

# CLOUD FORMATION TEMPLATE

A **CloudFormation template** is a **declarative text file (written in YAML or JSON)** that describes the infrastructure you want to create in **AWS**. It allows you to provision and manage AWS resources in an **automated and repeatable way**.

## Why Use CloudFormation Templates?

- **Automates Infrastructure Deployment:** No manual clicking in the console.
- **Infrastructure as Code (IaC):** Treat your infrastructure like code (version control, reuse, etc.).
- **Repeatable and Consistent:** Deploy the same environment again and again without errors.
- **Simplifies Resource Management:** All resources are managed as a single unit (called a stack).

## Key Sections:

1. **AWS::TemplateFormatVersion:** Template version (optional).
2. **Description:** A short description of what the template does (optional).
3. **Metadata:** Provides additional information about the template (optional).
4. **Parameters:** User inputs like instance types, key pairs, etc (optional).
5. **Rules:** Defines rules to validate parameter values (optional).
6. **Mappings:** Defines fixed values based on region, environment, etc (optional).
7. **Conditions:** Defines conditions to control resource creation based on parameter values (optional).
8. **Resources:** The actual AWS resources you want to create (mandatory).
9. **Outputs:** Outputs after the stack is created (optional).

## How CloudFormation Works:

1. **Write the Template:** Describe your infrastructure in YAML/JSON.
2. **Upload to AWS CloudFormation:** AWS reads the template.
3. **CloudFormation Provisions Resources:** AWS creates all the resources automatically.
4. **Manage with Stacks:** All resources are treated as one unit called a **stack**.

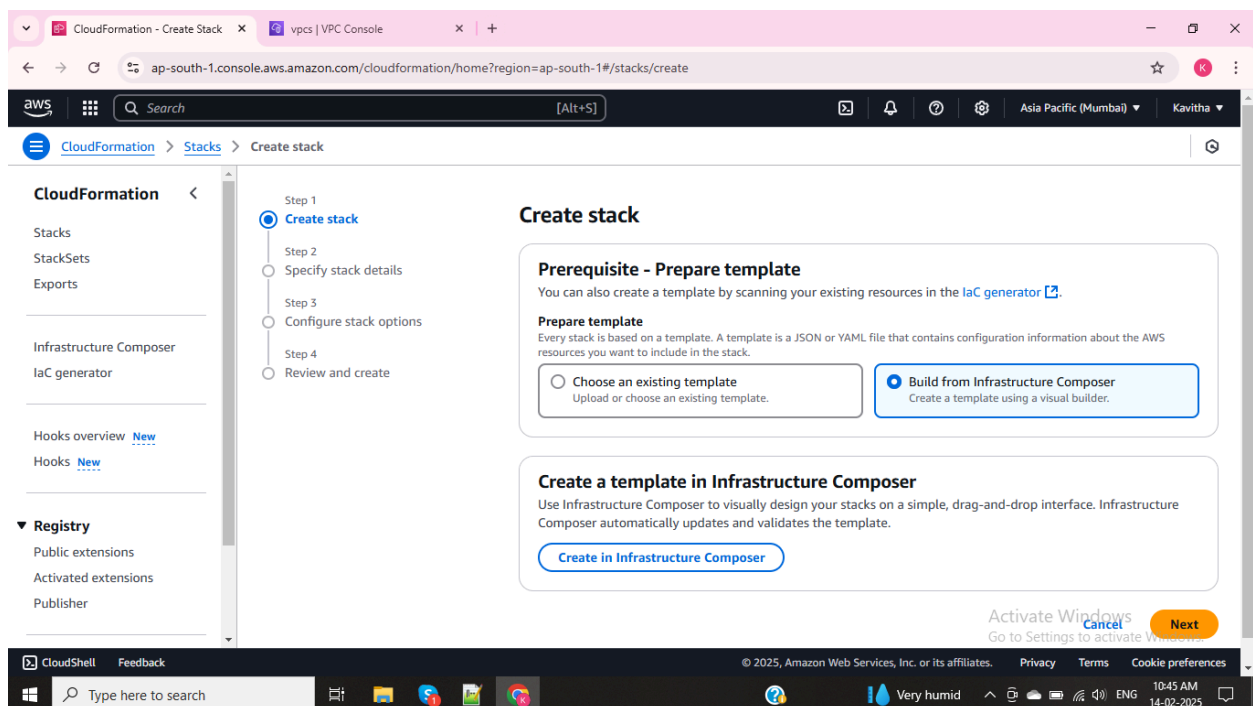
## Example:

You can create an **EC2 instance**, **VPC**, **subnets**, and more using one CloudFormation template, instead of manually creating each one in the AWS console.

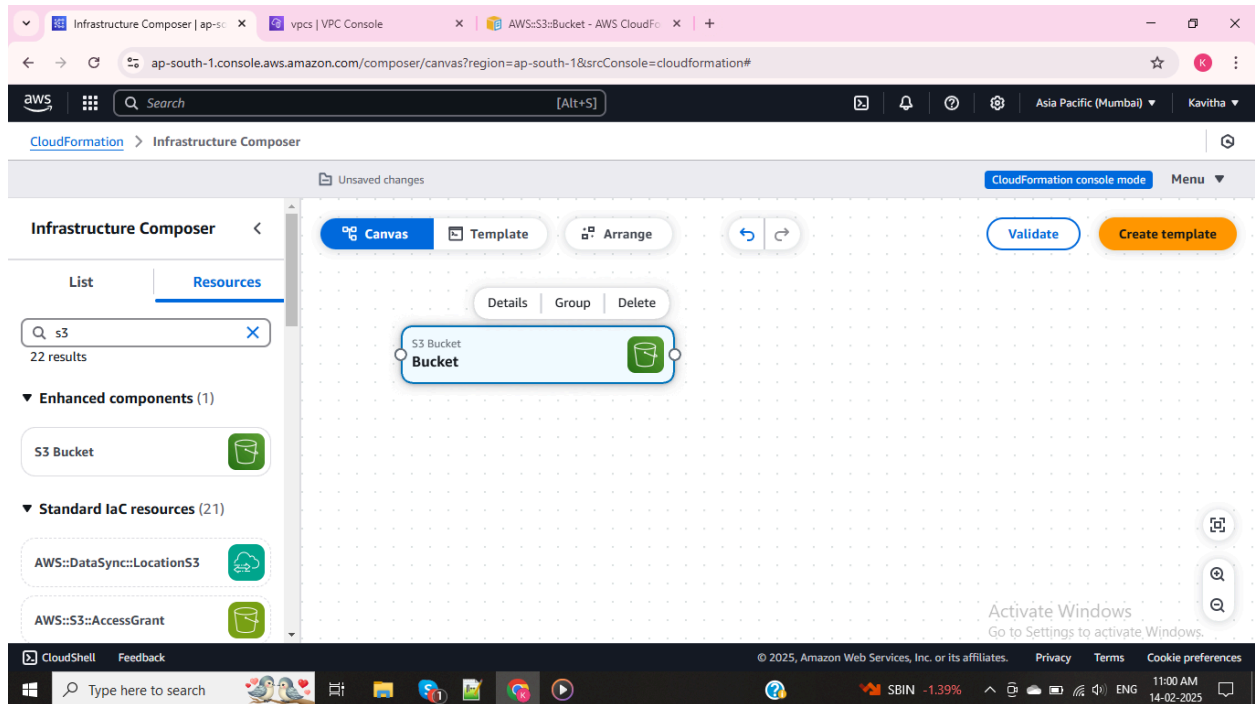
## TASK 1: Creating S3 Bucket using CloudFormation Template

