4.2. Student Handout

AWS CI/CD Tools: Student Handout

Introduction to AWS CI/CD Tools

Welcome to Day 4! Today, we will explore AWS services that automate the Continuous Integration (CI) and Continuous Delivery (CD) process. By the end of this session, you'll understand how AWS services streamline the software development lifecycle.

What is CI/CD?

- **Continuous Integration (CI)**: Frequently integrating code changes into a shared repository, verified by automated builds and tests.
- **Continuous Delivery (CD)**: Automatically deploying code changes to production after passing CI, ensuring the application is always deployable.

Overview of the AWS DevOps Pipeline for CI/CD

AWS provides services that automate the CI/CD process, creating a pipeline from development to production with minimal manual intervention:

1. **AWS CodeCommit**: Source control service for storing and versioning code.
2. **AWS CodeBuild**: Build service that compiles code, runs tests, and packages it for deployment.
3. **AWS CodeDeploy**: Deployment service that automates application deployment to various environments.
4. **AWS CodePipeline**: Orchestration service that automates the CI/CD process by connecting CodeCommit, CodeBuild, and CodeDeploy.

AWS Code Services for Automating CI/CD Processes

1. AWS CodeCommit: Storing and Versioning Code

- **Description**: A fully managed source control service for securely storing code in the cloud.
- **Key Features**:
- Supports Git commands.
- Secure and scalable.
- Integrated with AWS services like CodeBuild and CodePipeline.

Examples:

- Store and manage a JavaScript application repository.
- Track changes in a Python project.
- Collaborate on a team-based Java codebase.

2. AWS CodeBuild: Automating the Build Process

- **Description**: A fully managed build service that compiles source code, runs tests, and packages it for deployment.
- **Key Features**:
- Supports multiple programming languages (Java, Python, Node.js, etc.).
- Scalable resource provisioning.
- Integrated with CodeCommit, CodePipeline, and other AWS services.

Examples:

- Compile a Java application and run unit tests.
- Build a Docker image for a Node.js application.
- Execute integration tests for a Python project.

3. AWS CodeDeploy: Automating the Deployment Process

- **Description**: Automates the deployment of applications to various environments, such as EC2 instances, Lambda functions, or on-premises servers.
- **Key Features**:
- Supports multiple deployment targets (EC2, Lambda, ECS, etc.).
- Performs rolling updates and blue/green deployments.
- Integrated with CodePipeline for end-to-end automation.

Examples:

- Deploy a web application to an EC2 instance.
- Update a Lambda function with new code.
- Roll out a new version of a containerized application on ECS.

4. AWS CodePipeline: Orchestrating the CI/CD Process

- **Description**: A fully managed service that automates the CI/CD process by connecting CodeCommit, CodeBuild, and CodeDeploy.
- **Key Features**:

- Fully automated CI/CD pipeline.
- Integration with third-party tools like GitHub and Jenkins.
- Real-time monitoring and notifications.

Examples:

- Create a pipeline for a microservices architecture.
- Integrate with GitHub for source code management.
- Automate the deployment of a serverless application.

Integration with Other AWS Services for a Full CI/CD Pipeline

1. Using AWS Lambda and Amazon S3 in a Serverless CI/CD Pipeline

- **Description**: AWS Lambda runs code without managing servers, and Amazon S3 stores static assets.
- **Example**: Use CodePipeline to trigger a Lambda function that processes data stored in an S3 bucket.

2. Amazon ECS and EKS for Containerized Application Delivery

- **Description**: Amazon ECS and EKS deploy and manage containers at scale.
- **Example**: Use CodePipeline to deploy a Docker container to ECS or EKS.

Hands-On: Setting Up a Simple CI/CD Process Using AWS Code Services

Step-by-Step Guide

1. **Create a CodeCommit Repository**:

- Go to the AWS Management Console and navigate to CodeCommit.
- Create a new repository and push your code to it.

2. **Set Up CodeBuild**:

- Navigate to CodeBuild and create a new build project.
- Configure the build settings (e.g., source code location, build commands).

3. **Set Up CodeDeploy**:

- Navigate to CodeDeploy and create a new deployment group.
- Configure the deployment settings (e.g., target environment, deployment strategy).

4. **Set Up CodePipeline**:

- Navigate to CodePipeline and create a new pipeline.
- Add stages for source (CodeCommit), build (CodeBuild), and deploy (CodeDeploy).

5. **Test the Pipeline**:

- Make a change to your code and push it to CodeCommit.
- Watch as CodePipeline automatically triggers the build and deployment process.

Conclusion

AWS services like CodeCommit, CodeBuild, CodeDeploy, and CodePipeline automate the CI/CD process, allowing developers to focus on writing code rather than managing infrastructure. By using these services, you can create a fully automated pipeline that takes your code from development to production efficiently.

Diagram: AWS CI/CD Pipeline

```
+---------------+   +----------------+   +-------------+   +---------------+
| CodeCommit    |-->| CodeBuild      |-->| CodeDeploy |-->| Production    |
| (Source Code)|   | (Build & Test)|   | (Deploy)    |   | (Environment)|
+---------------+   +----------------+   +-------------+   +---------------+
```

Potential Gaps or Unclear Points

1. **Understanding the Role of Each Service**: Ensure clarity on the distinct roles of CodeCommit, CodeBuild, CodeDeploy, and CodePipeline.
2. **Integration with Other AWS Services**: Clarify that services like Lambda, S3, ECS, and EKS are optional and can be added as needed.

Final Thoughts

AWS provides powerful tools to automate the CI/CD process, enhancing automation and efficiency in software development. By leveraging AWS services, you can streamline your development workflow and focus on delivering high-quality applications.