AUTOMATING INFRASTRUCTURE CREATION, BUILD, TEST, DEPLOY APPLICATIONS TO CLOUD USING DEVOPS

[1]MOHAMMAD IMRAN ANSARI, [1]YOGINI CHOUDHARI, [2]SRIVARAMANGAI. R

[1] University Department of Information Technology, University of Mumbai, Mumbai 400008, Maharashtra, India
[1] University Department of Information Technology, University of Mumbai, Mumbai 400008, Maharashtra, India
[2] University Department of Information Technology, University of Mumbai, Mumbai 400008, Maharashtra, India

Email: [1]imranonline2@yahoo.co.in , [1]chaudhariyogini2@gmail.com, [2]rsrimangai@udit.mu.ac.in

Abstract: Technology is growing at an exponential rate beyond human manageable abilities, In order to keep the pace with every growing industry we also need to make sure the practices which we follow are error free and well organized and governed which will lead to faster time to market, secure deployments, planned iteratively and well tested automatically where less human intervention is needed. It is seen that we are using a lot of toolsets for every stage of our deployment pipeline while deploying our applications, but to make it a smooth and error free deployment proper planning and choosing the right tool for every stage is very important. It has been seen that products which are much better in terms of quality fail to perform well as they don't reach the market in time, features which are game changers don't reach the audience in time which give advantage to our competitors to grow. This research work is an attempt to find the best suitable method of deployment which will be secure cloud migration by analyze the existing research literature, comparing the techniques. In this paper. Thus, in this paper, first, we have tried to review the possible toolsets and concepts which we can adopt and solve our problems from the existing literature. Next, we focus on using the specific toolsets like Microservices and CI/CD with each of our stages in our deployment pipeline. Following that we are discuss the deployment strategies and have listed the whole pipeline together and see how the tools fits in it. submit our conclusions with respect to investigating which is the best and the fastest way to securely deploy to cloud with an MVP (Minimal Viable Product).

Keywords: DevOps, Agile, Continuous Integration, Continuous Testing, Microservices, Cloud

1. INTRODUCTION

Every organization is running behind moving to cloud, but just moving to cloud is not the only challenge when the transformation takes place. The challenge is to make sure how do you leverage being on Cloud, how do you automate every process which makes the continuous delivery and easy deployment, faster response, well tracked and less bugs. The aim of this work is to reduce the amount that organizations are stumbling in cloud deployment due to various factors such as the changes that need to be made, the people involved which requires cloud expertise, the processes that needs to be followed and the last one is the technology itself. During the transition, though the first three factors can be dealt with, the most important thing is the technological tools that we will use to deploy since the other factors are dependent on the choice of these tools. When we make applications for cloud we also need to plan accordingly and make use of microservices approach to efficiently scale our applications on one hand and save cost on the other hand. Agile and DevOps go hand in hand so being Agile solves majority of challenges which we will see further in this paper. Automatically creating and destroying infrastructure using IAC (Infrastructure as Code) tools would make our provisioning process fast and well governed. Focus on being cloud agnostic to make sure we can hop on any cloud without spending a penny. And finally focusing on the type of deployment making sure we get the high availability all the time.

For a better understanding the paper has been structured covering the various aspects of our work as section 1: Introduction, section 2: The DevOps techniques, section 3: Microservices, section 4: describing the Continuous Integration & Continuous Deployment and section 5: Describing various Deployment strategies. 6: Project Structure

2. DETAILS LEARNING OF DEVOPS TECHNIQUES

DevOps is a combination of Dev + Ops, which is helping us to solve the problems of these two teams working in Silos and thinking about a combine outcome futuristic solution. It is also known to build a culture change which will help the organization deliver faster with stable deployments. DevOps Techniques and tools have a large landscape, with the daily growth of new technologies and requirements to automate more and more new toolsets and new tools are coming into market which makes the process more automated and less of human intervention is needed. The DevOps Techniques gives you the real factor cloud as an excellent infrastructure for DevOps deployment to bust cloud business efficiency to maximize profit while maintaining quality service delivery[4]. E-Business are totally relied on the DevOps Deployment to faster time to market. In the Survey paper, Lets first have a look at how DevOps Process and techniques are placed together. Muhammad Owais Khan, Awais Khan Jumani*, Farhan, Waqas Ahmed Siddique and Asad Ali Shaikh have explained in the research Paper "Fast Delivery, Continuously Build, Testing and Deployment with DevOps Pipeline Techniques on Cloud" have focussed on how automating everything using DevOps techniques gives us better and faster results[5]. Focussing on continuous Integration and continuous Delivery Implementation on Cloud. There has been recommendation to use Infrastructure as Code practices as one of the DevOps Techniques to have stable environments post deployments[6]. Deploying on Same configuration of servers avoid complexity and discrepancy. It's also found that the Software Quality is improved significantly after adopting devops techniques, beautifully explained in "Improve Software Quality through practicing DevOps Automation" by Sikender[11].

