Mesosphere DC/OS

SMACK Stack Hands-on Tutorial

for DC/OS 1.10

Introduction

Welcome to Mesosphere's SMACK Stack hands-on tutorial for DC/OS. This tutorial is designed to guide you through the process of deploying the SMACK Stack components including Spark, Cassandra, and Kafka on an DC/OS Mesos cluster. Additionally, you will be guided through the process of deploying Apache Hadoop HDFS and a few other services to compliment the SMACK Stack components.

While this tutorial does not require any previous DC/OS or SMACK Stack experience, it would be helpful to have knowledge of how clustered servers work together (master nodes and worker nodes) and experience using the Linux operating system and BASH shell.

While working with DC/OS and the SMACK Stack components, you will be using Mesosphere's DC/OS Dashboard, the DC/OS Command Line Interface (CLI) and occasionally, plain Linux shell commands.

If you would like to review documentation on Mesosphere's DC/OS and the Apache Mesos Project, refer to these links:

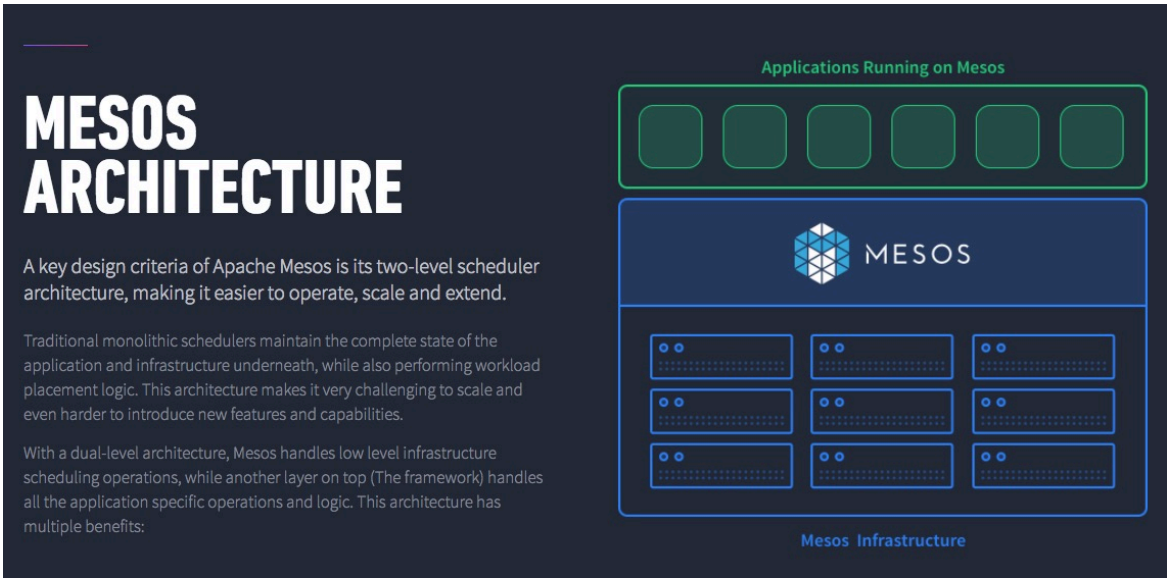
- Mesosphere's Enterprise DC/OS: http://mesosphere.io
- Open Source DC/OS: http://dcos.io
- Apache Mesos Project: http://mesos.apache.org

The environment you will use in this tutorial should be staged in advance, including a DC/OS cluster running on AWS, Azure, Google Cloud Platform or on prem. To run the SMACK Stack, you should have at least 10 private agent nodes with enough CPU, Memory and Disk to support all of the tasks to be deployed on the cluster. Contact your Mesosphere sales representative to get help installing

an Enterprise DC/OS cluster, or if you are not a customer yet, deploy an Open Source DC/OS cluster.

Enterprise DC/OS and Data Services

Apache Mesos is the open-source distributed systems kernel at the heart of the Mesosphere DC/OS. It abstracts the entire datacenter into a single pool of computing resources, simplifying running distributed systems at scale.

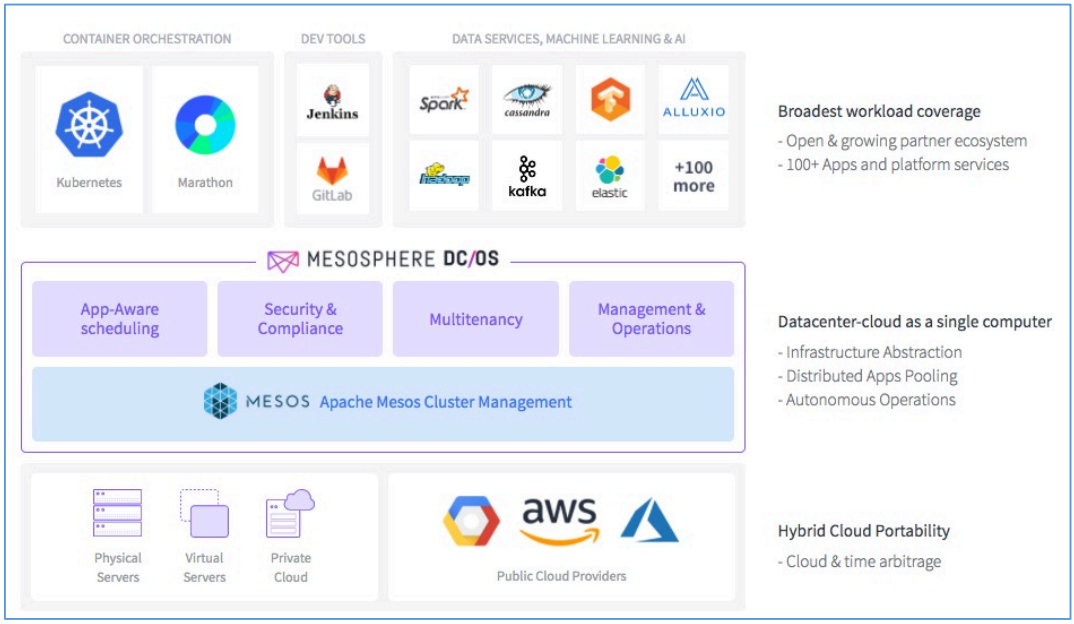


A key design criteria of Apache Mesos is its two-level, application aware, scheduler architecture, making it easier to operate, scale and extend.

Enterprise DC/OS is the most flexible platform for containerized, data intensive application.

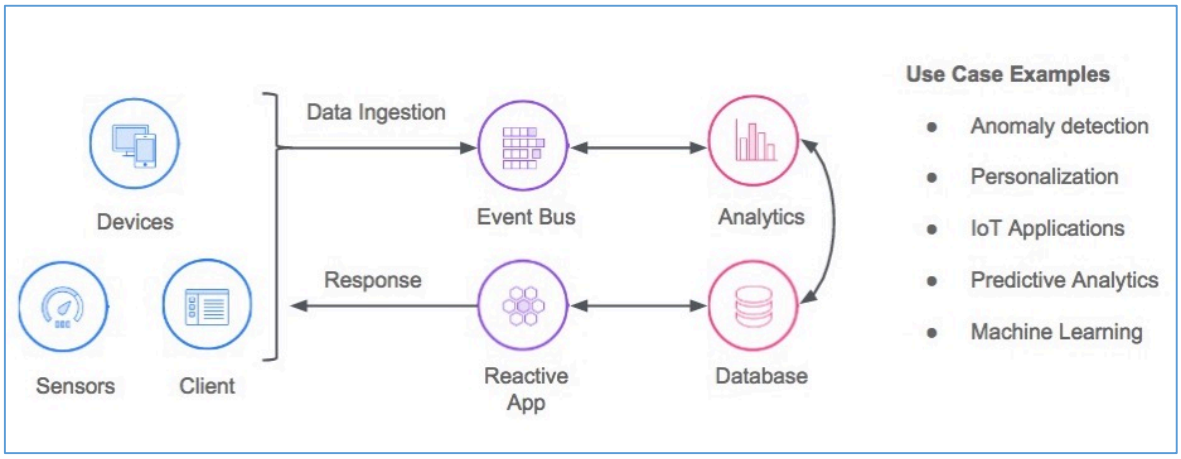
Extending the Mesosphere philosophy of emphasizing “freedom of choice” on DC/OS, Marathon and Kubernetes are both available for container orchestration. Development teams can now choose container orchestrators on our platform as easily as they choose data services, CI/CD, or networking tools. Kubernetes on DC/OS brings a public cloud-like “Containers-as-a-Service” experience to any infrastructure, and allows you to run Kubernetes applications alongside big data services with a common set of security, maintenance, and management tools.

Kubernetes on DC/OS will allow operators to easily install, scale and upgrade multiple production-grade Kubernetes clusters on Mesosphere DC/OS. Infrastructure owners will be able to offer application developers Kubernetes for Docker container orchestration alongside other data services or legacy applications, all on shared DC/OS infrastructure while maintaining high availability and isolation. All of these services running on DC/OS benefit from complete hybrid cloud portability on an open platform.

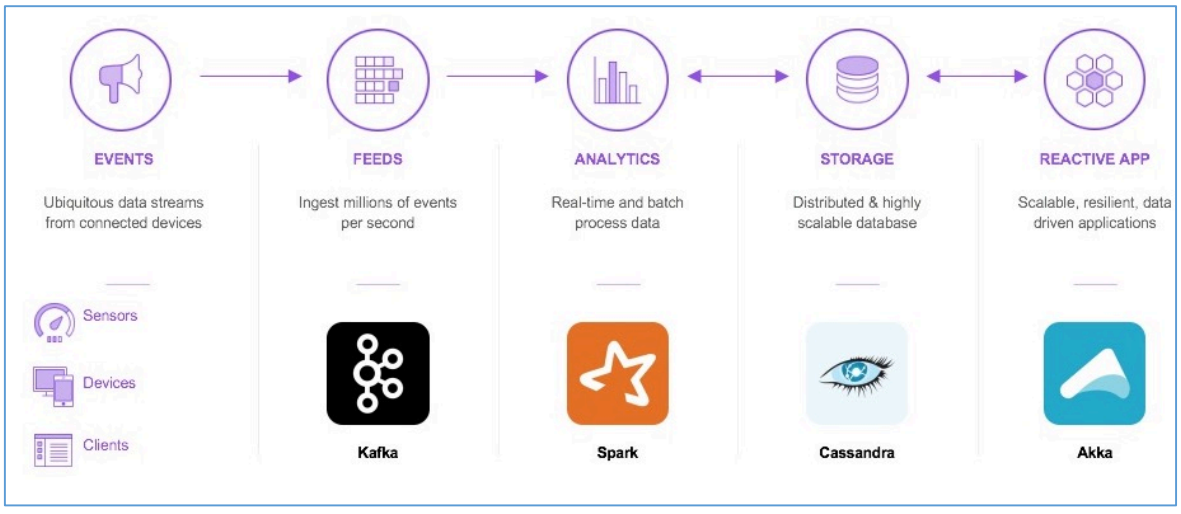


Many IT organizations are developing and deploying a new generation of highly integrated, data intensive applications that process data in a real-time or semi

real-time basis. These new applications requiring running containerized applications in the same environment as their analytics and data storage applications. Mesosphere's DC/OS is supremely suited for supporting these types of mix-workload requirements.



By allowing the SMACK Stack to run in the same deployment environment, DC/OS allows custom containerized applications, often implemented as microservices, to run right next to stateful services like Kafka, for messaging, Spark for analytics and Cassandra for highly scalable storage.

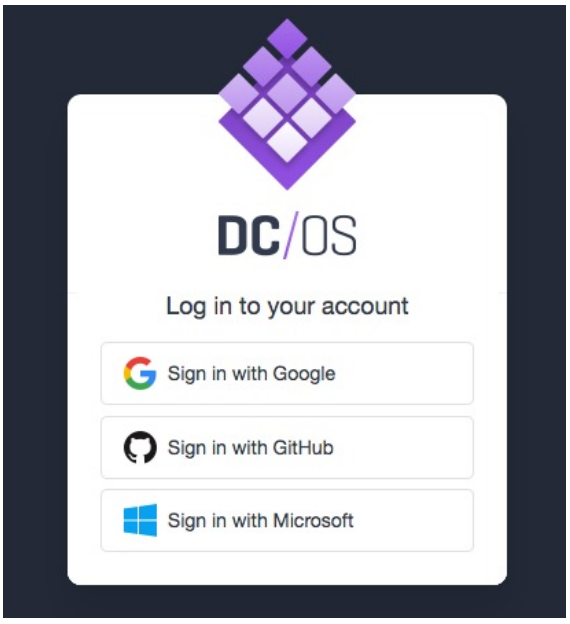


DC/OS Dashboard

In this section of the tutorial you will log into the DC/OS Web based Dashboard and create an environment for deploying the SMACK Stack components.

Open Source DC/OS

The open source version of DC/OS supports the OAuth authentication method using an OpenID authentication server. To log into your open source DC/OS Dashboard, you can authenticate with your Google, GitHub or Microsoft account. Point your Web browser to your master node URL to see the sign in prompt.

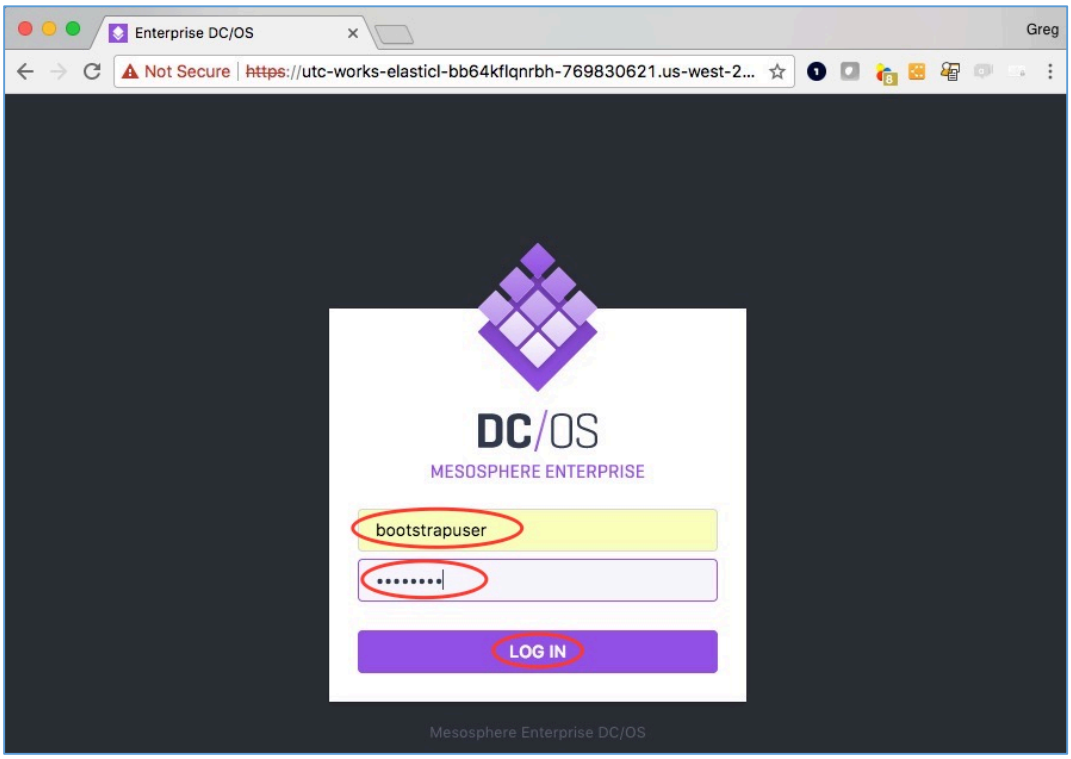


Click on the service that you would like to use to authenticate.

