AMIS|Conclusion SIG Workshop

Data Wrangling with Jupyter Notebooks

21st February 2019

Data Wrangling is a crucial stage in the data science workflow. Or in any workflow that starts from raw data and hopes to achieve business insight – and perhaps ready to run well trained machine learning models. Data wrangling encompasses various steps and activities -from gathering raw data, exploring, validating, filtering, filling in missing values, joining, enriching, aggregating, shape shifting, unifying. No clear break may be apparent between gathering, preparing, exploring, visualizing and modelling the data. These are iterative steps – with data wrangler moving back and forth in the process.

One of the most used data wrangling workbenches is the Jupyter Notebook – most commonly powered by a Python engine and leveraging the many libraries and frameworks that make the Python ecosystem such an attractive place for data professionals. In this workshop, you will get an introduction to Jupyter Notebook and Python – a platform for Data Analytics (data wrangling, visualization, reporting and machine learning). We will start with getting going with your very own Jupyter Notebook environment, running in a Docker container either locally on your laptop or on a cloud environment. Then you will get acquainted with a simple introduction notebook and subsequently work through a predefined notebook for a somewhat more complicated business case. Finally, you are on your own. We will give some suggestions for next steps – creating your own Notebook and finding answers to business questions on your own.

Of course, the time we have is limited and the scope is potentially vast. So we will barely scratch the surface of both the area of data wrangling as well as the use of Jupyter Notebooks and Python as workbench. However, you will probably get a good inkling of what they have to offer and hopefully get you enthusiastic and inspired to take it from here and move forward.

In this workshop:

* Get access to a Jupyter Notebook environment; you can choose between two options
  + Run Jupyter Notebook in a Docker container on your own laptop
  + Make use of the declarative, cloud based Katacoda playground that does not require any local installation (in fact, only requires you to have a browser and internet access)
* Get acquainted with Jupyter Notebook in an introductory (“Hello World” style) notebook
* Explore the case of the Oracle OpenWorld & CodeOne session catalog using predefined, ready to run notebooks
  + And perhaps tweak and extend these notebooks a little
* On your own: get started on your very own Jupyter Notebook.

### Resources

Sources for this workshop: <https://github.com/lucasjellema/DataAnalytics--IntroductionDataWrangling-JupyterNotebooks>

# 1. Prepare your Jupyter Notebook environment

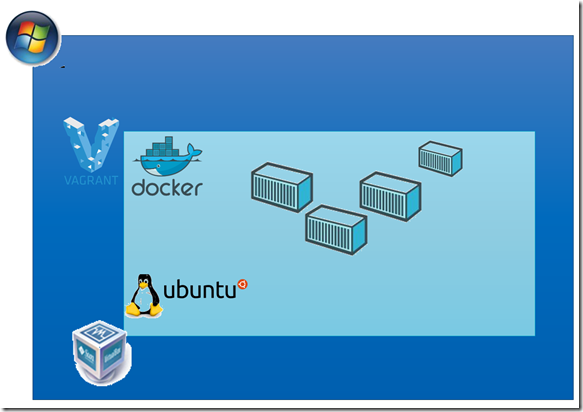
As stated before, you can choose between two options. One requires some experience with Docker and a fairly powerful laptop on which you can install software and run virtual machines. The other one requires nothing from you besides a browser – and leads you through a number of declarative, very low code steps. However, the environment you create in this scenario will exist only for a few hours, whereas the environment you install locally is yours for as long as you want.

## Local installation of Jupyter Notebook on Docker

Note: [this blog article](https://technology.amis.nl/2019/02/06/running-an-enriched-jupyter-notebook-runtime-in-a-docker-container-locally-or-cloud-side/) provides a little additional background for these steps.

### Starting Point, Prerequisites, Preparation

We will assume that you can start the installation with a Docker host – an environment where you can start Docker containers. The steps will be on Linux – if you are on Windows running Docker for Windows, you need to convert to the Windows counterpart commands. Or, alternatively use the combination of VirtualBox and Vagrant to manage Linux Ubuntu VMs with Docker inside and keep your Windows environment very uncluttered.



To quickly get going with that combination follow the instructions in this article: <https://technology.amis.nl/2018/05/21/rapidly-spinning-up-a-vm-with-ubuntu-and-docker-on-my-windows-machine-using-vagrant-and-virtualbox/>.

Note: you may have to install two vagrant plugins – in order to provide docker-compose into the VM and to allocate a greater than default disk size:

vagrant plugin install vagrant-docker-compose

vagrant plugin install vagrant-disksize

### Running Jupyter on Docker

Here we are. You can run Docker containers. But nothing in your environment is yet Jupyter Notebook or even Python specific. Good. Let’s change that.

I will assume that you are at the command line with Linux at your fingertips and Docker running in the backgroud. docker ps needs to return nothing at this point.

Many container images are available that contain Jupyter Notebooks in some form or shape. I will use the jupyter/scipy -notebook image from [Jupyter Docker Stacks](https://jupyter-docker-stacks.readthedocs.io/en/latest/index.html).The jupyter/scipy -notebook image is fairly rich image. It contains:

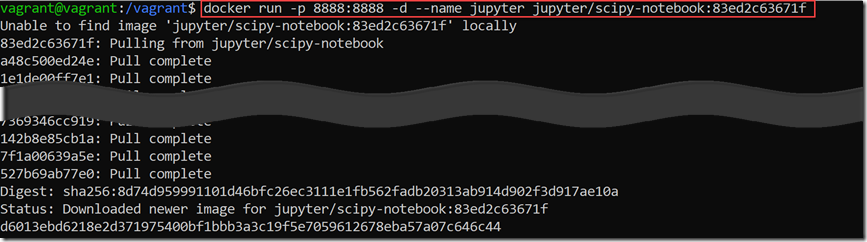
* Minimally-functional Jupyter Notebook server
* Miniconda Python 3.6
* Pandoc and TeX Live for notebook document conversion
* git, emacs, jed, nano, and unzip
* pandas, numexpr, matplotlib, scipy, seaborn, scikit-learn, scikit-image, sympy, cython, patsy, statsmodel, cloudpickle, dill, numba, bokeh, sqlalchemy, hdf5, vincent, beautifulsoup, protobuf, and xlrd packages
* ipywidgets for interactive visualizations in Python notebooks
* Facets for visualizing machine learning datasets

We will add a few other libraries on this ‘base’ image.

As first step: run a Docker container based on the image

docker run -p 8888:8888 -d –name jupyter jupyter/scipy-notebook:83ed2c63671f

Note: the Docker image tag (id) is no strictly necessary; if you strip it off (jupyter/scipy-notebook) you will get the latest – which may do everything you need. This particular id is from early February 2019 and it seems to work for me. See all Docker image tags:  <https://hub.docker.com/r/jupyter/scipy-notebook/tags/> .

[](https://technology.amis.nl/wp-content/uploads/2019/02/SNAGHTMLaf6e435.png)

The container image is quite sizable – close to 2 GB. Downloading is bound to take a while – depending on the network capacity you can leverage.

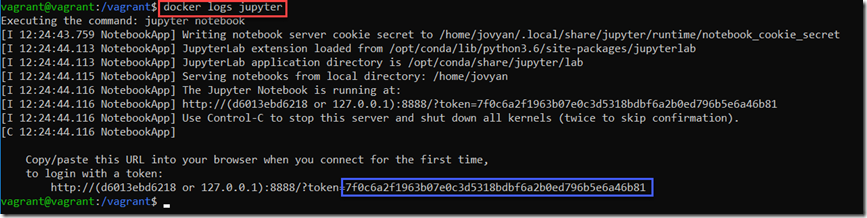
When downloading and extracting is complete for all layers, the container will be running. It exposes port 8888. The Jupyter server is accessible at that port.

[image](https://technology.amis.nl/wp-content/uploads/2019/02/image-45.png)

Access the Jupyter Notebook environment from a browser on your laptop; the endpoint depends on the IP address of the host running the Docker container. In my case, using the Vagrant file in the GitHup repo associated with this article, I will access the Jupyter Notebook at: [http://192.168.188.144:8888](http://192.168.188.144:8888/) .

