
# Containerisation of Web Application & Deploying in Hybrid Cloud

HIS - Cloud Computing, Summer Semester 2022

**Prepared By**
Mohammad Sayedur Rahman
Saiful islam
Shrabanti Saha Rimi,
Ashis Banik
Ta-Seen Junaid

**Under the guidance of**
Prof. Dr. Christian Baun
Frankfurt University of Applied Sciences

# Table of content

# Cloud Computing

The concept of “Cloud” came from the metaphor for the Internet. Cloud Computing is the transmission of different computing services such as servers, networking, databases, applications, and programs etc. over the internet.

## Types of Cloud

Cloud computing has two types of models. Deployment models and Service models. Deployment model consists of Public Cloud, Private Cloud, Multi Cloud and Hybrid Cloud. And there are three types of service models, Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Different deployment models and service models are explained below.

### Deployment Models

**Public Cloud:** The cloud infrastructure of a public cloud is accessible to the general public via the internet. Public cloud owned and operated by different 3rd party cloud providers like Microsoft Azure, Google Cloud etc. who distribute different computer services like storage and servers over the internet [1].

**Private Cloud:** In a private cloud, the cloud infrastructure is owned and operated by an organization, though it can be managed by the organization or a third party. A company’s onsite data center is an example of a private cloud [1].

**Multi-cloud:** In multi-cloud, multiple cloud computing and storage services from different service providers can be integrated into a single architecture. It reduces dependency on a single service provider and uses any service as desired.

**Hybrid Cloud:** Hybrid cloud is the combination of public and private cloud that shares data and applications between them. Hybrid cloud architecture consists of Infrastructure-as-a-Service (IaaS) platform. The common IaaS platforms are Amazon Web Services, Microsoft Azure and Google Cloud platform. In hybrid cloud, resources can be stored on premises or off premises. A wide area network (WAN) is needed for hybrid cloud administration to connect public cloud and private cloud [2]. As data can be moved between private and public cloud it brings flexibility, portability, optimization, and security in the organization.

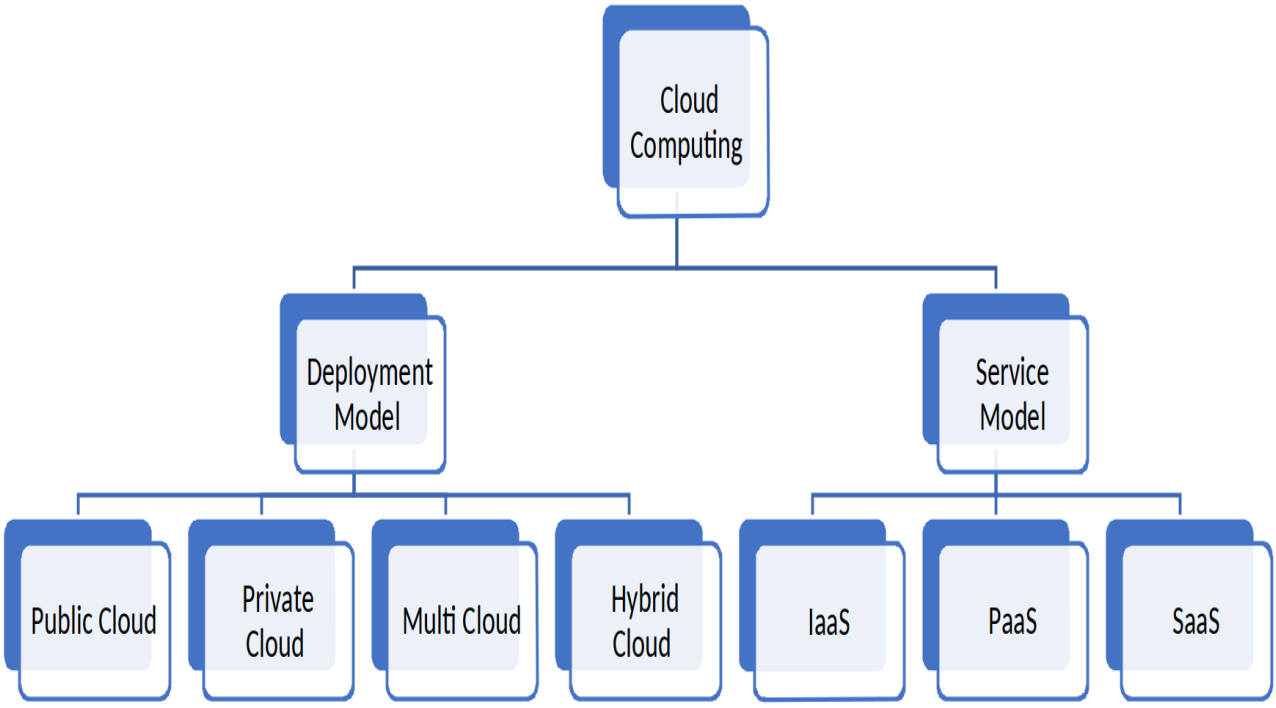


Figure: Types of Cloud Computing

## Service Models

**Infrastructure-as-a-Service (IaaS):** IaaS is a cloud service model where users can access basic computing infrastructure. It is commonly used by IT administrators. An organization rents IT infrastructure such as servers, virtual machines, storage, networks etc. from the cloud providers on a pay-as-you-go basis. Shifting an organization to an IaaS system reduces maintenance of on-premise data centers, reduces hardware costs [1]. Amazon Web Services (AWS), Google Compute Engine (GCE), IBM Cloud, and Microsoft Azure are some examples of IaaS.

