Test Infrastructure Management Platform to Launch Short Instances On Demand

Vismitha N, Prof. Geetha V

Student, Department of Information Science,
R.V College of Engineering, Karnataka, India

Assistant Professor, Department of Information Science,
R.V College of Engineering, Karnataka, India

ABSTRACT

A Data pipeline facilitates setting up various databases and data warehouses for analytics with minimal effort. Each of these data sources is to be tested by the QA team before release.These infrastructures should be made available for the Testing team as and when they demand. In order to automate the availability of these data sources, a Test Infrastructure Management Platform is developed that facilitates availability of these infra as short instances,on demand.When a group of people are working on the same application, there can be inconsistencies in operating systems,environment that the application runs on ,correct versions of the services etc. In order to overcome these drawbacks, infrastructures such as databases are made available within Docker containers as short lived instances, on demand.This facilitates qualifying releases for each kind of database and test feature branches.

Keywords :- *Docker, Dockerfile, Docker Image, Docker Container, Boto3, Jenkins, Django, Amazon ECS, Amazon ECR, AWS Fargate*

1. INTRODUCTION

A Data Pipeline integrates with a variety of databases , applications, and webhooks to help users replicate data from these heterogeneous sources to a destination of users' choice. A near-real time data integration platform simplifies data integration challenges in any data analytics project. One can set up databases or data warehouses for analytics with minimal effort.Each of these data sources are tested by the QA team,which is made available by the developers.

A platform is developed to facilitate a test infrastructure management system. Test infrastructure includes Test Database and Test instances.The data sources being tested can be called Test Databases.Test Database is intended for qualifying releases for each kind of database that are made available to the QA team ,by the developers as short instances On-Demand. Test Instances is intended for testing out feature branches.When the instance is started,the IP address, port on which the instance is running and other details are provided using which the QA team can carry out the testing.Each instance has an up-time period set. Once the time runs out,instances will be deleted.
This facilitates an automated integrated test infrastructure platform within the company to carry out testing and qualifying releases of the heterogeneous data sources.The subsequent sections of this paper gives a relation work in the respective areas, proposed solutions with procedure and results, concluding with future directions.

2. LITERATURE SURVEY

2.1 A. Ahmed, A. Mohan, G. Cooperman G.Pierre[1]

This paper explains the process of speeding up the response time of the applications proposed to snapshot the state of containers that are completely deployed and restart container instances in the future with the previously saved states of the application.

2.2 B. I. Ismail et al[2]

This paper describes how to overcome problems such as High latency, network bottleneck and network congestion. We can achieve this from moving centralized to decentralized paradigm, Edge computing will be able to reduce application response time for better user experience. Edge computing is enabled with Docker, a platform of container-based technology that has more advantages over VM based Edge computing. This paper mainly evaluates the fundamental requirements for EC that are 1) Deployment and Termination which mainly describes the platform that provides an easy way to manage, install and configure services to deploy the low- end devices. 2) Resource and Service Management that allows users to use the services even when the resources are out of limit. 3) Fault Tolerance which relies on the High availability and reliability to the user.

2.3 Bashari Rad, Babak & Bhatti, Harrison & Ahmadi, Mohammad [3]

This paper explains about an open platform that can be used for building, distributing, and running applications in a portable, lightweight runtime and packaging tool, known as Docker Engine and the Docker Hub, which is a cloud service for sharing applications.

2.4 Casalicchio, Emiliano[4]

This paper explains about the cloud application issues and how container technology can solve them. for example the application portability problem and the virtual machine performance overhead problem. One of many research challenges include container orchestration, which makes it possible to define the procedure to select, deploy, monitor, and dynamically control the configuration of multi-container packaged applications in the cloud. This paper presents the problem of autonomic container orchestration,analyzes the state of the art solutions and discusses open challenges.

2.5 D. N. Jha, S. Garg, P. P. Jayaraman, R. Buyya, Z. Li and R. Ranjan[5]

This paper provides Experimental study on the performance evaluation of Docker containers running heterogeneous sets of microservices concurrently,conducting a comprehensive set of experiments following CEEM (Cloud Evaluation Experiment Methodology) to measure the interference between containers running either competing or independent microservices.

2.6 Fawaz Paraiso, Stéphanie Challita, Yahya Al-Dhuraibi, Philippe Merle[6]

This paper focuses on management of docker containers, mainly where users finds low-level system issues and it describes how modelling Docker containers helps to achieve sustainable deployment and management of Docker containers.This paper has presented a model-driven approach for verification and synchronization and its future work will investigate atomic changes performed in managed containers architecture.

