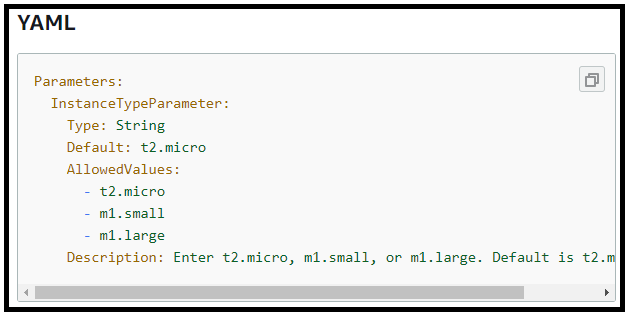
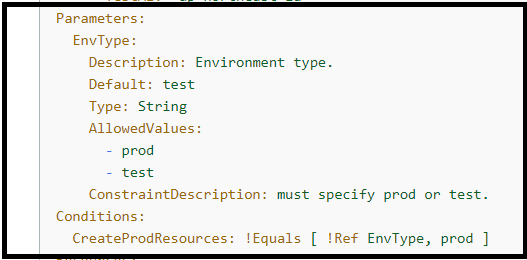
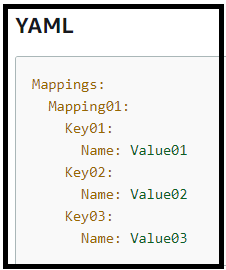
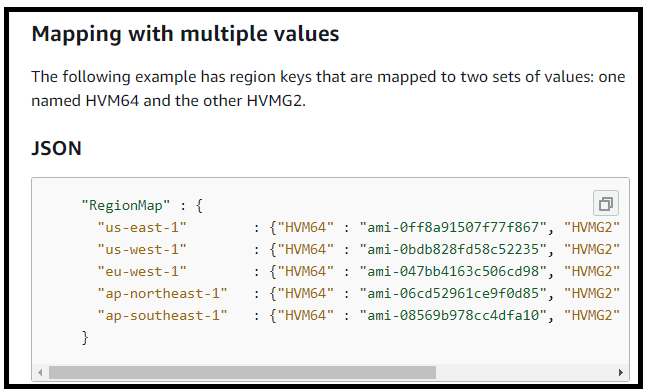
