is already set up** — OS, software, configurations — in minutes.

Advantages of AMI

Advantage	Explanation
🔁 **Reusability**	Quickly launch multiple identical instances using the same AMI.
🚀 **Fast Scaling**	Easily spin up pre-configured servers for auto-scaling during traffic spikes.
🛡️ **Backup & Recovery**	Take AMI snapshots as backups of your EC2 instance — useful before major changes.
📦 **Custom Configuration**	Tailor the image with your own app, tools, and security policies.
🌍 **Cross-region Deployment**	Copy AMIs across regions for global deployment.
👨‍👩‍👧 **Team Standardization**	Developers and testers use the exact same environment, reducing "it works on my machine" issues.

Example Use Case

Scenario: An e-commerce company sets up a web server with:

- NGINX + Node.js + MongoDB
- SSL certificate and monitoring tools

Instead of setting this up every time, they **create an AMI** called `Ecom-WebServer-2025`. When auto-scaling triggers (e.g., during a festive sale), new instances **launch instantly from this AMI**, serving users in seconds.

Interview Tip

> "I usually create custom AMIs after setting up base environments or before deploying major updates, so that I can roll back easily or replicate the setup across staging and production."

Mistakes/Lessons learnt:

1. Earlier, I used to create AMIs without proper tagging. Later, it became difficult to track which AMI belonged to which environment or release.

2. Initially, I didn't delete old unused AMIs and snapshots, which led to unnecessary storage costs. Now, I automate AMI cleanup with lifecycle policies.
3. Once I forgot to update the AMI after a critical patch, and newly launched instances were missing the fix. Since then, I've made **AMI refresh part** of our **deployment checklist**.

A sample AMI creation command

```
aws ec2 create-image \
  --instance-id i-0abc1234def5678gh \
  --name "MyApp-Image-April2025" \
  --description "An AMI for my app as of April 2025" \
  --no-reboot
```

How do you create a custom AMI and when would you use one?
Use `create-image` command or console; used for consistent deployments, backups, or launching multiple identical servers.

1. **What are the differences between public, private, and shared AMIs?**
   - **Public**: Available to all AWS users.
   - **Private**: Visible only to the owner.
   - **Shared**: Explicitly shared with specific AWS accounts.
2. **Can you create an AMI without stopping your instance?**
   Yes, using `--no-reboot` option, but the AMI might miss in-memory data or file system changes.

1. In-Memory Data

- **Application cache** (e.g., Redis, Memcached running on RAM)
- **Session data** stored in memory by application servers (like Node.js, Python, Java apps)
- **Running database transactions** in memory (e.g., data in a PostgreSQL, MySQL buffer pool not yet flushed to disk)
- **Log files** that are buffered in memory and not yet flushed to disk
- **Temporary processing data** held in RAM (e.g., image processing jobs, video transcoding data)

---

2. File System Changes (Not Flushed Yet)

- Files that were created or modified but still sitting in **OS disk cache** and not flushed to disk (especially if `fsync` wasn't called)

- **Database writes** that are still in the buffer/cache (example: uncommitted writes in MySQL's InnoDB buffer pool)
- **Active downloads/uploads** where parts of the file are not yet written to disk
- **Ongoing software installations/updates** that have not yet finalized disk writes

3. **What happens when you launch an EC2 instance from a custom AMI?**
   A new EC2 instance is created with the exact OS, configuration, and data as in the AMI.
4. **How do you ensure an AMI is shared securely between accounts?**
   Share it using `modify-image-attribute` and also share the snapshot; avoid making it public.
5. **How would you automate AMI creation and retention policies?**
   Use AWS Lambda with CloudWatch Events or AWS Systems Manager Automation with lifecycle rules.

Scenario	Why automate AMI creation + retention?
Nightly backup of production servers	So you always have a fresh backup AMI in case of crash
Before critical deployments (e.g., app updates)	So you can roll back if the update fails
For compliance and audits	Certain industries require regular snapshots and retention for X days

6. **What limitations or considerations apply when copying AMIs across regions?**
   You must copy the AMI manually; ensure associated snapshots are available in the destination region.

   An **AMI (Amazon Machine Image)** created in one AWS region **is specific to that region** — it is **not automatically available** in other regions.

   aws ec2 copy-image --source-region us-east-1 --source-image-id ami-1234567890abcdef0 --name "MyCopiedAMI" --region us-west-2

   Important Points:

- The snapshot linked to the AMI will also be copied automatically.
- Copying across regions may incur additional costs (for snapshot storage and transfer).
- Make sure that any associated permissions (e.g., if AMI is shared) are also handled after copying.

7. **How do AMIs work in Auto Scaling Groups or Launch Templates?**
   The AMI ID is referenced in the launch template/configuration to spin up identical instances automatically.
8. **Can you update an existing AMI? If not, what's the best practice?**
   No, AMIs are immutable. Best practice is to launch from the old AMI, make changes, and create a new AMI.

1. What does an Amazon Machine Image (AMI) include?

A. Only the operating system
B. The OS, application server, and application data
C. Only the instance type
D. Only the snapshot of a volume

✅ **Answer:** B. The OS, application server, and application data

---

2. Which AWS CLI command is used to create an AMI?

A. `aws ec2 start-image`
B. `aws ec2 build-image`
C. `aws ec2 create-image`
D. `aws ami create`

✅ **Answer:** C. `aws ec2 create-image`