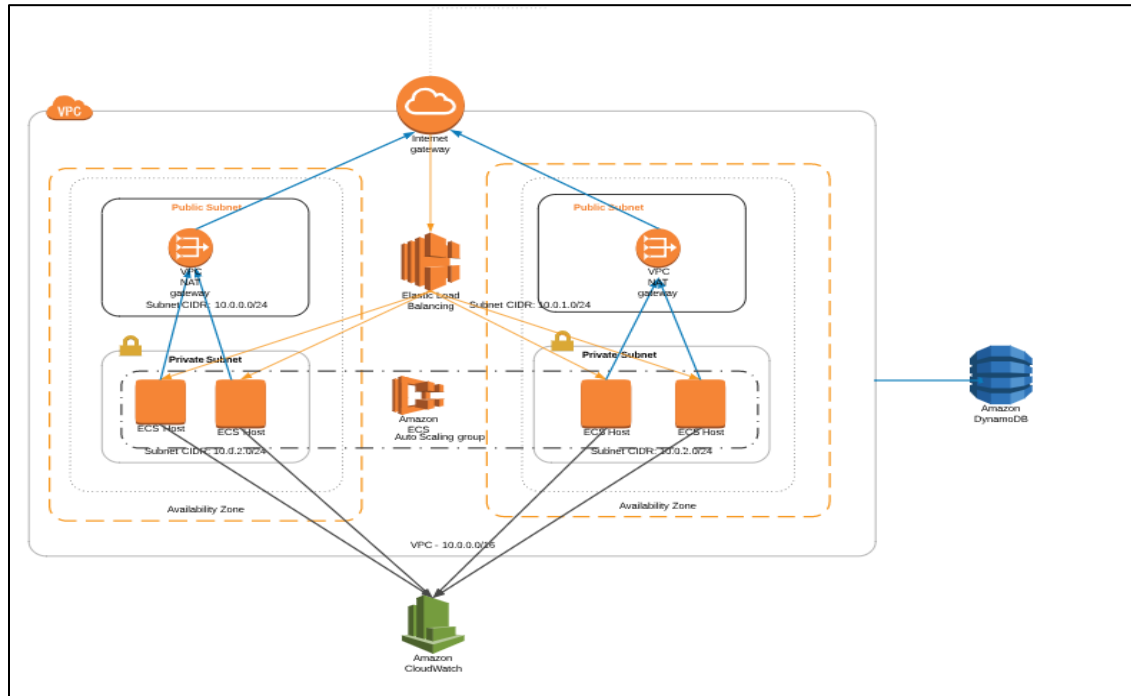


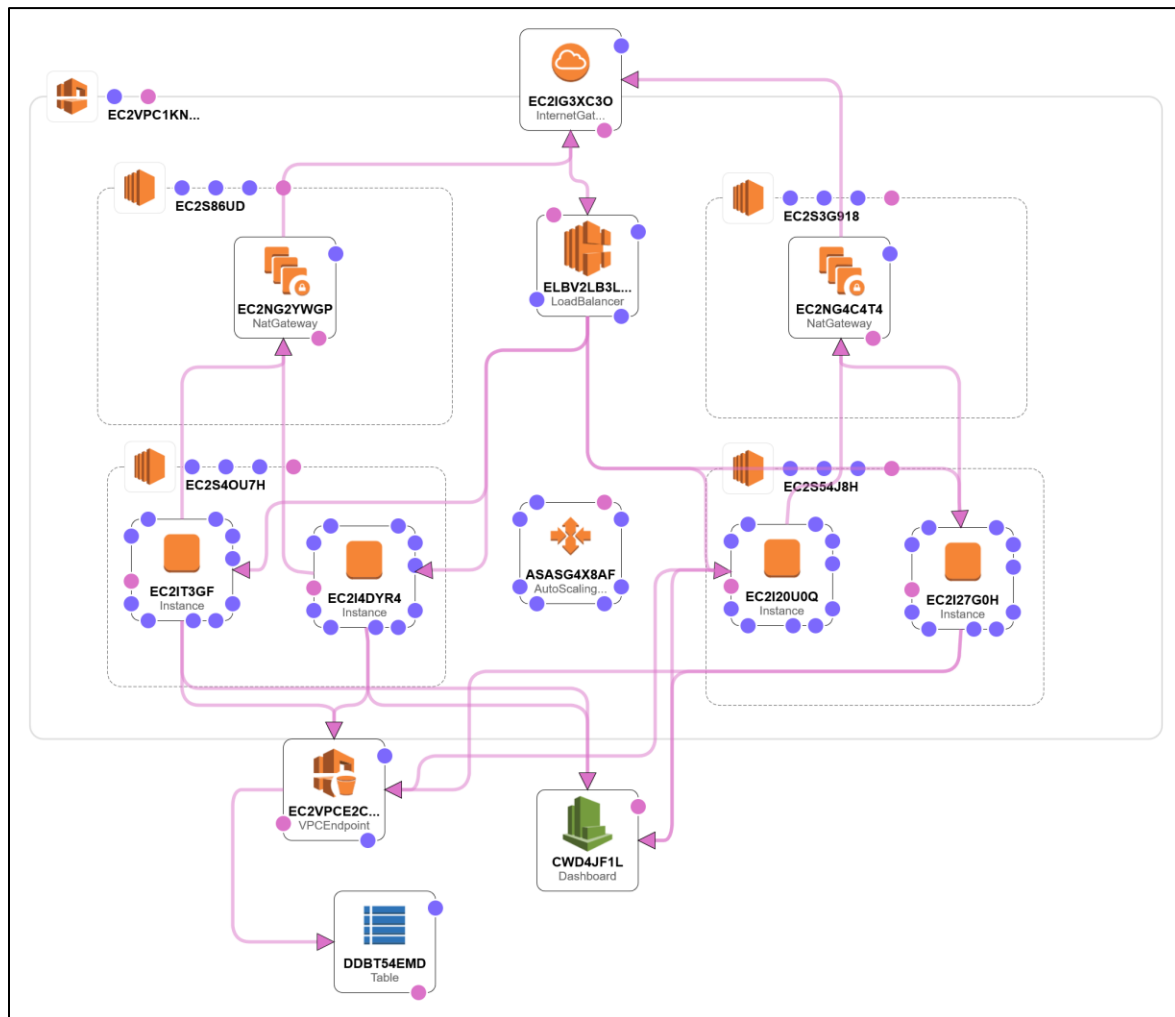
1.

Cloud formation design

Below diagram shows, the cloud formation design implemented using AWS cloud formation



Below shows the developed deployment diagram using AWS cloud formation designer



b. Component use in cloud formation designing.

Microservices, also known as microservice architectures are independently deployable and scalable. From a microservice, it can highly expect the resiliency, scalability and observability.

Resiliency: The ability of continues functioning of an application without occurring any failures is called resiliency. That mean even if the failures happen there should be fully functioning condition following failures. High availability and disaster recovery are main aspect in resiliency.

Scalability: Is the ability to change in volume according to the user requests. It can be scale-out or scale-in , in order to utilize the resources as per network traffic.

Observability : is the monitoring with time about the condition of microservice and their component performances and failures.

1. VPC (Virtual private cloud)

The created VPC of microservice is consist with two availability zones, two public and two private subnets.

- Availability zones.

Two availability zones have used in this cloud formation. If one availability zone goes down, then other availability zone will handle the requests. So application availability increase by using two availability zones. By using two availability zones, user can access to the application without any failures if one zone is down. Load balancer can direct the traffic to the available zones only without sending them to the failure zones. So this help to increase the resiliency also.

- Public and Private Subnet

There are two public subnets that routs through internet gateway and two private subnets that routs through Nat Gateways.

2. Internet gate way

Internet gate way allows the communication between instances in the VPC and the internet .Internet gateway provides target to VPC rout tables for internet routable traffic and serves network address translation(NAT) to instances.

3.NAT gateways

NAT gateways enables the instances in private subnets to the internets. Meanwhile NAT gateways can perform to prevent internet from initiating connection with instances.

4. Application load balancers

Application load balancer consist with three main component, Load balancers, Listeners and Target groups.

- Load Balancers

Application load balancer is a type of an elastic load balancer that act as a single point of contact for clients. This will distribute the incoming application traffic across EC2 instances in multiple availability zones.

- Listeners

Listener is a component of application load balancers .Listener will check the request from clients as per given protocol and ports and based on the define rules those request will forward to the targets groups.

- Target groups.

Target group is a component of application load balancer, this will routes request to registered target groups such as EC2 instances.

