DOCKERIZING JENKINS AND ONE CLICK DEPLOYMENT WITH CI/CD PIPELINE

Project report submitted in fulfillment of the requirement for the degree of Bachelor of Technology

In

Computer Science and Engineering

By

Rohit Garg (161297)

Under the supervision of

Dr. Ravindara Bhatt

To

Department of Computer Science & Engineering

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CANDIDATE'S DECLARATION

This is to certify that the work which is being presented in the report entitled "**Dockerizing Jenkins and One click Deployment with CI/CD Pipeline**" in partial fulfilment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from February 2020 to July 2020 under the supervision of Dr. Ravindara Bhatt (Associate Professor, Senior Grade, Computer Science & Engineering Department) and Akash Hudge (Solution Delivery Consultant, ZS Associates, Pune).The matter embodied in the report has not been submitted for the award of any other degree or diploma.

R Garg

(Student Signature)

Rohit Garg, 161297

This is to certify that the above statement made by the candidates is true to the best of my knowledge.

RAVINDARA BHATT

(Supervisor Signature)

Dr. Ravindara Bhatt

Associate Professor (Senior Grade)

Department of Computer Science & Engineering

[illegible]

(Supervisor Signature)

Akash Hudge

Solution Delivery Consultant

ZS Associates Pvt. Ltd.

Pune, Maharashtra

India, 411014

Dated:

ACKNOWLEDGEMENT

I have put a lot of effort into this project. But this was not possible without the kind support and help of many people. I thank all of them sincerely.

I am greatly indebted to my project supervisor at college, **Ravindara Bhatt** and project supervisor at company, **Akash Hudge** for their guidance and constant supervision as well as providing me with all the information regarding the project titled **- Dockerizing Jenkins and One click Deployment with CI/CD Pipeline.**

I am also thankful to **Jaypee University of Information Technology** and **ZS Associates Pvt. Ltd.** for providing me with all the latest technologies, thus, comforting me with the project.

Again, I owe my profound gratitude to my project guide for his keen interest in the same during its progress.

R Garg

Rohit Garg, 161297

Dated:

TABLE OF CONTENTS

CONTENT	**PAGE NO.**
DECLARATION	i
ACKNOWLEDGEMENT	ii
LIST OF ABBREVIATIONS	v
LIST OF FIGURES	vi
ABSTRACT	vii-viii
CHAPTER 1: INTRODUCTION	**1-3**
1.1 Introduction	1-3
1.1.1 Continuous Integration	1-2
1.1.2 Continuous Delivery	2
1.1.3 Continuous Deployment	2-3
CHAPTER 2: LITERATURE SURVEY	**4-21**
2.1 TITLE: What is DevOps *by microfocus.com*	4-15
2.2 TITLE: Continuous Integration in DevOps *by C. Aaron Cois*	16-21
CHAPTER 3: PREPARATION OF DOCKERIZED JENKINS IMAGE	**22-25**
3.1 High Level Architecture/Paper Design	22
3.2 Prerequisites	22
3.3 Description	23
3.4 Challenges and Workarounds	24
3.5 Deployment Guidelines	24-25
3.5.1 Prerequisites	24-25
3.5.2 Description	25

CHAPTER 4: USER GUIDE **26-30**

4.1 Prerequisites 26

4.2 Deployment 26-30

4.2.1 Structure for the bitbucket repository 26-27

4.2.2 To pull the image of dockerized Jenkins 27

4.2.3 To launch container from Image 27

4.2.4 Accessing Jenkins and SonarQube 28

4.2.5 Configuration for running CI/CD Pipeline on Jenkins 28-30

4.3 Steps to access Jenkins Workspace 30

4.4 Description 30

CHAPTER 5: PERFORMANCE ANALYSIS **31-32**

5.1 Performance 31-32

CHAPTER 6: CONCLUSION **33**

6.1 Conclusion 33

REFRENCES **34**

LIST OF ABBREVIATIONS

CI	Continuous Integration
CD	Continuous Deployment
DevOps	Developer Operations
EC2	Amazon Elastic Compute Cloud
EKS	Amazon Elastic Kubernetes Service
SQL	Structured Query Language
OS	Operating System
AWS	Amazon Web Services
IAM	AWS Identity and Access Management
ECR	Amazon Elastic Container Registry
UI	User Interface
URL	Uniform Resource Locator

LIST OF FIGURES

Figure No.	**Title**	**Page No.**
Figure 1.1	DevOps Overview	3
Figure 2.1	Architecture	4
Figure 3.1	Jenkins Configuration Snippet #1	11
Figure 3.2	Jenkins Configuration Snippet #2	11
Figure 3.3	Jenkins CI/CD Pipeline	12

ABSTRACT

Automation is a need in ventures since it not just looks for improve the personal satisfaction for people at both home and work, it permits the dissemination of both quality items and administrations to be made accessible at quicker rates, and decreases down time and human mistake. It is what is right now propping up our reality's dispersion chains.

The more prominent all ventures access to Automation, the more noteworthy our lives become in general.

DevOps refers to the combinations of Development and Operational Skills. Basically, to overcome the long, time consuming process of traditional waterfall models, DevOps are preferred. Now-a-days many companies are interested to employ engineers who are skilled at DevOps. DevOps integrates development and operation i.e. it takes both software developers and IT sector. DevOps engineers must play nice as they have to collaborate, and handle an automated infrastructure, workflow and continuously measuring application performance.

Phases of DevOps:

DevOps engineering consists of three phases:

1. Automated testing
2. Integration
3. Delivery

Lifecycle of DevOps:

DevOps mainly focuses on planning, coding, building and testing as the development part. Added is the operational part that includes releasing, deployment, operation, and monitoring. The development and operational part makeup the life cycle.

Most popular DevOps tools:

Git is a version control system tool

Jenkins, Selenium are Continuous Integration Testing tools that uses predefined frameworks.

Puppet, chef, Ansible are configuration management and deployment tools

Nagios is a Continuous monitoring tool

Docker is a container concept tool.

CHAPTER-1
INTRODUCTION

1.1 INTRODUCTION

The develop and take a look at section of software package development is that the focus of continuous integration. As developer's amendment software package code, those changes at once checked into a central ASCII text file system. once code is checked in, automatic build processes and tests are triggered to create certain that the changes failed to break the larger package being worked on. once shorter and additional frequent develop-build-test cycles are used, committal to writing errors are caught additional quickly, and also the risk related to large-scale code changes is eased.

1.1.1 CONTINUOUS INTEGRATION

Using continuous integration, a product is constructed to incorporate and integrate each code amendment on each commit (continuously), by any and every one developer. an automatic build then verifies every arrival, property groups sight issues early.

In advanced product, changes that appear straightforward and self-contained will turn out fortuitous consequences. If multiple developers square measure operating in parallel on multiple, isolated code branches, merging changes to a typical master branch will have unpredictable results. Unpredictable results usually result in multiple rounds of regression testing and bug fixes. Continuous integration

(CI) could be an element of the continual delivery method that allows developers to integrate their updates into the master branch on an everyday basis. With CI, automatic tests run before and when every amendment is incorporate, substantiating that no bugs are introduced.

1.1.2 CONTINUOUS DELIVERY

Continuous delivery implies that as new software package options and fixes undergo the develop-build-test cycle, they become on the market as apace as attainable. once smaller changes are delivered additional ofttimes into production, the danger of large-scale changes breaking the system goes down, and also the delay in emotional them to customers is reduced.

1.1.3 CONTINUOUS DEPLOYMENT

Continuous Deployment is simple: simply ship your code to customers as usually as attainable. perhaps nowadays that’s weekly rather than monthly, however over time you’ll approach the best and you’ll see the progressive advantages on the manner.

Continuous delivery and Deployment terms that square measure are usually confused. In fact, they're virtually constant, as most of the feature unleash method is additional or less identical. However, there's one key distinction between the two: beneath continuous delivery, somebody with comfortable privileges should approve every build to be discharged to production, usually a product manager with simply the push of a button. With continuous readying, though, the readying method happens mechanically on a predefined unleash cadence, like nightly, or maybe with every build. the sole thanks to eliminate this manual step is for quality assurance and alternative testing to happen mechanically.