**Platform as a Service (PaaS):** PaaS provides platforms and runtime environments for developing, managing and testing applications. The goal of PaaS is to free developers from having to set up or manage the servers, storage, networks, and databases required for development in order to swiftly create online or mobile apps. A web application for scheduling appointments operated via Google App Engine could be an example of PaaS.

**Software as a Service (SaaS):** SaaS uses cloud services to host and administer software programs. Vendors provide the needs for both software and hardware, so the company doesn't have to consider any of these solutions. The SaaS approach suits a firm that doesn't want to be

burdened by any IT infrastructure. SaaS cloud service providers handle all of the organization's requirements for solutions. Google Workspace (formerly GSuite), Dropbox, Salesforce, Cisco WebEx, SAP Concur are some of the examples of SaaS.

## Advantages of Cloud Computing

**Cost Effective:** As cloud computing services run by pay per you go method, the organization that uses the cloud services only pays as much as they use, such as data storage use. This exclusive feature has a positive outcome in increasing revenue for the organization.

**Security:** Although most firms prefer not to discuss the prospect of internal data theft out loud, the reality is that an astonishingly high percentage of data thefts are carried out inside by employees. When this happens, keeping critical information offsite can actually be much safer. According to RapidScale, 94 percent of firms reported improved security as a result of moving to the cloud, and 91 percent said the cloud made it simpler to comply with regulatory requirements. The encryption of data being communicated over networks and kept in databases is the key to this increased security. As with the majority of cloud-based services, various security settings can be adjusted based on the user as an additional security protection [3].

**Flexibility:** In general, using the cloud gives businesses more flexibility than hosting on a local server. Additionally, a cloud-based solution may rapidly match an organization's demand for additional bandwidth rather than requiring a complicated (and costly) update to its IT infrastructure. The overall effectiveness of business could be significantly boosted by this increased freedom and flexibility [3].

**Insight:** For a bird's-eye perspective of data, several cloud-based storage options provide integrated cloud analysis. Cloud Service users can quickly deploy tracking systems and create custom reports to examine information across the entire organization when the data is stored in the cloud. Users can develop strategies to accomplish organization objectives and boost efficiencies as a result of those findings. For instance, Sunny Delight, a beverage firm, used cloud-based business analytics to boost earnings by around $2 million annually while reducing employee costs by $195,000 [3].

# Hybrid Cloud

Hybrid cloud is the combination of private and public cloud which refers to a mixed computing, storage, and services environment made up of on-premises infrastructure, private cloud services, and public cloud services.

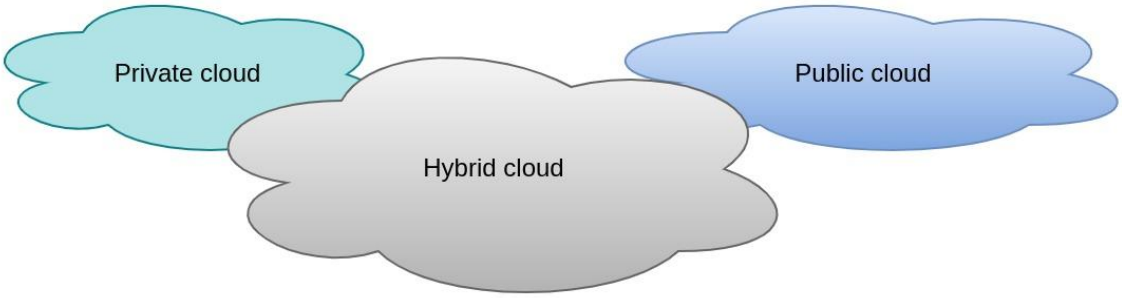


Figure: Hybrid cloud

# Why Hybrid Cloud

## The Perfect World

In hybrid cloud model, if one cloud platform is unavailable to provide services still all our services will be run properly due to the fact that all the services are still available through the other cloud platform.

## Taking the Best from both World

Choosing the hybrid cloud approach we can take the advantages of public cloud, private cloud, on-premises infrastructure and multi cloud where we also can avoid the disadvantages of each world.

## More Control over Costs

With hybrid cloud computing, you are not totally dependent on another company and it will give you the opportunity about how much control you want in your hand and how much control you want to give the cloud provider.

## Separating Critical Workloads from Less Sensitive Workloads

You might run critical services like store sensitive financial or customer information on your private cloud while using a public cloud to run the rest of your enterprise applications.

## Modernization and Migration at your own pace

With a hybrid cloud you can modernize and migrate applications to the different clouds or on-premises at the pace which will be most effective and efficient for your business and transform your technical infrastructure over time.

### Maintain Regulatory Compliance

Many industries and organizations have rules surrounding where and how your app can operate. Hybrid systems can help us with the consistent adoption wherever workloads are deployed and managed.

### Flexibility for the Future

The future is uncertain and dynamic and to match the needs of the future hybrid system can ensure flexibility which will be most effective and efficient for your business.

# Containerization

Containerization is the encapsulation of software code with its run time environment that packages application code as a single, portable, lightweight executable package together with related operating system (OS), libraries, configuration files, dependencies etc.

