M13-01 · GIBALJA VISUAL MANUAL

규모와 위험에 맞는 아키텍처 선택하기

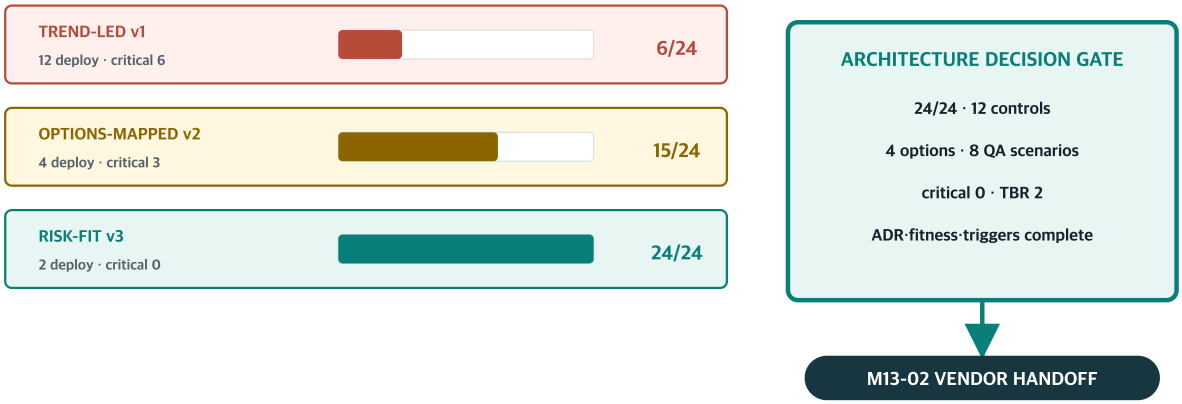
한 문장 목표: `M12-04 linked source → context·stakeholder → driver·quality scenario → 4 options → structure·data·failure·security·operations·cost → trade-off·ADR → fitness·evolution → Architecture Decision Gate → M13-02 handoff` 를 evidence로 연결합니다.

학습 안전 경계: 이 장의 team·traffic·latency·RTO·RPO·retention·cost·architecture는 모두 합성 학습 값입니다. `architecture_decision_ready` 는 실제 architecture·capacity·availability·security/privacy certification·budget·vendor·contract·build·pilot·release 승인이나 보장이 아닙니다.

M13-01 · 15

세 version을 비교하고 Architecture Decision Gate로 닫습니다

유행 중심 기준선에서 요구 연결 중간안을 거쳐 risk-fit 후보로 개선하고 외주 제안·검수 입력을 만듭니다.

