Color Spectrum:

Next Paradigm for Decentralized Apps

Table of Contents

Table of Contents 1

Introduction 1

Color Spectrum Overview 3

Two-tier Architecture of Color Spectrum 4

Clouds in Color Spectrum 4

- Logic Runner 5
- Invoking dApps 6
- Component of Cloud Infrastructure 8

Bridging Cloud with Blockchain 10

Minimal Viable Product 10

DApp development and deployment 11

Introduction

Color Spectrum is a powerful new technology that leverages the blockchain in a robust cloud environment for an optimal balance of performance and security.

Motivation

All modern blockchain technologies face several major issues:

1. **Simplicity of smart contracts**. Blockchains have inherent limitations on duration of code execution.
2. **Sequential execution of transactions** in sequential blocks downgrade the whole network to the power of a single computer, that produces the blocks.

Color Spectrum is devoted to overcome these limitations to equip developers with powerful coding tools and provide them with an efficient execution environment.

Decentralized Applications, dApp - is the new buzzword, that was coined along with blockchain and distributed ledger. “Decentralized” means that neither user, nor even developer know where exactly in the infrastructure an application runs. This execution model was called “fog computation”, similar to cloud computing.

When we designed Color Spectrum, we kept in mind the limitations induced by blockchain design, but still keeping the value of decentralized execution of dApps.

Many blockchain projects deemed to overcome the shortcomings of dApp approach, first introduced in Ethereum. Generate blocks faster to increase performance (e.g. EOS), introduce block DAG instead of chain to make concurrent execution (e.g. RChain). But they all keep the original feature of “fog computing” - calls to dApps are mixed with data, and code is executed by miners (or block producers) in sequence while building a block.

But running the code while building blocks is the root of all limitations on smart contract execution. The time duration of code execution is limited, otherwise block build time becomes prohibitively long. Execution is sequential and is performed by a single computer - the one, that builds the block at the moment. The limitations are inherent in the blockchain design, especially in code execution.

To overcome these limitations Color Spectrum took completely other way. In Color Spectrum we decoupled code execution from storing data in the irreversible and non-modifiable history (i.e. the blockchain).

The key point of Color Spectrum is the following: **the code of dApps is executed before transactions are written to the blockchain**.

To save the value of decentralization Color Spectrum uses technologies of cloud computing to conceal details of dApp deployment from the outer world. Furthermore, cloud computing infrastructure gives the value of concurrent execution and efficient utilization of computing resources.

Cloud infrastructure isolates execution of dApps while still running them on the same platform. Container-based clouds provide small start-up time of service nodes and deliver the property of serverlessness. On top of that, containers consume almost zero resources until they are invoked to handle actual requests.

Color Spectrum Overview

The key difference between Color Spectrum and many other blockchain platforms that support smart contracts execution (e.g. Ethereum, NEO, EOS, and others) is the way how transactions are executed. In other platforms typical lifecycle of a transaction includes the following stages:

- **Build Block:** Clients propose their transactions to a miner / bockbuilder and the network disseminates it among all nodes, where one of the nodes builds a block using the transactions.
- **Execute:** Transactions are sequentially executed one by one during construction of a block by a miner / block builder and re-executed by other nodes to verify state update that the block builder saved as a result of the block.

In order for all peers to synchronize the state, transactions must always execute deterministically: the same transaction must always create the same result on each node, no matter whether the block is being constructed or validated.

Furthermore, transaction execution time should be as short as possible. Sophisticated code, with rich business logic takes longer time to execute, thus introducing latencies and delays into block construction. This is the reason why many blockchain platforms put limitations on smart contract execution - either by higher fees for CPU time (like gas in Ethereum) or by limiting CPU time (like 30ms in EOS).

