SECTION A: Theory Questions

Q1) Which of the following is an economic benefit of utilizing the cloud?
(A) No up-front capitalized expenditures (capex)

Q2) Which of the following are categories of recommendations provided by AWS Trusted Advisor? (Choose three)
(A) Cost Optimization, (B) Fault Tolerance, (C) Performance

Q3) Which element of AWS global infrastructure enables caching of assets in a global content delivery network to reduce latency for end users?
(C) Edge location

Q4) You have multiple web servers deployed across multiple availability zones. What service would provide the ability to route incoming requests between these servers?
(D) Elastic Load Balancing

Q5) After research, you determine that for a specific cloud workload it is cheaper to run it on AWS than in your own data centre when factoring in hardware costs, setup labour, and maintenance. What cloud computing benefit is best illustrated with this use case?
(A) Benefit from massive economies of scale

Q6) Which element of the AWS Global Infrastructure includes multiple availability zones, enabling you to create fault-tolerant applications in the cloud that span multiple availability zones?
(B) Region

Q7) What is the minimum AWS support plan that provides a dedicated Technical Account Manager (TAM)?
(A) Enterprise

Q8) Your company wants to ensure that they have phone, chat, and email access to AWS support with a response within 1 hour when a production system is down. What is the minimum level of support that would meet these criteria?
(B) Enterprise

Q9) When your team moved an application into the cloud, they implemented elasticity so that the application could scale to meet the current user demand. Which cloud computing benefit is best illustrated with this scenario?
(D) Stop guessing capacity

Q10) Your organization is looking to leverage both their traditional data centre and the public cloud. What type of cloud architecture does this describe?
(D) Hybrid cloud

Docker Questions

Q11) <…> is technology that lets you create useful IT services using resources that are traditionally bound to hardware.
(A) Virtualization

Q12) A <…> is computer software, firmware, or hardware that creates and runs virtual machines.
(A) hypervisor

Q13) Which one of the following is not an important advanced Docker command?
(D) None of the above

Q14) In order to start working with Docker SWARM, what is the very first command that you need to run?
(B) docker swarm init

Q15) What are the steps for the Docker container life cycle?
(D) All of the above

Q16) What is the docker command to list down only running containers?
(C) docker ps

Q17) What are the three components of Docker Architecture?
(D) All of the above

Q18) What are the states of Docker containers?
(E) All of the above

Q19) Choose three unique features of Docker over other containerization technologies.
(A) We can run our Docker container either on our PC or on our enterprise IT system, (B) Along with the Docker Hub, which is a repository of all containers, we can deploy and download all our applications from a central location, (C) We can share our applications with the containers that we create

Q20) What are the advantages of using Docker containers?
(E) All of the above

Kubernetes Questions

Q21) What are the main components of Kubernetes?
(D) All of the Above

Q22) Which of the following commands gives you detailed info on a Pod?
(A) kubectl describe pod

Q23) <…> are the Kubernetes controllers.
(C) Both Deployment & Replicaset

Q24) Google Kubernetes comes under which of the following services?
(C) LoadBalancer

Q25) The different services within Kubernetes are <…>
(D) All of the above

Q26) <…> are the types of Kubernetes pods.
(D) Both A and C

Q27) A <…> set is used to keep replica pods stable.
(C) Both A and B

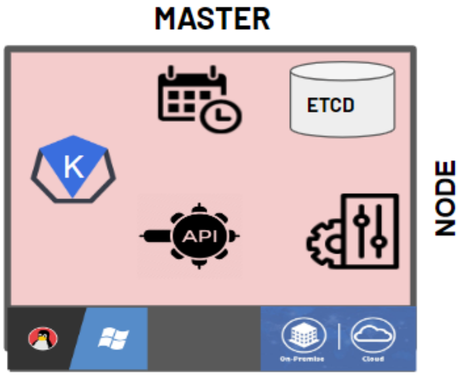
Q28) <…> is the platform using which you can pass commands to the cluster.
(B) Kubectl

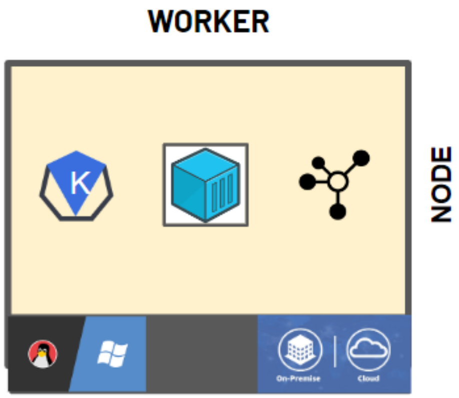
Q29) What is the default range of ports used to expose a Node Port service?
(A) 30000-32767

Q30) You want to deploy two tightly coupled containers that share a volume and some memory. What is the best option?
(C) Deploy them in a single Pod

SECTION B: Diagram Description [20 MARKS]

Given the diagrams below, please describe these two nodes in detail and explain their responsibilities in a Kubernetes cluster. You will be awarded 10 points for correct node descriptions and further 10 points for explaining how these nodes work.





Master Node and Worker Node in a Kubernetes Cluster

A **Kubernetes cluster** consists of two main types of nodes: the **Master Node** and the **Worker Node**. Each has specific tasks and responsibilities that help the cluster function properly. Here's a detailed explanation of these nodes and how they work together.

Master Node (Control Plane)

The **Master Node** is the "brain" of the Kubernetes cluster. It controls and manages everything in the cluster and ensures the desired state of the system is maintained. The Master Node includes several important components:

1. **Scheduler**:
   - The Scheduler decides which worker node will run a new pod (a set of containers) based on the available resources (like CPU and memory).
   - It ensures that the workload is evenly distributed across the nodes.
2. **ETCD**:
   - ETCD is a key-value database that stores all the configuration data about the cluster, including what applications are running, where they are located, and what resources they need.
   - This is a critical part of the system, as it holds the current state and desired state of the cluster. It helps the Master Node keep track of what should be running.
3. **API Server**:
   - The API Server is the central communication point for the entire Kubernetes system. It handles all the commands and requests that come from users, administrators, or other components of the cluster.
   - Whenever you use commands like `kubectl`, you are interacting with the API server.
4. **Controller Manager**:

- This component oversees the various controllers that keep the system running correctly. For example, if a pod dies or fails, the controller will create a new one to maintain the desired number of running pods.
- It also handles other tasks like scaling, ensuring high availability, and maintaining proper replication of services.

