

Docker & Containers, The Future of Microservices

Goutam Kamate¹, S.G Raghavendra Prasad²

¹Information Technology, R.V College of Engineering, Bengaluru, India 1RV18SIT06

²Professor, Dept of Information Science & Engineering, R.V College of Engineering

Abstract: Docker is a container technology and is also called as container-based virtualization which has multiple instances on operating system running over a single kernel. Here the operating system's kernel runs on the underlying hardware with isolated virtual machines called as containers. Micro-service architecture is not a new thing but started getting attention when the docker was introduced and many companies around the world are shifting from a standalone architecture to micro-service architecture. With lot of pros, Docker is a very good fit for micro- service architecture application. In this paper we will discuss the micro-services architecture and how docker will help to resolve the challenges in micro-service architecture.

Keywords: cloud, docker, container, container orchestration virtualization, hypervisor, automation, micro services

1. Introduction

Docker is a container platform which can be used to run the applications with in lightweight containers or also called as running the application in isolated environment. The docker came into existence on the year 2013 changed the way of software deployment, Since then the popularity of operating system virtualization gained popularity, before docker there was a virtualization technology called hypervisor where we can only run one virtual machine at a time and running second could impact on the first virtual machine's performance and it had various problems like lack of flexibility and performance compared to docker.

Nowadays the most popular cloud vendors such as amazon web service, Google cloud, Microsoft azure and many more cloud vendors added container as a service to their platform to support container technology.

A formal approach of building the application in last generation was by using monolithic architecture where one deployment has several responsibilities and handling this kind of application was a headache because if anything goes wrong, they have to debug the entire system to check what has gone wrong. Monolithic is good for small scale applications but in very large applications it's a very big barrier.

So here comes micro-service architecture in the picture to avoid the limitations of monolithic design, so as the code base gets bigger it's better to decouple the application into multiple services so that the application can be handled easily and also application have the advantage of flexibility. There were some downsides for micro- service approach so docker was introduced as the solution to solve micro service architecture.

2. Microservices Architecture

Microservice is a collection of small services that work together to fulfill the business requirements. In this section we discuss overview, its benefits, drawbacks and characteristics.

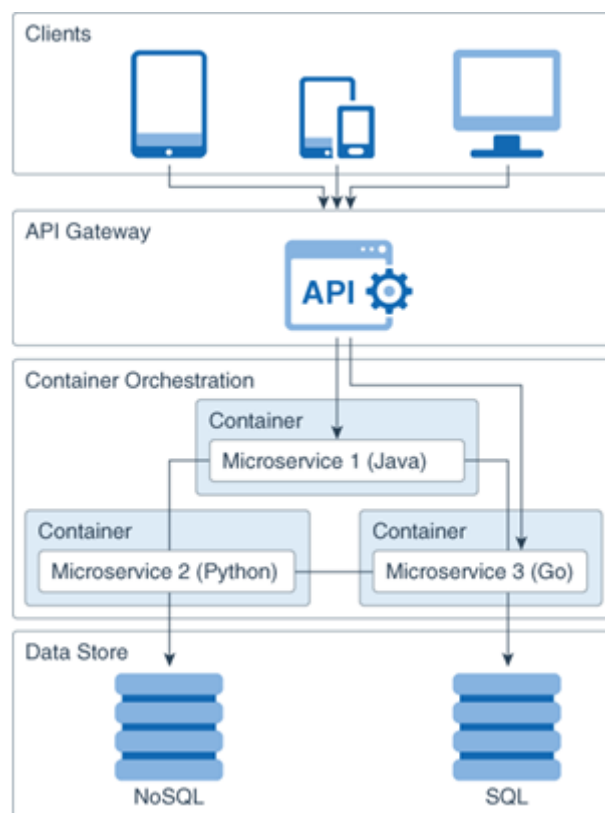


Figure 1: Microservice architecture overview

The above diagram represents the microservice architecture where java, python, SQL, API acts as a different service.

