IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, April 2022
Impact Factor: 6.252

The Environment with Docker in the Microservice Ecosystem

Narayan Hari Yadav
Graduate, Information-Technology
S. S. & L. S. Patkar College of Arts & Science & V. P. Varde College of Commerce & Economics, Mumbai
aryany214214@gmail.com

Abstract: *Microservice Applications are playing an important role in Artificial Intelligence, Machine Learning, Big Data, and Data preprocessing. Microservices are made up of several small, self-contained components. Each module (referred to as a service container) is modular, reusable, and self-contained, and communicates with other service containers via language-independent protocols such as hypertext transfer protocol (HTTP) and representational state transfer (REST). It may be a collection of single services or multiple services. This service should be accessed and run on any environment either Windows or Linux, to make it cross-platform we are going to use a docker container. Docker is an open-source container technology that enables the faster distribution of source code, examination of earlier deployments, and setup. In this paper, we present a strategy for microservice applications that use the Docker container service.*

Keywords: Microservice, Docker, Docker-compose, Docker-Orchestration, Containers

I. INTRODUCTION

Nowadays many microservice are running with the tech era, and in this, tech era techies are playing with multiple services every day. These services might be monolithic or microservices in nature. Microservices are small, self-contained services that collaborate. Let's take a closer look at the characteristics that distinguish microservices from other types of software. Independent services are approached in the same way as microservices. Microservices are designed to execute a single task and do not require a large number of contexts. As a result, each microservice can be assigned a specific responsibility. You should be able to easily shut down and spin the images without interfering with other workstreams or tasks. Microservices don't care about the language they're written in. Microservices free you from dependency difficulties by not tying you to a single stack. This is impossible to achieve in a monolithic program. Microservices make it very easy to solve various dependencies in a complex application. Organize everything so that you can run several runtimes and libraries on the same system. They're all separately updated and upgraded components, so you wouldn't have to modify the entire application; only the tasks/items you desire would need to be changed. Our service boundaries are focused on business boundaries, making it clear where code for a given piece of functionality resides. We also avoid the temptation for this service to grow too large, with all the issues that this might bring, by keeping it focused on a defined boundary. The development and testing environment were necessary to create these microservices. We will learn about the Docker environment for microservice development in this paper. You can't run five different applications at the same time.

Docker is a container management service. The keywords of Docker are developed**,** ship, and run anywhere. The whole idea of Docker is for developers to easily develop applications, and ship them into containers which can then be deployed anywhere. The technological principle of Docker isolates applications by packing them with their dependencies, libraries, and configuration files into one consistent image-based environment like an atomic unit, called containers according to [2]. Those container services can be seamlessly moved between hosts or within cloud providers and we combined different approaches to prove effectiveness with equipped. Section II presents techniques and tools for creating state-of-the-art container methodologies for microservice with Docker Container Service. Clarifies the proposed technique for microservice with Container Service. We are going to demonstrate the ASP.net Rest API Microservices containerization on Visual Studio 2019 with installed docker in the working machine.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, April 2022
Impact Factor: 6.252

II. Proposed Solution for Microservices

A lot of od researchers have proposed modern microservices with cloud platforms. Microservice architecture usually requires enthusiastic teams delivering specific business potential. A service container (See Fig. 1) should be designed as an independent product delivering a business capability that is well documented with the container, easy to use, and accountable for one and only one business capability; this is known as the single accountability principle (SAP). A moral rule of thumb is that the microservice containers should be as small as possible but as big as necessary to represent the domain concept or business capability it represents.

2.1 Docker

Docker is an open-source virtualization technology known as a containerization platform for software containers. These containers provide a means of enclosing an application, including its filesystem, into a single, replicable package.

- A Docker image is a read-only template of Docker containers. For example, the Docker image could be full of an Ubuntu operating system with Apache Server and your web application can be installed within the docker image. Docker Images are used to create multiple containers along with specific services.
- A Docker container holds all that is required for an application to run the service container. Each service container is created from the Docker image. Docker containers will be started with container service, stopped with service container, and can be deleted

Docker has three tools for microservices orchestration:

- Docker Machine: You can ask any cloud vendor for a Docker Machine instance. You can provide them a file and say I need this container; they will help you set up a Docker daemon, and you'll be running the command line as if you're somewhere on the cloud.
- Docker Composer: You'd need it if you had one or more containers supporting a single-use case. You may define all of your photos and containers with it. It designates multiple ports or allies through which containers can communicate with one another. You can also execute this on the cloud with docker-machine.
- Docker Swarm: This allows you to orchestrate on a much larger scale.

2.2 Orchestration Layers

Considering the growing number of software device types and the fact that more companies' business strategies are dependent on providing value to customers on several devices and it is no longer optimal to have granular resource-based application programming interfaces (APIs) that closely represent the data model.

2.3 Feature Extraction

- Availability: The golden rule of accessibility states that containers should anticipate failures and design their systems accordingly, ensuring that the container service system is available 99.9 percent of the time. It means that for a whole year, the Docker container service system can only be down for 5.5 minutes. The cloud cluster paradigm is designed to ensure the high availability of container services, and it recommends running the set of services in an Active-Active or Active-Standby configuration. [1].
- Scalability: Microservices must be scale-able both horizontally and vertically with a service container. Being container with horizontal scalable, we can have various instances of the microservice to increase the performance of the container system.
- Usability: When considering the usability of microservice, the design looks at hiding the application's inner design, tech, architecture, and other difficulties to the end-user or another system. The APIs should be considered in a normalized way so that it is easy to achieve the necessary services with a minimal number of API calls.
- Flexibility measures the adaptability to change. Each independent service is owned by different teams and established in agile methodology; modification will happen faster than in any other system. The microservices may not inter-operate if they don't settle in or accommodate the change in other systems.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, April 2022
Impact Factor: 6.252

2.4 Build Container with Docker

In this section, I am going to explain and learn the working of docker orchestration in microservice. That is how we can create a docker image with the help of docker orchestration. For that we are going to demonstrate with Visual; Studio 2019 tools. Here, we have tested our proposed microservice segmentation technique using docker images taken from publicly available sources. The performance of the proposed technique is compared with the monolithic applications and the obtained results for the microservice are evaluated through evaluation metrics namely, availability and accuracy.

