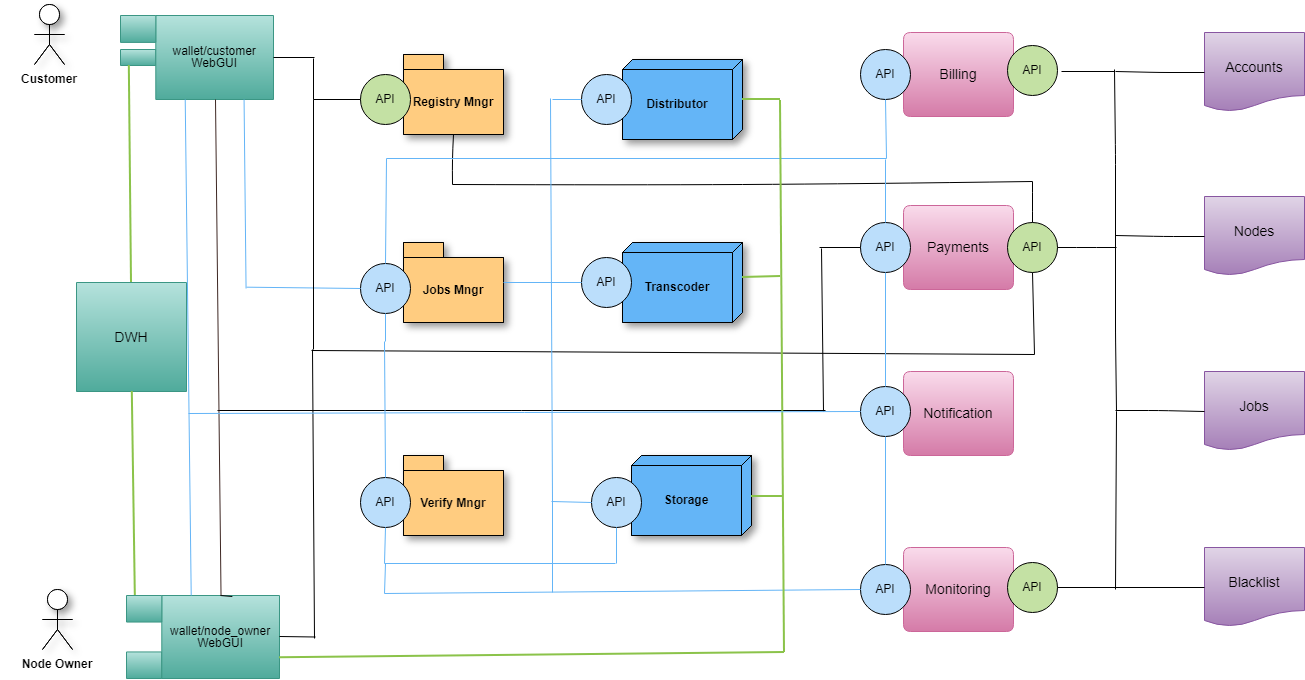
**Microservices decomposition and API commands**

**(first draft)**



**Microservices:**

* Customer GUI
* DWH
* Node owner GUI
* Registry manager
* Jobs manager
* Verify manager
* transcoding service
* Distributor service
* Storage service
* Billing
* Payments
* Notification
* Monitoring

**API’s**:

* Blue circle blue line (AMQP transport, async usage, RabbitMQ)
* Green circle black line (http json-rpc, sync usage, geth and web3py)
* Green line - software installation

