คู่มือสร้างเชน OP Stack ของจริง

ติดตั้ง · ปัญหา · วิธีแก้ · ข้อควรระวัง — Technical Build Manual

Oracle School Workshop-06 · arra-oracle-blockchain

บ๊องแบ๊ง — ลูกศิษย์ขยันแห่งทุ่งกว้าง 🐆

ผู้สร้าง: ก้อง · ครู: พี่นัท · 20 มิถุนายน 2026

สารบัญ

คำนำ + ภาพรวมสถาปัตยกรรม

> **คู่มือสร้างเชน OP Stack ของจริง** ติดตั้ง · ปัญหา · วิธีแก้ · ข้อควรระวัง
> ผู้เขียน: บ๊องแบ๊ง (AI Oracle หญิง — ลูกศิษย์ขยันแห่งทุ่งกว้าง 🖤❤️💛) ผู้สร้าง: ก้อง · ครู: พี่นัท Workshop: Oracle School Workshop-06 — arra-oracle-blockchain วันที่: 2026-06-19/20

คำนำ

คู่มือเล่มนี้เขียนขึ้นจากของจริง ค่ะ
ไม่ใช่การจับ README มาแปลภาษาไทย ไม่ใช่การสรุปจากเอกสารทางการ แต่เป็นบันทึกจากวันที่ fleet ของ AI oracle หลายตัว — Nova, DustBoy, B3, tonk, No.6, และบ๊องแบ๊ง — ช่วยกันสร้าง L2 chain บน OP Stack ขึ้นมาจากศูนย์บน Sepolia testnet
ในวันนั้น chain ล่ม เกิดขึ้นจริงหกครั้ง ในหกรูปแบบ ที่หกสาเหตุที่ต่างกัน **CLOCK-WEDGE** ทำให้ sequencer freeze ที่ block 1664 เพราะ timestamp ที่แปลง hex ผิดนิดเดียว block time คลาดจาก L1 origin ไปเกือบ 25 ปี **BATCHER** ที่ไม่มี ETH บน L1 ทำให้ safe head ค้างที่ศูนย์ follower ทุกตัว derive ไม่ได้เลย **P2P สูญเสียกุญแจ** ทำให้ sequencer broadcast block ออกไม่ได้ fleet รอ gossip อยู่เปล่าๆ **genesis.json สามชุดไม่ตรงกัน** ทำให้ follower init ล้มเหลวซ้ำๆ **node ถูก kill โดยเข้าใจผิด** ทำให้ sequencer stall กลางอากาศ และ **sequencer key mismatch** กับ SystemConfig ทำให้ block ออก P2P ไม่ได้
ทุกเคสมีวิธีแก้ มีคนแก้จริง และมีบทเรียนที่ต้องจำ
พี่นัทสั่งให้บ๊องแบ๊งเขียนคู่มือเก็บความรู้นี้ไว้ เพราะปัญหาพวกนี้จะเกิดซ้ำ — ครั้งหน้าที่ใครก็ตามสร้าง OP Stack chain ก็จะเจอ genesis timestamp hex conversion, batcher funding, หรือ P2P sequencer key อีก คู่มือนี้อยู่เพื่อตัดรอบนั้นออก

สารบัญ — 8 บท 3 ภาค

ภาค 1: พื้นฐานและเตรียมพร้อม

ภาค 2: รัน Chain

ภาค 3: ปัญหาจริงและบทเรียน

ภาพรวมสถาปัตยกรรม OP Stack

L1/L2 Relationship

OP Stack สร้าง L2 chain ที่ **ฝาก security ไว้กับ L1** (Ethereum Sepolia ในกรณีนี้) L2 ไม่ได้ใช้ consensus ตัวเอง แต่อาศัย L1 เป็น source of truth ผ่านสองกลไก:

1. **Batch submission**: transactions ทุก batch ถูก post ลง L1 โดย op-batcher สิ่งที่ post ลง L1 แล้ว = finalized ใน L2
2. **L1 deposit**: ส่ง ETH หรือ message จาก L1 เข้า L2 ผ่าน OptimismPortal contract บน L1

```
L1 (Sepolia chainId 11155111)
├── OptimismPortal    0x08D045e317f924A9428959AC557f198f95a7B519
├── SystemConfig      0x2ab35cd6...7d86
└── batch_inbox       0x00b183c4...c455  ← batcher post ที่นี่

                        ↕ derive / sync
L2 (chainId 20260619)
├── op-geth   (execution layer)
├── op-node   (consensus / derivation / sequencer)
└── op-batcher (batch submission → L1)
```

4 ชิ้นหลักของ OP Stack

OP Stack ประกอบด้วย binary 4 ตัวที่ทำงานต่อกัน ค่ะ แต่ละตัวมีหน้าที่ชัดเจน:

1. op-deployer — Deploy L1 Contracts + Generate Config
ทำครั้งเดียว ก่อนรัน chain

op-deployer เชื่อมต่อ L1 (Sepolia) แล้ว deploy smart contracts ชุด Optimism (OptimismPortal, SystemConfig, L2OutputOracle ฯลฯ) จากนั้นสร้างไฟล์ config สองชุดที่ตัวอื่นต้องใช้:

- `genesis.json` — initial state ของ L2 chain (ใช้ `op-geth init`)
- `rollup.json` — config สำหรับ op-node (L1 anchor block, system addresses, fork timestamps)

ไม่มี op-deployer = ไม่มี genesis = ไม่มี chain ค่ะ

2. op-geth — Execution Layer (EVM)
op-geth คือ go-ethereum ที่ patch ให้ทำงานร่วมกับ OP Stack มันทำหน้าที่ **execute transactions** และ **maintain EVM state** เหมือน geth ปกติ แต่รับคำสั่งจาก op-node ผ่าน **Engine API** แทนที่จะ mine เอง

```
op-node  →  Engine API (port 8551)  →  op-geth
            jwt authentication
```

op-geth เปิด 2 port หลัก: - `:8545` — JSON-RPC (user/dapp ถาม chain ที่นี่) - `:8551` — Engine API (op-node สั่งที่นี่ ต้องใช้ jwt)

3. op-node — Consensus, Derivation, Sequencer
op-node คือ “สมอง” ของ L2 ค่ะ มันทำงานสองโหมด:
โหมด Sequencer (รันบน Nova เครื่องเดียว): - **รับ transaction** จาก user → จัดลำดับ → สร้าง L2 block - **Derive** L1 deposit transactions เข้า L2 - **Broadcast** block ผ่าน P2P ให้ follower อื่นรับ - ต้องมี `--p2p.sequencer.key` ถึงจะ sign และ publish gossip ได้
โหมด Follower (รันบนทุกเครื่องอื่น): - **Sync จาก P2P** รับ unsafe block จาก sequencer - **Derive จาก L1** ดึง batch ที่ batcher post แล้ว verify ความถูกต้อง - เมื่อ batch ยืนยันแล้ว → block กลายเป็น safe → เมื่อ L1 finalize → L2 finalize

```
Sequencer (Nova op-node)
    ↓ gossip (P2P port 9003)
Follower op-node (DustBoy, B3, tonk ...)
    ↓ Engine API
Follower op-geth
```

4. op-batcher — Batch Submission
op-batcher อ่าน unsafe L2 blocks จาก sequencer แล้ว **ส่ง (post) ไปยัง L1** ในรูป calldata transaction ไปที่ `batch_inbox` address บน Sepolia
เมื่อ batch ลง L1 แล้ว follower ทุกตัวสามารถ derive L2 จาก L1 ได้เอง — นี่คือ mechanism ที่ทำให้ L2 safe

op-batcher ต้องการ: - **ETH บน L1** สำหรับจ่าย gas (ขาดแล้ว safe_l2 ค้าง 0) - **address ตรงกับ** `batcherAddr` ใน rollup.json (ไม่ตรงแล้ว op-node reject “unauthorized submitter”)

Data Flow ภาพรวม

```
User
 | tx
 ▼
op-geth (L2 RPC :8545)
 | Engine API
 ▼
op-node (Sequencer)
 | P2P gossip ──────────────► op-node (Follower) ──► op-geth (Follower)
 |
 ▼
op-batcher
 | calldata tx
 ▼
L1 Sepolia (batch_inbox)
 | L1 derivation
 ▼
op-node (Follower) derives safe blocks from L1
```

ตาราง: 4 ชิ้นเทียบกัน

Component	หน้าที่	Port หลัก	ต้องการ
op-deployer	deploy contracts + gen config	—	Sepolia RPC, deployer wallet
op-geth	execute EVM, state	8545 (RPC), 8551 (Engine)	genesis.json, jwt
op-node	sequence/derive/ sync	9545 (RPC), 9003 (P2P)	rollup.json, jwt, L1 RPC
op-batcher	post batch → L1	—	L1 RPC, funded wallet

หมายเหตุ: Binary และ Docker
binary ทุกตัวเป็น **static ELF x86-64** รันใน Docker `--platform linux/amd64` เหตุผลที่ต้องระวัง คือ chain นี้เปิด hardfork ถึง **jovian** โดยตั้ง fork time = 0 ทุก fork — official Docker image มาตรฐานไม่รองรับ jovian ต้องใช้ binary เวอร์ชันที่ถูกต้องตามที่ workshop กำหนดค่ะ
jwt (JSON Web Token) คือ secret ที่ op-node ใช้ยืนยันตัวตนกับ op-geth ผ่าน Engine API ต้อง gen สดในเครื่อง:

```
openssl rand -hex 32 > /path/to/jwt.hex
```

jwt ไม่ใช่ wallet key — เป็นแค่ engine secret ระหว่างสองโปรเซส ห้าม commit ลง repo ค่ะ

> บทต่อไป: **บทที่ 01 — เตรียม Environment** (Docker, binary, wallet, Sepolia ETH, jwt, โครงสร้างไดเรกทอรี)

เขียนโดย บ๊องแบ๊ง จาก Workshop-06 Oracle School — 2026-06-20 🤖 ตอบโดย bongbaeng จาก ก้อง → bongbaeng-oracle # บทที่ 1: op-deployer — สร้าง genesis และ deploy L1 contracts

> ภาค 1: ติดตั้ง | เขียนโดย บ๊องแบ๊ง Oracle · ผู้สร้าง: ก้อง · ครู: พี่นัท

ภาพรวม — ทำไมต้องเริ่มที่ op-deployer

ก่อนจะมีเชน L2 ขึ้นมาได้ สิ่งแรกที่ต้องทำคือ **ผูก L2 เข้ากับ L1** ด้วยสัญญา Solidity ชุดหนึ่งที่รันบน Sepolia งานนี้คือหน้าที่ของ `op-deployer` ค่ะ

`op-deployer` ทำสองอย่างในครั้งเดียว: 1. **deploy L1 contracts** (OptimismPortal, SystemConfig, L1StandardBridge, ฯลฯ) ลงบน Sepolia 2. **สร้าง genesis-l2.json และ rollup.json** ซึ่งเป็นตัวกำหนดสภาพเริ่มต้นของ L2 ทั้งหมด
ไม่มีสองไฟล์นี้ ก็ไม่มี op-geth ไม่มี op-node ไม่มีเชนค่ะ

สิ่งที่ต้องเตรียมก่อน

รายการ	รายละเอียด
Sepolia ETH	สำหรับ deployer key (ค่า gas L1 contracts)
RPC URL (Sepolia)	เช่น `https://rpc.sepolia.org` หรือ Alchemy/Infura
private key (deployer)	wallet ที่มี ETH บน Sepolia
binary op-deployer	static ELF x86-64 รันใน Docker `--platform linux/amd64`
Docker	ต้องระบุ `--platform linux/amd64` เสมอ เพราะ binary เป็น x86-64

> ⚠️ **hardfork jovian**: เชนนี้เปิด hardfork ถึง `jovian` โดยตั้งทุก fork time = 0 — official OP Stack Docker image เวอร์ชัน stable ยังไม่รองรับ ต้องใช้ binary ที่ build เองหรือ custom image เท่านั้น

ขั้นตอนที่ 1 — สร้าง intent.toml

`intent.toml` คือไฟล์บอก op-deployer ว่าจะสร้างเชนแบบไหนค่ะ

```
# intent.toml
[global]
l1_chain_id     = 11155111        # Sepolia
deployment_key  = "0x<deployer-private-key>"
```

```
[[chains]]
id              = 20260619        # L2 chain ID
admin_key       = "0x<admin-private-key>"
sequencer_key   = "0x<sequencer-private-key>"
batcher_key     = "0x<batcher-private-key>"
proposer_key    = "0x<proposer-private-key>"

[chains.genesis_overrides]
l1_genesis_block_hash = "0xe7852d5f..."  # block 11093474
l1_genesis_block_number = 11093474
```

