CloudFormation

Automate the management of complex infrastructure and environments in a safe, reliable and repeatable manner.

Provisions AWS Infrastructure in a safe, repeatable, easy to maintain with less manual action required, and removes custom scripts

The Cloud significantly sped up provisioning hardware (aka servers) from weeks to minutes

The problem becomes… how to manage a lot of servers manually?

As we grow the number of our servers within the cloud environment, we need a reliable way not just to provision but also configure these steps in a repeatable and safe manner

The next natural progression from manual management is to automate using custom scripts. There were a lot of repeatable steps which evolved to a set of custom scripts to save them manual effort. As the company's use of the cloud matures the scripts take on more responsibilities. The scripts start to become very error prone. They become difficult to change and maintain especially across teams, and they are rarely made truly resuable.

The answer to this major issue is INFRASTRUCTURE AS CODE

Takes writing custom scripts to automate tasks to the next level.

Handles the following:

Configurations,

Provisioning,

Deployments

Utilizes tools/practices used in software/application development including:

Version Control,

Testing,

CI / CD (continuous integration and continuous deployment pipelines)

Benefits of using this tooling on infrastructure does the following:

Cost reduction, less time and effort

Speed (automation with fast execution and more visibility)

Less risk (less chance of human error)

Start with a template which can be seen as a blueprint of servers or infrastructure you would like to provision in their desired state

Create a template (can leverage code reviews, version control, and ci/cd practices to do so)

The template is run through some **tooling** which will pass the template and work out what needs to be done to the infrastructure to get it into the desired state. Such tooling includes CloudFormation, Terraform, Chef and Puppet.

The tooling will then make API calls to the target environment to provision the servers. The tooling also handles changing infrastructure. Which means it can calculate the delta between the current state and the desired end state. So the tooling enables you to focus on the template to reflect the desired end state rather than worrying about the custom procedural scripting, timing or calculating those delta changes. (To me a simple example of this would be if a server goes down then it will replace it since CloudFormation will make sure that the infrastructure returns to the desired state)

What is CloudFormation

-CloudFormation is an Infrastructure as Code tool for AWS

-CloudFormation is designed for AWS

-CloudFormation is a free* AWS Service (but you will be charged for the infrastructure that you provision and maintain)

-No additional charge (but you will pay for the underlying resources)

CloudFormation manages the lifecycle of your infrastructure in the cloud

1) Create - Creates AWS Infrastructure based on a template / blueprint
2) Update - Has the ability to update the AWS infrastructure as required in a predictable as repeatable fashion)
3) Delete - Easily delete / clean up any AWS Infrastructure you've created safely and reliably

CloudFormation templates are written in JSON or YAML format. They describe the desired end state of the infrastructure (the templates can be peer reviewed version controlled and used in integration and deployment tools.

This is uploaded to CloudFormation service via S3 which then kicks off CloudFormation orchestration. CloudFormation confirms the action it needs to take in provisioning, updating, or deleting your infrastructure. And then orchestrates the API calls to various services to get the job done.

CloudFormation has many benefits including:

-an ability to model and set up your AWS infrastructure using templates

-an ability to automate provisioning and configuring your infrastructure

-manages your dependencies between your resources

-enables you to easily control and track changes to the infrastructure

-provides the ability to rollback or delete cleanly after use

---

LAB 1

-A simple lab to introduce CloudFormation

-Create an EC2 Instance

-Instance Type: t2.micro

-AMI: Amazon Linux

We will update this Ec2 instance and follow it through to cleanup, so that you can see the entire lifecycle of this instance.

We are going to look at a simple example of a CloudFormation template provisioning an EC2 Instance. We are going to provision a t2.micro with an AMI of Amazon Linux

★★★Open up a text editor such as Visual Studio Code and create a file called SimpleExample.yaml

and then copy & paste in the below 6 lines of code and hit save:

```
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: t2.micro
```

ImageId: ami-02354e95b39ca8dec # Amazon Linux 2 AMI in N.Virginia

Our top level section is **Resources** on line one

The logical name for this resource is going to be **Ec2Instance** which is on line two

The **Type** we have defined on line three is AWS::EC2::Instance

We have defined the properties for the Ec2Instance on line five which is an **InstanceType** of t2.micro

And on line six we've got the **ImageID** (or AMI ID for Amazon Linux AMI in the N. Virginia region)

So the first thing we are going to do with this template is create our stack.