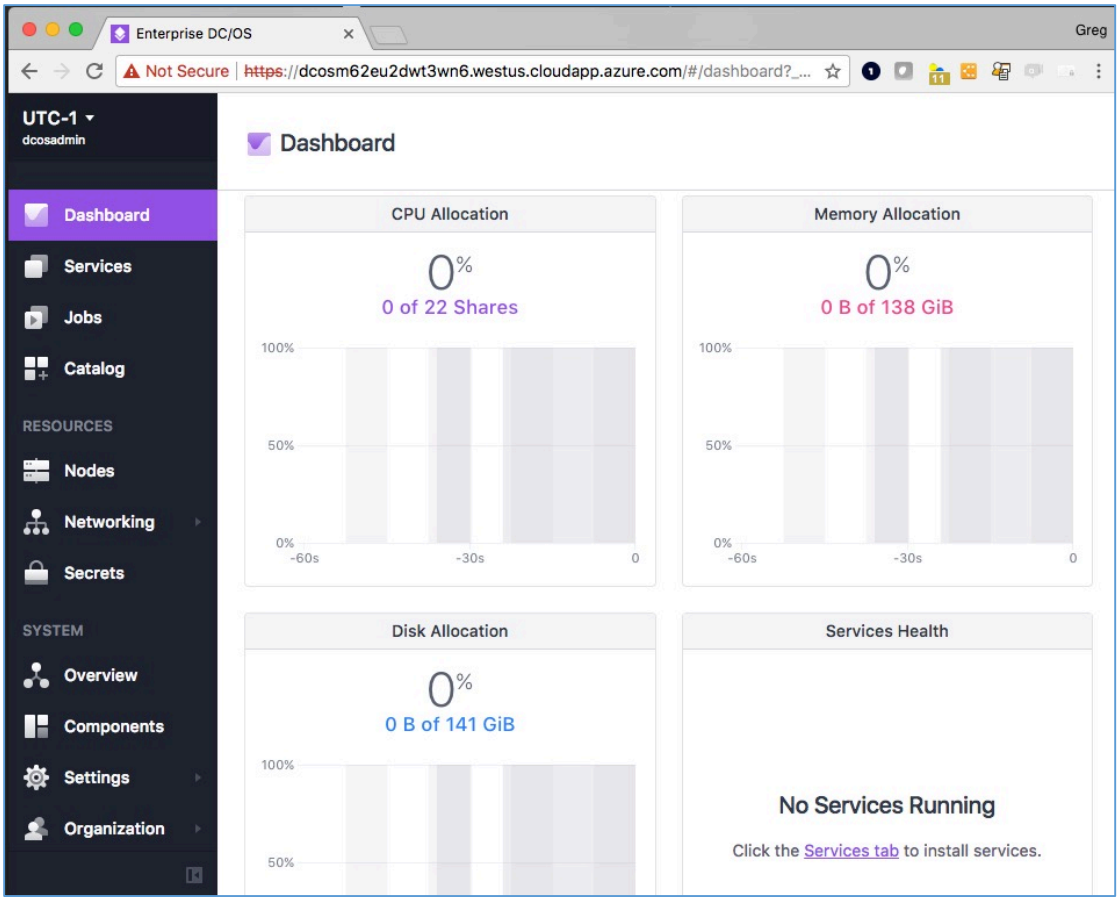
Enterprise DC/OS

Enterprise DC/OS has the ability to link to your AD/LDAP directory service or integrate with your SAML 2.0 and OAuth2 servers. But in this tutorial you will be using a local DC/OS user.

At this time, log in using the DC/OS administrator user and the password provided by your system administrator. Click on the LOG IN button.

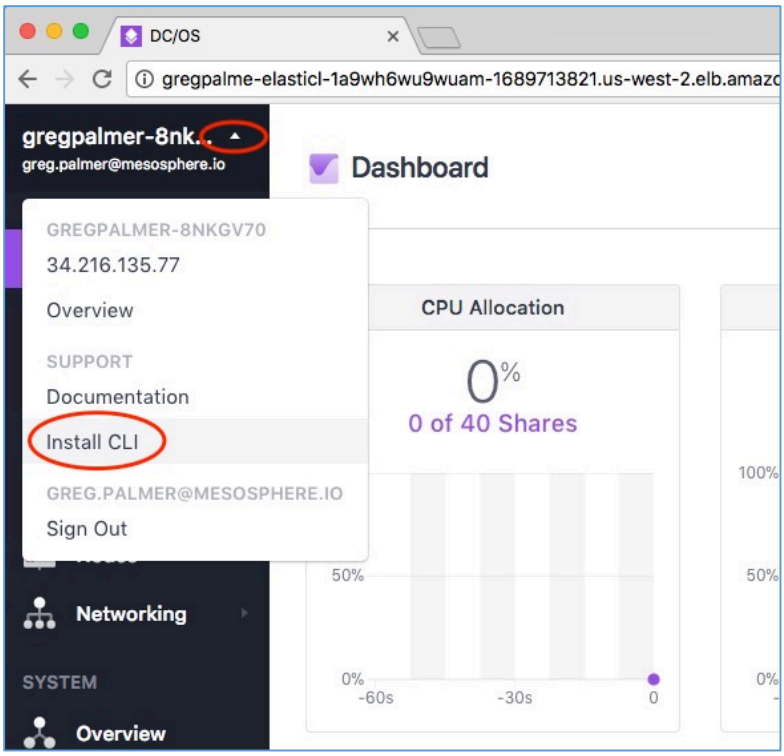


When successfully logged in, you will be presented with the main DC/OS Dashboard screen. The Dashboard shows the menu options down the left side and the resource allocations and service health on the right side. Since this is a newly launched DC/OS cluster, there are no resources allocated at this time.

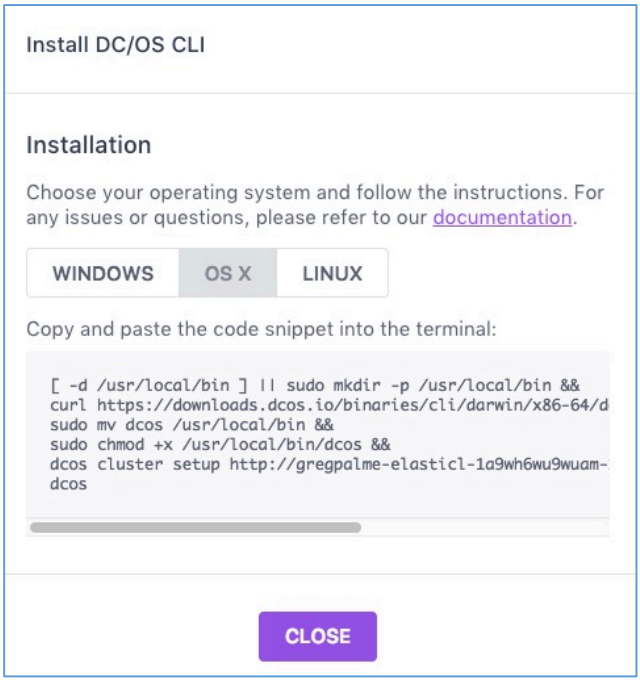


DC/OS Command Line Interface (CLI)

Several steps in this tutorial require the use of the DC/OS Command Line Interface or CLI. The CLI is available on Windows, Mac OS X and Linux operating systems. Install the CLI using the commands provided from the Dashboard's pull down menu in the upper left hand corner of the Dashboard page. Click on the cluster name in the upper left corner to view the Install CLI link.

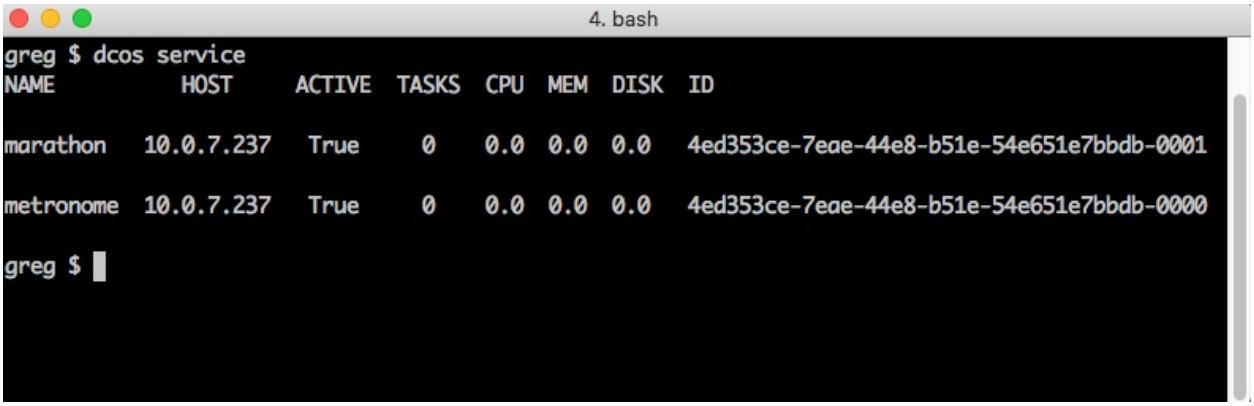


Click on the Install CLI link to view the detailed instructions for installing the CLI in your OS environment. Copy and paste those commands and run them on your laptop or other client computer.



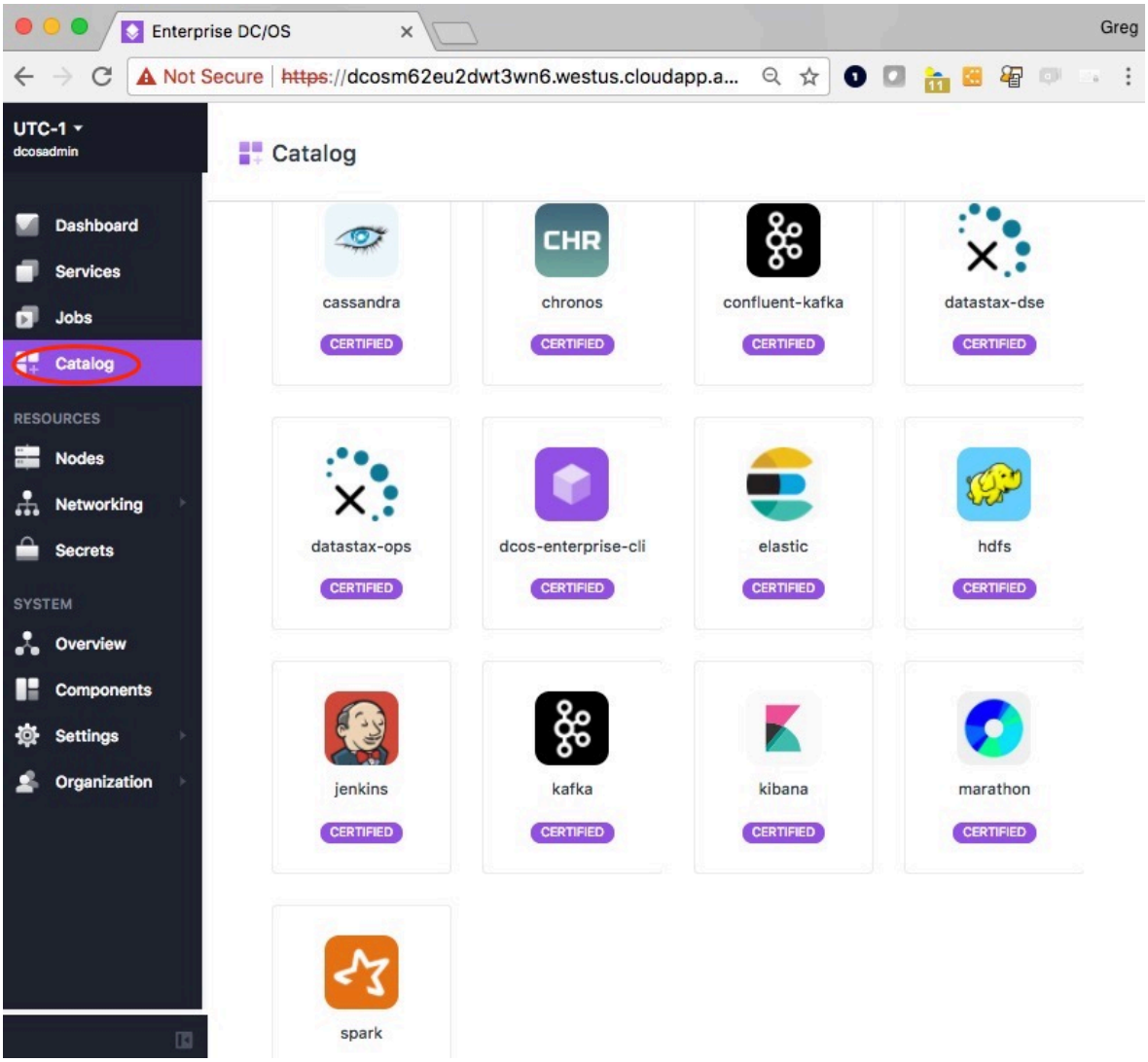
Follow the prompts and you will be successfully logged into the cluster via the CLI. Test the CLI with a command to list the running services:

```
$ dcos service
```



Begin the Tutorial

In this tutorial you will be configuring and deploying the SMACK Stack and other packages from the DC/OS service catalog. Mesosphere has created the service catalog as a way to quickly deploy complex services that require multiple tasks to be launched in a specific order and on various agent nodes in the cluster. Click on the Catalog menu option on the left to see the packages available. If you scroll down, you will see over 100 packages available from the community includes databases, analytical tools, microservice and container tools and more.



Apache Cassandra

DC/OS Apache Cassandra is an automated service that makes it easy to deploy and manage Apache Cassandra on DC/OS. Apache Cassandra is a distributed NoSQL database offering high availability, fault tolerance and scalability across data centers.

For more information on Apache Cassandra, see the Apache Cassandra documentation at:

http://cassandra.apache.org/doc/latest

Features

- Easy installation
- Simple horizontal scaling of Cassandra nodes
- Straightforward backup and restore of data out of the box
- Multi-datacenter replication support

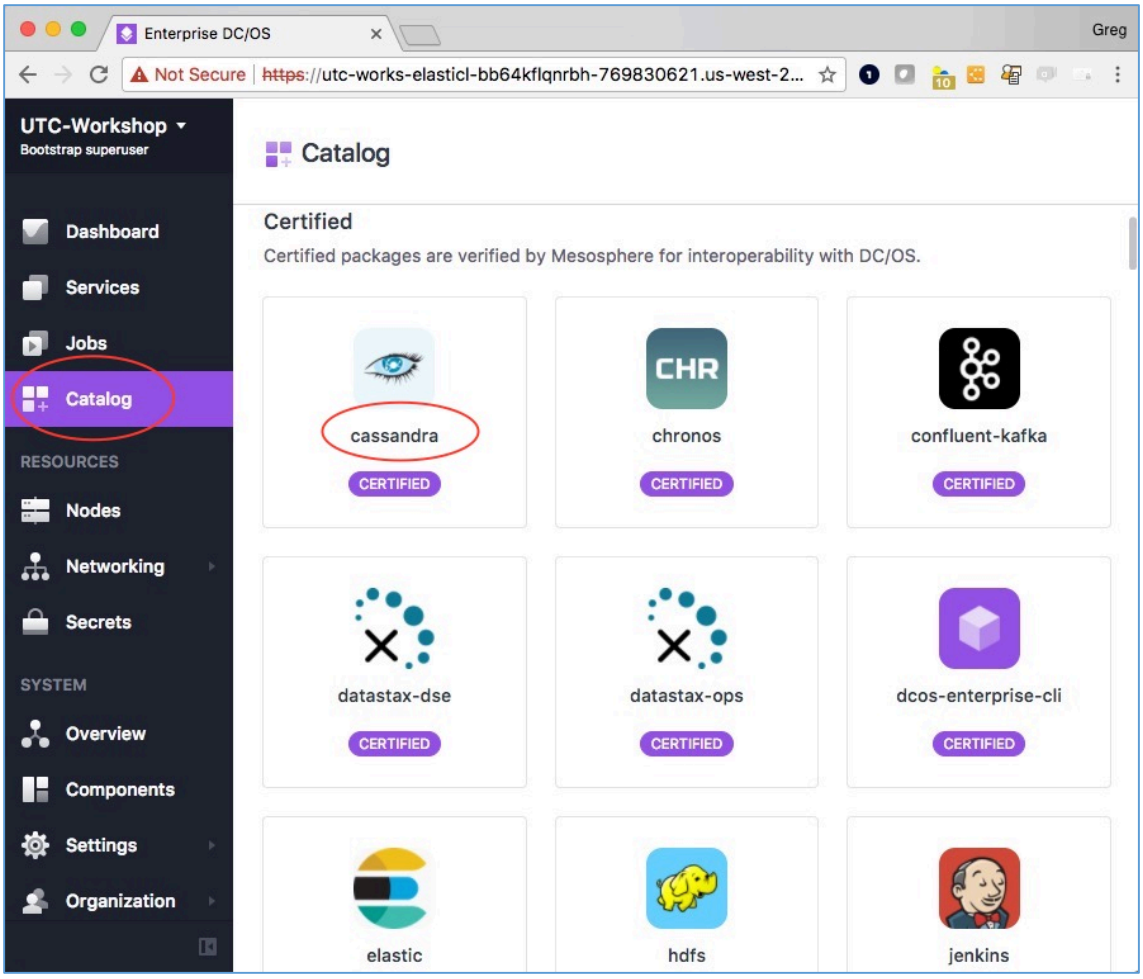
See the Mesosphere DC/OS Cassandra documentation at:

https://docs.mesosphere.com/service-docs/cassandra

In this section of the tutorial, you will deploy the Apache Cassandra distributed database on the DC/OS agent nodes.

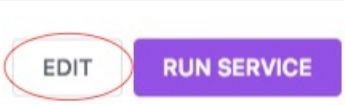
Configure and Deploy Cassandra on DC/OS

In the DC/OS Dashboard, click on the Catalog menu option on the left and display the data services packages in the DC/OS Catalog. Then click on the Cassandra package.



