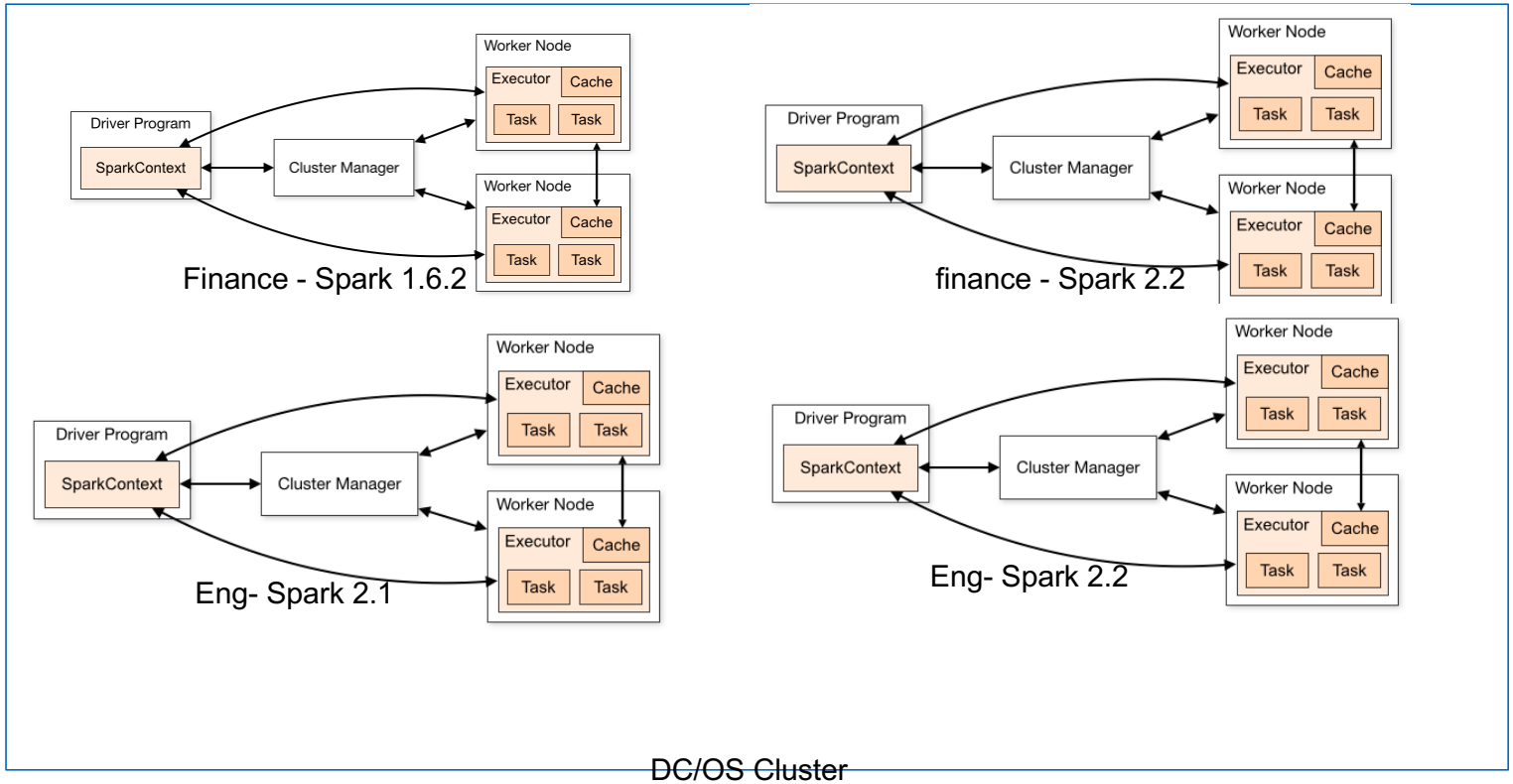
Spark Data Processing and DC/OS

Part I: Getting started with Spark and exploring a sampling of data tools

Presented by Mark Johnson

Spark on DC/OS Overview



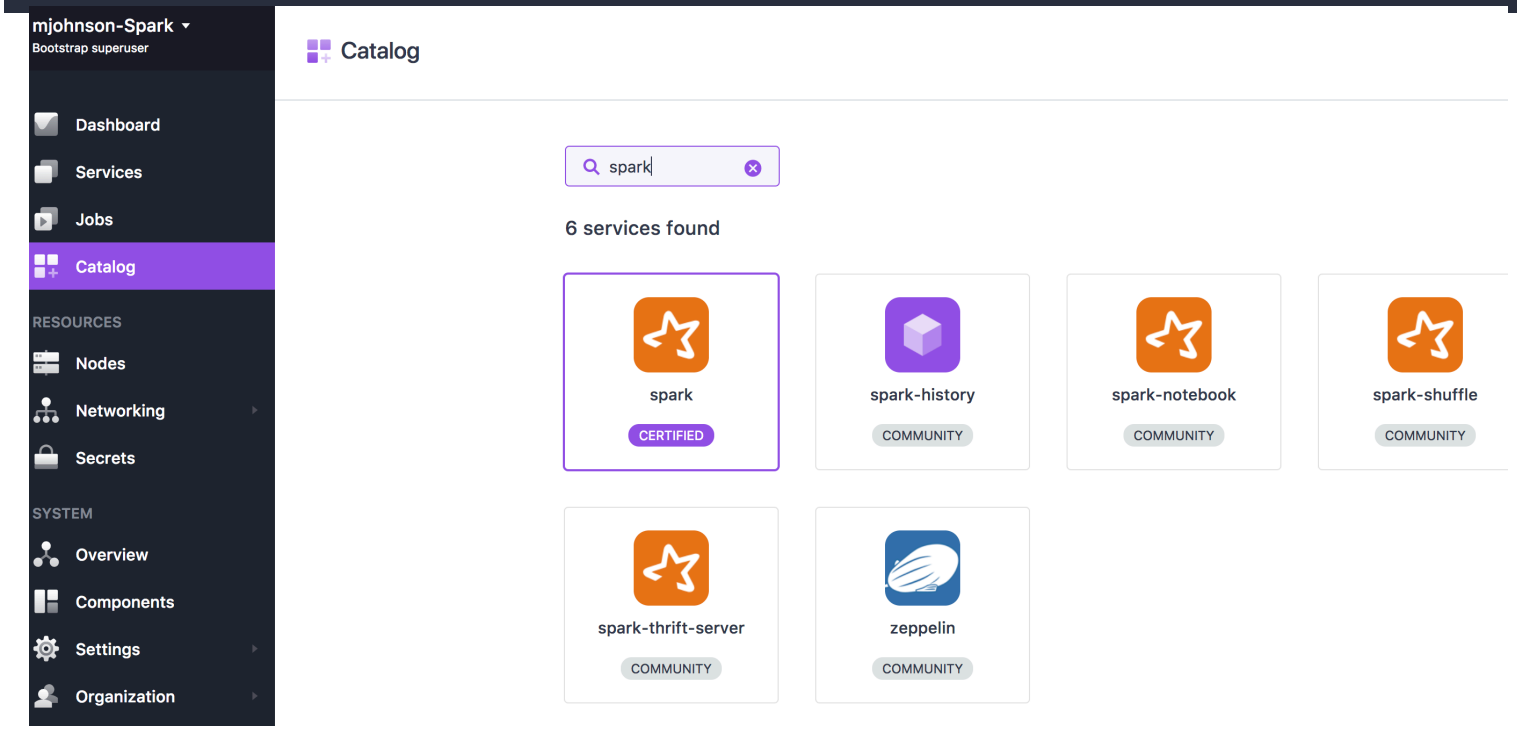
- Support polygot database analytics
- In memory emphasis improves query performance
- Compatible with wide range of tools

Agenda

1. Installing and configuring Spark on DC/OS
2. Running Spark Programs
3. Debugging and Monitoring
4. Spark Data Access
5. Spark Data Notebooks

Part I: Spark Installation & Setup

GUI Based Spark installation



Catalog > spark

spark

CERTIFIED 2.0.1-2.2.0-1

CONFIGURE DEPLOY

By deploying you agree to the terms and conditions

Description

Spark is a fast and general cluster computing system for Big Data. Documentation: https://docs.mesosphere.com/service-docs/spark/

