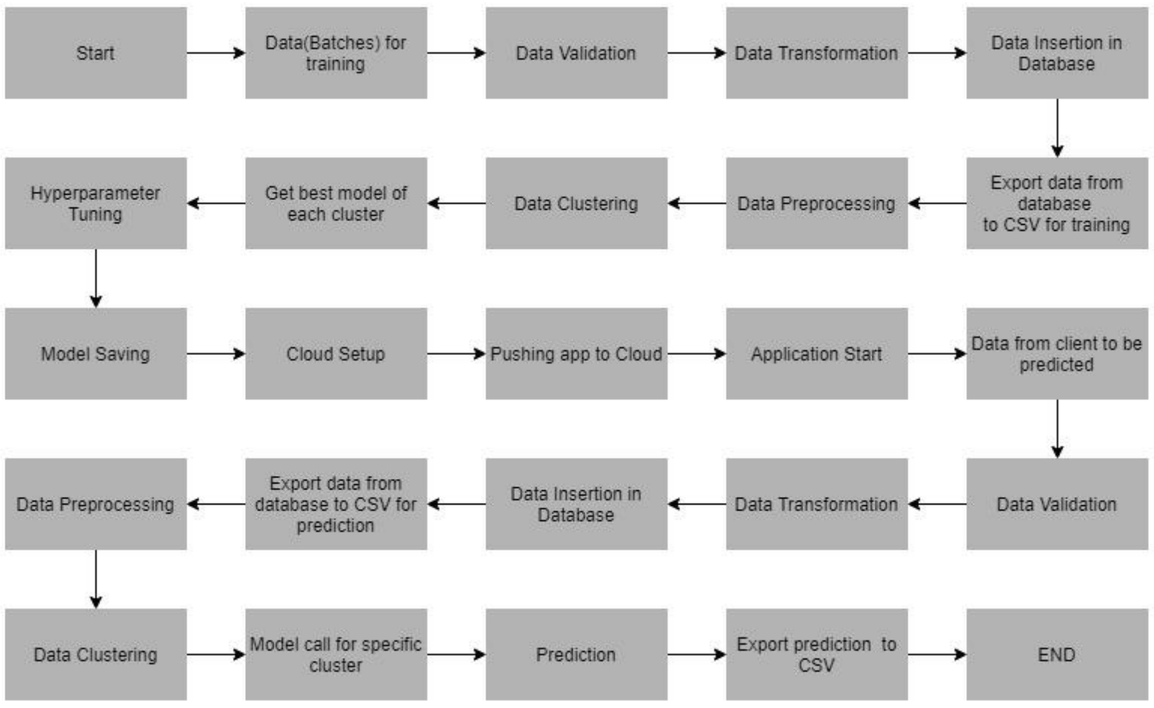
https://www.youtube.com/yogeshmurumkar

Problem Statement

To build a classification methodology to determine whether a person makes over 50K per year.

Architecture


Start
Data(Batches) for training
Data Validation
Data Transformation
Data Insertion in Database
Export data from database to CSV for training
Data Preprocessing
Data Clustering
Get best model of each cluster
Hyperparameter Tuning
Model Saving
Cloud Setup
Pushing app to Cloud
Application Start
Data from client to be predicted
Data Validation
Data Transformation
Data Insertion in Database
Export data from database to CSV for prediction
Data Preprocessing
Data Clustering
Model call for specific cluster
Prediction
Export prediction to CSV
END

Data Description

The client will send data in multiple sets of files in batches at a given location. The data has been extracted from the census bureau.

The data contains 32561 instances with the following attributes:
Features:

1. **age:** continuous. It denotes the age of the person.
2. **workclass:** It denotes the working class of the person. Sample values: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
3. **fnlwgt:** continuous.
4. **education:** It denotes the educational qualification of the person. Sample values: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm,

Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

5. **education**-num: continuous. It denotes the quantitative values with reference to education.
6. **marital-status**: It denotes the marital status of the person. Sample values: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
7. **occupation**: It denotes the occupation of a person. Sample values: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
8. **relationship**: It denotes the people present in the family. Sample values: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
9. **race**: It denotes the person's origins. Sample values: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
10. **sex**: It denotes the person's gender. Sample values: Female, Male.
11. **capital-gain**: continuous. It denotes the monitory gains by the person.
12. **capital-loss**: continuous. It denotes the monitory loss by the person.
13. **hours-per-week**: continuous. It denotes the number of working hours per week by the person.
14. **native-country**: It denotes the country to which the person belongs. Sample values: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Target Label:

Whether a person earns more or less than 50000 dollars per year.

`15. Income: >50K, <=50K.`

Apart from training files, we also require a "schema" file from the client, which contains all the relevant information about the training files such as:

Name of the files, Length of Date value in FileName, Length of Time value in FileName, Number of Columns, Name of the Columns, and their datatype.

Data Validation

In this step, we perform different sets of validation on the given set of training files.

1. Name Validation- We validate the name of the files based on the given name in the schema file. We have created a regex pattern as per the name given in the schema file to use for validation. After validating the pattern in the name, we check for the length of date in the file name as well as the length of time in the file name. If all the values are as per requirement, we move such files to "Good_Data_Folder" else we move such files to "Bad_Data_Folder."

2. Number of Columns - We validate the number of columns present in the files, and if it doesn't match with the value given in the schema file, then the file is moved to "Bad_Data_Folder."

3. Name of Columns - The name of the columns is validated and should be the same as given in the schema file. If not, then the file is moved to "Bad_Data_Folder".

4. The datatype of columns - The datatype of columns is given in the schema file. It is validated when we insert the files into Database. If the datatype is wrong, then the file is moved to "Bad_Data_Folder".

5. Null values in columns - If any of the columns in a file have all the values as NULL or missing, we discard such a file and move it to "Bad_Data_Folder".

Data Insertion in Database

1) Database Creation and connection - Create a database with the given name passed. If the database has already been created, open a connection to the database.

2) Table creation in the database - Table with name - "Good_Data", is created in the database for inserting the files in the "Good_Data_Folder" based on given column names and datatype in the schema file. If the table is already present, then the new table is not created, and new files are inserted in the already present table as we want training to be done on new as well as old training files.
3) Insertion of files in the table - All the files in the "Good_Data_Folder" are inserted in the above-created table. If any file has invalid data type in any of the columns, the file is not loaded in the table and is moved to "Bad_Data_Folder".

Model Training

1) **Data Export from Db** - The data in a stored database is exported as a CSV file to be used for model training.

2) **Data Preprocessing**

a) Drop the columns not required for prediction.
b) Remove the unwanted spaces in data.
c) For this dataset, the null values were replaced with '?' in the client data. Those '?' have been replaced with NaN values.
d) Check for null values in the columns. If present, impute the null values using the categorical imputer.
e) Replace and encode the categorical values with numeric values.
f) Scale the numeric values using the standard scaler.
g) Handle the imbalanced dataset using oversampling.

3) **Clustering -** KMeans algorithm is used to create clusters in the preprocessed data. The optimum number of clusters is selected by plotting the elbow plot, and for the dynamic selection of the number of clusters, we are using "KneeLocator" function. The idea behind clustering is to implement different algorithms

The Kmeans model is trained over preprocessed data, and the model is saved for further use in prediction.

4) **Model Selection** – After the clusters have been created, we find the best model for each cluster. We are using two algorithms, "Random Forest" and "XGBoost". For each cluster, both the algorithms are passed with the best parameters derived from GridSearch. We calculate the AUC scores for both models and select the model with the best score. Similarly, the model is selected for each cluster. All the models for every cluster are saved for use in prediction.

Prediction Data Description

The Client will send the data in multiple sets of files in batches at a given location. Data will contain the annual income of various persons.

Apart from prediction files, we also require a "schema" file from the client, which contains all the relevant information about the training files such as:

Name of the files, Length of Date value in FileName, Length of Time value in FileName, Number of Columns, Name of the Columns and their datatype.

Data Validation

In this step, we perform different sets of validation on the given set of training files.

1) Name Validation- We validate the name of the files based on given Name in the schema file. We have created a regex pattern as per the name given in the schema file, to use for validation. After validating the pattern in the name, we check for the length of date in the file name as well as the length of the timestamp in the file name. If all the values are as per requirement, we move such files to "Good_Data_Folder" else we move such files to "Bad_Data_Folder".

2) Number of Columns - We validate the number of columns present in the files, and if it doesn't match with the value given in the schema file, then the file is moved to "Bad_Data_Folder".