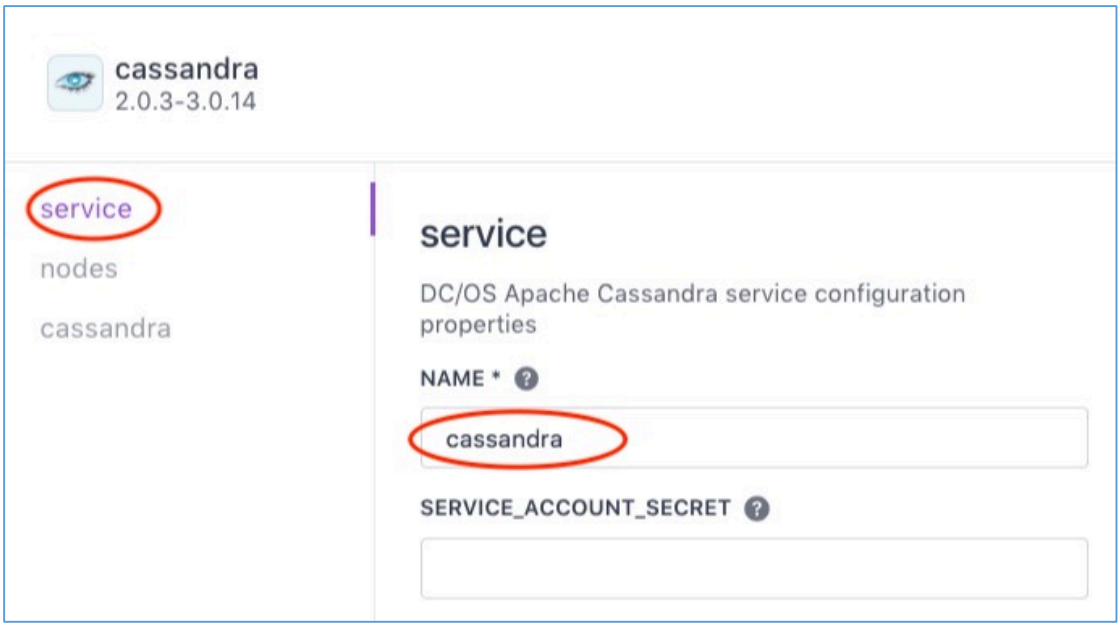
You will see some details about the Cassandra service on DC/OS. Click on the REVIEW & RUN button.

Then click on the EDIT button to modify the configuration.

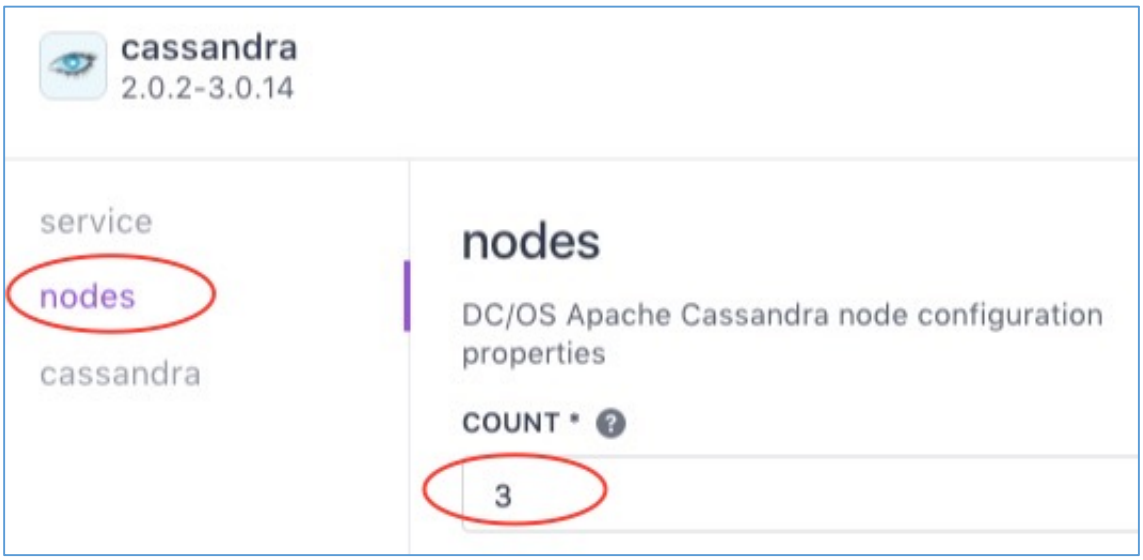


The DC/OS Cassandra package configuration screens allow you to modify the default configuration and in this tutorial you will be using the default configuration settings for deployment.

Click on the service category and keep the name of the Cassandra service as cassandra.

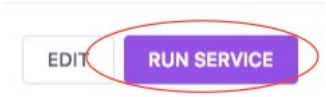


Then click on the nodes category and keep the number of nodes at 3. This will cause the Cassandra service to start three nodes on three different agent nodes.

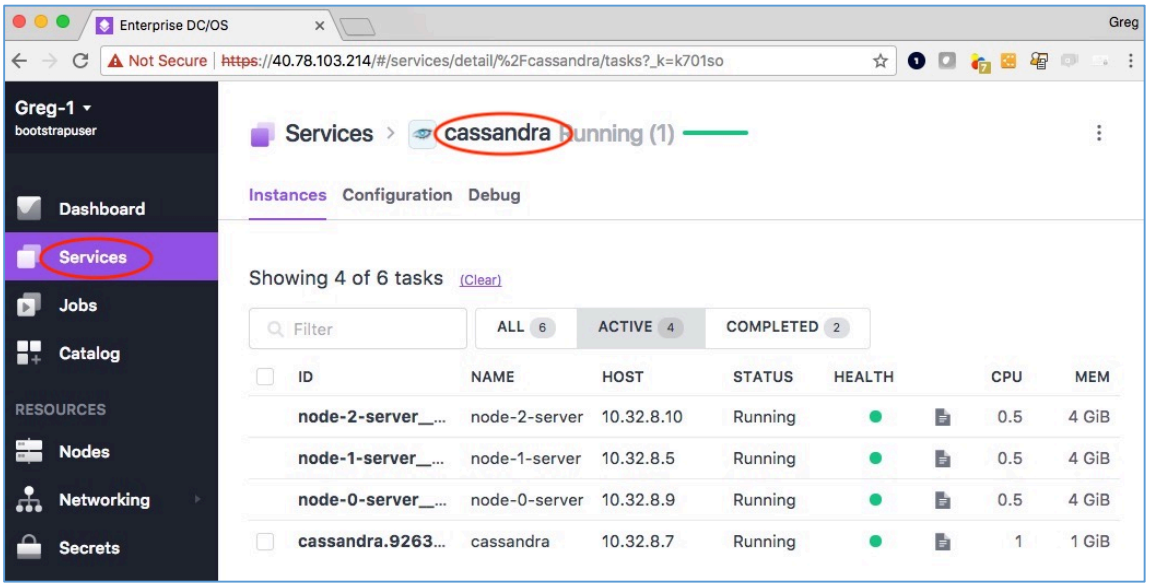


Now that you have completed the changes needed to deploy the Cassandra service on DC/OS, click the REVIEW & RUN button.

Then click the RUN SERVICE button.



After the Cassandra service starts up and passes its health check, you will see the tasks running on the DC/OS Mesos cluster. Click on the Services menu option on the left and then click on the cassandra service name. You will see the Cassandra node managers running on three different DC/OS agent nodes and you will see the Cassandra Mesos framework running as well.



Apache Kafka

DC/OS Apache Kafka is an automated service that makes it easy to deploy and manage Apache Kafka on Mesosphere DC/OS, eliminating nearly all of the complexity traditionally associated with managing a Kafka cluster. Apache Kafka is a distributed high-throughput publish-subscribe messaging system with strong ordering guarantees. Kafka clusters are highly available, fault tolerant, and very durable. See the Apache Kafka documentation here:

http://kafka.apache.org/documentation.html

DC/OS Kafka gives you direct access to the Kafka API so that existing producers and consumers can interoperate. You can configure and install DC/OS Kafka in moments. Multiple Kafka clusters can be installed on DC/OS and managed independently, so you can offer Kafka as a managed service to your organization. See the Mesosphere DC/OS Kafka documentation here:

https://docs.mesosphere.com/service-docs/kafka

Benefits

DC/OS Kafka offers the following benefits of a semi-managed service:

- Easy installation
- Multiple Kafka clusters
- Elastic scaling of brokers
- Replication for high availability
- Kafka cluster and broker monitoring

Features

DC/OS Kafka provides the following features:

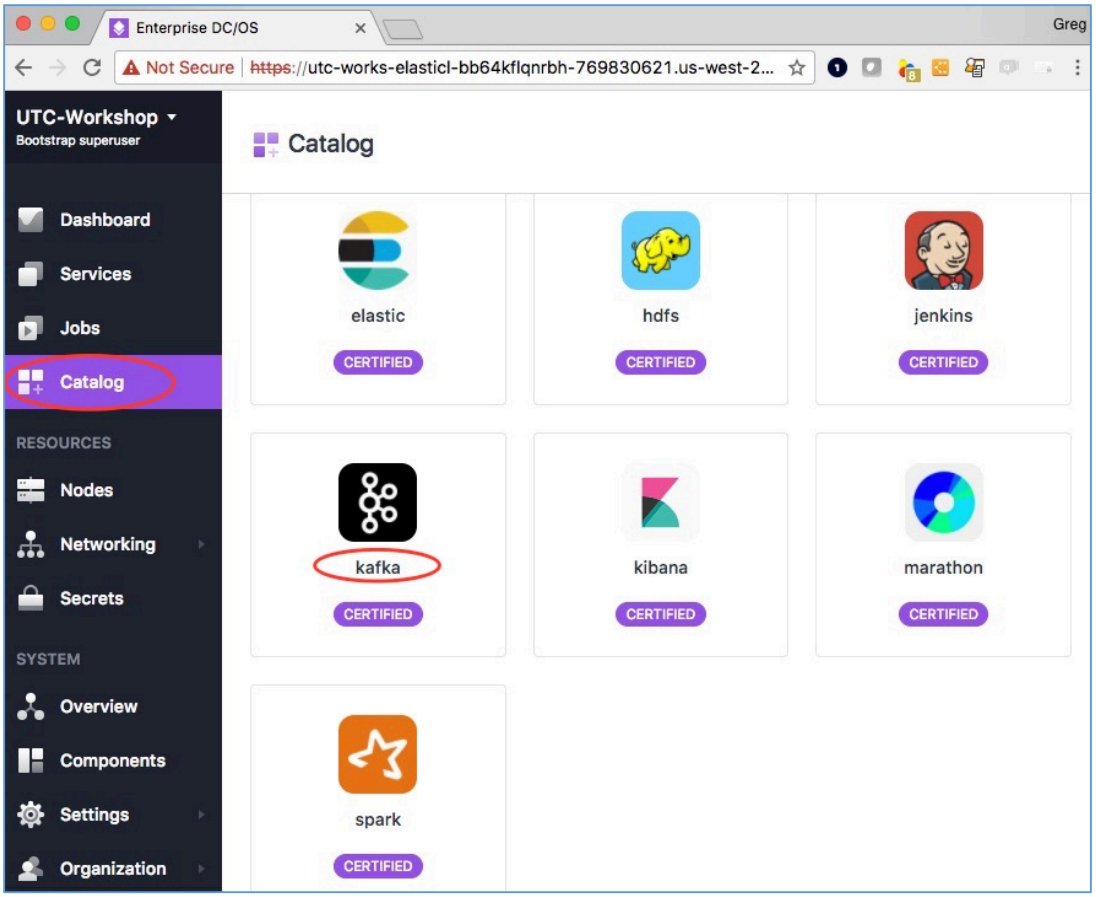
- Single-command installation for rapid provisioning

- Multiple clusters for multiple tenancy with DC/OS
- High availability runtime configuration and software updates
- Storage volumes for enhanced data durability, known as Mesos Dynamic Reservations and Persistent Volumes
- Integration with syslog-compatible logging services for diagnostics and troubleshooting
- Integration with statsd-compatible metrics services for capacity and performance monitoring

In this section of the tutorial, you will deploy the Apache Kafka messaging environment on the DC/OS agent nodes.

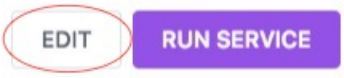
Configure and Deploy Kafka on DC/OS

In the DC/OS Dashboard, click on the Catalog menu option on the left and display the data services packages in the DC/OS Catalog. Then click on the Kafka package.



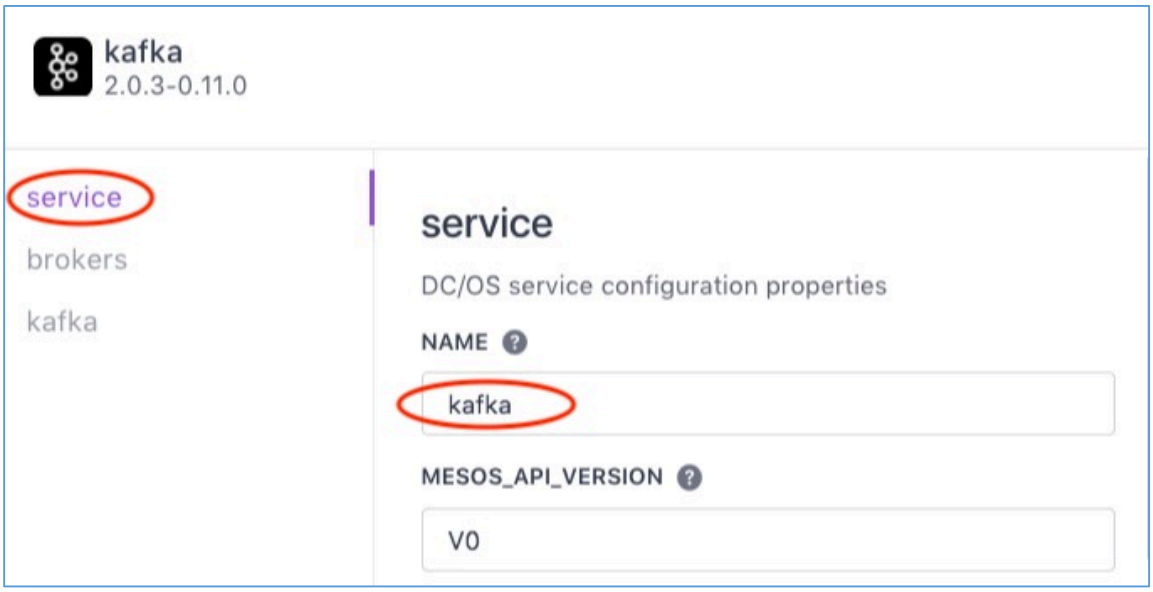
You will see some details about the Kafka service on DC/OS. Click on the REVIEW & RUN button.

Then click on the EDIT button to modify the configuration.

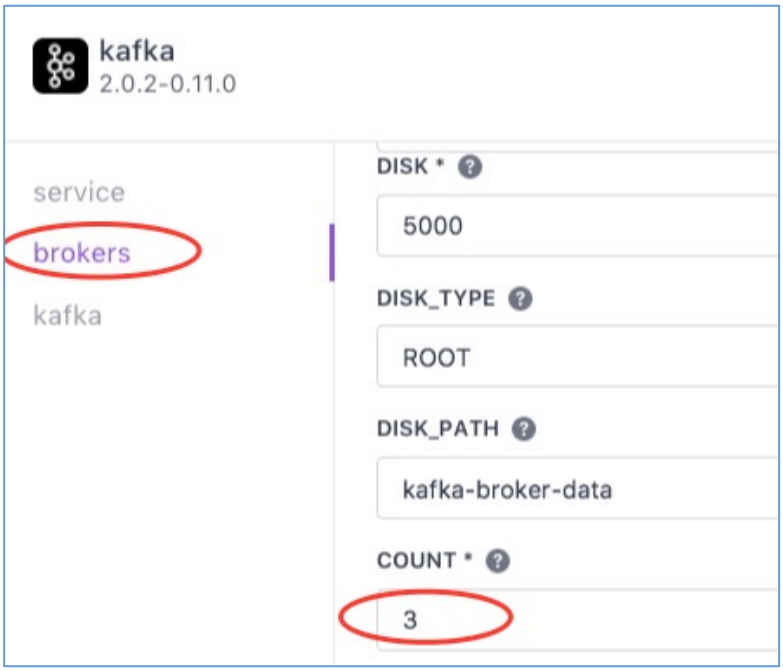


The DC/OS Kafka package configuration screens allow you to modify the default configuration and in this tutorial you will be using those defaults.

Click on the service category and keep the name of the Kafka service as kafka.

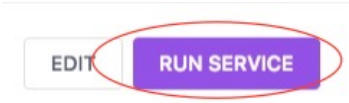


Then, click on the brokers category and keep the number of brokers to deploy as 3.

