😊 Amazon CloudFormation

Amazon CloudFormation is a service provided by Amazon Web Services (AWS) that allows you to automate the deployment of infrastructure resources in a consistent and repeatable manner. It uses templates written in JSON or YAML format to describe the desired state of your infrastructure, including things like EC2 instances, databases, networking configurations, and more.

With CloudFormation, you define your infrastructure as code, which allows you to version control your infrastructure changes, track modifications, and easily replicate environments across different regions or AWS accounts. CloudFormation manages the provisioning and configuration of resources, handling dependencies and ensuring that resources are created or updated in the correct order.

You can create, update, and delete entire stacks of resources using CloudFormation, making it a powerful tool for managing AWS infrastructure at scale. It also integrates with other AWS services, such as AWS CloudTrail for logging and AWS Identity and Access Management (IAM) for controlling access to resources.

😄 Use cases of CloudFormation:

CloudFormation is widely used across various industries and organizations for automating the deployment and management of infrastructure resources on AWS. Some common use cases include:

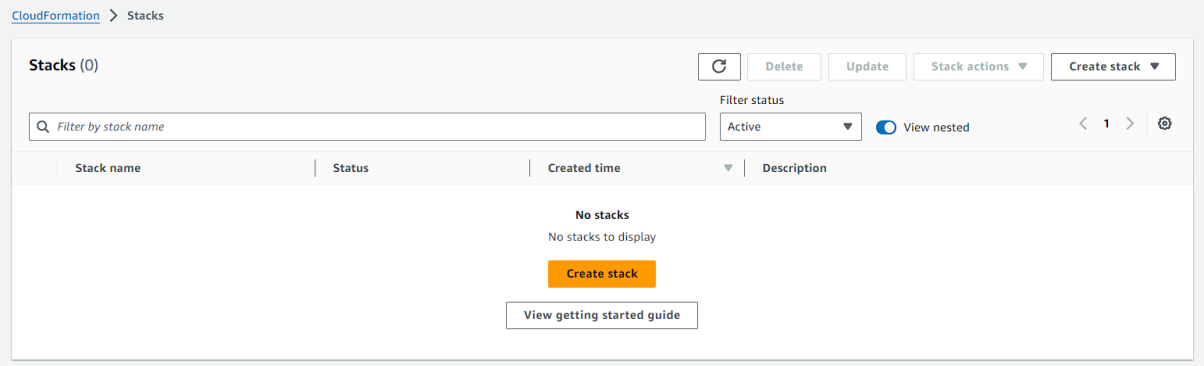
1. **Application Deployment:** CloudFormation allows you to define all the resources needed for your application, such as EC2 instances, load balancers, databases, and storage, in a template. This enables you to consistently and reliably deploy your application across different environments, such as development, staging, and production.
2. **Microservices Architecture:** In a microservices architecture, applications are composed of multiple independent services. CloudFormation can manage the infrastructure for each service separately, making it easier to scale, update, and maintain the individual components of your application.
3. **Infrastructure as Code (IaC):** CloudFormation enables you to treat your infrastructure as code, allowing you to version control your infrastructure changes, track modifications, and collaborate more effectively with your team members. This helps ensure that your infrastructure configurations are reproducible and can be easily audited and reviewed.
4. **Disaster Recovery:** CloudFormation templates can be used to create backup infrastructure in a separate AWS region, enabling you to quickly recover your applications and data in the event of a disaster or outage in your primary region.
5. **DevOps Automation:** CloudFormation integrates with other DevOps tools and services, such as AWS CodePipeline and AWS CodeDeploy, to automate the deployment pipeline. This allows you to automatically trigger CloudFormation stacks

based on code changes, perform automated testing, and deploy applications to production environments.

6. **Compliance and Governance:** CloudFormation can help enforce compliance and governance policies by defining security configurations, IAM roles, and resource tagging standards in your templates. This ensures that your infrastructure meets regulatory requirements and security best practices.
7. **Cost Optimization:** CloudFormation enables you to define cost-saving measures, such as auto-scaling policies, instance types, and resource tagging strategies, in your templates. This helps you optimize your infrastructure costs by right-sizing resources and minimizing waste.

