바이브 코딩으로 만든 서비스, 출시 후 30일 운영 체크리스트

> 클라우드·인프라 현업 개발자가 정리한 **배포 · 모니터링 · 비용 · 백업 · 보안** 5영역 50개 항목
>
> - 클로드 코드에게 점검 맡기는 프롬프트 + 장애 대응 런북 템플릿

- 만든 사람: cronlog (현업 개발자 · 얼굴 없는 기록 계정)
- 버전: v1.0 (2026-11) · 무료 · 출처 표시하면 자유롭게 공유해도 돼요
- 최신본: cronlog 블로그 자료실 `[링크: 다운로드 페이지 URL]`

들어가며 — 왜 「출시 후」 체크리스트인가요

바이브 코딩 덕분에 서비스를 「만드는」 건 정말 쉬워졌어요. 주말 이틀이면 로그인 되고 결제 붙은 웹앱이 나와요.

그런데 운영 일을 하면서 보면, 사고는 대부분 만들 때가 아니라 **띄워 놓은 뒤**에 나요.

- 아무도 모르게 서비스가 죽어 있었다
- 청구서를 보고 나서야 비용이 튄 걸 알았다
- DB가 날아갔는데 백업이 없었다
- 키가 공개 저장소에 올라가 있었다

이 자료는 「배포 버튼 누른 다음 30일」을 버티는 데 필요한 최소한을 모았어요. 대기업 SRE 팀 기준이 아니라 **혼자 또는 두세 명이 운영하는 서비스** 기준이에요.

[경험: 직접 운영하던 개인 프로젝트(또는 지인 서비스)에서 출시 후에 처음 터진 문제 하나 — 무엇이 · 언제 · 어떻게 알았는지 2~3줄. 회사·사내 시스템 이야기 제외]

이런 분께 맞아요

- 클로드 코드·커서 등으로 만든 서비스를 실제로 띄웠거나 곧 띄울 분
- 프론트는 자신 있는데 서버·클라우드는 「일단 되니까 둔」 분
- 사용자가 생기기 시작해서 슬슬 불안해진 분

이런 분께는 안 맞아요

- 이미 팀에 인프라 담당자가 있고 SLA·온콜 체계가 있는 경우
- 규제 산업(금융·의료) 수준의 보안 감사가 필요한 경우 → 전문 점검을 받으세요

이 자료 쓰는 법

1. **먼저 「출시 당일 최소 10개」(1장)만 하세요.** 30분~2시간이면 끝나요.
2. 나머지 5영역은 한 번에 다 하지 말고, **매주 한 영역씩** 보세요(8장 30일 루틴).
3. 항목마다 이렇게 생겼어요.
    - **왜** — 안 하면 생기는 일
    - **확인** — 지금 상태를 보는 방법
    - **통과 기준** — 체크해도 되는 조건
4. 각 장 끝에 **클로드 코드에게 맡기는 점검 프롬프트**가 있어요. 단, AI의 답은 「후보 목록」이에요. 마지막 확인은 사람이 직접 하세요.

표시 규칙

- `[필수]` 사용자가 1명이라도 있으면 해야 하는 것
- `[권장]` 사용자가 늘면 해야 하는 것
- 특정 서비스(Vercel, Supabase, AWS 등) 이름은 예시예요. 무료 한도·요금·메뉴 위치는 자주 바뀌니까 **반드시 공식 문서에서 현재 값을 확인**하세요. 이 자료에는 일부러 한도 숫자를 적지 않았어요.

1장. 출시 당일 최소 10개

시간이 없으면 이것만 하세요. 나머지는 이 10개를 조금씩 넓히는 거예요.

#	할 일	영역	걸리는 시간(대략)
1	코드·git 기록에 API 키·비밀번호가 없는지 확인	보안	10분
2	클라우드·AI API 콘솔에 **예산 알림** 걸기	비용	10분
3	AI API(LLM) 키에 **사용 한도** 걸기, 브라우저 코드에 키 없는지 확인	비용·보안	10분
4	외부 업타임 체크 1개 + 알림을 휴대폰으로	모니터링	10분
5	DB 자동 백업 켜져 있는지 확인	백업	5분
6	백업에서 **실제로 한 번 복구**해 보기(별도 DB로)	백업	30분
7	다른 사람 데이터가 내 계정으로 보이지 않는지 확인	보안	20분
8	직전 버전으로 되돌리는 방법을 한 줄로 적어 두기	배포	10분
9	에러가 나면 어디서 보는지 정해 두기(로그·에러 추적)	모니터링	15분
10	관리자 페이지·디버그 모드·테스트 계정 닫기	보안	10분

[경험: 이 10개 중 직접 운영하면서 가장 자주 빠뜨리는 걸 본 항목 하나와 이유]

2장. 배포 — 「다시 띄울 수 있는가」

배포의 목표는 「한 번 올리기」가 아니라 **언제든 같은 결과로 다시 올리고, 문제가 생기면 되돌릴 수 있는 것**이에요.

