

School: CLOUDBOOSTA	Course: DevOps
Name: AKOSA SAMUEL ONYEJEKWE	Lab-1_ Docker-1
	Week: 4 of 8
Student No: CBALC0168	Issue Date: May 23, 2026

Tasks:

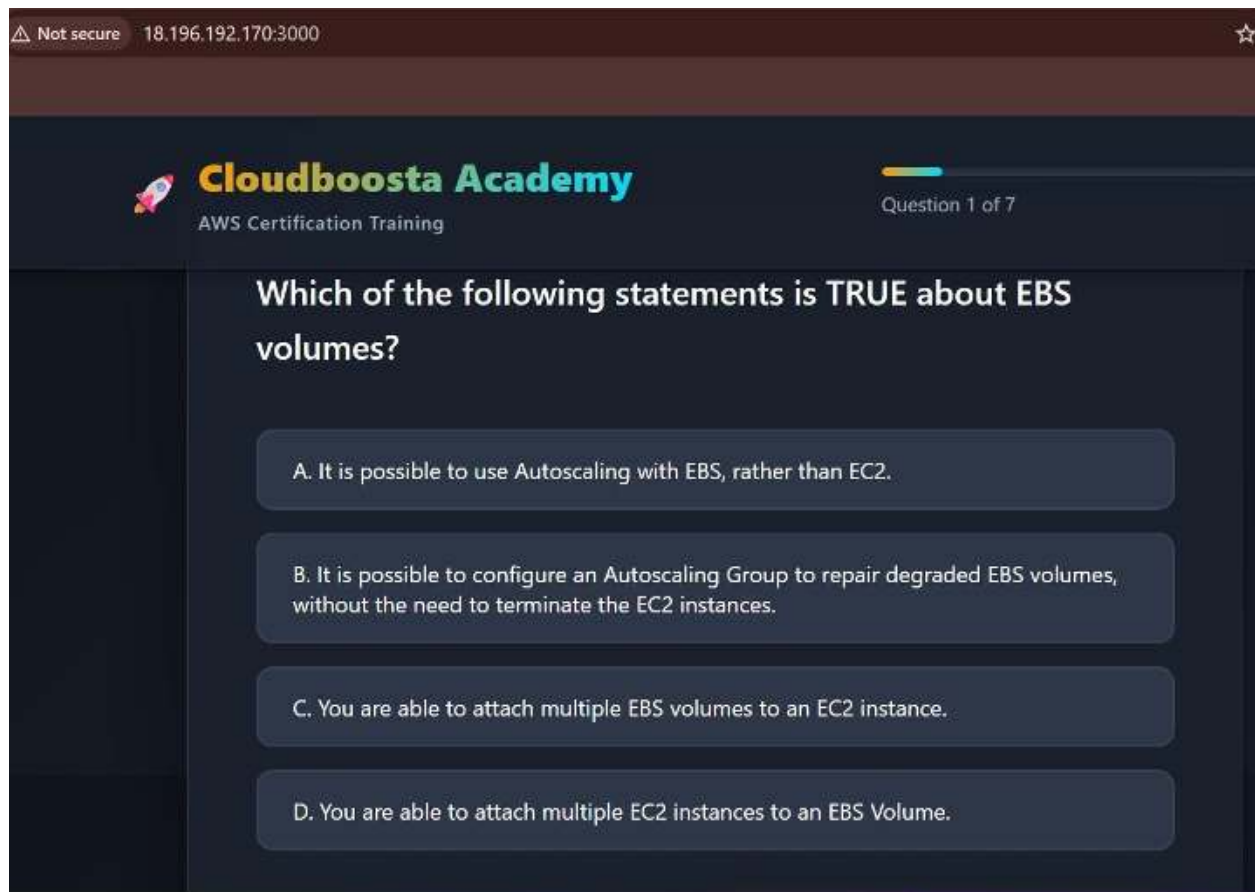
1. Launch Docker EC2 server instance
2. Configure Ubuntu Server 24.04 EC2 instance
3. Configure Docker EC2 security groups
4. Configure inbound security group rules
5. Configure EC2 advanced instance settings
6. Configure EC2 user data bootstrap automation
7. Install Docker Engine on Ubuntu server
8. Install Docker Compose plugin
9. Configure Docker user permissions
10. Start Docker daemon service
11. Verify Docker installation status
12. Connect to Docker EC2 server instance
13. Verify EC2 public IPv4 accessibility
14. Configure Visual Studio Code remote SSH connection
15. Clone WebApp-Quiz-B GitHub repository
16. Verify cloned project source files
17. Configure Dockerfile container image definition
18. Configure Node.js application containerization
19. Build Docker application image
20. Verify Docker image creation
21. Run Docker container instance
22. Configure Docker port mapping
23. Verify running Docker containers
24. Verify Docker container process execution
25. Access deployed web application using browser
26. Verify Cloudboosta Academy application interface
27. Verify application frontend rendering
28. Validate Dockerized application deployment workflow

