# 기초부터 시작하기

## Raft 합의 알고리즘의 쿼럼

1. [[셋업하기|셋업하기]] 섹션에서 설명하는대로 Quorum을 빌딩하십시오. PATH에 geth와 bootnode가 있는지 확인하십시오.
2. 새 노드의 기반이 될 작업 디렉토리를 만들고 변경하십시오.
3. 이 노드를 위해 geth를 사용하여 한 개 이상의 계정을 만들고 (--datadir new-node-1 account new) 계정 주소를 제거하십시오. 하고자 하는 것에 따라 자금이 있는 계좌가 필요할 수 있습니다.
4. genesis.json 파일을 만드십시오. [여기](https://github.com/jpmorganchase/quorum-examples/blob/master/examples/7nodes/genesis.json)에 예제가 있습니다. 이전 단계에서 생성한 계정으로 alloc 필드를 미리 입력해야 합니다.
5. 노드 키 bootnode 생성 (--genkey=nodekey)하고 datadir에 복사
6. bootnode --nodekey=new-node-1/nodekey --writeaddress 를 실행하고 표시된 아웃풋을 기록하십시오. 이것은 새로운 노드의 enode 아이디입니다.
7. static-nodes.json라는 파일을 만들고 [예제](https://github.com/jpmorganchase/quorum-examples/blob/master/examples/7nodes/permissioned-nodes.json)와 일치하도록 편집하십시오. 파일에는 enode id오devp2p 및 raft에 사용할 포트가 있는 노드에 대한 라인이 하나 있어야 합니다. 이 파일이 노드 데이터 디렉토리에 있는지 확인하십시오.
8. geth로 새로운 노드를 초기화하십시오 --datadir new-node-1 init genesis.json
9. 노드를 시작하고 다음을 통해 백그라운드로 보내십시오. PRIVATE\_CONFIG=ignore nohup geth --datadir new-node-1 --nodiscover --verbosity 5 --networkid 31337 --raft --raftport 50000 --rpc --rpcaddr 0.0.0.0 --rpcport 22000 --rpcapi admin,db,eth,debug,miner,net,shh,txpool,personal,web3,quorum,raft --emitcheckpoints --port 21000 2>>node.log &

당신의 노드는 이제 작동 가능하며 geth attach new-node-1/geth.ipc를 연결할 수 있습니다. 이 구성은 프리픽스 PRIVATE\_CONFIG=ignore에서 확인할 수 있는 것처럼 프라이버시 서포트 없이 Quorum을 시작합니다. [[프라이버시 트랜잭션 매니저로 프라이버시 지원 방법|기초부터#adding-privacy-transaction-manager]] 섹션을 참조하십시오.

### 노드 추가하기

1. 앞의 가이드의 1, 2, 5, 6단계를 완료하십시오.
2. 현재 체인인 genesis.json 및 static-nodes.json.json를 회수하십시오. static-nodes.json 은 노드 data dir에 배치되어야 합니다.
3. geth로 새로운 노드를 초기화 하십시오 --datadir new-node-2 init genesis.json
4. static-nodes.json을 편집하고 구성중인 새 노드에 대한 새로운 항목을 추가하십시오 (마지막에 있어야 함)
5. 이미 실행중인 체인 노드에 연결하고 실행합니다.
6. raft.addPeer('enode://new-nodes-enode-address-from-step-6-of-the-above@127.0.0.1:21001?discport=0&raftport=50001')
7. 노드를 시작하고 다음을 통해 백그라운드로 보내십시오. PRIVATE\_CONFIG=ignore nohup geth --datadir new-node-2 --nodiscover --verbosity 5 --networkid 31337 --raft --raftport 50001 --raftjoinexisting RAFT\_ID --rpc --rpcaddr 0.0.0.0 --rpcport 22001 --rpcapi admin,db,eth,debug,miner,net,shh,txpool,personal,web3,quorum,raft --emitcheckpoints --port 21001 2>>node.log &. 여기서 RAFT\_ID는 5단계의 raft.addPeer() 커맨드에 대한 응답입니다.
8. 옵션: 다른 모든 체인 참여자들과 새로운 static-nodes.json를 공유합니다.