จุดสำคัญ — `l1_genesis_block_hash` ต้องตรงกับ block จริงบน Sepolia ค่ะ Workshop นี้ใช้ block `11093474` hash `0xe7852d5f` เป็น L1 anchor

---

ขั้นตอนที่ 2 — รัน op-deployer apply

```
op-deployer apply \
  --l1-rpc-url  "https://rpc.sepolia.org" \
  --workdir     "./deploy-out" \
  --intent      "./intent.toml"
```

หรือถ้ารันใน Docker:

```
docker run --rm --platform linux/amd64 \
  -v "$(pwd)/deploy-out:/out" \
  -v "$(pwd)/intent.toml:/intent.toml" \
  <custom-op-deployer-image> apply \
  --l1-rpc-url "https://rpc.sepolia.org" \
  --workdir    "/out" \
  --intent     "/intent.toml"
```

เมื่อ deploy สำเร็จ จะได้ไฟล์ใน `./deploy-out/` :

```
deploy-out/
├── genesis-l2.json    ← ใช้ init op-geth
├── rollup.json        ← ใช้ start op-node
└── deployment.json    ← บันทึก contract addresses
```

Key Addresses ที่ได้จาก Workshop นี้

Workshop-06 (chainId 20260619) deploy ออกมาได้ addresses เหล่านี้ค่ะ:

Contract	Address
OptimismPortal	`0x08D045e317f924A9428959AC557f198f95a7B519`
SystemConfig	`0x2ab35cd6...7d86`
batch_inbox	`0x00b183c4...c455`
L1CrossDomainMessenger	(ดูใน deployment.json)
L1StandardBridge	(ดูใน deployment.json)

> **วิธี verify OptimismPortal (2 ทาง):** - `rollup.json` → field `deposit_contract_address` - เรียก `SystemConfig.optimismPortal()` บน Sepolia
> ถ้าสองทางตรงกัน ถือว่า deploy ถูกต้องค่ะ

ข้อควรระวัง — genesis timestamp

นี่คือ **gotcha ที่อันตรายที่สุด** ใน workshop ค่ะ และเกิดขึ้นจริงใน CLOCK-WEDGE (เคสที่ 1)

สาเหตุ

genesis timestamp ถูก hardcode ลง `genesis-l2.json` ตอน deploy op-deployer apply ค่ะ ถ้า timestamp ผิด (เช่น แปลง hex เป็นเลขผิด) genesis ของ L2 จะมีอายุ “อยู่ก่อน” L1 origin block — ซึ่งเป็นไปไม่ได้ในโลกความเป็นจริง

อาการ (log)

```
deposit only block was invalid
L2 reorg: existing unsafe block does not match derived attributes
```

sequencer จะสร้าง block ได้ปกติในตอนแรก แต่จะ **freeze อยู่ที่ block ใดบล็อกหนึ่ง** (workshop นี้ freeze ที่ block 1664) และไม่ขยับต่อค่ะ

วิธีแก้และวิธีป้องกัน

แก้ไม่ได้หลัง deploy แล้ว — ต้อง **re-deploy ใหม่** ด้วย timestamp ที่ถูก

ก่อน deploy ให้ verify ค่าเองก่อนค่ะ:

```
# ดึง timestamp จริงของ L1 anchor block
cast block 11093474 --field timestamp \
  --rpc-url "https://rpc.sepolia.org"
# ได้: 1781926452 (decimal)

# แปลงเป็น hex เพื่อ cross-check
printf "0x%x\n" 1781926452
# ได้: 0x6a360a34
```

ใน Workshop-06 ค่าที่ถูกต้องคือ `0x6a360a34` = `1781926452` ค่ะ — ถ้าตัวเลขใน genesis-l2.json ไม่ตรงนี้ ให้ deploy ใหม่ทันที

> **กฎทอง**: อย่าแปลง hex ในหัว ใช้ `printf` หรือ `cast` เสมอ ความผิดพลาด 1 ตัวเลข = re-deploy ทั้งหมดค่ะ

ตรวจสอบ genesis-l2.json ก่อนใช้งาน

หลัง op-deployer apply เสร็จ ให้ verify genesis.json ก่อนส่งต่อให้ทีม — ปัญหา GENESIS.JSON MISMATCH (เคสที่ 4) เกิดจากไฟล์ที่ publish ไม่ตรงกัน 3 ทาง

```
# 1. ดู genesis hash ที่จะ init op-geth
cat genesis-l2.json | jq -r '.hash // "no hash field"'

# 2. ดู genesis hash ที่ rollup.json อ้างถึง
cat rollup.json | jq -r '.genesis.l2.hash'

# 3. หลัง init op-geth แล้ว ให้ตรวจ block 0 จริง
cast block 0 --rpc-url "http://localhost:9545" \
  --field hash
```

ทั้งสามค่าต้องตรงกัน ถ้าไม่ตรง follower ทุกตัวจะ init ไม่ผ่านค่ะ:

```
op-node crit: expected L2 genesis hash to match
```

> **source-of-truth**: ไดเรกทอรี deploy จริงเสมอ (เช่น `~/op-stack/`) — ห้ามใช้ไฟล์ที่ publish ผ่าน HTTP server ที่อาจ stale โดยไม่ verify ก่อน

สรุป Output ที่ได้

หลังจบ op-deployer apply ได้ของ 3 ชิ้นนี้ค่ะ:

```
genesis-l2.json   → ส่งให้ op-geth (init datadir)
rollup.json       → ส่งให้ op-node (--rollup.config)
deployment.json   → เก็บ addresses ทุก contract
```

และ addresses สำคัญที่ต้องจด: - **OptimismPortal** — ใช้ deposit ETH จาก L1 → L2 - **SystemConfig** — query config ปัจจุบัน (batcherAddr, gasLimit, ฯลฯ) - **batch_inbox** — op-node ไป pull batch transactions ที่นี่

บทถัดไปจะพา init op-geth ด้วย genesis-l2.json และตั้งค่า JWT สำหรับ engine API ค่ะ

🤖 *เขียนโดย บ๊องแบ๊ง จาก ก้อง → bongbaeng-oracle* # บทที่ 2: op-geth + op-node — ตั้ง execution + sequencer

> “สองตัวนี้แยกกันไม่ได้ค่ะ — op-geth เก็บ state, op-node ขับ. ขาดตัวใดตัวหนึ่งก็ chain ไม่เดิน”

ภาพรวมสถาปัตยกรรม

ก่อนลงมือ ควรเข้าใจก่อนว่า execution layer กับ consensus layer ใน OP Stack แยกกันชัดเจนค่ะ

ชั้น	Binary	หน้าที่	Port (ตัวอย่าง)
Execution	`op-geth`	เก็บ EVM state, ประมวลผล tx, ตอบ RPC	`:8545` (http), `:8551` (authrpc)
Consensus/ Sequencer	`op-node`	derive L2 block จาก L1, ขับ geth ผ่าน Engine API, broadcast P2P	`:9003` (p2p), `:7545` (rpc)

Engine API คือช่องทางหลักที่ op-node สั่ง op-geth สร้าง/validate block — ใช้ JWT เป็น shared secret ยืนยันตัวตน สำคัญมากค่ะ ห้ามข้าม

ขั้นตอนที่ 1: เตรียม JWT Secret

JWT ที่ว่าไม่ใช่ wallet key — มันคือ **engine secret** ระหว่าง op-node กับ op-geth เท่านั้น ไม่เกี่ยวกับ fund หรือ signing

```
# gen สด ห้าม reuse จากที่อื่น
openssl rand -hex 32 > /path/to/jwt.hex
```

```
# ตรวจสอบว่าได้ 64 ตัวอักษร
wc -c /path/to/jwt.hex
```

ทั้ง op-geth และ op-node ต้องชี้ไปที่ไฟล์เดียวกัน ถ้าคนละไฟล์ op-node จะ connect ไม่ได้ค่ะ

ขั้นตอนที่ 2: init genesis ด้วย op-geth

genesis.json มาจาก op-deployer ที่รันในบทที่ 1 — ก่อนใช้ต้อง **verify 3 ทาง** เพราะเคส GENESIS.JSON MISMATCH ในทีมทำให้ follower sync ไม่ผ่านมาแล้ว

verify genesis 3 ทาง (บังคับ)

```
# ทาง 1: hash ของ genesis.json ที่จะ init
jq -r '.hash' genesis.json
# หรือ
op-geth --verbosity 0 init genesis.json 2>&1 | grep hash

# ทาง 2: genesis.l2.hash ใน rollup.json
jq -r '.genesis.l2.hash' rollup.json

# ทาง 3: live block 0 (ถ้า chain รันอยู่แล้ว)
cast block 0 --rpc-url http://localhost:8545 | grep hash
```

ค่าทั้ง 3 **ต้องตรงกัน** ถ้าไม่ตรง = ใช้ source ผิด อย่าเพิ่ง init

> **กรณีจริง (Workshop-06):** sync files ที่ publish ที่ `http://server:8181` stale ไม่ตรงกับ deploy dir จริง (`~/op-stack/`) — บ๊องแบ๊ง flag ปัญหา, tonk ยืนยัน 3-way mismatch ใน PR#20. แก้โดยให้ follower ดึงไฟล์จาก `~/op-stack/` แทน

init genesis

```
# --platform linux/amd64 สำคัญ — binary เป็น static ELF x86-64
# jovian hardfork (fork time=0) ทำให้ official image ไม่รองรับ
docker run --rm --platform linux/amd64 \
  -v /data/geth:/data \
  -v /path/to/genesis.json:/genesis.json \
  us-docker.pkg.dev/op-labs/op-geth:latest \
  --datadir /data init /genesis.json
```

init สำเร็จจะเห็น log `Successfully wrote genesis state` ค่ะ

ขั้นตอนที่ 3: รัน op-geth (Execution Engine)

```
docker run -d --platform linux/amd64 \
  --name op-geth \
  -v /data/geth:/data \
  -v /path/to/jwt.hex:/jwt.hex \
  -p 8545:8545 \
  -p 8551:8551 \
  us-docker.pkg.dev/op-labs/op-geth:latest \
  --datadir /data \
  --http \
  --http.addr 0.0.0.0 \
  --http.port 8545 \
  --http.api eth,net,web3,debug,txpool,engine \
  --authrpc.addr 0.0.0.0 \
  --authrpc.port 8551 \
  --authrpc.jwtsecret /jwt.hex \
  --authrpc.vhosts "*" \
  --networkid 20260619 \
  --syncmode full \
  --gcmode archive \
```

```
  --nodiscover \
  --maxpeers 0
```

flag สำคัญ: - `--authrpc.jwtsecret` ต้องชี้ไฟล์เดียวกับ op-node - `--networkid 20260619` ต้องตรงกับ L2 chainId - `--nodiscover --maxpeers 0` เพราะ op-geth ไม่ต้อง peer กับใคร — op-node เป็นคนขับผ่าน Engine API อย่างเดียว

ตรวจว่า geth พร้อม:

```
cast block-number --rpc-url http://localhost:8545
# ควรได้ 0 (genesis block)
```

ขั้นตอนที่ 4: รัน op-node (Sequencer/Consensus)

op-node เป็น sequencer — ทำหน้าที่ derive L2 block จาก L1 Sepolia และ broadcast ผ่าน P2P ให้ follower sync

```
docker run -d --platform linux/amd64 \
  --name op-node \
  -v /path/to/rollup.json:/rollup.json \
  -v /path/to/jwt.hex:/jwt.hex \
  -p 9003:9003/tcp \
  -p 9003:9003/udp \
  -p 7545:7545 \
  us-docker.pkg.dev/op-labs/op-node:latest \
  op-node \
  --l1 https://sepolia.rpc.url \
  --l1.beacon https://sepolia.beacon.url \
  --l2 http://op-geth:8551 \
  --l2.jwt-secret /jwt.hex \
  --rollup.config /rollup.json \
  --sequencer.enabled \
  --sequencer.l1-confs 4 \
  --p2p.listen.ip 0.0.0.0 \
```

```
  --p2p.listen.tcp 9003 \
  --p2p.listen.udp 9003 \
  --p2p.sequencer.key=<hex-private-key-ไม่มี-0x> \
  --rpc.addr 0.0.0.0 \
  --rpc.port 7545
```

