

International Institute of Information Technology, Bangalore

Project Elective Report

Question, Answer and Option Generation from text data

Under the Guidance of Prof. B. Thangaraju



Pratik Pawar
MT2020121

Chetan Gulecha
MT2020052

Contents

1	Abstract	3
2	Introduction	3
3	System Configuration	4
4	Source Code and Output	4
5	Software Development Life Cycle (SDLC)	10
5.1	Source Code Management (SCM)	10
5.2	Continuous Integration (CI)	12
5.2.1	Jenkins Installation	12
5.2.2	Jenkins Pipeline	12
5.3	Continuous Delivery	15
5.3.1	Docker Installation	15
5.3.2	Building docker image	15
5.3.3	Publishing docker image	17
5.4	Continuous Deployment	18
5.4.1	Rundeck Installation	19
5.4.2	Creating Rundeck Project and Job	19
5.4.3	Jenkins Pipeline	23
5.5	Monitoring	25
5.5.1	Install and run ELK stack	25
5.5.2	Parse log file using logstash	26
5.5.3	Visualizing logs in Kibana	28
6	Future Scope and Conclusion	30
7	References	30

1. Abstract

Our project aims at generating questions, it's exact answer and creating options which are similar to answer of question (helpful in MCQ type questions). This project is useful in sense that user doesn't need to go through the whole text and he can create question and multiple choice options as well. This can also help users in summarizing the text quickly.

2. Introduction

The motive of this project is to generate multiple choice answers from text, following steps are followed to achieve desired results:

Identify keywords – Identify keywords from the given text and use them as answers to the questions.

Replace the answer – Replace the answer word/words from the sentence with blank space.

Generate option – Find words that are similar to the answer and they can be used as incorrect answers.

In this project, command line Question-Answer is created and DevOps tools are used to automate the process. Currently, the application takes text input (paragraph) , number of questions to be generated as input from user and displays output. Application can be installed in docker container environment.

What is DevOps?

DevOps is a set of practices that combines Dev and Ops. It is more over a philosophy, mindset, concept which extends agile principles beyond the boundaries to entire delivered services.

Why should one use DevOps?

- i. Improved development frequency.
- ii. Faster time to market.
- iii. Lower failure rate of new release.
- iv. Shortened lead time between fixes.
- v. Faster mean time to recovery in the event of a new release crashing.

3. System Configuration

The CI/CD pipeline is built using following tools:

1. Development
 - a. OS: Ubuntu 18.04
 - b. Language: Python
 - c. Python environment: python 3.6
 - d. SCM: Git, Github
2. Integration: Jenkins
3. Delivery: Docker, Dockerhub, Jenkins
4. Deployment: Docker, Rundeck, Jenkins
5. Monitoring: ELK stack

The entire source code can be found here:

<https://github.com/WidoutSyntax/Question-Answer-Generator>

4. Source Code and Output

The source code is written in python and after deployment phase docker image is pulled on user machine and container is running in detached mode. User needs to execute docker container and run the python script. Commands used are as follows:

- `docker exec -it question_answer_container /bin/bash`
- `python Final.py`

The program will ask for two inputs:

- 1) How many questions?
- 2) Enter the paragraph

Question, answer and options for that question will be printed as output.

Let's jump to execution of project

1) EDA on Standard Question Answer Dataset

Before we could start with the project, we wanted to know how exactly question and answers are formed from paragraphs. For this purpose the following dataset is used:

<https://www.kaggle.com/stanfordu/stanford-question-answering-dataset>

More insights on EDA can be checked at <https://github.com/WidoutSyntax/Question-Answer-Generator/blob/main/dataSQuAD.ipynb>

2) Feature engineering on SQuAD

Words from the text can be used as answers for questions. Here, spacy is used for word tagging.

Basically, each non-stop word is extracted from paragraph of each question in the SQuAD dataset and following features are added on those words:

- i. Word count
- ii. Is alpha
- iii. Named Entity Relation
- iv. Shape
- v. Part of speech

isAnswer column is also added in the data frame as answers are already known.

	text	isAnswer	titleId	paragraghId	sentenceId	wordCount	NER	POS	TAG	DEP	shape
0	Architecturally	False	0	0	0.0	1	None	ADV	RB	advmod	Xxxxx
1	school	False	0	0	0.0	1	None	NOUN	NN	nsubj	xxxx
2	Catholic	False	0	0	0.0	1	NORP	None	None	None	Xxxxx
3	character	False	0	0	0.0	1	None	NOUN	NN	dobj	xxxx
4	Atop	False	0	0	1.0	1	None	ADP	IN	prep	Xxxx

	text	isAnswer	titleId	paragraghId	sentenceId	wordCount	NER	POS	TAG	DEP	shape
21	the Main Building	True	0	0	3.0	3	ORG	None	None	None	xxx Xxxx Xxxxx
39	Saint Bernadette Soubirous	True	0	0	5.0	3	GPE	None	None	None	Xxxxx Xxxxx Xxxxx

This entire word data set of 10 titles is dumped as wordsDf.pkl (pickle format). The result gets better with more number of words from other paragraphs.

More insights on feature engineering can be checked at <https://github.com/WidoutSyntax/Question-Answer-Generator/blob/main/featureEngineering.ipynb>

3) Model training

In model training, we have to take care of categorical values. After loading of wordsDf pickle, one hot encoding is used to take care of all categorical features.

```
df.head()
```

	text	isAnswer	titleId	paragraphId	sentenceId	wordCount	shape	NER_CARDINAL	NER_DATE	NER_EVENT	...	DEP_oprd	DEP_parataxis	I
0	Architecturally	False	0	0	0.0	1	Xxxxx	0	0	0	...	0	0	
1	school	False	0	0	0.0	1	xxxx	0	0	0	...	0	0	
2	Catholic	False	0	0	0.0	1	Xxxxx	0	0	0	...	0	0	
3	character	False	0	0	0.0	1	xxxx	0	0	0	...	0	0	
4	Atop	False	0	0	1.0	1	Xxxx	0	0	0	...	0	0	

5 rows × 101 columns

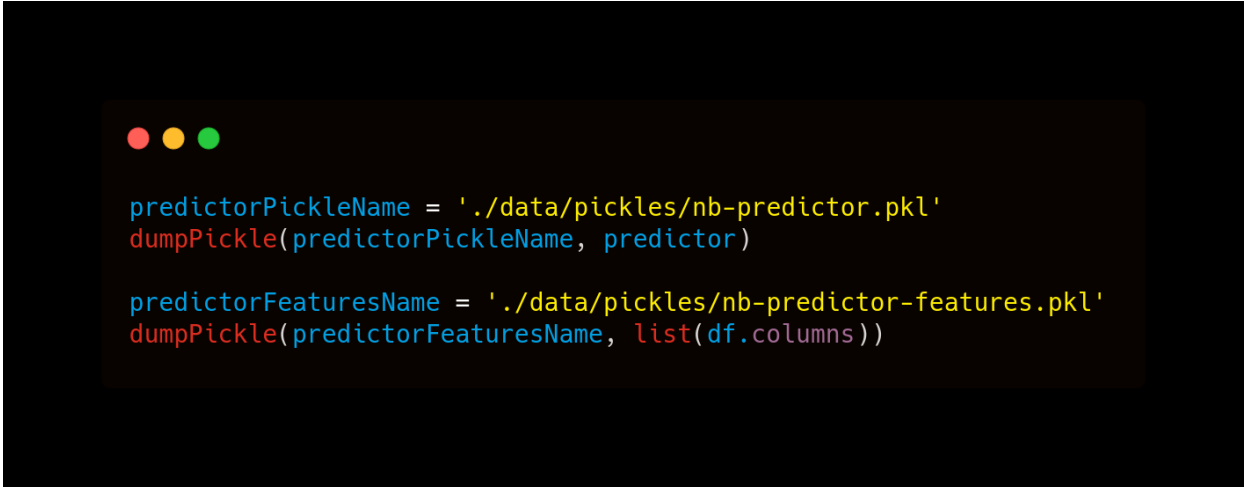
Unnecessary columns such as titleId, paragraphId, sentenceId are removed and data is split into train and test data.

```
df.head()
```

