**PROJECT REPORT**

**Abstract**

A journey of understanding and innovation, from English to French using modern technologies. With the help of DynamoDB, Docker, and Kubernetes, the project aims to bridge the gap between the two languages and allow for efficient translation. By creating an efficient and reliable translation infrastructure, this project created a new level of understanding between used technologies in distributed system.

**Introduction**

The English to Fench translator uses a combination of different technologies:

FrontEnd: React

Backend: Python

Persistence: DynamoDB

Containerization and Deployment: Docker + Kubernetes

This project aims the following DSD topics:

Concurrency  
Scalability

Fault Tolerance

**System**

Docker :- Docker is an open-source platform for creating, deploying and managing applications using containers. Containers are isolated, lightweight environments that share the same underlying operating system and resources, but are separate from each other, allowing multiple applications and services to be run simultaneously on the same machine. For example, you can use Docker to deploy a web server, a database server, and an application server all on the same machine.

Uvicorn :- Uvicorn is a lightning-fast ASGI server implementation, which is built on top of the Uvloop and httptools libraries. It is based on the well-established ASGI standard and provides spec-compliant support for the most popular asynchronous frameworks, including Starlette, FastAPI, Responder, and Django ASGI. Example: To start a Uvicorn server, you can use the following command: uvicorn --host 0.0.0.0 --port 8000 myproject.asgi:application

FastApi :- FastAPI is an open-source, modern web framework for building APIs with Python 3.6+ based on standard Python type hints. It is designed to be high-performance and easy to learn. With FastAPI, you can build APIs with automatic data validation, serialization, authentication and authorization, interactive documentation, and more. For example, you can use FastAPI to develop a RESTful API that takes in user input data and returns a response based on that input. You can define the input and output data types, specify the parameters that the API should accept, and define the authentication and authorization levels. The API will automatically validate the user input and return a response based on that data.

Text

Description automatically generated

Above image is of dockerFile which is responsible to create container and image

* FROM in a Dockerfile is used to specify the base image used to build the Docker image. This is the first instruction in a Dockerfile and is required. The FROM instruction specifies the base image to use when building the image.
* WORKDIR is a Dockerfile instruction that sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile. It can be used multiple times in a Dockerfile to switch to a different directory.
* COPY in a Dockerfile is a command used to copy files from the host machine into the Docker image. It is used to bring in files, scripts, applications, and other items into the Docker image.
* In a Dockerfile, the RUN command is used to execute commands in a new layer on top of the existing image. This is often used to install software packages and run other system operations.
* CMD is a command in a Dockerfile that specifies which command should be run when a container is launched from the image created by the Dockerfile.

**Commands to create containers**

docker build -t image . :- This command is used to build a Docker image called "image" from the current directory (indicated by the '.'). The "-t" flag is used to specify a tag for the image, which can be used to refer to the image when running it.

docker run -it --name container -p 80:80 image :- This command will create a Docker container called "container" from the specified image, and map port 80 of the host to port 80 of the container. The "-it" flag launches the container interactively, so you can execute commands within the container.

**Api creation Code**

**Text

Description automatically generated**

**Kubernetes**

Using .yaml files, you can manage deployments as code. It orchestrates all the components based on ruled and event. In conjunction with Metrics server, it monitors PODS cpu usage and memory.

We created the following components:

**HPA:** Stands for Horizontal Pod Autoscaling. If average 70% POD utilization is reached, scale and create other POD with 1 container. Min 1, max 10.

**Load balancer:** Distributes load amongst containers using Round-Robin scheduling.

Deployment: One container per pod.

