International Journal of Trend in Scientific Research and Development (IJTSRD)
International Open Access Journal
ISSN No: 2456 - 6470 | www.ijtsrd.com | Volume - 2 | Issue – 4

Microservice Oriented Application Development

Sumeet Baburao Patond, Shubham Ramnath Satpute, Smita Sidramappa Patil
Aishwarya Shivkumar Kapase, Prof. D. D. Sapkal
Pune Vidyarthi Griha's College of Engineering and Technology, Pune, Maharashtra, India

ABSTRACT

About a decade ago developers used to follow monolithic approach for developing application. There used to be a single large application containing millions of lines of code. And it would take hours just to build an application. Many people used to work together to build an application. If bug was found in any single line of code then we had to take down application. So other people's work too couldn't get out. So this would delay the release of an application. Now a days the concept of microservices came into picture. In this approach we divide single large monolithic application into multiple smaller chunks. This would reduce the build time from hours to seconds. Developers can work on these chunks independently. So it would enable them to release their code on their own pace. These microservices don't coordinate directly. Each microservice provide interface that can be used by others to get services. Now we need a platform to deploy these microservices. Therefore we wrap them into containers. Containers offer portability and it would take seconds to deploy the container. Docker is an opensource tool which provides a platform to build, deploy and run containers. For orchestration and management of cluster of docker container we can use tools like Docker Swarm or Kubernetes. These tools automates the deployment process and also provide horizontal scaling.

Keywords: *Monolithic Architecture, Microservice Oriented Architecture, Docker Docker Swarm, Kubernetes*

INTRODUCTION

A monolithic application is a combination of data access and user interface in a single program forming a single platform. Monolithic application is designed without modularity.

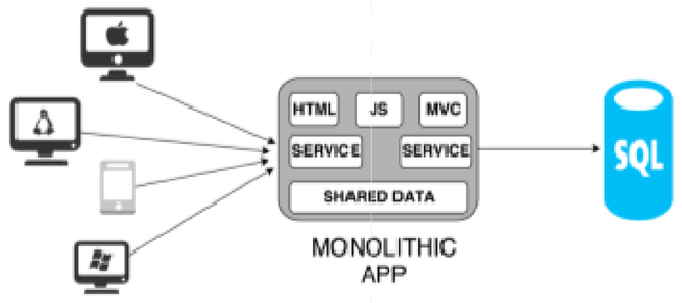

Figure: 1. Monolithic Architecture

Microservices architecture overcomes the drawbacks of monolithic architecture. In microservice architecture applications are deployed as a combination of different simple, independent, executable, scalable services instead of deploying it as a single complex unit. These services can be executed independently. Output of the whole application can be given by combining these smaller services.