	isAnswer	wordCount	NER_CARDINAL	NER_DATE	NER_EVENT	NER_FAC	NER_GPE	NER_LANGUAGE	NER_LAW	NER_LOC	...	DEP_oprd	DEP_par
0	False	1	0	0	0	0	0	0	0	0	...	0	
1	False	1	0	0	0	0	0	0	0	0	...	0	
2	False	1	0	0	0	0	0	0	0	0	...	0	
3	False	1	0	0	0	0	0	0	0	0	...	0	
4	False	1	0	0	0	0	0	0	0	0	...	0	

5 rows × 96 columns

Gaussian Naive Bayes algorithm is used to classify each word whether it's an answer or not. Here, model is train on 10 titles data set. That model is saved as nb-predictor.pkl and features on which the model has been trained are saved as nb-predictor-features.pkl.



```
predictorPickleName = './data/pickles/nb-predictor.pkl'  
dumpPickle(predictorPickleName, predictor)  
  
predictorFeaturesName = './data/pickles/nb-predictor-features.pkl'  
dumpPickle(predictorFeaturesName, list(df.columns))
```

More insights on training can be checked at <https://github.com/WidoutSyntax/Question-Answer-Generator/blob/main/train.ipynb>

4) Creating question

The answer sentence is converted to question by simply placing a blank space in the position of answer word/words.

Code can be seen at <https://github.com/WidoutSyntax/Question-Answer-Generator/blob/main/questionBlank.ipynb>

5) Incorrect answers

Incorrect answers are generated using word embeddings. Pretrained glove embedding (glove.6B.50d.txt) model is used which finds similar words for a given word.

Note: If we use better model like glove.6B.300d.txt we will end up with better results as in if we use glove.6B.50d.txt we might not find similar words for each and every word.

```
model.most_similar(positive=['india'], topn=10)
```

```
[('indian', 0.8648794889450073),  
 ('pakistan', 0.8529723286628723),  
 ('malaysia', 0.816650927066803),  
 ('bangladesh', 0.8154239058494568),  
 ('delhi', 0.8142766952514648),  
 ('indonesia', 0.7939143180847168),  
 ('thailand', 0.7864409685134888),  
 ('sri', 0.7809487581253052),  
 ('lanka', 0.7792481780052185),  
 ('africa', 0.7728373408317566)]
```

```
model.most_similar(positive=['1944'], topn=10)
```

```
[('1943', 0.9888142943382263),  
 ('1942', 0.9829348921775818),  
 ('1941', 0.9744484424591064),  
 ('1945', 0.9693483114242554),  
 ('1940', 0.967719554901123),  
 ('1916', 0.9390915036201477),  
 ('1917', 0.9382231831550598),  
 ('1915', 0.9366674423217773),  
 ('1918', 0.9296469688415527),  
 ('1939', 0.9279517531394958)]
```

More insights on incorrect answers can be check at <https://github.com/WidoutSyntax/Question-Answer-Generator/blob/main/incorrectAnswers.ipynb>

5) Final script for user

All these models are fetched in script Final.py so that user can run only this script and get desired result.

Final output is shown below:



```
pratik@pratik-Inspiron-5559:~$ docker exec -it question_answer_container /bin/bash
root@3ad3cf2298ea:/# python Final.py
Enter number of questions you want to create: 4
Paste your paragraph here: Lamborghini was founded in 1963 by Ferruccio Lamborghini. He built tractors
after the Second World War. He founded Lamborghini because he wanted to build sports cars to compete
against Ferraris. In 1963, he released a first prototype of his ideal sports car, the 350 GTV.
```

```
Question1:
In 1963, he released a first prototype of his _____ sports car, the 350 GTV.

Answer: ideal

Options:
1.unique
2.ideal
3.suitable
4.depends
*****

Question2:
He built _____ after the Second World War.

Answer: tractors

Options:
1.tractors
2.motorcycles
3.carts
4.bicycles
*****
```

```
Question3:
He founded Lamborghini because he wanted to build sports _____ to compete against Ferraris.

Answer: cars

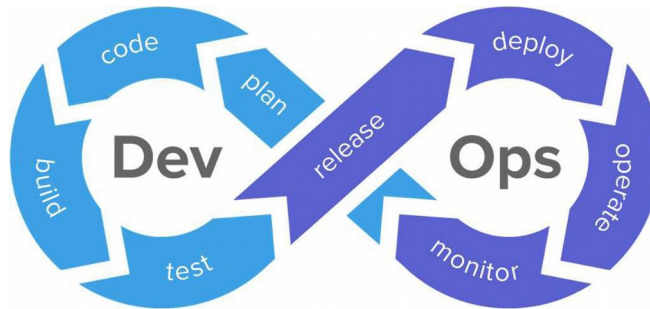
Options:
1.car
2.vehicles
3.cars
4.trucks
*****

Question4:
He founded Lamborghini because he _____ to build sports cars to compete against Ferraris.

Answer: wanted

Options:
1.want
2.why
3.did
4.wanted
*****
```

5. Software Development Life Cycle (SDLC)



There are many stages in software development life cycle. These stages are automated using various DevOps tools. There are no tools for planning as such. For code step Git and Github are used. For building docker images is used. At delivery step, docker image is built and pushed to Dockerhub. Deployment is done using Rundeck and in this this project we are deploying project's docker image on docker container. And at last, for monitoring ELK stack is used.

5.1 Source Code Management (SCM)

Source Code Management(SCM) is used to track modifications to a source code repository. SCM tracks a running history of changes that are made to code base and helps resolve conflicts when merging updates from multiple contributors. SCM is also synonyms with Version Control. Here, Git is used as SCM. Git is a distributed version control system. Github is an online repository hosting system.

The development of this project is done incrementally. Some of the important increments are as follows:

1. Add data files.
2. Add requirement file.
3. Add train and prediction notebook.
4. Add form question notebook.
5. Add Dockerfile to build image.