### ✓ 1. Using **AWS Console**

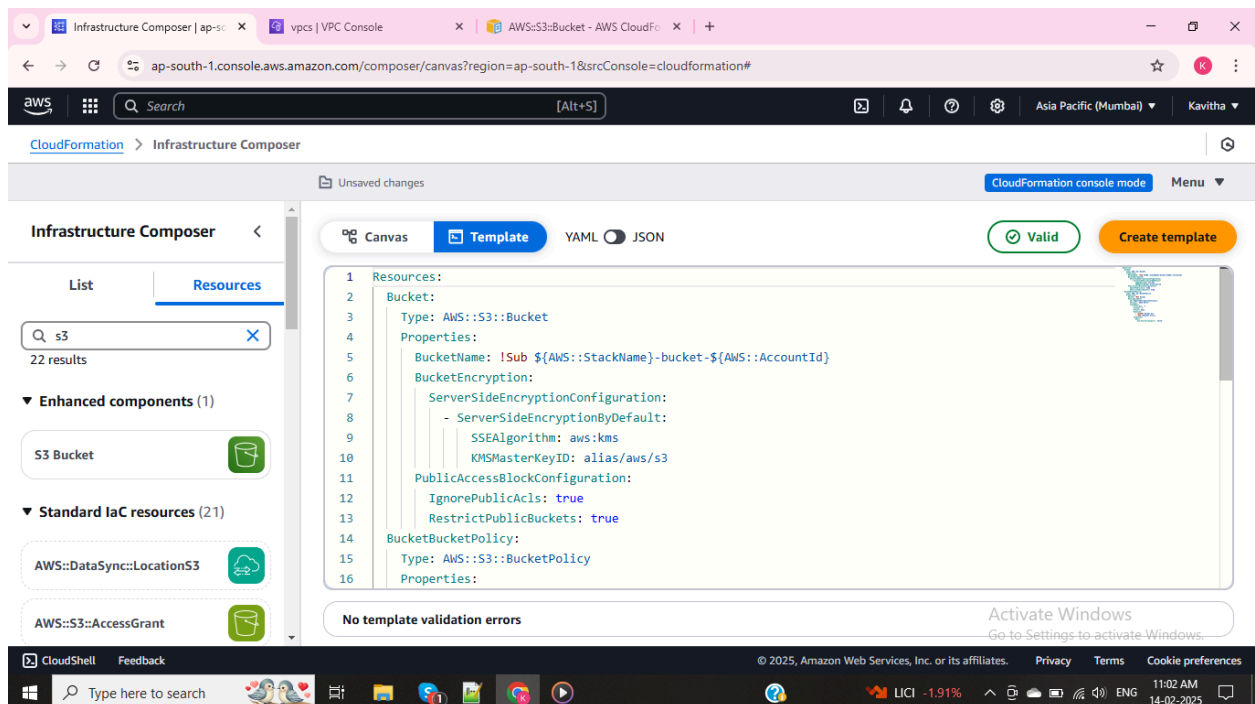
1. Go to AWS Management Console → **CloudFormation Template**
2. Click **Create Stack**
3. Choose Build from Infrastructure Composer



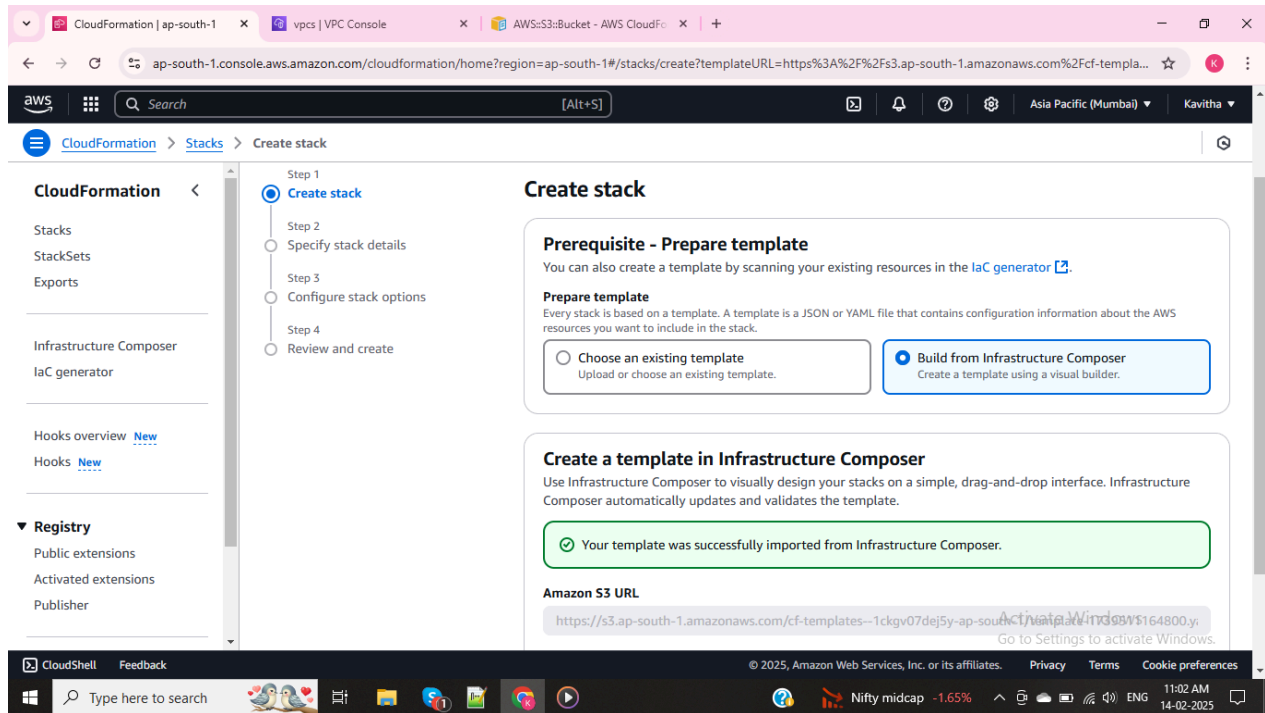
4. Click **Next** --> In Canvas search **S3 BUCKET** and drag into right side --> Create Template



