

CS lab 4



How to create AWS Services using Template. Step1: Select a Region. For Example: Mumbai

Step2: Go to Cloud Formation in AWS

Step3: Save code using yaml/JSON formatted file

AWSTemplateFormatVersion: '2010-09-09'

Description: Create a VPC, two subnets, and an EC2 instance.

Resources:

Create VPC

MyVPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: 10.0.0.0/16

EnableDnsSupport: true

EnableDnsHostnames: true

Tags: - Key: Name

Value: MyVPC

Create Internet Gateway

MyInternetGateway:

Type: AWS::EC2::InternetGateway

AttachGateway:

Type: AWS::EC2::VPCGatewayAttachment

Properties:

VpcId: !Ref MyVPC

InternetGatewayId: !Ref MyInternetGateway

Create Public Route Table

PublicRouteTable:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref MyVPC

Create Route to Internet

PublicRoute:

Type: AWS::EC2::Route

Properties:

RouteTableId: !Ref PublicRouteTable

DestinationCidrBlock: 0.0.0.0/0

GatewayId: !Ref MyInternetGateway

Subnet 1

Subnet1:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref MyVPC

CidrBlock: 10.0.1.0/24

AvailabilityZone: !Select [0, !GetAZs ""]

Subnet 2

Subnet2:

```
Type: AWS::EC2::Subnet
Properties:
VpcId: !Ref MyVPC
CidrBlock: 10.0.2.0/24
AvailabilityZone: !Select [1, !GetAZs ""]
# Associate Subnet 1 with Public Route Table
Subnet1RouteTableAssociation:
Type: AWS::EC2::SubnetRouteTableAssociation
Properties:
SubnetId: !Ref Subnet1
RouteTableId: !Ref PublicRouteTable
# Create Security Group
MySecurityGroup:
Type: AWS::EC2::SecurityGroup
Properties:
GroupDescription: Allow SSH and HTTP
VpcId: !Ref MyVPC
SecurityGroupIngress: - IpProtocol: tcp
FromPort: 22
ToPort: 22
CidrIp: 0.0.0.0/0 - IpProtocol: tcp
FromPort: 80
ToPort: 80
CidrIp: 0.0.0.0/0
# Launch EC2 Instance in Subnet 1
MyEC2Instance:
Type: AWS::EC2::Instance
Properties:
InstanceType: t2.micro
ImageId: ami-0e35ddab05955cf57 # Replace with a valid AMI ID for your region
SubnetId: !Ref Subnet1
SecurityGroupIds:
- !Ref MySecurityGroup
KeyName: my-key-pair2 # Replace with your key pair name
Step4: Create Stack.
Step5: Upload a template file, Upload your template directly to the console.(code should be in JSON
or YAML
formatted file).
Step6: Give stack name and submit (create it)
```

Here's a **clear, step-by-step guide** to create AWS services using **AWS CloudFormation templates** (YAML/JSON):

Step-by-Step: Create AWS Resources using CloudFormation Template

Step 1: Select a Region

Choose the AWS region where you want to deploy the resources.


 **Example:** `Asia Pacific (Mumbai)` → Region code: `ap-south-1`

Step 2: Go to AWS CloudFormation

1. Sign in to [AWS Management Console](#).
 2. In the **Search bar**, type **CloudFormation** and click the service.
 3. Choose "**Create Stack**" → **With new resources (standard)**.
-

Step 3: Prepare and Save Template File

Save the below **YAML** code in a file.

 Example file name: `my-vpc-setup.yaml`

yaml

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Create a VPC, two subnets, and an EC2 instance.
```

Resources:

```
# Create VPC
```

```
MyVPC:
```

```
  Type: AWS::EC2::VPC
```

```
  Properties:
```

```
    CidrBlock: 10.0.0.0/16
```

```
    EnableDnsSupport: true
```

```
    EnableDnsHostnames: true
```

```
  Tags:
```

```
    - Key: Name
```

```
      Value: MyVPC
```

```
# Create Internet Gateway
```

```
MyInternetGateway:
```

```
  Type: AWS::EC2::InternetGateway
```

```
AttachGateway:
```

```
  Type: AWS::EC2::VPCGatewayAttachment
```

```
  Properties:
```

```
    VpcId: !Ref MyVPC
```

```
    InternetGatewayId: !Ref MyInternetGateway
```

```
# Create Route Table
```

```
PublicRouteTable:
```

```
  Type: AWS::EC2::RouteTable
```

```
  Properties:
```

```
    VpcId: !Ref MyVPC
```

```
# Route to Internet
```

```
PublicRoute:
```

```
  Type: AWS::EC2::Route
```

```
  Properties:
```

```
    RouteTableId: !Ref PublicRouteTable
```

```
    DestinationCidrBlock: 0.0.0.0/0
```

```
    GatewayId: !Ref MyInternetGateway
```

```
# Subnet 1
```

```
Subnet1:
```

```
  Type: AWS::EC2::Subnet
```

```
  Properties:
```

```
    VpcId: !Ref MyVPC
```

```
    CidrBlock: 10.0.1.0/24
```

```
    AvailabilityZone: !Select [0, !GetAZs '']
```

```
# Subnet 2
Subnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref MyVPC
    CidrBlock: 10.0.2.0/24
    AvailabilityZone: !Select [1, !GetAZs '']

