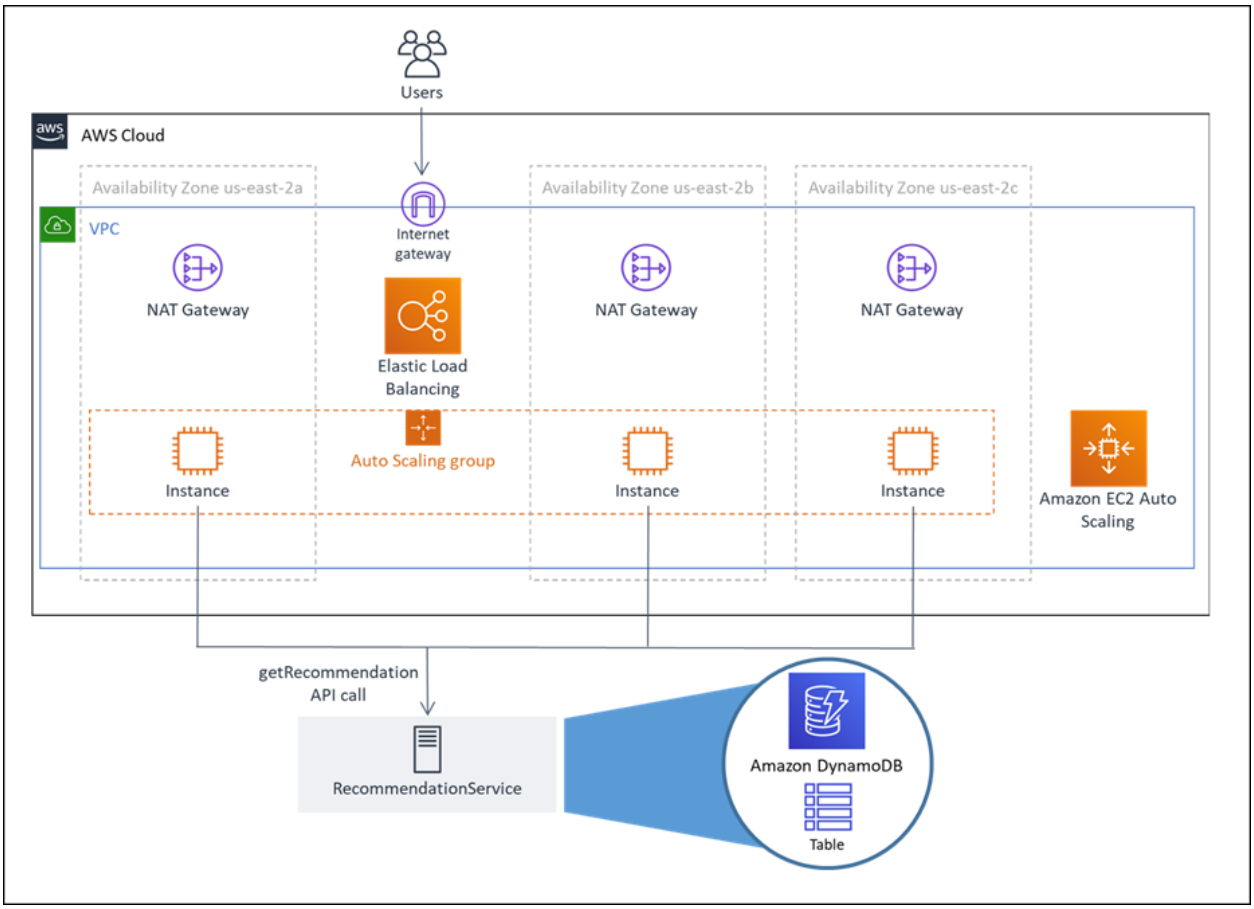
Deploy a Reliable Multi-tier Infrastructure using CloudFormation

Introduction

This hands-on lab will guide you through the steps to improve reliability of a service by using automation to deploy a reliable cloud infrastructure. When this lab is completed, you will have deployed two CloudFormation templates. The first will deploy an Amazon Virtual Private Cloud (VPC). The second will deploy into your VPC, a reliable 3-tier infrastructure using Amazon EC2 distributed across three Availability Zones. You will then review the features of the deployed infrastructure and learn how they contribute to reliability.

The architecture of the infrastructure you will deploy is represented by this diagram:



Goals

By the end of this lab, you will be able to:

- Automate infrastructure deployment for a workload
- Understand how the deployed workload infrastructure contributes to reliability of the workload

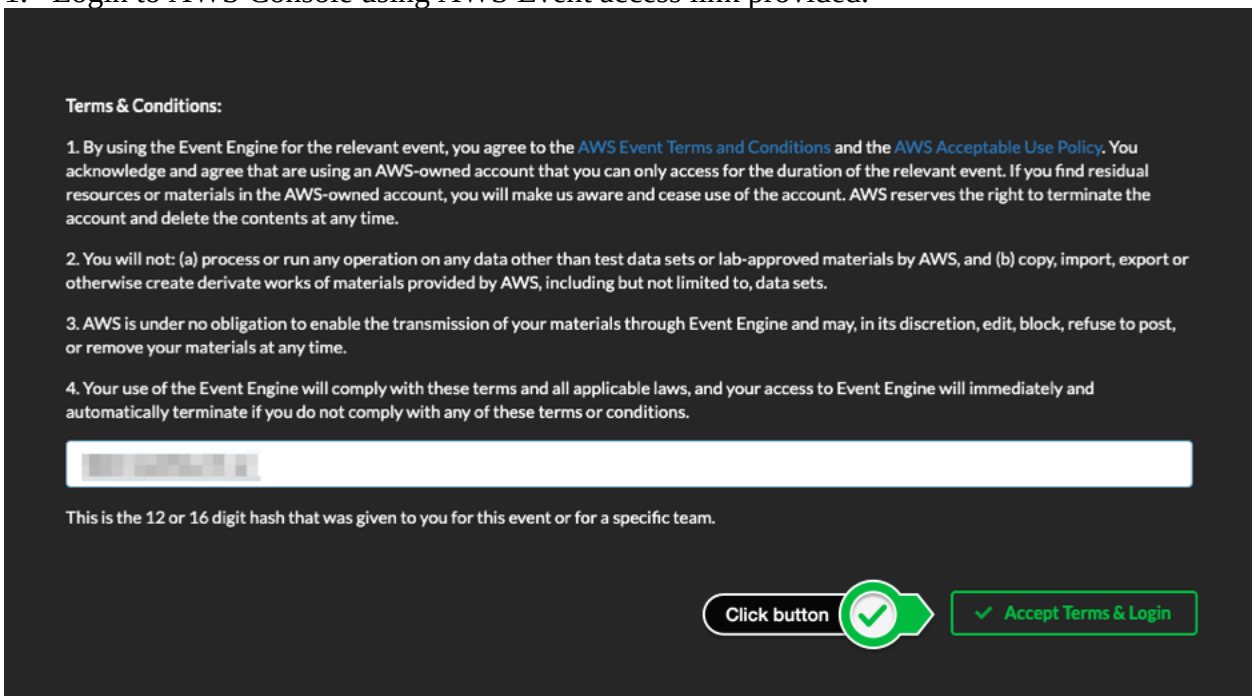
Steps

Deploy VPC using CloudFormation
Examine VPC resources created using CloudFormation
Deploy Web Application and Infrastructure using CloudFormation
Explore the Web Application
Explore the CloudFormation Template
Tear down this lab

