

---

**mitum**  
*Release v2*

**Aug 25, 2022**



---

## Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Deploy</b>	<b>3</b>
<b>3</b>	<b>Command Usage</b>	<b>9</b>



# CHAPTER 1

---

## Introduction

---

mitum is a general purpose blockchain with flexible and resilient way. mitum can be used for various kind of purposes, public and private blockchain like cryptocurrency network, data-centric blockchain for arbitrary data, or secure anonymity voting system, etc.



## 2.1 Build

### 2.1.1 Requirement

- POSIX-compliant environment
- Install latest go lang

---

**Note:** We have tested in Linux(amd64 and arm64) and the latest Darwin machine. If you are in other platform, you can easily find the way to build.

---

### 2.1.2 Get source from github

Official mitum github repository is <https://github.com/spikeekips/mitum2> .

---

**Note:** Originally official mitum was <https://github.com/spikeekips/mitum2>, but at this time, we are preparing *mitum2*, previous *mitum* will be obsolete for restricted usage.

---

```
$ git clone https://github.com/spikeekips/mitum2
$ cd mitum2/example
$ go build -o ./mitum-example *.go
```

You can build without warnings or errors.

### 2.1.3 Get from release

You can find the latest release from <https://github.com/spikeekips/mitum2/releases> .

---

**Note:** At this time, there are no releases. The first release will come.

---

## 2.2 Config

This is example node configuration file.

### 2.2.1 *first-node.yml*

Listing 1: first-node.yml

```
1 address: no0sas
2 privatekey: EYc4WdFjP9qkgfwJZfnsVXeh827rsNppm5HUSjSDeMFFmpr
3 network_id: mitum; Sat 26 Dec 2020 05:29:13 AM KST
4 network:
5   bind: 0.0.0.0:4320
6   tls_insecure: true
7 storage:
8   base: /mitum-data
9   database: leveldb://
10 sync_sources:
11   - type: sync-source-node
12     address: no1sas
13     publickey: 25AZEiKTPPhNkpcj6B1mofXHFyJRR8DaEMcNjc2WSvvW8Jmpu
14     publish: localhost:4321
15     tls_insecure: true
```

- address:

Local node mitum address. You can make new address by these rules;

- Must ends with *hint type*, *sas*
- Length should be  $6 \leq \text{length} \leq 100$  including hint type, *sas*
- Empty characters(space, tab, etc) should not be included
- Regexp,  $^[a-zA-Z0-9][\w\-\.\!\$\*\@]*[a-zA-Z0-9]\$$  should be passed

- privatekey:

*privatekey* for local node. It will be used to sign the messages from local node.

New *privatekey* and it's *publickey* can be easily made:

```
$ ./mitum-example key new
{
  "privatekey": "685WQHnt51eaETuQ1WUYEVEEdMsBS5XD5SCaU6NuiqHV4mpr",
  "publickey": "gGvk6uzEDWbu7DXTNuiQGRGFUThEbst3EHL79YF3cCKkmpu",
  "hint": "mpr-v0.0.1",
  "seed": "",
  "type": "privatekey"
}
```

For more details about `key new`, see “*Command Usage*”



- `network_id`:

*network id* indicates which mitum network local node are in. *network id* should not be empty and not be greater than 300 bytes.

- `network.bind`:

bind address. *port* should be set. For example,

- `0.0.0.0:4320`: Listen from anywhere
- `:4320`: same with `0.0.0.0:4320`
- `127.0.0.1:4320`: Listen only from localhost

- `network.tls_insecure`:

If local node uses self-signed TLS certificate, it should be `true`.

- `storage.base`:

*base* indicates the directory for produced data. If not set, current directory will be used.

- `storage.database`:

*database* indicates the database uri. By default, `leveldb://`. If not set, default database will be used, `leveldb://<storage.base>/db`.

- `sync_sources`:

*Trust nodes*. You can set multiple *trust nodes*.

- `sync_sources.type`:

*type* indicates how to fetch the *source\_source*.

