LivyProposal

this proposal has now been put to a vote, please do not edit

Abstract

Livy is web service that exposes a REST interface for managing long running Apache Spark contexts in your cluster. With Livy, new applications can be built on top of Apache Spark that require fine grained interaction with many Spark contexts.

Proposal

Livy is an open-source REST service for Apache Spark. Livy enables applications to submit Spark applications and retrieve results without a co-location requirement on the Spark cluster.

We propose to contribute the Livy codebase and associated artifacts (e.g. documentation, web-site context etc) to the Apache Software Foundation.

Background

Apache Spark is a fast and general purpose distributed compute engine, with a versatile API. It enables processing of large quantities of static data distributed over a cluster of machines, as well as processing of continuous streams of data. It is the preferred distributed data processing engine for data engineering, stream processing and data science workloads. Each Spark application uses a construct called the SparkContext, which is the application's connection or entry point to the Spark engine. Each Spark application will have its own SparkContext.

Livy enables clients to interact with one or more Spark sessions through the Livy Server, which acts as a proxy layer. Livy Clients have fine grained control over the lifecycle of the Spark sessions, as well as the ability to submit jobs and retrieve results, all over HTTP. Clients have two modes of interaction: RPC Client API, available in Java and Python, which allows results to be retrieved as Java or Python objects. The serialization and deserialization of the results is handled by the Livy framework. HTTP based API that allows submission of code snippets, and retrieval of the results in different formats.

Multi-tenant resource allocation and security: Livy enables multiple independent Spark sessions to be managed simultaneously. Multiple clients can also interact simultaneously with the same Spark session and share the resources of that Spark session. Livy can also enforce secure, authenticated communication between the clients and their respective Spark sessions.

More information on Livy can be found at the existing open source website: http://livy.io/

Rationale

Users want to use Spark's powerful processing engine and API as the data processing backend for interactive applications. However, the job submission and application interaction mechanisms built into Apache Spark are insufficient and cumbersome for multi-user interactive applications.

The primary mechanism for applications to submit Spark jobs is via spark-submit (http://spark.apache.org/docs/latest/submitting-applications.html), which is available as a command line tool as well as a programmatic API. However, spark-submit has the following limitations that make it difficult to build interactive applications:
It is slow: each invocation of spark-submit involves a setup phase where cluster resources are acquired, new processes are forked, etc. This setup phase runs for many seconds, or even minutes, and hence is too slow for interactive applications. It is cumbersome and lacks flexibility: application code and dependencies have to be pre-compiled and submitted as jars, and can not be submitted interactively.

Apache Spark comes with an ODBC/JDBC server, which can be used to submit SQL queries to Spark. However, this solution is limited to SQL and does not allow the client to leverage the rest of the Spark API, such as RDDs, MLlib and Streaming.

A third way of using Spark is via its command-line shell, which allows the interactive submission of snippets of Spark code. However, the shell entails running Spark code on the client machine and hence is not a viable mechanism for remote clients to submit Spark jobs.

Livy solves the limitations of the above three mechanisms, and provides the full Spark API as a multi-tenant service to remote clients.

Since the open source release of Livy in late 2015, we have seen tremendous interest among a diverse set of application developers and ISVs that want to build applications with Apache Spark. To make Livy a robust and flexible solution that will enable a broad and growing set of applications, it is important to grow a large and varied community of contributors.

Initial Goals

- Move existing codebase, website, documentation and mailing lists to Apache-hosted infrastructure
- Work with the infrastructure team to implement and approve our code review, build, and testing workflows in the context of the ASF
- Incremental development and releases per Apache guidelines

Current Status

The Livy project began at Cloudera, as a part of the Hue project. Cloudera soon realized the broad applicability of Livy, and separated it out into an independent project in Nov 2015.

Releases

Livy has undergone two public releases, tagged here:

- https://github.com/cloudera/livy/releases/tag/v0.2.0
- https://github.com/cloudera/livy/releases/tag/v0.3.0

Tarballs and zip files were created for each release and hosted on github. Upon joining the incubator, we will adopt a more typical ASF release process.

