토토 관련 커뮤니티에서 움직이는 정보는 속도가 빠르고 수명이 짧다. 공지 글이 몇 시간 만에 내려가기도 하고, 링크가 하루 새 바뀌기도 한다. 수동으로 모니터링하다 보면 누락이 생기고, 데이터의 일관성도 유지하기 어렵다. 그래서 자동화를 설계한다. 핵심은 두 가지다. 첫째, 변화에 강한 파이프라인을 만든다. 둘째, 합법과 윤리의 경계를 분명하게 그어 불필요한 위험을 차단한다. 이 글은 토토갤러리 게시판을 기준 예시로 삼아, 안전하게 운영되는 데이터 수집 자동화를 어떻게 설계하고 운영할지 구체적으로 다룬다. 실무에서 겪은 단선과 우회, 선택과 포기까지 함께 담는다.

무엇을, 왜 모으는가

목표 정의가 설계의 절반을 좌우한다. 토토갤러리는 속보성 이슈, 커뮤니티 공지, 링크 교체 알림, 사용자 제보 등 다양한 게시물이 섞인다. 그중 자동화로 가치가 높은 것은 다음 유형이다. 첫째, 운영 공지와 규칙 변경처럼 장기 참조가 필요한 문서. 둘째, 링크 변경, 일시 중단 안내 등 시계열 감시가 필요한 항목. 셋째, 반복 질문과 사례 정리처럼 FAQ로 승격 가능한 콘텐츠. 넷째, 외부 위험 신고나 보안 이슈 경고처럼 즉시 알림이 필요한 건.

이때 목적을 착실하게 쪼개야 한다. 아카이빙, 검색, 경보, 요약, 통계 중 무엇이 주력인지에 따라 데이터의 스키마부터 인프라 비용까지 달라진다. 예를 들어 안전공원주소처럼 주소체계가 자주 바뀌는 주제는 변경 탐지와 신뢰도 등급이 중요하다. 반면 운영 공지 수집은 원문 보존과 변동 이력 관리가 우선이다.

합법과 윤리, 금지선 설정

웹 수집은 합법 영역을 분명히 해야 한다. 국내의 경우 사이트 약관, 저작권법, 정보통신망법, 개인정보보호법이 교차한다. 로봇 배제 표준을 존중하고, 로그인이나 우회 기술이 필요한 비공개 영역은 건드리지 않는다. 게시물 내 전화번호, 계좌번호 같은 민감 정보는 수집 자체를 차단하거나 해시 처리로 익명화한다. 공익적 경고 성격의 게시물을 아카이브하더라도, 개인 신상 식별 가능 정보는 저장하지 않는 편이 안전하다. 비식별화 규칙은 정규식으로만 끝내지 말고 사례 기반 테스트를 붙인다. 과함은 모자람보다 낫다.

또 하나, 속도 규율이 필요하다. 페이지당 최소 2초 이상 간격, 사이트 평균 0.3 requests per second 이하, 야간 집중 크롤 금지가 안정적이다. 한번 운영하다 보면 제한을 높이고 싶은 유혹이 오는데, 커뮤니티 서버는 상업 매체보다 여유가 적다. 예전에 1 rps로 잠시 올렸다가 20분 만에 차단된 적이 있다. 회복까지 하루가 걸렸고 그날 데이터는 구멍이 났다. 탐욕은 장기 운영의 적이다.

소스의 패턴을 먼저 읽는다

토토갤러리처럼 커뮤니티형 게시판은 리스트 페이지 구조와 상세 페이지 구조로 나뉜다. 리스트는 페이지네이션, 고정 공지, 조회수와 작성시간, 간혹 비밀글 표시가 있다. 상세는 본문, 이미지, 링크, 작성자 닉네임, 댓글로 구성된다. 운영하며 발견한 자주 바뀌는 부분은 두 가지였다. HTML 클래스명이 배포마다 빈번히 변경되고, 무한 스크롤 방식이 업데이트로 도입됐다가 다시 페이지네이션으로 회귀했다. 그 결과 CSS 선택자 기반 크롤러는 자주 깨졌다. 내린 결론은 이렇다. 텍스트와 위치 기반의 다중 파서 전략을 쓰자. 클래스명이 바뀌어도 헤더 텍스트나 DOM 상대 위치는 상대적으로 안정적이라 복원력이 높다.

