**Problem Statement:**

You work for XYZ Corporation. Your corporation wants to launch a new web-based application. The development team has prepared the code but it is not tested yet. The development team needs the system admins to build a web server to test the code but the system admins are not available.

**Tasks To Be Performed:**

1. Web tier: Launch an instance in a public subnet and that instance should allow HTTP and SSH from the internet.

2. Application tier: Launch an instance in a private subnet of the web tier and it should allow only SSH from the public subnet of Web Tier-3.

3. DB tier: Launch an RDS MYSQL instance in a private subnet and it should allow connection on port 3306 only from the private subnet of Application Tier-4.

4. Setup a Route 53 hosted zone and direct traffic to the EC2 instance.

You have been also asked to propose a solution so that:

1. Development team can test their code without having to involve the system admins and can invest their time in testing the code rather than provisioning, configuring, and updating the resources needed to test the code.

2.Make sure when the development team deletes the stack, RDS DB instance should not be deleted.

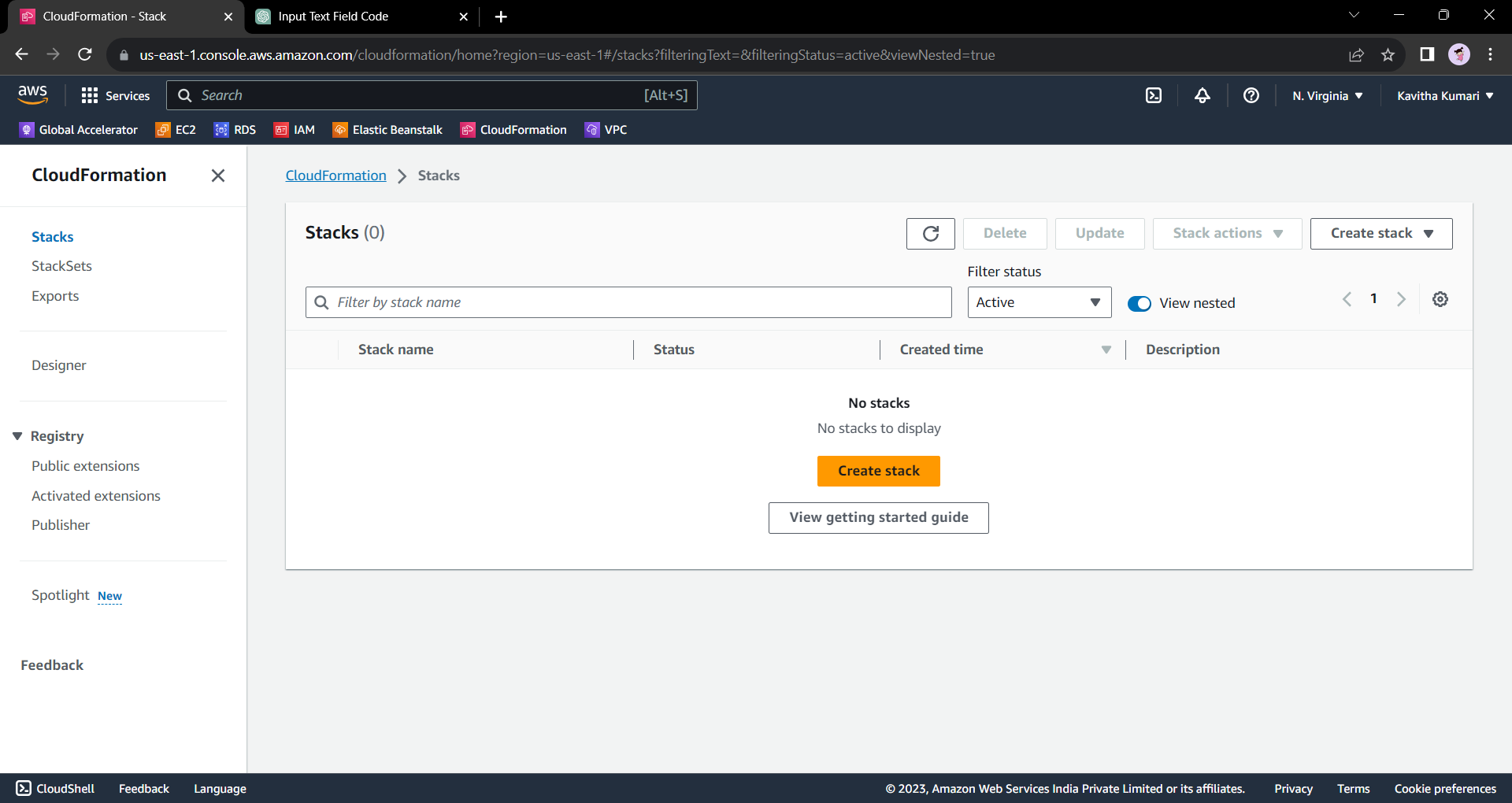
**Procedure: -**

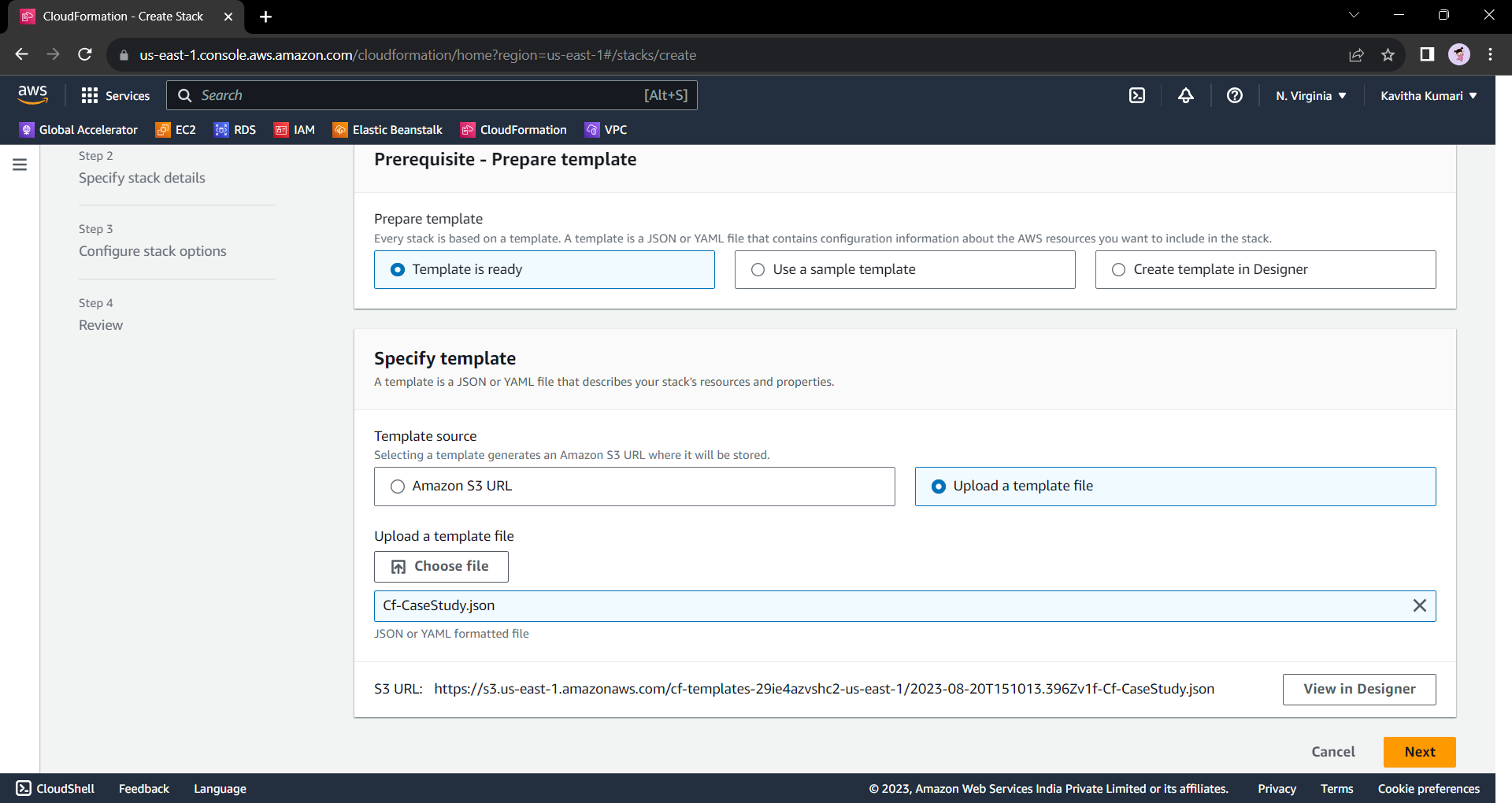
To achieve the tasks and propose a solution for automating the setup and testing environment, you can use AWS CloudFormation along with AWS services like EC2, RDS, and Route 53. CloudFormation allows you to define and provision infrastructure as code, making it easier to create and manage your resources consistently. Additionally, you can use CloudFormation Stack Policies to prevent the deletion of specific resources like the RDS DB instance.

Let us create the stack which has the following resources.

