Deployment of Flask Application on the Cloud

Researched by: Teo Nickola[1] and Vipul H Harihar[2]

[1]Full Stack Intern — GenTrust, LLC
[2]Lead Software Engineer — GenTrust, LLC

Dated: June 25, 2024

Contents

Deployment of Flask Application on the Cloud

Researched by Teo Nickola and Vipul H Harihar

1 Abstract

This paper details the implementation process of deploying a Flask application to the cloud, specifically focusing on Microsoft Azure. With the advent of cloud services, like Amazon Web Services and Microsoft Azure, it has become highly convenient and cost-efficient to deploy applications to the cloud. By leveraging the power of microservices, a software development approach that breaks large applications into smaller, independent services, this guide aims to help developers ensure their applications are resilient, easily scalable, and maintainable.

2 Introduction

Initially, organizations ran applications on physical servers. When multiple applications run on a single server, one could consume most resources, causing others to underperform. To address this, virtualization was introduced. It enables running multiple Virtual Machines (VMs) on a single server's CPU, optimizing resource utilization, enhancing scalability, facilitating easier application updates, and reducing hardware costs[1](2023).

Containerization is the process of packaging application code along with all its required files and libraries. A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another[2]. Containers function similarly to Virtual Machines, but offer more relaxed isolation properties, allowing them to share the Operating System (OS) among applications. This makes containers lightweight[1]. A container image is a single, executable file used to create a container. The image contains all libraries and dependencies needed for the container to run[2]. There are many ways to containerize an application, but the most common and efficient is through Docker. Docker is an open platform for developing, shipping, and running applications. It allows users to ship, test, and deploy applications in any environment without worrying about compatibility issues, regardless of the machine configuration settings[3].

Flask is a widely used web framework based on the Python programming language, used for designing web applications, APIs, microservices, and more. Flask's framework is simple, flexible, scalable, and reliable, enabling it to process increased traffic. Its ability to handle a wide range of application needs makes it an extremely popular choice for building web-based solutions[4]

[1]Kubernetes
[2]Docker
[3]Azure
[4]Flask

Cloud computing is the delivery of various computing services over the internet, such as databases, servers, software, analytics, and more. It offers faster innovation, flexible resources, and economies of scale. Cloud computing allows companies to optimize IT costs by eliminating the expense of purchasing hardware and equipment to run data centers. Most cloud computing services are on demand and can be provisioned in minutes. A significant component is the scalability of cloud computing, which delivers more resources during high demand and reduces resources when demand is low. Microsoft Azure is Microsoft's cloud computing platform that provides a broad range of cloud services, including computing, analytics, storage, and networking[5]. **Kubernetes** is a flexible, open-source platform designed to manage containerized workloads and services. In a production environment, it's crucial to manage the containers running the applications and ensure continuous uptime. Kubernetes offers a framework to run distributed systems reliably, handling application scaling, failover, deployment patterns, and more[6].

3 Process

3.1 Set Up Azure Account

3.1.1 Create AZ Account

Go to the Azure homepage, select a plan, and create an account, unless one has already been created

3.1.2 Install AZ CLI

Follow the instruction on Azure CLI installation based on the Operating System

3.1.3 Login To Azure

In the terminal:

```
az login
```

3.2 Set Up Azure Resources

3.2.1 Create Resource Group

```
az group create --name ResourceGroupName --location eastus
```

3.2.2 Create Container Registry

```
az acr create --resource-group ResourceGroupName --name ContainerRegistry --sku Basic
```

[5] Azure
[6] Kubernetes

3.2.3 Create Kubernetes Cluster and get Credentials

```
az aks create --resource-group ResourceGroupName --name AKSClusterName --node-
      count 1 --enable-addons monitoring --generate-ssh-keys
az aks get-credentials --resource-group ResourceGroupName --name AKSClusterName
```

3.3 Containerize The Flask Application

3.3.1 Create Docker File

Create a Dockerfile in the root directory of the flask application. This file instructs Docker on how to build the container image. Displayed Below is an example file:

```
#Use an official Python runtime as a parent image
FROM --platform=linux/amd64 python:3.12.3-slim

#Set the working directory in the container
WORKDIR /app

#Copy the environment file
COPY environment.yaml /app

#Install Miniconda3 and create the environment
RUN apt-get update && apt-get install -y --no-install-recommends \
    wget \
    bzip2 \
    && wget --quiet https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-
                                            x86_64.sh -O ~/miniconda3.sh \
    && /bin/bash ~/miniconda3.sh -b -p /opt/conda \
    && rm ~/miniconda3.sh \
    && /opt/conda/bin/conda update -n base -c defaults conda \
    && /opt/conda/bin/conda env create -f /app/environment.yaml \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

#Make sure the environment is activated when running the container
ENV PATH /opt/conda/envs/vipulenv/bin:$PATH

#Copy the rest of the application
COPY . /app

#Make port 80 available to the world outside this container
EXPOSE 80

#Run the Flask app with Gunicorn
CMD ["gunicorn", "-w", "4", "-b", "0.0.0.0:80", "--timeout", "120", "base:app"]
```

3.3.2 Build Docker Image

Open up the terminal and navigate to the directory containing the Docker File. Build the Docker Image by using the command displayed below

```
docker build -t dockerhub-username/image-name .
```

3.3.3 Login to Azure Container Registry

```
az acr login ——name ContainerRegistryName
```

3.3.4 Tag the image as v1 and push it to Azure Container Registry

```
docker tag dockerhub—username/image—name containerregistryname.azurecr.io/image—name:v1
docker push containerregistryname.azurecr.io/image—name:v1
```

