DOCTOR’S OFFICE APPOINTMENT APPLICATION

ENPM818R

Group 2 Midterm Report

Table of Contents

1. Executive Summary… 3
2. Architecture Diagram… 4
3. Docker Image Creation…..9
3.1 Command-line output of docker-build showing successful image creation… 9
3.2 Docker Desktop Dashboard… 9
3.3 Verification of Docker image running locally (docker run output)...................................9
4. EKS Cluster Setup… 10
4.1 EKS on AWS Console…...10
4.2 IAM roles and permissions set up for EKS and worker nodes… 10
4.3 Terminal output showing kubectl connected to the EKS Cluster…...................................11
5. Kubernetes Deployment… 12
5.1 Command-line output of kubectl get for each component…..12
5.2 kubectl get services output, showing the LoadBalancer and other service details… 12
6. Domain Setup and SSL Configuration…...14
6.1 LoadBalancer Configuration… 14
6.2 Route 53 Hosted Zone Configuration for Domain Mapping… 14
6.3 AWS Certificate Manager for SSL Configuration… 15
6.4 Domain Access Verification… 15
7. Monitoring… 17
7.1 CloudWatch Metrics…...17
7.1.1 Dashboard for CPU Utilization Metrics of Jenkins Web Server….............................17
7.1.2 Dashboard for Network Traffic of the Jenkins Web Server… 17
7.1.3 Dashboard for CPU Utilization Across All Worker Nodes…......................................18
7.1.4 Dashboard for Network Traffic Across All Worker Nodes… 18
7.2 CloudWatch Alarms… 19
7.2.1 Alarm for High Outgoing Network Traffic for MongoDB… 19
7.2.2 Alarm for Outgoing Traffic on EC2 Instances in the Backend…...............................20
7.2.3 Alarm for Frontend MongoDB Input Network Traffic…... 20
7.2.4 Alarm for Backend Input Network Traffic… 21
7.2.5 Alarm for monitoring an EFS (Amazon Elastic File System) storage…....................22
7 2.6 Alarm for monitoring CPU utilization for a MongoDB instance running on EC2…23
7.2.7 Alarm for CPU utilization for an EC2 instance for a Worker Node… 24
7.2.8 Alarm for monitoring the health of an EKS cluster's worker nodes… 25
7.2.9 Alarm for CapacityProviderReservation for an EKS cluster frontend infrastructure.27
7.2.10 Alarm for monitoring the lower threshold of Capacity Provider Reservation…......28
7.2.11 Alarm for monitoring the EKS cluster's capacity provider reservation… 28
7.3 AWS Simple Notification Service (SNS) Subscription Configuration…...........................29
8. CI/CD Pipeline…...31
8.1 AWS Set up… 31

8.2 Jenkins Set up…..31
8.3 Pipeline Creation…...33
8.4 Automating with GitHub Webhooks… 36
9. Testing and Debugging…..**38**
9.1 Frontend UI Testing… 38
9.2 Backend Testing - Appointments Fetch… 38
9.3 Backend-Testing - Appointment Creation…..39
10. Conclusion…...**41**
11. References… **42**

1. Executive Summary

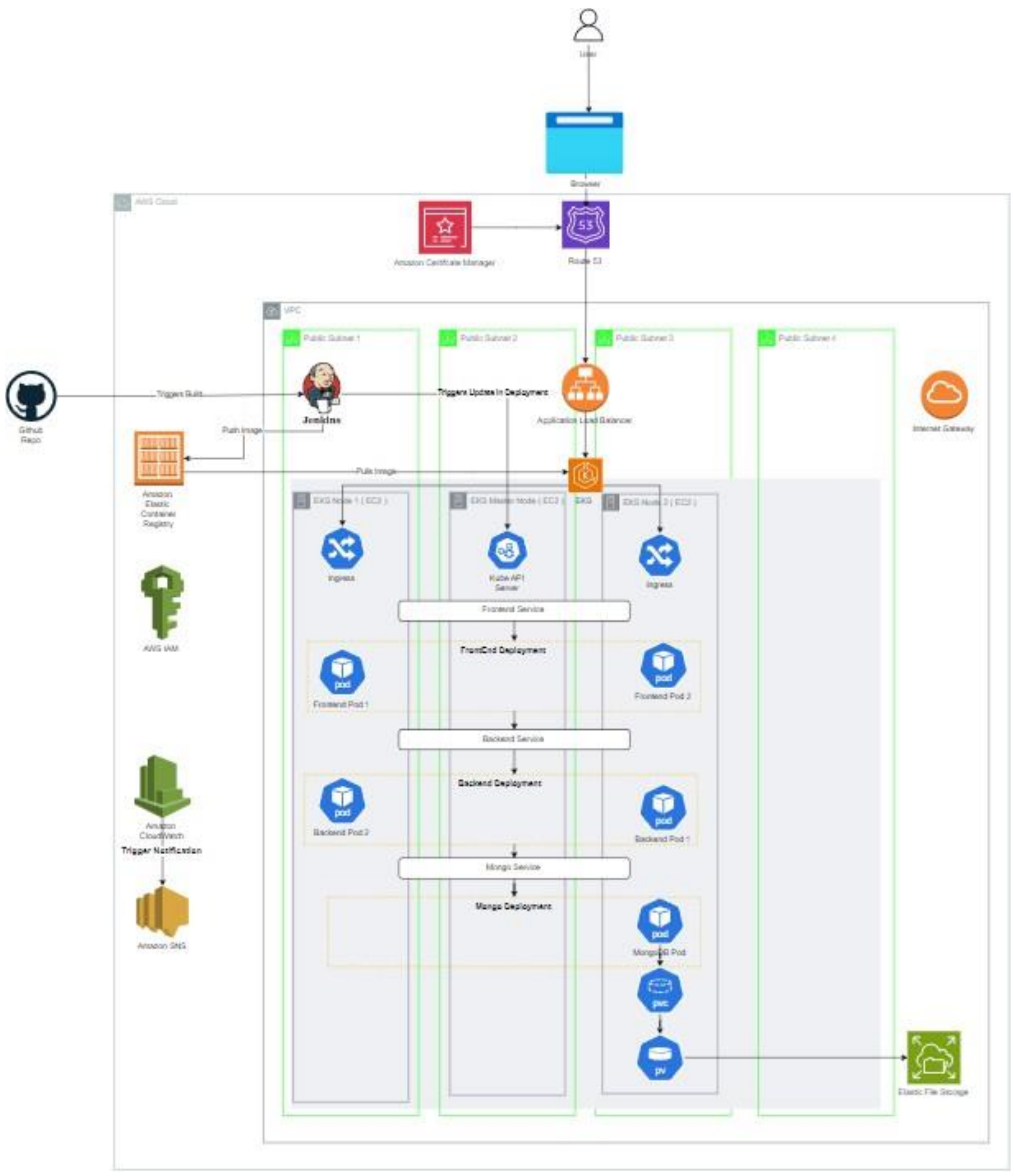
The Doctor's Office Appointment Application is a cloud-native, microservices-based platform developed to streamline the scheduling and management of appointments in a healthcare setting. Utilizing Amazon Web Services (AWS) infrastructure, this solution offers a reliable, scalable, and secure environment that efficiently handles increasing user demands and complex operations involved in managing appointments. Key technologies include a React-based front-end for an intuitive user experience, a Node.js and Express back-end API for handling application logic, and MongoDB as the primary data store for patient and appointment data.

To ensure a high degree of scalability, each application component is containerized with Docker and orchestrated on Amazon Elastic Kubernetes Service (EKS). This approach supports the seamless scaling of individual services based on demand, alongside robust fault tolerance. AWS services, including Elastic Container Registry (ECR) for Docker image management and Elastic Load Balancing (ELB) for traffic distribution, were selected to enable efficient deployment and operational reliability. Additionally, AWS CloudWatch and Kubernetes auto-scaling mechanisms monitor system performance and automatically adjust resources based on real-time metrics.

Security and data privacy are paramount in this project. AWS Certificate Manager (ACM) is used to secure communication through HTTPS, while sensitive information is safeguarded using AWS Secrets Manager and IAM role-based access. Continuous integration and continuous deployment (CI/CD) practices are implemented via Jenkins, ensuring rapid, consistent updates and minimized downtime during deployment cycles.

This project exemplifies best practices in cloud-native application development, emphasizing performance optimization, security, and user accessibility. The final deliverables include a fully operational, containerized application deployed on AWS, along with comprehensive documentation, version-controlled source code, and a presentation showcasing the application's architecture and functionality.

2. Architecture Diagram



Here’s the link to the Architecture Diagram shown above: [Group 2 Midterm Architecture Diagram]

This architecture diagram illustrates a cloud-based deployment for a Doctor's Office Appointment App using AWS services, Kubernetes, Docker containers, and CI/CD pipelines. Below is an explanation of every component and its role, interactions, and importance in the system.

- **GitHub Repository:**
  - The GitHub repository is where the application's code is stored and version-controlled.
  - Developers push code changes here, which then trigger actions in the CI/CD pipeline (Jenkins).
- **CI/CD Pipeline with Jenkins**
  - This is the CI/CD tool that monitors the GitHub repository for changes. When changes are detected, it pulls the updated code and begins a build process.
  - Jenkins builds the application and creates Docker images of each microservice.
  - After building the images, Jenkins pushes them to Amazon Elastic Container Registry (ECR), a secure, fully managed Docker container registry.
- **Amazon Elastic Container Registry (ECR)**
  - ECR is a container registry that stores Docker images created by Jenkins.
  - These images are then used by Kubernetes for deploying the application across the cluster.
- **Amazon Elastic Kubernetes Service (EKS) Cluster**
  - EKS hosts the Kubernetes cluster, which includes several nodes (EC2 instances) in different public subnets within a VPC (Virtual Private Cloud).
  - The Kubernetes cluster runs the containerized applications, split into frontend, backend, and MongoDB services.
  - Each component (frontend, backend, MongoDB) is deployed across multiple pods for high availability and scalability.
- **Public Subnets (1-4)**
  - The VPC contains four public subnets. These subnets host EC2 instances, which are worker nodes in the EKS cluster.
  - The Kubernetes Ingress Controller and Application Load Balancer (ALB) are also deployed here.
- **Kubernetes Ingress Controller and Application Load Balancer (ALB)**
  - Ingress Controller manages incoming HTTP/HTTPS traffic and routes it to the appropriate services in the cluster.
  - Application Load Balancer (ALB) distributes traffic across multiple pods for each service (frontend and backend) to ensure load balancing and fault tolerance.
- **Frontend and Backend Services**
  - These services represent the two main parts of the application:
  - The frontend service hosts the user interface and interacts with users.

- The backend service processes requests from the frontend, performs business logic, and communicates with the MongoDB database.
- Each service is deployed across multiple pods, providing scalability and redundancy. If a pod fails, Kubernetes will replace it automatically.

- **MongoDB Service**
  - MongoDB is used as the database for this application and is deployed as a StatefulSet in Kubernetes, ensuring that data remains consistent and persistent.
  - Persistent Volume (PV) is allocated to MongoDB for data storage, so even if pods are recreated, the data remains intact.
- **Amazon Route 53**
  - Route 53 is AWS's DNS service and is used to route users' requests to the correct resources.
  - It directs traffic to the ALB, which then routes the traffic to the Ingress controller within the EKS cluster.
- **Amazon Certificate Manager (ACM)**
  - ACM is responsible for managing SSL/TLS certificates, which are used to encrypt traffic between the user and the ALB.
  - This ensures secure communication by enabling HTTPS.
- **Elastic File System (EFS)**
  - Amazon EFS provides shared, persistent storage that can be mounted across multiple EC2 instances or pods.
  - MongoDB uses EFS for persistent data storage, ensuring high availability and data consistency even if individual instances or pods are terminated.
- **IAM (Identity and Access Management)**
  - IAM manages permissions and access control, ensuring that each service and component has only the necessary permissions.
  - It's used to secure interactions between AWS services and components within this architecture.
- **Internet Gateway**
  - The Internet Gateway allows communication between resources inside the VPC and the internet.
  - It provides outbound access for EKS nodes and allows users to access the application from the internet.

Interaction Flow and Importance

- **User Accesses the Application**
  - The user interacts with the application via a web browser and enters the application's domain name, for example, g2today.co.

    - Amazon Route 53 handles DNS resolution and routes the user’s request to the Application Load Balancer (ALB). Route 53 is configured to resolve the domain name to the ALB's endpoint, which sits in front of the Kubernetes cluster.

- **Application Load Balancer (ALB)**
    - The ALB receives the user’s HTTP/HTTPS request. This ALB is configured to accept requests and route them to the appropriate resources in the EKS (Elastic Kubernetes Service) cluster.
    - If HTTPS is used, AWS Certificate Manager (ACM) handles SSL/TLS termination at the ALB level, ensuring secure and encrypted communication between the user’s browser and the application.
    - The ALB directs the request to the Kubernetes Ingress Controller based on predefined routing rules.
- **Kubernetes Ingress Controller**
    - The Ingress Controller manages traffic within the EKS cluster and routes incoming requests to the appropriate services based on the request path and other rules.
    - The Ingress Controller acts as the "traffic director" within the EKS cluster, ensuring that user requests reach the correct pods.
- **Frontend Service (UI)**
    - When the user accesses the main interface of the application, the Ingress Controller forwards the request to the Frontend Service, which is responsible for rendering the user interface.
    - The Frontend Service runs in multiple pods (containers) for scalability and resilience. It serves static assets like HTML, CSS, JavaScript, and other UI elements to the user’s browser, providing them with the visual interface of the application.
    - The Frontend Service communicates with the Backend Service for dynamic content or data required for the UI.
- **Backend Service (API)**
    - When the user performs actions that require data processing (e.g., searching for appointments, submitting forms, etc.), the Frontend Service makes API calls to the Backend Service.
    - The Backend Service is also deployed in multiple pods and is responsible for handling business logic, processing requests, and interacting with the database (MongoDB).
    - The Backend Service interprets API requests, processes any business logic, and if needed, communicates with MongoDB to retrieve or update data.
- **Database Interaction - MongoDB Service**
    - For requests that require data storage or retrieval, the Backend Service communicates with MongoDB, which is running in a Kubernetes StatefulSet.

- MongoDB holds the application's structured data, like user profiles, appointment records, etc.
- MongoDB is set up with Persistent Volumes (PV), typically backed by Amazon EFS (Elastic File System), ensuring data persistence and availability even if MongoDB pods are restarted or replaced.
- After querying or updating data, MongoDB sends the response back to the Backend Service.

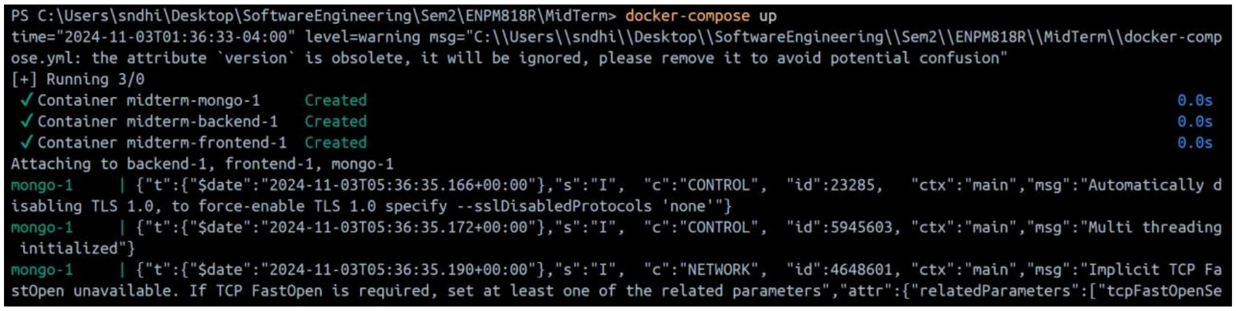
- **Backend Response to Frontend**
  - The Backend Service processes the data received from MongoDB, formats it as needed, and sends a response back to the Frontend Service.
  - This data could include information such as a list of available appointments, user details, or any other relevant information requested by the frontend.
- **Frontend Service Renders the Response**
  - The Frontend Service receives the data from the Backend Service and updates the user interface accordingly.
  - This might involve displaying data in tables, charts, or forms based on the user's interaction with the application.
- **User Sees the Updated Interface**
  - The user's browser renders the updated UI based on the response from the Frontend Service.
  - Any further interactions (like clicking a button or submitting another request) would follow the same flow: routed through the ALB → Ingress Controller → appropriate service within the EKS cluster.

3. Docker Image Creation:

3.1 Command-line output of docker-build showing successful image creation

The **docker-compose up** command initializes and starts all the services which we have defined in the **docker-compose.yaml** file. Three containers are created and started here:

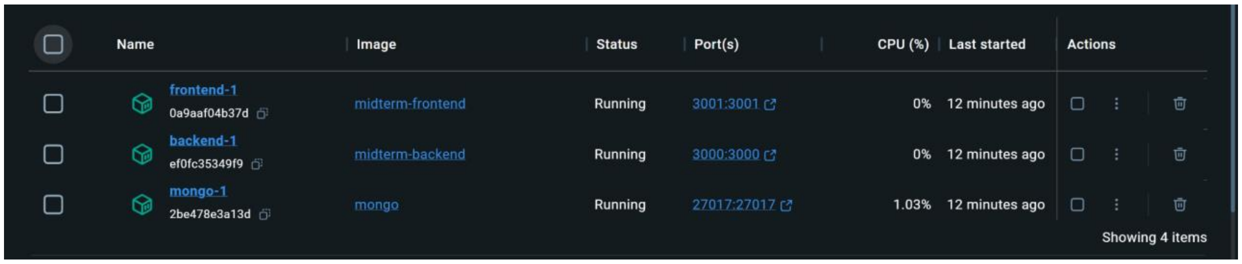
1. midterm-backend-1
2. midterm-frontend-1
3. midterm-mongo-1



```
PS C:\Users\sndhi\Desktop\SoftwareEngineering\Sem2\ENPM818R\MidTerm> docker-compose up
time="2024-11-03T01:36:33-04:00" level=warning msg="C:\\Users\\sndhi\\Desktop\\SoftwareEngineering\\Sem2\\ENPM818R\\MidTerm\\docker-comp
ose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 3/0
 ✓ Container midterm-mongo-1     Created                                                                                           0.0s
 ✓ Container midterm-backend-1   Created                                                                                           0.0s
 ✓ Container midterm-frontend-1  Created                                                                                           0.0s
Attaching to backend-1, frontend-1, mongo-1
mongo-1     | {"t":{"$date":"2024-11-03T05:36:35.166+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Automatically d
isabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
mongo-1     | {"t":{"$date":"2024-11-03T05:36:35.172+00:00"},"s":"I",  "c":"CONTROL",  "id":5945603, "ctx":"main","msg":"Multi threading
 initialized"}
mongo-1     | {"t":{"$date":"2024-11-03T05:36:35.190+00:00"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"main","msg":"Implicit TCP Fa
stOpen unavailable. If TCP FastOpen is required, set at least one of the related parameters","attr":{"relatedParameters":["tcpFastOpenSe
```

3.2 Docker Desktop Dashboard

The below screenshot shows a visual summary of the running containers along with the port mapping:



Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
frontend-1 0a9aaf04b37d	midterm-frontend	Running	3001:3001	0%	12 minutes ago	
backend-1 ef0fc35349f9	midterm-backend	Running	3000:3000	0%	12 minutes ago	
mongo-1 2be478e3a13d	mongo	Running	27017:27017	1.03%	12 minutes ago	

Showing 4 items

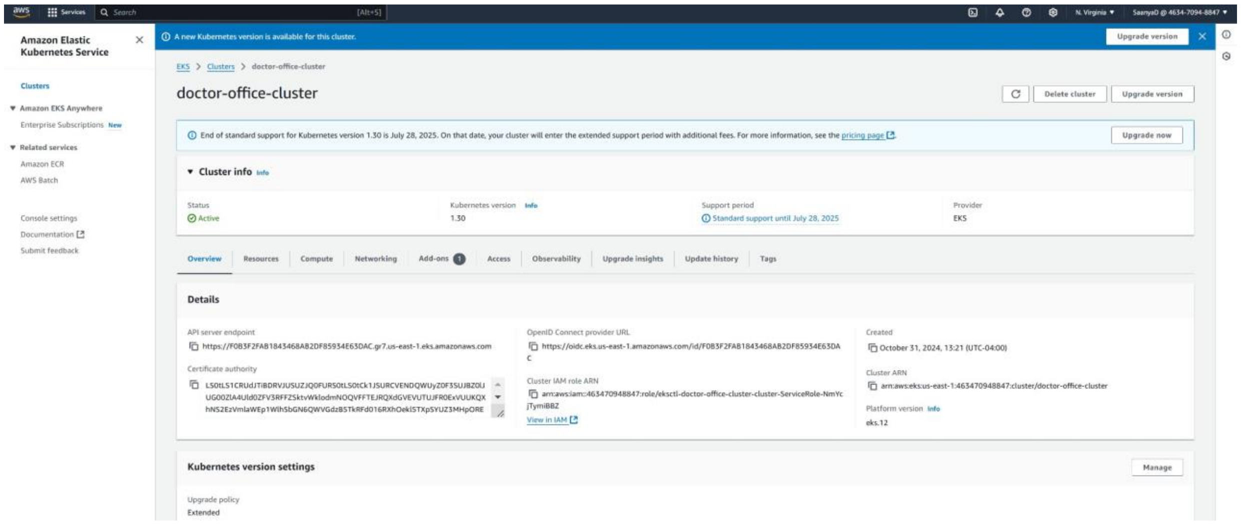
3.3 Verification of Docker image running locally (docker run output)

The below screenshot shows the user interface for the "Doctor's Office Appointments: application, which is accessible on localhost:3001. This can be accessed when docker-compose up command is successfully executed.

4. EKS Cluster Set up

4.1 EKS on AWS Console

The screen capture below displays the **Amazon Elastic Kubernetes Service (EKS)** console for a cluster we have created, **doctor-office-cluster**. Key information like API access, IAM roles, and other configuration settings can be found on this dashboard.


A new Kubernetes version is available for this cluster.

EKS > Clusters > doctor-office-cluster

doctor-office-cluster

End of standard support for Kubernetes version 1.30 is July 28, 2025. On that date, your cluster will enter the extended support period with additional fees. For more information, see the pricing page.