Color platform introduces a new paradigm of smart contracts Execute-Validate-Build block:

- **Execute:** Transactions are executed in parallel. Developers use well-known and powerful languages to code their dApps - Python, Javascript, Java.
- **Validate:** block builders check the results of transaction execution during block construction and consensus.
- **Build Block:** validated transactions are saved in blocks one by one.

Transactions in Color Spectrum are executed before they are put in a block. This allows nodes to execute transactions in parallel, greatly improving throughput.

In the Color Spectrum execution model, the results of executing dApp code for a transaction are explicitly agreed upon before the transaction is added to the ledger.

Color Spectrum focuses on providing developers with advanced coding practices. They can develop dApps in almost any language they like, such as Python, Javascript, Java or even C.

Two-tier Architecture of Color Spectrum

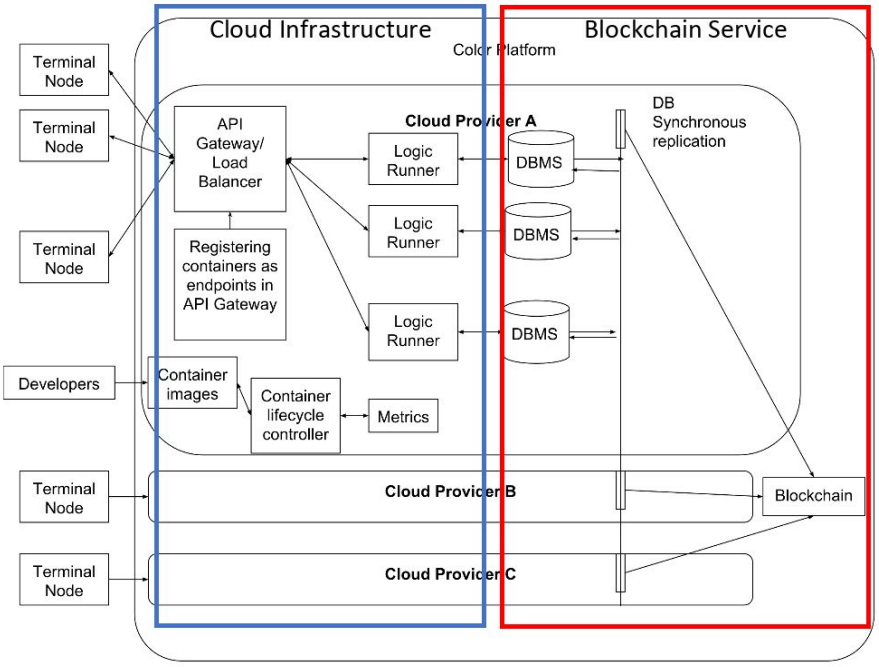

Figure 1. Two tiers of Color Spectrum

Decoupling the transaction execution from blockchain data storage leads to a two-layer architecture. In the first layer Logic Runners are introduced, which execute the business logic of decentralized applications. The second layer implements blockchain operations for dApps.

Clouds in Color Spectrum

Running dApps in Logic Runners implies a number of requirements on the infrastructure that hosts them.

We believe that developers should be able to implement the business logic of their dApps in well-known programming languages, such as Python, Java, PHP, Javascript.

To prevent code injection and data modification, and ensure decentralization, the Infrastructure should be able to launch dApp code at different physical locations, even while handling subsequent requests. The infrastructure should monitor resources consumption and launch Logic Runners on demand from clients.

Modern cloud infrastructures meet all the requirements listed above. They are language-agnostic and execute code in any language thanks to containers. Clouds monitor running containers and launch them at arbitrary nodes within their

infrastructure (this is why they are called “clouds”) and implement various resource allocations policies.

The two-tier architecture of Color Spectrum resembles Hyperleger Fabric, but Color Spectrum makes a great step forward. Hyperleger Fabric has a concept of “Endorsers”, the nodes that execute a transaction before building the blocks. They name it “execute-order” strategy. The Hyperleger Fabric introduces their own policies and protocols to manage endorsers, support a very limited set of software technologies, but provide almost no protection against cartels and code modification aimed at endorsers. The cloud architecture in the Color Spectrum is much more advanced, flexible, reliable and stable technology.

