Building, Configuring & Installing Apache Atlas

Building Atlas

```
git clone https://git-wip-us.apache.org/repos/asf/atlas.git atlas
cd atlas
```

If you're using Atlas at **version 0.8.x or below**

```
export MAVEN_OPTS="-Xmx1024m -XX:MaxPermSize=256m" && mvn clean install
```

If you're on levels above this, including master as of 27 Oct 2017, use:

```
export MAVEN_OPTS="-Xmx1024m" && mvn clean install
```

as Atlas has now moved to Java8, and the -XX:MaxPermSize is no longer valid with this level

Once the build successfully completes, artifacts can be packaged for deployment.

```
mvn clean package -DskipTests -DskipCheck=true
```

Tar can be found in atlas/distro/target/apache-atlas-${project.version}-bin.tar.gz

Tar is structured as follows

```
|- bin
   |- atlas_start.py
   |- atlas_stop.py
   |- atlas_config.py
   |- quick_start.py
   |- cputil.py
|- conf
   |- application.properties
   |- client.properties
   |- atlas-env.sh
   |- log4j.xml
   |- solr
      |- currency.xml
      |- lang
         |- stopwords_en.txt
      |- protowords.txt
      |- schema.xml
      |- solrconfig.xml
      |- stopwords.txt
      |- synonyms.txt
|- docs
|- server
   |- webapp
      |- atlas.war
|- README
|- NOTICE.txt
|- LICENSE.txt
|- DISCLAIMER.txt
|- CHANGES.txt
```

Installing & Running Atlas

Installing Atlas

```
tar -xzvf apache-atlas-${project.version}-bin.tar.gz
* cd atlas-${project.version}
```

Configuring Atlas

By default config directory used by Atlas is {package dir}/conf. To override this set environment variable METADATA_CONF to the path of the conf dir.

atlas-env.sh has been added to the Atlas conf. This file can be used to set various environment variables that you need for you services. In addition you can set any other environment variables you might need. This file will be sourced by atlas scripts before any commands are executed. The following environment variables are available to set.

```
# The java implementation to use. If JAVA_HOME is not found we expect java and jar to be in path
#export JAVA_HOME=

# any additional java opts you want to set. This will apply to both client and server operations
#export METADATA_OPTS=

# any additional java opts that you want to set for client only
#export METADATA_CLIENT_OPTS=

# java heap size we want to set for the client. Default is 1024MB
#export METADATA_CLIENT_HEAP=

# any additional opts you want to set for atlas service.
#export METADATA_SERVER_OPTS=

# java heap size we want to set for the atlas server. Default is 1024MB
#export METADATA_SERVER_HEAP=

# What is is considered as atlas home dir. Default is the base locaion of the installed software
#export METADATA_HOME_DIR=

# Where log files are stored. Defatult is logs directory under the base install location
#export METADATA_LOG_DIR=

# Where pid files are stored. Defatult is logs directory under the base install location
#export METADATA_PID_DIR=

# where the atlas titan db data is stored. Defatult is logs/data directory under the base install location
#export METADATA_DATA_DIR=

# Where do you want to expand the war file. By Default it is in /server/webapp dir under the base install dir.
#export METADATA_EXPANDED_WEBAPP_DIR=
```

NOTE for Mac OS users
If you are using a Mac OS, you will need to configure the METADATA_SERVER_OPTS (explained above).

In {package dir}/conf/atlas-env.sh uncomment the following line

```
#export METADATA_SERVER_OPTS=
```

and change it to look as below

```
export METADATA_SERVER_OPTS="-Djava.awt.headless=true -Djava.security.krb5.realm= -Djava.security.krb5.kdc="
```

Configuring ATLAS application properties

All configuration in Atlas uses java properties style configuration.

Application Properties

The main configuration file is application.properties which is in the **conf** dir at the deployed location. It consists of the following sections:

Graph Database Configs

Graph persistence engine - BerkeleyDB

Refer link for more details. The example below uses BerkeleyDBJE.

```
atlas.graph.storage.backend=berkeleyje
atlas.graph.storage.directory=data/berkley
```

Graph persistence engine - **Hbase**

Basic configuration

```
atlas.graph.storage.backend=hbase
```

```
#For standalone mode , specify localhost
#for distributed mode, specify zookeeper quorum here - For more information refer http://s3.thinkaurelius.com/docs
/titan/current/hbase.html#_remote_server_mode_2
atlas.graph.storage.hostname=<ZooKeeper Quorum>
```

Advanced configuration

Refer http://s3.thinkaurelius.com/docs/titan/0.5.4/titan-config-ref.html#_storage_hbase

Graph Search Index - ElasticSearch

This section sets up the graph db - titan - to use an search indexing system. The example configuration below setsup to use an embedded Elastic search indexing system.

```
atlas.graph.index.search.backend=elasticsearch
```

```
atlas.graph.index.search.directory=data/es
```

```
atlas.graph.index.search.elasticsearch.client-only=false
```

```
atlas.graph.index.search.elasticsearch.local-mode=true
```

```
atlas.graph.index.search.elasticsearch.create.sleep=2000
```

Graph Search Index - Solr

For Solr, please refer the "**Configuring SOLR as the Indexing Backend for the Graph Repository" section** below.