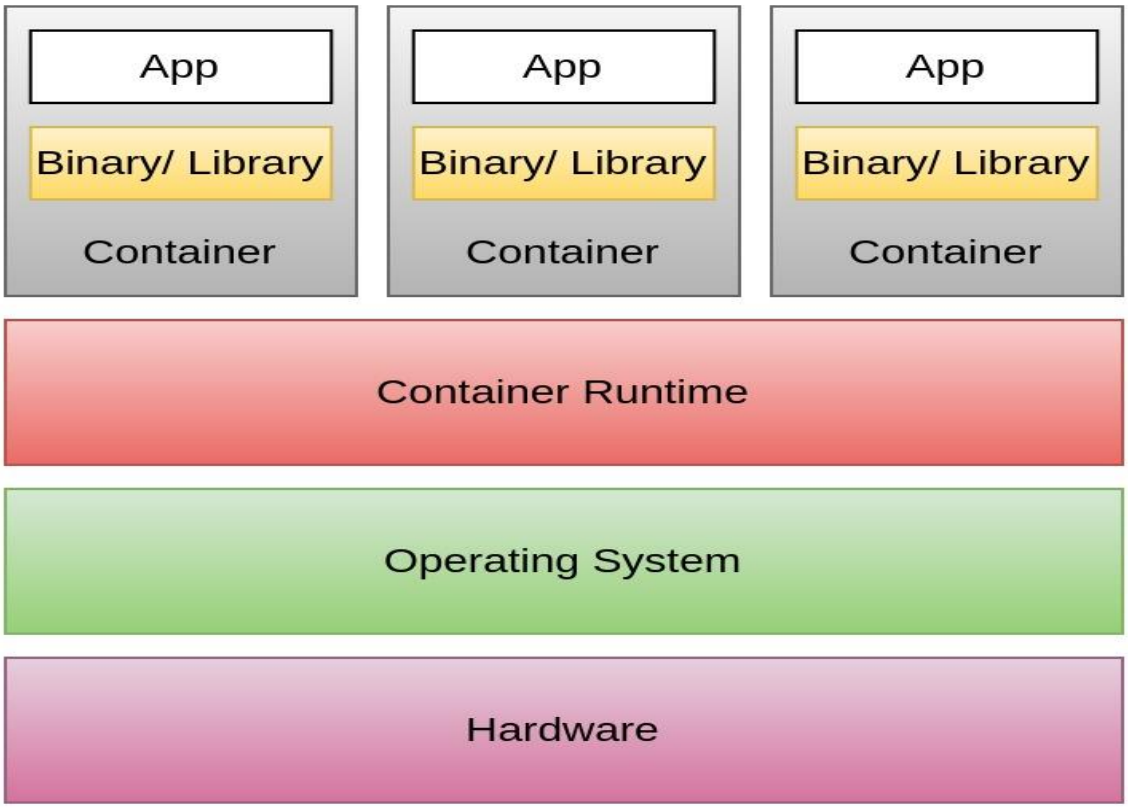


Figure: Containerization techniques

The single executable file created by containers runs upon container runtime which is installed on the host operating system. Though virtual machines also have resource isolation and allocation benefits, they need hardware level and OS kernel level virtualization where containers

virtualize the operating system only. That is why containers are called lightweight as they share the machine's OS kernel for different containers. Moreover, common container layers can be shared among each other.  All of those make containers smaller in size with a faster start up and less overhead.

## Benefits of Containerization

- Portability
- Agility
- Speed
- Scalability
- Dev and Ops separation of concerns
- Developer-Friendly
- Continuity
- Environmental consistency across development, testing, and production
- Ease of Management
- Plug and play
- Security
- Continuous development, integration, and deployment
- Fault Isolation
- Loosely coupled, distributed, elastic, liberated micro-services
- Efficiency
- Resource utilization
- Lightweight
- Resource isolation

# Container Orchestration

Container orchestration is the automation of managing and coordinating the life cycles of containers in dynamic environments.

A containerized based microservices application might translate into operating hundreds or thousands of containers which can introduce significant complexity if managed manually. Container orchestration automates the provisioning, coordinating, deployment, scheduling, networking, configuring, scaling, availability, and lifecycle management of containers.

## Benefits of Container Orchestration

- Load balancing
- Traffic routing
- Service discovery of containers
- Storage orchestration
- Automated rollouts and rollbacks

- Automatic bin packing
- Disaster recovery or backup and restore
- Keeping interactions between containers secure
- Self-healing
- High availability or no downtime
- Provisioning of containers
- Deployments of containers
- Secret management
- Configuration management
- Scalability or high performance
- Monitoring container health
- Resource allocation
- Configuring and scheduling of containers

# Technology Used

## Docker

Docker is a compartment virtualization innovation. Docker and Docker holders consider packaging an application with its circumstances and execution environment into a standardized, deployable unit, ensuring that the application performs constantly and dependably on the different enrolling stages [4].Thus, it resembles an exceptionally lightweight virtual machine [VM]. Docker's natural framework contains various parts, including a docker client to allow the client to interface with a running docker daemon. As well as building holders, we give what we call a designer work process, which is truly about assisting individuals with building compartments and applications inside compartments and afterward dividing those between their colleagues [5]. The docker daemon runs a holder from a close-by image or pulls an image clearly from the vault.

### Docker File

A Docker document is a text report that contains all of the orders (rules) a client could move toward the request line to assemble an image. Docker Instructions are determined inside a Docker file and are utilized by Docker for consequently fabricating an image. Using Docker gather, clients can make a motorized structure that executes a couple of request line rules in movement [4]. Typically, they demonstrate how a given venture is based on a base image. Docker records normally contain bits of knowledge concerning the base image, biological factors, comments, and orders to execute shell orders, present circumstances, present programming like orchestrating and interfacing, open/uncover ports for outside access, and start the association [4]. A Docker file can contain every one of the orders a client could approach the

order line to gather an image. For example, RUN direction to execute any requests or COPY direction to copy records or lists to the compartment's filesystem at the way true region [4].

