DevOps Project Report

-Aditya Ankush

-Shiloni Pipalia

-Suraj Singh

1. Introduction

A. Why DevOps?

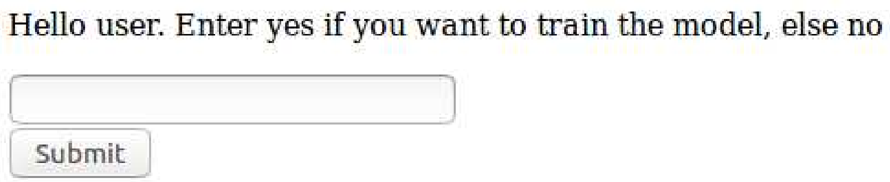
Adopting DevOps strategies leads to benefits such as:

- Shorter development cycles, Faster innovation
- Reduced deployment failures, rollbacks, and recovery time
- Improvement in communication leading to more collaboration
- Increase in efficiency
- Reduced cost and manpower

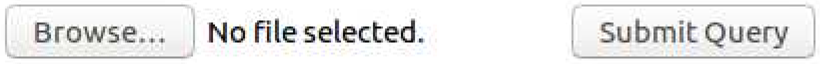
B. About the Application

The Application is an ML based project for handwriting recognition of numeral characters using the MNIST dataset. The application is a web-based portal where the user can choose to train or test the model, upon training the trained model is saved on the user's system and is utilized for testing until overwritten by a new train call. The output is finally saved in the form of a csv file stored on the user's system.

Choose whether to train or test(IMG 1):


Hello user. Enter yes if you want to train the model, else no

Upload Train/Test file(IMG 2):


No file selected.

2. Software Development Lifecycle

A. Scope of the Project

The scope of the project is to build a ML based handwriting recognition tool and use DevOps tools for deploy, integrate, test and monitor. Our focus was on using DevOps tools. In order to achieve our goal, we have successfully used DevOps tools like GitHub, Jenkins, docker, Rundeck, Filebeat and ELK.

B. Project architecture, workflows

We tried to build a completely automated pipeline. The stages included in the pipeline are:

- Clone/pull from GitHub.
- Build the docker image
- Push to Docker hub.
- Trigger Rundeck for deployment (More information present in the Rundeck section).
- ELK for monitoring

C. SCM

The source control tool used as mentioned above is GitHub. We linked the Jenkins pipeline from the Source Control Management option and linked it to the git repository. Whenever a change is pushed to git, the SCM polling in Jenkins keeps checking for updates and will find one when an update is made. Some of the advantages of Git are as follows:

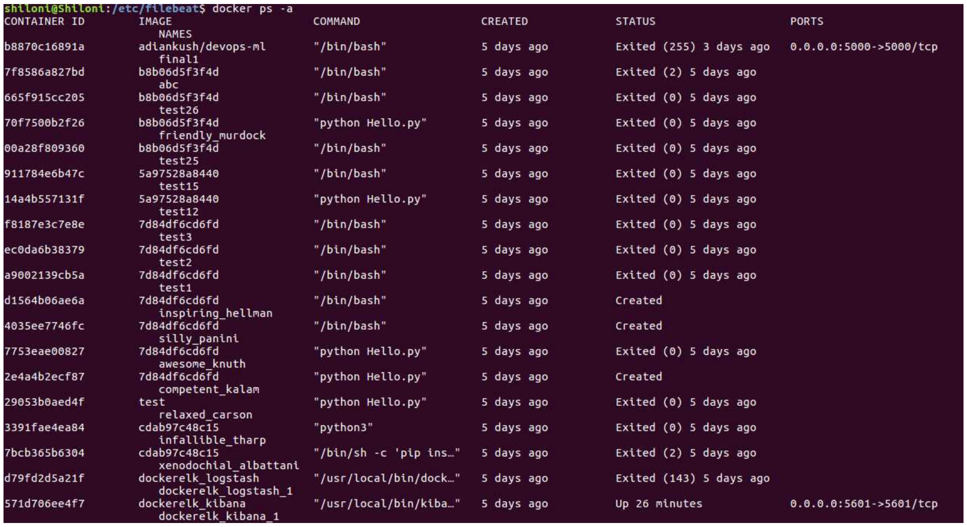
- Git is one of the most widely-used version control systems in use today. Git is Open Source and helps us implement access control.
- The project can be worked on simultaneously by multiple team members using branches. It is easy to understand and merge the branches.
- Git is faster for performing network operations such as download and upload of project files to the file server.
- Git supports projects up to 1 Gb. Since our application wouldn’t cross a GB, we preferred to use this instead of other SCM tools like GitLab, BitBucket etc.