Source

Livy's source is currently hosted on Github at: https://github.com/cloudera/livy This repository will be transitioned to Apache's git hosting during incubation.

Code review

Livy's code reviews are currently public and hosted on github as pull request reviews at: https://github.com/cloudera/livy/pulls The Livy developer community so far is happy with github pull request reviews and hopes to continue this after being admitted to the ASF.

Issue Tracking

Livy's bug and feature tracking is hosted on JIRA at: https://issues.cloudera.org/projects/LIVY/summary This JIRA instance contains bugs and development discussion dating back 1 year and will provide an initial seed for the ASF JIRA

Community Discussion

Livy has several public discussion forums:

- https://groups.google.com/a/cloudera.org/forum/#!forum/livy-dev
- https://groups.google.com/a/cloudera.org/forum/#!forum/livy-user

Development Practices

The Livy project follows a review before commit philosophy. Every commit automatically runs through the unit tests and generates coverage reports presented as a pull request comment. Our experience with this process leads us to believe that it helps ease new contributors into the project. They get feedback quickly on common mistakes, lowering the burden on reviewers. Those same reviewers get to lead by example, showing the new contributors that we value feedback within our community even when changes are done by more experienced folks.

Meritocracy

We believe strongly in meritocracy when electing committers and PMC members. In the past few months, the project has added two new committers from two different organisations, in recognition of their significant contributions to the project. We will encourage contributions and participation of all types, and ensure that contributors are appropriately recognized.

Community

Though Livy is relatively new as a standalone open source project, it has already seen promising growth in its community across several organizations: Cloudera is the original development sponsor for Livy Microsoft pushed the development of the interpreter fixing high availability issues and adding additional features. Hortonworks has contributed the security features to Livy allowing kerberos and impersonation to work with Spark IBM is starting to make contributions to the Livy project A number of other patches contributed by community members

Livy currently relies on Google Groups for mailing lists. These lists have been active since the end of 2015/start of 2016. Currently, Livy's user mailing list has 173 subscribers and has hosted a total of 227 topic threads. Livy's developer list has 49 subscribers and has hosted 79 topic threads.

Core Developers

The early contributions to Livy were made by Cloudera engineers. In 2016, engineers from Microsoft and Hortonworks joined the core developer community.

Alignment

Livy is built upon Apache Spark, and other Apache projects like Apache Hadoop YARN. It's used as a building block by Apache Zeppelin. These community connections combined with our focus on development practices that emphasize community engagement with a path to meritocratic recognition naturally align us with the ASF.

Known Risks

Orphaned Products

The risk of Livy being abandoned is low because it is supported by three major big-data software vendors. Moreover, Livy is already used to power multiple releases of services and products used in production.

Inexperience with Open Source

Several of the initial committers are experienced open source developers, several being committers and/or PMC members on other ASF projects (Spark, YARN).

Homogenous Developers

The project already has a diverse developer base. It has contributions from 3 major organisations (Cloudera, Microsoft and Hortonworks), and is used in diverse applications, in diverse settings (On-Prem and Cloud).

Reliance on salaried Developers

The contributions to the Livy project to date have been made by salaried engineers from Cloudera, Microsoft and Hortonworks. One of the individuals on the initial committer list has since left Microsoft and is currently unaffiliated. The remaining contributors are from Cloudera and Hortonworks. Since there are at least two major organizations involved, the risk of reliance on a single group of salaried developers is mitigated. The Livy user base is diverse, with users from across the globe, including users from academic settings. We aim to further diversify the Livy user and contributor base.

