http://AppAgile.de
@appagile

be fast, be agile, At Scale

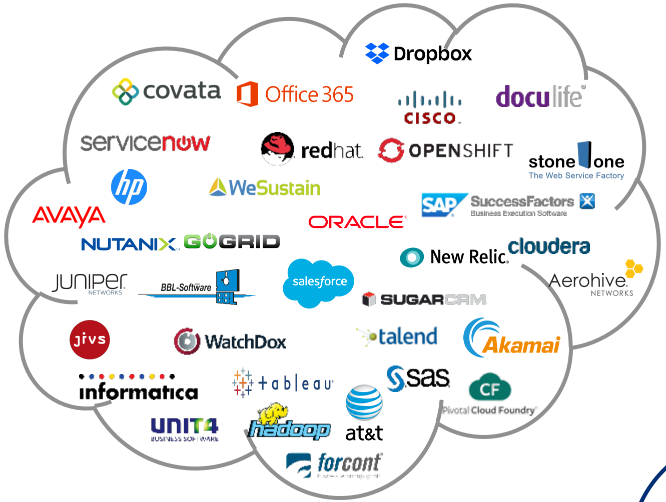
Appagile Data Science Workstation - Analytics from the cloud

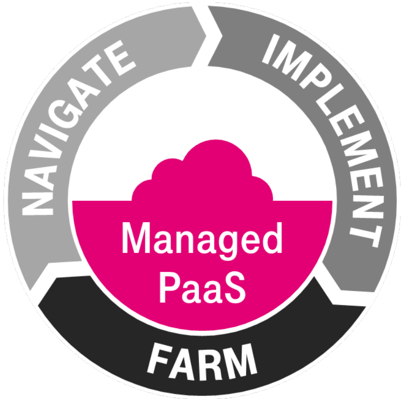
Stefan Zosel

May, 2018 – Openshift Anwenderforum

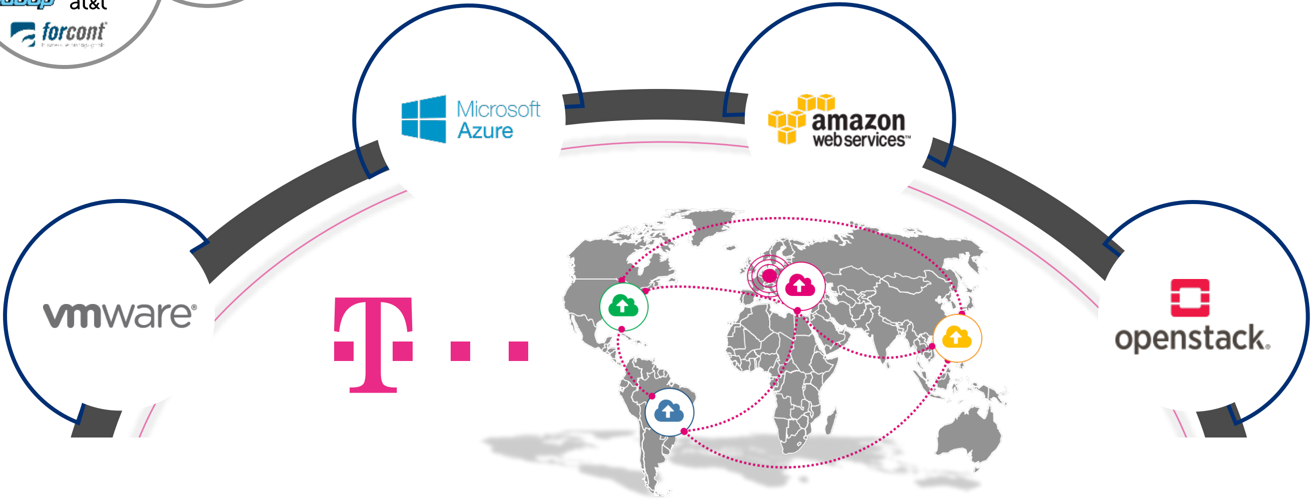
T · · Systems·

Digitization with T-Systems





- Managed Cloud
- Services
- Security approved operation
- Openshift for Enterprises



Warum DataScience auf Openshift?