D. Artifact

An artifact is one of many by-products produced during the software development.

The artifact of our project was one docker image of the application itself which was pushed to DockerHub.

Image pushed to DockerHub(IMG 3):



```
shiloni@Shiloni:/etc/filebeat$ docker ps -a
CONTAINER ID        IMAGE                     COMMAND                  CREATED             STATUS                    PORTS
                       NAMES
b8870c16891a        adiankush/devops-ml       "/bin/bash"              5 days ago          Exited (255) 3 days ago   0.0.0.0:5000->5000/tcp
                       final1
7f8586a827bd        b8b06d5f3f4d              "/bin/bash"              5 days ago          Exited (2) 5 days ago
                       abc
665f915cc205        b8b06d5f3f4d              "/bin/bash"              5 days ago          Exited (0) 5 days ago
                       test26
70f7500b2f26        b8b06d5f3f4d              "python Hello.py"        5 days ago          Exited (0) 5 days ago
                       friendly_murdock
00a28f809360        b8b06d5f3f4d              "/bin/bash"              5 days ago          Exited (0) 5 days ago
                       test25
911784e6b47c        5a97528a8440              "/bin/bash"              5 days ago          Exited (0) 5 days ago
                       test15
14a4b557131f        5a97528a8440              "python Hello.py"        5 days ago          Exited (0) 5 days ago
                       test12
f8187e3c7e8e        7d84df6cd6fd              "/bin/bash"              5 days ago          Exited (0) 5 days ago
                       test3
ec0da6b38379        7d84df6cd6fd              "/bin/bash"              5 days ago          Exited (0) 5 days ago
                       test2
a9002139cb5a        7d84df6cd6fd              "/bin/bash"              5 days ago          Exited (0) 5 days ago
                       test1
d1564b06ae6a        7d84df6cd6fd              "/bin/bash"              5 days ago          Created
                       inspiring_hellman
4035ee7746fc        7d84df6cd6fd              "/bin/bash"              5 days ago          Created
                       silly_panini
7753eae00827        7d84df6cd6fd              "python Hello.py"        5 days ago          Exited (0) 5 days ago
                       awesome_knuth
2e4a4b2ecf87        7d84df6cd6fd              "python Hello.py"        5 days ago          Created
                       competent_kalam
29053b0aed4f        test                      "python Hello.py"        5 days ago          Exited (0) 5 days ago
                       relaxed_carson
3391fae4ea84        cdab97c48c15              "python3"                5 days ago          Exited (0) 5 days ago
                       infallible_tharp
7bcb365b6304        cdab97c48c15              "/bin/sh -c 'pip ins…"   5 days ago          Exited (2) 5 days ago
                       xenodochial_albattani
d79fd2d5a21f        dockerelk_logstash        "/usr/local/bin/dock…"   5 days ago          Exited (143) 5 days ago
                       dockerelk_logstash_1
571d706ee4f7        dockerelk_kibana          "/usr/local/bin/kiba…"   5 days ago          Up 26 minutes             0.0.0.0:5601->5601/tcp
                       dockerelk_kibana_1
```

Advantages:

- Simple and faster configuration.
- Docker enables you to build a container image and use that same image across every step of the deployment process.
- Rapid deployment
- Docker ensures your applications and resources are isolated and segregated.