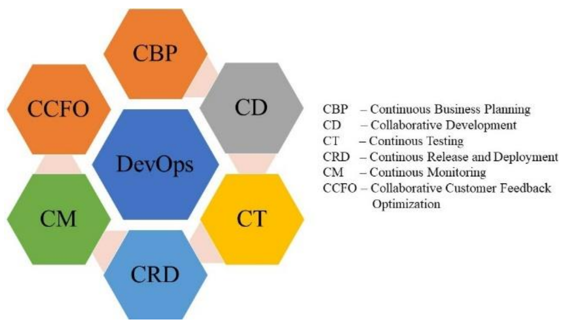

CBP
CD
CT
CRD
CM
CCFO
DevOps
CBP – Continuous Business Planning
CD – Collaborative Development
CT – Continous Testing
CRD – Continous Release and Deployment
CM – Continous Monitoring
CCFO – Collaborative Customer Feedback Optimization

Figure 1: DevOps Cycle

Four considerations are significant when competing with the elite of the DevOps industry [12]. They are:

i. Lead time
ii. Frequency of release
iii. Time for recovery
iv. Change rate shift

Spotify: A Case Study with Containerization using DevOps Technique - Spotify is a digital music service that uses microservice architecture which delivers a whole host of features like music streaming, logging in through social media credentials globally. This application keeps 60 million subscribers happy with their ultimate music streaming capability. Spotify enables microservices through containerization of a bundle of requests so that a single request will provide all the information through its interface, which helps the user not to make any repeated requests to match specific information. Spotify has been using this technology since 2014 and has been a key pillar for strategical scalability [27].

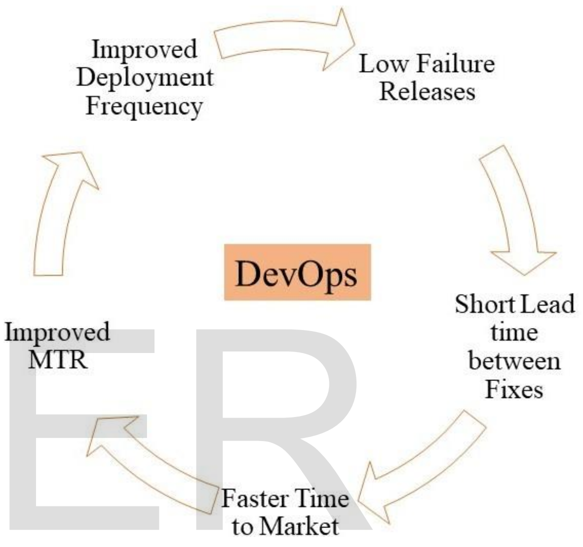


Figure 2: DevOps Metrics

3. MICROSERVICES

Microservice is a collection of small services that work together to fulfil the business requirements. In this section we discuss overview, its benefits, drawbacks, and characteristics [24]. Microservices approach is a term coined for breaking our application in to meaningful pieces which can be loosely coupled and can make sure our whole application doesn't go down at once and at the same time save some cost. Microservices can be considered as one distributed set of applications which can be deployed at many places and still work as a single unit but in terms of high. Availability and scaling its easy. We can take an example of Lambda functions, a Login module of an application can be deployed to Lambda and can save cost as its only needed when the user login and later it's not used, Lambda will only charge of its usage of login function but not hosting of the login function code. Archana Yadav Chikitsak Samuha's S. S" in their research paper "Containerized Microservices architecture"[15] has completely contemplates microservices building plan alongside the different points of interest and impediments of containerized microservices, objectives and the latest technology used. It has also been found in their research that how containers fit the need of microservices and how lightweight it is. How it separates the duties of each component. Goutam Kamat, S.G Raghavendra Prasad in their Research paper "Docker & Containers, The Future of Microservices" [24] has mentioned about the micro-services architecture and how docker will help to resolve the challenges in micro-service architecture. Portability of applications are also an important factor when we consider containerization as a Microservice tool. Efficiency of resource usage is also a reason why docker is preferred as a best for microservices [24].