All other commits can be seen here:

<https://github.com/WidoutSyntax/Question-Answer-Generator/commits>

WidoutSyntax / Question-Answer-Generator

<> Code ⓘ Issues 🔗 Pull requests ⚙️ Actions 📁 Projects 📖 Wiki 🛡️ Security 📈 Insights ⚙️ Settings

main 1 branch 0 tags

Go to file Add file Code

Pratik and Pratik Add model train notebook458a18b 1 hour ago🕒 36 commits

data	Added sample text file	2 days ago
.gitattributes	Add data files	19 days ago
Dockerfile	Add Dockerfile to build image	2 days ago
Final.py	Add some code changes	1 hour ago
README.md	Add README.md	19 days ago
app.log	Add log file	1 hour ago
dataSQuAD.ipynb	Add data exploration study notebook	19 days ago
featureEngineering.ipynb	Add feature engineering notebook	19 days ago
incorrectAnswers.ipynb	Add incorrect answer model notebook	1 hour ago
logstash.conf	Added logstash file	16 hours ago
predict.ipynb	Add train and prediction notebook	19 days ago
questionBlank.ipynb	Add form question notebook	19 days ago
requirements.txt	Add requirement file	2 days ago
train.ipynb	Add model train notebook	1 hour ago

5.2 Continuous Integration (CI)

Continuous integration is a development practice where developers integrate code into shared repository frequently, preferably multiple times a day. Each integration can then be verified by an automated build and automated testing. Jenkins is used for CI. Basically, Jenkins keeps an eye on SCM system for changes in the code and builds the code as and when it detects the changes.

5.2.1 Jenkins Installation

To install Jenkins follow the step given below:

1. `wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -`
2. `sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'`
3. `sudo apt update`
4. `sudo apt install jenkins`
5. `sudo systemctl enable --now jenkins`

Jenkins will run on <http://localhost:8080/>

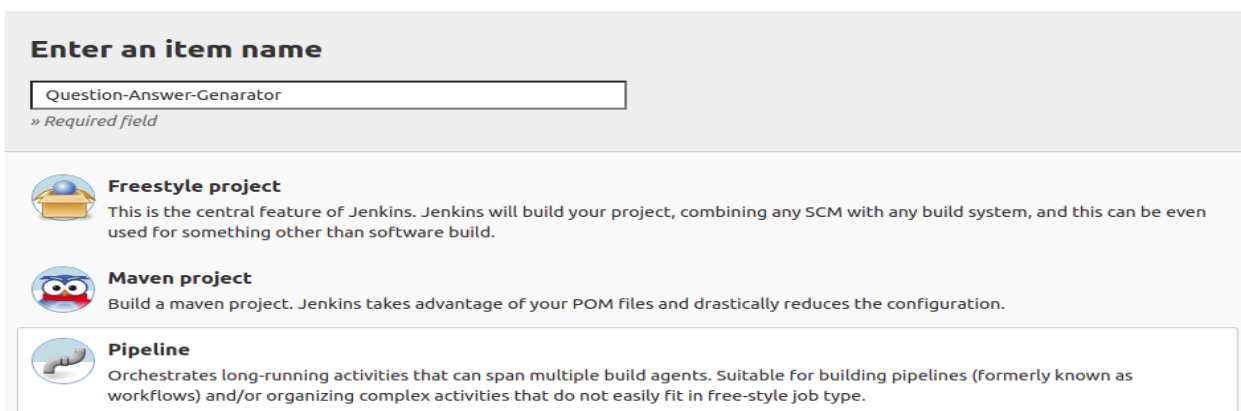
Create new user.

Install Git, Docker, Rundeck plugins.

5.2.2 Jenkins Pipeline

Follow the steps to create Jenkins pipeline:

1. Create new Jenkins pipeline as shown below

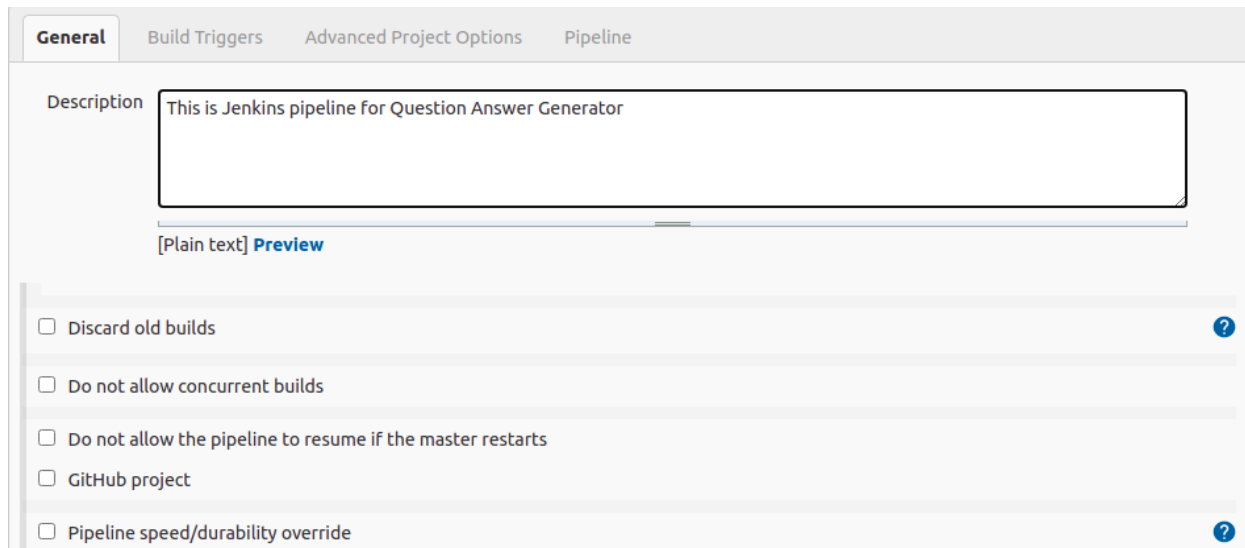


The screenshot shows the Jenkins 'New Item' page. At the top, there is a section titled 'Enter an item name' with a text input field containing 'Question-Answer-Genarator' and a note '» Required field'. Below this, there are three options for creating a new item, each with an icon and a description:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Maven project**: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

The 'Pipeline' option is highlighted with a light blue border.

2. Add some description



The image shows the 'General' tab of the Jenkins Pipeline configuration page. At the top, there are four tabs: 'General' (selected), 'Build Triggers', 'Advanced Project Options', and 'Pipeline'. Below the tabs, there is a 'Description' section with a text area containing the text 'This is Jenkins pipeline for Question Answer Generator'. Below the text area, there is a '[Plain text] Preview' link. Below the description section, there is a list of checkboxes for various options: 'Discard old builds', 'Do not allow concurrent builds', 'Do not allow the pipeline to resume if the master restarts', 'GitHub project', and 'Pipeline speed/durability override'. Each checkbox has a corresponding help icon (a blue circle with a question mark) to its right.