Innovationen aus der Cloud

Neue UseCases

AI/ML

BIG Data trifft PaaS

Neue Methoden

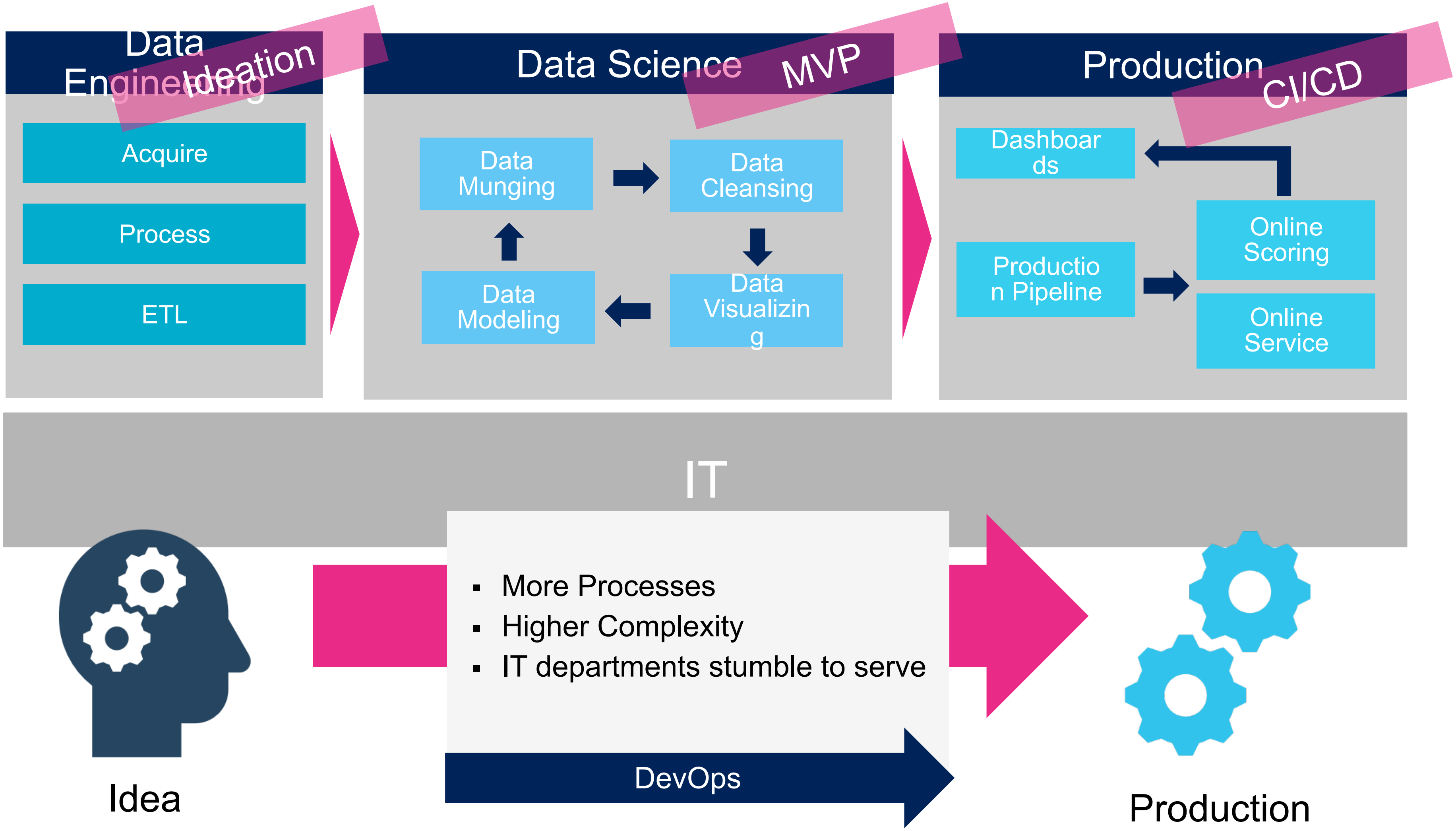
BIG Data trifft Cloud

Viele neue SW & Services

Self Service Bedarf

Data Science - DevOps

From data to product



- More Processes
- Higher Complexity
- IT departments stumble to serve

Data Science DevOps – not Admin!

The pain points

Sound Similar to Developer

Setup

- Provision of environments takes forever
- Compatibility problems

- Provision of data access takes time
- Connection to data sources is cumbersome

Connection

Scalability

- Hardware resources on desktops limited
- Models can only be trained and tested on samples
- Popular frameworks are not parallelized.

- Language and version conflicts
- Sharing results takes extra effort

Sharing

Data Science DEVOPS
What else Would be nice?

Dockerization in the cloud

Sandboxing
Try new ideas **without destroying** anything

PoC
Develop your **PoC** agile, **at fast pace** in the cloud

Training
Train your team in the cloud on **new ML frameworks**

Locality
Run **analyses** in the cloud where your **data is located**

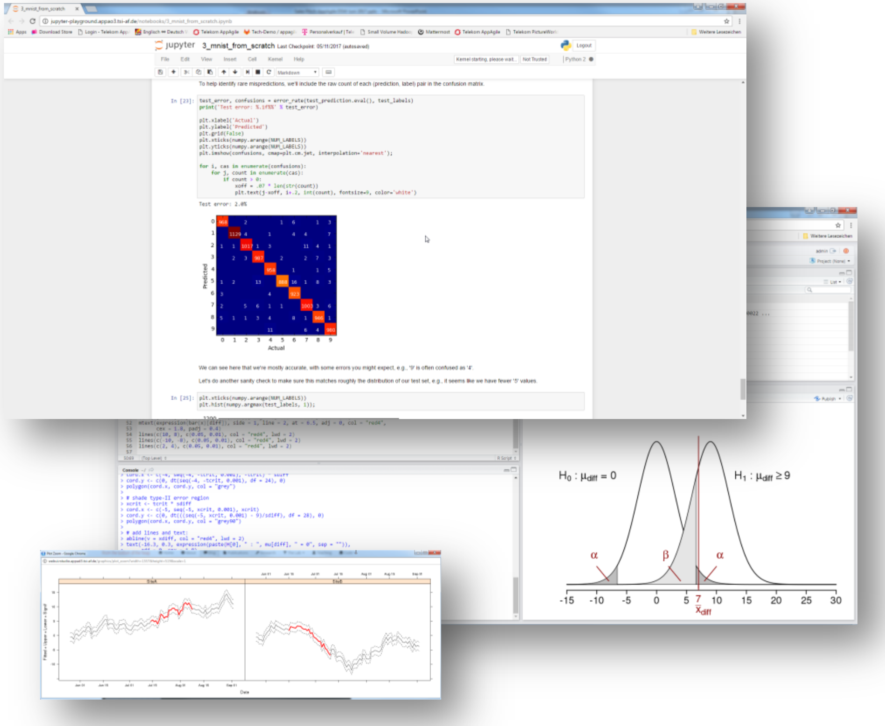
Ad-hoc Analysis
Set up quickly an environment **with all tools**

CI/CD
Work in **parallel** and **contiuously** on project **modules**

Presenting Appagile‘s Data science Workstation

be fast, be agile, at scale

- Quick and easy self-service
- **Dedicated** Data Science Engineer
- Working with **AppAgile‘s Small Volume Hadoop** cluster
- R, Python and Scala in multiple versions
- Common machine learning **frameworks**
- **Visualization** via notebooks, markdown and R Shiny Server
- Available on **OTC**, **vCloud**, and **Microsoft Azure**
- From sample to big data



To help identify rare mispredictions, we'll include the raw count of each (prediction, label) pair in the confusion matrix.

In [23]:
```
test_error, confusions = error_rate(test_prediction.eval(), test_labels)
print('Test error: %.1f%%' % test_error)

plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.grid(False)
plt.xticks(numpy.arange(NUM_LABELS))
plt.yticks(numpy.arange(NUM_LABELS))
plt.imshow(confusions, cmap=plt.cm.jet, interpolation='nearest');

for i, cas in enumerate(confusions):
    for j, count in enumerate(cas):
        if count > 0:
            xoff = .07 * len(str(count))
            plt.text(j-xoff, i+.2, int(count), fontsize=9, color='white')
```

```
Test error: 2.0%
```

We can see here that we're mostly accurate, with some errors you might expect, e.g., '9' is often confused as '4'.

Let's do another sanity check to make sure this matches roughly the distribution of our test set, e.g., it seems like we have fewer '5' values.

