Configuration Manual

MSc Research Project
Cloud Computing

Hamza Niaz
Student ID: 23251298

School of Computing
National College of Ireland

Supervisor: Shreyas Setlur Arun

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Hamza Niaz
Student ID:	23251298
Programme:	MSc in Cloud Computing
Year:	2024
Module:	Msc Research Report
Lecturer:	Shreyas Setlur Arun
Submission Due Date:	29/01/2025
Project Title:	Blockchain and AWS Integration for Financial Transaction
Word Count:	1293
Page Count:	8

Signature:	Hamza
Date:	28/01/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	✓ □
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	✓ □
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	✓ □

Office Use Only	
Signature:	

Date:	
Penalty Applied (if applicable):	

Configuration Manual

Hamza Niaz
23251298

1. Hardware Resources for the Experiment are listed below:

➢ Okay, let's get started withAWS EC2 (Elastic Compute Cloud) Instance

- Instance Type: t2.large
- vCPU: 2 vCPUs
- RAM: 8 GiB
- Storage: Elastic Block Store (EBS) volume with dynamic allocation based on demand.
- Network: Virtual Private Cloud (VPC) architecture with customized subnets for improved security
- Purpose: To host Hyperledger Fabric nodes.

➢ Local system:

- Processor: Like ex. Intel i5
- RAM: suggest 16 gb
- Storage: 500+ gb
- Disk Operating System: Ubuntu 20.04 for Docker or Go and Node. Js.

➢ Networking:

- Bandwidth: Min 10 Mbps
- Security: Security groups that are set up to allow specified ports (7050 for Hyperledger Fabric, 22 for SSH, 443 for HTTPS).

➢ Backup Storage:

- Blockchain data logs and backups AWS S3
- S3 CLASS FOR GENERAL PURPOSE STORAGE.

2. Software Resources for the Experiment are listed below:

➢ Operating Systems:

- Ubuntu 20.04:
- The Base OS for AWS EC2 instances and local development
- Chosen for compatibility with Blockchain tools such as Docker and Hyperledger Fabric.

➢ Blockchain Framework:

- Hyperledger Fabric:
- Version : Hyperledger Fabric v2 x

- Features: (custom consensus mechanisms via modularity, smart contract support (Go-written chaincode))

➢ Programming Languages:

- Go:
- Version: 1.16 or above.
- Writing chaincode (smart contracts)Purpose Node.js:
- Version: 14.x LTS or above.
- Purpose: Building the web front end and client-side applications.

➢ Development Tools:

  - Docker:
  - Version: 20.x or above.
  - Abstract: This model was made to containerize Hyperledger Fabric nodes so they may be deployed more easily.
  - Git:
  - It is a Version Control System for managing source code.

➢ AWS Services:

- EC2
- IAM
- Amazon Managed Blockchain AWS S3:
- Use: Logs and transaction backups storage AWS CloudWatch:
- Why: To monitor and log performance.
- AWS CloudTrail:
- Use Case: Auditing and monitoring API calls for security/compliance.

➢ Security Tools:

- SSL/TLS: To encrypt data in motion
- AWS Key Management Service (KMS): Data encryption at rest IAM (Identity and Access Management): To impose strict access policies.

➢ Additional Software:

- Hyperledger Fabric CA: Identity Management Certificate authority.
- cURL: For API testing.
- Postman: For working with APIs.

3. Procedure for the Experiment:

➢ First Set up the AWS account for my experiment

 Create an AWS account

➢ Now Install the required tools on my local system

➢ Now set up the AWS Services

- Now Lanch the EC2 Instance and Navigate the EC2 Dashboard in Aws Console. The instance we select is t2.large and choose the operating system Ubuntu 20.04. Now we set the security group and allow ports 22(SSH), (Hyperledger Fabric)7050 and (HTTPS)443. The below image shows available and launch new instances. We use this for handle blockchain transations.

