It-chain - Customizable and Lightweight Blockchain

Kosslab 5기 전담 개발자
발표자: 이준범

소개

이준범

- Kosslab 5기 전담 개발자
- 카이스트 소프트웨어 아키텍처 연구실 석사과정
- 중앙대학교 컴퓨터 공학부 졸업
- 취미: 카페에서 코딩하기
- 관심 분야

  소프트웨어 아키텍처, 머신러닝(딥러닝,NMT), 블록체인

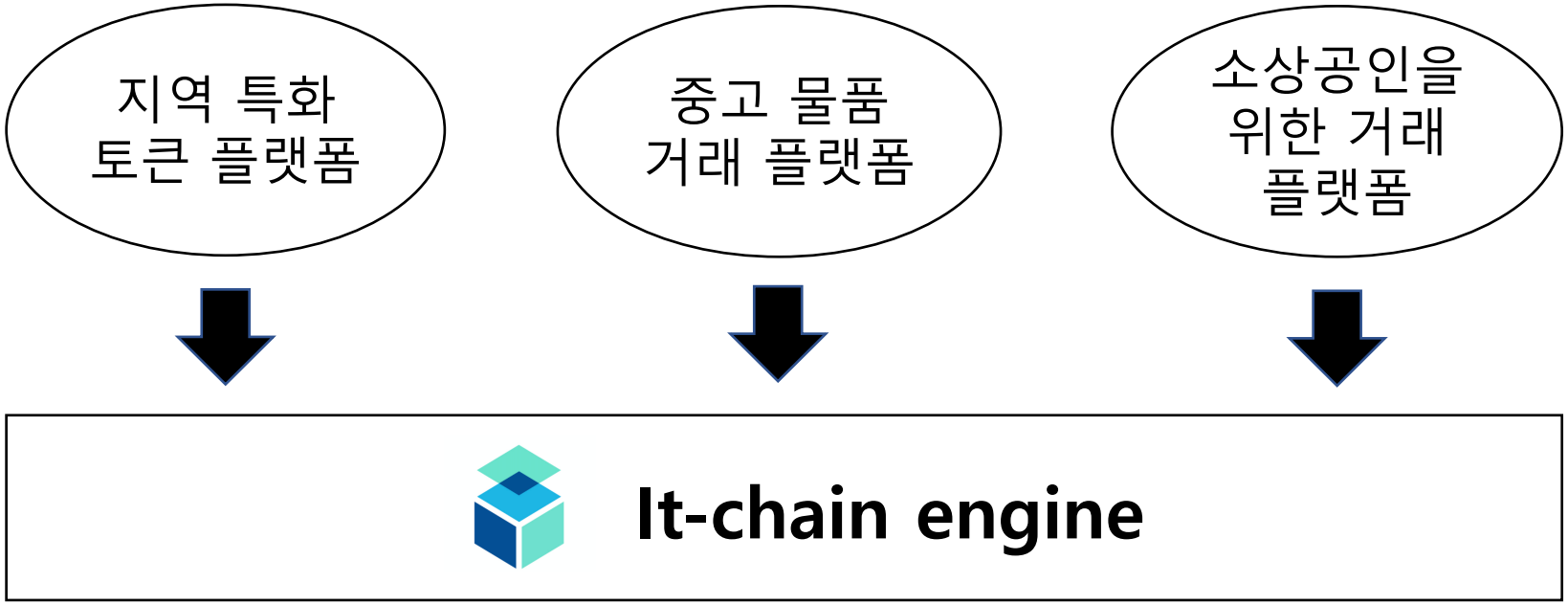
목차

It-chain

중소규모 커뮤니티에서 유연하고 자유롭게 이용 가능한

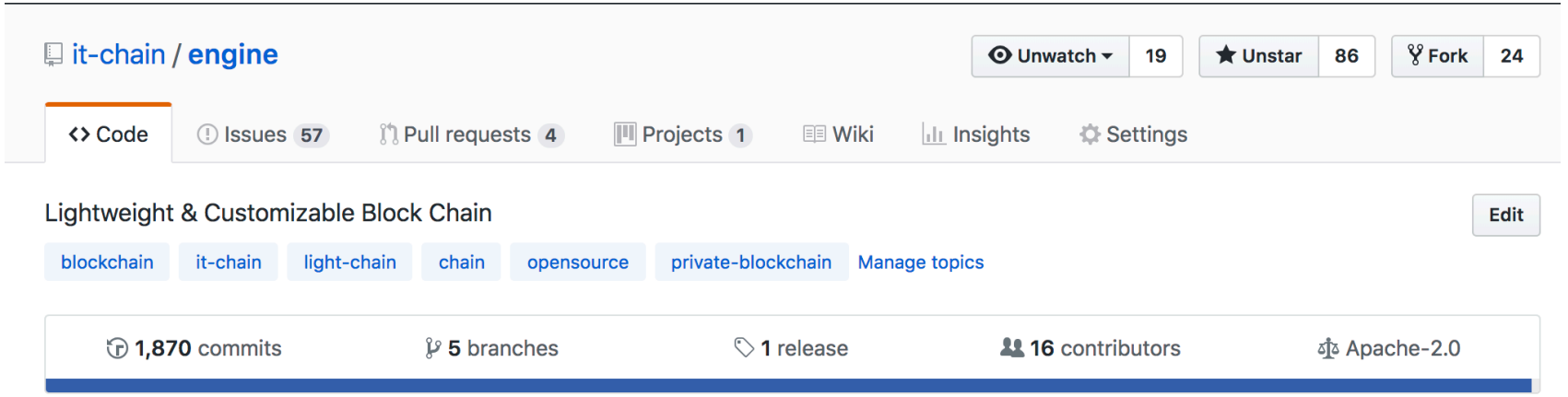
'경량 맞춤형 블록체인 엔진' 개발 및 블록체인 코어 공부

- 핵심 서비스의 모듈화를 통해 목적에 맞게 쉽게 교체 가능 하도록 개발



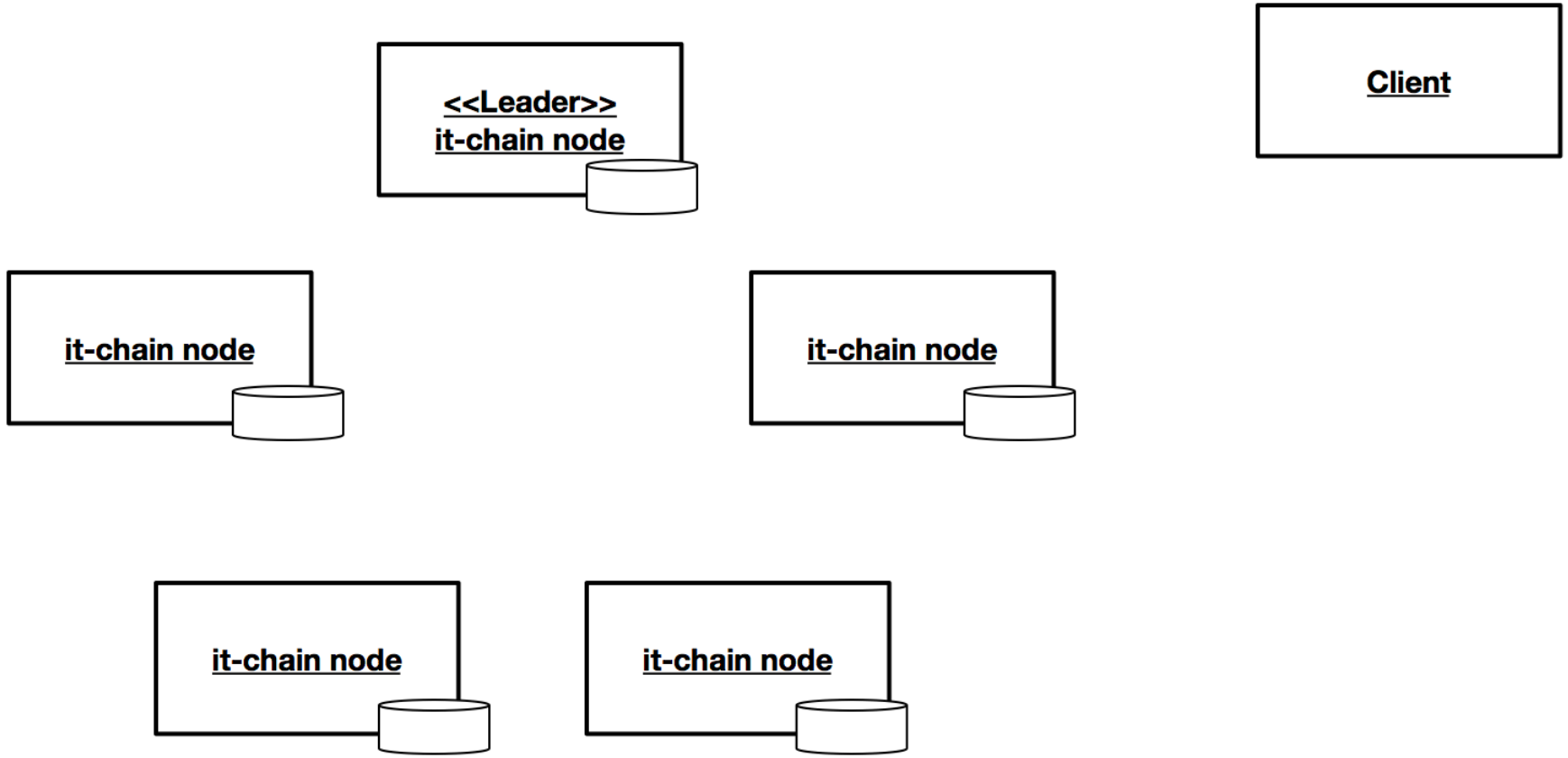
It-chain

- It-chain 오픈소스 커뮤니티 (2018/01 ~ 진행중)
- Github: https://github.com/it-chain
- 커미터: 8명(대학생, 대학원생, 직장인 등으로 구성)
- 컨트리뷰터: 16명



