Enhancing Security to the MicroService (MS) Architecture by Implementing Authentication and Authorization (AA) Service using Docker and Kubernetes

Sunil Bhutada, Kanuga Karuna Jyothi

Abstract*— As the market in addition to business draw out to improve; single solid design is confronting a lot more troubles now-a-days. So the broadening of distributed computing in addition to holder aptitude support MicroSerice(MS) engineering turned out to be increasing respected. While the low coupling, fine granularity, versatility, adaptability just as autonomy of the MS development get facilitate, the natural unpredictability of the conveyed framework makes the security of MS engineering critical in addition to troublesome. This paper intends to become familiar with the Authentication in addition to Authorization (AA) Service for MS engineering for backend get to. By contrasting and the regular measures in addition to breaking down on current innovation. This paper we actualize an Authentication service (as appeared in result) where client login to get to the asset of database, to send effectively and use anyplace on the planet, we manufacture our service as Docker image and introduced it through Kubernetes (Orchestration stage i.e, to manage Docker containers) and we generate JWT token to approved client, as a key to get database. This paper propose a gathering of AA Service procedures appropriate for MS building, for instance dispersed session, SingleSignOn plans, Customer side JSON Web Token(JWT) using Docker and Kubernetes.

Keywords*— MicroService (MS) Architecture, Authentication and Authorization (AA), Kubernetes, Docker Container, Json Web Token(JWT) and DataBase(DB).

1. INTRODUCTION

In the present incredibly aggressive business condition, clients are more requesting than any time in recent memory and will desert a business that is too moderate to try and consider responding. This has put an onus on IT to convey arrangements that give an all-encompassing and uniform involvement to the client, overall business channels. MS design can possibly address this business move; it is tied in with accomplishing pace and wellbeing at scale and it gives the adaptability to pick and pick innovation for actualizing an answer. This methodology positions IT as a colleague as opposed to in a customary help job.

To allow level scaling, substantial scale programming structures are regularly administered in addition to along these lines incorporate a few machines collaborating through systems. In this specific circumstance, MicroServices are a novel improvement test for such structures wherein the general usefulness is provided with the guide of an enormous assortment of little programming segments.

MS design comprises of a suite of autonomously deployable, little, particular, and compassable administrations. Each administration runs an extraordinary procedure and conveys through an all-around characterized, lightweight instrument to serve a business objective. It lines up with the business to manage changes in spry style, matches business changes with deft reaction, and conveys arrangements in a decentralized way.

S no	Docker Swarm	Kubernetes
1.	Docker's very own orchestration device	Google's open source arrangement device
2.	More youthful than Kubernetes	More seasoned than Swarm
3.	Easy to setup being local device to Docker	Bit complex to setup however once done offer more usefulness than Swarm
4.	Less people group around it yet Docker has astounding documentation	Being Google's item and more established has gigantic network support
5.	Straightforward application convey in type of administrations	Bit complex application send through cases, arrangements and administrations.
6.	Has just order line interface to oversee	Additionally offers GUI option to CLI
7.	Checking is accessible utilizing outsider applications	Offer local and outsider for observing and logging

Table1. Docker Swarm vs Kubenetes

AA is major issue in the MS. The services should communicate with each other in secured way. Now we have AA service as Docker image, to manage this, all are using Docker swarm, which orchestrates only 10-20 containers. For larger applications, we utilize 100s of container. These containers have to be maintain efficiently without any fail. Docker swarm fails here and to overcome this problem

Revised Manuscript Received on December 22, 2018.

Dr. Sunil Bhutada, IT Department Sreenidhi Institute of Science & Technology Yamnampet, Ghatkesar Hyderabad - 501 301,Telangana, India. (E-mail : sunilb@sreenidhi.edu.in)

Kanuga Karuna Jyothi, IT Department Sreenidhi Institute of Science & Technology Yamnampet, Ghatkesar Hyderabad - 501 301,Telangana, India. (E-Mail : karunajyothikanuga@gmail.com)

Kubernetes came into existence. Our application based on it.

