Unravel 4.0-4.1

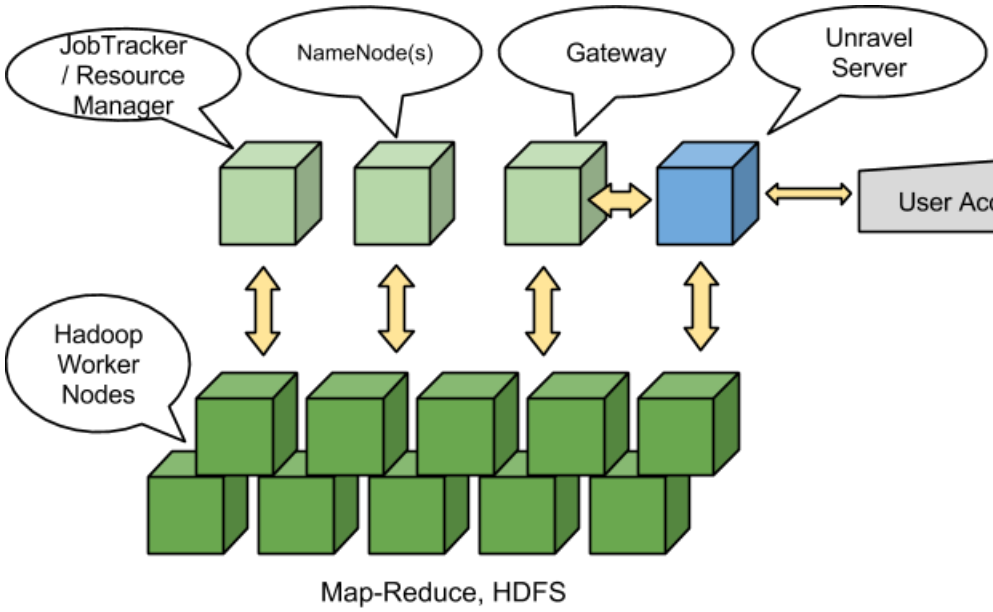
Chapter 1

Overview

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1

Where Does Unravel Server Reside?

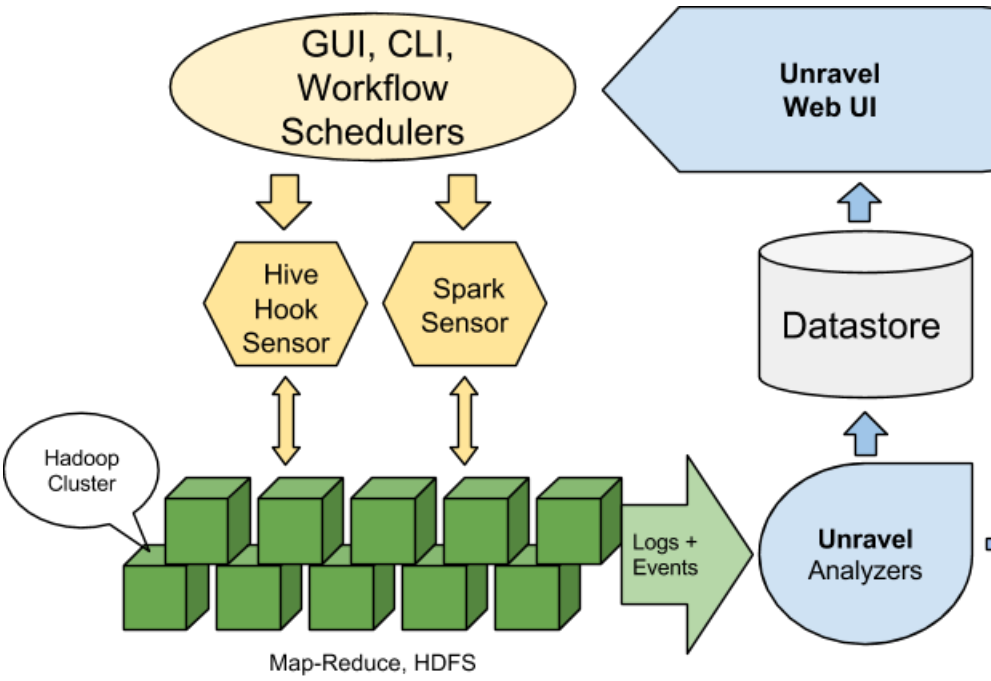
Unravel Server is positioned in a Hadoop cluster on its own Gateway/Edge node to communicate with HDFS and TaskTracker nodes, and to enable client access to Unravel Web UI. A basic deployment takes two hours or less.



The Unravel Server on a Gateway/Edge Node in a Hadoop Cluster

What Does a Basic Deployment Provide?

Once deployed, Unravel Server begins to collect information from relevant services like YARN, Hive, Oozie, Spark, Pig, and MapReduce jobs via HDFS, analyze this information, and store its analyses in its database. Unravel Web UI displays information from the database, as well as real-time event streams it receives directly from Unravel Server. The figure below illustrates the flow of information from the Hadoop cluster, to Unravel Server, to Unravel Web UI.

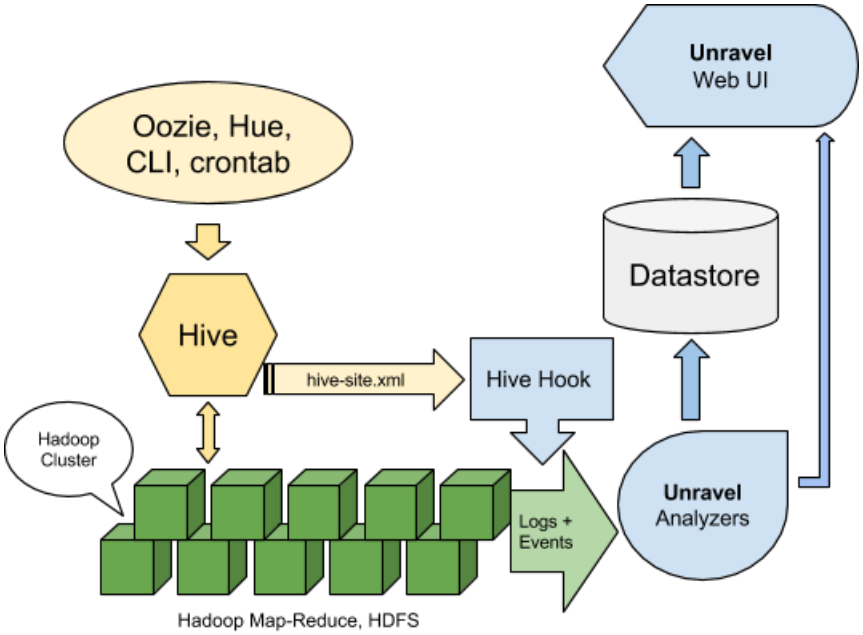


Information Flow from the Hadoop Cluster to Unravel Server to Unravel Web UI

What Are Advanced Deployment Options?

After basic deployment, advanced setup steps are optional, but may include:

- Deploying Unravel Sensors for Hive and Spark, which can push additional metadata to Unravel Server.



- In YARN, enabling log aggregation.
- In Oozie, pushing information via the Oozie REST API.

Attachments:

- overview3.png
- overview2.png
- overview1.png
- worddav95f1e8fd13958fae391afc8e6f9ad03d.png

Chapter 2

Installation Guides

Topics:

Installation Guide for Cloudera Distribution Including Apache Hadoop (CDH), with Cloudera Manager (CM)

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1

This guide is compatible with

- CDH 4.5 - 5.10
- Hive versions 0.10.0 through 1.2.x
- Spark versions 1.3, 1.5, 1.6, 2.0

Ordered Steps

- Page: Part 1: Install Unravel Server on CDH+CM
- Page: Part 2: Install Unravel Sensor Parcel on CDH+CM
- Page: Part 3: Additional Topics for CDH

Part 1: Install Unravel Server on CDH+CM

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Installation Guide for Cloudera Distribution Including Apache Hadoop (CDH), with Cloudera Manager (CM)

Introduction

This topic explains how to deploy Unravel Server 4.0 on a CDH gateway/edge node. For instructions that correspond to older versions of Unravel Server, contact Unravel Support.

Workflow Summary

1. Pre-installation check
2. Configure the host machine.
3. Install the Unravel Server RPM on the host machine.
4. Configure Unravel Server (basic/core options).
5. Log into Unravel Web UI.
6. (Optional) Configure Unravel Server (advanced options).

Table of Contents

Pre-Installation Check

The following installation requirements must be met for successful installation of Unravel.

Platform Compatibility

- CDH 4.5 - 5.10
- Hadoop 1.x - 2.x
- Kerberos
- Hive 0.9.x - 1.2.x
- Spark 1.3.x - 2.0.x

Hardware

- Architecture: x86_64
- Cores: 8
- RAM: 64GB minimum
- Disk: `/usr/local/unravel` with 2.5GB free minimum
- Disk: `/srv/unravel` with 500GB free minimum
- For 10,000+ MR jobs per day, two or more gateway/edge nodes are recommended

Software

- Operating System: RedHat/Centos 6.4 - 7.3
- `libaio.x86_64` installed
- `SELINUX=permissive` (or disabled) should be set in `/etc/selinux/config`
- HDFS+Hive+YARN client/gateway, Hadoop and Hive commands in `PATH`
- If Spark is in use, Spark client gateway
- LDAP (AD or Open LDAP) compatible for Unravel Web UI user authentication (Open signup by default)
- On Unravel Edge-node server, please **do not** have zookeeper installed in same server

Access Permissions

- If Kerberos is in use, a keytab for principal hdfs (or read-only equivalent) is required for access to:
    - Access to YARN's "done dir" in HDFS
    - Access to YARN's log aggregation directory in HDFS
    - Access to Spark event log directory in HDFS
    - Access to file sizes under HIve warehouse directory
- Access to YARN Resource Manager REST API
    - principal needs right to find out which RM is active
- JDBC access to the Hive Metastore (read-only user is sufficient)

Network

- Port 3000 (or 4020) from users and entire cluster to Unravel Web UI
- HDFS ports open from Hadoop cluster to Unravel Server(s)
- For MR1, TaskTracker port open from Hadoop cluster to Unravel Server(s)
- For MR1/YARN, Hive Metadata DB port open to Unravel Server(s) for partition reporting
- UDP and TCP port 4043 open from entire cluster to Unravel Server(s)
- For Oozie, port 11000 open from Unravel Server(s) to the Oozie server
- Resource Manager (RM) port 8032 from Unravel Server(s) to the RM server(s)
- Port 4176, 4181 through 4189, 3316, 4091 must be available for localhost communication between Unravel daemons or services

1. Configure the Host Machine

Allocate a Cluster Gateway/Edge/Client Host with HDFS Access

Use Cloudera Manager to create the gateway configuration for the Unravel server(s) that has client roles for HDFS, YARN, Spark, Hive.

2. Install the Unravel Server RPM on the Host Machine

Get the Unravel Server RPM

Copy the RPM from the Unravel distribution server to the host machine using the username and password given to you by Unravel Support:

- For the free trial version:

```
scp "unraveltrial@trial.unraveldata.com:unravel-4.0-*.x86_64.rpm" .
```

- For the enterprise version:

```
scp $USER@dist.unraveldata.com:unravel-4.0-*.x86_64.rpm.$timestamp .
```

The precise RPM filename will vary. The version has the structure `x.y.b` where `b` is a build number that is imposed as an RPM epoch which means it takes precedence over version numbers for determining what is more up-to-date. The Unravel build numbers always increase and are allocated so that they respect the ordering of `x.y` versioning.

Install the Unravel Server RPM

Replace the asterisks as needed to be more selective.

```
sudo rpm -U unravel-4.0*.x86_64.rpm*
```

The installation creates `/usr/local/unravel/` which contains the executables, scripts, and settings. User `unravel` is created. The initial internal database and other durable state are put in `/srv/unravel/` for larger storage. By default, the installation supports YARN. The master configuration file is in `/usr/local/unravel/etc/unravel.properties` and the logs are in `/usr/local/unravel/logs/`. The RPM installation creates user `unravel` if it does not already exist; `/etc/init.d/unravel_*` scripts for controlling its services as well as `/etc/init.d/unravel_all.sh` which can be used to manually stop, start, and get status of all daemons in proper order. The RPM installation also creates an HDFS directory for Hive Hook information collection. During initial install, a bundled database is used. This can be switched to use an externally managed MySQL for production. (The bundled database root mysql password will be stored in `/root/unravel.install.include` during installation.)

Do Host-Specific Post-Installation Actions

For CDH, there are no host-specific post-installation actions.

3. Configure Unravel Server (Basic/Core Options)

Enable/Disable Optional Daemons

Depending on your workload volume or kind of activity, you can enable or disable optional daemons at this point.

- If you are not using Oozie, disable `unravel_os3`:

```
sudo chkconfig unravel_os3 off
```

- If you are not using Spark, disable `unravel_sw_1`:

```
sudo chkconfig unravel_sw_1 off
```

- If you have 10000-20000 jobs per day, enable these workers:

```
sudo chkconfig --add unravel_ew_2
sudo chkconfig --add unravel_hhw_2
sudo chkconfig --add unravel_jcw2_2
```

- If you have 20000-30000 jobs per day, enable these workers:

```
sudo chkconfig --add unravel_ew_3
sudo chkconfig --add unravel_hhw_3
sudo chkconfig --add unravel_jcw2_3
```

- If you have more than 30000 jobs per day, enable these workers:

```
sudo chkconfig --add unravel_ew_4
sudo chkconfig --add unravel_hhw_4
sudo chkconfig --add unravel_jcw2_4
```

Modify `unravel.properties`

All values in unravel.properties can be modified through the in Unravel Web UI's configuration wizard, for CDH. Furthermore, some properties can *only* be modified by running the configuration wizard. Therefore, for CDH, this section is optional.

- Open `/usr/local/unravel/etc/unravel.properties` with `vi`.

```
sudo vi /usr/local/unravel/etc/unravel.properties
```

- Edit the values in `unravel.properties` using the guidelines and descriptions in the table below.

`com.unraveldata.advertised.url`	Defines the Unravel Server URL for HTTP traffic.	`com.unraveldata.advertised.url=http://LAN_DNS:3000`
`com.unraveldata.customer.organization`	Identifies your installation for reporting purposes.	`com.unraveldata.customer.organizati`
`com.unraveldata.history.maxSize.weeks`	Sets retention for search data.	`com.unraveldata.history.maxSize.wee`
`com.unraveldata.hive.hook.topic.num`	**Optional** Defines the number of threads. Default is 1. Depending on job volume, increase this property to *N* where *N* is between 1 and 4, or roughly *ThousandJobsPerDay*/10.	`com.unraveldata.hive.hook.topic.num`
`com.unraveldata.login.admins`	Defines the usernames that can access Unravel Web UI's admin pages. Default is `admin`.	`com.unraveldata.login.admins=admin`
`com.unraveldata.s3.batch.monitoring`	**Optional** Defines the monitoring frequency. Default is 300 seconds (5 minutes). Set this property to 60 for lower latency.	`com.unraveldata.s3.batch.monitoring`

If Kerberos is Enabled:

Add authentication for HDFS...

(a) Create a keytab for unravel for daemons that run as `unravel` and put the file in `/usr/local/unravel/etc/unravel.keytab` (for example).

(b) Create a keytab for hdfs for the Unravel daemons that run as user `hdfs` and put the file in `/etc/keytabs/hdfs.keytab` (for example).

(c) Tell Unravel Server about it (env var for hdfs keytab location; substitute correct hostname):

```
echo "export HDFS_KEYTAB_PATH=/etc/keytabs/hdfs.keytab
export HDFS_KERBEROS_PRINCIPAL=hdfs/myhost.mydomain@MYREALM
" | sudo tee -a /usr/local/unravel/etc/unravel.ext.sh
```

(d) Add properties for Kerberos:

```
echo "
com.unraveldata.kerberos.principal=unravel/myhost.mydomain@MYREALM
com.unraveldata.kerberos.keytab.path=/usr/local/unravel/etc/unravel.keytab
" | sudo tee -a /usr/local/unravel/unravel.properties
```

You can find the principal by using 'klist -kt KEYTAB_FILE'

If Sentry is Enabled:

Add these permissions...

`hdfs://user/unravel/HOOK_RESULT_DIR`	*	read+write	data transfer from Hive jobs when Unravel is not up
`hdfs://user/spark/applicationHistory`	hdfs	read	Spark event log
`hdfs://usr/history/done`	hdfs	read	MapReduce logs
`hdfs://tmp/logs`	hdfs	read	YARN aggregation folder
`hdfs://user/hive/warehouse`	hdfs	read	Obtain table partition sizes
Hive Metastore GRANT	hive	read	Hive table information

Do Host-Specific Configuration Steps

For CDH, there are no host-specific configuration steps.

Restart Unravel Server

After edits to `com.unraveldata.login.admins` in `/usr/local/unravel/etc/unravel.properties` it is necessary to run the following script in order to make changes take effect. The `echo` command shows the page to visit with your browser. If you are using an ssh tunnel or http proxy, you might need to make adjustments.

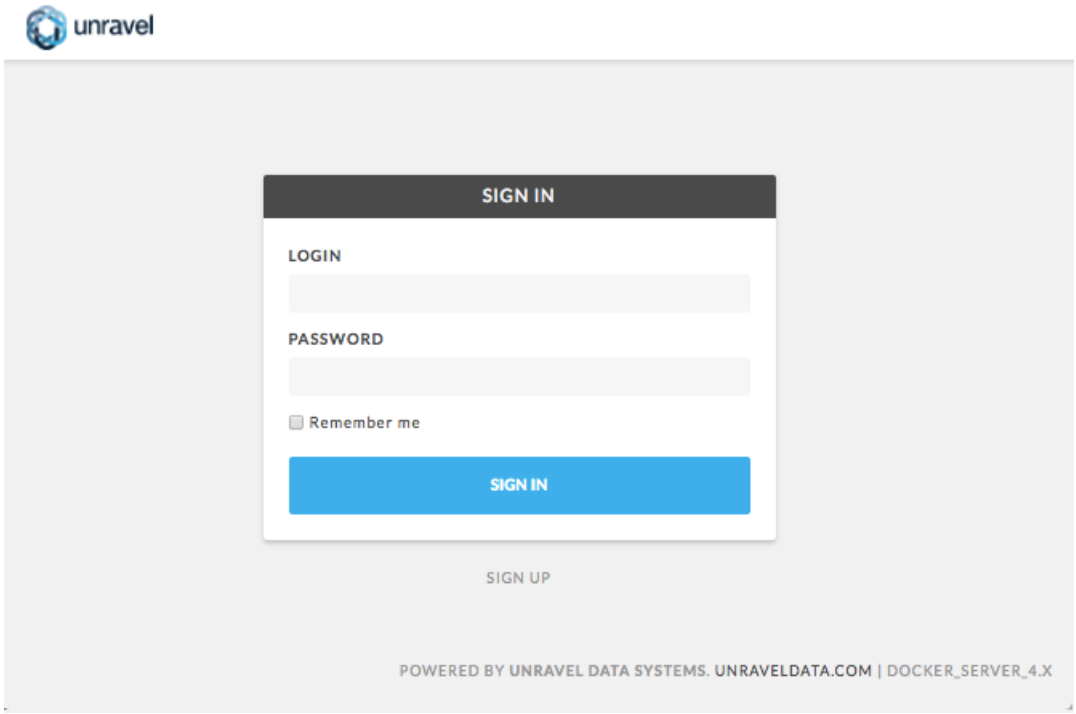
```
sudo /etc/init.d/unravel_all.sh start
sleep 60
echo "http://$(hostname -f):3000/"
```

This completes the basic/core configuration.

4. Log into Unravel Web UI

Using a web browser, navigate to `http://($hostname -f):3000` and login as user "`admin`" with password "`unraveldata`".

For the free trial version, use the Chrome web browser.



Unravel Web UI Login Screen

Congratulations!

Unravel Server is up and running. Unravel Web UI displays collected data. For instructions on using Unravel Web UI, see the User Guide.

5. (Optional) Configure Unravel Server (Advanced Options)

Enable Additional Data Collection/Instrumentation

Install the Unravel Sensor Parcel on gateway/edge/client nodes that are used to submit Hive queries to push additional information to Unravel Server. For details, see Part 2: Install Unravel Sensor Parcel on CDH+CM.

Run the Unravel Web UI Configuration Wizard

Run the Unravel Web UI configuration wizard to choose additional configuration options. For instructions on configuring advanced options, see the User Guide.

Attachments:

worddav95f1e8fd13958fae391afc8e6f9ad03d.png (image/png)

image2017-2-26_0-20-12.png (image/png)

logo_small.png (image/png)

Part 2: Install Unravel Sensor Parcel on CDH+CM

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1

3. Installation Guide for Cloudera Distribution Including Apache Hadoop (CDH), with Cloudera Manager (CM)

Created by Ada-C, last modified on Sep 29, 2017

Introduction

This topic explains how to install the Unravel Sensor 4.0 parcel on CDH clusters using Cloudera Manager (CM). The parcel includes Hive Hook and Spark instrumentation JARs. Hive Hook is used to collect information about Hive queries in Hadoop. The Spark instrumentation JARs measure Spark job performance.

These instructions apply to Unravel Sensor 4.0. For older versions of Unravel Server, contact Unravel Support.

Before following these instructions, follow the steps in Part 1: Install Unravel Server on CDH+CM.

Workflow Summary

1. Obtain and distribute the parcel from Unravel Server.
2. Put the Hive Hook JAR in AUX_CLASSPATH.
3. Configure the gateway automatic deployment of Hive instrumentation.
4. Configure the gateway automatic deployment of Spark instrumentation.

Highlighted text below indicates where you must substitute your particular values.

When Active Directory Kerberos is used, `UNRAVEL_HOST_IP` must be a fully qualified path.

In this section... **Table of Contents**

1. Obtain and Distribute the Parcel from Unravel Server

1. In Cloudera Manager, go to the **Parcels** page by clicking the parcels glyph () on the top of the page.
2. Click **Configuration** to see the **Parcel Settings** pop-up.
3. In the **Remote Parcel Repository URLs** section, click the + glyph to add a new entry.
4. Add `http://UNRAVEL_HOST_IP:3000/parcels/cdhX.Y/` (include trailing slash) where `X.Y` is the major, minor version of CDH version you are running and `UNRAVEL_HOST_IP` is the host name or LAN IP address of Unravel Server where `unravel_lr` is running. If you are running more than one version of CDH (multiple clusters) in Cloudera Manager, you can add more than one parcel directory from the `UNRAVEL_HOST_IP` .
5. Click **Save**.
6. Click **Check for New Parcels**.
7. On the **Parcels** page, pick a target cluster in the **Location** box.
8. In the list of **Parcel Names**, find the `UNRAVEL_SENSOR` parcel that matches the version of the target cluster and click **Download**.
9. Click **Distribute**.
10. If you have an old parcel from Unravel, you can deactivate it now.

11. Then click **Activate** on the new parcel.

2. Put the Hive Hook JAR in AUX_CLASSPATH

1. In Cloudera Manager, for the target cluster, click **Hive** | **Configuration**, and search for "hive-env" (without quotes).
2. In **Gateway Client Environment Advanced Configuration Snippet (Safety Valve)** for `hive-env.sh` enter the following:

```
AUX_CLASSPATH=${AUX_CLASSPATH}:/opt/cloudera/parcels/UNRAVEL_SENSOR/lib/
java/unravel_hive_hook.jar
```

1. In Cloudera Manager, click **YARN** | **Configuration**, and search for "hadoop-env" (without quotes).
2. In **Gateway Client Environment Advanced Configuration Snippet (Safety Valve)** for `hadoop-env.sh`, enter the following:

```
HADOOP_CLASSPATH=${HADOOP_CLASSPATH}:/opt/cloudera/parcels/UNRAVEL_SENSOR/
lib/java/unravel_hive_hook.jar
```

If Sentry is Enabled:

Sentry commands may also be needed to enable access to the Hive Hook JAR file. Grant privileges on the JAR files to the roles that run hive queries. Log into Beeline as user `hive` and use the Hive `SQL GRANT` statement to do so. For example (substitute `$ROLE` as appropriate):

```
GRANT ALL ON URI 'file:///opt/cloudera/parcels/UNRAVEL_SENSOR/lib/java/
unravel_hive_hook.jar' TO ROLE $ROLE
```

3. Configure the Gateway Automatic Deployment of Hive Instrumentation

Use Cloudera Manager to deploy the `hive-site.xml` snippet, which is the content of `/usr/local/unravel/hive-hook/hive-site.xml.snip` on Unravel Server.

Note: On a multi-host Unravel Server deployment, use host2's `/usr/local/unravel/hive-hook/hive-site.xml.snip`

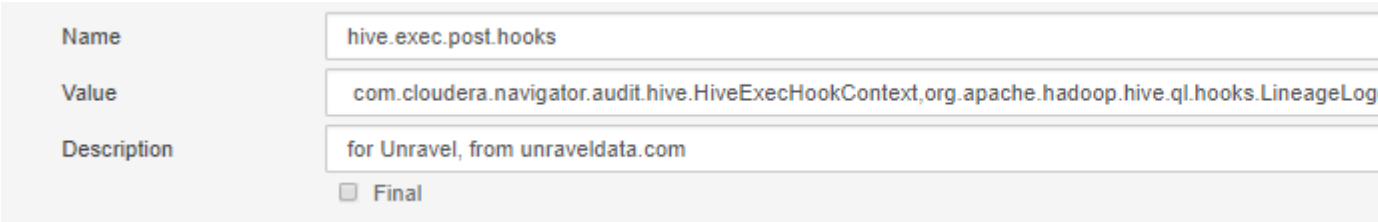
Set the hive-site.xml Snippet in Cloudera Manager, and Deploy the Hive Client Configuration to Gateways

In Cloudera Manager (CM):

1. Go to Hive service.
2. Select the **Configuration** tab.
3. Search for `hive-site.xml` in the middle of the page.
4. Add the xml snippet to **Hive Client Advanced Configuration Snippet for hive-site.xml** (Gateway Default Group) (Click **View as XML**).

   If cluster has been configured with "Cloudera Navigator"; the *hive.exec.post.hooks* property should has exsiting value(s). Therefore append the unravel's value into the existing *hive.exec.post.hooks* property with a comma and no space. (see example below)

   com.cloudera.navigator.audit.hive.HiveExecHookContext,org.apache.hadoop.hive.ql.hooks.LineageLogger, **com.unraveldata.dataflow.hive.hook.HivePostHook**



5. Add the xml snippet to **HiveServer2 Advanced Configuration Snippet for hive-site.xml**. (Click **View as XML**). Like above step, if properties exists, append unravel's value.
6. Save the changes with optional comment "Unravel snippet in hive-site.xml".
7. Perform action **Deploy Hive Client Configuration** by clicking the deploy glyph () or by using the **Actions** pull-down menu.
8. Restart the Hive and YARN services (click **Restart Stale Services** if that is visible.)

Again, monitor the situation to see if all Hive queries are failing with a class not found or permission problems. **If they are failing**, you can back-out the `hive-site.xml` advanced snippet changes in Cloudera Manager, deploy client configuration, and restart the Hive service to revert, then refer to Troubleshooting below.

Check Unravel Web UI

If queries are running fine and appearing in Unravel Web UI, then you are done.

4. Configure the Gateway Automatic Deployment of Spark Instrumentation

1. In Cloudera Manager, select the target cluster, then select the Spark service.
2. Select **Configuration**.
3. Search for "spark-defaults".
4. In the **Spark Client Advanced Configuration Snippet (Safety Valve)** box for **spark-conf/spark-defaults.conf**, enter the following text, substituting the highlighted text for your particular values:

   On a multi-host Unravel Server deployment, use logical host2 for `UNRAVEL_HOST_IP` .

   Copy the text below and paste it into the Cloudera Manager's **Spark Client Advanced Configuration Snippet (Safety Valve)** box for **spark-conf/spark-defaults.conf**. Then, modify the value of `UNRAVEL_HOST_IP` . Do not paste this text into the machine's Linux shell.

```
spark.unravel.server.hostport=UNRAVEL_HOST_IP:4043

spark.driver.extraJavaOptions=-javaagent:/opt/cloudera/parcels/
UNRAVEL_SENSOR/lib/java/btrace-agent.jar=bootClassPath=/
opt/cloudera/parcels/UNRAVEL_SENSOR/lib/java/unravel-
boot.jar,systemClassPath=/opt/cloudera/parcels/
UNRAVEL_SENSOR/lib/java/unravel-sys.jar,scriptOutputFile=/
dev/null,script=DriverProbe.class:SQLProbe.class -
Dcom.sun.btrace.FileClient.flush=-1 -
Dcom.unraveldata.spark.sensor.disableLiveUpdates=true

spark.executor.extraJavaOptions=-javaagent:/opt/cloudera/parcels/
UNRAVEL_SENSOR/lib/java/btrace-agent.jar=bootClassPath=/opt/cloudera/
parcels/UNRAVEL_SENSOR/lib/java/unravel-boot.jar,systemClassPath=/
opt/cloudera/parcels/UNRAVEL_SENSOR/lib/java/unravel-
sys.jar,scriptOutputFile=/dev/null,script=ExecutorProbe.class -
Dcom.sun.btrace.FileClient.flush=-1

spark.eventLog.enabled=true
```

   Notice that in this code block, the blank lines separate only one full line of text that is wrapped due to length.

5. Save changes.

6. Deploy client configuration by clicking the deploy glyph () or by using the **Actions** pull-down menu.

Monitor the situation to see if all Spark queries are failing with a class not found or permission problems. **If they are failing**, you can back-out the `spark-defaults.conf` changes in Cloudera Manager, re-deploy client configuration, and then investigate and fix the issue.

5. Optional - Configure YARN - MapReduce (MR) JVM Sensor Cluster-Wide

Set in Cloudera Manager

In Cloudera Manager (CM):

1. Go to **YARN** service.
2. Select the **Configuration** tab.
3. Search for **ApplicationMaster Java Opts Base** and c oncatenate the following xml block properties snippet (ensure to start with a space and add below **).**

```
 -javaagent:/opt/cloudera/parcels/UNRAVEL_SENSOR/lib/java/
btrace-agent.jar=systemClassPath=/opt/cloudera/parcels/
UNRAVEL_SENSOR/lib/java/unravel-mr-sys.jar,bootClassPath=/
opt/cloudera/parcels/UNRAVEL_SENSOR/lib/java/unravel-mr-
boot.jar,scriptOutputFile=/dev/null  -Dcom.sun.btrace.FileClient.flush=-1
 -Dunravel.server.hostport=UNRAVEL_HOST_IP:4043
```

   Notice that in this code block, ensure "-" is a minus sign and customer need to modify the value of `UNRAVEL_HOST_IP` with their own Unravel server IP address. For multi-host Unravel installation, use the IP address of Host2.
4. Search for **MapReduce Client Advanced Configuration Snippet (Safety Valve) for mapred-site.xml** in the middle of the page.
5. Enter following xml four block properties snippet to **Gateway Default Group** (Click **View as XML**).

```
<property><name>mapreduce.task.profile</name><value>true</value></property>
<property><name>mapreduce.task.profile.maps</name><value>0-5</value></
property>
<property><name>mapreduce.task.profile.reduces</name><value>0-5</value></
property>
<property><name>mapreduce.task.profile.params</name><value>-javaagent:/opt/
cloudera/parcels/UNRAVEL_SENSOR/lib/java/btrace-agent.jar=systemClassPath=/
opt/cloudera/parcels/UNRAVEL_SENSOR/lib/java/unravel-mr-
sys.jar,bootClassPath=/opt/cloudera/parcels/UNRAVEL_SENSOR/lib/java/unravel-
mr-boot.jar,scriptOutputFile=/dev/null  -Dcom.sun.btrace.FileClient.flush=-1
 -Dunravel.server.hostport=UNRAVEL_HOST_IP:4043</value></property>
```

Notice that in this code block, ensure "-" is a minus sign and customer need to modify the value of `UNRAVEL_HOST_IP` with their own Unravel server IP address. For multi-host Unravel installation, use the IP address of Host2.

6. Save changes.

7. Deploy client configuration by clicking the deploy glyph ( i) or by using the **Actions** pull-down menu.

8. Restart the YARN services (click **Restart Stale Services** if that is visible. However, you can also perform this later when you have a planned maintenance)

Monitor the situation and you should see in Unravel UI a Resource Usage tab showing you mappers and reducers when you view the Application page for any completed MR job. Restart is important for MR sensor to be picked up by queries submitted via Hiveserver2.

Troubleshooting

Symptom	Problem	Remedy

`hadoop fs -ls /user/unravel/HOOK_RESULT_DIR/` shows directory does not exist	• Unravel Server RPM is not yet installed, or • Unravel Server RPM is installed on a different HDFS cluster, or • HDFS home directory for Unravel does not exist, or • kerberos/sentry actions are needed	Install Unravel RPM on Unravel service host: `sudo rpm -U unravel*.rpm*` OR Verify that `unravel` user exists and has a `/user/unravel/` directory in HDFS and unravel has write access.
`ClassNotFound` error for `com.unraveldata.dataflow.hive.hook` during Hive query execution	Unravel hive hook JAR was not found in `$HIVE_HOME/lib/`.	Check whether UNRAVEL_SENSOR parcel was distributed and activated in CM. OR Put the Unravel hive-hook JAR corresponding to `$HIVE_VER` in `JAR_DEST` on each gateway by: `cd /usr/local/unravel/hive-hook/; cp unravel-hive-HIVE_VER*hook.jar JAR_DEST`
Permission denied writing to `hdfs://user/unravel/HOOK_RESULT_DIR` during Hive query execution	`hdfs://user/unravel/HOOK_RESULT_DIR/*` directories are not writable	`hadoop fs -chmod 777 /user/unravel/HOOK_RESULT_DIR/*` OR Sentry command is needed to give permission OR revert to your previous `hive-site.xml` without Unravel Hive Hook clauses.

References

{+} http://www.cloudera.com/documentation/enterprise/5-3-x/topics/cm_mc_hive_udf.html#concept_nc3_mms_lr_unique_2+ see Creating Permanent Functions.

Attachments:

- worddav95f1e8fd13958fae391afc8e6f9ad03d.png (image/png)
- worddavc3ab5630500b15d73b161435cac14ebf.png (image/png)
- worddavaa4942fe2f3185b67e3c11198ae02b0c.png (image/png)
- logo_small.png (image/png)
- hive_exec_post_hook.PNG (image/png)

Part 3: Additional Topics for CDH

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Installation Guide for Cloudera Distribution Including Apache Hadoop (CDH), with Cloudera Manager (CM)

- Creating Active Directory Kerberos Principals and Keytabs for Unravel
- Sensors
  - Installing Unravel Sensor for Individual Applications Submitted Through spark-shell
  - Installing Unravel Sensor for Individual Applications Submitted Through spark-submit

Installation Guide for MapR

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1

This guide is compatible with

- MapR 5.1

Ordered Steps

- Page: Part 1: Install Unravel Server on MapR
- Page: Part 2: Enable Additional Data Collection / Instrumentation for MapR
- Page: Part 3: Additional Topics for MapR

Part 1: Install Unravel Server on MapR

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Installation Guide for MapR

Introduction

This topic explains how to deploy Unravel Server 4.0 on the MapR converged data platform. These instructions apply to Unravel Server 4.0. For older versions of Unravel Server, contact Unravel Support.

Workflow Summary

1. Pre-installation check
2. Configure the host machine.

3. Install the Unravel Server RPM on the host machine.
4. Configure Unravel Server (basic/core options).
5. Log into Unravel Web UI.
6. (Optional) Configure Unravel Server (advanced options).

Table of Contents

Pre-Installation Check

The following installation requirements must be met for successful installation of Unravel.

Platform Compatibility

- MapR 5.1
- Hadoop 1.x - 2.x
- Kerberos
- Hive 0.9.x - 1.2.x
- Spark 1.3.x - 2.0.x

Hardware

- Architecture: x86_64
- Cores: 8
- RAM: 64GB minimum
- Disk: `/usr/local/unravel` with 2.5GB free minimum
- Disk: `/srv/unravel` with 500GB free minimum

- For 10,000+ MR jobs per day, two or more gateway/edge nodes are recommended

Software

- Operating System: RedHat/Centos 6.4 - 7.3
- `libaio.x86_64` installed
- `SELINUX=permissive` (or disabled) should be set in `/etc/selinux/config`
- HDFS+Hive+YARN client/gateway, Hadoop and Hive commands in `PATH`
- If Spark is in use, Spark client gateway
- LDAP (AD or Open LDAP) compatible for Unravel Web UI user authentication (Open signup by default)
- On Unravel Edge-node server, please **do not** have zookeeper installed in same server

Access Permissions

- If Kerberos is in use, a keytab for principal hdfs (or read-only equivalent) is required for access to:
  - Access to YARN's "done dir" in HDFS
  - Access to YARN's log aggregation directory in HDFS
  - Access to Spark event log directory in HDFS
  - Access to file sizes under HIve warehouse directory
- Access to YARN Resource Manager REST API
  - principal needs right to find out which RM is active
- JDBC access to the Hive Metastore (read-only user is sufficient)

Network

- Port 3000 (or 4020) from users and entire cluster to Unravel Web UI
- HDFS ports open from Hadoop cluster to Unravel Server(s)
- For MR1, TaskTracker port open from Hadoop cluster to Unravel Server(s)
- For MR1/YARN, Hive Metadata DB port open to Unravel Server(s) for partition reporting
- UDP and TCP port 4043 open from entire cluster to Unravel Server(s)
- For Oozie, port 11000 open from Unravel Server(s) to the Oozie server
- Resource Manager (RM) port 8032 from Unravel Server(s) to the RM server(s)
- Port 4176, 4181 through 4189, 3316, 4091 must be available for localhost communication between Unravel daemons or services

1. Configure the Host Machine

Allocate a Cluster Gateway/Edge/Client Host with HDFS Access

Install `mapr-client` to enable the `hadoop fs` command.

Configure the Host *Before* installing the RPM

- Run the following commands on Unravel Server as `root`:

```
sudo useradd -g mapr unravel
hadoop fs -mkdir /user/unravel
hadoop fs -chown unravel:mapr /user/unravel
```

- If MapR tickets are enabled, check mapr ticket for users `unravel` and `mapr` on target host now. You might need to export ticket environment variables in `/etc/unravel_ctl` first.
- Check available RAM to ensure availability:

```
free -g
```

- For instructions on adjusting RAM allocated to MapR-FS (mfs), see https://community.mapr.com/docs/DOC-1209. For example, edit `/opt/mapr/conf/warden.conf` as follows:

  ```
  service.command.mfs.heapsize.maxpercent=10
  ```

  (only change this setting on the Unravel gateway/client machine). And the restart mfs.

2. Install the Unravel Server RPM on the Host Machine

Get the Unravel Server RPM

Copy the RPM from the Unravel distribution server to the host machine using the username and password given to you by Unravel Support:

- For the free trial version:

  ```
  scp "unraveltrial@trial.unraveldata.com:unravel-4.0-*.x86_64.rpm" .
  ```

- For the enterprise version:

  ```
  scp $USER@dist.unraveldata.com:unravel-4.0-*.x86_64.rpm.$timestamp .
  ```

The precise RPM filename will vary. The version has the structure *x.y.b* where *b* is a build number that is imposed as an RPM epoch which means it takes precedence over version numbers for determining what is more up-to-date. The Unravel build numbers always increase and are allocated so that they respect the ordering of *x.y* versioning.

Install the Unravel Server RPM

Replace the asterisks as needed to be more selective.

```
sudo rpm -U unravel-4.0*.x86_64.rpm*
```

The installation creates `/usr/local/unravel/` which contains the executables, scripts, and settings. User `unravel` is created. The initial internal database and other durable state are put in `/srv/unravel/` for larger storage. By default, the installation supports YARN. The master configuration file is in `/usr/local/unravel/etc/unravel.properties` and the logs are in `/usr/local/unravel/logs/`. The RPM installation creates user `unravel` if it does not already exist; `/etc/init.d/unravel_*` scripts for controlling its services as well as `/etc/init.d/unravel_all.sh` which can be used to manually stop, start, and get status of all daemons in proper order. The RPM installation also creates an HDFS directory for Hive Hook information collection. During initial install, a bundled database is used. This can be switched to use an externally managed MySQL for production. (The bundled database root mysql password will be stored in `/root/unravel.install.include` during installation.)

Do Host-Specific Post-Installation Actions

Run the following commands on Unravel Server:

```
sudo /etc/init.d/unravel_all.sh stop
sudo /usr/local/unravel/install_bin/switch_to_mapr.sh
```

3. Configure Unravel Server (Basic/Core Options)

Enable/Disable Optional Daemons

Depending on your workload volume or kind of activity, you can enable or disable optional daemons at this point.

- If you are not using Oozie, disable `unravel_os3`:

  ```
  sudo chkconfig unravel_os3 off
  ```

- If you are not using Spark, disable `unravel_sw`:

```
sudo chkconfig unravel_sw off
```

- If you have 10000-20000 jobs per day, enable these workers:

```
sudo chkconfig --add unravel_ew_2
sudo chkconfig --add unravel_hhw_2
sudo chkconfig --add unravel_jcw2_2
```

- If you have 20000-30000 jobs per day, enable these workers:

```
sudo chkconfig --add unravel_ew_3
sudo chkconfig --add unravel_hhw_3
sudo chkconfig --add unravel_jcw2_3
```

- If you have more than 30000 jobs per day, enable these workers:

```
sudo chkconfig --add unravel_ew_4
sudo chkconfig --add unravel_hhw_4
sudo chkconfig --add unravel_jcw2_4
```

Modify `unravel.properties`

- Open `/usr/local/unravel/etc/unravel.properties` with `vi`.

```
sudo vi /usr/local/unravel/etc/unravel.properties
```

- Edit the values in `unravel.properties` using the guidelines and descriptions in the table below. Some of these need to be uncommented (remove the leading # char) after you edit to enable them.

`com.unraveldata.advertised.url`	Defines the Unravel Server URL for HTTP traffic.	`com.unraveldata.advertised.url=http LAN_DNS:3000`
`com.unraveldata.customer.organization`	Identifies your installation for reporting purposes.	`com.unraveldata.customer.organizati`
`com.unraveldata.tmpdir`	Location where Unravel's temp file will reside	`com.unraveldata.tmpdir=/ srv/unravel/tmp`
`com.unraveldata.history.maxSize.week`	Sets retention for search data.	`com.unraveldata.history.maxSize.wee`
`com.unraveldata.hive.hook.topic.num`	**Optional** Defines the number of threads. Default is 1. Depending on job volume, increase this property to *N* where *N* is between 1 and 4, or roughly *ThousandJobsPerDay*/10.	`com.unraveldata.hive.hook.topic.nu`
`com.unraveldata.job.collector.done`	HDFS path to "done" directory of MR logs	`com.unraveldata.job.collector.done var/mapr/cluster/yarn/ rm/staging/history/done`
`com.unraveldata.job.collector.log.`	An HDFS path that helps locate MR job logs to process	`com.unraveldata.job.collector.log. tmp/logs/*/logs/`
`com.unraveldata.login.admins`	Defines the usernames that can access Unravel Web UI's admin pages. Default is `admin`.	`com.unraveldata.login.admins=admin`
`com.unraveldata.spark.eventlog.loca`	Where to find Spark event logs	`com.unraveldata.spark.eventlog.loca apps/spark`
`yarn.resourcemanager.webapp.address`	Resource Manager web app address	`yarn.resourcemanager.webapp.addres example.localdomain:8088`
`oozie.server.url`	Oozie URL	`oozie.server.url=http:// example.localdomain:11000/ oozie`

If Kerberos is Enabled:

Add authentication for HDFS...

(a) Create a keytab for unravel for daemons that run as `unravel` and put the file in `/usr/local/unravel/etc/unravel.keytab` (for example).

(b) Create a keytab for hdfs for the Unravel daemons that run as user `hdfs` and put the file in `/etc/keytabs/hdfs.keytab` (for example).

(c) Tell Unravel Server about it (env var for hdfs keytab location; substitute correct hostname):

```
echo "export HDFS_KEYTAB_PATH=/etc/keytabs/hdfs.keytab
export HDFS_KERBEROS_PRINCIPAL=unravel/myhost.mydomain@MYREALM
" | sudo tee -a /usr/local/unravel/etc/unravel.ext.sh
```

(d) Add properties for Kerberos:

```
echo "
com.unraveldata.kerberos.principal=unravel/myhost.mydomain@MYREALM
com.unraveldata.kerberos.keytab.path=/usr/local/unravel/etc/unravel.keytab
" | sudo tee -a /usr/local/unravel/unravel.properties
```

If Sentry is Enabled:

- Add these permissions...

`hdfs://user/unravel/HOOK_RESULT_DIR`	*	read+write

Do Host-Specific Configuration Steps

For MapR, there are no host-specific configuration steps.

Restart Unravel Server

After edits to `com.unraveldata.login.admins` in `/usr/local/unravel/etc/unravel.properties` it is necessary to run the following script in order to make changes take effect. The `echo` command shows the page to visit with your browser. If you are using an ssh tunnel or http proxy, you might need to make adjustments.

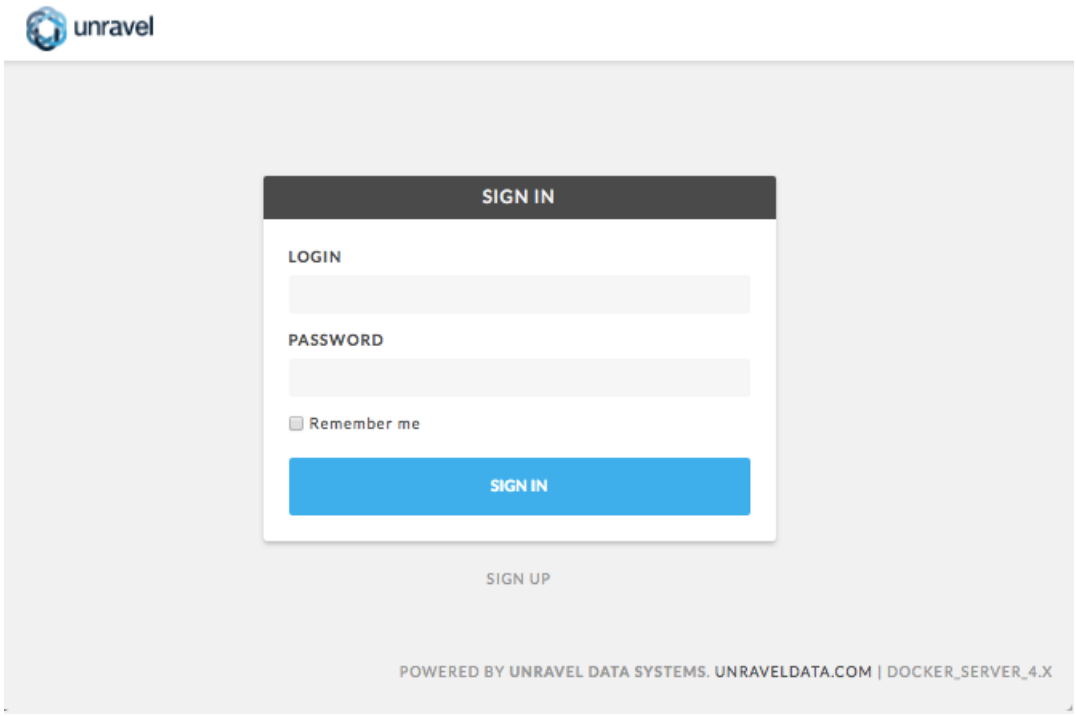
```
sudo /etc/init.d/unravel_all.sh start
sleep 60
 echo "http://$(hostname -f):3000/"
```

This completes the basic/core configuration.

4. Log into Unravel Web UI

Using a web browser, navigate to `http://($hostname -f):3000` and login as user "`admin`" with password "`unraveldata`".

For the free trial version, use the Chrome web browser.



Unravel Web UI Login Screen

Congratulations!

Unravel Server is up and running. Unravel Web UI displays collected data. Check Unravel Web UI for MR jobs loading: on the **Applications** page, select the **Map Reduce** tab.

For instructions on using Unravel Web UI, see the User Guide.

5. (Optional) Configure Unravel Server (Advanced Options)

Enable Additional Data Collection/Instrumentation

Install the Unravel Sensor Parcel on gateway/edge/client nodes that are used to submit Hive queries to push additional information to Unravel Server. For details, see Part 2: Enable Additional Data Collection / Instrumentation for MapR.

Run the Unravel Web UI Configuration Wizard

Run the Unravel Web UI configuration wizard to choose additional configuration options. For instructions on configuring advanced options, see the Unravel Web UI User Guide.

Attachments:

- worddav95f1e8fd13958fae391afc8e6f9ad03d.png (image/png)
- image2017-2-26_0-20-12.png (image/png)
- logo_small.png (image/png)

Part 2: Enable Additional Data Collection / Instrumentation for MapR

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Installation Guide for MapR

Created by Ada-C, last modified on Sep 29, 2017

Introduction

This topic explains how to enable additional data collection or instrumentation on the MapR converged data platform. These instructions apply to Unravel Server v4.0. For older versions of Unravel Server, contact Unravel Support.

Workflow Summary

1. Enable additional instrumentation on Unravel Server's host.
2. Enter correct value for Hive Metastore, Resource Manager and Oozie properties.
3. Confirm that Unravel Web UI shows additional data.
4. Confirm and adjust the settings in `yarn-site.xml`.
5. Enable additional instrumentation on other hosts in the cluster.

1. Enable Additional Instrumentation on Unravel Server's Host

Best practice is to enable instrumentation on Unravel Server itself for testing, practice, and demonstration purposes, before enabling instrumentation on other gateway/edge/client machines that do real client work (Hive queries, Map-Reduce jobs, Spark, and so on).

Run the shell script `unravel_mapr_setup.sh` on Unravel Server (on **host1** for multi-host Unravel installs). This script installs Unravel Sensor (Hive Hook) on Unravel Server's host:

```
sudo yum install -y wget
cd /usr/local/unravel/install_bin/unraveldata-clients
sudo ./unravel_mapr_setup.sh install -y --unravel-server $UNRAVEL_HOST:3000
 --spark-version SPARK_VERSION --hive-version HIVE_VERSION
```

Note: In the code above, substitute valid values for:

- *UNRAVEL_HOST_IP - fully qualified host name or IP address*
- *SPARK_VERSION - target Spark version*
- *HIVE_VERSION - target Hive version*

2. Enter correct value for Hive Metastore, Resource Manager, and Oozie properties

Use `vi` to edit **/usr/local/unravel/etc/unravel.properties**: un-comment relevant respective Hive Metastore, Resource Manager, and Oozie properties:

```
#     hive metastore
#
### Uncomment and enter correct HM java.jdo.option.Connection properties ###
#javax.jdo.option.ConnectionURL=jdbc:mysql://example.localdomain:3306/hive
#javax.jdo.option.ConnectionURL=jdbc:postgresql://example.localdomain:7432/
hive_zzzzzz
#javax.jdo.option.ConnectionDriverName=com.mysql.jdbc.Driver
#javax.jdo.option.ConnectionDriverName=org.postgresql.Driver
#javax.jdo.option.ConnectionUserName=hiveuser?
#javax.jdo.option.ConnectionPassword=???????

#     optional selectivity of databases to analyze in metastore
#com.unraveldata.metastore.databasePattern=s*|t*|d*

#     Resource Manager (RM)
#
#     Enable https access to Resource Manager
#https.protocols=TLSv1.2
#
### Uncomment and enter correct below properties RM ###
#yarn.resourcemanager.webapp.address=http://example.localdomain:8088

#     Resource Manager username to login
#yarn.resourcemanager.webapp.username=foo

#     Resource Manager password to login
```

```
#yarn.resourcemanager.webapp.password=?????

#
#     oozie
#
### Uncomment below oozie properties when oozie is used ###
# oozie.server.url=http://<oozie-hostname-IP-address>:11000/oozie
```

Then restart Unravel Server:

```
sudo /etc/init.d/unravel_all.sh restart
```

3. Confirm that Unravel Web UI Shows Additional Data

Run a Hive job using a test script provided by Unravel Server:

 This is where you can see the effects of the instrumentation setup. Best practice is to run this test script on Unravel Server rather than on a gateway/edge/client node. That way you can verify that instrumentation is working first, and then enable instrumentation on other gateway/edge/client nodes.

```
sudo -u $someUser /usr/local/unravel/install_bin/hive_test_simple.sh
```

This script creates a uniquely named table in the default database, adds some data, runs a Hive query on it, and then deletes the table.