Cluster info

Status	Kubernetes version	Support period	Provider
Active	1.30	Standard support until July 28, 2025	EKS

Details

API server endpoint: https://F0B3F2FAB1843468AB2DF85934E63DAC.gr7.us-east-1.eks.amazonaws.com

Certificate authority:
```
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJ
UG00ZlA4Uld0ZFV3RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQX
hNS2EzVmlaWEp1WlhSbGN6QWVGdzB5TkRFd016RXhOekl5TXpSYUZ3MHpORE
```

OpenID Connect provider URL: https://oidc.eks.us-east-1.amazonaws.com/id/F0B3F2FAB1843468AB2DF85934E63DAC

Cluster IAM role ARN: arn:aws:iam::463470948847:role/eksctl-doctor-office-cluster-cluster-ServiceRole-NmYcjTymiBBZ

Created: October 31, 2024, 13:21 (UTC-04:00)

Cluster ARN: arn:aws:eks:us-east-1:463470948847:cluster/doctor-office-cluster

Platform version: eks.12

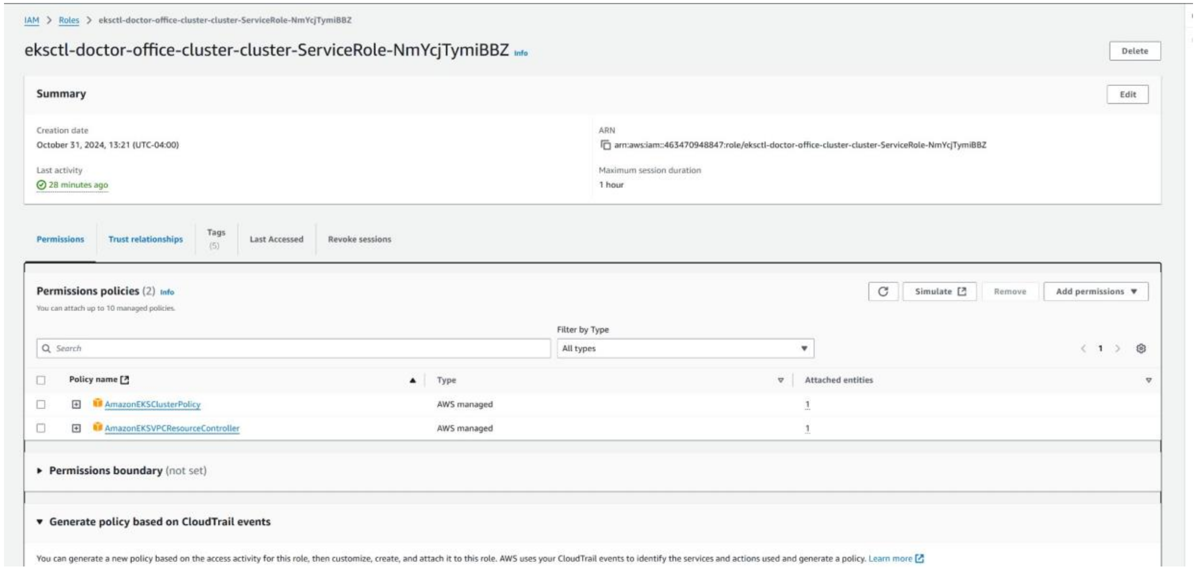
Kubernetes version settings

Upgrade policy: Extended

4.2 IAM roles and permissions set up for EKS and worker nodes

The IAM Role Details for the EKS cluster, doctor-office-cluster. 2 permissions have been attached with this IAM Role:

1. **AmazonEKSClusterAdminPolicy:** Grants administrator access to manage the EKS cluster.
2. **AmazonEKSAdminPolicy:** Provides broader EKS-related permissions such as managing and operating the EKS environment.


IAM > Roles > eksctl-doctor-office-cluster-cluster-ServiceRole-NmYcjTymiBBZ

eksctl-doctor-office-cluster-cluster-ServiceRole-NmYcjTymiBBZ

Summary

Creation date: October 31, 2024, 13:21 (UTC-04:00)

ARN: arn:aws:iam::463470948847:role/eksctl-doctor-office-cluster-cluster-ServiceRole-NmYcjTymiBBZ

Last activity: 28 minutes ago

Maximum session duration: 1 hour

Permissions policies (2)

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AmazonEKSClusterPolicy	AWS managed	1
AmazonEKSVPCResourceController	AWS managed	1

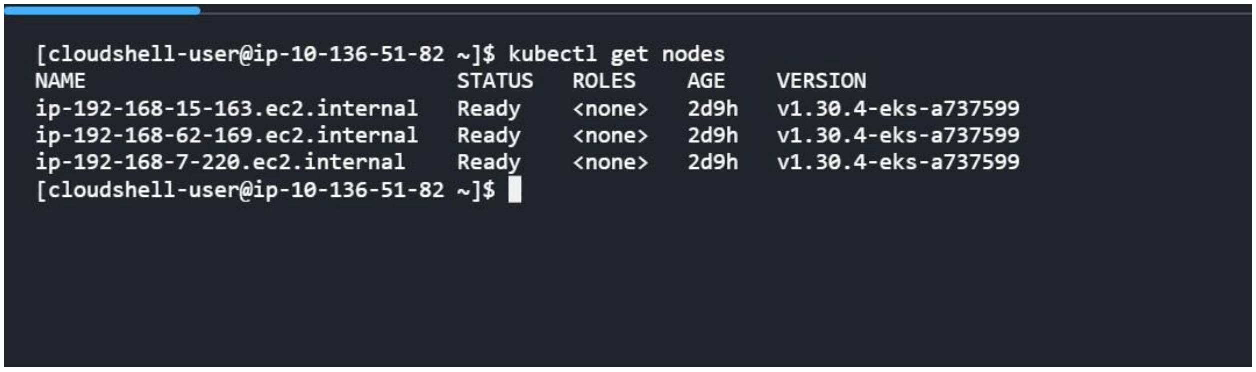
Permissions boundary (not set)

Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. Learn more

4.3 Terminal output showing kubectl connected to the EKS Cluster

kubectl get nodes was executed to retrieve the list of nodes in the connected EKS cluster. The status ready for each nodes indicates that all nodes are healthy and available to handle workloads.



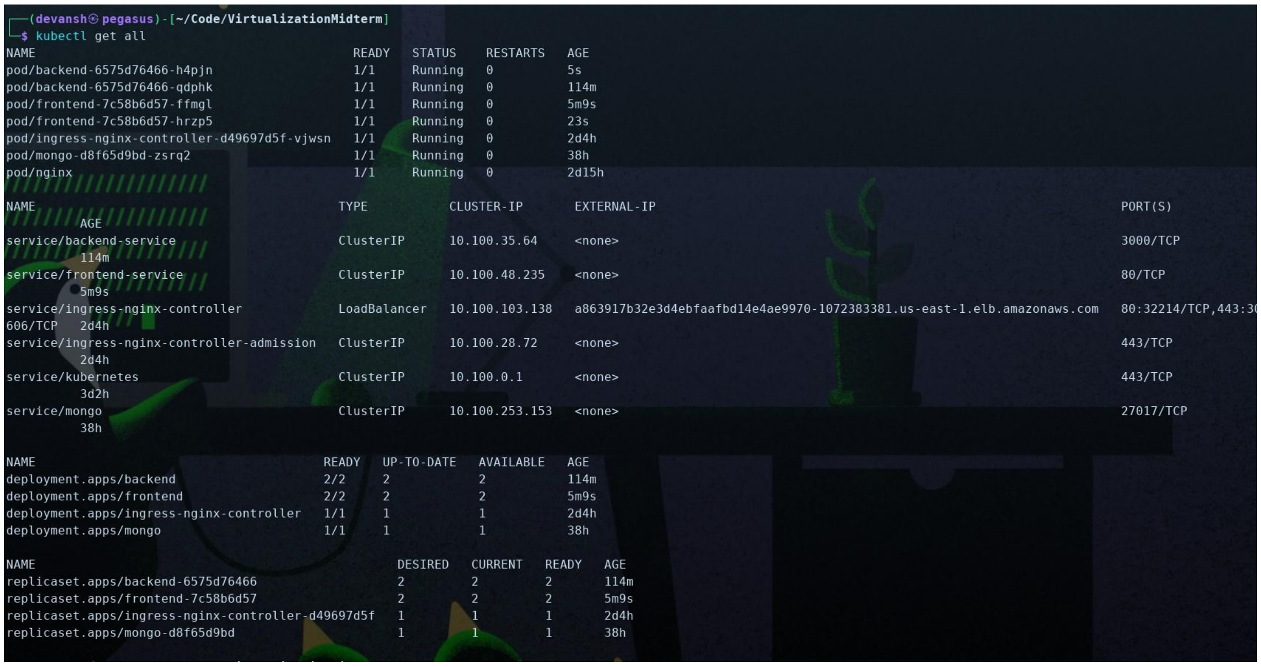
```
[cloudshell-user@ip-10-136-51-82 ~]$ kubectl get nodes
NAME                             STATUS   ROLES    AGE    VERSION
ip-192-168-15-163.ec2.internal   Ready    <none>   2d9h   v1.30.4-eks-a737599
ip-192-168-62-169.ec2.internal   Ready    <none>   2d9h   v1.30.4-eks-a737599
ip-192-168-7-220.ec2.internal    Ready    <none>   2d9h   v1.30.4-eks-a737599
[cloudshell-user@ip-10-136-51-82 ~]$ 
```

5. Kubernetes Deployment

5.1 Command-line output of kubectl get for each component, showing successful deployment

The below screenshot shows the result of **kubectl get all** command, which provides an overview of all Kubernetes resources running in the EKS cluster. Output of this command majorly shows the following resources:

1. **Pods**: Each application component is represented as a pod. This section shows individual pods running within the cluster, along with their status and uptime. Each pod's status is listed as Running, which means that they are operational.
2. **Services:** Lists all services, along with the type, external IP, ports, and age.
3. **Deployments:** This section displays deployment configurations, listing the number of replicas and their status. Here, we currently have 4 deployments, namely, backend, frontend, ingress-nginx-controller, and mongo, all of which are fully available with 1 replica of mongo, 2 of Frontend and Backend each
4. **ReplicaSets:** Shows the ReplicaSets created by each deployment to manage pod scaling. As mentioned above, all four deployments have one replica each.

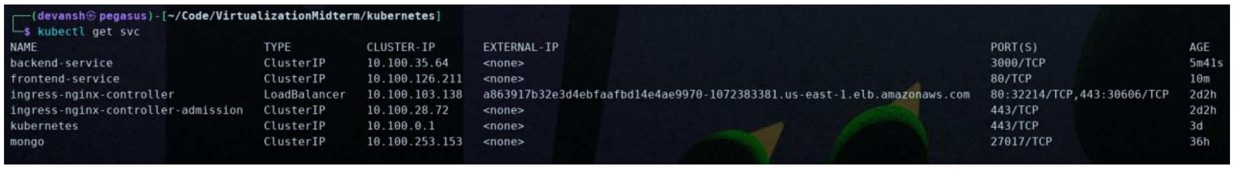


```
┌──(devansh㉿pegasus)-[~/Code/VirtualizationMidterm]
└─$ kubectl get all
NAME                                           READY   STATUS    RESTARTS   AGE
pod/backend-6575d76466-h4pjn                   1/1     Running   0          5s
pod/backend-6575d76466-qdphk                   1/1     Running   0          114m
pod/frontend-7c58b6d57-ffmgl                   1/1     Running   0          5m9s
pod/frontend-7c58b6d57-hrzp5                   1/1     Running   0          23s
pod/ingress-nginx-controller-d49697d5f-vjwsn   1/1     Running   0          2d4h
pod/mongo-d8f65d9bd-zsrq2                      1/1     Running   0          38h
pod/nginx                                      1/1     Running   0          2d15h

NAME                                         TYPE           CLUSTER-IP       EXTERNAL-IP                                                              PORT(S)
          AGE
service/backend-service                      ClusterIP      10.100.35.64     <none>                                                                   3000/TCP
          114m
service/frontend-service                     ClusterIP      10.100.48.235    <none>                                                                   80/TCP
          5m9s
service/ingress-nginx-controller             LoadBalancer   10.100.103.138   a863917b32e3d4ebfaafbd14e4ae9970-1072383381.us-east-1.elb.amazonaws.com   80:32214/TCP,443:3
606/TCP   2d4h
service/ingress-nginx-controller-admission   ClusterIP      10.100.28.72     <none>                                                                   443/TCP
          2d4h
service/kubernetes                           ClusterIP      10.100.0.1       <none>                                                                   443/TCP
          3d2h
service/mongo                                ClusterIP      10.100.253.153   <none>                                                                   27017/TCP
          38h

NAME                                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/backend                    2/2     2            2           114m
deployment.apps/frontend                   2/2     2            2           5m9s
deployment.apps/ingress-nginx-controller   1/1     1            1           2d4h
deployment.apps/mongo                      1/1     1            1           38h

NAME                                                 DESIRED   CURRENT   READY   AGE
replicaset.apps/backend-6575d76466                   2         2         2       114m
replicaset.apps/frontend-7c58b6d57                   2         2         2       5m9s
replicaset.apps/ingress-nginx-controller-d49697d5f   1         1         1       2d4h
replicaset.apps/mongo-d8f65d9bd                      1         1         1       38h
```

5.2 kubectl get services output, showing the LoadBalancer and other service details

In order to get details of all services in the cluster, we have used the **kubectl get svc** command. Currently, we have 6 ongoing services active in our cluster, namely backend-service, frontend-service, ingress-nginx-controller (which is LoadBalancer type, and allows external access to the application), ingress-nginx-controller-admission, kubernetes, and mongo.

Services like backend-service, frontend-service, ingress-nginx-controller-admission, kubernetes, and mongo are **Internal Services**, and are only accessible within the cluster for secure and internal communication, while we also have an **external service**, ingress-nginx-controller, which provides external access to the application via an AWS Load Balancer (ELB), exposing ports 80 and 443.



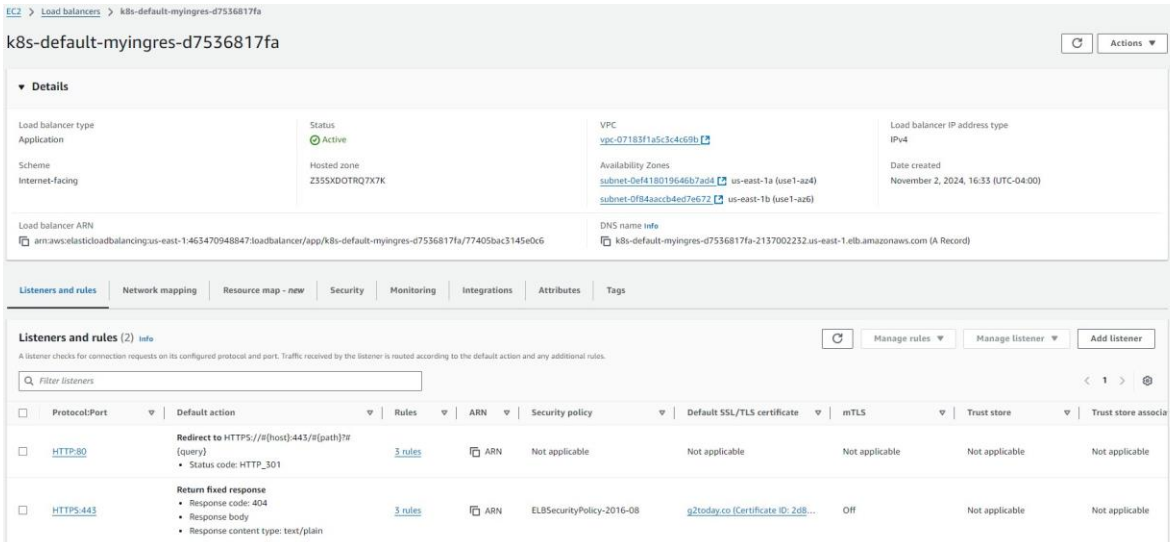
```
┌──(devansh㉿pegasus)-[~/Code/VirtualizationMidterm/kubernetes]
└─$ kubectl get svc
NAME                                 TYPE           CLUSTER-IP       EXTERNAL-IP                                                               PORT(S)                      AGE
backend-service                      ClusterIP      10.100.35.64     <none>                                                                    3000/TCP                     5m41s
frontend-service                     ClusterIP      10.100.126.211   <none>                                                                    80/TCP                       10m
ingress-nginx-controller             LoadBalancer   10.100.103.138   a863917b32e3d4ebfaafbd14e4ae9970-1072383381.us-east-1.elb.amazonaws.com   80:32214/TCP,443:30606/TCP   2d2h
ingress-nginx-controller-admission   ClusterIP      10.100.28.72     <none>                                                                    443/TCP                      2d2h
kubernetes                           ClusterIP      10.100.0.1       <none>                                                                    443/TCP                      3d
mongo                                ClusterIP      10.100.253.153   <none>                                                                    27017/TCP                    36h
```

6. Domain Setup and SSL Configuration:

6.1 LoadBalancer Configuration:

To facilitate secure and scalable access to our Doctor’s Office Appointment application, we configured an Application LoadBalancer (ALB) in AWS. The ALB routes traffic to the microservices deployed in our EKS cluster, balancing incoming requests and enforcing HTTPS for secure connections.

The HTTPS listener is configured with the SSL certificate issued by AWS Certificate Manager (ACM) for our custom domain, **g2today.co.**



EC2 > Load balancers > k8s-default-myingres-d7536817fa

k8s-default-myingres-d7536817fa

Details

Load balancer type: Application	Status: Active	VPC: vpc-07183f1a5c3c4c69b	Load balancer IP address type: IPv4
Scheme: Internet-facing	Hosted zone: Z35SXDOTRQ7X7K	Availability Zones: subnet-0ef418019646b7ad4 us-east-1a (use1-az4); subnet-0f84aaccb4ed7e672 us-east-1b (use1-az6)	Date created: November 2, 2024, 16:33 (UTC-04:00)

Load balancer ARN: arn:aws:elasticloadbalancing:us-east-1:463470948847:loadbalancer/app/k8s-default-myingres-d7536817fa/77405bac3145e0c6

DNS name: k8s-default-myingres-d7536817fa-2137002232.us-east-1.elb.amazonaws.com (A Record)

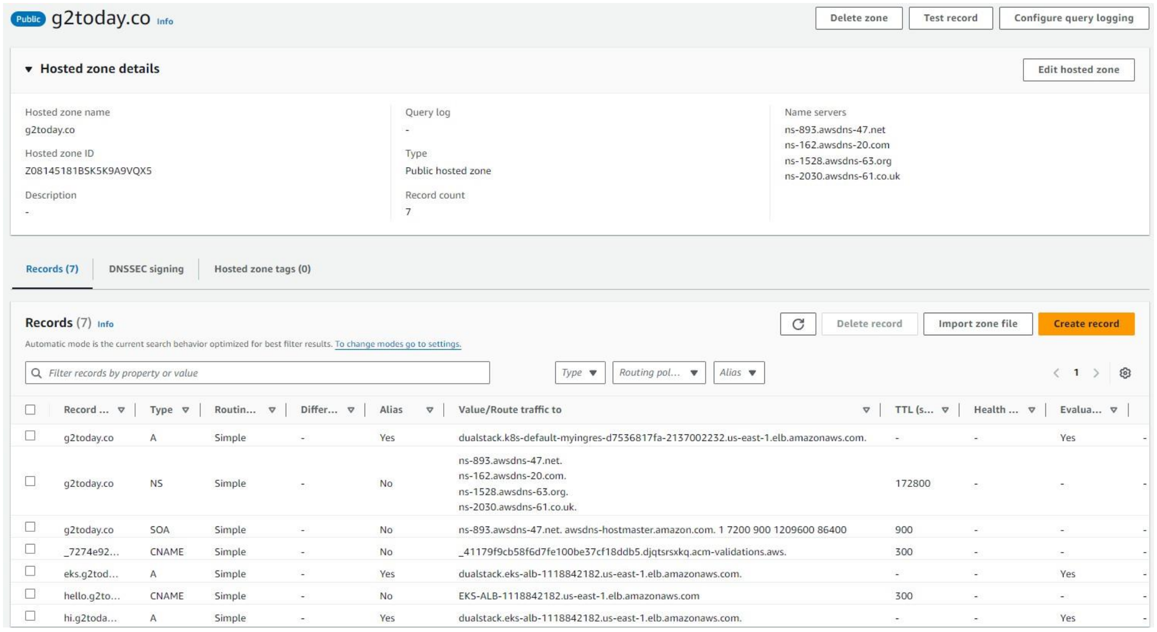
Listeners and rules (2)

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Trust store	Trust store associa
HTTP:80	**Redirect to** HTTPS://#{host}:443/#{path}?#{query} • Status code: HTTP_301	3 rules	ARN	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable
HTTPS:443	**Return fixed response** • Response code: 404 • Response body • Response content type: text/plain	3 rules	ARN	ELBSecurityPolicy-2016-08	g2today.co (Certificate ID: 2d8...	Off	Not applicable	Not applicable

6.2 Route 53 Hosted Zone Configuration for Domain Mapping:

The screenshot demonstrates the Route 53 hosted zone configuration for g2today.co, showing how domain traffic is routed to our Kubernetes-based application infrastructure.



