Microservice Architectural Style

Swathi[1]
1RV18SSE17
MTech, Dept. of ISE
R V College of Engineering
Bangalore, India

Rashmi R[2]
Assistant Professor, Dept. of ISE
R V College of Engineering
Bangalore, India

---***---

***Abstract*—** Microservice are architectural style, which are tremendously growing in Industry and scientific research in academics to achieve agility, scalability, and speed of delivery while developing Microservices. More companies are adopted to this technology architectural style for improve in the all life cycle of product. In this way the monolithic architecture, migrate towards microservice. Microsrvices are breaking the large software component into smaller independent services which helps in software industry to modernizing there product. However, microservices architecture approach also introduces a lot of new complexity and requires application developers a certain level of maturity in order to confidently apply the architectural style. Docker is one of the technology, which used to build and deploy the independent microservices into one application. Docker is a good technology to implementing microservices architecture style. In this paper, we will discuss about migration towards monolithic to microservice architecture. How Docker can effectively help in deploying mircoservices architecture and Docker process.

Keywords*— *Microservices Architecture, Migration towards Microservice, Dockers, ontainers,

I. INTRODUCTION

The most common way to build software application is monolithic architecture. The **Monolithic** application is a single-tiered software application in which different services combined into a single program from a single platform. Monolithic architecture developed using single database and runs on single process. The traditional N-tier monolithic application architecture having a database layer, business layer, and presentation layer. The modules A, B, and C speak to three diverse business capacities.. The layers in the diagram represent a separation of architecture concerns. Each layer holds every one of the three-business abilities relating to this layer. The introduction layer has web segments of all the three modules, the business layer has business segments of all the three modules, and the database has tables of all the three modules.

Monolithic Architecture is good for small-scale project and teams, but when flexibility, scalability, speed of the requirement like quick development, adopting to new technologies, changes in frameworks or languages will affect an entire application. Microservices Architecture styles introduced to overcome the problems of Monolithic Approach. Microservices are having single database for each services in the application. Microservices build and deploy using the Dockers and containerization technology. The further paper explain about the Microservices and how microservices can be deployed using Dockers.

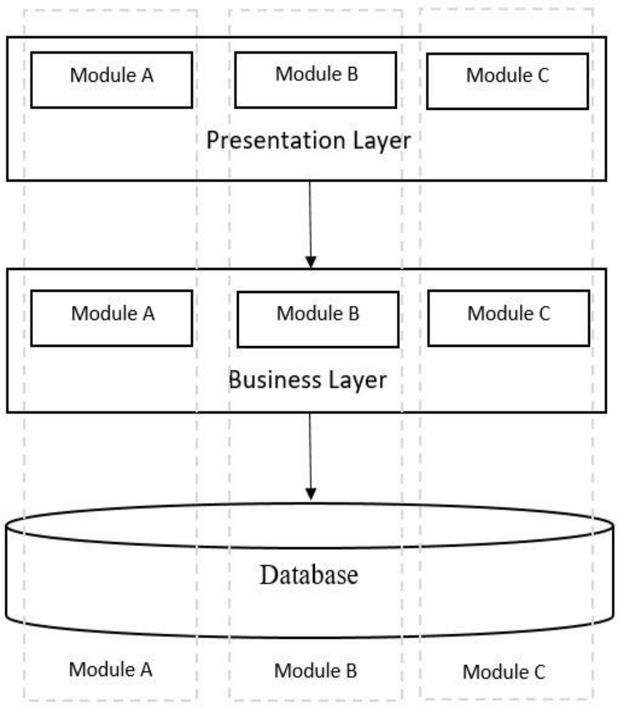

Figure 1: Monolithic Approach

II. MICROSERVICE ARCHITECTURE

Microservice Architectural styles introduced to solve the problem of traditional approach that is monolithic architectural patterns. Microservices are small and autonomous services that work together. Microservices is an Architectural style, in which large software complex applications are consists out of one or more services. Microservices can be build and deploy independently of each other and are loosely coupled and each one of these microservices focuses on completing one task.