**Smart Contracts (Ethereum POA)**

* Accounts
* Nodes
* Jobs
* Blacklist

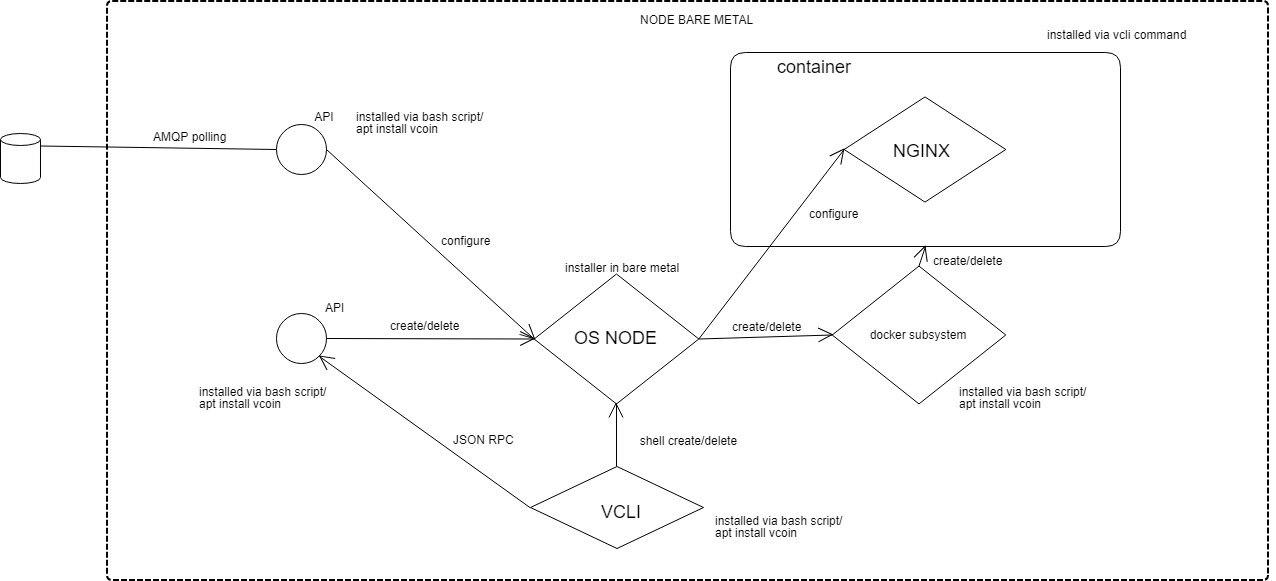
**vcli - API commands guide**

* **help** # Help about any command
* **version** # Show version
* **register** # Manage registry
  + **customer** # register customer account
    - remove customer
  + **supplier** # register node owner account
    - add node # add node address to SC
    - remove node # remove node address to SC
    - remove supplier # remove node owner address from SC
    - list # list of the current nodes for supplier
* **login** # login customer with address
* **payment** # payment control
  + balance # get current balance
  + deposit # deposit tokens to SC
  + withdraw # withdraw tokens from SC
  + transfer # transfer tokens to any address
* **order** # Job order from customer
  + create # create order for job, using GUI or file.json (result is job id and pipeline.json)
  + cancel # cancel job with job\_id
  + list # list of current jobs
  + status # status job with job\_id
* **pipeline** # pipeline management by customer
  + list # list of current pipelines
  + status # pipeline with job\_id
  + start # start pipeline from GUI or using pipeline.json
  + stop # stop pipeline with job\_id
  + monitor # add pipeline to monitor
  + close # close and destroy pipeline
* **node** # node management by node owner
  + create
    - transcoder # install transcoder docker image from DWH
    - storage # install storage docker image from DWH
    - distributor # install distributor docker image from DWH
  + list # list of current nodes
  + remove # remove specific node
  + purge # remove all nodes
  + hardware # ask for computer resources in use
  + free\_hardware # ask for free computer resources
  + status # status for specific node
  + maintenance # schedule node maintenance
  + pipeline # list of current pipelines for the node

**Using the docker of the nginx-rtmp container to install and configure the node (distributor, transcoder, storage)**

1. Creating json format for configuring nodes with docker data
2. Changing json format for reconfiguring nodes and dividing formats into public / private
3. Preparing a docker image to work with this file (project in github + image in docker repo)
4. Modification of the read.py receiver to work with the docker container
5. Modification sender send.py for public configuration commands
6. real configuration tests

**The general scheme of working with the docker container**



**Json format unification**

It is proposed to divide the json configuration file for nodes into 2 parts:

The api private part contains the data associated with the installation of the docker container node

The api public part is related to the direct configuration of the node for performing a specific task (node role). The process itself is opaque to the user and serves solely to simplify the understanding and configuration of the node file.

**node create**

**api public:**

{

"\_comments": "version 1.0 API. Top level describe ref: https://docs.google.com/document/d/1fFYMT2GRLFBQrtF0fltHrq\_-3tciqQ6-fT\_43CoPBS0, former t5\_task\_gen.json",

"apiVersion": "1.00",

"typeApi": "public",

"command": "node",

"subcommand": "create",

"params": "distributor",

"node\_id":"88aee28aa0a28177d2c676bfafda48d03ff44eb8"

}

**node create**

**api private:**

{

"\_comments": "version 1.0 API. Top level describe ref: https://docs.google.com/document/d/1fFYMT2GRLFBQrtF0fltHrq\_-3tciqQ6-fT\_43CoPBS0, former t5\_task\_gen.json",

