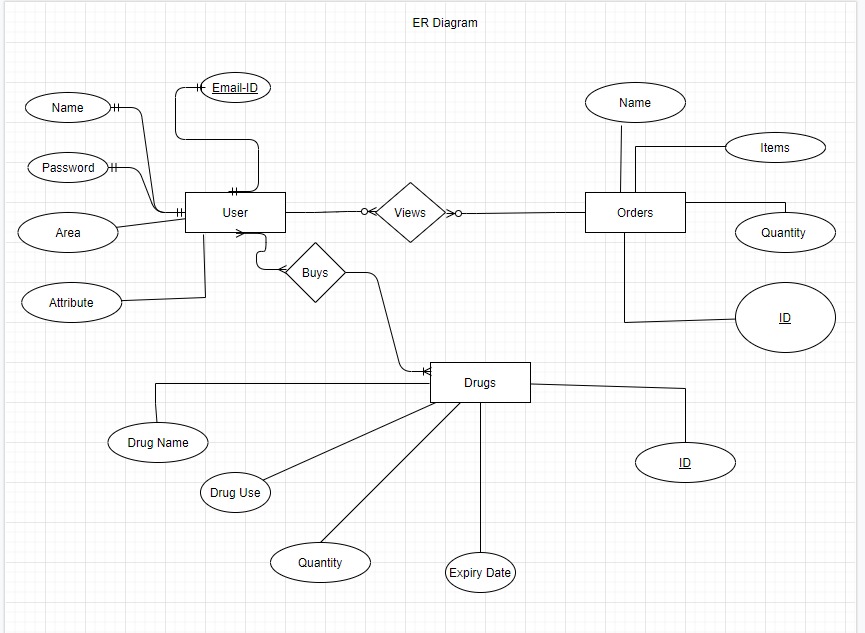
**ECO-SQUAD | CSE-D |5TH SEM** 

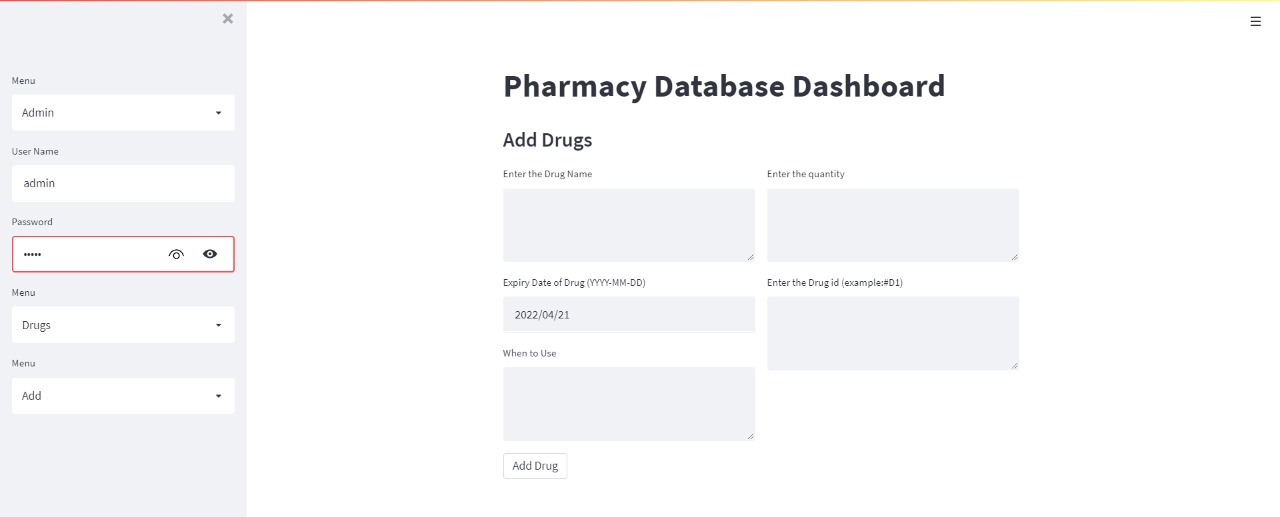
**PHARMACY MANAGEMENT SYSTEM**

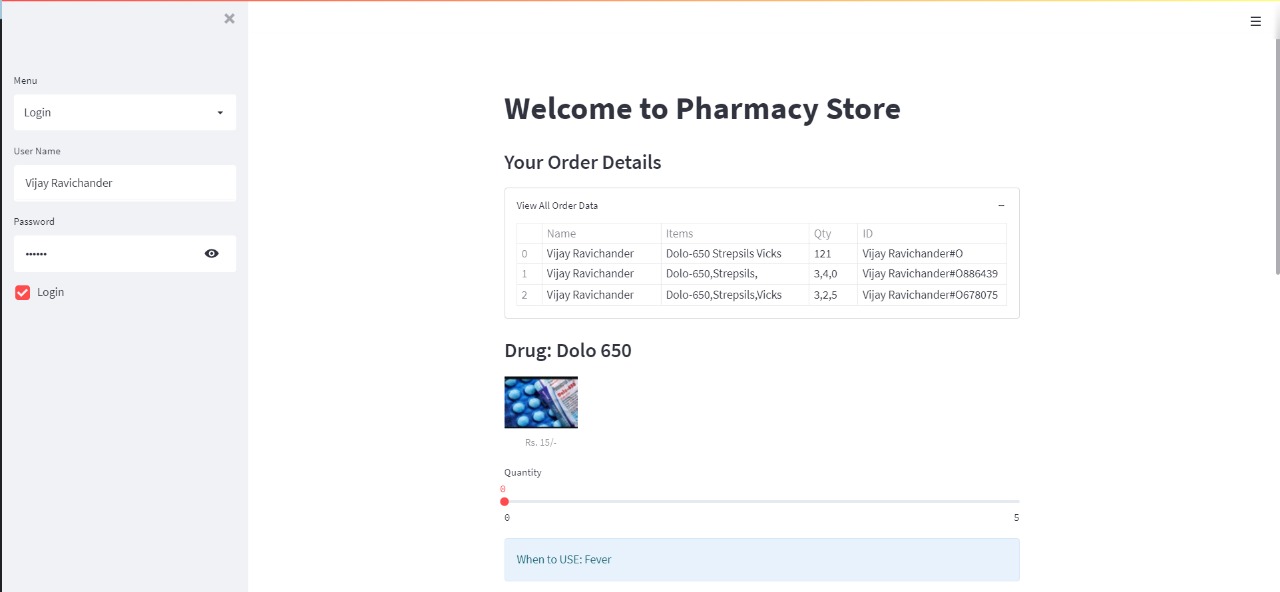
**TEAM MEMBERS:-**

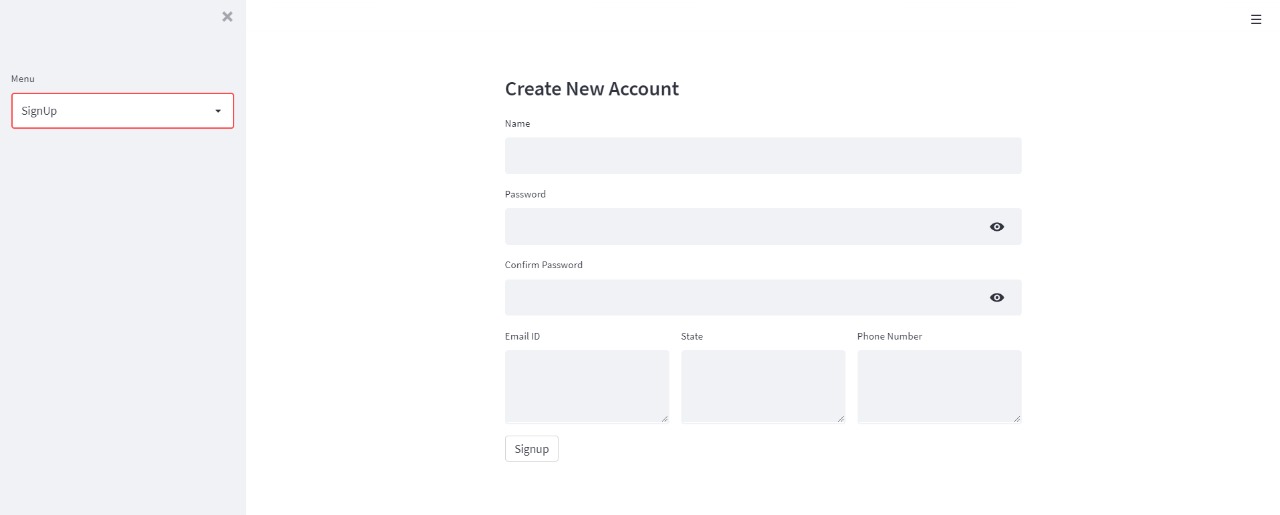
|  |  |
| --- | --- |
| NISCHITH S | 4NI22CS259 |
| ROHAN P N | 4NI22CS257 |
| SUHAS B H | 4NI23CS220 |
| SUHAS B M | 4NI22CS221 |
| THANMAN Mahesh | 4NI22CS236 |
| VISWAS H T | 4NI23CS248 |