Deploy VPC using CloudFormation

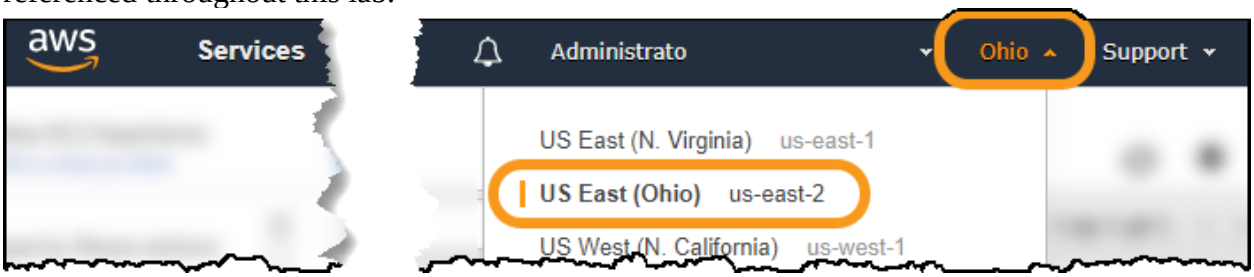
1.1 Log into the AWS console

1. Login to AWS Console using AWS Event access link provided.



1.2 Configure your AWS Region

1. Select the **Ohio** region. This region is also known as **us-east-2**, which you will see referenced throughout this lab.

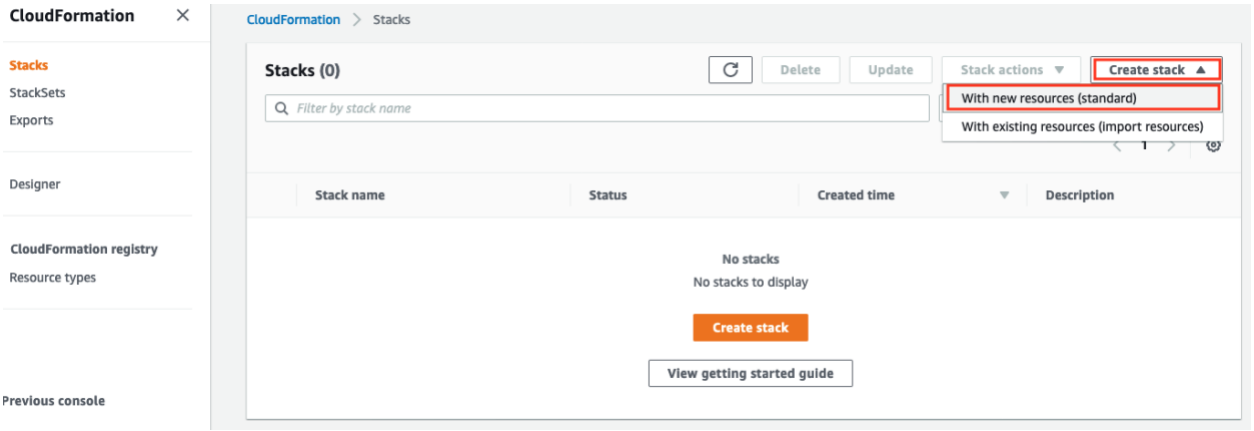


   - AWS offers you the ability to deploy to over 20 regions located across the globe
   - Each region is fully isolated from the others to isolate any issues and achieve high availability,
   - Each region is comprised of multiple Availability Zones, which are fully isolated partitions of our infrastructure (more on this later)

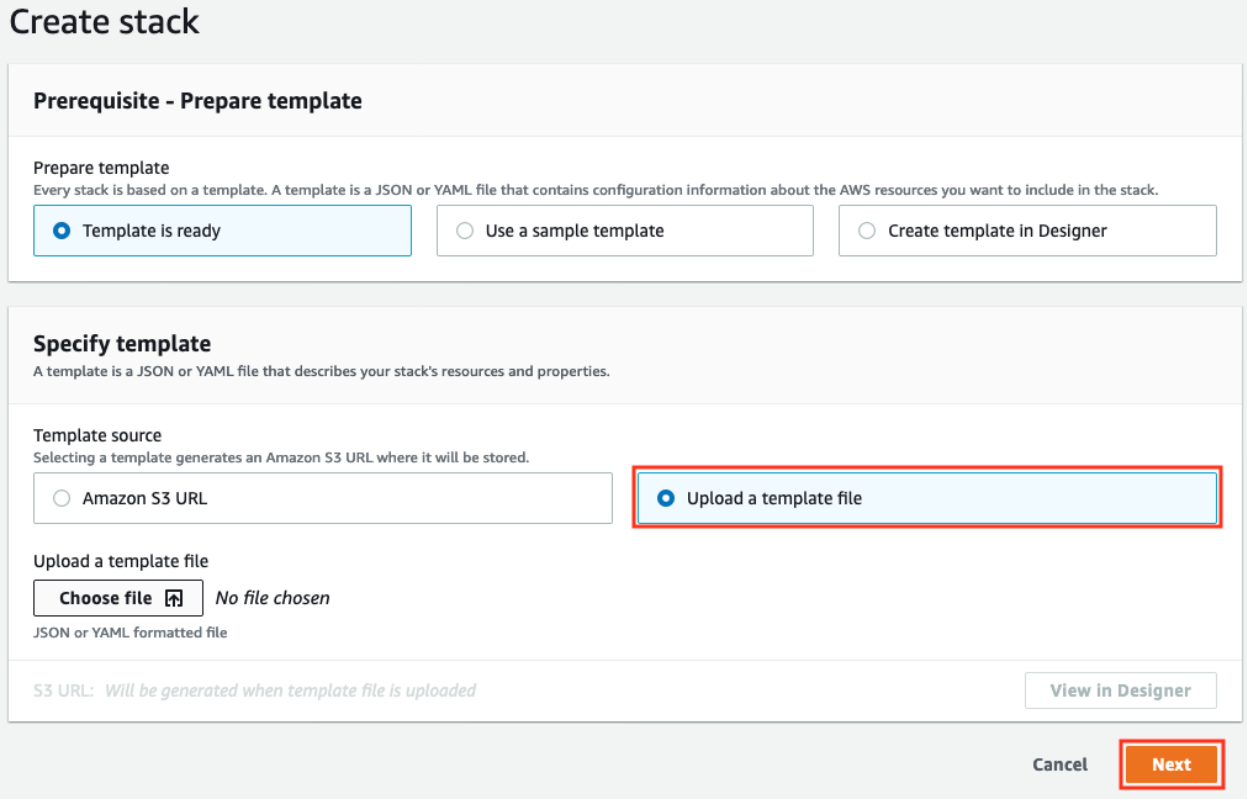
1.3 Deploy the VPC infrastructure

This step will create the VPC and all components using the example CloudFormation template.

