Start Treating your Data Pipelines as Code

Nadav Har Tzvi
@pythonesta
nadavha@apache.org

APACHE Amaterasu

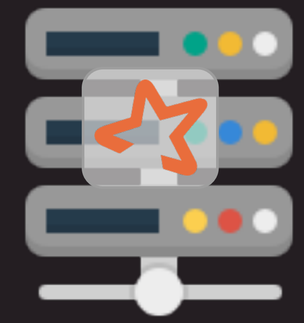
Most Big Data Projects are Failing

What's gone wrong?

Data Dependencies - Data Changes System Behaviour
Different Development Environments
Project Complexity

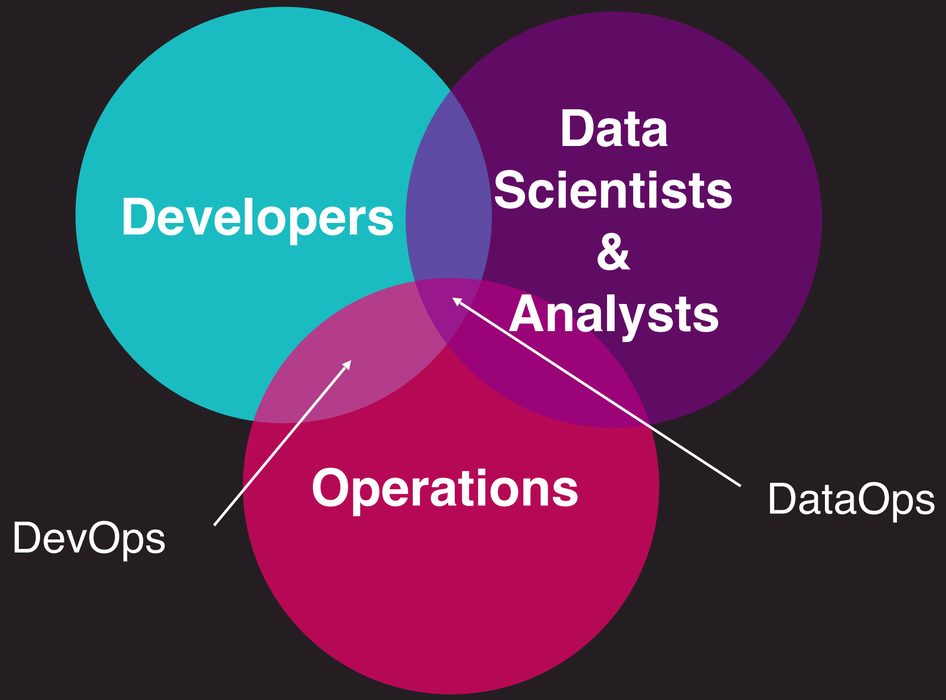
model.py

np.random.rand(...)
...