In [25]:
```
plt.xticks(numpy.arange(NUM_LABELS))
plt.hist(numpy.argmax(test_labels, 1));
```

```
mtext(expression(bar(x)[diff]), side = 1, line = 2, at = 6.5, adj = 0, col = "red4",
      cex = 1.8, padj = 0.4)
lines(c(10, 8), c(0.05, 0.01), col = "red4", lwd = 2)
lines(c(-10, -8), c(0.05, 0.01), col = "red4", lwd = 2)
lines(c(2, 4), c(0.05, 0.01), col = "red4", lwd = 2)
```

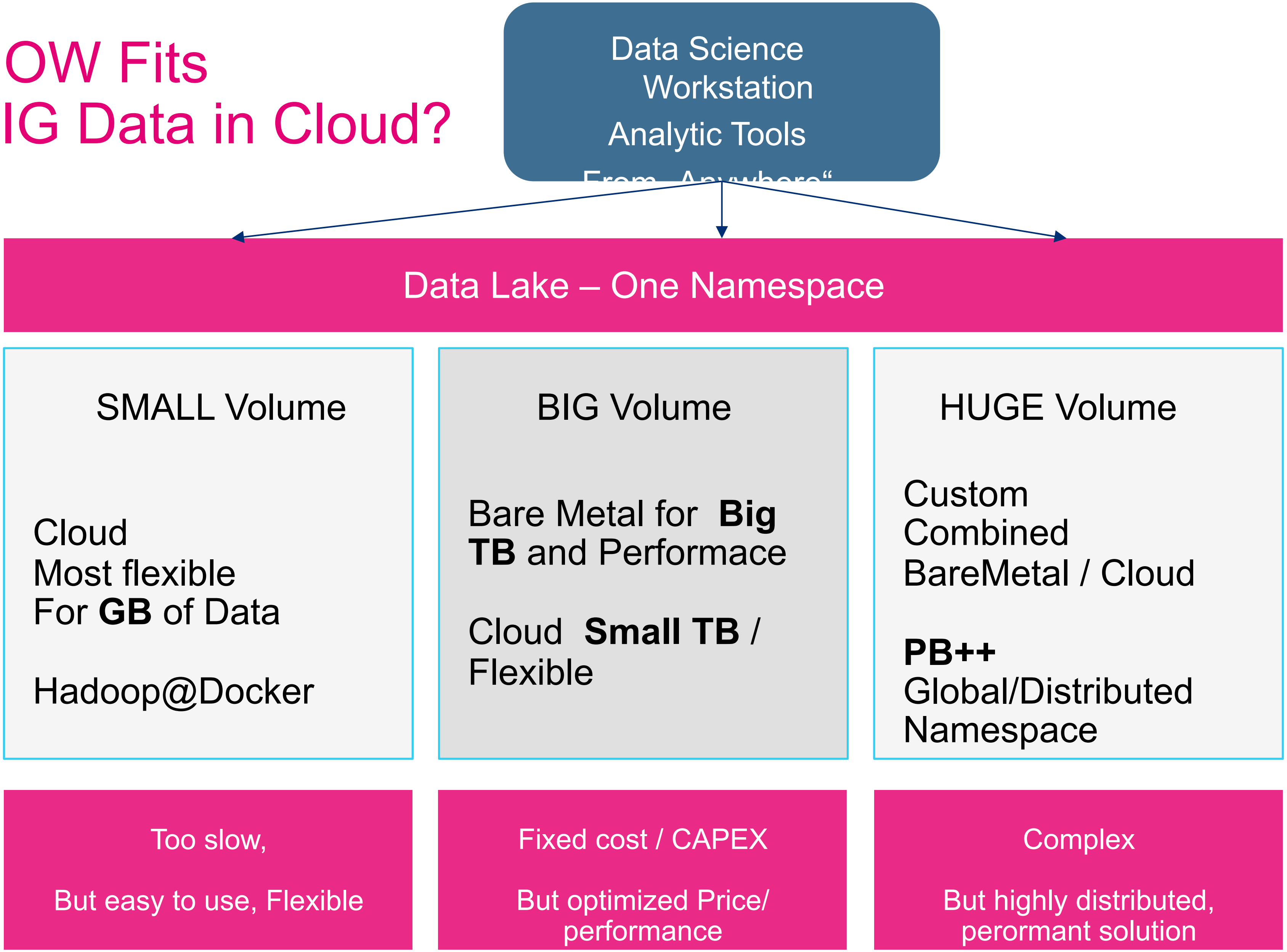
```
> cord.y <- c(0, dt(seq(-4, -tcrit, 0.001), df = 24), 0)
> polygon(cord.x, cord.y, col = "grey")
>
> # shade type-II error region
> xcrit <- tcrit * sdiff
> cord.x <- c(-5, seq(-5, xcrit, 0.001), xcrit)
> cord.y <- c(0, dt(((seq(-5, xcrit, 0.001) - 9)/sdiff), df = 28), 0)
> polygon(cord.x, cord.y, col = "grey90")
>
> # add lines and text:
> abline(v = xdiff, col = "red4", lwd = 2)
> text(-16.3, 0.3, expression(paste(H[0], " : ", mu[diff], " = 0", sep = "")),
```

RED HAT OPENSHIFT

BIG DATA APP AGILE PAAS DEVOPS hadoop

R	python	Spark	Scala
hadoop	jupyter	RStudio	Shiny by RStudio

HOW Fits
BIG Data in Cloud?