또한 이미지나 외부 링크는 원본 서버 도메인이 변경되는 경우가 있다. 이미지 프록시나 링크 정규화 단계에서 가능한 상대경로를 절대경로로 재작성하고, 리다이렉트 체인을 기록해 추후 고아 링크를 복원할 수 있게 한다. 작은 수고가 몇 달 뒤의 감사 로그에서 큰 차이를 만든다.

아키텍처 개요, 단단하지만 가벼운 뼈대

내가 실무에서 안정적으로 쓰는 구조는 큐 중심의 비동기 파이프라인이다. 진입부에는 수집 스케줄러가 있다. 페이지 리스트 수집과 상세 수집을 분리해 큐에 넣는다. 리스트 수집은 5분 주기, 상세는 새 글 감지 시 즉시. 중간에는 rate limiter와 재시도 정책을 갖춘 워커 풀이 있고, 하단에는 정규화 파서가 있다. 저장소는 두 계층으로 나눈다. 장기 보관을 위한 객체 스토리지의 냉동창고, 질의 성능을 위한 검색 인덱스의 작업창고. 알림과 요약은 인덱스에 올라온 새 문서만 구독한다.

이 구조의 장점은 세 가지다. 확산을 제어하는 큐, 변화에 강한 파서, 이력 추적이 쉬운 저장 전략. 장애가 나도 큐에 남아 재처리가 가능하고, 파서 버전을 바꿔 재처리하면 이전 데이터도 재정규화가 된다. 운영자는 한밤중에 서버로 뛰어가지 않고도 조정 레버를 몇 개만 만지면 된다.

수집 스케줄과 페이싱

주기 설정은 비즈니스 목적과 서버 배려의 균형이다. 급변 게시판은 2분, 일반은 10분, 심야는 30분으로 배수 조절을 둔다. 페이지네이션의 끝까지 다 훑는 전수 크롤은 하루 1회로 제한한다. 보통 3개월 이전 글은 검색 인덱스에만 남기고 상세 크롤을 중단해도 충분하다. 일 최대 요청량은 10,000 페이지 정도면 작은 VM 두 대로 무난하게 돌릴 수 있다. 이 수치에서 중요한 것은 피크를 낮추는 것이다. 워커를 6개로 제한하고, 워커별 동시연결을 1로 고정하면 예측 가능성이 커진다.

사이트에 로드가 감지되면 자동으로 페이싱을 낮추는 서킷 브레이커를 둔다. 타임아웃이 3회 연속이면 5분 쉬고, HTTP 429가 나오면 30분 쉬는 단순 규칙으로도 효과가 있다. 과거 500 에러 폭주 상황에선 후퇴 지연을 두지 않아 서버에 더 큰 부담을 준 적이 있다. 페이싱은 예의이자 자기 방어다.

파서, 한 번에 깨지지 않게 만들기

DOM 파서는 취약점의 온상이다. 그래서 다층 파서를 쓴다. 1차로 구조적 파서가 클래스명과 태그를 타고 값을 뽑는다. 실패하면 2차로 텍스트 주변 단서에 의존하는 휴리스틱 파서가 동작한다. 예를 들어 작성시간은 "작성일" 레이블 옆 텍스트를 찾거나, 날짜 정규표현식으로 유사 텍스트를 스캔한다. 3차로 백업 스냅샷에서 OCR을 시도하는 극단적 방법도 남겨두는데, 이는 비용이 커서 알림성 글에 한정한다. 이 다층 구조는 유지보수 비용을 올리지만 다운타임을 크게 줄여준다. DOM이 바뀐 밤에도 2차 파서가 최소 정보를 확보해 알림을 끊지 않는다.

이미지는 원본 다운로드 대신 원격 해시만 계산해 중복을 판별한다. 대역폭을 아낀다. 텍스트는 광고성 서명과 공통 머리말을 제거해 노이즈를 줄인다. 링크는 URL 정규화, 트래킹 파라미터 제거, 리다이렉트 추적을 거친다. 댓글은 정책적으로 수집 범위 밖에 두는 것이 안전하지만, 이슈 추적 목적이면 작성자, 시간, 본문만 남기고 식별 가능 요소는 모두 버린다.