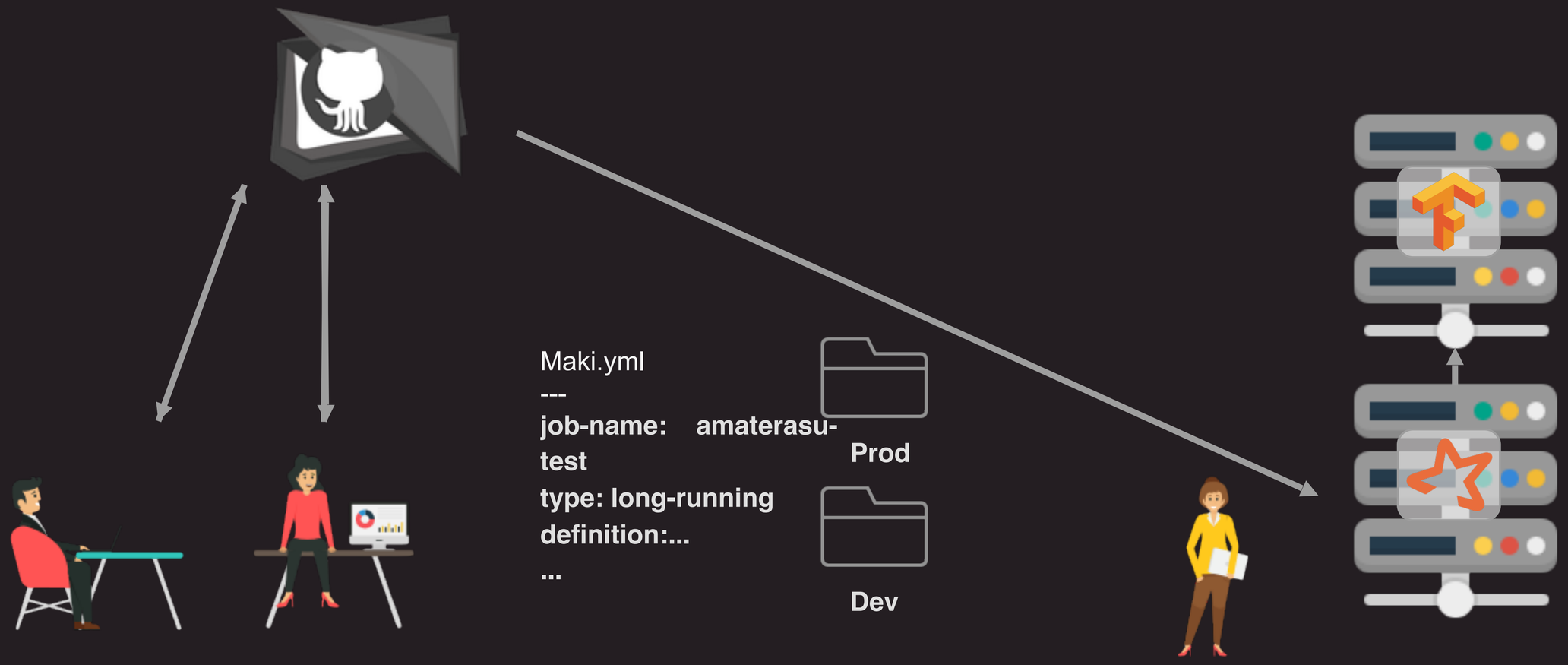
$$$ $$$ $$$ $$$ $$$ $$$ $$$ $$$ $$$ $$$ $$$ $$$

How we Fix it - DataOps



Apache Amaterasu (incubating) is a modern configuration management and deployment framework built in the spirit of tools such as Chef and Ansible
Designed for Big Data and AI processing pipelines

Frictionless Big Data Deployments with Amaterasu



Amaterasu Repositories

- Jobs are defined in repositories
  - Current implementation - git repositories
  - tarballs support is planned for future release
- Repos structure
  - **maki.yml -** The workflow definition
  - **src -** a folder containing the actions (spark scripts, etc.) to be executed
  - **env -** a folder containing configuration per environment
  - **deps -** dependencies configuration
- Benefits of using git:
  - Tooling
  - Branching

Pipeline DSL - maki.yml (Version 0.2.0)

```
---
job-name: amaterasu-test
flow:
  - name: start
    runner:
        group: spark
        type: scala
    file: file.scala
    - exports:
      odd: parquet
  - name: step2
      runner:
       group: spark
       type: pyspark
    file: file2.py
    error:
      name: handle-error
      runner:
          group: spark
          type: scala
      file: cleanup.scala
...
```


Actions are components of the pipeline

Data-structures to be used in downstream actions

Error handling actions

APACHE
Amaterasu

Amaterasu is **not** a workflow engine,
it's a deployment tool that understands that Big Data applications are rarely deployed independently of other Big Data applications

APACHE
Amaterasu

Pipeline != Workflow

Pipeline DSL (Version 0.3.0)

```
---
job-name: amaterasu-test
type: long-running
def:
   - name: start
     type: long-running
     runner:
         group: spark
         type: scala
     file: file.scala
     - exports:
       odd: parquet
  - name: step2
        type: scheduled
        schedule: 10 * * * *
        runner:
         group: spark
         type: pyspark
       artifact:
   - groupid: io.shonto
     artifactId: mySparkStreaming
     version: 0.1.0
…
```