3) Name of Columns - The name of the columns is validated and should be same as given in the schema file. If not, then the file is moved to "Bad_Data_Folder".

4) Datatype of columns - The datatype of columns is given in the schema file. This is validated when we insert the files into Database. If the datatype is incorrect, then the file is moved to "Bad_Data_Folder".

5) Null values in columns - If any of the columns in a file has all the values as NULL or missing, we discard such file and move it to "Bad_Data_Folder".

Data Insertion in Database

1) Database Creation and connection - Create a database with the given name passed. If the database is already created, open the connection to the database.

2) Table creation in the database - Table with name - "Good_Data", is created in the database for inserting the files in the "Good_Data_Folder" based on given column names and datatype in the schema file. If the table is already present, then a new table is not created, and new files are inserted into the already present table as we want training to be done on new as well old training files.

3) Insertion of files in the table - All the files in the "Good_Data_Folder" are inserted in the above-created table. If any file has invalid data type in any of the columns, the file is not loaded in the table and is moved to "Bad_Data_Folder".

Prediction

1) Data Export from Db - The data in the stored database is exported as a CSV file to be used for prediction.

2) Data Preprocessing :

a) Drop the columns not required for prediction.
b) Remove the unwanted spaces in data.
c) For this dataset, the null values were replaced with '?' in the client data. Those '?' have been replaced with NaN values.
d) Check for null values in the columns. If present, impute the null values using the categorical imputer.
e) Replace and encode the categorical values with numeric values.
f) Scale the numeric values using the standard scaler.
g) Handle the imbalanced dataset using oversampling

3) Clustering - KMeans model created during training is loaded, and clusters for the preprocessed prediction data is predicted.

4) Prediction - Based on the cluster number, the respective model is loaded and is used to predict the data for that cluster.

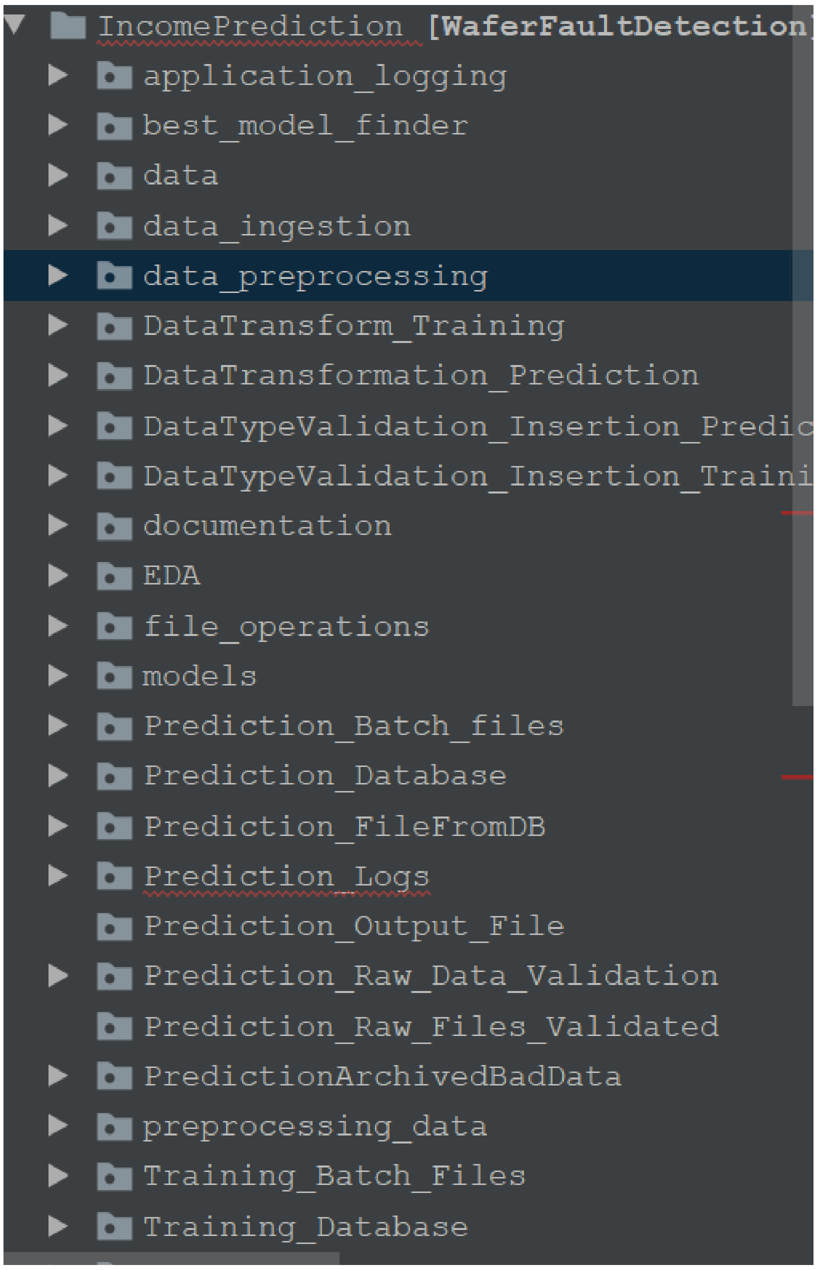
5) Once the prediction is made for all the clusters, the predictions along with the Wafer names are saved in a CSV file at a given location, and the location is returned to the client.

Deployment

We will be deploying the model to the Pivotal Cloud Foundry platform.

This is a workflow diagram for the prediction of using the trained model.

Now let’s look at the project folder structure:

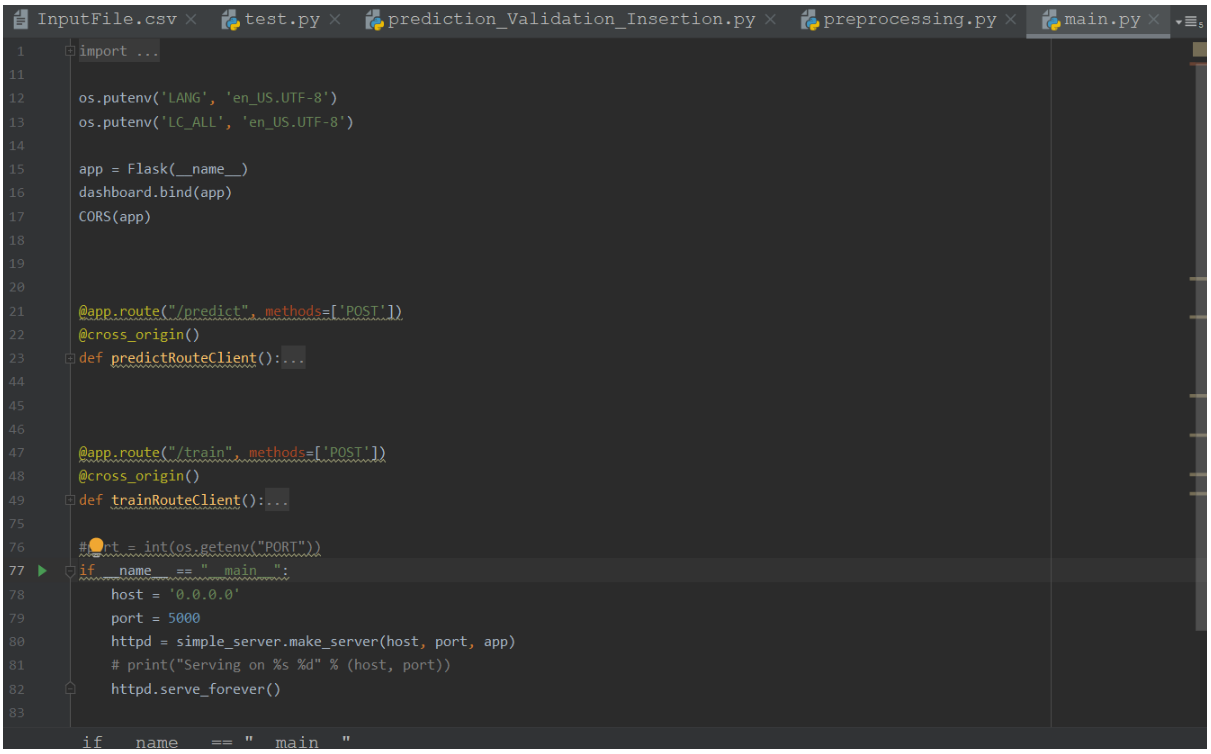


```
▼ IncomePrediction [WaferFaultDetection]
    ▶ application_logging
    ▶ best_model_finder
    ▶ data
    ▶ data_ingestion
    ▶ data_preprocessing
    ▶ DataTransform_Training
    ▶ DataTransformation_Prediction
    ▶ DataTypeValidation_Insertion_Predic
    ▶ DataTypeValidation_Insertion_Traini
    ▶ documentation
    ▶ EDA
    ▶ file_operations
    ▶ models
    ▶ Prediction_Batch_files
    ▶ Prediction_Database
    ▶ Prediction_FileFromDB
    ▶ Prediction_Logs
      Prediction_Output_File
    ▶ Prediction_Raw_Data_Validation
      Prediction_Raw_Files_Validated
    ▶ PredictionArchivedBadData
    ▶ preprocessing_data
    ▶ Training_Batch_Files
    ▶ Training_Database
```

requirements.txt file consists of all the packages that you need to deploy the app in the cloud.