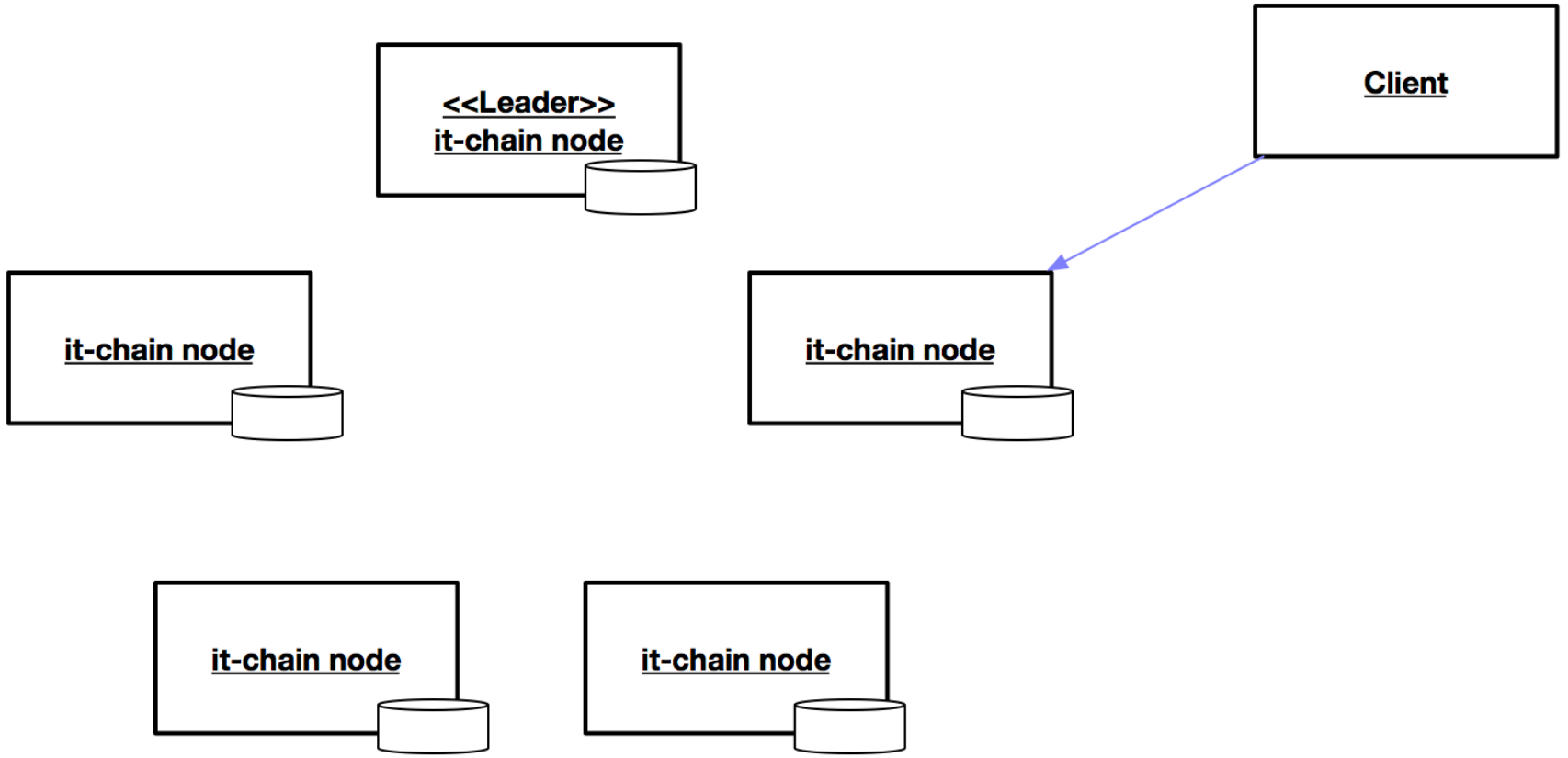
It-chain

- Network 구성



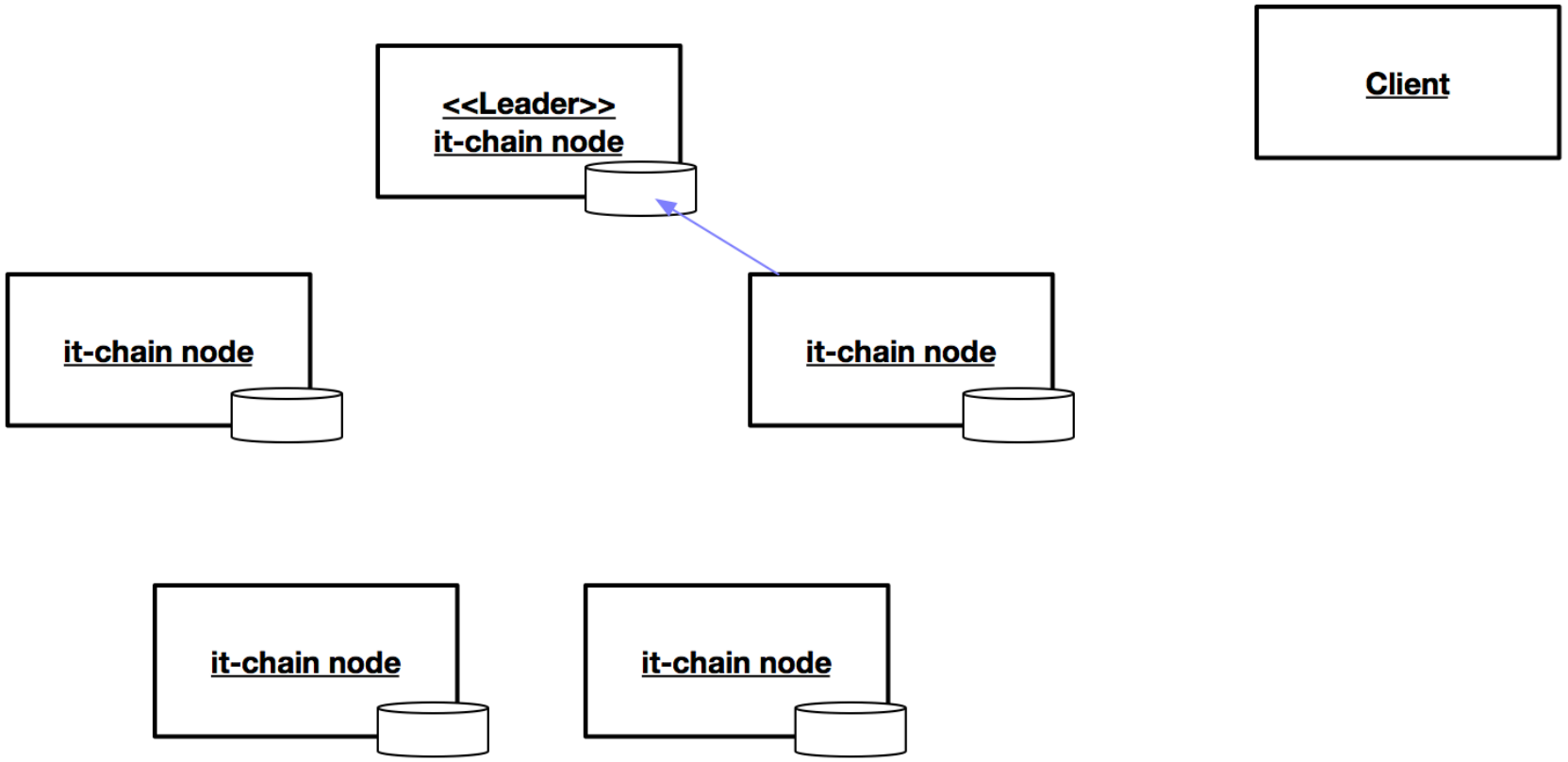
It-chain

- **Submit Request**



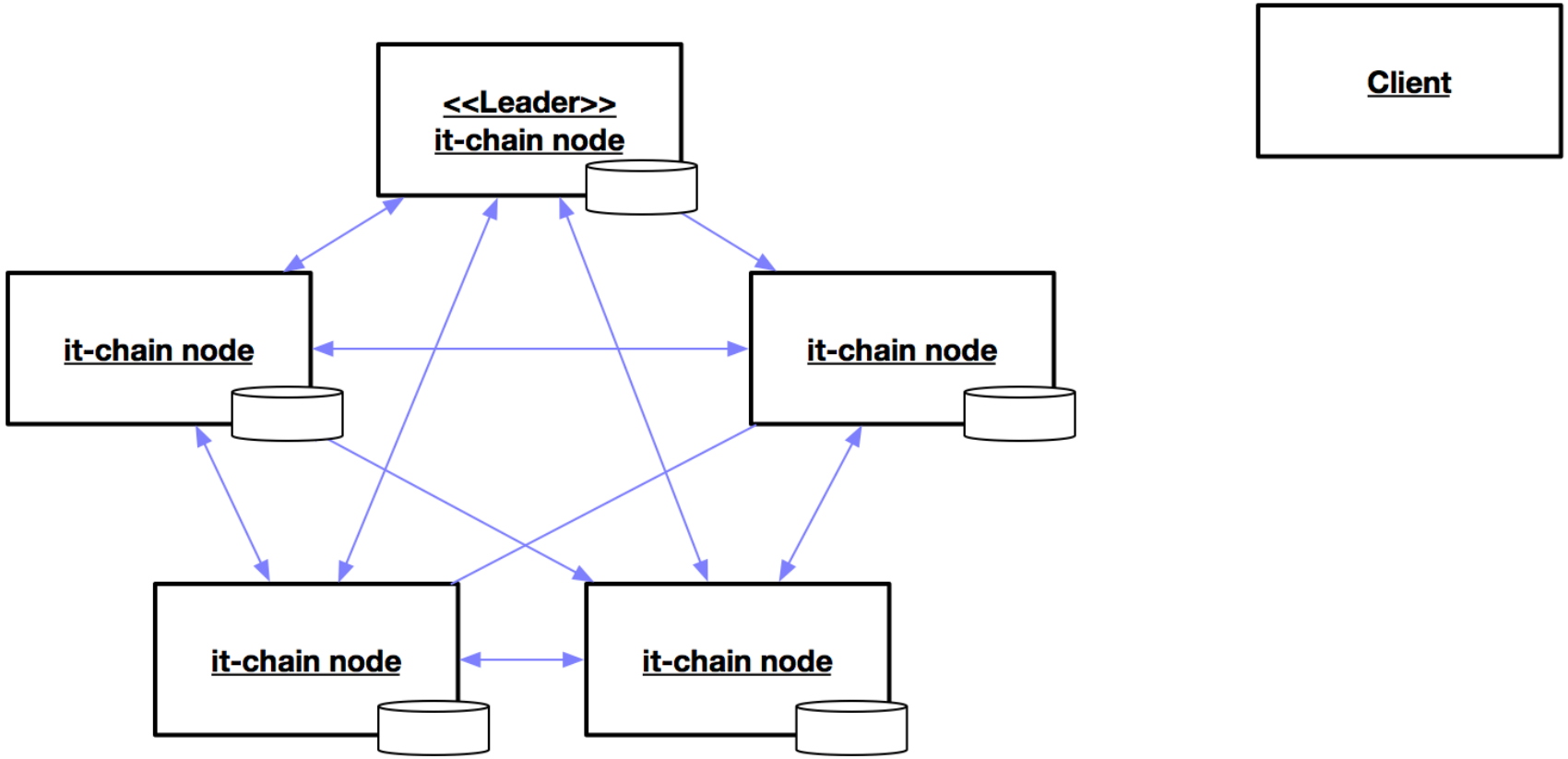
It-chain

