AWS CloudFormation

aws

What's in it for you

AWS CloudFormation Deepdive	
S.NO.	**AGENDA**
1	**Why use AWS CloudFormation?**
2	**What is CloudFormation?**
3	**How does AWS CloudFormation work?**
4	**AWS CloudFormation concepts:** - Template in AWS CloudFormation - Stack in AWS CloudFormation
5	**CloudFormation access control**
6	**Demo - LAMP stack on EC2 Instance**

Speakers

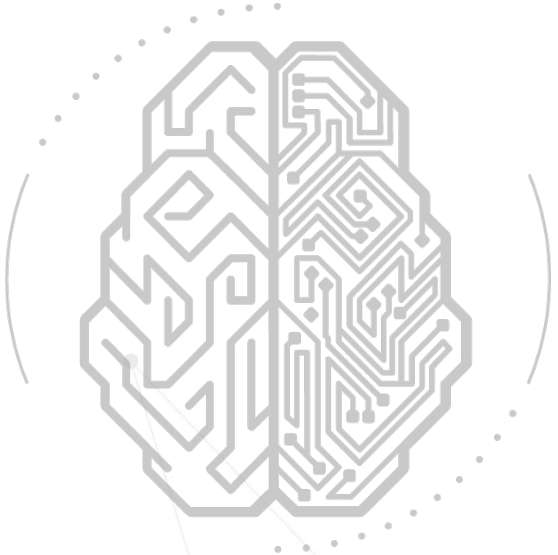
Sanchit Jain

Lead Architect - AWS at Quantiphi
AWS APN Ambassador

Why use AWS CloudFormation?

Support for Different Resources:	AWS CF assists a smorgasbord of resources, which lets you make a highly reliable, available, and scalable or upgradeable AWS infrastructure to cater to your specific application requirements.
Easy to use:	With CloudFormation, it's easy to classify and station a suite of the secure platform's resources. It allows you to describe any contingencies or special yardsticks to pass in during runtime. You may employ any of the several CloudFormation sample templates verbatim or as a beginning point.
Flexible and Declarative:	To erect the infrastructure you require, you list out the platform's resources, interconnections, and configuration values you want the template to have and then permit CloudFormation to take care of the rest with some basic clicks in the console
Customization through parameters:	You may use parameters for customizing specific template aspects at run time. For instance, you could pass the Amazon EC2 instance types, Amazon EBS volume size, RDS database size, server port numbers, and database to the platform when creating a stack.
Drag and drop UI to visualize and edit:	AWS CloudFormation Designer offers a template diagram vhth icons denoting the Amazon platform's resources and arrow signs indicating relationships. You could create and modify templates using the interface and then alter template details with the help of the inbuilt JSON text editor.

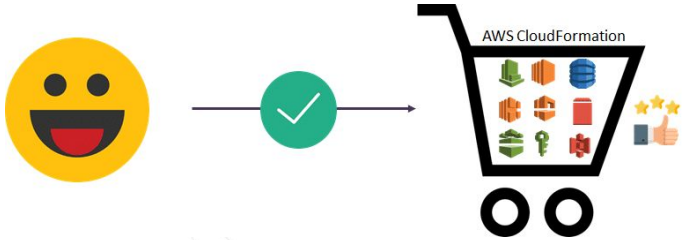
What is AWS CloudFormation?



What is AWS CloudFormation?

AWS CloudFormation provides users a simple way to create and manage a collection of AWS resources by provisioning and updating them in an orderly and predictable way

Create and manage AWS resources



In simple terms, it allows you to create and model your infrastructure and applications without having to perform manual actions

What is AWS CloudFormation?

AWS CloudFormation provides a simple way to create and manage a collection of AWS resources by provisioning and updating them in an orderly and predictable way

