**Mastering Multi-Cloud**

**Labs: Containers, DevSecOps, and AI/ML in Multi-Cloud**

**Lab 01: Creating and Deploying a Google Kubernetes Engine**

**Case Study**

Kakao Mobility is South Korea's leading mobility platform company, providing a wide range of services including taxi-hailing, navigation, shared bikes, and delivery. To deliver seamless mobility experiences, real-time analytics, and personalized services to millions of users, Kakao Mobility uses a microservices-based architecture. As the platform scaled and expanded its services globally, Kakao Mobility required a reliable container orchestration system to support continuous deployment, automatic scaling, and high availability, especially during high-peak commercial traffic periods. The company adopted Google Kubernetes Engine (GKE) as its core container management platform.

By using Google Kubernetes Engine, Kakao Mobility modernized its traditional IT infrastructure, improved release speed, and increased system reliability. GKE’s built-in monitoring, auto-scaling, and deep integration with Google Cloud services enabled Kakao Mobility to handle significant traffic surges during major events and ensure reliable service delivery to its platform users while optimizing operational costs.

**Business Challenge**

As Kakao Mobility continued to grow, managing thousands of microservices across its diverse mobility services became increasingly complex. Traditional deployment approaches led to slower release cycles, configuration inconsistencies, and scalability limitations. Engineers faced challenges with manual deployments, massive traffic spikes during peak hours, and maintaining high availability for critical services like taxi-hailing and navigation.

Kakao Mobility needed a fully managed Kubernetes platform that could automate operations, support DevOps practices (such as DORA principles), and allow developers to deploy applications quickly and consistently. The company required a scalable solution that minimized infrastructure management and improved cloud security while ensuring high performance and cost-efficiency through FinOps practices.

**Solution**

Kakao Mobility implemented Google Kubernetes Engine to modernize its application delivery process and enhance its cloud security posture. GKE provided a fully managed Kubernetes environment, allowing development teams to focus on building innovative mobility features rather than maintaining underlying infrastructure.

In this lab, you follow a similar real-world deployment approach by using Google Cloud Shell to containerize an application, authenticate with Google Container Registry, push images to the registry, and create a Kubernetes cluster using command-line tools. The lab then guides you to deploy the application to the cluster, expose it externally using a LoadBalancer service, and verify accessibility through a public IP address.

Through this process, you gain hands-on experience with cloud-native application deployment, container orchestration, and production-level workload management. This mirrors how organizations like Kakao Mobility deploy scalable, resilient applications and achieve elite DevOps status using Google Kubernetes Engine.

**Lab Implementation**

In this hands-on lab, use the Cloud Shell command line from start to finish, first containerizing an app and then deploying and exposing it.

Shape

Description automatically generated

**Enable the Kubernetes Engine API**

1. From the main Google Cloud console navigation menu, choose **APIs & Services** > **Library**.

Graphical user interface, application, Teams

Description automatically generated

2. Search for **Kubernetes** and enable the **Kubernetes Engine API**.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

**Follow the given steps to clone the repo to access the files**

3. Activate Cloud Shell by clicking the icon in the top row.

Graphical user interface, text, application, email

Description automatically generated

4. Clone the GitHub repository by using the following command: **git clone https://github.com/linuxacademy/content-gc-essentials**.

**Text

Description automatically generated**

5. Run the following command: **cd content-gc-essentials/gke-lab-01** to change the directory.

Text

Description automatically generated

**Follow the given steps to create the Docker image.**

6. Run the following command: **docker build -t la-container-image** to build the Docker image.

Text

Description automatically generated

7. Configure the Docker command line to authenticate to Container Registry by using the following command: **gcloud auth configure-docker**.

Text

Description automatically generated

8. Tag the registry image by using the following command: **docker tag la-container-image gcr.io/<PROJECT\_ID>/la-container-image:v1** (you can replace your project ID, indicated by highlighting the code in yellow in the Cloud Shell, which will automatically add it to your clipboard):

Push the image forward (substituting your project ID where indicated): **docker push gcr.io/<PROJECT\_ID>/la-container-image:v1**.

Text

Description automatically generated

Text

Description automatically generated

9. Create the Kubernetes Engine Cluster.

Configure the zone by using the following command: **gcloud config set compute/zone us-central1-a**.

Text

Description automatically generated

10. Run the following command: **gcloud container clusters create la-gke-1 --num-nodes=4** to create the clusters.

Text

Description automatically generated

11. Get authentication credentials by using the following command: **gcloud container clusters get-credentials la-gke-1**.

12. Deploy the app (substituting your project ID where indicated) by running the following command:

**kubectl create deployment la-greetings --image=gcr.io/<PROJECT\_ID>/la-container-image:v1**.

**Expose the Deployed Workload**

Text

Description automatically generated

13. To expose the deployment, run the following command: **kubectl expose deployment la-greetings --type=LoadBalancer --name=la-greetings-service --port=80 --target-port=8**.

