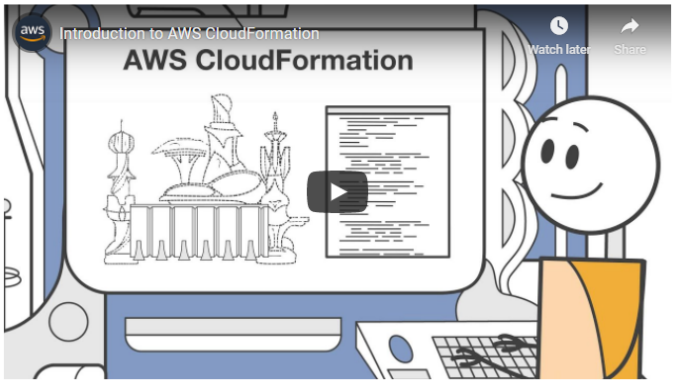
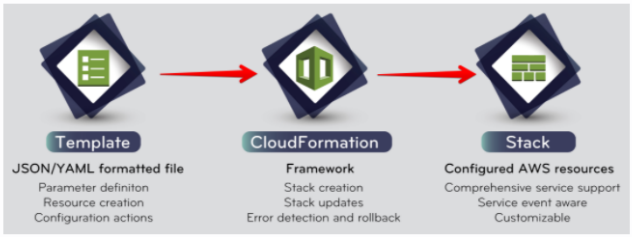
Introduction to CloudFormation

What is CloudFormation?

CloudFormation is an AWS service which enables you to create, manage, configure, replicate and delete AWS resources easily and rapidly using templates. Creating one or few resources, setting and deleting them when you are done are not so difficult and time consuming, but what if you have to deal with hundreds of them and even in different regions? Creating, setting, and managing **stacks** (collection of AWS resources) can be done in an optimal way, AWS CloudFormation lets you do these processes with **templates**, formatted text files in JSON or YAML.



Stacks and Templates



Stacks and Templates

Stacks and templates are the main components of AWS CloudFormation. A **template** is a JSON or YAML formatted text file which you specify the AWS resources you want to create. Templates can have the extensions `.json`, `.yaml`, `.template`, or `.txt`. You create resources, define the parameters and configure your settings with templates.

A **stack** on the other hand is a single unit composed of the AWS resources provisioned by CloudFormation. A stack can be either a single EC2 instance or a very complex VPC configuration including hundreds of resources like S3 buckets, databases and NAT Gateways. All of the resources in a stack are defined and configured by the related template.

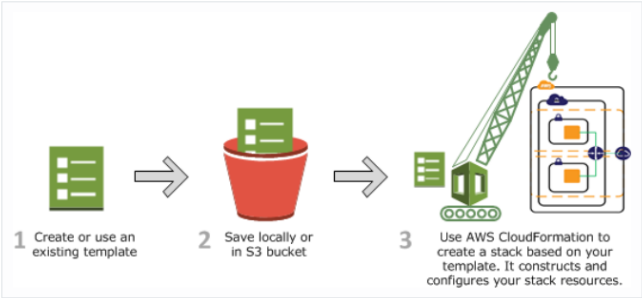
Example JSON Syntax:

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "A simple EC2 instance",
  "Resources" : {
    "MyEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "ami-0ff8a91507f77f867",
        "InstanceType" : "t1.micro"
      }
    }
  }
}
```

Example YAML Syntax:

```
AWSTemplateFormatVersion: '2010-09-09'
Description: A simple EC2 instance
Resources:
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: t1.micro
```

CloudFormation's Working Process



CloudFormation's Working Process

First, you create a template using a text editor or AWS CloudFormation Designer (AWS's graphical interface for creating templates). You can also specify a sample template provided by AWS for your project. The template has to describe the resources and the properties of these resources you need in your stack.

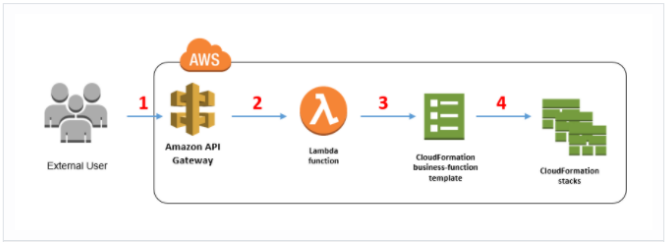
Second, you save your template in an S3 bucket or locally in your computer.

And last, you create a CloudFormation stack by specifying the path of the template file (i.e. a local folder or an S3 URL). You can define parameters to specify input values while the stack is being created.

CloudFormation provisions and configures resources by making calls to the relevant AWS services that are described in your template. You can use your stack after CloudFormation reports it's ready. If the creation of the stack fails somehow, CloudFormation rolls back the changes by deleting the resources created.

You can also update or delete your stack whenever you want.

Accessing and Pricing



CloudFormation API Accessing

Accessing:

AWS enables you to access CloudFormation via:

- **AWS Management Console**: You can use Management Console to easily access CloudFormation like many other services.
- **AWS Command Line Interface**: You can use also CLI to access CloudFormation. Please check this link if you need more information about AWS CloudFormation CLI.
- **CloudFormation API**: AWS supports accessing CloudFormation via API, if you want to use. Please see AWS CloudFormation API Reference for information about making API requests and API actions.

Pricing:

AWS doesn't require any additional fee for using CloudFormation, you only pay for the resources you use. Yet, some third party resource providers might be an exception, please check AWS CloudFormation Pricing for more information. If you are a free tier user, you can leverage CloudFormation as if you use the AWS resources manually.

Understanding Templates

A template is the core of a CloudFormation stack. Remember the options that you encounter while creating a simple EC2 instance, you specify an AMI, select an instance type, choose a key pair, etc. While creating a stack with CloudFormation, you don't make these selections via Management Console, you specify all of this information within the template. Hence, understanding a template is crucial.

A Template has the following high-level JSON Structure:

```
{
  "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
  }
}
```

Templates consist of sections. All of the sections, except the resources, are optional. The resources section is mandatory. Some sections in a template can be in any order. However, while building a template, it can be helpful to use the logical order shown in the following list, because values in one section might refer to values from a previous section.

AWSTemplateFormatVersion: This is the format version of the template. The latest template format version is 2010-09-09 and currently the only valid one. If you don't specify a value, AWS CloudFormation assumes the latest template format version.

Description: Templates can have various information and can make different jobs, so here you can make a useful description of the template to know what it exactly does.

Metadata: You can use metadata to add arbitrary JSON or YAML objects that provide more information about the template, i.e. details about your resources.

Parameters: While all of the inputs can be predefined in a template, some inputs may be needed while creating a stack. For example, you define every feature of the EC2 instance in your template, but ask the number of the instances to the user each time a stack is being created. Instead of changing the template each time to set the number of instances, you ask it as an input from the user per deployment. Parameters let you make these inputs.

Mappings: Mappings section of CloudFormation helps to organize parameters by named keys and corresponding values for each group. A mapping of keys and associated values that you can use to specify conditional parameter values is similar to a lookup table. You can match a key to a corresponding value in the Resources and Outputs sections.

Conditions: You can set conditions if you want some resources to be created, or some properties to be set under certain situations.

Transform: This section specifies one or more macros that AWS CloudFormation uses to process your template.

Resources: AWS resources are defined and configured here. While AWS has many resources with numerous properties, it is not possible to memorize them all for your template. Please check AWS Resource and Property Types Reference for detailed information.

Outputs: If you need some data output after your stack has been created, you can specify it here. The info you defined in your template (e.g. the IP address of the EC2 instance created) will be displayed on the console after creation.

Let's check the simple example template that creates an EC2 instance:

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Description" : "Create an EC2 instance running the Amazon Linux 32 bit AMI.
    "Parameters" : {
        "KeyPair" : {
            "Description" : "The EC2 Key Pair to allow SSH access to the instanc
            "Type" : "String"
        }
    },
    "Resources" : {
        "Ec2Instance" : {
            "Type" : "AWS::EC2::Instance",
            "Properties" : {
                "KeyName" : { "Ref" : "KeyPair" },
                "ImageId" : "ami-3b355a52"
            }
        }
    },
    "Outputs" : {
        "InstanceId" : {
            "Description" : "The InstanceId of the newly created EC2 instance",
            "Value" : {
                "Ref" : "Ec2Instance"
            }
        }
    }
}
```

As you can see, an EC2 instance will be created. The key pair will be asked as an input while the stack is being created, and the Id of the instance will be displayed after being created.

Templates can be very functional but also very complex depending on your project, so it is not possible to cover every topic about them in this lesson. Please see Template Reference for further information about templates.

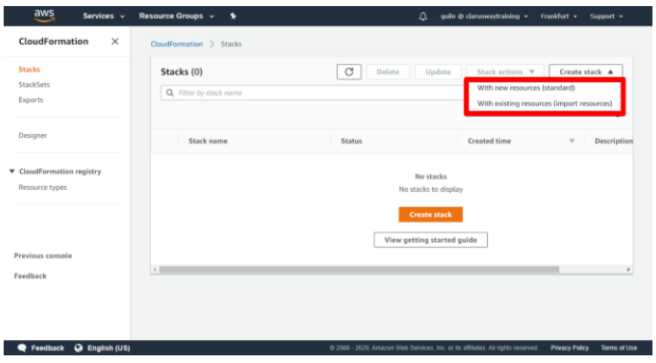
Creating Stack

Getting Started

As mentioned in the introduction, a stack is a single unit composed of the AWS resources from instantiating a template and it can vary from a simple resource to a complicated collection of resources.

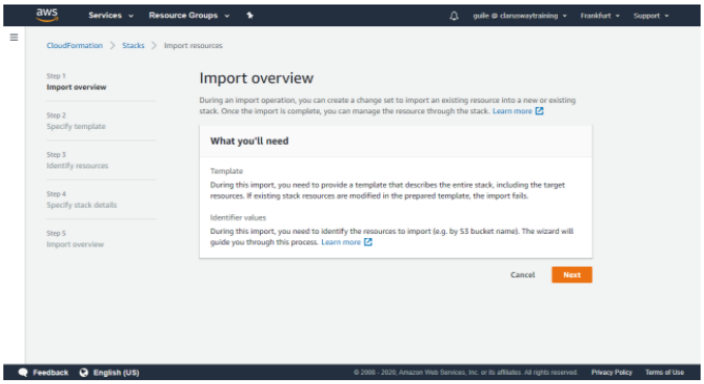
Let's go to the AWS management console to create a new stack. Type CloudFormation in the search box and select it. You can also find and click it under Management & Governance section.

You will see the **Stacks** menu as default when you first get into the CloudFormation service. Here, you can view, create, delete, update, search and filter your stacks. As you can see we don't have any stacks yet. You can start creating a stack either clicking the orange **Create stack** button right in the middle of the console, or clicking the drop-down **Create stack** button on the top right corner to choose the method.



