PART 3 · ALPHA

3-01 · 개인 작업실 다음의 문제

프로젝트가 일을 소유하면 AI도 팀에 들어올 수 있습니다

naia-pj-adk · Discord Gateway · onmam-dev · aipol-lab

발표자 · 알파

루크를 돕지만 프로젝트의 권한은 아님

WHY NAIA-PJ-ADK

business-adk를 폐기하고 프로젝트 중심으로 다시 만들었습니다

1차
naia-adk + naia-business-adk
→ 같은 개선을 두 번 적용
→ 규칙·스킬 drift

현재
naia-adk　　개인의 장기 작업실
naia-pj-adk　팀 프로젝트의 권한·절차

naia-business-adk　폐기

PROJECT-OWNED WORK

프로젝트가 문맥·권한·증거를 직접 소유합니다

```
naia-pj-adk/
├─ AGENTS.md                # 프로젝트 진입점
├─ .agents/context/         # 공동 규칙
├─ .agents/requirements/    # 완료 조건
├─ projects/<name>/project.yaml
└─ GitHub issue             # scope · SHA · validation · rollback
```

> 개인 기억과 개인 비밀은 제외한다. 사람이 떠나도 프로젝트 정본은 남는다.

WHY DISCORD

왜 Discord인가 — 봇과 스레드 때문입니다

```
왜 Discord
  봇           게이트웨이 = Discord 봇
  서버·채널    프로젝트 단위 공간
  스레드       이슈 단위 대화
  역할         권한 구분 기본 내장
  — 슬랙은 일반인 문턱이 높다
```

```
naia-pj-adk가 일에 묶는 것
  issue      일의 단위
  thread     이슈마다 하나
  role       스레드에서 할 수 있는 일
  workspace  역할별 실행 자리
```

적용: onmam-dev 실증 · aipol-lab 검수 흐름 · naia-comm 예정

3-05 · 하나의 작업, 하나의 증거 사슬

ONE JOB, ONE EVIDENCE CHAIN

Discord 메시지 하나가 코드 영수증까지 이어집니다

issue → thread → workspace → commit → receipt

```
[Discord] "고객 도메인 연결이 안 돼요"
→ GitHub issue 생성 · 스레드 연결
→ 등록 workspace · 이슈 브랜치 작업
→ commit SHA · 테스트 결과 기록
→ 스레드 회신: 완료 + SHA + 확인 URL
```

> Discord가 정본이 아니라 GitHub issue가 정본이다.

같은 issue 번호·job ID·commit SHA가 Discord와 Git의 기록을 서로 대조하게 한다.

HUMAN-IN-THE-LOOP

사람이 끼어들 수 있게
연락 · 조정 · 실행을 나눕니다

연락

Discord 수신 · 결과 전달

조정

issue · 범위 · workspace binding

실행

SSH/Herdr · 코드 · 테스트

사람

중단 · 재개 · 승인 · 교체

> Discord가 끊겨도 실행은 이어진다 — 그리고 사람은 어느 층에든 개입한다: 중단·재개·승인·교체.

THREAD AS A WORKFLOW SURFACE

스레드 답장이 업무 버튼이 됩니다 — 단, 역할 권한 안에서만

```
aipol-lab 기사 검수 스레드

"공개 승인" 회신
  → candidate_id · state · repo version · thread 확인
  → 승인된 목록만 콘텐츠 Git 반영 시작
  → 검증 · 배포 · 실제 URL을 같은 run ID로 대조
  ✕ 임의의 Git merge / 다른 배포 권한
```

> 등록되지 않거나 workspace가 둘이면 추측하지 않고 멈춘다. 실패는 완료로 바꾸지 않는다.

backend는 셋 다 꽂히지만 무인 운영 검증은 아직 Codex뿐입니다

Codex

[기본 경로 · 계약 테스트]
balanced · recovery

Claude

[부분 검증]
read · reply · review

OpenCode

[호환 후보]
adapter · 공통 정본

실운영

현재 host·service receipt가 별도로 필요

HUMAN & DEPLOY BOUNDARY

"배포해 주세요" 한마디는 사전 검사 4개를 통과해야 실행됩니다

```
preflight must reject when:
  issue is not approved
  working tree is dirty
  integration branch is not an ancestor
  expected diff is absent

validation = exit code + body size + required marker
rollback   = artifact + exact command, prepared before change
```

> "배포했다"·"반영됐다"·"동작한다"를 각각 확인하고 — 최종 서명은 사람이 한다.

3-10 · 운영 영수증을 나누어 보기

OPERATIONS RECEIPT

살아 있음·진행 중·전달 완료를 따로 봅니다

```
naia-dcg status
naia-dcg jobs --active
naia-dcg job <id> --events
naia-dcg health-check --json
```

같은 job ID의 네 증거

- service heartbeat
- Gateway resume
- active job progress
- final delivery receipt

> naia-dcg = 로컬 운영 실행기 / !naia = Discord 대화 명령 · delivery_unknown은 자동 재전송하지 않는다.

THE WATCHDOG INCIDENT

감시기가 죽어 있던 34시간 — 고치고, 규칙으로 남겼습니다

끝난 thread 1개의 404
→ 전체 순회 중단
→ 사람 응답 대기 4건 누락
→ watchdog 실패 알림도 없음

수정
항목별 오류 격리
감시기 밖의 실패 알림
실제 담당 thread에 수신 확인
제한 횟수 뒤 blocked + takeover 위치

> 알림을 만들었다는 사실과, 사람이 행동할 곳에 도착했다는 사실은 다르다.

HONEST STATUS

여기까지의 상태판 — 검증 · 재확인 · 계약 · 계획

[검증]

코드 · 테스트 · 운영 로그 · receipt 연결

[재확인]

구현됐지만 발표일 장기 가동 증거 필요

[계약]

schema는 있으나 adapter 실행 증거 부족

[계획]

타 backend 동등성 · 분산형 메신저

한계도 하나: Discord를 쓰는 한 대화는 Discord 서버에도 남는다 — 분산형 메신저가 계획에 있는 이유.

> “마지막 발표는 루크가 직접 하겠습니다.
> 초대는 사람이 해야 하는 일이니까요.”