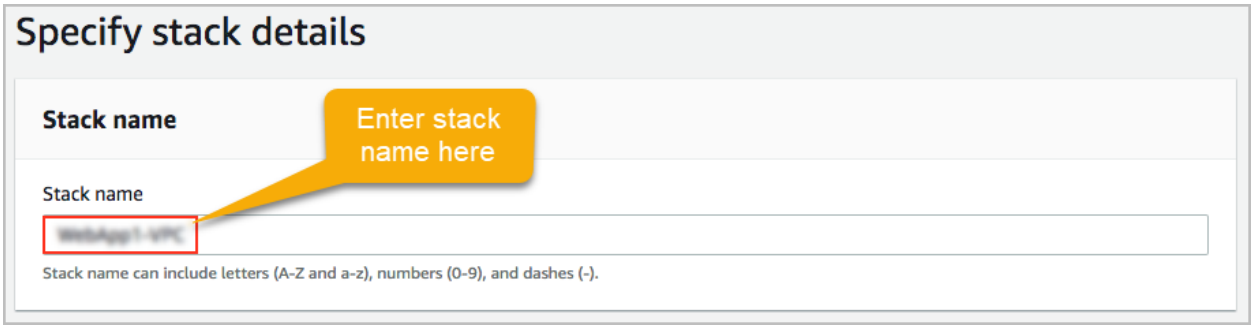
1. Download the latest version of the CloudFormation template here: vpc-alb-app-db.yaml
2. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation/
3. Click **Create Stack**, then **With new resources (standard)**.



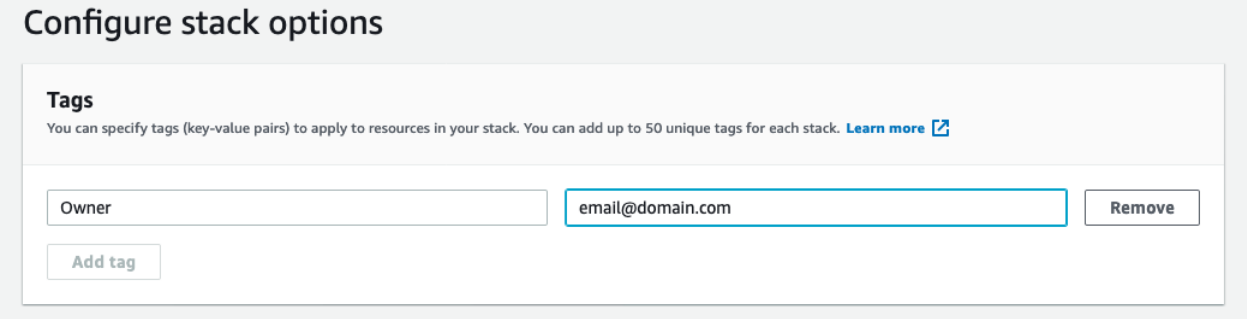
4. Click **Upload a template file** and then click **Choose file**.



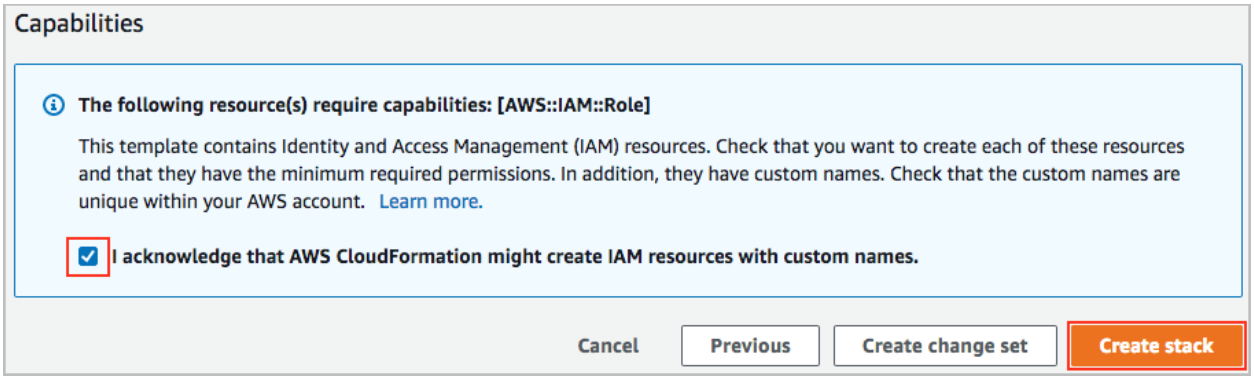
5. Choose the CloudFormation template you downloaded in step 1, return to the CloudFormation console page and click **Next**.
6. Enter the following details:
    - **Stack name**: The name of this stack. For this lab, use **WebApp1-VPC** and match the case.
    - **Parameters**: Parameters may be left as defaults; you can find out more in the description for each.



7. At the bottom of the page click **Next**.
8. In this lab, we use tags, which are key-value pairs, that can help you identify your stacks. Enter *Owner* in the left column which is the key, and your email address in the right column which is the value. We will not use additional permissions or advanced options so click **Next**. For more information, see [Setting AWS CloudFormation Stack Options]



9. Review the information for the stack. When you're satisfied with the configuration, at the bottom of the page check **I acknowledge that AWS CloudFormation might create IAM resources with custom names** then click **Create stack**.

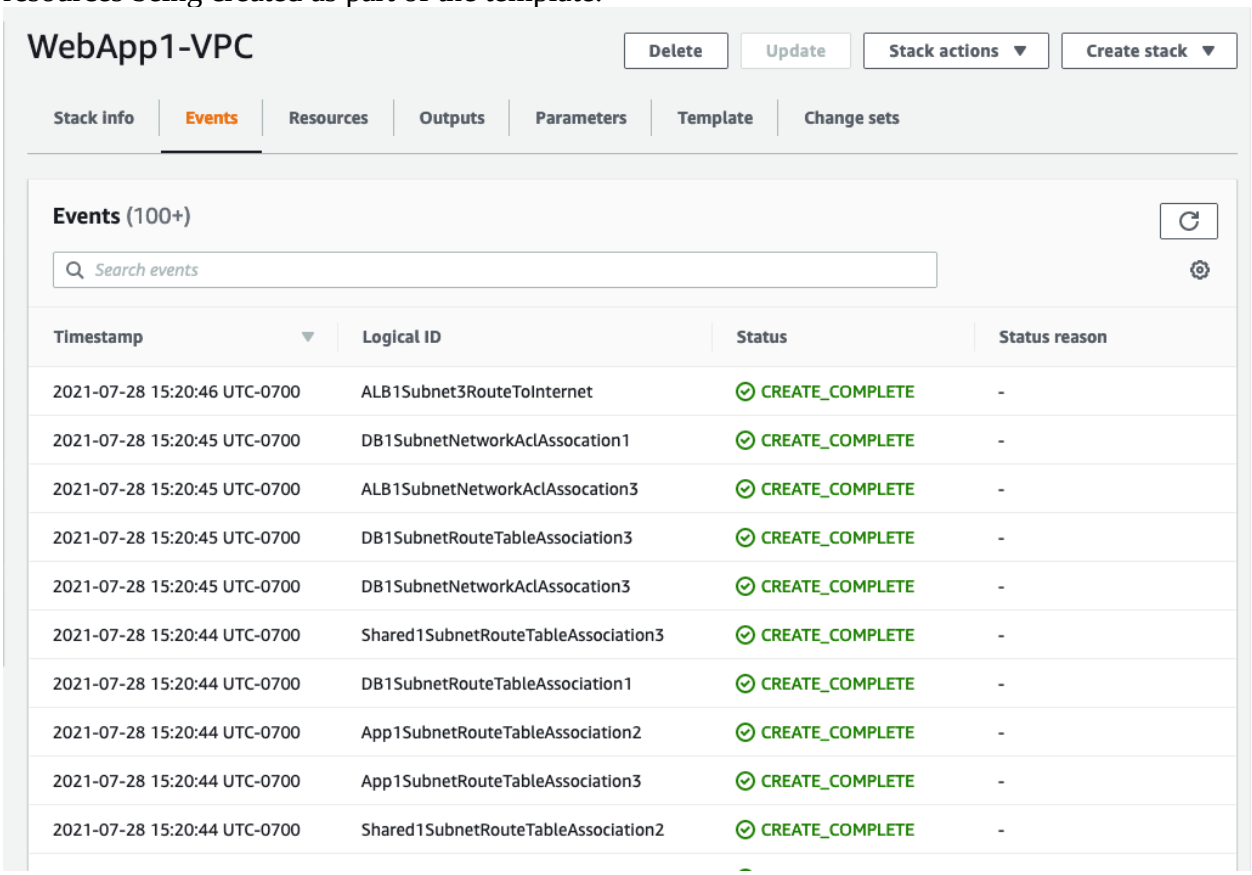


10. Click on **Create stack** and you will notice your stack creation started with status **CREATE_IN_PROGRESS.** CloudFormation begins creating the resources that are specified in the template.