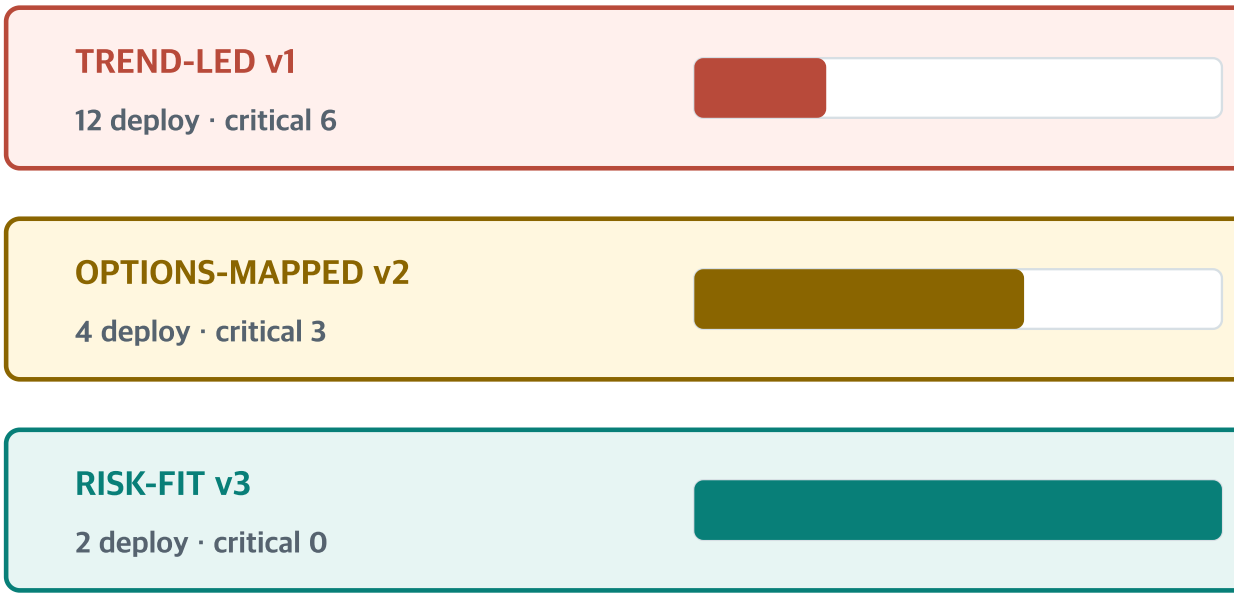


architecture_decision_ready는 M13-02 입력 후보이며 실제 vendor 선정·contract·architecture·build·release 승인이 아닙니다.

한눈에 보기. 유행 중심 6/24에서 risk-fit 24/24로 개선하고 evidence·risk·TBR·handoff를 닫습니다.

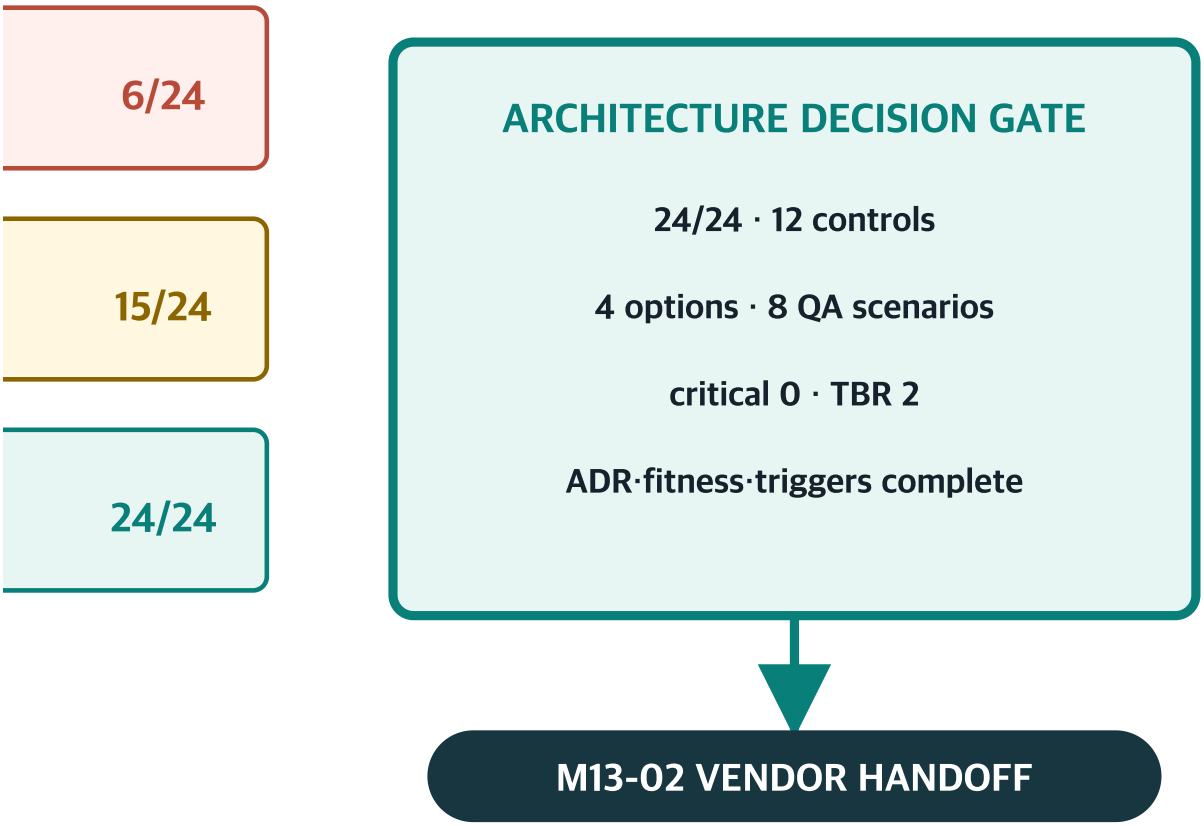
세 Version을 나란히 Architecture

유행 중심 기준선에서 요구 연결 중간안을 거쳐 risk-fit 후보로 개선하고 외



Decision Gate 입력 구성

주 제안·검수 입력을 만듭니다.