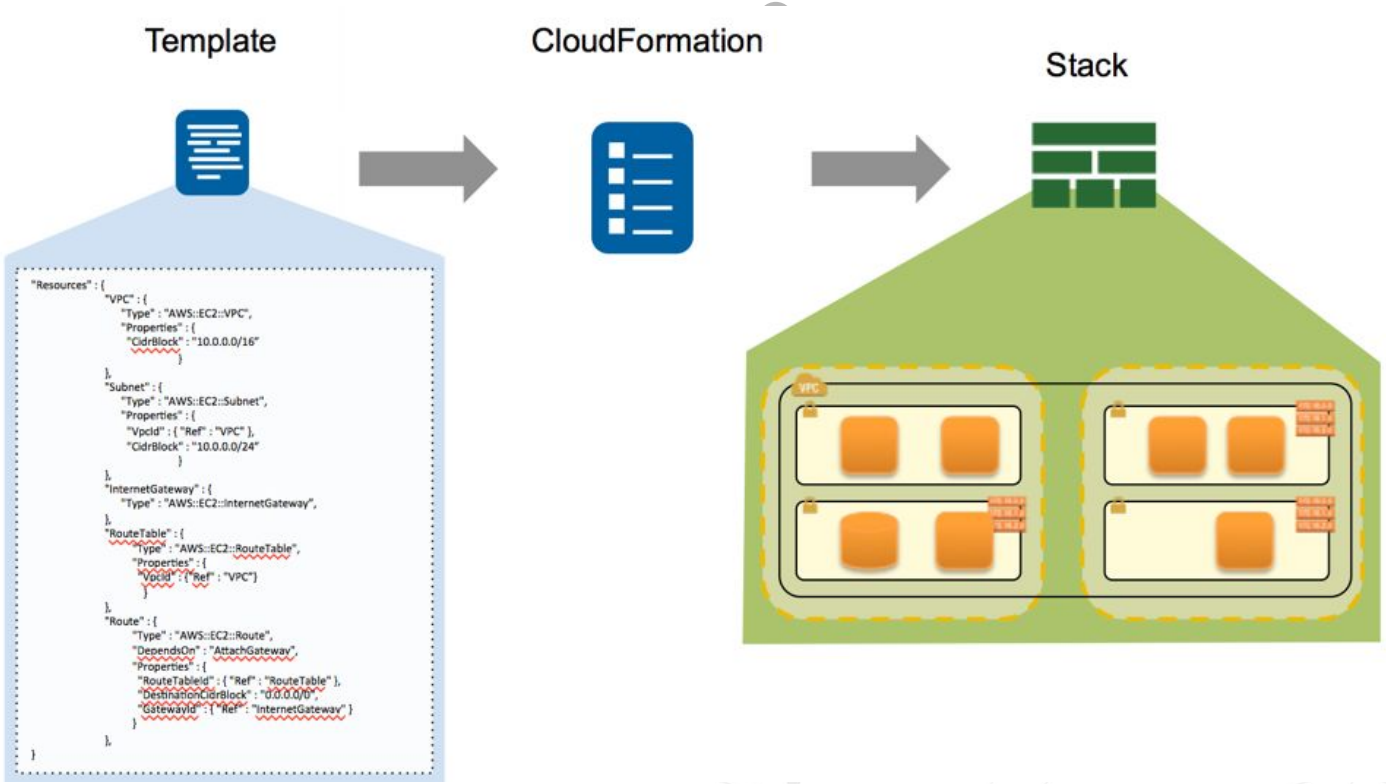
For Example

With AWS CloudFormation, Expedia is able to deploy and easily manage its entire front and backend AWS resources into its cloud environment

Create and manage AWS resources

In simple terms, it allows you to create and model your infrastructure and applications without having to perform manual actions

What is AWS CloudFormation?



What is AWS CloudFormation?

All the resources required by user in an application can be deployed easily using templates

Also, you can reuse the template to replicate your infrastructure in multiple environments

To make the templates reusable, use the parameters, mappings, and conditions sections, in the template so that you can customize your stacks when you create them

How AWS CloudFormation work?

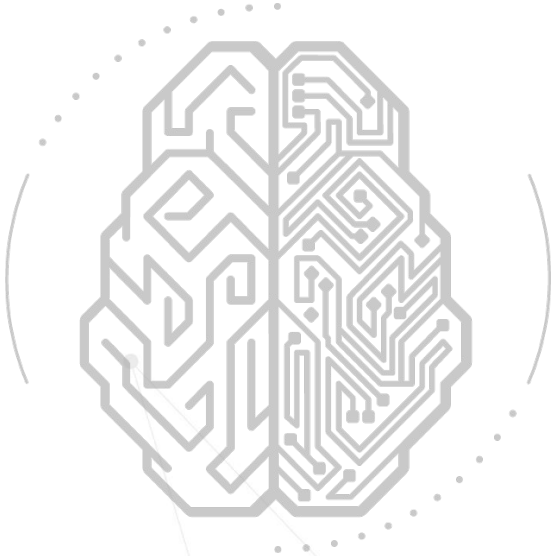
Code your infrastructure from scratch with the CloudFormation template language, in either YAML or JSON format, or start from many available sample templates

