**CloudFormation**

* AWS CloudFormation is a service that provides users with a simple way to create and manage a collection of AWS resources by provisioning and updating them in an orderly and predictable way.
* Basically, it is a service that helps us to model and set up our AWS resources so that we can spend less time on managing resources and show more focus on the application that runs on top of it.
* By using AWS CloudFormation it enables you to manage your complete infrastructure or AWS resources through a text file that will be a JSON / YAML file called as **Template**.
* All the resource required by a user in an application can easily be deployed.
* You can reuse your template to replicate your infrastructure in multiple environment.
* For making it reusable, use parameters, mapping and conditions sections in the template so you can customize your stacks when you create them.

**Workflow**

1. Create / use the existing CloudFormation template using JSON/YAML format so that one can develop a template based on the requirements i.e. number of EC2 instance, load balancer, Windows server, Database server and applications used in template.
2. Now the written code template will be saved locally or in S3 bucket.
3. Use AWS CloudFormation to create a stack on your template.
4. Now, CloudFormation analysis the template, validates it and then it identifies the dependencies between them and then it starts provisioning the resource one after the other.

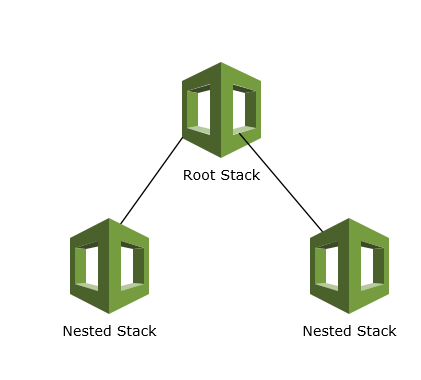
**Concepts**

1. **TEMPLATE**

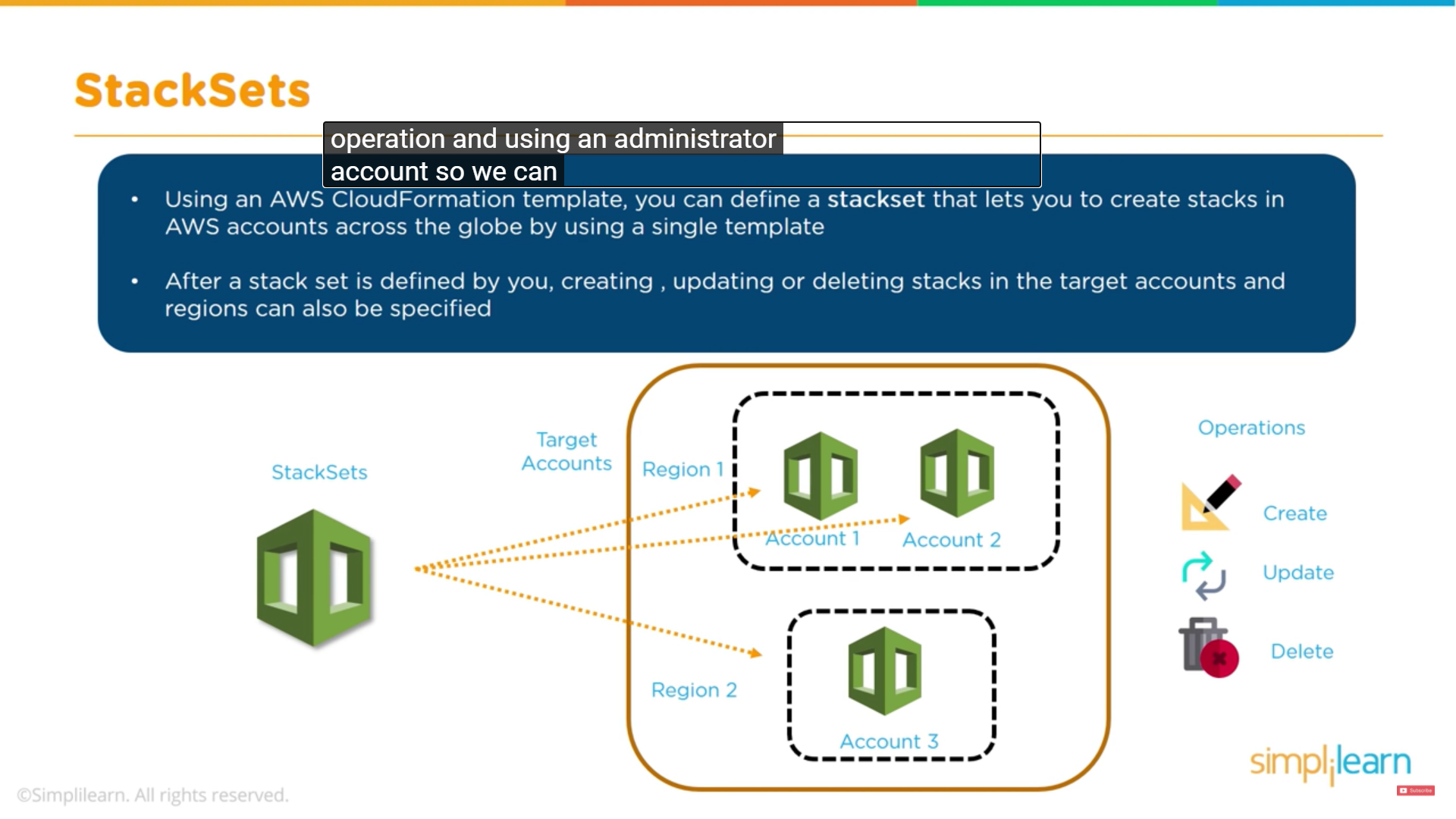
* In CloudFormation it is a formatted text file written in JSON/YAML that describes the AWS infrastructure.
* For creating, viewing or modelling templates you can use AWS CloudFormatation Designer or any text editing tool.
* Main 9 Objects of the Template.

1. **Format Version –** It identifies the capabilities of the template. The latest template format version is 2010-09-09
2. **Description –** It is and optional section and enables you to include comments about your template.
3. **Metadata –** This section is used to include arbitrary JSON or YAML objects that provide details about the template
4. **Parameters –** Parameters enable you to input custom values to your template each time you create or update a stack
5. **Mappings –** This section matches a key to a corresponding set of named values
6. **Conditions –** This section contains statements that define the circumstances under which entities are created or configured
7. **Transform –** This section specifies one or more macros that AWS CloudFormation uses to process your template.
8. **Resources –** The required Resources section declares the AWS resources that you want to include in the stack, such as an Amazon EC2 instance or an Amazon S3 bucket.
9. **Outputs -** This section declares output values that you can import into other stacks, return in response, or view on the AWS CloudFormation console.
10. **STACK**

* Stack is a collection of AWS resources and can be managed in a single unit.
* CloudFormation’s template defines a stack in which the resources can be created, deleted or updated in a predictable way.
* A stack has all the resources like web server, database server, etc. that can be required to run a web application.
* NESTED STACK –
* It is hierarchy of stacks.
* By using the CloudFormation stack resource, one can create a nested stack within another stack.



* WINDOWS STACK –
* Windows stack gives your ability to update and configure your own stack in windows instance.
* By using the CloudFormation, you can create Windows stakc for Amazon EC2’s Windows AMI (Amazon Machine Images).
* STACK SETS -
* By defining a StackSet you can create stacks in AWS account across the globe by using a single template.
* After a StackSet is defined, you can create, update and delete stacks in the target accounts and regions that you specify.



**CloudFormation Access Control**

1. **IAM User Access**

* By using IAM User Access, CloudFormation can have access control for users and ensure that only IAM users can create, update and delete stacks.

1. **Service Role**

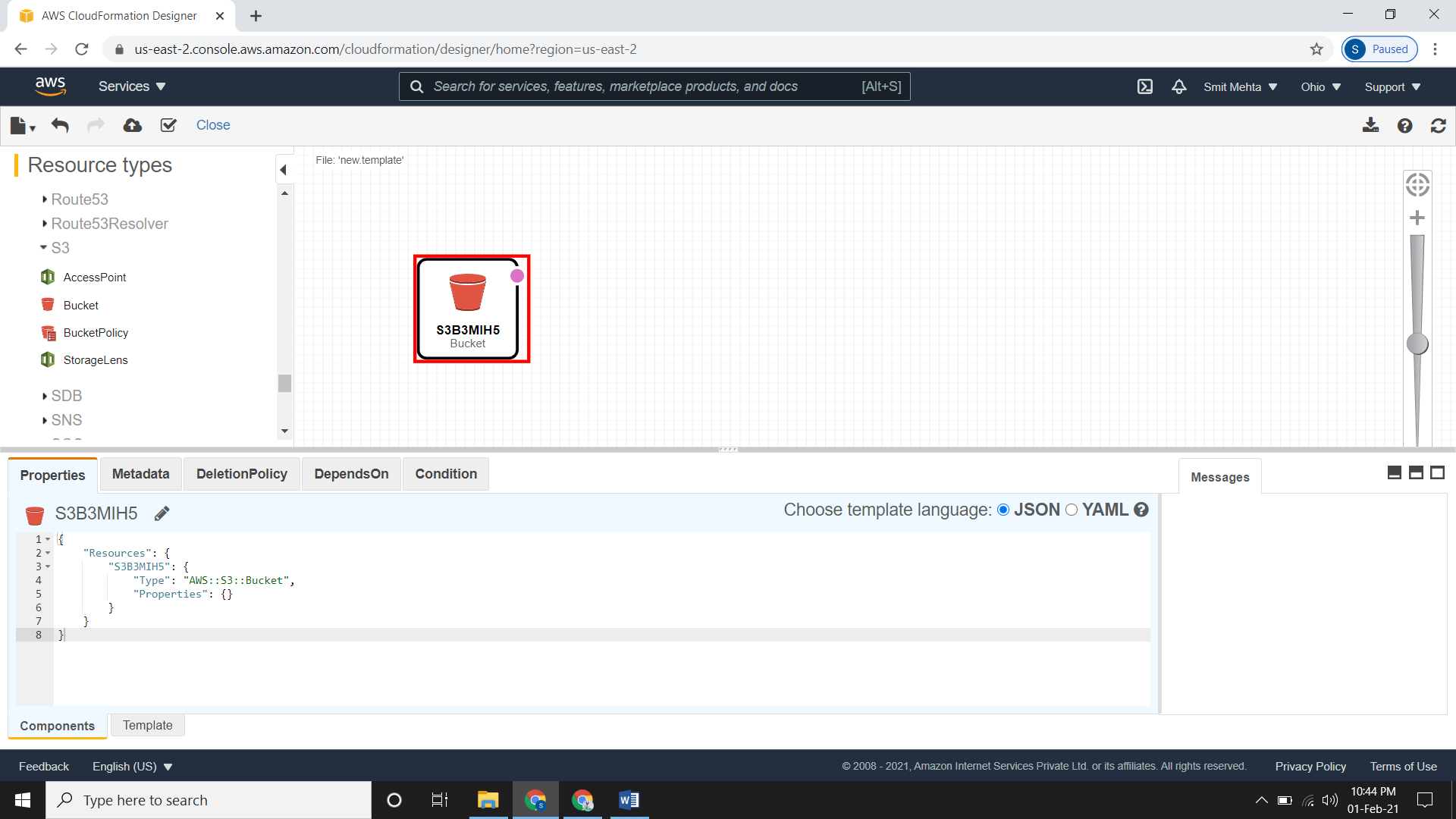
* On the user’s behalf a service role allows AWS CloudFormation to make calls to resources in stack.