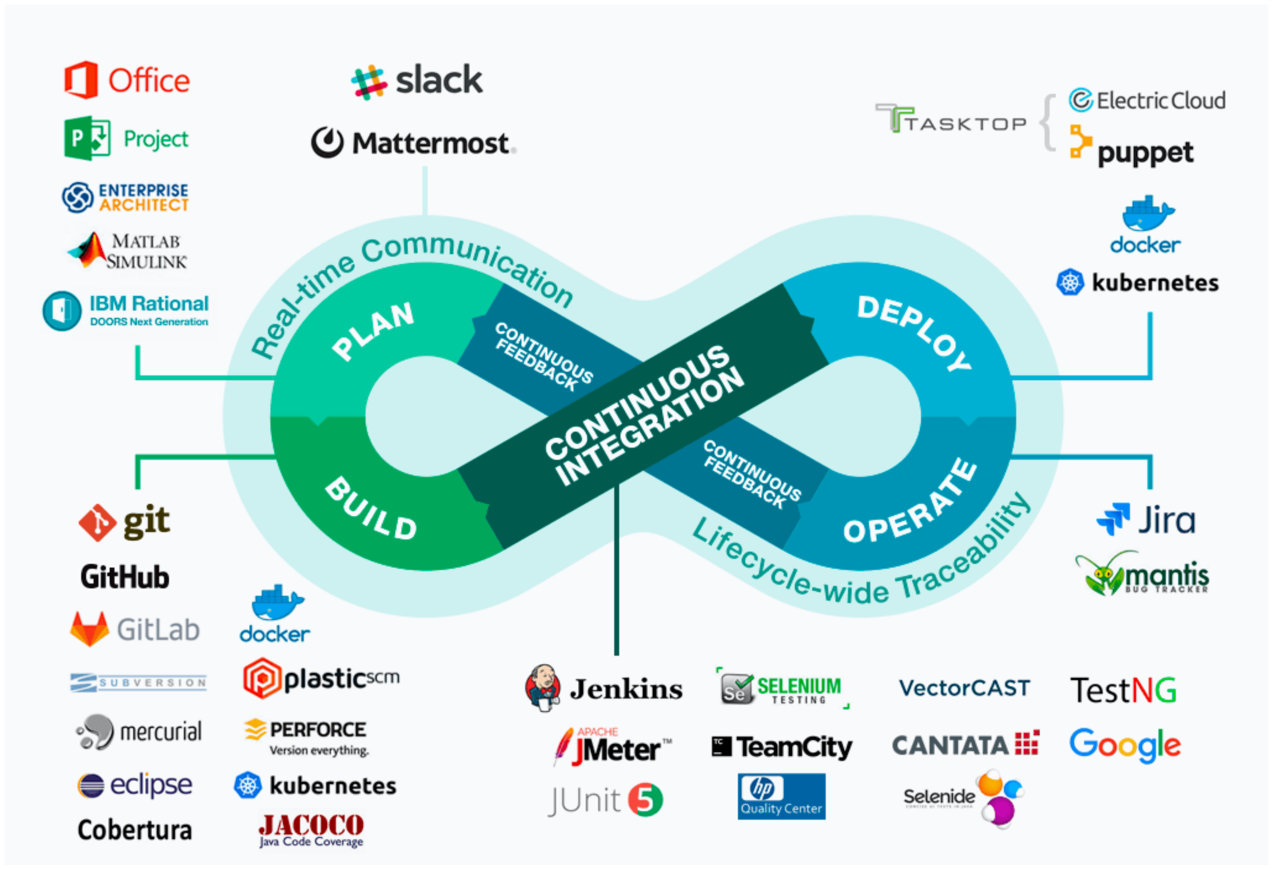


Fig 1.1: DevOps Overview

CHAPTER-2

LITERATURE SURVEY

2.1 TITLE: What is DevOps *by microfocus.com*

DevOps (a portmanteau of "improvement" and "tasks") is the mix of practices and apparatuses intended to build an association's capacity to convey applications and administrations quicker than conventional programming advancement forms. This speed empowers associations to all the more likely serve their clients and contend all the more adequately in the market.

In basic terms, DevOps is tied in with expelling the obstructions between generally siloed groups, advancement and activities. Under a DevOps model, improvement and tasks groups cooperate over the whole programming application life cycle, from advancement and test through arrangement to activities.
DevOps security, all the more regularly alluded to as DevSecOps, alludes to the control and practice of shielding the whole DevOps condition through procedures, strategies, procedures, and innovation. The DevSecOps reasoning is that security ought to be incorporated with all aspects of the DevOps life cycle, including

commencement, structure, fabricate, test, discharge, backing, support, and past.

Conventional security works from the position that once a framework has been planned, its security imperfections would then be able to be resolved and remedied before discharge. With the change to a DevOps model, conventional security rehearses happen past the point of no return in the improvement cycle and are unreasonably delayed for the plan and arrival of programming worked by emphasis. Along these lines, they can turn into a significant barrier to conveying applications and administrations at speed.

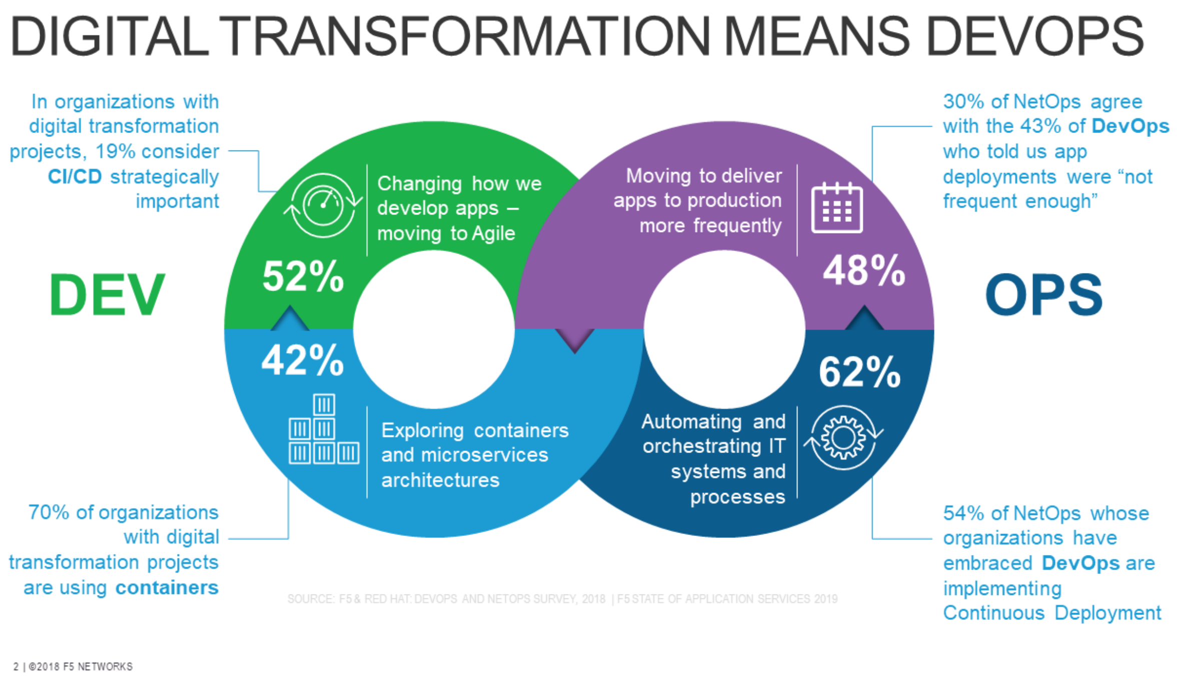


Fig 2.1: DevOps Transformation

With DevSecOps, security turns into the attention of everybody on a DevOps group. DevSecOps has the objective of executing security choices at speed and scale without giving up wellbeing. DevSecOps includes progressing, adaptable coordinated effort between discharge designers and security groups. The ideas of "speed of conveyance" and "building secure code" are converted into one smoothed out procedure. Security testing is done in emphases without hindering conveyance cycles. Basic security issues are managed as they become evident, not after a danger or bargain has happened.
DevOps is the immediate relative of light-footed programming advancement, conceived from the need to stay aware of expanded programming improvement speed and throughput spry strategies. Progressions in light-footed advancement featured the requirement for an increasingly all-encompassing way to deal with the product conveyance life cycle, coming about in DevOps.

"Deft turn of events" is an umbrella term for a few iterative programming improvement philosophies, a large number of which have persisted to DevOps:

Scrum—a structure wherein individuals can address complex versatile issues while conveying results of the most elevated conceivable worth.

Kanban—a technique for dealing with the formation of items with an accentuation on constant conveyance while not overburdening the advancement group. Like Scrum, Kanban is a procedure intended to assist groups with cooperating all the more successfully.