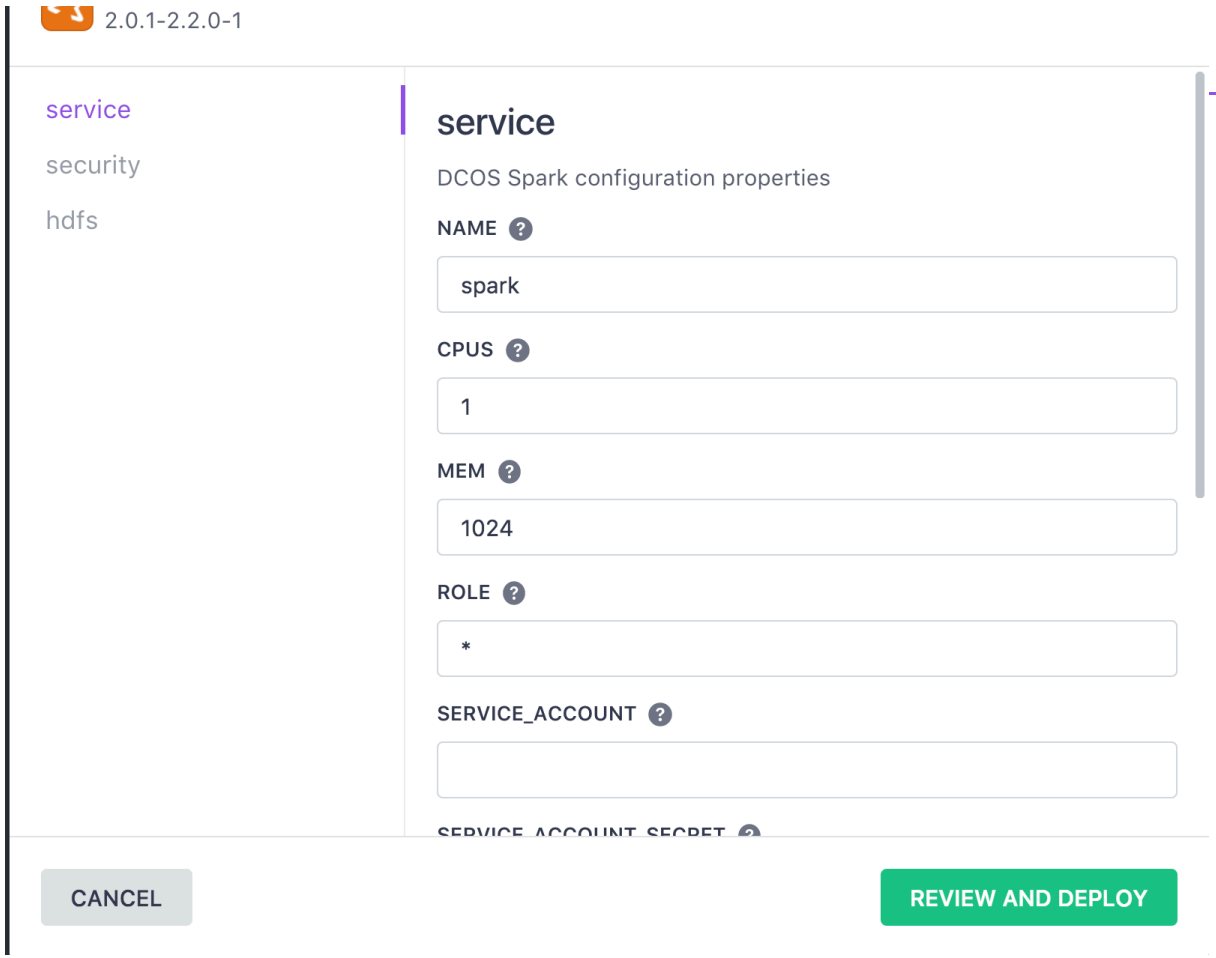
Information

SCM: https://github.com/apache/spark.git

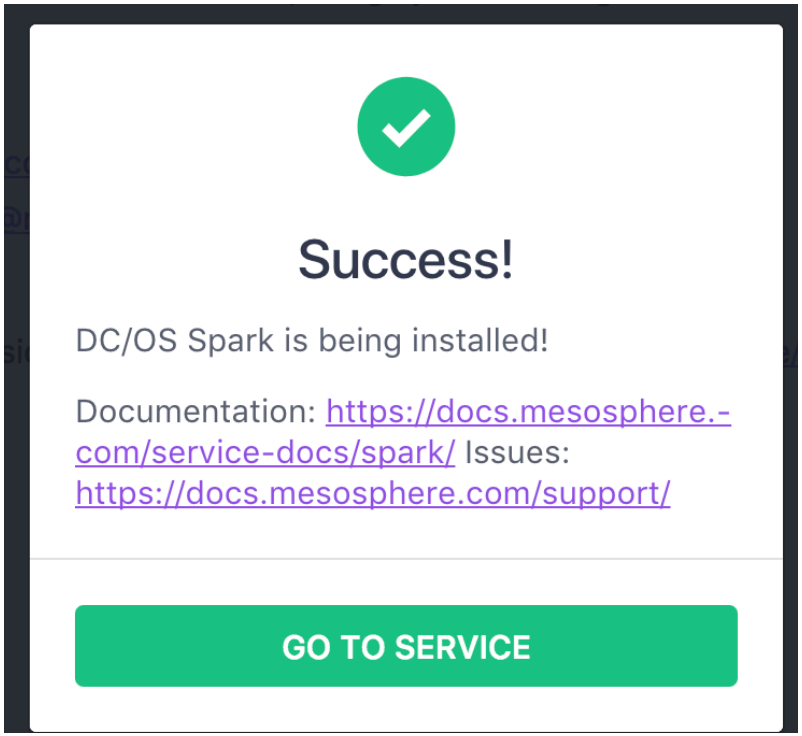
Maintainer: support@mesosphere.io

Licenses

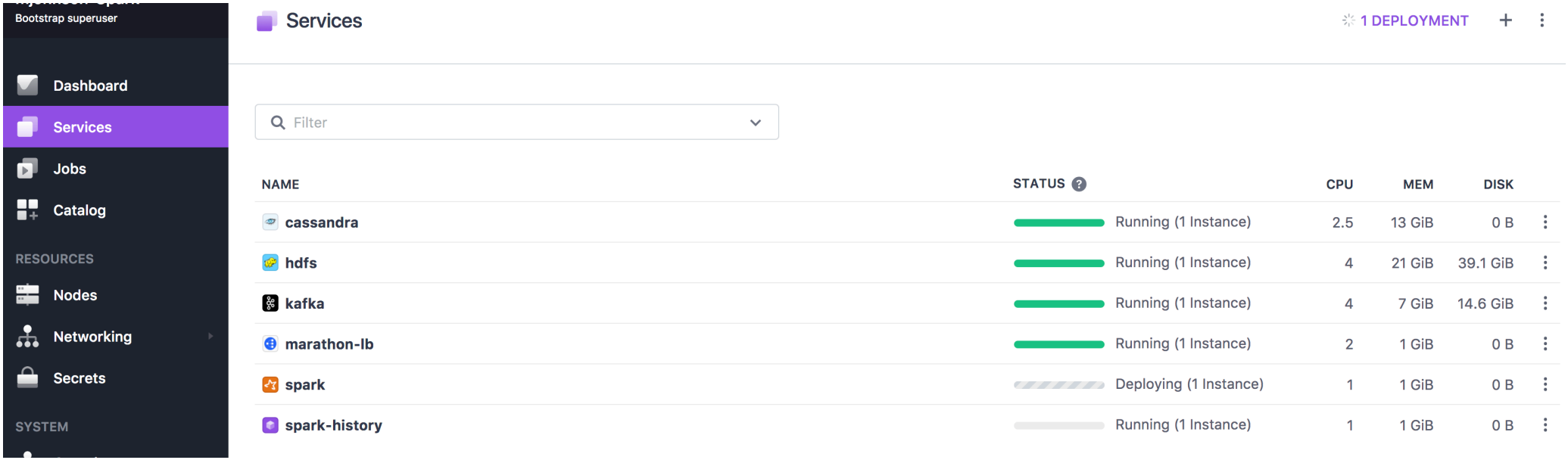
Apache License Version 2.0: https://raw.githubusercontent.com/apache/spark/master/LICENSE



MESOSPHERE



MESOSPHERE



MESOSPHERE

Command Line Spark Installation

- `dcos package install spark --app-id=/production`
- `dcos package install spark --package-version=1.0.1-1.6.2 --app- id=/spark162`
- `dcos package install --yes spark --options=spark-dispatcher-options.json`

DC/OS Spark Command Line Options

- dcos spark run …
- dcos spark status {submission id}
- dcos spark log {submission id}
- dcos spark kill {submission id}
- dcos webui

Part II: Running Spark Programs

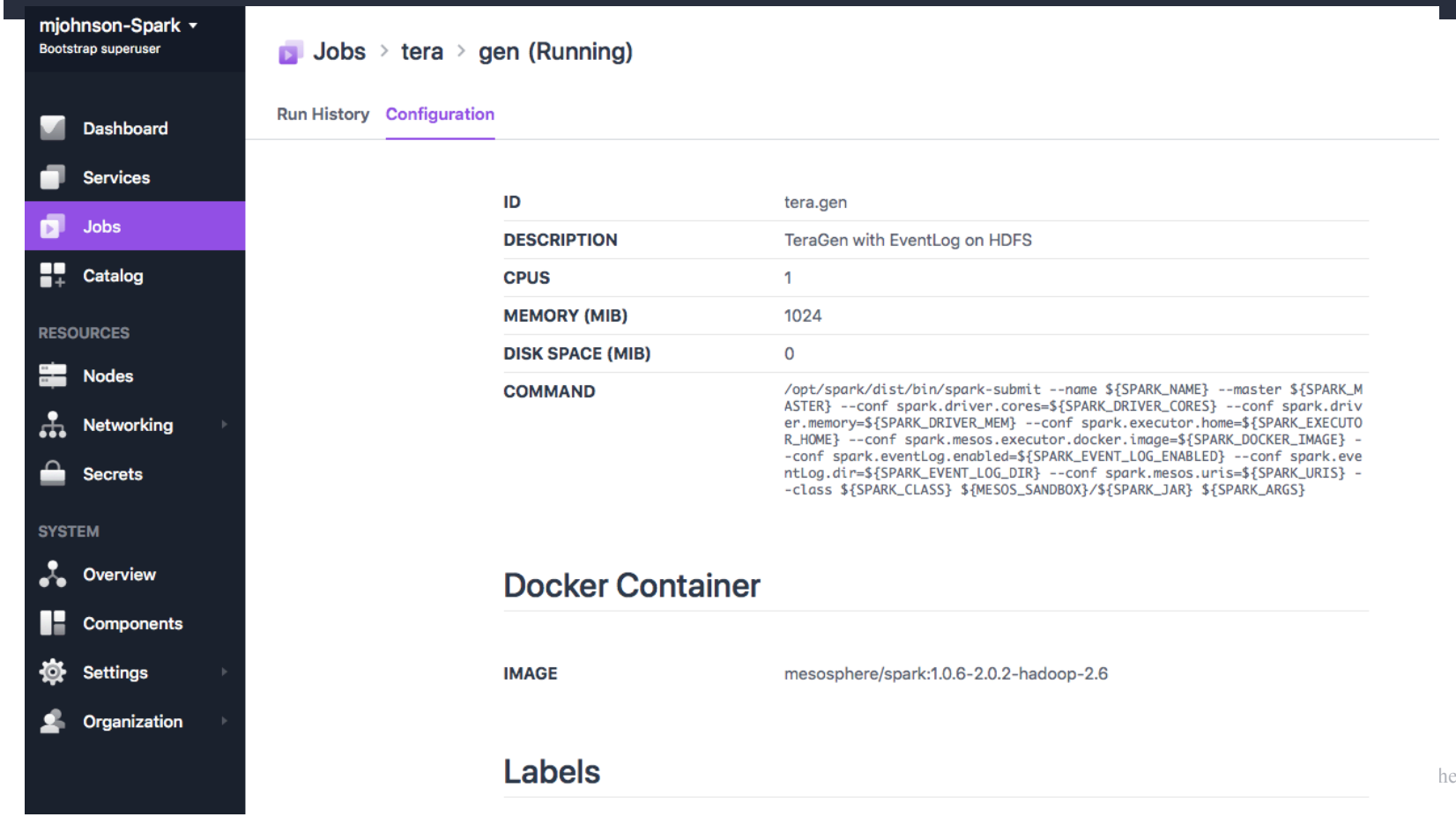
Run Spark from the command line

- `dcos spark run --submit-args="--conf` **`spark.mesos.executor.docker.image=mesosphere/spark:1.1.0-2.1.1-hadoop-2.6`** `https://downloads.mesosphere.com/spark/examples/pi.py 30" --verbose`