1. 이 장에서 완성할 것

산출물	핵심 내용	합성 완료 신호
Architecture input	M12-04 source·context·driver·constraint·TBR	21 linked artifacts·TBR 2
Quality portfolio	8개 six-part scenario	measure·owner·evidence
Option portfolio	OPT-A~D	same boundary·same scenarios
Architecture views	context·container·module·data·failure·trust	title·scope·legend·labels
Decision package	matrix·ADR·fitness·trigger	24/24·critical0
M13-02 handoff	vendor response·acceptance·inspection input	not vendor approval

2. 그림부터 읽는 5분 지도

그림 묶음	먼저 볼 질문	설명할 수 있어야 할 것
01~03	무엇을 왜 누구 관점에서 결정하나	source·boundary·viewpoint
04~06	품질을 어떻게 후보 비교 기준으로 바꾸나	driver·six-part scenario·4 options
07~10	선택 후보의 구조·data·failure·trust는 무엇인가	container·ownership·outbox·security
11~13	Trade-off와 결정 이유를 어떻게 보존하나	quality balance·matrix·ADR
14~15	언제 다시 보고 무엇을 다음 단계로 넘기나	fitness·trigger·Gate·handoff

3. 합성 사례 카드

항목	합성 값	경계
Primary user	5~20명 규모 팀의 운영 조정 담당자	real user research 아님

항목	합성 값	경계
Team	제품 엔지니어 2명, 운영 지원 주 0.5명, 별도 플랫폼 팀 없음	real staffing approval 아님
Scale	20 org·200 WAU·peak 20 req/s·30 jobs/min	forecast·capacity guarantee 아님
Performance	p95 800ms signal	SLA 아님
Recovery	process 10m·RTO4h·RPO1h	availability guarantee 아님
Cost	USD 600/month signal	budget·quote 아님
Data	250k records/year·retention365d	privacy/legal approval 아님
Decision	OPT-B·ADR-001 proposed	production approval 아님