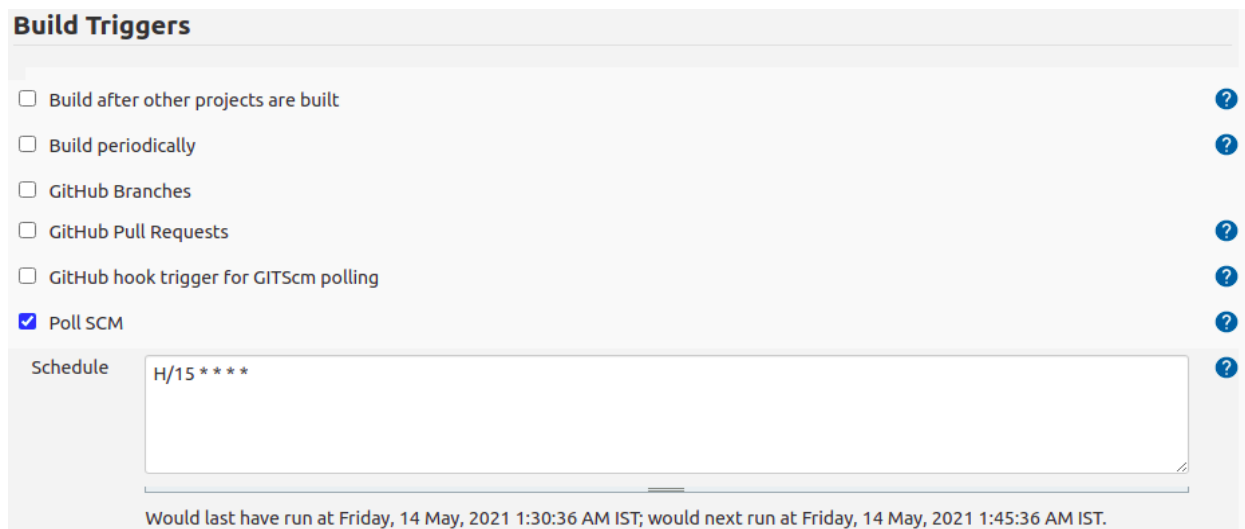
General Build Triggers Advanced Project Options Pipeline

Description This is Jenkins pipeline for Question Answer Generator

[Plain text] [Preview](#)

- ☐ Discard old builds ?
- ☐ Do not allow concurrent builds
- ☐ Do not allow the pipeline to resume if the master restarts
- ☐ GitHub project
- ☐ Pipeline speed/durability override ?

3. Choose build trigger



The image shows the 'Build Triggers' tab of the Jenkins Pipeline configuration page. At the top, there is a section header 'Build Triggers'. Below the header, there is a list of checkboxes for various build triggers: 'Build after other projects are built', 'Build periodically', 'GitHub Branches', 'GitHub Pull Requests', 'GitHub hook trigger for GITScm polling', and 'Poll SCM'. The 'Poll SCM' checkbox is checked. Each checkbox has a corresponding help icon (a blue circle with a question mark) to its right. Below the list of checkboxes, there is a 'Schedule' section with a text area containing the text 'H/15 * * * *'. Below the text area, there is a line of text: 'Would last have run at Friday, 14 May, 2021 1:30:36 AM IST; would next run at Friday, 14 May, 2021 1:45:36 AM IST.'

Build Triggers

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☐ GitHub Branches ?
- ☐ GitHub Pull Requests ?
- ☐ GitHub hook trigger for GITScm polling ?
- ☒ Poll SCM ?

Schedule H/15 * * * *

Would last have run at Friday, 14 May, 2021 1:30:36 AM IST; would next run at Friday, 14 May, 2021 1:45:36 AM IST.

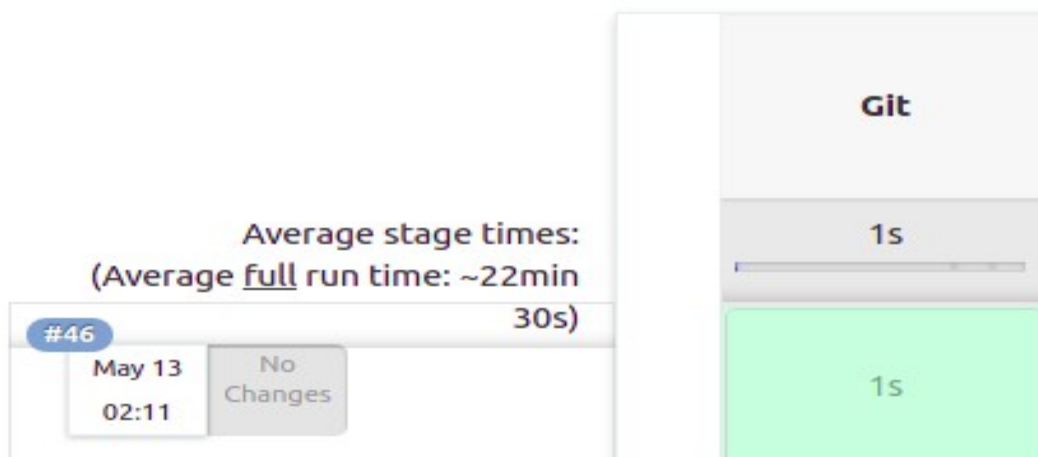
4. Add the following code in pipeline script:

```
pipeline {  
  agent any  
  
  stages {  
    stage('Git') {  
      steps {  
        // Get some code from a GitHub repository  
        git branch: 'main', url: 'https://github.com/WidoutSyntax/Question-Answer-  
Generator.git'  
      }  
    }  
  }  
}
```

5. Save the pipeline

6. Click the build now button to check everything is working fine.

Stage View



Until now, SCM pulling is automated.

5.3 Continuous Delivery

Continuous delivery is an extension of continuous integration. It makes code deployment ready for a testing or pre-production environment after the build stage. This is like automated release process.

In the delivery stage of our pipeline, docker image of application is built using Dockerfile and then that image is delivered to Dockerhub. Docker and Jenkins are used for continuous delivery.

5.3.1 Docker Installation

Docker is a tool designed to make it easier to create, deploy and run applications by using containers. Containers allow to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package. In a way, Docker is like a virtual machine but rather than creating whole virtual operating system, docker allows applications to use the same linux kernel as the system they are running on and only requires applications be shipped with things required to run application.

Follow the steps to set your system for docker:

1. To install docker
>apt-get install docker.io
2. Give permission to Jenkins to run docker commands
>sudo usermod -aG docker jenkins

5.3.2 Building docker image

A Dockerfile is a text document a user could call on the command line to assemble an image. In the docker image required python modules and python scripts are included. To build docker image, following code is added in pipeline script. To add code click on configure button.

```

pipeline {
    /* The environment specifies the credentials required to push my image to dockerhub */
    environment {
        registry = "pratikiiitb/question-answer-generator"
        registryCredential = 'dockerhub'
        dockerImage = ''
    }

    agent any

    stages {
        stage('Building docker image') {
            steps{
                script {
                    dockerImage = docker.build registry + ":latest"
                }
            }
        }
    }
}

```

Requirements.txt has all python modules required for python code.
 Requirements can be checked on github.
 Dockerfile:

```

FROM python:3
COPY ./ ./
WORKDIR ./
RUN pip install -r requirements.txt
CMD ["python", "Final.py"]

```


5.3.3 Publishing docker image

Now, pratikiiitb/question-answer-generator docker image is present on machine on which Jenkins is running.

To publish this docker image on dockerhub first sign up on dockerhub. Dockerhub is a hosted repository service provided by Docker for finding and sharing container images. Add the following script in pipeline so that Jenkins will run it after all previous scripts are successfully executed.

