AWS CloudFormation Master Class v2 [2022]

Course Objectives:

* Learn CloudFormation and master all its concepts.
* Go through Hands On examples to practice what we learned.
* Learn how to use YAML to write the CloudFormation templates.
* Learn how to write your infrastructure as code(laC).
* Launch Several templates.
* Advanced concepts overview

Who is this course for?

* Developers who want to learn about AWS CloudFormation and how to write templates.
* Devops who want to learn how deploy and orchestrate CloudFormation templates.
* Solutions Architect who wants to understand the benefit of using CloudFormation to manage infrastructure and steer their team to use CloudFormation.

Prerequisites:

* Basic Knowledge of AWS is required.
* AWS IAM
* AWS S3
* AWS EC2, Security Groups, Autoscaling groups.
* AWS Lamba
* A Few others…
* Knowledge of JSON and/or YAML is preferred.
* Recent MacOS /Linux /Windows Machine
* Lots of desire to learn new exciting things!!

Lesson 2:

What is CloudFormation:

* CloudFormation is a declarative way of outlining your AWS Infrastructure, for any resource (most of them are supported).
* For Example, within a CloudFormation template, you say:
* I want a security Group.
* I want two EC2 Instance using this security Group.
* I want two Elastic Ips for these EC2 instances.
* I want an S3 Bucket.
* I want a load Balancer (ELB) in front of these EC2 Instance.
* Then CloudFormation creates those for you, in the right order, with the exact configuration that you specify.

CloudFormation Template Example:


        A screenshot of the Designer with its panes and components numbered.
      

Benefits of AWS CloudFormation(l/2)

* Infrastructure as code :
* No resources are manually created, which is excellent for control.
* The code can be version controlled for example using Git.
* Changes to the infrastructure are reviewed through code.
* Cost:
* Each resource within the stack is tagged with an identifier so you can easily see how much a stack cost you.
* You can estimate the costs of your resource using the CloudFormation template.
* Savings strategy: In Dev, you could automation deletion of templates at 5PM And recreated at 8 AM, Safely.

Benefits of AWS CloudFormation (2/2)

* Productivity
* Ability to destroy and re-create an infrastructure on the cloud on the fly.
* Automated generation of Diagram for your templates!
* Declarative programming (no need to figure out ordering and orchestration).
* Separation of concern: Create many stacks for many apps, And many layers. EX:
* VPC stacks
* Networks stacks
* App Stacks
* Don’t re-invent the wheel.
* Leverage existing templated on the web!
* Leverage the documentation.

What this course won’t do?

* This course won’t go over every AWS Services.
* There are 200+ services,700+ Resource types so it’s impossible to cover them all.
* Instead, we’ll go over understanding how to write a CloudFormation template in the perfect way.
* Then you just pick up the documentation for your services and it will be as easy as 1..2..3!

Course Cost:

* AWS CloudFormation is free, but the resources it may create are not.
* Some resources are covered by the AWS Free Tier, and some are not.
* Overall, if you want to experiment after the class, budget around $10.
* Don’t forget to delete your CloudFormation stacks after the lectures!!

Quiz 1:

Question 1:

Using AWS CloudFormation is free, but you are charged for the resources it creates (e.g., EC2 Instances, Elastic Load Balancers, etc.).

And:True

Question 2:

Which AWS service helps you create your AWS Infrastructure resources in the right order you specify?

Ans:AWS Cloud Formation.

**Section -2: Code Download & Tools Setup:**

**Code Download**

Please follow the instructions at [**https://courses.datacumulus.com/downloads/aws-cloudformation-1b5**](https://courses.datacumulus.com/downloads/aws-cloudformation-1b5) to download the code.

**VSCode Setup**

**Tools used in the course**

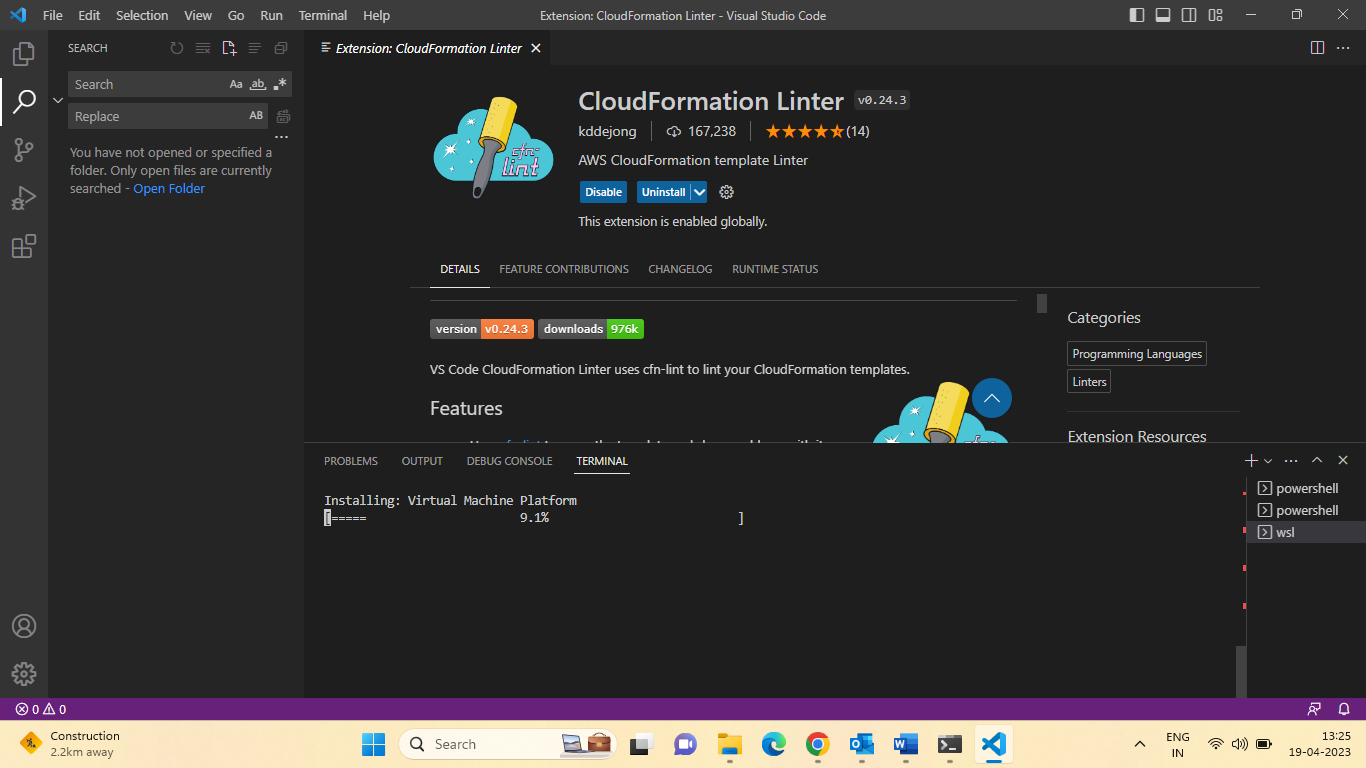
Note: the setup is done in video form in the next lecture