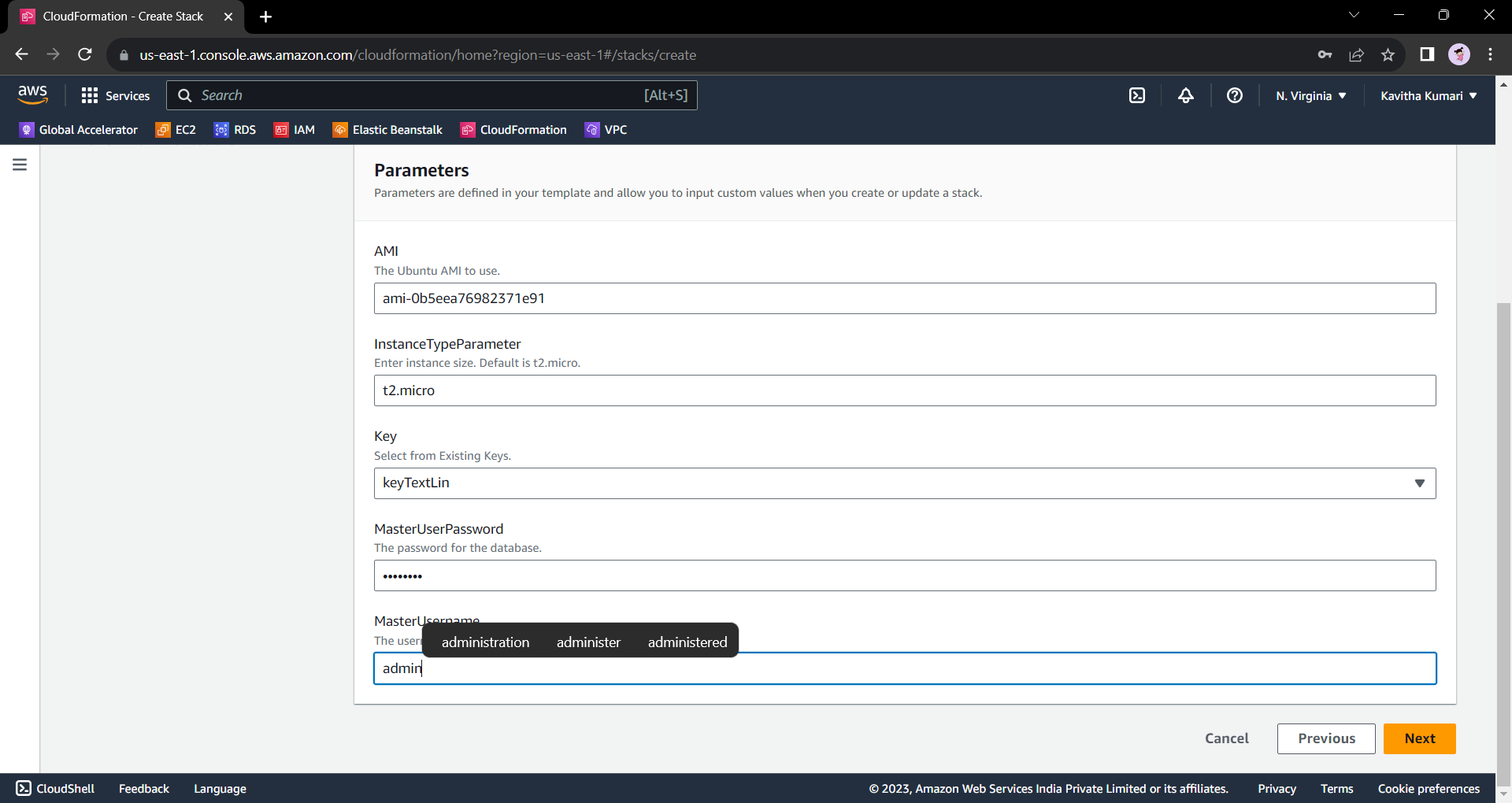
* VPC with the CIDR block as 10.10.0.0/16
* Internet gateway attached to the VPC.
* SubnetA with the CIDR block as 10.10.1.0/24 and create a Route table where we give the target of internet gateway which we create to make it as public route table. SubnetA will be associated with the Public route table.
* SubnetB with the CIDR block as 10.10.2.0/24 and create the Route table which is private and associate SubnetB with it.
* NAT gateway where we will associate the subnetA and we will allocate the Elastic IP address for it. In the Public route table, we will give the destination CIDR block as 0.0.0.0/0 and target as NAT gateway.
* Security group for the VPC and we will allow the TCP with the port number 22 and it will allow traffic from anywhere.
* Security group for the database.
* MYSQL database.
* Linuxpublic intance and Linuxpublic instance
* Hosted zone
* <https://github.com/Origamini/JSONFiles/blob/main/Cf-CaseStudy.json>
* Form the above link you can directly fork the JSON file for the project.

Let us create a stack.