Check out your template code locally, or upload it into an S3 bucket

Use AWS CloudFormation via the browser console, command line tools or APIs to create a stack based on your template code

AWS CloudFormation provisions and configures the stacks and resources you specified on your template

AWS CloudFormation Concepts

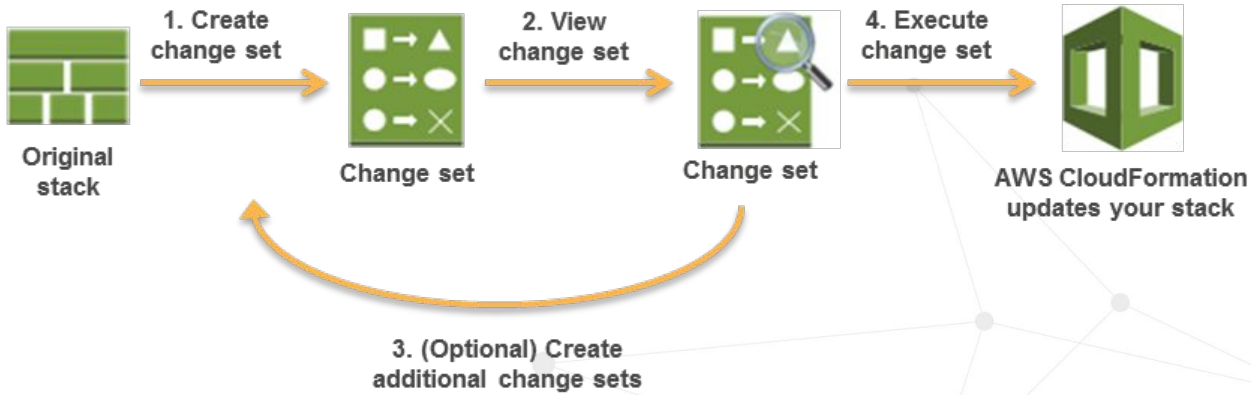


AWS CloudFormation Concepts

Template	CloudFormation	Stack
JSON/YAML formatted file	Framework	Configured AWS resources
Parameter definition	Stack creation	Comprehensive service support
Resource creation	Stack updates	Service event aware
Configuration actions	Error detection and rollback	Customizable

Templates

- A template in AWS CloudFormation is a formatted text file in JSON or YAML language that describes your AWS infrastructure
- To create, view and modify templates you can use AWS CloudFormation Designer or any text editor tool



Templates

- A template in AWS CloudFormation is a formatted text file in JSON or YAML language that describes your AWS infrastructure
- To create, view and modify templates you can use AWS CloudFormation Designer or any text editor tool
- Templates in AWS CloudFormation consists of 9 main objects

Format version

Description

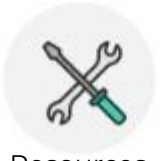
Metadata

Parameters

Mappings

Conditions

Transform



Resources

Outputs

Template Structure

JSON

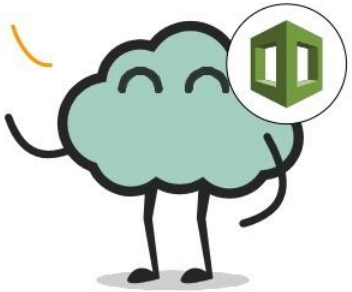
```
{
  "AWSTemplateFormatVersion" : "version date",

  "Description" : "JSON string",

  "Metadata" : {
    template metadata
  },

  "Parameters" : {
    set of parameters
  },

  "Mappings" : {
    set of mappings
  },

  "Conditions" : {
    set of conditions
  },

  "Transform" : {
    set of transforms
  },

  "Resources" : {
    set of resources
  },

  "Outputs" : {
    set of outputs
  }
}
```

YAML

```
---
AWSTemplateFormatVersion: "version date"

Description:
  String

Metadata:
  template metadata

Parameters:
  set of parameters

Mappings:
  set of mappings

Conditions:
  set of conditions

Transform:
  set of transforms

Resources:
  set of resources

Outputs:
  set of outputs
```