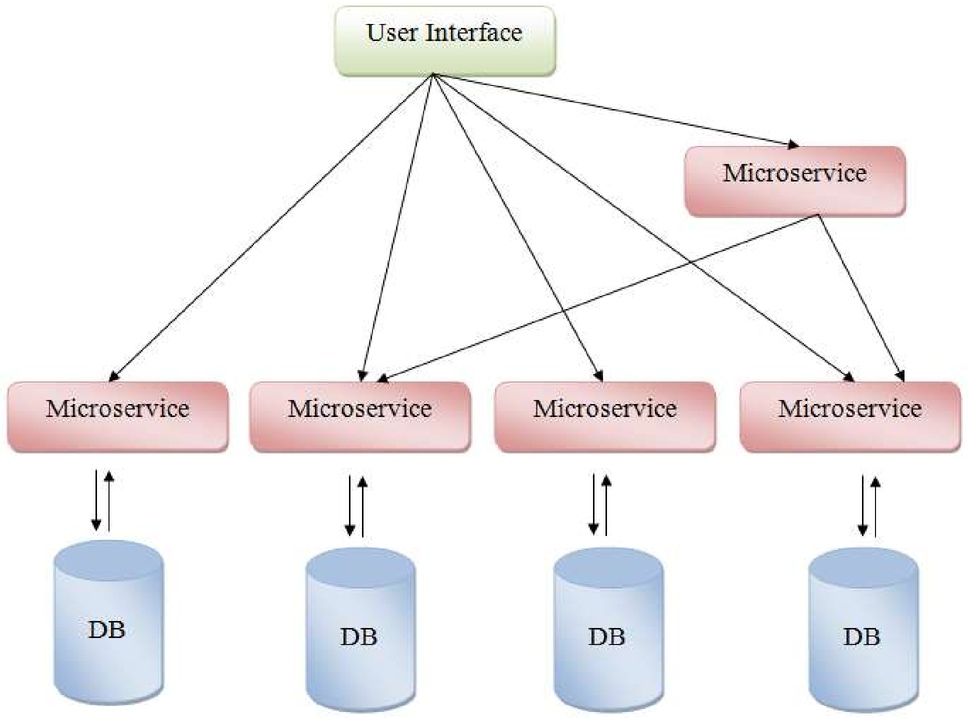

Figure: 2. Microservice Oriented Architecture

[2] MONOLITHIC ARCHITECTURE VS MICROSERVICE ORINTED ARCHITECTURE

- Architecture
- Monolithic Application

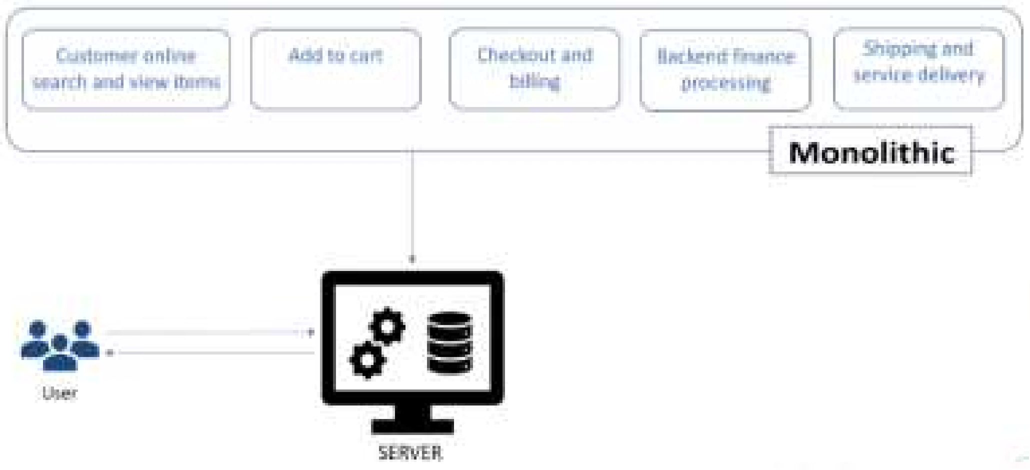
Customer online search and view items
Add to cart
Checkout and billing
Backend finance processing
Shipping and service delivery
Monolithic
User
SERVER

Figure: 3. Monolithic Architecture for Online Shopping Application

Here the above components are written in a single application file and they are deployed in a single high-performance server. And if any changes are to be made in shopping and service delivery' the whole application needs to be complied and deployed. If there is any problem associated with any part of the application, the whole application is not accessible.