Public g2today.co

Hosted zone details

Hosted zone name: g2today.co	Query log: -	Name servers: ns-893.awsdns-47.net, ns-162.awsdns-20.com, ns-1528.awsdns-63.org, ns-2030.awsdns-61.co.uk
Hosted zone ID: Z08145181BSK5K9A9VQX5	Type: Public hosted zone	
Description: -	Record count: 7	

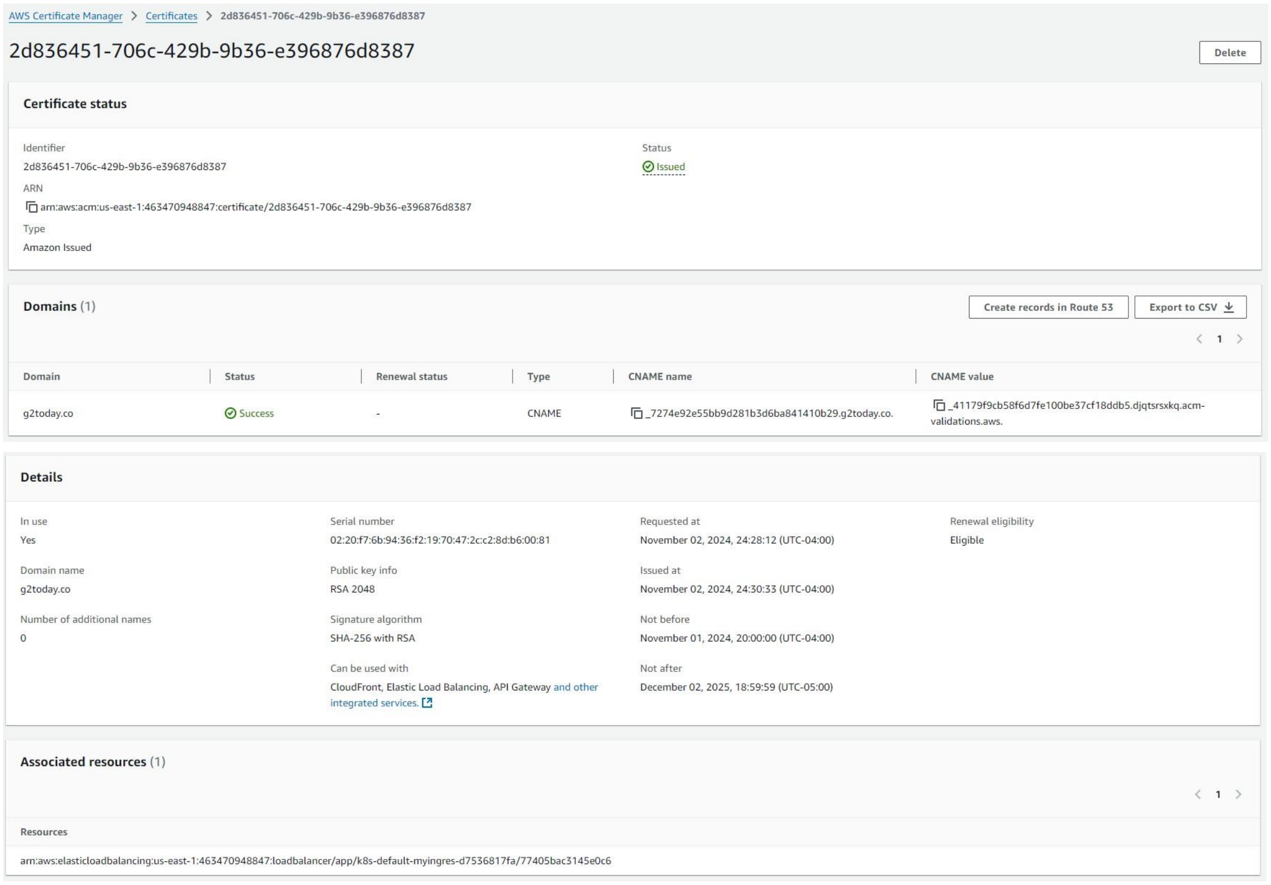
Records (7)

Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Record ...	Type	Routin...	Differ...	Alias	Value/Route traffic to	TTL (s...	Health ...	Evalua...
g2today.co	A	Simple	-	Yes	dualstack.k8s-default-myingres-d7536817fa-2137002232.us-east-1.elb.amazonaws.com.	-	-	Yes
g2today.co	NS	Simple	-	No	ns-893.awsdns-47.net. ns-162.awsdns-20.com. ns-1528.awsdns-63.org. ns-2030.awsdns-61.co.uk.	172800	-	-
g2today.co	SOA	Simple	-	No	ns-893.awsdns-47.net. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400	900	-	-
_7274e92...	CNAME	Simple	-	No	_41179f9cb58f6d7fe100be37cf18ddb5.djqtsrsxkq.acm-validations.aws.	300	-	-
eks.g2tod...	A	Simple	-	Yes	dualstack.eks-alb-1118842182.us-east-1.elb.amazonaws.com.	-	-	Yes
hello.g2to...	CNAME	Simple	-	No	EKS-ALB-1118842182.us-east-1.elb.amazonaws.com	300	-	-
hi.g2toda...	A	Simple	-	Yes	dualstack.eks-alb-1118842182.us-east-1.elb.amazonaws.com.	-	-	Yes

6.3 AWS Certificate Manager for SSL Configuration:

The screenshots below demonstrate the SSL/TLS certificate configuration for g2today.co, showing both the certificate details and its integration with AWS services.



AWS Certificate Manager > Certificates > 2d836451-706c-429b-9b36-e396876d8387

2d836451-706c-429b-9b36-e396876d8387

Certificate status

Identifier
2d836451-706c-429b-9b36-e396876d8387

ARN
arn:aws:acm:us-east-1:463470948847:certificate/2d836451-706c-429b-9b36-e396876d8387

Type
Amazon Issued

Status
Issued

Domains (1)

Domain	Status	Renewal status	Type	CNAME name	CNAME value
g2today.co	Success	-	CNAME	_7274e92e55bb9d281b3d6ba841410b29.g2today.co.	_41179f9cb58f6d7fe100be37cf18ddb5.djqtsrsxkq.acm-validations.aws.

Details

In use
Yes

Domain name
g2today.co

Number of additional names
0

Serial number
02:20:f7:6b:94:36:f2:19:70:47:2c:c2:8d:b6:00:81

Public key info
RSA 2048

Signature algorithm
SHA-256 with RSA

Can be used with
CloudFront, Elastic Load Balancing, API Gateway and other integrated services.

Requested at
November 02, 2024, 24:28:12 (UTC-04:00)

Issued at
November 02, 2024, 24:30:33 (UTC-04:00)

Not before
November 01, 2024, 20:00:00 (UTC-04:00)

Not after
December 02, 2025, 18:59:59 (UTC-05:00)

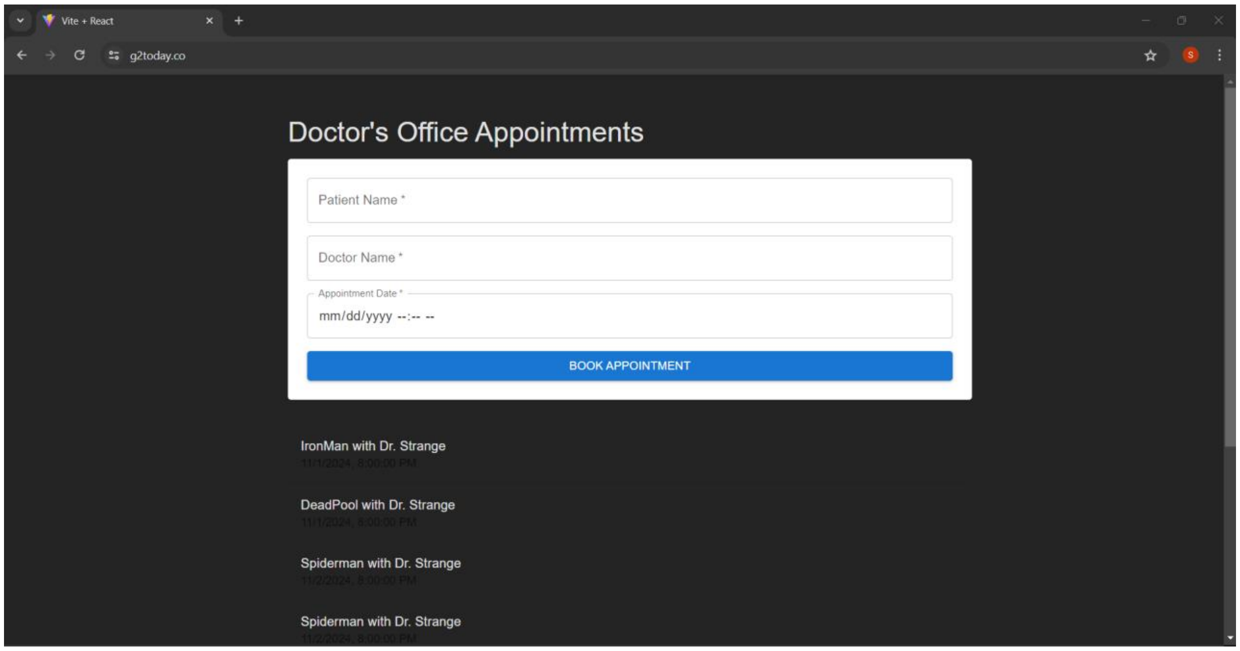
Renewal eligibility
Eligible

Associated resources (1)

Resources
arn:aws:elasticloadbalancing:us-east-1:463470948847:loadbalancer/app/k8s-default-myingres-d7536817fa/77405bac3145e0c6

6.4 Domain Access Verification:

The images below show the successful deployment of the application and secure external access. The **SSL implementation** has been successfully validated through both browser indicators and detailed security panel information, confirming secure HTTPS connectivity and valid certificate implementation.



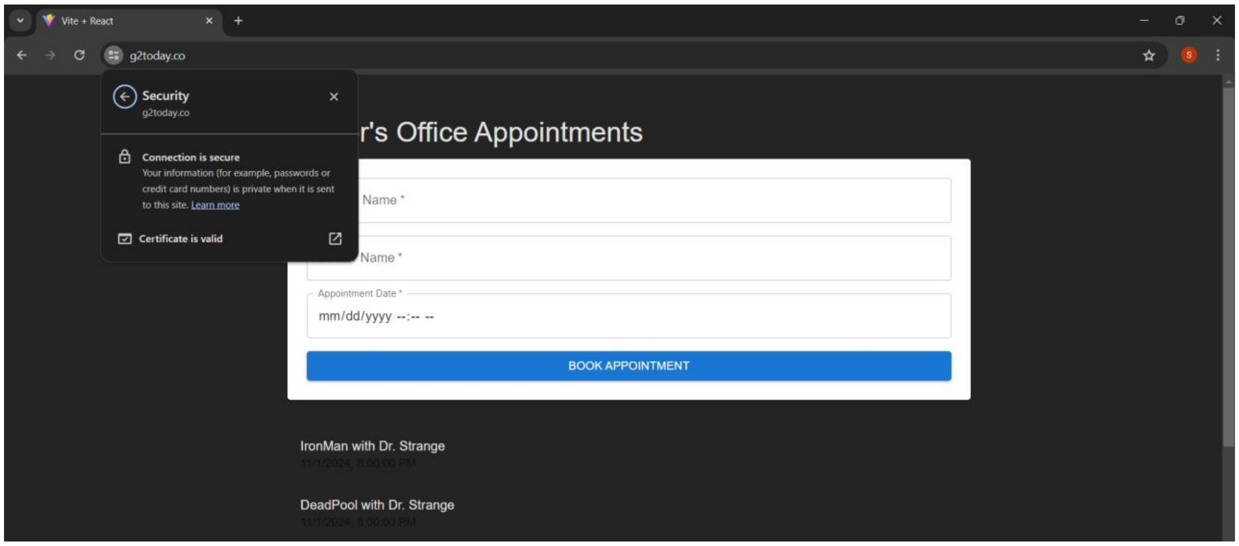
Doctor's Office Appointments

IronMan with Dr. Strange
11/1/2024, 8:00:00 PM

DeadPool with Dr. Strange
11/1/2024, 8:00:00 PM

Spiderman with Dr. Strange
11/2/2024, 8:00:00 PM

Spiderman with Dr. Strange
11/2/2024, 8:00:00 PM



Connection is secure
Your information (for example, passwords or credit card numbers) is private when it is sent to this site. Learn more

Certificate is valid

r's Office Appointments

IronMan with Dr. Strange
11/1/2024, 8:00:00 PM

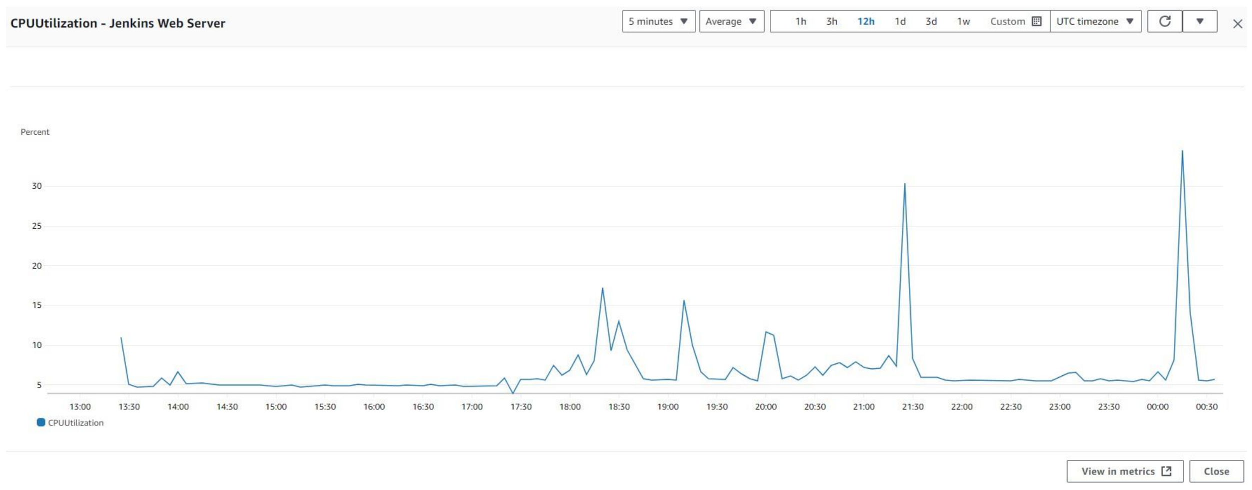
DeadPool with Dr. Strange
11/1/2024, 8:00:00 PM

7. Monitoring

7.1 CloudWatch Metrics:

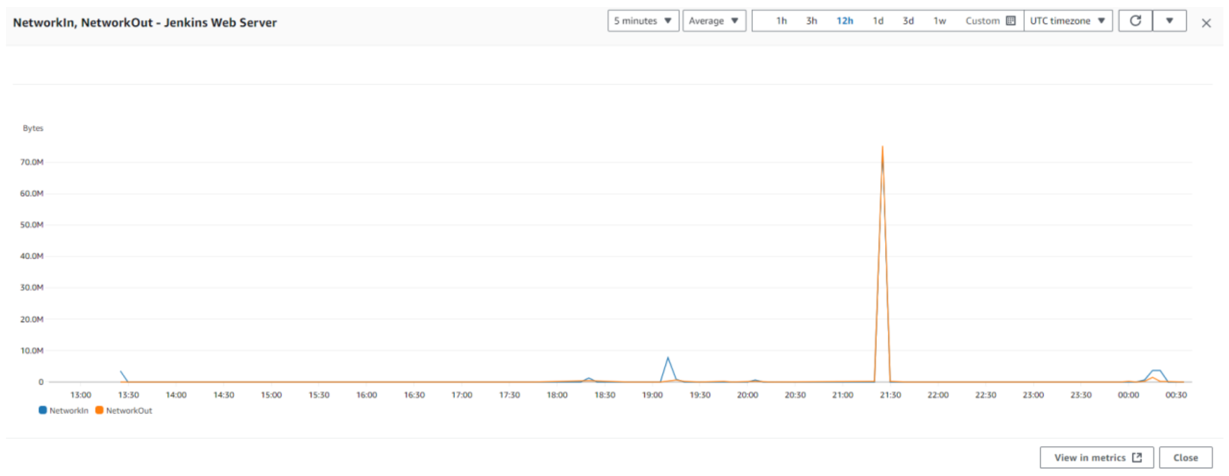
7.1.1 Dashboard for CPU Utilization Metrics of Jenkins Web Server

The dashboard below shows the CPU utilization of a Jenkins web server over approximately a 12-hour period. The server handles peak loads with remarkable efficiency, reaching only 30% CPU utilization during the most intensive builds. The consistent baseline of 5% CPU usage during regular hours shows a stable, well-maintained system.



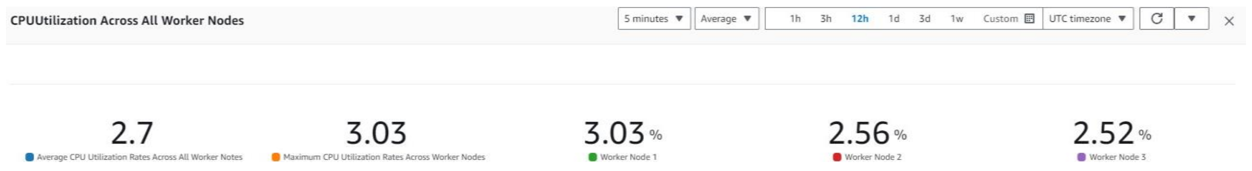
7.1.2 Dashboard for Network Traffic of the Jenkins Web Server

The dashboard shown below tracks how much data is moving in and out of the Jenkins server. It's basically a health monitor for our Jenkins server's network activity - helping us make sure our CI/CD pipeline isn't getting clogged up with too much data traffic at once and that everything's flowing smoothly. This network traffic dashboard is crucial for monitoring Jenkins as it shows when large artifacts are being uploaded/downloaded and helps identify if network bottlenecks are affecting build times, thereby validating that CI/CD pipelines are running as scheduled.



7.1.3 Dashboard for CPU Utilization Across All Worker Nodes:

The dashboard below shows the CPU utilization metrics across multiple worker nodes in the EKS cluster. It shows a consistently low CPU utilization across all worker nodes in the cluster. The current CPU utilization patterns indicate significant unused capacity across the worker node cluster. All nodes are operating at less than 4% of their total CPU capacity, suggesting these worker nodes are currently under very light load. The minimal variation between nodes (range of 0.51 percentage points) demonstrates balanced workload distribution across the cluster.


CPUUtilization Across All Worker Nodes

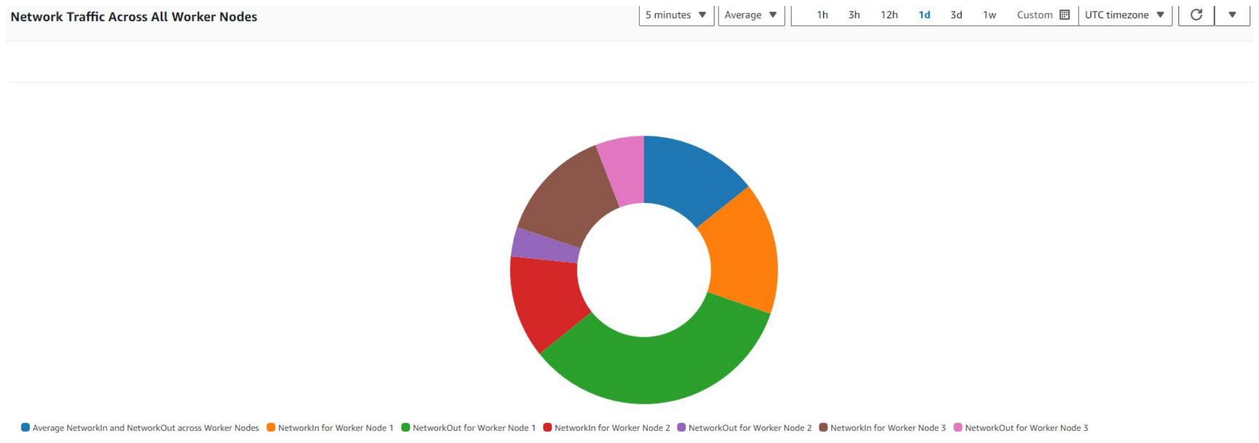
5 minutes ▼ | Average ▼ | 1h 3h 12h 1d 3d 1w Custom | UTC timezone ▼

2.7	3.03	3.03 %	2.56 %	2.52 %
● Average CPU Utilization Rates Across All Worker Notes	● Maximum CPU Utilization Rates Across Worker Nodes	● Worker Node 1	● Worker Node 2	● Worker Node 3

7.1.4 Dashboard for Network Traffic Across All Worker Nodes

The image below is that of a pie chart showing network traffic distribution across multiple worker nodes, with both inbound (NetworkIn) and outbound (NetworkOut) traffic. This network traffic visualization demonstrates a well-functioning distributed system. Worker Node 3's higher NetworkIn (44.03%) suggests it's successfully handling bulk data ingestion or serving as a primary entry point. Worker Node 1's balanced in/out traffic indicates it's effectively processing and transforming data. Worker Node 2's lighter traffic could be optimally serving as a backup or handling specialized tasks.

The varying traffic patterns suggest resource optimization, with each node serving its intended purpose. The system appears to be maintaining steady operations without any nodes showing signs of traffic overflow. The distributed nature of the traffic helps ensure system reliability and fault tolerance.

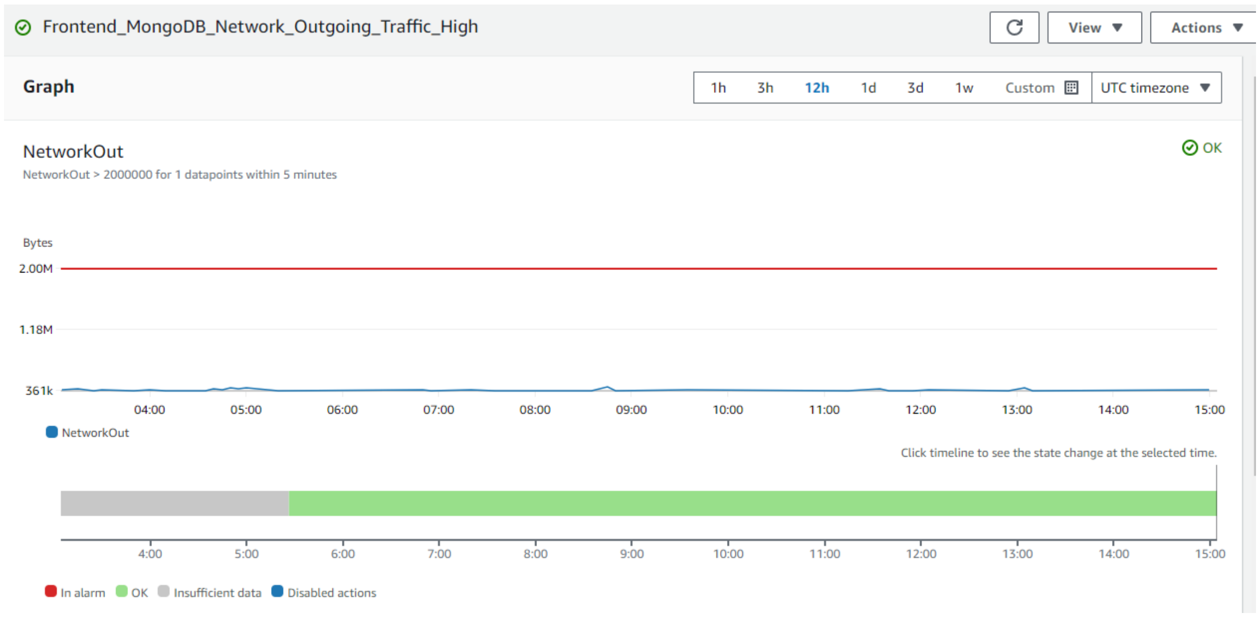


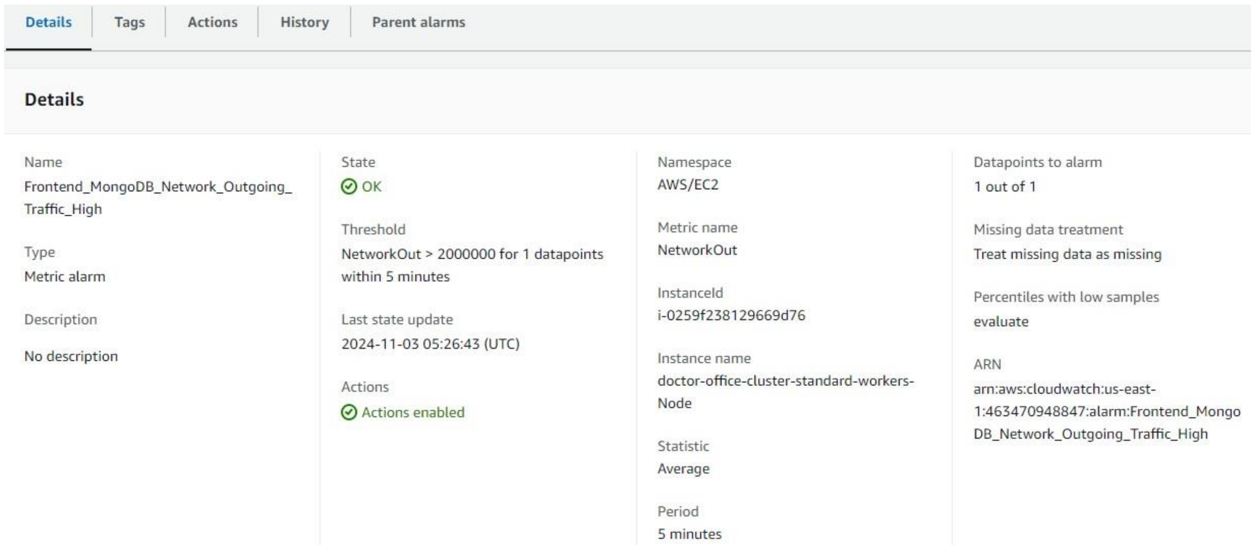
7.2 CloudWatch Alarms

7.2.1 Alarm for High Outgoing Network Traffic for MongoDB

The two images below show the CloudWatch Metric Alarm, monitoring outgoing network traffic for MongoDB. The threshold is set at 2,000,000 bytes (2M) for data points within 5-minute intervals and shows that the traffic is consistently stable around 361k bytes, well below the 2M threshold. No spikes or anomalies are visible, suggesting stable application performance.