The **Master Node** is crucial because it makes all the important decisions regarding the cluster. It can run on different platforms, including **Linux or Windows** operating systems, and can be hosted in **on-premise data centers or in the cloud** (as the icons in the diagram suggest).

Worker Node (Where Applications Run)

The **Worker Node** is where the actual work happens in the Kubernetes cluster. It runs the containerized applications and services. It includes the following key components:

1. **Kubelet**:
    - The Kubelet is an agent that runs on each worker node. It is responsible for making sure that containers (small, packaged versions of applications) are running as they should.
    - It communicates with the Master Node, listens to instructions (like "run this pod here"), and ensures the containers are healthy.
2. **Container Runtime**:
    - This is the software that actually runs the containers. One common container runtime is **Docker**, but there are others like **containerd**.
    - The container runtime creates and manages the lifecycle of containers, ensuring that the applications inside them are running properly.
3. **Kube-Proxy**:
    - Kube-Proxy manages network traffic between different containers, both within the same node and between different nodes.
    - It makes sure that requests to different services are routed correctly and load balances the traffic between containers if necessary.

The **Worker Node** is like the "hands" of the Kubernetes cluster. While the Master Node makes decisions, the Worker Node executes those decisions by running the actual applications. Worker Nodes can also run on **Linux or Windows** and can be either **on-premise or in the cloud**, as shown in the diagram.

How the Master and Worker Nodes Work Together

In a Kubernetes cluster, the **Master Node** and the **Worker Nodes** work in coordination to keep the system running smoothly:

- The **Master Node** takes on the role of **control**. It decides what needs to be done, like which applications to run, where to run them, and how many replicas should be created. It manages the overall health and status of the cluster.
- The **Worker Nodes** are responsible for **executing the tasks**. They run the containers that make up the applications. Each worker node listens to instructions from the Master Node via the Kubelet, pulls the necessary container images, and runs the applications.

If something goes wrong, like a pod (group of containers) failing, the Master Node notices this through ETCD and the API Server. It then uses the **Controller Manager** to fix the issue, maybe by restarting a pod or moving the workload to a healthier Worker Node.

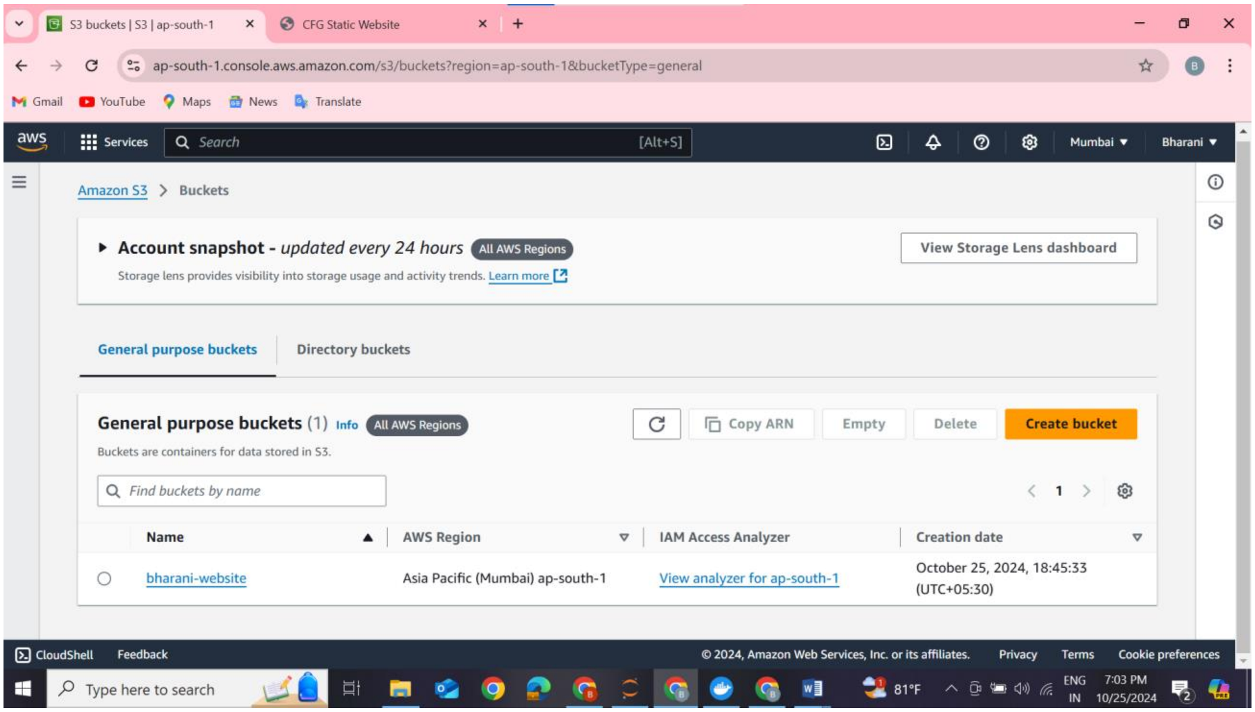
Summary

- **Master Node**: The control center of the Kubernetes cluster. It schedules pods, stores cluster data (ETCD), handles communication (API Server), and manages the state of the system (Controller Manager).
- **Worker Node**: The node that actually runs the containers. It uses Kubelet to manage the containers, Kube-Proxy for network traffic, and the container runtime (like Docker) to execute the containers.