1.1 Docker

A Docker Image is a record, involved different layers, used to execute code in a Docker compartment. A picture is basically worked from the guidelines for a total and executable adaptation of an application, which depends on the host OS bit.

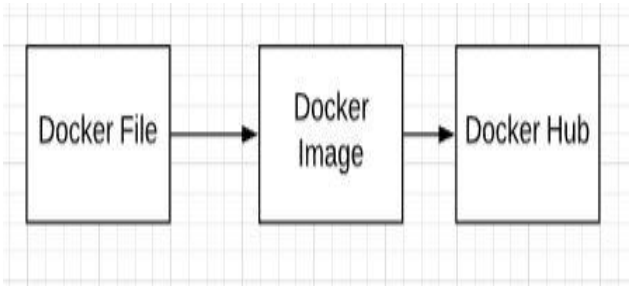

Fig1. Docker

At the point when the Docker client runs a picture, it ends up one or numerous examples of that compartment. Docker center point is the world's biggest vault of compartment pictures with a variety of substance sources.

Docker is an open source working framework level virtualization accommodated Linux and Windows Docker utilizes asset seclusion highlights of the OS piece, for example, cgroups in Linux, to run different free holders on a similar OS.

A compartment that moves starting with one Docker condition then onto the next with a similar OS will work without changes, on the grounds that the picture incorporates the majority of the conditions expected to execute the code. A holder varies from a virtual machine (VM), which typifies a whole OS with the executable code on a deliberation layer from the physical equipment assets.

1.2 Kubernetes

Kubernetes plus Docker are both extensive true answers for keenly oversee containerized applications and give ground-breaking capacities, and from this some perplexity has developed.

Kubernetes consists few major components in it. They are Kubeadm, Kubectl, Kubelet, Kubeproxy and etcd. Kubeadm is used for administrating the kubernetes cluster, kubectl is utilized for controlling the setups on different hubs inside the cluster and Kubelet is an agent, which works on each worknode and communication between nodes. Kubernetes follows Master-Node methodology. Master node consist of API Server, Scheduler, Controller Manager and etcd.

Kubernetes is a holder organization framework for Docker compartments that is more broad than Docker Swarm and is intended to arrange bunches of hubs at scale underway in an industrious way. It works around the idea of cases, which are planning units (and can contain at least one holder) in the Kubernetes, environment and they conveyed among hubs to give high accessibility. One can without much expand run a Docker expand on a Kubernetes group, yet Kubernetes itself is definitely not a total arrangement and intended to incorporate custom modules.

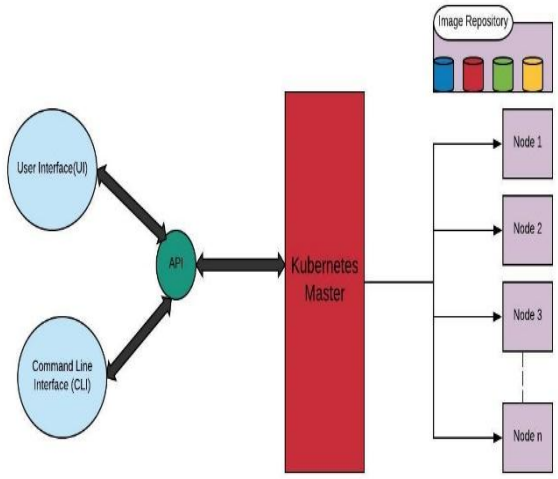

Fig2. Kubernetes Architecture

Kubernetes and Docker are both on very basic level distinctive advancements however, they work great together, and both encourage the administration and arrangement of holders in a conveyed engineering.

1.3 PostgreSQL

PostgreSQL is a stunning, open source object-social database system that uses and grows the SQL language joined with various features that safely store and scale the most befuddled data remarkable jobs needing to be done.

PostgreSQL goes with various features expected to empower designers to make applications, chiefs to verify data decency and develop blame tolerant conditions, and help you manage your data enormous or little the dataset.