Docker: A Container run time which helps us to host our application as a container. Docker permits you to run compartments. A holder is a sandboxed procedure running an application and its conditions on the host working framework. The application inside the compartment believes itself to be the main procedure running on the machine while the machine can run different holders freely [15]. Now we have our Docker Image which has our application, but the most important thing is to make it run in an organized manner which can help us scale and expose it to the world, we need an Orchestrator. Container Orchestrator tools like Kubernetes and Docker Swarm. **Kubernetes** provides more accessible, more efficient, and faster to convey administrations and applications [17].

Infrastructure as Code :

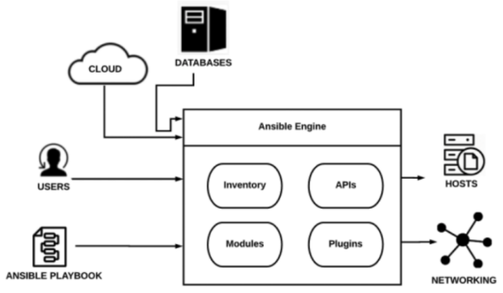


Figure. 2: Ansible Architecture [13]

Majority of the issue which developers faced where different type of infrastructure configuration causes most of the issues, This happens due to Human err, Many a time we fail to update few components, are build agents are not same, our server configurations are different, may be with different versions which doesn't make the output always as expected. IAC is getting popular these days in DevOps tool. The infrastructure is configurable and reusable with use of high-level language in Terraform It provides the facility to create a blue-print of infrastructure and can be versioned too [18]. IAC helps us to automate the manual process of Infrastructure setup and configuration. Manual and Repetitive – In manual base, there are multiple server, multiple instance, and the business never can grow. Change not Tracked – How we do track the changes, we need to understand what change of the last version. In manual way it is very difficult to track the changes [5]. It replaces thousands of clicks in one single command which does exactly what its instructed to do. The infrastructure spined up using IAC tools are less buggy and are of same as what the configuration says. It can happen that a Human forgot to select an option while installing a tool or configuring a server but when we document it as source code it will always be applied by the tool or the machine.

IAC can be classified as Provisioning and Configuration management.

For provisioning we can use Cloud Formation, Terraform etc.

Terraform is an infrastructure as code service/tool that allows user to build, change, and version infrastructure with safety and efficient manner [18].

Ansible is an IT automation tool. It can configure systems, run software, and configure advanced IT functions such as continuous deployment or non-rollback updates. The main objectives of Ansible are simple and easy to use [23] .

For Configuration management we can use Ansible which is most widely used

4. CONTINUOUS INTEGRATION / CONTINUOUS DEPLOYMENT

Cloud Agnostic is a very important factor in these days, As the Cloud Demand Surged, many vendors started selling their services which have mostly similar functionality but different names, in some cases different architectures and platforms. Now, if we take an example of an application which is hosted on Microsoft Azure and if I want to move it to AWS, it does take some efforts to move and migrate the application which might also not possible in some cases where we are using a native service/application of a Cloud Vendor. So, we need to plan our deployment models and process in such a way that it's not locked to a specific vendor, it should be movable from one cloud to another with minimal code change and should help us navigate faster in the crowd of vendors while using the best offerings from each of them.

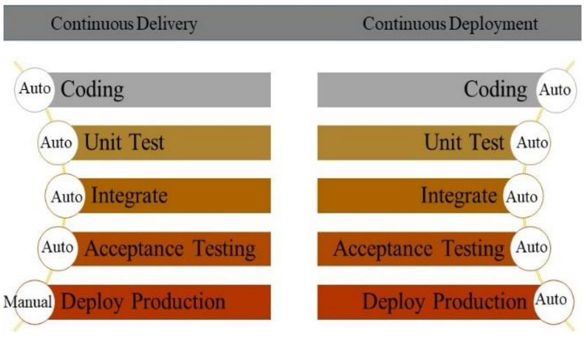


Figure 3: Continuous delivery and deployment

CI/CD Tools : Delivering and deploying a new release manually is error-prone and can lead to delays and additional expenses[3]. CI/CD Tools helps us in designing the complete Build and Deployment Pipeline or a Release Pipeline We have a variety of CI/CD tools in the market which can help in achieve our goal to build and deploy our application anywhere. We can design and control the security aspects, Audit, Approvals and Notifications etc. Jenkins is the most used Open-source tool for CI/CD with variety of plugins and Features which are built and u
pdated regularly. GITLAB is also emerging as a strong player in the Market, Azure DevOps on the other hand has 2 offerings one is Azure DevOps other is GITHUB Actions mostly based on YAML configuration, and you can focus on the logic. Bamboo, TeamCity, Circle CI, Travis CI are few examples of CI/CD offerings [8].