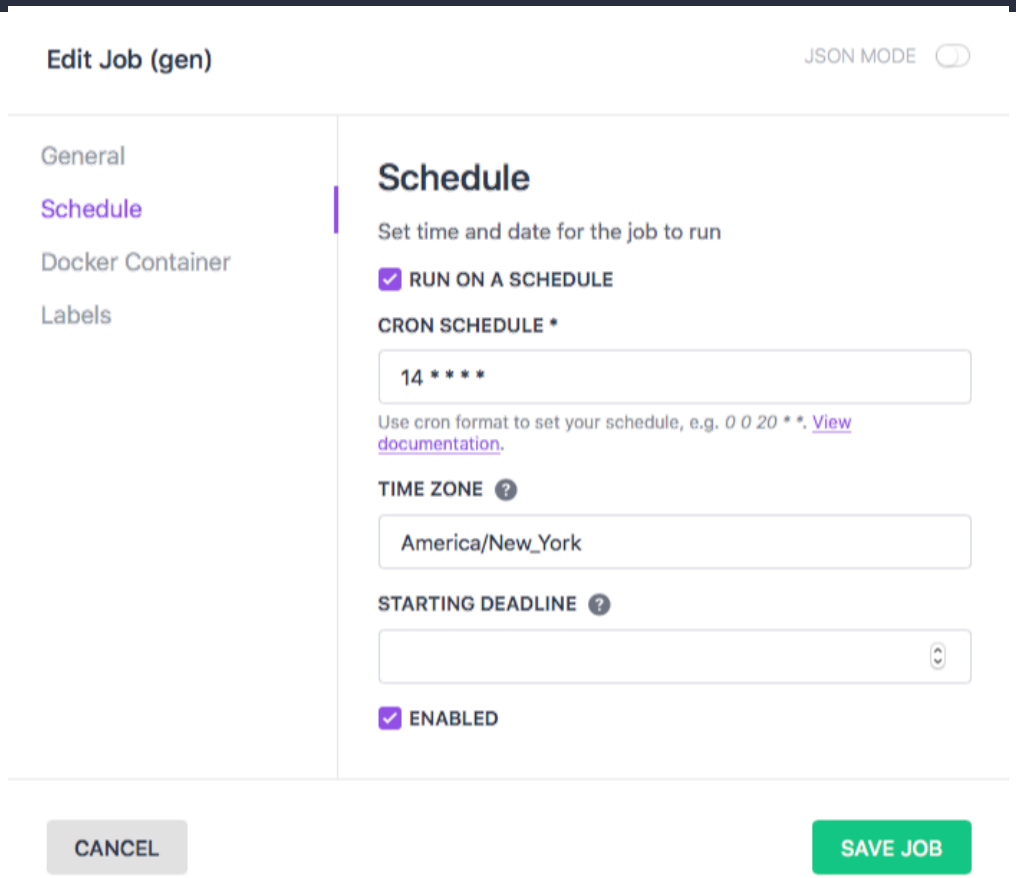
Run Spark from the command line

- `dcos spark run --submit-args="--conf spark.mesos.executor.docker.image=mesosphere/spark:1.1.0-2.1.1-hadoop-2.6 https://downloads.mesosphere.com/spark/examples/pi.py 30" --verbose`

DC/OS Jobs



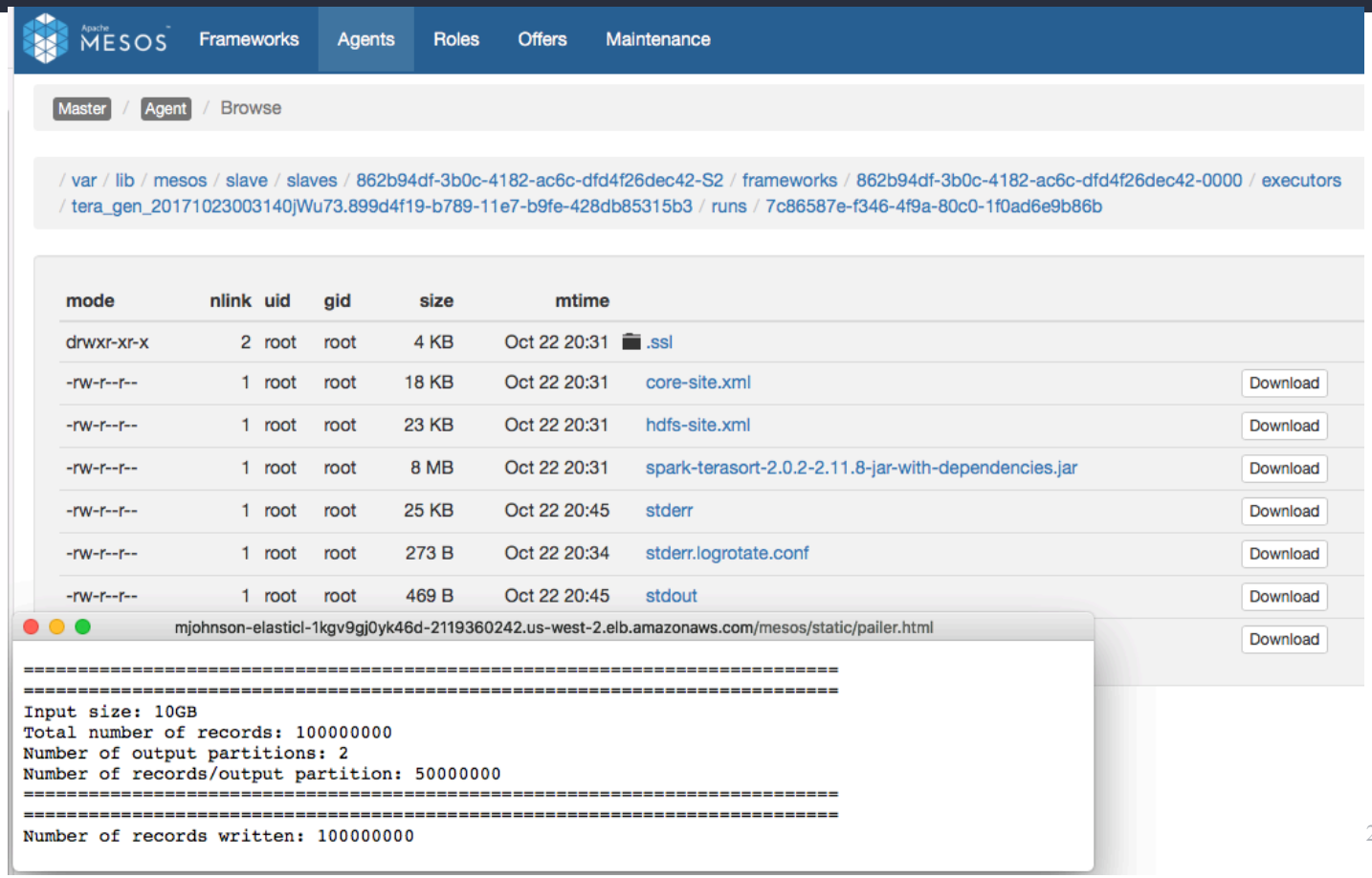
Run every 15 minutes



DC/OS Jobs

```
{
  "description": "TeraGen with EventLog on HDFS",
  "id": "tera.gen",
  "run": {
    "cpus": 1,
    "mem": 1024,
    "disk": 0,
    "user": "root",
    "cmd": "/opt/spark/dist/bin/spark-submit --name ${SPARK_NAME} --master ${SPARK_MASTER} --conf spark.driver.cores=${SPARK_DRIVER_CORES}
  --conf spark.driver.memory=${SPARK_DRIVER_MEM} --conf spark.executor.home=${SPARK_EXECUTOR_HOME} --conf spark.mesos.executor.docker
  .image=${SPARK_DOCKER_IMAGE} --conf spark.eventLog.enabled=${SPARK_EVENT_LOG_ENABLED} --conf spark.eventLog.dir=${SPARK_EVENT_LOG_DIR}
  --conf spark.mesos.uris=${SPARK_URIS} --class ${SPARK_CLASS} ${MESOS_SANDBOX}/${SPARK_JAR} ${SPARK_ARGS}",
    "env": {
      "SPARK_NAME": "TeraGen",
      "SPARK_DOCKER_IMAGE": "mesosphere/spark:1.0.6-2.0.2-hadoop-2.6",
      "SPARK_MASTER": "mesos://zk://zk-1.zk:2181,zk-2.zk:2181,zk-3.zk:2181,zk-4.zk:2181,zk-5.zk:2181/mesos",
      "SPARK_CORES_MAX": "10",
      "SPARK_DRIVER_CORES": "1",
      "SPARK_DRIVER_MEM": "4g",
      "SPARK_EXECUTOR_CORES": "2",
      "SPARK_EXECUTOR_MEM": "6g",
      "SPARK_EXECUTOR_HOME": "/opt/spark/dist",
      "SPARK_EVENT_LOG_ENABLED": "true",
      "SPARK_EVENT_LOG_DIR": "hdfs://hdfs/history",
      "SPARK_URIS": "http://api.hdfs.marathon.l4lb.thisdcos.directory/v1/endpoints/hdfs-site.xml,http://api.hdfs.marathon.l4lb.thisdcos
  .directory/v1/endpoints/core-site.xml,https://s3.amazonaws.com/vishnu-mohan/spark-terasort-2.0.2-2.11.8-jar-with-dependencies.jar",
      "SPARK_CLASS": "com.github.ehiggs.spark.terasort.TeraGen",
      "SPARK_JAR": "spark-terasort-2.0.2-2.11.8-jar-with-dependencies.jar",
      "SPARK_ARGS": "10G hdfs://hdfs/terasort/terasort_in"
    },
    "docker": {
      "image": "mesosphere/spark:1.0.6-2.0.2-hadoop-2.6"
    },
    "artifacts": [
      {"uri": "http://api.hdfs.marathon.l4lb.thisdcos.directory/v1/endpoints/hdfs-site.xml"...},
      {"uri": "http://api.hdfs.marathon.l4lb.thisdcos.directory/v1/endpoints/core-site.xml"...},
      {"uri": "https://s3.amazonaws.com/vishnu-mohan/spark-terasort-2.0.2-2.11.8-jar-with-dependencies.jar"...}
    ]
  },
  "labels": {"location": "olympus"...}
  "labels": {
    "location": "olympus",
    "owner": "zeus"
```