1. **Stack Policy**

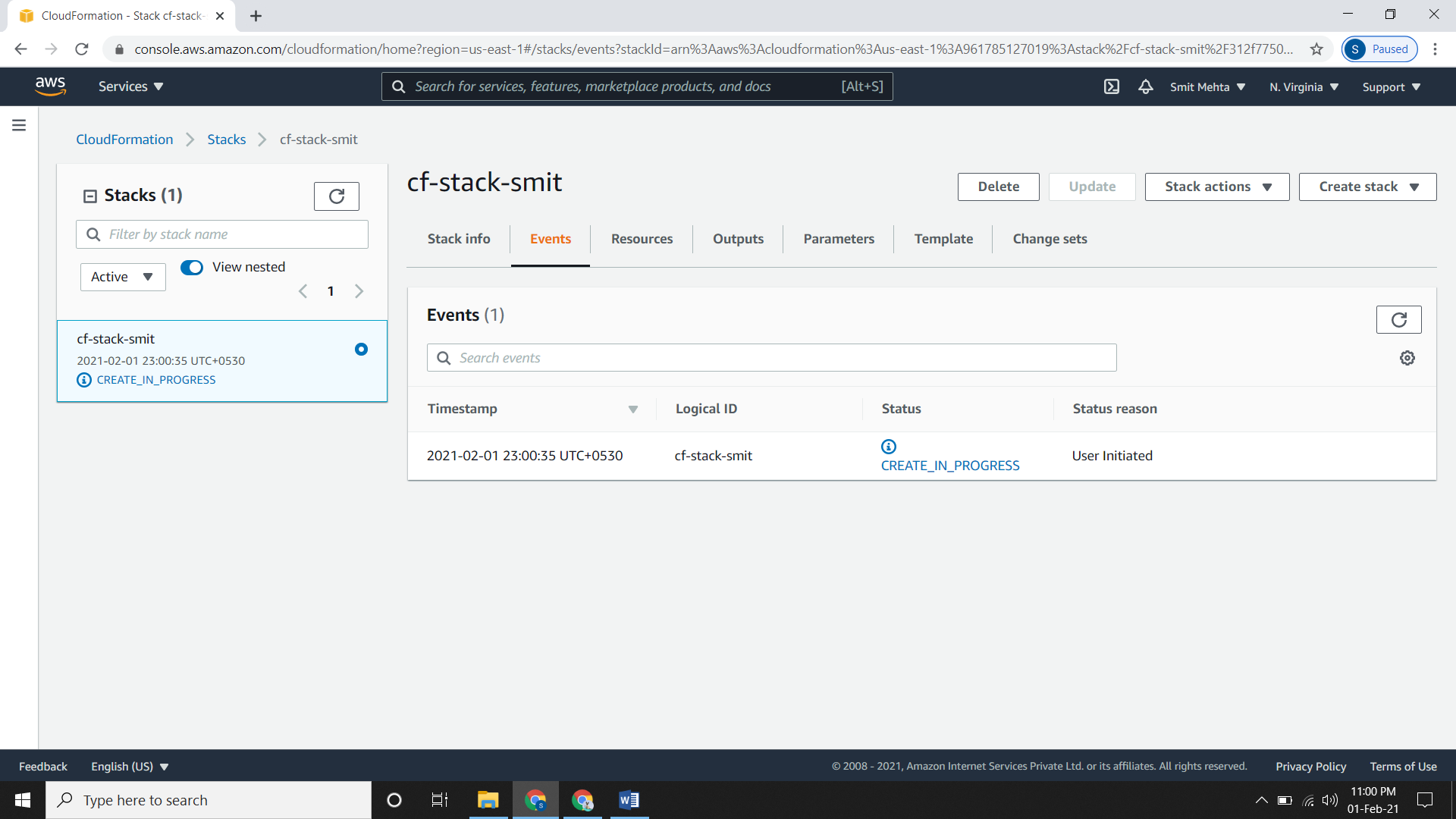
* It is applied to all AWS CloudFormation users who attempt to update the stack. Here you cannot include different users with different stack policies.

**Creating a Template**

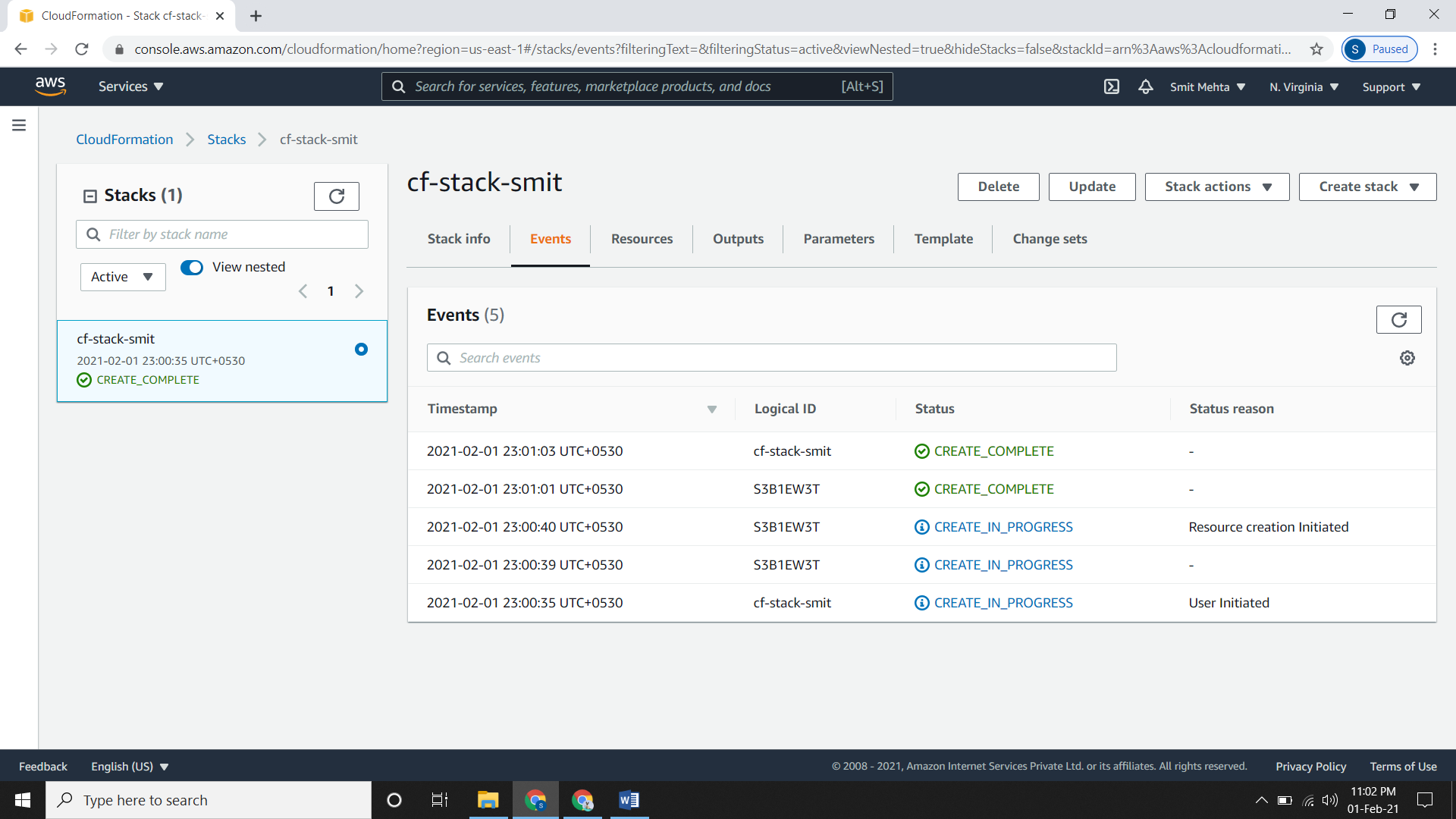
* Go to the **CloudFormation** console and click on **Designer**.
* Under **Resource types** click on the resource you want, drag and drop on the main window.
* Here we will create a template for **S3 Bucket**.



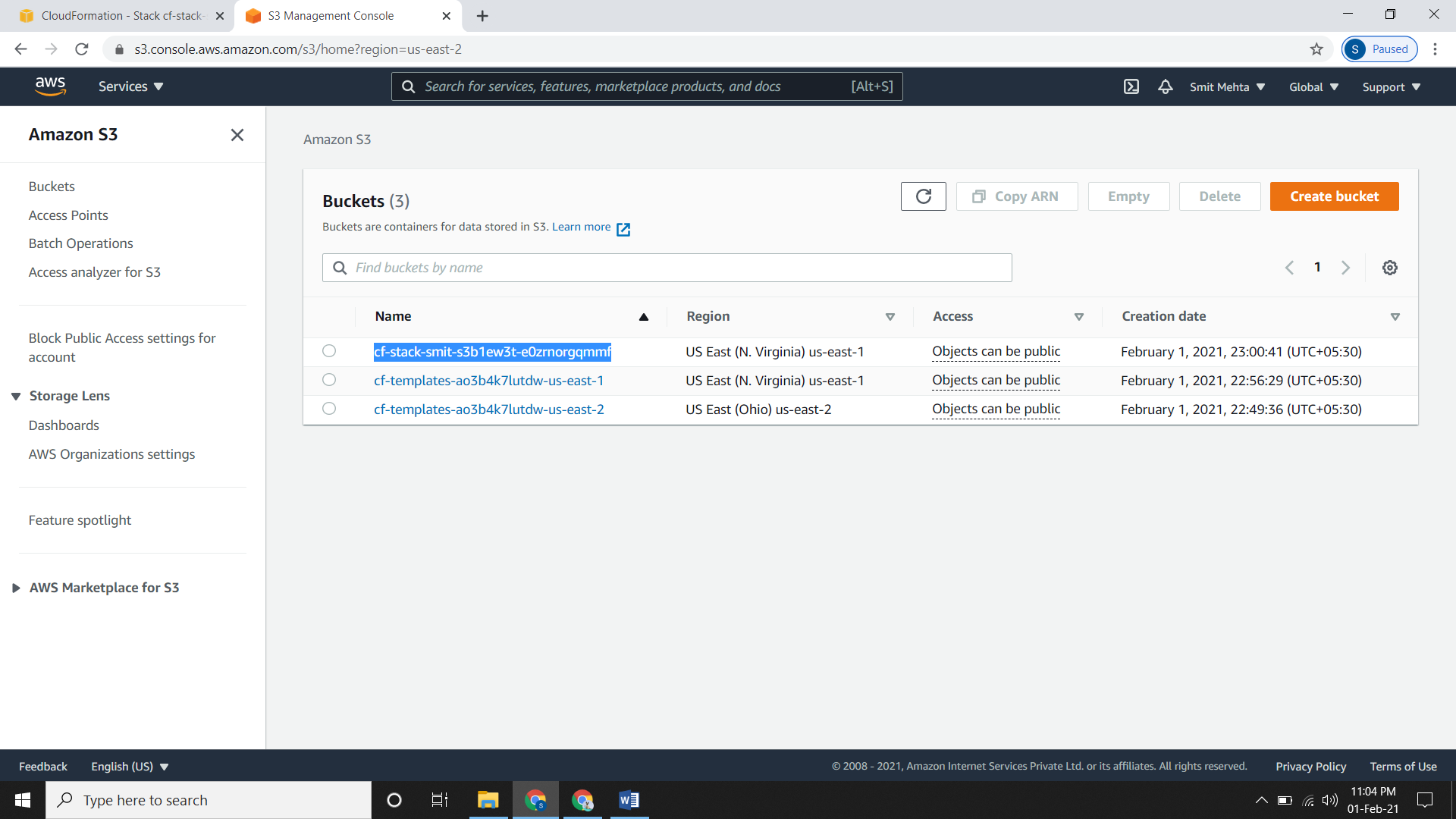
* Here you can change the name and other non-default things.
* After updating all thing click on the **upload** **type** icon as **create** **stack**.
* It will redirect you to the **Create stack** page, give your detail here and then click on next.
* Give a **name** to the stack and under **Configure stack options** configure it accordingly and then create it.



