BS Final Year Project

K8s Learning Portal

Program: BSCS
Batch: 2019-2023

Project Supervisor: Mr. Saqib Rasool

Submitted By

Khaleel Ur Rehman	*19014119-066*
Sami Ullah	*19014119-102*
Mohtasham Mujahid	*19014119-104*

FACULTY OF COMPUTING & INFORMATION TECHNOLOGY
UNIVERSITY OF GUJRAT

STATEMENT OF SUBMISSION

This is to certify that **Khaleel Ur Rehman** Roll No. **19014119-066**, **Sami Ullah** Roll No. **19014119-102** and **Mohtasham Mujahid** Roll No. **19014119-104** has successfully completed the final year project named as **K8s Learning Portal** at the Department of Computer Science, University of Gujrat, to fulfill the requirement of the degree of **BSCS** in Computer Science.

____

Project Supervisor

____

Project Coordination Office
Faculty of C&IT -UOG

____

Head of the Department

Acknowledgement

We truly acknowledge the cooperation and help make by Sir Saqib Rasool, PMO, Department of Computer Science Evening, University of Gujrat. He has been a constant source of guidance throughout the course of this project. We are also thankful to our friends and families whose silent support led us to complete our project.

1.Khaleel Ur Rehman
2.Sami Ullah
3.Mohtasham Mujahid

Date:

Abstract

Kubernetes has emerged as an essential skill for modern software professionals due to its ability to manage the complexities of containerized applications. As organizations shift to microservices and container-based architectures, Kubernetes offers automated deployment, scaling, and management. This abstract highlights the necessity of learning Kubernetes to ensure application availability, streamline collaboration between teams, enable efficient scaling, and abstract infrastructure complexities. Mastering Kubernetes empowers professionals to navigate the challenges of contemporary software development and operations, making it a crucial asset in today's rapidly evolving technology landscape.

TABLE OF CONTENTS

Chapter 1: Project Feasibility Report

1.1. Introduction

The Kubernetes Learning and Evaluation Portal provides a unique learning experience that combines theory and practice, making it an ideal resource for anyone looking to enhance their knowledge and skills in Kubernetes. Whether you're a student, a developer, or an IT professional, this portal is an excellent platform to learn Kubernetes and take your career to the next level.

1.2. Project/Product Feasibility Report

The Kubernetes Learning and Evaluation Portal is a web-based project designed to provide an interactive and comprehensive learning experience in Kubernetes. This portal aims to address the challenges faced by learners in gaining hands-on experience in Kubernetes and to evaluate and assess their skills in the platform. This project feasibility report will assess the viability of the Kubernetes Learning and Evaluation Portal project, including technical, operational, economic, schedule, specification, information, motivational, legal, and ethical feasibility

1.2.1. Technical Feasibility

The Kubernetes Learning and Evaluation Portal is technically feasible as it is built using well-established and reliable technologies, including React, Node.js, and MongoDB. These technologies provide a robust and scalable architecture that can handle large amounts of traffic and data. The project also utilizes Kubernetes clusters to simulate a real-world Kubernetes environment, enabling users to practice their skills and experiment with different configurations. The project team has the required technical skills and expertise to develop and maintain the platform.

1.2.2. Operational Feasibility

The Kubernetes Learning and Evaluation Portal is operationally feasible as it is designed to be user-friendly, interactive, and accessible from anywhere with an internet connection. The platform provides a comprehensive learning experience, including quizzes, coding exercises, and practical projects, which are evaluated and assessed in real-time. The platform is also scalable, allowing for the addition of new content and features as the platform evolves. The project team has planned for the operational aspects of the project, including server maintenance, software updates, and content creation.

1.2.3. Economic Feasibility

The Kubernetes Learning and Evaluation Portal is economically feasible as it has the potential to generate revenue through subscription-based models, advertisements, and partnerships with educational institutions and organizations. The project's operational costs include server maintenance, software updates, and content creation, which can be offset by the revenue generated from the platform. The initial investment in hardware and software can also be recovered over time through revenue generation. The project team has conducted a cost-benefit analysis and identified revenue streams to ensure the project's economic viability.

1.2.4. Schedule Feasibility

The Kubernetes Learning and Evaluation Portal is schedule feasible as the project team has developed a detailed project plan with clear milestones and timelines. The team has identified potential risks and challenges and has planned for contingencies. The project timeline is realistic, considering the scope and complexity of the project and the available resources.

1.2.5. Specification Feasibility

The Kubernetes Learning and Evaluation Portal is specification feasible as the project team has defined clear project requirements and specifications. The team has conducted user research and incorporated feedback from potential users to ensure that the platform meets their needs. The team has also established quality assurance processes to ensure that the platform meets the required specifications and standards.

1.2.6. Information Feasibility

The Kubernetes Learning and Evaluation Portal is information feasible as the project team has planned for the collection, storage, and management of user data. The team has incorporated privacy and security measures to protect user data and comply with relevant regulations. The team has also established processes for data backup and disaster recovery.

1.2.7. Motivational Feasibility

The Kubernetes Learning and Evaluation Portal is motivational feasible as the project team has identified the target audience and their motivation to use the platform. The team has designed the platform to be engaging, interactive, and rewarding, providing users with a sense of achievement and progress. The team has also incorporated social features, allowing users to interact with each other and share their experiences.

1.2.8. Legal & Ethical Feasibility

The Kubernetes Learning and Evaluation Portal is legal and ethical feasible as the project team has considered relevant legal and ethical issues. The team has incorporated privacy and security measures to protect user data and comply with relevant regulations. The team has also established ethical guidelines for content creation and user interaction, ensuring that the platform promotes a positive learning experience.

1.3. Project/Product Scope

The Kubernetes Learning and Evaluation Portal is a web-based platform that will provide users with an interactive and comprehensive learning experience for Kubernetes, an open-source container orchestration platform. The portal will provide users with a set of tools and resources to help them learn and evaluate their understanding of Kubernetes concepts and best practices.

The portal will have the following features:

1. Kubernetes Learning Resources: The portal will provide a comprehensive set of learning resources such as tutorials, videos, and interactive labs to help users learn Kubernetes concepts.
2. Assessment and Evaluation: The portal will provide a series of quizzes, assignments, and projects to help users evaluate their understanding of Kubernetes concepts and best practices.
3. User Dashboard: The portal will provide users with a personalized dashboard to track their progress, view their assessment results, and access learning resources.
4. Discussion Forum: The portal will include a discussion forum to encourage community engagement and collaboration. Users can ask questions, share their knowledge and experience, and interact with other users.
5. Integration with Kubernetes API: The portal will integrate with the Kubernetes API to provide users with a hands-on experience of deploying and managing Kubernetes clusters.
6. Customizable User Profile: The portal will allow users to create and customize their profiles, track their progress, and view their achievements.
7. Mobile Compatibility: The portal will be designed to be mobile-friendly, allowing users to access the platform from their smartphones and tablets.

Deliverables: The following are the deliverables for the project:

1. Frontend and Backend Development: The development of the web-based platform that includes user interface design, server-side development, and integration with Kubernetes API.
2. Learning and Assessment Content Development: The creation of tutorials, videos, interactive labs, quizzes, assignments, and projects to help users learn and evaluate their understanding of Kubernetes concepts and best practices.
3. User Documentation: The creation of user guides, help documents, and FAQ to help users navigate and use the portal.
4. Testing and Quality Assurance: The testing of the platform to ensure that it is bug-free, secure, and scalable.
5. Deployment and Maintenance: The deployment of the portal on a web server and the maintenance of the platform to ensure that it is up-to-date and functioning correctly.

1.4. Project/Product Costing

A metric is some measurement we can make of a product or process in the overall development process. Metrics are split into two broad categories:

- Knowledge oriented metrics: these are oriented to tracking the process to evaluate, predict or monitor some part of the process.
- Achievement oriented metrics: these are often oriented to measuring some product aspect, often related to some overall measure of quality of the product.

Most of the work in the cost estimation field has focused on algorithmic cost modeling. In this process costs are analyzed using mathematical formulas linking costs or inputs with metrics to produce an estimated output. The formulae used in a formal model arise from the analysis of historical data. The accuracy of the model can be improved by calibrating the model to your specific development environment, which basically involves adjusting the weightings of the metrics.

1.4.1. Project Cost Estimation By Function Point Analysis

Function Point Analysis (FPA) is a technique used for software cost estimation based on the functionality provided by the software. To estimate the project cost of web-based project named "Kubernetes Learning and Evaluation Portal" using FPA, we will need to follow the steps below:

Step 1: Identify the functional requirements of the system The first step in FPA is to identify the functional requirements of the system. In your case, the functional requirements of your system could include:

- User registration and authentication
- Learning materials (text, video, audio)
- Quizzes and assessments
- Personalized learning paths
- Feedback and progress tracking
- Integration with Kubernetes API

Step 2: Categorize the functional requirements Once you have identified the functional requirements, you need to categorize them into different types of functions. The categories for FPA are:

- External Inputs (EI): User input that results in data processing within the system
- External Outputs (EO): Data sent to the user as a result of processing within the system
- External Inquiries (EQ): Data requested by the user that requires data processing within the system
- Internal Logical Files (ILF): Data maintained within the system
- External Interface Files (EIF): Data stored outside the system that is referenced by the system

For example, user registration and authentication would be an External Input, learning materials would be an Internal Logical File, and quizzes and assessments would be an External Inquiry.

Step 3: Assign complexity levels Each function identified in Step 1 needs to be assigned a complexity level based on the number of data elements and the number of processing steps required. The complexity levels are:

- Low (3): Up to 4 data elements and up to 5 processing steps
- Average (4): 5 to 15 data elements and 6 to 19 processing steps
- High (6): More than 15 data elements and more than 19 processing steps

For example, user registration and authentication might be assigned a complexity level of Low, while personalized learning paths might be assigned a complexity level of High.

Step 4: Calculate the Unadjusted Function Points (UFP) The Unadjusted Function Points (UFP) for a project is the sum of the complexity levels of each function. For example, if you had 3 Low complexity functions, 2 Average complexity functions, and 1 High complexity function, the UFP would be calculated as follows:

UFP = (3 x 3) + (2 x 4) + (1 x 6) = 19

Step 5: Calculate the Adjusted Function Points (AFP) The Adjusted Function Points (AFP) takes into account the complexity of the project's environment. The complexity factors that need to be considered are:

- Data communications
- Distributed functions
- Performance
- Heavily used configuration
- Transaction rate
- Online data entry
- End-user efficiency
- Complex processing
- Reusability
- Installation ease
- Operational ease
- Multiple sites

Each complexity factor is assigned a weight between 0 and 5, with 0 meaning no influence and 5 meaning strong influence. You need to evaluate each complexity factor for your project and assign the appropriate weight. Once you have assigned weights to all complexity factors, you can calculate the Complexity Adjustment Factor (CAF) using the following formula:

CAF = 0.65 + (0.01 x Sum of Complexity Weights)

For example, if you assigned the following weights for each complexity factor:

- Data communications: 3
- Distributed functions: 2
- Performance: 4
- Heavily used configuration: 2
- Transaction rate: 3
- Online data entry: 3
- End

1.4.2. Project Cost Estimation by using COCOMO’81 (Constructive Cost Model)

COCOMO'81 is a widely used algorithm for software project cost estimation. It takes into account the size of the project, the complexity of the project, and the experience of the development team.

For web-based project named "Kubernetes Learning and Evaluation Portal," we would need to estimate the project size in terms of lines of code or function points. Assuming a size of around 20,000 lines of code, we can estimate the cost using the COCOMO'81 model as follows:

1. Calculate the basic effort required for the project:

   Basic Effort = a * (KLOC)^b, where KLOC is the size of the project in thousands of lines of code. For your project, assuming a = 2.94 and b = 1.05 (values for the Organic mode), we get:

Basic Effort = 2.94 * (20)^1.05 = 72.24 Person-Months

2. Calculate the development time:

   Development Time = c * (Basic Effort)^d, where c and d are mode-specific constants. For the Organic mode, assuming c = 3.67 and d = 0.28, we get:

   Development Time = 3.67 * (72.24)^0.28 = 14.87 Months

3. Calculate the number of developers required:

   Number of Developers = Basic Effort / Development Time

   Number of Developers = 72.24 / 14.87 = 4.85, which we can round up to 5.