Application load balancer perform distributing the network traffic among the ECS host. Whenever instance scaling up load balancer will identify and direct the request to the new instance. Instance failure situation, load balancer will terminate the sending request to that particular instance. It helps to load SSL traffic and defend DDoS attacks Therefore, application load balancer perform to increase the availability and resiliency of application.

5. Auto-scaling group.








This will perform to create new EC2 instance (Scale up) and ECS task when in high network traffic. When traffic goes down again this will scale-In (reduce) the instances. So this will increase a high availability of the application

6. Cloud Watch




Cloud watch perform to monitor AWS cloud resources and the application running on it. This will collect track metrics, log files and set alarms. Cloud watch will provide a high observability of microservice resources and it can provide insights for running better application with high reaction and smoothly running.

2. Steps to run the AWS cloud formation script.

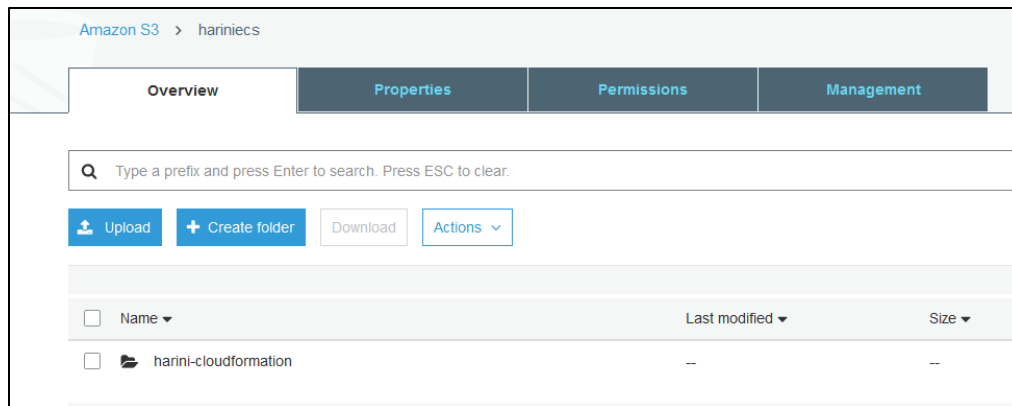
AWS cloud formation script include following yaml files for creating microservice infrastructure.

 access.yaml	12/31/2018 11:24 ...	YAML File	1 KB
 dynamo-db.yaml	1/2/2019 8:15 AM	YAML File	3 KB
 ecs-cluster.yaml	1/1/2019 8:57 PM	YAML File	12 KB
 lifecyclehook.yaml	11/24/2018 3:10 AM	YAML File	6 KB
 load-balancers.yaml	11/24/2018 3:10 AM	YAML File	2 KB
 security-groups.yaml	11/24/2018 3:10 AM	YAML File	3 KB
 vpc.yaml	1/1/2019 9:16 PM	YAML File	7 KB

Masternew.yaml file will create the e above component accordingly and provides the required resources link for creating each stack.

 infrastructure	1/1/2019 8:41 PM	File folder	
 services	1/1/2019 8:12 PM	File folder	
 masternew.yaml	1/2/2019 10:17 AM	YAML File	5 KB

1. Upload the cloud formation file to S3 bucket



2. Creating a new stack

Under cloud formation, create new stack by providing Amazon S3 URL that contain **masternew.yaml**

Select Template

Select the template that describes the stack that you want to create. A stack is a group of related resources that you manage as a single unit.

Design a template Use AWS CloudFormation Designer to create or modify an existing template. [Learn more.](#)

Choose a template A template is a JSON/YAML-formatted text file that describes your stack's resources and their properties. [Learn more.](#)

☐ Select a sample template

☐ Upload a template to Amazon S3

No file selected.