- The orange button in the middle will start creating a stack with new resources which is the standart (default) option.
- If you select **With new resources (standart)** from the upper right button, it will also start creating a stack with new resources.
- And if you select **With existing resources (import resources)**, CloudFormation will use your existing resources (e.g. an EC2 instance or an S3 bucket already created) to create a stack.

Choosing existing resources will direct you to an **Import overview** process. Explanations will guide you to create a stack with your resources. This process is similar to the one which uses new resources with some exceptions.



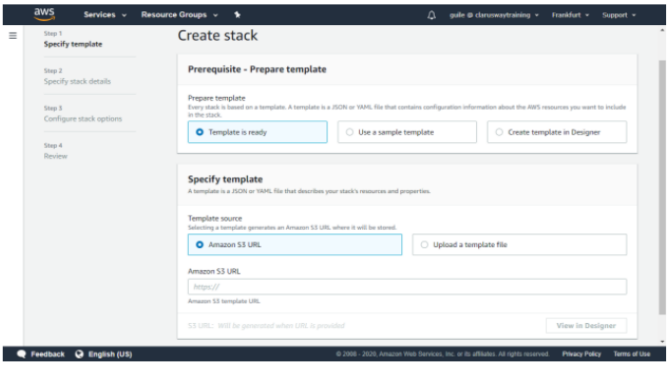
Here, we will create a new stack with new resources. After completing and comprehending this lesson, you can also use your existing resources to create a new stack. Thus, click the orange **Create stack** button in the middle, and move on to the next topic.

Specifying Template

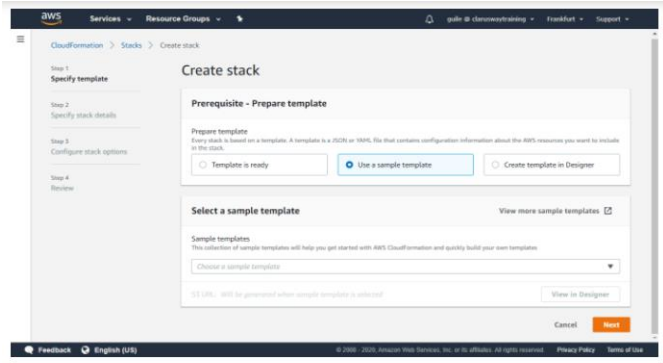
To create a stack, first step is to specify a template to AWS. Thus, having a template is a prerequisite. Here, AWS offers us three options to specify the template which the stack will be based upon:

- Template is ready
- Use a sample template
- Create template in Designer

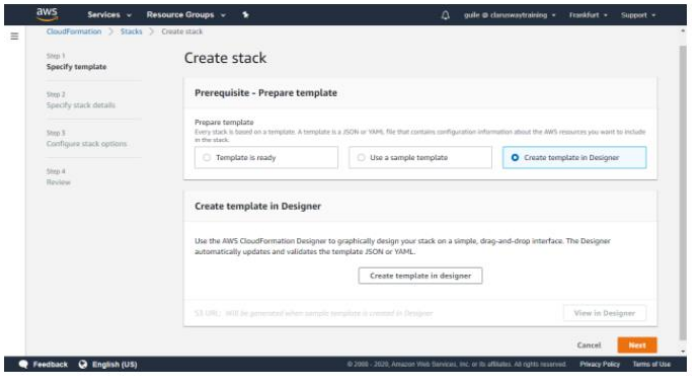
Template is ready: Select this option, if you have already created your template. All you have to do is, typing the S3 template URL in the box if you have one in a bucket or choosing the file to upload locally from your computer.



Use a sample template: AWS provides a collection of sample templates for users to build their stacks easily. These sample templates also can be modified according to the users needs. Click the box to see the list of few sample templates or click **View more sample templates** to see more of them. Because of the different requirements for different regions, one template working in one region might not work properly in another. So, please see the Sample Templates for each region.

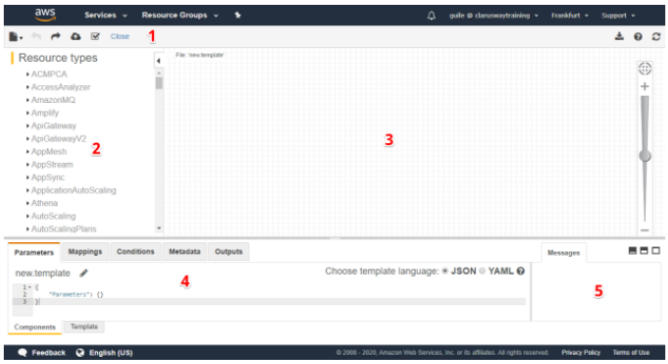


Create template in Designer: AWS CloudFormation Designer is a graphical user interface which enables you to build your templates using a drag and drop menu. You choose the resources and configure them in a visualized environment, Designer compiles your template accordingly. Designer will be covered in the next page.



CloudFormation Designer

If you select creating a template with Designer, you will see the Designer page below. This graphical user interface helps you to create templates to build stacks. By using Designer, you spend less time manually coding your templates and more time designing your AWS stacks. In Designer, you drag and drop new resources to add them to your template, and you drag connections between resources to establish relationships. Designer automatically modifies the JSON or YAML.



Let's check the Designer's panes and main components.

