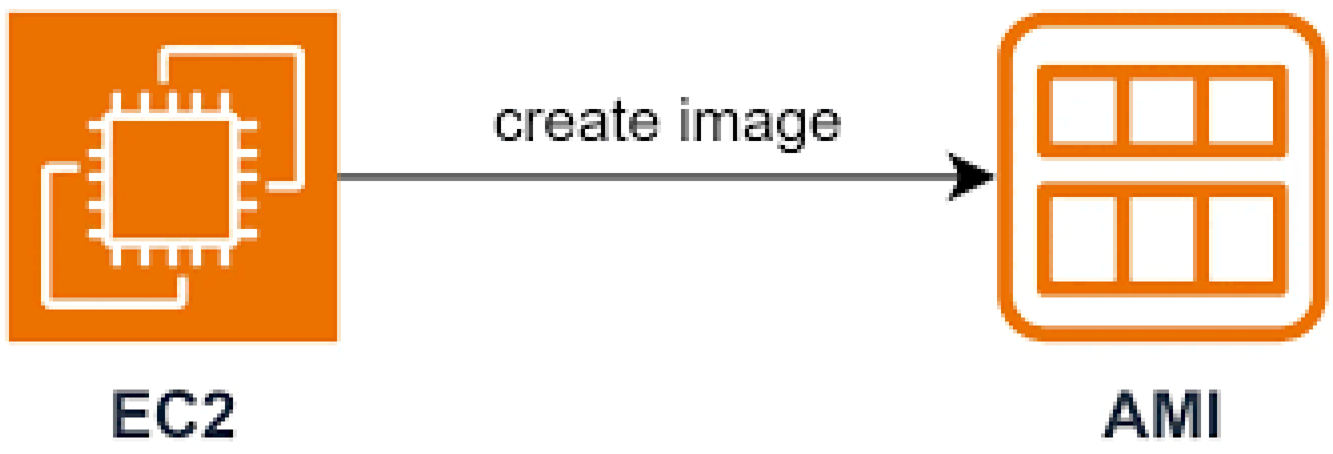
Amazon Machine Image (AMI)

An **Amazon Machine Image (AMI)** is a pre-configured virtual machine template used to launch instances in Amazon Web Services (AWS). It serves as a blueprint that defines the software stack, operating system, application servers, and applications required to run an instance.



Key Features of AMI

1. **Pre-configured Templates**
   - Contains an operating system (OS), application server, and applications.
   - Includes necessary configuration settings and system libraries.
2. **Customizable**
   - Users can create custom AMIs to include specific software, settings, and configurations tailored to their use case.
3. **Regional Availability**
   - AMIs are specific to an AWS region but can be copied to other regions.
4. **Sharing and Access Control**
   - AMIs can be private, shared with specific AWS accounts, or made public for broader use.
   - Permissions can be managed via AWS Identity and Access Management (IAM).
5. **Types of AMIs**
   - Public AMIs: Provided by AWS or other users for general use.
   - Private AMIs: Created and used within an AWS account.
   - Marketplace AMIs: Pre-built images provided by vendors, often with licensed software.

Components of an AMI

1. **Root Volume Template**
   - Specifies the operating system, application server, and applications.

2. **Launch Permissions**
    - Controls who can use the AMI to launch instances.
3. **Block Device Mapping**
    - Defines the storage volumes (EBS or instance store) attached to the instance when launched.

Creating an AMI

An AMI can be created from an existing EC2 instance or a virtual machine. Below are the steps:

1. Prepare the Instance

- Update the operating system and applications to their desired state.
- Remove sensitive data or temporary files.
- Ensure the instance is in a stopped state.