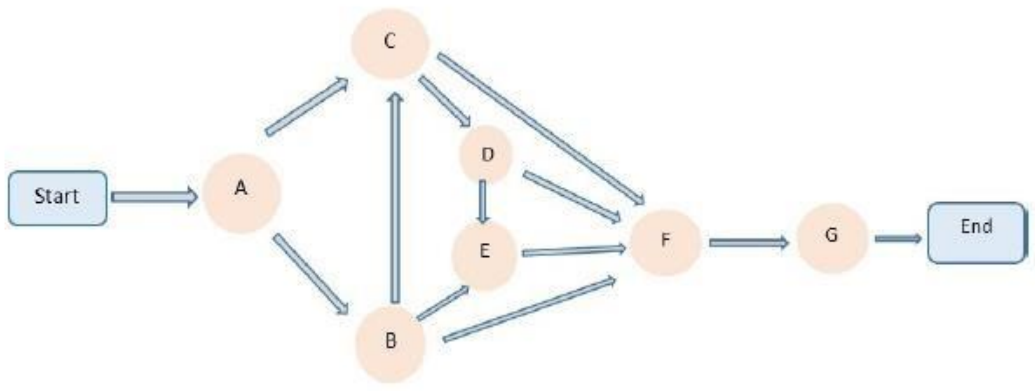
1.4.3. Activity Based Costing

Sr.	Activities	Resources	Cost Rate	Duration
1	Web design, layout and Structure	CodeLabs	5k	4
2	Front end	Html css	5k	5
3	Development of web components	golang,md	Free	8
4	Back end	CodeLabs	5k	7
5	Database connectivity	-	Free	5
6	Integration	claat	Free	5
7	Testing	Visual Studio Code	5k	4
8	Documentation	MS Office	Free	2

1.5. Task Dependency Table

Task	Task name	Dependency	Duration (Weeks)
A	Requirement Gathering	None	3
B	Web design, layout and Structure	A	4
C	Development of web components	A, B	8
D	Back end	C	6
E	Database connectivity	B, D	5
F	Integration	B, C, D, E	6
G	Testing	F	3

1.6. CPM - Critical Path Method



Activity Table

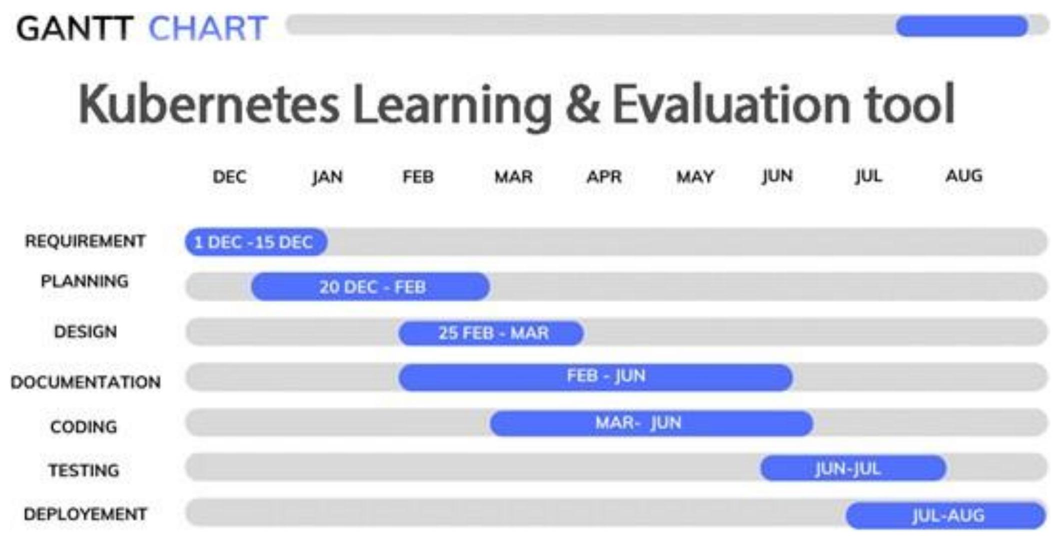
Activity	Duration	ES	EF	LS	LF	Slack Time	Critical path

						(ES-LS)	
A	3	0	3	0	3	0	Yes
B	4	3	7	3	7	0	Yes
C	8	7	15	7	15	0	Yes
D	6	15	21	15	21	0	Yes
E	5	21	26	21	26	0	Yes
F	6	26	32	26	32	0	Yes
G	3	32	35	32	35	0	Yes

Possible paths are:
A-C-F-G=3+8+6+3=20
A-C-DF-G=3+8+6+6+3=26
A-B-C-F-G=3+4+8+6+3=24
A-B-C-D-F-G=3+4+8+6+6+3=30
A-B-C-D-E-F-G=3+4+8+6+5+6=3=35
A-B-E-F-G=3+4+5+6+3=21
A-B-F-G=3+4+6+3=16
A-C-D-E-F-5=3+8+6+5+6+3=31
The critical path is the earliest time by which project ends and it is the longest path amongst all, so critical path of our project is:
A, B, C, D, E, F, G

Possible paths are:
A-C-F-G=3+8+6+3=20
A-C-D-F-G=3+8+6+6+3=26
A-B-C-F-G=3+4+8+6+3=24
A-B-C-D-F-G=3+4+8+6+6+3=30
A-B-C-D-E-F-G=3+4+8+6+5+6=3=35
A-B-E-F-G=3+4+5+6+3=21
A-B-F-G=3+4+6+3=16
A-C-D-E-F-5=3+8+6+5+6+3=31 **The critical path is the earliest time by which project ends and it is the longest path amongst all, so critical path of our project is: A, B, C, D, E, F, G**

1.7. Gantt chart

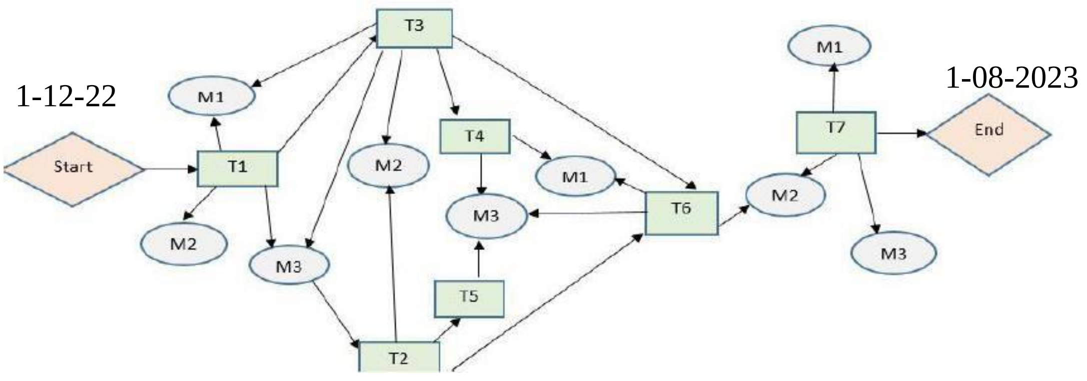


1.8. Introduction to Team member and their skill set

Members name	Skill set	Task
Khaleel Ur Rehman	Front end, back end and database	Development of module, design, structure implementation
Sami Ullah	Front end and testing	Development of module, design, structure implementation, testing
Mohatsham Mujahid	Documentation and testing	Project testing, Documentation

1.9. Task and Member Assignment Table

Task	Duration (days)	Dependencies	Members
T1	21		**M1, M2, M3**
T2	28	**T1**	**M2**
T3	56	T1, T2	M1, M2, M3
T4	42	T3	M1
T5	35	T2, T4	M3
T6	42	T2, T3, T4, T5	M1, M2, M3
T7	21	T6	M1, M2, M3



Task durations and dependencies

15-12-22	*18-1-23*	*28-3-23*	*15-4-23*	*20-6-23*	*20-7-23*
M2,M3					
	M2,T1,T2				
		M1,T1,T2,T3			
			M1,T1,T2,T3, T4		
				M2,M1,T1,T2,T3,T4	
					M1,M2,M3,T1,T2,T3,T4,T5, T6

Allocation of People to Activities:

Task	Member
T1	**Khaleel, Sami, Mohatsham**
T2	**Sami Ullah**
T3	**Khaleel**
T4	**Mohatsham**
T5	**Sami, Mohatsham**
T6	**Khaleel**
T7	**Khaleel, Sami, Mohatsham**

1.10. Tools and Technology with reasoning

1. **Golang, CodeLabs, claat**
   GoLand is an IDE that provides a comprehensive set of tools for Go programming, and it can be used for creating and testing Go code examples that are part of your codelab. On the other hand, Codelabs (claat) is a tool for creating interactive tutorials with a focus on structured content and interactive coding experiences for learners. Using both GoLand and Codelabs together can help you

create effective and engaging Google Codelabs that involve programming with the Go language.

2. **Sass:**
   Sass is an extension to CSS. It's a CSS pre-processor. Saas is completely compatible with all versions of CSS. It's is free to download and use.
3. **Bootstrap**
   Bootstrap is a free, open-source front-end development framework for the creation of websites and web apps.
   Reason: It is very helpful in terms of front-end CSS styling and time saver.
4. **Java Script**
   Java script is a cross-platform, object-oriented scripting language used to make web pages interactive as well as effective for writing algorithms.
   Java script contains a standard library of objects, such as array, data and math and a core set of language elements such as operators, control structures and statements
   Reason: It is top leading language to provide secure sessions of the websites.
5. **Visual studio Code**
   Visual Studio is an Integrated Development Environment(IDE) developed by Microsoft to develop GUI(Graphical User Interface), console, Web applications, web apps, mobile apps, cloud, and web services, etc. With the help of this IDE, you can create managed code as well as native code. It uses the various platforms of Microsoft software development software like Windows store, Microsoft Silverlight, and Windows API, etc.
   Reason: It is top leading IDE in the market.

6. **Microsoft word**
   Microsoft Word or MS Word is a popular word-processing program used mainly for creating documents, such as brochures, letters, learning activities, quizzes, tests, and students' homework assignments.
   Reason: It is top leading and very easy to use document writer in the market.

7. **SQL**
   Structured Query Language or SQL is a standard Database language which is used to create, maintain and retrieve the data from relational databases like MySQL, Oracle, SQL Server, PostGres, etc.
   Reason: It is top leading personal database system.

1.11. Vision Document

The Kubernetes Learning and Evaluation Portal is a web-based platform that provides a comprehensive set of resources to learn and evaluate Kubernetes. The portal aims to simplify the learning curve of Kubernetes and provide a user-friendly interface for users to experiment with Kubernetes without having to set up complex infrastructure.

Purpose

The purpose of the project is to create a platform that serves as a one-stop-shop for Kubernetes learning and evaluation. The portal will be designed to cater to different audiences, including beginners who are new to Kubernetes, developers who want to improve their Kubernetes skills, and organizations who want to evaluate Kubernetes for their projects.

Goals

The goals of the project are as follows:

- To provide an interactive and engaging learning experience for users to learn Kubernetes
- To provide a user-friendly interface for users to experiment with Kubernetes without having to set up complex infrastructure
- To provide a comprehensive set of resources that cover all aspects of Kubernetes, from basic concepts to advanced topics
- To provide a platform that is accessible to all users, regardless of their technical background or expertise
- To provide a platform that is continuously updated with the latest Kubernetes developments and trends

Scope

The scope of the project includes the following features:

- A dashboard that provides an overview of Kubernetes and its components
- A set of interactive tutorials that cover Kubernetes concepts and best practices
- A set of practical exercises that allow users to experiment with Kubernetes in a sandbox environment
- A set of quizzes and assessments to test users' knowledge of Kubernetes
- A community forum where users can interact and share their experiences with Kubernetes
- A resource library that includes documentation, videos, and other resources related to Kubernetes

Stakeholders

The stakeholders of the project include the following:

- Developers who want to learn and improve their Kubernetes skills
- Beginners who are new to Kubernetes and want to learn more about it
- Organizations who want to evaluate Kubernetes for their projects
- Trainers and educators who want to use the platform to teach Kubernetes

- Contributors who want to improve the platform by adding new content or features

1.12. Risk List

The risk list for the Kubernetes Learning and Evaluation Portal project includes technical, organizational, user, and environmental risks. The project team should prioritize these risks, develop mitigation strategies, and monitor their impact throughout the project's lifecycle. By effectively managing these risks, the project can achieve its goals and deliver a high-quality platform that meets the needs of its stakeholders.

1.13. Product Features/ Product Decomposition

User Interface

- Modern, user-friendly interface that is easy to navigate and visually appealing
- Personalized user accounts to track progress and customize learning experiences
- Progress tracking system to keep track of completed courses and assessments

Learning Content

- High-quality, comprehensive Kubernetes learning materials such as articles, tutorials, and video courses
- Interactive learning experiences such as coding challenges and quizzes
- A variety of Kubernetes-related topics, including containerization, orchestration, deployment, and scaling

Evaluation and Assessment

- Hands-on exercises and projects to evaluate the user's practical understanding of Kubernetes concepts
- Automated assessments to provide real-time feedback on the user's progress
- Ability to earn certificates and badges to acknowledge completed learning and demonstrate skills to potential employers

Community and Collaboration

- Discussion forums to foster community engagement and facilitate collaborative learning
- Peer review system to enable users to evaluate and learn from each other's work
- Access to experts and mentors to provide guidance and support

Integration and Customization

- Integration with popular tools and platforms used in the Kubernetes ecosystem
- Customization options to adapt the learning experience to the user's needs and preferences
- Open-source codebase for easy customization and extension

Chapter 2: Software Requirement Specification (For Object Oriented Approach)

2.1 Introduction:

2.1.1 Systems Specifications

Introduction

The Kubernetes Learning and Evaluation Portal is a web-based platform that provides users with high-quality learning materials, hands-on exercises, and assessments to help them gain practical understanding of Kubernetes concepts. This system specification document outlines the technical specifications for the Kubernetes Learning and Evaluation Portal project.

Existing System

There is currently no system that provides a comprehensive platform for learning Kubernetes concepts. Existing systems such as online courses and documentation provide limited hands-on experience and do not provide a means for assessing practical knowledge.

Organizational Chart

The Kubernetes Learning and Evaluation Portal will be developed by a team of four developers, under the supervision of a project manager. The project manager will be responsible for managing the project, including project planning, scheduling, and budgeting. The development team will consist of one frontend developer, one backend developer, one database developer, and one QA engineer.

Scope of the System

The Kubernetes Learning and Evaluation Portal will provide users with the following features:

1. User Management
    - User authentication and authorization
    - Account creation and management
    - Password reset
2. Learning Content Management
    - Kubernetes learning materials including articles, tutorials, and video courses
    - Search and filtering of learning materials by category and topic
    - Bookmarking of learning materials
3. Interactive Learning Experiences Management
    - Coding challenges and quizzes
    - Immediate feedback on progress
    - Progress tracking

4. Hands-on Exercises and Projects Management
    - Kubernetes cluster for completing hands-on exercises and projects
    - Submission of completed exercises and projects for evaluation
5. Automated Assessments Management
    - Automated assessments to evaluate understanding of Kubernetes concepts
    - Immediate feedback on assessments
    - Progress tracking
6. Community and Collaboration Management
    - Discussion forums for community engagement
    - Peer review system for evaluating and learning from each other's work
    - Access to experts and mentors for guidance and support

Summary of Requirements (Initial Requirements)

The following requirements have been identified for the initial development phase of the Kubernetes Learning and Evaluation Portal:

1. User Management
    - User authentication and authorization
    - Account creation and management
    - Password reset
2. Learning Content Management
    - Display Kubernetes learning materials including articles, tutorials, and video courses
    - Search and filtering of learning materials by category and topic
    - Bookmarking of learning materials
3. Interactive Learning Experiences Management
    - Display coding challenges and quizzes
    - Immediate feedback on progress
    - Progress tracking
4. Hands-on Exercises and Projects Management
    - Provision of a Kubernetes cluster for completing hands-on exercises and projects
    - Submission of completed exercises and projects for evaluation
5. Automated Assessments Management
    - Automated assessments to evaluate understanding of Kubernetes concepts
    - Immediate feedback on assessments
    - Progress tracking
6. Community and Collaboration Management
    - Discussion forums for community engagement
    - Peer review system for evaluating and learning from each other's work
    - Access to experts and mentors for guidance and support

2.1.2. Identifying External Entities

Phase 1: **Over-specifying Entities from Abstract**

External entities refer to entities that are external to the system but interact with it in some way. In the context of the Kubernetes Learning and Evaluation Portal, the following external entities can be identified:

1. Users: Users are external entities who will interact with the system to access learning materials, complete hands-on exercises, and take assessments.
2. Kubernetes Cluster: The Kubernetes cluster is an external entity that will be used by the system to provision resources for hands-on exercises and projects.
3. Payment Gateway: The payment gateway is an external entity that will be used by the system to process payments for premium features.
4. Third-party APIs: The system will use various third-party APIs for functionalities such as user authentication, content delivery, and analytics.
5. Mentors and Experts: Mentors and experts are external entities who will interact with the system to provide guidance and support to users.