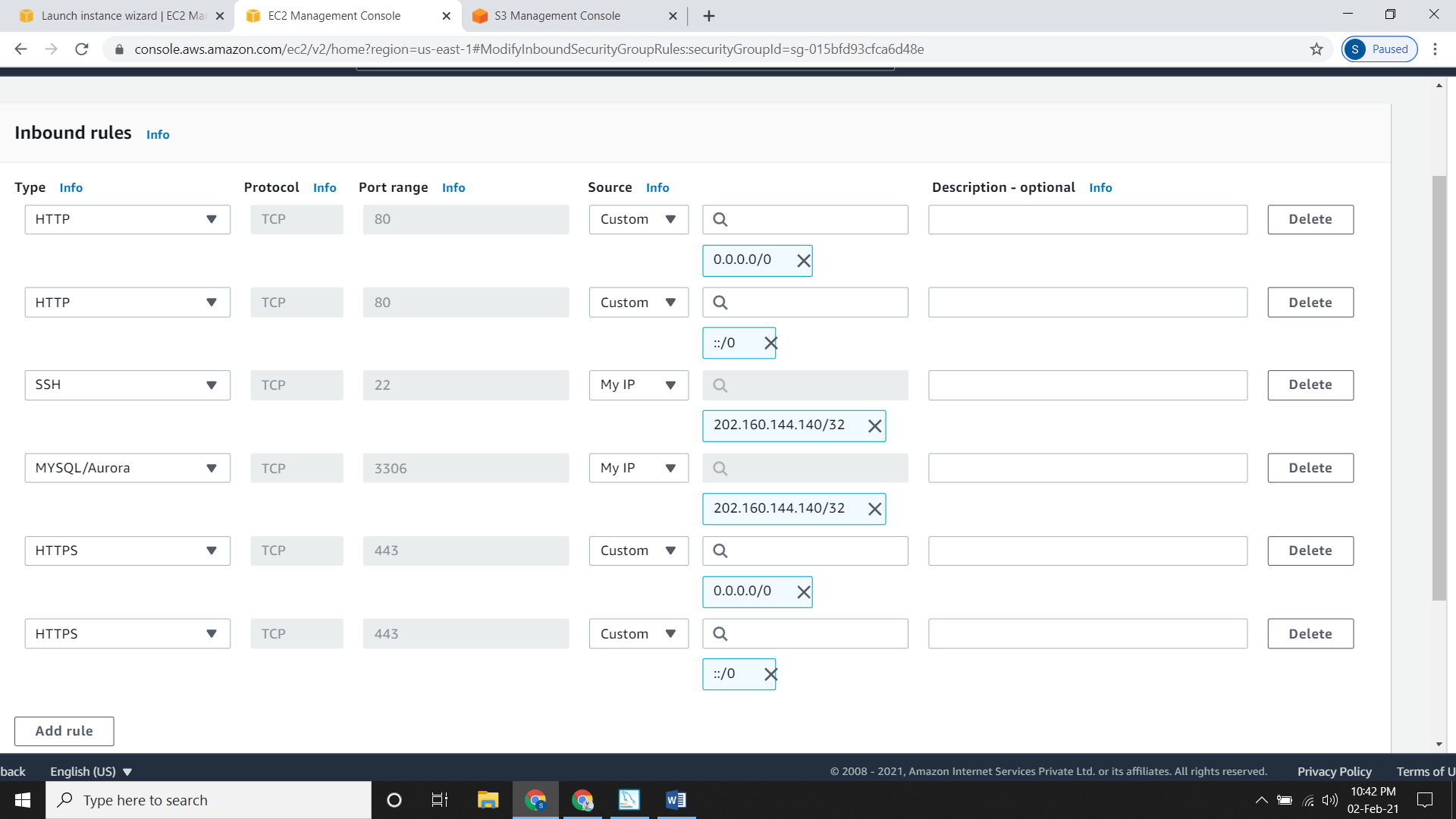
* It will take a few minutes to be in ready state.