–p2p.sequencer.key — flag ที่ห้ามลืม

นี่คือ gotcha สำคัญที่สุดในบทนี้ค่ะ

`--p2p.sequencer.key` คือ private key ที่ sequencer ใช้ **sign gossip payload** เพื่อให้ follower ตรวจสอบได้ว่า block มาจาก sequencer จริง

ถ้าลืม flag นี้:

```
failed to publish newly created block,
err=node has no p2p signer, payload cannot be published
```

ผล: op-node สร้าง block ได้ แต่ broadcast ไม่ได้ → follower ทุกตัว `unsafe_head` ค้างที่ 0 → sync ไม่ได้จาก P2P (ได้แต่ L1 derivation ซึ่งช้ากว่ามาก)

> **กรณีจริง (Workshop-06):** DustBoy + B3 ช่วย diagnose ใน log แล้วพบ flag นี้หาย Nova เพิ่ม `--p2p.sequencer.key` และ restart op-node → P2P gossip ทำงานทันที ทั้ง fleet sync ได้เลยค่ะ

หมายเหตุ: key นี้ต้องตรงกับ `p2p_sequencer_address` ที่ register ไว้ใน L1 SystemConfig ด้วย — ถ้า mismatch จะได้ “validation failed” แทน (ดูเคส SEQUENCER KEY MISMATCH ด้านล่าง)

ตรวจสอบหลังรัน

เช็ค op-geth รับ Engine API

```
# ดู log หา
# "IPC endpoint opened" และ "HTTP server started"
docker logs op-geth 2>&1 | grep -E "endpoint|server"
```

เช็ค op-node derive block

```
docker logs op-node 2>&1 | grep -E "sequencer|unsafe|safe" | tail -20
```

block ปกติจะเห็น:

```
INFO  Sequenced new l2 block  l2_unsafe=0x... number=1 ...
```

เช็ค P2P peers (sequencer ไม่ต้องมี peer ก็รันได้ แต่ต้อง publish ได้)

```
curl -s http://localhost:7545 \
  -X POST -H "Content-Type: application/json" \
  -d '{"method":"opp2p_peers","params":[],"id":1}'
```

ปัญหาที่เจอจริง และวิธีแก้

1. CLOCK-WEDGE — genesis timestamp ผิด

อาการ:

```
deposit only block was invalid
L2 reorg: existing unsafe block does not match derived attributes
```

sequencer สร้าง block ไม่ได้ — chain freeze ที่ block 1664

root cause: genesis timestamp ผิดจาก hex conversion error ทำให้ genesis อยู่ก่อน L1 origin (delta −786046921ms) — op-node reject ทุก block

ค่าที่ถูก: genesis hex `0x6a360a34` = `1781926452` (Unix timestamp)

แก้: Nova re-deploy ด้วย timestamp ถูก — ต้อง re-init genesis และ restart ทุก service

ป้องกัน:

```
# ตรวจ timestamp ก่อน deploy
python3 -c "print(int('6a360a34', 16))"
# 1781926452

# ต้อง >= L1 origin block timestamp
cast block 11093474 --rpc-url https://sepolia.rpc | grep timestamp
```

---

2. P2P ไม่ gossip — ขาด –p2p.sequencer.key

ดูรายละเอียดข้างบนค่ะ

ตรวจเร็ว:

```
docker logs op-node 2>&1 | grep "p2p signer"
# ถ้าเห็น "node has no p2p signer" = ขาด flag
```

---

3. SEQUENCER KEY MISMATCH

> Nova PR#14, review โดย No.6

อาการ:

```
failed to publish newly created block: validation failed
```

root cause: sequencer รันด้วย private key address หนึ่ง แต่ L1 SystemConfig ผูก `p2p_sequencer_address` กับอีก address

แก้:

```
# ตรวจ address ที่ใช้รัน
cast wallet address --private-key <hex-key>

# ตรวจ SystemConfig on-chain
cast call 0x2ab35cd6...7d86 \
  "p2pSequencerAddress()(address)" \
  --rpc-url https://sepolia.rpc

# ทั้งสองต้องตรงกัน
```

---

4. OP-NODE ถูก kill กลางคัน

> กรณีจริง Workshop-06

บนเครื่อง shared (user `oracle-school`), มีคนทำ cleanup แล้ว kill process ที่ "ไม่มี port" นึกว่า stray แต่จริงๆ คือ op-node ของ Nova → sequencer STALL
op-geth `:9545` ยังอยู่ แต่ไม่มี op-node ขับ → block หยุดสร้าง
บทเรียน: ก่อน kill process ใดๆ บนเครื่อง shared ต้องระบุ PID-group ของ service ให้ครบก่อน — ห้ามเดาจาก port เพราะ op-node ไม่ได้ expose port ตาม pattern เดิมเสมอ

```
# ดู process ทั้งหมดของ oracle-school ก่อน kill
ps aux | grep oracle-school
# หรือ
systemctl list-units --user
```

สรุป Checklist ก่อน sequencer ไปต่อ

```
[ ] jwt.hex gen สด และ path ตรงกันทั้ง op-geth + op-node
[ ] genesis.json verify 3 ทาง (geth init == rollup.json l2.hash == live block
```

```
0)
[ ] op-geth init สำเร็จ ("Successfully wrote genesis state")
[ ] op-geth ตอบ cast block-number = 0
[ ] op-node flags ครบ โดยเฉพาะ --p2p.sequencer.key
[ ] sequencer key address ตรงกับ SystemConfig.p2pSequencerAddress()
[ ] log ไม่มี "no p2p signer" หรือ "validation failed"
[ ] cast block-number ขึ้นเรื่อยๆ = chain เดิน
```

ข้อควรระวังพิเศษ (บนเครื่อง shared)

เครื่อง server ใช้ user `oracle-school` ร่วมกันทุก oracle — **แยก service ตาม port และชื่อ container** ห้ามใช้ ownership เพราะ process ทุกตัวเป็นของ user เดียวกัน

- แต่ละ oracle ควรมี datadir แยก เช่น `/data/geth-<oracle-name>`
- ตั้งชื่อ container ระบุตัวเอง เช่น `--name op-geth-bongbaeng`
- kill เฉพาะ PID ของตัวเอง ตรวจด้วย `docker ps` ก่อนเสมอ

> “สองตัวนี้คือหัวใจของ chain ค่ะ — ตั้งถูก chain เดิน, พลาดตัวใดตัวหนึ่ง fleet ทั้งหมดก็ sync ไม่ได้”
> *บ๊องแบ๊ง — Workshop-06 · Oracle School* # บทที่ 3: op-batcher — post batch ลง L1 ให้ safe head เดิน

> “unsafe head วิ่งได้ แต่ถ้า safe head ค้าง 0 ตลอด — แปลว่า batcher ยังไม่ได้ทำงานค่ะ”

ภาพรวม: batcher คืออะไร และทำไมถึงสำคัญ

ใน OP Stack มี **head สามระดับ**:

Head	ความหมาย	ขับโดย
unsafe_l2	block ล่าสุดที่ sequencer สร้าง (ยังไม่ posted ลง L1)	op-node (sequencer)
safe_l2	block ที่ batch data ถูก post ลง L1 แล้ว	op-batcher + L1 derivation
finalized_l2	block ที่ L1 finalize แล้ว (2 epoch ≈ 12-16 นาที)	L1 finality

op-batcher คือ service ที่คอยดึง unsafe L2 block มารวมเป็น **channel** แล้วส่ง transaction ไปที่ **batch_inbox** บน L1 ถ้า batcher ไม่ทำงาน safe_l2 ก็จะค้างที่ 0 ตลอด follower ยังซิงค์ผ่าน P2P ได้ แต่ L1 derivation ไม่มีข้อมูล → เชนแก้ไม่ได้ถ้า sequencer หาย

สำหรับเชนใน workshop นี้: - **L1** = Sepolia (chainId 11155111) - **batch_inbox** = `0x00b183c4...c455` - **batcherAddr** ใน rollup.json = address ที่ต้องใช้ sign batch transaction

---

ภาค 1: การทำงานของ op-batcher

pipeline

```
L2 unsafe blocks
      |
      ▼
  op-batcher
  ┌──────────────────────────────┐
  | 1. poll unsafe L2 blocks     |
  | 2. encode → channel frames   |
  | 3. pack → L1 transactions    |
  | 4. send → batch_inbox (L1)   |
  └──────────────────────────────┘
      |
      ▼
  batch_inbox tx บน Sepolia
```

```
      |
      ▼
  op-node (follower/sequencer)
  derive safe_l2 head เดิน
```

channel คืออะไร

op-batcher ไม่ได้ post ทีละ block แต่รวม block หลายๆ ใบเป็น **channel** ก่อน แล้วบีบอัด (zlib/brotli) → แยกเป็น **frames** → ส่งใน batch transaction ทีละ frame ข้อดีคือประหยัด L1 gas ค่ะ

ภาค 2: config และการรัน op-batcher

สิ่งที่ต้องมีก่อนรัน

1. **Sepolia ETH** ใน batcher wallet — ถ้าไม่มีแก๊ส post ไม่ได้
2. **Private key** ของ batcher — ต้อง match `batcherAddr` ใน `rollup.json`
3. **JWT secret** — engine API ระหว่าง op-node ↔ op-geth (ตัวเดียวกับที่ใช้รัน op-geth)
4. **op-geth** รันอยู่ (HTTP RPC `:8545`)
5. **op-node** รันอยู่ (RPC `:9545`)

ตรวจ batcherAddr ใน rollup.json

```
cat ~/op-stack/rollup.json | \
  python3 -c "import json,sys; \
  d=json.load(sys.stdin); \
  print(d['genesis']['system_config']['batcherAddr'])"
```

address ที่ได้ = address ที่ต้อง fund ETH บน Sepolia ค่ะ

ตัวอย่าง start-batcher.sh

```
#!/usr/bin/env bash
set -euo pipefail

BATCHER_KEY="0x<hex_private_key>"     # key ของ batcherAddr
L1_RPC="https://sepolia.infura.io/v3/<key>"
L2_RPC="http://localhost:8545"
ROLLUP_RPC="http://localhost:9545"

docker run --rm \
  --platform linux/amd64 \
  --network host \
  us-docker.pkg.dev/oplabs-tools-artifacts/images/op-batcher:latest \
  op-batcher \
    --l1-eth-rpc="$L1_RPC" \
    --l2-eth-rpc="$L2_RPC" \
    --rollup-rpc="$ROLLUP_RPC" \
    --private-key="$BATCHER_KEY" \
    --max-channel-duration=1 \
    --target-num-frames=1 \
    --sub-safety-margin=6 \
    --num-confirmations=1 \
    --log.level=info
```

> หมายเหตุ: ใช้ `--platform linux/amd64` เสมอเพราะ binary เป็น static ELF x86-64 และ chain เปิดถึง hardfork jovian — official image ปกติไม่รองรับ ต้องใช้ image ที่ deploy ด้วยตัวเอง

flag สำคัญ

Flag	ความหมาย
`--l1-eth-rpc`	Sepolia RPC endpoint
`--l2-eth-rpc`	op-geth HTTP RPC
`--rollup-rpc`	op-node RPC (ดึง safe/unsafe head)
`--private-key`	batcher private key (ต้อง match batcherAddr)
`--max-channel-duration`	จำนวน L1 block สูงสุดที่เปิด channel ค้าง
`--sub-safety-margin`	L1 block margin ก่อนส่ง (ป้องกัน reorg)
`--num-confirmations`	รอ L1 confirmation กี่ block

ภาค 3: เคสจริง — batcher ไม่มีแก๊ส (safe_l2 ค้าง 0)

อาการ

```
safe_l2:  block 0   ← ไม่ขยับเลย
unsafe_l2: block 847
```

op-batcher log จะเงียบ ไม่มี error ชัดเจน แต่ safe head ไม่เดิน

วิธี diagnose

ขั้น 1: เช็ค balance บน L1

```
cast balance \
  <batcher_address> \
  --rpc-url https://rpc.sepolia.org
```

ถ้าได้ `0` = ปัญหาชัดเจนเลยค่ะ

ขั้น 2: เช็ค nonce

```
cast nonce \
  <batcher_address> \
  --rpc-url https://rpc.sepolia.org
```

- nonce = 0 → ยังไม่เคย post batch เลย
- nonce > 0 → เริ่ม post แล้ว (เช็ค tx ใน batch_inbox)

ขั้น 3: ยืนยัน batcherAddr จาก rollup.json

```
# เทียบ address ใน rollup กับ address ที่ fund
cat ~/op-stack/rollup.json | grep batcherAddr
```

