iJETRM
International Journal of Engineering Technology Research & Management

CONTAINERIZED WEB APPLICATION AND DEPLOYMENT ON CLOUD

Siddharth Sindhur
Dr. Bhagya R
R V College of Engineering, Mysore Road, Bengaluru
sidsindhur123@gmail.com
bhagyar@rvce.edu.in

ABSTRACT
Over the last few years, Cloud has gradually earned the reputation of being the holy grail of application delivery. Anyone who enters the cloud today would find it tough to choose between the plurality of cloud service providers and face the issue of lock-in.Virtualization based on containers improves the performance and reliability in contrast with traditional hypervisors, since additional resources are removed for operating system. The virtualization based on container fits in a situation where a single operating system is required.It is better to deploy the applications on redundant infrastructure across several zones and spread the workload using load balancing option to secure cloud services against failures and natural disasters. Kubernetes is an open source framework that helps in deployment, maintenance and scaling of container based applications across a host cluster.

KEYWORDS:
Container as a Service, Virtualization, Deployment, application, instances, Key Performance Indicators.

INTRODUCTION

Cloud computing is a digital model that develops and makes the use of IT simpler for customers every day. It provides consumers streamlined applications remotely and in a way that can be accessed on a daily basis. This software can be accessed by as many people as allowed within an entity without having to bother with the maintenance of such an application. It also offers a platform for developing and installing user applications like their storage space and database without disrupting the operating system underlying it.

Additionally, having the option of scaling from a huge number of machines (may be thousands of machines) requires considerable complexity in handling distribution, maintaining, power, scaling and whatever can be hinted by dealing with complex systems.

Cloud computing offers a simple response to the agility criterion. IT organizations can now concentrate on adding market value to their goods, without contemplating selling and maintaining hardware. It is reasonable to believe that certain cloud-based businesses, such as Amazon, are hosting data centers with more than 50,000 servers per region of availability with more than one center per zone.

It is therefore of paramount importance for the servers of an organization to have their systems mounted, configured and operated as quickly as possible and as reliably as possible to help minimize cost. It translates into the automatic setup and delivery.

Container technology is an operating system level virtualization technology, which can provide different file systems, network, process and space (memory) for each running server without changing the operating environment of the host.

Container network, as an essential entity of container technology, resolves the question of contact between containers where container isolation is ensured. Furthermore, failure to transmit the dignity of the operating system makes the container lighter and quicker than the virtual machine that meets cloud data center requirements.

iJETRM
International Journal of Engineering Technology Research & Management

OBJECTIVES

The objective of this project is to evaluate the integration of an application to a cloud environment i.e., Kubernetes, to establish a migration plan for it, to identify Key Performance Parameters (KPIs) and to monitor its performance post deployment.

METHODOLOGY

The approach is the suggested workflow to achieve all project work objectives as shown in figure 1. First, build the application's docker image via its docker-file. This image is built basically from the instructions for a complete and executable version of an application that is based on the host OS kernel. This application image is pushed through to a public repository built on a Docker Hub account. Then the image setup container must run and run locally. Build the cluster to run the picture of the container, as it is stored in the repository. Create deployment of YAML file of the application on Kubernetes environment along with updated version and scale it using replica parameter in YAML file. The scaling will allow more instances of the application to run simultaneously and hence will run more efficiently. In the end the application performance is tested using JMeter tool.

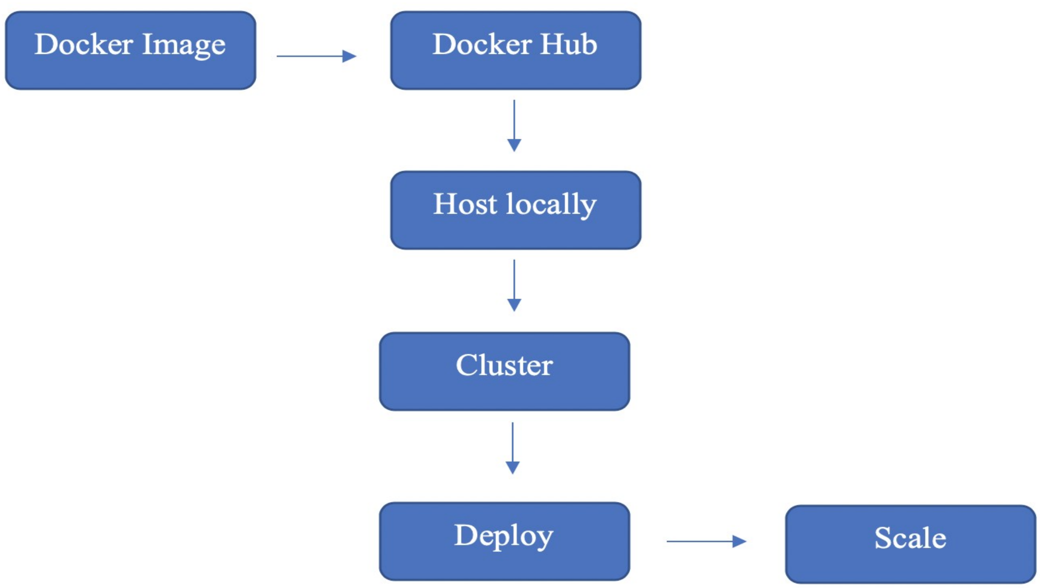


Figure 1 Flowchart for Deployment

RESULTS AND DISCUSSION

First, installation and setup of Docker is required. Then docker image of the application is uploaded to Docker Hub by following steps:

- **Orientation and Setup:** Use docker run hello-world command to test the installation as shown in figure 2.

```
ar-siddharth.sindhur@RINMAC226 ~ % docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Figure 2 Setup

- **Build the Docker Image:** Clone a github project using this command:
  git clone https://github.com/dockersamples/node-bulletin-board
  cd node-bulletin-board/bulletin-board-app

Build and test the docker image as shown in figure 3.

```
ar-siddharth.sindhur@RINMAC226 bulletin-board-app % docker build --tag bulletinboard:1.0 .

Sending build context to Docker daemon  45.57kB
Step 1/7 : FROM node:current-slim
 ---> 46ae64084208
Step 2/7 : WORKDIR /usr/src/app
 ---> Using cache
 ---> 77de27cfa996
Step 3/7 : COPY package.json .
 ---> Using cache
 ---> a51b75f4012b
Step 4/7 : RUN npm install
 ---> Using cache
 ---> 832706323f32
Step 5/7 : EXPOSE 8080
 ---> Using cache
 ---> 5cc8e302ecb5
Step 6/7 : CMD [ "npm", "start" ]
 ---> Using cache
 ---> b8343cfb22cf
Step 7/7 : COPY . .
 ---> 0c7045f8f0d9
Successfully built 0c7045f8f0d9
Successfully tagged bulletinboard:1.0
```

Figure 3 Build and test Image

iJETRM
International Journal of Engineering Technology Research & Management

The Docker-file for the bulletin board app is shown in the figure 4.

```
# Use the official image as a parent image.
FROM node:current-slim

# Set the working directory.
WORKDIR /usr/src/app

# Copy the file from your host to your current location.
COPY package.json .

# Run the command inside your image filesystem.
RUN npm install

# Inform Docker that the container is listening on the specified port at runtime.
EXPOSE 8080

# Run the specified command within the container.
CMD [ "npm", "start" ]

# Copy the rest of your app's source code from your host to your image filesystem.
COPY . .
```

Figure 4 Bulletin-board-app Docker-file

Use **docker ps -a command** to list all the docker images as shown in figure 5.