Mesos Sandbox : Job Output



Park III: Spark Debugging and Monitoring

Mesos Task Screen

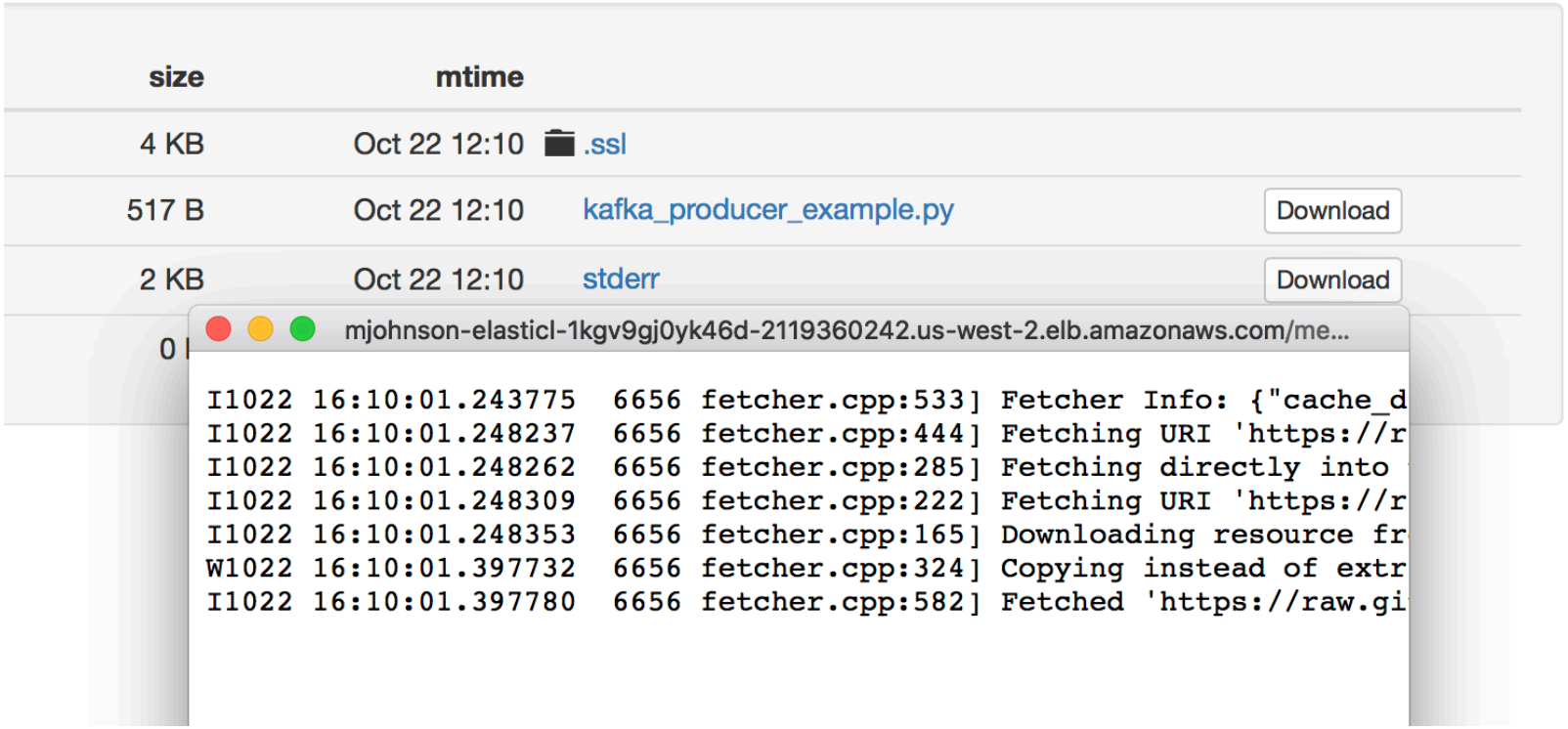
Completed Tasks

Find...

Framework ID	Task ID	Task Name	Role	State	Started ▼	Stopped	Host	
862b94df-3b0c-4182-ac6c-dfd4f26dec42-0005	driver-20171022161000-0001	Driver for kafka_producer_example.py	*	KILLED		5 minutes ago	10.0.1.138	Sandbox

Mesos Sandbox

Mesos Sandbox: Viewing Logs



View Sandbox on Host

```
~/demos/spark/spark_demo/kafka_spark >dcos node ssh --master-proxy --private-ip=10.0.1.138
Running `ssh -A -t core@35.165.198.240 ssh -A -t core@10.0.1.138 `
The authenticity of host '10.0.1.138 (10.0.1.138)' can't be established.
ECDSA key fingerprint is SHA256:imY1GxlD87E9Mu0aNGnF4u1kfWF1Fb74jRPLujOXlAg.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.1.138' (ECDSA) to the list of known hosts.
Container Linux by CoreOS stable (1235.12.0)
Update Strategy: No Reboots
Failed Units: 1
  update-engine.service
core@ip-10-0-1-138 ~ $ ls -lt /var/lib/mesos/slave/slaves/862b94df-3b0c-4182-ac6c-dfd4f26dec42-S7/frameworks/862b94df-3b0c
-4182-ac6c-dfd4f26dec42-0005/executors/driver-20171022161000-0001/runs/d46639c6-d67a-4f7f-9be4-e370e2fcfd61/
total 8
-rw-r--r--. 1 root root  517 Oct 22 16:10 kafka_producer_example.py
-rw-r--r--. 1 root root 2050 Oct 22 16:10 stderr
-rw-r--r--. 1 root root    0 Oct 22 16:10 stdout
core@ip-10-0-1-138 ~ $
```

SSH TUNNELS - Spark Program Debugging

- Ssh –N –L {desired port}:{DC/OS VIP or DNS}:{Endpoint Port} {user id}{dc/os url}
- Use '`dcos {framework} endpoints {desired endpoint}`' to identify the VIP or DNS options.
- Change your source program to reference 'localhost:{desired port}'

```
~/dcos_scripts >dcos beta-kafka --name=kafka endpoints
[
  "broker",
  "zookeeper"
]
~/dcos_scripts >dcos beta-kafka --name=kafka endpoints broker
{
  "address": [
    "10.0.2.60:1025",
    "10.0.0.96:1025",
    "10.0.2.205:1025"
  ],
  "dns": [
    "kafka-0-broker.kafka.autoip.dcos.thisdcos.directory:1025",
    "kafka-1-broker.kafka.autoip.dcos.thisdcos.directory:1025",
    "kafka-2-broker.kafka.autoip.dcos.thisdcos.directory:1025"
  ],
  "vip": "broker.kafka.l4lb.thisdcos.directory:9092"
}
```

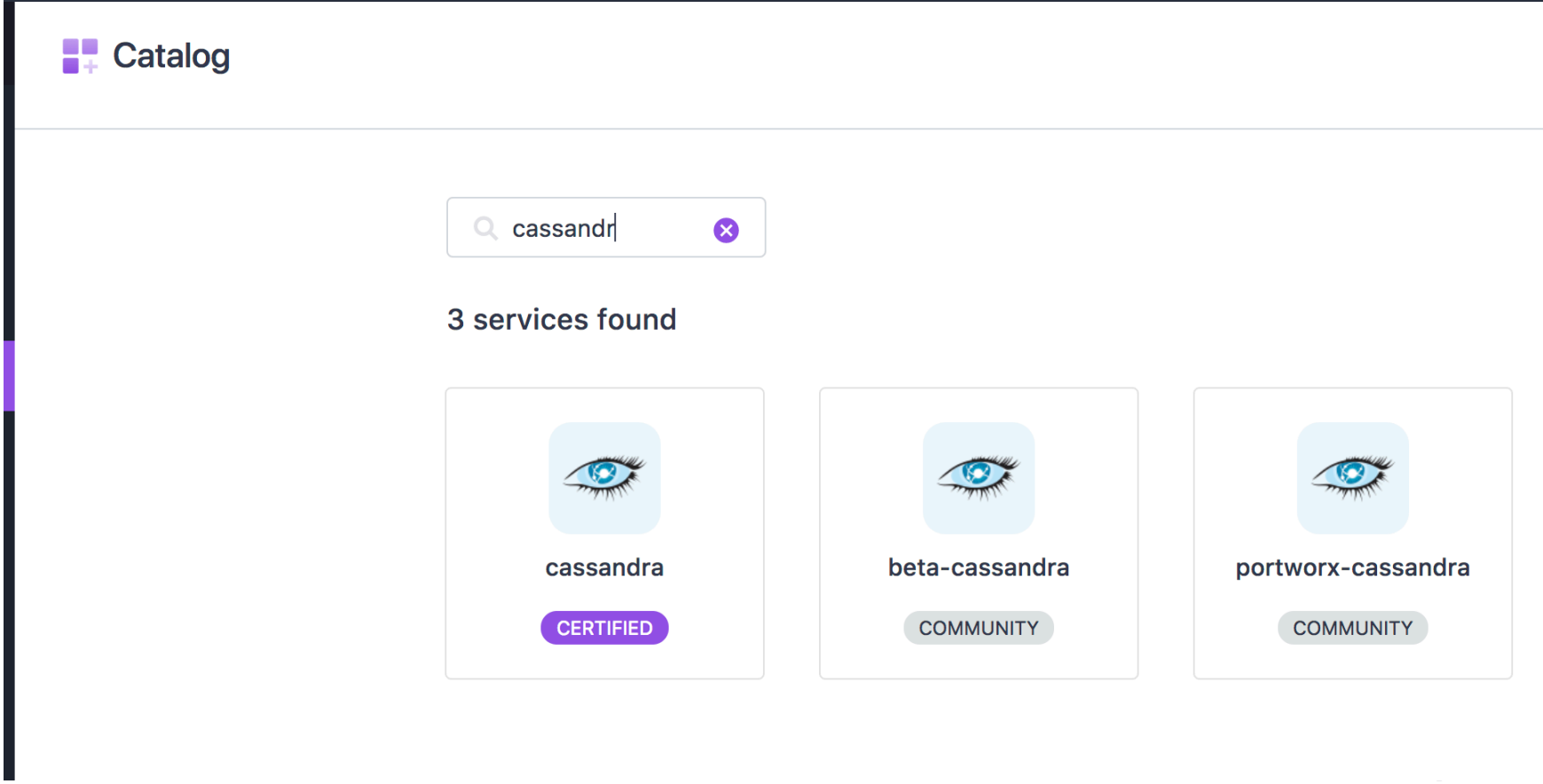
```
~/demos/spark/spark_demo/kafka_spark >ssh -N -L 9092:broker.kafka.l4lb.thisdcos.directory:9092 core@35.165.198.240
```

```
from kafka import KafkaConsumer, KafkaProducer
from kafka.errors import KafkaError

#kafka_url = "broker.kafka.l4lb.thisdcos.directory:9092"
kafka_url = "localhost:9092"
topic_name = "example_topic"


def readTopic():
    consumer = KafkaConsumer(bootstrap_servers=kafka_url,
                             auto_offset_reset='earliest',
                             group_id="PrimaryGroup",
                             client_id="client-1",
                             enable_auto_commit=True, api_version=(0,10))
```