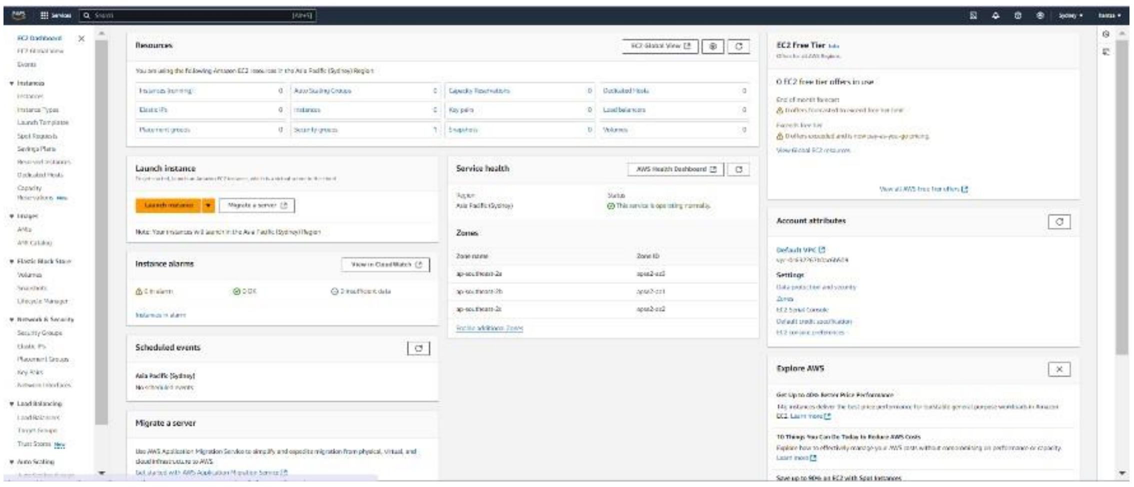
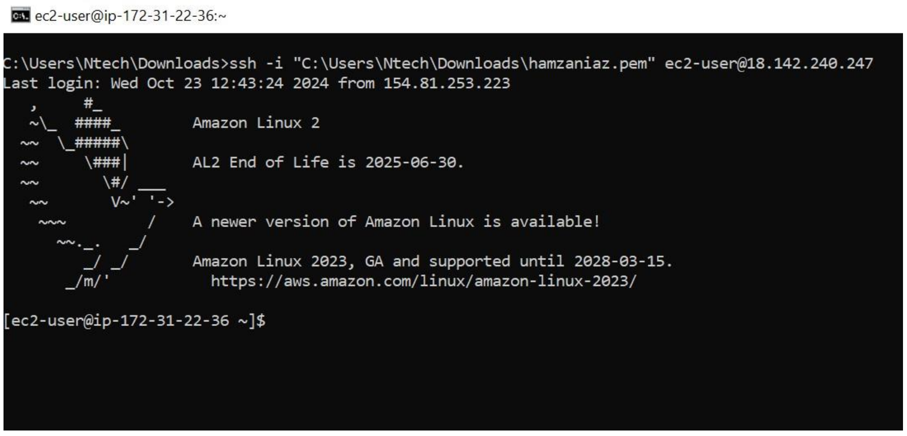


Figure 1: EC2 Dashboard

➢ Now we give the EC2 connection which is shown in the figure.

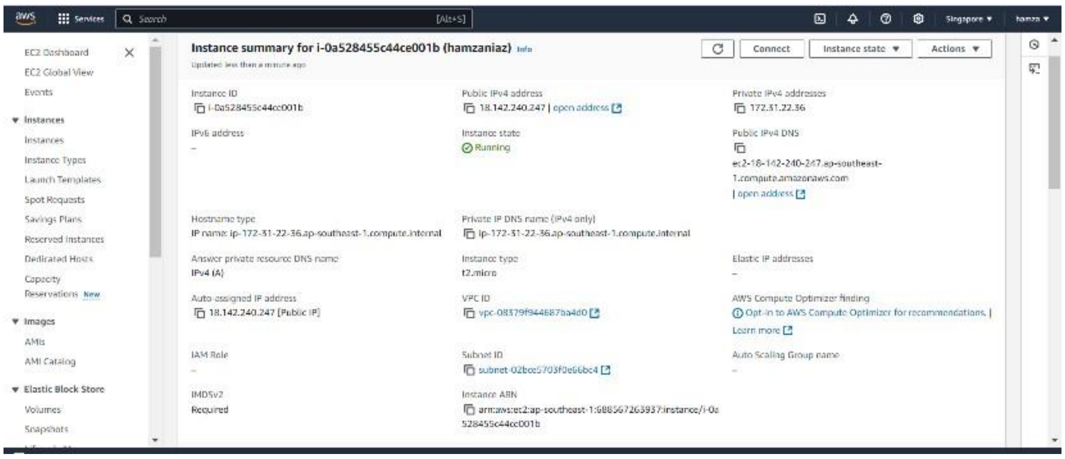


```
ec2-user@ip-172-31-22-36:~

C:\Users\Ntech\Downloads>ssh -i "C:\Users\Ntech\Downloads\hamzaniaz.pem" ec2-user@18.142.240.247
Last login: Wed Oct 23 12:43:24 2024 from 154.81.253.223
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2025-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-22-36 ~]$
```

Figure 2: EC2 Connection

➢ Now we set up blockchain on AWS
  - We use the Amazon Managed Blockchain service of the AWS where we can create the blockchain network and this service offer a popular frameworks that is Hyperledger Fabric and Ethereum and we have the freedom to choose the framework so we use hyperledger fabric for our experiment.