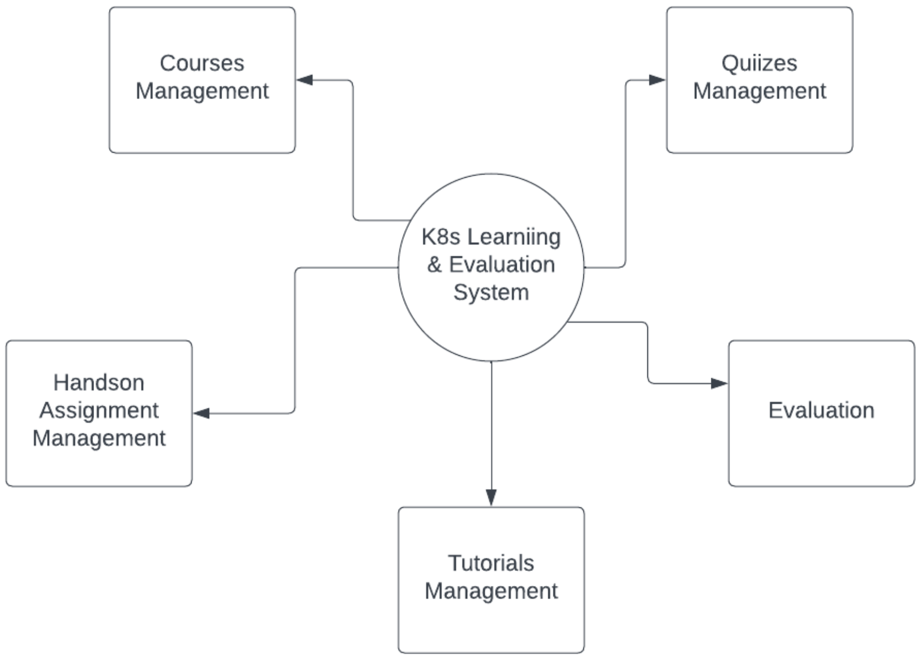
Phase 2: **Refinement**

After over-specifying the external entities, it is important to refine the list by considering their interactions with the system and their impact on it. The refined list of external entities for the Kubernetes Learning and Evaluation Portal is as follows:

1. Users: The system will interact with users through the user interface to provide access to learning materials, hands-on exercises, and assessments. Users will also interact with the system to create and manage their accounts, reset their passwords, and provide feedback.
2. Kubernetes Cluster: The system will use the Kubernetes cluster to provision resources for hands-on exercises and projects. The cluster will be managed externally and accessed through the system's backend.
3. Payment Gateway: The payment gateway will be used by the system to process payments for premium features. The gateway will be integrated with the system's backend and frontend to facilitate payments.
4. Third-party APIs: The system will use various third-party APIs for functionalities such as user authentication, content delivery, and analytics. The APIs will be integrated with the system's backend and frontend to provide the required functionalities.
5. Mentors and Experts: Mentors and experts will interact with the system to provide guidance and support to users. They will access the system through the frontend to participate in discussion forums, peer review systems, and to provide feedback.

.

2.1.3. Context Level Data Flow Diagram:



2.1.4. Capture "shall" Statements:

ID	Requirement	Shall Statement
1.1	User Authentication	The system shall provide a login interface for users to authenticate with their credentials.
1.2	Account Management	The system shall allow users to create, edit, and delete their accounts.
1.3	Course Selection	The system shall provide a list of available courses for users to choose from.
1.4	Learning Materials	The system shall provide access to learning materials, such as videos, articles, and tutorials, for each course.
1.5	Hands-on Exercises	The system shall allow users to complete hands-on exercises for each course using a provisioned Kubernetes cluster.
1.6	Assessments	The system shall provide assessments, such as quizzes and tests, for users to complete after each course.
1.7	Discussion Forums	The system shall provide discussion forums for users

		to interact with mentors, experts, and other users.
1.8	Review System	The system shall allow users to submit their hands-on exercises for review.
1.9	Analytics	The system shall provide analytics on user progress, course completion rates, and other relevant metrics.
1.10	Compatibility	The system shall be compatible with modern web browsers, including Chrome, Firefox, Safari, and Edge.

2.1.5. Allocate Requirements:

Requirements		**Use Case**	**Description**
1.1		Login	Allows the user to log in to their account and access the system.
1.2		Manage Account	Allows the user to create, edit, and delete their account.
1.3		Browse Courses	Allows the user to browse available courses and view their details.
1.3 1.4 1.5		Learning materials	Allows the user to enroll in a course and gain access to its learning materials and exercises.
1.5		Complete Exercise	Allows the user to complete a hands-on exercise in the provisioned Kubernetes cluster.
1.8		Submit Exercise for Review	Allows the user to submit their completed exercise for peer review.
1.6		Take Assessment	Allows the user to take an assessment after completing a course.
1.7		Participate in Discussion Forums	Allows the user to participate in discussion forums with other users, mentors, and experts.
1.9		View Analytics	Allows the user to view their progress and performance analytics.

2.1.6. Prioritize Requirements:

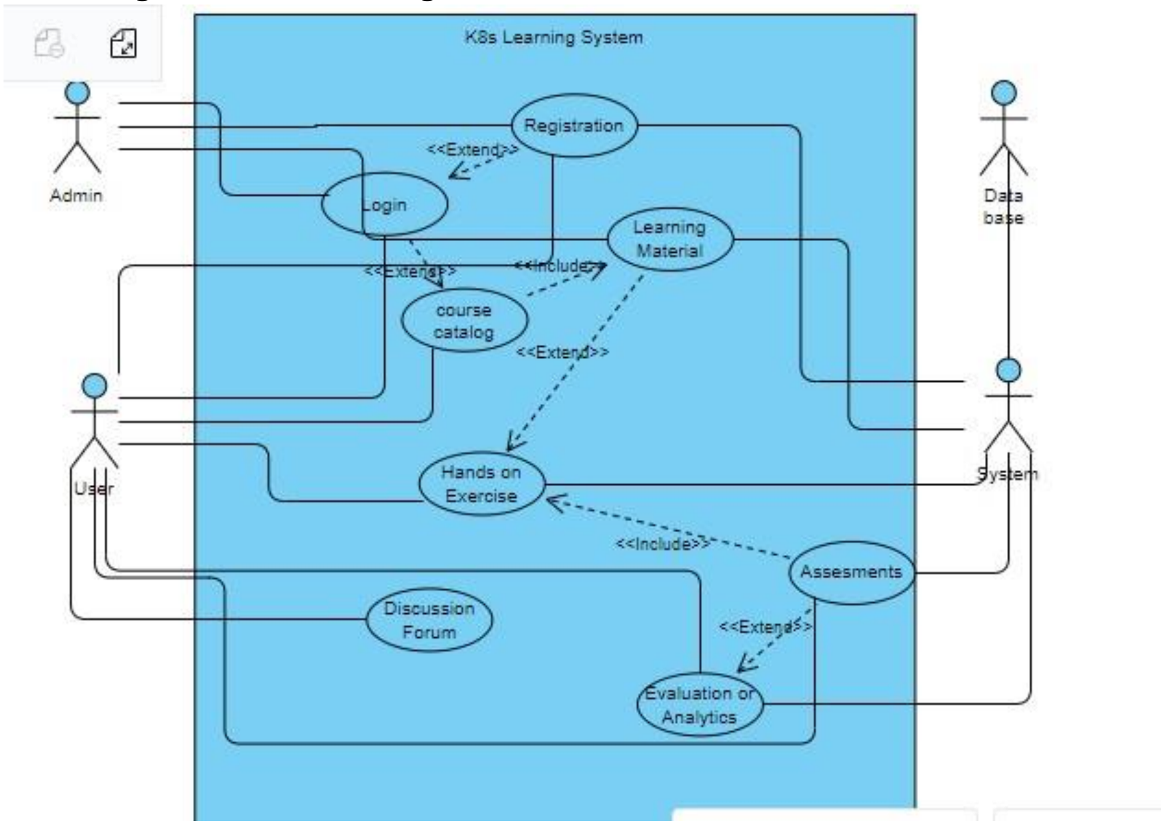
ID	Use Case ID	Use Case	Shall Statement	Priority
1.1	UC01	Login	The system shall require users to enter a valid email and password to access the system.	High
1.2	UC02	Manage Account	The system shall allow users to create and edit their account information, including their email and password.	High
1.3	UC03	Learning material	The system shall allow users to select and enroll in a course of their choice.	High
1.4	UC04	Browse Courses	The system shall display the details of each available course, including the course description and learning objectives.	Medium
1.5	UC05	Enroll in Course	The system shall provide access to the learning materials for each enrolled course, including videos, slides, and text documents.	Medium
1.6	UC06	Complete Exercise	The system shall provision a Kubernetes cluster for each user to complete the hands-on exercises.	High
1.7	UC07	Submit Exercise for Review	The system shall allow users to submit their completed exercises for peer review by other users.	Medium
1.8	UC08	Take Assessment	The system shall provide assessments at the end of each course to evaluate the user's understanding of the material.	High
1.9	UC09	Participate in Discussion Forums	The system shall allow users to participate in discussion forums with other users, mentors, and experts.	Medium
1.10	UC10	View Analytics	The system shall provide users with analytics on their progress and performance, including completion rates and assessment scores.	Low

2.1.7. Requirements Trace-ability Matrix:

Requirement ID	Use Case ID	Shall Statement	Build	Category
1.1	UC01	The system shall require users to enter a valid email and password to access the system.	Build 1	Professional

1.2	UC02	The system shall allow users to create and edit their account information, including their email and password.	Build 1	Professional
1.3	UC03	The system shall allow users to select and enroll in a course of their choice.	Build 1	Professional
1.4	UC04	The system shall display the details of each available course, including the course description and learning objectives.	Build 1	Professional
1.5	UC05	The system shall provide access to the learning materials for each enrolled course, including videos, slides, and text documents.	Build 1	Professional
1.6	UC06	The system shall provision a Kubernetes cluster for each user to complete the hands-on exercises.	Build 2	Professional
1.7	UC07	The system shall allow users to submit their completed exercises for peer review by other users.	Build 2	Professional
1.8	UC08	The system shall provide assessments at the end of each course to evaluate the user's understanding of the material.	Build 2	Professional
1.9	UC09	The system shall provide the option for users to purchase premium features, such as access to expert mentors.	Build 3	Professional
1.10	UC10	The system shall allow users to participate in discussion forums with other users, mentors, and experts.	Build 3	Professional
1.11	UC11	The system shall provide users with analytics on their progress and performance, including completion rates and assessment scores.	Build 3	Professional

2.2.10. High Level Usecase Diagram:



2.2.11. Analysis Level Usecase Diagram:

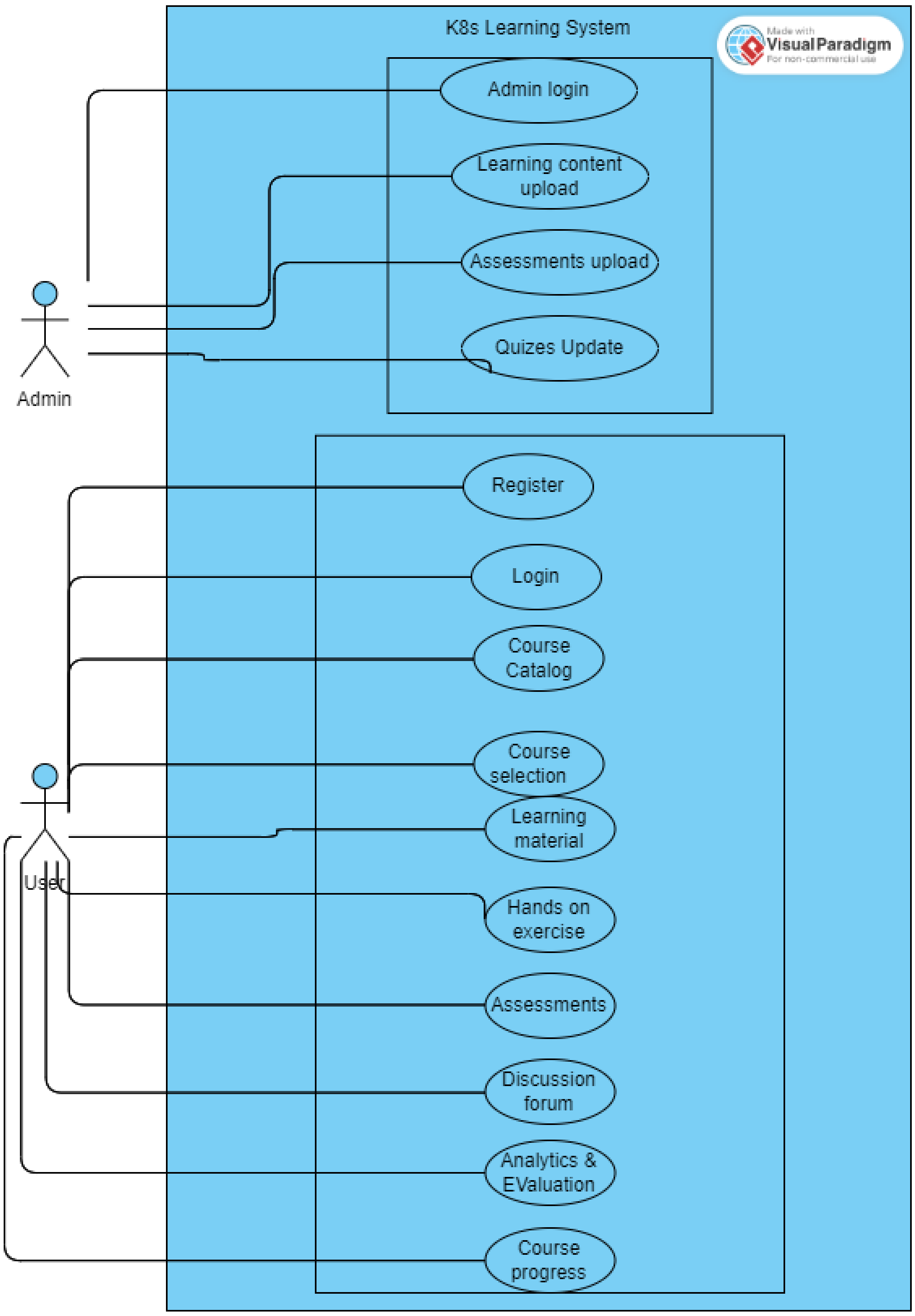


Table: 2.6. open the application

1	**Use Case Name**	**Search for website**
2	**Scope**	App Access
3	**Level**	User Goal
4	**Primary Actor**	User
5	**Stakeholder**	User
6	**Pre-condition**	System must be connected with internet
7	**Post-condition**	User access web application
8	**Success Scenario**	
	User ● open application	**System** ● Show application
9	**Extensions**	● If the application is wrong, user will not be accessible application.