Goal:

To configure a cloud-native Docker deployment environment using Amazon EC2, Ubuntu Server, Docker Engine, Docker containerization, Visual Studio Code remote development, Node.js web application deployment, Docker image management, container orchestration, and frontend application hosting within AWS cloud infrastructure.

Result:

- Docker EC2 server instance configured successfully
- Ubuntu Server 24.04 instance configured successfully
- Docker security groups configured successfully
- Docker Engine installed successfully
- Docker Compose plugin installed successfully
- Docker daemon configured successfully
- EC2 SSH remote access configured successfully
- Visual Studio Code remote SSH connection configured successfully
- GitHub repository cloned successfully
- Dockerfile configured successfully
- Node.js application containerized successfully
- Docker image built successfully
- Docker container deployed successfully
- Docker port mapping configured successfully
- Cloudboosta Academy web application deployed successfully
- Docker container processes validated successfully
- Frontend application rendering validated successfully
- Dockerized cloud-native deployment workflow validated successfully



Action:

Step 1: Launched Docker EC2 Server Instance

- Opened: EC2 → Launch Instance
- Configured:
 - Instance name: docker_server
 - Ubuntu Server 24.04 LTS AMI
 - Instance type: t3.medium
- Verified Docker EC2 instance launched successfully

Step 2: Configured EC2 Key Pair Authentication

- Configured EC2 key pair successfully
- Verified secure SSH authentication configured successfully

Step 3: Configured Docker EC2 Networking

- Configured default VPC successfully
- Configured subnet successfully
- Enabled auto-assigned public IP successfully
- Confirmed EC2 networking configured successfully

Step 4: Configured Docker Security Group

- Configured security group name: docker_SG
- Configured security group description successfully
- Verified Docker security group created successfully

Step 5: Configured Inbound Security Rules

- Added SSH rule:
 - Port: 22
 - Source: Anywhere
- Added HTTP application rule:
 - Port: 8080
 - Source: Anywhere
- Added frontend application rule:
 - Port: 3000
 - Source: Anywhere
- Confirmed inbound security group rules configured successfully

Step 6: Configured EC2 Storage

- Configured root volume successfully
- Verified EC2 storage configuration completed successfully

Step 7: Configured EC2 Advanced Settings

- Configured detailed monitoring successfully
- Configured termination protection successfully
- Verified EC2 advanced settings configured successfully

Step 8: Configured EC2 User Data Bootstrap Automation

- Configured Docker Engine installation commands successfully
- Configured Docker Compose plugin installation successfully
- Configured Docker daemon startup commands successfully
- Configured Docker user group permissions successfully
- Verified EC2 bootstrap automation configured successfully

Step 9: Verified Docker EC2 Instance Status

- Opened: EC2 → Instances
- Verified docker_server instance state: Running
- Confirmed EC2 health checks passed successfully

Step 10: Verified EC2 Public IPv4 Address

- Verified public IPv4 address generated successfully
- Confirmed Docker EC2 server accessible successfully

