# Deploying a Web Server with CloudFormation Designer

01:08:23 Exit Lab Complete Lab

1 hour 15 minutes duration Practitioner Rate this lab

Videos Guide

# Deploying a Web Server with CloudFormation Designer

## Introduction

AWS CloudFormation Designer (Designer) is a graphic tool for creating, viewing, and modifying AWS CloudFormation templates. In this hands-on lab, we will use the drag-and-drop capabilities of CloudFormation Designer to create a full web architecture.

## Solution

Log in with the credentials provided, and make sure you are in the `us-east-1` (N. Virginia) region.

You can download the solution CloudFormation template on GitHub.

## Create an AWS CloudFormation Template and Stack with CloudFormation Designer

### Use Drag-and-Drop Interface to Begin Template

1. Navigate to CloudFormation.
2. Click **Create stack**.

3. In the *Prerequisite - Prepare template* section, select **Create template in Designer**.
4. Click **Create template in designer**.
5. Select the **YAML** button.
6. Click the pencil icon next to the template name at the bottom to rename it "BasicWebServerInVPC.template", and then click the checkbox.
7. In the *Resource types* pane, click to expand **EC2**.
8. Click **VPC** and drag it over into the window next to the *Resource types* pane.
9. Expand the VPC box to take up most of the window.
10. In the *Properties* pane, click the pencil icon to rename it "VPC".
11. Refresh the Designer.
12. In the *Resource types* pane, drag **Subnet** over into the VPC window.
13. In the *Properties* pane, click the pencil icon to rename it "PublicSubnet".
14. In the *Resource types* pane, drag **Instance** over into the PublicSubnet window.
15. In the *Properties* pane, click the pencil icon to rename it "WebServerInstance".
16. In the *Resource types* pane, drag **SecurityGroup** over into the VPC window.
17. In the *Properties* pane, click the pencil icon to rename it "WebServerSecurityGroup".
18. In the *Resource types* pane, drag **InternetGateway** over into the main window *outside* of the VPC.
19. In the *Properties* pane, click the pencil icon to rename it "InternetGateway".
20. Click the purple dot in the top right corner of the internet gateway (which will say *VPCGatewayAttachment* when you hover over it), and drag it to the VPC. It will turn red when the connection is made, so then you can release it.
21. In the *Resource types* pane, drag **RouteTable** over into the VPC window.
22. In the *Properties* pane, click the pencil icon to rename it "PublicRouteTable".
23. In the *Resource types* pane, drag **Route** over into the PublicRouteTable window.
24. In the *Properties* pane, click the pencil icon to rename it "PublicRoute".
25. Click the purple dot in the top right of the PublicRoute (which will say *(Property: GatewayId)* when you hover over it), and drag it to the internet gateway. It will turn red when the connection is made, so then you can release it.
26. Click the pink dot in the top right of the PublicRoute (which will say *DependsOn* when you hover over it), and drag it to the purple dot in the top right corner of the internet gateway (the *VPCGatewayAttachment* from before).

27. Click the pink dot in the WebServerInstance (which will say *DependsOn* when you hover over it), and drag it to the PublicRoute.
28. Find the purple dot in the PublicRouteTable and drag it to the PublicSubnet.
29. Refresh the Designer.
30. Hover over the paper icon in the top left corner above *Resource types*, and click **Save**.
31. Select **Local file**, and click **Save**.

