### 

[What is Cloudformation?](#_df4nypqjzwq3)

[How can you upload Cloudformation files?](#_rl3ncxeqgf1o)

[What is a Cloudformation stack?](#_254s26nt8glb)

[Steps to upload Cloudformation via YML file:](#_p8t52bw90ew9)

[Creating EC2 via Cloudformation:](#_ik17sddy7yhw)

[Creating EC2 auto-scaling group via Cloudformation:](#_9ld7wrlda79h)

[Creating S3 via Cloudformation:](#_nj7wh7igdqjg)

[Creating Lambda via Cloudformation:](#_mmb19blglpt9)

### 

### What is Cloudformation?

AWS CloudFormation is a service that allows you to define and manage your AWS infrastructure as code. CloudFormation uses templates to describe and provision resources in a predictable and repeatable way, helping you automate the creation, deployment, and management of your AWS resources.

With CloudFormation, you can:

* Declare all the AWS resources needed to run your application in a single, human-readable template.
* Provision and manage infrastructure resources, such as Amazon EC2 instances, Amazon S3 buckets, and Amazon RDS database instances, using templates and declarative syntax.
* Control and manage the dependencies between resources to ensure they are created in the correct order.
* Automatically roll back changes and undo any updates or deletions that fail to complete successfully.
* Manage multiple versions of your infrastructure and track changes over time.
* Reuse templates across your organization, making it easy to replicate environments across different accounts and regions.
* Integrate with other AWS services, such as AWS CloudTrail and AWS Config, to monitor and audit changes to your infrastructure.
* Overall, CloudFormation helps you manage your infrastructure resources more efficiently and reduces the likelihood of errors and inconsistencies in your AWS environment.

Let’s dive into some sample hands-on that you can follow. Do take note of the fields that you need to edit before you upload.

### How can you upload Cloudformation files?

You can upload a CloudFormation file to AWS using the AWS Management Console, AWS CLI, or AWS SDKs. Here are the steps to upload a CloudFormation file using the AWS Management Console:

* Go to the AWS Management Console and navigate to the CloudFormation service.
* Click on the "Create Stack" button to start creating a new stack.
* In the "Create stack" page, select the option "Upload a template file" and click on the "Choose file" button to select the CloudFormation file from your local system.
* Once you have selected the file, click on the "Next" button to proceed to the "Specify stack details" page.
* On the "Specify stack details" page, provide a stack name and any other required parameters for your CloudFormation stack.
* Click on the "Next" button to proceed to the "Configure stack options" page.
* On the "Configure stack options" page, you can optionally specify additional options for your stack, such as tags and permissions.
* Click on the "Next" button to review your stack settings on the "Review" page.
* Review the settings and click on the "Create stack" button to start creating your CloudFormation stack.
* Once the stack creation process is complete, AWS CloudFormation will provision the resources defined in your CloudFormation template. You can monitor the status of the stack creation process in the AWS CloudFormation console.

### What is a Cloudformation stack?

A CloudFormation stack is a collection of AWS resources that are created, updated, and deleted as a single unit. A stack is defined by a CloudFormation template, which describes the desired state of the resources that should be provisioned within the stack.