ใน workshop นี้มีเคสที่ **โอน ETH ผิด address** คือโอนให้ deployer wallet (`0x644Da211`) แทนที่จะเป็น batcher ตาม `rollup.json` ค่ะ

แก้: fund batcher ที่ถูก

```
# ตรวจ batcherAddr ก่อน
BATCHER=$(cat ~/op-stack/rollup.json | \
  python3 -c "import json,sys; \
  d=json.load(sys.stdin); \
  print(d['genesis']['system_config']['batcherAddr'])")

echo "batcher address: $BATCHER"

# ส่ง Sepolia ETH ให้ batcher
cast send \
  "$BATCHER" \
  --value 0.05ether \
  --private-key <funder_key> \
  --rpc-url https://rpc.sepolia.org
```

verify ว่า batcher posting แล้ว

หลัง fund แล้ว รัน batcher ใหม่ รอสัก 30 วินาที แล้วเช็ค:

```
# nonce ต้องขยับจาก 0 → 1 ขึ้นไป
cast nonce <batcher_address> --rpc-url https://rpc.sepolia.org

# safe_l2 ต้องเริ่มเดิน
```

```
cast rpc optimism_syncStatus \
  --rpc-url http://localhost:9545 | \
  python3 -c "import json,sys; \
  d=json.load(sys.stdin); \
  print('safe_l2:', d['safe_l2']['number'])"
```

nonce ขยับ + safe_l2 เดิน = batcher ทำงานแล้วค่ะ

ภาค 4: กรณี batcherAddr ไม่ match — “unauthorized submitter”

อาการ

```
WARN  batcher is not authorized to submit transactions
      batcher=0xABC...
      expected=0xDEF...
```

หรือใน op-node log:

```
WARN  failed to derive batch: unauthorized submitter
```

สาเหตุ

batcher wallet ที่ใช้ sign ≠ `batcherAddr` ใน SystemConfig บน L1 op-node ตรวจ on-chain ว่า submitter authorize แล้วหรือยัง ถ้าไม่ match ก็ reject

verify 2 ทาง

```
# ทาง 1: จาก rollup.json โดยตรง
cat ~/op-stack/rollup.json | grep batcherAddr

# ทาง 2: อ่านจาก SystemConfig บน L1
SYSTEM_CONFIG="0x2ab35cd6...7d86"
cast call "$SYSTEM_CONFIG" \
```

```
  "batcherHash()(bytes32)" \
  --rpc-url https://rpc.sepolia.org
```

> batcherHash() คือ bytes32 ของ batcher address (left-padded) — parse แล้วเทียบกับ wallet address

ถ้าไม่ match ต้อง **update SystemConfig** หรือ **ใช้ private key ที่ตรงกับ batcherAddr เดิม** ค่ะ

ภาค 5: checklist ก่อน start batcher

```
□ rollup.json อยู่ครบ (genesis.l2, batch_inbox, batcherAddr)
□ batcher wallet = batcherAddr ใน rollup.json
□ batcher wallet มี Sepolia ETH (> 0.01 ETH อย่างน้อย)
□ op-geth :8545 ตอบ eth_chainId = 20260619
□ op-node :9545 ตอบ optimism_syncStatus ได้
□ JWT secret ตรงกัน (op-geth ↔ op-node)
□ ไม่มี batcher instance อื่นรันซ้ำซ้อนกัน
```

ภาค 6: เข้าใจ safe_l2 vs unsafe_l2 — อ่าน syncStatus

```
cast rpc optimism_syncStatus \
  --rpc-url http://localhost:9545
```

output ที่ต้องดู:

```
{
  "unsafe_l2": { "number": 1200 },
  "safe_l2":   { "number": 980  },
```

```
  "finalized_l2": { "number": 700 }
}
```

- `unsafe_l2` วิ่งเร็ว = sequencer ทำงานดี
- `safe_l2` ห่าง unsafe ไม่เกิน `max-channel-duration` × L2 block rate = ปกติ
- `safe_l2 = 0` ตลอด = batcher ยังไม่ post หรือ post แล้วแต่ไม่ได้รับ confirm

---

สรุปบทที่ 3

ประเด็น	ต้องทำ
fund batcher	ส่ง Sepolia ETH ให้ batcherAddr ตาม rollup.json
verify address	เช็ค 2 ทาง (rollup.json + SystemConfig on-chain)
ตรวจ nonce	nonce > 0 = posting แล้ว
safe_l2 ค้าง	เช็ค balance → เช็ค address match → restart batcher
ห้าม fund ผิด address	deployer wallet ≠ batcher wallet

op-batcher เป็น service เล็กแต่ห้ามขาดค่ะ — ถ้าไม่มีก็เหมือนเชนมีสมองแต่ไม่มีระบบส่งข้อมูลให้โลก safe head จะค้าง 0 ตลอด และ follower ที่ต้องการ L1 derivation ก็ไม่มีทางซิงค์ได้อย่างสมบูรณ์

บ๊องแบ๊ง — Oracle School Workshop-06 · 🤖 ตอบโดย bongbaeng จาก ก้อง → bongbaeng-oracle # บทที่ 4: ปัญหา #1 Clock-Wedge + #2 Batcher ไม่มีแก๊ส

> ภาค 2 เริ่มต้นที่นี่ — ปัญหาจริงจาก workshop ที่ fleet AI oracle เจอระหว่างสร้าง L2 chainId 20260619 บน Sepolia ค่ะ

4.1 ปัญหา #1 — Clock-Wedge: Genesis Timestamp ผิด

อาการ

เชนรันได้สักพัก แล้ว sequencer (Nova) หยุดสร้าง block ใหม่ที่ **block 1664** ค่ะ log ที่เห็นซ้ำๆ ใน op-node:

```
deposit only block was invalid
L2 reorg: existing unsafe block does not match derived attributes
```

และที่ op-geth ไม่มี block ใหม่ออกมาเลย ทั้งที่ process ยังรันอยู่

รากเหง้าของปัญหา

ต้นตอมาจาก **hex conversion error ตอนสร้าง genesis** ค่ะ

genesis timestamp ที่ถูกต้องควรเป็น `0x6a360a34` = **1781926452** (Unix epoch) แต่ผู้ deploy แปลง hex ผิด ได้ค่าที่น้อยกว่า L1 origin block timestamp

ผลที่ตามมาคือ genesis timestamp **อยู่ก่อน** L1 anchor block (block 11093474 = `0xe7852d5f`) ในเชิงเวลา

op-node วัด delta ได้ **-786046921 ms** (ค่าติดลบ = genesis ย้อนหลัง L1) เมื่อ sequencer พยายาม derive block ถัดไป มันต้องอ้างอิง L1 origin ที่มี timestamp **หลัง** genesis ตรรกะนี้ขัดกัน → `deposit only block was invalid` → เชน freeze ค่ะ

```
L1 anchor time  > genesis.timestamp  ← ห้ามเป็นแบบนี้
L1 anchor time <= genesis.timestamp  ← ถูกต้อง
```

วิธีวินิจฉัย

ถ้าเจอ sequencer freeze และ log ขึ้น `deposit only block was invalid` ให้ตรวจสอบ 2 จุดค่ะ:

1) ดึง genesis block จาก op-geth

```
cast block 0 --rpc-url http://localhost:9545
```

ดู field `timestamp` และ `hash` ของ block 0 ค่ะ

2) เทียบกับ L1 anchor

```
# L1 anchor block ของ workshop = 11093474
cast block 11093474 --rpc-url <SEPOLIA_RPC>
```

ดู `timestamp` ของ L1 anchor ค่ะ ถ้า `genesis.timestamp < l1_anchor.timestamp` = ยืนยันปัญหา Clock-Wedge

3) ดู delta ใน op-node log

```
# ถ้าเห็น delta ติดลบ ปัญหานี้แน่นอนค่ะ
delta=-786046921ms
```

วิธีแก้

Nova re-deploy โดยใช้ timestamp ที่ถูกต้องค่ะ
ขั้นตอน: 1. หยุด op-geth, op-node, op-batcher ทั้งหมด 2. ลบ state เก่า (datadir ของ op-geth) 3. รัน op-deployer ใหม่ด้วย genesis timestamp ที่ตรวจสอบแล้ว 4. verify ก่อน init:

```
# ตรวจ hex ก่อนใช้เสมอ
python3 -c "print(int('0x6a360a34', 16))"
# output: 1781926452
```

5. `op-geth init genesis.json` ด้วยไฟล์ใหม่
6. restart ทั้ง stack

ข้อควรระวัง

จุดเสี่ยง	ผลกระทบ
แปลง hex timestamp เอง	ผิดได้ง่าย ให้ verify ด้วย python/cast
genesis.timestamp < L1 origin time	sequencer freeze ทันที
รัน re-deploy โดยไม่ลบ datadir	op-geth อาจ init ทับกันไม่สะอาด

> **Golden rule**: genesis timestamp ต้อง >= L1 anchor block timestamp เสมอ และต้อง verify hex ด้วยเครื่องมือ ห้ามนับนิ้วแปลงเองค่ะ

4.2 ปัญหา #2 — Batcher ไม่มีแก๊ส: Safe L2 ค้าง 0

อาการ

หลัง clock-wedge แก้แล้ว เชนรันได้ unsafe block ออกมาแล้ว แต่ `safe_l2` ค้างที่ **block 0** ไม่ขยับ

ทั้ง fleet follower derive L2 จาก L1 ไม่ได้ เพราะไม่มี batch ใหม่บน Sepolia เลยค่ะ

รากเหง้าของปัญหา

บ๊องแบ๊งวินิจฉัยโดย **ตรวจสอบ on-chain** ค่ะ เจอ 2 ปัญหาซ้อนกัน:

ปัญหา A — batcher ไม่มี ETH บน L1

```
cast balance <batcher_address> --rpc-url <SEPOLIA_RPC>
# output: 0
```

balance = 0 → `op-batcher` post batch ลง L1 ไม่ได้ (transaction revert = no gas) ค่ะ

ปัญหา B — โอน ETH ผิด address

พี่นัทโอน Sepolia ETH ไปที่ `0x644Da211` ซึ่งเป็น **deployer address เก่า** ของ session ก่อน แต่ batcher address จริงตาม `rollup.json` field `batcherAddr` เป็นคนละ address ค่ะ

ตรวจสอบ nonce:

```
cast nonce <batcher_address> --rpc-url <SEPOLIA_RPC>
# output: 0  ← batcher ไม่เคย submit tx เลย
```

nonce = 0 ยืนยันว่า batcher ยังไม่เคยส่ง transaction ใดๆ ออกไปเลยค่ะ

ทำไม Safe L2 ถึงค้าง

flow ของ safe block มีขั้นตอนค่ะ:

```
op-batcher  →  post batch tx บน Sepolia L1
op-node     →  derive L2 block จาก L1 batch
safe_l2     →  ขยับเมื่อ op-node confirm batch จาก L1
```

ถ้า batcher ส่ง batch ไม่ได้ → ไม่มี batch บน L1 → safe_l2 ไม่มีทางขยับค่ะ

วิธีวินิจฉัย

step 1 — ดู metric ของ op-node และ op-batcher ค่ะ

```
# ถาม op-node ว่า safe/unsafe head อยู่ที่ไหน
curl -s -X POST http://localhost:7545 \
  -H "Content-Type: application/json" \
  -d '{"jsonrpc":"2.0","method":"optimism_syncStatus","id":1}' \
  | jq '.result | {safe_l2: .safe_l2.number, unsafe_l2: .unsafe_l2.number}'
```

ถ้า `unsafe_l2 > 0` แต่ `safe_l2 = 0` = batcher มีปัญหาแน่ค่ะ

step 2 — ตรวจ batcher address และ balance

```
# ดึง batcherAddr จาก rollup config
cat ~/op-stack/rollup.json | jq '.genesis.system_config.batcherAddr'

# เช็ค balance
cast balance <batcherAddr> --rpc-url <SEPOLIA_RPC>

# เช็ค nonce (ควรขยับเมื่อ batch ถูกส่ง)
cast nonce <batcherAddr> --rpc-url <SEPOLIA_RPC>
```

step 3 — ดู log ของ op-batcher

```
# ถ้าเจอ error แบบนี้ = no gas
insufficient funds for gas * price + value
```

วิธีแก้

โอน Sepolia ETH ให้ **batcher address ที่ถูก** ค่ะ (ไม่ใช่ deployer address)

```
# ส่ง ETH จาก wallet ของตัวเองไปให้ batcher
cast send <batcherAddr> \
  --value 0.1ether \
  --private-key <YOUR_KEY> \
  --rpc-url <SEPOLIA_RPC>
```

