Big Data Technology Warsaw Summit 2017
INTERNATIONAL CONFERENCE
FEBRUARY 9, 2017
WARSAW, POLAND
bigdata
Independent Big Data conference with
purely technical presentations

One jupyter to rule them all

Mariusz Strzelecki (Allegro Group)

About me

- Senior Data Engineer @ Allegro
- pySpark advocate
- tiny contributions to Spark, Hue, Hive
- stackoverflow pyspark tag watcher

allegro

- the biggest marketplace platform in Central Europe
- **52M** active offers
- **18M** transactions per month (population of Poland: 38M)
- Hadoop: **4PB** of storage, **12TB** of RAM

Data processing story @ allegro

- Legacy (2015)
  - **MapReduce** in Java
  - SQL in Hive (via Hue)

Data processing story @ allegro

- Legacy (2015)
  - **MapReduce** in Java
  - SQL in Hive (via Hue)
- Actual workflow
  - SQL in Hive (via Hue) $\leftarrow$
  - exporting results to CSV
  - analyzing aggregates in Excel $\lrcorner$

Data processing story @ allegro

- Legacy (2015)
  - **MapReduce** in Java
  - SQL in Hive (via Hue)
- Actual workflow
  - SQL in Hive (via Hue)
  - exporting results to CSV
  - analyzing aggregates in Excel
- Java microservices
  - connecting to Hive via JDBC
  - a lot of Kerberos issues

Data processing story @ allegro

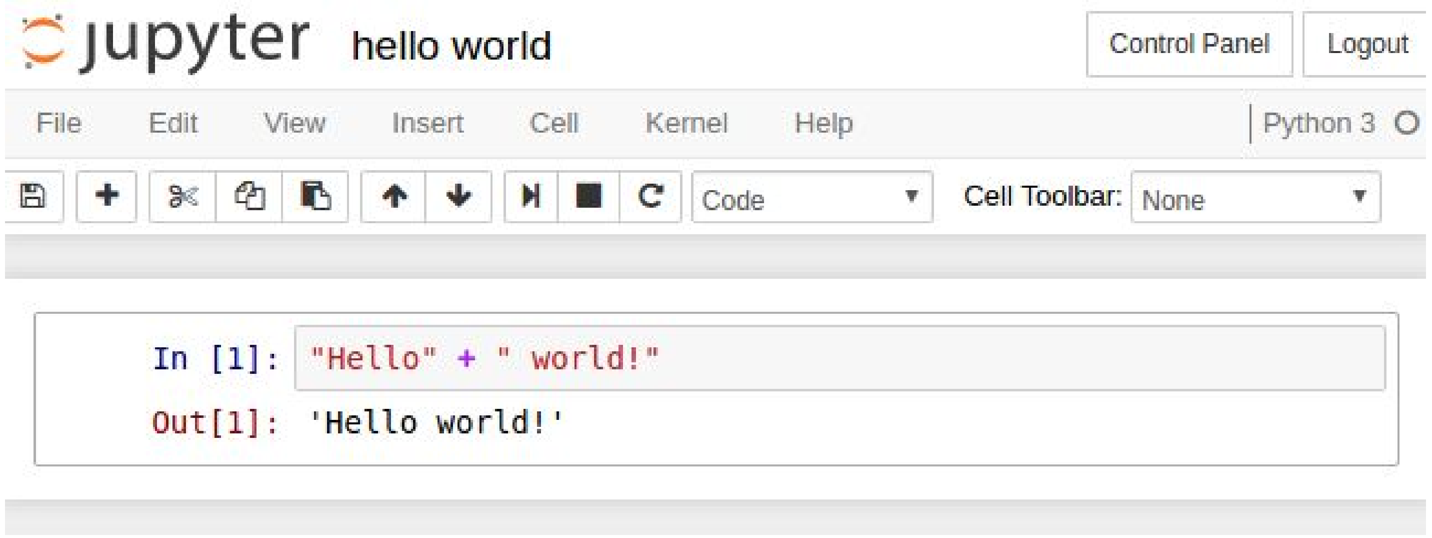
- **Spark** using java and scala
    - JVM dependency hell
    - cumbersome workflow: build jar → scp → test → fix → repeat
    - spark-shell is OK, but not usable by analysts

Data processing story @ allegro

- **Spark** using java and scala
  - JVM dependency hell
  - cumbersome workflow: build jar → scp → test → fix → repeat
  - spark-shell is OK, but not usable by analysts
- **PySpark**
  - more understandable api
  - still CLI, spark sessions via `ssh` with `screen`

Jupyter for the rescue!

