**Spring boot API deployment via pipeline in ECS**

**Local setup**

* Created spring boot book project from STS IDE or spring initializer <https://start.spring.io/>
* Added web, data jpa , h2, open api ui dependency
* Adde get books, get book by id, update book and delete book API and configured h2 in file database
* Run spring boot app and test in local

**Git and GitHub repository**

**Add local repo to github -** [**https://github.com/dnayenshwar-kale/books**](https://github.com/dnayenshwar-kale/books)

|  |
| --- |
| git init  git add --no-warn-embedded-repo .  git commit -m "first commit"  git branch -M master  git remote add origin https://github.com/dnayenshwar-kale/books.git  git push -u origin master |

**Run spring boot API and test**

|  |
| --- |
| java -jar target/books-0.0.1-SNAPSHOT.jar or mvn spring-boot:run |

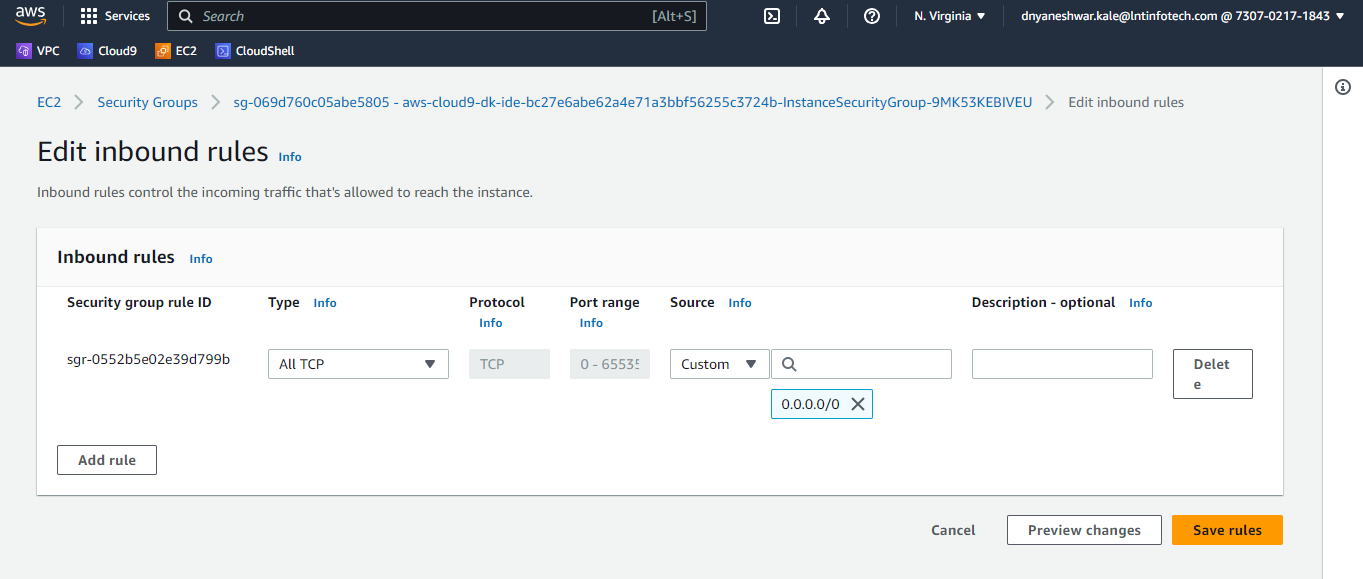
test using swagger ui <http://localhost:8080/swagger-ui/index.html>

**Run app on Cloud9 IDE**

Cloud9 setup

Create cloud9 IDE

Go to EC2 and edit inbound rule for SG



|  |
| --- |
| git clone <https://github.com/dnayenshwar-kale/books.git>  sudo yum install maven  Updated pom.xml to maven compatibility fix with cloud9  mvn clean install  mvn spring-boot:run  #for Created new branch aws\_cloud9 <https://github.com/dnayenshwar-kale/books/tree/aws_cloud9>  git checkout -b aws\_cloud9  git add .  git commit -m "inital commit for aws cloud9 compatible"  git push origin aws\_cloud9 |

**Run spring boot API and test**

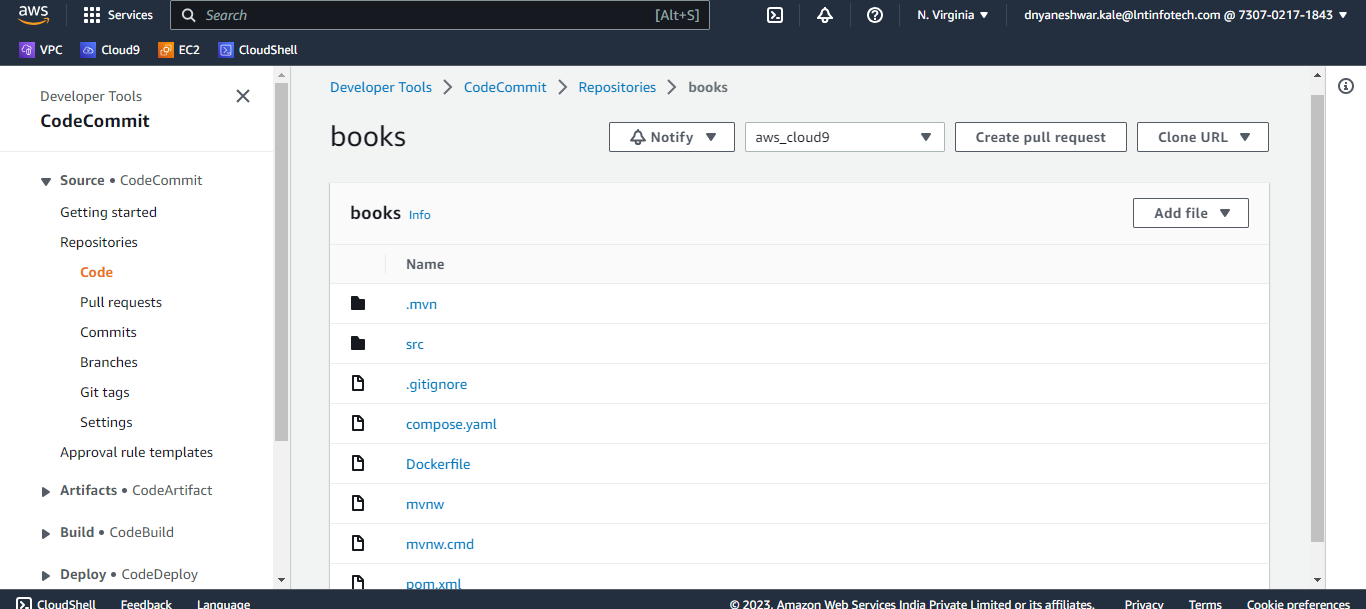
|  |
| --- |
| java -jar target/books-0.0.1-SNAPSHOT.jar or mvn spring-boot:run |

Test using swagger ui <https://bc27e6abe62a4e71a3bbf56255c3724b.vfs.cloud9.us-east-1.amazonaws.com/swagger-ui/index.html>

**Push to aws code commit**

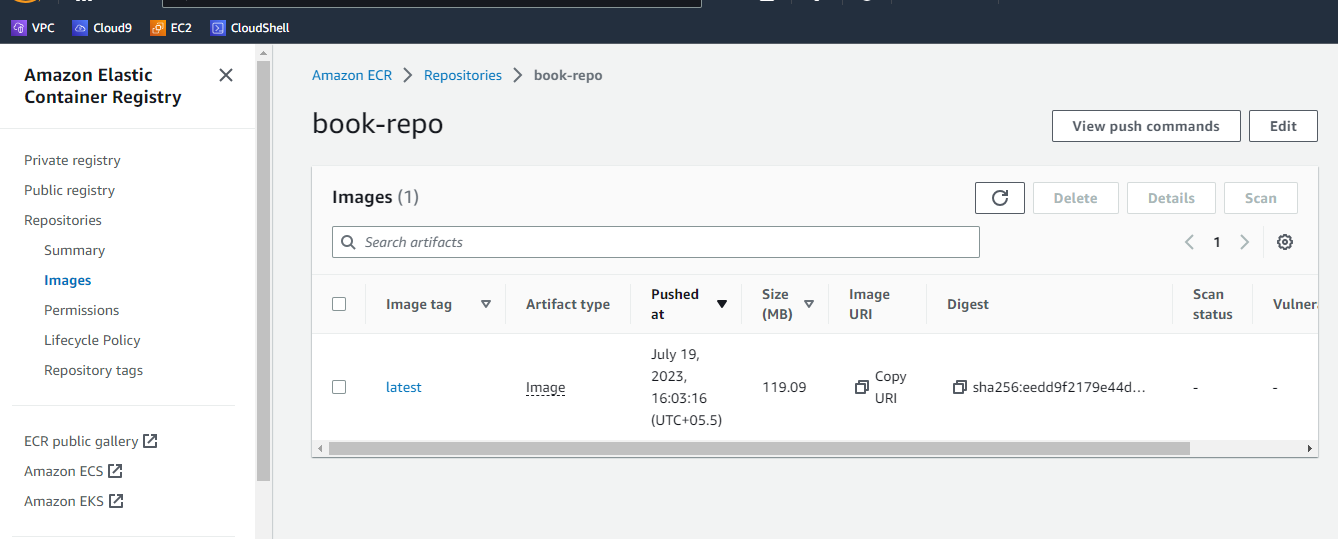
Create repo books git clone <https://git-codecommit.us-east-1.amazonaws.com/v1/repos/books>

|  |
| --- |
| git remote set-url origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/books git push -u origin aws\_cloud9 |



**Docker build, run and push to Aws ECR**

|  |
| --- |
| # Build the image with the tag spring-boot-app:latest  docker build -t books.jar:latest .  #Get list of images for local docker repo  docker image ls  # run a Docker image as a container  docker run books or sudo docker run books.jar -p 8080:80  # to run book.jar image as a container named book, in detached mode, with port 8080 mapped to port 80 on the host, with a volume mounted from /home/user/config on the host to /app/config on the container, and with a restart policy of always  docker run -d --name books -p 8080:80 -v /home/user/config:/app/config --restart=always books.jar  #find and kill running docker  docker ps  docker kill books  # Create an ECR repository named book-repo  aws ecr create-repository --repository-name book-repo --region us-east-1  # Get the login password and login to ECR  docker login -u AWS -p $(aws ecr get-login-password --region us-east-1) 730702171843.dkr.ecr.us-east-1.amazonaws.com  # Tag the image with the ECR repository URI  docker tag books.jar 730702171843.dkr.ecr.us-east-1.amazonaws.com/book-repo  # Push the image to ECR  docker push 730702171843.dkr.ecr.us-east-1.amazonaws.com/book-repo  #push to docker hub  docker push dpkalehub/books:latest |



**Push to aws code commit**

Clone repo from git

git clone <https://github.com/dnayenshwar-kale/books.git>

Cd . .

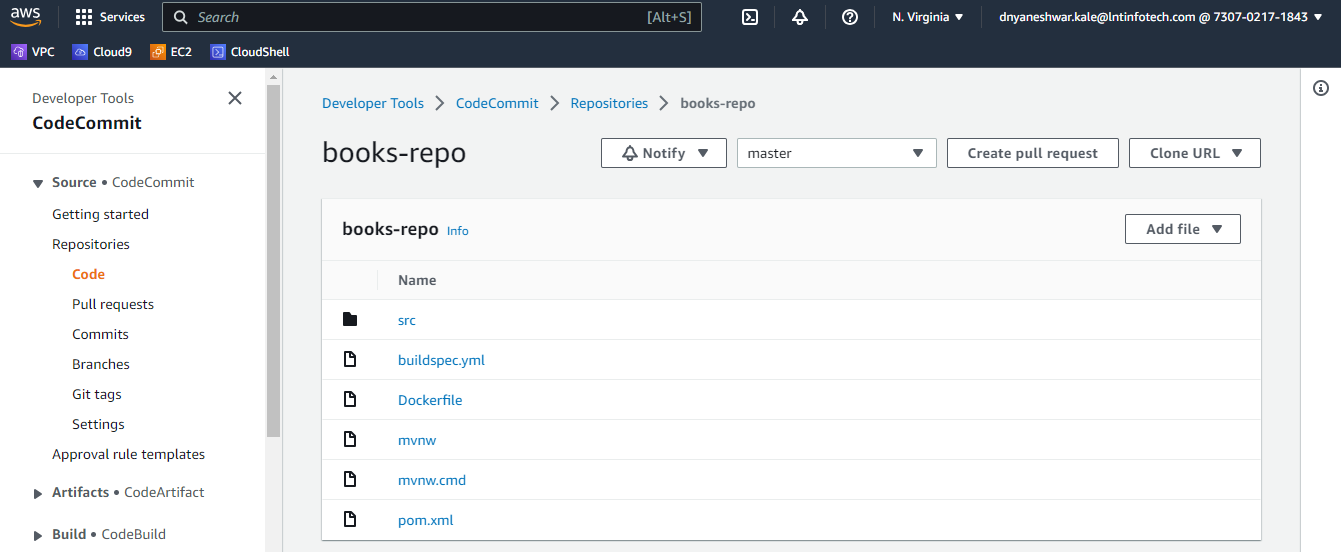
Create repo books and clone it

git clone <https://git-codecommit.us-east-1.amazonaws.com/v1/repos/books-repo>

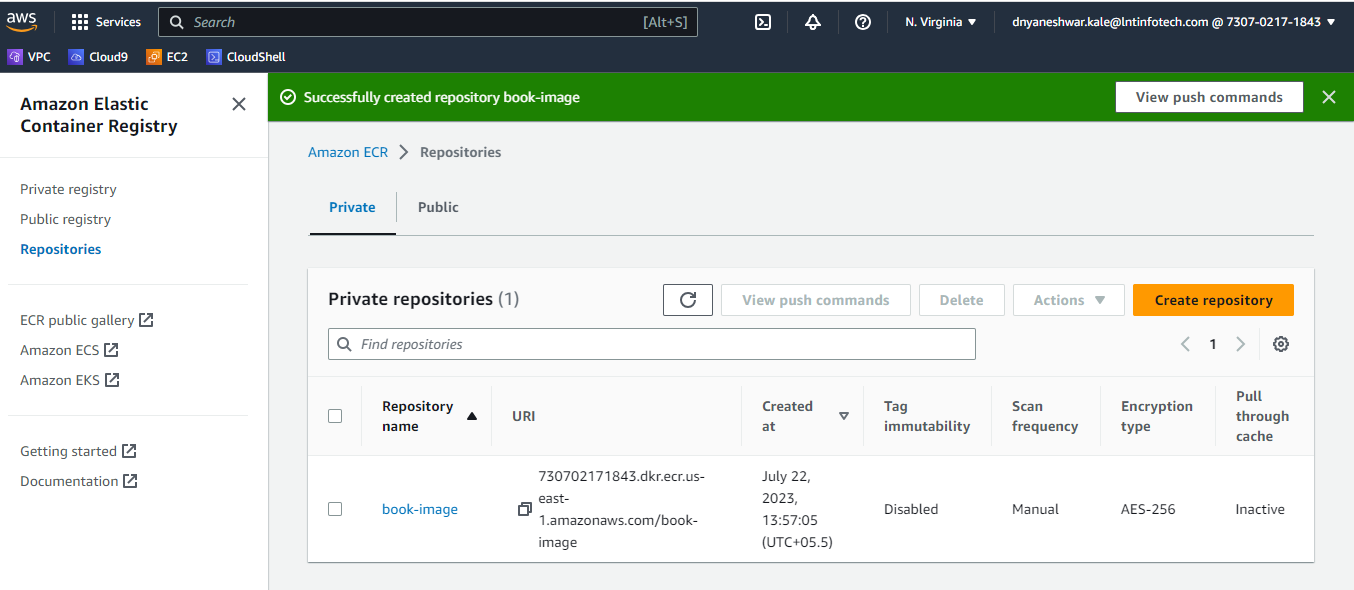
Cd books-repo

Git add .

