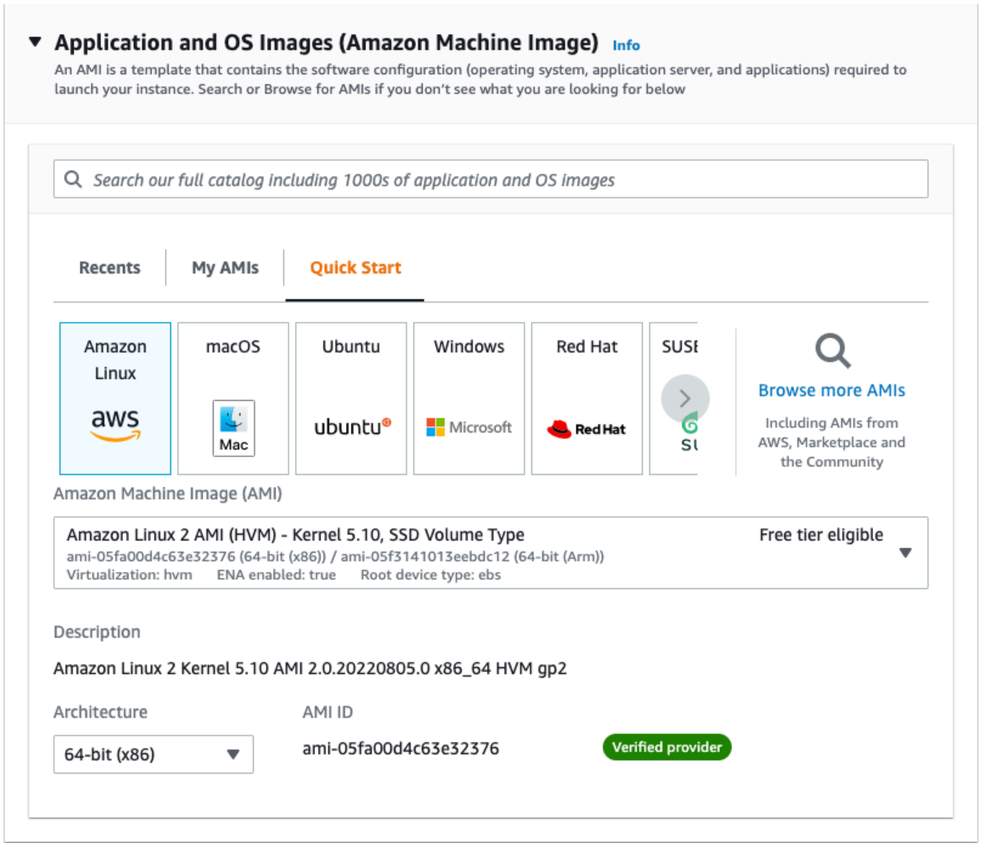
CS 548—Spring 2023
Enterprise Software Architecture and Design
Assignment One—Cloud Computing

In this assignment, you will set up a software platform in the cloud. You will set this up in the Amazon Elastic Compute Cloud (EC2), which provides an infrastructure-as-a-service (IaaS) for configuring a custom software stack. You will use Docker to ensure you will have the same environment for local development and for deployment[1], and to isolate application components that you deploy.

Step 1: Launch an EC2 instance

Launch an EBS-backed EC2 instance. Launch the instance from a bare **Amazon Linux 2** 64-bit AMI, without any additional software installed. Elastic Block Store (EBS) is essentially a virtual disk that Amazon provides for backing storage.


Application and OS Images (Amazon Machine Image)
Info
Search our full catalog including 1000s of application and OS images
Recents
My AMIs
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type — Free tier eligible
ami-05fa00d4c63e32376 (64-bit (x86)) / ami-05f3141013eebdc12 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

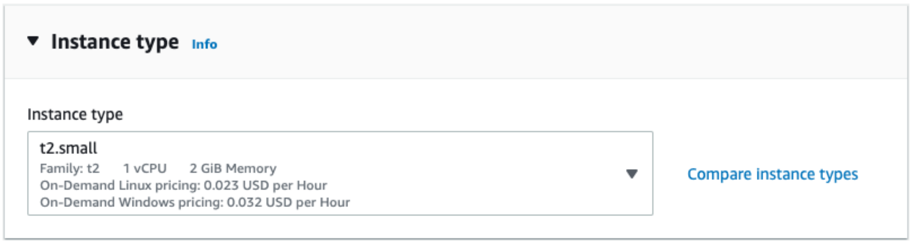
Amazon Linux 2 Kernel 5.10 AMI 2.0.20220805.0 x86_64 HVM gp2

Architecture: 64-bit (x86)

AMI ID: ami-05fa00d4c63e32376 — Verified provider

[1] If your development machine is Windows or Intel Mac, you should install Docker Desktop. This will install a Linux VM, and the docker command line tool will manage containers running on this VM. This will allow you to develop on your local machine, and then deploy your working application on EC2. Unfortunately, the Payara image we will be using is not currently available for Apple ARM, so you will have to develop on EC2. Alternatively, you can for now develop natively on your Mac (installing Payara locally) and demonstrate deployment on EC2.