https://www.youtube.com/yogeshmurumkar

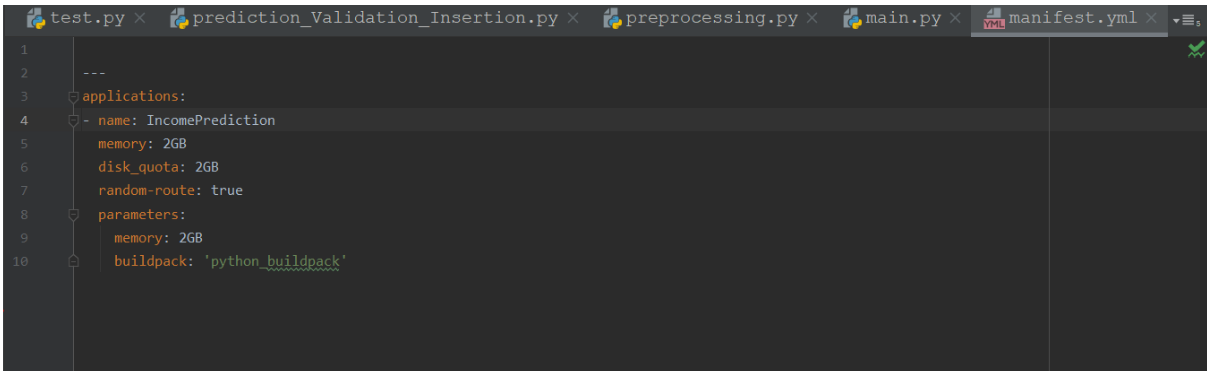
main.py is the entry point of our application, where the flask server starts. Here we will have two routes, one for training and the other for prediction.



```python
import ...

os.putenv('LANG', 'en_US.UTF-8')
os.putenv('LC_ALL', 'en_US.UTF-8')

app = Flask(__name__)
dashboard.bind(app)
CORS(app)



@app.route("/predict", methods=['POST'])
@cross_origin()
def predictRouteClient():...



@app.route("/train", methods=['POST'])
@cross_origin()
def trainRouteClient():...

#port = int(os.getenv("PORT"))
if __name__ == "__main__":
    host = '0.0.0.0'
    port = 5000
    httpd = simple_server.make_server(host, port, app)
    # print("Serving on %s %d" % (host, port))
    httpd.serve_forever()
```

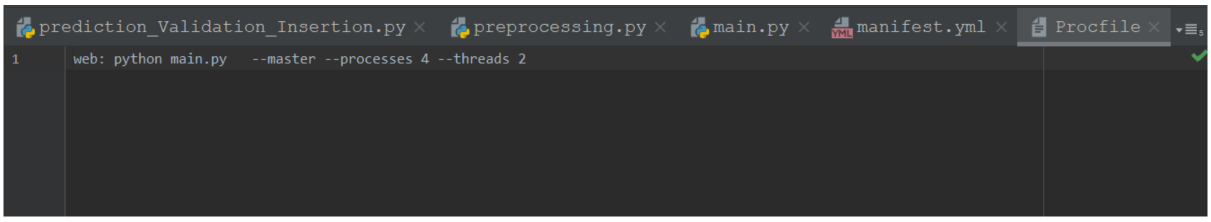
if __name__ == "__main__"

manifest.yml:- This file contains the instance configuration, app name, and build pack language.



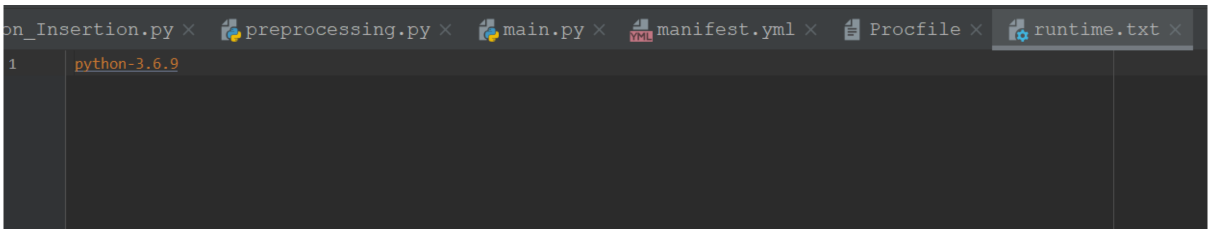
```yaml

---
applications:
- name: IncomePrediction
  memory: 2GB
  disk_quota: 2GB
  random-route: true
  parameters:
    memory: 2GB
    buildpack: 'python_buildpack'
```

Procfile:- It contains the entry point of the app.



```
web: python main.py   --master --processes 4 --threads 2
```

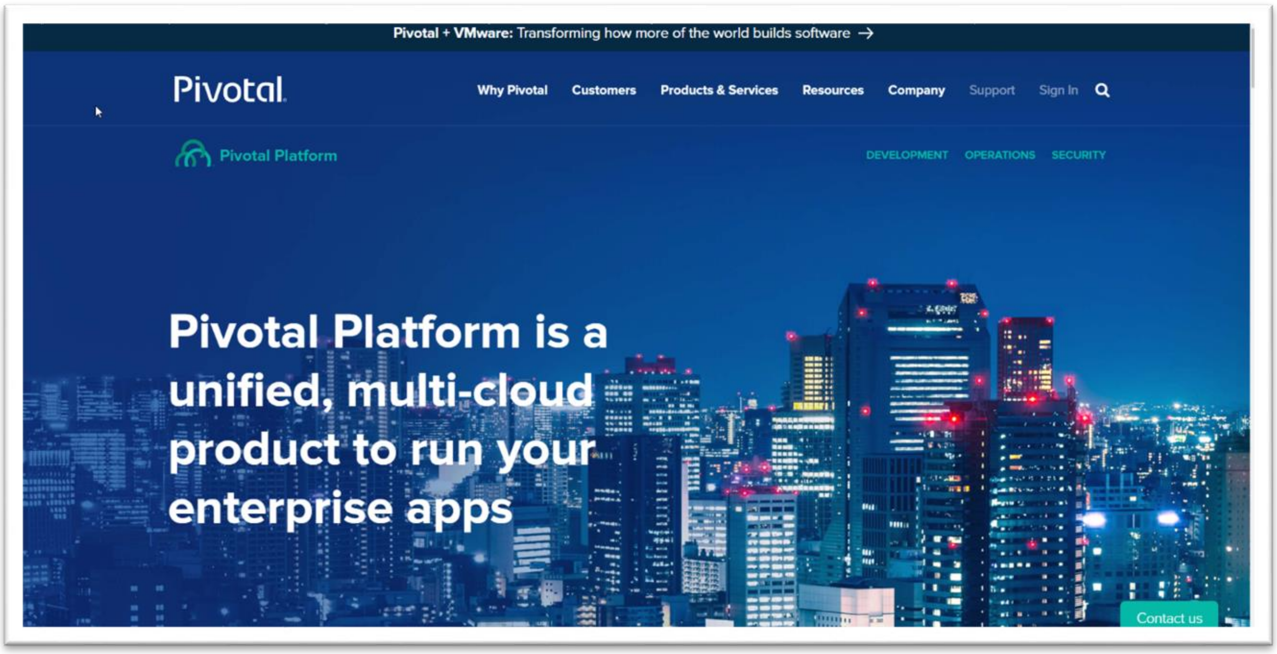
runtime.txt:- It contains the Python version number.