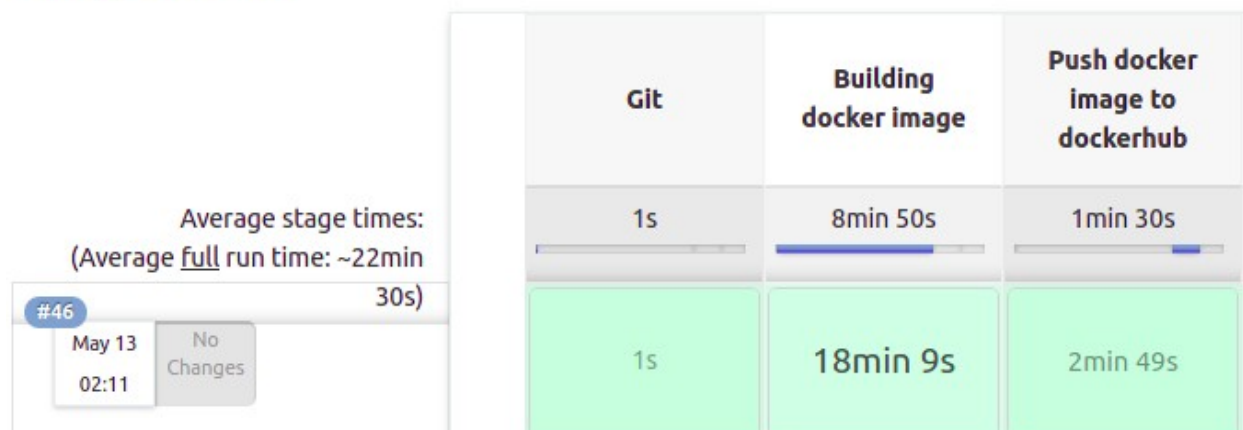
```
pipeline {
    /* The environment specifies the credentials required to push my image to dockerhub */
    environment {
        registry = "pratikiiitb/question-answer-generator"
        registryCredential = 'dockerhub'
        dockerImage = ''
    }

    agent any

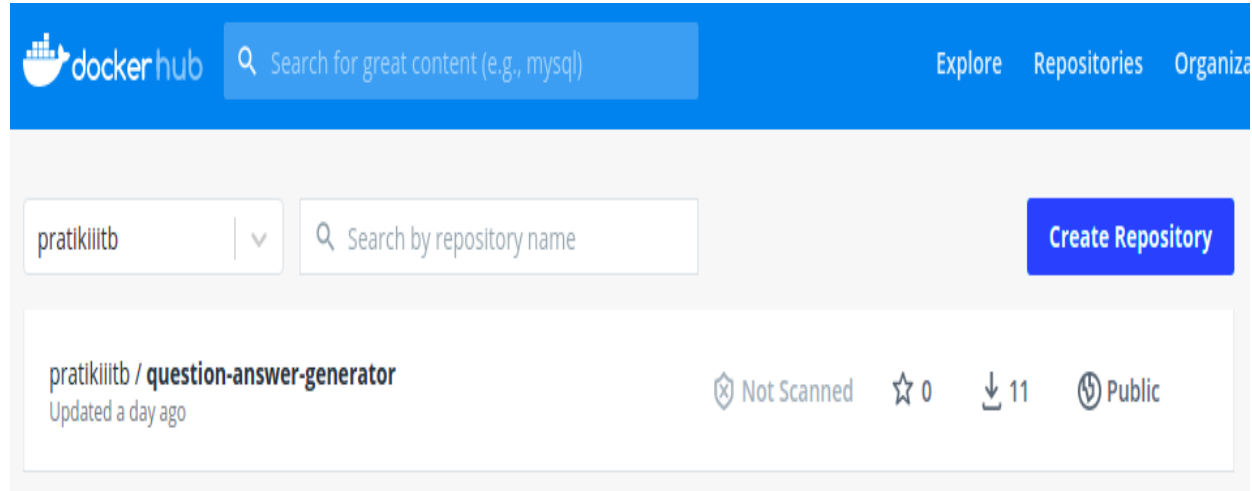
    stages {
        stage('Push docker image to dockerhub') {
            steps {
                script {
                    docker.withRegistry( '', registryCredential ) {
                        dockerImage.push()
                    }
                }
            }
        }
    }
}
```

Click build now to ensure that everything is working fine till continuous delivery

Stage View



Dockerhub:



Docker image link:

<https://hub.docker.com/repository/docker/pratikiiitb/question-answer-generator>

So far, we are done with continuous integration and continuous delivery. Now the next step is to deploy that latest docker image on managed and controlled nodes.

5.4 Continuous Deployment

Continuous deployment is one step ahead to continuous delivery. With continuous deployment every change that passes all stages of pre production pipeline is released to customers. Human intervention is not required and only a failed test will prevent a new change to be deployed to production. Continuous deployment is a way to accelerate the feedback loop with customers and take pressure off the team as there is not a Release day anymore. Developers can focus on building softwares, and their work can go live in minutes.

After the deliverable (docker image in our case) is created and published to Dockerhub, Rundeck is used to fetch the image and deploy it in a container. And the Rundeck job is invoked by Jenkins.

5.4.1 Rundeck Installation

Rundeck is a centralized console for managing the automation in environment. It allows to control who can do what and leverage existing scripts. One can interact with Rundeck using web interface, command line or API. Rundeck is an automation tool that executes Rundeck jobs on Rundeck nodes. Rundeck jobs can be thought of a sequence of instructions and Rundeck nodes could be anything like a web server, container, etc.

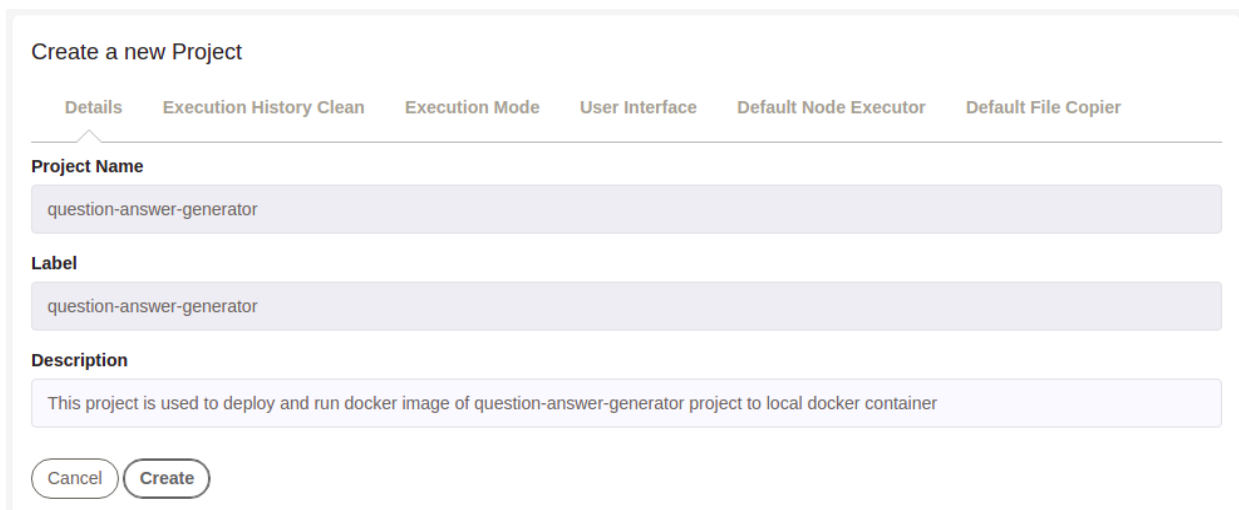
To install Rundeck, follow the steps:

1. Rundeck requires java 8 on your system check for java 8 if not present change to java 8 environment.
2. `>echo "deb https://rundeck.bintray.com/rundeck-deb /" | sudo tee -a /etc/apt/sources.list.d/rundeck.list`
3. `>curl 'https://bintray.com/user/downloadSubjectPublicKey?username=bintray' | sudo apt-key add -`
4. `>sudo apt-get update`
5. `>sudo apt-get install rundeck`
6. `>sudo service rundeckd start`

Rundeck runs on <http://localhost:4440/> and default username and password is admin and admin.