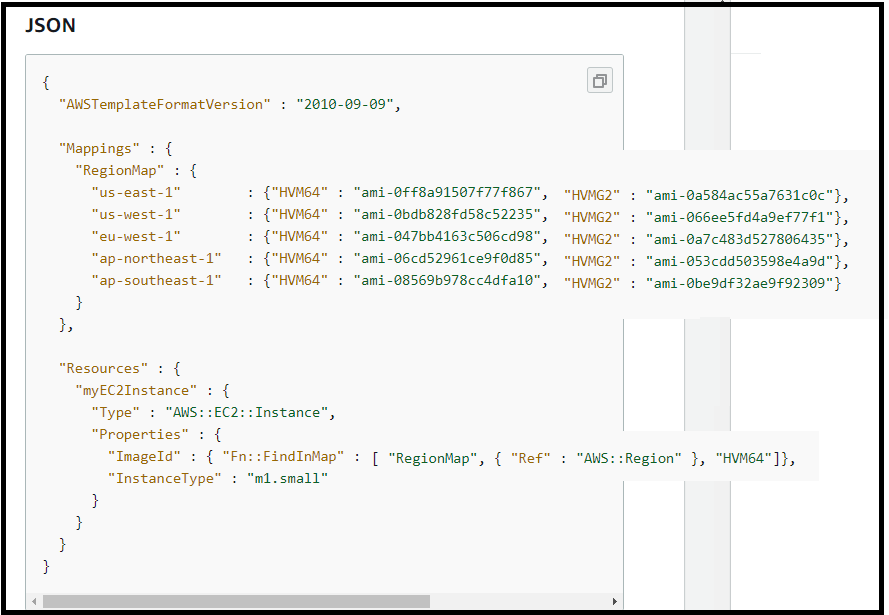
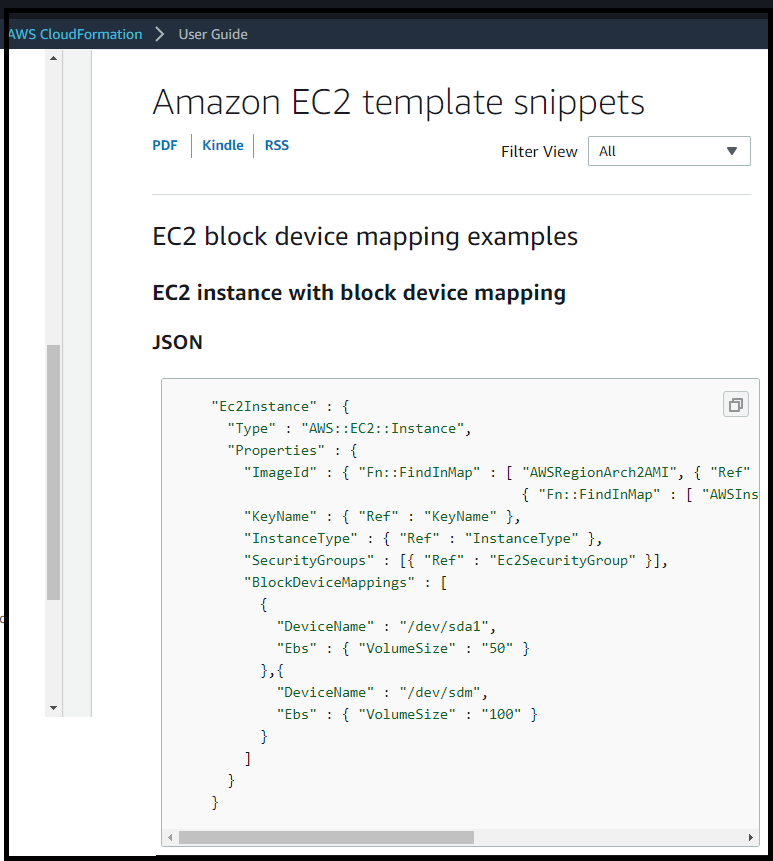
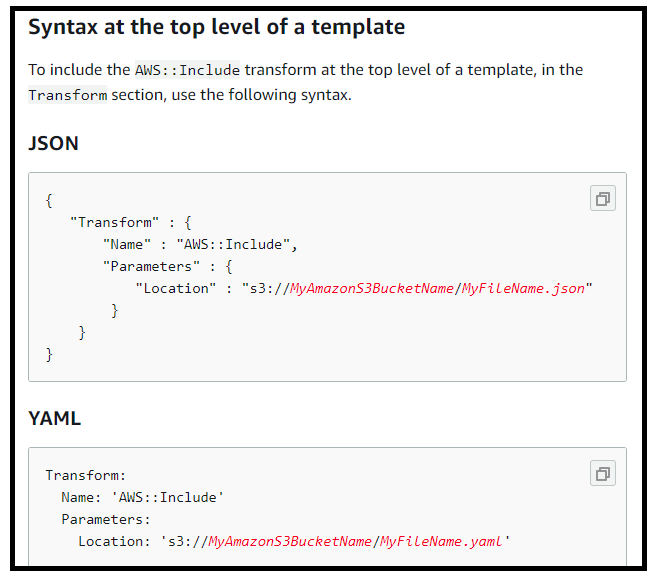
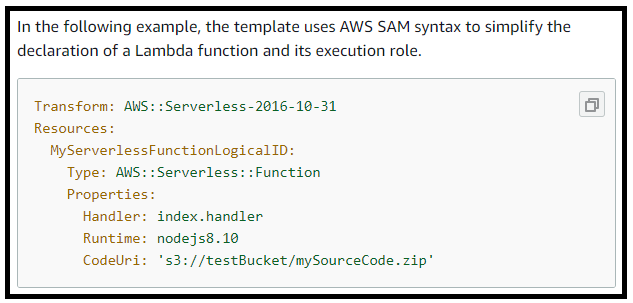
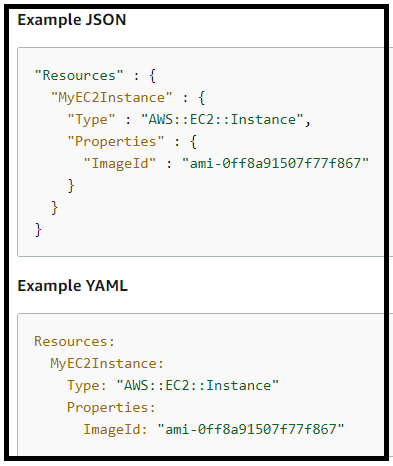