☒ Specify an Amazon S3 template URL

Then create the stack with `maternew.yaml` and this will initiate to create other stacks as below,

Create Stack Actions Design template

Filter: Active By Stack Name

Showing 10 stacks

Stack Name	Created Time	Status	Drift Status	Description
<input type="checkbox"/> harini1-FlowersService-5515K... NESTED	2019-01-02 19:03:40 UTC+0550	CREATE_COMPLETE	NOT_CHECKED	This is an example of a long running E...
<input type="checkbox"/> harini1-AddService-117PZDC2... NESTED	2019-01-02 19:03:39 UTC+0550	CREATE_COMPLETE	NOT_CHECKED	This is an example of a long running E...
<input type="checkbox"/> harini1-LifecycleHook-9DM2X... NESTED	2019-01-02 19:03:39 UTC+0550	CREATE_COMPLETE	NOT_CHECKED	This template deploys a Lambda Func...
<input type="checkbox"/> harini1-ALB-WVKHO2NUGHLF NESTED	2019-01-02 18:59:32 UTC+0550	CREATE_COMPLETE	NOT_CHECKED	This template deploys an Application ...
<input type="checkbox"/> harini1-ECS-1DMV3NE32NHU... NESTED	2019-01-02 18:59:32 UTC+0550	CREATE_COMPLETE	NOT_CHECKED	This template deploys an ECS cluster ...

Overview Outputs Resources Events Template Parameters Tags Stack Policy Change Sets Rollback Triggers

Filter by: Status Search events

7 more events available to display

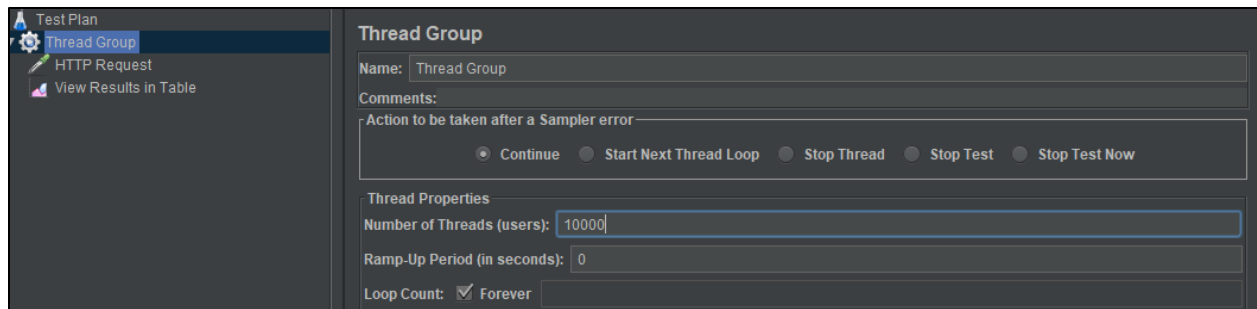
2019-01-02	Status	Type	Logical ID	Status Reason
19:01:48 UTC+0550	CREATE_COMPLETE	AWS::CloudFormation::Stack	ALB	
18:59:32 UTC+0550	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	ALB	Resource creation Initiated
18:59:32 UTC+0550	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	ECS	Resource creation Initiated
18:59:31 UTC+0550	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	ALB	

Above figure shows, the creating stacks using `masternew.yaml`

3. Apache Jmeter to test the environment.

To test the environment use the Apache Jmeter with following parameters,

- Tread Group

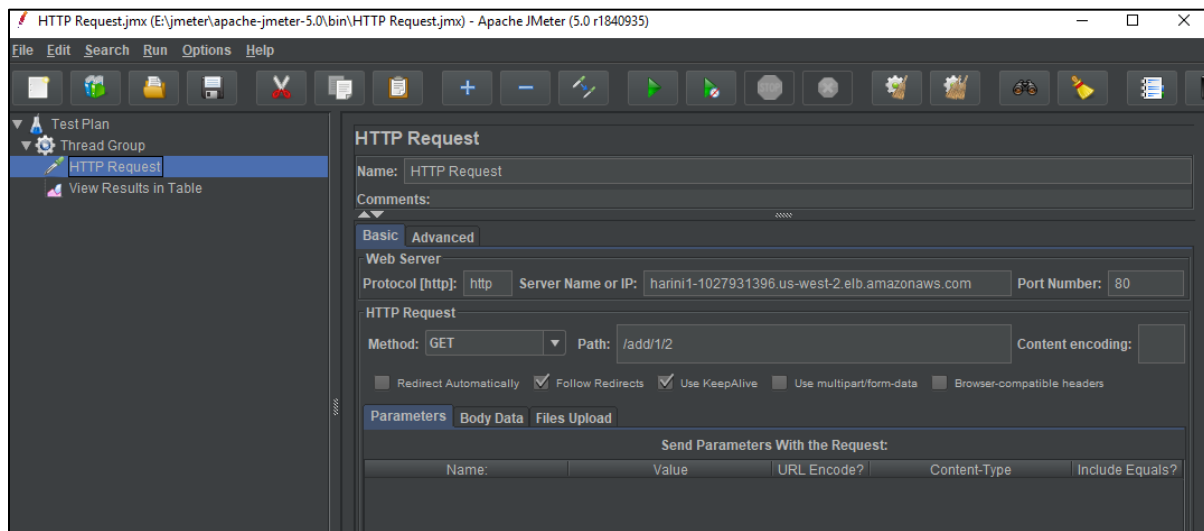


- HTTP Request.