D-1 `[필수]` 배포 방법이 기록돼 있다

- **왜**: 「그때 어떻게 올렸더라」가 장애를 길게 만들어요. AI랑 대화하며 올렸다면 더 그래요. 대화 기록은 금방 묻혀요.
- **확인**: README나 `DEPLOY.md`에 배포 명령·필요한 환경변수 이름·도메인 설정이 적혀 있나요?
- **통과 기준**: 이 문서만 보고 새 환경에 다시 배포할 수 있다.

D-2 `[필수]` 되돌리는 방법을 안다

- **왜**: 새 버전이 깨졌을 때 고치는 것보다 되돌리는 게 거의 항상 빨라요.
- **확인**: 호스팅 서비스의 「이전 배포로 롤백」 기능 위치, 또는 git 태그로 되돌리는 명령을 알고 있나요?
- **통과 기준**: 직전 버전으로 되돌리는 걸 **한 번 실제로 해 봤다**.

D-3 `[필수]` 개발용과 운영용이 분리돼 있다

- **왜**: 테스트하다가 운영 DB를 지우는 사고는 생각보다 흔해요. AI 에이전트에게 「DB 초기화해줘」라고 했는데 운영 DB에 연결돼 있으면 끝이에요.
- **확인**: 로컬 `.env`의 DB 주소가 운영 DB와 다른가요? 운영 키가 개발 PC에 그대로 있나요?
- **통과 기준**: 개발 환경에서 운영 DB에 바로 쓸 수 없다(최소한 다른 계정·다른 키).

D-4 `[필수]` 환경변수 목록이 정리돼 있다

- **왜**: 키를 하나 빠뜨리면 배포는 성공하는데 특정 기능만 조용히 죽어요.
- **확인**: `.env.example`(값 없이 이름만)이 있고 실제 사용 중인 변수와 일치하나요?
- **통과 기준**: 필요한 변수가 빠지면 앱이 **시작할 때** 에러를 낸다(조용히 넘어가지 않는다).

D-5 `[필수]` HTTPS와 도메인 갱신

- **왜**: 인증서·도메인 만료는 서비스 전체를 한 번에 멈춰요. 그리고 꼭 바쁠 때 와요.
- **확인**: 인증서 자동 갱신이 켜져 있나요? 도메인 자동 연장과 결제 카드 유효기간은요?
- **통과 기준**: 도메인 자동 연장 ON · 만료 알림 받을 이메일이 실제로 보는 주소.

D-6 `[권장]` DB 구조 변경(마이그레이션)은 파일로 남긴다

- **왜**: DB 콘솔에서 직접 컬럼을 바꾸면 다음 배포나 복구 때 구조가 어긋나요.
- **확인**: 스키마 변경이 코드 저장소에 마이그레이션 파일로 있나요?
- **통과 기준**: 빈 DB에 마이그레이션만 돌려도 운영과 같은 구조가 나온다.

D-7 `[권장]` 배포 전 자동 확인 1단계

- **왜**: 빌드가 깨진 채로 올라가는 걸 막는 최소한의 안전망이에요.
- **확인**: 푸시하면 빌드·타입 체크·간단한 테스트가 자동으로 도나요(GitHub Actions 등)?
- **통과 기준**: 빌드 실패 시 배포가 진행되지 않는다.

D-8 `[권장]` 의존성 버전이 고정돼 있다

- **왜**: 같은 코드인데 어제는 되고 오늘은 안 되는 이유 1순위예요.
- **확인**: lock 파일(`package-lock.json`, `pnpm-lock.yaml`, `poetry.lock` 등)이 커밋돼 있나요?
- **통과 기준**: lock 파일이 저장소에 있고 배포가 그걸 사용한다.

D-9 `[권장]` 배포 직후 확인 루틴

- **왜**: 「배포 성공」 표시는 「서비스 정상」이 아니에요.
- **확인**: 배포 후 로그인 → 핵심 기능 1개 → 결제(있다면)를 직접 눌러 보나요?
- **통과 기준**: 3분짜리 수동 확인 순서가 적혀 있다.

D-10 `[권장]` 배포 시간을 정한다

- **왜**: 금요일 밤 배포 → 주말 장애는 운영하는 사람들 사이의 오래된 농담이에요. 본업이 있으면 더 위험해요.
- **통과 기준**: 「문제 생기면 바로 볼 수 있는 시간」에만 배포한다.

[경험: 본인의 배포 시간 규칙(예: 평일 퇴근 후 몇 시까지만) 또는 금요일 배포로 고생한 일반화된 사례]

클로드 코드 점검 프롬프트 — 배포

```
이 저장소의 배포 준비 상태를 점검해줘. 코드는 수정하지 말고 보고만 해.
1. 배포 방법이 문서화돼 있는지(README, DEPLOY.md 등) 찾아보고, 없으면 지금 코드 기준으로 필요한 배포 단계를 초안으로 적어줘.
2. 코드에서 읽는 환경변수를 전부 찾아서 목록으로 만들고, .env.example과 비교해 빠진 것/안 쓰는 것을 표로 보여줘. 값은 절대 출력하지 마.
3. 필수 환경변수가 없을 때 앱이 시작 단계에서 실패하는지, 조용히 넘어가는지 알려줘.
4. lock 파일과 마이그레이션 파일이 있는지, DB 스키마가 코드로 관리되는지 알려줘.
5. 롤백할 때 걸릴 만한 것(되돌릴 수 없는 마이그레이션 등)이 있으면 짚어줘.
결과는 "통과 / 확인 필요 / 없음" 3단계 표로.
```

3장. 모니터링 — 「죽으면 내가 먼저 아는가」

목표는 단순해요. **사용자보다 내가 먼저 알기.** 사용자가 DM으로 알려줄 때는 이미 여러 명이 떠난 뒤예요.