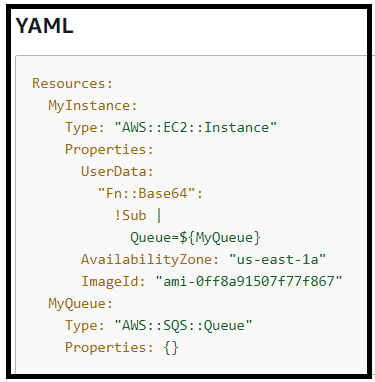
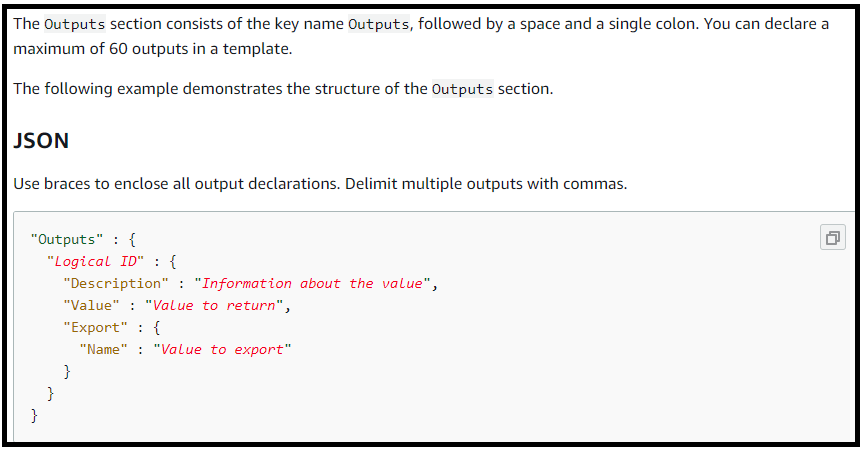
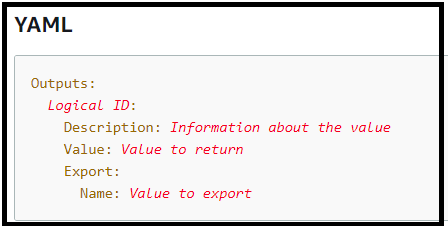
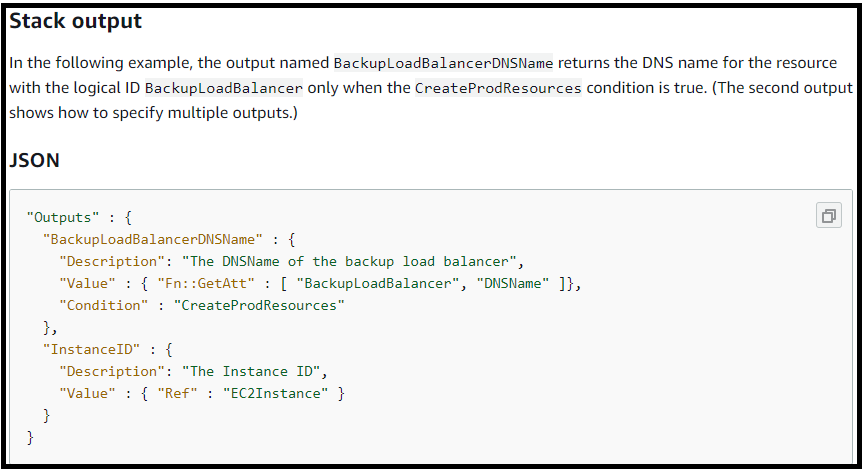
