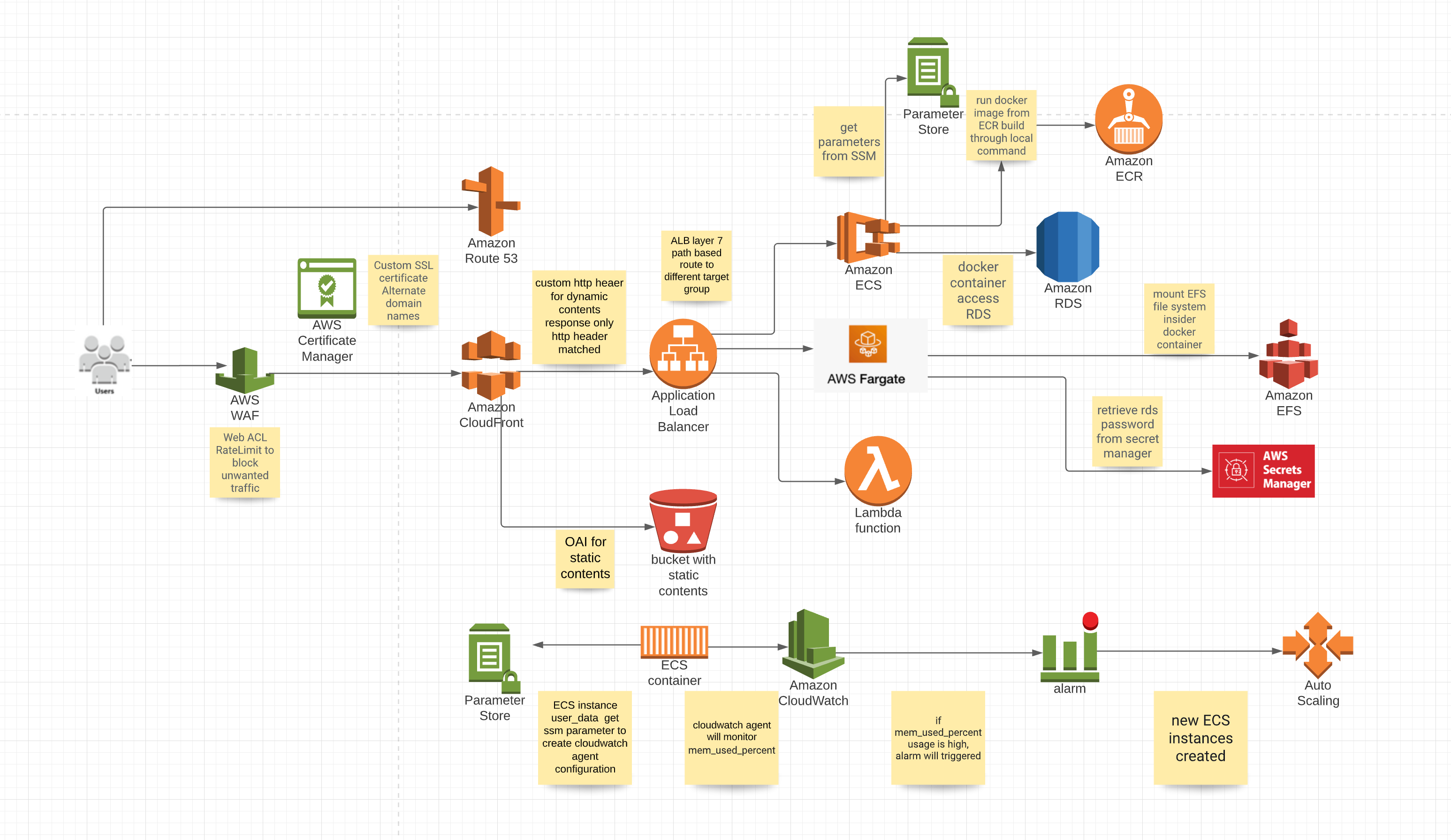
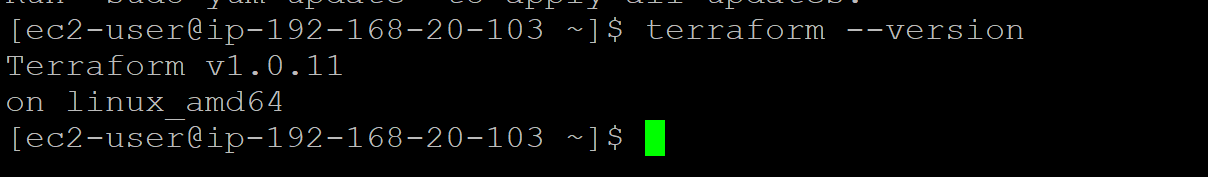
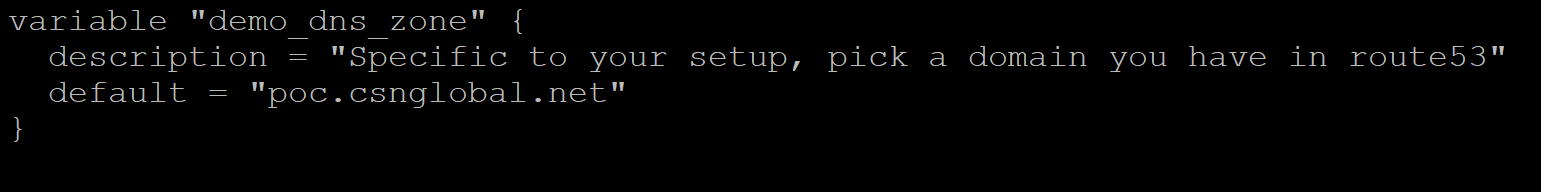
**Basic diagram as following**



**I used following terraform version**

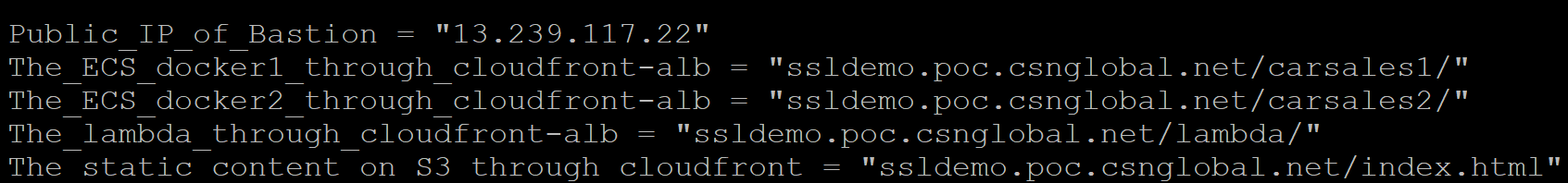


Please modify variables.tf file to change environment variable, most of variables can use default value, the one need change is demo\_dns\_zone, must be one AWS account registered Route53 public zone.