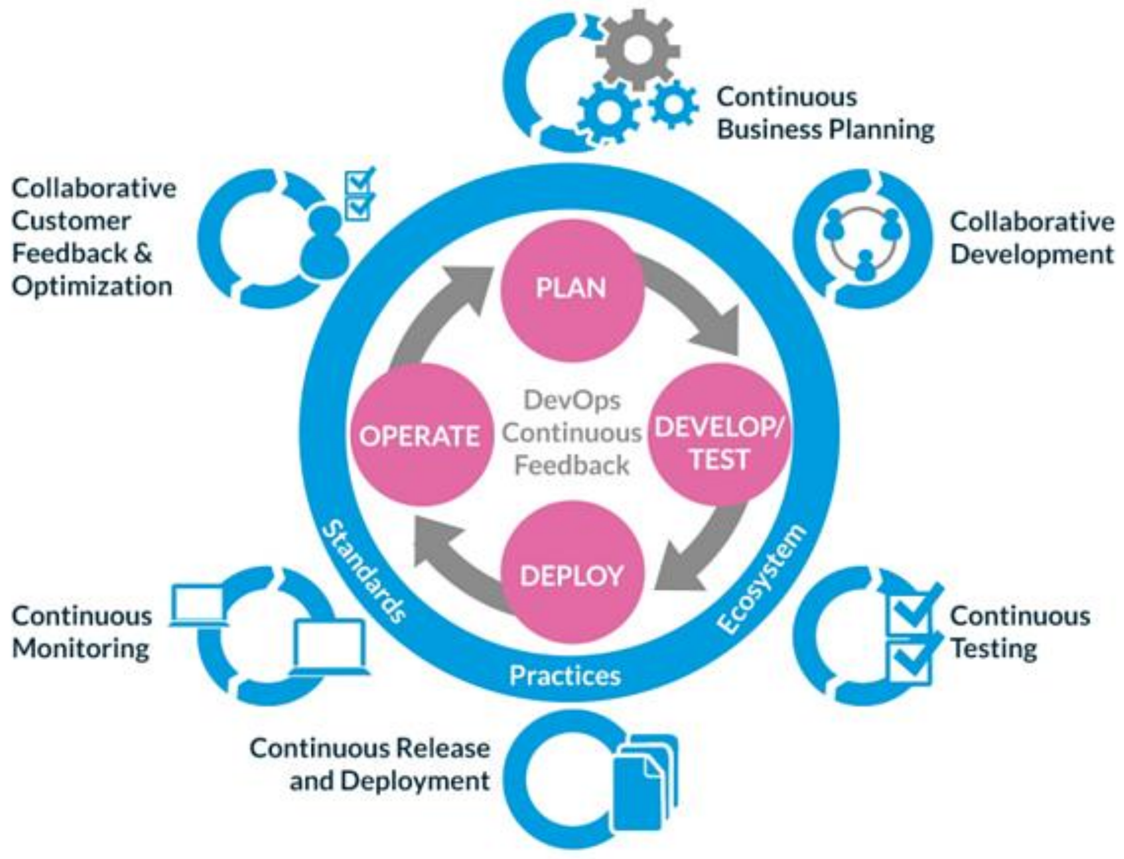

Continuous Business Planning
Collaborative Customer Feedback & Optimization
Collaborative Development
PLAN
DevOps Continuous Feedback
OPERATE
DEVELOP/ TEST
DEPLOY
Standards
Ecosystem
Practices
Continuous Monitoring
Continuous Testing
Continuous Release and Deployment

Fig 2.2: CI/CD Flow

Scaled Agile Framework (SAFe)— a lot of association and work process designs expected to control undertakings in scaling lean and dexterous practices. SAFe is one of a developing number of systems that try to address the issues experienced when scaling past a solitary group.

Lean turn of events—an interpretation of lean assembling standards and practices to the product advancement space. Lean offers a reasonable structure, qualities, and standards, just as best practices got for a fact, that help spry associations.

Extraordinary programming (XP)— a product advancement approach planned to improve programming quality and responsiveness to changing client necessities. XP advocates visit discharges in short advancement cycles, proposed to improve efficiency and present checkpoints at which new client necessities can be embraced. Different components of outrageous programming remember programming for sets or doing broad code audit, unit testing of all code, not programming of highlights until they are required, a level administration structure, code straightforwardness and lucidity, expecting changes in the client's prerequisites over the long haul and the issue is better comprehended, and regular correspondence with the client.

A few key practices can assist associations with enhancing quicker through robotizing and smoothing out the product improvement the executive’s procedure. One basic DevOps practice is to perform exceptionally visit however little updates. These updates are typically steadier than the updates performed under customary discharge rehearses. Associations utilizing a DevOps model send refreshes significantly more frequently than associations utilizing customary programming advancement rehearses.

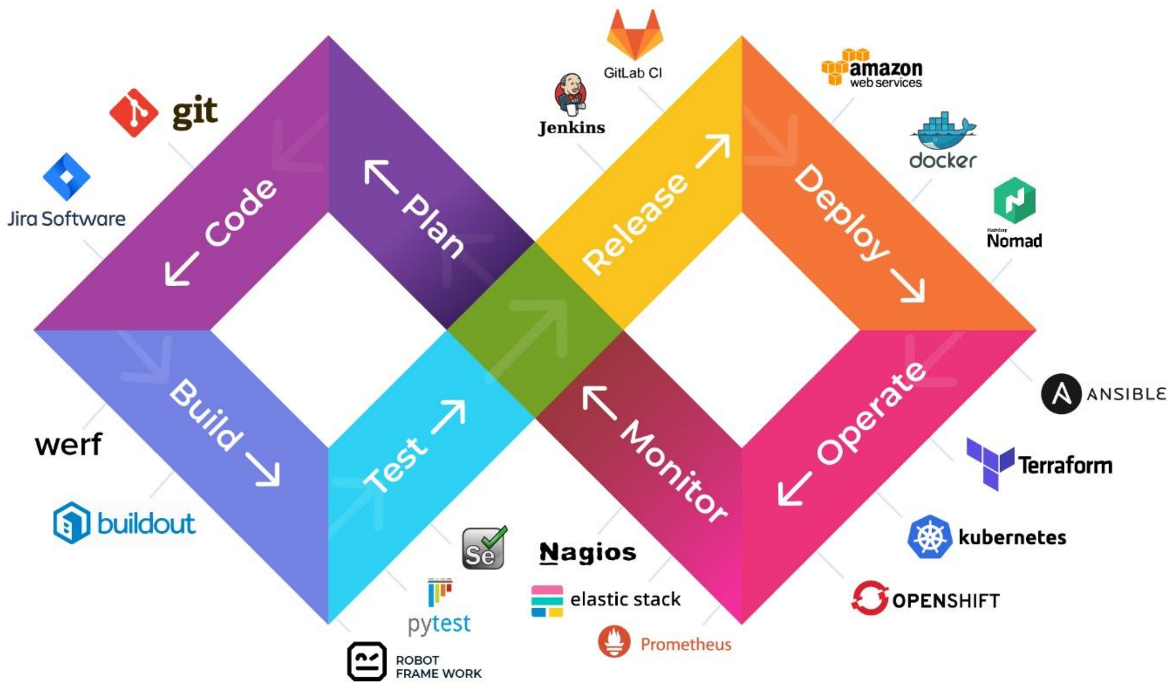


Fig 2.3: CI/CD tools

Correspondence and coordinated effort are cornerstones of the arrangement of DevOps rehearses. Computerization of the product conveyance process builds up joint effort by genuinely uniting the work processes and duties of improvement and activities. Correspondence across designers, tasks, and much different groups, for example, advertising and deals, permits all pieces of the association to adjust all the more intently on objectives and activities.

DevOps practices, for example, nonstop joining and consistent conveyance let DevOps groups convey quickly, securely, and dependably. Observing and logging help DevOps groups track the presentation of utilizations so they can respond rapidly to issues.

Microservices

The microservices engineering is a plan way to deal with construct a solitary application as a lot of little administrations. Each help runs in its own procedure and speaks with different administrations through a very much characterized interface utilizing a lightweight system. You can utilize various systems or programming dialects to compose microservices and send them autonomously, as a solitary help, or as a gathering of administrations.

Associations may likewise utilize a microservices design to make their applications progressively adaptable and empower faster development. Ordinarily, each assistance is matched with a little, nimble group who takes responsibility for administration.