"apiVersion": "1.00",

"typeApi": "private",

"command": "node",

"subcommand": "create",

"params": "distributor",

"node\_id": "88aee28aa0a28177d2c676bfafda48d03ff44eb8",

"spec": {

"containers":

{

"image": "77phnet/nginx-rtmp",

"ports": {"1935":"1935","8080":"80"},

"volume-mounts": {

"/opt/nginx/nginx.conf": "/opt/nginx/nginx.conf",

"/opt/nginx/app-enabled": "/opt/nginx/app-enabled",

"/opt/nginx/sites-enabled": "/opt/nginx/sites-enabled",

"/opt/data":"/opt/data"

}

}

}

A similar scheme is used for pipeline configuration files:

**pipeline start**

**api public:**

{

"\_comments": "version 1.0 API. Top level describe ref: https://docs.google.com/document/d/1fFYMT2GRLFBQrtF0fltHrq\_-3tciqQ6-fT\_43CoPBS0, former t5\_task\_gen.json",

"apiVersion": "1.00",

"typeApi": "public",

"command": "pipeline",

"subcommand": "start",

"pipeline":

{

"type\_source": "file",

"job\_id": "0200000000000000000001",

"number\_of\_streams": "3",

"source": {

"resolution\_name": "\_4k",

"resolution": "3840x2160",

"bitrate": "50000000"

},

"transcoding": [

{

"resolution\_name": "\_1080p",

"resolution": "1920x1080",

"bitrate": "6000000"

},

{

"resolution\_name": "\_720p",

"resolution": "1280x720",

"bitrate": "3000000"

},

{

"resolution\_name": "\_480p",

"resolution": "854x480",

"bitrate": "1000000"

}

],

"nodes": [

{

"type": "distributor",

"node\_id": "1d94baec6903bd722953d1111d3b03ea3fa99378",

"ip": "35.198.30.225",

"failover\_ip": "35.221.152.9"

},

{

"type": "storage",

"node\_id": "2f45731160c02f69cad1ff8ab9a48492dc3b2022",

"ip": "35.242.172.58",

"failover\_ip": "35.242.172.59"

},

{

"type": "transcoder",

"node\_id": "88aee28aa0a28177d2c676bfafda48d03ff44eb8",

"ip": "35.233.205.84",

"failover\_ip": "35.233.205.85"

},

{

"type": "transcoder",

"node\_id": "d8a9d07b9884d32da3700634aab1d373fffd36ef",

"ip": "35.197.27.59",

"failover\_ip": "35.197.27.60"

},

{

"type": "transcoder",

"node\_id": "4586f87cdeb706f037d4cddcac051c04b2f55d16",

"ip": "35.197.4.197",

"failover\_ip": "35.197.4.198"

}

]

}

}

**pipeline start**

**api private:**

{

"\_comments": "version 1.0 API. Top level describe ref: https://docs.google.com/document/d/1fFYMT2GRLFBQrtF0fltHrq\_-3tciqQ6-fT\_43CoPBS0, former t5\_task\_gen.json",

"apiVersion": "1.00",

"typeApi": "private",

"command": "pipeline",

"subcommand": "start",

"pipeline": {

"tasks": [{

"node\_id": "1d94baec6903bd722953d1111d3b03ea3fa99378",

"msg": {

"command": "config",

"type": "distributor",

"nginx-rtmp-core": {

"listen": "1935",

"application": "0200000000000000000001"

},

"nginx-rtmp-live": {

"live": "on"

},

"ffmpeg-rtmp": {

"outputs": [{

"variant": "\_4k",

"ip": "35.242.172.58",

"failover\_ip": "35.242.172.59"

},

{

"ip": "35.233.205.84",

"failover\_ip": "35.233.205.85"

},

{

"ip": "35.197.27.59",

"failover\_ip": "35.197.27.60"

},

{

"ip": "35.197.4.197",

"failover\_ip": "35.197.4.198"

}

]

}

}

},

{

"node\_id": "2f45731160c02f69cad1ff8ab9a48492dc3b2022",

"msg": {

"command": "config",

"type": "storage",

"nginx-rtmp-core": {

"listen": "1935",

"application": "0200000000000000000001"

},

"nginx-rtmp-live": {

"live": "on"

},

"nginx-rtmp-hls": {

"hls\_fragment": "5",

"hls\_playlist\_length": "30",

"hls\_fragment\_naming": "timestamp",

"hls\_fragment\_naming\_granularity": "2",

"hls\_fragment\_slicing": "aligned",

"hls\_variant": [{

"variant": "\_4k",

"params": "BANDWIDTH=50000000,RESOLUTION=3840x2160"