Since you will be installing both Postgresql and Payara on your instance, a micro-sized instance will not have enough memory. Pick a small-sized instance. Provided you do not leave an instance running, your charges should be minimal:

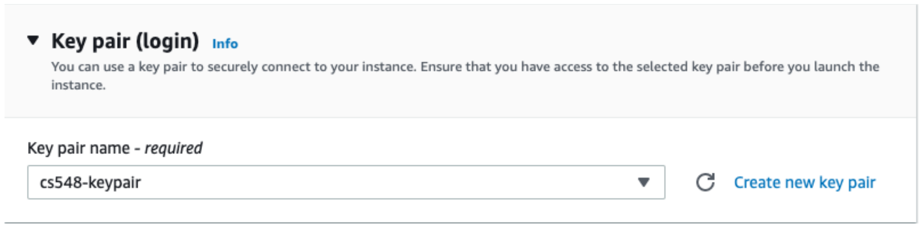


▼ Instance type Info

Instance type

t2.small
Family: t2 1 vCPU 2 GiB Memory
On-Demand Linux pricing: 0.023 USD per Hour
On-Demand Windows pricing: 0.032 USD per Hour

If this is your first time creating an EC2 instance, you will want to create a (RSA) key pair and save this on your laptop. You will need the key pair in order to access the EC2 instance. Be sure to protect this key pair and never share it with anyone.



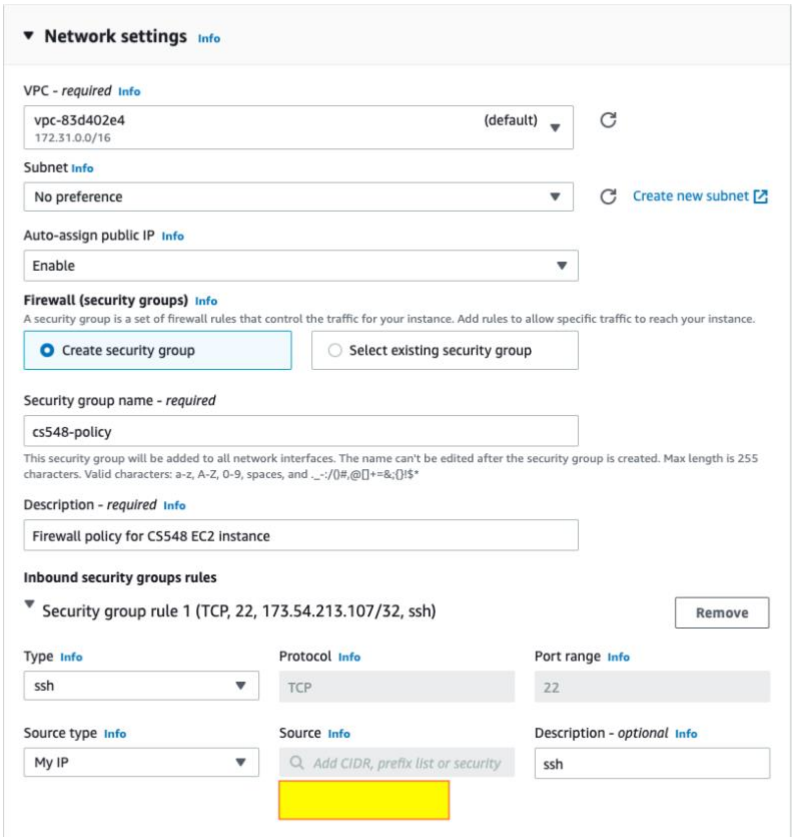
▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

cs548-keypair

For network settings, the EC2 wizard will automatically create a virtual private cloud (VPC) and public subnet with that cloud for your instance. You must set up a firewall policy to protect your instance. **You should only access from your laptop.** By default, the EC2 wizard will create a rule allowing you to ssh to your instance to administer it:



Network settings Info

VPC - *required* Info

vpc-83d402e4 (default)
172.31.0.0/16

Subnet Info

No preference

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*

cs548-policy

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 ._-:/()#,@[]+=&;{}!$*

Description - *required* Info

Firewall policy for CS548 EC2 instance

Inbound security groups rules

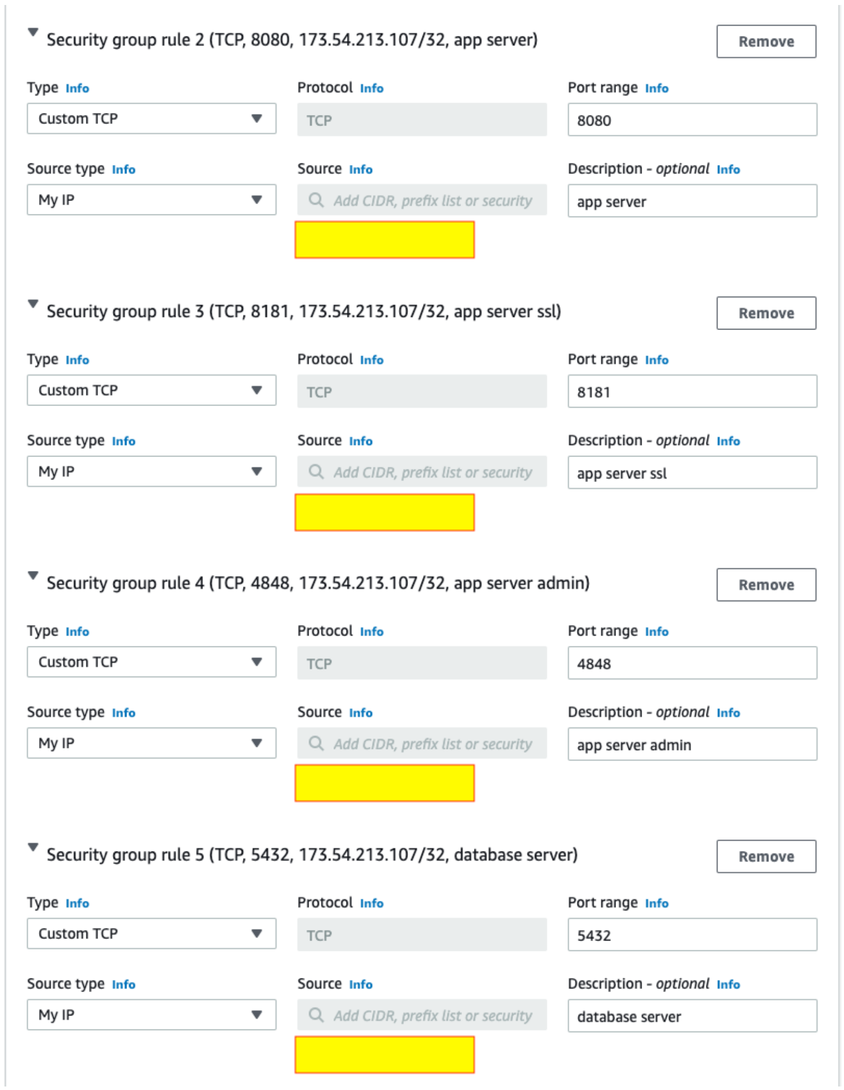
Security group rule 1 (TCP, 22, 173.54.213.107/32, ssh)

Remove

Type	Protocol	Port range
ssh	TCP	22

Source type	Source	Description - *optional*
My IP	Add CIDR, prefix list or security	ssh

You should add other rules to the firewall policy, to allow access to ports 4848 (for administering the application server), 8080 and 8181 (accessing applications on the application server) and 5432 (for accessing the database server). The latter is in case you want to connect to the database using Eclipse, which you would not be doing in a production environment. **Make sure that your instance can only be accessed from your IP address.**



▼ Security group rule 2 (TCP, 8080, 173.54.213.107/32, app server) — Remove

Type Info	Protocol Info	Port range Info
Custom TCP	TCP	8080
Source type Info	Source Info	Description - *optional* Info
My IP	Add CIDR, prefix list or security [redacted]	app server

▼ Security group rule 3 (TCP, 8181, 173.54.213.107/32, app server ssl) — Remove

Type Info	Protocol Info	Port range Info
Custom TCP	TCP	8181
Source type Info	Source Info	Description - *optional* Info
My IP	Add CIDR, prefix list or security [redacted]	app server ssl

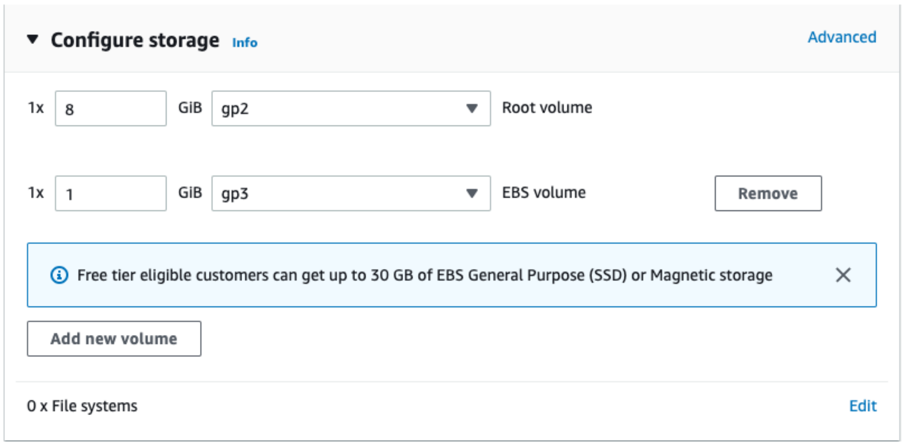
▼ Security group rule 4 (TCP, 4848, 173.54.213.107/32, app server admin) — Remove