M-1 `[필수]` 외부에서 보는 업타임 체크

- **왜**: 서버 안에서 도는 감시는 서버와 같이 죽어요.

- **확인**: 외부 업타임 모니터링 서비스(무료 플랜이 있는 곳 많아요)로 메인 페이지와 헬스 체크 주소를 몇 분 간격으로 보고 있나요?
- **통과 기준**: 서비스를 일부러 내렸을 때 알림이 실제로 왔다.

M-2 `[필수]` 알림이 「보는 곳」으로 온다

- **왜**: 아무도 안 여는 메일함으로 가는 알림은 없는 거랑 같아요.
- **통과 기준**: 휴대폰 푸시·메신저 등 실제로 확인하는 채널. 알림이 너무 많으면 결국 끄게 되니 **처음엔 「서비스 다운」 하나만**.

M-3 `[필수]` 헬스 체크 주소가 진짜를 본다

- **왜**: 「200 OK」만 돌려주는 헬스 체크는 DB가 죽어도 정상이라고 해요.
- **확인**: `/health` 같은 주소가 DB 연결 등 핵심 의존성을 가볍게 확인하나요?
- **통과 기준**: DB 연결이 끊기면 헬스 체크가 실패한다. 단, 응답에 내부 정보(버전·호스트명·에러 원문)는 넣지 않는다.

M-4 `[필수]` 에러를 모아서 본다

- **왜**: 콘솔에 찍히고 사라지는 에러는 재현이 안 돼요.
- **확인**: 에러 추적 도구(무료 플랜 있는 곳 다수) 또는 호스팅의 로그 화면에서 최근 24시간 에러를 볼 수 있나요?
- **통과 기준**: 새로운 종류의 에러가 생기면 알 수 있다.

M-5 `[필수]` 로그에 비밀이 찍히지 않는다

- **왜**: 디버깅하려고 요청 전체를 로그로 찍으면 비밀번호·토큰·개인정보가 같이 남아요. 로그 서비스는 보통 접근 관리가 느슨해요.
- **확인**: 로그에서 `password`, `token`, `authorization`, 이메일·전화번호를 검색해 보세요.
- **통과 기준**: 민감 값은 가려지거나 아예 안 찍힌다.

M-6 `[권장]` 핵심 지표 3개만 정한다

- **왜**: 대시보드가 화려하면 정작 이상 신호를 못 봐요.
- **예**: ① 응답 실패율(5xx 비율) ② 느린 응답 ③ 가입·결제 같은 핵심 행동 수(갑자기 0이 되면 어딘가 깨진 것)
- **통과 기준**: 이 3개를 한 화면에서 볼 수 있다.

M-7 `[권장]` 예약 작업(cron)이 「안 돈 것」도 안다

- **왜**: 매일 도는 백업·정산·메일 발송 작업은 실패보다 **아예 안 돈 것**을 늦게 알아요.
- **확인**: 작업이 끝날 때 신호를 보내고, 신호가 안 오면 알림이 오는 방식(하트비트 체크)을 쓰나요?
- **통과 기준**: 예약 작업을 일부러 꺼 뒀을 때 알림이 왔다.

M-8 [권장] 외부 API 장애에 대비한다

- **왜**: AI API·결제·로그인 제공자가 느려지면 내 서비스도 같이 멈춰요.
- **확인**: 외부 호출에 타임아웃이 있나요? 실패하면 사용자에게 뭐가 보이나요?
- **통과 기준**: 외부 API가 죽어도 내 서비스 전체가 멈추지 않고, 사용자에게 알아들을 수 있는 메시지가 나온다.

M-9 [권장] 디스크·DB 용량

- **왜**: 로그나 업로드 파일이 쌓여 디스크가 꽉 차면 DB까지 같이 멈출 수 있어요.
- **통과 기준**: 용량이 일정 비율을 넘으면 알림이 온다(또는 매주 직접 본다).

M-10 [권장] 상태 공지 방법

- **왜**: 장애 때 침묵하면 사용자는 「망했나?」라고 생각해요.
- **통과 기준**: 장애 시 공지할 곳(서비스 내 배너·SNS·상태 페이지 중 하나)이 정해져 있다.