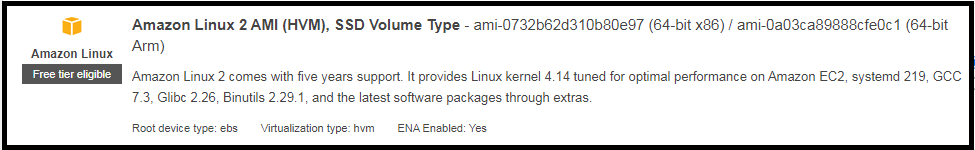
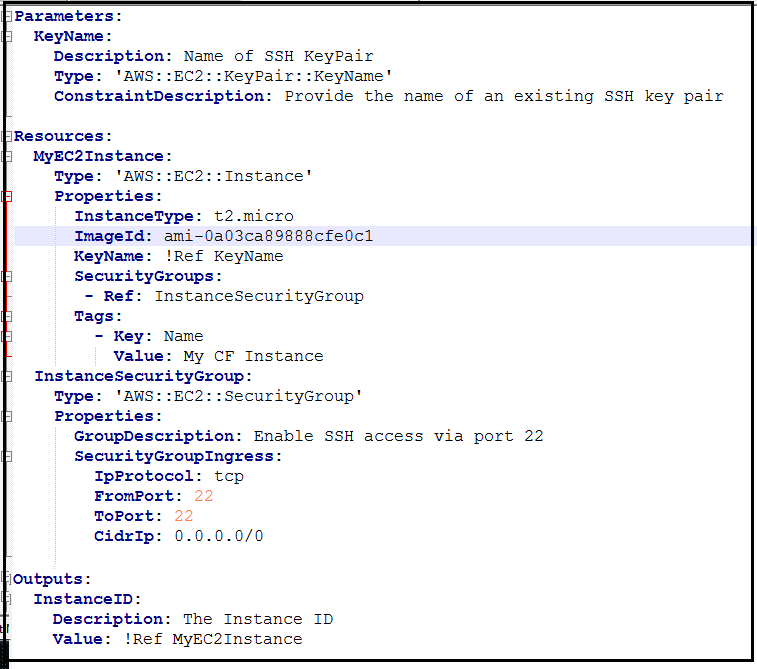
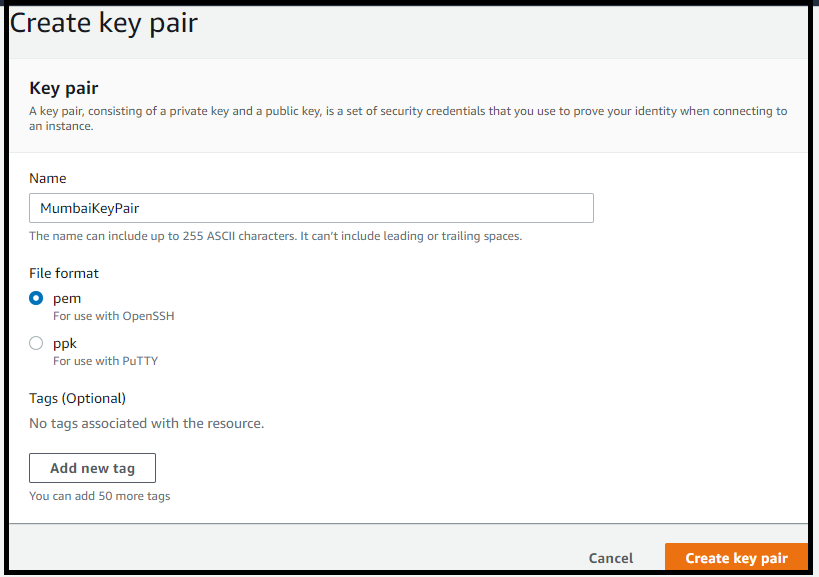
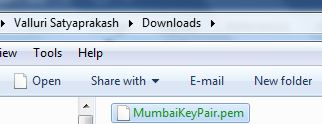