The Jupyter server will prompt you for a token – to ensure not just anyone can access the environment. Back on the command line, execute this statement:

docker logs jupyter

[](https://technology.amis.nl/wp-content/uploads/2019/02/image-46.png)

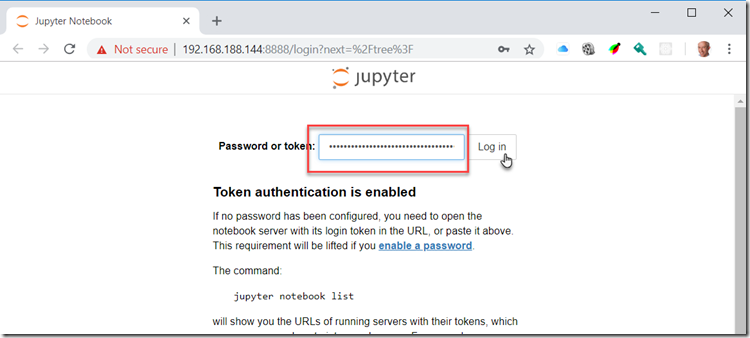
This will show something like:

Copy/paste this URL into your browser when you connect for the first time,

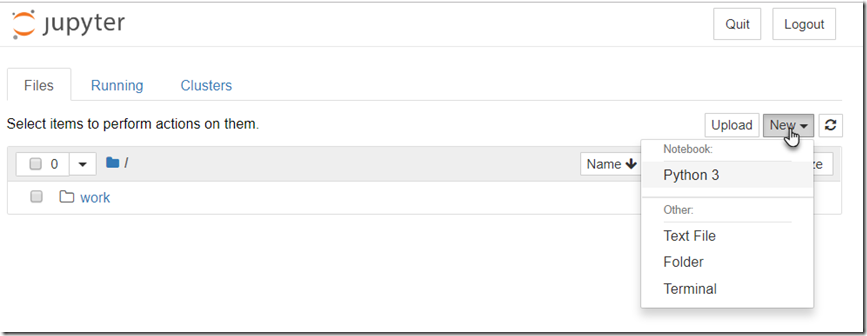
to login with a token:

http://localhost:8888/?token=f89b02dd78479d52470b3c3a797408b20cc5a11e067e94b8

The token is the value behind `/?token=`. You need that for logging in.

[](https://technology.amis.nl/wp-content/uploads/2019/02/image-47.png)

After pasting the token, click on the Log In button:

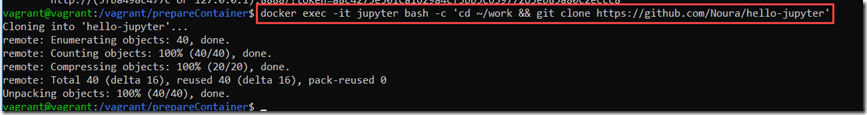
[](https://technology.amis.nl/wp-content/uploads/2019/02/image-48.png)

At this point, you can start creating your own notebook or upload a notebook from your laptop’s file system. The container currently does not contain any Jupyter Notebooks that we can open and run. We will change that in our next section.

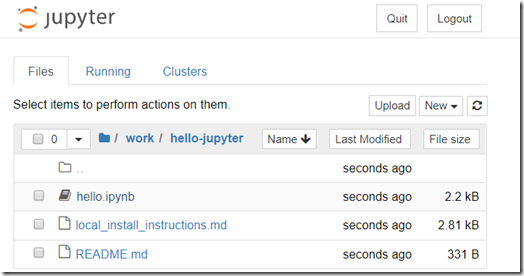
This section shows you how to add Python libraries to the running Jupyter container. This is not particularly complex, but a useful thing to know how to do. Additionally, you will install a number of packages required for the workshop that this article was originally written for. We will also look at adding a Jupyter Notebook from a GitHub repository into the running container. Again, a useful trick – and a necessary step in our workshop.

You can easily add more notebooks to the container, by cloning them from GitHub straight into the container and subsequently opening them in Jupyter Notebook. For example – to grab the world’s most trivial notebook:

docker exec -it jupyter bash -c ‘cd ~/work && git clone [https://github.com/Noura/hello-jupyter’](https://github.com/Noura/hello-jupyter')

[](https://technology.amis.nl/wp-content/uploads/2019/02/image-49.png)

After executing this command, this notebook can be opened in the Jupyter Notebook browser window from the folder `work/hello-jupyter`.

[](https://technology.amis.nl/wp-content/uploads/2019/02/image-50.png)

For more extensive manipulation of the Docker container, we can use a script that we copy into the container and then execute inside it – using docker cp (to copy files from the Docker host into a running container – or vice versa)  and docker exec (to execute a command inside the container).

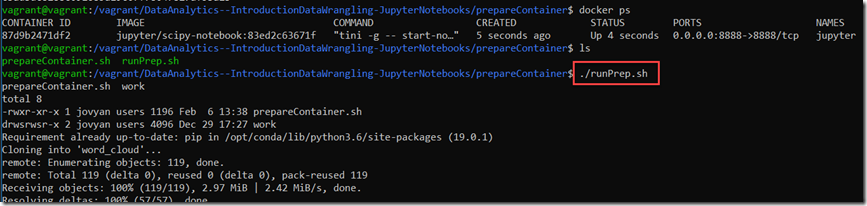
The [GitHub repo](https://github.com/lucasjellema/DataAnalytics--IntroductionDataWrangling-JupyterNotebooks) for this article has a folder prepareContainer that contains two scripts. You can run runPrep.sh to copy the script prepareContainer.sh into the container and execute – to install some packages and git clone a few notebooks.

Execute these steps – on the command line of your Docker host:

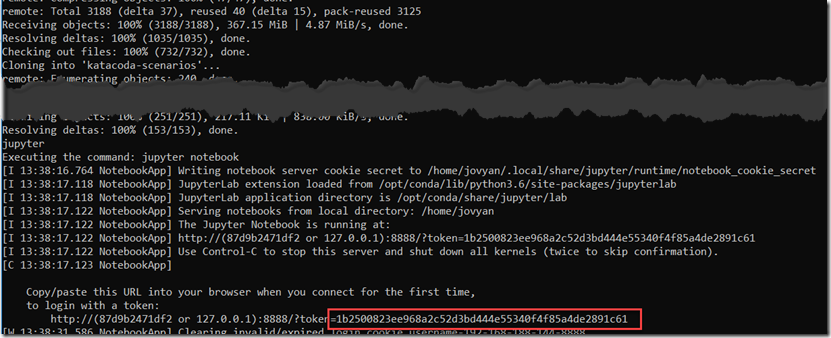
cd prepareContainer

./runPrep.sh

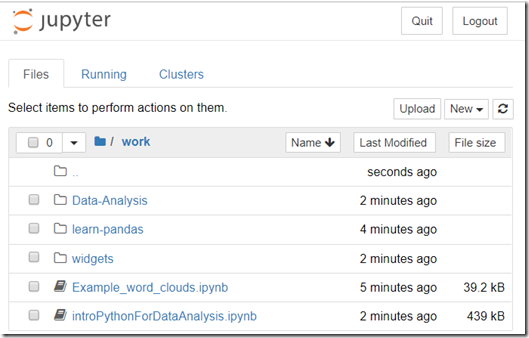
The script prepareContainer.sh is copied into the container and made executable. Then it is executed. It installs various Python packages using pip and git clones Jupyter notebooks into the container.

[](https://technology.amis.nl/wp-content/uploads/2019/02/image-51.png)

When the actions inside the container are done – note: this can take a few minutes – the container is restarted to have the Jupyter Notebook server pick up all changes. You may need a new token from the restarted server to login to the Jupyter Notebook environment in the browser.

[](https://technology.amis.nl/wp-content/uploads/2019/02/SNAGHTMLb3a7a07.png)

When you next enter the Jupyter Notebook environment in the browser, you will see a number of notebooks that were not there before.

[](https://technology.amis.nl/wp-content/uploads/2019/02/image-52.png)

For example: open and run pythonForDataAnalysis.ipynb in the work folder. Or open and run Example\_word\_clouds.ipynb.

Skip the next session on Cloud based Jupyter: you do not need that because you have your very own self controlled container running.