VSCode: <https://code.visualstudio.com/>

With the cfn-lint extension: <https://github.com/aws-cloudformation/cfn-lint-visual-studio-code>

Install cfn-lint here: <https://github.com/aws-cloudformation/cfn-lint>

<https://www.techielass.com/install-cfn-lint-on-windows/>

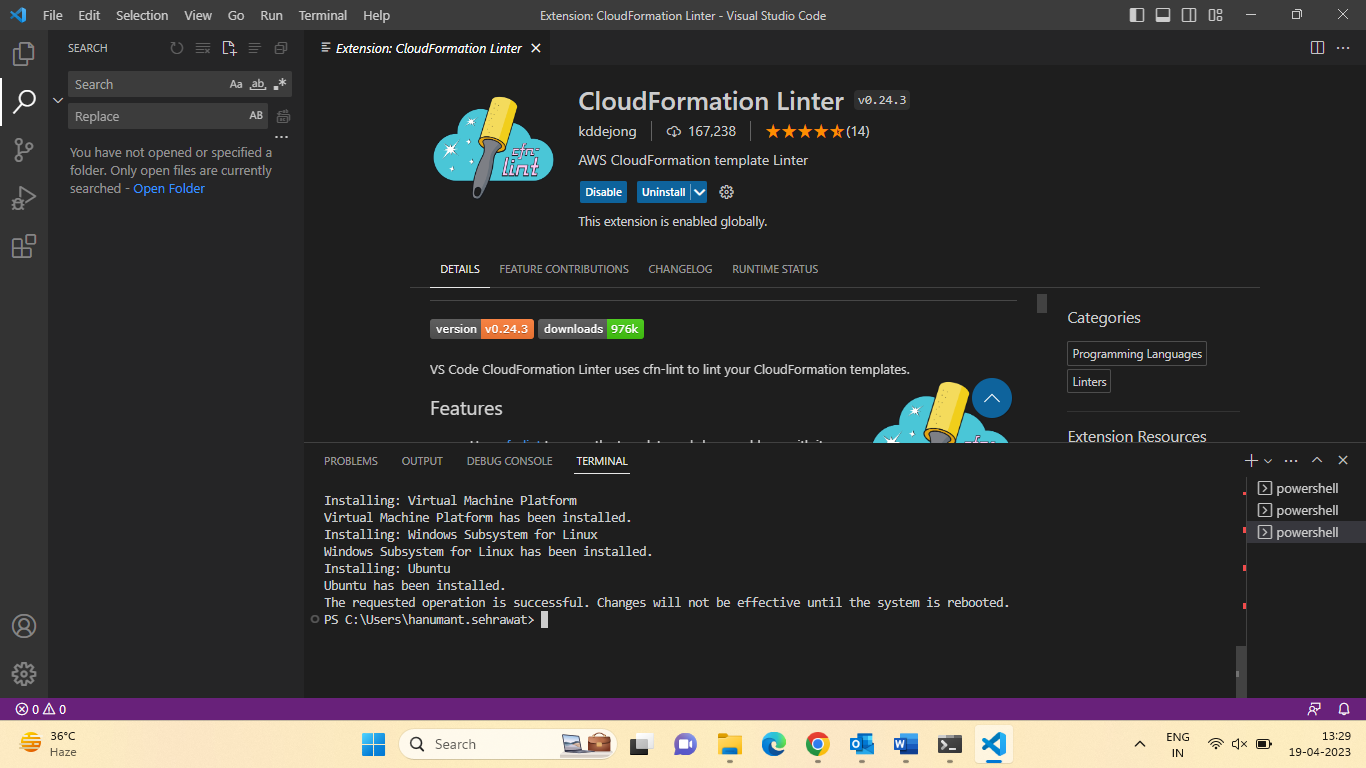
wsl –install 

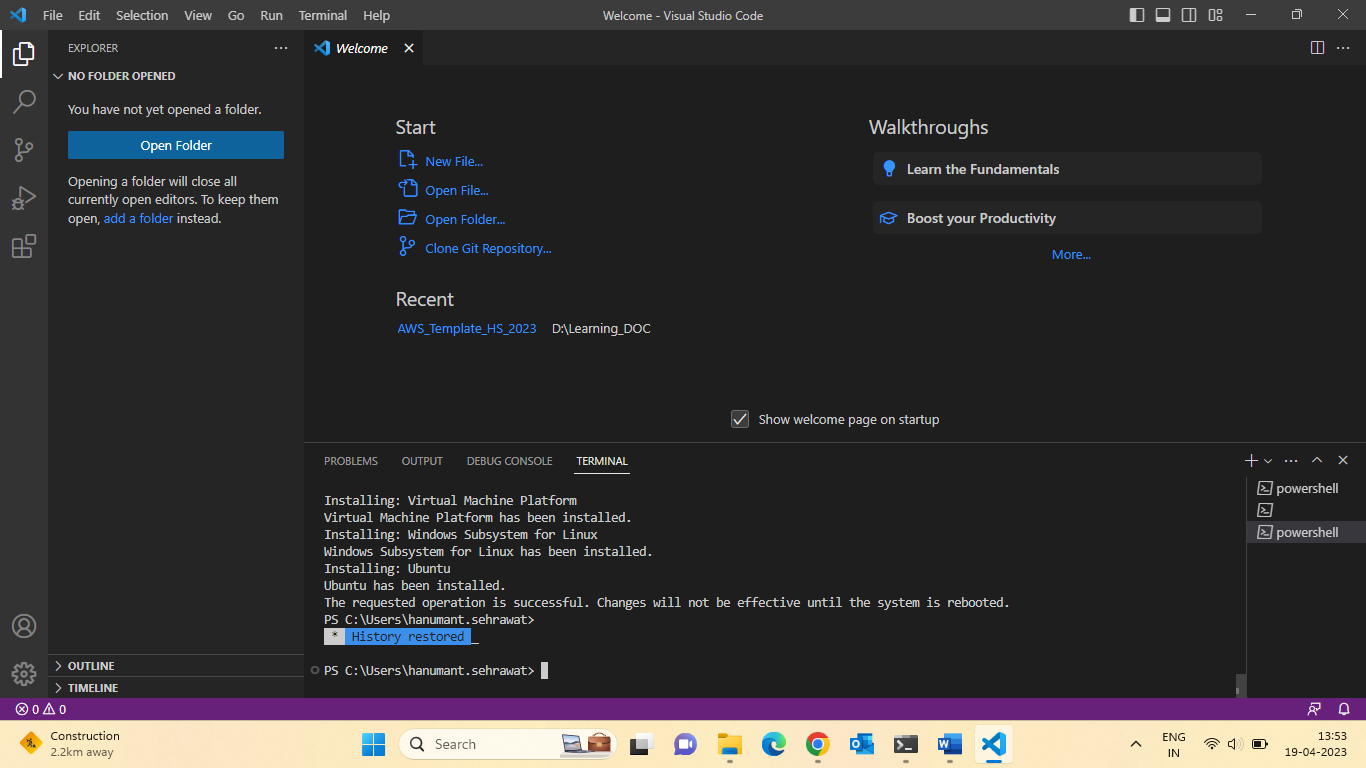
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated





Also install python

Let install the CloudFormation extension in visual studio code.

1.CloudFormation Linter

In visual studio code install

1. pip install cfn-lint

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

For install upgrade python:

Python.exe -m pip install -- upgrade pip

Pip install pydot

**Section 3: Cloud Formation Introduction:**

Introductory Example:

* We are going to create a simple EC2 Instance:
* Add we are going to add security group to it.

Webserver (open port :20,80)

WebServerSecurityGroup

* For now, forget about the code syntax.
* We all look at the structure of the files later.
* We will see how in no-time we are able to ger started with CloudFormation!!