git commit -m "init"  
git push -u origin master

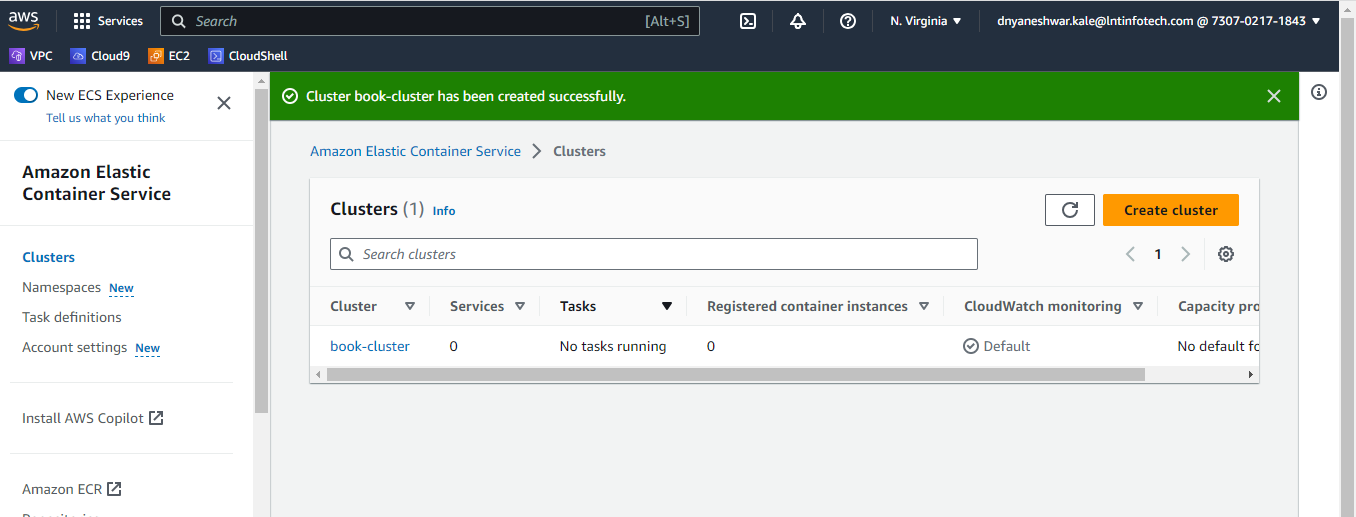


Create ECR repo

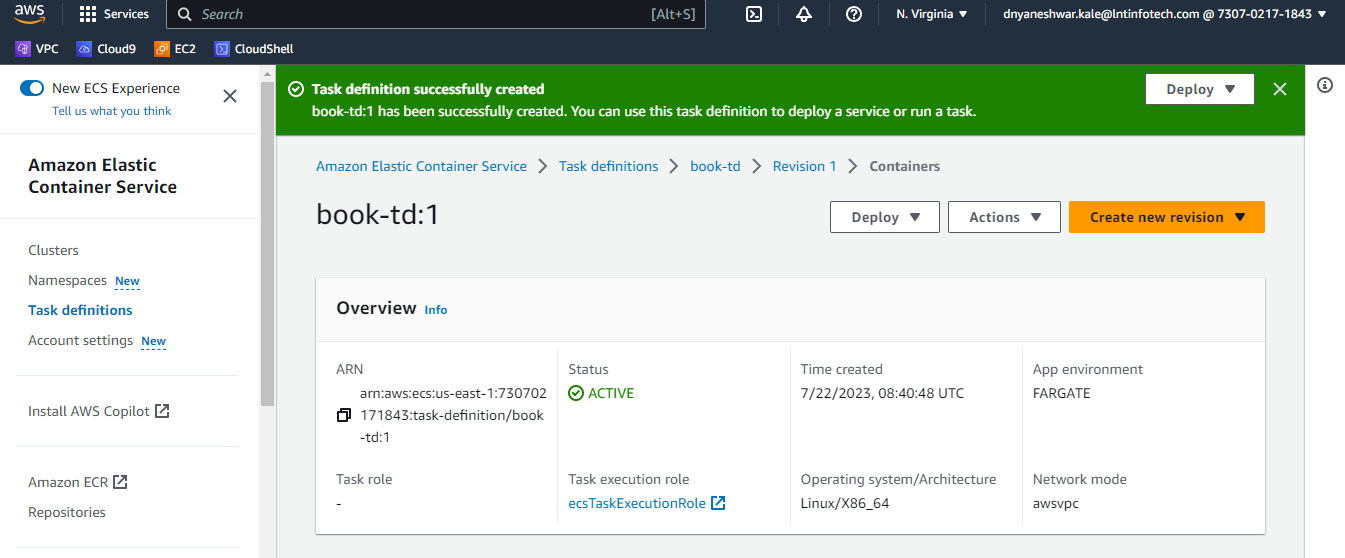


Prepare ECS env to deploy via pipeline

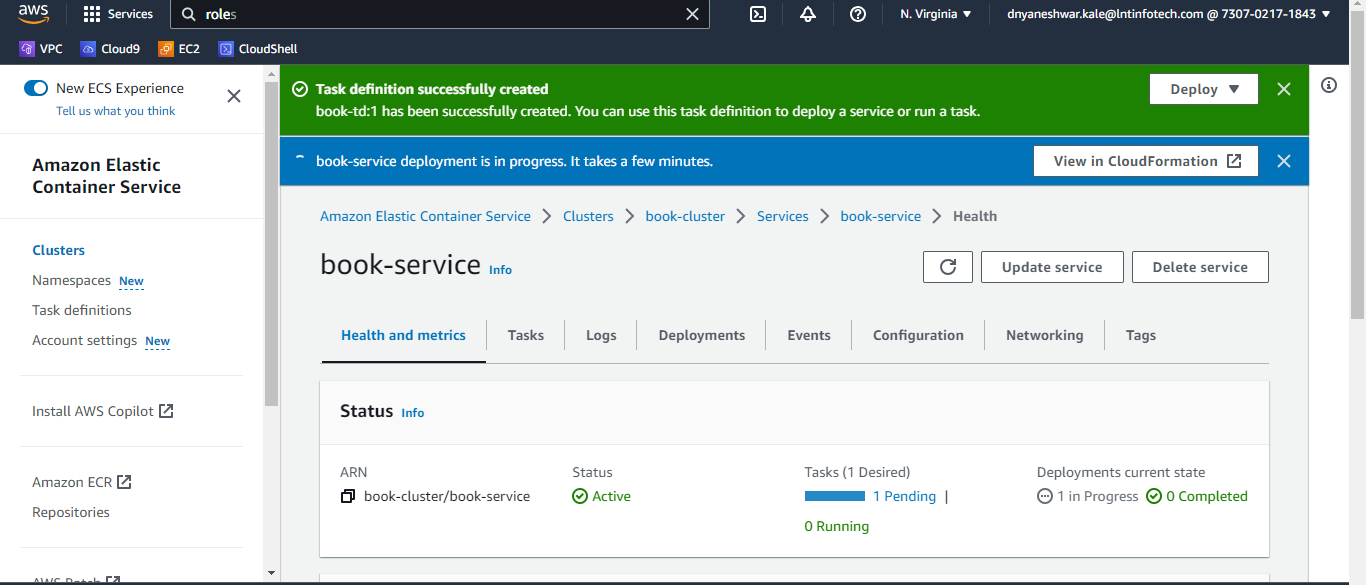
Create cluster



Create task definition



Create service



**Create build project**

**1 for Sonar scan integration**

<https://aws.amazon.com/blogs/devops/integrating-sonarcloud-with-aws-codepipeline-using-aws-codebuild/>

**Create secrete in aws secret manager,**

Sonarqube – secret name - prod/sonar

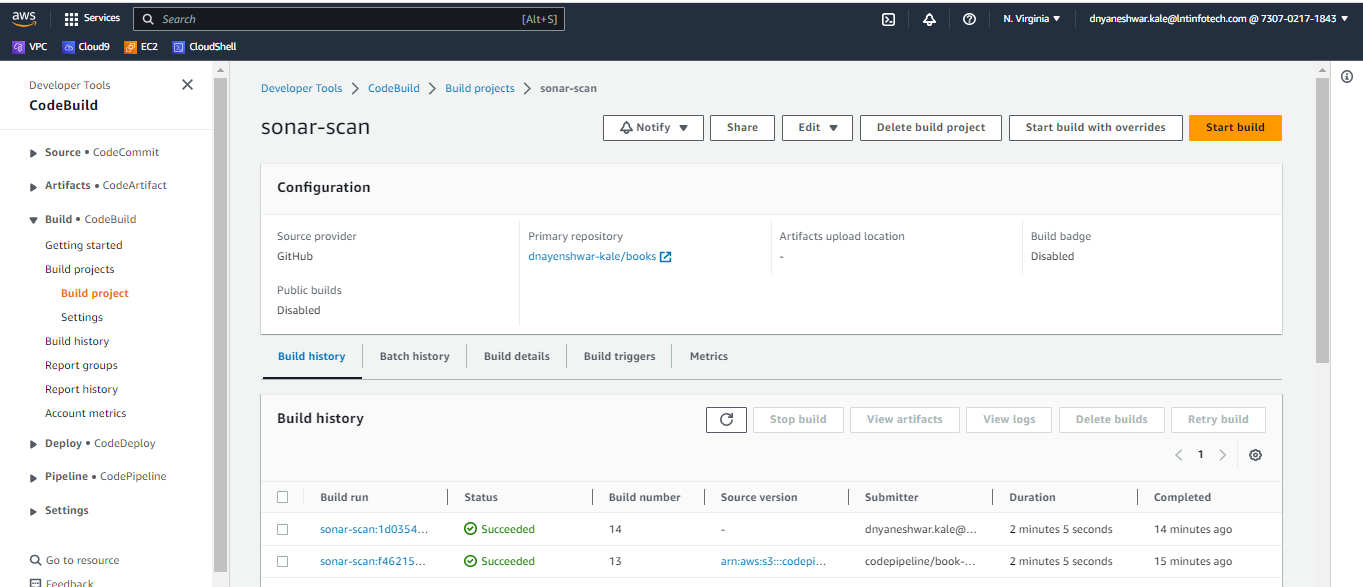
token a192c39382d9f21f8ace68235dea6ea07c0e9070

HOST [https://sonarcloud.io](https://sonarcloud.io/)

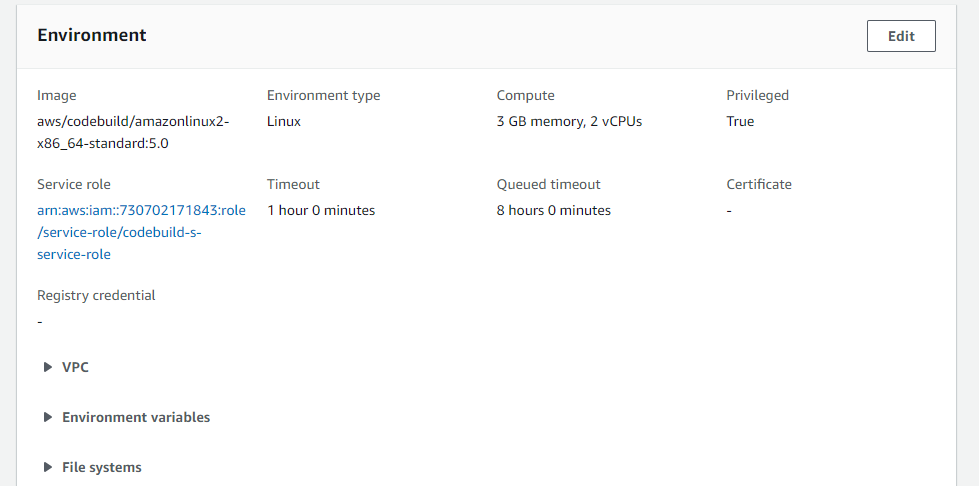
dnayenshwar-kale

dnayenshwar-kale\_books

Build project With enable Privileged



Go to build project -> sonar scan -> build details, go down in Environment check service role

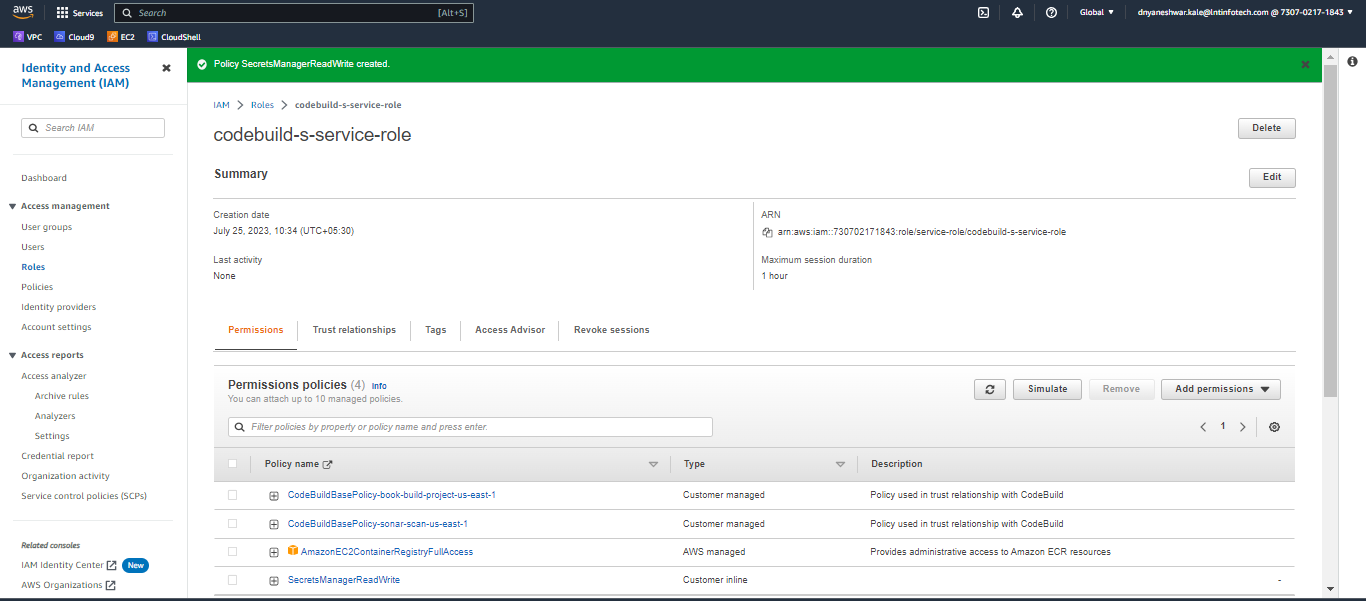


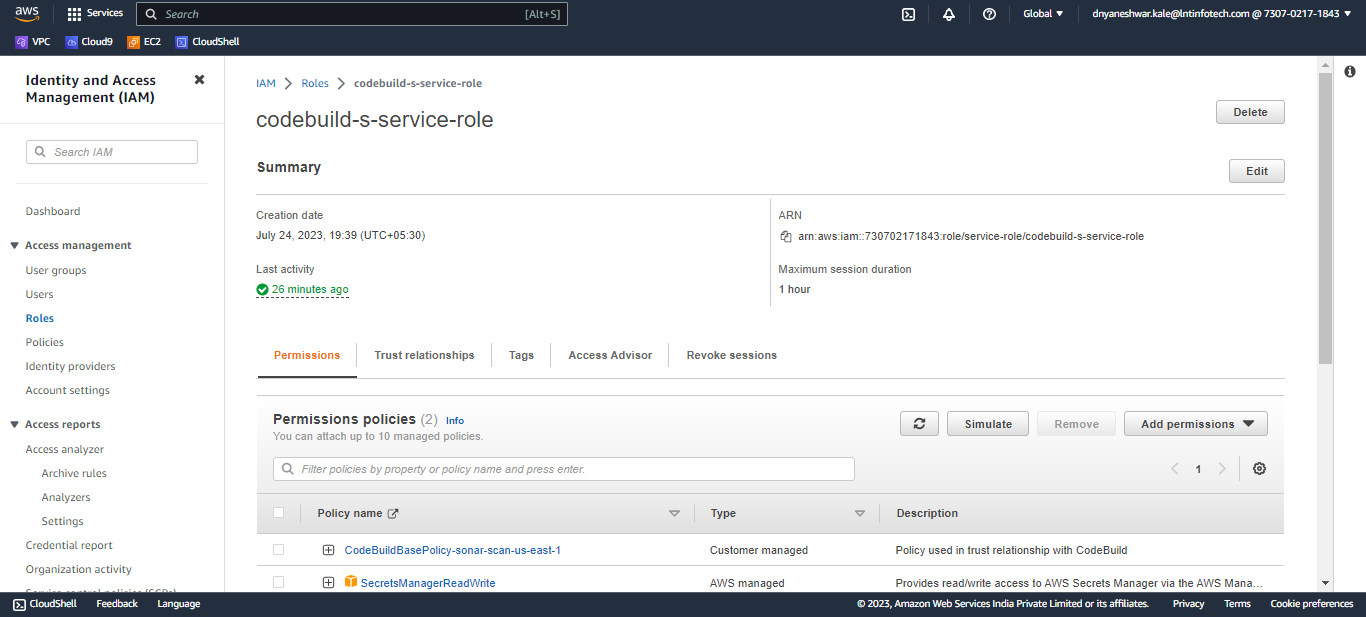
Then click on role go to permission -> add permission -> Attach policy to codebuild-n-service-role

Search “SecretsManagerReadWrite”

if not here Create new inline policy with name **SecretsManagerReadWrite**

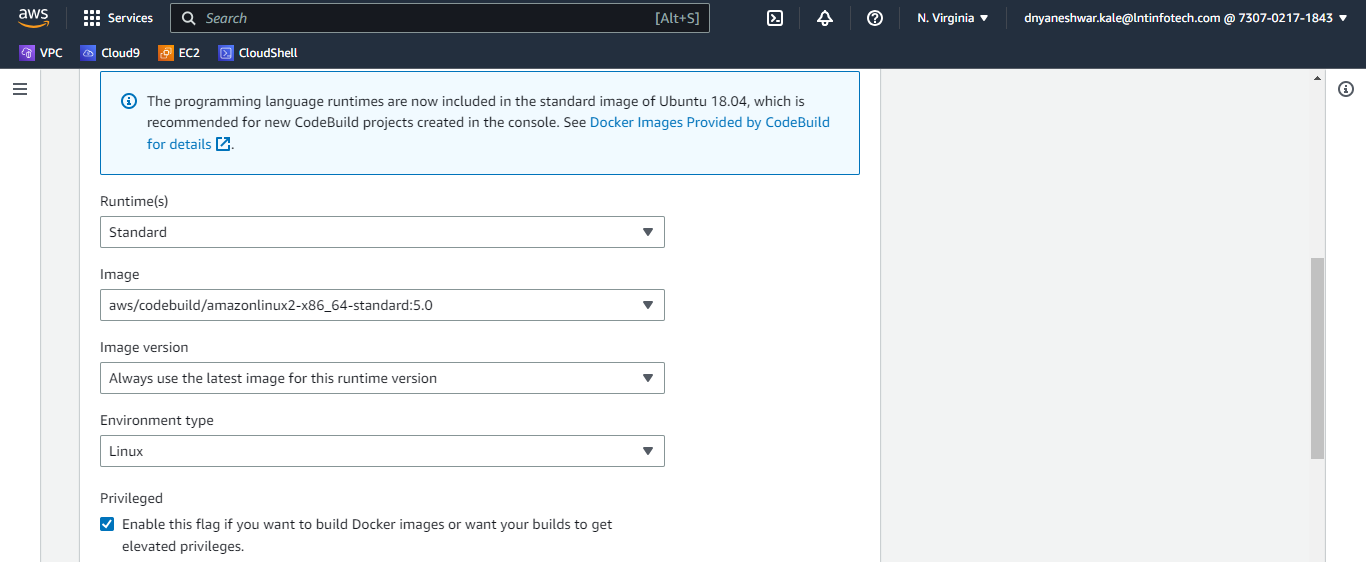
|  |
| --- |
| {  "Version": "2012-10-17",  "Statement": [  {  "Effect": "Allow",  "Action": "secretsmanager:GetSecretValue",  "Resource": "arn:aws:secretsmanager:us-east-1:730702171843:secret:prod/sonar"  }  ]  } |

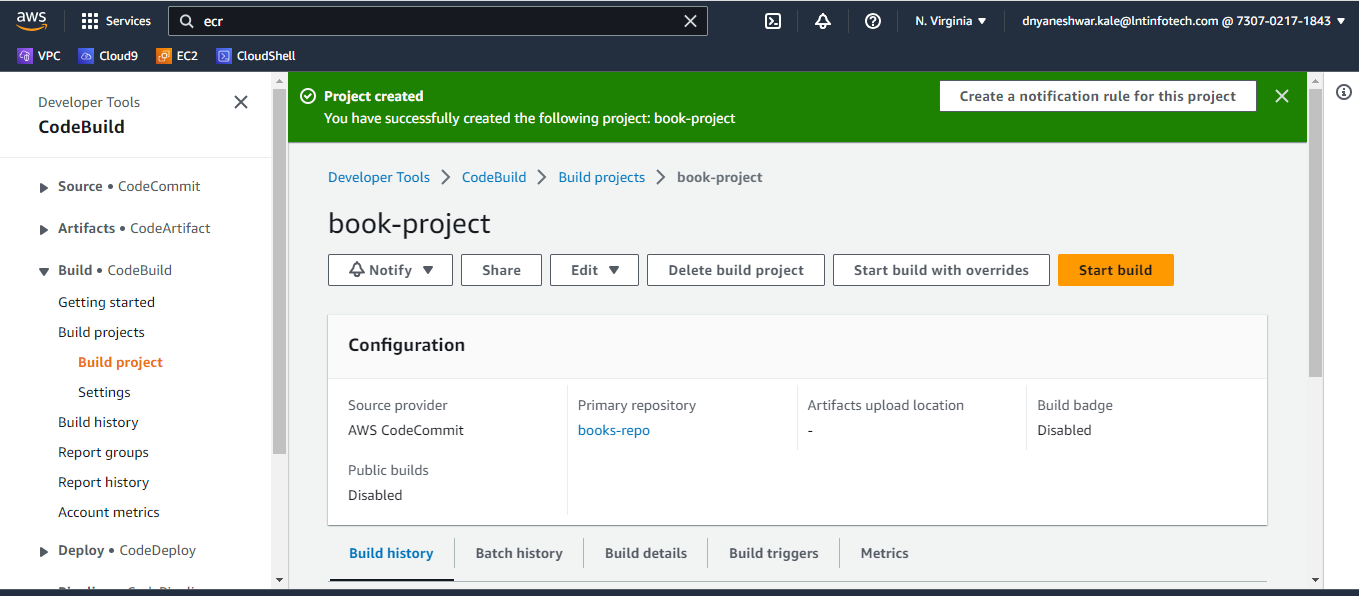




**2 for build docker image and push to ECR**

With enable Privileged





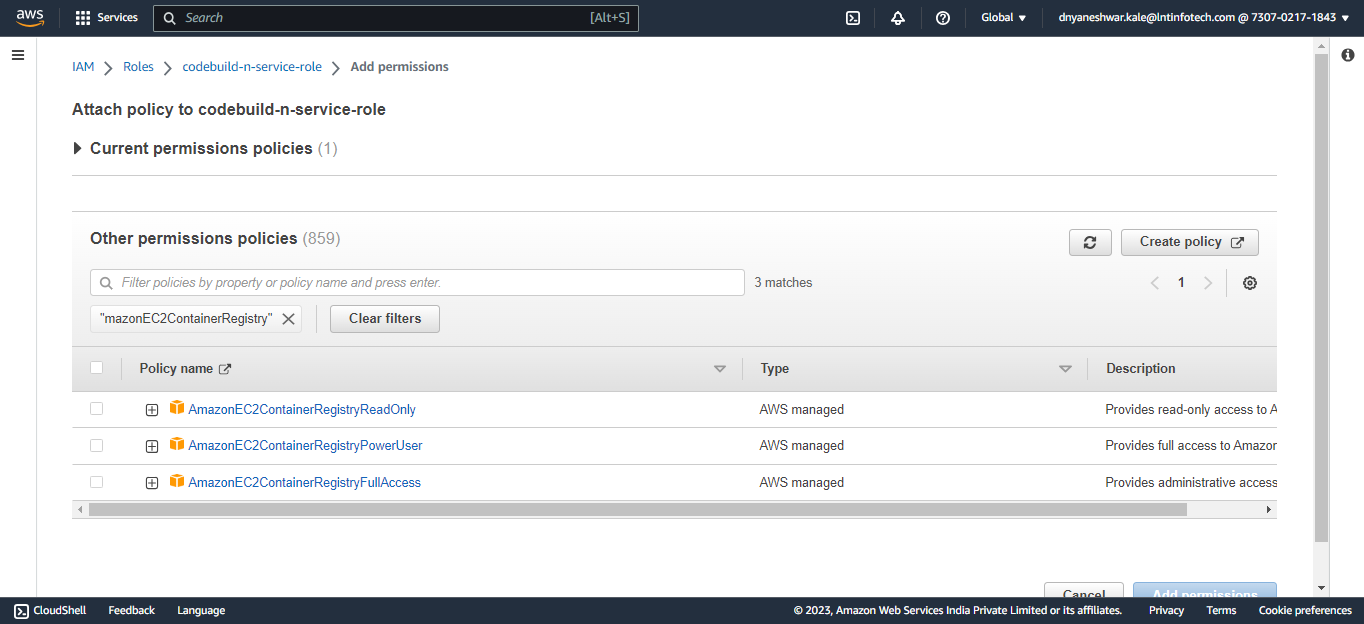
Go to build project -> book-project-> build details, go down in Environment check service role