2.7 Nüst D, Sochat V, Marwick B, Eglen SJ, Head T, Hirst T, et al. [7]

This paper discusses the set of rules that are followed by researchers to write sensible Dockerfiles for typical workflows in the field of data science. These rules listed in this paper are intended to help researchers, especially aspirants who are new to the concept of containerisation, how to use containers for open and effective scholarly communication and collaboration while avoiding the drawbacks in a research lifecycle.

2.8 P. Rai, Madhurima, S. Dhir, Madhulika,A. Garg[8]

This paper describes an open source continuous integration tool: Jenkins, which is on the whole a server-oriented arrangement that runs in a servlet-like container. It supports various Source Control Management tools including, Subversion, Mercurial, Perforce, Clear case and Rational Team Concert. The architecture, functions, and applications of Jenkins are put forth in this paper. The aim of this paper is to emphasize on the Jenkins Integration Development Environment and five software integration tools wrt their effectiveness and their usability.

2.9 S. Singh and N. Singh[9]

This paper gives insights about Container-based virtualization which is a type of operating system virtualization and in this approach, the kernel of the operating systems runs on the hardware node
with different isolated guest virtual machines (VMs) called containers.

2.10 V. Armenise[10]

This paper illustrates the evolution of Jenkins from being a pure Continuous Integration Platform to a Continuous Delivery one, showcasing the new feature tendency where not only the build but also the automation of the release and the delivery process of the product. In this scenario Jenkins becomes the orchestrator tool for all the teams/roles concerned within the software lifecycle, due to which Development, Quality Assurance and Operations teams will work closely together. Goal of this paper is not only to prioritize Jenkins as a hub for Continuous Delivery, however conjointly introduce the challenges that also have to be compelled to be solved in order to enhance Jenkins' tracking capabilities.

3. MODEL METHODOLOGY

The proposed paper makes use of Docker containers to run the databases as a service. A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

Dockerfiles are created which is a simple text file with instructions on how to build Docker images. A Docker image is a file, composed of multiple layers, that is used to execute code in a Docker container. An image is essentially built from the instructions for a complete and executable version of an application, which relies on the host OS kernel. When the Docker user runs an image, it can become one or multiple instances of that container.

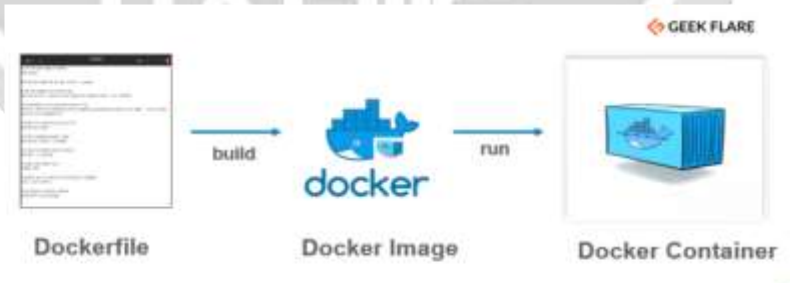


Fig 1*:* Building Docker Container

3.1 Docker Registry

A Registry is a stateless and scalable server side application that stores and distributes Docker images. This project uses AWS Elastic Cloud Registry(ECR) which is a managed container image registry service to manage ,store and deploy docker containers. Amazon ECR provides a registry which is secure, scalable, and reliable,to store Docker images or Open Container Initiative (OCI) images.

Amazon ECR supports resource-based permissions through IAM ,where private repositories can be accessed by specific users or Amazon EC2 instances. One can eliminate manual or in house container registry management,by using ECR.