Rather than sharing a single database as in Monolithic application, each of these microservices has its own database.

Having a database for each services helps in speed of the development, increase scalability, etc., since it ensure loose coupling. Each of the microservices has its own database. In addition, services can utilize a kind of database that is most appropriate to its needs.

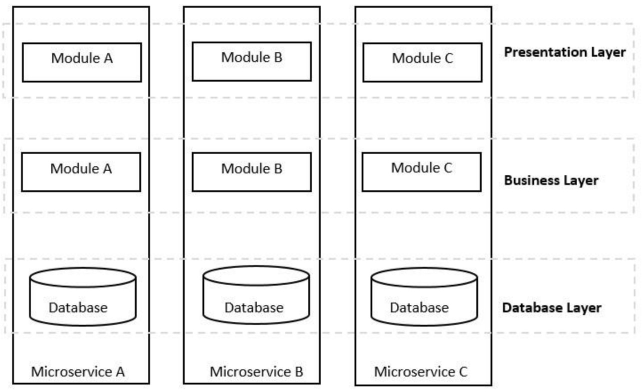

Figure 2: Microservice Approach

Microservices are small independent services that work together to satisfy a business prerequisite. This area will examine about some fundamental ideas and characteristics in micoservice architecture.

1. Small and Focus: Microservices is dividing large applications into independent, small services. Each services is powered by a small, hyper-focused team that is responsible for their service and select the appropriate processes, technologies, and tools for that service. From development perspective, each independent services should treat as one application having their own source code and repository.

2. Language Neutral: Microservices are build using different technology for each independent services. Developer can use the any of the programming language in which they are comfortable. Therefore each microservices can be written in different language depends on their convenient. Communication with microservices is through language-neutral APIs, typically an Hypertext Transfer Protocol (HTTP)-based resource API, such as REST.

3. Loose coupling: Loose coupling is a main characteristic of microservices. Each independent microservice needs to be deployed with the zero coordination with other services.

4. Bounded Context: Each model must have a context implicitly defined within a sub-domain, and every context defines boundaries. In other words, the service owns its data and is responsible for its integrity and mutability. It supports the most important feature of microservices, which is independence and decoupling.

There are some challenges in building microservices that need to address before producing the benefits of microservice.

- Configuration and Management: The microservices need to maintain configuration across the various environment. Since each component divided into different independent services, the configuration of these services is important to communicate between all services.

- Debugging: When multiple service running difficult to debug the each services. Therefore, we need centralized logging and dashboard to make it easy. Therefore, debug is one of the biggest challenge.

- Consistency: It requires decentralized around the language platform technology and tools, which used for implementing deploying monitoring the microservices.

- Independency: One of major principles of a microservices based system is making services highly decoupled. That means it's critical to keep the independency between services so that each of them can be developed, deployed independently without effecting each other.

- Scalability: Scalability is one of the important challenging of the microservices. Since an application is composed of multiple independent microservices which share no external dependencies, scaling a specific micro service instance in the flow is greatly simplified: if a specific microservice in a stream turns into a bottleneck because of moderate execution, that services can be run on more powerful hardware for increased performance if required, or one can run various cases of the Microservice on various machines to process information components in equal.

III. DOCKERS

Docker is open source platform designed to create, deploy and run the application using containers. Docker tool used to deploy the microservices. Before Docker, microservices were deployed using virtualization, multiple virtual machine installed on a single host machine. Individual machine used to run individual microservices. Below figure 2 shows stack flow of virtualization for deploying microservices.