Protocol: http

Name or IP : harini1-1027931396.us-west-2.elb.amazonaws.com

Path : /add/1/2



After setup the parameters s above run the Jmeter to send the request to the application. Below figure shows the sending request to the application.

HTTP Request.jmx (E:\jmeter\apache-jmeter-5.0\bin\HTTP Request.jmx) - Apache JMeter (5.0 r1840935)

File Edit Search Run Options Help

Test Plan
Thread Group
HTTP Request
View Results in Table

Filename: Browse... Log/Display Only: Errors Successes

Sample #	Start Time	Thread Name	Label	Sample Time	Status	Bytes	Sent Bytes	Latency
1	19:19:16.861	Thread Grou...	HTTP Request	832	✓	233	153	83
2	19:19:16.863	Thread Grou...	HTTP Request	849	✓	233	153	84
3	19:19:16.861	Thread Grou...	HTTP Request	863	✓	233	153	86
4	19:19:16.935	Thread Grou...	HTTP Request	789	✓	233	153	78
5	19:19:16.859	Thread Grou...	HTTP Request	865	✓	233	153	86
6	19:19:16.937	Thread Grou...	HTTP Request	787	✓	233	153	78
7	19:19:16.862	Thread Grou...	HTTP Request	862	✓	233	153	86
8	19:19:16.935	Thread Grou...	HTTP Request	789	✓	233	153	78
9	19:19:16.938	Thread Grou...	HTTP Request	786	✓	233	153	78
10	19:19:16.937	Thread Grou...	HTTP Request	787	✓	233	153	78
11	19:19:16.934	Thread Grou...	HTTP Request	790	✓	233	153	79
12	19:19:16.940	Thread Grou...	HTTP Request	784	✓	233	153	78
13	19:19:16.939	Thread Grou...	HTTP Request	785	✓	233	153	78
14	19:19:16.862	Thread Grou...	HTTP Request	862	✓	233	153	86
15	19:19:16.939	Thread Grou...	HTTP Request	785	✓	233	153	78
16	19:19:16.940	Thread Grou...	HTTP Request	802	✓	233	153	80
17	19:19:16.936	Thread Grou...	HTTP Request	806	✓	233	153	80
18	19:19:16.940	Thread Grou...	HTTP Request	802	✓	233	153	80
19	19:19:16.937	Thread Grou...	HTTP Request	806	✓	233	153	80
20	19:19:16.938	Thread Grou...	HTTP Request	805	✓	233	153	80
21	19:19:16.937	Thread Grou...	HTTP Request	806	✓	233	153	80
22	19:19:16.941	Thread Grou...	HTTP Request	802	✓	233	153	80
23	19:19:16.938	Thread Grou...	HTTP Request	805	✓	233	153	80
24	19:19:16.940	Thread Grou...	HTTP Request	813	✓	233	153	81
25	19:19:16.939	Thread Grou...	HTTP Request	814	✓	233	153	81

Scroll automatically? Child samples? No of Samples: 14260 Latest Sample: 18643 Average: 20.101