😄 To begin with the Lab:

1. Login to AWS Console and navigate to CloudFormation.
2. There you need to create a stack. So, click on create stack.

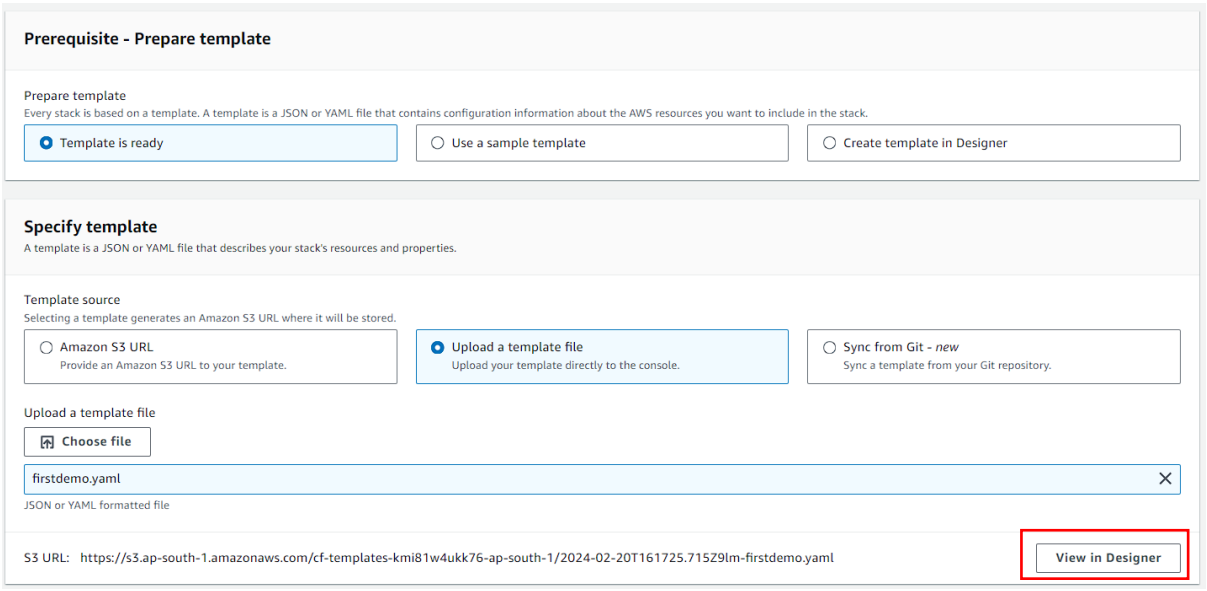


3. And here you can choose from three options, but for now we are going to click on Template is ready because I have a template ready with me.
4. You can also get this template from GitHub. The code written in this template is in YAML format.
5. So, in this code you will see that, we're going to Launch an EC2 instance which is of the type t2.micro. Here you will see the Image ID which you can get while launching the Instance. You can copy that ID and mention that in the code.
6. In the last you will see that we have defined a tag for our instance.

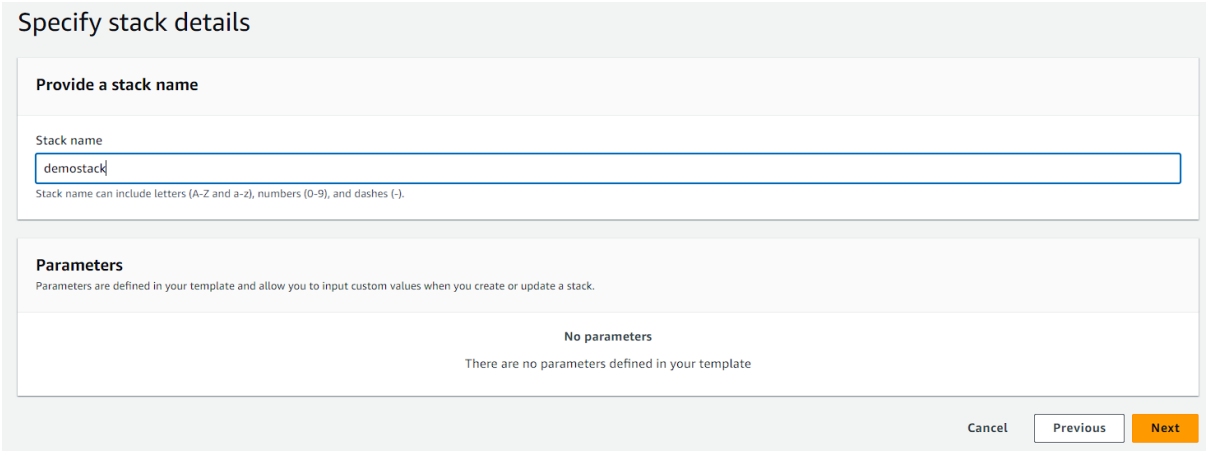
```yaml
Resources:
  MyInstance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: t2.micro
      ImageId: ami-06b72b3b2a773be2b
      Tags:
        - Key: "Name"
          Value: "MyDemoInstance"
```