**PHARMACY MANAGEMENT SYSTEM:-ER DIAGRAM**



**OUTPUT:**





***PYTHON CODE:-***

import streamlit as st

import pandas as pd

from PIL import Image

#from drug\_db import \*

import random

## SQL DATABASE CODE

import sqlite3

conn = sqlite3.connect("drug\_data.db",check\_same\_thread=False)

c = conn.cursor()

def cust\_create\_table():

    c.execute('''CREATE TABLE IF NOT EXISTS Customers(

                    C\_Name VARCHAR(50) NOT NULL,

                    C\_Password VARCHAR(50) NOT NULL,

                    C\_Email VARCHAR(50) PRIMARY KEY NOT NULL,

                    C\_State VARCHAR(50) NOT NULL,

                    C\_Number VARCHAR(50) NOT NULL

                    )''')

    print('Customer Table create Successfully')

def customer\_add\_data(Cname,Cpass, Cemail, Cstate,Cnumber):

    c.execute('''INSERT INTO Customers (C\_Name,C\_Password,C\_Email, C\_State, C\_Number) VALUES(?,?,?,?,?)''', (Cname,Cpass,  Cemail, Cstate,Cnumber))

    conn.commit()

def customer\_view\_all\_data():

    c.execute('SELECT \* FROM Customers')

    customer\_data = c.fetchall()

    return customer\_data

def customer\_update(Cemail,Cnumber):

    c.execute(''' UPDATE Customers SET C\_Number = ? WHERE C\_Email = ?''', (Cnumber,Cemail,))

    conn.commit()

    print("Updating")

def customer\_delete(Cemail):

    c.execute(''' DELETE FROM Customers WHERE C\_Email = ?''', (Cemail,))

    conn.commit()

def drug\_update(Duse, Did):

    c.execute(''' UPDATE Drugs SET D\_Use = ? WHERE D\_id = ?''', (Duse,Did))

    conn.commit()

def drug\_delete(Did):

    c.execute(''' DELETE FROM Drugs WHERE D\_id = ?''', (Did,))

    conn.commit()

def drug\_create\_table():

    c.execute('''CREATE TABLE IF NOT EXISTS Drugs(

                D\_Name VARCHAR(50) NOT NULL,

                D\_ExpDate DATE NOT NULL,

                D\_Use VARCHAR(50) NOT NULL,

                D\_Qty INT NOT NULL,

                D\_id INT PRIMARY KEY NOT NULL)

                ''')

    print('DRUG Table create Successfully')

def drug\_add\_data(Dname, Dexpdate, Duse, Dqty, Did):

    c.execute('''INSERT INTO Drugs (D\_Name, D\_Expdate, D\_Use, D\_Qty, D\_id) VALUES(?,?,?,?,?)''', (Dname, Dexpdate, Duse, Dqty, Did))

    conn.commit()

def drug\_view\_all\_data():

    c.execute('SELECT \* FROM Drugs')

    drug\_data = c.fetchall()

    return drug\_data

def order\_create\_table():

    c.execute('''

        CREATE TABLE IF NOT EXISTS Orders(

                O\_Name VARCHAR(100) NOT NULL,

                O\_Items VARCHAR(100) NOT NULL,

                O\_Qty VARCHAR(100) NOT NULL,

                O\_id VARCHAR(100) PRIMARY KEY NOT NULL)

    ''')

def order\_delete(Oid):

    c.execute('DELETE FROM Orders WHERE O\_id = ?', (Oid,))

    conn.commit()

def order\_add\_data(O\_Name,O\_Items,O\_Qty,O\_id):

    c.execute('''INSERT INTO Orders (O\_Name, O\_Items,O\_Qty, O\_id) VALUES(?,?,?,?)''',

              (O\_Name,O\_Items,O\_Qty,O\_id))

    conn.commit()

def order\_view\_data(customername):

    c.execute('SELECT \* FROM ORDERS Where O\_Name == ?',(customername,))

    order\_data = c.fetchall()

    return order\_data

def order\_view\_all\_data():

    c.execute('SELECT \* FROM Orders')

    order\_all\_data = c.fetchall()

    return order\_all\_data

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

def admin():

    st.title("Pharmacy Database Dashboard")

    menu = ["Drugs", "Customers", "Orders", "About"]

    choice = st.sidebar.selectbox("Menu", menu)

    ## DRUGS

    if choice == "Drugs":

        menu = ["Add", "View", "Update", "Delete"]

        choice = st.sidebar.selectbox("Menu", menu)

        if choice == "Add":

            st.subheader("Add Drugs")

            col1, col2 = st.columns(2)

            with col1:

                drug\_name = st.text\_area("Enter the Drug Name")

                drug\_expiry = st.date\_input("Expiry Date of Drug (YYYY-MM-DD)")

                drug\_mainuse = st.text\_area("When to Use")

            with col2:

                drug\_quantity = st.text\_area("Enter the quantity")

                drug\_id = st.text\_area("Enter the Drug id (example:#D1)")

            if st.button("Add Drug"):

                drug\_add\_data(drug\_name,drug\_expiry,drug\_mainuse,drug\_quantity,drug\_id)

                st.success("Successfully Added Data")

        if choice == "View":

            st.subheader("Drug Details")

            drug\_result = drug\_view\_all\_data()

            #st.write(drug\_result)

            with st.expander("View All Drug Data"):

                drug\_clean\_df = pd.DataFrame(drug\_result, columns=["Name", "Expiry Date", "Use", "Quantity", "ID"])

                st.dataframe(drug\_clean\_df)

            with st.expander("View Drug Quantity"):

                drug\_name\_quantity\_df = drug\_clean\_df[['Name','Quantity']]

                #drug\_name\_quantity\_df = drug\_name\_quantity\_df.reset\_index()

                st.dataframe(drug\_name\_quantity\_df)

        if choice == 'Update':

            st.subheader("Update Drug Details")

            d\_id = st.text\_area("Drug ID")

            d\_use = st.text\_area("Drug Use")

            if st.button(label='Update'):

                drug\_update(d\_use,d\_id)

        if choice == 'Delete':

            st.subheader("Delete Drugs")

            did = st.text\_area("Drug ID")

            if st.button(label="Delete"):

                drug\_delete(did)

    ## CUSTOMERS

    elif choice == "Customers":

        menu = ["View", "Update", "Delete"]

        choice = st.sidebar.selectbox("Menu", menu)

        if choice == "View":

            st.subheader("Customer Details")

            cust\_result = customer\_view\_all\_data()

            #st.write(cust\_result)

            with st.expander("View All Customer Data"):

                cust\_clean\_df = pd.DataFrame(cust\_result, columns=["Name", "Password","Email-ID" ,"Area", "Number"])

                st.dataframe(cust\_clean\_df)

        if choice == 'Update':

            st.subheader("Update Customer Details")

            cust\_email = st.text\_area("Email")

            cust\_number = st.text\_area("Phone Number")

            if st.button(label='Update'):

                customer\_update(cust\_email,cust\_number)

        if choice == 'Delete':

            st.subheader("Delete Customer")

            cust\_email = st.text\_area("Email")

            if st.button(label="Delete"):

                customer\_delete(cust\_email)

    elif choice == "Orders":

        order\_menu = ["View", "Delete"]

        order\_choice = st.sidebar.selectbox("Order Actions", order\_menu)

        # Set session state flag for order deletion tracking

        if 'order\_deleted' not in st.session\_state:

            st.session\_state.order\_deleted = False

        if order\_choice == "View":

            st.subheader("Order Details")

            # If an order was recently deleted, refresh data

            if st.session\_state.order\_deleted:

                order\_result = order\_view\_all\_data()  # Re-fetch the latest order data

                st.session\_state.order\_deleted = False  # Reset the deletion flag

            else:

                order\_result = order\_view\_all\_data()  # Load order data if no recent deletion

            # Display the updated order data

            with st.expander("View All Order Data"):

                order\_clean\_df = pd.DataFrame(order\_result, columns=["Name", "Items", "Qty", "ID"])

                st.dataframe(order\_clean\_df)

        elif order\_choice == "Delete":

            st.subheader("Delete an Order")

            order\_id = st.text\_input("Enter Order ID to delete")

            if st.button("Delete Order"):

                order\_delete(order\_id)

                st.success(f"Order with ID '{order\_id}' has been deleted successfully.")

                # Set the deletion flag to true to refresh "View" on next load

                st.session\_state.order\_deleted = True

    elif choice == "About":

        st.subheader("DBMS Mini Project")

        st.subheader("By")

        st.subheader("Nischith S (259)")

        st.subheader("Suhas B M (221)")

        st.subheader("Suhas B H (220)")

        st.subheader("Rohan P N (257)")

        st.subheader("Vishwas H T (248)")

        st.subheader("Thanman Mahesh (236)")

def getauthenicate(username, password):

    c.execute('SELECT C\_Password FROM Customers WHERE C\_Name = ?', (username,))

    cust\_password = c.fetchall()

    # Check if any result is returned

    if cust\_password and cust\_password[0][0] == password:

        return True

    else:

        return False

###################################################################

def customer(username, password):

    if getauthenicate(username, password):

        print("In Customer")

        st.title("Welcome to Pharmacy Store")

        st.subheader("Your Order Details")

        order\_result = order\_view\_data(username)

        # st.write(cust\_result)

        with st.expander("View All Order Data"):

            order\_clean\_df = pd.DataFrame(order\_result, columns=["Name", "Items", "Qty", "ID"])

            st.dataframe(order\_clean\_df)

        drug\_result = drug\_view\_all\_data()

        print(drug\_result)

        st.subheader("Drug: "+drug\_result[0][0])

        img = Image.open('images/dolo650.jpg')

        st.image(img, width=100, caption="Rs. 15/-")

        dolo650 = st.slider(label="Quantity",min\_value=0, max\_value=5, key= 1)

        st.info("When to USE: " + str(drug\_result[0][2]))

        st.subheader("Drug: " + drug\_result[1][0])

        img = Image.open('images/strepsils.JPG')

        st.image(img, width=100 , caption="Rs. 10/-")

        strepsils = st.slider(label="Quantity",min\_value=0, max\_value=5, key= 2)

        st.info("When to USE: " + str(drug\_result[1][2]))

st.subheader("Drug: " + drug\_result[2][0])

        img = Image.open('images/vicks.JPG')

        st.image(img, width=100, caption="Rs. 65/-")

        vicks = st.slider(label="Quantity",min\_value=0, max\_value=5, key=3)

        st.info("When to USE: " + str(drug\_result[2][2]))

        if st.button(label="Buy now"):