This alarm helps ensure the MongoDB database isn't experiencing network bottlenecks and provides early warning if traffic approaches concerning levels. This allows for capacity planning based on actual usage patterns and helps prevent potential outages or performance issues.



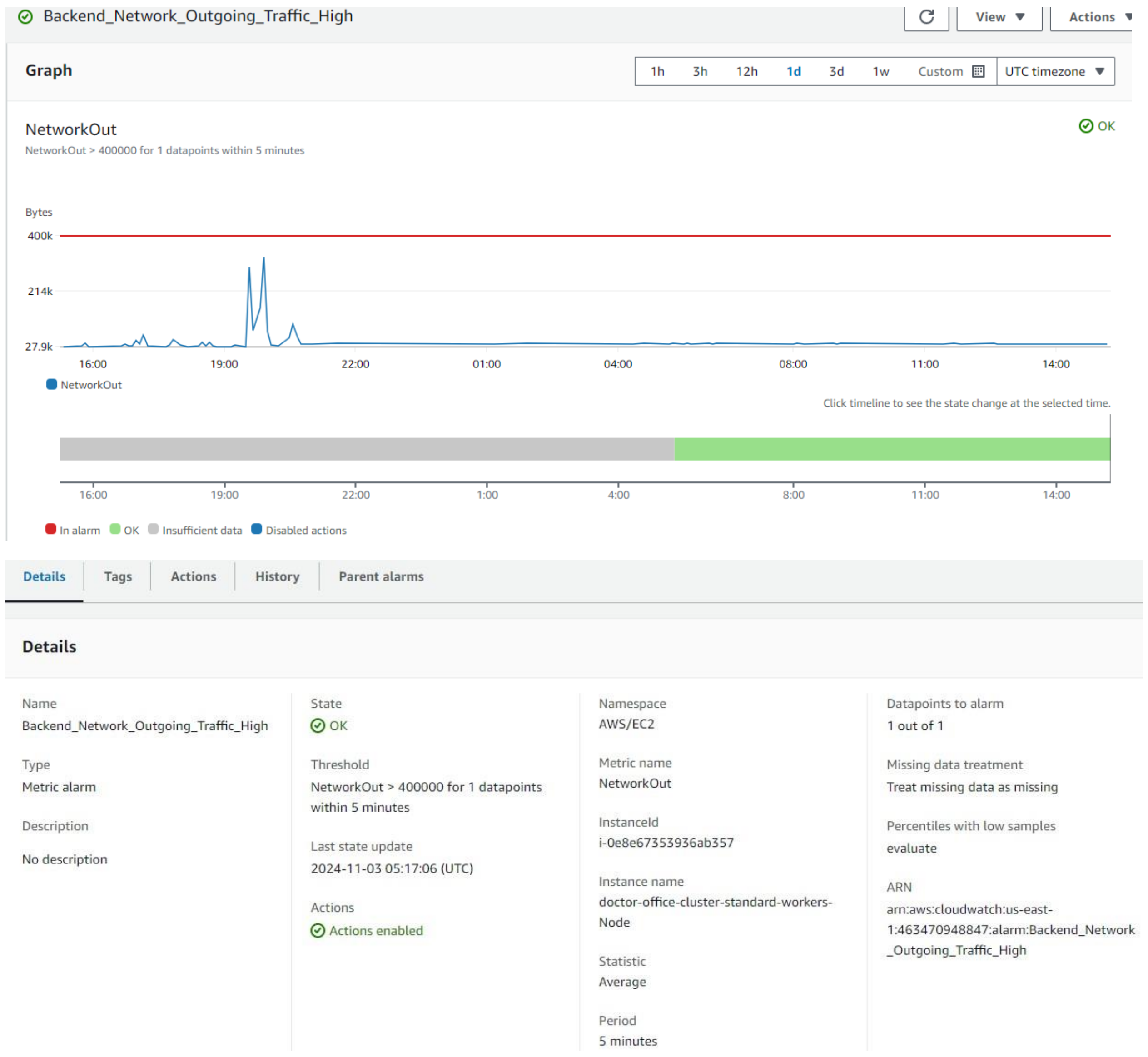


Details

Name: Frontend_MongoDB_Network_Outgoing_Traffic_High	State: OK	Namespace: AWS/EC2	Datapoints to alarm: 1 out of 1
Type: Metric alarm	Threshold: NetworkOut > 2000000 for 1 datapoints within 5 minutes	Metric name: NetworkOut	Missing data treatment: Treat missing data as missing
Description: No description	Last state update: 2024-11-03 05:26:43 (UTC)	InstanceId: i-0259f238129669d76	Percentiles with low samples: evaluate
	Actions: Actions enabled	Instance name: doctor-office-cluster-standard-workers-Node	ARN: arn:aws:cloudwatch:us-east-1:463470948847:alarm:Frontend_MongoDB_Network_Outgoing_Traffic_High
		Statistic: Average	
		Period: 5 minutes	

7.2.2 Alarm for Outgoing Traffic on EC2 Instances in the Backend

The two images shown below are alarms that are monitoring the Backend Network Outgoing Traffic on AWS EC2 instances. The threshold is set at 400,000 bytes (400k) within 5-minute intervals. The baseline traffic is around 27.9k bytes, which is well below the threshold level. This demonstrates that the backend is handling traffic well below its threshold and shows system stability with quick recovery from traffic spikes.



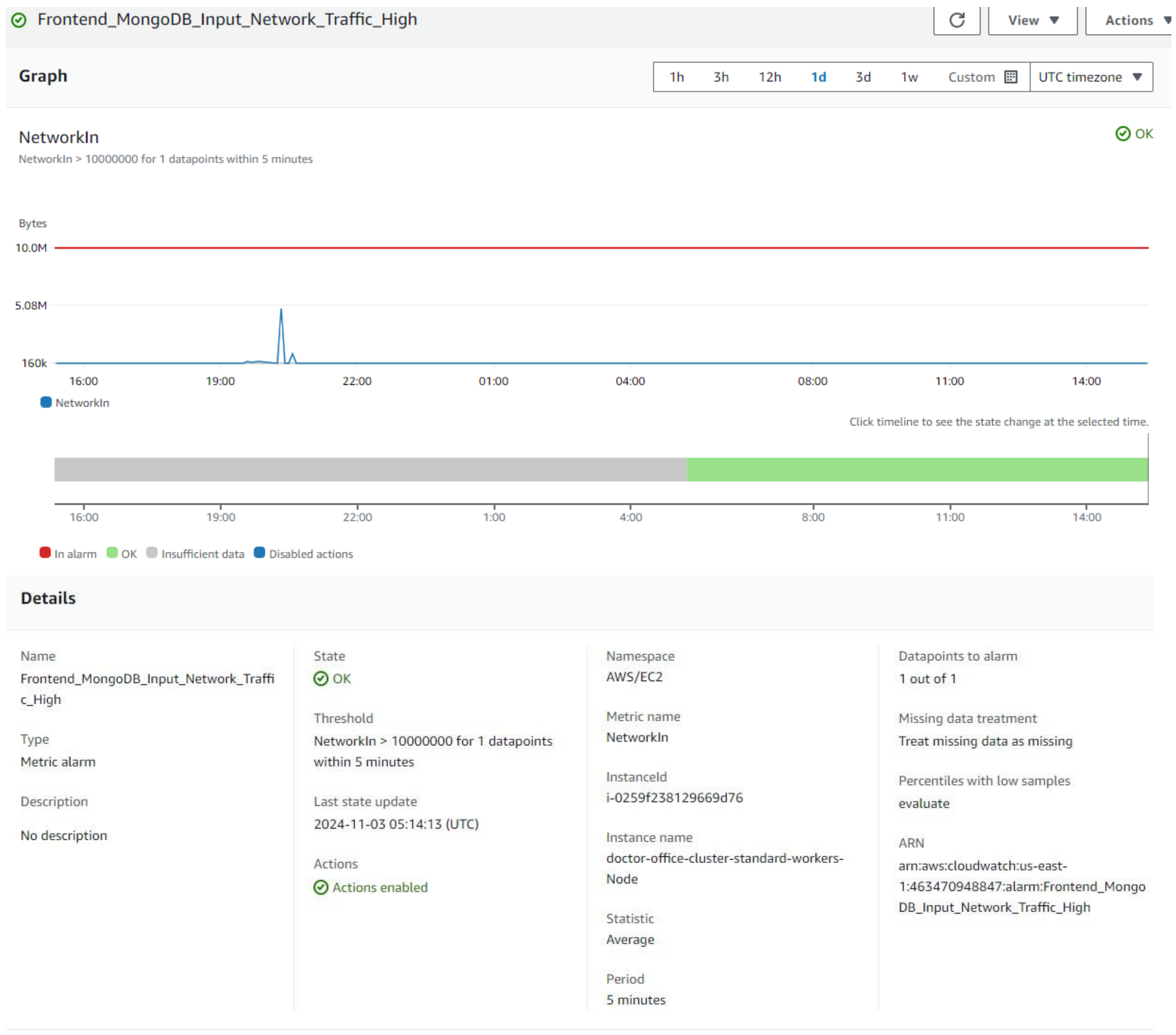
Details

Name Backend_Network_Outgoing_Traffic_High	State OK	Namespace AWS/EC2	Datapoints to alarm 1 out of 1
Type Metric alarm	Threshold NetworkOut > 400000 for 1 datapoints within 5 minutes	Metric name NetworkOut	Missing data treatment Treat missing data as missing
Description No description	Last state update 2024-11-03 05:17:06 (UTC)	InstanceId i-0e8e67353936ab357	Percentiles with low samples evaluate
	Actions Actions enabled	Instance name doctor-office-cluster-standard-workers-Node	ARN arn:aws:cloudwatch:us-east-1:463470948847:alarm:Backend_Network_Outgoing_Traffic_High
		Statistic Average	
		Period 5 minutes	

7.2.3 Alarm for Frontend MongoDB Input Network Traffic

The two images below show alarms monitoring Frontend MongoDB Input Network Traffic on AWS EC2 instances. The threshold is set at 10,000,000 bytes (10M) within 5-minute intervals. The baseline traffic is at approximately 160k bytes, which indicates a very stable system. There is a notable spike at around 19:00-22:00 hrs, reaching about 5.08M bytes, but there is a quick recovery to baseline after the spike.

This dashboard is particularly crucial as it monitors the data input to MongoDB, which handles important healthcare records. The higher threshold compared to output traffic (10M vs 2M) suggests the system is well-designed for handling large data ingests while maintaining stability.



Frontend_MongoDB_Input_Network_Traffic_High

Graph

NetworkIn

NetworkIn > 10000000 for 1 datapoints within 5 minutes

OK

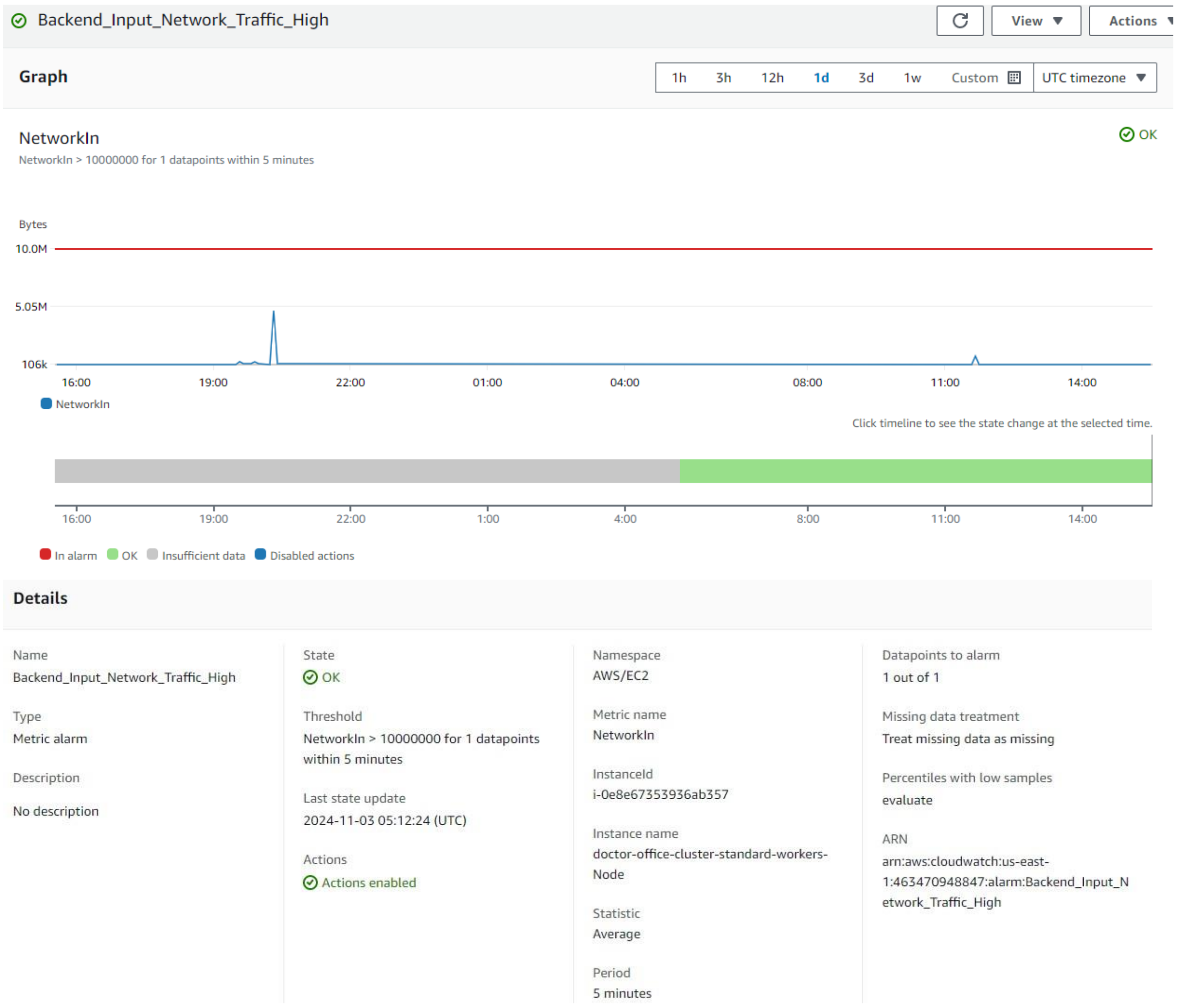
Details

Name Frontend_MongoDB_Input_Network_Traffic_High	State OK	Namespace AWS/EC2	Datapoints to alarm 1 out of 1
Type Metric alarm	Threshold NetworkIn > 10000000 for 1 datapoints within 5 minutes	Metric name NetworkIn	Missing data treatment Treat missing data as missing
Description No description	Last state update 2024-11-03 05:14:13 (UTC)	InstanceId i-0259f238129669d76	Percentiles with low samples evaluate
	Actions Actions enabled	Instance name doctor-office-cluster-standard-workers-Node	ARN arn:aws:cloudwatch:us-east-1:463470948847:alarm:Frontend_MongoDB_Input_Network_Traffic_High
		Statistic Average	
		Period 5 minutes	

7.2.4 Alarm for Backend Input Network Traffic

This monitors the Backend Input Network Traffic on AWS EC2 instances. The threshold is set at 10,000,000 bytes (10M) within 5-minute intervals. The baseline traffic appears to be around 106k bytes, which indicates a steady system. There is a significant spike near 22:00 hrs, reaching about 5.05M bytes, but there is a quick return to baseline after the spike.

This dashboard is particularly important as it monitors all incoming traffic to the backend services, which is crucial for maintaining overall system performance and reliability in a healthcare application environment.

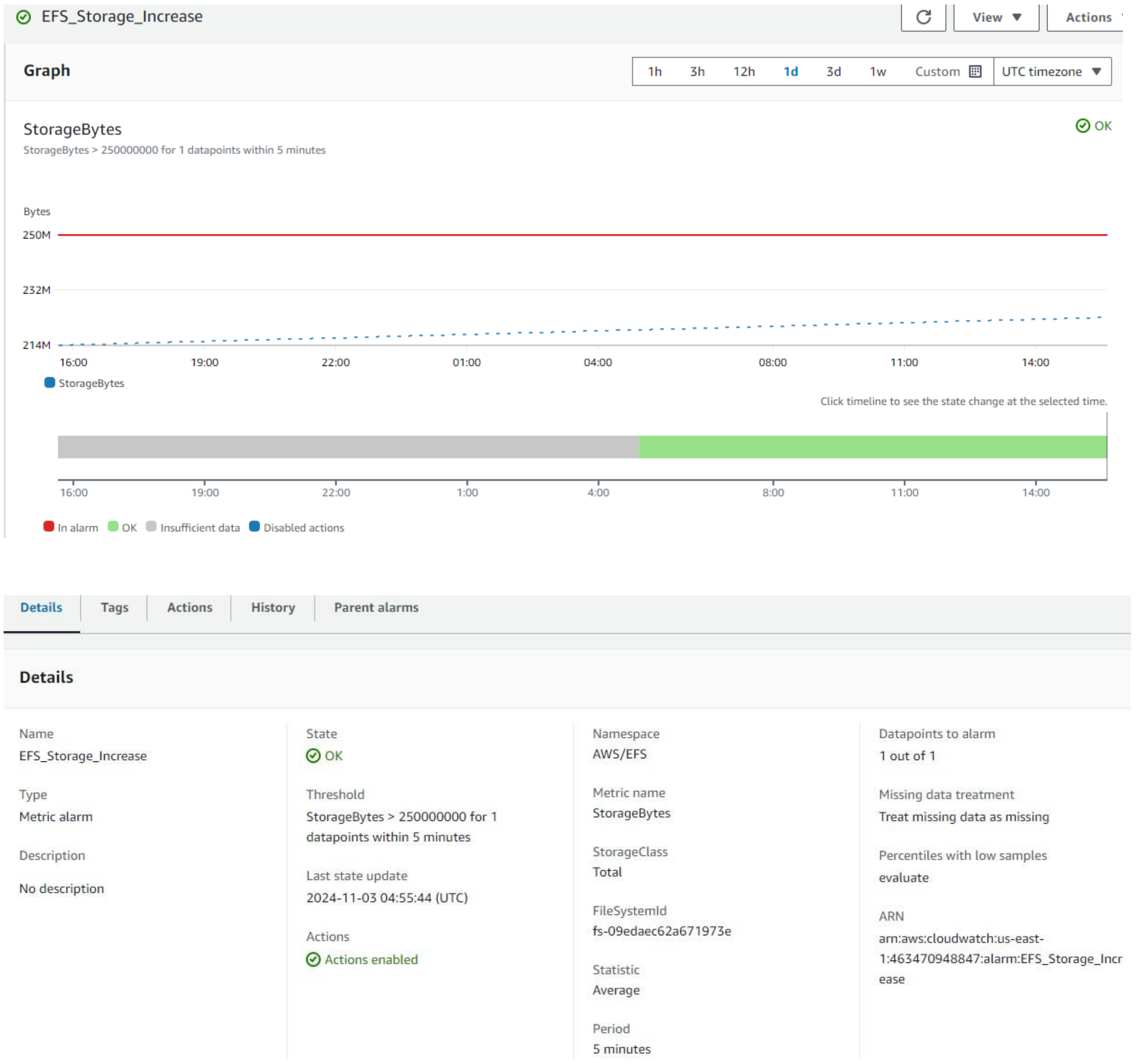


Details

Name Backend_Input_Network_Traffic_High	State OK	Namespace AWS/EC2	Datapoints to alarm 1 out of 1
Type Metric alarm	Threshold NetworkIn > 10000000 for 1 datapoints within 5 minutes	Metric name NetworkIn	Missing data treatment Treat missing data as missing
Description No description	Last state update 2024-11-03 05:12:24 (UTC)	InstanceId i-0e8e67353936ab357	Percentiles with low samples evaluate
	Actions Actions enabled	Instance name doctor-office-cluster-standard-workers-Node	ARN arn:aws:cloudwatch:us-east-1:463470948847:alarm:Backend_Input_Network_Traffic_High
		Statistic Average	
		Period 5 minutes	

7.2.5 Alarm for monitoring an EFS (Amazon Elastic File System) storage

The images below are that of a dashboard monitoring an EFS (Amazon Elastic File System) storage metric. The threshold here is set at 250,000,000 bytes (approximately 250MB). This dashboard helps monitor EFS storage growth to prevent unexpected storage expansion and associated costs. The alarm is set to trigger if storage exceeds 250MB within a 5-minute window, allowing for proactive management before storage issues become critical.



