DEVELOPING PYTHON APPLICATIONS ON CLOUD

PyCon2016, HK

Xiulei ZHU
Engineer - IBM Bluemix
xiuleizh@cn.ibm.com

Agenda

- Bluemix Technical Introduce
- Running Python Web Application on Bluemix
- Developing Analytic Application Using Apache Spark and Python
- Bluemix Services Update

Does This Sound Like Dev Annoyance?

- Install runtime, container, and all libraries
- Install needed services (databases, mobile, etc)
- Bind the services to the application, ports/ips/firewalls
- Setup dynamic routing and load-balancer
- Setup four layers of built-in High-Availability
- Setup streaming logging aggregation
- Setup application performance monitoring
- Scale the application up to X instances
- Then repeat for dev, test, and production

What is Bluemix?

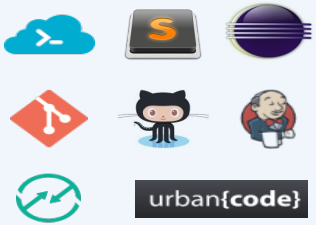
A platform for running virtually any application in the cloud without having to worry about the hardware, software and networking. You choose how you build, deploy, and manage your apps. Bluemix takes care of the rest.

Compute

Choose the level of **infrastructure abstraction** based on your app's architectural needs.

Dev Tooling

From editors to source code management to continuous delivery, you can **use Bluemix' powerful tooling** or easily **bring your own**.

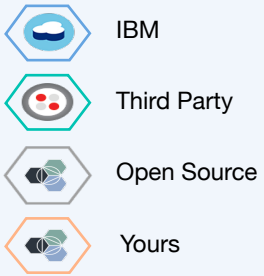


Location

Deploy apps to Bluemix **Public** (in a growing number of geos), your own **dedicated cloud** Bluemix, or one that runs **within your data center (Local*)**.

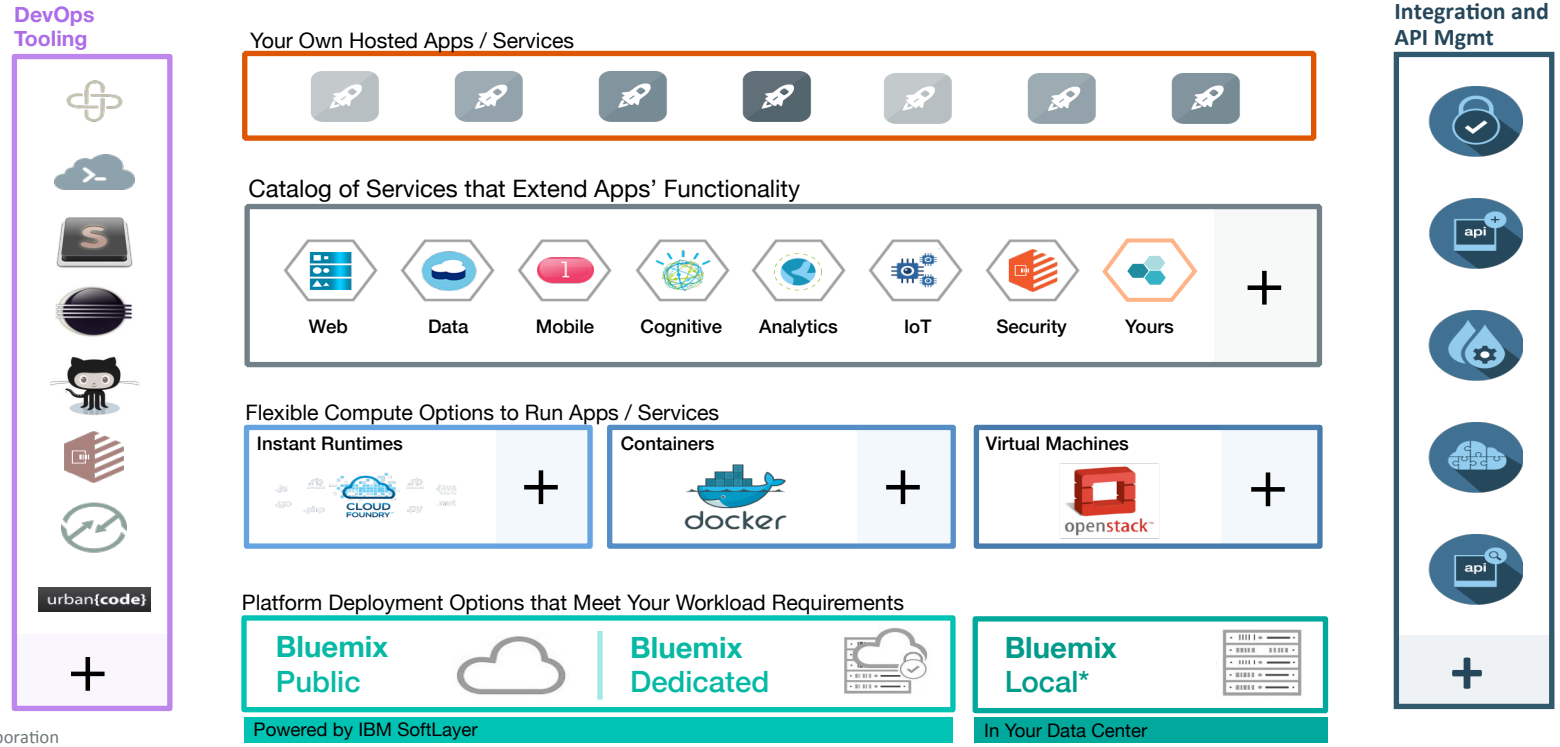
Services

Pick from a catalog of **IBM**, **third party**, **open source**, or **your own** services to extend your apps.



How does Bluemix Work?

Bluemix is underlined by three key open compute technologies: **Cloud Foundry**, **Docker**, and **OpenStack**. It extends each of these with a growing number of **services**, robust **DevOps tooling**, **integration** capabilities, and a seamless **developer experience**.



Why are Developers using Bluemix?

To **rapidly** bring products and services to market at **lower cost**

Go from zero to running code in a matter of minutes.

To **continuously deliver** new functionality to their applications

Automate the development and delivery of many applications.

To extend **existing investments** in IT infrastructure

Extend existing investments by connecting securely to on-premise infrastructure.

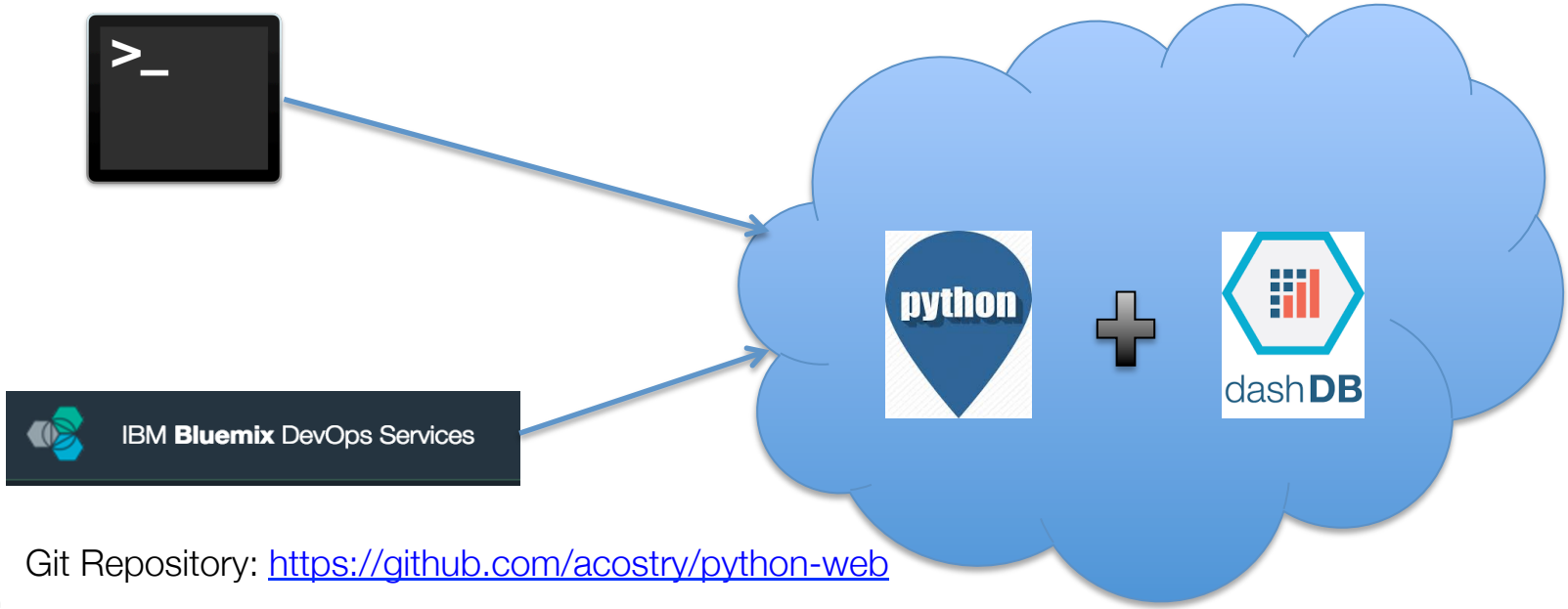
IBM

Agenda

- Bluemix Technical Introduce
- Running Python Web Application on Bluemix
- Developing Analytic Application Using Apache Spark and Python
- Bluemix Services Update

Scenario

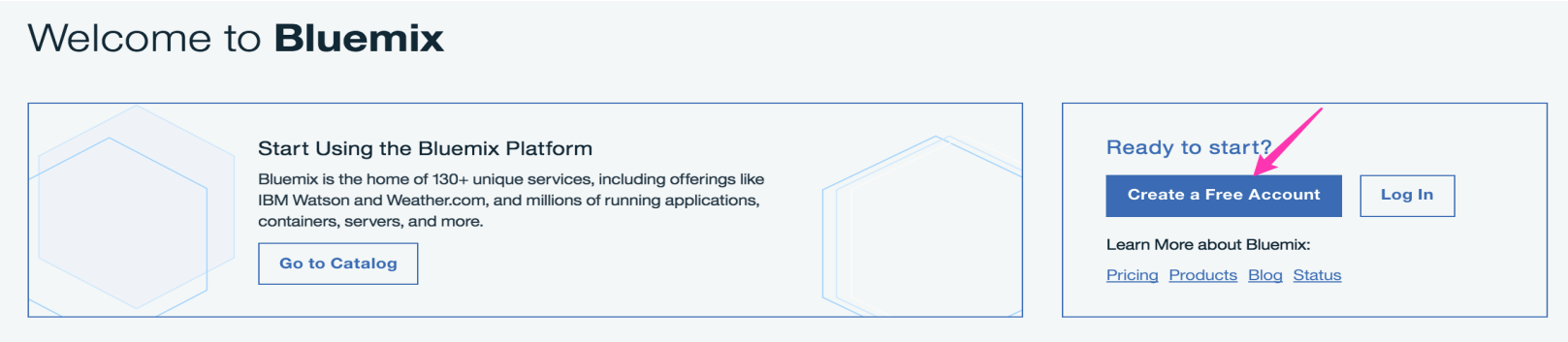
- The sample shows how to deploy a Python web application to IBM Bluemix that connects to the dashDB service. You can use the IBM Bluemix console user interface to manage your application and use Cloud Foundry command line tool to setup the dashDB, bind the service to your application, and deploy the application to Bluemix



Git Repository: https://github.com/acostry/python-web

Prerequisites

- Sign up for Bluemix
  - Access Bluemix website: http://www.bluemix.net



- Install Cloud Foundry CLI
  - Go to: https://github.com/cloudfoundry/cli

	Mac OS X 64 bit	Windows 64 bit	Linux 64 bit
Installers	pkg	zip	rpm / deb
Binaries	tgz	zip	tgz

Write the Code

This sample application contains four files— **Procfile**, **requirements.txt**, **runtime.txt** and **app.py**

- **Procfile** - Contains the command to run when your application starts on Bluemix

```
web: python app.py
```

- **requirements.txt** - Contains information about the application dependencies

```
Flask
ibm_db
```

- **runtime.txt** - Specify the version of Python to be used

```
python-2.7.10
```