PostgreSQL is exceedingly extensible. For example, you can describe your own one of a kind data types, work out custom limits, and even create code from different programming tongues without recompiling your database! PostgreSQL endeavors to suit with the SQL standard where such conformance does not revoke regular features or could provoke poor basic decisions. Immense quantities of the features required by the SQL standard are maintained, anyway at times with insignificantly differentiating language structure or limit.

2. RELATED WORK

Throughout the years, get to the executives has advanced from a security to empowering agent of business development by empowering associations to complete their occupations safely, productively and cost successfully, in the end turning into a key apparatus for hazard and resource the board in associations. PrivX is get to the executives programming created by SSH Communications Security Oyj. Here, Rajagopalan Ranganathan containerized PrivX and conveyed it with Kubernetes. The fundamental objectives of this work is to setup a custom holder arrangement condition that would empower PrivX to be conveyed both on-premise and on open cloud specialist co-ops. Creator did a component correlation of holder organization systems: Kubernetes, Docker Swarm and

Nomad. Author assessed these highlights concerning PrivX necessities and as far as help from open source network. Author found that Kubernetes is the most appropriate choice for PrivX arrangement. Next, creator containerized the MSs in PrivX. The database was likewise made as an administration and typified in a Docker compartment with tenacious capacity. Creator at that point made a robotized custom Kubernetes group that was utilized to convey the Dockerized MSs of PrivX. The versatility of PrivX in his model framework (Kubernetes organization demonstrates) was tried with the 'savvy scaling apparatus' that was made. It was seen that PrivX could scale-out on a level plane when the necessity to deal with more traffic was there and downsize flawlessly when the traffic diminished with the 'keen scaling apparatus'. Also, the guard dog administration isn't required in the Kubernetes demonstrate as it is locally accessible in Kubernetes. Kubernetes locally gives high-accessibility of administrations dependent on the reproduction include referenced the organization detail of each administration. This disposed of the requirement for complex set-up contents utilized in the RPM sending model. Notwithstanding the way that the aggregate picture sizes of the individual administrations (dockerized) is more noteworthy than the single RPM circulation, the administrations are independent and can be sent over a bunch or any Docker compartment upheld structure. Next he assessed the administration recuperation time for every one of the administrations in PrivX both in RPM organization show and in the model Kubernetes sending model. The results conclude, there is no vast peculiarity between the two methods. The administrations were inaccessible amid their recuperation time in the RPM organization demonstrate, while, the administrations were constantly accessible in the Kubernetes arrangement because of their higher reproduction tally. At long last, creator found that Kubernetes helps in simple and fast arrangement of PrivX and gives a bound together answer for both on-reason and cloud sending models. They likewise proposed another verified key foundation conspire. ECIES considers disconnected key determination which is helpful being used situations where the customer and asset server can't straightforwardly convey.

Almeida, Washington Henrique Carvalho, et al. exhibited the components that ought to be considered for the improvement of arrangements dependent on MSs, depicting how the design dependent on MSs is characterized, distinguishing the components identified with their execution in the distributed computing condition, clarifying the security show pertinent and relating the components that incorporate the gauges and arrangements connected to the engineering dependent on MicroServices.

3. FRAMEWORK

3.1. Proposed System Architecture

The aim of the proposed system is to implement the AA of the end users in MSs architectures. To develop this technique we are using Docker Image and Kubernetes. The engineering of the proposed framework is as per the following:

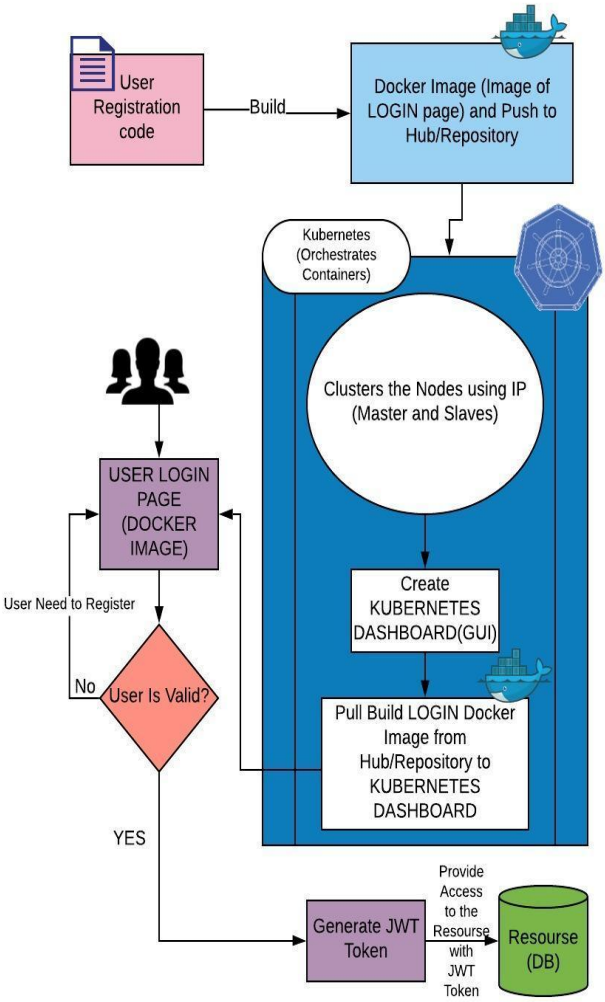

Fig3. Proposed System Architecture

3.2. Technologies

Monolithic Architecture is very complex to understand by the newly joined team member. It has overloaded IDE (Integrated Development Environment) and web container as well, Scaling and Continuous development is difficult. In order to overcome all these limitations, MicroServies came into existence. Our proposed method is also based on MicroServices. AA for the backend access is our moto. Technologies included in proposed system are:

I. MicroServices
II. Docker
III. JWT Token
IV. Kubernetes

3.2.1 MicroServices:

MicroServices are an engineering paradigm style a solid application is decayed into little modest miniaturized scale applications, which are packages and deployed independently.

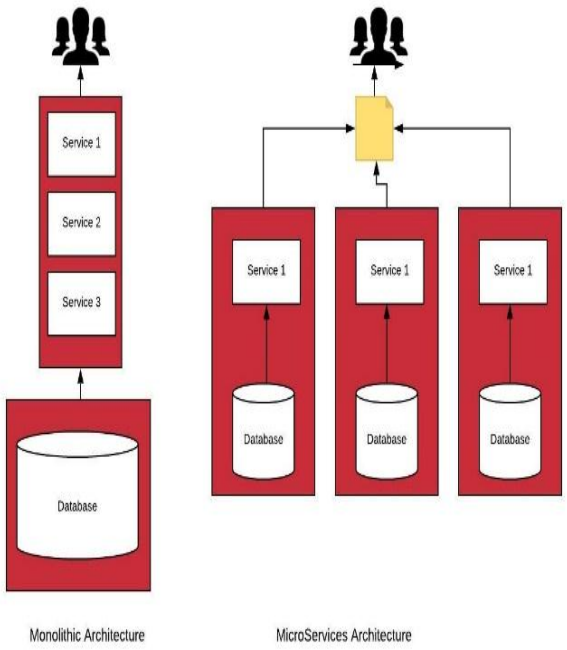

Fig4. Architecture of MicroServices

They speak with one another using REST API or Message Bus. Each service can concentrate on a solitary business capability, which lead to better quality. API is an entry point for clients.

MicroServices can refresh a current administration without revamping and deploy the entire service. Management and Service Discovery are the components of MicroServices.

3.2.2 Docker:

Docker is a Linux container technology, which packages applications into standard format and run those packages in all existing platforms.