1.4 Monitor CloudFormation Stack Creation

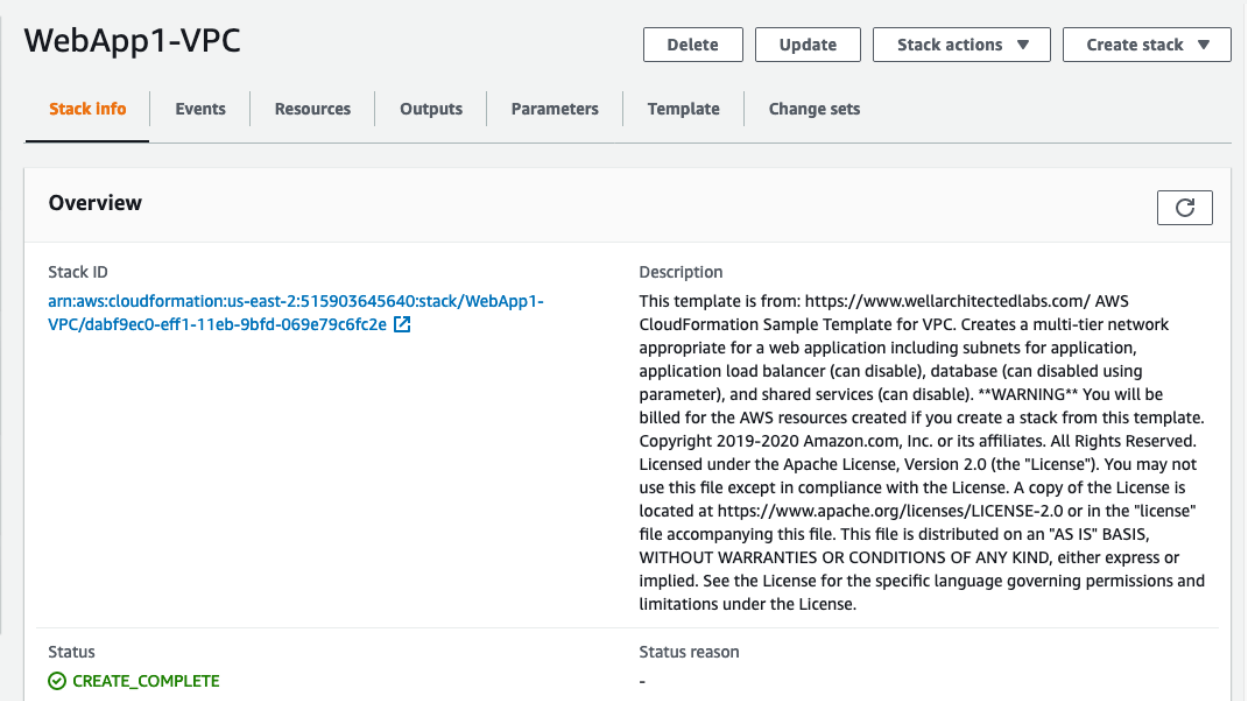
1. Explore the **Events** tab where you can see progress as your stack get created. Click the refresh icon to the right to view updates. The Events tab shows detailed progress of all

resources being created as part of the template.

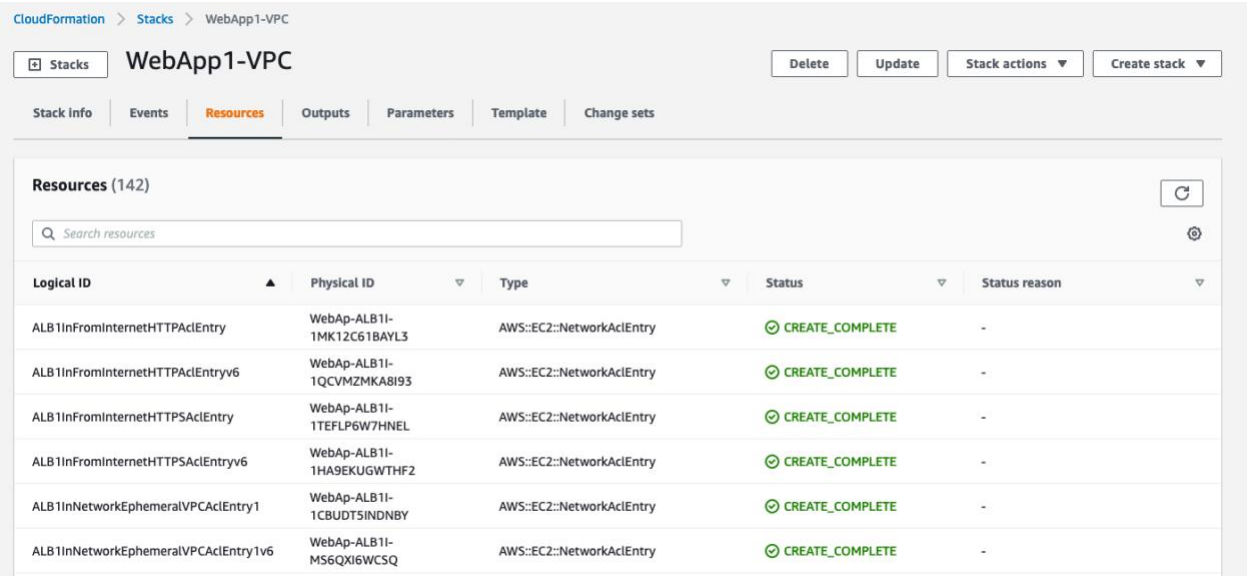


2. While you are waiting, explore all other tabs like the **Template** tab to review your template and the **Parameters** tab to see parameter values. You will examine the Template in more detail later in the lab.

3. Click on Stack Info tab to view the Overall status for the stack. Once stack status changes to **CREATE_COMPLETE,** you can proceed to next step.



4. Once stack creation is complete, visit the **Resources** tab to see all the resources that were created by this AWS CloudFormation template.