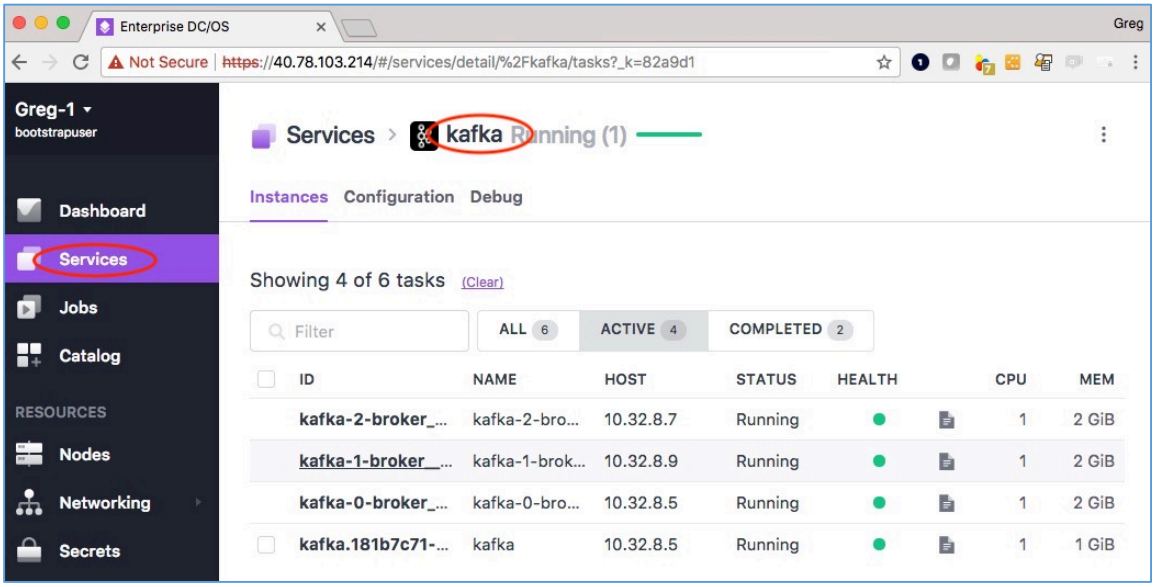


Next, deploy the Kafka service on DC/OS by clicking the REVIEW & RUN button.

Then click the RUN SERVICE button.



After the Kafka service starts up and passes its health check, you will see the tasks running on the DC/OS Mesos cluster. Click on the Services menu option on the left and then click on the kafka service name and you will see the three Kafka brokers running on three different DC/OS agent nodes. You will also see the Kafka Mesos framework running. This is the tasks that coordinates the launching of the other Kafka tasks.



Apache Hadoop HDFS

DC/OS Apache HDFS is a managed service that makes it easy to deploy and manage an HA Apache HDFS cluster on Mesosphere DC/OS. Apache Hadoop Distributed File System (HDFS) is an open source distributed file system based on Google's Google File System(GFS) paper. It is a replicated and distributed file system interface for use with "big data" and "fast data" applications.

You can find the Apache Hadoop documentation here:

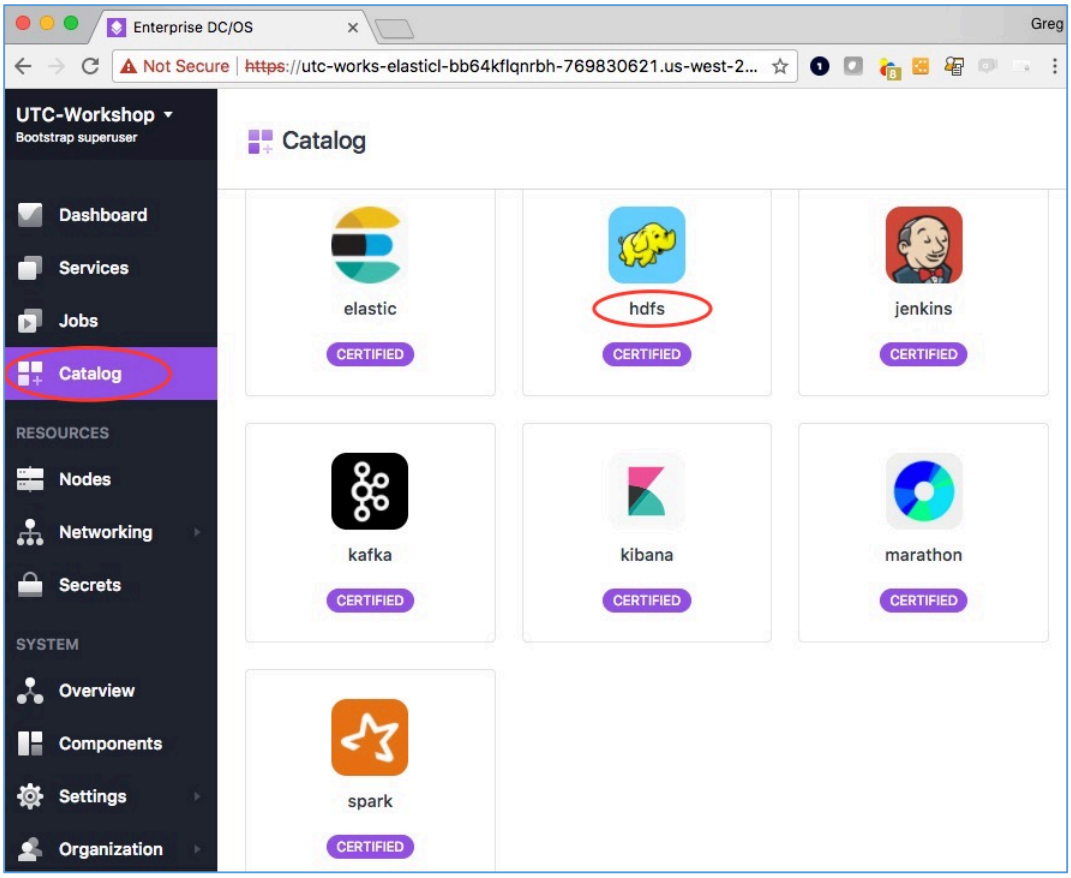
http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

And you can find the Mesosphere DC/OS HDFS documentation here:

https://docs.mesosphere.com/service-docs/hdfs/

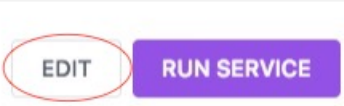
Configure and Deploy HDFS on DC/OS

In the DC/OS Dashboard, click on the Catalog menu option on the left and display the data services packages in the DC/OS Catalog. Then click on the HDFS package.



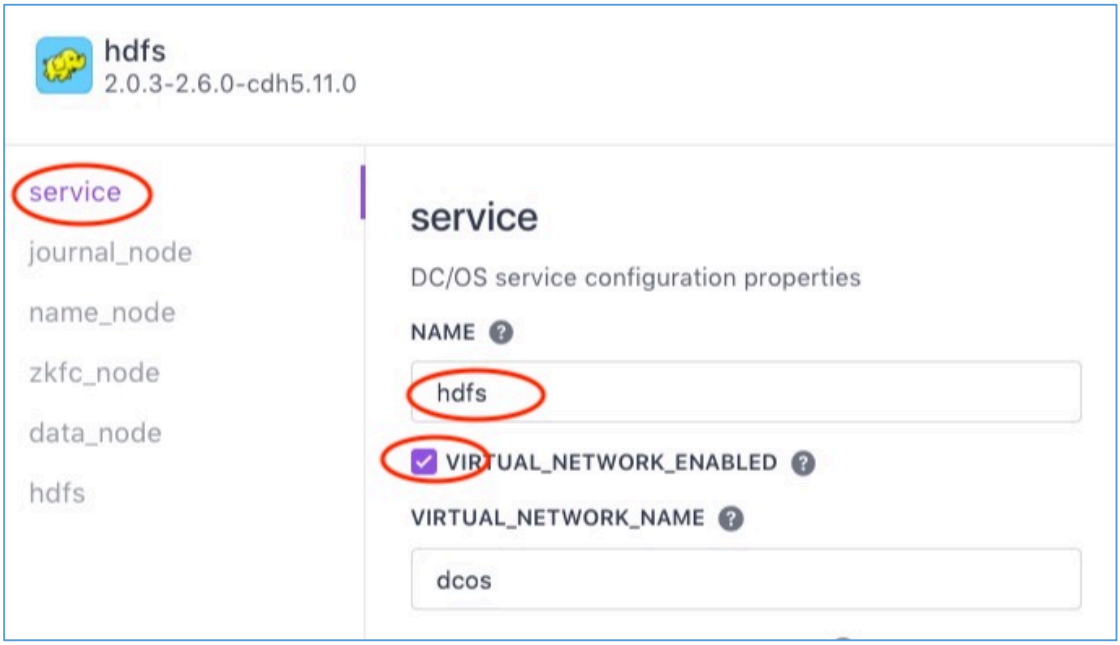
You will see some details about the HDFS service on DC/OS. Click on the REVIEW & RUN button.

Then click on the EDIT button to modify the configuration.

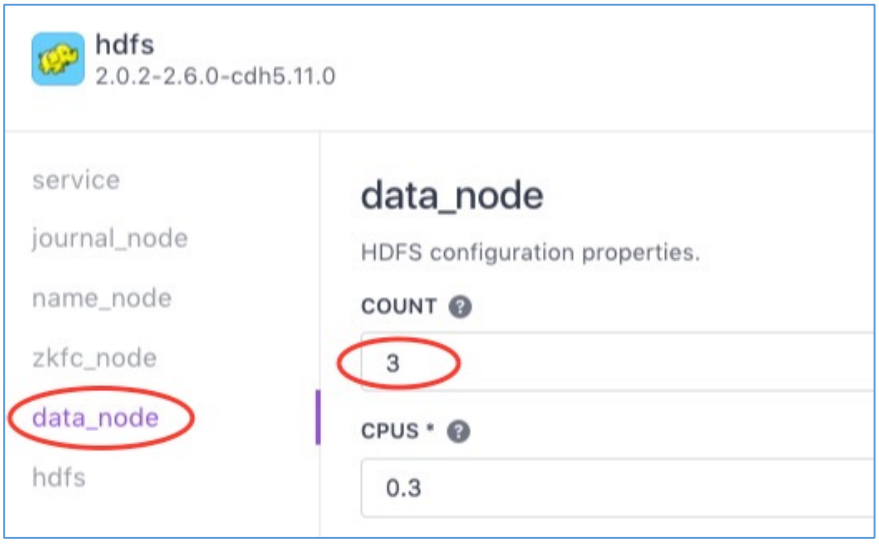


The DC/OS HDFS package configuration screens allow you to modify the default configuration and in this tutorial you will modify the virtual networking option, and the number of HDFS data nodes to deploy.

Click on the service category and keep the name of the HDFS service as hdfs. Also, click on the check box next to the VIRTUAL_NETWORK_ENABLED option. This will allow applications running on the cluster to access the HDFS service without knowing on which DC/OS agent nodes the various HDFS components are running.

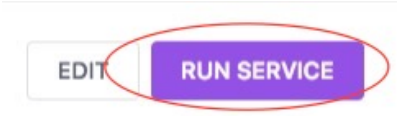


Next, click on the data_node category and keep the data_node count as 3. This will start three data node tasks on three different DC/OS agent nodes.

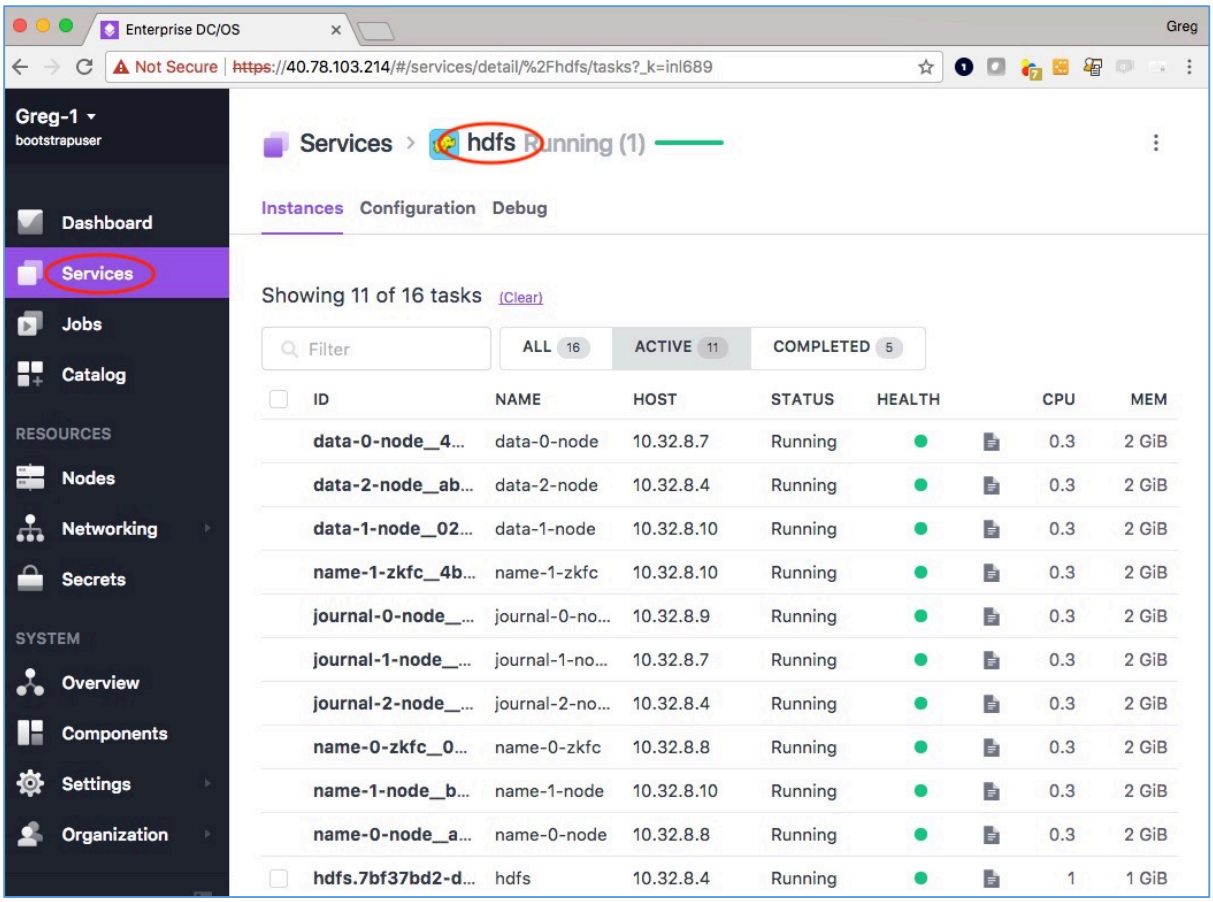


Now that you have completed the changes needed to deploy the HDFS service on DC/OS, click the REVIEW & RUN button.

Then click the RUN SERVICE button.



After the HDFS service starts up and passes its health check, you will see the tasks running on the DC/OS Mesos cluster. Click on the Services menu option on the left and then click on the hdfs service name and you will see the three name nodes, three journal nodes and three data nodes running on various DC/OS agent nodes and you will see the HDFS Mesos framework running as well.

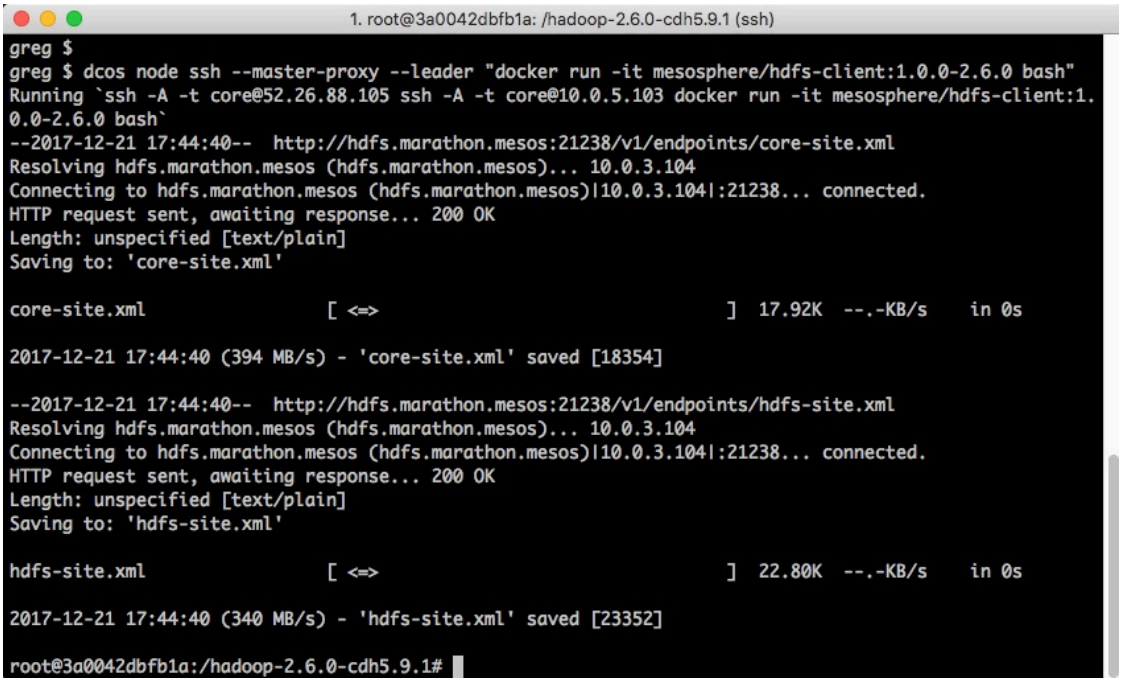


Using the HDFS Service

Next, launch an HDFS client shell session and run some Hadoop commands.