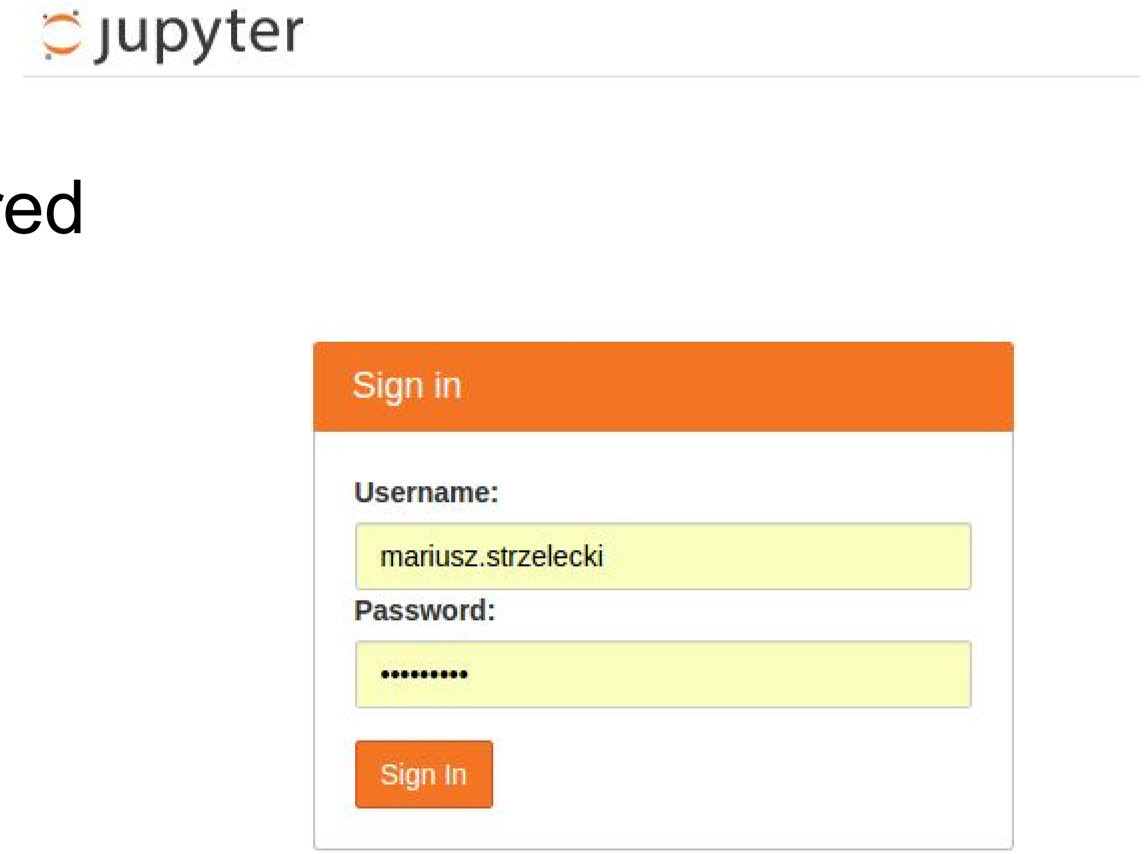
- **no CLI** anymore



```
In [1]: "Hello" + " world!"
Out[1]: 'Hello world!'
```

Jupyter for the rescue!