Type Info	Protocol Info	Port range Info
Custom TCP	TCP	4848
Source type Info	Source Info	Description - *optional* Info
My IP	Add CIDR, prefix list or security [redacted]	app server admin

▼ Security group rule 5 (TCP, 5432, 173.54.213.107/32, database server) — Remove

Type Info	Protocol Info	Port range Info
Custom TCP	TCP	5432
Source type Info	Source Info	Description - *optional* Info
My IP	Add CIDR, prefix list or security [redacted]	database server

For storage, EC2 will allocate an 8G virtual disk for the instance, to hold the OS and application. However, if you terminate the instance, this disk is deleted. Allocate an additional EBS volume to hold the persistent state of the database:



▼ **Configure storage** Info — Advanced

1x 8 GiB gp2 Root volume

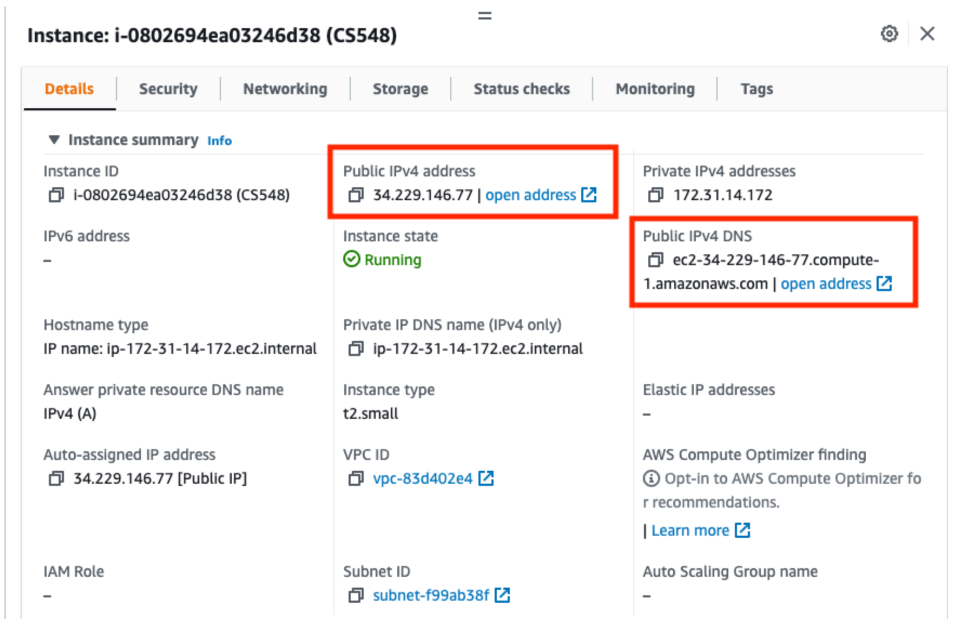
1x 1 GiB gp3 EBS volume — Remove

ⓘ Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Add new volume

0 x File systems — Edit

Once you have launched your instance, you can see it running in your EC2 dashboard. Note the public DNS address and IP address for your running instance, you will use one of these, with the key pair, to ssh to the instance to set it up:



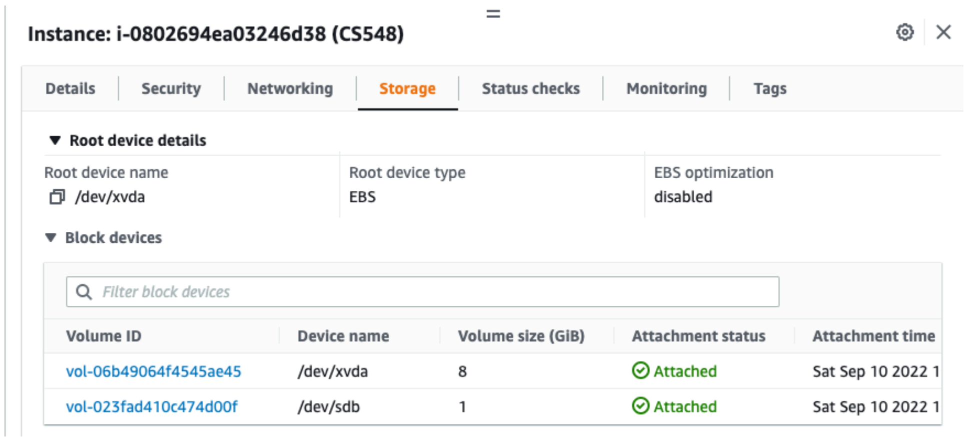
Instance: i-0802694ea03246d38 (CS548)