토토갤러리 안에서 신뢰 신호 찾기

커뮤니티에는 신뢰 신호가 있다. 고정 공지, 운영자 닉네임, 특정 뱃지, 게시물 형식 같은 것들이다. 안전공원주소처럼 위험 회피 목적의 게시물은 오판 가능성이 크다. 글 제목에 키워드가 있어도 본문은 경고일 수도 있다. 그래서 규칙을 단단하게 만든다. 제목의 단어만 보지 말고 본문에 경고 동사, 신고 문구, 조심하라는 표현이 있는지 살핀다. 운영자나 검증된 닉네임의 글인지, 댓글 상단에 안내 문구가 붙었는지, 이전에 같은 링크로 문제가 제기된 이력이 있는지도 신호다. 이 신호를 점수화해 임계치 이상일 때만 알림을 보낸다. 과열된 알림은 결국 무시된다.

또 하나의 신호는 시간대다. 링크 교체는 대개 새벽 1시에서 3시 사이 빈도가 높았다. 반대로 운영 공지는 오후 2시 전후에 올라오는 경우가 많았다. 시간대 기반 가중치를 두면 알림 큐가 과밀해지는 걸 막는다. 작은 패턴도 운영에선 효율을 바꾼다.

데이터 스키마, 나중의 질문을 견디게

초기 설계가 가벼워 보여도 스키마는 넉넉히 잡는다. 필수 필드로는 source, postid, title, bodytext, authorlabel, createdat, fetchedat, url, contenthash. 옵션 필드로 tags, riskscore, linkentities, imagehashes, moderationflags, parseversion, domsnapshotref. 특히 parseversion은 금쪽같다. 파서를 새로 배포한 뒤 과거 문서를 다시 태워 품질을 끌어올릴 수 있다. Contenthash는 중복 게시물 병합에 유용하고, domsnapshot_ref는 이의 제기나 사실 검증에서 원형을 보여준다.

저장은 쓰임새에 맞춰 분화한다. 검색성과 필터가 중요하면 Elasticsearch 계열, 장기 보관과 비용 최적화가 중요하면 객체 스토리지와 압축 아카이브. 나는 최근 6개월치만 검색 인덱스에 유지하고, 그 이전은 Zstd로 압축해 S3 호환 스토리지에 보관한다. 비용은 월 수천 원 단위로 떨어지는데, 복구는 분 단위로 가능하다.

알림 설계, 시끄럽지 않게 정확히

경보는 적고 정확해야 쓸모가 있다. 임계치를 다단으로 둔다. 낮은 임계치 알림은 슬랙의 조용한 채널로, 높은 임계치는 푸시와 SMS. 같은 이벤트를 여러 번 보내지 않도록 디듀플리케이션 키를 linkentities 해시와 postid 조합으로 만든다. 재현 가능성이 필요하면 알림에 파서 버전과 스냅샷 링크를 건다. 과거에는 제목만 보내다 보니 현장을 떠난 다음엔 뭘 봐야 할지 헷갈렸다. 메타데이터는 알림의 절반이다.

알림과 함께 요약을 붙이는 게 논쟁거리다. 실시간 요약은 계산 비용이 있고, 때로는 오독을 만든다. 내 경험으론 위험 알림에는 키 포인트 2줄 정도의 추출 요약만, 주간 리포트에는 더 긴 생성 요약을 붙이는 절충이 좋았다. 이 균형이 트래픽과 컴퓨팅 비용을 크게 절약한다.

장애와 회복, 실제 사건 다섯 가지

한 번은 사이트가 무한 스크롤로 바뀌면서 페이지네이션 파서가 전멸했다. 2차 파서가 리스트에서 날짜 패턴을 못 찾았고, 신규 글이 일시적으로 끊겼다. 해결은 간단했다. 브라우저리스 렌더링을 켜고 스크롤 이벤트를 일정 횟수만 실행했다. 렌더링 비용이 부담돼, 정상화 이후에는 프리렌더된 JSON 엔드포인트를 찾았고, 그 뒤로는 렌더링을 다시 껐다.

또 다른 날에는 이미지 프록시 도메인이 바뀌며 해시가 모두 달라졌다. 중복 판단이 깨졌고, 스토리지 사용량이 이틀 만에 40% 증가했다. 링크 정규화 모듈에서 도메인 매핑 테이블을 뒀고, 해시 계산을 원본 URL 기준으로 바꿨다. 교훈은 해시의 기준점을 명확히 하자는 것.