E. Deploy

For continuous deployment we have used Rundeck. In Rundeck we create jobs by defining a single step or a workflow that can execute any set of commands, scripts, or tools on any number of local or remote nodes. These jobs can be triggered by a scheduler or on demand via the web interface or API. Advantages:

- Ability to provide inputs to jobs at runtime.
- Nodes can be physical (servers, network devices, etc.) or logical (VM, service accounts, containers, etc.) endpoints.
- Node data can be pulled from anywhere, including Chef, Puppet, Amazon EC2, OpenStack, Docker, VMware, in-house CMDBs, or even monitoring tools.

- Multiple user-friendly ways to watch your jobs execute in real time. There is a summary view, node-oriented view, workflow view, and collated log view.

F. Monitor

ELK stack stands for Elasticsearch, Logstash and Kibana. We use this tool for continuous monitoring. Elastic search is used for deep search and data analytics. It's an open source distributed NoSQL database, built in Java and based on Apache Lucene. Lucene takes care of storing disk data, indexing, and document scanning while Elastic Search keeps document updates, APIs, and document distribution between Elastic Search instances in the same cluster.

Logstash is used for centralized logging, log enrichment, and parsing. It's an ETL (Extract, Transfer, Load) tool that transforms and stores logs within Elastic Search.

Kibana is a web-based tool through which Elastic Search databases visualize and analyze stored data.

Advantages:

- Elastic search provides flexibility and scalability. It operates in real time.

- Logstash can collect data from multiple systems into a central system.

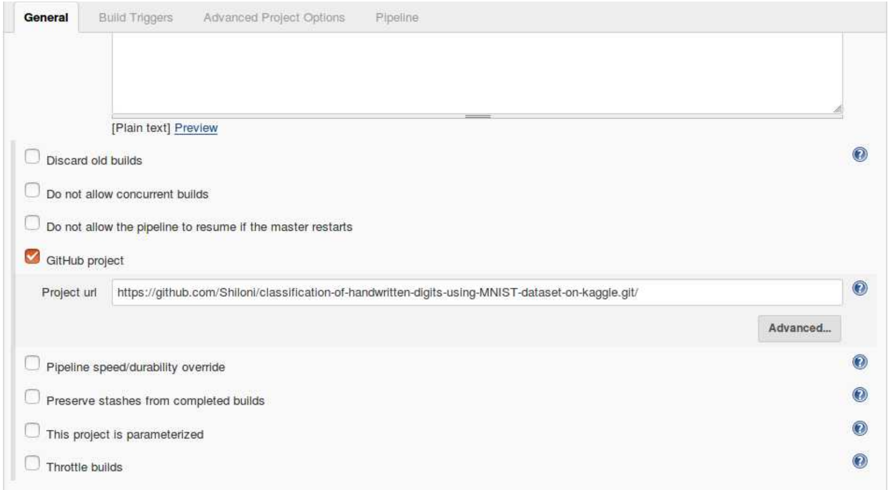
- Kibana offers powerful and easy to use features such as histograms, line graphs, pie charts, heat maps, etc.

As shown in the figure, the pie chart shows the different ids (partitions) when the website is visited. We can also visualize the pie chart with the different IP addresses that reflect that the website is visited from different devices.

3. CI/CD Pipeline

In the pipeline, we have used the SCM git option and provided it with the GitHub project repository URL (https://github.com/Shiloni/classification-of-handwritten-digits-using-MNIST-dataset-on-kaggle). We have provided it with the credentials of GitHub which were already saved in Jenkins. In the script path, we have mentioned it to use Jenkinsfile.

GitHub Repo linking(IMG 4):



□ Discard old builds

□ Do not allow concurrent builds

□ Do not allow the pipeline to resume if the master restarts

☑ GitHub project

Project url: https://github.com/Shiloni/classification-of-handwritten-digits-using-MNIST-dataset-on-kaggle.git/

