AWS CloudFormation notes

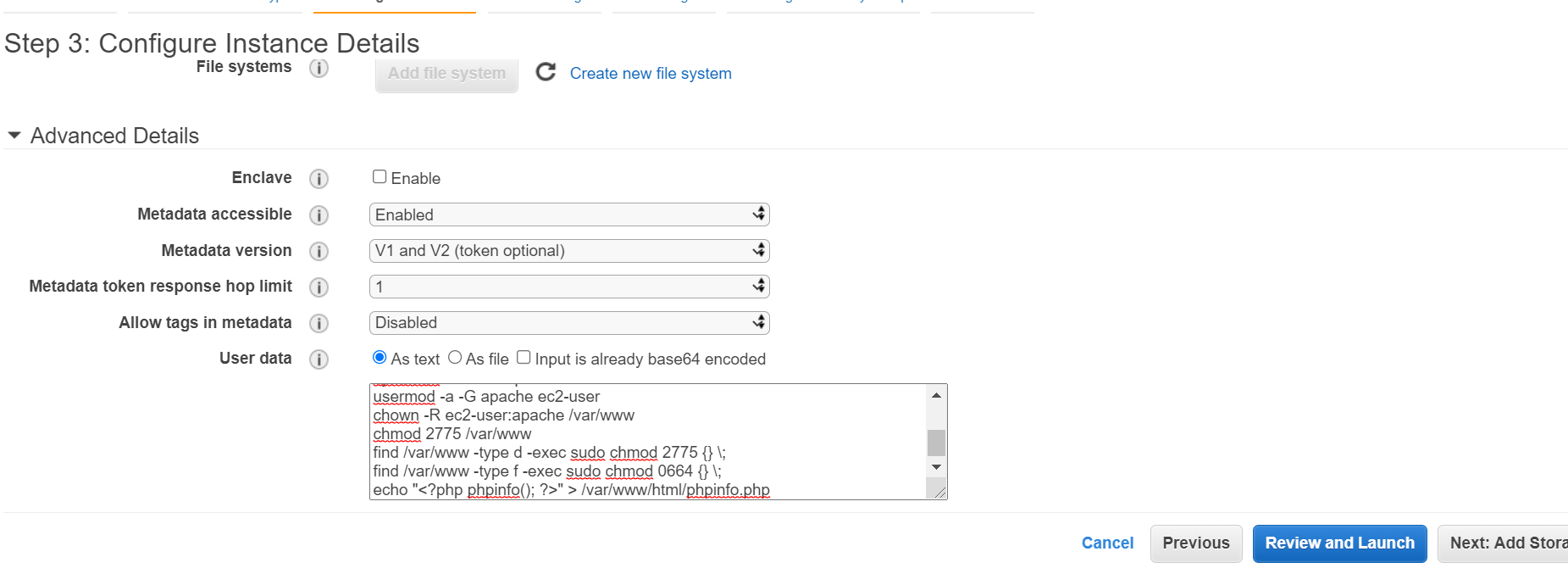
**EC2 User data:**

Ec2 user data is set of commands we mentioned while provisioning the EC2 instance so that while instance get crated this script will auto run so that what we want to configure it get configured automatically right after the EC2 up and running.

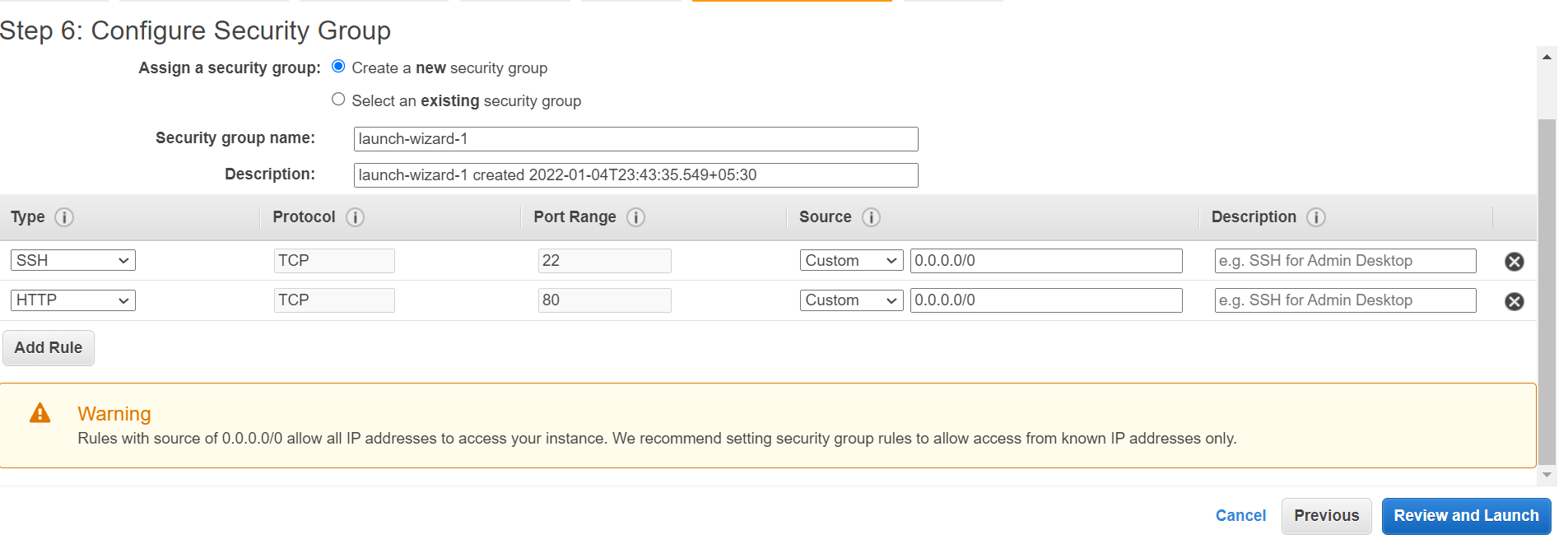
Task : Create a EC2 and Setup a Web server (Using User data)

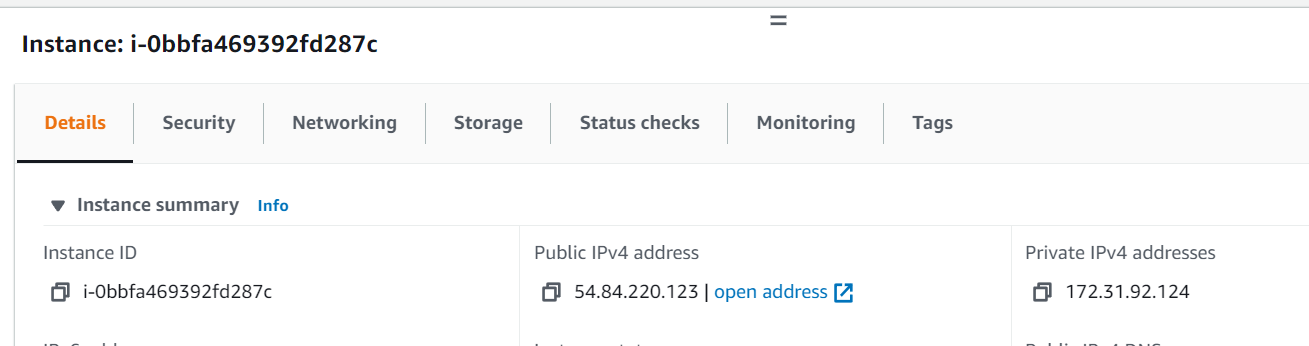
Here we are going to create an EC2 and configure it as a web server using EC2 user data approach manually

Create an EC2 from Console use 0-ec2-user-data.sh userdata

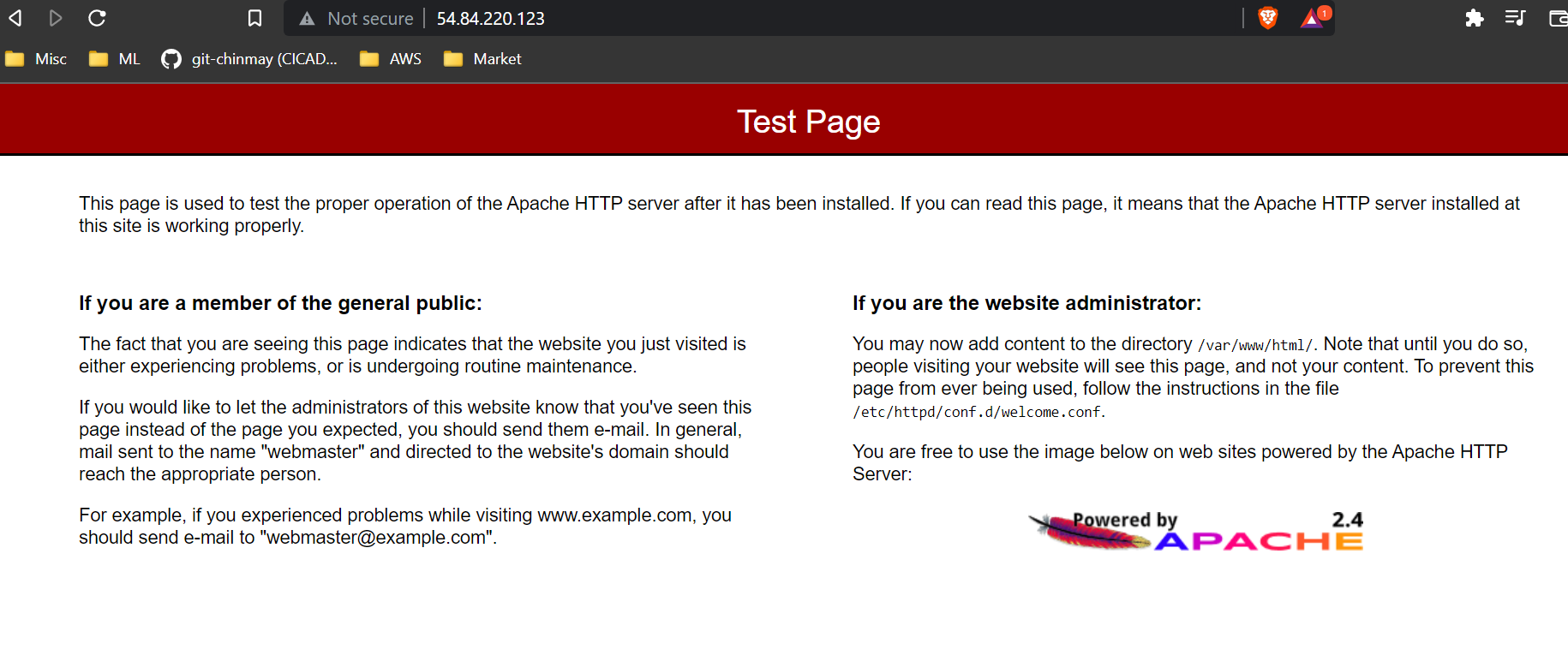


Configure security

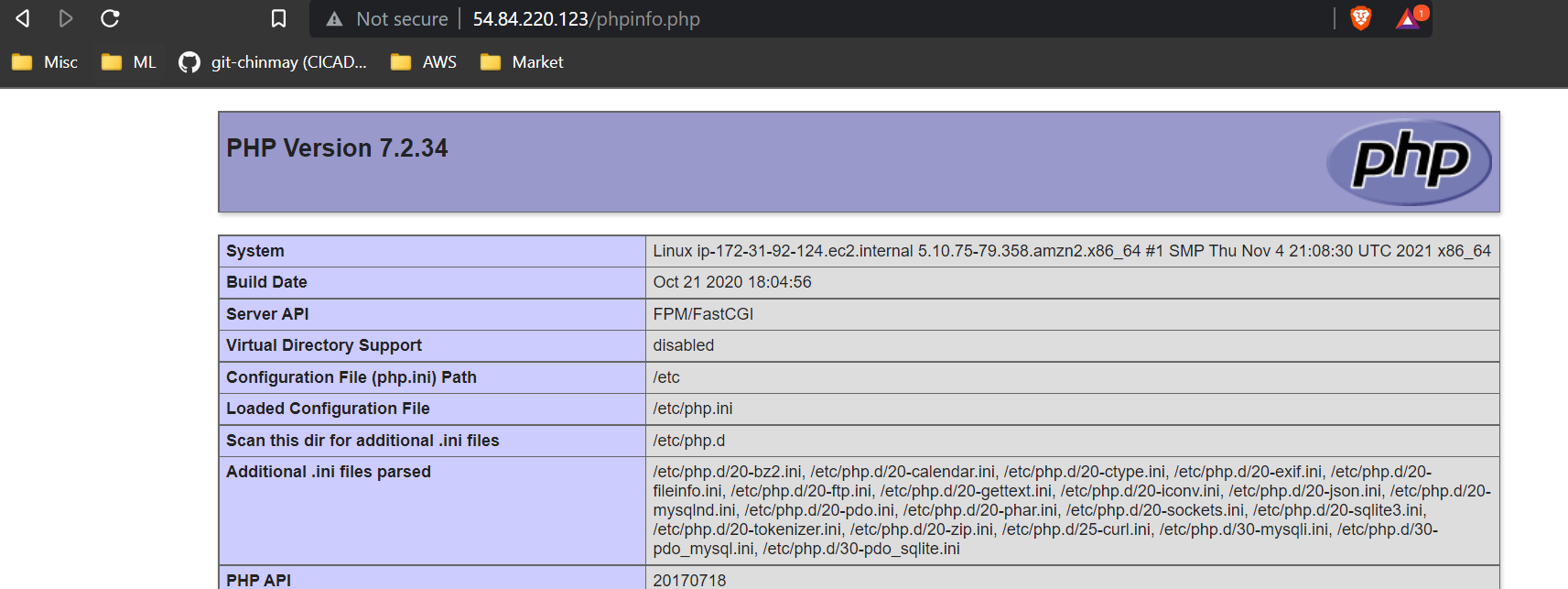
Open the public IP in browser. It will take time to get configured so have some patients (Copy the IP, don’t use the open address option)



Webpage is ready.