Logic Runner

Logic Runners are at the core of the Color Spectrum. They host the business logic of dApps, perform processing of users’ operations, and store results to the underlying blockchain. Business Logic of a dApp, hosted by a Logic Runner, is executed within a devoted environment, provided by the Color Virtual Machine (CVM).

CVM provides execution environment for dApp code including language environments, system software, middleware to communicate with API layer, storage and blockchain.

The first release of Color Platform will include CVM capable running applications in Java, Javascript, Python and (being considered) C/C++. A large part of our resources will be devoted to this effort.

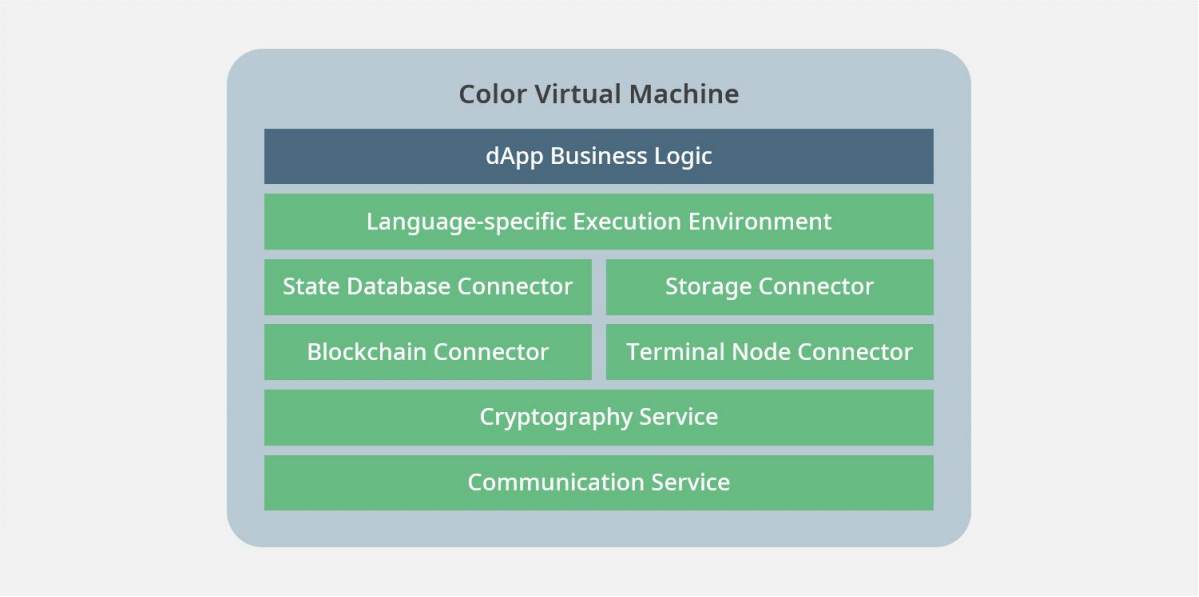

Figure 2 Logic Runner

Language-specific Execution Environment provides the code base necessary to run dApp business logic. For Java apps it is Java Runtime Environment, for Javascript - Node.js or Web-assembly runtime, for C++ - necessary runtime libraries.

Connectors

dApps in the Logic Runners need to communicate with the environment. Such means are provided by connectors. While all connectors are the same for all languages, each runtime environment provide language-specific bindings.

State Database Connector allows dApp code to keep persistent variables between invocations, such as wallet balance, asset owners, existing bets and so on. Through connector dApps read and write values to state database.

Storage Connector provides dApps the capability to access distributed storage provided by the platform. Using this connector dApps can create files, write to them, read files or get status information. Color Platform will provide secure file vault for personal data, and will be accessed through this connector in a secure and efficient way.