- **app.py** - The Python script. Retrieves service information from the VCAP_SERVICES environment variable for connecting to database and executes a select query

```
if 'VCAP_SERVICES' in os.environ:
    db2info = json.loads(os.environ['VCAP_SERVICES'])['dashDB'][0]
    db2cred = db2info["credentials"]
db2conn = ibm_db.connect("DATABASE="+db2cred['db']+";HOSTNAME="+db2cred['hostname']+";PORT="+str(db2cred['port'])+";UID="+db2cred['username']
+";PWD="+db2cred['password']+";","","")
================================================================================
# main page to dump some environment information
@app.route('/')
def index():
    ......
    return page
================================================================================
if __name__ == '__main__':
    # Bind to PORT/HOST if defined, otherwise default to 5050/localhost.
    PORT = int(os.getenv('VCAP_APP_PORT', '5050'))
    HOST = str(os.getenv('VCAP_APP_HOST', 'localhost'))
    app.run(host=HOST, port=PORT)
```

Deploy the Sample Application

- Open a command line on your operating system and navigate to the directory containing your application
- Log in to Bluemix with `cf api https://api.ng.bluemix.net and cf login`
- Push the sample application to the Bluemix runtime environment with

```
cf push pyconhk2016-dashdb --no-start
```

```
Creating app pyconhk2016-dashdb in org xiuleizh@cn.ibm.com / space dev as xiuleizh@cn.ibm.com...
OK
Creating route pyconhk2016-dashdb.mybluemix.net...
OK
Binding pyconhk2016-dashdb.mybluemix.net to pyconhk2016-dashdb...
OK
Uploading pyconhk2016-dashdb...
Uploading app files from: /Users/acostry/B_Work/PyconHK2016/python-web-demo/py-flask-dashdb
Uploading 2.7K, 4 files
Done uploading
OK
```

Create and Bind dashDB to Application

- Create a dashDB Entry plan instance by running the following command:

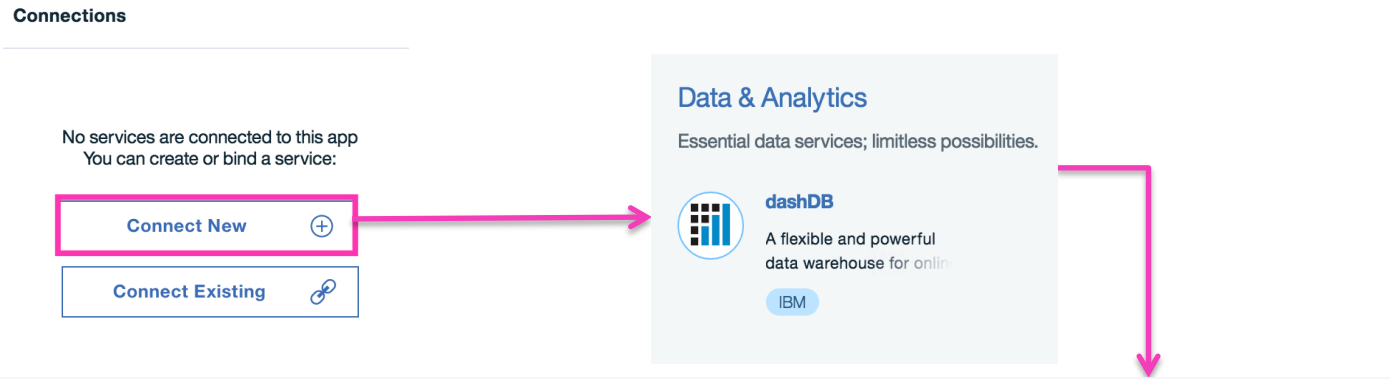
  ```
  cf create-service dashDB Entry dashDB-Sample
  ```

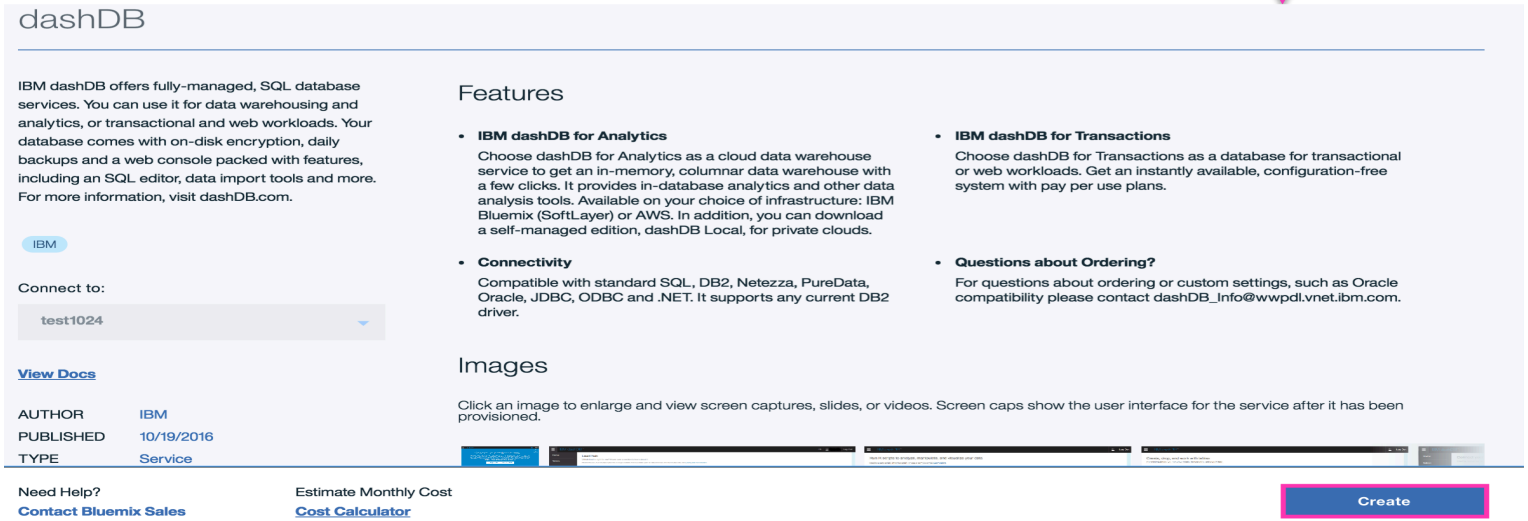
- Use the following command to bind dashDB service to your application:

  ```
  cf bind-service pyconhk2016-dashdb dashDB-Sample
  ```


```
{
  "dashDB": [
    {
      "credentials": {
        "port": 50000,
        "db": "BLUDB",
        "username": "dash5103",
        "ssljdbcurl": "jdbc:db2://dashdb-entry-yp-dal09-10.services.dal.bluemix.net:50001/BLUDB:sslConnection=true;",
        "host": "dashdb-entry-yp-dal09-10.services.dal.bluemix.net",
        "https_url": "https://dashdb-entry-yp-dal09-10.services.dal.bluemix.net:8443",
        "dsn": "DATABASE=BLUDB;HOSTNAME=dashdb-entry-yp-dal09-10.services.dal.bluemix.net;PORT=50000;PROTOCOL=TCPIP;UID=dash5103;PWD=uN0lzRM95xJ4;",
        "hostname": "dashdb-entry-yp-dal09-10.services.dal.bluemix.net",
        "jdbcurl": "jdbc:db2://dashdb-entry-yp-dal09-10.services.dal.bluemix.net:50000/BLUDB",
        "ssldsn": "DATABASE=BLUDB;HOSTNAME=dashdb-entry-yp-dal09-10.services.dal.bluemix.net;PORT=50001;PROTOCOL=TCPIP;UID=dash5103;PWD=uN0lzRM95xJ4;Security=SSL;",
        "uri": "db2://dash5103:uN0lzRM95xJ4@dashdb-entry-yp-dal09-10.services.dal.bluemix.net:50000/BLUDB",
        "password": "uN0lzRM95xJ4"
      },
      "syslog_drain_url": null,
      "label": "dashDB",
      "provider": null,
      "plan": "Entry",
      "name": "dashDB-Sample",
      "tags": [
        "big_data",
```