- **no CLI** anymore
- **no Kerberos knowledge** required

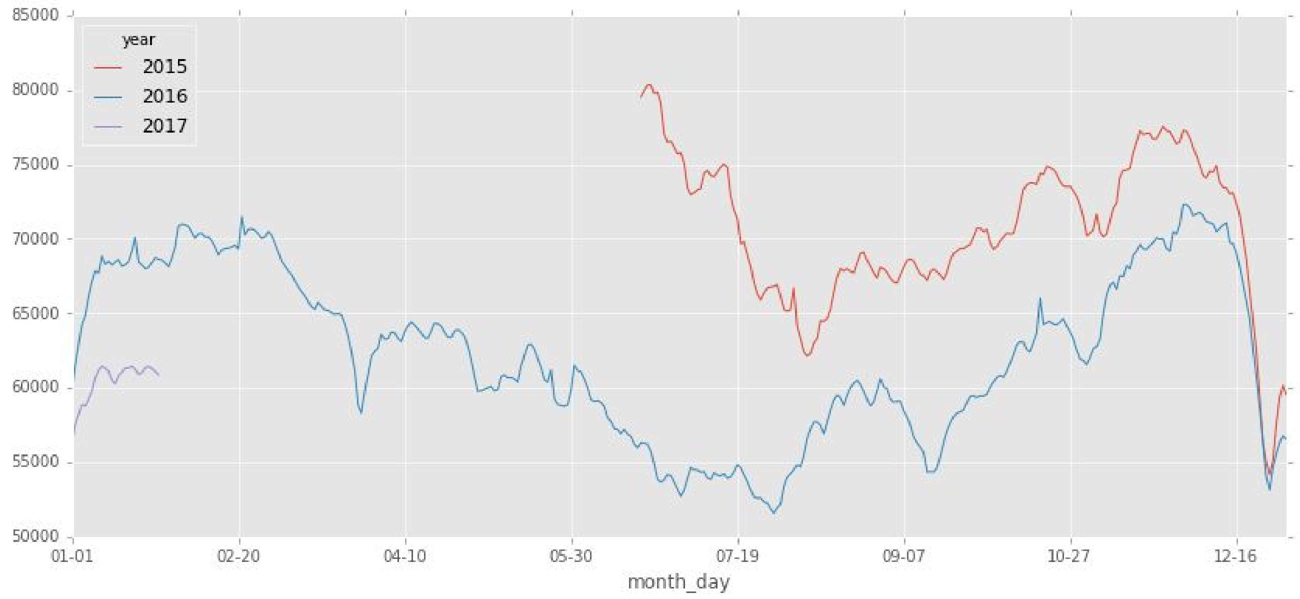


mariusz.strzelecki

••••••••••

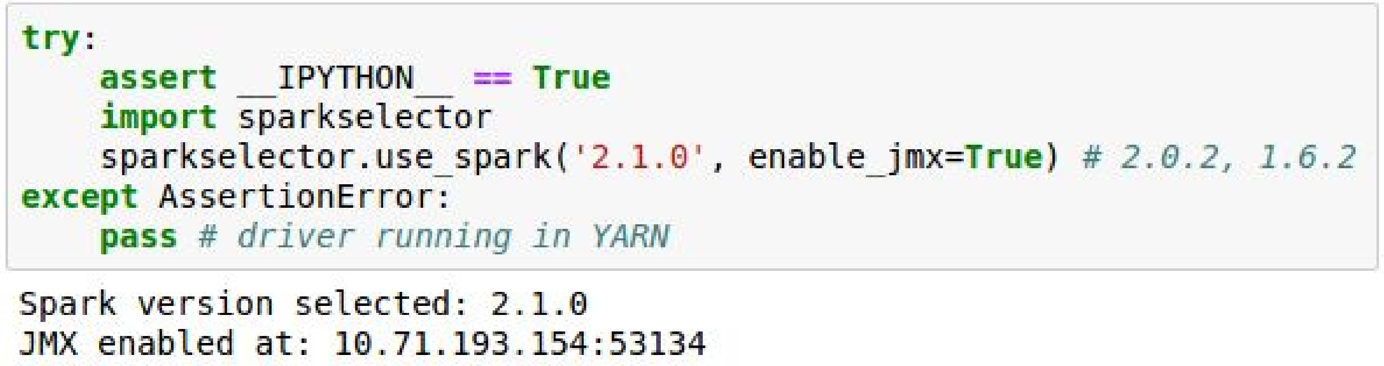
Jupyter for the rescue!