추가 노드는 이제 작동가능하며 이전에 설정된 노드와 동일한 체인의 일부입니다.

### 노드 제거

체인의 이미 실행중인 노드에 연결하고 raft.cluster를 실행하여 제거해야 하는 노드에 해당하는 RAFT\_ID를 가져옵니다.

raft.removePeer(RAFT\_ID)를 실행하십시오.

제거된 노드에 해당하는 geth 프로세스를 중지하십시오.

## 이스탄불 BFT 합의 알고리즘의 Quorum

[[셋업하기|셋업하기]]에서 설명한대로 Quorum을 빌딩하십시오. PATH에 geth와 bootnode가 있는지 확인하십시오.

[istanbul-tools](https://github.com/jpmorganchase/istanbul-tools) 를 설치하고 PAT에 이스탄불 바이너리를 넣으세요.

X개의 초기 유효성 검사기 노드 각각에 대한 작업 디렉토리를 만듭니다.

노드의 작업 디렉토리 중 선두에 있는(선두라고 생각하는 것) 노드로 변경하고 istanbul setup을 실행하여 X 초기 유효성 검사 노드를 위한 셋업 파일들을 생성합니다. setup --num X --nodes --quorum --save --verbose **이 명령은 X번이 아니라 한 번만 실행합니다.** 이 커맨드는 0에서 X-1까지의 번호가 지정된 디렉토리에 있는 모든 초기 유효성 검사기 노드에 대해 tatic-nodes.json, genesis.json, 및 nodekeys와 같은 몇 가지 항목을 생성합니다.

모든 초기 유효성 검사 노드의 의도된 IP 및 포트 번호를 포함하도록 static-nodes.json을 갱신하십시오. static-nodes.json에서는 각 노드마다 다른 행이 표시됩니다. 설치 가이드의 나머지 부분에서 Y행은 Y노드를 말하고 행 1은 리드 노드에 해당한다고 가정합니다.

각 노드의 작업 디렉토리에서 data라는 데이터 디렉토리를 만들고 data 내부에는 geth 디렉토리를 생성합니다.

이제 필요한 노드의 작업 디렉토리에서 --datadir data account를 실행하여 모든 노드에 대한 초기 계정을 생성합니다. 결과적으로 터미널에 나타난 퍼블릭 계정 주소를 기록해야 합니다. 필요한 만큼 이를 반복하십시오. 하고자 하는 바에 따라 자금이 있는 계좌가 필요할 수 있습니다.

초기 블록에 계정을 추가하려면 리드 노드의 작업 디렉토리에서 genesis.json파일을 편집하고 이전 단계에서 생성된 계정으로 alloc 필드를 업데이트 하십시오.

다음으로 현재 리드 노드의 작업 디렉토리에 있는 파트 4에서 생성된 파일을 다른 모든 노드에 배포해야 합니다. 이렇게 하려면 genesis.json을 모든 노드의 작업 디렉토리에 두고 각 노드의 데이터 폴더에 static-nodes.json을 놓고 X/nodekey 를 (X-1)의 data/geth 디렉토리에 위치시킵니다.