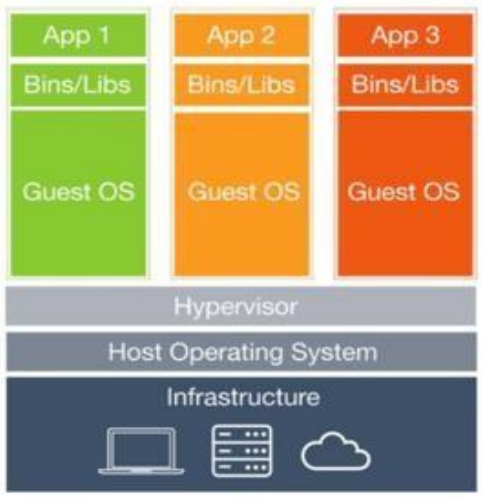

Figure 3: Virtualization infrastructure stack

- Disadvantage: wastage of resources
- To overcome this problem Dockers comes in to picture

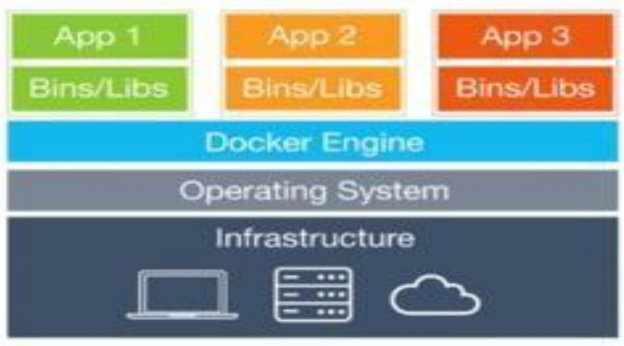

Figure 4: Docker infrastructure stack

Docker is an open source tool designed to make it easier to create, deploy, and run application by using containers. Figure 4 shows stack flow of Docker infrastructure. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package. Docker is an open source project that makes it easy to create containers and container-based apps. Docker is a software platform for building applications based on *containers* — small and lightweight execution environments that make shared use of the operating system Kernel. Originally built for Linux, Docker now runs on Windows and MacOS as well. To understand how Docker works, let us look at some of the components you would use to create Docker-containerized applications.

A. Docker Architecture and Components

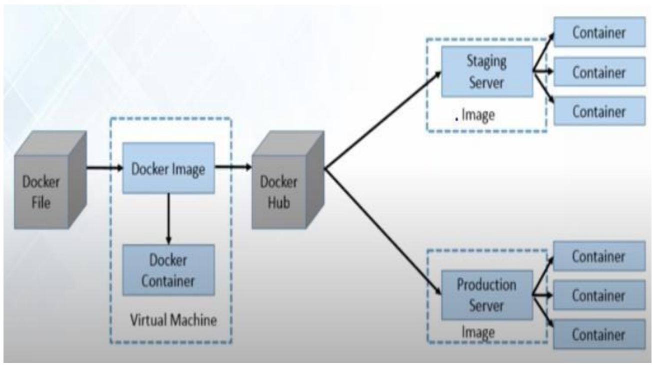

Figure 5: Docker Architecture

- **Docker File:** Each Docker container starts with a Docker file. A Docker file is a text file written in an easy-to-understand syntax that includes the instructions to build a Docker *image* (more on that in a moment). A Docker file specifies the operating system that will underlie the container, along with the languages, environmental variables, file locations, network ports and other components.
- **Docker Image:** Docker image is a portable file containing the specifications for which software components the container will run and how. Because a Docker file will probably include instructions about grabbing some software packages from online repositories.
- **Docker's run utility** is the command that actually launches a container. Each container is an *instance* of an image. Containers designed to be transient and temporary, but they can be stopped and restarted, which launches the container into the same state as when it was stopped.
- **Docker Hub:** Docker Hub is a SaaS repository for sharing and managing containers, where you will find official Docker images from open-source projects and software vendors and unofficial images from the public. You can download container images containing useful code, or upload your own, share them openly,or make them private instead. Moreover, can create our own a local Docker registry.

B. How docker can be used to deploy microservices

Docker Compose: Compose is a tool for defining and running multi container Docker applications. Docker Compose gives an approach to orchestrate various container that work together. Examples include a service that processes requests and a front-end web site, or a service that uses a supporting function such as a Redis cache. If application are developed using microservice model. Then use the Docker compose tool to run application as single unit and communicate between the each independent services using web request. Docker

Compose created by Docker to simplify the process of developing and testing multi-container applications. It is a command-line tool, reminiscent of the Docker client, which takes in a specially formatted descriptor file to assemble applications out of multiple containers and run them in concert on a single host. How Docker works with example: Suppose there are three containers running in one YAML, file and running those containers with single command using Docker compose.

