Cloud and DevOps Lab Experiments Report

1) Launch EC2 Instance, Configure CloudWatch and CloudTrail

Step 1: Launch EC2 Instance

- Go to AWS Console > EC2 > Launch Instance

- Choose Amazon Linux AMI, t2.micro, and configure security group with SSH port 22 open.

Step 2: Install CloudWatch Agent

$ sudo yum update -y

$ sudo yum install -y amazon-cloudwatch-agent

Step 3: Create CloudWatch Agent Config

$ sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-config-wizard

Step 4: Start CloudWatch Agent

$ sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a start

Step 5: Enable CloudTrail

- Go to CloudTrail > Create trail

- Choose S3 bucket and enable for all regions

Step 6: Simulate Activity

- SSH into instance using: ssh -i key.pem ec2-user@<public-ip>

- CloudTrail logs login and API actions

Step 7: View Logs

- Go to CloudTrail > Event history

- Go to CloudWatch > Logs > View log group created

2) Git Operations: Clone, Commit, Push, Fetch, Pull, Checkout, Reset, Delete Branch

Step 1: Clone Repo

$ git clone https://github.com/your/repo.git

Step 2: Make Changes

$ echo "New Line" >> test.txt

Step 3: Add and Commit

$ git add test.txt

$ git commit -m "Added new line"

Step 4: Push Changes

$ git push origin main

Step 5: Fetch and Pull

$ git fetch origin

$ git pull origin main

Step 6: Checkout Branch

$ git checkout -b new-feature

Step 7: Reset Commit

$ git reset --hard HEAD~1

Step 8: Delete Branch

$ git branch -d new-feature

$ git push origin --delete new-feature

3) Clone GitHub Repo, Modify File, Commit, Push, View Git Log

Step 1: Clone Repository

$ git clone https://github.com/your/repo.git

Step 2: Modify a File

$ echo "Modified line" >> index.html

Step 3: Commit Changes

$ git add index.html

$ git commit -m "Updated index"

Step 4: Push to GitHub

$ git push origin main

Step 5: View Git Log

$ git log --oneline

4) Simulate Brute-force SSH Login and Detect with CloudTrail/CloudWatch

Step 1: Simulate Brute-force

Run from another system:

$ for i in {1..10}; do ssh wrong@<ec2-ip>; done

Step 2: Enable CloudTrail (if not already)

- Go to CloudTrail > Trails > Create trail

Step 3: Send Auth Logs to CloudWatch

- Edit CloudWatch agent config to include /var/log/secure (Amazon Linux)

Step 4: Create Metric Filter

- Go to CloudWatch > Logs > Metric filters

- Filter pattern: "Failed password"

Step 5: Create Alarm

- CloudWatch > Alarms > Create alarm on metric filter

- Action: Send notification via SNS

Step 6: Check Logs

- View log stream for SSH failure entries

5) Dockerize a Simple Web Application

Step 1: Create Files

- app.py:

```
from flask import Flask
app = Flask(__name__)
```

@app.route('/')

def hello(): return "Hello from Docker!"

if __name__ == "__main__": app.run(host="0.0.0.0", port=5000)

- requirements.txt:

flask

Step 2: Dockerfile

FROM python:3.9

WORKDIR /app

COPY . .

RUN pip install -r requirements.txt

EXPOSE 5000

CMD ["python", "app.py"]

Step 3: Build Docker Image

$ docker build -t flask-web-app .

Step 4: Run Container

$ docker run -d -p 5000:5000 flask-web-app

Step 5: Access in Browser

Go to http://localhost:5000

6) Create Git Repository & Perform Git Commands

- git init
- git remote add origin <repo-url>
- git clone <repo-url>
- Modify file, then: git add ., git commit -m "msg", git push
- Fetch remote changes: git fetch
- Pull latest: git pull
- Create/switch branch: git checkout -b feature
- Reset: git reset --hard HEAD~1
- Delete branch: git branch -d feature, git push origin --delete feature

Viva Questions

1. What is the difference between CloudWatch and CloudTrail?
2. How can you detect unauthorized access in AWS?
3. What is the difference between git fetch and git pull?
4. How does Docker containerization work?
5. What is the use of a Dockerfile?
6. How do metric filters work in CloudWatch?
7. How do you secure EC2 SSH access?
8. What is the purpose of git reset?
9. How can CloudTrail logs be stored securely?
10. How do you check for brute-force attempts using logs?