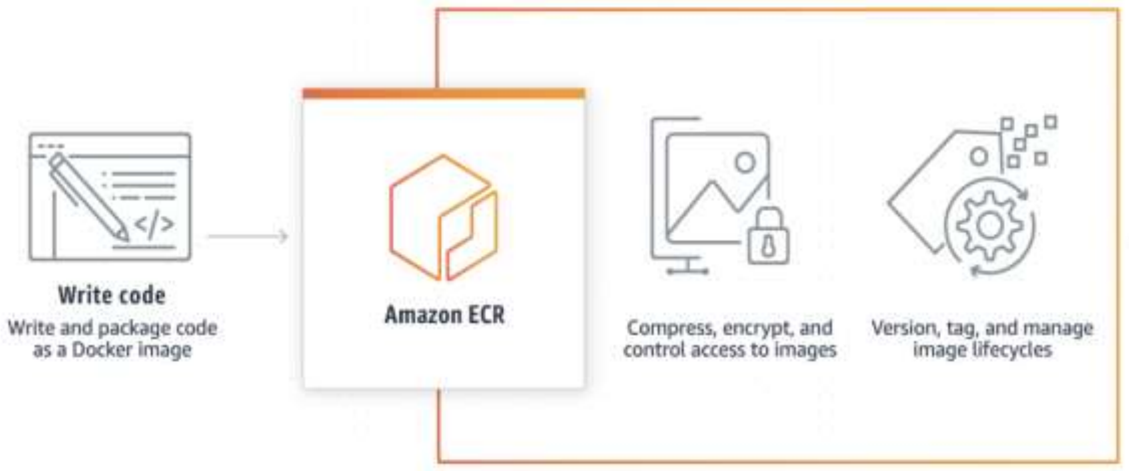

Write code
Write and package code as a Docker image
Amazon ECR
Compress, encrypt, and control access to images
Version, tag, and manage image lifecycles

Fig 2*:* AWS Elastic Cloud Registry

3.2 AWS Fargate

AWS Fargate [Container as a Service (CaaS) technology] is a serverless compute engine that hosts containers working with both Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS).By removing the need for infrastructure management, AWS Fargate reduces development costs on new projects by 75%.

Fargate facilitates users to run containers without the need to monitor the infrastructures, servers or clusters of Amazon EC2 instances.
Users need to specify the resources that every container requires, and Fargate will handle the rest for the users.

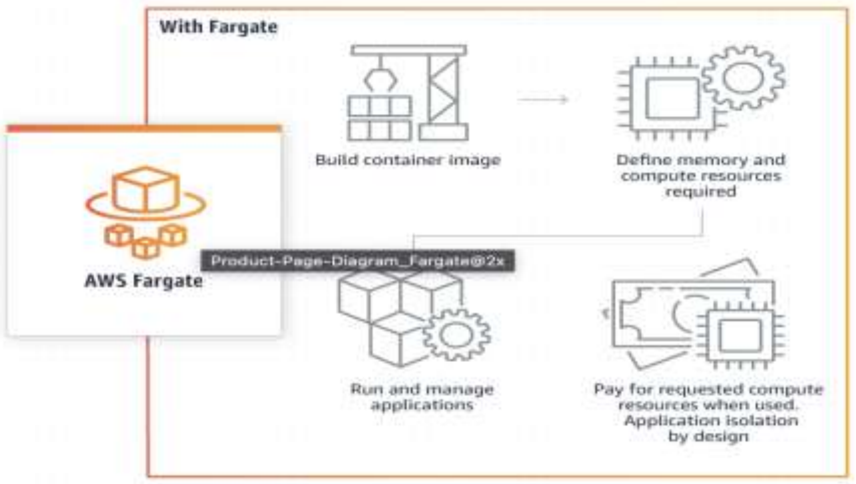

With Fargate
AWS Fargate
Build container image
Define memory and compute resources required
Product-Page-Diagram_Fargate@2x
Run and manage applications
Pay for requested compute resources when used. Application isolation by design

Fig 3*:* AWS Fargate

4. MODEL IMPLEMENTATION

4.1. Creation of Docker containers

Database Management Systems like MySQL, Postgres, Percona etc. are fed with Sample Databases.Each of these data sources are tested by
the QA team,which is made available by the developers in the form of Docker containers.

A generic docker file is created from the Base images of Databases which assembles the docker images.A container is then created specifying the port address and other parameters like mount point or binding volume.Docker Machine is used to create container services. A Repository is then created to store and manage these docker files, on AWS ECR. The docker images are pushed into the repository.