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

▼ Instance summary Info

Instance ID i-0802694ea03246d38 (CS548)	Public IPv4 address 34.229.146.77 | open address	Private IPv4 addresses 172.31.14.172
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-34-229-146-77.compute-1.amazonaws.com | open address
Hostname type IP name: ip-172-31-14-172.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-14-172.ec2.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.small	Elastic IP addresses –
Auto-assigned IP address 34.229.146.77 [Public IP]	VPC ID vpc-83d402e4	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	Subnet ID subnet-f99ab38f	Auto Scaling Group name –

You can also view the volumes that have been allocated, one for the software you will install, the other for the persistent database content:



Instance: i-0802694ea03246d38 (CS548)

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

▼ Root device details

Root device name	Root device type	EBS optimization
/dev/xvda	EBS	disabled

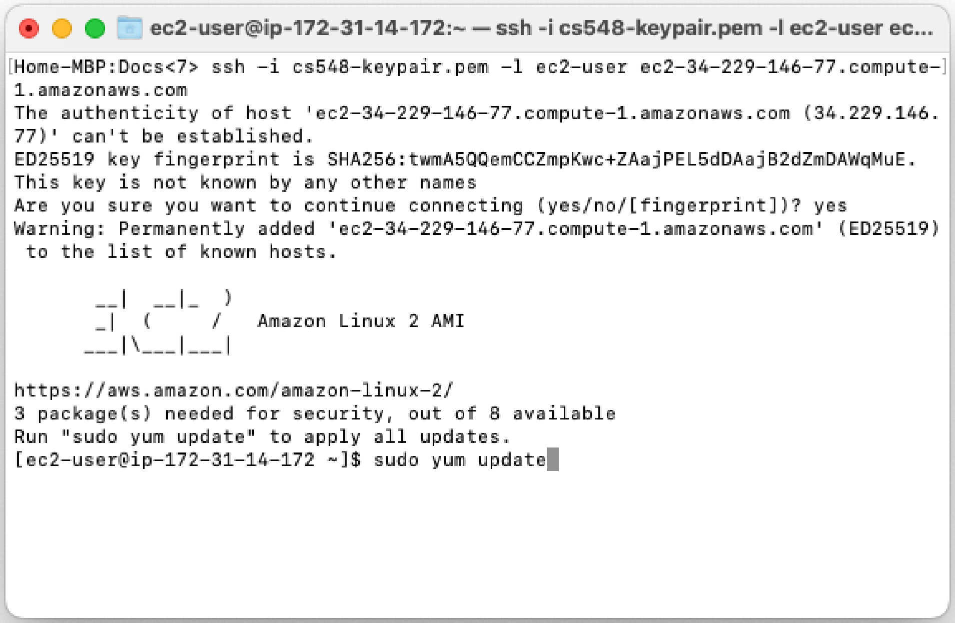
▼ Block devices

Filter block devices

Volume ID	Device name	Volume size (GiB)	Attachment status	Attachment time
vol-06b49064f4545ae45	/dev/xvda	8	Attached	Sat Sep 10 2022 1
vol-023fad410c474d00f	/dev/sdb	1	Attached	Sat Sep 10 2022 1

Step 2: Mount the disk for the database

Use ssh to access your instance:



```
Home-MBP:Docs<7> ssh -i cs548-keypair.pem -l ec2-user ec2-34-229-146-77.compute-
1.amazonaws.com
The authenticity of host 'ec2-34-229-146-77.compute-1.amazonaws.com (34.229.146.
77)' can't be established.
ED25519 key fingerprint is SHA256:twmA5QQemCCZmpKwc+ZAajPEL5dDAajB2dZmDAWqMuE.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-34-229-146-77.compute-1.amazonaws.com' (ED25519)
 to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
3 package(s) needed for security, out of 8 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-14-172 ~]$ sudo yum update
```

Install any software updates. It is very important that you keep the software on your machine up-to-date, particularly with security updates. Then edit the mount table `/etc/fstab` to automatically mount the external volume as a Linux file system whenever the machine boots. In what follows, `[ec2-user]` denotes the default user prompt, while `[root]` denotes the superuser prompt, while ****** denotes the UUID you obtain from `file -s`:

```
[ec2-user] sudo su -
[root] fdisk -l
[root] mkfs -t ext3 /dev/xvdb
[root] file -s /dev/xvdb
[root] echo "UUID=***** /data ext3 noatime 0 0" >> /etc/fstab
[root] mkdir /data
[root] mount /data
[root] fdisk -l
[root] exit
```

Step 3: Install Docker on your instance

Install Docker, which is available as a package, and start the Docker service. You will be using the command-line Docker client to manage Docker containers and images:

```
[ec2-user] sudo yum install -y docker
```

```
[ec2-user] sudo service docker start
```

```
[ec2-user] sudo usermod -a -G docker ec2-user
```

The last command makes it unnecessary to use sudo to execute the Docker client. You will need to log out and log back in again, and you may need to reboot the instance (do not terminate it). If you reboot, you will have to restart the docker service.