5. DEPLOYMENT STRATEGIES

Deployment strategies plays a very important role, before we understand how we deploy our application let's talk about the deployment types

5.1 Blue Green: Blue Green Deployment is a deployment strategy which helps us to deploy on Active and Passive slots. In this type of setup, we have 2 sets of Servers one is active and other is inactive, you deploy your application to the Inactive slot, test it and then redirect the traffic to the inactive one so the live traffic starts getting served from the server and marks the active server as Inactive. This approach is good but its costly as we must maintain 2 sets of infrastructure and only one set is being used while we must pay for both the sets.

5.2 Canary: We have seen how big organizations release their product to a small set of public/audience and once they get positive feedback, they scale it slowly to everyone. Canary is the concept behind this. A deployment is done in % ratio like 10% or 20% and slowly scaled to 100% audience.

5.3 Rolling Updates: This is a concept in Kubernetes deployment by default where a new deployment is created and then the old one is deleted gradually; this helps us to move the traffic slowly and steadily from one version to other. Kubernetes does health check of the applications to ensure it does not kill all your machines at the same time if something goes wrong during the deployment Kubernetes rollbacks the change for us[1].

6. PROJECT STRUCTURE

In this Project we have 3 repositories

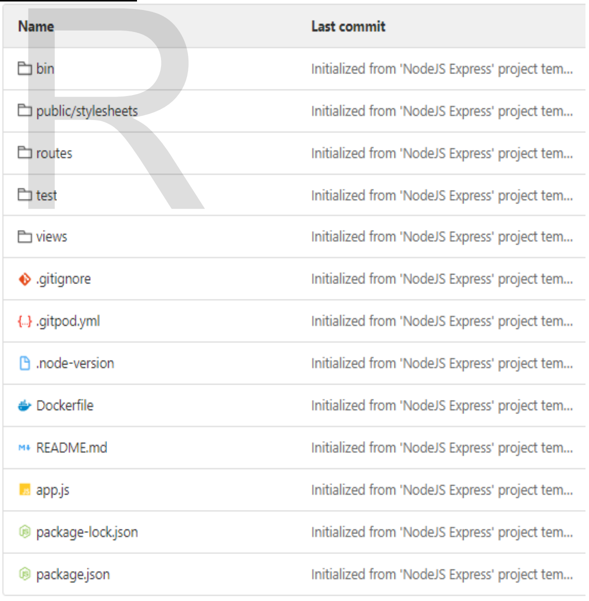
1. Microservice App using Node JS, containerized using Docker, deployed using Kubernetes