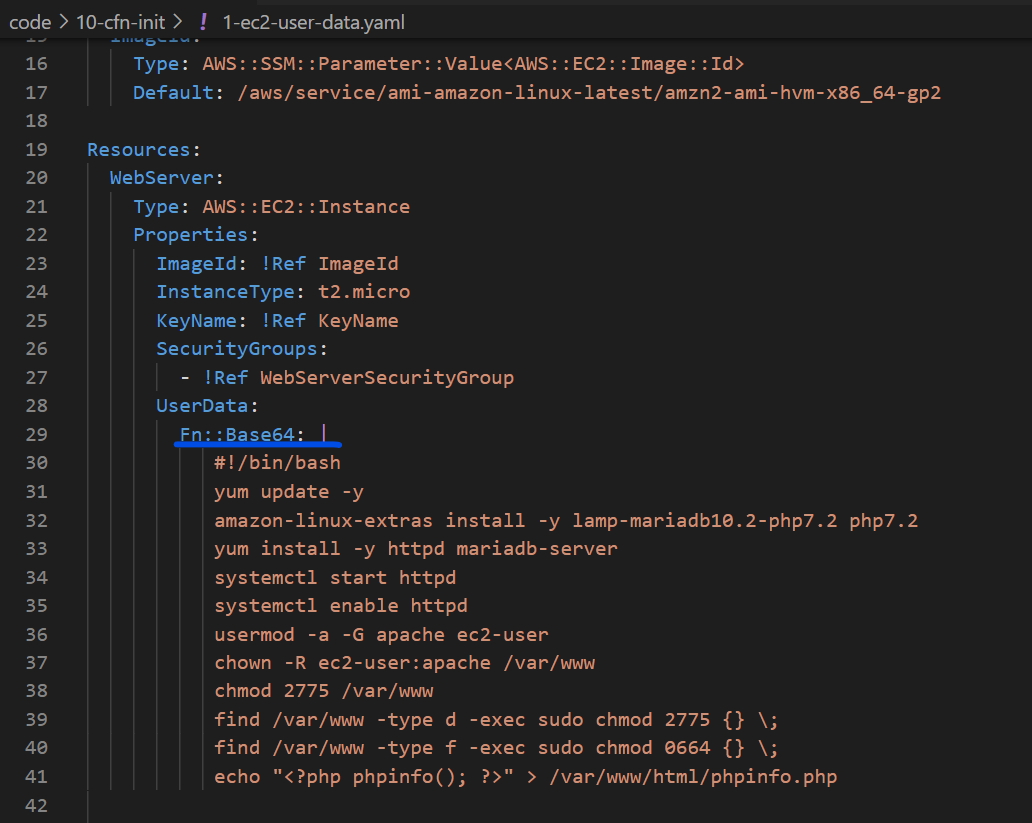


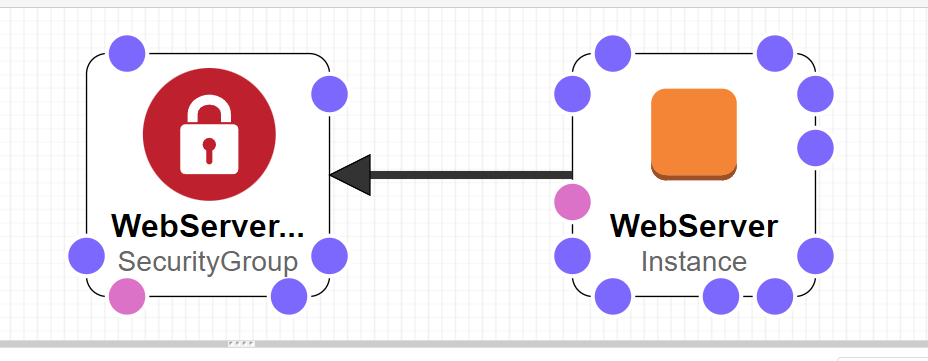
Php also running



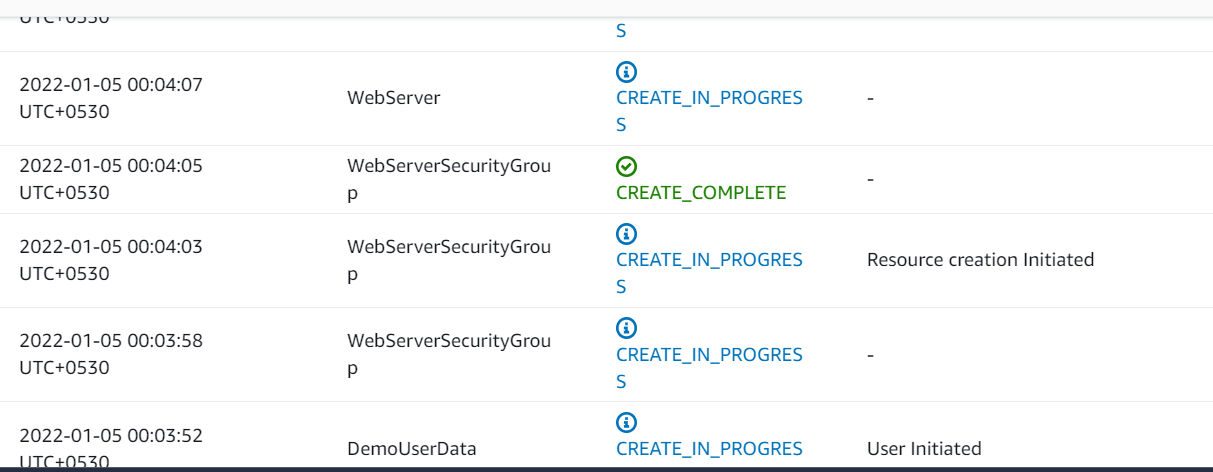
We can perform all above steps using CloudFormation Template

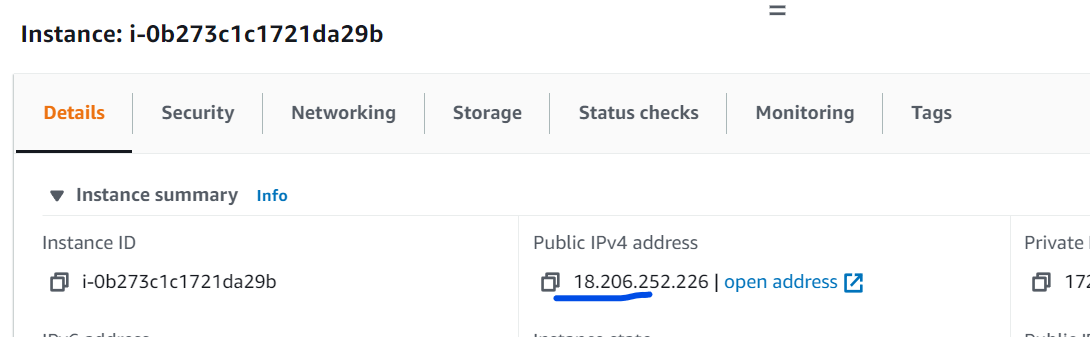
Important function is to notice use of Fn::Base64 to declare the userdata script