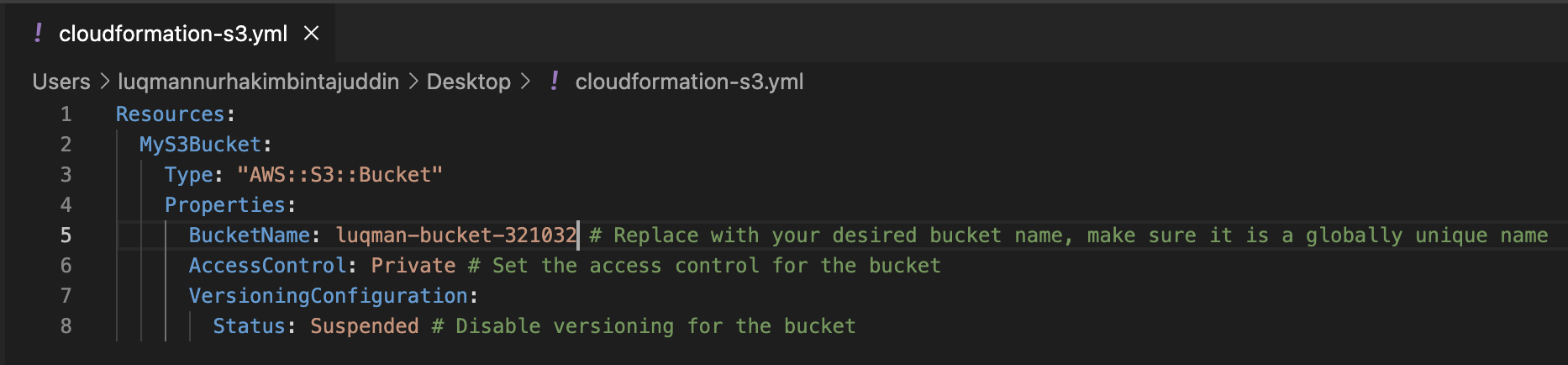
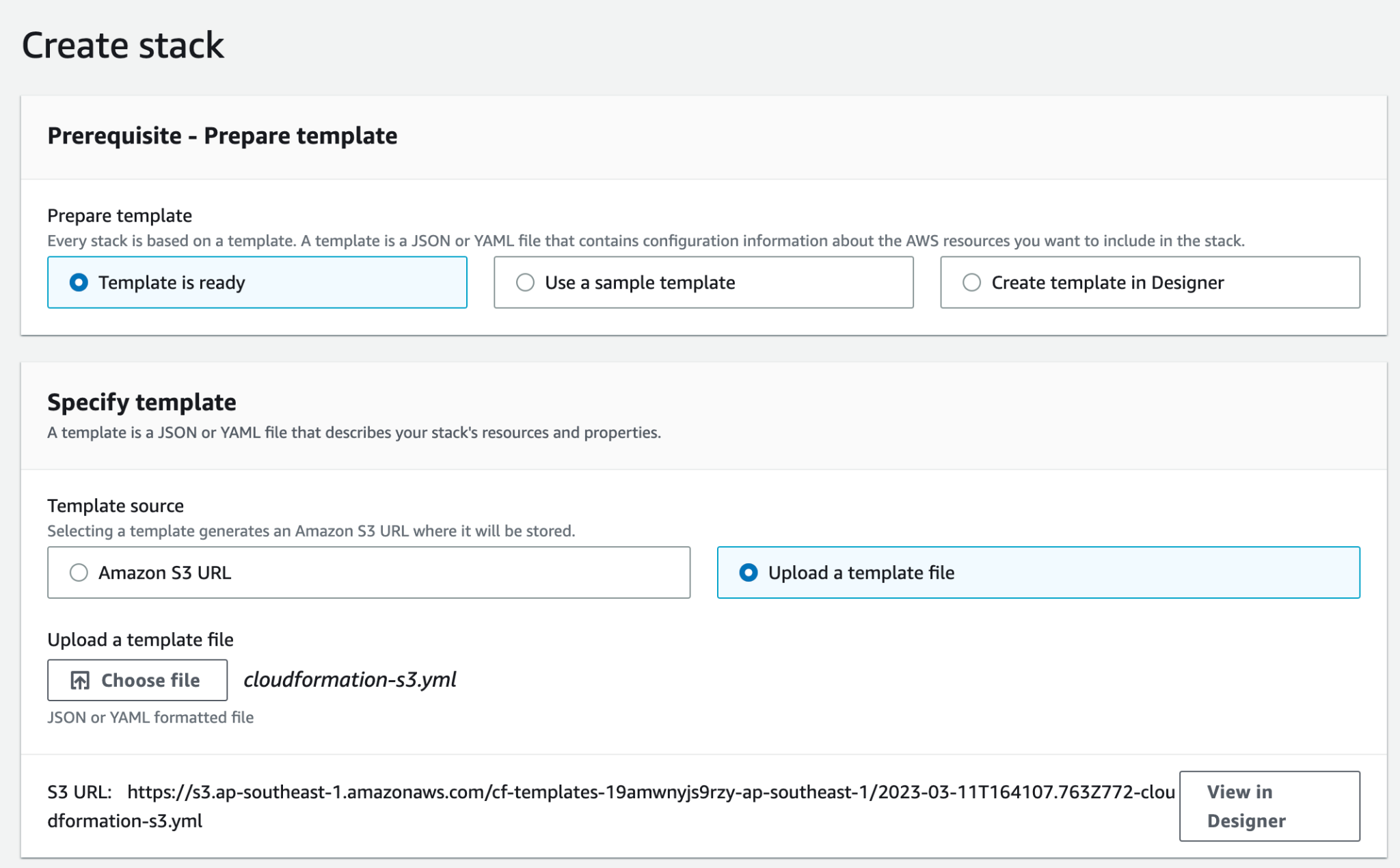
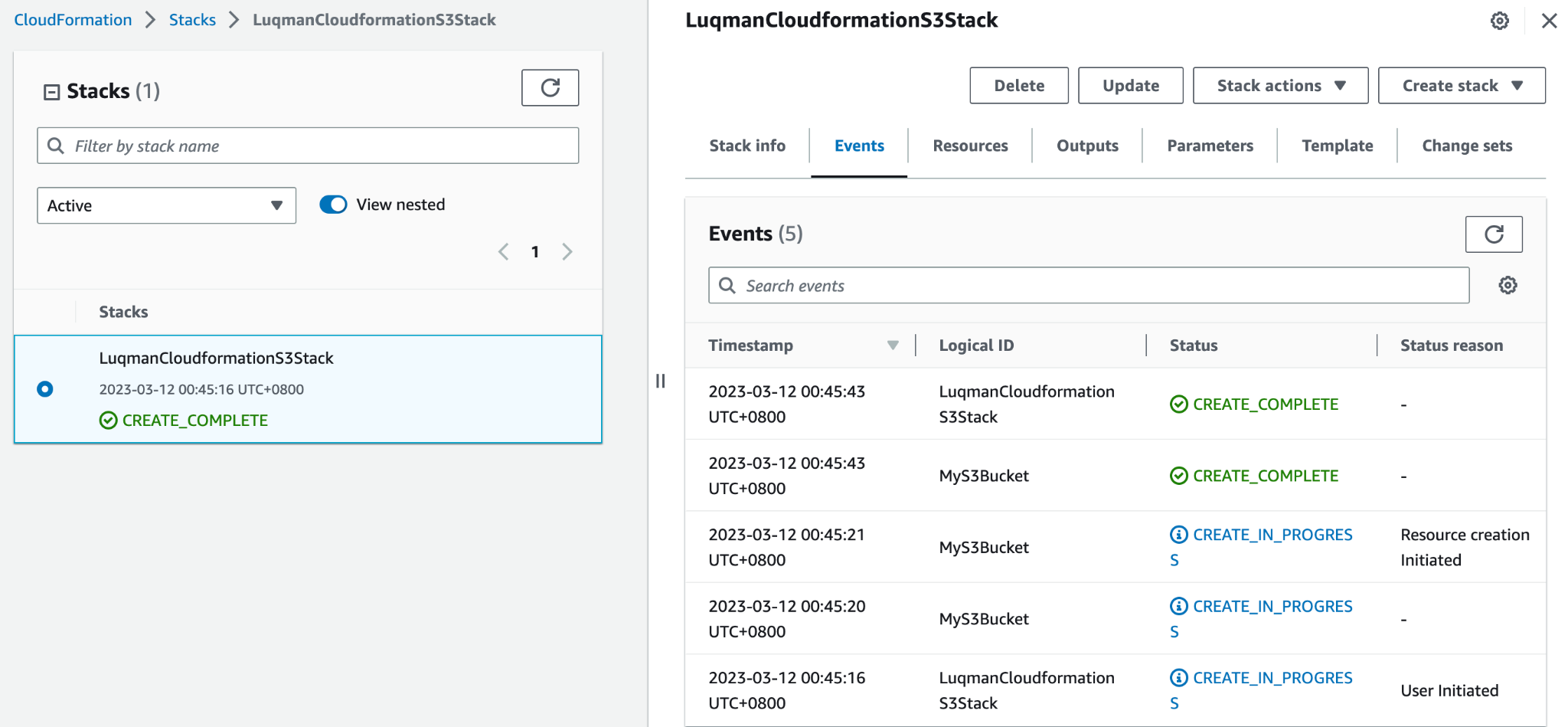
When you create a CloudFormation stack, AWS CloudFormation provisions the resources defined in the template in the order specified in the template. This helps to ensure that resources are created in the correct order and that dependencies between resources are managed correctly.