1. Toolbar:

The toolbar provides quick access to commands for common actions, such as opening and saving templates, undoing or redoing changes, creating a stack, and validating your template. You can also download the diagram as an image, get help, or refresh the diagram in the canvas pane.

2. Resource types pane:

The Resource types pane lists all of the template resources that you can add to your template, categorized by their AWS service name. You add resources by dragging them from the Resource types pane to the canvas and you can connect them on the canvas.

3. Canvas pane:

The canvas pane displays your template resources as a diagram. You use it to add or remove resources, create relationships between resources, and arrange their layout. The changes that you make in the canvas automatically modify the template's JSON or YAML. For more information, see Canvas Pane.

4. Integrated JSON and YAML editor pane:

In the integrated editor, you specify the details of your template, such as resource properties or template parameters. You can see the components of your template (Parameters, Mappings, Conditions etc.) in tabs or you can view the whole template. When you select an item in the canvas, Designer highlights the related JSON or YAML in the editor. After editing the JSON or YAML, you must refresh the canvas to update the diagram. You can convert a valid template between JSON and YAML by selecting the appropriate radio button in Choose template language.

5. Messages pane:

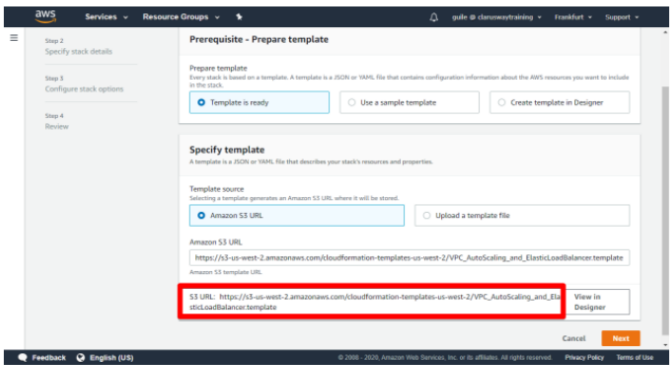
When you convert a template from JSON to YAML or vice-versa, the Messages pane displays a success or failure message. When you open, validate, or attempt to create a stack with an invalid template, the Messages pane displays validation errors.

You will go through an example on the next topic. You will see how a template looks like in Designer.

Template Selection

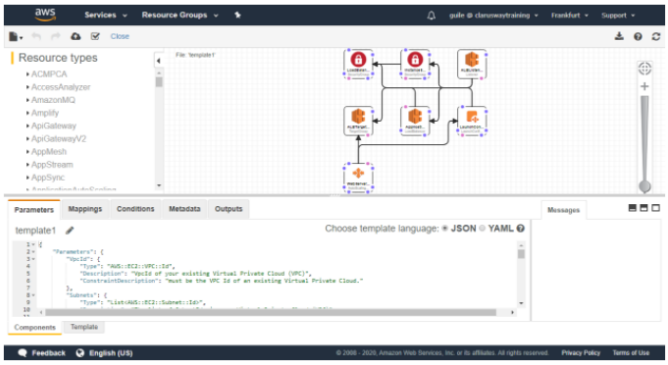
Now that we learned how we can specify our template, let's get back to select our template for stack creation. In this demonstration we will use a template specifying an S3 URL. This sample template will create an Auto Scaling Group behind a Load Balancer and launch EC2 instances hosting a simple Apache Web Server-based PHP page. Please copy the URL below, paste it into the Amazon S3 URL box and hit Enter. See the S3 URL recognised by AWS.

Sample Template: `https://s3-us-west-2.amazonaws.com/cloudformation-templates-us-west-2/VPC_AutoScaling_and_ElasticLoadBalancer.template`



Before moving to the next step please click on the Sample Template and open it. This template will define how our stack is going to be and what it will create on behalf of us. Please take some time to read and understand it. You can't comprehend it totally but you will easily see how it is going on.

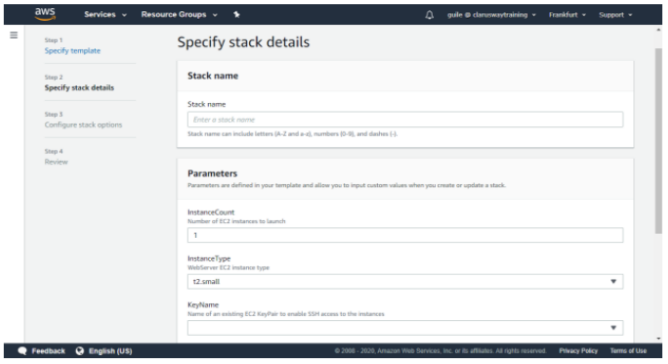
And if you finished reading the template, now click on the **View in Designer** to open it in Designer. That is how this sample template looks like in Designer. Click on the resources/connections on the canvas and check the integrated editor pane to understand what Designer does.



If you feel comfortable with the template and the Designer hit the Next button and move on to the next step.

Specifying Stack Details

In this step we will specify stack details. First section is the stack name. This section is standart for all templates. But unlike the name section, Parameters section depends on the template specified. All of the info required in this section is explicitly defined in our sample template. If you have read it on the previous topic, you can recall the parameters required. These parameters are intentionally required for the stack, because different projects need different inputs.



