### 

[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.

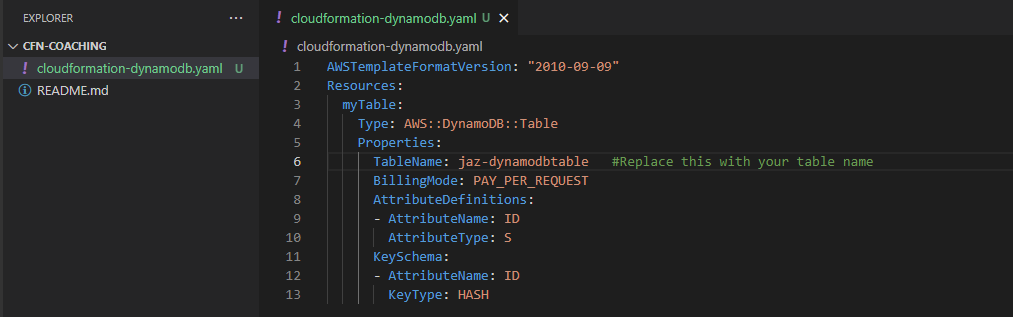
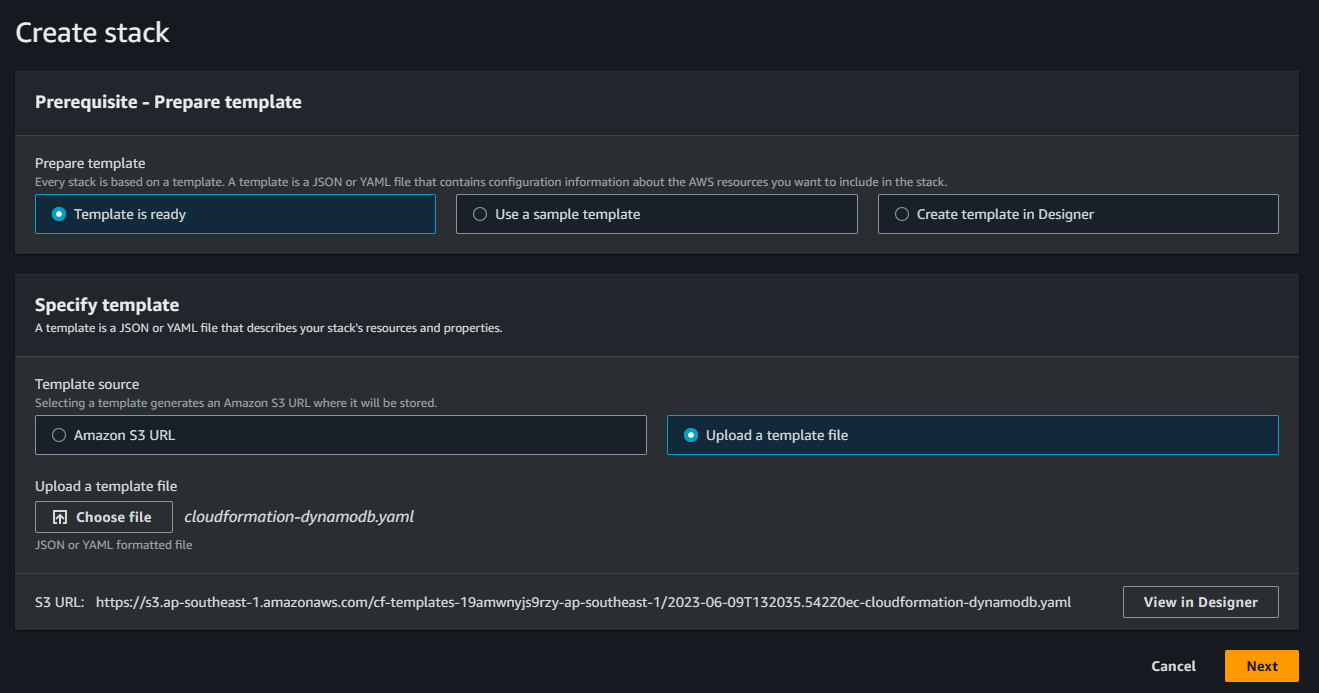
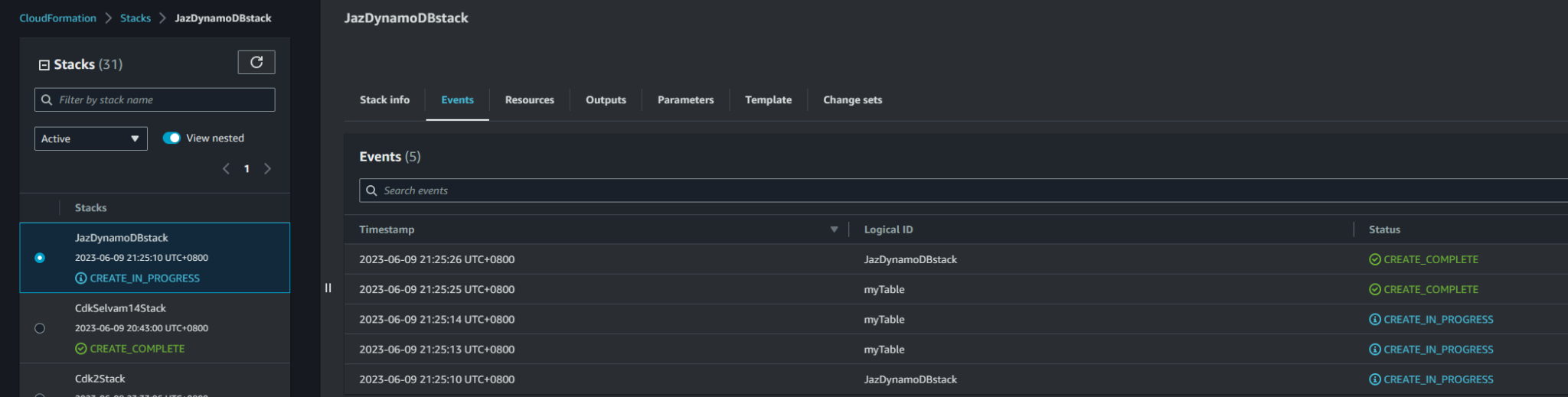
When you create a CloudFormation stack, AWS CloudFormation provisions the resources defined in the template in the order specified in the template. This helps to ensure that resources are created in the correct order and that dependencies between resources are managed correctly.