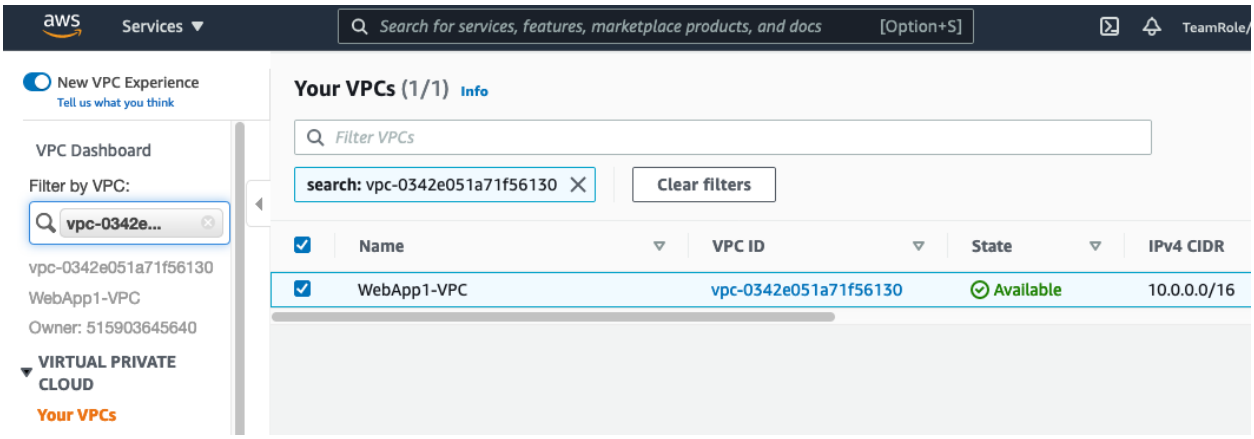


Examine VPC resources created using CloudFormation

In this task, you will examine a few of the VPC resources that were created together with the code from the CloudFormation template that created the resources.

1. In the AWS Console search bar, Search for and Select **VPC**

In **Filter by VPC** in the top-left corner, select the VPC with Name tag: **WebApp1-VPC.** Click on **Your VPCs.**



Here is the code from the CloudFormation template that created this VPC:

```
Resources:
  VPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      CidrBlock: !Ref VPCIPv4CidrBlock
      EnableDnsSupport: true
      EnableDnsHostnames: true
      InstanceTenancy: default
      Tags:
        - Key: Name
          Value: !Join
            - '-'
            - - !Ref NamingPrefix
              - VPC
```

The **Type** parameter in the above code declares the type of resources being created by CloudFormation.

The **Properties** section specifies more information about the resource to create. In this case, it defines:

- **CidrBlock:** The IP address range associated with the VPC.
- **EnableDnsSupport:** Configures the VPC to use Route 53 to resolve DNS hostnames
- **EnableDnsHostnames:** Configures the VPC to associate DNS names with Amazon EC2 instances.
- **Tags:** Adds a friendly name to the resource.

The **CidrBlock** property is defined by using the intrinsic function `Ref`, which in this example returns the value of the *parameter* `<VPCIPv4CidrBlock>` defined in the Parameters section of the CloudFormation template

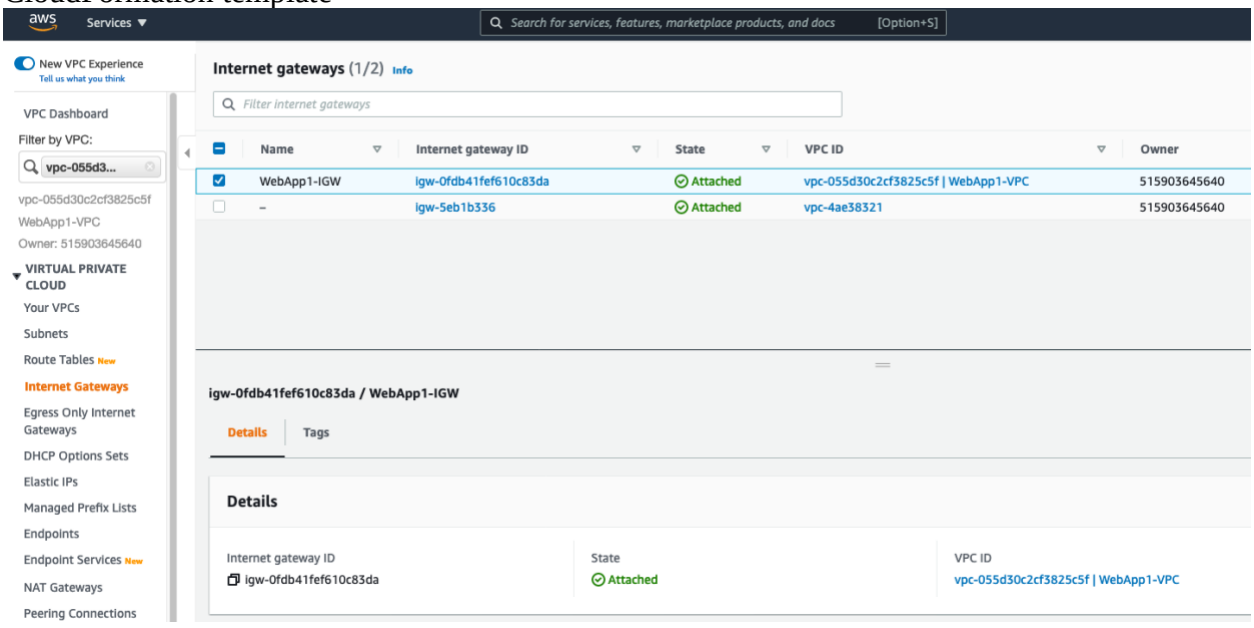
```
Parameters:
<..>
  VPCIPv4CidrBlock:
    Description: VPC CIDR block for IPv4. Default of 10.0.0.0/16 is
recommended for testing.
    Type: String
    Default: 10.0.0.0/16
<..>
```

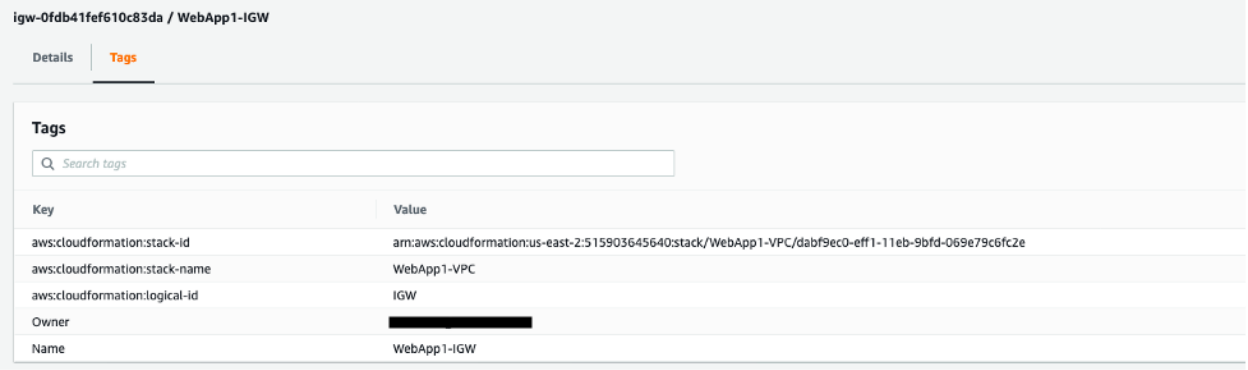
The **Tag** Parameter assigns a Name Tag to the VPC, which uses the `!Join` function to assign a name to the VPC: **WebApp1-VPC**

2. In the left navigation pane for the VPC Dashboard, click **Internet Gateways**. The Internet Gateway created using CloudFormation and all other created resources use the same naming convention. <WebApp1>

Click the checkbox to the left of the WebApp1-IGW Name and navigate to the **Tags** Tab at the bottom.