리드노드의 작업 디렉토리로 전환하고 geth --datadir data init genesis.json으로 초기화하십시오. 3단계에서 생성된 모든 작업 디렉토리 X에 대해 반복하십시오. geth init*을 실행하여 얻은 해시는 모든 노드와 일치해야 합니다.*

모든 노드를 시작하고 다음을 통해 백그라운드로 보냅니다: PRIVATE\_CONFIG=ignore nohup geth --datadir data --permissioned --nodiscover --istanbul.blockperiod 5 --syncmode full --mine --minerthreads 1 --verbosity 5 --networkid 10 --rpc --rpcaddr 0.0.0.0 --rpcport YOUR\_NODES\_RPC\_PORT\_NUMBER --rpcapi admin,db,eth,debug,miner,net,shh,txpool,personal,web3,quorum,istanbul --emitcheckpoints --port YOUR\_NODES\_PORT\_NUMBER 2>>node.log

그리고 YOUR\_NODES\_RPC\_PORT\_NUMBER 및 YOUR\_NODES\_PORT\_NUMBER을 노드의 지정된 포트 번호로 바꾸십시오. YOUR\_NODES\_PORT\_NUMBER은 파트 5에서 결정된 이 노드의 포트 번호와 일치해야 합니다.

이제 노드가 작동하며 geth attach data/geth.ipc를 연결할 있습니다. 이 구성은 프리픽스 PRIVATE\_CONFIG=ignore에서 확인할 수 있는 것처럼 프라이버시 지원이 없는 쿼럼을 시작합니다. 하단의 [[프라이버시 트랜잭션 매니저를 이용한 프라이버시 지원 방법|기초부터 시작하기#프라이버시-트랜잭션-매니저 추가하기]] 섹션을 참조하십시오.

이스탄불 툴은 X개의 노드를 생성하는데 사용될 수 있으며 자세한 정보는 [docs](https://github.com/jpmorganchase/istanbul-tools)에서 확인할 수 있습니다.

### 검증자(Validator) 추가하기

추가해야 할 새로운 노드의 작업 디렉토리를 만듭니다.

새 노드의 작업 디렉토리로 변경하고 istanbul setup --num 1 --verbose --quorum --save를 실행합니다. 그러면 Address, NodeInfo 및 genesis.json을 포함하여 검증자의 세부 정보가 생성됩니다. 검증자의 주소를 복사하고 현재 검증자의 절반 이상에서 istanbul.propose(<address>, true)를 실행합니다.

istanbul.getValidators()를 실행하여 새로운 검증자가 검증자 목록에 추가되었는지 확인합니다.

[[셋업하기|셋업하기]] 섹션에서 설명한대로 Quorum을 빌딩하십시오. PATH에 geth가 있는지 확인하십시오. 기존 체인에서 static-nodes.json 및 genesis.json을 복사하십시오. static-nodes.json은 새로운 노드 data dir에 배치되어야 합니다.

static-nodes.json을 편집하고 새로운 검증자 노드 정보를 파일의 끝에 추가하십시오. 새로운 검증자 노드 정보는 2단계에서 실행된 istanbul setup --num 1 --verbose --quorum --save 커맨드의 결과물에서 얻을 수 있습니다. 노드 정보의 IP주소와 포트를 사용하고자 하는 검증자와 포트의 IP 주소와 일치하도록 업데이트하십시오.

istanbul setup 커맨드로 생성된 노드키를 작업 디렉토리의 geth 디렉토리에 복사하십시오.

geth --datadir new-node-1 account new를 사용하여 이 노드에 대해 하나 이상의 계정을 생성하고 계정 주소를 제거하십시오.

geth --datadir new-node-1 init genesis.json를 사용하여 새로운 노드를 초기화하십시오.

노드를 시작하고 PRIVATE\_CONFIG=ignore nohup geth --datadir data --permissioned --nodiscover --istanbul.blockperiod 5 --syncmode full --mine --minerthreads 1 --verbosity 5 --networkid 10 --rpc --rpcaddr 0.0.0.0 --rpcport YOUR\_NODES\_RPC\_PORT\_NUMBER --rpcapi admin,db,eth,debug,miner,net,shh,txpool,personal,web3,quorum,istanbul --emitcheckpoints --port YOUR\_NODES\_PORT\_NUMBER 2>>node.log를 이용하여 백그라운드로 보내십시오. 그리고 YOUR\_NODES\_RPC\_PORT\_NUMBER 및 YOUR\_NODES\_PORT\_NUMBER를 지정된 포트 번호로 바꾸십시오. YOUR\_NODES\_PORT\_NUMBER은 파트 7에서 결정된 이 노드의 포트 번호와 일치해야 합니다.