Table: 2.7. Registration.

1	**Use Case Name**	**Registration**
2	**Scope**	Login AP Web application
3	**Level**	User Goal
4	**Primary Actor**	User
5	**Stakeholder**	User
6	**Pre-condition**	None
7	**Post-condition**	User will be logged in.
8	**Success Scenario**	
	User ● Enter Username,last name date of birth, Email, Password,.	**System** ● Send Verification Email ● Save User ● Redirect to Login

9	**Extensions**	● If Email is already taken the system will give warning “email already in use”.

Table: 2.7. Login.

1	**Use Case Name**	**Login**
2	**Scope**	Login application
3	**Level**	User Goal
4	**Primary Actor**	User
5	**Stakeholder**	User
6	**Pre-condition**	User should be registered.
7	**Post-condition**	User will be logged in.
8	**Success Scenario**	
	User ● Enter email and password	**System** ● Verify email and password ● Redirect to profile
9	**Extensions**	● If username or password is wrong system will give warning “Incorrect username/password”.

Table: 2.7. Course catalog

1	**Use Case Name**	**Course catalog**
2	**Scope**	Login application
3	**Level**	User Goal
4	**Primary Actor**	User
5	**Stakeholder**	User
6	**Pre-condition**	User should be logged in.
7	**Post-condition**	User can be used profile content
8	**Success Scenario**	

	User ● User logged in application	**System** ● Redirect to the Course caralog

9	**Extensions**	• If the Information is incomplete system will notify that "Please Complete Your Information"

Table: 2.6. Course Selection.

1	**Use Case Name**	**Course Selection**
2	**Scope**	Course Description and selection
3	**Level**	User Goal
4	**Primary Actor**	User
5	**Stakeholder**	User
6	**Pre-condition**	Select course and learning material
7	**Post-condition**	User will learn for material
8	**Success Scenario**	
	User • User will manage tutorials and learn from material	**System** • System perform reload the learning material
9	**Extensions**	• If quiz is on ahead the user will have to complete the tutorial to access quiz

Table: 2.6. Assessments.

1	**Use Case Name**	**Assessments**
2	**Scope**	Evaluation for learning
3	**Level**	User Goal
4	**Primary Actor**	User
5	**Stakeholder**	User
6	**Pre-condition**	Select course and learning material
7	**Post-condition**	User evaluation quizzes and handson

		excercises
8	**Success Scenario**	
	User ● User will check analytics of its learning and progress	**System** ● System perform reload the progress of course
9	**Extensions**	● If quiz is on ahead the user will have to complete the tutorial to access quiz

Chapter 3: Software Requirement Specification and Design Document (For structured Approach)

3.1. Introduction:

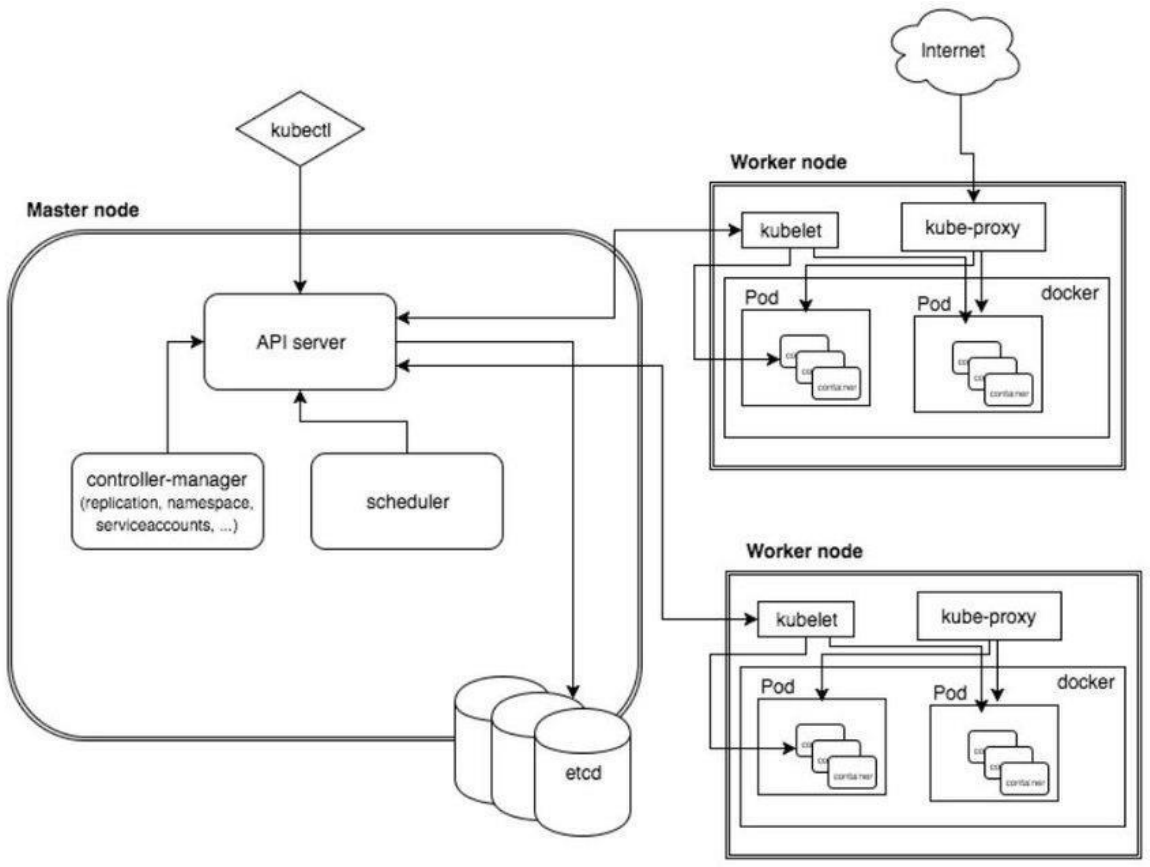
Analysis & Design Model for structured approach must contain following artifacts:

1. Entity Relationship Diagram
2. Data Flow Diagram (Functional Model)
3. State Transition Diagram (Behavioral Model)
4. Architecture Design
5. Component Level Design

3.2. Entity Relationship Diagram:

In the analysis model, Entity Relationship Diagram is used to understand the system under consideration with respect to entities involved and their relationships. Each entity is documented by extracted its attributes, cardinality, and modality.

Example:

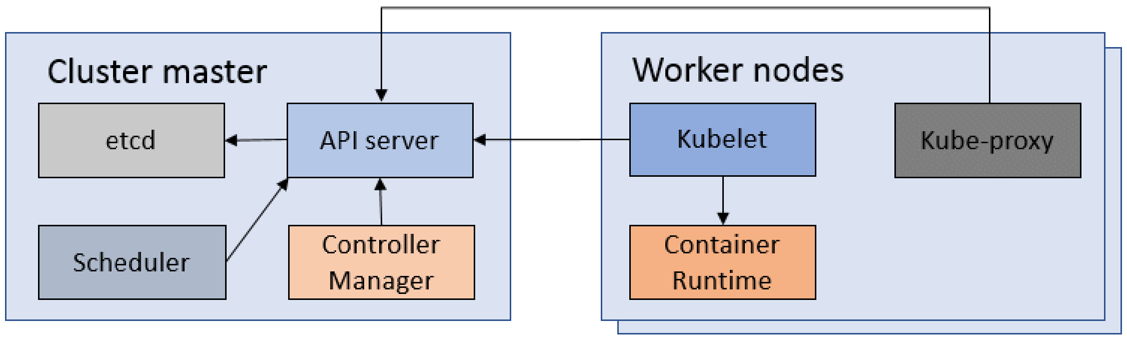


Cardinality
In a Kubernetes cluster, a single master node orchestrates and coordinates various operations within the cluster. This pivotal component ensures the smooth execution of tasks and management of resources. The master node encompasses essential elements such as the API Server, etcd, Controller Manager, and Scheduler, which collectively contribute to the seamless functioning of the cluster.

Identify Attributes
Master Node, Worker Nodes, Pods, Services, Deployment Strategies and Scaling

3.3. Data flow diagram (Functional Model)

DFD is all about to identify the major processes in your system and develop Data Flow Diagram up to required level.



1. **Define Scope:** Clearly define the system's scope, inputs, and outputs.
2. **Identify Data Flow:** Understand data movement within the system.
3. **Break Down Processes:** Decompose system into data-handling processes.
4. **Create Context Diagram:** Represent entire system with external entities.
5. **Refine Processes:** Break processes into sub-processes for detail.
6. **Identify Data Stores and Entities:** Recognize data stores and external entities.
7. **Draw Lower-Level DFDs:** Detail data flow within sub-processes.
8. **Use Standard Symbols:** Apply consistent symbols for processes, data flows, stores, and entities.
9. **Label Components:** Clearly label each component for clarity.
10. **Connect Components:** Use arrows to depict data flow and direction.
11. **Review and Revise:** Ensure coherence and accuracy.
12. **Document Symbols:** Include a legend explaining symbols.
13. **Maintain Clarity:** Prioritize simplicity for easy comprehension.
14. **Use Diagramming Tools:** Utilize software for neat diagrams.
15. **Add Descriptions:** Include context-providing descriptions.
16. **Iterate and Improve:** Continuously refine based on feedback and understanding.

DFDs visually elucidate data flow, aiding understanding of system operations.

3.4. State Transition Diagram

State Transition Diagram is developed to represent the behavior of the system under consideration. The constructs of STD are as follows:

State
A set of observable circum-stances that characterizes the behavior of a system at a given time

State transition
The movement from one state to another

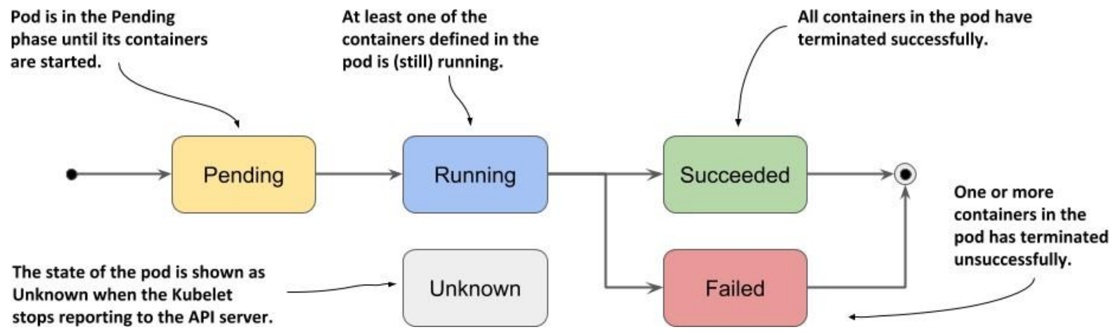
Event
An occurrence that causes the system to exhibit some predictable form of behavior

Action
Process that occurs as a consequence of making a transition

Guidelines for developing a state transition diagram

- Make a list of the different states of a system (How does the system behave?)
- Indicate how the system makes a transition from one state to another (How does the system change state?)
- Indicate event
- Indicate action
- Draw a state transition diagram

Example


Pod is in the Pending phase until its containers are started.
At least one of the containers defined in the pod is (still) running.
All containers in the pod have terminated successfully.
Pending
Running
Succeeded
One or more containers in the pod has terminated unsuccessfully.
The state of the pod is shown as Unknown when the Kubelet stops reporting to the API server.
Unknown
Failed

State Transition Diagram for Microwave

3.5. Architectural design

The Focus of architecture design is the Mapping of Requirements into Software Architecture. DFD prepared in analysis model is analyzed to do it.

Two major structural patterns or two major alternatives are Transform (Flow) Analysis and Transaction (Flow) Analysis.