- `sync-source-node`: `node`
- `sync-source-suffrage-nodes`: get it's suffrage nodes from other node
- `sync-source-sync-sources`: get it's sync source nodes from other node
- `sync-source-url`: get sync source nodes thru url

- **`sync_sources.type=sync-source-node`:**

```

1 - type: sync-source-node
2   address: nolsas
3   publickey: 25AZEiKTPHNkpcj6B1mofXHFyJRR8DaEMcNjc2WSvvW8Jmpu
4   publish: localhost:4321
5   tls_insecure: true

```

- **`sync_sources.type=sync-suffrage-nodes`:**

```

1 - type: sync-source-suffrage-nodes
2   publish: 1.2.3.4:4321
3   tls_insecure: true

```

- **`sync_sources.type=sync-source-sync-sources`:**

```

1 - type: sync-source-sync-nodes
2   publish: 1.2.3.4:4321
3   tls_insecure: true

```

- **`sync_sources.type=sync-source-url`**: Set url; remote url should return the json list of *NodeConnInfo*.

```

1 - https://a.b.c.d/sync-sources.json

```

```
[
  {
    "conn_info": "1.2.3.4:4321#tls_insecure",
    "address": "node0sas",
    "publickey": "yjXAD7m9knauk7qgo44S2BVpKRc8QXcgDQrAqwwxKtbhmpu",
    "_hint": "node-conninfo-v0.0.1"
  },
  {
    "conn_info": "4.3.2.1:1234#tls_insecure",
    "address": "node1sas",
    "publickey": "tZSrHbkNtDzHvjfbKocgaPwNjx99buMBFdhKTUPxXCafmpu",
    "_hint": "node-conninfo-v0.0.1"
  }
]
```

## 2.3 Run node: *init*

To deploy, mitum needs to be initialized its database and storage fs. With node config, *init* needs another config, which defines operations for genesis block, initial network policy and other configurations.

### 2.3.1 Genesis Config

Listing 2: genesis-design.yml

```
1 facts:
2   - _hint: suffrage-genesis-join-fact-v0.0.1
3     nodes:
4       - _hint: node-v0.0.1
5         address: no0sas
6         publickey: bXTT1hoetSKYPUmfu3bMRcs8aU342MTTzhgeCQ1bTavBmpu
7
8   - _hint: genesis-network-policy-fact-v0.0.1
9     policy:
10      _hint: network-policy-v0.0.1
11      max_operations_in_proposal: 99
12      suffrage_candidate_lifespan: 333333333
13      suffrage_candidate_limiter:
14        _hint: fixed-suffrage-candidate-limiter-rule-v0.0.1
15        limit: 1
16      max_suffrage_size: 3
```

- facts lists the facts of genesis operations. It's shape is almost identical of json output of operation fact.

#### Facts

- `_hint: suffrage-genesis-join-fact-v0.0.1`  
Defines the initial suffrage nodes
- `_hint: genesis-network-policy-fact-v0.0.1`  
Defines the initial network policy

### 2.3.2 `init`

```
$ ./mitum-example init genesis-degign.yml --design=first-node.yml
```

This will initialize storage and generates genesis block.

## 2.4 Run node: *run*

After `init` is done, you can run mitum node.

```
$ ./mitum-example run --design=first-node.yml
```

Done.



---

## Command Usage

---

```
$ ./mitum-example --help
Usage: mitum-example <command>

Flags:
  -h, --help    Show context-sensitive help.

logging
  --log.out="stderr"      log output file: {stdout, stderr, <file>}
  --log.format="terminal" log format: {json, terminal}
  --log.level=debug      log level: {trace, debug, info, warn, error}
  --[no-]log.force-color log force color

Commands:
  import <node design> <from directory>
    import from block data

  init <node design> <genesis design>
    init node

  run <node design>
    run node

  network client <header> [<remote>]
    network client

  key new [<seed>]
    generate new key

  key load <key string>
    load key

  key sign <privatekey> <network-id> <body>
    sign
```

(continues on next page)

(continued from previous page)

```
Run "mitum-example <command> --help" for more information on a command.
```

### 3.1 Command: *init*

```
$ ./mitum-example init --help
Usage: a init <node design> <genesis design>

init node

Arguments:
  <genesis design>    genesis design

Flags:
  -h, --help        Show context-sensitive help.

logging
  --log.out="stderr"      log output file: {stdout, stderr, <file>}
  --log.format="terminal" log format: {json, terminal}
  --log.level=debug      log level: {trace, debug, info, warn, error}
  --[no-]log.force-color log force color

design
  --design=./config.yml    design uri; 'file:///config.yml', 'https://a.b.
↪c.d/config.yml'
  --[no-]design.https.tls_insecure  https tls insecure
```

### 3.2 Command: *run*

```
$ ./mitum-example run --help

Usage: a run <node design>

run node

Flags:
  -h, --help        Show context-sensitive help.

  --discovery=ConnInfo,...  member discovery
  --hold=HEIGHT-FLAG      hold consensus states

logging
  --log.out="stderr"      log output file: {stdout, stderr, <file>}
  --log.format="terminal" log format: {json, terminal}
  --log.level=debug      log level: {trace, debug, info, warn, error}
  --[no-]log.force-color log force color

design
  --design=./config.yml    design uri; 'file:///config.yml', 'https://a.b.
↪c.d/config.yml'
  --[no-]design.https.tls_insecure  https tls insecure
```

### 3.2.1 Flags

- `--discovery`

*Discovery* flag is used to find the other suffrage nodes through *gossip* protocol. With valid *discovery* connection information, you can connect to the other suffrage nodes and vice versa.