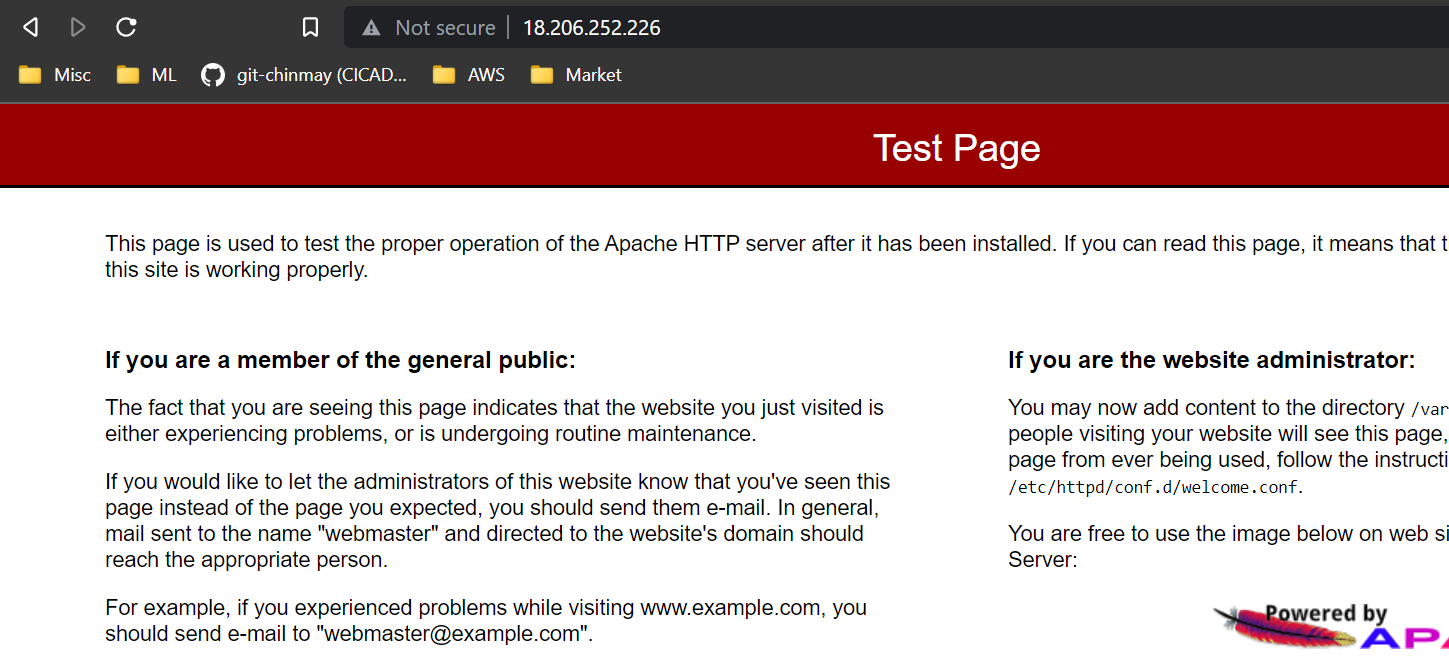




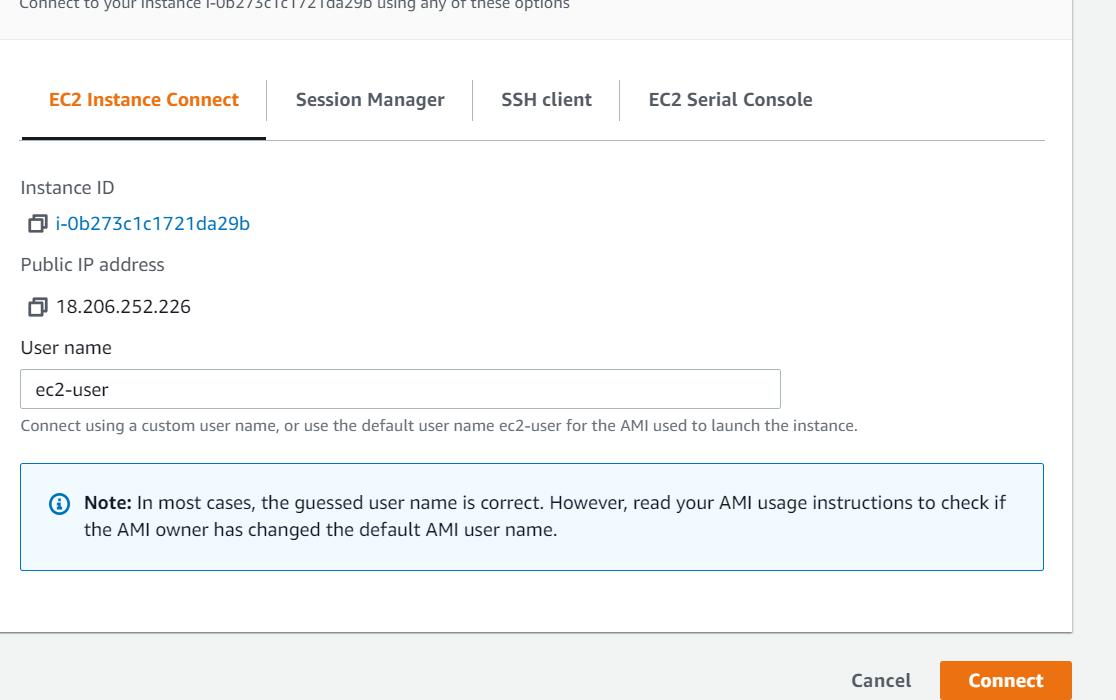
Create the stack

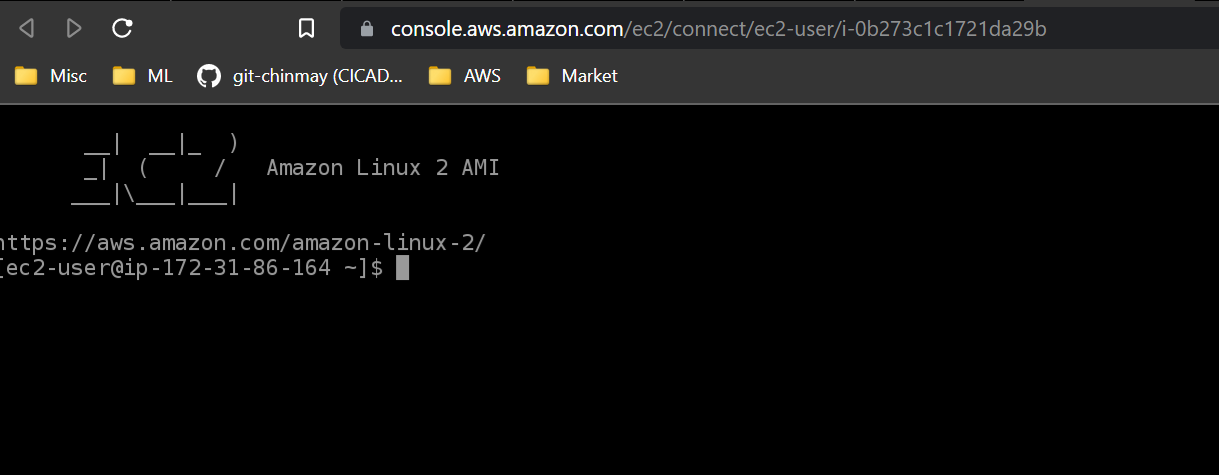




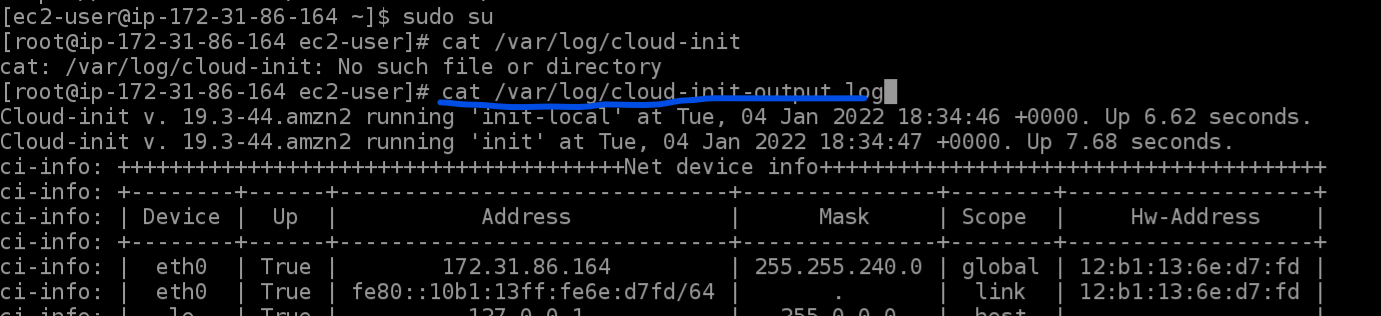


If we connect to the EC2





If we want to see what are the commands runs we can see inside the EC2 logs



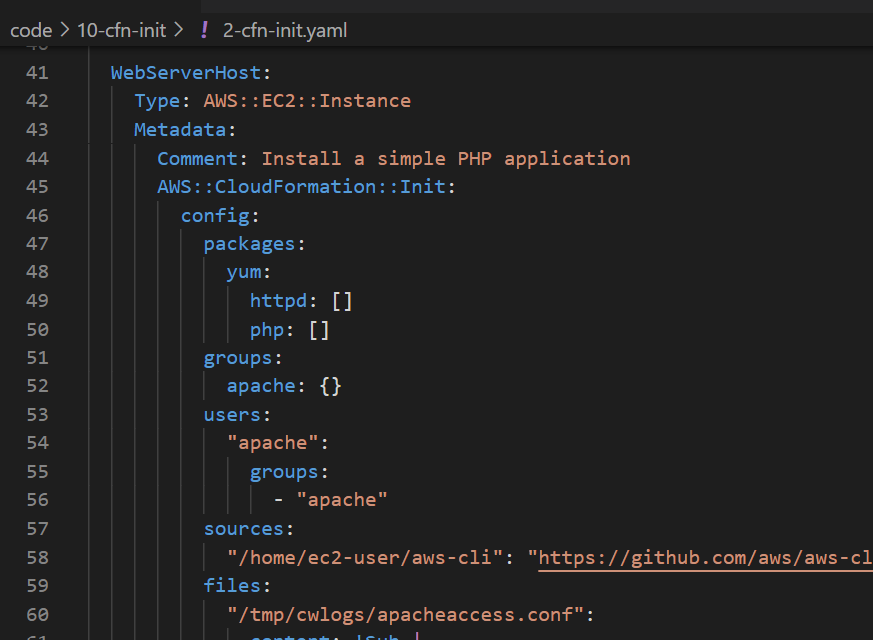
**CloudFormation helper Script:**

There are 4 python scripts come default in Amazon AMI Linux 2, we can also install them using yum on non Amazon Linux2 machines.

* cfn-init : Used to retrieve and interpret the resource metadata, installing packages, creating files and starting services.
* cfn-signal : wrapper script used by resources to talk back to CloudFormation for example to say resource creating was successful or not
* Cfn-get-metadata : A wrapper script to retrieve all metadata of a resource or path
* Cfn-hup : A daemon to check the updates to metadata and run custom hooks if foun any update.

**Cfn-init :**

Init structure is like below



Packages: used to download and install packages

Groups: Define user groups

Users: Define users and which group they belong to

Sources : Downlaod or archive files inside the Ec2

Files: Created files inside the EC2 using inline or can be pulled from a URL

Commands : run a seies of commands

Services : Run series of services using sysvinit

**AWS::CloudFormation::Authentication:**

Used to specify authentication credentials for files and sources in AWS::CloudFormation::Init

Two types:

* Basic : When the source is a URL
* S3 : When the source is an S3 bucket

**Function Fn::Sub : (!Sub)**

* Substitute function. Substitute a variable from a text.
* We can use it combing with Reference and Pseudo variables.
* String must contain ${<variable name>}

**Services:**

* Launch a bunch of services at EC2 instance launch.
* 
* It ensures services are started when file changed or package are updated.