➢ Now we set up the EC2 instance for blockchain node
  - First we create an EC2 instance for the blockchain node which is the heart of a given blockchain network. The below figure show the summary of the EC2 instance for processing transactions on the blockchain network in the blockchain node.

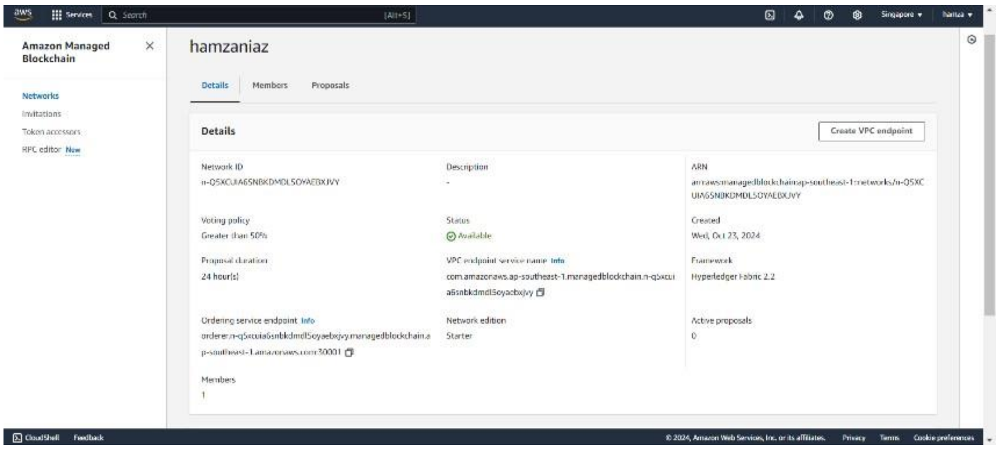

Instance summary for i-0a528455c44ce001b (hamzaniaz)

Updated less than a minute ago

Instance ID: i-0a528455c44ce001b
Public IPv4 address: 18.142.240.247 | open address
Private IPv4 addresses: 172.31.22.36
IPv6 address: –
Instance state: Running
Public IPv4 DNS: ec2-18-142-240-247.ap-southeast-1.compute.amazonaws.com | open address
Hostname type: IP name: ip-172-31-22-36.ap-southeast-1.compute.internal
Private IP DNS name (IPv4 only): ip-172-31-22-36.ap-southeast-1.compute.internal
Answer private resource DNS name: IPv4 (A)
Instance type: t2.micro
Elastic IP addresses: –
Auto-assigned IP address: 18.142.240.247 [Public IP]
VPC ID: vpc-08379f944687ba4d0
AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role: –
Subnet ID: subnet-02bce5703f0e66bc4
Auto Scaling Group name: –
IMDSv2: Required
Instance ARN: arn:aws:ec2:ap-southeast-1:688567263937:instance/i-0a528455c44ce001b

Figure 3: EC2 Instance Summary

- We use the Hyperledger Fabric framework and create a private blockchain network for secure financial transactions.