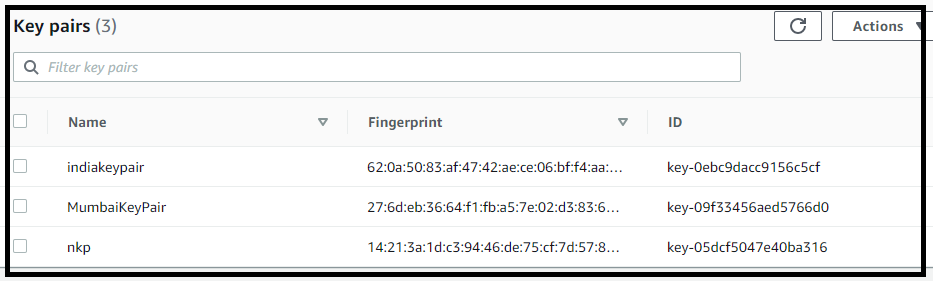
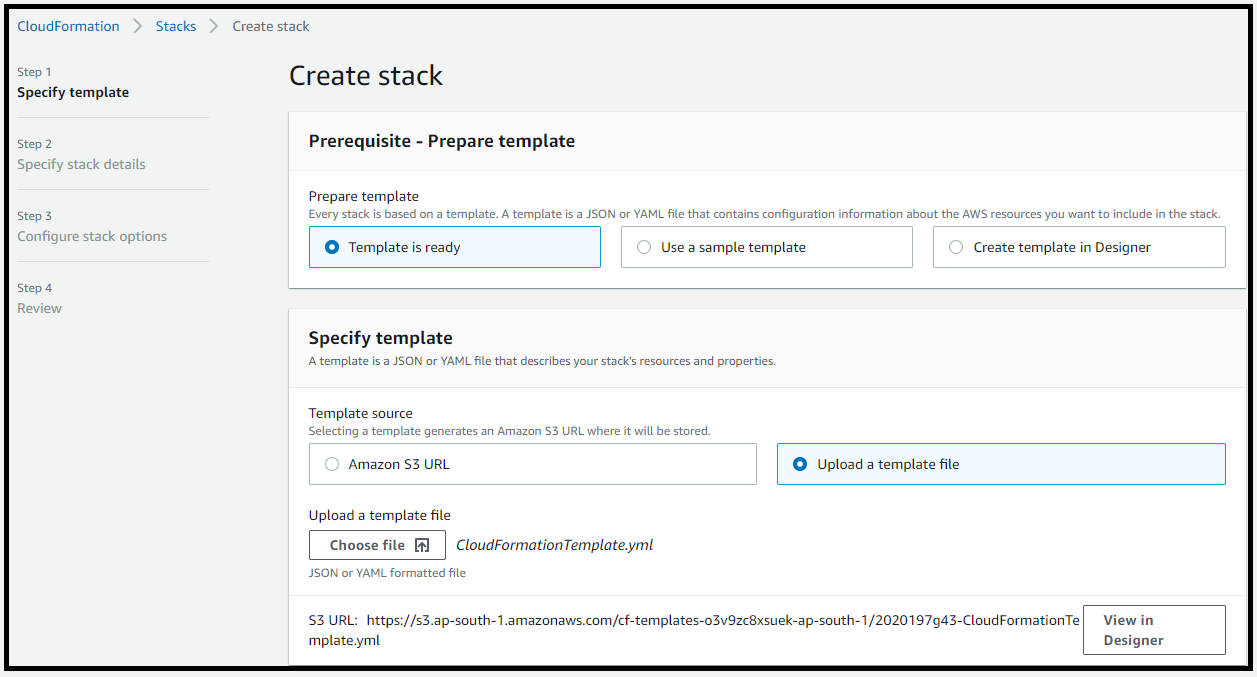
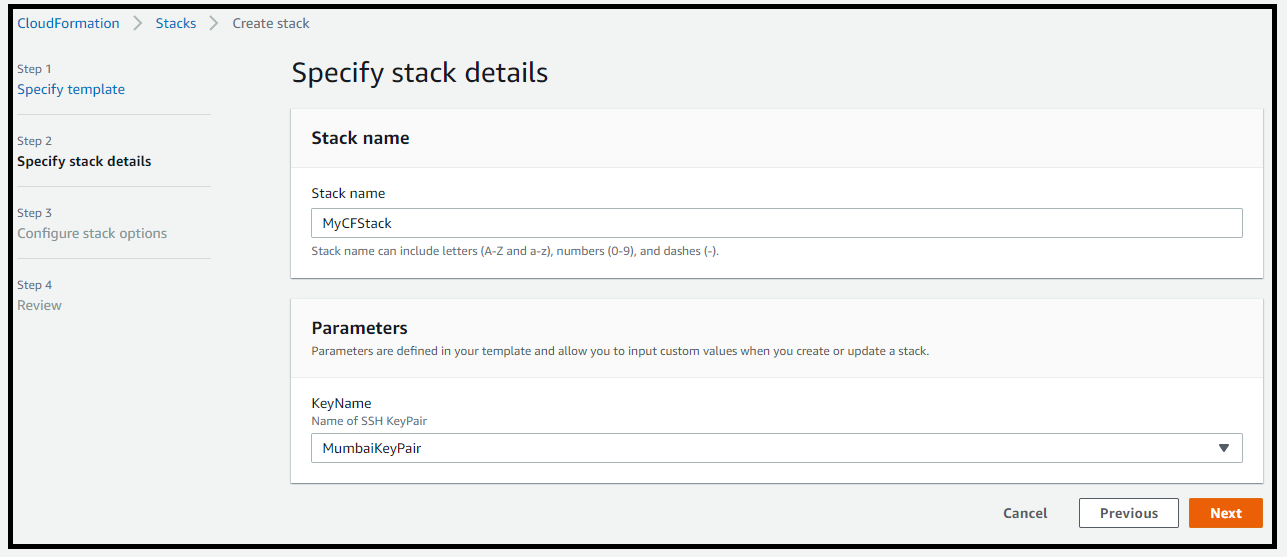
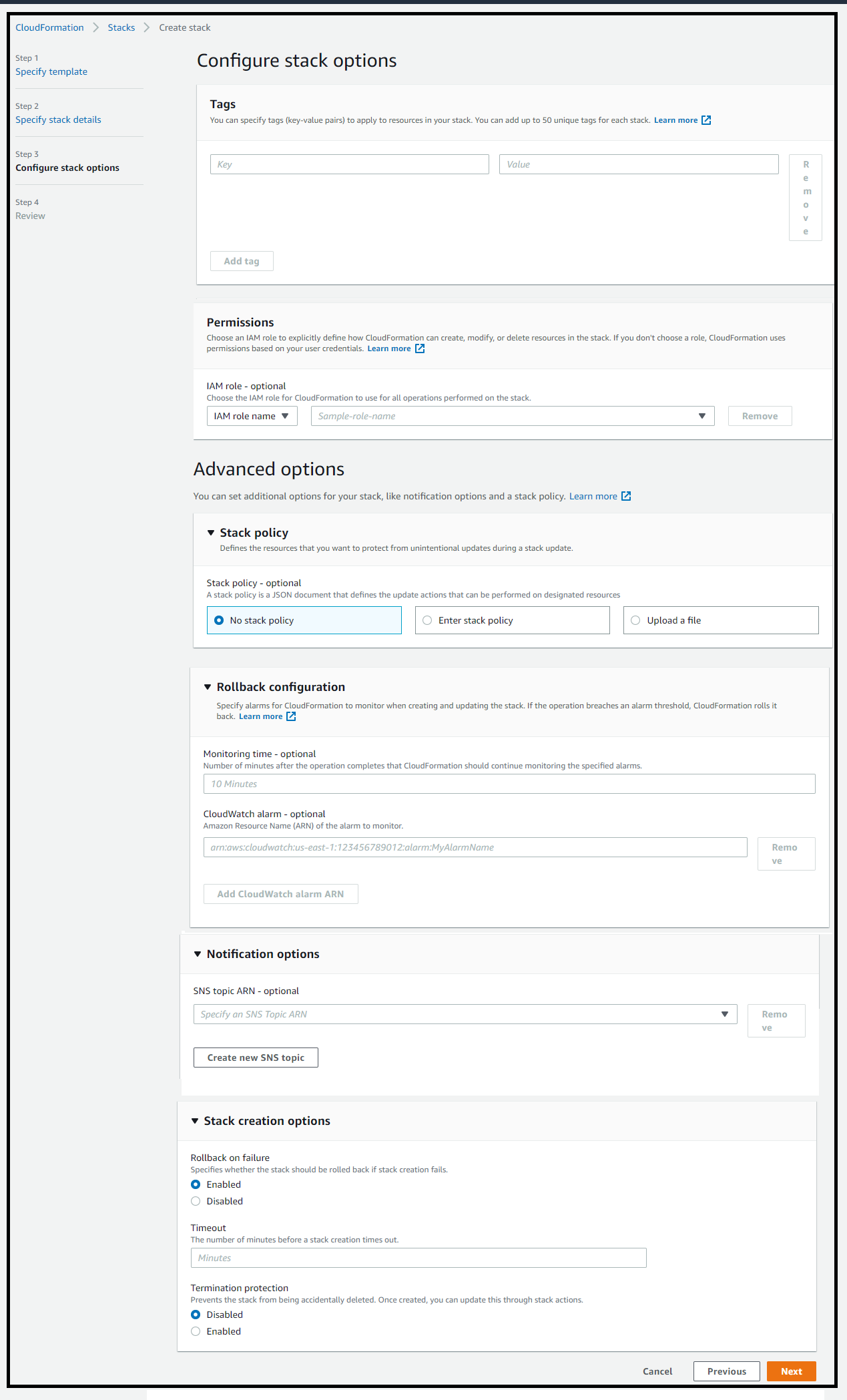
