Faculty of Engineering and Physical Sciences

Department of Computer Science

MSc programmes in Computer Science

Academic Year 2024-2025

COMM065 – Cloud Security

CW1: Cloud Automation

Group 8

A project supervised by: [Mark Buckwell]

University of Surrey

School of Computer Science and Electronic Engineering

Guildford, Surrey GU2 7XH

United Kingdom.

S.No	Group Member name	Email
1.	Gulab Chopra	gc00996@surrey.ac.uk
2.	Mukesh Kumar	mk02731@surrey.ac.uk
3.	Rohith Krishn S M	rs02609@surrey.ac.uk

1. Introduction

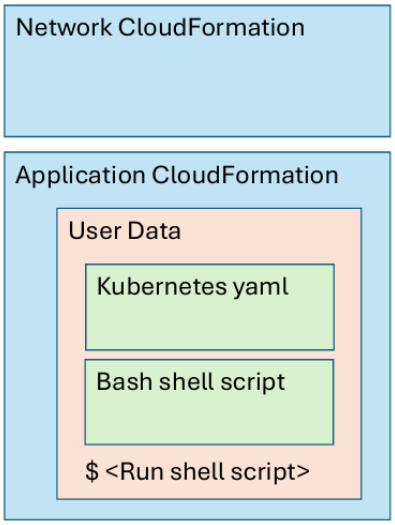
This coursework explores the deployment of a LAMP stack on Kubernetes within an Amazon EC2 environment, using CloudFormation automation to ensure efficient and repeatable cloud infrastructure provisioning. The assignment requires integrating bash scripting with user data, embedding it within CloudFormation templates, and carefully uploading network and application stack. A critical aspect of the work involves not only the technical implementation but also the validation and troubleshooting of the deployment to ensure the automated setup is reliable and optimized.

2. Prerequisites

Before beginning the deployment of the CloudFormation templates, several requirements must be addressed to ensure a smooth and efficient process. An active AWS account with sufficient permissions is required to deploy CloudFormation stacks, launch EC2 instances, and configure network resources. Additionally, access to the AWS Management Console or AWS CLI is necessary, along with the EC2 key pair [**`vockey.pem`**] for establishing secure SSH connections. Familiarity with core AWS services such as CloudFormation, EC2, VPC, and security groups is crucial, as these underpin the network and application layers. Furthermore, a working knowledge of Kubernetes, Docker, and the LAMP stack architecture is recommended to understand the deployed environment and troubleshoot effectively.

3. Files Provided

S.No	File name
1.	`lab-lampstack.yaml`
2.	`lab-network.yaml`
3.	`cloud-init-output.log`



4. Creating A Bash shell Script and Testing

A bash shell script is developed to install Docker, Minikube, and the necessary Kubernetes components, ensuring that all configurations required for the LAMP stack are correctly established. The script is incorporated into the User Data section when launching an EC2 instance to facilitate automatic execution during system startup. Furthermore, add this bash script to the **`lab-lampstack.yaml`** template in the UserData field to use it in Cloud Formation, ensuring uniform automation across all deployed instances.

5. Deploy the Network Stack

To create a network stack in AWS CloudFormation, define a template specifying key components like VPC, subnets, security groups, and route tables, then upload it to the CloudFormation Console.

Steps to be followed:

1. Log in to AWS Academy Learner Lab and start the lab session.
2. Navigate to the Cloud Formation service in the AWS Management Console.
3. Click on **`Create stack > With new resources (standard)`**
4. On the "Create stack" page:
    - Select **`Upload a template file`**
    - Click **`Choose file`** and upload the **`lab-network.yaml`** file
    - Click Next
5. Enter **`lab-network`** as the **`Stack name`** and click Next.
6. On the "Configure stack options" page, leave the default settings and click Next.
7. On the "Review" page:

- Click **`Create stack`**

8. Wait for the stack creation to complete. The status will change to "CREATE_COMPLETE" when finished.

6. Deploying the Application Layer

To create the Application stack, use the **`'lab-lampstack.yaml'`** to deploy an EC2 instance that bootstraps Docker, Minikube, and a Kubernetes-managed LAMP application via a user data script.

Steps for creating Application stack:

1. Once the network stack is created successfully, navigate back to CloudFormation.
2. Click on **`Create stack > With new resources (standard)`**
3. On the "Create stack" page:
    - Select **`Upload a template file`**
    - Click **`Choose file`** and upload the **`lab-lampstack.yaml`** file
    - Click Next
4. On the "Specify stack details" page:
    - Enter **`lab-lampstack`** as the **`stack name`**
    - For the "Network Stack Name" parameter, use **`lab-network`** (this should be the default)
    - For "KeyName", leave as **`vockey`** (this is pre-created in AWS Academy)
    - For "InstanceType", leave as **`t2.medium`**
    - Click Next
5. On the "Configure stack options" page, leave the default settings and click Next.
6. On the "Review" page, click **`Create stack.`**

Wait for the stack creation to complete. This will take approximately 5-7 minutes. The EC2 instance is being created, and the UserData script is installing and configuring Docker, Minikube, and deploying the LAMP stack. The status will change to "CREATE_COMPLETE" when finished.

7. Verify the Deployment

1. After successfully deploying the cloud Formation stack, a new EC2 instance will be created by the cloud Formation.
2. Logged into EC2 Amazon Console by clicking the **'connect'** option.
3. Once connected to the EC2 instance, verify Minikube and the LAMP stack are running:
   - Check Minikube status

```
$ minikube status
$ minikube start            #if minikube stopped
```

   - Check Kubernetes pods and services

```
$ minikube kubectl -- get pods
$ minikube kubectl -- get services

                          #LoadBalancer External IP <pending>
```

   - View Kubernetes deployment details

```
$ minikube kubectl -- get deployments
```

8. Accessing the Kubernetes dashboard

1. To access the Kubernetes dashboard, we need to expose the service. Run the tunnel script that was created during deployment:

```
$ ./tunnel-service.sh          #or
$ minikube tunnel
$ minikube kubectl -- get services
```

This script starts a tunnel that exposes the Load Balancer service to external traffic. Press Ctrl + Z to used next commands

2. To view the Kubernetes dashboard:

```
$ minikube dashboard --url          #port number 42363
```

   This will start the dashboard in the background.

3. In a new SSH session (open another terminal window and connect to the EC2 instance), find the external IP assigned to the load balancer:

```
$ ssh -i ./labsuser.pem -L 8081:localhost:42363 ec2-user@54.159.4.9
```

Take the **labsuser.pem** form **AWS details > Download PEM** and External IP form Amazon CLI interface, Press ctrl + z to end the process

4. Once the tunnel is running and the service has an external IP, you can access the Kubernetes dashboard in your web browser using:

   http://127.0.0.1:8081/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/

9. Accessing the DockLamp Application

To access the DockLamp Application, we need internal IP and a port number

```
$ minikube service lamp-app-service --url
```

5. Open a new terminal in local desktop and run the following command to connect to DockLamp app using ssh connection, find the external IP assigned to the LAMP service:

```
$ ssh -i labsuser.pem -L 8080:192.168.49.2:31048 ec2-user@54.159.4.9
```

6. Once the tunnel is running and the service has an external IP, you can access the LAMP stack in your web browser using:

   Then open http://localhost:8080 in your browser.

10. Conclusion

The implementation of automation through bash scripting and CloudFormation enhances deployment reliability and operational efficiency. By utilising the User Data field and cloud-init capabilities, the system ensures consistent configuration and smooth initialization. This approach supports scalable, secure cloud environments, aligning with industry standards for automated infrastructure management.