Note: Replace $someUser with a user that can create tables in the default database. If you need to use a different database, copy the script and edit it to change the target database.

This script runs the query twice using different workflow tags so you can clearly see the two different runs of the same workflow in Unravel Web UI.

4. Confirm and Adjust the Settings in `yarn-site.xml`

Check specific properties only Unravel srv in `/opt/mapr/hadoop/hadoop-2.7.0/etc/hadoop/yarn-site.xml` to be sure that these settings are present (with your particular values for your Resource Manager web app address):

- `yarn.resourcemanager.webapp.address`:

```
<property>
<name>yarn.resourcemanager.webapp.address</name>
<value>10.0.0.110:8088</value>
<source>yarn-site.xml</source>
</property>
```

- `yarn.log-aggregation-enable`:

```
<property>
<name>yarn.log-aggregation-enable</name>
<value>true</value>
<description>For log aggregations</description>
</property>
```

5. Enable Additional Instrumentation on Other Hosts in the Cluster

To instrument more servers, you can use the setup script we provide or see the effect it has and replicate it using your own provisioning automation system. If you already have a way to customize and deploy hive-site.xml, yarn-site.xml and user defined function jars, you can add the changes and jar from Unravel to your existing mechanism.

- Run the shell script `unravel_mapr_setup.sh` on each node of the cluster, just like it was run on the Unravel server above.
- Copy the newly edited (in the previous step 4) yarn-site.xml to all nodes.

- Do a rolling-restart of HiveServer2

Attachments:

logo_small.png (image/png)

image2017-2-26_0-20-12.png (image/png)

Part 3: AdditionalTopics for MapR

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Installation Guide for MapR

- Creating Active Directory Kerberos Principals and Keytabs for Unravel
- Sensors
  - Installing Unravel Sensor for Individual Applications Submitted Through spark-shell
  - Installing Unravel Sensor for Individual Applications Submitted Through spark-submit
  - Installing Unravel Sensor for Individual Hive Queries
- Workflows
  - Monitoring Airflow Workflows
  - Monitoring Oozie Workflows
  - Tagging Workflows
- Custom Configurations
  - Adding More Admins to Unravel Web UI
  - Configuring Multiple Hosts for Unravel Server
  - Creating Multiple Workers for High Volume Data
  - Defining a Custom TC Port
  - Integrating LDAP Authentication for Unravel Web UI
  - Setting Retention Time in Unravel Server
  - Setting Up Email for Auto Actions and Collaboration
- Connecting to a Hive Metastore
- Creating an AWS RDS CloudWatch Alarm for FreeStorageSpace
- Creating Application Tags
- Troubleshooting
  - Running Verification Scripts and Benchmarks
  - Sending Diagnostics to Unravel Support
- Uninstalling Unravel Server
- Upgrading Unravel Server

Installation Guide for Amazon Elastic MapReduce (Amazon EMR) and Qubole Clusters

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1

This guide is compatible with

- Amazon EMR 3.4 - 4.7

Ordered Steps

- Page: Part 1: Install Unravel Server for Amazon EMR or Qubole Cluster
- Page: Part 2: Install Unravel Hive Sensor on Qubole Hadoop2/Hive Cluster
- Page: Part 3: Install Unravel Spark Sensor on New or Existing Qubole Spark Cluster
- Page: Part 4: Modify Amazon EMR Cluster Bootstrap/Setup
- Page: Part 5: Additional Topics for EMR or Qubole

Part 1: Install Unravel Server for Amazon EMR or Qubole Cluster

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Installation Guide for Amazon Elastic MapReduce (Amazon EMR) and Qubole Clusters

Table of Contents

Introduction

This topic explains how to deploy Unravel Server 4.0 on Amazon EMR or Qubole. For instructions that correspond to older versions of Unravel Server, contact Unravel Support.

Steps marked [HVD] are for production high volume and durable installations can be skipped for PoCs.

Workflow Summary

1. Configure the cluster.
2. Install the Unravel Server RPM on the cluster.
3. Configure Unravel Server (basic/core options).
4. Log into Unravel Web UI.
5. (Optional) Enable additional data collection/instrumentation.

Requirements Checklist

Platform Compatibility

- Amazon EMR 3.4 - 4.7
- Hadoop 1.x - 2.x
- Kerberos
- Hive 0.9.x - 1.2.x
- Spark 1.3.x - 1.6.x

Software

- Operating System: RedHat/Centos 6.4 - 7.3
- libaio.x86_64 installed
- `SELINUX=permissive` should be set in `/etc/selinux/config`
- HDFS+Hive+YARN client/gateway, hadoop and hive commands in `PATH`
- If Spark is in use, Spark client gateway
- Open signup or LDAP for Unravel Web UI user authentication

Hardware

- Architecture: x86_64
- Cores: 8
- RAM: 64GB minimum
- Disk: `/` with 2.5GB free minimum
- Disk: `/srv` with 500GB free minimum
- For 10,000+ MR jobs per day, two or more gateway/edge nodes are recommended

Network

- Port 3000 (or 4020) for Unravel Web UI access
- HDFS ports open from Hadoop cluster to Unravel Server(s)
- For MR1, TaskTracker port open from Hadoop cluster to Unravel Server(s)
- For MR1/YARN, Hive Metadata DB port open to Unravel Server(s) for partition reporting
- UDP and TCP ports 4041-4043 open from Hadoop cluster to Unravel Server(s) unless Hive-hook via HDFS option is used
- For Oozie, port 11000 open to Unravel Server(s)

1. Configure the Cluster

Provision an Instance

- Instance Type: r3.2xlarge
- OS: centos7
- Server must be "Optimized for EBS" if EBS durable storage is used
- Set boot partition to 12GB or larger
- You **must** add the SSD ephemeral (instance storage) disk(s) included with the instance
- Must be in same region as the EMR clusters Unravel Server will monitor
- Note the Availability Zone (AZ) of the instance because the EBS partition you create/mount below must be in the same AZ. [HVD]
- Security Group
  - Unravel Server works with multiple EMR clusters, including existing clusters and new clusters. A TCP and UDP connection is needed from the master node of each EMR cluster to Unravel Server.
  - Simplest approach it to make Unravel server a member of the security group ElasticMapReduce-master when the instance running Unravel Server is first launched.
  - If Unravel Server is already started or if you prefer more security groups, open ports 3000 (TCP) and 4041-4043 (TCP and UDP) from the ElasticMapReduce-master group to a new server security group called 'unravel'.
- Start `ntpd` and check whether time is accurate.

Configure the Environment at First Login

- Disable `selinux`:

```
# sudo setenforce Permissive
```

- Edit the file to make sure setting persists after reboot. Make sure SELINUX=permissive

```
# vi /etc/selinux/config
```

- Install `libaio.x86_64`:

```
# sudo yum -y install libaio.x86_64
```

- Install `lzop`:

```
# sudo yum install lzop.x86_64
```

Configure Ephemeral Storage

Find the available ephemeral storage (also called *instance storage*). Use `lsblk` to find unmounted devices. If no ephemeral/instance storage is allocated, terminate the server and re-provision it, being sure to add storage of type "instance storage".

Tip

It can be convenient to use `/srv` as the mount point when only one ephemeral device is present.

- Find block devices with no mount point:

```
# lsblk
```

- For each unmounted area do the following steps, substituting correct value for `{ Z}` , and substituting `{EPHEMERAL}` for the mount point you chose:

```
# sudo mkfs.ext4 /dev/xvd{Z}
```

- If necessary, create mount point and check if mounted,

```
# mkdir $EPHEMERAL
```

```
# echo "/dev/xvd{Z} $EPHEMERAL ext4 defaults,noatime,nodiratime 1 2" |
 sudo tee -a /etc/fstab
# sudo mount -a
# df -h $EPHEMERAL
```

Make a note of the path to the mounted ephemeral store, referred to as `UNRAVEL_EPHEMERAL` below.

Configure Durable Storage [HVD]

In a PoC or test install, this step can be skipped if there is sufficient disk space (at least 100GB) on / or under /srv mounted from an ephemeral ('instance storage') disk area.

Here we create a "Provisioned IOPS" EBS volume, setting the maximum IOPS, based on the size 300GB.

- In AWS EC2 console, **Volumes**, create an EBS Provisioned IOPS SSD volume of 300GB and 3000 IOPS *in the same AZ as the Unravel server* and associate it with the Unravel server.
- On Unravel server, find the letter `{ Z}` of the new un-mounted device with `lsblk`
- Use `dd` to pre-warm the new EBS area (volume `{ Z}`) before mounting it (this can take an hour!):

  ```
  # sudo dd if=/dev/zero of=/dev/xvd{Z} bs=1M
  ```

  Reference:[|http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-prewarm.html] {+} http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-prewarm.html+
- Format the volume as `ext4`, substituting the correct value for `Z` :

  ```
  # sudo mkfs.ext4 /dev/xvd{Z}
  ```

- Mount the new volume (for example, as `/mnt/unravel_durable` and referred to as `UNRAVEL_DURABLE` below) by adding an entry to `/etc/fstab` and then doing `mount -a` while substituting the correct `Z` :

  ```
  # echo "/dev/xvd{Z} $UNRAVEL_DURABLE ext4 defaults,noatime,nodiratime 1 2"
   | sudo tee -a /etc/fstab
  # sudo mount -a
  ```

- Check if the volume is mounted:

  ```
  # df -h $UNRAVEL_DURABLE
  ```

2. Install the Unravel Server RPM on the Cluster

The precise RPM filename will vary. The version has the structure *`x.y.b`* where *`b`* is a build number that is imposed as an RPM epoch which means it takes precedence over version numbers for determining what is more up-to-date. The Unravel build numbers always increase and are allocated so that they respect the ordering of *`x.y`* versioning.

Replace the asterisks below as needed to be more selective.

Get the Unravel Server RPM

Copy the RPM from the Unravel distribution server to the host machine using the username and password given to you by Unravel Support:

- For the free trial version:

  ```
  # scp "unraveltrial@trial.unraveldata.com:unravel-4.0-*.x86_64.EMR.rpm" .
  ```

- For the enterprise version:

  ```
  # scp $USER@dist.unraveldata.com:unravel-4.0-*.x86_64.EMR.rpm.$timestamp .
  ```

Install the Unravel Server RPM

```
# sudo rpm -U unravel-4.0*.x86_64.EMR.rpm*
```

The installation creates `/usr/local/unravel/` which contains the executables, scripts, and settings. User `unravel` is created. The initial internal database and other durable state are put in `/srv/unravel/` for larger storage. By default, the installation supports YARN. The master configuration file is in `/usr/local/unravel/etc/unravel.properties` and the logs are in `/usr/local/unravel/logs/`. The RPM installation creates user `unravel` if it does not already exist; `/etc/init.d/unravel_*` scripts for controlling its services, as well as `/etc/init.d/unravel_all.sh` which can be used to manually stop, start, and get status of all daemons in proper order. During initial install, a bundled database is used. This can be switched to use an externally managed MySQL for production. (The bundled database root mysql password is stored in `/root/unravel.install.include` during installation.)

Grant Access to Unravel Server

- Assign elastic IP to Unravel Server using AWS Console [unless you use VPC]
- Adjust internal DNS for new IP [for access via a browser]
- Request reverse DNS change from AWS [if you plan on adding SSL]

Restriction

Do not make Unravel Server accessible on the public Internet because doing so would violate your licensing terms.

Supply Credentials Needed for EMR and EC2

Create IAM User(s) and Credentials

Open the AWS IAM (Identity and Access Management) console with your browser and then make the following users and credentials so that Unravel Server can access EMR logs stored in S3 and use EMR read-only permission to find EMR clusters for efficient data loading. These credentials are described separately, but can be combined into one user if desired. Multiple accounts can also be created per access kind here if Unravel Server is going to monitor multiple accounts; just create multiple credential files.

S3 Read-Only Access

1. Create a group named `s3ro` with managed policy `AmazonS3ReadOnlyAccess`.
2. Create a user named `s3unravel`, and save the access credentials for configuring Unravel.
3. Add user `s3unravel` to group `s3ro`.

EMR API Read-Only Access

1. Create a group named `emrro` with managed policy `AmazonElasticMapReduceReadOnlyAccess`.
2. Create a user named `emrunravel`, and save the access credentials for configuring Unravel.
3. Add user `emrunravel` to group `emrro`.

Cloudwatch API Read-Only Access

1. Create a group named `cwro` with managed policy `AmazonCloudWatchReadOnlyAccess`.
2. Create a user named `cwunravel`, and save the access credentials for configuring Unravel.
3. Add user `cwunravel` to group `cwro`.

3. Configure Unravel Server (Basic/Core Options)

The Unravel Server's configuration directory is located at `/usr/local/unravel/etc` on Unravel Server. You will need to put your credential files as property files there, modify `unravel.properties`, and restart Unravel Server, as detailed below.

Copy S3 and EMR Credential Files to Unravel Server

```
# scp -i ~/rsa.pem s3ro.properties root@xx.xxx.xx.xxx:/usr/local/unravel/
etc
# scp -i ~/rsa.pem emrro.properties root@xx.xxx.xx.xxx:/usr/local/unravel/
etc
```

```
# scp -i ~/rsa.pem cwro.properties root@xx.xxx.xx.xxx:/usr/local/unravel/etc
```

where:

- `rsa.pem` is your RSA key,
- `root@xx.xxx.xx.xxx` is the LOCAL_IP of the Unravel Server, and
- `s3ro.properties`, `emrro.properties`, and `cwro.properties` are your credential files containing the respective credentials in the following format:

```
[default]
aws_access_key_id = {your access key}
aws_secret_access_key = {your secret key}
```

You can create multiple credentials of the same type for multiple accounts by creating multiple files with the same base name and appending ".1" for the second account, ".2" for the third account, and so on. For example, using the file naming convention suggested above, copy these additional files into `/usr/local/unravel/etc/`:

```
s3ro.properties.1
emrro.properties.1
cwro.properties.1
```

All three files are required for each account.

Open an SSH Session to Unravel Server

Replace `{ somefile.pem}` with the correct filename.

`UNRAVEL_HOST_IP` - must be a fully qualified path.

```
# ssh -i {somefile.pem} root@$UNRAVEL_IP
```

Set Correct Permissions on the Unravel Configuration Directory

```
# cd /usr/local/unravel/etc
# sudo chown unravel:unravel *.properties
# sudo chmod 644 *.properties
```

Modify `unravel.properties`

The settings file `/usr/local/unravel/etc/unravel.properties` is created during initial install and subsequent RPM upgrades will not change it because your site-specific properties are put into this file.

- Open `/usr/local/unravel/etc/unravel.properties` with `vi`.

  ```
  # sudo vi /usr/local/unravel/etc/unravel.properties
  ```

- Edit the following values in `unravel.properties`:

  ```
  com.unraveldata.s3.profile.config.file.path=/usr/local/unravel/etc/
  s3ro.properties
  com.unraveldata.emr.profile.config.file.path=/usr/local/unravel/etc/
  emrro.properties
  com.unraveldata.cloudwatch.profile.config.file.path=/usr/local/unravel/
  etc/cwro.properties
  ```

- Adjust other values in `unravel.properties` using the guidelines and descriptions in the table below.

`com.unraveldata.advertised.url`	Defines the Unravel Server URL for HTTP traffic.	`com.unraveldata.advertised.url= http://LAN_DNS:3000`

`com.unraveldata.customer.organization`	Identifies your installation for reporting purposes.	`com.unraveldata.customer.organizatio`
`com.unraveldata.tmpdir`	Location where Unravel's temp file will reside	`com.unraveldata.tmpdir=/srv/unravel/tmp`
`com.unraveldata.history.maxSize.week`	Sets retention for search data.	`com.unraveldata.history.maxSize.week`
`com.unraveldata.hive.hook.topic.num`	**Optional**. Defines the number of threads. Default is 1. Depending on job volume, increase this property to *N* where *N* is between 1 and 4, or roughly *ThousandJobsPerDay*/10.	`com.unraveldata.hive.hook.topic.num`
`com.unraveldata.s3.profile.config.file.path`	Location of Unravel s3 read-only access & secret key filename s3ro.properties.	`com.unraveldata.s3.profile.config.f usr/local/unravel/etc/s3ro.properties`
`com.unraveldata.emr.profile.config.file.path`	Location of Unravel EMR read-only access & secret key filename emrro.properties.	`com.unraveldata.emr.profile.config usr/local/unravel/etc/emrro.properties`
`com.unraveldata.cloudwatch.profile.config.file.path`	Location of Unravel Cloud Watch read-only access & secret key filename cwro.properties.	`com.unraveldata.cloudwatch.profile usr/local/unravel/etc/cwro.properties`

`com.unraveldata.spark.s3.profilesToBuckets`	s3 profile associated to the s3 bucket	`For 1 bucket, follow example as follows:`
		
		`com.unraveldata.spark.s3.profiles'`
		
		`For 2 buckets follow below example:`
		
		`com.unraveldata.spark.s3.profiles'`

`com.unraveldata.login.admins`	Defines the usernames that can access Unravel Web UI's admin pages. Default is `admin`.	`com.unraveldata.login.admins=admin`

com.unraveldata.s3.batch.monitoring.interval	**Optional** Defines the monitoring frequency. Default is 300 seconds (5 minutes). Set this property to 60 for lower latency.	com.unraveldata.s3.batch.monitoring
com.unraveldata.spark.eventlog.location	Where to find Spark event logs	com.unraveldata.spark.eventlog.location=hdfs://example.localdomain:8020/user/spark/spark-history/
oozie.server.url	Oozie URL	oozie.server.url=http://example.localdomain:11000/oozie

Enable/Disable Optional Daemons

Depending on your workload volume or kind of activity, you can enable or disable optional daemons at this point.

- If you are not using Oozie, disable `unravel_os3`:

```
# sudo chkconfig unravel_os3 off
```

- If you are not using Spark, disable `unravel_sw_1`:

```
# sudo chkconfig unravel_sw_1 off
```

- If you have 10000-20000 jobs per day, enable these workers:

```
# sudo chkconfig --add unravel_ew_2
# sudo chkconfig --add unravel_hhw_2
# sudo chkconfig --add unravel_jcw2_2
```

- If you have 20000-30000 jobs per day, enable these workers:

```
# sudo chkconfig --add unravel_ew_3
# sudo chkconfig --add unravel_hhw_3
# sudo chkconfig --add unravel_jcw2_3
```

- If you have more than 30000 jobs per day, enable these workers:

```
# sudo chkconfig --add unravel_ew_4
# sudo chkconfig --add unravel_hhw_4
# sudo chkconfig --add unravel_jcw2_4
```

Enable Collection from Hive Metastore

If you have a central Hive Metastore, you can inform Unravel Server to enable more monitoring and analysis:

For MySQL use avax.jdo.option.ConnectionDriverName=com.mysql.jdbc.Driver instead of the postgresql driver

Substitute thecorrect values for your site.

```
# echo "
 javax.jdo.option.ConnectionURL=jdbc:postgresql://10.0.0.10:7432/hive_nqz
 javax.jdo.option.ConnectionDriverName=org.postgresql.Driver
 javax.jdo.option.ConnectionUserName=hive_nqz
 javax.jdo.option.ConnectionPassword=123456789abc
 " | sudo tee -a /usr/local/unravel/unravel.properties
```

Adjust Storage Locations [HVD]

Prepare symlinks from `/srv/unravel` to ephemeral (`$UNRAVEL_EPHEMERAL`) and durable (`$UNRAVEL_DURABLE`) storage here. During initial install, areas under `/srv` are created and we will move them for performance reasons because the boot partition does not have the best performance (although the boot partition can be useful for low volume because it is durable). Some areas should be ephemeral and some durable/provisioned-iops:

- Make sure daemons are stopped **[HVD]** :

```
# sudo /etc/init.d/unravel_all.sh stop
```

- Check that all Unravel daemons are stopped:

```
# ps -U unravel -f
```

- If any processes are owned by Unravel, stop them with a kill command.
- Check that destination areas are present **[HVD]** :

```
# df -h $UNRAVEL_EPHEMERAL
# df -h $UNRAVEL_DURABLE
```

- Move files and create symlinks **[HVD]** :

```
# sudo /bin/mv /srv/unravel/k_data $UNRAVEL_DURABLE/k_data
# sudo ln -s $UNRAVEL_DURABLE/k_data /srv/unravel/k_data

# sudo /bin/mv /srv/unravel/zk_1_data $UNRAVEL_DURABLE/zk_1_data
# sudo ln -s $UNRAVEL_DURABLE/zk_1_data /srv/unravel/zk_1_data

# sudo /bin/mv /srv/unravel/zk_2_data $UNRAVEL_DURABLE/zk_2_data
# sudo ln -s $UNRAVEL_DURABLE/zk_2_data /srv/unravel/zk_2_data
```

- This `zk` is intentionally on a different kind of storage: [HVD]

```
# sudo /bin/mv /srv/unravel/zk_3_data $UNRAVEL_EPHEMERAL/zk_3_data
# sudo ln -s $UNRAVEL_EPHEMERAL/zk_3_data /srv/unravel/zk_3_data

# sudo /bin/mv /srv/unravel/db_data $UNRAVEL_DURABLE/db_data
# sudo ln -s $UNRAVEL_DURABLE/db_data /srv/unravel/db_data

# sudo /bin/mv /srv/unravel/s_1_data $UNRAVEL_DURABLE/s_1_data
# sudo ln -s $UNRAVEL_DURABLE/s_1_data /srv/unravel/s_1_data

# sudo /bin/mv /srv/unravel/tmp $UNRAVEL_EPHEMERAL/tmp
# sudo ln -s $UNRAVEL_EPHEMERAL/tmp /srv/unravel/tmp

# sudo /bin/mv /srv/unravel/log_hdfs $UNRAVEL_EPHEMERAL/log_hdfs
# sudo ln -s $UNRAVEL_EPHEMERAL/log_hdfs /srv/unravel/log_hdfs

# sudo /bin/mv /srv/unravel/tmp_hdfs $UNRAVEL_EPHEMERAL/tmp_hdfs
# sudo ln -s $UNRAVEL_EPHEMERAL/tmp_hdfs /srv/unravel/tmp_hdfs
```

Start Unravel Server

```
# sudo /etc/init.d/unravel_all.sh start
```

Set Up an External DB [HVD]

For performance and ease of management, using an RDS MySQL instead of the bundled mysql is recommended.

Set Up RDS MySQL [HVD]

- RDS Security Group: create on VPC of Unravel Server and allow access from Unravel Server security group

- Create db instance
  - Select multi-AZ
  - Select `db.m3.large`
  - Select provisioned IOPS 1000
  - Select SSD size (capacity depends on activity level)
    - 770GB for 180 days retention (number of days set in unravel.properties)
    - 1.54TB for 360 days retention
  - No read-only replicas needed
  - Prefer overlap with Unravel Server AZ
  - Retain 7 days of snapshots
  - Specify `unravel` or `ElasticMapReduce-master` RDS Security group, depending on which was picked earlier
  - Name db instance "`unravelX` " (X=p for production, X=d for development, X=t for test)
  - Use MySQL 5.5.42
  - Disable auto-minor-upgrade
- Define a parameter group "`unravel`"
  - key_buffer_size = 268435456
  - max_allowed_packet = 33554432
  - table_open_cache = 256
  - read_buffer_size = 262144
  - read_rnd_buffer_size = 4194304
  - max_connect_errors=2000000000
  - open_files_limit=9000
  - innodb_open_files=9000
  - character_set_server=utf8
  - collation_server = utf8_unicode_ci
  - innodb_autoextend_increment=100
  - innodb_additional_mem_pool_size = 20971520
  - innodb_log_file_size = 134217728
  - innodb_log_buffer_size = 33554432
  - innodb_flush_log_at_trx_commit = 2
  - innodb_lock_wait_timeout = 50
- Modify `unravelX` instance (X=p for production, X=d for development, X=t for test)
  - Use `unravel` parameter group
  - Take effect immediately
- create "`unravel`" user
  - Pick a db password for user `unravel`
    - DB_PASSWORD="$(cat /dev/urandom | tr -cd 'a-zA-Z0-9' | head -c10)"
  - Log into RDS mysql instance from Unravel Server as admin/master user and do the following commands, substituting above `DB_PASSWORD` below:

```
CREATE DATABASE unravel_mysql_prod;
 COMMIT;
 CREATE USER 'unravel'@'%' IDENTIFIED BY 'changeme';
 GRANT ALL PRIVILEGES ON unravel_mysql_prod.* TO 'unravel'@'%';
 FLUSH PRIVILEGES;
 UPDATE user SET Password=PASSWORD('${DB_PASSWORD}') WHERE
 user.User='unravel';
 FLUSH PRIVILEGES;
 COMMIT;
```

```
QUIT;
```

- Log into RDS mysql as user `unravel` to test `DB_PASSWORD`.

Dump Bundled DB with Schema [HVD]

On Unravel Server, do the following to dump the db with schema:

```
# sudo /etc/init.d/unravel_all.sh stop
# sudo /etc/init.d/unravel_db start
 RPW=$(grep 'DB_ROOT_PASSWORD' /root/unravel.install.include | awk -F=
 '{ print $2 }')
 [ ! "$RPW" ] && echo "could not find Unravel bundled db root password"
 DEST_FILE=/tmp/unravel.backup.$(export TZ=UTC;date '+%Y%m%dt%H%MZ').sql.gz
 /usr/local/unravel/mysql/bin/mysqldump --host=127.0.0.1 -u root --port=3316
 --opt \
 --complete-insert --tz-utc --skip-comments --single-transaction --insert-
ignore \
 unravel_mysql_prod -p$RPW | gzip > $DEST_FILE
```

Load DB with Schema Into RDS MySQL [HVD]

Load the initial db with schema into the RDS MySQL instance, substituting `$RDS_HOST` for host of the RDS MySQL db; change the port if a different port is used. The password for `unravel` user on mysql will be needed:

```
# gunzip --stdout $DEST_FILE | /usr/local/unravel/mysql/bin/mysql --host=
$RDS_HOST -u unravel -p --port=3306 --force unravel_mysql_prod
```

Configure to Use RDS MySQL [HVD]

Configure `unravel.properties` to use the new database:

- Edit the file:

  ```
  # sudo vi /usr/local/unravel/etc/unravel.properties
  ```

- Adjust existing properties to point to new RDS MySQL. Change example values as needed:

  ```
  unravel.jdbc.username=unravel
  unravel.jdbc.url=jdbc:mysql://
  unravelrds.something.REGION.rds.amazonaws.com:3306/unravel_mysql_prod
  unravel.jdbc.password=******
  ```

Restart Unravel Server

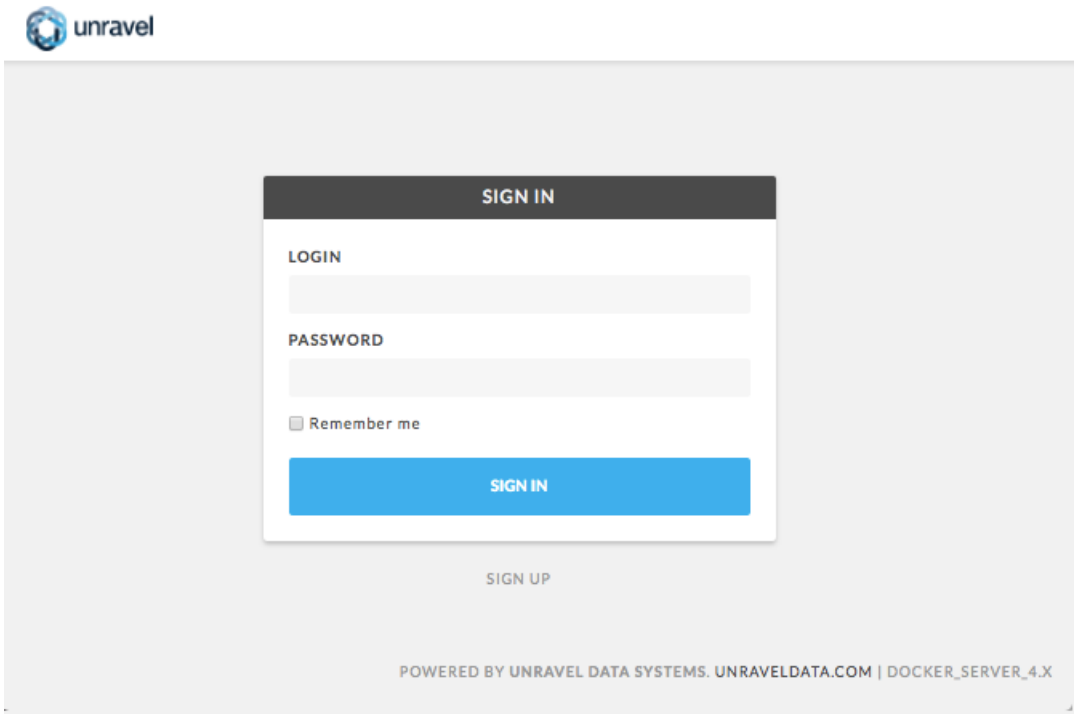
After edits to `com.unraveldata.login.admins` in `/usr/local/unravel/etc/unravel.properties` it is necessary to restart Unravel Server in order for changes to take effect. The `echo` command shows the page to visit with your browser. If you are using an SSH tunnel or HTTP proxy, you might need to make adjustments to the host/IP of the URL:

```
# sudo /etc/init.d/unravel_all.sh start
# sleep 60
# echo "http://$(hostname -f):3000/"
```

4. Log into Unravel Web UI

Using a web browser, navigate to `http://($hostname -f):3000` and login as user "`admin`" with password "`unraveldata`".

For the free trial version, use the Chrome web browser.



Unravel Web UI Login Screen

Congratulations!

Unravel Server is up and running. You should begin to see information collected. For instructions on using Unravel Web UI, see the User Guide.

5. (Optional) Enable Additional Data Collection/Instrumentation

For instructions, see Part 2: Install Unravel Hive Sensor on Qubole Hadoop2/Hive Cluster, Part 3: Install Unravel Spark Sensor on New or Existing Qubole Spark Cluster and Part 4: Modify Amazon EMR Cluster Bootstrap/Setup.

Attachments:

- worddav95f1e8fd13958fae391afc8e6f9ad03d.png (image/png)
- worddavb613e73f110587889f8b58fd6cdcdfa4.png (image/png)
- worddav17bd11fa4b44a52a180fec44ac2a622c.png (image/png)
- image2017-2-26_0-20-12.png (image/png)
- logo_small.png (image/png)

Part 2: Install Unravel Hive Sensor on Qubole Hadoop2/Hive Cluster

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Installation Guide for Amazon Elastic MapReduce (Amazon EMR) and Qubole Clusters

Follow these steps only if you have a Qubole-based Hadoop2/Hive cluster.

Highlighted text indicates where you must substitute your particular values.

Introduction

This topic explains how to install Unravel Sensor (Hive Hook) on Qubole-based Hadoop2/Hive clusters. Unravel Sensor collects information about Hive queries in Hadoop and pushes it to Unravel Server.

Workflow Summary

1. Step 1 is a one-time only step.
2. If the Qubole cluster already exists (is already running), do a "setup" procedure: follow the steps 3 to 5 in Hive Bootstrap and Unravel Hive Hook Sensor Setup.

Hive Bootstrap and Unravel Hive Hook Sensor Setup:

{location_in_s3_where_unravel_jar_folder} - is the s3 location where the unravel hive hook jar will be accessed

{Unravel_Hostname_IP} - is the Unravel server's fully qualified internal IP address, preferably the LAN (private) IP if in the same availability zone

1. Add a one-time Unravel Hive Bootstrap into Qubole's control panel on the left-hand side in the "Hive Bootstrap" section, as follows:

```
add jar  s3n://{location_in_s3_where_unravel_jar_folder} /unravel-
hive-0.13.0-hook.jar;

set com.unraveldata.hive.hdfs.dir=/user/unravel/HOOK_RESULT_DIR;
set
 hive.exec.driver.run.hooks=com.unraveldata.dataflow.hive.hook.HiveDriverHook;
set hive.exec.pre.hooks=com.unraveldata.dataflow.hive.hook.HivePreHook;
set hive.exec.post.hooks=com.unraveldata.dataflow.hive.hook.HivePostHook;
set
 hive.exec.failure.hooks=com.unraveldata.dataflow.hive.hook.HiveFailHook;
set com.unraveldata.host=<Unravel_Hostname_FQDN_Internal_IP>;
set com.unraveldata.hive.hook.tcp=true;
```

2. Determine the Hive version that Qubole uses, and use that value for `HIVE_VER` in the commands below.

   To determine the Hive version Qubole uses, see http://docs.qubole.com/en/latest/faqs/hive/version-hive-qubole-provide.html.

   `HIVE_VER` must be a Hive version that Unravel Server supports: either `1.2.0` or `0.13.0`. Be sure to use either of these values in exactly this format.
3. On the master node of the Qubole Hadoop2/Hive cluster, check that Unravel Server is reachable.

   Ensure the security group on the master/slave allows TCP port #3000 and #4043 accessible.

```
# curl http://{Unravel_Hostname_IP}:3000/version.txt
```

   If the version information is not visible, adjust security groups and routing and try again.
4. SSH to the master server of the existing Qubole Hadoop2/Hive cluster, run the following commands:

   a. Use `curl` to get the script from Unravel Server:

```
# curl http://{Unravel_Hostname_IP}:3000/hh/unraveldata-clients/
unravel_qubole_setup.sh > unravel_qubole_setup.sh
```

   b. Ensure `wget` is installed, and run the script on each server in the cluster:

   Files will be installed under:

   Hive hook jar is /usr/local/unravel_client/.

   Spark jar is /usr/local/unravel-spark/.

   Unravel ES is /usr/local/unravel_es/.

On the master node, /etc/init.d/unravel_es service will be defined and started.

```
# yum install -y wget
# chmod 755 unravel_qubole_setup.sh
# sudo ./unravel_qubole_setup.sh install -y --unravel-server
 UNRAVEL_HOST_IP:3000
```

5. Verify if the setup works in Qubole by invoking Analyze and execute following Hive test query:

```
set hive.on.master=true ;
select count(*) from default_qubole_memetracker;
```

Attachments:

worddav95f1e8fd13958fae391afc8e6f9ad03d.png (image/png)

logo_small.png (image/png)

Part 3: Install Unravel Spark Sensor on New or Existing Qubole Spark Cluster

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Installation Guide for Amazon Elastic MapReduce (Amazon EMR) and Qubole Clusters

Introduction

Unravel Sensor can collect information about Spark applications. This guide describes how to install Unravel Sensor for Qubole-based Spark clusters. The information here applies to Spark versions 1.5.x through 2.0.x and Unravel 3.4-4.1. Highlighted text indicates where you must substitute your particular values.

`HIGHLIGHTED` text and text with brackets ({ }), unless otherwise noted, indicates where you must substitute your particular values for the text.

`UNRAVEL_HOST_IP` must be a fully qualified DNS or the IP address.

`PATH_TO_SENSOR_JARS` : is the absolute path to the location of the sensor jars.

Add Unravel Spark Instrumentation to New Qubole Spark Cluster

1. Configure Unravel s3 read-only credentials in `/usr/local/unravel/etc/s3ro.properties` as follows:

   This sample `s3ro.properties` file has multiple s3 profiles and access keys.

```
[default]
aws_access_key_id = {ACCESS_KEY_VALUE1}
aws_secret_access_key = {SECRET_KEY_VALUE1}

[profile_name_2]
aws_access_key_id = {ACCESS_KEY_VALUE2}
aws_secret_access_key = {SECRET_KEY_VALUE2}
```

   Note: Substitute actual values for ACCESS_KEY_VALUEx and SECRET_KEY_VALUEx.

2. Edit `/usr/local/unravel/etc/unravel.properties` so that Unravel Server incorporates `s3ro.properties`:

```
com.unraveldata.s3.profile.config.file.path=/usr/local/unravel/etc/
s3ro.properties
com.unraveldata.spark.s3.profilesToBuckets=<default>:<s3ro_bucket1>,<profile_name_2>:
```

3. Restart the Unravel ETL daemon(s):

```
# sudo /etc/init.d/unravel_all.sh stop-etl
# sudo /etc/init.d/unravel_all.sh start
```

4. Ensure ports 3000 (for web UI access) and 4043 (from cluster) are open for incoming traffic. These should **not** be open to the public Internet.
5. Verify that port 3000 is open by running a `curl` command from a potential Unravel user machine as follows :

   If the version information is not visible (request timeout), then adjust security groups, firewalls, etc. and try again. For VPCs, it might be necessary to add a route.

```
# curl http://{UNRAVEL_HOST_IP}:3000/version.txt
```

6. Copy the Unravel Spark bootstrap file from Unravel Server to your Spark Qubole cluster, using `curl`:

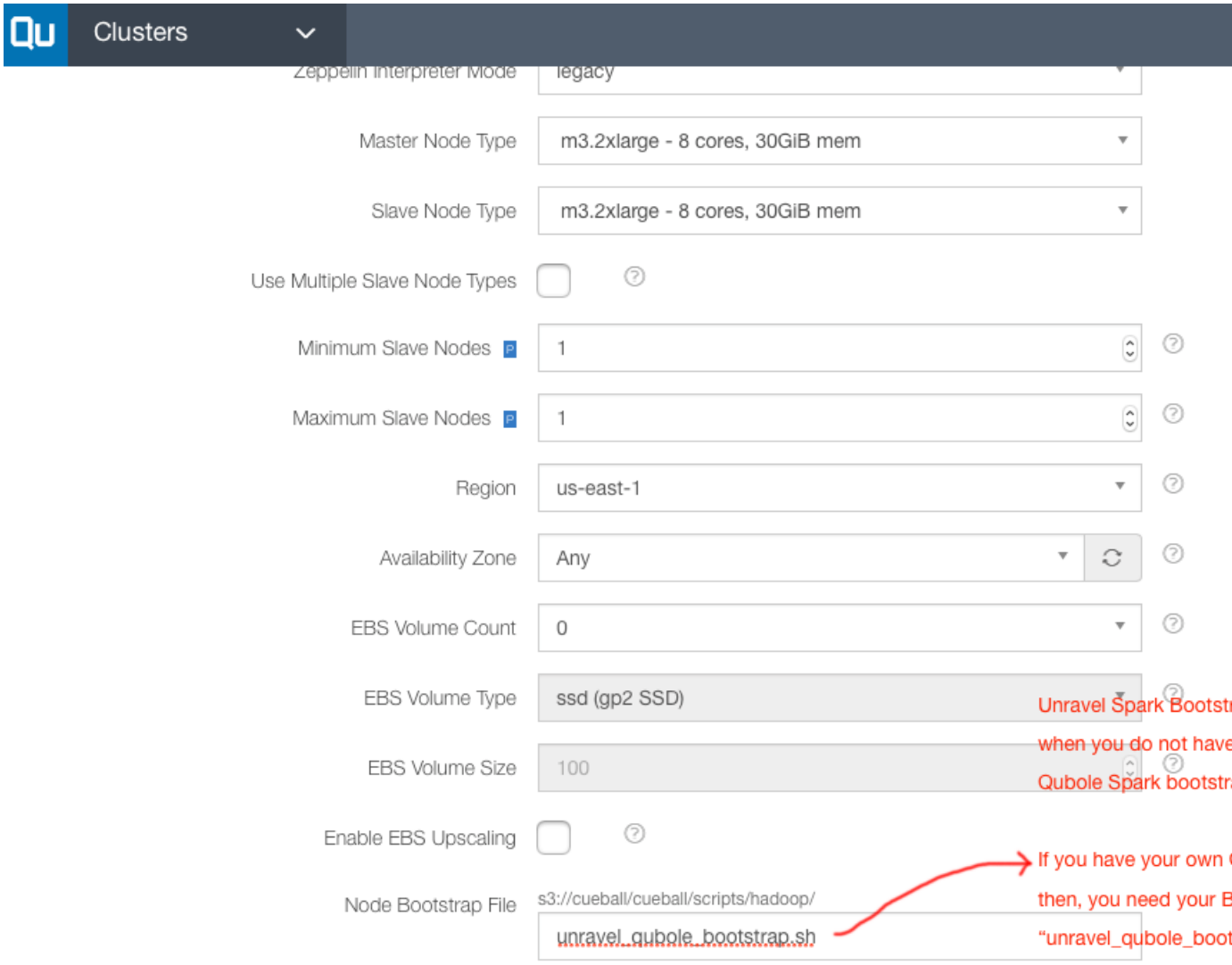
```
# curl http://{UNRAVEL_HOST_IP}:3000/hh/unraveldata-clients/
unravel_qubole_bootstrap.sh > unravel_qubole_bootstrap.sh
```

   Note: Substitute the actual host for UNRAVEL_HOST_IP.

7. Copy Unravel Spark bootstrap file to your s3:// `Bootstrap_location_for_Spark Qubole Cluster`

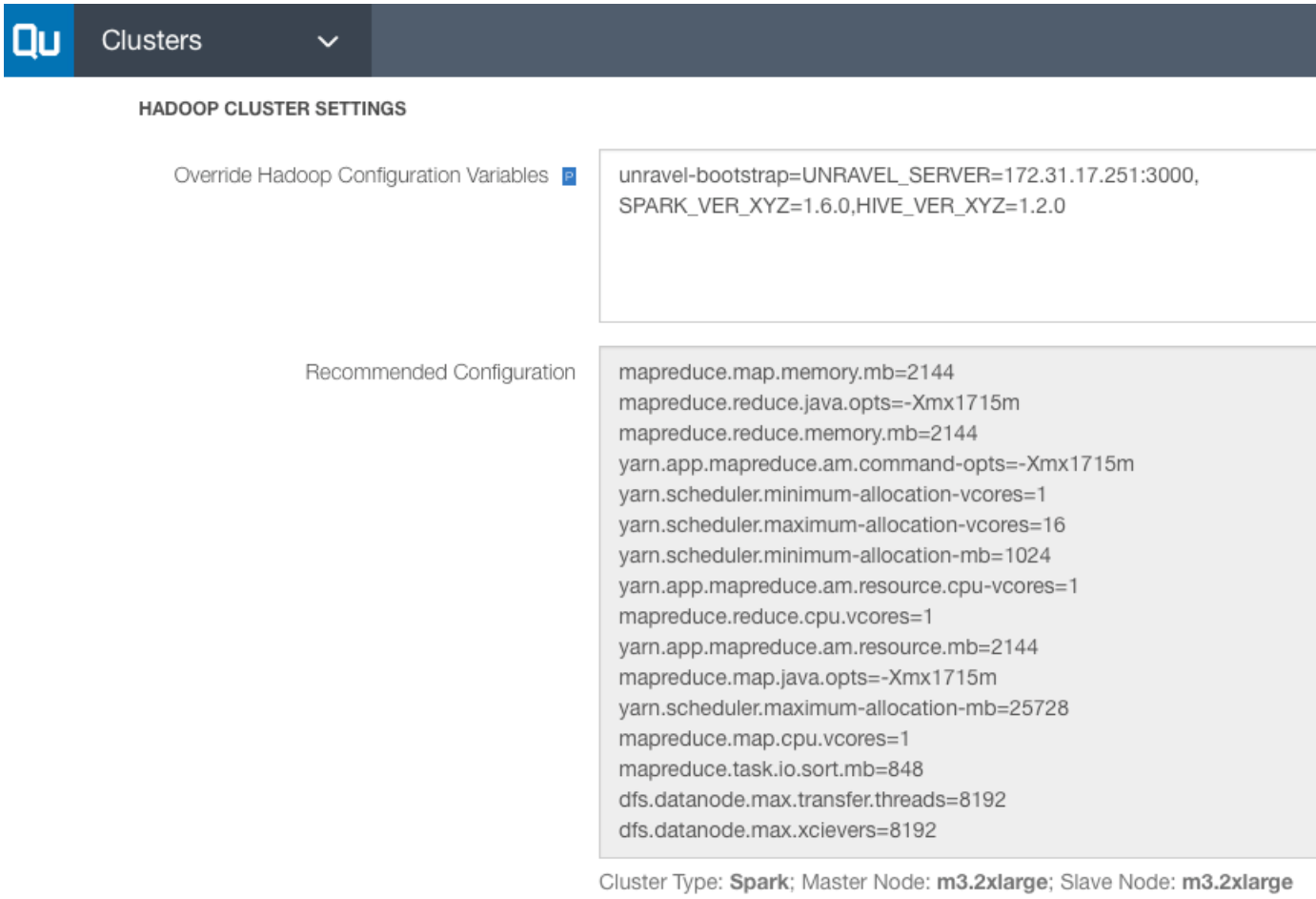
```
aws s3 cp unravel_qubole_bootstrap.sh s3://{Bootstrap_location_for_Spark
 Qubole cluster}/unravel_qubole_bootstrap.sh
```

8. In Qubole's **Edit Cluster Setting**, if you **do not** have your own customized Node Bootstrap file, add `unravel_qubole_bootstrap.sh` into this field as shown in the diagram:



9. In Qubole, scripts do not take input parameters. Therefore, Unravel's bootstrap script takes all the required parameters from the Hadoop configuration. You can customize your Hadoop configuration through specific parameters within the **Override Hadoop Configuration Variables** section as shown in the diagram below. As you can see, **unravel-bootstrap** is the property name you must use

prepend to all configuration parameters relevant to Unravel. *Note: Separate parameters with commas.*



10. Add these settings to `unravel-bootstrap` for Spark clusters:

```
unravel-bootstrap=UNRAVEL_SERVER=UNRAVEL_HOST_AND_PORT,
 SPARK_VER_XYZ=SPARK_VER_XYZ[,SPARK_APP_LOAD_MODE=SPARK_APP_LOAD_MODE,WRAPPED_SCRIPT=
```

`SPARK_VERSION _X.Y.Z - target Spark version (eg. 1.3.0, 1.6.0, 2.0.1, etc.)`

Note:

`The parameters in square brackets [ ] above are optional. Their meanings are:`

`SPARK_APP_LOAD_MODE` is either OPS or DEV mode, and if not explicitly specified defaults to OPS mode. For details on application loading modes, see Appendix.

`WRAPPED_SCRIPT` is the full S3 path to your original bootstrap script. Your bootstrap script is invoked from `unravel_qubole_bootstrap.sh`.

11. Confirm that the `unravel_es` service is installed:

a. Open an SSH session to the Qubole master node.

b. Run this command to check that the unravel_es service has been started:

```
# ps -aux | grep unravel_es
```

(Optional) Add Unravel Spark Instrumentation to an Existing Qubole Spark Cluster

1. Ensure that you have configured s3 read-only credentials in `/usr/local/unravel/etc/s3ro.properties` for Unravel, as described in steps 1 and 2 above.
2. Obtain following essential Spark version files from Unravel Server:
   a. To obtain Spark version 1.6.x zip file:

   ```
   # wget http://{UNRAVEL_HOST_IP}:3000/hh/unravel-sensor-for-spark-bin-1.6.zip
   ```

   b. To obtain Spark version 2.0.x zip file:

   ```
   # wget http://{UNRAVEL_HOST_IP}:3000/hh/unravel-sensor-for-spark-bin-2.0.zip
   ```

   Unzip the archive `unravel-sensor-for-spark-bin-1.6.zip` and ensure via the bootstrap action that the included jars are deployed on Qubole nodes at `PATH_TO_SENSOR_JARS` which is used in the configuration settings outlined in step 4.
3. Obtain EMR jar files and snippet script that should be added to the bootstrap action to start the unravel_es process:

   ```
   #wget http://{UNRAVEL_HOST_IP}:3000/hh/unraveldata-clients/snippets/run-es.sh
   #wget http://{UNRAVEL_HOST_IP}:3000/hh/unravel-emr-sensor.jar
   ```

   Ensure that `unravel-emr-sensor.jar` is deployed on the Qubole nodes via the bootstrap action. Preferably, deploy `unravel-emr-sensor.jar` at the same path of `PATH_TO_SENSOR_JARS` as used above.
4. Edit `run-es.sh` snippet script to customize following two parameters to fit into your Qubole Spark environment:

   ```
   # Replace following two parameters into your Qubole Spark environment #
   UNRAVEL_HOST={UNRAVEL_HOST_IP}
   UNRAVEL_EMR_SENSOR_JAR={PATH_TO_SENSOR_JARS}
   ```

5. Spark configuration can be provided in the Qubole console at bootstrap or directly inside `spark-defaults.conf` configuration file, if the cluster is already started.

   ```
   spark.unravel.server.hostport {UNRAVEL_HOST_IP}:4043
   spark.driver.extraJavaOptions -javaagent:{PATH_TO_SENSOR_JARS}/
   btrace-agent.jar=bootClassPath={PATH_TO_SENSOR_JARS}/
   unravel-boot.jar,systemClassPath=<PATH_TO_SENSOR_JARS>/
   unravel-sys.jar,script=DriverProbe.class:SQLProbe.class -
   Dcom.sun.btrace.FileClient.flush=-1
   spark.executor.extraJavaOptions -javaagent:{PATH_TO_SENSOR_JARS}/
   btrace-agent.jar=bootClassPath={PATH_TO_SENSOR_JARS}/unravel-
   boot.jar,systemClassPath={PATH_TO_SENSOR_JARS}/unravel-
   sys.jar,script=ExecutorProbe.class -Dcom.sun.btrace.FileClient.flush=-1
   ```

6. Edit `zeppelin-env.sh` by appending the following values to `ZEPPELIN_JAVA_OPTS` as follows:

   ```
   export ZEPPELIN_JAVA_OPTS="-Dcom.sun.btrace.FileClient.flush=-1 -
   javaagent:{PATH_TO_SENSOR_JARS}/btrace-
   agent.jar=bootClassPath={PATH_TO_SENSOR_JARS}/unravel-
   boot.jar,systemClassPath={PATH_TO_SENSOR_JARS}/unravel-
   sys.jar,script=DriverProbe.class:SQLProbe.class"
   ```

   The Zeppelin configuration is located at `/usr/lib/zeppelin/conf/zeppelin-env.sh`.

Appendix

Application Loading Modes for Spark Applications: OPS, DEV, and BATCH

There are three modes in which Spark applications can be loaded into Unravel Web UI:

- OPS mode shows applications in the UI after the application is done.
- DEV mode shows applications in the UI as soon as the first job of the Spark application completes.
- BATCH mode loads applications for which the sensor was not enabled at the time the application has been run.

You can specify the application load mode for the bootstrap script by setting `SPARK_APP_LOAD_MODE` to `OPS` or `DEV`. By default, if not explicitly specified, `SPARK_APP_LOAD_MODE` is `OPS`. All three modes can coexist in the same Unravel Server. For example, all Spark applications from Qubole cluster A can be loaded in OPS mode, all Spark applications from Qubole cluster B can be loaded in DEV mode, and all Spark applications from Qubole cluster C can be loaded in BATCH mode.

When to Use Which Mode

- Unravel recommends using OPS mode as the cluster-side default for all Qubole clusters. The OPS mode has been rigorously benchmarked to have less than 1.3% CPU and memory overhead. Both the OPS and DEV modes use the Unravel Spark sensor, which is enabled via modifications to `spark.driver.extraJavaOptions`. The key difference is that OPS mode sets `-Dcom.unraveldata.spark.sensor.disableLiveUpdates=true` while DEV mode sets `-Dcom.unraveldata.spark.sensor.disableLiveUpdates=false` (false is the default).
- DEV mode is useful during Spark application development. This mode shows a Spark application as soon as the first Spark job of the application finishes. For long running applications, this functionality is useful, as the application is shown in the UI while the application is running. In addition, DEV mode shows applications even when the Spark event log is not being persisted to HDFS or S3. This is an advantage in situations like Spark on Mesos.
- In a Qubole cluster that is using OPS mode as the default, DEV mode can be obtained for individual Spark applications by overriding `spark.driver.extraJavaOptions` to include `-Dcom.unraveldata.spark.sensor.disableLiveUpdates=false`.
- BATCH mode is always on and does not interfere with application performance in any way since the loading is entirely outside the application execution path. For details see the next section.