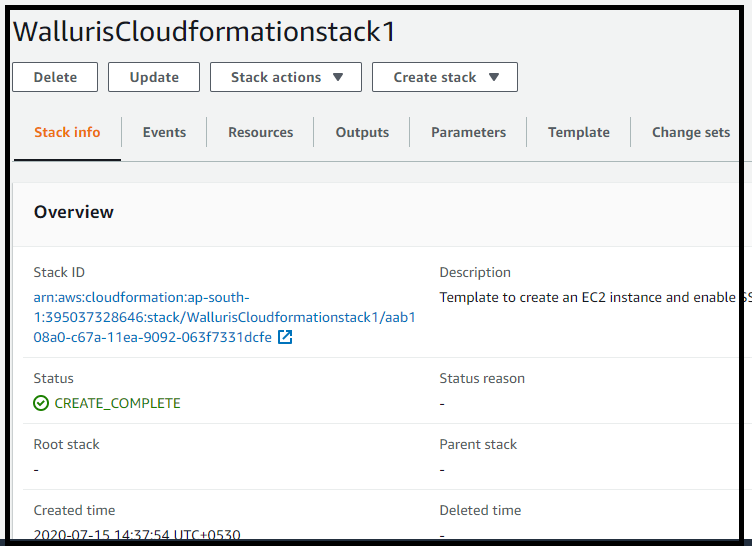
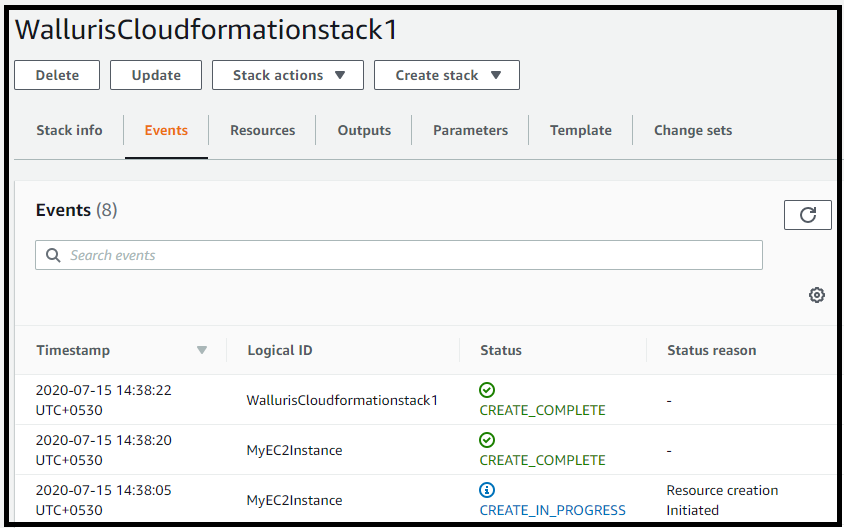
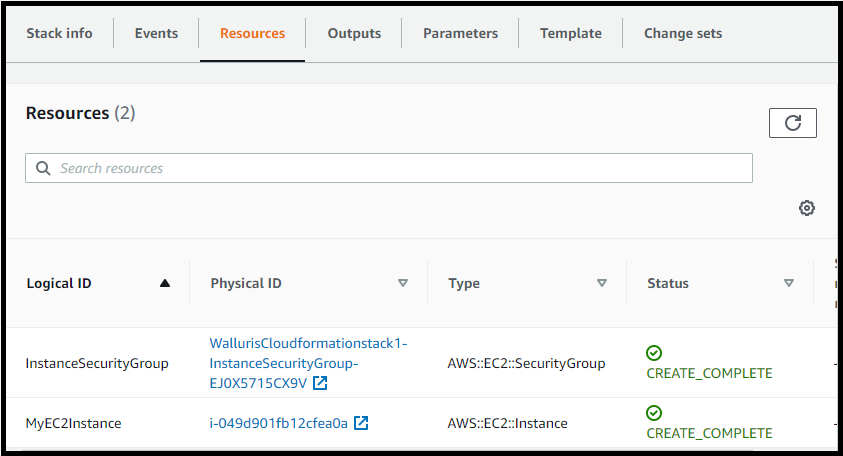
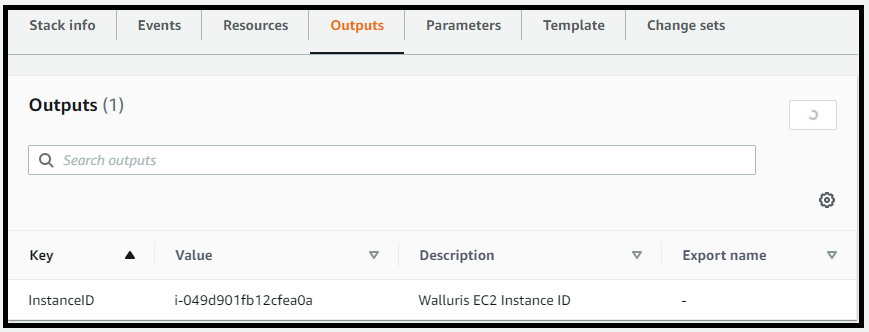
 AWS CLOUD FORMATION.  
  