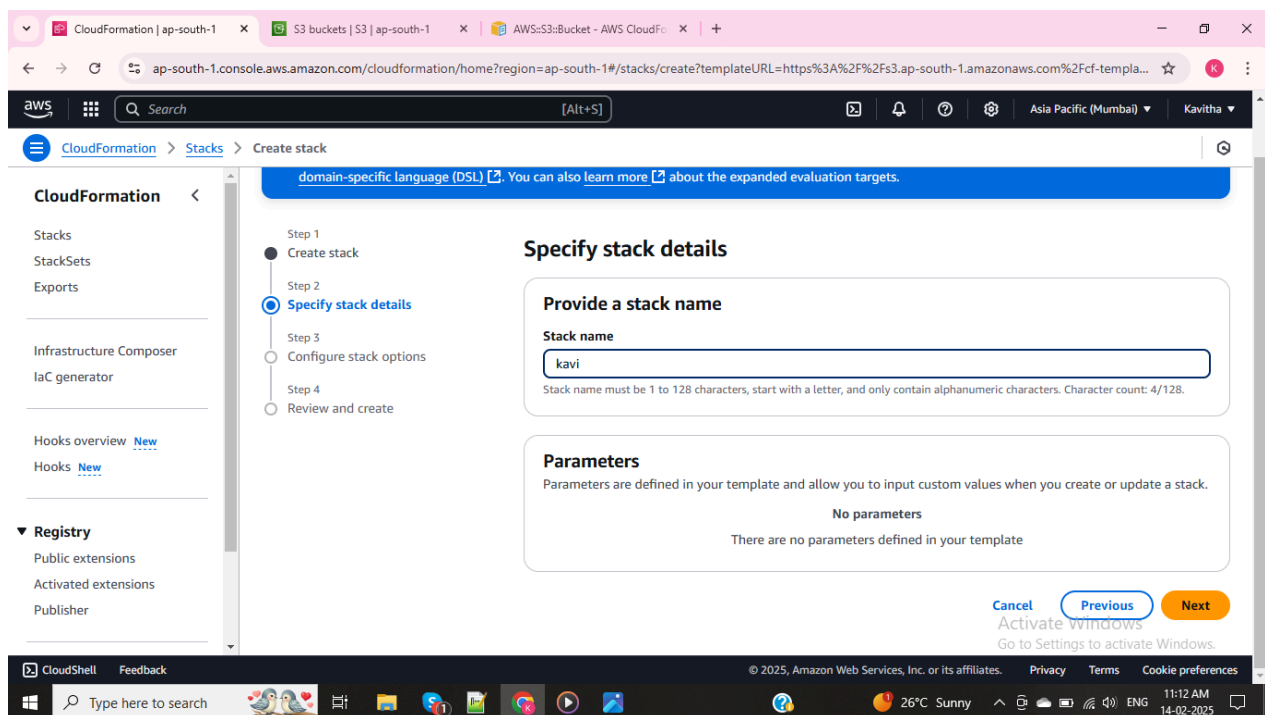
Automatically Template is created for S3 Bucket and validate whether code is correct



## 6. Template created successfully --> Next



## Provide Stack name



## 7. Stack created completed

The screenshot shows the AWS CloudFormation console for the 'kavi' stack. The stack is in the 'CREATE\_COMPLETE' state, as indicated by the green checkmark and the 'CREATE\_COMPLETE' status in the 'Events' tab. The 'Events' tab shows two events: 'kavi' and 'BucketBucketPolicy', both with a status of 'CREATE\_COMPLETE'.

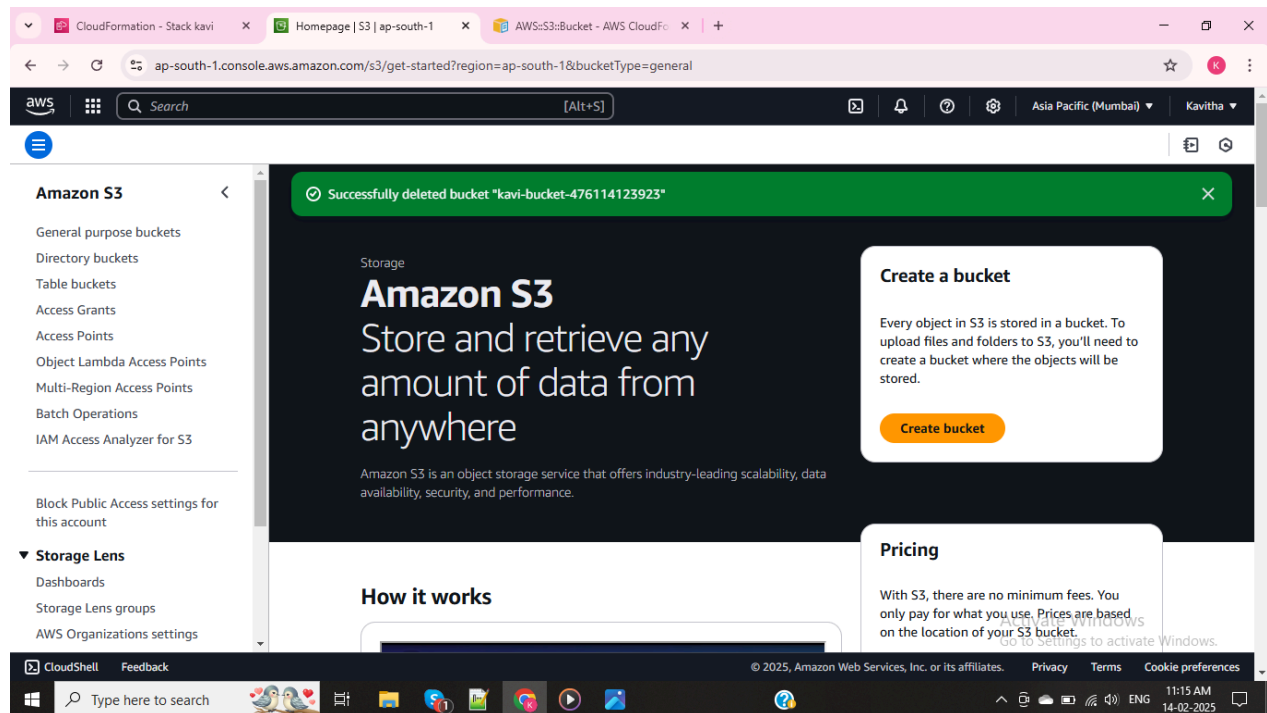
Timestamp	Logical ID	Status	Details
2025-02-14 11:13:26 UTC+0530	kavi	CREATE_COMPLETE	-
2025-02-14 11:13:25 UTC+0530	BucketBucketPolicy	CREATE_COMPLETE	-

Go to S3 service and check S3 Bucket is created or not and **OUTPUT**

The screenshot shows the AWS S3 console for the 'kavi-bucket-476114123923' bucket. The bucket is listed under 'General purpose buckets' and is in the 'Asia Pacific (Mumbai) ap-south-1' region. The bucket was created on February 14, 2025, at 11:13:11 (UTC+05:30).