Loading Applications in Batch Mode

In order to load Spark apps in BATCH mode, Unravel Server must pull the Spark event log file either from S3 or from HDFS. The data collected in the UI is less detailed than when Unravel Sensor is enabled (for instance, detailed resource usage metrics are unavailable). The BATCH mode of operation is helpful to load all of the applications that have been run in the past, before Unravel Sensor was installed. To enable the batch mode, add the following property to `/usr/local/unravel/etc/unravel.properties` and restart the Spark worker daemon(s) as listed earlier:

```
com.unraveldata.spark.eventlog.location=hdfs://NAMENODE_IP_PORT/user/spark/
applicationHistory/
```

Note:

Currently, only one event log location is supported.

Attachments:

logo_small.png (image/png)

worddav95f1e8fd13958fae391afc8e6f9ad03d.png (image/png)

image2017-6-2_19-21-20.png (image/png)

hadoop_cluster_settings1.png (image/png)

hadoop_cluster_settings2.png (image/png)

Part 4: Modify Amazon EMR Cluster Bootstrap/Setup

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Installation Guide for Amazon Elastic MapReduce (Amazon EMR) and Qubole Clusters

Follow these steps only if you have an Amazon EMR cluster.

Introduction

This topic explains how to modify and Amazon EMR cluster's bootstrap/setup for Unravel Server 4.0. For instructions that correspond to older versions of Unravel Server, contact Unravel Support.

Steps marked [HVD] are for production high volume and durable installations can be skipped for PoCs.

Workflow Summary

1. Add your AWS account number(s) to the Unravel Data main s3 bucket policy.
2. Get the bootstrap script(s).
3. Integrate the bootstrap script(s) into your Amazon EMR cluster(s).

Table of Contents

1. Add Your AWS Account Number(s) to the Unravel Data Main S3 Bucket Policy

Prior to doing below steps, ensure that your AWS account number(s) is added to the Unravel Data main s3 bucket policy for `s3://unraveldata-client` by asking Unravel Support to do this; alternatively, the scripts can be found on Unravel Server in the directory `/usr/local/unravel/install_bin/unraveldata-clients`.

2. Get the Bootstrap Script(s)

To gain access to the s3 bucket that contains the bootstrap scripts (`s3://` unraveldata-clients/), add a one-time new Inline policy to **EMR_DefaultRole** as follows:

1. In the IAM management console on the left-hand side, click **Roles** | **EMR_DefaultRole** .
2. Click **Inline Policies** | **Create Role Policy**, and select **Custom Policy**.
3. Click **Select**.
4. Enter the policy name as you wish.
5. Copy and paste below policy into the **Policy Document:**

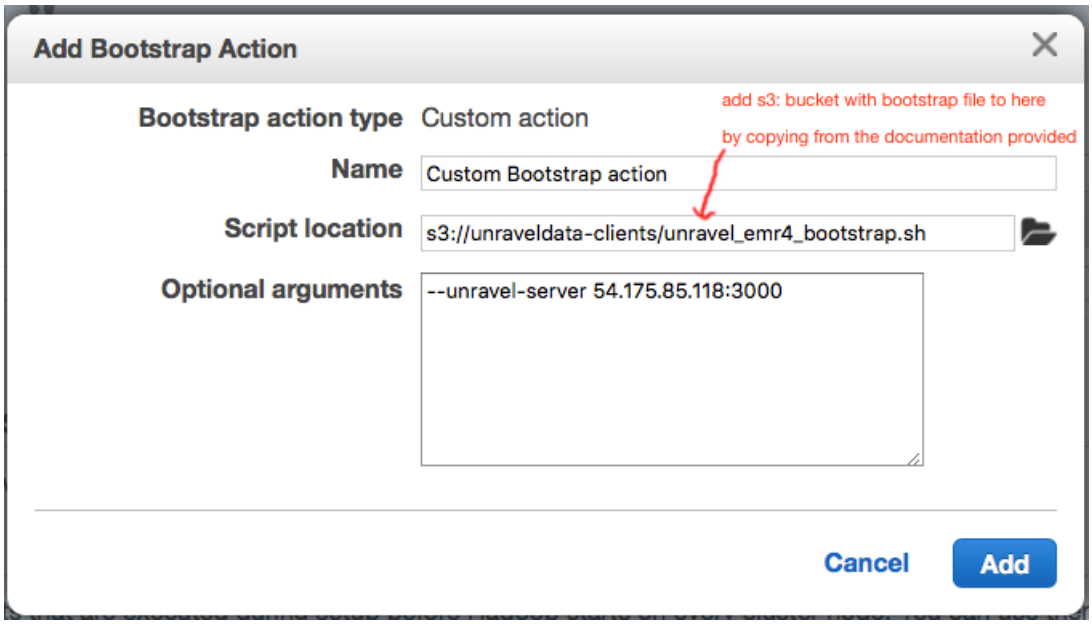
```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "getunraveldataclients3files",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:Get*"
            ],
            "Resource": [
              "arn:aws:s3:::unraveldata-clients/*"
            ]
        }
    ]
}
```

6. Save it by clicking **Apply Policy**.

When you create a new Amazon EMR cluster, be sure to add a bootstrap action as shown in the IAM screenshot below. You need to copy and paste the full pathname of the bootstrap action (script) into the **Script location** field.

The full pathname is the s3 bucket name plus the filename of the script. Important Note: **Do not** use the folder glyph



For guidance on which script to use, see the table below.

`unravel_emr_bootstrap.sh`	`s3://unraveldata-clients/install_bin/unraveldata-clients/`		Amazon EMR 3.x Hive
`unravel_emr4_bootstrap.sh`	`s3://unraveldata-clients/install_bin/unraveldata-clients/`		Amazon EMR 4.x Hive

3. Integrate the Bootstrap Script(s) into Your Amazon EMR Cluster(s)

For Persistent ("Long-Running" or "Existing") Amazon EMR Clusters

Hive Applications:

Unravel does not load data from a cluster until the cluster is instrumented. Use the steps here to set up an existing cluster.

1. Identify the LOCAL_IP address of your Unravel Server.
2. Download `s3://unraveldata-clients/unravel_emr_setup.sh` using the S3 console or '`aws s3`' command or other utility OR get it from the Unravel Server in `install_bin/unraveldata-clients`.
3. scp `unravel_emr_setup.sh` to `/tmp` of the cluster's master node (ssh user is `hadoop`).
4. Open an SSH session to the cluster's master node (ssh as user `hadoop`) and then do as `hadoop`:

```
cd /tmp
aws s3 cp s3://unraveldata-clients/unravel_emr_setup.sh .
chmod +x unravel_emr_setup.sh
./unravel_emr_setup.sh --unravel-server $LOCAL_IP:3000
```

To uninstall Hive instrumentation on an Amazon EMR cluster (perhaps because you want to upgrade the instrumentation), you simply run the same install script again with the `uninstall` argument:

```
cd /tmp
aws s3 cp s3://unraveldata-clients/unravel_emr_setup.sh .
chmod +x unravel_emr_setup.sh
./unravel_emr_setup.sh --uninstall
```

For Transient Amazon EMR Clusters

Hive Applications:

This is similar to the previous section on integrating an existing cluster except the script used as a bootstrap step is one of the following files:

File	S3 Bucket	Local Directory	Applies To
unravel_emr_bootstrap.sh	s3://unraveldata-clients/	`install_bin/unraveldata-clients/`	Amazon EMR 3.x Hive
unravel_emr4_bootstrap.sh	s3://unraveldata-clients/	`install_bin/unraveldata-clients/`	Amazon EMR 4.x Hive

Spark Applications:

The Unravel Server does not load data from a Spark cluster until the cluster is instrumented. Use the steps here to set up an existing cluster.

1. Identify the LOCAL_IP address of your Unravel Server
2. Download `s3://unraveldata-clients/unravel_emr_spark_setup.sh` using the S3 console or '`aws s3`' command or other utility OR get it from the Unravel Server in `install_bin/unraveldata-clients`.
3. scp `unravel_emr_setup.sh` to `/tmp` of the EMR cluster master node (ssh user is `ec2-user`).
4. Open an SSH session to the cluster's master node (ssh user is `ec2-user`) and then do as `ec2-user`:

```
cd /tmp
sudo ./unravel_emr_spark_setup.sh --unravel-server \
 $LOCAL_IP:3000 --client
```

Change `--client` to `--cluster` if you use the Spark driver in cluster mode.

Attachments:

- image2017-2-26_0-20-12.png (image/png)
- worddav95f1e8fd13958fae391afc8e6f9ad03d.png (image/png)
- logo_small.png (image/png)

worddavb613e73f110587889f8b58fd6cdcdfa4.png (image/png)
worddav17bd11fa4b44a52a180fec44ac2a622c.png (image/png)

Part 5: Additional Topics for EMR or Qubole

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Installation Guide for Amazon Elastic MapReduce (Amazon EMR) and Qubole Clusters

- Creating Active Directory Kerberos Principals and Keytabs for Unravel
- Sensors
  - Installing Unravel Sensor for Individual Applications Submitted Through spark-shell
  - Installing Unravel Sensor for Individual Applications Submitted Through spark-submit
  - Installing Unravel Sensor for Individual Hive Queries
- Workflows
  - Monitoring Airflow Workflows
  - Monitoring Oozie Workflows
  - Tagging Workflows
- Custom Configurations
  - Adding More Admins to Unravel Web UI
  - Configuring Multiple Hosts for Unravel Server
  - Creating Multiple Workers for High Volume Data
  - Defining a Custom TC Port
  - Integrating LDAP Authentication for Unravel Web UI
  - Setting Retention Time in Unravel Server
  - Setting Up Email for Auto Actions and Collaboration
- Connecting to a Hive Metastore
- Creating an AWS RDS CloudWatch Alarm for FreeStorageSpace
- Creating Application Tags
- Troubleshooting
  - Running Verification Scripts and Benchmarks
  - Sending Diagnostics to Unravel Support
- Uninstalling Unravel Server
- Upgrading Unravel Server

Installation Guide for Hortonworks Data Platform (HDP)

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1

This guide is compatible with:

- HDP 2.2-2.6

Ordered Steps

- Page: Part 1: Install Unravel Server on HDP
- Page: Part 2: Enable Additional Data Collection / Instrumentation for HDP
- Page: Part 3: Additional Topics for HDP

Part 1: Install Unravel Server on HDP

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Installation Guide for Hortonworks Data Platform (HDP)

Created by Paul Baclace, last modified by Ada-C on Aug 18, 2017

Introduction

This topic explains how to deploy Unravel Server 4.0-4.1 on HDP 2.2.x-2.5.x clusters using Ambari Web UI (AWU). Unravel Hive Hook is used to collect information about Hive queries in Hadoop. The information here applies to Hive versions 0.10.0 through 2.1.x and Unravel 4.0-4.1. For older versions of Unravel Server, contact Unravel Support.

Workflow Summary

1. Pre-installation check
2. Configure the host machine.
3. Install the Unravel Server RPM on the host machine.
4. Configure Unravel Server (basic/core options).
5. Log into Unravel Web UI.
6. (Optional) Configure Unravel Server (advanced options).

Table of Contents

Pre-Installation Check

The following installation requirements must be met for successful installation of Unravel.

Platform Compatibility

- HDP 2.2-2.6
- Hadoop 1.x - 2.x
- Hive 0.9.x - 1.2.x
- Spark 1.3.x - 2.0.x

Hardware

- Architecture: x86_64
- Cores: 8
- RAM: 64GB minimum
- Disk: `/usr/local/unravel` with 2.5GB free minimum
- Disk: `/srv/unravel` with 500GB free minimum
- For 10,000+ MR jobs per day, two or more gateway/edge nodes are recommended

Software

- Operating System: RedHat/Centos 6.4 - 7.3
- `libaio.x86_64` installed
- `SELINUX=permissive` (or disabled) should be set in `/etc/selinux/config`
- HDFS+Hive+YARN client/gateway, Hadoop and Hive commands in `PATH`
- If Spark is in use, Spark client gateway
- LDAP (AD or Open LDAP) compatible for Unravel Web UI user authentication (Open signup by default)
- On Unravel Edge-node server, please **do not** have zookeeper installed in same server

Access Permissions

- If Kerberos is in use, a keytab for principal hdfs (or read-only equivalent) is required for access to:
  - Access to YARN's "done dir" in HDFS
  - Access to YARN's log aggregation directory in HDFS
  - Access to Spark event log directory in HDFS
  - Access to file sizes under HIve warehouse directory
- Access to YARN Resource Manager REST API
  - principal needs right to find out which RM is active
- JDBC access to the Hive Metastore (read-only user is sufficient)

Network

- Port 3000 (or 4020) from users and entire cluster to Unravel Web UI
- HDFS ports open from Hadoop cluster to Unravel Server(s)
- For MR1, TaskTracker port open from Hadoop cluster to Unravel Server(s)
- For MR1/YARN, Hive Metadata DB port open to Unravel Server(s) for partition reporting
- UDP and TCP port 4043 open from entire cluster to Unravel Server(s)
- For Oozie, port 11000 open from Unravel Server(s) to the Oozie server
- Resource Manager (RM) port 8032 from Unravel Server(s) to the RM server(s)
- Port 4176, 4181 through 4189, 3316, 4091 must be available for localhost communication between Unravel daemons or services

1. Configure the Host Machine

Allocate a Cluster Gateway/Edge/Client Host with HDFS Access

For HDP, use Ambari Web UI to create a Gateway node configuration. The hadoop command must be present on the Unravel target server.

2. Install the Unravel Server RPM on the Host Machine

Get the Unravel Server RPM

Copy the RPM from the Unravel distribution server to the host machine using the username and password given to you by Unravel Support:

- For the free trial version:

```
scp "unraveltrial@trial.unraveldata.com:unravel-4.*.x86_64.rpm" .
```

- For the enterprise version:

```
scp $USER@dist.unraveldata.com:unravel-4.*.x86_64.rpm.$timestamp .
```

The precise RPM filename will vary. The version has the structure *x.y.b* where *b* is a build number that is imposed as an RPM epoch which means it takes precedence over version numbers for determining what is more up-to-date. The Unravel build numbers always increase and are allocated so that they respect the ordering of *x.y* versioning.

Install the Unravel Server RPM

Replace the asterisks as needed to be more selective.

```
sudo rpm -U unravel-4.*.x86_64.rpm*
```

The installation creates `/usr/local/unravel/` which contains the executables, scripts, and settings. User `unravel` is created. The initial internal database and other durable state are put in `/srv/unravel/` for larger storage. By default, the installation supports YARN. The master configuration file is in `/usr/local/unravel/etc/unravel.properties` and the logs are in `/usr/local/unravel/logs/`. The RPM installation creates user `unravel` if it does not already exist; `/etc/init.d/unravel_*` scripts for controlling its services as well as `/etc/init.d/unravel_all.sh` which can be used to manually stop, start, and get status of all daemons in proper order. The RPM installation also creates an HDFS directory for Hive Hook information collection. During initial install, a bundled database is used. This can be switched to use an externally managed MySQL for production. (The bundled database root mysql password will be stored in `/root/unravel.install.include` during installation.)

Do Host-Specific Post-Installation Actions

For HDP, there are no host-specific post-installation actions.

3. Configure Unravel Server (Basic/Core Options)

Modify `unravel.properties`

- Open `/usr/local/unravel/etc/unravel.properties` with `vi`.

```
sudo vi /usr/local/unravel/etc/unravel.properties
```

- Edit the values in `unravel.properties` using the guidelines and descriptions in the table below.

`com.unraveldata.advertised.url`	Defines the Unravel Server URL for HTTP traffic.	`com.unraveldata.advertised.url=http://LAN_DNS:3000`
`com.unraveldata.customer.organization`	Identifies your installation for reporting purposes.	`com.unraveldata.customer.organizatio`
`com.unraveldata.tmpdir`	Location where Unravel's temp file will reside	`com.unraveldata.tmpdir=/srv/unravel/tmp`
`com.unraveldata.history.maxSize.week`	Sets retention for search data.	`com.unraveldata.history.maxSize.week`

`com.unraveldata.hive.hook.tcp`	**Optional.** Defines the number of threads. Default is 1. Depending on job volume, increase this property to *N* where *N* is between 1 and 4, or roughly *ThousandJobsPerDay*/10.	`com.unraveldata.hive.hook.topic.num`
`com.unraveldata.job.collector.done`	Only modifiable through Unravel Web UI's configuration wizard.	`com.unraveldata.job.collector.done` `mr-history/done`
`com.unraveldata.job.collector.log.a`	Only modifiable through Unravel Web UI's configuration wizard.	`com.unraveldata.job.collector.log.a` `app-logs/*/logs/`
`com.unraveldata.login.admins`	Defines the usernames that can access Unravel Web UI's admin pages. Default is `admin`.	`com.unraveldata.login.admins=admin`
`com.unraveldata.s3.batch.monitoring`	**Optional.** Defines the monitoring frequency. Default is 300 seconds (5 minutes). Set this property to 60 for lower latency.	`com.unraveldata.s3.batch.monitoring`

`com.unraveldata.spark.eventlog.loca`	Where to find Spark event logs	`com.unraveldata.spark.eventlog.loca`
		`example.localdomain:8020/`
		`spark-history/`

`yarn.resourcemanager.webapp.address`	YARN resource manager web address URL	`yarn.resourcemanager.webapp.address` `example.localdomain:8088`

`oozie.server.url`	Oozie URL	`oozie.server.url=http://`
		`example.localdomain:11000/`
		`oozie`

If Kerberos is Enabled:

- Add authentication for HDFS...

(a) Create a keytab for unravel for daemons that run as `unravel` and put the file in `/etc/keytabs/unravel.keytab` (for example).

(b) Create a keytab for hdfs for the Unravel daemons that run as user `hdfs` and put the file in `/etc/keytabs/hdfs.keytab` (for example).

(c) Tell Unravel Server about it (env var for hdfs keytab location; substitute correct realm and /hostname, if applicable):

```
echo "export HDFS_KEYTAB_PATH=/etc/keytabs/hdfs.keytab
export HDFS_KERBEROS_PRINCIPAL=hdfs/myhost.mydomain@MYREALM
" | sudo tee -a /usr/local/unravel/etc/unravel.ext.sh
```

(d) Add properties for Kerberos:

```
echo "
com.unraveldata.kerberos.principal=unravel/myhost.mydomain@MYREALM
com.unraveldata.kerberos.keytab.path=/etc/keytabs/unravel.keytab
" | sudo tee -a /usr/local/unravel/unravel.properties
```

You can find the principal by using 'klist -kt KEYTAB_FILE'

If Ranger is Enabled:

Add these permissions...

`hdfs://user/unravel/HOOK_RESULT_DIR`	*	read+write	data transfer from Hive jobs when Unravel is not up
`hdfs://user/spark/applicationHistory`	hdfs	read	Spark event log
`hdfs://usr/history/done`	hdfs	read	MapReduce logs
`hdfs://tmp/logs`	hdfs	read	YARN aggregation folder
`hdfs://user/hive/warehouse`	hdfs	read	Obtain table partition sizes
Hive Metastore GRANT	hive	read	Hive table information

4. Convert Your Unravel Installation to HDP

Run the following commands on Unravel Server:

```
sudo /etc/init.d/unravel_all.sh stop
sudo /usr/local/unravel/install_bin/switch_to_hdp.sh
```

Note: This change will stick after RPM upgrades.

5. Update Site-Specific HDP Properties

The following site-specific properties in `/usr/local/unravel/etc/unravel.properties` were added or modified by the script above (`switch_to_hdp.sh`). Check `/usr/local/unravel/etc/unravel.properties` to see if you need to modify any of these properties:

```
# Repoint Unravel application logs directory
com.unraveldata.job.collector.done.log.base=/mr-history/done
com.unraveldata.job.collector.log.aggregation.base=/app-logs/*/logs/
com.unraveldata.spark.eventlog.location=hdfs:///user/spark/
applicationHistory/

# Add Hive Metastore database information for Unravel Hive Config
javax.jdo.option.ConnectionURL=jdbc:mysql://{hostname}:3306/{database_name}
javax.jdo.option.ConnectionDriverName=com.mysql.jdbc.Driver
javax.jdo.option.ConnectionUserName={HiveMetastoreUserName}
javax.jdo.option.ConnectionPassword={HiveMetastorePassword}
```

Verify above site-specific values using the Ambari Web UI (AWU):

- Verify `com.unraveldata.job.collector.done.log.base`

    - In AWU, on the left-hand side, click **MapReduce2** | **Configs**. Select the **Advanced** tab, and then select **Advanced mapred-site**.
        - Take note of the value of `mapreduce.jobhistory.done-dir` to enter into `unravel.properties` of "`com.unraveldata.job.collector.done.log.base=`"
- Verify `com.unraveldata.job.collector.log.aggregation.base`
    - In AWU, on the left-hand side, click **YARN** | **Configs**. Select the **Advanced** tab, and then select **Node Manager**.
        - Take note of the value of `yarn.nodemanager.remote-app-log-dir` to enter into `unravel.properties` of "`com.unraveldata.job.collector.log.aggregation.base=`"
- Verify Hive Metastore `javax.jdo.option.Connection`
    - In AWU, on the left-hand side, click **Hive** | **Configs**. Select the **Advanced** tab, and then select **Hive Metastore**.
        - Take note of following properties and their values to be entered into `unravel.properties`:
            - **Database URL**
            - **Database Host**
            - **JDBC Driver Class**
            - **Database Name**
            - **Database Username**
            - **Database Password** (if visible)

Restart Unravel Server

After edits to `com.unraveldata.login.admins` in `/usr/local/unravel/etc/unravel.properties` it is necessary to run the following script in order to make changes take effect. The `echo` command shows the page to visit with your browser. If you are using an ssh tunnel or http proxy, you might need to make adjustments.

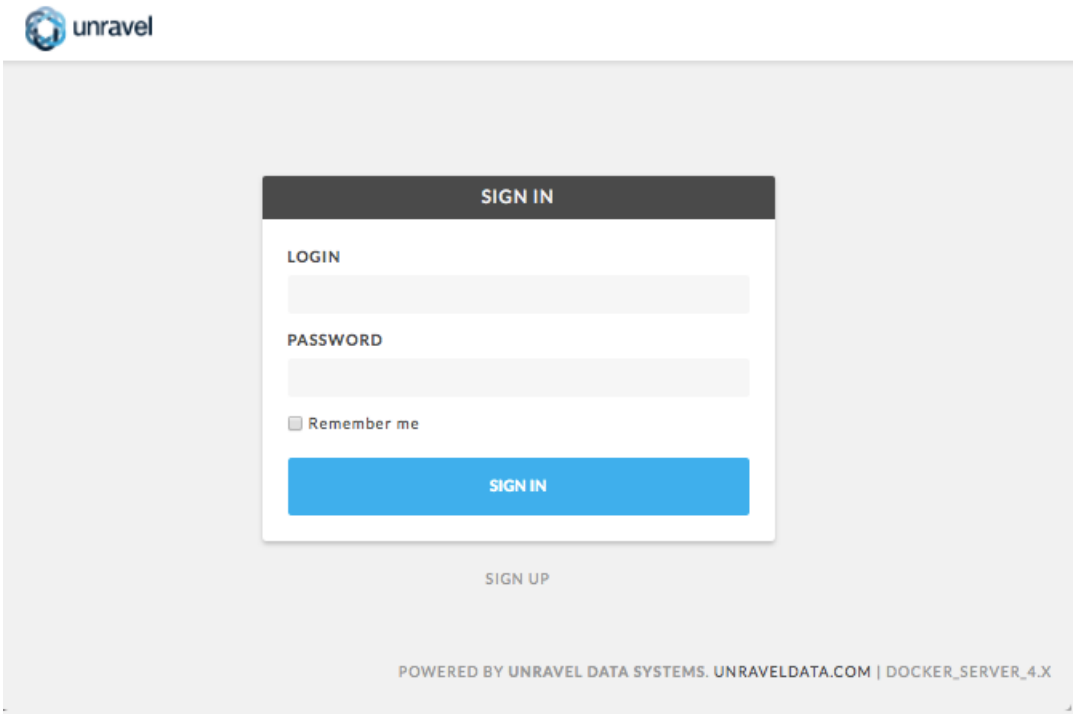
```
sudo /etc/init.d/unravel_all.sh start
sleep 60
echo "http://$(hostname -f):3000/"
```

This completes the basic/core configuration.

4. Log into Unravel Web UI

Using a web browser, navigate to `http://($hostname -f):3000` and login as user "`admin`" with password "`unraveldata`".

For the free trial version, use the Chrome web browser.



Unravel Web UI Login Screen

Congratulations!

Unravel Server is up and running. Unravel Web UI displays collected data. For instructions on using Unravel Web UI, see the User Guide.

5. (Optional) Configure Unravel Server (Advanced Options)

Enable Additional Data Collection/Instrumentation

Install the Unravel Sensor Parcel on gateway/edge/client nodes that are used to submit Hive queries to push additional information to Unravel Server. For details, see Part 2: Enable Additional Data Collection / Instrumentation for HDP.

Run the Unravel Web UI Configuration Wizard

Run the Unravel Web UI configuration wizard to choose additional configuration options. For instructions on configuring advanced options, see the User Guide.

Attachments:

- logo_small.png (image/png)
- image2017-2-26_0-20-12.png (image/png)

Part 2: Enable Additional Data Collection / Instrumentation for HDP

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Installation Guide for Hortonworks Data Platform (HDP)

Introduction

This guide describes how to install the Unravel Sensor for Hive Hook and Spark on HDP 2.2.x-2.5.x clusters using **Ambari Web UI** (AWU). Unravel Hive Hook is used to collect information about Hive queries in Hadoop. The information here applies to Hive versions 0.10.0 through 2.1.x and Unravel 4.0-4.1. Highlighted text indicates where you must substitute your particular values.

1. Start Unravel Server

Note: Unravel needs to be up for the next step to complete.

```
sudo /etc/init.d/unravel_all.sh start
```

2. Install Unravel Hive Hook and Spark Sensor Onto HDP Servers

Install Unravel Hive Hook and Spark Sensor onto HDP servers (JAR files) as follows.

```
# login as root to do below steps
# ensure wget is install and use below script to install sensors
# /usr/local/unravel/install_bin/unraveldata-clients/unravel_hdp_setup.sh
# from Unravel server (eg. edge node)
# run on each server that will use instrumentation:
yum install -y wget
cd /usr/local/unravel/install_bin/unraveldata-clients/
sudo ./unravel_hdp_setup.sh install -y --unravel-server UNRAVEL_HOST_IP:3000
 --spark-version SPARK_VERSION --hive-version HIVE_VERSION
```

Substitute valid values for:

- UNRAVEL_HOST_IP - fully qualified host name or IP address
- SPARK_VERSION - target Spark version
- HIVE_VERSION - target Hive version

Files will be installed under:

Hive hook jar is located in /usr/local/unravel_client/.

Spark jar is located in /usr/local/unravel-spark/jars/.

Once the files are installed under /usr/local/unravel_client/ & /usr/local/unravel-spark/ on edge host where Unravel rpm is installed, you can tar these two directories up and put on other hosts, if that is more convenient than running the script. In all cases, instrumented nodes must be able to open port 4043 of Unravel Server (host2 if multi-host Unravel install).

3. Add Unravel Hive Hook hive-site Settings to All of HDP's Servers in the Cluster Using AWU

IMPORTANT: Completion of this step will require a restart of all affected Hive services in Ambari UI. If the env steps below are not deployed or use incorrect paths, then Hive jobs could fail with ClassNotFoundException when the hive-site change takes effect after the Hive service is restarted.

- Add `AUX_CLASSPATH` to AWU's **hive-env** template:
  - In AWU, on the left-hand side, click **Hive**, next click **Configs**, go to the **Advanced** tab, and select/click **Advanced hive-env**.
  - Next, inside the **Advanced hive-env** template, towards the end of line add below:

    ```
    AUX_CLASSPATH=${AUX_CLASSPATH}:/usr/local/unravel_client/unravel-
    hive-1.2.0-hook.jar
    ```

    Hold off on restarting any services until the next step.
- Add `HADOOP_CLASSPATH` to AWU's **hadoop-env** template:
  - In AWU, on the left-hand side, click **HDFS**, next click **Configs**, go to the **Advanced** tab, and select/click **Advanced hadoop-env**.

- Next, inside the **hadoop-env** template, towards the end of line add below:

```
HADOOP_CLASSPATH=${HADOOP_CLASSPATH}:/usr/local/unravel_client/unravel-
hive-1.2.0-hook.jar
```

You can optionally restart the services in Ambari, as prompted, at this point in order to verify that the environment change is done correctly. After the restart, look at hadoop-env.sh and hive-env.sh on an edge node and check the path to the jar file. Hint: find the files with `sudo find /etc -name '*env.sh' -newerct '1 hour ago'`

- Add the contents of `/usr/local/unravel/hive-hook/hive-site.xml.snip` from Unravel Gateway server (from host2 if multi-host Unravel so you get the proper UNRAVEL_HOST_IP) into AWU's **Custom hive-site** under **Hive**:
  - In AWU, on the left-hand side, click **Hive** | **Configs**. Select the **Advanced** tab, and then select **Custom hive-site** and **Add Property** below and use  **Bulk property add mode.**

```
hive.exec.driver.run.hooks=com.unraveldata.dataflow.hive.hook.HiveDriverHook
com.unraveldata.hive.hdfs.dir=/user/unravel/HOOK_RESULT_DIR
com.unraveldata.hive.hook.tcp=true
com.unraveldata.host=UNRAVEL_HOST_IP

-- Find below properties as it may already exists, concatenate it with a
 comma & no spaces --
hive.exec.pre.hooks=com.unraveldata.dataflow.hive.hook.HivePreHook
hive.exec.post.hooks=com.unraveldata.dataflow.hive.hook.HivePostHook
hive.exec.failure.hooks=com.unraveldata.dataflow.hive.hook.HiveFailHook
```

You should restart the services in Ambari now, as prompted, to test whether Hive queries can succeed after the above configuration change. If you get ClassNotFoundException during a query, then make corrections or revert.

4. If Possible, Ensure that **hive.execution.engine** is Set to `MapReduce` in your Hive query

```
set hive.execution.engine=mr;
```

5. **Optionally for Spark on YARN**, Enable Unravel Spark Instrumentation on All of HDP's Servers in the Cluster

IMPORTANT: Completion of this step will require a restart of all affected Spark services in Ambari UI.

- Add Spark properties into AWU's **Custom spark-defaults**:
  - In AWU, on the left-hand side, click **Spark** | **Configs** | **Custom spark-defaults**.
- Inside **Custom spark-defaults**, click **Add Property** for Spark as follows and use  **Bulk property add mode**:

```
spark.unravel.server.hostport=UNRAVEL_HOST_IP:4043

spark.driver.extraJavaOptions=-javaagent:/usr/local/unravel-
spark/jars/btrace-agent.jar=bootClassPath=/usr/local/unravel-
spark/jars/unravel-boot.jar,systemClassPath=/usr/local/
unravel-spark/jars/unravel-sys.jar,scriptOutputFile=/dev/
null,script=DriverProbe.class:SQLProbe.class

spark.executor.extraJavaOptions=-javaagent:/usr/local/unravel-spark/
jars/btrace-agent.jar=bootClassPath=/usr/local/unravel-spark/jars/
unravel-boot.jar,systemClassPath=/usr/local/unravel-spark/jars/unravel-
sys.jar,scriptOutputFile=/dev/null,script=ExecutorProbe.class
```

```
spark.eventLog.enabled=true
```

Notice that in this code block, the blank lines separate single full lines of text that are wrapped due to length. Also, ensure you replace "UNRAVEL_HOST_IP" with your Unravel server's IP address.

6. **Optionally for MapReduce2** (MR) JVM Sensor Cluster-Wide

IMPORTANT: Completion of this step will require a restart of all affected HDFS, MAPREDUCE2, YARN and HIVE services in Ambari UI.

- In AWU, on the left-hand side, click **MapReduce2**, next click **Configs**, go to the **Advanced** tab, and select/click **Advanced mapred-site**
- Search for **MR AppMaster Java Heap Size** property and concatenate by copying and paste below code block property for "yarn.app.mapreduce.am.command-opts" MR JVM sensor as follows:

(Note: please leave a white space in-between the current and the following new property)

- On the top notification banner, click Save

```
  -javaagent:/usr/local/unravel-spark/jars/btrace-
agent.jar=systemClassPath=/usr/local/unravel-spark/jars/
unravel-mr-sys.jar,bootClassPath=/usr/local/unravel-
spark/jars/unravel-mr-boot.jar,scriptOutputFile=/dev/
null -Dunravel.server.hostport=UNRAVEL_HOST_IP:4043 -
Dcom.sun.btrace.FileClient.flush=-1
```

Notice that in this code block, the blank lines separate single full lines of text that are wrapped due to length. Also, ensure you replace " UNRAVEL_HOST_IP" with your Unravel server's IP address.

- In AWU, on the left-hand side, click **MapReduce2**, next click **Configs**, go to the **Advanced** tab, and select/click **Custom mapred-site:**
- Inside **Custom mapred-site**, click **Add Property** for MR JVM sensor as follows and use  **Bulk property add mode**:
- On the top notification banner, click Save

```
mapreduce.task.profile=true
mapreduce.task.profile.maps=0-5
mapreduce.task.profile.reduces=0-5
mapreduce.task.profile.params=-javaagent:/usr/local/unravel-
spark/jars/btrace-agent.jar=systemClassPath=/usr/local/
unravel-spark/jars/unravel-mr-sys.jar,bootClassPath=/usr/
local/unravel-spark/jars/unravel-mr-boot.jar,scriptOutputFile=/
dev/null -Dunravel.server.hostport=UNRAVEL_HOST_IP:4043 -
Dcom.sun.btrace.FileClient.flush=-1
```

Notice that in this code block, the blank lines separate single full lines of text that are wrapped due to length. Also, ensure you replace " UNRAVEL_HOST_IP" with your Unravel server's IP address.

- Following instructions are for Unravel rpm 4.0.x and 4.1.x

  Please propagate Unravel MR jar onto all the servers in the cluster as follows**:**

  Ensure you have already installed unzip, curl, and Unravel port# 3000 must be open

  - ssh to the Unravel gateway host server and use guided steps to unzip the Unravel MR jars.
    - With root or sudo access, change directory to /usr/local/unravel-spark and run the below curl get command:

```
cd /usr/local/unravel-spark
curl http://localhost:3000/hh/unravel-sensor-for-mapreduce-bin.zip -o
 unravel-sensor-for-mapreduce-bin.zip
unzip -d jars unravel-sensor-for-mapreduce-bin.zip
```

- Now, tar up the /usr/local/unravel-spark directory, and propagate to all the servers in the HDP cluster

```
cd /usr/local/
tar -cvf unravel-spark.tar  ./unravel-spark
---> copy the unravel-spark.tar file to all your servers, and
 untar to /usr/local directory because when you untar unravel-spark
 directory will appear
```

Attachments:

worddav95f1e8fd13958fae391afc8e6f9ad03d.png (image/png)

worddav0175c0ac4efc905578b516dee96677a5.png (image/png)

logo_small.png (image/png)

Part 3: Additional Topics for HDP

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Installation Guide for Hortonworks Data Platform (HDP)

- Creating Active Directory Kerberos Principals and Keytabs for Unravel
- Sensors
  - Installing Unravel Sensor for Individual Applications Submitted Through spark-shell
  - Installing Unravel Sensor for Individual Applications Submitted Through spark-submit
  - Installing Unravel Sensor for Individual Hive Queries
- Workflows
  - Monitoring Airflow Workflows
  - Monitoring Oozie Workflows
  - Tagging Workflows
- Custom Configurations
  - Adding More Admins to Unravel Web UI
  - Configuring Multiple Hosts for Unravel Server
  - Creating Multiple Workers for High Volume Data
  - Defining a Custom TC Port
  - Integrating LDAP Authentication for Unravel Web UI
  - Setting Retention Time in Unravel Server
  - Setting Up Email for Auto Actions and Collaboration
- Connecting to a Hive Metastore
- Creating an AWS RDS CloudWatch Alarm for FreeStorageSpace
- Creating Application Tags
- Troubleshooting
  - Running Verification Scripts and Benchmarks
  - Sending Diagnostics to Unravel Support
- Uninstalling Unravel Server
- Upgrading Unravel Server

Chapter 3

User Guide

Topics:

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1

The key value proposition of Unravel is to help you analyze, optimize, and troubleshoot big data applications and operations.

Getting Started

The Operations Tab

The Applications Tab

The Data Tab

Use Cases

Optimizing the Performance of Spark Applications

Detecting Resource Contention in the Cluster

Identifying Rogue Applications

Notifications

Setting Up Auto Actions (Alerts)

Attachments:

logo_small.png

image2017-2-26_0-20-12.png

admin_manage_configuration.png

gui_admin_manage_configuration.png

gui_email.png

gui_applications.png

gui_global_search.png

gui_spark_app_mgr.png

gui_spark_app_mgr2.png

gui_spark_app_mgr3.png

gui_hive_app_mgr1.png

gui_hive_app_mgr2.png

gui_mapreduce_app_mgr1.png

gui_workflow_mgr1.png

gui_workflow_mgr2.png

gui_spark_app_mgr3.png

gui_workflow_mgr3.png

gui_events_hive.png

gui_events_spark.png

gui_events_workflow.png

gui_error_view.png

gui_ops_dashboard.png

gui_ops_cluster_metrics.png

gui_ops_chargeback.png

gui_data_table.png

gui_data_table_summary.png

gui_data_table_details.png

gui_data_table_details2.png

gui_data_table_details3.png

gui_data_table_rules1.png

gui_data_table_rules2.png

gui_auto_actions1.png

gui_auto_actions2.png

gui_report_ops.png

gui_report_users.png

Unravel-Product-Use-Case-Demo-1.mp4

Unravel-Product-Use-Case-Demo2.mp4

Unravel-Product-Use-Case-Demo3.mp4

Unravel-Product-Use-Case-Demo4.mp4

spark-insights3.png

spark-insights3.png

caching-opportunity.png

hive-recommendation.png

gui_hive_app_mgr1.png

gui_hive_app_mgr1_events.png

gui_applications.png

gui_auto_actions1.png

gui_auto_actions2.png

gui_auto_actions_select.png

why-unravel-sketch-v3-2-1200x687-analyze.png

why-unravel-sketch-v3-2-1200x687-optimize.png

why-unravel-sketch-v3-2-1200x687-troubleshoot.png

why-unravel-sketch-v3-2-1200x687-analyze.png

why-unravel-sketch-v3-2-1200x687-troubleshoot.png

why-unravel-sketch-v3-2-1200x687-optimize.png

why-unravel-sketch-v3-2-1200x687-analyze.png

why-unravel-sketch-v3-2-1200x687-troubleshoot.png

why-unravel-sketch-v3-2-1200x687-optimize.png

why-unravel-sketch-v3-2-1200x687-analyze.png

why-unravel-sketch-v3-2-1200x687-troubleshoot.png

why-unravel-sketch-v3-2-1200x687-optimize.png (image/png)

Getting Started

Use Case Videos

Use Case #1: How to Search for Applications and Optimize/Tune a Hive Application

Use Case #2: How to "Root Cause" Issues with a Workflow that Missed Its SLA

Use Case #3: How to Debug Failed Applications

Use Case #4: How to Review Spark Applications and Identify Areas for Performance Improvements

Running the Configuration Wizard

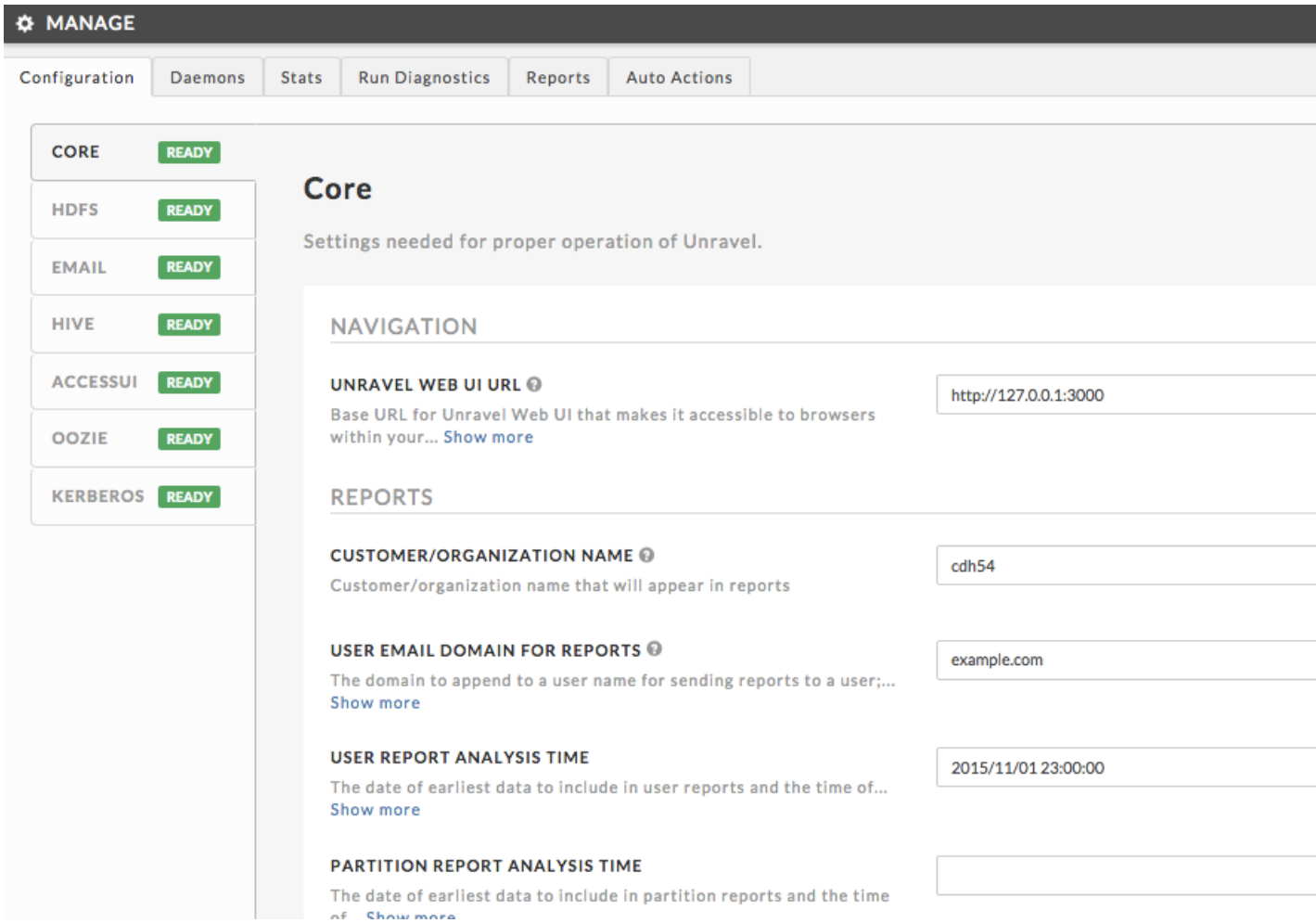
After you install the Unravel Server RPM, you can run Unravel Web UI's configuration wizard to:

- Set up access to various Big Data components (HDFS, Hive, Oozie, and so on).
- Create users
- Set up email/SMTP, LDAP, Kerberos

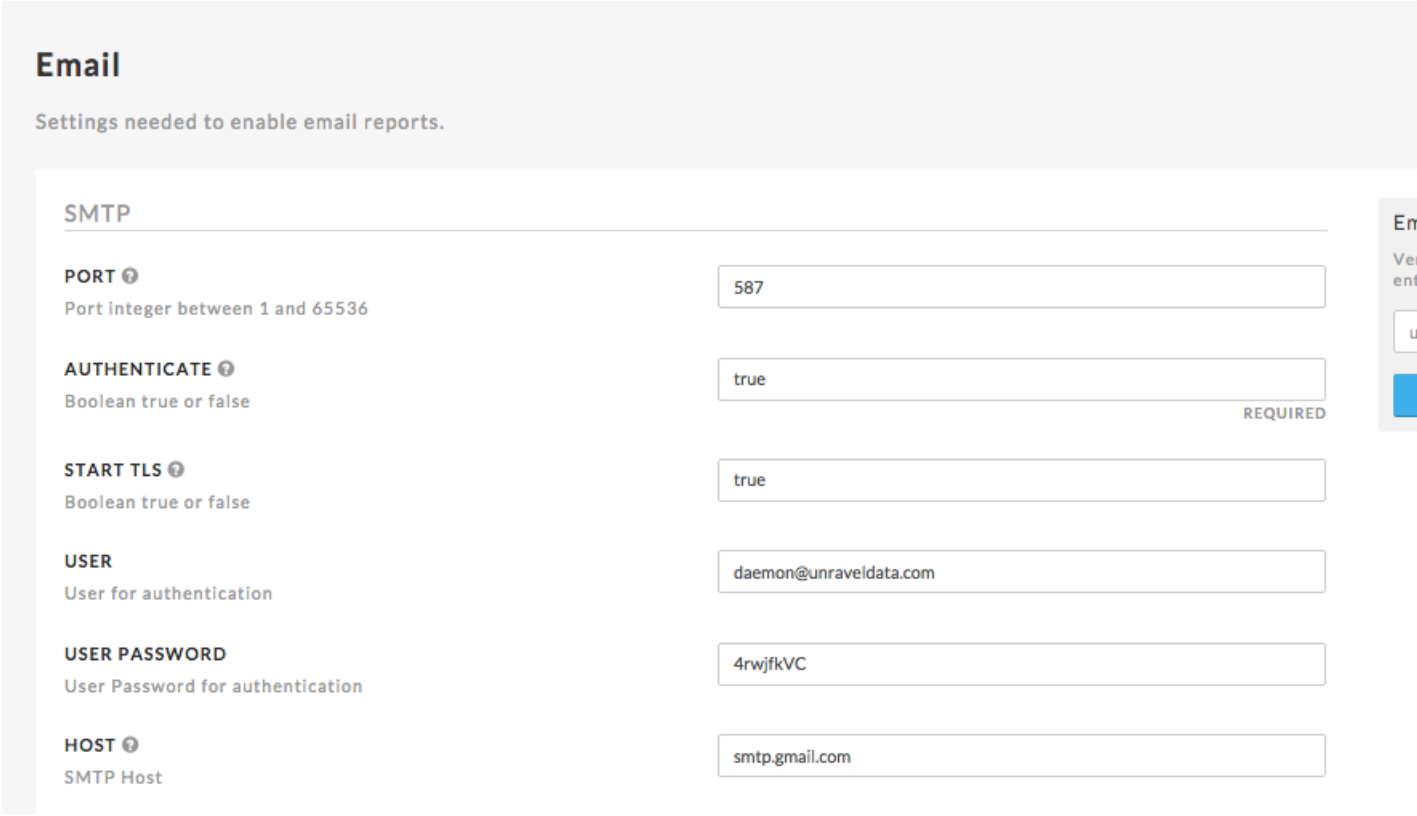
The configuration wizard informs you of errors and continuously checks for failed and incorrect settings. To start the configuration wizard, click **Admin** | **Manage** | **Configuration**.

The Unravel Web UI configuration wizard is available only for the `admin` user.

Setting Up Access to Big Data Components



Creating Users and Setting Up Email/SMTP, LDAP, Kerberos



Attachments:
gui_auto_actions_select.png ▪ gui_auto_actions2.png ▪ gui_auto_actions1.png ▪ gui_applications.png ▪ gui_hive_app_mgr1_events.png ▪ gui_hive_app_mgr1.png ▪ hive-recommendation.png ▪ caching-opportunity.png ▪ spark-insights3.png ▪ Unravel-Product-Use-Case-Demo4.mp4 ▪ Unravel-Product-Use-Case-Demo3.mp4 ▪ Unravel-Product-Use-Case-Demo2.mp4 ▪ Unravel-Product-Use-Case-Demo-1.mp4 ▪ gui_report_users.png ▪ gui_report_ops.png ▪ gui_data_table_rules2.png ▪ gui_data_table_rules1.png ▪ gui_data_table_details3.png ▪ gui_data_table_details2.png ▪ gui_data_table_details.png ▪ gui_data_table_summary.png ▪ gui_data_table.png ▪ gui_ops_chargeback.png ▪ gui_ops_cluster_metrics.png ▪ gui_ops_dashboard.png ▪ gui_error_view.png ▪ gui_events_workflow.png ▪ gui_events_spark.png ▪ gui_events_hive.png ▪ gui_workflow_mgr3.png ▪ gui_spark_app_mgr3.png ▪ gui_workflow_mgr2.png ▪ gui_workflow_mgr1.png ▪ gui_mapreduce_app_mgr1.png ▪ gui_hive_app_mgr2.png ▪ gui_spark_app_mgr2.png ▪ gui_spark_app_mgr.png ▪ gui_global_search.png ▪ gui_email.png ▪ gui_admin_manage_configuration.png ▪ admin_manage_configuration.png ▪ image2017-2-26_0-20-12.png ▪ logo_small.png

The Operations Tab

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. User Guide

Table of Contents

- Cluster Summary
- Cluster Compare

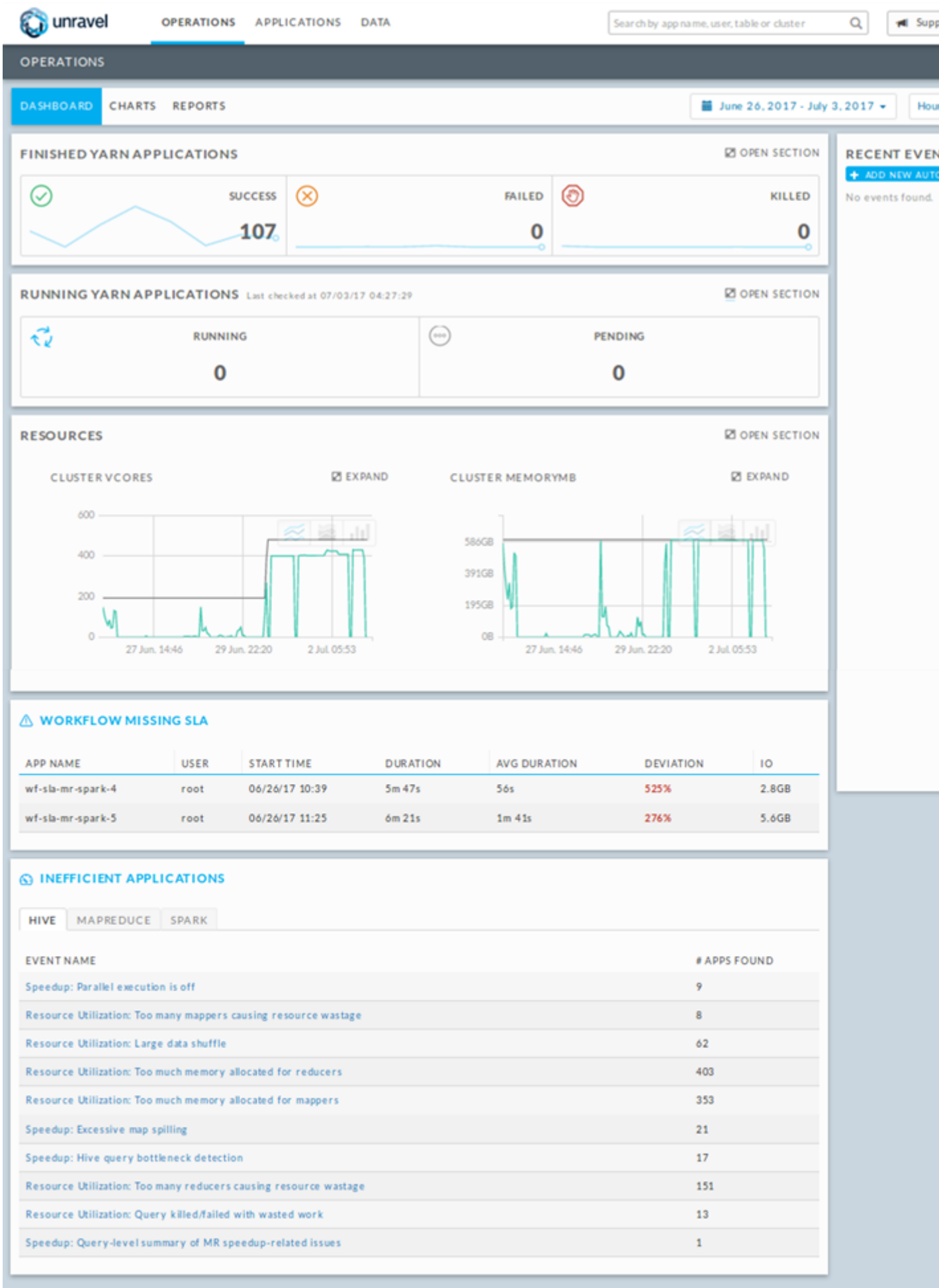
Dashboard

To view the Dashboard, click **Operations** | **Dashboard**.