□ Pipeline speed/durability override

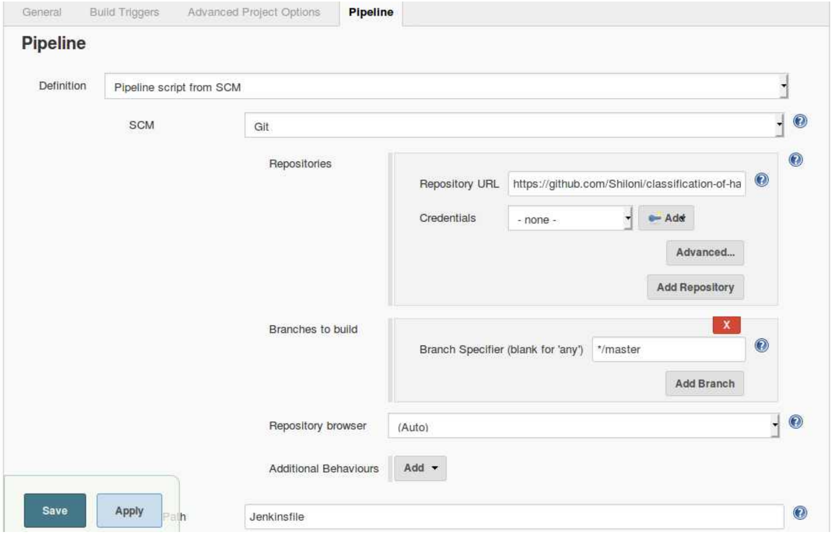
□ Preserve stashes from completed builds

□ This project is parameterized

□ Throttle builds

Once the SCM triggered, the build will start. Jenkins will search for the file ’Jenkinsfile’ because we have mentioned it in the pipeline. All the instructions which Jenkins need to provide are written in this file. Along with this, the GitHub repository along with branch name is mentioned and the credentials are also specified.

Providing Jenkinsfile path(IMG 5)



Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/Shiloni/classification-of-ha

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

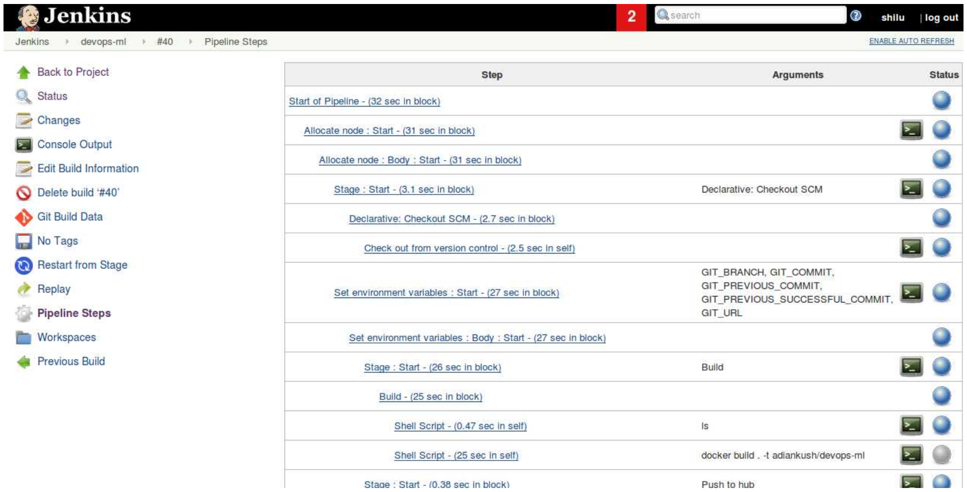
Additional Behaviours

Path: Jenkinsfile

For source code management we use GitHub which contains all the files relevant to the code including the Dockerfile and Jenkinsfile, We have used jenkins for continuous integration in which we created a pipeline project (named devops-ml) linked to the Github repository consisting of three stages:

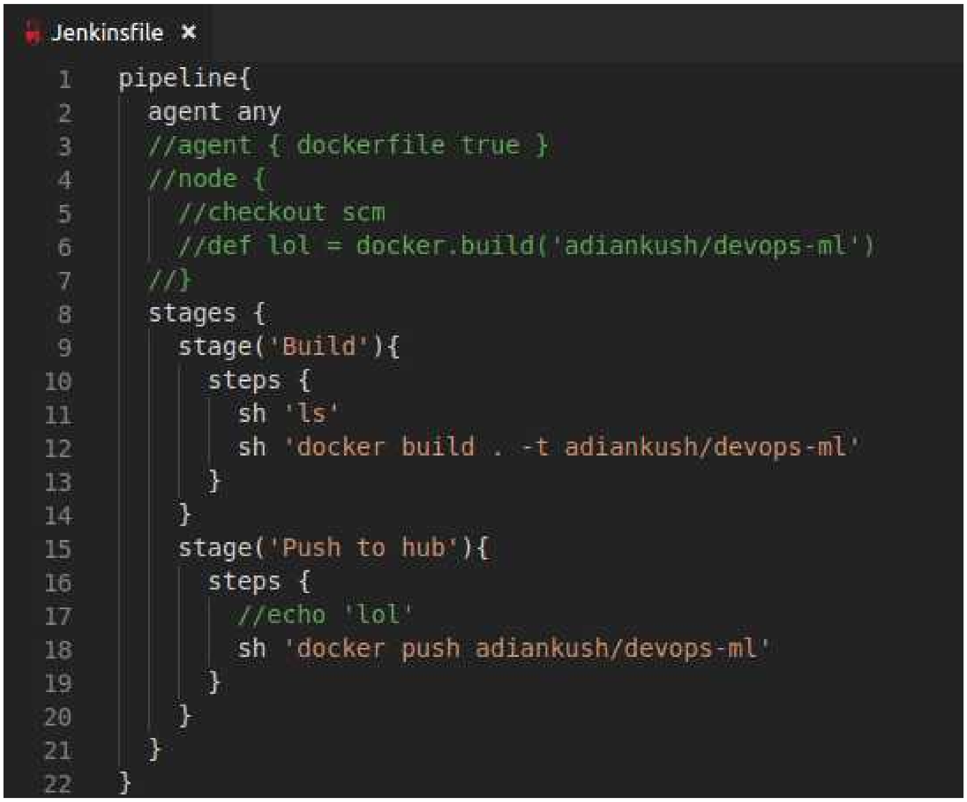
1. SCM polling to pull the files from the repository every time it is updated.
2. Building the Docker image utilizing the pulled files.
3. Pushing the Docker image to DockerHub.

Jenkins Pipeline(IMG 6):



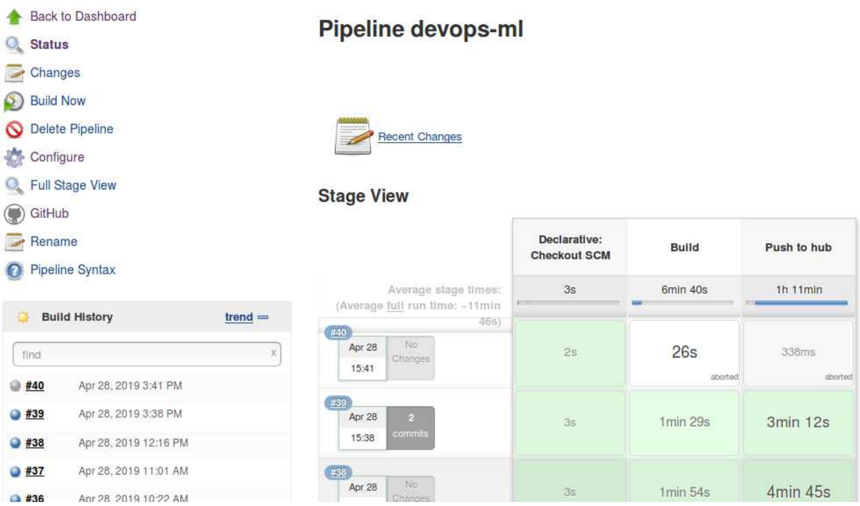
Step	Arguments	Status
Start of Pipeline - (32 sec in block)		●
Allocate node : Start - (31 sec in block)		●
Allocate node : Body : Start - (31 sec in block)		●
Stage : Start - (3.1 sec in block)	Declarative: Checkout SCM	●
Declarative: Checkout SCM - (2.7 sec in block)		●
Check out from version control - (2.5 sec in self)		●
Set environment variables : Start - (27 sec in block)	GIT_BRANCH, GIT_COMMIT, GIT_PREVIOUS_COMMIT, GIT_PREVIOUS_SUCCESSFUL_COMMIT, GIT_URL	●
Set environment variables : Body : Start - (27 sec in block)		●
Stage : Start - (26 sec in block)	Build	●
Build - (25 sec in block)		●
Shell Script - (0.47 sec in self)	ls	●
Shell Script - (25 sec in self)	docker build . -t adiankush/devops-ml	○
Stage : Start - (0.38 sec in block)	Push to hub	●

