Case Study 2- NLP Services on FastAPI, Streamlit and Snowflake

| **Summary** | In this codelab, I have used different technologies to built NLP services like Summarization, Named Entity Recognition and Word Embedding, For backend I have used FastAPI and to store data I have used Snowflake, and Frontend I have used Streamlit and deployed it. |
| --- | --- |
| **URL** |  |
| **Category** | Natural Language Processing |
| **Environment** | FastAPI, Streamlit, Snowflake, Docker, Git, Visual Studio Code |
| **Status** | Not Published |
| **Author** | Tanvi Tembhurne |

[Introduction](#_30j0zll)

[Overview](#_3znysh7)

[SnowFlake](#_2et92p0)

[HuggingFace](#_tyjcwt)

[Tensorflow Hub](#_3dy6vkm)

[FastAPI](#_1t3h5sf)

[IBM](#_4d34og8)

[Streamlit](#_2s8eyo1)

[Docker](#_vtazuq6z9xeq)

[Requirements for FastAPI](#_17dp8vu)

[Requirements for Streamlit](#_3rdcrjn)

[Snowflake Code Snippet](#_26in1rg)

[Transformer-HuggingFace-Summarization-FastAPI Code Snippet](#_lnxbz9)

[Tensorflow-BertModel-Named Entity Recognition- FastAPI Code Snippet](#_35nkun2)

[Tensorflow-Wiki\_words\_250Model-Word Embedding-FastAPI Code Snippet](#_1ksv4uv)

[Connecting FastAPI to Streamlit Code Snippet](#_44sinio)

[Connecting IBM\_API to Streamlit Code Snippet](#_2jxsxqh)

[Dockerize FastAPI](#_z337ya)

[Dockerize Streamlit](#_3j2qqm3)

[Docker-Compose (Merging two container into one)](#_1y810tw)

[FastAPI WebInterface](#_4i7ojhp)

[Streamlit WebInterface](#_2xcytpi)

[Deployment on Web](#_xmvd83tmv3sc)

# Introduction

This case study is dedicated to do Summarization, Named Entity Recognition and Word Embedding for any given data using Transformers and pretrained model from tf.hub and transforming the functions to APIs and calling them from the Frontend that is Streamlit.

# 

# Overview

**Logo

Description automatically generated**

**A picture containing company name

Description automatically generatedLogo

Description automatically generated**

**Logo

Description automatically generated**





**A picture containing text, sign

Description automatically generated Shape

Description automatically generated with low confidence**

**Logo

Description automatically generated**

## Snowflake

Snowflake is **a data warehouse built on top of the Amazon Web Services or Microsoft Azure cloud infrastructure**. The Snowflake architecture allows storage and compute to scale independently, so customers can use and pay for storage and computation separately.

Installation

Follow the official [Link](https://docs.snowflake.com/en/user-guide/python-connector-install.html) for installing python connector



## Hugging Face

The Hugging Face transformers package is **an immensely popular Python library providing pretrained models** that are extraordinarily useful for a variety of natural language processing (NLP) tasks.



## Tensorflow Hub

TensorFlow Hub is **a repository of trained machine learning models**.TensorFlow Hub is a repository of trained machine learning models ready for fine-tuning and deployable anywhere. Reuse trained models like BERT and Faster R-CNN with just a few lines of code.

Graphical user interface, text, application

Description automatically generated

## FastAPI

FastAPI is a Web framework for developing RESTful APIs in Python. FastAPI is based on Pydantic and type hints to validate, serialize, and deserialize data, and automatically auto-generate OpenAPI documents. It fully supports asynchronous programming and can run with Uvicorn and Gunicorn.

## IBM

Used IBM models and used their API to call from Streamlit.