*Discovery* connection information format is, `<host>:<port>[#tls_insecure]`. *host* and *port* should not be omitted. and *tls insecure* is optional. If host does not use public signed certificates, set `#tls_insecure`.

Examples:

```
- localhost:4320: ok
- localhost: bad
- localhost:4320#tls_insecure: ok
- 1.2.3.4:4320: ok
- 1.2.3.4:4320#tls_insecure: ok
```

- `--hold`

It is used only for testing. If `--hold` is given, node will stop every processes and do nothing.

Examples:

```
- --hold: At start time, node does not do anything
- --hold=10: After node stores block, height, 10, node holds
```

## 3.3 Command: *import*

```
$ ./mitum-example import --help

Usage: a import <node design> <from directory>

import from block data

Arguments:
  <from directory>    block data directory to import

Flags:
  -h, --help        Show context-sensitive help.

logging
  --log.out="stderr"    log output file: {stdout, stderr, <file>}
  --log.format="terminal"  log format: {json, terminal}
  --log.level=debug     log level: {trace, debug, info, warn, error}
  --[no-]log.force-color  log force color

design
  --design=./config.yml    design uri; 'file:///config.yml', 'https://a.b.
↪c.d/config.yml'
  --[no-]design.https.tls_insecure  https tls insecure
```

You can import the existing block data. By default, mitum creates and manages block data like this;

```
$ find ./no0
.
```

(continues on next page)

(continued from previous page)

```

./data
./data/000/000/000/000/000/019/454
./data/000/000/000/000/000/019/454/map.json
./data/000/000/000/000/000/019/454/voteproofs.ndjson
./data/000/000/000/000/000/019/454/proposal.json
./data/000/000/000/000/000/000/000
./data/000/000/000/000/000/000/000/map.json
./data/000/000/000/000/000/000/000/voteproofs.ndjson
./data/000/000/000/000/000/000/000/states.ndjson.gz
./data/000/000/000/000/000/000/000/operations.ndjson.gz
./data/000/000/000/000/000/000/000/proposal.json
./data/000/000/000/000/000/000/000/operations_tree.ndjson.gz
./data/000/000/000/000/000/000/000/states_tree.ndjson.gz
./data/temp
./db
./db/000277.ldb
./db/000281.ldb
./info.json

```

For importing from this block data, set `./no0/data`.

```
$ ./mitum-example import first-node.yml ./no0/data
```

## 3.4 Command: *network client*

```

$ ./mitum-example network client --help

Usage: a network client <header> [<remote>]

network client

Arguments:
  <network-id>  network-id
  <header>      request header; 'example' will print example headers
  [<remote>]    remote node conn info

Flags:
  -h, --help          Show context-sensitive help.

  --timeout=duration  timeout
  --body=BODY         body
  --dry-run           don't send

logging
  --log.out="stderr"    log output file: {stdout, stderr, <file>}
  --log.format="terminal" log format: {json, terminal}
  --log.level=debug     log level: {trace, debug, info, warn, error}
  --[no-]log.force-color log force color