### Docker Hub

A Docker Hub is a Docker vault for putting away docker images. Docker image makes a docker holder. Compartments hold the entire pack expected for an application, so the application can be run in a disengaged manner [1]. The images are put away in Docker Hub in storehouses, each containing various renditions of a comparative image. The image shows are put away as a JSON record, while the layers are put away in a packed organization. For instance, assume there is an image of Ubuntu OS with SQL SERVER, when this image is run with the docker run order, then, at that point, a compartment will be made and SQL SERVER will be running on Ubuntu OS [6]. Clients can transfer, search or download images. The images given by Docker Inc or accomplices are called official images and have the sole name of the structure, while the client transferred images are contained in the store in the arrangement.

### Cloud-based Development

Cloud-based Development class is connected to using Docker to motorize the most well-known approach to setting the item improvement environment on the server-side related to the cloud-based structure and various organizations like an informational index [7]. The parts outlining an application are usually conveyed on cloud stages by relying upon virtualization developments [4].

Compartment-based virtualization is getting progressively more energy in this present circumstance, as it gives an isolated and lightweight virtual runtime environment. Even more expressly Docker is used in the setting up of the workspace that is ready to code where all of the circumstances expected by the source code are collected, running from gadgets and robotized testing, and live to the portion code [7]. Docker contains the genuine standard for compartment based virtualization, and it awards packaging programming parts in Docker images, which are then exploited as examined only designs to make and run Docker holders. Compartment orchestrators are then used to robotize the association and the leading group of containerized applications at an immense extension.

## Kubernetes

Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management which was originally designed and developed by Google and is maintained by Cloud Native Computing. Kubernetes describes a lot of building blocks that overall give parts that send, stay aware of, and scale applications considering CPU, memory, or custom estimations. Kubernetes are estimated coupled and extensible to meet different obligations. The Kubernetes configuration coordinates the possibility of a case, a consideration that adds up to a lot of compartments for specific normal resources at a comparative host machine [8]. It plays a basic figure in the overall show of Kubernetes. Cloud systems enable computational resources to be obtained on demand and according to application essentials. The inward parts, as well as expansions and holders that unexpected

spike sought after for Kubernetes, rely upon the Kubernetes API. Clients can rent computational resources of different sorts: virtual machines (VMs), holders, master gear, or uncovered metal resources, each having its own characteristics and cost. The stage applies its control over the cycle and limits resources by portraying resources as Objects, which can then be directed in this manner [9]. Thusly, understanding the presentation related to sending, finishing, and keeping a holder that has that ability is basic, as it impacts the limit of a provider to offer better-grained blaming decisions for clients of stream assessment or taking care of use essentials.

# AWS

AWS (Amazon Web Services) is an extensive, evolving cloud computing platform which is offered by Amazon. It combines infrastructure as a service (IaaS), platform as a service (PaaS), and packaged software as a service (SaaS) . An enterprise may benefit from AWS services by receiving tools like computing power, database storage, and content delivery services.

Amazon.com developed AWS in 2006 from the internal infrastructure to manage its online retail activities. Pay-as-you-go cloud computing model that scales to provide users with compute, storage or throughput as required was built by AWS which was one of the first companies to offer it.

AWS is divided into various services, each of which can be customized based on the requirements of the user. An AWS service's configuration options and individual server mappings should be visible to users.
The portfolio of Amazon Web Services includes more than 100 services, such as those for computation, databases, managing infrastructure, developing applications, and security. these services consist of:

- Compute
- Storage databases
- Data management
- Hybrid cloud
- Networking
- Analytics
- AI
- Messages and notifications
- Security

# AWS EKS

Anyone can easily operate Kubernetes on AWS and on-premises with the help of Amazon Elastic Kubernetes Service (Amazon EKS), a managed Kubernetes service. An open-source platform called Kubernetes automates the administration, scalability, and management of containerized applications. Applications running on upstream Kubernetes are compatible with Amazon EKS because it has been recognized as Kubernetes-conformant.

The availability and scalability of the Kubernetes control plane nodes responsible for scheduling containers, monitoring application availability, storing cluster data, and other crucial functions is automatically managed by Amazon EKS.

You can use Amazon EKS to execute your Kubernetes apps on both AWS Fargate and Amazon Elastic Compute Cloud (Amazon EC2). With Amazon EKS, you can benefit from all the performance, scale, reliability, and availability of AWS infrastructure in addition to integrations with AWS networking and security services such as role-based access control (RBAC) integration with AWS Identity and Access Management (IAM) and support for pod networking in AWS Virtual Private Cloud (VPC), application load balancers (ALBs) for load distribution, and so forth.

## Helm

Helm is a package manager for Kubernetes. The set of Kubernetes materials that together specify an application is packaged as charts. These charts can describe a single resource, such as a Redis pod, or a full stack of a web application: HTTP servers, databases and caches. By default, Helm is attached with a repository of organized Kubernetes applications that are brought up in the official charts repository. It's also easy to establish a private chart repository for intestinal usage.