Here you will find additional Tags that were applied to identify resources created using the CloudFormation template





Below is the code from the CloudFormation template that created this IGW and attached it to **WebApp1-VPC**:

```
IGW:
    Type: 'AWS::EC2::InternetGateway'
    Properties:
      Tags:
        - Key: Name
          Value: !Join
            - '-'
            - - !Ref NamingPrefix
              - IGW
IGWAttach:
    Type: 'AWS::EC2::VPCGatewayAttachment'
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref IGW
```

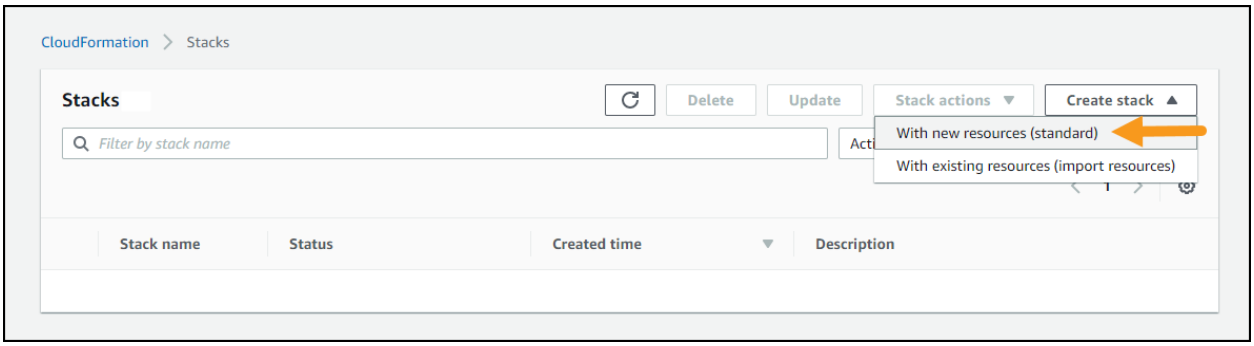
The `IGWAttach` code snippet also uses the `!Ref` function. In this example, the `!Ref` function is used to reference other resources, the VPC and IGW, that were created within the template.

Deploy Web Application and Infrastructure using CloudFormation

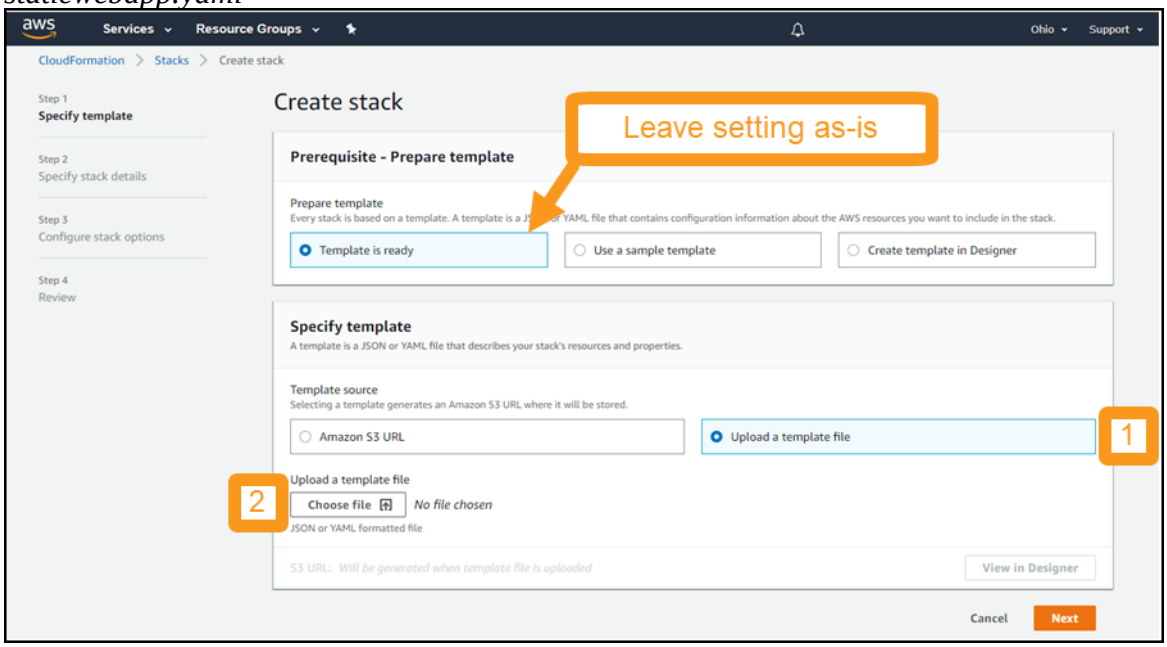
Download the CloudFormation template that will be used to deploy a Web Application in the VPC that was created in the first lab: *staticwebapp.yaml*

1. Go to the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation and click **Create Stack** > **With new**

resources

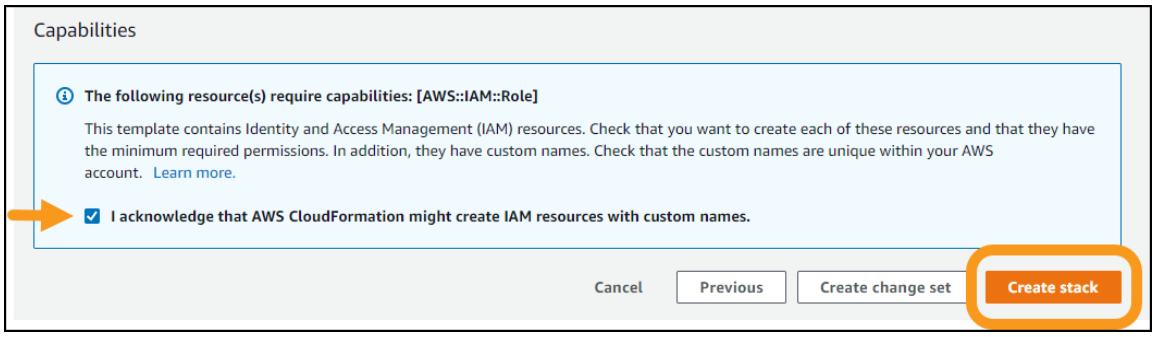


2. Leave **Prepare template** setting as-is
    - For **Template source** select **Upload a template file**
    - Click **Choose file** and supply the CloudFormation template you downloaded: *staticwebapp.yaml*