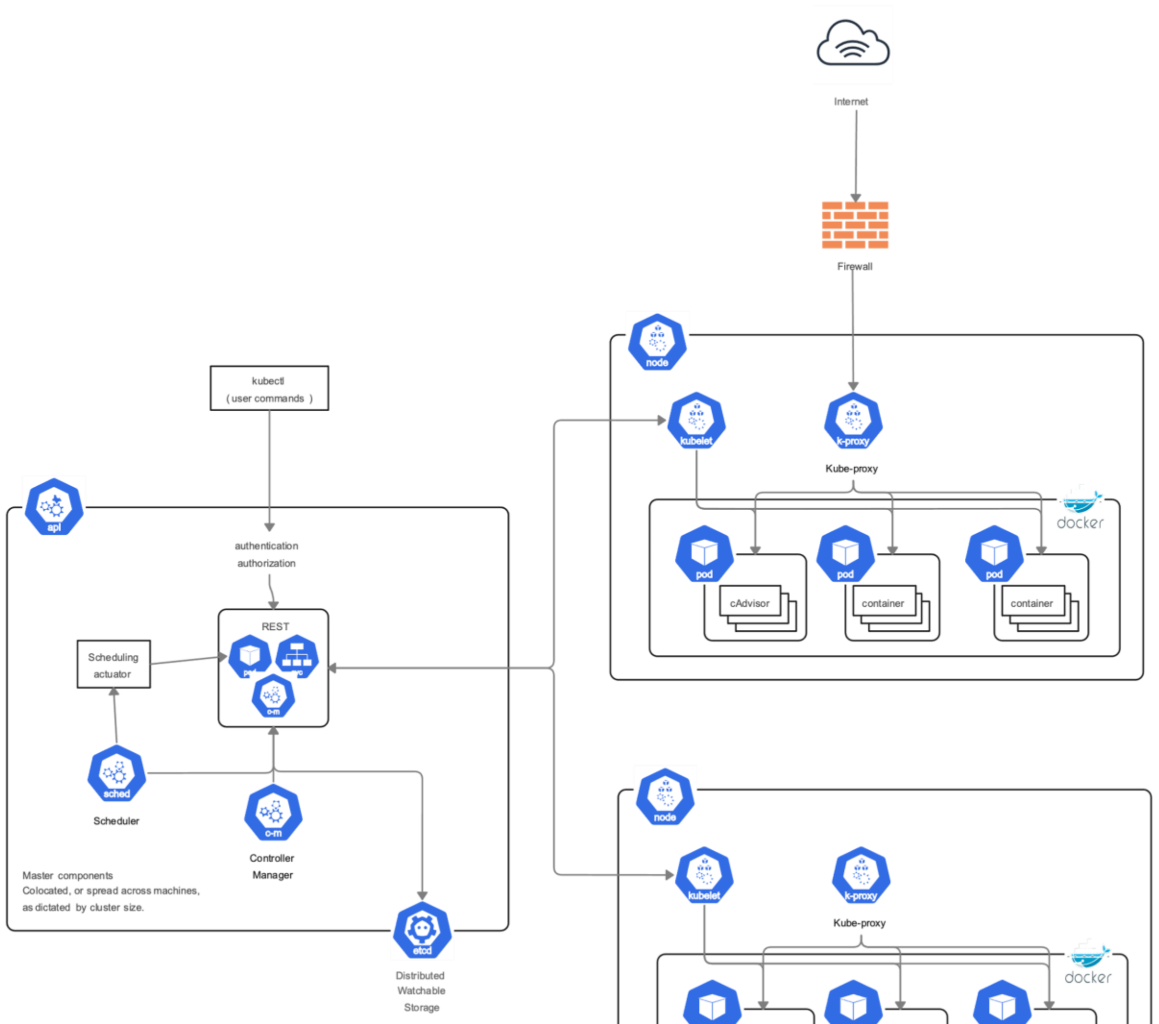
Beginning the Design Process

- Review the fundamental system model i.e. DFD and Software Requirement Specification document.
- Review and refine data flow diagrams for the software
- Determine whether DFD exhibits transform or transaction characteristics

Example

- There is a string conversion system.
- It has the ability to reverse strings, count number of characters, and append new strings with an old string.
- A user would be using this system and would be providing the string to it. The string would be validated. If approved the system would be displaying different choices including reversal of string, character counting and appending of strings.
- The user would select a choice and enter the appropriate selection. According to the selected choice the system would perform the required action.
- If “Reverse String” is selected the valid string is attained and reversed.
- If “Count Characters” is selected the valid string is attained and the number of characters are counted.
- If “Append String” is selected the valid string is attained and the user also enters a new string. Both the strings are appended together and the result produced.
- Whatever the choice selected the result is displayed.

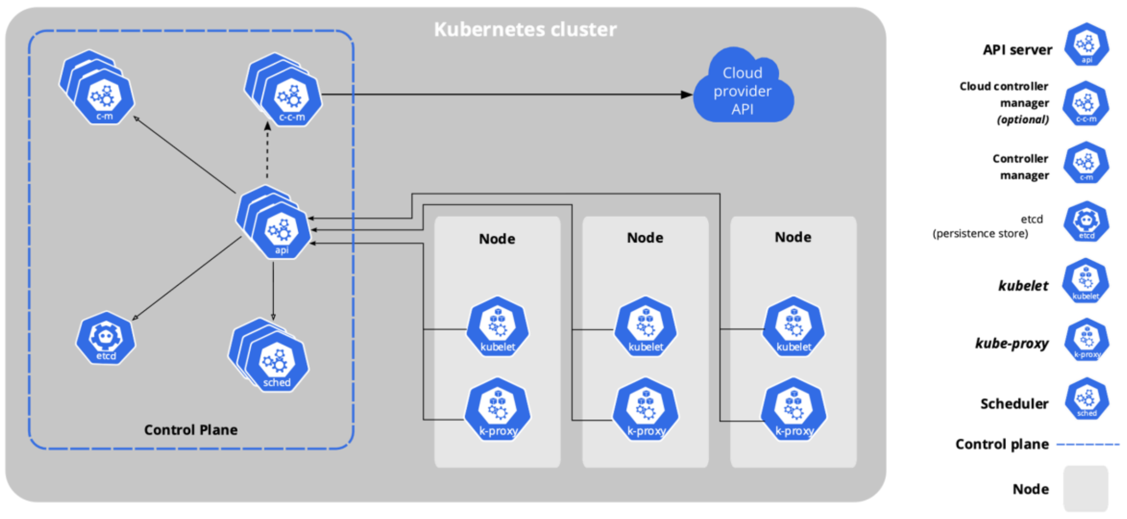
See the diagrams on next page.


Internet
Firewall
node
kubelet
k-proxy
Kube-proxy
docker
pod
cAdvisor
container
kubectl
(user commands)
api
authentication
authorization
REST
Scheduling
actuator
sched
Scheduler
c-m
Controller
Manager
Master components
Colocated, or spread across machines,
as dictated by cluster size.
etcd
Distributed
Watchable
Storage

3.6. Component Level Design

Every component, which is appearing in program structure/design architecture, is logically analyzed and pseudocode or flow chart is prepared for each module. This flow chart is then given to programmer who translates it into a machine-readable code. The options available for component level design are:

- Flow chart
- Box Diagram
- Decision Table
- Psuedocode



Chapter 4: User Interface Design

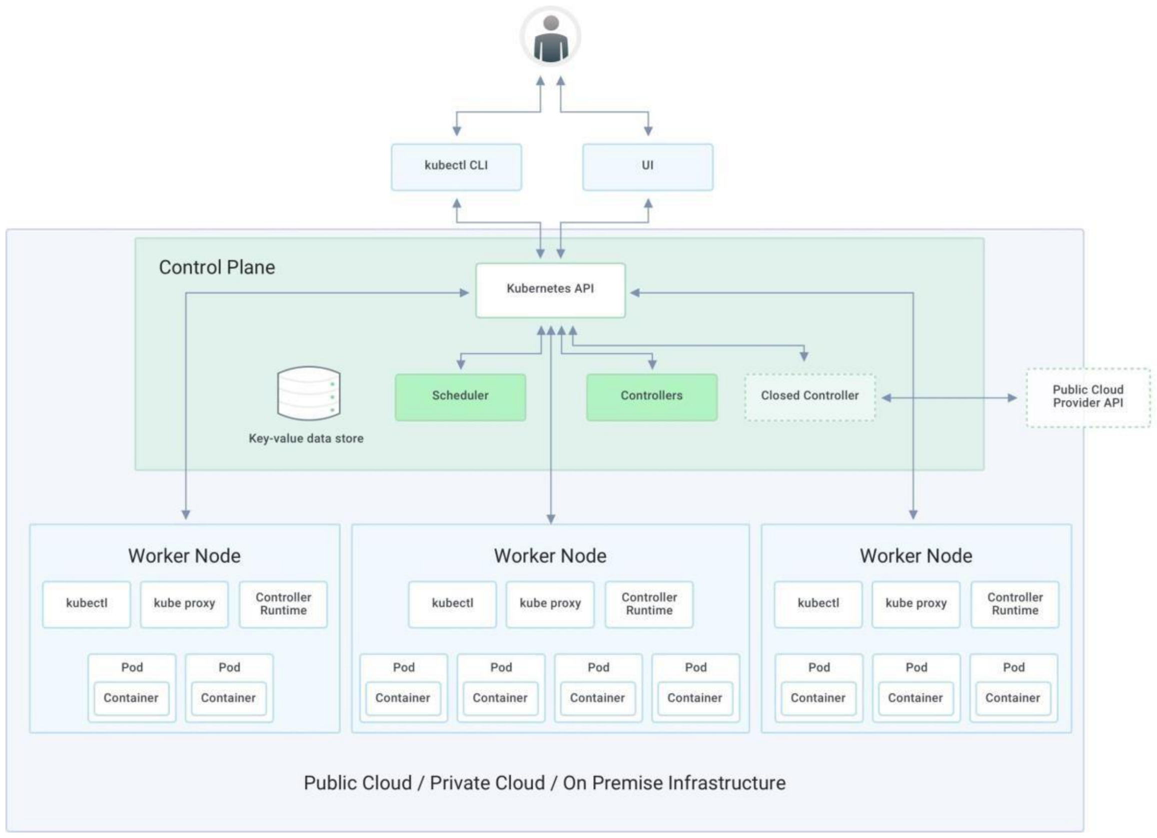
4.1. Introduction

A user interface design consists of three main parts:

Page elements should be visualized on paper before building them in the computer. Just as you draw a site map to plan the site, use cartoons and storyboards to begin blocking out the site's appearance and navigational scheme.

1. Site maps
2. Storyboards
3. Navigational maps
4. Traceability Matrix

4.2. Site Maps



4.3. Story boards

Once upon a time, in the realm of cloud-native computing, there existed a powerful orchestrator known as Kubernetes. It was the hero of developers, system administrators, and operations teams, bringing order to the chaos of containerized applications.
As we venture deeper into this digital landscape, we come across the majestic Home. Here, the journey begins with an Introduction to Kubernetes, where curious souls learn about its magic. They discover its Features and Benefits, understanding how Kubernetes simplifies deployment, scaling, and management of applications.
Eager to embark on their own Kubernetes adventure, our travelers delve into Getting Started. They follow the Installation Guide, equipping themselves with the tools to conquer Kubernetes. With their cluster ready, they learn about Kubernetes Concepts – Pods, the basic building blocks, and Services, the envoys for application communication. Deployments, akin to a grand orchestration, ensure applications are replicated, managed, and resilient.
As the story progresses, our heroes explore the Architecture of Kubernetes. They uncover the secrets of Master Node Components: the wise API Server, the memory keeper etcd, the diligent Controller Manager, and the scheduler with its art of orchestration. The Worker Node Components also reveal themselves – the trusty Kubelet, the guardian Kube Proxy, and the mighty Container Runtime.
With newfound knowledge, our adventurers dive into Deploying Applications. Using kubectl, they command the Kubernetes universe. YAML Configuration becomes their incantation, allowing them to define the desired state of their applications. And then

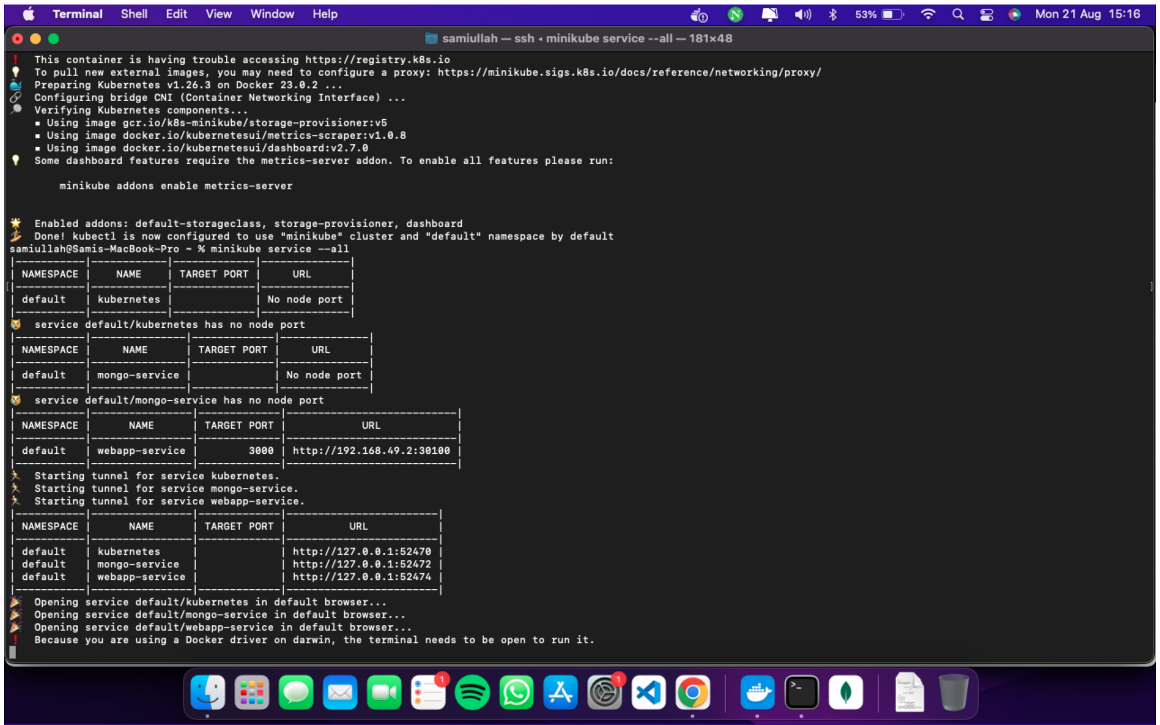
there's Helm, the helm chart, steering their applications to the right course. They also discover the magic of Operators, bringing domain-specific expertise to applications.
Scaling and Load Balancing beckon our explorers next. The enigmatic Horizontal Pod Autoscaling helps applications adapt to varying demands. Services, the bridge between applications and the world, guide traffic with Load Balancing. Ingress Controllers emerge as the gatekeepers to their kingdom, routing requests to the right services.
A new chapter emerges: Managing State. Persistent Volume Claims offer storage for applications with memory. Storage Classes lay the foundation for data persistence, while StatefulSets nurture stateful applications, creating an order among data chaos.
Networking unfolds as another realm. Service Discovery aids applications in finding each other, and Network Policies provide boundaries to the network realm.
But what is power without responsibility? Our adventurers learn about Security. They delve into Authentication and Authorization, ensuring only the rightful wielders have access. Role-Based Access Control (RBAC) assigns roles and permissions, while Pod Security Policies set the rules for pods.
Monitoring and Logging beckon them next. Prometheus becomes their beacon, shining light on system health, while Grafana paints a vivid picture. The ELK Stack emerges – Elasticsearch, Logstash, and Kibana – revealing the tales hidden within logs.
In the realm of CI/CD and DevOps, our heroes integrate Kubernetes into their pipelines. GitOps becomes their mantra, as Continuous Deployment unfolds with precision. Canary Deployments ensure a gentle rollout of new features.
The saga is enriched with Best Practices. Application Design becomes an art as they consider microservices and decoupling. Resource Management becomes a responsibility, as they allocate CPU and memory wisely. High Availability becomes a mission, and Disaster Recovery plans are forged.
Guided by the compass of Community and Resources, they explore Kubernetes Documentation, GitHub Repositories, and join online Communities and Forums. A Glossary deciphers the terminology, while FAQs answer their burning questions. And when they seek help, Contact Us is there to extend a helping hand.
As our heroes conclude their journey through the Kubernetes realm, they've not just learned the ways of containers and orchestration, but also the art of crafting resilient, scalable, and efficient applications.