Together, the **Master Node** manages the cluster's desired state, while the **Worker Nodes** handle the actual work of running the applications. This relationship ensures that the Kubernetes cluster is highly available, scalable, and self-healing.

SECTION C: Hosting a Static Website with S3 [20 MARKS]

Please use the contents of the folder called “static-webpage” that have been shared with you at the beginning of this assessment. Open index.html in any browser to see an example of a simple static webpage. The task is to register and host this webpage (and all required contents like images and css) on cloud with the help of AWS S3 service.



Amazon S3 > Buckets

▸ **Account snapshot -** *updated every 24 hours* All AWS Regions

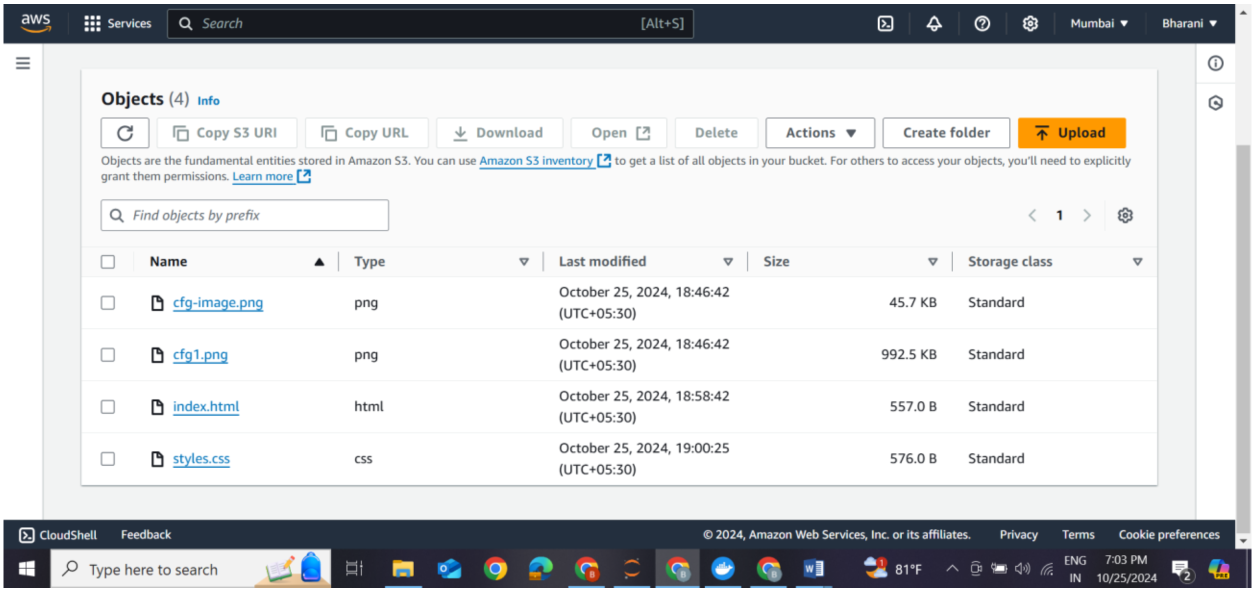
Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets (1) Info All AWS Regions

Buckets are containers for data stored in S3.

	Name ▲	AWS Region	IAM Access Analyzer	Creation date
○	bharani-website	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	October 25, 2024, 18:45:33 (UTC+05:30)

© 2024, Amazon Web Services, Inc. or its affiliates.