## NATS

NATS is an open-source messaging system. It is a single technology that enables applications to safely communicate across any combination of cloud vendors, on-premise, edge, web and mobile, and devices. It confirms publisher-subscriber, request-reply and messaging queue models. The publisher-subscriber system of NATS consists of publishers publishing messages to NATS subjects. Subscribers constructively listening onto these subjects receive the messages. NATS server called as gnatsd provides for scalability by cutting off subscriptions if there is a timeout in connection to the server. Other features of NATS include clustered mode of servers and an always on dial tone for publisher-subscriber system. NATS streaming is a data streaming service for NATS servers. The distinguishing feature between NATS and NATS streaming is that a NATS streaming server embeds a NATS server. NATS streaming API is used to communicate with the NATS server. Channels are the subjects in NATS Streaming in which clients receive data and producers send data to be put in message logs. NATS Streaming provides additional features such as at least once delivery of message, enhanced message protocol using Google protocol buffers, message persistence that's useful for message replay and durable subscriptions. Unfortunately, it does not support wildcard matching. Durable subscriptions essentially means that if a client were to restart, then the server will start delivery with the earliest message that's unacknowledged by that subscriber.

# System Architecture

Our Hybrid cloud system consist of:

- Remote cluster
    - AWS EKS
- Local cluster
    - On-premises cluster with Kubernetes
- Tunnel connection between remote and local cluster via NATS
- Bi-directional conversation in between two systems running in local and Remote cluster

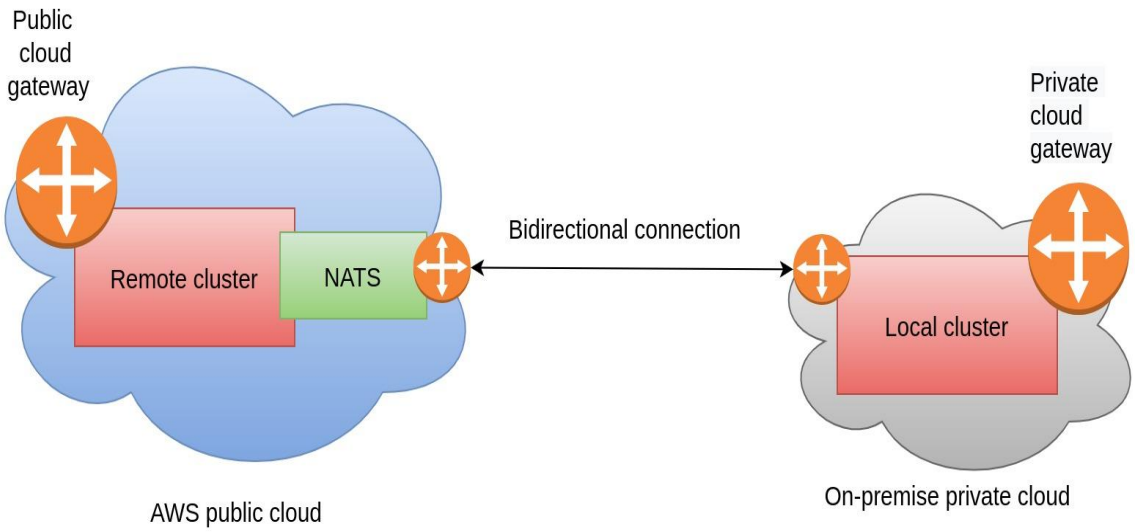


Figure: System Architecture

# Installation Guide

## Prerequisites

Here we are using Ubuntu 20.04.4 LTS (Focal Fossa) OS for operating our system.

### Docker Engine Installation

- Removing the old versions of Docker if they have previously been installed

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

- Install packages that allow APT to use HTTPS

```
sudo apt-get update
sudo apt-get install -y \
```

```
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
```

- Add Docker's GPG key

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
-
```

- Add the Docker repository

```
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
```

- Install the Docker engine

```
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
```

- Add your login to the Docker group

```
sudo usermod -aG docker $USER
```

- Logout of your SSH session and log back in for the changes to take effect.
- Verify that the engine is installed correctly

```
docker run hello-world
```

Output

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs
the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent
```

```
it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

## Docker Compose Installation

Docker compose is a tool to run multi-container Docker applications with the help of YAML files.

- Download a copy of Docker Compose

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.28.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

- Make the app executable

```
sudo chmod +x /usr/local/bin/docker-compose
```

- Test the installation

```
docker-compose --version
```

Output

```
docker-compose version 1.28.4, build unknown
```

## Kubectl Installation

Kubectl is the Kubernetes command-line tool which helps us to deploy and manage Kubernetes clusters [10].

- Download kubectl

```
sudo curl -L "https://storage.googleapis.com/kubernetes-release/release/`curl -s
```

```
https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl" -o /usr/local/bin/kubectl
```

- Make the app executable

```
sudo chmod +x /usr/local/bin/kubectl
```

- Test the installation

```
kubectl version --short --client
```

Output

```
Client Version: v1.20.2
```

## Go Installation

Go is a programming language which is designed at Google [11].

- Download the latest GoLang archive

```
curl -OL https://golang.org/dl/go1.18.3.linux-amd64.tar.gz
```

- Check SHA256 Checksum just in case

```
sha256sum go1.17.3.linux-amd64.tar.gz
```

- Extract everything to the usr/local directory

```
sudo tar -C /usr/local -xvf go1.17.3.linux-amd64.tar.gz
```

- Update PATH variable in ~/.profile file

```
sudo nano ~/.profile
```

- Add new row with export at the end of the ~/.profile file

```
export PATH=$PATH:/usr/local/go/bin
```

- Save the changes and exit the nano editor. Now we have to refresh your profile. Run this command

```
source ~/.profile
```

- Check the version

```
go version
```

Output

```
go version go1.18.3 linux/amd64
```

## Kind Installation

Kind stands for Kubernetes in Docker which is a tool for running local Kubernetes clusters using Docker.

- Download Kind