4.4. Navigational maps:

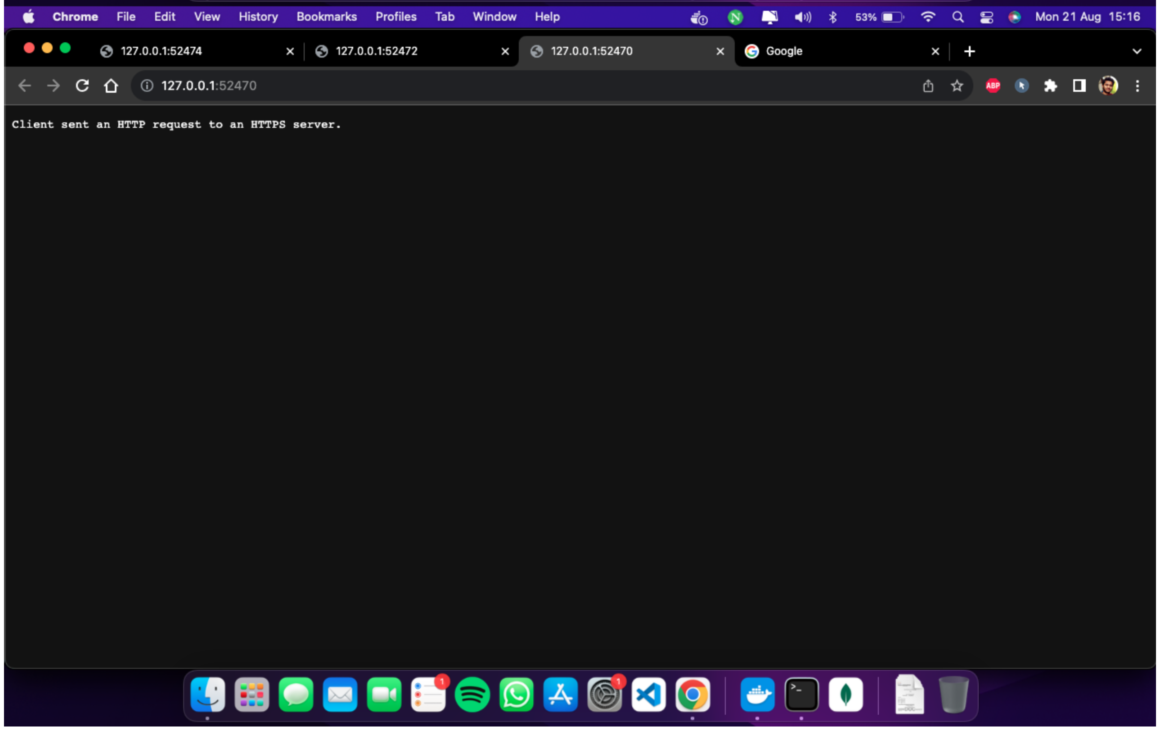
1. **Introduction**
2. **Getting Started**
   - Installation
   - Cluster Setup
   - Kubernetes Concepts
3. **Architecture**
   - Master Node
   - Worker Node
4. **Deployment**
   - kubectl
   - YAML Configuration

- Helm Charts
- Operators

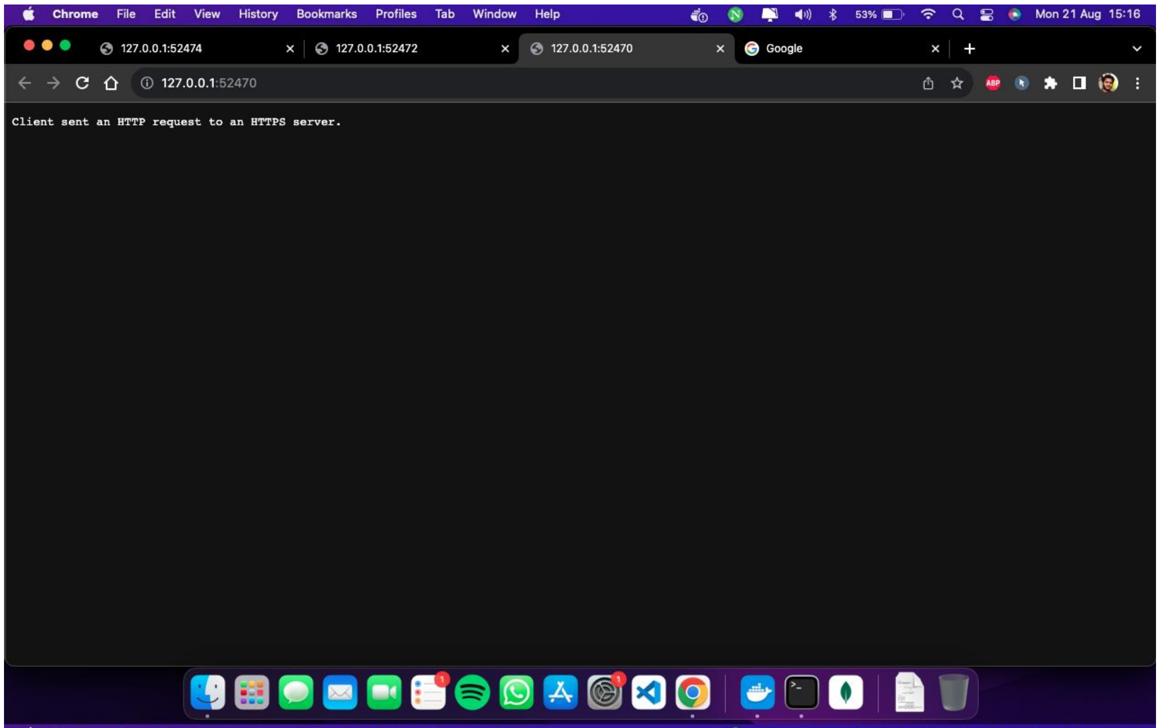
5. **Scaling and Balancing**
    - Horizontal Autoscaling
    - Service Balancing
    - Ingress Control
6. **State Management**
    - Volume Claims
    - Storage Classes
    - StatefulSets
7. **Networking**
    - Discovery
    - Models
    - Policies
8. **Security**
    - Authentication
    - RBAC
    - Security Policies
    - Secrets
9. **Monitoring & Logging**
    - Prometheus
    - Grafana
    - ELK Stack
10. **CI/CD and DevOps**
    - Pipeline Integration
    - GitOps & Deployment
    - Canary Deployment
11. **Best Practices**
    - Design
    - Resource Management
    - Availability
    - Recovery
12. **Community & Resources**
    - Documentation
    - GitHub
    - Blog & Events
    - Communities & Forums
13. **Glossary**
14. **FAQs**
15. **Contact Us**



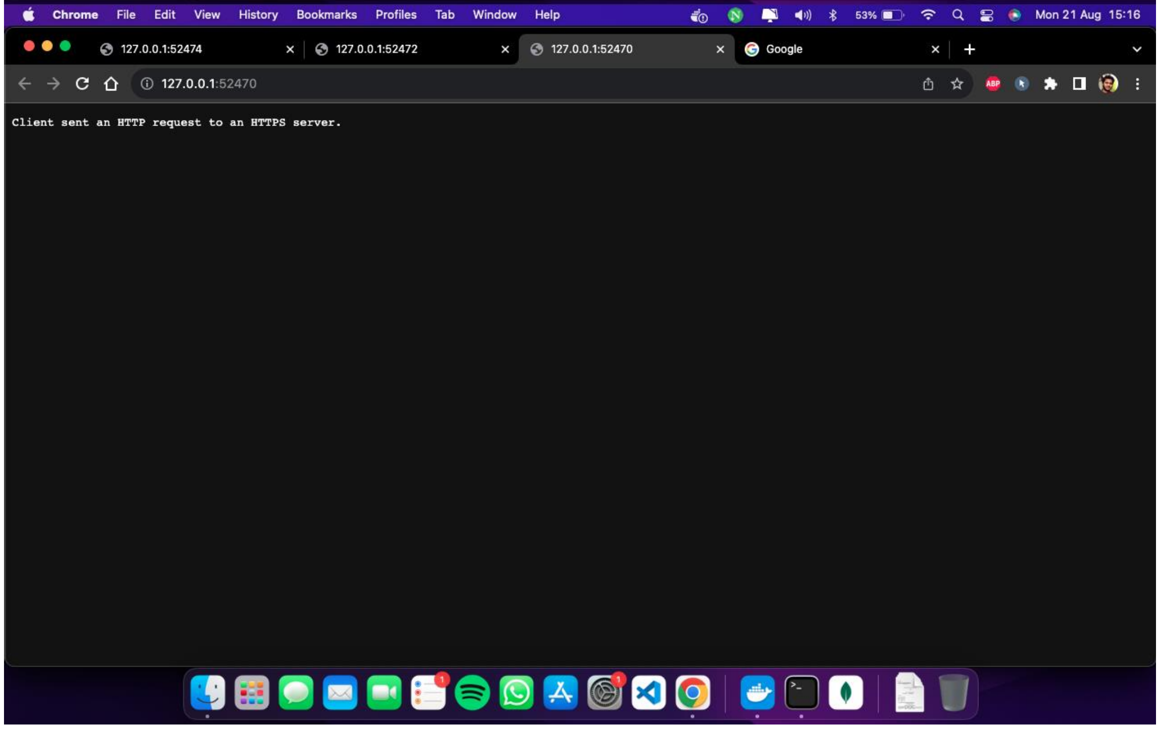
```
❗  This container is having trouble accessing https://registry.k8s.io
💡  To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
🐳  Preparing Kubernetes v1.26.3 on Docker 23.0.2 ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
    ▪ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
    ▪ Using image docker.io/kubernetesui/dashboard:v2.7.0
💡  Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server


🌟  Enabled addons: default-storageclass, storage-provisioner, dashboard
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
samiullah@Samis-MacBook-Pro ~ % minikube service --all
|-----------|------------|-------------|--------------|
| NAMESPACE |    NAME    | TARGET PORT |     URL      |
|-----------|------------|-------------|--------------|
| default   | kubernetes |             | No node port |
|-----------|------------|-------------|--------------|
😿  service default/kubernetes has no node port
|-----------|---------------|-------------|--------------|
| NAMESPACE |     NAME      | TARGET PORT |     URL      |
|-----------|---------------|-------------|--------------|
| default   | mongo-service |             | No node port |
|-----------|---------------|-------------|--------------|
😿  service default/mongo-service has no node port
|-----------|----------------|-------------|---------------------------|
| NAMESPACE |      NAME      | TARGET PORT |            URL            |
|-----------|----------------|-------------|---------------------------|
| default   | webapp-service |        3000 | http://192.168.49.2:30100 |
|-----------|----------------|-------------|---------------------------|
🏃  Starting tunnel for service kubernetes.
🏃  Starting tunnel for service mongo-service.
🏃  Starting tunnel for service webapp-service.
|-----------|----------------|-------------|------------------------|
| NAMESPACE |      NAME      | TARGET PORT |          URL           |
|-----------|----------------|-------------|------------------------|
| default   | kubernetes     |             | http://127.0.0.1:52470 |
| default   | mongo-service  |             | http://127.0.0.1:52472 |
| default   | webapp-service |             | http://127.0.0.1:52474 |
|-----------|----------------|-------------|------------------------|
🎉  Opening service default/kubernetes in default browser...
🎉  Opening service default/mongo-service in default browser...
🎉  Opening service default/webapp-service in default browser...
❗  Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```



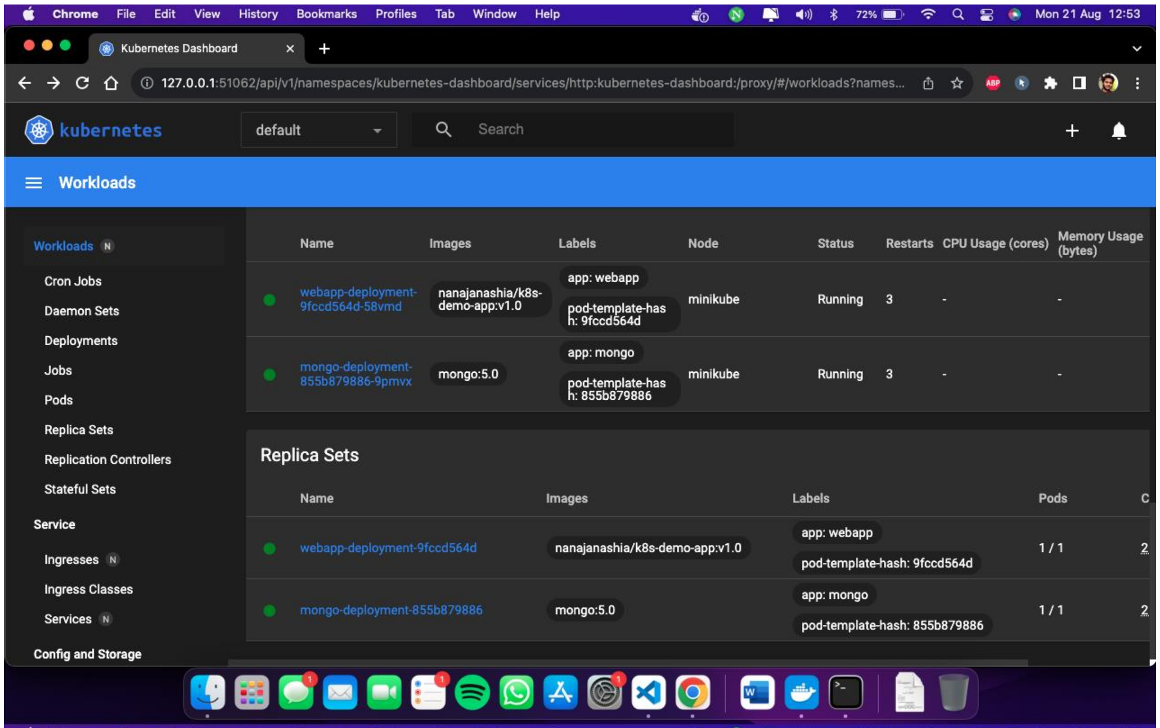
```
Client sent an HTTP request to an HTTPS server.
```