EFS_Storage_Increase
View
Actions
Graph
1h 3h 12h 1d 3d 1w Custom
UTC timezone
OK
Bytes
250M
232M
214M
16:00 19:00 22:00 01:00 04:00 08:00 11:00 14:00
StorageBytes
Click timeline to see the state change at the selected time.
16:00 19:00 22:00 1:00 4:00 8:00 11:00 14:00
In alarm OK Insufficient data Disabled actions
Details Tags Actions History Parent alarms

StorageBytes
StorageBytes > 250000000 for 1 datapoints within 5 minutes

Details

Name
EFS_Storage_Increase

Type
Metric alarm

Description
No description

State
✓ OK

Threshold
StorageBytes > 250000000 for 1 datapoints within 5 minutes

Last state update
2024-11-03 04:55:44 (UTC)

Actions
✓ Actions enabled

Namespace
AWS/EFS

Metric name
StorageBytes

StorageClass
Total

FileSystemId
fs-09edaec62a671973e

Statistic
Average

Period
5 minutes

Datapoints to alarm
1 out of 1

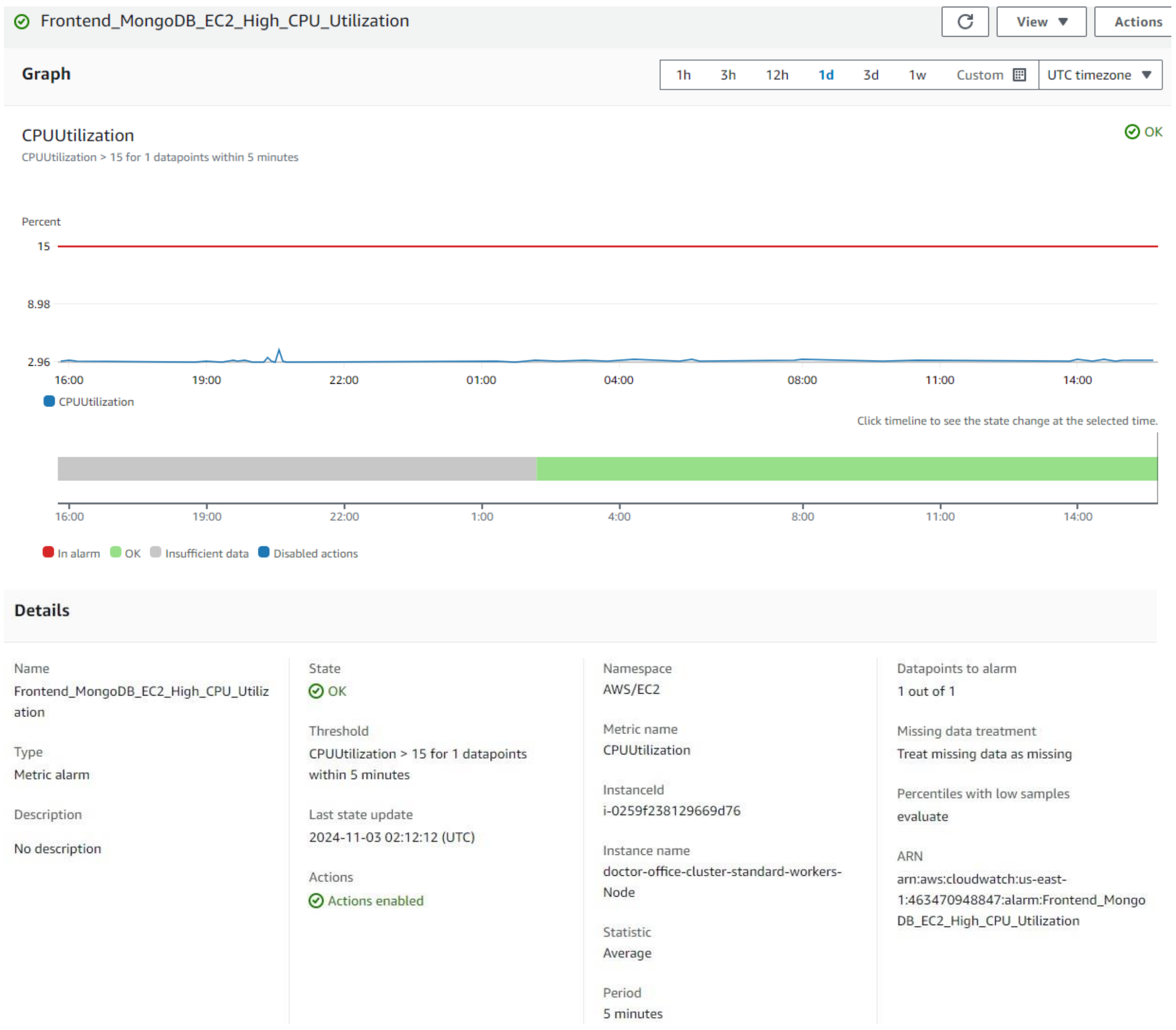
Missing data treatment
Treat missing data as missing

Percentiles with low samples
evaluate

ARN
arn:aws:cloudwatch:us-east-1:463470948847:alarm:EFS_Storage_Increase

7.2.6 Alarm for monitoring CPU utilization for a MongoDB instance running on EC2

The two images below are for a dashboard, monitoring CPU utilization for a MongoDB instance running on EC2. The threshold is set at 15% CPU usage. The CPU utilization is consistently running at around 2.96%. This alarm helps ensure the MongoDB database server isn't experiencing excessive CPU load that could impact application performance.



Details

Name
Frontend_MongoDB_EC2_High_CPU_Utilization

Type
Metric alarm

Description
No description

State
OK

Threshold
CPUUtilization > 15 for 1 datapoints within 5 minutes

Last state update
2024-11-03 02:12:12 (UTC)

Actions
Actions enabled

Namespace
AWS/EC2

Metric name
CPUUtilization

InstanceId
i-0259f238129669d76

Instance name
doctor-office-cluster-standard-workers-Node

Statistic
Average

Period
5 minutes

Datapoints to alarm
1 out of 1

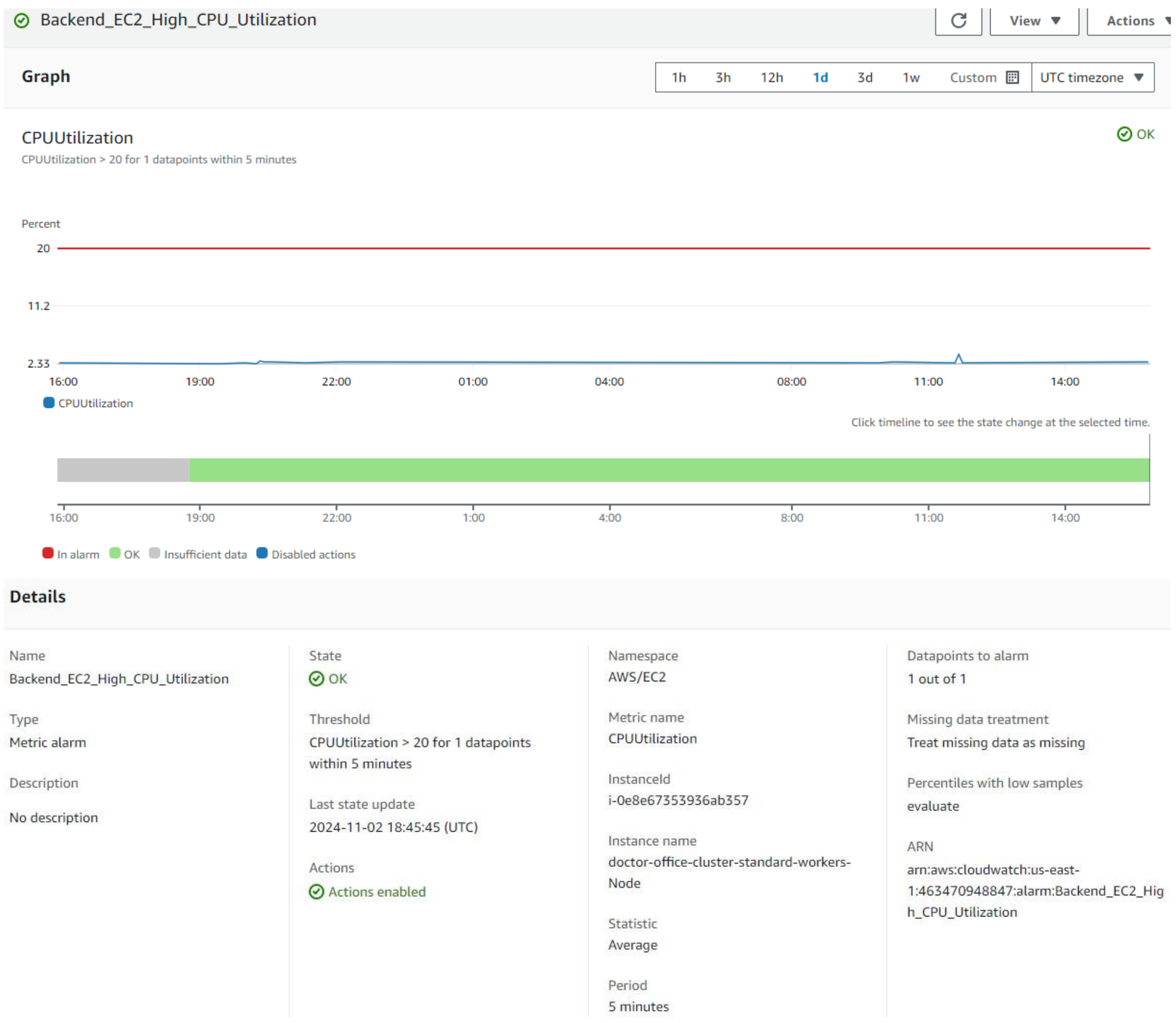
Missing data treatment
Treat missing data as missing

Percentiles with low samples
evaluate

ARN
arn:aws:cloudwatch:us-east-1:463470948847:alarm:Frontend_MongoDB_EC2_High_CPU_Utilization

7.2.7 Alarm for CPU utilization for an EC2 instance for a Worker Node

The two images below are for an alarm that tracks the CPU utilization for an EC2 instance, specifically for a backend worker node from the EKS cluster. The threshold alert here is set at 20%. This alarm helps ensure that the backend system isn't being overloaded.



Details

Name Backend_EC2_High_CPU_Utilization	State ✓ OK	Namespace AWS/EC2	Datapoints to alarm 1 out of 1
Type Metric alarm	Threshold CPUUtilization > 20 for 1 datapoints within 5 minutes	Metric name CPUUtilization	Missing data treatment Treat missing data as missing
Description No description	Last state update 2024-11-02 18:45:45 (UTC)	InstanceId i-0e8e67353936ab357	Percentiles with low samples evaluate
	Actions ✓ Actions enabled	Instance name doctor-office-cluster-standard-workers-Node	ARN arn:aws:cloudwatch:us-east-1:463470948847:alarm:Backend_EC2_High_CPU_Utilization
		Statistic Average	
		Period 5 minutes	

7.2.8 Alarm for monitoring the health of an Amazon EKS (Elastic Kubernetes Service) cluster's worker nodes through an Auto Scaling Group

This alarm monitors the health of an Amazon EKS (Elastic Kubernetes Service) cluster's worker nodes through an Auto Scaling Group. The red line shows the minimum threshold of 3 instances. The graph shows consistent service levels with no drops below the threshold and the Y-axis scale goes from 2 to 4 instances, providing good visibility of the threshold range.

This dashboard is critically important for several reasons:

1. High Availability Monitoring:

- The 3-instance minimum threshold ensures high availability for the Kubernetes cluster
- Multiple instances provide redundancy and fault tolerance
- Maintaining at least 3 nodes allows for proper distribution of workloads and enables:
    - Rolling updates without service interruption

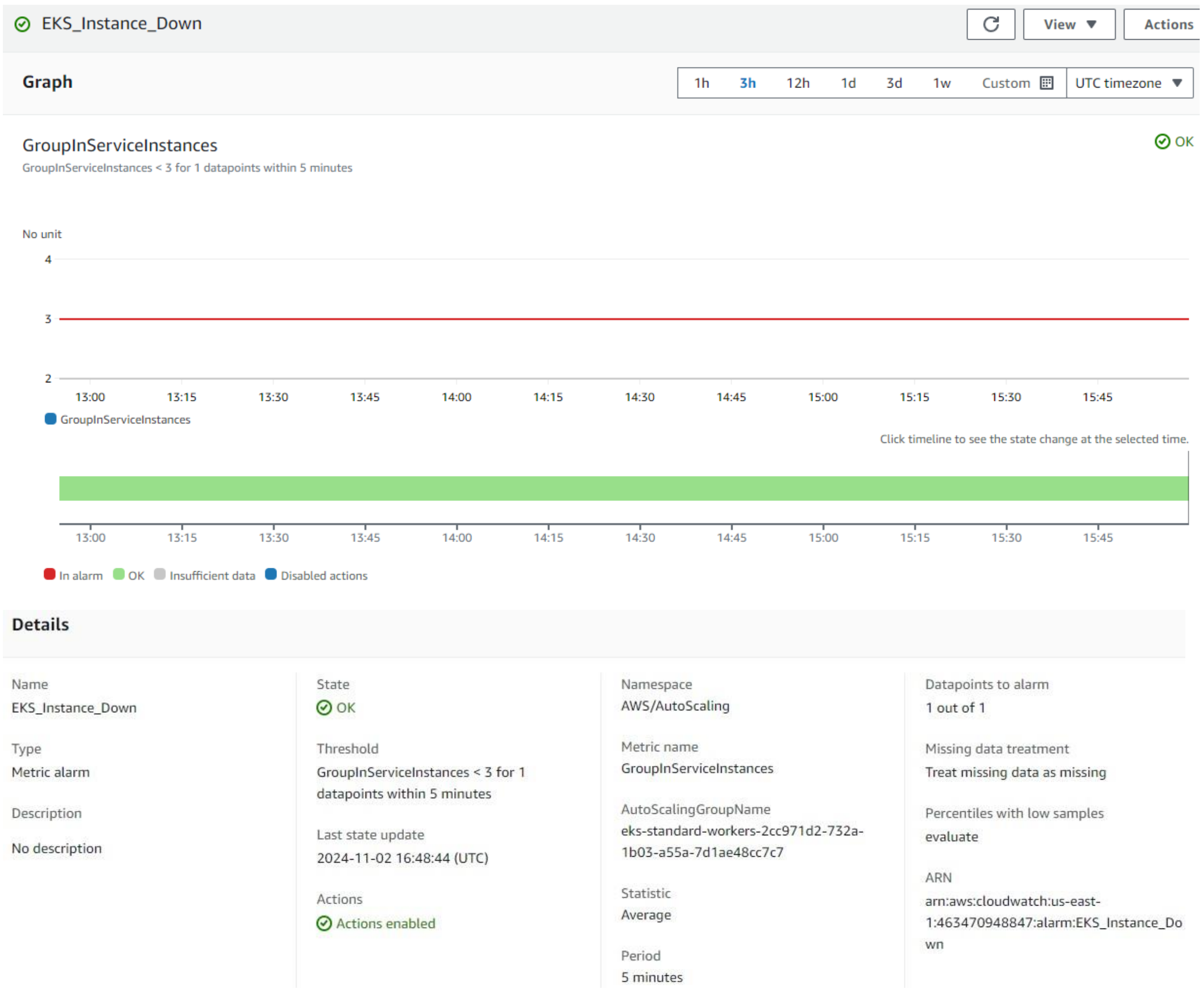
- Load balancing
    - Failover capability

2. Production Stability:

- This metric is crucial for maintaining production stability because:
    - It ensures enough compute capacity for running pods
    - Prevents single points of failure
    - Enables proper pod distribution and scheduling
    - Maintains system resilience

3. Cost and Resource Management:

- Helps maintain the optimal balance between:
    - Having enough instances for reliability
    - Not over-provisioning resources
    - Meeting minimum requirements for Kubernetes operations
    - Cost efficiency while ensuring service quality



GroupInServiceInstances

GroupInServiceInstances < 3 for 1 datapoints within 5 minutes

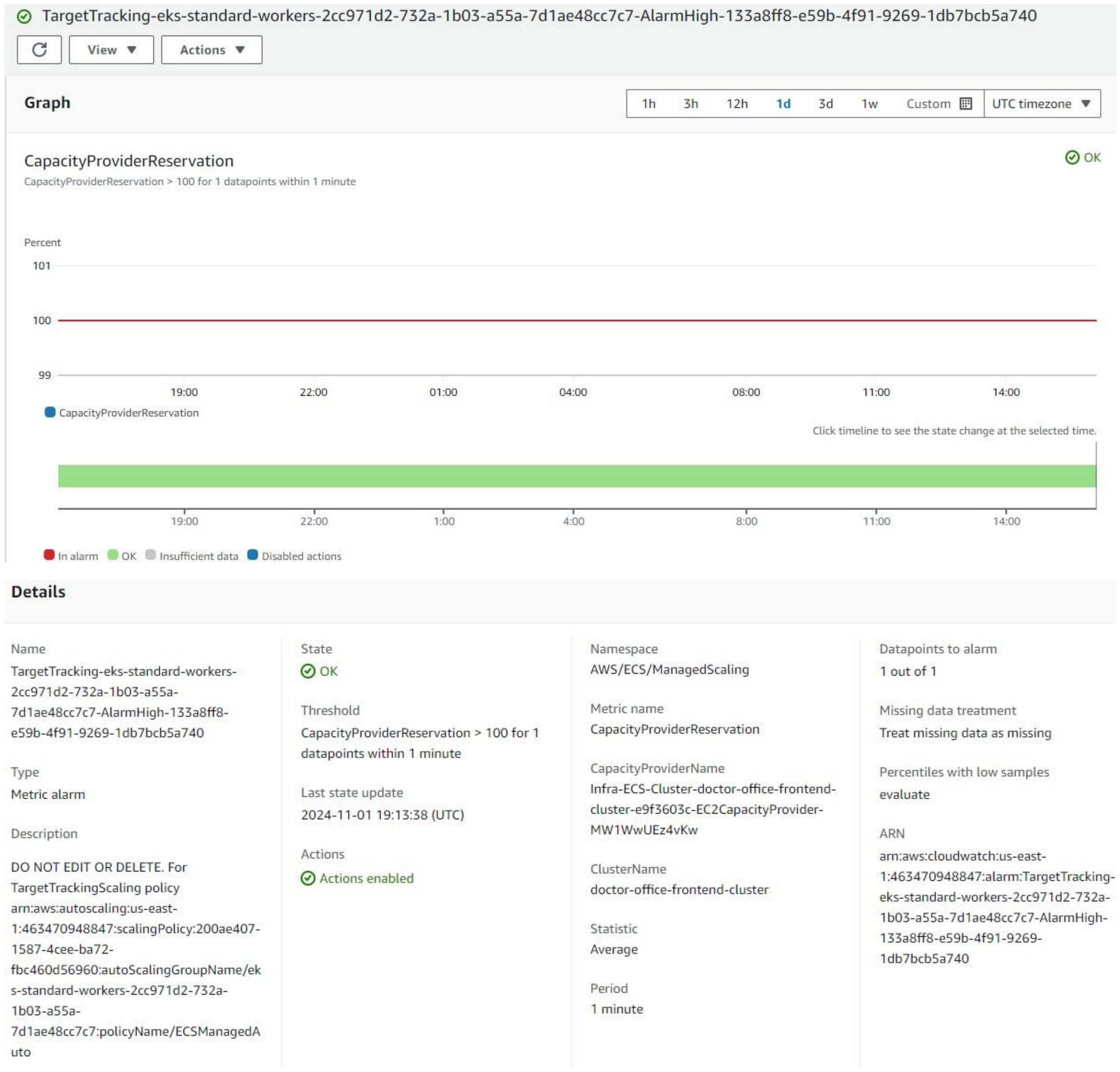
Click timeline to see the state change at the selected time.

Details

Name EKS_Instance_Down	State OK	Namespace AWS/AutoScaling	Datapoints to alarm 1 out of 1
Type Metric alarm	Threshold GroupInServiceInstances < 3 for 1 datapoints within 5 minutes	Metric name GroupInServiceInstances	Missing data treatment Treat missing data as missing
Description No description	Last state update 2024-11-02 16:48:44 (UTC)	AutoScalingGroupName eks-standard-workers-2cc971d2-732a-1b03-a55a-7d1ae48cc7c7	Percentiles with low samples evaluate
	Actions Actions enabled	Statistic Average	ARN arn:aws:cloudwatch:us-east-1:463470948847:alarm:EKS_Instance_Down
		Period 5 minutes	

7.2.9 Alarm for Capacity Provider Reservation for an EKS (Elastic Kubernetes Service) cluster's frontend infrastructure

This alarm monitors the Capacity Provider Reservation for an EKS (Elastic Kubernetes Service) cluster's frontend infrastructure. The threshold here is set at 100%. The graph shows a 24-hour period (from 19:00 to 14:00 the next day). The metric appears stable and below the threshold level.
The continuous green status bar indicates healthy operation.
This dashboard is critically important as it tracks how effectively the EKS cluster is utilizing its allocated capacity, helps prevent resource exhaustion, and ensures optimal resource allocation for the frontend services.


TargetTracking-eks-standard-workers-2cc971d2-732a-1b03-a55a-7d1ae48cc7c7-AlarmHigh-133a8ff8-e59b-4f91-9269-1db7bcb5a740
View
Actions
Graph
1h 3h 12h 1d 3d 1w Custom
UTC timezone
CapacityProviderReservation
CapacityProviderReservation > 100 for 1 datapoints within 1 minute
OK
Percent
101
100
99
19:00 22:00 01:00 04:00 08:00 11:00 14:00
CapacityProviderReservation
Click timeline to see the state change at the selected time.
19:00 22:00 1:00 4:00 8:00 11:00 14:00
In alarm
OK
Insufficient data
Disabled actions

Details

Name
TargetTracking-eks-standard-workers-2cc971d2-732a-1b03-a55a-7d1ae48cc7c7-AlarmHigh-133a8ff8-e59b-4f91-9269-1db7bcb5a740

Type
Metric alarm

Description
DO NOT EDIT OR DELETE. For TargetTrackingScaling policy arn:aws:autoscaling:us-east-1:463470948847:scalingPolicy:200ae407-1587-4cee-ba72-fbc460d56960:autoScalingGroupName/eks-standard-workers-2cc971d2-732a-1b03-a55a-7d1ae48cc7c7:policyName/ECSManagedAuto

State
OK

Threshold
CapacityProviderReservation > 100 for 1 datapoints within 1 minute

Last state update
2024-11-01 19:13:38 (UTC)

Actions
Actions enabled

Namespace
AWS/ECS/ManagedScaling

Metric name
CapacityProviderReservation

CapacityProviderName
Infra-ECS-Cluster-doctor-office-frontend-cluster-e9f3603c-EC2CapacityProvider-MW1WwUEz4vKw

ClusterName
doctor-office-frontend-cluster

Statistic
Average

Period
1 minute

Datapoints to alarm
1 out of 1

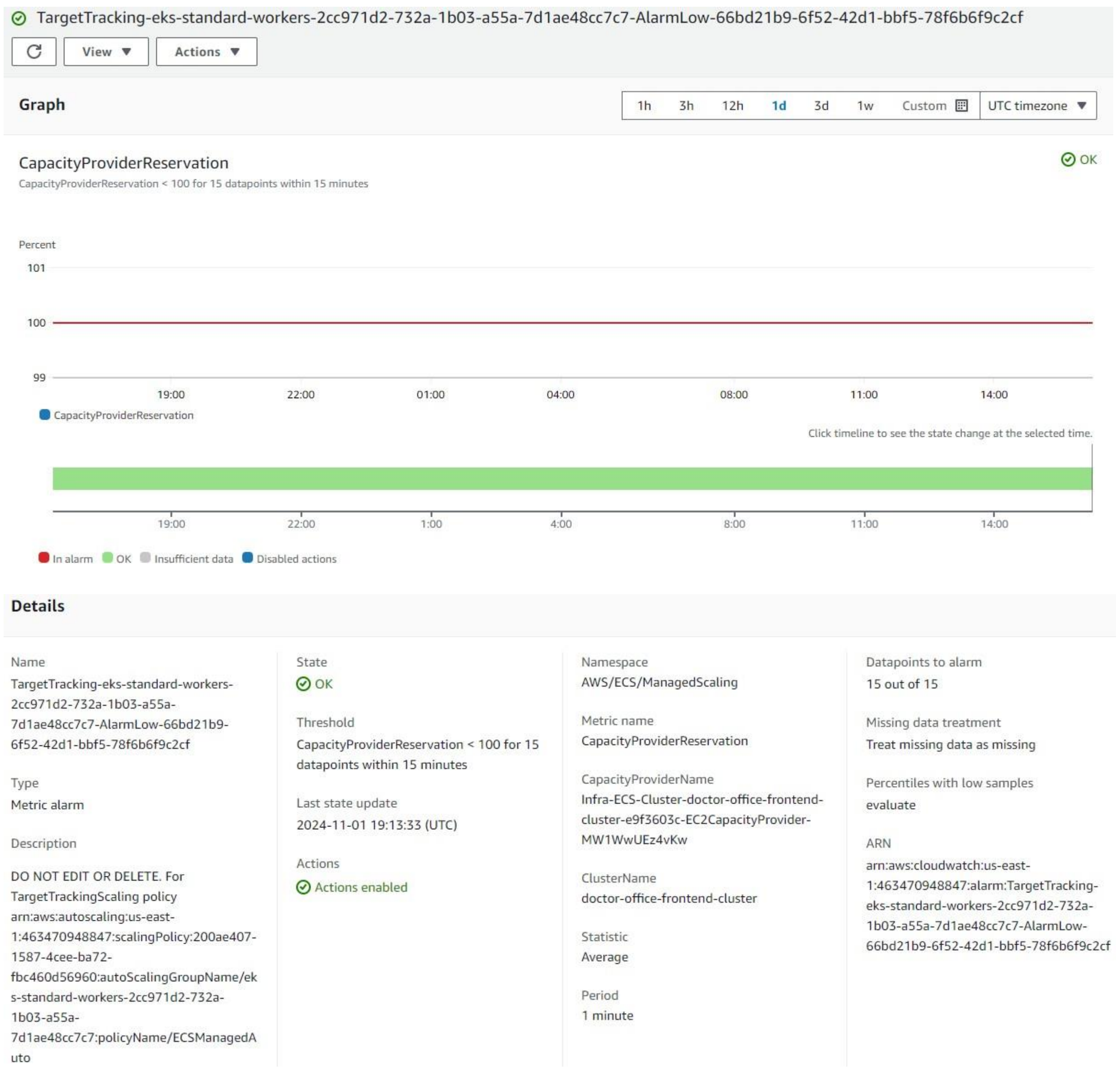
Missing data treatment
Treat missing data as missing

Percentiles with low samples
evaluate

ARN
arn:aws:cloudwatch:us-east-1:463470948847:alarm:TargetTracking-eks-standard-workers-2cc971d2-732a-1b03-a55a-7d1ae48cc7c7-AlarmHigh-133a8ff8-e59b-4f91-9269-1db7bcb5a740

7.2.10 Alarm for monitoring the lower threshold of Capacity Provider Reservation for the EKS cluster's frontend infrastructure

This alarm monitors the lower threshold of Capacity Provider Reservation for the EKS cluster's frontend infrastructure but with different alerting criteria. The alarm alerts when CapacityProviderReservation is less than 100% for 15 data points within 15 minutes. This dashboard is important as it monitors for sustained periods of underutilization.