Name	AWS Region	IAM Access Analyzer	Creation date
cf-templates--1ckgv07dej5y-ap-south-1	Asia Pacific (Mumbai) ap-south-1	<a href="#">View analyzer for ap-south-1</a>	February 14, 2025, 11:12:22 (UTC+05:30)
kavi-bucket-476114123923	Asia Pacific (Mumbai) ap-south-1	<a href="#">View analyzer for ap-south-1</a>	February 14, 2025, 11:13:11 (UTC+05:30)

Now, S3 bucket created using Cloudformation Template but bucket was deleted manually



**Drift Detection** in AWS CloudFormation is a feature that helps you identify whether the actual state of your AWS resources differs from the expected state defined in your CloudFormation templates.

### Why Drift Detection Matters:

- Over time, manual changes (outside CloudFormation) may be made to AWS resources.
- Drift detection helps ensure that your infrastructure remains consistent with your original template definitions.

### Key Concepts:

- **Drifted:** A resource has been changed manually outside CloudFormation.
- **In-Sync:** A resource matches its CloudFormation template definition.
- **Drift Detection:** The process of comparing your current stack resources with the template definitions.

## OUTPUT

**Drifts**

**Stack drift status**

Drift detection enables you to detect whether a stack's actual configuration differs, or has drifted, from its template configuration. [Learn more](#)

**Drift status**  
⚠️ DRIFTED

**Last drift check time**  
2025-02-14 11:16:09 UTC+0530

Only resources which currently support drift detection are displayed here. To view all of your stack resources, see your stack details page. [Learn more](#)

**Resource drift status (1)**

View drift details Detect drift for resource

Search resources

Logical ID	Physical ID	Type	Drift status	Timestamp	Module
Bucket	<a href="#">kavi-bucket-476114123923</a>	AWS::S3::Bucket	⚠️ DELETED	2025-02-14 11:16:09 UTC+0530	-

## TASK 2: Creating EC2 Instance using CloudFormation Template

Create another **Stack** --> Choose Build from infrastructure composer

**Create stack**

**Prerequisite - Prepare template**

You can also create a template by scanning your existing resources in the [IaC generator](#).

**Prepare template**

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

☐ Choose an existing template  
Upload or choose an existing template.

☒ **Build from Infrastructure Composer**  
Create a template using a visual builder.

**Create a template in Infrastructure Composer**

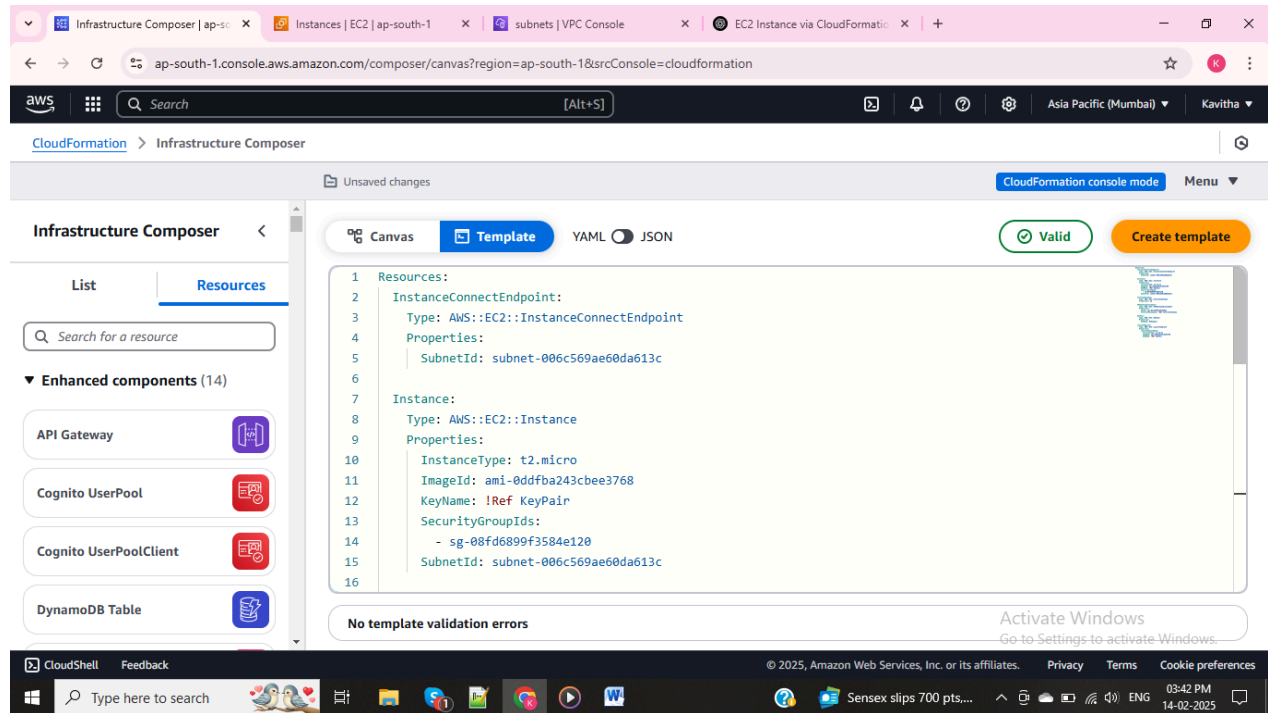
Use Infrastructure Composer to visually design your stacks on a simple, drag-and-drop interface. Infrastructure Composer automatically updates and validates the template.

✓ Your template was successfully imported from Infrastructure Composer.