First issue the command to launch an hdfs-client Docker container on a node in the cluster. Here are the commands to use:

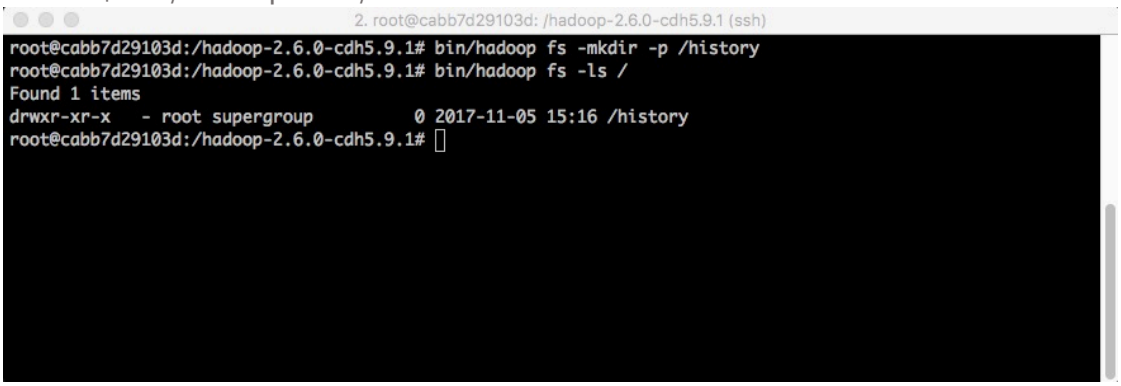
```
# NOTE: You may have to use the ssh-add command to get your private ssh key
#           to automatically offer the key to the remote ssh server. Use these commands:
$ eval "$(ssh-agent)"
$ ssh-add ~/.ssh/my-private-key.key

$ dcos node ssh --master-proxy --leader "docker run -it mesosphere/hdfs-client:1.0.0-2.6.0 bash"
```



Create an HDFS directory for the Spark History Server with the Hadoop command:

```
$ bin/hadoop fs -mkdir -p /history
```

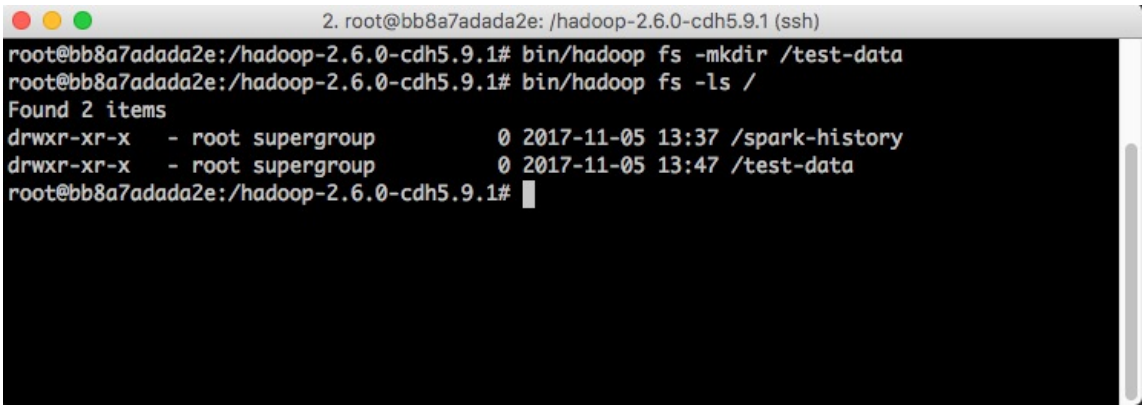
```
$ bin/hadoop fs -ls /
```



Copy some test data to an HDFS file. First, create a directory in HDFS to hold your data file. Use these commands:

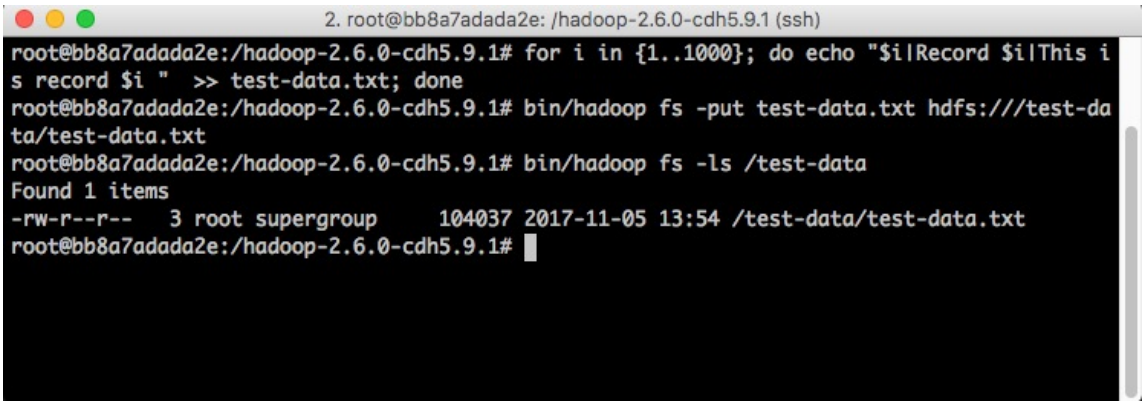
```
$ bin/hadoop fs -mkdir /test-data
```

```
$ bin/hadoop fs -ls /
```



Then create a data file with 1000 records and upload it to the HDFS directory. Use these commands:

```
$ dd if=/dev/urandom of=test-data.txt  bs=1048576  count=10

$ bin/hadoop fs -put test-data.txt hdfs:///test-data/test-data.txt

$ bin/hadoop fs -ls /test-data
```



Extract the data from HDFS and check the size of the new file using the commands:

```
$  bin/hadoop fs -get hdfs:///test-data/test-data.txt ./test-data-2.txt

$  ls -alh  ./test-data-2.txt
```

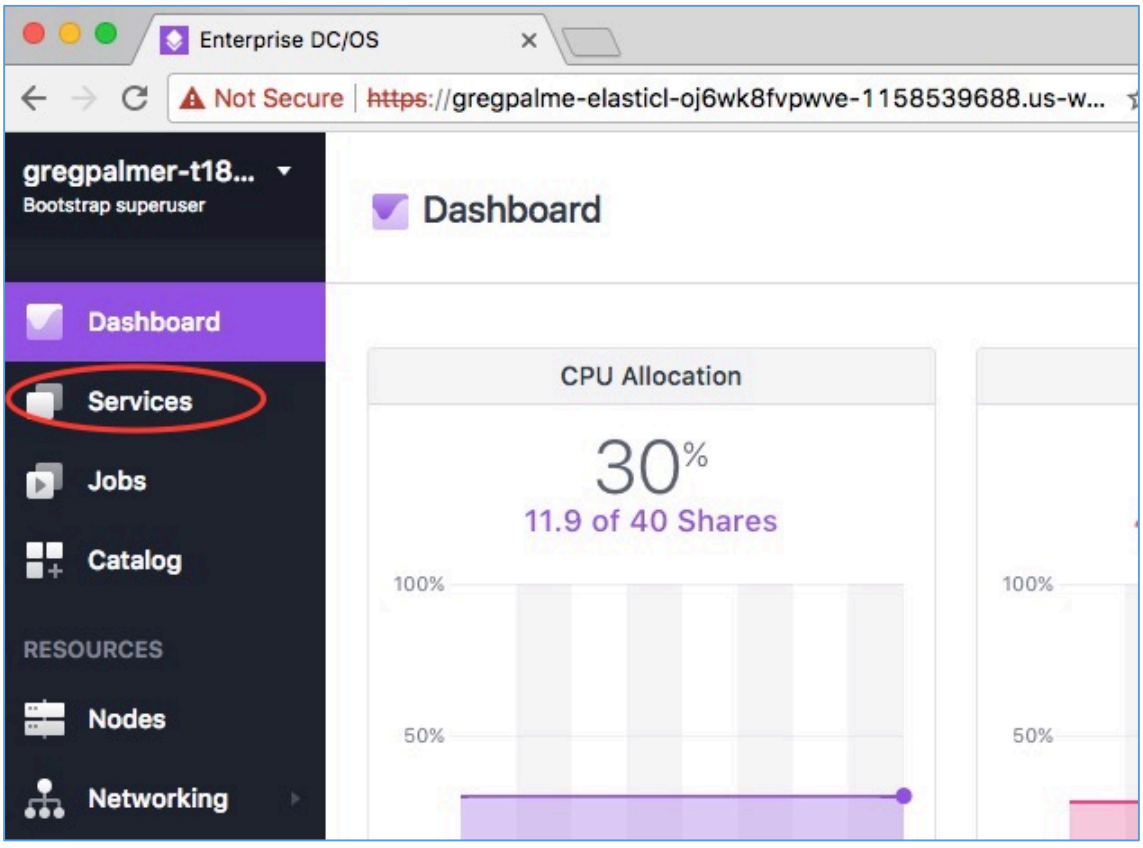
Exit out of the HDFS client and return to your DC/OS CLI session. Use this commands:

```
$ exit
```

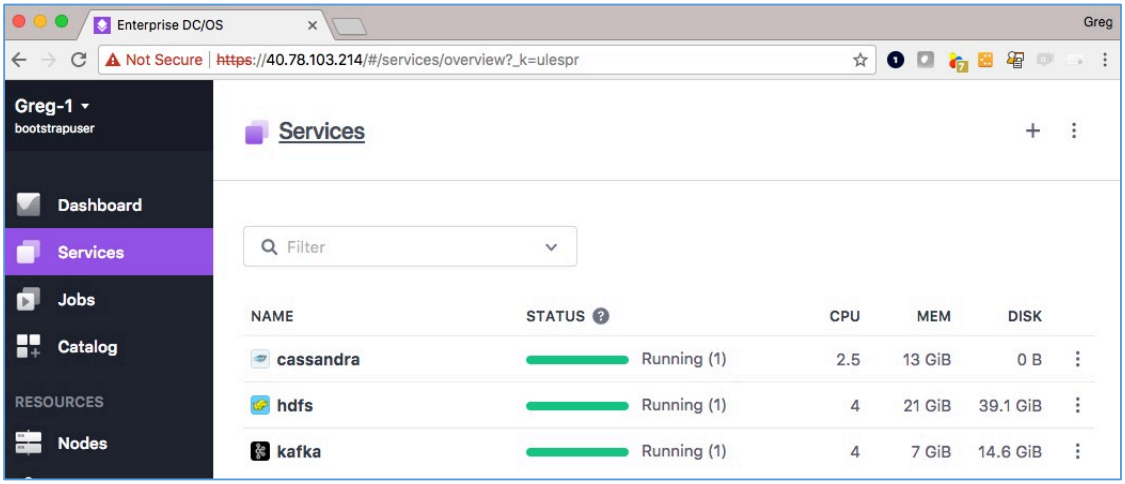
Cassandra, Kafka, and HDFS running on DC/OS

At this point in the tutorial, you have configured and deployed three data services on the DC/OS cluster. In the DC/OS Dashboard, you can click on the Services option on the left menu to see the services running.

Click on the Service menu option:



Notice that each of the data services frameworks has its own CPU, MEM and DISK resource allocations. If you click on the Kafka service, you will see that three Kafka brokers have been started on three different DC/OS agent nodes. Later, you can experiment with modifying the configuration of Kafka, Cassandra and HDFS and add brokers, Cassandra nodes and data nodes to the services.



Spark History Server

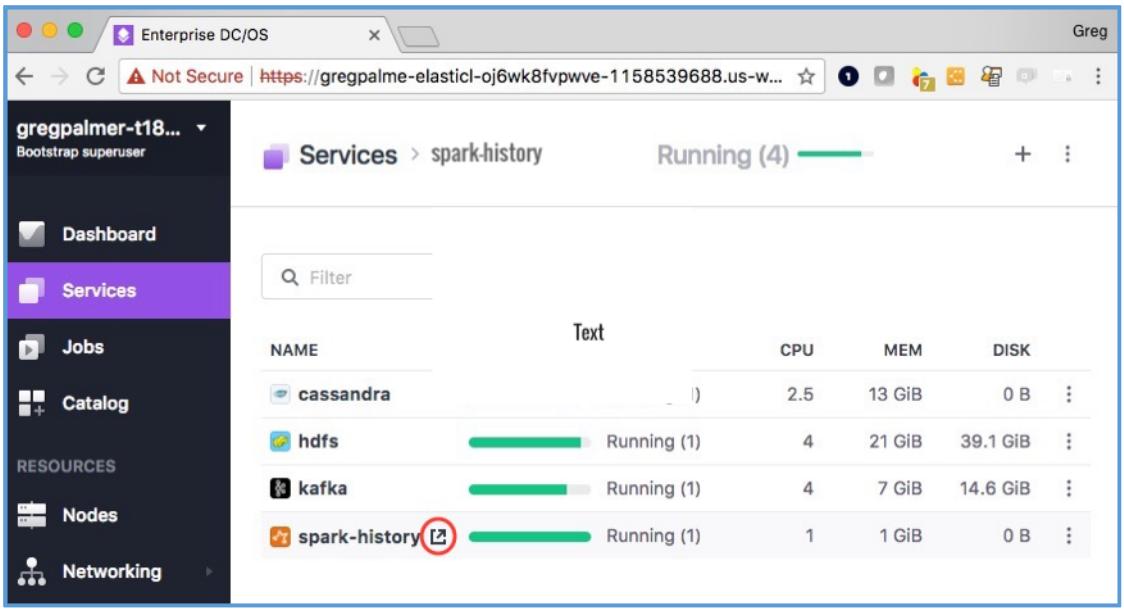
The Spark History Server can be used to track the progress and history of the Spark jobs you submit on the DC/OS cluster and the History Server stores its data in the HDFS directory you created above (hdfs:///history). You will not be using a Spark History Server package in the DC/OS Catalog for this part of the tutorial, instead, you will start the Spark History Server using Marathon and an application configuration file in JSON format.

From your DC/OS CLI session, create the JSON file using these commands:

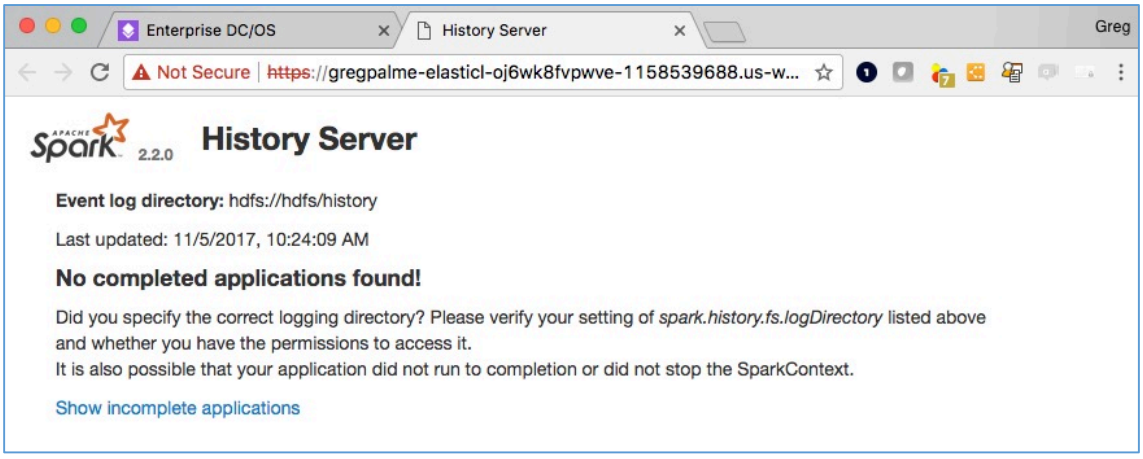
```
$ cat > spark-history-options.json <<EOF
{
  "name": "spark-history",
  "hdfs-config-url": "http://api.hdfs.marathon.l4lb.thisdcos.directory/v1/endpoints"
}
EOF

$ dcos package install spark-history --options=spark-history-options.json --yes
```

Once the Spark History Server starts up, you can view the history server console by clicking on the console launch icon on the DC/OS Dashboard.

From the DC/OS Dashboard's Services panel, view the Spark History Server running. Place your mouse cursor just to the right of the spark-history service name and you will see an arrow icon appear. Click on that icon to launch the Spark History Server console.



Because you have not yet launched the Spark service on the DC/OS cluster and you have not yet submitted any Spark jobs, you will not see any job history at this time.



Next you will configure and deploy the Spark service on the DC/OS cluster.