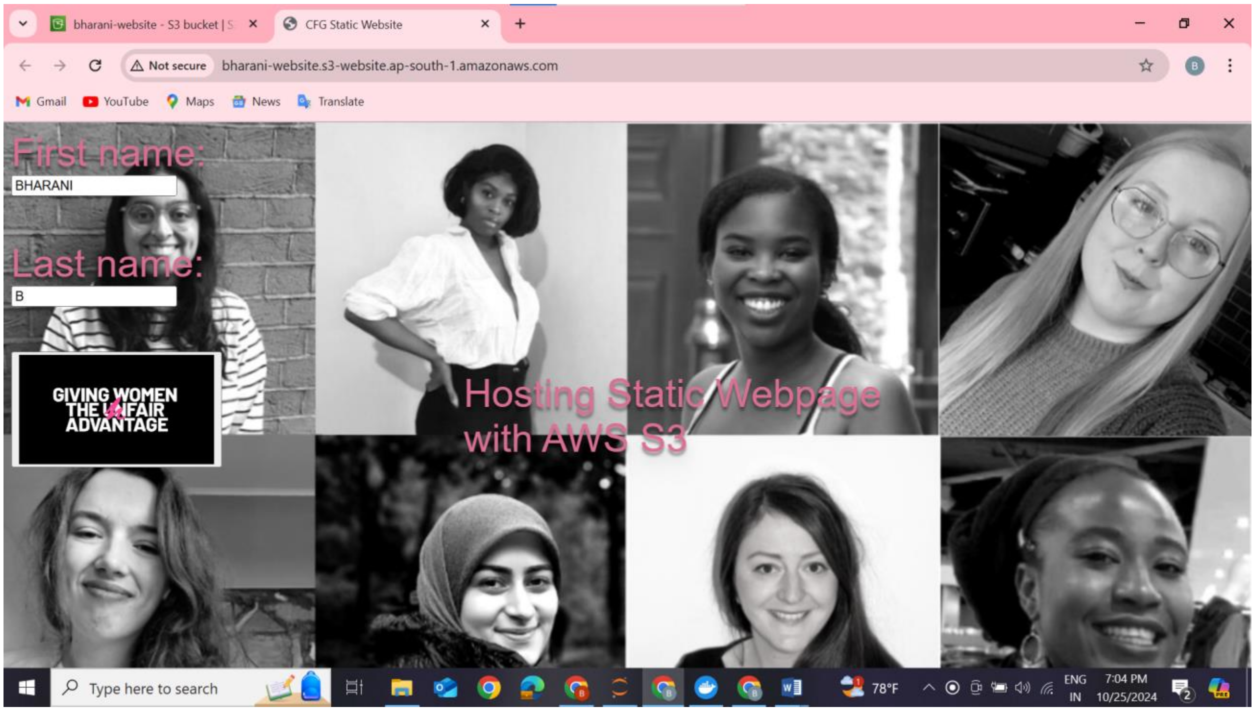


Objects (4) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name ▲	Type	Last modified	Size	Storage class
□	cfg-image.png	png	October 25, 2024, 18:46:42 (UTC+05:30)	45.7 KB	Standard
□	cfg1.png	png	October 25, 2024, 18:46:42 (UTC+05:30)	992.5 KB	Standard
□	index.html	html	October 25, 2024, 18:58:42 (UTC+05:30)	557.0 B	Standard
□	styles.css	css	October 25, 2024, 19:00:25 (UTC+05:30)	576.0 B	Standard

© 2024, Amazon Web Services, Inc. or its affiliates.


bharani-website - S3 bucket | S
CFG Static Website
Not secure
bharani-website.s3-website.ap-south-1.amazonaws.com
Gmail
YouTube
Maps
News
Translate
GIVING WOMEN THE UNFAIR ADVANTAGE
Type here to search
78°F
ENG
IN
7:04 PM
10/25/2024

First name: BHARANI

Last name: B

Hosting Static Webpage with AWS S3

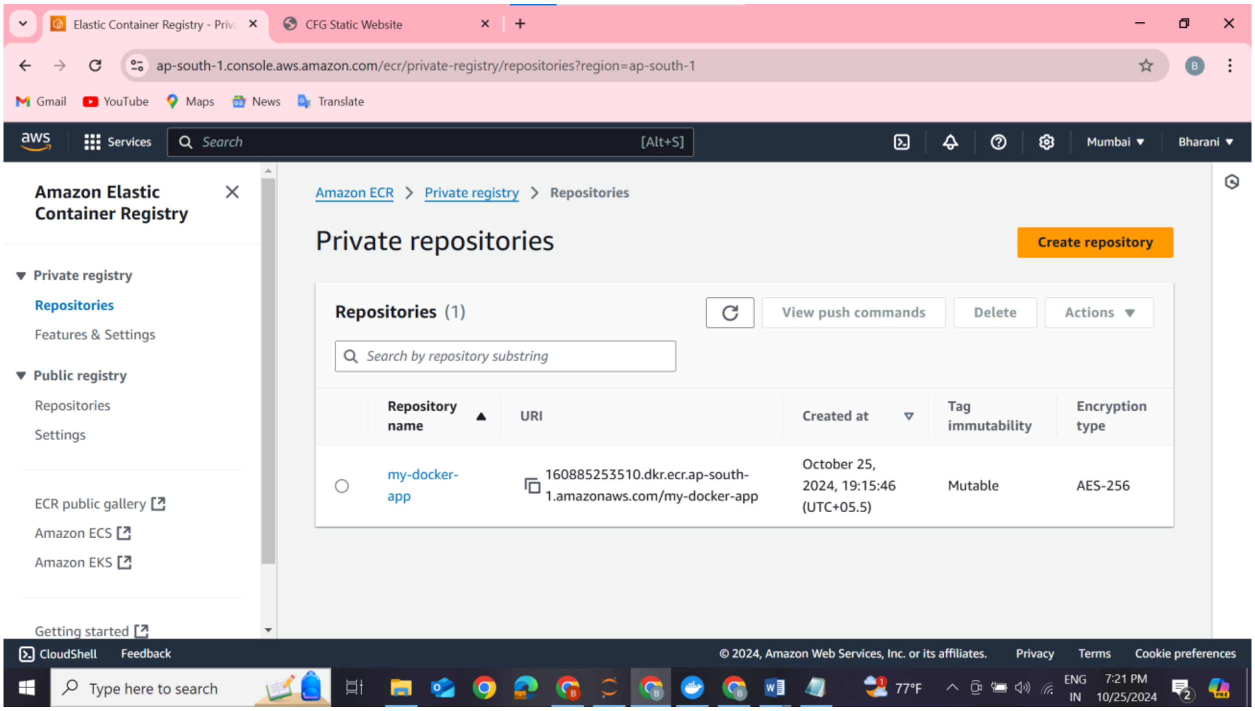
http://bharani-website.s3-website.ap-south-1.amazonaws.com/

SECTION D: Hosting Docker App with AWS [30 MARKS]

Please use the contents of the folder called “docker-app” that have been shared with you at the beginning of this assessment. Open index.html in any browser to see an example of a simple static webpage. You DO NOT NEED the index.html script. It is only an example of what to expect

The task is:

1. Use app.py and Dockerfile to create a new Docker Image
2. Add this new Docker App to AWS ECR
3. Then host the Docker App in the cloud with the help of AWS ECS
4. Your containerised App should be up and running on AWS and accessible by anyone who knows the URL.