- Forward to leader



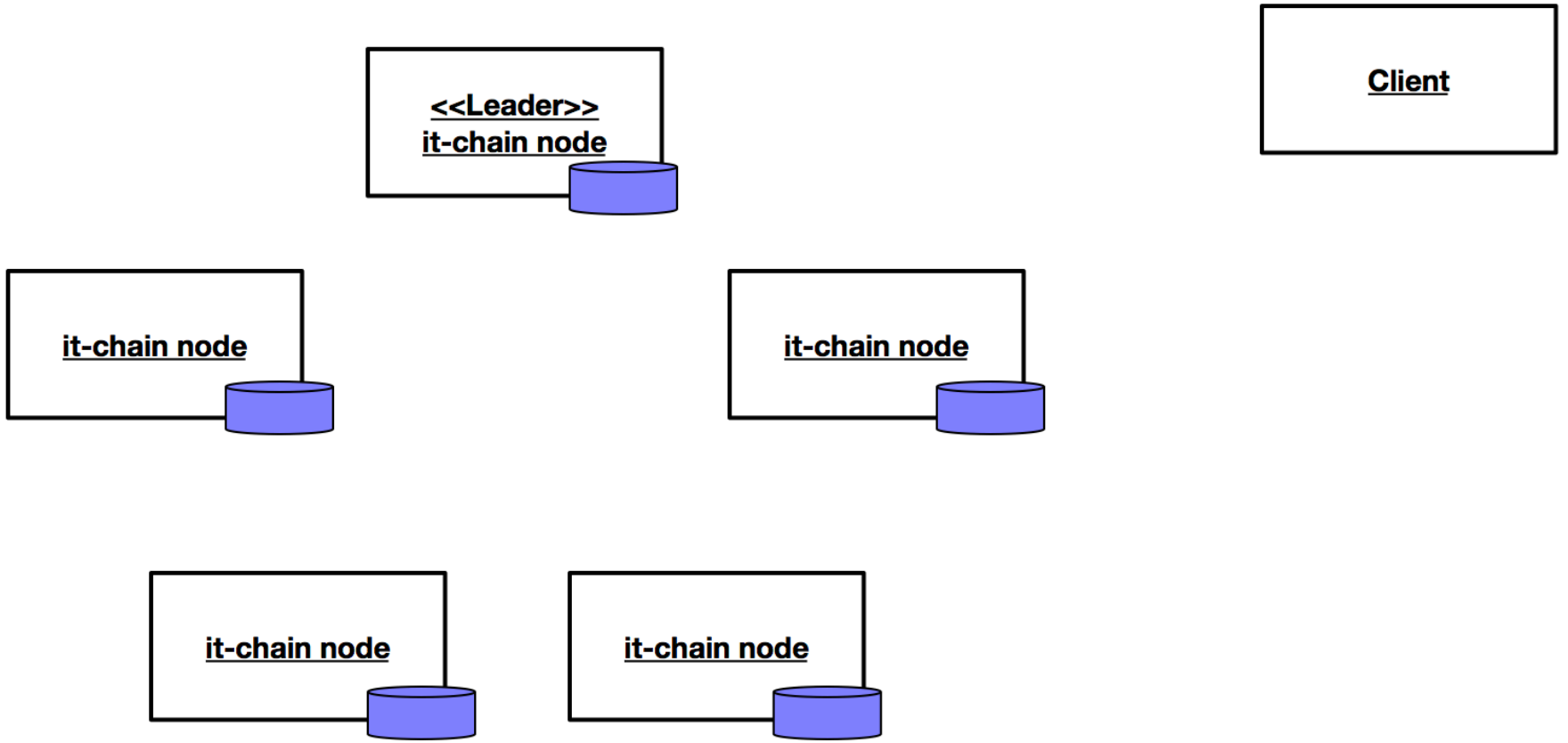
It-chain

- Start PBFT



It-chain

- Add block and update state



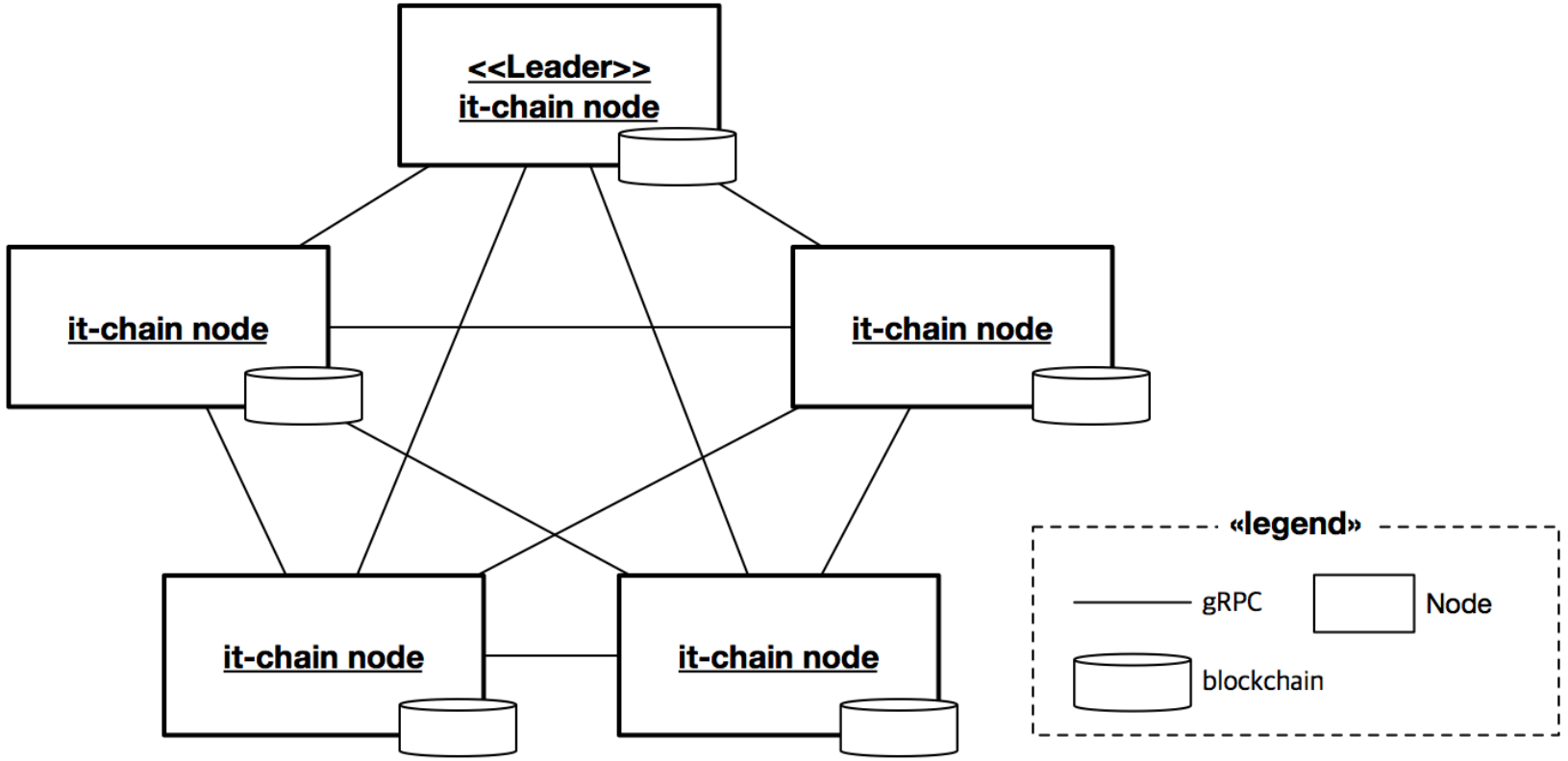
Architecture - It-chain 초기 설계 과정

- **요구사항**
  - 제3자가 손쉽게 수정/확장할 수 있으면 좋겠음
  - 오픈소스로 해야 하니까, 최대한 개발자들이 자기 영역에서 간섭받지 않고 개발을 진행할 수 있으면 좋겠군
  - 중앙 관리자 없이 P2P구조를 가져야 할 것 같아
  - 기본적인 블록체인 기능들(블록저장, 합의, 인증)을 지원해야해
  - Consensus는 PBFT를 지원하면 좋겠어