Step 4: Create the database container

Rather than install Postgresql natively using yum, you will instead install a docker image that includes a Postgresql installation, and run this as a container. First, create a virtual network and then the container that runs on that network. *For local development only, it will be useful to add "`-p 5432:5432`" as an option when running the database container, to expose the port so Eclipse can connect to the database server:*

```
[ec2-user] docker network create --driver bridge cs548-network
```

```
[ec2-user] docker pull postgres
```

```
[ec2-user] docker run -d --name cs548db --network cs548-network -p
5432:5432 -v /data:/var/lib/postgresql/data -e POSTGRES_PASSWORD=XXXXXX
-e PGDATA=/var/lib/postgresql/data/pgdata postgres
```

```
[ec2-user] docker ps
```

The `docker run` command will create and start the database container in the background (due to the `–d` flag). The container will run in the virtual network `cs548-network`, with virtual host name cs548db. You have mounted the external disk at `/data`, and this is now mounted at `/var/lib/postgresql/data` in the container. The `PGDATA` environment variable sets the container directory where the database will be stored. The `POSTGRES_PASSWORD` environment variable defines the password for the database superuser (default superuser is `postgres`).

Note: If you do not see the running container, you can see all docker containers including those that have stopped, and view the logs for a particular docker container by specifying its container id:

```
docker ps -a
```

```
docker logs <container-id>
```

You can stop a container using `docker stop`, remove a container using `docker rm`, and remove all stopped docker containers using `docker container prune`.

With the database container still running, run a bash shell container from the same image in the same virtual network, and use a Postgresql command line tool in the shell to create a user for the database you will be creating. This command will connect to the running database. In this case, the `-it` flag runs the command interactively, while the `--rm` flag removes the shell container when you are done[2]:

```
[ec2-user] docker run -it --rm --network cs548-network postgres
/bin/bash
# createuser cs548user -P --createdb -h cs548db -U postgres
Enter password for new role: YYYYYY
Enter it again: YYYYYY
Password: XXXXXX (see above)
# exit
```

Note that the `createuser` command connects to the virtual host `cs548db`, as superuser `postgres`. Run a `psql` shell to create the database for this user[3]:

```
[ec2-user] docker run -it --rm --network cs548-network postgres psql -h
cs548db -U postgres
postgres=# create database cs548 with owner cs548user;
postgres=# \q
```

Step 5: Create the server container

We will be using the Payara Server docker image for now[4]. Create a custom app server image that includes the latest JDBC driver. First download the driver (go to `http://jdbc.postgresql.org` to find the link for the latest version):

```
[ec2-user] mkdir cs548-payara
[ec2-user] cd cs548-payara
[ec2-user] wget <link to the driver>
```

Create a file called `Dockerfile` that sets some environment variables and copies the JDBC driver into the server libraries[5]:

```
FROM payara/server-full:6.2023.1-jdk17
COPY <JDBC driver file name>
            ${PAYARA_DIR}/glassfish/domains/${DOMAIN_NAME}/lib/
```

[2] Alternatively, you might use docker exec to attach to a running container and execute a command, such as starting a shell.
[3] If you later find you need to drop the database because you've changed the schema, you can do so (as user postgres) with this psql command: `drop database cs548 with (force);`
[4] https://docs.payara.fish/community/docs/documentation/ecosystem/docker-images/server-image-overview.html.
[5] There are two lines, the second line is broken to fit in this document.

Save this file, and create a custom app server image with the JDBC driver:

```
[ec2-user] docker build -t cs548/server .
[ec2-user] docker images
[ec2-user] cd ..
```

Create and start the server container, on the same virtual network as the database. Note that you specify the same image name as above when you executed `docker build`; this image will have been cached locally. You will need to expose several ports to allow access from your Web browser, through your EC2 firewall:

```
[ec2-user] docker run -d --name payara --network cs548-network -p
4848:4848 -p 8080:8080 -p 8181:8181 cs548/server
```

Use a Web browser to verify that the application server is running by going to the admin console at port 4848. The default user name and password are "`admin`", you should change the password in the admin console.