5.4.2 Creating Rundeck Project and Job

1. Create new Rundeck project



The screenshot shows the 'Create a new Project' form in the Rundeck web interface. At the top, there are tabs for 'Details', 'Execution History Clean', 'Execution Mode', 'User Interface', 'Default Node Executor', and 'Default File Copier'. The 'Details' tab is selected. Below the tabs, there are three input fields: 'Project Name' with the value 'question-answer-generator', 'Label' with the value 'question-answer-generator', and 'Description' with the value 'This project is used to deploy and run docker image of question-answer-generator project to local docker container'. At the bottom of the form, there are two buttons: 'Cancel' and 'Create'.

2. Create new Rundeck job by clicking on jobs tab on left side and do the following configuration settings

Create New Job: Deploy image to container Upload Definition

Details Workflow Nodes Schedule Notifications Other

Job Name Deploy image to container deployment Choose

Description Edit

1	This job is used to do following tasks:
2	1. Pull the question-answer-generator docker image from dockerhub
3	2. Deploy and run it in local container

The first line of the description will be shown in plain text, the rest will be rendered with Markdown. [More...](#)

Cancel Create

3. Workflow

Create New Job: Deploy image to container Upload Definition

Details Workflow Nodes Schedule Notifications Other

Options Undo Redo

No Options

+ Add an option

Workflow If a step fails:

☒ Stop at the failed step. ☐ Run remaining steps before failing.

Strategy: Node First





Execute all steps on a node before proceeding to the next node.

Explain >

4. Add following steps

Global Log Filters + add

Undo Redo Revert All Changes

1.  `docker rm -f question_answer_container`
[Remove previous container](#) ⚙️ ✖️ 📄 ⬆️
2.  `docker rmi -f pratikiitb/question-answer-generator:latest`
[Remove previous image](#) ⚙️ ✖️ 📄 ⬆️
3.  `docker pull pratikiitb/question-answer-generator:latest`
[Pull latest docker image from dockerhub](#) ⚙️ ✖️ 📄 ⬆️
4.  `docker run -t -d --name question_answer_container pratikiitb/question-answer-generator:latest`
[Run the question_answer_geerator container in detach mode](#) ⚙️ ✖️ 📄 ⬆️

+ Add a step

5. Nodes

Create New Job: Deploy image to container

Details Workflow **Nodes** Schedule Notifications Other

Nodes ☐ Dispatch to Nodes ☒ Execute locally

Choose whether the Job will run on filtered nodes or only on the local node.

Cancel

Create

6. Note the UUID of the job

📁 deployment

Deploy image to container

This job is used to do follwing tasks: [Less](#)

1. Pull the question-answer-generator docker image from dockerhub
2. Deploy and run it in local container

f60168bc-8701-4357-92c3-00c7ffe98efb

7. Run this job in Rundeck to check everything is working fine.

deployment

✓ Deploy image to container

This job is used to do following tasks: [More...](#)

Succeeded

0.02:45 at 1:37 am

you

#70

Run Again

Log Output

100% 1/1	COMPLETE	0 FAILED	0 INCOMPLETE	0 NOT STARTED
Node			Start time	Duration
localhost	All Steps OK			0.00:04
> Remove previous container	OK		1:34:18 am	0.00:00
> Remove previous image	OK		1:34:19 am	0.00:03
> Pull latest docker image from dockerhub	OK		1:34:22 am	0.02:36
> Run the question_answer_geerator container in detach mode	OK		1:36:59 am	0.00:04

```
pratik@pratik-Inspiron-5559:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
pratikiiitb/question-answer-generator	latest	1023d55725fd	21 hours ago	1.95GB
python	3	a6a0779c5fb2	27 hours ago	886MB

```
pratik@pratik-Inspiron-5559:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
d96ba7721846	pratikiiitb/question-answer-generator:latest	"python Final.py"	About a minute ago	Up About a minute		question_answer_container
60bb3ff2ae2f	ad08e37032f9	"java -jar Scientifi.."	2 months ago	Exited (143) 2 months ago		calcontainer

```
pratik@pratik-Inspiron-5559:~$
```

Now this Rundeck job should be triggered from Jenkins.

5.4.3 Jenkins Pipeline

Follow the steps given below to trigger the Rundeck job from Jenkins.

1. Add Rundeck instance in Jenkins
(Manage Jenkins → Configure System → Add Rundeck)

Name

question-answer-generator

URL

http://localhost:4440

?

Login

admin

?

Password

.....

?

Auth Token

?

API Version

?

Your Rundeck instance is alive, and your credentials are valid !

Test Connection

Delete Rundeck

Add Rundeck

List of Rundeck instances

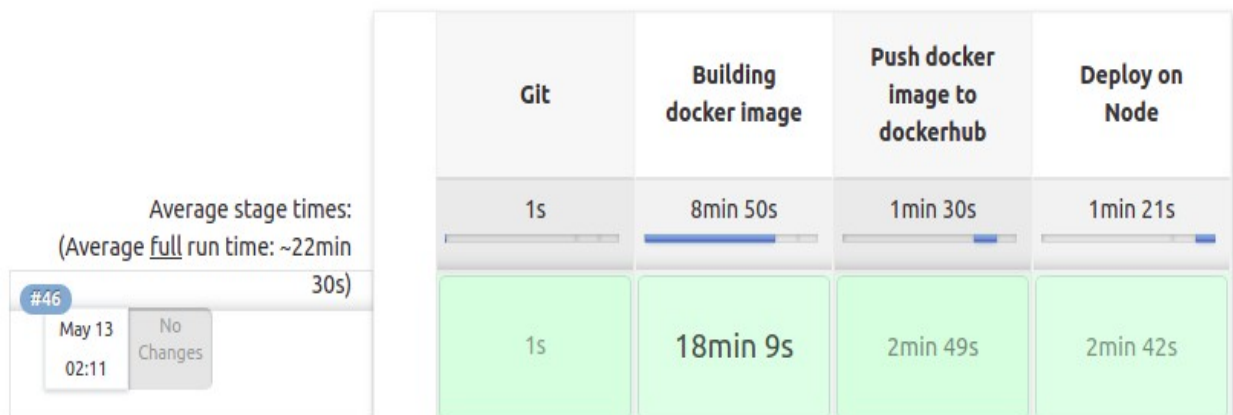
2. Add the following script in pipeline script

```
stage('Deploy on Node') {
    steps {
        script {
            step([
                $class: "RundeckNotifier",
                includeRundeckLogs: true,
                rundeckInstance: "question-answer-generator",
                jobId: "f60168bc-8701-4357-92c3-00c7ffe98efb",
                shouldWaitForRundeckJob: true,
                shouldFailTheBuild: true,
                tailLog: true
            ])
        }
    }
}
```

3. Click on Save.

Final pipeline view

Stage View



1. Developers push the code on Github
2. Jenkins pull the code periodically and do further steps in there are any new commits.
3. Jenkins trigger the job of building docker image.
4. Push that docker image on Dockerhub.
5. Jenkins trigger Rundeck job to deploy docker image on container.