Work with dashDB from Bluemix UI





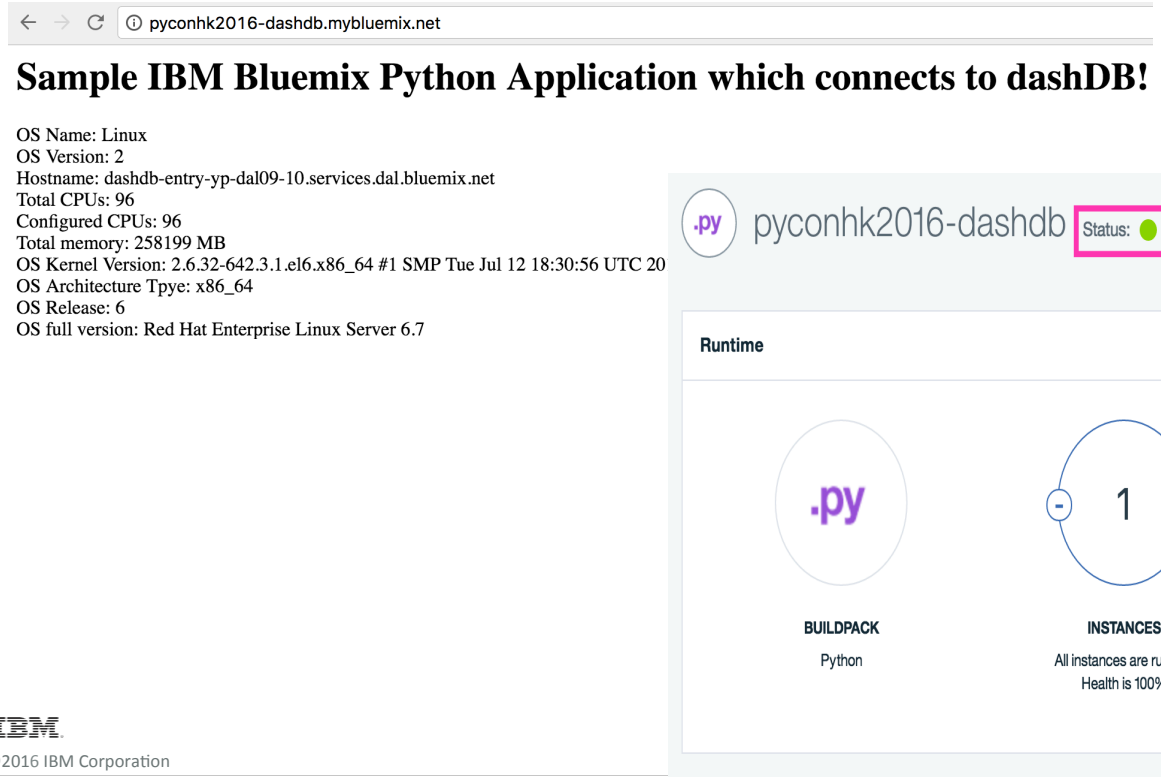
Start Sample Application

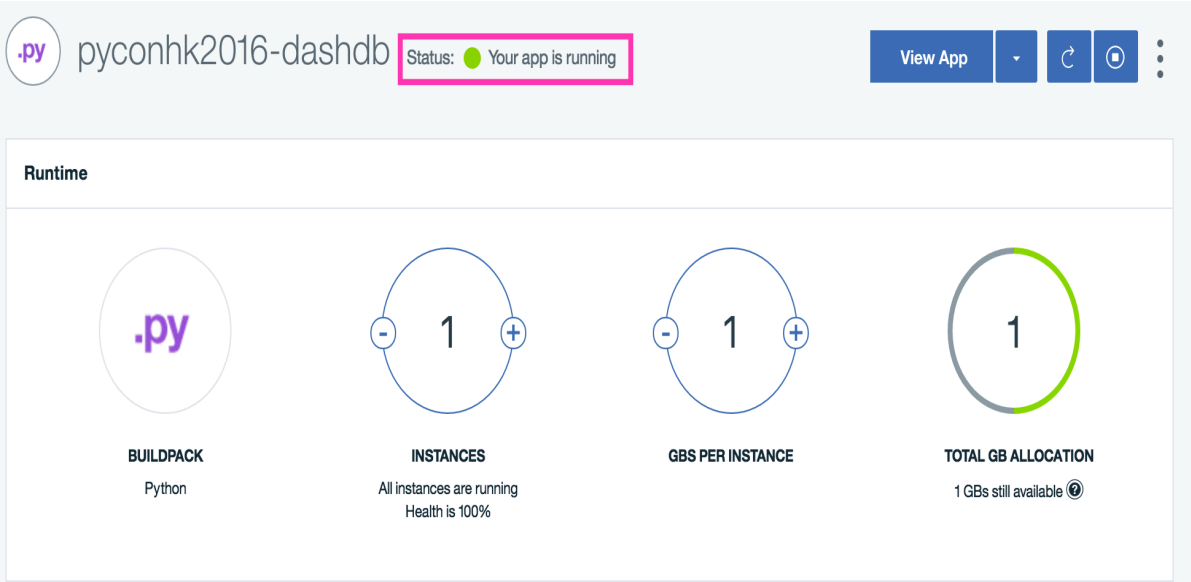
- Use `cf start pyconhk2016-dashdb` to start application

```
cf start pyconhk2016-dashdb
Starting app pyconhk2016-dashdb in org xiuleizh@cn.ibm.com / space dev as xiuleizh@cn.ibm.com...
-----> Downloaded app package (4.0K)
-------> Buildpack version 1.5.5
-----> Installing python-2.7.10
Downloaded [file:///var/vcap/data/dea_next/admin_buildpacks/69221686-9e15-442c-bdec-
fd0fb5fc5470_ca60152b78be3f93067cd4f03d01e2db810ac2a8/dependencies/https___pivotal-buildpacks.s3.amazonaws.com_concourse-
binaries_python_python-2.7.10-linux-x64.tgz]
     $ pip install -r requirements.txt
.......
-----> Uploading droplet (62M)
0 of 1 instances running, 1 starting
1 of 1 instances running
App started
OK
App pyconhk2016-dashdb was started using this command `python app.py`
Showing health and status for app pyconhk2016-dashdb in org xiuleizh@cn.ibm.com / space dev as xiuleizh@cn.ibm.com...
OK
requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: pyconhk2016-dashdb.mybluemix.net
last uploaded: Fri Oct 21 08:50:51 UTC 2016
stack: cflinuxfs2
buildpack: python 1.5.5

     state     since                    cpu    memory        disk           details
#0   running   2016-10-21 04:55:39 PM   0.1%   81.5M of 1G   214.6M of 1G
```

Verify Sample Application is Running

- Use the browser URL from the application dashboard to run the application. The URL will look something like `http://pyconhk2016-dashdb.mybluemix.net`

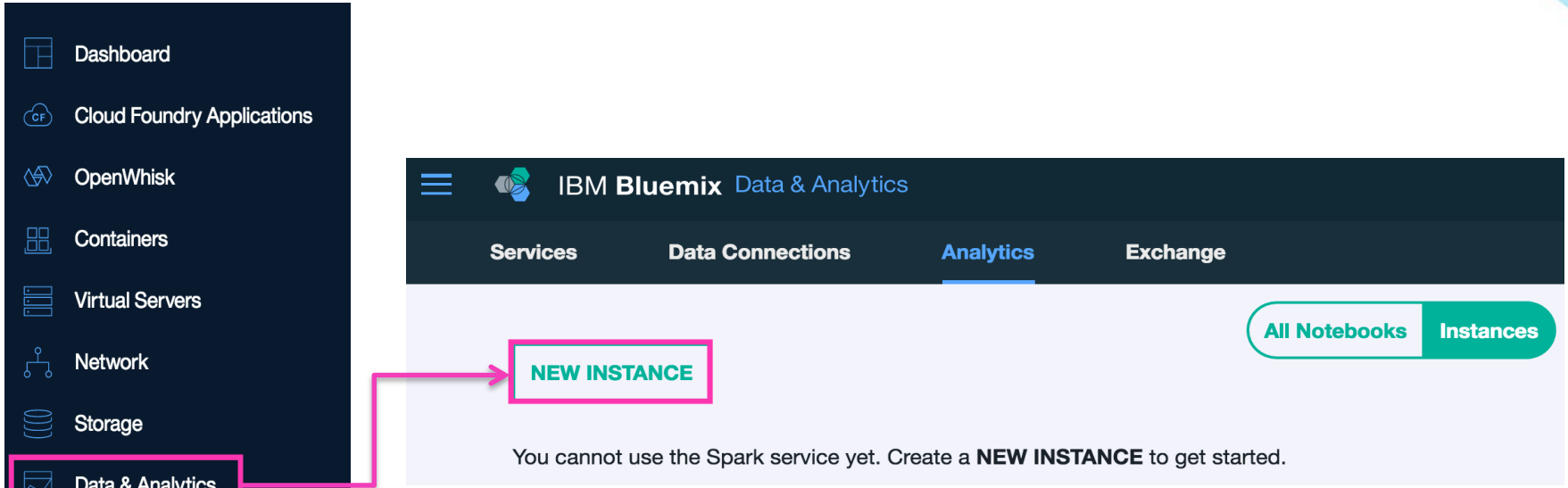




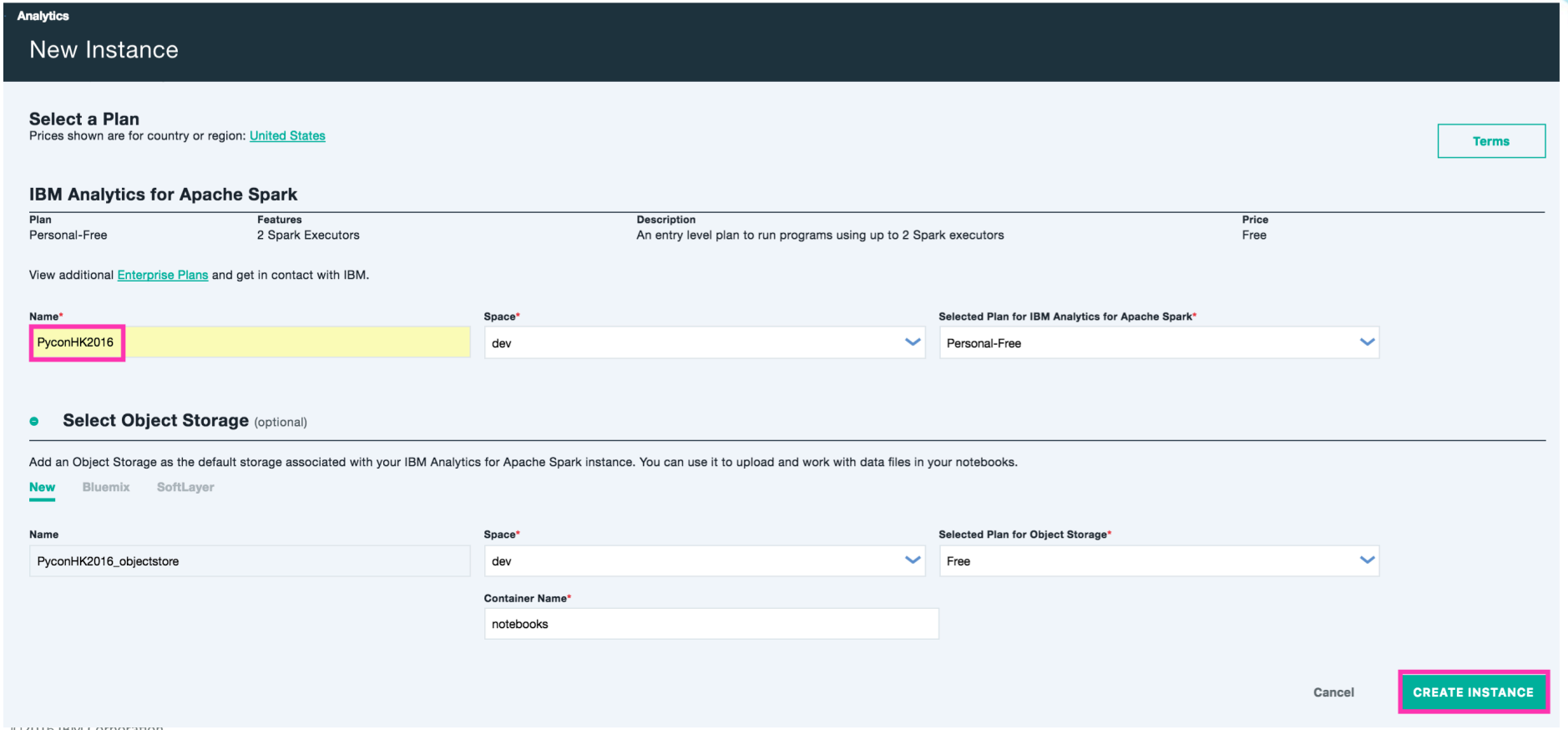
Agenda

- Bluemix Technical Introduce
- Running Python Web Application on Bluemix
- Developing Analytic Application Using Apache Spark and Python
- Bluemix Services Update