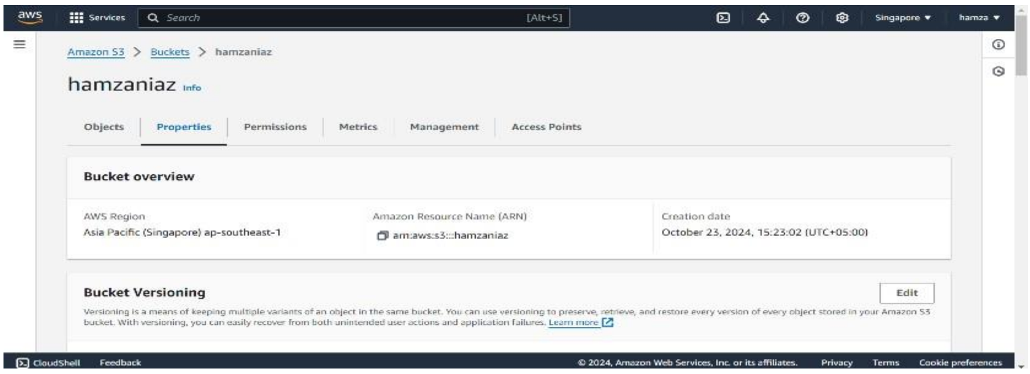

hamzaniaz

Details

Network ID: n-Q5XCUIA6SNBKDMDL5OYAEBXJVY
Description: -
ARN: arn:aws:managedblockchain:ap-southeast-1::networks/n-Q5XCUIA6SNBKDMDL5OYAEBXJVY
Voting policy: Greater than 50%
Status: Available
Created: Wed, Oct 23, 2024
Proposal duration: 24 hour(s)
VPC endpoint service name: com.amazonaws.ap-southeast-1.managedblockchain.n-q5xcuia6snbkdmdl5oyaebxjvy
Framework: Hyperledger Fabric 2.2
Ordering service endpoint: orderer.n-q5xcuia6snbkdmdl5oyaebxjvy.managedblockchain.ap-southeast-1.amazonaws.com:30001
Network edition: Starter
Active proposals: 0
Members: 1

Figure 4: Blockchain Network Creation

➢ Now we set up AWS S3 and create S3 bucket and configured it which is shown in the below figure.


Amazon S3 > Buckets > hamzaniaz

hamzaniaz

Bucket overview

AWS Region: Asia Pacific (Singapore) ap-southeast-1
Amazon Resource Name (ARN): arn:aws:s3:::hamzaniaz
Creation date: October 23, 2024, 15:23:02 (UTC+05:00)

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Figure 5: S3 Bucket for Backup

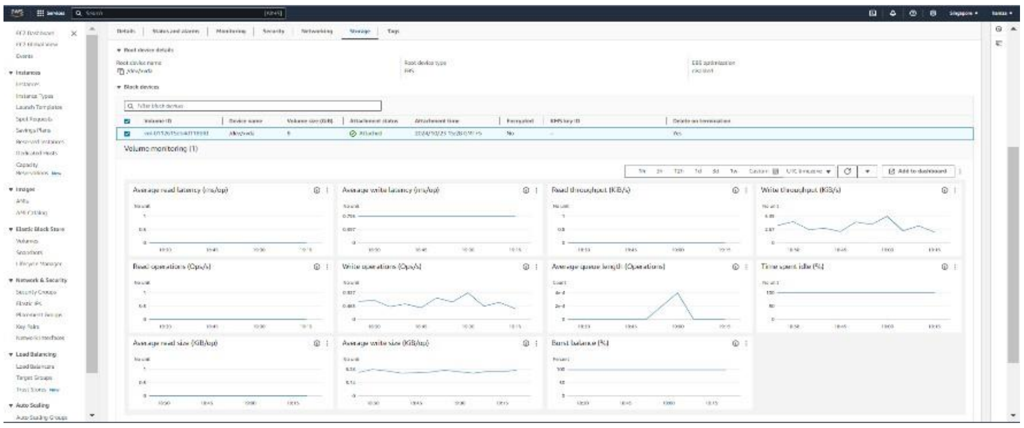
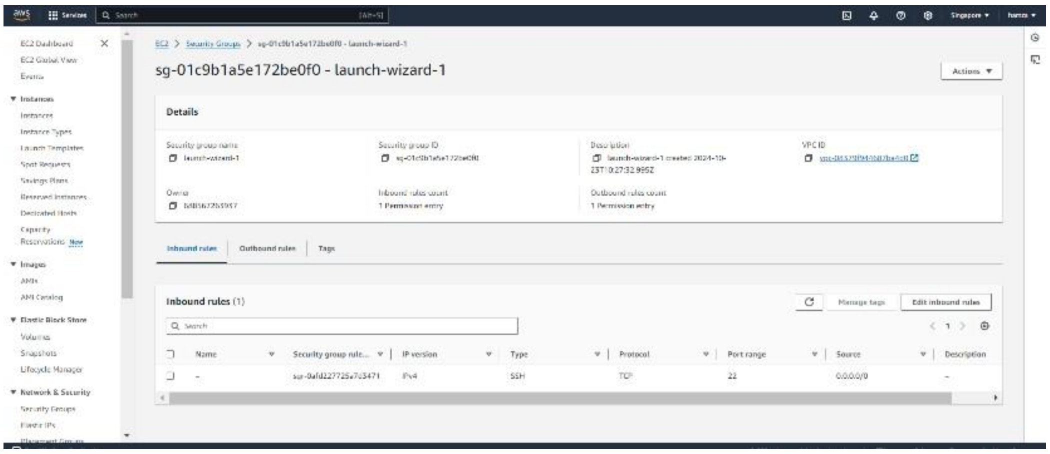


Figure 1: Storage and EBS Monitoring

➢ Now we set up the security group configuration
- We use the AWS IAM whch is a popular aspect of secure services that addresses permissions and access to blockchain nodes and users and it allows admin to give more than one roles and policies for engage with the blockchain network.