7. Now you need to select Upload a template file and upload your YAML file here.

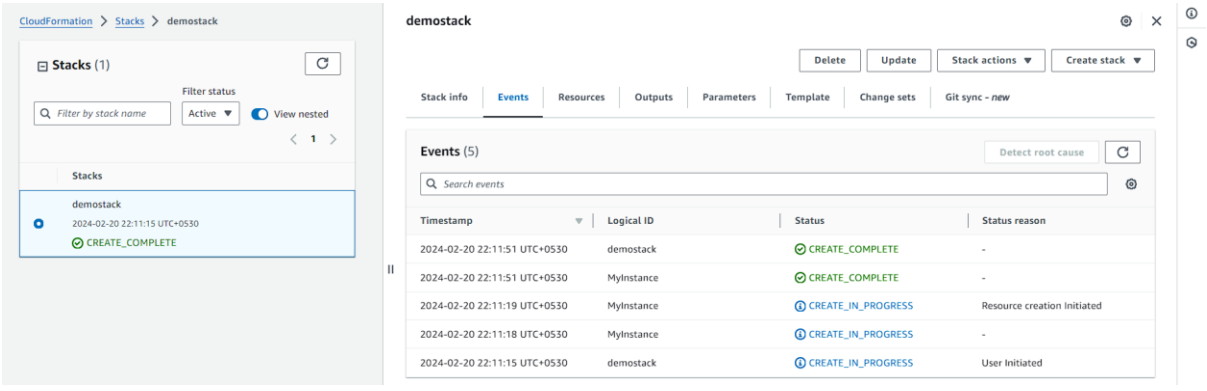
8. After that click on View in Designer and validate your template. This is an optional step if you want to do it then you can.



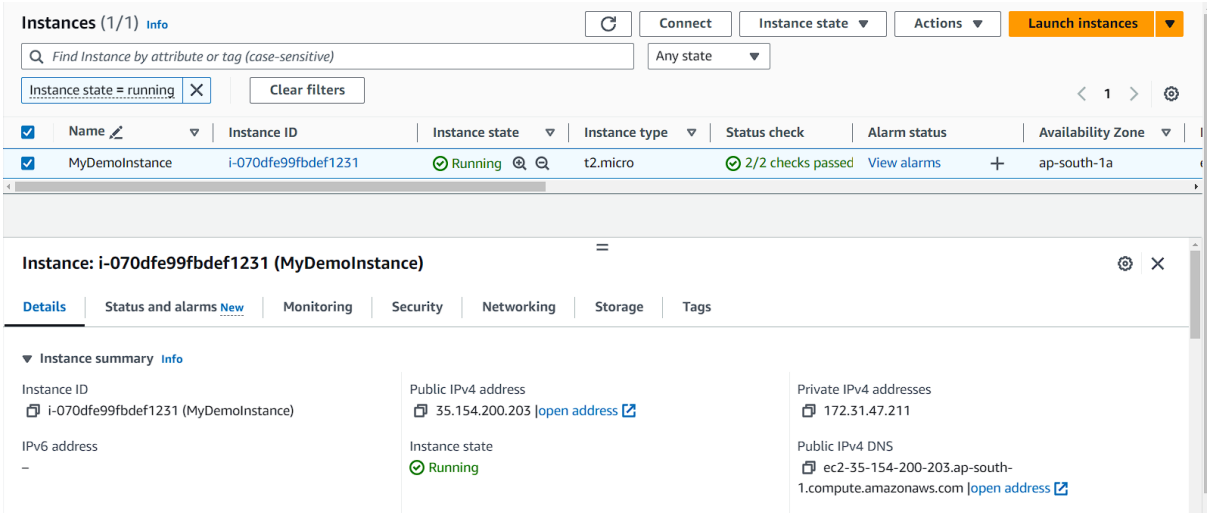
9. Now give your stack a name and click on next.



10. Now you can skip step 3 and step 4 for now and directly create your stack.
11. After about 2-3 minutes our stack gets created successfully.



12. Now if you will go to EC2, you will see that you instance has been launched successfully.



13. Now to perform a cleanup, select your stack and click on delete.
14. It will delete all of the resources for you, even the EC2 instance.