First create the EC2 :

**Resources**:

**MyInstance:**

    Type: AWS::EC2::Instance

    Properties:

      AvailabilityZone: ap-south-1b

      ImageId: ami-07d3a50bd29811cd1

      InstanceType: t2.micro

Let update the EC2 open the some port:

**Resources**:

**MyInstance:**

    Type: AWS::EC2::Instance

    Properties:

      AvailabilityZone: ap-south-1b

      ImageId: ami-07d3a50bd29811cd1

      InstanceType: t2.micro

      SecurityGroups:

        - !Ref SSHSecurityGroup

        - !Ref ServerSecurityGroup

  # an elastic IP for our instance

**MyEIP:**

    Type: AWS::EC2::EIP

    Properties:

      InstanceId: !Ref MyInstance

  # our EC2 security group

**SSHSecurityGroup:**

    Type: AWS::EC2::SecurityGroup

    Properties:

      GroupDescription: Enable SSH access via port 22

      SecurityGroupIngress:

      - CidrIp: 0.0.0.0/0

        FromPort: 22

        IpProtocol: tcp

        ToPort: 22

  # our second EC2 security group

**ServerSecurityGroup:**

    Type: AWS::EC2::SecurityGroup

    Properties:

      GroupDescription: allow connections from specified CIDR ranges

      SecurityGroupIngress:

      - IpProtocol: tcp

        FromPort: 80

        ToPort: 80

        CidrIp: 0.0.0.0/0

      - IpProtocol: tcp

        FromPort: 22

        ToPort: 22

        CidrIp: 172.31.0.0/16

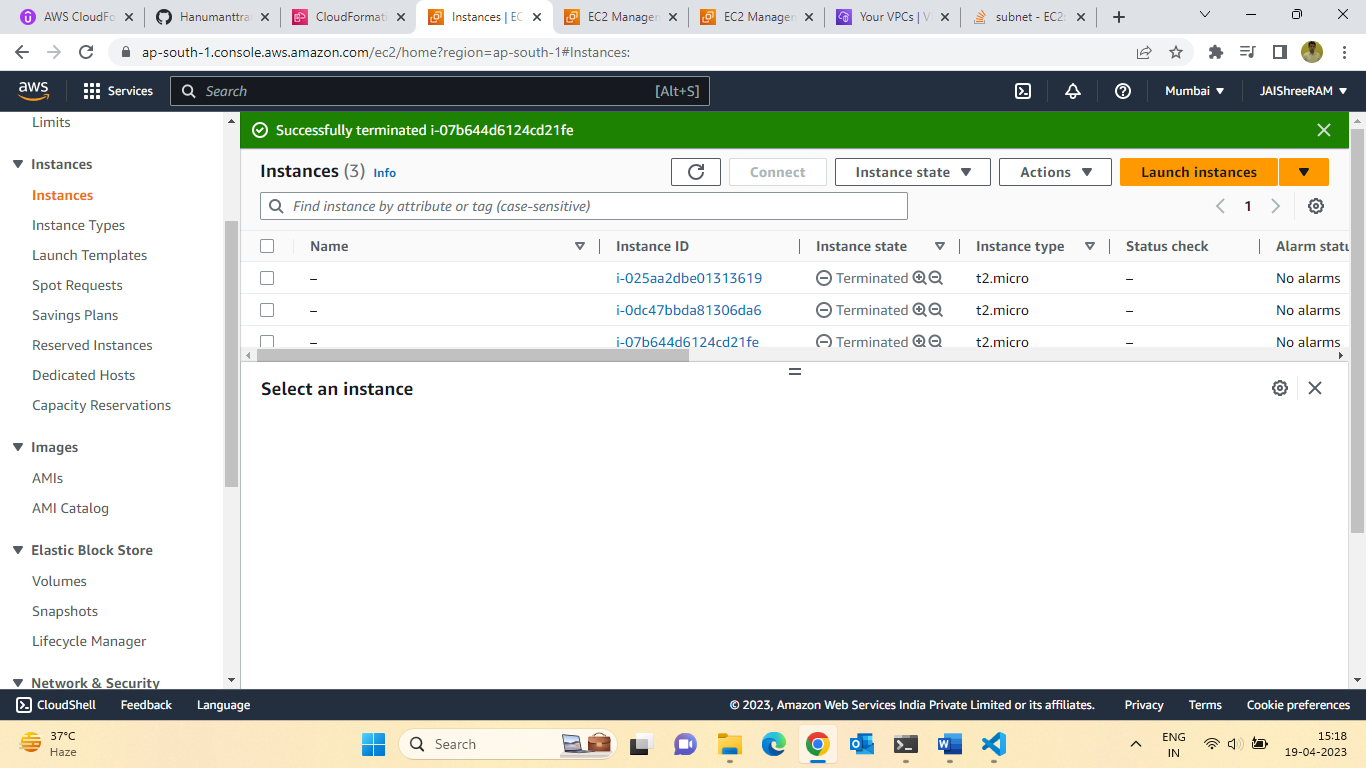
Let delete the stack & check ec2 delete automatically.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



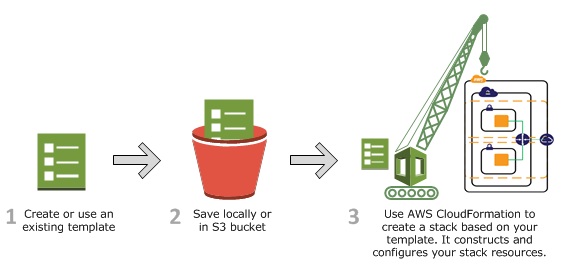
8.Summary of how cloud formation

How CloudFormation Works:

* Templates must be upload in S3 and the referenced in CloudFormation.
* TO Update a template, we cannot edit previous ones. We have to reupload a new version.

Of the template to AWS.

* Stacks are identified by a name.
* Deleting a stack deletes every single artifact that was created by CloudFormation.



**CloudFormation Introduction Quiz**

Q.1 When you create a CloudFormation Stack, the CloudFormation template must be first uploaded to ...

Ans: Amazon Simple Storage Services (S3)

Q.2 To update a CloudFormation Stack, you can edit the previous CloudFormation template you uploaded before.

Ans:False

**9.YAML Tutorial:**

YAML Crash Course:

* YAML & JSON are the languages you can use for CloudFormation.
* JSON is horrible for CF.
* YAML is great in so many ways.
* Let’s learn a bit about it:
* Key Value Pairs
* Nested objects:
* Support Arrays (-)
* Multi line strings
* Can include comments!

Exercise: Try to convert this JSON in YAML

* Check out 2-First-Hands-on/0-exercise.json.

Url for covert JSON to YAML Code.

11.S3 Bucket Creation using CloudFormation:

Hands-On: Creating a S3 Bucket:

* Creating a S3 Bucket is free
* S3 is the AWS services for storing static files in a replicated and globally available way.
* It Powers many websites, Single Page Apps, Hosts, All the Netflix video content,
* We will use CloudFormation to provision a S3 Bucket!