EC2 > Security Groups > sg-01c9b1a5e172be0f0 - launch-wizard-1

sg-01c9b1a5e172be0f0 - launch-wizard-1

Details

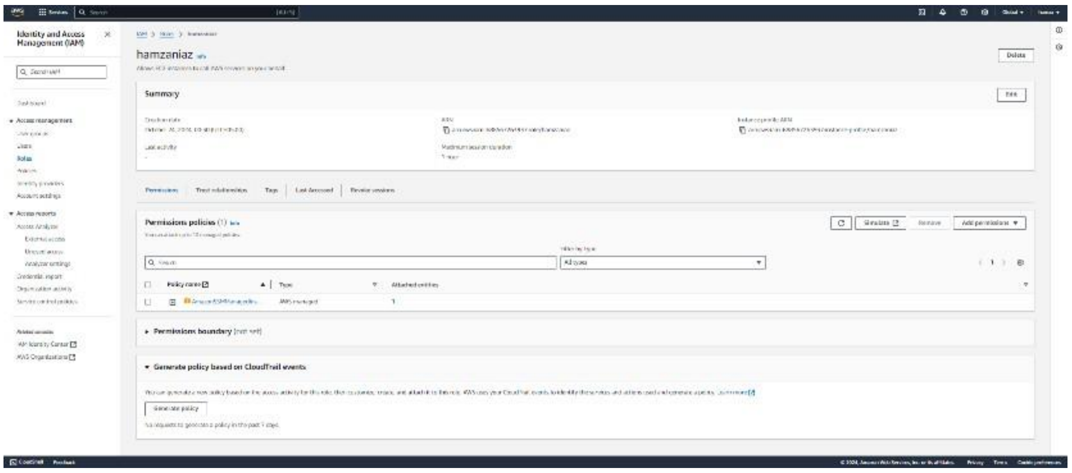
Security group name: launch-wizard-1
Security group ID: sg-01c9b1a5e172be0f0
Description: launch-wizard-1 created 2024-10-23T10:27:32.995Z
Owner: [illegible]
Inbound rules count: 1 Permission entry
Outbound rules count: 1 Permission entry

Inbound rules (1)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0afd227725a7c3471	IPv4	SSH	TCP	22	0.0.0.0/0	-

Figure 7: Secure Group Configuration

- We set the IAM role which make the secure access to AWS resources Like EC2 and S3 in the blockchain node.


hamzaniaz

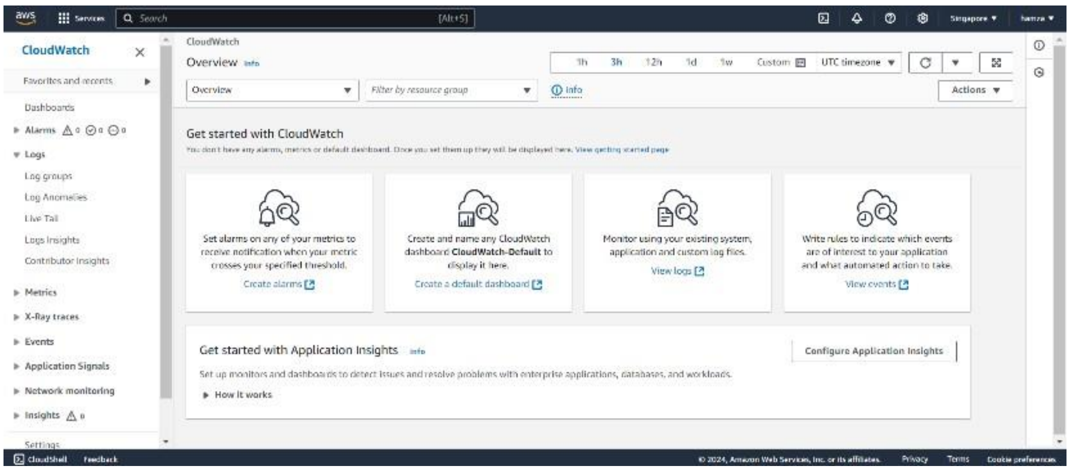
Summary

Permissions policies (1)

Figure 8: IAM Role for Permissions

➢ Now we set the AWS cloudwatch for monitoring

- It has a function of monitor the health and function of blockchain network and it works with collecting and monitoring logs and metrics that is coming from different AWS resources like the EC2 instances and S3 buckets that gives real time views and information on how the system works.