4. 기술 이름보다 결정의 연결 흐름 세우기

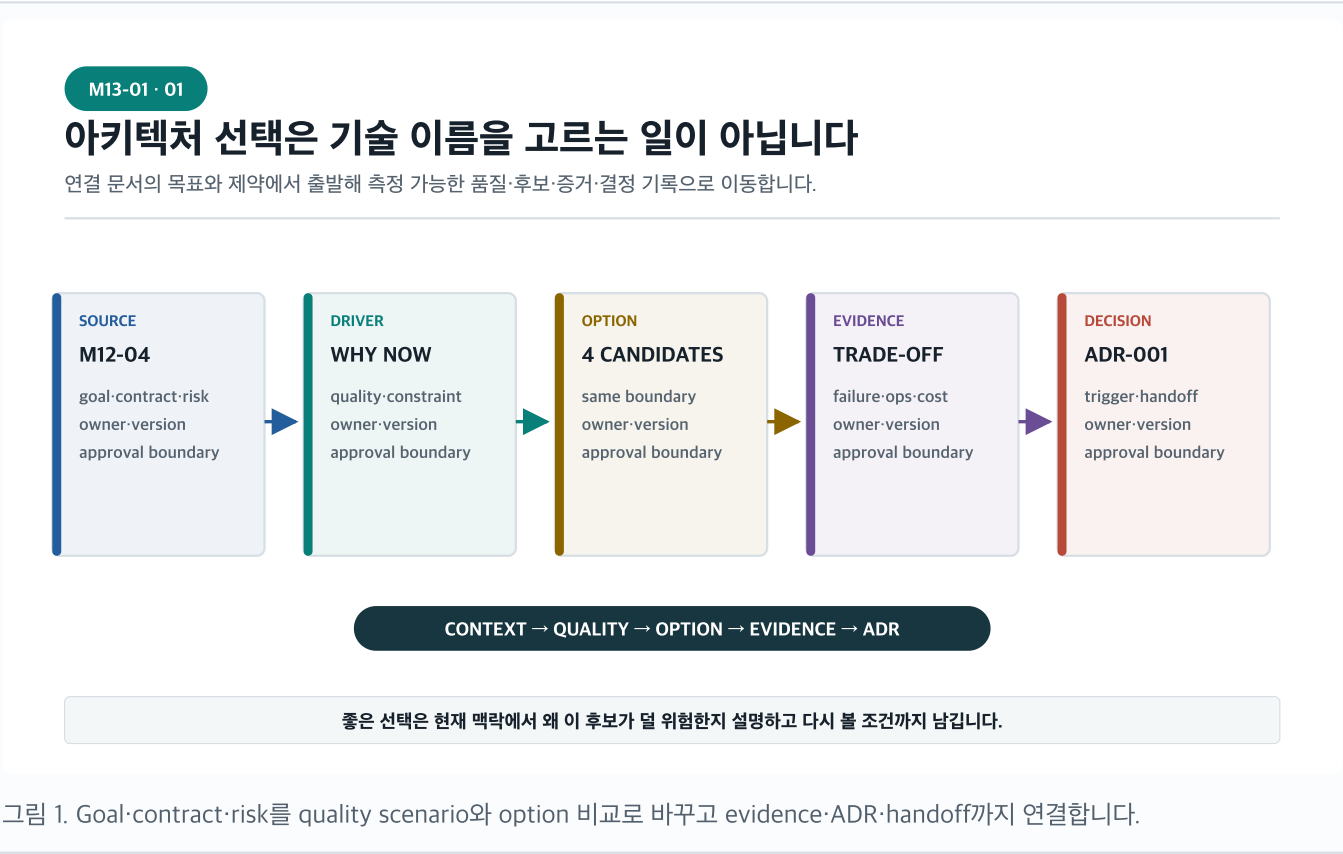
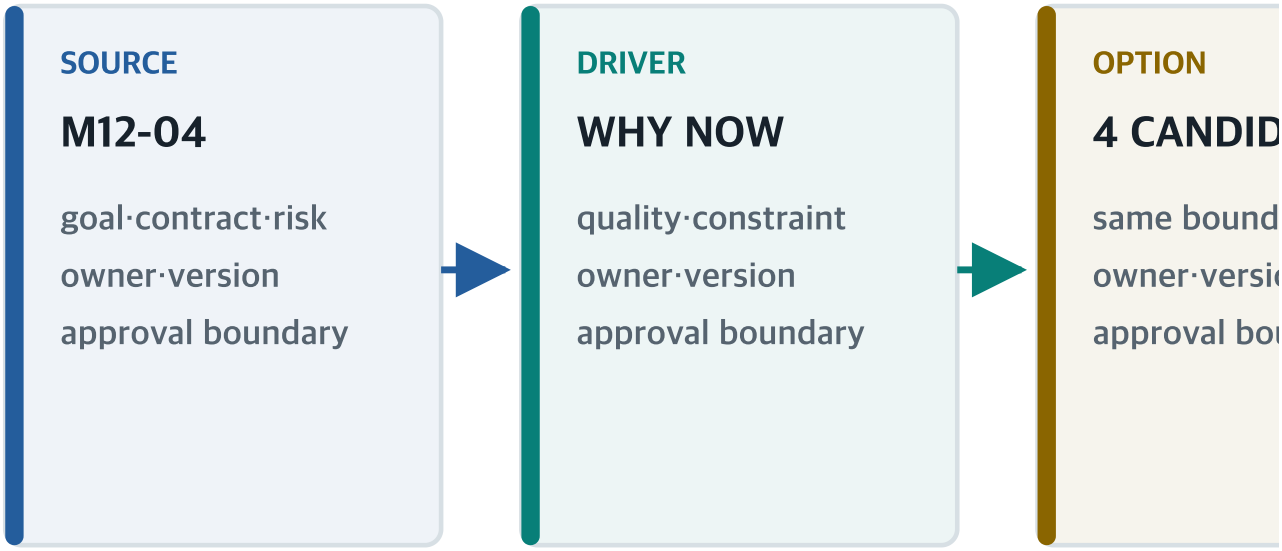
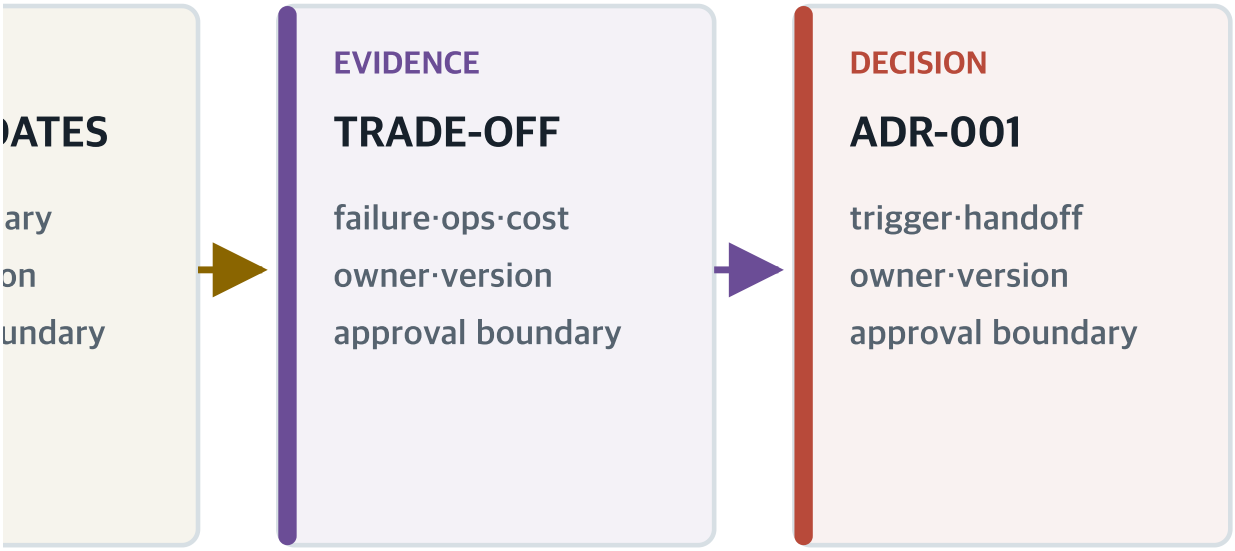