Let's go to the AWS management console.

★★★On the top right make sure you are in the N. Virginia region because we are going to be launching an AMI that corresponds to that region

Click **Services** and then click **CloudFormation** which is located underneath **Management & Governance**

Click **Create Stack**

Prepare Template: ✓Template is ready ← select Template is ready

Specify Template: ✓Upload a template file ← select Upload a template file

Click **Choose file**

Choose the **SimpleExample.yaml** file that you saved earlier ...(the one with the 6 lines of YAML code)

Then click **Next**

Stack name: ASimpleExample

Then click **Next**

This will direct us to the Configure stack options page

We can leave everything as default and simply click **Next**

This will direct us to the Review page

Just scroll down to the bottom of the page and click **Create stack**

At this point CloudFormation is going to pass the template that we have just uploaded and it is going to orchestrate the EC2 service to provision the t2.micro EC2 instance that we have requested.

Click on the Events tab and you will see that the Status for our stack says CREATE_IN_PROGRESS

Now click **Services** and click **EC2** and click **Running instances** and you will see the instance that we just provisioned.

Once the instance has successfully been launched and is running, click **Services** and then click **CloudFormation** and you will see that the status for our stack says CREATE_COMPLETE

(so our stack has been created)

Now click **Services** and click **EC2** and click **Running instances**

✓Select the EC2 instance that we created

You will notice the instance type is t2.micro which is what we specified in line 5 of our template.

Also, click the **Description** tab and scroll to where it displays the AMI ID which is ami-467ca739

That AMI ID matches the AMI ID on line 6 of the template that we created. Just refer to line 6 of our YAML code below and you will see it is a perfect match:

```
Resources:
 Ec2Instance:
   Type: AWS::EC2::Instance
   Properties:
     InstanceType: t2.micro
     ImageId: ami-02354e95b39ca8dec # Amazon Linux 2 AMI in N.Virginia
```

So we have provisioned an EC2 instance as defined by a template. Now let's update the stack. A simple update to this stack would be to give the EC2 instance a name tag. We will create a tag with the Key of **Name** and a Value of **Our Updated Instance**

Return to your text editor (such as Visual Studio Code) and open up the file that you created called SimpleExample.yaml

In your existing code add line 7, 8, and 9 per what you see below:

```
Resources:
 Ec2Instance:
   Type: AWS::EC2::Instance
   Properties:
     InstanceType: t2.micro
     ImageId: ami-02354e95b39ca8dec # Amazon Linux 2 AMI in N.Virginia
```

Tags:

- Key: Name

  Value: Our Updated Instance

★★★Now that you have added line 7, 8, and 9 (which gives our EC2 instance a tag), make sure to save the changes you made to the file.

Go back to the AWS console and click **Services** and click **CloudFormation**

✓Select the stack we created called ASimpleExample

Click the **Update** button

Prepare template: ✓Replace current template ←select Replace current template

Template source: ✓Upload a template file

Click **Choose file**

And choose **SimpleExample.yaml**

Then click **Next**

This will take you to the specify stack details page

Simply click **Next**

This will take you to the Configure stack options page

Simply scroll down and click **Next**

This will take you to the Review page

Simply scroll down and click **Update stack**

The CloudFormation service is now going to pass the template and figure out the delta based on what is existing and what we have requested and then orchestrate the changes required.

So CloudFormation will request EC2 to add a name tag to the existing EC2 instance. The EC2 instance that we have will **not** be removed and a new EC2 instance will **not** be created. The only thing that will be added is a tag.

Refresh the CloudFormation console after a minute or two and you will see that the Status of our update says UPDATE_COMPLETE

Now click **Services** and click **EC2** and click **Running instances**

You can see right away that our EC2 instance now has a Name tag that says **Our Updated Instance**

Now let's complete the lifecycle by returning to CloudFormation and removing the stack. So click **Services** and click **CloudFormation**

✓Select the stack we created called ASimpleExample

And then click the **Delete** button

The Status of our stack called ASimpleExample will say DELETE_IN_PROGRESS

Let's return to the EC2 console, so click **Services** and then click **EC2** and click **Running instances**

And in a minute or so hit the refresh button in the console. The instance state will say shutting-down and eventually will say terminated. Once it is terminated, let's return to the CloudFormation console. So click **Services** and then click **CloudFormation** and you will notice that our stack has now disappeared.