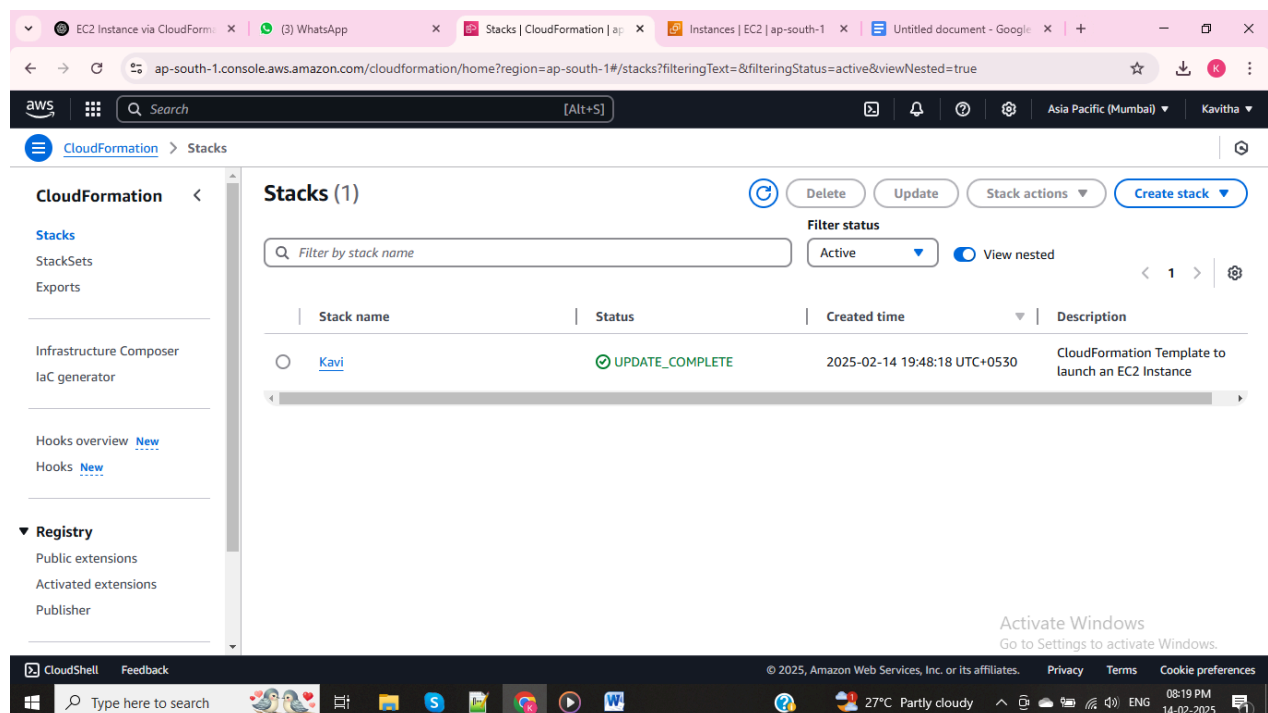
**Amazon S3 URL**

<https://s3.ap-south-1.amazonaws.com/cf-templates--1ckgv07dej5y-ap-south-1/template-1739512126056.ya>