}, {

"variant": "\_1080p",

"params": "BANDWIDTH=6000000,RESOLUTION=1920x1080"

}, {

"variant": "\_720p",

"params": "BANDWIDTH=3000000,RESOLUTION=1280x720"

}, {

"variant": "480",

"params": "BANDWIDTH=1000000,RESOLUTION=854x480"

}]

}

}

},

{

"node\_id": "88aee28aa0a28177d2c676bfafda48d03ff44eb8",

"msg": {

"command": "config",

"type": "transcoder",

"nginx-rtmp-core": {

"listen": "1935",

"application": "0200000000000000000001"

},

"nginx-rtmp-live": {

"live": "on"

},

"ffmpeg-rtmp": {

"output": {

"application": "0200000000000000000001",

"variant": "\_1080p",

"codec": "libx264",

"bitrate": "6000000",

"resolution": "1920x1080",

"ip": "35.233.205.84",

"failover\_ip": "35.233.205.85"

}

}

}

},

{

"node\_id": "d8a9d07b9884d32da3700634aab1d373fffd36ef",

"msg": {

"command": "config",

"type": "transcoder",

"nginx-rtmp-core": {

"listen": "1935",

"application": "0200000000000000000001"

},

"nginx-rtmp-live": {

"live": "on"

},

"ffmpeg-rtmp": {

"output": {

"application": "0200000000000000000001",

"variant": "\_720p",

"codec": "libx264",

"bitrate": "3000000",

"resolution": "1280x720",

"ip": "35.197.27.59",

"failover\_ip": "35.197.27.60"

}

}

}

},

{

"node\_id": "4586f87cdeb706f037d4cddcac051c04b2f55d16",

"msg": {

"command": "config",

"type": "transcoder",

"nginx-rtmp-core": {

"listen": "1935",

"application": "0200000000000000000001"

},

"nginx-rtmp-live": {

"live": "on"

},

"ffmpeg-rtmp": {

"output": {

"application": "0200000000000000000001",

"variant": "\_480p",

"codec": "libx264",

"bitrate": "1000000",

"resolution": "854x480",

"ip": "35.197.4.197",

"failover\_ip": "35.197.4.198"

}

}

}

}

]

}

}

**Nginx-rtmp docker image**

To work with unified json, a nginx-rtmp fork was created with an empty default configuration for nginx:

<https://github.com/77ph/docker-nginx-rtmp>

repo:

<https://hub.docker.com/r/77phnet/nginx-rtmp/>

**changes for read.py**

* Reading unified json files
* Adding an option to work with a new docker container

**changes for send.py**

* Sending unified json files

**Test send.py/read.py with real configuration**

The full synthetic test performs [**modules-test.sh**](https://drive.google.com/open?id=1OuMKYg0zMqLvy9UFMXr5NnRFdw8ReCJF)

# 1 TEST - creating a distributor node

# 2 TEST - distributor node configuration for application job\_id

# 3 TEST - create transcoder node

# 4 TEST - configuration transcoder node for application job\_id

# 5 TEST - creating storage node

# 6 TEST - storage node configuration for application job\_id

Example:

TEST #1. Create distributor node. node\_id 1d94baec6903bd722953d1111d3b03ea3fa99378

Send task to query 1d94baec6903bd722953d1111d3b03ea3fa99378. Step pass

Waiting 15s

Task received and container created. Step pass

607375c4ba3a 77phnet/nginx-rtmp "/opt/nginx/sbin/ngi…" 13 seconds ago Up 11 seconds 0.0.0.0:1935->1935/tcp, 0.0.0.0:8080->80/tcp distributor

The files match test/nginx.conf and /opt/nginx/nginx.conf. Step pass

TEST #2. Pipeline to distributor node. jobid 0200000000000000000001. node\_id 1d94baec6903bd722953d1111d3b03ea3fa99378

Send pipline to query 1d94baec6903bd722953d1111d3b03ea3fa99378. Step pass

Waiting 15s

Task received and app created. Step pass

The files match test/distributor-app.conf and /opt/nginx/app-enabled/0200000000000000000001. Step pass

rollback to zero state

TEST #3. Create transcoder node. node\_id 88aee28aa0a28177d2c676bfafda48d03ff44eb8

Send task to query 88aee28aa0a28177d2c676bfafda48d03ff44eb8. Step pass

Waiting 15s

Task received and container created. Step pass

d5dd2dbfc9da 77phnet/nginx-rtmp "/opt/nginx/sbin/ngi…" 12 seconds ago Up 11 seconds 0.0.0.0:1935->1935/tcp, 0.0.0.0:8080->80/tcp transcoder