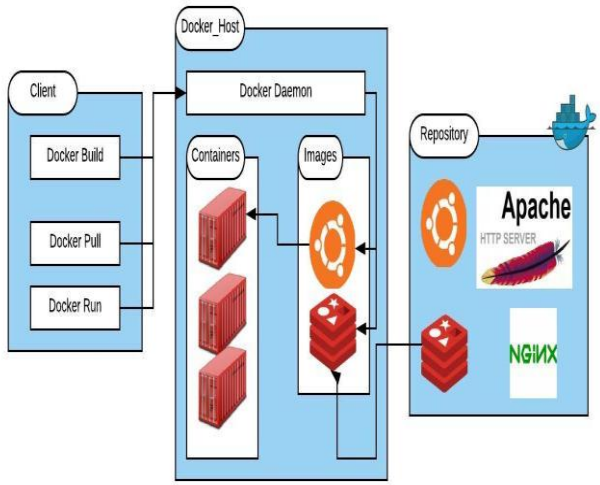

Fig5. Docker Architecture

Docker has its own libraries. It virtualizes the OS Kernel not the Hardware. Docker Images are blueprints of our application. Docker container are running Instance if Docker Image. There is no need of pre-allocation of RAM in the containers.

We can pull the Image from the repository, Build our own Docker Image and Run the Image. The repository can be public or private, as per our requirements. The Image in Public repository can be gotten to by everyone whereas the image in Private repository is restricted to be or particular organization or an individual.

3.2.3 JWT Token:

The outdated approach is to use a session on the server side to save the user state. As the server is stateful it affects the even development of the server. It is suggested to utilize token to record client login status in the MS design. The foremost dissimilarity amongst Token and Session is where the storage is different. Sessions are stored in the server centrally; tokens were stored by themselves and are regularly put away in the program as cookies. The Token consists of the user's identity information, and for each session, the request is sent to the server, the server will be able to identify the visitor and verifies the visitor whether to give the access or discard the access to the requested resource. The Token is utilized to show the client's character. Thusly, the substance of the Token should be encoded to stay away from misrepresentation by the requester or the outsider.

JWT (JSON Web Token) is an open standard RFC 7519 that expresses the token organization content scrambles it and gives lib to different dialects.

Fig6. JWT Token

The structure of JWT Token is extremely basic and comprises of three sections:

Header
Payload
Signature

These three sections are combined using dot(.) as (AAAAA.BBBBB.CCCCC). First block represents Header, contains Hashing algorithm, second block represents Payload, which contains Attributes and third block represents Signature, which is Hashing of Header+"."+Payload+"."+Secret key. The Header, Payload, Signature are then Base64Url encoded to form above three parts. Client uses this token for user authentication.

The server checks the user existed information and provides token only after successful registration. The client needs to send the request to get the JWT token for every session.

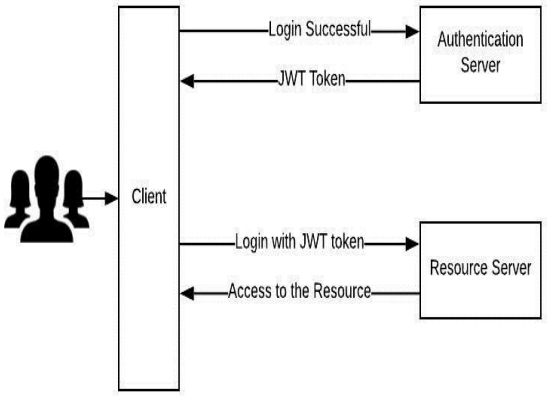

Fig7. Token based AA

3.2.4 Kubernetes:

Docker swam is also an Orchestration platform but in can handle 10-20 containers, making of clustering is very simple than in Kubernetes, there is no GUI for Docker Swarm, Rolling updates done synchronously, no automatic rollbacks and finally all the containers share volumes. These are the limitations of Docker Swarm. Kubernetes overcome the limitations of Docker Swarm. Kubernetes setup is shown below:

	At both Master and Slave	Only at Master	Only at Slave
	Update your Repository.	Initiate Kubernetes Cluster	Join the Cluster
	Turn off swap space.	Install the pod Network.	-
	Update HostName, Hosts and set Static IP.	Setup the Kubernetes Dashboard.	-
	Install OpenSSH Server and Docker.	-	-
	Install Kubeadm, Kubelet and Kubectl.	-	-

Fig8. Kubernetes SetUp

Kubernetes is also an Orchestration platform, It can handle 100-1000s containers, Clustering is complex with number of steps but its worth of it, consists of GUI, Rolling updates are sent to the pods and Automatic rollbacks available in case of failure of updates and finally containers in that particular pod can share volumes.