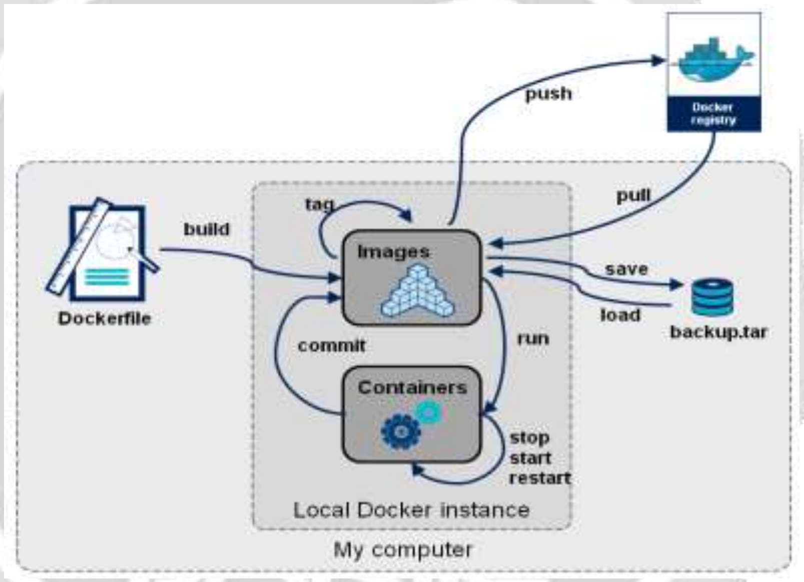


Fig 4 : Creation of Docker containers and pushing to ECR

4.2. Launching Docker Container

The docker containers thus built are launched on AWS Fargate orchestration service and made available to the end users. A ECS cluster is required which is a group of container instances.The resources required to run services/applications inside Docker containers are allowed to the cluster. A task is then defined that specifies the container information. An ECS service specifies the number of task instances to be created. A security group is added to the service,specifying the Inbound and Outbound port and Ip address. This service is launched using AWS Fargate which is a serveless container orchestration application that automates resource scalability. The Public IP address, port on which the services inside the docker containers are running,along with other information, is displayed to the user.

4.3. APIs and Db Status

APIs are created using the Django REST framework. APIs let users interact with the platform through HTTP. List of APIs include:

1. List all of the Databases with status
2. List all of the database types
3. Get information about a database
4. Start a database or extend end-time if already started
5. Stop a database

Available status of a database resource.

1. pending_start: A DB resource was requested by a user.
2. starting: In the process of starting the DB.
3. config_pending: DB infra has been allocated and waiting for DB to come online.
4. started: DB is online and can be used.
5. pending_stop: A DB resource termination has been requested.
6. stopping: In the process of stopping the DB.
7. stopped: The DB has been Stopped.
8. failed: There was an error while starting or stopping the DB check the error message.

4.4. Testing

Pytest Fixtures are used for testing the model. Pytest fixtures are functions attached to the tests which run before the test function is executed. Fixtures are used to feed some data to the tests such as database connections, URLs to test and some sort of input data. Therefore, instead of running the same code for every test, one can attach a fixture function to the tests which runs and returns the data to the test before executing each test.Pytest fixture function is automatically called by the pytest framework when the name of the argument and the fixture is the same.

Two fixture functions are used:

1. To populated Test databases with database type, status , start and end time.

2. To populate Test Instances with instance name, base IP address, status, start and end time.

4.5. Deployment

Once the model has been tested on a local machine , the proposed system is exposed via REST .The RESTful interface is created by exposing an API using the Django framework. A generic endpoint of each of the APIs is created. The Databases are deployed on AWS Fargate by Jenkins.

Jenkins is an open source automation server. It helps automate the process of software development associated with building, testing, and deploying, facilitating continuous integration and continuous delivery. A Jenkinsfile is created that describes the steps needed for running a Jenkins pipeline.A Pipeline is a group of events interlinked with each other in a sequence.Group of events include Cloning sample data,Building Docker images,Pushing images to ECR and creating ECS task definitions.Boto3 is a AWS Software Development Kit (SDK) for Python,that allows to write software which makes use of services like Amazon ECR and Amazon ECS.APIs provided by Boto3 are used to interact and operate AWS ECR and Fargate.Figure below shows the Deployment architecture.