Elastic Container Registry - Priv
CFG Static Website
ap-south-1.console.aws.amazon.com/ecr/private-registry/repositories?region=ap-south-1
Gmail
YouTube
Maps
News
Translate
aws
Services
Search
[Alt+S]
Mumbai ▼
Bharani ▼

Amazon Elastic Container Registry

▼ Private registry
- Repositories
- Features & Settings

▼ Public registry
- Repositories
- Settings

ECR public gallery
Amazon ECS
Amazon EKS

Getting started

Amazon ECR > Private registry > Repositories

Private repositories

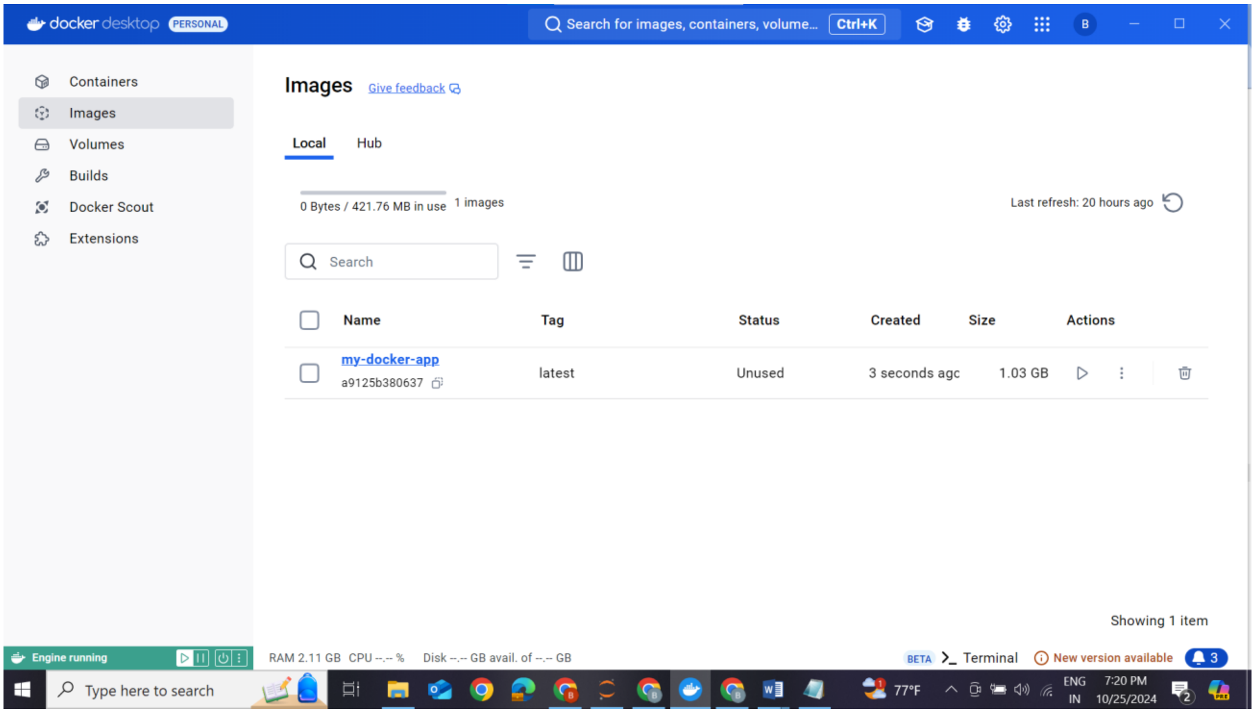
Create repository

Repositories (1)

View push commands | Delete | Actions ▼

Search by repository substring

	Repository name ▲	URI	Created at ▽	Tag immutability	Encryption type
○	my-docker-app	160885253510.dkr.ecr.ap-south-1.amazonaws.com/my-docker-app	October 25, 2024, 19:15:46 (UTC+05.5)	Mutable	AES-256



- Containers
- Images
- Volumes
- Builds
- Docker Scout
- Extensions

Images Give feedback

Local | Hub

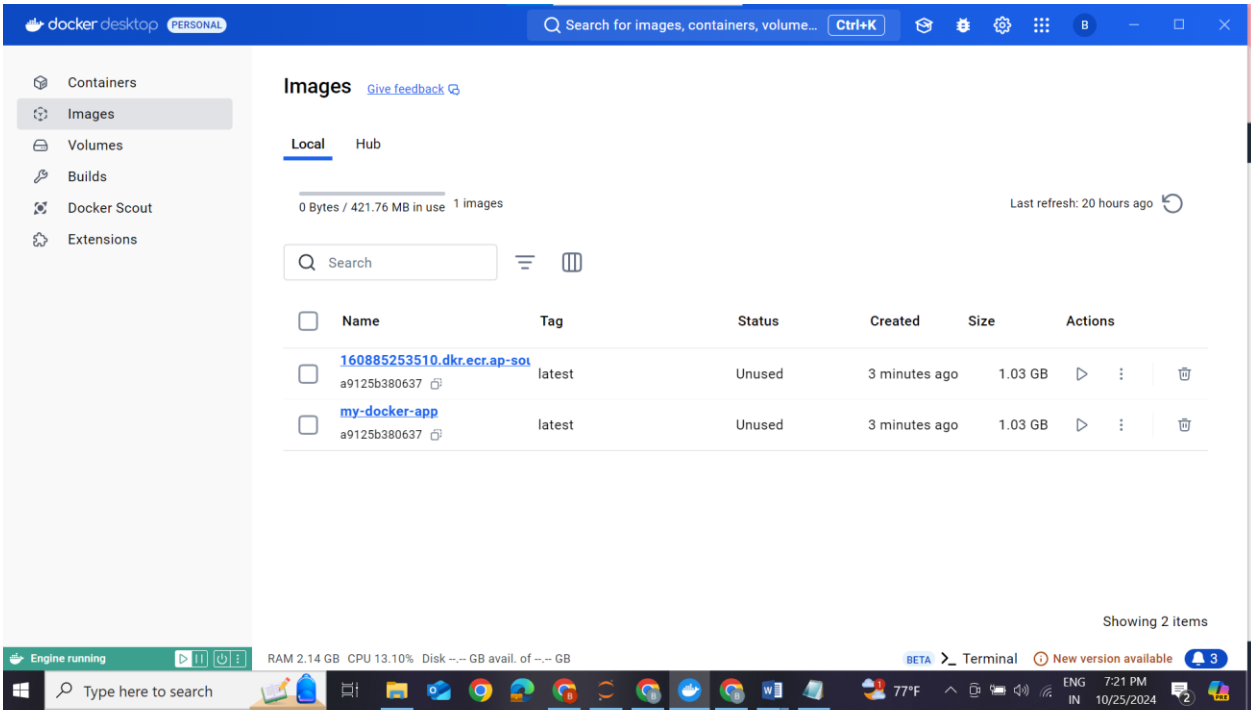
0 Bytes / 421.76 MB in use 1 images

Last refresh: 20 hours ago

Search

☐	Name	Tag	Status	Created	Size	Actions
☐	my-docker-app a9125b380637	latest	Unused	3 seconds agc	1.03 GB	▷ ⋮ 🗑

