Cello

2016-10-01

# HIP identifier

Blockchain as a Service for Hyperledger

# Sponsor

*Baohua Yang*

IBM Research

Building 19A, ZhongGuanCun Software Park

Beijing, China, 100193

[yangbaohua@gmail.com](mailto:yangbaohua@gmail.com)

*Haitao Yue*

IBM Research

Building 19A, ZhongGuanCun Software Park

Beijing, China, 100193

[bjhtyue@cn.ibm.com](mailto:bjhtyue@cn.ibm.com)

*Makoto Takemiya*

Soramitsu Co., Ltd.

1-7-3 No. 2 Watanabe Blg. 7F, Minato-ku,

Tokyo, Japan 106-0044

[takemiya@soramitsu.co.jp](mailto:takemiya@soramitsu.co.jp)

*Zhipeng Huang*

Huawei Co., Ltd

Industrial Base Bantian, Longgang 518129.

Shenzhen, Guangdong China

[huangzhipeng@huawei.com](mailto:huangzhipeng@huawei.com)

*Ryan Beck-Buysse*

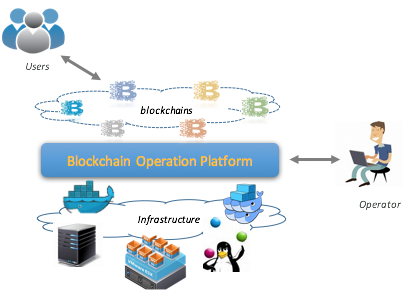
Intel

[ryanx.beck-buysse@intel.com](mailto:ryanx.beck-buysse@intel.com)

# Abstract

[*Cello*](https://github.com/yeasy/cello) aims to help provide a Blockchain as a Service (BaaS), which can reduce the effort required for manipulating (e.g., create and destroy) chains manually. With Cello, *operators* can create and manage multiple blockchains in a pool through a dashboard, at the same time *users* (typically the chaincode developers) can obtain blockchains instantly with a single request, as illustrated in the figure below.

Cello will support existing Hyperledger blockchain implementations including [Fabric](https://github.com/hyperledger/fabric), [SawToothLake](https://github.com/hyperledger/sawtooth-core) and [Iroha](https://github.com/hyperledger/iroha). We have been evaluating Cello in several environments, e.g., Cello in a [POWER-based Cloud](https://crl.ptopenlab.com:8800/bc) has supported thousands of chains for over half a year.



# Context

The Hyperledger community by Linux Foundation has initialized several ledger projects (e.g., [Fabric](https://github.com/hyperledger/fabric), [SawToothLake](https://github.com/hyperledger/sawtooth-core) and [Iroha](https://github.com/hyperledger/iroha)) for the Decentralized Ledger Technology (DLT) ecosystems. Those projects provide various ledger implementations targeting performance, stability, permissions, scalability, etc. Cello hopes to help build the community by providing the blockchain service functionality and attracting more contributors into the Hyperledger project.

Today, to boot a chain, developers needs to adopt the installation scripts, e.g., docker-compose scripts in Fabric. If multiple tenants requires to obtain separate chains at the same time, they have to modify the scripts carefully and create these chains manually. This procedure is time consuming, and even worse, leads to possible misconfigurations as discussed in Slack channels.

The initial focus of the Cello project is to bring the Cloud service model into Blockchain ecosystem, to provide a multi-tenant chain service efficiently and automatically.

# Motivation

Taking Hyperledger Fabric as an example, currently, the solution is to create a Hyperledger Fabric chain, which includes:

* Manual installation of each peer node on different servers. This requires much effort and is error prone.
* Setup scripts (e.g., Docker-Compose) to start a fabric network. This requires a specific server configuration, which makes it hard to share resources and dynamically create multiple chains.

Cello solves these problems in a different way, by maintaining a pool of chains automatically. Users will get chains with various configurations instantly, while operators can dynamically scale the physical resources through a dashboard.