* it will create one Route53 record in the zone, the default is ssldemo, make sure there is no existing one in your Route53 public domain.
* It will create one ECR repo, the default name is test\_ecr\_repo,make sure there is no ECR repo under your account.
* It will create one S3 bucket name [www.test1234567890](http://www.test1234567890) which used for cloudfront to store static contents. There should be no name conflict there.
* It will create one ECS cluster name carsales , change it through modify variable "ecs\_cluster\_name"

Terraform will output Cloudfront alias different path URL and bastion host public IP



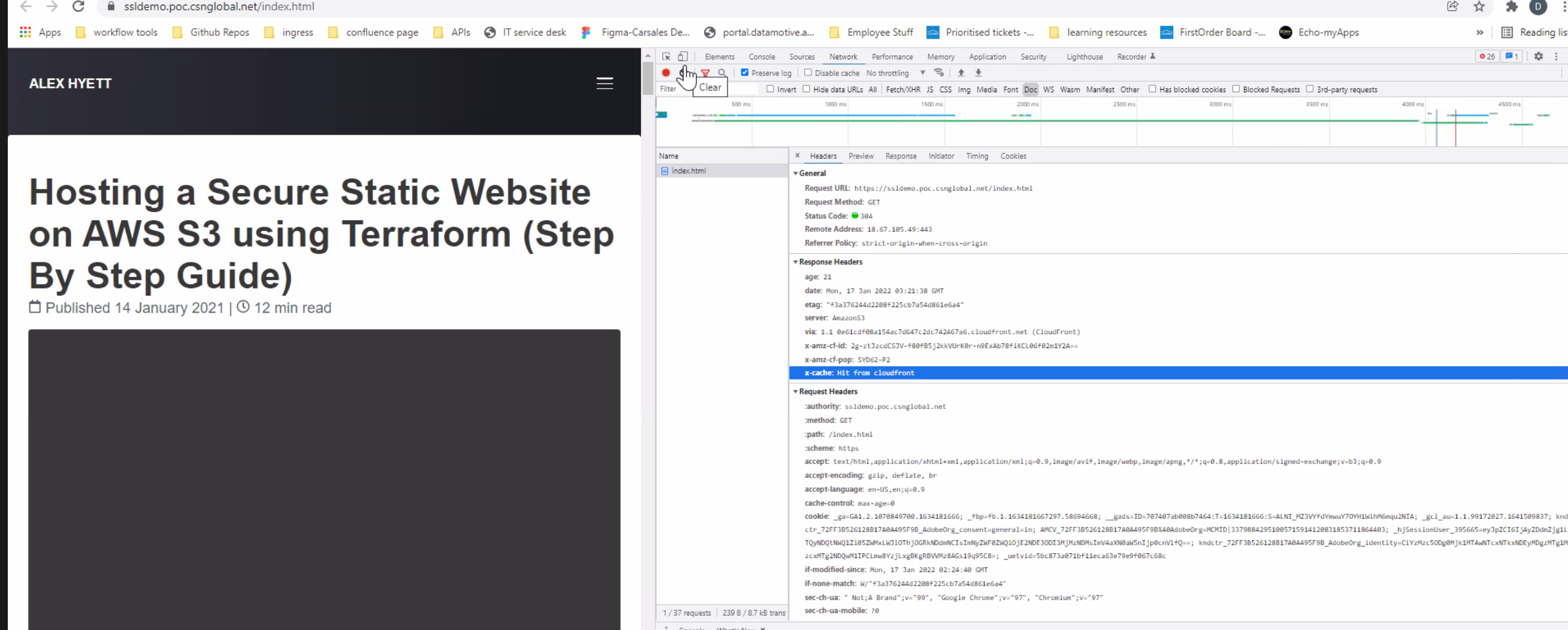
<https://ssldemo.poc.csnglobal.net/carsales1/> backend is AWS ECS EC2 image from AWS ECR

<https://ssldemo.poc.csnglobal.net/carsales2/> backend is AWS ECS Fargate image from public repo

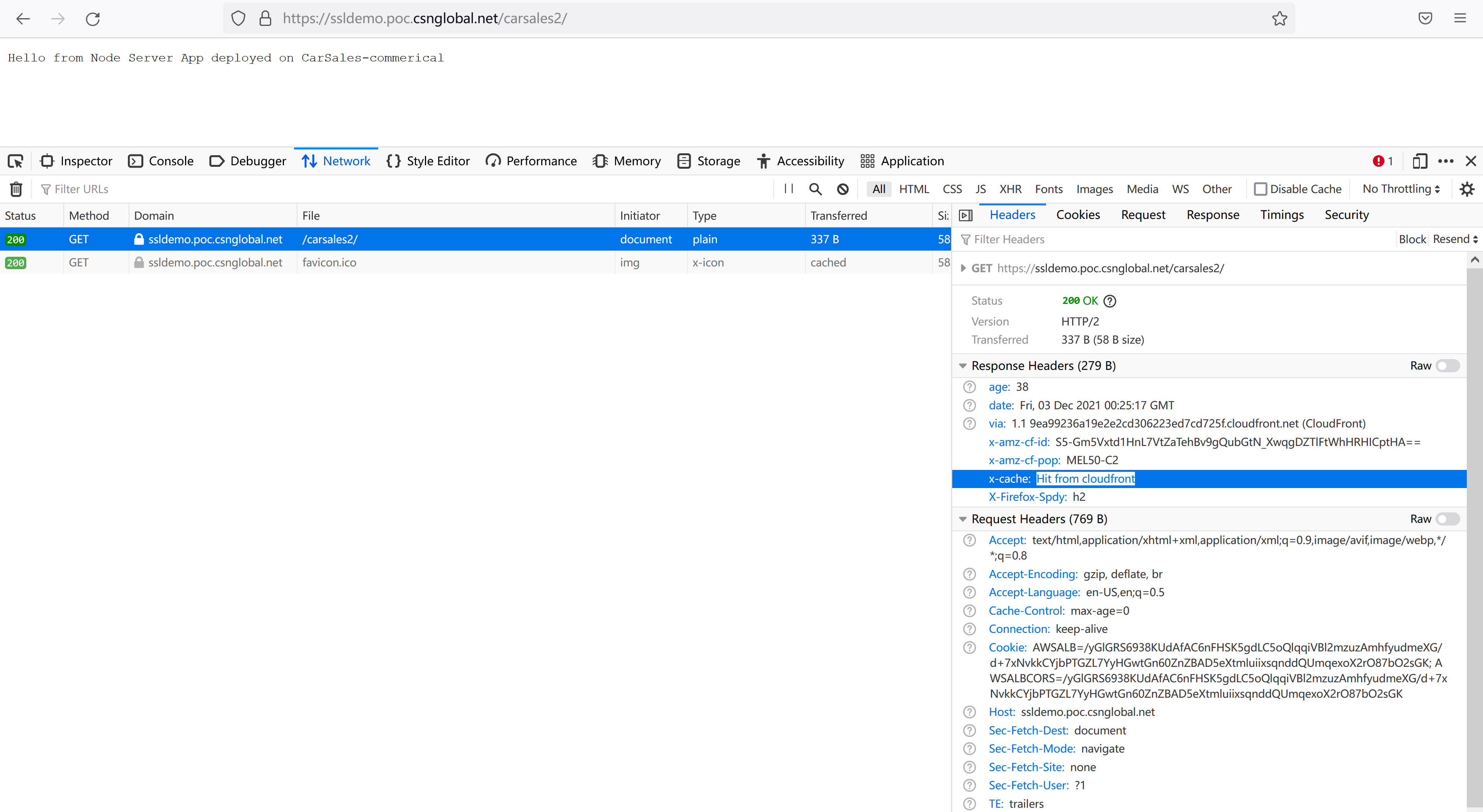
<https://ssldemo.poc.csnglobal.net/lambda/> backend is AWS lambda function

<https://ssldemo.poc.csnglobal.net/index.html> backend is AWS S3 bucket

When you first open <https://ssldemo.poc.csnglobal.net/index.html> ,check network information, it will show miss from cloudfront,



When you refresh the page again, it will show hit from cloudfront.



