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**

key points of using dynamoDB.

1. Scalability: DynamoDB is designed for scalability and offers seamless scaling for your applications. It automatically increases or decreases the throughput capacity of your tables, so you don’t have to worry about provisioning or de-provisioning capacity.

2. Performance: DynamoDB offers consistent, single-digit millisecond latency for read and write operations. With DynamoDB, you can access data quickly, knowing that your requests will always be served within the specified latency.

3. Security: DynamoDB has built-in security features, such as encryption at rest and encryption in transit, that ensure data security and compliance. It also supports fine-grained access control, so you can control who has access to which items in your database.

4. Flexibility: DynamoDB allows you to store and retrieve data in any format including JSON, HTML, XML, and text. This makes it easy to integrate DynamoDB with existing applications and services.

5. Cost-effectiveness: DynamoDB is cost-effective because it offers a pay-as-you-go pricing model. This means that you only pay for the resources you use, so you can save money compared to traditional on-premises database solutions.

key points of using containers for scaling in a distributed systems.

1. Improved Efficiency: Containers are much more efficient than traditional virtualization, as they make use of the host kernel and don’t require a full operating system. This improves utilization and reduces overhead.

2. Enhanced Portability: Containers allow applications to be easily moved from one system to another. This makes it easy to scale applications across multiple systems.

3. Increased Resource Utilization: Containers enable the efficient use of system resources. This helps to reduce costs and improve performance.

4. Improved Isolation: Containers provide an isolated environment for running applications. This helps to ensure that applications can be safely deployed without interfering with other applications or the host system.

5. Automated Deployment: Containers make it easy to automate deployment and scaling of applications. This helps to streamline development and deployment

6. Flexible Deployment: Containers make it easy to deploy applications in multiple different environments, such as on-premises, in the cloud, or in hybrid environments.

**key points of using Kubernetes.**

**1. Automated Scheduling and Orchestration: Kubernetes automates the deployment, scaling, and management of containerized applications. It can even self-heal and replace failed containers.**

**2. High Availability: Kubernetes can replicate your application across multiple nodes for high availability.**

**3. Infrastructure Agnostic: Kubernetes can be deployed on any cloud or on-premise infrastructure.**

**4. Service Discovery and Load Balancing: Kubernetes can automatically discover and load balance services across nodes.**

**5. Storage Orchestration: Kubernetes can manage the storage needs of your applications.**

**6. Security: Kubernetes offers built-in security features to protect your applications and data.**

**7. Easy Deployment: Kubernetes provides an easy way to deploy and manage applications in a distributed environment.**

**8. Cost Savings: Using Kubernetes can help reduce costs associated with cloud infrastructure and operational costs.**

**key points of using matrix in Kubernetes.**

**1. Automate deployment and scaling of applications: Matrix allows you to quickly and easily deploy and scale applications and services on Kubernetes.**

**2. Simplify application management: Matrix simplifies the management of application deployments through the use of labels and custom resources.**

**3. Monitor the health of applications: Matrix provides a powerful way to monitor the health of applications, including their resource utilization and overall performance.**

**4. Increase efficiency: Matrix helps to reduce the manual effort required to manage and scale applications.**

**5. Reduce complexity: Matrix reduces the complexity of managing multiple applications and services on Kubernetes by providing a unified approach to deploying and managing them.**

**advantages of using react in distributed.**

**1. Code Reusability: React allows developers to use and reuse components across the application. This makes the code much cleaner and easier to maintain.**

**2. Flexible and Scalable: React is a highly flexible and scalable library that can easily be used to build complex applications.**

**3. Easy to Learn and Use: React is easy to learn and use, especially if you already know JavaScript.**

**4. High Performance: React is extremely fast and efficient, making it ideal for distributed applications.**

**5. Easy to Debug: React has a great debugging and error-finding tool, which makes it easy to find and fix errors.**

**6. Testable: React makes it easy to write unit tests, which makes it easier to maintain and debug applications.**

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.

Key Points:

- Distributes workloads to multiple computing resources

- Optimizes application or service performance

- Ensures high availability of applications

- Reduces risk of service disruption

- Can be hardware, software, application delivery controllers, or cloud load balancers

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.