

Apache Ignite Extensions -Modularization

Saikat Maitra Twitter @samaitra Github samaitra



Motivation

- To keep Apache Ignite core modules and extensions modules to have separate release lifecycles.
- Few integrations which are no longer in use can be deprecated.
- Help Apache Ignite community to support core and extensions separately (test, release, fix, continue development).



Project Structure

- Ignite-core
- Memory-Centric Storage, Native Persistence, RDBMS Integration (CacheStore for RDBMS)
- Key-Value APIs
- SQL
- Compute Grid
- Service Grid
- Machine Learning APIs
- Advanced queries -scan, continuous
- Transactions
- Data Structures and Atomics
- Ignite Messaging
- Core Streaming APIs such as IgniteDataStreamer
- Logging
- Metrics & Tracing framework
- Command line tools and scripts such as Visor and control.sh
- Standard (aka. thick clients) Java, .Net, C++.
- Spring Core needed for configuration needs.

- Ignite-modules
 - Spark Integration
 - SpringData and SpringBoot
 - TensorFlow Integration
 - Cassandra Integration

🔲 Project 👻

- Y 🖿 modules
- > aop [ignite-aop]
- > 📑 apache-license-gen [ignite-apache-license-gen]
- ➤ aws [ignite-aws]
- > 🖿 benchmarks
- > assandra [ignite-cassandra]
- > 📑 clients [ignite-clients]
- > all cloud [ignite-cloud]
- > 📑 codegen [ignite-codegen]
- > 🖿 compatibility
- > **a** compress **[ignite-compress]**
- > Econtrol-utility [ignite-control-utility]
- > 📑 core [ignite-core]
- > hereit dev-utils [ignite-dev-utils]
- > har direct-io [ignite-direct-io]
- 🔉 🖿 extdata
- > 📑 gce [ignite-gce]



Apache Ignite Extensions

- Flink Ignite Flink Streamer consumes messages from an Apache Flink consumer endpoint and feeds them into an Ignite cache.
- Flume IgniteSink is a Flume sink that extracts events from an associated Flume channel and injects into an Ignite cache.
- Twitter Ignite Twitter Streamer consumes messages from a Twitter Streaming API and inserts them into an Ignite cache.
- ZeroMQ Ignite ZeroMQ Streamer consumes messages from a ZeroMQ consumer endpoint and feeds them into an Ignite cache.
- RocketMQ Ignite RocketMQ Streamer consumes messages from an Apache RocketMQ consumer endpoint and feeds them into an Ignite cache.



Apache Ignite Extensions

- Storm Ignite Storm Streamer consumes messages from an Apache Storm consumer endpoint and feeds them into an Ignite cache.
- MQTT Ignite MQTT Streamer consumes messages from a MQTT topic and feeds transformed key-value pairs into an Ignite cache.
- Camel Ignite Camel streamer consumes messages from an Apache Camel consumer endpoint and feeds them into an Ignite cache.
- JMS Ignite JMS Data Streamer consumes messages from JMS brokers and inserts them into Ignite caches.
- Kafka Apache Ignite Kafka Streamer module provides streaming from Kafka to Ignite cache.



The release dependencies





The release process





Apache Ignite Extensions release process





Migration Guidelines

- An extension can be released separately from Apache Ignite core.
- An extension has to be tested with existing testing tools like TeamCity and Travis.
- Each extension is validated against every Apache Ignite core release and a new version of extension to be released along with Apache Ignite code if changes are required.
- Extensions can continue to have their own specific version release and need not aligned with Apache Ignite core release version.



Migration Risks

- Modification of existing build pipeline and testing procedures.
- Release policies have to be updated to ensure that modules & core versions compatibility matrix is updated regularly

Migration Benefits

- Faster release cycles for Apache Ignite Extensions.
- Less overhead for Release Manager when planning Apache Ignite releases.
- Individual extension module can be released independently.
- Less scope for validation and quick testing cycles for releases.

New Apache Ignite Extensions

- Pub-Sub Pub/Sub module is a streaming connector to inject Pub/Sub data into Ignite cache.
- Spring Boot Autoconfigure Apache Ignite Spring Boot Autoconfigure module provides autoconfiguration capabilities for Spring-boot based applications.
- Spring Boot Thin Client Autoconfigure Apache Ignite Client Spring Boot Autoconfigure module provides autoconfiguration capabilities for Spring-boot based applications.



Apache Ignite Extensions - Upgrade

Current Maven - POM

<dependency>

<groupId>org.apache.ignite</groupId>

<artifactId>ignite-flink</artifactId>

<version>2.8.1</version>

</dependency>

New Maven - POM

<dependency>

<groupId>org.apache.ignite</groupId>

<artifactId>ignite-flink-ext</artifactId>

<version>1.0.0</version>

</dependency>



Data loading and Streaming

- Data streamers publish continuous stream of unbounded set of data into Ignite Cluster
- Data get partitioned and distributed evenly between Ignite nodes.
- Streamed data can be processed in parallel.
- Ignite clients can also perform concurrent SQL queries in data.





More Info

• https://cwiki.apache.org/confluence/display/IGNITE/IEP-36:+Modularization