Spark History

Part IV: Data Access with Spark

Spark & Cassandra



Cassandra DC/OS Addresses

```
~/demos/spark/spark_demo/cassandra_spark >dcos cassandra endpoints node
{
  "address": [
    "10.0.3.244:9042",
    "10.0.3.150:9042",
    "10.0.2.205:9042"
  ],
  "dns": [
    "node-0-server.cassandra.autoip.dcos.thisdcos.directory:9042",
    "node-1-server.cassandra.autoip.dcos.thisdcos.directory:9042",
    "node-2-server.cassandra.autoip.dcos.thisdcos.directory:9042"
  ],
  "vip": "node.cassandra.l4lb.thisdcos.directory:9042"
}
```

Cassandra SQL Execute Example

```
from cassandra.cluster import Cluster
from pyspark import SparkContext, SparkConf, SQLContext
from pyspark.sql import SparkSession

cassandra_url = 'node.cassandra.l4lb.thisdcos.directory:9042'
#cassandra_url = 'localhost'

spark = SparkSession.builder \
    .appName("PySpark Cassandra Read Write Example") \
    .config("spark.sql.crossJoin.enabled", "true") \
    .getOrCreate()


cluster = Cluster([cassandra_url])
session = cluster.connect()

session.execute("""CREATE KEYSPACE IF NOT EXISTS dcos_example WITH
    REPLICATION = { \'class\' : 'NetworkTopologyStrategy', 'datacenter1' : 3 }; """)

session.execute("USE dcos_example;")
session.execute("CREATE TABLE IF NOT EXISTS sample_table ( user_id timeuuid PRIMARY KEY,
added_date timestamp, first_name text, last_name text, email text);")
```

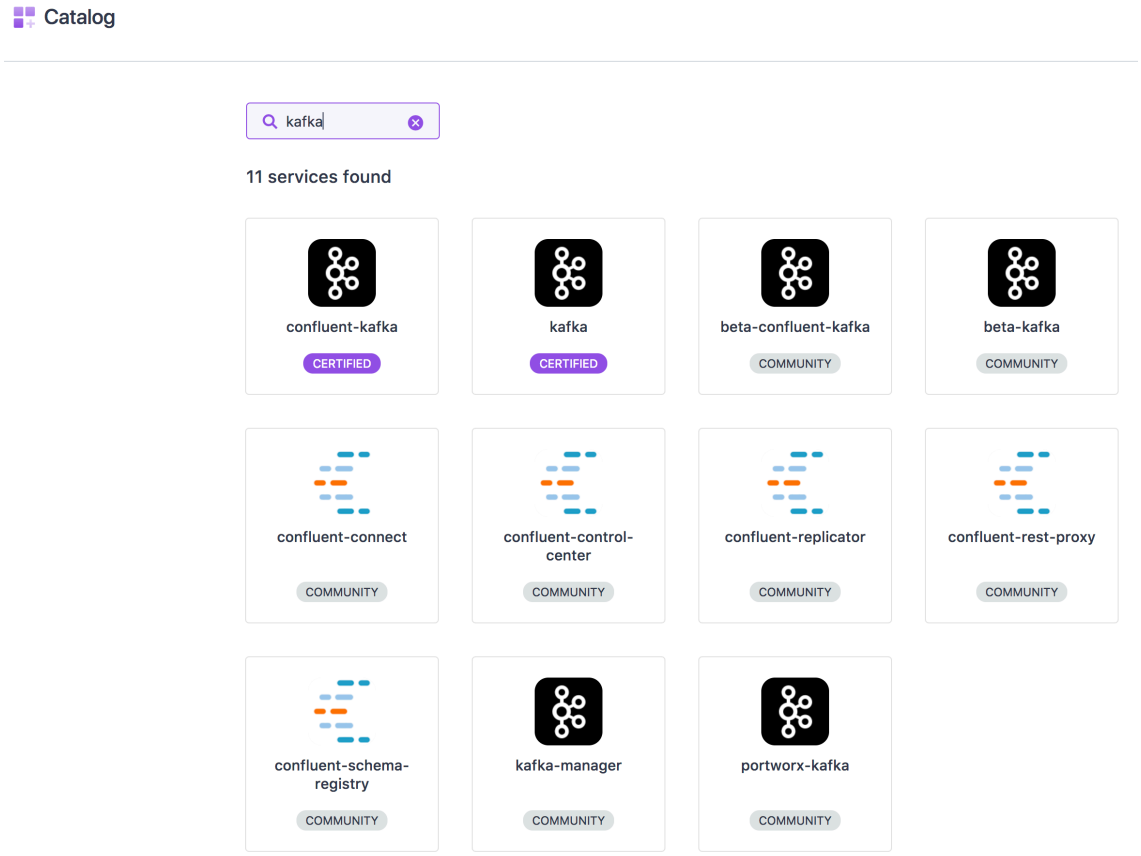
Spark tasks and driver

Active Tasks

Find...

Framework ID	Task ID	Task Name	Role	State	Started ▼	Host	
862b94df-3b0c-4182-ac6c-dfd4f26dec42-0005-driver-20171023032008-0028	2	Task 2	*	STAGING		10.0.1.103	Sandbox
862b94df-3b0c-4182-ac6c-dfd4f26dec42-0005-driver-20171023032008-0028	1	Task 1	*	STAGING		10.0.1.138	Sandbox
862b94df-3b0c-4182-ac6c-dfd4f26dec42-0005	driver-20171023032008-0028	Driver for Cassandra_rw.py	*	RUNNING	just now	10.0.2.60	Sandbox

Spark & Kafka



Identify Kafka Broker Endpoints

```
~/demos/spark/spark_demo/kafka_spark >dcos beta-kafka --name=kafka endpoints broker
{
  "address": [
    "10.0.2.60:1025",
    "10.0.0.96:1025",
    "10.0.2.205:1025"
  ],
  "dns": [
    "kafka-0-broker.kafka.autoip.dcos.thisdcos.directory:1025",
    "kafka-1-broker.kafka.autoip.dcos.thisdcos.directory:1025",
    "kafka-2-broker.kafka.autoip.dcos.thisdcos.directory:1025"
  ],
  "vip": "broker.kafka.l4lb.thisdcos.directory:9092"
}

~/demos/spark/spark_demo/kafka_spark >
```

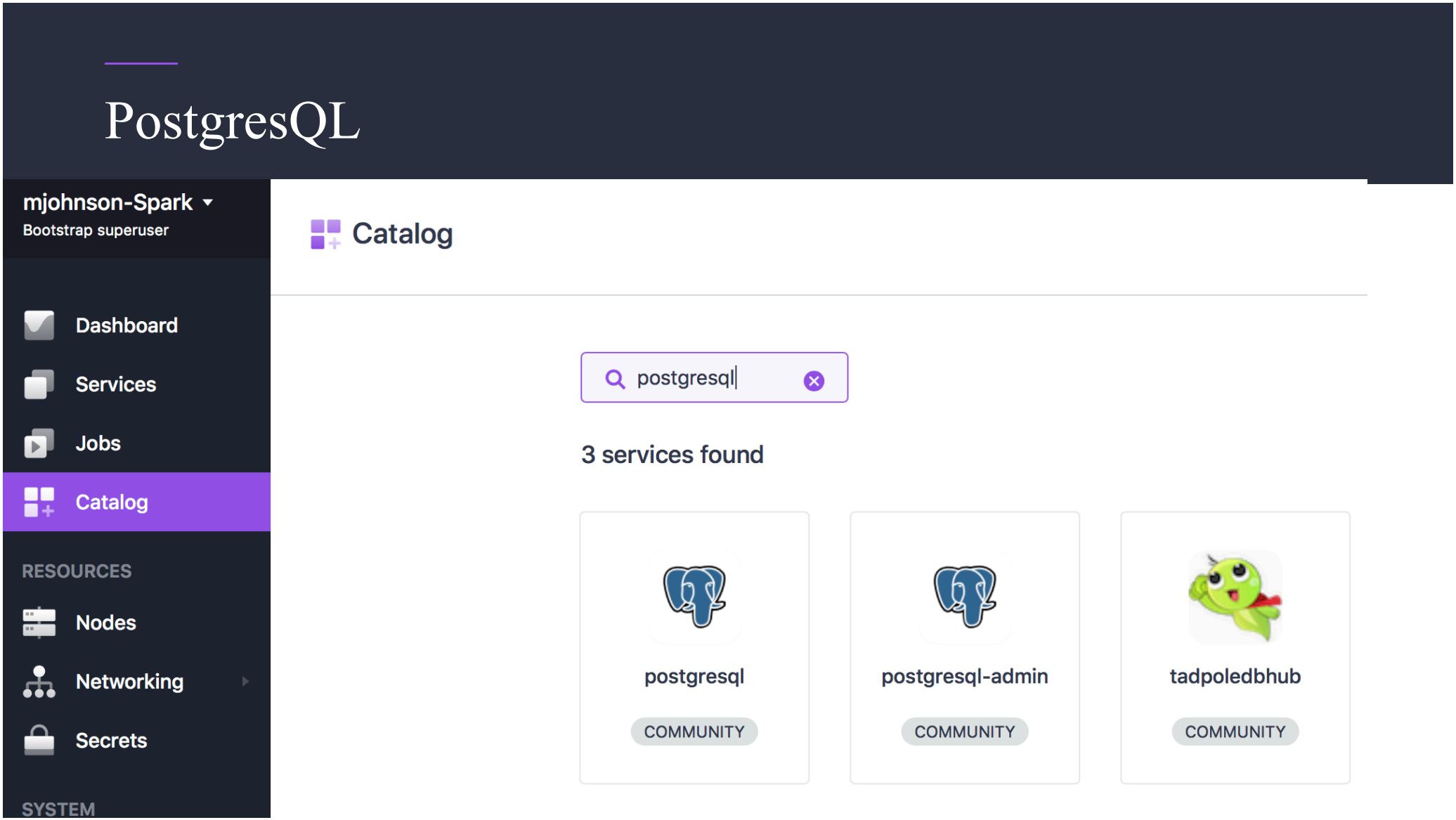
Kafka Producer

```
from kafka import KafkaProducer
import time

kafka_url = ["kafka-0-
broker.kafka.autoip.dcos.thisdcos.directory:1025","kafka-1-
broker.kafka.autoip.dcos.thisdcos.directory:1025", "kafka-2-
broker.kafka.autoip.dcos.thisdcos.directory:1025"]
topic_name = "example_topic"

p = KafkaProducer(bootstrap_servers=kafka_url,api_version=(0,10))

iter_count = 0

while True:
    msg = b"test message - {}".format(iter_count)
    print msg
    p.send(topic_name, msg)
    time.sleep(1)
    iter_count = iter_count + 1
```

Kafka Consumer

```
om kafka import KafkaConsumer

print("Start of Kafka consumer program")
kafka_url = "broker.kafka.l4lb.thisdcos.directory:9092"
#kafka_url = "localhost:9092"

topic_name = "example_topic"

print("Connecting to {}".format(kafka_url))
consumer = KafkaConsumer(topic_name, bootstrap_servers=kafka_url,
                         group_id="PrimaryGroup",
                         enable_auto_commit=True,
                         api_version=(0,10))
print ('Start consuming')
for message in consumer:
    print(message)
print("Done consuming")
```

PostgresQL


mjohnson-Spark
Bootstrap superuser
Dashboard
Services
Jobs
Catalog
RESOURCES
Nodes
Networking
Secrets
SYSTEM
Catalog
postgresql
3 services found
postgresql
COMMUNITY
postgresql-admin
COMMUNITY
tadpoledbhub
COMMUNITY

Catalog › postgresql

postgresql

CONFIGURE DEPLOY

COMMUNITY 9.6-0.2

By deploying you agree to the terms and conditions

Description

PostgreSQL is an object-relational database management system (ORDBMS) with an emphasis on extensibility and standards-compliance.