Graphical user interface, text, application

Description automatically generated

Named Entity Recognition

Text

Description automatically generated with medium confidence

## Streamlit

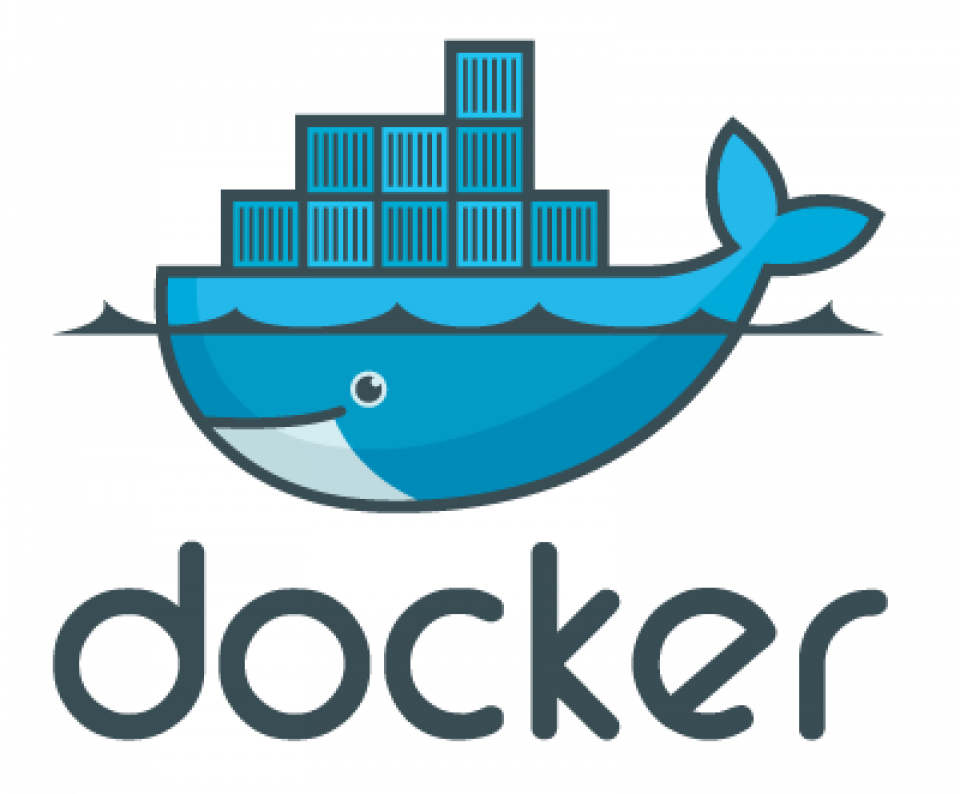
**Streamlit** is an open-source app framework for Machine Learning and Data Science teams. Create beautiful data apps in hours, not weeks. All in pure Python.

Text

Description automatically generated

## Docker

Docker is an open source containerization platform. It **enables developers to package applications into containers**—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.



# Requirements for FastAPI

There is requirements.txt file in FastAPI folder which contains all the libraries required

Graphical user interface, text, application

Description automatically generated

# Requirements for Streamlit

There is requirements.txt file in Streamlit folder which contains all the libraries required

Text

Description automatically generated

# Snowflake Code Snippet

A picture containing graphical user interface

Description automatically generated

Text

Description automatically generated with low confidence

Graphical user interface, application

Description automatically generated

# Transformer-HuggingFace-Summarization-FastAPI Code Snippet

Text

Description automatically generated

# Tensorflow-BertModel-Named Entity Recognition- FastAPI Code Snippet

Text

Description automatically generated

# Tensorflow-Wiki\_words\_250Model-Word Embedding-FastAPI Code Snippet

Text

Description automatically generated

# Connecting FastAPI to Streamlit Code Snippet

Text

Description automatically generated

# Connecting IBM\_API to Streamlit Code Snippet

Text

Description automatically generated

# Dockerize FastAPI

Text

Description automatically generated

# Dockerize Streamlit

Text

Description automatically generated

# Docker-Compose (Merging two container into one)

Text

Description automatically generated

# FastAPI WebInterface

Graphical user interface, text, application, email

Description automatically generated

# Streamlit Web Interface

A screenshot of a computer

Description automatically generated with medium confidence

Command to run the Lab

Text

Description automatically generated

# Deploy using Google cloud Platform

Used CloudRun to deploy the application

FastAPI url: <https://nlpservices-fastapi-v1-th4rn5qo6a-uc.a.run.app>

Streamlit url: <https://nlpservices-streamlit-v1-th4rn5qo6a-uc.a.run.app>