Step 11: Connected to Docker EC2 Server

- Opened: EC2 → Connect → SSH Client
- Verified:
 - SSH connection command
 - Public DNS name
 - SSH key authentication
- Confirmed SSH connectivity configured successfully

Step 12: Configured Visual Studio Code Remote SSH Connection

- Opened Visual Studio Code successfully
- Connected to Ubuntu EC2 server using Remote SSH successfully
- Verified remote development environment configured successfully

Step 13: Verified Docker Installation

- Executed:

- `docker --version`
- Verified Docker Engine installed successfully
- Confirmed Docker service operational successfully

Step 14: Cloned WebApp-Quiz-B Repository

- Executed Git clone command successfully
- Verified WebApp-Quiz-B repository cloned successfully

Step 15: Verified Cloned Application Files

- Verified project files:
 - `app.js`
 - `server.js`
 - `package.json`
 - `Dockerfile`
 - `assets` directory
 - `dist` directory
- Confirmed application source files available successfully

Step 16: Configured Dockerfile Application Definition

- Opened: `Dockerfile`
- Configured:
 - Node.js base image
 - Working directory
 - Package installation commands
 - Application copy instructions
 - Port exposure configuration
 - Container startup command
- Verified Dockerfile configured successfully

Step 17: Configured Node.js Containerization Workflow

- Configured Node.js application dependencies successfully
- Configured application runtime environment successfully

- Verified Node.js application containerization configured successfully

Step 18: Built Docker Application Image

- Executed:
 - `docker build -t myapp .`
- Verified Docker image build completed successfully
- Confirmed Docker image exported successfully

Step 19: Verified Docker Image Creation

- Executed:
 - `docker images`
- Verified Docker image:
 - `myapp:latest`
- Confirmed Docker image repository created successfully

Step 20: Executed Docker Container Deployment

- Executed Docker run command successfully
- Configured:
 - Port mapping: 3000:3000
 - Container name: myapp-container
- Verified Docker container deployed successfully

Step 21: Verified Running Docker Containers

- Executed:
 - `docker ps`
- Verified running container status successfully
- Confirmed Docker container uptime validated successfully

Step 22: Verified Docker Container Process Execution

- Verified:
 - Container ID
 - Docker image name
 - Port mappings
 - Running container process
- Confirmed Docker process execution functioning successfully

Step 23: Accessed Deployed Web Application

- Opened browser using EC2 public IP and port 3000 successfully
- Verified frontend application accessible successfully

Step 24: Verified Cloudboosta Academy Application Interface

- Verified:
 - Cloudboosta Academy branding
 - AWS Certification Training interface
 - Quiz application interface
 - Question rendering functionality
- Confirmed application frontend interface rendered successfully

Step 25: Verified Frontend Application Rendering

- Verified:
 - Question navigation buttons
 - Quiz answer options
 - Application styling
 - Responsive frontend rendering
- Confirmed frontend application rendering validated successfully

Step 26: Verified Dockerized Application Deployment Workflow

- Verified:
 - Docker image build workflow
 - Docker container deployment workflow

- Node.js application hosting
 - Browser accessibility
- Confirmed Dockerized deployment workflow functioning successfully

Summary

- Docker EC2 cloud infrastructure configured successfully
- Docker Engine and Docker Compose installed successfully
- Visual Studio Code remote SSH environment configured successfully
- GitHub application repository cloned successfully
- Dockerfile containerization workflow configured successfully
- Docker image creation validated successfully
- Docker container deployment validated successfully
- Cloudboosta Academy web application deployed successfully
- Frontend application accessibility validated successfully
- Dockerized cloud-native deployment workflow implemented successfully

Lab Objective Achieved:

Implemented Docker cloud-native deployment infrastructure, Ubuntu EC2 server provisioning, Docker Engine installation, Docker Compose configuration, Node.js application containerization, Visual Studio Code remote development integration, Docker image management, container deployment orchestration, frontend application hosting, and browser-accessible cloud-native application deployment successfully within AWS EC2 cloud infrastructure.