The Dashboard provides a good overview of all activities in the cluster. It contains tiles that display cluster KPI time series, application summaries, and highlights of inefficient applications and workflows missing SLAs.

To see details within a specific tile, click that tile.

To configure the Dashboard for a specific time range or cluster, select options from the pull-down menus in the banner.


unravel
OPERATIONS
APPLICATIONS
DATA
Search by app name, user, table or cluster
OPERATIONS
DASHBOARD
CHARTS
REPORTS
June 26, 2017 - July 3, 2017
FINISHED YARN APPLICATIONS
OPEN SECTION
SUCCESS
107
FAILED
0
KILLED
0
RECENT EVEN
ADD NEW AUTO
No events found.
RUNNING YARN APPLICATIONS
Last checked at 07/03/17 04:27:29
OPEN SECTION
RUNNING
0
PENDING
0
RESOURCES
OPEN SECTION
CLUSTER VCORES
EXPAND
600
400
200
0
27 Jun, 14:46
29 Jun, 22:20
2 Jul, 05:53
CLUSTER MEMORYMB
EXPAND
586GB
391GB
195GB
0B
27 Jun, 14:46
29 Jun, 22:20
2 Jul, 05:53
WORKFLOW MISSING SLA
APP NAME
USER
START TIME
DURATION
AVG DURATION
DEVIATION
IO
wf-sla-mr-spark-4
root
06/26/17 10:39
5m 47s
56s
525%
2.8GB
wf-sla-mr-spark-5
root
06/26/17 11:25
6m 21s
1m 41s
276%
5.6GB
INEFFICIENT APPLICATIONS
HIVE
MAPREDUCE
SPARK
EVENT NAME
APPS FOUND
Speedup: Parallel execution is off
9
Resource Utilization: Too many mappers causing resource wastage
8
Resource Utilization: Large data shuffle
62
Resource Utilization: Too much memory allocated for reducers
403
Resource Utilization: Too much memory allocated for mappers
353
Speedup: Excessive map spilling
21
Speedup: Hive query bottleneck detection
17
Resource Utilization: Too many reducers causing resource wastage
151
Resource Utilization: Query killed/failed with wasted work
13
Speedup: Query-level summary of MR speedup-related issues
1

Charts

To view charts, click **Operations** | **Charts**.

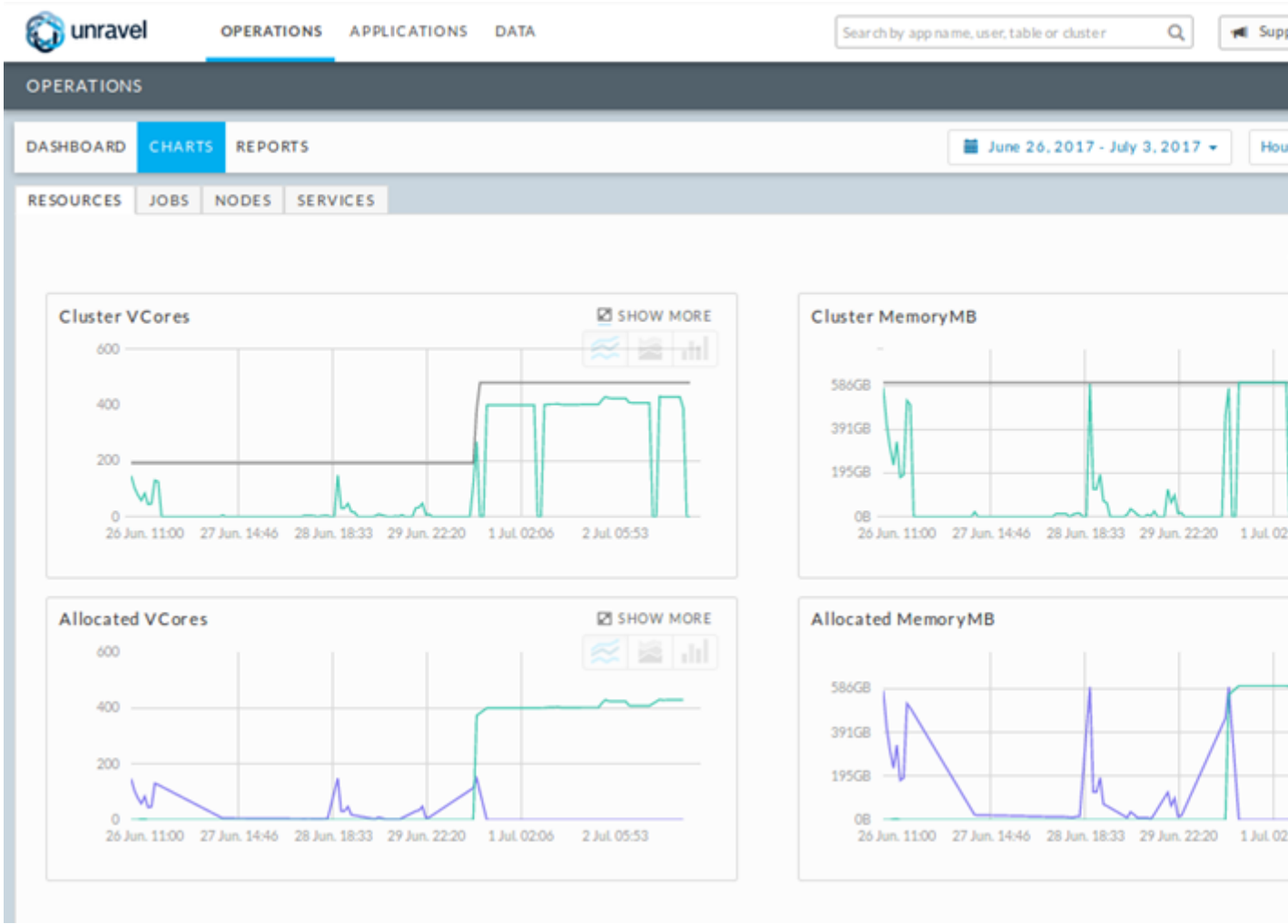
Charts address one of the big challenges in managing a multi-tenant Hadoop clusters: understanding how resources are being used or abused by the various applications running in the clusters. Unravel Web UI provides a unique forensic view into each cluster's key performance indicators (KPIs) over time and how they are related to the applications running in the cluster.

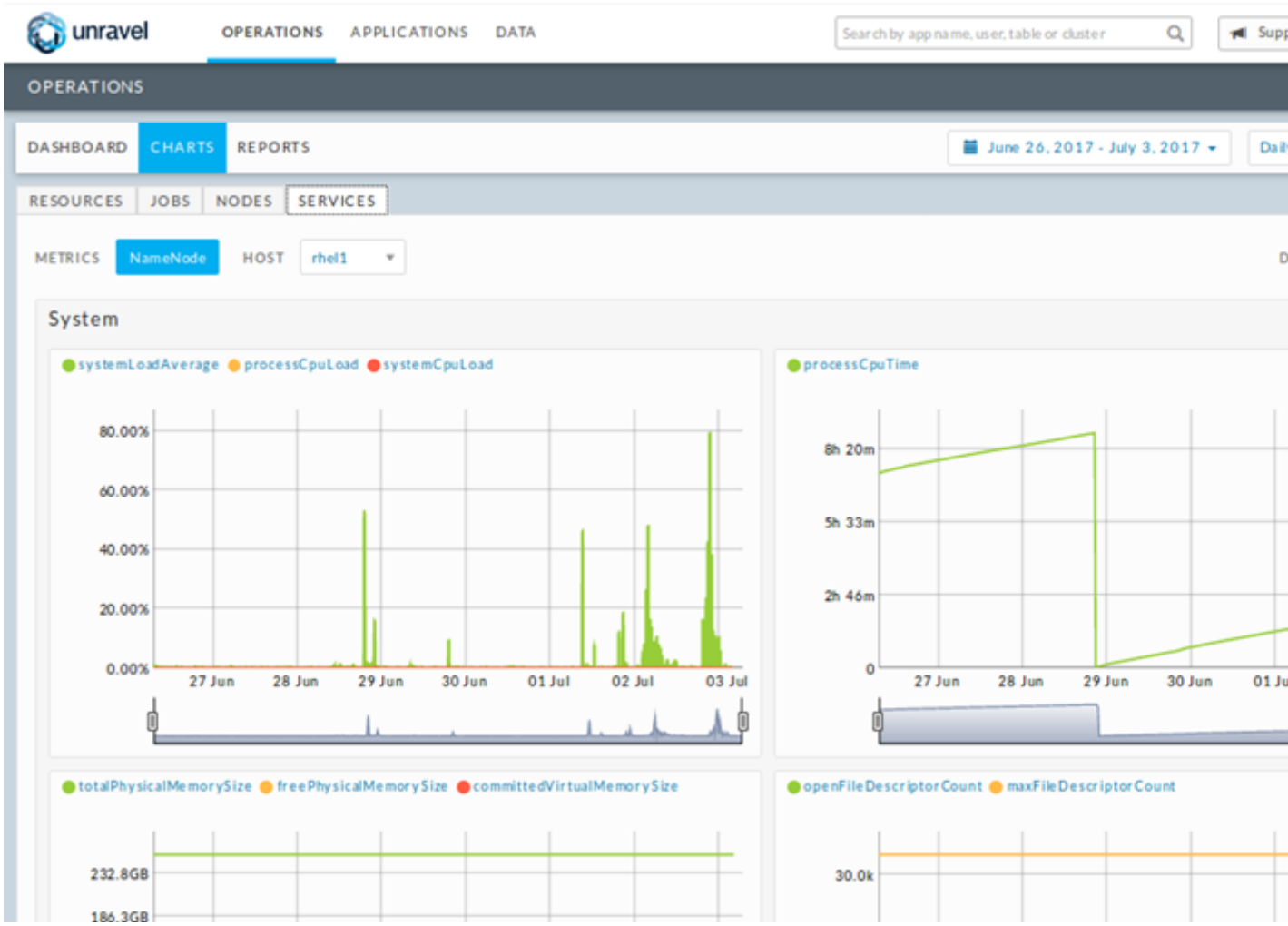
To see KPIs for resources, jobs, nodes, or services, select the appropriate secondary tab in the **Charts** ribbon.

To see a detailed view of any of KPI, click the tile for that KPI.

To configure **Charts** for a specific time range or cluster, select options from the pull-down menus in the banner.

For example, if you notice a sudden spike in the total vCores or memory usage of the cluster, using Unravel you can correctly pinpoint the applications that resulted in this spike. More importantly, you can then drill down into these specific applications to understand their behavior and apply any recommendations that Unravel suggests for optimizing them.





Reports

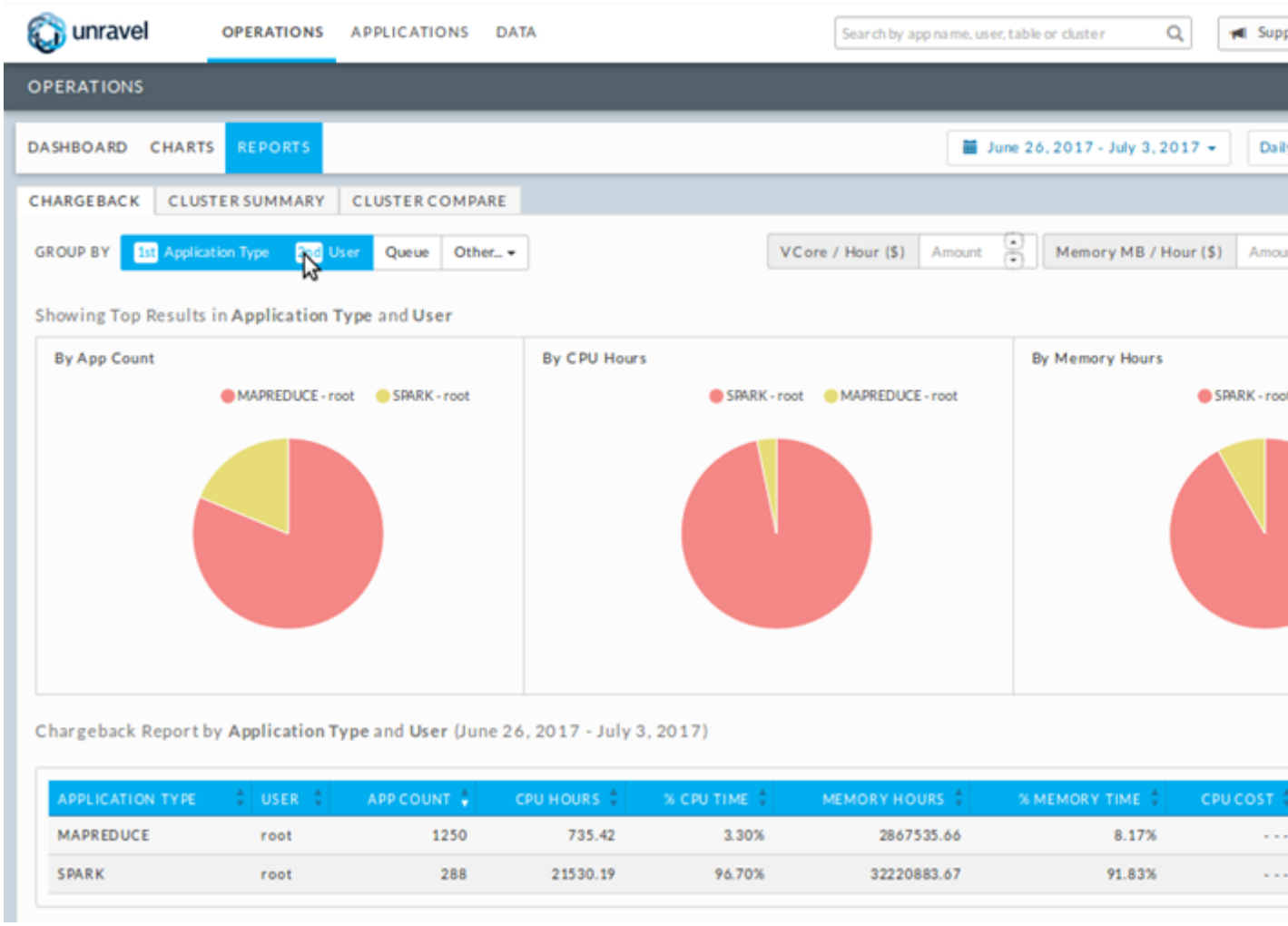
To view reports, click **Operations** | **Reports**.

Chargeback

Unravel provides an easy way to create chargeback reports for multi-tenant cluster usage. You can generate usage reports categorized by application type, queue, users, and so on.

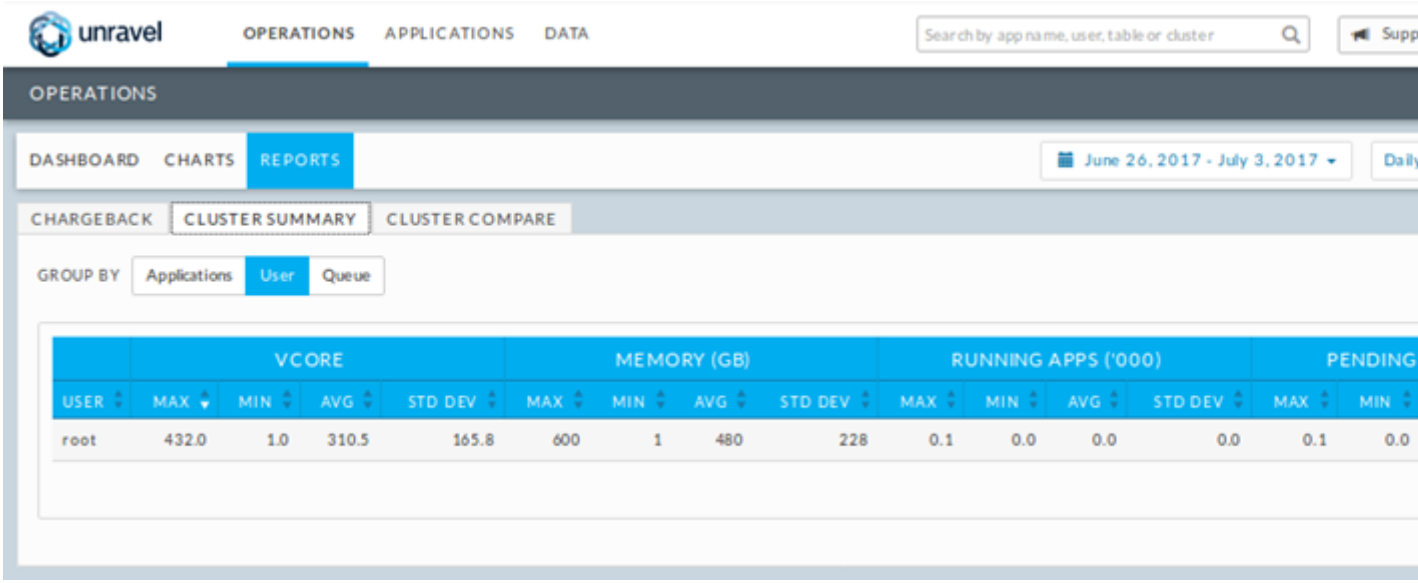
To configure **Chargeback** for a specific time range or cluster, select options from the pull-down menus in the banner.

For example, you can generate reports categorized first by application type and secondarily by users and queue; then you can specify base costs (VCore/Hour ($) and Memory MB/Hour ($)) to estimate actual costs across various dimensions.



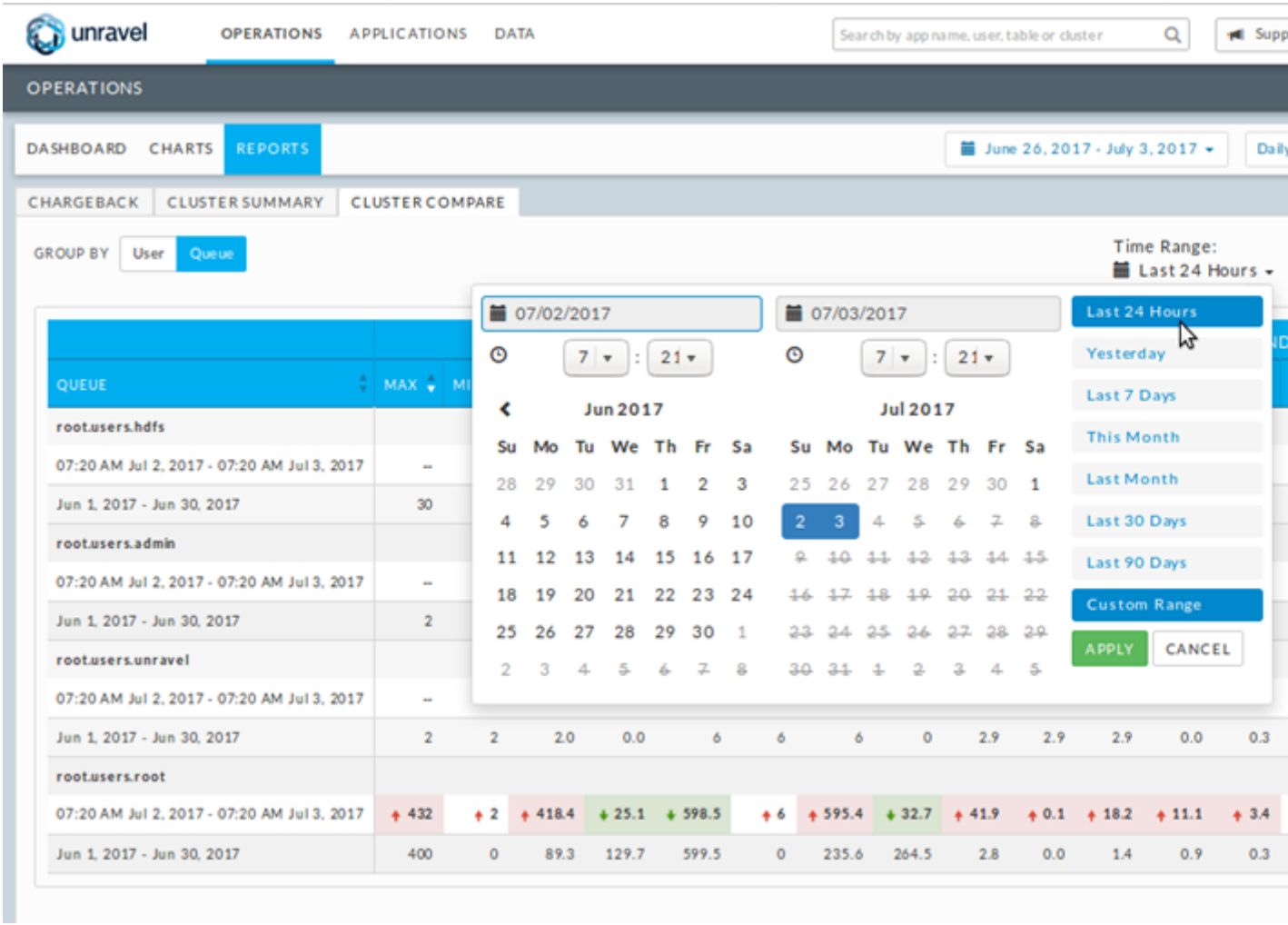
Cluster Summary

The **Cluster Summary**, a secondary tab of **Reports**, summarizes the cluster health. You can group the summary by application, user, or queue.



Cluster Compare

The **Cluster Compare**, a secondary tab of **Reports**, provides a visual assessment of cluster health. You can group the comparison by user or queue.



Attachments:

- gui_auto_actions_select.png (image/png)
- gui_auto_actions2.png (image/png)
- gui_auto_actions1.png (image/png)
- gui_applications.png (image/png)
- gui_hive_app_mgr1_events.png (image/png)
- gui_hive_app_mgr1.png (image/png)
- hive-recommendation.png (image/png)
- caching-opportunity.png (image/png)
- spark-insights3.png (image/png)
- Unravel-Product-Use-Case-Demo4.mp4 (video/mp4)
- Unravel-Product-Use-Case-Demo3.mp4 (video/mp4)
- Unravel-Product-Use-Case-Demo2.mp4 (video/mp4)
- Unravel-Product-Use-Case-Demo-1.mp4 (video/mp4)
- gui_report_users.png (image/png)
- gui_report_ops.png (image/png)
- gui_data_table_rules2.png (image/png)
- gui_data_table_rules1.png (image/png)
- gui_data_table_details3.png (image/png)
- gui_data_table_details2.png (image/png)
- gui_data_table_details.png (image/png)
- gui_data_table_summary.png (image/png)
- gui_data_table.png (image/png)
- gui_ops_chargeback.png (image/png)
- gui_ops_cluster_metrics.png (image/png)
- gui_ops_dashboard.png (image/png)
- gui_error_view.png (image/png)

gui_events_workflow.png (image/png)

gui_events_spark.png (image/png)

gui_events_hive.png (image/png)

gui_workflow_mgr3.png (image/png)

gui_spark_app_mgr3.png (image/png)

gui_workflow_mgr2.png (image/png)

gui_workflow_mgr1.png (image/png)

gui_mapreduce_app_mgr1.png (image/png)

gui_hive_app_mgr2.png (image/png)

gui_spark_app_mgr2.png (image/png)

gui_spark_app_mgr.png (image/png)

gui_global_search.png (image/png)

gui_email.png (image/png)

gui_admin_manage_configuration.png (image/png)

admin_manage_configuration.png (image/png)

image2017-2-26_0-20-12.png (image/png)

logo_small.png (image/png)

gui_ops_dashboard.png (image/png)

gui_ops_cluster_metrics.png (image/png)

gui014.png (image/png)

gui_ops_chargeback.png (image/png)

gui028.png (image/png)

gui037.png (image/png)

The Applications Tab

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. User Guide

Topics

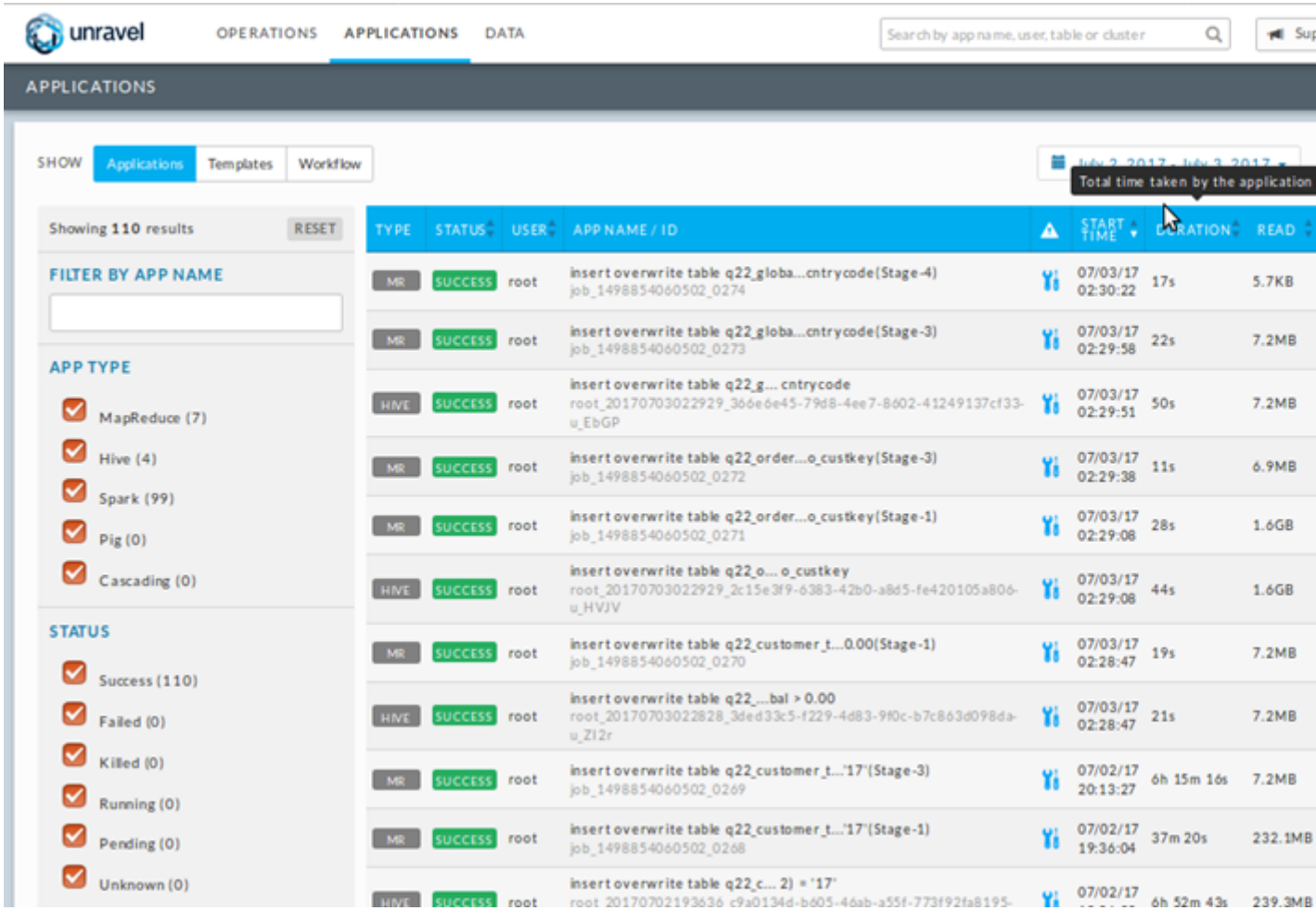
- Finding Applications
- Application-Specific Managers

The **Applications** tab is the main interface for finding your applications and analyzing their performance. Unravel supports both ad-hoc applications and repeated running workflows or data pipelines. Ad-hoc applications are generally Hive queries, MR jobs, Spark applications, Impala queries, and so on; these are generated by end user tools (such as BI tools like Tableau, Microstrategy etc.) or submitted via CLI. Repeated running workflows or data pipelines are created using cron or schedulers like Oozie, AirFlow or ETL tools like Informatica, Pentaho and others.

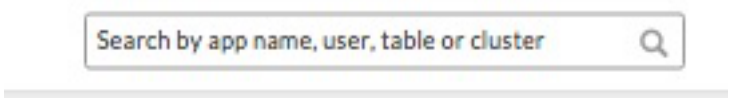
 The performance and reliability of an application depends on several factors such as quality of the code, types of joins used, configuration settings, data size, scheduler settings, contention with other applications, and so on. Therefore, it takes significant expertise and effort to get to the root cause of problems of an application. Unravel provides insights into an application. These insights are called *events*. For more information about events, see *Key Performance Indicators* within the appropriate application "manager" below.

Finding Applications

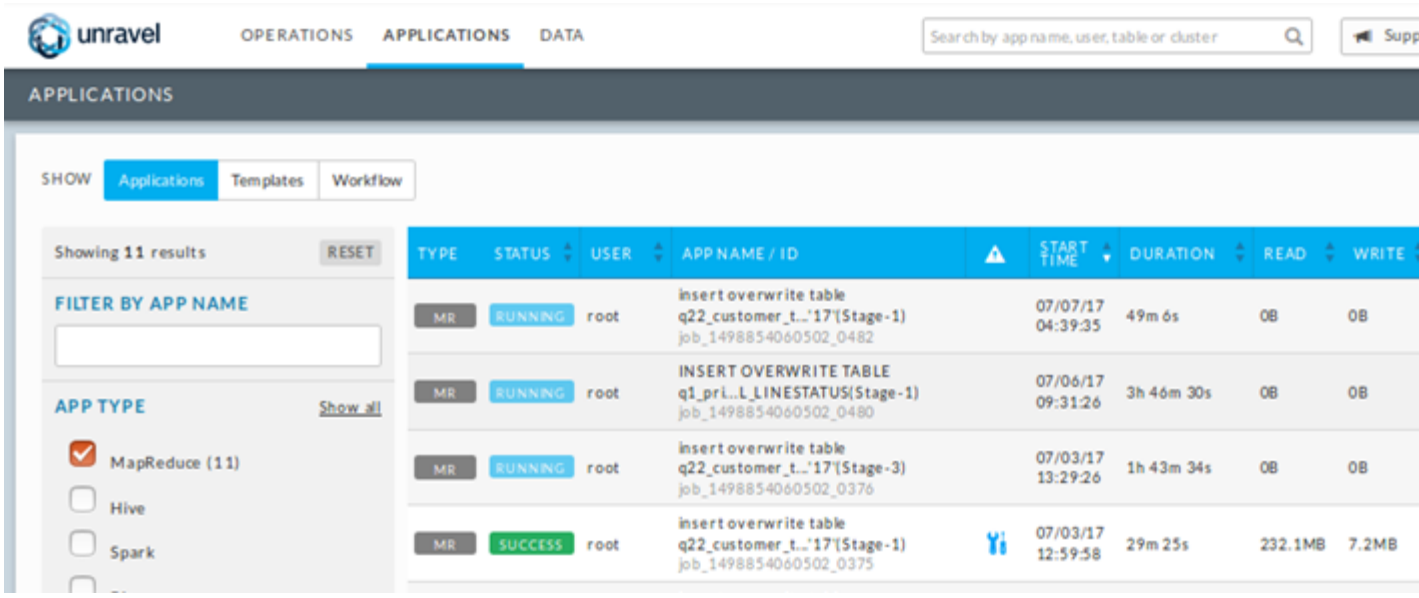
You can search on various dimensions (app type, status, queue, user, cluster, duration, and so on) to find your application(s). Use the date pull-down menu to limit results to a specific time range. Search results are ordered by the most recent start time. To reorder the results by another property, click the appropriate header in the results table.



You can also use the global search bar in the top banner to search by full job ID, user name, table name and cluster ID.



Search results list individual jobs, and job IDs. If the job is part of a Hive query, Pig script, and/or a workflow, then a link to that Hive query/Pig script/workflow page appears on the same line. To go to the job-specific page, click the job ID. To go to the application-specific manager for a query/script/job/workflow, click the icon under its **GoTo** column as highlighted in the image below.

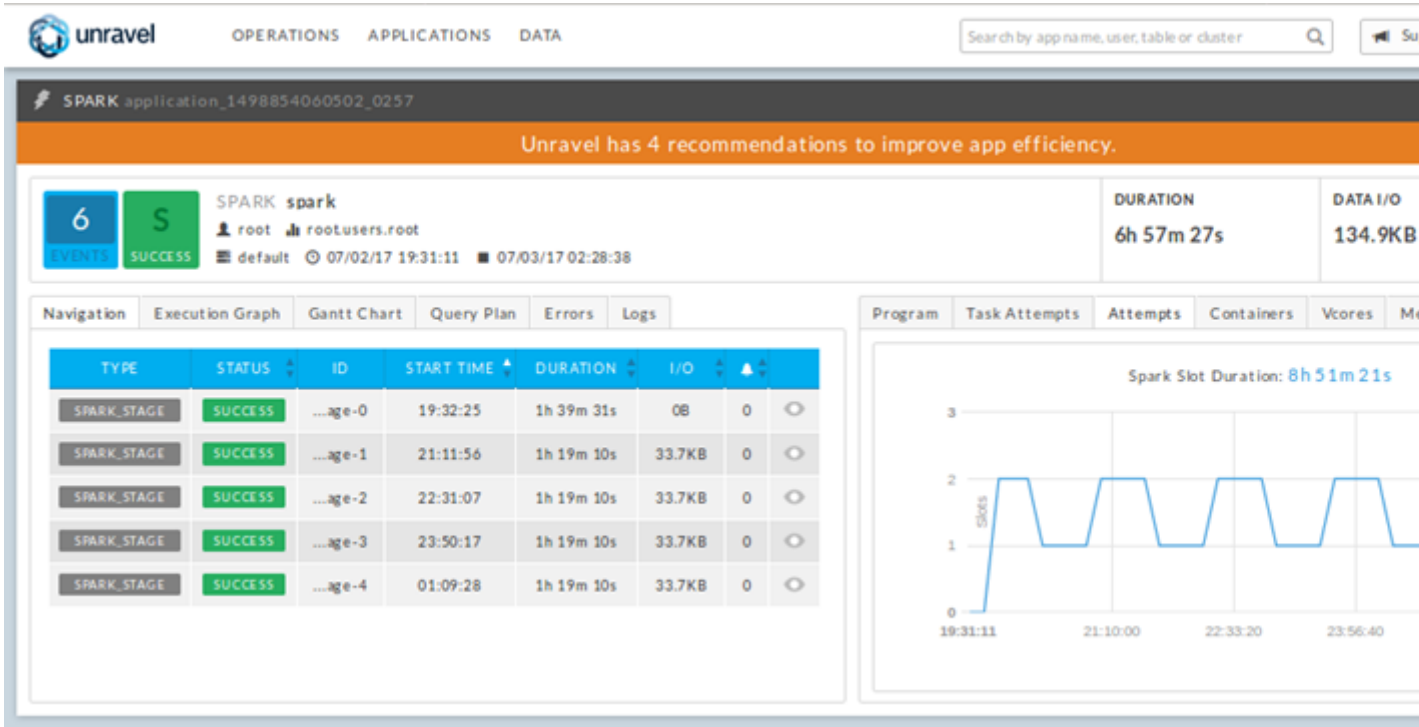


Application-Specific Managers

Spark Application Manager

The Spark Application Manager provides a detailed view into the behavior of Spark applications. You can use this view to:

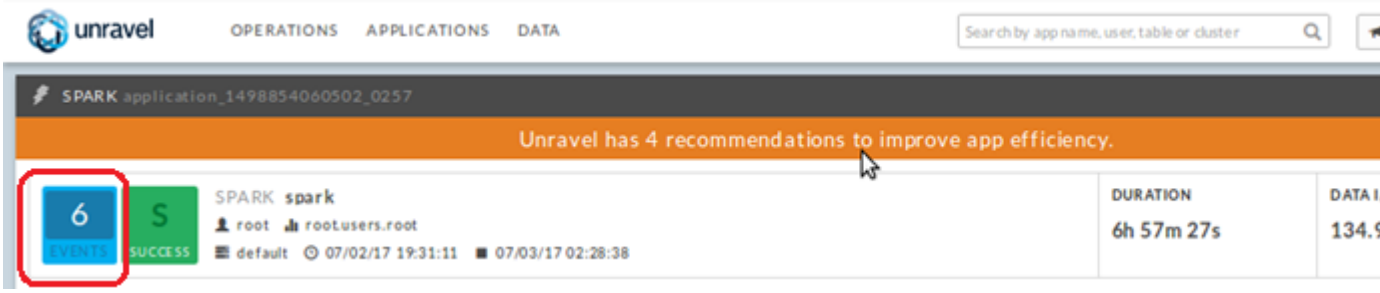
- Resolve inefficiencies, bottlenecks, and reasons for failure within applications
- Optimize resource allocation for Spark executors
- Detect and fix poor partitioning
- Detect and fix inefficient and failed Spark apps
- Tune JVM settings for Spark drivers/executors



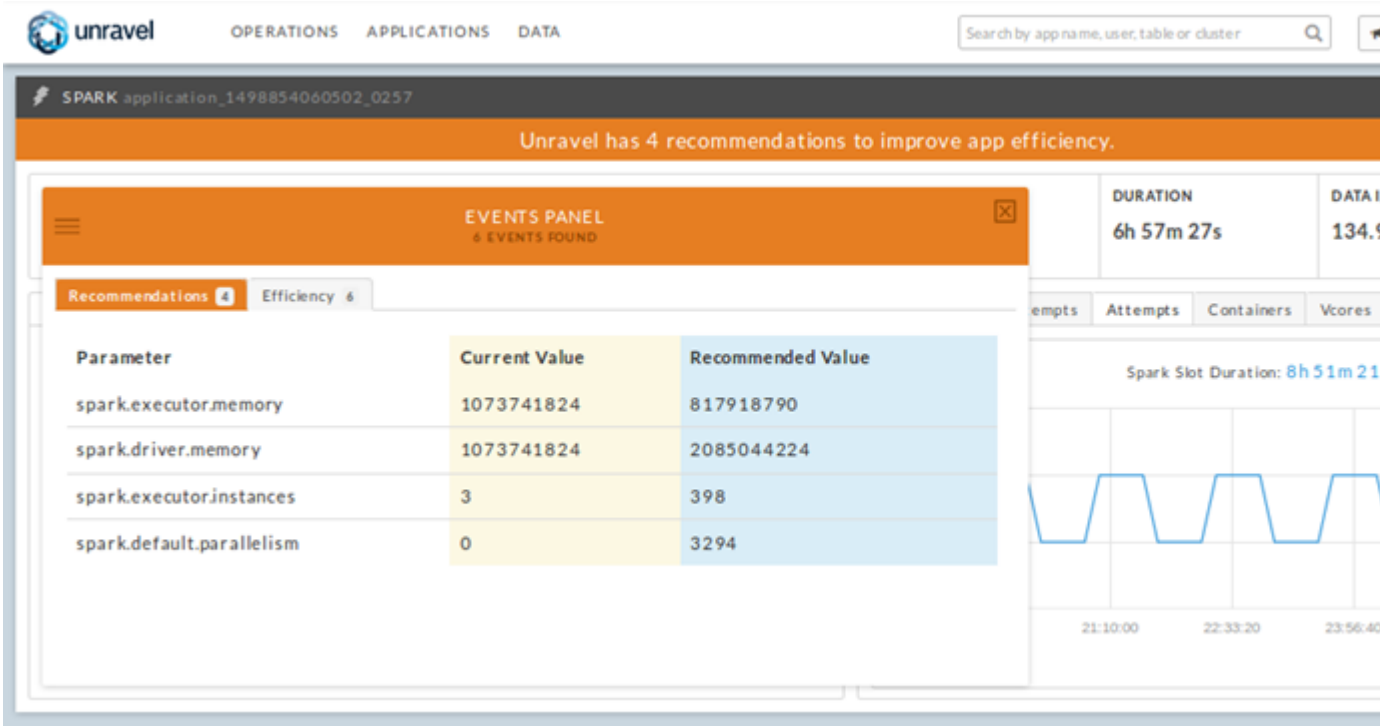
Key Performance Indicators (KPIs)

The key performance indicators (KPIs) at the top provide the most important information about the Spark application. The Spark Application Manager displays the following KPIs:

- **Events**: The number of Unravel recommendations or insights for this query. To see details, click the **Events** icon as highlighted in the image below.



The performance and reliability of your Spark application depends on several factors such as quality of the code, types of joins used, configuration settings, data size, scheduler settings, contention with other applications, and so on. Therefore, it takes significant expertise and effort to get to the root cause of problems in a Spark application. Unravel Events are designed to save you time and effort by automatically providing insights into the application. These events capture reasons for failed and killed queries as well as provide recommendations to improve application performance.

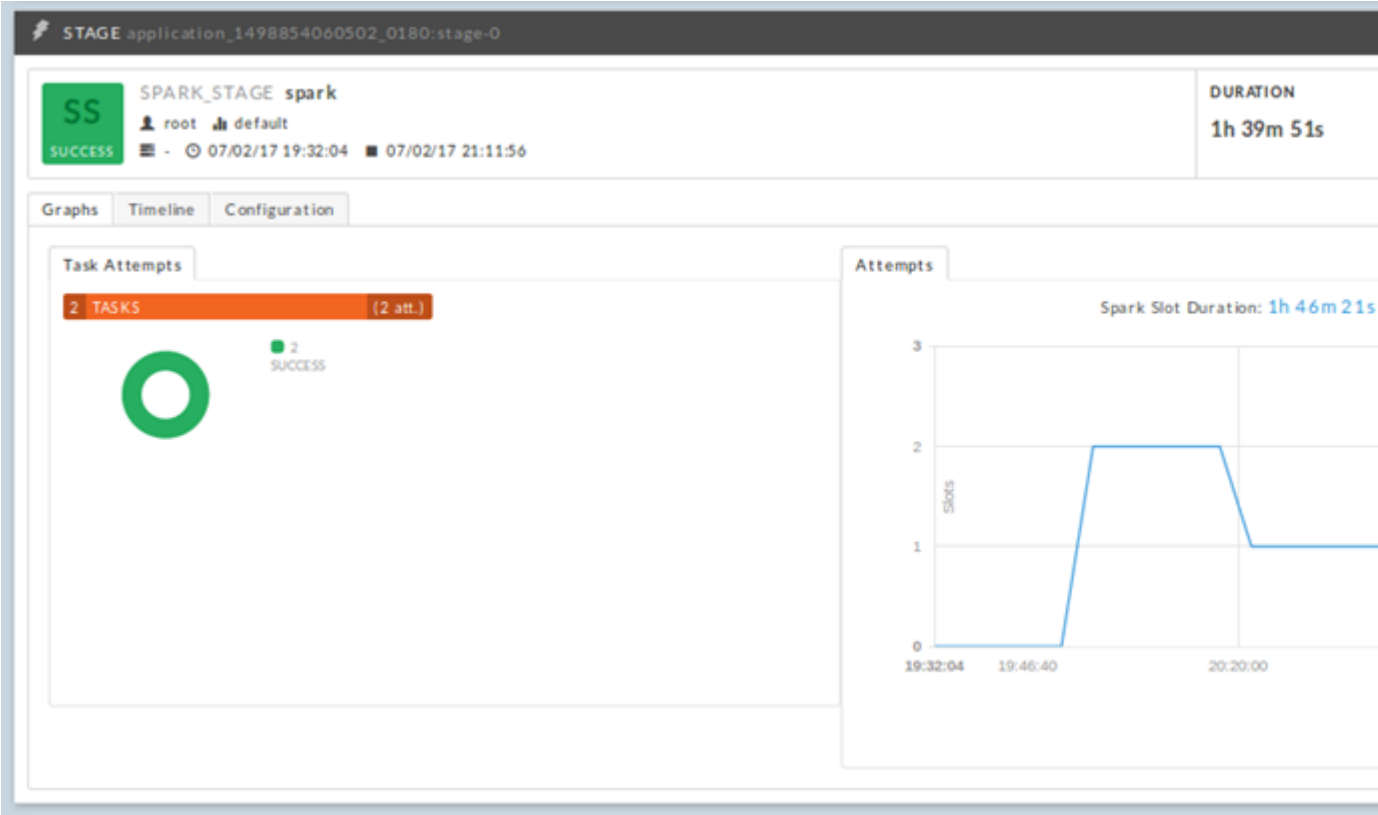


- **Duration**: Total time taken by the application to complete execution
- **Data I/O**: Total data read and written by the application
- **Number of Stages**: The number of stages that make up the Spark application and their status

Sub-Tabs

The Spark Application Manager contains multiple sub-tabs, each of which is described below.

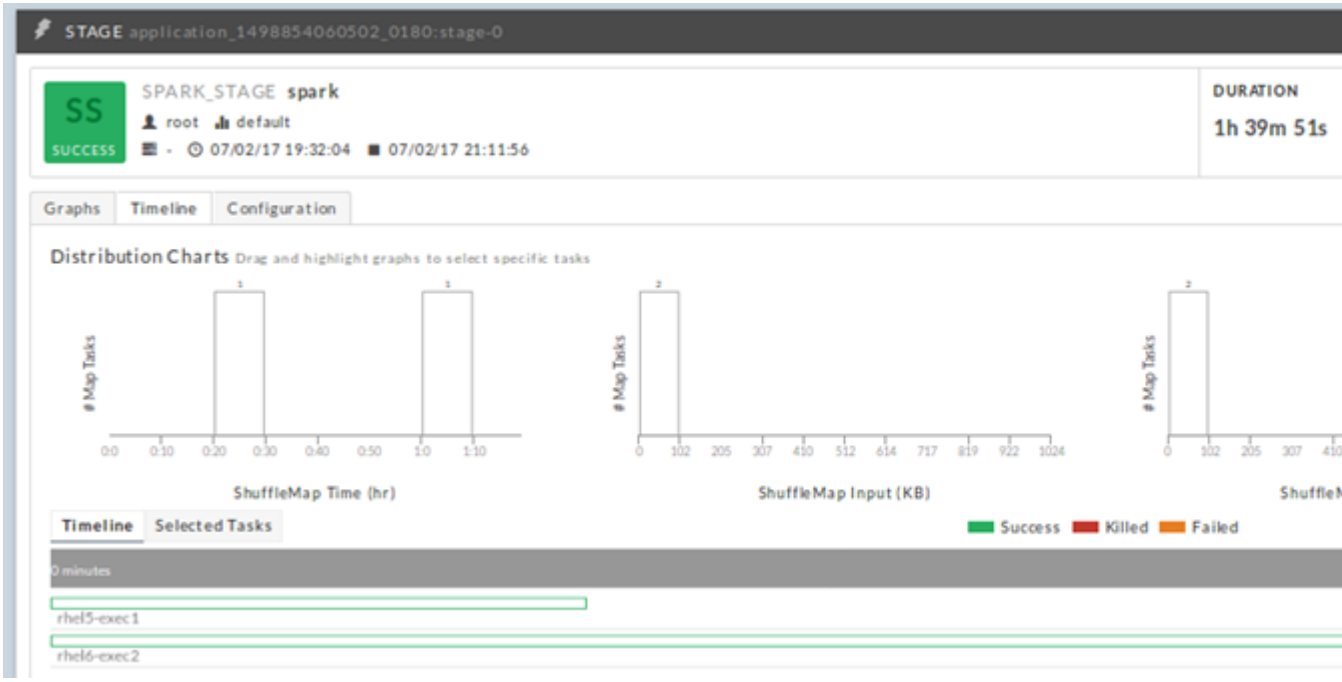
- **Navigation**: A table of the stages associated with this application. To see details about a stage, click its row in the table. This displays the Spark Stage view, as illustrated in the image below.

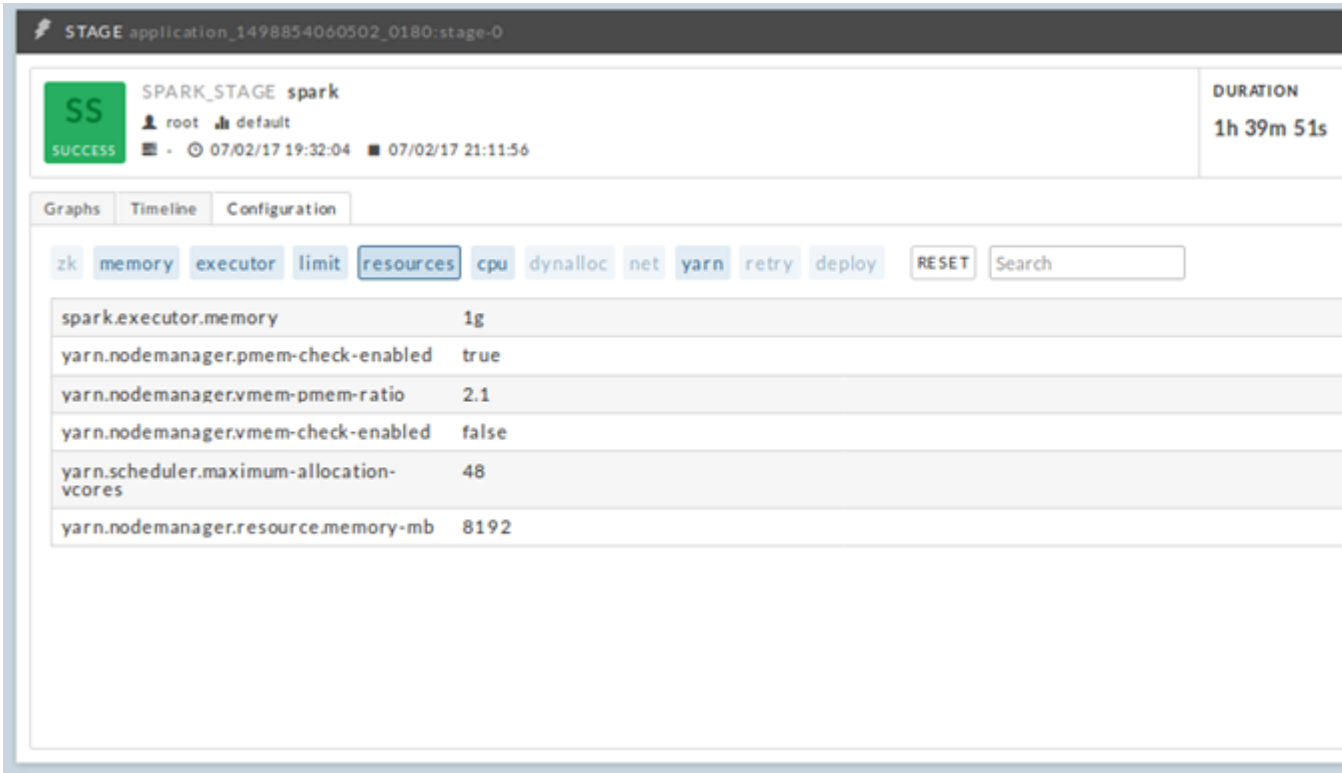


The Spark Stage view provides detailed information about each Spark stage. It includes:

- **Graphs**: General task/slot statistics of the stage
- **Timeline**: Timeline and histogram of task attempts, duration, and bytes shuffled/spilled. This information is very useful for identifying data skew.
- **Configuration**: Key/value pairs of configuration settings for the application

Click here to see sample screenshots...