[경험: 본인이 실제로 쓰는(또는 개인 프로젝트에 붙여 본) 업타임·에러 추적 도구 이름과 고른 이유 1~2줄 — 제휴 아님을 확인]

클로드 코드 점검 프롬프트 — 모니터링

```
이 서비스의 모니터링 준비 상태를 점검해줘. 코드는 수정하지 말고 보고만 해.
1. 헬스 체크 엔드포인트가 있는지, 있다면 DB 등 핵심 의존성을 확인하는지, 응답에 내부 정보가 노출되는지 알려줘.
2. 로그를 찍는 코드 중에 비밀번호, 토큰, Authorization 헤더, 이메일, 전화번호 같은 민감 값이 그대로 남을 수 있는 곳을 파일:줄 단위로 찾아줘.
3. 외부 API 호출(결제, AI API, 인증 등) 목록과 각각 타임아웃·재시도·실패 처리가 있는지 표로 보여줘.
4. 예약 작업(cron, 스케줄러)이 있다면 실패나 미실행을 알 수 있는 장치가 있는지 알려줘.
5. 에러가 사용자에게 어떻게 보이는지(스택 트레이스 노출 여부) 확인해줘.
```

4장. 비용 — 「청구서를 보고 놀라지 않는가」

바이브 코딩 서비스에서 제일 아픈 사고는 보안보다 **비용**인 경우가 많아요. 보안 사고는 「혹시」 지만, 비용은 설정 하나 잘못하면 「반드시」 와요.

먼저 꼭 알아야 할 한 가지:

> **대부분의 클라우드 「예산 알림」은 알림일 뿐, 사용을 멈춰 주지 않아요.** 알림을 받고 사람이 끄기 전까지 요금은 계속 쌓여요. 자동으로 멈추게 하려면 별도 설정(지출 한도, 자동 중지 등)이 필요하고, 서비스마다 지원 여부가 달라요.

C-1 `[필수]` 모든 유료 계정에 예산 알림

- **대상**: 클라우드(AWS·GCP·Azure 등), 호스팅(Vercel·Netlify·Cloudflare 등), DB(Supabase·PlanetScale 등), AI API(Anthropic·OpenAI 등)
- **통과 기준**: 월 예상액의 50% · 100%에서 알림. 각 계정마다 따로.

C-2 `[필수]` AI API에 하드 한도

- **왜**: LLM 호출은 사용자 수가 아니라 **호출 수 × 길이**로 과금돼요. 무한 루프 한 번, 키 유출 한 번이면 하룻밤에 한 달 치가 나갈 수 있어요.
- **확인**: AI API 콘솔에서 월 지출 한도·사용량 제한을 걸 수 있는지 보세요. 키를 서비스별로 나눴나요?
- **통과 기준**: 한도를 넘으면 호출이 **멈춘다**(알림만이 아니라).

C-3 `[필수]` 사용자당 호출 제한

- **왜**: 한 사용자(또는 봇)가 AI 기능을 무한정 누르면 그 비용은 전부 내 몫이에요.
- **확인**: 로그인 없이 AI 기능을 쓸 수 있나요? 사용자당·IP당 횟수 제한이 있나요?
- **통과 기준**: 1인당 하루 사용량 상한이 있고, 비로그인 사용은 더 낮게 제한된다.

C-4 `[필수]` 무료 한도를 넘으면 무슨 일이 생기는지 안다

- **왜**: 어떤 서비스는 무료 한도를 넘으면 **멈추고**, 어떤 서비스는 **자동으로 과금**해요. 반대로 생각하고 있으면 둘 다 사고예요.
- **확인**: 쓰는 서비스마다 「한도 초과 시 동작」을 공식 문서에서 확인하고 한 줄씩 적어 두세요.
- **통과 기준**: 서비스별로 「초과 시 멈춤 / 과금」을 안다.

C-5 `[필수]` 반복 실행되는 것을 의심한다

- **비용 폭탄 단골 패턴**
  - 서버리스 함수가 자기 자신을 다시 호출(이벤트 → 저장 → 이벤트 …)
  - 프론트에서 `useEffect` 같은 곳의 의존성 실수로 API를 계속 호출
  - 실패한 작업이 무한 재시도
  - 크롤러·봇이 이미지 변환·AI 생성 같은 비싼 주소를 반복 호출
- **통과 기준**: 재시도에 횟수 제한과 간격이 있고, 비싼 작업은 캐시하거나 로그인 뒤에만 열려 있다.

C-6 `[권장]` 나가는 트래픽(전송량)을 본다

- **왜**: 저장은 싸도 **내보내는 데이터**가 비싼 서비스가 많아요. 큰 이미지·동영상을 원본 그대로 서빙하면 여기서 튀어요.
- **통과 기준**: 이미지는 적당한 크기로 줄이고, 정적 파일은 CDN·캐시를 거친다.

C-7 `[권장]` 잊힌 리소스 정리

- **왜**: 테스트용 서버·DB·스냅샷·고정 IP는 안 써도 요금이 나와요. 특히 다른 리전에 만든 건 잘 안 보여요.
- **확인**: 결제 화면에서 「서비스별·리전별」로 나눠 보세요.

- **통과 기준**: 모든 리소스의 용도를 설명할 수 있다. 모르는 건 이름표(태그)를 달거나 지운다.

C-8 `[권장]` 로그 보관 기간

- **왜**: 로그를 무기한 보관하면 조용히 비용이 늘어요.
- **통과 기준**: 보관 기간이 정해져 있다(예: 필요한 기간만).