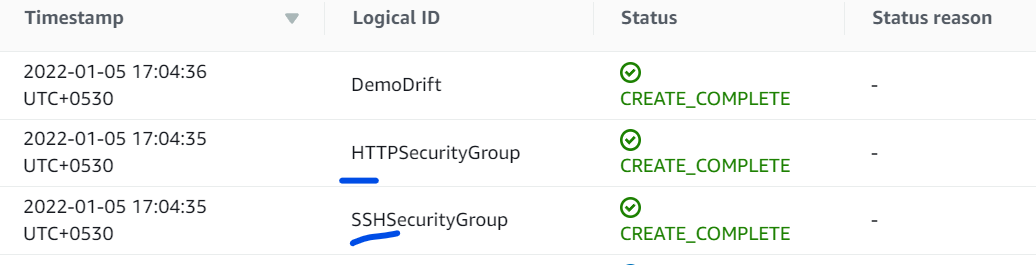
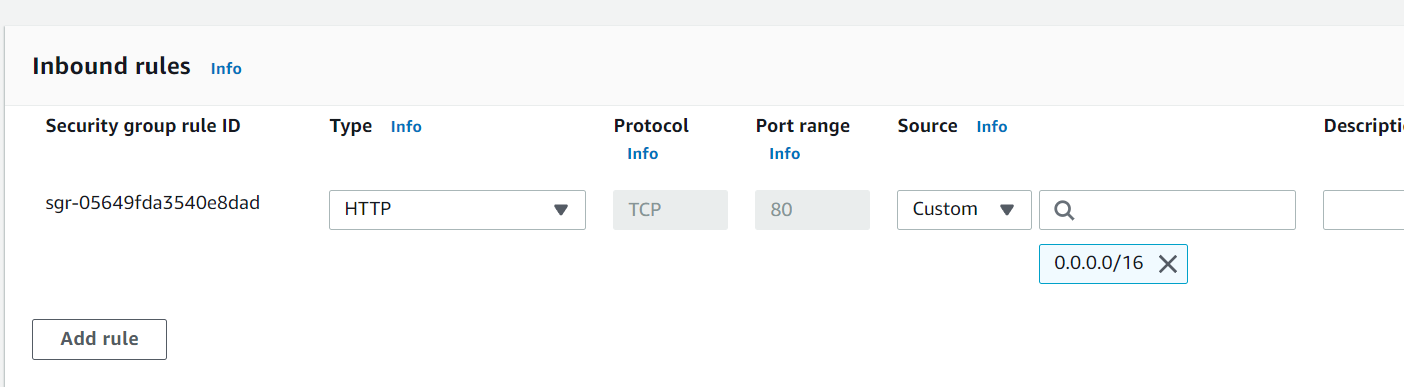
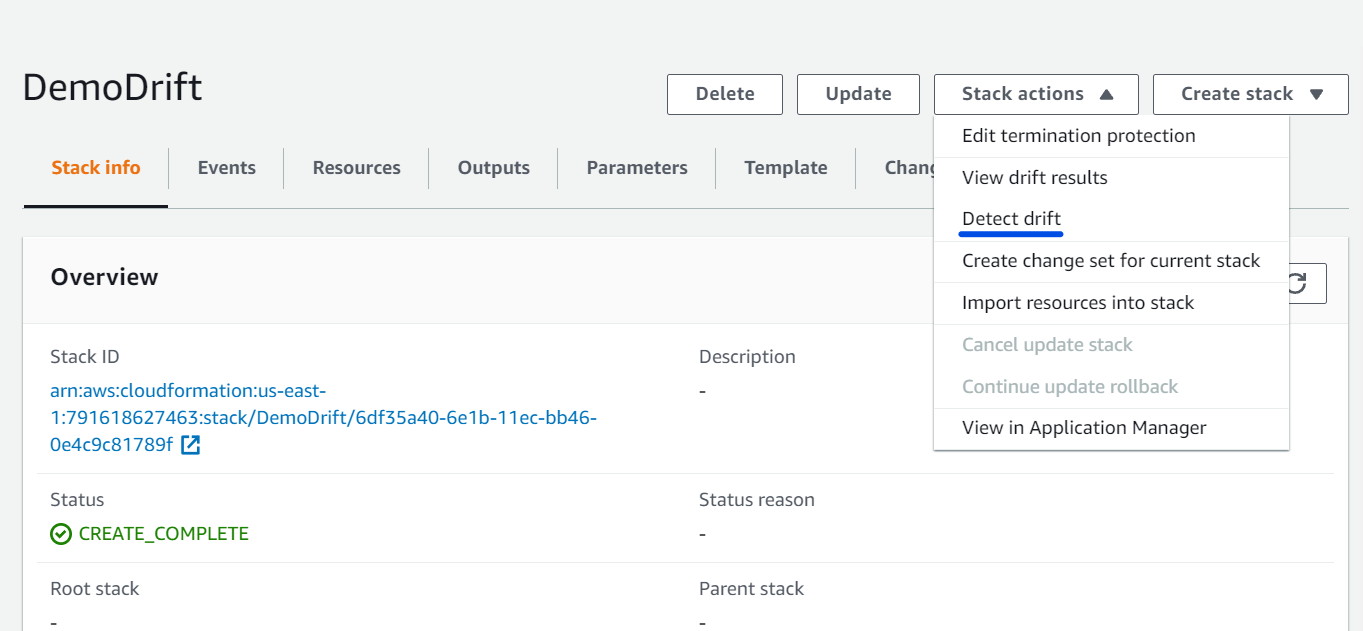
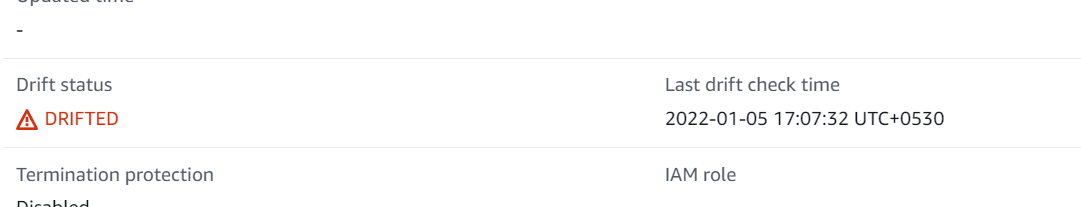
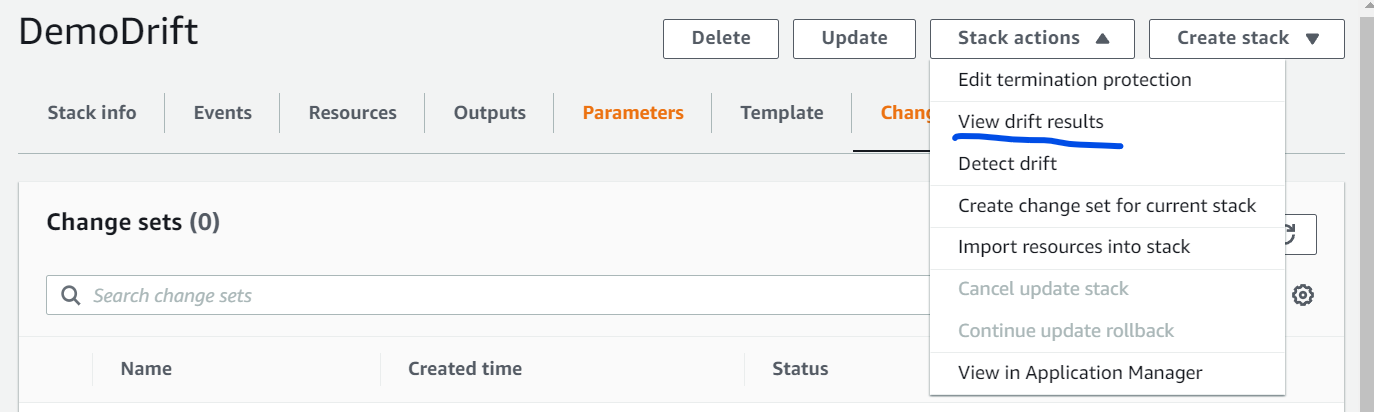
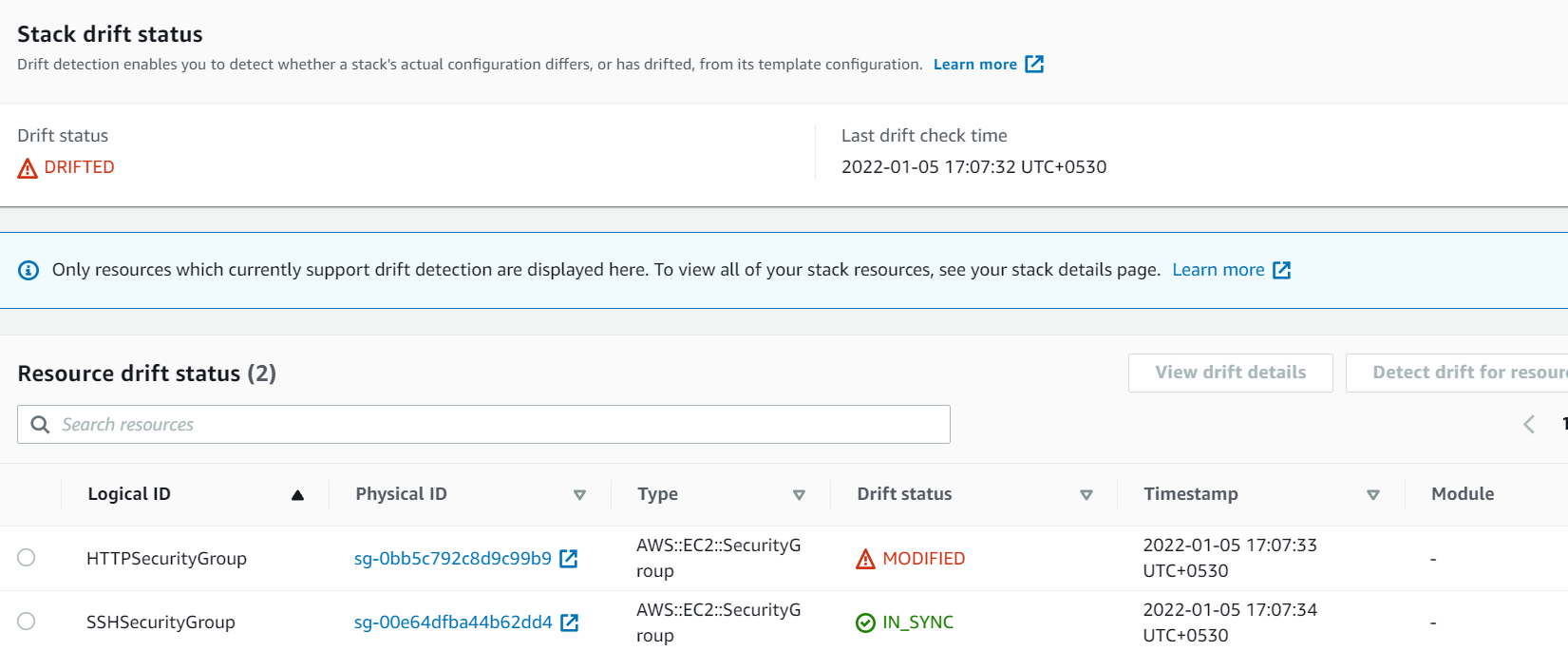
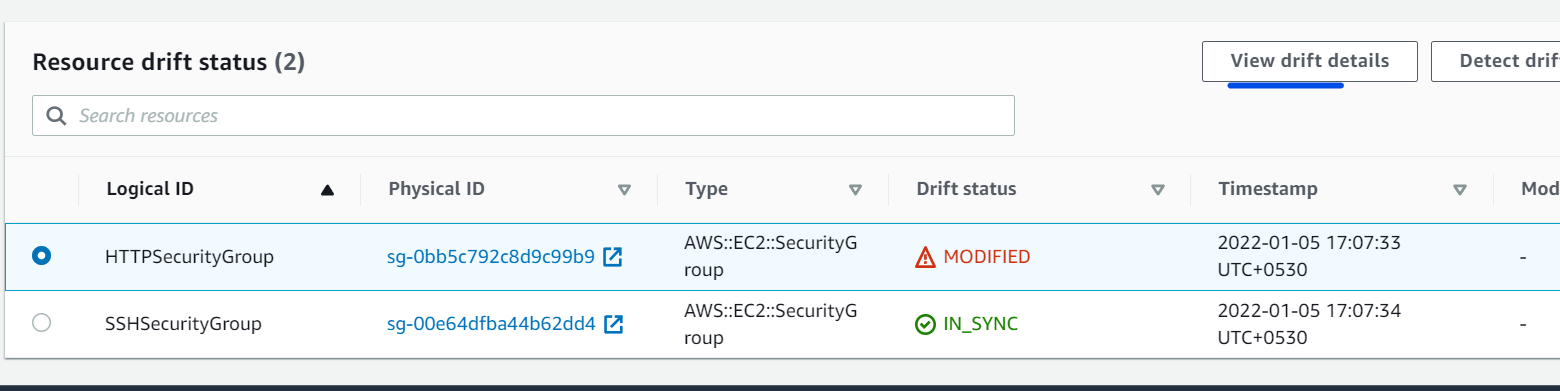
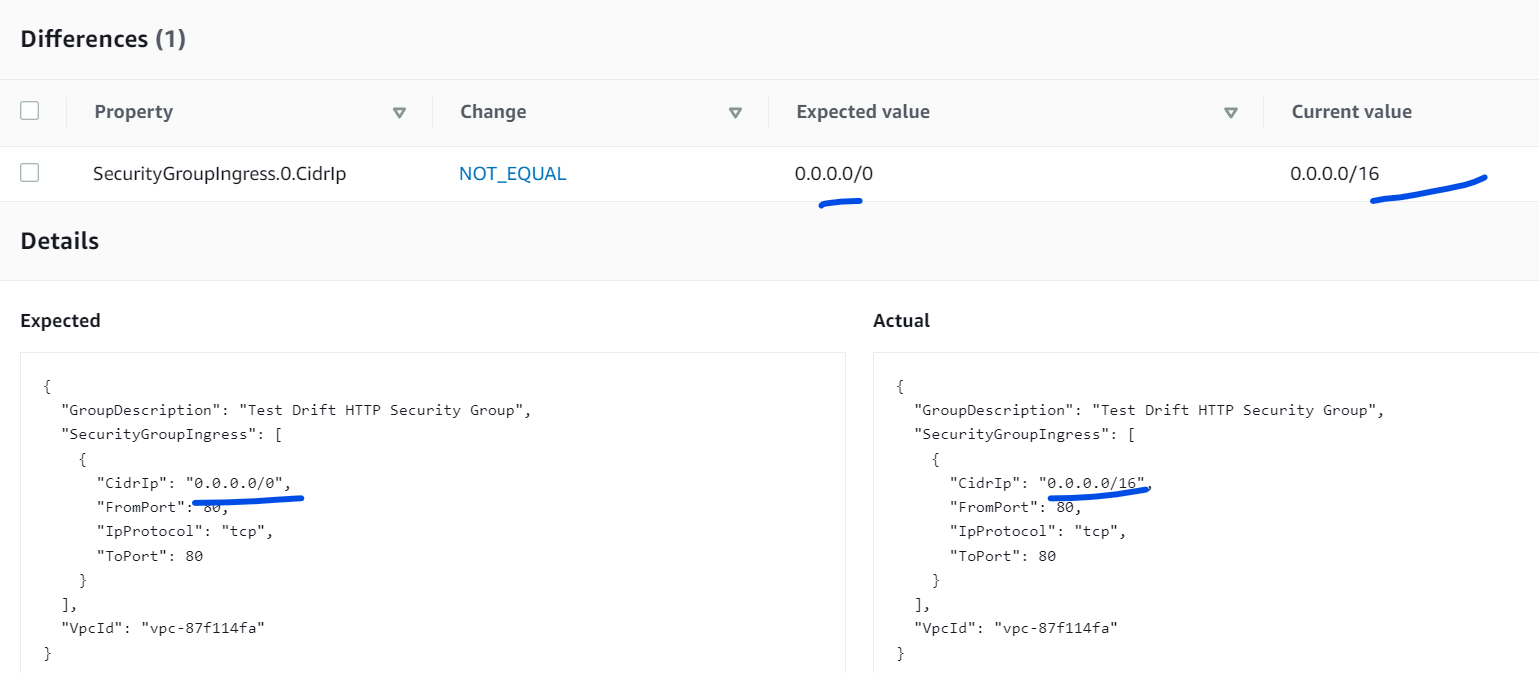
**User Data vs CloudFormation::Init vs Helper Scripts**

* EC2 User data is an imperative way to provision/bootstrap the EC2 instance using Shell syntax
* AWS::CloudFormation::Init is a declarative way to provision/bootstrap the EC2 instance using YAML or JSON syntax
* AWS::CloudFormation::Init is useless if it’s NOT triggered by a script within the EC2 User Data
* Triggering AWS::CloudFormation::Init inside EC2 User Data is done by using cfn-init or cfn-hup

**CloudFormation Drift:**

* By using template, we are able to build the infract structure but nothing prevents individual users to modify the configurations.
* CloudFormation Drift compares the entire stack or individual resources in it and shows if any difference is there with respect to original template.