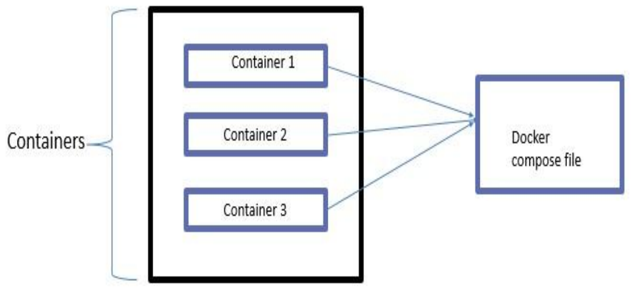

Figure 6: Virtualization infrastructure stack

CONCLUSION

The Conclusion of the paper presents about the motivation and initial steps of an investigation on microservice architectures concepts. Further more explains about how Docker technology can be used to build and deploy the microservices. This helps in successfully applying the beneficial of architectural style.

REFERENCES

[1] J.Thönes, "Microservices," IEEE Software, Dublin, Ireland, Jan 2015.

[2] Irina Astrova, Arne Koschel, Jeremias Dotterl, "Making the Move to Microservice Architecture", International Conference on Information Society (i-Society), 2017.

[3] Paolo Di Francesco, "Architecturing Microservices", IEEE International Conference on Software Architecture Workshops (ICSAW), Gothenburg, Sweden, JUNE 2017.

[4] Alan Sill "The Design and Architecture of Micriservices", IEEE Cloud Computing, Volume: 3, Sept 2016.

[5] Raja Mubashir Munaf, Jawwad Ahmed, Faraz Khakwani, Tauseef Rana "Microservices Architecture: Challenges and Proposed Conceptual Design", International Conference on Communication Technologies (ComTech), 2019.

[6] David Jaramillo, Robert Smart, Duy V Nguyen "Leveraging Microservice Architecture by using Docker Technology", IEEE, 2016.

[7] Gaston Marquez, Hernan Astudillo, "Actual Use of Architectural Patterns in Microservices-based Open Source Project", Asia-Pacific Software Engineering Conference (APSEC), 2018.

[8] Nuha Alshuqayran, Nour Ali and Roger Evans, "A Systematic Mapping Study in Microservices Architecture", 9th International Conference on Service-Oriented Computing and Applications, Brighton, UK, 2016.

[9] G. Granchelli, M. Cardarelli, P. Di Francesco, I. Malavolta, L. Iovino, and A. Di Salle. Microart: "A software architecture recovery tool for maintaining microservice based systems". IEEE International Conference on Software Architecture (ICSA), 2017.

[10] G. Kecskemeti, A. C. Marosi, and A. Kertesz. "The entice approach to decompose monolithic services into microservices". In High Performance Computing & Simulation (HPCS), 2016 International Conference on, pages 591–596. IEEE, 2016.

[11] S. Newman. "Building Microservices". O'Reilly Media, Inc., 2015.

[12] Sachchidanand Singh, Nirmala Singh, "Containers and Dockers: Emerging Roles and Future Technology", International Conference on Applied and Theoretical Computing and Communication Technology, April 2017.

[13] Devki Nandan Jha, Saurabh Garg, Prem PrakashJayaraman, Rajkumar Buyya, Zheng Li, Rajiv Ranjan, "A Holistic Evaluation of Docker Containers for Interfacing Microservices", IEEE International Conference on Services Computing (SCC), September 2018.

[14] Charles Anderson, "Docker (Software Engineering)", IEEE Software Volume: 32, May-June 2015

[15] Fawaz Paraiso, Stéphanie Challita, Yahya Al-Dhuraibi, PhilippeMerle, "Model-Driven Management of Docker Containers", IEEE 9thInternational Conference on Cloud Computing (CLOUD), 2016.