Jenkins file(IMG 7):



```
pipeline{
  agent any
  //agent { dockerfile true }
  //node {
    //checkout scm
    //def lol = docker.build('adiankush/devops-ml')
  //}
  stages {
    stage('Build'){
      steps {
        sh 'ls'
        sh 'docker build . -t adiankush/devops-ml'
      }
    }
    stage('Push to hub'){
      steps {
        //echo 'lol'
        sh 'docker push adiankush/devops-ml'
      }
    }
  }
}
```

Pipeline GUI(IMG 8):



Pipeline devops-ml

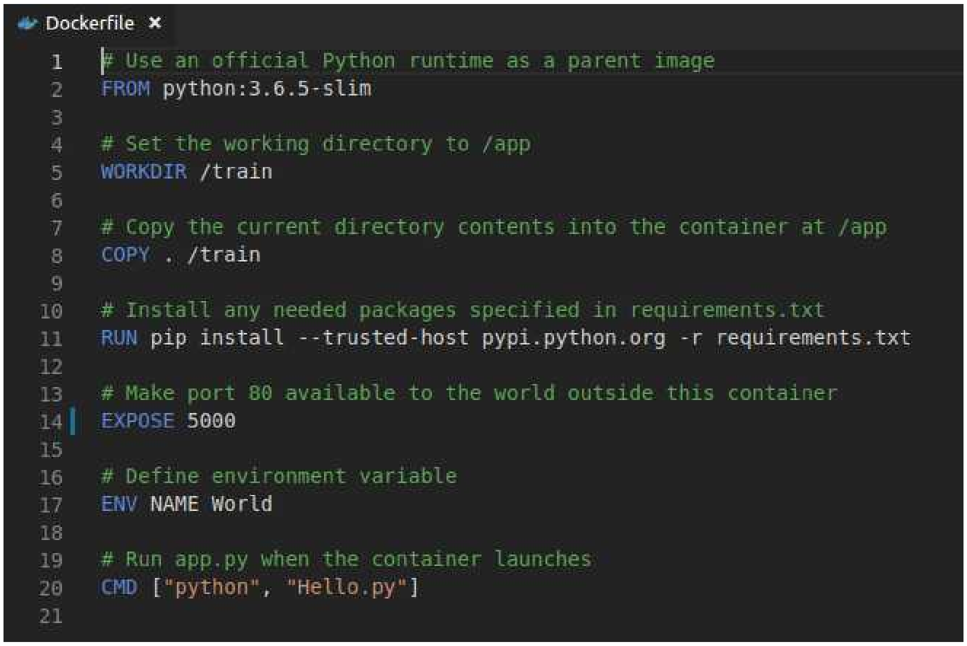
Build History

#40	Apr 28, 2019 3:41 PM
#39	Apr 28, 2019 3:38 PM
#38	Apr 28, 2019 12:16 PM
#37	Apr 28, 2019 11:01 AM
#36	Apr 28, 2019 10:22 AM

Stage View

	Declarative: Checkout SCM	Build	Push to hub
Average stage times: (Average full run time: ~11min 46s)	3s	6min 40s	1h 11min
#40 Apr 28 15:41 No Changes	2s	26s aborted	338ms aborted
#39 Apr 28 15:38 2 commits	3s	1min 29s	3min 12s
#38 Apr 28 No Changes	3s	1min 54s	4min 45s

Docker file(IMG 9):