The Hyperledger community now has projects of [SDK](https://wiki.hyperledger.org/groups/fabric-sdk/fabric-sdk-wg), [blockchain-explorer](https://github.com/hyperledger/blockchain-explorer) and [chaintool](https://github.com/hyperledger/fabric-chaintool). Cello can be a good complement. For example, Cello can boot a blockchain with blockchain-explorer as the dashboard, with SDK and chaintool as the interface to operate chaincode.

# Status

The work started in May, 2016, and is adopted and evaluated in several services. Currently it is still under development, and the latest source code is at [Github](https://github.com/yeasy/cello).

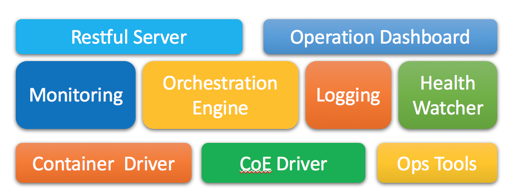
Several possible ongoing todo tasks include:

1. Support more efficient scheduling algorithms like feedback loops based.
2. Refine the web portal with [react.js](https://github.com/reactjs) for better performance.
3. Support more cluster platforms like [Kubernetes](http://kubernetes.org) and [Mesos](http://mesos.org).
4. Support other blockchain platform solutions, e.g., [SawToothLake](https://github.com/hyperledger/sawtooth-core) and [Iroha](https://github.com/hyperledger/iroha) (with helps from the community).
5. Support Hyperledger Fabric 1.0 code, which will release in Mar, 2017.

# Solution

Cello leverages the Docker APIs to manage the blockchain clusters in remote hosts, including physical servers and virtual machines. Hence Cello can be easily deployed to Cloud environments that provide virtual machines on demand. More details can be found at the [deployment documentation](https://github.com/yeasy/cello/blob/master/docs/deployment.md).

The design architecture is as follows.



* Orchestration Engine: Core to handle resource management and workload scheduling, which is mainly implemented in Python;
* Dashboard: Operational interface, implemented with JavaScript.
* Restful Server: Operational interface, which is implemented with Python.
* Drivers: Currently we utilize Docker API lib, to support native host and [Swarm](https://docs.docker.com/swarm/) cluster. The driver layer is designed to be pluggable to support more types in future.
* Tools: We have also designed several tools to do the tasks like Monitoring and logging, which are mainly implemented in Golang. However, the framework is pluggable, hence we can also integrate existing open-source tools.

The license is Apache License v2.0.

# Effort and resources

Currently two persons (Baohua Yang and Haitao Yue, IBM Research) are committed part-time to developing and maintaining the project. This is not the main thrust of their job however. To realize its full potential, other developers and test engineers are encouraged to involve.

# How to

The project is now hosted at [github](http://github.com/yeasy/cello).

The CI is tracked by [travis-ci](https://travis-ci.org/), while we will follow the Hyperledger community’s guide for the CI once the proposal gets accepted.

Currently we have some basic test cases. More test cases will be added.

Documentation is well written, with [README](https://github.com/yeasy/cello/blob/master/README.md), [guidelines](https://github.com/yeasy/cello/tree/master/docs) on deployment, operations and developments, and also detailed code documentation.

# Closure

The project will succeed if people use it in their services, and find it helpful. We can use some acknowledgement page to track the projects that integrate this code in their implementations.

# Acknowledgement

Thanks for all the insightful comments from all reviewers.

The names are collected in the below table, every review please feel free to add yours.

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Organization | Email |
| 1 | Muhammad Altaf | Fujitsu Australia Software Technology Pty Ltd | [Muhammada@fast.au.fujitsu.com](mailto:Muhammada@fast.au.fujitsu.com) |
| 2 | Tianqing Wang | MaDaiLiCai | [wang.tianqing.cn@gmail.com](mailto:wang.tianqing.cn@gmail.com) |
| 3 | Zhipeng Huang | Huawei | [huangzhipeng@huawei.com](mailto:huangzhipeng@huawei.com) |
| 4 | Kewei Sun | IBM | [sunkewei@cn.ibm.com](mailto:sunkewei@cn.ibm.com) |
| 5 | Makoto Takemiya | Soramitsu | [takemiya@soramitsu.co.jp](mailto:takemiya@soramitsu.co.jp) |
|  |  |  |  |