Showing 1 item



Images

0 Bytes / 421.76 MB in use 1 images

Last refresh: 20 hours ago

Name	Tag	Status	Created	Size	Actions
160885253510.dkr.ecr.ap-sou a9125b380637	latest	Unused	3 minutes ago	1.03 GB	
my-docker-app a9125b380637	latest	Unused	3 minutes ago	1.03 GB	

Showing 2 items

RAM 2.14 GB CPU 13.10% Disk --.-- GB avail. of --.-- GB

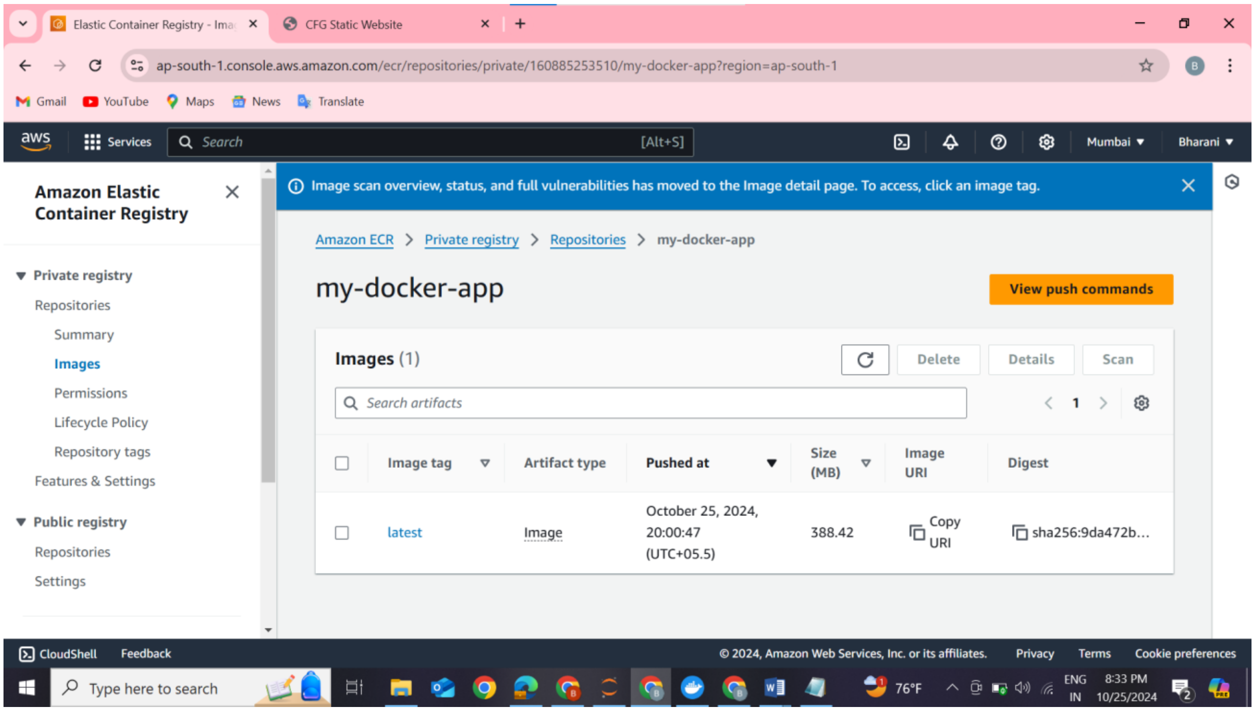


Image scan overview, status, and full vulnerabilities has moved to the Image detail page. To access, click an image tag.

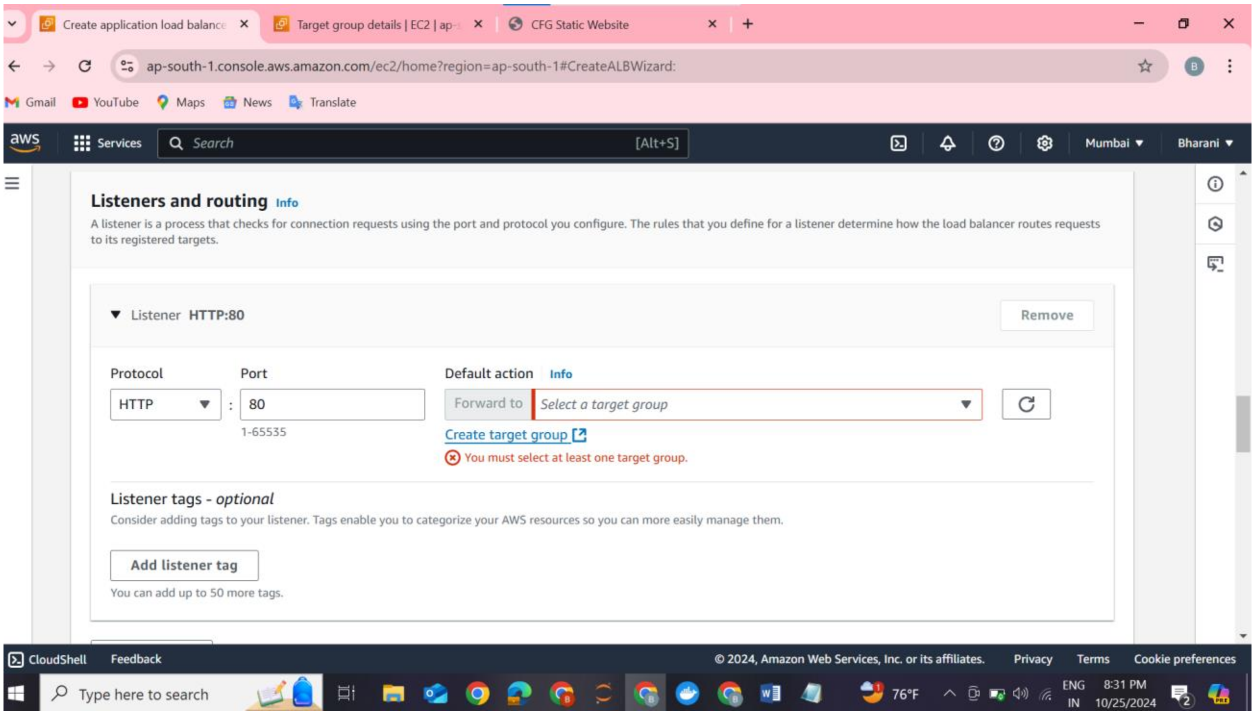
Amazon ECR > Private registry > Repositories > my-docker-app