Hive Lineage Configs

The higher layer services like hive lineage, schema, etc. are driven by the type system and this section encodes the specific types for the hive data model.

# This models reflects the base super types for Data and Process

```
atlas.lineage.hive.table.type.name=DataSet atlas.lineage.hive.process.type.name=Process atlas.lineage.hive.
process.inputs.name=inputs atlas.lineage.hive.process.outputs.name=outputs  ## Schema atlas.lineage.hive.table.
schema.query=hive_table where name=?, columns
```

Security Properties

SSL config

The following property is used to toggle the SSL feature.

```
atlas.enableTLS=false
```

Configuring SOLR as the Indexing Backend for the Graph Repository

By default, Atlas uses Titan as the graph repository and is the only graph repository implementation available currently.
For configuring Titan to work with Solr, please follow the instructions below
* Install solr if not already running. Versions of SOLR supported are 4.8.1 or 5.2.1.

*** Start solr in cloud mode.**
SolrCloud mode uses a ZooKeeper Service as a highly available, central location for cluster management.
For a small cluster, running with an existing ZooKeeper quorum should be fine. For larger clusters, you would want to run separate multiple ZooKeeper quorum with atleast 3 servers.
Note: Atlas currently supports solr in "cloud" mode only. "http" mode is not supported. For more information, refer solr documentation - https://cwiki.apache.org/confluence/display/solr/SolrCloud

*** Run the following commands from SOLR_HOME directory to create collections in Solr corresponding to the indexes that Atlas uses**

```
bin/solr create -c vertex_index -d ATLAS_HOME/conf/solr -shards #numShards -replicationFactor #replicationFactor
bin/solr create -c edge_index -d ATLAS_HOME/conf/solr -shards #numShards -replicationFactor #replicationFactor
bin/solr create -c fulltext_index -d ATLAS_HOME/conf/solr -shards #numShards -replicationFactor #replicationFactor
```

Note: If numShards and replicationFactor are not specified, they default to 1 which suffices if you are trying out solr with ATLAS on a single node instance.
Otherwise specify numShards according to the number of hosts that are in the Solr cluster and the maxShardsPerNode configuration.
The number of shards cannot exceed the total number of Solr nodes in your SolrCloud cluster

*** Change ATLAS configuration to point to the Solr instance setup.**

Please make sure the following configurations are set to the below values in ATLAS_HOME//conf/application.properties

```
atlas.graph.index.search.backend=<'solr' for solr 4.8.1>/<'solr5' for solr 5.2.1>
atlas.graph.index.search.solr.mode=cloud
atlas.graph.index.search.solr.zookeeper-url=<the ZK quorum setup for solr as comma separated value> eg: 10.1.6.4:2181,10.1.6.5:2181
```

For more information on Titan solr configuration , please refer http://s3.thinkaurelius.com/docs/titan/0.5.4/solr.html

---

Starting Atlas Server

```
bin/atlas_start.py [-port <port>]
```

By default,
* To change the port, use -port option.
* atlas server starts with conf from {package dir}/conf. To override this (to use the same conf with multiple atlas upgrades), set environment variable METADATA_CONF to the path of conf dir

Stopping Atlas Server

```
bin/atlas_stop.py
```

Using Atlas

* Verify if the server is up and running

```
  curl -v http://localhost:21000/api/atlas/admin/version
  {"Version":"v0.1"}
```

* List the types in the repository

```
  curl -v http://localhost:21000/api/atlas/types
  {"results":["Process","Infrastructure","DataSet"],"count":3,"requestId":"1867493731@qtp-262860041-0 - 82d43a27-7c34-4573-85d1-a01525705091"}
```

* List the instances for a given type

```
  curl -v http://localhost:21000/api/atlas/entities?type=hive_table
  {"requestId":"788558007@qtp-44808654-5","list":["cb9b5513-c672-42cb-8477-b8f3e537a162","ec985719-a794-4c98-b98f-0509bd23aac0","48998f81-f1d3-45a2-989a-223af5c1ed6e","a54b386e-c759-4651-8779-a099294244c4"]}

  curl -v http://localhost:21000/api/atlas/entities/list/hive_db
```

* Search for entities (instances) in the repository

```
  curl -v http://localhost:21000/api/atlas/discovery/search/dsl?query="from hive_table"
```

Dashboard

Once atlas is started, you can view the status of atlas entities using the Web-based dashboard. \You can open your browser at the corresponding port to use the web UI.