```
[ar-siddharth.sindhur@RINMAC226 ~ % docker ps -a
CONTAINER ID   IMAGE             COMMAND                 CREATED        STATUS
03af4ad71623   bulletinboard:1.0 "docker-entrypoint.s…"  6 weeks ago    Exited (255) 4 weeks ago
57f352bcd5a7   hello-world       "/hello"                6 weeks ago    Exited (0) 6 weeks ago
ar-siddharth.sindhur@RINMAC226 ~ %
```

Figure 5 Docker Images

Upload Image to Docker Hub: Create an account on Docker Hub and create a public Repository as shown in figure 6.

IJETRM
International Journal of Engineering Technology Research & Management

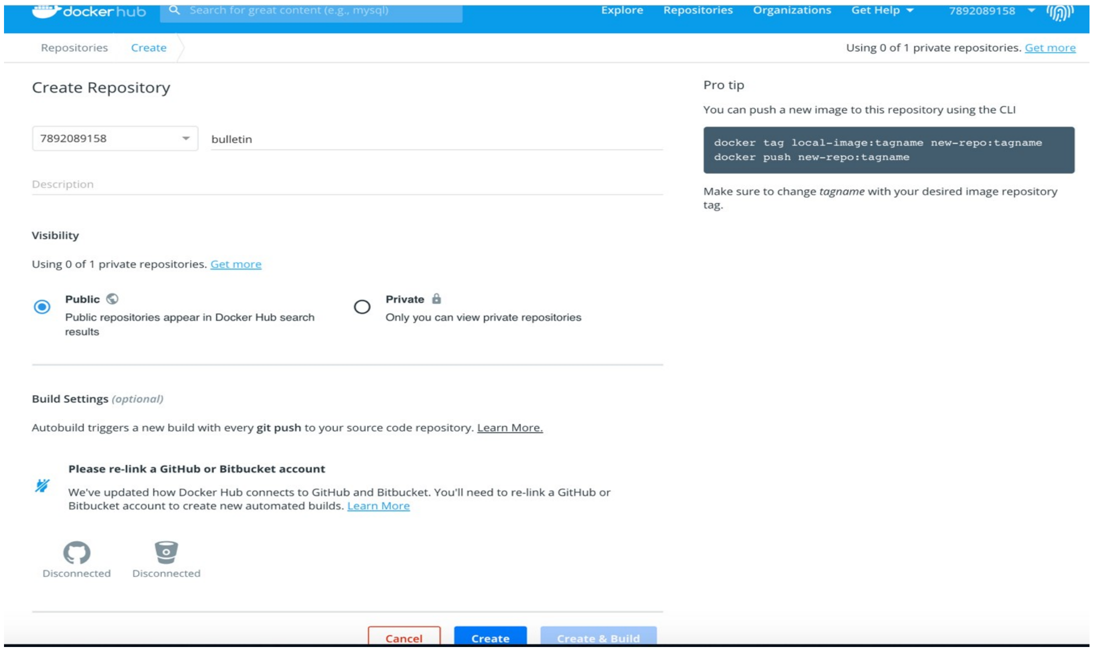


Figure 6 Docker Hub Repository

Push the image to repository as shown in the figure 7.

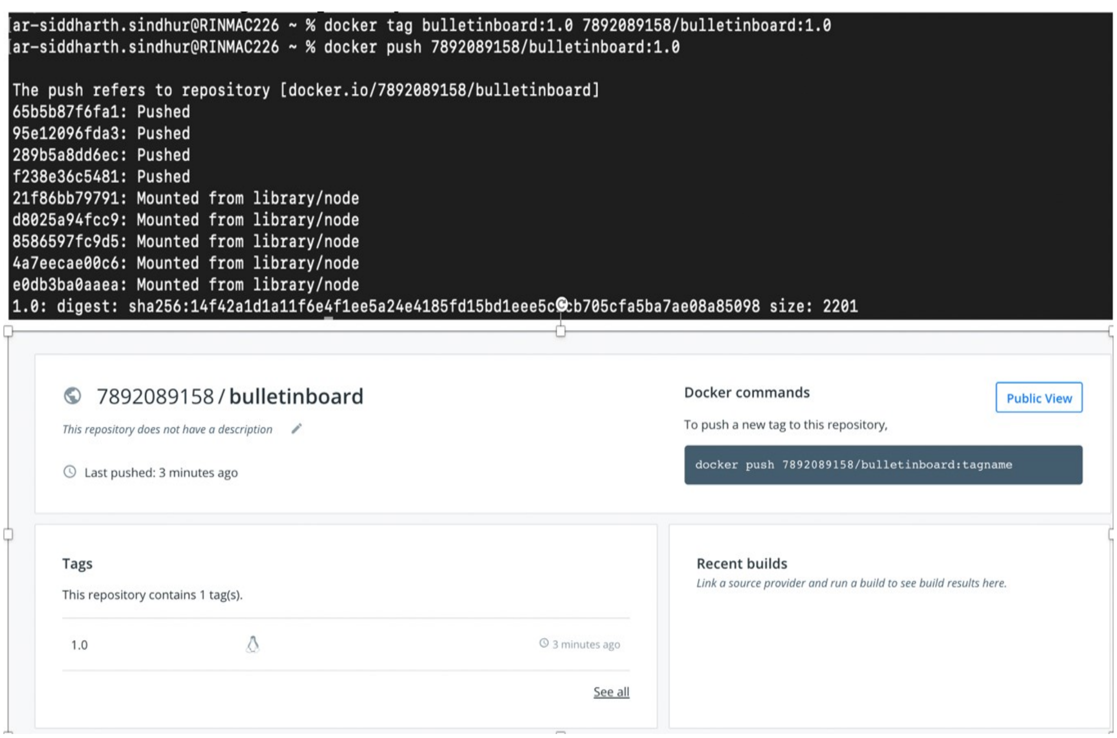


Figure 7 Docker Image in repository

iJETRM
International Journal of Engineering Technology Research & Management

Run the Application on Kubernetes
Start the Kubernetes cluster and check version as shown in figure 8.