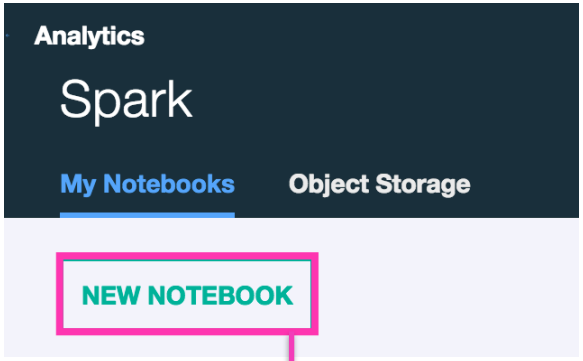
Create a Spark Instance

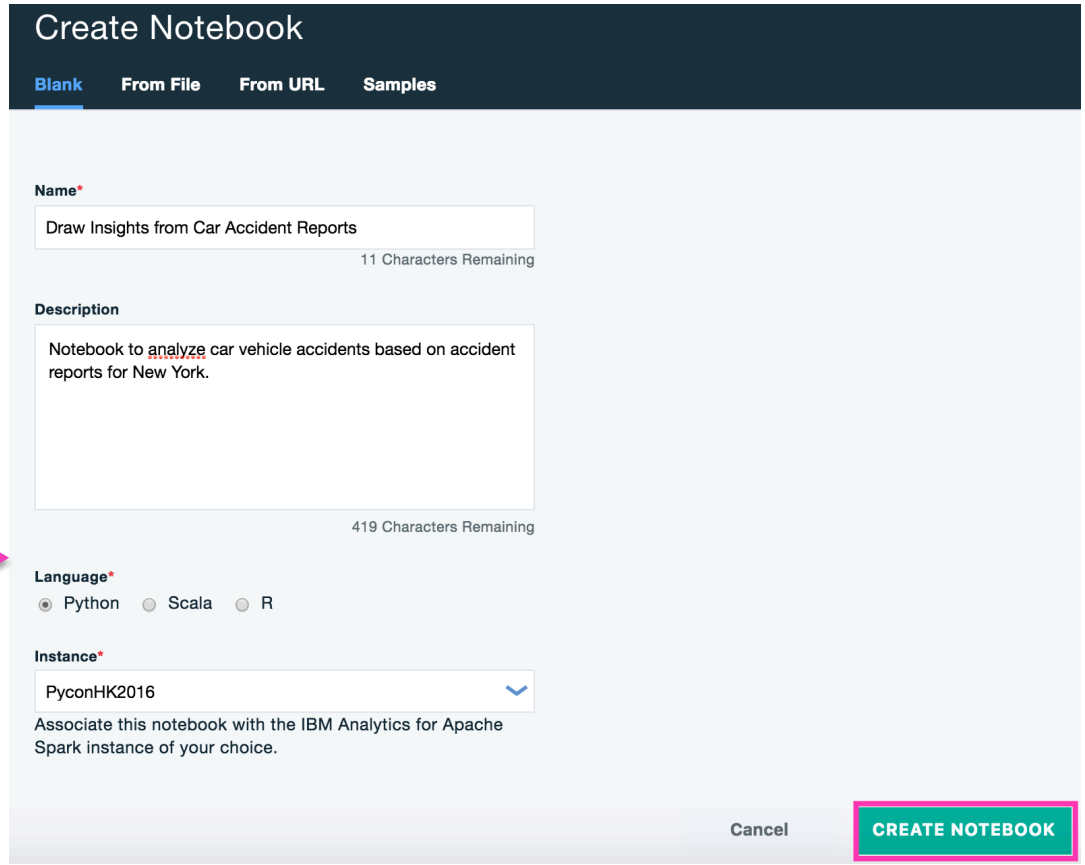


Create New Spark Instance

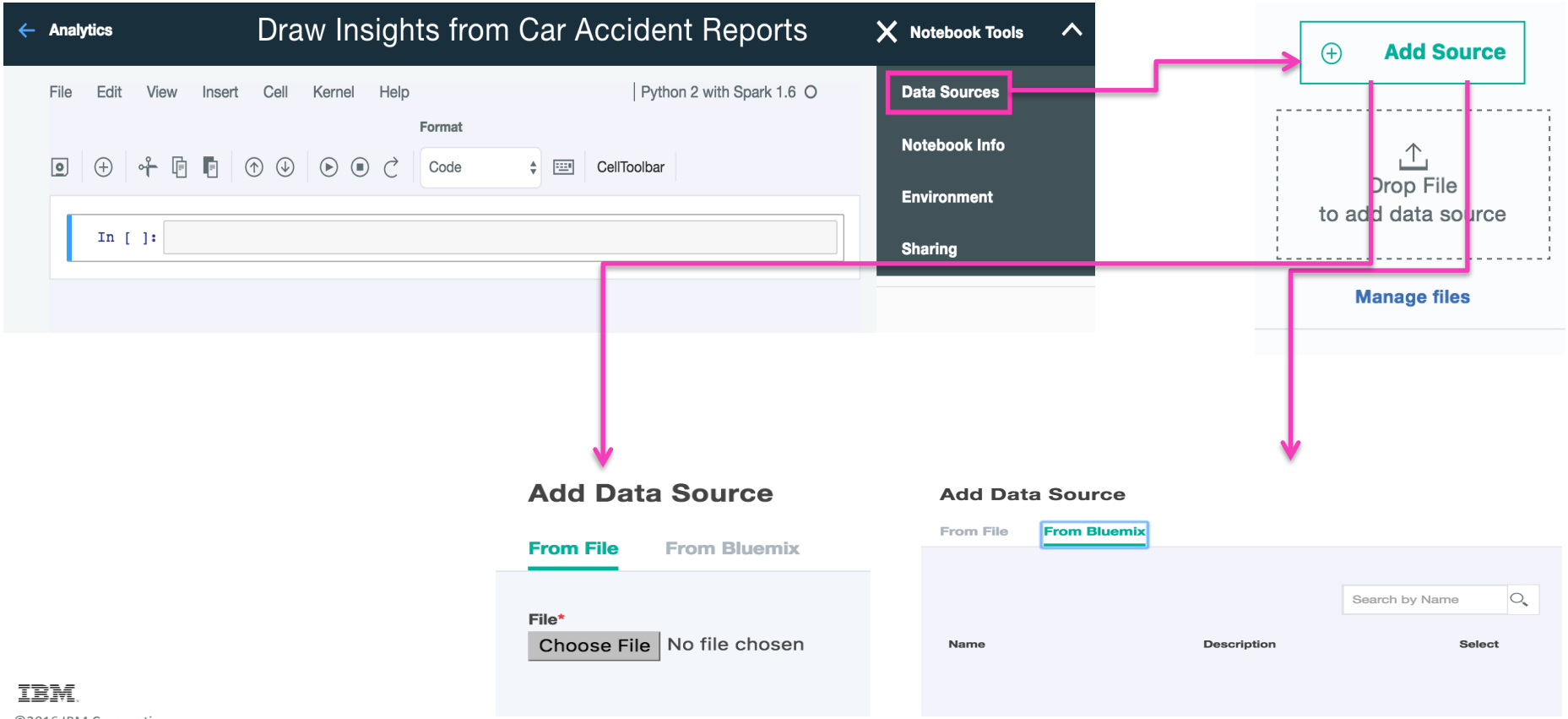


Create New IPython Notebook

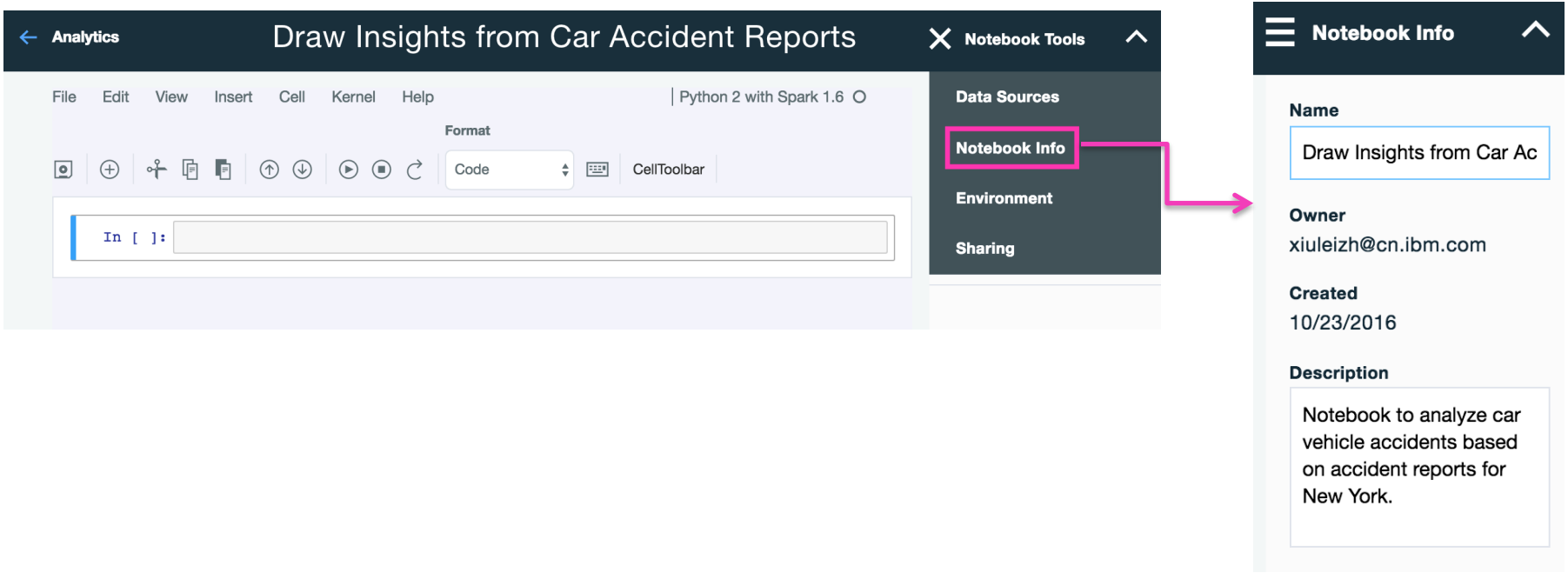




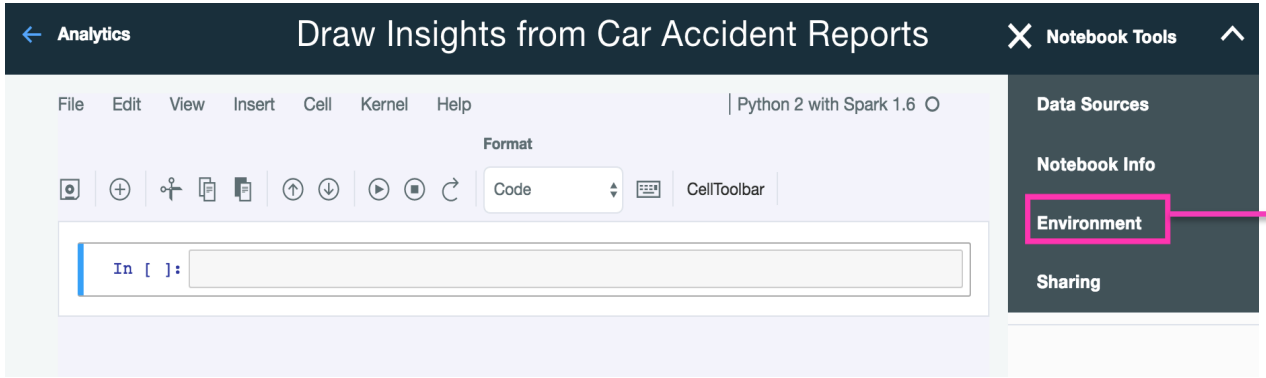
Notebook: Data Source

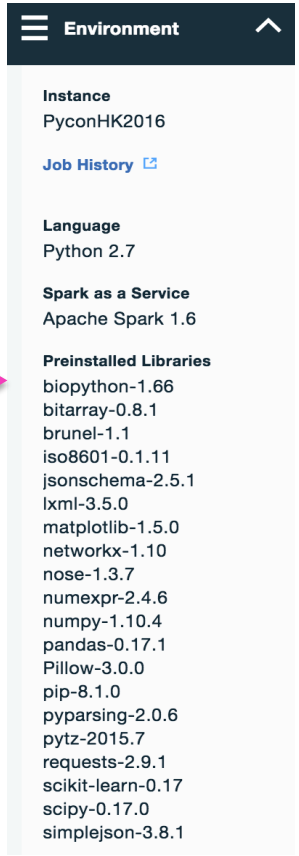


Notebook: Info

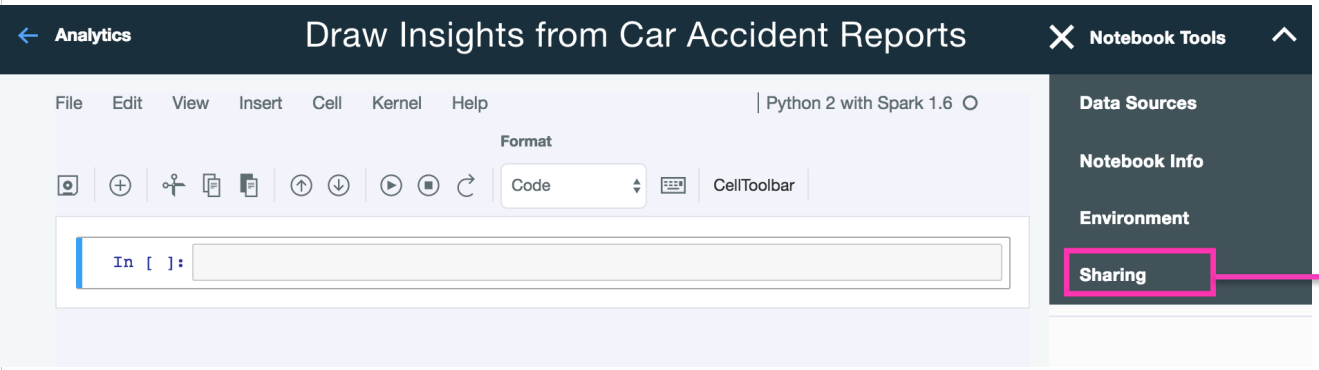


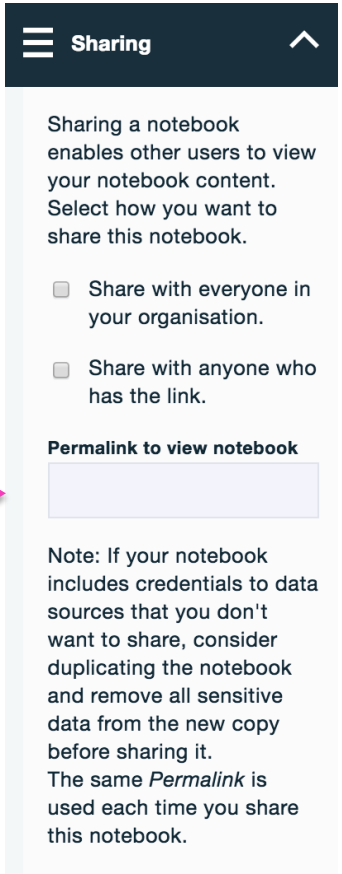
Notebook: Environment





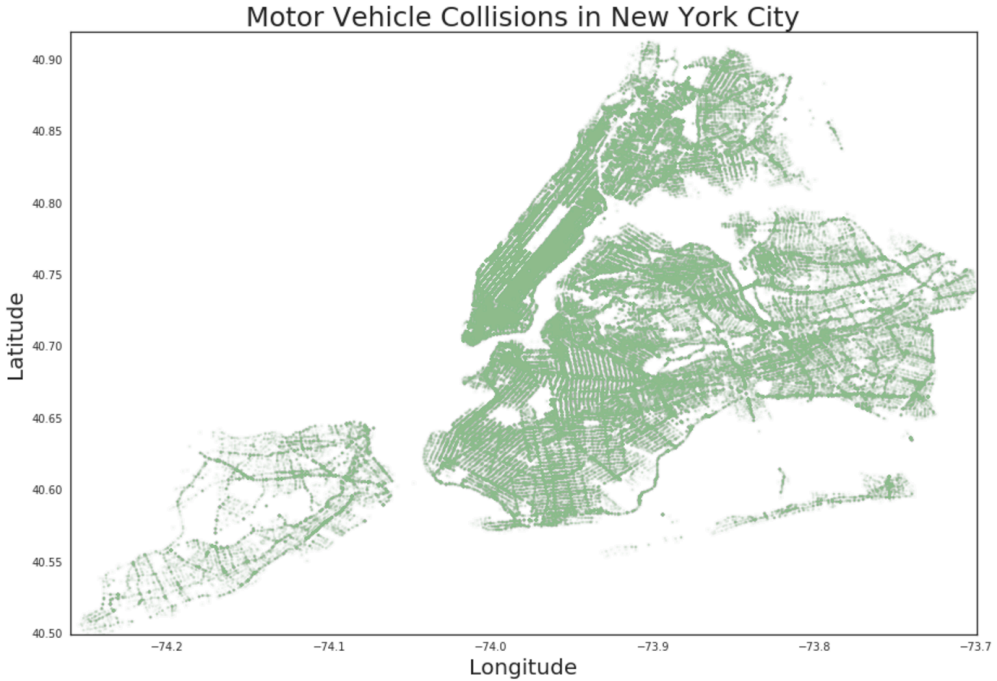
Notebook: Sharing





Draw Insights from Car Accident Reports

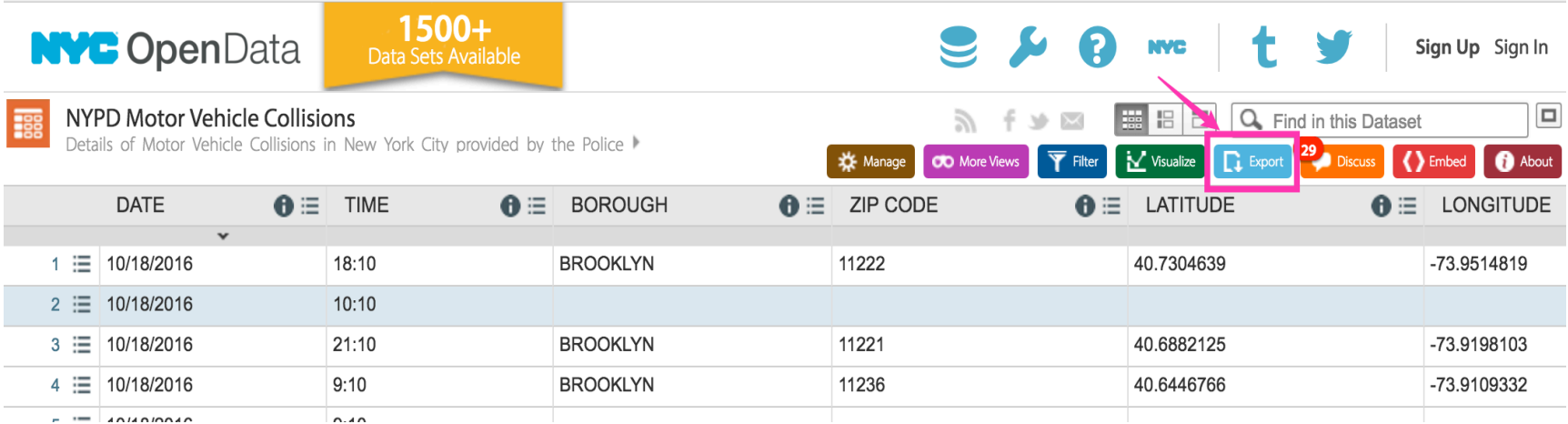
- This demo shows how to analyze car vehicle accidents based on accident reports for New York
- The analysis steps show how you can use the information about accidents to learn more about the possible causes for collisions
- You will learn how to install additional Python packages, how to add external PySpark modules, and how to perform descriptive data analysis



Git Repository: https://github.com/acostry/bmx-spark-analyze

Get Data

- Download data abut car accidents in the New York area from https://data.cityofnewyork.us/Public-Safety/NYPD-Motor-Vehicle-Collisions/h9gi-nx95
- Click Export to download the data as a CSV file



	DATE	TIME	BOROUGH	ZIP CODE	LATITUDE	LONGITUDE
1	10/18/2016	18:10	BROOKLYN	11222	40.7304639	-73.9514819
2	10/18/2016	10:10				
3	10/18/2016	21:10	BROOKLYN	11221	40.6882125	-73.9198103
4	10/18/2016	9:10	BROOKLYN	11236	40.6446766	-73.9109332

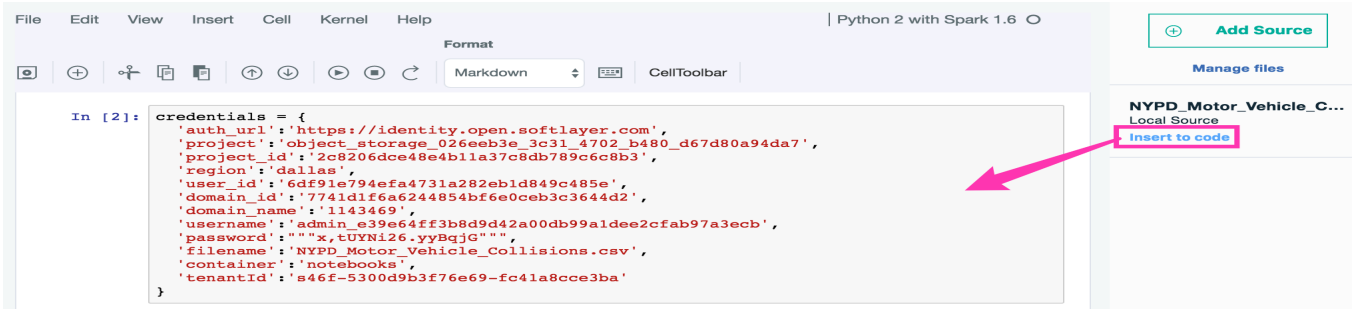
- After download the file, load the data file to the notebook by clicking **Palette**>**Data Sources**. Click **Add Source**, select From file, and browse to the data file.

Access Data

- Set the Hadoop configuration with the Object Storage instance service credentials (Note: You will not be using Hadoop in this sample; however Spark leverages some Hadoop components.)

```
def set_hadoop_config(credentials):
    prefix = "fs.swift.service." + credentials['name']
    hconf = sc._jsc.hadoopConfiguration()
    hconf.set(prefix + ".auth.url", credentials['auth_url']+'/v3/auth/tokens')
    hconf.set(prefix + ".auth.endpoint.prefix", "endpoints")
    hconf.set(prefix + ".tenant", credentials['project_id'])
    hconf.set(prefix + ".username", credentials['user_id'])
    hconf.set(prefix + ".password", credentials['password'])
    hconf.setInt(prefix + ".http.port", 8080)
    hconf.set(prefix + ".region", credentials['region'])
    hconf.setBoolean(prefix + ".public", True)
```

- Object Storage instance service credentials



- Set the Hadoop configuration and give it a name, for example, `keystone`:

```
credentials['name'] = 'keystone'
set_hadoop_config(credentials)
```

Load Data

- Use the pyspark-csv module load the NYPD motor vehicle collisions data set from Object Storage into an RDD and then convert the RDD into a Spark DataFrame.(pyspark-csv is an external PySpark module and works like the pandas read_csv function.)