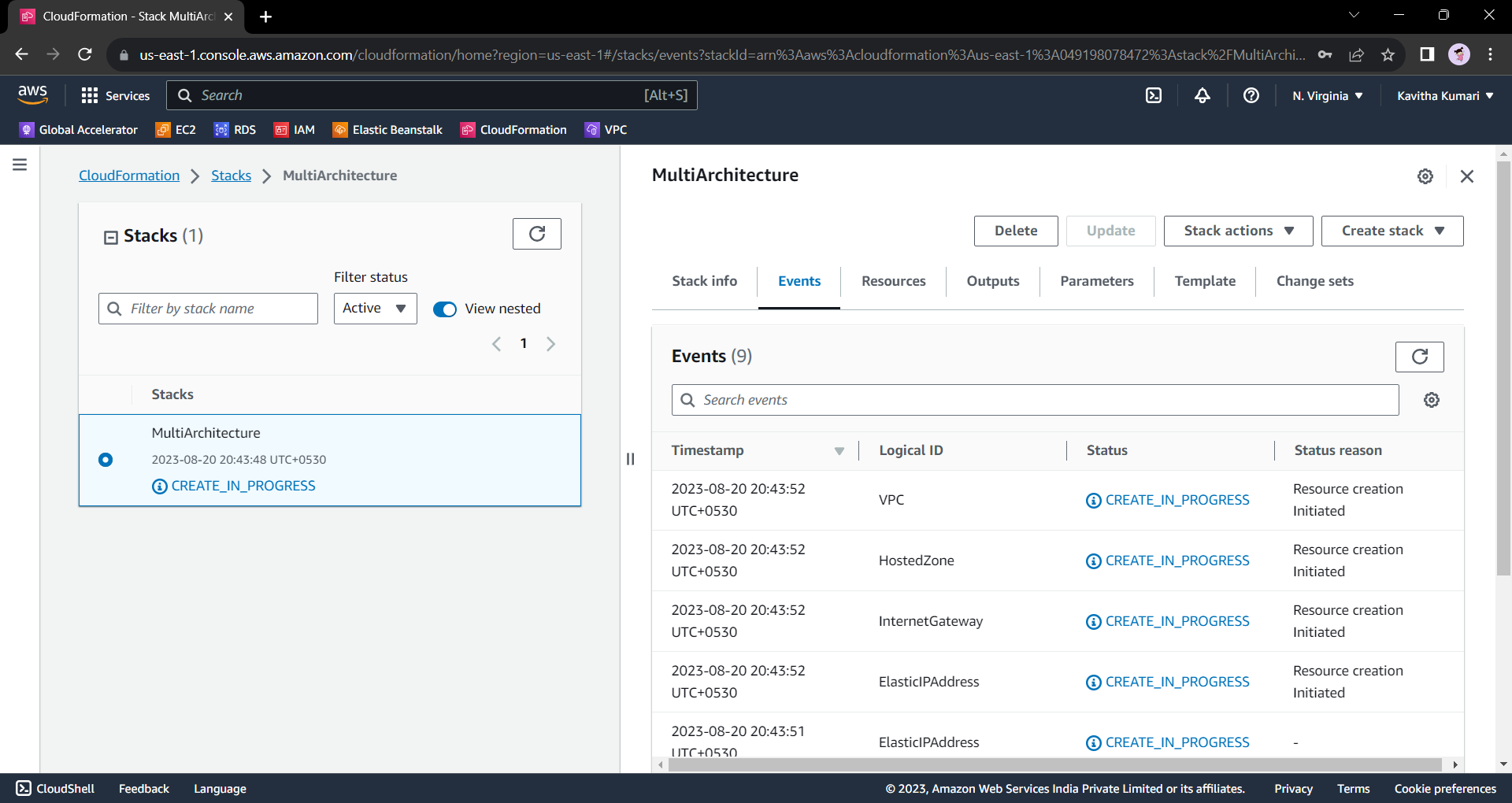




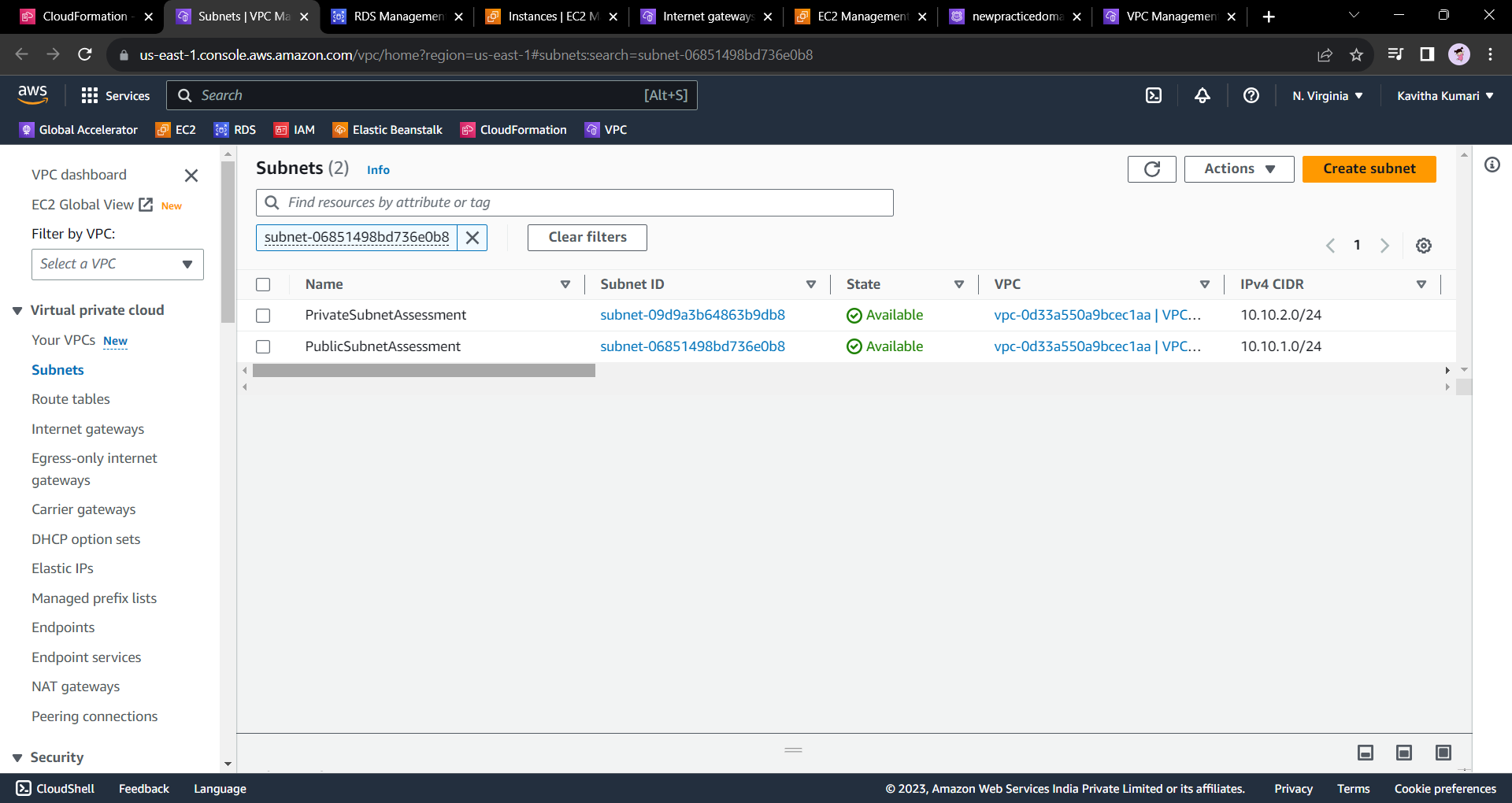