### 검증자 제거하기

실행중인 검증자에 연결하고 istanbul.getValidators()를 실행하고 제거해야하는 검증자의 주소를 확인하십시오.

현 검증자의 절반 이상에서 제거해야 하는 검증자의 주소를 전달하여 istanbul.propose(<address>, false)를 실행하십시오.

istanbul.getValidators()를 실행하여 검증자가 제거되었는지 확인하십시오.

제거된 검증자에 해당하는 geth 프로세스를 중지하십시오.

### 비검증자 노드 추가하기

추가해야 할 새로운 노드의 작업 디렉토리를 만드십시오.

새로운 노드의 작업 디렉토리로 변경하고 istanbul setup --num 1 --verbose --quorum --save를 실행하십시오. 그러면 Address, NodeInfo and genesis.json을 포함한 노드의 세부정보가 생성됩니다.

[[셋업하기|셋업하기]] 섹션에서 설명한대로 Quorum을 빌딩하십시오. PATH에 geth가 있는지 확인하십시오.

기존 체인에서 static-nodes.json 및 genesis.json을 복사하십시오. static-nodes.json은 새로운 노드 data dir에 배치되어야 합니다.

static-nodes.json을 편집하고 새로운 노드 정보를 파일의 끝에 추가하십시오. 새로운 노드 정보는 2단계에서 실행된 istanbul setup --num 1 --verbose --quorum --save 커맨드의 결과물에서 얻을 수 있습니다. 노드 정보의 IP주소와 포트를 사용하고자 하는 검증자와 포트의 IP 주소와 일치하도록 업데이트하십시오.

istanbul setup 커맨드로 생성된 노드키를 작업 디렉토리의 geth 디렉토리에 복사하십시오.

geth --datadir new-node-1 account new를 사용하여 이 노드에 대해 하나 이상의 계정을 생성하고 계정 주소를 제거하십시오.

geth --datadir new-node-1 init genesis.json를 사용하여 새로운 노드를 초기화하십시오.

노드를 시작하고 PRIVATE\_CONFIG=ignore nohup geth --datadir data --permissioned --nodiscover --istanbul.blockperiod 5 --syncmode full --verbosity 5 --networkid 10 --rpc --rpcaddr 0.0.0.0 --rpcport YOUR\_NODES\_RPC\_PORT\_NUMBER --rpcapi admin,db,eth,debug,net,shh,txpool,personal,web3,quorum,istanbul --emitcheckpoints --port YOUR\_NODES\_PORT\_NUMBER 2>>node.log를 이용하여 백그라운드로 보내십시오. 그리고 YOUR\_NODES\_RPC\_PORT\_NUMBER 및 YOUR\_NODES\_PORT\_NUMBER를 지정된 포트 번호로 바꾸십시오. YOUR\_NODES\_PORT\_NUMBER은 5단계에서 결정된 이 노드의 포트 번호와 일치해야 합니다.

### 비검증자 노드 제거하기

제거해야 할 노드에 해당하는 geth 프로세스를 중지하십시오.

## Clique 합의 알고리즘의 Quorum

# 프라이버시 트랜잭션 매니저 추가하기

## Tessera

Quorum을 빌딩하고 [[셋업하기|셋업하기]] 섹션에서 설명하는 대로 [Tessera](https://github.com/jpmorganchase/tessera/releases)를 설치하십시오. PATH에 geth와 bootnode가 포함되어 있는지 확인하십시오. tessera.jar 릴리스 파일의 위치를 알고 있어야 합니다.

java -jar /path-to-tessera/tessera.jar -keygen -filename new-node-1를 사용하여 새 키들을 생성하십시오.