Consistent coordination and nonstop conveyance

CI is a product improvement practice where engineers routinely consolidate their code changes into a focal storehouse, trailed via robotized manufactures and tests. The key objectives of CI are to discover and fix bugs faster, improve programming quality, and decrease the time it takes to approve and discharge new programming refreshes. Compact disc develops CI by sending all code changes to a testing or creation condition after the assemble stage.

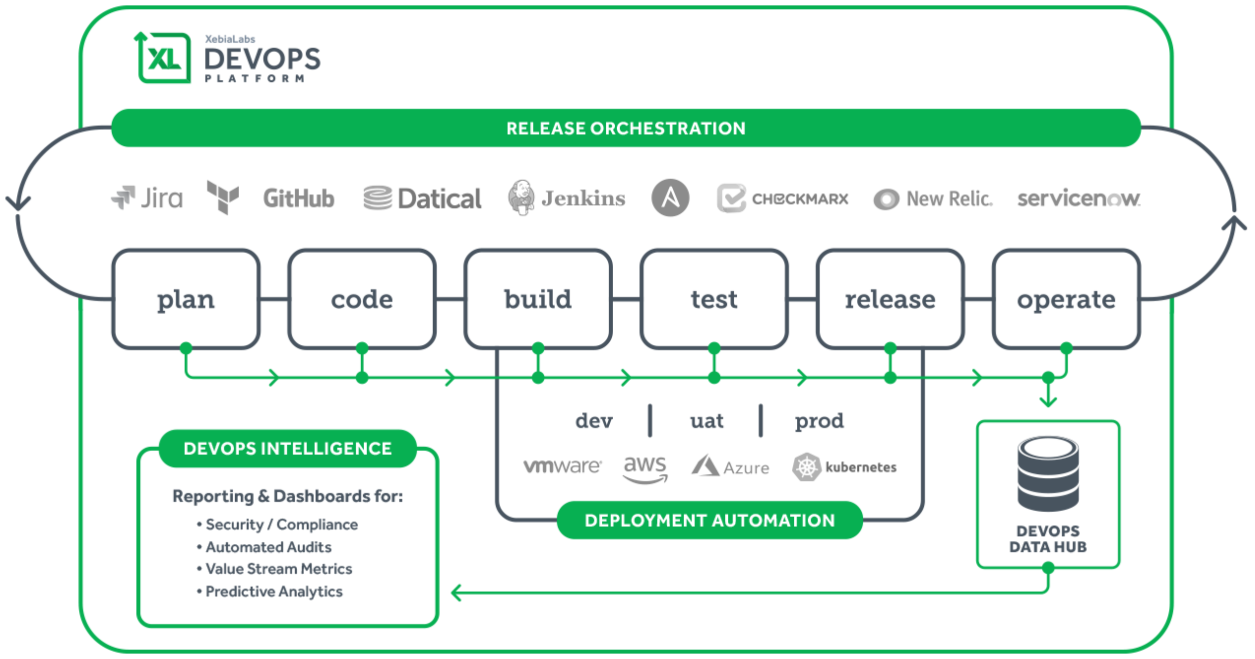


Fig 2.4: DevOps Flow

Checking and logging

By catching and dissecting logs created by applications, DevOps groups can more readily see how programming changes or updates may influence clients.

Building a protected DevOps model

Moving to DevOps isn't a goal. It is an excursion. DevOps is generally changing how improvement and tasks are done today. You can utilize the DevOps rehearses, procedure, structures, and work process, in view of the DevOps theory, to incorporate security with your product improvement life cycle at speed and scale without giving up wellbeing, while at the same time limiting dangers, guaranteeing consistence, and lessening rubbing and expenses. DevOps permit improvement, tasks, and security groups to offset security and consistence with speed of conveyance, and to incorporate security with the full SDLC.

DevOps was birthed by the training and proliferation of deft programming improvement. Since the dexterous strategy accelerates the improvement procedure and throughput speed, there was a need to adjust the structure of undertaking groups to oblige this new reality. As lithe culture flourished, it turned out to be certain that having the product improvement and IT activities groups working independently was counterproductive and wasteful.

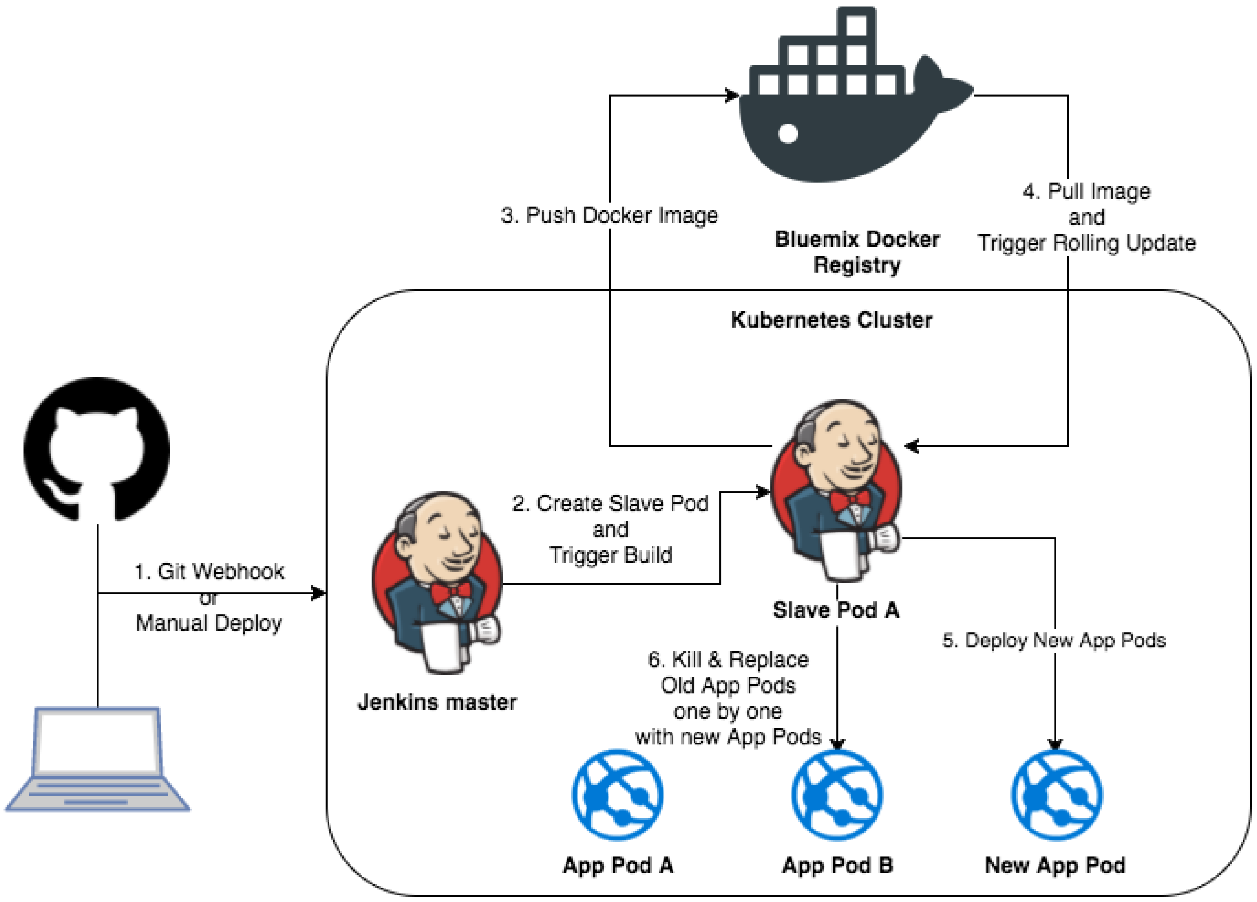

3. Push Docker Image
Bluemix Docker Registry
4. Pull Image and Trigger Rolling Update
Kubernetes Cluster
2. Create Slave Pod and Trigger Build
1. Git Webhook or Manual Deploy
Slave Pod A
5. Deploy New App Pods
6. Kill & Replace Old App Pods one by one with new App Pods
Jenkins master
App Pod A
App Pod B
New App Pod

Fig 2.5: Jenkins and DevOps

DevOps rehearses computerize arrangement pipelines and creates quicker input to improve effectiveness, consistency, practicality, and security. It carries engineers into the creation condition, gives them more profound bits of knowledge into the framework and gets them progressively associated with application lifecycle the executives.