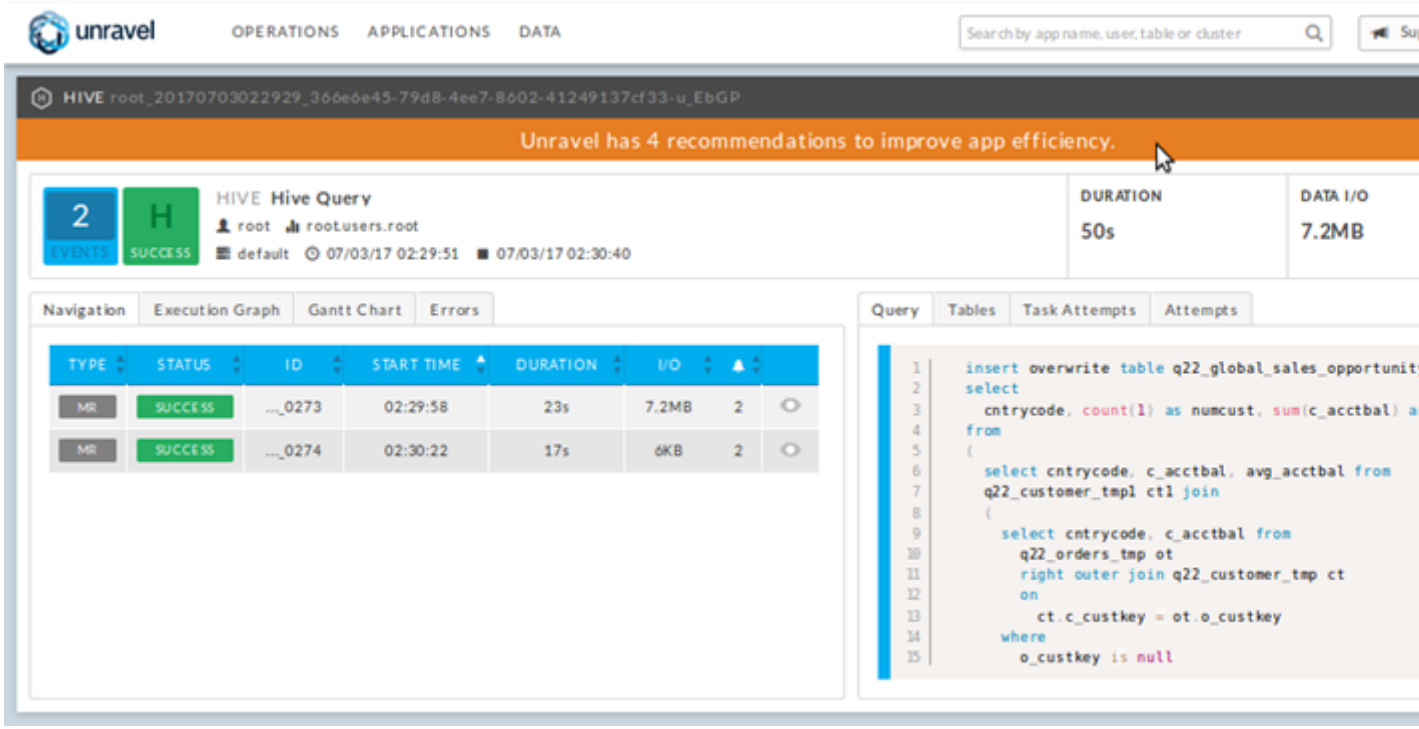


- **Execution Graph**: Displays both the stage view and RDD view associated with this application.
- **Gantt Chart**: A timeline of application stages
- **Query Plan**: The logical plan of the SparkSQL query
- **Errors**: Exceptions, errors, and warnings associated with this application
- **Logs**: Logs for the driver and executors of this application, and a skyline of task attempts within the stage
- **Program**: Source code of a general purpose Spark application, or the SQL query for a SparkSQL query
- **Task Attempts**: Statistics about the task attempts that are executed as part of the current application
- **Containers**: Utilization of slot containers over time

- **Vcores**: Utilization of slot vcores over time
- **Memory**: Utilization of slot memory over time
- **Resource**: Graphs of JVM-level metrics at the executor and driver level. To show the graph for a specific metric, select that metric from the **METRIC** pull-down menu and click **Get Data**. For a list of resource-level metrics, see Resource Metrics. These graphs are very useful for identifying critical resources that caused a performance degradation.

Hive Application Manager

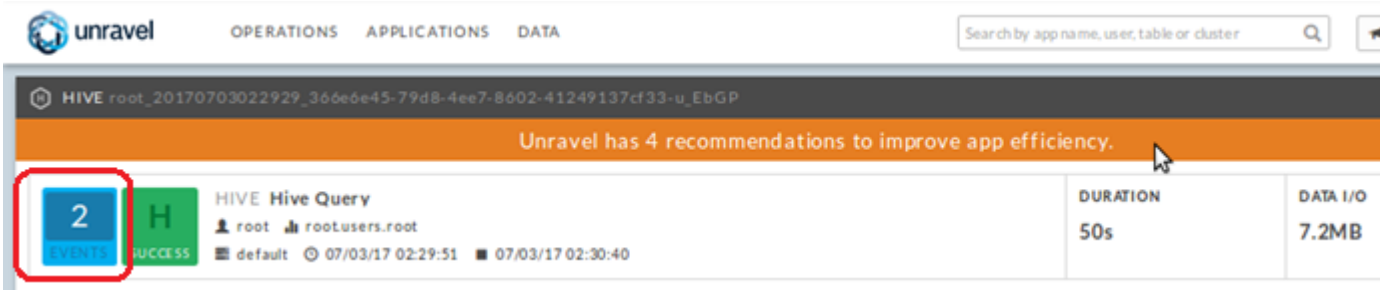
The Hive Application Manager provides a detailed view into the behavior of Hive queries. You can use this view to resolve inefficiencies, bottlenecks and reasons for failure within applications. Typical users are Hadoop DBAs or application owners (engineers, BI team, analysts). The Hive Application Manager uses Unravel's Intelligence Engine to automatically identify certain inefficiencies with the application and to provide recommendations on how to improve efficiency.



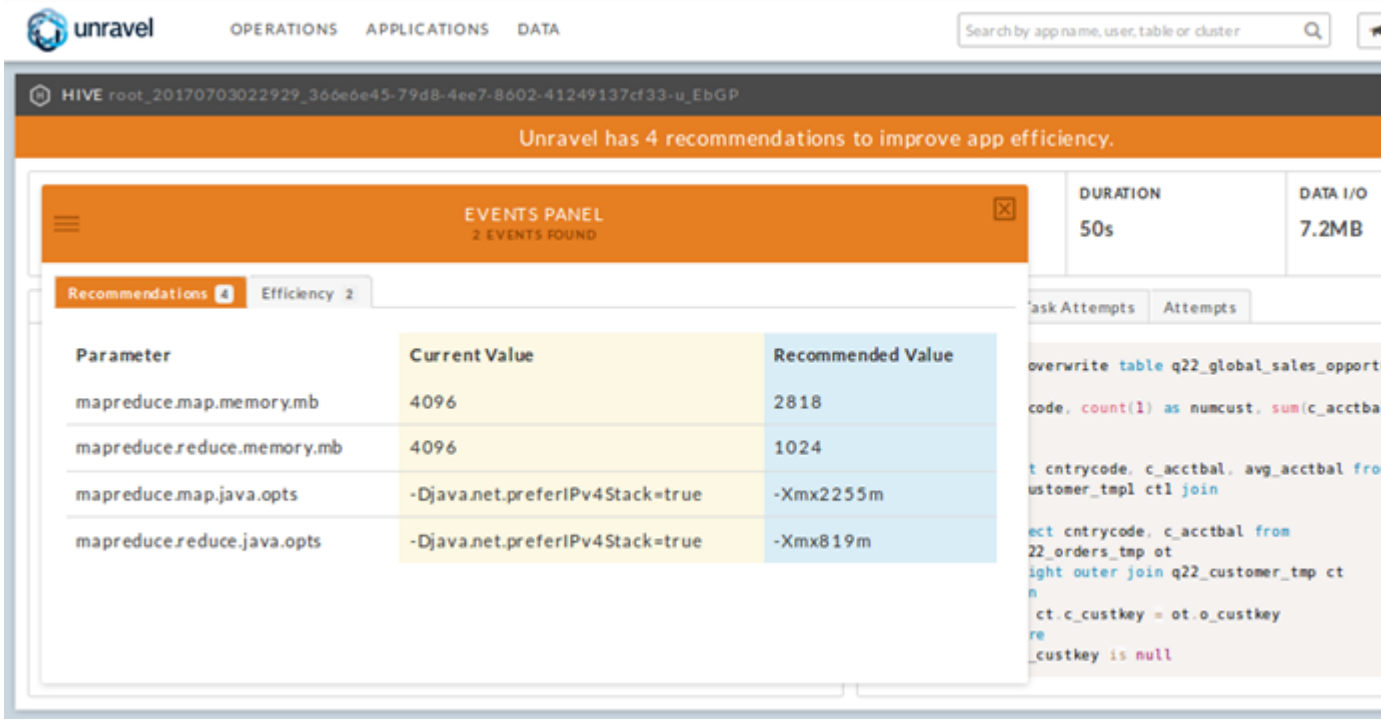
Key Performance Indicators (KPIs)

The key performance indicators (KPIs) at the top provide the most important information about the Hive query. The Hive Application Manager displays the following KPIs:

- **Events**: The number of Unravel recommendations or insights for this query. To see details, click the **Events** icon as highlighted in the image below.



The performance and reliability of your Hive query depends on several factors such as quality of the code, types of joins used, configuration settings, data size, scheduler settings, contention with other applications, and so on. Therefore, it takes significant expertise and effort to get to the root cause of problems in a Hive query. Unravel Events are designed to save you time and effort by automatically providing insights into the application. These events capture reasons for failed and killed queries as well as provide recommendations to improve application performance.



- **Duration**: Total time taken by the application to complete execution
- **Data I/O**: Total data read and written by the application
- **Number of YARN apps**: The number of YARN apps that make up the Hive query

Sub-Tabs

The Hive Application Manager contains multiple sub-tabs, each of which is described below.

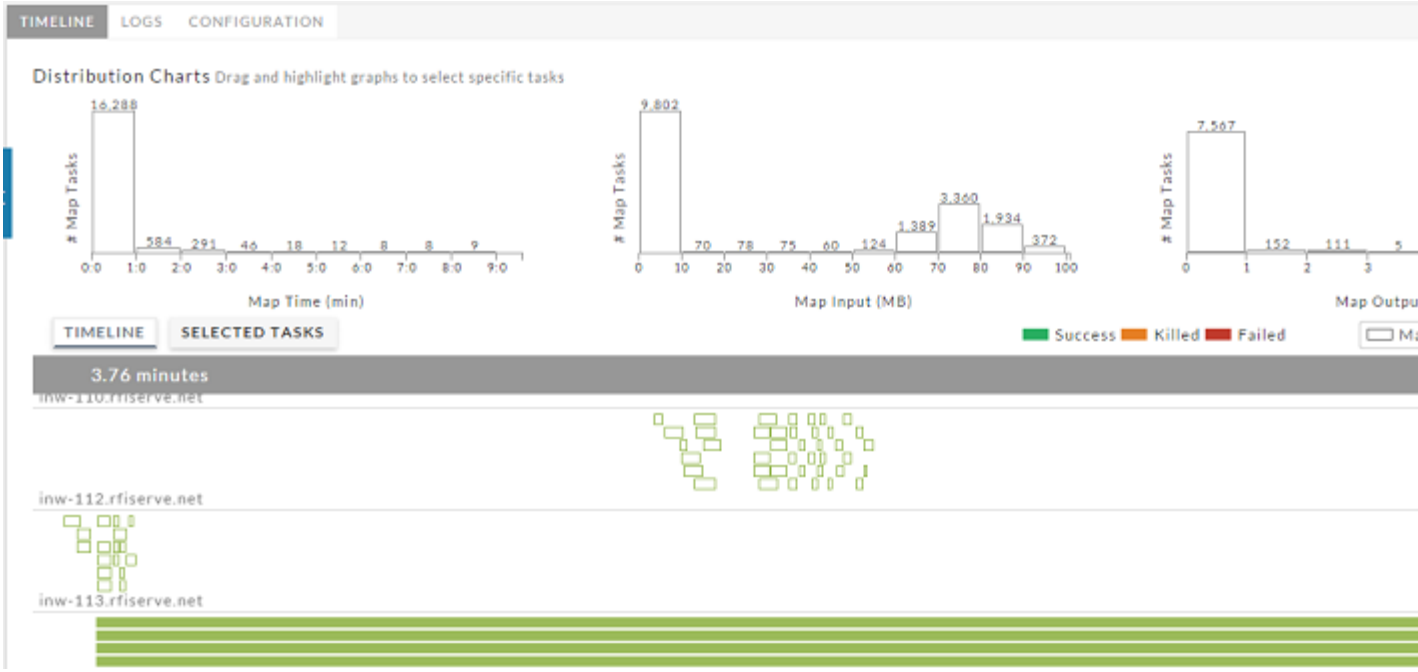
- **Navigation**: Provides an easy way to understand the breakdown of the application and drill down into MapReduce jobs that make up the application. It includes information about MapReduce jobs such as duration, I/O, resources, events, job ID, and job status. When you select a MapReduce job, its details are shown in a pane below it. This detailed view includes the MapReduce timeline, MapReduce task attempts, and slot usage graph.
- **Execution Graph**: Provides insights into the execution of the application. It shows detailed information about MapReduce stages and their relationship with one another. The execution view is split into two areas; the execution plan and the expanded info area.  This view helps identify bottlenecks and inefficiencies.
- **Gantt Chart**: Shows the relationship between MapReduce stages and high level information about each stage. The information shown in each stage box includes MapReduce job ID, base table name, time taken by the stage, percentage of total run time taken by the stage, and execution status.
- **Errors**
- **Query**: Shows detailed information about a stage when selected. The expanded info shows each of the map reduce functions, tables usage information, timings of each function and input paths used by the stage.
- **Task Attempts**: Provides an easy way to understand efficiency and status of MapReduce task attempts by breaking down attempted tasks by successful, failed and killed.
- **Attempts**

- **Slot Usage**: Shows slot usage by Map and Reduce jobs over time.

MapReduce Application Manager

The MapReduce Application Manager provides a detailed view into the behavior of MapReduce applications. It is used by Hadoop DBAs or application owners (engineers, BI team, analysts) to resolve inefficiencies, bottlenecks and reasons for failure within applications. The MapReduce Application Manager uses the Unravel intelligence engine to automatically identify certain inefficiencies with the application and provides solutions on how to the fix the problem.

It contains similar sections to the Hive Application Manager and additionally shows the timeline view of MapReduce job execution, logs and configuration.



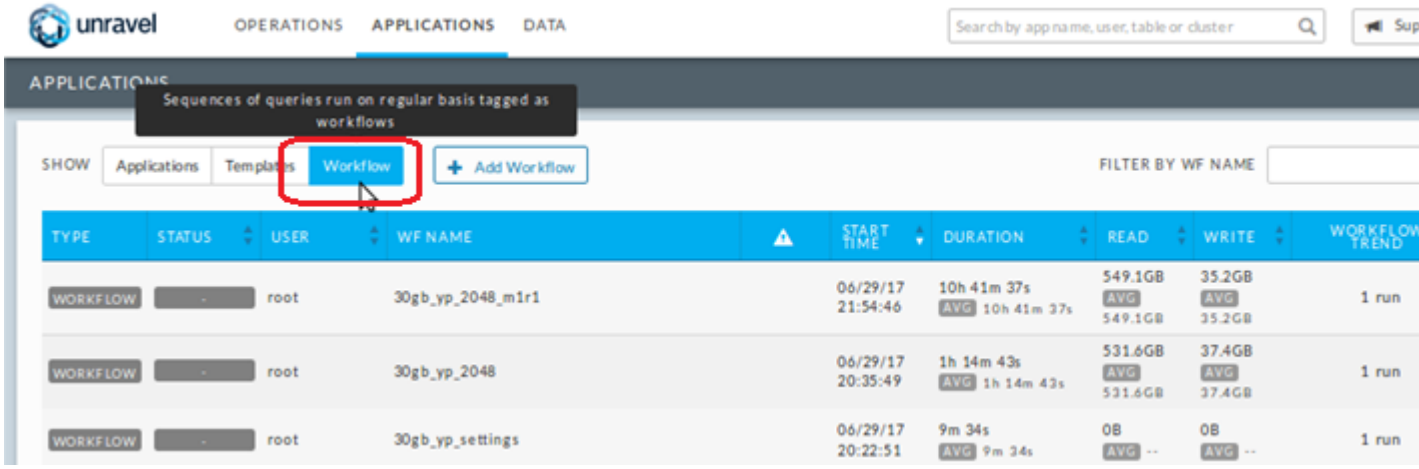
Sub-Tabs

The MapReduce Application Manager contains multiple sub-tabs, each of which is described below.

- **Timeline**: Provides a detailed view into a MapReduce job execution. The Timeline shows execution of each MapReduce task on the machine that those tasks ran on and allows drill-down into each task to obtain further information. The timeline view comes with filters which can be used to display only map tasks, reduce tasks and killed/failed tasks. The histograms above the Timeline show the distribution of MapReduce tasks along time and data size. This histogram can also be used as a filter to zoom in on specific tasks.
- **Logs**: Provides comprehensive information about each application including task and job logs. The logs section intelligently selects interesting tasks and presents its logs in an organized manner.
- **Configuration**: Provides a complete list of configuration settings used during the application execution.

Finding Workflows

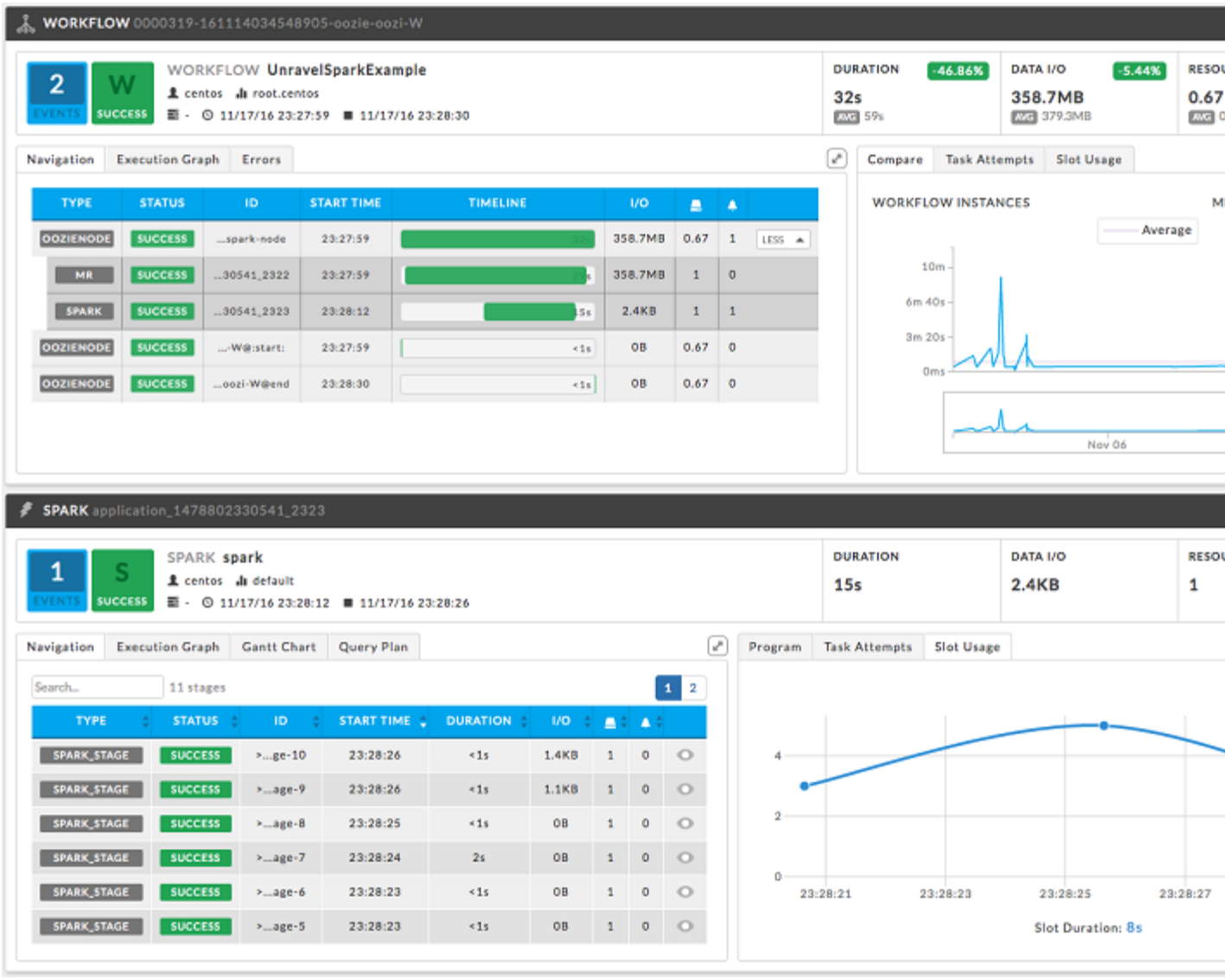
To find workflows, select **SHOW** | **Workflow** as highlighted in the image below.



Workflow Manager

The Workflow Manager provides a comprehensive view to understand workflows and their patterns of execution. It is used by Workflow (Pipelines) owners to identify anomalies, inefficiencies and bottlenecks in workflow instances. The Workflow Manager uses the Unravel intelligence engine to automatically identify inefficiencies with the workflow and provides solutions on how to the fix the problem. The Workflow Manager helps pipeline owners easily maintain SLAs.

The workflow header provides primary information about the workflow such as name, user name, queue that the workflow was submitted on, start time and tags that the workflow has been given.



Key Performance Indicators (KPIs)

- **Events**: The number of Unravel recommendations or insights for this workflow. To see details, click the **Events** icon.
- **Duration**: Total time taken by the workflow to complete execution
- **Data I/O**: Total data read and written by the workflow
- **Resources**
- **Number of Apps**: The number of apps that make up this workflow

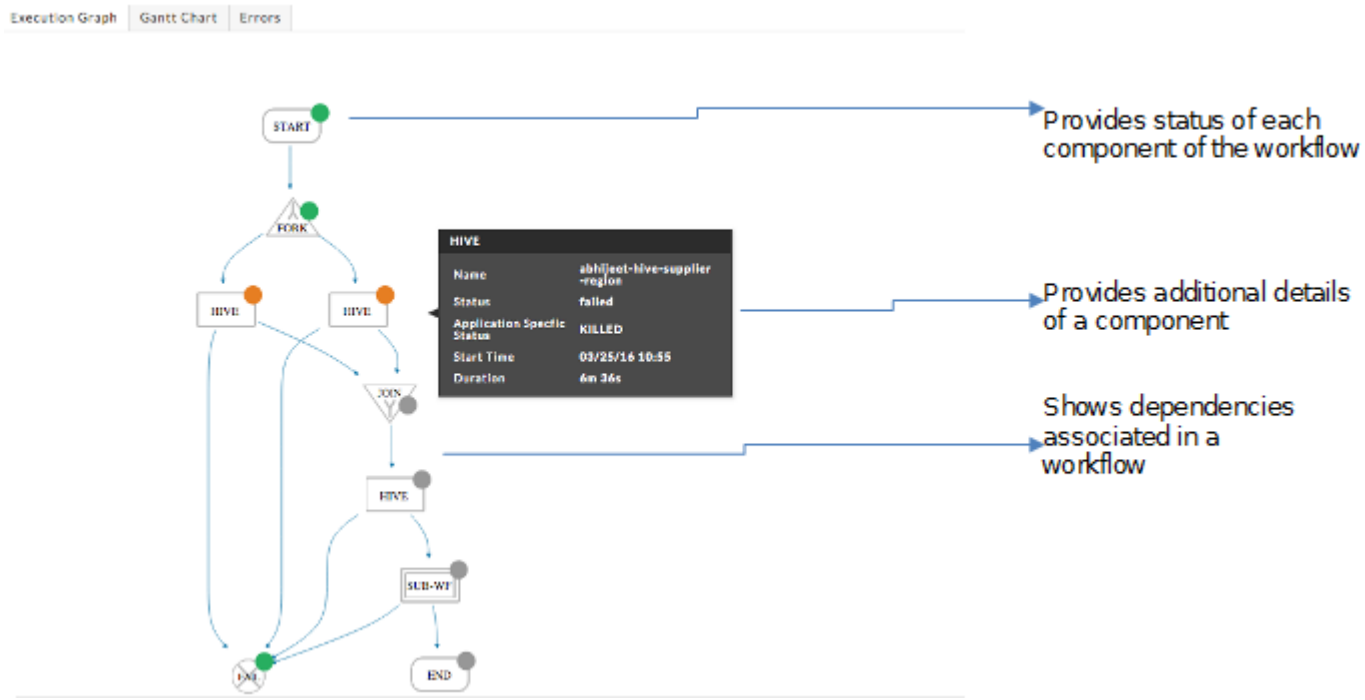
Sub-Tabs

The Workflow Manager contains multiple sub-tabs, each of which is described below.

- **Navigation**: Provides an easy way to understand the breakdown of the workflow and drill down into the Hive queries, Spark jobs, and MapReduce jobs that make up the application. It includes information about duration, I/O, resources, events, job IDs, and job status.

- **Gantt Chart**: Shows the dependency between the various components and the time taken by each. It helps you to identify stuck or incomplete components which could be affecting the overall completion of the workflow.

- **DAG View**: Clearly shows the dependencies between various components of the workflow and the status of components. To see details about status, hover over a status box to see the tool tips.

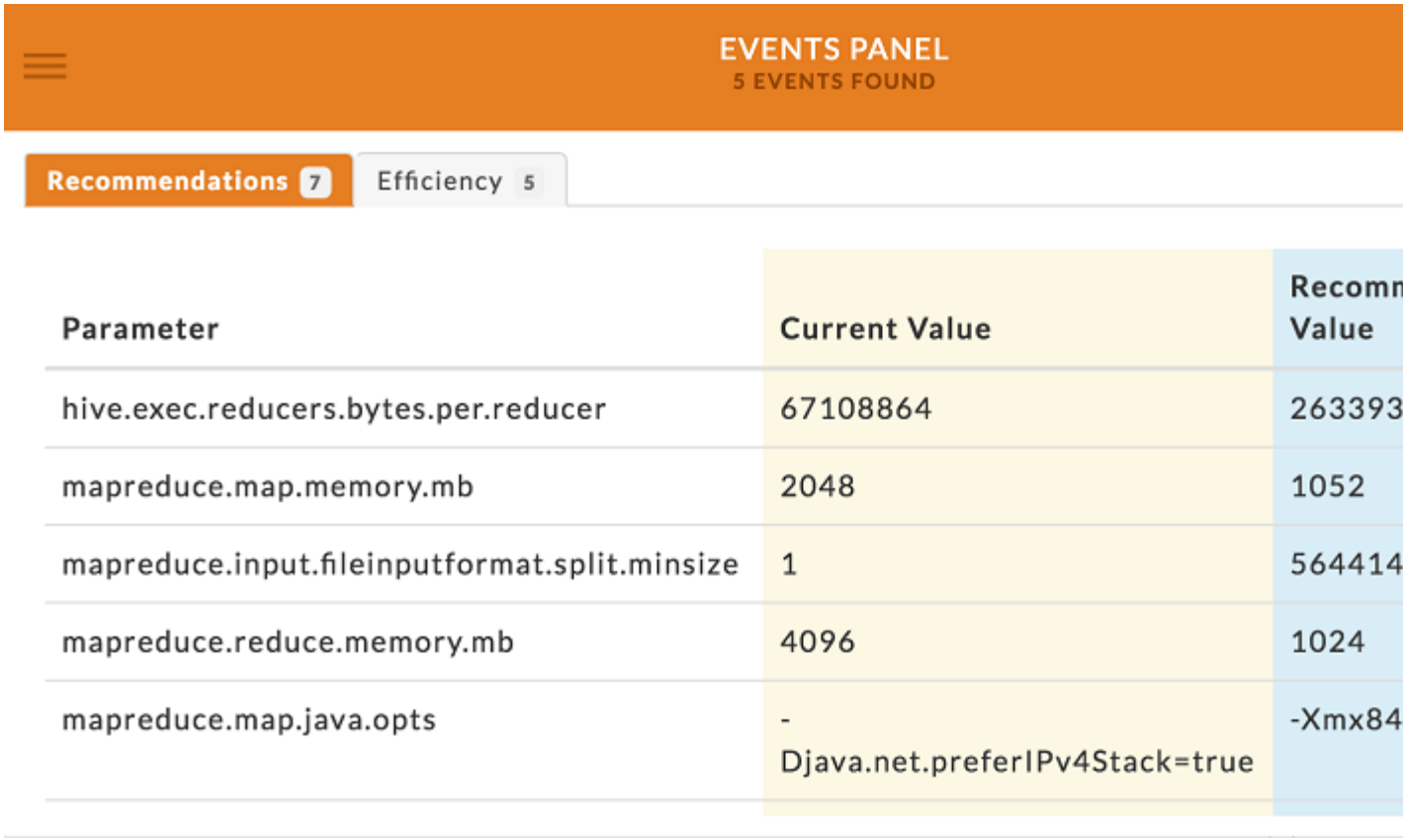


- **Compare**: Provides a quick way to understand how well a workflow run compares to its other runs. Hovering your pointer over instances displays top KPIs such as duration, data I/O, resources, and the number of jobs in that instance. Clicking on a point on the chart loads that particular instance for inspection.

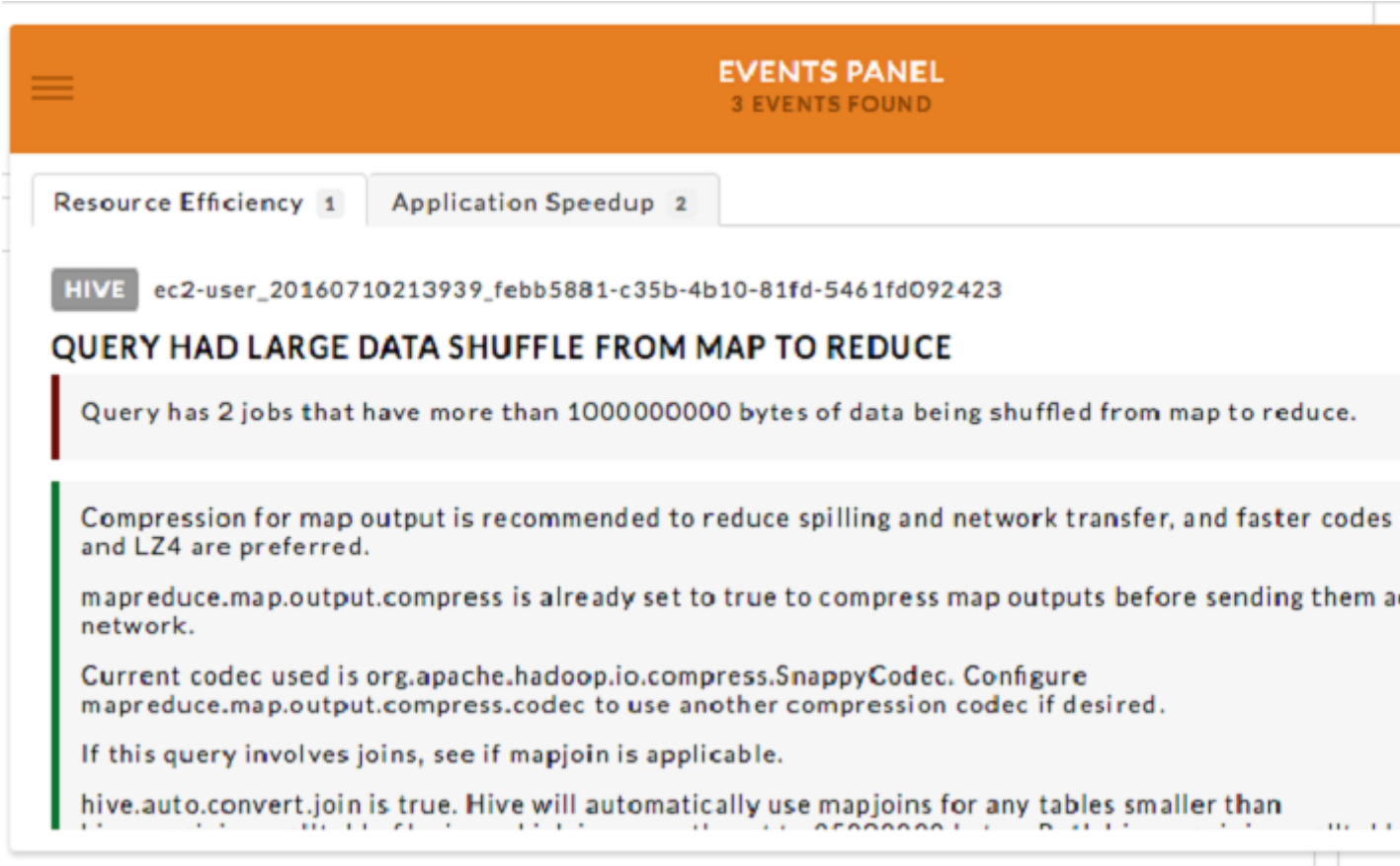
Events

Unravel Events are designed to save you time and effort by automatically providing **recommendations** and **insights** into the application. Recommendations are concrete suggestions at the level of configuration settings that can be applied at subsequent application runs to speed up the application performance, and to improve the resource efficiency. Insights, on the other hand, are identifying performance bottlenecks, data skew problems, capture reasons for failed, killed, and slow applications. In addition, insights provide useful guidelines that can be used to adjust the current configuration settings of the application. Unravel Events fall into the following categories: Application Speedup, Application Failure, Resource Efficiency, SLA Management.

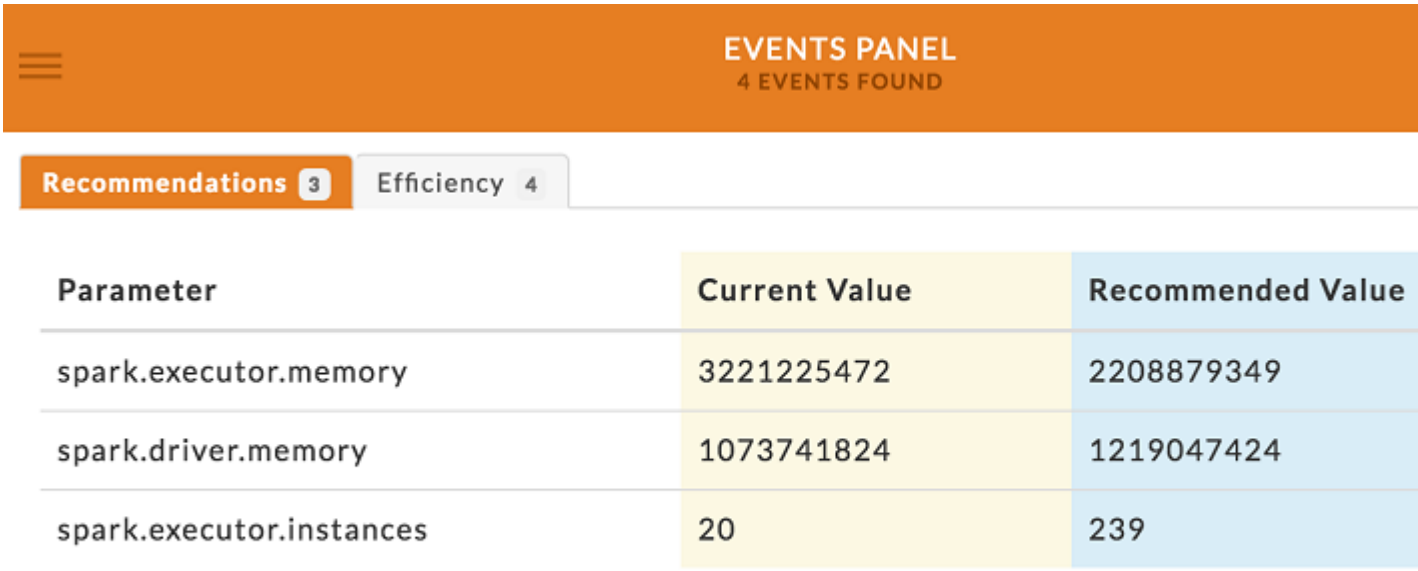
Example: Hive Recommendations



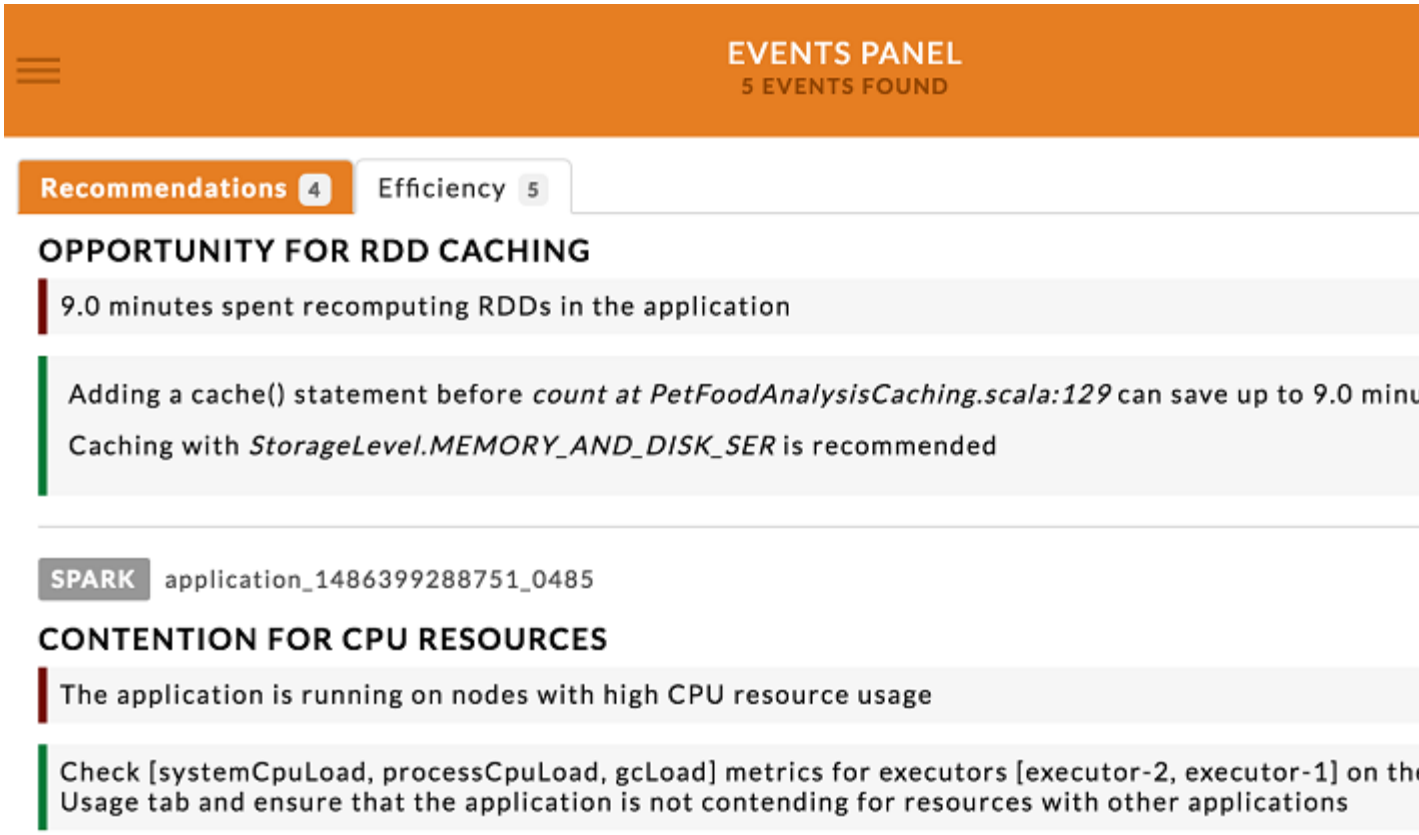
Example: Hive Insights



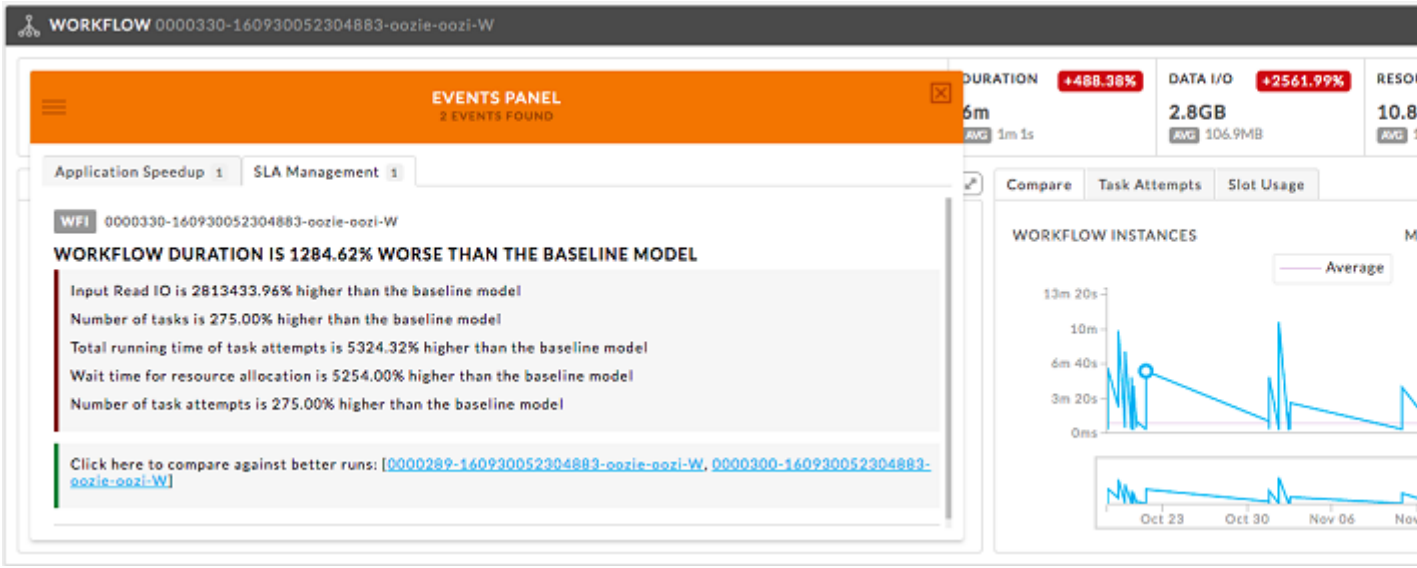
Example: Spark Recommendations



Example: Spark Insights

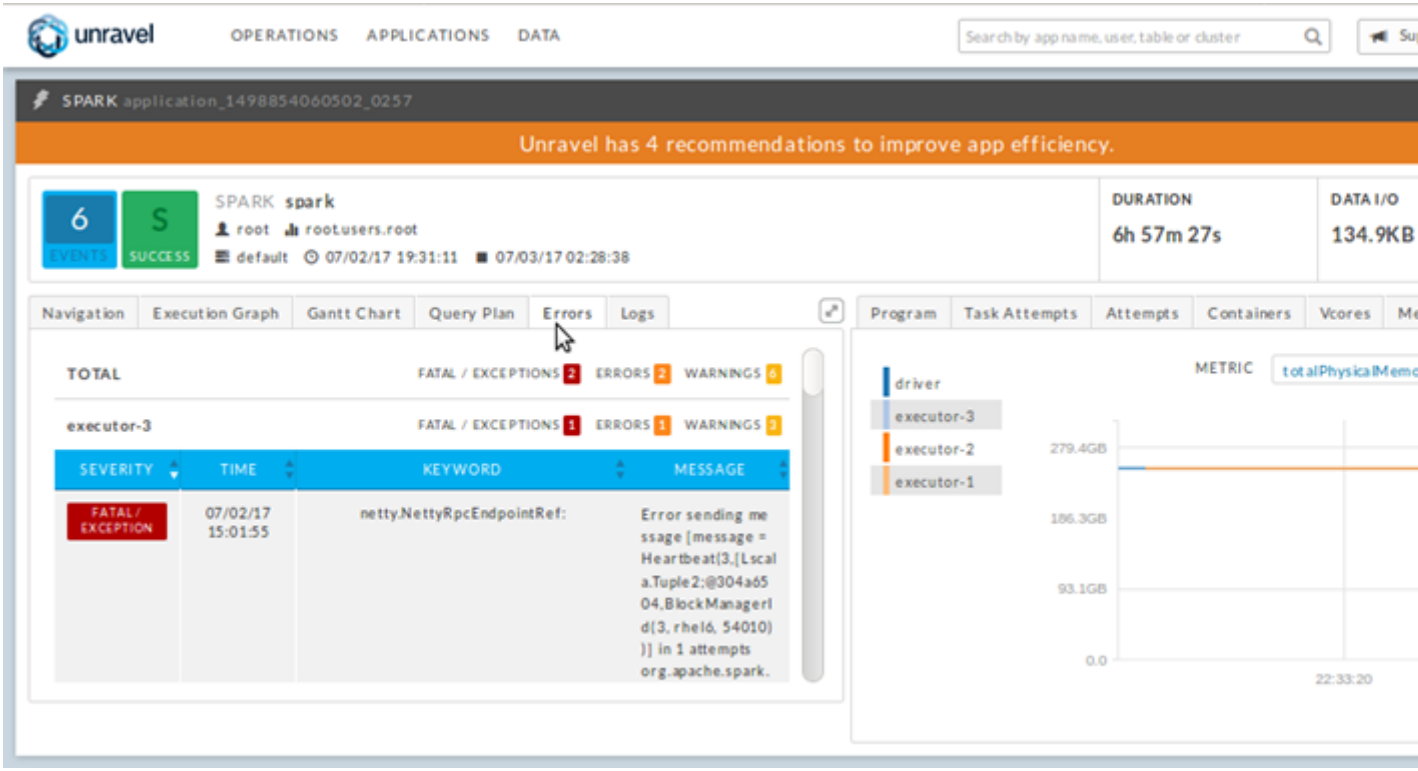


Example: Workflow Events



Error View

This new feature allows to quickly identify errors affecting your applications. Error views are available for MR, Hive and Oozie applications.



Errors for each application are categorized by Severity type and also include Keywords and details associated with each. Keywords extract important details from the errors messages/log data that can help developers/operators quickly root cause issue. Examples of keywords include Oozie errors code(s), Java run time error(s) etc.

Attachments:

- logo_small.png
- image2017-2-26_0-20-12.png
- admin_manage_configuration.png
- gui_admin_manage_configuration.png
- gui_email.png
- gui_global_search.png
- gui_spark_app_mgr.png
- gui_spark_app_mgr2.png
- gui_hive_app_mgr2.png
- gui_mapreduce_app_mgr1.png
- gui_workflow_mgr1.png
- gui_workflow_mgr2.png
- gui_spark_app_mgr3.png

- gui_workflow_mgr3.png
- gui_events_hive.png
- gui_events_spark.png
- gui_events_workflow.png
- gui_error_view.png
- gui_ops_dashboard.png
- gui_ops_cluster_metrics.png
- gui_ops_chargeback.png
- gui_data_table.png
- gui_data_table_summary.png
- gui_data_table_details.png
- gui_data_table_details2.png
- gui_data_table_details3.png
- gui_data_table_rules1.png
- gui_data_table_rules2.png
- gui_report_ops.png
- gui_report_users.png
- Unravel-Product-Use-Case-Demo-1.mp4
- Unravel-Product-Use-Case-Demo2.mp4
- Unravel-Product-Use-Case-Demo3.mp4
- Unravel-Product-Use-Case-Demo4.mp4
- spark-insights3.png
- caching-opportunity.png
- hive-recommendation.png
- gui_hive_app_mgr1.png
- gui_hive_app_mgr1_events.png
- gui_applications.png
- gui_auto_actions1.png

- gui_auto_actions2.png
- gui_auto_actions_select.png
- gui_applications.png
- gui_applications.png
- gui043goto.png
- gui_spark_app_mgr.png
- gui_hive_app_mgr1.png
- gui076.png
- gui075events.png
- gui137.png
- gui043goto.png
- gui_spark_app_mgr2.png
- gui_spark_app_mgr2.png
- gui_spark_app_mgr3.png
- gui_mapreduce_app_mgr1.png
- gui_workflow_mgr1.png
- gui_workflow_mgr2.png
- hive-recommendation.png
- gui_events_hive.png
- spark-insights3.png
- gui_events_workflow.png
- gui_error_view.png
- caching-opportunity.png
- gui053events.png
- gui054.png
- gui085cropped.png

Resource Metrics

1. Unravel 4.0-4.1

2. Unravel 4.0-4.1
3. User Guide
4. The Applications Tab

The table below lists the resource metrics collected by Unravel. The availability of a particular metric is dependent on the underlying OS and JVM GC algorithm in use.