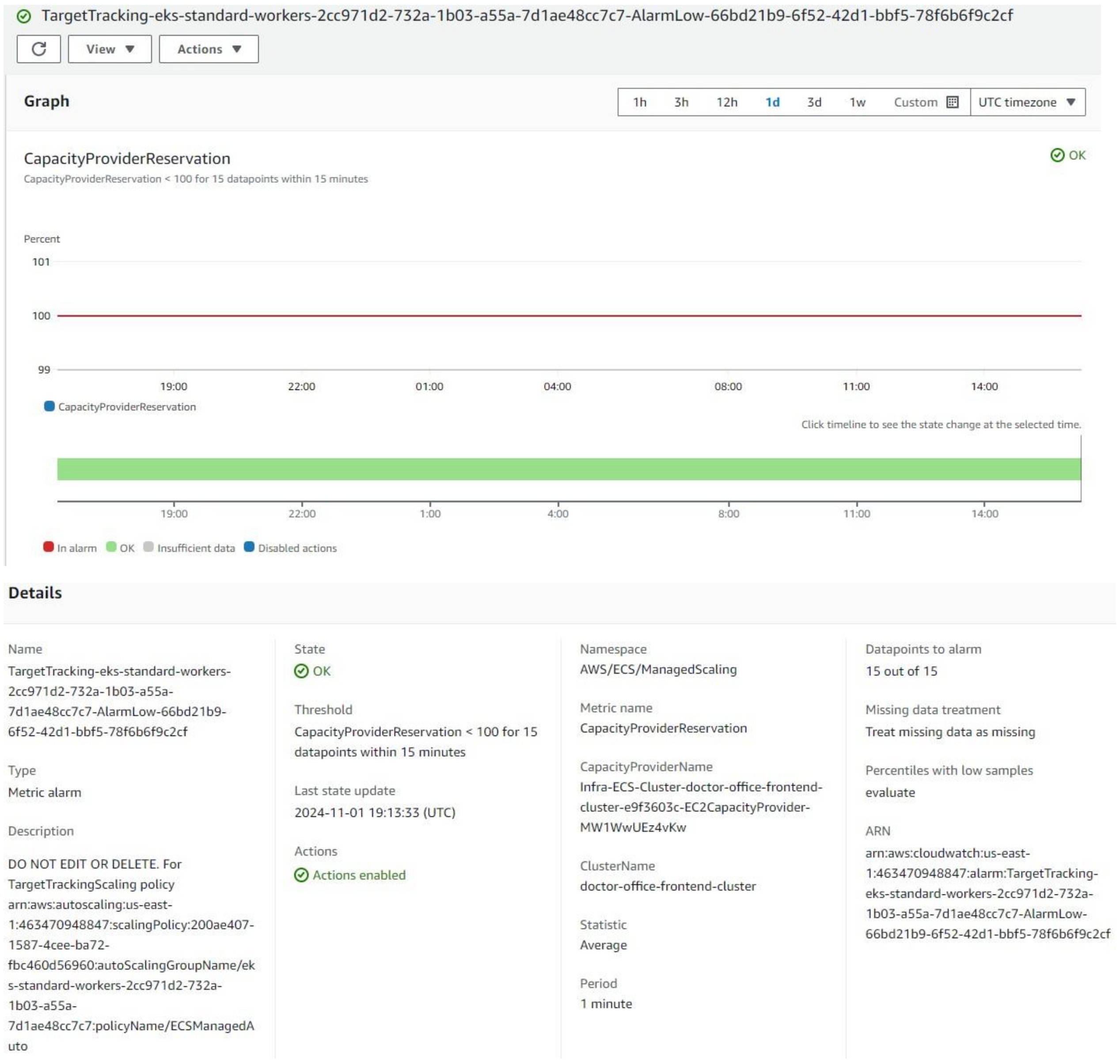
TargetTracking-eks-standard-workers-2cc971d2-732a-1b03-a55a-7d1ae48cc7c7-AlarmLow-66bd21b9-6f52-42d1-bbf5-78f6b6f9c2cf
View
Actions
Graph
1h 3h 12h 1d 3d 1w Custom
UTC timezone
CapacityProviderReservation
CapacityProviderReservation < 100 for 15 datapoints within 15 minutes
OK
Percent
101
100
99
19:00 22:00 01:00 04:00 08:00 11:00 14:00
CapacityProviderReservation
Click timeline to see the state change at the selected time.
19:00 22:00 1:00 4:00 8:00 11:00 14:00
In alarm OK Insufficient data Disabled actions

Details

Name TargetTracking-eks-standard-workers-2cc971d2-732a-1b03-a55a-7d1ae48cc7c7-AlarmLow-66bd21b9-6f52-42d1-bbf5-78f6b6f9c2cf	State ✓ OK	Namespace AWS/ECS/ManagedScaling	Datapoints to alarm 15 out of 15
Type Metric alarm	Threshold CapacityProviderReservation < 100 for 15 datapoints within 15 minutes	Metric name CapacityProviderReservation	Missing data treatment Treat missing data as missing
Description DO NOT EDIT OR DELETE. For TargetTrackingScaling policy arn:aws:autoscaling:us-east-1:463470948847:scalingPolicy:200ae407-1587-4cee-ba72-fbc460d56960:autoScalingGroupName/eks-standard-workers-2cc971d2-732a-1b03-a55a-7d1ae48cc7c7:policyName/ECSManagedAuto	Last state update 2024-11-01 19:13:33 (UTC)	CapacityProviderName Infra-ECS-Cluster-doctor-office-frontend-cluster-e9f3603c-EC2CapacityProvider-MW1WwUEz4vKw	Percentiles with low samples evaluate
	Actions ✓ Actions enabled	ClusterName doctor-office-frontend-cluster	ARN arn:aws:cloudwatch:us-east-1:463470948847:alarm:TargetTracking-eks-standard-workers-2cc971d2-732a-1b03-a55a-7d1ae48cc7c7-AlarmLow-66bd21b9-6f52-42d1-bbf5-78f6b6f9c2cf
		Statistic Average	
		Period 1 minute	

7.2.11 Alarm for monitoring the EKS cluster's capacity provider reservation

This alarm is monitoring an EKS (Elastic Kubernetes Service) cluster's capacity provider reservation, which is a critical metric for container orchestration and auto-scaling. The threshold here is set to alert when CapacityProviderReservation < 100 for 15 data points within 15 minutes.


TargetTracking-eks-standard-workers-2cc971d2-732a-1b03-a55a-7d1ae48cc7c7-AlarmLow-66bd21b9-6f52-42d1-bbf5-78f6b6f9c2cf
View
Actions
Graph
1h
3h
12h
1d
3d
1w
Custom
UTC timezone
CapacityProviderReservation
OK
Percent
101
100
99
19:00
22:00
01:00
04:00
08:00
11:00
14:00
CapacityProviderReservation
Click timeline to see the state change at the selected time.
In alarm
OK
Insufficient data
Disabled actions

CapacityProviderReservation < 100 for 15 datapoints within 15 minutes

Details

Name
TargetTracking-eks-standard-workers-2cc971d2-732a-1b03-a55a-7d1ae48cc7c7-AlarmLow-66bd21b9-6f52-42d1-bbf5-78f6b6f9c2cf

Type
Metric alarm

Description
DO NOT EDIT OR DELETE. For TargetTrackingScaling policy arn:aws:autoscaling:us-east-1:463470948847:scalingPolicy:200ae407-1587-4cee-ba72-fbc460d56960:autoScalingGroupName/eks-standard-workers-2cc971d2-732a-1b03-a55a-7d1ae48cc7c7:policyName/ECSManagedAuto

State
OK

Threshold
CapacityProviderReservation < 100 for 15 datapoints within 15 minutes

Last state update
2024-11-01 19:13:33 (UTC)

Actions
Actions enabled

Namespace
AWS/ECS/ManagedScaling

Metric name
CapacityProviderReservation

CapacityProviderName
Infra-ECS-Cluster-doctor-office-frontend-cluster-e9f3603c-EC2CapacityProvider-MW1WwUEz4vKw

ClusterName
doctor-office-frontend-cluster

Statistic
Average

Period
1 minute

Datapoints to alarm
15 out of 15

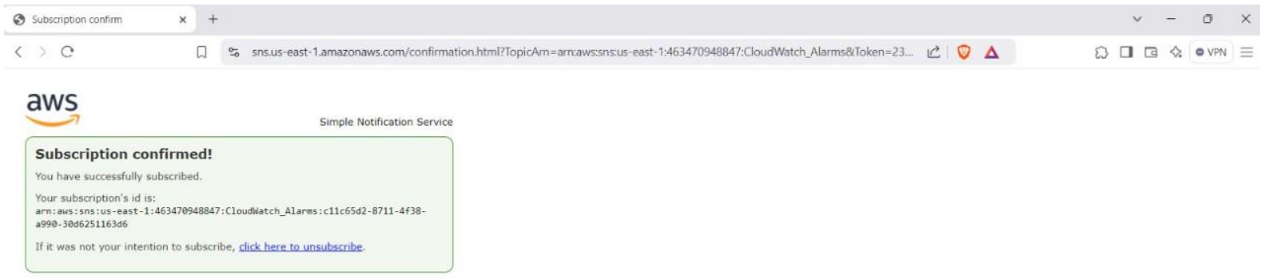
Missing data treatment
Treat missing data as missing

Percentiles with low samples
evaluate

ARN
arn:aws:cloudwatch:us-east-1:463470948847:alarm:TargetTracking-eks-standard-workers-2cc971d2-732a-1b03-a55a-7d1ae48cc7c7-AlarmLow-66bd21b9-6f52-42d1-bbf5-78f6b6f9c2cf

7.3 AWS Simple Notification Service (SNS) Subscription Configuration

The screenshot below shows the successful confirmation of a subscription to Amazon SNS.



Subscription confirm

sns.us-east-1.amazonaws.com/confirmation.html?TopicArn=arn:aws:sns:us-east-1:463470948847:CloudWatch_Alarms&Token=23...

Simple Notification Service

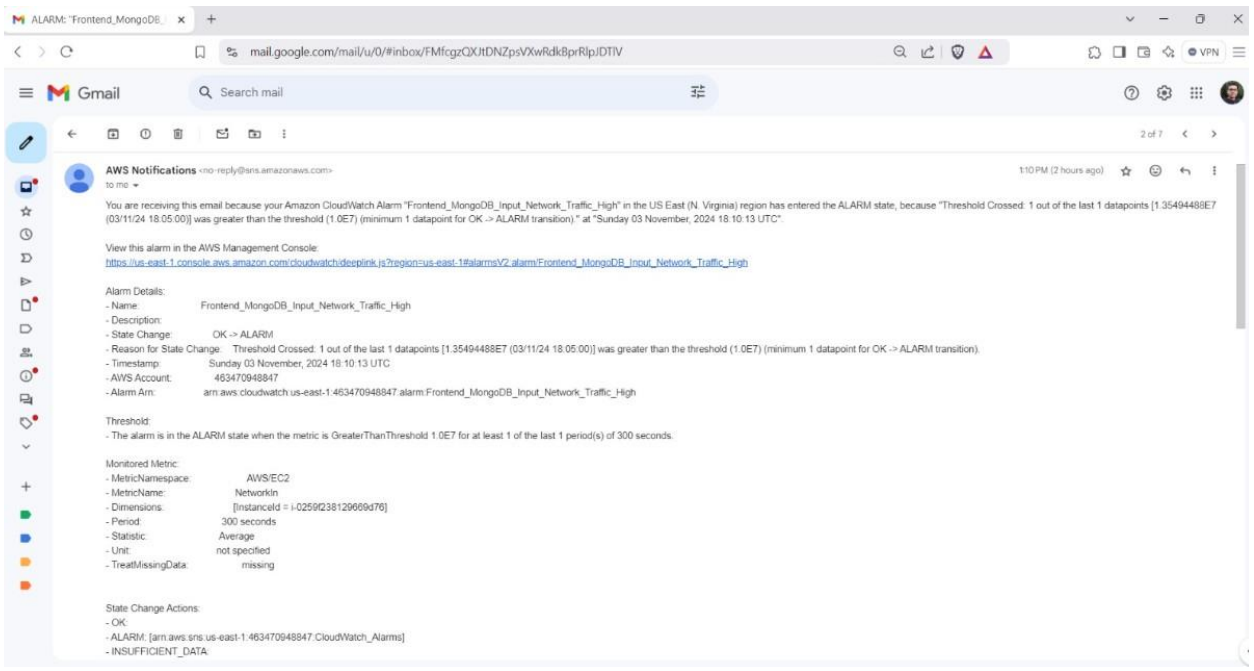
Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:
`arn:aws:sns:us-east-1:463470948847:CloudWatch_Alarms:c11c65d2-8711-4f38-a990-30d6251163d6`

If it was not your intention to subscribe, click here to unsubscribe.

The screenshot below demonstrates a CloudWatch alarm notification received via email through Amazon SNS. This alarm, titled "Frontend_MongoDB_Input_Network_Traffic_High," was triggered because the monitored metric exceeded the defined threshold. The email provides details such as the alarm name, state change, timestamp, and monitored metric details, confirming that the SNS subscription is correctly forwarding CloudWatch alarm notifications to the subscribed email address.



ALARM: "Frontend_MongoDB_...

mail.google.com/mail/u/0/#inbox/FMfcgzQXJtDNZpsVXwRdkBprRlpJDTlV

AWS Notifications <no-reply@sns.amazonaws.com>
to me

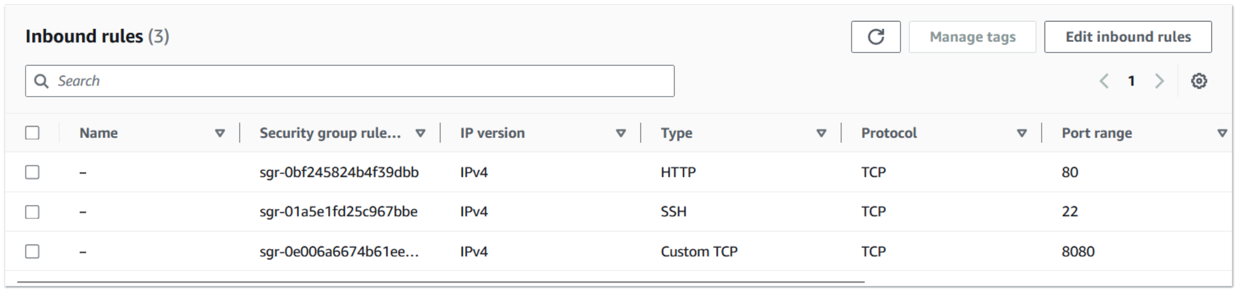
1:10 PM (2 hours ago)

```
You are receiving this email because your Amazon CloudWatch Alarm "Frontend_MongoDB_Input_Network_Traffic_High" in the US East (N. Virginia) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [1.35494488E7 (03/11/24 18:05:00)] was greater than the threshold (1.0E7) (minimum 1 datapoint for OK -> ALARM transition)." at "Sunday 03 November, 2024 18:10:13 UTC".

View this alarm in the AWS Management Console:
https://us-east-1.console.aws.amazon.com/cloudwatch/deeplink.js?region=us-east-1#alarmsV2:alarm/Frontend_MongoDB_Input_Network_Traffic_High

Alarm Details:
- Name:                     Frontend_MongoDB_Input_Network_Traffic_High
- Description:
- State Change:             OK -> ALARM
- Reason for State Change:  Threshold Crossed: 1 out of the last 1 datapoints [1.35494488E7 (03/11/24 18:05:00)] was greater than the threshold (1.0E7) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp:                Sunday 03 November, 2024 18:10:13 UTC
- AWS Account:              463470948847
- Alarm Arn:                arn:aws:cloudwatch:us-east-1:463470948847:alarm:Frontend_MongoDB_Input_Network_Traffic_High

Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanThreshold 1.0E7 for at least 1 of the last 1 period(s) of 300 seconds.

Monitored Metric:
- MetricNamespace:          AWS/EC2
- MetricName:               NetworkIn
- Dimensions:               [InstanceId = i-0259f238129669d76]
- Period:                   300 seconds
- Statistic:                Average
- Unit:                     not specified
- TreatMissingData:         missing


State Change Actions:
- OK:
- ALARM: [arn:aws:sns:us-east-1:463470948847:CloudWatch_Alarms]
- INSUFFICIENT_DATA:
```

8. CI/CD Pipeline:

8.1 AWS Setup

To set up a CI/CD pipeline, first, create a security group that allows inbound HTTP access and inbound SSH access from the ip address of the machine used for setup. Set the source for the HTTP and TCP to 0.0.0.0/0 and the source for SSH to <Computer IP Address>/32.



Name	Security group rule...	IP version	Type	Protocol	Port range
–	sgr-0bf245824b4f39dbb	IPv4	HTTP	TCP	80
–	sgr-01a5e1fd25c967bbe	IPv4	SSH	TCP	22
–	sgr-0e006a6674b61ee...	IPv4	Custom TCP	TCP	8080

Then create a key pair to connect the instance securely through SSH. To give the server the ability to modify AWS resources, create an IAM role with the "AmazonEC2ContainerRegistryFullAccess" policy. Finally, launch an Amazon Linux EC2 instance with a previously created security group, role, and access key.