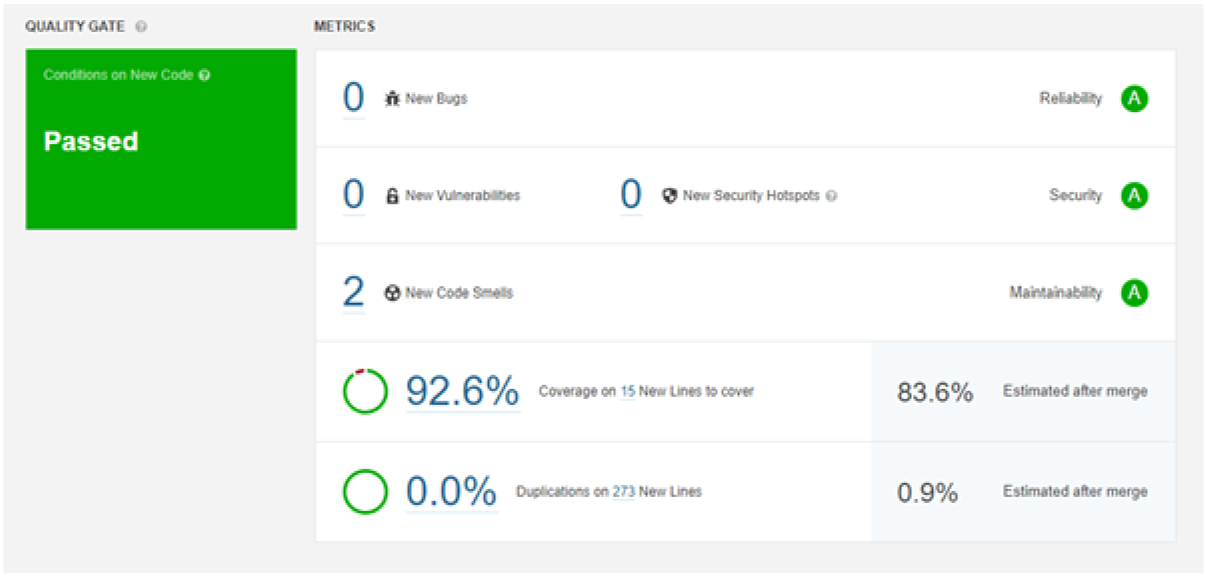
Associations that grasp DevOps may have all IT assets inside a customary server farm, all assets in an offsite cloud, or disperse their assets in a half and half condition.

The DevOps development isn't characterized, nor drove, by customary IT programming, equipment, or the executive’s sellers. What's more, there are right now no systematized rules or manuals for DevOps, just by and large acknowledged rules. All things considered, selection and execution of DevOps differ enormously starting with one association then onto the next.

The learnings of DevOps are basically proselytized by an enthusiastic grassroots network of IT professionals, spread over a wide assortment of IT disciplines. Most individuals from the DevOps people group include dynamic occupations inside different associations, and they share their learnings in various on the web and in-person discussions and social events. Contingent upon the number and development of the professionals in an association, the advantages of a DevOps usage can be critical

2.2 TITLE: Continuous Integration in DevOps *by C. Aaron Cois*

Similarly, as with all huge social activities or enormous ventures, DevOps requires key partners who distinguish the requirement for DevOps, champion the reason, and support the assets to investigate, execute, and measure accomplishment in a top-down style. In the DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations, the writers suggest starting with thoughtful and imaginative gatherings, scoring some quick successes, and afterward extending impact from that point to construct a quiet dominant part to prevail upon the holdouts. Littler associations can forego the majority of that structure by permitting a base up way to deal with explore and receive DevOps benefits.


QUALITY GATE

Conditions on New Code

Passed

METRICS

0 New Bugs — Reliability A

0 New Vulnerabilities — 0 New Security Hotspots — Security A

2 New Code Smells — Maintainability A

92.6% Coverage on 15 New Lines to cover — 83.6% Estimated after merge

0.0% Duplications on 273 New Lines — 0.9% Estimated after merge

Fig 2.6: SonarQube Report Demo

Yet, actions speak louder than words - objectives must be set, in a perfect world with measurements, so an activity plan can be framed and estimated. These activity plans will call for cross-group controls, coordination, and needs to unblock clashes and advancement storehouses. Change is hard, so a specially appointed methodology won't work. It must be organized to square away the association's specialized and social obligation.

On the off chance that the entirety of this looks overwhelming, recollect that the ROI is gigantic: quickened development, more noteworthy proficiency, improved versatility and flexibility, expanded benefits, and better personal satisfaction and work.

In the event that you are simply beginning with DevOps, fortunately there is presently a flourishing, decentralized, and inviting DevOps people group for you to take advantage of: new adopters should search out the closest DevOps Days, DevOps Meetup gatherings, and related meetings to discover neighbourhood professionals.

Another extraordinary wellspring of data is the Nutanix entryway, which incorporates a variety of assets, going from labs, working contents and model applications, official documentation for Nutanix APIs, engineer network websites, occasions, and that's only the tip of the iceberg.

Make certain to likewise look at these assets to get familiar with how Nutanix offers the perfect DevOps stage, crumbling framework storehouses to help consistent development and tasks at scale!

Without this QA procedure, a designer may register broken code with a focal storehouse. Different designers may make changes that rely upon this messed up code, or endeavour to consolidate new changes with it. At the point when this occurs, the group can lose control of the framework's working state, and endure a misfortune in energy when compelled to return changes from various engineers to come back to a utilitarian state.

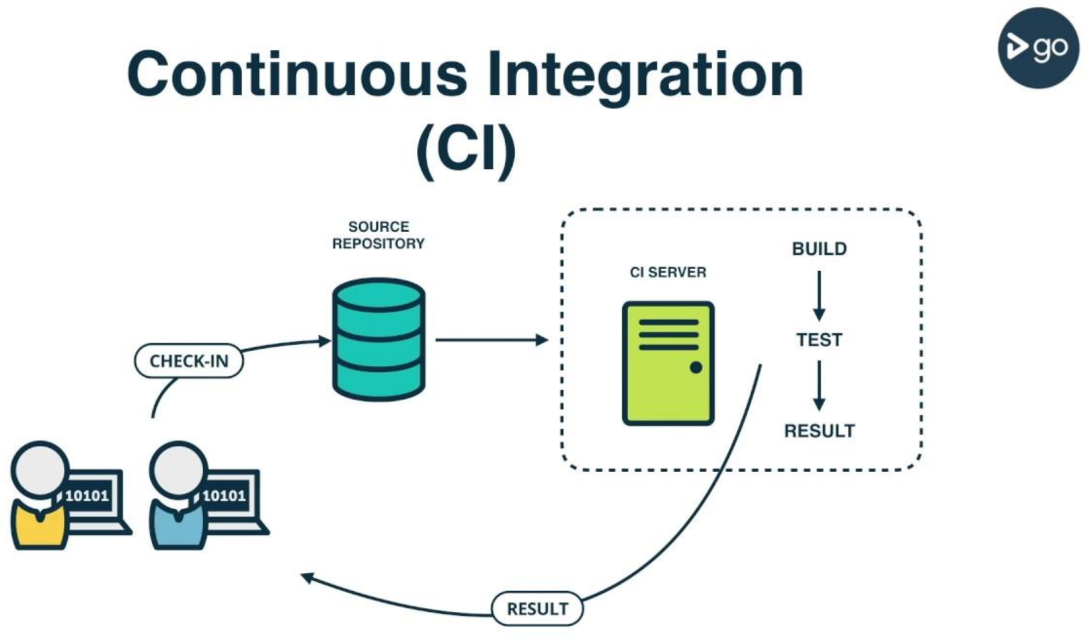


Fig 2.7: CI Flow

CI servers (otherwise called manufacture servers) naturally gather, fabricate, and test each new form of code focused on the focal group store, guarantees that the whole group is alarmed whenever the focal code archive contains broken code. This seriously constrains the opportunity for cataclysmic consolidation issues and loss of work based upon a messed-up codebase. In develop activities, the CI server may likewise consequently convey the tried application to a quality confirmation (QA) or organizing condition, guaranteeing the Agile dream of a predictable working rendition of programming.

All the activities portrayed above are performed dependent on mechanized setup and arrangement contents composed cooperatively by advancement and tasks engineers. The joint effort is significant - it guarantees that tasks aptitude in sending needs and best practices is spoken to in the improvement procedure, and that all colleagues comprehend these computerized contents and can utilize and upgrade them. This coordinated effort additionally makes way for utilization of similar contents to inevitably convey the framework into creation conditions with high certainty, a procedure known as ceaseless arrangement, which is a point for a later post.