2. Infrastructure as Code Repo to create an AWS Environment

3. Infrastructure as Code Repo to create a GKE Environment

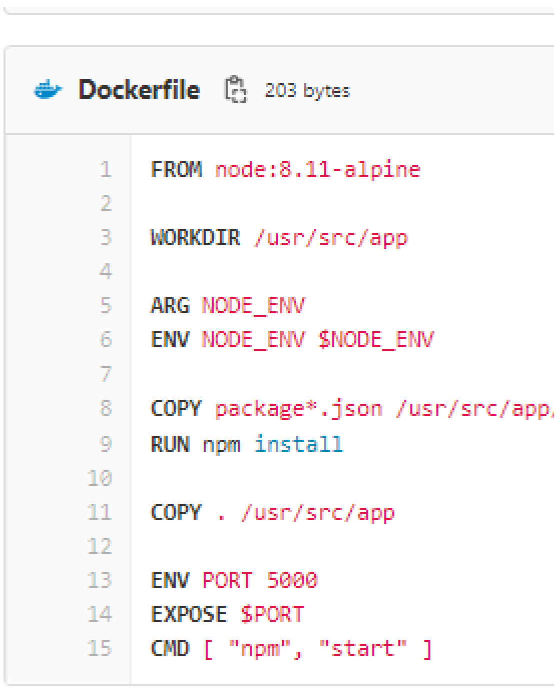
REPO 1: Microservice App

a) **Source Code for Microservice App using Node Js**



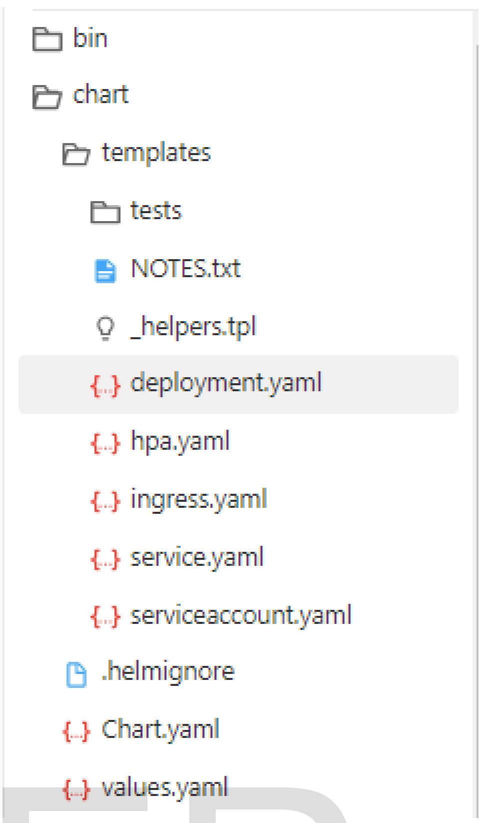
Name	Last commit
bin	Initialized from 'NodeJS Express' project tem...
public/stylesheets	Initialized from 'NodeJS Express' project tem...
routes	Initialized from 'NodeJS Express' project tem...
test	Initialized from 'NodeJS Express' project tem...
views	Initialized from 'NodeJS Express' project tem...
.gitignore	Initialized from 'NodeJS Express' project tem...
.gitpod.yml	Initialized from 'NodeJS Express' project tem...
.node-version	Initialized from 'NodeJS Express' project tem...
Dockerfile	Initialized from 'NodeJS Express' project tem...
README.md	Initialized from 'NodeJS Express' project tem...
app.js	Initialized from 'NodeJS Express' project tem...
package-lock.json	Initialized from 'NodeJS Express' project tem...
package.json	Initialized from 'NodeJS Express' project tem...

b) **Docker file for the Microservice App**


Dockerfile 203 bytes

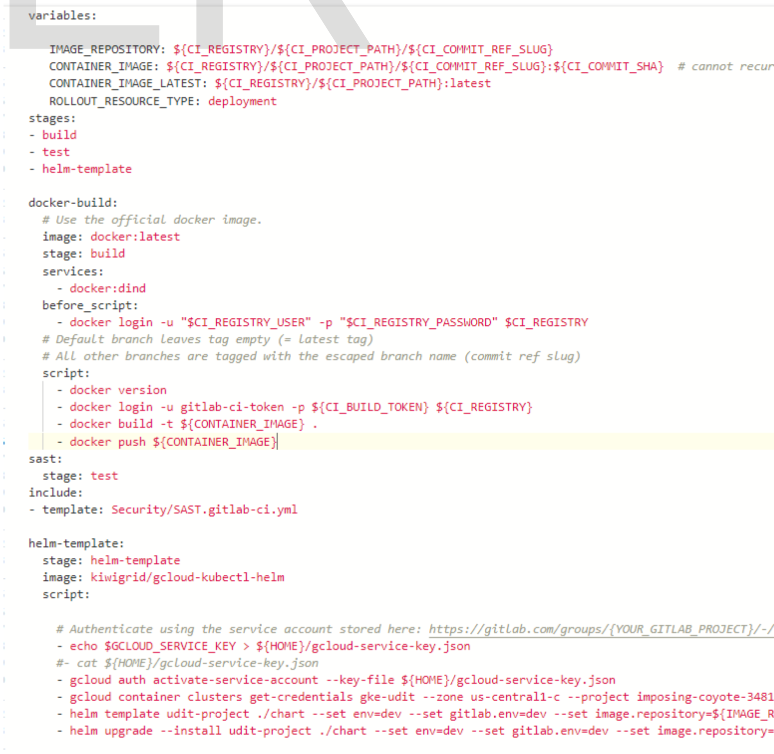
```
FROM node:8.11-alpine

WORKDIR /usr/src/app

ARG NODE_ENV
ENV NODE_ENV $NODE_ENV

COPY package*.json /usr/src/app,
RUN npm install

COPY . /usr/src/app

ENV PORT 5000
EXPOSE $PORT
CMD [ "npm", "start" ]
```