- **no CLI** anymore
- **no Kerberos knowledge** required
- great tool for **visualizations**



Jupyter for the rescue!

- **no CLI** anymore
- **no Kerberos knowledge** required
- great tool for **visualizations**
- easy **switching** between **Spark versions**



```
try:
    assert __IPYTHON__ == True
    import sparkselector
    sparkselector.use_spark('2.1.0', enable_jmx=True) # 2.0.2, 1.6.2
except AssertionError:
    pass # driver running in YARN
```

```
Spark version selected: 2.1.0
JMX enabled at: 10.71.193.154:53134
```

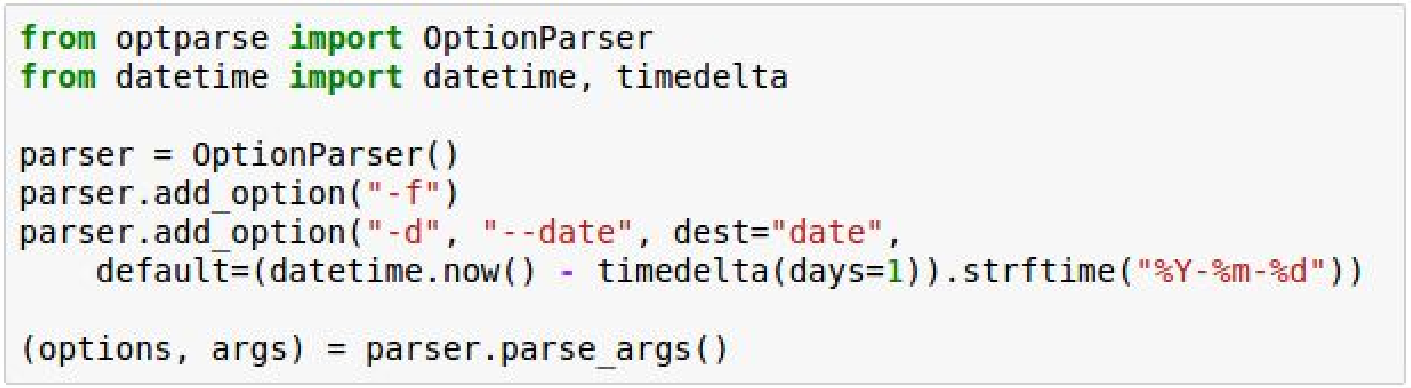
But… what about cyclical jobs and ETLs?

- filling up **BI dashboards**
- looking for **anomalies**
- feeding **search engine** with **bestmatch** features
- updating **machine learning models**
- creating **snapshots** of microservices data
- exporting data from **HDFS to databases**

Let’s make notebook a job

Let’s make notebook a job

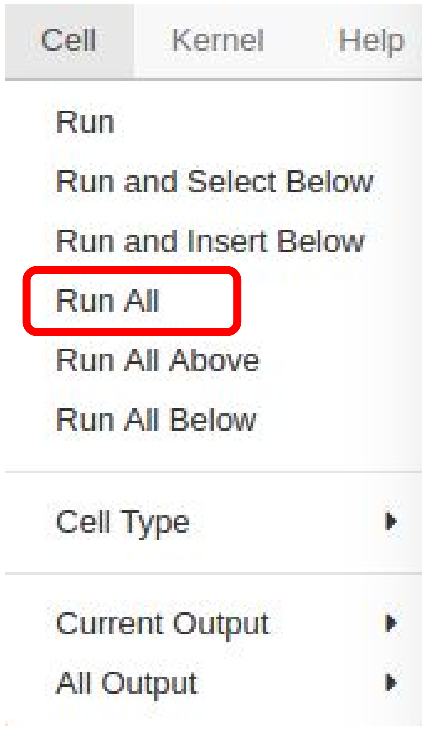
- Add **parametrization**



```
from optparse import OptionParser
from datetime import datetime, timedelta

parser = OptionParser()
parser.add_option("-f")
parser.add_option("-d", "--date", dest="date",
    default=(datetime.now() - timedelta(days=1)).strftime("%Y-%m-%d"))

(options, args) = parser.parse_args()
```