Architecture - It-chain 초기 설계 과정

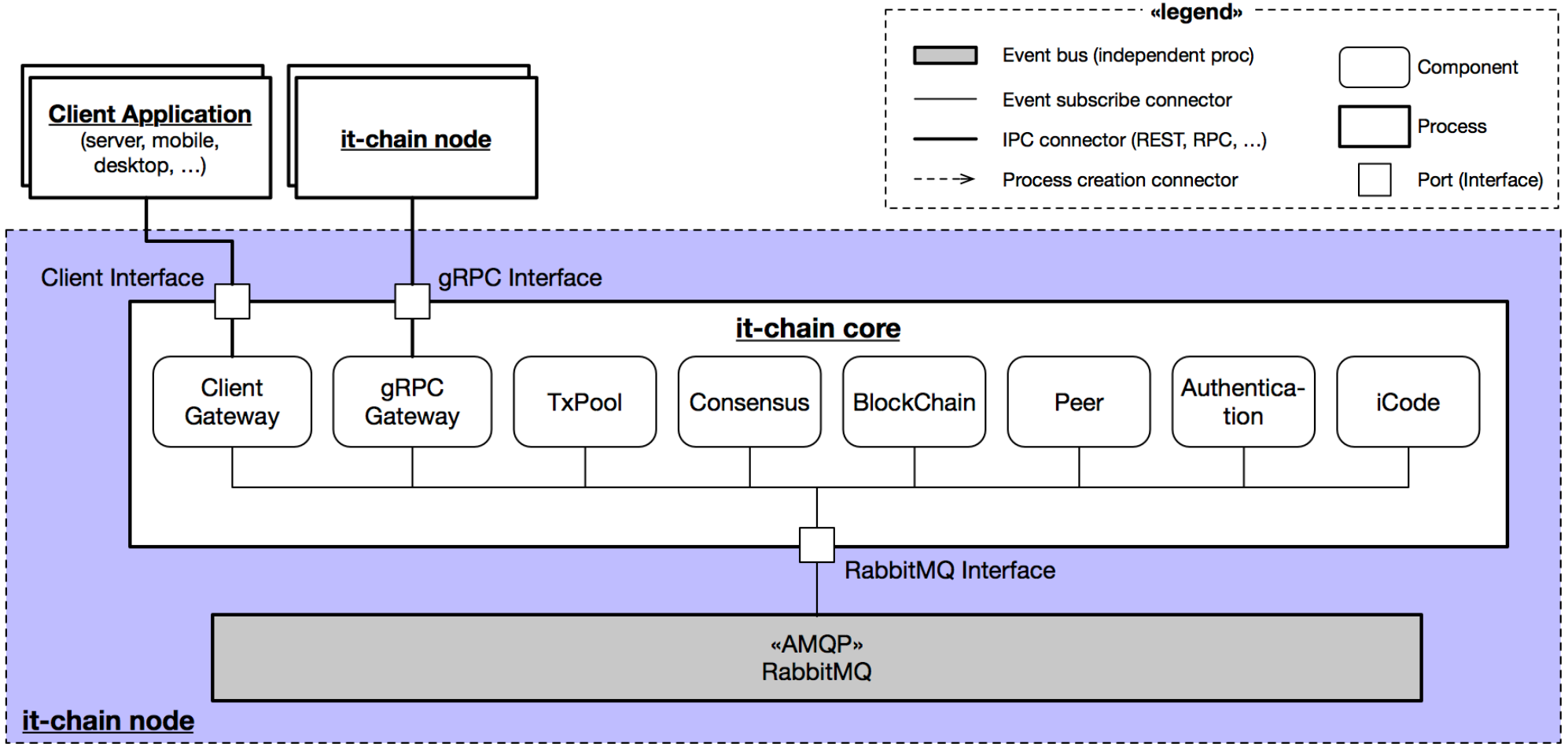
- **아키텍처 설계**
  - 제3자가 손쉽게 수정/확장할 수 있으면 좋겠음
  - 오픈소스로 해야 하니까, 최대한 개발자들이 자기 영역에서 간섭받지 않고 개발을 진행할 수 있으면 좋겠군
    - **각 블록체인 기능을 지원하는 컴포넌트들을 독립적으로 구성**
    - **Event Driven 아키텍처 스타일 도입**
  - 중앙 관리자 없이 P2P구조를 가져야 할 것 같아
    - **P2P Network 구성**
  - 기본적인 블록체인 기능들(블록저장, 합의, 인증)을 지원해야해
    - **Block Storage 컴포넌트**
    - **Authentication 컴포넌트**
    - **Consensus 컴포넌트**
  - Consensus는 PBFT를 지원하면 좋겠어
    - **Transaction pool 컴포넌트, 리더 노드 개념 도입**

Architecture - 현재



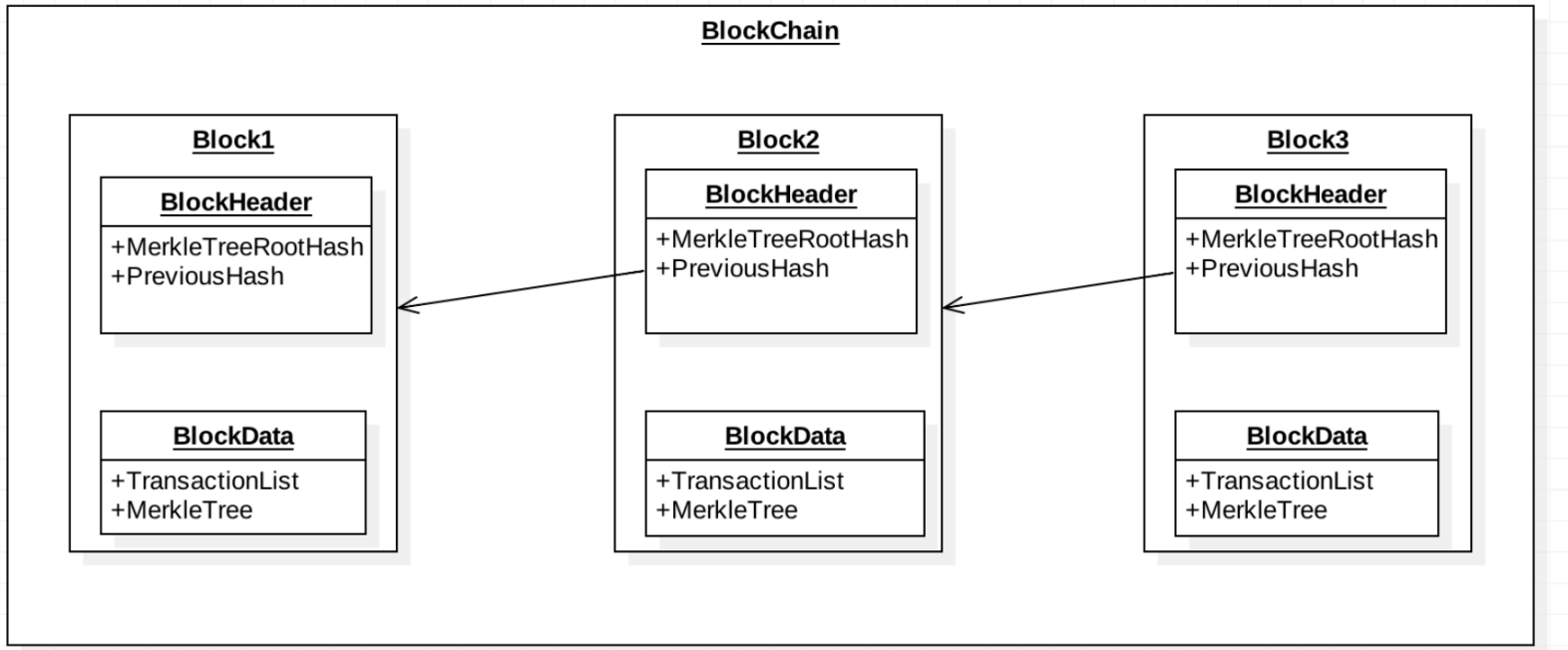
it-chain network는 it-chain node로 구성된 P2P network
Leader와 Peer로 구성(PBFT)

Architecture - 현재



it-chain은 6개의 독립적으로 동작하는 핵심 컴포넌트들로 구현되며
각각은 AMQP(Asynchronous Message Queue Protocol)를 통해 커뮤니케이션한다.