1. Cloud formation is a service that allows you to manage, configure, and provision your aws infrastructure as code.  
  
We provision the resources by mentioning them in the cloud formation template, which is uploaded in S3.  
  
Cloud formation takes the template, it interprets it and makes the appropriate api calls in order to create the resources that we have defined in the template.  
  
Supported formats are YML and JSON.  
  
2. Benefits ?  
  
CF allows you to provision your infrastructure in a consistent manner, with fewer mistakes.  
  
It takes less time and effect than configuring manually.  
  
You can version control your templates.  
  
It can be used to manage updates to existing cloud formation platforms.  
  
It can also handle dependencies between different resources, to ensure that resources are created in a correct order.  
  
It can be use to quickly rollback and delete the entire stack as well.

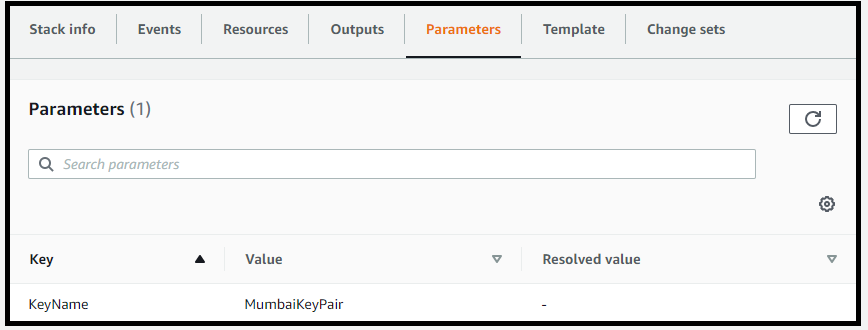
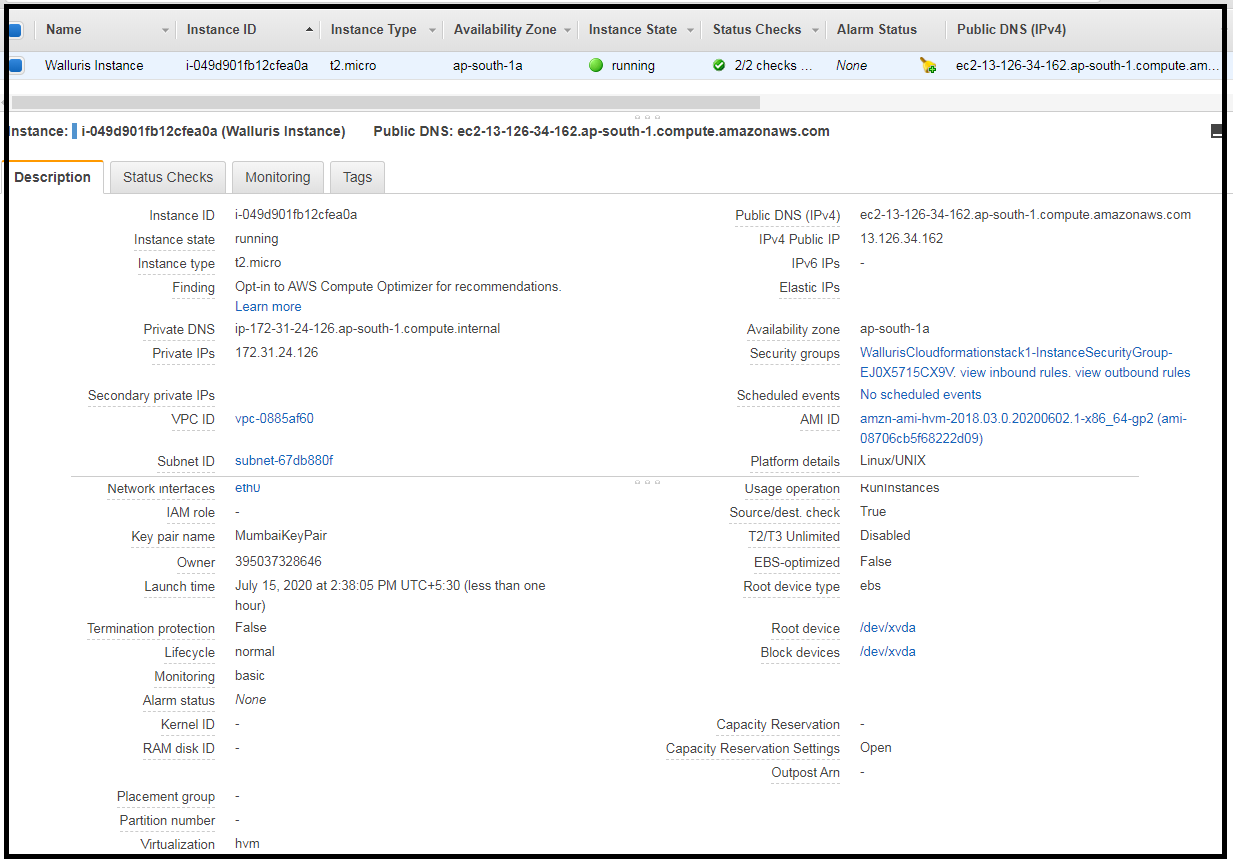
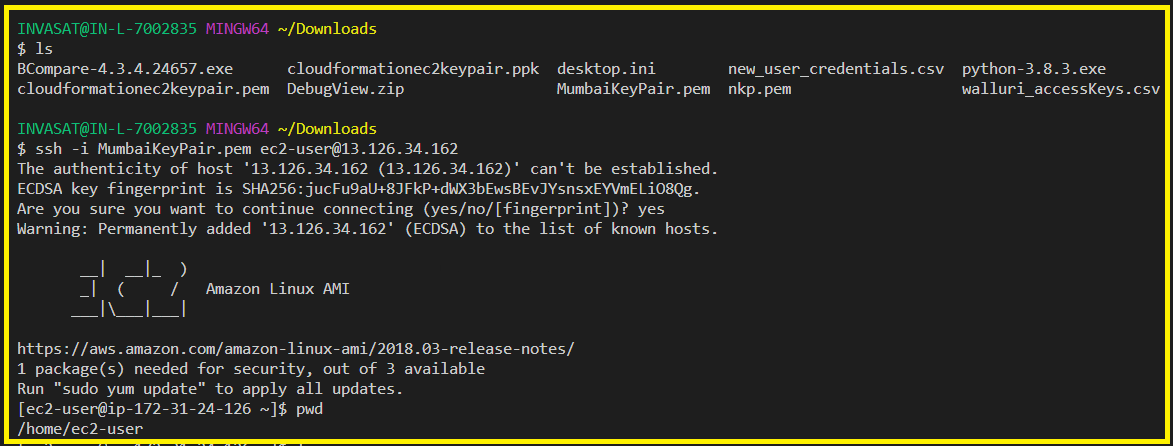
3. CLOUD FORMATION TEMPLATE STRUCTURE.  
  