Apache Spark

Apache Spark is a fast and general-purpose cluster computing system for big data. It provides high-level APIs in Scala, Java, Python, and R, and an optimized engine that supports general computation graphs for data analysis. It also supports a rich set of higher-level tools including Spark SQL for SQL and DataFrames, MLlib for machine learning, GraphX for graph processing, and Spark Streaming for stream processing. For more information, see the Apache Spark documentation at:

http://spark.apache.org/documentation.html

DC/OS Apache Spark consists of Apache Spark with a few custom commits. See:

https://github.com/mesosphere/spark

It also has some DC/OS-specific packaging. See:

https://github.com/mesosphere/spark-build

DC/OS Apache Spark includes:

- Mesos Cluster Dispatcher
- Spark History Server
- DC/OS Apache Spark CLI
- Interactive Spark shell

Benefits

- Utilization: DC/OS Apache Spark leverages Mesos to run Spark on the same cluster as other DC/OS services
- Improved efficiency
- Simple Management

- Multi-team support
- Interactive analytics through notebooks
- UI integration
- Security, including file- and environment-based secrets

Features

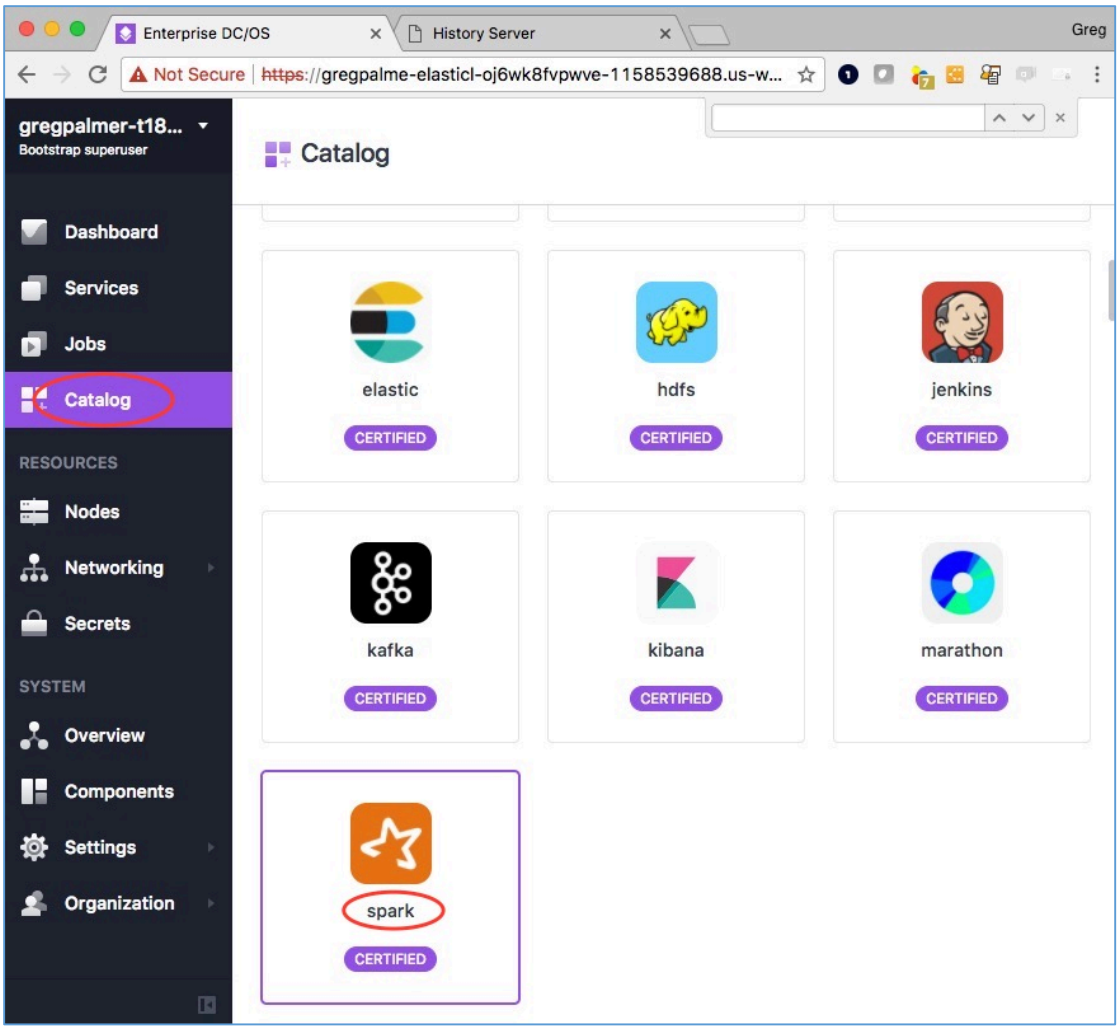
- Multiversion support
- Run multiple Spark dispatchers
- Run against multiple HDFS clusters
- Backports of scheduling improvements
- Simple installation of all Spark components, including the dispatcher and the history server
- Integration of the dispatcher and history server
- Zeppelin integration
- Kerberos and SSL support

You can review the Mesosphere DC/OS Spark documentation here:

https://docs.mesosphere.com/service-docs/spark/

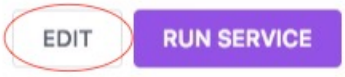
Configure and Deploy Spark on DC/OS

In the DC/OS Dashboard, click on the Catalog menu option on the left and display the data services packages in the DC/OS Catalog. Then click on the Spark package.



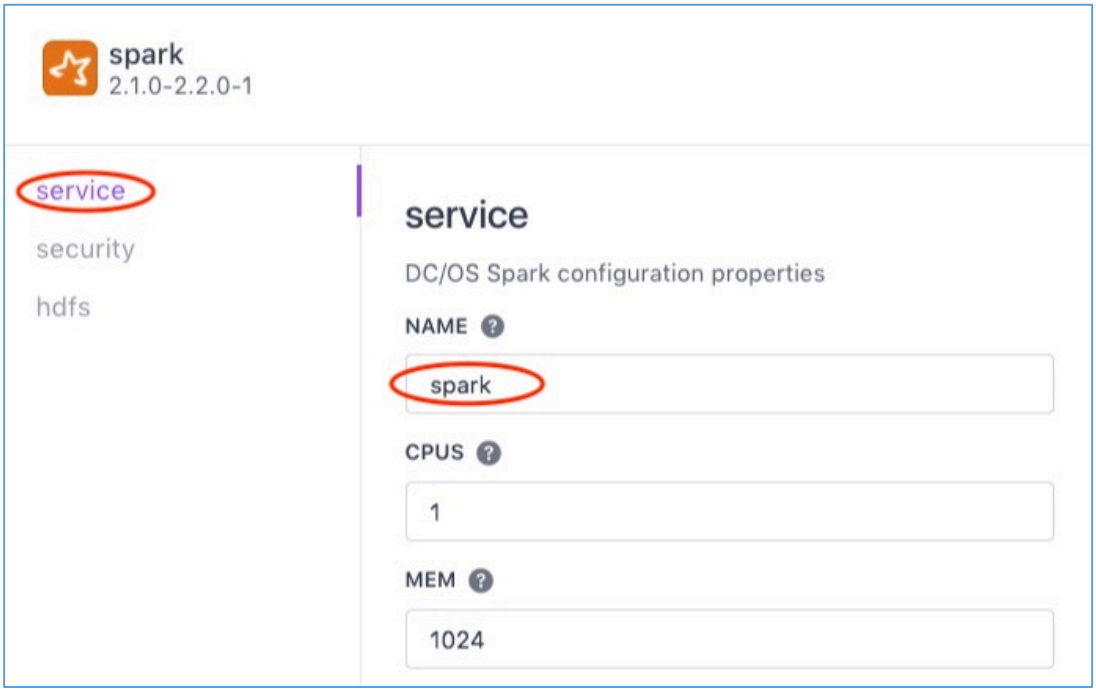
You will see some details about the HDFS service on DC/OS. Click on the REVIEW & RUN button.

Then click on the EDIT button to modify the configuration.

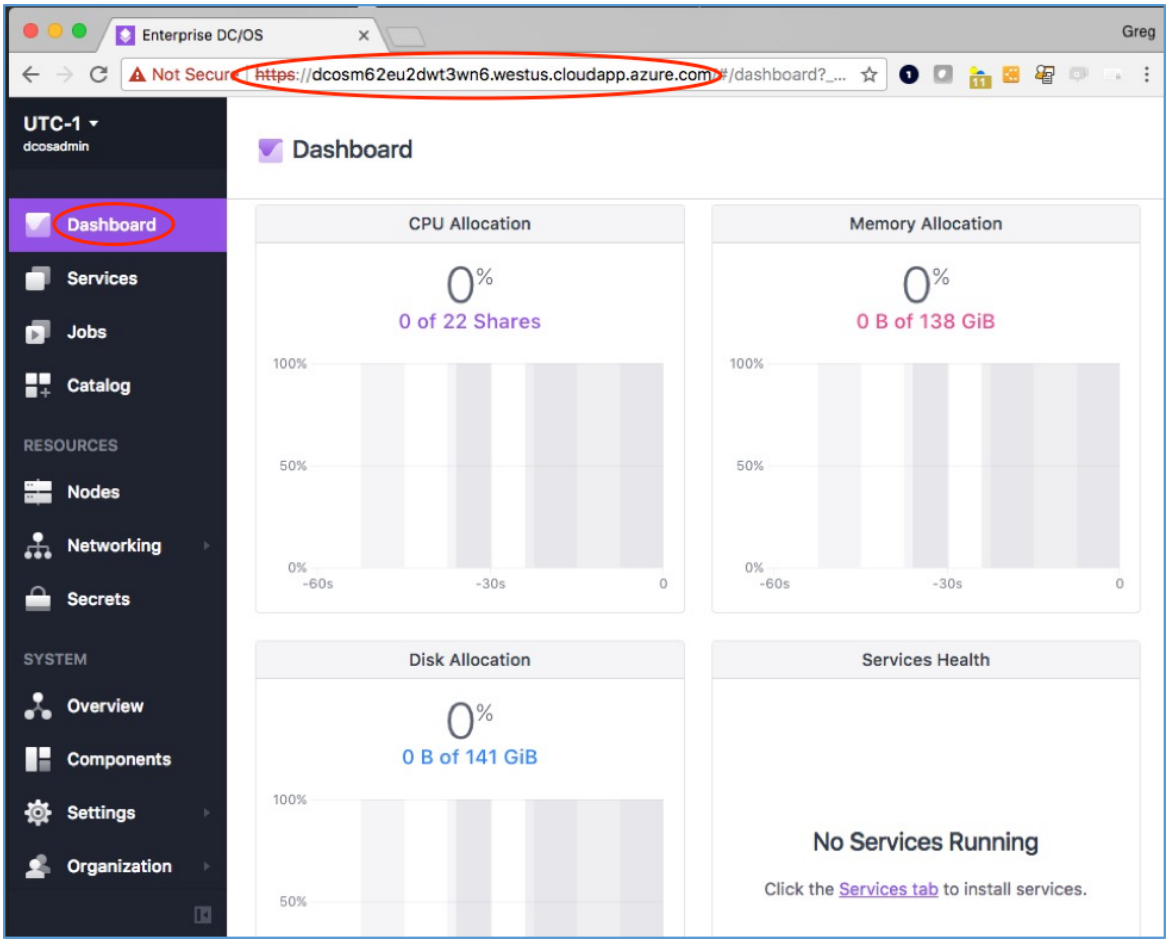


The DC/OS Spark package configuration screens allow you to modify the default configuration and in this tutorial you will be modifying the URL to the HDFS service.

Click on the service category and keep the name of the Spark service as spark.

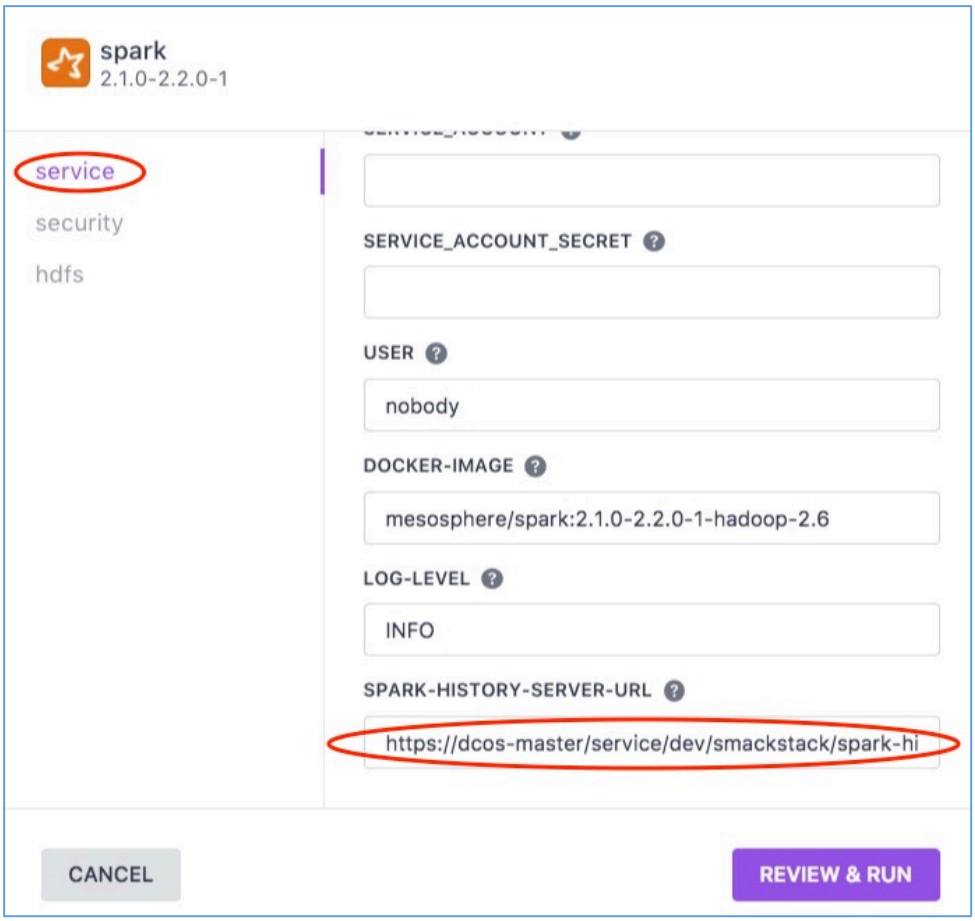


Also in the service category, enter the URL to the Spark History Service that you deployed previously. To get this URL, click on the Dashboard panel in your DC/OS Web console. Copy that Web address into your paste buffer, but only include up to the main hostname or IP address. Do not include the remainder of the Web address. See below:



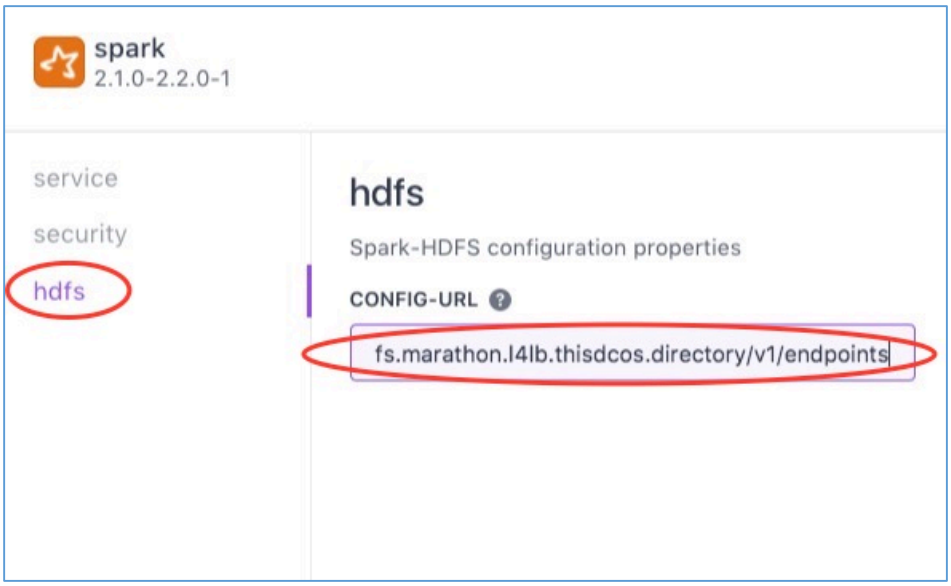
In the SPARK-HISTORY-SERVER-URL field, paste the contents of your paste buffer container and add the rest of the specification for the Spark History Server like this:

<pasted Web address>/service/dev/smackstack/spark-history



Finally, change the URL to your HDFS service so that the Spark service can download the core-site.xml and hdfs-site.xml configuration scripts. Enter this value in the CONFIG-URL field:

http://api.hdfs.marathon.l4lb.thisdcos.directory/v1/endpoints

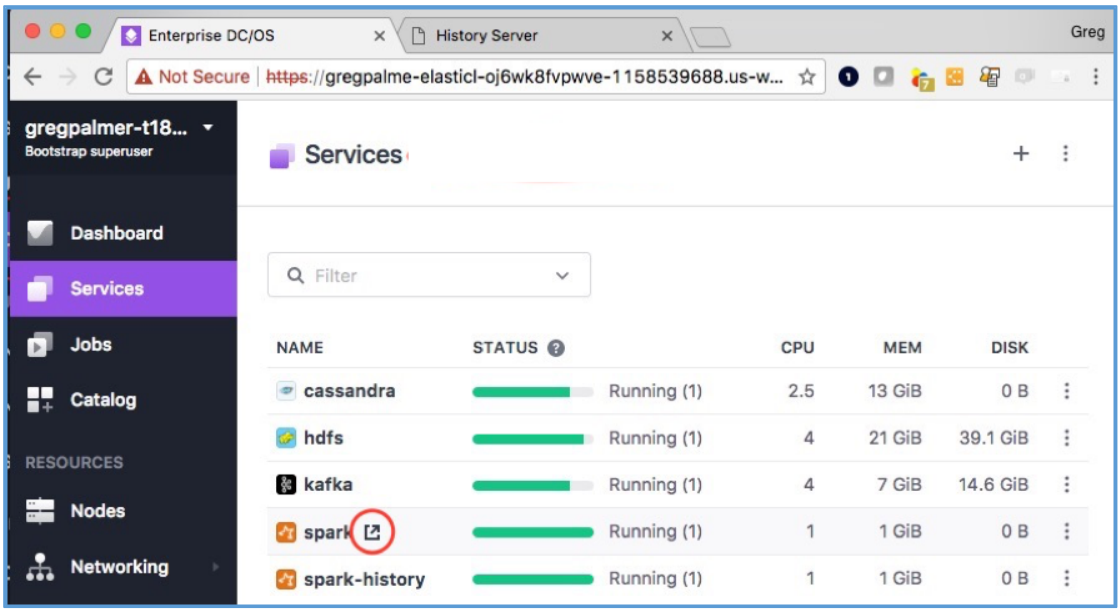


Now that you have completed the changes needed to deploy the Spark service on DC/OS, click the REVIEW & RUN button.