            O\_items = ""

            if int(dolo650) > 0:

                O\_items += "Dolo-650,"

            if int(strepsils) > 0:

                O\_items += "Strepsils,"

            if int(vicks) > 0:

                O\_items += "Vicks"

            O\_Qty = str(dolo650)+str(',') + str(strepsils) + str(",") + str(vicks)

            O\_id = username + "#O" + str(random.randint(0,1000000))

            #order\_add\_data(O\_Name, O\_Items,O\_Qty, O\_id):

            order\_add\_data(username, O\_items, O\_Qty, O\_id)

if \_\_name\_\_ == '\_\_main\_\_':

    drug\_create\_table()

    cust\_create\_table()

    order\_create\_table()

    menu = ["Login", "SignUp","Admin"]

    choice = st.sidebar.selectbox("Menu", menu)

    if choice == "Login":

        username = st.sidebar.text\_input("User Name")

        password = st.sidebar.text\_input("Password", type='password')

        if st.sidebar.checkbox(label="Login"):

            customer(username, password)

    elif choice == "SignUp":

        st.subheader("Create New Account")

        cust\_name = st.text\_input("Name")

        cust\_password = st.text\_input("Password", type='password', key=1000)

        cust\_password1 = st.text\_input("Confirm Password", type='password', key=1001)

        col1, col2, col3 = st.columns(3)

with col1:

            cust\_email = st.text\_area("Email ID")

        with col2:

            cust\_area = st.text\_area("State")

        with col3:

            cust\_number = st.text\_area("Phone Number")

        if st.button("Signup"):

            if (cust\_password == cust\_password1):

                customer\_add\_data(cust\_name,cust\_password,cust\_email, cust\_area, cust\_number,)

                st.success("Account Created!")

                st.info("Go to Login Menu to login")

            else:

                st.warning('Password dont match')

    elif choice == "Admin":

        username = st.sidebar.text\_input("User Name")

        password = st.sidebar.text\_input("Password", type='password')

        # if st.sidebar.button("Login"):

        if username == 'admin' and password == 'admin':

admin()

**SQL CODE:-**

CREATE SCHEMA drugdatabase;

USE drugdatabase;

CREATE TABLE customer (

uid varchar(20) NOT NULL,

pass varchar(20) DEFAULT NULL,

fname varchar(15) DEFAULT NULL,

lname varchar(15) DEFAULT NULL,

email varchar(30) DEFAULT NULL,

address varchar(128) DEFAULT NULL,

phno bigint DEFAULT NULL,

PRIMARY KEY (uid)

);

CREATE TABLE seller (

sid varchar(15) NOT NULL,

sname varchar(20) DEFAULT NULL,

pass varchar(20) DEFAULT NULL,

address varchar(128) DEFAULT NULL,

phno bigint DEFAULT NULL,

PRIMARY KEY (sid)

);

CREATE TABLE product (

pid varchar(15) NOT NULL,

pname varchar(20) DEFAULT NULL,

manufacturer varchar(20) DEFAULT NULL,

mfg date DEFAULT NULL,

exp date DEFAULT NULL,

price int DEFAULT NULL,

PRIMARY KEY (pid),

UNIQUE KEY pname (pname)

);

CREATE TABLE inventory (

pid varchar(15) NOT NULL,

pname varchar(20) DEFAULT NULL,

quantity int unsigned DEFAULT NULL,

sid varchar(15) NOT NULL,

PRIMARY KEY (pid,sid),

CONSTRAINT fk01 FOREIGN KEY (pid) REFERENCES product (pid) ON DELETE CASCADE,

CONSTRAINT fk02 FOREIGN KEY (pname) REFERENCES product (pname) ON DELETE CASCADE,

CONSTRAINT fk03 FOREIGN KEY (sid) REFERENCES seller (sid) ON DELETE CASCADE

);

CREATE TABLE orders (

oid int NOT NULL AUTO\_INCREMENT,

pid varchar(15) DEFAULT NULL,

sid varchar(15) DEFAULT NULL,

uid varchar(15) DEFAULT NULL,

orderdatetime datetime DEFAULT NULL,

quantity int unsigned DEFAULT NULL,

price int unsigned DEFAULT NULL,

PRIMARY KEY (oid),

CONSTRAINT fk04 FOREIGN KEY (pid) REFERENCES product (pid) ON DELETE CASCADE,

CONSTRAINT fk05 FOREIGN KEY (sid) REFERENCES seller (sid) ON DELETE CASCADE,

CONSTRAINT fk06 FOREIGN KEY (uid) REFERENCES customer (uid) ON DELETE CASCADE

);

ALTER TABLE orders AUTO\_INCREMENT=1000;

DELIMITER //

CREATE TRIGGER updatetime BEFORE INSERT ON orders FOR EACH ROW

BEGIN

SET NEW.orderdatetime = NOW();

END//

DELIMITER ;

DELIMITER //

CREATE TRIGGER inventorytrigger AFTER INSERT ON orders

FOR EACH ROW

begin

DECLARE qnty int;

DECLARE productid varchar(20);

SELECT pid INTO productid

FROM orders

ORDER BY oid DESC

LIMIT 1;

SELECT quantity INTO qnty

FROM orders

ORDER BY oid DESC

LIMIT 1;

UPDATE inventory

SET quantity=quantity-qnty

WHERE pid=productid;

END//

DELIMITER ;

DELIMITER //

CREATE PROCEDURE getsellerorders(IN param1 VARCHAR(20))

BEGIN

SELECT \* FROM orders where sid=param1;

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE getorders

(IN param1 VARCHAR(20))