Metric	Unit	Description
snapshotTs	TIMESTAMP (milliseconds)	The time the metric was read
startTs	TIMESTAMP (milliseconds)	The time when the collection process started
totalPhysicalMemory	BYTES	The total physical memory in the operating system
freePhysicalMemory	BYTES	The free physical memory in the operating system
committedVirtualMemory	BYTES	The committed virtual memory in the operating system
freeSwap	BYTES	The free swap size
availableMemory	BYTES	An estimate of memory available for launching new processes
vmRss	BYTES	The resident set size of the complete process tree
vmRssDir	BYTES	The resident set size of the process
totalSwap	BYTES	The total swap size
processCpuLoad	PERCENTS	Average process CPU load for the last minute (all cores)
systemCpuLoad	PERCENTS	Average system CPU load for the last minute (all cores)
fullGcCount	COUNT	Number of full GC runs
minorGcCount	COUNT	Number of minor GC runs
minorGcTime	DURATION (nanoseconds)	Accumulated time spent in minor GC
fullGcTime	DURATION (nanoseconds)	Accumulated time spent in full GC
gcEdenSurvivedAvg	BYTES	Average number of bytes moved from eden to survivor space. Might not be available for particular GC algorithms
gcSurvivorPromotedAvg	BYTES	Average number of bytes moved from survivor to old space. Might not be available for particular GC algorithms
gcYoungLiveAvg	BYTES	Average number of bytes alive in the young generation (eden + survivor spaces). Might not be available for particular GC algorithms
allocatedBytes	BYTES	Accumulated number of allocated bytes

edenPeakUsage	BYTES	Maximum memory usage in the eden space
survivorPeakUsage	BYTES	Maximum memory usage in the survivor space
oldPeakUsage	BYTES	Maximum memory usage in the old space
avgMinorInterval	DURATION (nanoseconds)	Average interval between two subsequent minor GCs. Might not be available for particular GC algorithms
gcLoad	PERCENTS	Percentage of CPU time spent in GC
blockingRatio	PERCENTS	Estimated percentage of CPU time spent in kernel blocking operations
avgFullGcInterval	DURATION (nanoseconds)	Average interval between two subsequent full GCs. Might not be available for particular GC algorithms
gcOldLiveAvg	BYTES	Average number of bytes alive in the old generation. Might not be available for particular GC algorithms
initHeap	BYTES	Initial heap size
maxHeap	BYTES	Maximum heap size
usedHeap	BYTES	Used heap size
committedHeap	BYTES	Committed heap size
initNonHeap	BYTES	Initial non-heap size
maxNonHeap	BYTES	Maximum non-heap size
usedNonHeap	BYTES	Used non-heap size
committedNonHeap	BYTES	Committed non-heap size
currentThreadCpuTime	DURATION (nanoseconds)	Current thread CPU time elapsed since the start of the measurement. Might not be available on some operating systems
currentThreadUserTime	DURATION (nanoseconds)	Current thread user time elapsed since the start of the measurement. Might not be available on some operating systems

The Data Tab

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. User Guide

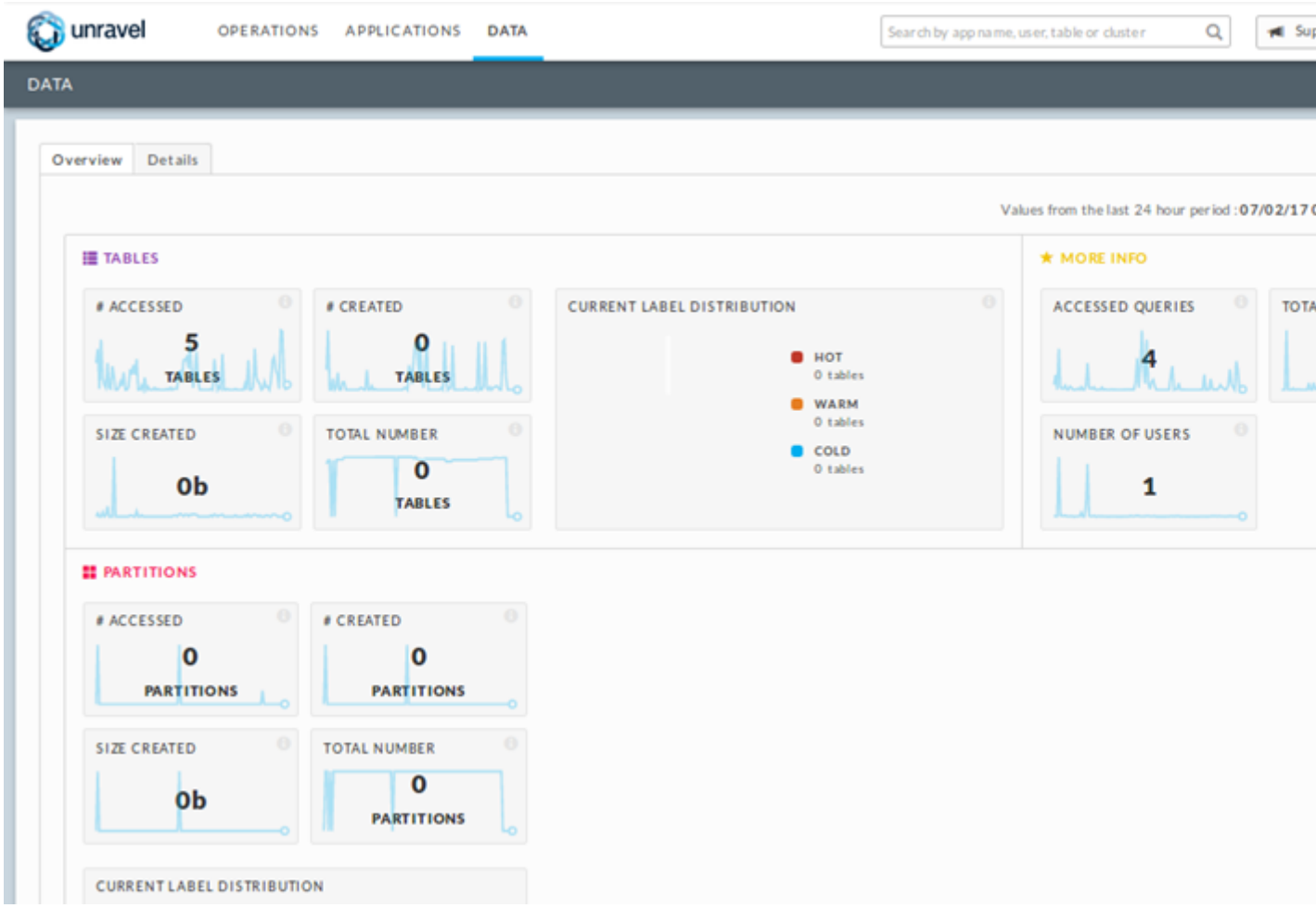
Table of Contents

- Table/Partition Usage

- Table Details
- Partition Details

Table/Partition Usage

The Data tab provides a quick snapshot of the tables and partitions in the cluster. To see table and partition details, click the **Details** tab.



On the **Details** tab, you can sort the data on any column by clicking that column's header.

- Click on any column to sort by that metric such as:
  - Last Access - to find tables used most recently
  - Created At - to sort oldest/newest tables
  - Read I/O - to find tables which have performed most I/O
  - Attempts - to find tables which have had most number of read/write attempts on them
  - Apps - to find tables which have most number of applications using it
  - Users - to find tables which have the most number of users using it
  - Other Tables - shows other tables commonly used along with a particular table in applications
  - Labels - view tables which have been labeled Hot, Warm or Cold
- Hover on columns like Users and Other Tables to know respectively which users accessed this table and which other tables are accessed along with this table.
- View trends of table usage and access by accessing the drop-down menu on the top right corner. Metrics available are Read I/O, Total Users, Total Apps, Total Attempts as shown below.

- You can also search for a particular table by typing part of the table name in the table search box as shown below. In our example we are looking for tables with the word customer in it.

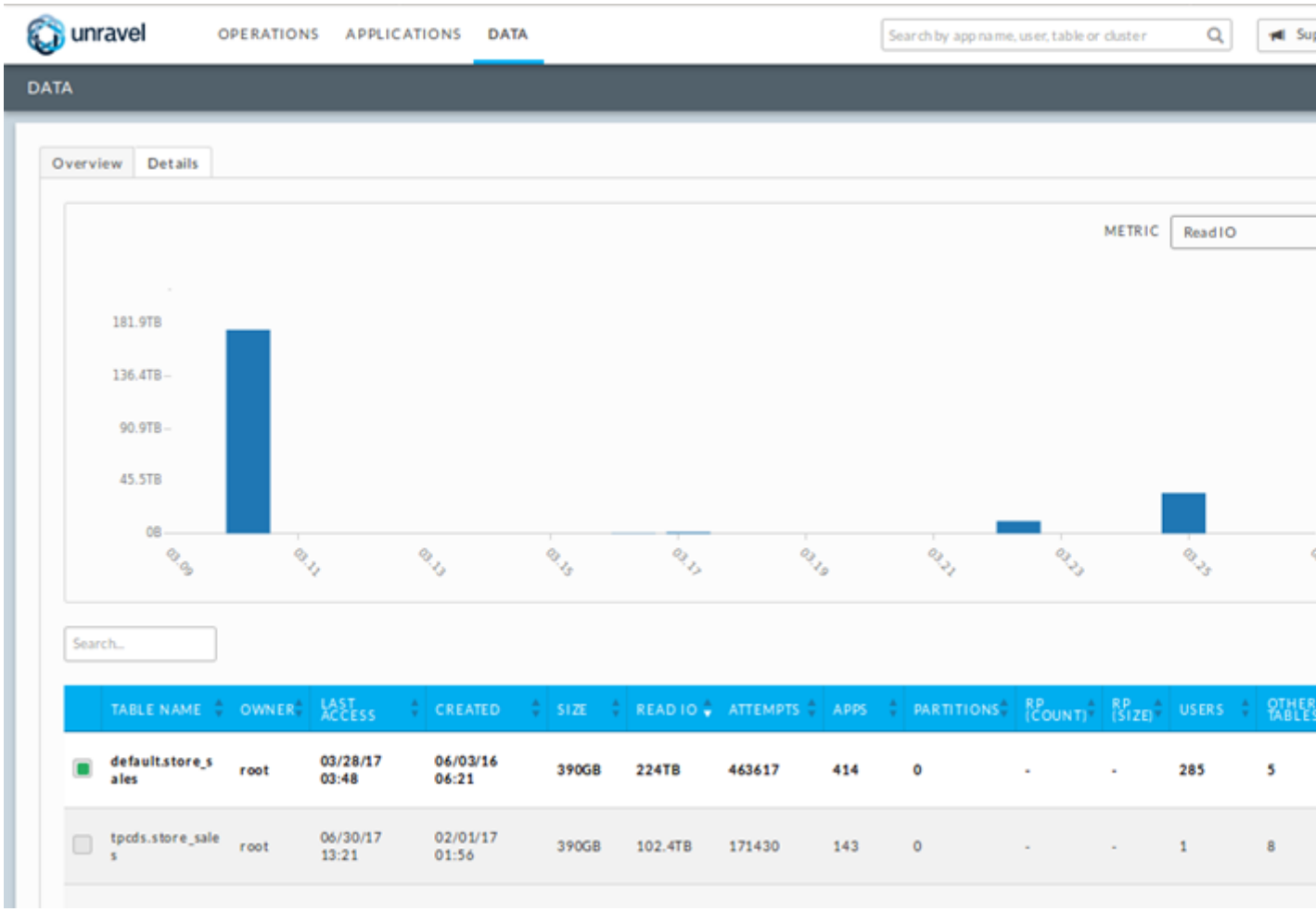
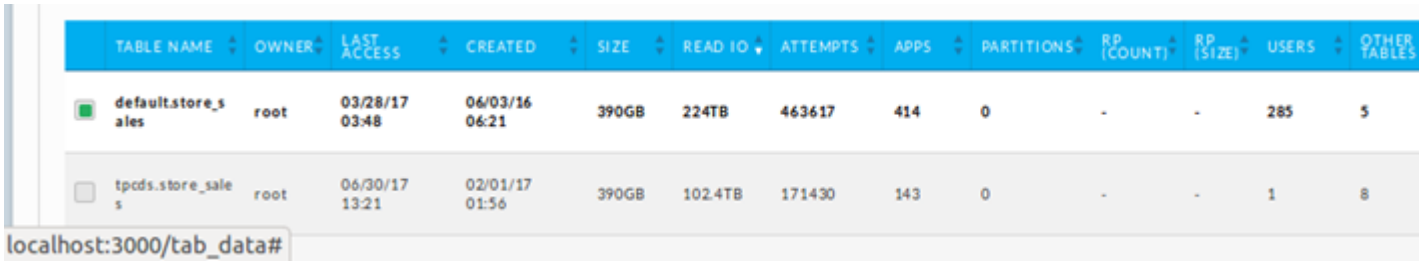
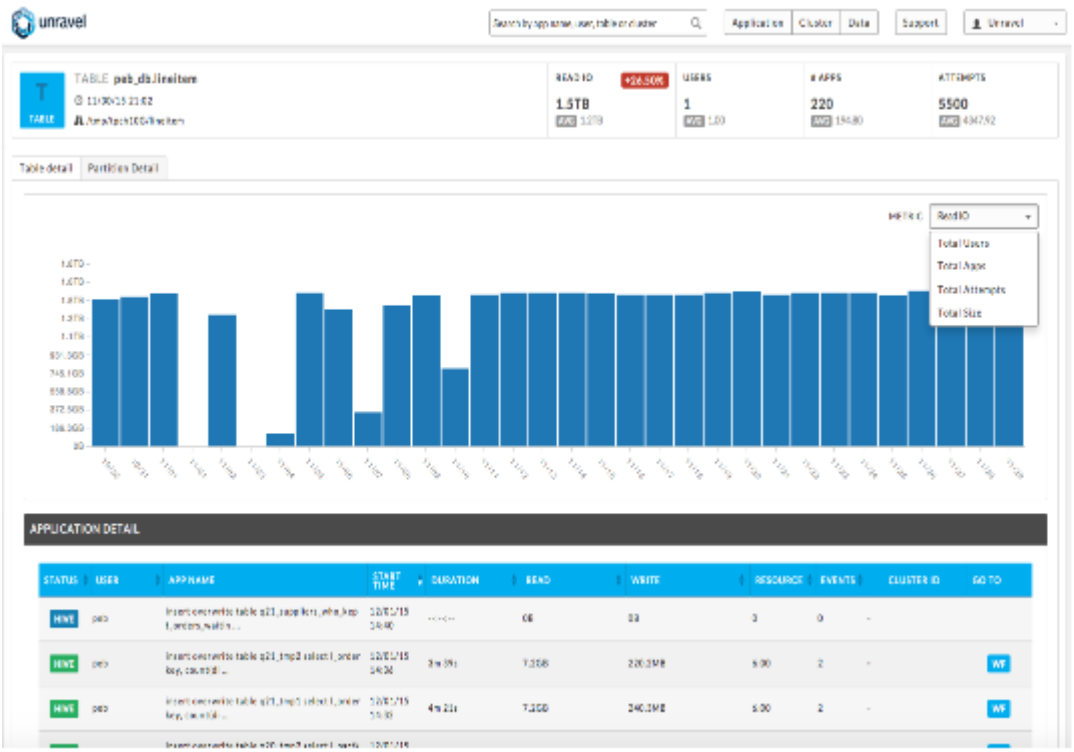


Table Details

To drill down into a particular table click its **More Info** icon.



This opens a **Table Detail** pane as shown below.



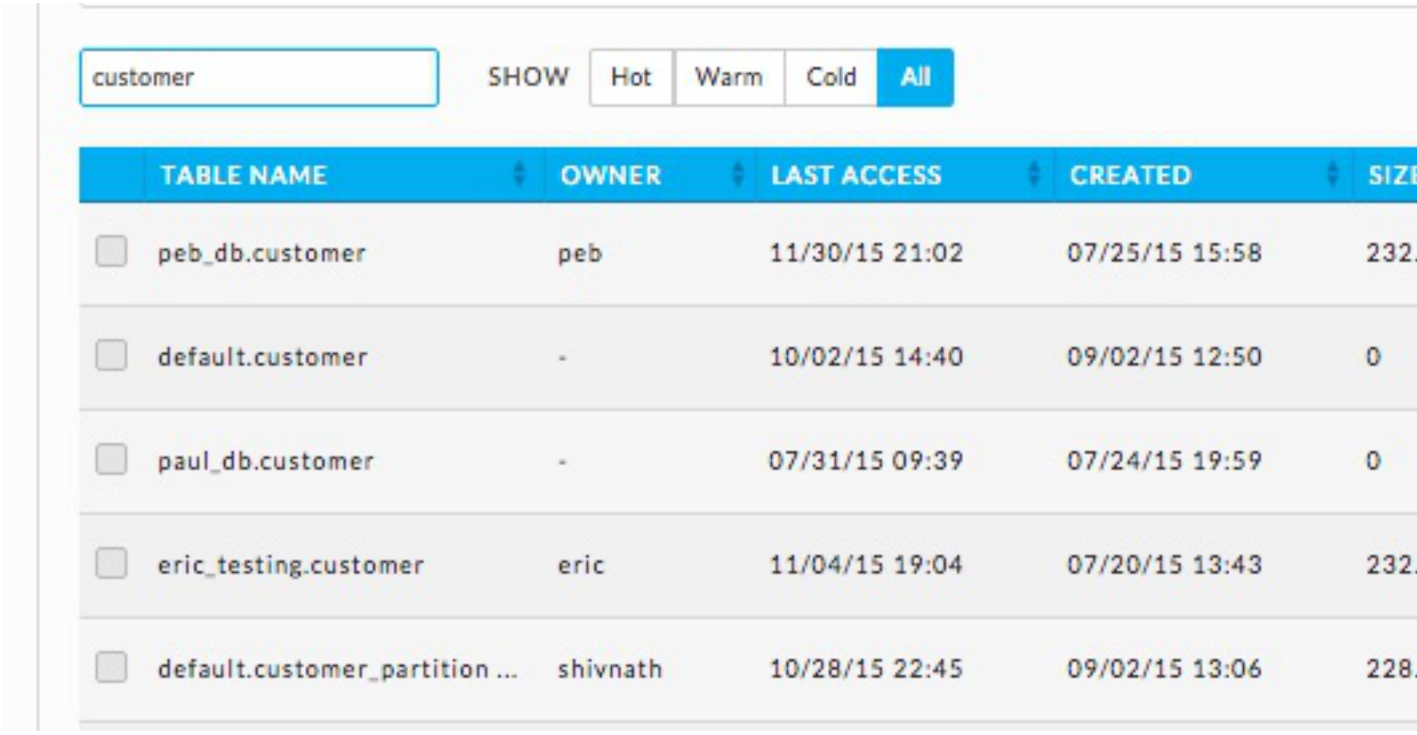
You can view different metric trends on this pane including:

- Read IO
- Number of Users
- Number of Apps
- Number of Attempts

It also allows you to view the historical usage and access information about this table. The table detail view also shows a list of applications and users that accessed the table in the given time range. This list of applications and users is also sortable for easy search and browse capabilities.

Partition Details

To view detailed information about partitions associated with a Table, click the Partition Detail tab in the Table Detailed View and you will open up a Partition detail page as shown below.



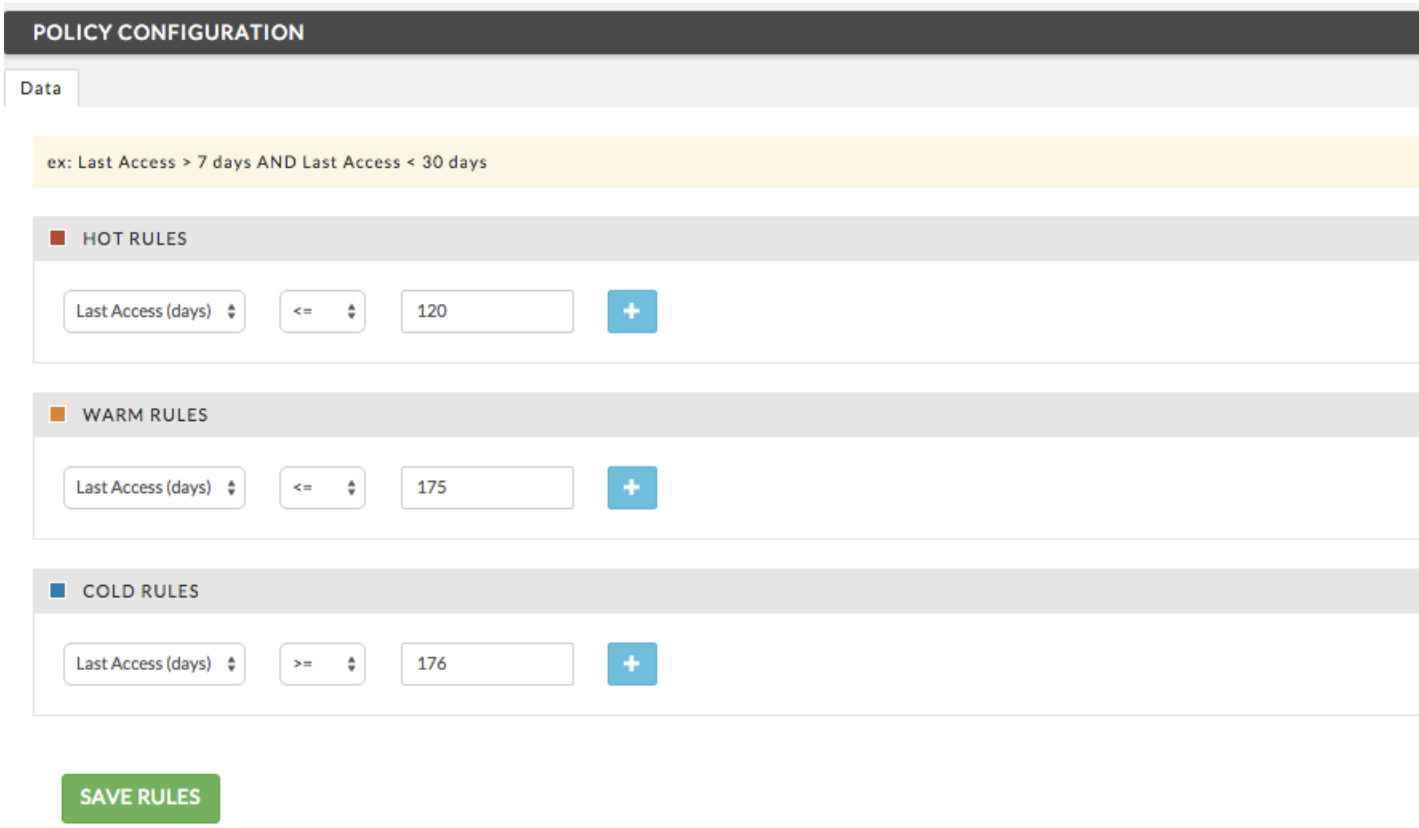
The Partition histogram provide the Access Age of Partitions (including the Max Access Age). This information is used to calculate the number and size of Reclaimable Partitions. This can be explained as follows:

- Reclaimable Partitions Found = number of Partitions with Last Access Time < (Current Time - Max Access Age of Partitions). In the example above:
- No partition is accessed by any application more than 4 days 14 hrs. after it is created. Thus, the partitions created on Dec 1, 2015, will not be accessed after Dec 5, 2015.
- In Unravel, the Max Access Age of Partitions for the above Table will be computed as 4 days 14 hrs. based on the history of accesses to partitions in the above Table
- So on Dec 15, 2015, if the above Table is reviewed:
- Reclaimable Partitions found = number of Partitions with Last Access Time < (Dec 14 2015 - 4days 14 hrs.) i.e., (Dec 10, 2015). Thus, the partitions created on any day before Dec 10, 2015 will be marked as a Reclaimable Partition. In this case, it will be all the 26 partitions.

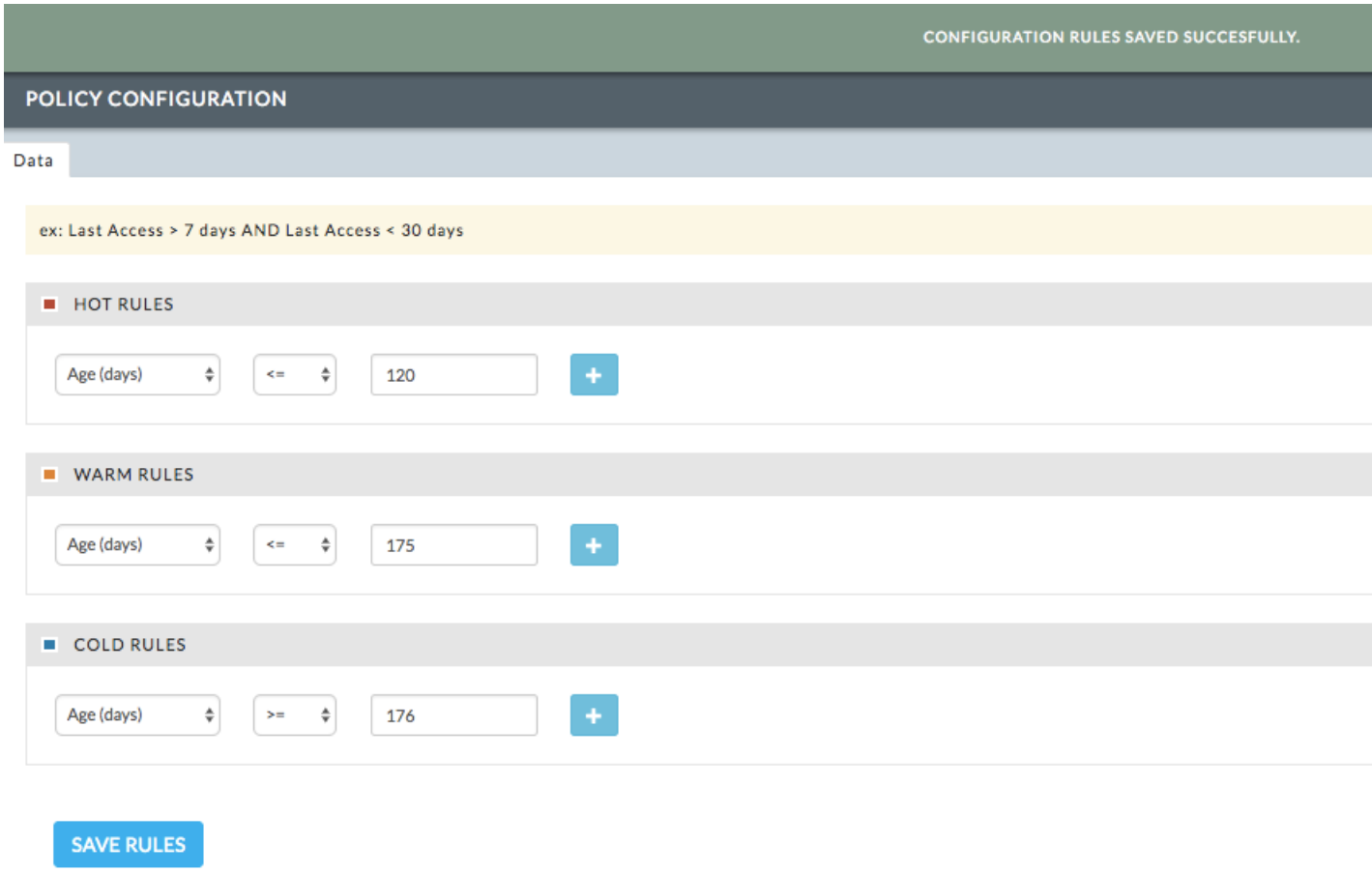
The page also list all Partitions by key KPI's, including:

- Last access date/time
- Create date/time
- Current Size
- # of Users accessing the Partition

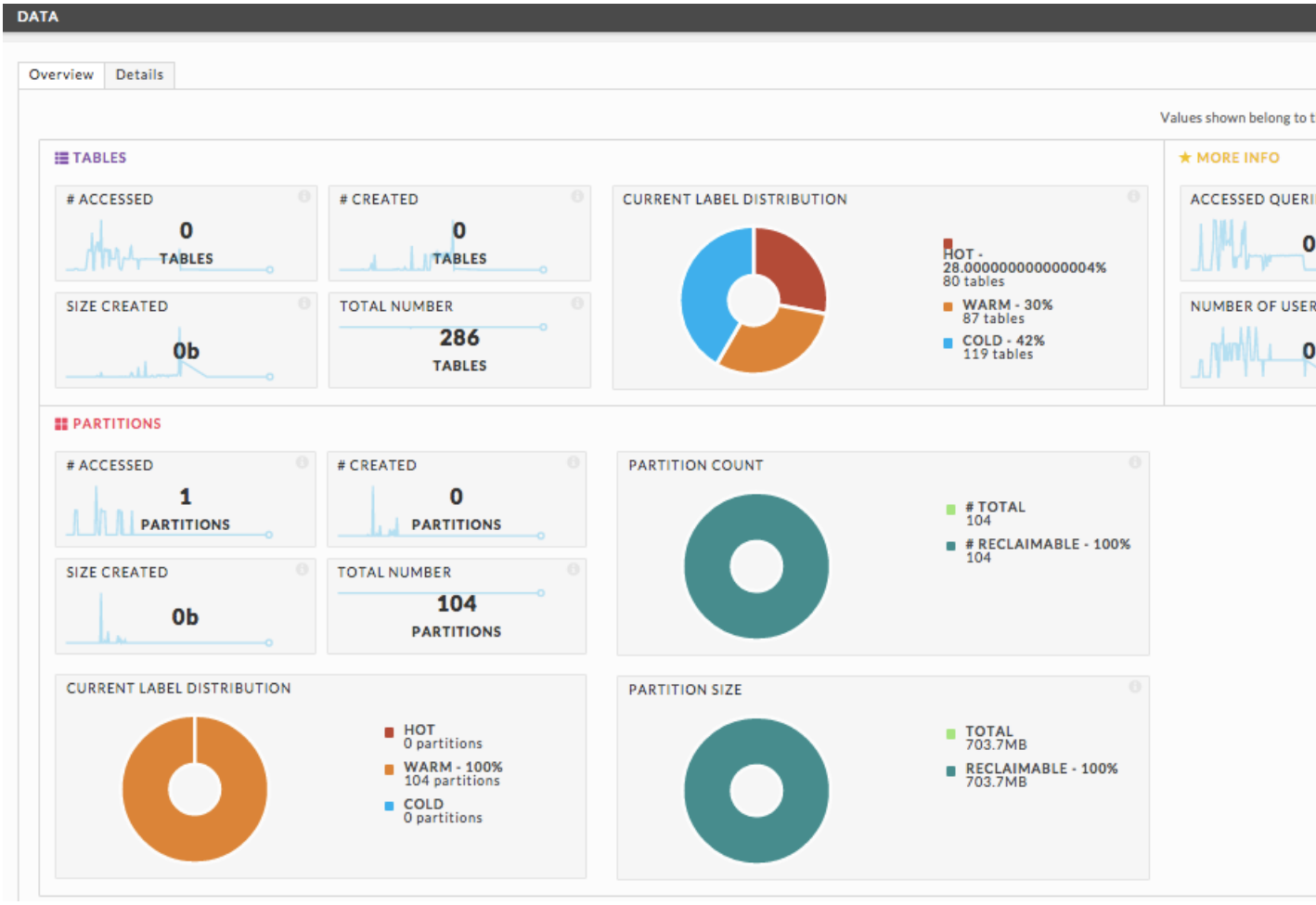
Unravel provides an easy way to label Tables/Partitions as “HOT”, “WARM” and “COLD” based on access patterns – last access time or age (last access time – create time).



Once you have the rule configured as above, please click on the "SAVE RULES" and you will see a confirmation as below.



While the "HOT", "WARM" and "COLD" labels are associated with the Tables right away, the main dashboard page with appropriate pie chart values are typically populated in 24 hrs.



As illustrated above, HOT rule can be "Last Access <= 120 days", WARN rule can be "Last Access <= 175 days and > 120 days" and COLD rule can be "Last Access >= 176 days"

These rules are checked periodically (every 24 hrs.) and tables/partition are labeled and classified accordingly. The Data overview page provides a quick summary information based on these labels. Operators can use this information to identify unused or frequently used tables/partitions and take appropriate actions.

Attachments:

- gui_auto_actions_select.png (image/png)
- gui_auto_actions2.png (image/png)
- gui_auto_actions1.png (image/png)
- gui_applications.png (image/png)
- gui_hive_app_mgr1_events.png (image/png)
- gui_hive_app_mgr1.png (image/png)
- hive-recommendation.png (image/png)
- caching-opportunity.png (image/png)

- spark-insights3.png (image/png)
- Unravel-Product-Use-Case-Demo4.mp4 (video/mp4)
- Unravel-Product-Use-Case-Demo3.mp4 (video/mp4)
- Unravel-Product-Use-Case-Demo2.mp4 (video/mp4)
- Unravel-Product-Use-Case-Demo-1.mp4 (video/mp4)
- gui_report_users.png (image/png)
- gui_report_ops.png (image/png)
- gui_data_table_rules2.png (image/png)
- gui_data_table_rules1.png (image/png)
- gui_data_table_details3.png (image/png)
- gui_data_table_details2.png (image/png)
- gui_data_table_details.png (image/png)
- gui_data_table_summary.png (image/png)
- gui_data_table.png (image/png)
- gui_ops_chargeback.png (image/png)
- gui_ops_cluster_metrics.png (image/png)
- gui_ops_dashboard.png (image/png)
- gui_error_view.png (image/png)
- gui_events_workflow.png (image/png)
- gui_events_spark.png (image/png)
- gui_events_hive.png (image/png)
- gui_workflow_mgr3.png (image/png)
- gui_spark_app_mgr3.png (image/png)
- gui_workflow_mgr2.png (image/png)
- gui_workflow_mgr1.png (image/png)
- gui_mapreduce_app_mgr1.png (image/png)
- gui_hive_app_mgr2.png (image/png)
- gui_spark_app_mgr2.png (image/png)

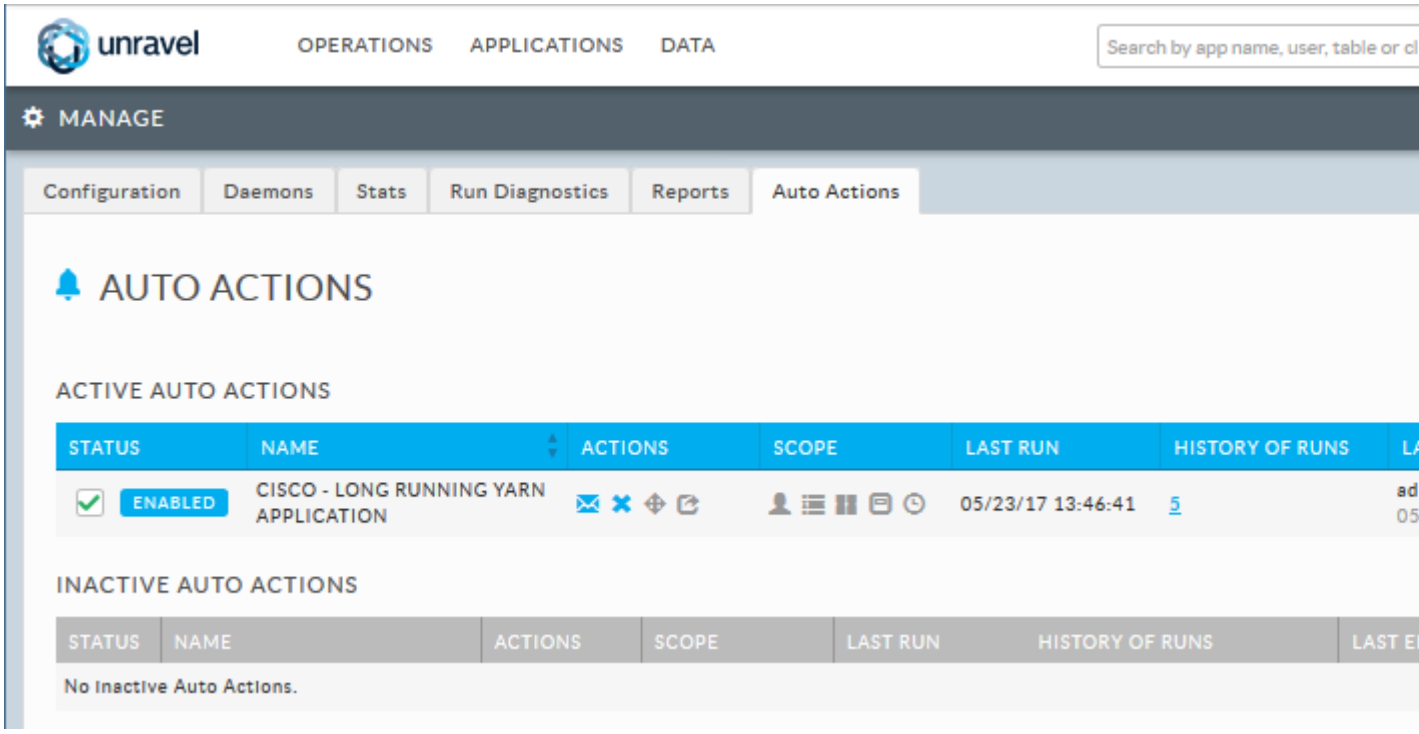
- gui_spark_app_mgr.png (image/png)
- gui_global_search.png (image/png)
- gui_email.png (image/png)
- gui_admin_manage_configuration.png (image/png)
- admin_manage_configuration.png (image/png)
- image2017-2-26_0-20-12.png (image/png)
- logo_small.png (image/png)
- Screen Shot 2017-07-06 at 1.14.55 PM.png (image/png)
- gui_data_table.png (image/png)
- gui_data_table.png (image/png)
- gui_data_table_summary.png (image/png)
- gui091moreinfo.png (image/png)

Setting Up Auto Actions (Alerts)

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. User Guide

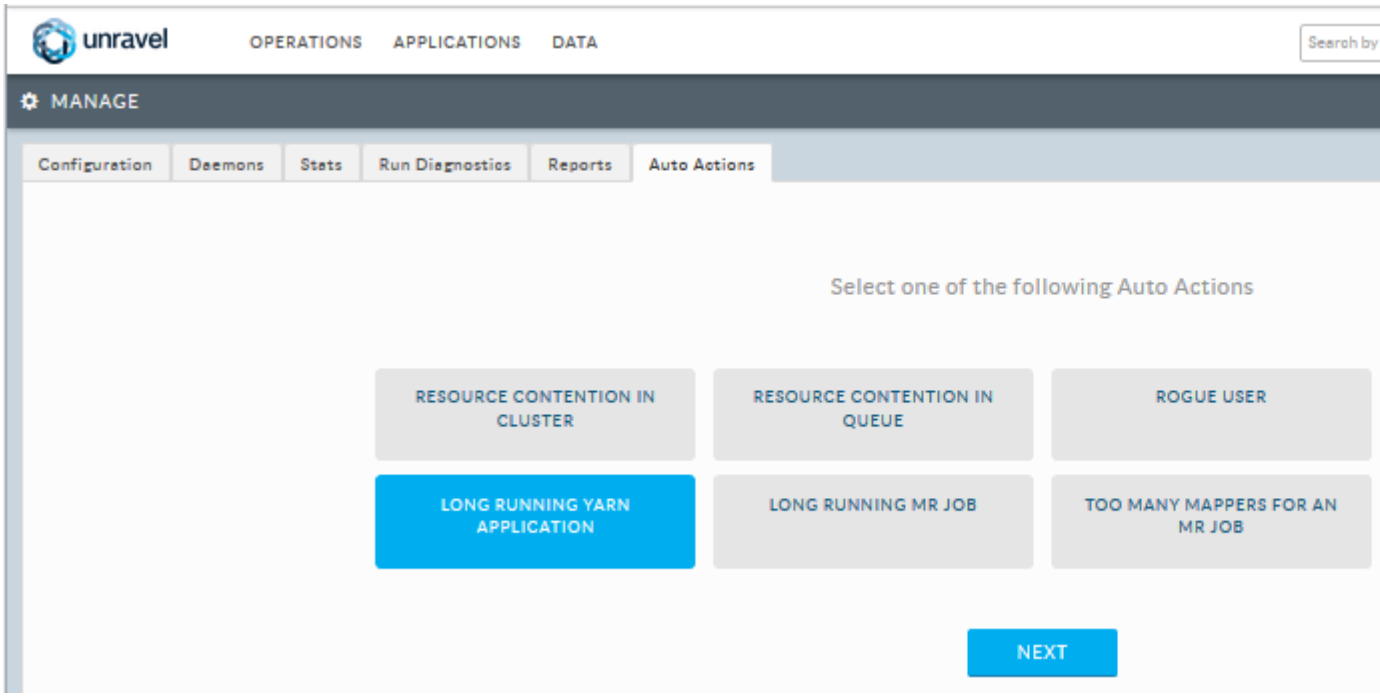
Auto Actions

An Auto Action is basically a policy such that when it is violated, an action is taken automatically. You can use an Auto Action to alert you to a situation that needs manual intervention, such as resource contention, stuck jobs, failed queries, and so on.



To create an Auto Action:

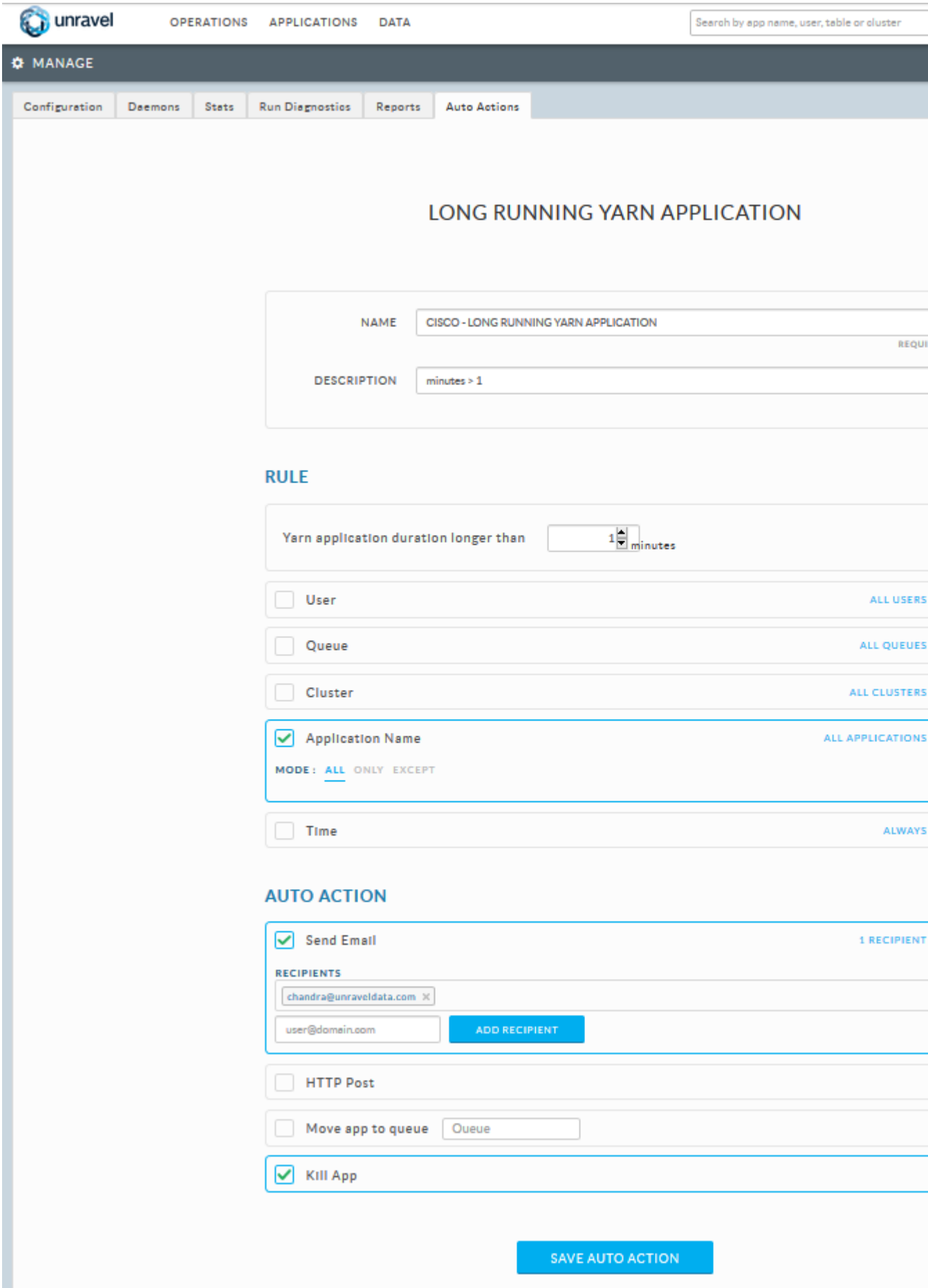
1. Click the **Admin** pull-down menu, and select **Manage**.
2. On the **Manage** page, select the **Auto Actions** tab.
3. Click **ADD NEW AUTO ACTION**.
4. Select the desired Auto Action type, and then click **Next**.



5. Enter a name for the new Auto Action, and specify its prerequisite conditions, defining conditions, and actions:
   - Prerequisite conditions: A set of boolean conditions such that when they are all met, Unravel will evaluate the defining conditions of the Auto Action. Examples include: whether this Auto Action should be evaluated during a given time, whether this Auto Action should be evaluated for a job belonging to a given user, etc.

- Defining conditions: A set of boolean conditions defining the Auto Action. When they are all met, the corresponding action defined as part of the Auto Action will be taken automatically. Examples include: Is this job running for too long? Does it use too many mappers?

- Actions: A set of actions to be taken when the defining conditions are all evaluated to be true. Examples include: send an email to admin (i.e., “alerting”), kill the job, etc.



6. Click **SAVE AUTO ACTION**.

Attachments:

- logo_small.png (image/png)
- image2017-2-26_0-20-12.png (image/png)
- admin_manage_configuration.png (image/png)
- gui_admin_manage_configuration.png (image/png)
- gui_email.png (image/png)
- gui_global_search.png (image/png)
- gui_spark_app_mgr.png (image/png)
- gui_spark_app_mgr2.png (image/png)
- gui_hive_app_mgr2.png (image/png)
- gui_mapreduce_app_mgr1.png (image/png)
- gui_workflow_mgr1.png (image/png)
- gui_workflow_mgr2.png (image/png)
- gui_spark_app_mgr3.png (image/png)
- gui_workflow_mgr3.png (image/png)
- gui_events_hive.png (image/png)
- gui_events_spark.png (image/png)
- gui_events_workflow.png (image/png)
- gui_error_view.png (image/png)
- gui_ops_dashboard.png (image/png)
- gui_ops_cluster_metrics.png (image/png)
- gui_ops_chargeback.png (image/png)
- gui_data_table.png (image/png)
- gui_data_table_summary.png (image/png)
- gui_data_table_details.png (image/png)
- gui_data_table_details2.png (image/png)
- gui_data_table_details3.png (image/png)

gui_data_table_rules1.png (image/png)

gui_data_table_rules2.png (image/png)

gui_report_ops.png (image/png)

gui_report_users.png (image/png)

Unravel-Product-Use-Case-Demo-1.mp4 (video/mp4)

Unravel-Product-Use-Case-Demo2.mp4 (video/mp4)

Unravel-Product-Use-Case-Demo3.mp4 (video/mp4)

Unravel-Product-Use-Case-Demo4.mp4 (video/mp4)

spark-insights3.png (image/png)

caching-opportunity.png (image/png)

hive-recommendation.png (image/png)

gui_hive_app_mgr1.png (image/png)

gui_hive_app_mgr1_events.png (image/png)

gui_applications.png (image/png)

gui_auto_actions1.png (image/png)

gui_auto_actions2.png (image/png)

gui_auto_actions_select.png (image/png)

Chapter 4

Unravel Server Reference

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1

This reference covers both the on-premises and Unravel for EMR editions. Some daemons and properties do not apply to the EMR edition.

Unravel Server Daemons

The Unravel service is composed of many daemons which are summarized in the next table. The single script `/etc/init.d/unravel_all.sh` can be used with a `start` or `stop` or `restart` argument to control all the daemons on a host, in the correct order. The suffix `_N` means 1, 2, 3, up to 4 separate daemons.

Daemon Logical Name	Description
unravel_tc	bundled TomCat (port 3000), Web UI
unravel_db	bundled db (on a custom port)
unravel_zk_N	bundled Zookeeper (on a custom port)
unravel_k	bundled Kafka (on a custom port)
unravel_hhs	Hive Hook Sensor
unravel_hhw	Hive Hook Worker
unravel_hhwe	Hive Hook Worker EMR
unravel_hvw	Hive Worker
unravel_jcs1	Job Collector Sensor MRv1
unravel_jcs2	Job Collector Sensor YARN
unravel_jcse2	Job Collector Sensor YARN for EMR
unravel_jcw1_N	Job Collector Sensor Worker MRv1
unravel_jcw2_N	Job Collector Sensor Worker YARN
unravel_lr	Log Receiver
unravel_mrw	Map Reduce Worker
unravel_ja	"Job Analyzer" summarizes jobs
unravel_td	"Tidy Dir" cleans up and archives hdfs directories, db retention cleaner

unravel_os3	Oozie v3 Sensor
unravel_os4	Oozie v4 Sensor
unravel_tw	Table Worker
unravel_pw	Partition Worker
unravel_ew_N	Event Worker
unravel_sw_N	Spark Worker

Unravel Server Adjustable Properties

TITLE_HERE

The file `/usr/local/unravel/etc/unravel.properties` contains settings that can be preserved during an RPM upgrade. These properties are described in the following table:

Property Name	Default Value	Description
`com.unraveldata.tmpdir`	`/srv/unravel/tmp` or `$UNRAVEL_DATA_DIR/tmp`	Determines where daemons will put temporary files. Depending on the daemon role, up to 10GB might be used temporarily.
`com.unraveldata.logdir`	`/usr/local/unravel/logs`	Do not set directly; set UNRAVEL_LOG_DIR in etc/unravel.ext.sh instead and this property will be derived from that
`com.unraveldata.zk.quorum`	`127.0.0.1:4181`	embedded Zookeeper ensemble in form host1:port1,host2:port2,…
`com.unraveldata.kafka.brokerlist`	`127.0.0.1:4091`	embedded Kafka
`unravel.jdbc.username`	`unravel`	MySQL (embedded or external) username for db
`unravel.jdbc.password`	random generated for bundled MySQL	MySQL (embedded or external) password for db
`unravel.jdbc.url`	`jdbc:mysql://127.0.0.1:3306/unravel_mysql_prod`	This is JDBC URL without username and password

com.unraveldata.hdfs.interactive.monitoring.interval.sec	30	Number of seconds between checks for presence of hive queries and MR logs to load into Unravel for interactive visibility; should be between 5 and 60 (inclusive)
com.unraveldata.hdfs.batch.monitoring.interval.sec	300	Number of seconds between checks for presence of hive queries and MR logs to load into Unravel for batch visibility; should be between 300 and 1800 (inclusive)
com.unraveldata.longest.job.duration.days	2	Number of days for the longest running Map Reduce job ever expected; should be between 2 and 7 (inclusive)
oozie.server.url	http://localhost:11000/oozie	URL for accessing Oozie to track workflows
com.unraveldata.oozie.fetch.num	200	Max number of jobs to fetch during an interval
com.unraveldata.oozie.fetch.interval.secs	120	Seconds between intervals for fetching Oozie workflow status

Adjustable Environment Settings

The optional file `/usr/local/unravel/etc/unravel.ext.sh` is `source`'d to allow site-specific env variables to be set. The following table shows possible choices:

Env Variable	Default if not set	Description
JAVA_HOME	Should use update-alternatives to make correct Java first choice	The standard way to specify the home dir. of Oracle Java so that $JAVA_HOME/bin/java is the jvm
JAVA_EXT_OPTS	*unset*	Last chance arguments to jvm to override other settings
HADOOP_CONF	as discovered by running "hadoop fs -ls "	The directory containing the hadoop config files core-site.xml, hdfs-site.xml, and mapred-site.xml

UNRAVEL_DATA_DIR	/srv/unravel	A base dir. owned by user unravel, during installation unravel creates multiple sub dirs for holding persistent data (db_data, k_data, zk_data) and also tmp data if property com.unraveldata.tmpdir is not set.
UNRAVEL_LISTEN_PORT	3000	The Web UI port on the primary or standalone Unravel installation (service unravel_tc) which listens on 0.0.0.0 ; the property com.unraveldata.tc.port automatically reflects this env var.
UNRAVEL_LOG_DIR	/usr/local/unravel/logs	A destination dir. owned by user unravel for log files
UNRAVEL_TC_SHUTDOWN_PORT	3005	An unoccupied port used for cleanly stopping the Web UI (service unravel_tc) which listens on 127.0.0.1

Unravel Server Directories and Files

The following is a cross-reference of notable directories and files used by Unravel Server:

Path	Purpose	Expected Size	Notes
/usr/local/unravel	Unravel Server installation directory	1-2.5GB	This directory is created by installing the Unravel RPM; this is a fixed destination
/usr/local/unravel/logs	Logs written by Unravel daemons	~1.5GB max	Each daemon will have a maximum of 100MB of logs, auto-rolled; use a symlink to put on another partition.