G

Subnet Info

subnet-030b6e11b6dd7cbbb

VPC: vpc-06057fb779c797663 · Owner: 009850210027

Availability Zone: eu-central-1a (euc1-az2) · Zone type: Availability Zone

IP addresses available: 4091 · CIDR: 172.31.16.0/20

Create new subnet

Auto-assign public IP Info

Enable

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

Security group name - required

docker_5G

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and . _ : / 0 # , @ [+ = & ; [] \$ *

Description - required Info

docker security group

Inbound Security Group Rules

Security group rule 1 (TCP, 22, 0.0.0.0/0)

Remove

Type ssh Protocol TCP Port range 22

Source type Anywhere Source 0.0.0.0/0 Description - optional e.g. SSH for admin desktop

Security group rule 2 (TCP, 8080, 0.0.0.0/0)

Remove

Type Custom TCP Protocol TCP Port range 8080

Source type Anywhere Source 0.0.0.0/0 Description - optional e.g. SSH for admin desktop

Security group rule 3 (TCP, 3000, 0.0.0.0/0)

Remove

Type Custom TCP Protocol TCP Port range 3000

Source type Anywhere Source 0.0.0.0/0 Description - optional e.g. SSH for admin desktop

Add security group rule

Advanced network configuration

Configure storage Info Advanced

1x 8 GiB gp3 Root volume, 3000 IOPS, Not encrypted

Add new volume

The selected AMI contains instance store volumes, however the instance does not allow any instance store volumes. None of the instance store volumes from the AMI will be accessible from the instance

Click refresh to view backup information
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

File systems

☐ S3 Files - new

☐ EFS

☐ FSx

☒ None

▼ Advanced details [Info](#)

Domain join directory [Info](#)

Select

[Create new directory](#)

IAM instance profile [Info](#)

Select

[Create new IAM profile](#)

Hostname type [Info](#)

IP name

DNS Hostname [Info](#)

☒ Enable IP name IPv4 (A record) DNS requests

☐ Enable resource-based IPv4 (A record) DNS requests

☐ Enable resource-based IPv6 (AAAA record) DNS requests

Instance auto-recovery [Info](#)

Select

Shutdown behavior [Info](#)

Stop

Stop - Hibernate behavior [Info](#)

Select

Termination protection [Info](#)

Select

Stop protection [Info](#)

Select

Detailed CloudWatch monitoring [Info](#)

Select

Credit specification [Info](#)

Unlimited

Additional charges apply

Placement group [Info](#)

Select

[Create new placement group](#)

EBS-optimized instance [Info](#)

Enable

Instance bandwidth configuration [Info](#)

Select

Purchasing option [Info](#)

☒ None

☐ Interruptible Capacity Reservations

Launch instances for your Interruptible Capacity Reservations

☐ Spot instances

Request Spot Instances at the Spot price, capped at the On-Demand price

Capacity reservation [Info](#)

Select

Tenancy [Info](#)

Select

RAM disk ID [Info](#)

Select

Kernel ID [Info](#)

Select ▼

Nitro Enclave [Info](#)

Select ▼

License configurations [Info](#)

Select ▼

**CPU options - update** [Info](#)

Configure CPUs for your instance to optimize performance and save on licensing costs.

- ☒ Use default CPU options
- ☐ Specify CPU options

Default active vCPUs

2

Total vCPUs

2

Nested virtualization [Info](#)

Select ▼

Nested virtualization is not supported with the selected instance type.

Metadata accessible [Info](#)

Enabled ▼

Metadata IPv6 endpoint [Info](#)

Select ▼

Metadata version [Info](#)

V2 only (token required) ▼



For V2 requests, you must include a session token in all instance metadata requests. Applications or agents that use V1 for instance metadata access will break.

Metadata response hop limit [Info](#)

2

Allow tags in metadata [Info](#)

Select ▼

User data - optional [Info](#)

Upload a file with your user data or enter it in the field.