## Cloud based Jupyter Notebook Playground using Katacoda

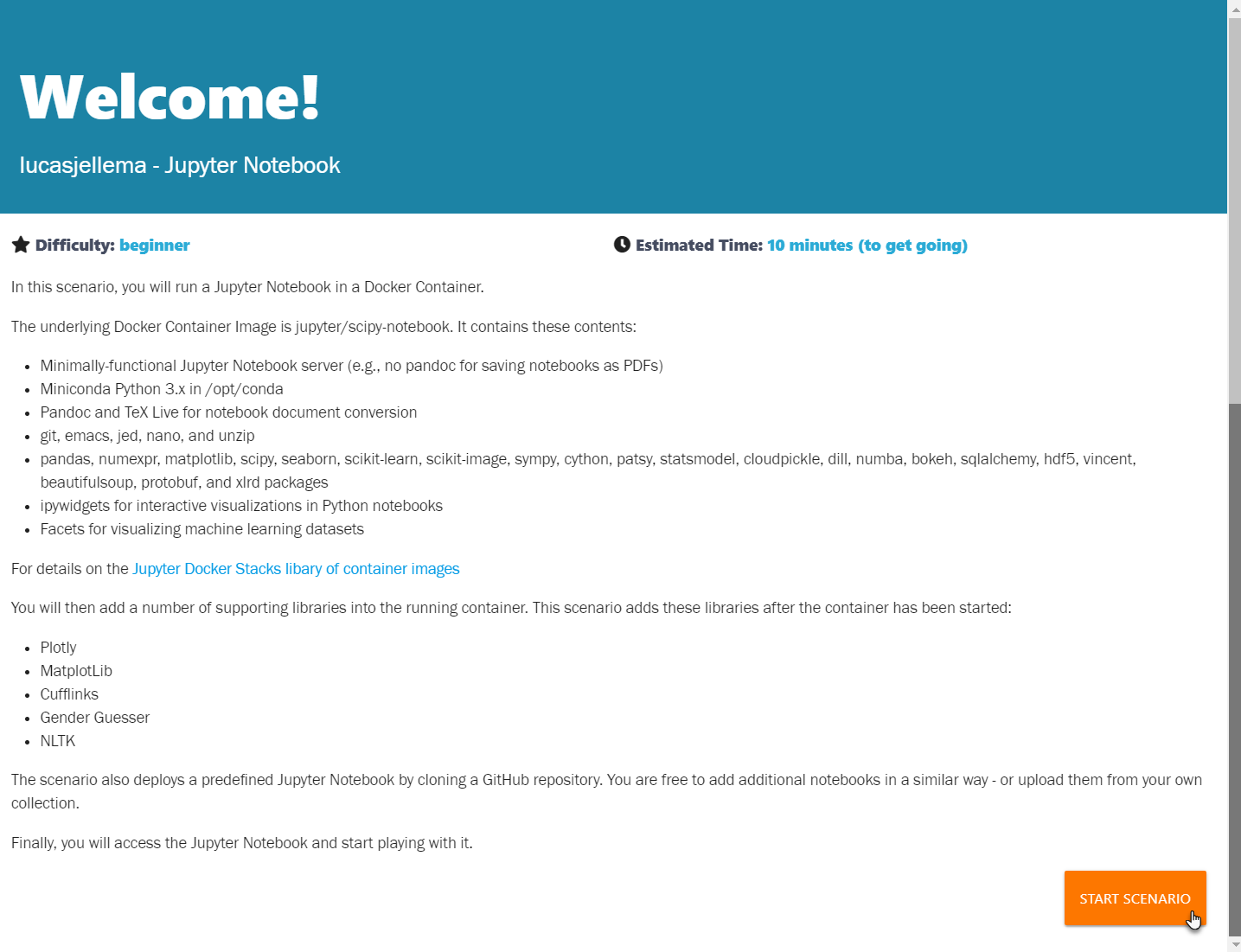


Katacoda is an online platform that offers hundreds of scenarios and sandbox environments to learn about and play with different kinds of technologies. Katacoda is special in that it not only offers the handson instructions – it also provides the runtime environment in which these steps can be executed instantly. Examples of such environments: Linux server, Docker engine, Kubernetes cluster and almost anything that can be run as container. I have created a Katacoda scenario for Jupyter Notebook – with a number of handpicked Python libraries and a Notebook git cloned from GitHub; with all the resources we need for our workshop.

In order to get your running Jupyter Notebook environment, you have to take these steps:

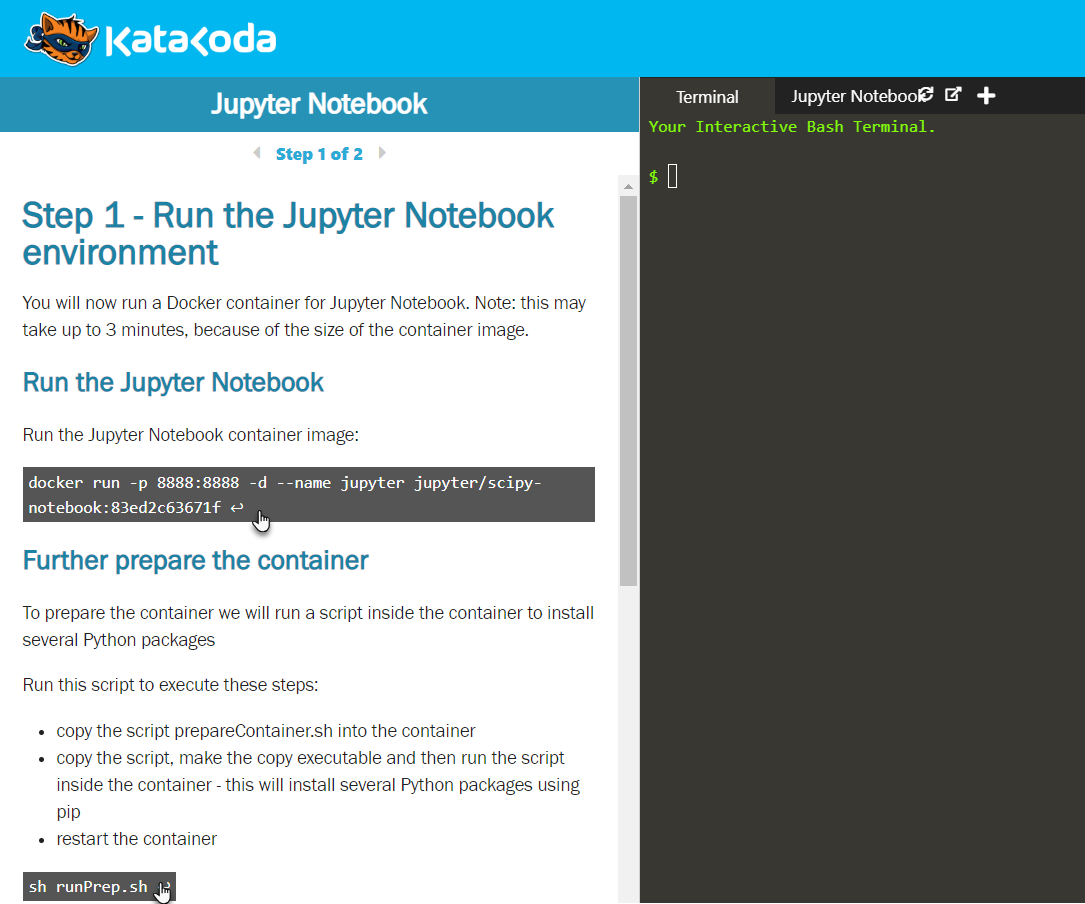
1. Open your browser at <https://www.katacoda.com/lucasjellema/scenarios/jupyter-notebook>