## Creating Template for EC2 Instance and validated

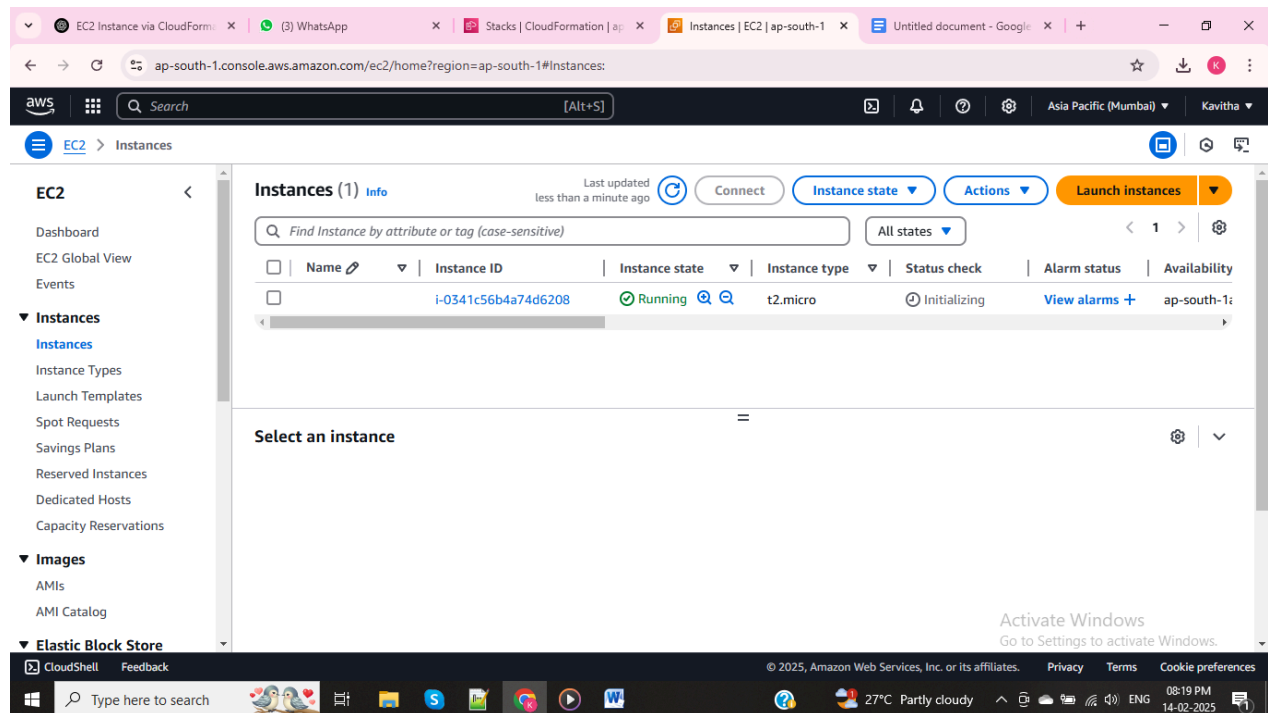


## Stack was created successfully

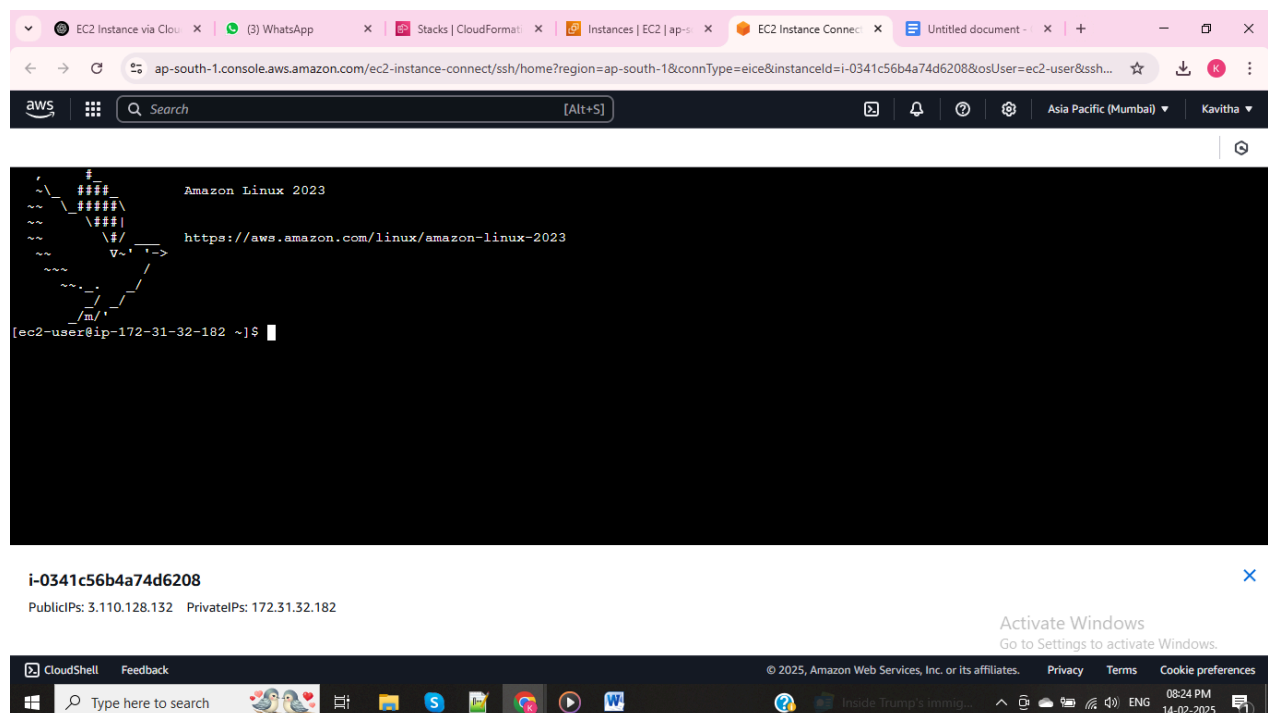




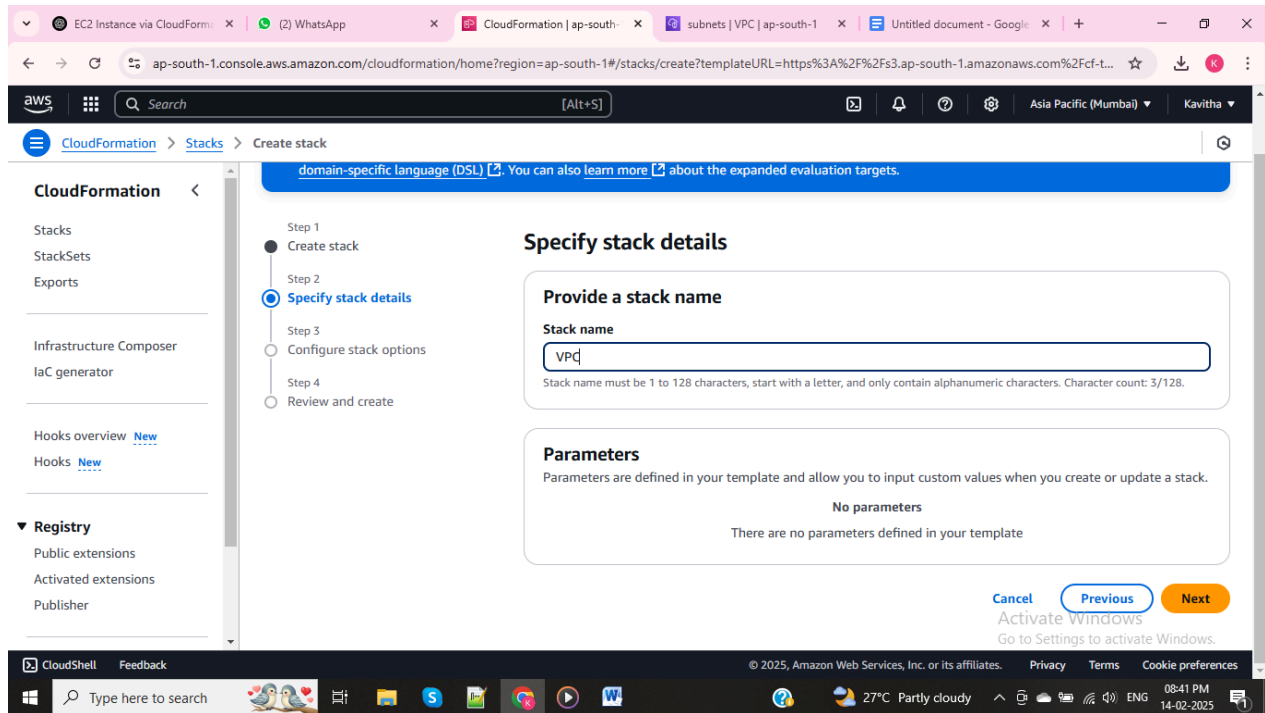
Now, Go to EC2 Instance dashboard and check Instance was created & **OUTPUT**