a) Benefits:

- Easy to build and maintain the apps: Since the application is divided into small services it easy to build small components or services and are easily maintainable
- Improved productivity and speed: Different teams can work on different services simultaneously. Separate services are easy to modify
- Better testability: Since the services are smaller and are easy to test
- It can help you organize development effort with multiple and autonomous teams

b) Drawbacks:

- Developer must implement inter-service communication mechanism
- Testing the interactions between multiple services
- Since it's a multi service architecture it comes with deployment complexity
- Increased memory consumption.
- Coordination between the services that communicate with each other.

c) Hypervisor virtualization and Operating system level virtualization

Hypervisor based virtualization is a windows specific and is only found in Microsoft windows operating system. The focus of hypervisor-based virtualization is to imitate the underlying physical hardware and create virtual hardware and on top of all an OS is installed.

Basically, you run a hypervisor on windows and on top of it you can run or install Linux or windows or any platform operating system on hypervisor and vice versa. Hypervisor based virtualization consumes full underlying hardware of the current operating system, so it may affect the performance of the system running hypervisor and at a single time if user tries to run multiple operating system on top of hypervisor, the system may crash or not respond properly.

The basic thing all users or geeks should understand about hypervisor is that everything is done on hardware level.

The below figure shows the architecture

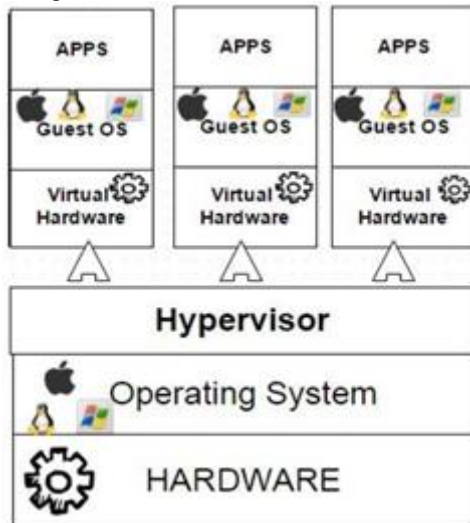


Figure 2: Hypervisor architecture

The main performance impacts for hypervisor happens due to the fact that there are multiple memory and cpu managers for guest operating system, so it may cause an overhead, results in performance.

On the other hand, container-based virtualization which is based on hardware level but is done at an operating system level.

The main difference between container virtualization and hypervisor is each container shares the same kernel of the

base system where as hypervisor needs separate kernel for each guest OS.

Containers are smaller in size and are lightweight compared to virtualized guest OS. Basic idea behind containers is it can provide separate environment, without a virtual hardware emulation which is same as virtualization, but each container deals with its own network and filesystem.

d) Language-agnostic(cross-language)

Microservices are built with the language that developers are comfortable with, development should not restrict to any specific programming language, meaning while developing microservice application the developers has the freedom of choosing any technology by which developers fulfill the requirements. The technology in microservice can be anything that is C, C++, Java, Python an many more recent technologies.

Since microservice is language neutral, communication with each service becomes important factor and the developers or users has the option of using REST based communication. It's not necessary that each service can have same programming language and can use different languages.

3. What is Docker?

Docker is a opensource tool or an application which is designed to run applications by containers.

Containers help developers to package an application with all the required dependencies and act as one package. With help of containers developers are assured that the application can run in any platform without any issue of dependency missing of incompatibility which helps them writing and testing easy.

Docker allows application to share only needed libraries of kernel or operation system and avoid full installation of specific guest OS.

To build docker containers first step is to write a dockerfile with contains instructions for installation of required dependencies and fetching the code from git repository or download the required docker image from container to use as a base image from docker hub, which is like a repository for docker images.

Second step is to build the docker image if you write docker file. Docker image is a packaged version of all project which acts as one application or a mini operating system.