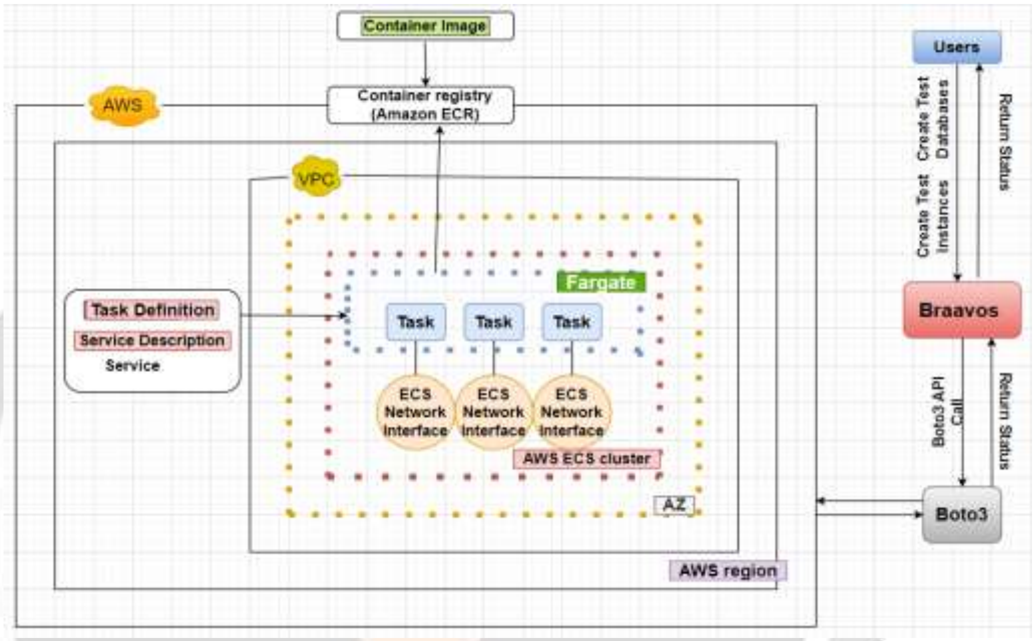
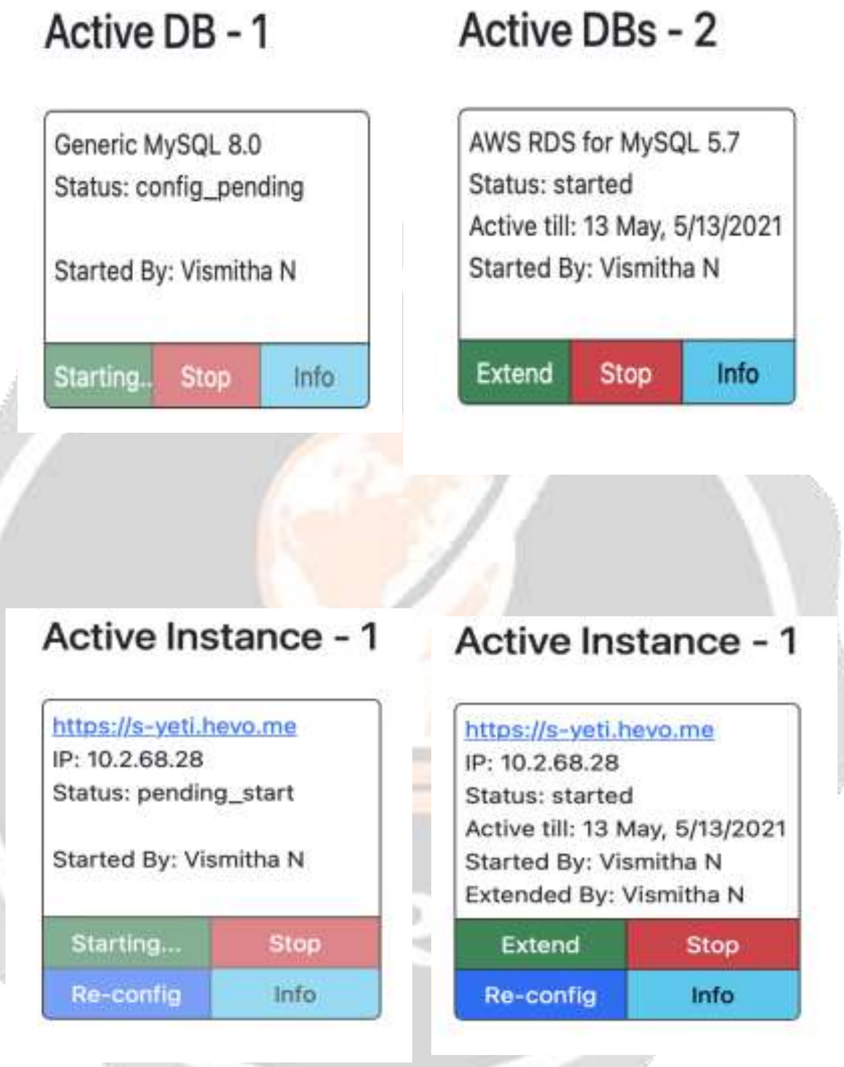


Fig 5 *:* Deployment Architecture

4.6. Results

The features of the final model includes:

1. Automation of Testing and Qualifying phases of Product Release by providing an Infrastructure platform.
2. Authentication of users by providing OAuth Key and Secret used to make API calls.
3. APIs to interact with Test Databases and Instances.
4. Provides credentials of databases like Port number on which the service is running, username and password to login and the IP address of the container in which the service is running.
5. Ability to Extend the uptime of Test Databases and Instances to maximum of 3hrs from the current time.
6. Simultaneously access of Test Databases by multiple users. Hence supporting the multi-user model.
7. Provides status of database resources like pending_start, config_pending, starting, stopped, failed etc.