Graphical user interface, application

Description automatically generated

14. Execute the following command: **kubectl get services la-greetings-service** to check its status.

Graphical user interface, text

Description automatically generated

15. When an external IP address has been generated, copy and paste it into a new browser tab. It should result in our application.

**Lab 02: Generative AI with Amazon Bedrock**

**Case Study**

Expedia Group is one of the world’s largest online travel platforms, helping millions of customers plan trips, book flights, reserve hotels, and manage travel itineraries. To improve customer experience and operational efficiency, Expedia leverages artificial intelligence to provide personalized recommendations, automated customer support, and real-time travel insights.

As user expectations increased, Expedia needed more advanced AI capabilities to generate human-like responses, summarize travel data, and assist customers through conversational interfaces. The company adopted AWS Generative AI services, including Amazon Bedrock, to access foundation models from leading AI providers without managing complex infrastructure.

By using Amazon Bedrock, Expedia integrated large language models into its applications to enhance chatbots, improve search experiences, and generate personalized travel suggestions. This enabled the company to deliver faster support, better recommendations, and more engaging customer interactions at scale.

**Business Challenge**

Expedia faced challenges handling millions of daily customer interactions across its platforms. Traditional rule-based chatbots and search systems struggled to understand complex travel queries, provide contextual responses, and deliver personalized suggestions.

The company needed a secure and scalable way to integrate generative AI into its applications without building or managing its own machine learning models. Data privacy, model governance, and seamless integration with existing AWS services were also critical concerns. Expedia required a managed solution that allowed rapid experimentation, reliable performance, and enterprise-level security.

**Solution**

Expedia implemented Amazon Bedrock to access foundation models through simple APIs while maintaining full control over data security and compliance. Amazon Bedrock allowed the company to experiment with multiple generative AI models, fine-tune responses, and integrate AI capabilities directly into its customer-facing applications.

In this lab, learners follow a similar approach by using Amazon Bedrock to interact with foundation models for text generation, summarization, and conversational AI. Students explore how to securely invoke models, customize prompts, and integrate AI-generated responses into real-world use cases.

This hands-on lab demonstrates how organizations like Expedia use Amazon Bedrock by guiding learners through signing in to AWS, opening the Bedrock console, selecting a supported region, and accessing the Chat/Text Playground. Learners choose a foundation model, enter prompts, view AI-generated responses, compare outputs across models, and explore text and image generation features. These steps show how generative AI can be used to build intelligent applications, automate customer interactions, and deliver personalized user experiences on AWS.

|  |
| --- |
| **Lab Diagram**    **Step 1: Sign in to AWS**   1. Open your web browser and go to **https://aws.amazon.com**. 2. Log in using your AWS account credentials (email and password). 3. Once logged in, you will see the AWS Management Console home page.     **Step 2: Open Amazon Bedrock**   1. In the top search bar, type **Bedrock**.      1. Click on **Amazon Bedrock** from the search results.      1. This will open the **Amazon Bedrock** console.     **Step 3: Select the AWS Region**   1. Amazon Bedrock is available in specific regions. From the top-right corner, click the **Region** selector (a dropdown menu). Choose one of the supported regions, for example: **us-west-2 (Oregon)**.     **Step 4: Open the Playground**   1. From the left-hand side menu, select **Chat / Text Playground**.      1. You will see an editor where you can type prompts and choose a model.     **Step 5: Choose a Model**   1. Under Mode, choose **Single prompt**.      1. Under **Model Selection**, click on the dropdown and choose a model such as **Anthropic Claude 3 Haiku**. This is fast, efficient, and perfect for beginners.      1. Click **Apply** to confirm your selection.     **Step 6: Enter a Prompt**   1. In the large text box labeled **Enter your prompt here**, type a simple question like:   **What is the capital city of Australia?**     1. Click **Run** to send the prompt to the model.     **Step 7: View the Output**   1. Wait a few seconds for the model to process your request, and you will see the AI’s response appear below, such as: **The capital city of Australia is Canberra.**     **Step 8: Try More Prompts**  Try a few more examples to explore what the AI can do:   * **Write a two-line poem about the sun.**      * **Summarize why exercise is important.**      * **Explain cloud computing in simple words.**     **Step 9: Compare Models**   1. Change the **Mode** to **Chat**.      1. Enable **Compare Mode**.      1. Select two models (for example, **Claude 3 Haiku** and **Claude 3 Sonnet**).      1. Type the same question and click **Run** to see how each model’s response differs.    * One might give a shorter answer    * Another might give a more detailed explanation     **Step 10: Explore Further**   1. Try the **Image Playground** to generate images from text (for example, **A mountain landscape at sunset**). |