- Microservice Application

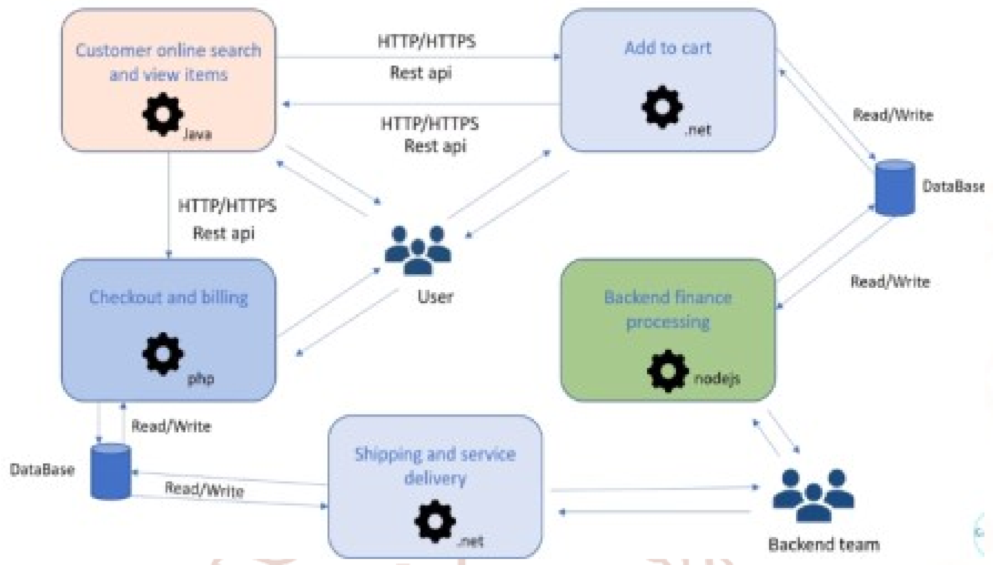

Figure: 4. Microservice Oriented Architecture for Online Shopping Application

Here all the components are deployed separately. These components can be developed in different programming languages such as java,.net,etc. If any changes are to be made in 'Shipping and service delivery' then only this part needs to be compiled and tested and not the whole application. Also if any problem occurs in any part of the application only that part is not accessible.

- Scalability

Microservice[7] applications are easy scale as compared to monolithic application. Suppose in monolithic application the single server is not able to handle the load of 'Customer online search View items' service then there is need for another server and the whole application needs to be deployed in the second server, resulting in doubled cost of the server. But if the similar problem occurs in microservice application as there are servers for all the components of the application, only the server of 'Customer online search View items' service needs to be scaled and not the servers for other components are kept as it is. Also the components in microservice architecture are included according to their usage.

- Database

Monolithic architecture has a large database with single schema containing multiple indexes. It becomes difficult to manage such database. Whereas microservice oriented architecture have database for every service making it simpler than monolithic architecture.

- Deployment time

In monolithic application even for a small change, the whole application has to be recompiled and tested thereby lessening the deployment frequency while in microservices only the changed part has to be recompiled and tested and not the whole application so it has more deployment frequency than monolithic architecture.

[3] DOCKER

Docker [3] is an opensource project which provides a platform to build, deploy and run distributed applications. Docker is supported by Google Cloud Platform and by Google Kubernetes Engine. There are other options to Docker in the field e.g.Rocket[4] and LXD [5] for linux, in addition to Drawbridge [6] for windows. Docker uses a very different kind of virtualization called *containers*. Container shares the kernel of host operating system .Therefore container image doesn't need to contain operating system. Thus it reduces the overload on memory and CPU. It only contains an application with all the dependencies and libraries it needs. A container behaves as if it has its own OS, file system and anything else that can be expected in a virtualized machine. It creates resource isolation between containers. Docker makes it possible to run that containerized application on any other Linux machine irrespective of any customized settings. Since containers are small in size, it provides lightweight virtualized environment.