Access ECS EC2 instance from bastion host or AWS Session Manager. Login to container and check command env output (you can find RDS db\_url and RDS database password which can be used to access database from docker container if application needed ) and df -hT (EFS mount) to store data outside in case docker container scale in or docker container crash.



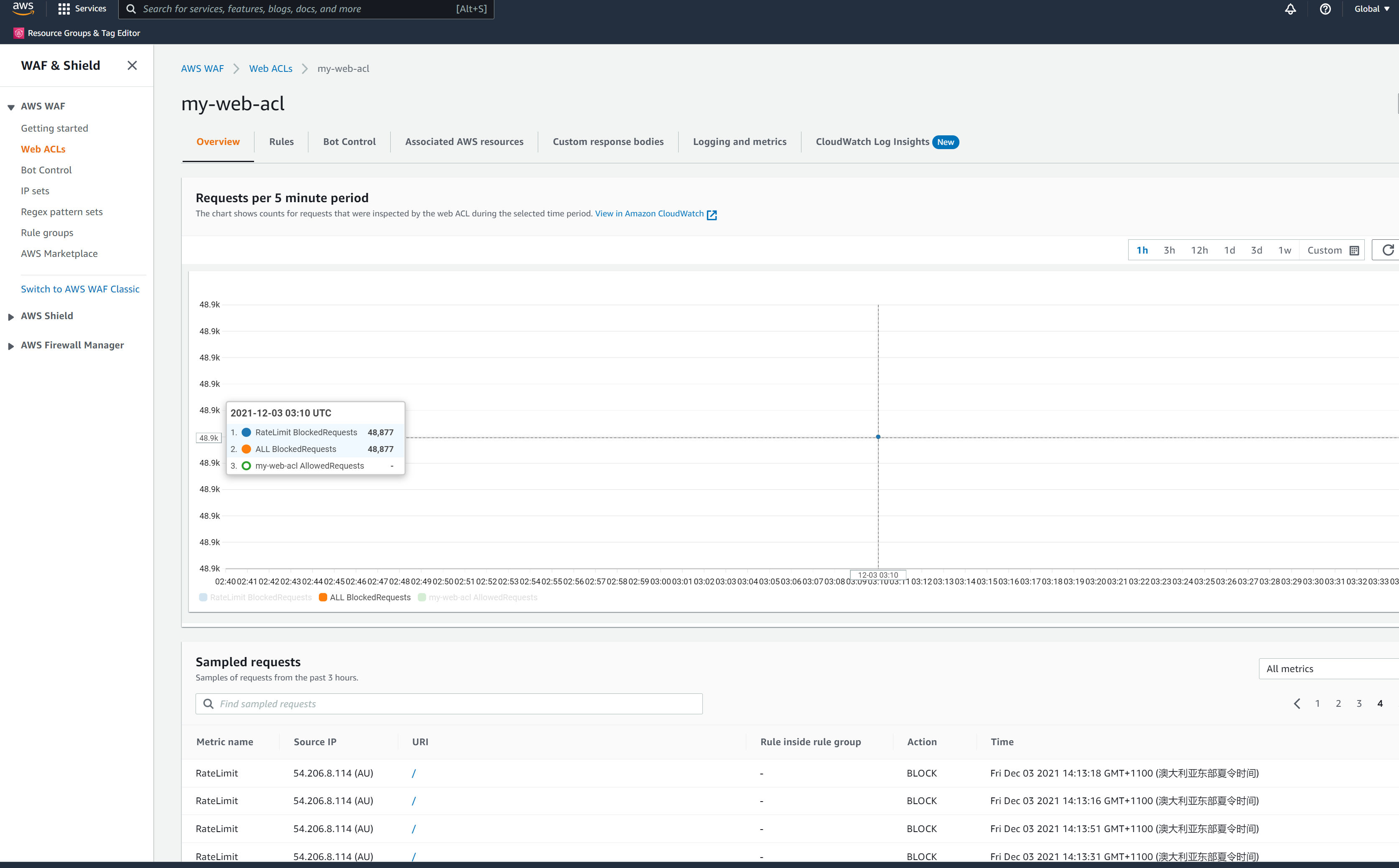
Security consideration

* When check http of url( curl -iI http://ssldemo.poc.csnglobal.net/carsales1/) , it will redirect to https