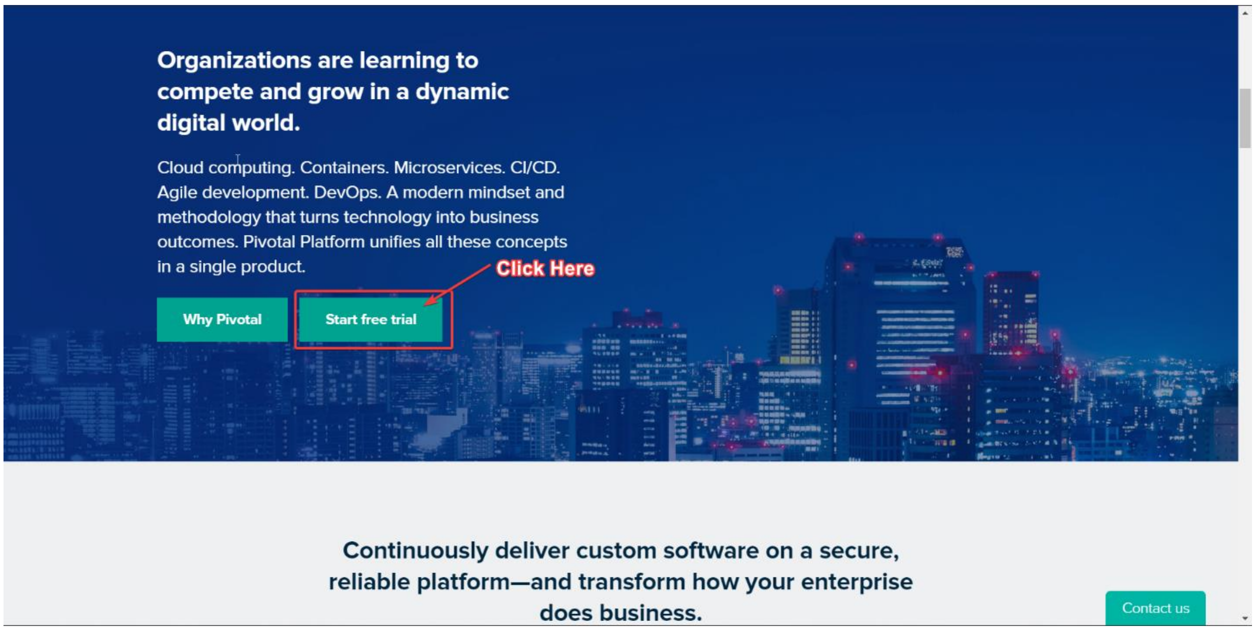
on_Insertion.py × preprocessing.py × main.py × manifest.yml × Procfile × runtime.txt ×

```
python-3.6.9
```

To Cloud:


Pivotal + VMware: Transforming how more of the world builds software →
Pivotal
Why Pivotal
Customers
Products & Services
Resources
Company
Support
Sign In
Pivotal Platform
DEVELOPMENT
OPERATIONS
SECURITY
Pivotal Platform is a unified, multi-cloud product to run your enterprise apps
Contact us

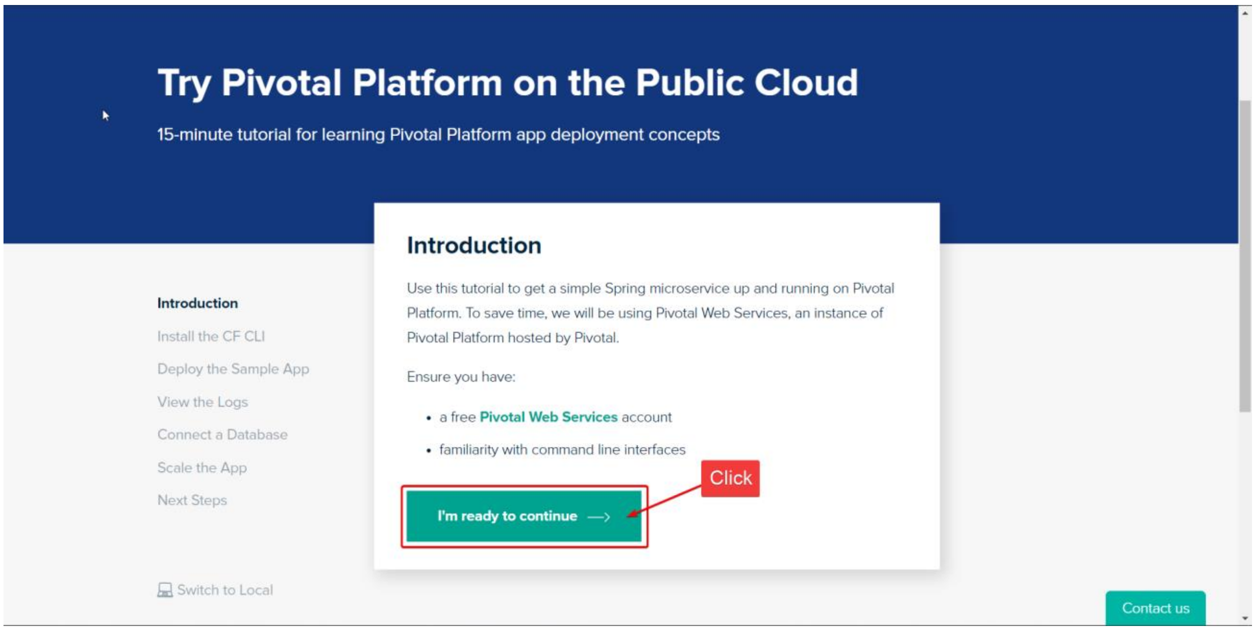
Visit the official website **https://pivotal.io/platform.**


Organizations are learning to compete and grow in a dynamic digital world.

Cloud computing. Containers. Microservices. CI/CD. Agile development. DevOps. A modern mindset and methodology that turns technology into business outcomes. Pivotal Platform unifies all these concepts in a single product.

Continuously deliver custom software on a secure, reliable platform—and transform how your enterprise does business.

Scroll Down to see the **Start Trial Button**


Try Pivotal Platform on the Public Cloud

15-minute tutorial for learning Pivotal Platform app deployment concepts

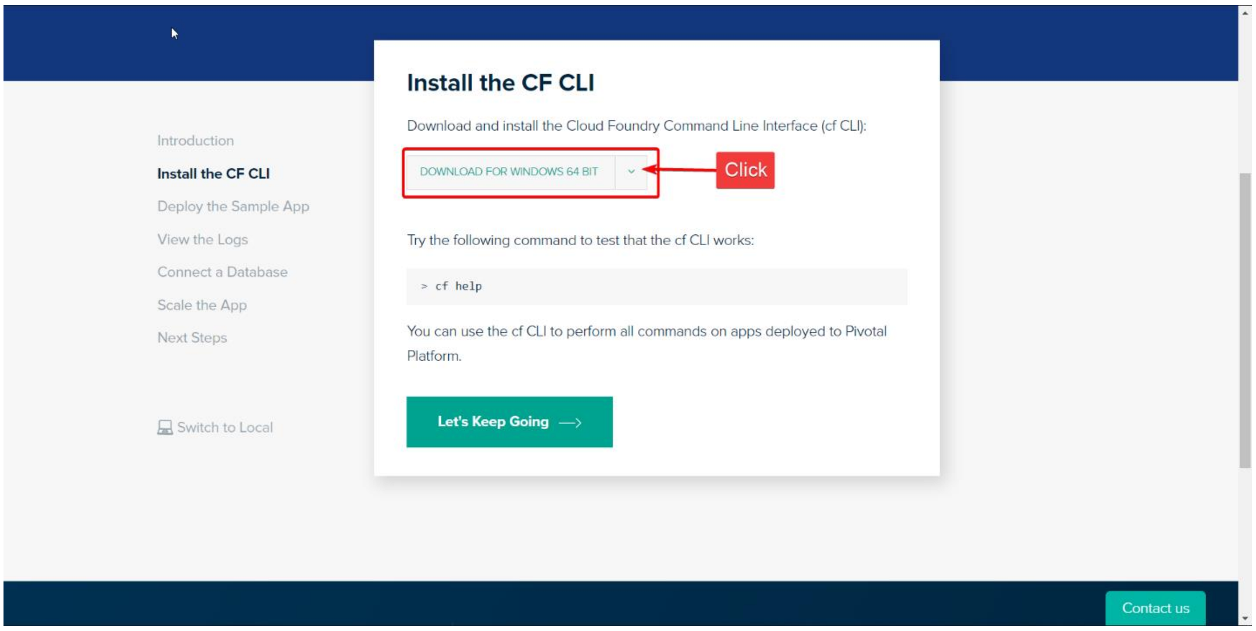
Introduction

Use this tutorial to get a simple Spring microservice up and running on Pivotal Platform. To save time, we will be using Pivotal Web Services, an instance of Pivotal Platform hosted by Pivotal.

