Building Blockchain Enterprise Solutions

Rahul Golash
Chief Blockchain Architect

13th June, 2018
Content

1. About ‘Aeries Blockchain Corporation’
2. Our Offerings
3. Technology Stack
4. BAeTH Blockchain Solution
5. How to take MVP to Production
   5.1. Upgradable Contracts
   5.2. Non Functional Requirement Considerations
   5.3. Deployment and Administration
6. BAeTH App Screenshots
7. Q&A
About Aeries Blockchain Corporation

Business Profile

Aeries Blockchain Corporation (ABC) is a Blockchain focused technology company. ABC’s senior leadership has held key positions in corporates like IBM, Oracle, HP, Broadridge, Siemens, & CA,

Headquartered in US with a global development centre in Bangalore, India.

Focused in providing Blockchain consulting and development services to ISVs, SaaS, Medium and large Enterprises.

Value Proposition

ABC has developed Blockchain based accelerator framework which enable us to quickly build secure and scalable solutions faster.

ABC empowers businesses to digitise your transaction workflow through a highly secured, shared and replicated ledger.

Experienced in delivering Smart Contract, Crypto token based financial derivatives, Supply Chain, eKYC & Digital Signature.
Our Offerings - Service Portfolio

What we do
Partner with customer towards their business goals:

• DApp Development, Testing and Support on private and public blockchain
• Decentralised exchange protocol development on Crypto currencies
• Ethereum and Hyperledger Fabric
• ICO crypto tokens
Our Offerings - Smart Solutions

What we do
Leverage blockchain capability to Improve:

- Supply Chain Traceability
- Transaction and Verification
- Process Efficiency
- Transparency
# Technologies Stack

<table>
<thead>
<tr>
<th>Distributed Ledgers</th>
<th>🍀 ETHEREUM</th>
<th>🍀 HYPERLEDGER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Languages</td>
<td>🍀 ANGULAR</td>
<td>🍀 Solidity</td>
</tr>
<tr>
<td>Frameworks</td>
<td>🍀 TRUFFLE</td>
<td>🍀 node js</td>
</tr>
<tr>
<td>CI/CD</td>
<td>🍀 remix</td>
<td>🍀 uport</td>
</tr>
<tr>
<td>NO SQL/Storage</td>
<td>🍀 docker</td>
<td>🍀 CHEF</td>
</tr>
<tr>
<td></td>
<td>🍀 Cassandra</td>
<td>🍀 MongoDB</td>
</tr>
</tbody>
</table>
BAeTH Blockchain Solution

Personal Instant Loan App on Blockchain
Executive Summary

- BAeTH is the masked project name of an Global fin-tech company which has instant personal mobile app for millennials.
- $50m total loans lent and repaid with 127,000 total no of loans
- Client wants to implement Distributed Ledger (Blockchain) enabled Digital Tokens using Smart Contracts on its lending platform
- Smart Contract based Distributed Ledger records all lending transactions in an open and transparent manner, thus allowing BAeTH and the borrower to execute a trusted lending transaction that is transparent and tamper proof.
- ABC is building and productionizing BAeTH Blockchain Solution.
BAeTH Blockchain Solution - Salient Points

- Open source Ethereum platform (latest release) with solidity, web3 as core-tech stack
- Private testnet and mainnet will be used for blockchain network
- Proof of Authority - PoA used for blockchain consensus mechanism
- Each user, merchant, bank, admin (/operators) will be given blockchain account
- ERC 20/223 standards to use for BAeTH tokens
- Mapping of loans and funds to actual token values
- Implement multi-sig wallet for collateral lock-in
- Blockchain Indexed log events to support User, Merchant, Bank wise filter
BAeTH Blockchain Solution Architecture
## BAeTH Blockchain Solution - Proposed Entities