Stacks can include a wide range of AWS resources, such as EC2 instances, Amazon S3 buckets, Amazon RDS databases, and Elastic Load Balancers. Once a stack has been created, you can update it to modify its resources or add new ones. You can also delete a stack to remove all the resources that were created within it.

CloudFormation stacks provide several benefits, including:

* Automation: Stacks allow you to automate the creation and management of your AWS resources using a declarative template-based approach.
* Consistency: Stacks help to ensure that your AWS resources are consistently configured and deployed across different environments.
* Rollback: Stacks provide an automatic rollback feature that can be used to undo changes if any resources fail to be created or updated correctly.
* Visibility: Stacks provide visibility into the resources that have been provisioned within them, making it easier to track changes and manage updates.
* Overall, CloudFormation stacks make it easier to manage complex AWS environments by providing a unified way to provision, manage, and update resources.

### Steps to upload Cloudformation via YML file:

1. Ensure that you have saved your cloudformation file using a text editor like Visual Studio Code with a meaningful name e.g. cloudformation-create-s3.yml
2. 
3. On the AWS console, search for Cloudformation and click Create stack.
4. Choose “Template is ready” as the option.
5. Under Specify Template, select “Upload a template file”
6. Select the yml file that you have saved earlier and upload this file on the console.
7. 
8. Click next
9. Give the stack a meaningful name e.g. LuqmanCloudformationS3Stack and click next.
10. Under Stack failure options, choose “Roll back all stack resources”. This ensures that when failures occur, you undo all the changes.
11. Click next and submit to create the stack.
12. You can see the status of your Cloudformation upload. Here, any error messages will be printed.
13. 
14. You can validate all the resources you intended to create by going to the different resources from the console.
15. Once you’re done, we will delete all the resources created. To do this, click “Delete” on the stack and wait for all the resources to be deleted.

### Creating EC2 via Cloudformation:

—------------------------------------------------------------

| Resources:  EC2Instance:  Type: 'AWS::EC2::Instance'  Properties:  ImageId: ami-0c94855ba95c71c99 # Replace with the AMI ID for your desired EC2 instance image  InstanceType: t2.micro # Replace with the instance type you want to use  KeyName: my-key-pair # Replace with the name of your EC2 key pair  SubnetId: !Ref DefaultSubnet # Reference the default subnet in the VPC  SecurityGroupIds:  - !Ref DefaultSecurityGroup # Reference the default security group in the VPC   DefaultSecurityGroup:  Type: 'AWS::EC2::SecurityGroup'  Properties:  GroupDescription: Default security group for the default VPC  VpcId: !Ref DefaultVPC  SecurityGroupIngress:  - IpProtocol: tcp  FromPort: 22  ToPort: 22  CidrIp: 0.0.0.0/0 # Allow SSH access from any IP address   DefaultSubnet:  Type: 'AWS::EC2::Subnet'  Properties:  VpcId: !Ref DefaultVPC  CidrBlock: 10.0.1.0/24 # Replace with the CIDR block of your desired subnet   DefaultVPC:  Type: 'AWS::EC2::VPC'  Properties:  CidrBlock: 10.0.0.0/16 # Replace with the CIDR block of your desired VPC  Tags:  - Key: Name  Value: Default VPC |
| --- |

—------------------------------------------------------------

In this template, we create four resources:

* An EC2 instance (EC2Instance) with the specified image ID, instance type, key pair name, and the default VPC's subnet ID and security group ID.
* A default security group (DefaultSecurityGroup) with a rule allowing SSH access from any IP address.
* A default subnet (DefaultSubnet) within the default VPC with the specified CIDR block.
* A default VPC (DefaultVPC) with the specified CIDR block and name tag.
* You can launch this template using the AWS CloudFormation console, AWS CLI, or any other supported deployment tool. Make sure to replace the placeholders with appropriate values specific to your use case, such as the AMI ID, instance type, key pair name, and CIDR blocks.