Get started with CloudWatch

You don't have any alarms, metrics or default dashboard. Once you set them up they will be displayed here. View getting started page

Set alarms on any of your metrics to receive notification when your metric crosses your specified threshold. Create alarms

Create and name any CloudWatch dashboard CloudWatch-Default to display it here. Create a default dashboard

Monitor using your existing system, application and custom log files. View logs

Write rules to indicate which events are of interest to your application and what automated action to take. View events

Get started with Application Insights

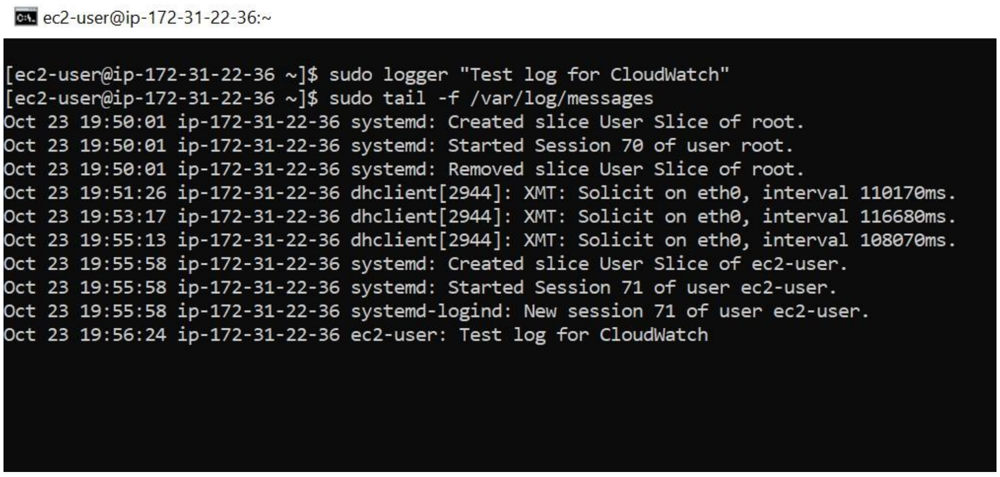
Set up monitors and dashboards to detect issues and resolve problems with enterprise applications, databases, and workloads.

How it works

Configure Application Insights

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

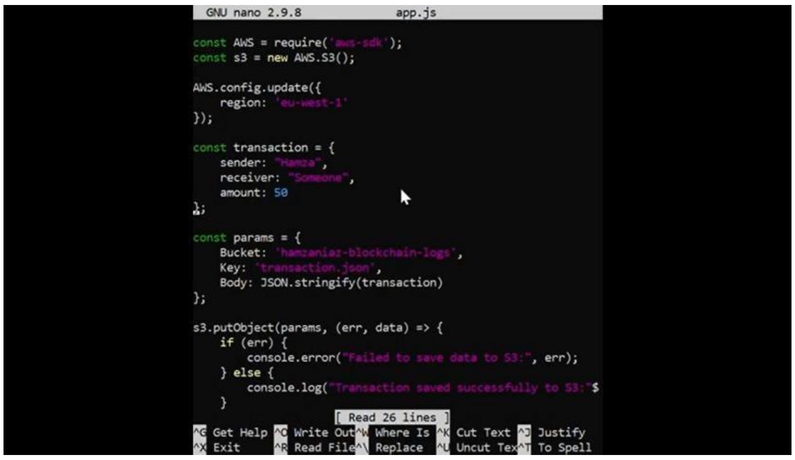
Figure 9: CloudWatch Monitor



```
ec2-user@ip-172-31-22-36:~

[ec2-user@ip-172-31-22-36 ~]$ sudo logger "Test log for CloudWatch"
[ec2-user@ip-172-31-22-36 ~]$ sudo tail -f /var/log/messages
Oct 23 19:50:01 ip-172-31-22-36 systemd: Created slice User Slice of root.
Oct 23 19:50:01 ip-172-31-22-36 systemd: Started Session 70 of user root.
Oct 23 19:50:01 ip-172-31-22-36 systemd: Removed slice User Slice of root.
Oct 23 19:51:26 ip-172-31-22-36 dhclient[2944]: XMT: Solicit on eth0, interval 110170ms.
Oct 23 19:53:17 ip-172-31-22-36 dhclient[2944]: XMT: Solicit on eth0, interval 116680ms.
Oct 23 19:55:13 ip-172-31-22-36 dhclient[2944]: XMT: Solicit on eth0, interval 108070ms.
Oct 23 19:55:58 ip-172-31-22-36 systemd: Created slice User Slice of ec2-user.
Oct 23 19:55:58 ip-172-31-22-36 systemd: Started Session 71 of user ec2-user.
Oct 23 19:55:58 ip-172-31-22-36 systemd-logind: New session 71 of user ec2-user.
Oct 23 19:56:24 ip-172-31-22-36 ec2-user: Test log for CloudWatch
```