```
from __future__ import division
import numpy as np

from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)

# adding the PySpark modul to SparkContext
sc.addPyFile("https://raw.githubusercontent.com/seahboonsiew/pyspark-csv/master/pyspark_csv.py")
import pyspark_csv as pycsv

collisions = sc.textFile("swift://" + credentials['container'] + "." + credentials['name'] + "/NYPD_Motor_Vehicle_Collisions.csv")

def skip_header(idx, iterator):
    if (idx == 0):
        next(iterator)
    return iterator

collisions_header = collisions.first()


collisions_header_list = collisions_header.split(",")
collisions_body = collisions.mapPartitionsWithIndex(skip_header)

# filter not valid rows
collisions_body = collisions_body.filter(lambda line : len(line.split(","))>29)

# create Spark DataFrame using pyspark-csv
collisions_df = pycsv.csvToDataFrame(sqlContext, collisions_body, sep=",", columns=collisions_header_list)
collisions_df.cache()
```

Load Visualization Packages

- **Seaborn**, a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics. You must install the seaborn package by using !pip.

```
!pip install --user seaborn
```

- **Matplotlib**, a basic plotting library for Python

```
%matplotlib inline

import matplotlib.pyplot as plt
# matplotlib.patches allows us create colored patches, we can use for legends in plots
import matplotlib.patches as mpatches
# seaborn also builds on matplotlib and adds graphical features and new plot types
import seaborn as sns
import pandas as pd
```

Explore Data

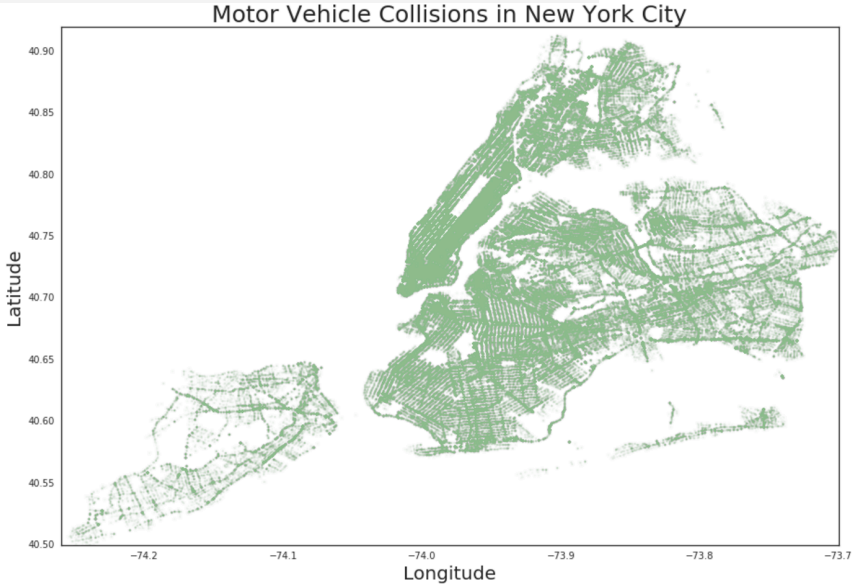
- Start exploring it and visualizing patterns by using scatter plots. In the code cell, select the columns with data that you want to explore, for example, NUMBER OF PERSONS INJURED or CONTRIBUTING FACTOR VEHICLE, and transform these columns into a pandas DataFrame.

```
collisions_pd = collisions_df[collisions_df['LATITUDE'] != 0][['LATITUDE', 'LONGITUDE', 'DATE', 'TIME',
                                                               'BOROUGH', 'ON STREET NAME', 'CROSS STREET NAME',
                                                               'NUMBER OF PERSONS INJURED', 'NUMBER OF PERSONS KILLED',
                                                               'CONTRIBUTING FACTOR VEHICLE 1']].toPandas()

collisions_pd.columns = ['Latitude', 'Longitude', 'Date', 'Time', 'Borough', 'On Street',
                         'Cross Street', 'Persons Injured', 'Persons Killed', 'Contributing Factor']

#divide dataset in accidents which are: fatal, non-lethal but with person damage, non of the above
killed_pd = collisions_pd[collisions_pd['Persons Killed']!=0]
injured_pd = collisions_pd[np.logical_and(collisions_pd['Persons Injured']!=0, collisions_pd['Persons Killed']==0)]
nothing_pd = collisions_pd[np.logical_and(collisions_pd['Persons Killed']==0, collisions_pd['Persons Injured']==0)]
```