---

apiVersion: v1

kind: Service

metadata:

name: mycoolloadbalancer

spec:

ports:

- name: 80-tcp

port: 80

protocol: TCP

targetPort: 80

selector:

run: backend-deployment

type: LoadBalancer

status:

loadBalancer: {}

---

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: backend-deployment

labels:

run: backend-deployment

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: backend-deployment

minReplicas: 1

maxReplicas: 10

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 70

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: backend-deployment

labels:

run: backend-deployment

spec:

replicas: 1

selector:

matchLabels:

run: backend-deployment

template:

metadata:

labels:

run: backend-deployment

spec:

containers:

- name: backend-deployment

image: thebackendcontainer

ports:

- containerPort: 80

protocol: TCP

resources:

limits:

cpu: 500m

requests:

cpu: 200m

restartPolicy: Always

Commands used

kubectl apply -f backend-deployment.yaml.:- it is used to create or update resources in a Kubernetes cluster. Specifically, the command will apply the configuration in the backend-deployment.yaml file to the cluster. This could include creating new deployments, services, and other objects in the cluster.

kubectl describe deployment backend-deployment. :- It details about a Kubernetes deployment in a distributed system. It will provide information such as the deployment's name, labels, replicas, selector, and strategy as well as the pod and container templates it uses. It will also provide information about the deployment's status, including the number of replicas currently running, the number of desired replicas, and any events related to the deployment.

kubectl get pods :- It is a command used to list the pods in a Kubernetes cluster. It lists all the pods in the cluster along with their state (e.g. Running, Pending, etc.), IP address, node, and other details. This command is used to monitor the status of the pods in a distributed system and ensure that they are healthy and running.

Kubectl get svc is a command used in a distributed system to get a list of all the services available in a Kubernetes cluster. It will return a list of all the services, their IP addresses, ports, and labels associated with them. This can be used to troubleshoot and debug issues in the system as well as to get a general overview of what services are running in the cluster.

Kubectl get hpa is a command used to retrieve information about Horizontal Pod Autoscalers (HPAs) in a distributed system. It provides information about the number of replicas that have been created, the current target utilization, and any other associated metrics. This command is useful for monitoring the performance of the autoscaler and ensuring that it is scaling resources as needed.

kubectl run -i --tty load-generator --rm --image=busybox:1.28 --restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done". This command will create a deployment called load-generator, using the busybox:1.28 image, that will never restart and will send a constant stream of HTTP requests to the php-apache server.

The command 'kubectl get hpa backend-deployment --watch' will list Horizontal Pod Autoscaler (HPA) information for the deployment 'backend-deployment' and continuously watch for changes in that information.

**Load balancer**

A load balancer is a device that distributes workloads to multiple computing resources, such as servers, in order to optimize the performance of an application or service. It helps to ensure high availability of applications by distributing incoming traffic across multiple resources. This helps to prevent any single resource from becoming overloaded and reduces the risk of service disruption.

Examples of load balancers include hardware load balancers, software load balancers, application delivery controllers, and cloud load balancers.

FrontEnd

We used React for creating the front end of our project. The motive of this project was to learn the concept of DSD, that why we kept the front-end part of our project as simple as possible. We have a search bar on our home page that calls our backend API with respected English phrases. In return, we get the french version of that phrase below the search bar. The technology used: React hooks, Material UI and API.

**Demo scenario**

1. Run backend and frontend container
2. Send POST requests to Load Balancer and hit /api/get with the following script kubectl run -i --tty load-generator --rm --image=busybox:1.28 --restart=Never -- /bin/sh -c "while sleep 0.0001; do wget -q -O- http://10.103.198.90; done". This will create an infinite loop of requests and will increase the CPU utilization.
3. This will make backend to query the Amazon Dynamo DB and respond with the translated text.
4. Expect replicas to increase with the following formula

Replicas = podutilization/threshold

E.g. 421/70 = 6 replicas

The following command output demonstrates the successful result.

NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE

backend-deployment Deployment/backend-deployment 33051648/40Mi, 0%/5% 1 10 4 2d7h

**Conclusion**

Kubernetes + Metrics server are great resources to manage docker containers, deploy and manage services and capacity minimizing the risks of downtimes, reduction in quality of service and improve customer satisfaction, response times and revenue of businesses.