Pre-Install Notes

This DC/OS Service is currently in preview. There may be bugs, incomplete features, incorrect documentation, or other discrepancies.

`Advanced Installation options notes`

storage / *persistence*: create local persistent volumes for internal storage files to survive across restarts or failures.

storage / persistence / *external*: create external persistent volumes. This allows to use an external storage system such as Amazon EBS, OpenStack Cinder, EMC Isilon, EMC ScaleIO, EMC XtremIO, EMC VMAX and Google Compute Engine persistent storage. *NOTE*: To use external volumes with DC/OS, you MUST enable them during CLI or Advanced installation.

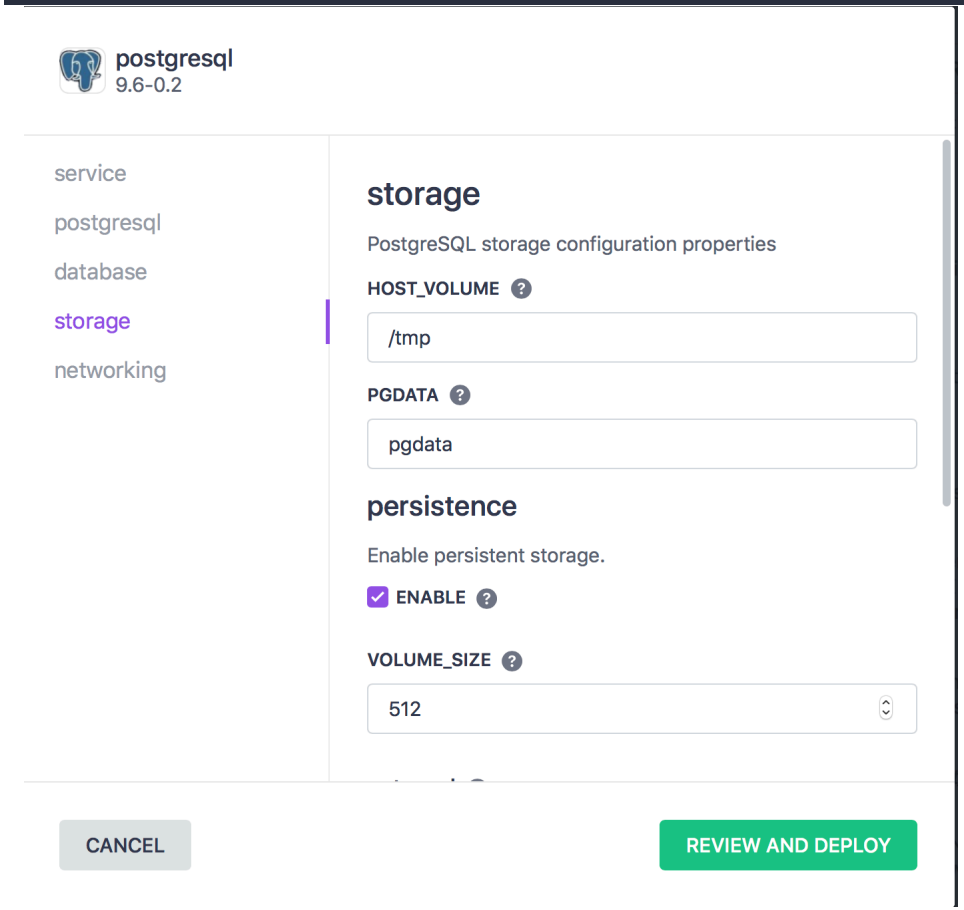
storage / *host_volume*: if persistence is not selected, this package can use a local volume in the host for storage, like a local directory or an NFS mount. The parameter *host_volume* controls the path in the host in which these volumes will be created, which MUST be the same on all nodes of the cluster.

NOTE: If you didn't select persistence in the storage section, or provided a valid value for *host_volume* on installation, YOUR DATA WILL NOT BE SAVED IN ANY WAY.

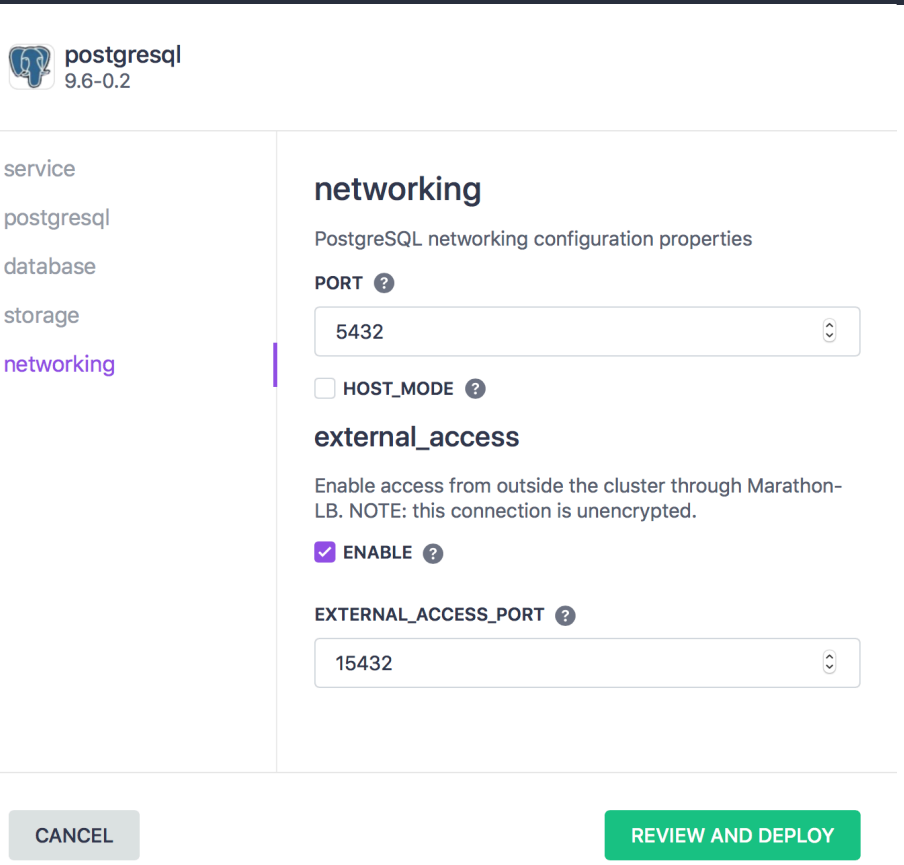
networking / *port*: This DC/OS service can be accessed from any other application through a NAMED VIP in the format `service_name.marathon.l4lb.thisdcos.directory:port` . Check status of the VIP in the *Network* tab of the DC/OS Dashboard (Enterprise DC/OS only).

networking / *external_access*: create an entry in Marathon-LB for accessing the service from outside of the cluster

PostgresQL - Storage



PostgresQL – Networking – EXTERNAL ACCESS



External access PostgresQL

HAProxy version 1.7.6, released 2017/06/16

Statistics Report for pid 27552

> General process information

pid = 27552 (process #1, nbproc = 1)
uptime = 0d 0h00m00s
system limits: memmax = unlimited; ulimit-n = 100036
maxsock = 100036; **maxconn =** 50000; **maxpipes =** 0
current conns = 1; current pipes = 0/0; conn rate = 1/sec
Running tasks: 1/10; idle = 100 %

- active UP
- active UP, going down
- active DOWN, going up
- active or backup DOWN
- active or backup DOWN for maintenance (MAINT)
- active or backup SOFT STOPPED for maintenance
- backup UP
- backup UP, going down
- backup DOWN, going up
- not checked

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:

- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:

- Primary site
- Updates (v1.7)
- Online manual

stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				1	1	-	1	1	10 000	1			0	0	0	0	0					OPEN									
Backend	0	0		0	0		0	0	1 000	0	0	0s	0	0	0	0		0	0	0	0	0s UP		0	0	0		0		

marathon_http_in

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	10 000	0			0	0	0	0	0					OPEN									

marathon_http_appid_in

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	10 000	0			0	0	0	0	0					OPEN									

marathon_https_in

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	10 000	0			0	0	0	0	0					OPEN									

postgresql_15432

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	10 000	0			0	0	0	0	0					OPEN									

postgresql_15432

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
10_0_0_96_13831	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	0s UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Backend	0	0		0	0		0	0	1 000	0	0	?	0	0	0	0		0	0	0	0	0s UP		1	1	0		0	0s	