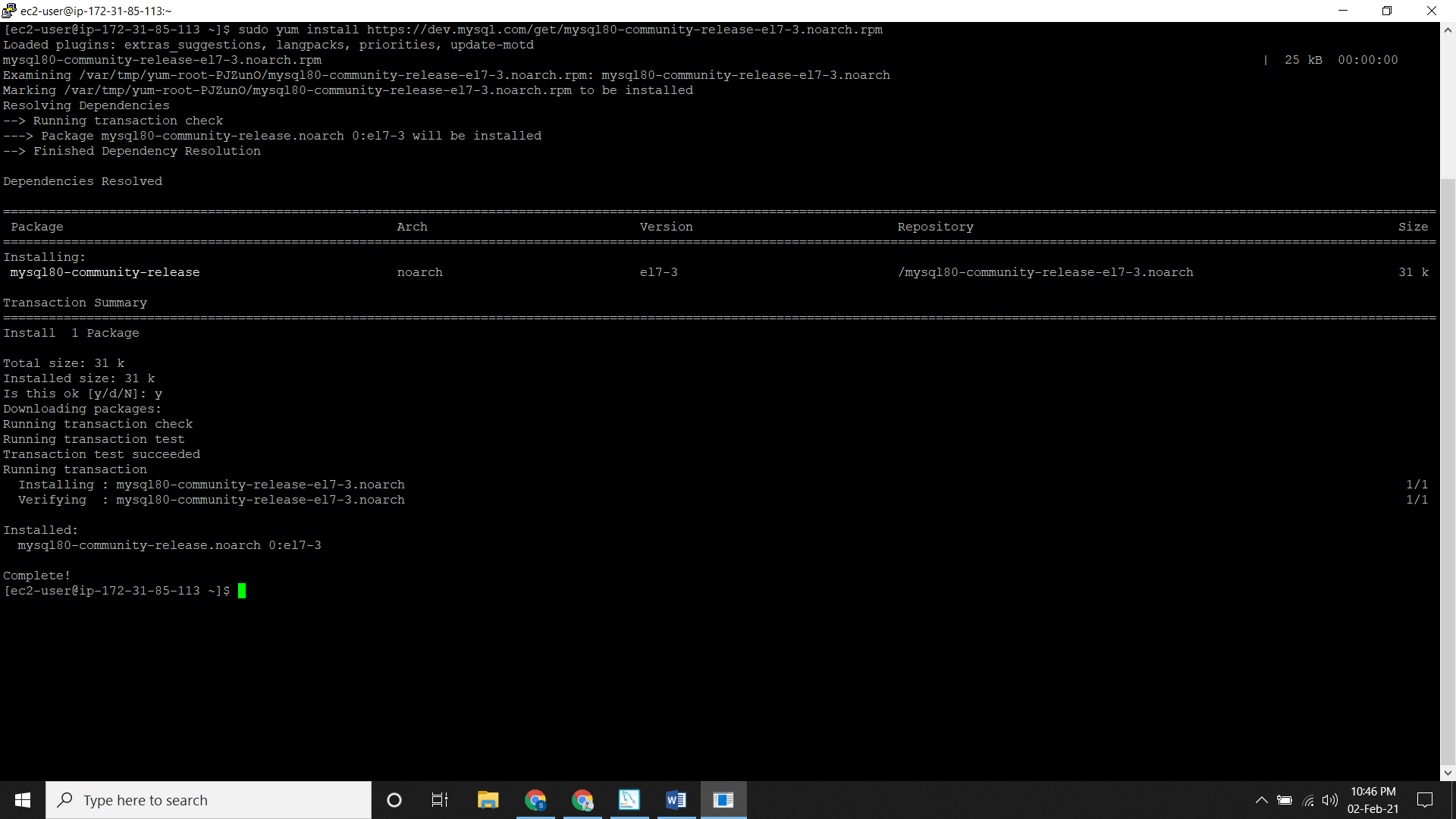


* Checking the S3 bucket

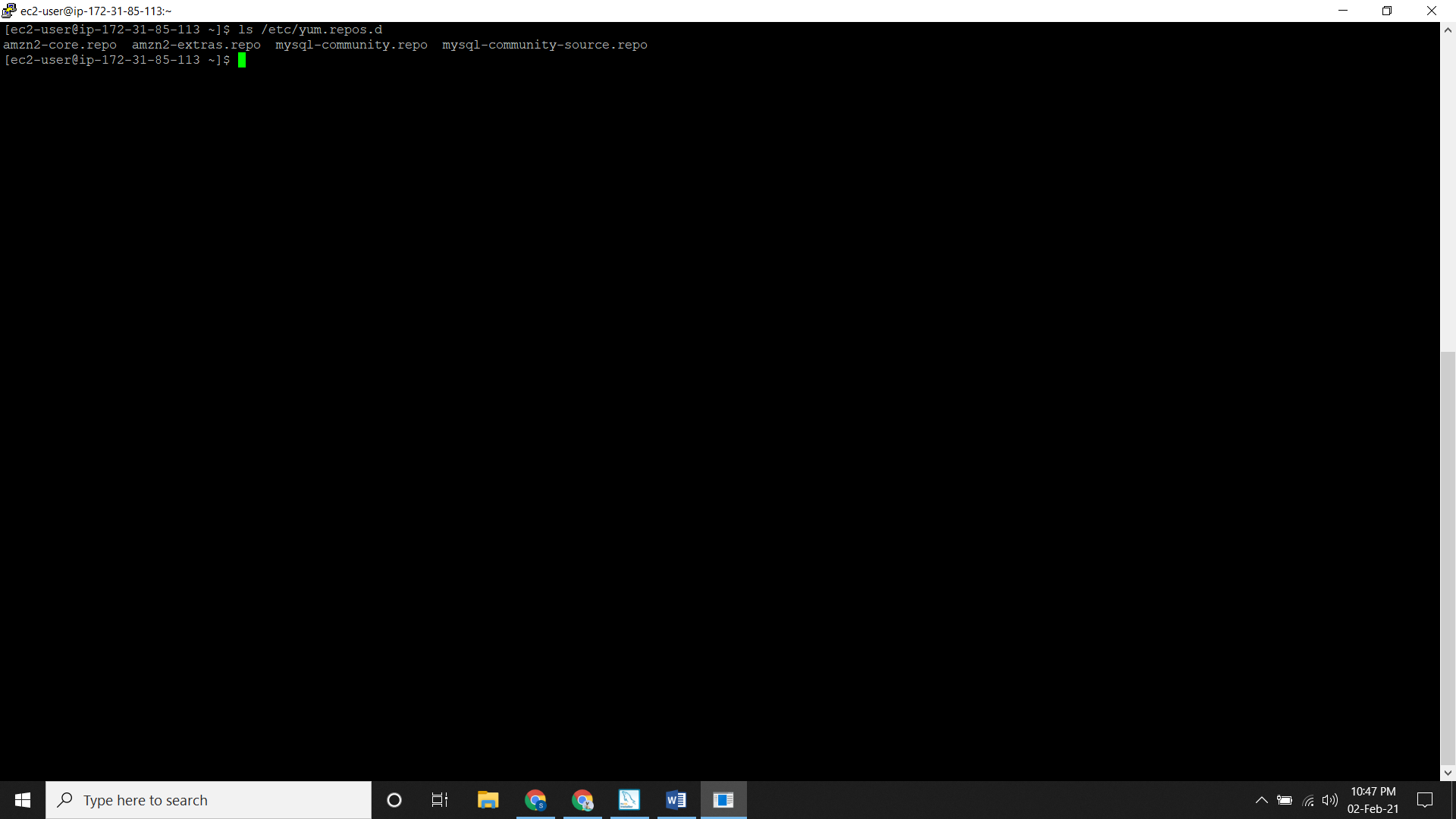


**Install SQL on EC2 Instance**

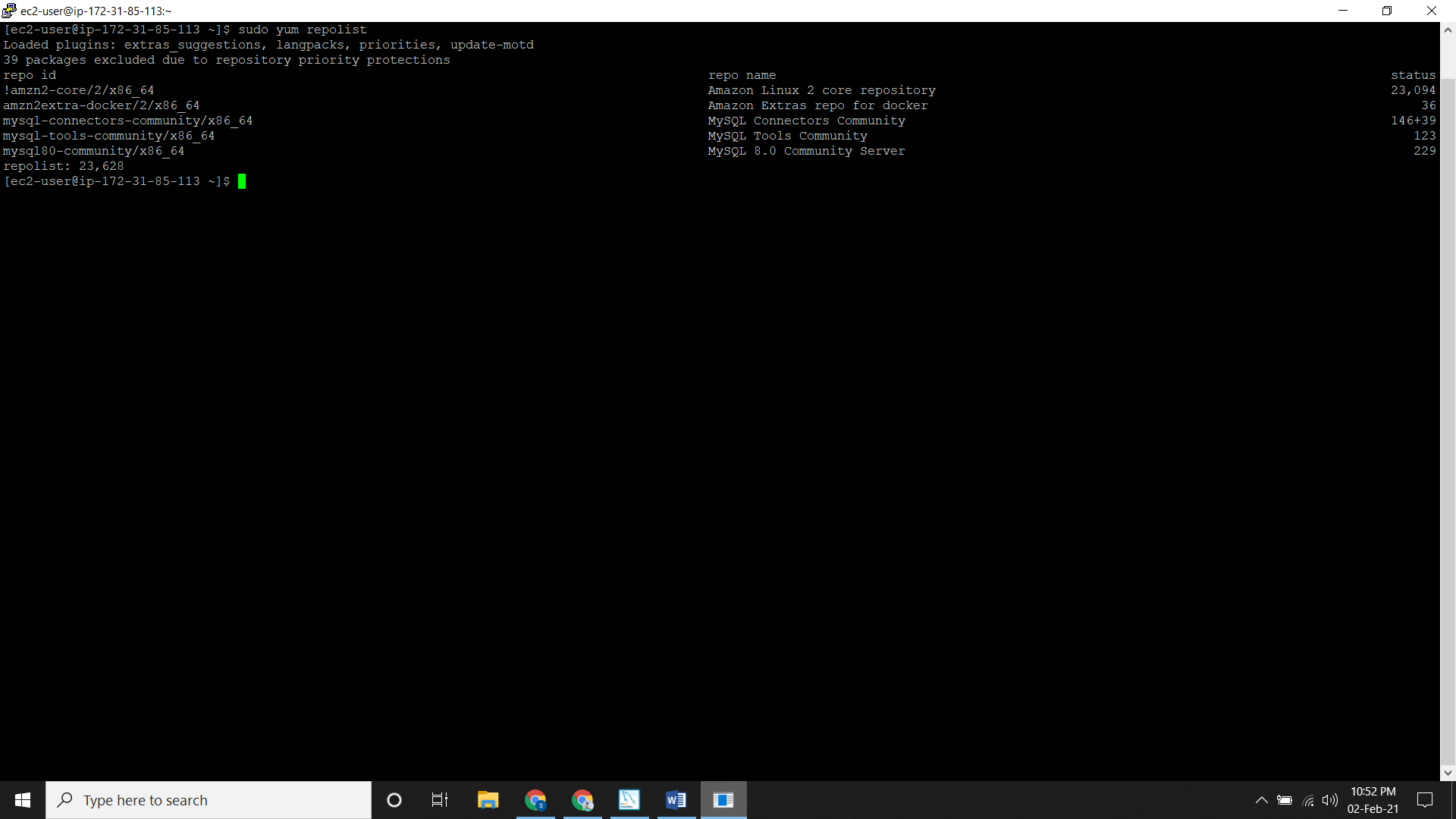
* Go to EC2 dashboard, click on **Launch Instances**
* Leave the configuration part as default but in security group, create a new security group.
* Review it, give a key pair and then create it.
* Connect the instance using Putty.
* Installing mysql80-community