You need to create a JDBC connection pool so applications deployed in the application server can connect to the database. In the admin console, navigate to `Resources | JDBC | Connection Pools.` Create a connection pool with name `cs548Pool`, with type `javax.sql.ConnectionPoolDataSource`, and with vendor `PostgreSQL`. Set these data source properties[6]:

- `DatabaseName=cs548`
- `Password=`*`YYYYYY`* (see above)
- `ServerName=cs548db`
- `PortNumber=5432`
- `User=cs548user`
- `URL=jdbc:postgresql://cs548db:5432/cs548`

When you have saved these changes, select the new connection pool and click `Ping`. This will make sure that the application server is able to connect to the database server.

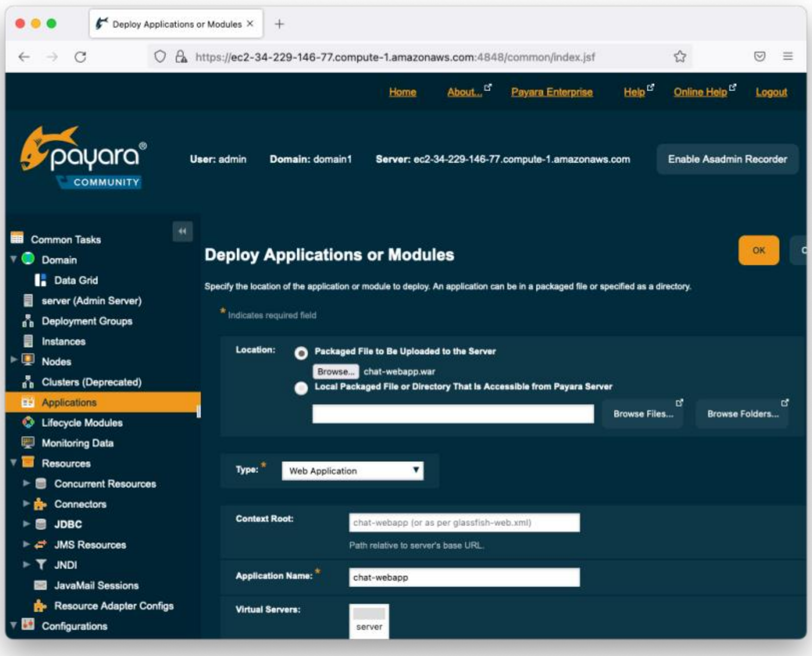
You still need to set up a JDBC resource for the connection pool, which your deployed applications can inject. In the admin console, navigate to `Resources | JDBC | JDBC Resources`. Create a new JDBC resource with these properties:

- JNDI Name: `jdbc/cs548`
- Connection pool: `cs548Pool`

Step 7: Deploy an application

[6] Later we will see how to define in the application how to create this database connection when the application is deployed, and with the database password passed to the application as part of deployment.

In the admin console, select the Applications tab and deploy the WAR file provided for this assignment:



User: admin Domain: domain1 Server: ec2-34-229-146-77.compute-1.amazonaws.com

Deploy Applications or Modules

Specify the location of the application or module to deploy. An application can be in a packaged file or specified as a directory.

* Indicates required field

Location: [● Packaged File to Be Uploaded to the Server] chat-webapp.war
○ Local Packaged File or Directory That Is Accessible from Payara Server

Type: * Web Application

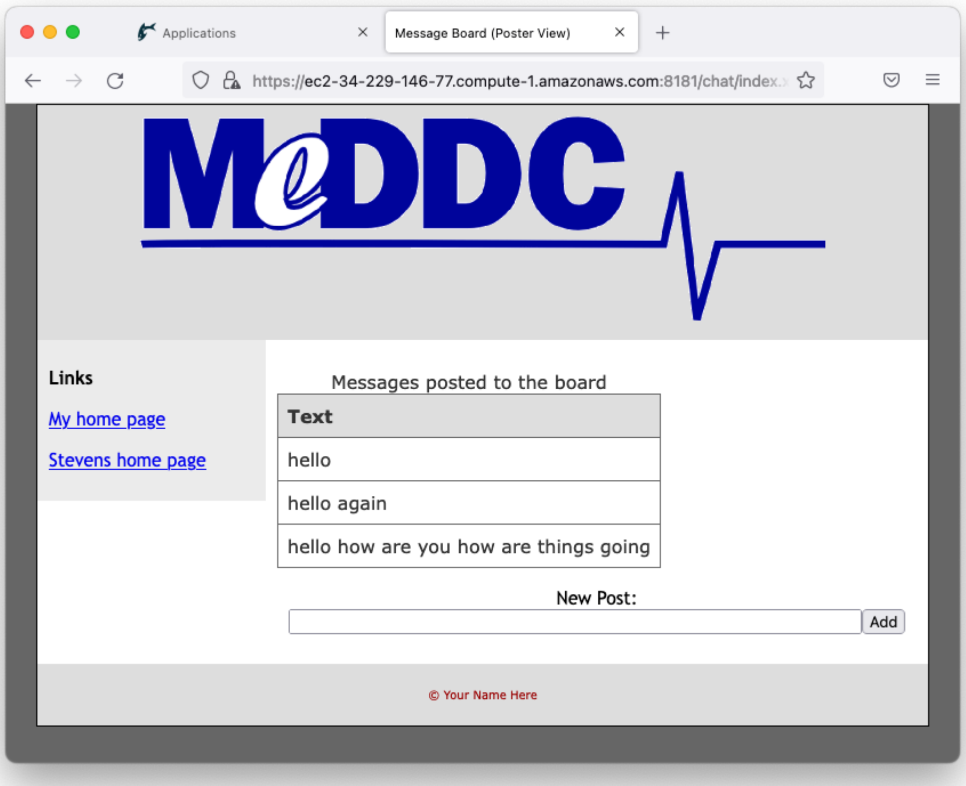
Context Root: chat-webapp (or as per glassfish-web.xml)
Path relative to server's base URL.