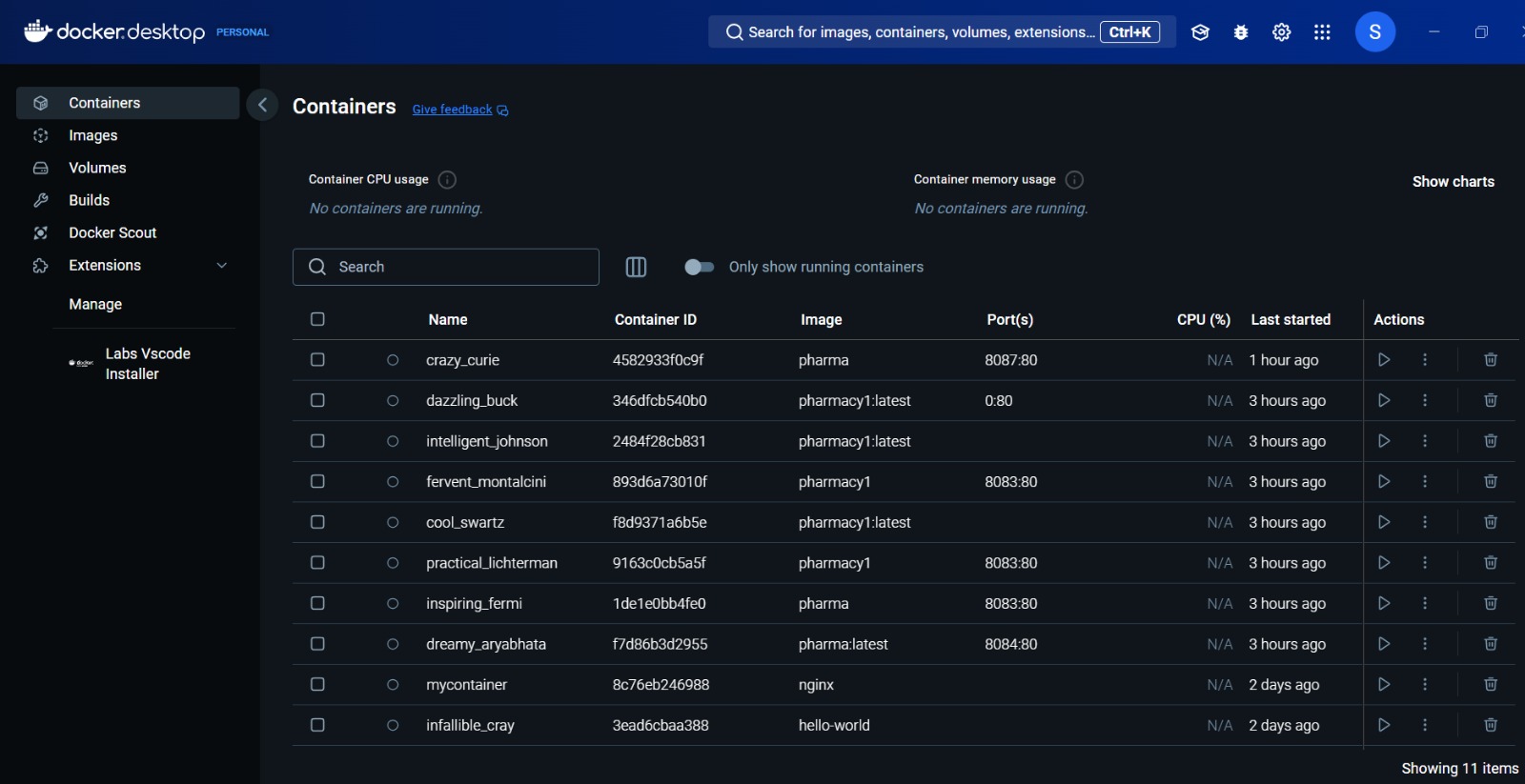
BEGIN

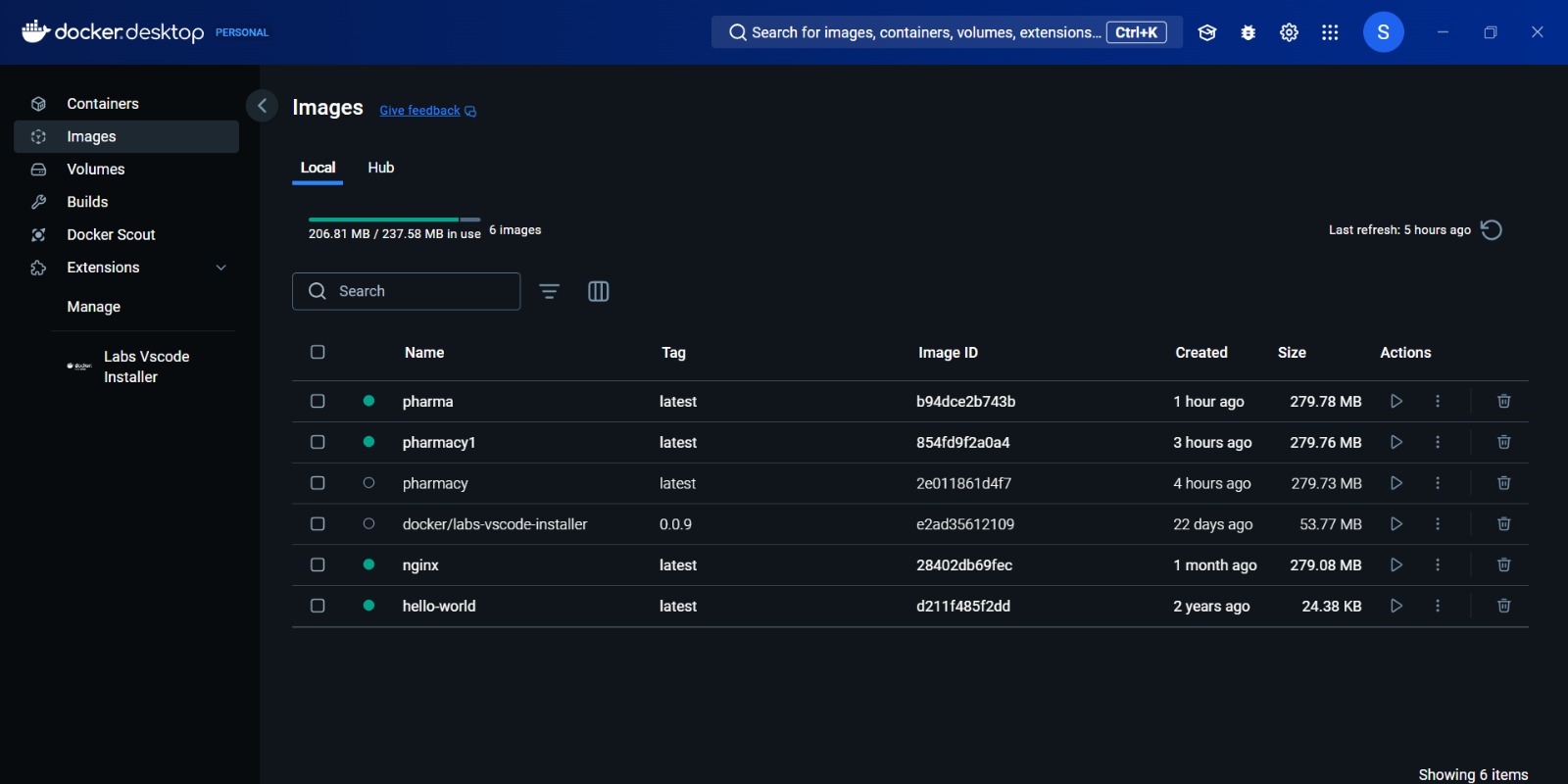
SELECT \* FROM orders WHERE uid=param1;

END //

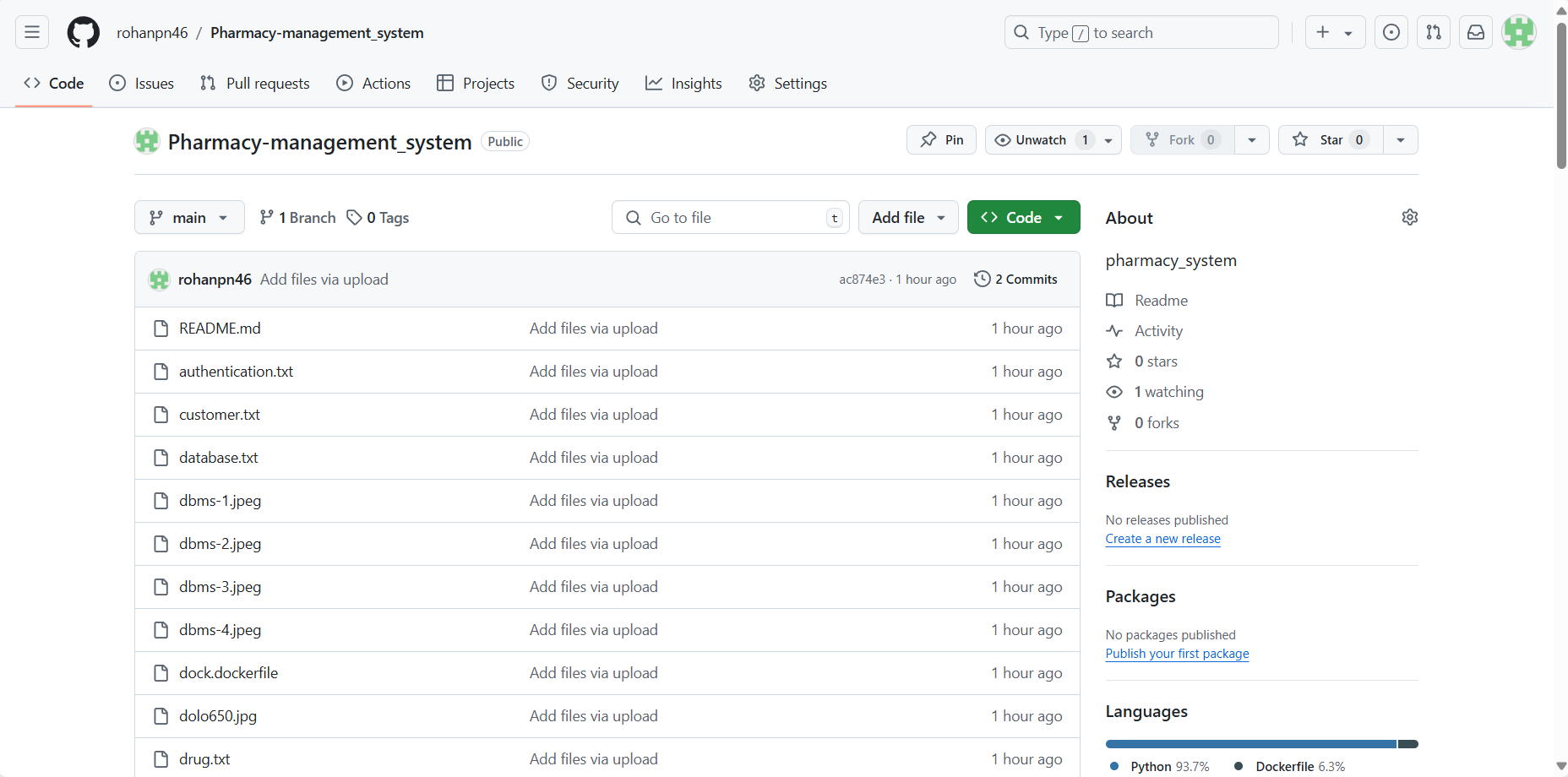
DELIMITER ;