Blockchain Connector allows low-level access to the blockchain, for reading. This connector is designed for block viewers, analysis tools and other other facilities.

Color Platform doesn't allow direct write access to the blockchain since it might ruin synchronization between the Ledger and the State Database. All writes to the blockchain should go through transactions to the State Database.

Terminal Node Connector supports the communication channel between dApp and terminal nodes. It receives requests from terminal nodes and sends back responses from dApp. It could be used to implements push notifications or complex message exchanges in AJAX style.

When dApp developers want to protect certain data types in their applications, the ***Cryptography Service*** will allow for the encryption and decryption of critical data. The file service, meanwhile, will store dApp data in a distributed decentralized network storage system such as IPFS or BigchainDB. These services will work together to mitigate the difficulties of storing off-chain app logic in a decentralized way.

Communication Service implements low-level network operations that all other connectors use. It provides transport and P2P protocols, secure channels and other network-related operations required by connectors.

Invoking dApps

The Internet works behind curtains. The average Internet user doesn't understand how data packets are routed from Sydney to Tokyo — the Internet should just work.

The Color Spectrum intends to make blockchain accessible to the average user by simplifying the complexities and separating the back end for developers. When launching a dApp, Color developers won't have to worry about orchestrating a complex smart contract or designing a decentralized storage system.

Color Spectrum provides unified Application Interface for application developers to invoke their dApps hosted by cloud infrastructures. Load Balancing distributes tasks to Logic Runners that execute dApps code and store results in the underlying blockchain.

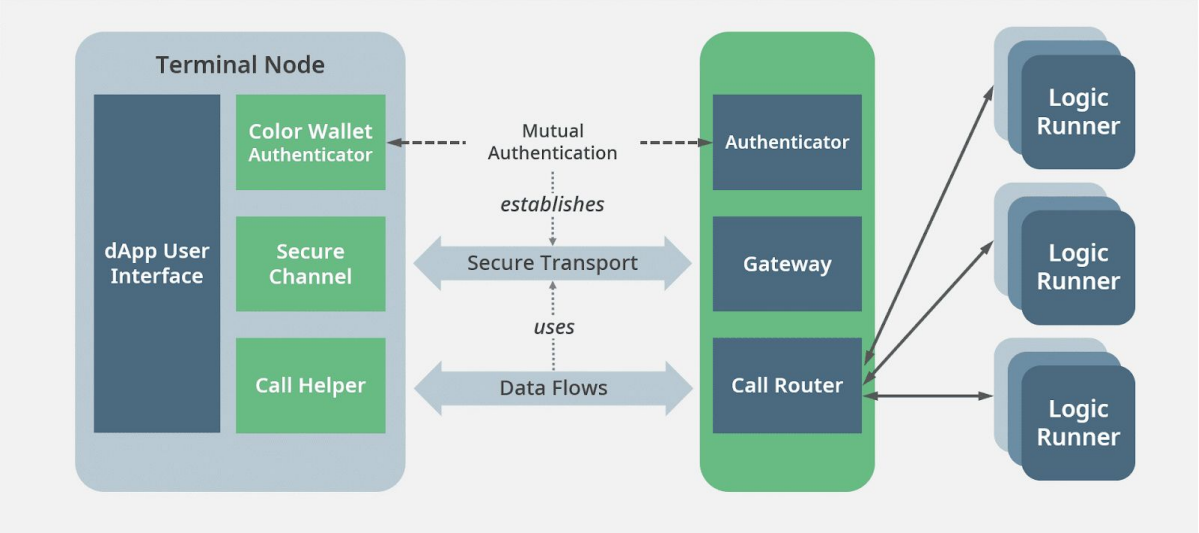

Figure 3 Invocation of dApps

In Color Spectrum the business logic of dApps is implemented within Logic Runners. Application Protocol Interface is the gateway that encapsulates the Logic Runners from the outer world and routes data between Terminal Nodes and Logic Runners.