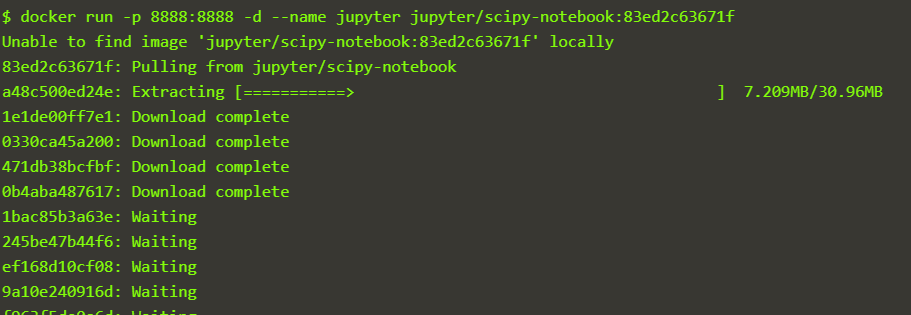
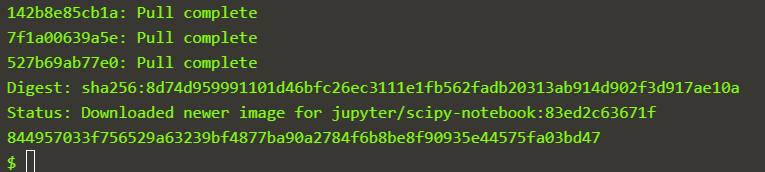
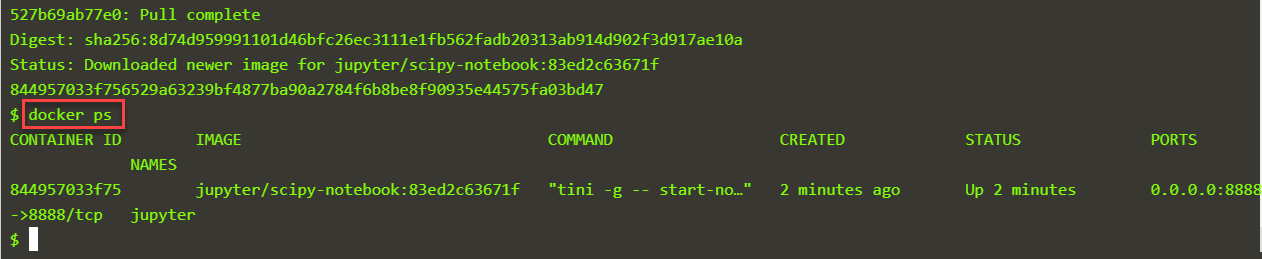
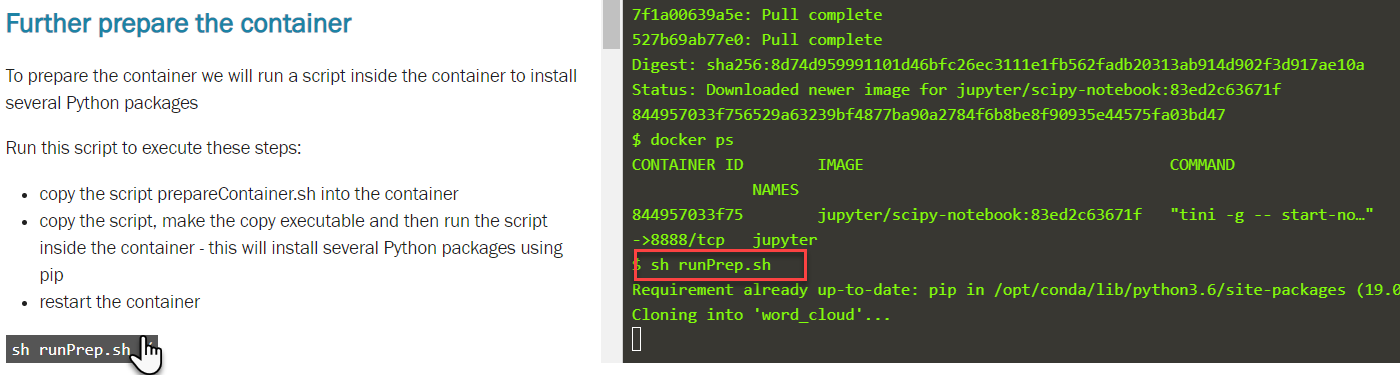
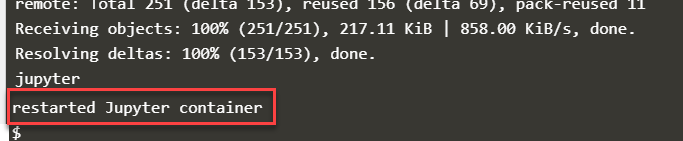
You will come to a page that looks like this figure:



2. Click on the button *Start Scenario*

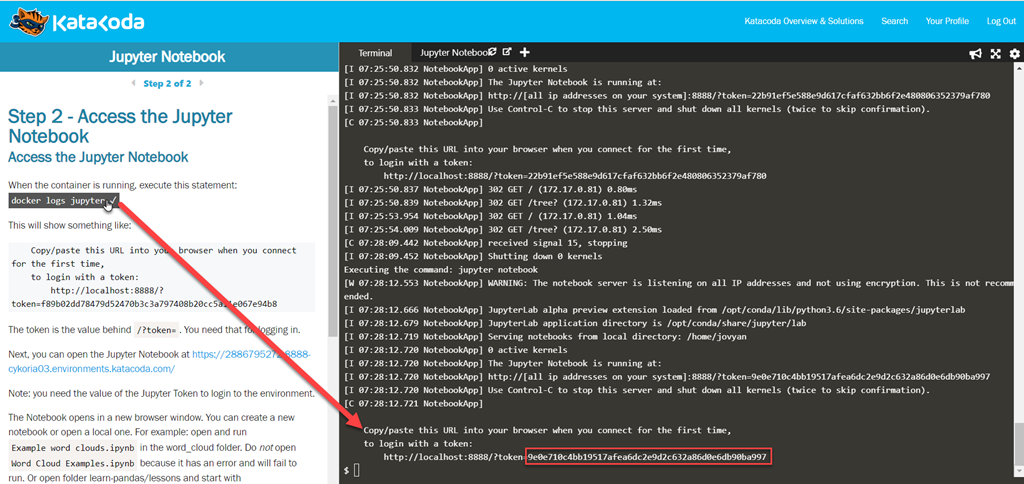
3. Follow the instructions in Step 1

The first step in the scenario looks as shown in the next figure.   
  
You have to click twice – to have two key commands executed that will configure your environment:

1. Click on the first dark gray area that contains a command starting with *docker run*. By clicking on this command, you cause it to be copied to the terminal window shown on the right where it is immediately executed.  
     
   This command instructs the Docker container engine to start the indicated container image jupyter/scipy-notebook. Because that image is not available yet, it needs to be downloaded. This will take some time (the download size is around 2 GB) – typically 2-4 minutes.   
     
     
   When the container is running – the terminal will show something like this:  
     
   If you feel brave, you could type a little something into the terminal window, for example: *docker ps*. This requests the Docker engine to show a list of all running containers:  
   
2. Click on the second gray area – containing a command *sh runPrep.sh*. This command will copy and execute a script. This script will do a number of things. This all contributes to making sure the container contains all required resources. Note: running this script will also take considerable time – another 2-4 minutes is typical.   
     
     
   When the script is complete – it will restart the Docker container to ensure all changes take effect.   
     
     
     
   Now you can proceed to ***step 2*** in the scenario.

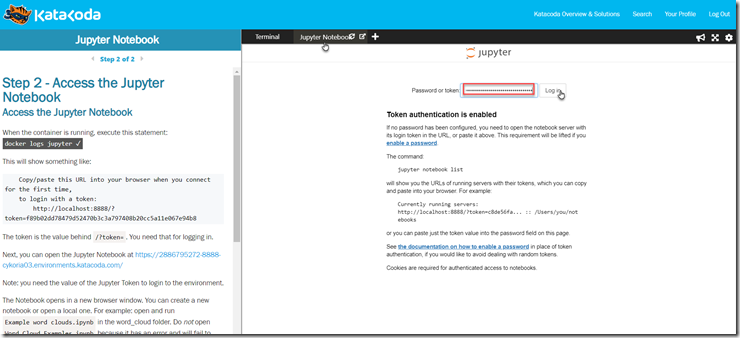
4. Perform the tasks in Step 2 in the Katacoda scenario

In Step 2, you will access the Jupyter Notebook server that is running at this point. In order to access the environment in a browser, we need a token. We can read the token from the log-files produced in the container. Click on the next code block – that contains *docker logs jupyter* -- to retrieve these logs. The logs are shown in the terminal and they will contain the token we need:

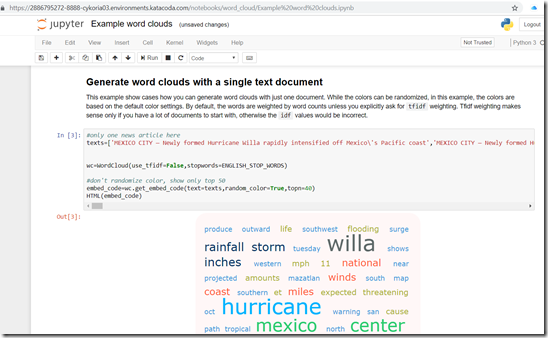


Copy the token shown in the terminal to the clipboard.

