5.2. Student Handout

AWS CLI for Templated Files: Student Handout

Introduction to AWS CLI and Templated Files

- **AWS CLI**: A tool to interact with AWS services via command line.
- **Templated Files**: Define infrastructure in YAML or JSON for AWS resources.
- **Key Services**:
- **AWS CloudFormation**: Manages AWS infrastructure.
- **AWS SAM**: Focuses on serverless applications.

CloudFormation and SAM Templates

- **CloudFormation Templates**: Define AWS resources in YAML/JSON.
- **SAM Templates**: Extend CloudFormation for serverless resources.

Examples:

1. **CloudFormation**: Define an EC2 instance.

```
Resources:

MyEC2Instance:

Type: "AWS::EC2::Instance"

Properties:

InstanceType: "t2.micro"

ImageId: "ami-0abcdef1234567890"
```

2. **SAM**: Define a Lambda function.

```
Resources:

MyLambdaFunction:

Type: "AWS::Serverless::Function"

Properties:

Handler: index.handler

Runtime: nodejs14.x

CodeUri: ./src
```

3. **CloudFormation**: Create an S3 bucket.

```
Resources:

MyS3Bucket:

Type: "AWS::S3::Bucket"

Properties:

BucketName: "my-unique-bucket-name"
```

Use Cases and Benefits

- **Infrastructure as Code (IaC)**: Version control infrastructure.
- **Automation**: Automate resource management.
- **Reusability**: Use templates across environments.
- **Rollback and Recovery**: Automatic rollback on errors.

Creating and Managing CloudFormation Stacks with AWS CLI

- **CloudFormation Stack**: Collection of AWS resources managed as a unit.

Steps:

1. **Prepare the Template**: Write in YAML/JSON.
2. **Upload the Template**: Use S3 or local file.
3. **Create Stack**:

```
aws cloudformation create-stack --stack-name MyStack --template-body
file://my-template.yaml
```

4. **Monitor Stack**:

```
aws cloudformation describe-stacks --stack-name MyStack
```

5. **Update/Delete Stack**:

```
aws cloudformation update-stack --stack-name MyStack --template-body
file://updated-template.yaml
```

```
aws cloudformation delete-stack --stack-name MyStack
```

Examples:

1. **Create Stack**: Deploy an EC2 instance.
2. **Update Stack**: Modify S3 bucket properties.
3. **Delete Stack**: Remove all resources in a stack.

Writing CloudFormation Templates

- **Sections**:

- **Resources**: Define AWS resources.
- **Parameters**: Pass dynamic values.
- **Outputs**: Return values after stack creation.

Example:

```
Resources:

MyS3Bucket:

Type: "AWS::S3::Bucket"

Properties:

BucketName: "my-unique-bucket-name"
```

Using AWS CLI for CloudFormation Stacks

- **Deploy**:

```
aws cloudformation create-stack --stack-name MyStack --template-body
file://my-template.yaml
```

- **Update**:

```
aws cloudformation update-stack --stack-name MyStack --template-body
file://updated-template.yaml
```

- **Delete**:

```
aws cloudformation delete-stack --stack-name MyStack
```

Examples:

1. **Deploy Stack**: Create a VPC.
2. **Update Stack**: Add a security group.
3. **Delete Stack**: Remove a Lambda function.

Working with SAM Templates

- **SAM Templates**: Simplify serverless resource definitions.

Example:

```
Resources:

MyLambdaFunction:

Type: "AWS::Serverless::Function"

Properties:

Handler: index.handler

Runtime: nodejs14.x

CodeUri: ./src
```

Deploying SAM Templates

- **AWS CLI**: Use `create-stack` or `update-stack`.
- **SAM CLI**:

1. **Build**:

```
sam build
```

2. **Deploy**:

```
sam deploy --guided
```

Examples:

1. **Build SAM**: Package a Lambda function.
2. **Deploy SAM**: Deploy an API Gateway.
3. **Update SAM**: Modify a DynamoDB table.

Managing Stack Parameters and Outputs

- **Parameters**: Pass values using `--parameters`.

```
aws cloudformation create-stack --stack-name MyStack --template-body
file://my-template.yaml --parameters
ParameterKey=BucketName,ParameterValue=my-bucket
```

- **Outputs**: Retrieve using `describe-stacks`.

```
aws cloudformation describe-stacks --stack-name MyStack
```

Examples:

1. **Pass Parameter**: Set EC2 instance type.
2. **Retrieve Output**: Get S3 bucket URL.
3. **Modify Parameter**: Change Lambda memory size.

Hands-on: Deploying a Full Stack

1. **Write Template**: Define S3 bucket and Lambda function.

```
Resources:

MyS3Bucket:

Type: "AWS::S3::Bucket"

Properties:

BucketName: "my-unique-bucket-name"



MyLambdaFunction:

Type: "AWS::Lambda::Function"

Properties:

Handler: index.handler

Runtime: nodejs14.x

Code:

ZipFile: |

exports.handler = async (event) => {

console.log("Hello from Lambda!");

};
```

2. **Deploy Stack**:

```
aws cloudformation create-stack --stack-name MyFullStack --template-body
file://my-full-stack.yaml
```

3. **Verify Stack**:

```
aws cloudformation describe-stacks --stack-name MyFullStack
```

4. **Test Lambda**:

```
aws lambda invoke --function-name MyLambdaFunction output.txt
```

Examples:

1. **Deploy Full Stack**: Create a web application infrastructure.
2. **Verify Resources**: Check EC2 instance status.
3. **Test Functionality**: Invoke a Lambda function.

Conclusion

- AWS CLI and templated files enable efficient cloud infrastructure management.
- Use CloudFormation and SAM for defining and automating AWS resources.
- Practice deploying, updating, and managing stacks for hands-on experience.