5.5 Monitoring

Monitoring provides feedback from production and delivers information about an performance and usage patterns. When there is any performance or other issues detected, relevant data about the issues are sent back to development teams through automated monitoring.

Here, ELK stack is used for monitoring.

Elasticsearch is a modern search and analytics engine which is based on Apache Lucene. It is used as to store and index logs and can be then queried to extract meaningful insights. It can be used for numerous types of data including textual, numerical, geospatial, structured, and unstructured.

Logstash is a tool that is used for parsing logs. It is very useful in parsing unstructured logs and giving them structure so that logs can be efficiently searched and analyzed. Log aggregated and processed by Logstash go through 3 stages – collection, processing and dispatching.

Kibana adds a visualization layer to the Elastic Stack. It is a browser-based user interface that can be used to search, analyze and visualize the data stored in Elasticsearch indices.

elasticsearch, logstash, kibana can be downloaded and run locally or elasticsearch and kibana can be run on cloud and logs can be sent using logstash which runs on local environment. We used the later approach.

5.5.1 Install and run ELK stack

Download Elasticsearch archive for your OS:

- I. Download elastic search:
<https://www.elastic.co/downloads/elasticsearch>
- II. Download Logstash:
<https://www.elastic.co/downloads/logstash>
- III. Download Kibana:
<https://www.elastic.co/downloads/kibana>

You can extract the tar.gz file using following commands

- `tar -cvf path/to/file.tar.gz`

To run elasticsearch run the following command after you change directory to elasticsearch

- `./bin/elasticsearch`

Elasticsearch runs on port number 9200: <http://localhost:9200>

Run kibana next, run the following command after you change directory to kibana

- `./bin/kibana`

Finally run logstash using following commands

- `./bin/logstash -f pathto/configuration/file`

5.5.2 Parse log file using logstash

Different plugins are used for Logstash filters to derive structure out of unstructured data. It helps you to define a search and extract parts of your logline into structured fields.

We used logging module in python to collect logs. When the program is being run on any node the logs get collected in app.log file.

1. So, after deployment of docker image to container run program in container log file app.log will get generated
2. Copy log file on server side i.e., localhost in my case
>`docker cp <container_name>:app.log /home/chetan/app.log`
3. Run the logstash using following command
>`bin/logstash -f /home/logstash.conf`

logstash.conf is a configuration file. Output is given to elasticsearch running locally. Note index pattern `question_generator` is used as index , we will require this in kibana.

```
chetan@chetan-HP-Pavillon-Notebook: ~/Downloads/logstash-7.11.1
[2021-05-14T10:43:09,247][INFO ][logstash.javapipeline ][main] Pipeline started {"pipeline.id"=>"main"}
[2021-05-14T10:43:09,400][INFO ][filewatch.observingtail ][main][1602f333ed91f7da6df9bd6b753507aeddcc1eb9273bd08ffad4a034840840b2e] START, crea
ting Discoverer, Watch with file and sinedb collections
[2021-05-14T10:43:09,409][INFO ][logstash.agent ] Pipelines running {:count=>1, :running_pipelines=>[:main], :non_running_pipelines=>[:]}
[2021-05-14T10:43:09,886][INFO ][logstash.agent ] Successfully started Logstash API endpoint {:port=>9600}

{"level" => "INFO",
 "path" => "/home/chetan/app.log",
 "@timestamp" => 2021-05-11T04:22:49.647Z,
 "host" => "chetan-HP-Pavillon-Notebook",
 "message" => "11/May/2021:09:52:49 647 [root] [INFO] - 4",
 "count" => 4,
 "@version" => "1",
 "user" => "root"}

{"level" => "INFO",
 "path" => "/home/chetan/app.log",
 "@timestamp" => 2021-05-11T05:22:45.123Z,
 "host" => "chetan-HP-Pavillon-Notebook",
 "message" => "11/May/2021:10:52:45 123 [root] [INFO] - 4",
 "count" => 4,
 "@version" => "1",
 "user" => "root"}

{"level" => "INFO",
 "path" => "/home/chetan/app.log",
 "@timestamp" => 2021-05-11T05:29:09.471Z,
 "host" => "chetan-HP-Pavillon-Notebook",
 "message" => "11/May/2021:10:59:09 471 [root] [INFO] - 2",
 "count" => 2,
 "@version" => "1",
 "user" => "root"}

{"level" => "INFO",
 "path" => "/home/chetan/app.log",
 "@timestamp" => 2021-05-11T04:17:49.152Z,
 "host" => "chetan-HP-Pavillon-Notebook",
 "message" => "11/May/2021:09:47:49 152 [root] [INFO] - 3",
```

```
input {
  file {
    path => "/home/chetan/app.log"
    start_position => "beginning"
  }
}

filter {
  grok {
    match => [
      "message", "%{HTTPDATE:timestamp_string} \\[%{GREEDYDATA:user}\\] \\[%{LOGLEVEL:level}\\] \\- %{GREEDYDATA:count}"
    ]
  }

  date {
    match => ["timestamp_string", "dd/MMM/YYYY:HH:mm:ss SSS"]
  }

  mutate {
    remove_field => [timestamp_string]
  }
}

output {
  elasticsearch {
    hosts => ["http://localhost:9200"]
    index => "question_generator"
  }

  stdout {
    codec => rubydebug
  }
}
```

5.5.3 Visualizing Logs in Kibana

Defining an index pattern

In Kibana, go to Management → Stack Management

Look for Kibana → Index Patterns → Create Index Pattern

Give the index pattern mentioned in the logstash config. file

And in the next step select @timestamp as your Time field

Hit Create index pattern

Step 1 of 2: Define an index pattern

Index pattern name

question_generator

Next step >

Use an asterisk (*) to match multiple indices. Spaces and the characters \, /, ?, ", <, >, | are not allowed.