3.1  
AWSTemplateFormatVersion : The only supported version is 2010-09-09. This refers to the version of the template format.  
Description : Describe what you want this template does.  
Metadata: It is used to mention data about ‘data’. You can put in a custom field.  
  
3.2   
Parameters :These are input values that you input to cloud formation when we launch a stack using this template.  
Example. The parameter name is ‘InstanceTypeParameter’ and you have to provide string values which have to be amongst the values listed in ‘AllowedValue’ field.  
  
  
3.3 Conditions : It is completely user defined and we can use this to test a condition And take action based on the outcome of that condition.  
Example.  
  
  
  
  
3.4  
Mappings.  
We can use mappings to set our own user defined values.  
  
  
  
  
Another Example:  
  
  
Using Mapping in an Example.  
  
  
  
  
3.5   
Transform : This allows you to include snippets of code outside the main template. You can either include code for a lambda function, you could include your own code that you want to re-use, use a lot of code snippets that aws provides (cloud formation snippets.).  
  
This example is in JSON but you can get the YAML also. We can create re-usable pieces of code, which you can store in S3 and cloud formation will allow you to reference that with in your cloud formation template.  
  
Thus This feature helps you to re-use code and maintain consistency among your code base.  
  
  
AWS::Include transform.  
The ‘include’ transform lets you create a reference to a template snippet in a S3 bucket.  


AWS::Serverless transform  
This is a macro hosted by AWS cloud formation.  
It takes an entire template written in AWS Serverless Application Model syntax, And then transforms and expands it into a compliant AWS cloud formation template.  
  
  
  
3.6   
Resources.  
This is where you define aws resources that you want cloud formation to deploy for you.  
  
  
  
3.7 outputs.  
It is completely user defined.  
You can see them displayed out to the console or they could also be inputs to other cloud formation stack as well.  
The output section declares output values that you can import to other stacks, return in response to describe stack calls, or view in the aws cloud formation console.

For example : You can output the s3 bucket name for a stack to make the bucket easier to find.  
  
  
  
Example:  
  
  
4.CLOUD FORMATION LAB.  
Firstly we have a template that will create a new EC2 Instance and SSH into that new instance.  
  
  
  
  
  
  
  


After creating the stack with the above configuration, the dashboard looks like below.  
  
  
  
  
  
  


5. Cloud Formation Nested Stacks.  
  
They are stacks that create other stacks within cloud formation.  
They allow you to re-use your cloud formation code for common use cases.  
  
For example a standard configuration for your load balancer / web server / application server.  
  
We can store re-usable code with in one cloud formation template and reference it from another cloud formation template.   
  
Nested stacks are to be declared within the ‘Resources’ section of the template.  
It has to be of type AWS::CloudFormation::Stack  
There a few supported properties for this type.  
NotificationARNs : existing SNS topic.  
Parameters : parameters that you want to pass to cloud formation when the nested stack is created.  
TemplateURL : url of the template which specifies the nested stack you want to create. S3 link.  
TimeoutInMinutes : default – no timeout.