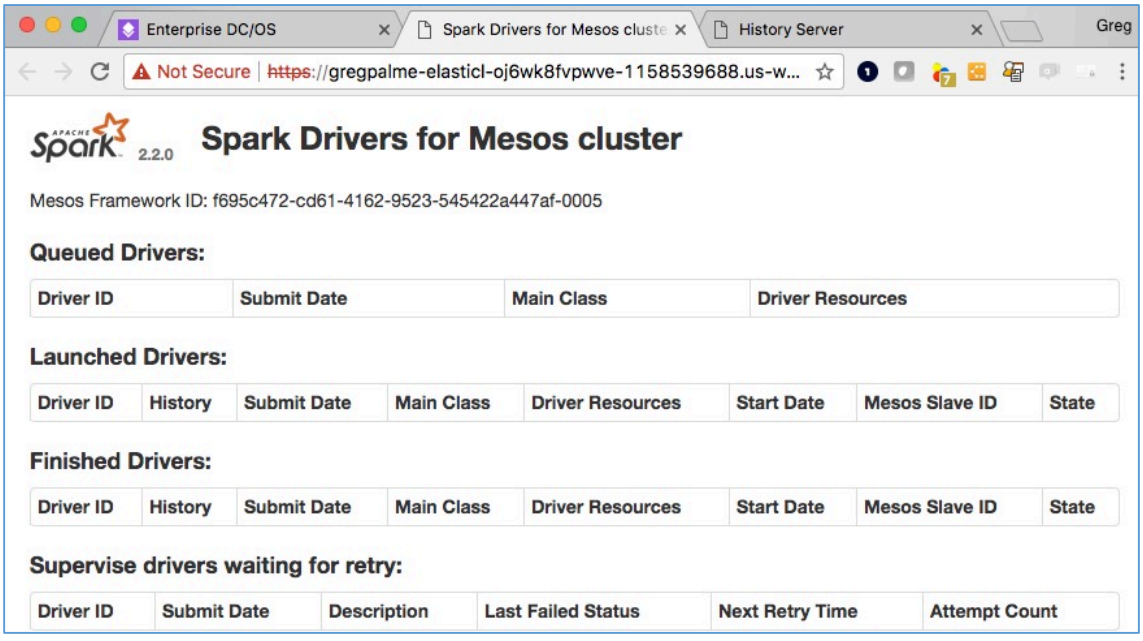
Then click the RUN SERVICE button.

After the Spark service starts up and passes its health check, you will see the tasks running on the DC/OS Mesos cluster. Click on the Services menu option on the left and then click on the spark service name and you will see the Spark Mesos Dispatcher task running on one of the DC/OS agent nodes.

You can view the Spark dispatcher console by clicking on the arrow icon just to the right of the spark service name.



The Spark Mesos Dispatcher console will display in a new Web browser tab. Because you have not yet submitted a Spark job, no Spark drivers will be shown.

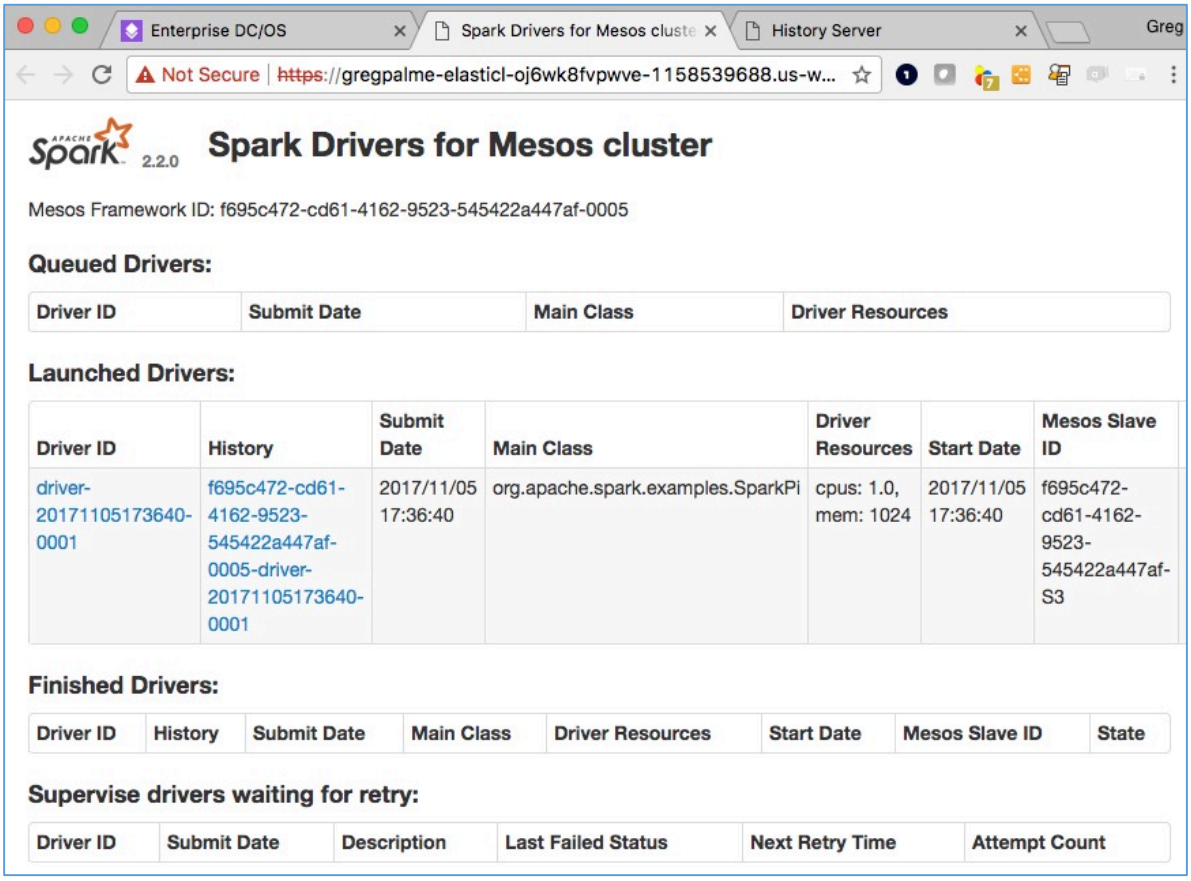


Submit Your First Spark Job

Now that the Spark Service and the Spark History Service are running on the DC/OS cluster, you can submit your first Spark job.

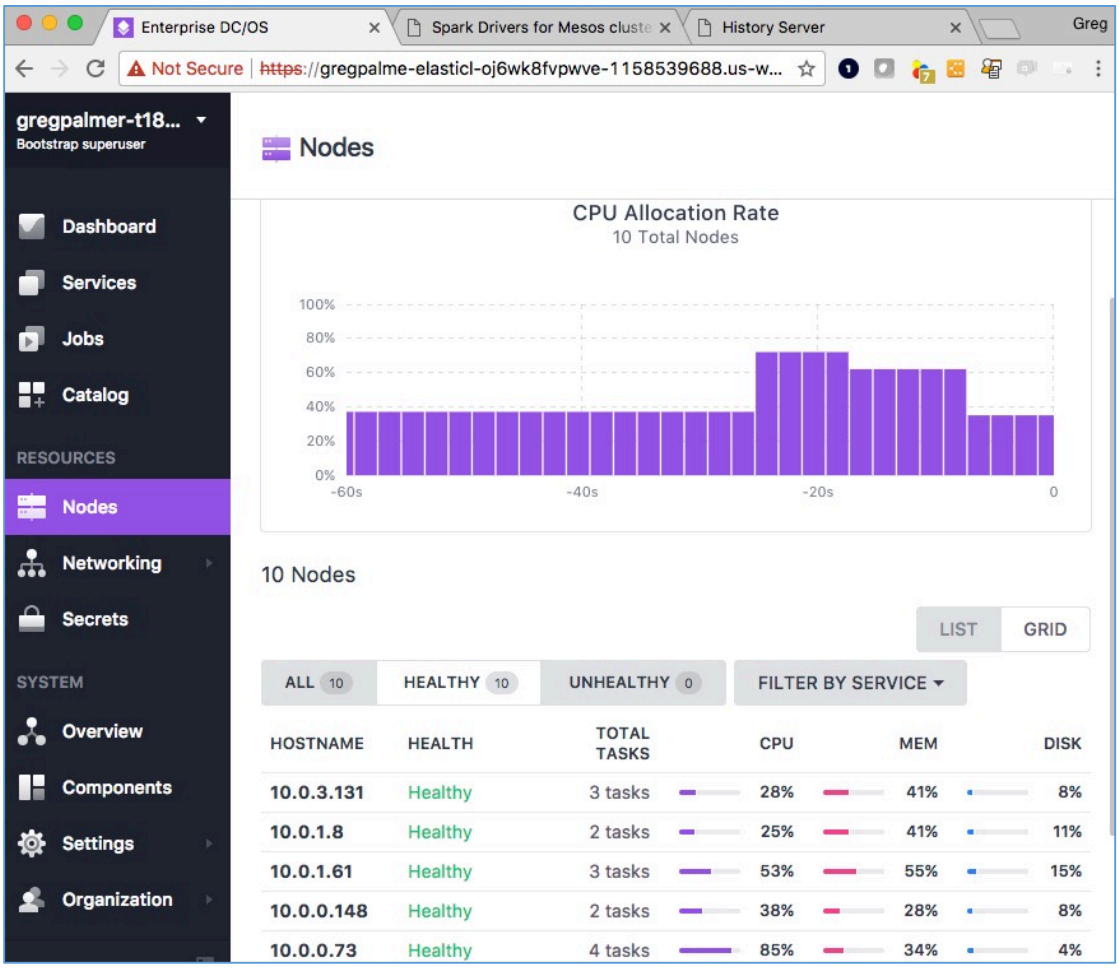
```
$ dcos package install spark --cli --yes

$ dcos spark run --name 'spark' --submit-args='--conf spark.eventLog.enabled=true --
conf spark.eventLog.dir=hdfs://hdfs/history --conf spark.mesos.coarse=true --conf
spark.cores.max=4 --conf spark.executor.memory=1g --driver-cores 1 --driver-memory
1g --class org.apache.spark.examples.SparkPi
https://downloads.mesosphere.com/spark/assets/spark-examples_2.10-1.4.0-
SNAPSHOT.jar 50'
```

Once your Spark job is submitted successfully, you can see the Spark Driver program that was launched by the Spark Dispatcher. Click on the Web browser tab that contains the Spark Dispatcher console that you opened previously. Your new job shows up in the Launched Drivers list.

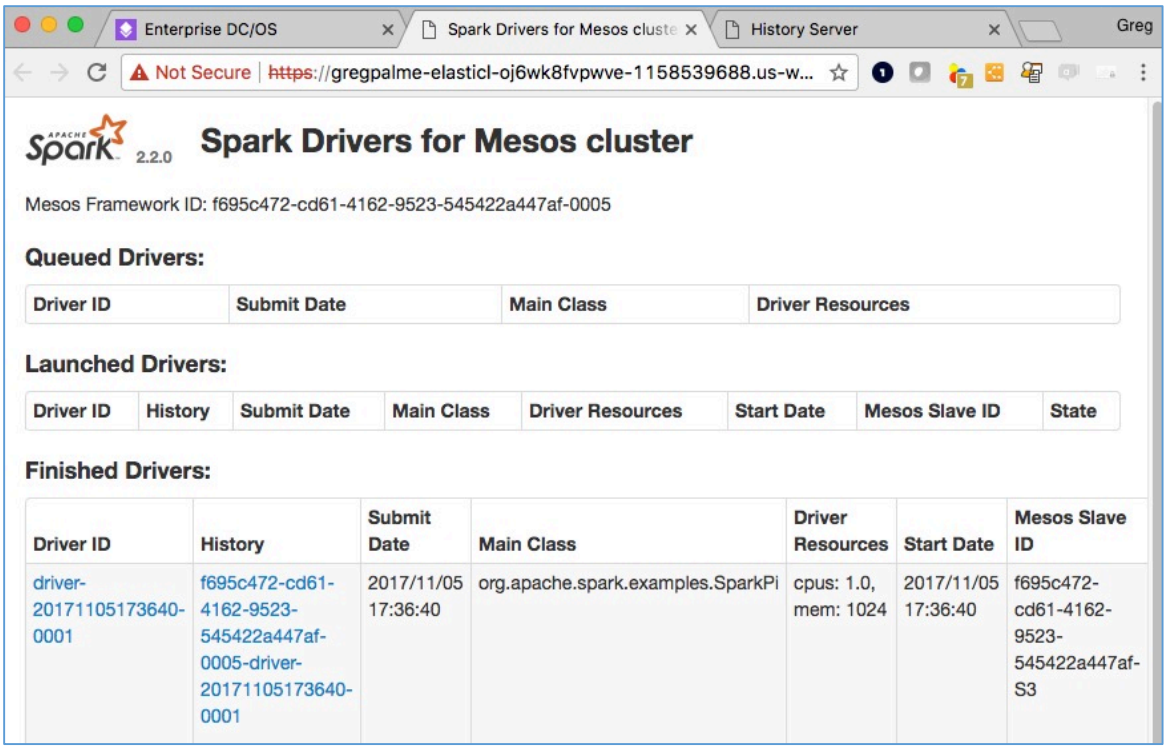


While your Spark job is running, the Spark Driver program will launch tasks that will use CPU and memory resource offers from the Mesos scheduler. Open the

DC/OS Dashboard's Nodes panel and you will see more CPU and memory being allocated on the Mesos cluster.



Once your Spark job is completed, it will be shown in the Finished Drivers list.



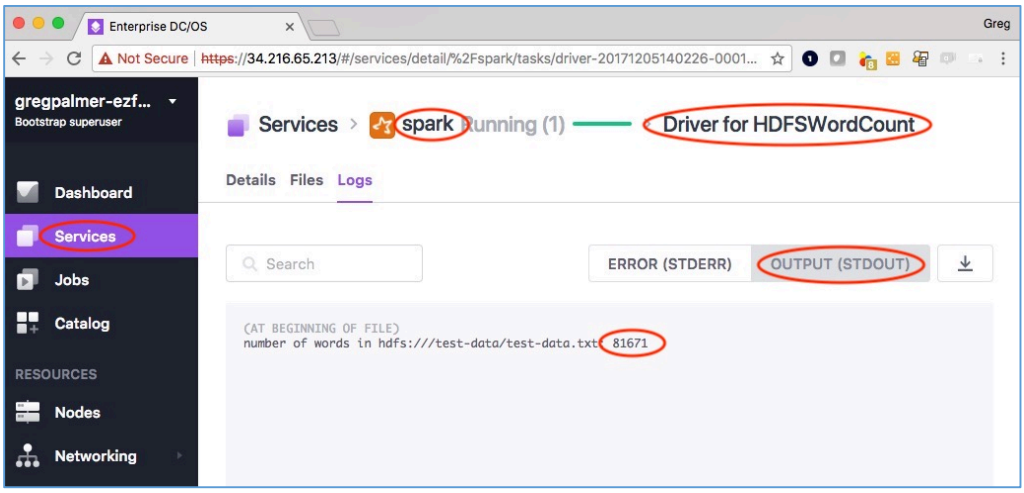
Submit a Spark Job that Uses HDFS

Previously, you created a test data file with 1000 lines of data and uploaded it to the HDFS service running on your DC/OS cluster. In this section, you will submit a Spark job that reads the contents of that file in HDFS and counts the number of lines. From your DC/OS command line, run these commands:

```
$ dcos package install --cli spark

$ dcos spark run --name 'spark' --submit-args='--conf spark.eventLog.enabled=true --conf spark.eventLog.dir=hdfs://hdfs/history --conf spark.mesos.coarse=true --conf spark.cores.max=4 --conf spark.executor.memory=1g --driver-cores 1 --driver-memory 1g --class HDFSWordCount http://infinity-artifacts.s3.amazonaws.com/spark/sparkjob-assembly-1.0.jar hdfs:///test-data/test-data.txt'
```

Just like before, you can view the progress of the Spark job by viewing the tasks on the DC/OS Dashboard, or the Spark Dispatcher Web console, or the Spark History Server Web console. In the Spark Service task list, you can click on the

“logs” icon for the HDFSWordCount driver task and then click on the STDOUT tab to view the results of the Spark job reading the file from HDFS.