Ensure you have:

- a free Pivotal Web Services account
- familiarity with command line interfaces

Click on the start trial button, and the next interface will open. Then I will click on **I'm ready to continue**



Install the CF CLI

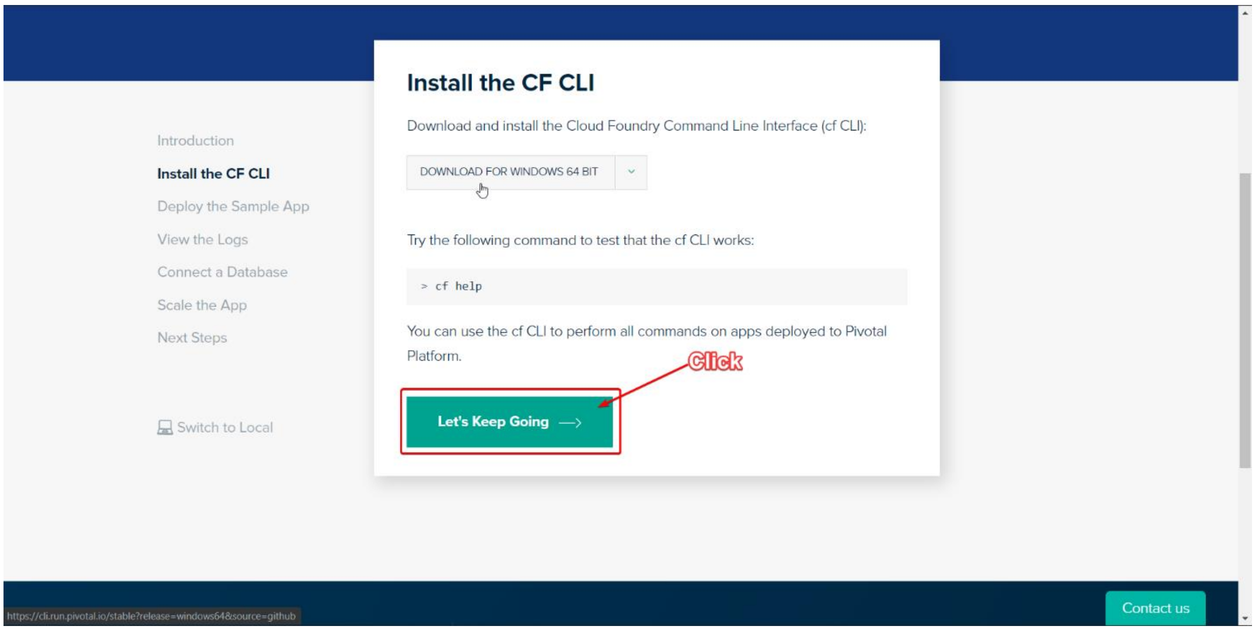
Download and install the Cloud Foundry Command Line Interface (cf CLI):

Try the following command to test that the cf CLI works:

```
> cf help
```

You can use the cf CLI to perform all commands on apps deployed to Pivotal Platform.

Click on Download for **Windows 64 bit, and** then zip file will be downloaded. Keep it for future uses.



Install the CF CLI

Download and install the Cloud Foundry Command Line Interface (cf CLI):

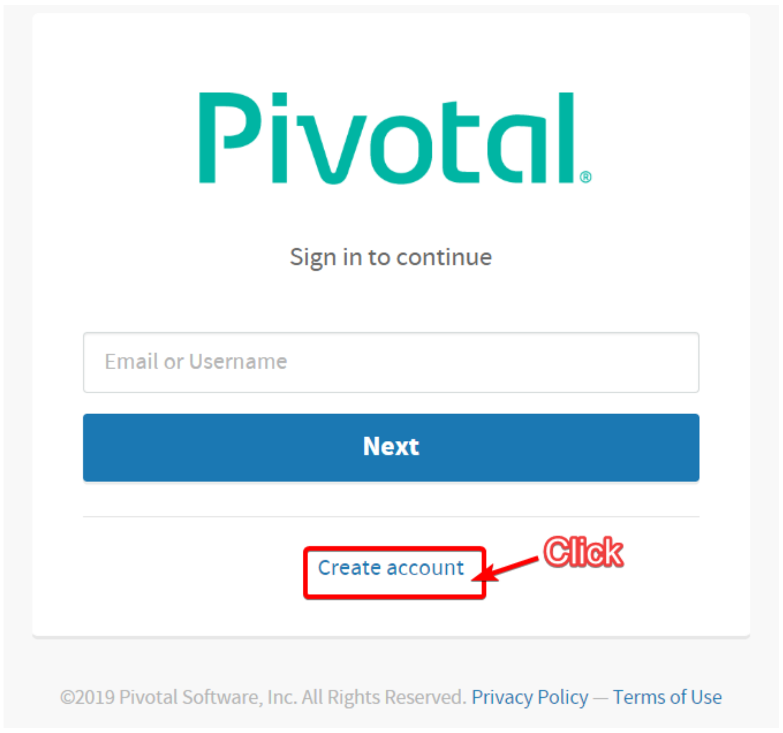
Try the following command to test that the cf CLI works:

```
> cf help
```

You can use the cf CLI to perform all commands on apps deployed to Pivotal Platform.