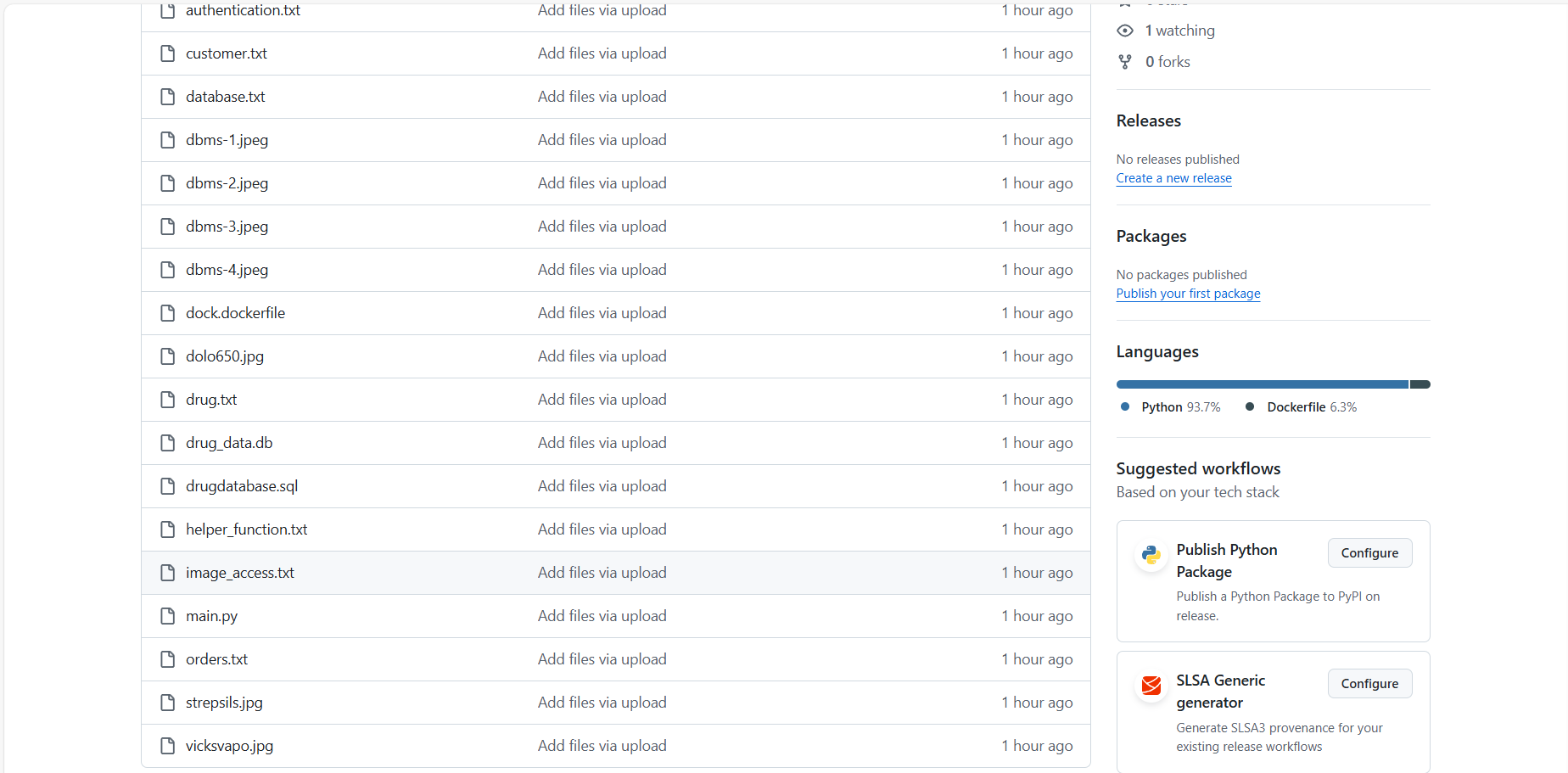
**DOCKER :-**

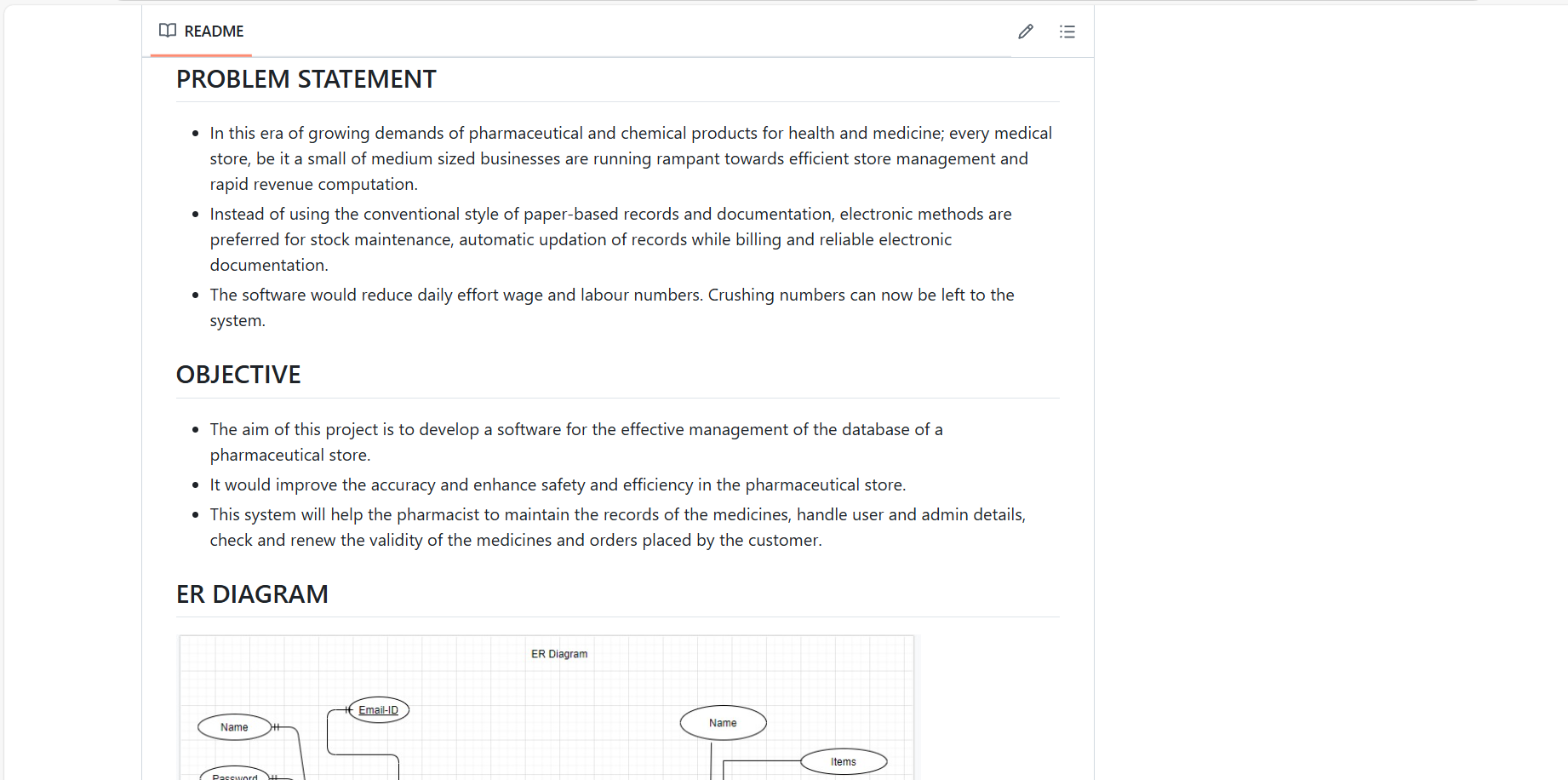


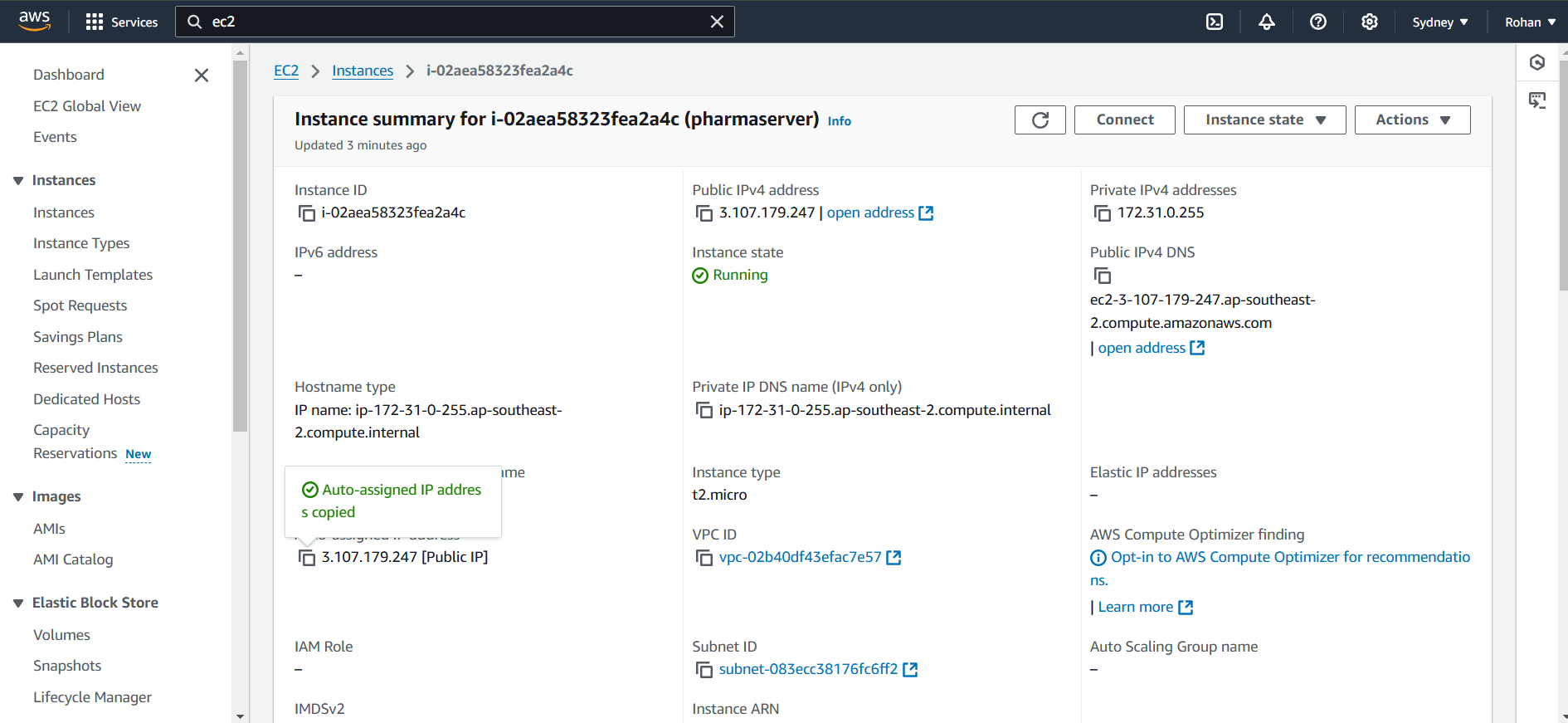