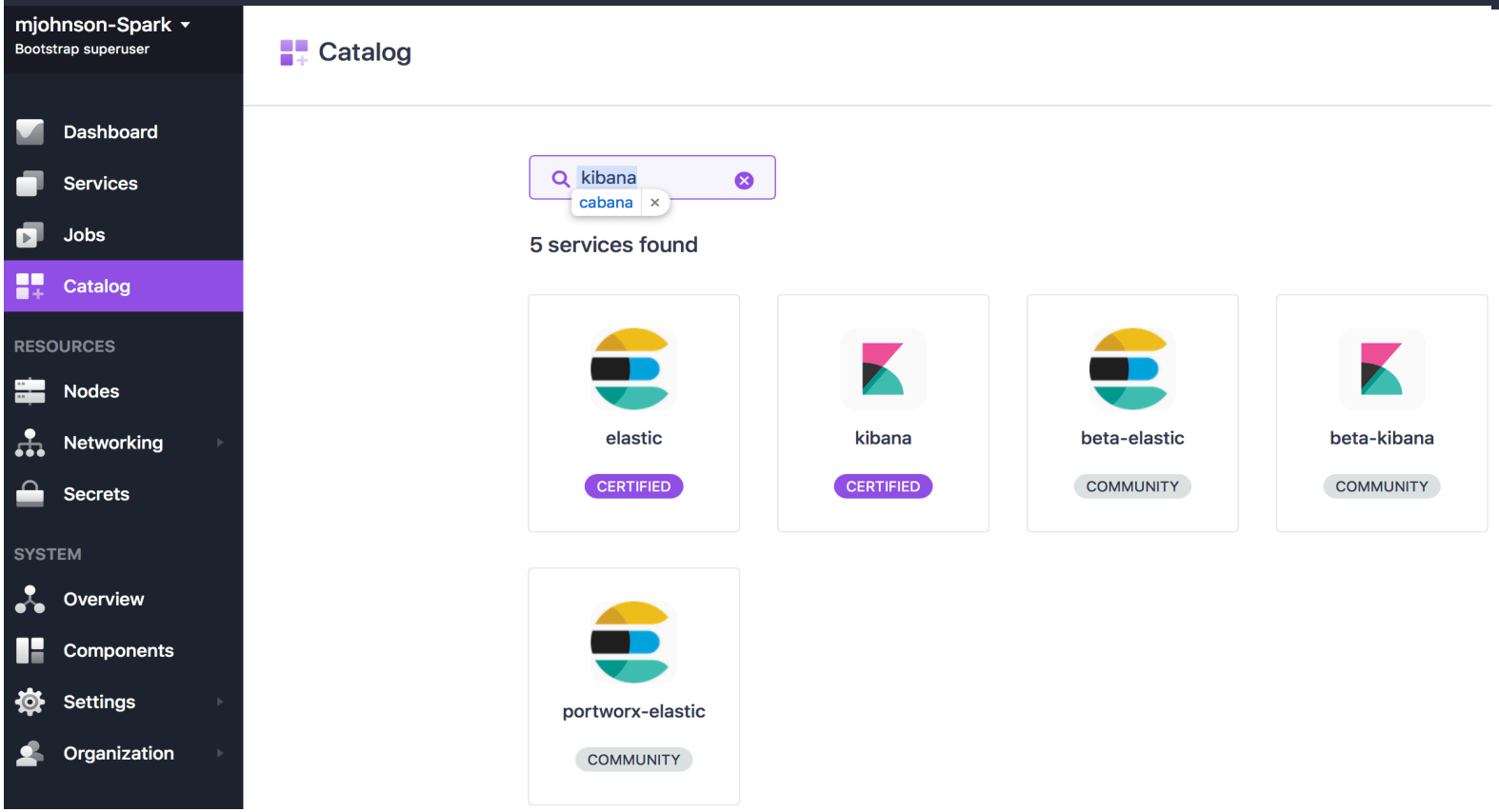
PostgresQL Example Program

```
import psycopg2
import sys
from psycopg2.extensions import ISOLATION_LEVEL_AUTOCOMMIT

con = None
host = "34.215.42.192"
psql_port = "15432"

try:
    con = psycopg2.connect("host='{}' dbname='postgres' user='admin' password='password' port={}".format(host,psql_port))

    con.set_isolation_level(ISOLATION_LEVEL_AUTOCOMMIT)
    cur = con.cursor()
    cur.execute("CREATE DATABASE test_db;")
    cur.execute("CREATE TABLE link (ID serial PRIMARY KEY, url VARCHAR (255) NOT NULL, name VARCHAR (255) NOT
NULL,"
            + "description VARCHAR (255),rel VARCHAR(50)); ")
    cur.execute("INSERT INTO link (url, name) VALUES('http://www.postgresqltutorial.com','PostgreSQL Tutorial');")

    con.commit()
except psycopg2.DatabaseError, e:
    if con:
        con.rollback()

    print 'Error %s' % e
    sys.exit(1)

finally:
    if con:
        con.close()
```

```
/System/Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7 /Users/markjohnson/demos/spark/spark_demo/postgresQL_Spark/postgresql_rw.py

Show me the Rows:

http://www.postgresqltutorial.com, PostgreSQL Tutorial

Process finished with exit code 0
```

Spark & HDFS

Spark & Elastic Search



ElasticSearch identify coordinator

```
~/demos/spark/spark_demo/elastic_spark >dcos elastic endpoints
[
  "coordinator-http",
  "coordinator-transport",
  "data-http",
  "data-transport",
  "master-http",
  "master-transport"
]
~/demos/spark/spark_demo/elastic_spark >
```

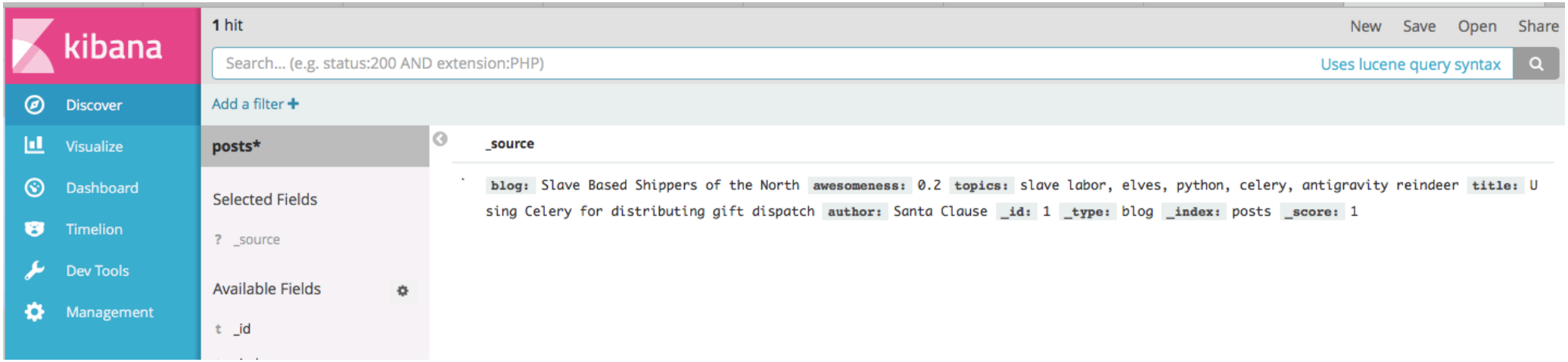
```
~/demos/spark/spark_demo/elastic_spark >dcos elastic endpoints coordinator-http
{
  "address": ["10.0.1.138:1025"],
  "dns": ["coordinator-0-node.elastic.autoip.dcos.thisdcos.directory:1025"],
  "vip": "coordinator.elastic.l4lb.thisdcos.directory:9200"
}

~/demos/spark/spark_demo/elastic_spark >
```

ElasticSearch post example

```
import elasticsearch

elastic_url = 'coordinator-0-node.elastic.autoip.dcos.thisdcos.directory'
elastic_url = 'localhost'
elastic_port = 1025

print('Connecting to Elastic Search url={},
port={}'.format(elastic_url,elastic_port))
es = elasticsearch.Elasticsearch([{'host':elastic_url,'port':elastic_port}])
es.index(index='posts', doc_type='blog', id=1, body={
    'author': 'Santa Clause',
    'blog': 'Slave Based Shippers of the North',
    'title': 'Using Celery for distributing gift dispatch',
    'topics': ['slave labor', 'elves', 'python',
            'celery', 'antigravity reindeer'],
    'awesomeness': 0.2
})
```

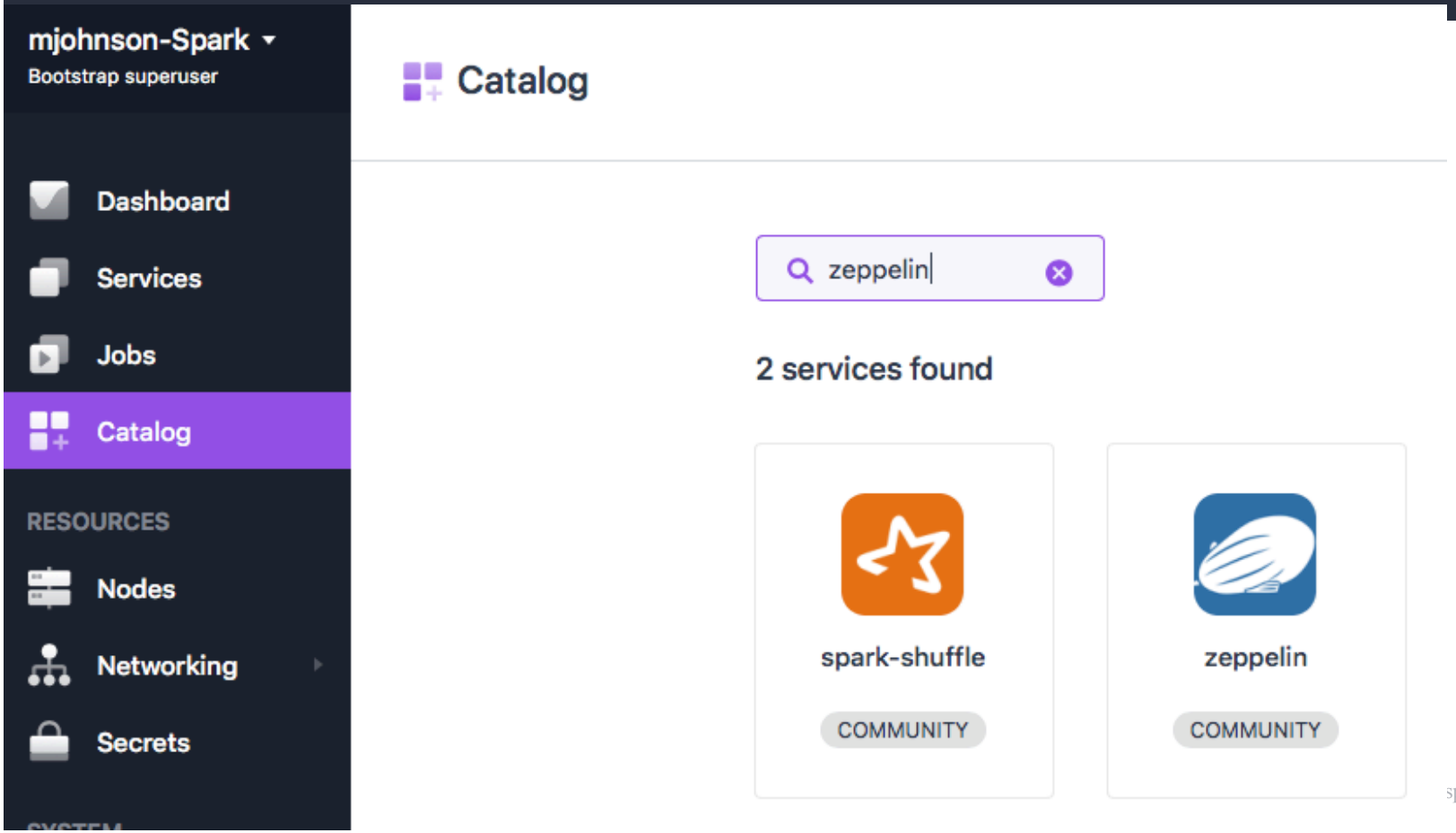
- Use ‘dcos elastic
- Example writes to the index posts one entry