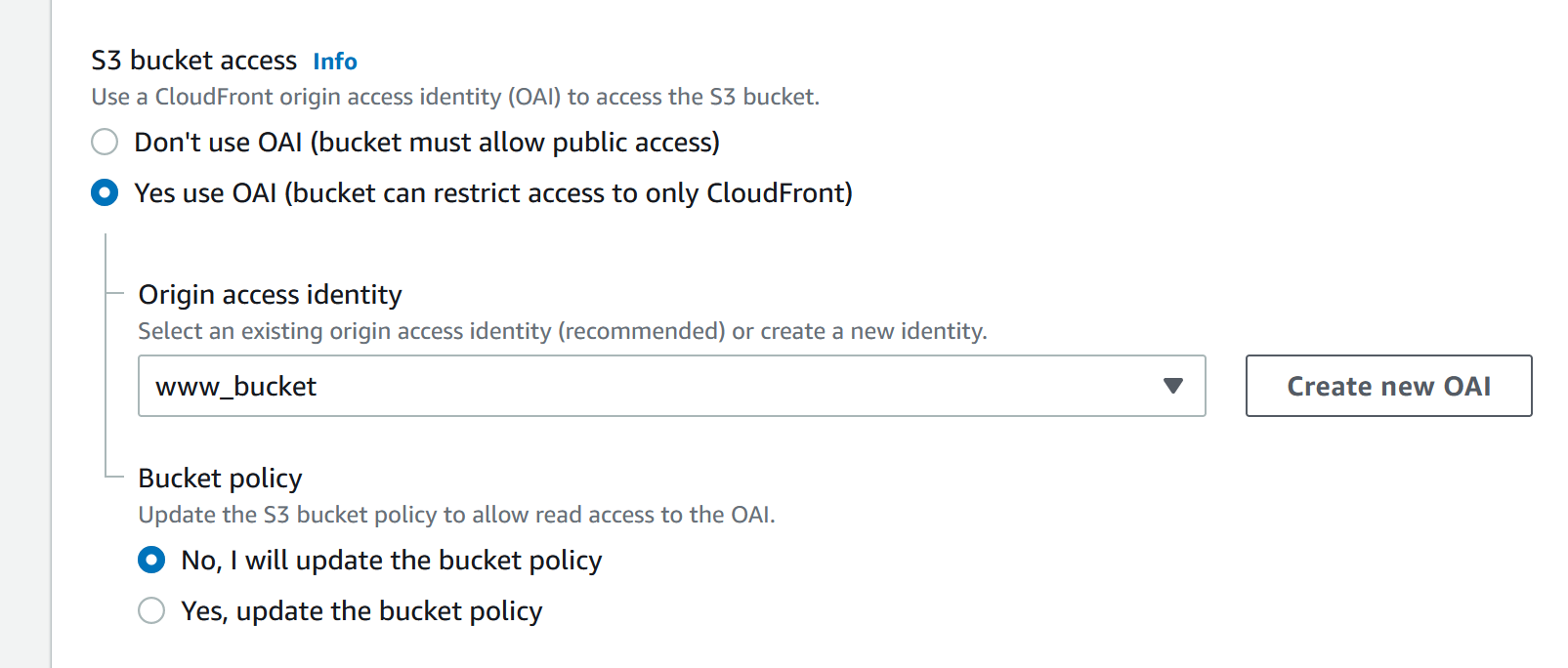


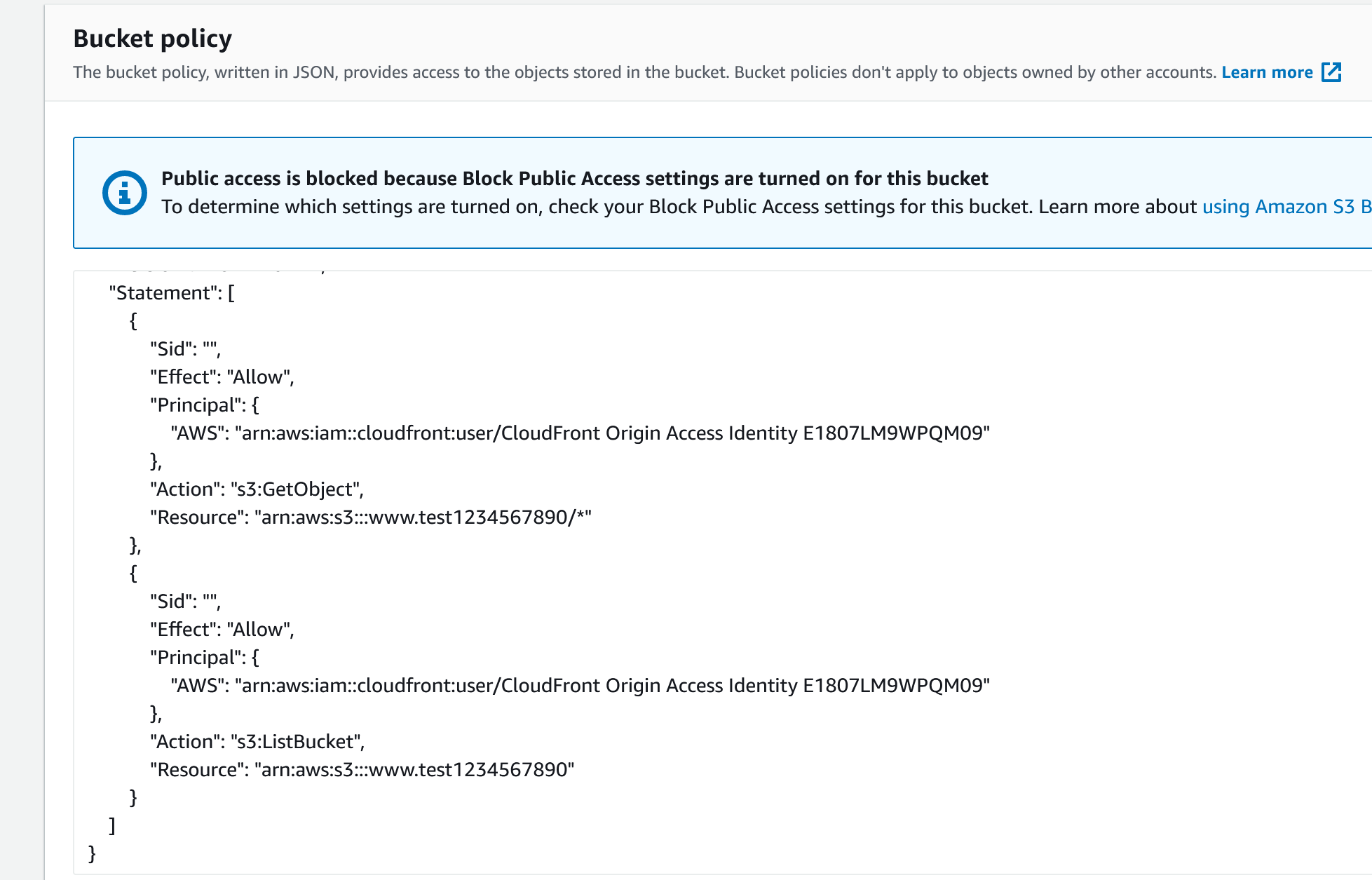
* Create Custom SSL certificate for cloudfront (ACM in us-east-1 region for cloudfront) to protect your data in transit
* Cloudfront is protected by WAF, it has one Web ACL name my-web-acl that rate limit. We can use the ab (apache benchmark) tool to simulate huge traffic from one single IP to verify WAF will block the repeated traffic.from command line output, we find failed request is 48877, same with AWS WAF result.



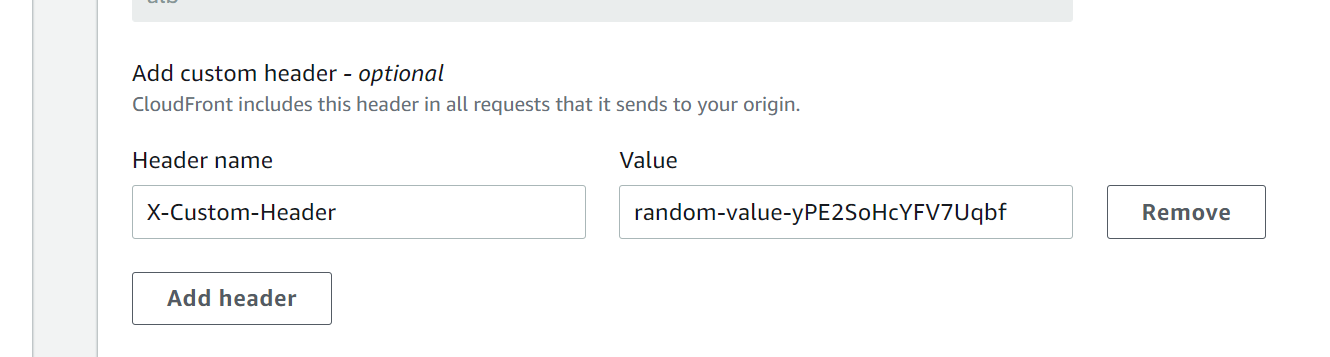


* S3 static contents use OAI ,bucket can restrict access to only CloudFront, no public access and bucket policy also updated to allow cloudfront access.