Now click on **Let's Keep Going**

https://www.youtube.com/yogeshmurumkar


Pivotal

Sign in to continue

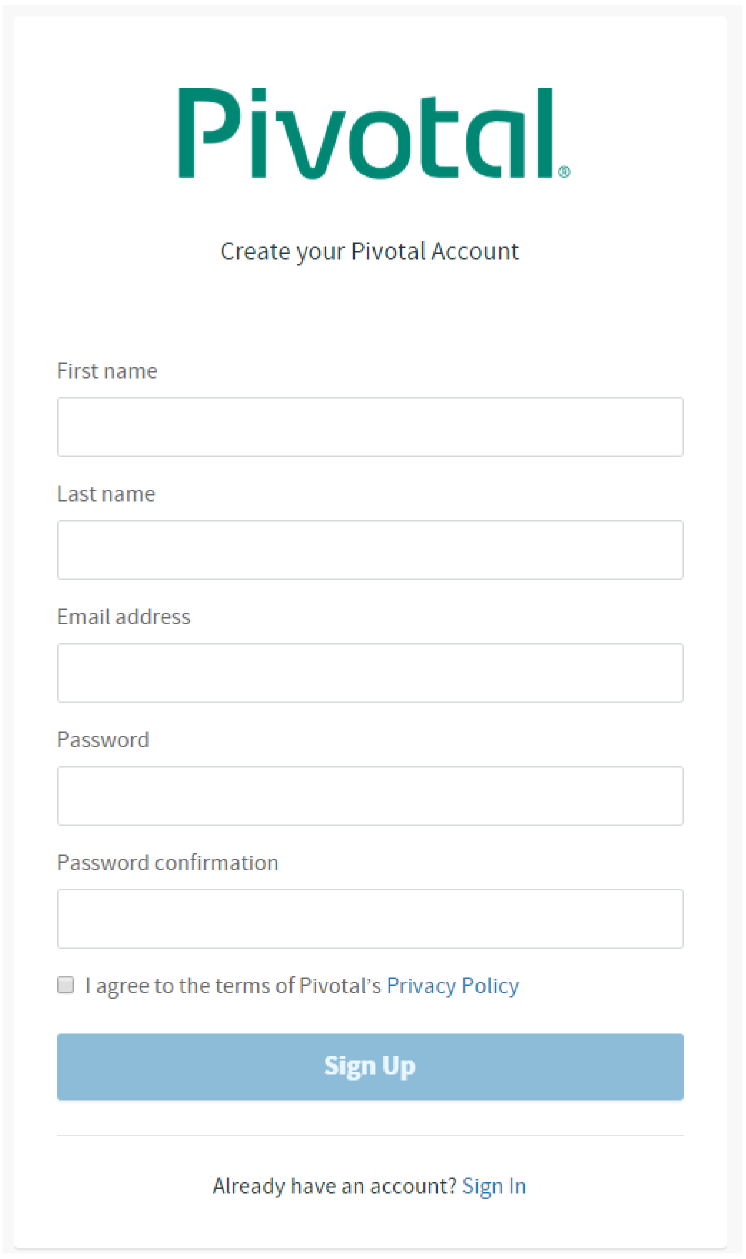
Email or Username

Next

Create account

©2019 Pivotal Software, Inc. All Rights Reserved. Privacy Policy — Terms of Use

Click on **Create Your Account**


Create your Pivotal Account

First name

Last name

Email address

Password

Password confirmation

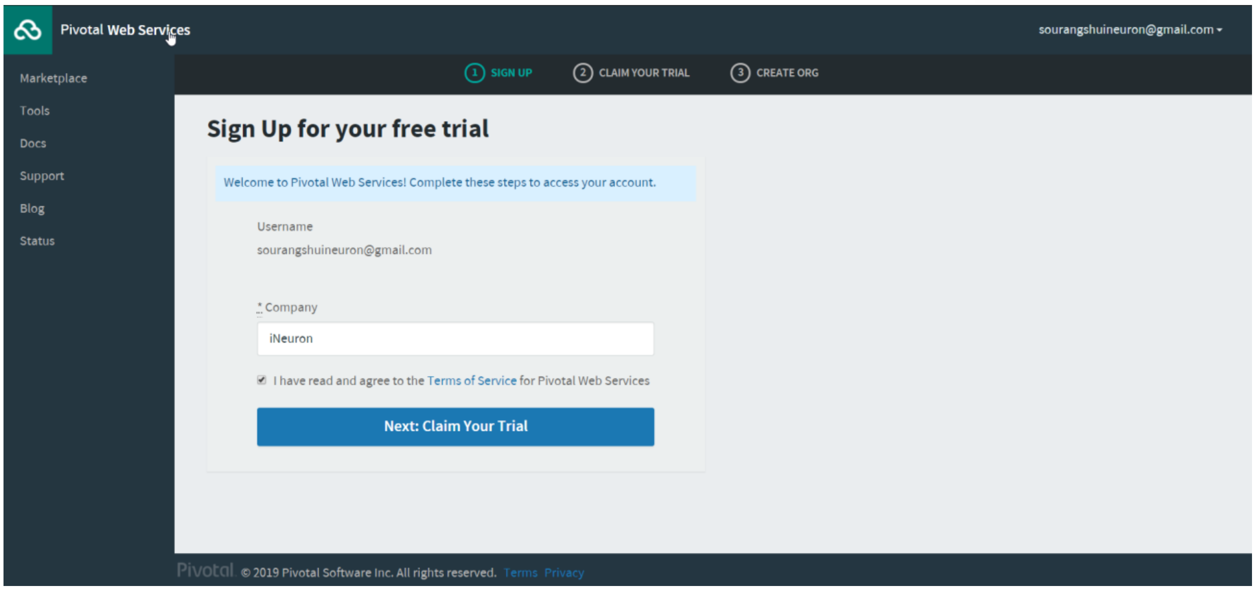
☐ I agree to the terms of Pivotal's Privacy Policy

Already have an account? Sign In

Fill Up your Details For registration

Do the email verification

Then login in again



Sign Up for your free trial

Welcome to Pivotal Web Services! Complete these steps to access your account.

Username
sourangshuineuron@gmail.com

* Company
iNeuron

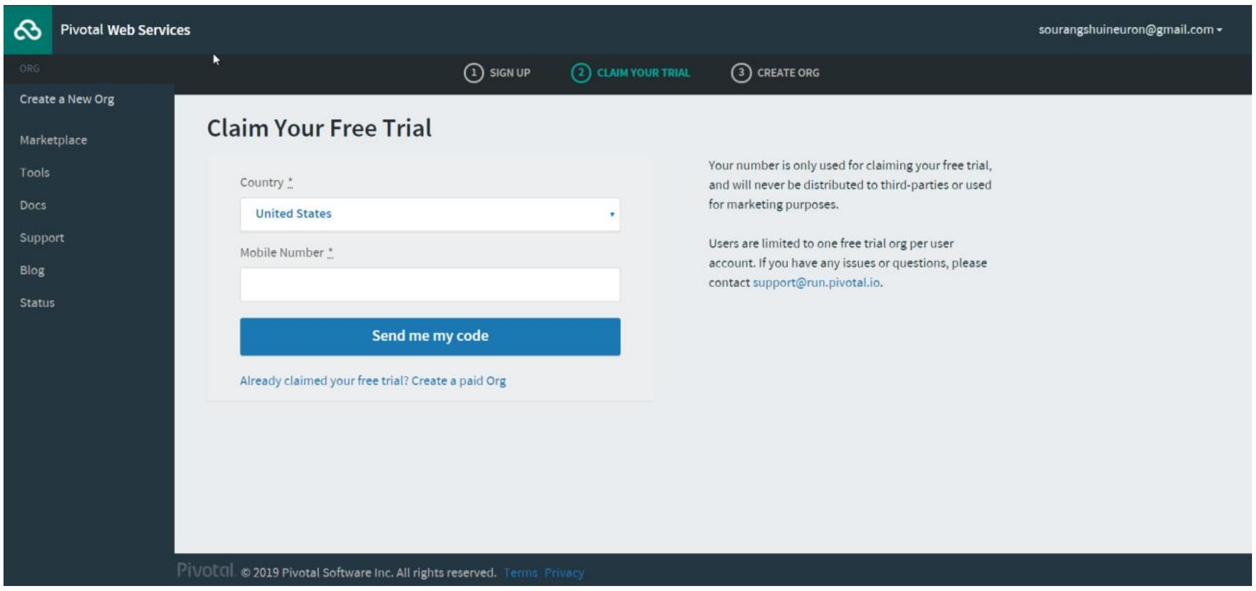
☑ I have read and agree to the Terms of Service for Pivotal Web Services

Next: Claim Your Trial

Pivotal © 2019 Pivotal Software Inc. All rights reserved. Terms Privacy

After logging in, you will see this screen below and start your free trial.

Write any Company or whichever one you prefer.



Claim Your Free Trial

Country *
United States

Mobile Number *

Send me my code

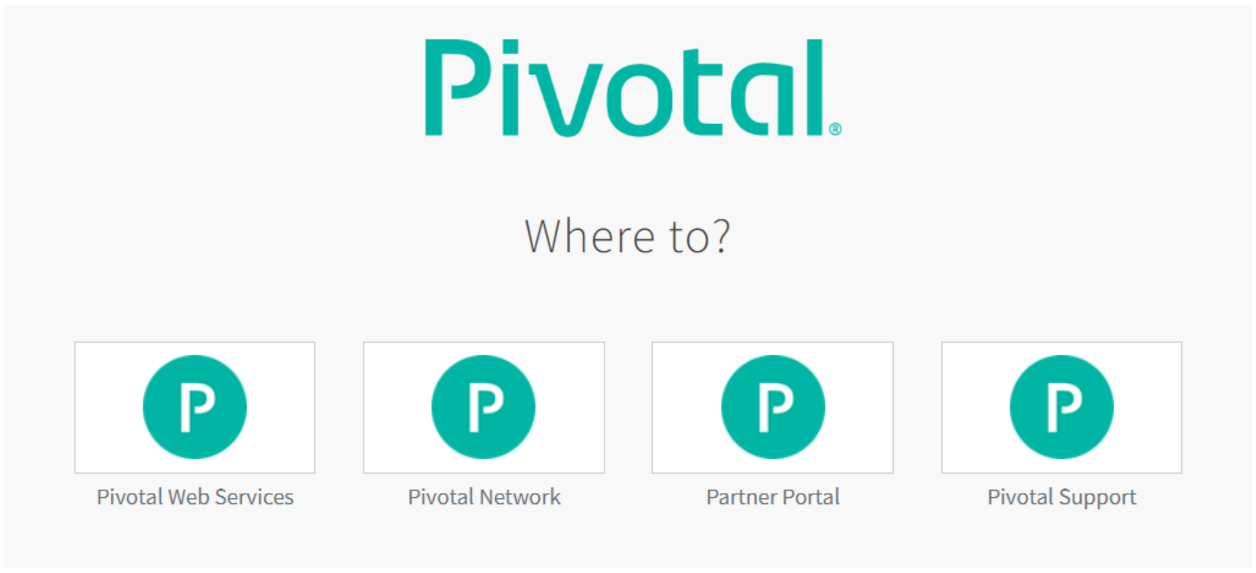
Already claimed your free trial? Create a paid Org

Your number is only used for claiming your free trial, and will never be distributed to third-parties or used for marketing purposes.