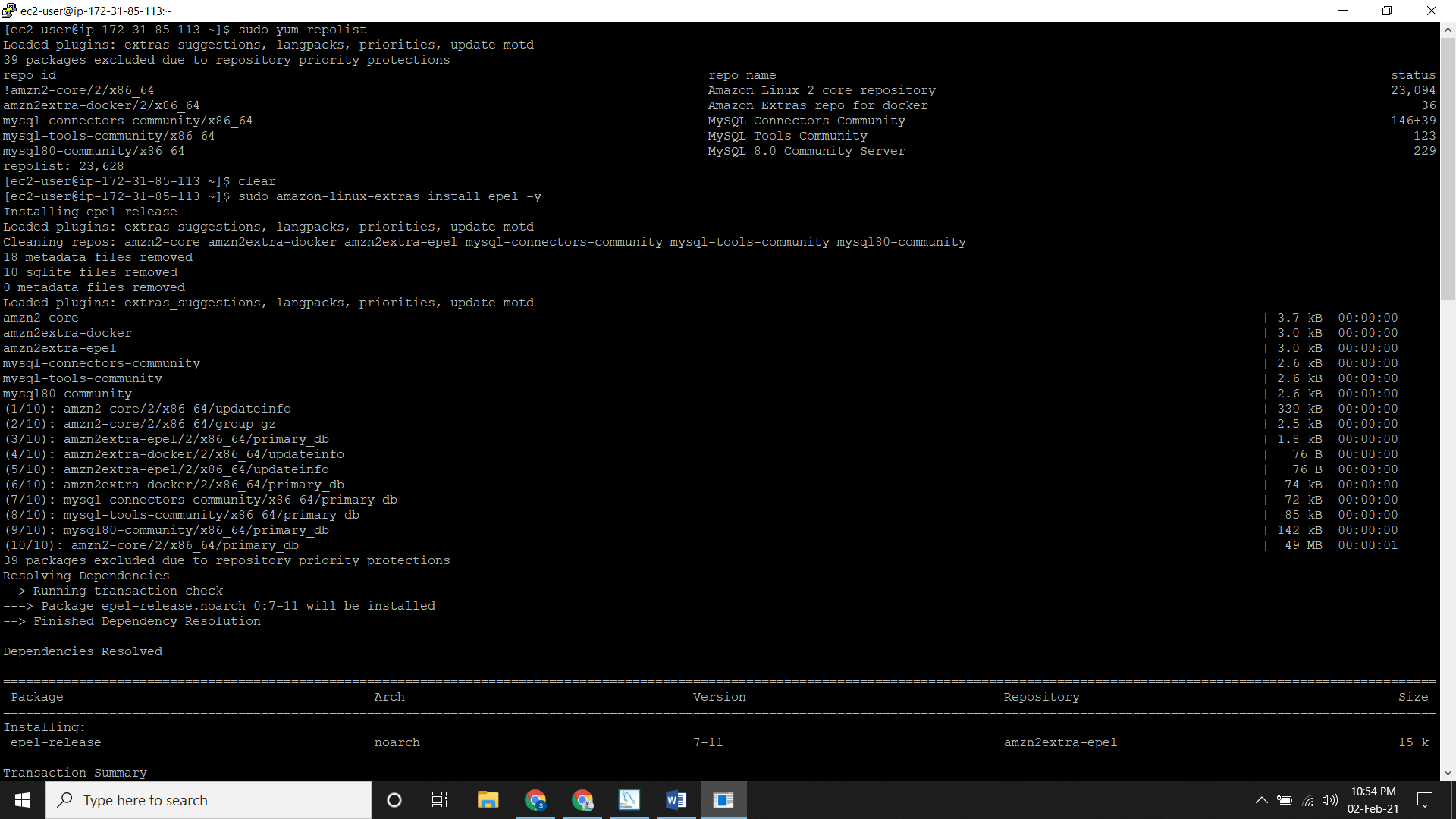
* Checking the directories

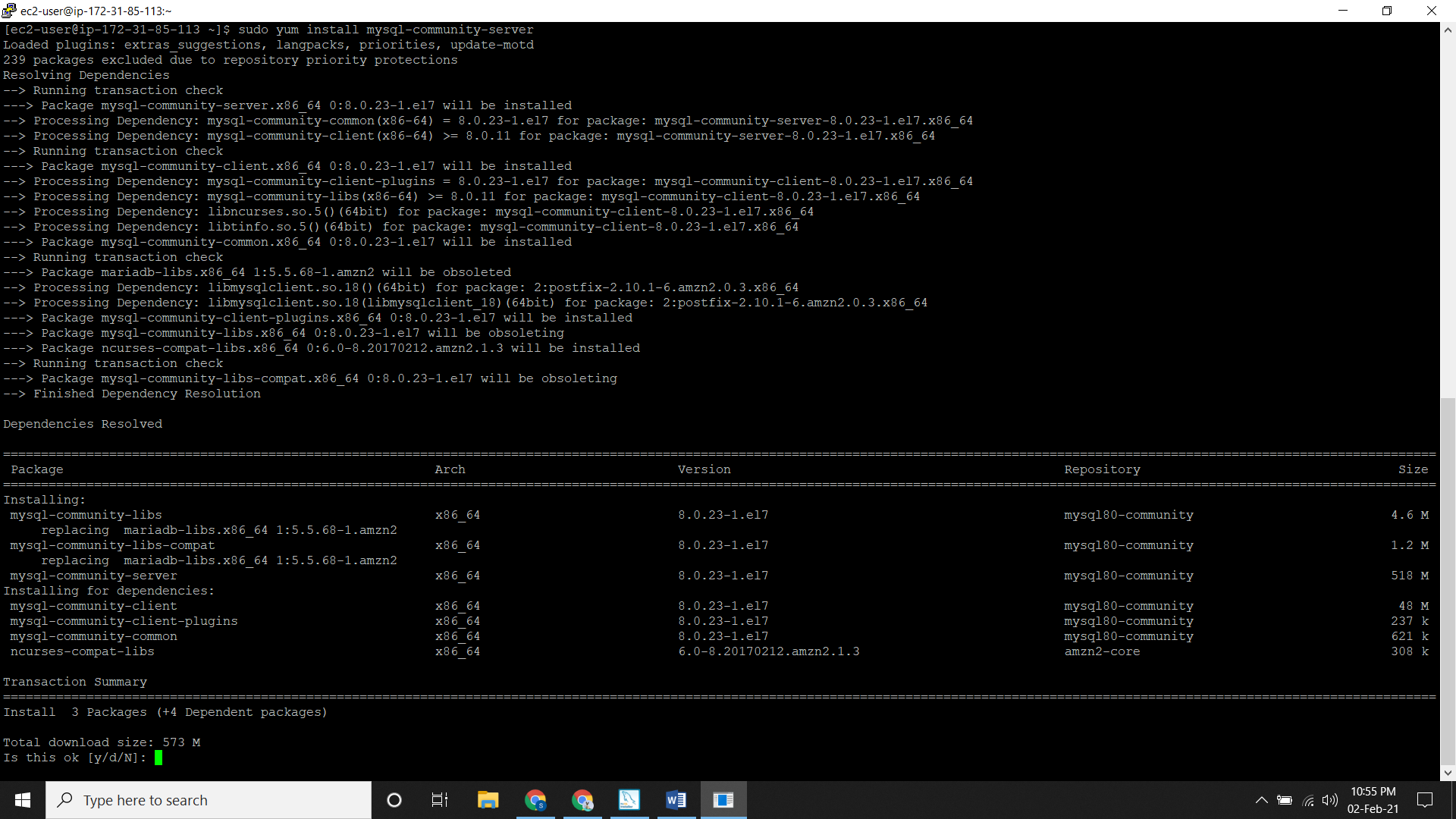


* Listing the configured repositories

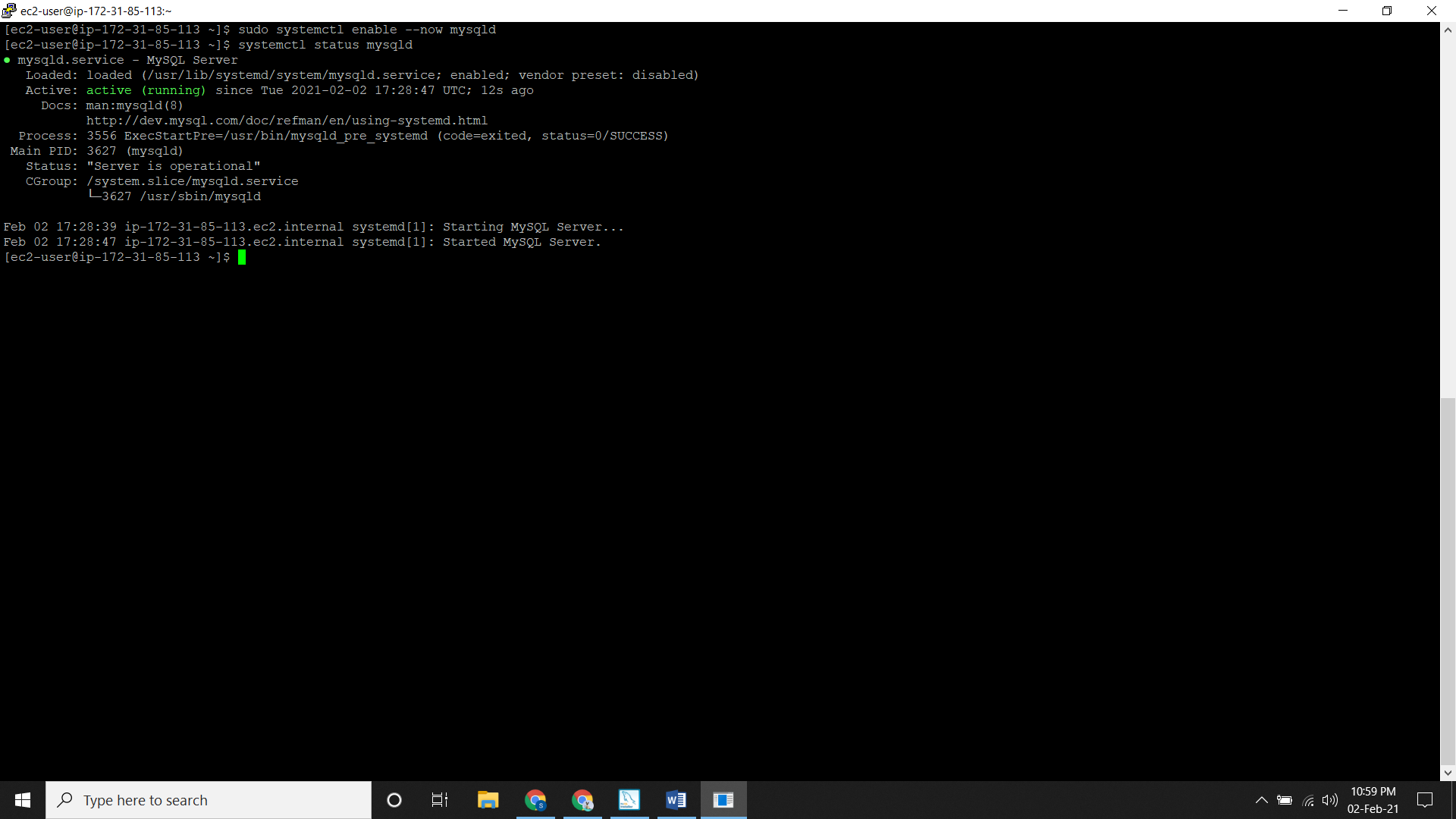


* Installing MySQL





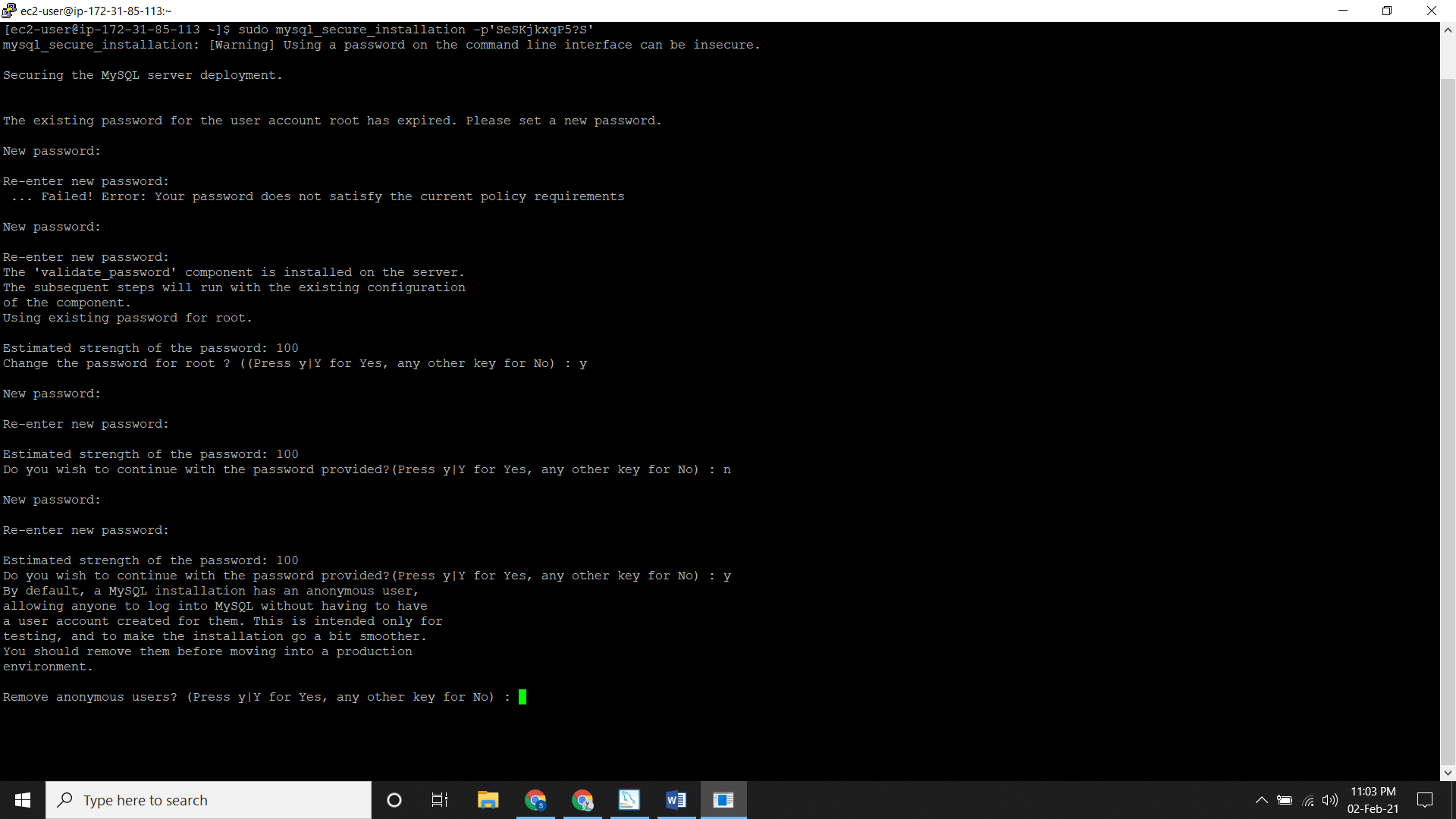
* Starting MySQL server and checking it.



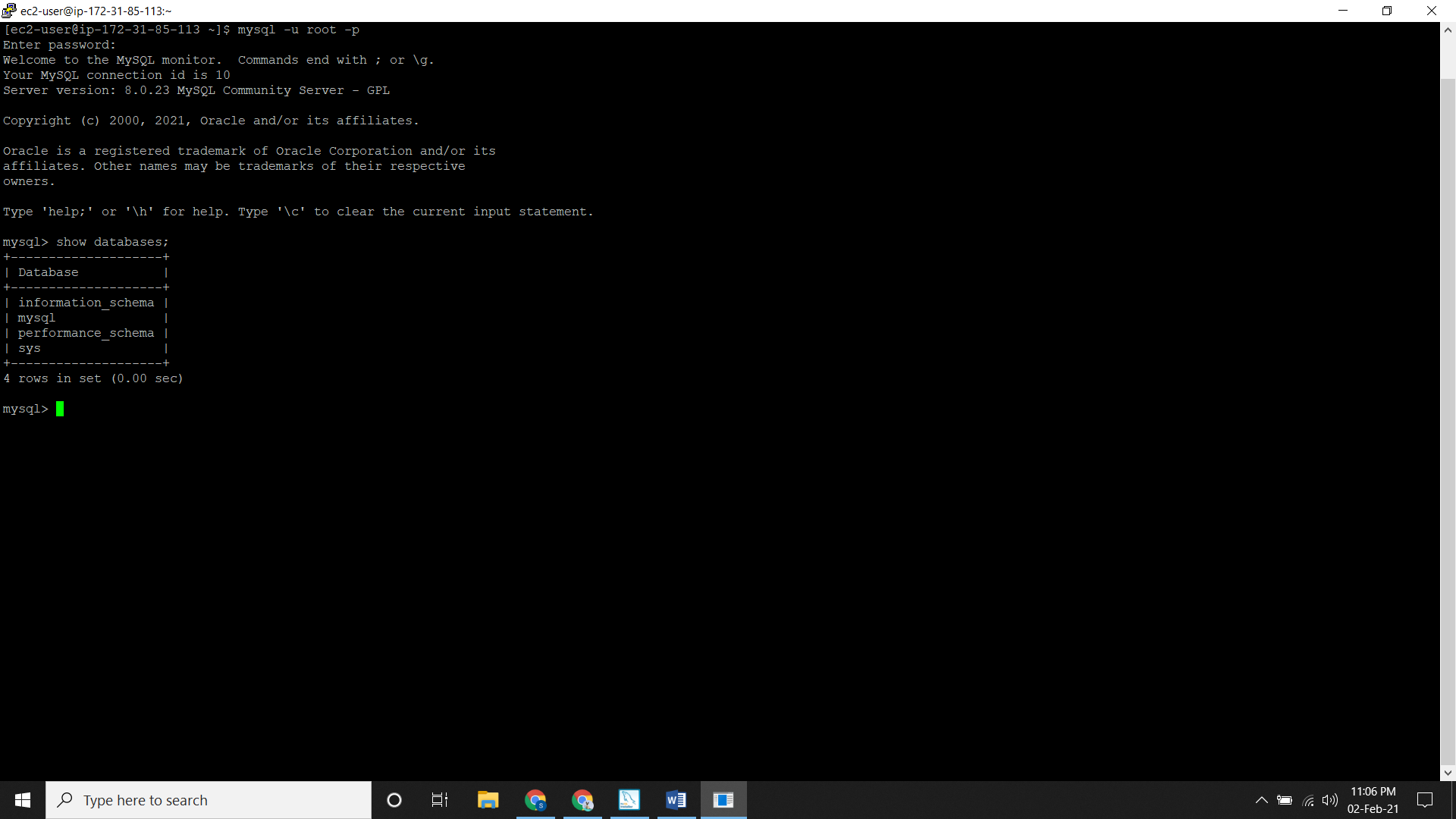
* Getting the temporary password.



* Using this password to set a new password.