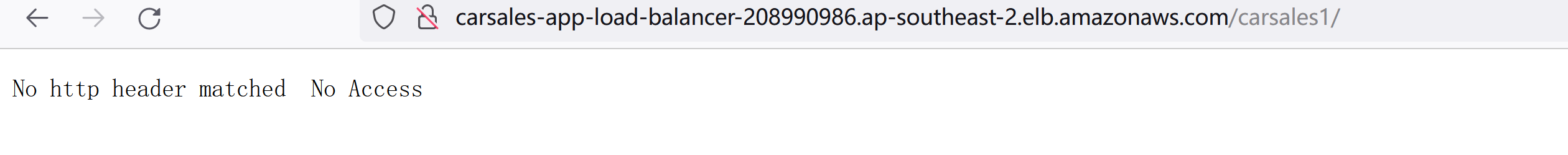




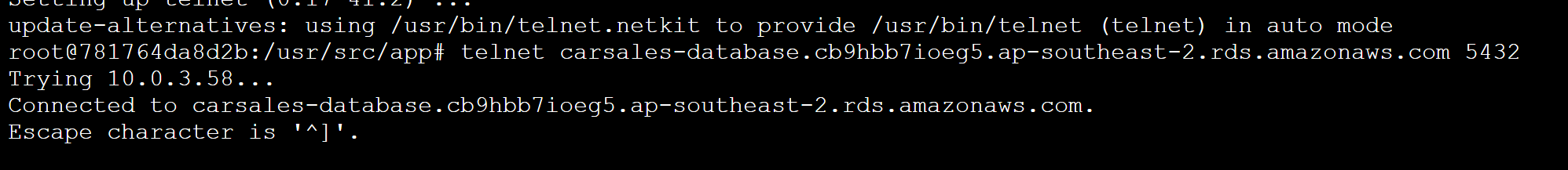
* ALB and Cloudfront use custom http header to communicate,if http header not matched, ALB refused to forward traffic. The header value stored in SSM (/production/myapp/random-httpheader)

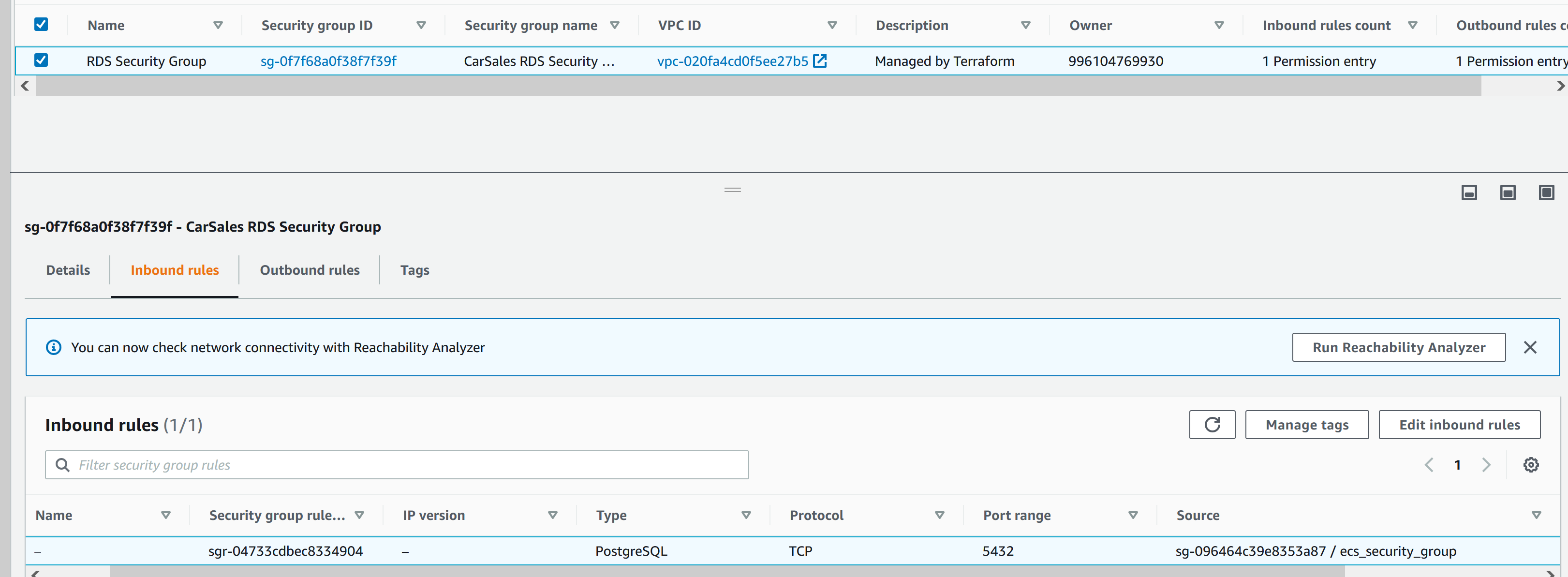






* For secure reason RDS Security group only has one allowed source SG, can only be accessed from ECS EC2 instances.