Kafka Revisited

In this section of the tutorial, you will use the Kafka messaging environment to show how producers can put data into Kafka topics in a reliable and redundant fashion and how consumers can retrieve that data from the topics.

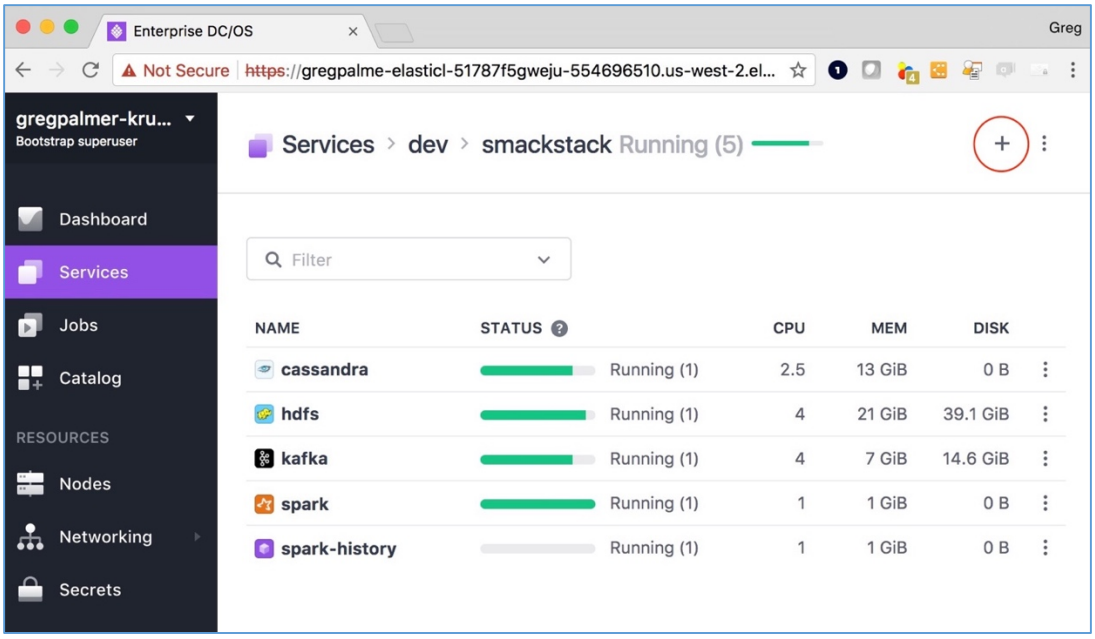
Show the current list of Kafka brokers and topics with these commands:

```
$ dcos package install --cli kafka --yes

$ dcos kafka broker list

$ dcos kafka topic list
```

Create a Kafka topic called my-topic using this command.

```
$ dcos kafka topic create my-topic --partitions=3 --replication=3
```

Run a containerized application to read from the new Kafka topic. From the DC/OS Dashboard, click on the Services menu option. Then click on the plus sign to create a new service manually.



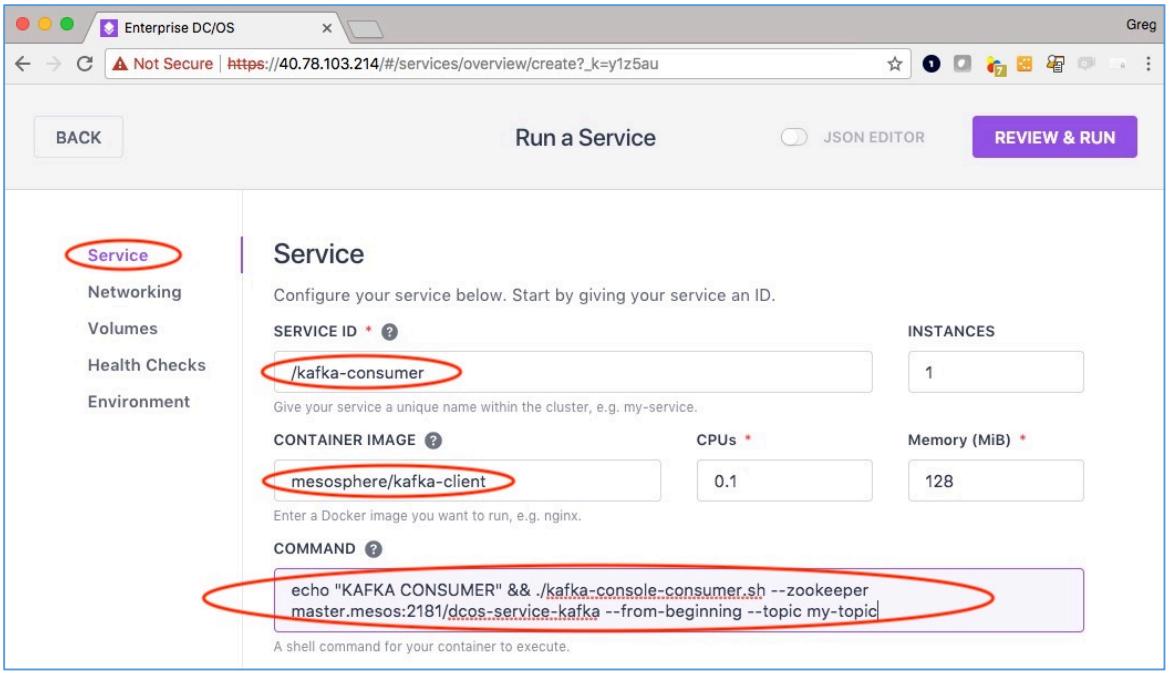
Click on the Single Container button to display the Run a Service page.

Fill in the following configuration settings:

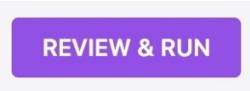
SERVICE ID: kafka-consumer

CONTAINER IMAGE: mesosphere/kafka-client

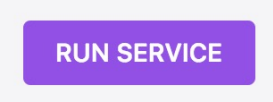
CMD: echo "#### KAFKA CONSUMER ####" && ./kafka-console-consumer.sh --zookeeper master.mesos:2181/dcos-service-kafka --from-beginning --topic my-topic



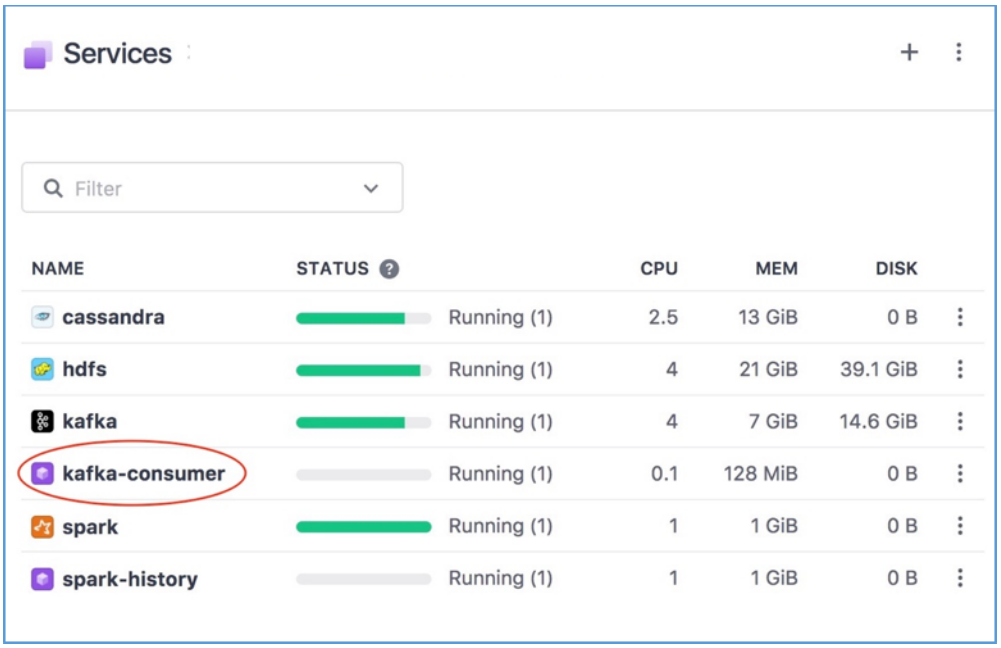
Click the RUN & REVIEW button:



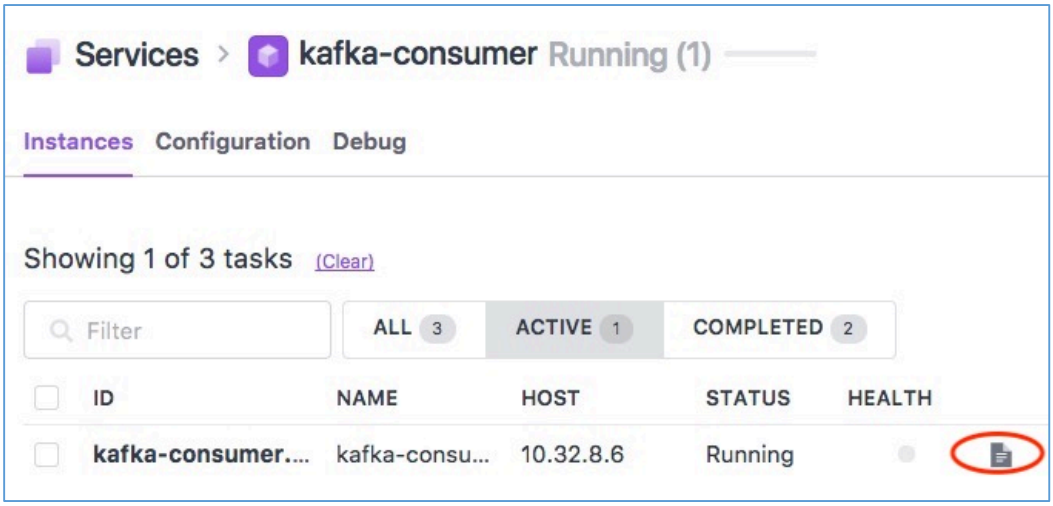
Then click the RUN SERVICE button to run this new service:



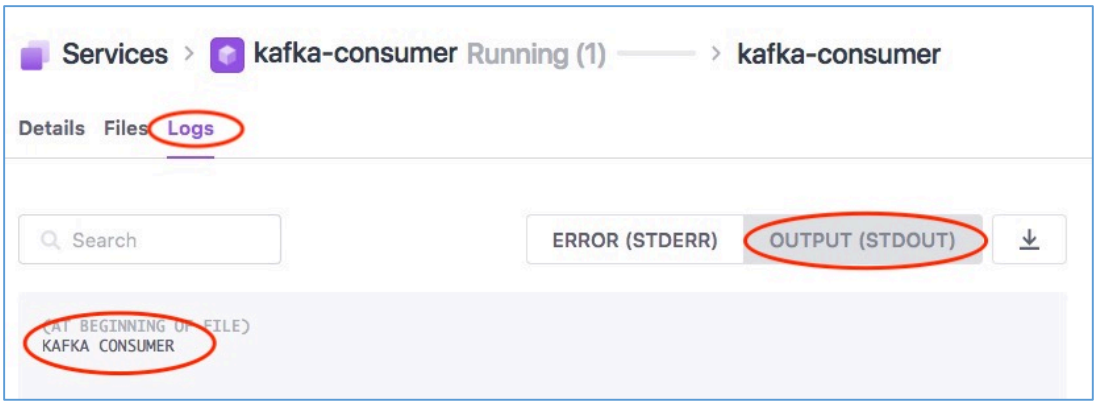
When your service completes the startup process, it will show up in the list of services running in the application group.



Let's view the output of the service you just started. Click on the service name, kafka-consumer, and then click on the logs icon (the page icon) on the right of the service name.



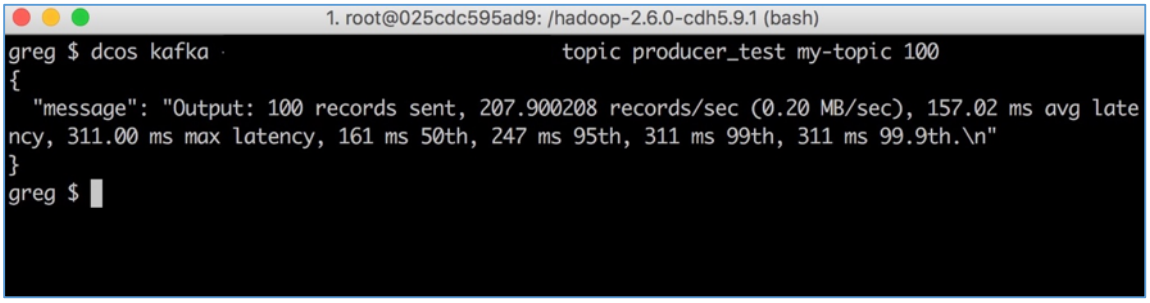
You will see the STDERR and STDOUT log files for this service. Click on the STDOUT button to see the current standard output.



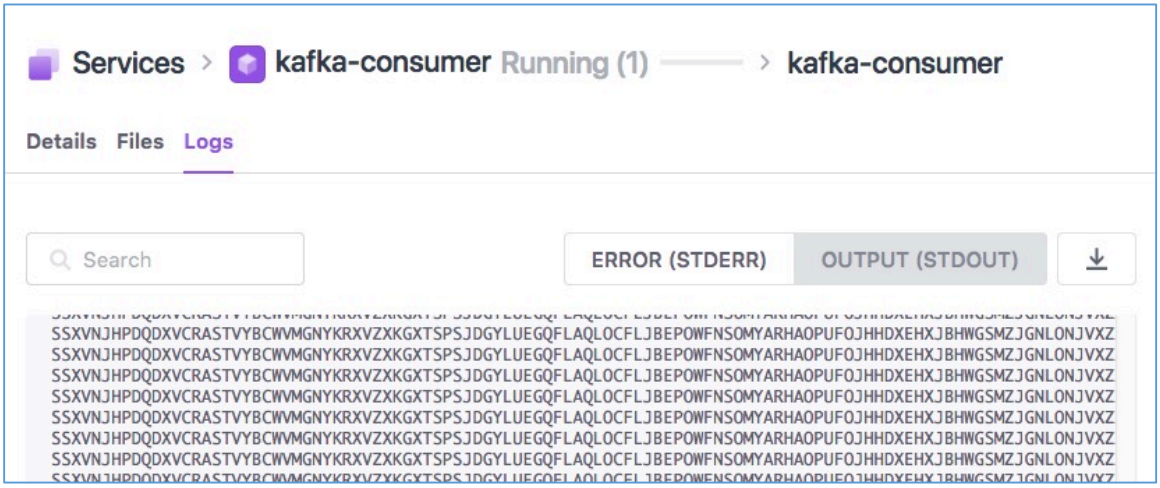
Produce some messages in the new topic using the command:

```
$ dcos kafka topic producer_test my-topic 100
```

This will generate some test data and place entries into the my-topic message queue in Kafka.



Then go back to your STDOUT console for the kafka-consumer service and view the Kafka messages.



```
$ dcos kafka topic producer_test my-topic 100

$ dcos spark run --submit-args='--conf spark.eventLog.enabled=true --conf
spark.eventLog.dir=hdfs://hdfs/history --conf spark.mesos.coarse=true --conf
spark.cores.max=4 --conf spark.executor.memory=1g --driver-cores 1 --driver-memory
1g --class org.apache.spark.examples.streaming.KafkaWordCount
https://downloads.mesosphere.com/spark/assets/spark-examples_2.10-1.4.0-
SNAPSHOT.jar mesos://leader.mesos:5050 zk-1.zk,zk-2.zk,zk-3.zk my-consumer-group
my-topic 1'
```

Summary

This tutorial guided you through the process of deploying the components that make up the SMACK Stack and also showed you how to run a Spark job that reads from the HDFS service and from the Kafka service. Additionally, this tutorial showed you how to test the Kafka service with consumers and producers.

If you would like to quickly deploy these components you can use the pre-built startup script named start-smackstack.sh found here in the scripts directory:

https://github.com/gregpalmr/smack-stack-tutorial

If you would like to review the Mesosphere Advanced SMACK Stack tutorial, you can find that here:

https://github.com/gregpalmr/smack-stack-advanced-tutorial