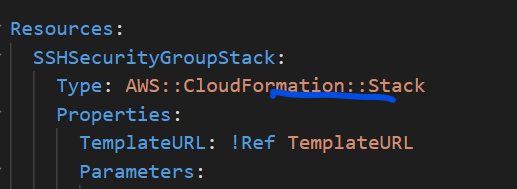
**Task: Create a stack and modify some configuration and detect the changes using Drift**

* Create the Stack using drift-security-group.yml template
* 
* Now modify one of the Security Group
* Modified the inbound rule
* 
* Now run the Drift
* 
* 
* 
* 
* 
* 

**Nested Stacks:**

Stacks that are part of another Stack. Nested stacks are not sharable.

As soon as ou saw below type for a resource understood that that it’s a nested stack



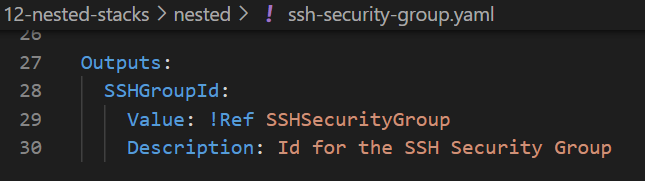
**Task: Nested stack demo**

We will have two stacks

* Stack-1 = Will create a Security group
* Stack-2 = Will create an EC2 instance

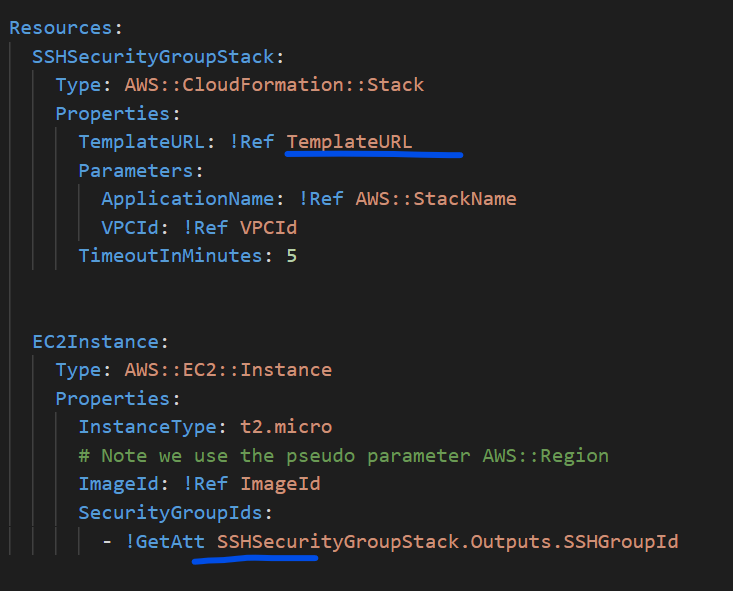
But we will use stack-1 for assigning the Security group to Stack-2 EC2 creations. We will basically add Output to expose the SG ids and will use it in Stack-2.

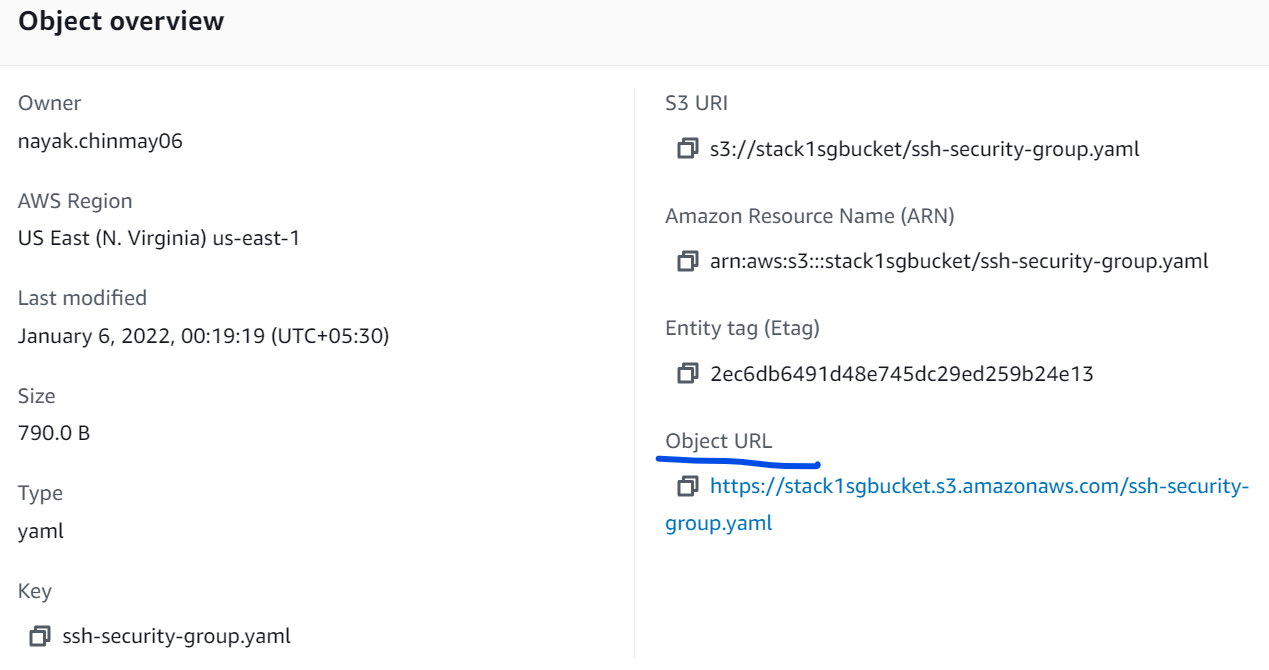
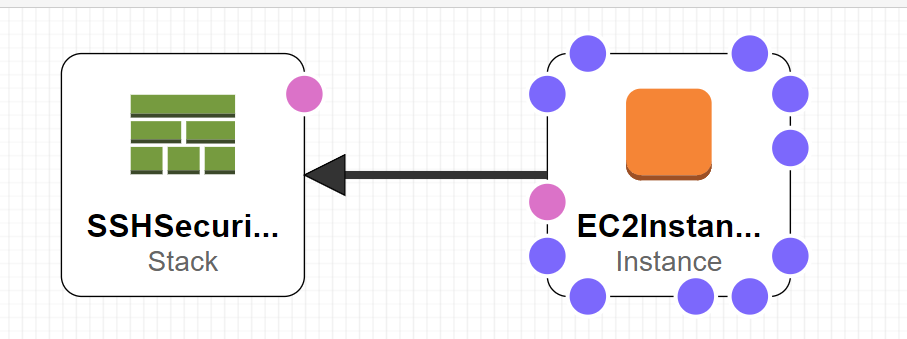
Stack-1



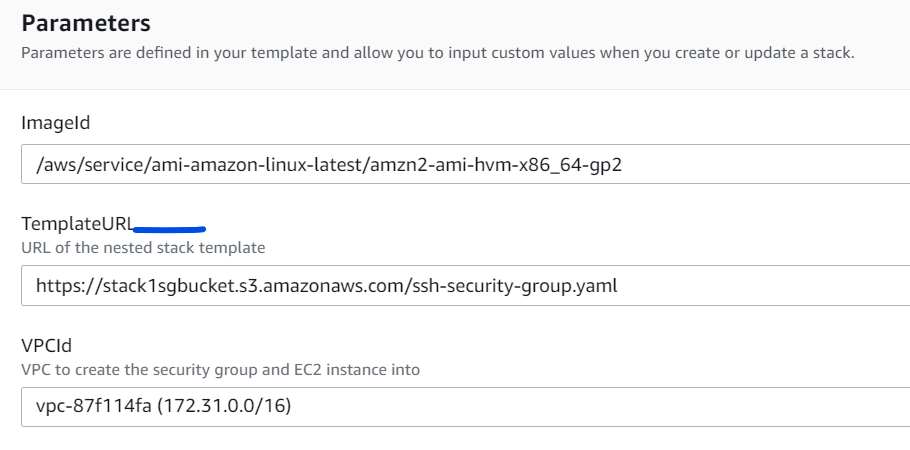
Stack-2

* Template URL we will get from Stack-1 properties
* Securitygroupids we are fetching using the GetAtr function.

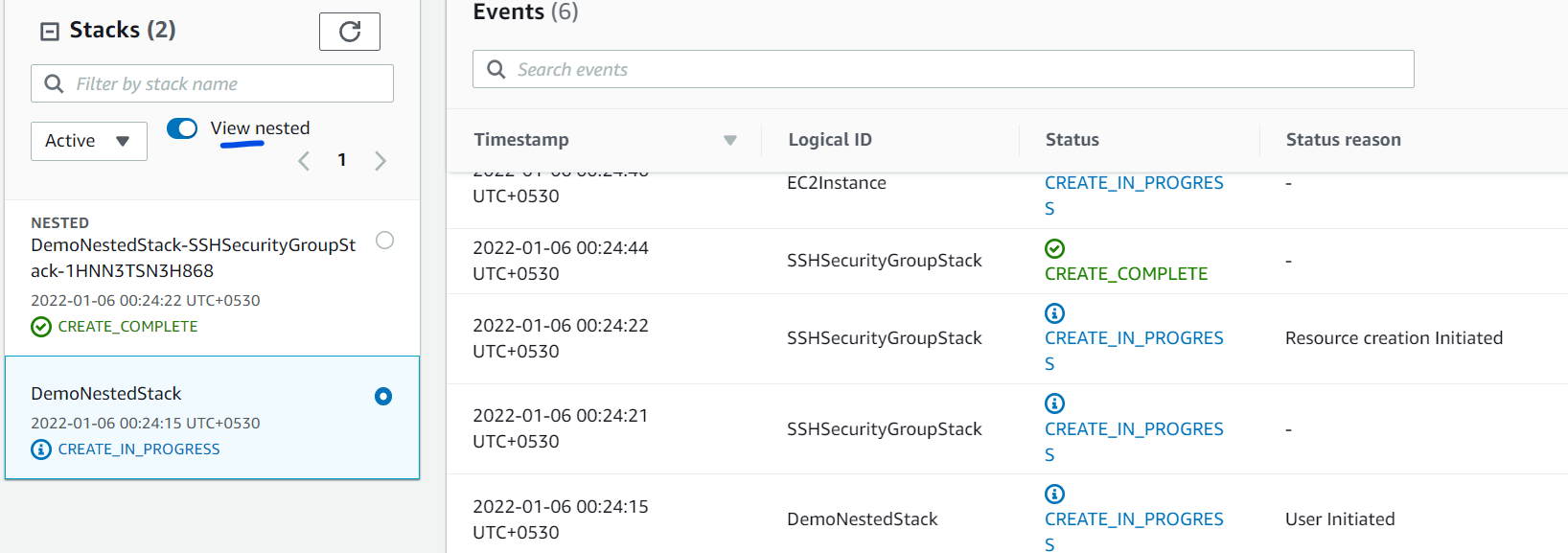


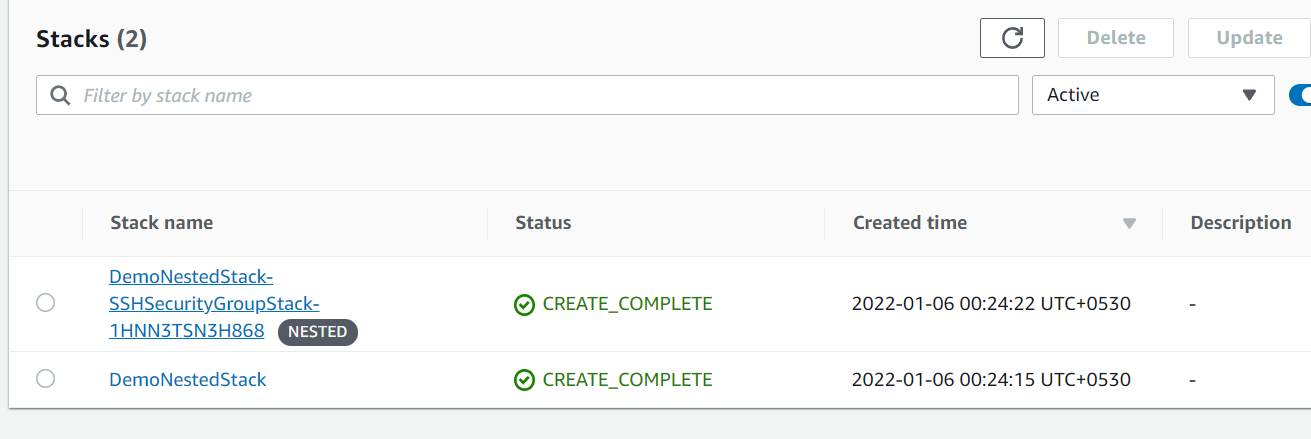
* First put the Stack-1 template into a S3 bucket so tat we get a template URL
* Copy the object URI
* 
* <https://stack1sgbucket.s3.amazonaws.com/ssh-security-group.yaml>
* Now create the Stck-2 on CloudFormation
* 