* Click on next

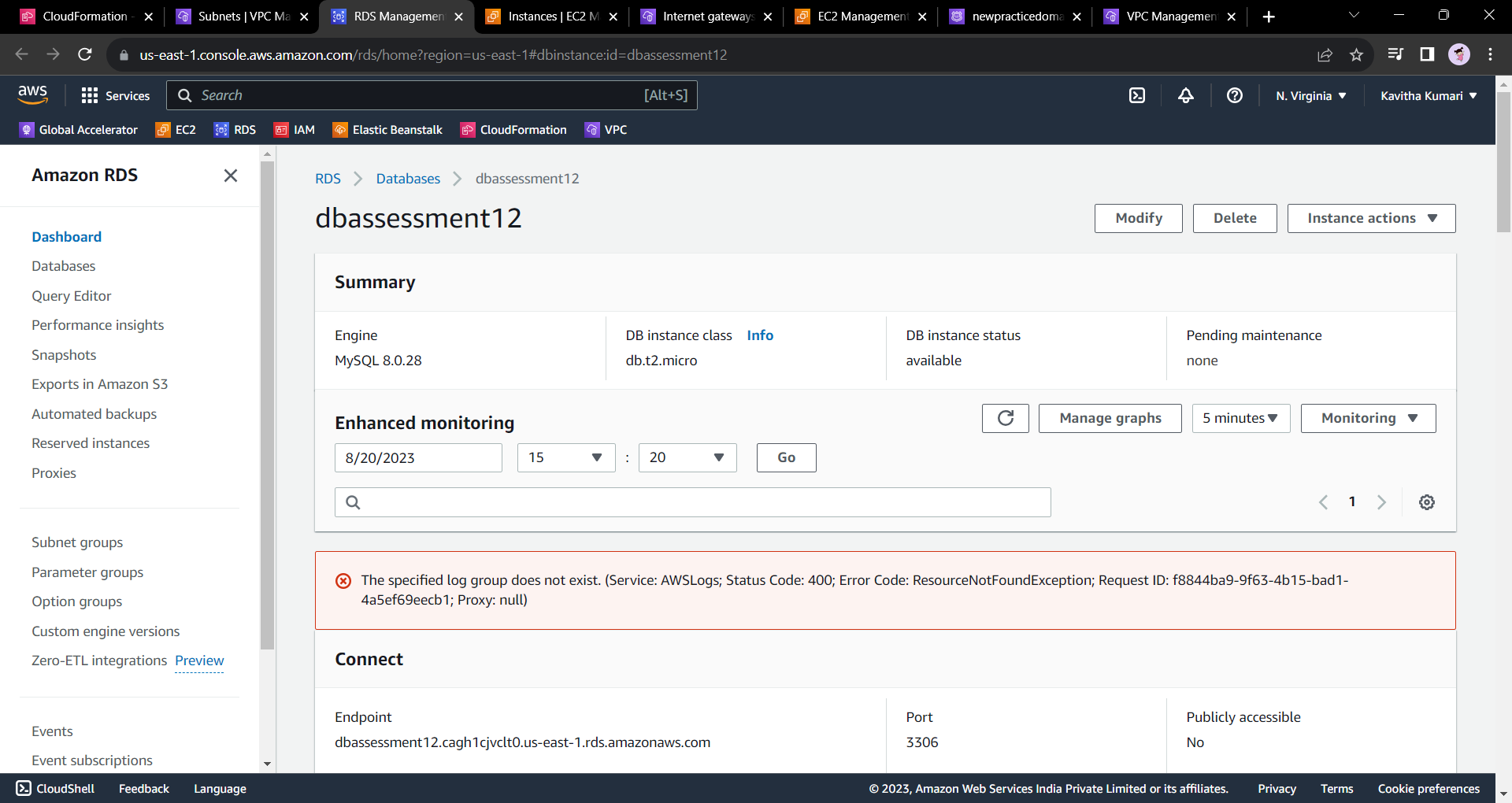