```
[ar-siddharth.sindhur@RINMAC226 ~ % minikube start
😄  minikube v1.7.3 on Darwin 10.15.3
✨  Using the hyperkit driver based on existing profile
⌛  Reconfiguring existing host ...
🏃  Using the running hyperkit "minikube" VM ...
⚠️  Node may be unable to resolve external DNS records
⚠️  VM is unable to access k8s.gcr.io, you may need to configure a proxy or set --image-repository
🐳  Preparing Kubernetes v1.17.3 on Docker 19.03.6 ...
🚀  Launching Kubernetes ...
🌟  Enabling addons: default-storageclass, storage-provisioner
🏄  Done! kubectl is now configured to use "minikube"
[ar-siddharth.sindhur@RINMAC226 ~ % kubectl version
Client Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.3", GitCommit:"06ad960bfd03b39c8310aaf92d1e7c12ce618213"
, Compiler:"gc", Platform:"darwin/amd64"}
Server Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.3", GitCommit:"06ad960bfd03b39c8310aaf92d1e7c12ce618213"
, Compiler:"gc", Platform:"linux/amd64"}
```

Figure 8 Minikube - Cluster

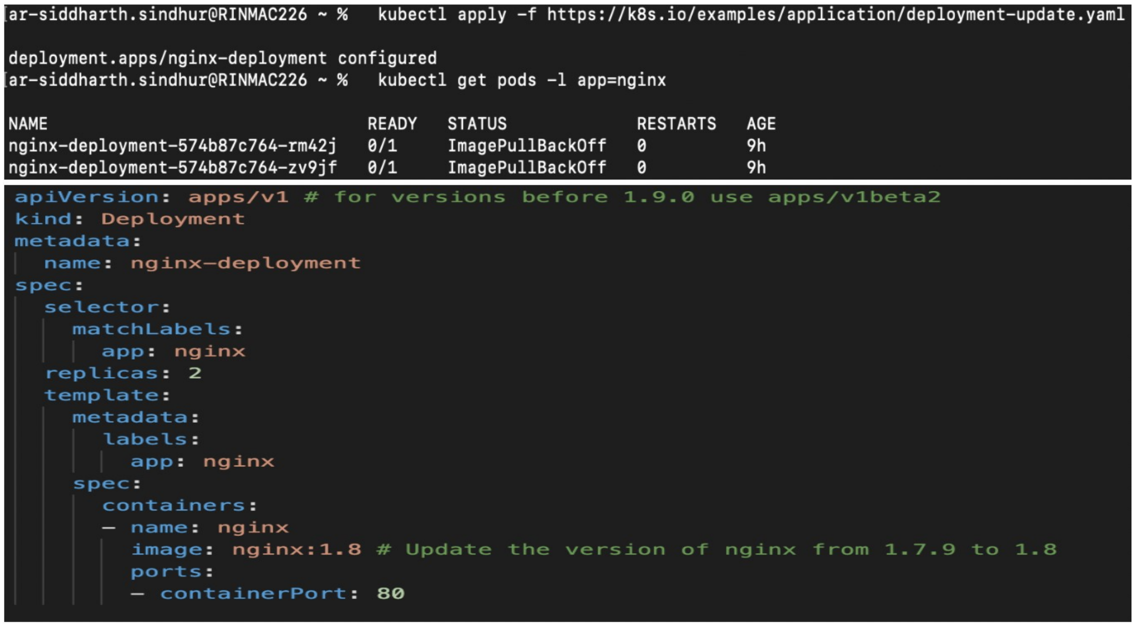
Create deployment for **deployment.yaml** file and verify the description as shown in figure 9.

```
[ar-siddharth.sindhur@RINMAC226 ~ %  kubectl apply -f https://k8s.io/examples/application/deployment.yaml

deployment.apps/nginx-deployment created
[ar-siddharth.sindhur@RINMAC226 ~ %  kubectl describe deployment nginx-deployment

Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Sat, 16 May 2020 12:43:47 +0530
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
                        kubectl.kubernetes.io/last-applied-configuration:
                          {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},"name":"nginx-deployment","namespace":"default"},"spec":{"replica...
Selector:               app=nginx
Replicas:               2 desired | 2 updated | 2 total | 0 available | 2 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx:1.14.2
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      False   MinimumReplicasUnavailable
  Progressing    True    ReplicaSetUpdated
OldReplicaSets:  <none>
NewReplicaSet:   nginx-deployment-574b87c764 (2/2 replicas created)
Events:
  Type    Reason             Age        From                   Message
  ----    ------             ----       ----                   -------
  Normal  ScalingReplicaSet  <invalid>  deployment-controller  Scaled up replica set nginx-deployment-574b87c764 to 2
```

Figure 9 Deployment

iJETRM
International Journal of Engineering Technology Research & Management

Update the deployment with different docker image version by creating new .yaml file as shown in figure 10

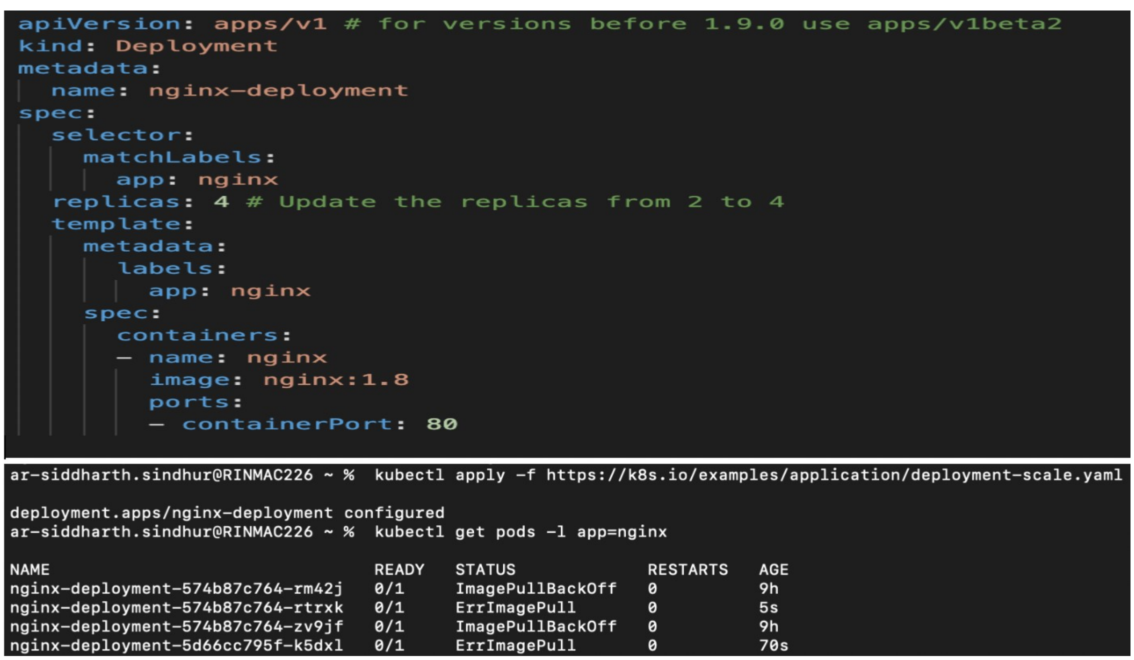


```
ar-siddharth.sindhur@RINMAC226 ~ %  kubectl apply -f https://k8s.io/examples/application/deployment-update.yaml

deployment.apps/nginx-deployment configured
ar-siddharth.sindhur@RINMAC226 ~ %  kubectl get pods -l app=nginx

NAME                                READY   STATUS             RESTARTS   AGE
nginx-deployment-574b87c764-rm42j   0/1     ImagePullBackOff   0          9h
nginx-deployment-574b87c764-zv9jf   0/1     ImagePullBackOff   0          9h
```

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.8 # Update the version of nginx from 1.7.9 to 1.8
        ports:
        - containerPort: 80
```

Figure 10 Deployment-update

Scale the application by changing the no. of replicas in the **deployment-scale.yaml** file as shown in figure 11.



```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 4 # Update the replicas from 2 to 4
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.8
        ports:
        - containerPort: 80
```

```
ar-siddharth.sindhur@RINMAC226 ~ %  kubectl apply -f https://k8s.io/examples/application/deployment-scale.yaml

deployment.apps/nginx-deployment configured
ar-siddharth.sindhur@RINMAC226 ~ %  kubectl get pods -l app=nginx

NAME                                READY   STATUS             RESTARTS   AGE
nginx-deployment-574b87c764-rm42j   0/1     ImagePullBackOff   0          9h
nginx-deployment-574b87c764-rtrxk   0/1     ErrImagePull       0          5s
nginx-deployment-574b87c764-zv9jf   0/1     ImagePullBackOff   0          9h
nginx-deployment-5d66cc795f-k5dxl   0/1     ErrImagePull       0          70s
```

Figure 11 Deployment-scale

iJETRM
International Journal of Engineering Technology Research & Management

Application Performance Testing using JMeter

JMeter is used to test the performance of application when subjected to increase in Query-per-second (QPS). It will show all the necessary parameters required to analyze the performance of the web application.
Firstly, we add Blazemeter extension to browser and start recording as shown in figure 12. As the application loads, it creates a .jmx file which records all the paramters.

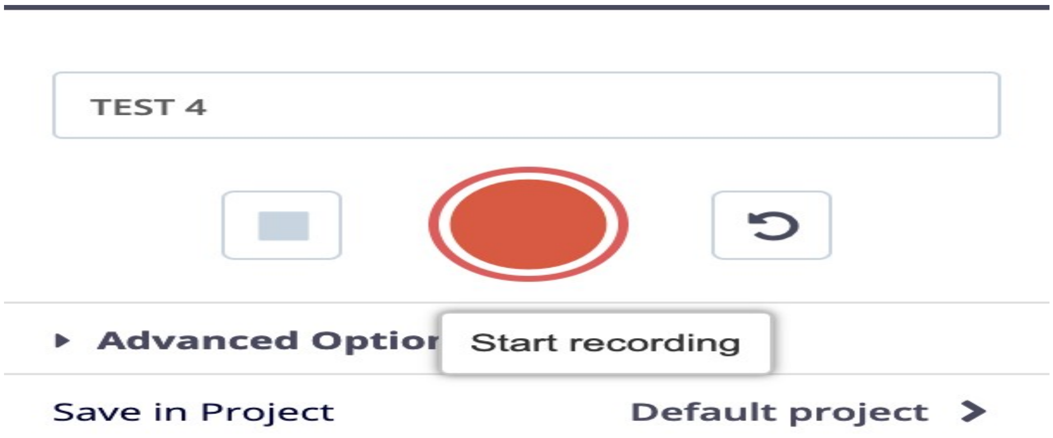


Figure 12 Blazemeter

Give a name to the project and click on **Save in project** to save the file.
Then, open Apache JMeter and open that .jmx file and add different parameter to be tested as shown in figure 13