[4] DOCKER SWARM

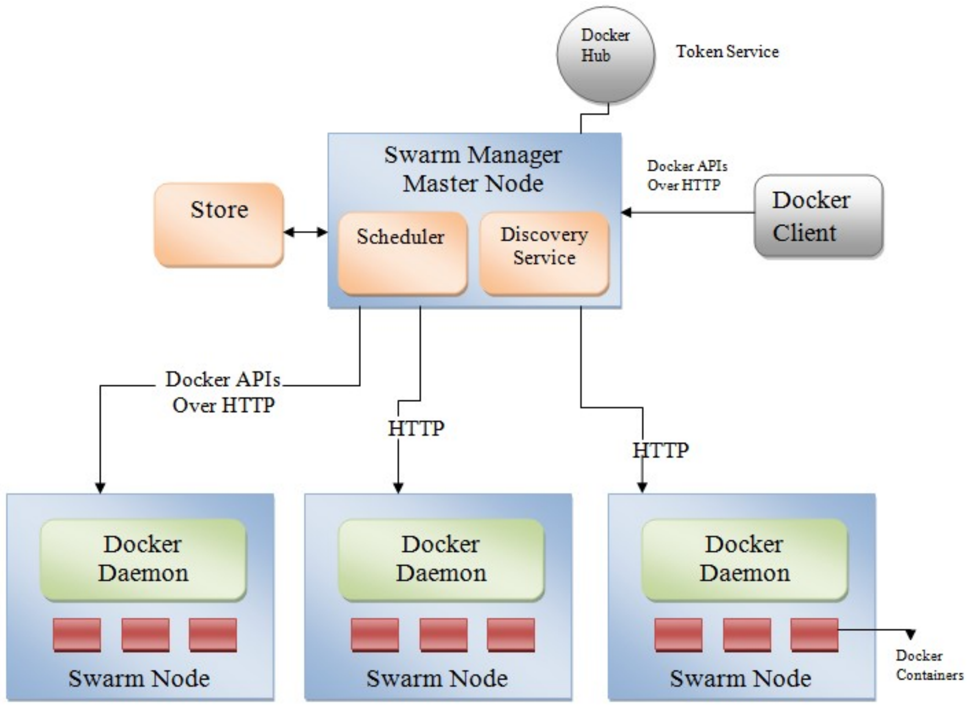

Figure: 5. Docker Swarm Architecture

Docker Swarm is a technique of creating and managing a cluster of docker containers. Swarm cluster is a network of docker engines connected to each other. Swarm manager initializes the whole swarm. Swarm manager manages the execution of applications running over swarm nodes. Main job of swarm manager is load balancing. Load balancing is done by dividing the applications on different swarm nodes and then executing them.

[4.1] DOCKER SWARM FEATURES

Although if a particular node goes down, the services on that node are executed on other active nodes. Therefore even if the node is down, the service is not hampered. Load is balanced among the nodes by the swarm manager for the efficient execution of the services. There is no need to balance the load; it is done automatically, because the swarm manager manages the internal DNS server. The DNS server takes care of the connection among the nodes and it balances the coming traffic over the cluster. The swarm nodes or the swarm manager node can be accessed from anywhere. After accessing the node the services deployed in the respective nodes can be accessed. Servers can be scaled by simply executing a single command. Accordingly the services can be deployed into the servers. If the servers are deployed in virtual machines, their updates can be done by individually updating each service. But with docker the rolling updates functionality can be used where delay is specified. Each service in the node is updated for delay amount of time. So even if one service is getting updated the other services are available for use, thereby resulting in high availability of the services.

[5] KUBERNETES

Kubernetes[8] is an open source, container orchestration platform for automating deployment, scaling and management of containerized applications. For the deployment of large number of applications more numbers of containers are required, management of these containers becomes difficult. So there must be configurations to deploy services to different nodes and we should know about the services contained in containers, therefore the orchestration is required and it is done by kubernetes.