Click on the tab Jupyter Notebook – or the link in the text on the left hand side – to open the Jupyter Notebook web console. Paste in the token and press Login:

[](https://technology.amis.nl/wp-content/uploads/2019/02/image-21.png)

This will open the Jupyter web environment. Open for example the preloaded notebook Example Word Cloud (from File | Open), click on Cell | Run All and see the result:

[](https://technology.amis.nl/wp-content/uploads/2019/02/image-22.png)

Now you are ready for the next assignment.

Note: [this blog article](https://technology.amis.nl/2019/02/03/creating-a-katacoda-scenario-a-tailor-made-on-line-tutorial-environment-for-example-for-jupyter-notebook/) explains in detail how this playground was set up on Katacoda as a scenario. You will find that the scenario contains almost the same steps as are used for the local installation. You can also create your own Katacoda scenarios, for Jupyter Notebook, other data analytics activities as well as virtually anything you can run on a standard Linux environment.

# 2. Get acquainted with Jupyter Notebook - Hello World notebook

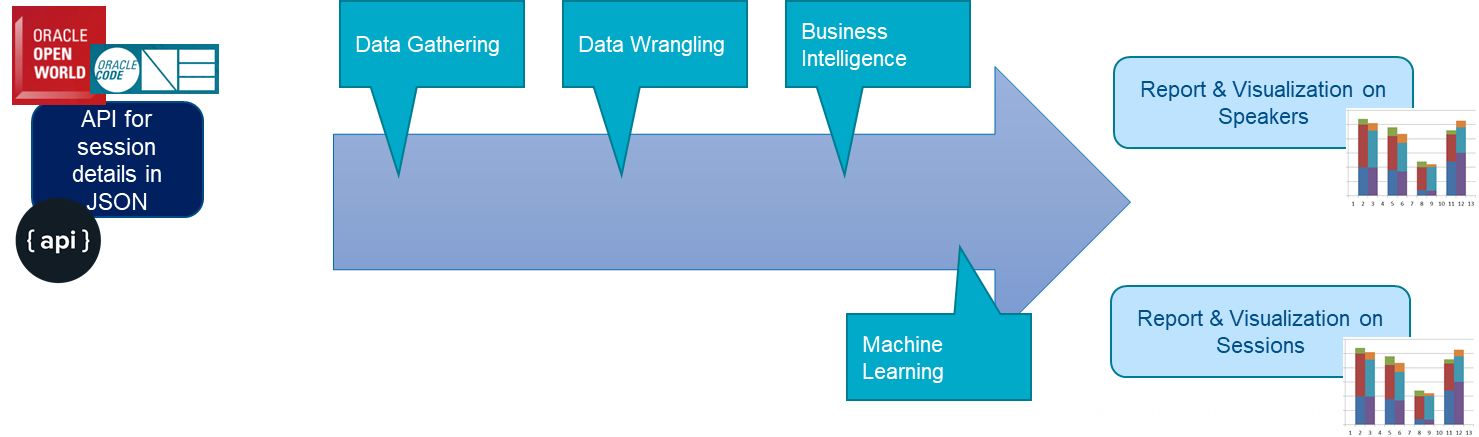
# 3. Explore the case of the Oracle OpenWorld & CodeOne session catalog

Oracle OpenWorld 2018 was a conference that took place in October 2018 in San Francisco. Over 30,000 attendees participated and visited some 2000 sessions. In this section, we will explore some of the data around this event. We will take a closer look at the speakers and their origin and at the sessions: what are they about, how are they tagged, what are their target audiences and how are they scheduled logistically?

Raw data from the session catalog is available from an API – a REST service that is consumed by the Session Catalog Web UI. This API does not seem intended for data analysts such as we. The data is not in great shape.

The next figure illustrates the steps we will go through in order to get from raw data to answers to burning questions about Oracle OpenWorld 2018 and the collocated CodeOne conference.

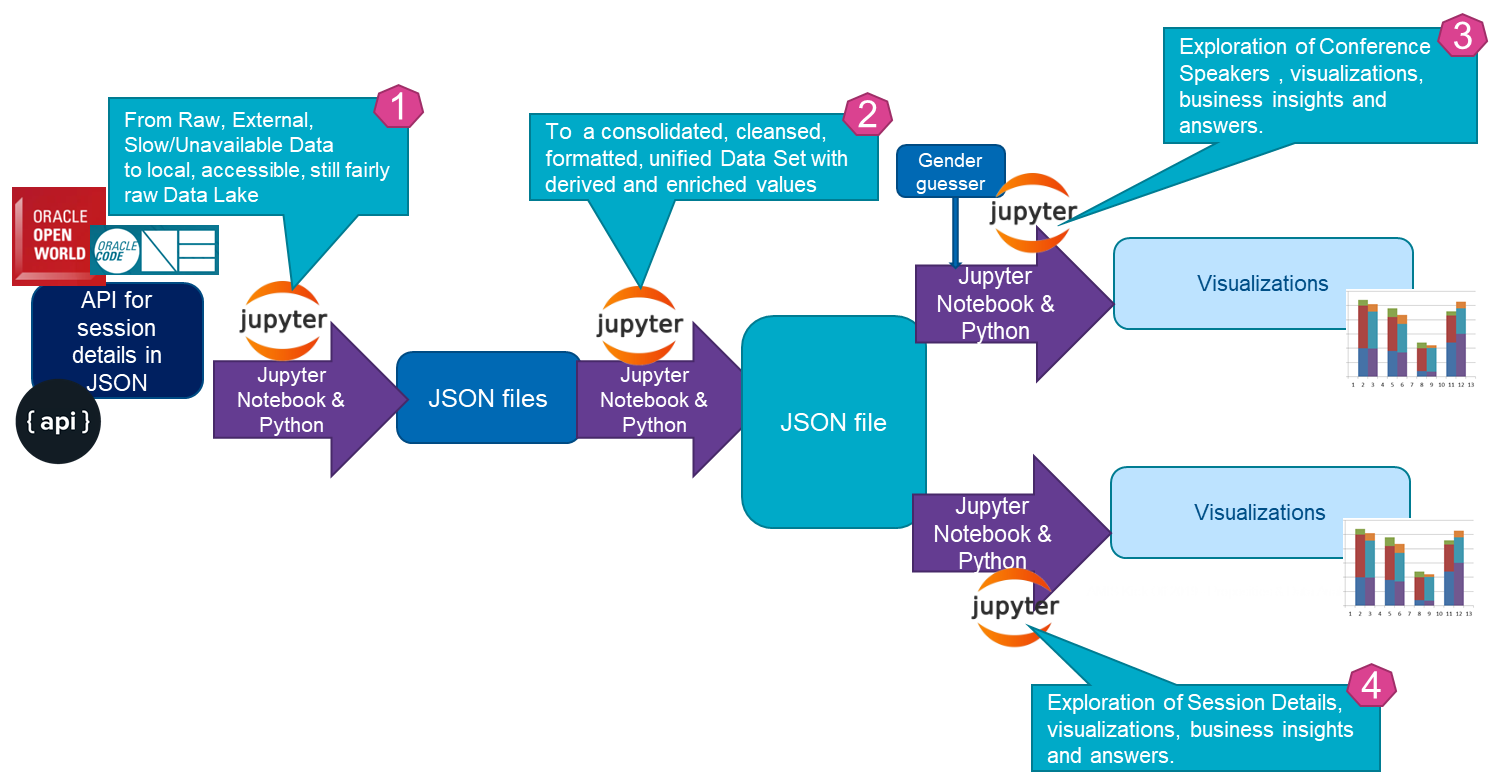
From the raw source, we will gather the data into our local datalake. Next, we will wrangle this data – whip it into shape, filter, cleanse, enrich it so it is easily usable for business intelligence and data science tasks such as machine learning.