4. Interpreting the test results

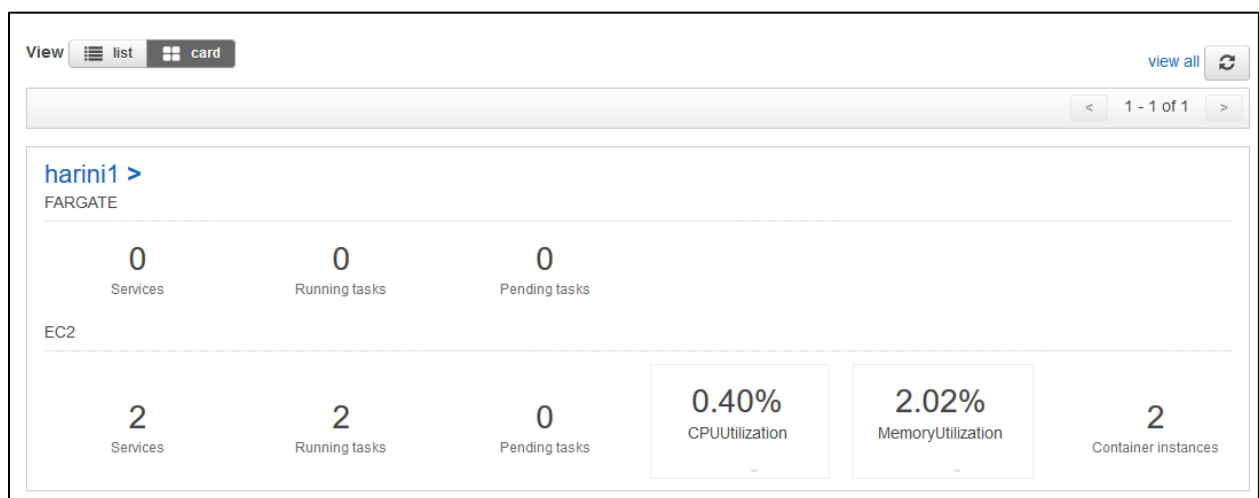
- Before sending the request to the application there are two instances running ,

<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
<input type="checkbox"/>	harini1 ECS ...	i-0a6a8cd8ccce61aef	t2.micro	us-west-2b	running	✓ 2/2 checks ...	None	
<input type="checkbox"/>	harini1 ECS ...	i-0ca9064cf0c78efdb	t2.micro	us-west-2a	running	✓ 2/2 checks ...	None	

After sending, the requests there were three instance running.

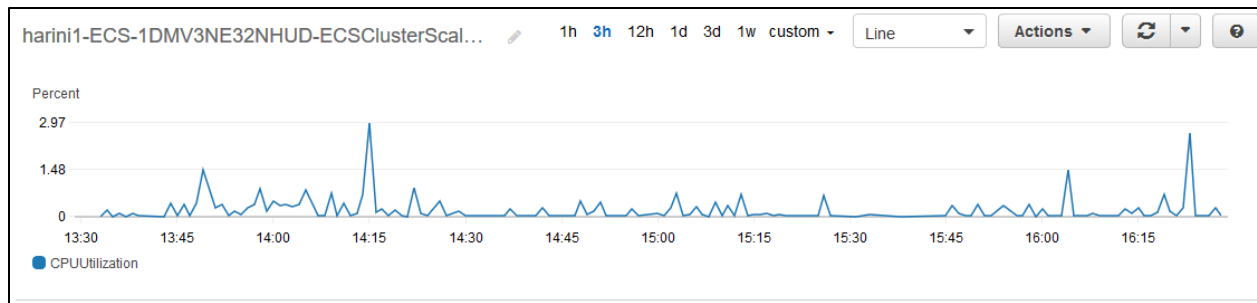
<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IP
<input type="checkbox"/>	harini1 ECS ...	i-0101f23968c0af70b	t2.micro	us-west-2b	terminated		None		-
<input type="checkbox"/>	harini1 ECS ...	i-05a71709f097deb2e	t2.micro	us-west-2a	running	2/2 checks ...	None		-
<input type="checkbox"/>	dev ECS host	i-07e753d75a64e70d7	t2.micro	us-west-2a	terminated		None		-
<input type="checkbox"/>	harini1 ECS ...	i-0a6a8cd8ccce61aef	t2.micro	us-west-2b	running	2/2 checks ...	None		-
<input type="checkbox"/>	harini1 ECS ...	i-0ca9064cf0c78efdb	t2.micro	us-west-2a	running	2/2 checks ...	None		-
<input type="checkbox"/>	dev ECS host	i-0f7276e552ab8d694	t2.micro	us-west-2b	terminated		None		-

From the jmeter test, it can be seen that when the traffic become high scale-up the instances by creating new instance. And ECS increase its task to perform in high traffic. After terminating the request from jmeter with time again instance become to scale-in by removing extra instance.