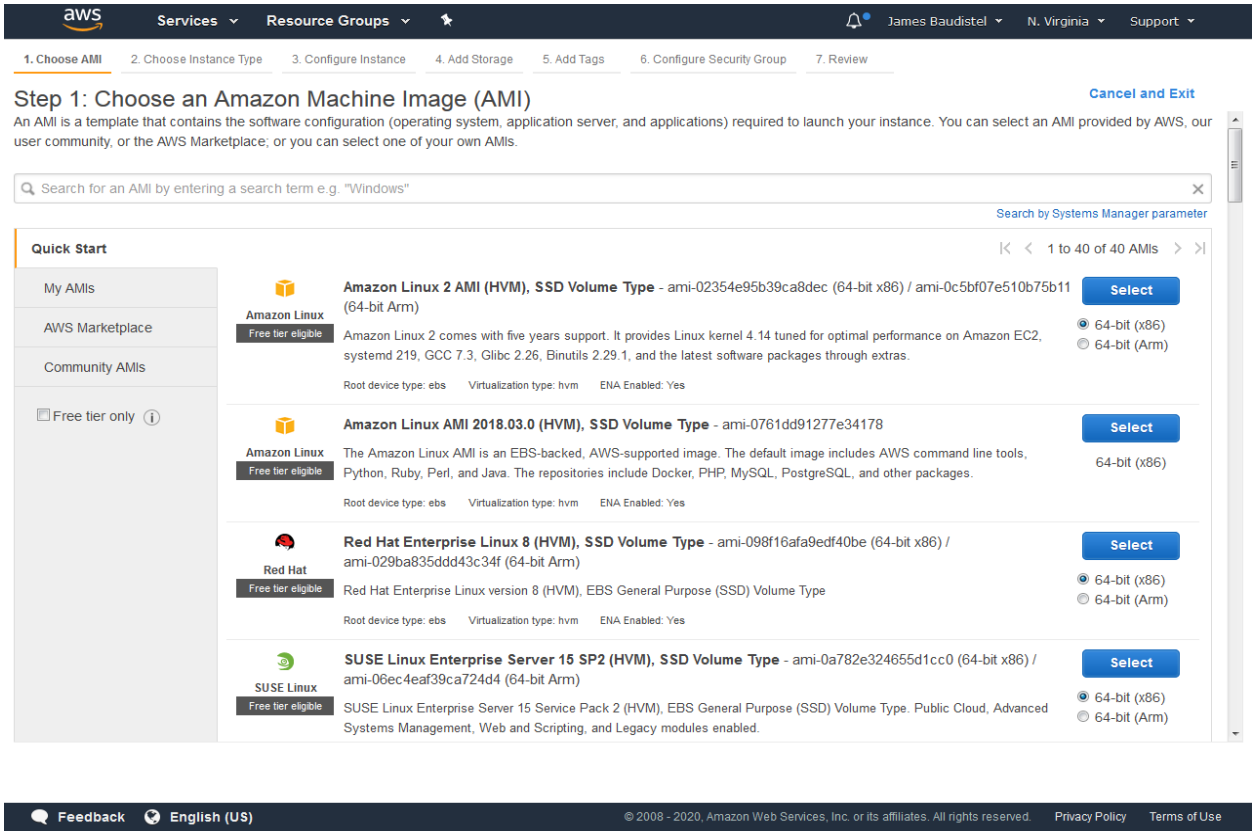
So in this tutorial we successfully created an EC2 instance using a template. We then updated the template, which gave our EC2 instance a name tag. Then we cleaned it all up by deleting the stack.

So what we have witnessed is a simple life cycle of a CloudFormation stack; creating, updating and deleting resources.

★★★Keep in mind that that AMI ID for an Amazon Linux AMI varies between regions. The easiest way to find the AMI ID for the region that you are working in is to go to the EC2 console. So click **Services** and then click **EC2**. Then click the **Instances** tab in the left margin. Then click **Launch Instance**

This will display a list of quick start AMIs. It is best to use the Amazon Linux 2 AMI which is at the top of the list (as it is Free tier eligible). The AMI ID is ami-02354e95b39ca8dec

★In the screenshot below you can see the AMI ID for the Amazon Linux 2 AMI after it says SSD Volume Type



Alternatively, instead of uploading a template file as we just did, we can create a template in the Designer. Let's do the same lab/tutorial as above, but we will use the Designer this time.