c) Kubernetes Helm Chart



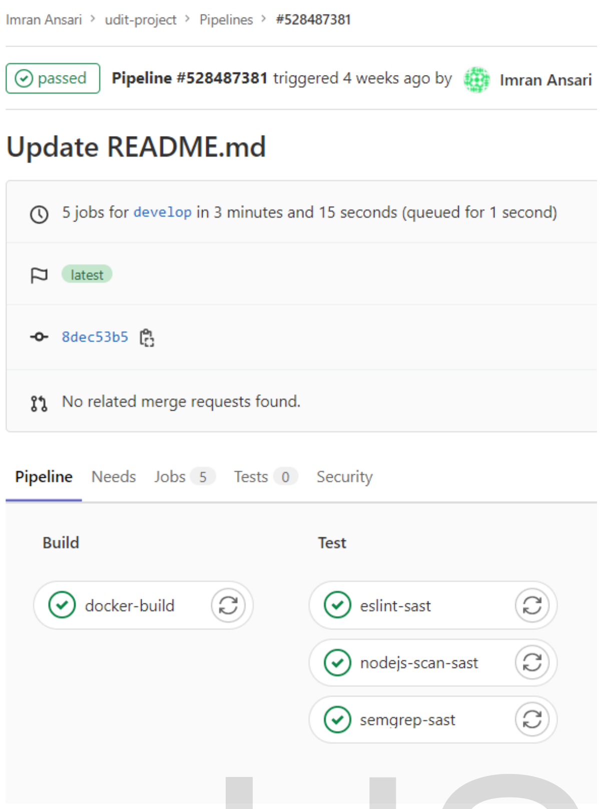
```
bin
chart
  templates
    tests
    NOTES.txt
    _helpers.tpl
    deployment.yaml
    hpa.yaml
    ingress.yaml
    service.yaml
    serviceaccount.yaml
  .helmignore
  Chart.yaml
  values.yaml
```

d) CI CD Using Gitlab pipeline with gitlab-ci.yaml file



```
variables:

   IMAGE_REPOSITORY: ${CI_REGISTRY}/${CI_PROJECT_PATH}/${CI_COMMIT_REF_SLUG}
   CONTAINER_IMAGE: ${CI_REGISTRY}/${CI_PROJECT_PATH}/${CI_COMMIT_REF_SLUG}:${CI_COMMIT_SHA}  # cannot recur
   CONTAINER_IMAGE_LATEST: ${CI_REGISTRY}/${CI_PROJECT_PATH}:latest
   ROLLOUT_RESOURCE_TYPE: deployment
stages:
- build
- test
- helm-template

docker-build:
  # Use the official docker image.
  image: docker:latest
  stage: build
  services:
    - docker:dind
  before_script:
    - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD" $CI_REGISTRY
  # Default branch leaves tag empty (= latest tag)
  # All other branches are tagged with the escaped branch name (commit ref slug)
  script:
    - docker version
    - docker login -u gitlab-ci-token -p ${CI_BUILD_TOKEN} ${CI_REGISTRY}
    - docker build -t ${CONTAINER_IMAGE} .
    - docker push ${CONTAINER_IMAGE}
sast:
  stage: test
include:
- template: Security/SAST.gitlab-ci.yml

helm-template:
  stage: helm-template
  image: kiwigrid/gcloud-kubectl-helm
  script:

    # Authenticate using the service account stored here: https://gitlab.com/groups/{YOUR_GITLAB_PROJECT}/-/
    - echo $GCLOUD_SERVICE_KEY > ${HOME}/gcloud-service-key.json
    #- cat ${HOME}/gcloud-service-key.json
    - gcloud auth activate-service-account --key-file ${HOME}/gcloud-service-key.json
    - gcloud container clusters get-credentials gke-udit --zone us-central1-c --project imposing-coyote-3481
    - helm template udit-project ./chart --set env=dev --set gitlab.env=dev --set image.repository=${IMAGE_R
    - helm upgrade --install udit-project ./chart --set env=dev --set gitlab.env=dev --set image.repository=
```