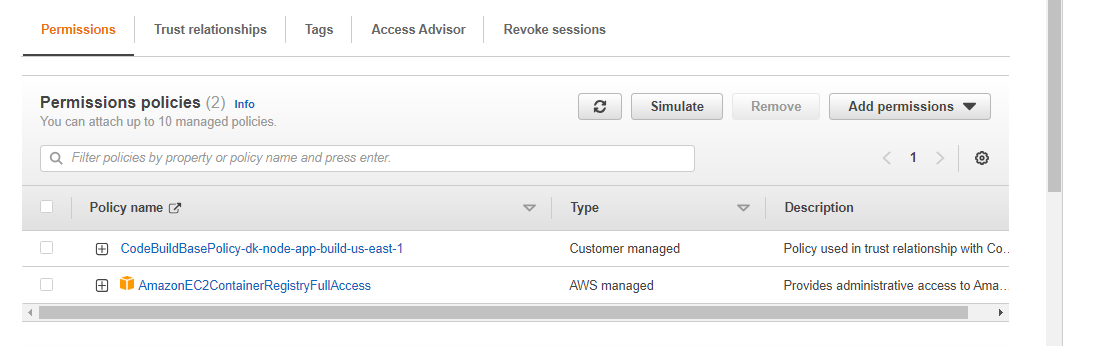
Click on service role

Then go to permission -> add permission -> Attach policy to codebuild-n-service-role

Search “AmazonEC2ContainerRegistry”

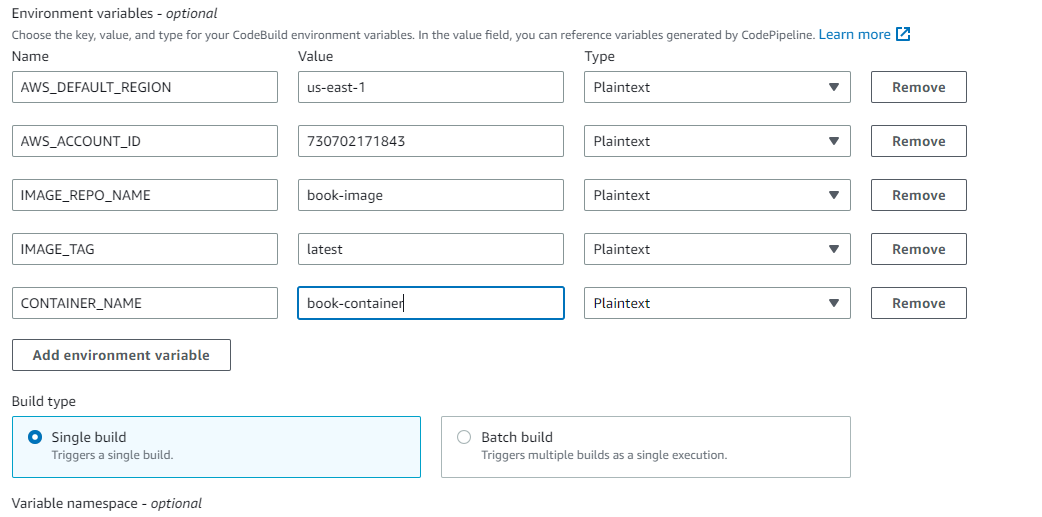


[AmazonEC2ContainerRegistryFullAccess](https://us-east-1.console.aws.amazon.com/iamv2/home#/policies/details/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAmazonEC2ContainerRegistryFullAccess)

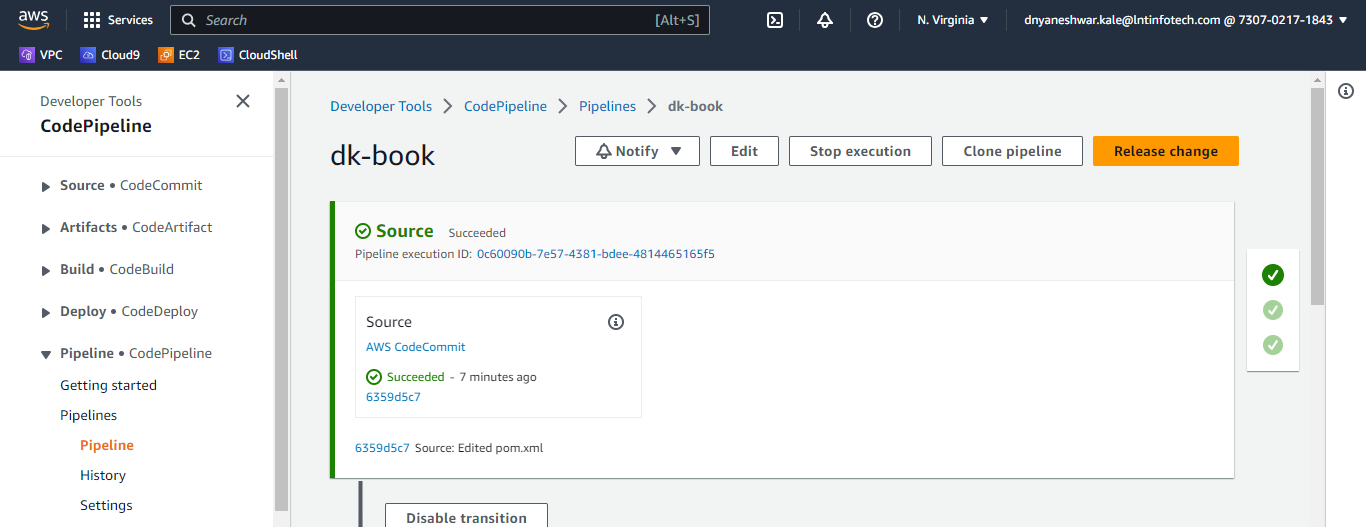


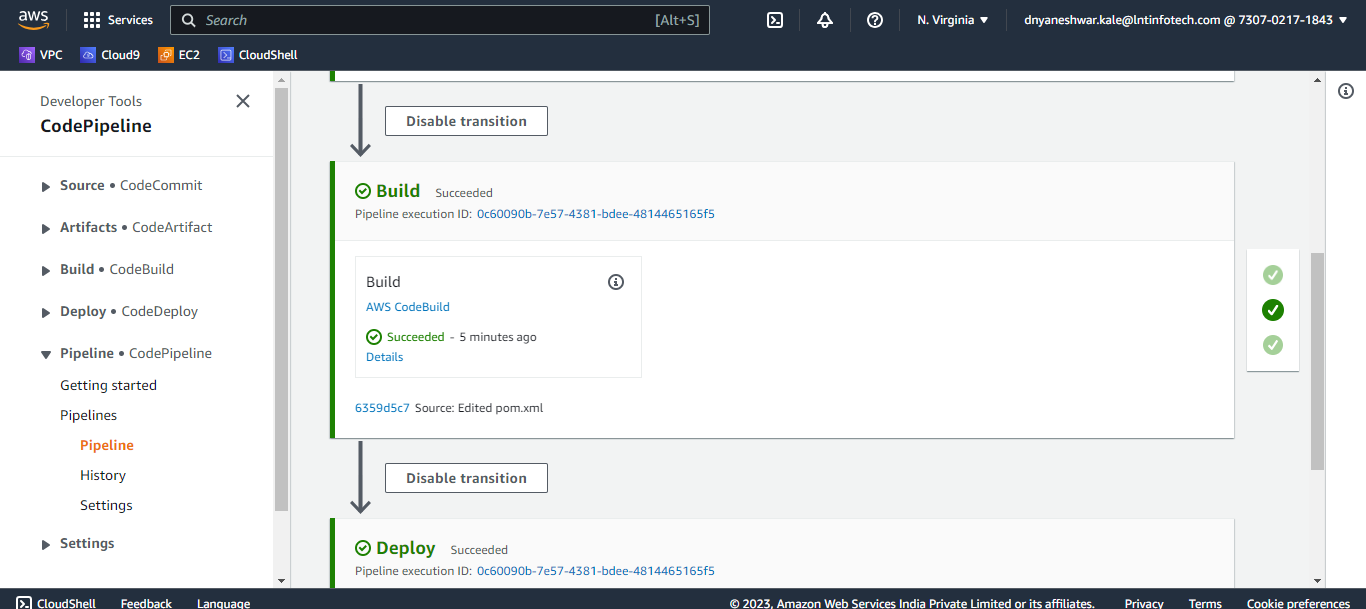
**Create pipeline**

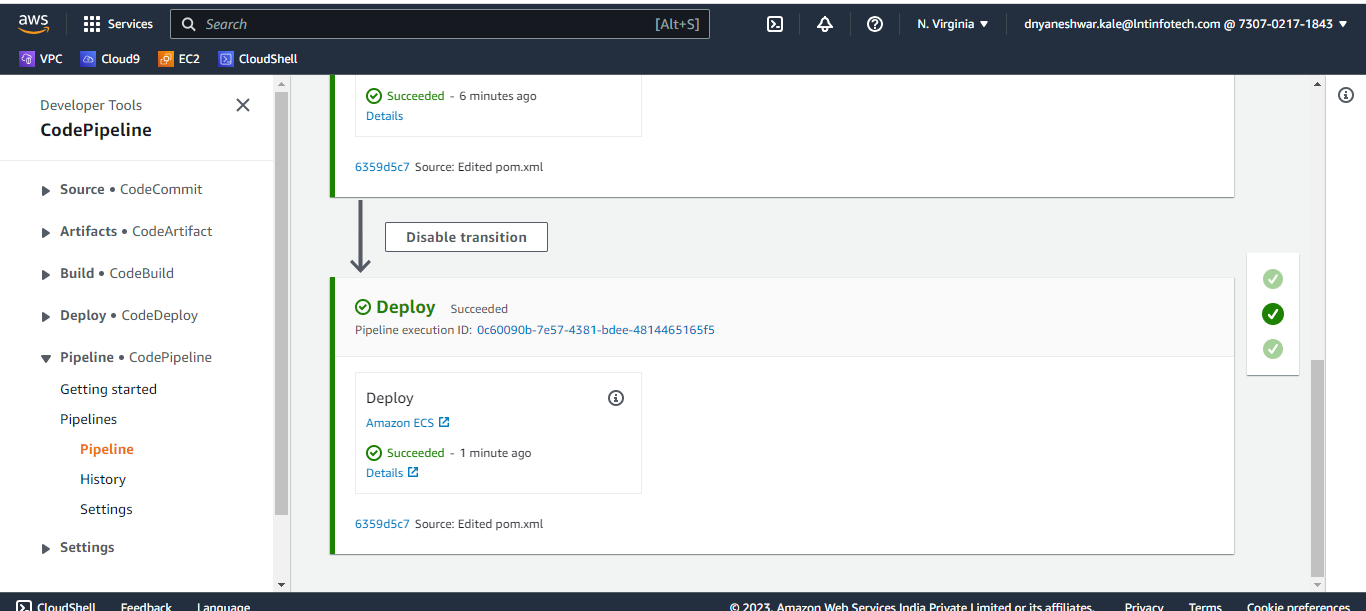
Add Environment variables



Validate build







If getting error in logs as below ,

Running command docker build -t $REPOSITORY\_URI:$IMAGE\_TAG . DEPRECATED: The legacy builder is deprecated and will be removed in a future release. Install the buildx component to build images with BuildKit: <https://docs.docker.com/go/buildx/> Sending build context to Docker daemon 17.39MB Step 1/7 : FROM node:18 18: Pulling from library/node toomanyrequests: You have reached your pull rate limit. You may increase the limit by authenticating and upgrading: <https://www.docker.com/increase-rate-limit> [Container] 2023/07/18 11:00:08 Command did not exit successfully docker build -t $REPOSITORY\_URI:$IMAGE\_TAG . exit status 1 [Container] 2023/07/18 11:00:08 Phase complete: BUILD State: FAILED

To resolve above error below is solution

increase the limit by authenticating and upgrading you aws ECR Repository

To increase your Amazon ECR pull rate limit, you need to authenticate and upgrade your account. You can find more information on how to do this here: <https://docs.aws.amazon.com/AmazonECR/latest/userguide/pull-rate-limits.html>

Here are the steps you need to follow:

1. Go to the Amazon ECR console at <https://console.aws.amazon.com/ecr/repositories>.
2. Choose the repository for which you want to increase the pull rate limit.
3. Choose the “Permissions” tab.
4. Choose “Edit policy JSON”.
5. Add the following statement to the policy:

Triger the build again

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"AWS": "730702171843"

},

"Action": [

"ecr:GetDownloadUrlForLayer",

"ecr:BatchGetImage",

"ecr:BatchCheckLayerAvailability"

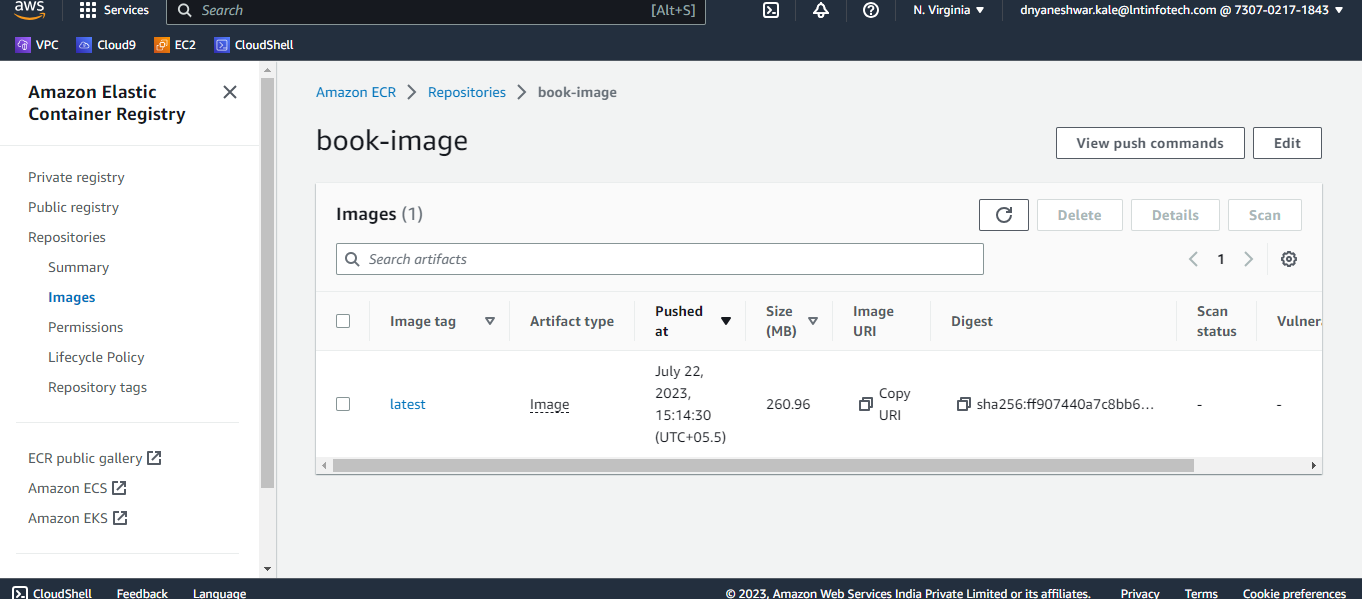
]

}

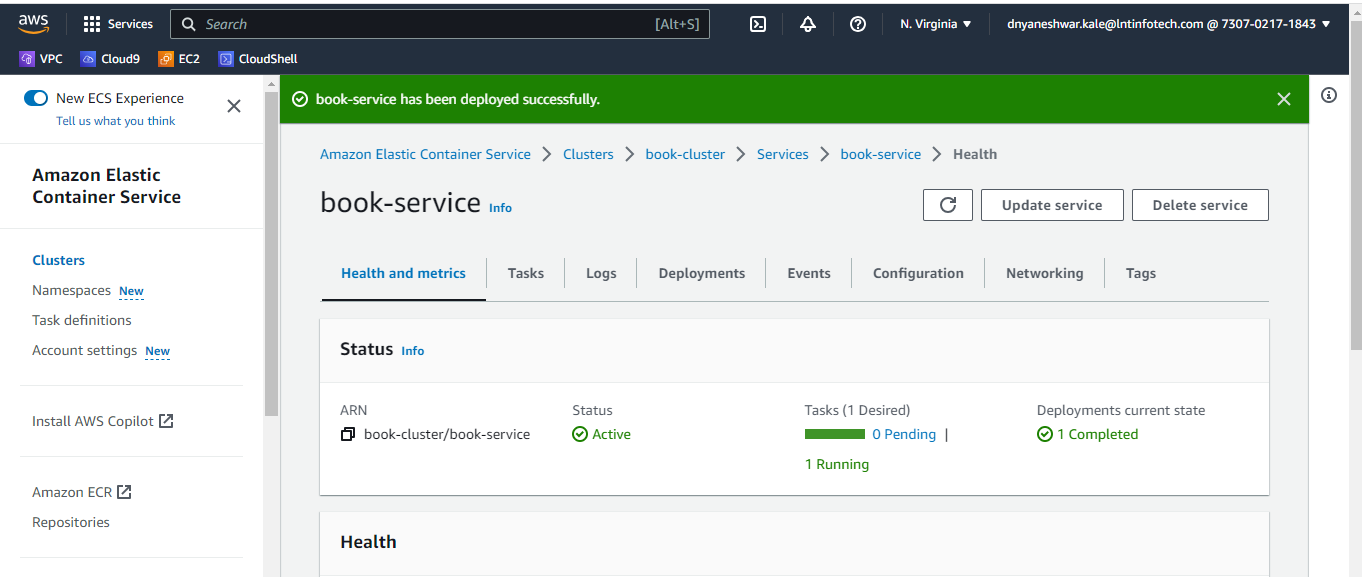
]