<table>
<thead>
<tr>
<th>Microservice - Node.js, Front end + web3</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>• User login</td>
<td>• BAeTH Core nodes</td>
</tr>
<tr>
<td>• Banks</td>
<td>• Banks</td>
</tr>
<tr>
<td>• Regulators</td>
<td>• Merchants</td>
</tr>
<tr>
<td>• Credit Rating Agency</td>
<td>• Regulators</td>
</tr>
<tr>
<td>• Administrator’s operations &amp; reporting console</td>
<td>• Credit Rating Agency</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Roles</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>• User</td>
<td>• Token(s)</td>
</tr>
<tr>
<td>• Banks</td>
<td>• Loan</td>
</tr>
<tr>
<td>• Merchants</td>
<td>• User Profile &amp; eKYC</td>
</tr>
<tr>
<td>• Regulators</td>
<td>• User interactions</td>
</tr>
<tr>
<td>• Credit Rating Agency</td>
<td></td>
</tr>
</tbody>
</table>
# Used Technology Stack

<table>
<thead>
<tr>
<th>Distributed Ledgers</th>
<th>Ethereum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Languages</td>
<td>Angular, Solidity, TRUFFLE, Node.js, Python</td>
</tr>
<tr>
<td>Frameworks</td>
<td>Remix, uPort, MyEtherWallet, Metamask, Infura</td>
</tr>
<tr>
<td>CI/CD</td>
<td>Docker, Chef, Amazon Web Services, Jenkins</td>
</tr>
<tr>
<td>NO SQL/Storage</td>
<td>MongoDB, Couchbase</td>
</tr>
</tbody>
</table>
How to take MVP to Production
Core Considerations

- Right Consensus Mechanism e.g. choice between PoW, PoS, PoA
- Upgradable Contracts
- Follow Solidity/Ethereum Coding Best Practices
- Follow Engineering best practices e.g.
  - Code Repo & BugTracking Tool
  - CI/CD Pipeline
  - Docker/Swarm setup
  - Deployment in scalable and secured environment
- Non Functional Requirement e.g.
  - Security,
  - Scalability,
  - Performance and
  - Robustness
Why choose Proof of Authority - PoA?

• Suitable for all private blockchain

• Transactions and blocks are validated by approved accounts, known as validators or sealers

• Validators identity is approved ahead of time and hence allow only selected (authorised) nodes to join network

• No need to mining incentive

• Manage consensus with more than one authorized node

• Signer can sign at most one of a number of consecutive blocks (floor(SIGNER_COUNT / 2) + 1).

• The same consensus is applied when an authority node is removed from the network.

• Each banker will have one blockchain account
Upgradable Contracts
BAeTH Upgradable Smart Contracts
Upgradable Contracts - Best Practices

• Ethereum contracts are immutable so once deployed, can not be changed
• Contract Registry - Smart contract that assembles all other contracts we use
• Contract Manager - Smart contract enables us to not hardcode the address and look for registry before each function call
• Each contract will have a Storage-Implementation (Library) design to separate data from logic
• Managing data migration in chunk
• Index will maintain the latest version of each smart contract
• Use libraries to encapsulate logic
Solidity/Ethereum Coding Best Practices

Followed the best practices of security & solidity code from https://consensys.github.io/smart-contract-best-practices/

• **Race Conditions** - This can result into major bug and result into DAO’s collapse.
• **Reentrancy** - This can result into different invocations of the function to interact in destructive ways.
• **Cross-function Race Conditions** - This is similar to race conditions using two functions that share the same state.
• **Transaction-Ordering Dependence (TOD) / Front Running** - Can be avoided using batch transaction or pre-commit
• **Timestamp Dependence** - Business logic based on Timestamp should be carefully considered, since a node can change the local timestamp.
• **Integer Overflow and Underflow** - Smaller data-types like uint8, uint16, uint24...etc: can even more easily hit their maximum value, there are around 20 cases for overflow and underflow.
• **DoS with (Unexpected) revert and DoS with Block Gas Limit**
• **Token loss** due to contract misbehaviour
• **Availability loss**: external contracts e.g. regulators can not interact with the token contract due to its errors
Non Functional Requirement
Non Functional Requirements - (1/3)

Security

• All communications of mobile to BAeTH backend to BAeTH microservice using HTTPS (TLS 1.2) with a Level 3 SSL certificate

• The entire system is hosted within AWS cloud infrastructure with microservice API access only from whitelisted IPAddress and port control using EC2 security group.