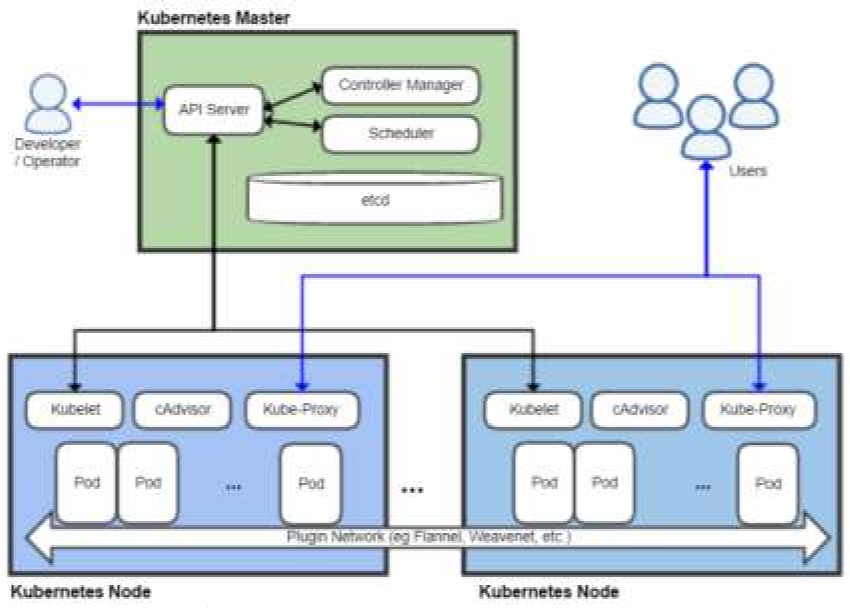

Figure: 6. Kubernetes Architecture

Master Node

Master node manages the cluster and it orchestrates the Kubernetes nodes. Operator interacts with the system through *API Server* by executing commands. *API Server* is the only component which talks to the *etcd* cluster. *ETCD* cluster is the distributed key value store. One can read or enter the data through REST API or by using the command line tools.All the states and configurations are defined in *etcd*. *Controller Manager* takes care of the clusters and it executes the commands entered through *API Server*. *Scheduler* finds the free node and it schedules the workload on that free node.

Kubernetes Node

Kubelet is responsible for the scheduling of pods. It takes care that the assigned *pods* to it are running and they are in required configured state. *Pods* are the combination of one or more tightly coupled containers. The containers in *pod* share IP addresss, namespaces, storage, cgroups,etc. Depending on the state of the server and service pods are created and destroyed. Routing of the packets over the network is managed by *kube-proxy*.

[5.1] KUBERNETES FEATURES

It arranges the containers depending on the resource requirements and constraints without compromising the availability. Application can be scaled by simply executing a command or by using the Graphical user Interface or based on CPU utilization it can be scaled automatically. If the user wants to modify something for e.g schedule then it can be modified easily by visiting the kubernetes repository and modifying it according to the use. Kubernetes can be run on any public cloud. Depending on the choice of user, it automatically mounts the storage system from local storage, AWS or from NFS, cinder,etc. Whenever a particular node fails or goes on then it replaces the containers and again schedules them. It restarts the failed containers. It destroys the containers which does not respond correctly to users.

[6] CONCLUSION

Using microservice approach instead of monolithic has really speeded up the development process. Wrapping these microservices into containers provides the portability. It offers ability to move our application from developer's computer to datacenter. Docker provides ability to build and deploy the container. Tools like Docker Swarm and Kubernetes helps in managing the cluster of containers, automating deployment of containers and provide scaling.

REFERENCES

1) Andreas Habl, Orthodoxos Kipouridis, Johannes Fottner : Deploying Microservices for a cloud-based design of system-of-systems in Intralogistics (Munich, Germany, 861-866,2017)

2) Rui Chen, Shanshan Li, Zheng Li: From Monolith to Microservices:A Dataflow-Driven Approach, 2017, pp. 466-475.

3) What is Docker, [Online Available] https://www.docker.com

4) Rocket-a pod native container, [Online Available] https://github.com/rkt/rkt

5) LXD-a next generation container manager, [Online Available] https://linuxcontainers.org/lxd/

6) Drawbridge, [Online Available]

7) Microservices, [Online Available]

http://microservices.io/

8) Kubernetes, [Online Available] https://kubernetes.io/

9) Kubernetes vs Docker https://vexxhost.com/blog/kubernetes-vs-docker-swarm/

10) Deploying microservices with docker

https://linode.com/docs/applications/containers/deploying-microservices-with-docker/