Application Development using Sawtooth[Hands-On]

Santosh & Reginald

https://github.com/santhoshzcv/sawtooth

Launch Sawtooth BareMetal? or Docker?

Launch Sawtooth on local machine requires atleast 6 separate terminals & execute commands.

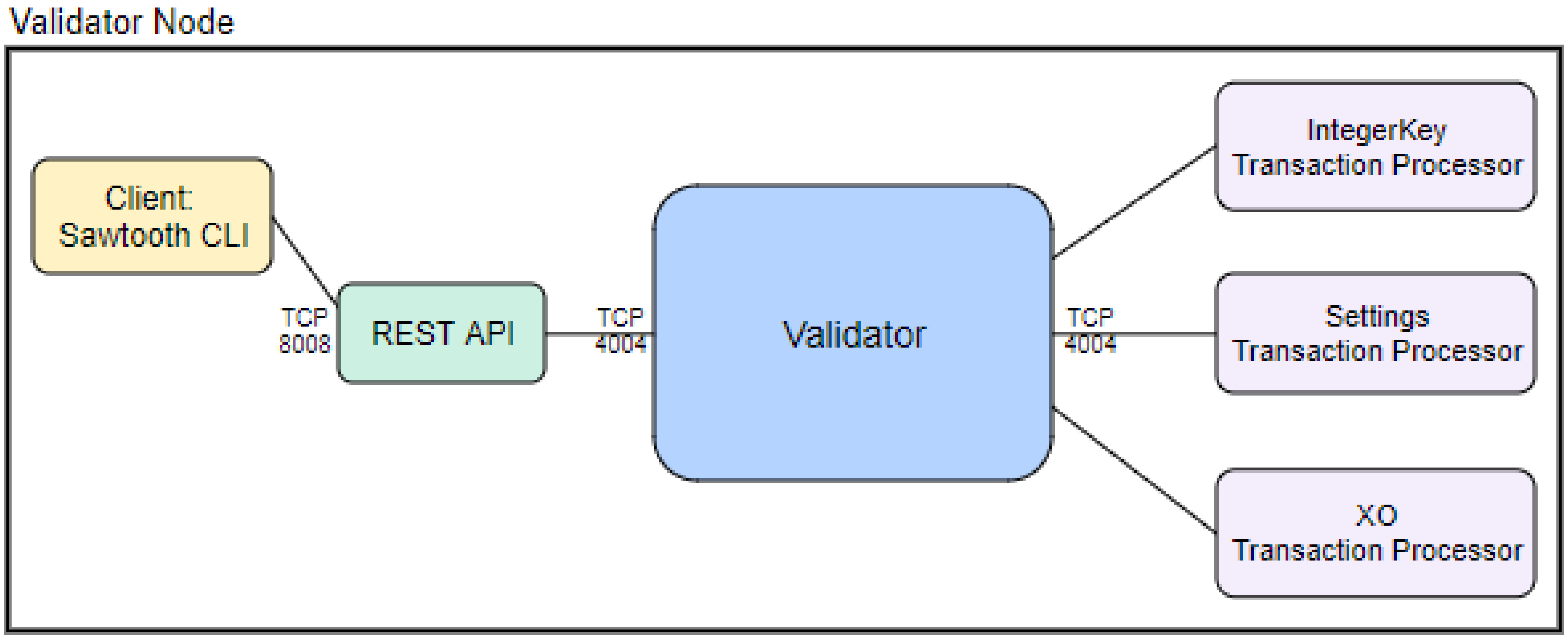
- **user@validator$** sawtooth keygen
- **user@validator$** sawset genesis
- **user@validator$** sudo -u sawtooth sawadm genesis config-genesis.batch
- **user@validator$** sudo sawadm keygen
- **user@validator$** sudo -u sawtooth sawtooth-validator -vv
- **user@consensus$** sudo -u sawtooth devmode-engine-rust -vv --connect tcp://localhost:5050
- **user@rest-api$** sudo -u sawtooth sawtooth-rest-api -v
- **user@settings$** sudo -u sawtooth settings-tp -v
- **user@client$** sawtooth settings list
- **user@intkey$** sudo -u sawtooth intkey-tp-python -v
- **user@xo$** sudo -u sawtooth xo-tp-python -v