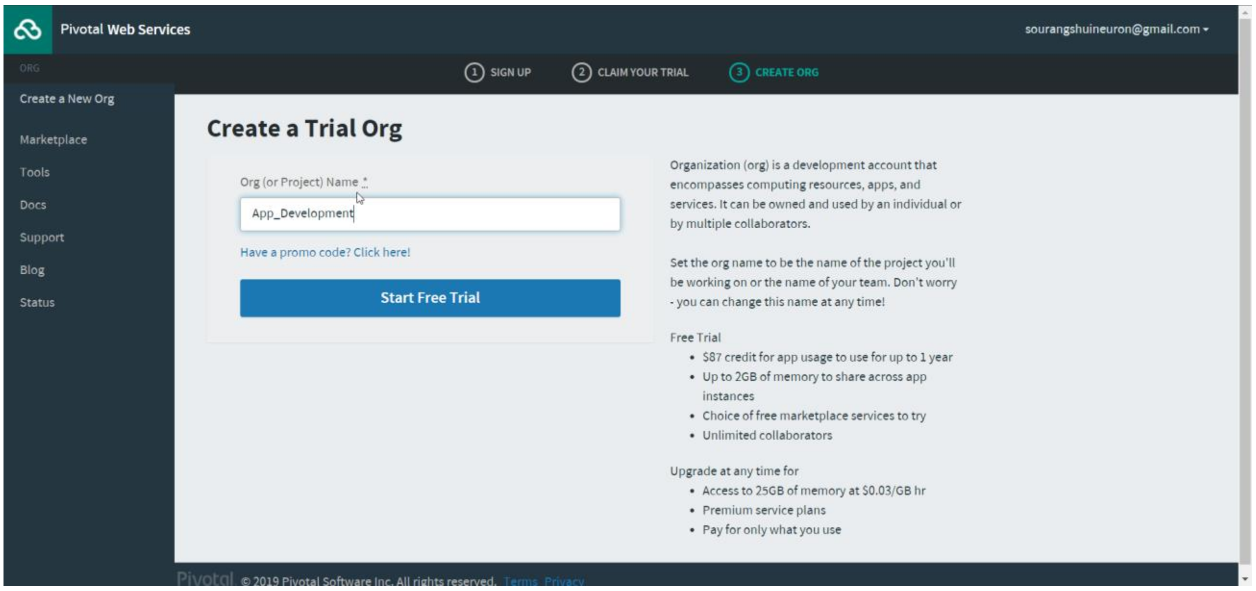
Users are limited to one free trial org per user account. If you have any issues or questions, please contact support@run.pivotal.io.

Pivotal © 2019 Pivotal Software Inc. All rights reserved. Terms Privacy

Enter your **Mobile Number** for Verification



Click on **Pivotal Web Services**



Create a Trial Org

Org (or Project) Name *

App_Development

Have a promo code? Click here!

Organization (org) is a development account that encompasses computing resources, apps, and services. It can be owned and used by an individual or by multiple collaborators.

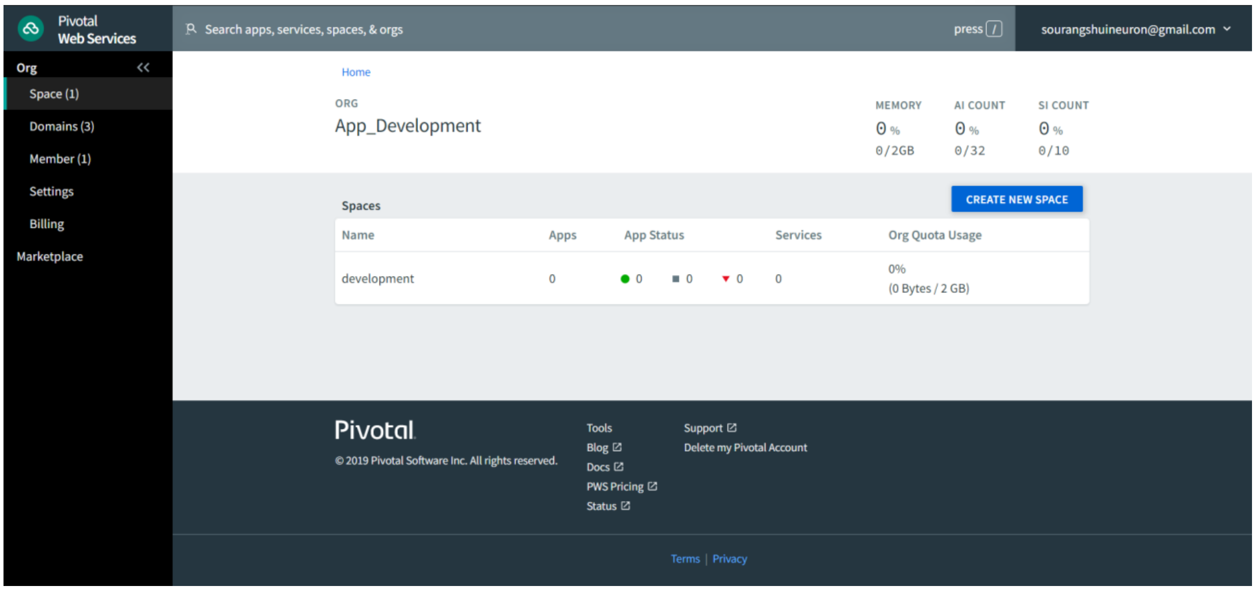
Set the org name to be the name of the project you'll be working on or the name of your team. Don't worry - you can change this name at any time!

Free Trial
- $87 credit for app usage to use for up to 1 year
- Up to 2GB of memory to share across app instances
- Choice of free marketplace services to try
- Unlimited collaborators

Upgrade at any time for
- Access to 25GB of memory at $0.03/GB hr
- Premium service plans
- Pay for only what you use

Pivotal © 2019 Pivotal Software Inc. All rights reserved. Terms Privacy

Give any **Org** name



Name	Apps	App Status			Services	Org Quota Usage
development	0	● 0	■ 0	▼ 0	0	0% (0 Bytes / 2 GB)

Pivotal
© 2019 Pivotal Software Inc. All rights reserved.
Tools
Blog
Docs
PWS Pricing
Status
Support
Delete my Pivotal Account
Terms | Privacy

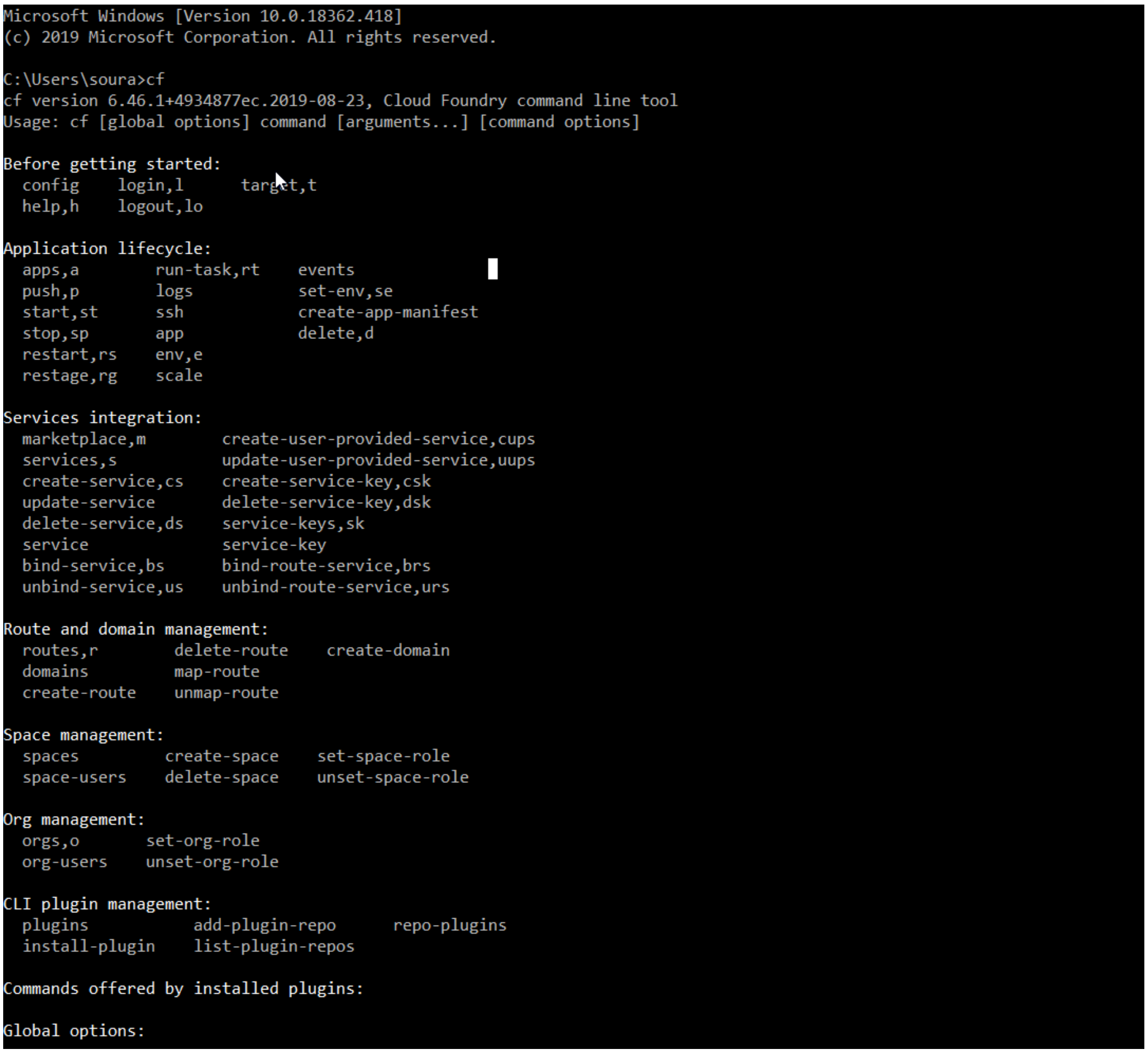
Now you are inside your Org, and by default, development space is created in your org. You can push your apps here.