2. Create an Image

- **AWS Console:**
    1. Go to the **Instances** page in the EC2 dashboard.
    2. Select the instance and choose **Actions** > **Image and templates** > **Create Image**.
    3. Provide a name, description, and instance settings (e.g., volume size).
    4. Click **Create Image**.
- **AWS CLI:**

```
aws ec2 create-image --instance-id i-xxxxxxxxxxxx --name "MyCustomAMI" –
description "Custom AMI with pre-installed software"
```

3. Monitor AMI Creation

- View the progress in the **AMI** section of the EC2 dashboard.
- Once complete, the AMI is available for launching instances.

Using an AMI

1. **Launch an Instance**
    - **AWS Console:**
        1. Go to the **Instances** page and click **Launch Instance**.
        2. Select an AMI (public, private, or from the AWS Marketplace).
        3. Configure instance type, storage, and network settings.
        4. Launch the instance.
    - **AWS CLI:**

```
aws ec2 run-instances --image-id ami-xxxxxxxxxxxx --count 1 --instance-type t2.micro
```

2.Copy AMI to Another Region

- Use this feature for disaster recovery or cross-region replication.

- **AWS Console:**
  1. Navigate to the **AMI** section.
  2. Select the AMI and choose **Actions** > **Copy AMI**.
  3. Specify the destination region and settings.
- **AWS CLI:**

aws ec2 copy-image --source-region us-east-1 --source-image-id ami-xxxxxxxxxxxx --region us-west-2 --name "CopiedAMI"

3.Share an AMI

- Grant other AWS accounts access to your AMI.
- **AWS Console:**
  1. Select the AMI and go to **Actions** > **Modify Image Permissions**.
  2. Add the AWS account ID of the recipient.
- **AWS CLI:**
- aws ec2 modify-image-attribute --image-id ami-xxxxxxxxxxxx --attribute launchPermission --operation-type add --user-ids 123456789012

AMI Lifecycle Management

1. **Automated AMI Creation**
   - Use **AWS Systems Manager** or **Amazon Data Lifecycle Manager (DLM)** to automate periodic creation of AMIs.
2. **Versioning**
   - Maintain version control for AMIs to track changes and ensure rollback options.
3. **Decommissioning**
   - Deregister outdated AMIs to avoid confusion and manage costs:
     - **AWS CLI:**
     - aws ec2 deregister-image --image-id ami-xxxxxxxxxxxx

Types of Storage for AMIs

1. **EBS-backed AMIs**
   - The root volume is stored on EBS.
   - Supports stopping, restarting, and resizing the instance.
2. **Instance Store-backed AMIs**
   - The root volume is stored on ephemeral instance storage.
   - Data is lost when the instance stops or terminates.

Best Practices for Working with AMIs

1. **Security**
   - Ensure the AMI does not contain sensitive information like private keys or credentials.
   - Use encryption for sensitive data and secure AMI sharing permissions.
2. **Standardization**
   - Create base AMIs with consistent configurations for repeatable deployments.
3. **Optimization**

    - Regularly update AMIs with the latest patches and application updates.
4. **Documentation**
    - Clearly document AMI versions, configurations, and purposes.
5. **Cost Management**
    - Delete unused or obsolete AMIs and snapshots to avoid unnecessary costs.

Use Cases for AMIs

1. **Application Deployment**
    - Launch pre-configured instances to host web applications, databases, or analytics tools.
2. **Scaling and Automation**
    - Use AMIs to create auto-scaling groups for handling variable workloads.
3. **Disaster Recovery**
    - Maintain AMIs as backups for quickly restoring services after failures.
4. **DevOps and CI/CD**
    - Use custom AMIs in pipelines to provide consistent development, testing, and production environments.
5. **Multi-region and Cross-account Deployments**
    - Copy or share AMIs for global or collaborative projects.

Monitoring AMIs

1. **AWS CloudWatch**
    - Monitor usage patterns of AMIs, such as instance launches.
2. **AWS CloudTrail**
    - Track API calls related to AMIs for auditing and compliance.
3. **IAM Policies**
    - Review and enforce strict permissions to control access to sensitive AMIs.

-