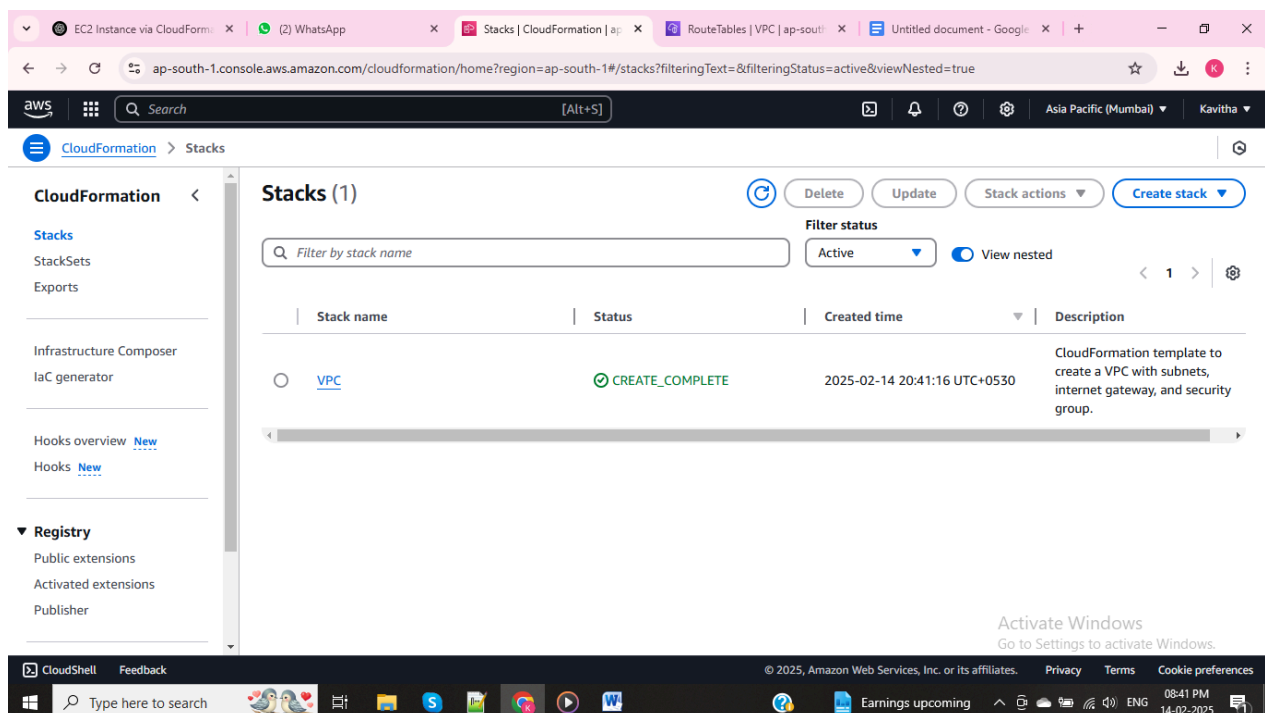
## Connecting the EC2 Instance



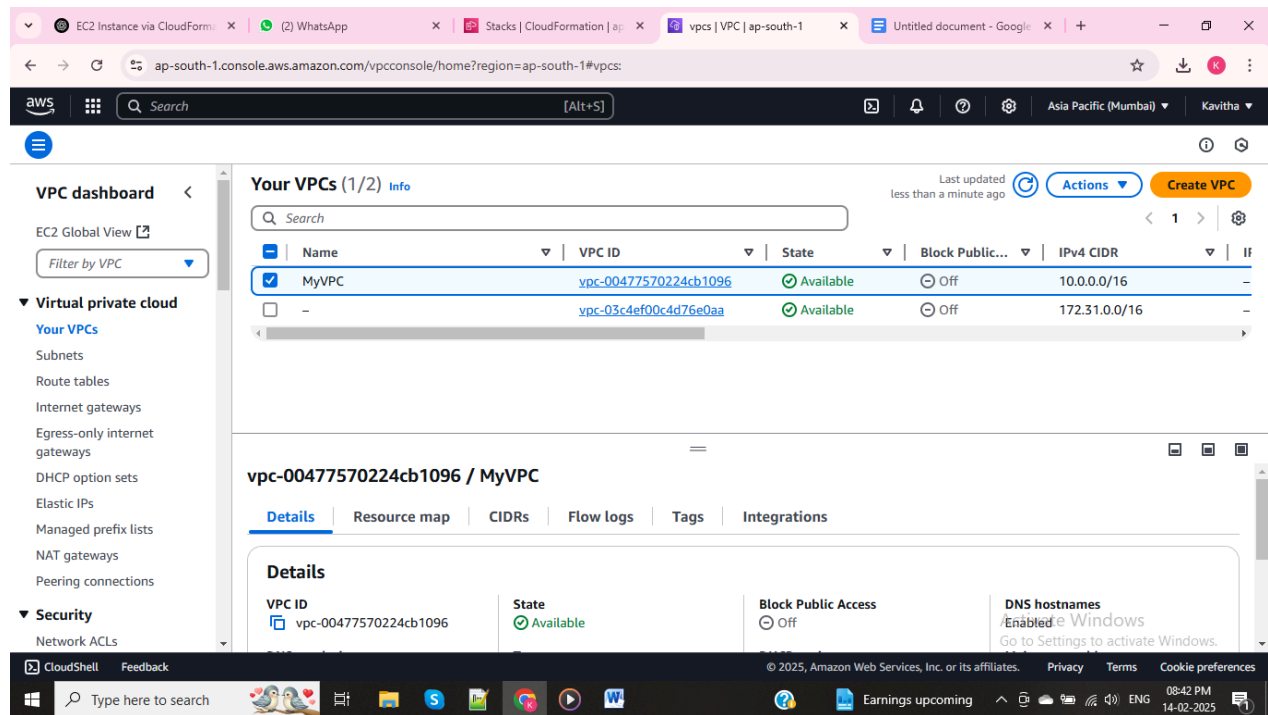
## TASK 3: Creating CloudFormation Template for Virtual Private Cloud (VPC)



