continuous integration and continuous deployment (CI/CD) pipeline using AWS

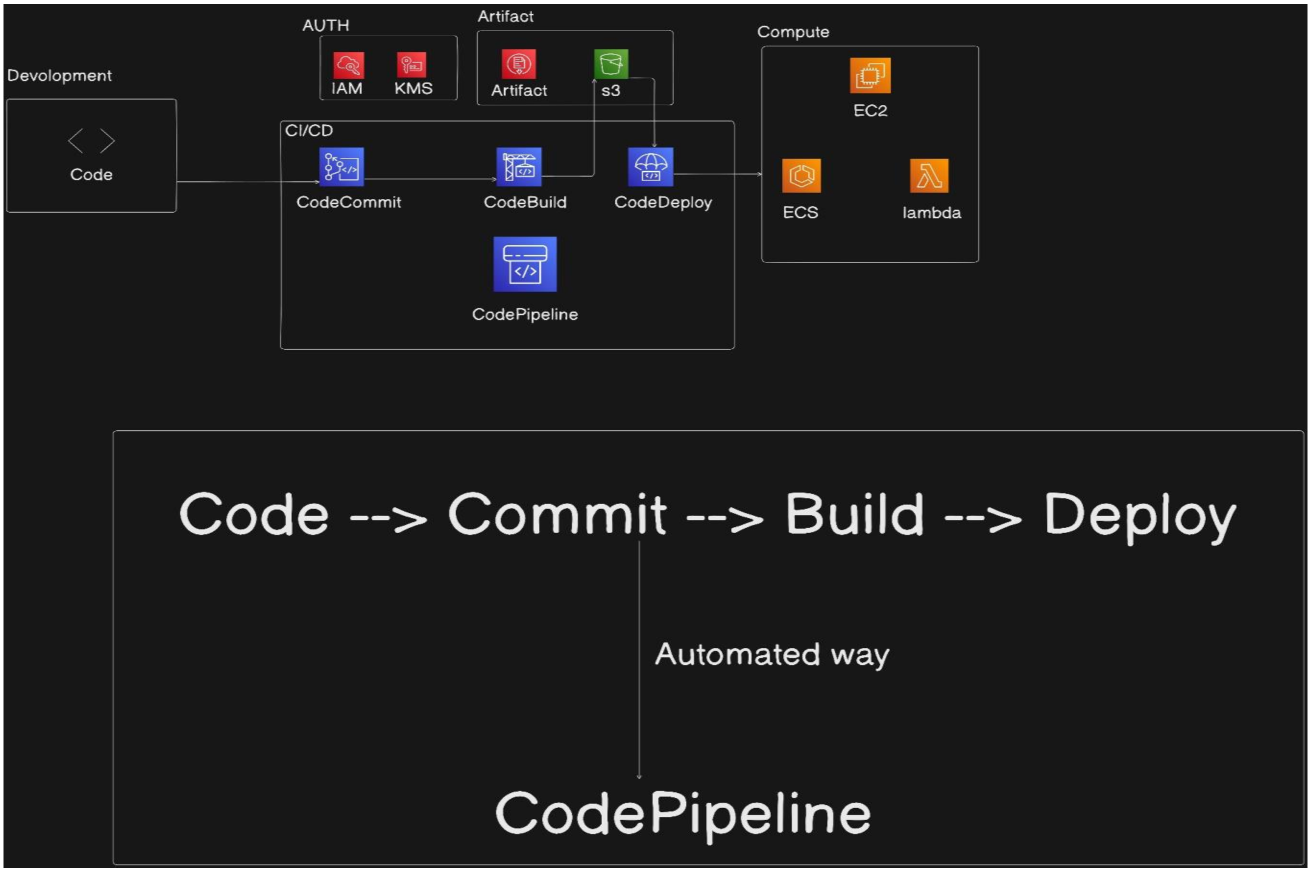
services –> CodeCommit, CodeBuild, CodeDeploy, and CodePipeline, with the mentioned authentication mechanisms and artifact storage options.

Step 1: Set up IAM Roles and Policies

1. **IAM Roles**:
   - Create IAM roles for services to access each other. These roles will be used for authentication.
   - For example, you'll need roles for CodeBuild, CodeDeploy, and CodePipeline. Ensure these roles have the necessary permissions.
2. **KMS** (Key Management Service):
   - Set up KMS if you need encryption for sensitive data.
   - Create and configure KMS keys as required. Storage
3. **S3 Bucket**:
   - Create an S3 bucket to store artifacts generated during the build process.
   - Configure the bucket policy to restrict access if needed.
   - Optionally, you can enable versioning and encryption using KMS. Configure
4. **Create a CodeCommit Repository**:
   - Go to CodeCommit in the AWS Management Console and create a new repository.
5. Clone the repository locally or push an existing repository.
6. **Create a Build Project**:
   - Go to CodeBuild in the AWS Management Console and create a new build project.
   - Configure source provider as CodeCommit.
   - Choose the appropriate runtime environment and build specifications.
7. **Create an Application**:
   - Go to CodeDeploy in the AWS Management Console and create a new application.
8. Set up deployment groups based on your requirements.
9. **Create a Pipeline**:
   - Go to CodePipeline in the AWS Management Console and create a new pipeline.
   - Configure your pipeline stages:
     - Source: CodeCommit
     - Build: CodeBuild
10. Deploy: CodeDeploy
11. **Pipeline Configuration**:

   - Connect each stage of the pipeline and configure actions.
   - Define triggers for pipeline execution (e.g., based on code commits).
   - Specify IAM roles for each stage to allow access to required services.

12. **Test Pipeline**:
    - Trigger the pipeline manually or through a code commit to test its functionality.
    - Monitor each stage for errors and troubleshoot if necessary.
13. **Deploy**:
    - Once tested, the pipeline will automatically deploy changes to your target environment using CodeDeploy.
14. **Compute Options**:
    - Based on your application requirements, you can configure CodeBuild to use EC2, ECS, or Lambda as the build environment.
15. **IAM and KMS Integration**:
    - Ensure IAM roles and policies are correctly configured for authentication.
    - Integrate KMS for encryption if needed.


AUTH
IAM
KMS
Artifact
Artifact
s3
Compute
EC2
Devolopment
Code
CI/CD
CodeCommit
CodeBuild
CodeDeploy
ECS
lambda
CodePipeline
Code --> Commit --> Build --> Deploy
Automated way
CodePipeline