}

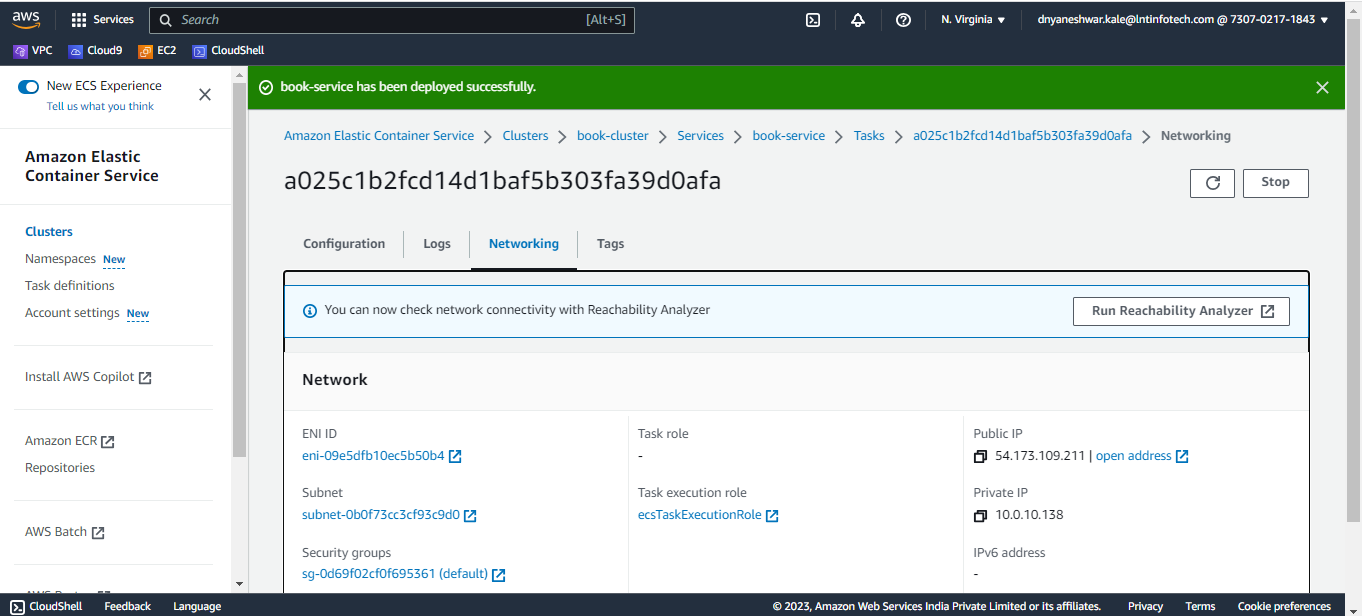
Verify image in ECR



Verify deployment and test in ECS

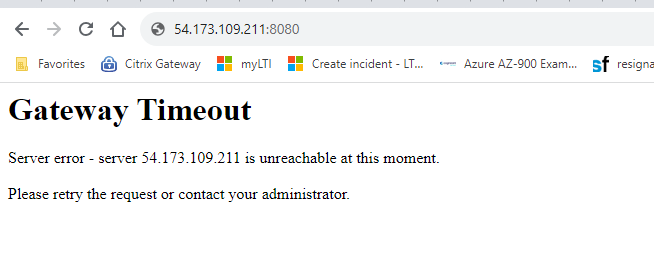


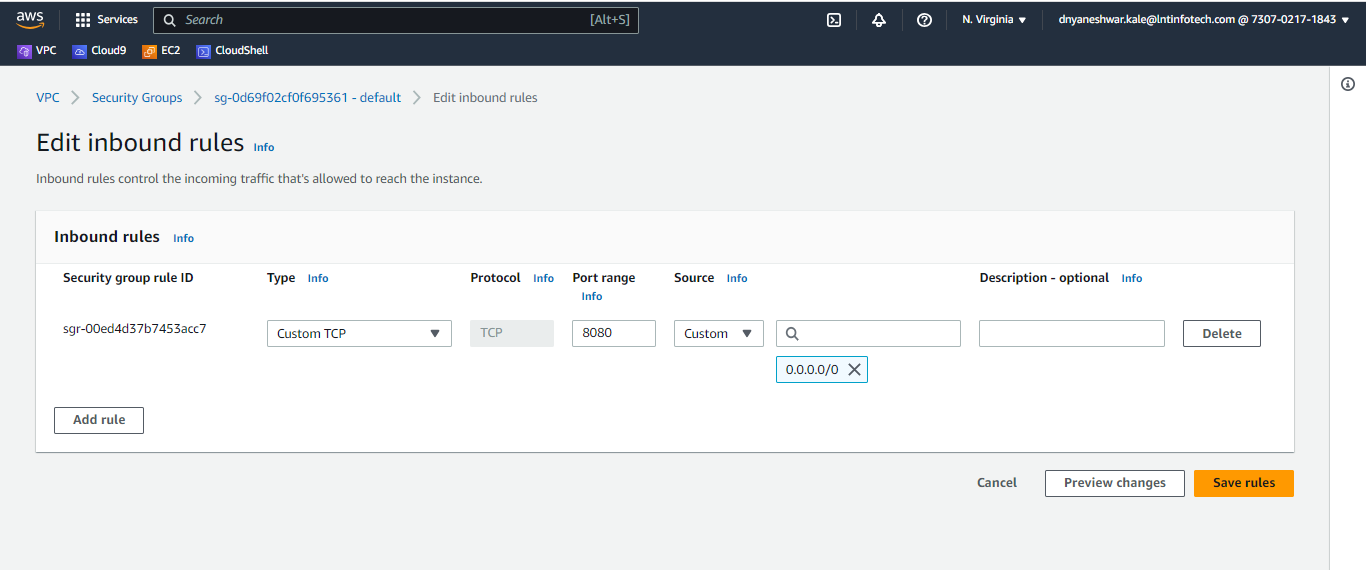
Go to service ->task and copy public ip and test api



<http://54.173.109.211:8080/swagger-ui/index.html>#

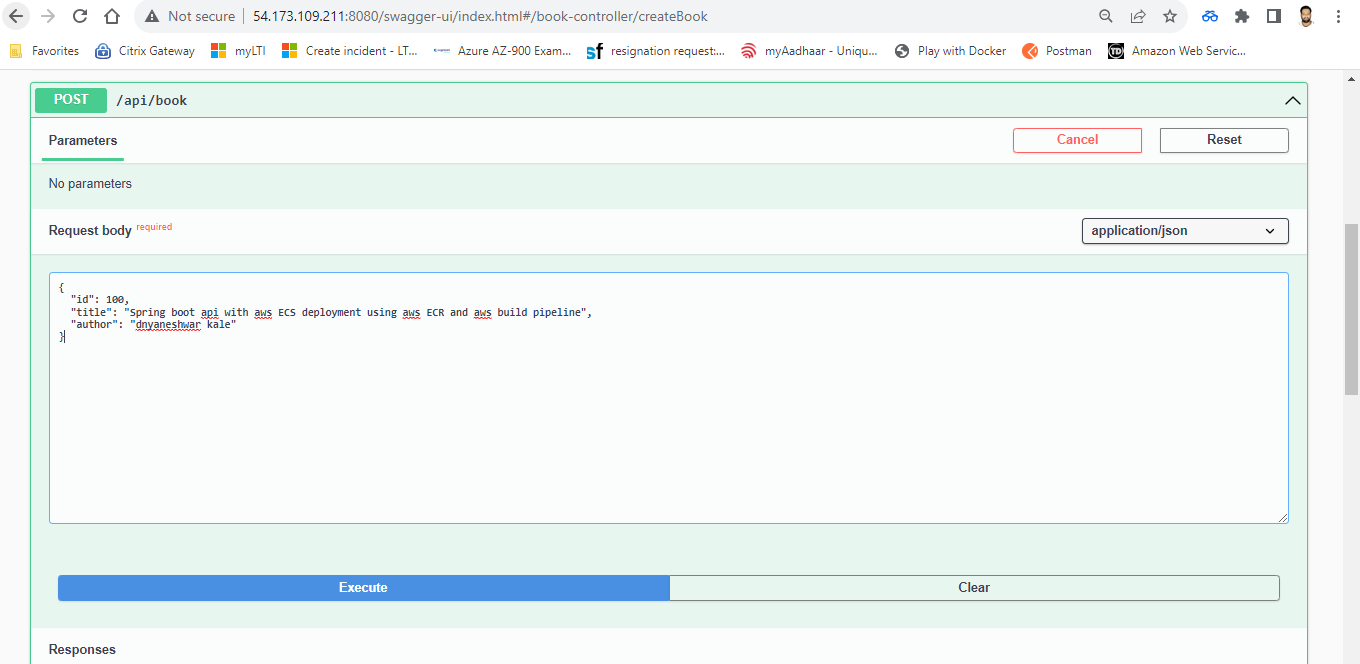
Note if you are not able to access then go to SG of that task and open inbound rule for port 8080

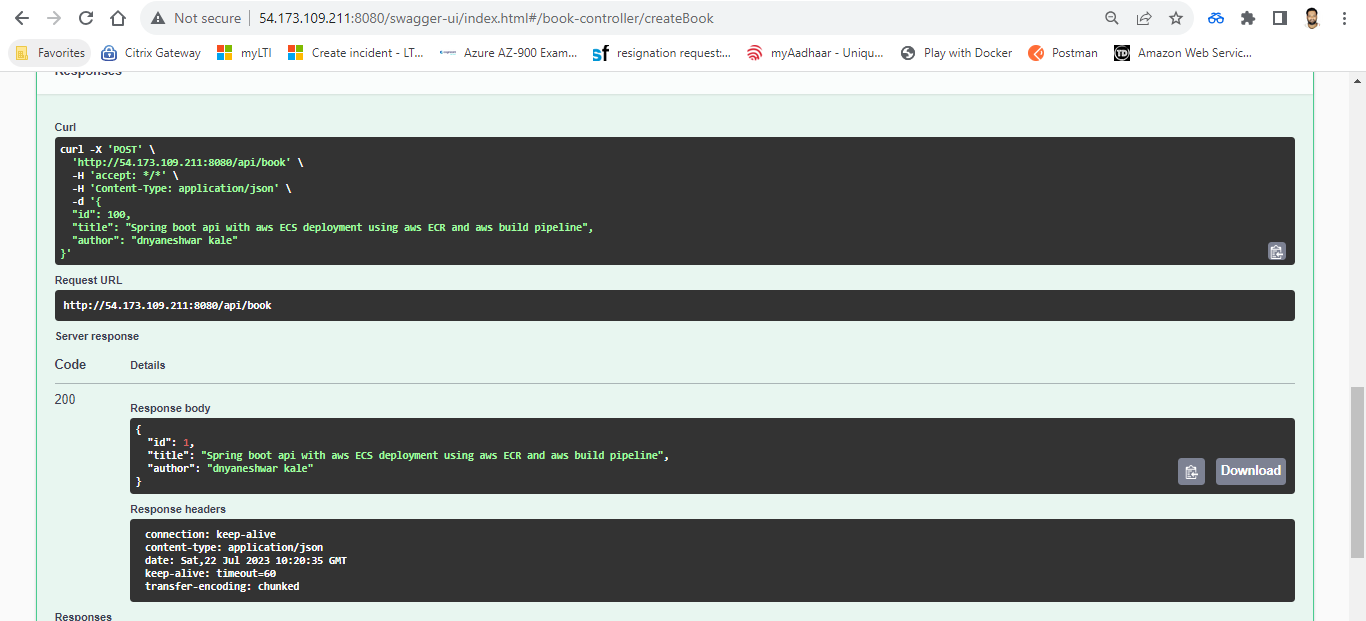




Test API

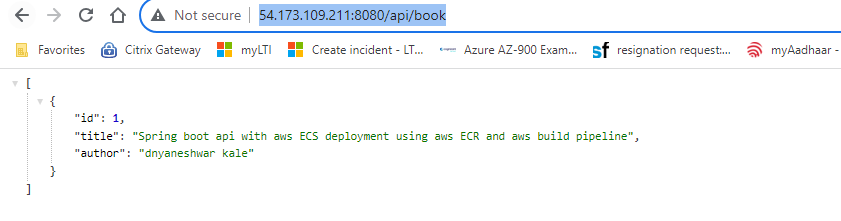
<http://54.173.109.211:8080/swagger-ui/index.html>#





Get all books

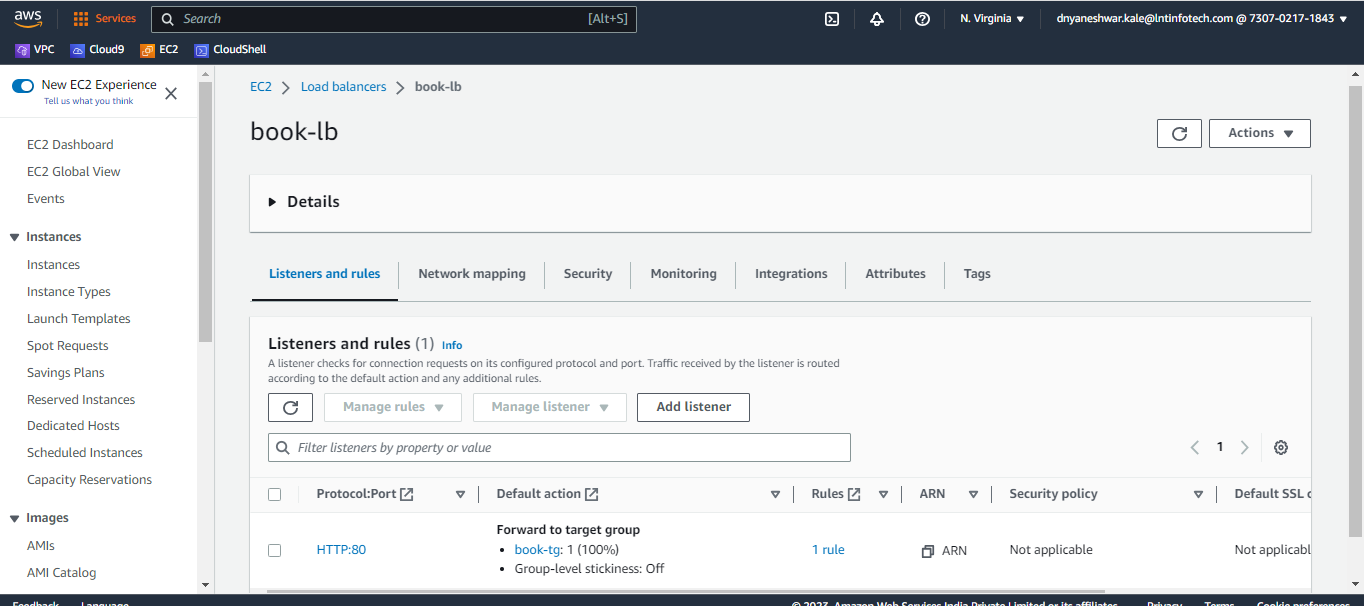
<http://54.173.109.211:8080/api/book>



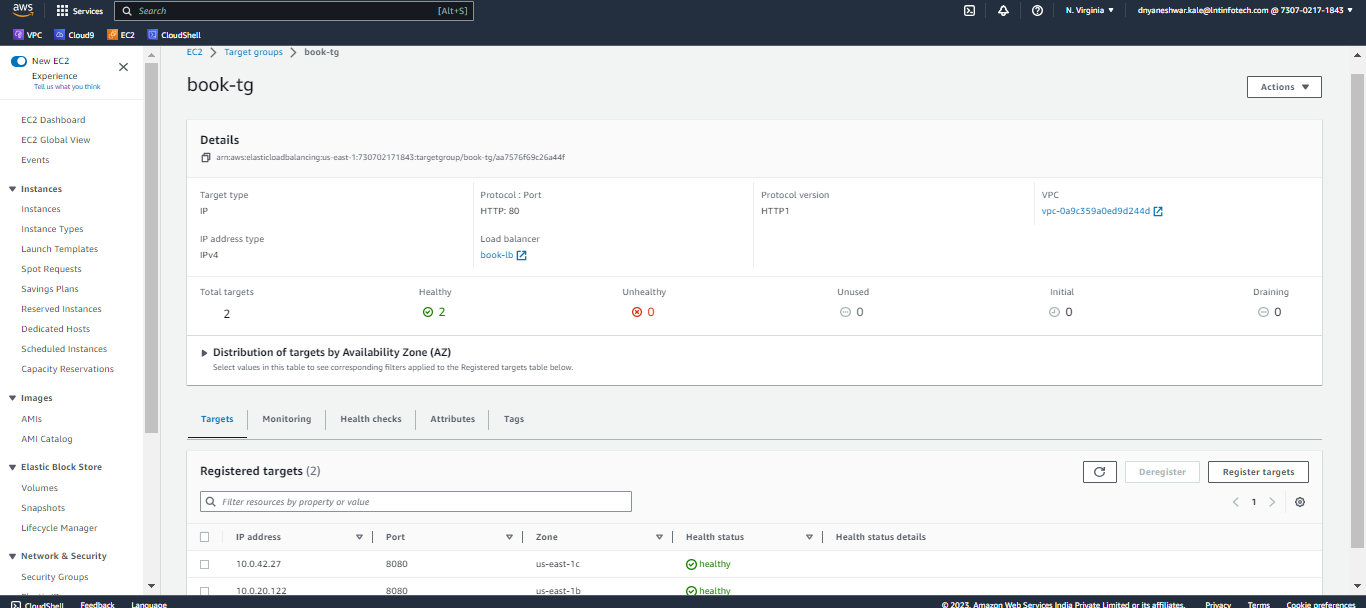
**Load balanacer**

**Accept request on 80 and TG on 8080 so open both inbound port on SG**

**Create service with application load balancer**



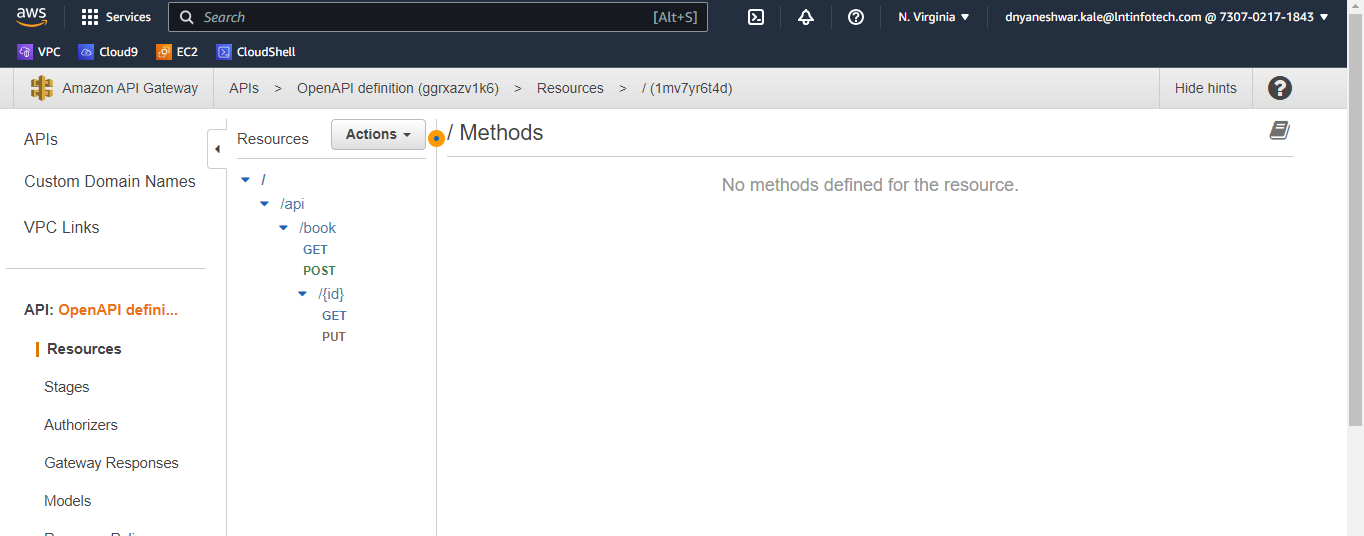
TG health point is - **/actuator/health**