/srv/unravel/	Base directory for Unravel server data kept separately from installation directory; contains messaging data for process coordination, bundled db, Elasticsearch indexes, temporary files	2-900GB ; depending on activity level, retention	This directory or subdirectories can be a symlinks to other volumes for disk io performance reasons to distribute load over multiple volumes. If this is an EBS on AWS, then it must be provisioned for max available IOPS and the Unravel server must be EBS optimized.
/etc/init.d/unravel_all.sh	convenience script for Unravel start, restart, status, and stop ; controls daemons in proper order	n/a	Note for multi-host installations: run this script on both primary and secondary Unravel host
/usr/local/unravel/etc/unravel.properties	site-specific settings for Unravel	n/a	Keep a "golden" copy of this file; `rpm -U` upgrades will not overwrite
/usr/local/unravel/etc/unravel.version.properties	version-specific values for Unravel like version number, build timestamp	n/a	Updated during upgrades; do not modify this reference file to preserve traceability
/usr/local/unravel/etc/unravel.ext.sh	an optional file for overriding JAVA_HOME or other settings as shown in table above	n/a	Optional; example syntax: `export JAVA_HOME=/path`
/srv/unravel/log_hdfs	log directory for daemons that run as user hdfs (for YARN, when applicable)	<2GB	Needed for YARN. Owned by user hdfs; not applicable for MRv1 or EMR
/srv/unravel/tmp_hdfs	tmp directory for daemons that run as user hdfs (for YARN, when applicable)	<1GB	Needed for YARN. Owned by user hdfs; not applicable for MRv1 or EMR

Attachments:

worddav95f1e8fd13958fae391afc8e6f9ad03d.png

logo_small.png

Chapter 5

Advanced Topics

Topics:

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1

Creating Active Directory Kerberos Principals and Keytabs for Unravel

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics

Define HOST Variable for Unravel Server as an FQDN

(Replace `UNRAVEL_HOST` with your host's FQDN):

```
HOST=$UNRAVEL_HOST
```

Define REALM Variable

(Use upper case for all; replace `EXAMPLEDOTCOM` with your realm):

```
REALM=$EXAMPLEDOTCOM
```

Create the Active Directory (AD) Kerberos Principals and Keytabs

Use the two variables you defined above to replace the magenta text below.

1. Verify that Unravel Server host is running ntpd service and that time is accurate.
2. For proper Kerberos operation with AD-KDC, DNS entries, including reverse DNS entries, must be in place.
3. On AD server, logged in as AD Administrator, add 2 Managed Service Accounts `unravel` and `hdfs`:
   a. Open the **Active Directory Users and Computers** snap-in.
   b. Confirm that the **Managed Service Account** container exists under the target `REALM`.
   c. Right-click the **Managed Service Account** container and choose **New->User**.
   d. Set names (`unravel` and `hdfs`) to account in first screen and click **Next**.
   e. Set a strong password to account (the password will not be used) and
      1. Check **Password never expires**.
      2. **UN**check **Password must be changed**.
      3. Check **Password cannot be changed**.
   f. Right-click the created user, choose **Properties**, and select the **Account** tab.
   g. In the **Account Options** panel, check **Kerberos AES256-SHA1**.
4. On AD server, logged in as AD Administrator, create the Service Principal Names:
   a. The commands to run in a cmd or powershell are the following:
   b. `setspn -A unravel/ HOST unravel`
   c. `setspn -A hdfs/ HOST hdfs`
5. On AD server, logged in as AD Administrator, generate keytab files that Unravel Server will use to authenticate with Kerberos using the `ktpass` utility in Active Directory:
   a. `ktpass -princ unravel/ HOST @ REALM -mapUser unravel -Target REALM +rndPass -out unravel.keytab -ptype KRB5_NT_PRINCIPAL -crypto AES256-SHA1`
   b. `ktpass -princ hdfs/HOST@REALM -mapUser hdfs -Target REALM +rndPass -out hdfs.keytab -ptype KRB5_NT_PRINCIPAL -crypto AES256-SHA1`
6. Copy the two keytabs (`unravel.keytab` and `hdfs.keytab`) from AD server to the Unravel Server at `HOST` into `/etc/keytabs/` (create the destination directory if need be) and
   a. `sudo chmod 700 /etc/keytabs/*`
   b. `sudo chown unravel:unravel /etc/keytabs/unravel.keytab`
   c. `sudo chown hdfs:hdfs /etc/keytabs/hdfs.keytab`

Assurances: `hdfs.keytab` is only usable on Unravel Server and is only used to access HDFS log files and Hive metastore (if applicable).

Attachments:

worddav95f1e8fd13958fae391afc8e6f9ad03d.png (image/png)

logo_small.png (image/png)

Sensors

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics

Installing Unravel Sensor for Individual Applications Submitted Through spark-shell

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics
4. Sensors

Introduction

This topic explains how to set up Unravel Sensor for Spark to profile **specific Spark applications** executed via the **spark-shell** (in other words, *per-application profiling*, also called "dev mode"), rather than *cluster-wide profiling*. The Unravel Sensor for Spark uses Java agents to intercept Spark applications at runtime. Four JARs are included in the Unravel Sensor .zip package: `btrace-agent.jar`, `btrace-boot.jar`, `unravel-boot.jar, and unravel-sys.jar`. The JARs are fired up when you define `spark.driver.extraJavaOptions` and `spark.executor.extraJavaOptions` as part of your `spark-submit` command.

The information here applies to Spark versions 1.3.x through 2.0.x. Brown text indicates where you must substitute your particular values.

1. Obtain the Sensor

The Unravel Sensor is included in the Unravel Server RPM installation. After installing the Unravel Server RPM on `UNRAVEL_HOST` , you can obtain the sensor either from the filesystem on the Unravel Server host, or by HTTP from `http://UNRAVEL_HOST:3000/hh/spark-VERSION/unravel-sensor-for-spark-bin.zip`.

To obtain Unravel Sensor from the filesystem on the Unravel Server host:

- Go to the `/usr/local/unravel/webapps/ROOT/hh/spark-VERSION/` directory on the Unravel Server host, where `VERSION` is
  - `2.0` for Spark 2.0.x
  - `1.6` for Spark 1.6.x
  - `1.5` for Spark 1.5.x
  - `1.3` for Spark 1.3.x
- Within this directory, locate the sensor file: `unravel-sensor-for-spark-bin.zip`.

2. Run the Sensor to Intercept Spark Apps executed via the spark-shell

To intercept Spark apps executed via the spark-shell, you need to unzip the Unravel Sensor .zip file on the client node at a location on the local file system that is readable by all users, referred to as `UNZIPPED_ARCHIVE_DEST` below. We suggest `/usr/local/unravel-spark` .

If you use multiple hosts as clients, do this step for **each client**.

```
mkdir $UNZIPPED_ARCHIVE_DEST
cd $UNZIPPED_ARCHIVE_DEST
wget http://$UNRAVEL_HOST:3000/hh/spark-VERSION/unravel-sensor-for-spark-
bin.zip
```

```
unzip unravel-sensor-for-spark-bin.zip
```

Important

Please keep the original `unravel-sensor-for-spark-bin.zip` file inside `UNZIPPED_ARCHIVE_DEST`.

Define `spark.driver.extraJavaOptions` and `spark.executor.extraJavaOptions` as part of your `spark-shell` command. Please note that spark-shell is always executed in client mode, so a deployment mode setting is not available.

To use the example below, be sure to replace `UNRAVEL_SENSOR_PATH`, `UNRAVEL_SERVER_IP_PORT`, `SPARK_EVENT_LOG_DIR`, and `PATH_TO_SPARK_EXAMPLE_JAR` with your values.

- `UNRAVEL_SENSOR_PATH` : Parent directory of the Unravel Sensor .zip file, `unravel-sensor-for-spark-bin.zip`. If you put this file on HDFS, `UNRAVEL_SENSOR_PATH` is the parent directory on HDFS.
- `UNRAVEL_SERVER_IP_PORT` : IP address and port of the `unravel_lr` service in the format `IP:PORT`. Port is 4043 by default. Sample value: `10.0.0.142:4043`.
- `SPARK_EVENT_LOG_DIR` : Location of the event log directory on HDFS, S3, or local file system. If a remote address is used, include the namenode IP address and port.
- `UNZIPPED_ARCHIVE_DEST` : Directory on the local file system that contains the unzipped content of `unravel-sensor-for-spark-bin.zip`. It also contains the original `unravel-sensor-for-spark-bin.zip` file.

For example,

```
export UNZIPPED_ARCHIVE_DEST=$UNZIPPED_ARCHIVE_DEST
export UNRAVEL_SERVER_IP_PORT=$UNRAVEL_SERVER_IP_PORT
export SPARK_EVENT_LOG_DIR=$SPARK_EVENT_LOG_DIR

export ENABLED_SENSOR_FOR_DRIVER="spark.driver.extraJavaOptions=-
javaagent:$UNZIPPED_ARCHIVE_DEST/btrace-agent.jar=bootClassPath=
$UNZIPPED_ARCHIVE_DEST/unravel-boot.jar,systemClassPath=
$UNZIPPED_ARCHIVE_DEST/unravel-sys.jar,scriptOutputFile=/
dev/null,script=DriverProbe.class:SQLProbe.class -
Dcom.sun.btrace.FileClient.flush=-1 -
Dcom.unraveldata.spark.sensor.disableLiveUpdates=true"


export ENABLED_SENSOR_FOR_EXECUTOR="spark.executor.extraJavaOptions=-
javaagent:unravel-sensor-for-spark-bin.zip/btrace-
agent.jar=bootClassPath=unravel-sensor-for-spark-bin.zip/unravel-
boot.jar,systemClassPath=unravel-sensor-for-spark-bin.zip/unravel-
sys.jar,scriptOutputFile=/dev/null,script=ExecutorProbe.class  -
Dcom.sun.btrace.FileClient.flush=-1"



/usr/lib/spark/bin/spark-shell \
    --archives $UNZIPPED_ARCHIVE_DEST/unravel-sensor-for-spark-bin.zip \
    --conf "$ENABLED_SENSOR_FOR_DRIVER" \
    --conf "$ENABLED_SENSOR_FOR_EXECUTOR" \
    --conf "spark.unravel.server.hostport=$UNRAVEL_SERVER_IP_PORT" \
    --conf "spark.eventLog.dir=${SPARK_EVENT_LOG_DIR}" \
    --conf "spark.eventLog.enabled=true" \
<<EOF
// your spark-shell code snippet will follow here
// For exemplifying, we use a snippet of RDDRelation below

import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.sql.SQLContext
```

```
import org.apache.spark.sql.functions._

import sqlContext.implicits._

case class Record(key: Int, value: String)

    val df = sc.parallelize((1 to 100).map(i => Record(i, s"val_
$i"))).toDF()

    // Any RDD containing case classes can be registered as a table.  The
 schema of the table is
    // automatically inferred using scala reflection.
    df.registerTempTable("records")

    // Once tables have been registered, you can run SQL queries over them.
    println("Result of SELECT *:")
    sqlContext.sql("SELECT * FROM records").collect().foreach(println)

    // Aggregation queries are also supported.
    val count = sqlContext.sql("SELECT COUNT(*) FROM
 records").collect().head.getLong(0)
    println(s"COUNT(*): $count")

exit

EOF
```

Note that a full blank line separates lengthy lines that are wrapped, except for the spark-shell that uses line continuation backslashes.

Attachments:

- worddav95f1e8fd13958fae391afc8e6f9ad03d.png (image/png)

Installing Unravel Sensor for Individual Applications Submitted Through spark-submit

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics
4. Sensors

Introduction

This topic explains how to set up Unravel Sensor for Spark to profile **specific Spark applications** only (in other words, *per-application profiling*, also called "dev mode"), rather than *cluster-wide profiling*. The Unravel Sensor for Spark uses Java agents to intercept Spark applications at runtime. Four JARs are included in the Unravel Sensor .zip package: `btrace-agent.jar`, `btrace-boot.jar`, `unravel-boot.jar, and unravel-sys.jar`. The JARs are fired up when you define `spark.driver.extraJavaOptions` and `spark.executor.extraJavaOptions` as part of your `spark-submit` command.

The information here applies to Spark versions 1.3.x through 2.0.x. Brown text indicates where you must substitute your particular values.

1. Obtain the Sensor

The Unravel Sensor is included in the Unravel Server RPM installation. After installing the Unravel Server RPM on `UNRAVEL_HOST` , you can obtain the sensor either from the filesystem on the Unravel Server host, or by HTTP from `http://UNRAVEL_HOST:3000/hh/spark-VERSION/unravel-sensor-for-spark-bin.zip`.

To obtain Unravel Sensor from the filesystem on the Unravel Server host:

- Go to the `/usr/local/unravel/webapps/ROOT/hh/spark-VERSION/` directory on the Unravel Server host, where `VERSION` is
  - `2.0` for Spark 2.0.x
  - `1.6` for Spark 1.6.x
  - `1.5` for Spark 1.5.x
  - `1.3` for Spark 1.3.x
- Within this directory, locate the sensor file: `unravel-sensor-for-spark-bin.zip`.

2. Run the Sensor to Intercept Spark Apps

Option A: If You Run Spark Apps in `yarn-cluster` Mode (Default)

Put the sensor on the host node(s) from which you will run `spark-submit` by first creating a destination directory that is readable by all users.

We suggest that `UNRAVEL_SENSOR_PATH` be `/usr/local/unravel-spark`.

- If `spark-submit` is used from a single client node:

```
mkdir $UNRAVEL_SENSOR_PATH
cd $UNRAVEL_SENSOR_PATH
wget http://$UNRAVEL_HOST:3000/hh/spark-$VERSION/unravel-sensor-for-spark-
bin.zip
```

- If `spark-submit` is used from multiple client nodes, copy the sensor .zip file to HDFS instead of copying it to every client node, and set `UNRAVEL_SENSOR_PATH` accordingly. For example, copy it to `hdfs:///tmp`:

```
mkdir $UNRAVEL_SENSOR_PATH
cd $UNRAVEL_SENSOR_PATH
wget http://$UNRAVEL_HOST:3000/hh/spark-$VERSION/unravel-sensor-for-spark-
bin.zip
cd $UNRAVEL_SENSOR_PATH
hdfs fs -copyFromLocal unravel-sensor-for-spark-bin.zip /tmp
set UNRAVEL_SENSOR_PATH="hdfs:///tmp"
```

Define `spark.driver.extraJavaOptions` and `spark.executor.extraJavaOptions` as part of your `spark-submit` command.

To use the example below, be sure to replace `UNRAVEL_SENSOR_PATH`, `UNRAVEL_SERVER_IP_PORT`, `SPARK_EVENT_LOG_DIR`, and `PATH_TO_SPARK_EXAMPLE_JAR` with your values.

- `UNRAVEL_SENSOR_PATH`: Parent directory of the Unravel Sensor .zip file, `unravel-sensor-for-spark-bin.zip`. If you put this file on HDFS, `UNRAVEL_SENSOR_PATH` is the parent directory on HDFS.
- `UNRAVEL_SERVER_IP_PORT`: IP address and port of the `unravel_lr` service in the format `IP:PORT`. Port is 4043 by default. Sample value: `10.0.0.142:4043`.
- `SPARK_EVENT_LOG_DIR`: Location of the event log directory on HDFS, S3, or local file system. If a remote address is used, include the namenode IP address and port.
- `PATH_TO_SPARK_EXAMPLE_JAR`: **Absolute** path to the jar file used in the `spark-submit` command.

For example,

```
export UNRAVEL_SENSOR_PATH=$UNRAVEL_SENSOR_PATH
export UNRAVEL_SERVER_IP_PORT=$UNRAVEL_SERVER_IP_PORT
export SPARK_EVENT_LOG_DIR=$SPARK_EVENT_LOG_DIR
export PATH_TO_SPARK_EXAMPLE_JAR=$PATH_TO_SPARK_EXAMPLE_JAR

export ENABLED_SENSOR_FOR_DRIVER="spark.driver.extraJavaOptions=-
javaagent:unravel-sensor-for-spark-bin.zip/btrace-
agent.jar=bootClassPath=unravel-sensor-for-spark-bin.zip/unravel-
boot.jar,systemClassPath=unravel-sensor-for-spark-bin.zip/unravel-
sys.jar,scriptOutputFile=/dev/null,script=DriverProbe.class:SQLProbe.class -
```

```
Dcom.sun.btrace.FileClient.flush=-1 -
Dcom.unraveldata.spark.sensor.disableLiveUpdates=true"


export ENABLED_SENSOR_FOR_EXECUTOR="spark.executor.extraJavaOptions=-
javaagent:unravel-sensor-for-spark-bin.zip/btrace-
agent.jar=bootClassPath=unravel-sensor-for-spark-bin.zip/unravel-
boot.jar,systemClassPath=unravel-sensor-for-spark-bin.zip/unravel-
sys.jar,scriptOutputFile=/dev/null,script=ExecutorProbe.class  -
Dcom.sun.btrace.FileClient.flush=-1"


spark-submit \
    --class org.apache.spark.examples.sql.RDDRelation \
    --master yarn-cluster \
    --archives $UNRAVEL_SENSOR_PATH/unravel-sensor-for-spark-bin.zip \
    --conf "$ENABLED_SENSOR_FOR_DRIVER" \
    --conf "$ENABLED_SENSOR_FOR_EXECUTOR" \
    --conf "spark.unravel.server.hostport=$UNRAVEL_SERVER_IP_PORT" \
    --conf "spark.eventLog.dir=${SPARK_EVENT_LOG_DIR}" \
    --conf "spark.eventLog.enabled=true" \
    $PATH_TO_SPARK_EXAMPLE_JAR
```

Note that a full blank line separates lengthy lines that are wrapped, except for the spark-submit that uses line continuation backslashes.

Option B: If You Run Spark Apps in `yarn-client` Mode

To intercept Spark apps running in `yarn-client` mode, you need to unzip the Unravel Sensor .zip file on the client node at a location on the local file system that is readable by all users, referred to as `UNZIPPED_ARCHIVE_DEST` below. We suggest `/usr/local/unravel-spark`.

If you use multiple hosts as clients, do this step for **each client**.

```
mkdir $UNZIPPED_ARCHIVE_DEST
cd $UNZIPPED_ARCHIVE_DEST
wget http://$UNRAVEL_HOST:3000/hh/spark-VERSION/unravel-sensor-for-spark-
bin.zip
unzip unravel-sensor-for-spark-bin.zip
```

Important

Please keep the original `unravel-sensor-for-spark-bin.zip` file inside `UNZIPPED_ARCHIVE_DEST`.

Define `spark.driver.extraJavaOptions` and `spark.executor.extraJavaOptions` as part of your `spark-submit` command.

To use the example below, be sure to replace `UNRAVEL_SENSOR_PATH`, `UNRAVEL_SERVER_IP_PORT`, `SPARK_EVENT_LOG_DIR`, and `PATH_TO_SPARK_EXAMPLE_JAR` with your values.

- `UNRAVEL_SENSOR_PATH` : Parent directory of the Unravel Sensor .zip file, `unravel-sensor-for-spark-bin.zip`. If you put this file on HDFS, `UNRAVEL_SENSOR_PATH` is the parent directory on HDFS.
- `UNRAVEL_SERVER_IP_PORT` : IP address and port of the `unravel_lr` service in the format `IP:PORT`. Port is 4043 by default. Sample value: `10.0.0.142:4043`.
- `SPARK_EVENT_LOG_DIR` : Location of the event log directory on HDFS, S3, or local file system. If a remote address is used, include the namenode IP address and port.
- `PATH_TO_SPARK_EXAMPLE_JAR` : **Absolute** path to the jar file used in the `spark-submit` command. Sample value: `spark-examples.jar`.

- `UNZIPPED_ARCHIVE_DEST` : Directory on the local file system that contains the unzipped content of `unravel-sensor-for-spark-bin.zip`. It also contains the original `unravel-sensor-for-spark-bin.zip` file.

For example,

```
export UNZIPPED_ARCHIVE_DEST=$UNZIPPED_ARCHIVE_DEST
export UNRAVEL_SERVER_IP_PORT=$UNRAVEL_SERVER_IP_PORT
export SPARK_EVENT_LOG_DIR=$SPARK_EVENT_LOG_DIR
export PATH_TO_SPARK_EXAMPLE_JAR=$PATH_TO_SPARK_EXAMPLE_JAR

export ENABLED_SENSOR_FOR_DRIVER="spark.driver.extraJavaOptions=-
javaagent:$UNZIPPED_ARCHIVE_DEST/btrace-agent.jar=bootClassPath=
$UNZIPPED_ARCHIVE_DEST/unravel-boot.jar,systemClassPath=
$UNZIPPED_ARCHIVE_DEST/unravel-sys.jar,scriptOutputFile=/
dev/null,script=DriverProbe.class:SQLProbe.class -
Dcom.sun.btrace.FileClient.flush=-1 -
Dcom.unraveldata.spark.sensor.disableLiveUpdates=true"


export ENABLED_SENSOR_FOR_EXECUTOR="spark.executor.extraJavaOptions=-
javaagent:unravel-sensor-for-spark-bin.zip/btrace-
agent.jar=bootClassPath=unravel-sensor-for-spark-bin.zip/unravel-
boot.jar,systemClassPath=unravel-sensor-for-spark-bin.zip/unravel-
sys.jar,scriptOutputFile=/dev/null,script=ExecutorProbe.class  -
Dcom.sun.btrace.FileClient.flush=-1"


spark-submit \
    --class org.apache.spark.examples.sql.RDDRelation \
    --master yarn-client \
    --archives $UNZIPPED_ARCHIVE_DEST/unravel-sensor-for-spark-bin.zip \
    --conf "$ENABLED_SENSOR_FOR_DRIVER" \
    --conf "$ENABLED_SENSOR_FOR_EXECUTOR" \
    --conf "spark.unravel.server.hostport=$UNRAVEL_SERVER_IP_PORT" \
    --conf "spark.eventLog.dir=${SPARK_EVENT_LOG_DIR}" \
    --conf "spark.eventLog.enabled=true" \
    $PATH_TO_SPARK_EXAMPLE_JAR
```

Note that a full blank line separates lengthy lines that are wrapped, except for the spark-submit that uses line continuation backslashes.

Attachments:

worddav95f1e8fd13958fae391afc8e6f9ad03d.png (image/png)

Installing Unravel Sensor for Individual Hive Queries

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics
4. Sensors

The MapReduce JVM sensor is a prepackaged distribution of JVM agent which enables collection of additional information, including resource usage metrics. The sensor binary is distributed as `unravel-sensor-for-mapreduce-bin.zip` . This archive contains the following 4 jars:

- `btrace-agent.jar`
- `btrace-boot.jar`
- `unravel-mr-boot.jar`
- `unravel-mr-sys.jar`

When you enable this sensor, it uses the standard MapReduce profiling extension. You can copy and paste the following configuration snippets for a quick bootstrap:

Option A: Installing the MapReduce JVM Sensor on Cloudera

1. Enable profiling:

```
set mapreduce.task.profile=true;
```

2. Select map and reduce profiles:

```
set mapreduce.task.profile.maps=0-5;
set mapreduce.task.profile.reduces=0-5;
```

3. Enable the JVM agent for map and reduce tasks:

   Set UNRAVEL_HOST_IP of your Unravel gateway server. Port #4043 is where Unravel LR server is running.

```
set
mapreduce.task.profile.params=-javaagent:/opt/cloudera/parcels/
UNRAVEL_SENSOR/lib/java/btrace-agent.jar=systemClassPath=/
opt/cloudera/parcels/UNRAVEL_SENSOR/lib/java/unravel-mr-
sys.jar,bootClassPath=/opt/cloudera/parcels/UNRAVEL_SENSOR/
lib/java/unravel-mr-boot.jar,scriptOutputFile=/dev/null
 -Dunravel.server.hostport=<UNRAVEL_HOST_IP>:4043 -
Dcom.sun.btrace.FileClient.flush=-1;
```

4. Enable the JVM agent for application master:

```
set
yarn.app.mapreduce.am.command-opts=-javaagent:/opt/cloudera/
parcels/UNRAVEL_SENSOR/lib/java/btrace-agent.jar=systemClassPath=/
opt/cloudera/parcels/UNRAVEL_SENSOR/lib/java/unravel-mr-
sys.jar,bootClassPath=/opt/cloudera/parcels/UNRAVEL_SENSOR/
lib/java/unravel-mr-boot.jar,scriptOutputFile=/dev/
null -Dunravel.server.hostport=UNRAVEL_HOST_IP:4043 -
Dcom.sun.btrace.FileClient.flush=-1;
```

Option B: Installing the MapReduce JVM Sensor on Cloudera Manager for Hive

See *Optional - Configure YARN - MapReduce (MR) JVM Sensor Cluster-Wide* in Part 2: Install Unravel Sensor Parcel on CDH+CM.

Option C: Installing the MapReduce JVM Sensor on HDFS

Change your `*init` file as follows:

1. Set the path to the JVM sensor archive:

```
set mapreduce.job.cache.archives=path_in_hdfs/unravel-sensor-for-
mapreduce-bin.zip;
```

2. Enable profiling:

```
set mapreduce.task.profile=true;
```

3. Select map and reduce profiles:

```
set mapreduce.task.profile.maps=0-5;
set mapreduce.task.profile.reduces=0-5;
```

4. Enable the JVM agent for map and reduce tasks:

Set UNRAVEL_HOST_IP of your Unravel gateway server. Port #4043 is where Unravel LR server is running.

```
set
mapreduce.task.profile.params=-javaagent:unravel-sensor-for-
mapreduce-bin.zip/btrace-agent.jar=systemClassPath=unravel-sensor-
for-mapreduce-bin.zip/unravel-mr-sys.jar,bootClassPath=unravel-
sensor-for-mapreduce-bin.zip/unravel-mr-boot.jar,scriptOutputFile=/
dev/null -Dunravel.server.hostport=<UNRAVEL_HOST_IP>:4043 -
Dcom.sun.btrace.FileClient.flush=-1;
```

5. Enable the JVM agent for application master:

```
set yarn.app.mapreduce.am.command-opts=-javaagent:unravel-sensor-for-
mapreduce-bin.zip/btrace-agent.jar=systemClassPath=unravel-sensor-
for-mapreduce-bin.zip/unravel-mr-sys.jar,bootClassPath=unravel-
sensor-for-mapreduce-bin.zip/unravel-mr-boot.jar,scriptOutputFile=/
dev/null -Dunravel.server.hostport=<UNRAVEL_HOST_IP>:4043 -
Dcom.sun.btrace.FileClient.flush=-1;
```

Option D: Installing the MapReduce JVM Sensor on the Local File System

1. Add the JVM sensor archive to the PATH environment variable.
2. Enable profiling:

```
set mapreduce.task.profile=true;
```

3. Select map and reduce profiles:

```
set mapreduce.task.profile.maps=0-5;set
 mapreduce.task.profile.reduces=0-5;
```

4. Enable the JVM agent for map and reduce tasks:

   Set UNRAVEL_HOST_IP of your Unravel gateway server. Port #4043 is where Unravel LR server is running.

```
set mapreduce.task.profile.params=-javaagent:unravel-sensor-for-
mapreduce-bin.zip/btrace-agent.jar=systemClassPath=unravel-sensor-
for-mapreduce-bin.zip/unravel-mr-sys.jar,bootClassPath=unravel-
sensor-for-mapreduce-bin.zip/unravel-mr-boot.jar,scriptOutputFile=/
dev/null -Dunravel.server.hostport=<UNRAVEL_HOST_IP>:4043 -
Dcom.sun.btrace.FileClient.flush=-1;
```

5. Enable the JVM agent for application master.

```
set yarn.app.mapreduce.am.command-opts=-javaagent:unravel-sensor-for-
mapreduce-bin.zip/btrace-agent.jar=systemClassPath=unravel-sensor-
for-mapreduce-bin.zip/unravel-mr-sys.jar,bootClassPath=unravel-
sensor-for-mapreduce-bin.zip/unravel-mr-boot.jar,scriptOutputFile=/
dev/null -Dunravel.server.hostport=<UNRAVEL_HOST_IP>:4043 -
Dcom.sun.btrace.FileClient.flush=-1;
```

Attachments:

- logo_small.png (image/png)

Workflows

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1

3. Advanced Topics

Monitoring Airflow Workflows

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics
4. Workflows

This article describes how to set up Unravel Server to monitor Airflow workflows so that you can see them in Unravel Web UI.

All the following steps are on the Unravel Server host that runs the `unravel_jcs2` daemon. Highlighted text indicates where you need to substitute your particular values.

Before you start, ensure that the Unravel Server host and the server that runs Airflow web service are in the same cluster.

If You Use Http For Airflow Web UI Access

Add the following 3 settings to `/usr/local/unravel/etc/unravel.properties`:

Replace {airflow web url} with the full URL, starting with `http://` .

```
com.unraveldata.airflow.protocol=http
com.unraveldata.airflow.server.url={airflow web url}
com.unraveldata.airflow.available=true
```

Then restart the `unravel_jcs2` daemon:

```
sudo /etc/init.d/unravel_jcs2 restart
```

If You Use Https For Airflow Web UI Access

Add the following 4 settings to `/usr/local/unravel/etc/unravel.properties`:

Replace {airflow web url} with the full URL, starting with `http://` .

```
com.unraveldata.airflow.server.url={airflow web url}
com.unraveldata.airflow.available=true
com.unraveldata.airflow.login.name={airflow web UI username}
com.unraveldata.airflow.login.password={airflow web UI password}
```

Then restart the `unravel_jcs2` daemon:

```
sudo /etc/init.d/unravel_jcs2 restart
```

Changing the Range of Monitoring

By default, Unravel Server ingests all the workflows that started within the last 5 days. If you wish to change the date range to the last *x* days, do the following:

1. Add the following configuration to `/usr/local/unravel/etc/unravel.properties`. Note there's a "-" (minus sign) in the value.

   ```
   airflow.look.back.num.days=-x
   ```

2. Restart the `unravel_jcs2` daemon:

   ```
   sudo /etc/init.d/unravel_jcs2 restart
   ```

Attachments:

logo_small.png (image/png)

Monitoring Oozie Workflows

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics
4. Workflows

Enabling Oozie

In `unravel.properties` , update the `oozie.server.url` property with appropriate value.

```
sudo vi /usr/local/unravel/etc/unravel.properties
```

Enabling AirFlow

The script below, `spark-test.py`, is a sample script for Spark.

spark-test.py

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.operators import PythonOperator
from datetime import datetime, timedelta
import subprocess


default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2015, 6, 1),
    'email': ['airflow@airflow.com'],
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
    # 'queue': 'bash_queue',
    # 'pool': 'backfill',
    # 'priority_weight': 10,
    # 'end_date': datetime(2016, 1, 1),
}
```

In Oozie/Airflow, workflows are represented by directed acyclic graphs (DAGs). For example,

```
dag = DAG('spark-test', default_args=default_args)
```

1. Add Hooks for Unravel Instrumentation

The example below shows the contents of a bash script, `example-hdp-client.sh`, that can be invoked to submit an Airflow Spark application via `spark-submit`. This script adds hooks for Unravel instrumentation by setting three Unravel-specific configuration parameters for Spark applications:

- `spark.driver.extraJavaOptions`
- `spark.executor.extraJavaOptions`
- `spark.unravel.server.hostport`

Setting these parameters on a **per-application** basis is recommended when you only want to monitor/profile certain applications, rather than all the applications running in the cluster. Alternatively, you can specify these parameters in `spark-defaults.conf`.

This script references the following variables, which you would need to edit:

- `PATH_TO_SPARK_EXAMPLE_JAR=/usr/hdp/2.3.6.0-3796/spark/lib/spark-examples-*.jar`
- `UNRAVEL_SERVER_IP_PORT=10.20.30.40:4043`
- `SPARK_EVENT_LOG_DIR=hdfs://ip-10-0-0-21.ec2.internal:8020/user/ec2-user/eventlog`

example-hdp-client.sh

```
hdfs dfs -rmr pair.parquet
spark-submit \
--class org.apache.spark.examples.sql.RDDRelation \
--master yarn-cluster \
--conf "spark.driver.extraJavaOptions=-javaagent:/usr/local/unravel_client/
btrace-
agent.jar=unsafe=true,stdout=false,noServer=true,startupRetransform=false,bootClassPath=
usr/local/unravel_client/unravel-boot.jar,systemClassPath=/
usr/local/unravel_client/unravel-sys.jar,scriptOutputFile=/
dev/null,script=DriverProbe.class:SQLProbe.class -
Dcom.sun.btrace.FileClient.flush=-1 -
Dcom.unraveldata.spark.sensor.disableLiveUpdates=true" \
--conf "spark.executor.extraJavaOptions=-javaagent:/usr/local/
unravel_client/btrace-
agent.jar=unsafe=true,stdout=false,noServer=true,startupRetransform=false,bootClassPath=
usr/local/unravel_client/unravel-boot.jar,systemClassPath=/
usr/local/unravel_client/unravel-sys.jar,scriptOutputFile=/dev/
null,script=ExecutorProbe.class -Dcom.sun.btrace.FileClient.flush=-1" \
--conf "spark.unravel.server.hostport=$UNRAVEL_SERVER_IP_PORT" \
--conf "spark.eventLog.dir=$SPARK_EVENT_LOG_DIR" \
--conf "spark.eventLog.enabled=true" \
$PATH_TO_SPARK_EXAMPLE_JAR
```

2. Submit the Workflow

Operators (also called tasks) determine execution order (dependencies). In the example below, `t1` and `t2` are operators created by `BashOperator` or `PythonOperator`. They invoke `example-hdp-client.sh`, which submits the workflow for execution.

Note: The pathname of `example-hdp-client.sh` is relative to the current directory, not to `~/airflow/dags` as in the Airflow operator above!

```
t1 = BashOperator(
task_id='example-hdp-client',
bash_command="example-scripts/example-hdp-client.sh",
retries=3,
dag=dag)


def spark_callback(**kwargs):
sp
 = subprocess.Popen(['/bin/bash',
'airflow/dags/example-scripts/example-hdp-client.sh'],
stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
print sp.stdout.read()


t2 = PythonOperator(
task_id='example-python-call',
```

```
provide_context=True,
python_callable=spark_callback,
retries=1,
dag=dag)


t2.set_upstream(t1)
```

You can test the operators first. For example, in Airflow:

```
airflow test spark-test example-python-call 2016-08-30
```

3. Monitor the Workflow

To see the new Oozie workflow in Unravel Web UI, select **APPLICATIONS** | **Workflows**, and then click **Add Workflow**.

Attachments:

logo_small.png (image/png)

Tagging Workflows

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics
4. Workflows

About Unravel Tags

You can add two Unravel tags (key-value pairs) to mark queries and jobs that belong to a particular workflow:

- `unravel.workflow.name`: a string that represents the name of the workflow. Recommended format is *`TenantName-ProjectName-WorkflowName`* .
- `unravel.workflow.utctimestamp`: a timestamp in *`yyyyMMddThhmmssZ`* format that represents the logical time of a run of the workflow in UTC/ISO format. In UNIX/LINUX bash, you can get a timestamp in UTC format by running the command "`$(date -u '+%Y%m%dT%H%M%SZ')`".

  Do not put quotes ("") or blank spaces in/around the tag keys or values. For example:

  - `SET unravel.workflow.name="ETL-Workflow";` [Wrong usage]
  - `SET unravel.workflow.name=ETL-Workflow;` [Correct usage]

Please note the following:

- Different runs of the **same** workflow have the **same** value for `unravel.workflow.name`.
- Different runs of the **same** workflow have **different** values for `unravel.workflow.utctimestamp`.
- Different workflows have different values for `unravel.workflow.name`.

The example below shows a Hive query that is marked as part of the `Financial-Tenant-ETL-Workflow` workflow that was run on February 1, 2016:

```
SET unravel.workflow.name=Financial-Tenant-ETL-Workflow;
SET unravel.workflow.utctimestamp=20160201T000000Z;

SELECT foo FROM table WHERE … [Rest of Hive Query text goes here]
```

Easy Recipes for Tagging Workflows

First, export the workflow name and UTC timestamp from your top-level script that schedules each run of the workflow:

Export the workflow name:

```
export WORKFLOW_NAME=Financial-Tenant-ETL-Workflow
```

Export the UTC timestamp for this run of the workflow. Here, we use bash's `date` command:

```
export UTC_TIME_STAMP=$(date -u '+%Y%m%dT%H%M%SZ')
```

Then follow the appropriate instructions below:

- Tagging a Hive query
- Tagging a Sqoop job
- Tagging a direct MapReduce job
- Tagging a Spark job
- Tagging a Pig job
- Tagging a Impala query

How to Tag a Hive Query Using `SET` Commands in Hive

```
hive -f hive/simple_wf.hql
```

In `hive/simple_wf.hql`:

```
SET unravel.workflow.name=${env:WORKFLOW_NAME};
SET unravel.workflow.utctimestamp=${env:UTC_TIME_STAMP};
select count(1) from lineitem;
```

How to Tag a Sqoop Job Using `-D` Command Line Parameters

```
sqoop export \
 *-D"unravel.workflow.name=$WORKFLOW_NAME" -D"unravel.workflow.utctimestamp=
$UTC_TIME_STAMP" * \
 --connect jdbc:mysql://127.0.0.1:3316/unravel_mysql_prod --table settings -
m 1 \
 --export-dir /tmp/sqoop_test --username unravel --verbose --password foobar
```

How to Tag a Direct MapReduce Job Using `-D` Command Line Parameters

```
hadoop jar libs/ooziemr-1.0.jar com.unraveldata.mr.apps.Driver \
*-D"unravel.workflow.name=$WORKFLOW_NAME" -D"unravel.workflow.utctimestamp=
$UTC_TIME_STAMP" * \
-p /wordcount.properties -input /tmp/soumitra/data/small -output /tmp/
soumitra/outsmoke
```

How to Tag a Spark Job Using `--conf` Command Line Parameters

For Spark jobs, you must prefix the Unravel tags with "`spark.`". For example, `unravel.workflow.name` becomes `spark.unravel.workflow.name` .

```
spark-submit \
 *   --conf "spark.unravel.workflow.name=$WORKFLOW_NAME" *
 *   --conf "spark.unravel.workflow.utctimestamp=$UTC_TIME_STAMP" *
    --conf "spark.eventLog.enabled=true" \
    --class org.apache.spark.examples.SparkPi \
    --master yarn-cluster \
    --deploy-mode cluster
```

How to Tag a Pig Job Using `-param` and `SET` Commands

```
pig \
*-param WORKFLOW_NAME=$WORKFLOW_NAME -param UTC_TIME_STAMP=$UTC_TIME_STAMP *
 \
-x mapreduce -f pig/simple.pig
```

In `pig/simple.pig`:

```
SET unravel.workflow.name $WORKFLOW_NAME;
SET unravel.workflow.utctimestamp $UTC_TIME_STAMP;

lines = LOAD '/tmp/soumitra/data/small' using PigStorage('|') AS
 (line:chararray);
words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word;
grouped = GROUP words BY word;
wordcount = FOREACH grouped GENERATE group, COUNT(words);
DUMP wordcount;
```

How to Tag a Impala Job Using `--query_options` and `SET` Command

User can set DEBUG_ACTION in Impala script to pass the workflow tags into impala queries:

```
SET DEBUG_ACTION=|unravel.workflow.name::{$WORKFLOW_NAME}|
unravel.workflow.utctimestamp::{$WORKFLOW_TIMESTAMP};
```

According to this CDH article: https://www.cloudera.com/documentation/enterprise/5-14-x/topics/impala_query_options.html, user can pass in query options via --query_options in CDH 5.14 / Impala 2.11 and later:

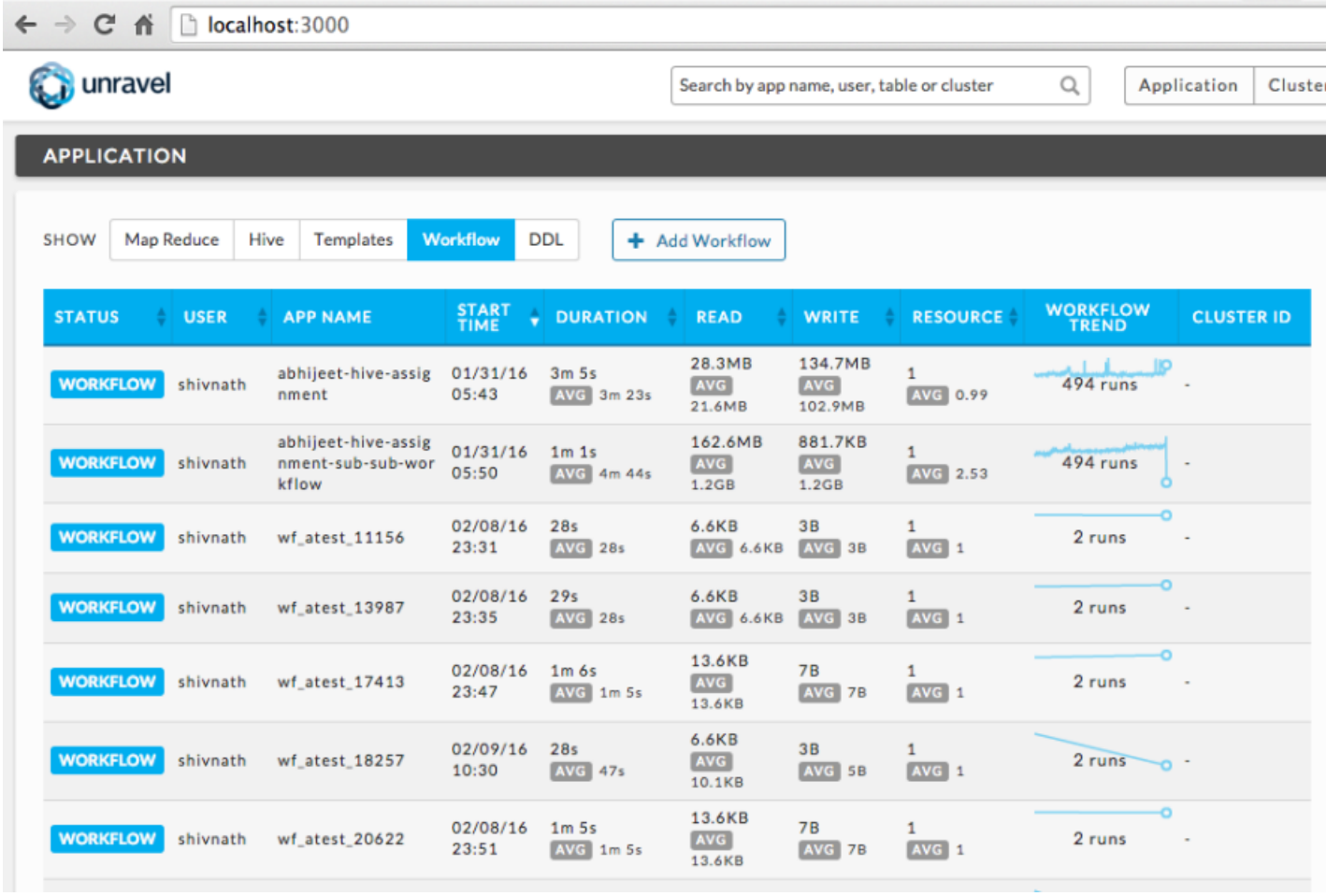
```
impala-shell --query_option=DEBUG_ACTION=|unravel.workflow.name::
{$WORKFLOW_NAME}|unravel.workflow.utctimestamp::{$WORKFLOW_TIMESTAMP}| -f
 impala.script
```

Note: the tagging option should be turned on for this tagging to work. ie, the followings need to be set in unravel.properties:

```
# Tagging
com.unraveldata.tagging.script.enabled=true
com.unraveldata.app.tagging.script.path=/tmp/app_tag.py
com.unraveldata.app.tagging.script.method.name=get_tags
```

Finding Workflows in Unravel Web UI

Once your tagged workflows have been run, go log into Unravel Web UI and select **Application** | **Workflow** to start exploring Unravel's Workflow Management features, as illustrated in the screenshot below.



Attachments:

- logo_small.png (image/png)
- worddav45f3dd6c1009f4ad8588eb86ede42f01.png (image/png)

Custom Configurations

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics

Adding More Admins to Unravel Web UI

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics
4. Custom Configurations

1. On Unravel Server, open `/usr/local/unravel/etc/unravel.properties` with `vi`.

```
sudo vi /usr/local/unravel/etc/unravel.properties
```

2. Append comma-separated usernames to the value of `com.unraveldata.login.admins`. For example,

```
com.unraveldata.login.admins=admin,admin1,admin2
```

Configuring Multiple Hosts for Unravel Server

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics
4. Custom Configurations

This topic explains how to configure multiple hosts for Unravel Server. This is useful for volume/throughput/reliability requirements.

Prerequisites

- If you use Kerberos, set that up first on **host1**.
- Install the same Unravel RPM on one or two additional hosts, hereafter referred to as **host2** and **host3**. This distinction is important because certain instructions only apply to specific hosts.

The internal DNS or IP address of a host is specific to your installation.

Each host is assigned unique roles identified by daemon names that start with `unravel_`.

Expected Result

When you complete the steps below, the expected result is

- Multiple Unravel hosts work together in an ensemble.
- Each host has a unique role, and is identified by a daemon named `unravel_xyz` or `unravel_xyz_n` (where *n* is 1, 2, 3,...), which runs on exactly one host.
- Unravel Web UI (`unravel_tc`) runs on **host1**.
- Port 4043 `unravel_lr` runs on **host2**.
- If you do not use an external database (db), `unravel_db` runs on **host1**. However, `unravel_db` can also run harmlessly if an external db is used.
- `/usr/local/unravel/etc/unravel.properties` is identical on all Unravel hosts in the ensemble.

  The file `unravel.properties` is never changed by an RPM upgrade because it contains site-specific information, but some one-time changes occur when setting up multiple hosts. The file `unravel.properties` must be identical on all hosts, so if you are using data center automation like Puppet, Chef, Salt, Ansible, CloudFormation, Cfengine, and so on, you must maintain one "golden" `unravel.properties` per Unravel ensemble. After you modify `unravel.properties` as described below, you must update the "golden" `unravel.properties` file for your site.
- Daemons are enabled/disabled via `chkconfig`. `chkconfig` sets symbolic links which are persistent across Unravel Server RPM upgrades.

1. Stop Unravel Server

On each Unravel host, run this command:

```
sudo /etc/init.d/unravel_all.sh stop
```

2. Modify `unravel.properties` on host1

1. Pick a machine to be **host1**, where the Unravel Web UI will run.
2. **If the bundled db is in use**, edit `/usr/local/unravel/etc/unravel.properties` on **host1** to change:

   Replace `127.0.0.1` with your LAN IP address or fully qualified hostname, replace `3316` with your port number, and replace `unravel_mysql_prod` with your database name.

To find your fully qualified hostname, type `hostname -I` at the OS prompt.

```
unravel.jdbc.url=jdbc:mysql://127.0.0.1:3316/unravel_mysql_prod
```

2. Copy host1's `unravel.properties` to Other Hosts

1. Copy `/usr/local/unravel/etc/unravel.properties` and `/usr/local/unravel/etc/unravel.ext.sh` from **host1** to **host2** (and **host3**, if you are using three hosts). For example,

```
# host1
scp /usr/local/unravel/etc/unravel.properties host2:/usr/local/unravel/
etc/
# host1
scp /usr/local/unravel/etc/unravel.ext.sh host2:/usr/local/unravel/etc/
```

2. Verify that the ownership of `unravel.properties` and `unravel.ext.sh` is `unravel:unravel`.

Important Note

The scripts invoked below will make an identical change to the `unravel.properties` file on each machine.

3. Assign Roles

Use these scripts to assign unique roles to the hosts. To reduce the chance of errors, the command line arguments are the same on each host, but notice that the script name is different. The arguments are the hostnames IP addresses of the hosts in the ensemble.

- For a 2-host ensemble (substitute host):

```
# host1
sudo /usr/local/unravel/install_bin/switch_to_1of2.sh\
host1 host2
# host2
sudo /usr/local/unravel/install_bin/switch_to_2of2.sh\
host1 host2
```

- For a 3-host ensemble (substitute host):

```
# host1 sudo /usr/local/unravel/install_bin/switch_to_1of3.sh\
host1 host2 host3
# host2 sudo /usr/local/unravel/install_bin/switch_to_2of3.sh\
host1 host2 host3
# host3 sudo /usr/local/unravel/install_bin/switch_to_3of3.sh\
host1 host2 host3
```

These scripts establish the roles each host plays in the ensemble. The main effect is to assign specific Unravel logical daemons to one host and only one host. Note that some daemons have names like `unravel_xyz_1` or `unravel_xyz_2`, and so on. The entire name with the instance numeric suffix is set to run on one host in the ensemble. In some cases, multiple *xyz* numeric instances run on one machine, but the overall name with suffix runs on one host only. The `switch_to_*` scripts change `unravel.properties` in a coordinated fashion and also create a file, `unravel.id.properties`, to hold the integer property indicating which role the particular machine has (1, 2, or 3).

4. Set Up Zookeeper and Kafka

Assign Kafka Partitions

Kafka partition assignment (for 3 host installs) is done by evenly distributing a topic over the hosts that exist at topic create time. Topics must be created anew when a new host is added in order to have proper distribution.

Redistribute Zookeeper Topics

Perform these steps, *in sequential order, on the specific hosts* as indicated by the prompt name. Skip **host3** if you are only using 2 hosts for Unravel Server:

- Stop all and clear Zookeeper and Kafka data areas on each host:

```
# host1
sudo /usr/local/unravel/install_bin/kafka_clear.sh
# host2
sudo /usr/local/unravel/install_bin/kafka_clear.sh
# host3
sudo /usr/local/unravel/install_bin/kafka_clear.sh
```

- Start up Zookeeper ensemble:

```
# host1
sudo /etc/init.d/unravel_all.sh start-zk
# host2
sudo /etc/init.d/unravel_all.sh start-zk
# host3
sudo /etc/init.d/unravel_all.sh start-zk
```

- Wait 15sec for Zookeeper quorum to settle:

```
sleep 15
```

- Start up Kafka ensemble:

```
# host1
sudo /etc/init.d/unravel_all.sh start-k
# host2
sudo /etc/init.d/unravel_all.sh start-k
# host3
sudo /etc/init.d/unravel_all.sh start-k
```

- Wait 10sec for Kafka coordination:

```
sleep 10
```

- Create the Kafka topics (only on one host):

```
# host1
sudo /usr/local/unravel/install_bin/kafka_create_topics.sh
```

5. Start Unravel Server

Finish multi-host installation by starting up Unravel Server:

```
# host1
sudo /etc/init.d/unravel_all.sh start
# host1
echo "http://$(hostname -f):3000/"
# host2
sudo /etc/init.d/unravel_all.sh start
```

6. Edit Hive-site Snippet for Hive-Hook

The port 4043 is on **host2** and that means the `hive-site.xml` file changes needed for Unravel hive-hook are in `/usr/local/unravel/hive-hook/hive-site.xml.snip` on **host2**. If `hive-site.xml` was already configured for **host1**, then modify it for **host2**.

7. Snapshot unravel.properties as new golden file

Attachments:

- worddav95f1e8fd13958fae391afc8e6f9ad03d.png (image/png)
- logo_small.png (image/png)

Creating Multiple Workers for High Volume Data

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics
4. Custom Configurations

These instructions apply to single host Unravel deployments only; for multihost deployments, please contact Unravel Support.

If you have 10000-20000 jobs per day, run these commands on Unravel Server to enable these workers:

```
sudo chkconfig --add unravel_ew_2
sudo chkconfig --add unravel_hhw_2
sudo chkconfig --add unravel_jcw2_2
# If Spark is in use:
sudo chkconfig --add unravel_sw_2
```

If you have 20000-30000 jobs per day, run these commands on Unravel Server to enable these workers:

```
sudo chkconfig --add unravel_ew_3
sudo chkconfig --add unravel_hhw_3
sudo chkconfig --add unravel_jcw2_3
# If Spark is in use:
sudo chkconfig --add unravel_sw_3
```

If you have more than 30000 jobs per day, run these commands on Unravel Server to enable these workers:

```
sudo chkconfig --add unravel_ew_4
sudo chkconfig --add unravel_hhw_4
sudo chkconfig --add unravel_jcw2_4
# If Spark is in use:
sudo chkconfig --add unravel_sw_4
```

Start Unravel Server

Run the following command to start the additional daemons you enabled above:

```
sudo /etc/init.d/unravel_all.sh start
```

Attachments:

- logo_small.png (image/png)

Defining a Custom TC Port

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics
4. Custom Configurations

These instructions apply to any deployment.

Run the following command on Unravel Server **host1**. Replace `18080` with your chosen port number:

```
echo 'export UNRAVEL_LISTEN_PORT=18080' \
>>/usr/local/unravel/etc/unravel.ext.sh
```

Attachments:

- logo_small.png (image/png)

Integrating LDAP Authentication for Unravel Web UI

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics
4. Custom Configurations

To configure lightweight directory access protocol (LDAP) or active directory (AD) authentication for Unravel Web UI, use settings identical to the settings for HiveServer2. In other words, if you have HiveServer2 set up for AD-based LDAP, use the same values for Unravel Web UI. If you do not use HiveServer2 LDAP, then follow the steps below.

1. Modify `unravel.properties`

Add/modify these properties in `/usr/local/unravel/etc/unravel.properties`:

Change `QA` and `ldap://LDAP_HOST` to your specific values:

- For `QA` , use a value in uppercase that matches your setup, like QA.EXAMPLE.COM.
- For `ldap://LDAP_HOST` , use `ldaps://LDAP_HOST` if SSL is in use. If ldaps (SSL), a truststore must be set under `unravel/jre/` if the CA is not well-known or an external JDK can be set in unravel/etc/unravel.ext.sh. You can append a port number, if needed; for example, `ldap://ldap_host:9999` . You can also specify multiple LDAP hosts by using a SPACE to separate them. This is useful for HA purposes.

For Active Directory (AD):

```
com.unraveldata.login.mode=ldap
com.unraveldata.ldap.use_jndi=true
hive.server2.authentication.ldap.url=ldap://LDAP_HOST
hive.server2.authentication.ldap.Domain=QA
```

Change LDAP_HOST and QA to appropriate value for your installation.

For Open LDAP, example 1:

```
com.unraveldata.login.mode=ldap
com.unraveldata.ldap.use_jndi=true
hive.server2.authentication.ldap.url=ldap://LDAP_HOST
hive.server2.authentication.ldap.baseDN=ou=myunit,dc=example,dc=com
```

Change the example ou=myunit,dc=example,dc=com to appropriate value for your installation.

For Open LDAP, example 2:

In this example, we expect a typical DN to be uid=%s,ou=myunit,dc=example,dc=com where %s is the login name as typed in the login form. In some cases 'cn' is used in place of 'uid'.