e) CI/CD pipeline


Imran Ansari > udit-project > Pipelines > #528487381

passed Pipeline #528487381 triggered 4 weeks ago by Imran Ansari

Update README.md

5 jobs for develop in 3 minutes and 15 seconds (queued for 1 second)

latest

8dec53b5

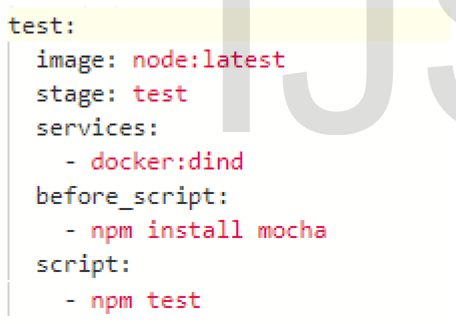
No related merge requests found.

Pipeline Needs Jobs 5 Tests 0 Security

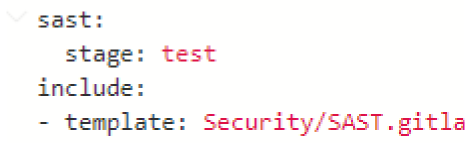
Build: docker-build

Test: eslint-sast, nodejs-scan-sast, semgrep-sast

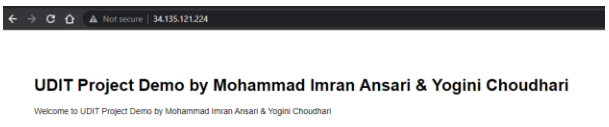
f) Test Cases


```
test:
  image: node:latest
  stage: test
  services:
    - docker:dind
  before_script:
    - npm install mocha
  script:
    - npm test
```

g) Testing using SAST


```
sast:
  stage: test
include:
- template: Security/SAST.gitla
```

h) Post Deployment of Microservices Application


Not secure | 34.135.121.224

UDIT Project Demo by Mohammad Imran Ansari & Yogini Choudhari

Welcome to UDIT Project Demo by Mohammad Imran Ansari & Yogini Choudhari

7. CONCLUSIONS

Cloud deployment of existing native applications and migrating to cloud application development though have been eased with so much technological tools, the usage of correct tools and deployment is still an area of concern and needs a special expertise to resolve the issues. In this research, it is found that Cloud Deployment can be done using the above-mentioned tools and strategies based on the Cloud Native Approach, from initiation of the idea to the deployment of the application. It's been observed how important is Agile to DevOps, how we can work iteratively and deliver the project faster. The strategic cultural aspects and trainings can help us achieve our goals faster and can make us automate more and more things. To achieve success, we need to make sure we execute the best practices and proper guidance in automating the complete cloud deployment process using DevOps tools. From the literature review and comparison with the tools it's been found that the Microservices Approach with the inclusion of the best practices of CI/CD in Infrastructure as Code implementation can provide us with the best results.

REFERENCES

1. Mandeep Kumar, "Combination of Cloud Computing and DevOps", International Journal of Modern Computer Science (IJMCS) Volume 4, Issue 5, 2016, pp.2-5.

2. P. Maragathavalli1 and M. Seshankkumar, "Automation Pipeline and Build Infrastructure using DevOps", International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 8, Issue XI, 2020, pp: 2-7.

3. Sikender Mohsienuddin Mohammad, "Streamlining DevOps automation for Cloud Applications", International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 6, Issue 4, 2018, pp: 1-5.

4. Desmond Bala Bisandu, "Cloud Devops Future of E-Business", International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS), Volume VII, Issue XI, 2018, pp: 1-5.

5. Muhammad Owais Khan, Awais Khan Jumani, Farhan, Waqas Ahmed Siddique and Asad Ali Shaikh, "Fast Delivery, Continuously Build, Testing and Deployment with DevOps Pipeline Techniques on Cloud", Indian Journal of Science and Technology, Volume 13, Issue 5, 2020, pp: 552-575.

6. Ravi Teja Yarlagadda, "DevOps for Better Software Security in the Cloud", JETIR, Volume 7, Issue 9, 2020, pp: 1-5.

7. Zhenhua Li, Yun Zhang, and Yunhao Liu, "Towards a Full-Stack DevOps Environment (Platform-as-a-Service) For Cloud-Hosted Applications", Tsinghua Science and Technology, Volume 22, Issue 1, 2017, pp: 1-9.