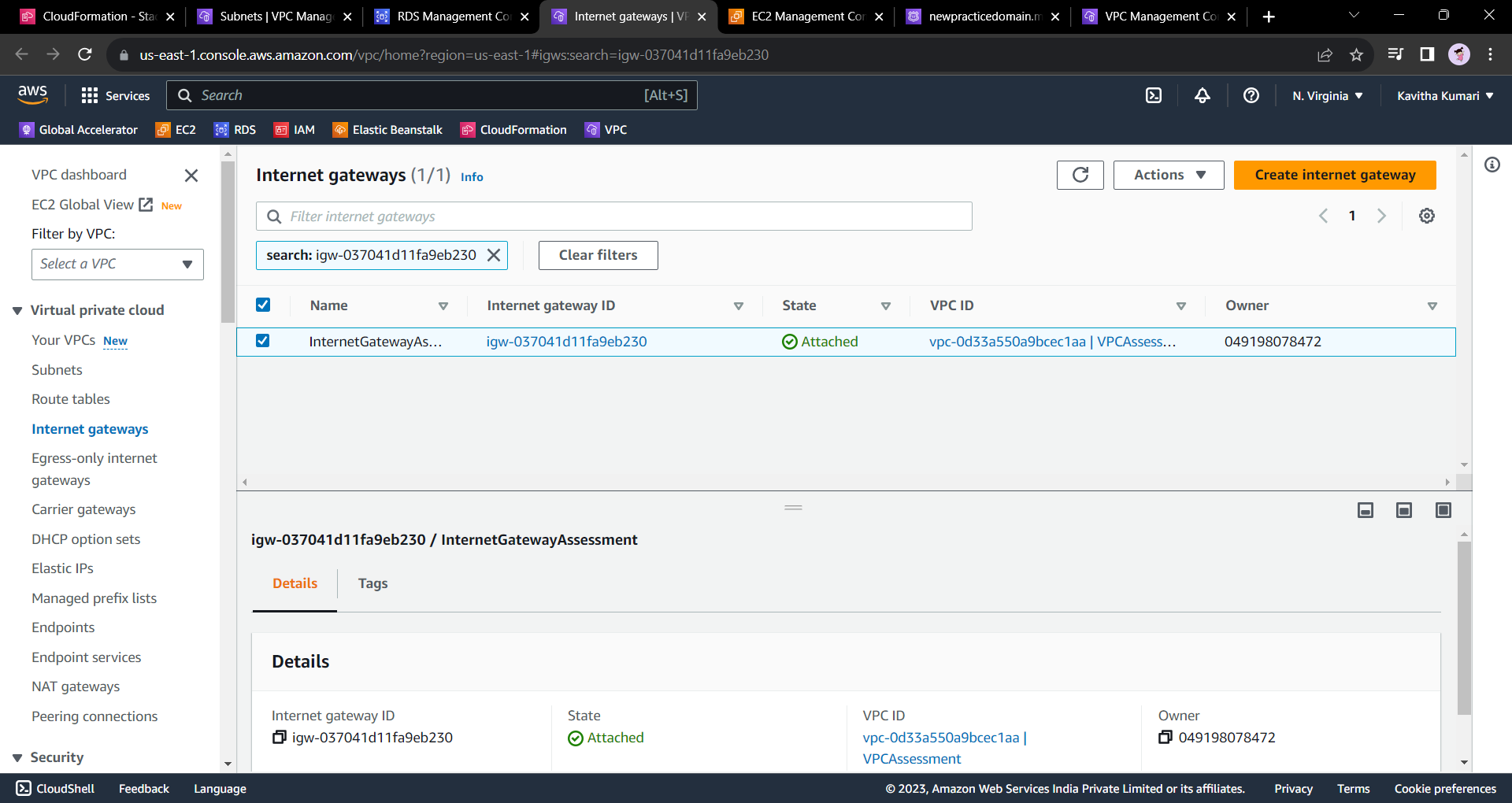
* Leave everything default and click on next.