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-dynamodb.yaml
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 yaml 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. JazeelDynamoDBStack 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. E.g. searching for dynamodb and going to the service
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 DynamoDB via Cloudformation:

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

| AWSTemplateFormatVersion: "2010-09-09"  Resources:  myTable:  Type: AWS::DynamoDB::Table  Properties:  TableName: jaz-dynamodbtable #Replace this with your table name  BillingMode: PAY\_PER\_REQUEST  AttributeDefinitions:  - AttributeName: ID  AttributeType: S  KeySchema:  - AttributeName: ID  KeyType: HASH |
| --- |

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

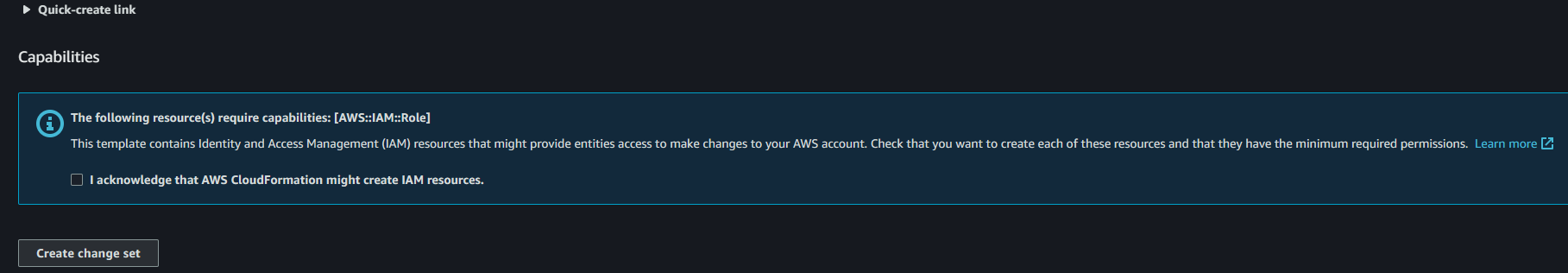
This template creates a DynamoDB table with on-demand mode(No servers created), and thus it operates based on pay per request basis.