```
Client sent an HTTP request to an HTTPS server.
```



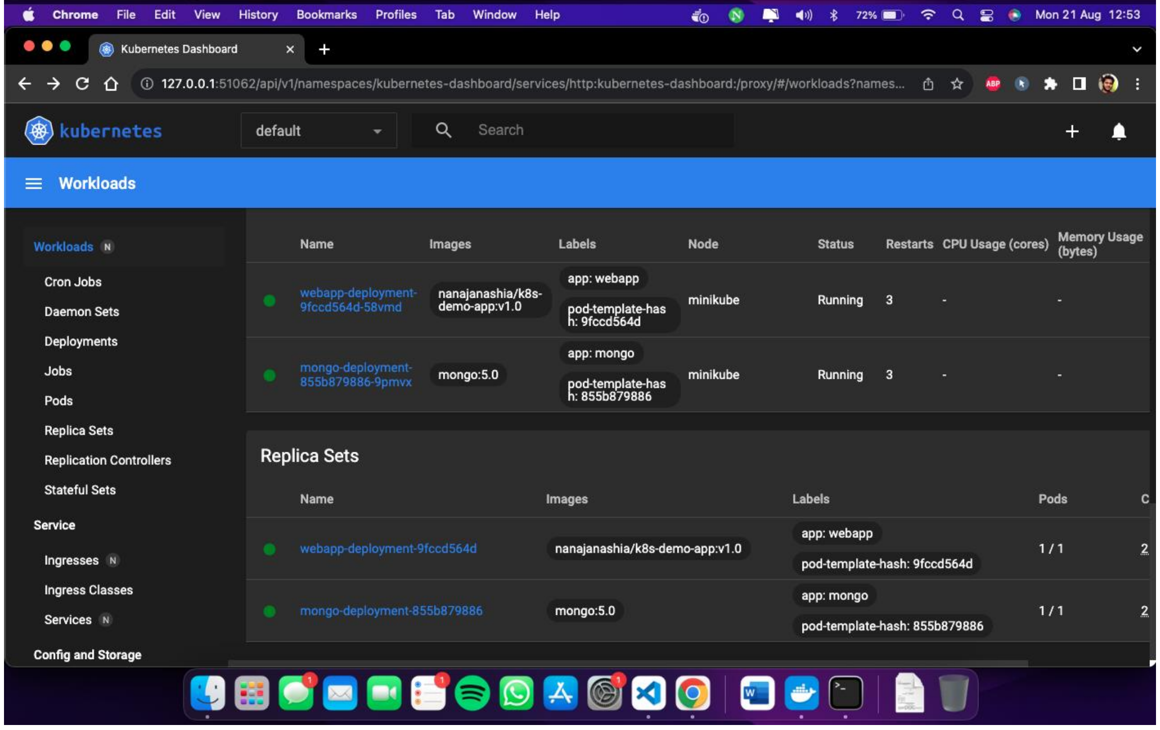
```
Client sent an HTTP request to an HTTPS server.
```



Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)
webapp-deployment-9fccd564d-58vmd	nanajanashia/k8s-demo-app:v1.0	app: webapp pod-template-hash: 9fccd564d	minikube	Running	3	-	-
mongo-deployment-855b879886-9pmvx	mongo:5.0	app: mongo pod-template-hash: 855b879886	minikube	Running	3	-	-

Replica Sets

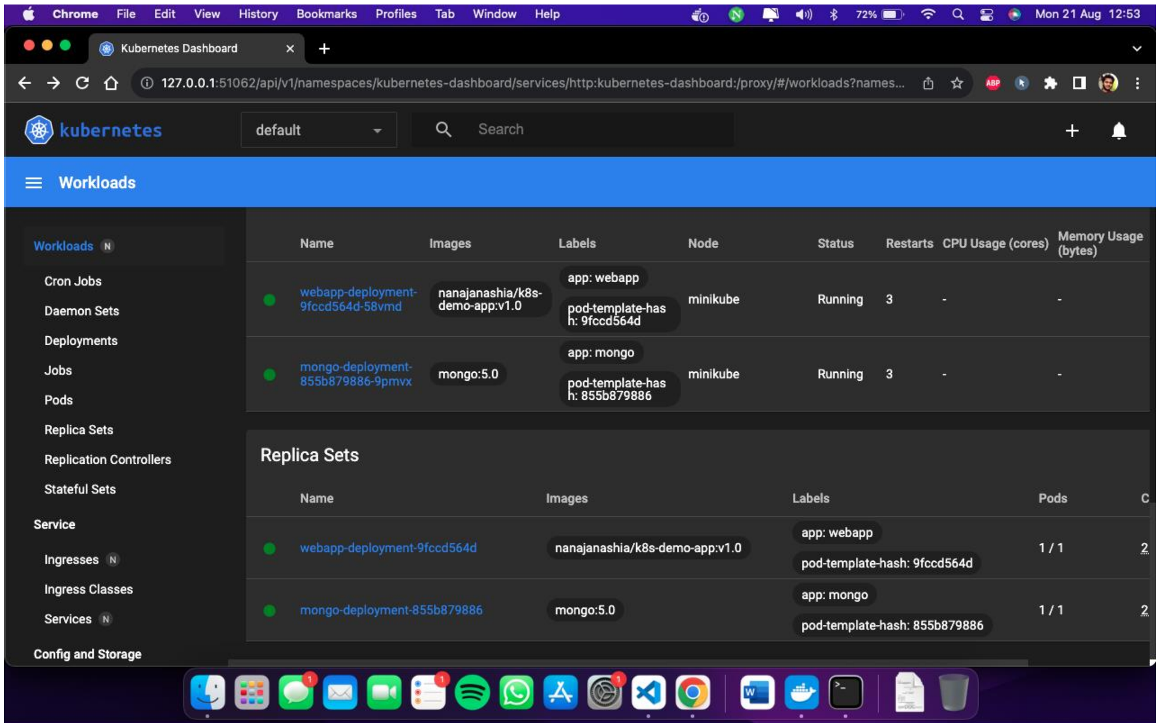
Name	Images	Labels	Pods	C
webapp-deployment-9fccd564d	nanajanashia/k8s-demo-app:v1.0	app: webapp pod-template-hash: 9fccd564d	1 / 1	2
mongo-deployment-855b879886	mongo:5.0	app: mongo pod-template-hash: 855b879886	1 / 1	2



Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)
webapp-deployment-9fccd564d-58vmd	nanajanashia/k8s-demo-app:v1.0	app: webapp pod-template-hash: 9fccd564d	minikube	Running	3	-	-
mongo-deployment-855b879886-9pmvx	mongo:5.0	app: mongo pod-template-hash: 855b879886	minikube	Running	3	-	-

Replica Sets

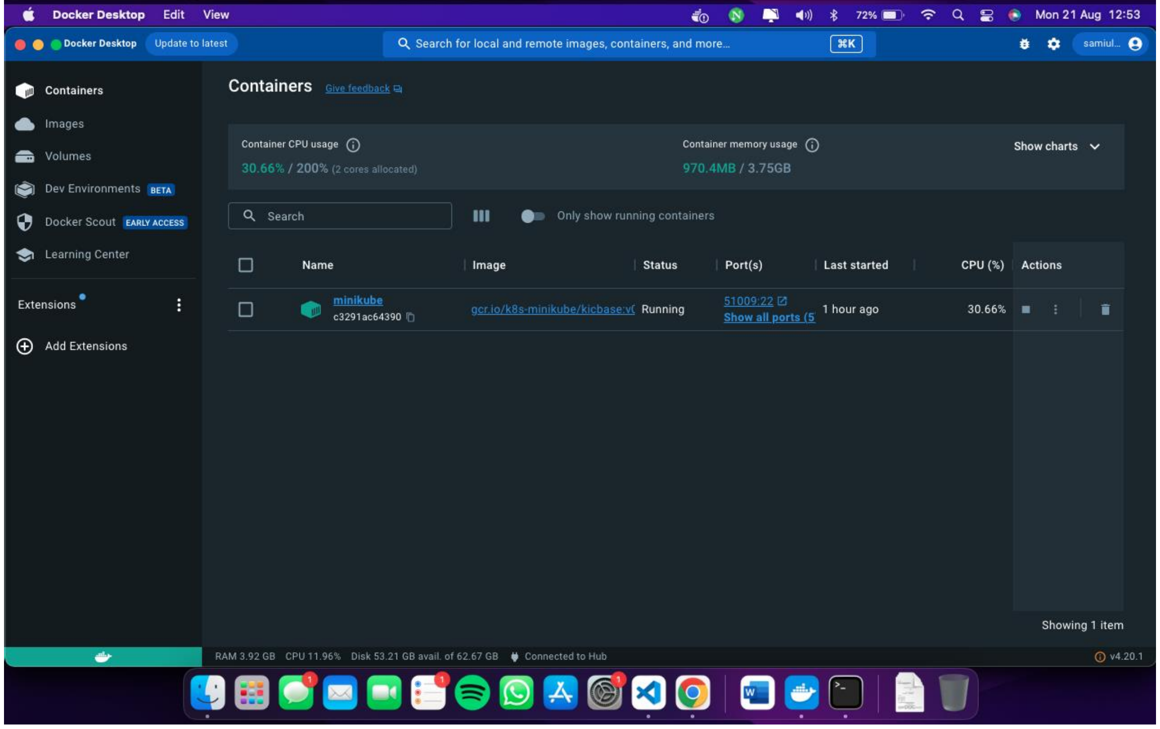
Name	Images	Labels	Pods	C
webapp-deployment-9fccd564d	nanajanashia/k8s-demo-app:v1.0	app: webapp pod-template-hash: 9fccd564d	1 / 1	2
mongo-deployment-855b879886	mongo:5.0	app: mongo pod-template-hash: 855b879886	1 / 1	2



Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)
webapp-deployment-9fccd564d-58vmd	nanajanashia/k8s-demo-app:v1.0	app: webapp pod-template-hash: 9fccd564d	minikube	Running	3	-	-
mongo-deployment-855b879886-9pmvx	mongo:5.0	app: mongo pod-template-hash: 855b879886	minikube	Running	3	-	-

Replica Sets

Name	Images	Labels	Pods	C
webapp-deployment-9fccd564d	nanajanashia/k8s-demo-app:v1.0	app: webapp pod-template-hash: 9fccd564d	1 / 1	2
mongo-deployment-855b879886	mongo:5.0	app: mongo pod-template-hash: 855b879886	1 / 1	2



Containers

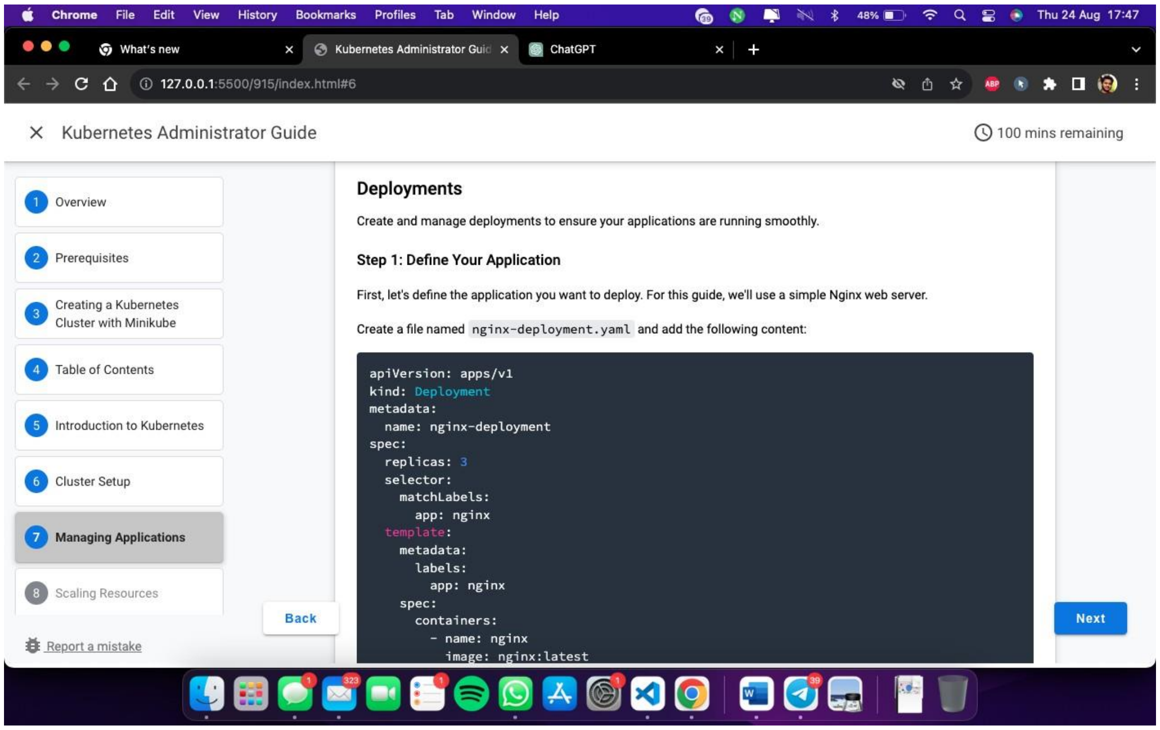
Container CPU usage
30.66% / 200% (2 cores allocated)

Container memory usage
970.4MB / 3.75GB

Name	Image	Status	Port(s)	Last started	CPU (%)	Actions
minikube c3291ac64390	gcr.io/k8s-minikube/kicbase:v(	Running	51009:22 Show all ports (5	1 hour ago	30.66%	

Showing 1 item

RAM 3.92 GB CPU 11.96% Disk 53.21 GB avail. of 62.67 GB Connected to Hub v4.20.1


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Thu 24 Aug 17:47
What's new
Kubernetes Administrator Guid
ChatGPT
127.0.0.1:5500/915/index.html#6
Kubernetes Administrator Guide
100 mins remaining
1 Overview
2 Prerequisites
3 Creating a Kubernetes Cluster with Minikube
4 Table of Contents
5 Introduction to Kubernetes
6 Cluster Setup
7 Managing Applications
8 Scaling Resources
Report a mistake
Back
Next

Deployments

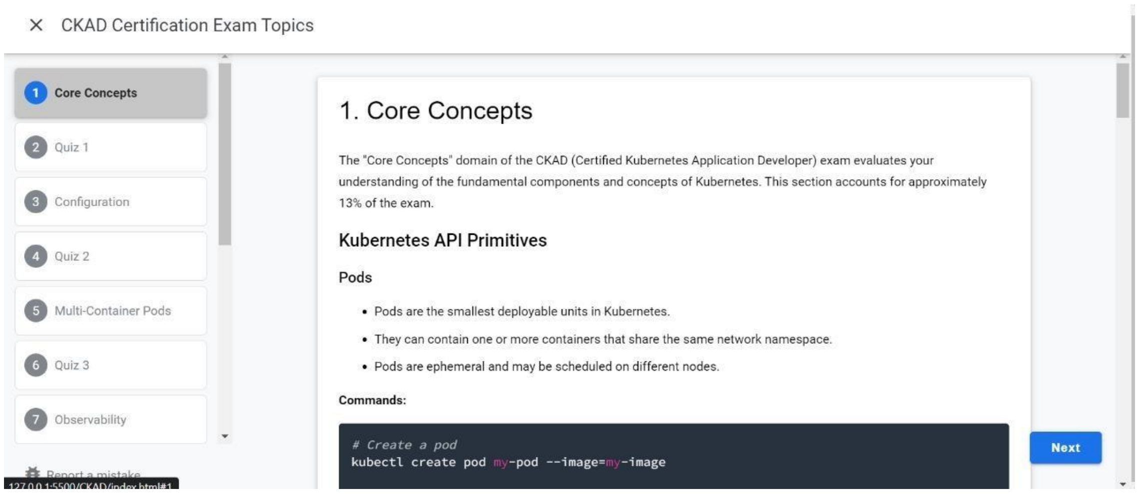
Create and manage deployments to ensure your applications are running smoothly.

Step 1: Define Your Application

First, let's define the application you want to deploy. For this guide, we'll use a simple Nginx web server.

Create a file named `nginx-deployment.yaml` and add the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
```