## Set the Parameters, Mappings, and Outputs for the Template

1. Click on the canvas outside the VPC to view the parameters lower on the screen.
2. Select **YAML**.
3. In the *Parameters* section, paste in the following (also found in the solution template in the beginning of the lab guide):

```
Parameters:
  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t2.small
    AllowedValues:
      - t1.micro
      - t2.nano
      - t2.micro
      - t2.small
      - t2.medium
      - t2.large
      - m1.small
      - m1.medium
      - m1.large
      - m1.xlarge
      - m2.xlarge
      - m2.2xlarge
      - m2.4xlarge
      - m3.medium
      - m3.large
      - m3.xlarge
      - m3.2xlarge
```

```
      - m4.large
      - m4.xlarge
      - m4.2xlarge
      - m4.4xlarge
      - m4.10xlarge
      - c1.medium
      - c1.xlarge
      - c3.large
      - c3.xlarge
      - c3.2xlarge
      - c3.4xlarge
      - c3.8xlarge
      - c4.large
      - c4.xlarge
      - c4.2xlarge
      - c4.4xlarge
      - c4.8xlarge
      - g2.2xlarge
      - g2.8xlarge
      - r3.large
      - r3.xlarge
      - r3.2xlarge
      - r3.4xlarge
      - r3.8xlarge
      - i2.xlarge
      - i2.2xlarge
      - i2.4xlarge
      - i2.8xlarge
      - d2.xlarge
      - d2.2xlarge
      - d2.4xlarge
      - d2.8xlarge
      - hi1.4xlarge
      - hs1.8xlarge
      - cr1.8xlarge
      - cc2.8xlarge
      - cg1.4xlarge
    ConstraintDescription: must be a valid EC2 instance type.
  KeyName:
    Description: Name of an EC2 KeyPair to enable SSH access to the
```

```
instance.
    Type: 'AWS::EC2::KeyPair::KeyName'
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
  SSHLocation:
    Description: ' The IP address range that can be used to access the
web server using SSH.'
    Type: String
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: '(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})'
    ConstraintDescription: must be a valid IP CIDR range of the form
x.x.x.x/x.
```