세 번째는 크롤러가 429를 반복해서 받았는데 재시도 정책이 공격적으로 설정돼 있었다. 그대로 차단까지 이어졌다. 이후 429는 즉시 장기 백오프로 보내고, 운영자가 수동으로 해제할 때까지 쉬도록 규칙을 고쳤다. 자동화의 오판은 빠르고 집요하다.

네 번째는 운영자 닉네임이 사칭당한 사건이다. 신뢰 신호에 닉네임만 크게 반영하던 탓에 위조 게시물이 알림을 뚫고 들어왔다. 그 뒤로는 닉네임만으로는 높은 점수를 주지 않고, 계정 생성 시기나 고정 공지와의 교차 링크 여부 같은 추가 신호를 합쳤다.

마지막은 데이터 삭제 요청이다. 특정 게시물의 원문 보관에 대해 삭제를 요구받았고, 정당성을 인정해 영구 삭제했다. 이때를 대비해 소프트 딜리트만 두지 말고, 완전 삭제 경로와 이력 기록을 별도 암호화 테이블로 유지하는 게 안전하다. 삭제 자체의 로그는 남기되 원문은 남기지 않는다.

운영 보안, 작은 구멍이 큰 사고를 부른다

크롤러의 사용자 에이전트는 투명하게 밝혀둔다. 문의용 이메일을 넣으면 오해를 줄인다. 접근 키는 환경 변수로만 관리하고, 저장소 버킷은 공개 차단. 빌드 아티팩트에는 테스트 스냅샷을 절대 포함하지 않는다. 실수로 커밋된 적이 있는데, 그 스냅샷에 민감 표현이 있었다. 덕분에 접근권한 분리와 프리커밋 훅을 도입했다. 백업은 암호화한 뒤 별도 리전에 이중화한다. 키 순환 주기는 분기마다, 감사 로그는 1년 보관이 안전했다.

비용과 성능, 숫자를 들여다보기

소규모 운영 기준으로 계산해보자. 한 달에 30만 요청, 평균 응답 200 KB라고 하면 총 전송량은 약 60 GB다. 객체 스토리지에 월 200 GB 저장, 일일 추가 3 GB 수준이면 중소 클라우드에서 월 수천 원대다. 브라우저리스 렌더링을 5% 비율로 제한하면 CPU 시간도 감당 가능하다. 검색 인덱스는 50만 문서 이하면 싱글 노드로 충분하고, 1GB 안전공원주소 RAM 컨테이너에서도 무리 없다. 요약 같은 부가기능은 야간 배치로 돌리고, 낮에는 실시간 추출 요약만 쓰면 비용이 내려간다.

성능 최적화의 첫 단추는 파싱 단계에서의 제거다. 광고 블록, 서명, 중복 이미지, 불필요한 메타 태그를 과감히 버리면 인덱스 크기가 20% 이상 줄어든다. 문서 해시로 중복을 합치면 검색 결과 품질도 좋아진다. 작은 절약이 품질과 비용을 동시에 잡는다.

키워드 취급, 안전공원주소와 간격 유지

커뮤니티에서 안전공원주소 같은 키워드는 단번에 시선을 끈다. 그러나 오판 시 피해가 크다. 자동화는 이 키워드를 직접적으로 추천하거나 랭킹을 매기지 않는다. 대신 세 가지 절차를 둔다. 과거 게시물과의 링크 중복 여부, 운영자 공지에서의 언급 여부, 사용자 신고 기록의 존재. 이 세 신호가 모일 때만 높은 위험 경보를 띄운다. 또한 주소, 연락처, 계좌번호 같은 민감 정보는 원문 보존 대상이더라도 요약이나 알림에는 절대 포함하지 않는다. 이 원칙은 판단 부담을 줄이고, 법적 리스크를 낮춘다.

품질 관리, 숫자와 표본의 병행

품질은 자동 점수와 사람의 표본 검토가 함께 만든다. 자동 점수로는 수집 성공률, 파서 1차 성공 비율, 알림의 진위율, 중복 제거율, 인덱스 지연 시간을 본다. 주간으로 그래프를 그리면 추세를 읽고, 파서 배포 전후를 비교한다. 표본 검토는 매주 50건 정도 무작위로 뽑아 수집본과 원문을 대조한다. 이 과정에서 레이블링을 함께 하면 향후 휴리스틱 개선에 쓸 훈련 데이터가 생긴다. 숫자만으로는 감이 오지 않는 작은 결함을 표본이 잡아준다.