Block, Transaction - Yggdrasil



블록 체인은 끊임없이 증가하는 블록 리스트로, 다음 블록은 이전 블록의 해시 값을 가진다. (위변조가 어려움)

Block, Transaction - Yggdrasil

Block

- Yggdrasil/Impl에서 DefaultBlock.go 제공
- Header와 Data로 구분
- Header에는 Block에 대한 Meta정보 제공

```
type DefaultBlock struct {
    Header       BlockHeader
    MerkleTree   [][]string
    Transactions []tx.Transaction
}

type BlockHeader struct {
    Height              uint64
    PreviousHash        string
    Version             string
    MerkleTreeRootHash  string
    TimeStamp           time.Time
    CreatorID           string
    Signature           []byte
    BlockHash           string
    MerkleTreeHeight    int
    TransactionCount    int
}
```

Block, Transaction - Yggdrasil

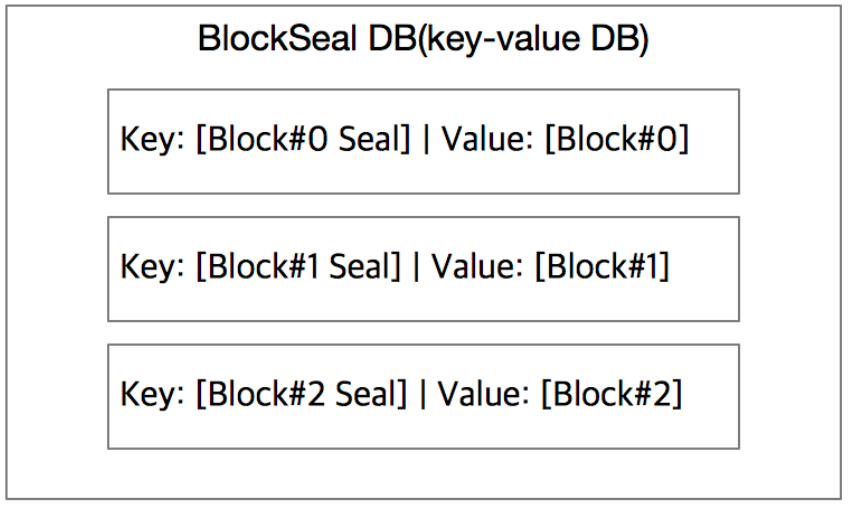
Transaction

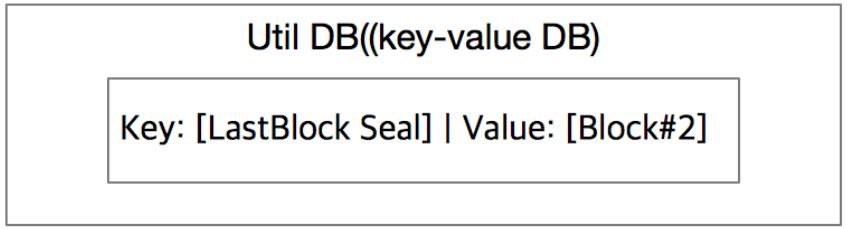
- It-chain의 Transaction은 ICode실행 요청
- JSON-RPC

```
// TxData의 Type을 정의하는 상수들
const (
    Invoke TxDataType = "invoke"
    Query  TxDataType = "query"
)

// Params 구조체는 Jsonrpc에서 invoke하는 함수의 패러미터를 정의한다.
type Params struct {
    Type     int
    Function string
    Args     []string
}

// TxData 구조체는 Jsonrpc에서 invoke하는 함수를 정의한다.
type TxData struct {
    Jsonrpc string
    Method  TxDataType
    Params  Params
    ID      string
}

// DefaultTransaction 구조체는 Transaction 인터페이스의 기본 구현체이다.
type DefaultTransaction struct {
    ID        string
    Status    Status
    PeerID    string
    Timestamp time.Time
    TxData    *TxData
    Signature []byte
}
```

Block, Transaction - Yggdrasil

Block 저장



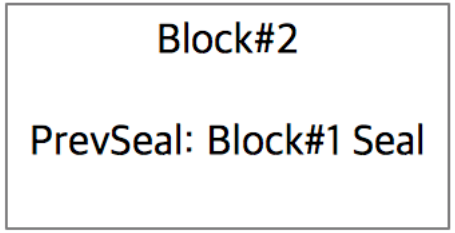


LastBlock Seal은 Predefined Key

LastBlock은 항상 util DB에 저장

Block 조회

- Block Seal 값을 key로 Value 조회
- LastBlock으로 부터 순차적으로 조회가능



Block, Transaction - Yggdrasil Code

```
serializedBlock, err := block.Serialize()
if err != nil {
    return err
}

err = y.validateBlock(block)
if err != nil {
    return err
}

utilDB := y.DBProvider.GetDBHandle(utilDB)
blockSealDB := y.DBProvider.GetDBHandle(blockSealDB)
blockHeightDB := y.DBProvider.GetDBHandle(blockHeightDB)
transactionDB := y.DBProvider.GetDBHandle(transactionDB)
err = blockSealDB.Put(block.GetSeal(), serializedBlock, true)
if err != nil {
    return err
}

err = blockHeightDB.Put([]byte(fmt.Sprint(block.GetHeight())), block.GetSeal(), true)
if err != nil {
    return err
}

err = utilDB.Put([]byte(lastBlockKey), serializedBlock, true)
if err != nil {
    return err
}
```

```
for _, tx := range block.GetTxList() {
    serializedTX, err := tx.Serialize()
    if err != nil {
        return err
    }
    err = transactionDB.Put([]byte(tx.GetID()), serializedTX, true)
    if err != nil {
        return err
    }
    err = utilDB.Put([]byte(tx.GetID()), block.GetSeal(), true)
    if err != nil {
        return err
    }
}
```

Consensus

It-chain Network

- Permissioned P2P Network(Private Network)
- 소수의 노드로 구성
- 빠른 합의 속도

➔ Tendermint(PBFT) Consensus 알고리즘 사용

Consensus

Consensus 알고리즘

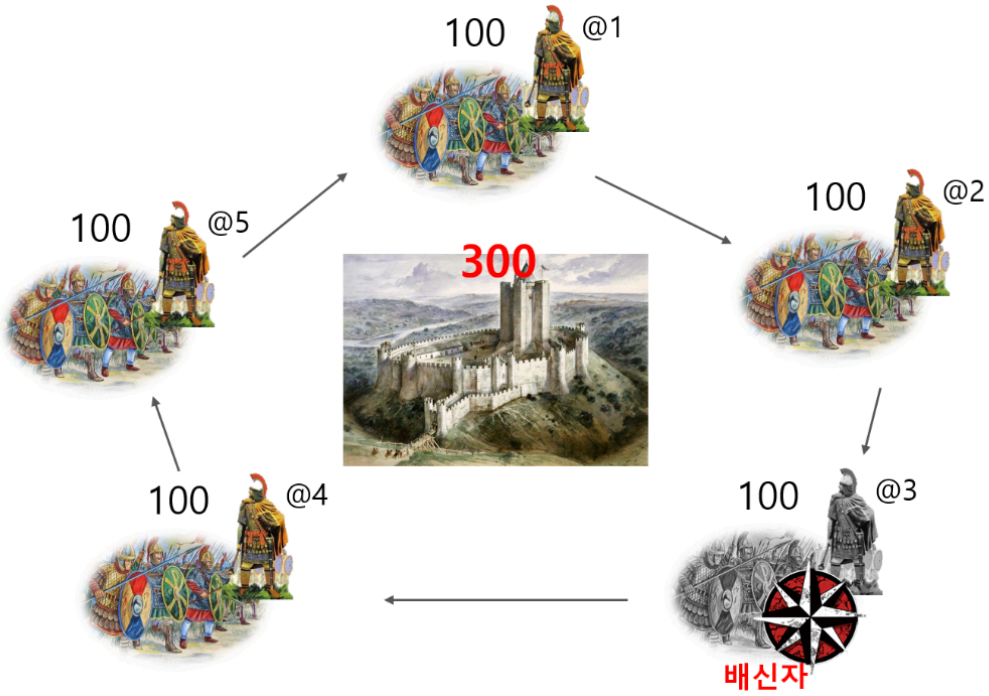
- 분산된 프로세스 또는 시스템간의 단일 데이터 값에 대한 합의를 달성하는데 사용되는 알고리즘
- 여러 개의 신뢰할 수 없는 노드가 포함된 네트워크에서 안정성을 달성하도록 설계
- 비잔틴 장군 문제 해결