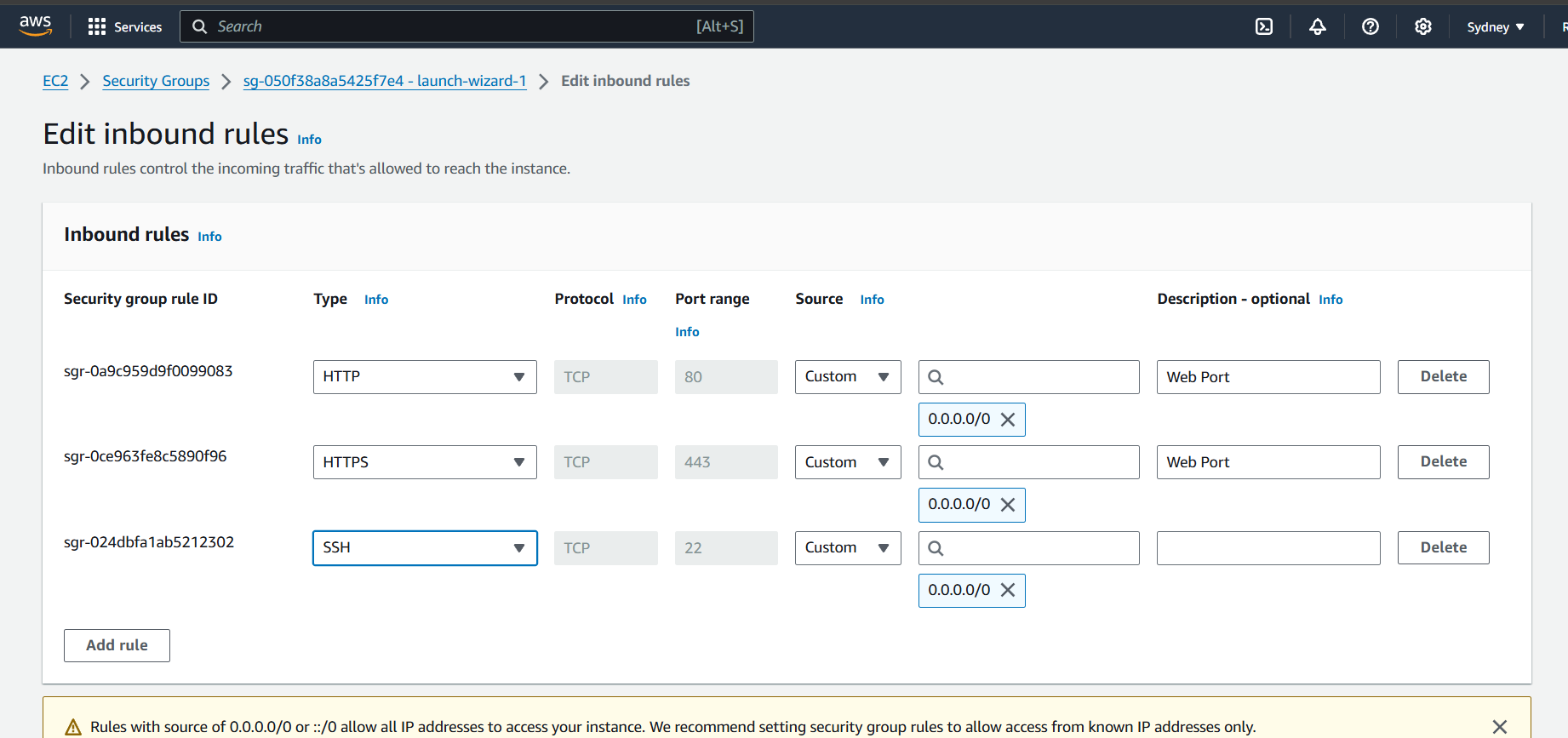
**GITHUB:-**

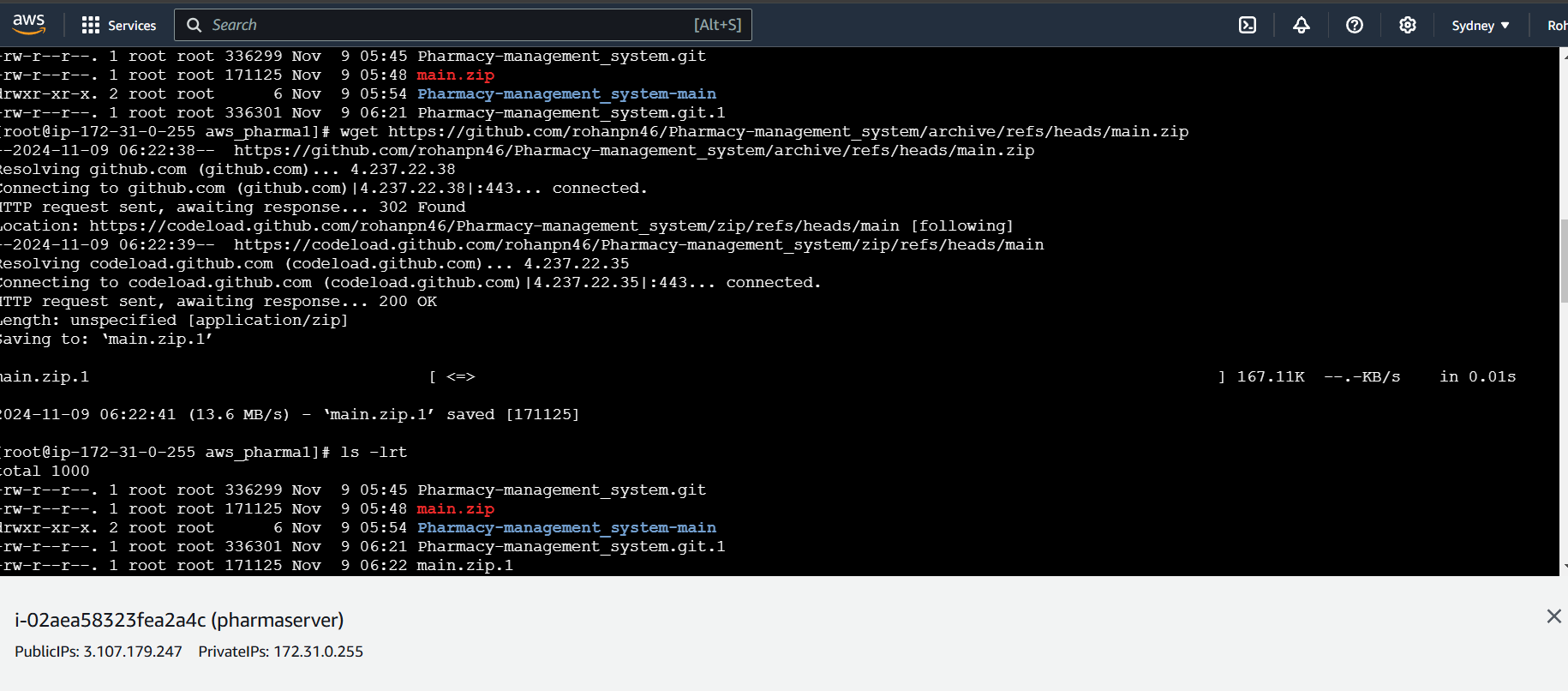
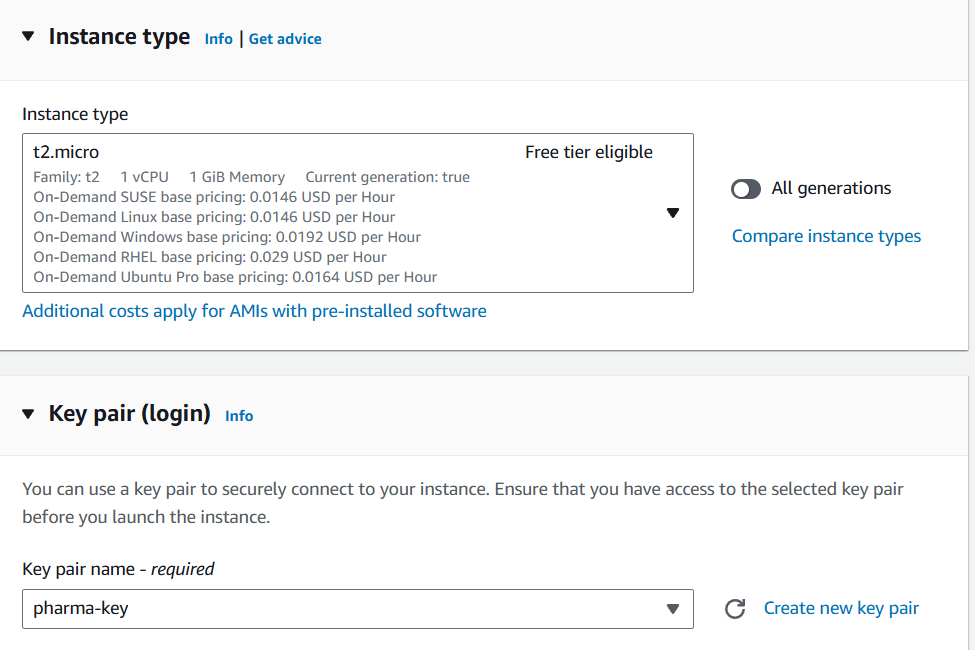
