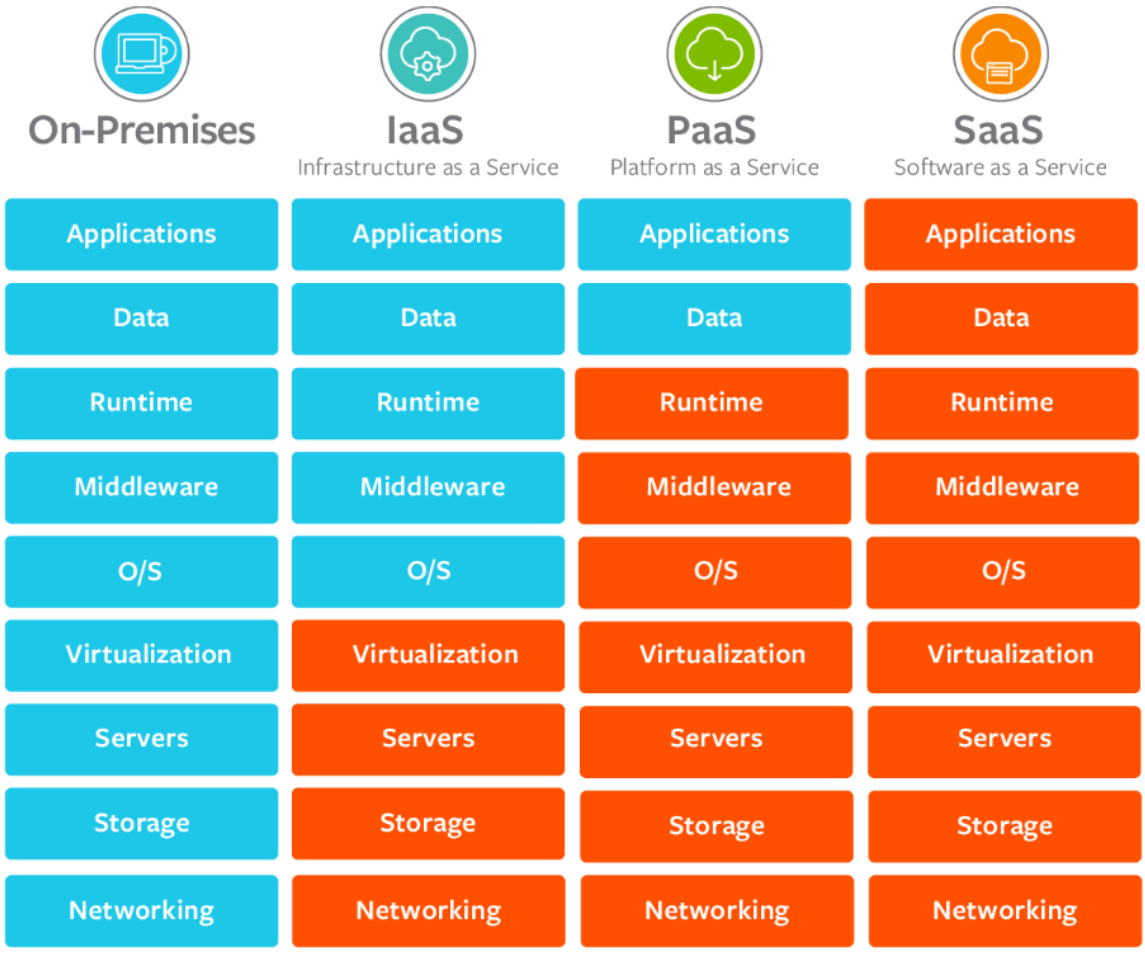
Deployment using Flask and Heroku

1.Understanding local vs servers (Cloud Service)



The above table shows the services provided by each platform. Where the services in the blue blocks need to operate by the user end and the services in the orange block are automated by the online cloud computing providers. The famous cloud service provider under IaaS is AWS, AZURE. Whereas under Paas is Heroku.

2. Flask
We will be referring to a blog present in the medium to build your 1st python web app with flask. Please click here to view the blog.

3. Steps to be followed to create the Python app and deploy it using Flask and Heroku.
    1. Train your model
    2. Create a web app using flask.
    3. Commit the code in Github
    4. Create an account in Heroku (PaaS)
    5. Link your Github with Heroku
    6. Deploy the model
    7. Web App is ready to use.

4. Let's start with creating a Python app and deploy it using Flask and Heroku.
    1. Choose your data.
    2. Do the required preprocessing
    3. Create a model
    4. Validate your model
    5. Create a Pickle file.

Let's understand what is a pickle file:

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it "serializes" the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

6. Creating an App.py file on the flask.
   Note: While creating the above App.py you will also require HTML index.
7. Create HTML index file.
8. Once App.py file is ready you can run it on your local platform.
   Your app is successfully create using flask and running on On-premises

Let's Deploy the Flask work on Heroku (PaaS)

Note: To Deploy your flask work on Heroku you will require a few confirmation files.

9. Create a procfile

Let's understand what is Gunicorn

Green Unicorn (Gunicorn)

Green Unicorn, commonly shortened to "Gunicorn", is a Web Server Gateway Interface (WSGI) server implementation that is commonly used to run Python web applications.

Why is Gunicorn important?

Gunicorn is one of many WSGI server implementations, but it's particularly important because it is a stable, commonly-used part of web app deployments that are powered some of the largest Python-powered web applications in the world, such as Instagram.

Gunicorn implements the PEP3333 WSGI server standard specification so that it can run Python web applications that implement the application interface. For example, if you write a web application with a web framework such as Django, Flask or Bottle, then your application implements the WSGI specification.

10. Create requirement file containing information of all the required packages.
11. Create a Heroku Account
12. Go to Github Create a repository, push all your file here.
13. Connect Heroku to your Github
14. Search for Github repository and Deploy it.
15. Wait for all installations
    Your model is been successfully deployed and the link is ready.

Github - https://github.com/shubhamkotal/Deployment

Thank you