C-9 `[권장]` 월 고정비 표 한 장

- **통과 기준**: 도메인·호스팅·DB·AI API·이메일 발송 등 매달 나가는 돈이 한 표에 있다. 「서비스가 벌어야 하는 최소 금액」이 보인다.

C-10 `[권장]` 결제 카드 분리

- **왜**: 개인 카드 한 장에 다 물려 있으면 사고 때 어디서 새는지 찾기 어렵고, 한도 설정도 어려워요.
- **통과 기준**: 서비스용 결제 수단(한도 낮은 카드·가상 카드 등)이 따로 있다.

[경험: 본인이 실제로 쓰는 월 고정비 구성 — 항목과 실제 금액(대략이라도 실제 값), 또는 「비용이 튄 적 있다면」 원인과 금액. 없으면 0원이라고 그대로]

클로드 코드 점검 프롬프트 — 비용

```
이 저장소에서 비용 사고로 이어질 수 있는 코드를 찾아줘. 코드는 수정하지 말고 보고만 해.
1. 외부 유료 API(특히 AI/LLM API) 호출 위치를 전부 찾고, 각각 사용자 인증·사용자별 횟수 제한·최대 토큰/길이 제한이 있는지 표로 보여줘.
2. 재시도 로직을 찾아서 최대 횟수와 대기 간격이 있는지 확인해줘.
3. 프론트엔드에서 렌더링/상태 변경 때문에 API가 반복 호출될 수 있는 곳을 찾아줘.
4. 서버리스 함수나 이벤트 트리거가 스스로를 다시 실행시킬 수 있는 구조가 있는지 봐줘.
5. 이미지/파일을 원본 크기로 내려보내는 곳, 캐시 헤더가 없는 정적 파일 응답이 있는지 알려줘.
위험도(높음/중간/낮음)와 함께 파일:줄로.
```

5장. 백업·복구 — 「날아가도 돌아오는가」

백업에서 중요한 건 「백업이 있다」가 아니라 **「복구해 봤다」**예요. 복구 안 해 본 백업은 백업이라고 부르기 어려워요.

B-1 `[필수]` DB 자동 백업이 켜져 있다

- **확인**: 관리형 DB라면 백업 설정 화면에서 주기·보관 기간을 보세요. **무료 플랜은 백업이 없거나 짧은 경우가 있어요** → 공식 문서 확인.
- **통과 기준**: 자동 백업이 켜져 있고, 보관 기간을 안다.

B-2 `[필수]` 복구 리허설을 했다

- **확인**: 백업을 **운영 DB가 아닌 새 DB**로 복구하고, 데이터가 맞는지 몇 개 조회해 보세요.

- **통과 기준**: 복구에 걸린 시간을 안다. 그 시간이 「복구 목표 시간」이에요.

B-3 `[필수]` 백업이 같은 곳에만 있지 않다

- **왜**: 같은 계정 안에만 백업이 있으면 계정이 잠기거나 실수로 프로젝트를 지울 때 같이 사라져요.
- **통과 기준**: 최소 주 1회는 다른 저장소(다른 계정·다른 서비스)에 사본이 있다. (흔히 말하는 3-2-1 원칙: 사본 3개 · 매체 2종 · 1개는 외부)

B-4 `[필수]` 사용자 업로드 파일도 백업 대상이다

- **왜**: DB만 백업하고 이미지·첨부 파일 저장소는 빠뜨리는 경우가 많아요. DB엔 파일 주소만 있는데 파일이 없으면 복구해도 깨져요.
- **통과 기준**: 파일 저장소의 버전 관리·백업 여부를 안다.

B-5 `[필수]` 「복구에 필요한 것」 목록

- **왜**: 데이터만 있다고 서비스가 돌아오지 않아요. 환경변수·도메인 설정·외부 서비스 설정(OAuth 콜백 주소, 웹훅 주소)도 필요해요.
- **통과 기준**: 서비스를 처음부터 다시 띄울 때 필요한 항목 목록이 있다. **비밀값 자체는 이 목록에 쓰지 말고** 비밀번호 관리자 등 안전한 곳에 둔다.

B-6 `[권장]` 삭제는 「바로 삭제」가 아니게

- **왜**: 사용자가 실수로 지운 데이터를 살려 달라는 요청은 꼭 와요. 운영자가 실수로 지우는 일도요.
- **통과 기준**: 중요한 데이터는 일정 기간 「삭제 표시」 후 실제 삭제(소프트 삭제). 단, 개인정보 삭제 요청은 법적 기한 안에 실제 삭제.

B-7 `[권장]` 위험한 명령에 안전장치

- **왜**: AI 에이전트나 내가 `DROP`, `DELETE ... WHERE` 빠뜨림, `rm -rf`를 운영에 실행하는 사고.
- **확인**: 운영 DB 접속 계정이 개발 때 쓰는 계정과 같은가요? AI 코딩 도구가 운영 DB에 접근할 수 있나요?
- **통과 기준**: AI 도구에는 운영 DB 쓰기 권한을 주지 않는다. 클로드 코드라면 위험 명령을 막는 권한 설정·훅을 쓴다.