In Version 3 Pipeline and actions can be either long running or scheduled

Scheduling is defined using Cron format

Actions can be pulled from other application or git repositories

Actions DSL (Spark)

- Your Scala/Python/SQL Future languages Spark code (R support is in our backlog)
- Few changes:
  - Don’t create a new sc/sqlContext, use the one in scope or access via **AmaContext.spark** **AmaContext.sc** and **AmaContext.sqlContext**
  - **AmaContext.getDataFrame** is used to access data from previously executed actions

Actions DSL - Spark Scala

Action 1 (“start”)

```
import org.apache.amaterasu.runtime._

val data = Array(1, 2, 3, 4, 5)

val rdd = AmaContext.sc.parallelize(data)
val odd = rdd.filter(n => n%2 !=
0).toDf()
```

```
- name: start
  runner:
      group: spark
      type: scala
  file: file.scala
  - exports:
    odd: parquet
```

Action 2

```
import org.apache.amaterasu.runtime._

val highNoDf = AmaContext.getDataFrame("start",
“odd")
                .where(“_1 > 3")

highNoDf.write.json("file:///tmp/test1")
```

Actions DSL - PySpark

Action 1 (“start”)

```
data = range(1, 1000)

rdd = ama_context.sc.parallelize(data)
odd = rdd.filter(lambda n:  n % 2 != 0)
        .map(row)
        .toDf()
```

```
- name: start
  runner:
      group: spark
      type: pyspark
  file: file.py
  - exports:
    odd: parquet
```

Action 2

```
high_no_df = ama_context
        .get_dataframe(“start”, “odd")
                .where("_1 > 100”)
high_no_df.write.save(“file:///tmp/test1”,
format=“json”)
```

APACHE Amaterasu

Actions DSL - SparkSQL

```
select * from
    ama_context.start_odd
where
    _1 > 100
```

```
- name: acttion2
  runner:
      group: spark
      type: sql
  file: file.sql
- exports:
  high_no: parquet
```

APACHE Amaterasu

Environments

- Configuration is stored per environment
- Stored as YAML files in an environment folder
- Contains:
  - Input/output path
  - Work dir
  - User defined key-values

APACHE Amaterasu

env/prduction/job.yml

```
name: default
inputRootPath: hdfs://prdhdfs:9000/user/amaterasu/input
outputRootPath: hdfs://prdhdfs:9000/user/amaterasu/
output
workingDir: hdfs://prdhdfs:9000/user/ama_workdir
configuration:
    spark.cassandra.connection.host: cassandraprod
    sourceTable: documents
```

APACHE Amaterasu

env/dev/job.yml

```
name: test
inputRootPath: file:///tmp/input
outputRootPath: file:///tmp/output
workingDir: file:///tmp/work/
configuration:
    spark.cassandra.connection.host: 127.0.0.1
    sourceTable: documents
```

APACHE Amaterasu

Environments in the Actions DSL

```
import io.shinto.amaterasu.runtime._

val highNoDf = AmaContext.getDataFrame("start", “x”)
                         .where("_1 > 3”)

highNoDf.write.json(Env.outputPath)
```

APACHE Amaterasu

Version 0.2.0-incubating main futures

- YARN support
- Spark SQL, PySpark support
- Extend environments to support:
  - Pure YAML support (configuration used to be JSON)
  - Full spark configuration
    - **spark.yml** - support all spark configurations
    - **spark_exec_env.yml** - for configuring spark executors environments
- SDK Preview - for building framework integration

APACHE Amaterasu

Future Development

- Long running pipelines and streaming support
- Kubernetes support
- Better tooling
  - ama-cli (version 0.2.1)
  - Web console
- Other frameworks: TensorFlow, Apache Flink, Apache Beam, Hive, Presto…
- SDK improvements

Getting started

Website
http://amaterasu.incubator.apache.org

GitHub
https://github.com/apache/incubator-amaterasu

Mailing List
dev@amaterasu.incubator.apache.org

Slack
http://apacheamaterasu.slack.com

Twitter
@ApacheAmaterasu

APACHE
Amaterasu

Thank you!

@pythonesta
nadavha@apache.org