[DynamoDB on-demand and provisioned capacity - Serverless Applications Lens (amazon.com)](https://docs.aws.amazon.com/wellarchitected/latest/serverless-applications-lens/capacity.html)

### Creating Lambda via Cloudformation:

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

Follow Steps 1 - 9 again for this stack(Give a unique name for the cloudformation stack as well.E.g. JazeelLambdaStack) However on the last page, you will need to tick the following checkbox before you click on Submit.



| Resources:  MyLambdaFunction:  Type: "AWS::Lambda::Function"  Properties:  Code:  ZipFile: |  def lambda\_handler(event, context):  print("Hello, World!")  return "Hello, World!"  Handler: index.lambda\_handler  FunctionName: jaz-cloudformation-lambda ##Replace this with your function name e.g. <your-name>-cloudformation-lambda  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.

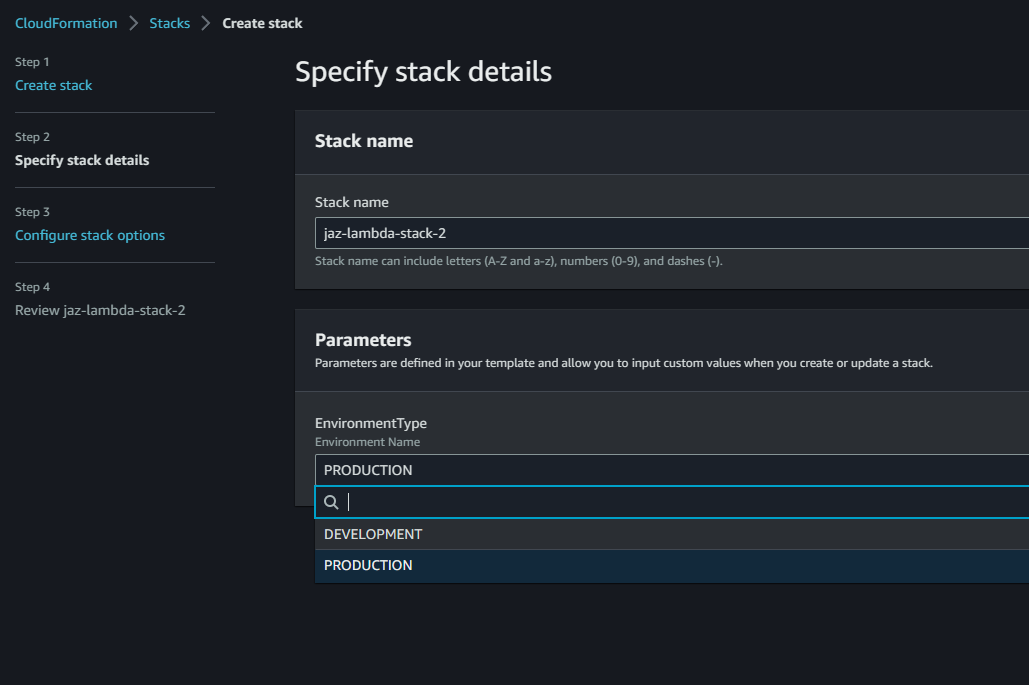
Likewise to the previous example, once you have created the stack, you can go to the lambda console to view the lambda function created. Once that is done, you can return to the cloudformation stack and delete your cloudformation stack.

### Creating Multi Environment Lambda via Cloudformation:

Traditionally, in the working world, we tend to work with multiple environments(e.g. development,UAT, Production etc.). What if you can use a single cloudformation file to deploy into multiple aws accounts or environments? That is when mappings and parameters come in. In this example, we would be creating a cloudformation stack using the following template file below to create another lambda function.

Save the following code into a yaml file as well and follow the earlier steps to create your new lambda stack.

However this time round in the console, you will see something new. You get to select either “DEVELOPMENT” or “PRODUCTION”. If you select “DEVELOPMENT”, Your lambda will get the value of LAMBDANAME under the DEVELOPMENT map assigned to its name, and vice-versa if you select PRODUCTION.



Description: This template deploys a lambda function in Development and Production Environments

Parameters:

EnvironmentType:

Description: Environment Name

Type: String

Default: DEVELOPMENT

AllowedValues: [DEVELOPMENT,PRODUCTION]

Mappings:

Environments:

DEVELOPMENT:

LAMBDANAME: jaz-lambda-development #Change this to your lambda name

PRODUCTION:

LAMBDANAME: jaz-lambda-production #Change this to your lambda name

Resources:

MyLambdaFunction:

Type: "AWS::Lambda::Function"

Properties:

Code:

ZipFile: |

def lambda\_handler(event, context):

print("Hello, World!")

return "Hello, World!"

Handler: index.lambda\_handler

FunctionName: !FindInMap [Environments, !Ref EnvironmentType, LAMBDANAME]

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"