Fabric Composer

2017-03-07

# HIP identifier

Fabric Composer

# Sponsors

* Simon Stone, IBM ([sstone1@uk.ibm.com](mailto:sstone1@uk.ibm.com))
* Kathryn Harrison, IBM ([harriska@us.ibm.com](mailto:harriska@us.ibm.com))
* Cong Tang, Oxchains ([tangcong@oxchains.com](mailto:tangcong@oxchains.com))

# Abstract

An application development framework that accelerates the formation of business networks and the development of blockchain or distributed ledger applications.

# Context

Fabric Composer, or Composer for short, is a layer of abstraction that runs on top of a blockchain or distributed ledger. Composer itself is not an implementation of a blockchain or distributed ledger.

All work done on Composer to date has been done on top of Hyperledger Fabric. However, Composer has been designed such that it can be ported to run on other blockchain or distributed ledger technologies, such as Iroha or Sawtooth Lake.

# Motivation

As we worked to develop and deploy blockchain applications in hundreds of POCs, we quickly understood that all blockchain use cases share certain elements common to business networks - namely assets, participants, identities, transactions, and registries. With existing blockchain or distributed ledger technologies, it can difficult for organizations to take a blockchain use case and map the business concepts into working chaincode.

The primary motivation of the Composer project is to accelerate the formation of business networks and the development of blockchain applications. Composer is a layer of abstraction that provides business analysts and blockchain application developers with a framework that they can use to easily and quickly model their business networks.

# Status

The implementation effort started in September 2016. The code is hosted on public GitHub, and all of the code has been licensed under the Apache v2.0 license.

# Solution

Composer contains a set of code libraries and tools for developing business networks, deploying them, and integrating running business networks with existing business processes.

The main components are:

* The modelling language; a simple but expressive business centric (the language features keywords such as **asset** and **participant**) language that allows business analysts and developers to model their business network. The modelling language also supports modelling of relationships and data validation rules.
* The ability to encode business logic as transaction processor functions that are written in standard JavaScript. We chose JavaScript because it is a modern, rapidly evolving programming language that is used by millions of developers around the world, as well as giving us the ability to run the code anywhere that supports standard JavaScript.
* Declarative access control using access control lists, that allows developers to describe what resources can be accessed by which participants. Access control is automatically enforced by the runtime.
* Client and administrative APIs, as well as a “composer” CLI application that allows developers and operators to deploy and interact with business networks from Node.js applications or automation scripts.
* A web based “playground” that allows new and experienced users to learn the language, model their business network, and test that business network from the comfort of their web browser. The playground can work in both “disconnected” mode, using a simulated blockchain,and when connected to real running business networks.
* REST API support and integration capabilities; a LoopBack connector for business networks has been developed that exposes a running business network as a business centric REST API which can easily be consumed by client applications.
* Syntax highlighting support for two popular open-source editors, Atom and VS Code, with future plans about how we could include testing/debugging capabilities.
* Application generation using the Yeoman framework; client application developers can quickly generate a skeleton Angular 2 or CLI application to use as a starting point, allowing them to focus on UI/UX rather than business network interactions.

We believe that by providing such an open development toolset, Composer can accelerate the development and deployment of business networks running on top of blockchain or distributed ledger technology.

We have open sourced all of the above Composer code under the Apache 2.0 license, and our proposal is to submit all of the above Composer code to the Hyperledger Project as an incubator project.

Future work that we are currently considering includes:

* First class support for events; being able to model the structure of events, being able to publish events from a business network and allowing client applications to subscribe to and receive those events.
* Links between multiple running business networks; for example, allowing one business network to reference assets that are stored in a business network that has been deployed to the same blockchain or even other blockchains.
* Complex and historical query support, to allow powerful queries over assets, participants, and transactions and the relationships between those resources that have been recorded in a business network.
* Investigate adding support for Iroha and Sawtooth Lake. Composer already includes an pluggable adapter layer that makes this possible. The majority of the Composer codebase is runtime agnostic. The runtime specific code is restricted to the connector implementations (client to blockchain) and the runtime container implementations (blockchain to common runtime).
* Automatic, formal validation of Composer business network definitions.

# Effort and resources

IBM is committing resources to the ongoing development of Composer. The resources include engineers, UI/UX designers, and information developers.

Proposed maintainers:

Simon Stone (IBM)

Daniel Selman (IBM)

# How to

Documentation for Composer, including a Getting Started guide, is available from the public website: <https://fabric-composer.github.io>

Composer is currently hosted on public GitHub under its own organization: <https://github.com/fabric-composer>

All of our work, both planned and in-progress, is managed using GitHub issues: <https://github.com/fabric-composer/fabric-composer/issues>

The current list of repositories is:

* fabric-composer - the majority of the code (including APIs, runtime, and UI)
* fabric-composer.github.io - the public website content (built from fabric-composer)
* sample-networks - a collection of sample business network definitions
* sample-applications - a collection of sample applications
* sample-models - a collection of sample model files
* composer-atom-plugin the plugin for the Atom editor
* composer-vscode-plugin - the plugin for the VS Code editor
* tools - additional tools that are not part of the Composer “core”

The majority of the Composer source code is JavaScript, and can be built and tested using the standard npm toolchain (“npm install” and “npm test”). We have aimed to use standard JavaScript tools as part of our development process: npm, Mocha (test driver), Chai (test assertions), Sinon (mocks), Istanbul (code coverage), etc.

Our CI builds are automated using Travis CI and run automatically on pull requests, merge builds and release builds. The automated builds run all unit and system tests, as well as enforcing linting rules and minimum code coverage levels.

We automatically (as part of our CI builds and release process) publish the Composer modules to npm: <https://www.npmjs.com/search?q=fabric+composer>

We automatically build and publish Docker images to Docker Hub: <https://hub.docker.com/u/fabriccomposer/>

We automatically deploy a public instance of the Composer Playground to Bluemix: <http://fabric-composer.mybluemix.net>

# References

None

# Closure

By working with the community, we would like to continue to develop Composer to be a powerful and complete development framework that allows users to easily and rapidly build blockchain/distributed ledger applications.