배포와 롤백, 무리하지 않는 절차

변경은 작게, 배포는 단계적으로, 롤백은 즉시 가능하게. 새 파서는 트래픽의 10%만 태워서 A/B 비교를 하고, 품질 지표에서 유의한 차이를 보이면 50%, 그 다음 100%. 실패 시에는 환경변수 하나로 구버전 파서를 복귀하게 해둔다. 롤백이 빨라야 실험이 과감해진다. 배포 로그에는 파서 버전, 변경 요지, 예상 영향, 되돌리기 방법을 남긴다. 야간 대규모 배포는 피하고, 업무 시간 내에만 시도한다. 문제가 생겼을 때 사람이 깨어 있어야 한다.

간단 점검 체크리스트

- 로봇 배제와 약관 준수 여부를 문서화했는가
- 개인식별정보 필터가 최신인지, 테스트 케이스가 있는가
- rate limit와 백오프 정책이 적용됐는가
- 파서 다층 구조와 parse_version 추적이 준비됐는가
- 알림 디듀플리케이션과 임계치가 과하지 않은가

예시 워크플로, 하루의 리듬

새벽 1시, 리스트 수집이 주기를 당겨 활동 급증을 감지한다. 링크 변경 의심 게시물이 세 건 포착되고, risk_score가 임계치에 근접한다. 슬랙에 조용한 알림이 전달된다. 20분 뒤 운영자 계정에서 공지가 올라오자 점수가 임계치를 넘어선다. 푸시 알림이 두 채널로 간다. 동시에 링크 엔티티가 추출되고, 과거에 유사 도메인으로 경고가 있었는지 교차검증한다.

오전 시간엔 신규 게시물의 상세 수집이 평소 대비 30% 많다. 큐의 길이가 늘어나자 워커가 6개에서 8개로 자동 확장된다. HTTP 429가 두 차례 발생하자 회로 차단기가 15분 쉬어간다. 그 사이 리스트 수집은 저강도로 유지된다. 점심 무렵에는 야간 급변에 대한 요약 리포트가 준비된다. 상위 키워드, 공지 모음, 경보 내역, 실패율이 정리된 1페이지다. 사람은 이 리포트를 보고 필요하면 레이블을 보정한다.

오후에는 파서 소폭 개정 배포가 예정돼 있다. A/B로 10% 트래픽만 새 파서에 흘린다. 2시간 동안 수집 성공률이 0.8% 상승했고, 알림 진위율은 변함없다. 50%로 확대한다. 저녁에는 완전 전환을 마친다. 새벽엔 전수 크롤이 하루 한 번 돌아가며 오래된 문서를 보관소로 밀어낸다. 인덱스는 6개월치 범위로 유지되고, 오래된 주제는 압축 백업으로 안전하게 내려간다.

무엇을 자동화하지 않을 것인가

안정적인 시스템은 자동화의 범위를 아는 데서 나온다. 법적 위험이 있는 로그인, 보안 우회, 인증 토큰 모사 같은 영역은 금지한다. 사람의 해석이 필수인 민감 판단은 자동화하지 않는다. 예컨대 개인 피해 사례 접수는 제목만 기준으로 경보를 띄우지 않는다. 또한 커뮤니티 내부의 분쟁 글은 수집하더라도 알림과 요약 대상에서는 제외한다. 자동화가 불을 키우지 않게 설계를 보수적으로 가져간다.

마무리 대신, 오래 가는 운영의 습관

토토갤러리와 같은 커뮤니티 연계 자동화는 기술의 싸움이면서도 태도의 싸움이다. 서버를 존중하고, 약관을 읽고, 개인정보를 다루지 않으며, 알림을 절제하는 태도. 거기에 작은 실패를 기록하고 반복하지 않게 만드는 운영 습관. 이 두 가지가 있으면 시스템은 느리지만 정확하게 좋아진다. 아키텍처는 단단하고, 파서는 변화에 견디고, 알림은 조용히 필요한 때만 온다. 안전공원주소처럼 민감한 주제도 차분하게 다루게 되고, 데이터의 수명은 길어진다.

운영을 하다 보면 어느 날 구조가 바뀌고, 또 어느 날은 모두가 쉬는 날처럼 조용하다. 자동화는 그 변화를 골고루 기록하고, 과장되지 않은 방식으로 전달한다. 그게 목표다. 필요한 것만, 하지만 빠지지 않게.