This component authenticates wallets that users' utilize for transactions, establishes secure connection between Terminal Nodes and Color Platform.

Call Router is key to operations of the platform. It receives requests from terminal nodes, and selects the Logic Runner to serve the request. Sophisticated algorithms of load balancing, load distribution, and load accounting ensure high performance of the Platform and minimize the risk of data races.

Authentication

Each data stream from a Terminal Node to the Color Platform must carry a distinctive identifier of a user that initiated it. The Color Platform utilizes wallet authentication - the data streams are attributed with wallet ID, which is proved by a corresponding private key during authentication phase.

Authentication doesn't mean that a user needs to enter personal ID or password. It's major purpose is to ensure that all operations executed on behalf a specific wallet do originate from that wallet.

Modern blockchains use digital signatures to ensure authenticity of requests, but Color Platform targets much higher throughput. This is why we use secure channels, protected with modern cryptography, to exchange data with Terminal Nodes. The secure channel is established on mutual authentication, and the platform treats all data transported within it as authenticated. Still requests that are know to result in blockchain write must be explicitly signed by the Terminal Node.

Coin Transfer API

Color Platform provides dApps with facilities to pay and receive payments in Color Coins or COL-based tokens. The Platform API includes dedicated calls to transfer coins, lock them in a specific wallet, transfer between two wallets, setup allowance, approve transactions, etc. dApps will obtain rich set of services to build business with Color Coins.

Color Transfer API allows third-party wallets to view balance and perform transfers with Color Coins.

Component of Cloud Infrastructure

In order to host dApp, cloud service providers that partner with Color Platform and host Logic Runners, must deploy a number of components.