Create an explorative scatter plot of the data

- Using an explorative scatter plot is a way to analyze certain characteristics of the data set.
- Create an initial explorative scatter plot of all collisions by using the latitude and longitude information in the raw data:

```
#adjust settings
sns.set_style("white")
plt.figure(figsize=(15,10))

#create scatterplots
plt.scatter(collisions_pd.Longitude, collisions_pd.Latitude, alpha=0.05, s=4, color='darkseagreen')

#adjust more settings
plt.title('Motor Vehicle Collisions in New York City', size=25)
plt.xlim((-74.26,-73.7))
plt.ylim((40.5,40.92))
plt.xlabel('Longitude',size=20)
plt.ylabel('Latitude',size=20)

plt.show()
```

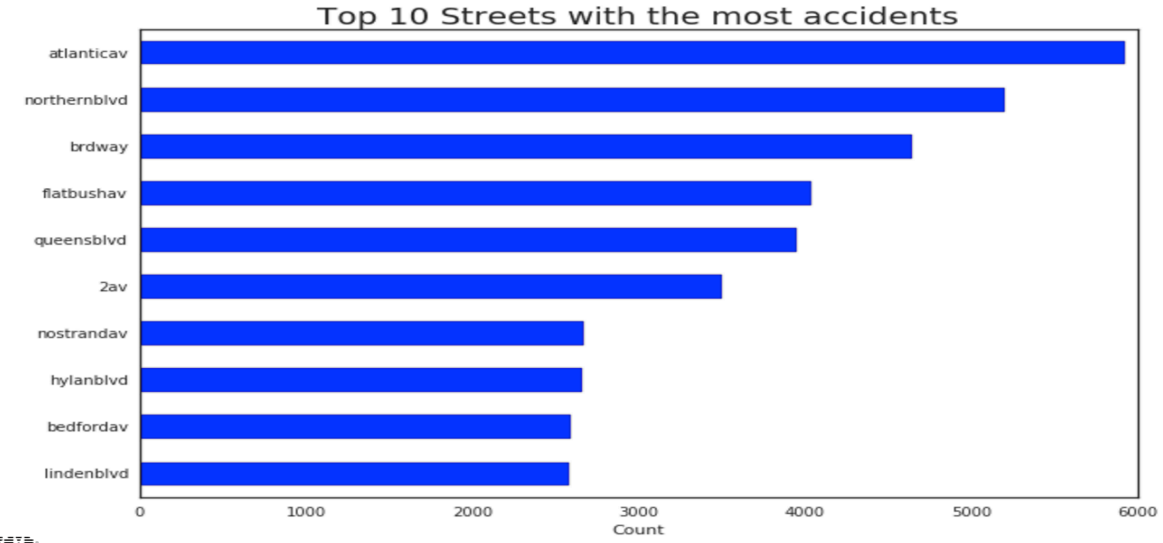


- Although this scatter plot is not a street map of New York City, the scatter plot dots roughly map to the street map of New York City. You see very few collisions in Central Park or on bridges, as opposed to street crossings and curves, where there is a noticeably higher density of collisions.

Determine the streets with the most accidents

- Find the top ten streets in New York where the most accidents occurred. Display the results in a bar graph and as a scatter plot

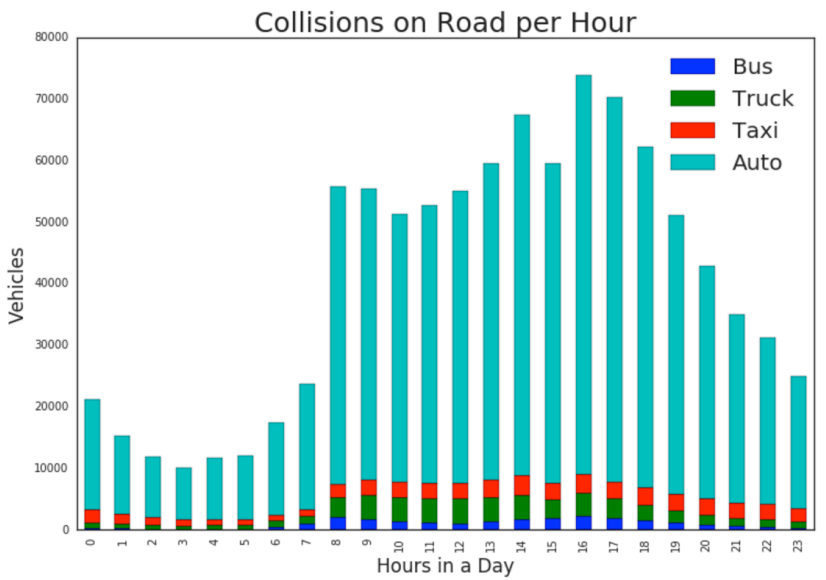
```
from pyspark.sql import functions as F
collisions_final_df = collisions_final.toDF()
plottingdf = collisions_final_df.groupBy("Borough", "Street").agg(F.sum("NumberOfAccidents").alias("sum(NumberOfAccidents)")).\
sort(F.desc('sum(NumberOfAccidents)')).limit(10).toPandas()
plottingdf[['sum(NumberOfAccidents)']].plot(kind='barh', figsize=(11,7), legend=False)
plt.title('Top 10 Streets with the most accidents', size=20)
plt.xlabel('Count')
plt.yticks(range(10), plottingdf['Street'])
plt.gca().invert_yaxis()
plt.show()
```



Determine when the most accidents occurred

- Find out at what time of the day the most accidents occurred and see if you can detect any interesting patterns by running the following cell

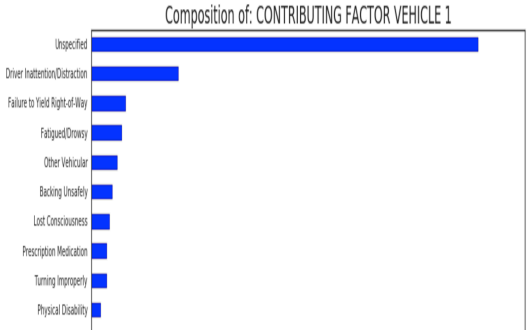
```
from pyspark.sql import functions as F

hourplot = collisions_final_df[['Bus','Truck','Taxi','Other','Hour','Auto']].groupBy('Hour')\
.agg(F.sum("Bus").alias("Bus"), F.sum("Truck").alias("Truck"), F.sum("Taxi").alias("Taxi"),\
F.sum("Other").alias("Other"),F.sum("Auto").alias("Auto")).toPandas()

hourplot[['Bus', 'Truck', 'Taxi', 'Auto']].plot(stacked=True, kind='bar',figsize=(12,8), alpha=1)
#'SUM(Other)',
plt.xlabel('Hours in a Day', size=17)
plt.ylabel('Vehicles', size=17)
plt.legend(loc='best', prop={'size':20}, framealpha=0)
plt.title('Collisions on Road per Hour', size=25)
plt.show()
```

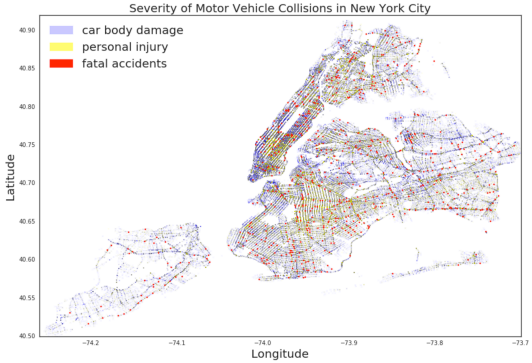


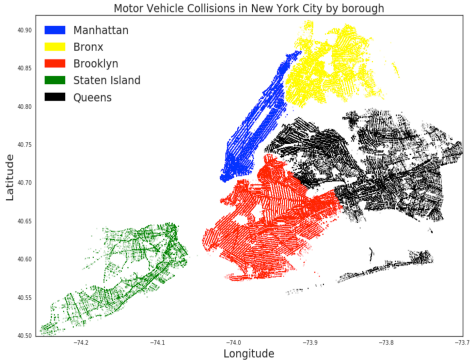
- This plot shows collisions spread across a day, with peaks during the morning and afternoon rush hours. You can see that significantly more collisions occurred during the afternoon rush hour than during the morning rush hour. Also, the most collisions involve cars by far, while buses, taxis, and trucks are involved in accidents a lot less frequently.

Summary

- The demo show you how to analyze car car accidents and how you can use this information to learn more about the causes for collisions. If you extract this type of information from the data, you can use it to help develop measures for preventing car accidents in accident hotspots.
- Above demo is not a complete version, get more details from Git repository: https://github.com/acostry/bmx-spark-analyze







Agenda

- Bluemix Technical Introduce
- Running Python Web Application on Bluemix
- Developing Analytic Application Using Apache Spark and Python
- Bluemix Services Update

Build your apps, your way

- Use a combination of the most prominent open-source compute technologies to power your apps. Then, let Bluemix handle the rest.

OpenWhisk

Event-driven apps, deployed in a serverless environment.

Instant Runtimes

App-centric runtime environments based on Cloud Foundry.

CLOUD FOUNDRY

IBM Containers

Portable and consistent delivery of your app without having to manage an OS.

docker

Virtual Service

Get the most flexibility and control over your environment with VMs.

openstack POWERED

Bare Metal

For the ultimate performance and scale

openstack POWERED

Ease of getting started → Full stack Control

Broad range of IBM & 3rd Party Services

- **Starters**
  - Boilerplates
- **Compute**
  - Runtimes
  - Containers
- **Services**
  - Watson
  - Mobile
  - DevOps
  - Web and Application
  - Integration
  - Data & Analytics
  - Security
  - Business Analytics
  - Internet of Things
- **Provider**
  - Beta
  - My Org

Commerce

Healthcare

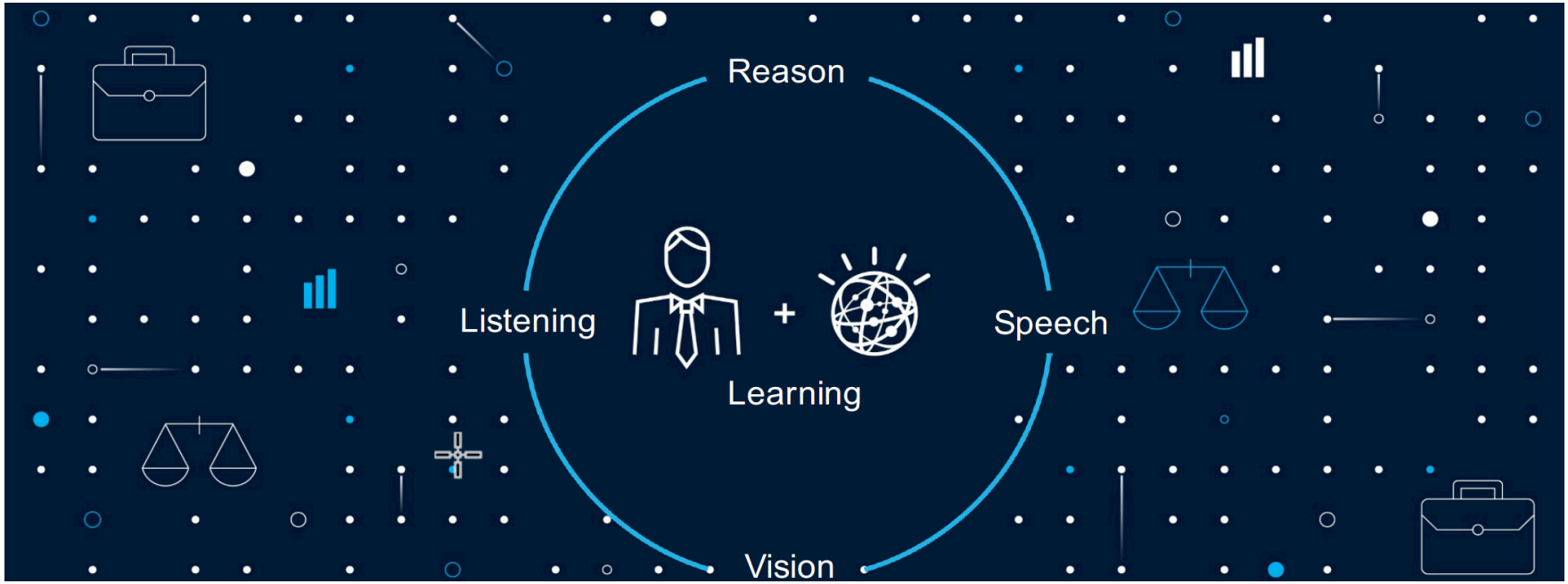
Analytics

Watson

Security

Watson on Bluemix

- Watson is creating a new partnership between people and computers that enhances, scales and accelerates human expertise