그림 1. Goal·contract·risk를 quality scenario와 option 비교로 바꾸고 evidence·ADR·handoff까지 연결합니다.

연결 문서의 목표와 제약에서 출발해 측정 가능한 품질·후보·증거·결정 기



으로 이동합니다.



TION → EVIDENCE → ADR

핵심 원칙: 아키텍처는 기술 목록이 아니라 맥락에 대한 중요한 결정의 체계입니다.

유행하는 기술을 먼저 고르면 scale·data·quality·team·cost가 나중에 그 구조에 맞춰 왜곡됩니다. 합성 사례는 M12-04의 21개 linked artifact를 source로 가져와 질문→driver→option→evidence→decision을 순서대로 닫습니다.

관점	합성 사례	확인할 증거
Source	M12-04 goal·scope·interface·risk	versioned manifest
Driver	quality·constraint·scale·team	priority+owner
Option	같은 경계의 네 후보	selected·deferred·rejected
Evidence	failure·ops·security·cost	test·drill·measure
Decision	ADR·trigger·handoff	proposed status

흔한 오답: 먼저 microservices와 Kubernetes를 선택하고 PRD에서 근거를 찾습니다.

고친 예: M12-04 source에서 8개 quality scenario와 hard constraint를 만들고 네 option을 같은 기준으로 비교합니다.

그림을 보며 4단계 연습

1. Architecture question 한 문장을 씁니다.
2. Source artifact와 version을 연결합니다.
3. Driver와 후보를 분리합니다.
4. 결정 뒤 필요한 evidence와 handoff를 적습니다.

5. Architecture·description·approval 경계 구분하기

그림 2. Architecture concept, description, proposed decision, build, release는 서로 다른 질문과 authority를 가집니다.