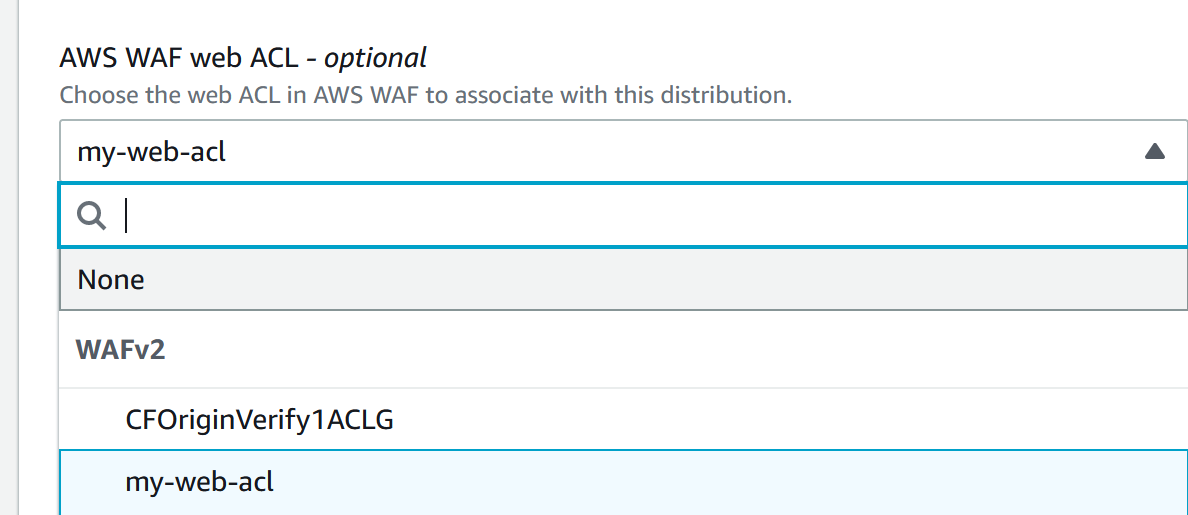


Test ECS service autoscale based on load

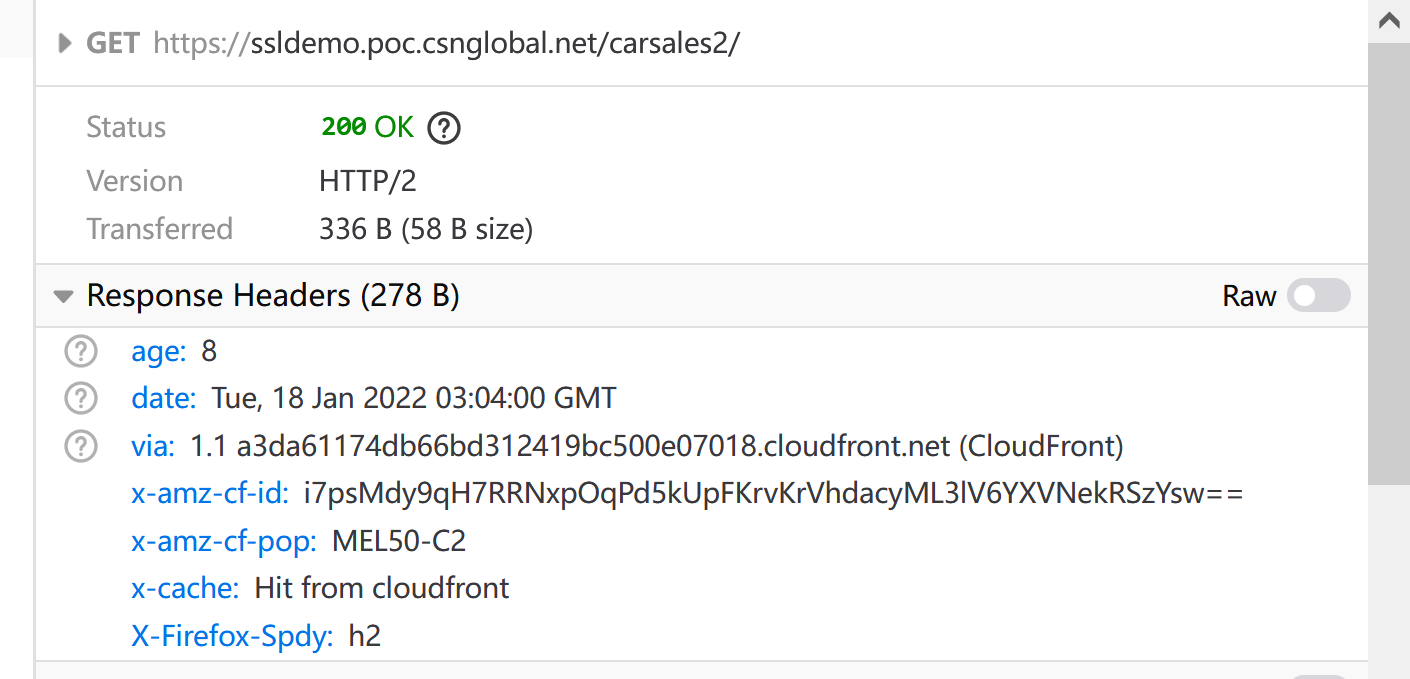
Use blazemeter image to simulate load on target docker.

There are some prerequisite for this test

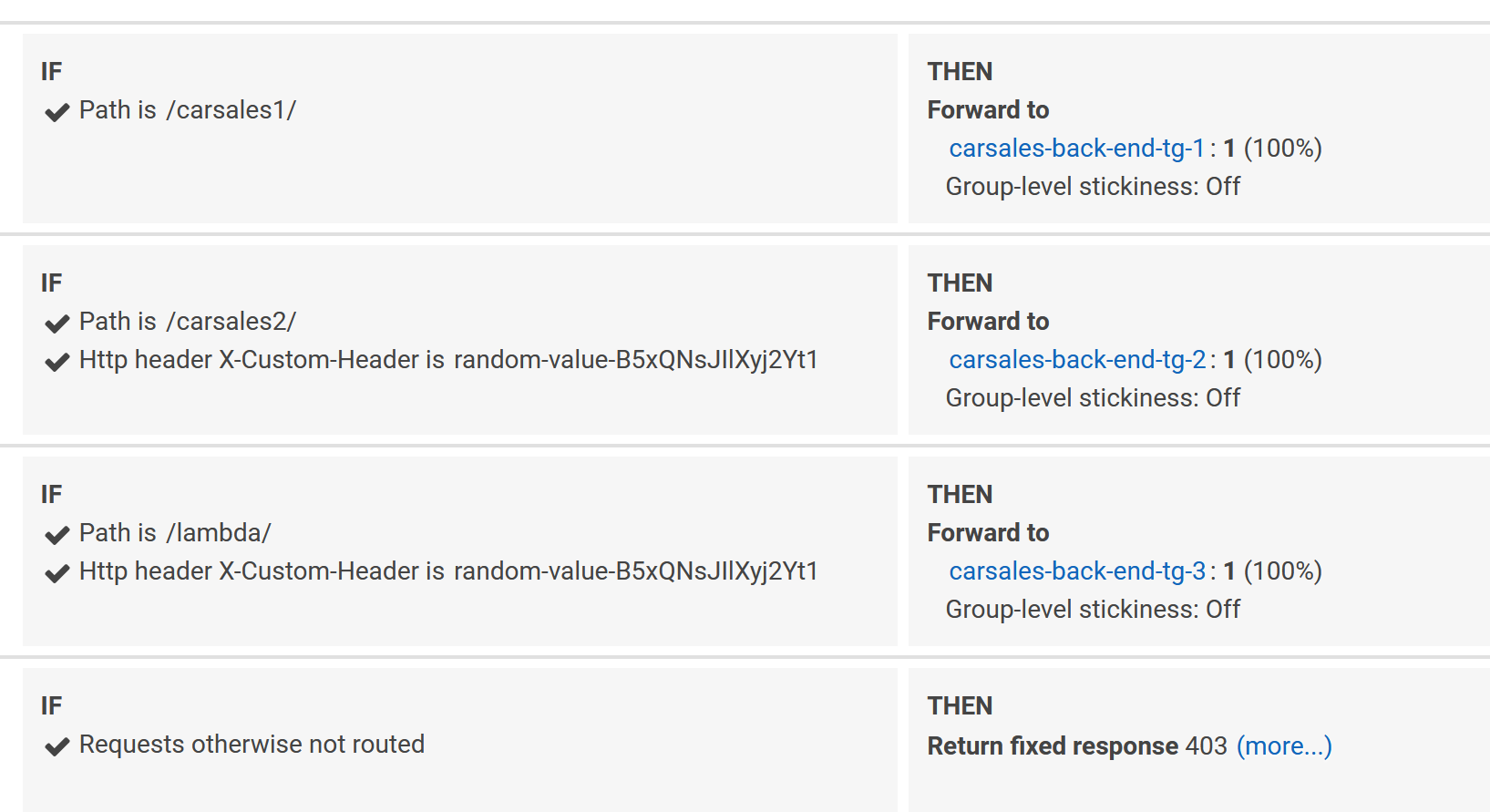
* Disable AWS WAF web ACL for cloudfront ,change from my-web-acl to none or it will block duplicate traffic from one ip when reach the limit.



* Access ALB target directly, or it will cached by cloudfront instead of reach target ECS container.