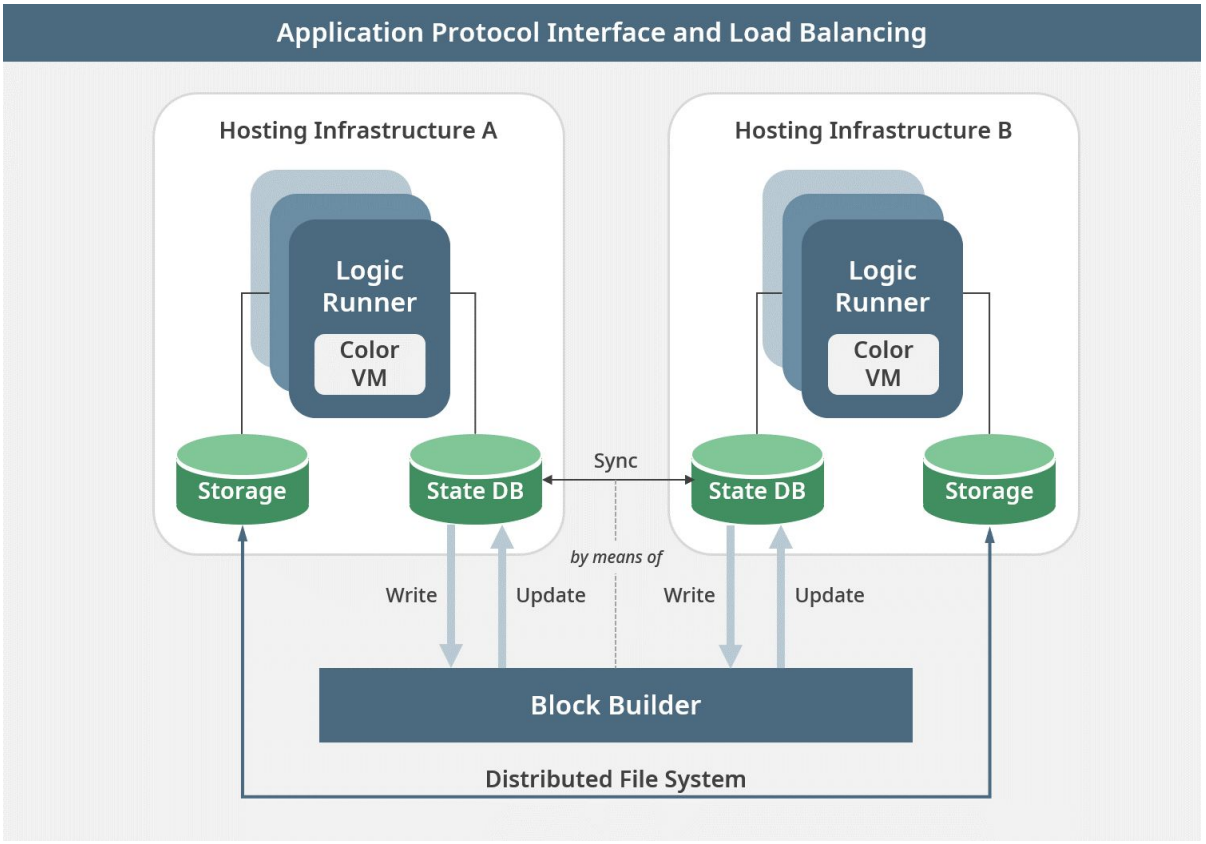

Figure 4 Cloud Components

Application Protocol Interface and Load Balancing

Color Spectrum uses JSON to encapsulate messages between terminal nodes and corresponding Logic Runners, which execute the dApp business logic. Color Spectrum prevents terminal nodes from direct connection to Logic Runners because such direct connection might result significant security risks due to potential misbehaving and compromise of Logic Runners.

Application Protocol Interface is a boundary that encapsulates the platform, protects is from DDoS and (in the future) might be able to provide sophisticated analysis tools to prevent malicious behavior.

Load Balancing distributes requests from terminal nodes to Logic Runners that run corresponding dApp business logic. Each dApp is deployed at multiple Logic Runners. Load Balancing enforces randomization in servicing terminal nodes to prevent risk of running dApp at compromised specific Logic Runner.

Another task of Load Balancing facility is tracking the load of actual Logic Runners. Amount of load is used to count the compensation from the network for running Logic Runners.

State Database and Storage

Like almost all other blockchain-based platform the Color Platform is equipped with a **State Database**. This database stores dApp persistent variables, such as wallet balance or a list of purchases. dApp business logic that runs inside Color VM on Logic Runners connects to the State DB to read and write variables, to update the state of the application.

The State DB is tightly coupled with **Block Builder** that uses blockchain to store permanently updates to DB.

When a dApp commits a transaction to the State DB, the latter fires a transaction with new state values to the Color Ledger. One of the nodes in the ledger's network takes this transaction to build a block. Other nodes validate the block and add them to the network.

Each new block from the network is used to update the state. Special care is taken to resolve conflicts, when a state variable is used in a running dApp transaction and at the same time it arrives in a new block. In this case the state is updated from the new block and the transaction is restarted.

More complicated case happens when Block Builder faces two committed transactions starting from the same state. In this case the Block Builder will reject one of them and inform the Logic Runner that produced the rejected transaction that it needs to re-execute the transaction again.

Color Platform equips dApps with **Storage** to keep files and big blobs of data. Operation with the storage will be synched with Block Builder transactions.

Bridging Cloud with Blockchain

To be completed soon

Minimal Viable Product

Minimal Viable Product of Color Spectrum will be released in November-December 2018. It will include a cloud infrastructure with a couple of dApps deployed and blockchain backend.