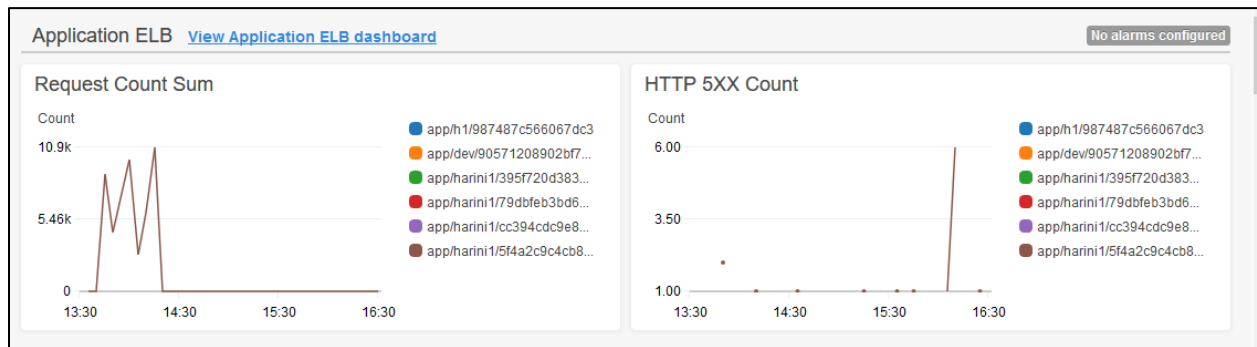


5. AWS Cloud watch graphs

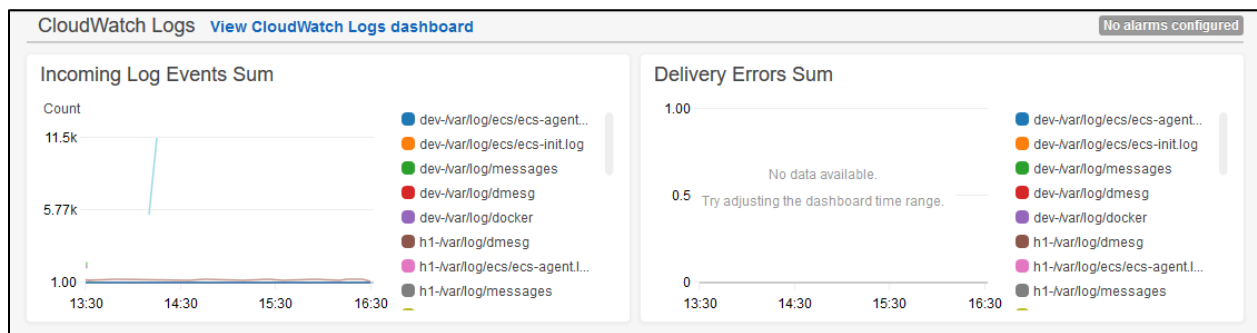
Below dashboards show the performance of each component during the application run time.