Template Structure

Now, let's discuss each and every object of template structure



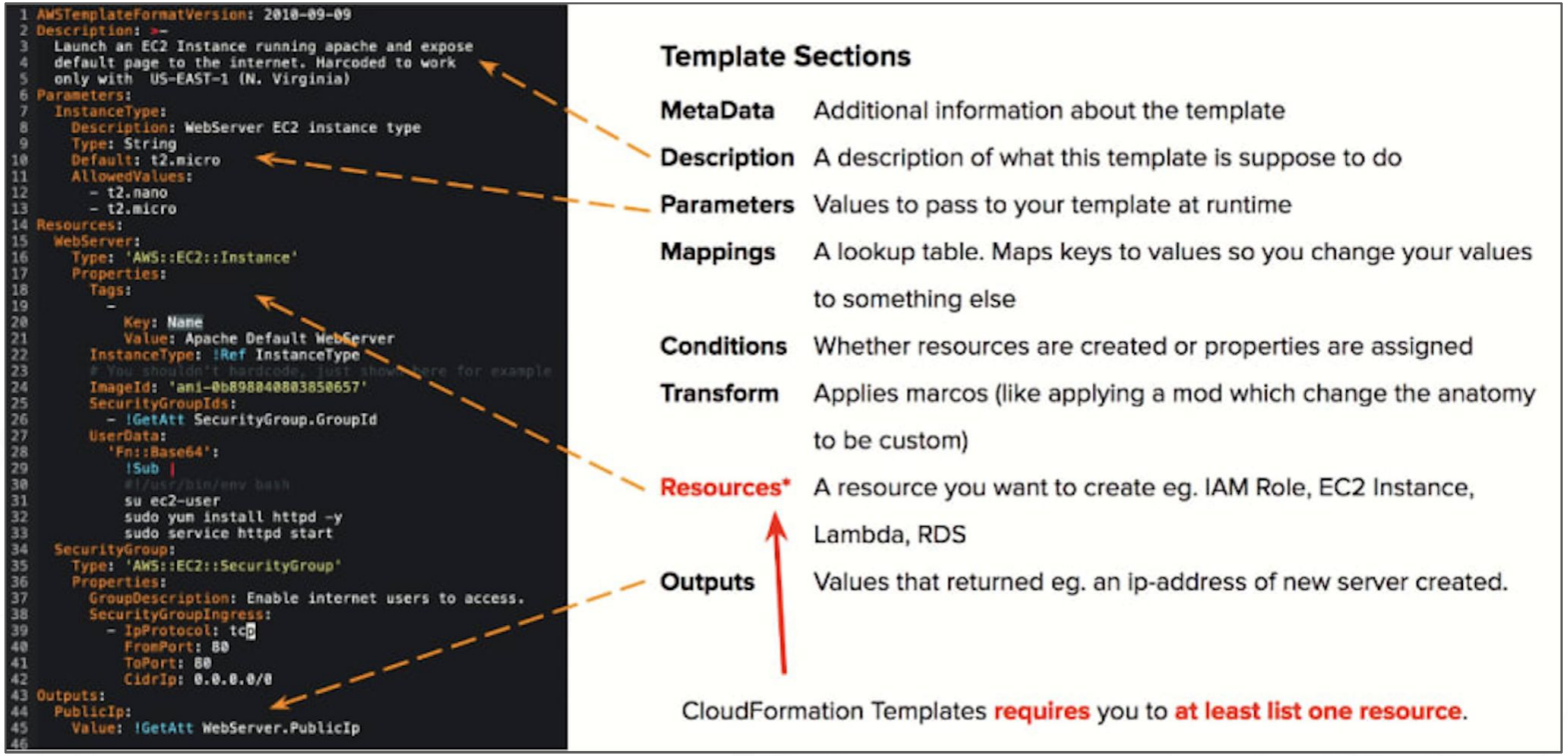
JSON

```
{
  "AWSTemplateFormatVersion" : "version date",

  "Description" : "JSON string",

  "Metadata" : {
    template metadata
  },

  "Parameters" : {
    set of parameters
  },

  "Mappings" : {
    set of mappings
  },

  "Conditions" : {
    set of conditions
  },

  "Transform" : {
    set of transforms
  },

  "Resources" : {
    set of resources
  },

  "Outputs" : {
    set of outputs
  }
}
```