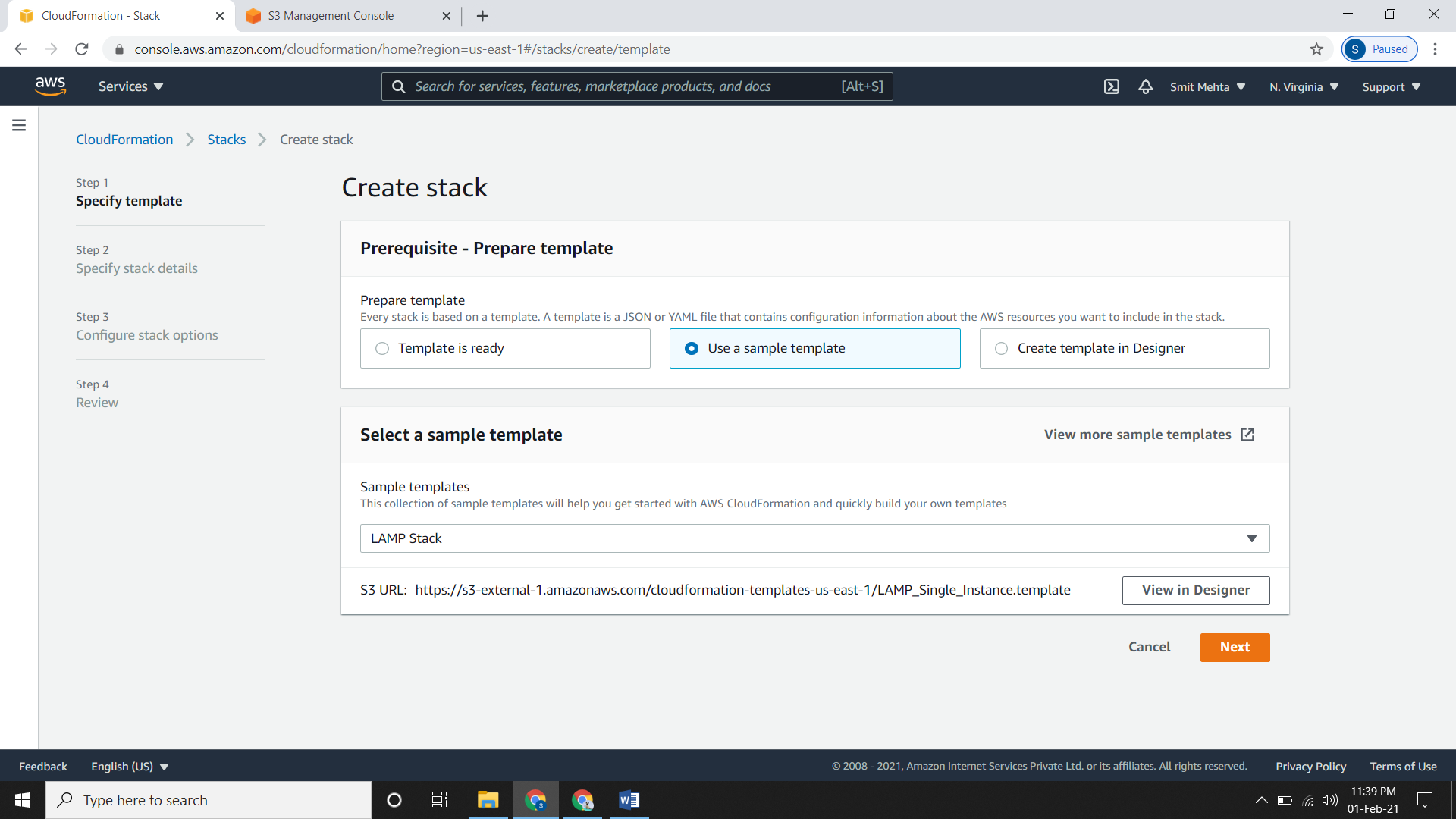


* Entering into MySQL server using the new password

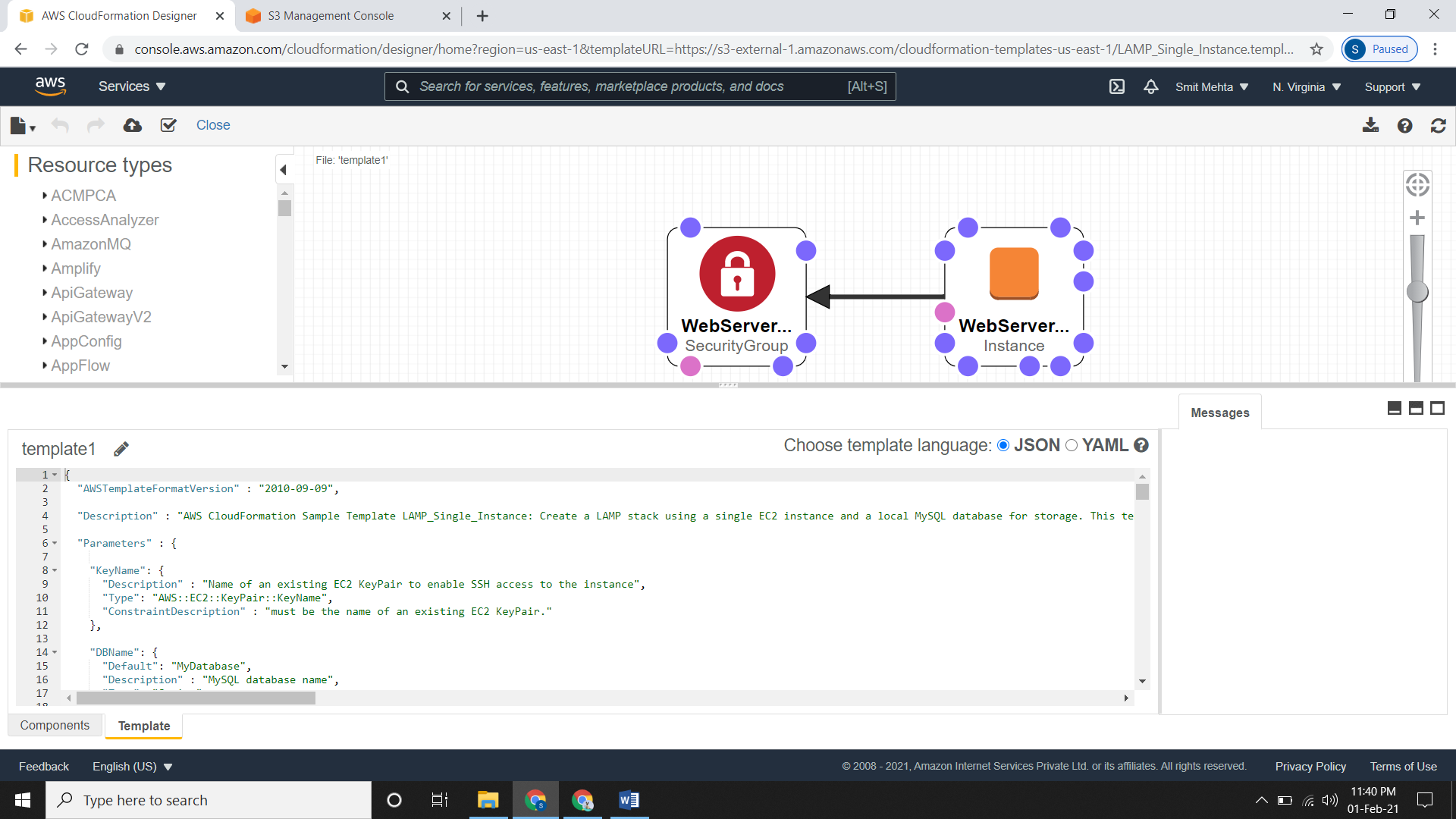


**Creating LAMP STACK**

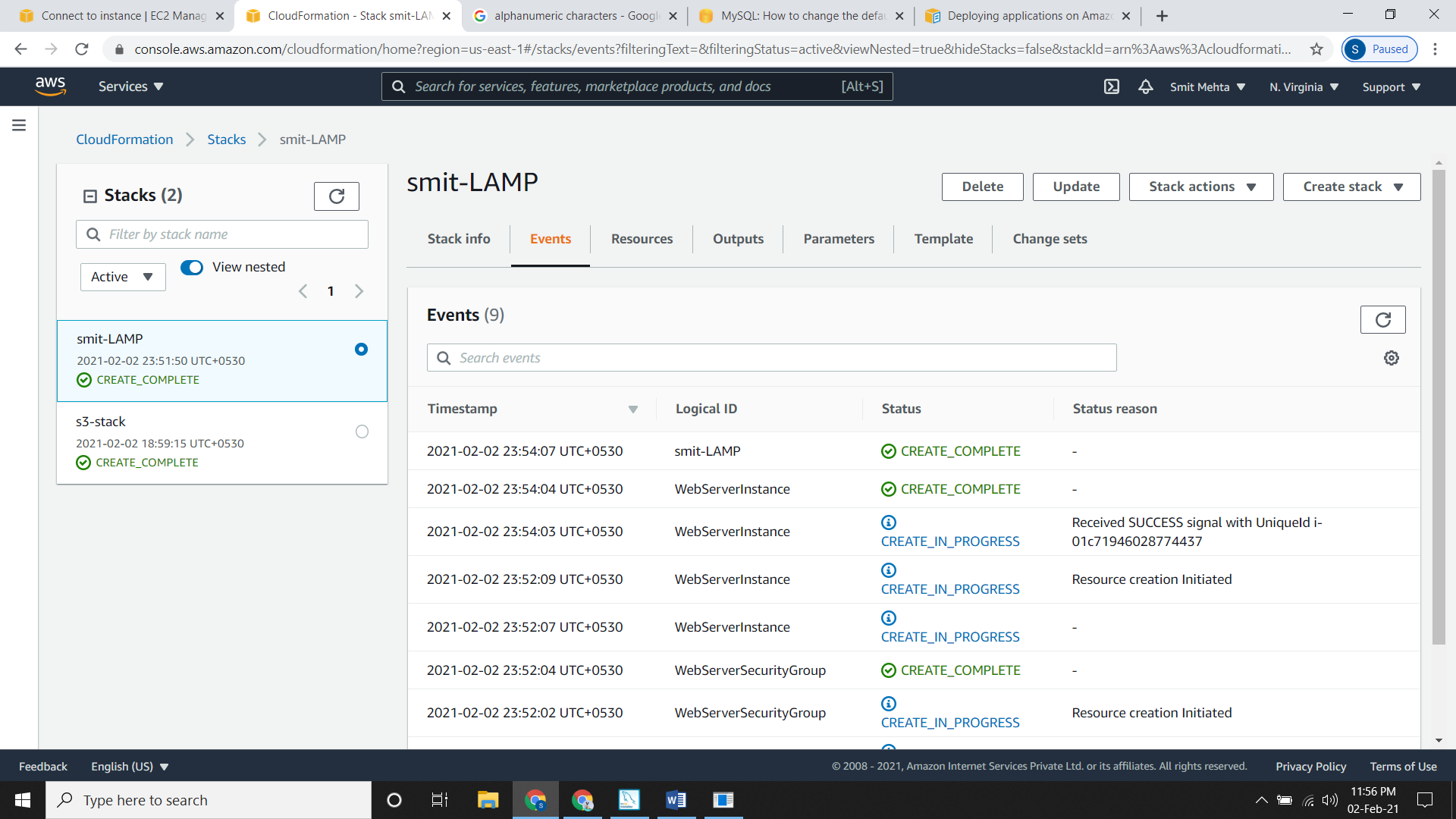
* A **LAMP Stack** is a sample template in AWS.
* Go to the **CloudFormation** console and click on **Create Stack.**
* Since it is inbuilt stack, under **Choose a template** click on Select a sample template and select the **LAMP Stack**.