3.4 Create Kubernetes Resource

3.4.1 Create a Kubernetes deployment file

Create a new file named deployment.yaml and follow the instructions:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flaskapp-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: flaskapp
  template:
    metadata:
      labels:
        app: flaskapp
    spec:
      containers:
      -name: flaskapp
        image: registry-name.azurecr.io/flaskapp:v1
        ports:
        -containerPort: 80
```

3.4.2 Create a Kubernetes service file

Create a new file named service.yaml and follow the instructions:

```
apiVersion: v1
kind: Service
metadata:
  name: flaskapp
spec:
  type: LoadBalancer
  ports:
  -port: 8080
  selector:
    app: flaskapp
```

3.5 Deploy flask application and service

```
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
```

3.6 Acess the Application

After deploying the service, it might take a few minutes for the external IP to become available. Check the service and get the external IP

```
kubectl get service flaskapp
```

Once the external IP is available, access the flask application in a web browser at `http://<EXTERNAL-IP>`.

A Appendix

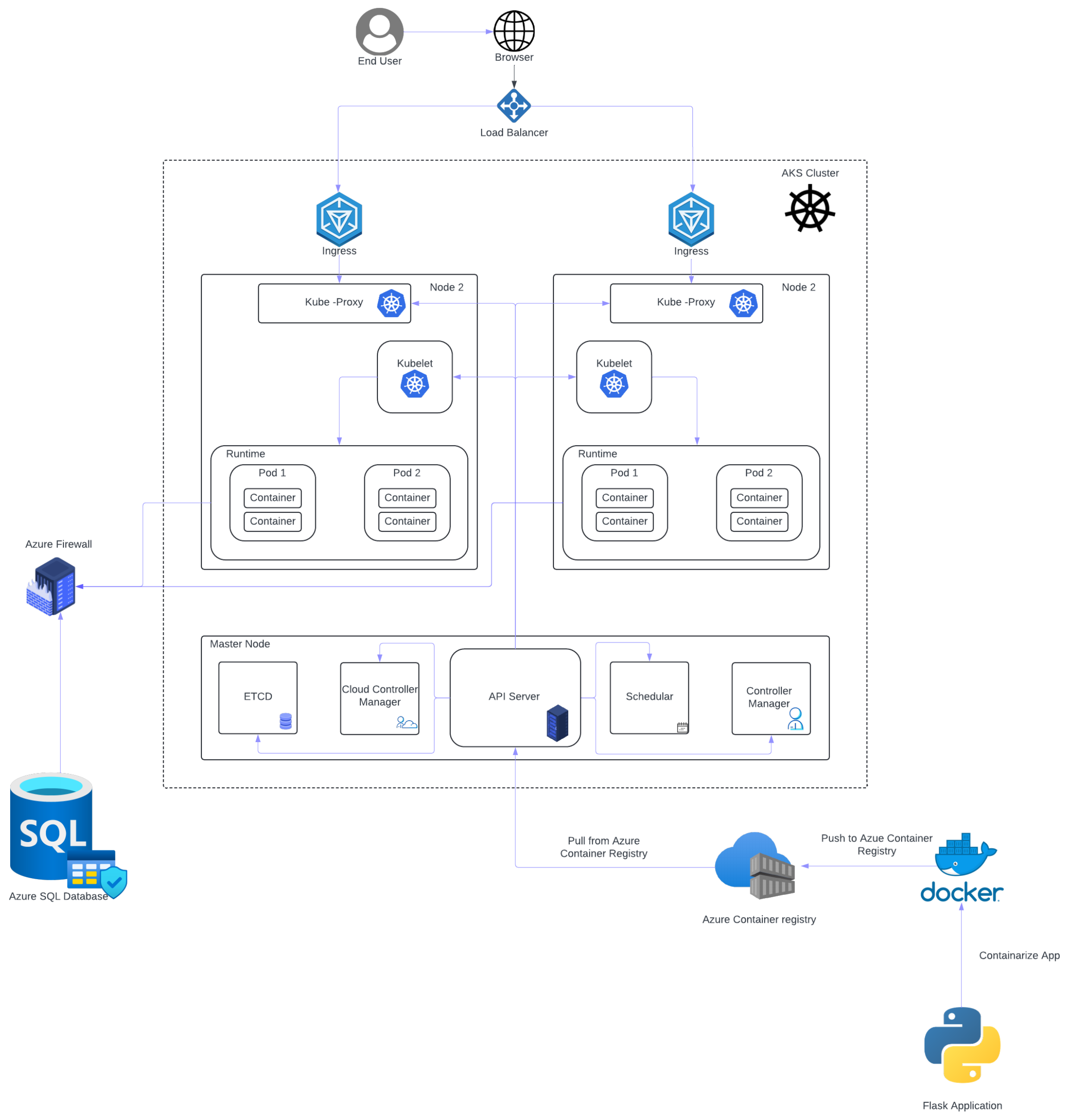

Figure 1: Kubernetes Architecture

References

[1] `https://www.docker.com/resources/what-container/`

[2] `https://www.dimensiona.com/en/what-is-docker-and-what-are-its-advantages/#:~:text=Docker%20is%20a%20container%20platform,for%20it%20to%20run%20independently.`

[3] `https://www.bairesdev.com/blog/what-is-flask/`

[4] `https://azure.microsoft.com/en-in/resources/cloud-computing-dictionary/what-is-cloud-computing#:~:text=Simply%20put%2C%20cloud%20computing%20is,resources%2C%20and%20economies%20of%20scale.`

[5] `https://kubernetes.io/docs/concepts/overview/`