We will use various Jupyter Notebooks – powered by Python – to go through these four stages:

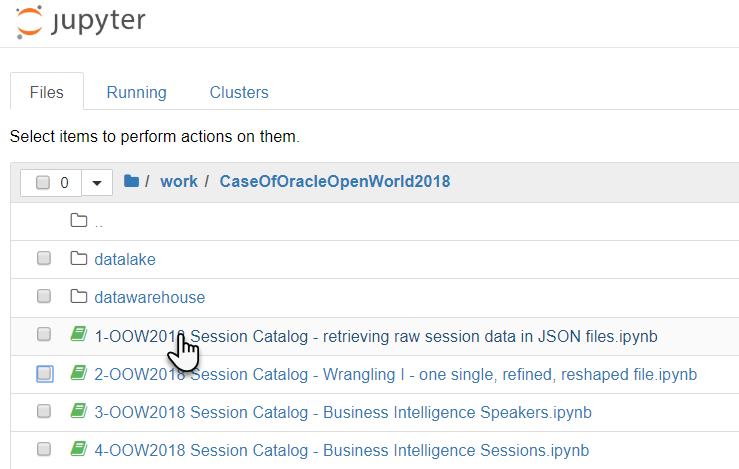
* 1-OOW2018 Session Catalog - retrieving raw session data in JSON files
* 2-OOW2018 Session Catalog - Wrangling I - one single, refined, reshaped file
* 3-OOW2018 Session Catalog - Business Intelligence Speakers
* 4-OOW2018 Session Catalog - Business Intelligence Sessions

The next figure gives a high level overview of what each of these notebooks does:

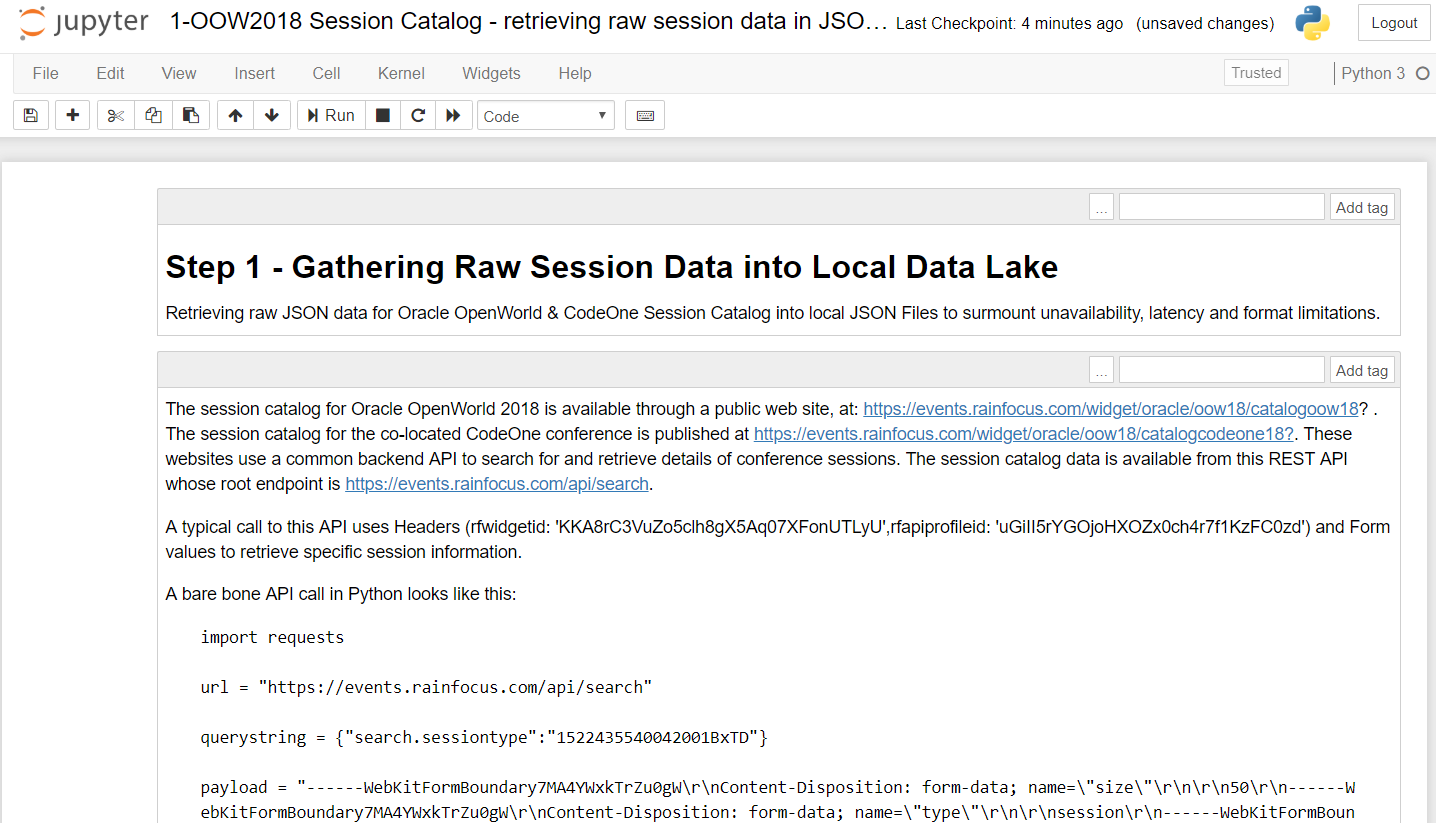


## 1-OOW2018 Session Catalog - retrieving raw session data in JSON files

From the Jupyter Notebook client running in your browser, open the file *1-OOW2018 Session Catalog - retrieving raw session data in JSON files.ipynb* notebook, from the */work/CaseOfOracleOpenWorld2018* folder.



Note: you do not actually need to run this notebook, although you can. Its final output is already available in the */datalake* folder. You can simply browse through the notebook to see how the data retrieval and local storage is done.

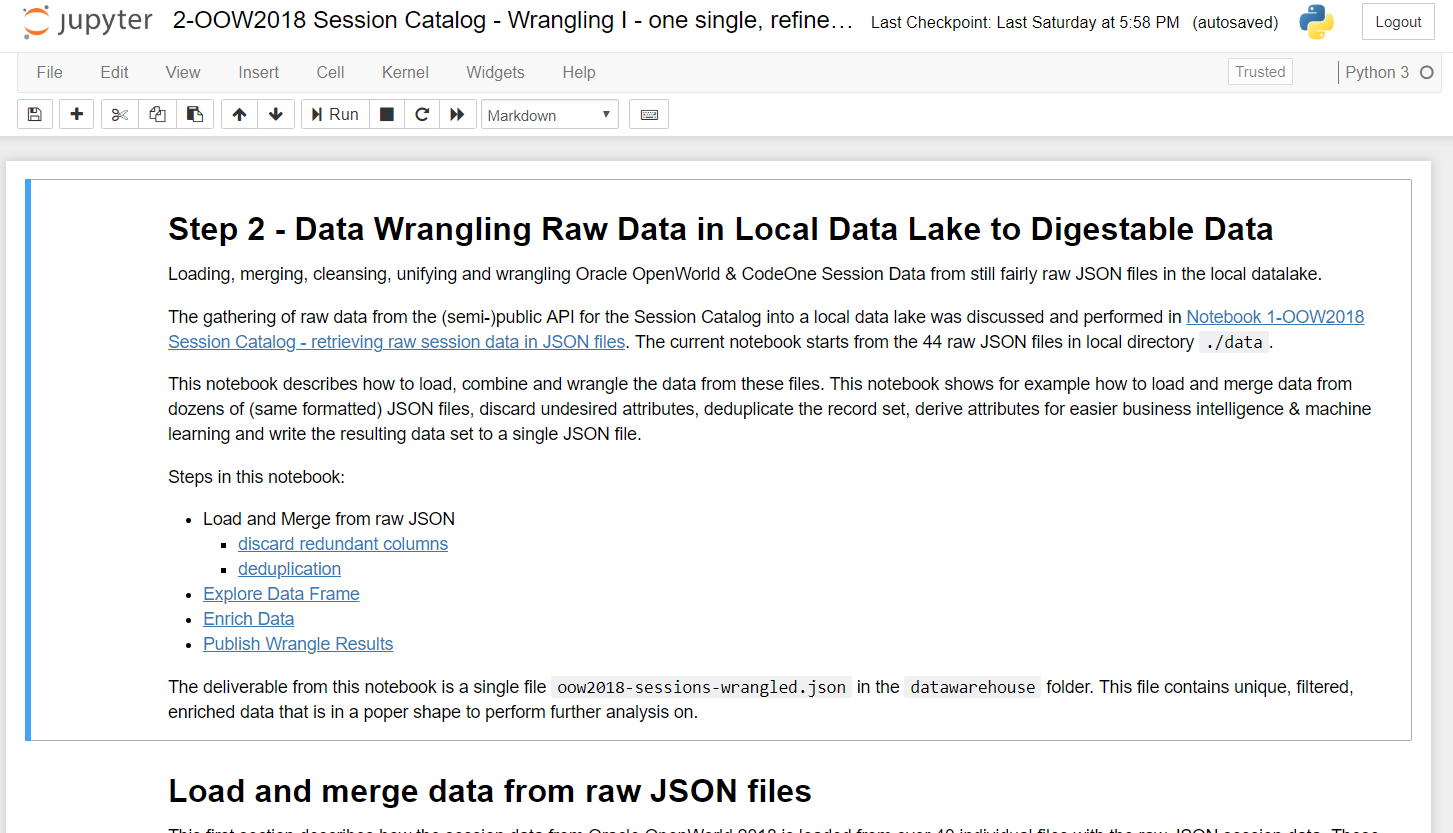


This notebook retrieves the raw data from the remote API – using subsequent HTTP REST requests. The data is gathered using calls per session type (22 different types), per event (two events – Oracle OpenWorld and CodeOne) and per batch of 50. This means that close to 60 HTTP calls are made. The resulting data is in JSON format. This data is written to local files per event and session type (44 in total) in the folder */datalake*.