• Application Seed and Customer Seed

Performance

• Asynchronous by design so as to allow maximum number of operations to take place including logging operations

• Using compiled libraries for encrypt/decrypt operations

• Using inbuilt libraries to perform tasks as opposed to using an external module

• Using HTTP 2.x (if required)

• Using Cluster module to make the Node.JS application use more than one core if available
Non Functional Requirements - (2/3)

Scalability

• The node application is deployed using Docker containers making the application horizontally scalable.

• Packages like PM2 also allow node applications to run on clusters while having an inbuilt load balancer to control number of instances.

• Using PoA as the consensus algorithm to increase block times.

• Increasing the block gas limit to facilitate more transactions per block
Non Functional Requirements - (3/3)

Highlighted Geth commands options which are used:

--datadir : Points to the data directory for storing ethereum data
--port : tells geth to use the port provided for inter node communication
--rpc : to enable the rpc communication with Web3.JS
--rpcaddr : allows to set the address on which the client will listen
--rpcport : The port on which the client rpc will run
--rpcapi 'personal,eth,web3,': restricts the exposure to web3 and eth
--networkid : custom network id
--gasprice '1' : limit the minimum gas price to decrease number of ether spent
--unlock : optional unlock of the coinbase account
--password : password for the coinbase account
--mine : start mining
--targetgaslimit 90000000: increases the number of transactions capable in a block
console '*' : enable the console interface to make admin changes
--nodiscover : disable peer discovery (adding peers manually the first time)
--rpccorsdomain : limit the usage of RPC to a particular ip/domain
--ws : enable the web socket interface to receive events faster
--wsorigins : set the web socket domain to control access

Additional - Clique block ‘period’ - 1 sec and ‘epoch’ being the default value
Deployment and Administration
CI/CD Pipeline

- Bitbucket tools for code repo

Static Analysis:
- **Mythril** - Reversing and bug hunting framework for the Ethereum blockchain
- **Oyente** - Analyze Ethereum code to find common vulnerabilities, based on this paper.

Test Coverage
- **Solidity-coverage** - Code coverage for Solidity testing.

Linters
Linters improve code quality by enforcing rules for style and composition, making code easier to read and review.
- **Solint** - Solidity linting that helps you enforce consistent conventions and avoid errors in your Solidity smart-contracts.
- **Solium** - Yet another Solidity linting.
- **Solhint** - A linter for Solidity that provides both Security and Style Guide validations.
Deployment Staging Environment
BAeTH AWS Deployment

- One Elastic Load Balancer
- Two instances of micro-services under autoscaling group
- Credit Rating Node on different VPC
- Regulator and Auditor Nodes on the same private subnet
BAeTH App Screenshots
Apply Loans and Award BAeTH Tokens

User has 0 tokens and no loan

User asks loan of INR 5000

3450 BAeTH tokens in Wallet
Apply Loans and Award BAeTH Tokens

<table>
<thead>
<tr>
<th>Request Details</th>
<th>Response Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://13.177.199.8:8000/customers/balance">http://13.177.199.8:8000/customers/balance</a></td>
<td>Balance: 0 tokens</td>
</tr>
<tr>
<td>Application Seed</td>
<td></td>
</tr>
<tr>
<td>Booth</td>
<td></td>
</tr>
<tr>
<td>Customer Seed</td>
<td></td>
</tr>
<tr>
<td>5571968</td>
<td></td>
</tr>
</tbody>
</table>

Stage 0 - User has 0 tokens in the wallet
Apply Loans and Award BAeTH Tokens

Transaction shown in EthExplorer
Apply Loans and Award BAeTH Tokens

Stage 1 - User has 3450(00) tokens in the wallet
Transfer BAeTH Tokens to a friend

User screen

User to transfers 200 BAeTH

User enters the details of recipient
Stage 0 - Recipient has 377389(00) tokens in the wallet
Transfer BAeTH Tokens to a friend

Transaction shown in EthExplorer
Transfer BAeTH Tokens to a friend

Stage 1 - Recipient has 377589(00) tokens in the wallet
Thank You

Q & A

Contact :

Rahul Golash
rahul@aeries.io
+61 435 228670