Choose file

```
# Update package index again after adding the repository
apt-get update

# Install Docker Engine, CLI, Containerd, and the Docker Compose plugin
apt-get install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin

# Add the 'ubuntu' user to the docker group so you can run docker
commands without sudo
usermod -aG docker ubuntu

# Enable the Docker service to start on boot
systemctl enable docker

# Start the Docker service
systemctl start docker
```

☐ User data has already been base64 encoded

EC2 > Instances

EC2

Dashboard

AWS Global View

Events

Instances

Instances (1/1) Info

Last updated less than a minute ago

Connect

Instance state

Actions

Launch instances

Saved filter sets

Choose filter set

Find Instance by attribute or tag (case-sensitive)

< 1 >

☒	Name	Instance ID	Instance state	Instance type	Status check	Availability Zone
☒	docker_server	i-02e1438ae0a39b308	Running	t3.medium	3/3 checks passed	eu-central-1a

i-02e1438ae0a39b308 (docker_server)

Instance ID

 i-02e1438ae0a39b308

IPv6 address

—

Public IPv4 address

 18.196.192.170 | [open address](#) 

Instance state

 Running

aws

Search

[Alt+S]

Europe (Frankfurt)

Samuel A. O. (009850210027)

cba_admin

EC2

Instances

I-02e1438ae0a39b308

Connect to instance

EC2 Instance Connect

SSM Session Manager

SSH client

EC2 serial console

Instance ID

I-02e1438ae0a39b308 (docker_server)

1. Open an SSH client.

2. Locate your private key file. The key used to launch this instance is sparta-key.pem

3. Run this command, if necessary, to ensure your key is not publicly viewable.

chmod 400 "sparta-key.pem"

4. Connect to your instance using its Public DNS:

ec2-18-196-192-170.eu-central-1.compute.amazonaws.com

Example:

ssh -i "sparta-key.pem" ubuntu@ec2-18-196-192-170.eu-central-1.compute.amazonaws.com

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

CloudShell

Feedback

Console Mobile App

© 2026, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

FileEditSelectionViewGoRun

ubuntu [SSH: ec2-18-196-192-170.eu-central-1.compute.amazonaws.com]

08

EXPLORER

UBUNTU [SSH: EC2-18-196-192-170.EU-CENTRAL-1...]

.cache

.copilot

.ssh

.vscode-server

webApp-Quizz-B

dist

assets

css

js

JS app.js

index.html

.gitignore

package.json

pnpm-lock.yaml

server.js

.bash_history

.bash_logout

.bashrc

.profile

.wget-hsts

OUTLINE

TIMELINE

JS app.js

webApp-Quizz-B > dist > assets > js > JS app.js > ...

1

2

3

4

5

6

7

8

9

10

11

// Define your quiz questions and answers

const quizQuestions = [

{

question: "Which of the following statements is TRUE about EBS volumes?",

choices: [

"A. It is possible to use Autoscaling with EBS, rather than EC2.",

"B. It is possible to configure an Autoscaling Group to repair degraded EB

"C. You are able to attach multiple EBS volumes to an EC2 instance.",

"D. You are able to attach multiple EC2 instances to an EBS Volume."

],

correctAnswer: 2

}

]

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

bash - ubuntu

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-21-127:~\$ docker --version
Docker version 29.5.2, build 79eb04c

ubuntu@ip-172-31-21-127:~\$ cd ~

git clone https://github.com/akinwunmi-akinrimisi/webApp-Quizz-B.git
Cloning into 'webApp-Quizz-B'...
remote: Enumerating objects: 37, done.
remote: Counting objects: 100% (37/37), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 37 (delta 8), reused 34 (delta 7), pack-reused 0 (from 0)
Receiving objects: 100% (37/37), 18.29 KiB | 3.66 MiB/s, done.
Resolving deltas: 100% (8/8), done.

ubuntu@ip-172-31-21-127:~\$

SSH: ec2-18-196-192-170.eu-central-1.com...

main

0 0 0

Ln 1, Col 1 Spaces: 2 UTF

File Edit Selection View Go Run ... ubuntu [SSH: ec2-18-196-192-170.eu-central-1.compute.amazonaws.com]