Data Science Workstation

Analytic Tools

From „Anywhere“

Data Lake – One Namespace

SMALL Volume

Cloud
Most flexible
For **GB** of Data

Hadoop@Docker

BIG Volume

Bare Metal for **Big TB** and Performace

Cloud **Small TB** / Flexible

HUGE Volume

Custom Combined BareMetal / Cloud

PB++
Global/Distributed Namespace

Too slow,

But easy to use, Flexible

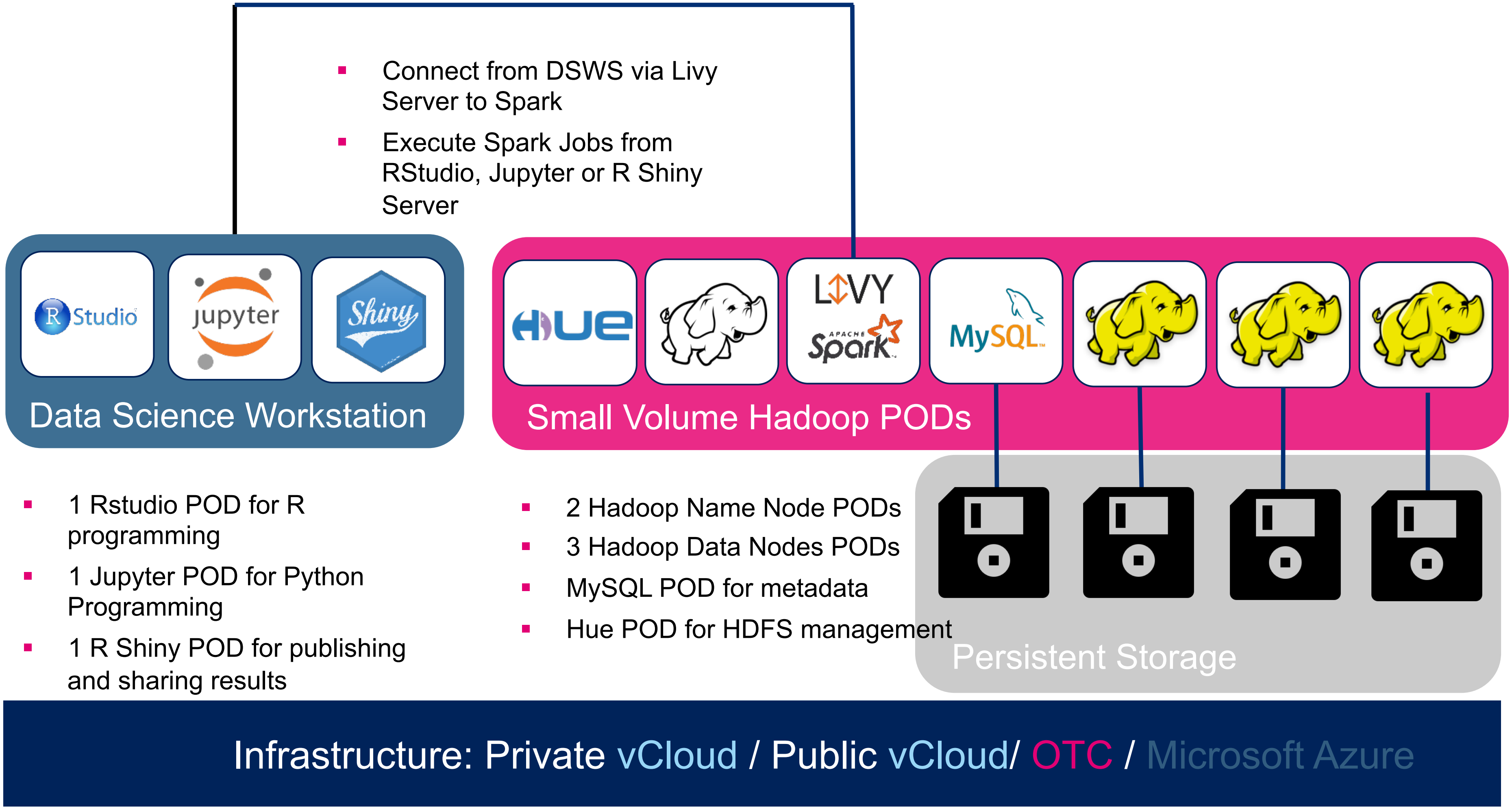
Fixed cost / CAPEX

But optimized Price/ performance

Complex

But highly distributed, perormant solution

Appagile Small volume hadoop
data Science Setup Example



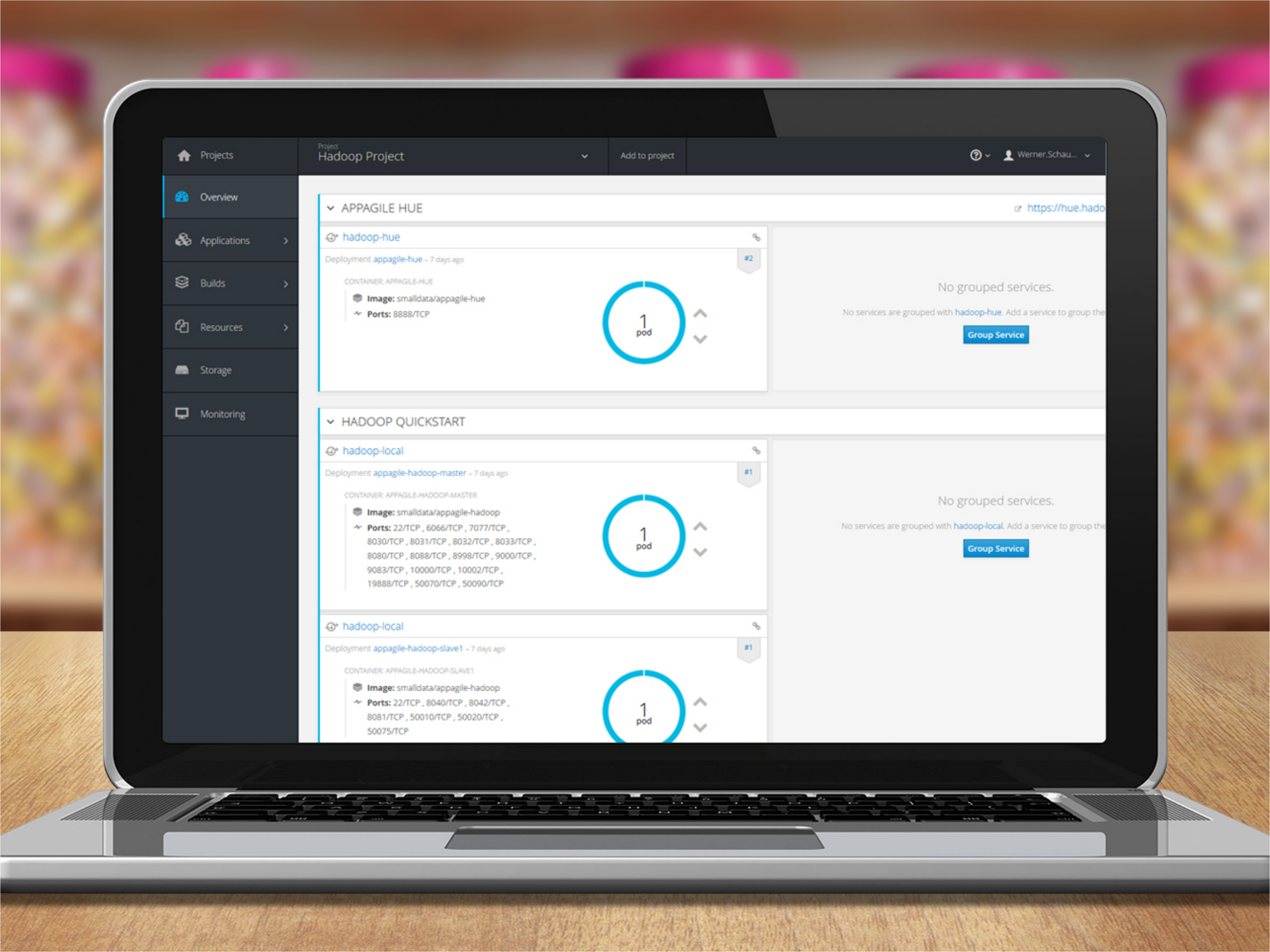
- Connect from DSWS via Livy Server to Spark
- Execute Spark Jobs from RStudio, Jupyter or R Shiny Server

- 1 Rstudio POD for R programming
- 1 Jupyter POD for Python Programming
- 1 R Shiny POD for publishing and sharing results

- 2 Hadoop Name Node PODs
- 3 Hadoop Data Nodes PODs
- MySQL POD for metadata
- Hue POD for HDFS management

Infrastructure: Private vCloud / Public vCloud/ OTC / Microsoft Azure

Some Screens



APPAGILE HUE

https://hue.hado

hadoop-hue

Deployment appagile-hue - 7 days ago

#2

CONTAINER: APPAGILE-HUE

Image: smalldata/appagile-hue

Ports: 8888/TCP

1 pod

No grouped services.