B-8 `[권장]` 코드 저장소도 백업이다

- **통과 기준**: 원격 저장소(GitHub 등)에 올라가 있고, 로컬에만 있는 커밋이 없다.

B-9 `[권장]` 계정 복구 수단

- **왜**: 클라우드·도메인·DB 계정에 로그인 못 하면 백업이 있어도 못 써요.
- **통과 기준**: 2단계 인증 + 복구 코드를 안전한 곳에 보관. 담당자가 나 혼자라면 「내가 연락이 안 될 때」 방법도 생각해 둔다.

B-10 `[권장]` 복구 순서를 적어 둔다

- **통과 기준**: 6장 런북에 「DB 복구」 절차가 3~5줄로 있다.

[경험: 직접 해 본 복구 리허설(또는 실제 복구) — 무엇을, 걸린 시간, 예상과 달랐던 점]

클로드 코드 점검 프롬프트 — 백업

```
이 서비스를 처음부터 다시 띄워야 한다고 가정하고 "복구에 필요한 것" 목록을 만들어줘. 코드는 수정하지 말고 보고만 해.
1. 데이터 저장소(DB, 파일 저장소, 캐시, 큐)를 전부 찾아서 목록으로.
2. 코드에서 참조하는 외부 서비스 설정(OAuth 콜백 URL, 웹훅 URL, 도메인, CORS 허용 주소 등)을 목록으로.
3. 필요한 환경변수 이름 목록(값은 출력하지 마).
4. 하드 삭제(DELETE, DROP, 파일 삭제)가 일어나는 코드 위치와, 복구 불가능한 작업이 있는지.
5. 마이그레이션이 빈 DB에서 처음부터 끝까지 돌 수 있는 구조인지.
마지막에 "복구 런북 초안"을 단계별로 적어줘.
```

6장. 보안 — 「남의 데이터가 내 화면에 보이지 않는가」

여기서는 **방어·점검**만 다뤄요. 그리고 「배포 전 보안 점검」을 한 번 했더라도, 운영하면서 기능이 늘면 다시 봐야 해요.

S-1 `[필수]` 비밀값은 코드 밖에

- **확인**: 저장소 전체와 **git 기록**에서 키 형태 문자열을 검색하세요. 지금 파일에서 지웠어도 예전 커밋에 남아 있으면 노출된 거예요.
- **통과 기준**: 코드·커밋 기록에 비밀값 없음. `.env`는 `.gitignore`에 있음. GitHub의 시크릿 스캐닝·푸시 보호 같은 기능을 켰다.
- **이미 올라갔다면**: 커밋을 지우는 것보다 **키 폐기·재발급이 먼저**예요.

S-2 `[필수]` 브라우저로 내려가는 코드에 비밀값 없음

- **왜**: 프론트엔드 코드는 누구나 볼 수 있어요. 프레임워크마다 「브라우저에 공개되는 환경변수」 접두어(`NEXT_PUBLIC_`, `VITE_` 등)가 있는데, 여기에 비밀 키를 넣는 실수가 정말 흔해요.
- **확인**: 배포된 사이트에서 개발자 도구 → 소스에서 키 형태 문자열 검색.
- **통과 기준**: AI API 키·결제 비밀 키·DB 관리자 키는 서버에만 있다.

S-3 `[필수]` 남의 데이터 접근 차단(권한 확인)

- **왜**: 「로그인했는지」만 보고 「이 데이터의 주인인지」는 안 보는 코드가 AI 생성 코드에서 자주 보여요. 주소의 숫자 하나만 바꾸면 남의 주문·메시지가 보이는 식이에요.
- **확인**: 테스트 계정 2개를 만들어서, A로 만든 데이터의 주소·ID를 B 계정으로 열어 보세요(내 서비스, 내 테스트 계정으로만).
- **통과 기준**: B로는 A의 데이터를 보거나 고치거나 지울 수 없다. API 직접 호출로도.

S-4 `[필수]` DB 행 단위 보안(해당 시)

- **왜**: Supabase·Firebase처럼 브라우저가 DB에 직접 붙는 구조는 **DB 규칙이 곧 보안**이에요. 규칙이 꺼져 있거나 「테스트 모드」로 열려 있으면 공개 키만으로 데이터를 읽고 쓸 수 있어요.
- **확인**: 모든 테이블에 행 단위 보안(RLS)이 켜져 있고 정책이 있나요? Firebase 보안 규칙이 전체 허용이 아닌가요?
- **통과 기준**: 공개 키로 다른 사용자 데이터 조회가 안 된다(S-3 방법으로 확인).

S-5 `[필수]` 관리자·디버그 기능 닫기

- **확인**: `/admin`, 디버그 모드, API 문서 페이지, 테스트용 계정·백도어 코드가 운영에 남아 있나요?
- **통과 기준**: 운영에서 디버그 OFF, 관리자 기능은 권한 확인 + 가능하면 별도 보호.

S-6 `[필수]` 에러 메시지에 내부 정보 없음

- **통과 기준**: 사용자에게는 「문제가 생겼어요」 수준만. 스택 트레이스·SQL·파일 경로는 로그에만.