EXPLORER

- UBUNTU [SSH: EC2-18-196-192-170.EU-CENTRAL-1...]
 - .cache
 - .copilot
 - .ssh
 - .vscode-server
 - webApp-Quizz-B
 - dist
 - assets
 - css
 - js
 - app.js
 - index.html
 - .gitignore
 - Dockerfile
 - package.json
 - pnpm-lock.yaml
 - server.js
 - .bash_history
 - .bash_logout
 - .bashrc
 - .profile
 - .wget-hsts
 - OUTLINE
 - TIMELINE

Dockerfile

```
1 FROM node:14
2 # Set the working directory inside the container
3 WORKDIR /usr/src/app
4 # Copy package.json and package-lock.json to the container
5 COPY package.json ./
6 # Install dependencies
7 RUN npm install
8 # Copy the app files to the container
9 COPY dist/ ./dist/
10 COPY server.js ./
11 # Specify the port on which the server is to be run
12 ENV PORT=3000
13 # Expose the port on which the app will run
14 EXPOSE 3000
15 # Set the command to run your app when the container starts
16 CMD [ "node", "server.js" ]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

ubuntu@ip-172-31-21-127:~\$

SSH: ec2-18-196-192-170.eu-central-1.com... main* 0 0 0 Ln 16

File Edit Selection View Go Run ... ubuntu [SSH: ec2-18-196-192-170.eu-central-1.compute.amazonaws.com]

EXPLORER

UBUNTU [SSH: EC2-18-196-192-170.EU-CENTRAL-1...

- .cache
- .copilot
- .docker
- .ssh
- .vscode-server
- webApp-Quizz-B
 - dist
 - assets
 - css
 - js
 - app.js
 - index.html
 - gitignore
 - Dockerfile
 - package.json
 - pnpm-lock.yaml
 - server.js
 - .bash_history
 - .bash_logout
 - .bashrc
 - .profile
 - .wget-hsts
 - OUTLINE
 - TIMELINE

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - webApp-Quizz-B

```
ubuntu@ip-172-31-21-127:~$ cd ~/webApp-Quizz-B

docker build -t myapp .
=> => sha256:2ff1d7c41c74a25258bfa6f0b8adb0a727f84518f55f65ca845ebc747976 50.45MB / 50.45MB 0.9s
=> => extracting sha256:2ff1d7c41c74a25258bfa6f0b8adb0a727f84518f55f65ca845ebc747976c408 2.4s
=> => extracting sha256:b253aeafeaa7e0671bb60008df01de101a38a045ff7bc656e3b0fbfc7c05cca5 0.3s
=> => extracting sha256:3d2201bd995ccc12851a50820de03d34a17011dcb9ac9fd3a50c952cbb131 0.2s
=> => extracting sha256:1de76e268b103d05fa8960e0f77951ff54b912b63429c34f5d6adfd09f5f9ee2 2.1s
=> => extracting sha256:d9a8df5894511ce28a05e2925a75e8a4acbd0634c39ad734fd0fba8e23d1b1569 6.3s
=> => extracting sha256:6f51ee005deac0d99898e41b8ce60ebf250ebe1a31a0b03f613aec6bbcb9b83d8 0.0s
=> => extracting sha256:5f32ed3c3f278edda4fc571c880b5277355a29ae8f52b52cdf865f058378a590 1.7s
=> => extracting sha256:0c8cc2f24a4dcb64e602e086fc9446b0a541e8acd9ad72d2e90df3ba22f158b3 0.1s
=> => extracting sha256:0d27a8e861329007574c6766fba946d48e20d2c8e964e873de352603f22c4ceb 0.0s
=> [internal] load build context
=> => transferring context: 17.66kB
=> [2/6] WORKDIR /usr/src/app
=> [3/6] COPY package.json ./
=> [4/6] RUN npm install
=> [5/6] COPY dist/ ./dist/
=> [6/6] COPY server.js ./
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:85e6822aac02bb49dfd1e605ccddc109e414b963fc80e68d682ca118a3f 0.0s
=> => exporting config sha256:6c4d1b73d7eccd5349a2a5bee7ed260bc54dc53042c8bb58df26b6aadf424 0.0s
=> => exporting attestation manifest sha256:d46a01143f004747066c26853c33f61b8ec443b02ad074f 0.0s
=> => exporting manifest list sha256:1872e992b2ac6594683d9ba9f5b8791679c3157f1999da8e99c577 0.0s
=> => naming to docker.io/library/myapp:latest
=> => unpacking to docker.io/library/myapp:latest 0.4s