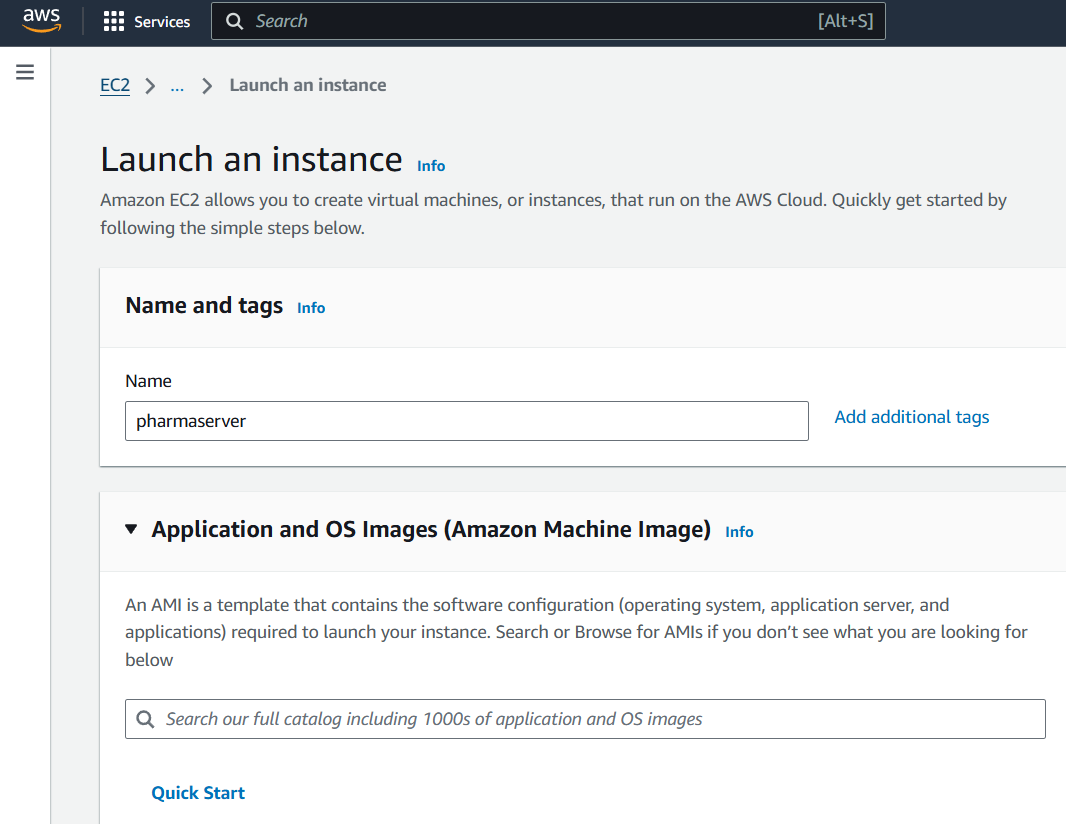
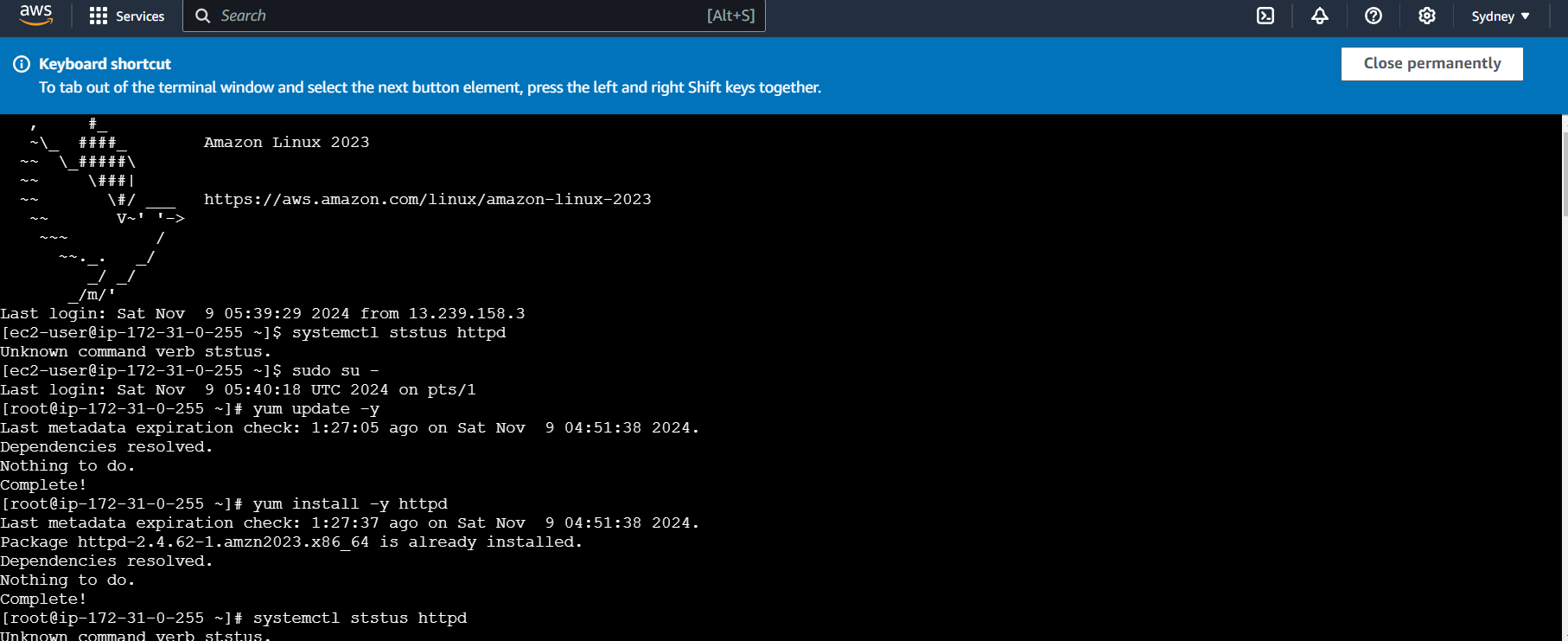
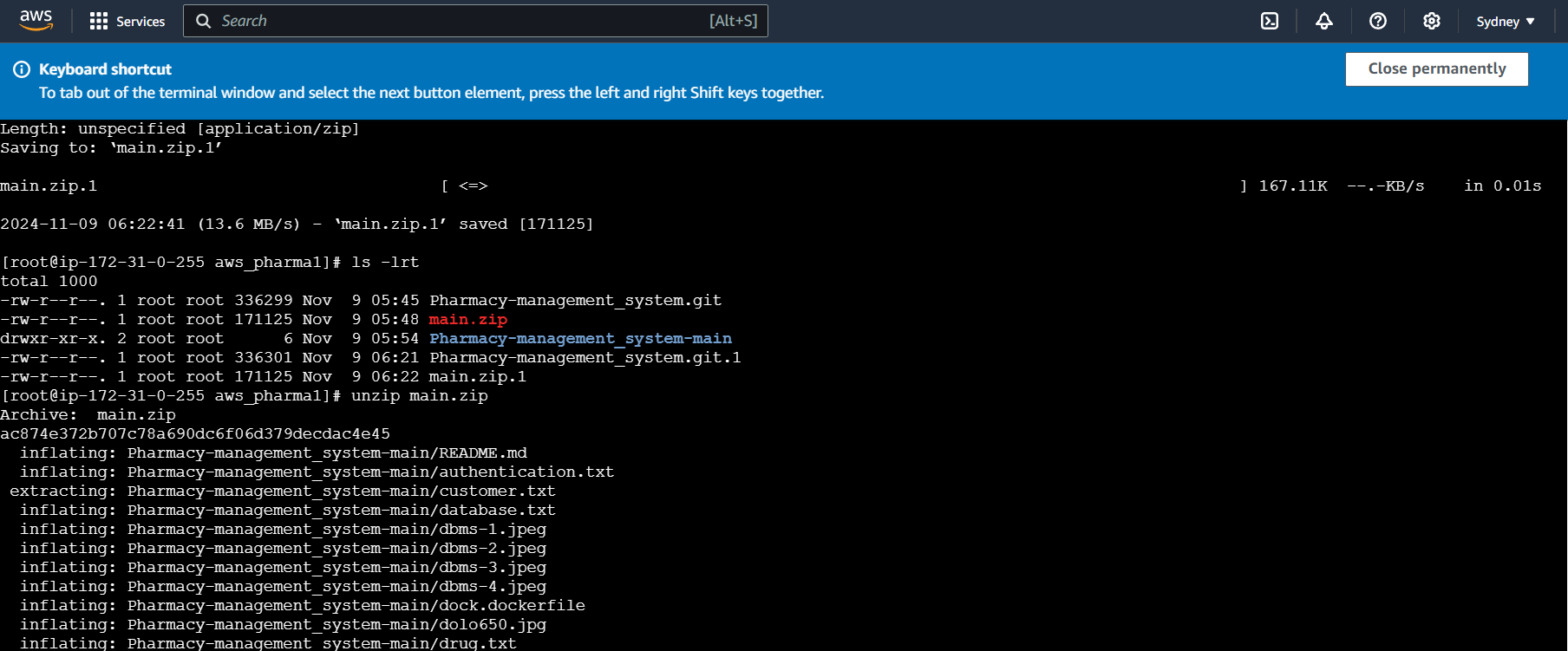