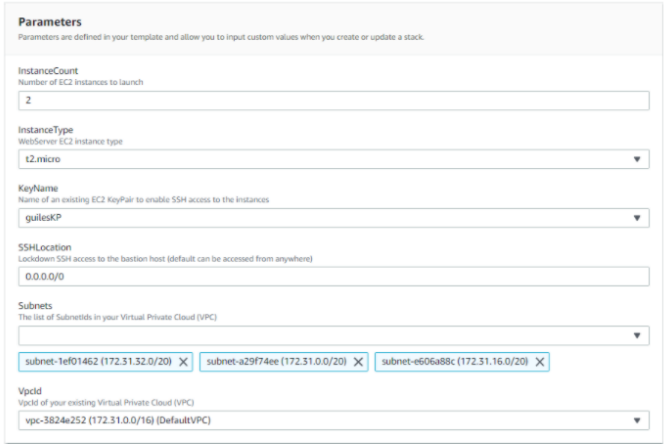
Now, let's start defining these inputs.

Name:

For this demo, we will use **clarusway-stack** for name.

Parameters:

- **InstanceCount:** This is the number of EC2 instances to launch. Let's select **2**.
- **InstanceType:** An eligible EC2 instance type should be selected for the WebServer here. We will choose **t2.micro** to keep up with AWS Free Tier.
- **KeyName:** Name of an existing EC2 KeyPair should be specified to enable SSH access to the instances. **guilesKP** will be selected for this demo. If you don't see any key pair when you click in the box, that means you don't have a valid one. For the sake of this demo we will not login to the instances via SSH. So keep it as is, unless you want an SSH access.
- **SSHLocation:** You can limit SSH access defining the IP range here. Default is accessible from anywhere. We will keep the default **(0.0.0.0/0)**.
- **Subnets:** You will see the list of SubnetIds in your Virtual Private Cloud (VPC) here. Let's select **all subnets**.
- **VpcId:** You will see the VpcId of your existing Virtual Private Cloud (VPC). Let's select **Default VPC**.



If you are done, click Next and move on to the next step.

Configuring Stack Options

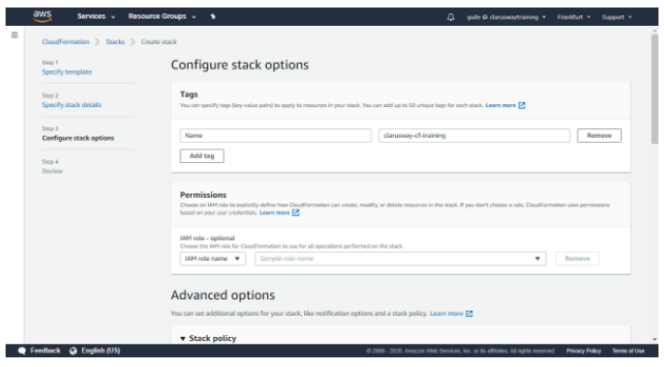
In this step we will configure some basic and advanced stack options. These additional settings are optional.

Tags:

CloudFormation enables us to tag our resources created for the stack. As mentioned before, a stack can consist numerous resources and to tag each one-by-one is not a good idea, thus if you just specify tags here, all of your resources will be instantiated tagged automatically. In addition it will be more easy to seperate the existing resources from those which have been created by CloudFormation. For this demo, let's use **Name : clarusway-cf-training** key-value pair for the resources.

Permissions:

Here, you can choose an IAM role to explicitly define how CloudFormation can create, modify, or delete resources in the stack. If you don't choose a role, CloudFormation uses permissions based on your user credentials. You can select an **IAM role name** or an **IAM role ARN**. For this demo we don't need any permissions for our stack, hence, don't specify anything and leave it as is.

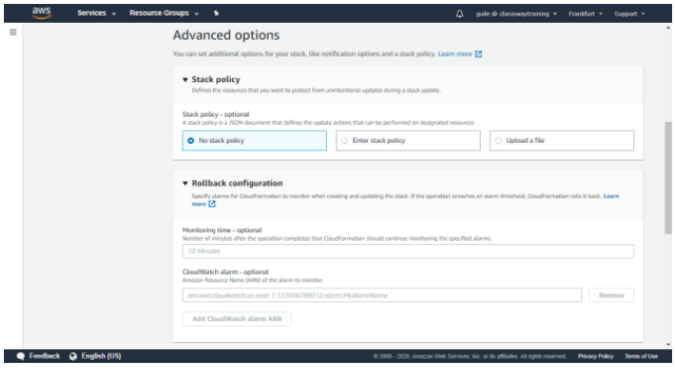


Stack Policy:

CloudFormation lets you specify a policy to define the resources that you want to protect from unintentional updates during a stack update. By default, all resources can be updated in the course of a stack update. During an update, some resources might require an interruption or be completely replaced, resulting in new physical IDs or completely new storage. By using a stack policy, you can prevent stack resources from being unintentionally updated or deleted during a stack update. You can either enter the stack policy directly as JSON, or upload a JSON file containing the stack policy. We don't need one here, so leave the default **No stack policy**.

Rollback configuration:

Rollback is the process which CloudFormation cancels creating the stack and starts deleting the resources that created until that time. Rollback triggers enable you to have CloudFormation monitor the state of your application during stack creation and updating. If the application breaches the threshold of any of the alarms you've specified, CloudFormation starts rolling it back. CloudFormation monitors the specified alarms during the stack create or update operation, and for the specified amount of time (Monitoring time) after all resources have been deployed. The monitoring time can be set from the default of 0 up to 180 minutes. During this time, AWS CloudFormation monitors all the rollback triggers after the stack creation or update operation deploys all necessary resources. If any of the alarms goes to ALARM state during the stack operation or this monitoring period, AWS CloudFormation rolls back the entire stack operation. We will not set any alarms for this demo.