Relationships with other Apache projects

Livy is closely tied to the Apache Spark project and currently addresses the scenarios for a REST based batch and interactive gateway for Spark jobs on YARN. Given the growing number of integrations with Livy, keeping it outside of Apache Spark aligns with the desire of the Apache Spark community to reduce the number of external dependencies in the Spark project. Specifically, the Apache Spark community has previously expressed a desire to keep job servers independent from the project. [1] Furthermore, while Livy common usage is closely tied to Spark deployments right now, its core building blocks can be reused elsewhere. Livy's Remote REPL could be used as a library for interactive scenarios in non-Spark projects. In the future, integrations with cluster managers like Apache Mesos and others could also be added.

The features provided by Livy have already been integrated with existing projects like Jupyter and Apache Zeppelin for their interactive Spark use cases. This validates the need for a project like Livy and provides an active downstream user base that the Livy community can interact with to seed future interest in the project.

Livy serves a similar purpose to Apache Toree (incubating) but differs in making session management, security and impersonation a focal design point.

An Excessive Fascination with the Apache Brand

The primary motivation for submitting Livy to the ASF is to grow a diverse and strong community. We wish to encourage diverse organisations, including ISVs, to adopt Livy and contribute to Livy without any concerns about ownership or licensing.

Documentation

Documentation can be found on the Livy website http://livy.io/
The Livy web site is version controlled on the 'gh-pages' branch of the above repository Additional documentation is provided on the github wiki: https://github.com/cloudera/livy/wiki APIs are documented within the source code as JavaDoc style documentation comments.

Initial Source

The initial source code for Livy is hosted at https://github.com/cloudera/livy

Source and Intellectual Property submission plan

The Livy codebase and web site is currently hosted on GitHub and will be transitioned to the ASF repositories during incubation. Livy is already licensed under the Apache 2.0 license. Cloudera has collected ICLAs and CCLAs from all committers. There are, however, some contributions recently from authors that have not signed the CCLA and ICLA. If necessary for a successful SGA, we'll seek the necessary documentation or replace the contributions.

The "Livy" name is not a registered trademark. We will need to do a trademark search and make sure it is available for the Apache Foundation prior to graduation.

Cloudera currently owns the domain name: http://livy.io/. Once all the documentation has moved over to ASF infrastructure, the main landing page will become livy.incubator.apache.org and the old domain will just act as a redirect.

External Dependencies

The list below covers the non-Apache dependencies of the project and their licenses.

- Jetty: Apache 2.0
- Dropwizard Metrics: Apache 2.0
- FasterXML Jackson: Apache 2.0
- Netty: Apache 2.0
- Scala: BSD
- Py4J: BSD
- Scalatra: BSD

Build/test-only dependencies:

- Mockito: MIT
- JUnit: Eclipse

Required Resources

Mailing Lists

- private@livy.incubator.apache.org (PPMC)
- dev@livy.incubator.apache.org (dev mailing list)
- user@livy.incubator.apache.org (User questions)
- commits@livy.incubator.apache.org (subscribers shouldn't be able to post)
- issues@livy.incubator.apache.org (subscribers shouldn't be able to post)

Git Repository

git://git.apache.org/incubator-livy

Issue Tracking

We would like to import our current JIRA project into the ASF JIRA, such that our historical commit message and code comments continue to reference the appropriate bug numbers.

Initial Committers

- Marcelo Vanzin (vanzin@cloudera.com)
- Alex Man (alex@alexman.space)
- Jeff Zhang (zjffdu@gmail.com)
- Saisai Shao (sshao@hortonworks.com)
- Kostas Sakellis (kostas@cloudera.com)

Affiliations

The initial set of committers includes people employed by Cloudera and Hortonworks as well as one currently independent contributor.

Additional Interested Contributors

Those interested in getting involved with the project as we enter incubation are encouraged to list themselves here.

- Ismaël Mejía (iemejia@apache.org)

Sponsors

Champion

Sean Busbey (busbey@apache.org)

Nominated Mentors

- Bikas Saha (bikas@apache.org)
- Brock Noland (brock@phdata.io)
- Luciano Resende (lresende@apache.org)

Sponsoring Entity

We ask that the Incubator PMC sponsor this proposal.

1. See, for example, discussion of the Ooyala Spark Job Server in SPARK-818 ↩