Launch Sawtooth BareMetal? or Docker?

```
user@client$ ps aux | grep [s]awtooth-rest-api

user@client$ intkey create_batch --count 10 --key-count 5
user@client$ intkey load -f batches.intkey
user@client$ sudo bash -c "tail -10 /var/log/sawtooth/intkey-*-debug.log"

user@client$ intkey create_batch --count 10 --key-count 5
user@client$ sawtooth batch submit -f batches.intkey

user@client$ sawtooth block list
user@client$ sawtooth block show {BLOCK_ID}
user@client$ sawtooth state list
user@client$ sawtooth state show {STATE_ADDRESS}

user@client$ sudo ls -1 /var/log/sawtooth
```

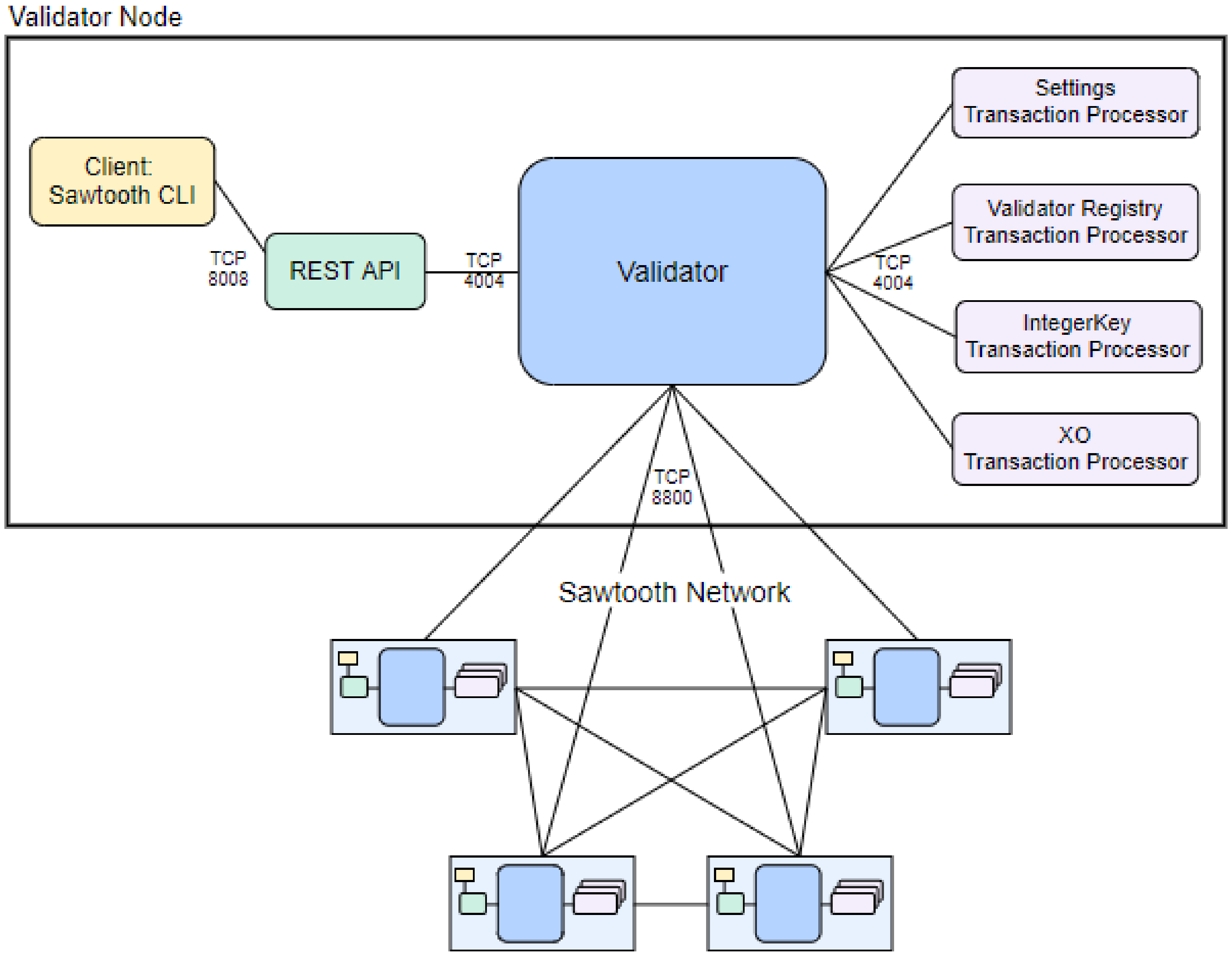
Sawtooth Setup walkthrough

- Creating the genesis block
- Generate a root key – associated with validator node
- Start components(in order) starting with
  - validator, consensus engine, REST API, and transaction processors
- Check status of REST API
- Using Sawtooth commands to submit transactions, display block data, and view global state
- Examine Sawtooth logs
- Stop Sawtooth and resetting the development environment