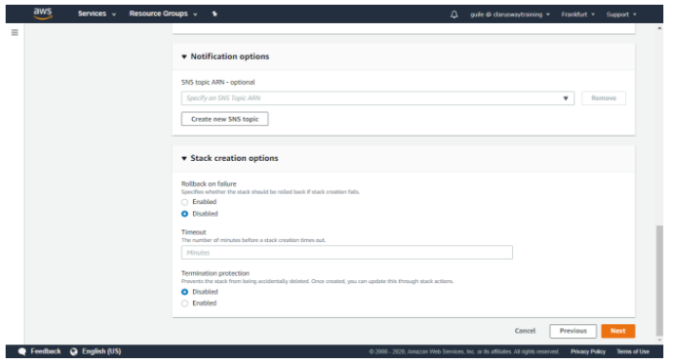


Notification options:

Here, you can specify an SNS (Simple Notification Service) topic ARN to get notifications. If you don't have one and want to create a new topic, click Create new SNS topic. For this demo we don't need any notifications.

Stack creation options:

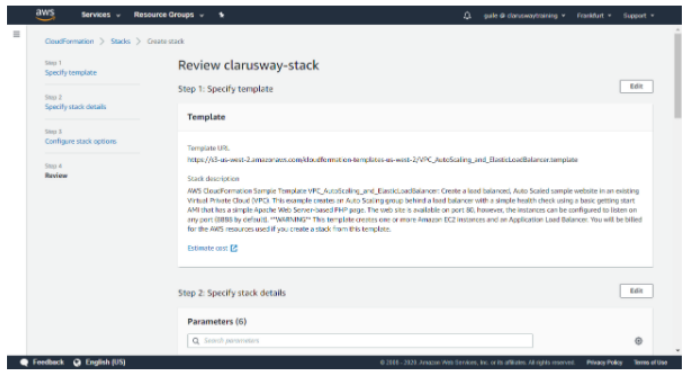
- Rollback on failure: Keeping the default, Enable, will let CloudFormation start rollback if the stack creation fails. A fail might happen due to a resources creation failure or a bug in your template. Let's select Disable.
- Timeout: If CloudFormation can not create the entire stack in the time (mins.) you specified here, it fails the stack creation due to timeout and rolls back the stack. Let's keep it as is.
- Termination protection: Prevents a stack from being accidently deleted. If a user attempts to delete a stack with termination protection enabled, the deletion fails and the stack remains unchanged. It's Disabled by default. Let's leave it default.



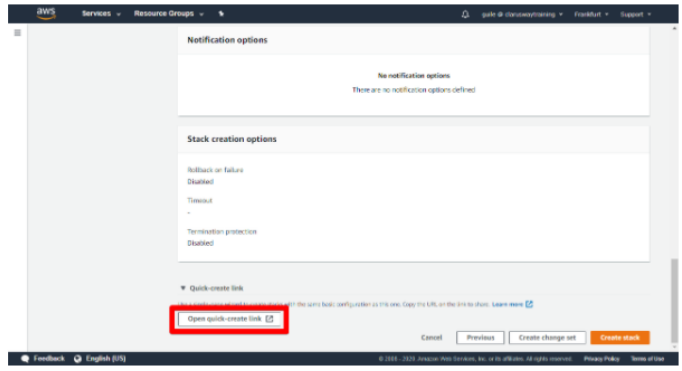
Now that this step is completed, let's click Next and move on to the next step.

Reviewing Stack Settings

As many other AWS services, here, you can review your configuration and edit your settings. Please, check your review page, and edit if you need.

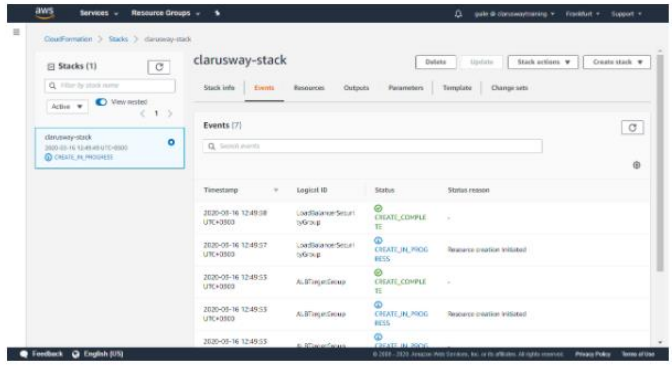


At the end of the review page, you will see a **Quick-create link**, click to expand it. Quick-create links enables you to get stacks up and running quickly from the AWS CloudFormation console. It simplifies the process of creating stacks by reducing the number of wizard pages and the amount of user input that's required. It also optimizes template reuse because you can create multiple URLs that specify different values for the same template.



If you click **Open quick-create link**, you will see a page similar to the review page. You can use this link to edit your parameters more rapidly. If you check the URL, it is a quite long one containing the template URL, stack name, and template parameters in URL query parameters to prepopulate a single Create Stack Wizard page. See quick-create links for more information.

If you are good with your stack review, click **Create stack**.