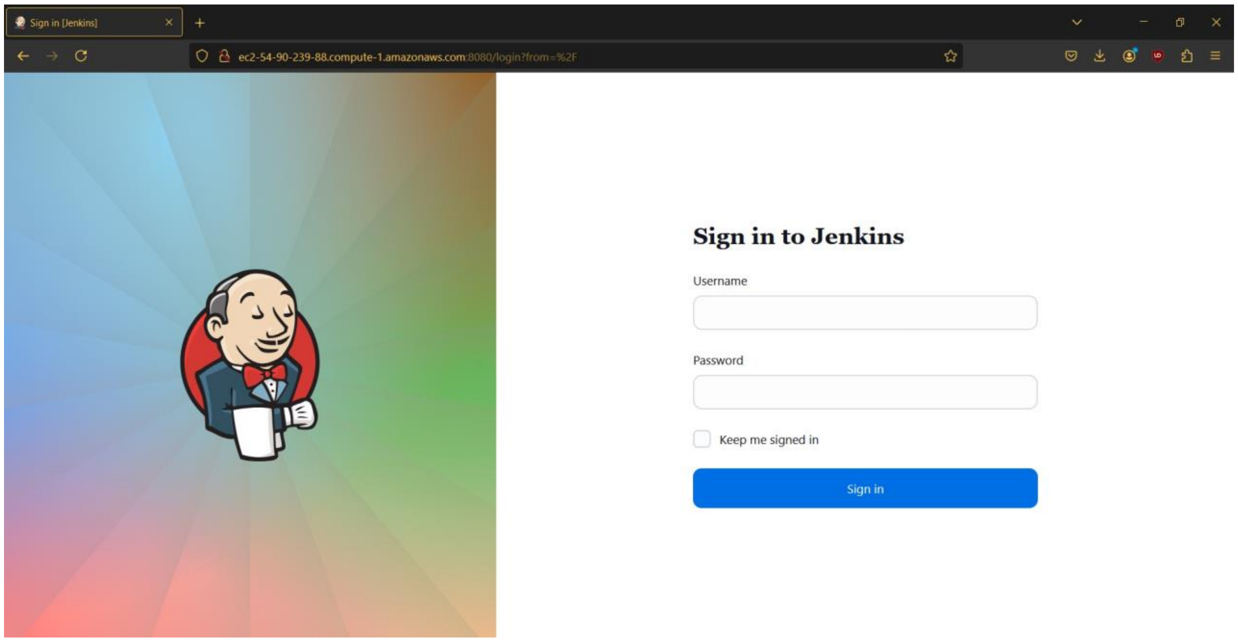
8.2 Jenkins Setup

SSH into the EC2 instance and run the following commands to install Jenkins, docker, git, and kubectl.

```
sudo yum update –y
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
sudo yum upgrade
sudo amazon-linux-extras install epel
sudo yum install DNF
sudo dnf install java-17-amazon-corretto -y
sudo yum install jenkins -y
sudo systemctl enable jenkins
sudo systemctl start jenkins

sudo amazon-linux-extras install docker
sudo systemctl enable docker
sudo systemctl start docker
sudo usermod -a -G docker ec2-user
sudo chmod 666 /var/run/docker.sock

sudo yum install git-all
```

```
curl -O
https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.0/2024-09-12/bin/linux/arm64/kubectl
chmod +x ./kubectl
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export
PATH=$HOME/bin:$PATH
```

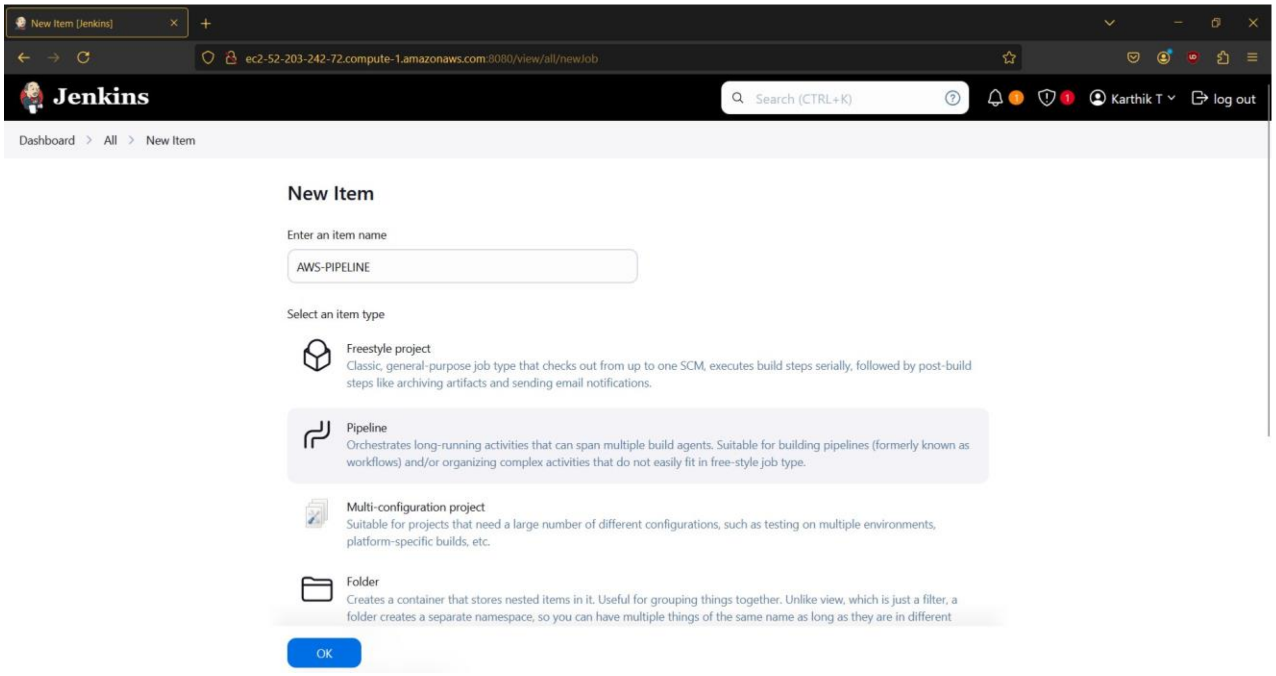
Now visit http://<EC2 Instance Public IPv4 DNS>:8080 in a web browser and it will ask for an initial admin password to unlock the jenkins server which can be found with "sudo cat /var/lib/jenkins/secrets/initialAdminPassword". After unlocking, follow the provided instructions to create an administrator account for the Jenkins Server. Once that's complete a sign-in page will be available and use the credentials to log in.



After logging in, go to "Manage Jenkins">"Plugins">"Available plugins" and install the following plugins if they aren't already installed: docker, docker pipeline, Amazon EC2, Git, and GitHub.

8.3 Pipeline Creation

In the Jenkins dashboard, select “New Item”, then name the pipeline and select “Pipeline”.


New Item [Jenkins]
ec2-52-203-242-72.compute-1.amazonaws.com:8080/view/all/newJob
Jenkins
Search (CTRL+K)
Karthik T
log out
Dashboard > All > New Item

New Item

Enter an item name

AWS-PIPELINE

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

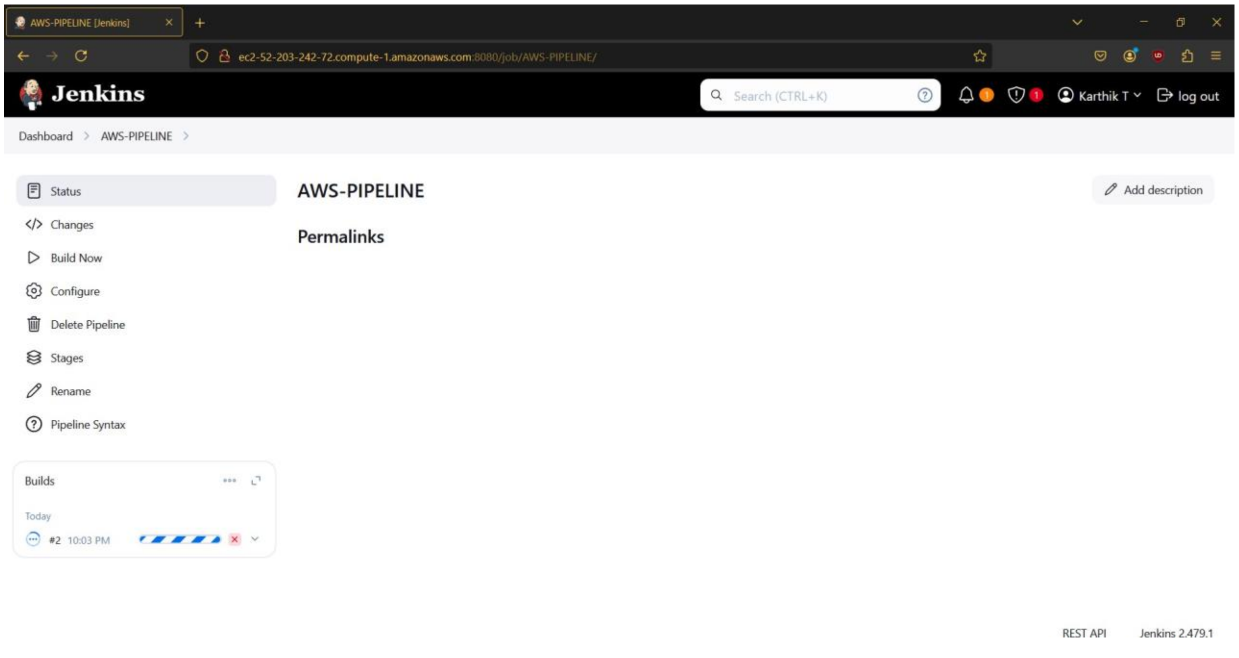
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different

OK

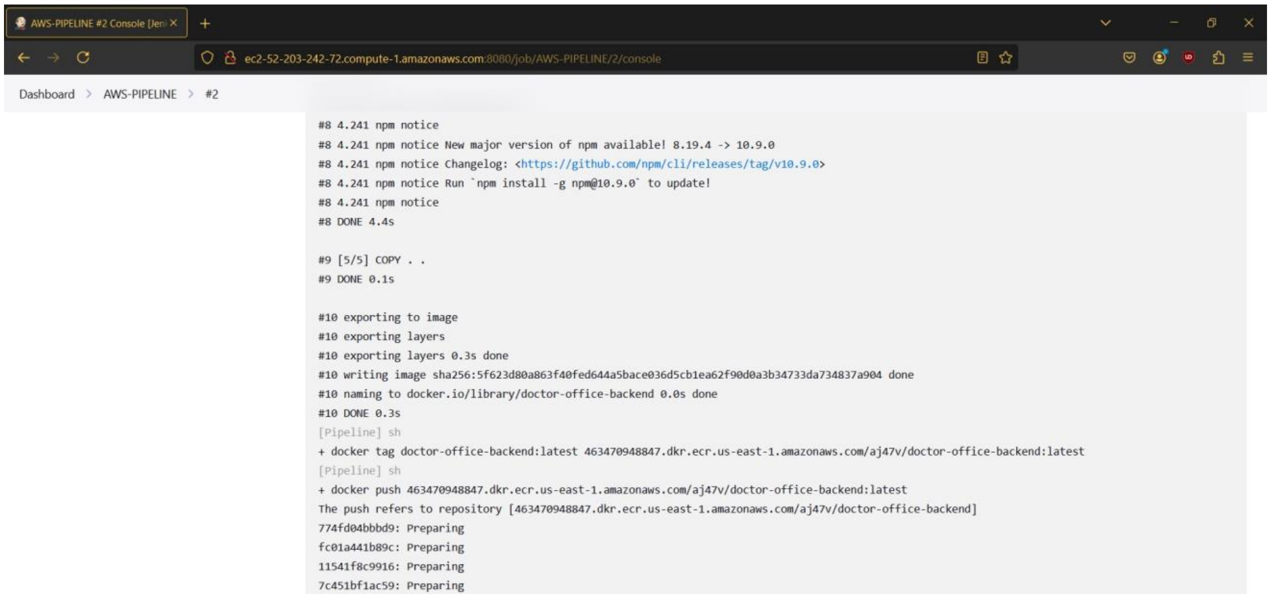
After pressing OK, scroll to the end of the next page, and leave the default setting and then under “Definition” select “Pipeline Script” and paste the following script. Make any necessary adjustments to the environment variables or git repository link in the script.

```
pipeline {
agent any
environment {
AWS_REGION = 'us-east-1'
ECR_BACKEND_REPO = 'doctor-office-backend'
ECR_FRONTEND_REPO = 'doctor-office-frontend'
FRONTEND_PATH = './frontend'
BACKEND_PATH = './backend'
KUBERNETES_PATH = './kubernetes'
NAMESPACE = 'aj47v'
AWS_ACCOUNT_ID = '463470948847'
        REGISTRY_URL =
"${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com"
}
stages {
stage('SCM Checkout') {
steps {
git branch: 'main', url: 'https://github.com/DevanshhN/k8s-doctor-office.git'
}
}
stage('AWS Authentication') {
steps {
sh "aws ecr get-login-password --region ${AWS_REGION} | docker login --username AWS
--password-stdin ${REGISTRY_URL}"
}
}
stage('Build and Push Backend Docker Image') {
steps {
script {
sh "docker build -t $ECR_BACKEND_REPO $BACKEND_PATH"
sh "docker tag $ECR_BACKEND_REPO:latest
${REGISTRY_URL}/${NAMESPACE}/${ECR_BACKEND_REPO}:latest"
sh "docker push ${REGISTRY_URL}/${NAMESPACE}/${ECR_BACKEND_REPO}:latest"
}
}
}
stage('Build and Push Frontend Docker Image') {
steps {
script {
sh "docker build -t $ECR_FRONTEND_REPO $FRONTEND_PATH"
sh "docker tag $ECR_FRONTEND_REPO:latest
${REGISTRY_URL}/${NAMESPACE}/${ECR_FRONTEND_REPO}:latest"
sh "docker push ${REGISTRY_URL}/${NAMESPACE}/${ECR_FRONTEND_REPO}:latest"
}
}
}
stage('Apply Kubernetes Manifests') {
steps {
script {
sh "kubectl apply -f ${KUBERNETES_PATH}/backend-deployment.yaml"
sh "kubectl apply -f ${KUBERNETES_PATH}/frontend-deployment.yaml"
}
}
}
}
}
```

Now click on the newly created pipeline in the dashboard and then select “Build Now”. A build should appear in the bottom left corner.

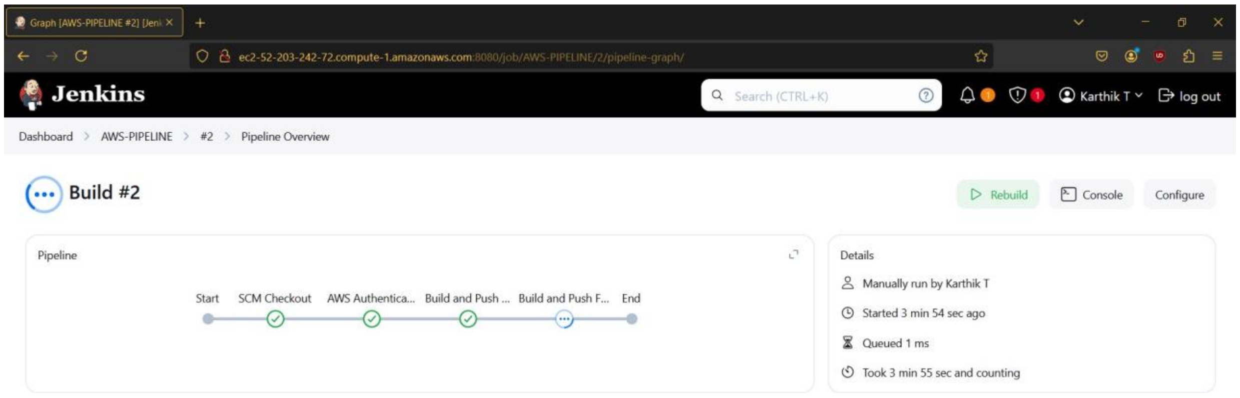


Click on the build and select "Console Output" to see logs of the commands to ensure the steps are running correctly.


AWS-PIPELINE #2 Console [Jen
ec2-52-203-242-72.compute-1.amazonaws.com:8080/job/AWS-PIPELINE/2/console
Dashboard > AWS-PIPELINE > #2

```
#8 4.241 npm notice
#8 4.241 npm notice New major version of npm available! 8.19.4 -> 10.9.0
#8 4.241 npm notice Changelog: <https://github.com/npm/cli/releases/tag/v10.9.0>
#8 4.241 npm notice Run `npm install -g npm@10.9.0` to update!
#8 4.241 npm notice
#8 DONE 4.4s

#9 [5/5] COPY . .
#9 DONE 0.1s

#10 exporting to image
#10 exporting layers
#10 exporting layers 0.3s done
#10 writing image sha256:5f623d80a863f40fed644a5bace036d5cb1ea62f90d0a3b34733da734837a904 done
#10 naming to docker.io/library/doctor-office-backend 0.0s done
#10 DONE 0.3s
[Pipeline] sh
+ docker tag doctor-office-backend:latest 463470948847.dkr.ecr.us-east-1.amazonaws.com/aj47v/doctor-office-backend:latest
[Pipeline] sh
+ docker push 463470948847.dkr.ecr.us-east-1.amazonaws.com/aj47v/doctor-office-backend:latest
The push refers to repository [463470948847.dkr.ecr.us-east-1.amazonaws.com/aj47v/doctor-office-backend]
774fd04bbbd9: Preparing
fc01a441b89c: Preparing
11541f8c9916: Preparing
7c451bf1ac59: Preparing
```

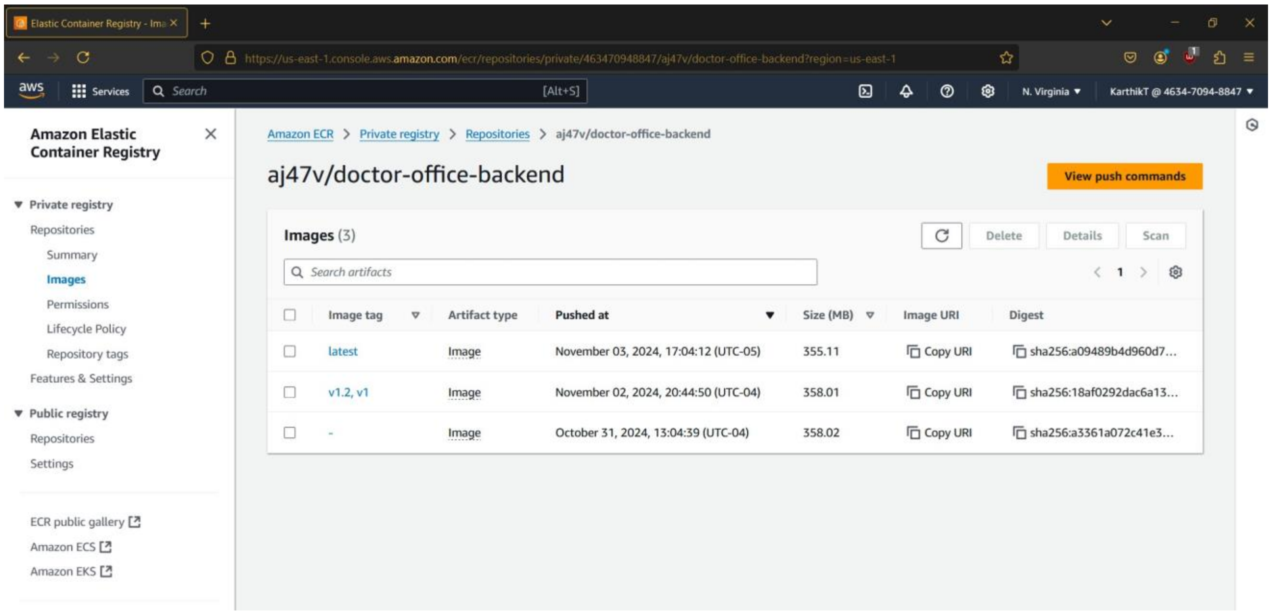
"Pipeline Overview" shows the status of each step in the build.


Graph [AWS-PIPELINE #2] [Jenk
ec2-52-203-242-72.compute-1.amazonaws.com:8080/job/AWS-PIPELINE/2/pipeline-graph/
Jenkins
Search (CTRL+K)
Karthik T
log out
Dashboard > AWS-PIPELINE > #2 > Pipeline Overview
Rebuild
Console
Configure
Pipeline
Start
SCM Checkout
AWS Authentica...
Build and Push ...
Build and Push F...
End

Build #2

Details
- Manually run by Karthik T
- Started 3 min 54 sec ago
- Queued 1 ms
- Took 3 min 55 sec and counting

As shown above, the backend image has been built and pushed and this can be verified by checking ECR in AWS.


Elastic Container Registry - Ima
https://us-east-1.console.aws.amazon.com/ecr/repositories/private/463470948847/aj47v/doctor-office-backend?region=us-east-1
aws
Services
Search
[Alt+S]
N. Virginia
KarthikT @ 4634-7094-8847
Amazon Elastic Container Registry
Private registry
Repositories
Summary
Images
Permissions
Lifecycle Policy
Repository tags
Features & Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
Amazon ECR > Private registry > Repositories > aj47v/doctor-office-backend
View push commands
Delete
Details
Scan
Search artifacts

aj47v/doctor-office-backend

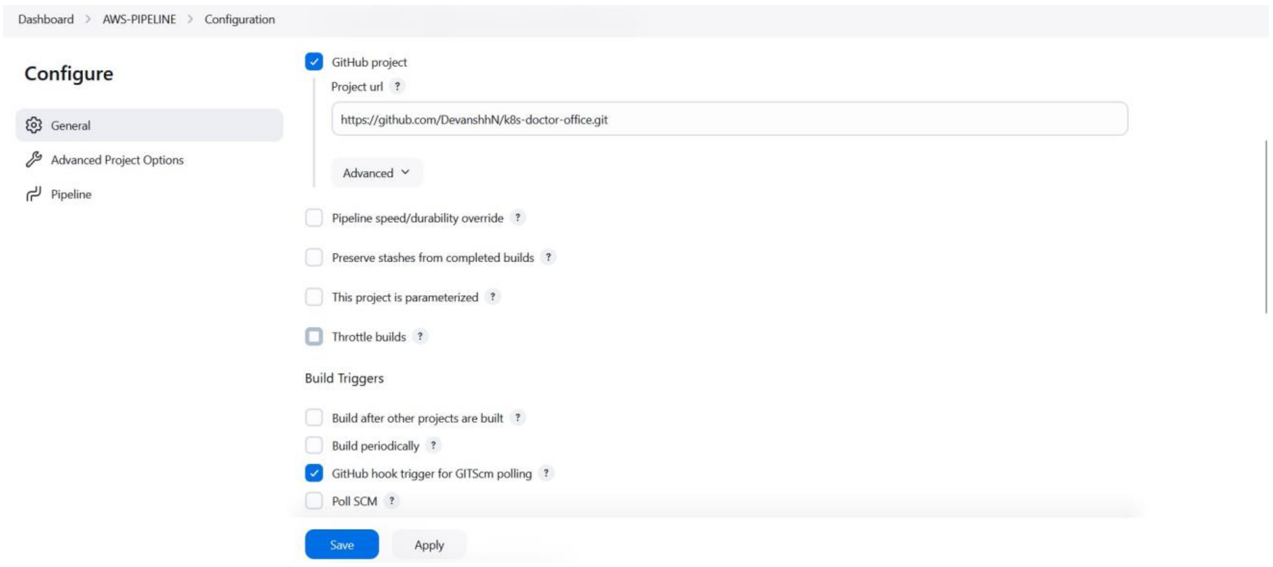
Images (3)

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
latest	Image	November 03, 2024, 17:04:12 (UTC-05)	355.11	Copy URI	sha256:a09489b4d960d7...
v1.2, v1	Image	November 02, 2024, 20:44:50 (UTC-04)	358.01	Copy URI	sha256:18af0292dac6a13...
-	Image	October 31, 2024, 13:04:39 (UTC-04)	358.02	Copy URI	sha256:a3361a072c41e3...

8.4 Automating with GitHub Webhooks

Currently running this pipeline requires someone to log into the Jenkins instance and manually start the build. To speed up development we can automate this by making a build trigger any time a push is made to the git repository.

Select the pipeline in the dashboard and press “Configure”, then select GitHub project and paste the project URL and select “GitHub hook trigger for GITScm polling” and press save.