Third step is by using the built docker image, run the application. The common terminologies we hear in docker is docker image, containers, run, process, volumes.

Docker containers are loaded into container ship with one or more containers which in-turn has an application running inside it and communication from one container to another makes it easy.

a) Docker Architecture

Docker uses a client-server architecture an comprises of

docker client, Host, Network and storage components and the docker hub. The below diagram illustrates architecture diagram of docker.

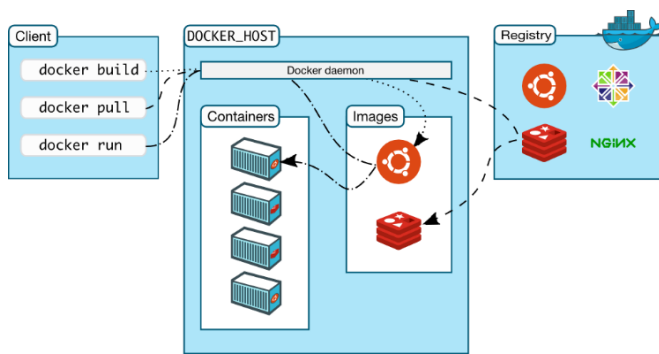


Figure 3: Docker architecture

Docker client: Client enables users to interact with docker. Client can reside on same host as remote host. Docker client can communicate with more than one daemon, It provides CLI by which we can run, build and stop containers.

Docker Image: Images are binary template used to build containers. Basically, Images contain metadata that describe the containers capabilities. Docker images are core part and they enable collaboration between developers.

Docker containers: Containers are form of operating system virtualization; a single container might be used to run anything from small microservice or software.

Docker networking: Docker provides various options in networking and maintains abstraction in network, each container has its own network configuration. The other networks and administrator can configure the same. There are by default three types of network

- a) Bridge network: which forwards traffic to container network
- b) Overlay network: this network is needed when containers are running on separate host.

4. Why docker is preferred as best for Microservice

Docker is best for microservice because docker has faster start time i.e. Containers starts with in a matter of seconds because its just a small process and not a whole operating system.

Docker has a faster deployment and no need of separate environment setup for to run applications, developers only need to download an image from repository or docker hub and play with it. As faster deployment helps start application or service faster.

Easier management of containers makes microservices architecture feasible and easy to manage services.

Proper and efficient usage of resources can be achieved for multiple services using docker by which the application running inside container may have less performance issue.

Since docker supports multiple platforms you can package

and run your docker image in any operating system and don't have to bother about dependencies or environment issues.

5. Conclusion

In this paper we studied about microservice architecture, benefits, drawbacks of the same and also we studied about the overview of docker then we studied why docker is a good fit for microservice architecture and other necessary fundamentals which help in solving the problems using docker.

References

- [1] Sachchidanand Sing, Nirmala Singh, "Containers & Docker: Emerging Roles & Future of Cloud Technology," 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)
- [2] Nitin Naik "Docker Container-Based Big Data Processing System in Multiple Clouds for Everyone" 17th IEEE International Conference on Computer and Information Technology (CIT). IEEE, 2017 .
- [3] Docker.com. (2017) Manage data in containers. [Online]. Available: <https://docs.docker.com/engine/tutorials/dockervolumes>.
- [4] Containers vs. VMs: What's the difference? <http://searchservervirtualization.techtarget.com/answer/C/containers-vs-VMs-Whats-the-difference>.
- [5] Virtual Machines Vs. Containers: A Matter Of Scope- <http://www.networkcomputing.com/cloud-infrastructure/virtual-machines-vs-containers-matter-scope/2039932943>.
- [6] Infrastructure for container projects- <https://linuxcontainers.org/>.
- [7] Containers: Fundamental to the cloud's evolution- <http://www.zdnet.com/article/containers-fundamental-to-the-evolution-of-the-cloud/>.