4. EXPERIMENTAL RESULTS

In this experiment, we used the Docker image and kubernetes for AA of MS. Docker Image is a ReadOnly template used to create containers. The kubernetes is Orchestration of containers, used to cluster the nodes by using their IP address of the nodes in the network. Kubernetes follows Master and Slave Methodology. Docker Hub, which we can push our images and store as Public image or Private image as per our convenient.

After the clustering, the Kubernetes dashboard created and the login Docker image will be pulled from the repository/hub to the dashboard (GUI) then the login page will be unmistakable to every one of the hubs in the bunch. User registers and login into it to get access of the database (backend). Here is the dashboard of the Kubernetes.

While login, if the user is valid, then the JWT token will be generated to verify the client validity. The client, (i.e., who is trying to login) can access the database only if he/she AA using login procedure else discarded.

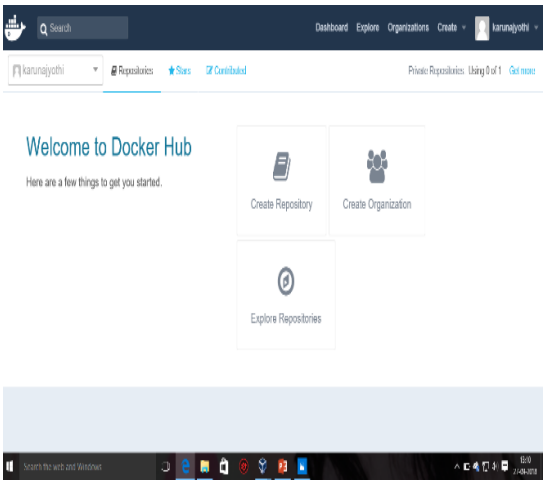

Fig9: Docker Hub

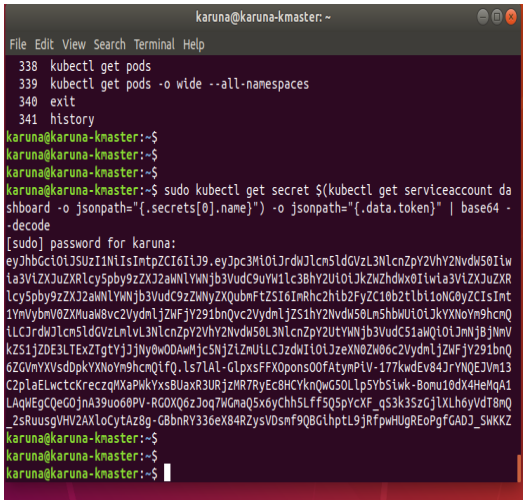

Fig10. Token to access Kubernetes Dashboard