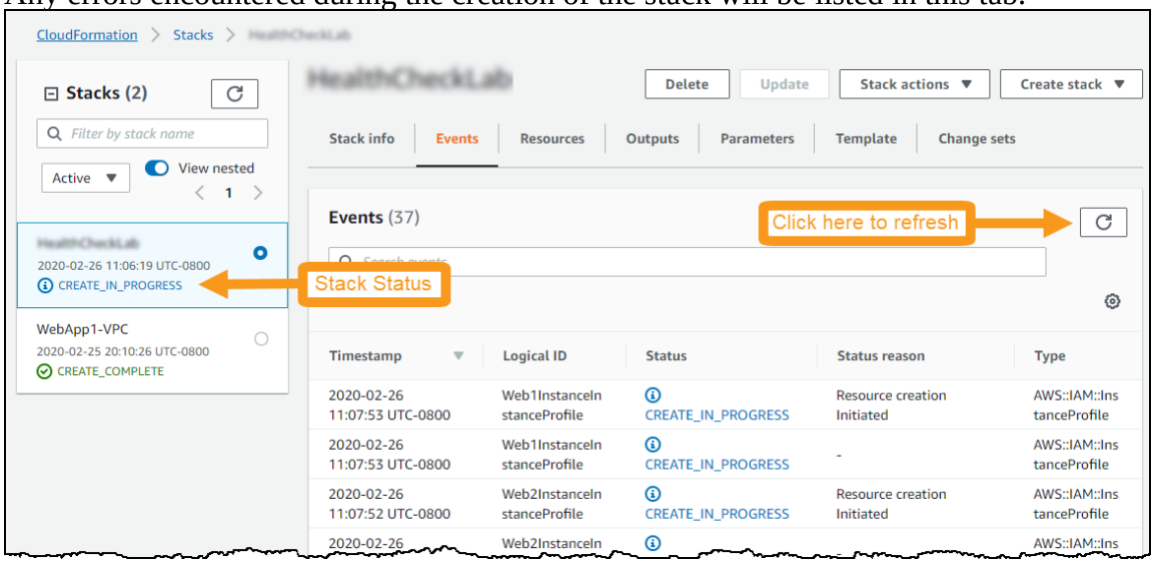


3. Click **Next**
4. For **Stack name** use **CloudFormationLab**
5. **Parameters**
    - Look over the Parameters and their default values.
    - Click **Next**
6. For **Configure stack options** we recommend configuring tags, which are key-value pairs, that can help you identify your stacks and the resources they create. For example, enter *Owner* in the left column which is the key, and your email address in the right column which is the value. We will not use additional permissions or advanced options so click **Next**
7. For **Review**
    - Review the contents of the page
    - At the bottom of the page, select **I acknowledge that AWS CloudFormation might create IAM resources with custom names**

- Click **Create stack**



8. This will take you to the CloudFormation stack status page, showing the stack creation in progress.
   - Click on the **Events** tab
   - Scroll through the listing. It shows the activities performed by CloudFormation (newest events at top), such as starting to create a resource and then completing the resource creation.
   - Any errors encountered during the creation of the stack will be listed in this tab.



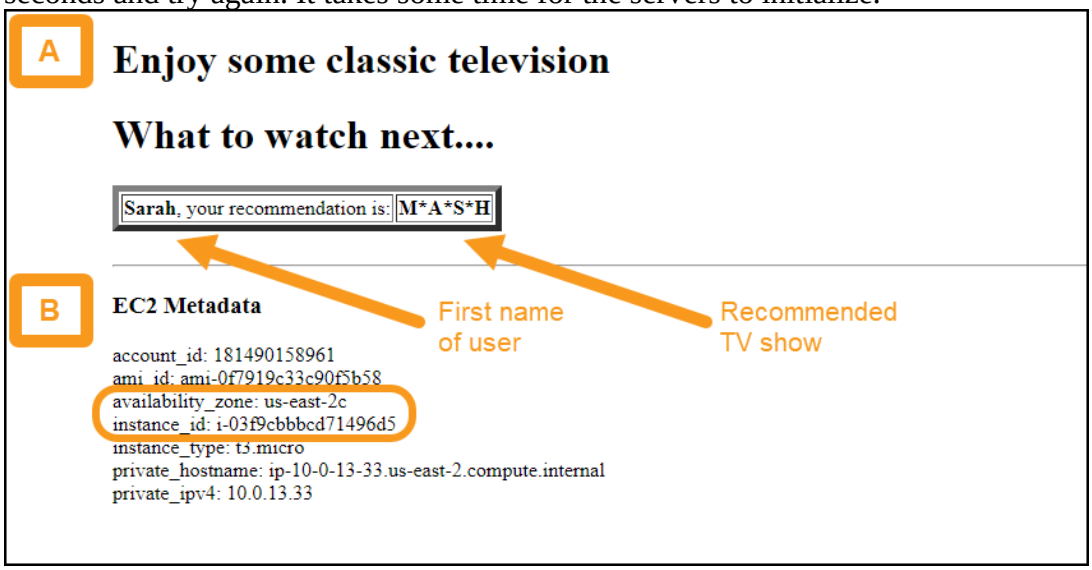
9. When it shows **status** *CREATE_COMPLETE*, then you are finished with this step. This should take about four minutes.

Explore the Web Application

1. Go to the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
   - Click on the **CloudFormationLab** stack
   - Click on the **Outputs** tab
   - For the Key **WebsiteURL** copy the value. This is the URL of your test web service

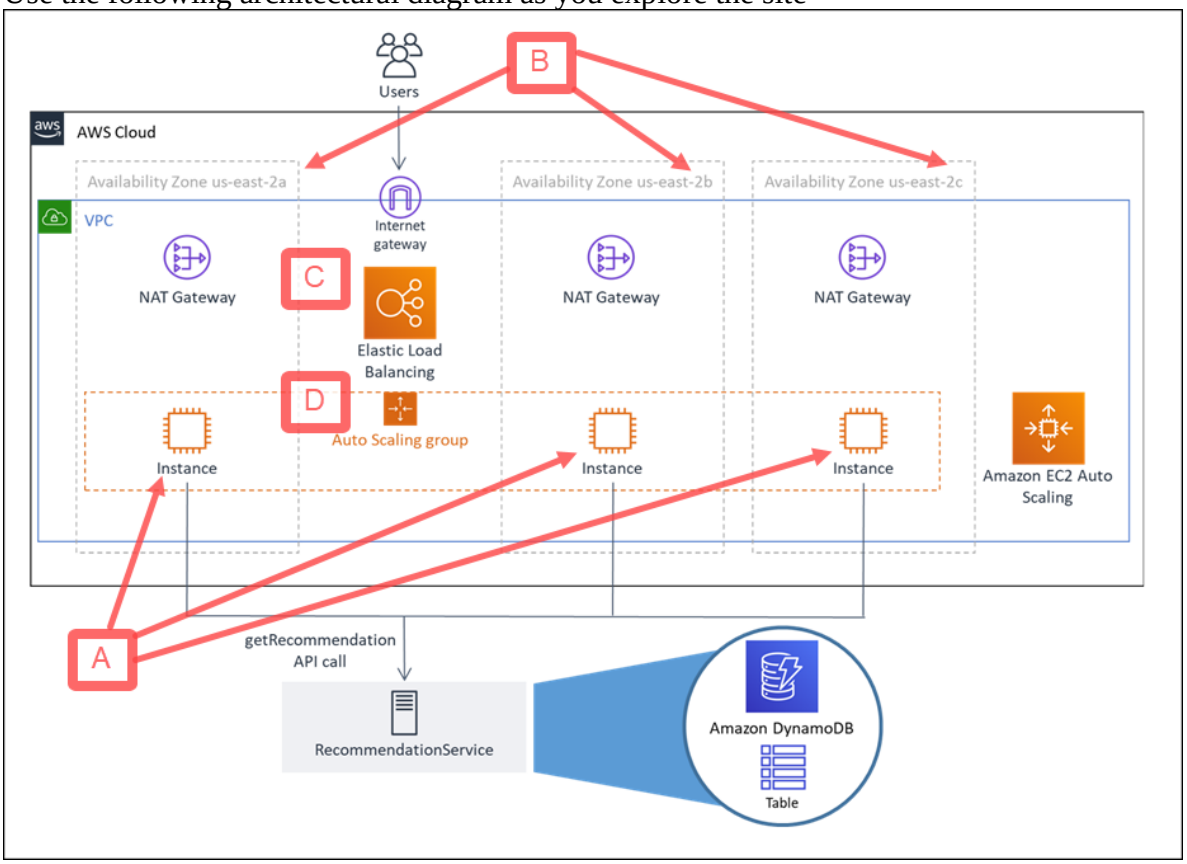
- *Hint*: it will start with `http://healt-alb` and end in `<aws region>.elb.amazonaws.com`

2. Click the URL and it will bring up the website:
   - *Troubleshooting*: if you see an error such as *502 Bad Gateway*, then wait 60 seconds and try again. It takes some time for the servers to initialize.