**Intergrate API gateway**

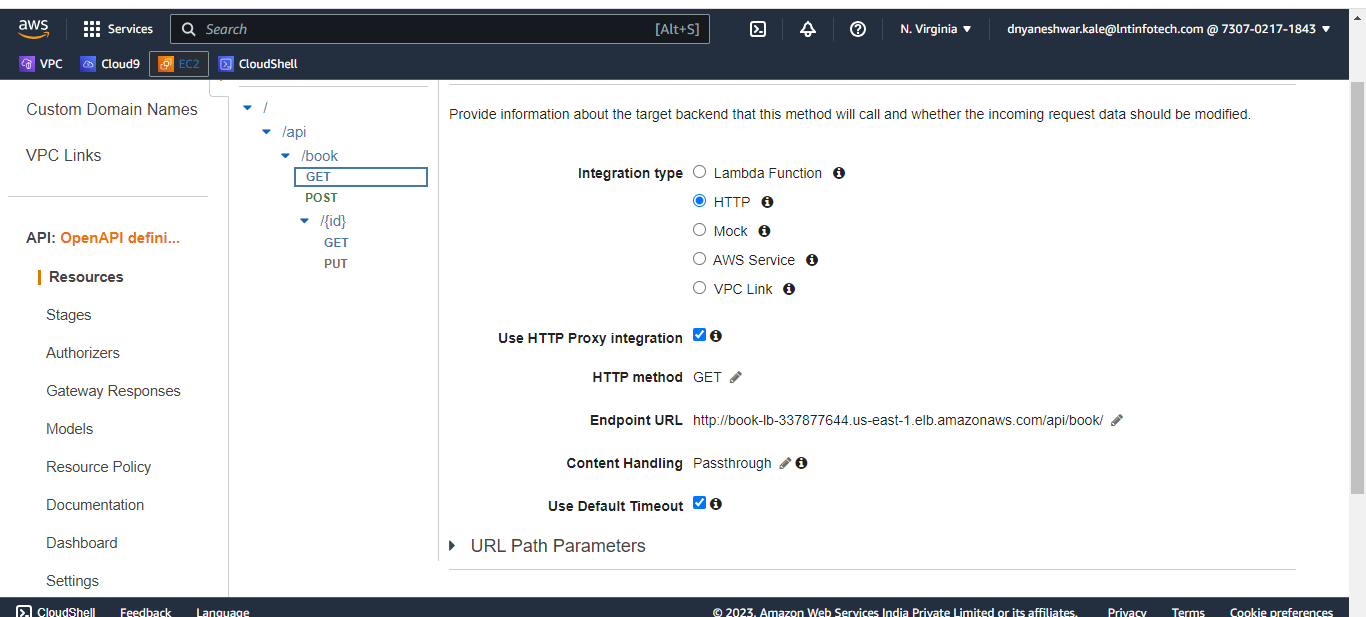
**Copy swagger docs** [**http://book-lb-337877644.us-east-1.elb.amazonaws.com/v3/api-docs**](http://book-lb-337877644.us-east-1.elb.amazonaws.com/v3/api-docs)

**Create rest api in aws api gateway using swagger api as below**

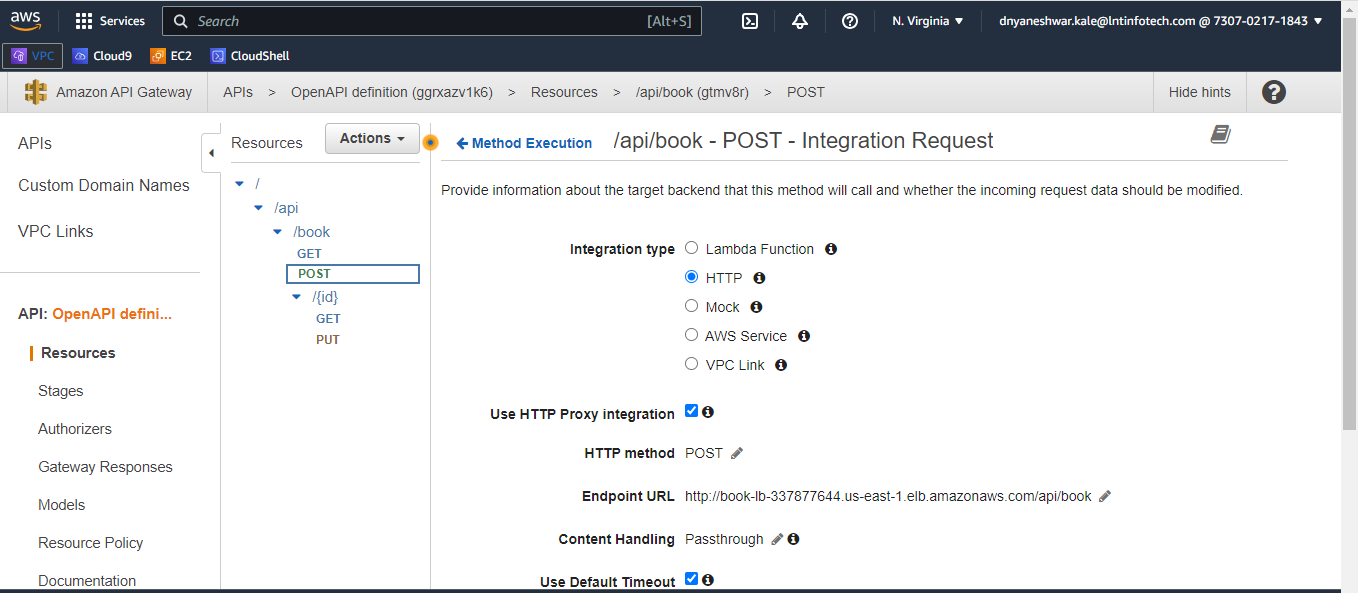


**Edit method and integrate with appropriate urls with integration type HTTP as below**

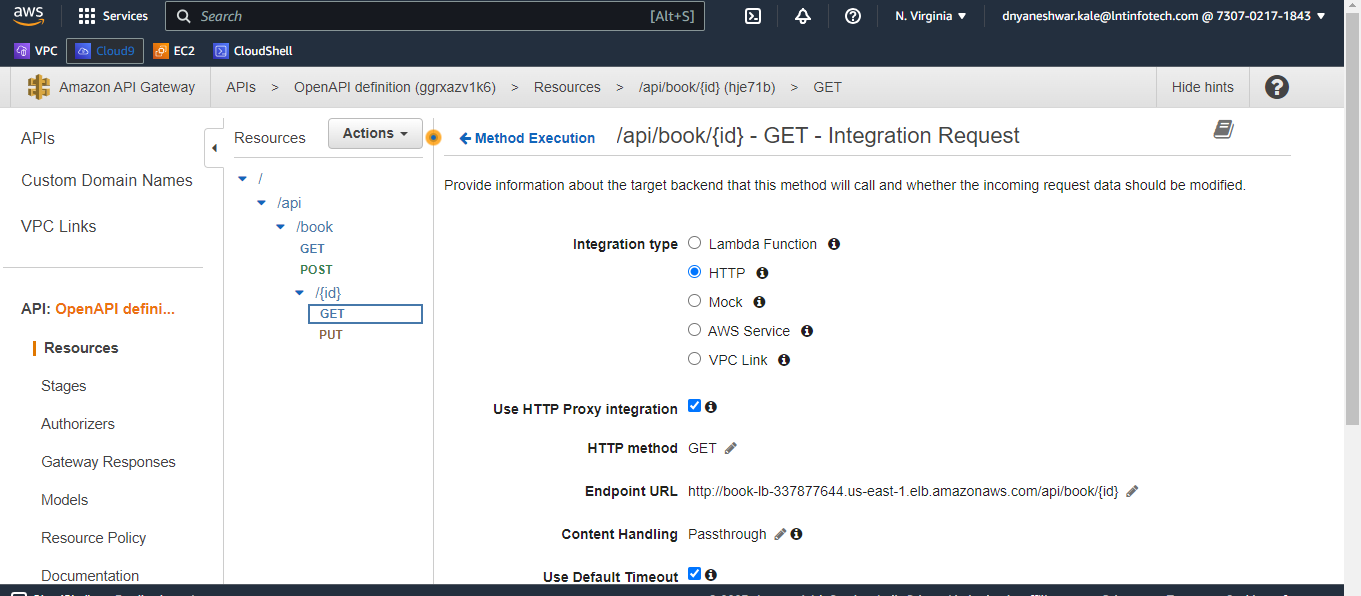
GET <http://book-lb-337877644.us-east-1.elb.amazonaws.com/api/book/>



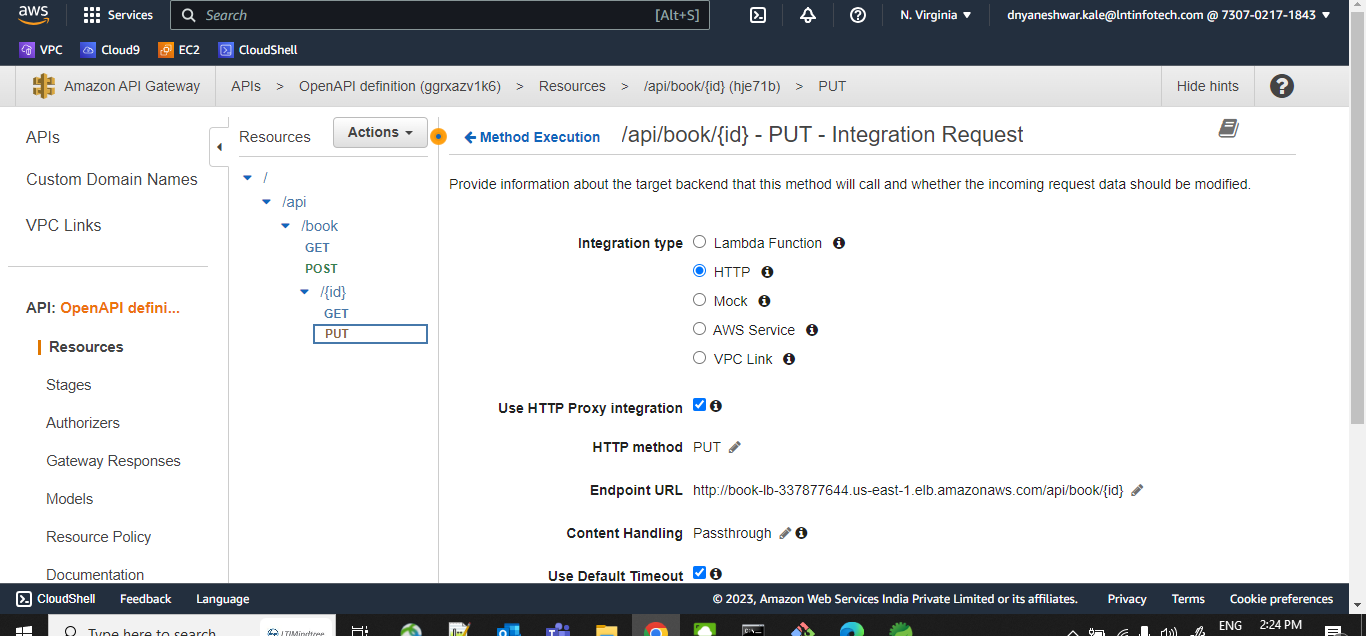
POST <http://book-lb-337877644.us-east-1.elb.amazonaws.com/api/book/>



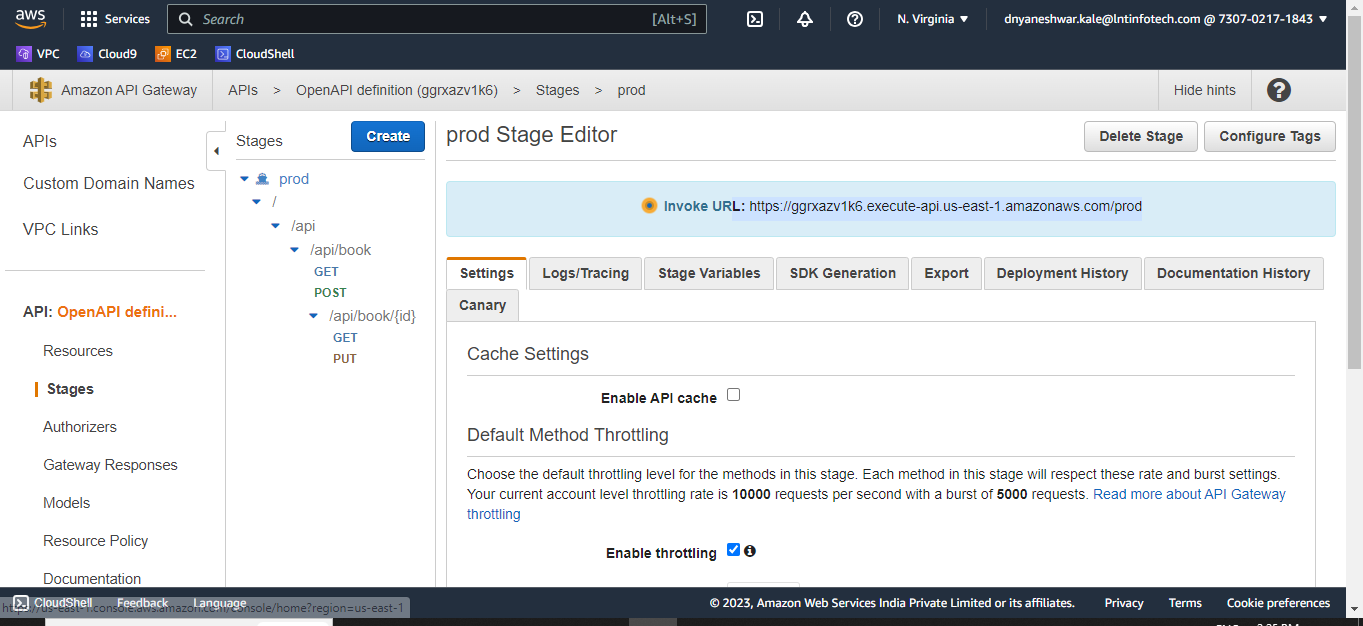
GET <http://book-lb-337877644.us-east-1.elb.amazonaws.com/api/book/{id>}



PUT <http://book-lb-337877644.us-east-1.elb.amazonaws.com/api/book/{id>}



**Deploy api , copy url**

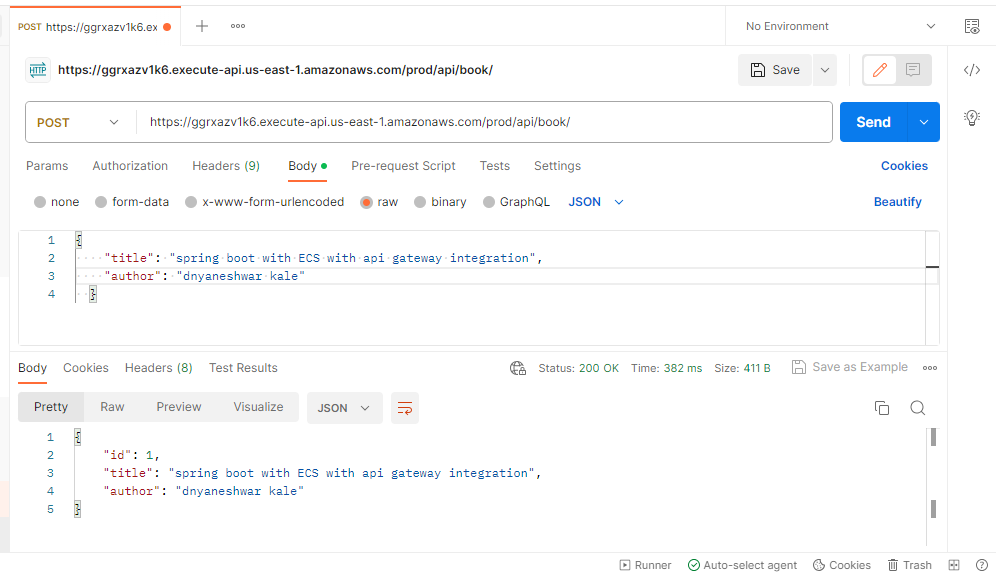


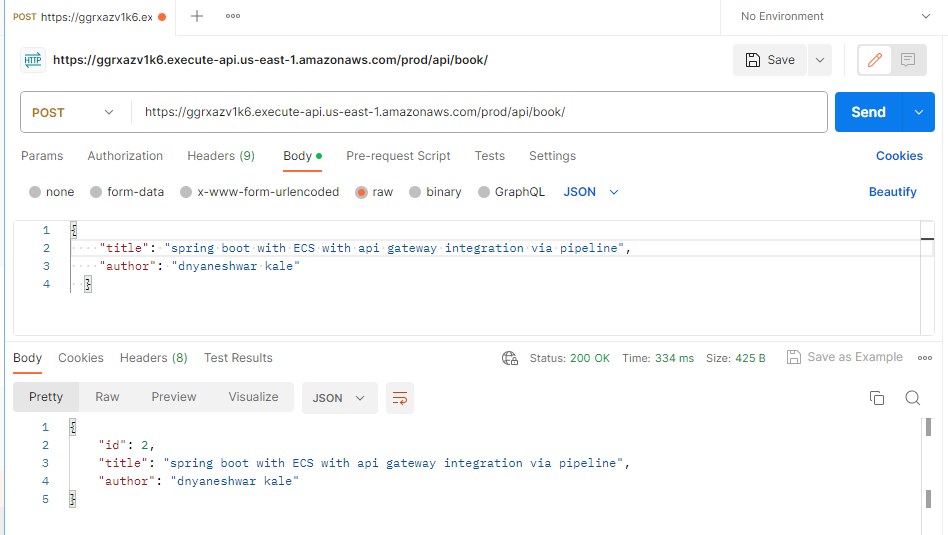
<https://ggrxazv1k6.execute-api.us-east-1.amazonaws.com/prod>

