Build and Install

1. How to Build

1.1 Get the code of HAWQ

```
# The Apache HAWQ source code can be obtained from the the following link:
# Apache Repo: https://git.apache.org/repos/asf/hawq.git
# GitHub Mirror: https://github.com/apache/hawq.git
# Gitee Mirror: https://gitee.com/mirrors/hawq.git

git clone https://git.apache.org/repos/asf/hawq.git
```

1.2 Setup an environment with the dependencies installed

1.3 Compile and Install HAWQ

Once you have an environment with the necessary dependencies installed and Hadoop is ready, the next step is to get the code and build HAWQ

```
# The code directory is hawq.
CODE_BASE=`pwd`/hawq

cd $CODE_BASE

# Run command to generate makefile.
./configure

# You can also run the command with --help for more configuration.
./configure --help

# Run command to build and install
# To build concurrently , run make with -j option. For example, make -j8
# On Linux system without large memory, you will probably encounter errors like
# "Error occurred during initialization of VM" and/or "Could not reserve enough space for object heap"
# and/or "out of memory", try to set vm.overcommit_memory = 1 temporarily, and/or avoid "-j" build,
# and/or add more memory and then rebuild.
# On mac os, you will probably see this error: "'openssl/ssl.h' file not found".
# "brew link openssl --force" should be able to solve the issue.
make -j8

# Install HAWQ
make install
```

2. Init/Start/Stop HAWQ

2.1 Install and Start Hadoop

Please follow the steps here: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html

Note:

- you might need to build hadoop from source on Red Hat/CentOS 6.x if the downloaded hadoop package has higher glibc version requirement. When that happens, you will probably see the warning below when running start-dfs.sh." WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform"
- You will also need to set the port for fs.defaultFS to 8020 in etc/hadoop/core-site.xml (The example above set it as 9000.)
- HDFS is a must, but YARN is optional. YARN is only needed when you want to use YARN as the global resource manager.
- must setup passphraseless ssh, otherwise there will be some problems of "hawq init cluster" in the following step.

Your need to verify your HDFS works.

```
# start HDFS
start-dfs.sh

# Do some basic tests to make sure HDFS works
echo "test data" >> ./localfile
hadoop fs -mkdir /test
hadoop fs -put ./localfile /test
hadoop fs -ls /
hadoop fs -get /test/localfile ./hdfsfile
```

2.2 Init/Start/Stop HAWQ

```
# Before initializing HAWQ, you need to install HDFS and make sure it works.

source /hawq/install/path/greenplum_path.sh

# Besides you need to set password-less ssh on the systems.
# Exchange SSH keys between the hosts host1, host2, and host3:
hawq ssh-exkeys -h host1 -h host2 -h host3
hawq init cluster # after initialization, HAWQ is started by default

# Now you can stop/restart/start the cluster by using:
hawq stop/restart/start cluster

# HAWQ master and segments are completely decoupled. So you can also init, start or stop the master and
segments separately.
# For example, to init: hawq init master, then hawq init segment
# to stop: hawq stop master, then hawq stop segment
# to start: hawq start master, then hawq start segment
```

3. Connect and Run basic queries

```
psql -d postgres
create table t ( i int );
insert into t values(1);
insert into t select generate_series(1,10000);
select count(*) from t;
```

4. Query external hadoop data(optional)

You will need to use PXF to query external hadoop/hive/hbase data. Refer to PXF Build & Install document.

5. Test HAWQ

```
# Unit test. To do unit test, go to the src/backend and run unittest.
cd $CODE_BASE/src/backend
make unittest-check

# Code coverage
cd $CODE_BASE
./configure --enable-coverage --enable-debug (for debug build), or ./configure --enable-coverage (for opt build)
make -j8
make install
run some test to exercise hawq (i.e., unit test, install check, feature test, etc)
make coverage-show to see summary code coverage information in console, and detailed code coverage information
in CodeCoverageReport (html format)
make coverage-show filter="./src/backend/executor/nodeAgg.c -d ./src/backend/commands" to see code coverage for
specific files or directories
make coverage-reset to clear code coverage statistics

# Installcheck-good test. After installing HAWQ, please ensure HDFS work before initializing HAWQ.
source /install/dir/greenplum_path.sh
hawq init cluster
make installcheck-good

# Feature test
cd $CODE_BASE
make feature-test
cd src/test/feature
./feature-test to run all feature test, or ./feature-test --gtest_filter=TestCommonLib.TestSqlUtil to run test
suite TestCommonLib.TestSqlUtil
```

6. Running catalog tidycat perl modules(optional)

The JSON Perl Module is required to run the set of Perl scripts (src/include/catalog). The versioned JSON formatted catalog files are stored in tools/bin /gppylib/data/<version>.json. In order to install the JSON module, the developer will need to make the module available from CPAN. The following was validated on a Macbook Pro OS X 10.11.6 using the information from the **Perl on Mac OSX** section (http://www.cpan.org/modules/INSTALL.html). Below you will see the session which performs the following steps:

1. Validate JSON module is not in the environment. Receive appropriate error message.
2. Run **cpan install JSON** command to install the JSON Perl module. In the example below, the module is installed locally (local::lib) and not in the system's Perl installation.
3. Execute the environment variable updates added to the .bashrc file by the installation process.
4. Validate the tidycat.pl command can now be run without receiving error.

Note:

- JSON Module version 2.27 and the latest 2.90 have been used to validate they generate the proper catalog JSON formatted file.
- The scripts are essentially validating the evaluation of *require JSON* passes otherwise the error message is displayed:

    Fatal Error: The required package JSON is not installed -- please download it from www.cpan.org

```
00:02 $ perl tidycat.pl -dd 2.0.json -df json *.h
Fatal Error: The required package JSON is not installed -- please download it from www.cpan.org
00:02 $
00:02 $ cpan install JSON
        [many output stuff....]
00:05 $
00:05 $ perl tidycat.pl -dd foo.json -df json *.h
Fatal Error: The required package JSON is not installed -- please download it from www.cpan.org
00:05 $
00:05 $ PATH="/Users/espino/perl5/bin${PATH:+:${PATH}}"; export PATH;
00:05 $ PERL5LIB="/Users/espino/perl5/lib/perl5${PERL5LIB:+:${PERL5LIB}}"; export PERL5LIB;
00:05 $ PERL_LOCAL_LIB_ROOT="/Users/espino/perl5${PERL_LOCAL_LIB_ROOT:+:${PERL_LOCAL_LIB_ROOT}}"; export
PERL_LOCAL_LIB_ROOT;
00:05 $ PERL_MB_OPT="--install_base \"/Users/espino/perl5\""; export PERL_MB_OPT;
00:05 $ PERL_MM_OPT="INSTALL_BASE=/Users/espino/perl5"; export PERL_MM_OPT;
00:05 $
00:05 $ perl tidycat.pl -dd 2.0.json -df json *.h
00:05 $
```

7. Build optional extension modules(optional)

Extension	How to enable	Pre-build steps on Mac
PL/R	./configure --with-r	#install R before build brew tap homebrew/science brew install r
PL/Python	./configure --with-python	
PL/Java	./configure --with-java	
PL/PERL	./configure --with-perl	
pgcrypto	./configure --with-pgcrypto --with-openssl	
gporca	./configure --enable-orca	
rps	./configure --enable-rps	brew install tomcat@6