Application Name: * chat-webapp

Virtual Servers: server

Once you have successfully deployed the app, you can access it via the URL:

`https://`*`ec2-instance-public-dns-name`*`:8181/chat`


Applications
Message Board (Poster View)
https://ec2-34-229-146-77.compute-1.amazonaws.com:8181/chat/index.
MeDDC
Links
My home page
Stevens home page
New Post:
Add
© Your Name Here

Messages posted to the board

Text
hello
hello again
hello how are you how are things going

If you have any problems, you can view the raw logs using the `docker logs` command. For example, save the logs into a file and then view the last 100 lines of the file:

```
[ec2-user] docker logs container-id >& LOG
[ec2-user] tail -100 LOG
```

It is well worth your while learning at least some rudimentary bash commands, and how to view a file using the vim editor[7].

Step 8: Enable autostart of the database and application server on boot

As root, create the file `/etc/systemd/system/cs548db.service`:

```
[Unit]
Wants=docker.service
After=docker.service
```

[7] Bash and vim are available in MacOS. On Windows, install Cygwin.

```
[Service]
RemainAfterExit=yes
ExecStart=/usr/bin/docker start cs548db
ExecStop=/usr/bin/docker stop cs548db

[Install]
WantedBy=multi-user.target
```

Now you can start the database as a service:

```
$ sudo systemctl start cs548db
```

Enable the service to be executed during boot:

```
$ sudo systemctl enable cs548db
```

Similarly set up the Payara server to run as a boot service (call it cs548, for example).

Submission

Do the following to save your personal information in the instance:

```
cd
echo "YOUR-NAME" > info.txt
echo "YOUR-CWID" >> info.txt
echo "YOUR-EMAIL" >> info.txt
```

For your submission, provide videos and a report in PDF format that describes what you did, including each of the following:

a. Screenshot of your AWS Console showing the volumes you have allocated on EBS.
b. Output of Linux command "fdisk -l" in the instance.
c. Information on how to access your EC console using IAM permissions (see the additional specification for this).
d. The administrator password you chose for Payara. This password should be secure ("`abc123`" or "`admin`" or "`c548`" are not acceptable), but not ones that you use for any other business outside of this class.
e. Video of a demonstration of your deployment working. In this video, demonstrate the starting of the database server and the starting of the application server from the command line, using `docker start` and `docker ps`, then your logging in to the application server administration console, deployment of the `chat-webapp` application, and finally your running this application (adding some messages to test the database connection). Include your name in at least one message.
f. Provide a completed rubric with your submission (see the provided rubric).

You should not leave the EC2 instance running, since you may incur bills of hundreds of dollars from Amazon if you do this. Instead, leave your instance stopped, and use IAM to provide graders with access to your EC2 console, so they can start the instance. Both the database and application servers must automatically start when the instance boots. See the separate document detailing how to grant graders access to the instance.

You are also strongly encouraged to back up your instance as an Amazon machine image (AMI). This way, if the instance ever becomes corrupted in some way, you can instantiate a new version from the AMI that you have created. However you do not need to share the AMI with the graders, they will just be accessing the instance that you have stopped. Be sure to include, in the report for this and later assignment submissions, complete instructions on how to start the instance from your EC2 console. Payara and postgresql should start automatically when the instance starts, if you have set it up right.

Make sure that your name appears at the beginning of the video. For example, display the contents of a file that provides your name. *Do not provide private information such as your email or cwid in the video*. Be careful of any "free" apps that you download to do the recording, there are known cases of such apps containing Trojan horses, including key loggers. **Your video must be MP4 format!**

Your submission should be uploaded via the Canvas classroom, as a zip file. This zip file should have the same name as your Stevens username. It should unzip to a folder with this same name, which should contain the files and subfolders with your submission.

It is important that you provide a document that documents your submission, included as a PDF document in your submission root folder. Name this document README.pdf. This should document the video(s) that you provide demonstrating that you have correctly set up your cloud environment. You should also provide a completed rubric.