Fill the template url parameter



We will observe a change in left side

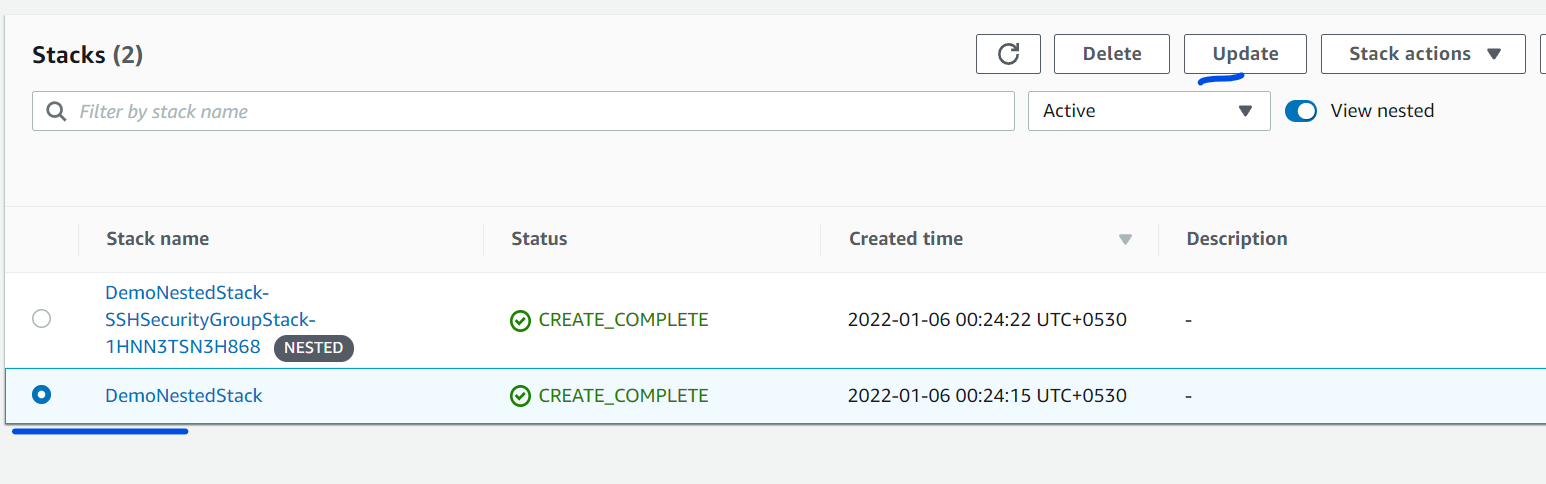




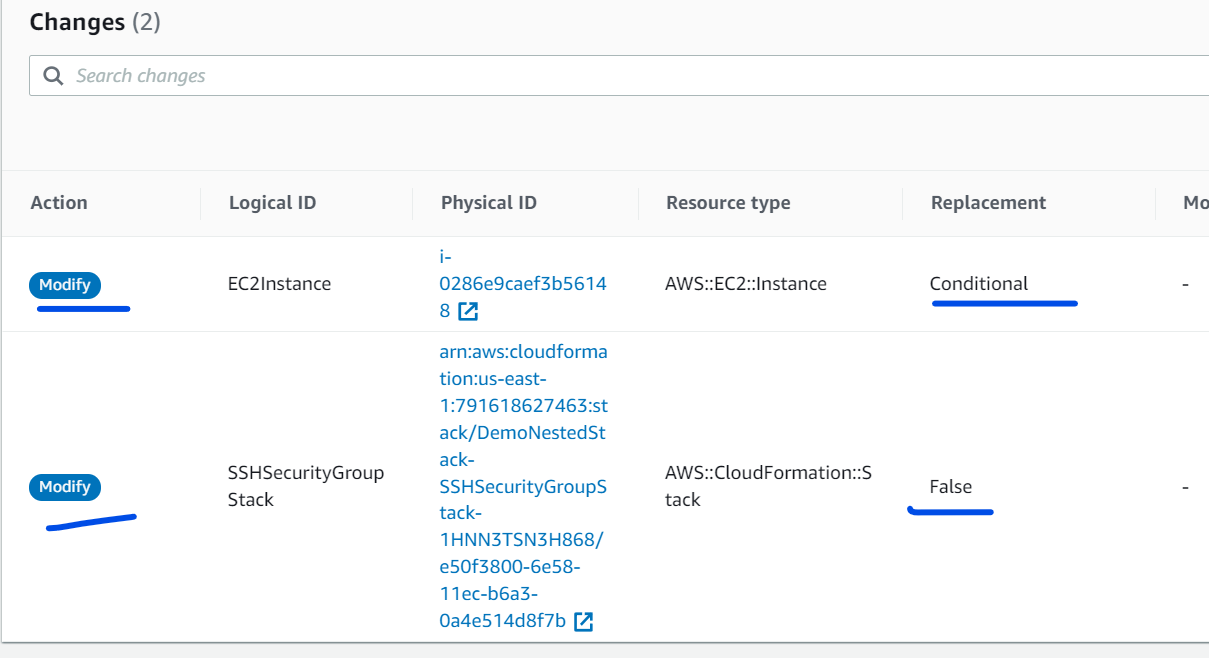
If you update the child template then we have to redo the same steps

Upload the updated template to S3

Come to the CloudFormation and update only the Root/Parent Stack

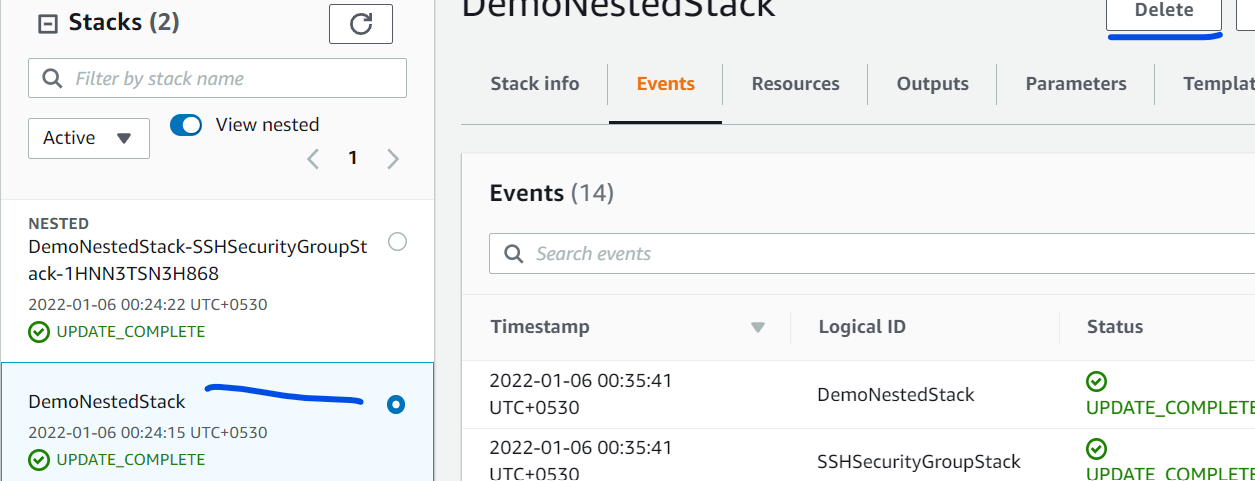


Although we are not changing anything on root template still its showing modified bcz it may be affected due to change in child template.



Deleting a Nested Stack:

Never ever delete the child/nested stack. Always delete the Root/Parent stack.



Exported Stack Output Values vs. Using Nested Stacks:

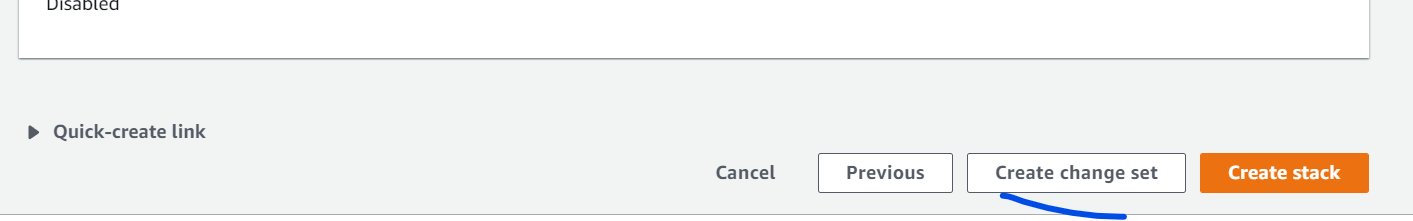
* If you have a central resource that is shared between many different other stacks, use Exported Stack Output Values
* If you need other stacks to be updated right away if a central resource is updated, use Exported Stack Output Values
* If the resources can be dedicated to one stack only and must be re-usable pieces of code, use Nested Stacks
* Note that you will need to update each Root stack manually in case of Nested stack updated

**CloudFormation StackSets:**

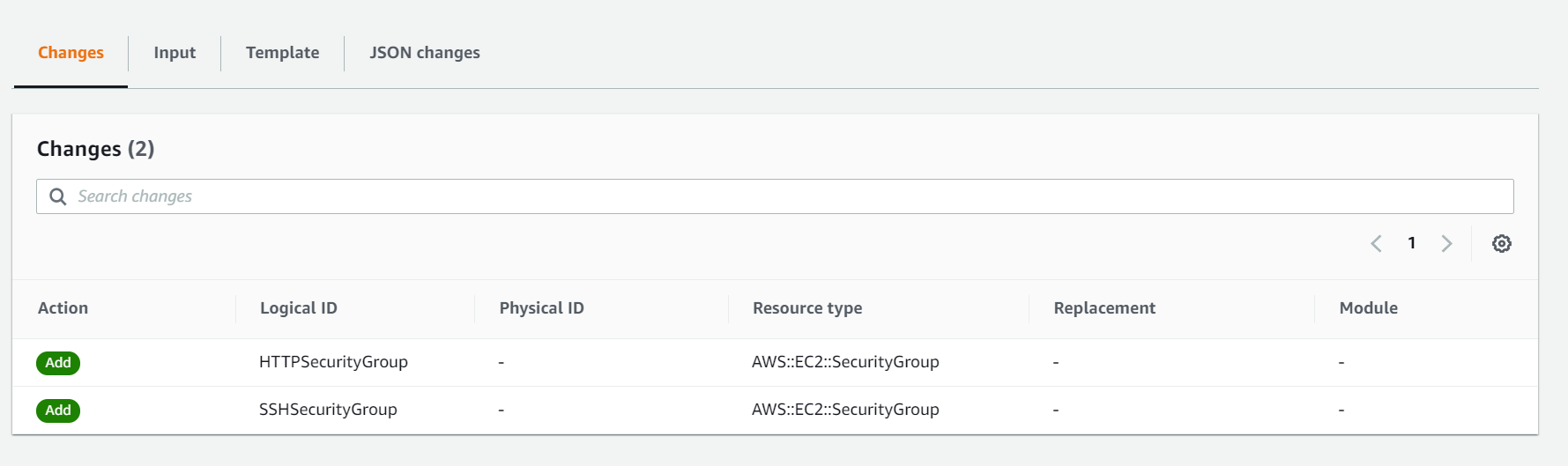
* StackSets allows create, update, deploy and delete of a stack on multiple regions/accounts.
* The change in stackset automatically made changes to all concerned region’s/account’s stacks.
* Administrator account should create the StackSet.
* Updates always affects all the stacks. Selective update not possible.
* We can delete a stack and resources for particular region/ account
* Delete a Stack from StackSet
* Delete all Stacks from StackSet (Preparing the StackSet Deletion)
* To delete the StackSet first we have to detach the stacks inside it and then delete the set. To clean it completely we have to delete the Stacks individually as we so normally.