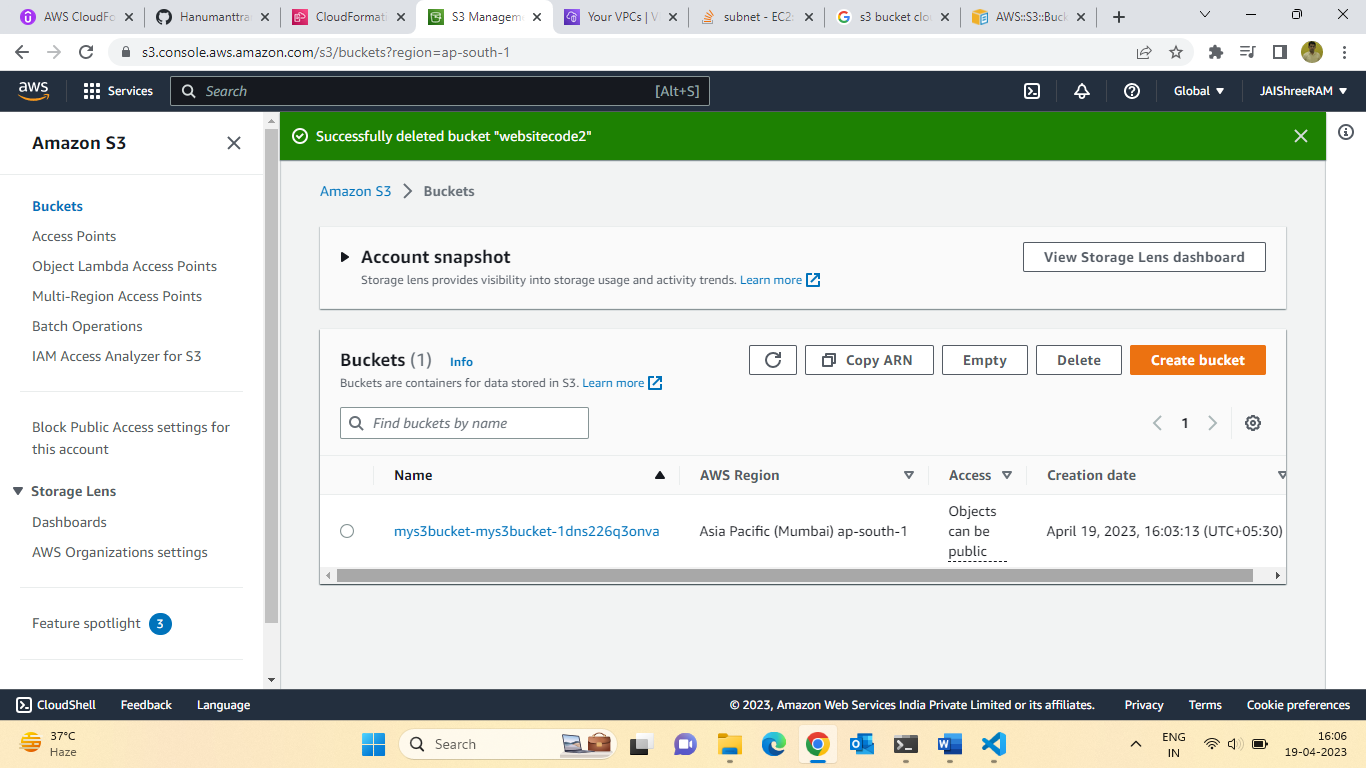
**Resources**:

**MyS3Bucket:**

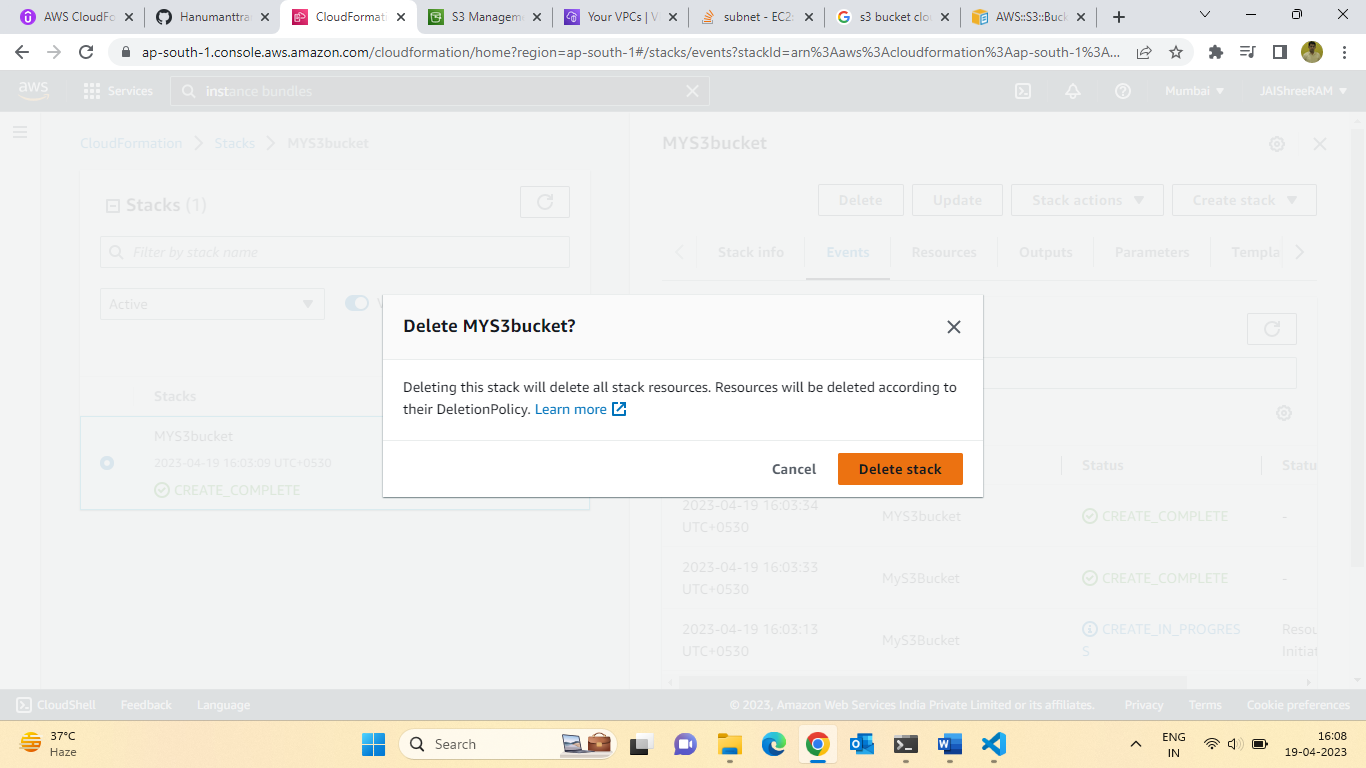
    Type: AWS::S3::Bucket

    Properties: {}

Let check the bucket is provision or not:



Let delete the stack of check automatic delete bucket:



12: Let create the S3 Bucket for upgrading using CloudFormation:

CloudFormation Update Behaviour:

* CloudFormation updates resources based on difference between what you.

Submit and stack’s current template.

* Which method to use depends on which property you update for a resource.
* Update with No interruption
* Without disrupting resource operation & without changing physical ID.
* Example: Updating the IAM instance profile (IAMInstanceProfile) of an EC2 Instance.
* Update with Some interruption:
* Example: Updating an EC2 instance Type (InstanceType) from t2. micro to t2.small (etc).
* Replacement
* Recreating the resource with new physical ID.
* Creates the new resources, change reference from the other resource to the new resource, then deletes the old resource.
* Example, Updating an RDS DB instance availability zone (Availability Zone).

Hands -on: Updating a S3 Bucket:

* We will consider two types of updates:
* Updates with no interruption (adding Access Control)
* Replacement Updates (updating the name of the bucket)
* We will see how CloudFormation handles these two cases:

1st cases:

Updates with no interruption (adding Access Control)

**Resources**:

**MyS3Bucket:**

    Type: AWS::S3::Bucket

    Properties:

      AccessControl: PublicRead

2nd cases:

Replacement Updates (updating the name of the bucket)

**Resources**:

**MyS3Bucket:**

    Type: AWS::S3::Bucket

    Properties:

      AccessControl: PublicRead

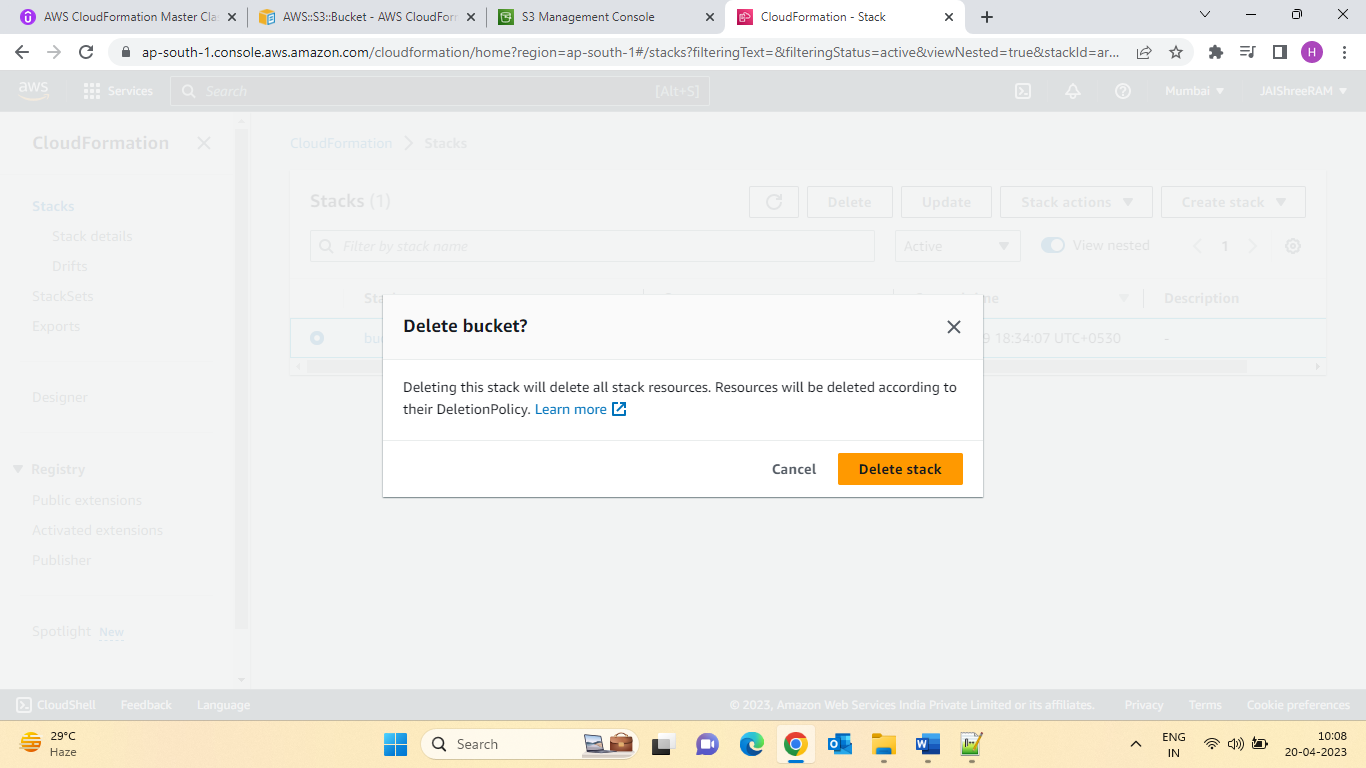
      BucketName: "mys3bucket21234"

Topics :13 Deleting an S3 Bucket using cloud Formation:

Hands-on: deleting a S3 bucket:

* You can’t delete a non-empty S3 Bucket
* To delete a non-empty S3 Bucket, you must delete all the objects inside it first.
* We’ll delete our stack and see how CloudFormation reacts.