	PoW	PoS	PBFT
Major Blockchain Platform	Bitcoin, Ethereum	Cardano	Hyperledger fabric 0.6

Consensus

비잔틴 장군 문제

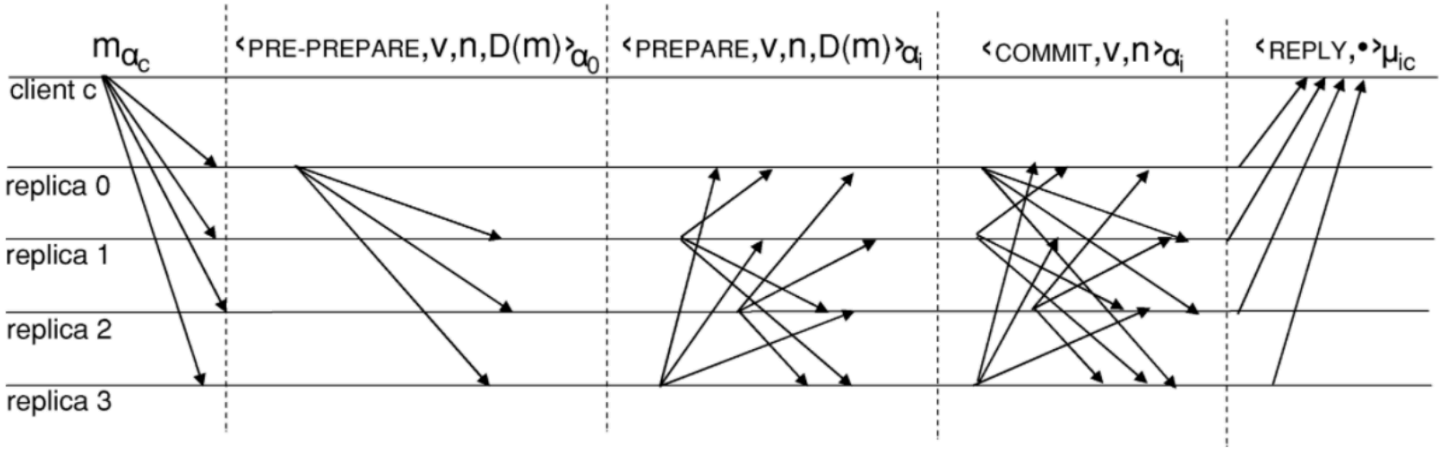
- 5명중 3명 이상의 장군들이 같은 시각에 공격해야만 승리
- 장군들 중에는 배신자가 있어서 서로 신뢰할 수 없음



Consensus

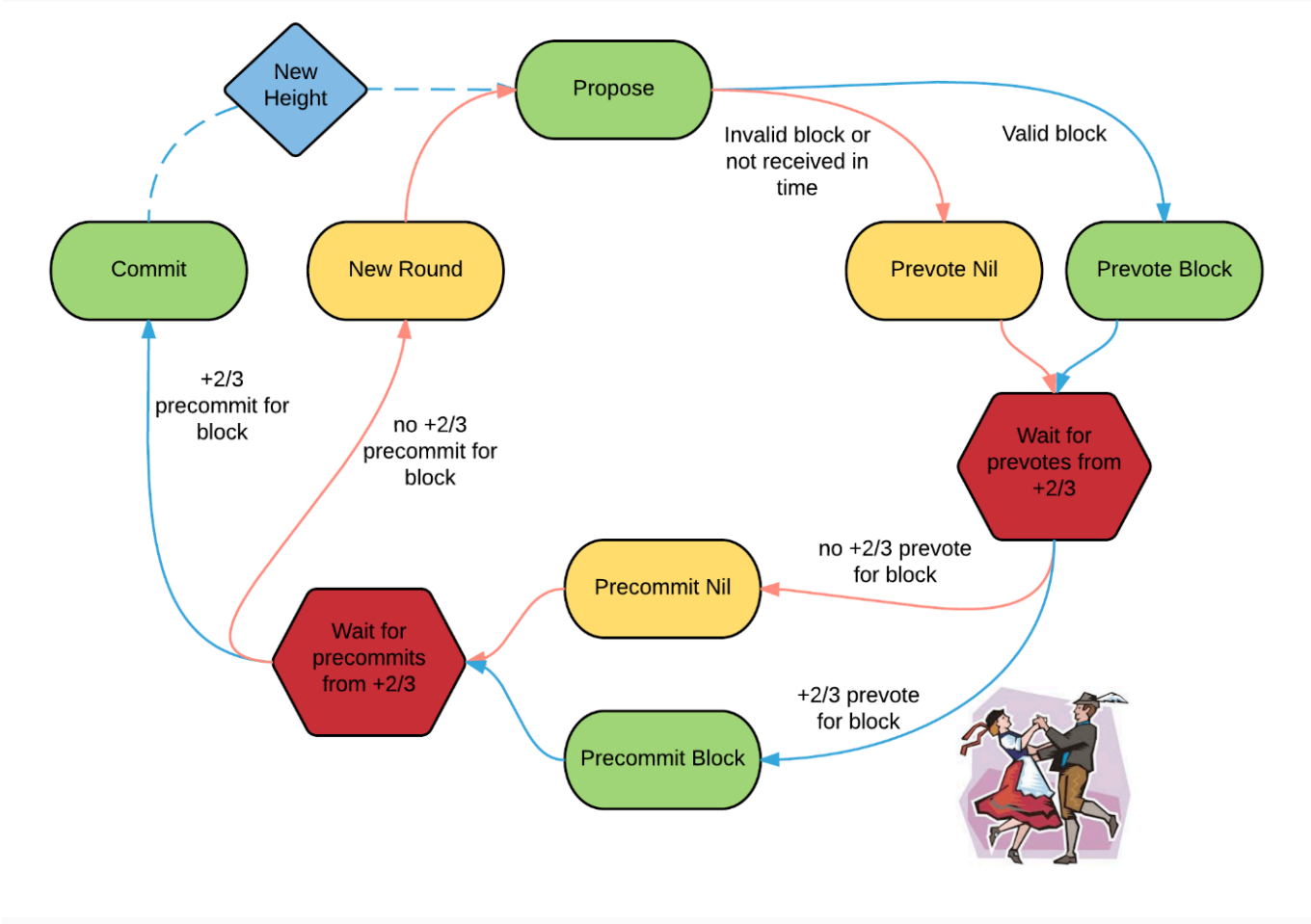
PBFT

- 1999년 Miguel Castro와 Barbara Riskop에 의해 도입 된 알고리즘
- 네트워크의 모든 노드는 사전에 알고 있어야하며, 한 노드는 리더



Consensus

PBFT(Tendermint)