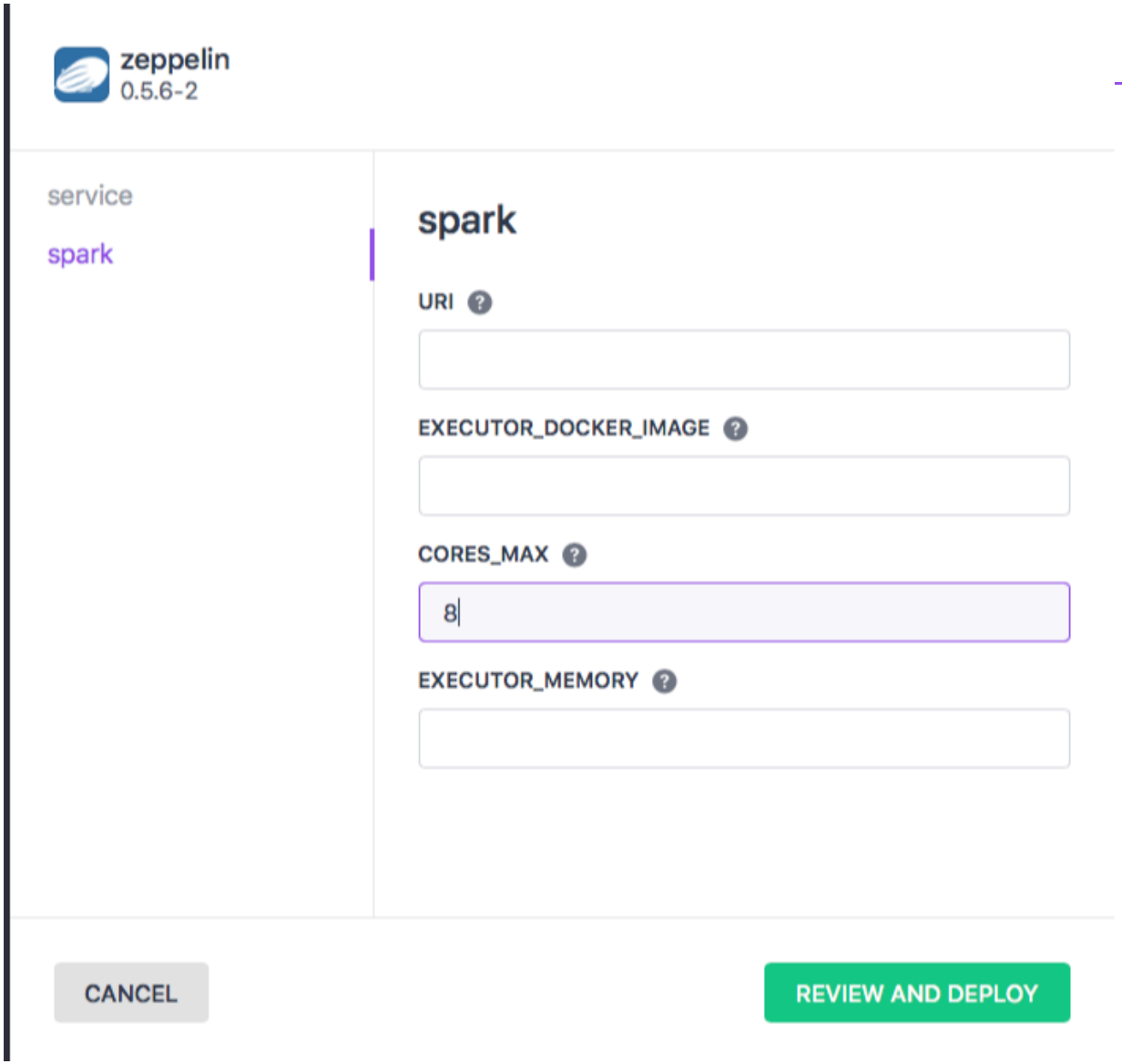
Kibana Search Example

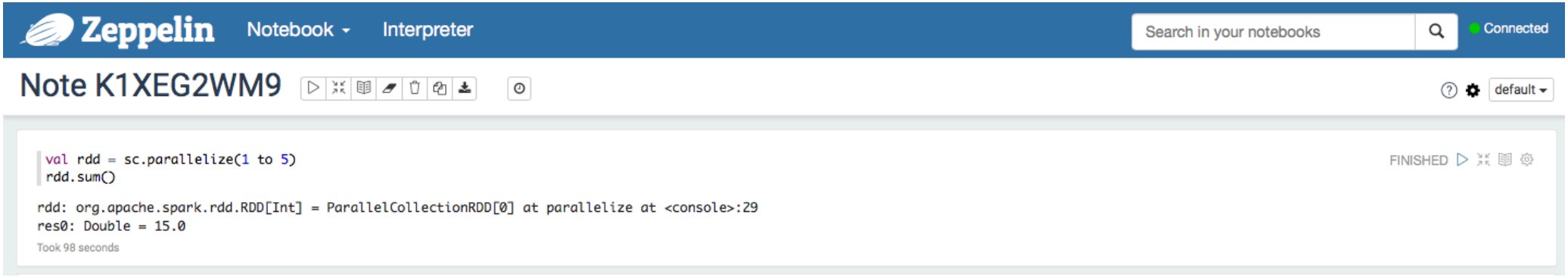


Part V: Spark Notebooks

Zeppelin Notebooks







MESOSPHERE

Zeppelin Notebook Interpreter Search in your notebooks Connected

Zeppelin Tutorial

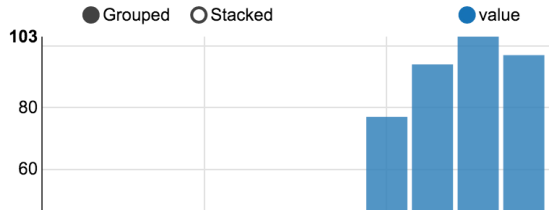
default

```
// Zeppelin creates and injects sc (SparkContext) and sqlContext (HiveContext or SqlContext)
// So you don't need create them manually

// load bank data
val bankText = sc.parallelize(
    IOUtils.toString(
        new URL("https://s3.amazonaws.com/apache-zeppelin/tutorial/bank/bank.csv"),
        Charset.forName("utf8")).split("\n"))

case class Bank(age: Integer, job: String, marital: String, education: String, balance: Integer)

val bank = bankText.map(s => s.split(";")).filter(s => s(0) != "\"age\"").map(
    s => Bank(s(0).toInt,
            s(1).replaceAll("\"", ""),
            s(2).replaceAll("\"", ""),
            s(3).replaceAll("\"", ""),
            s(5).replaceAll("\"", "").toInt
        )
).toDF()
bank.registerTempTable("bank")
```

```
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
bankText: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[0] at parallelize at <console>:37
defined class Bank
bank: org.apache.spark.sql.DataFrame = [age: int, job: string, marital: string, education: string, balance: int]
```

Took 15 seconds (outdated)

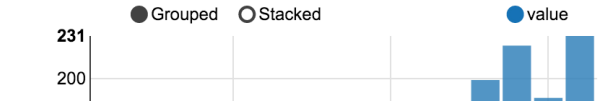
settings FINISHED



maxAge FINISHED

35

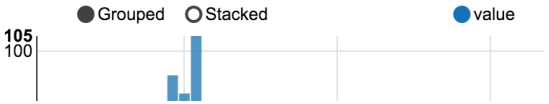
settings



marital FINISHED

single

settings



Linking Data sources

Additional Resources

- Presentation Examples: `https://github.com/markfjohnson/dcos_spark_demo.git`
- DC/OS Certified Services Documentation: https://docs.mesosphere.com/service-docs/
- Zeppelin Examples Doc: https://github.com/dcos/examples/tree/master/zeppelin/1.9
- Misc Examples: www.dcosexamples.com
- Mesosphere Blog: https://mesosphere.com/blog/
- Developers Guide to the Universe: https://mesosphere.com/blog/a-developers-guide-to-the-universe/

Spark and DC/OS

Next Session: Spark Security & HDFS Programming

Presented by Mark Johnson

1.10 MESOSPHERE DC/OS ENTERPRISE FEATURES

New for 1.10

			Mesosphere DC/OS Open source platform for modern apps	Mesosphere DC/OS Enterprise Production-grade solution for mission-critical apps
Features	Platform Services	Kubernetes (Beta) container orchestration	✓	✓
		Marathon container orchestration with pods support	✓	✓
		Production-certified data services (e.g., Spark, Kafka, Cassandra)	✓	✓
		Distributed fault tolerant jobs scheduler (Cron)	✓	✓
	DC/OS Catalog	Universe of 100+ platform services from easily deployed with a single single-click or CLI	✓	✓
		command; Developed by Mesosphere, DC/OS community, and commercial partners	✓	✓
	Resource Management	Apache Mesos distributed systems kernel w/ Universal Container Runtime	✓	✓
		Management of persistent and external volumes	✓	✓
		Virtual networks with IP per container and Container Network Interface (CNI) Support	✓	✓
		Distributed load balancer, service discovery and name-based VIPs	✓	✓
		GPU-based scheduling	✓	✓
	Management & Monitoring	Guided on-premise and cloud installation templates (AWS, Azure)	✓	✓
		Powerful CLI and GUI	✓	✓
		Non-disruptive DC/OS upgrades	✓	✓
	Ops & troubleshoot	Platform monitoring & troubleshooting tools	✓	✓
		Application-level logging, metrics & debugging	✓	✓
		In-Place Upgrade for Data services		✓
		Validated DC/OS upgrades with automated pre and post upgrade health checks		✓
	Security	Role-Based Access Control for containers, jobs, and data services		✓
		Identity management integration (Active Directory/LDAP/SAML 2.0/OpenID Connect)		✓
		Secrets Management (Key/Value and File-based)		✓
		Public key infrastructure w/ Custom certificate authority integration		✓
		Security audit logging		✓
	Multi-tenancy	Fine-grained access control lists for containers and services w/ folder integration		✓
		Service accounts		✓
	Adv. Network	High Performance L4/L7 Ingress load balancer (Edge-LB)		✓
Support	Emergency patching			✓
	Support for open-source Mesosphere-developed frameworks*		Cassandra, Kafka, Spark, Jenkins, Elastic, HDFS	Cassandra, Kafka, Spark, Jenkins, Elastic, HDFS
	Support options		Premium	Standard, Premium
Price	List price		Premium: $3,000	Standard: $3,250 Premium: $3,950

* Does not include support for baseline technology, e.g. Apache Spark