**Changeset:**

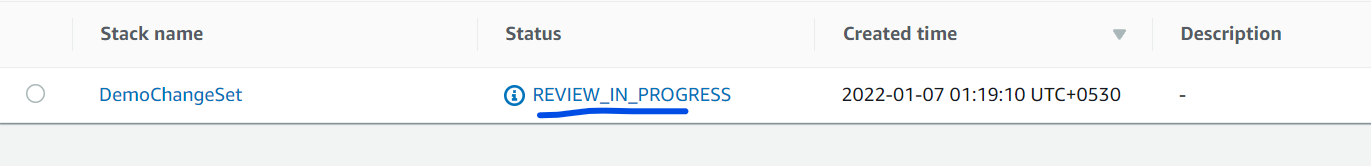
Its gives us a preview of the Stack before the actual creation of the stacks. We can examine the various configuration before the actual creation.



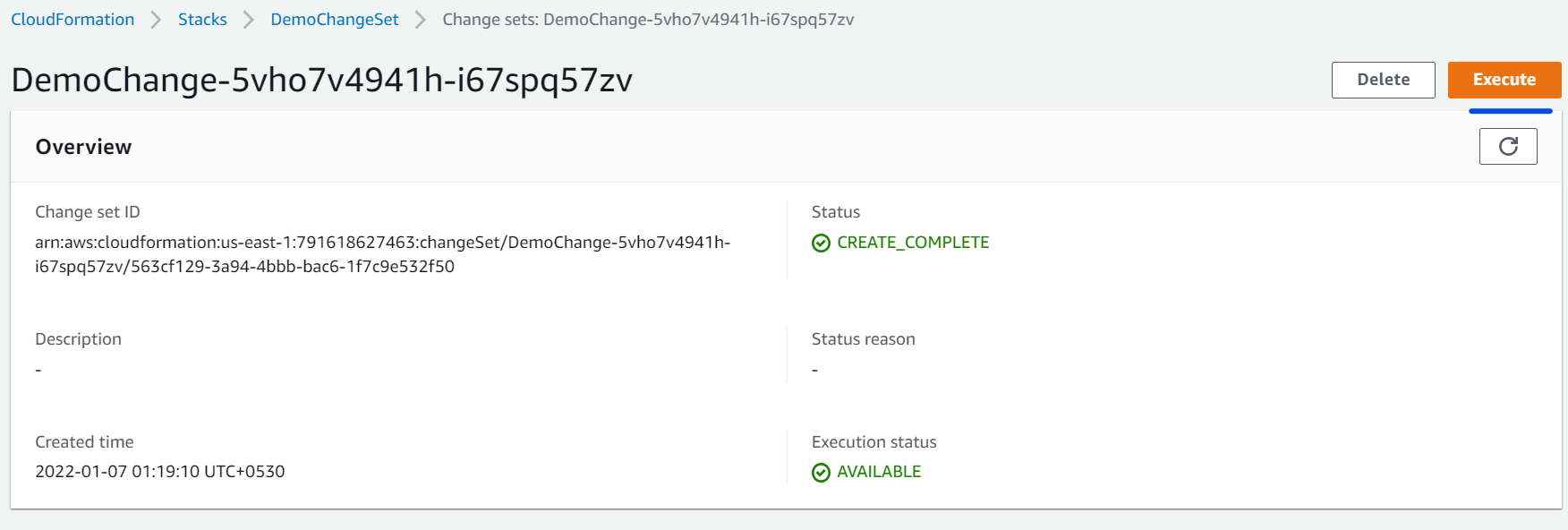
Click on Create Change Set

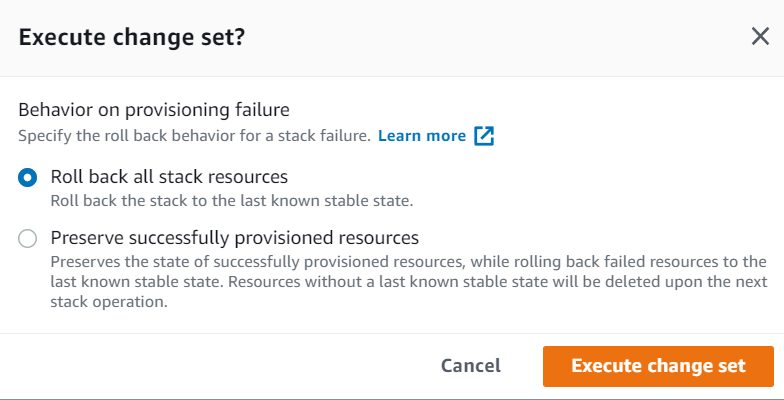


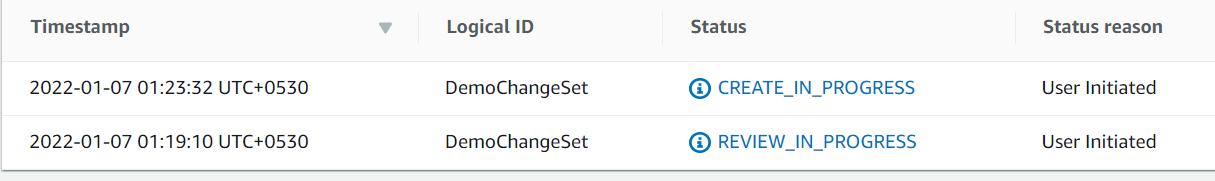
We can see all the sections as previews. If we go back to the Stack Page it will be in Review in progress state. Select the stack



If you are happy with the review we can then got for Execution







This is a great option in organisation to do the review of the stack by senior member before deploying it.

**Rollback Triggers:**

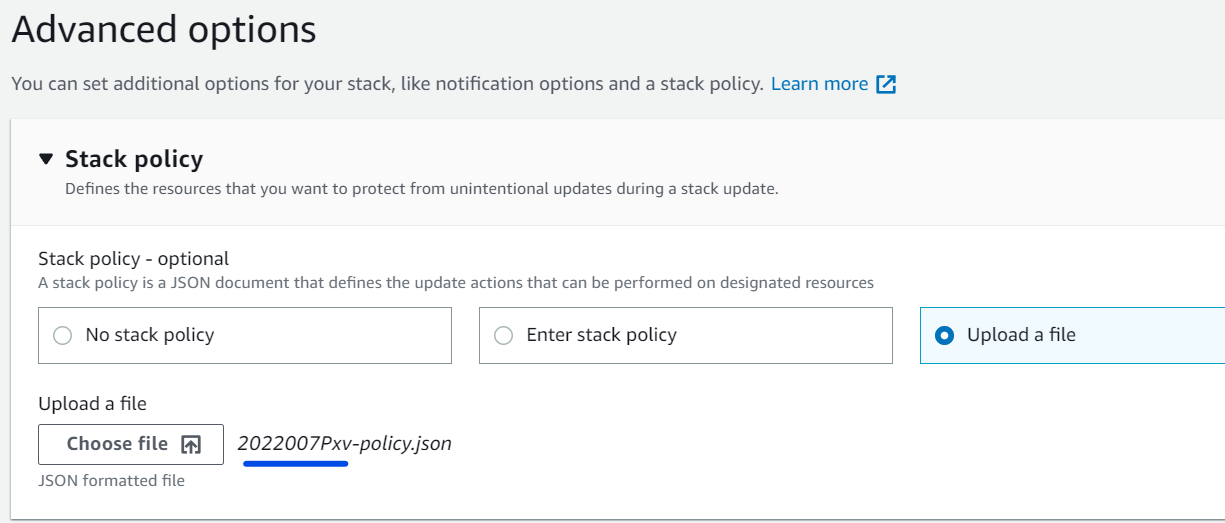
* When a stack creation/update is ging on we can monitor the CloudWatch alarm and if it status at alarm state then it can trigger the rollback of the stack operations.
* CloudFormations monitors the specified CloudWatch alarms.
* We can have upto 5 CloudWatch alarams.
* In Nested stacks rollback of the parent stack will also rollback all the child stacks.

**Stack Policy:**

StackPolicy provides an extra layer of protection to Stack resources while getting updating or deleting.

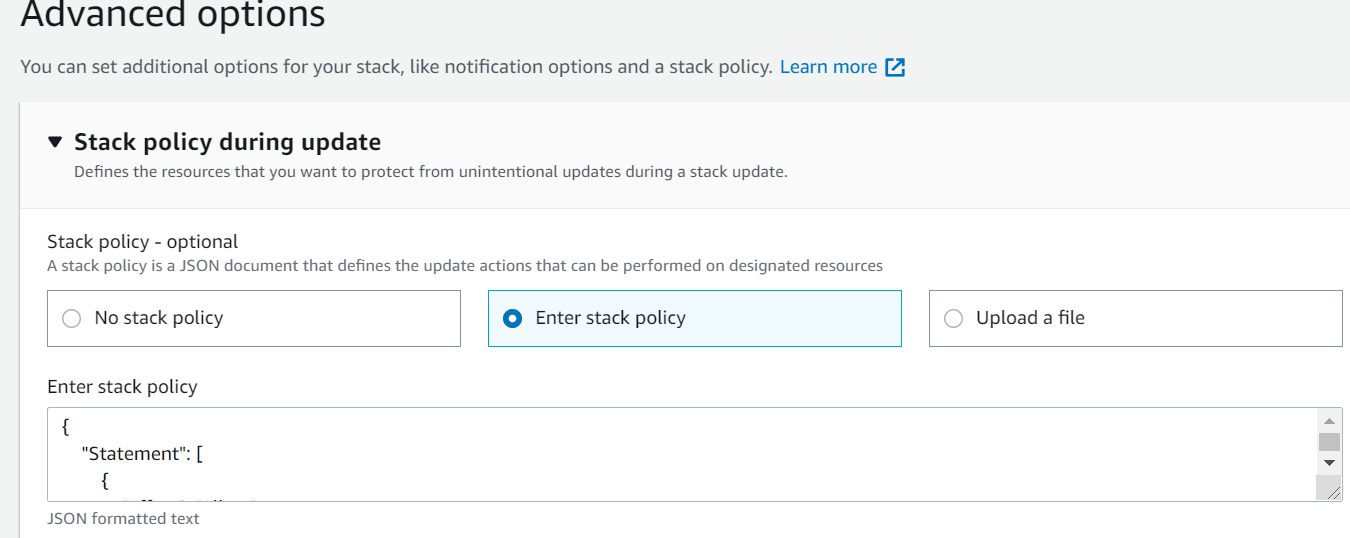
Task: Create a stack with a policy and then try to upload the stack which didn’t follow the policy and update get failed.

We can add the poliy under Advance section of the Stack creation page

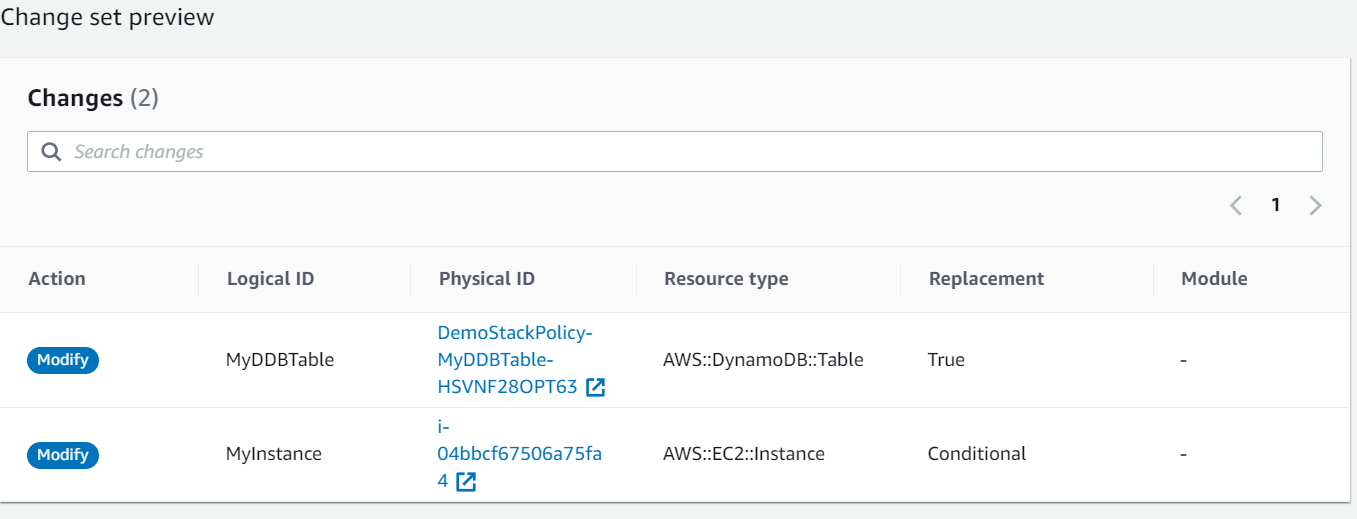


The above stack basically created an EC2 and a Dynamo DB and in Policy we defined that the update to stack is possible but anything change to DynamoDB will not be allowed. Hence when we will update the Stack mentioning to change in DynamoDB then policy will breech and cause the rollback.