Validator Node
Client:
Sawtooth CLI
TCP
8008
REST API
TCP
4004
Validator
TCP
4004
IntegerKey
Transaction Processor
Settings
Transaction Processor
XO
Transaction Processor

Docker commands

- **docker-compose -f sawtooth-default.yaml up**
- To log into client container
  - run command “**docker exec -it sawtooth-shell-default bash**”.
  - Once inside, you can access the sawtooth cli commands.
- Confirm that a validator runs and reachable from inside
  - docker container with “**curl http://rest-api:8008/blocks**” command
  - host machine by “**curl http://localhost:8008/blocks**” command


Validator Node
Settings Transaction Processor
Client: Sawtooth CLI
Validator Registry Transaction Processor
TCP 8008
REST API
TCP 4004
Validator
TCP 4004
IntegerKey Transaction Processor
XO Transaction Processor
TCP 8800
Sawtooth Network

Addressing Namespace Restriction

- Namespace is not necessarily a 1-to-1 relationship between namespaces and transaction families
- Ensure transaction families cannot write data at addresses that is only intended to be read. This is the goal of *namespace restriction* feature, when explicitly activated
  - By default and for better flexibility, this namespace restriction is not enforced by the validators.
  - To activate some namespace restrictions, appropriate settings must be published on-chain using the Settings Transaction Family
  - If family sawtooth settings does not indicate any namespace, the validator will let it write at any address
  - Validators verify that transaction processors only perform *set* operations where addresses have a prefix in common with one of the family's specified namespace prefix(es).

Sawtooth Application-Project Structure

- {application_name}
    - xxx_tp
        - index.js   - registers transaction handler with validator
        - handler.js   - implement business logic
        - state.js   – implement state get/set methods
    - xxx_tp_client
        - index.js
        - Key_manager.js   - create/save/retrieve public and private keys for end-user
        - prepare_transaction.js - create transaction payloads and batch payloads
        - submit_transaction.js
        - event_subscription.js
    - shared
        - Addressing.js   - Generate address based on namespace, PREFIX   and inputs
        - env.js   - contains settings related to environment variables, network variables,   namespace.

Sawtooth application 1 Walkthrough

Demonstrates simple Sawtooth application

- Transaction Processor (TP) and client written in NodeJS
    - Supported Operation using normal payload
- Transaction Processor
    - Execute Action based on normal payload
- Client
    - Generate Keys
    - Create Transaction
        - Wraps transaction in a batch
        - Submits to Validator via REST API

ProtoBuf

- https://developers.google.com/protocol-buffers/
- Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data.
    - think XML, but smaller, faster, and simpler.
- You define data to be structured once
    - Then, you can use special generated source code to easily write and read your structured data to and from a variety of data streams and using a variety of languages.
- Each protocol buffer message is a small logical record of information, containing a series of name-value pairs

Sawtooth application 2 Walkthrough

Demonstrates simple Sawtooth application

- Transaction Processor (TP) and client written in NodeJs
    - Supported Operation using **ProtoBuf** payload
- Transaction Processor
    - Execute Action based on **Protobuf** payload
- Client
    - Generate Keys
    - Create Transaction
        - Wraps transaction in a batch
        - Submits to Validator via REST API
    - **Subscribe to Events**

Subscribe to Sawtooth events

Hyperledger Sawtooth supports creating and broadcasting events

Event subscription enables an application to perform the following:

- Subscribe to events that occur related to the blockchain
- Notify of transaction execution back to clients without storing that data in state
- Perform event catch-up to gather information about state changes from a specific point on the blockchain

An application can

  - React immediately to each event or
  - Store event data for later processing and analysis

Core Sawtooth events

Core Sawtooth events are prefixed with sawtooth. The core events are:

- **sawtooth/block-commit**: event occurs when a block is committed. This event contains information about the block, such as the block ID, block number, state root hash, and previous block ID
- sawtooth/state-delta : event occurs when a block is committed and contains all state changes that occurred at a given address for that block.