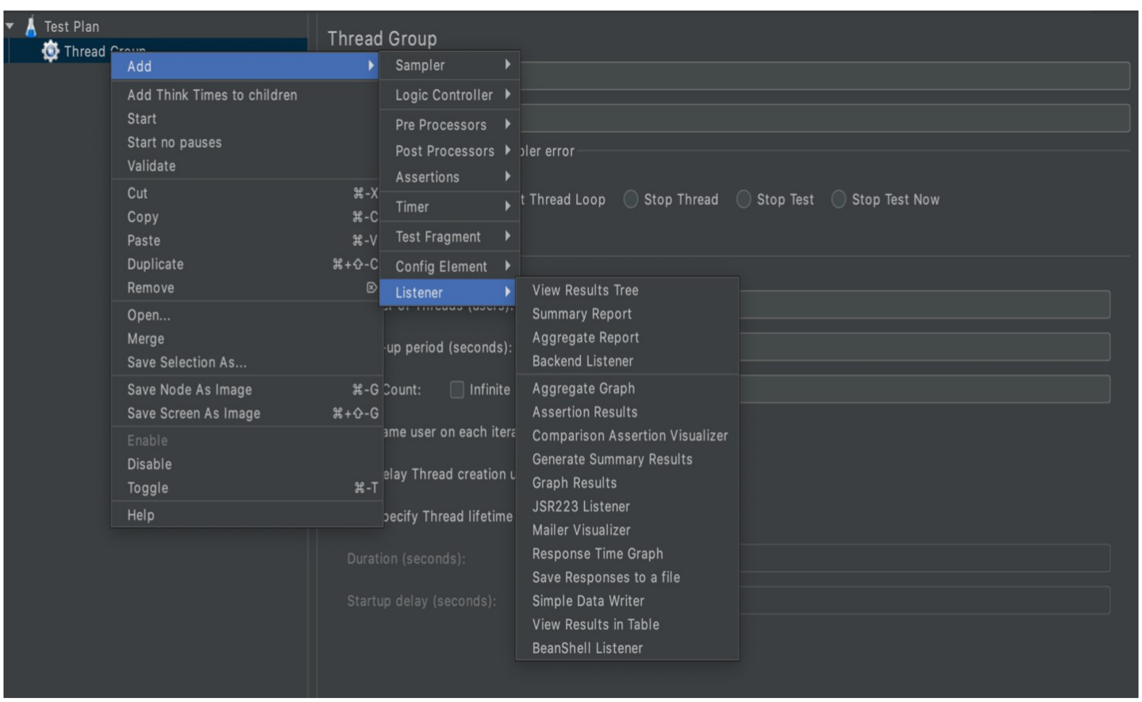


Figure 13 JMeter Listeners

iJETRM
International Journal of Engineering Technology Research & Management

Go to **Thread Groups** and change the no. of users accordingly as shown in figure 14. Make sure it's a gradual increase or else if a high no. is given the test fails. As the website crashes due to excessive load.