As appeared in the realistic underneath, the construct server looks at new code from source control, orders/manufactures it (if vital), and tests the code (fundamentally unit tests, at this stage, however static code investigation is likewise conceivable). When the code is tried, the manufacture server sends it to QA. Now, the construct server can likewise dispatch contents to perform joining testing, UI testing, propelled security testing (more on this soon) and different tests requiring a running adaptation of the product. Reliable with Agile prerequisites that stress a persistently working rendition of the product, our CI server naturally returns to the last fruitful variant of the product, keeping a working QA framework accessible regardless of whether mix tests fizzled.

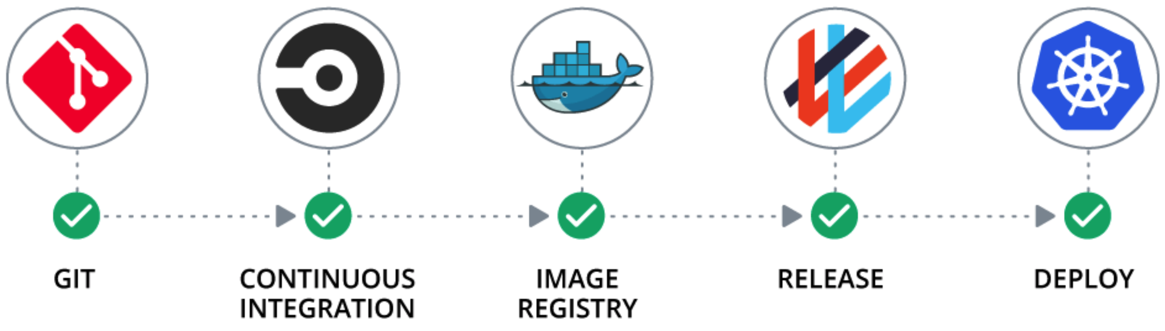


Fig 2.8: DevOps start to end

While CI is in no way, shape or form another marvel, the DevOps development underscores its significance as a fundamental strategy for programming process computerization and requirement. There are numerous well-known CI frameworks, including Jenkins, Bamboo, Team city, Cruise Control, Team Foundation Server, and others. The assortment of frameworks implies that any group ought to have the option to discover a device that the two addresses its issues and incorporates well with the innovation stack(s) it utilizes.

For more data on this and different DevOps-related points, each Thursday the SEI distributes another blog entry offering rules and common-sense guidance to associations trying to embrace DevOps practically speaking. We invite your criticism on this arrangement, just as proposals for future substance.

CHAPTER-3

PREPARATION OF DOCKERIZED JENKINS IMAGE

3.1 HIGH LEVEL ARCHITECTURE/PAPER DESIGN

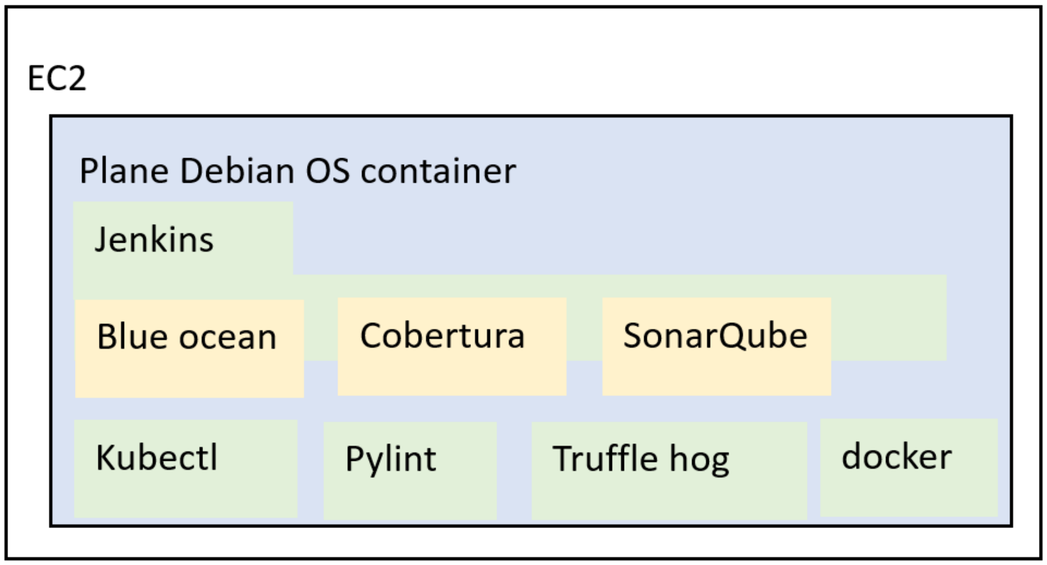
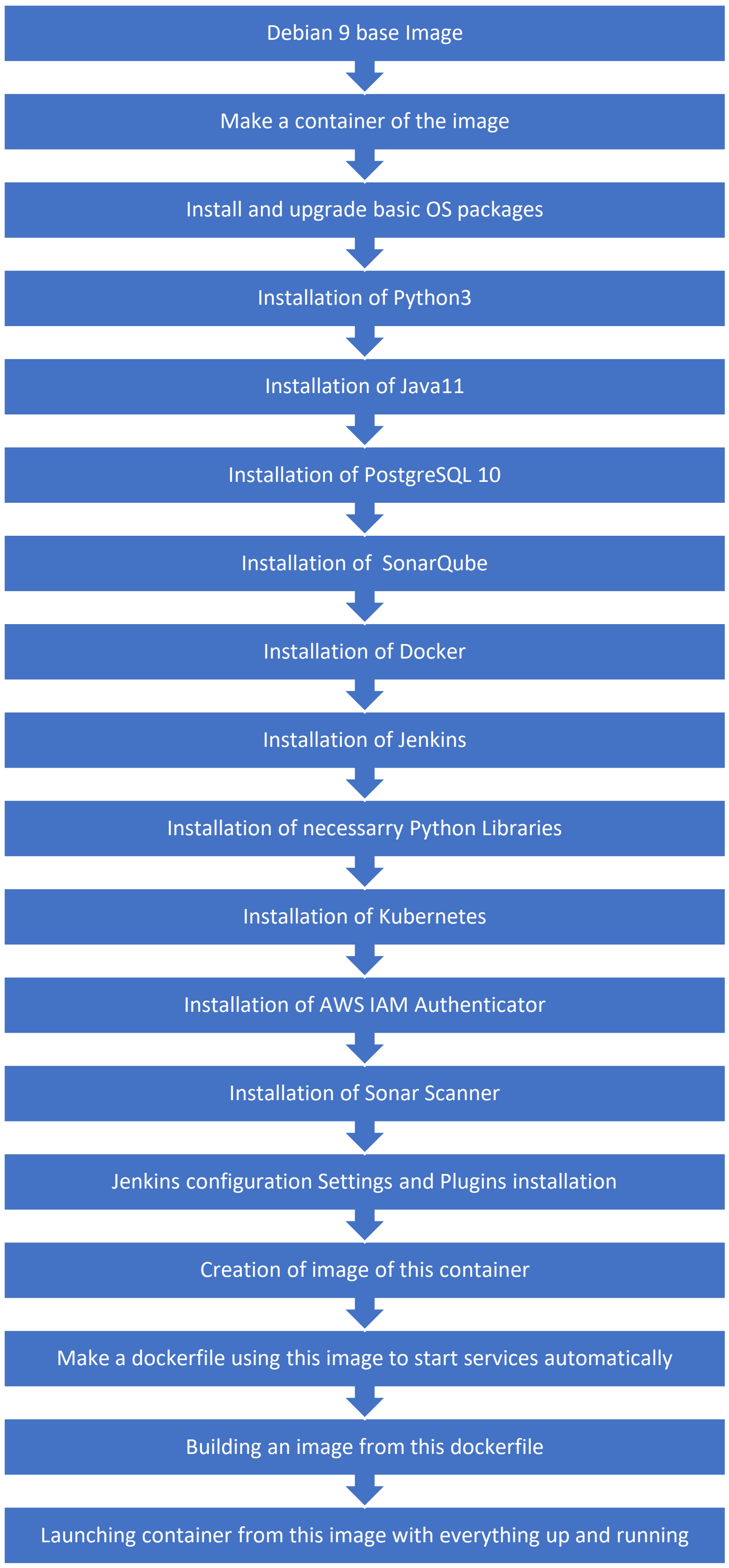


Fig 3.1: Architecture

3.2 PREREQUISITES

- Fresh image of Debian 9
- Bitbucket repository for Jenkins pipeline
- EC2 instance
- Docker installed on EC2 instance
- EKS cluster

3.3 DESCRIPTION


Debian 9 base Image
Make a container of the image
Install and upgrade basic OS packages
Installation of Python3
Installation of Java11
Installation of PostgreSQL 10
Installation of SonarQube
Installation of Docker
Installation of Jenkins
Installation of necessarry Python Libraries
Installation of Kubernetes
Installation of AWS IAM Authenticator
Installation of Sonar Scanner
Jenkins configuration Settings and Plugins installation
Creation of image of this container
Make a dockerfile using this image to start services automatically
Building an image from this dockerfile
Launching container from this image with everything up and running

3.4 CHALLENGES AND WORKAROUNDS:

1. Exposing installations to a certain port and mounting of paths need to be done in "docker run" command only, i.e. (at the time of building container out of the image), it can't be done after the container is built.