Test via postman by adding below header

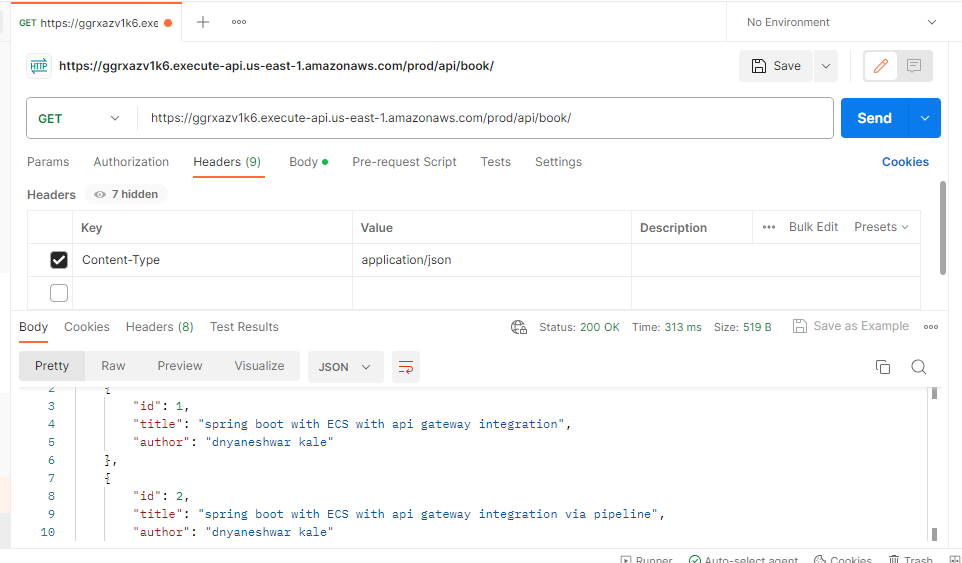
Content-Type - > application/json

POST

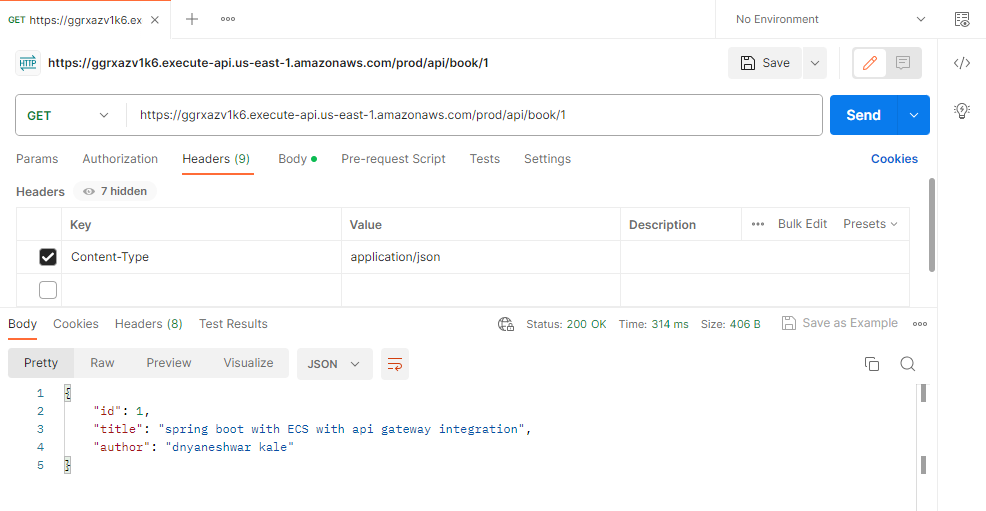




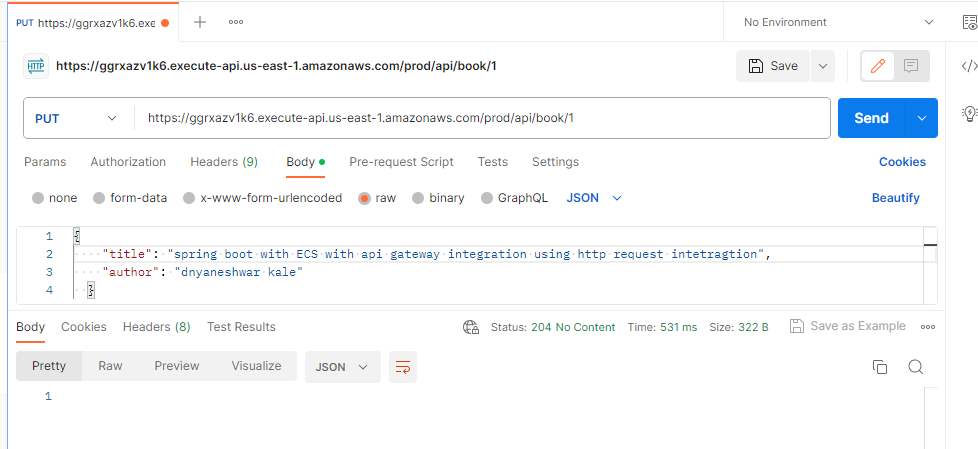
**GET ALL**

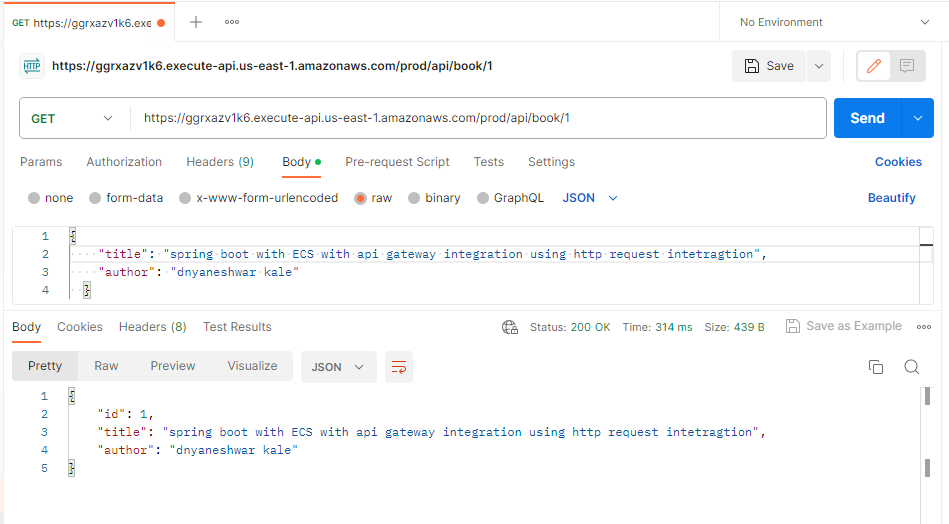


GET by id

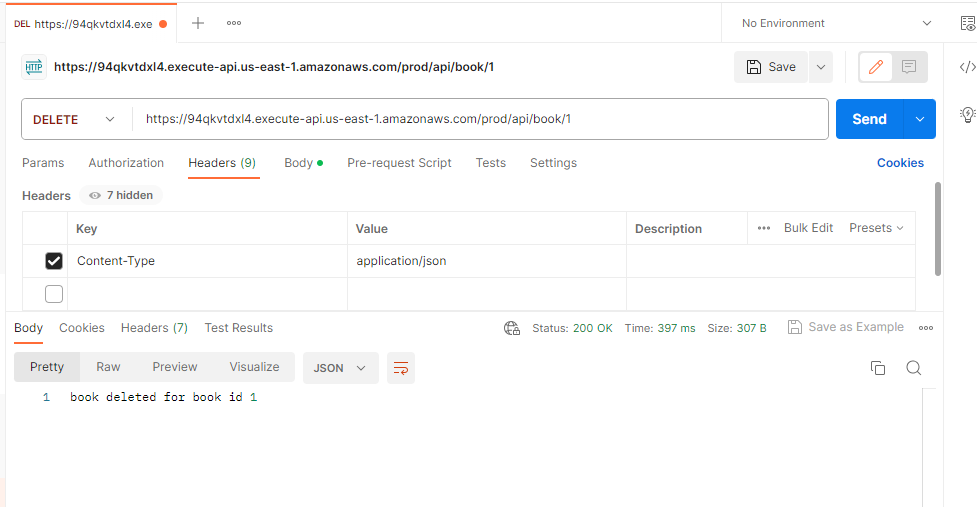


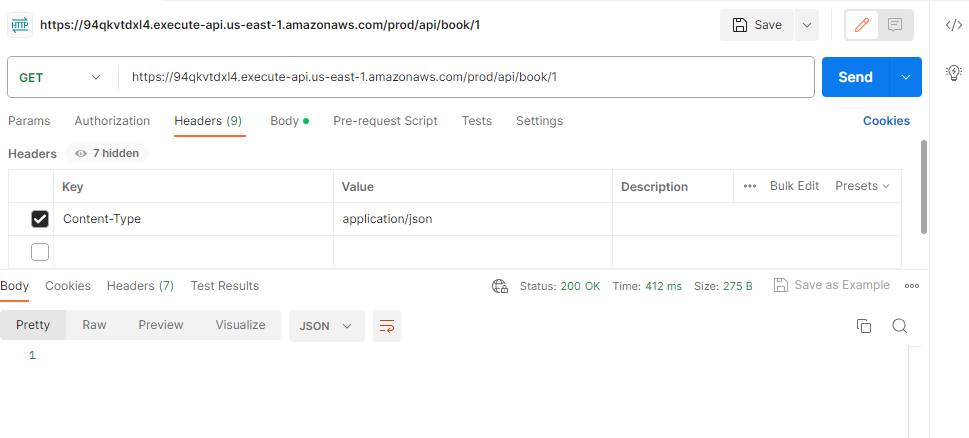
Update by id

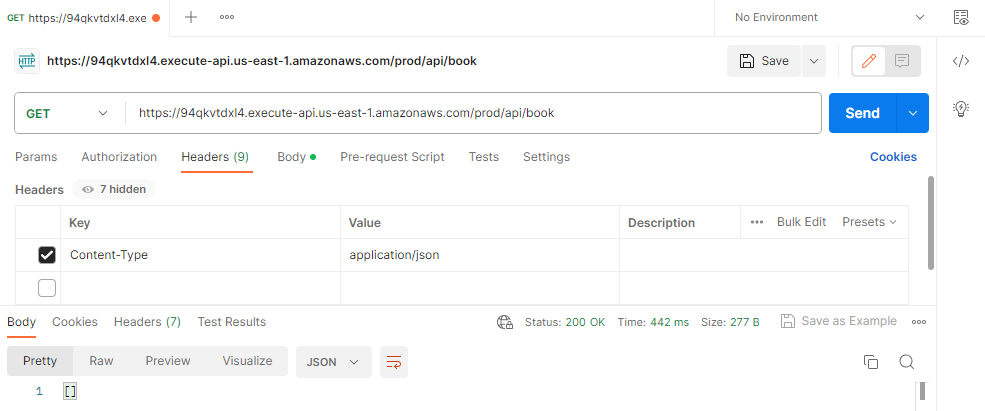




delete by id

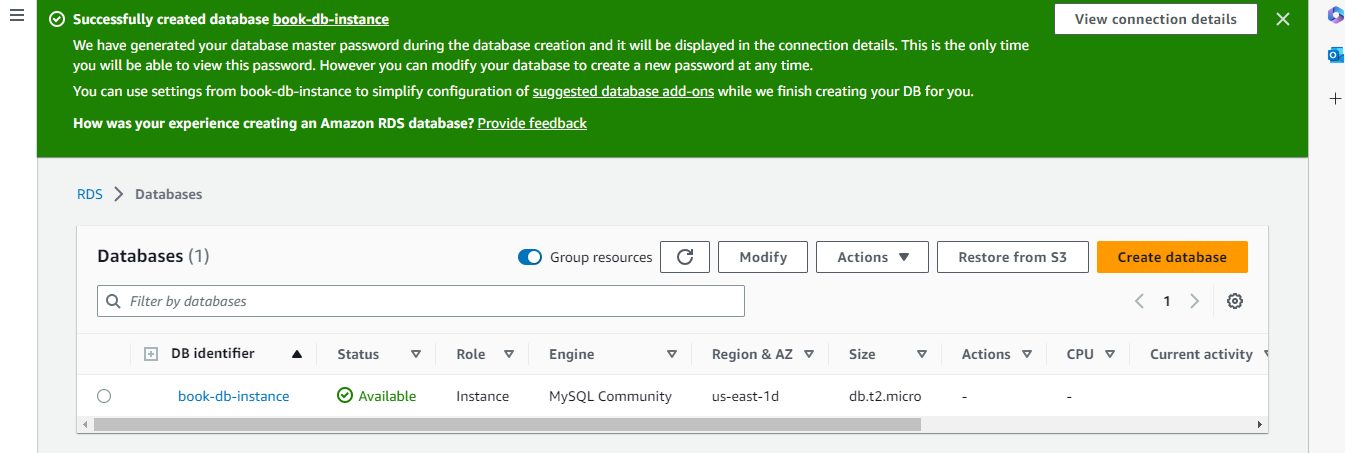






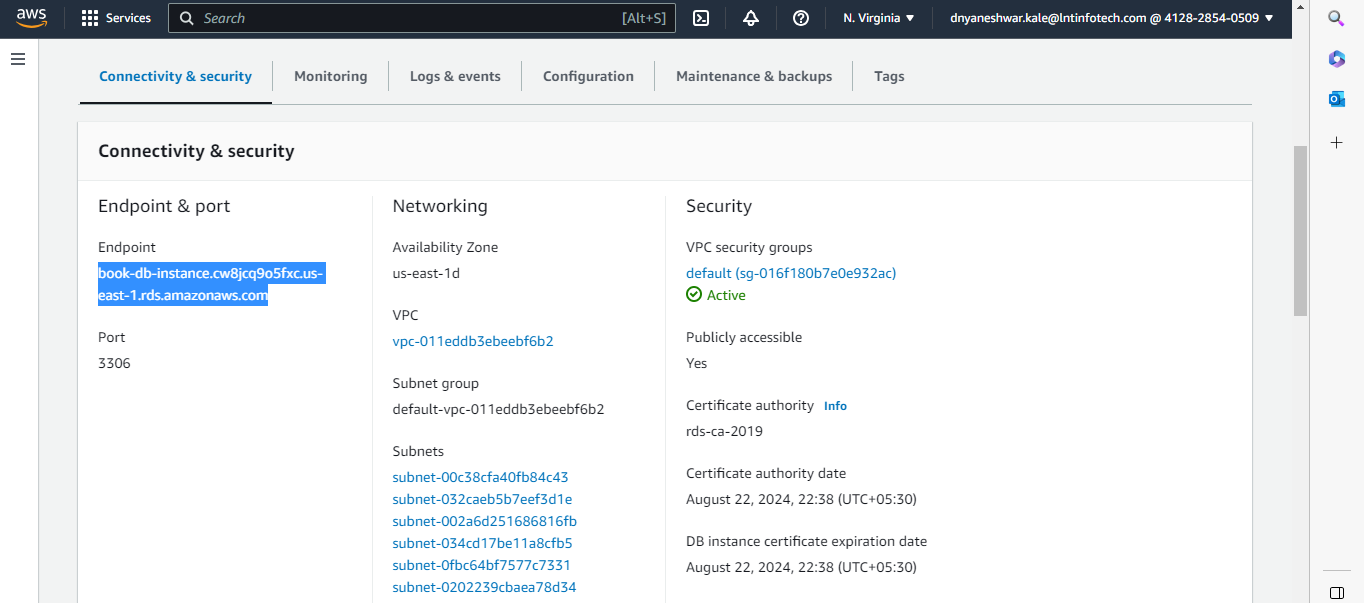
Create MYSQL RDS

MYSQL DB



Master username admin

Master password Lalp0xddNaVMauRM6qys



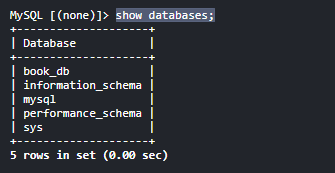
Endpoint - book-db-instance.cw8jcq9o5fxc.us-east-1.rds.amazonaws.com

Connect DB from Cloud shell

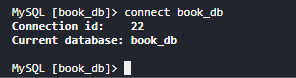
mysql -h book-db-instance.cw8jcq9o5fxc.us-east-1.rds.amazonaws.com -P 3306 -u admin -p



show databases;



connect book\_db ;



show tables;

**Spring boot integration with RDS**

Update pom.xml and application.yml file as per RDS secret

[Integrate AWS Secrets Manager in Spring Boot | Baeldung](https://www.baeldung.com/spring-boot-integrate-aws-secrets-manager)

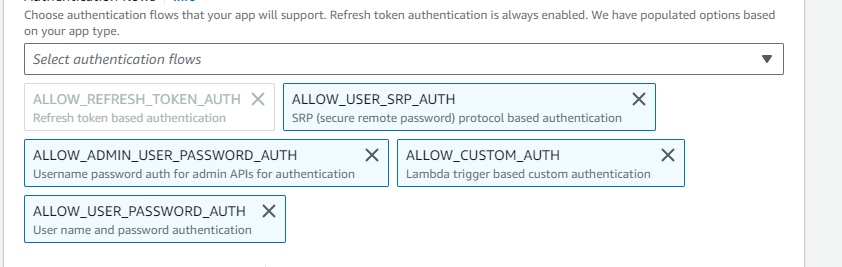
Working soltuion - [Spring Boot CRUD API, Amazon RDS for MySQL, AWS Secrets Manager - example - DEV Community](https://dev.to/aws-builders/spring-boot-crud-api-amazon-rds-for-mysql-aws-secrets-manager-example-599g)

Create aws cognito for authentication

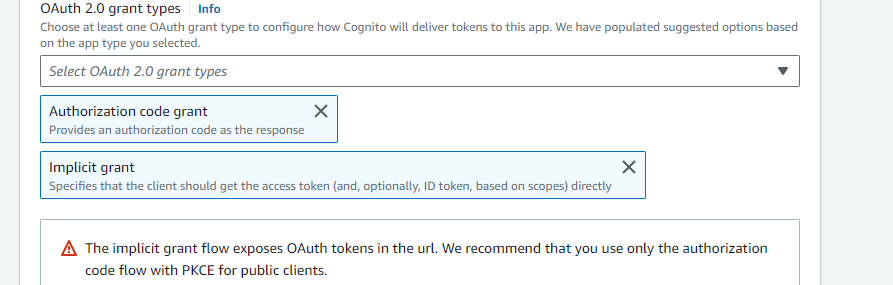
<https://mydeveloperplanet.com/2022/01/25/how-to-secure-aws-api-gateway-with-cognito-user-pool/>

<https://www.youtube.com/watch?v=LI31QxfAgho>

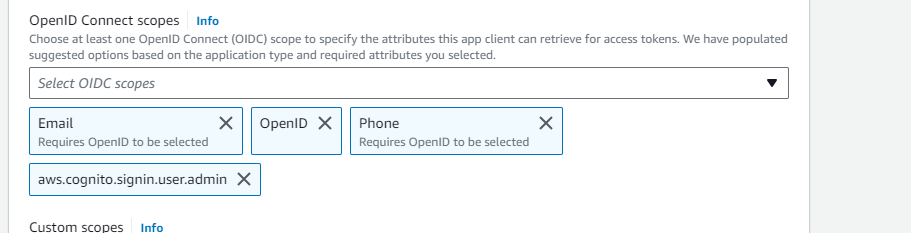
Below check box should be checked while creating client app or you can edit later