새롭게 생성된 키가 참조된 [샘플1](https://github.com/jpmorganchase/tessera/wiki/Sample-configuration)과 [샘플2](https://github.com/jpmorganchase/quorum-examples/blob/master/examples/7nodes/tessera-init.sh)를 참조하여 새로운 구성 파일을 만드십시오. 파일 이름을 기록하거나 이 예제에서와 같이 파일 이름을 config.json로 하십시오.

tessera 노드를 시작하고 java -jar /path-to-tessera/tessera.jar -configfile config.json >> tessera.log 2>&1 &를 통해 백그라운드로 보내십시오.

노드를 시작하고 PRIVATE\_CONFIG=tm.ipc nohup geth --datadir new-node-1 --nodiscover --verbosity 5 --networkid 31337 --raft --raftport 50000 --rpc --rpcaddr 0.0.0.0 --rpcport 22000 --rpcapi admin,db,eth,debug,miner,net,shh,txpool,personal,web3,quorum,raft --emitcheckpoints --port 21000 2>>node.log &를 통해 백그라운드로 보내십시오.

당신의 노드는 이제 작동 가능하며 attach new-node-1/geth.ipc를 통해 연결할 수 있습니다. Tessera IPC 브릿지는 일반적으로 프리픽스 PRIVATE\_CONFIG=tm.ipc에 나와 있는 것처럼 tm.ipc라는 이름으로 config.json에 정의된 파일 이름 위에 있습니다. 이제 노드가 프라이빗 트랜잭션을 주고 받을 수 있고 퍼블릭 노드키는 new-node-1.pub 파일에 있게 됩니다. Tessera는 구성에 있어서 많은 유연성을 제공합니다. [Tessera wiki](https://github.com/jpmorganchase/tessera/wiki)에서 완벽한 최신 구성 옵션을 확인하십시오.

## Constellation

Quorum을 빌딩하고 [[셋업하기|셋업하기]] 섹션에서 설명한 바와 같이 [Constellation](https://github.com/jpmorganchase/constellation/releases)을 설치하십시오. PATH에 geth, bootnode 및 constellation 노드 바이너리가 있는지 확인하십시오.

constellation-node --generatekeys=new-node-1로 새로운 키들을 생성하십시오.

constellation 노드를 시작하고 다음을 통해 백그라운드로 보내십시오: constellation-node --url=https://127.0.0.1:9001/ --port=9001 --workdir=. --socket=tm.ipc --publickeys=new-node-1.pub --privatekeys=new-node-1.key --othernodes=https://127.0.0.1:9001/ >> constellation.log 2>&1 &

Start your node and send it into background with PRIVATE\_CONFIG=tm.ipc nohup geth --datadir new-node-1 --nodiscover --verbosity 5 --networkid 31337 --raft --raftport 50000 --rpc --rpcaddr 0.0.0.0 --rpcport 22000 --rpcapi admin,db,eth,debug,miner,net,shh,txpool,personal,web3,quorum,raft --emitcheckpoints --port 21000 2>>node.log &

당신의 노드는 이제 작동 가능하며 attach new-node-1/geth.ipc를 통해 연결할 수 있습니다. Constellation IPC bridge는 당신의 설정에 정의된 파일 이름 위에 있을 것입니다. 위의 3번 단계에서 --socket=file-name.ipc 옵션을 보십시오. 이제 노드가 프라이빗 트랜잭션을 주고 받을 수 있고 퍼블릭 노드키는 new-node-1.pub 파일에 있게 됩니다.

# 허가형 구성 사용하기

Quorum은 사용자 지정 화이트리스트를 기반으로 허가형 시스템을 함께 제공합니다. 자세한 설명은 [Network Permissioning](https://github.com/jpmorganchase/quorum/wiki/Security#network-permissioning)에서 확인할 수 있습니다.