# Associate Subnet1 with Route Table
Subnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref Subnet1
    RouteTableId: !Ref PublicRouteTable

# Create Security Group
MySecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow SSH and HTTP
    VpcId: !Ref MyVPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0

# Launch EC2 Instance
MyEC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t2.micro
    ImageId: ami-0e35ddab05955cf57 # ✅ Replace with a valid AMI ID for Mumbai
    SubnetId: !Ref Subnet1
    SecurityGroupIds:
      - !Ref MySecurityGroup
    KeyName: my-key-pair2 # ✅ Replace with your actual key pair name
```

Step 4: Create Stack

1. In **CloudFormation**, click **"Create stack → With new resources"**.
2. Under **Specify template**, choose **"Upload a template file"**.
3. Click **Choose file**, select your ``my-vpc-setup.yaml``.

Step 5: Stack Details

- **Stack name:** ``MyVPCStack`` (or any name you prefer)
- Click **Next** and optionally add tags.
- Click **Next** again.

Step 6: Review & Create

- Review your settings.
- Acknowledge IAM resource creation if needed.

- Click **Create stack**.

✓ Result

After a few minutes, your VPC, subnets, route table, internet gateway, security group, and EC2 instance will be created in the **Mumbai region**.

Let me know if you want this same template in **JSON format** or with additional services like S3, RDS, Lambda, etc.



JSON format

Here's the **JSON version** of the AWS CloudFormation template that:

- Creates a VPC
- Adds two subnets
- Attaches an Internet Gateway
- Creates a public route
- Launches an EC2 instance

 **Save as:** `my-vpc-setup.json`

```
json

{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Create a VPC, two subnets, and an EC2 instance.",
  "Resources": {
    "MyVPC": {
      "Type": "AWS::EC2::VPC",
      "Properties": {
        "CidrBlock": "10.0.0.0/16",
        "EnableDnsSupport": true,
        "EnableDnsHostnames": true,
        "Tags": [
          {
            "Key": "Name",
            "Value": "MyVPC"
          }
        ]
      }
    },
    "MyInternetGateway": {
      "Type": "AWS::EC2::InternetGateway"
    },
    "AttachGateway": {
      "Type": "AWS::EC2::VPCGatewayAttachment",
      "Properties": {
        "VpcId": {
          "Ref": "MyVPC"
        },
        "InternetGatewayId": {
          "Ref": "MyInternetGateway"
        }
      }
    }
  }
}
```

```

    },
    "PublicRouteTable": {
      "Type": "AWS::EC2::RouteTable",
      "Properties": {
        "VpcId": {
          "Ref": "MyVPC"
        }
      }
    },
    "PublicRoute": {
      "Type": "AWS::EC2::Route",
      "Properties": {
        "RouteTableId": {
          "Ref": "PublicRouteTable"
        },
        "DestinationCidrBlock": "0.0.0.0/0",
        "GatewayId": {
          "Ref": "MyInternetGateway"
        }
      }
    },
    "Subnet1": {
      "Type": "AWS::EC2::Subnet",
      "Properties": {
        "VpcId": {
          "Ref": "MyVPC"
        },
        "CidrBlock": "10.0.1.0/24",
        "AvailabilityZone": {
          "Fn::Select": [
            0,
            {
              "Fn::GetAZs": ""
            }
          ]
        }
      }
    },
    "Subnet2": {
      "Type": "AWS::EC2::Subnet",
      "Properties": {
        "VpcId": {
          "Ref": "MyVPC"
        },
        "CidrBlock": "10.0.2.0/24",
        "AvailabilityZone": {
          "Fn::Select": [
            1,
            {
              "Fn::GetAZs": ""
            }
          ]
        }
      }
    },
    "Subnet1RouteTableAssociation": {
      "Type": "AWS::EC2::SubnetRouteTableAssociation",
      "Properties": {
        "SubnetId": {
          "Ref": "Subnet1"
        },
        "RouteTableId": {
          "Ref": "PublicRouteTable"
        }
      }
    },
    "MySecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupDescription": "Allow SSH and HTTP",
        "VpcId": {
          "Ref": "MyVPC"
        },
        "SecurityGroupIngress": [

```

```

    {
      "IpProtocol": "tcp",
      "FromPort": 22,
      "ToPort": 22,
      "CidrIp": "0.0.0.0/0"
    },
    {
      "IpProtocol": "tcp",
      "FromPort": 80,
      "ToPort": 80,
      "CidrIp": "0.0.0.0/0"
    }
  ]
},
{
  "MyEC2Instance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
      "InstanceType": "t2.micro",
      "ImageId": "ami-0e35ddab05955cf57",
      "SubnetId": {
        "Ref": "Subnet1"
      },
      "SecurityGroupIds": [
        {
          "Ref": "MySecurityGroup"
        }
      ],
      "KeyName": "my-key-pair2"
    }
  }
}
}
}

```

⚠️ Replace the following before deployment:

- `"ImageId": "ami-0e35ddab05955cf57"`
 - Replace with a **valid AMI ID** for **Mumbai** (`ap-south-1`) region. Use Amazon Linux 2 or Ubuntu.
- `"KeyName": "my-key-pair2"`
 - Replace with your **actual EC2 key pair name** created in AWS.

Would you like me to help you find the correct AMI ID for Mumbai or guide you on creating a key pair?