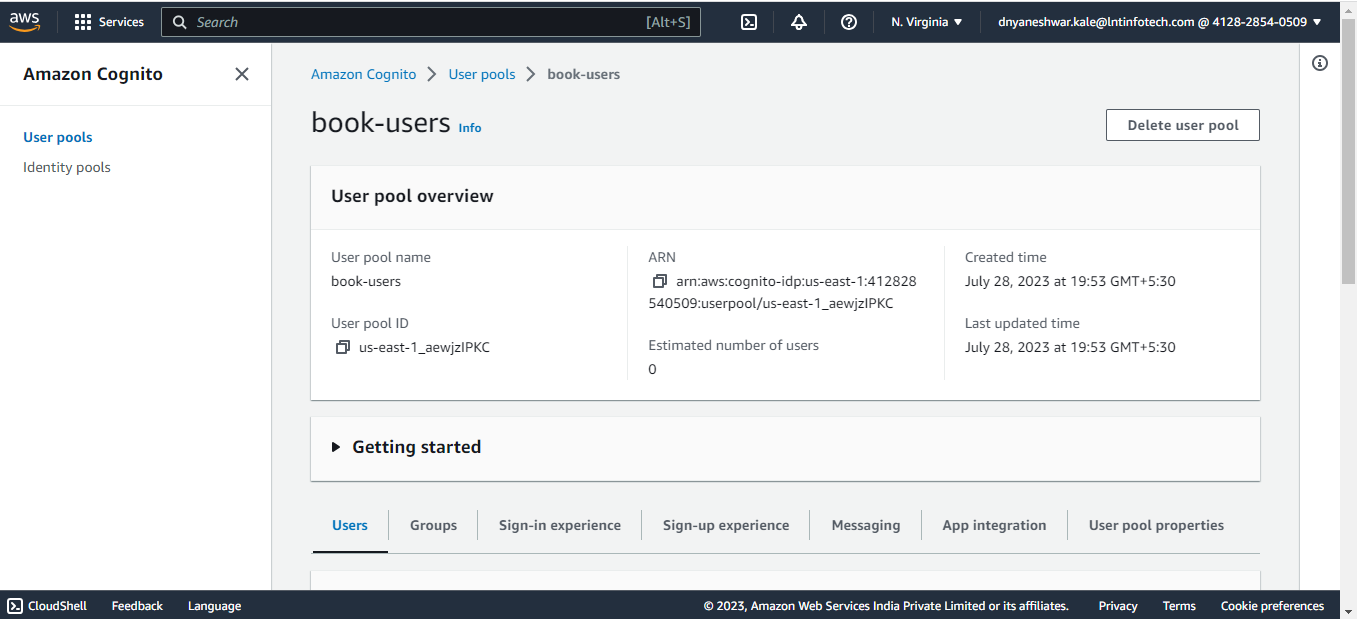


OAuth 2.0 grant types -> Implicit grant checked for allow idtoken response type from sign in page

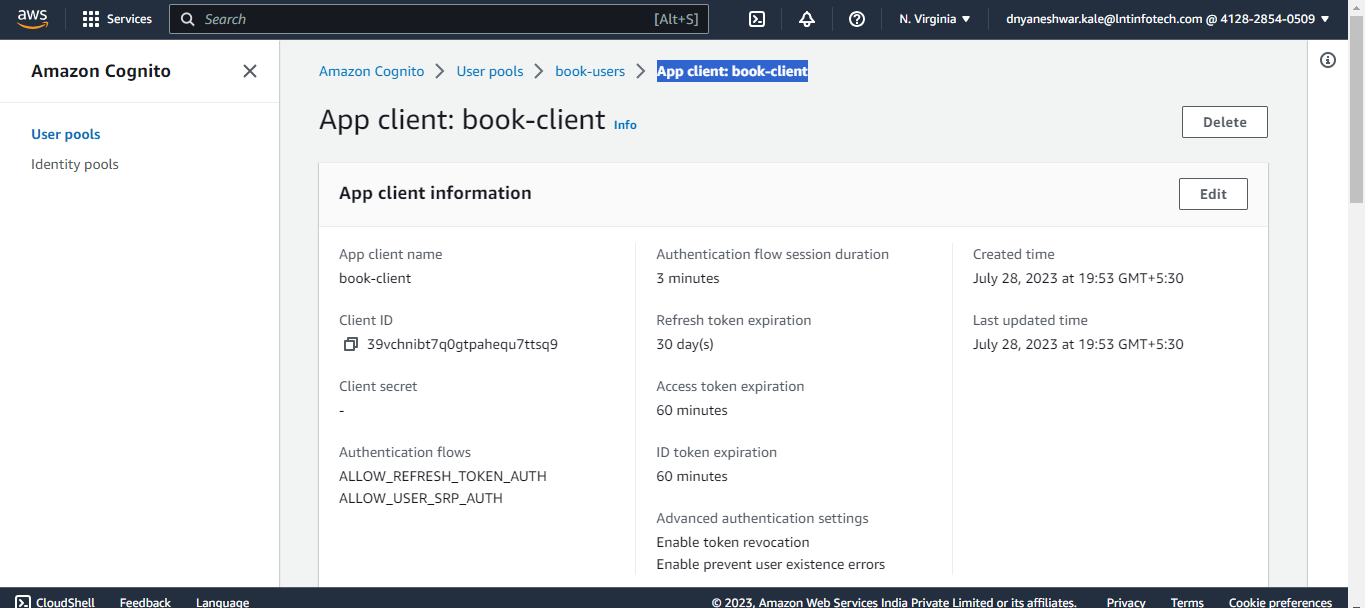


OpenID Connect scopes -> aws.cognito.signin.user.admin and Profile checked for allow idtoken response type from sign in page



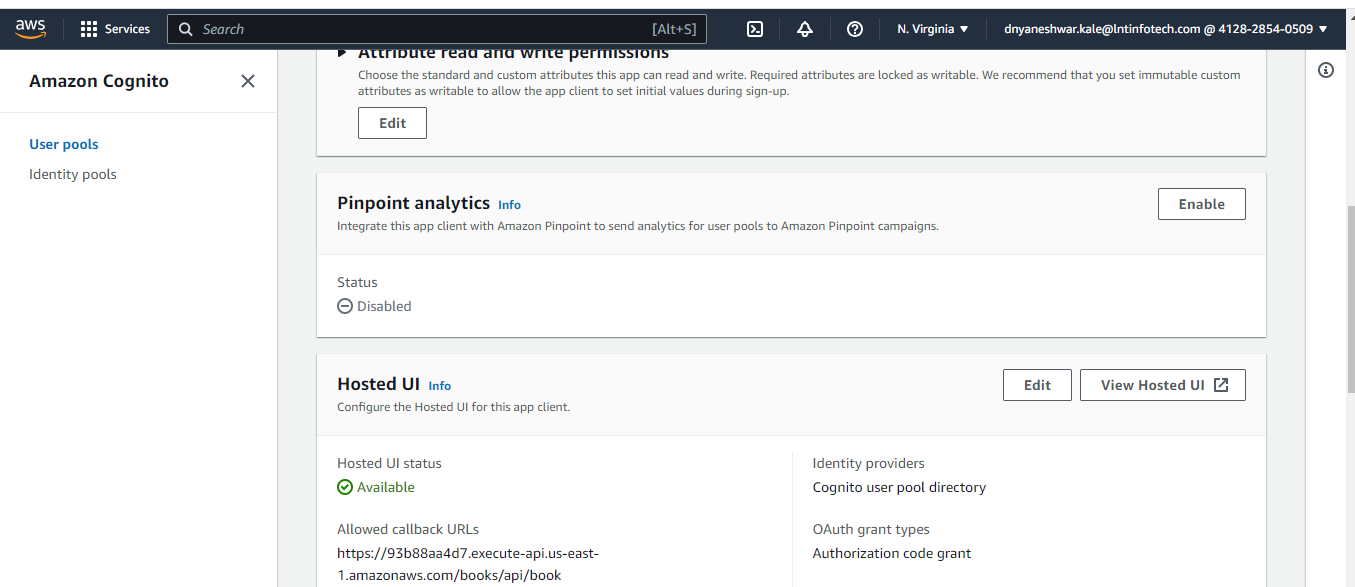


Go to app intergration -> **App client: book-client**



Click on view hosted UI

<https://book-demo.auth.us-east-1.amazoncognito.com/login?client_id=39vchnibt7q0gtpahequ7ttsq9&response_type=code&scope=email+openid+phone&redirect_uri=https%3A%2F%2F93b88aa4d7.execute-api.us-east-1.amazonaws.com%2Fbooks%2Fapi%2Fbook>

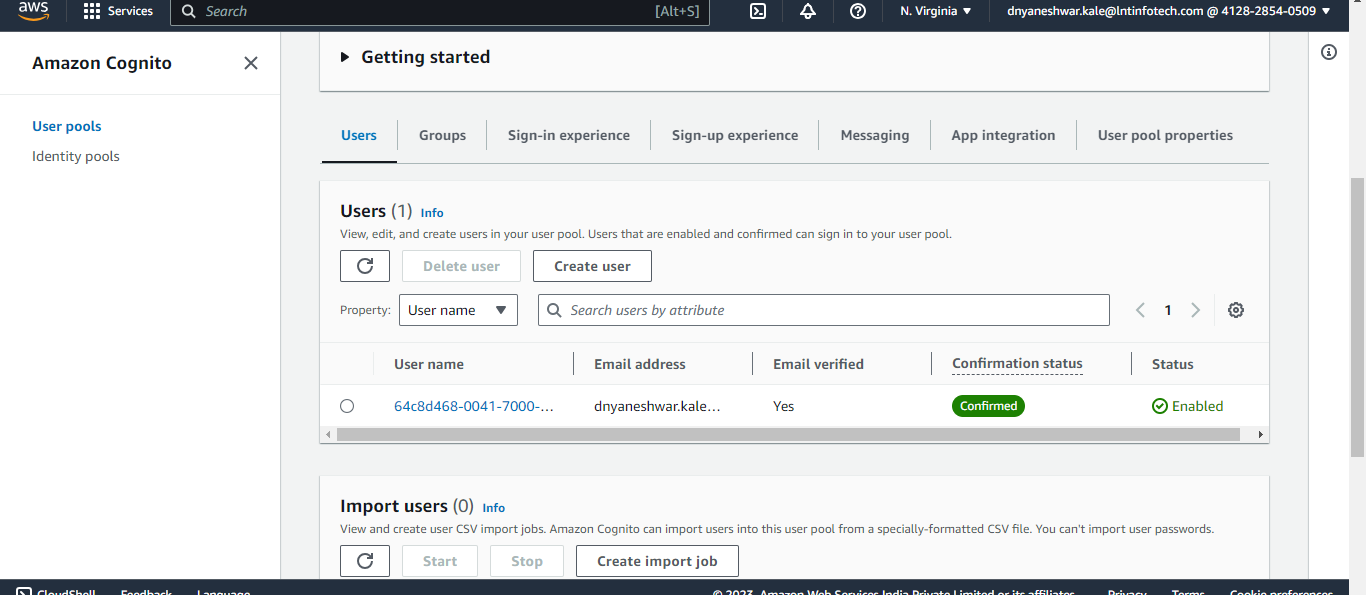




Sign up



User is added and able to redirect on api gateway





Sign in by updating url replace response\_type -> code with token to generate idtoken

https://book-app.auth.us-east-1.amazoncognito.com/login?client\_id=5i0ql0rglc9ju438m486v3cakk&response\_type=token&scope=aws.cognito.signin.user.admin+email+openid+phone+profile&redirect\_uri=https%3A%2F%2F93b88aa4d7.execute-api.us-east-1.amazonaws.com%2Fbooks%2Fapi%2Fbook

Copy token from url

https://93b88aa4d7.execute-api.us-east-1.amazonaws.com/books/api/book#id\_token=eyJraWQiOiJsWk1ubmZqZFY2YXVkR3RMZVZaRks1K3NVVjVDU1hiUGYyOWRPbitHb2NRPSIsImFsZyI6IlJTMjU2In0..YW61hIJOEHilPg2\_Z7ioj9keAPuIu2ZkcBjAuwchwe1Dd5ZEBWkFoS4xh4DNGPEBs1j19Vv0hgKtO8P5EnuOqgxeqs\_U\_lNwO2SwP-0wY\_KiH4GqPc-8l\_oc30NGSt6vTYA2bCy1\_ojANHFOHkI9yoYhcRLShhqU77SHRiEFkerZAixr1-Eg-taVAL0jxNWd9Jupog0\_jCbc9RNfkBKj5hHpGXz43ahrrYBV-ERpkty7j\_xvt4FYvb7p1KhcgTrm2VA3NrGHED3z-nNho60QwWU2tXs9\_iToWpS4mBNpKrg2MwDBp0W-Q-Rx\_ACNkxf7cm6Eq5uC8KOUoClz05fCJA&access\_token=eyJraWQiOiJqNzZSM2traVk1WDV5dzEyQjI1S0t0N3hzNWVDbG5cL3NDclR2OWF3NzA4RT0iLCJhbGciOiJSUzI1NiJ9..B-uBipoSFmyGYE3PVWydWnQQK5kpKJRkIjL\_4BgNlnTuzAuzBlwCIGblhfbSzeeaLmEtvqpPvM8uOeeG6FS4wdvXvq6MAtPEGs4oi2XQZBaGwQmgDOdpEl0q78ICAzyTaT6rfDYcbHVsbGFMj\_ORZ-2nbPr-UFGeIUFkSK8tQj5mczbyrd1nYxDDLosuMqO05pqQh\_WsN6ZPzxWoRs7QejzfmVGDtoW9KJnsabgsb5n7JfThv3dteMfEq9RQSKy44INy8RLvTEoQxbqDH-AD3hoGYQCrALX2U3AMylDE8Jj3Gl9xrOsZ68TBuTLBMFYWT2XcbRwzD5lr901949j8jw&expires\_in=3600&token\_type=Bearer

OR

**Generate token using aws cli**

aws cognito-idp admin-initiate-auth --user-pool-id us-east-1\_eZSFmpME2 --client-id 1cpfhk4mtilmm9eh8nfu0ejbtm --auth-flow ADMIN\_NO\_SRP\_AUTH --auth-parameters USERNAME=dnyaneshwar.kale@lntinfotech.com,PASSWORD=Ltim@123

{

"ChallengeParameters": {},

"AuthenticationResult": {

"AccessToken": "eyJraWQiOiJqNzZSM2traVk1WDV5dzEyQjI1S0t0N3hzNWVDbG5cL3NDclR2OWF3NzA4RT0iLCJhbGciOiJSUzI1NiJ9..YmBNVp6Mu2t3KqzIBA2DOufw67bLoGOscTsiQc4JtJoDCMiFUjqnfsTKotnbX-FOtyZ-yJX7GHRZEYY6Eu\_dyZAGQW9Ki3J5DvO0iq7yJ29PQWKQilt5ZdL89FHRFwi3PdWOJ862fgBc4Zn0mHs33fWxlFrD9WcJZB-YjnM\_ZXla4KjWH-RPn7SPk\_Jw6jLclcLxaJLekQ6v--SNKEPckjIfhdL48pPg9mW3Eix4CwNRW2as1hl9\_AVHyF0\_Ohe7nc585tRw-DIK65JKa4uF5P1kxMjk0gt-NpFaQ2QsVwyadMGRFPvFTGRR0s8-gxuuj7GSYrcJoyhgrtEeU9vFIg",

"ExpiresIn": 3600,

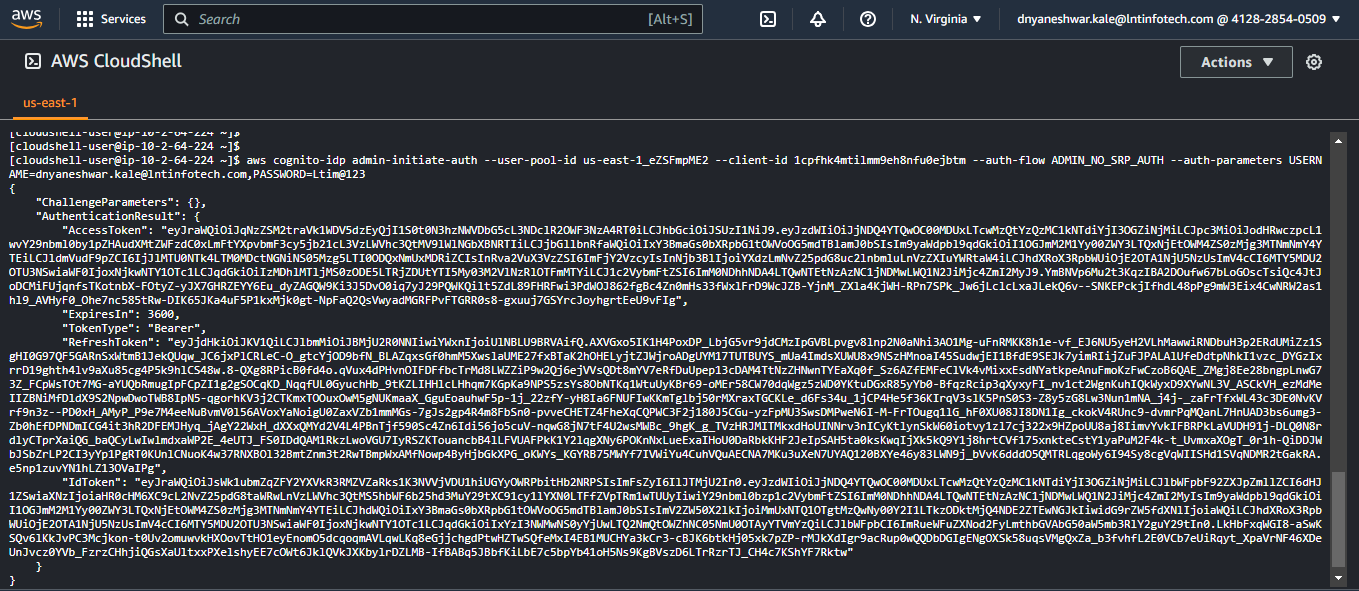
"TokenType": "Bearer",

"RefreshToken": "eyJjdHkiOiJKV1QiLCJlbmMiOiJBMjU2R0NNIiwiYWxnIjoiUlNBLU9BRVAifQ.AXVGxo5IK1H4PoxDP\_LbjG5vr9jdCMzIpGVBLpvgv8lnp2N0aNhi3AO1Mg-uFnRMKK8h1e-vf\_EJ6NU5yeH2VLhMawwiRNDbuH3p2ERdUMiZz1SgHI0G97QF5GARnSxWtmB1JekQUqw\_JC6jxPlCRLeC-O\_gtcYjOD9bfN\_BLAZqxsGf0hmM5XwslaUME27fxBTaK2hOHELyjtZJWjroADgUYM17TUTBUYS\_mUa4ImdsXUWU8x9NSzHMnoaI45SudwjEI1BfdE9SEJk7yimRIijZuFJPALAlUfeDdtpNhkI1vzc\_DYGzIxrrD19ghth4lv9aXu85cg4P5k9hlCS48w.8-QXg8RPicB0fd4o.qVux4dPHvnOIFDFfbcTrMd8LWZZiP9w2Qj6ejVVsQDt8mYV7eRfDuUpep13cDAM4TtNzZHNwnTYEaXq0f\_Sz6AZfEMFeClVk4vMixxEsdNYatkpeAnuFmoKzFwCzoB6QAE\_ZMgj8Ee28bngpLnwG73Z\_FCpWsTOt7MG-aYUQbRmugIpFCpZI1g2gSOCqKD\_NqqfUL0GyuchHb\_9tKZLIHHlcLHhqm7KGpKa9NPS5zsYs8ObNTKq1WtuUyKBr69-oMEr58CW70dqWgz5zWD0YKtuDGxR85yYb0-BfqzRcip3qXyxyFI\_nv1ct2WgnKuhIQkWyxD9XYwNL3V\_ASCkVH\_ezMdMeIIZBNiMfDldX9S2NpwDwoTWB8IpN5-qgorhKV3j2CTKmxTOOuxOwM5gNUKmaaX\_GguEoauhwF5p-1j\_22zfY-yH8Ia6FNUFIwKKmTglbj50rMXraxTGCKLe\_d6Fs34u\_1jCP4He5f36KIrqV3slK5PnS0S3-Z8y5zG8Lw3Nun1mNA\_j4j-\_zaFrTfxWL43c3DE0NvKVrf9n3z--PD0xH\_AMyP\_P9e7M4eeNuBvmV0l56AVoxYaNoigU0ZaxVZb1mmMGs-7gJs2gp4R4m8FbSn0-pvveCHETZ4FheXqCQPWC3F2j180J5CGu-yzFpMU3SwsDMPweN6I-M-FrTOugq1lG\_hF0XU08JI8DN1Ig\_ckokV4RUnc9-dvmrPqMQanL7HnUAD3bs6umg3-Zb0hEfDPNDmICG4it3hR2DFEMJHyq\_jAgY22WxH\_dXXxQMYd2V4L4PBnTjf590Sc4Zn6Idi56jo5cuV-nqwG8jN7tF4U2wsMWBc\_9hgK\_g\_TVzHRJMITMkxdHoUINNrv3nICyKtlynSkW60iotvy1zl7cj322x9HZpoUU8aj8IimvYvkIFBRPkLaVUDH91j-DLQ0N8rdlyCTprXaiQG\_baQCyLwIwlmdxaWP2E\_4eUTJ\_FS0IDdQAM1RkzLwoVGU7IyRSZKTouancbB4lLFVUAFPkK1Y2lqgXNy6POKnNxLueExaIHoU0DaRbkKHF2JeIpSAH5ta0ksKwqIjXk5kQ9Y1j8hrtCVf175xnkteCstY1yaPuM2F4k-t\_UvmxaXOgT\_0r1h-QiDDJWbJSbZrLP2CI3yYp1PgRT0KUnlCNuoK4w37RNXBOl32BmtZnm3t2RwTBmpWxAMfNowp4ByHjbGkXPG\_oKWYs\_KGYRB75MWYf7IVWiYu4CuhVQuAECNA7MKu3uXeN7UYAQ120BXYe46y83LWN9j\_bVvK6dddO5QMTRLqgoWy6I94Sy8cgVqWIISHd1SVqNDMR2tGakRA.e5np1zuvYN1hLZ13OVaIPg",