S-7 `[권장]` 로그인·가입·비밀번호 재설정에 횟수 제한

- **왜**: 무차별 대입과 스팸 가입, 그리고 메일·문자 발송 비용(C-3과 연결).
- **통과 기준**: 짧은 시간에 반복 시도하면 막히거나 지연된다.

S-8 `[권장]` CORS·쿠키 설정

- **통과 기준**: CORS 허용 출처가 `*`가 아니라 내 도메인. 로그인 쿠키에 `HttpOnly`, `Secure`, 적절한 `SameSite`.

S-9 `[권장]` 의존성 취약점 알림

- **통과 기준**: 저장소에 의존성 취약점 알림(Dependabot 등)을 켰고, 한 달에 한 번은 본다.

S-10 `[권장]` 업로드 파일 제한

- **통과 기준**: 파일 크기·형식 제한이 서버에서 확인된다. 업로드한 파일이 실행되거나 내 도메인에서 HTML로 열리지 않는다.

[경험: 운영하면서 본 「권한 확인 누락」류 문제를 회사·시스템이 특정되지 않게 일반화한 사례 1개]

클로드 코드 점검 프롬프트 — 보안

```
이 저장소를 방어 관점에서 점검해줘. 코드는 수정하지 말고, 발견한 비밀값은 값 대신 위치만 알려줘.
1. 코드와 git 기록에 비밀값처럼 보이는 문자열이 있는지. 프론트엔드에 공개되는 환경변수(NEXT_PUBLIC_, VITE_ 등)에 비밀 키가 들어가는지.
2. 데이터를 조회/수정/삭제하는 모든 API에서 "로그인 여부"뿐 아니라 "이 데이터의 주인인지"를 확인하는지 표로 정리해줘. 확인이 없는 곳은 위험도 높음으로.
3. Supabase/Firebase 등을 쓴다면 RLS나 보안 규칙이 모든 테이블/컬렉션에 적용돼 있는지.
```

```
4. 운영에 남아 있으면 안 되는 것: 디버그 모드, 관리자 경로, 테스트 계정, 하드코딩된 우회 조건.
5. 로그인/가입/재설정의 횟수 제한, CORS 허용 출처, 쿠키 속성, 파일 업로드 검증.
위험도 순으로 파일:줄과 함께. 그리고 "사람이 직접 확인해야 할 것"을 따로 정리해줘.
```

> AI 점검은 놓치는 게 있어요. 특히 S-3(남의 데이터 접근)은 **테스트 계정 2개로 직접 눌러 보는 것**이 가장 확실해요.

7장. 장애 났을 때 — 30분 런북 템플릿

장애는 대부분 본업 회의 중이거나 자려고 누웠을 때 와요. 그때 머리로 생각하지 않도록 **미리 적어 두는 것**이 런북이에요. 아래를 복사해서 내 서비스에 맞게 채우세요.

```
# [서비스명] 장애 런북

## 0. 먼저 (5분)
- [ ] 정말 장애인가? 다른 네트워크(휴대폰 데이터)로 접속해 본다
- [ ] 언제부터? 업타임 알림 시각 / 최근 배포 시각 확인
- [ ] 영향 범위: 전체 / 특정 기능 / 특정 사용자
- [ ] 공지: [공지 위치]에 "확인 중" 한 줄

## 1. 최근 바뀐 것부터 (10분)
- [ ] 최근 24시간 배포가 있었다 → 바로 롤백: [롤백 방법 한 줄]
- [ ] 환경변수·설정을 바꿨다 → 원복
- [ ] 외부 서비스 상태 페이지 확인: [호스팅] [DB] [AI API] [결제]

## 2. 어디가 아픈가 (10분)
- [ ] 에러 로그 위치: [주소]
- [ ] DB 연결 / 디스크 용량 / 메모리
- [ ] 비용 한도 도달로 서비스가 멈춘 것은 아닌가 (C-4)

## 3. 데이터가 깨졌다면
- [ ] 쓰기 막기(점검 모드): [방법]
- [ ] 복구 절차: [B-10의 3~5줄]
- [ ] 복구 후 확인: 로그인 → 핵심 기능 → 결제

## 4. 끝나고 (다음 날)
- [ ] 무엇이 / 언제 / 왜 / 어떻게 알았나 / 어떻게 고쳤나 — 5줄 기록
- [ ] 재발 방지 1개만: 체크리스트에 항목 추가 또는 알림 추가
- [ ] 사용자 공지: 원인과 조치 (개인정보 영향이 있으면 관련 법령에 따른 신고·통지 의무 확인)
```

[경험: 본인 런북에 실제로 적어 두는(또는 적어 두면 좋았을) 항목 하나]

8장. 30일 운영 루틴

한 번에 다 하면 지쳐요. 이렇게 나눠 보세요. 본업 있는 사람 기준으로 **주 30분~1시간**이에요.