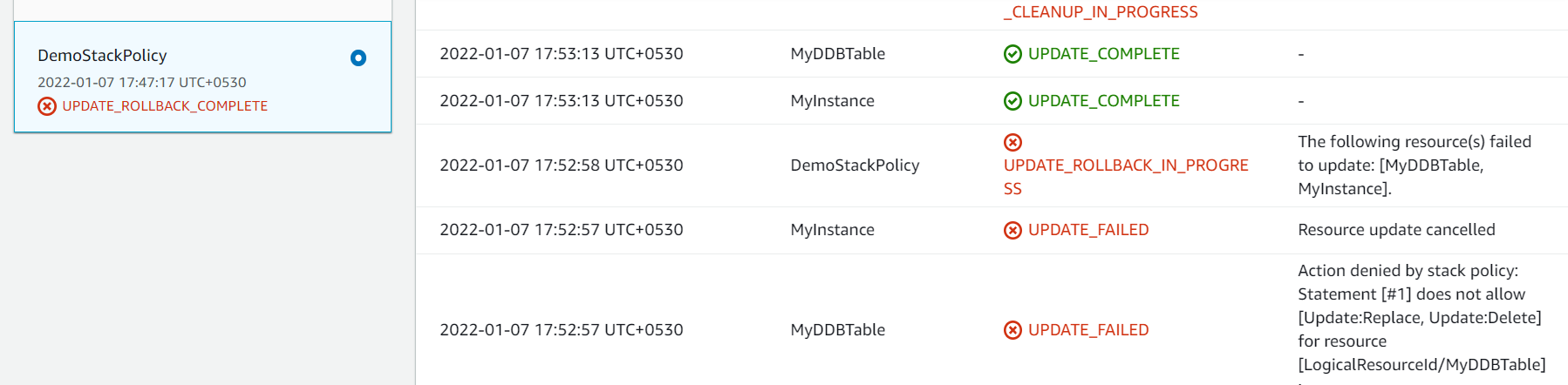
While updating the Stack also we can update the policy.



But here we will go with old policy



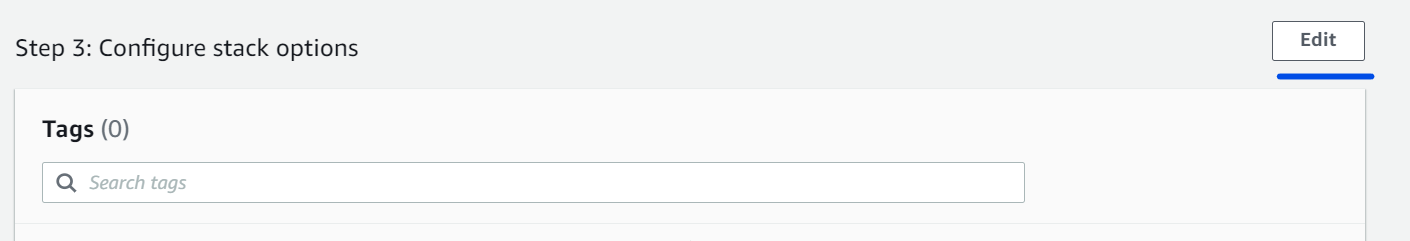
Rollback



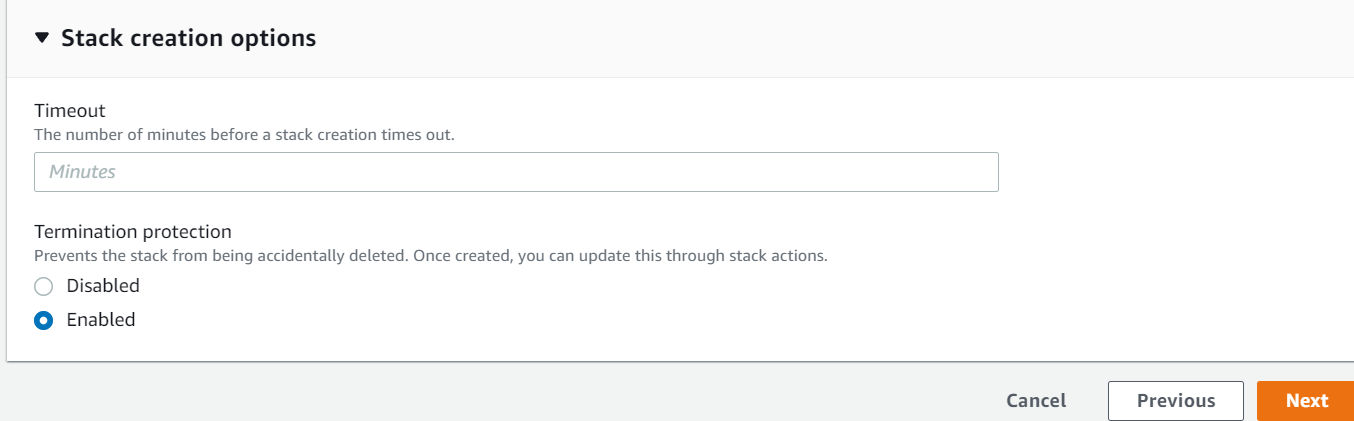
**Termination Protection:**

* Like EC2 we can also enable termination protection to prevent any accidental delete of the Stack resources.
* Applied to any Nested Stacks
* We can tighten the IAM policies (Only specific users can delete)

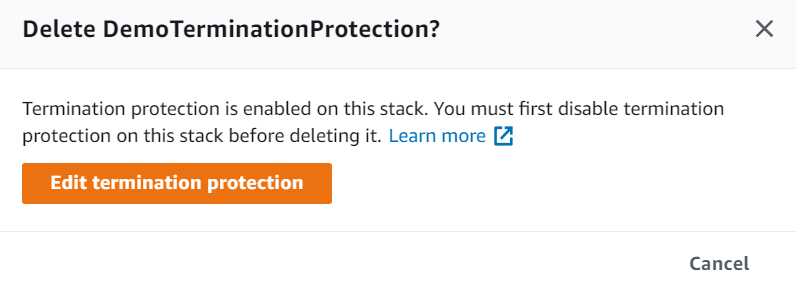
While creating the Stack



Then enable the termination protection



Now when you delete the Stack the extra message will pop up to disable the Termination



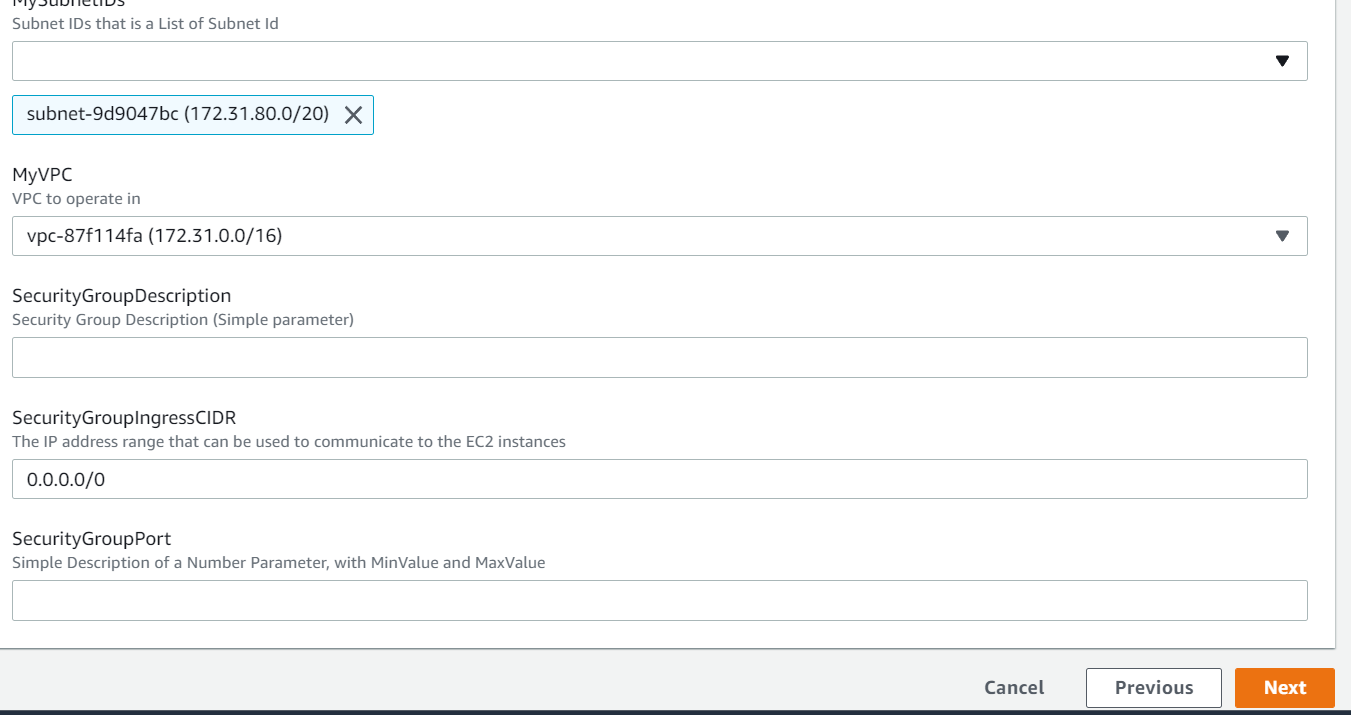
**Service Role:**

* IAM role that allows CloudFormation to create/update/delete stack resources on your behalf
* By default, CloudFormation uses a temporary session that it generates from your credentials.

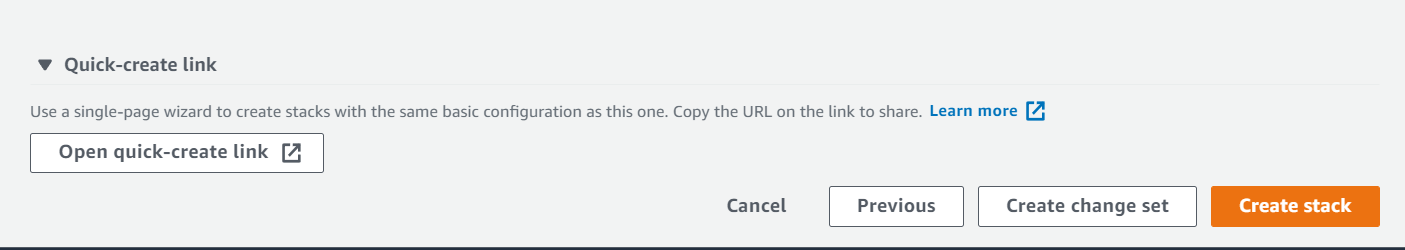
**Quick-Create Link for Stacks:**

A CloudFormation feature that allows you to create CloudFormation stacks quickly from AWS Console without pass through the wizard pages.

Upload the template and fill the parameters



UnderReview section in just before giving the create task



<https://console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/quickcreate?templateUrl=https%3A%2F%2Fs3-external-1.amazonaws.com%2Fcf-templates-gn1d3ra8incb-us-east-1%2F2022007DM6-0-parameters-hands-on.yaml&stackName=Demo-Quick-Link&param_DbSubnetIpBlocks=10.0.48.0%2F24%2C%2010.0.112.0%2F24%2C%2010.0.176.0%2F24&param_InstanceType=t2.small&param_KeyName=myEC-keypair&param_MySubnetIDs%5B%5D=subnet-9d9047bc&param_MyVPC=vpc-87f114fa&param_SecurityGroupDescription=&param_SecurityGroupIngressCIDR=0.0.0.0%2F0&param_SecurityGroupPort=>

Give this link whoever you want to create the task.