1. Core Concepts

The "Core Concepts" domain of the CKAD (Certified Kubernetes Application Developer) exam evaluates your understanding of the fundamental components and concepts of Kubernetes. This section accounts for approximately 13% of the exam.

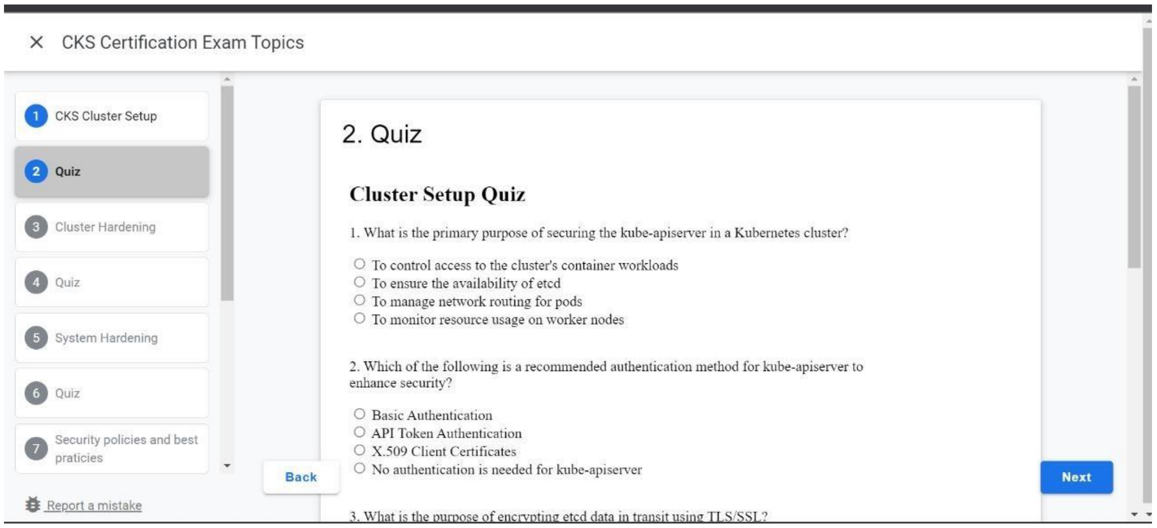
Kubernetes API Primitives

Pods

- Pods are the smallest deployable units in Kubernetes.
- They can contain one or more containers that share the same network namespace.
- Pods are ephemeral and may be scheduled on different nodes.

Commands:

```
# Create a pod
kubectl create pod my-pod --image=my-image
```



2. Quiz

Cluster Setup Quiz

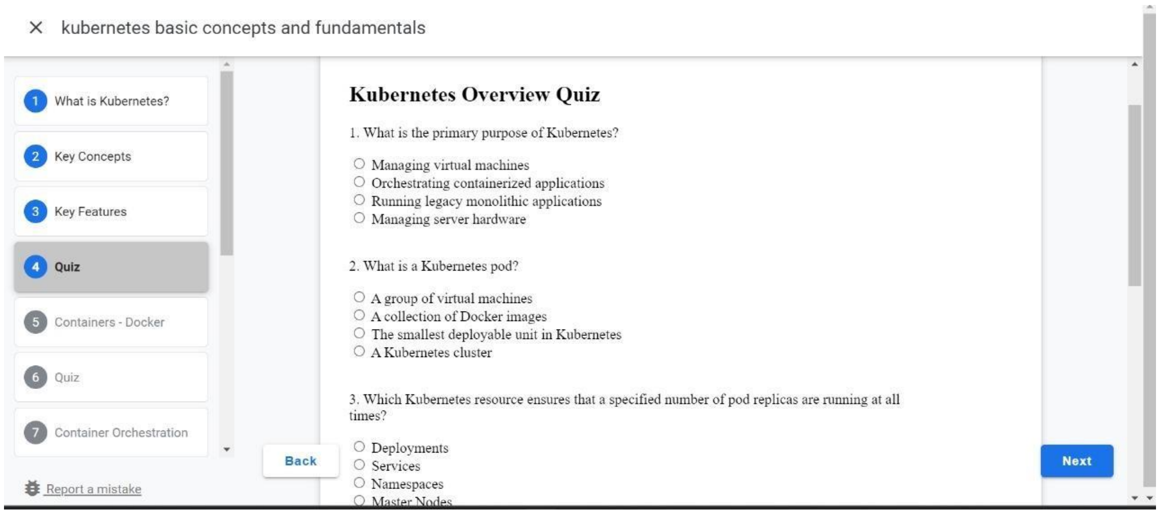
1. What is the primary purpose of securing the kube-apiserver in a Kubernetes cluster?

○ To control access to the cluster's container workloads
○ To ensure the availability of etcd
○ To manage network routing for pods
○ To monitor resource usage on worker nodes

2. Which of the following is a recommended authentication method for kube-apiserver to enhance security?

○ Basic Authentication
○ API Token Authentication
○ X.509 Client Certificates
○ No authentication is needed for kube-apiserver

3. What is the purpose of encrypting etcd data in transit using TLS/SSL?



Kubernetes Overview Quiz

1. What is the primary purpose of Kubernetes?

○ Managing virtual machines
○ Orchestrating containerized applications
○ Running legacy monolithic applications
○ Managing server hardware

2. What is a Kubernetes pod?

○ A group of virtual machines
○ A collection of Docker images
○ The smallest deployable unit in Kubernetes
○ A Kubernetes cluster

3. Which Kubernetes resource ensures that a specified number of pod replicas are running at all times?

○ Deployments
○ Services
○ Namespaces
○ Master Nodes

Chapter 5: Software Testing

5.1 Introduction:

This deliverable is based on the IEEE standard of software testing i.e. IEEE SOFTWARE TEST DOCUMENTATION Std 829-1998. This standard describes a set of basic test documents that are associated with the dynamic aspects of software testing (i.e., the execution of procedures and code). The standard defines the purpose, outline, and content of each basic document. While the documents described in the standard focus on dynamic testing, several of them may be applicable to other testing activities (e.g., the test plan and test incident report may be used for design and code reviews). This standard may be applied to commercial, scientific, or military software that runs on any digital computer. Applicability is not restricted by the size, complexity, or criticality of the software. However, the standard does not specify any class of software to which it must be applied. The standard addresses the documentation of both initial development testing and the testing of subsequent software releases. For a particular software release, it may be applied to all phases of testing from module testing through user acceptance. However, since all of the basic test documents may not be useful in each test phase, the particular documents to be used in a phase are not specified. Each organization using the standard will need to specify the classes of software to which it applies and the specific documents required for a particular test phase.

The standard does not call for specific testing methodologies, approaches, techniques, facilities, or tools, and does not specify the documentation of their use. Additional test documentation may be required (e.g., code inspection checklists and reports). The standard also does not imply or impose specific methodologies for documentation control, configuration management, or quality assurance. Additional documentation (e.g., a quality assurance plan) may be needed depending on the particular methodologies used.

Following are standard artifacts, which must be included in this deliverable:

1. Test Plan
2. Test Design Specification
3. Test Case Specification
4. Test Procedure Specification
5. Test Item Transmittal Report
6. Test Log
7. Test Incident Report
8. Test Summary Report

1. Test Plan: The Test Plan serves as a strategic blueprint for conducting comprehensive testing within your Kubernetes project. In the context of Kubernetes:

- **Scope:** Specify that you'll be focusing on testing the deployment process, scalability of pods, networking configurations, and the system's resilience to failures.

- **Objectives:** Define your goals, such as identifying potential performance bottlenecks and ensuring that container orchestration functions optimally.
- **Testing Environment:** Describe your Kubernetes cluster setup, including the version of Kubernetes, the number of nodes, and any specific configurations.
- **Testing Types:** Outline your approach to unit testing for individual components, integration testing for interactions between services, and performance testing to evaluate system behavior under different loads.
- **Test Schedule:** Provide a timeline, including start and end dates for each testing phase, accounting for initial setup, testing iterations, and final assessments.
- **Resources:** List tools like **kubectl**, testing frameworks, and monitoring tools. Also, detail the hardware and personnel required for executing tests.
- **Risks and Mitigations:** Identify potential risks, such as cluster downtime during testing, and propose mitigation strategies like setting up a parallel testing environment.
- **Exit Criteria:** Define conditions that signify completion, like a successful deployment, no critical issues, and meeting predefined performance thresholds.

2. Test Design Specification: The Test Design Specification elaborates on your chosen testing strategies and methodologies within the Kubernetes environment:

- **Test Approach:** Detail your approach—perhaps black-box testing for validating user interactions and white-box testing for scrutinizing internal components.
- **Test Data:** Provide concrete examples of input data and configurations for each test, including boundary cases and corner scenarios.
- **Test Environment:** Specify the Kubernetes cluster settings, versions of Kubernetes components, and any additional tools like Prometheus for monitoring.
- **Test Cases:** Highlight individual scenarios, such as testing a rolling update of pods or examining how load balancing behaves in a service.

3. Test Case Specification: The Test Case Specification offers in-depth descriptions of individual test scenarios, specifically within your Kubernetes project:

- **Test Case ID:** Assign a unique identifier to each test case for easy reference.
- **Test Objective:** Clearly state the purpose of each test case, like verifying the behavior of pod scaling.
- **Test Steps:** Provide step-by-step instructions, including **kubectl** commands, API calls, or interactions with Kubernetes resources.
- **Expected Result:** Define the anticipated outcomes, like successful pod scaling resulting in a balanced distribution of workload.
- **Test Data:** Specify any input data, configurations, or resources needed for each test.
- **Preconditions:** List any necessary prerequisites, such as a running Kubernetes cluster with specific configurations.

4. Test Procedure Specification: The Test Procedure Specification offers detailed guidance on executing specific tests within the Kubernetes environment:

- **Procedure ID:** Assign a unique identifier to each procedure, linking it to the associated test case.
- **Test Case ID:** Clearly reference the test case that corresponds to the procedure.
- **Procedure Steps:** Provide precise instructions, including **kubectl** commands, API calls, and specific actions to perform within the Kubernetes environment.

- **Expected Outcome:** Define what the outcome should be at each step, aiding testers in recognizing deviations from expectations.
- **Tools and Resources:** Specify any tools, scripts, or resources required to execute the test procedure effectively.

5. Test Item Transmittal Report: The Test Item Transmittal Report documents the transition of tested software to the next phase or team:

- **Tested Items:** Enumerate the Kubernetes components, features, or deployments that have undergone testing.
- **Test Results:** Summarize the outcomes of tests performed on each item, including pass/fail status and any identified issues.
- **Issues Found:** Detail any discovered defects, anomalies, or performance bottlenecks along with their severity levels.

6. Test Log: The Test Log tracks the execution of test cases, recording outcomes and any deviations from expected results:

- **Date and Time:** Timestamp when the test was executed.
- **Test Case ID:** Identify the specific test case being executed.
- **Test Environment:** Document the Kubernetes cluster's state, version, and any modifications.
- **Actual Outcome:** Record the actual result of the test, highlighting any inconsistencies with the expected outcome.

7. Test Incident Report: The Test Incident Report documents unexpected events during testing:

- **Incident ID:** Assign a unique identifier for each incident.
- **Date and Time:** Note when the incident occurred.
- **Description:** Provide a detailed account of the incident, such as a sudden service crash during load testing.
- **Immediate Actions:** Outline the steps taken to address or mitigate the incident.
- **Root Cause:** If known, explain the underlying reason for the incident.

8. Test Summary Report: The Test Summary Report offers a consolidated overview of the testing process and its outcomes:

- **Testing Scope:** Recap which aspects of Kubernetes were tested, emphasizing deployment, scalability, networking, and system resilience.
- **Testing Results:** Summarize the overall results, including pass rates, failures, and issues found during testing.
- **Lessons Learned:** Reflect on insights gained, acknowledging successful strategies and areas for enhancement.
- **Recommendations:** Offer suggestions for future Kubernetes testing iterations or improvements based on your findings.