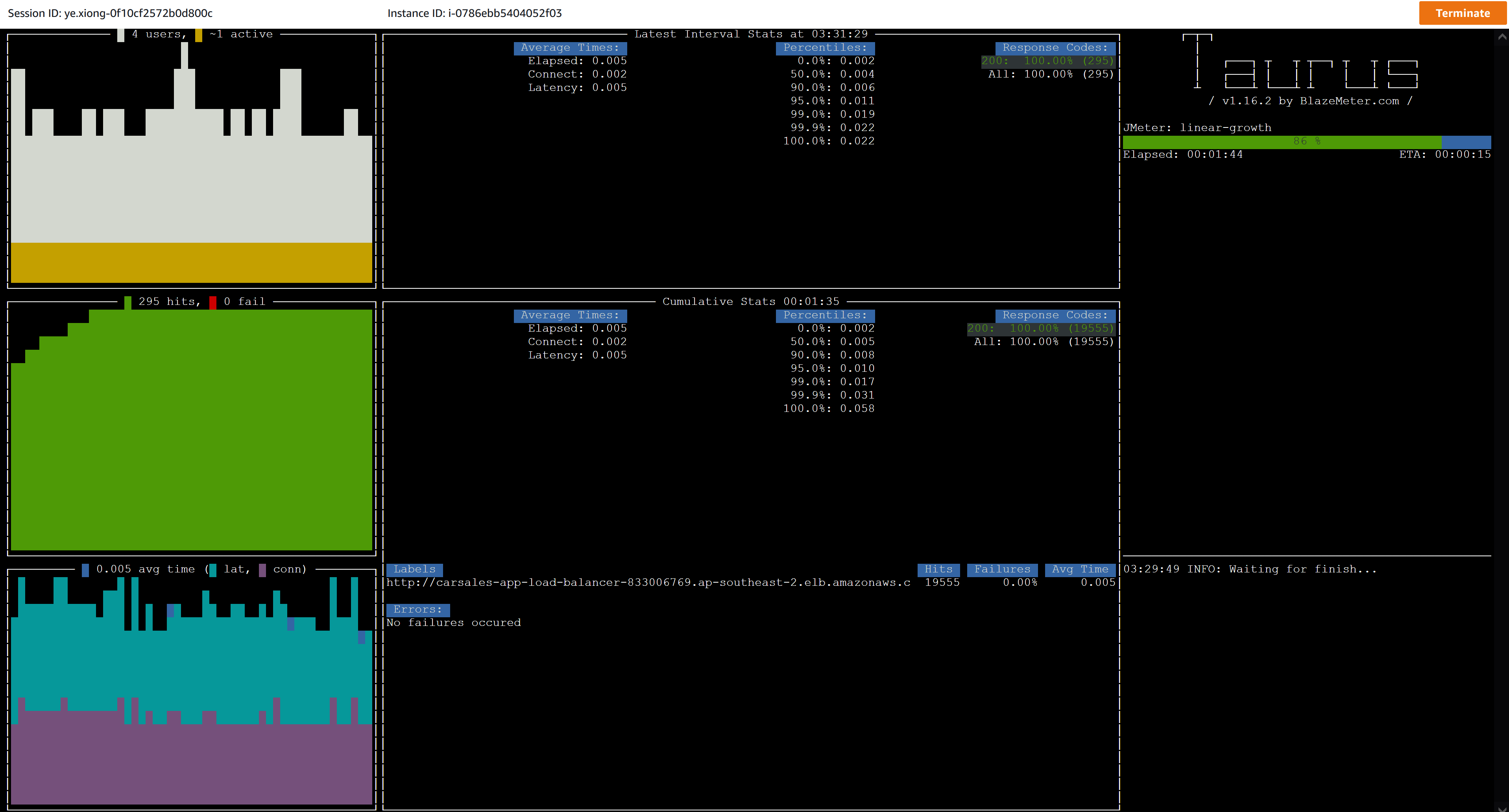


* Disable http header check or it will response 403 from load balancer instead of reach target container.

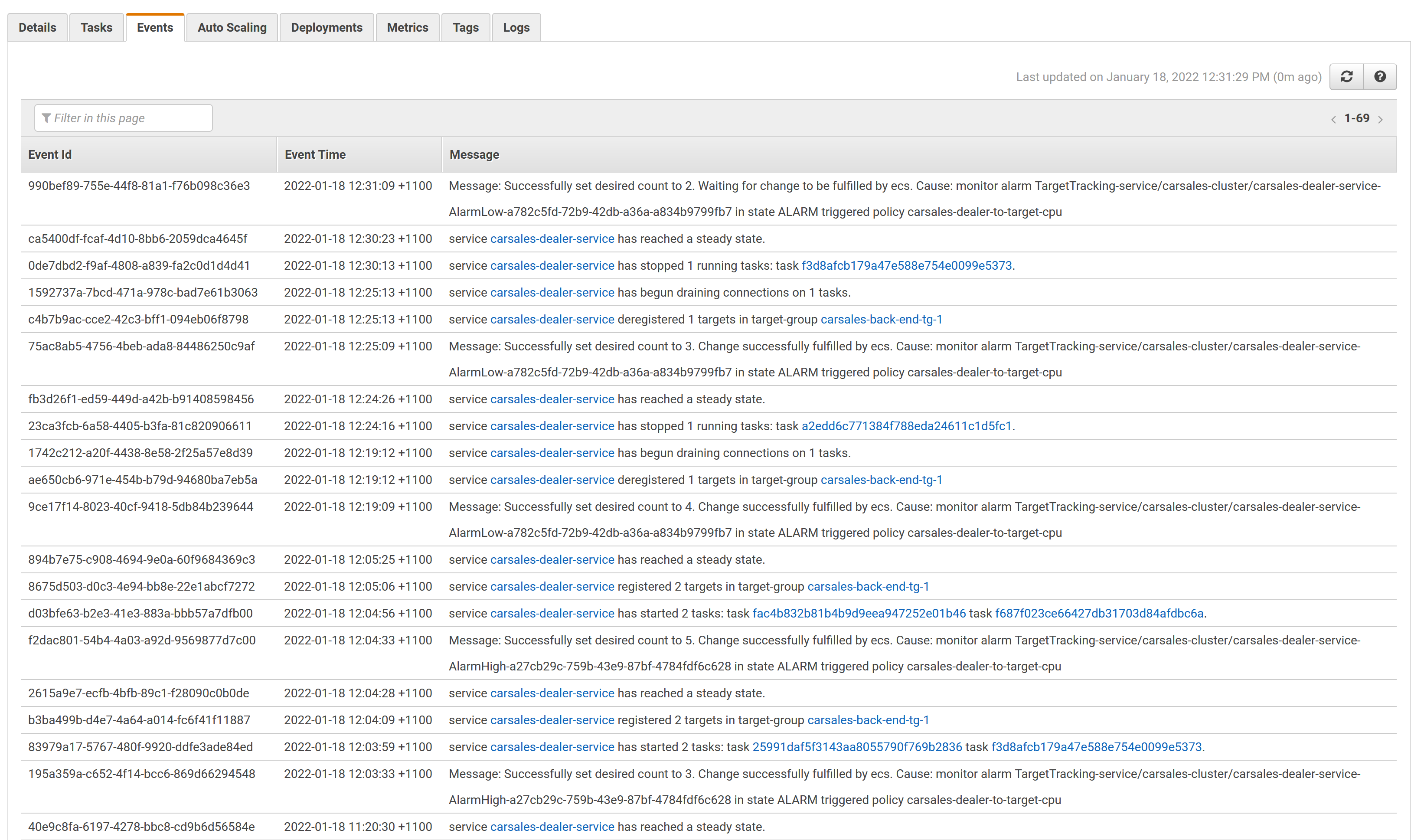


Run following command after previous prerequisite finished ,change target alb dns name accordingly based on target environment.

sudo docker run -it dbaxy770928/blazemeter <http://carsales-app-load-balancer-833006769.ap-southeast-2.elb.amazonaws.com/carsales1/>