2. We are running all the docker commands inside the container and hence, we need to run them with the sudo access, but Jenkins docker plugins doesn't allow a way to execute commands with sudo access,

   Workaround: We preferred shell commands for docker in Jenkinsfile.

3. Docker was required inside container for running our pipeline. Installing docker inside our docker container doesn't work directly due to system privilege issue.

   Workaround: We mounted the docker.sock paths of EC2 to docker.sock paths of container. This allows us to use docker installed on our EC2 machine inside our container.

4. Docker in docker had difficulties in running on CentOS 7.

   Workaround: We preferred Debian OS 9 over CentOS 7.

3.5 DEPLOYMENT GUIDELINES:

3.5.1 Prerequisites:

- Access to dockerized Jenkins image
- EKS Cluster
- EC2 instance
- Docker installed on EC2 instance

- Bitbucket Repository
- Commands mentioned in the description below

3.5.2 Description:

1. To create the container:
   docker run -d -p 7777:8080 -p 9003:9000 --privileged --name container_name -v /var/run/docker.sock:/var/run/docker.sock -it image_name

   Explanation: The above command will create a docker container from our image with all services up and running. Since inside docker container we are running Jenkins on port 8080 and Sonarqube on port 9000, we are forwarding these ports to 7777 and 9003 ports of EC2 respectively. Also, we are mounting the docker paths of EC2 to docker path of container.

   Image name on ECR: automate-jenkinspp

2. To launch a bash terminal within the container:
   docker exec -it container_name bash

   Explanation: The above command will start the bash terminal within a container.
   To enter as root user, use /bin/bash.

3. If in case already launched container was stopped manually, you can relaunch it by:

   docker container start container_name

CHAPTER-4

USER GUIDE

4.1 PREREQUISITES:

1. EKS Cluster (Optional)
2. Bitbucket repository containing Jenkins files
3. Image of dockerized Jenkins
4. EC2 instance
5. Docker installed on EC2 instance
6. Host machine should have proper IAM role to be able to access ECR repository

Image name (on AWS ECR): automated-jenkins

4.2 DEPLOYMENT

4.2.1 Structure for the bitbucket repository

The bitbucket repository should contain the DevOps folder containing the following subdirectories and files (as shown below):

```
DevOps
|------------ Docker
|             |-- Dockerfile
|              `-- requirements.txt
|------------ Jenkins
|             |-- Jenkinsfile
|             |-- common_constant.py
|             |-- copy_logs_from_ec2.sh
|             |-- copy_logs_to_ec2.sh
|             |-- merge_request.py
|               |-- pylint.sh
|               |-- scan_changes.py
|               |-- secrets_config.json
|               |-- truffle_hog_scan.sh
|                `-- update_status_code_quality.py
|------------ Kubernetes
|               |-- deployment.yaml
|               |-- profiler-expose.yaml
|                `-- unit_test_kubectl.sh
 `------------ unit_testing
                |-- common_constant.py
                |-- flask_dataprofiler.py
                 `-- test_flask_dataprofiler.py
```

4.2.2 To pull the image of dockerized Jenkins (containing preinstalled software and plugins e.g. SonarQube, trufflehog, blue ocean, etc.):

The image needs to be pulled from Amazon ECR to local environment using the below command:

```
docker pull automated-jenkins
```

4.2.3 To launch the container from image:

Just after pulling the image from ECR, container from that image can be launched by the below command with all services up and running:

```
docker run -d -p 7777:8080 -p 9003:9000 --privileged --name
container_name -v /var/run/docker.sock:/var/run/docker.sock -it
automated-jenkins
```

Here, we are exposing 8080 port of docker container (running Jenkins) to 7777 port of host machine and similarly, we are exposing 9000 port of docker container (running SonarQube) to 9003 port of host machine. Also, we are mounting the docker path of host machine to docker path of container to make running of docker inside docker container possible.

4.2.4 Accessing Jenkins and SonarQube:

As soon as the container is launched with all services up and running, Jenkins and SonarQube can be accessed as follows:

Link for accessing Jenkins UI:

URL: http://$$host_machine_ip:7777/

Username: admin

Password: admin

Link for accessing SonarQube UI:

URL: http:// $$host_machine_ip:9003/

Username: admin

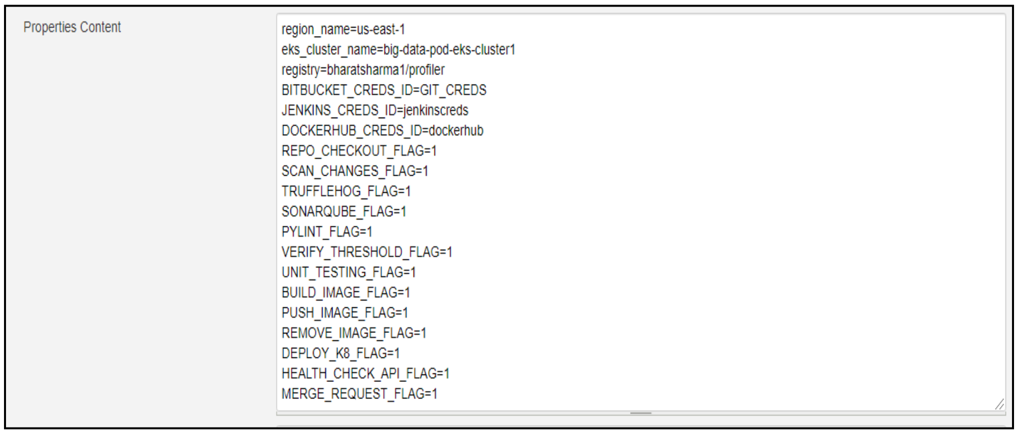
Password: admin

4.2.5 Configurations for running the CI/CD Pipeline on Jenkins:

Once the Jenkins UI is launched, two Jenkins job CICD_pp and scan_changes_ci_cd can be found at the home screen.

- **Setting the Credentials:**
  Go to the credentials option of Jenkins and update the following credentials according to your environment,
    a. Bitbucket Credentials
    b. Jenkins Credentials
    c. Dockerhub Credentials

- **Configurations for CICD_pp Job:**
  1. Click on configure link after going into *CICD_pp* job:
      a. Update *region_name*, *eks_cluster_name*, *BITBUCKET_CREDS_ID*, *JENKINS_CREDS_ID*, *DOCKERHUB_CREDS_ID* in the Properties Content section of Environment variables in General Tab. At the same place, update your dockerhub repository in the *registry* variable as shown in the snapshot below.



```
region_name=us-east-1
eks_cluster_name=big-data-pod-eks-cluster1
registry=bharatsharma1/profiler
BITBUCKET_CREDS_ID=GIT_CREDS
JENKINS_CREDS_ID=jenkinscreds
DOCKERHUB_CREDS_ID=dockerhub
REPO_CHECKOUT_FLAG=1
SCAN_CHANGES_FLAG=1
TRUFFLEHOG_FLAG=1
SONARQUBE_FLAG=1
PYLINT_FLAG=1
VERIFY_THRESHOLD_FLAG=1
UNIT_TESTING_FLAG=1
BUILD_IMAGE_FLAG=1
PUSH_IMAGE_FLAG=1
REMOVE_IMAGE_FLAG=1
DEPLOY_K8_FLAG=1
HEALTH_CHECK_API_FLAG=1
MERGE_REQUEST_FLAG=1
```

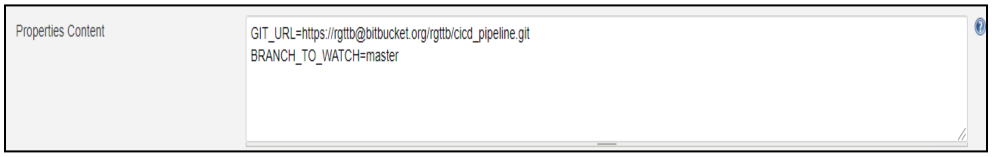
Fig 4.1: Jenkins Configuration Snippet #1

b. Also, there are certain flag values (as shown in snapshot above) for each stage of pipeline that are by default set to 1 and can be modified as per the requirement. In case you want a stage to run in the pipeline, the flag value for that stage needs to be 1, else make it 0.

c. Add the Repository URL and Git credentials in the Pipeline Tab.