Let’s make notebook a job

- Add **parametrization**
- Ensure it runs **from the top** to the bottom



Let’s make notebook a job

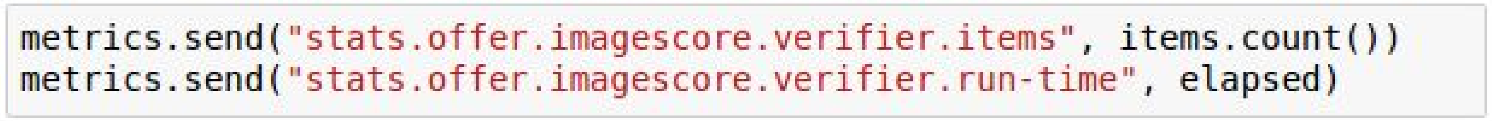
- Add **parametrization**
- Ensure it runs **from the top** to the bottom
- Add some **logging**

```
logger.warn("Corrupted record {} in {}".format(record, dataset))
```

```
logger.warn("Duplicated data found: " + duplicates.count())
```

Let’s make notebook a job

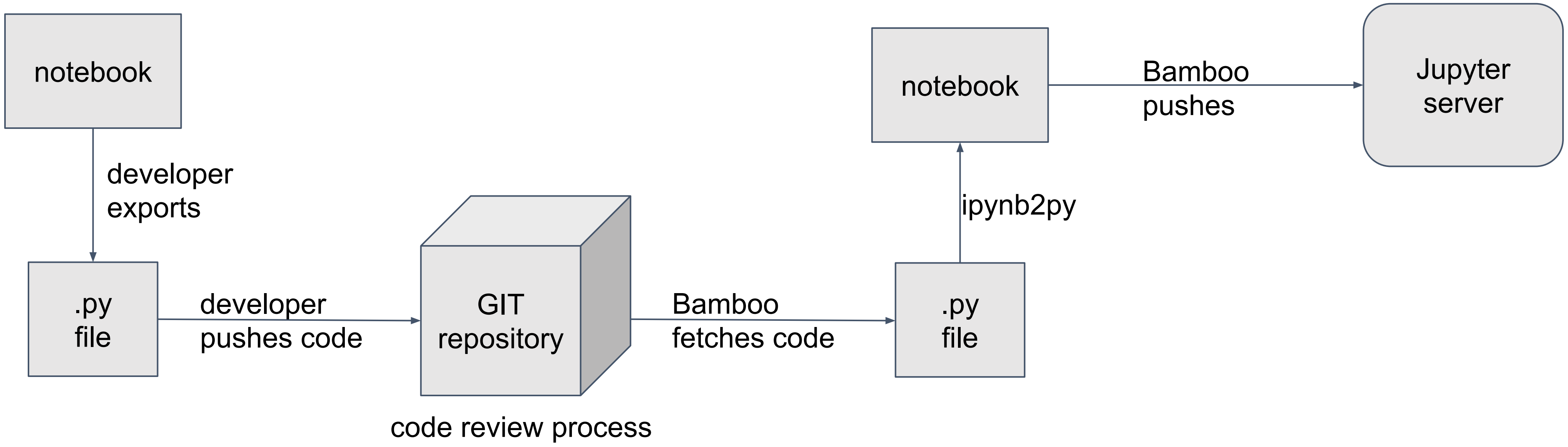
- Add **parametrization**
- Ensure it runs **from the top** to the bottom
- Add some **logging**
- Send **metrics**



```
metrics.send("stats.offer.imagescore.verifier.items", items.count())
metrics.send("stats.offer.imagescore.verifier.run-time", elapsed)
```