4. Click the **Mappings** tab, and paste in the following (also found in the solution template):

```
Mappings:
  AWSInstanceType2Arch:
    t1.micro:
      Arch: HVM64
    t2.nano:
      Arch: HVM64
    t2.micro:
      Arch: HVM64
    t2.small:
      Arch: HVM64
    t2.medium:
      Arch: HVM64
    t2.large:
      Arch: HVM64
    m1.small:
      Arch: HVM64
    m1.medium:
      Arch: HVM64
    m1.large:
      Arch: HVM64
    m1.xlarge:
      Arch: HVM64
    m2.xlarge:
      Arch: HVM64
```

```
      m2.2xlarge:
        Arch: HVM64
      m2.4xlarge:
        Arch: HVM64
      m3.medium:
        Arch: HVM64
      m3.large:
        Arch: HVM64
      m3.xlarge:
        Arch: HVM64
      m3.2xlarge:
        Arch: HVM64
      m4.large:
        Arch: HVM64
      m4.xlarge:
        Arch: HVM64
      m4.2xlarge:
        Arch: HVM64
      m4.4xlarge:
        Arch: HVM64
      m4.10xlarge:
        Arch: HVM64
      c1.medium:
        Arch: HVM64
      c1.xlarge:
        Arch: HVM64
      c3.large:
        Arch: HVM64
      c3.xlarge:
        Arch: HVM64
      c3.2xlarge:
        Arch: HVM64
      c3.4xlarge:
        Arch: HVM64
      c3.8xlarge:
        Arch: HVM64
      c4.large:
        Arch: HVM64
      c4.xlarge:
        Arch: HVM64
```

```
    c4.2xlarge:
      Arch: HVM64
    c4.4xlarge:
      Arch: HVM64
    c4.8xlarge:
      Arch: HVM64
    g2.2xlarge:
      Arch: HVMG2
    g2.8xlarge:
      Arch: HVMG2
    r3.large:
      Arch: HVM64
    r3.xlarge:
      Arch: HVM64
    r3.2xlarge:
      Arch: HVM64
    r3.4xlarge:
      Arch: HVM64
    r3.8xlarge:
      Arch: HVM64
    i2.xlarge:
      Arch: HVM64
    i2.2xlarge:
      Arch: HVM64
    i2.4xlarge:
      Arch: HVM64
    i2.8xlarge:
      Arch: HVM64
    d2.xlarge:
      Arch: HVM64
    d2.2xlarge:
      Arch: HVM64
    d2.4xlarge:
      Arch: HVM64
    d2.8xlarge:
      Arch: HVM64
    hi1.4xlarge:
      Arch: HVM64
    hs1.8xlarge:
      Arch: HVM64
```

```
    cr1.8xlarge:
      Arch: HVM64
    cc2.8xlarge:
      Arch: HVM64
  AWSRegionArch2AMI:
    us-east-1:
      HVM64: ami-0ff8a91507f77f867
      HVMG2: ami-0a584ac55a7631c0c
    us-west-2:
      HVM64: ami-a0cfeed8
      HVMG2: ami-0e09505bc235aa82d
    us-west-1:
      HVM64: ami-0bdb828fd58c52235
      HVMG2: ami-066ee5fd4a9ef77f1
    eu-west-1:
      HVM64: ami-047bb4163c506cd98
      HVMG2: ami-0a7c483d527806435
    eu-west-2:
      HVM64: ami-f976839e
      HVMG2: NOT_SUPPORTED
    eu-west-3:
      HVM64: ami-0ebc281c20e89ba4b
      HVMG2: NOT_SUPPORTED
    eu-central-1:
      HVM64: ami-0233214e13e500f77
      HVMG2: ami-06223d46a6d0661c7
    ap-northeast-1:
      HVM64: ami-06cd52961ce9f0d85
      HVMG2: ami-053cdd503598e4a9d
    ap-northeast-2:
      HVM64: ami-0a10b2721688ce9d2
      HVMG2: NOT_SUPPORTED
    ap-northeast-3:
      HVM64: ami-0d98120a9fb693f07
      HVMG2: NOT_SUPPORTED
    ap-southeast-1:
      HVM64: ami-08569b978cc4dfa10
      HVMG2: ami-0be9df32ae9f92309
    ap-southeast-2:
      HVM64: ami-09b42976632b27e9b
```

```
    HVMG2: ami-0a9ce9fecc3d1daf8
  ap-south-1:
    HVM64: ami-0912f71e06545ad88
    HVMG2: ami-097b15e89dbdcfcf4
  us-east-2:
    HVM64: ami-0b59bfac6be064b78
    HVMG2: NOT_SUPPORTED
  ca-central-1:
    HVM64: ami-0b18956f
    HVMG2: NOT_SUPPORTED
  sa-east-1:
    HVM64: ami-07b14488da8ea02a0
    HVMG2: NOT_SUPPORTED
  cn-north-1:
    HVM64: ami-0a4eaf6c4454eda75
    HVMG2: NOT_SUPPORTED
  cn-northwest-1:
    HVM64: ami-6b6a7d09
    HVMG2: NOT_SUPPORTED
```

5. Click the **Outputs** tab, and paste in the following (also in the solution template):

```
Outputs:
  URL:
    Value: !Join
      - ''
      - - 'http://'
        - !GetAtt
          - WebServerInstance
          - PublicIp
    Description: Newly created application URL
```

## Set the Properties for the Resources in the Template

1. Click on the VPC part of the template design, and paste the following (also in the solution template) into the *Properties* window:

```
Resources:
  VPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      EnableDnsSupport: 'true'
```

```
      EnableDnsHostnames: 'true'
      CidrBlock: 10.0.0.0/16
```

2. Click on the PublicSubnet part of the template design, and paste the following (also in the solution template) into the *Properties* window:

```
Resources:
  PublicSubnet:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref VPC
      CidrBlock: 10.0.0.0/24
```

3. Click on the PublicRoute part of the template design, and paste the following (also in the solution template) into the *Properties* window:

```
Resources:
  PublicRoute:
    Type: 'AWS::EC2::Route'
      Properties:
        RouteTableId: !Ref PublicRouteTable
        GatewayId: !Ref InternetGateway
        DestinationCidrBlock: 0.0.0.0/0
```

4. Click on the WebServerSecurityGroup part of the template design, and paste the following (also in the solution template) into the *Properties* window:

```
Resources:
  WebServerSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      VpcId: !Ref VPC
      GroupDescription: Allow access from HTTP and SSH traffic
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: '80'
          ToPort: '80'
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: '22'
          ToPort: '22'
          CidrIp: !Ref SSHLocation
```