YAML

```
---
AWSTemplateFormatVersion: "version date"

Description:
  String

Metadata:
  template metadata

Parameters:
  set of parameters

Mappings:
  set of mappings

Conditions:
  set of conditions

Transform:
  set of transforms

Resources:
  set of resources

Outputs:
  set of outputs
```

Template Structure



Metadata

Metadata can be used in the template to provide further information using JSON or YAML objects

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
          default: 'Amazon EC2
Configuration'
        Parameters:
          - InstanceType

    ParameterLabels:
      InstanceType:
        default: 'Type of EC2 Instance'
```

Mapping

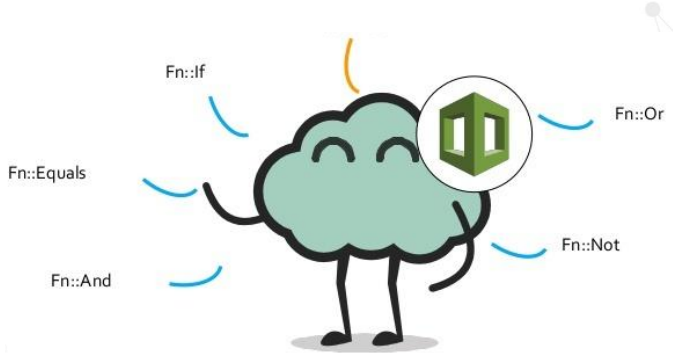
- For Example : Based on a region, you can set values. In a template, you can create a mapping that uses a key and holds the values that you want to specify for each specific region

```
{
  ...
  "Mappings" : {
    "RegionMap" : {
      "us-east-1" : { "32" : "ami-6411e20d", "64" : "ami-7a11e213" },
      "us-west-1" : { "32" : "ami-c9c7978c", "64" : "ami-cfc7978a" },
      "eu-west-1" : { "32" : "ami-37c2f643", "64" : "ami-31c2f645" }
      }},
  "Resources" : {
    "myEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "32"]},
        "InstanceType" : "m1.small"
      } }
  }
}
```

Conditions

- Conditions can be used when you want to reuse the templates by creating resources in different context
- In a template, during stack creation, all the conditions in your template are evaluated
- All resources that are associated with a true condition are created and the invalid conditions are ignored automatically

*You can use **intrinsic functions** to define conditions*



Conditions

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Parameters": {
        "EnvType": {
            "Description": "Environment type.",
            "Default": "test",
            "Type": "String",
            "AllowedValues": ["prod","test"],
            "ConstraintDescription": "must specify prod or test."
    }},
    "Conditions": {
        "CreateProdResources": {
            "Fn::Equals": [
                {
                    "Ref": "EnvType"
                },
                "prod"
     ]}},
    "Resources": {
        "EC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "ami-0ff8a91507f77f867"
            }
       }}}
```

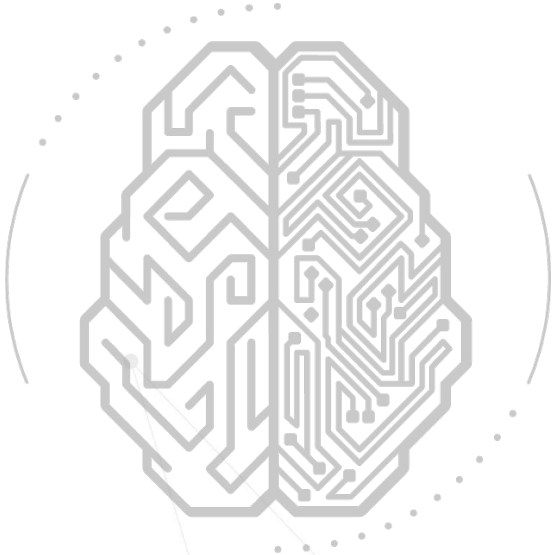
Transform

- Transform build a simple declarative language for CloudFormation and enables reuse of template components
- Here, you can declare a single transform or multiple transforms within a template

```
// Start of processable content for MyMacro and AWS::Serverless
Transform:
  - MyMacro
  - 'AWS::Serverless'
Resources:
  WaitCondition:
    Type: 'AWS::CloudFormation::WaitCondition'
  MyBucket:
    Type: 'AWS::S3::Bucket'
    Properties:
        BucketName: MyBucket
        Tags: [{"key":"value"}]
        CorsConfiguration:[]
  // End of processable content for MyMacro and AWS::Serverless
```

