기술

저희가 배포한 규칙은 정말 팀에 도착했을까

방동걸 (주)웨이스 대표이사
2026년 8월 26일

팀이 사용하는 도구에 규칙을 하나 넣었다고 해 봅시다. 위험한 명령을 실행하기 전에 한 번 물어보는 확인, 파일을 고치기 전에 사본을 떠 두는 백업, 작업을 끝낼 때 빠뜨린 것이 없는지 훑는 점검 같은 것들입니다. 코드를 쓰고, 테스트를 돌리고, 초록불을 확인하고, 팀에 공지합니다. 여기까지 하면 보통 배포했다고 말합니다. VEXPLOR를 개발하는 웨이스가 사내 도구에 규칙을 배포하고 나서 직접 확인해 본 이야기입니다.

저희는 지난 몇 주 동안 사내 업무 도구에 이런 규칙을 계속 늘려 왔고, 그 과정에서 같은 형태의 실패를 여러 번 겪었습니다. 규칙은 분명히 만들어져 있는데 팀원의 컴퓨터에서는 돌지 않고 있었습니다. 그리고 매번, 아무 오류도 나지 않았습니다.

오류를 내는 실패는 금방 고쳐집니다. 누군가 빨간 글씨를 보고 신고하기 때문입니다. 오래 사는 쪽은 조용한 실패입니다. 화면은 정상이고 검사는 초록이고 아무도 불평하지 않는데 기능만 없습니다. 저희가 밟은 조용한 실패 여섯 자리와, 그 뒤에 만든 확인 질문 네 가지를 적습니다.

01

동기화가 폴더 하나를 통째로 빼고 있었습니다

저희 업무 폴더는 클라우드 동기화로 팀원들의 컴퓨터에 복제됩니다. 규칙 파일은 그 폴더 안의 설정 폴더에 들어 있었고, 설정 폴더 이름은 관례대로 점으로 시작했습니다.

그런데 동기화 클라이언트의 기본 필터가 점으로 시작하는 이름을 제외 대상으로 잡고 있었습니다. 발행하는 컴퓨터에는 파일이 있고, 서버에도 올라가고, 받는 쪽에서 조용히 버려집니다. 받는 쪽 로그에는 "필터에 걸려 차단됨"이 남지만 그 로그를 여는 사람은 없습니다. 화면에는 아무것도 뜨지 않습니다.

결과는 이렇습니다. 규칙을 적은 사람은 팀 전체에 규칙이 있다고 믿고, 팀원 여섯 명의 컴퓨터에는 그 규칙이 처음부터 없었습니다. 몇 주 동안 아무도 몰랐습니다. 알게 된 계기도 점검 도구가 아니라, 한 팀원이 다른 이유로 자기 컴퓨터를 들여다보다가 폴더가 비어 있는 것을 발견한 것이었습니다.

나중에 필터를 풀어 보려 했지만 그것도 실패했습니다. 설정 데이터베이스를 직접 고쳐도 프로그램이 시작할 때 기본값으로 되돌렸고, 유지에 성공한 뒤에도 다운로드가 계속 거부됐습니다. 더 중요한 사실은 따로 있었는데, **필터를 바꿔도 이미 제외된 파일을 거슬러 올라가 내려받지는 않는다**는 점이었습니다. 필터는 새로 들어오는 변화만 처리합니다. 즉 필터를 고치는 순간부터 미래는 해결되지만 과거는 그대로 비어 있습니다.

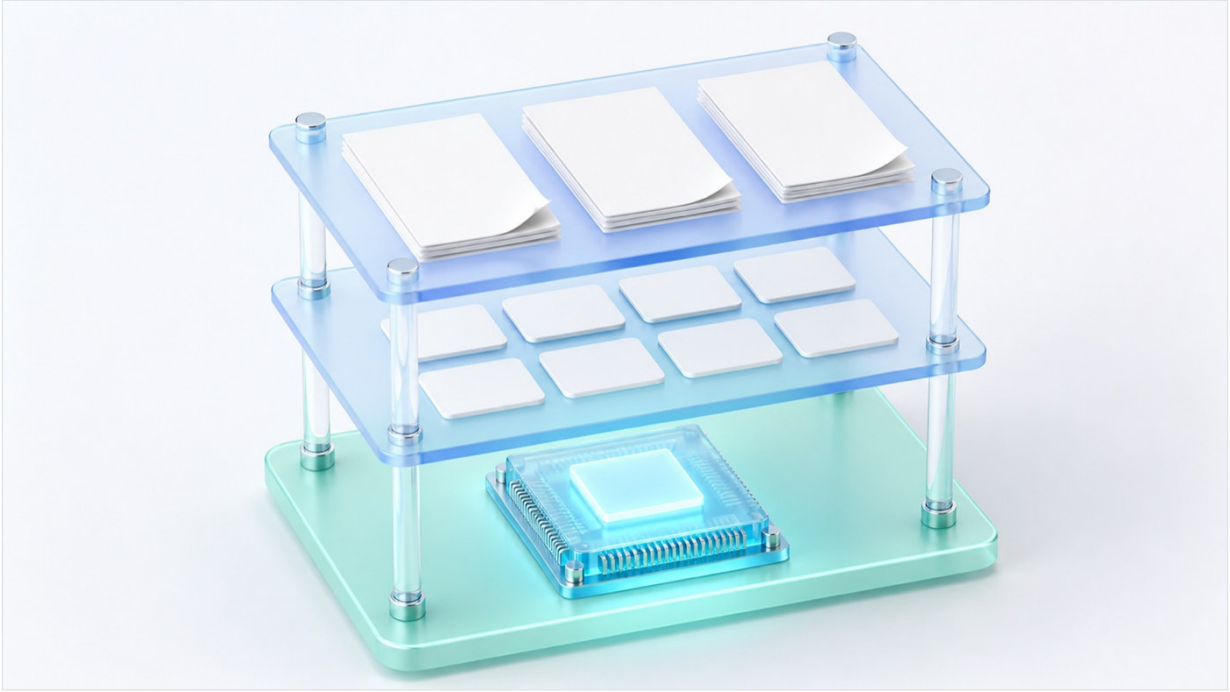
02

도착해도 실행되지 않았습니다

두 번째 자리는 더 단순합니다. 규칙을 실행하는 스크립트를 셸 스크립트로 썼는데, 저희 팀에서 이 도구를 사용하는 일곱 명 중 여섯 명이 다른 운영체제를 사용합니다. 파일이 도착해도 실행할 수단이 없습니다.

이것도 오류를 내지 않습니다. 도구는 규칙을 실행하려 시도하고, 실행에 실패하고, 실패를 무시하고 계속 진행합니다. 안전장치를 실패로 처리해 작업 자체를 멈추면 그게 더 큰 사고이기 때문에 대부분의 도구가 이렇게 설계돼 있습니다. 그러니 화면만 봐서는 안전장치가 하나도 없는 상태와 잘 도는 상태를 구분할 수 없습니다.

이 일을 겪고 나서 배포를 두 사건으로 갈라 보게 됐습니다. **파일이 도착했는가**와 **그 파일이 실행되는가**는 다른 질문이고, 둘 다 조용히 실패합니다.



규칙을 문서·판정 목록·실행부 세 겹으로 나눠 각각 다른 경로로 배달한다

03

그래서 규칙을 세 겹으로 나눴습니다

두 실패를 겪고 나서 규칙을 한 덩어리로 배포하는 방식을 버렸습니다. 규칙이라고 부르던 것 안에 성격이 다른 세 가지가 섞여 있었다는 것을 그때 알았습니다.

겹	담는 것	배달 방식
선언	사람이 읽는 문서	일반 폴더에 평문
데이터	판정 목록·경로	일반 폴더에 파일
실행부	실제로 판정하는 코드	앱 설치본에 포함

실행부는 파일로 나눠 주지 않고 앱 설치본에 넣었습니다. 코드를 파일로 나눠 주면 동기화 필터와 운영체제 차이를 둘 다 통과해야 하지만, 앱 설치본에 넣으면 앱이 깔린 컴퓨터에는 반드시 도착하고 앱이 실행하니 실행도 보장됩니다. 판정에 필요한 목록만 일반 파일로 두면 규칙을 바꿀 때 앱을 새로 배포하지 않아도 됩니다.

이렇게 나누고 나서야 "규칙이 있다"는 말을 겹마다 따로 하게 됐습니다. 선언은 전원에게 도착하고, 데이터도 전원에게 도착하고, 실행부는 앱을 설치한 사람에게만 있습니다. 이제 세 문장을 각각 따로 확인합니다. 그게 실질적인 소득이었습니다.

04

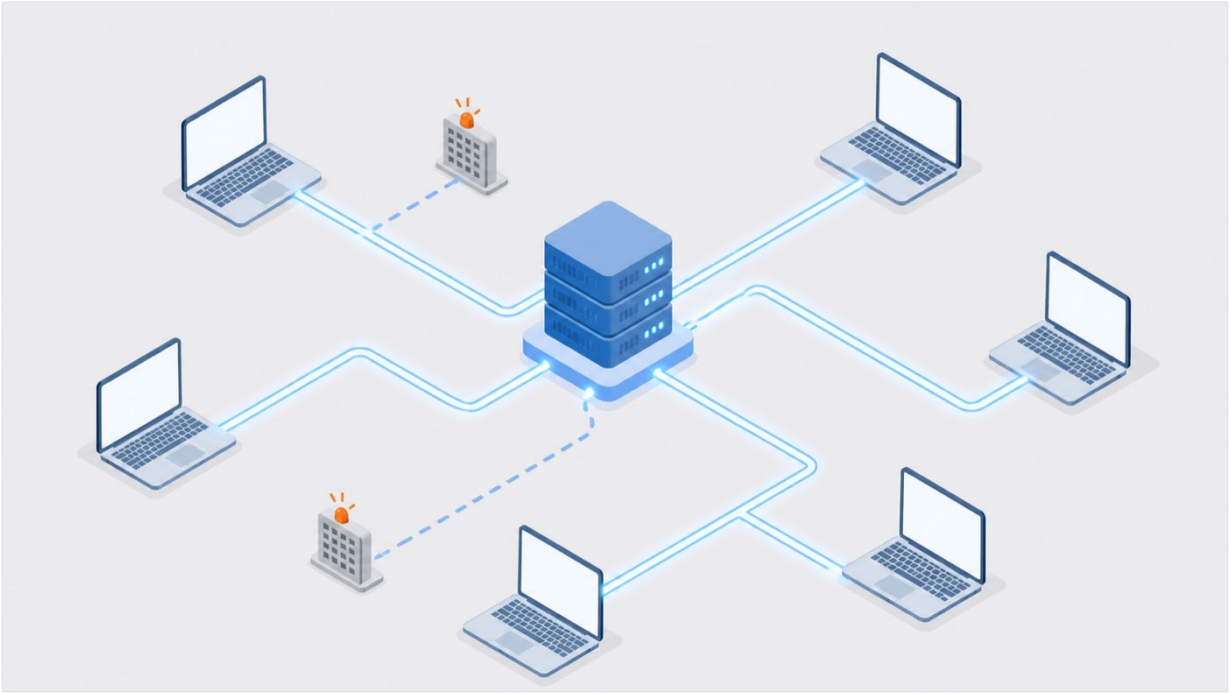
배달 이전에, 만드는 단계에서도 같은 일이 났습니다

여기까지가 배달의 문제였다면 그 앞단에도 같은 형태가 있었습니다.

새 버전을 발행할 때 자동 빌드가 돌고, 초록불이 뜨면 발행이 끝난 것으로 봤습니다. 그런데 어느 릴리스에서 산출물이 두 개만 올라와 있었습니다. 정상이면 일곱 개가 나옵니다. 한쪽 운영체제용 설치 파일이 통째로 빠져 있었고, 그쪽을 사용하는 팀원들은 몇 주 전 버전에 묶여 있었습니다.

빌드가 실패한 것이 아니었습니다. **빌드를 돌릴 컴퓨터가 꺼져 있어 작업이 시작도 못 하고 줄 서 있었습니다.** 대기 중인 작업은 실행 목록에 나타나지 않습니다. 화면에는 성공한 작업만 보이고, 그 성공은 진짜 성공입니다. 다른 하나가 아예 시작되지 않았다는 사실만 안 보입니다.

이걸 겪고 완료 판정을 바꿨습니다. 초록불이 아니라 **기대 산출물 개수가 맞는지**로 봅니다. 일곱 개가 나와야 하면 일곱 개를 세고, 두 개면 미완입니다. 상태를 보는 대신 개수를 세는 것 말고는 이 문제를 잡을 방법이 없었습니다.



중앙에서 나간 배달 경로 중 두 갈래가 동기화 필터에 막혀 끝나 있다

05

배달을 만들자 배달 자체가 위험이 됐습니다

파일을 자동으로 나눠 주는 경로를 새로 만들었더니, 그 경로 자체가 새로운 위험이 됐습니다. 만들자마자 서로 다른 네 관점으로 검토를 붙였고 결함 열두 종이 재현됐습니다. 링크를 따라가 엉뚱한 위치에 쓰는 문제, 쓰는 도중 프로그램이 죽으면 반쪽 파일이 남는 문제, 받는 쪽이 손으로 고쳐 둔 내용이 말없이 덮이는 문제 같은 것들이었습니다.

실제 피해는 없었습니다. 그 버전이 어느 컴퓨터에도 설치되기 전에 잠갔기 때문입니다. 다시 열기까지 여러 번의 릴리스가 걸렸고, 열 때는 두 가지를 앞에 세웠습니다. 하나는 **이 묶음을 발행한 컴퓨터가 발행 권한이 있는 컴퓨터인가**를 확인하는 것, 다른 하나는 **파일마다 지문을 대조해 어긋난 파일만 설치에서 빼는 것**입니다.

여기에는 있는 그대로 적어야 할 한계가 있습니다. 이 확인은 **파일이 발행된 그대로인지**를 보장할 뿐, **발행한 내용이 옳은지**는 보장하지 않습니다. 발행 위치에 쓸 수 있는 사람은 지문 목록도 함께 고칠 수 있습니다. 무결성 검사는 전송 중 손상과 중간 변조를 막는 장치이지 내부자를 막는 장치가 아닙니다.

06

무결성을 통과한다고 최신이라는 뜻은 아닙니다

가장 오래 안 보였던 자리가 여기입니다.

원본 파일을 고친 뒤, 배포 사본은 그대로 두고 한 시간 사십 분이 지난 상태를 발견한 적이 있습니다. 그동안 어느 검사에도 걸리지 않았습니다. 지문 대조는 전부 일치했고, 파일 개수를 세는 진단은 초록이었고, 발행 권한 확인도 통과했습니다.

이유는 생각해 보면 당연합니다. **사본과 지문 목록이 함께 낡기 때문**입니다. 사본을 만들 때 지문도 같이 만들었으니 둘은 영원히 서로 일치합니다. 서로를 아무리 대조해도 낡았다는 사실은 나타나지 않습니다. 낡음은 원본과 대조해야만 보이는데, 원본과 사본을 대조하는 절차가 어디에도 없었습니다.

구조가 원인이었습니다. 사본을 새로 만드는 동작이 앱이 시작할 때 한 번만 돌게 돼 있었고, 원본을 고치는 작업과 앱을 켜 두는 작업이 서로 다른 자리에서 일어납니다. 그래서 앱이 켜져 있는 동안의 모든 수정은 다음 재시작까지 나가지 않습니다. 이건 버그가 아니라 설계였고, 설계가 만든 사각지대를 아무도 보고 있지 않았을 뿐입니다.

무결성과 최신성은 다른 이야기입니다. 한쪽을 확인했다고 다른 쪽이 함께 확인되지는 않습니다.

07

고치는 쪽과 실제로 도는 쪽이 달랐습니다

마지막은 조금 허탈한 이야기입니다.

규칙 하나가 필요 이상으로 자주 경고를 띄우길래 원인을 찾아 고쳤습니다. 테스트도 통과했습니다. 그런데 경고는 계속 떴습니다.

확인해 보니 저희가 고친 파일은 참고용 사본이었고, 실제로 실행되는 것은 앱 설치본 안의 파일이었습니다. 설정에 적힌 실행 경로를 열어 보기 전까지는 알 수 없었습니다. 두 파일은 이름이 같고 내용이 거의 같아서 열어 봐도 다르다는 느낌이 들지 않습니다.

이때 뜻밖의 소득도 있었습니다. 두 파일을 같은 입력으로 각각 실행해 결과를 비교해 봤더니 판정이 갈리는 경우가 있었습니다. 게다가 이 규칙들이 얼마나 정확한지 그때까지 한 번도 재 본 적이 없었습니다. 회귀 테스트가 전부 "지난번에 터진 것"으로 채워져 있어서, 처음 보는 형태의 위반은 구조적으로 못 잡습니다. 테스트 항목을 서른 개로 늘려 처음으로 기준선을 재 봤고, 두 사본 모두 서른 개 중 스물한두 개를 맞혔습니다. 못 잡는 다섯 가지는 실제로 보호 대상 파일을 바꾸는 명령이었습니다.

숫자보다 더 걸리는 대목이 따로 있습니다. **그 숫자를 처음 재 본 때가 규칙을 만들고 한참 뒤였습니다.** 그 전까지 저희는 규칙이 얼마나 맞는지 모르면서 그 규칙을 믿고 있었습니다.

08

사본을 하나 더 만들 때 물어야 할 네 가지

여섯 자리를 겪고 나서 정한 확인 질문입니다. 파일이든 설정이든 코드든, 같은 내용을 담은 자리를 하나 더 만들 때마다 이 넷을 먼저 답합니다.

질문	답이 없으면	우리 상태
누가 두 사본을 대조하나	조용히 갈라진다	절반만
실제로 도는 쪽은 어디인가	안 도는 쪽을 고친다	지킴
도착 실패가 오류를 내나	없는 걸 있다고 믿는다	못 지킴
검사가 최신성도 보나	낡은 채로 초록이다	절반만

마지막 열이 저희 상태입니다. 네 기준을 만들어 놓고 저희가 완전히 지키는 것은 하나뿐입니다. 「도착 실패가 오류를 내나」는 지금도 못 지킵니다. 동기화 필터 차단은 여전히 조용하고, 저희는 그 경로를 고친 것이 아니라 앱 설치본이라는 다른 길로 우회했을 뿐입니다. 기준을 만들면서 자기가 못 지키는 항목을 빼면 그 기준표는 판정력을 잃습니다.

네 번째 질문이 가장 늦게 추가됐고 가장 값이 나갔습니다. 무결성 검사를 붙이고 나면 "검증했다"는 안도감이 생기는데, 그 안도감이 최신성 확인을 미루게 만듭니다.

09

규칙을 늘리기 전에 도달을 세어 봅니다

돌이켜 보면 순서를 잘못 잡고 있었습니다. 규칙을 몇 개 더 만들지를 고민하는 동안, 이미 만든 규칙이 몇 대의 컴퓨터에서 도는지는 세어 본 적이 없었습니다. 도달률이 절반인 상태에서 규칙 개수를 두 배로 늘리면 안 도는 규칙만 두 배가 됩니다.

그래서 지금은 새 규칙을 넣을 때 이 순서로 합니다. 먼저 이 규칙이 어느 겹에 속하는지 정하고, 그 겹의 배달 경로가 이미 검증돼 있는지 확인하고, 도착과 실행을 각각 확인할 방법을 정한 다음에 코드를 작성합니다. 코드가 마지막입니다.

검사를 붙였는데도 못 잡는 경우가 있다는 이야기는 검사 자체가 대상에 닿지 않는 문제로 따로 적어 두었습니다. 이 글이 "규칙이 도착했는가"를 다뤘다면 그 글은 "검사가 대상에 닿았는가"를 다룹니다. 둘은 붙어 있지만 처방이 다릅니다. 도달은 세어서 확인하고, 검사는 결함을 일부러 심어 빨간불이 뜨는지 봐서 확인합니다.

WACE(웨이스)는 공장이 스스로 판단하고 멈추지 않게 하는 제조 AI 운영체제 VEXPLOR를 만드는 회사다.

원문 blog.vexplor.com/post/did-the-rules-actually-reach-the-team
발행 2026년 8월 26일 · 방동걸 (주)웨이스 대표이사

회사 주식회사 웨이스 (WACE Inc.) · 대표이사 방동걸 · 사업자등록번호 439-87-02852
주소 (18462) 경기 화성시 동탄구 동탄영천로 150, 에이동(A동) 1702-1708호 (영천동, 실리콘앨리동탄)
연락처 전화 070-4730-0629 · contact@wace.me
홈페이지 wace.me · linkedin.com/company/wace-inc

원문 홈페이지 링크드인