

Hi Folks,

As discussed in our latest call, I am enclosing information regarding the 'hdiapps' framework that aims to facilitate integration of OSS applications into Ambari running on HDInsight.

The current version of the codebase is forked from the official repo. I will attempt to push through a PR in the next week or so. The current location: <https://github.com/jamesbak/laas-Applications/tree/titandb> contains all of the scripts + 2 full implementations (OpenTSDB & TitanDB). The OpenTSDB implementation is the most informative and complete (it includes metrics, Ambari UX config template, alerts & Web UX integrated as an Ambari View).

The requirements for integration are in 2 phases; 1) Ambari service development, and 2) HDInsight deployment integration. The steps are as follows:

1. Wrap the YARN service as an Ambari service as described here: <https://cwiki.apache.org/confluence/display/AMBARI/Custom+Services>. This can be a fairly rigorous implementation as illustrated by OpenTSDB (<https://github.com/jamesbak/laas-Applications/tree/titandb/OpenTSDB/services/OPENTSDB>), or more minimal such as TitanDB (<https://github.com/jamesbak/laas-Applications/tree/titandb/TitanDB/services/TITANDB>). Once the scripts for defining & controlling the Ambari service are complete, the entire package must be tar.gz (with the service name – in capitals – at the root of the archive) and the file uploaded to a publicly readable location (Azure blob storage works well for this). The structure of the archive follows this pattern:

```
OPENTSDB/  
OPENTSDB/alerts.json  
OPENTSDB/configuration/  
OPENTSDB/configuration/opentsdb-config.xml  
OPENTSDB/configuration/opentsdb-site.xml  
OPENTSDB/metainfo.xml  
OPENTSDB/metrics.json  
OPENTSDB/package/  
OPENTSDB/package/files/  
OPENTSDB/package/files/metrics_sink.sh  
OPENTSDB/package/scripts/  
OPENTSDB/package/scripts/opentsdb.py  
OPENTSDB/package/scripts/params.py  
OPENTSDB/package/scripts/tsd_proxy.py  
OPENTSDB/package/templates/  
OPENTSDB/package/templates/nginx-sites-available-default.j2  
OPENTSDB/themes/  
OPENTSDB/themes/theme.json  
OPENTSDB/widgets.json
```

Note that the scripts can use functionality in the **ambari_commons** module (specifically, the **resource_management** module is very useful), which is already in the module path. Any templates or static files referenced in your scripts are included in your archive in the appropriate location.

2. Verify that the Ambari service is working by manually installing onto a running cluster on **headnodehost**. New services are installed at; `/var/lib/ambari-server/resources/stacks/HDP/{stack-version}/services`. Note that once you have unpacked the service archive you will need to make a configuration change to `/etc/ambari-server/conf/ambari.properties` thus; `agent.auto.cache.update=true`. You will then need to make the service effective with the following 2 commands:

```
ambari-server refresh-stack-hash
systemctl restart ambari-server
```

You can then use the Ambari web UX to add the new service & verify that it deploys to the cluster correctly.

3. HDI deploy requires 2 additional parts; a service configuration file & an ARM deployment file. The configuration file (OpenTSDB example is here: <https://github.com/jamesbak/laas-Applications/blob/titandb/OpenTSDB/config/opentsdb-service-config.json>) and includes the location of the Ambari service archive + additional attributes + component topology definitions + initial configuration.

- a. The **configurations** attribute must have members that align with configuration files defined in the Ambari service package. The configuration values must also match the values in these files.
 - b. The **components** attribute defines the topology options for components previously defined in **metainfo.xml** in the Ambari service. Multiple topologies may be specified for each component. The valid options are; 'head', 'worker', 'region', 'edge'. These values may be overridden at runtime by the `--component-topologies` command line argument. The **startPrimary** attribute causes this component on only 1 host (selected at random) to be started before all of the other instances to prevent a race condition with other nodes for activities such as schema creation.
 - c. The **iframeViewer** attribute tells the script to generate a simple Ambari View jar file which simply embeds an IFrame element that points to the Web UX running on the edge node. The edge node Web UX will still be available directly, but integration into Ambari Views is a nice convenience feature for users.
4. The ARM deployment file should be modeled on the OpenTSDB example (<https://github.com/jamesbak/laas-Applications/blob/titandb/OpenTSDB/azuredeploy.json>). It does a very basic HDI cluster deployment with 1 edge node. The script run on the edge node is a common script that only writes an identifying value to the log (ie. It does NOT install anything). The main script run on the head nodes performs all of the installation & service configuration work. On the active headnode (headnodehost), the script detaches from itself and then polls Ambari to determine when the edge node is installed. Only

after the edge node has been completely installed, the script restarts Ambari (restarting Ambari during a HDI deployment causes HDI to fail) and then configures Ambari to install service components according to the specified topology.

Once the deployment is complete, users will be able to monitor and manage the service directly from the Ambari Web UX. The service will be listed as a peer of standard services (HDFS, Hive, HBase, etc.) and can be configured in the standard means. Service instances can also be controlled (started, stopped, restarted, etc.) and instances can be added to other nodes. Depending on the completeness of your Ambari service implementation, you will also see metrics monitoring and/or alerts.

Things to do:

1. Get PR merged with official HDInsight repository
2. Add support for service advisors (<https://cwiki.apache.org/confluence/display/AMBARI/Service+Advisor>) so that as the HDI cluster is scaled up or down, service instances are automatically added appropriately
3. Add tests

It would be great to get your feedback on this framework. The intent is make the integration process as easy as possible. If the friction is still too high, then please let me know & we can work on improving the framework.

Cheers,
James