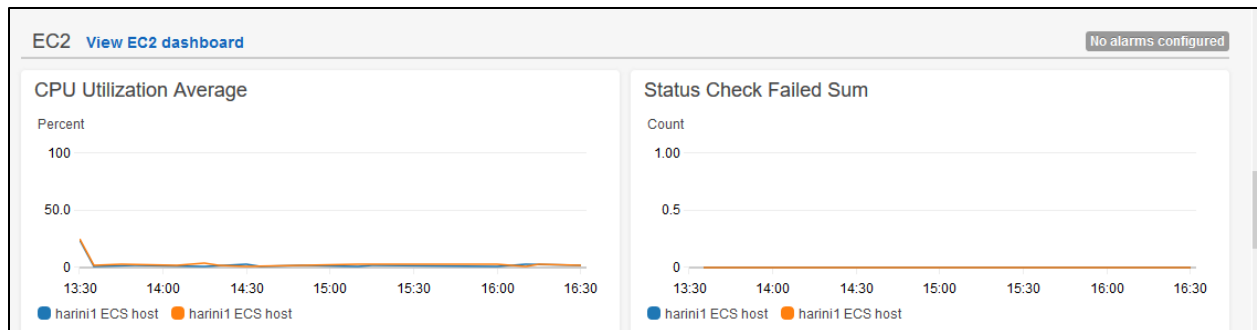
CPU utilization



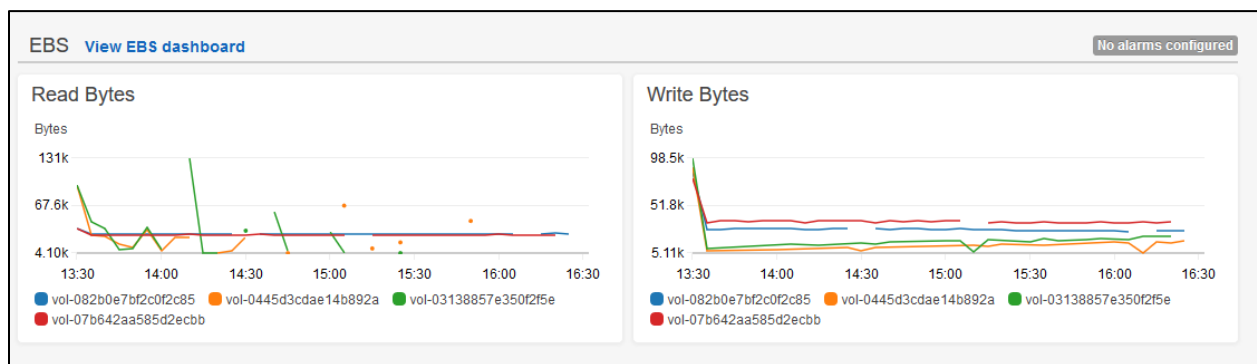
ELB dashboard



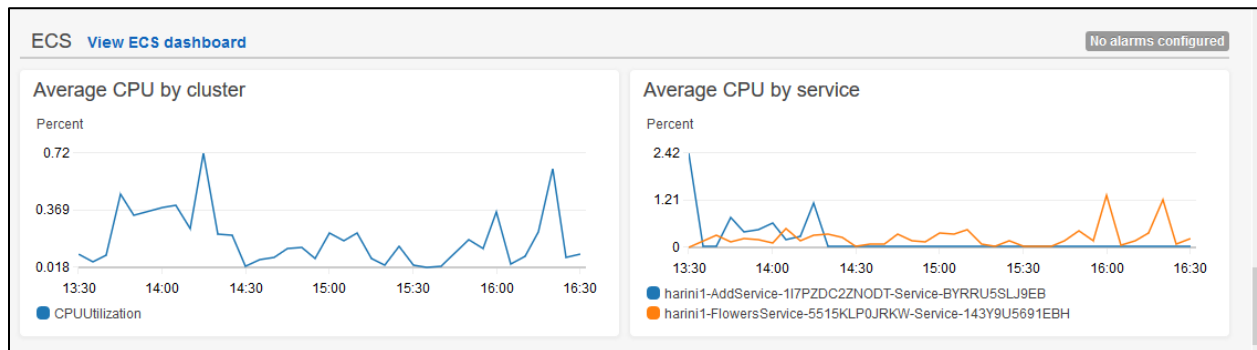
Cloud watch log dashboards



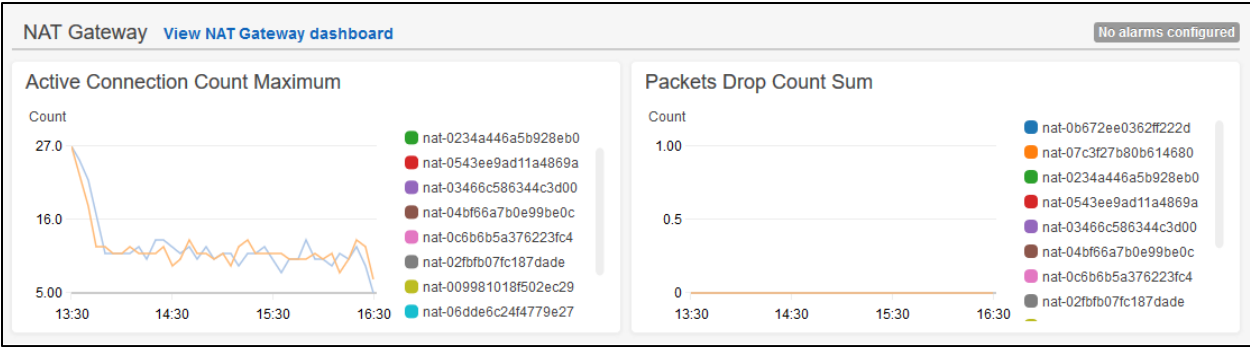
EC2 dashboard



EBS dashboard



ECS dashboard



NAT gateway