No services are grouped with hadoop-hue. Add a service to group the

HADOOP QUICKSTART

hadoop-local

Deployment appagile-hadoop-master - 7 days ago

#1

CONTAINER: APPAGILE-HADOOP-MASTER

Image: smalldata/appagile-hadoop

Ports: 22/TCP , 6066/TCP , 7077/TCP , 8030/TCP , 8031/TCP , 8032/TCP , 8033/TCP , 8080/TCP , 8088/TCP , 8998/TCP , 9000/TCP , 9083/TCP , 10000/TCP , 10002/TCP , 19888/TCP , 50070/TCP , 50090/TCP

1 pod

No grouped services.

No services are grouped with hadoop-local. Add a service to group the

hadoop-local

Deployment appagile-hadoop-slave1 - 7 days ago

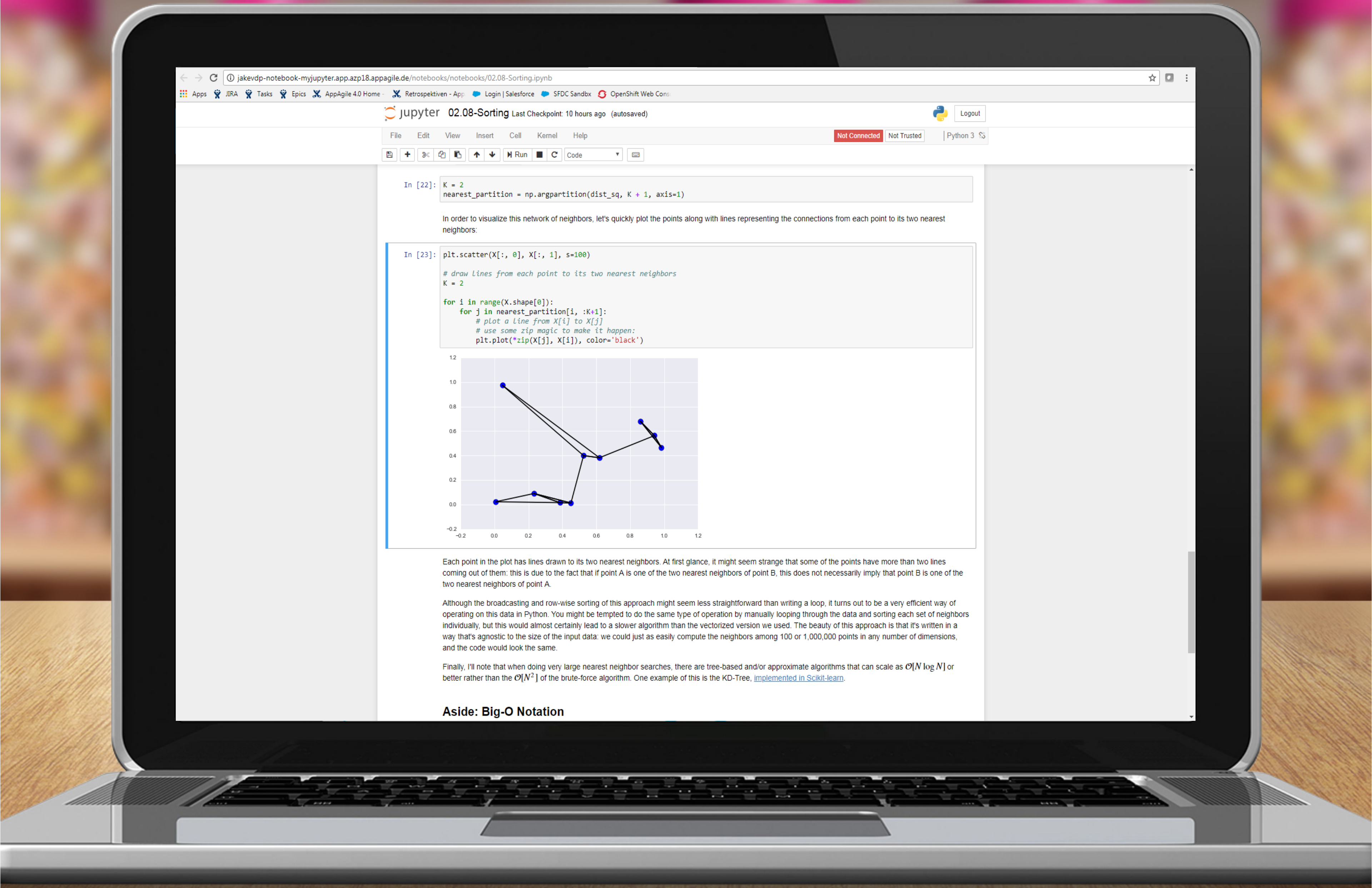
#1

CONTAINER: APPAGILE-HADOOP-SLAVE1

Image: smalldata/appagile-hadoop

Ports: 22/TCP , 8040/TCP , 8042/TCP , 8081/TCP , 50010/TCP , 50020/TCP , 50075/TCP

1 pod



```
In [22]: K = 2
         nearest_partition = np.argpartition(dist_sq, K + 1, axis=1)
```

In order to visualize this network of neighbors, let's quickly plot the points along with lines representing the connections from each point to its two nearest neighbors:

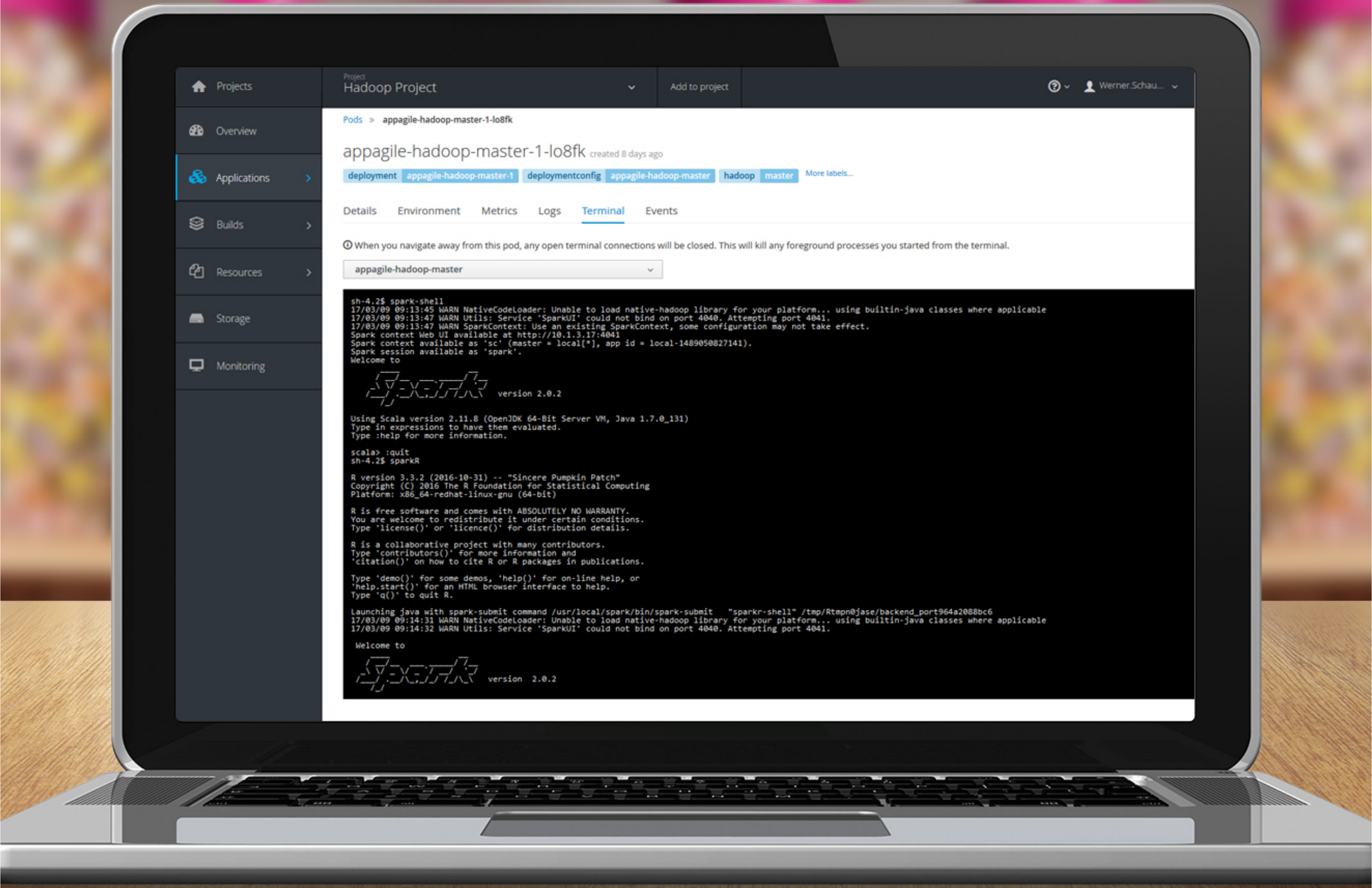
```
In [23]: plt.scatter(X[:, 0], X[:, 1], s=100)

         # draw lines from each point to its two nearest neighbors
         K = 2

         for i in range(X.shape[0]):
             for j in nearest_partition[i, :K+1]:
                 # plot a line from X[i] to X[j]
                 # use some zip magic to make it happen:
                 plt.plot(*zip(X[j], X[i]), color='black')
```

Each point in the plot has lines drawn to its two nearest neighbors. At first glance, it might seem strange that some of the points have more than two lines coming out of them: this is due to the fact that if point A is one of the two nearest neighbors of point B, this does not necessarily imply that point B is one of the two nearest neighbors of point A.

Although the broadcasting and row-wise sorting of this approach might seem less straightforward than writing a loop, it turns out to be a very efficient way of operating on this data in Python. You might be tempted to do the same type of operation by manually looping through the data and sorting each set of neighbors individually, but this would almost certainly lead to a slower algorithm than the vectorized version we used. The beauty of this approach is that it's written in a way that's agnostic to the size of the input data: we could just as easily compute the neighbors among 100 or 1,000,000 points in any number of dimensions, and the code would look the same.

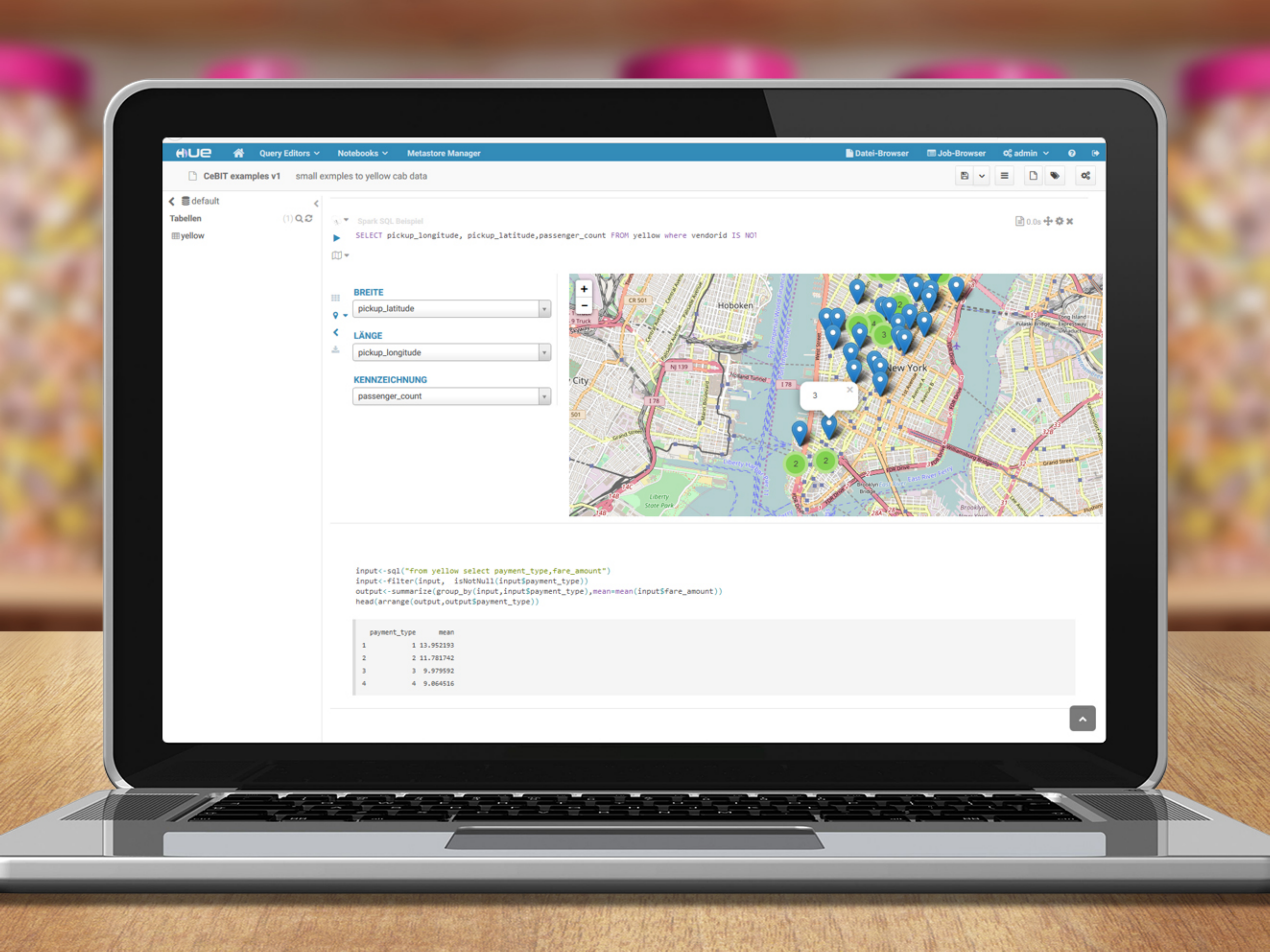
Finally, I'll note that when doing very large nearest neighbor searches, there are tree-based and/or approximate algorithms that can scale as $\mathcal{O}[N \log N]$ or better rather than the $\mathcal{O}[N^2]$ of the brute-force algorithm. One example of this is the KD-Tree, implemented in Scikit-learn.