```
com.unraveldata.login.mode=ldap
com.unraveldata.ldap.use_jndi=true
hive.server2.authentication.ldap.url=ldap://LDAP_HOST
hive.server2.authentication.ldap.guidKey=uid
hive.server2.authentication.ldap.userDNPattern=uid=
%s,ou=myunit,dc=example,dc=com
```

Change the example ou=myunit,dc=example,dc=com to appropriate value for your installation.

2. Restart unravel_tc

Restart `unravel_tc` on Unravel Server (on host1 if multiple host Unravel install):

```
sudo /etc/init.d/unravel_tc restart
```

Advanced Hive Properties

Below is a list of advanced properties that narrow the search for authorized users. For more detail, see the page User and Group Filtering LDAP in HiveServer2

The process of authentication is described next.

Authentication Process for Active Directory (AD)

1. Bind as username + at sign + domain, using the given password, with simple LDAP auth mode
   a. verbose log will show Connecting and then Connected when bind is successful
2. If there are no more qualifications for groups or filtering or custom query, then login to Unravel is deemed successful if bind succeeds
3. If custom query is specified, the query is made, and results are checked for a match against the user name, if there is a match, login to Unravel is successful
   a. verbose log will show results list and the match arguments in effect
   b. user or group filters will be ignored if custom query is used
4. If a group filter is specified, it is checked
5. if a user filter is specified, it is checked

Authentication Process for Open LDAP

1. Bind as cn or uid =username + baseDN using the given password, with simple LDAP auth mode
   a. the guidKey property determines whether cn or uid is used
   b. if userDNPattern is used, it takes precedence over baseDN, and each pattern is tried
2. If there are no more qualifications for groups or filtering or custom query, then login to Unravel is deemed successful if bind succeeds
3. If custom query is specified, the query is made, and results are checked for a match against the user name, if there is a match, login to Unravel is successful
   a. verbose log will show results list and the match arguments in effect
   b. user or group filters will be ignored if custom query is used
4. If a group pattern or filter is specified, it is checked
5. if a user filter is specified, it is checked

`hive.server2.authentication.ldap.baseDN`	LDAP base DN. Use your rootDN value if a custom LDAP query is applied. Needed for Open LDAP. See also userDNPattern as alternative.	`DC=qa,DC=example,DC=com`
`hive.server2.authentication.ldap.customLDAPQuery`	A full LDAP query that LDAP Atn provider uses to execute against LDAP Server. If this query returns a null resultset, the LDAP Provider fails the Authentication request, succeeds if the user is part of the resultset. If this property is set, filtering and group properties are ignored.	`(&(objectClass=group)(objectClass=top)(instanceType=4)(cn=Domain*))` `(&(objectClass=person)(|(sAMAccountName=admin)(|(memberOf=CN=Domain Admins,CN=Users,DC=domain,DC=com)(memberOf=CN=Administrators,CN=Buil`

`hive.server2.authentication.ldap.guidKey`	LDAP attribute name whose values are unique in this LDAP server. REQUIRED for advanced query except when setting custom query or groupDNPattern.	uid or CN
`hive.server2.authentication.ldap.groupDNPattern`	COLON-separated list of patterns to use to find DNs for group entities in this directory. Use %s where the actual group name is to be substituted for. Each pattern should be fully qualified. Do not set ldap.Domain property to use this qualifier.	`CN=%s,CN=Groups,DC=subdomain,DC=domain`
`hive.server2.authentication.ldap.groupMembership.key`	COMMA-separated list of LDAP Group names (short name not full DNs)	`HiveAdmins,HadoopAdmins,Administrat`
`hive.server2.authentication.ldap.userDNPattern`	COLON-separated list of patterns to use to find DNs for users in this directory. Use `%s` where the actual group name is to be substituted for. This is used like a list of baseDNs and baseDN is ignored if this is set.	`CN=%s,CN=Users,DC=subdomain,DC=domain,`
`hive.server2.authentication.ldap.userMembership.key`	COMMA-separated list of LDAP usernames (just short names, not full DNs).	`hiveuser,impalauser,hiveadmin,hadoo`
`hive.server2.authentication.ldap.groupMembership.key`	LDAP attribute name on the user entry that references a group that the user belongs to. Default is 'member'.	`member, uniqueMember or memberUid`
`hive.server2.authentication.ldap.groupClassKey`	LDAP attribute name on the group entry that is to be used in LDAP group searches.	`group, groupOfNames or groupOfUniqueNames`
`com.unraveldata.ldap.verbose`	enables verbose logging. Grep for "Ldap" entries in the unravel_tc_webapp.log file under /usr/local/unravel/logs/ ; when enabled, user names and group names can appear in this log, but raw passwords are not logged.	Can be true or false or not set; default is false

Attachments:

- logo_small.png (image/png)

Setting Retention Time in Unravel Server

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics
4. Custom Configurations

To adjust the retention time (*time horizon*), you need to change three settings. These settings can be made in /usr/local/unravel/etc/unravel.properties or via the **Manage** page, **Configuration** tab, **Core** section (tab on left) and

scroll down to the **Retention** heading. If you make the setting in the **Manage** page **Configuration** tab, then the corresponding property in unravel.properties *will be ignored* because settings made in the web UI take precedence.

When changing these settings, be aware that long retention requires significant disk space. As a rule of thumb, each map-reduce or Spark job requires about 500KB of disk space. That means about 2000 jobs per 1GB of disk.

1. In Unravel Web UI, select the **Manage** page, **Configuration** tab, and **Core** section (tab on left). The settings below are show the label in the web UI and the boxes show the corresponding property you can optionally set in /usr/local/unravel/etc/unravel.properties if you prefer.
2. The **TIME SERIES RETENTION DAYS** field is number of days to keep the heaviest data (such as error logs and drill-down details). This corresponds to the property in `unravel.properties`:

   ```
   com.unraveldata.retention.max.days=90
   ```

3. The **WEEKS TO SHOW FOR SEARCH RESULTS** field is number of weeks to show in search results. This corresponds to the property in `unravel.properties`:

   ```
   com.unraveldata.history.maxSize.weeks=7
   ```

   This value should be no larger than the next setting minus 1.
4. The **WEEKS TO SHOW FOR DEEP SEARCH RESULTS** field is number of weeks to retain for search results. This corresponds to the property in `unravel.properties`:

   ```
   com.unraveldata.recent.maxSize.weeks=14
   ```

   This value should be at least 1 week more than the setting immediately above.
5. After changing any of the settings above, restart unravel_td service:

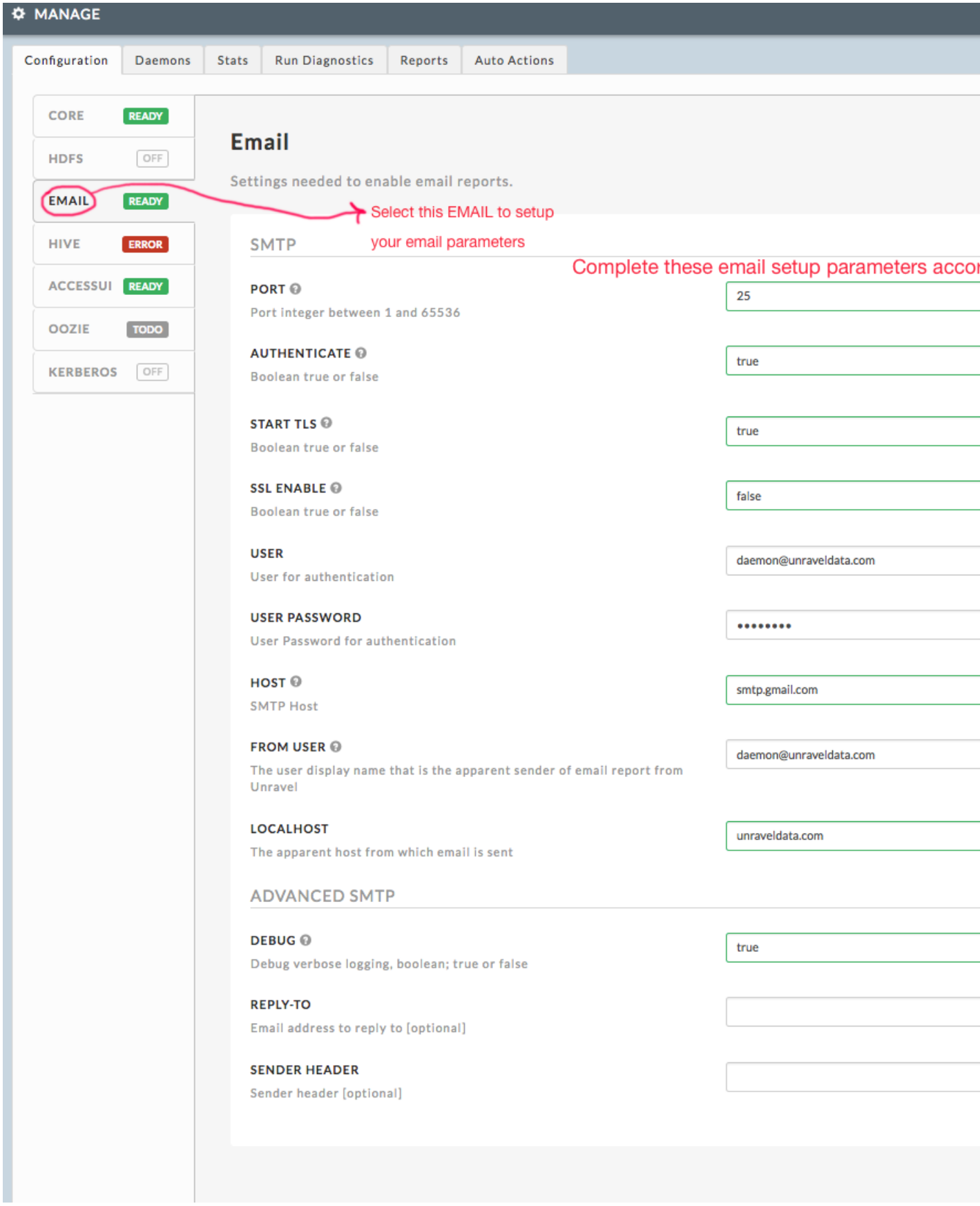
   ```
   sudo /etc/init.d/unravel_td restart
   ```

Setting Up Email for Auto Actions and Collaboration

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics
4. Custom Configurations

You can specify an SMTP server for Unravel Server so that it can send reports, alerts, and collaboration emails. Several examples are shown below. Adapt the one that is most similar to your environment.

You can set these values through Unravel Web UI's **Manage** page | **Configuration** | **Email** side tab as shown in diagram. The **Web UI** column in the table below corresponds to the values in the Unravel Web UI dialog box.


MANAGE
Configuration
Daemons
Stats
Run Diagnostics
Reports
Auto Actions
CORE READY
HDFS OFF
EMAIL READY
HIVE ERROR
ACCESSUI READY
OOZIE TODO
KERBEROS OFF
Email
Settings needed to enable email reports.
Select this EMAIL to setup
your email parameters
SMTP
Complete these email setup parameters accor
PORT
Port integer between 1 and 65536
25
AUTHENTICATE
Boolean true or false
true
START TLS
Boolean true or false
true
SSL ENABLE
Boolean true or false
false
USER
User for authentication
daemon@unraveldata.com
USER PASSWORD
User Password for authentication
••••••••
HOST
SMTP Host
smtp.gmail.com
FROM USER
The user display name that is the apparent sender of email report from Unravel
daemon@unraveldata.com
LOCALHOST
The apparent host from which email is sent
unraveldata.com
ADVANCED SMTP
DEBUG
Debug verbose logging, boolean; true or false
true
REPLY-TO
Email address to reply to [optional]
SENDER HEADER
Sender header [optional]

An alternative to using Unravel Web UI's **Manage** page is to enter the settings into `/usr/local/unravel/etc/unravel.properties` using the property name shown in the **Property** column.

If you specify a saved email setting in Unravel Web UI, that setting overrides the corresponding setting in the `unravel.properties` file.

Defaults

When you do not specify properties or configuration settings, Unravel Server tries to use the default 'classic' SMTP setting at localhost:25 ; this sometimes works for customers that set up SMTP spooling with sendmail or postfix, but it might block emails to external domains (for anti-spam reasons). On EC2, this sometimes works for small emails, but significant use is blocked for anti-spam reasons.

PORT	mail.smtp.port	25	Port
AUTHENTICATE	mail.smtp.auth	false	Enable SMTP authentication? If true, then USER (mail.smtp.user) and USER PASSWORD (mail.smtp.pw) are used when connecting
START TLS	mail.smtp.starttls.enable	false	Use start-TLS?
SSL ENABLE	mail.smtp.ssl.enable	false	Use SSL right from the start?
USER	mail.smtp.user	null	Username for SMTP authentication
USER PASSWORD	mail.smtp.pw	null	Password for SMTP authentication
HOST	mail.smtp.host	localhost	Host for SMTP server
FROM USER	mail.smtp.from	someone@example.com	Use a `From:` name that is appropriate for your environment
LOCALHOST	mail.smtp.localhost	localhost.local	A domain name for apparent sender; must have at least one dot (e.g. organization.com)
DEBUG	mail.smtp.debug	false	Enable debug mode? Set to true (temporarily) to see more details in logs.

GMail SMTP Example

These settings are for our internal use. Do **not** compile this into the product or otherwise use as a default in our source code. For security reasons, we don't want to mix our internal testing SMTP (or POP) with the external one.

PORT	mail.smtp.port	587	Port
AUTHENTICATE	mail.smtp.auth	true	Enable SMTP authentication?
START TLS	mail.smtp.starttls.enable	true	Use start-TLS?
SSL ENABLE	mail.smtp.ssl.enable	false	Use SSL right from the start?
USER	mail.smtp.user	someone@organization.com	Username for SMTP authentication

USER PASSWORD	mail.smtp.pw	********	Password for SMTP authentication
HOST	mail.smtp.host	smtp.gmail.com	Host for SMTP server
FROM USER	mail.smtp.from	someone@example.com	This sets the From header
LOCALHOST	mail.smtp.localhost	example.com	A domain name for apparent sender; must have at least one dot
DEBUG	mail.smtp.debug	false	Enable debug mode? Set to true (temporarily) to see more details in logs. Debug mode under "Advance SMTP" section

Unravel daemons to restart after email setup

Run the following commands on Unravel Server:

```
sudo /etc/init.d/unravel_tc restart
sudo /etc/init.d/unravel_all.sh stop-etl
sudo /etc/init.d/unravel_all.sh start
```

Verify email setup works

Run the following commands on Unravel Server:

```
sudo -u unravel /usr/local/unravel/install_bin/diag_email.sh
 someone@example.com
--> enter password from dist.unraveldata.com
--> should see following output in terminal mode and if you see "result is =
 null", then, setup is correct.
:
:
result is = null
At least one smtp pathway worked
for log output see /usr/local/unravel/logs/test_email.log
```

See the stdout. It will test smtp settings (either from unravel.properties or defaults or in settings table in db or command line overrides). It will also test "smtp2" email which is compiled-in as a backup for alerts to Unravel Support. Customer reports are **not** send via smtp2, so if only that one works, customer report email is not going to work.

Email setup for Auto-Actions

After above email setup has been completed in Unravel UI under Email Config Wizard, next, please do below steps to configure Auto-Actions.

1. Add following properties to /usr/local/unravel/unravel.properties on Unravel Server:

   ```
   mail.smtp.from=someone@example.com
   com.unraveldata.report.user.email.domain=example.com
   ```

2. Disable unneeded daemons:

   ```
   sudo service unravel_os3 stop
   sudo chkconfig unravel_os3 off
   ```

3. Restart daemons:

```
sudo /etc/init.d/unravel_all.sh restart
```

Attachments:

- logo_small.png (image/png)
- Unravel_email_setup.png (image/png)

Connecting to a Hive Metastore

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics

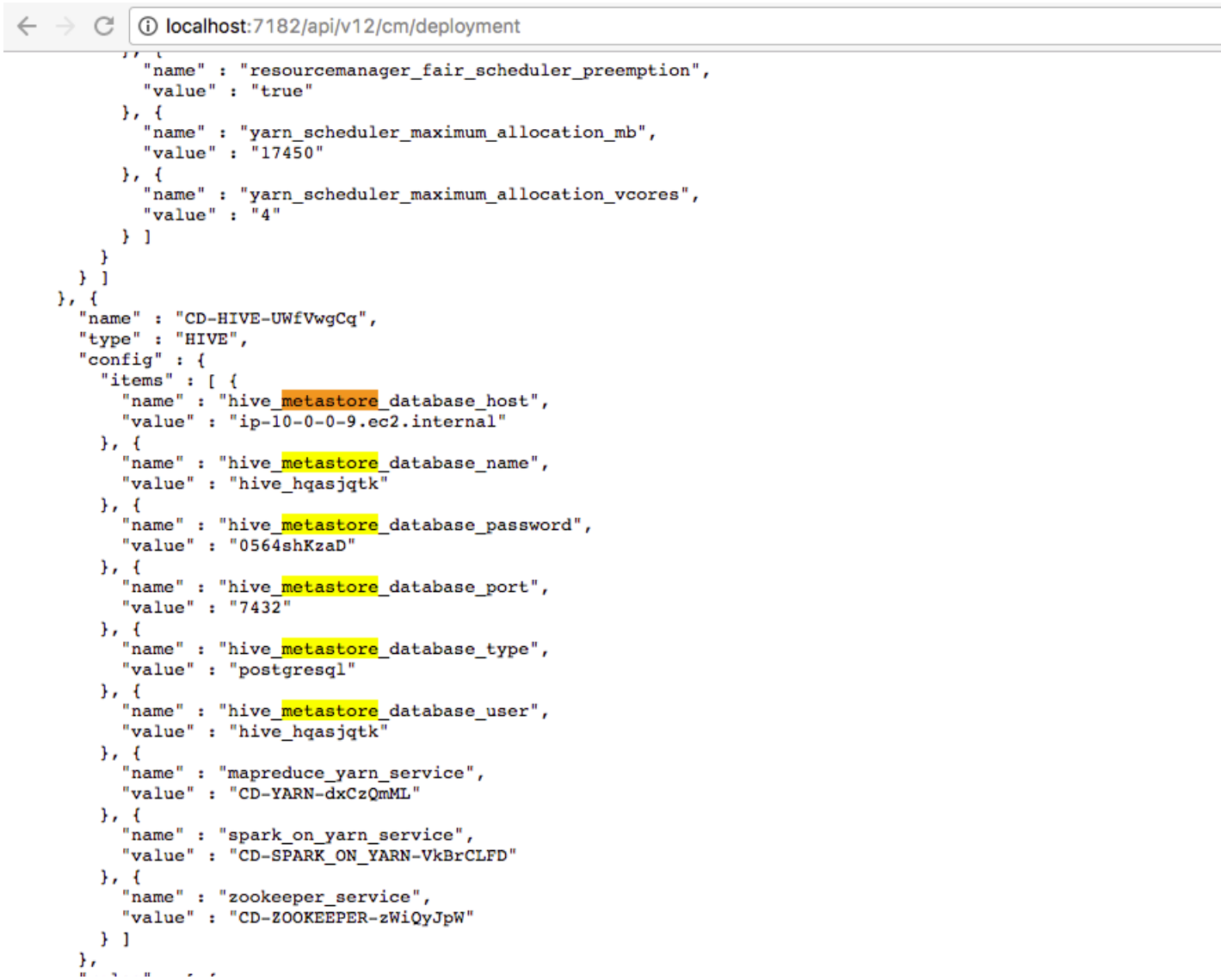
Connecting to a Hive metastore enables Unravel Server to collect instrumentation from it, which results in the population of the Data Page in Unravel Web UI.

For CDH+CM

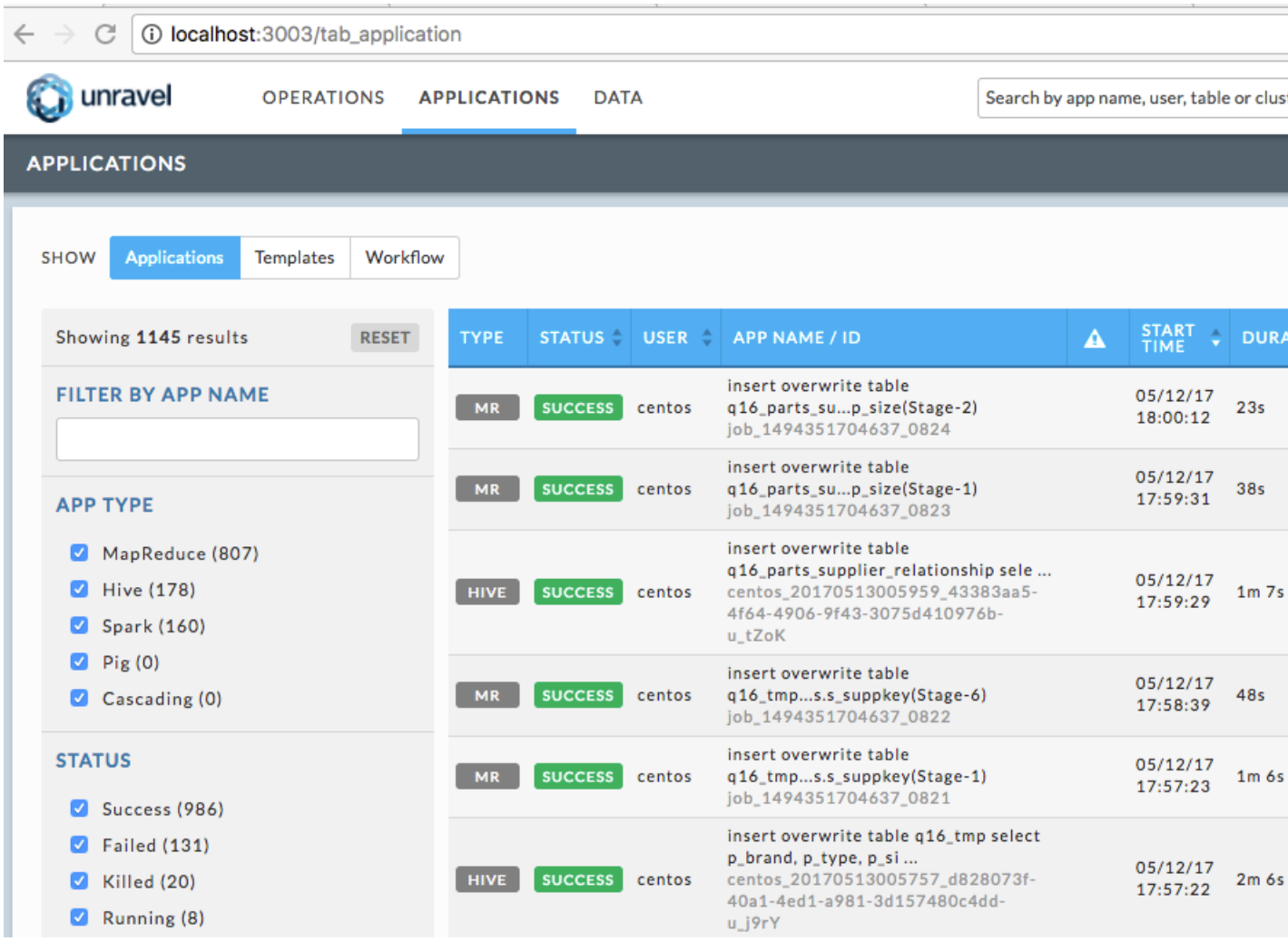
You can enable instrumentation for the Hive metastore through Unravel Web UI's configuration wizard, but first you need to obtain Hive metastore details from Cloudera Manager.

1. Obtain the Hive metastore details from the Cloudera Manager by using a CDH REST API

- From CDH version 5.5 onward, use the REST API "`http://CMGR_HOSTNAME_IP:7182/api/v12/cm/deployment`"
- Look at the response body, a JSON-like text format as in the image below.
- Search the response body for "`metastore`".

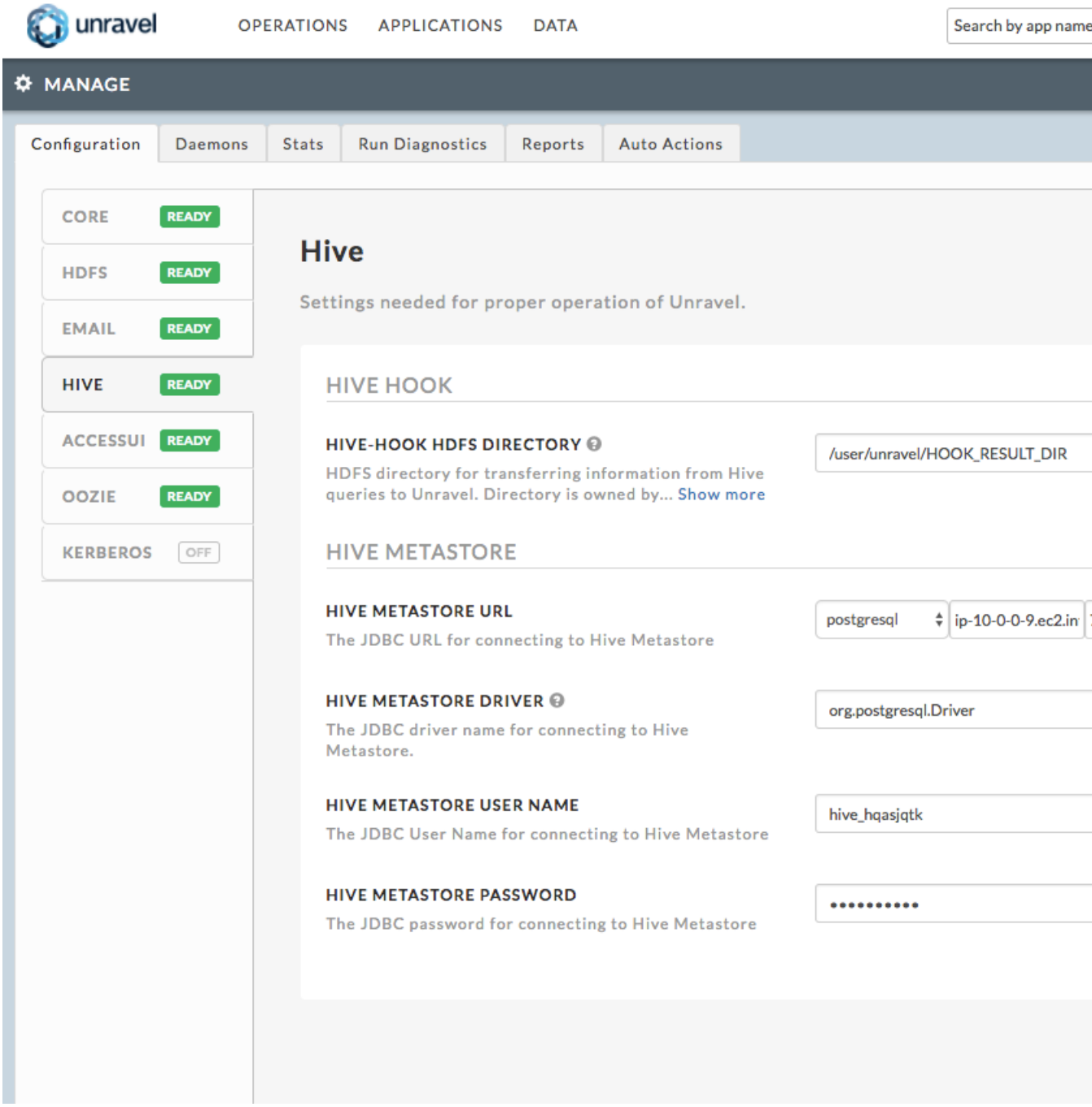


2. In Unravel Web UI, on the top right-hand corner, click **Admin** and, in the pull-down menu, select **Manage**.



3. On the left tab, click **Hive**, and fill in the values you obtained from Cloudera Manager:

- • HIVE METASTORE URL
  - HIVE METASTORE DRIVER
  - HIVE METASTORE USER NAME
  - HIVE METASTORE PASSWORD



4. Save the information when done: click **Save Changes**.

5. Restart Unravel Server:

```
sudo /etc/init.d/unravel_all.sh restart
```

6. After restart, confirm that Hive queries appear in Unravel UI in the **Application** tab.

For HDP

See Part 2: Enable Additional Data Collection / Instrumentation for HDP

For MapR

See Part 2: Enable Additional Data Collection / Instrumentation for MapR

Attachments:

- logo_small.png (image/png)
- UI_Admin_Manage_page.png (image/png)
- UI_Hive_Metastore_page.png (image/png)
- CMGR_rest_api_extracting_HM_pwd.png (image/png)

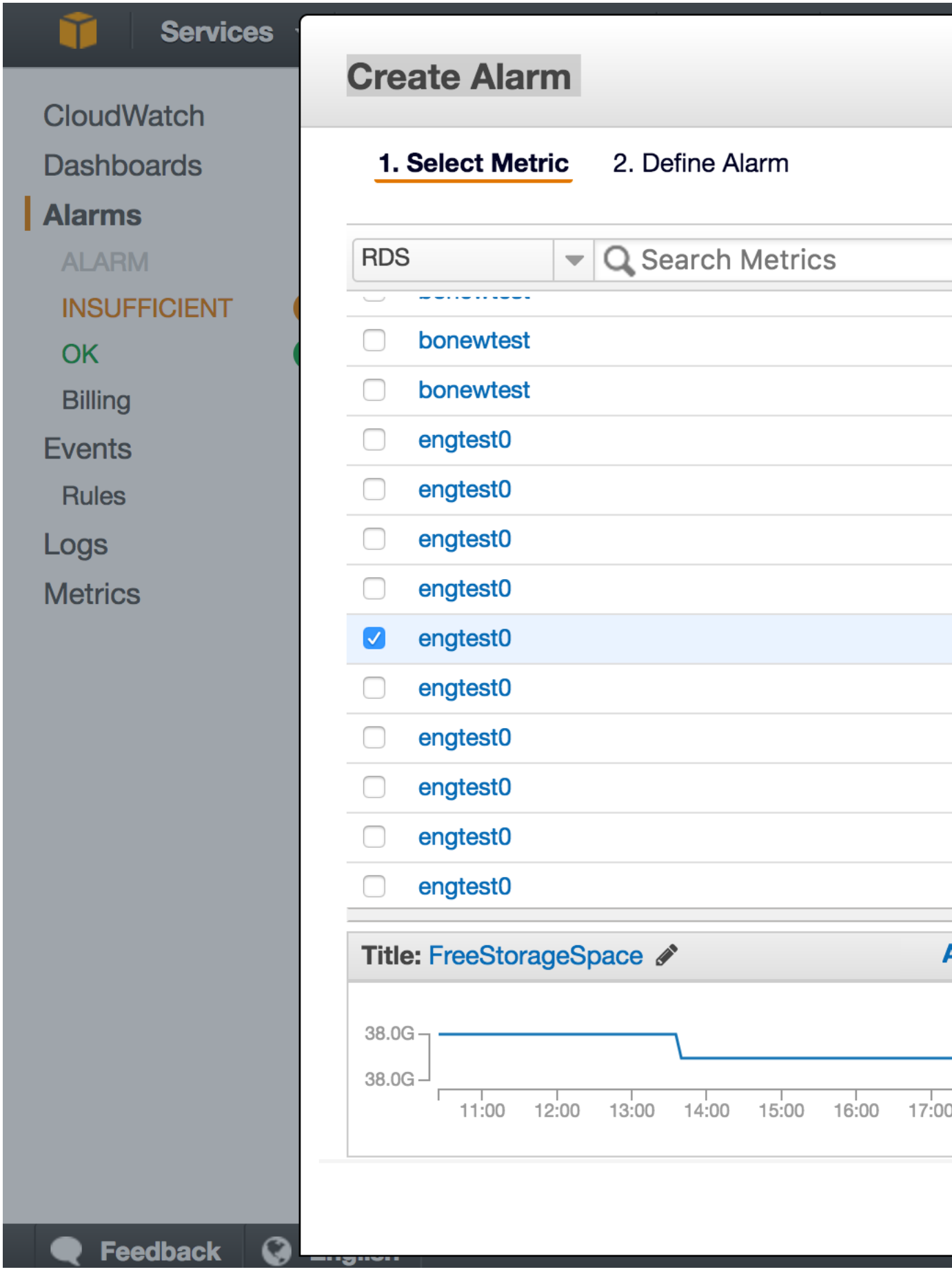
Creating an AWS RDS CloudWatch Alarm for FreeStorageSpace

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics

This guide is to configure an AWS RDS CloudWatch Alarm for Disk *FreeStorageSpace* Metrics as part of RDS monitoring:

1. Go to **AWS Cloud Watch**
2. Select on the left-hand corner tab for " **Alarms"**
3. Click on " **Create Alarm** "
4. On the right-hand section under " **RDS Metrics** " and click on " **Per-Database Metrics** "

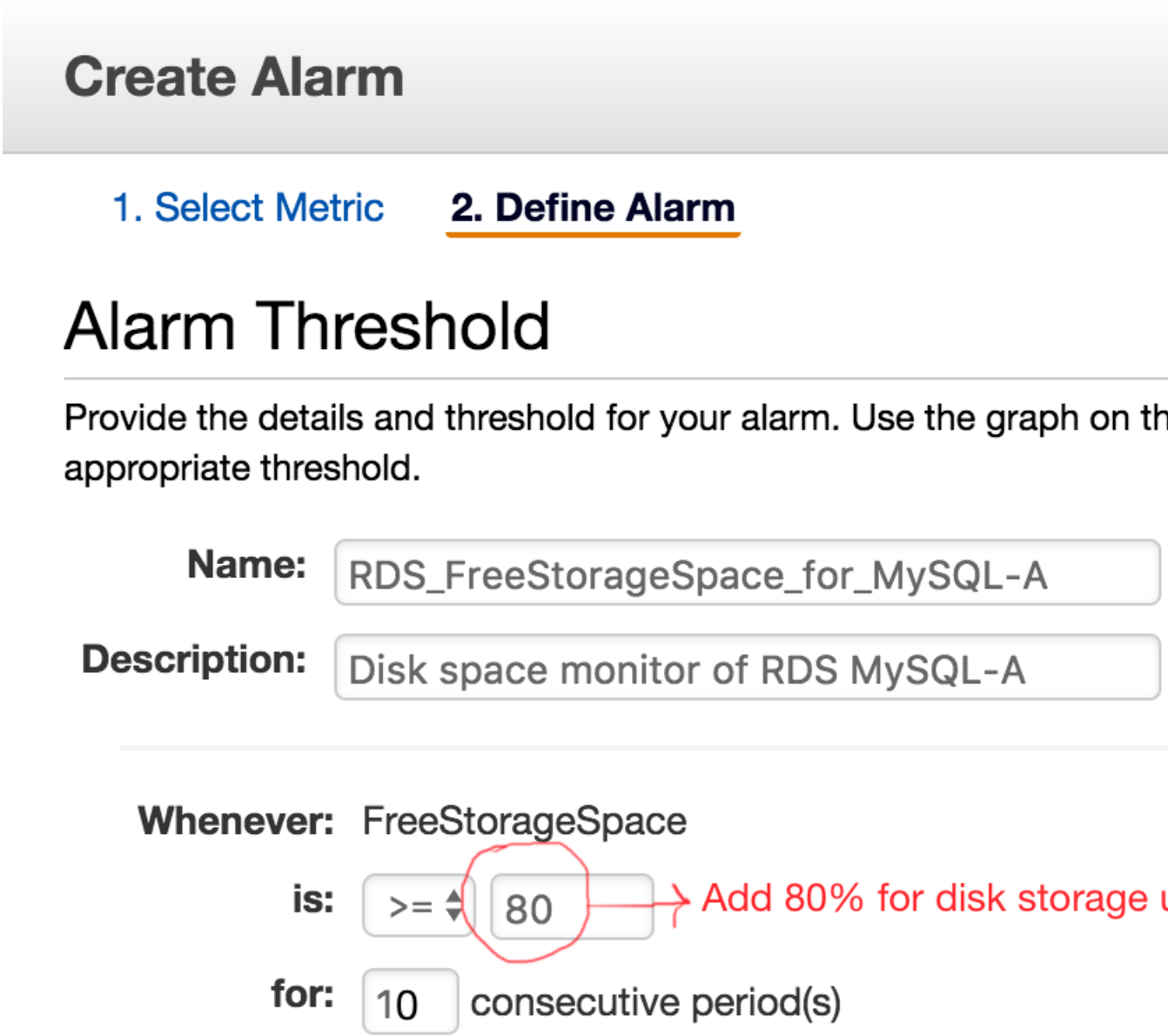
5. Under the column DBInstanceIdentifier, *select the database* you wish to monitor for " **FreeStorageSpace** " and click " **Next**" when you are done



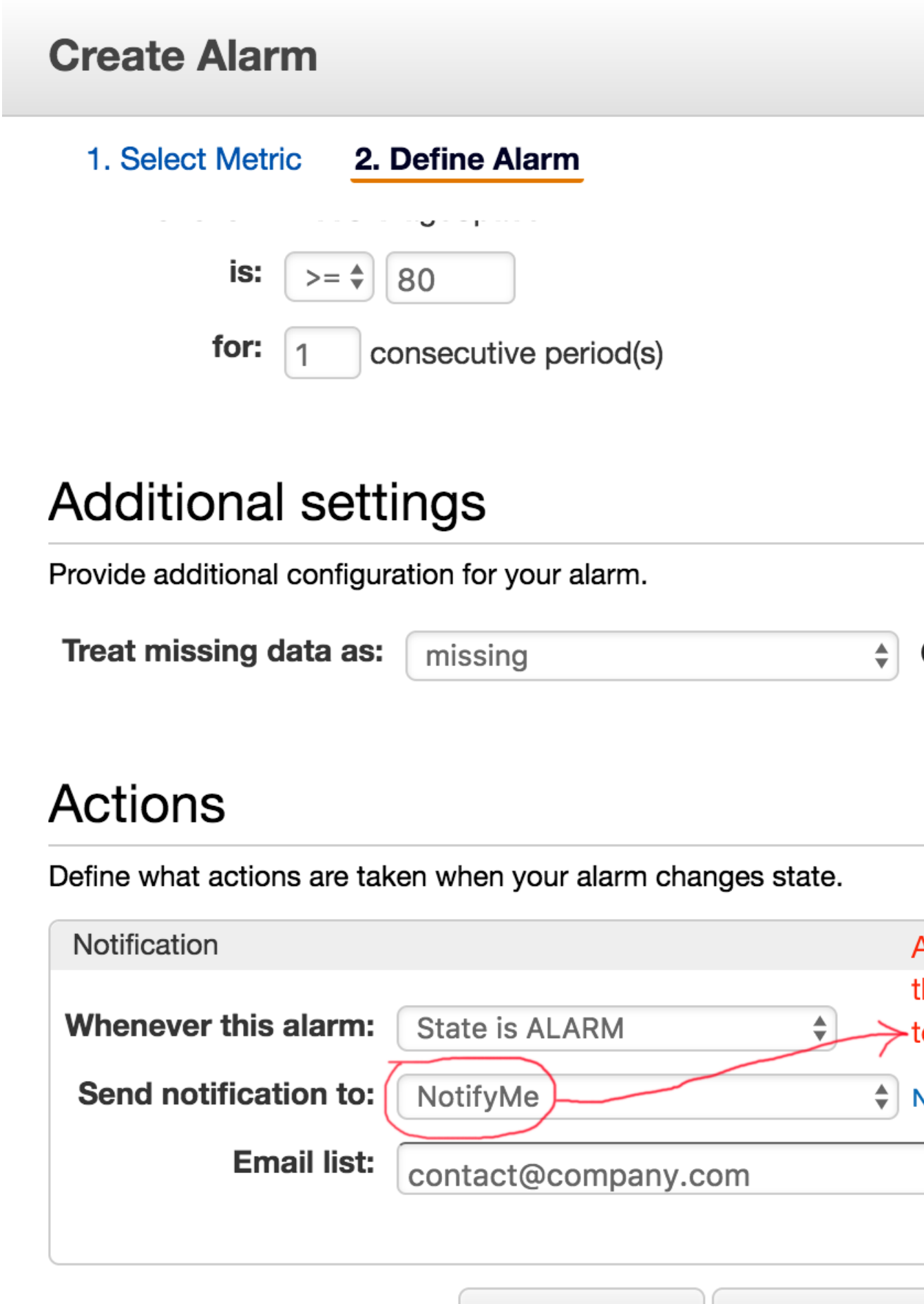
6. In " **Alarm Threshold** " panel, please add and complete following:
   a. " **Name** " - for this Database Metrics (e.g. RDS_FreeStorageSpace_for_MySQL-A)
   b. " **Description** " - describe what the above database metrics name you entered (e.g. Disk space monitor of RDS MySQL-A)
   c. Add free storage of 20% left to alert contact under " **Whenever FreeStorageSpace** " is <= **2 0**

      I have suggested adding 20% of free storage space left, however, you can tune this to be lower.

d. Add 10 for " **10** " **consecutive period(s)**



e. Under **Actions** , add " **Send notifications to** ", which is your SNS topic: *Note: this sns topic should already be setup before you add it.*



f. Click " **Create Alarm** " to create the Alarm metrics for RDS monitoring on Storage Space
g. Now, you will see in " **Alarms** " the alarm you just created, but do not be surprise when you see "INSUFFICIENT DATA" and it is ok since the alarm is not triggered, yet. Once, it is triggered, an ALARM appears under " **Alarms** " tab. In addition, email alerts will be distributed as defined in the SNS topic.
h. Click " **Create Alarm** " to create the Alarm metrics for RDS monitoring on Storage Space

Attachments:

AWS_CW_RDS_FreeStorageSpace.png (image/png)

AWS_CW_RDS_Disk_Used_80.png (image/png)

AWS_CW_RDS_Notify.png (image/png)

AWS_CW_RDS_Disk_Used_80.png (image/png)

Creating Application Tags

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics

Annotating applications with tags (key-value pairs) allows you to search, group, and charge back based on tags. It also allows you to track Unravel's insights based on tags. Thus, understanding how tags work in Unravel is crucial.

Application tags are immutable: once created they cannot be changed.

What is a Tag in Unravel?

A tag is key-value pair <*k,v*> that has been associated with an application *A*.

Thus, application *A* in Unravel can have zero or more tags associated with it: <*k1,v1*>, <*k2,v2*>, ...

How Does Unravel Use Tags?

Unravel Server and Web UI use tags to:

- Group applications for chargeback reports
- Provide access control for different users
- Search/group applications using tags
- Group applications for insights

What Types of Tags Are There?

There are two types of tags: Unravel tags and user-created tags.

Unravel Tags

All the tag names starting with `unravel` are internal to Unravel, and have a specific meaning. For example, `unravel.workflow.`*`name`* and `unravel.workflow.`*`utctimestamp`* are used to create workflows. Best practice is to use these tags only if you want to get the behavior associated with that tag.

User-Created Tags

You can create tags based on your use cases.

How Do I Create Tags?

There are two ways to create tags: by adding them to the configuration of your application, or by writing a Python script to inject them.

Adding Tags to your Application's Configuration

Add tags to your application's configuration (configuration file or `setConf`). Tags must be in key-value format (*`key1,value1,key2,value2,key3,value3,...`*). Keys and values can only contain alphanumeric characters. Unravel Server extracts these key-value pairs from the application's configuration file.

Injecting Tags Through a Python Script

You can write Python script which is invoked in the ingestion pipeline and is set up to access application metadata to create tags on the fly.

Unravel receives metadata about applications from different sources, and that metadata can be received out of order, but it is merged and eventually reaches a consistent state. For example, Spark receives data from Resource Manager, event log file, YARN aggregated logs, and sensors.

Your Python script must be idempotent, in other words, it must produce the same result over multiple invocations with different input (metadata) for the same application.

Precedence of Tags

Unravel gives precedence to tags in this order (low to high):

- Unravel tags defined in application configuration
- Tags extracted by Python script

Sample Use Cases

- Differentiate between job types such as pipeline and workflow
- Differentiate between projects, queues, departments, groups, users, and tenants

Troubleshooting

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics

Troubleshooting article

Provide solutions for commonly encountered problems.

<In progress>

If you can't reach Unravel Server, (i) ping LANS_DNS, (ii) try this workaround:

Attachments:

logo_small.png (image/png)

Running Verification Scripts and Benchmarks

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1

3. Advanced Topics
4. Troubleshooting

This topic explains how to run verification tests and benchmarks after you install or upgrade Unravel Server.

Why Run Verification Tests or Benchmarks?

- Verification tests highlight the value of Unravel's application performance management/analysis.
- Benchmarks verify that Unravel features are working correctly.

Running Verification Tests ("Smoke Tests")

Currently, we provide smoke tests for Spark jobs only. Please follow the instructions in the section that matches your deployment.

CDH

- On your Unravel Server host, run the spark_test_via_parcel.sh script. This script runs a Spark app. It's a good way to verify that Unravel Server captures the data (events) generated by the Spark app, even before you install and configure Unravel Sensor. You should be able to see the data generated by this Spark app on Unravel Web UI.

```
/usr/local/unravel/install_bin/spark_test_via_parcel.sh --unravel-server
 <unravel_host_IP_address>
```

Note: You can run this script without installing and configuring Unravel Sensor.

- After you install Unravel Sensor for Spark, run a SparkPI job on your Unravel Server host to verify that the sensor is installed and configured correctly:

```
/opt/cloudera/parcels/CDH/lib/spark/bin/spark-submit \
--class org.apache.spark.examples.SparkPi \
--deploy-mode client \
--master yarn \
/opt/cloudera/parcels/CDH/lib/spark/examples/lib/spark-examples-*.jar 1000
```

HDP

- After you install Unravel Sensor for Spark, run a SparkPI job on your Unravel Server host to verify that the sensor is installed and configured correctly:

```
/usr/bin/spark-submit \
--class org.apache.spark.examples.SparkPi --master yarn-client \
--num-executors 1 --driver-memory 512m --executor-memory 512m \
--executor-cores 1 /usr/hdp/current/spark-client/lib/spark-examples*.jar
 10
```

MapR

- After you install Unravel Sensor for Spark, run a SparkPI job on your Unravel Server host to verify that the sensor is installed and configured correctly:

```
/opt/mapr/spark/spark-1.6.1/bin/spark-submit \
--class org.apache.spark.examples.SparkPi --master yarn-client \
--num-executors 1 --driver-memory 512m --executor-memory 512m \
--executor-cores 1 /opt/mapr/spark/spark-1.6.1/lib/spark-examples*.jar 10
```

Running Benchmarks

We provide sample Spark, MapReduce, Hive, and WF apps that you can download from preview.unraveldata.com. These apps are useful for verifying that an upgrade is successful. Please follow the instructions below for the app you want.

Spark

1. Download our sample Spark app:

```
curl https://preview.unraveldata.com/img/spark-benchmarks1.tgz -o spark-benchmarks1.tgz
```

   This `.tgz` file includes both datasets and scripts.
2. Run `md5sum` on spark-benchmarks1.tgz to ensure it is intact:

```
md5sum spark-benchmarks1.tgz
```

3. Confirm that the output of `md5sum` is exactly as shown on the line below:

```
ff8e56b4d5abfb0fb9f9e4a624eeb771  spark-benchmarks1.tgz
```

4. Uncompress the `.tgz` file:

```
tar -zxvf spark-benchmarks1.tgz
```

5. Run the samples.

   Tip: Instructions on how to run the samples are included in the package itself, inside demo-benchmarks-for-spark/benchmarks/README.
6. After running the samples, check the Program and the Execution Graph tabs in Unravel Web UI. Click an RDD in the Execution Graph to see the corresponding line of code in the app.

Attachments:

- logo_small.png (image/png)

Sending Diagnostics to Unravel Support

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics
4. Troubleshooting

1. In the upper right corner of Unravel Web UI, click the pull-down menu, and select **Manage**.
2. Wait for the page to fully load.
3. Select the **Diagnostics** tab.
4. Click **Send Diagnostics to Unravel Support**.

This sends an email message with a diagnostics report to Unravel Support and also to the users listed in the `com.unraveldata.login.admins` property.

Attachments:

- logo_small.png (image/png)

Uninstalling Unravel Server

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics

Disable and uninstall cluster instrumentation before uninstalling Unravel Server.

The Unravel Server RPM has the service name `unravel`. That means you can uninstall it by running the following commands:

```
sudo rpm -e unravel
sudo /bin/rm -rf /usr/local/unravel /srv/unravel/* /etc/unravel_ctl
```

All data in the bundled database is deleted by the `rm` command.

Attachments:

- logo_small.png

Upgrading Unravel Server

1. Unravel 4.0-4.1
2. Unravel 4.0-4.1
3. Advanced Topics

This topic explains how to upgrade the Unravel Server RPM in a single or multi-host environment.

For single Unravel gateway or client host, run only the commands that are marked as `host1` in each step below.

1. Copy the new RPM to each Unravel host.
2. Stop each host **simultaneously**:

```
# host1
sudo /etc/init.d/unravel_all.sh stop
# host2
sudo /etc/init.d/unravel_all.sh stop
# host3
sudo /etc/init.d/unravel_all.sh stop
```

3. Upgrade the RPM on each host **simultaneously**:

```
# host1
sudo rpm -U unravel-4.*.x86_64.rpm*
# host2
sudo rpm -U unravel-4.*.x86_64.rpm*
# host3
sudo rpm -U unravel-4.*.x86_64.rpm*
```

4. Add your license key to `unravel.properties`.

   You must enter add license key to `unravel.properties` before starting/restarting Unravel Server.
5. Run **/usr/local/unravel/install_bin/await_fixups.sh** script after upgrade

```
sudo /usr/local/unravel/install_bin/await_fixups.sh
```

1. After all the RPM upgrades finish, restart Unravel Server on each host **simultaneously**:

```
# host1
sudo /etc/init.d/unravel_all.sh start
# host2
sudo /etc/init.d/unravel_all.sh start
# host3
sudo /etc/init.d/unravel_all.sh start
```

Attachments:

- logo_small.png (image/png)
- worddav95f1e8fd13958fae391afc8e6f9ad03d.png (image/png)