Go to the AWS console

Click **Services** then click **CloudFormation** then click **Create stack**

Prepare template: ✓Create template in Designer ← select Create template in Designer

Then click the **Create template in designer** button

Select the Parameters tab and then at the bottom of the screen click the **Template** tab

Choose template language: ✓ **YAML** ← select YAML instead of JSON

Remove what is currently in the template in replace it with the following 6 lines of code

Resources:

Ec2Instance:

Type: AWS::EC2::Instance

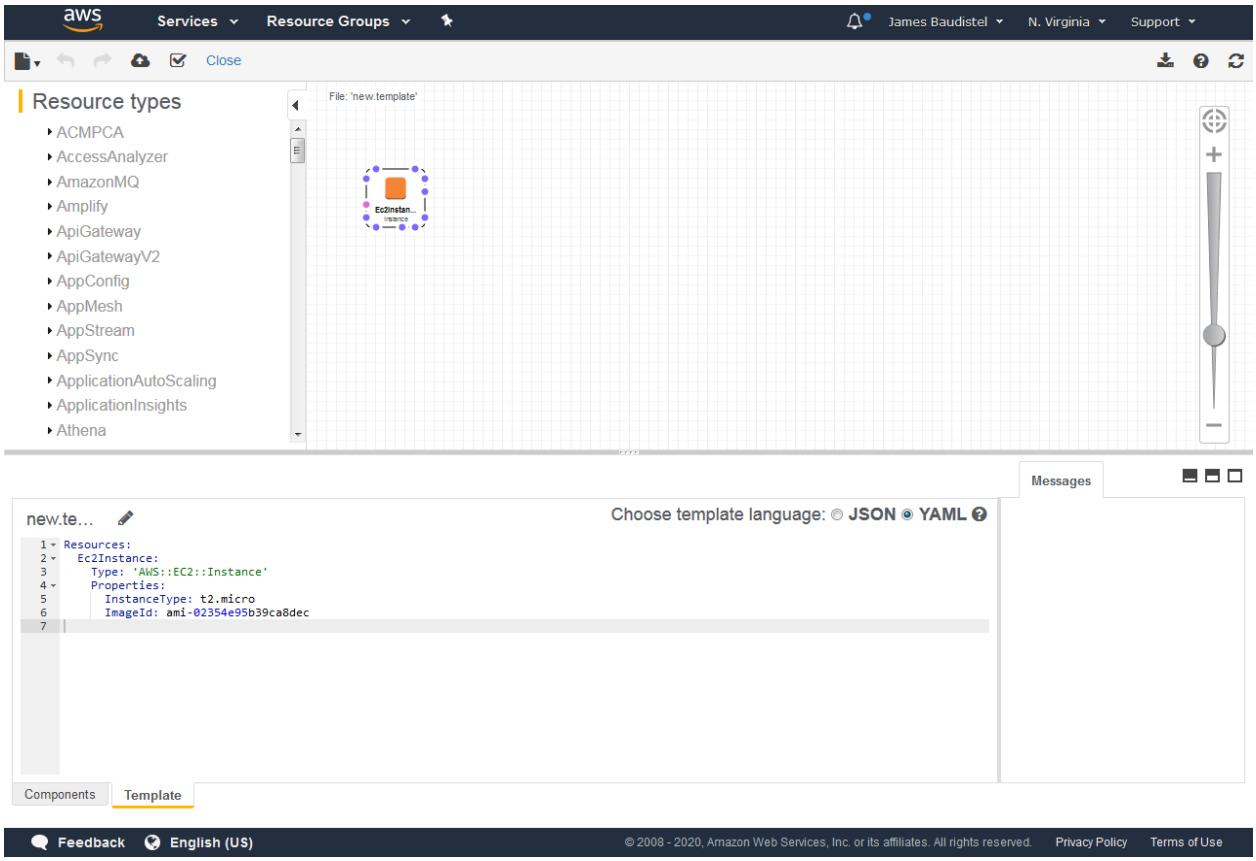
Properties:

InstanceType: t2.micro

ImageId: ami-02354e95b39ca8dec # Amazon Linux 2 AMI in N.Virginia

Once you have entered the 6 lines of YAML code click the **refresh diagram** icon near the upper-right corner of the page. Then click the **validate template** icon near the upper-left corner of the page. And then click the **create stack** icon which is located at the upper-left corner of the page.