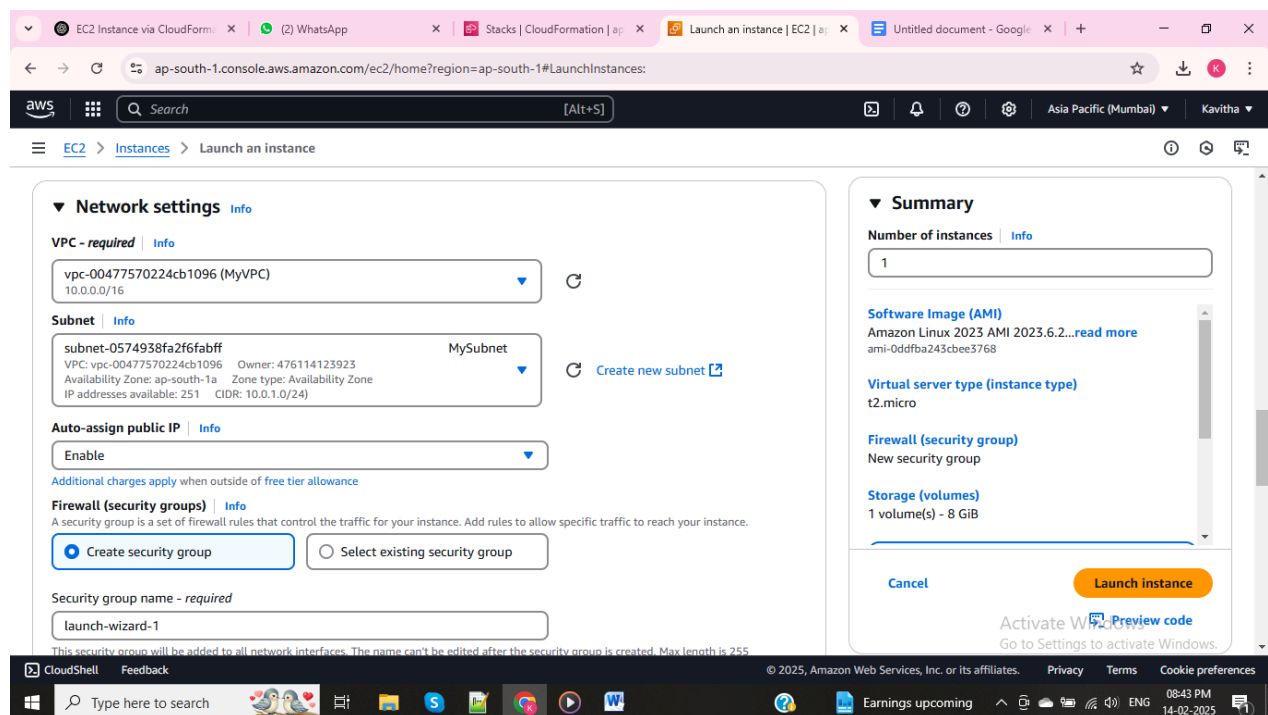
Stack created successfully



# OUTPUT



VPC created through CFT and checking the VPC is working, For this have to create EC2 Instance and attach created VPC



The screenshot displays the AWS Management Console interface for the 'ap-south-1' region. The main content area shows the 'Instances (1)' page, indicating one instance is currently running. The instance details table lists the instance name 'VPC\_CHECKING', its ID 'i-05d4c299f103a127e', and its state as 'Running'. The instance type is 't2.micro' and it has passed 2 out of 2 status checks. The left-hand navigation pane provides access to various AWS services, including EC2 Global View, Events, Instances, Images, and Elastic Block Store. The top navigation bar features the AWS logo, a search bar, and regional settings for Asia Pacific (Mumbai). The bottom of the image shows the Windows taskbar with the search bar and several application icons.

The screenshot shows a terminal window with a dark background. At the top, there's a logo for Amazon Linux 2023, which is a stylized tree-like structure. To the right of the logo, the text "Amazon Linux 2023" is displayed. Below the logo, the URL "https://aws.amazon.com/linux/amazon-linux-2023" is shown. The terminal prompt is "[ec2-user@ip-10-0-1-153 ~]\$". The user has entered the command "sudo yum update". The output of the command is "Amazon Linux 2023 Kernel Livepatch repository" followed by "131 kB/s | 14 kB" and "00:00". Below this, it says "Dependencies resolved." and "Nothing to do." and "Complete!". The terminal prompt is now "[ec2-user@ip-10-0-1-153 ~]\$".

## TEMPLATE for EC2 Instance

**AWSTemplateFormatVersion:** '2010-09-09'

**Description:** CloudFormation Template to launch an EC2 Instance

### Resources:

#### MyEC2Instance:

**Type:** AWS::EC2::Instance

#### Properties:

**InstanceType:** t2.micro

**ImageId:** ami-0ddfba243cbee3768

**KeyName:** demo

#### SecurityGroupIds:

- sg-08fd6899f3584e120

**SubnetId:** subnet-069adf10dc45d03f2

#### MySecurityGroup:

**Type:** AWS::EC2::SecurityGroup

#### Properties:

**GroupDescription:** Enable SSH and HTTP access

**VpcId:** vpc-03c4ef00c4d76e0aa

### SecurityGroupIngress:

- IpProtocol: tcp

FromPort: 22

ToPort: 22

CidrIp: 0.0.0.0/0 # Allows SSH from anywhere (restrict for production)

- IpProtocol: tcp

FromPort: 80

ToPort: 80

CidrIp: 0.0.0.0/0 # Allows HTTP traffic from anywhere

### Outputs:

InstanceId:

Description: Instance ID of the newly created EC2 instance

Value: !Ref MyEC2Instance

### VPC TEMPLATE

AWSTemplateFormatVersion: '2010-09-09'

Description: CloudFormation template to create a VPC with subnets, internet gateway, and security group.

## Resources:

### MyVPC:

Type: AWS::EC2::VPC

#### Properties:

CidrBlock: 10.0.0.0/16

EnableDnsSupport: true

EnableDnsHostnames: true

#### Tags:

- Key: Name

Value: MyVPC

### MySubnet:

Type: AWS::EC2::Subnet

#### Properties:

VpcId: !Ref MyVPC

CidrBlock: 10.0.1.0/24

AvailabilityZone: !Select [ 0, !GetAZs " ]

MapPublicIpOnLaunch: true

#### Tags:

- Key: Name

**Value:** MySubnet

**MyInternetGateway:**

**Type:** AWS::EC2::InternetGateway

**Properties:**

**Tags:**

- **Key:** Name

**Value:** MyInternetGateway

**AttachGateway:**

**Type:** AWS::EC2::VPCGatewayAttachment

**Properties:**

**VpcId:** !Ref MyVPC

**InternetGatewayId:** !Ref MyInternetGateway

**MyRouteTable:**

**Type:** AWS::EC2::RouteTable

**Properties:**

**VpcId:** !Ref MyVPC

**Tags:**



- **Key:** Name

**Value:** MyRouteTable

**MyRoute:**

**Type:** AWS::EC2::Route

**DependsOn:** AttachGateway

**Properties:**

**RouteTableId:** !Ref MyRouteTable

**DestinationCidrBlock:** 0.0.0.0/0

**GatewayId:** !Ref MyInternetGateway

**SubnetRouteTableAssociation:**

**Type:** AWS::EC2::SubnetRouteTableAssociation

**Properties:**

**SubnetId:** !Ref MySubnet

**RouteTableId:** !Ref MyRouteTable

**MySecurityGroup:**

**Type:** AWS::EC2::SecurityGroup

**Properties:**

**GroupDescription:** Enable SSH and HTTP access

**VpcId:** !Ref MyVPC

**SecurityGroupIngress:**

- **IpProtocol:** tcp

**FromPort:** 22

**ToPort:** 22

**CidrIp:** 0.0.0.0/0 # Allows SSH from anywhere

- **IpProtocol:** tcp

**FromPort:** 80

**ToPort:** 80

**CidrIp:** 0.0.0.0/0 # Allows HTTP from anywhere

**Outputs:**

**VPCId:**

**Description:** VPC ID

**Value:** !Ref MyVPC

**SubnetId:**

**Description:** Subnet ID

**Value:** !Ref MySubnet

**InternetGatewayId:**

**Description:** Internet Gateway ID

**Value:** !Ref MyInternetGateway

**SecurityGroupId:**

**Description:** Security Group ID

**Value:** !Ref MySecurityGroup