- IBM Watson services available on Bluemix are the building blocks for developers to create the next generation of cognitive applications to transform the way businesses engage with their customers, discover, innovate and make decisions

IBM Mobile Services

Core App & Platform Services

Core services that just about every App needs

Push
Increase User Engagement with Mobile Push Notifications

Cloudant
The Ultimate Data Store for Mobile Apps

Mobile Client Access
Simplified Mobile App User Authentication

StrongLoop Arc
Enterprise Node.js to build, deploy, scale manage and deploy your mobile APIs

App Lifecycle Services

Capabilities to manage the lifecycle of your Apps

Mobile App Server
Manage, secure, and optimize enterprise mobile apps

AppScan Mobile Analyzer
Identify Mobile App Security Exposures

Mobile Quality Assurance
Continuously Improve Mobile User Experience

Contextual Services

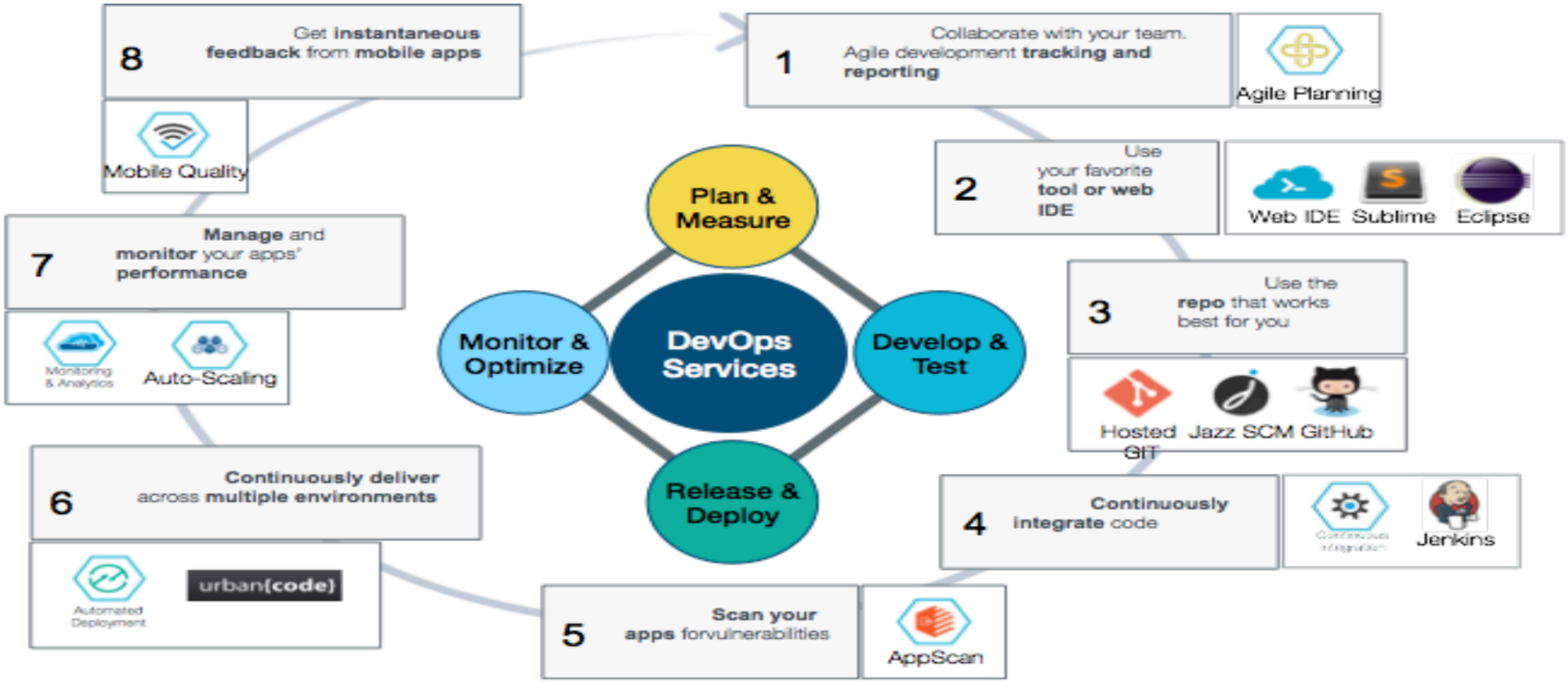
Capabilities to enable apps to make users more productive based on context

Mobile Content Manager
Create, manage, and deliver more relevant in-app content

Presence Insights
Enables Apps with Intelligent Mobile Location Data

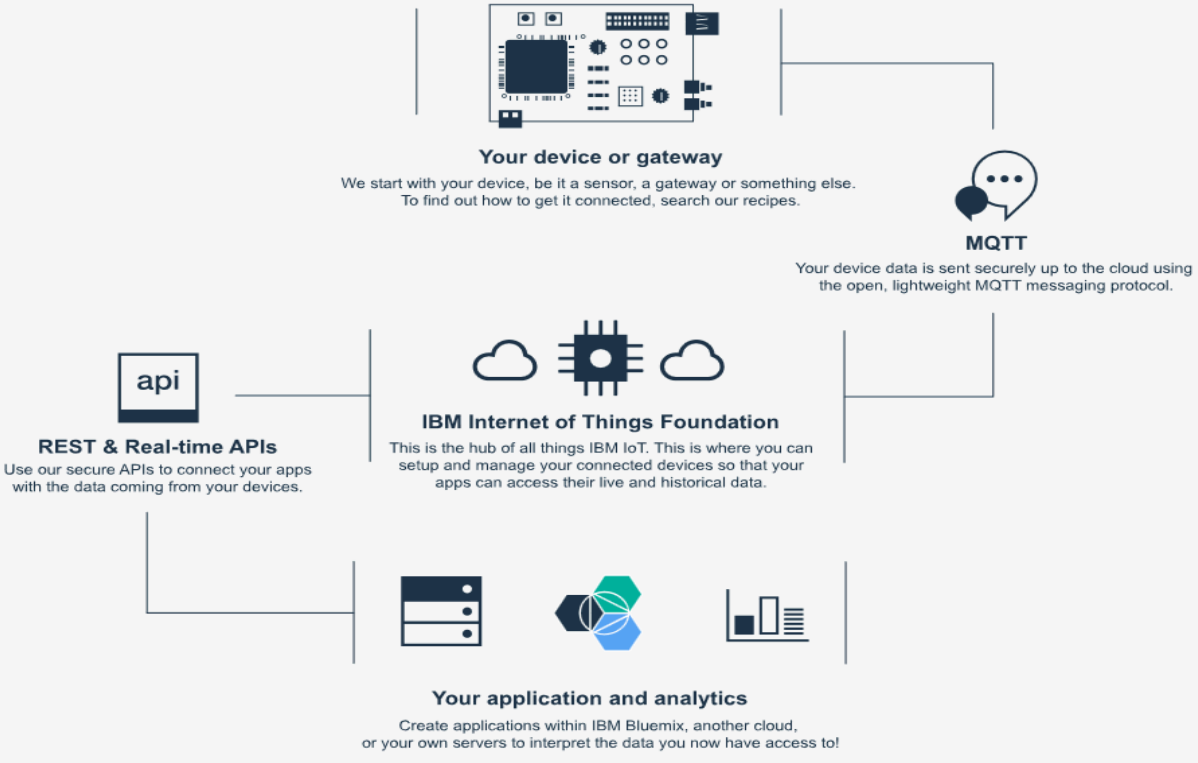
Continuous Integration & Delivery

- Fit for an enterprise, the DevOps experience is unified and open across compute technologies, Bluemix delivery methods, and integrated systems

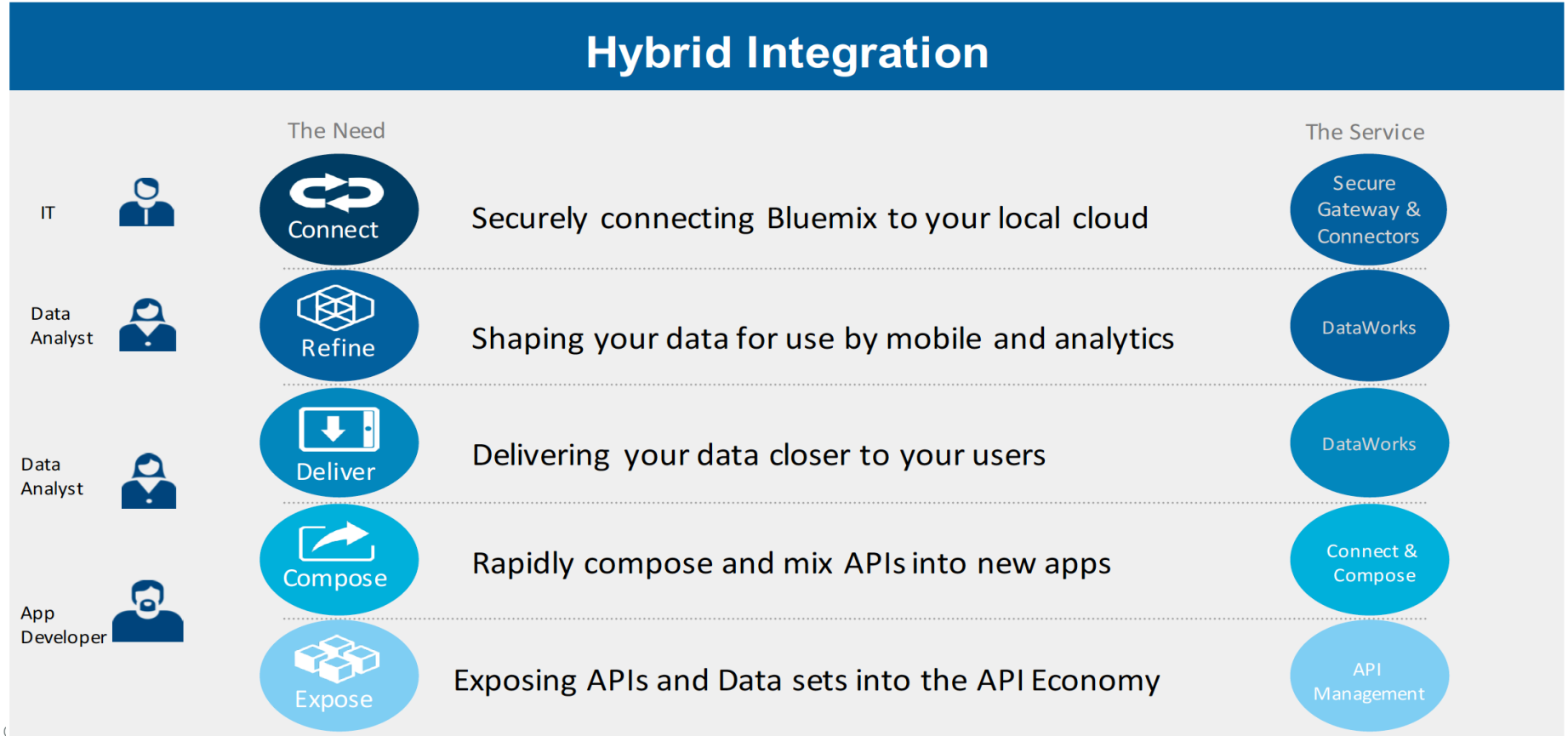


Internet of Things Foundation

- Set up and manage your connected devices so all your app's can access the data



New Patterns of Integration



How do I get started?

Sign up in minutes. Pay for what you use.

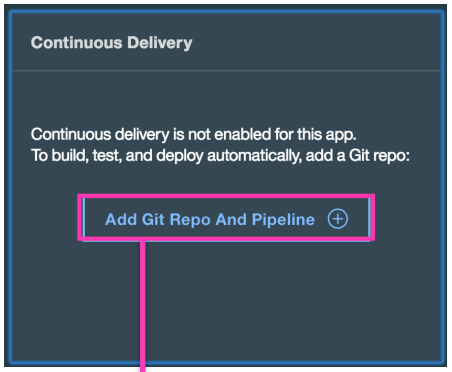
- Free adoption
  - **30 day trial (no credit card required)** – Designed to allow testing of an entire application on the platform
  - **Free for every service** - Encourages experimentation of new services for applications already running on Bluemix
- Multiple Commitment Models
  - **Pay-as-you-go** - optimized for flexibility, no term commitment
  - **Subscription** - term based optimized for cost, discounted from pay as you go rates
- Self Service
  - Zero to coding in **less than 5 minutes**
  - **Credit card** over the web in many countries

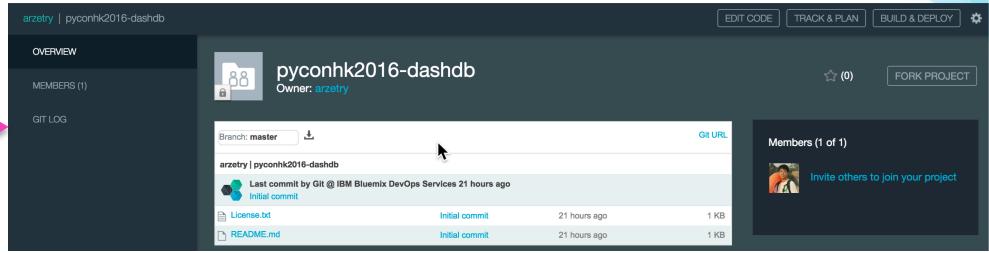
IBM **Bluemix**™

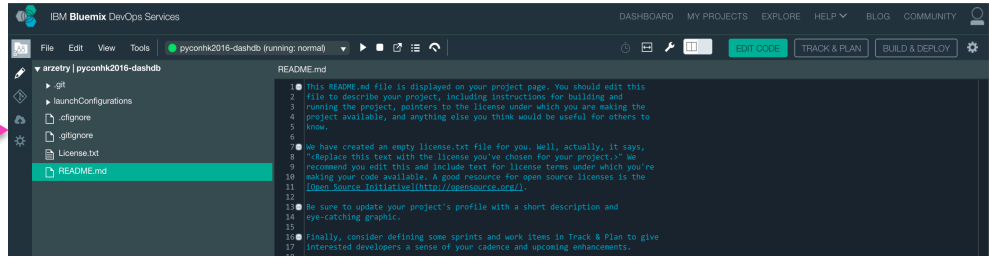
http://www.bluemix.net

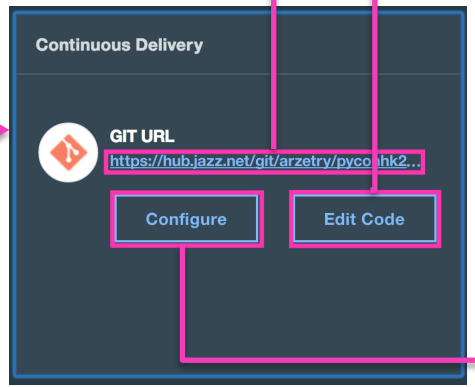
Backup

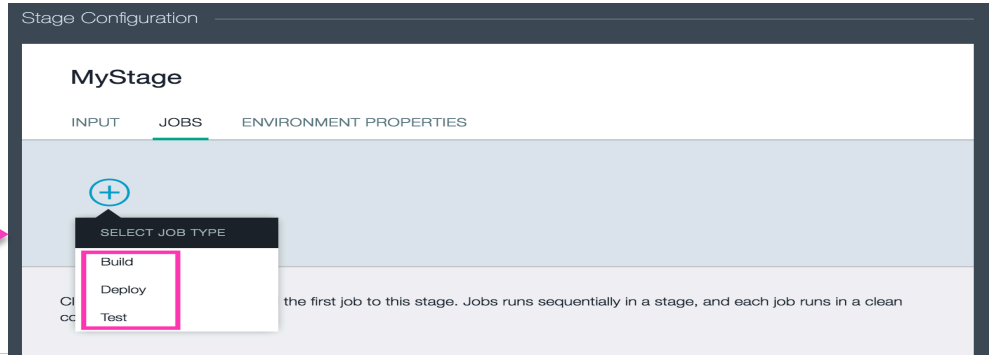
Continuous Delivery (Pipeline)







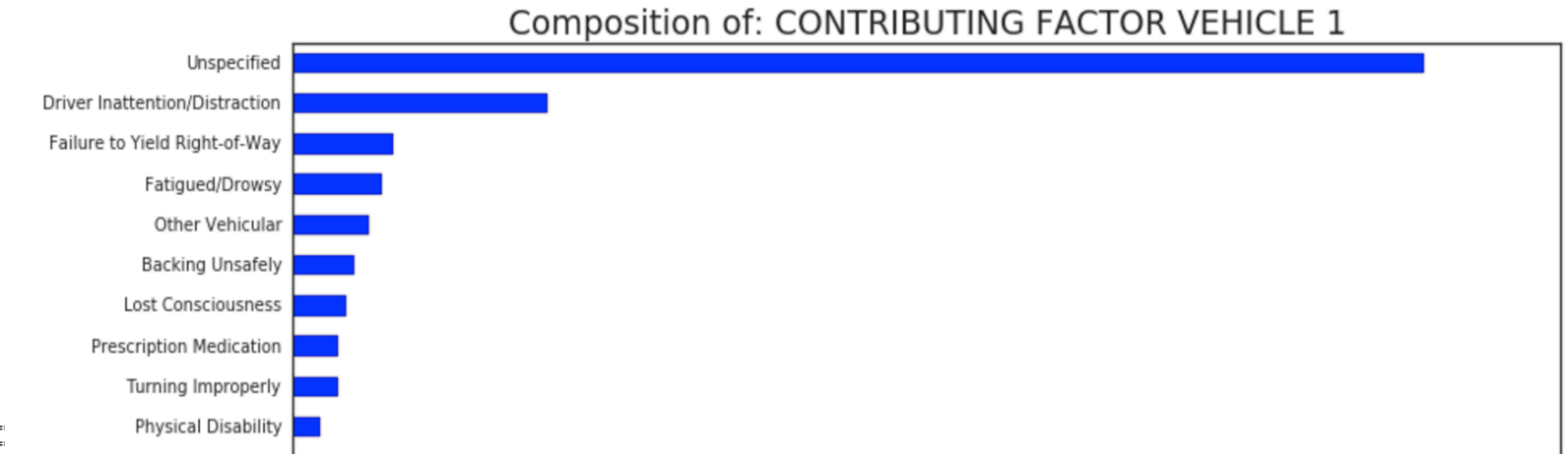




Investigating data attributes

- Plot the contributing factors of an accident

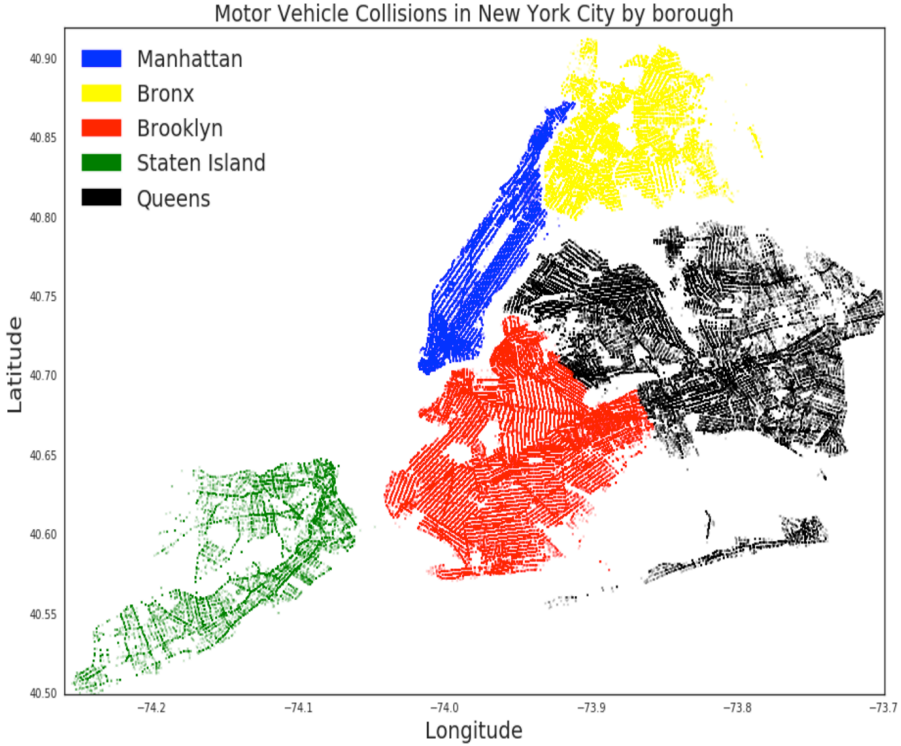
```
from pyspark.sql.functions import desc

collisions_out_df = collisions_out.toDF(sampleRatio=1)

factor = collisions_out_df.groupBy('CONTRIBUTING FACTOR VEHICLE 1').count().sort(desc('count')).toPandas()
factor = factor[0:24].sort_index(ascending=False)
factor.plot(kind='barh', legend=False, color='blue', figsize=(14,10))
plt.title('Composition of: ' + 'CONTRIBUTING FACTOR VEHICLE 1', size=20)
plt.xlabel('Count')
plt.yticks(range(len(factor))[::-1], factor['CONTRIBUTING FACTOR VEHICLE 1'][::-1])
plt.show()
```



Enhance the scatter plot with information about city boroughs

- Add information about the city boroughs and use a different color to depict borough on the scatter plot

```
manhattan = collisions_pd[collisions_pd['Borough']=='MANHATTAN']
bronx = collisions_pd[collisions_pd['Borough']=='BRONX']
brooklyn = collisions_pd[collisions_pd['Borough']=='BROOKLYN']
staten = collisions_pd[collisions_pd['Borough']=='STATEN ISLAND']
queens = collisions_pd[collisions_pd['Borough']=='QUEENS']

plt.figure(figsize=(15,10), dpi=0.1)

#create scatterplots
plt.scatter(manhattan.Longitude, manhattan.Latitude, s=1, color='blue', marker ='.')
plt.scatter(bronx.Longitude, bronx.Latitude, s=1, color='yellow', marker ='.')
plt.scatter(brooklyn.Longitude, brooklyn.Latitude, color='red', s=1, marker ='.')
plt.scatter(staten.Longitude, staten.Latitude, s=1, color='green', marker ='.')
plt.scatter(queens.Longitude, queens.Latitude, s=1, color='black', marker ='.')

#create legend
blue_patch = mpatches.Patch(label='Manhattan', color='blue')
yellow_patch = mpatches.Patch(color='yellow', label='Bronx')
red_patch = mpatches.Patch(color='red', label='Brooklyn')
green_patch = mpatches.Patch(color='green', label='Staten Island')
black_patch = mpatches.Patch(color='black', label='Queens')
plt.legend([blue_patch, yellow_patch, red_patch, green_patch, black_patch],
           ('Manhattan', 'Bronx', 'Brooklyn', 'Staten Island', 'Queens'),
           loc='upper left', prop={'size':20})

#adjust more settings
plt.title('Motor Vehicle Collisions in New York City by borough', size=20)
plt.xlim((-74.26,-73.7))
plt.ylim((40.5,40.92))
plt.xlabel('Longitude',size=20)
plt.ylabel('Latitude',size=20)
plt.show()
```



IBM

Adjust the scatter plot with severity of collisions

- Adjust the scatter plot settings to use color codes to indicate collisions resulting in car body damage, personal injury, and fatal accidents

```
#adjust settings
plt.figure(figsize=(15,10))

#create scatterplots
plt.scatter(nothing_pd.Longitude, nothing_pd.Latitude, alpha=0.04, s=1, color='blue')
plt.scatter(injured_pd.Longitude, injured_pd.Latitude, alpha=0.1, s=1, color='yellow')
plt.scatter(killed_pd.Longitude, killed_pd.Latitude, color='red', s=5)

#create legend
blue_patch = mpatches.Patch( label='car body damage', alpha=0.2, color='blue')
yellow_patch = mpatches.Patch(color='yellow', label='personal injury', alpha=0.5)
red_patch = mpatches.Patch(color='red', label='lethal accidents')
plt.legend([blue_patch, yellow_patch, red_patch],('car body damage', 'personal injury', 'fa
           loc='upper left', prop={'size':20})

#adjust more settings
plt.title('Severity of Motor Vehicle Collisions in New York City', size=20)
plt.xlim((-74.26,-73.7))
plt.ylim((40.5,40.92))
plt.xlabel('Longitude',size=20)
plt.ylabel('Latitude',size=20)
plt.savefig('anothertry.png')

plt.show()
```