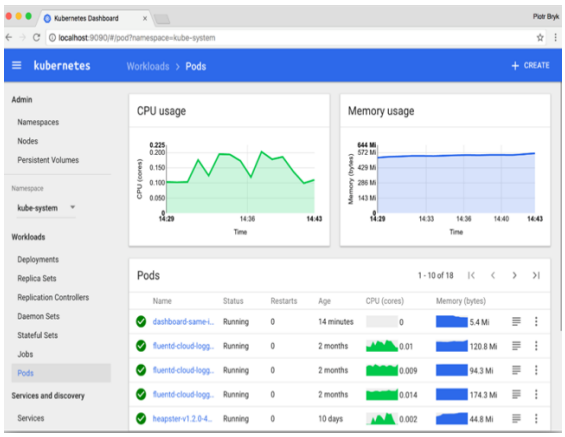

Fig11: Kubernetes Dashboard

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

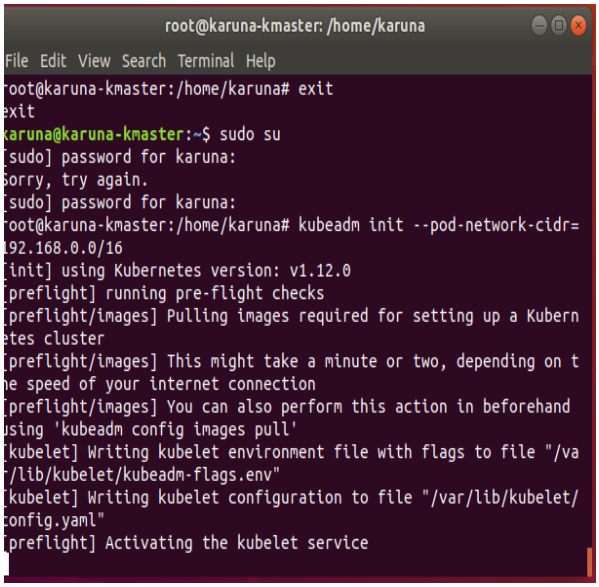

Fig12. Calico Pod Installation in Kubernetes

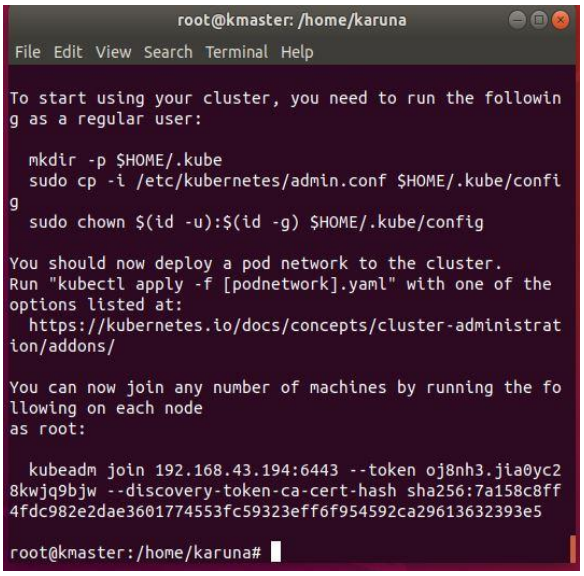

Fig13. Token to join in the cluster for Node (Slave)

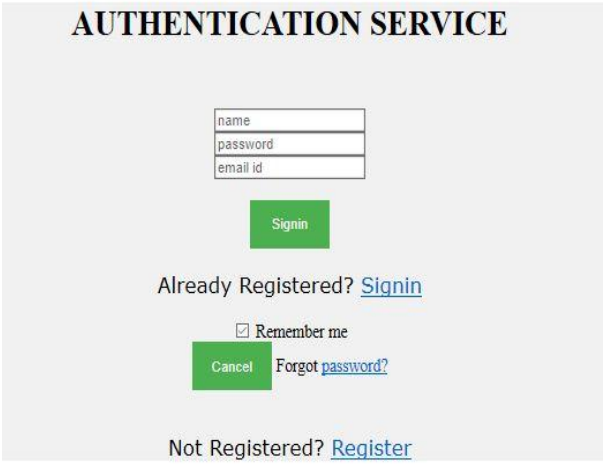

Fig14. Login Page

5. CONCLUSION

Security in MicroServices is frequently neglected plus responsibility is assigned to an API(Application Programming Interface) gateway, that guards internal services and provides AA as well. In this paper, we studied about the AA of the MicroServices to get access of the DB using JWT Token. We implemented AA system for end user under the MS Architecture by using the Docker and Kubernetes. By using this proposed system, we can control unauthorized activities and the more beneficial part is that we can use Kubernetes instead of Docker Swarm for orchestration and get efficient result. This will be more useful in an Industrial usage i.e., for large applications. Through this system, the software developers can manage their applications in secure manner.