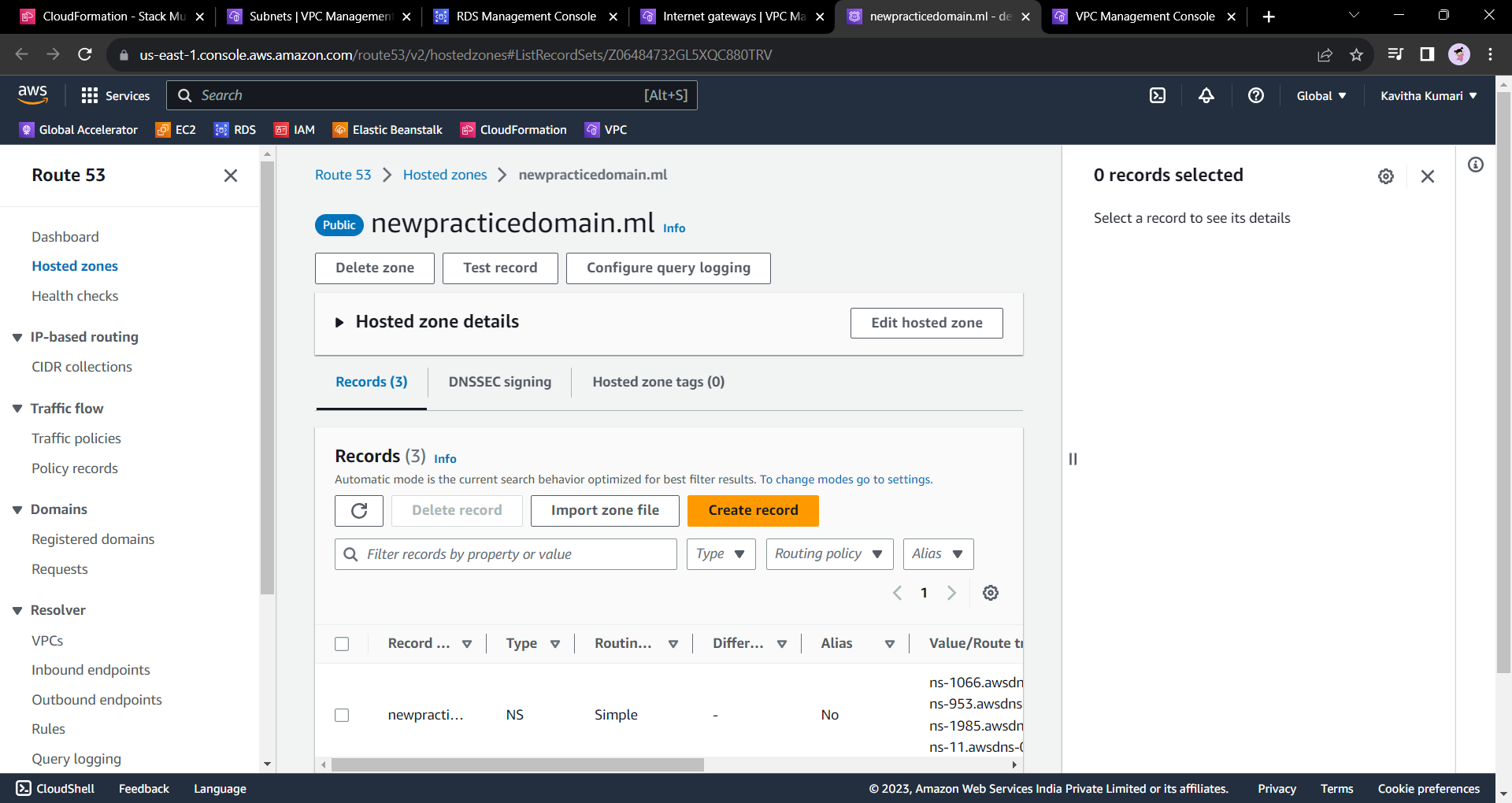


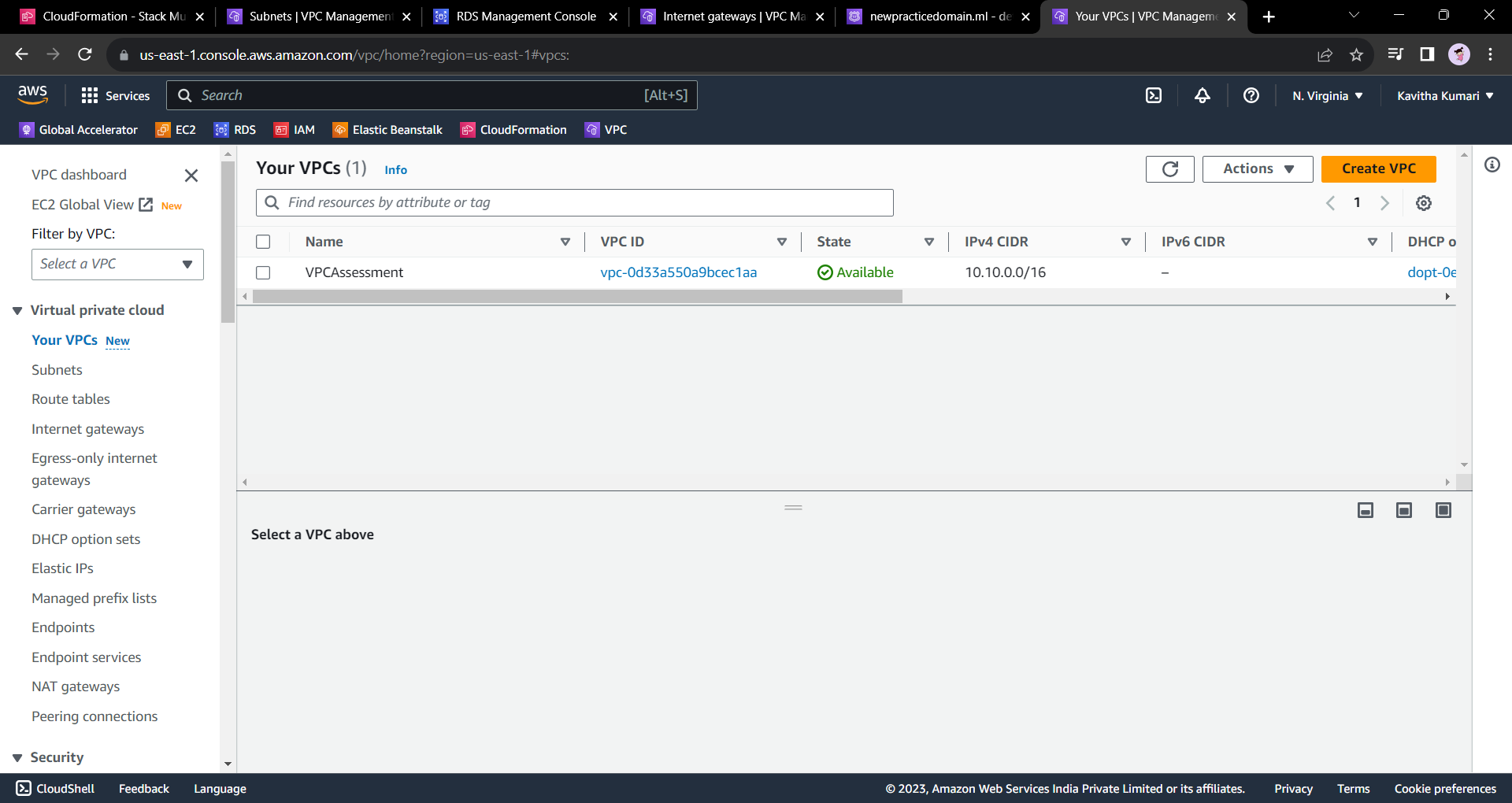
* The creation is in progress.
* Thus, the resources will be created.











**Proposed Solution:**

* **Automated Provisioning:** Provide the development team with a CloudFormation template and instructions on how to deploy the stack. They can launch the stack on their own using the AWS Management Console or AWS Command Line Interface (CLI).
* **Deletion Protection for RDS:** To prevent the RDS instance from being deleted when the stack is deleted, set up a CloudFormation stack policy that prevents updates or deletions of the RDS resource. This ensures that the development team can safely delete the stack without affecting the RDS instance.
* **Testing and Validation:** The development team can test their code on the launched instances in the designated subnets. They can access the instances using the provided IP addresses. The Route 53 hosted zone can be configured to route traffic to the WebInstance using the Public IP or an Elastic Load Balancer (not included in the template).

By providing the CloudFormation template and the solution approach, you empower the development team to create and manage their testing environment independently without needing the involvement of system administrators. This approach encourages automation, reduces manual intervention, and promotes faster development and testing cycles.

----End----