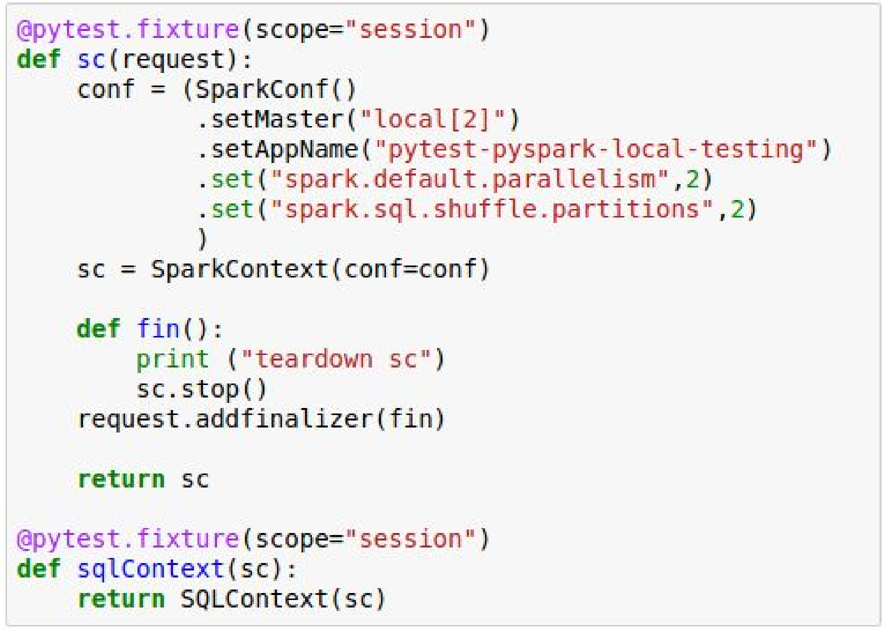
Fitting in applications ecosystem

- Code review and git flow

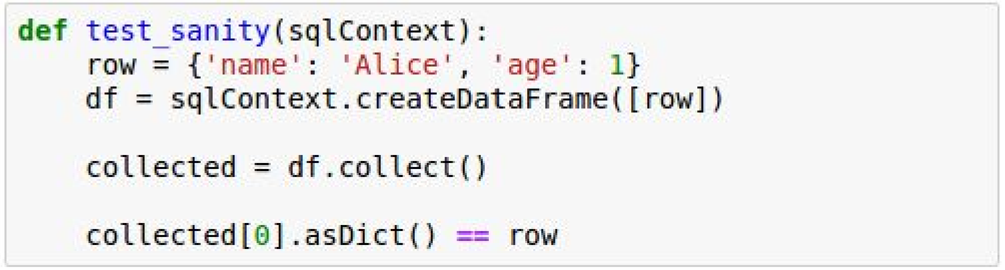


Fitting in applications ecosystem

- Code review and git flow ✔
- Unit tests
  - using py.test inside docker container