```
sudo curl -L "https://kind.sigs.k8s.io/dl/v0.14.0/kind-$(uname)-amd64" -o /usr/local/bin/kind
```

- Make the app executable

```
sudo chmod +x /usr/local/bin/kind
```

- Test the installation

```
kind version
```

Output

```
kind v0.14.0 go1.18.2 linux/amd64
```

## AWS Account Creation

The AWS account creation process is given below [12]:

- Open the Amazon Web Services home page
- Choose create an AWS account
- Enter your account information, and then choose Continue
- Choose personal or professional
- Enter your company or personal information
- Read and accept the AWS customer agreement
- Choose create account and continue
- Enter the payment information
- Verify your phone number
- Enter the code displayed in the CAPTCHA, and then submit
- When the automated system contacts you, enter the PIN you receive and then choose continue
- Select a support plan
- Finally, wait for your new account to be activated

To get your access key ID and secret access key [13]:

- Open the IAM console at https://console.aws.amazon.com/iam/

- On the navigation menu, choose Users
- Choose your IAM user name (not the check box)
- Open the Security credentials tab, and then choose Create access key
- To see the new access key, choose Show
- To download the key pair, choose Download .csv file. Store the .csv file with keys in a secure location

## AWS CLI Setup

We will install AWS CLI using APT which is available in the default repository of Ubuntu 20.04 [14].

- Update system packages and repository index to latest

```
sudo apt-get update
```

- Install AWS CLI

```
sudo apt-get install awscli -y
```

- Run the following command to verify the installation

```
aws --version
```

Output

```
aws-cli/1.25.5 Python/3.8.10 Linux/5.4.0-121-generic botocore/1.27.5
```

- Run the following command to configure access to aws account

```
aws configure
```

Enter the following details accordingly:

- AWS Access Key ID [IAM user's Access key]
- AWS Secret Access Key [IAM user's secret key]
- Default region name [Aws region]
- Default output format [JSON format is fine]

## Eksctl Installation

'eksctl' is the official CLI for Amazon EKS [15].

- Download and extract the latest release of eksctl with the following command

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
```

- Move the extracted binary to /usr/local/bin

```
sudo mv /tmp/eksctl /usr/local/bin
```

- Run the following command to verify the installation

```
eksctl version
```

Output

```
0.95.0
```

# Operations Guides

## Docker Build & Push Image to Docker Hub

Below command is used to build docker image & push it to Docker Hub. Here you have to give your docker registry name. **This step is avoidable if you want to use the docker image that is provided by us.**

```
cd chat-app

# docker build -t DOCKER_REGISTRY/chat-app .
docker build -t taseenjunaid/chat-app .

# docker push -t DOCKER_REGEISTRY/chat-app:latest .
docker push taseenjunaid/chat-app:latest
```

Update `deploy-local.yaml` and `deploy-remote.yaml` file with your image (such as `taseenjunaid/chat-app:latest`)

## Prepare Remote Cluster (AWS EKS)

- Create Cluster

```
eksctl create cluster --name remote-cluster
```

- Update cluster config with awscli

```
aws eks update-kubeconfig --region eu-central-1  --name remote-cluster
```

- Install NATS

```
helm repo add nats https://nats-io.github.io/k8s/helm/charts/
helm repo update
helm install my-nats nats/nats
```

- Go to K8s folder from project directory

```
cd K8s
```

- Expose NATS: Create load-balancer type service:

```
kubectl apply -f nats.yaml
```

- Get NATS public address: `LoadBalancer Ingress`

```
kubectl describe services nats-lb
```

Output

```
    Name:                     nats-lb
    Namespace:                default
    Labels:                   <none>
    Annotations:              <none>
    Selector:                 app.kubernetes.io/name=nats
    Type:                     LoadBalancer
    IP Family Policy:         SingleStack
    IP Families:              IPv4
    IP:                       10.100.202.198
    IPs:                      10.100.202.198
    LoadBalancer Ingress:
a7d62f596d6eb4e5dbc5e6b87ac63192-1340131761.eu-central-1.elb.amazonaws.com
    Port:                     nats  4222/TCP
    TargetPort:               4222/TCP
    NodePort:                 nats  31083/TCP
    Endpoints:                192.168.77.234:4222
    Port:                     leafnodes  7422/TCP
```

Here, it is `a7d62f596d6eb4e5dbc5e6b87ac63192-1340131761.eu-central-1.elb.amazonaws.com`

- Update `deploy-remote.yaml`: Set `NAT_URL` env to the latest one which we get from NATS public address, `LoadBalancer Ingress`.

- Deploy Chat-App to remote cluster

```
kubectl apply -f deploy-remote.yaml
```

- Describe the `chat-app-remote` service to get public address: `LoadBalancer Ingress`

```
kubectl describe services chat-app-remote
```

Output

```
    Name:                     chat-app-remote
    Namespace:                default
    Labels:                   <none>
    Annotations:              <none>
    Selector:                 app=chat-app-remote
    Type:                     LoadBalancer
    IP Family Policy:         SingleStack
    IP Families:              IPv4
    IP:                       10.100.133.76
    IPs:                      10.100.133.76
    LoadBalancer Ingress:
a30ebc0f526294f889093f141841c3b6-1399523946.eu-central-1.elb.amazonaws.com
    Port:                     <unset>  8000/TCP
    TargetPort:               8000/TCP
```