By perusing the notebook, you will get an idea of how to perform REST calls, how to process the response body and JSON data content, how to turn JSON content in a Pandas Data Frame and how to write the content of such a Data Frame to a JSON file on disk. You will get a glimpse of the raw data. In the next section, we will discuss the Jupyter Notebook that does the main data wrangling, to turn this raw, not very accessible set of data files into a single, refined data warehouse that will be to the delight of data analysts.

## 2-OOW2018 Session Catalog - Wrangling I - one single, refined, reshaped file

Open the notebook *2-OOW2018 Session Catalog - Wrangling I - one single, refined, reshaped file.ipynb* into your Jupyter Notebook client.



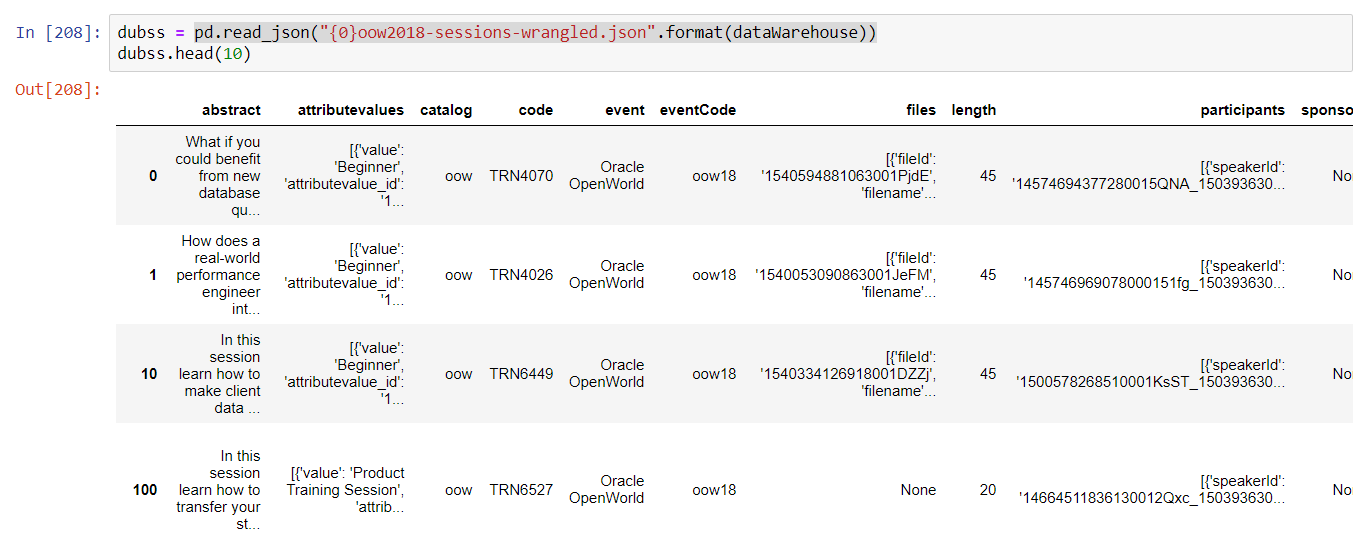
This notebook takes 44 separate JSON files in the Data Lake as its input and produces a single consolidated, cleansed and enriched JSON file as its output. The output set has been rid of unnecessary attributes and has newly engineered features that expose attributes for easy data analytics downstream in the data analytics pipeline.

By browsing or stepping through the notebook, you will some important steps in the Data Wrangling process. The 44 JSON files are loaded into Pandas Data Frames that are subsequently merged into one big Data Frame. This joined Data Frame is explored a little – what are the columns, what are the values in those columns. Then unneeded columns are removed – and the set of records [representing conference sessions] is deduplicated by *session code*. This takes the set down to just over 1700 sessions.

After a little digging around some of the columns in the Data Frame, we look closer at the attributeValues, participants and files columns. Each of these contains a nested JSON object that contains relevant data – albeit in a fairly inaccessible way. We expose these data elements by engineering new features, for example

* a flag indicating whether the session is (co-)presented by an Oracle employee,
* another flag indicating if a file (with the presentation) slides has been uploaded,
* flags for specials speaker designations (such as Oracle ACE Director and Java Champion)
* features for each of three audience levels (beginner, intermediate, advanced) and one for *all* in case all three are marked
* a feature for track(s) to which the session is assigned
* a feature for the number of speakers associated with a session
* a session instance count for the number of times the session is scheduled to take place
* a feature to indicate the room capacity of the (first) room in which the session is scheduled
* features for day, time(slot) and timestamp

Finally, with all these features added to the data set, the data is saved to disk in folder */datawarehouse* as single, consolidated file that is just waiting for data analysts to get their hands on.

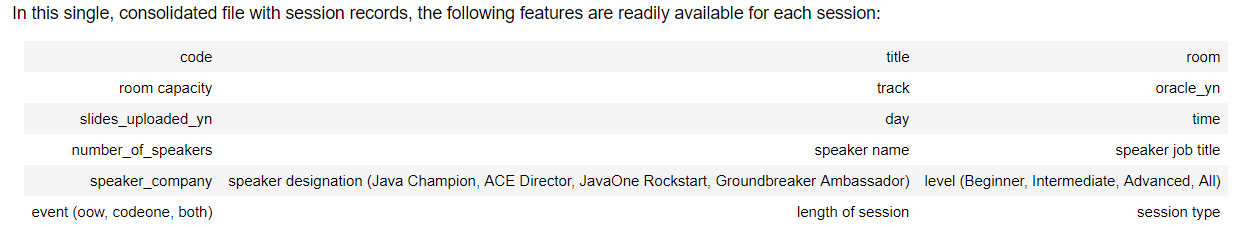


## 3-OOW2018 Session Catalog - Business Intelligence Speakers

The third notebook is the first to leverage the wrangled data, do some additional wrangling and then produce findings, answers on business questions and show some data visualizations.

Load *3-OOW2018 Session Catalog - Business Intelligence Speakers.ipynb* into the Jupyter Notebook client.

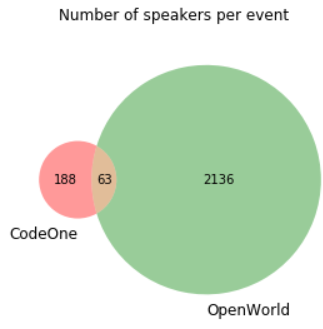
Here is a list of all features that our data set contains at this stage



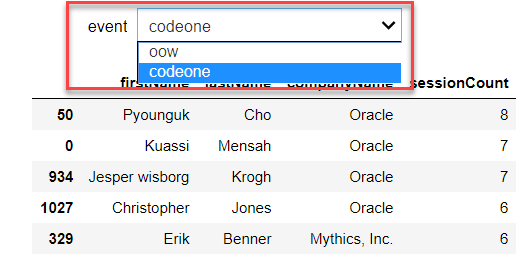
The first step after loading the wrangled dataset from the */datawarehouse* folder is further data preparation: a new Pandas Data Frame *sp* is created with speaker records. These records are produced from the *participants* feature in the sessions data set.

The *sp* data frame has the following features: firstName, lastName, jobTitle, companyName, bio, oracle\_employee (special Y/N flag), photoURL, twitter, sessionCount, oow (Y/N flag to indicate if the speaker performs at the Oracle OpenWorld event), codeone (Y/N flag to indicate that the speaker presents a session at CodeOne) and finally four Y/N flag columns for the special designations that speakers can have: 'JavaOne Rockstar', 'Oracle ACE Director', 'Oracle Java Champion', 'Groundbreaker Ambassador'.