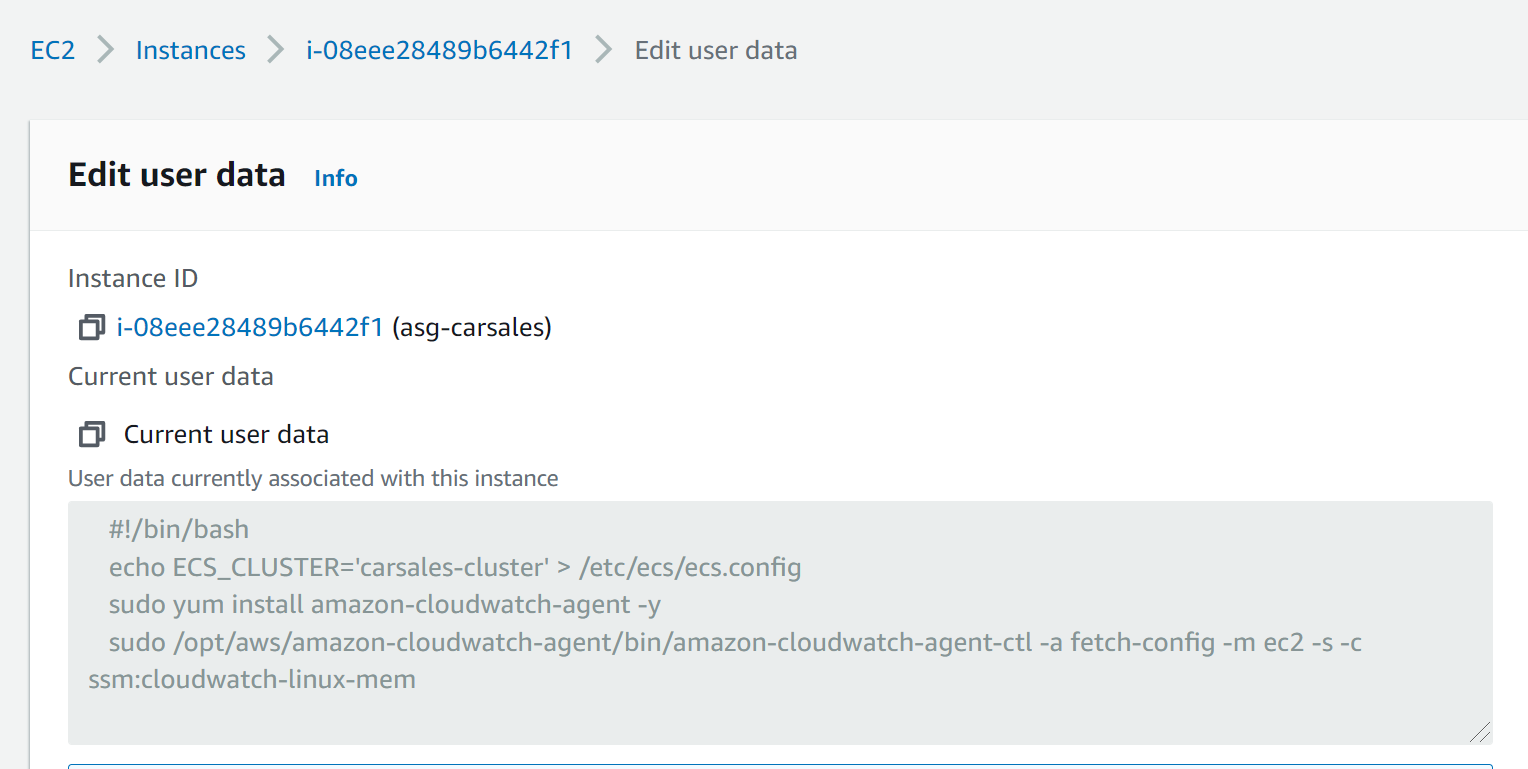


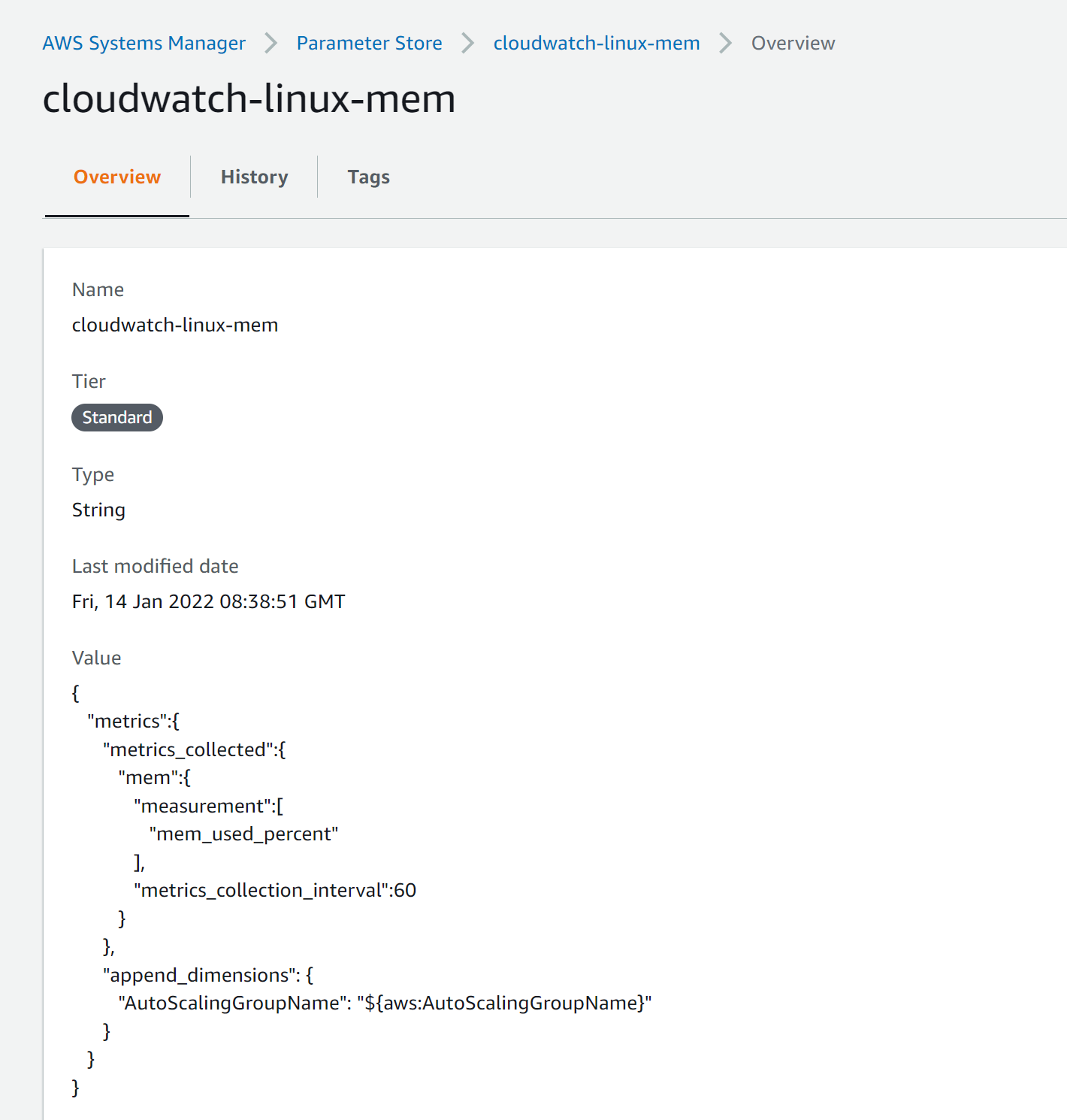
From ECS service events tab, we can find new ECS tasks added and removed based on actual workloads.



EC2 instances which compose ECS clusters auto scale based on workload

* Use user\_data to Get SSM parameter for cloudwatch log agent to report EC2 memory usage





* Use sudo docker run -ti --rm dbaxy770928/stress stress --cpu 2 --io 2 --vm 2 --vm-bytes 512M --timeout 100s --verbose to simulate pressure and trigger cloudwatch alert to ASG add new EC2 instance