REFERENCES

1. Cerny, Tomas; Donahoo, Michael J.; Trnka, Michal. Contextual understanding of MS architecture: current and future directions. ACM SIGAPP Applied Computing Review, 2018, 17.4: 29-45.
2. Pahl, Marc-Oliver; Donini, Lorenzo. Securing IoT MSs with Certificates., IEEE, 2018
3. Timothy Claeys, Franck Rousseau, Bernard Tourancheau “Securing Complex IoT Platforms with Token Based Access Control and Authenticated Key Establishment “, Feb 2018
4. Rajagopalan Ranganathan "A highly-available and scalable MS architecture for access management", Espoo, September 17, 2018
5. Cerny, Tomas; Donahoo, Michael J.; Trnka, Michal. Contextual understanding of MS architecture: current and future directions. ACM SIGAPP Applied Computing Review, 2018, 17.4: 29-45.
6. Almeida, Washington Henrique Carvalho, et al. Survey on MS Architecture-Security, Privacy and Standardization on Cloud Computing Environment. ICSEA 2017, 2017, 210
7. D. Escobar et al., “Towards the understanding and evolution of monolithic applications as MSs,” Proc. 2016 42nd Lat. Am. Comput. Conf. CLEI 2016, 2017.
8. Almeida, Washington Henrique Carvalho, et al. Survey on MS Architecture-Security, Privacy and Standardization on Cloud Computing Environment. ICSEA 2017, 2017, 210
9. D. Guo, W. Wang, G. Zeng, and Z. Wei, “MSs architecture based cloudware deployment platform for service computing,” Proc. - 2016 IEEE Symp. Serv. Syst. Eng. SOSE 2016, pp. 358–364, 2016.
10. J. Bogner and A. Zimmermann, “Towards Integrating MSs with Adaptable Enterprise Architecture,” Proc. - IEEE Int. Enterp. Distrib. Object Comput. Work. EDOCW, vol. 2016–Septe, pp. 158–163, 2016.
11. K. Bao, I. Mauser, S. Kochanneck, H. Xu, and H. Schmeck, “A MS Architecture for the Intranet of Things and Energy in Smart Buildings,” Proc. 1st Int. Work. Mashups Things APIs - MOTA ’16, pp. 1–6, 2016.
12. H. Kang, M. Le, and S. Tao, “Container and MS driven design for cloud infrastructure DevOps,” Proc. - 2016 IEEE Int. Conf. Cloud Eng. IC2E 2016 Co-located with 1st IEEE Int. Conf. Internet-of-Things Des. Implementation, IoTDI 2016, pp. 202– 211, 2016.
13. Alshuqayran, Nuha; Ali, Nour; Evans, Roger. A systematic mapping study in MS architecture. In: Service-Oriented Computing and Applications (SOCA), 2016 IEEE 9th International Conference on. IEEE, 2016. p. 44-51.

14. M. Schöfmann. Security challenges in MS implementations. Container Solutions Blog, January 2016. https:// container-solutions.com/security-challenges-in-MS-implementations/.
15. J. Stubbs, W. Moreira, and R. Dooley, "Distributed Systems of MSs Using Docker and Serfnode," Proc. - 7th Int. Work. Sci. Gateways, IWSG 2015, pp. 34–39, 2015.
16. Dmitry Namiot, Manfred Sneps-Sneppe. "On MSs Architecture". International Journal of Open Information Technologies ISSN: 2307-8 162 vol. 2, no. 9, 20 14. p24-27
17. X. Li. "The design of digital campus unified identity authentication system based on web services", Applied Mechanics and Materials Vols. 427-429 pp. 2301-2304, 2013.
18. J. Li, X. Li and H. Hu, "Research and Improvement of Secure Authentication Scheme Based on Session Initial Protocol", Computer Engineering 2009, vol.35, no.2, pp.162-163
19. N. Ji, X. Lin and J. Pu, "Research on secure authentication based on session initial protocol", Computer Applications (in Chinese), 2007, vol.27, no.3, pp.715-716.
20. Chris Schmidt. "Token Based Authentication - Implementation Demonstration". Workshop Friend of a Friend, Social Networking and the Semantic Web, 1st-2nd September 2004.