The files match test/nginx.conf and /opt/nginx/nginx.conf. Step pass

TEST #4. Pipeline to transcoder node. jobid 0200000000000000000001. node\_id 88aee28aa0a28177d2c676bfafda48d03ff44eb8

Send pipline to query 88aee28aa0a28177d2c676bfafda48d03ff44eb8. Step pass

Waiting 15s

Task received and app created. Step pass

The files match test/transcoder-app.conf and /opt/nginx/app-enabled/0200000000000000000001. Step pass

rollback to zero state

TEST #5. Create storage node. node\_id 2f45731160c02f69cad1ff8ab9a48492dc3b2022

Send task to query 2f45731160c02f69cad1ff8ab9a48492dc3b2022. Step pass

Waiting 15s

Task received and container created. Step pass

3052d7bfa736 77phnet/nginx-rtmp "/opt/nginx/sbin/ngi…" 12 seconds ago Up 11 seconds 0.0.0.0:1935->1935/tcp, 0.0.0.0:8080->80/tcp storage

The files match test/nginx.conf and /opt/nginx/nginx.conf. Step pass

TEST #6. Pipeline to storage node. jobid 0200000000000000000001. node\_id 2f45731160c02f69cad1ff8ab9a48492dc3b2022

Send pipeline to query 2f45731160c02f69cad1ff8ab9a48492dc3b2022. Step pass

Waiting 15s

Task received and app created. Step pass

The files match test/storage-app.conf and /opt/nginx/app-enabled/0200000000000000000001. Step pass

**Issues and future steps**

Closed issue:

1. Creating a nginx-rtmp as docker with python (like k8s) & config in volumes (t5\_read.py/t5\_send.py)

https://docs.docker.com/samples/library/nginx/#complex-configuration

https://stackoverflow.com/questions/31763172/mounting-nginx-conf-as-a-docker-volume-causes-system-error-boot2docker

https://www.shellhacks.com/ru/docker-reload-nginx-inside-container/

https://stackoverflow.com/questions/31697828/docker-run-name-is-already-in-use-by-container

idea for container:

docker run --name my-nginx -p 1935:1935 -p 8080:80 --rm nginx-rtmp -v /opt/nginx/nginx.conf:/opt/nginx/nginx.conf:ro -v /opt/nginx/app-enabled/:/opt/nginx/app-enabled/:ro -v /opt/data:/opt/data:rw -d nginx-rtmp

status: fixed

example:

t5\_send.py api/pipleline-with-commands.json equal vcli create pipeline pipeline.json

t5\_send.py api/node\_create-public.json

2. Unification API format (pipeline, node create)

https://docs.google.com/document/d/1Eca\_oKJ1Q4rKYD7GPkXBQdLkLRvdHEFk7LtvNABIIJY/edit

result: reference json files in api/ folder

status API: draft 1.0

3. Using new format API in t5\_read.py/t5\_send.py (ref: #2)

status: fixed

4. merge t6\_read.py (test with nginx/docker) & t5\_read.py

status: fixed, t6\* removed

5. "Write module level test scripts to test enhanced t5\_read.py"

status: fixed

result: modules\_test.sh

------------

Open issue in t5\_read/t5\_send:

1. 1 container per vps/bare metal server.

Source issue: For multi-nginx-container per server need solve:

1.1. Mapping volumes like

/opt/uuid/nginx/nginx.conf:/opt/nginx/nginx.conf

/opt/uuid/nginx/app-enabled/:/opt/nginx/app-enabled/

/opt/uuid/nginx/sites-enabled:/opt/nginx/sites-enabled/

/opt/uuid/data:/opt/data

vs

/opt/nginx/nginx.conf:/opt/nginx/nginx.conf

opportunity to fix: yes, complicated: medium

1.2. Renamed container to role-uuid vs role. Where role = ["distributor","transcoder","storage"]

opportunity to fix: yes, complicated: medium

1.3. Be responsible for correct use TCP port in mapping. Example: now 1935:1935 eq 1935 port of host => 1935 port of docker.

The only correct solution: consider free ports as server resources and assign the correct ports in pipeline.

Counting ports lies outside of send/read API.

Possible: test with different tcp port in task (1936,1937 ..). complicated: medium

2. Sending a result/stats from t5\_read.py back via rabbitmq as part of future architectures/API. It goes beyond the task.

**All scripts are here**

<https://drive.google.com/drive/folders/1ELnVJoOjt7-pQBEqGGNixdTcls68X5WJ?usp=sharing>