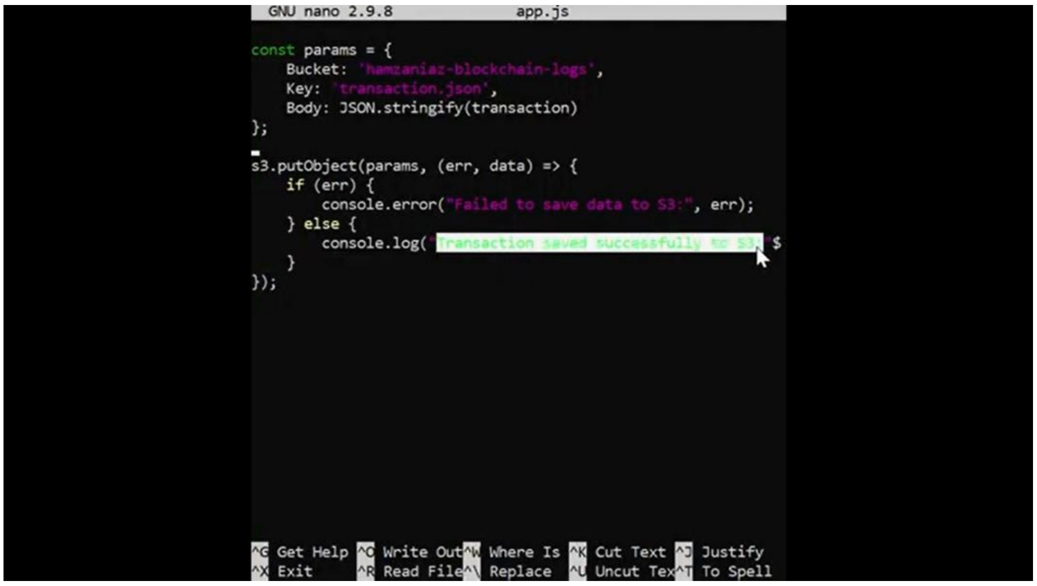
Figure 10: Log Stream for Blockchain Transactions

➢ Now we make the Transaction which is shown in the below figures



```
GNU nano 2.9.8                    app.js

const AWS = require('aws-sdk');
const s3 = new AWS.S3();

AWS.config.update({
    region: 'eu-west-1'
});

const transaction = {
    sender: "Hamza",
    receiver: "Someone",
    amount: 50
};

const params = {
    Bucket: 'hamzaniaz-blockchain-logs',
    Key: 'transaction.json',
    Body: JSON.stringify(transaction)
};

s3.putObject(params, (err, data) => {
    if (err) {
        console.error("Failed to save data to S3:", err);
    } else {
        console.log("Transaction saved successfully to S3:"$
    }
                     [ Read 26 lines ]
^G Get Help  ^O Write Out^W Where Is  ^K Cut Text ^J Justify
^X Exit      ^R Read File^\ Replace   ^U Uncut Tex^T To Spell
```