```

### 3.4.1 Arguments

- header



Request header. Currently mitum supports these requests by header.

```
$ ./mitum-example network client example
example headers:
- blockmap:
  {"_hint":"blockmap-header-v0.0.1","height":33}
- blockmap_item:
  {"item":"blockmapitem_operations","height":33,"_hint":"blockmap-item-header-
↪v0.0.1"}
- exists_instate_operation:
  {"fact":"9a4vpzrBNpSb18HsMv2sSEASHuCW732az7on5w7RjWbm","_hint":"exists-
↪instate-operation-header-v0.0.1"}
- last_blockmap:
  {"manifest":"Fri87d7BMCF7hv6gWwQ6mrQyeBQycW4SBFFs5uLzKRN8","_hint":"last-
↪blockmap-header-v0.0.1"}
- last_suffrage_proof:
  {"state":"rLeHwzRPegvRnUBTYpkrsvhTEFLfXie4bHLun9ZXrwK","_hint":"last-
↪suffrage-proof-header-v0.0.1"}
- node_challenge:
  {"input":"WFXHRFulRVWtL64c+Z3z1Q==","_hint":"node-challenge-header-v0.0.1"}
- node_info:
  {"_hint":"node-info-header-v0.0.1"}
- operation:
  {"operation":"EgvRANZANRfjxQRSkE7A64m7puz5kh7zq5PEQzDsuxRg","_hint":
↪"operation-header-v0.0.1"}
- pprof:
  {"label":"heap","seconds":5,"gc":true,"_hint":"pprof-header-v0.0.1"}
- proposal:
  {"proposal":"E7gga4PaWPorzAXMAahkZwcgtwUjjCyJEizqka5cSzdx","_hint":"proposal-
↪header-v0.0.1"}
- request_proposal:
  {"_hint":"request-proposal-header-v0.0.1","proposer":"proposersas","point":{
↪"height":33,"round":1}}
- send_operation: $ cmd <header> --body=<json body>
  {"_hint":"send-operation-header-v0.0.1"}
- state:
  {"hash":"676Gug3J4Ugf8YTrNU2nN7mFZKCxwEvJrVhJVQBLNqsZ","key":"suffrage","_
↪hint":"state-header-v0.0.1"}
- suffrage_node_conninfo:
  {"_hint":"suffrage-node-conninfo-header-v0.0.1"}
- suffrage_proof:
  {"_hint":"suffrage-proof-header-v0.0.1","suffrage_height":44}
- sync_source_conninfo:
  {"_hint":"sync-source-conninfo-header-v0.0.1"}

* see isaac/network/header.go
```

– For example, you can request blockmap:

```
$ ./mitum-example network client \
  '{"_hint":"blockmap-header-v0.0.1","height":33}' \
  "localhost:4320#tls_insecure" \
  --timeout=3s
```

– Send operation:

```
$ cat "new-operation.json" | ./mitum-example network client \
  '{"_hint":"send-operation-header-v0.0.1"}' \
```

(continues on next page)

(continued from previous page)

```

"localhost:4320#tls_insecure" \
--timeout=3s \
-body=-

$ ./mitum-example network client \
'{"_hint":"send-operation-header-v0.0.1"}' \
"localhost:4320#tls_insecure" \
--timeout=3s \
-body=new-operation.json

```

- remote

Remote node connection information. It is same format with `--discovery` of run command.

### 3.4.2 Flags

- `--timeout`

Set timeout for request. If not set, will wait until finished by server.

- 3s: 3 seconds
- 3000000000: 3 seconds in nanoseconds

- `--body`

Set request body to upload to server. It should be file or *stdin*. If *stdin*, set `-` instead of file name.

- `--dry-run`

It will not send request to server, just read body.

## 3.5 Command: *key*

### 3.5.1 Command: *key new*

Generate new mitum *keypair* (*privatekey* and *publickey*)

```

$ ./mitum-example key new --help
Usage: a key new [<seed>]

generate new key

Arguments:
  [<seed>]    seed for generating key

Flags:
  -h, --help    Show context-sensitive help.

logging
  --log.out="stderr"      log output file: {stdout, stderr, <file>}
  --log.format="terminal" log format: {json, terminal}
  --log.level=debug      log level: {trace, debug, info, warn, error}
  --[no-]log.force-color log force color

```

(continues on next page)

(continued from previous page)

```
$ ./mitum-example key new
{
  "privatekey": "EGGQu4bCWDy1p4RhEZCdE7vP4hP1UeN2jS8U8s7zAomhmpr",
  "publickey": "29Zrw2ZgWyeeHapnxc6cU8D19f5exjqkB4Mk2Bc2ACYyrmpu",
  "hint": "mpr-v0.0.1",
  "seed": "",
  "type": "privatekey"
}
```

## Arguments

- seed

Optional. With *seed* string, new kepair is generated from seed. As you expected, same *seed* will generated same kepair.

Without seed, different kepair will be generated in each time.