The cloud signup process is done, and the setup is ready for us to push the app.

Previously you have downloaded the **CLI.zip** file. Unzip the file and install the .exe file with admin rights.

After a successful installation, you can verify by opening your CMD and type **cf**.

Then you will get a screen which is shown below



```
Microsoft Windows [Version 10.0.18362.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\soura>cf
cf version 6.46.1+4934877ec.2019-08-23, Cloud Foundry command line tool
Usage: cf [global options] command [arguments...] [command options]

Before getting started:
  config     login,l      target,t
  help,h     logout,lo

Application lifecycle:
  apps,a        run-task,rt    events
  push,p        logs           set-env,se
  start,st      ssh            create-app-manifest
  stop,sp       app            delete,d
  restart,rs    env,e
  restage,rg    scale

Services integration:
  marketplace,m        create-user-provided-service,cups
  services,s           update-user-provided-service,uups
  create-service,cs    create-service-key,csk
  update-service       delete-service-key,dsk
  delete-service,ds    service-keys,sk
  service              service-key
  bind-service,bs      bind-route-service,brs
  unbind-service,us    unbind-route-service,urs

Route and domain management:
  routes,r        delete-route    create-domain
  domains         map-route
  create-route    unmap-route

Space management:
  spaces         create-space    set-space-role
  space-users    delete-space    unset-space-role

Org management:
  orgs,o       set-org-role
  org-users    unset-org-role

CLI plugin management:
  plugins           add-plugin-repo      repo-plugins
  install-plugin    list-plugin-repos

Commands offered by installed plugins:

Global options:
```

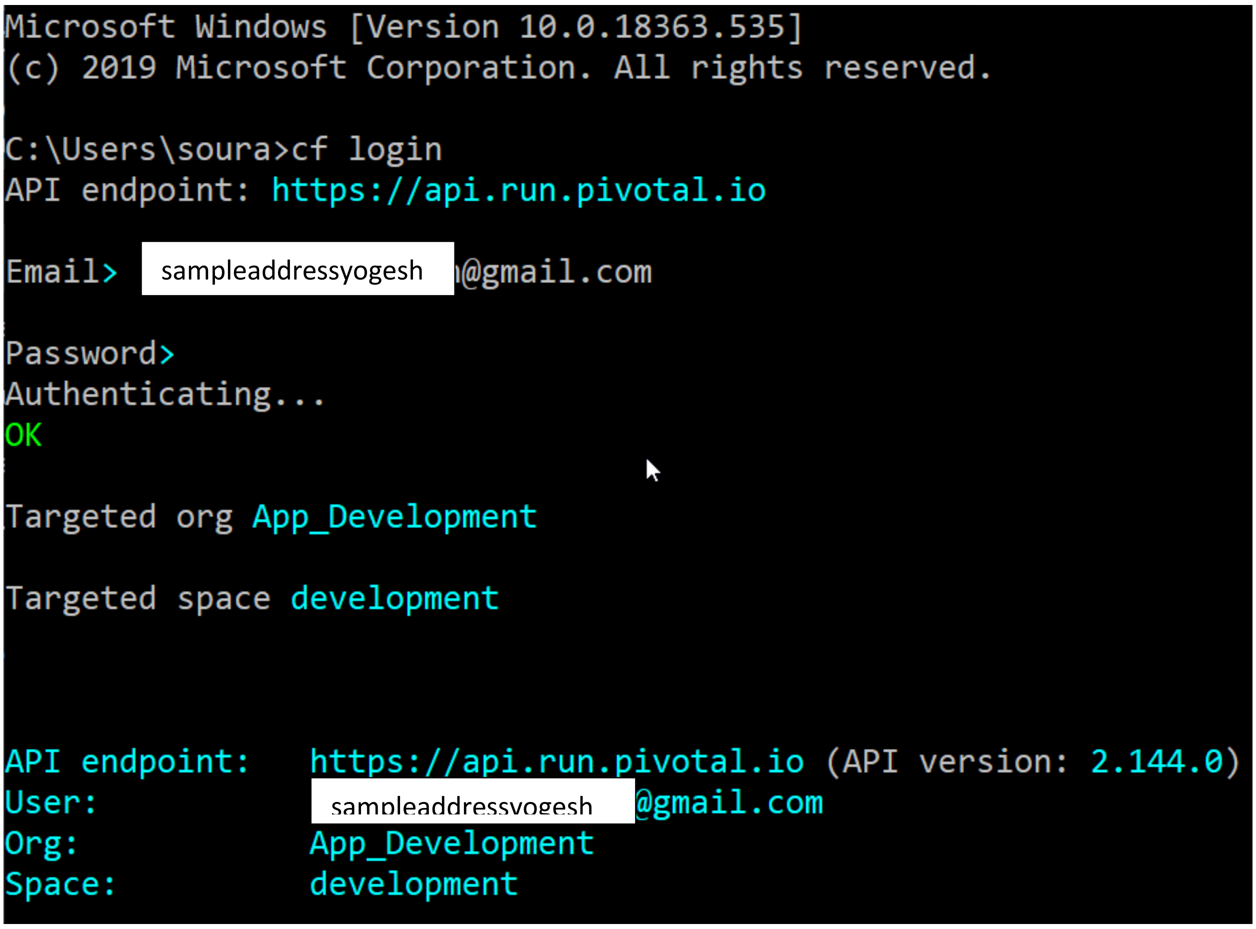
If you see this screen in your CMD, the installation is successful.

Now type the command to login via cf-cli

```
cf login -a https://api.run.pivotal.io
```

Next, enter your email and password. Now you are ready to push your app.

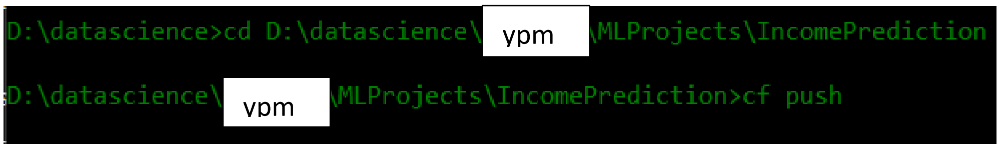
Now let’s go to the app which we have built.



```
Microsoft Windows [Version 10.0.18363.535]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\soura>cf login
API endpoint: https://api.run.pivotal.io

Email> sampleaddressyogesh@gmail.com

Password>
Authenticating...
OK

Targeted org App_Development

Targeted space development



API endpoint:   https://api.run.pivotal.io (API version: 2.144.0)
User:           sampleaddressyogesh@gmail.com
Org:            App_Development
Space:          development
```

Navigate to the project folder after downloading from the given below link:-

Then write cf push in the terminal.



```
D:\datascience>cd D:\datascience\ypm\MLProjects\IncomePrediction

D:\datascience\ypm\MLProjects\IncomePrediction>cf push
```

After the app is successfully deployed in the cloud, you will get the URL for your deployed app.

https://www.youtube.com/yogeshmurumkar