5. Click on the WebServerInstance part of the template design, and paste the following (also in the solution template) into the *Properties* window:

```
Resources:
  WebServerInstance:
    Type: 'AWS::EC2::Instance'
    Properties:
      InstanceType: !Ref InstanceType
      ImageId: !FindInMap
        - AWSRegionArch2AMI
        - !Ref 'AWS::Region'
        - !FindInMap
          - AWSInstanceType2Arch
          - !Ref InstanceType
          - Arch
      KeyName: !Ref KeyName
      NetworkInterfaces:
        - GroupSet:
            - !Ref WebServerSecurityGroup
          AssociatePublicIpAddress: 'true'
          DeviceIndex: '0'
          DeleteOnTermination: 'true'
          SubnetId: !Ref PublicSubnet
      UserData: !Base64
        'Fn::Join':
          - ''
          - - |
              #!/bin/bash -xe
            - |
              yum install -y aws-cfn-bootstrap
            - |
              # Install the files and packages from the metadata
            - '/opt/aws/bin/cfn-init -v '
            - '         --stack '
            - !Ref 'AWS::StackName'
            - '         --resource WebServerInstance '
            - '         --configsets All '
            - '         --region '
            - !Ref 'AWS::Region'
            - |+
```

```
              - |
                # Signal the status from cfn-init
              - '/opt/aws/bin/cfn-signal -e $? '
              - '         --stack '
              - !Ref 'AWS::StackName'
              - '         --resource WebServerInstance '
              - '         --region '
              - !Ref 'AWS::Region'
              - |+
```

6. Click the **Metadata** tab, and paste in the following:

```
Resources:
  WebServerInstance:
    Type: 'AWS::EC2::Instance'
    Metadata:
      'AWS::CloudFormation::Designer':
        id: 4c81200b-5e79-4b30-a37a-96f654b652dc
      'AWS::CloudFormation::Init':
        configSets:
          All:
            - ConfigureSampleApp
        ConfigureSampleApp:
          packages:
            yum:
              httpd: []
          files:
            /var/www/html/index.html:
              content: !Join
                - |+

                - - >-
                    <h1>Congratulations, you have successfully launched
the AWS
                    CloudFormation sample.</h1>
              mode: '000644'
              owner: root
              group: root
          services:
            sysvinit:
              httpd:
```

```
enabled: 'true'
ensureRunning: 'true'
```

7. Click the checkbox at the top, above *Resource types*, to validate the template.
8. Save the template again.

### Complete Creation of the Stack and Browse to the Web Server

1. Click the cloud icon with the up arrow to create the stack.
2. Open a new browser tab, and navigate to **EC2** > **Key Pairs**.
3. Click **Create Key Pair**.
4. Give it a *Key pair name* of "designerlab", and click **Create**.
5. Back in the stack creation browser tab, click **Next**.
6. On the stack details page, set the following values:
    - *Stack name*: **cfdesignerlab**
    - *InstanceType*: **t2.micro**
    - *KeyName*: **designerlab**
7. Click **Next**.
8. Leave the settings on the stack options page as-is, and click **Next**.
9. Click **Create stack**.
10. Once the creation of all resources and the stack is complete, click the **Outputs** tab.
11. Open the URL listed in a new browser tab. We should see a message letting us know we successfully launched the CloudFormation template.

## Conclusion

Congratulations on successfully completing this hands-on lab!

### Tools

Lab Diagram  Instant Terminal

### Credentials

How do I connect?

## AWS Account

Username

cloud_user

Password

QBT$_3Mt26sT=+qoseL7

Open Link in Incognito Window

## Additional Resources

Log in with the credentials provided, and make sure you are in the `us-east-1` (N. Virginia) region.

There is a CloudFormation template provided with the lab (solution template) on GitHub.

## Learning Objectives

0 of 2 completed

- [ ] Create an AWS CloudFormation Template and Stack with CloudFormation Designer
- [ ] Create and Test the Web Server