my-docker-app

Images (1)

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
latest	Image	October 25, 2024, 20:00:47 (UTC+05.5)	388.42	Copy URI	sha256:9da472b...

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#CreateALBWizard:

Listeners and routing Info

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener HTTP:80 — Remove

Protocol: HTTP ▼

Port: 80 (1-65535)

Default action Info: Forward to — *Select a target group* ▼

Create target group

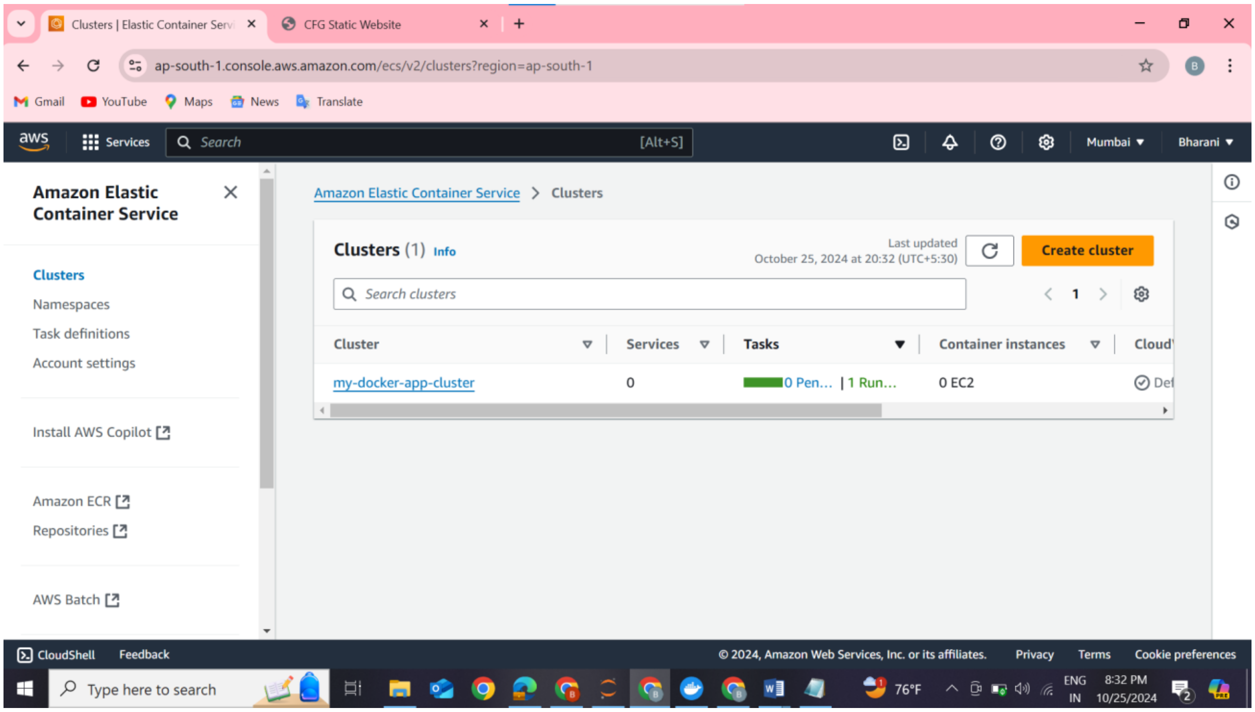
ⓧ You must select at least one target group.

Listener tags - *optional*

Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

Add listener tag

You can add up to 50 more tags.