- Go to the public address, `LoadBalancer Ingress` link with port. Here it is: `a30ebc0f526294f889093f141841c3b6-1399523946.eu-central-1.elb.amazonaws.com:8000` link. Make sure to mention the port `8000`.
- To get the pods, services and statefulsets

```
kubectl get all
```

## Prepare Local Cluster

- Go to the K8s folder from the project directory if you are not there.

```
cd K8s
```

- Create kind cluster with given config file (i.e. `kind-config.yaml`).

```
kind create cluster --config=kind-config.yaml
```

- Update `deploy-local.yaml`, Set `NAT_URL` env to the latest one which we get from NATS public address, `LoadBalancer Ingress`.
- Deploy the Chat-App to local cluster

```
kubectl apply -f deploy-local.yaml
```

- To get the pods, services and statefulsets

```
kubectl get all
```

- Now your app is accessible at localhost:30000 url.

### Delete Clusters

- To get all contexts

```
kubectl config get-contexts
```

- To go to remote context, here you have to put your remote cluster context name.

```
kubectl config use-context <remote cluster context name>
```

- To delete remote cluster from remote context

```
eksctl delete cluster --name=remote-cluster
```

- To go to local context

```
kubectl config use-context <local cluster context name>
```

- To delete local cluster from local context

```
kind delete cluster
```

# Demo

By following the operational guide, the setup of remote cluster and local cluster is successful. The pods, services and statefulsets of local cluster and remote cluster are shown in the following figures.

```
NAME                                  READY   STATUS    RESTARTS   AGE
pod/chat-app-remote-55cb5f484b-7c7rj  1/1     Running   0          2m41s
pod/my-nats-0                         3/3     Running   0          4m10s
pod/my-nats-box-54c676d4fc-gmpnz      1/1     Running   0          4m10s

NAME                     TYPE           CLUSTER-IP       EXTERNAL-IP                                                              PORT(S)
                                        AGE
service/chat-app-remote  LoadBalancer   10.100.151.219   a058a29e96d394feaa5339fc5e22cb71-169890637.eu-central-1.elb.amazonaws.com  8000:32691/
TCP                                     2m41s
service/kubernetes       ClusterIP      10.100.0.1       <none>                                                                   443/TCP
                                        17m
service/my-nats          ClusterIP      None             <none>                                                                   4222/TCP,62
22/TCP,8222/TCP,7777/TCP,7422/TCP,7522/TCP   4m11s
service/nats-lb          LoadBalancer   10.100.26.140    a3ce8e5069ac04214904d7866f13b865-1428657662.eu-central-1.elb.amazonaws.com  4222:31892/
TCP,7422:31934/TCP,7522:31794/TCP       3m53s

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/chat-app-remote   1/1     1            1           2m41s
deployment.apps/my-nats-box       1/1     1            1           4m11s

NAME                                        DESIRED   CURRENT   READY   AGE
replicaset.apps/chat-app-remote-55cb5f484b  1         1         1       2m42s
replicaset.apps/my-nats-box-54c676d4fc      1         1         1       4m12s

NAME                         READY   AGE
statefulset.apps/my-nats     1/1     4m12s
```

Figure: Remote cluster

```
NAME                                    READY   STATUS    RESTARTS   AGE
pod/chat-app-local-58fb4c6565-t5pcl     1/1     Running   0          3m7s

NAME                       TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/chat-app-local     NodePort    10.96.43.174    <none>        8000:30000/TCP   3m7s
service/kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP          5m32s

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/chat-app-local     1/1     1            1           3m7s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/chat-app-local-58fb4c6565     1         1         1       3m7s
```

Figure: Local cluster

We developed a hybrid cloud chat application which is accessible from both remote cluster and local cluster. The main purpose of this hybrid cloud chat is to visualize the bidirectional connection between remote cluster and local cluster