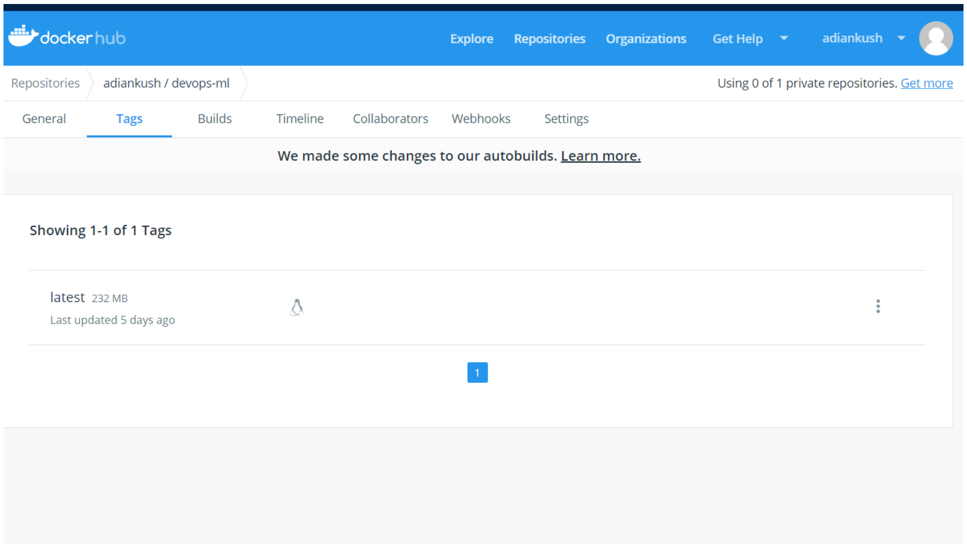


Dockerfile

```dockerfile
# Use an official Python runtime as a parent image
FROM python:3.6.5-slim

# Set the working directory to /app
WORKDIR /train

# Copy the current directory contents into the container at /app
COPY . /train

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 5000

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "Hello.py"]
```

On DockerHub we have created a project name adiankush/devops-ml where the latest artifact is pushed every time, Concluding the Continuous integration step.

DockerHub Artifact(IMG 10):



Using 0 of 1 private repositories. Get more

We made some changes to our autobuilds. Learn more.

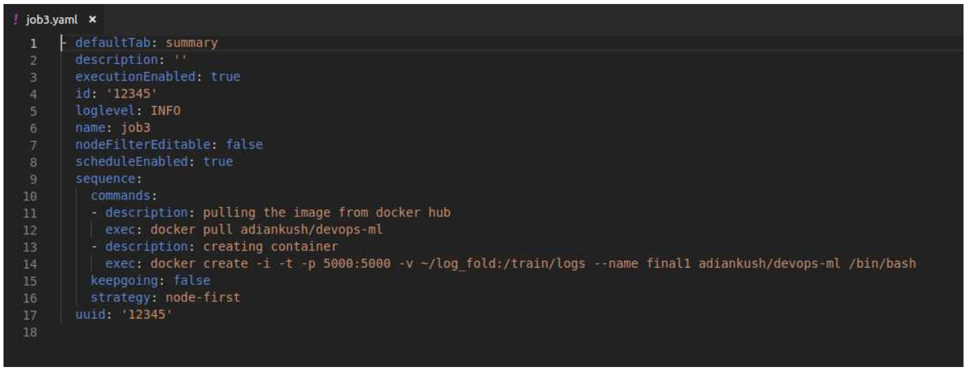
Showing 1-1 of 1 Tags

latest 232 MB
Last updated 5 days ago

Upon completion of the above pipeline a free-style project triggers as the post-build step(named rundeck-ml), This project is linked to the RunDeck profile containing a predefined job whose uid is provided while creating the project. This then calls the RunDeck job which then executes the job steps on the nodes linked to the RunDeck account, The job steps are as follows:

- Docker pull (To pull the latest image from DockerHub)
- Docker create (To create the container from pulled image on the remote nodes)

RunDeck Job(IMG 11):