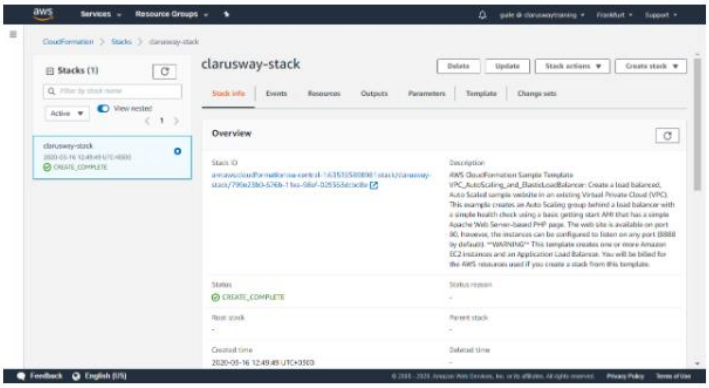


And CloudFormation starts creating stack. You will see the Events pane by default, so you can check the creation progress. Now, let's move on to the next lesson to see the details of the stack.

Monitoring Stack

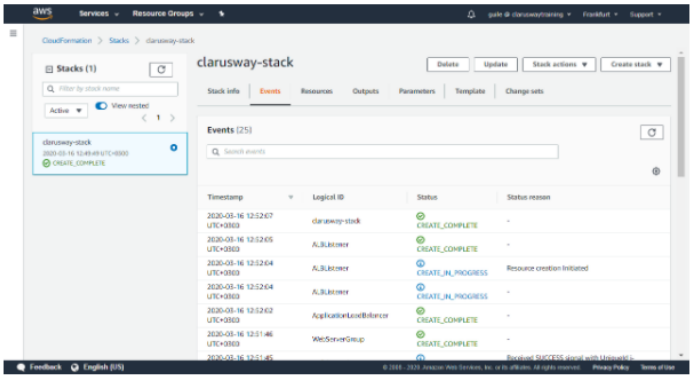
Displaying Stack Info

You can see an overview of your stack within Stack Info tab. You can check your Stack ID, description, status etc.



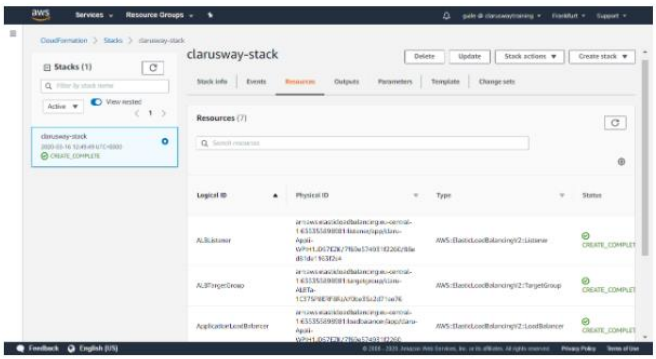
Displaying Stack Events

Here you can check the events regarding to your stack in timestamps.

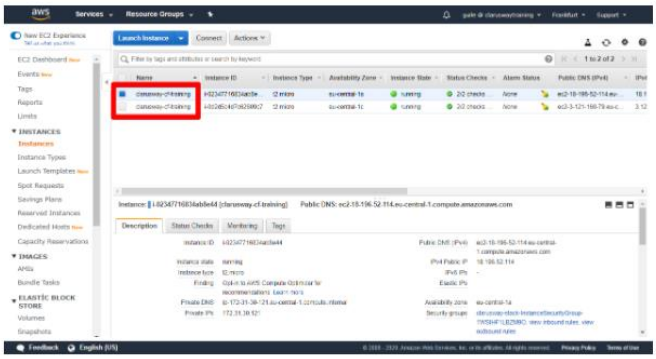


Viewing Resources

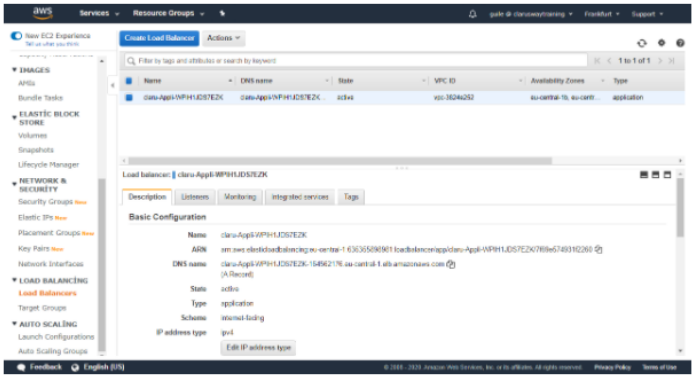
Here you can see your aws resources created in your stack. You can also see them in their own service windows.



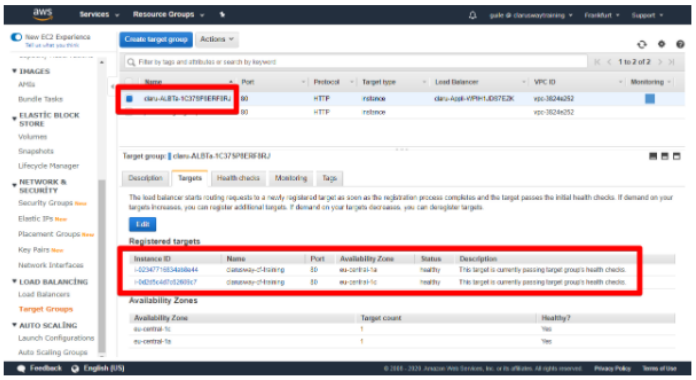
EC2 instances launched by CloudFormation:



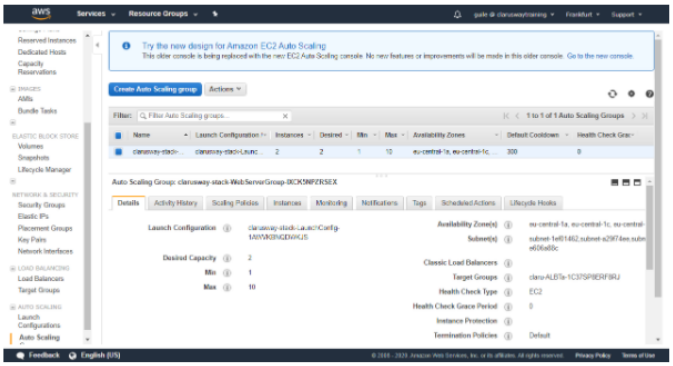
Load Balancer created by CloudFormation:



Target Group and registered instances:



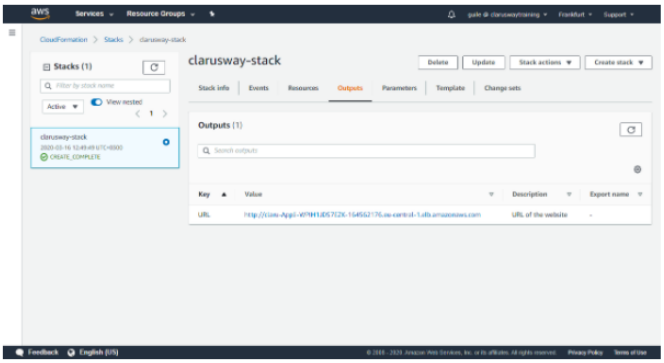
Auto Scaling Group:



Displaying Outputs

Here you can see the output of your stack which you defined in the template. If you check the "Outputs" section of the template, you can see we have defined a key "URL" which is the DNS name of our Application Load Balancer. So you can easily click and check your Web Page.

```
"Outputs" : {
    "URL" : {
      "Description" : "URL of the website",
      "Value" :  { "Fn::Join" : [ "", [ "http://", { "Fn::GetAtt" : [ "
    }
  }
```

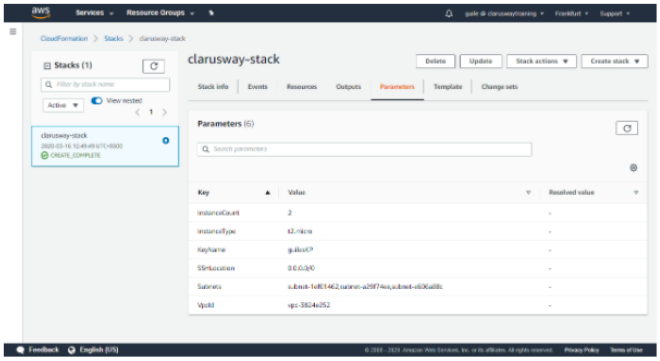


You have successfully launched your stack if you see this page.

Congratulations, you have successfully launched the AWS CloudFormation sample.

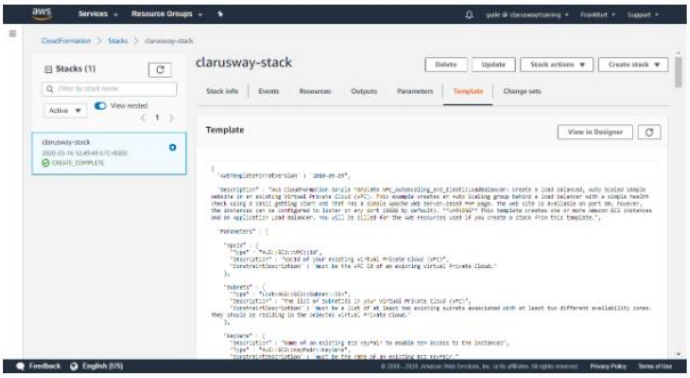
Displaying Parameters

Here you can check the parameters entered by the user at the beginning of the stack creation.



Viewing Template

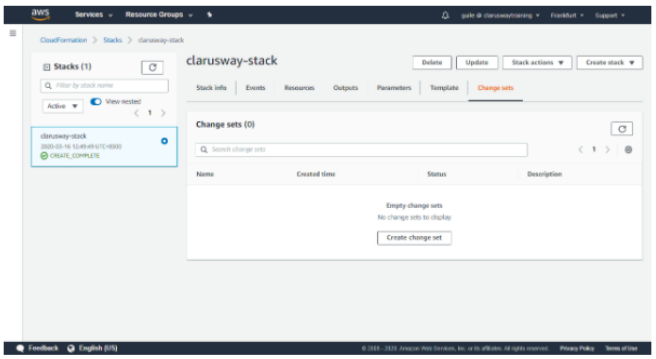
Here you can check the template which the stack has been created upon.



Change Sets

If you need to modify your stack, understanding how your changes will affect running resources before you implement them, you can create change sets. They allow you to preview how proposed changes to a stack might impact your running resources, for example, whether your changes will delete or replace any critical resources.

Since we didn't create any change sets, we don't have any.



Deleting Stack

When you are done with your stack, you can easily delete it. Deleting stack will delete all of the resources created by the CloudFormation.

If you are done with CloudFormation please delete your stack. Although all of the resources created in this tutorial are free tier eligible, you don't want unnecessary resources to consume your free tier usage.