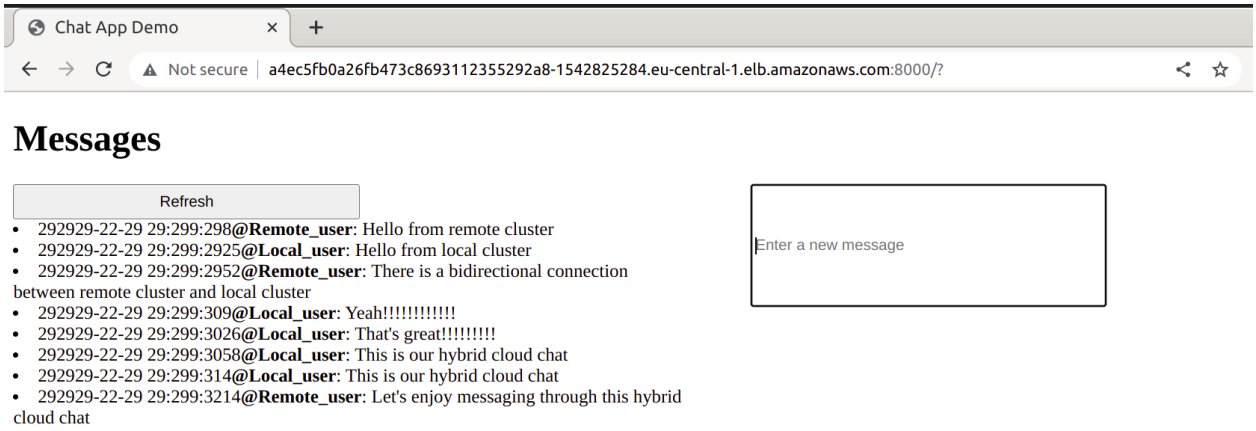


Figure: User interface from remote cluster

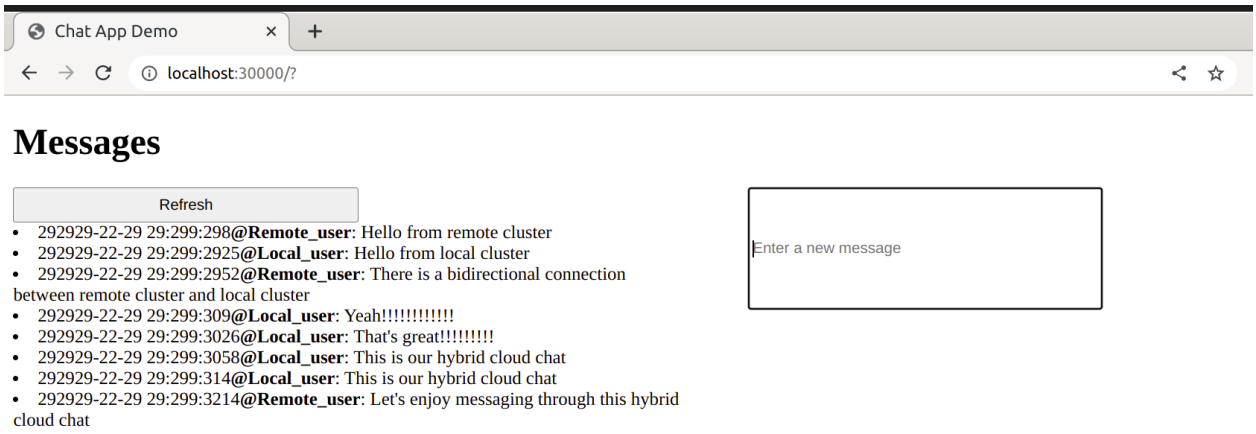


Figure: User interface from local cluster

# Conclusion

It is possible to build a perfect system or near perfect system with the help of a hybrid cloud model. By choosing the hybrid cloud approach we can take the advantages of public cloud,

private cloud, on-premises infrastructure and multi cloud where we also can avoid the disadvantages of each world. On the other hand, containerization technology gives our application perfect agility where container orchestration automates the life cycles of containers in dynamic environments.

# References


[1] *What is cloud computing? A beginner's guide: Microsoft azure*. What Is Cloud Computing? A Beginner's Guide | Microsoft Azure. (n.d.). Retrieved July 1, 2022, from https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-cloud-computing/#use

[2] *Hybrid cloud architecture: Selecting the best of Both worlds*. Cloudian. (2021, February 15). Retrieved July 1, 2022, from https://cloudian.com/guides/hybrid-it/hybrid-cloud-architecture/

[3] *12 benefits of cloud computing and its advantages*. Salesforce.com. (n.d.). Retrieved July 1, 2022, from https://www.salesforce.com/products/platform/best-practices/benefits-of-cloud-computing/

[4] Openja, M., Majidi, F., Khomh, F., Chembakottu, B. and Li, H., 2022. Studying the Practices of Deploying Machine Learning Projects on Docker. arXiv preprint arXiv:2206.00699.

[5] Anderson, C., 2015. Docker [software engineering]. Ieee Software, 32(3), pp.102-c3.

[6] Rad, B.B., Bhatti, H.J. and Ahmadi, M., 2017. An introduction to docker and analysis of its performance. International Journal of Computer Science and Network Security (IJCSNS), 17(3), p.228.

[7] Bogo, M., Soldani, J., Neri, D. and Brogi, A., 2020. Component-aware orchestration of cloud-
based enterprise applications, from TOSCA to Docker and Kubernetes. Software: Practice and Experience, 50(9), pp.1793-1821.

[8] Wikimedia Foundation. (2022, June 28). *Kubernetes*. Wikipedia. Retrieved July 1, 2022, from https://en.wikipedia.org/wiki/Kubernetes

[9] Medel, V., Tolosana-Calasanz, R., Bañares, J.Á., Arronategui, U. and Rana, O.F., 2018. Characterising resource management performance in Kubernetes. Computers & Electrical Engineering, 68, pp.286-297.

[10] *How do I setup kind on ubuntu to run the kubernetes cass-operator?* How do I setup KinD on Ubuntu to run the Kubernetes cass-operator? - Datastax Community. (n.d.). Retrieved July 1, 2022, from

https://community.datastax.com/questions/5369/how-do-i-setup-kind-on-ubuntu-to-run-the-kubernete.html
[11] Yudina, M. (2021, November 6). *How to install go on ubuntu 20.04*. DEV Community. Retrieved July 1, 2022, from https://dev.to/mariayudina/how-to-install-go-on-ubuntu-2004-2mn6

[12] Peterson, K. (2013). *The accounts*. Amazon. Retrieved July 1, 2022, from https://docs.aws.amazon.com/accounts/latest/reference/manage-acct-creating.html

[13] Vanhoo, F. (2021). *PowerShell*. Amazon. Retrieved July 1, 2022, from https://docs.aws.amazon.com/powershell/latest/userguide/pstools-appendix-sign-up.html

[14] Gautam, P. (2021, March 31). *How to install AWS CLI on ubuntu 20.04*. LinOxide. Retrieved July 1, 2022, from https://linoxide.com/how-to-install-aws-cli-on-ubuntu-20-04/

[15] Hansen, N. A. *Eks*. Amazon. Retrieved July 1, 2022, from https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html