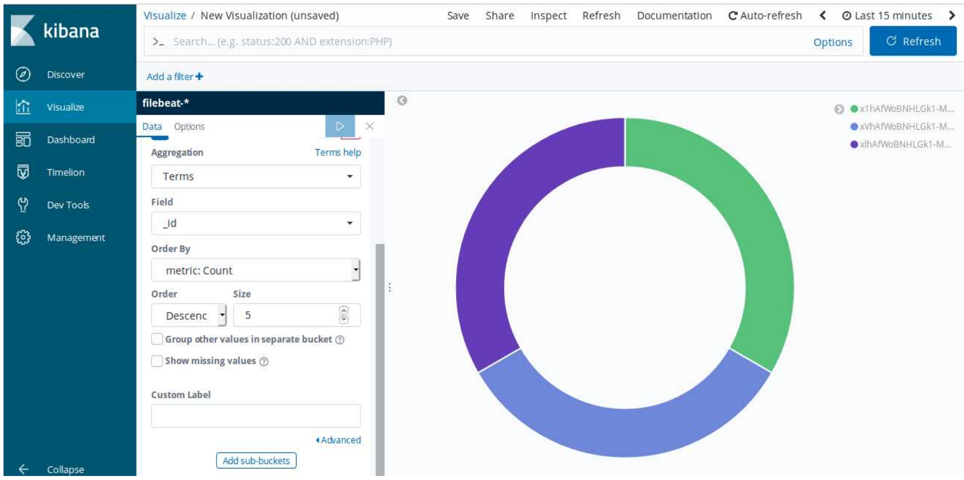


```yaml
defaultTab: summary
description: ''
executionEnabled: true
id: '12345'
loglevel: INFO
name: job3
nodeFilterEditable: false
scheduleEnabled: true
sequence:
  commands:
  - description: pulling the image from docker hub
    exec: docker pull adiankush/devops-ml
  - description: creating container
    exec: docker create -i -t -p 5000:5000 -v ~/log_fold:/train/logs --name final1 adiankush/devops-ml /bin/bash
  keepgoing: false
  strategy: node-first
uuid: '12345'
```

We have used ELK stack and Filebeat for continuous monitoring to which we provide the Log File generated by the flask python library (used to create website) within the code. The logfile is then generated within the container which we then create a copy of on the local machine using the –v(volume) extension on the Docker create step.

We provide the log file’s(copy) path to Filebeat as the config file that gets sent to Elastic search for the creation of a pattern index which can then be visualized through Kibana.

Kibana Visualization(IMG 12):



Aggregation: Terms

Field: _id

Order By: metric: Count

Order: Descenc

Size: 5

□ Group other values in separate bucket

□ Show missing values

Custom Label:

4. Results and Discussions

We were successfully completed the application and deployed it using the aforementioned tools. It can be observed in IMG 9 that all of the pipeline stages have been built without any error. The memory required for the project is 252MB. The time taken for total pipeline to complete is (Varies with internet speed): 5min 41 secs. Since we have employed the DevOps approach our application is scalable. This is because when we add/update a feature, all we need to do is, push it to GitHub and our pipeline will handle the rest. Theoretically, this should work but we cannot comment on it with confidence as our application is not public. The application performed as expected in our production environment, but this might not be true in the actual deployment environment where we can have millions of users.

5. Future Work

- We would like to make the website more user-friendly.
- The project can also be improved by having multiple visualizations based on different parameters.
- We could also allow one user to train and store multiple models simultaneously

6. Conclusion

We were successful in developing a simple ML based handwriting recognition application and deploying it using DevOps tools. The DevOps tools used were GitHub, Jenkins, Docker, Rundeck and ELK. A pipeline architecture was made, and the above-mentioned tools were integrated, and the build was automatically triggered. By using the DevOps approach, we could configure, automate, monitor and maintain the project easily.

7. Acknowledgement

We would like thank Prof. Thangaraju and all the TA's (Shayaan and Vineeth) for their endless and selfless help throughout the project. Thanks to help us grow.

8. References

https://www.rundeck.com/what-is-rundeck

https://github.com/rundeck-plugins/openssh-node-execution

https://github.com/deviantony/docker-elk

https://logz.io/learn/complete-guide-elk-stack/