Aside: Big-O Notation



ⓘ When you navigate away from this pod, any open terminal connections will be closed. This will kill any foreground processes you started from the terminal.

```
sh-4.2$ spark-shell
17/03/09 09:13:45 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
17/03/09 09:13:47 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
17/03/09 09:13:47 WARN SparkContext: Use an existing SparkContext, some configuration may not take effect.
Spark context Web UI available at http://10.1.3.17:4041
Spark context available as 'sc' (master = local[*], app id = local-1489050827141).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.0.2
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.7.0_131)
Type in expressions to have them evaluated.
Type :help for more information.

scala> :quit
sh-4.2$ sparkR

R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Launching java with spark-submit command /usr/local/spark/bin/spark-submit   "sparkr-shell" /tmp/Rtmpn0jase/backend_port964a2088bc6
17/03/09 09:14:31 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
17/03/09 09:14:32 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.

 Welcome to
    ____              __
   / __/__  ___ _____/ /__
  _\ \/ _ \/ _ `/ __/  '_/
 /___/ .__/\_,_/_/ /_/\_\   version  2.0.2
    /_/
```



CeBIT examples v1 small exmples to yellow cab data

Spark SQL Beispiel

```
SELECT pickup_longitude, pickup_latitude,passenger_count FROM yellow where vendorid IS NOT
```

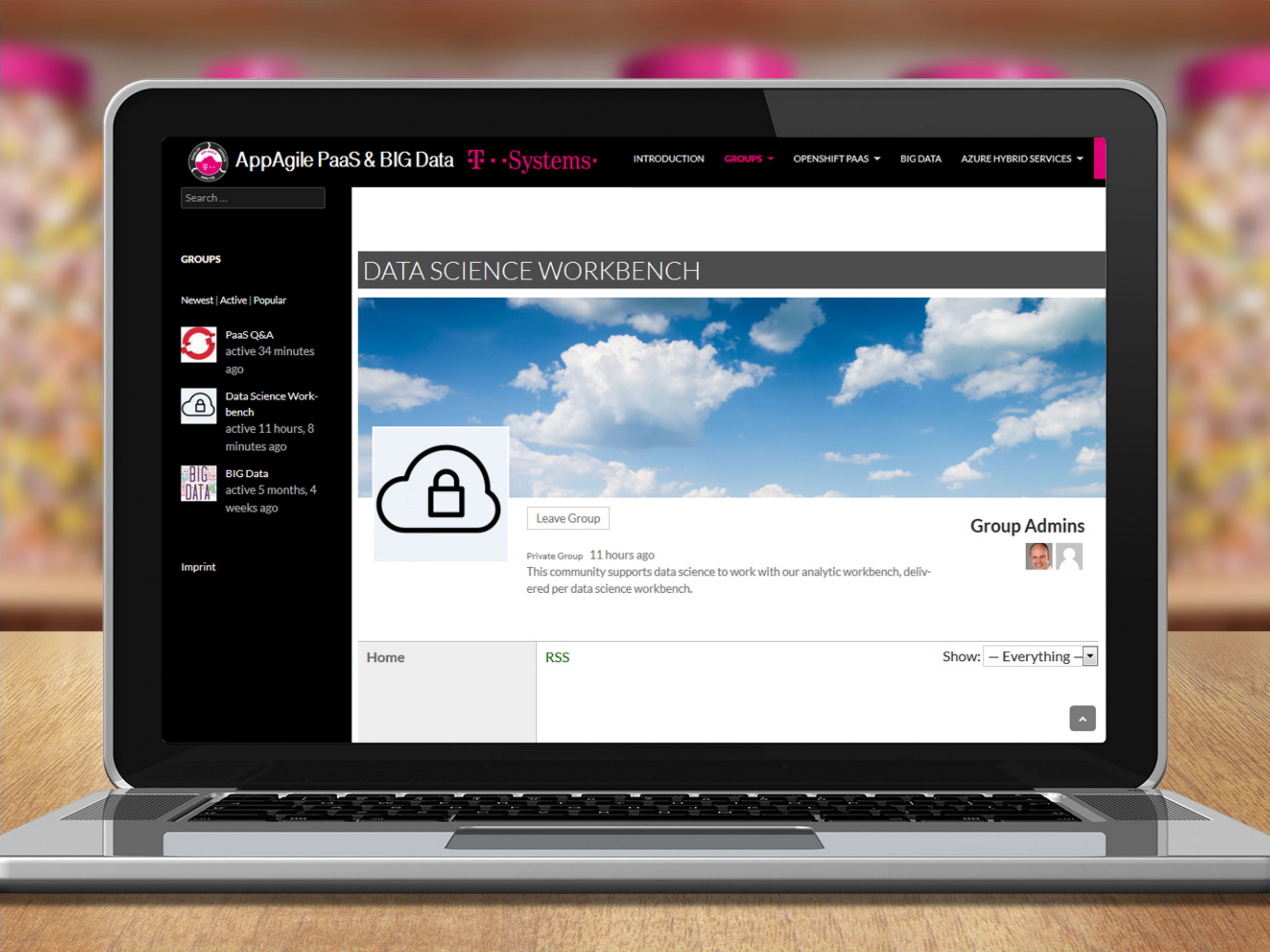
BREITE: pickup_latitude

LÄNGE: pickup_longitude

KENNZEICHNUNG: passenger_count

```
input<-sql("from yellow select payment_type,fare_amount")
input<-filter(input,  isNotNull(input$payment_type))
output<-summarize(group_by(input,input$payment_type),mean=mean(input$fare_amount))
head(arrange(output,output$payment_type))
```

```
  payment_type      mean
1            1 13.952193
2            2 11.781742
3            3  9.979592
4            4  9.064516
```