시점	할 일	관련 항목
출시 당일	1장 최소 10개	1장
1주차	배포 + 모니터링 장 전체	D-1~~10~~ · M-1~~10~~
2주차	비용 장 전체 + 첫 주 청구 내역 확인	C-1~10
3주차	백업 장 + 복구 리허설	B-1~10
4주차	보안 장 + 테스트 계정 2개로 권한 점검	S-1~10
30일차	런북 완성 + 이 표를 다시 보기	7장

그 뒤로 계속

- **매일(1분)**: 알림 왔는지만 본다
- **매주(10분)**: 에러 목록 새 항목 · 비용 추이 · 디스크 용량
- **매월(30분)**: 청구서 서비스별 확인 · 의존성 취약점 알림 · 잊힌 리소스 정리 · 백업 사본 확인
- **분기(1시간)**: 복구 리허설 다시 · 테스트 계정 권한 점검 다시 · 도메인·카드 만료일

[경험: 본인이 실제로 지키는 주간·월간 루틴 중 하나(요일·시간대 등)]

부록 A. 한 장 체크리스트 (출력용)

배포

- [ ] D-1 배포 방법 문서
- [ ] D-2 롤백 해 봄
- [ ] D-3 개발/운영 분리
- [ ] D-4 환경변수 목록 · 누락 시 시작 실패
- [ ] D-5 HTTPS·도메인 자동 갱신
- [ ] D-6 마이그레이션 파일
- [ ] D-7 배포 전 자동 빌드 확인
- [ ] D-8 lock 파일
- [ ] D-9 배포 직후 3분 확인
- [ ] D-10 배포 시간 규칙

모니터링

- [ ] M-1 외부 업타임 체크
- [ ] M-2 보는 채널로 알림
- [ ] M-3 진짜 헬스 체크

- [ ] M-4 에러 수집
- [ ] M-5 로그에 비밀 없음
- [ ] M-6 핵심 지표 3개
- [ ] M-7 예약 작업 미실행 감지
- [ ] M-8 외부 API 타임아웃
- [ ] M-9 디스크·DB 용량
- [ ] M-10 장애 공지 위치

비용

- [ ] C-1 모든 계정 예산 알림
- [ ] C-2 AI API 하드 한도
- [ ] C-3 사용자당 호출 제한
- [ ] C-4 무료 한도 초과 시 동작
- [ ] C-5 반복 실행 의심
- [ ] C-6 전송량
- [ ] C-7 잊힌 리소스
- [ ] C-8 로그 보관 기간
- [ ] C-9 월 고정비 표
- [ ] C-10 결제 수단 분리

백업·복구

- [ ] B-1 자동 백업
- [ ] B-2 복구 리허설
- [ ] B-3 외부 사본
- [ ] B-4 업로드 파일 백업
- [ ] B-5 복구에 필요한 것 목록
- [ ] B-6 소프트 삭제
- [ ] B-7 위험 명령 안전장치
- [ ] B-8 코드 원격 저장소
- [ ] B-9 계정 복구 수단
- [ ] B-10 복구 순서 문서

보안

- [ ] S-1 코드·git 기록 비밀값 없음
- [ ] S-2 프론트 코드 비밀값 없음
- [ ] S-3 남의 데이터 접근 차단
- [ ] S-4 RLS·보안 규칙
- [ ] S-5 관리자·디버그 닫기
- [ ] S-6 에러 메시지 정리
- [ ] S-7 인증 횟수 제한

- [ ] S-8 CORS·쿠키
- [ ] S-9 의존성 취약점 알림
- [ ] S-10 업로드 제한

부록 B. 용어 한 줄 정리

용어	한 줄
롤백	직전 정상 버전으로 되돌리기
헬스 체크	서비스가 정상인지 답하는 주소
업타임 모니터링	외부에서 주기적으로 접속해 보고 죽으면 알려 주는 것
마이그레이션	DB 구조 변경을 코드 파일로 관리하는 것
RLS	행 단위 보안. DB가 「이 행은 이 사용자만」을 직접 막는 규칙
하트비트 체크	작업이 「끝났다」는 신호가 안 오면 알려 주는 방식
3-2-1 백업	사본 3개 · 매체 2종 · 1개는 외부
소프트 삭제	바로 지우지 않고 「삭제됨」 표시만 해 두는 것
하드 한도	넘으면 실제로 멈추는 한도(알림과 다름)

마치며

이 체크리스트 50개를 다 해도 사고는 나요. 운영은 사고를 없애는 일이 아니라 **사고가 나도 빨리 알고, 빨리 되돌리고, 같은 사고를 두 번 안 내는 일**이에요.

빠진 게 보이거나, 직접 겪은 사고가 있으면 Threads `@cronlog`로 알려 주세요. 다음 버전에 반영할게요.

[경험: 마지막 한마디 — 이 자료를 만든 이유를 본인 말로 1~2줄]

- 이 자료는 일반적인 운영 점검 가이드예요. 서비스마다 환경이 달라서, 적용 전 각 서비스의 공식 문서를 확인하세요.
- 특정 서비스 이름은 예시이며, 이 자료는 어떤 업체와도 제휴·협찬 관계가 없어요. `[확인: 발행 시점에도 제휴 링크 없음]`
- © cronlog · 출처 표시 시 공유·인용 자유 · 상업적 재판매 금지