ap-south-1.console.aws.amazon.com/ecs/v2/clusters?region=ap-south-1

Amazon Elastic Container Service > Clusters

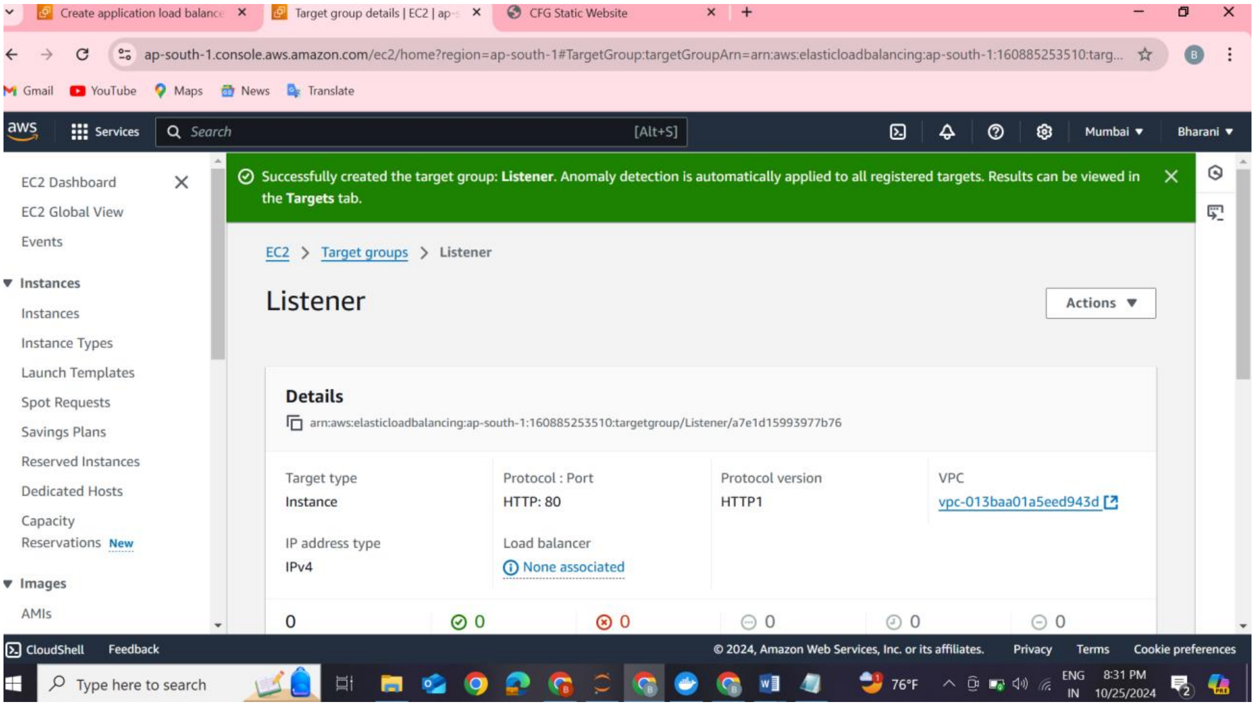
Clusters (1) Info

Last updated
October 25, 2024 at 20:32 (UTC+5:30)

Create cluster

Search clusters

Cluster	Services	Tasks	Container instances	Cloud
my-docker-app-cluster	0	0 Pen... | 1 Run...	0 EC2	Def



Successfully created the target group: **Listener**. Anomaly detection is automatically applied to all registered targets. Results can be viewed in the **Targets** tab.

EC2 > Target groups > Listener

Listener

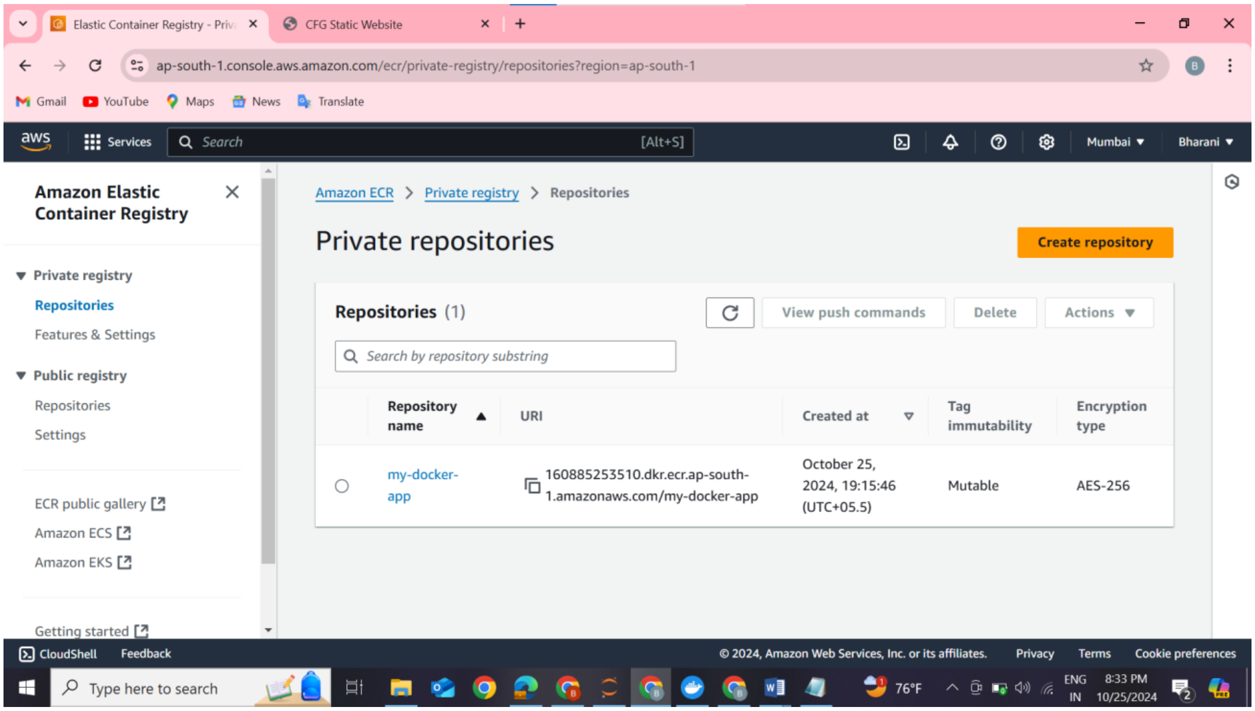
Actions ▼

Details

arn:aws:elasticloadbalancing:ap-south-1:160885253510:targetgroup/Listener/a7e1d15993977b76

Target type	Protocol : Port	Protocol version	VPC
Instance	HTTP: 80	HTTP1	vpc-013baa01a5eed943d
IP address type	Load balancer		
IPv4	None associated		

0 ✓ 0 ⊗ 0 ⊖ 0 ◷ 0 ⊖ 0



Amazon ECR > Private registry > Repositories

Private repositories

Create repository

Repositories (1)

View push commands | Delete | Actions ▼

Search by repository substring

	Repository name ▲	URI	Created at ▽	Tag immutability	Encryption type
○	my-docker-app	160885253510.dkr.ecr.ap-south-1.amazonaws.com/my-docker-app	October 25, 2024, 19:15:46 (UTC+05.5)	Mutable	AES-256