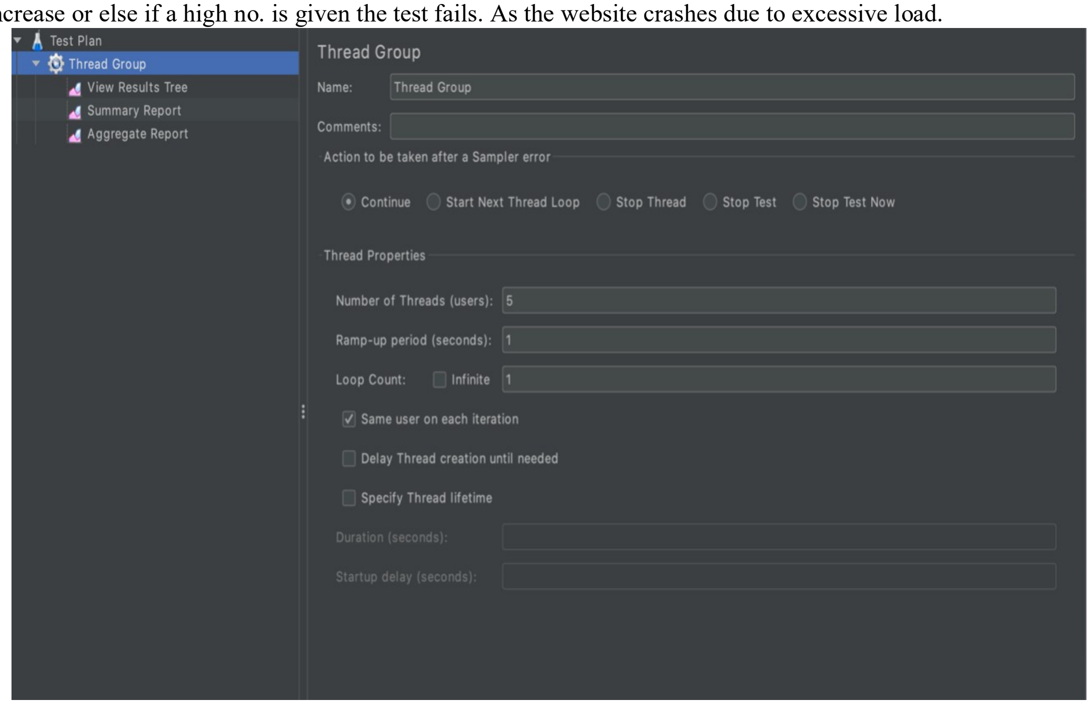


Figure 14 JMeter Thread Groups

Then Click on run and check the results obtained in Aggregate Report as shown in Figure 15.

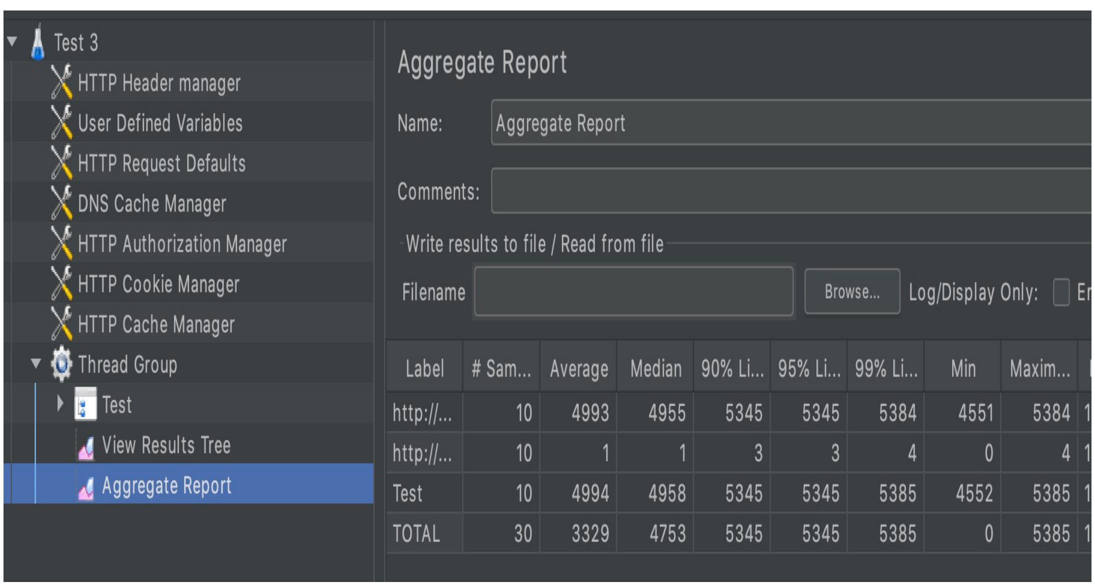


Figure 15 JMeter Aggregate Report

iJETRM
International Journal of Engineering Technology Research & Management

Similarly run the simulation for different no. of users a note down the required parameters as shown in table1.

No. of Users	Avg. Response time (ms)	Throughput	Received KB/s
5	5723	38.9/min	33.57
10	6583	58.4/min	50.38
15	23573	31.1/min	12.99
20	54571	12.3/min	8.58

Table 1

Container-based infrastructures and microservices provide a software delivery frontier, generating tremendous opportunities for companies pursuing massively scalable, mobile, and distributed applications. They have recently began standardizing on Kubernetes as a single target architecture, creating incentives for aligning DevOps activities around a common target for deployment.

Kubernetes is a software, which is open source, that allows containerized systems to be deployed, scaled, and managed. As an orchestrator, it handles the function of scheduling containers on a cluster, and also manages the workloads to ensure they run as expected. Because Kubernetes was developed from the outset with the concept of working together software development and operations, operational tasks are an integral component of Kubernetes architecture and design, and how they are performed. In Kubernetes virtually all uses declarative constructs that explain how applications are designed, how they communicate and how they are managed.