Let delete the Stack & Check if bucket is delete or not.



A screenshot of a computer

Description automatically generated

My Bucket is deleted :

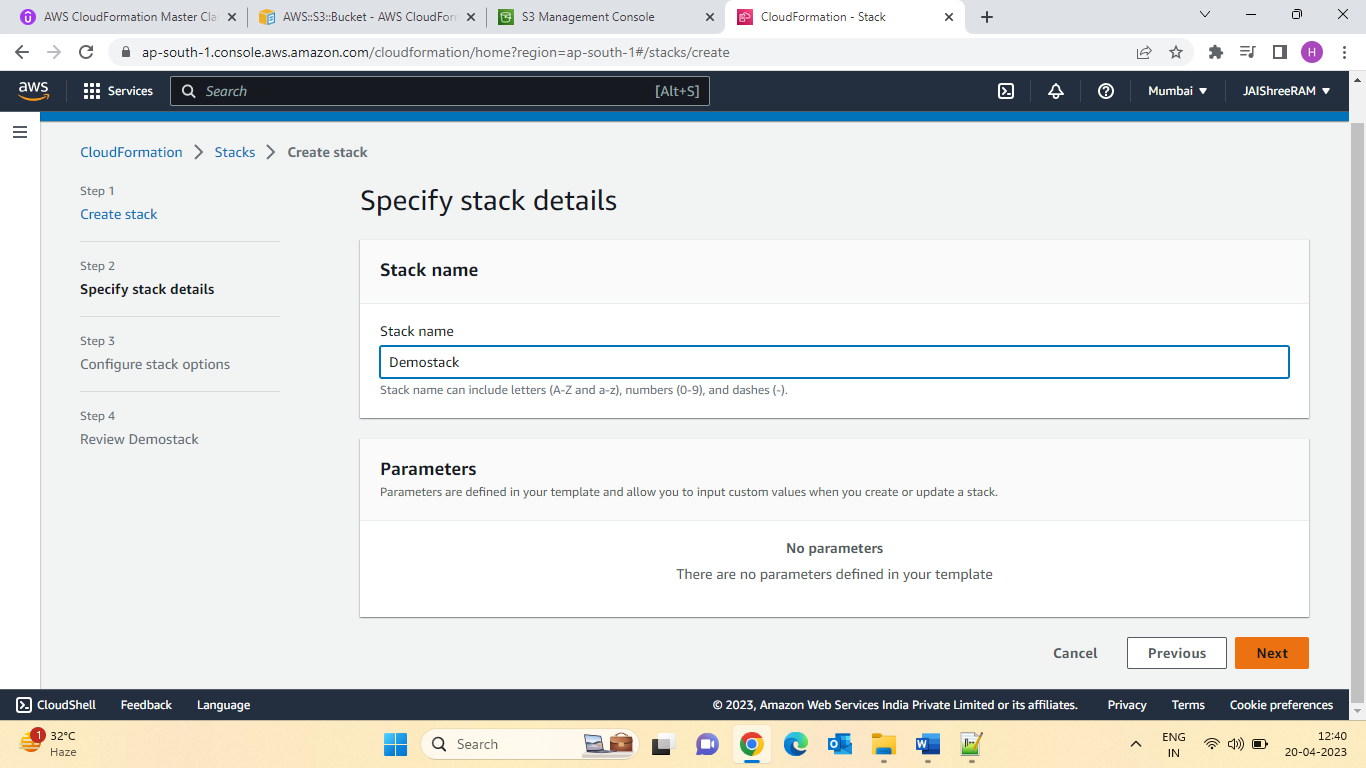
A screenshot of a computer

Description automatically generated

14.CloudFormation Templates Options:

Understanding the CloudFormation Template Options:

* Let’s learn about the parameters that are common to any CloudFormation template.
* Tags
* Permissions
* Notification Options
* Timeout
* Rollback on Failure
* Rollback Configuration (Monitoring time & CloudWatch Alarm)
* Stack Policy
* Termination Protection
* Quick-Start Link
* We’ will discuss them in greater details during the course!!