3. The website simulates a recommendation engine making personalized suggestions for classic television shows. You should note the following features:
   - Area A shows the personalized recommendation
     - It shows first name of the user and the show that was recommended
     - The workshop simulation is simple. On every request it chooses a user at random, and shows a recommendation statically mapped to that user. The user names, television show names, and this mapping are in a DynamoDB table, which is simulating the **RecommendationService**
   - Area B shows metadata which is useful to you during the lab
     - The **instance_id** and **availability_zone** enable you to see which EC2 server and Availability Zone were used for each request

4. Use the following architectural diagram as you explore the site



- **A** - There is one EC2 instance deployed per Availability Zone
- **B** - Refresh the website several times, note that the EC2 instance and Availability Zone change from among the three available
- **C** - Elastic Load Balancing (ELB) is used here. An Application Load Balancer receives each request and distributes it among the available EC2 server instances across Availability Zones.
  - The requests are stateless, and therefore can be routed to any of the available EC2 instances
- **D** - The EC2 instances are in an Amazon EC2 Auto Scaling Group . This Auto Scaling Group was configured to maintain three instances, therefore if one instance is detected as *unhealthy* it will be replaced to maintain three *healthy* instances.
  - AWS Auto Scaling can also be configured to scale up/down dynamically in response to workload conditions such as CPU utilization or request count.

The Architecture deployed with the provided CloudFormation templates follows AWS Well-Architected Best Practices for Reliability

Use highly available network connectivity for your workload public endpoints:
You used Elastic Load Balancing which provides load balancing across Availability Zones, performs Layer 4 (TCP) or Layer 7 (http/https) routing, integrates with AWS WAF, and integrates with AWS Auto Scaling to help create a self-healing infrastructure and absorb increases in traffic while releasing resources when traffic decreases.

Implement loosely coupled dependencies:
Dependencies such as load balancers are loosely coupled. Loose coupling helps isolate behavior of a component from other components that depend on it, increasing resiliency and agility.

Deploy the workload to multiple locations:
Distribute workload data and resources across multiple Availability Zones or, where necessary, across AWS Regions

Automate healing on all layers:
Upon detection of a failure, use automated capabilities to perform actions to remediate it

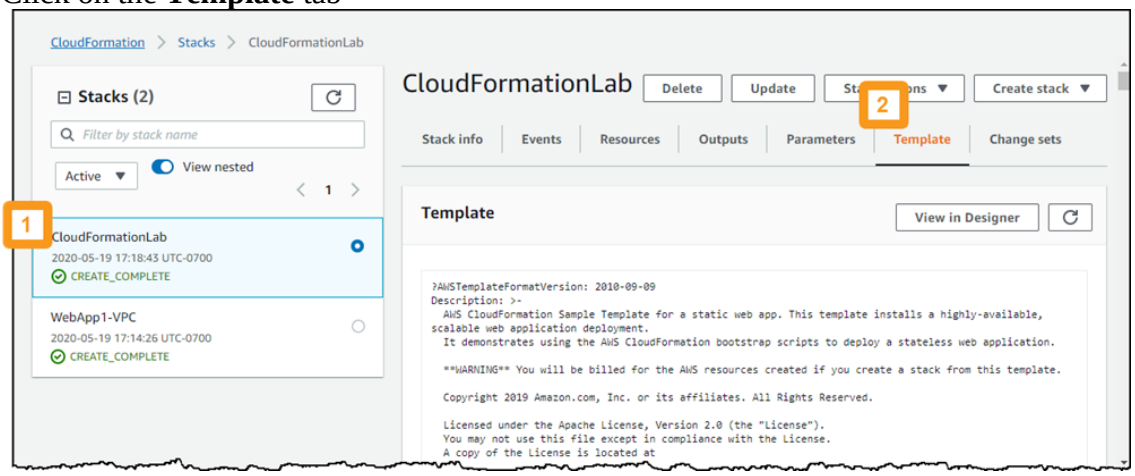
Additional best practices can be found in our AWS Well-Architected Framework.

Explore the CloudFormation Template used to create the WebApp

In this section you will explore the CloudFormation template and learn how you were able to deploy the web application infrastructure using it

- Go to the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation
    1. Click on the **CloudFormation** stack that you deployed

2. Click on the **Template** tab



- *Alternate*: You previously downloaded the CloudFormation Template *staticwebapp.yaml*. You can also view it in the text editor of your choice

The template is written in a format called YAML , which is commonly used for configuration files. CloudFormation templates can also be written in JSON.

Look through the template. You will notice several sections:

- The Parameters section is used to prompt for inputs that can be used elsewhere in the template. The template is asking for several inputs, but also provides default values for each one.
    - Look through these and start to reason about what some are used for.
    - For example, `InstanceType` is a parameter where the user can choose that Amazon EC2 instance type to deploy for the servers used in this Web App.
- The Conditions section is where you can setup *if/then*-like control of what happens during template deployment. It defines the circumstances under which entities are created or configured.
- The Resources section is the "heart" of the template. It is where you define the infrastructure to be deployed. Look at the *first* resource defined.
    - It is the Amazon DynamoDB table used as the mock for the **RecommendationService**
    - It has a *logical ID* which in this case is `DynamoDBServiceMockTable`. This logical ID is how we refer to the DynamoDB table resource within the CloudFormation template.
    - It has a `Type` which tells CloudFormation which type of resource to create. In this case a `AWS::DynamoDB::Table`
    - And it has `Properties` that define the values used to create the VPC
- The Outputs section is used to display selective information about resources in the stack.
    - In this case it uses the built-in function `!GetAtt` to get the DNS Name for the Application Load Balancer.
    - This URL is what you used to access the WebApp
- The Metadata section here is used to group and order how the CloudFormation parameters are displayed when you deploy the template using the AWS Console

Tear down this lab

If you are attending an in-person workshop and were provided with an AWS account by the instructor:

- There is no need to tear down the lab. Feel free to continue exploring. Log out of your AWS account when done.

If you are using your own AWS account:

- You may leave these resources deployed for as long as you want. When you are ready to delete these resources, see the following instructions

How to delete an AWS CloudFormation stack

1. Go to the AWS CloudFormation console: https://console.aws.amazon.com/cloudformation
2. Select the **CloudFormationLab** stack and click **Delete**
3. Wait for the **CloudFormationLab** CloudFormation stack to complete (it will no longer be shown on the list of actice stacks)
    - The **Status** changes to *DELETE_IN_PROGRESS*
    - Click the refresh button to update and status will ultimately progress to *DELETE_COMPLETE*
4. Select the **WebApp1-VPC** stack and click **Delete**

References & useful resources

AWS CloudFormation

- What is AWS CloudFormation?
- CloudFormation AWS Resource and Property Types Reference

AWS Resources that enable reliable architectures:

- What Is Amazon EC2 Auto Scaling?
- Elastic Load Balancing: What Is an Application Load Balancer?
- Availability Zones: AWS Global Infrastructure