ubuntu@ip-172-31-21-127:~/webApp-Quizz-B$
```

SSH: ec2-18-196-192-170.eu-central-1.com... main*

FileEditSelectionViewGoRun...< -> ubuntu [SSH: ec2-18-196-192-170.eu-central-1.compute.amazonaws.com]

EXPLORER

UBUNTU [SSH: EC2-18-196-192-170.eu-central-1.compute.amazonaws.com]

> .cache

> .copilot

> .docker

> .ssh

> .vscode-server

webApp-Quizz-B

dist

assets

css

js

app.js

index.html

.gitignore

Dockerfile

package.json

pnpm-lock.yaml

server.js

.bash_history

.bash_logout

.bashrc

.profile

.wget-hsts

OUTLINE

TIMELINE

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

bash - webApp-Quizz-B

ubuntu@ip-172-31-21-127:~/webApp-Quizz-B\$ docker images

IMAGE	ID	DISK USAGE	CONTENT SIZE	EXTRA
myapp:latest	1872e992b2ac	1.36GB	352MB	

ubuntu@ip-172-31-21-127:~/webApp-Quizz-B\$ docker run -p 3000:3000 -d myapp

b13b02dc7f94eaf11a808594be5d31a23000e711ad5ce9d24adfc67d13b30065

ubuntu@ip-172-31-21-127:~/webApp-Quizz-B\$ docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
		NAMES			
b13b02dc7f94	myapp	"docker-entrypoint.s..."	38 seconds ago	Up 37 seconds	0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp

ubuntu@ip-172-31-21-127:~/webApp-Quizz-B\$

SSH: ec2-18-196-192-170.eu-central-1.com...main*000



Which of the following statements is TRUE about EBS volumes?

- A. It is possible to use Autoscaling with EBS, rather than EC2.
- B. It is possible to configure an Autoscaling Group to repair degraded EBS volumes, without the need to terminate the EC2 instances.
- C. You are able to attach multiple EBS volumes to an EC2 instance.
- D. You are able to attach multiple EC2 instances to an EBS Volume.

← Back

Next Question →



Which of the following statements is TRUE about EBS volumes?

A. It is possible to use Autoscaling with EBS, rather than EC2.

B. It is possible to configure an Autoscaling Group to repair degraded EBS volumes, without the need to terminate the EC2 instances.

C. You are able to attach multiple EBS volumes to an EC2 instance.

D. You are able to attach multiple EC2 instances to an EBS Volume.

FileEditSelectionViewGoRun

ubuntu [SSH: ec2-18-196-192-170.eu-central-1.compute.amazonaws.com]

EXPLORER

UBUNTU [SSH: EC2-18-196-192-170.EU-CENTRAL-1...]

.aws

.cache

.copilot

.docker

.ssh

.vscode-server

webApp-Quizz-B

dist

assets

css

js

app.js

index.html

.gitignore

Dockerfile

package.json

pnpm-lock.yaml

server.js

.bash_history

.bash_logout

.bashrc

.lessht

.profile

OUTLINE

TIMELINE

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS 1

bash - tmp

ubuntu@ip-172-31-21-127:/tmp\$ docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
b9b72820fd18	myapp	"docker-entrypoint.s..."	26 minutes ago	Up 26 minutes	0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp

myapp-container

ubuntu@ip-172-31-21-127:/tmp\$ docker stop b9b72820fd18

b9b72820fd18

ubuntu@ip-172-31-21-127:/tmp\$ docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

ubuntu@ip-172-31-21-127:/tmp\$

SSH: ec2-18-196-192-170.eu-central-1.com...

main*

0 0 1