8. Sikender Mohsienuddin Mohammad, "Continuous Integration and automation", IJCRT, Volume 4, Issue 3, 2016, pp: 1-7.

9. Ravi Teja Yarlagadda, “The DevOps Paradigm with Cloud Data Analytics for Green Business Applications”, IJCRT, Volume 7, Issue 3, 2019, pp: 1-5.

10. Sikender Mohsienuddin Mohammad, “DevOps automation and Agile methodology”, IJCRT, Volume 5, Issue 3, 2017, pp: 1-4.

11. Sikender Mohsienuddin Mohammad, “Improve Software Quality Through Practicing Devops Automation”, IJCRT, Volume 6, Issue 1, 2018, pp: 1-6.

12. Sikender Mohsienuddin Mohammad, “An exploratory study of DevOps and it's future in the United States”, IJCRT, Volume 4, Issue 1, 2016, pp: 1-4.

13. Jadhav, Bhushan, Khushbu Hasija, Ritika Laharia, and Apurva MS. "A Comprehensive Framework for Implementing DevOps in Cloud for Education." Available at SSRN 3852757 (2021).

14. Ravi Teja Yarlagadda, “DevOps for Better Software Security in the Cloud”, JETIR, Volume 7, Issue 9, 2020, pp: 1-5.

15. Yadav, Archana. "Containerized Microservices architecture”, Chikitsak Samuha’s S. S. & L. S. Patkar College of Science & Commerce, Mumbai, India

16. Nishi Tiku, Sharan Iyengar, Kapil Kumar Trivedi, “Container First Approach for Micro, Small and Medium Enterprises - An Overview”, International Journal of Science and Research (IJSR), Volume 8, Issue 6, 2018, pp: 1520-1522.

17. Prajval Mohan, Tejas Jambhale, Lakshya Sharma, Simran Koul, Simriti Koul, “Load Balancing using Docker and Kubernetes: A Comparative Study”, International Journal of Science and Research (IJSR), Volume 9, Issue 2, 2020, pp: 1-11.

18. Pardeep Singh Virdi, “Categorize & Compare Cloud Automation & Devops Tools”, International Journal of Science and Research (IJSR), Volume 10, Issue 7, 2021, pp: 613-616.

19. Haider N. Hussain, “Recommendations to Evaluate and Choose Cloud Services and Deployment Models for e-Business Strategic Use: A Case Study of Malaysian SMEs”, International Journal of Science and Research (IJSR), Volume 4, Issue 11, 2015, pp: 1736-1745.

20. Prof. Hiral B. Patel, Prof. Nirali Kansara, “Cloud Computing Deployment Models: A Comparative Study”, International Journal of Innovative Research in Computer Science & Technology (IJIRCST), Volume 9, Issue 2, 2021, pp: 45-50.

21. Mervat Adib Bamiah and Sarfraz Nawaz Brohi, “Exploring the Cloud Deployment and Service Delivery Models”, International Journal of Research and Reviews in Information Sciences (IJRRIS), Volume 1, Issue 3, 2011, pp: 1-5.

22. Tavbulatova, Z. K., K. Zhigalov, S. Yu Kuznetsova, and A. M. Patrusova, "Types of cloud deployment.", In Journal of Physics: Conference Series, Volume 1582, Issue 1, 2020, p start. 012085.

23. Pranav T P, Charan S, Darshan M R, Girish L, “Devops Methods for Automation of Server Management using Ansible”, International Journal of Advanced Scientific Innovation, Volume 1, Issue 2, 2021, pp: 1-7.

24. Goutam Kamate1, S.G Raghavendra Prasad, “Docker & Containers, The Future of Microservices”, International Journal of Science and Research (IJSR), Volume 9, Issue 8, 2020 pp: 1-3.

25. Ravi Kumar, “An Introduction to Dockers: from Monolithic to Containerized Environment”, International Journal of Science and Research (IJSR), Volume 5, Issue 12, 2015, pp: 1-2.

26. Babak Bashari Rad, Harrison John Bhatti, Mohammad Ahmadi, “An Introduction to Docker and Analysis of its Performance”, International Journal of Computer Science and Network Security(IJCSNS), Volume 17, Issue 3, 2017, pp: 1-8.

27. Bellishree P, Dr. Deepamala. N, “A Survey on Docker Container and its Use Cases”, International Research Journal of Engineering and Technology (IRJET), Volume 7, Issue 7, 2020 pp: 1-5.

★★★