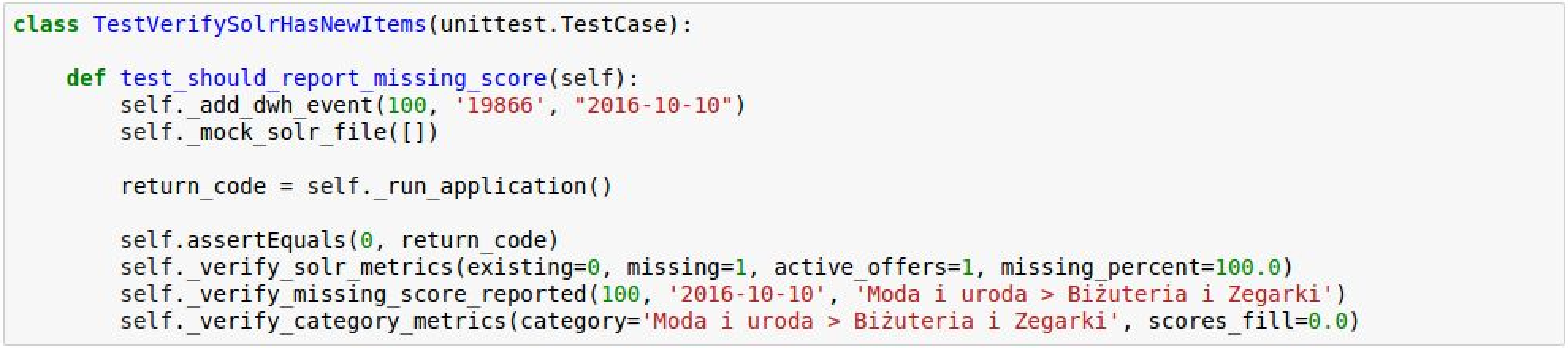
```python
@pytest.fixture(scope="session")
def sc(request):
    conf = (SparkConf()
            .setMaster("local[2]")
            .setAppName("pytest-pyspark-local-testing")
            .set("spark.default.parallelism",2)
            .set("spark.sql.shuffle.partitions",2)
            )
    sc = SparkContext(conf=conf)

    def fin():
        print ("teardown sc")
        sc.stop()
    request.addfinalizer(fin)

    return sc

@pytest.fixture(scope="session")
def sqlContext(sc):
    return SQLContext(sc)
```



```python
def test_sanity(sqlContext):
    row = {'name': 'Alice', 'age': 1}
    df = sqlContext.createDataFrame([row])

    collected = df.collect()

    collected[0].asDict() == row
```

Fitting in applications ecosystem

- Code review and git flow ✔
- Unit tests ✔
- Integration tests:
  - mocks data on local Hive and local HDFS and runs application inside docker container



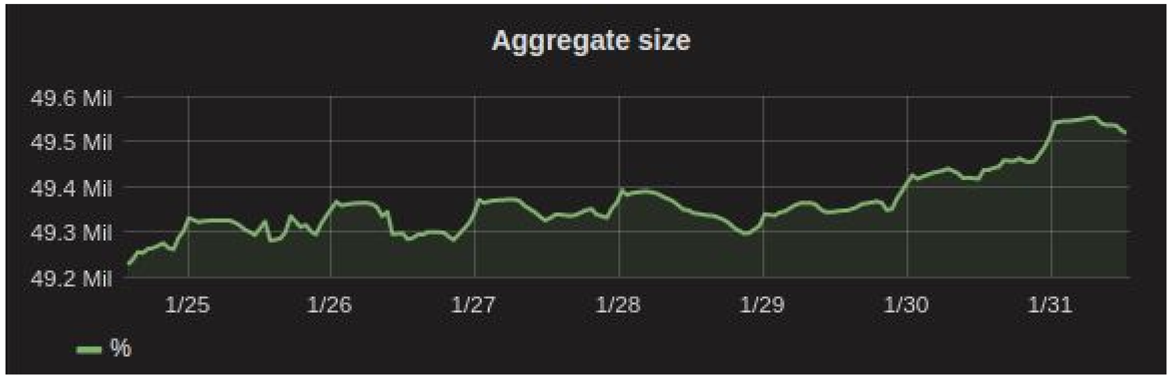
```
class TestVerifySolrHasNewItems(unittest.TestCase):

    def test_should_report_missing_score(self):
        self._add_dwh_event(100, '19866', "2016-10-10")
        self._mock_solr_file([])

        return_code = self._run_application()

        self.assertEquals(0, return_code)
        self._verify_solr_metrics(existing=0, missing=1, active_offers=1, missing_percent=100.0)
        self._verify_missing_score_reported(100, '2016-10-10', 'Moda i uroda > Biżuteria i Zegarki')
        self._verify_category_metrics(category='Moda i uroda > Biżuteria i Zegarki', scores_fill=0.0)
```