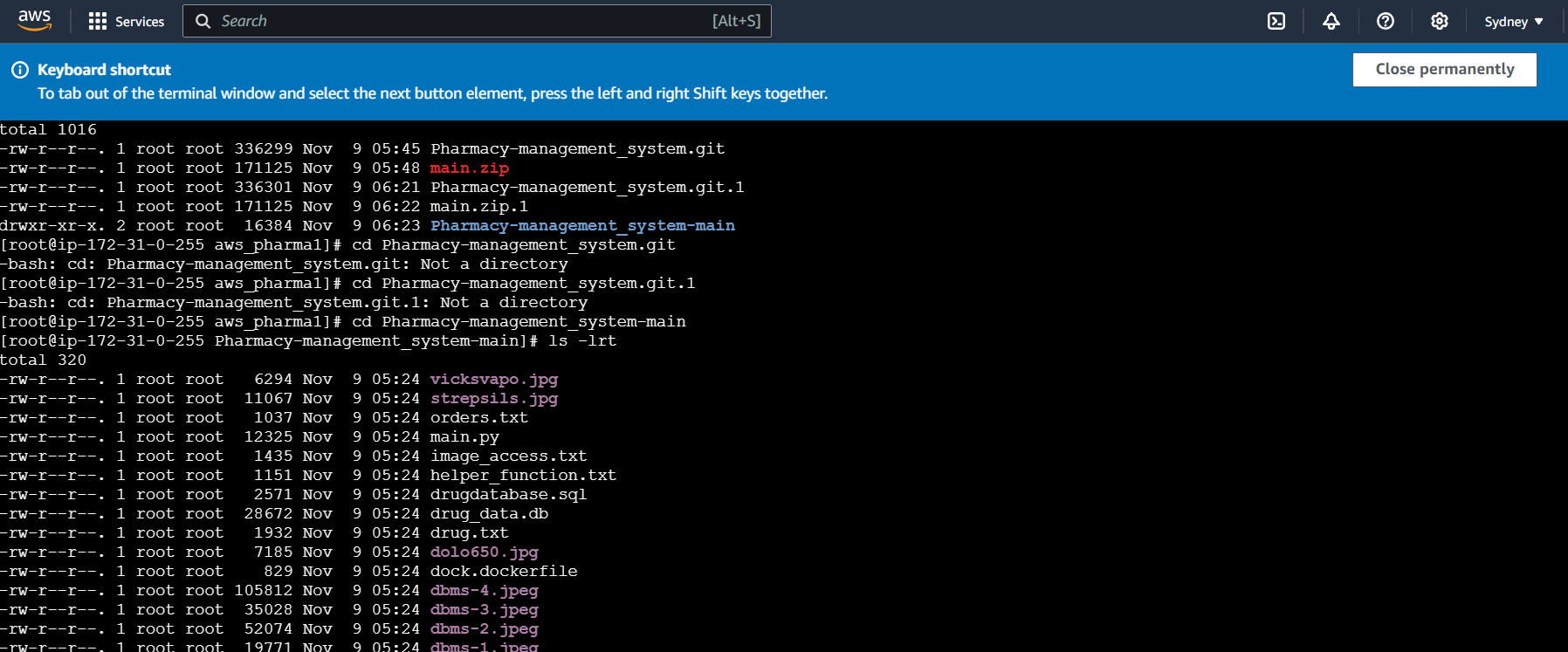
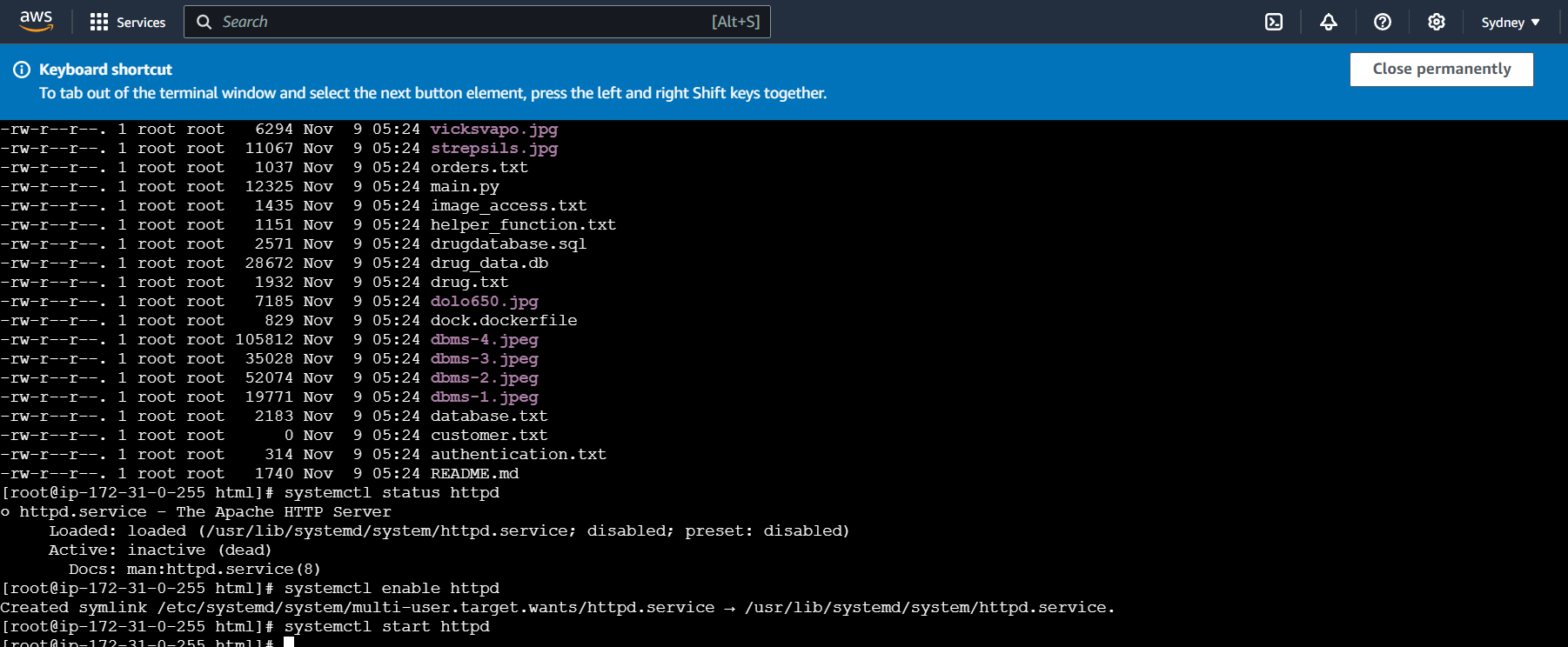
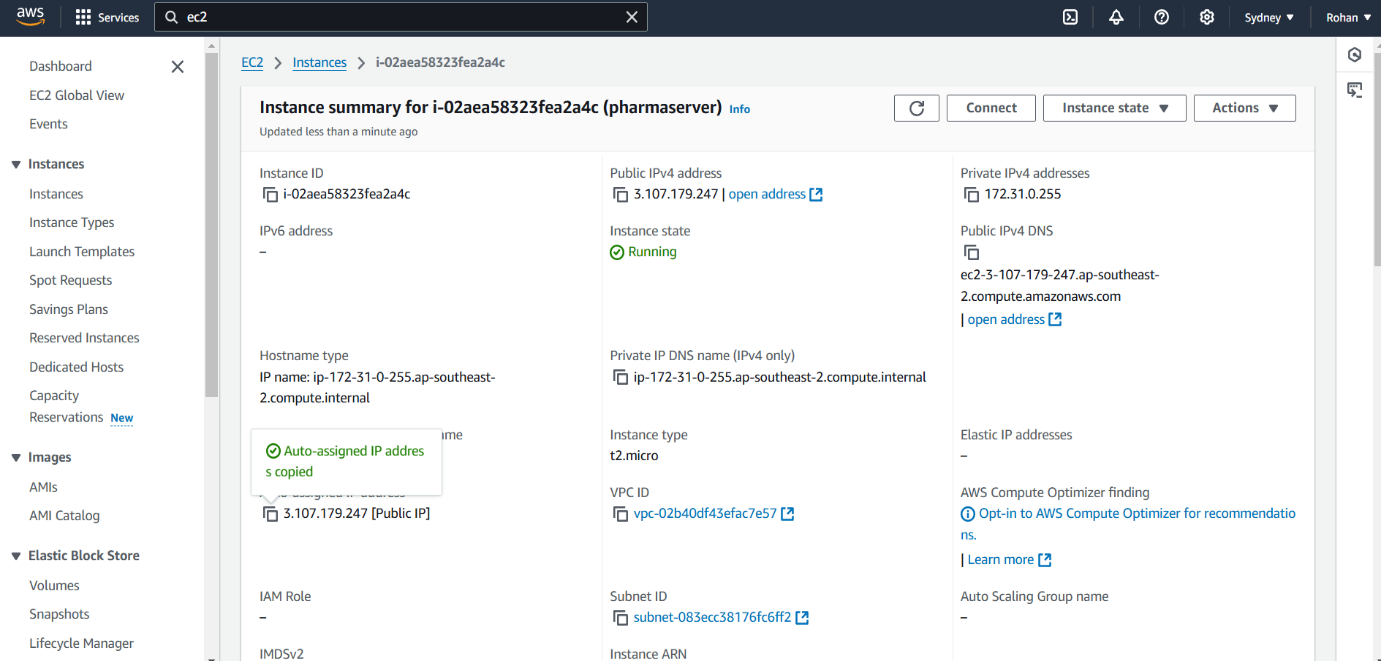
****





**AWS:-**





**NETLIFY:-**