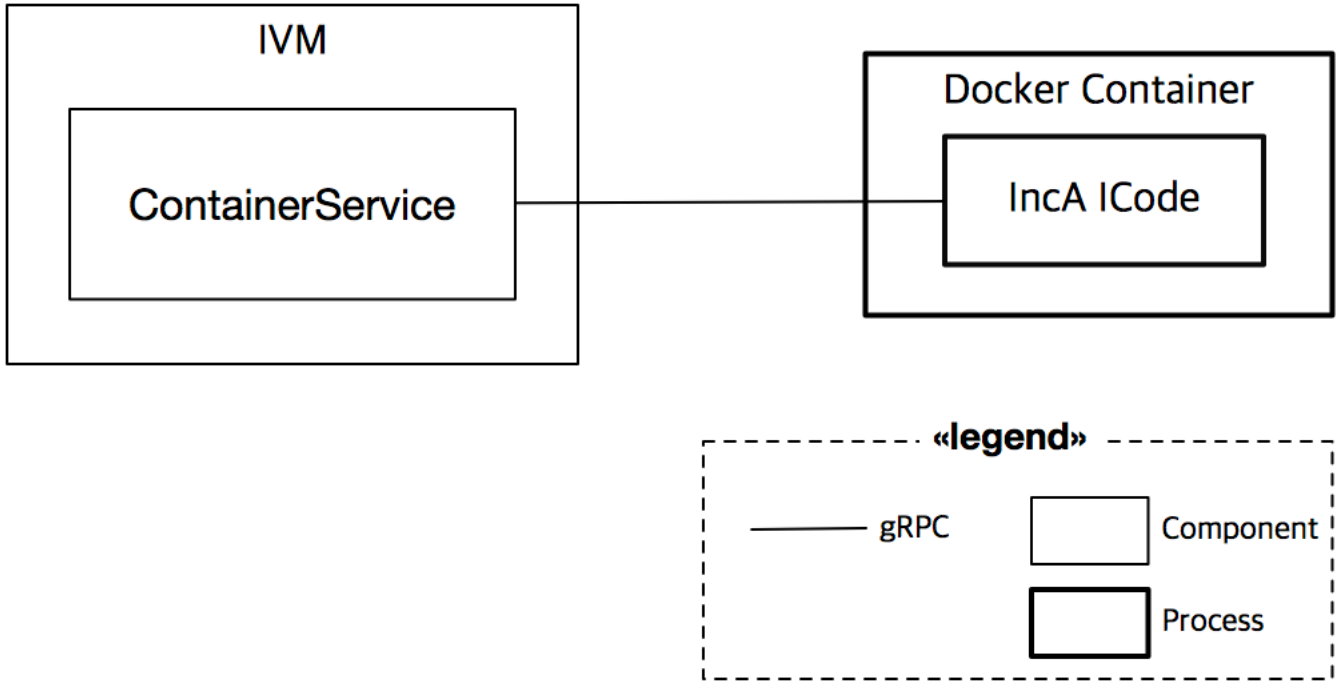


IVM(ICode Virtual Machine)

- It-chain 스마트 컨트랙 deploy, invoke 수행
- World State DB 관리
- Docker 컨데이터 기반의 독립적인 환경 제공
- SDK를 통해 다양한 언어 지원 가능(현재 go 지원)
- 예제: https://github.com/junbeomlee/learn-icode

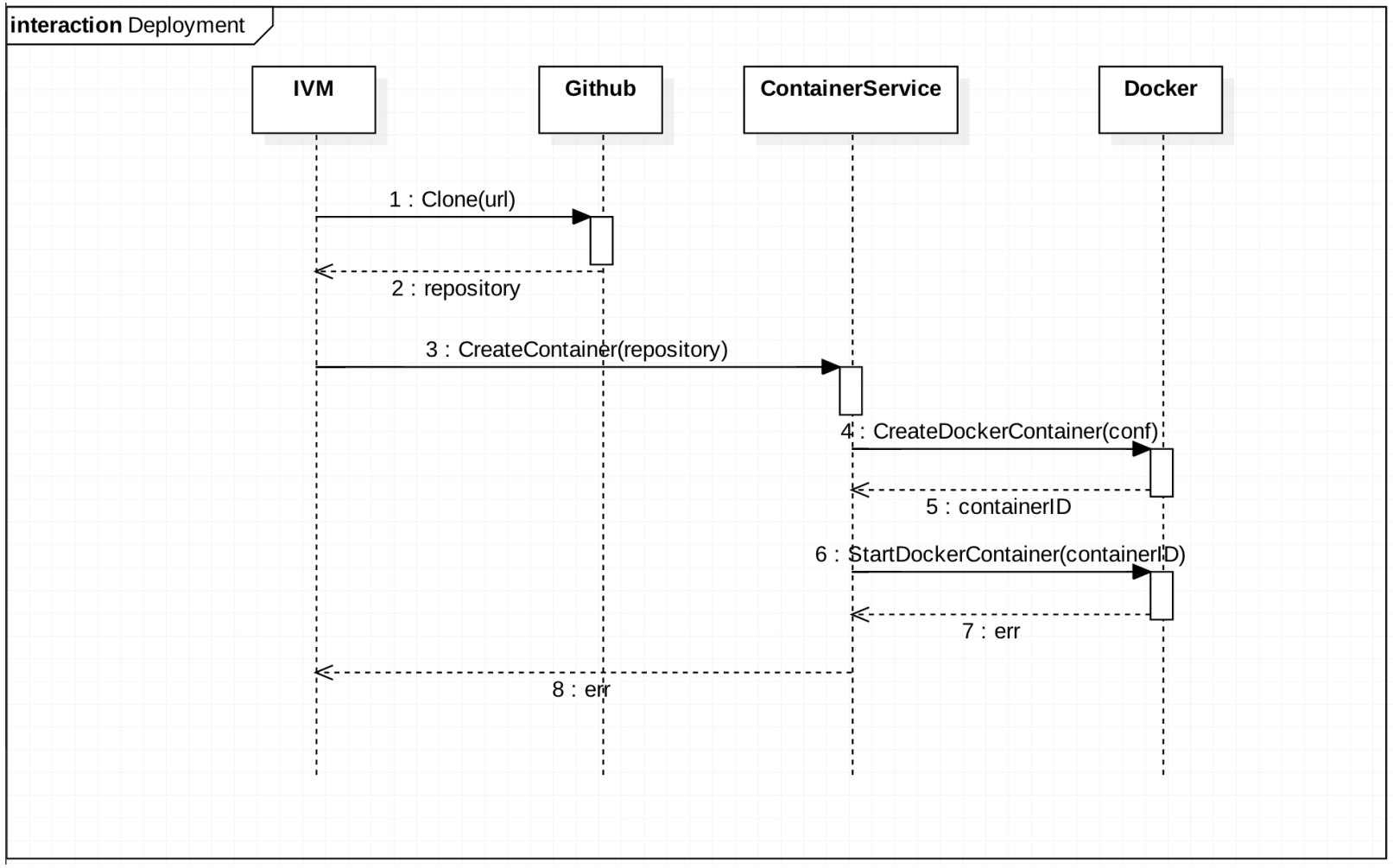
IVM

- **Architecture**
  - **ICode는 self-contained**
  - **gRPC로 통신**



IVM

- **Deployment**



./It-chain icode deploy github.com/junbeomlee/learn-icode ~/.ssh/id_rsa

IVM - It-chain SDK

- https://github.com/it-chain/sdk
- IVM에 올리기 위한 library제공
- RequestHandler interface를 구현한 Handler를 작성

```
package sdk

import "github.com/it-chain/sdk/pb"

type RequestHandler interface {
    Name() string
    Versions() []string
    Handle(request *pb.Request, cell *Cell) *pb.Response
}
```

IVM - ICode Example

```go
type HandlerExample struct {
}

func (*HandlerExample) Name() string {
    return "sample"
}

func (*HandlerExample) Versions() []string {
    vers := make([]string, 0)
    vers = append(vers, "1.0")
    vers = append(vers, "1.2")
    return vers
}

func (*HandlerExample) Handle(request *pb.Request, cell *sdk.Cell) *pb.Response {
    switch request.Type {
    case "invoke":
        return handleInvoke(request, cell)
    case "query":
        return handleQuery(request, cell)
    default:
        logger.Debug(nil, "unknown request type")
        err := errors.New("unknown request type")
        return responseError(request, err)
    }
}
```

```go
func handleInvoke(request *pb.Request, cell *sdk.Cell) *pb.Response {
    switch request.FunctionName {
    case "initA":
        err := cell.PutData("A", []byte("0"))
        if err != nil {
            return responseError(request, err)
        }
        return responseSuccess(request, nil)
    default:
        err := errors.New("unknown invoke method")
        return responseError(request, err)
    }
}
```

```proto
message Request {
    string uuid = 1;
    string Type = 2;
    string FunctionName = 3;
    repeated string Args = 4;
}
```

감사합니다.
QnA