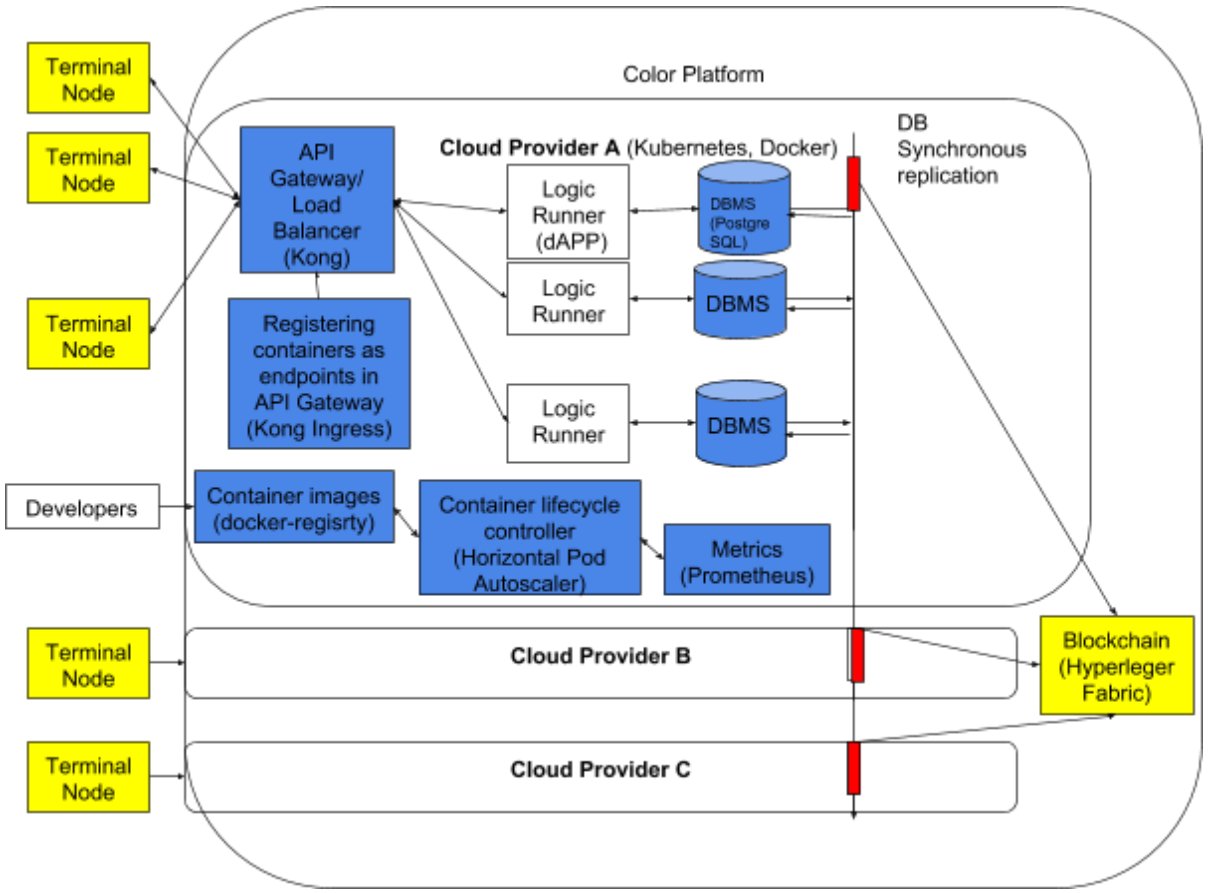

Figure 6 MVP Architecture

MVP is built using Kubernetes cloud infrastructure and Hyperleger Fabric. Application Protocol Interface and Load Balancer are based on Kong gateway. Registering containers as endpoints should be done automatically by using Kubernetes Ingress Controller for Kong. Container lifecycle controller monitors load and created new Docker containers if needed - it will be implemented using Horizontal Pod Autoscaler. System for getting metrics should be Prometheus as it can work both with Kong and Horizontal Pod Autoscaler.

For database we consider Foundation DB. This is an open source NoSQL database with ACID transaction support. For MVP we will develop a multi master synchronization engine that uses Hyperleger Fabric as synchronization protocol.

DApp development and deployment

To be completed soon