See the screenshot below for clarification:



Once you have clicked the create stack icon (which is shaped like a cloud with an upward arrow on it)

You will be directed to the **Create stack** page

Prepare template: ✓Template is Ready ← will already be selected. Keep it selected.

Template Source: ✓Amazon S3 URL ← will already be selected. Keep it selected.

Click **Next**

Stack name: ASimpleStack ← give it a name like so

Click **Next**

This will direct you to the **Configure stack options** page

Just leave everything as is and click **Next** at the bottom of the screen

This will direct you to the **Review** page

Click **Create stack** at the bottom of the page

Status will say CREATE_IN_PROGRESS

Let's go to the EC2 console, so click **Services** then click **EC2** and then click **Running instances**

Here you will see that our EC2 instance is being created. Wait a minute or two for the instance to be in the **running** state.

Now let's update our instance by giving it a Name tag.

Click **Services** then click **CloudFormation**

✓ASimpleStack ← select our stack

Then click the **Update** button

Prepare template: ✓ Edit template in Designer ← select Edit template in Designer

Then click **View in Designer**

Click the **template tab** at the bottom of the page and edit the code in the box to include Tags for the EC2 instance. Altogether the code will look like so:

```
Resources:
  Ec2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      InstanceType: t2.micro
```

ImageId: ami-02354e95b39ca8dec

Tags:

- Key: Name

  Value: Our Updated Instance

Once you have edited the code click the **refresh diagram** icon at the top-right corner of the page. Then click the **Validate template** icon at the top-left corner of the page. Then click the **Create stack** icon at the tope-left corner of the page.

This will direct you to the Update stack page. Leave everything as is. And click **Next.**

This will direct you to the Specify Stack Details page. Leave everything as is. And click **Next.**

This will direct you to the Configure Stack options page. Leave everything as is. And click **Next.**

This will direct you to the Review page. Click the **Update Stack** button at the bottom of the page.

Now let's check the EC2 console to see if our EC2 instance now has a name tag.

So click **Services**, then click **EC2**, and then click **Running instances.**

As you will see the name of our EC2 instance now says **Our Updated Instance.**

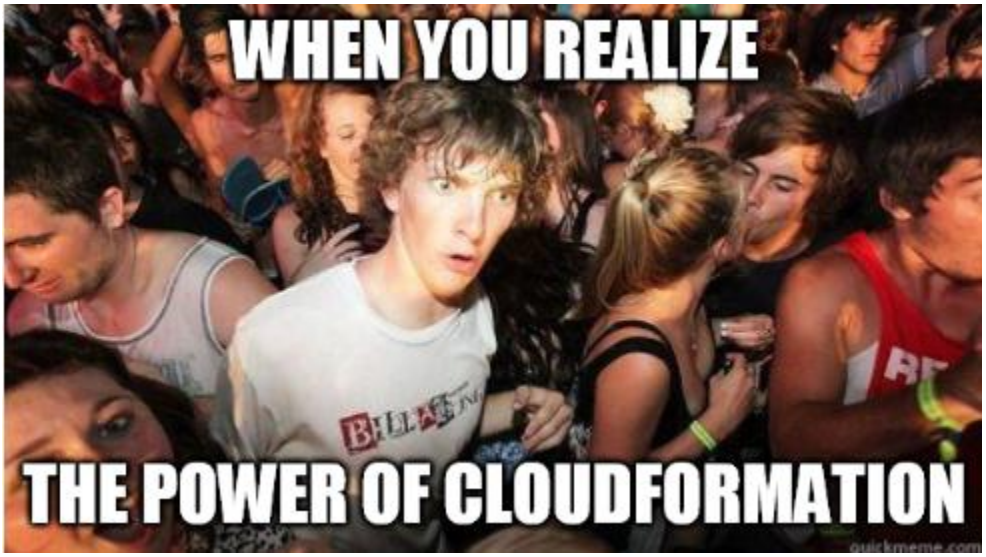
So the update was successful. Let's clean up.

Click **Services**. Click **CloudFormation.**

✓ASimpleStack ← select our stack

Then click the **Delete** button

Then click **Delete stack** in order to confirm that you want to delete the stack.


WHEN YOU REALIZE
THE POWER OF CLOUDFORMATION
quickmeme.com