* Now click on **View/Edit template in Designer.**

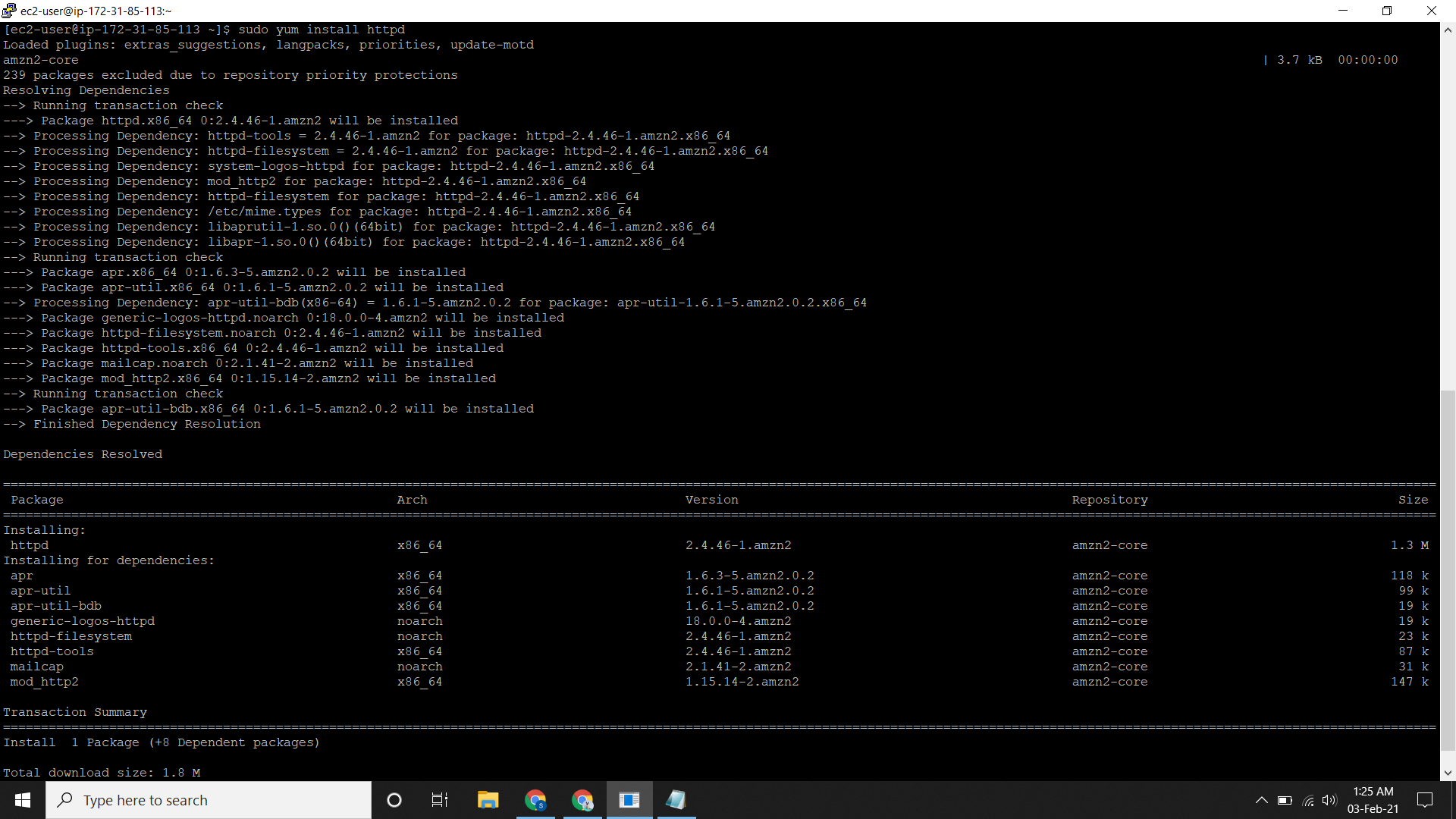


* A **LAMP Stack** is basically an EC2 instance with Linux, Apache, MySQL and PHP installed on it.
* Now click on the **upload** **type** icon as **create** **stack.**
* Leave the **Create Stack** page as default and click next.
* Here you need to give a **name**.
* Under **Parameter,** give a **name and password** to your database.
* Now give the root password for MySQL in **DBRootPassword** and username for MySQL database access in **DBUser.**
* Under **Instance Type** select the type of EC2 Web Server you want and give a **KeyName**
* Specify the IP address range for connecting to EC2 instances under **SSHLocation** and if not then leave it as default.
* Under **Configure stack options** configure it accordingly, review it and then create it.

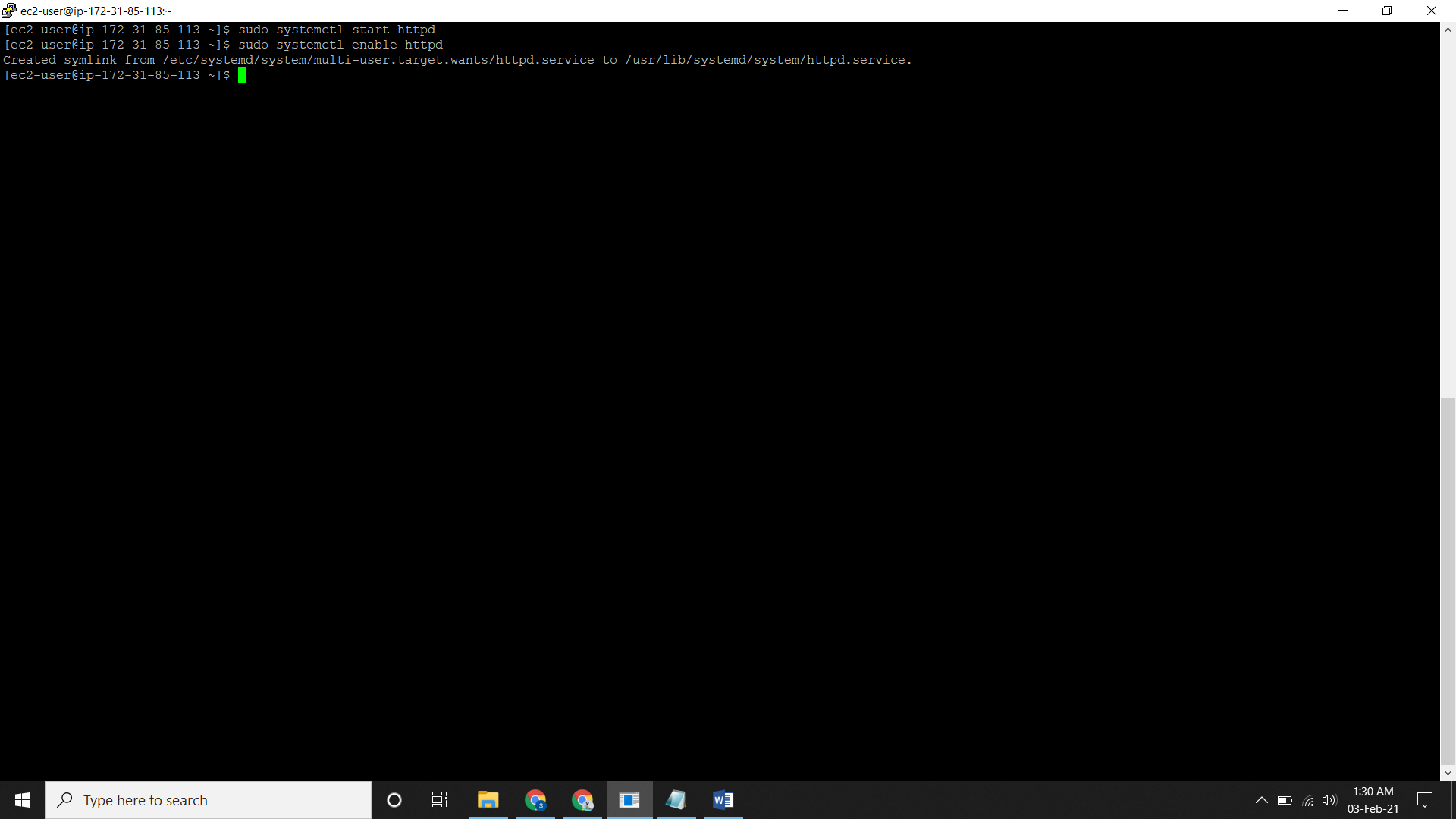


**Installing Apache**

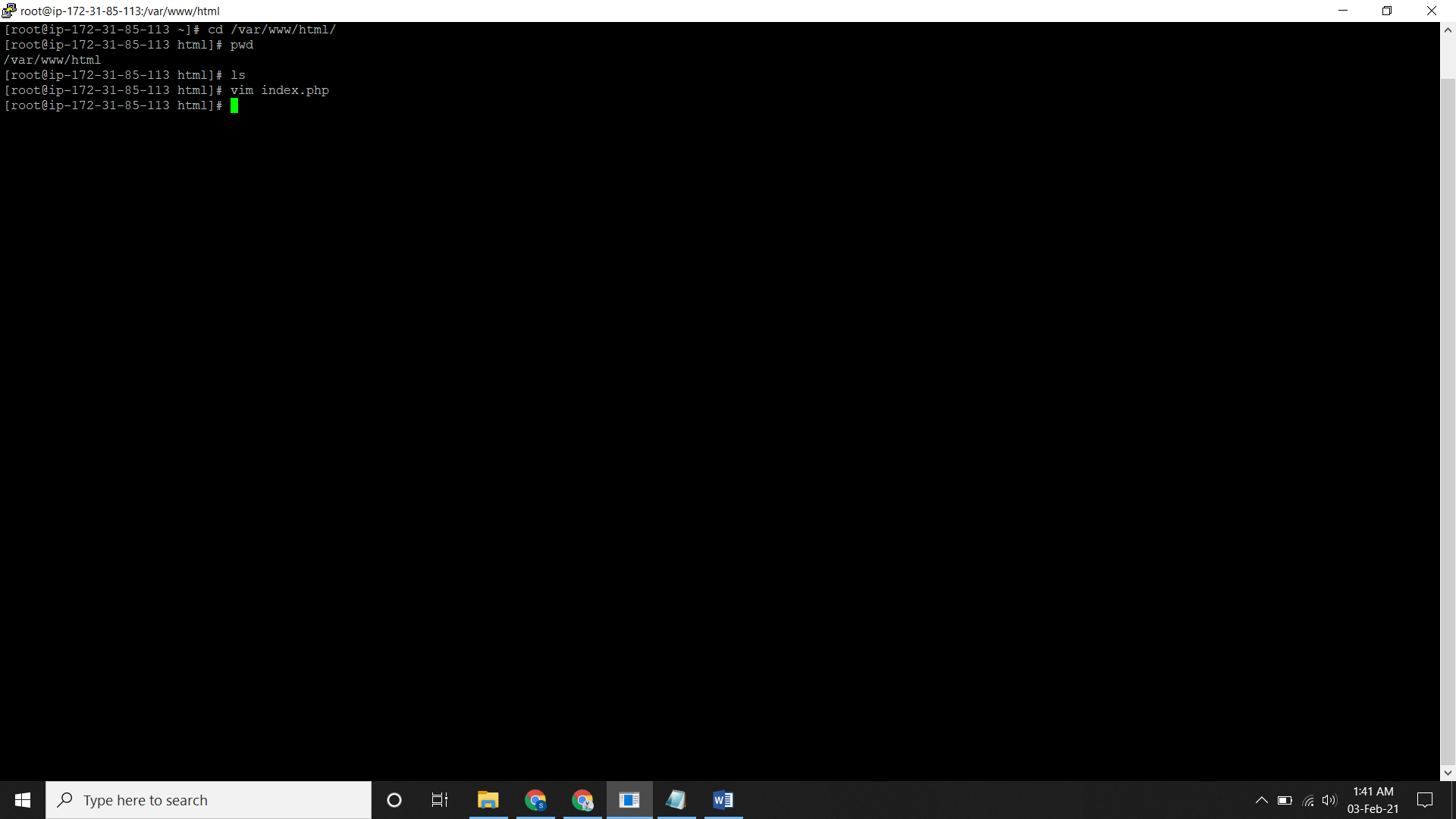
* Installing apache/httpd.



* Starting and enabling apache.