Dashboard > AWS-PIPELINE > Configuration

Configure

General
Advanced Project Options
Pipeline

☑ GitHub project
Project url
https://github.com/DevanshhN/k8s-doctor-office.git

Advanced

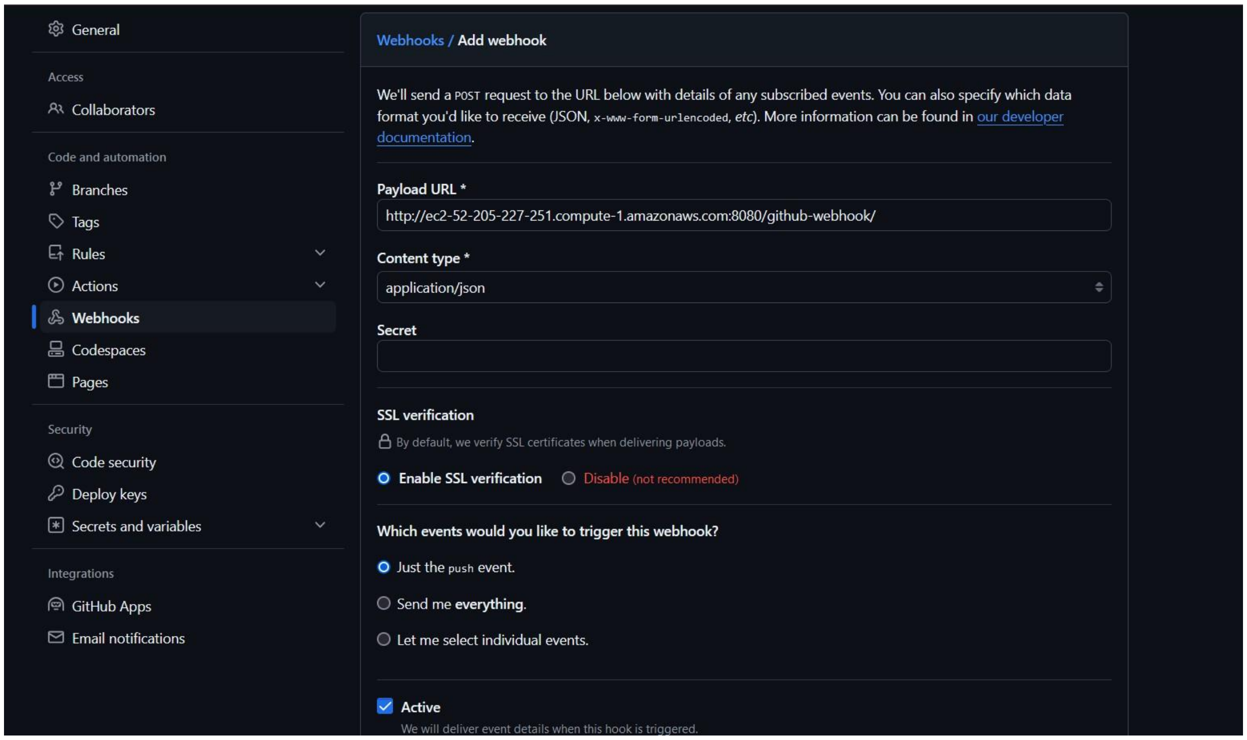
☐ Pipeline speed/durability override
☐ Preserve stashes from completed builds
☐ This project is parameterized
☐ Throttle builds

Build Triggers

☐ Build after other projects are built
☐ Build periodically
☑ GitHub hook trigger for GITScm polling
☐ Poll SCM

Save Apply

Open the GitHub repository in the web browser, navigate to the settings, and select "Webhooks" and press "Add webhook" on the page. For the Payload URL put "http://<Jenkins Server Adress>/github-webhook/" and then change the Content type to "application/json" and select "Just the push event." for the event to trigger the webhook. You will need administrator access to the git repository to perform this task.


General

Access
Collaborators

Code and automation
Branches
Tags
Rules
Actions
Webhooks
Codespaces
Pages

Security
Code security
Deploy keys
Secrets and variables

Integrations
GitHub Apps
Email notifications

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, *etc*). More information can be found in our developer documentation.

Payload URL *
http://ec2-52-205-227-251.compute-1.amazonaws.com:8080/github-webhook/

Content type *
application/json

Secret

SSL verification
By default, we verify SSL certificates when delivering payloads.
● Enable SSL verification ○ Disable (not recommended)

Which events would you like to trigger this webhook?
● Just the push event.
○ Send me everything.
○ Let me select individual events.

☑ Active
We will deliver event details when this hook is triggered.

Now, whenever a push is made to the repository a build should start in Jenkins.

Link to the Github Repository: https://github.com/DevanshhN/k8s-doctor-office

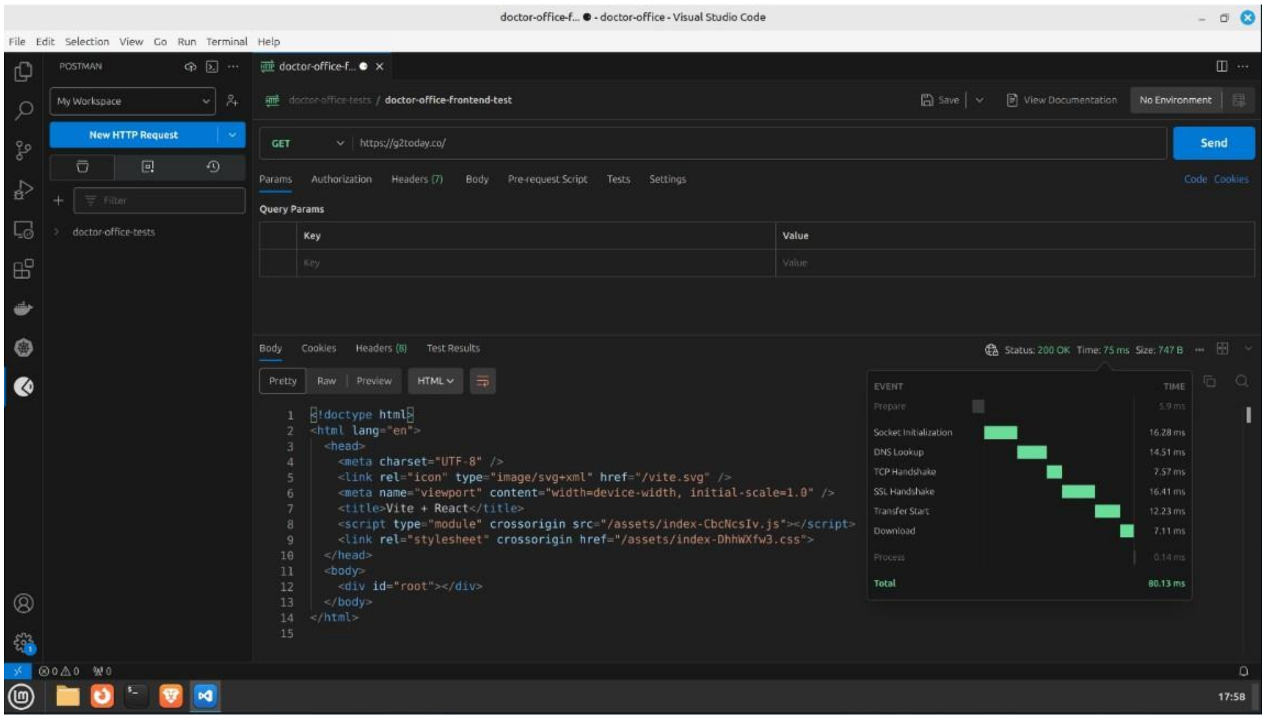
9. Testing and Debugging:

9.1 Frontend UI Testing:

The frontend UI test of https://g2today.co/ demonstrates a web application built with Vite and React. The server responded with a 200 OK status and delivered a lightweight payload of 747 bytes in a total response time of 80.13ms, indicating good performance. The request timeline shows reasonable performance across all network phases, with the longest times being socket initialization (16.28 ms) and DNS lookup (14.51 ms). The HTML structure follows modern best practices, implementing proper DOCTYPE, character encoding, and mobile-responsive viewport settings.

- Versioned assets (using hash in filenames) show proper cache management.
  src="/assets/index-CbcNcsIv.js"
  href="/assets/index-DhhWXfw3.css"

  The -CbcNcsIv and -DhhWXfw3 in the filenames are hash strings. These are automatically generated during the build process to ensure proper cache busting when files change.


doctor-office-f... ● - doctor-office - Visual Studio Code
File Edit Selection View Go Run Terminal Help
POSTMAN
My Workspace
New HTTP Request
doctor-office-tests
doctor-office-tests / doctor-office-frontend-test
Save
View Documentation
No Environment
Send
Params Authorization Headers (7) Body Pre-request Script Tests Settings
Code Cookies
Query Params
Body Cookies Headers (8) Test Results
Pretty Raw Preview HTML
17:58

GET https://g2today.co/

Status: 200 OK Time: 75 ms Size: 747 B

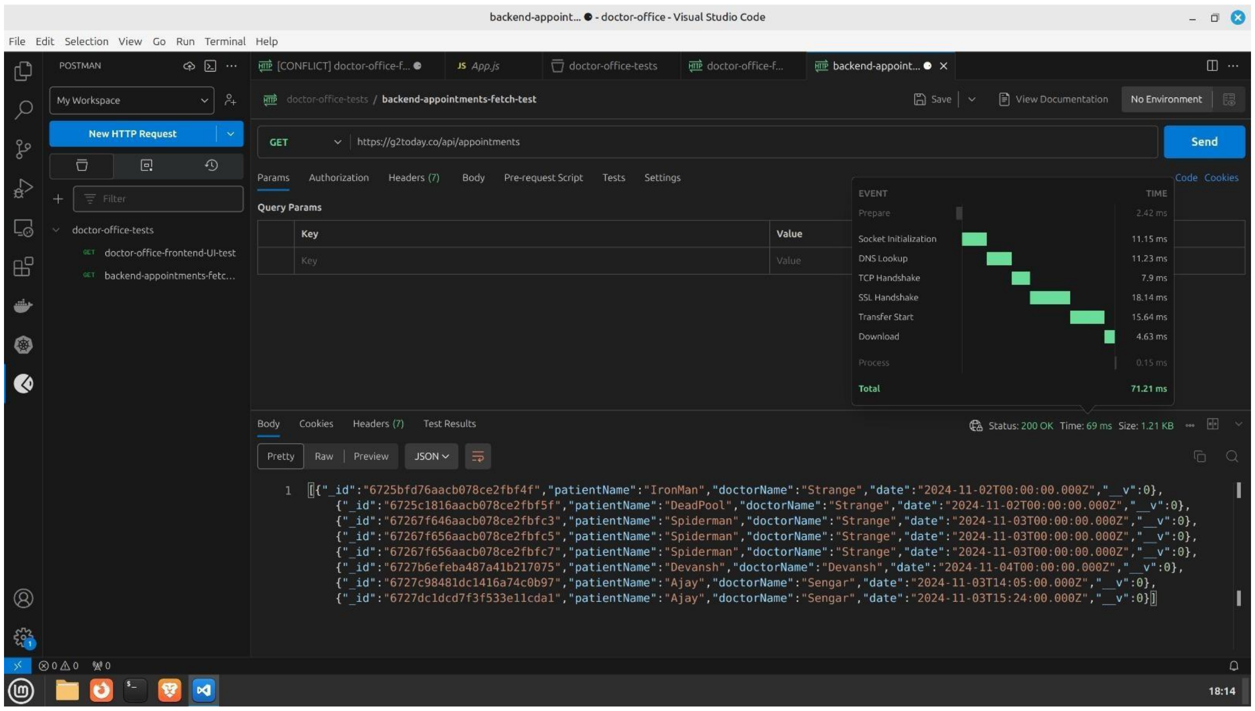
```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React</title>
    <script type="module" crossorigin src="/assets/index-CbcNcsIv.js"></script>
    <link rel="stylesheet" crossorigin href="/assets/index-DhhWXfw3.css">
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

EVENT	TIME
Prepare	5.9 ms
Socket Initialization	16.28 ms
DNS Lookup	14.51 ms
TCP Handshake	7.57 ms
SSL Handshake	16.41 ms
Transfer Start	12.23 ms
Download	7.11 ms
Process	0.14 ms
Total	80.13 ms

9.2 Backend Testing - Appointments Fetch

We have used the Postman tool to send API requests to fetch appointment data for the “Doctor’s Office Application.” To test the backend of the application, we have sent a GET request to the endpoint **https://g2today.co/api/appointments**, which is intended to retrieve appointment data.

This Postman request verifies that the backend endpoint for fetching appointments is working correctly because it is returning a **200 OK** status with valid JSON data.


backend-appoint... ● - doctor-office - Visual Studio Code
File Edit Selection View Go Run Terminal Help
POSTMAN
My Workspace
New HTTP Request
doctor-office-tests
GET doctor-office-frontend-UI-test
GET backend-appointments-fetc...
doctor-office-tests / backend-appointments-fetch-test
Save
View Documentation
No Environment
Send
Params Authorization Headers (7) Body Pre-request Script Tests Settings
Query Params
Key
Value
Body Cookies Headers (7) Test Results
Pretty Raw Preview JSON

GET https://g2today.co/api/appointments

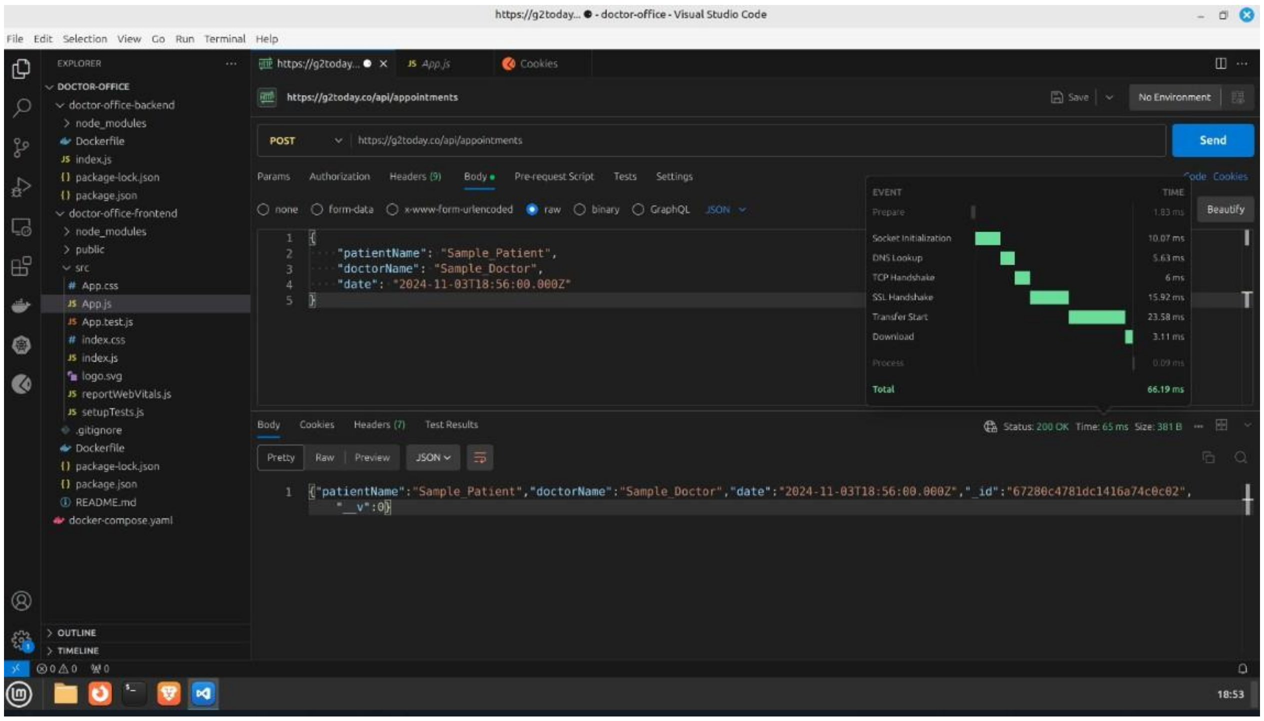
EVENT	TIME
Prepare	2.42 ms
Socket Initialization	11.15 ms
DNS Lookup	11.23 ms
TCP Handshake	7.9 ms
SSL Handshake	18.14 ms
Transfer Start	15.64 ms
Download	4.63 ms
Process	0.15 ms
Total	71.21 ms

Status: 200 OK Time: 69 ms Size: 1.21 KB

```
[{"_id":"6725bfd76aacb078ce2fbf4f","patientName":"IronMan","doctorName":"Strange","date":"2024-11-02T00:00:00.000Z","__v":0},
 {"_id":"6725c1816aacb078ce2fbf5f","patientName":"DeadPool","doctorName":"Strange","date":"2024-11-02T00:00:00.000Z","__v":0},
 {"_id":"67267f646aacb078ce2fbfc3","patientName":"Spiderman","doctorName":"Strange","date":"2024-11-03T00:00:00.000Z","__v":0},
 {"_id":"67267f656aacb078ce2fbfc5","patientName":"Spiderman","doctorName":"Strange","date":"2024-11-03T00:00:00.000Z","__v":0},
 {"_id":"67267f656aacb078ce2fbfc7","patientName":"Spiderman","doctorName":"Strange","date":"2024-11-03T00:00:00.000Z","__v":0},
 {"_id":"6727b6efeba487a41b217075","patientName":"Devansh","doctorName":"Devansh","date":"2024-11-04T00:00:00.000Z","__v":0},
 {"_id":"6727c98481dc1416a74c0b97","patientName":"Ajay","doctorName":"Sengar","date":"2024-11-03T14:05:00.000Z","__v":0},
 {"_id":"6727dc1dcd7f3f533e11cda1","patientName":"Ajay","doctorName":"Sengar","date":"2024-11-03T15:24:00.000Z","__v":0}]
```

9.3 Backend-Testing - Appointment Creation

The appointment creation endpoint at https://g2today.co/api/appointments demonstrates robust functionality with a POST request. The test shows a successful response (200 OK) handling JSON data for appointment scheduling. The response includes a unique appointment ID (67280c4781dc1416a74c0c02), validating proper database insertion. The system successfully processes key appointment details including patient name, doctor name, and timestamp.


POST https://g2today.co/api/appointments

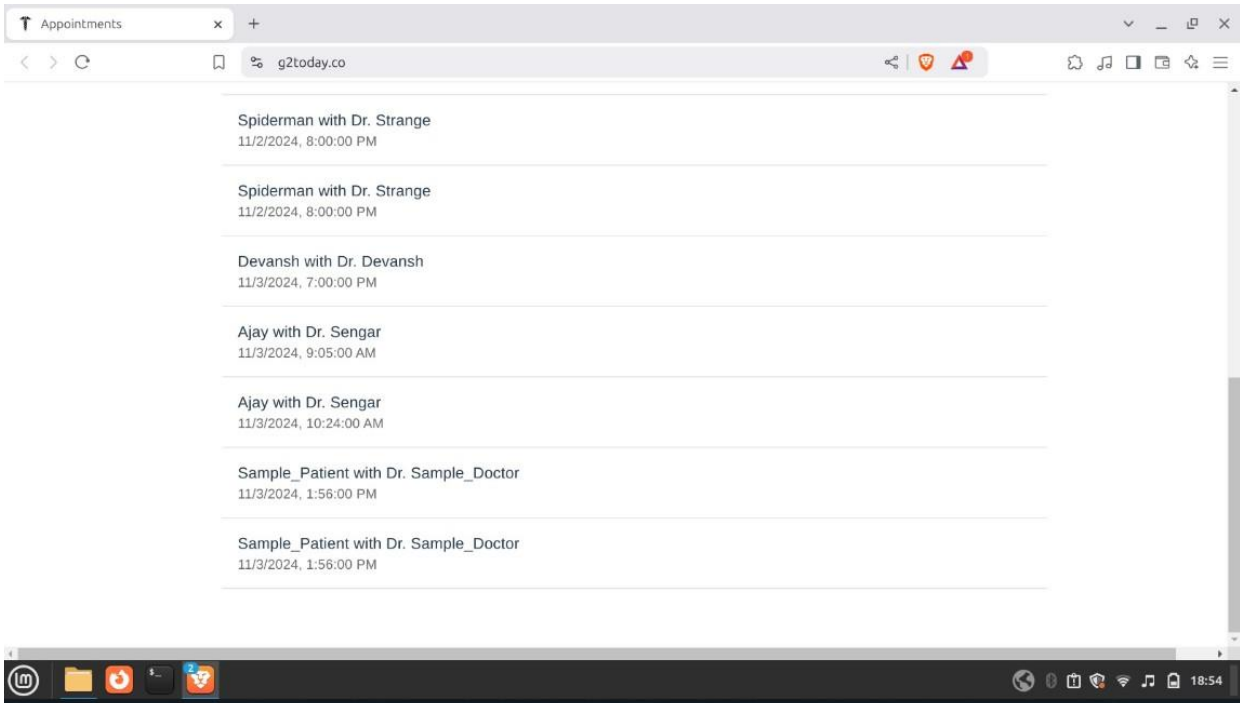
```
{
    "patientName": "Sample_Patient",
    "doctorName": "Sample_Doctor",
    "date": "2024-11-03T18:56:00.000Z"
}
```

EVENT	TIME
Prepare	1.83 ms
Socket Initialization	10.07 ms
DNS Lookup	5.63 ms
TCP Handshake	6 ms
SSL Handshake	15.92 ms
Transfer Start	23.58 ms
Download	3.11 ms
Process	0.09 ms
Total	66.19 ms

Status: 200 OK Time: 65 ms Size: 381 B

```
{"patientName":"Sample_Patient","doctorName":"Sample_Doctor","date":"2024-11-03T18:56:00.000Z","_id":"67280c4781dc1416a74c0c02",
    "__v":0}
```

The image below shows that records were successfully created at the end of the Appointment Booking Page:


Spiderman with Dr. Strange
11/2/2024, 8:00:00 PM

Spiderman with Dr. Strange
11/2/2024, 8:00:00 PM

Devansh with Dr. Devansh
11/3/2024, 7:00:00 PM

Ajay with Dr. Sengar
11/3/2024, 9:05:00 AM

Ajay with Dr. Sengar
11/3/2024, 10:24:00 AM

Sample_Patient with Dr. Sample_Doctor
11/3/2024, 1:56:00 PM

Sample_Patient with Dr. Sample_Doctor
11/3/2024, 1:56:00 PM

10. Conclusion

The development and deployment of the Doctor’s Office Appointment Application demonstrate the advantages of a microservices-based architecture in creating scalable, secure, and reliable applications in the healthcare sector. By leveraging AWS cloud infrastructure and container orchestration via Kubernetes, the application effectively meets the demands of an expanding user base, allowing for seamless management of patient data and appointment scheduling.

The combination of a React front-end, Node.js and Express back-end, and MongoDB database establishes a robust, responsive, and user-friendly system that ensures data integrity and ease of access. Containerization and deployment on Amazon Elastic Kubernetes Service (EKS) have enabled automated scaling and simplified management of application components, allowing resources to dynamically adjust to demand and minimizing downtime.

Security was a primary focus throughout the project. AWS Certificate Manager (ACM) ensures secure communication through SSL/TLS certificates, and IAM role-based access controls limit permissions, safeguarding sensitive healthcare data. Continuous integration and continuous deployment (CI/CD) pipelines, managed through Jenkins, support the rapid delivery of updates, reinforcing both system reliability and efficiency.

Overall, this project highlights the effectiveness of cloud-native solutions in addressing the complexities and regulatory requirements of healthcare applications. The Doctor’s Office Appointment Application exemplifies modern software engineering best practices, aligning with industry standards for performance, scalability, and data security. This project provides a foundation for future enhancements, including the potential for integrating advanced analytics, patient communications, and additional security layers to further improve patient care and operational efficiency.

11. References:

1. https://docs.aws.amazon.com/eks/
2. https://docs.aws.amazon.com/autoscaling/
3. https://kubernetes.io/docs/home/
4. https://docs.aws.amazon.com/cloudwatch/
5. https://docs.aws.amazon.com/iam/
6. https://docs.aws.amazon.com/route53/
7. https://docs.aws.amazon.com/acm/
8. https://youtu.be/_bEPuvrjB5Y?si=2WU4NS-q0G2NFcXi
9. https://www.jenkins.io/doc/
10. https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/AlarmThatSendsEmail.html
11. https://learning.postman.com/docs/sending-requests/requests/