MAJOR PROJECT REPORT

Demonstrating high-scalability using Kubernetes

Domain: Distributed Systems and Cloud Computing

By
Sana Simran Khazielakha
160117733007

ABSTRACT:

Today's applications offer multiple services to meet the needs of the customer. For example, a customer interacts with login, search, payment, and delivery (...etc.) services to get a product home delivered from Amazon.in. A traditional (web) application would have all the services as part of a monolithic architecture, where one application runs/supports all the services. But a monolithic architecture based application often breaks under a fluctuating user demand.

To solve this problem, a modern (web) application uses a micro-service architecture to decompose the business logic into discrete micro-services that are easy to deploy, maintain, and scale. Microservices are often containerised using platforms like Docker and are managed by an orchestration framework like Kubernetes. The aim of this project is to demonstrate high-scalability and fault-tolerance (to a certain extent) in micro-service oriented web-application.

INTRODUCTION:

When there is a sale going on any e-commerce website, we usually notice that the website crashes when we reload or during the payment. This usually happens because the system which is hosting the website is not able to hold and manage the traffic requests. This usually happens in a monolithic architecture.

In this project, a web application will be tested under high traffic. Its performance and ability to hold and manage the traffic will be demonstrated. This will be done using a concept called containerisation. A web application, its source code and all of its dependencies will be packed in a virtual ‘container’ using a software called Docker, Docker’s CLI (command-line interface) will help us containerise and send it for further deployment using another software called Kubernetes, which manage and scales these containers in accordance to the application needs.

Using Apache JMeter, ‘Fake Traffic’ or high amount of http requests will be created, which will then be handled by Kubernetes, showing how the resources in the micro-service architecture will be multiplied and reduced as and how needed. This is how high-scalability will be demonstrated in this project.

LITERATURE SURVEY:

BASE PAPERS:

1. Building Modern Clouds: Using Docker, Kubernetes and Google Cloud Platform

This paper has the the following objectives:

- Development of a modern cloud infrastructure or DevOps implementation that both Docker and Kubernetes have revolutionised the era of software development and operations
- Explains concepts like Docker containers and Kubernetes
- Shows the working and implementation of Kubernetes platform
- Hosts Wordpress using docker and Kubernetes

2. Self-hosted Kubernetes: Deploying docker container locally with Minikube

This paper has the following objectives:

- Deploys the container orchestration tool Kubernetes on a local system with a Docker sample container.
- Shows how the configurations and management needed for a Docker container is set successfully on the local system before it is deployed onto the cloud or on the premise.
- Shows how on-premise deployment use case is very important in domains such as finance and healthcare where organisations hesitate to upload confidential information on to the cloud for security reasons but still require scaling of their applications.

OBJECTIVES:

- To build a web-application ‘Foodie’ (recipe system)
- To containerise this web-application into container along with all its dependencies
- To deploy and manage this container using Kubernetes
- To show high-scalability of resources using Kubernetes

SOFTWARE/HARDWARE REQUIREMENTS:

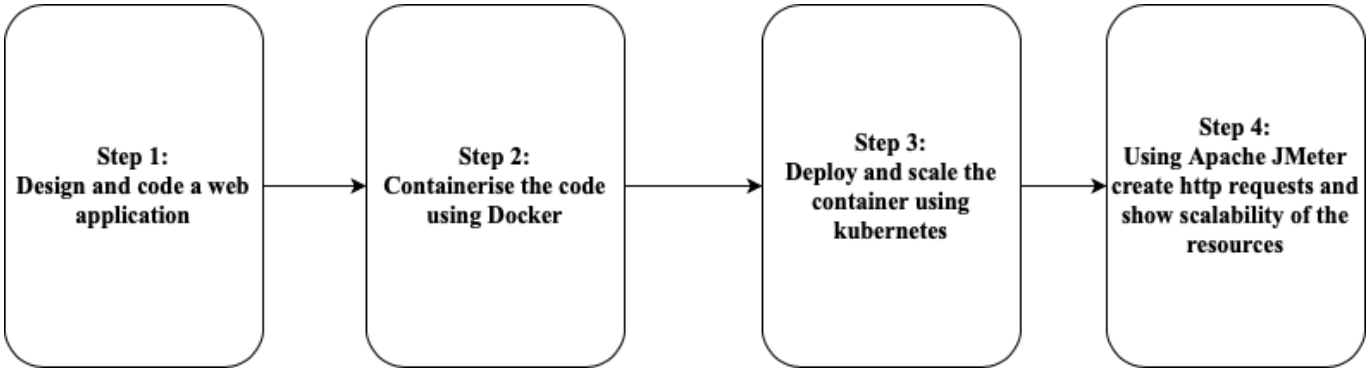
Software Requirements:

- Operating System (Windows/MacOS/Linux)
- Terminal
- Node.js
- React.js
- Docker
- Kubernetes
- Apache JMeter

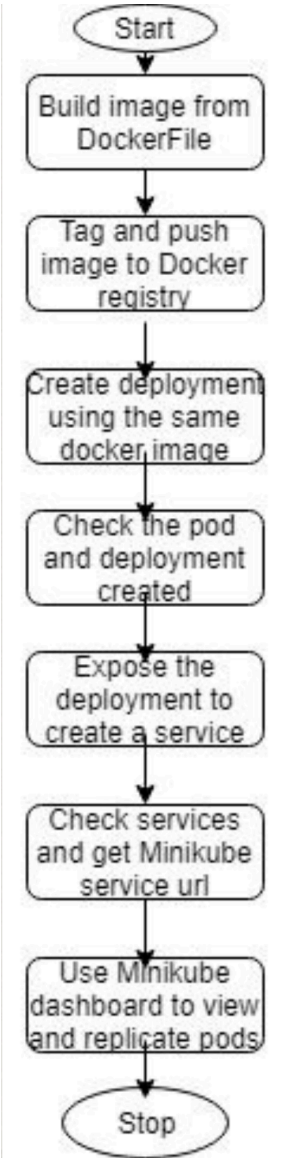
Hardware Requirements:

- CPU with intel i7 or more
- 8 or 16GB RAM
- At least 30 GB of space(SSD)

FLOWCHART:



Between step 2 and step 3:



IMPLEMENTATION:

This whole project can be divided into three steps:

1.Development of Foodie (Web Application)
A web application using React.js and Node.js with MySql as database will be made. This web application 'Foodie' is a recipe system for people who aren't familiar with the art of cooking. This system will have the following functions:

- Create Account
- Login (token-based authentication)
- Search for recipes based on ingredients
- Save Recipes

Features mentioned can be very helpful as not many are familiar with recipe names and what a dish is called, a user will have to just enter the ingredients he has and then recipes based on those ingredients will be shown from various cuisines, which gives the user a broader sense of choices to choose from.

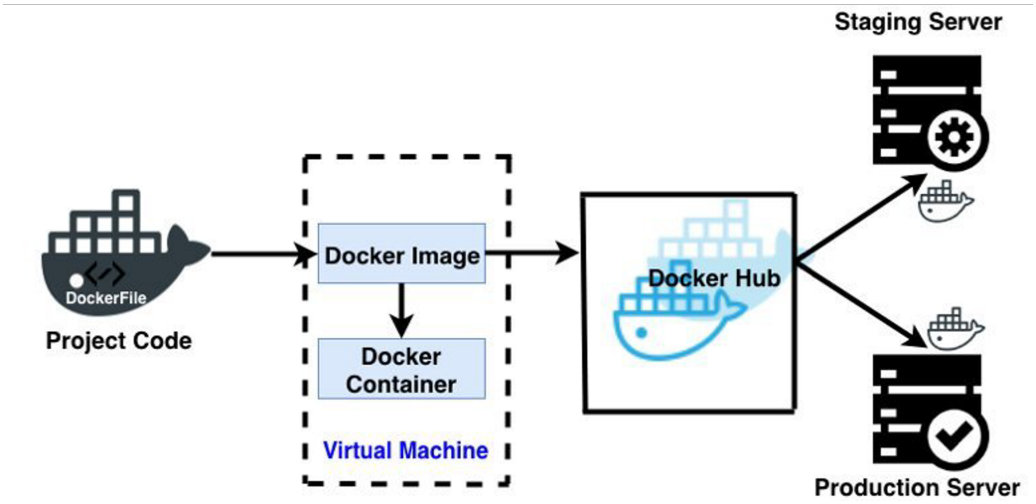
2.Containerisation of the micro-services using Docker
To understand this step, we will have to familiarise with what containers are:

- Running instance of a docker image are called as Containers. Docker images are like 'virtual boxes' which consists of an entire runtime environment: an application plus all its dependencies, libraries and other binaries and configuration files needed to run it, bundled into one package.

Containerisation:

- The process of bundling an application together into a package.
- 'Build once run anywhere' Can be run on any Operating System
- Light-weight alternative to Virtual Machines

Docker Software, which is a Command-line interface will be used to containerise step-1. Containers are registered in the Docker registry from which they can be accessed. Using the 'dockerbuild' command, it can be containerised in the docker daemon. These containers are known as images when they enter Docker registry.

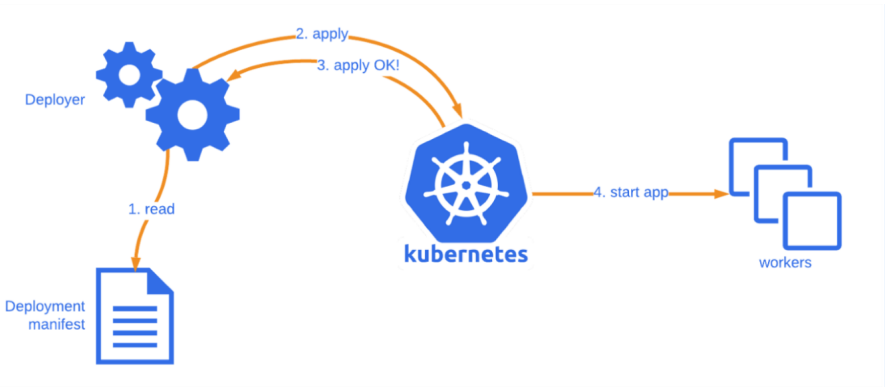


3. Working of an application using Kubernetes(K8s)

In this step, we will be deploying the image(container) from step 2 and managing it using a software called Kubernetes which is a CLI too. The following terms are needed to understand Kubernetes:

- Node: It is a worker machine
- Cluster: A group of nodes
- Pods: They are the atomic units which can contain one or more containers.
- Deployments: They provide the most common way to create and manage pods
- Service: A service declares how pods can be accessed. It is the virtual server inside the node. When pods need to communicate, they use 'services'

A developer reads the deployment manifest which is a set of rules for the deployment of the pod(container). These changes are then sent to K8s(Kubernetes) which reads and send the confirmation to the deployer and simultaneously makes the required changes and deploys the services to the users.



To demonstrate high scalability, Apache JMeter will be used to create a vast number of HTTP requests which sort of creates a fake traffic on the web application which is the pod deployed by K8s. Kubernetes will then scale the resources and show how smoothly the application is running without any crash or glitch.

APPENDIX:

PPT link:
https://drive.google.com/file/d/15M-JMl7QrGtPz4QwwMGA2V2jQtW0knNz/view?usp=sharing

* The above link open a ppt which shows a changed version due to google slides, when downloaded, it comes back to the original uploaded version.

REFERENCES:

1. R. Muddinagiri, S. Ambavane and S. Bayas, "Self-Hosted Kubernetes: Deploying Docker Containers Locally With Minikube," 2019 International Conference on Innovative Trends and Advances in Engineering and Technology (ICITAET), SHEGAON, India, 2019, pp. 239-243, doi: 10.1109/ICITAET47105.2019.9170208. , https://ieeexplore.ieee.org/document/9170208

2. J. Shah and D. Dubaria, "Building Modern Clouds: Using Docker, Kubernetes & Google Cloud Platform," 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2019, pp. 0184-0189, doi: 10.1109/CCWC.2019.8666479., https://ieeexplore.ieee.org/document/8666479