### Creating EC2 auto-scaling group via Cloudformation:

—------------------------------------------------------------

| Resources:  LaunchConfig:  Type: "AWS::AutoScaling::LaunchConfiguration"  Properties:  ImageId: ami-0c94855ba95c71c99 # Replace with the AMI ID for your desired EC2 instance image  InstanceType: t2.micro # Replace with the instance type you want to use  KeyName: my-key-pair # Replace with the name of your EC2 key pair  SecurityGroups:  - !Ref DefaultSecurityGroup # Reference the default security group in the VPC  UserData:  Fn::Base64: !Sub |  #!/bin/bash  echo "Hello, World!" > /var/www/html/index.html  AssociatePublicIpAddress: true  IamInstanceProfile: !Ref InstanceProfile   DefaultSecurityGroup:  Type: "AWS::EC2::SecurityGroup"  Properties:  GroupDescription: Default security group for the default VPC  VpcId: !Ref DefaultVPC  SecurityGroupIngress:  - IpProtocol: tcp  FromPort: 22  ToPort: 22  CidrIp: 0.0.0.0/0 # Allow SSH access from any IP address   DefaultVPC:  Type: "AWS::EC2::VPC"  Properties:  CidrBlock: 10.0.0.0/16 # Replace with the CIDR block of your desired VPC  Tags:  - Key: Name  Value: Default VPC   InstanceProfile:  Type: "AWS::IAM::InstanceProfile"  Properties:  Path: "/"  Roles:  - !Ref InstanceRole   InstanceRole:  Type: "AWS::IAM::Role"  Properties:  AssumeRolePolicyDocument:  Version: "2012-10-17"  Statement:  - Effect: "Allow"  Principal:  Service:  - "ec2.amazonaws.com"  Action:  - "sts:AssumeRole"  Path: "/"  Policies:  - PolicyName: "EC2DescribeInstancesPolicy"  PolicyDocument:  Version: "2012-10-17"  Statement:  - Effect: "Allow"  Action:  - "ec2:DescribeInstances"  Resource: "\*"   AutoScalingGroup:  Type: "AWS::AutoScaling::AutoScalingGroup"  Properties:  AvailabilityZones:  - !Select [0, !GetAZs ""] # Use the first Availability Zone in the region  LaunchConfigurationName: !Ref LaunchConfig  MinSize: 1 # Minimum number of instances in the group  MaxSize: 3 # Maximum number of instances in the group  DesiredCapacity: 1 # Desired number of instances in the group  VPCZoneIdentifier:  - !Ref DefaultSubnet # Reference the default subnet in the VPC |
| --- |

—------------------------------------------------------------

In this template, we create several resources:

* An Auto Scaling Group (AutoScalingGroup) with a minimum, maximum, and desired capacity of 1, 3, and 1 respectively. This group uses the specified launch configuration and is placed in the default subnet of the default VPC.
* A launch configuration (LaunchConfig) that specifies the image ID, instance type, key pair name, security group, user data script, and other settings for the EC2 instances in the Auto Scaling Group.
* A default security group (DefaultSecurityGroup) with a rule allowing SSH access from any IP address.

### Creating S3 via Cloudformation:

—------------------------------------------------------------

| Resources:  MyS3Bucket:  Type: "AWS::S3::Bucket"  Properties:  BucketName: my-bucket-name # Replace with your desired bucket name, make sure it is a globally unique name  AccessControl: Private # Set the access control for the bucket  VersioningConfiguration:  Status: Suspended # Disable versioning for the bucket |
| --- |

—------------------------------------------------------------

This template creates an S3 bucket named my-bucket-name with private access control and versioning disabled. You can customize the bucket name, access control settings, and other properties by modifying the Properties section of the resource.

### Creating Lambda via Cloudformation:

—------------------------------------------------------------

| Resources:  MyLambdaFunction:  Type: "AWS::Lambda::Function"  Properties:  Code:  ZipFile: |  def lambda\_handler(event, context):  print("Hello, World!")  return "Hello, World!"  Handler: index.lambda\_handler  Role: !GetAtt MyLambdaRole.Arn  Runtime: python3.7  Timeout: 3   MyLambdaRole:  Type: "AWS::IAM::Role"  Properties:  AssumeRolePolicyDocument:  Version: "2012-10-17"  Statement:  - Effect: "Allow"  Principal:  Service: "lambda.amazonaws.com"  Action: "sts:AssumeRole" |
| --- |

—------------------------------------------------------------

This CloudFormation template creates a Lambda function that uses Python 3.7 runtime and prints "Hello, World!" to the console. You can modify the code in the ZipFile section to customize the Lambda function's behavior.

—------------------------------------------------------------

—------------------------------------------------------------