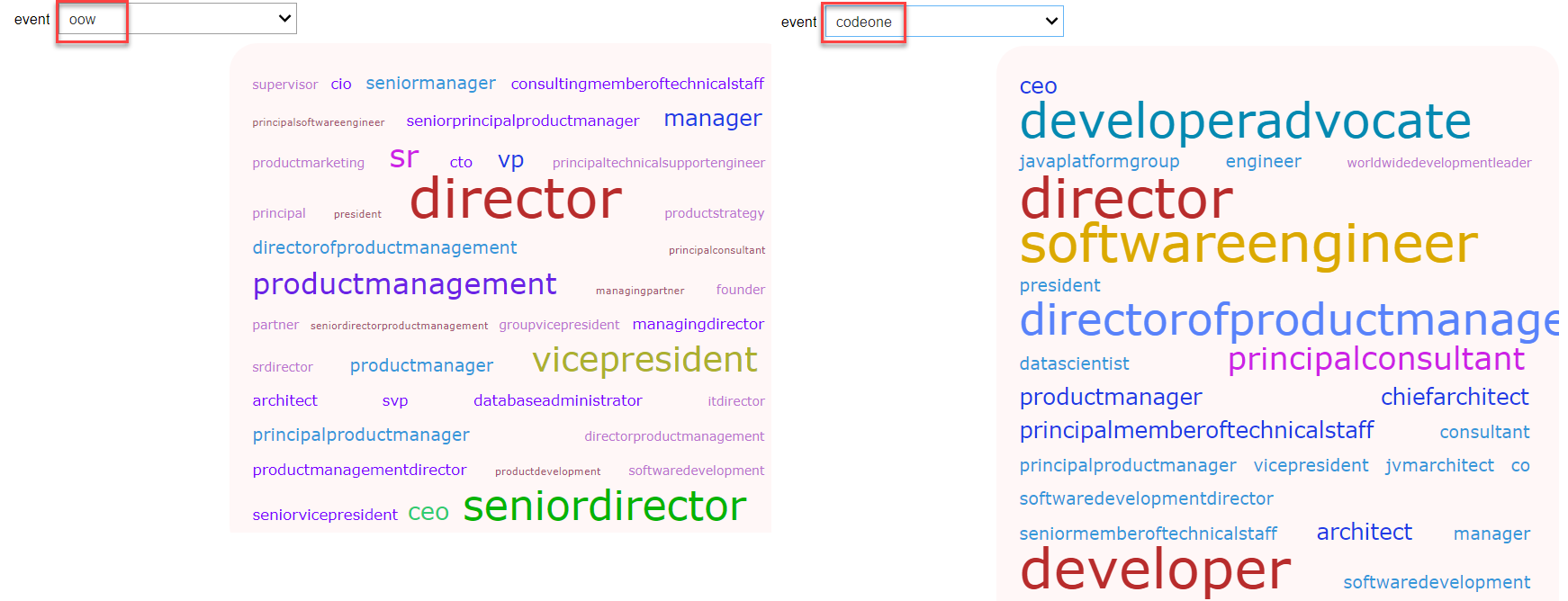
The first section in the notebook after the creation of the *sp* Data Frame for speakers is about simple counting: the number of speakers in general and per event. Here we also see te first visualization: a Venn diagram as alternative representation for data earlier presented in a Cross Table.



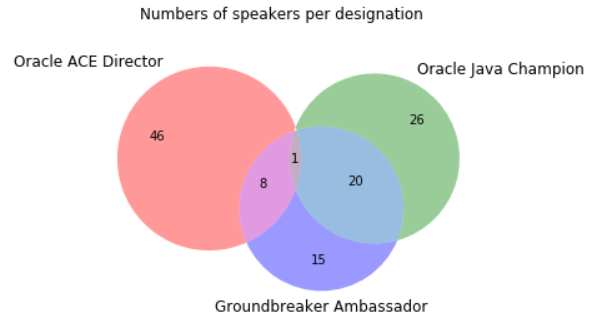
The notebook continues with an analysis of very active speakers. It turns out that one speaker takes part in now fewer than 12 sessions. It turns out that the top 10 of active speakers only has speakers working for Oracle. We then have a cell in the notebook that allows us to look at active speakers per event – for oow and codone respectively. This cell uses an interactive widget – in this case a dropdown list – that allows us to interact with the notebook:



The next section looks at Job Titles. The final visualization in this section uses a Word Cloud to show the most frequently used job titles – for the Oracle OpenWorld and CodeOne events side by side. The difference is striking:

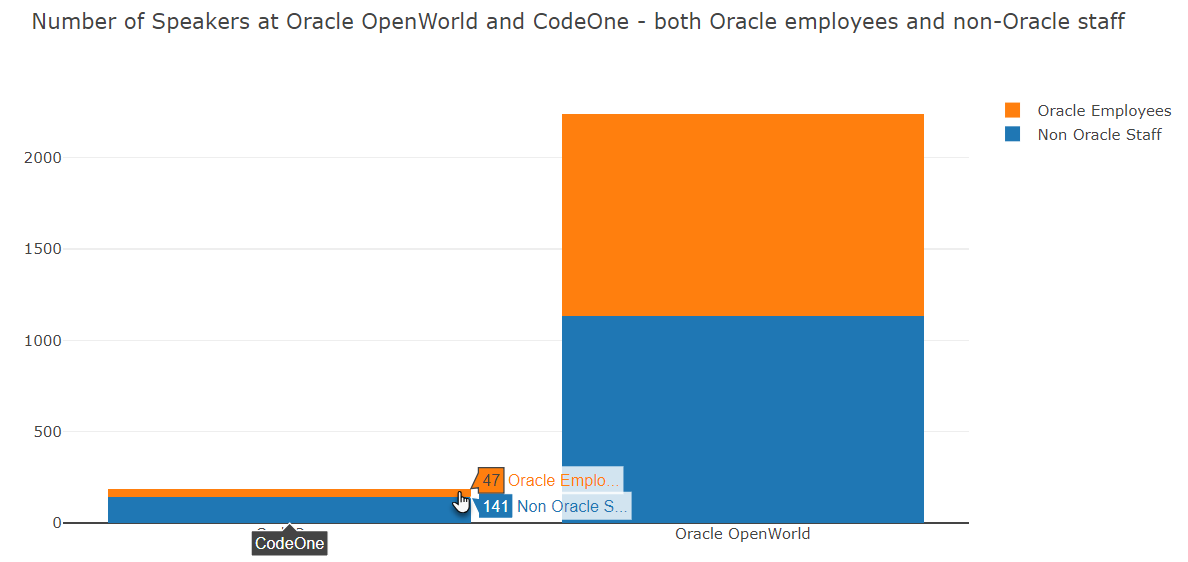


The notebook does a brief analysis of the designations or titles of speakers. It counts the number of speakers per designation and looks at how many speakers have specific combinations of titles. For some people in the community, this is a very important topic.



The companies that the speakers are working for is looked at next. It quickly turns out that in the top 10 of companies with the highest numbers of speakers, we find mainly sponsors of the events. Wells Fargo is the first non-sponsor company on the list.

The number of speakers employed by Oracle is very substantial, especially at Oracle OpenWorld:



A brief analysis of the incidence of first names quickly reveals that David, John and Michael are the top 3 first names among speakers.

### Gender Guessing

It is interesting to know if women appear as speakers at the two events - given our strive for more *women in tech*. We may be able to stimulate the participation of women if we better understand what type of event attracts female speakers and which tracks perhaps have a higher than average percentage of female speakers. However: our data set does not contain and straightforward gender information. Gender is not a feature of our speaker records.

Does our exploration end before it event started - for lack of data? Or can we be creative? Perhaps we can guess the gender from the first name feature that we do have for all speakers. And check the biographies or gender specific terms such as he and she, his, hum and her. Using these tricks – which turns out to be not very difficult at all - we get a fair idea about the gender distribution and whether there is a significant distinction between Oracle OpenWorld and CodeOne in this regard.

## 4-OOW2018 Session Catalog - Business Intelligence Sessions

Load *4-OOW2018 Session Catalog - Business Intelligence Sessions* into the Jupyter Notebook client.

# 4. On your own: get started on your very own Jupyter Notebook.

--

To be provided by Jeffrey