2.5 Experimental Issues

The docker experiment for microservice was conducted on the windows 10 machine with intel i5 10th gen, 8GB RAM. I have created a docker container with the help of Visual Studio 2019. The steps are below mentioned.

- Create your project as per your domain requirement.
- Choose Docker as well as during project creation.
- You will get Dockerfile for creating an image.
- Create your microservice and test it.
- If it is working fine, right-click on your project go with add, and select docker-compose.
- Select your target machine either Windows or Linux.
- After that add your docker-compose file, it should generate the docker-compose.yaml file. It will look like the below format.

```
version: '3.4'
services:
  project_name:
    image: ${DOCKER_REGISTRY-} project_name
    build:
      context: .
      dockerfile: Dockerfile
```

Build Your image.

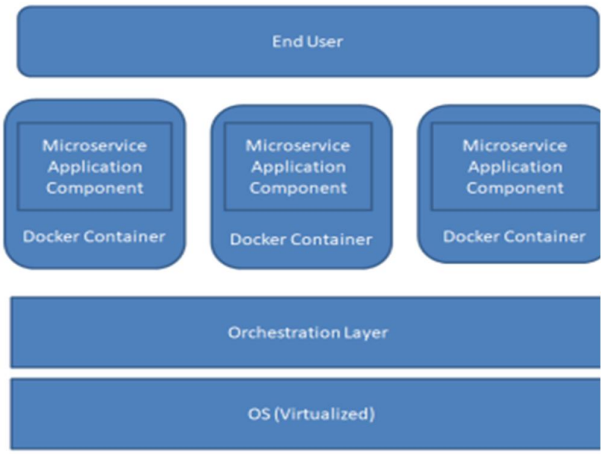

Figure 1: Docker containers with Microservices

2.6 Dockerfile Structure

Dockerfile is a key to building the docker container. Because all the dependence and requirements are bound with the docker file. We are going to learn and watch about ASP.net Rest APIs microservice Dockerfile,

```
FROM mcr.microsoft.com/dotnet/aspnet:3.1 AS base
WORKDIR /app
EXPOSE 80
```

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, April 2022
Impact Factor: 6.252

```
EXPOSE 443
FROM mcr.microsoft.com/dotnet/sdk:3.1 AS build
WORKDIR /src
COPY ["project.csproj", "."]
RUN dotnet restore "./project.csproj"
COPY.
WORKDIR "/src/."
RUN dotnet build "project.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "project.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "project.dll"]
```

Above is the structure of the docker file let's understand it step by step, the first line starts with FROM word which is the location of the image or name of the image. Others are explained below.

- WORKDIR: working directory of the current project
- COPY: copy all the code from one location to another location.

RUN: keyword for running any command in the docker file.

2.7 Study on Microservices

Scaling and maintaining a monolithic app is difficult and unreliable. Agile development and timely delivery are impossible due to outdated applications. Monolithic programs also make it harder to transition to new frameworks and languages. All modern-day digital architectures are supposed to be service-enabled through HTTP web services and RESTful APIs, Microservice architecture is a very specific variation of a service-enabled architecture that is more focused on agility. It can provide significant value in several areas such as:

- For a rapidly growing business where time-to-market is key
- Test the market situations where functionality needs to be rolled out in a low-cost manner focused on validating the demand.
- For improving the functionality of OOB (out-of-the-box) application or Software as a Service (SaaS).
- a service that reduces customization and hence makes future application upgrades easier.
- Refactoring and modernizing legacy applications
- The publish-subscribe messaging pattern employed in microservices architecture provides seamless asynchronous communication for streaming platforms to generate intelligent outputs by processing and analyzing data in real-time.
- When DevOps and microservices are used together, they perform wonderfully. By adopting a standard toolset that can be used for both development and operations, microservices help the DevOps team become more productive.

IV. CONCLUSION

Modern applications require a different foundation, one based on scalable, resilient, and adaptable services. (Microservices) provide a method for developing and deploying highly scalable, cross-platform services. distributed and scaled in a decentralized manner to give the most experienced modern application product team control and flexibility.

REFERENCES

[1]. "Docker Container Service for Microservice Applications." (2018).

[2]. Lewis and M.Fowler, "Microservices," 2014, Last access 21-Sep-2016.[Online].Available:http://martinfowler.com/articles/microser vices.html

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, April 2022
Impact Factor: 6.252

[3]. N. Huber, M. v. Quast, M. Hauck, and S. Konev, "Evaluating and modeling virtualization performance overhead for cloud environments," CLOSER, pp. 563–573, 2011

[4]. Microservices use-case: https://compugain.com/microservices-use-cases/

[5]. Docker: https://developer.ibm.com/articles/breaking-down-docker-and-microservices/

[6]. Docker: https://www.tutorialspoint.com/docker/docker_overview.htm