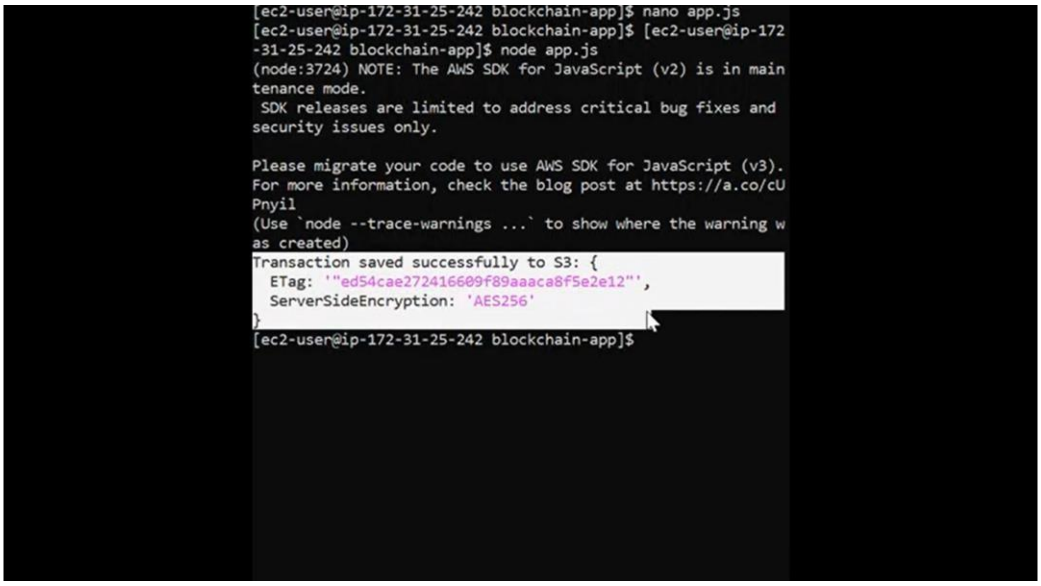
Figure 11: app.js



```
  GNU nano 2.9.8              app.js

const params = {
    Bucket: 'hamzaniaz-blockchain-logs',
    Key: 'transaction.json',
    Body: JSON.stringify(transaction)
};

s3.putObject(params, (err, data) => {
    if (err) {
        console.error("Failed to save data to S3:", err);
    } else {
        console.log("Transaction saved successfully to S3:"$
    }
});

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File ^\ Replace   ^U Uncut Tex ^T To Spell
```

Figure 12: app.js

- Now the transaction we make saved successfully to S3 which is shown in the below figure



```
[ec2-user@ip-172-31-25-242 blockchain-app]$ nano app.js
[ec2-user@ip-172-31-25-242 blockchain-app]$ [ec2-user@ip-172
-31-25-242 blockchain-app]$ node app.js
(node:3724) NOTE: The AWS SDK for JavaScript (v2) is in main
tenance mode.
 SDK releases are limited to address critical bug fixes and
security issues only.

Please migrate your code to use AWS SDK for JavaScript (v3).
For more information, check the blog post at https://a.co/cU
Pnyil
(Use `node --trace-warnings ...` to show where the warning w
as created)
Transaction saved successfully to S3: {
  ETag: '"ed54cae272416609f89aaaca8f5e2e12"',
  ServerSideEncryption: 'AES256'
}
[ec2-user@ip-172-31-25-242 blockchain-app]$
```

Figure 13: Transaction Saved to S3

References

Kushwaha, A., Singh, V., Srivastava, M. (2019). Blockchain-cloud integration: Use cases and future directions. Journal of Cloud Computing, 8(1), 1-14.

Rong, C., Nguyen, S. T., Jaitun, M. G. (2017). Beyond lightning: A survey on secure and scalable blockchain technologies. Future Generation Computer Systems, 97, 431454. Rittenhouse, J. W., Ransome, J. F. (2017). Cloud Computing: Implementation, Management, and Security. CRC press.

Bhojwani, V., Garg, S. (2020). Securing blockchain using cloud services. Journal of Security and Communication Networks, 2020, 1-10.

Gupta, V. (2018). Blockchain for financial services: Enhancing security, transparency, and scalability. Journal of Financial Innovation, 5(2), 56-70.

Tapscott, D., Tapscott, A. (2016). Blockchain Revolution: How the Technology Behind Bitcoin and Other Cryptocurrencies is Changing the World. Penguin.

Zohar, A. (2015). Regulation of bitcoin and blockchain technologies: Principles and insights. The Handbook of Digital Currency, 2015, 529-546.
Auer, R., Claessens, S. (2020). Regulating fintech: What is going on, and where are the challenges? BIS Quarterly Review, March 2020, 1-18.
Wu¨st, K., Gervais, A. (2018). Do you need a blockchain? In Proceedings of the 2018 Crypto Valley Conference on Blockchain Technology (pp. 45-54). IEEE. Kaur, S., Gera, R. (2020). Performance evaluation of blockchain-based financial transaction systems on cloud. Journal of Computer Science and Technology, 35(1), 97-109.