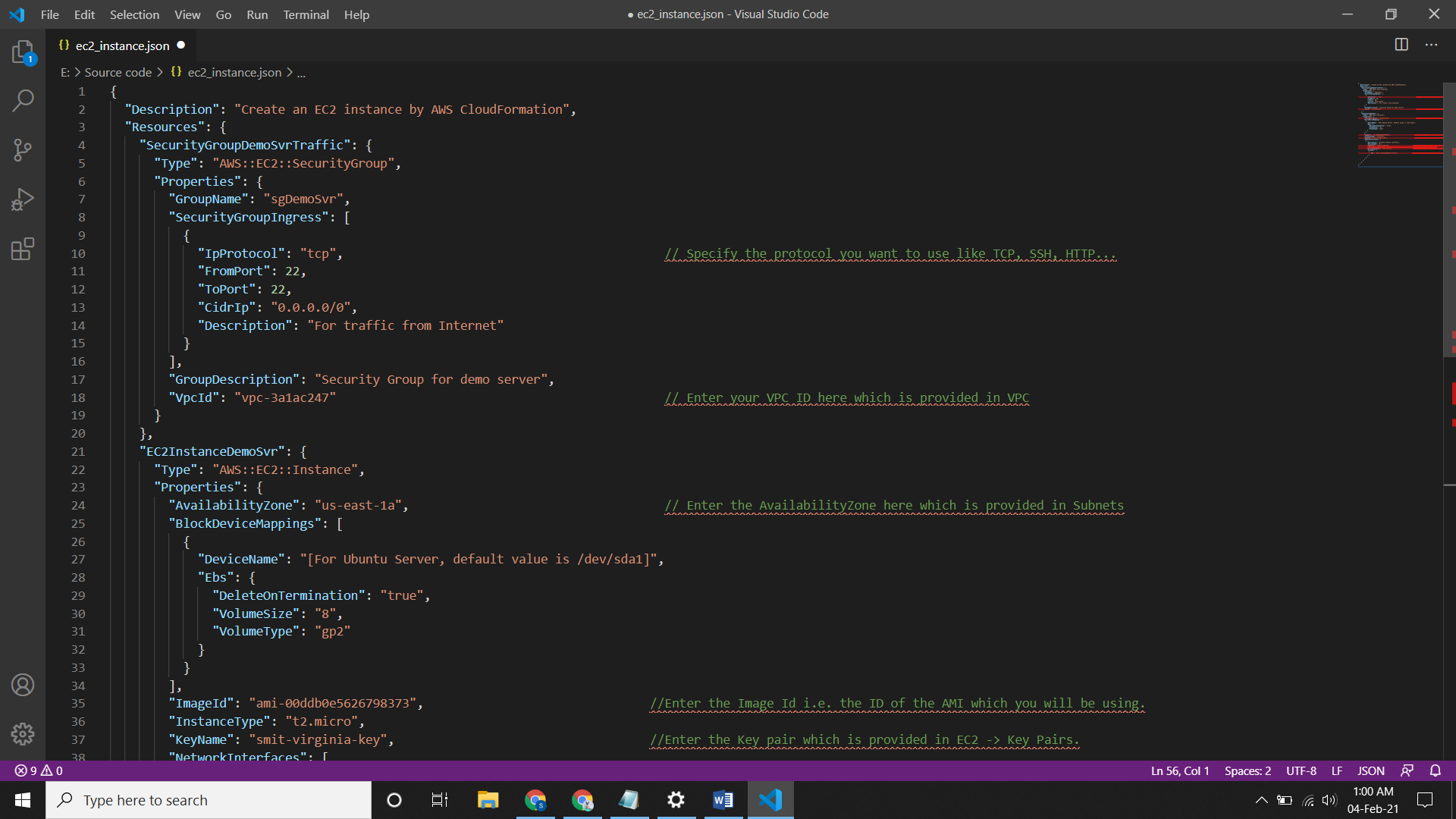


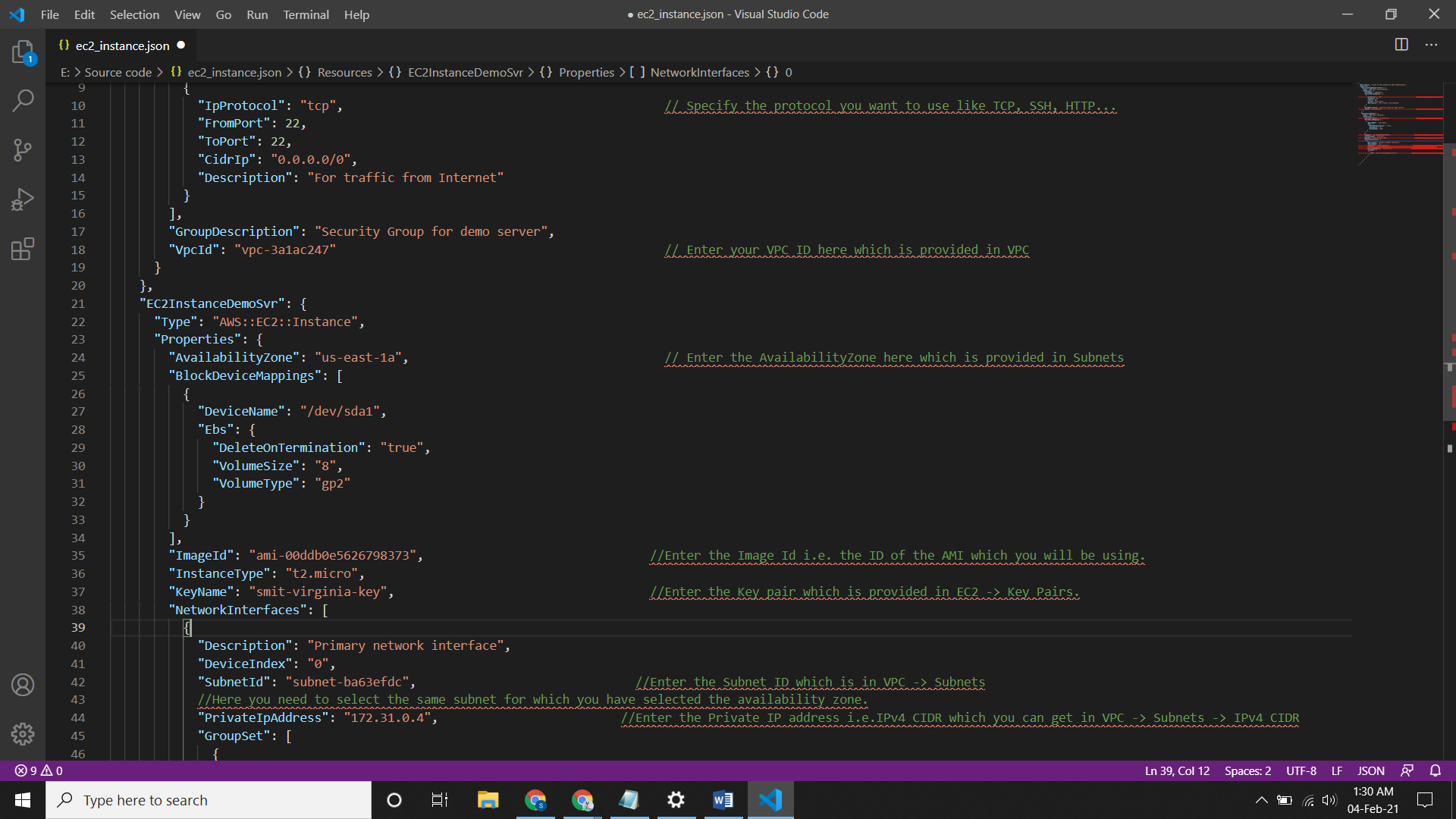
* Creating a sample file.



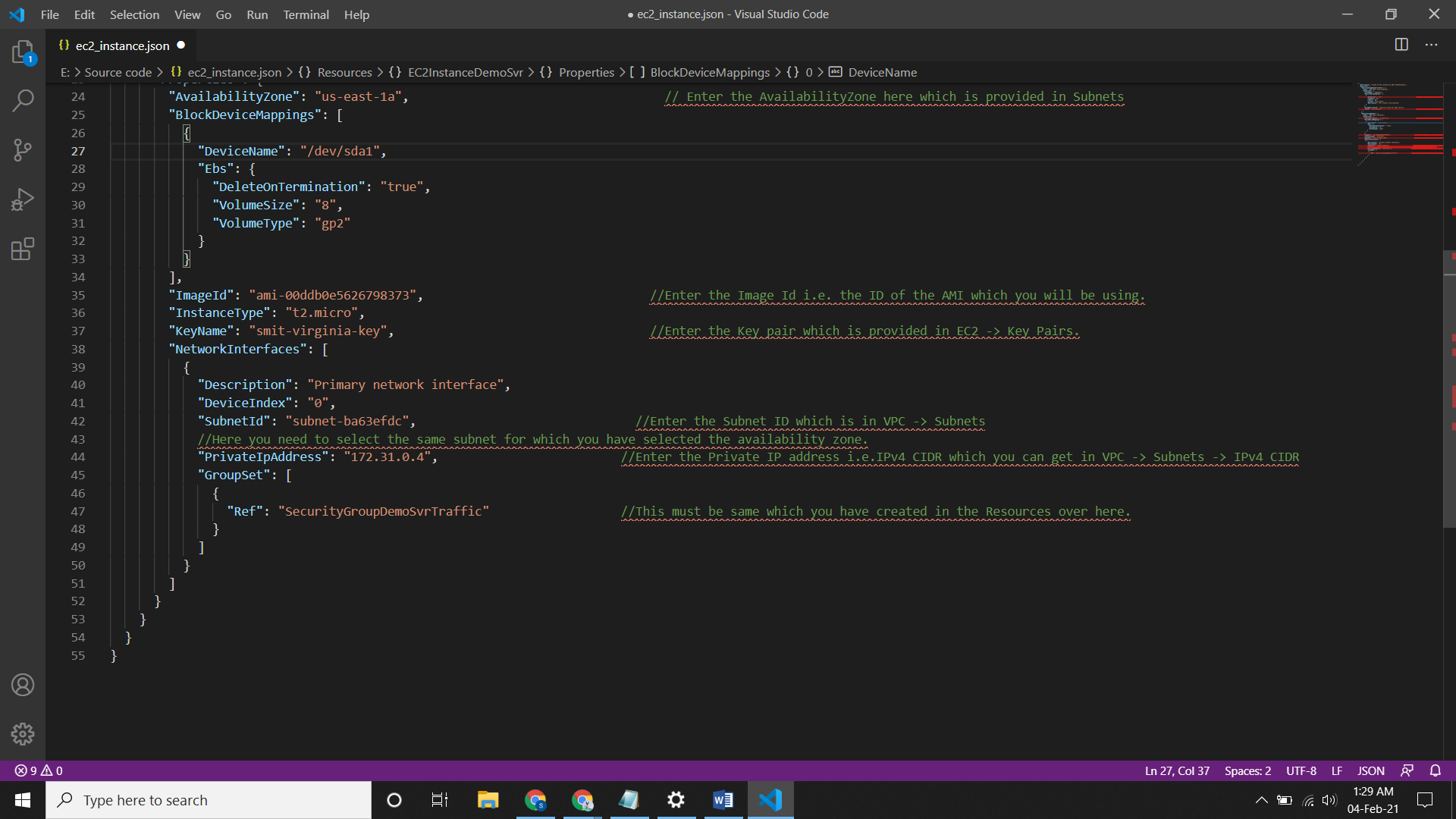
**Craeting EC2 Instance using AWS CloudFormation**

* Go to CloudFormation, click on **create a stack.**
* Under **Specify a Template,** click on **Upload a template file** and now upload the file.





Here the **DeviceName** will be “**/dev/sda1”**



Here Private IP address will always end with 4 because 0-3 are reserved by AWS and we can’t use it.

* Click on **next** and give a name to the stack.
* Now leave all the options as default and create the stack.
* It will take around 2-3 minutes to start.
* Now check the EC2 dashboard you will see a instance created.
* The availability zones, security groups and all the others things mentioned in the code will be same.