Active DB - 1

Generic MySQL 8.0
Status: config_pending

Started By: Vismitha N

Active DBs - 2

AWS RDS for MySQL 5.7
Status: started
Active till: 13 May, 5/13/2021
Started By: Vismitha N

Active Instance - 1

https://s-yeti.hevo.me
IP: 10.2.68.28
Status: pending_start

Started By: Vismitha N

Active Instance - 1

https://s-yeti.hevo.me
IP: 10.2.68.28
Status: started
Active till: 13 May, 5/13/2021
Started By: Vismitha N
Extended By: Vismitha N

Fig 6 *:* Active Test Databases and Active Test Instances

5. CONCLUSION

In a present scenario,creation and running of infrastructures on Cloud orchestration platform by the QA team,is a manual process which includes providing the credentials with restricted access, checking on the uptime of instances etc.It is a tedious process to monitor these infrastructures manually as it is used at high scale by organisation.

In an enterprise that integrated 100+ data sources and data warehouses, testing of each of them is the primary and continuous goal of the QA team. A management that facilitates operating all the required infrastructures in a single platform makes it easier for both Dev and QA team within the organisation.

As the data sources are available as Docker containers, it minimizes the inconsistency between different environments,configurations, accelerates testing setup and simplifies tasks for DevOps team by standardizing the configuration interface and makes operational setup simpler.

6. REFERENCES

[1]A. Ahmed, A. Mohan, G. Cooperman and G. Pierre, “Docker Container Deployment in Distributed Fog Infrastructures with Checkpoint/Restart,” 2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2020, pp. 55-62, doi: 10.1109/MobileCloud48802.2020.00016.

[2]B. I. Ismail et al., "Evaluation of Docker as Edge computing platform," 2015 IEEE Conference on Open Systems (ICOS), 2015, pp. 130-135, doi: 10.1109/ICOS.2015.7377291.

[3]Bashari Rad, Babak & Bhatti, Harrison & Ahmadi, Mohammad. (2017). “An Introduction to Docker and Analysis of its Performance,” IJCSNS International Journal of Computer Science and Network Security. 173. 8.

[4]Casalicchio, Emiliano. (2017). “Autonomic Orchestration of Containers: Problem Definition and ResearchChallenges,”10.4108/eai.25-10-2016.2266649.

[5]D. N. Jha, S. Garg, P. P. Jayaraman, R. Buyya, Z. Li and R. Ranjan, "A Holistic Evaluation of Docker Containers for Interfering Microservices," 2018 IEEE International Conference on Services Computing (SCC), 2018, pp. 33-40, doi: 10.1109/SCC.2018.00012.

[6]Fawaz Paraiso, Stéphanie Challita, Yahya Al-Dhuraibi, Philippe Merle. “Model-Driven Management of Docker Containers,” 9th IEEE International Conference on Cloud Computing (CLOUD), June 2016, San Francisco, United States. pp.718 - 725, 10.1109/CLOUD.2016.0100 . hal-01314827

[7]Nüst D, Sochat V, Marwick B, Eglen SJ, Head T, Hirst T, et al. (2020) “Ten simple rules for writing Dockerfiles for reproducible data science,” PLoS Comput Biol 16(11): e1008316. https://doi.org/10.1371/journal.pcbi.1008316.

[8]P. Rai, Madhurima, S. Dhir, Madhulika and A. Garg, "A prologue of JENKINS with comparative scrutiny of various software integration tools," 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015, pp. 201-205.

[9]S. Singh and N. Singh, "Containers & Docker: Emerging roles & future of Cloud technology," 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2016, pp. 804-807, doi: 10.1109/ICATCCT.2016.7912109.

[10]V. Armenise, "Continuous Delivery with Jenkins: Jenkins Solutions to Implement Continuous Delivery," 2015 IEEE/ACM 3rd International Workshop on Release Engineering, 2015, pp. 24-27, doi: 10.1109/RELENG.2015.19.