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

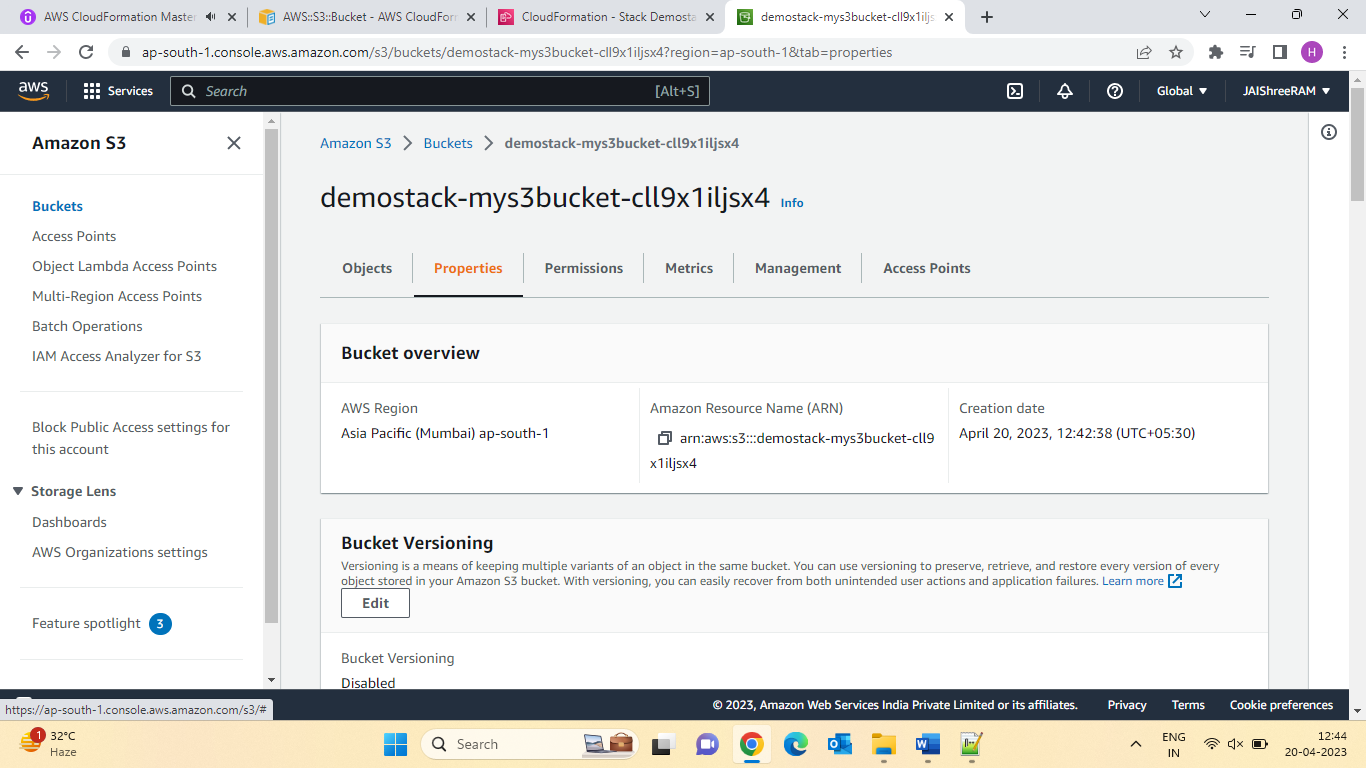
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Let check the preview in S3Bucket:



A screenshot of a computer

Description automatically generated

15.Using CloudFormation Designer:

* The CloudFormation Designer can really help visualize a CloudFormation stack.
* It ‘also very handy to just quickly draft a CloudFormation template.
* It has strong capabilities to verify that your template is also well written.
* Let’s learn how to use it!.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

16.CloudFormation Building Blocks:

Template’s Components (one course section for each 4-10):

1. AWS Template Format Version: identifies the capabilities of the template (“2010-09-09”)
2. Description comments about the template.
3. Transform: specifies one or more Macros that used to process the template
4. Metadata
5. Resources: Your AWS resources declared in the template (MANDATORY)
6. Parameters: the dynamic inputs for your template
7. Mappings: the static variables for your template
8. Outputs: Reference to what has been created
9. Conditionals: List of conditions to perform resource creation
10. Rules: validate a parameter(s) during stack creation/update

Templates’ helpers (learning as we encounter them):

1. references:
2. Functions

Section 17: Deploying CloudFormation Templates:

* Manual Way:
* Editing templates in CloudFormation Designer or code editor
* Using the console to input parameters, etc…
* We will mostly do this way in the course for learning purposes.

A picture containing text, screenshot, diagram, font

Description automatically generated

* Automated Way:
* Editing templates in a YAML File
* Using the AWS CLI (command line interface) to deploy the templates, or using a Continuous Delivery (CD) tool .
* Recommended way when you fully want to automate your flow.
* We’ll learn about it in the advanced section of the course.

Section 18: Cost estimates for your templates:

Quiz 3: Getting started Quiz.

Q.1.You can write CloudFormation templates using ....

Ans:Either JSON Or YAML

Q.2.When you update a CloudFormation stack, there is a probability that the updated resource will be replaced with a new one. What is the name of this Update Behaviour?

Ans: Replacement

Q.3. What tool can be used to automate CloudFormation template deployments?

Ans:AWS Cli.

Q.4. Which of the following is not part of the CloudFormation building blocks?

Ans:Security Groups

Section 5: Cloud Formation Parameters:

What are parameters?

* Parameters are way to provide inputs to your AWS CloudFormation template.
* They are important to Know about if:
* You want to reuse your templates across the company.
* Some inputs cannot determine ahead of time.
* Parameters are extremely powerful, controlled and can prevent errors from happing in your templates, thanks to types.
* Parameters can be cross validated using Rules (later in the course).

When should you use a parameter?

* Ask yourself this:
* Is this CloudFormation resource configuration likely to change in the future?
* If so, make it a parameter.
* You won’t have to re-upload a template to change its content.

For example:

**Parameters**:

**SecurityGroupDescrption:**

**Type:** String

**Description:** Security Group Description (simple parameter)

Parameters Settings:

* Parameters can be controlled by all these settings:
* Type
  + - String
    - Number
    - Comma Delimited List
    - List<Number>
    - AWS-Specific Parameter (to Help catch invalid -match against existing value in the AWS Account).
    - List<AWS-Specific Parameter>
    - SSM Parameter (get parameter value from SSM Parameter store)
    - Description
    - Constraint Description (String)
    - Min/Max length
    - Min/Max Value
    - Default
    - AllowedValues(Array)
    - Allowedpattern(regex)
    - NoEcho (Boolean)

How to Reference a Parameter

* The Fn::Ref function can be leveraged to reference parameters.
* Parameters can be user anywhere in a template, except:
* AWSTempalteFormatVersion
* Description
* Transform
* Mappings
* The shorthand for this in YAML is !Ref

For example:

DBsubnet1:

**Type:** AWS::EC2::Subnet

**Properties:**

   VpcId: !Ref Myvpc

* The function can also reference other elements within the template: