

RED HAT
SUMMIT

Data Analytics using Mobile, 3Scale and radanalytics.io

May 10th, 2018

Putting it all together in A Bike Sharing Story

Juana Nakfour
Senior Technical Account Manager

Vinay Bhalerao
Senior Solution Architect

Michael McCune
Principal Software Engineer



Who are we ?



*Juana Nakfour
Senior Technical Account Manager
Software Engineer
Red Hat Inc.*

Always Inventing



*Vinay Bhalerao
Senior Solution Architect
Red Hat Inc.*

I love APIs and Nutella!



*Michael McCune
Principal Software Engineer
Red Hat Inc.*

*Data processing on OpenShift
and OpenStack*

*I have a curiously strong love
of the Grateful Dead*

Agenda

Story of Renting Bikes

Putting it All Together- Architecture

Our API gateway using 3Scale

Data Analysis and Machine Learning using radanalytics.io

Actually Renting Bikes in a Demo

Thoughts

Story of Renting Bikes - Last Mile



"Dockless"



Where are the bikes?

Analytics on 3rd party app providers using our API?

Where should we stock bikes?

What are the bike renting location trends?

How do we integrate backend network components?

How are users using our mobile app?

How do we collect and organize our data?

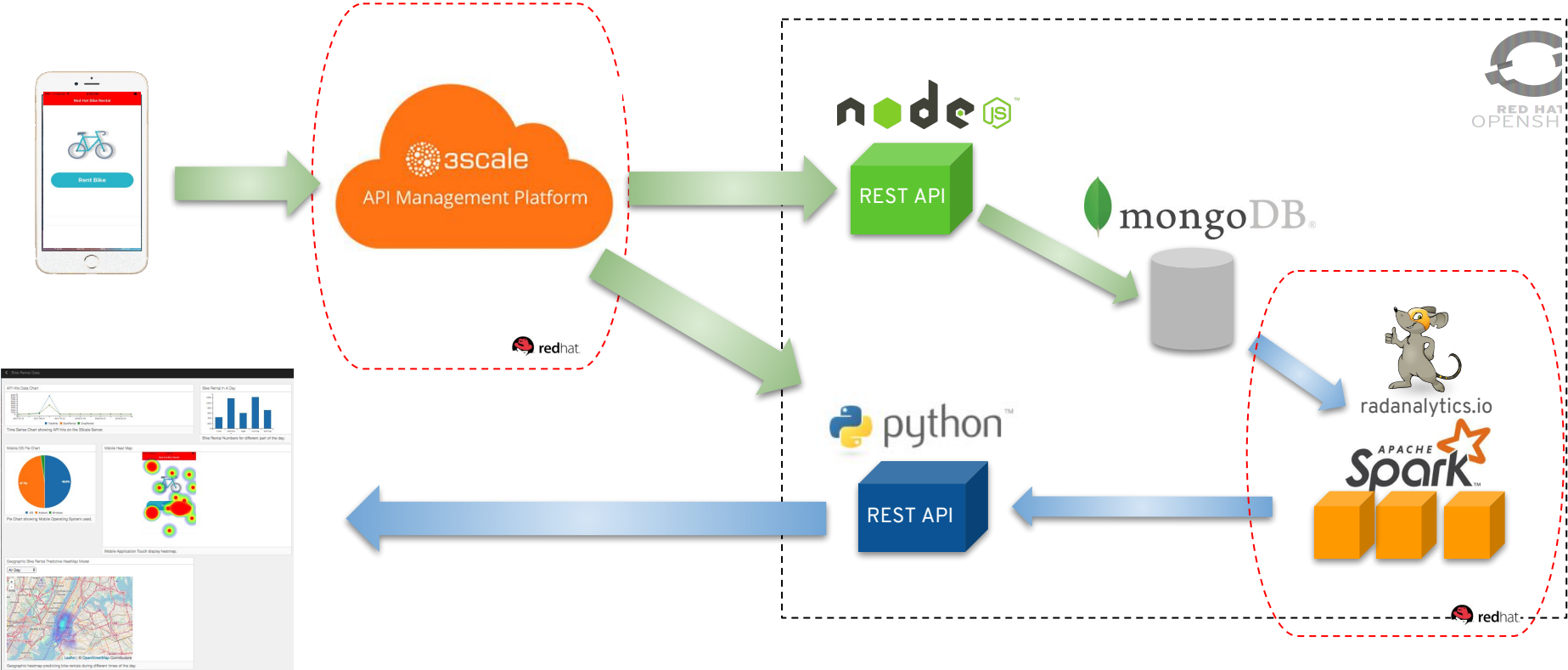
How do we get analytics data?

Which platform component can provide data?

Story of Renting Bikes - Last Mile



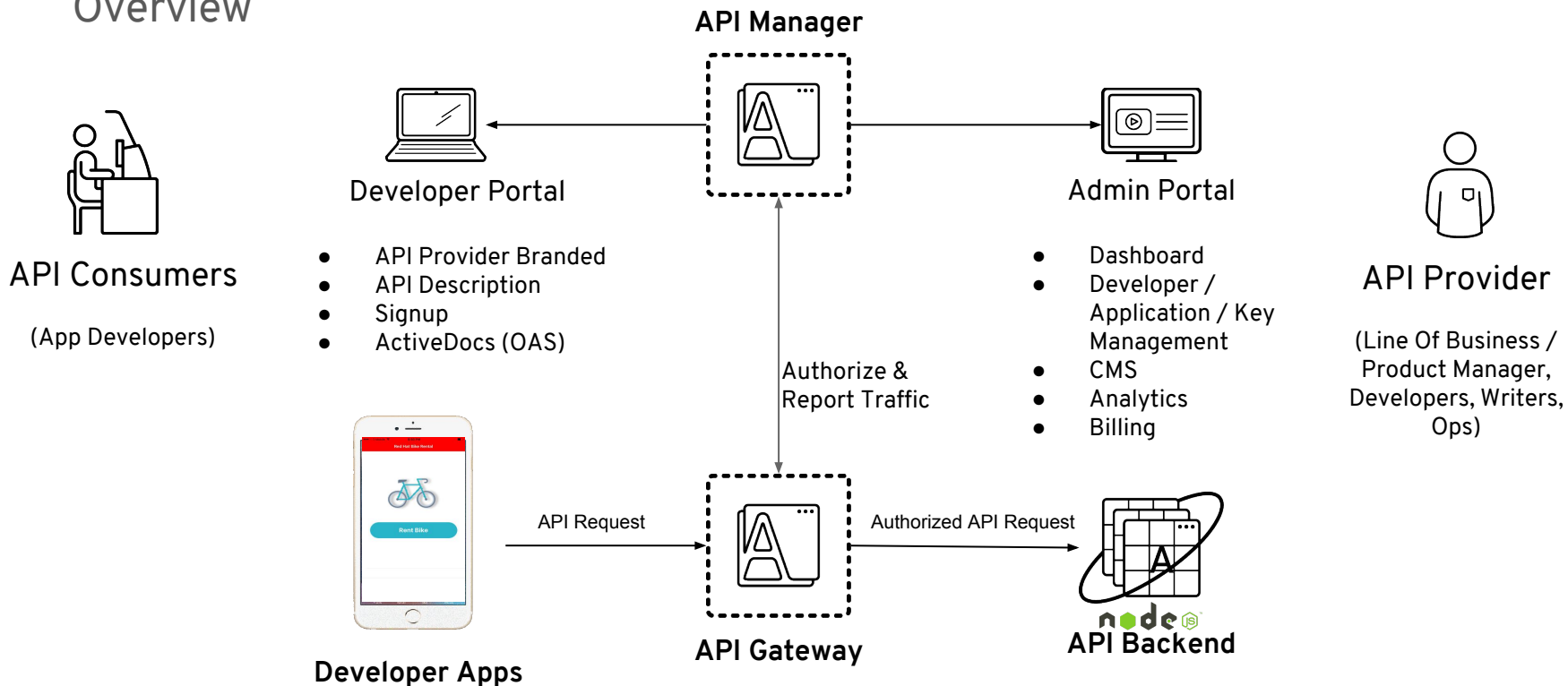
Putting it All Together Architecture



3Scale

Red Hat 3scale API Management

Overview



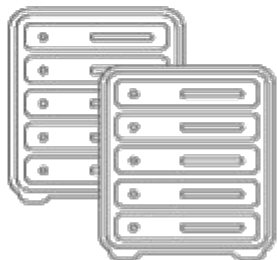
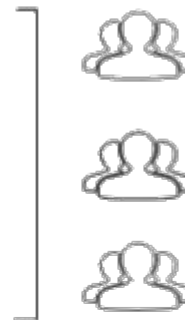
SECURITY & ACCESS CONTROL

Easy the consumption of APIs without losing control



Access control

How do you manage who gets access to your API? Can you establish different levels of access for different types of users? Can you control how different applications interact with your API? Access control features are essential to making sure you determine exactly who uses your API, how it is used and how much they can use it. We make it easy to centrally set up and manage policy and application plans for all your APIs on one platform



Security

It goes without saying that if you're planning to open an API, security needs to be carefully considered from the start. Whether your API is public, private or internal, with 3scale you can choose the authentication type most appropriate to your needs. We offer a range of authentication patterns and credentials to choose from, including unique API keys, and OAuth tokens.

YOUR API SECURITY

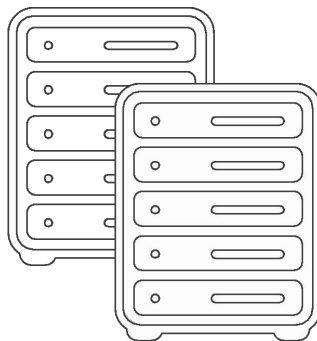
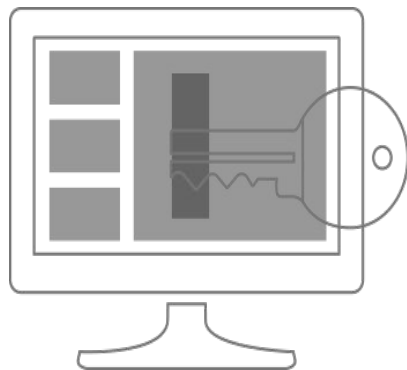
Authenticate and restrict access to your APIs. Protect backend services.

Multiple authentication mechanisms

- API Key

- App ID / App Key

- OpenID Connect



Authenticate traffic
Restrict by policy
Drop unwelcome calls
Protect backend services
Generate overage alerts
Impose rate limits

REPORTS & ANALYTICS

Track and monitor usage. Get reports by API, app, method and metric.



Gain and share API program insights.

Monitor and set alerts on traffic flow. Provide partners and developers with reports on their traffic with a user dashboard designed for them. Analyze your API traffic through detailed traffic analytics by account, application or service and share performance insights across the organization with crisp clear reporting.



High-level data at your fingertips

The Dashboard part of the Admin Portal gives you quick, centrally located visibility into any traffic and customer engagement opportunities or issues with your APIs. It is available now on all 3scale API Management plans from free through enterprise.

API GATEWAY

Gateway Layer Policy Enforcer

The API Gateway is responsible for enforcing the API policies that are defined in the API Manager Admin Portal.

The API Gateway consults with the API Manager on incoming calls, and enforces the policies, either returning an error or proxying the API call to the customer's API backend.





Our API gateway using 3-Scale- Demo

radanalytics.io



What is radanalytics.io?

An open source community working to empower intelligent application lifecycles on OpenShift

A collection of projects to enable analytics and machine learning frameworks on OpenShift

Basic architecture

Your Application



Apache Spark

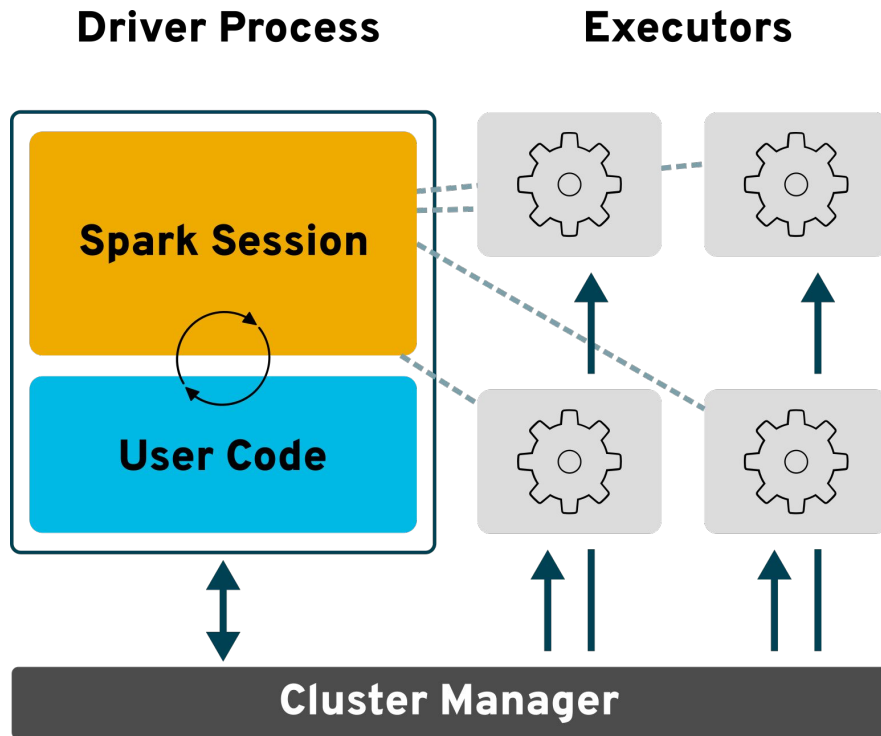


radanalytics.io



OpenShift

Apache Spark

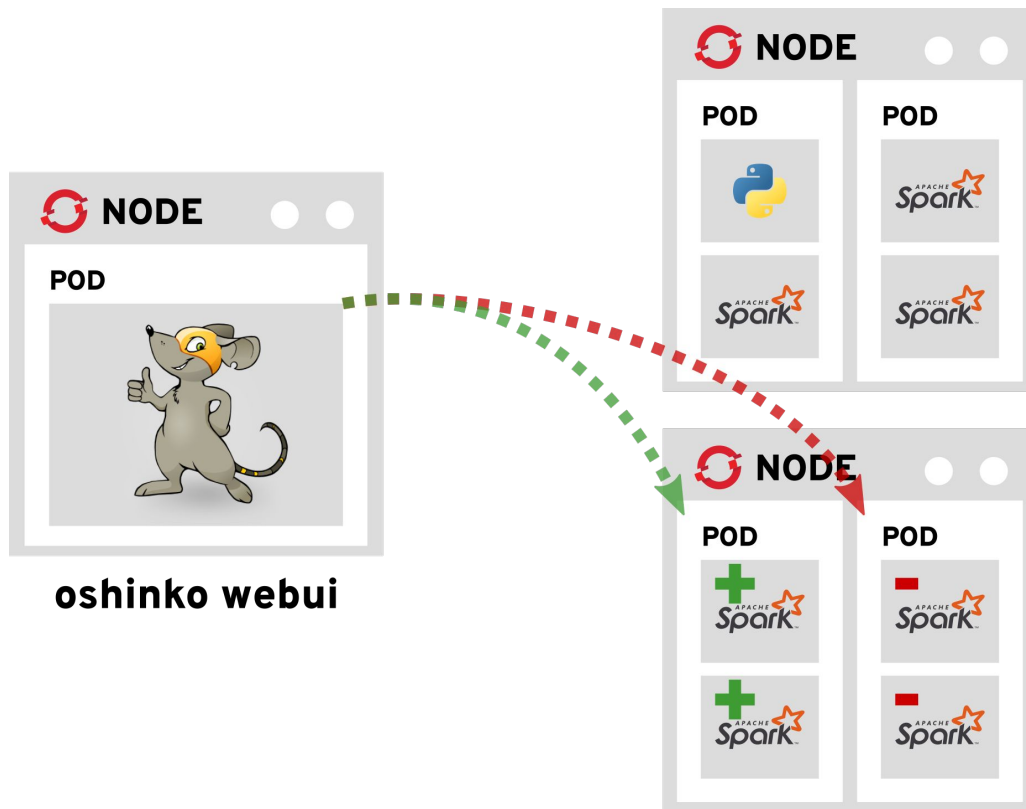


Project Oshinko


Deploy and manage Apache Spark clusters on OpenShift through browser and command line tooling

Utilize source-to-image based repositories to automatically deploy ephemeral Apache Spark clusters alongside your applications

Oshinko WebUI




Oshinko WebUI

← → ↻ 

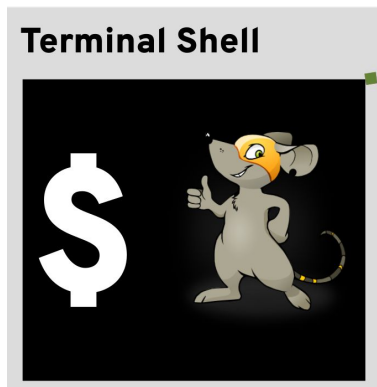
Oshinko Spark Cluster Management

Clusters

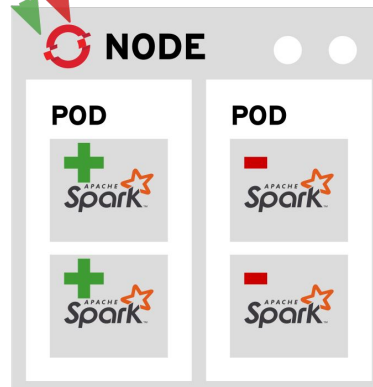
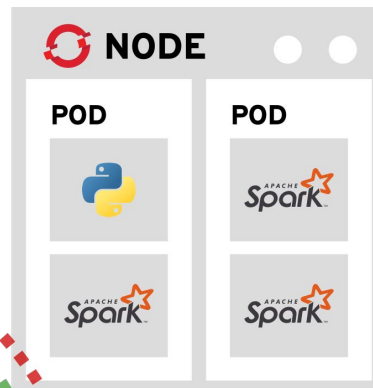
Spark Clusters [Deploy](#)

Name	Status	Master	Worker count	
mycluster	Running	spark://mycluster:7077	1	Scale 

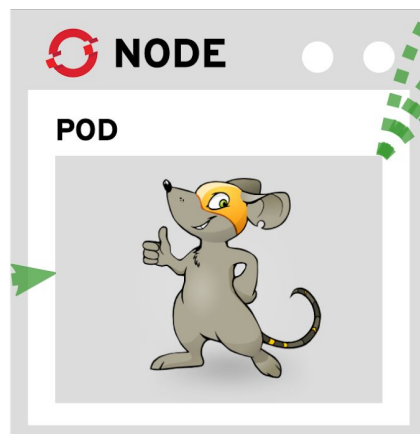
Oshinko Command Line



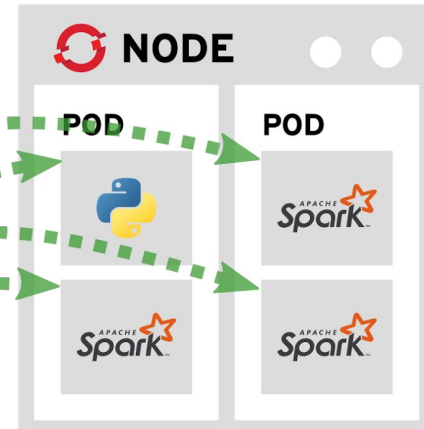
oshinko cli



Oshinko Source-to-Image



oshinko source-to-image



Source-to-Image language support

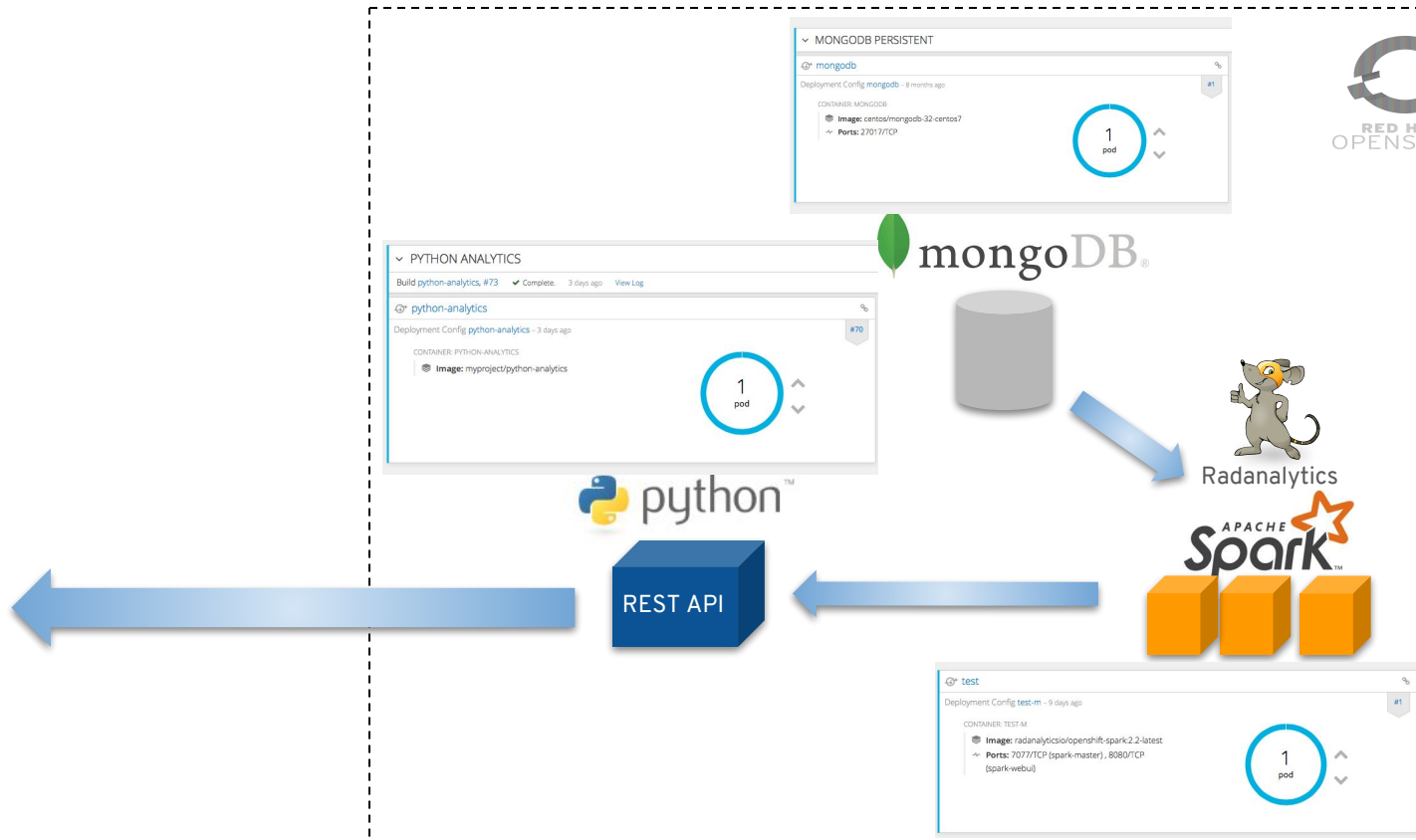
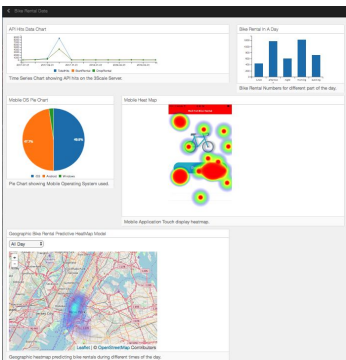


Learn more!



radanalytics.io

Data Analysis and Machine Learning using radanalytics.io - Demo



Data Analysis and Machine Learning using radanalytics.io - Demo

Creating Pod

```
oc new-app --template=oshinko-pyspark-build-dc  
-p APPLICATION_NAME=python-analytics  
-p GIT_URI=https://github.com/nakfour/mobile-analytics.git  
-p SPARK_OPTIONS='--packages org.mongodb.spark:mongo-spark-connector_2.11:2.2.0'  
OSHINKO_CLUSTER_NAME=test
```

Deployment Config

```
containers:  
- name: python-analytics  
  image: >-  
    172.30.1.1:5000/myproject/python-analytics@sha256:b0006cf0c2f4b0d3344d81f535t  
  env:  
    - name: OSHINKO_CLUSTER_NAME  
      value: test  
    - name: APP_ARGS  
    - name: SPARK_OPTIONS  
      value: '--packages org.mongodb.spark:mongo-spark-connector_2.11:2.2.0'
```

Data Analysis and Machine Learning using radanalytics.io - Demo



Creating Spark Session

```
spark = SparkSession.builder.appName("mobileanalytics").config("spark.mongodb.input.uri",  
"mongodb://admin:<pass>@mongodb/sampled.bikerental").config("spark.mongodb.output.uri",  
"mongodb://admin:<pass>@mongodb/sampled.bikerental").getOrCreate()
```

Data Analysis and Machine Learning using radanalytics.io - Demo



Creating DataFrames

```
bikerentaldf = spark.read.format("com.mongodb.spark.sql.DefaultSource").load()
```

Data Analysis and Machine Learning using radanalytics.io - Demo



Creating Linear Regression Model

```
assembler=VectorAssembler(inputCols=['startstationid', 'daypartInt', 'startstationlat', 'startstationlon'],
outputCol="features")
output = assembler.transform(bikrentaldf)
lr = LinearRegression(maxIter=10, regParam=0.3,
elasticNetParam=0.8,featuresCol=assembler.getOutputCol(),labelCol="rentalcount")
pipelineLG= Pipeline(stages=[assembler,lr])
modelLR = pipelineLG.fit(bikerentaldf)
```

Data Analysis and Machine Learning using radanalytics.io - Demo



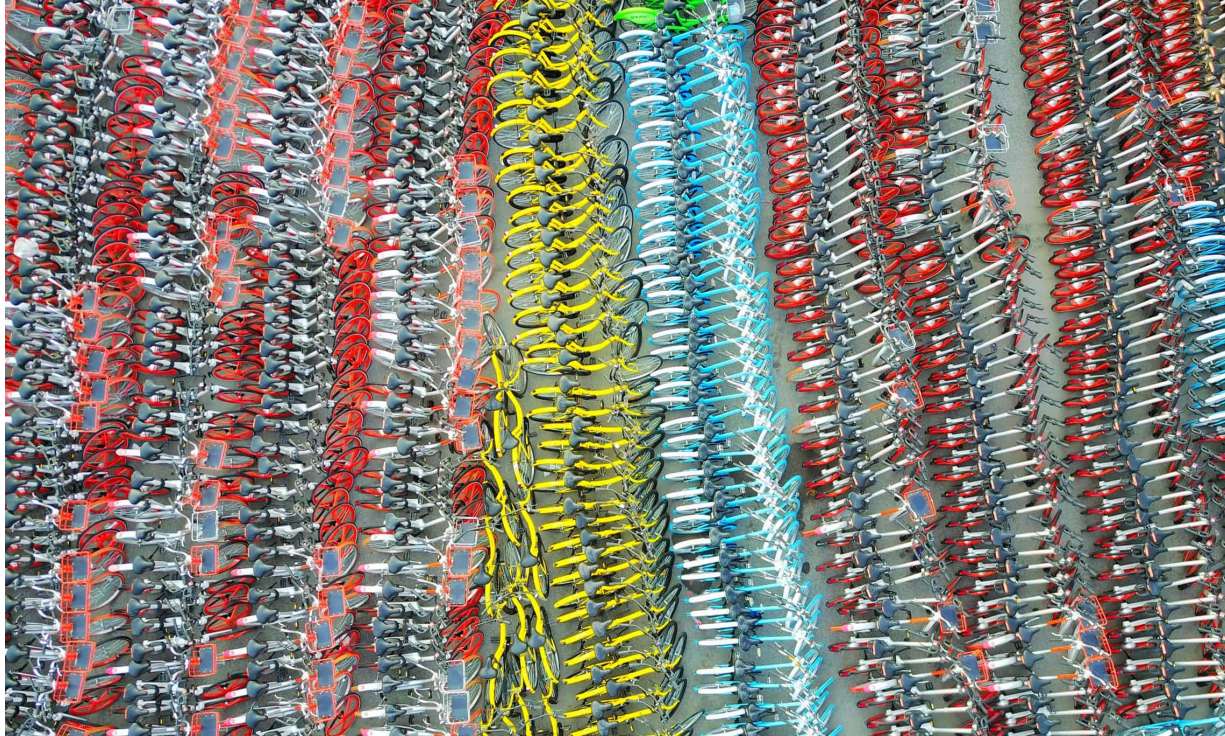
Applying model for Predictions

```
predictions = modelLR.transform(testbikerentaldf)
```

Demo

Actually Renting Bikes in a Demo

Lessons Learned



References



- radanalytics.io
- 3scale.net
- openshift.com

Data Source

- <https://www.citibikenyc.com/system-data>

Images

- <https://rampages.us/pedal2play/2016/10/13/post-4/>
- <https://timesofsandiego.com/business/2018/03/03/dockless-bicycles-pop-up-downtown-offering-transportation-flexibility/>
- <https://blog.producthunt.com/we-tried-every-shared-bike-and-scooter-in-san-francisco-bb766abd0a96>
- <https://mashable.com/2017/01/18/bike-sharing-pile-up-china/#whSoCvTV1PqA>
- <https://www.theguardian.com/cities/2017/mar/22/bike-wars-dockless-china-millions-bicycles-hangzhou>

RED HAT
SUMMIT

THANK YOU



plus.google.com/+RedHat



facebook.com/redhatinc



linkedin.com/company/red-hat



twitter.com/RedHat



youtube.com/user/RedHatVideos