DATA SCIENCE WORKBENCH

PaaS Q&A
active 34 minutes ago

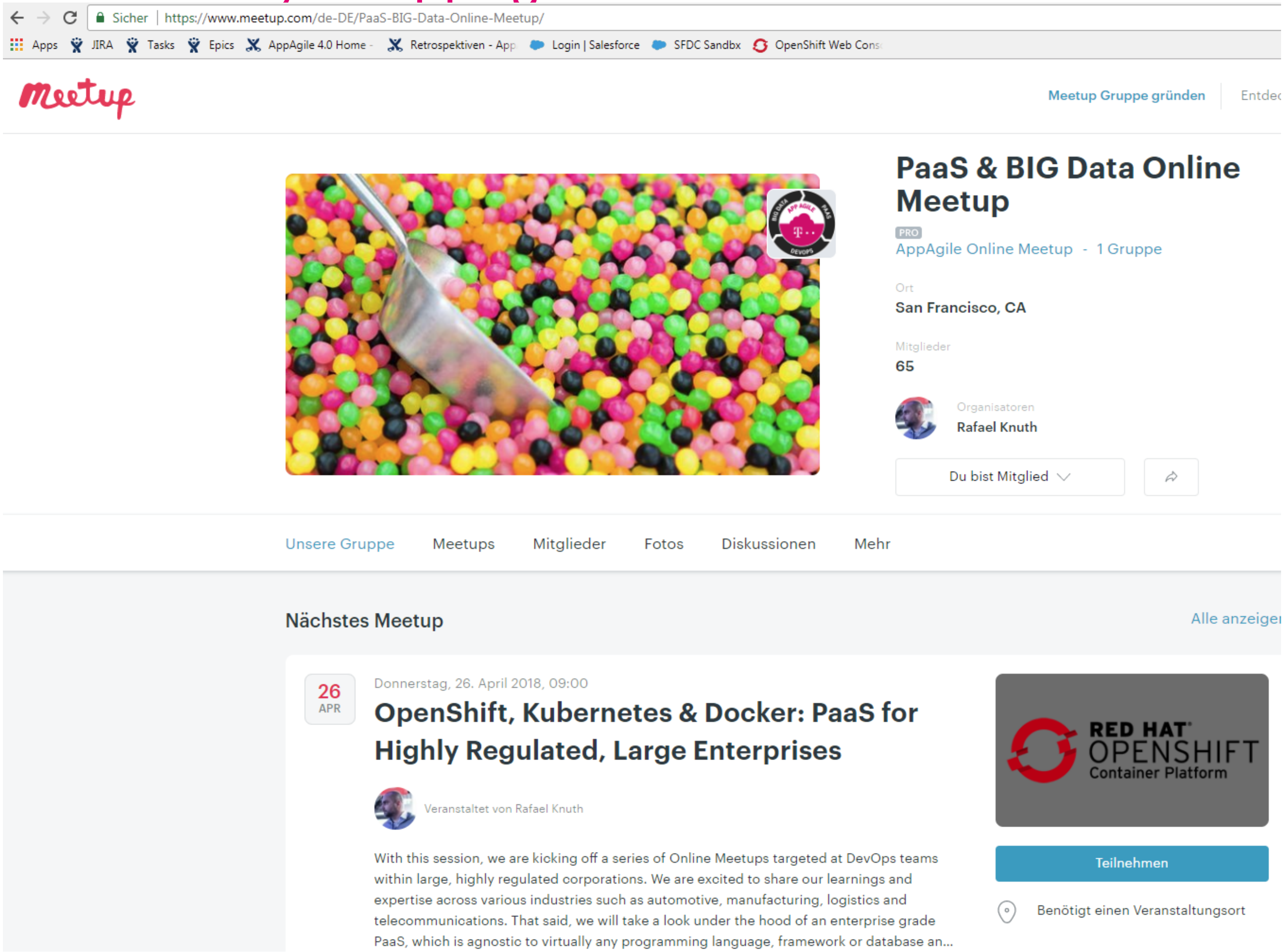
Data Science Workbench
active 11 hours, 8 minutes ago

BIG Data
active 5 months, 4 weeks ago

Private Group 11 hours ago

This community supports data science to work with our analytic workbench, delivered per data science workbench.

Community - AppAgile.IO



PaaS & BIG Data Online Meetup

PRO

AppAgile Online Meetup - 1 Gruppe

Ort
San Francisco, CA

Mitglieder
65

Organisatoren
Rafael Knuth

Nächstes Meetup

26
APR

Donnerstag, 26. April 2018, 09:00

OpenShift, Kubernetes & Docker: PaaS for Highly Regulated, Large Enterprises

Veranstaltet von Rafael Knuth

With this session, we are kicking off a series of Online Meetups targeted at DevOps teams within large, highly regulated corporations. We are excited to share our learnings and expertise across various industries such as automotive, manufacturing, logistics and telecommunications. That said, we will take a look under the hood of an enterprise grade PaaS, which is agnostic to virtually any programming language, framework or database an...

Benötigt einen Veranstaltungsort

Appagile data science workstation

What‘s in the Package:

Data Science Engineer

- For named users
- Dedicated to DS support
- Building of DS libraries
- In charge of DS community
- DS Q&A (Best Practise)

Analytics Services – Data Science Workstation

RStudio, Shiny, kafka, H2O, jupyter, TensorFlow, Apache Spark, Keras

Dedicated DS Engineer

- Ready-to-use workstation for DS
- DS self-service
- Predefined data and analytic modules

cassandra, NGINX, MySQL, Jenkins, PostgreSQL, MariaDB, mongoDB

Container Images

Data Science Community

Middleware - Small Volume Hadoop

OPENSHIFT

- Hadoop in dockers on OpenShift
- Horizontally scalable on demand
- Free assignment of capacities per node
- On public and private cloud
- Compatible to big data solution

HUE, Apache Spark, HIVE, hadoop

Container Repository

Data Science Workstation:

- For named user

Small Volume:

- Unmanaged - Free