For JMeter, UI listeners are perfect to debug / check with. Don't plan to touch large loads (> = users of 500 competitors), use them very sparingly. With very light loads, these listeners are configured to get metrics quickly when load tests are running within JMeter UI (<= 50 users participating). They may even be used for moderate load (100-300 concurrent users), but it is not expected to run distributed JMeter testing with JMeter UI. That's not the purpose. By default, Remember JMeter is set to 512 MB heap memory which is fairly low, while it will increase the memory allocated for JMeter, it feels water coping out of a boat that no longer floats.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragement helped me in completing the project successfully.

I acknowledge with great pleasure the guidance and support extended by **Dr. Bhagya R, Associate Professor**, Department of Telecommunication Engineering. I am extremely thankful to her for encouraging me and proving me with all the necessary support for developing a good system.

I would like to express my special thanks of gratitude to **Mr. Deepesh Reshi, Senior Software Engineer**, **Rakuten**, Bengaluru for mentoring me despite his busy schedule managing corporate affairs. His timely suggestions have contributed in bringing out this project.

I express my sincere thanks and appreciation to **Dr. K Sreelakshmi, Professor & Head of the Department**, Telecommunication Engineering, for her immense support and encouragement extended.

I thank **Dr. K N Subramanya, Principal**, for his help and support extended by providing the required facilities.

I express my gratitude and respect to all the staff members of the department who have helped me in the completion of the project.

iJETRM
International Journal of Engineering Technology Research & Management

CONCLUSION

With advancing cloud-technology, cloud services provided by public providers are becoming even more influential, guiding the cloud services ecosystem. Cloud infrastructure services are attractive in part because they allow customers to acquire and release on-demand infrastructure resources for applications as a response to load surges. The automated, efficient deployment of software is vital for the modern cloud providers nowadays.

Containers provide a framework for increasing performance by passing binaries to the operating system. This way helps to increase the host capacity and encourages patch management as there are less patching operating systems. A container keeps configuration file and its binaries for the application and stores the OS elements that are modified by the application.

Container-based technologies such as Docker allow micro-services hosting on Cloud infrastructures. These allow for bundling of applications and data in a way that enables their easy deployment and subsequent use. Docker is only one of many such solutions brought forward.

REFERENCES

[1] Marco Miglierina DEIB, *Application Deployment and Management in the Cloud*, 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 2014

[2] Benson, J.O., Prevost, J.J., & Rad, P. *Survey of automated software deployment for computationand engineering research*, 2016

[3] Hong He, *Applications Deployment on the SaaS Platform*, 5th International Conference on Pervasive Computing and Applications, 2010

[4] Manar Jammal, Ali Kanso, Parisa Heidari, Abdallah Shami, *Availability Analysis of Cloud Deployed Applications*, IEEE International Conference on Cloud Engineering, 2016

[5] Sachchidanand Singh, Nirmala Singh, *Containers & Docker: Emerging Roles & Future of Cloud Technology*, 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2016

[6] Hao Zeng, Baosheng Wang, Wenping Deng, WeiqiZhang, *Measurement and Evaluation for Docker Container Networking, International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, 2017

[7] Antonio Celesti, Davide Mulfari, Maria Fazio, Massimo Villari and Antonio Puliafito, *Exploring Container Virtualization in IoT Clouds*, IEEE International Conference on Smart Computing (SMARTCOMP), 2016.

[8] Coleman L, J. Sarran, Garegin Grigoryan, *Container Orchestration by Kubernetes for RDMA Networking*, IEEE 27th International Conference on Network Protocols (ICNP), 2019 .

[9] L. A. Vayghan, M. A. Saied, Maria Toeroe, Ferhat K., *Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned*, IEEE 11th International Conference on Cloud Computing, 2018.

[10] Jay Shah, Dushyant D., Building Modern Clouds: *Using Docker, Kubernetes & Google Cloud Platform*, IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), 2019.

[11] Biradar S. M., Dr.Shekhar R, Dr. A. P. Reddy, *Build Minimal Docker Container Using Golang*,IEEE 2nd International Conference on Intelligent Computing and Control Systems (ICICCS), 2018.

iJETRM
International Journal of Engineering Technology Research & Management

[12] Chia-Chen Chang, Shun-Ren Yang, *A Kubernetes-Based Monitoring Platform for Dynamic Cloud Resource Provisioning*, 2017.

[13] KittiKlinbua and WiwatVatanawood, *Translating TOSCA into Docker-Compose YAML File Using ANTLR*, 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2017.

[14] Paul Townend, Stephen Clement, *Improving Data Center Efficiency Through Holistic Scheduling In Kubernetes*, IEEE International Conference on Service-Oriented System Engineering (SOSE), 2019.

[15] Naik, Nitin. "Docker container-based big data processing system in multiple clouds for everyone." *2017 IEEE International Systems Engineering Symposium (ISSE)*. IEEE, 2017.