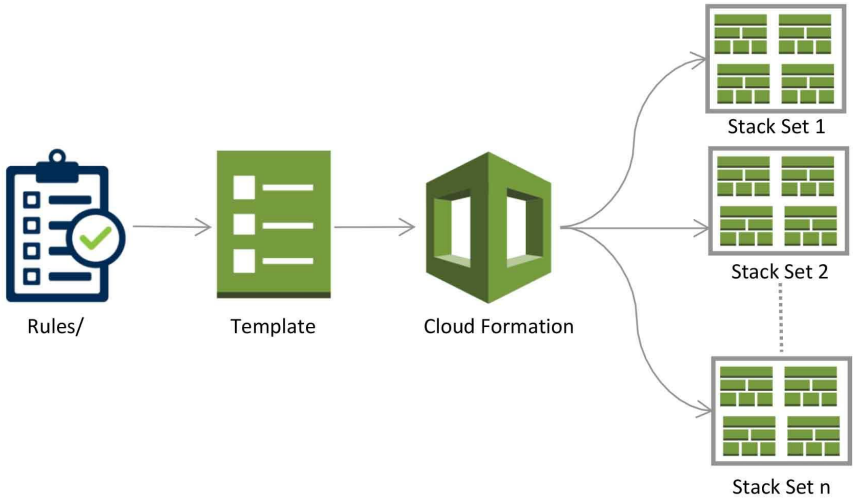
Reuse of template

Stacks in AWS CloudFormation



Stack

- A collection of AWS resources is called a Stack and it can be managed in a single unit
- CloudFormation's Template defines a stack in which the resources can be created, deleted, or updated in a predictable way
- A stack can have all the resources (web server, database etc) that can be required to run a web application

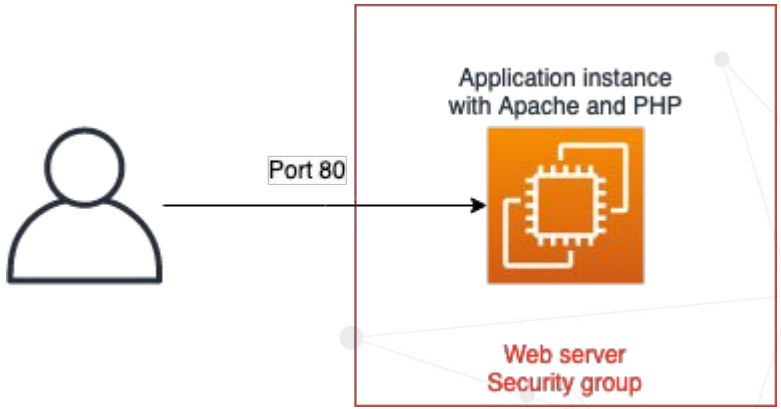


DEMO

Demo

- In this lab you will deploy an Apache Web server with a simple PHP application via UserData property.
    - First, you will bootstrap EC2 instance to install web server and content.
    - Then you will create an EC2 Security Group and allow access on port 80 to the instance.
    - Finally, you will view the content served by the web server.

The following diagram provides a high-level overview of the architecture you will implement.



Cloudformation Output

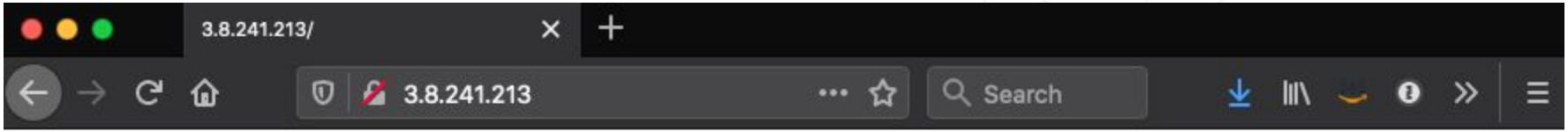
Stack info | Events | Resources | **Outputs** | Parameters | Template | Change sets

Outputs (3)

Search outputs

Key	Value	Description	Export name
WebServerElasticIP	3.8.241.213	Elastic IP assigned to EC2	-
WebServerPublicDNS	ec2-3-8-241-213.eu-west-2.compute.amazonaws.com	Public DNS of EC2 instance	-
WebsiteURL	http://3.8.241.213	Application URL	-

Browser Output



EC2 Instance ID: i-021b2dc6ccdfd1fee

Availability Zone: eu-west-2c

THANK YOU