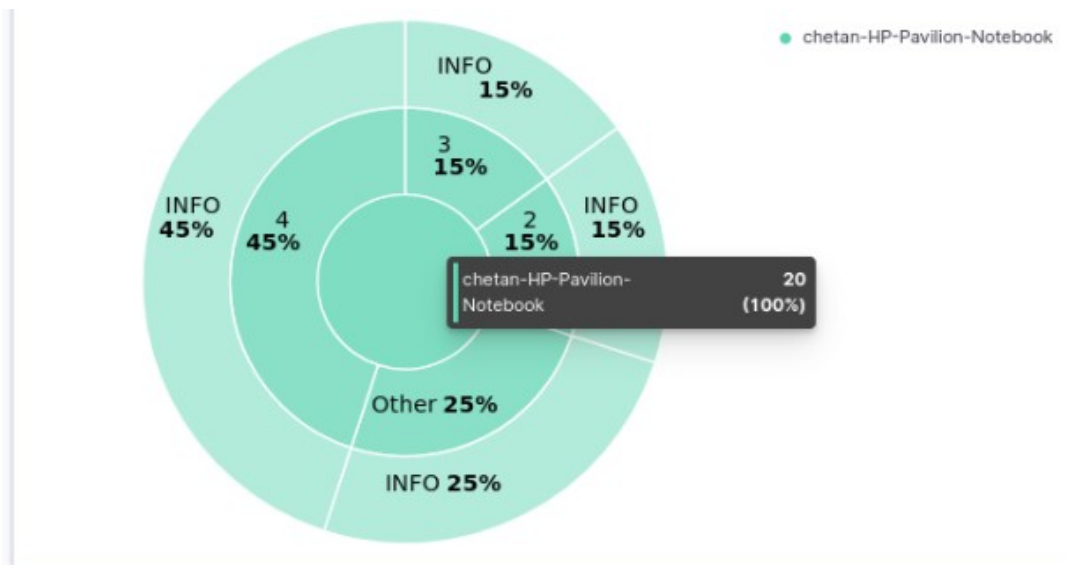
☐ Include system and hidden indices

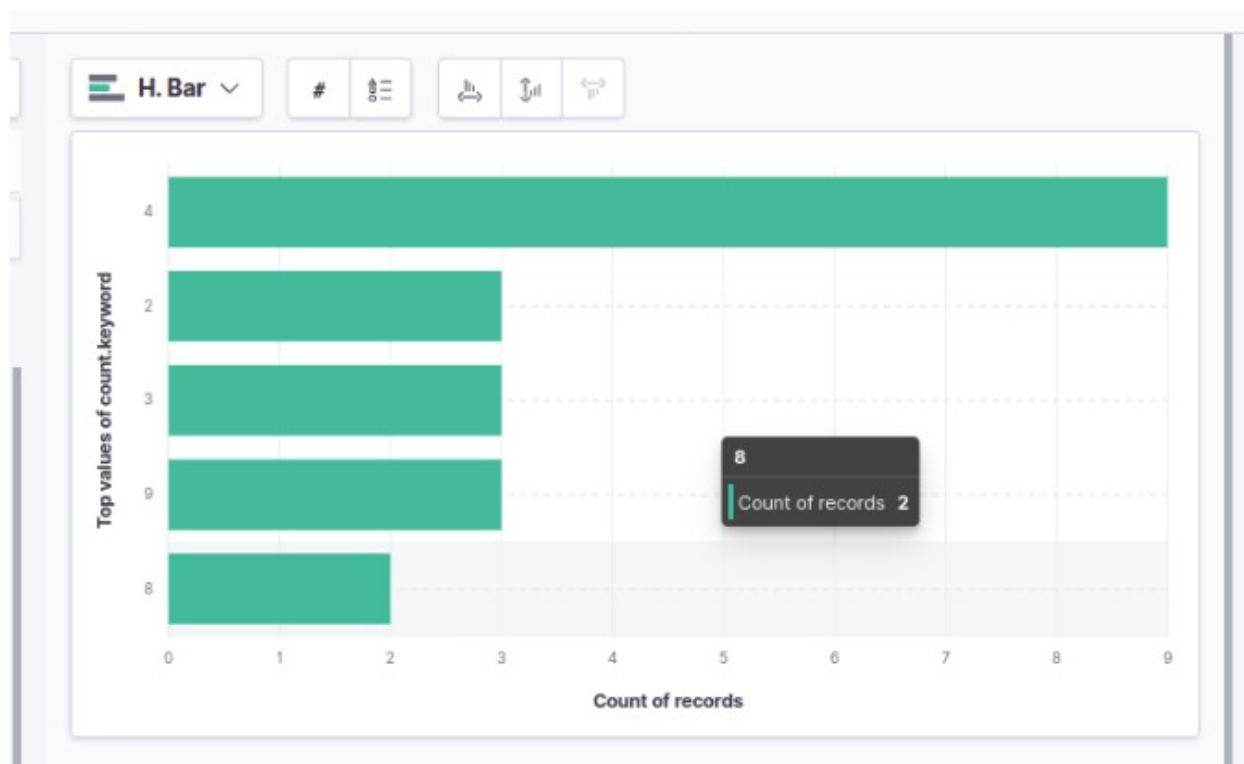
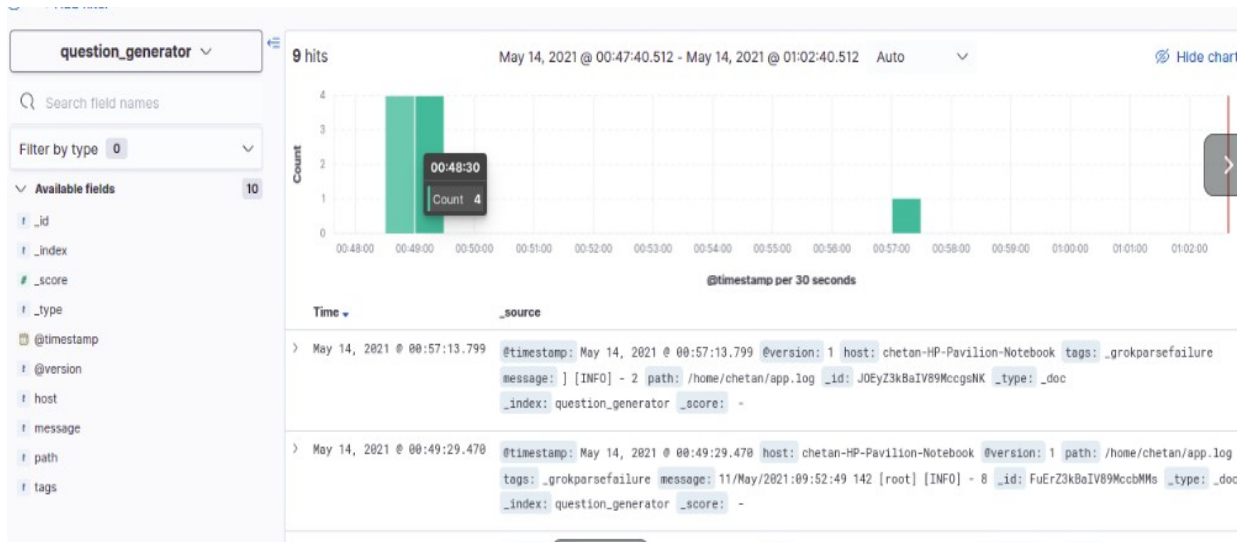
✓ Your index pattern matches 1 source.

question_generator Index

Rows per page: 10 ▾

Here are some visualizations:





6. Future Scope and Conclusion

In this project, we learn a lot regarding NLP and DevOps domain. We learn about model building and its usage in machine learning domain. DevOps tools made our life easy for smooth development and deployment process.

Using neural networks we might get better results in terms of getting more keywords and options as close to answer keywords.

7. References

Following are the references used while developing this project:

- I. <https://spacy.io/>
- II. <https://nlp.stanford.edu/projects/glove/>
- III. <https://www.kaggle.com/stanfordu/stanford-question-answering-dataset>
- IV. <https://git-scm.com/doc>
- V. <https://docs.docker.com/engine>
- VI. <https://iq.opengenus.org/gaussian-naive-bayes/>
- VII. All Lab Documents