"IdToken": "eyJraWQiOiJsWk1ubmZqZFY2YXVkR3RMZVZaRks1K3NVVjVDU1hiUGYyOWRPbitHb2NRPSIsImFsZyI6IlJTMjU2In0..LkHbFxqWGI8-aSwKSQv6lKkJvPC3Mcjkon-t0Uv2omuwvkHXOovTtHO1eyEnomO5dcqoqmAVLqwLKq8eGjjchgdPtwHZTwSQfeMxI4EB1MUCHYa3kCr3-cBJK6btkHj05xk7pZP-rMJkXdIgr9acRup0wQQDbDGIgENgOXSk58uqsVMgQxZa\_b3fvhfL2E0VCb7eUiRqyt\_XpaVrNF46XDeUnJvcz0YVb\_FzrzCHhjiQGsXaUltxxPXelshyEE7cOWt6JklQVkJXKbylrDZLMB-IfBABq5JBbfKiLbE7c5bpYb41oH5Ns9KgBVszD6LTrRzrTJ\_CH4c7KShYF7Rktw"

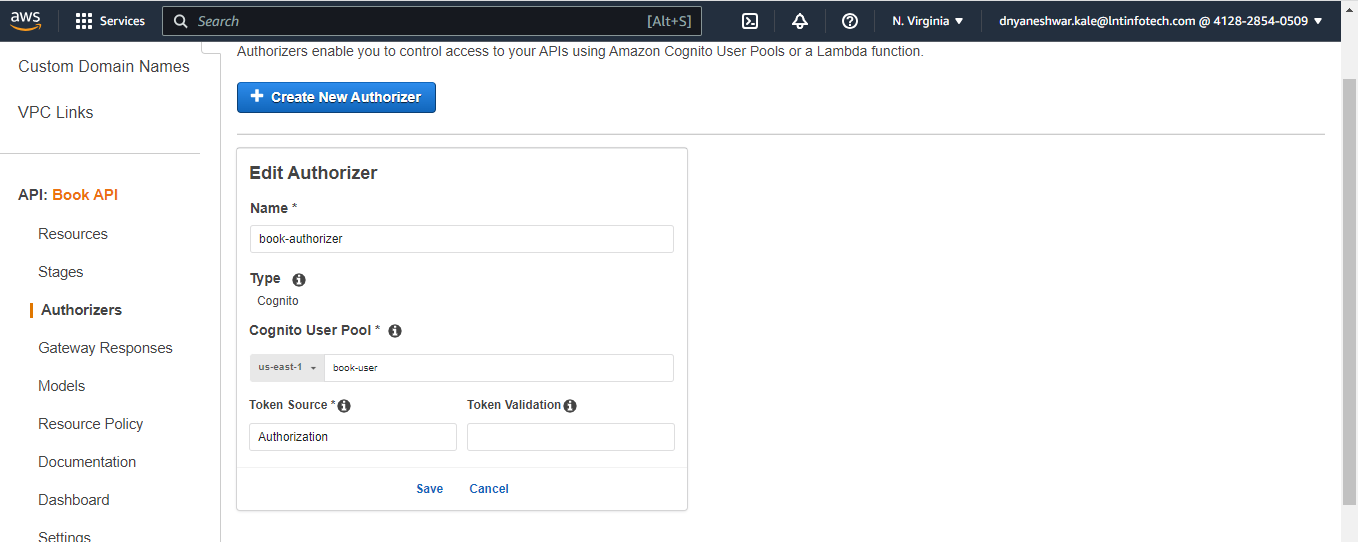
}

}

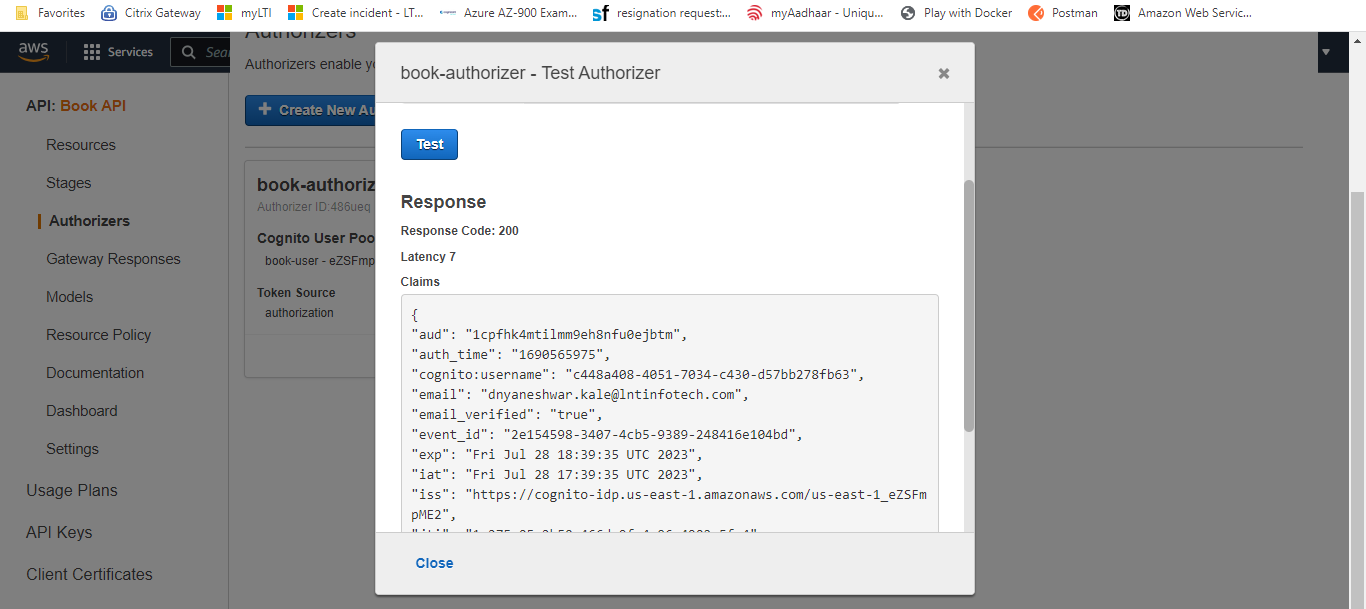


**Go to your api gateway -> Authorizers and create with cognito**

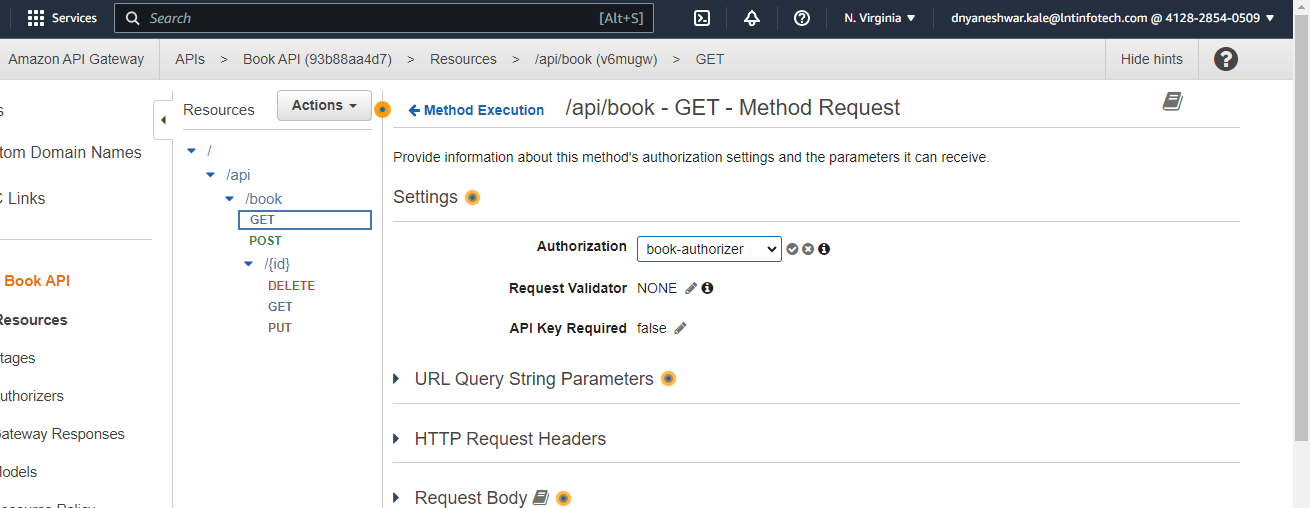
When creating client app



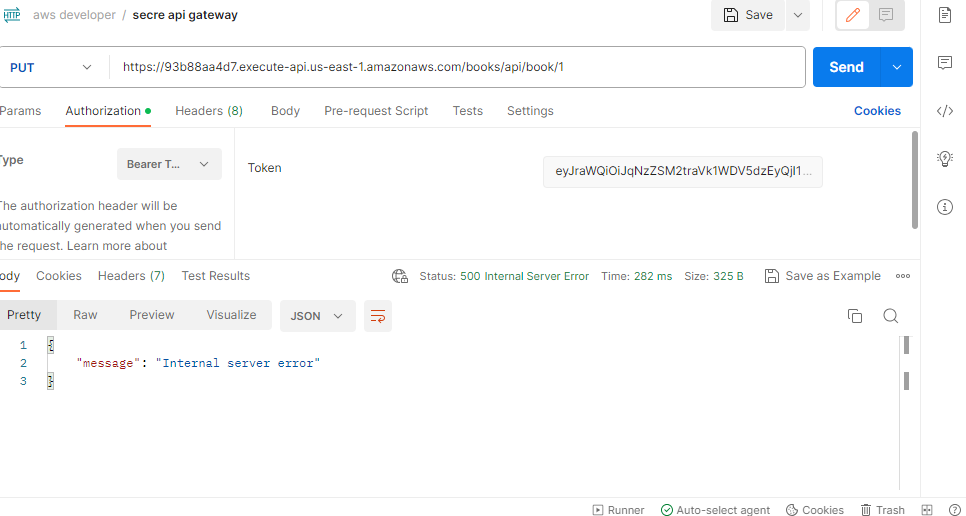
**Copy id token and test Authorizer**



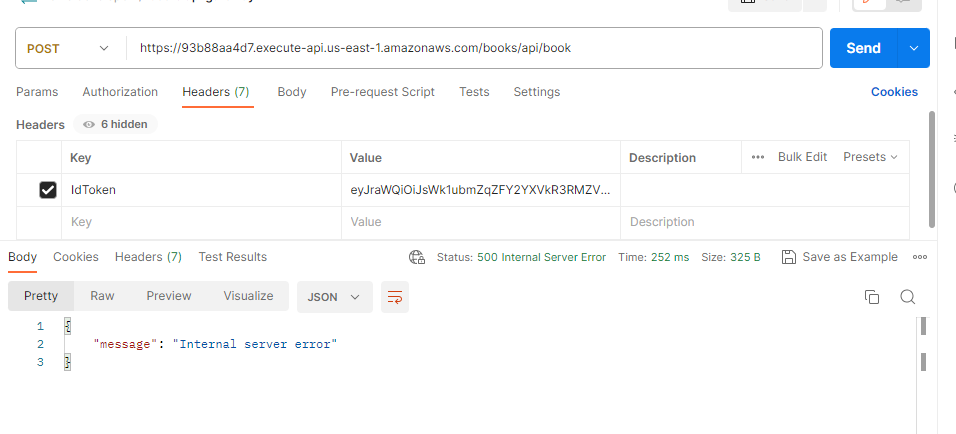
Add above authorizer in your resource method as below ,



**Deploy it and test from postman**

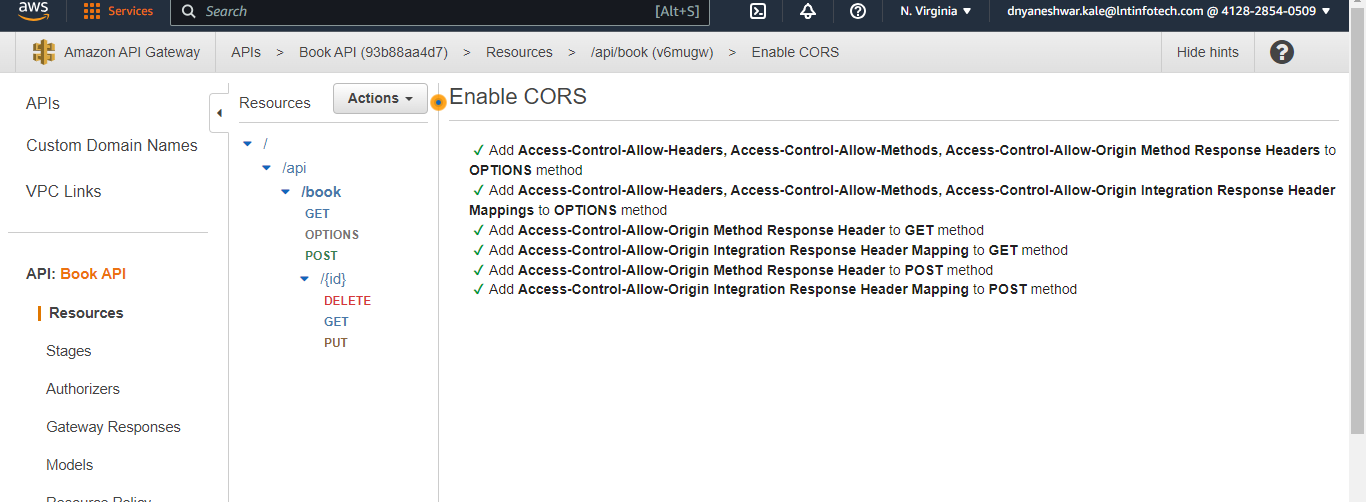


OR



**Enbale CORS policy**

[**https://medium.com/geekculture/simple-steps-to-enable-cors-in-api-gateway-through-console-cloud-formation-c09d9df31c07**](https://medium.com/geekculture/simple-steps-to-enable-cors-in-api-gateway-through-console-cloud-formation-c09d9df31c07)

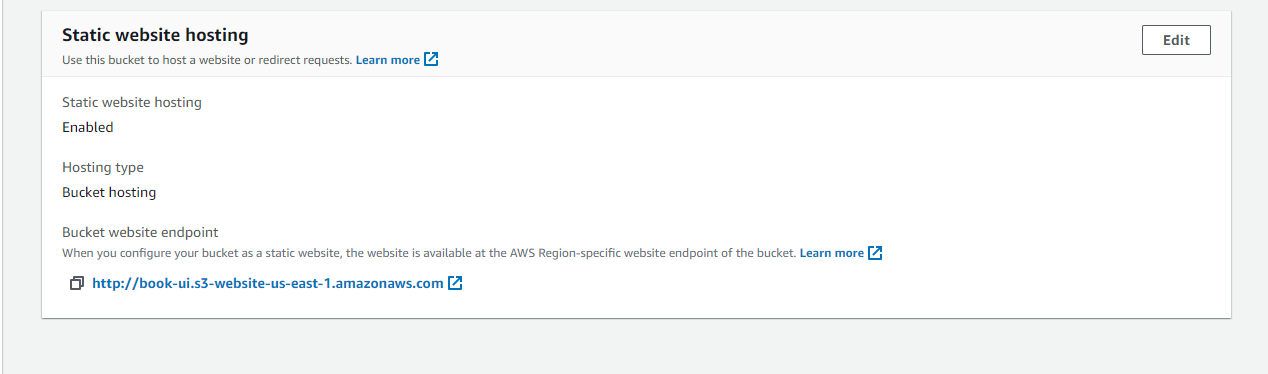


**Build and deploy in s3**

Build - ng build --configuration production --aot=false --build-optimizer=false

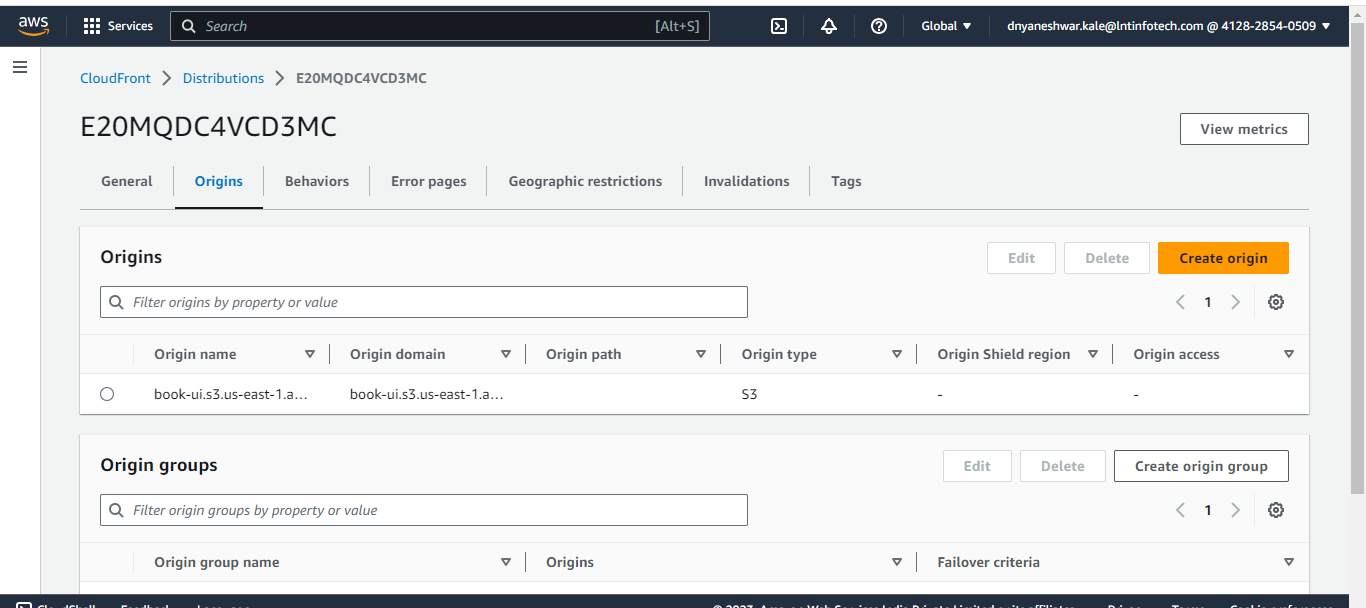
Deploy in s3

<https://baljindersingh013.medium.com/angular-app-deployment-with-aws-s3-42d9008734ab>



<http://book-ui.s3-website-us-east-1.amazonaws.com/>

Create CloudFormation for integrate with Cognito url redirect on https



**Clean up all resources**