### 3.5.2 Command: *key load*

Load mitum *key* (*privatekey* or *publickey*) and validate it.

```
$ ./mitum-example key load --help
Usage: a key load <key string>

load key

Arguments:
  <key string>    key string

Flags:
  -h, --help      Show context-sensitive help.

logging
  --log.out="stderr"      log output file: {stdout, stderr, <file>}
  --log.format="terminal" log format: {json, terminal}
  --log.level=debug       log level: {trace, debug, info, warn, error}
  --[no-]log.force-color log force color

# load the privatekey, EGGQu4bCWDy1p4RhEZCdE7vP4hP1UeN2jS8U8s7zAomhmpr
$ ./mitum-example key load EGGQu4bCWDy1p4RhEZCdE7vP4hP1UeN2jS8U8s7zAomhmpr
{
  "privatekey": "EGGQu4bCWDy1p4RhEZCdE7vP4hP1UeN2jS8U8s7zAomhmpr",
  "publickey": "29Zrw2ZgWyeeHapnxc6cU8D19f5exjqkB4Mk2Bc2ACYyrmpu",
  "hint": "mpr-v0.0.1",
  "string": "EGGQu4bCWDy1p4RhEZCdE7vP4hP1UeN2jS8U8s7zAomhmpr",
  "type": "privatekey"
}

# load the publickey, 29Zrw2ZgWyeeHapnxc6cU8D19f5exjqkB4Mk2Bc2ACYyrmpu
$ ./mitum-example key load 29Zrw2ZgWyeeHapnxc6cU8D19f5exjqkB4Mk2Bc2ACYyrmpu
{
  "publickey": "29Zrw2ZgWyeeHapnxc6cU8D19f5exjqkB4Mk2Bc2ACYyrmpu",
  "hint": "mpu-v0.0.1",
```

(continues on next page)

(continued from previous page)

```

"string": "29Zrw2ZgWyeeHapnxc6cU8D19f5exjqkB4Mk2Bc2ACYyrmpu",
"type": "publickey"
}

```

## Arguments

- key string

Load key, parse and validate it. Key can be *privatekey* or *publickey*.

### 3.5.3 Command: *key sign*

Sign message by the given *privatekey*

```

$ ./mitum-example key sign --help
Usage: a key sign <privatekey> <network-id> <body>

sign

Arguments:
  <privatekey>  privatekey string
  <network-id>  network-id
  <body>        body

Flags:
  -h, --help          Show context-sensitive help.
  --node=ADDRESS-FLAG node address
  --token=STRING      set fact token

logging
  --log.out="stderr"  log output file: {stdout, stderr, <file>}
  --log.format="terminal" log format: {json, terminal}
  --log.level=debug   log level: {trace, debug, info, warn, error}
  --[no-]log.force-color log force color

# suffrage-candidate-nolsas.json is the operation to be signed
$ cat ./suffrage-candidate-nolsas.json
{
  "fact": {
    "address": "nolsas",
    "publickey": "25AZEiKTPhNkpcj6BlmofXHFyJRR8DaEMcNjc2WSvvW8Jmpu",
    "token": "6qLkX1LfSXejcuzijomt+w==",
    "_hint": "suffrage-candidate-fact-v0.0.1"
  },
  "_hint": "suffrage-candidate-operation-v0.0.1"
}

# sign by the privatekey, CaSheUmWGeAYgAKwnwdYrDuJ5fkr2wsVXSpmGFTEUpYtmpr
$ ./mitum-example key sign \
  CaSheUmWGeAYgAKwnwdYrDuJ5fkr2wsVXSpmGFTEUpYtmpr \
  "mitum; Sat 26 Dec 2020 05:29:13 AM KST" \
  ./suffrage-candidate-nolsas.json \

```

(continues on next page)

(continued from previous page)

```

--node nolsas \
--token findme
{
  "hash": "9PJgJA17dtCiGkTDRTFaK1fLiD3zbPYTNrYDpQhzMcYg",
  "fact": {
    "address": "nolsas",
    "publickey": "25AZEiKTPhNkpcj6B1mofXHFyJRR8DaEMcNjc2WSvvW8Jmpu",
    "hash": "GxhAT8KoWyS8E9dlzM47c5anGrNFC8XiCwWEuAdtSexZ",
    "token": "ZmluZG1l",
    "_hint": "suffrage-candidate-fact-v0.0.1"
  },
  "signed": [
    {
      "node": "nolsas",
      "signed_at": "2022-08-07T04:00:45.499676834Z",
      "signer": "25AZEiKTPhNkpcj6B1mofXHFyJRR8DaEMcNjc2WSvvW8Jmpu",
      "signature":
↵ "AN1rKvtAWGE3APxii4jFfe6gzkTBAhqpQiMcLKKQJuSrWRgjGuMUsnG4aspLs3yJbsYgtkpsBLteVTn2vi4LhVn95GRubtWqf
↵ "
    }
  ],
  "_hint": "suffrage-candidate-operation-v0.0.1"
}

```

## Arguments

- `privatekey`  
Privatekey for signing
- `network-id`  
\*Network ID for signing
- `body`  
Message body to be signed

## Flags

- `node`  
Some message needs to be signed with *node address*
- `token`  
Update *token* of *operation*