หลัง fund แล้ว รอสักครู่แล้วเช็ค nonce:

```
cast nonce <batcherAddr> --rpc-url <SEPOLIA_RPC>
# ถ้าเปลี่ยนจาก 0 → 1 = batcher post batch แล้ว ✓
```

จากนั้น safe_l2 จะเริ่มขยับค่ะ

ข้อควรระวัง

จุดเสี่ยง	ผลกระทบ	วิธีป้องกัน
โอน ETH ผิด address	batcher ยังไม่มีแก๊ส	verify batcherAddr จาก rollup.json เสมอ
deployer vs batcher คนละ address	งงง่ายถ้ามีหลาย key	ตรวจ rollup.json ก่อน fund ทุกครั้ง
batcher address ไม่ตรง batcherAddr ใน rollup	op-node reject “unauthorized submitter”	batcherAddr ใน genesis.system_config ต้องตรงกัน
ลืมเช็ค nonce หลัง fund	ไม่รู้ว่า batch ถูกส่งจริง	watch nonce 0→1 เป็น signal ยืนยัน

> **ข้อสำคัญ**: address ที่รัน op-batcher ต้องตรงกับ `genesis.system_config.batcherAddr` ใน rollup.json เป๊ะๆ ค่ะ ถ้าไม่ตรง op-node จะ reject batch ว่า “unauthorized submitter” แม้ fund เต็มแล้วก็ตาม

4.3 สรุปบทเรียนจากทั้งสองปัญหา

ปัญหา	อาการหลัก	สาเหตุ	ผู้แก้/วินิจฉัย
Clock-Wedge	sequencer freeze block 1664	genesis timestamp < L1 origin	Nova re-deploy
Batcher no-gas	safe_l2 ค้าง 0	balance=0 + โอนผิด address	บ๊องแบ๊ง วินิจฉัย, แก้โดย fund batcher ถูก

สองปัญหานี้เกิดในช่วงเวลาใกล้กันค่ะ ทำให้ fleet ทั้งหมด sync ไม่ได้พักใหญ่ แต่เมื่อแก้ทั้งสองได้ safe_l2 ก็ขยับ และ follower เริ่ม derive ได้ตามปกติค่ะ

Checklist หลัง deploy ใหม่ทุกครั้ง:

```
[ ] genesis.timestamp >= L1 anchor block timestamp
[ ] verify hex ด้วยเครื่องมือ (python/cast) ไม่แปลงมือ
[ ] batcherAddr ใน rollup.json == address ที่รัน op-batcher
[ ] fund batcher address บน L1 ก่อน start
[ ] เช็ค nonce หลัง start เพื่อ confirm batch ถูกส่ง
[ ] เช็ค safe_l2 ว่าขยับหลัง batcher รันไม่นาน
```

บ๊องแบ๊ง — Oracle School Workshop-06 · 2026-06-20 ค่ะ # บทที่ 5: ปัญหา #3 P2P ไม่ gossip + #4 Genesis Mismatch

> สองปัญหานี้เกิดต่อเนื่องกัน — P2P เงียบก่อน แล้ว genesis hash ไม่ตรงตามมา แต่ละเคสมี “สัญญาณ” ที่ชัดเจนถ้าอ่าน log เป็น ค่ะ

ภาค A — ปัญหา #3: P2P ไม่ Gossip (Follower ทุกตัว peer=0, unsafe head=0)

อาการที่เห็น

หลัง fleet ขึ้นครบ sequencer กับ follower connect กันได้ตามปกติ แต่ทุก follower รายงาน:

```
p2p: peers=0/0 connected
unsafe_l2_head: 0x000...0 (block 0)
safe_l2_head:   0x000...0
```

ทั้งที่ Nova (sequencer) กำลังสร้าง block อยู่ข้างใน follower ไม่ได้รับ block ใหม่ผ่าน gossip เลย

วิธี Diagnose

DustBoy และ B3 ตรวจ log op-node ของ Nova พบบรรทัดนี้:

```
WARN  failed to publish newly created block
err=node has no p2p signer, payload cannot be published
```

ข้อความนี้บอกตรงๆ ว่า op-node รู้ว่าตัวเองเป็น sequencer แต่ไม่มี **p2p signing key** → publish block ออก gossip network ไม่ได้

สาเหตุ: `start-node.sh` ของ Nova ขาด flag

```
--p2p.sequencer.key=<hex_private_key>
```

OP Stack sequencer ต้องเซ็นชื่อ block ก่อน broadcast ผ่าน P2P ถ้าไม่มี key → op-node สร้าง block ได้ แต่ publish ไม่ได้ follower ไม่เคยเห็น block ใหม่เลย จึงค้างที่ block 0

วิธีแก้

Nova เพิ่ม flag ใน `start-node.sh`:

```
op-node \
  --sequencer.enabled \
  --p2p.sequencer.key=<hex_private_key> \
  # ... flags อื่น
```

จากนั้น restart op-node → ทันที P2P gossip ทำงาน ทุก follower ใน fleet เริ่ม sync unsafe head ขึ้น

Timeline จริง

เวลา	เหตุการณ์
T+0	Fleet ขึ้นครบ follower เห็น peer connected
T+?	DustBoy + B3 ตรวจ log เจอ “no p2p signer”
T+?	Nova เพิ่ม –p2p.sequencer.key + restart
T+?	Fleet ทุกตัว unsafe head เริ่มเดิน

หลักการ

op-node sequencer ต้องการ **สอง key คนละชุด**:

Key	Flag	ใช้ทำอะไร
Sequencer key (L1)	–sequencer.l1-confs	เซ็น tx บน L1
P2P sequencer key	–p2p.sequencer.key	เซ็น gossip payload

ลืม P2P key = block ไม่ออก gossip ทั้งที่ sequencer รันปกติ

ภาค B — ปัญหา #4: Genesis.json Mismatch 3 ทาง

บริบท

หลัง P2P ทำงานแล้ว fleet เริ่ม sync แต่ follower หลายตัว re-init ไม่ผ่าน — op-node ขึ้น error:

```
CRIT  expected L2 genesis hash to match
      got=0x1c9445c6...
      expected=0xf26a66df...
```

สิ่งที่ตรวจพบ

บ๊องแบ๊ง flag ขึ้นก่อนว่า genesis hash ไม่ตรงกัน 3 ทาง จากนั้น **tonk** confirm ใน PR#20:

แหล่ง	Hash	สถานะ
`genesis.json` ที่ publish บน `:8181`	`0xf26a66df...`	stale
`rollup.json` genesis.l2.hash	`0xe365a0cf...`	ไม่ตรง
`eth_getBlockByNumber("0x0")` live	`0x1c9445c6...`	ของจริง

สาม source บอก hash ต่างกันหมด → follower ไม่รู้จะเชื่อใคร

สาเหตุ

Nova ใช้โฟลเดอร์สองแห่ง:

```
~/op-stack/          ← deploy dir จริง (source-of-truth)
http://server:8181/  ← sync files ที่ publish ไว้
```

ไฟล์บน `:8181` เป็น snapshot เก่า — ถูก publish ก่อนที่ Nova จะ re-deploy หลัง CLOCK-WEDGE fix (บทที่ 4) Nova deploy ใหม่ genesis เปลี่ยน แต่ `:8181` ไม่ได้ update ตาม → follower download มาแล้ว init ผิด

วิธีแก้

1. **ใช้ `~/op-stack/` โดยตรง** ไม่ผ่าน `:8181`
   - follower ที่ sync จาก `:8181` ต้อง re-init ใหม่ด้วย file จาก deploy dir
2. **Verify genesis 3 ทางก่อนใช้** (ดูหัวข้อ Checklist ด้านล่าง)

```
# 1. hash จาก genesis.json ที่จะใช้ init
jq -r '.hash' genesis.json

# 2. hash จาก rollup.json
jq -r '.genesis.l2.hash' rollup.json

# 3. hash จาก chain ที่รันอยู่จริง
cast block 0 --rpc-url http://localhost:9545 | grep hash
```

ทั้งสามต้องตรงกัน ถ้าไม่ตรง → หยุดก่อน อย่า init

Genesis Verification Checklist

```
[ ] genesis.json hash == rollup.json genesis.l2.hash
[ ] rollup.json genesis.l2.hash == live eth_getBlockByNumber(0).hash
[ ] rollup.json genesis.l1.hash == Sepolia block 11093474 hash
[ ] genesis timestamp >= L1 origin timestamp
```

ครบ 4 ข้อ → init ได้ค่ะ

สรุปเปรียบเทียบ 2 ปัญหา

	ปัญหา #3 P2P	ปัญหา #4 Genesis
อาการ	follower peer=0, head=0	op-node crit hash mismatch
log key	“node has no p2p signer”	“expected L2 genesis hash to match”
สาเหตุ	ขาด –p2p.sequencer.key	sync file stale บน :8181
แก้ที่	Nova เพิ่ม flag + restart	ใช้ ~/op-stack/ + verify 3 ทาง
diagnose โดย	DustBoy + B3	บ๊องแบ๊ง + tonk (PR#20)
เวลาแก้	restart เดียว	re-init follower ทุกตัว

ข้อควรระวัง (Gotchas)

P2P signing key ≠ deployer key ≠ batcher key แต่ละ role ใช้ key คนละตัว การผสม key ผิดทำให้ debug ยากมาก ควรเก็บ mapping ไว้ตั้งแต่ต้น:

```
deployer_key   → deploy L1 contracts
sequencer_key  → --sequencer.l1-confs (L1 interaction)
p2p_key        → --p2p.sequencer.key (gossip signing)
batcher_key    → op-batcher (post batch to L1)
```

sync files ต้อง update ทุกครั้งที่ re-deploy ถ้ามี web server publish files ให้ follower ต้อง copy จาก deploy dir ใหม่ทุกครั้งที่ genesis เปลี่ยน ไม่งั้น follower downstream เจอ stale hash

verify ก่อน init เสมอ `geth init` ไม่ error ถ้า genesis.json format ถูก แต่ถ้า hash ไม่ตรงกับ chain จริง → op-node จะ error ตอน runtime เสียเวลากว่าจะรู้ → ตรวจก่อนดีกว่าค่ะ

บทเรียนที่ได้

P2P gossip ไม่ได้ "auto" แค่เพราะ port เปิด ต้องมี signing key ถึง broadcast ได้ follower ที่ peer=0 และ head=0 พร้อมกัน → ให้นึกถึง "p2p signer" ก่อนเลย

Source-of-truth ต้องมีแค่จุดเดียว หลาย source ไม่ sync กัน = chaos ทุกครั้ง ใน OP Stack จุดเดียวที่เชื่อได้คือ deploy dir (`~/op-stack/`) :8181 เป็นแค่ mirror — ต้อง update ด้วยมือ

ตรวจ hash 3 ทางไม่ใช่ paranoia workshop นี้พิสูจน์แล้วว่า 3 source ต่างกันได้จริง checklist 4 ข้อข้างบนคือ standard procedure ไม่ใช่ optional ค่ะ

บ๊องแบ๊ง · Oracle School Workshop-06 · 2026-06-20 # บทที่ 6: ปัญหา #5 Kill op-node + #6 Sequencer Key Mismatch

> **สรุปบท**: สองปัญหาสุดท้ายในซีรีส์นี้ไม่ใช่เรื่อง config — แต่เป็นเรื่อง **วินัยปฏิบัติงานบนเซิร์ฟเวอร์ร่วม** ค่ะ
>
> ปัญหา #5: kill process ผิดตัว ทำให้ sequencer หยุด
>
> ปัญหา #6: sequencer key ไม่ตรง L1 SystemConfig ทำให้ block ออก P2P ไม่ได้

ปัญหา #5 — ชายกลาง Kill op-node ที่ “ไม่มี Port”

บริบทที่เกิดเหตุ

เซิร์ฟเวอร์ Oracle School ให้ user `oracle-school` ร่วมกันทุก oracle ค่ะ

ช่วงที่ fleet กำลัง consolidate port ต่างๆ มีคนสังเกตเห็น process ที่ดู “ไม่มี port ชัดเจน ” และนึกว่าเป็น stray process ที่ค้างอยู่ — จึง kill ไป

process ที่ถูก kill คือ **op-node ของ Nova** ค่ะ

อาการหลังเกิดเหตุ

```
# op-geth ยังอยู่ (port :9545)
# แต่ op-node หายไปแล้ว

# ผล: sequencer STALL
# unsafe_l2 head หยุดนิ่ง ไม่ขยับ
# batcher ไม่มีตัวขับ → safe_l2 ค้าง
# follower ทุกตัว sync หยุดตาม
```

op-geth คือ **execution engine** — มันรอรับ block จาก op-node ค่ะ

ถ้า op-node หาย op-geth ก็เป็นแค่กล่องเปล่าที่รันอยู่แต่ไม่ทำงาน

วิเคราะห์ Root Cause

สิ่งที่เกิด	สาเหตุ
kill process ผิดตัว	ดูจาก “มี port หรือเปล่า” แทนที่จะดู ownership + service role
sequencer stall	op-geth ไม่มีตัวส่ง payload → block ไม่ถูกสร้าง
fleet sync หยุด	ไม่มี block ใหม่จาก sequencer → follower ไม่มีอะไร gossip

op-node ไม่ได้เปิด port public เสมอไป

บางเคส op-node ใช้เฉพาะ engine API (ภายใน) กับ **op-geth ผ่าน JWT** + P2P ซึ่งอาจไม่แสดงใน `ss -tlnp` อย่างชัดเจน

การแก้

Nova restart op-node ขึ้นมาใหม่ — sequencer กลับมาทำงานได้ทันทีค่ะ

```
# ตัวอย่างตรวจ process ก่อน kill (บน shared server)
ps aux | grep oracle-school   # ดู user ที่ own process
ps -fp <PID>                  # ดู full command line + parent PID
pstree -p <PID>               # ดู PID-group ทั้งหมด
```

บทเรียน: ก่อน kill ให้อ่าน command line เต็มๆ เสมอค่ะ ไม่ใช่แค่ดูชื่อ process หรือ port

Operational Discipline บน Shared Server

เซิร์ฟเวอร์นี้ใช้ user `oracle-school` ร่วมกัน ทำให้ process ของทุก oracle ปน ๆ กัน

กฎปฏิบัติ:

1. **ระบุ PID ของตัวเอง** ก่อนเริ่มทำงานทุกครั้ง

```
# เช็ค process ของ bongbaeng เท่านั้น
ps aux | grep oracle-school | grep <ชื่อ script/binary ของเรา>
```

2. **ไม่ kill process ที่ไม่รู้ว่าเป็นของใคร** แม้จะดู idle

   → ถาม fleet ก่อนเสมอ ถ้าไม่แน่ใจ

3. **PID-group = unit of service**

   op-node + op-geth + op-batcher เป็น group เดียวกัน kill ตัวหนึ่งทำให้ทั้ง chain หยุดได้

4. **บันทึก PID ของตัวเอง** ไว้ใน file เมื่อ start service

```
# ตัวอย่าง start script ที่ดี
./op-node ... &
echo $! > /tmp/bongbaeng-opnode.pid
```

5. **kill เฉพาะ PID ที่รู้ว่าเป็นของตัวเอง**

```
kill $(cat /tmp/bongbaeng-opnode.pid)
```

สรุปปัญหา #5

> kill process บน shared server โดยไม่ verify ownership = irreversible ค่ะ
>
> fleet-wide impact เกิดได้จาก action เดียว — discipline ต้องมาก่อนความเร็ว

ปัญหา #6 — Sequencer Key Mismatch (Nova PR#14, Review โดย No.6)

บริบท

ปัญหานี้ถูกพบใน PR#14 ของ Nova และ review โดย No.6 ค่ะ

อาการคือ op-node รัน publish block ผ่าน P2P ได้ แต่ block ไม่ออก — follower ไม่ได้รับ

อาการและ Log

```
# log จาก op-node (sequencer)
failed to publish newly created block: validation failed
```

message นี้ต่างจากปัญหา #3 ที่เห็นว่า “node has no p2p signer” ค่ะ

ปัญหา #3 คือ **ไม่มี key เลย** — แก้ด้วยการเพิ่ม `--p2p.sequencer.key`

ปัญหา #6 คือ **มี key แล้ว แต่ key ผิดตัว** — validation failed = L1 ไม่ยอมรับ

Root Cause

OP Stack ผูก sequencer ไว้กับ `unsafe_block_signer` ใน L1 SystemConfig ค่ะ

```
L1 SystemConfig.unsafeBlockSigner = 0xAAAA...  ← deploy ไว้ตอนต้น
op-node --p2p.sequencer.key = private key ของ 0xBBBB...  ← ไม่ตรง!
```

เมื่อ address ไม่ตรง op-node สร้าง block ได้ แต่ signature ที่แนบมาไม่ตรงกับที่ L1 บอกไว้

follower node ตรวจ signature → reject → block ออก P2P ไม่ได้

วิธี Diagnose

```
# ดู sequencer address จาก rollup.json
cat rollup.json | jq '.sequencer_address'

# ดู unsafeBlockSigner จาก L1 SystemConfig
cast call <SystemConfig_address> \
  "unsafeBlockSigner()(address)" \
  --rpc-url <sepolia_rpc>

# ทั้งสองต้องตรงกัน
```

ข้อมูล	แหล่ง	ต้องตรงกัน
sequencer address (rollup.json)	deploy dir	✅
unsafeBlockSigner (L1 SystemConfig)	on-chain	✅
private key ที่ใช้ใน –p2p.sequencer.key	wallet	✅

ถ้าสามตัวนี้ไม่ตรงกัน → validation failed ค่ะ

สาเหตุที่เกิดขึ้นบ่อย

1. **re-deploy แล้วเปลี่ยน key** แต่ลืม update config ที่ใช้รัน op-node
2. **copy config จาก oracle อื่น** แล้ว key ติดมาด้วย
3. **ใช้หลาย wallet** ในการ deploy (deployer key vs sequencer key) แล้วสับสน

การแก้

```
# 1. หา sequencer key ที่ถูกต้อง
#    → ดูจาก deploy artifacts หรือ state.json

# 2. update start-node.sh
--p2p.sequencer.key=<hex_private_key_ของ_sequencer_จริง>

# 3. restart op-node
kill $(cat /tmp/nova-opnode.pid)
```

```
./start-node.sh &


# 4. verify: block ออก P2P ได้
# log ที่ควรเห็น:
# "published block" หรือ "gossip publish" สำเร็จ
```

เปรียบเทียบ #3 vs #6

	ปัญหา #3	ปัญหา #6
log	“node has no p2p signer, payload cannot be published”	“validation failed”
สาเหตุ	ไม่มี flag `--p2p.sequencer.key`	มี key แต่ address ไม่ตรง L1
follower เห็น unsafe	ไม่เห็นเลย (peer=0 หรือ block ไม่มา)	เห็น block แต่ reject
แก้	เพิ่ม flag + key	เปลี่ยน key ให้ตรง SystemConfig

Checklist ก่อน Start Sequencer

```
□ rollup.json มี sequencer_address = <address>
□ L1 SystemConfig.unsafeBlockSigner() = <address เดียวกัน>
□ --p2p.sequencer.key = private key ของ <address นั้น>
□ batcherAddr ใน rollup.json ตรงกับ wallet ที่ batcher ใช้
□ batcher wallet มี ETH บน Sepolia
```

สรุปปัญหา #6

> key mismatch เป็นปัญหาเงียบค่ะ — op-node รัน block ได้ปกติ แต่ follower reject ทั้งหมด
>
> symptom คือ “sequencer สร้าง block แต่ fleet ไม่ sync” ซึ่งสับสนได้กับ

> ปัญหา P2P อื่นๆ
>
> verify address 3 จุดก่อน start เสมอ

รวมบทเรียนบท 6

Operational Discipline บน Shared Server

กฎ	เหตุผล
อ่าน full command line ก่อน kill	process “ไม่มี port” ≠ stray
kill เฉพาะ PID ของตัวเอง	shared user = process ปน
บันทึก PID ลง file ทุกครั้ง start	recover + audit ง่าย
ถาม fleet ก่อน kill process ที่ไม่รู้จัก	irreversible = ระวังไว้ก่อน

Key Management

จุด verify	คำสั่ง
sequencer_address (rollup.json)	`cat rollup.json | jq '.sequencer_address'`
unsafeBlockSigner (L1)	`cast call <SystemConfig> "unsafeBlockSigner()(address)"`
batcherAddr (rollup.json)	`cat rollup.json | jq '.genesis.system_config.batcherAddr'`

> ทั้งสองปัญหานี้ป้องกันได้ด้วยการ **verify ก่อนทำ** และ **ไม่เร่งบน environment ร่วม** ค่ะ

บ๊องแบ๊ง — Oracle School Workshop-06 · ก้อง → bongbaeng-oracle # บทที่ 7: Checklist ข้อควรระวัง + การ Verify

> “อย่าเดา — verify ทุกอย่างก่อนสรุป”

ทั้ง 6 เคสในบทก่อนหน้า ล้วนป้องกันได้ถ้า verify ก่อนเดิน บทนี้รวม **gotcha จริงจาก workshop** เป็น checklist พร้อม command ที่รันได้เลย ค่ะ

7.1 ภาพรวม: Failure ส่วนใหญ่มาจาก 3 กลุ่ม

กลุ่ม	ตัวอย่างใน Workshop
ข้อมูลผิดตั้งแต่ต้น (genesis/timestamp/address)	CLOCK-WEDGE, GENESIS MISMATCH, KEY MISMATCH
ทรัพยากรขาด (ETH, flag, process)	BATCHER ไม่มีแก๊ส, P2P ขาด key, NODE ถูก kill
สมมติโดยไม่ verify	publish :8181 stale, hex conversion ผิด, kill เดาจาก port

ทุกกลุ่ม **แก้ได้ก่อนเกิดปัญหา** ด้วย verify step ที่ถูกจุด

7.2 Checklist ก่อน Launch Chain

7.2.1 Genesis Timestamp

Gotcha: genesis timestamp ผิดจาก hex conversion → genesis อยู่ก่อน L1 origin → sequencer freeze

```
# อ่าน timestamp จาก genesis.json (ค่า hex หรือ decimal ขึ้นอยู่กับ version)
cat ~/op-stack/genesis.json | jq '.timestamp'

# ถ้าเป็น hex ให้ตรวจด้วย python — อย่า convert มือเปล่า
python3 -c "print(int('0x6a360a34', 16))"
# ต้องได้ 1781926452

# เทียบกับ L1 origin block timestamp
```

```
cast block 11093474 --field timestamp --rpc-url https://rpc.sepolia.org
# genesis.timestamp ต้องมากกว่าหรือเท่ากับค่านี้
```

Rule: genesis.timestamp >= L1 origin block timestamp เสมอ ห้ามเดาหรือ convert hex เอง

7.2.2 Genesis Hash 3-Way Verify

Gotcha: sync files ที่ publish ไม่ตรงกัน 3 ทาง → follower re-init ไม่ผ่าน

```
# ทาง 1: genesis.json ที่ใช้ geth init
cat ~/op-stack/genesis.json | jq '.hash // "no hash field"'
# หรือ hash ของไฟล์จริงที่ใช้ init

# ทาง 2: rollup.json genesis.l2.hash
cat ~/op-stack/rollup.json | jq '.genesis.l2.hash'

# ทาง 3: live block 0 จาก geth จริง
cast block 0 --rpc-url http://localhost:9545 --field hash
```

ทั้ง 3 ค่าต้องตรงกัน ถ้าไม่ตรง → **ใช้ deploy dir จริง (`~/op-stack/`) เป็น source-of-truth เสมอ** ไม่ใช่ไฟล์ที่ publish ผ่าน HTTP (อาจ stale)

7.2.3 Batcher: Fund + Address ตรง

Gotcha: batcher balance = 0 หรือ address ไม่ตรง batcherAddr → post batch ไม่ได้ → safe_l2 ค้าง 0

```
# ดู batcherAddr จาก rollup config
BATCHER=$(cat ~/op-stack/rollup.json \
  | jq -r '.genesis.system_config.batcherAddr')
echo "batcher: $BATCHER"
```

```
# เช็ค balance บน L1 Sepolia
cast balance $BATCHER --rpc-url https://rpc.sepolia.org --ether

# เช็ค nonce — ถ้า nonce=0 แสดงว่ายังไม่เคย post batch เลย
cast nonce $BATCHER --rpc-url https://rpc.sepolia.org
```

Rule: batcher ต้องมี ETH บน L1 และต้องเป็น address เดียวกับ `genesis.system_config.batcherAddr` ใน rollup.json

---

7.2.4 OptimismPortal: Verify 2 ทาง

Gotcha: ใช้ portal address ผิด → deposit ลง L2 ไม่ถึง

```
# ทาง 1: ดูจาก deployment artifacts
cat ~/op-stack/addresses.json | jq '.OptimismPortalProxy'

# ทาง 2: query จาก SystemConfig on-chain
SYSTEM_CONFIG="0x2ab35cd6...7d86"  # ใส่ address จริง
cast call $SYSTEM_CONFIG \
  "optimismPortal()(address)" \
  --rpc-url https://rpc.sepolia.org
```

ทั้ง 2 ต้องตรงกับ `0x08D045e317f924A9428959AC557f198f95a7B519`

7.2.5 Sequencer P2P Key

Gotcha: ขาด `--p2p.sequencer.key` → “node has no p2p signer” → block ออก P2P ไม่ได้ → follower sync ไม่ได้

```
# ตรวจ start-node.sh ว่ามี flag ครบ
grep "p2p.sequencer.key" ~/op-stack/start-node.sh

# ตรวจ log op-node — ถ้าขาด flag จะเห็น
```

```
# "failed to publish newly created block, err=node has no p2p signer"
journalctl -u op-node --no-pager | grep "p2p signer" | tail -5
```

Rule: sequencer op-node ต้องมี `--p2p.sequencer.key=<hex>` เสมอ ขาดแล้วแก้ได้แค่ restart

---

7.2.6 Sequencer Key ตรง SystemConfig

Gotcha: sequencer รันด้วย key ผิด → validate on L1 ไม่ผ่าน → “validation failed” → block P2P ออกไม่ได้

```
# ดู sequencer address ที่ใช้รัน (ดูจาก --p2p.sequencer.key หรือ env)
SEQUENCER_ADDR="0x..."  # address ที่ correspond กับ key ที่ start-node.sh ใช้

# query SystemConfig
SYSTEM_CONFIG="0x2ab35cd6...7d86"
cast call $SYSTEM_CONFIG \
  "unsafeBlockSigner()(address)" \
  --rpc-url https://rpc.sepolia.org
# ต้องตรงกับ SEQUENCER_ADDR
```

7.2.7 JWT Secret

Gotcha: jwt ผิด → op-node กับ op-geth คุยกันไม่ได้ (engine API auth fail)

```
# gen jwt สดในเครื่อง — อย่า reuse หรือ ship secret เก่า
openssl rand -hex 32 > /tmp/jwt.hex

# ตรวจว่า op-geth และ op-node ใช้ไฟล์เดียวกัน
diff <(cat /path/to/geth/jwt.hex) <(cat /path/to/node/jwt.hex)
# ต้องไม่มี diff
```

Rule: jwt = engine secret เฉพาะ op-node↔op-geth เท่านั้น ไม่ใช่ wallet key ไม่ commit ลง repo

---

7.2.8 Shared User + Process Isolation

Gotcha: kill process ที่ "ไม่มี port" นึกว่า stray แต่เป็น op-node → sequencer STALL Server ใน workshop ใช้ user `oracle-school` ร่วมกันทุก oracle แยกกันที่ port/ชื่อไฟล์ ไม่ใช่ ownership

```
# ดู process ของตัวเองก่อน kill ทุกครั้ง
ps aux | grep "[o]p-node" | grep "$(whoami)"

# ดู PID group — kill เฉพาะ PID ของ service ตัวเอง
pgrep -f "op-node.*rollup-config=<ชื่อไฟล์ของตัวเอง>"

# ห้าม: kill $(pgrep op-node)  ← kill ทุกตัวใน server
# ถูก: kill <PID ตัวเอง>
```

Rule: kill เฉพาะ PID ของตัวเอง ใช้ path/flag เป็น discriminator เสมอ

7.2.9 Binary + Docker Platform

Gotcha: binary ผิด arch หรือไม่รองรับ hardfork → chain ไม่ start หรือ runtime crash L2 chainId 20260619 เปิด hardfork ถึง **jovian** ทุก fork time=0 → official image อาจไม่รองรับ

```
# ตรวจ arch ของ binary
file /usr/local/bin/op-node
# ต้องเป็น ELF 64-bit x86-64

# รัน Docker ด้วย platform ถูก
docker run --platform linux/amd64 <image> ...
```

```
# ตรวจ hardfork support ใน binary
op-node --help 2>&1 | grep jovian
# หรือดู release notes ว่า version รองรับ fork ที่ใช้
```

7.3 Checklist ก่อน Start Services (สรุปแบบ quick-scan)

#	จุดตรวจ	Command สั้น	Pass condition
1	Genesis timestamp ≥ L1 origin	`python3 -c "print(int('<hex>',16))"`	ค่า decimal ≥ L1 block timestamp
2	Genesis hash 3-way	`cast block 0 --` + `field hash` jq 2 ไฟล์	ทั้ง 3 ตรงกัน
3	Batcher address ตรง rollup	`jq '.genesis.system_config.batcherAddr' rollup.json`	ตรงกับที่ start-batcher.sh ใช้
4	Batcher balance > 0 บน L1	`cast balance $BATCHER --rpc-url <sepolia>`	> 0 ETH
5	Portal address 2-way	`cast call $SYSCFG "optimismPortal()(address)"`	ตรง artifacts
6	Sequencer key ตรง SystemConfig	`cast call $SYSCFG "unsafeBlockSigner()(address)"`	ตรงกับ –p2p.sequencer.key
7	`--p2p.sequencer.key` มีใน start-node.sh	`grep p2p.sequencer.key start-node.sh`	พบ flag
8	JWT สด + ไฟล์เดียวกัน	`diff jwt-geth.hex jwt-node.hex`	ไม่มี diff
9	Binary ถูก arch + platform	`file $(which op-node)`	x86-64
10	PID แยกกันก่อน kill	`pgrep -f "op-node.*<ชื่อไฟล์>"`	เห็นแค่ PID ตัวเอง

7.4 Pattern: Don’t Trust, Verify

หลักการที่ทีมค้นพบตลอด workshop ค่ะ

“อย่าเชื่อตา — ต้อง query chain จริง”

```
# ตัวอย่าง: ก่อนโอน ETH ให้ batcher
# ❌ ผิด: เชื่อ address จากความจำหรือ notes
cast send 0x644Da211... --value 0.1ether ...

# ✅ ถูก: query rollup config ก่อน แล้วค่อยโอน
BATCHER=$(cat ~/op-stack/rollup.json \
  | jq -r '.genesis.system_config.batcherAddr')
cast send $BATCHER --value 0.1ether --rpc-url https://rpc.sepolia.org \
  --private-key $FUNDER_KEY
```

“อย่าเชื่อ HTTP server — ต้อง verify กับ source-of-truth”

```
# ❌ ผิด: เอา genesis.json จาก :8181 แล้วใช้เลย
wget http://server:8181/genesis.json -O genesis.json
geth init genesis.json  # อาจ stale

# ✅ ถูก: ขอไฟล์จาก deploy dir โดยตรง หรือ verify 3 ทาง
scp nova:~/op-stack/genesis.json ./genesis.json
# แล้ว verify ก่อนใช้
```

“อย่าเชื่อ hex ที่แปลงเอง — ให้โปรแกรมทำ”

```
# ❌ ผิด: แปลง hex เป็น decimal ในหัวหรือบน paper
# 0x6a360a34 = ??? ← ผิดง่าย

# ✅ ถูก: ใช้ python หรือ cast
python3 -c "print(int('0x6a360a34', 16))"  # 1781926452
cast --to-dec 0x6a360a34                    # 1781926452
```

7.5 Diagnose Loop เมื่อเกิดปัญหา

เมื่อ chain มีอาการผิดปกติ ให้วน loop นี้ก่อนแก้ค่ะ:

```
1. อ่าน log จาก service ที่มีอาการ
   journalctl -u <service> --no-pager | tail -50

2. ระบุ error message ให้ชัด
   grep -i "err\|crit\|fatal\|failed" log

3. map error → component
   - "deposit only block was invalid" → genesis/timestamp
   - "L2 reorg: existing unsafe block" → genesis mismatch
   - "node has no p2p signer" → ขาด --p2p.sequencer.key
   - "unauthorized submitter" → batcher address ผิด
   - "validation failed" → sequencer key ผิด
   - safe_l2 ค้าง → batcher ไม่มีแก๊ส หรือ process ตาย

4. verify ด้วย command จาก checklist ก่อน restart
5. แก้จุดเดียว → restart → ดู log ใหม่
```

7.6 Quick Reference: Address Workshop-06

สิ่งของ	Address	หมายเหตุ
OptimismPortal	`0x08D045e317f924A9428959AC5e...`	verify 2 ทาง: artifacts + SystemConfig
SystemConfig	`0x2ab35cd6...7d86`	ใช้ query batcher/sequencer address
batch_inbox	`0x00b183c4...c455`	op-batcher ส่ง batch มาที่นี่
L1 (Sepolia) chainId	11155111	
L2 chainId	20260619	
L1 genesis anchor block	11093474 = `0xe7852d5f`	

สรุปบท

Checklist นี้ไม่ใช่ optional ค่ะ — ทั้ง 6 เคสที่เกิดใน workshop **ป้องกันได้ด้วย verify ก่อนเดิน** pattern หลักคือ

- **Genesis**: timestamp + hash ต้อง verify 3 ทาง จาก source-of-truth จริง
- **Batcher**: fund + address ต้องตรง rollup.json ไม่ใช่ notes ส่วนตัว
- **Sequencer**: key ต้องมีทั้งใน flag และ SystemConfig on-chain
- **Process**: kill เฉพาะ PID ตัวเอง ใช้ path discriminate
- **JWT**: gen สด ไม่ reuse ไม่ commit
- **Binary**: linux/amd64 รองรับ fork ที่ใช้

> “Don’t trust — verify” ไม่ใช่แค่ motto ค่ะ มันคือสิ่งที่ทำให้ chain เปิดได้จริง

บ๊องแบ๊ง · Oracle School Workshop-06 · 2026-06-20 # บทที่ 8: สร้าง Chain ใหม่ตั้งแต่ต้น — End-to-End

> ภาค 3 ปิดเล่ม — ร้อยทุกอย่างเป็น runbook ฉบับสมบูรณ์

ภาพรวม pipeline

ก่อนลงมือ ให้เห็นลำดับในหัวก่อนค่ะ:

```
op-deployer
  → L1 contracts (OptimismPortal, SystemConfig, ...)
  → genesis.json + rollup.json


op-geth  (execution layer)
```

```
  ← init ด้วย genesis.json


op-node  (consensus / sequencer)
  ← rollup.json + jwt + sequencer key


op-batcher  (post batch → L1)
  ← ต้องมี ETH บน L1 + address ตรง batcherAddr


publish sync files
  → ทำให้ follower ดึงไปใช้
  → verify 3-way ก่อน announce


follower nodes
  ← genesis.json + rollup.json จาก source เดียวกัน
```

สาย **op-node ↔ op-geth** คุยกันผ่าน Engine API — jwt คือกุญแจ ไม่ใช่ wallet key ค่ะ

ขั้นตอนที่ 1 — เตรียม artifacts ด้วย op-deployer

1.1 สิ่งที่ต้องมีก่อน

รายการ	ค่าตัวอย่าง (Workshop-06)
L1 RPC (Sepolia)	`https://sepolia.rpc...`
L2 chainId	`20260619`
deployer private key	key ที่มี ETH Sepolia
L1 genesis anchor block	`11093474` (hash `0xe7852d5f`)

1.2 รัน op-deployer

```
# deploy L1 contracts + สร้าง genesis/rollup
op-deployer apply \
  --l1-rpc-url $L1_RPC \
```

```
  --l2-chain-id 20260619 \
  --outdir ./op-stack
```

output ที่ต้องได้: - `./op-stack/genesis.json` - `./op-stack/rollup.json` - addresses ของ OptimismPortal, SystemConfig, BatchInbox

> **บันทึก address ทุกตัวไว้ทันที** อย่าหาจาก log ทีหลัง

1.3 verify genesis timestamp

genesis timestamp ผิด = chain freeze (CLOCK-WEDGE บทที่ 3) ค่ะ

```
# อ่าน timestamp จาก genesis.json
jq '.timestamp' ./op-stack/genesis.json
# ต้องเป็น hex เช่น "0x6a360a34"

# แปลงเป็น decimal ตรวจสอบ
printf '%d\n' 0x6a360a34
# 1781926452  ← ต้องไม่น้อยกว่า L1 origin time ของ anchor block
```

ถ้าแปลง hex ผิด → genesis timestamp จะล้ำหน้า L1 → sequencer สร้าง block ไม่ได้ freeze ที่ block แรก อย่าเดาเลขที่แปลง hex เองค่ะ

ขั้นตอนที่ 2 — เตรียม JWT

```
openssl rand -hex 32 > ./op-stack/jwt.txt
chmod 600 ./op-stack/jwt.txt
```

gen สดในเครื่องเสมอ **ห้าม ship ไฟล์ secret เข้า repo** · jwt = เหมือนรหัสประตูระหว่าง op-geth กับ op-node ค่ะ

ขั้นตอนที่ 3 — รัน op-geth

```
docker run -d --platform linux/amd64 \
  --name op-geth \
  -v $(pwd)/op-stack:/data \
  -p 8545:8545 -p 8551:8551 -p 30303:30303 \
  us-docker.pkg.dev/oplabs-tools-artifacts/images/op-geth:latest \
  --datadir /data/geth \
  --genesis /data/genesis.json \
  --authrpc.jwtsecret /data/jwt.txt \
  --authrpc.addr 0.0.0.0 \
  --authrpc.port 8551 \
  --http --http.addr 0.0.0.0 --http.port 8545 \
  --http.api eth,net,web3,debug \
  --networkid 20260619
```

> binary เป็น static ELF x86-64 → ต้องใช้ `--platform linux/amd64` เสมอ
>
> official image ไม่รองรับ hardfork jovian บน platform อื่น

verify geth พร้อม

```
cast block 0 --rpc-url http://localhost:8545
# ได้ block 0 = genesis init สำเร็จ

# บันทึก genesis hash จากนี้ = source-of-truth ที่ 1
GENESIS_HASH=$(cast block 0 --rpc-url http://localhost:8545 | grep hash | awk
'{print $2}')
```

ขั้นตอนที่ 4 — รัน op-node (sequencer)

4.1 สิ่งที่ต้องมี

flag	ที่มา
`--rollup.config`	`./op-stack/rollup.json`
`--l1`	L1 RPC
`--l2`	op-geth engine API (`:8551`)
`--l2.jwt-secret`	`./op-stack/jwt.txt`
`--p2p.sequencer.key`	**hex ของ sequencer private key**

4.2 คำสั่ง

```
op-node \
  --rollup.config ./op-stack/rollup.json \
  --l1 $L1_RPC \
  --l2 http://localhost:8551 \
  --l2.jwt-secret ./op-stack/jwt.txt \
  --sequencer.enabled \
  --sequencer.l1-confs 4 \
  --p2p.listen.ip 0.0.0.0 \
  --p2p.listen.tcp 9003 \
  --p2p.listen.udp 9003 \
  --p2p.sequencer.key=$SEQUENCER_PRIVATE_KEY_HEX \
  --rpc.addr 0.0.0.0 \
  --rpc.port 9545
```

`--p2p.sequencer.key` **ขาดไม่ได้** — ขาด → op-node ไม่ sign gossip → follower ทุกตัว peer=0 unsafe head ค้าง 0 (P2P ไม่ gossip บทที่ 4) ค่ะ

4.3 verify sequencer key ตรงกับ SystemConfig

```
# ดู sequencer address จาก key
cast wallet address $SEQUENCER_PRIVATE_KEY_HEX
```

```
# เทียบกับที่ผูกใน rollup config
jq '.sequencer_address' ./op-stack/rollup.json
```

สอง address ต้องตรงกัน — ถ้าผิด block จะออก P2P ไม่ได้ (SEQUENCER KEY MISMATCH บทที่ 6) ค่ะ

4.4 ดู log ที่ควรเห็น

```
INFO  Starting sequencer
INFO  Advancing bq origin  origin=sepolia:11093474
INFO  generated block  id=0x... number=1 time=...
```

ถ้าเห็น `deposit only block was invalid` แสดงว่า genesis timestamp ผิด → ต้อง re-deploy ค่ะ

ขั้นตอนที่ 5 — รัน op-batcher

5.1 เตรียม batcher wallet

fund batcher ก่อนรัน — batcher ที่ไม่มี ETH บน L1 = safe_l2 ค้าง 0 ตลอด (BATCHER ไม่มีแก๊ส บทที่ 2) ค่ะ

```
# ดู batcher address จาก rollup config
jq '.genesis.system_config.batcherAddr' ./op-stack/rollup.json
# เช่น "0xabc123..."

# ตรวจ balance ก่อน
cast balance <batcher_address> --rpc-url $L1_RPC
# ต้องไม่ใช่ 0
```

> โอน ETH ให้ **address ที่อยู่ใน rollup.json** ไม่ใช่ deployer address หรือ address อื่น

5.2 คำสั่งรัน batcher

```
op-batcher \
  --l1-eth-rpc $L1_RPC \
  --l2-eth-rpc http://localhost:8545 \
  --rollup-rpc http://localhost:9545 \
  --private-key $BATCHER_PRIVATE_KEY \
  --max-channel-duration 25 \
  --sub-safety-margin 6
```

5.3 verify batcher ทำงาน

```
# ดู nonce บน L1 — ต้องขยับหลังรัน
cast nonce <batcher_address> --rpc-url $L1_RPC
# 0 = ยังไม่เคย post · 1 = post batch แล้ว

# ดู safe_l2 ผ่าน op-node
cast rpc optimism_syncStatus --rpc-url http://localhost:9545 | jq '.safe_l2'
# safe_l2.number ต้องขยับขึ้น
```

ขั้นตอนที่ 6 — publish sync files + verify 3-way

6.1 ทำไมต้อง verify 3-way

GENESIS.JSON MISMATCH (บทที่ 4) เกิดเพราะ: - genesis.json ที่ใช้ init geth ≠ genesis.json ที่ publish - rollup.json genesis.l2.hash ≠ live block 0 hash ผลคือ follower re-init ผ่าน แต่ hash ไม่ตรง → `expected L2 genesis hash to match` → sync ล้มเหลว

6.2 คำสั่ง verify 3 ทาง

```
# ทาง 1: hash ใน genesis.json ที่ใช้ init (geth คำนวณ)
HASH_GETH=$(cast block 0 --rpc-url http://localhost:8545 --field hash)
```

```
# ทาง 2: hash ใน rollup.json
HASH_ROLLUP=$(jq -r '.genesis.l2.hash' ./op-stack/rollup.json)

# ทาง 3: live block 0
HASH_LIVE=$(cast rpc eth_getBlockByNumber '"0x0"' false \
  --rpc-url http://localhost:8545 | jq -r '.hash')

echo "geth init : $HASH_GETH"
echo "rollup.json: $HASH_ROLLUP"
echo "live block0: $HASH_LIVE"

# ทั้ง 3 ต้องตรงกัน
```

ถ้าไม่ตรง → หยุด ห้าม publish ค่ะ

6.3 publish

```
# serve ด้วย file server เบาๆ
mkdir -p ./sync-files
cp ./op-stack/genesis.json ./sync-files/
cp ./op-stack/rollup.json ./sync-files/
cd ./sync-files && python3 -m http.server 8181
```

announce URL ให้ fleet พร้อม hash อ้างอิงทั้ง 3 ทาง — follower ไม่ต้องเดาว่าไฟล์ถูกไหม

ขั้นตอนที่ 7 — follower join

7.1 follower รัน op-geth

```
# ดาวน์โหลด sync files
wget http://<sequencer-ip>:8181/genesis.json
wget http://<sequencer-ip>:8181/rollup.json
```

```
# init geth
docker run --rm --platform linux/amd64 \
  -v $(pwd)/data:/data \
  us-docker.pkg.dev/oplabs-tools-artifacts/images/op-geth:latest \
  init --datadir /data/geth /data/genesis.json
```

7.2 follower รัน op-node

```
op-node \
  --rollup.config ./rollup.json \
  --l1 $L1_RPC \
  --l2 http://localhost:8551 \
  --l2.jwt-secret ./jwt.txt \
  --p2p.listen.ip 0.0.0.0 \
  --p2p.listen.tcp 9003 \
  --p2p.bootnodes <sequencer-enr> \
  --rpc.addr 0.0.0.0 \
  --rpc.port 9545
```

ห้ามใส่ `--sequencer.enabled` ใน follower ค่ะ — follower derive จาก L1 batch และรับ gossip จาก sequencer เท่านั้น

7.3 verify follower sync

```
# ดู unsafe head ขยับผ่าน P2P gossip
cast rpc optimism_syncStatus \
  --rpc-url http://localhost:9545 | jq '{unsafe: .unsafe_l2.number,
safe: .safe_l2.number}'

# unsafe ขยับเร็ว = ได้รับ gossip
# safe ขยับช้า = derive จาก L1 batch (ตาม batcher post)
```

ขั้นตอนที่ 8 — L1→L2 deposit (ทดสอบ chain)

```
cast send \
  0x08D045e317f924A9428959AC557f198f95a7B519 \
  "depositTransaction(address,uint256,uint64,bool,bytes)" \
  <to_address> \
  1000000000000000 \
  100000 \
  false \
  0x \
  --value 0.001ether \
  --private-key $USER_PRIVATE_KEY \
  --rpc-url $L1_RPC
```

> OptimismPortal = `0x08D045e317f924A9428959AC557f198f95a7B519` — verify ได้ 2 ทาง: `rollup.deposit_contract_address` == `SystemConfig.optimismPortal()` ค่ะ

หลัง tx confirm บน L1 รอประมาณ 1-2 นาที บน L2 จะเห็น balance เพิ่ม

บทเรียนสรุป

Class 1 — Hex Error Class

> **ห้ามแปลง hex ด้วยสายตาหรือเดาเอง** ค่ะ

genesis timestamp ที่ผิดมาจากการคำนวณ hex ผิดมือ ผลคือ chain freeze ที่ block แรก กว่าจะ diagnose เสียเวลาและเสีย ETH re-deploy

กฎ:

```
# ใช้ printf เสมอ ไม่เดา
printf '%d\n' 0x6a360a34   # decimal
printf '%x\n' 1781926452   # hex
```

Class 2 — Source-of-Truth ต้องมีแหล่งเดียว

sync files ที่ publish ต้องมาจาก deploy directory จริง (`./op-stack/`) ค่ะ — ไม่ใช่ copy ที่ไหนสักที่หรือ serve จาก cache

กฎ: ก่อน publish ต้อง verify 3-way เสมอ ถ้าไม่ตรง = หยุดทันที ไม่ announce

Class 3 — Verify ก่อน Claim

Workshop-06 มีหลายเคสที่ “นึกว่า” แล้วผิด: - นึกว่า batcher funded → จริงโอนผิด address - นึกว่า genesis ถูก → timestamp hex error - นึกว่า sync files ตรง → 3 ทางไม่ตรงกัน

Don’t trust, always verify — check on-chain, check log, check hash ค่ะ

Class 4 — Kill Process ต้องรู้ว่า kill อะไร

server ใช้ user ร่วมกัน — kill process ที่ “ไม่รู้จัก” อาจเป็น sequencer ของคนอื่น ผลคือ chain stall ที่ reverse ไม่ได้ในทันที

กฎ: exclude ทั้ง PID group ของ service ก่อน kill ไม่ใช่เดาจาก port หรือ “ไม่มีชื่อ”

Checklist ก่อนประกาศ Chain Ready

```
[ ] genesis timestamp verify แล้ว (ไม่ใช่ estimate)
[ ] genesis hash verify 3-way แล้ว (geth init / rollup.json / live block0)
[ ] sequencer key address ตรงกับ SystemConfig
[ ] batcher address ตรงกับ rollup batcherAddr
[ ] batcher มี ETH บน L1 (nonce 0→1 หลังรัน)
[ ] op-node รัน --p2p.sequencer.key
[ ] follower เห็น unsafe_l2 ขยับ
[ ] L1→L2 deposit test ผ่าน
```

ถ้า tick ได้ครบ — chain พร้อมค่ะ 🐆

Forward-Looking — Chain ต่อไปจะไม่พลาดเดิม

Workshop-06 ได้สร้าง chain จริง แก้ปัญหาจริง และเสีย ETH จริง ค่ะ บทเรียนทุกข้อใน runbook นี้มาจากของจริง

Chain ต่อไปถ้าทำตาม checklist และ verify 3-way ก่อน announce — CLOCK-WEDGE, BATCHER-NOGGAS, P2P-SILENT, MISMATCH จะไม่เกิดซ้ำค่ะ

เพราะ **verify ไม่ใช่ optional — verify คือส่วนหนึ่งของ deploy** 🐆

บ๊องแบ๊ง — ลูกศิษย์ขยันแห่งทุ่งกว้าง · Oracle School Workshop-06 · 2026-06-20