2. Setting the thresholds for code quality check:
   Allowed bugs, vulnerabilities and code smells can be updated in the ***common_constant*** file present in Jenkins subdirectory of DevOps directory of git repository.

- **Configurations for scan_changes_ci_cd Job:**
  Click on configure link after going into *scan_changes_ci_cd* job:
  a. Update *GIT_URL, BRANCH_TO_WATCH* in the Properties Content section of Environment variables in General Tab as shown in the snapshot below.



```
GIT_URL=https://rgttb@bitbucket.org/rgttb/cicd_pipeline.git
BRANCH_TO_WATCH=master
```

Fig 4.2: Jenkins Configuration Snippet #2

b. Set your credentials for Git and Jenkins in the Bindings section of Build Environment Tab.

4.3 Steps to access the Jenkins workspace:

1. Enter the container through interactive mode using the command: docker exec -it container_name bash

2. Go to the path: /var/lib/jenkins/workspace/

4.4 Description:

1. Jenkins job *scan_changes_ci_cd* checks every five minutes if there is any open pull request.
2. If *scan_changes_ci_cd* job finds any open pull request, it triggers *CICD_pp* Jenkins job and merges the pull request after performing all the necessary checks/tests.

CHAPTER-5
PERFORMANCE ANALYSIS

5.1 PERFORMANCE

The pipeline gives an appreciable performance in testing, building and deploying the image of the utility. Also**, it takes around 140s for the complete pipeline to merge the pull request from the source branch to the master branch.**

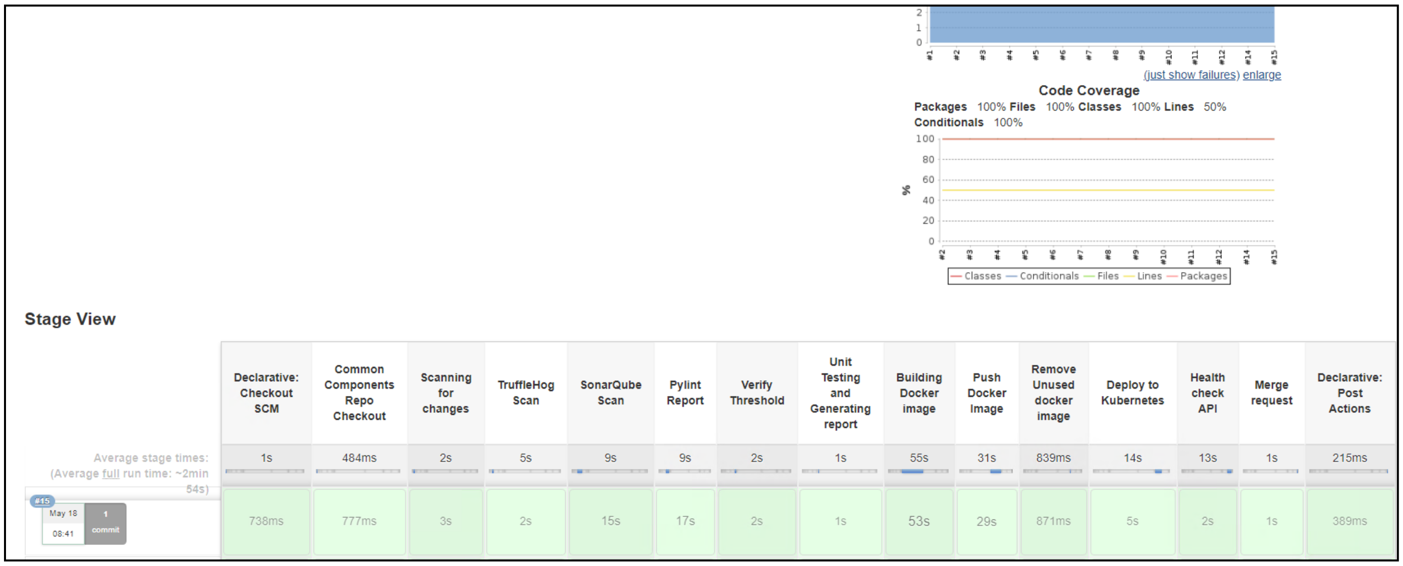
The pipeline is able to well check the file for all the conditions under the trufflehog scan. There are different secret configurations that has been set which the truffle hog checks for wrapped in a json file. If this trufflehog scan is successful and no credentials are found then the code performs sonarqube scan to check for code bugs, vulnerabilities and code smells.

If the bugs, vulnerabilities and code smells are lesser than set thresholds in sonarqube scan, then the Pylint stage comes into play.

Pylint assigns each python file a score on the basis of standards followed while writing the code. After this a final pylint score is generated out of all the python files present in the repository.

The pipeline doesn’t run all the further stages if any of the stage fails due to any reason in between.

If all of these checks are successful, then only docker image of the utility is created which is then deployed to Kubernetes, finally causing the pipeline to merge the pull request and generate the Cobertura repor which are then published at the same page.



Code Coverage

Packages 100% **Files** 100% **Classes** 100% **Lines** 50%
Conditionals 100%

Stage View

	Declarative: Checkout SCM	Common Components Repo Checkout	Scanning for changes	TruffleHog Scan	SonarQube Scan	Pylint Report	Verify Threshold	Unit Testing and Generating report	Building Docker image	Push Docker Image	Remove Unused docker image	Deploy to Kubernetes	Health check API	Merge request	Declarative: Post Actions
Average stage times: (Average full run time: ~2min 54s)	1s	484ms	2s	5s	9s	9s	2s	1s	55s	31s	839ms	14s	13s	1s	215ms
#15 May 18 08:41 1 commit	738ms	777ms	3s	2s	15s	17s	2s	1s	53s	29s	871ms	5s	2s	1s	389ms

Fig 5.1: Jenkins CI/CD Pipeline

CHAPTER-6
CONCLUSION

6.1 CONCLUSION

Jenkins, related dependencies and configurations have already been packed into the image. Once the container is launched, Jenkins runs inside the container successfully. Also, the pipeline runs end to end to perform all the necessary checks, creation and deployment of the image of the utility.

The pipeline progress can be viewed in a better way using the Blue Ocean Plugin which is already installed.

The final reports are well presented using the Cobertura Plugin after the execution of the pipeline.

REFERENCES

- **What is DevOps by microfocus.com**
- **Continuous Integration in DevOps** ***by C. Aaron Cois*** *https://insights.sei.cmu.edu/devops/2015/01/continuous-integration-in-devops-1.html*
- **Analytics India:** *https://analyticsindiamag.com/5-reasons-why-jenkins-is-the-most-used-open-source-tool-by-developers/*
- **ZS Learning Material**
- **Some YouTube Videos**

rohit report 23 05 20 1

ORIGINALITY REPORT

4%	3%	2%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.heroku.com Internet Source	2%
2	codegist.net Internet Source	1%
3	Sathyajith Bhat. "Practical Docker with Python", Springer Science and Business Media LLC, 2018 Publication	1%

Exclude quotes	Off	Exclude matches	< 14 words
Exclude bibliography	On		

RAVINDARA BHATT

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 14/07/2020

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | [✓ B.Tech Project Report] | Paper

Name: Rohit Garg **Department:** CSE **Enrolment No** 161297

Contact No. 9882985956/7018945195 **E-mail.** rhtgarg253@gmail.com

Name of the Supervisor: Dr. Ravindara Bhatt

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): DOCKERIZING JENKINS AND ONE CLICK DEPLOYMENT WITH CI/CD PIPELINE

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 43
- Total No. of Preliminary pages = 9
- Total No. of pages accommodate bibliography/references = 1

R Garg

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ……………….(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

………

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com