Fitting in applications ecosystem

- Code review and git flow ✔
- Unit tests ✔
- Integration tests ✔
- Continuous Integration
  - tests are performed after every commit to repository
  - pull request is blocked till tests succeed

Fitting in applications ecosystem

- Code review and git flow ✔
- Unit tests ✔
- Integration tests ✔
- Continuous Integration ✔
- Metrics
  - stored in Graphite, visualized in Grafana

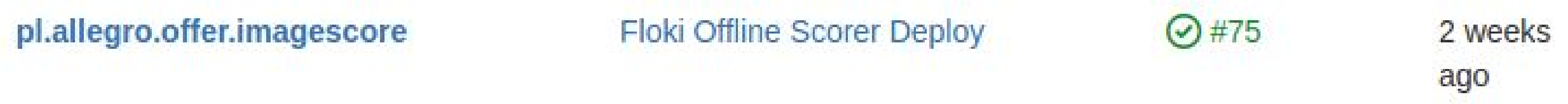


Fitting in applications ecosystem

- Code review and git flow ✔
- Unit tests ✔
- Integration tests ✔
- Continuous Integration ✔
- Metrics ✔
- Monitoring
  - conditions on Graphite metrics trigger incidents and alerts

Fitting in applications ecosystem

- Code review and git flow ✔
- Unit tests ✔
- Integration tests ✔
- Continuous Integration ✔
- Metrics ✔
- Monitoring ✔
- Deployment
  - Bamboo uploads .py files to HDFS after merge to master



pl.allegro.offer.imagescore	Floki Offline Scorer Deploy	✓ #75	2 weeks ago

Scheduling

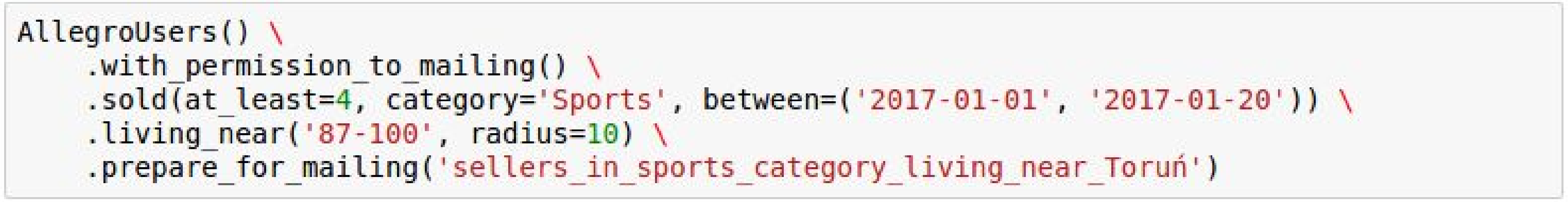
- Apache **Oozie**
  - very easy to use using **Hue UI**
  - **cannot switch** efficiently between **Spark versions**
  - **unreliable** execution and SLA monitoring
  - **poor Spark** jobs **maintaining**

Scheduling

- Apache **Oozie**
  - very easy to use using **Hue UI**
  - **cannot switch** efficiently between **Spark versions**
  - **unreliable** execution and SLA monitoring
  - **poor Spark** jobs **maintaining**
- Apache **Airflow**
  - matches almost **all our needs**
  - extended using **SparkOperator** https://git.io/airflow-spark-operator

Jupyter for non-technical staff

- **Domain Specific Language** that joins data from multiple sources and simplifies access



```
AllegroUsers() \
    .with_permission_to_mailing() \
    .sold(at_least=4, category='Sports', between=('2017-01-01', '2017-01-20')) \
    .living_near('87-100', radius=10) \
    .prepare_for_mailing('sellers_in_sports_category_living_near_Toruń')
```

Issues

- **Jupyter lacks git**/code review **support**
  - ipynb↔py "continuous conversion"
- **Shared RAM** for spark drivers on jupyter servers
- **Airflow** cannot handle **multitenancy**
- **YARN logs** of Spark apps are hard to debug
- Unable to **share notebooks** between users

Thank you!

- **Questions time!**