지금 문서가 답하는 질문과 다음 단계에서 검증하거나 승인할 질문을 분리

ARCHITECTURE

system의 구조·관계·원리

≠ 자동 승격

DESCRIPTION

view·model·decision 기록

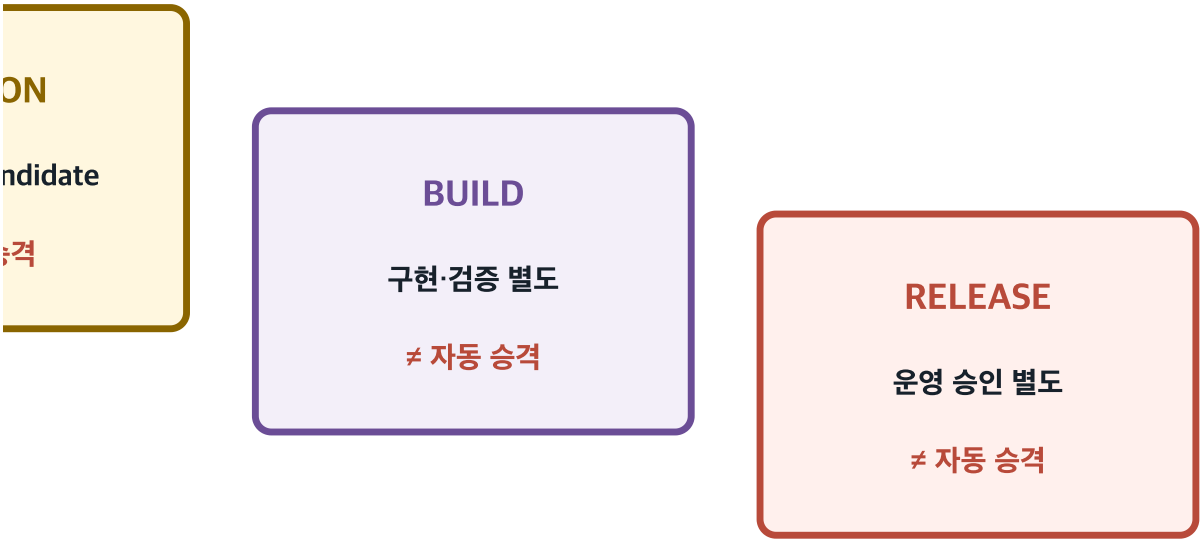
≠ 자동 승격

DECISI

proposed ca

≠ 자동 승

합니다.



핵심 원칙: 좋은 설명과 승인된 운영 시스템은 같은 상태가 아닙니다.

ISO/IEC/IEEE 42010은 architecture와 description을 구분하고 stakeholder concern·viewpoint·model의 구조를 다룹니다. 하지만 description이 잘 되었다고 capacity·security·cost·vendor·release가 승인되는 것은 아닙니다. Header에 status·answered question·unanswered question·authority boundary를 둡니다.

관점	합성 사례	확인할 증거
Architecture	요소·관계·원리	system concept
Description	view·model·rationale	review artifact
Decision	OPT-B candidate	proposed
Build	implementation·verification	별도 authority
Release	operation·assurance	별도 approval

흔한 오답: Architecture diagram이 완성됐으므로 개발과 배포를 승인합니다.

고친 예: Diagram은 decision candidate의 description이며 load·failure·security·restore·cost evidence와 build/release authority는 별도임을 씁니다.

그림을 보며 4단계 연습

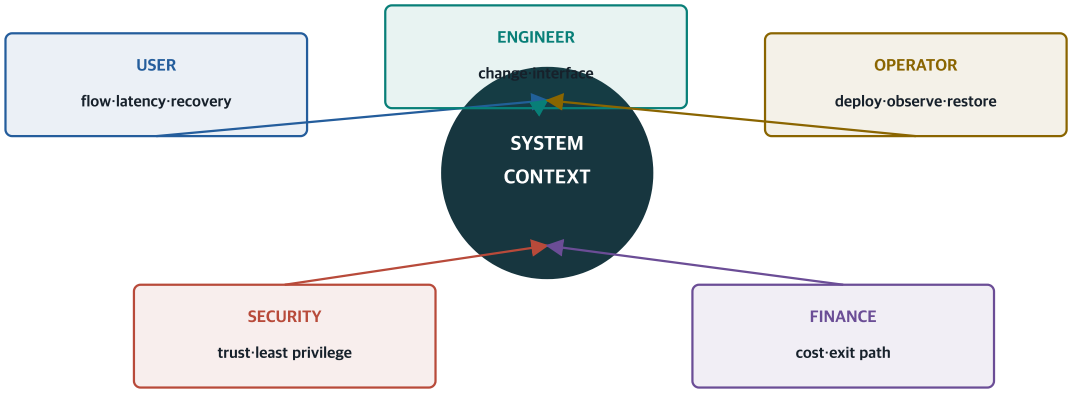
1. 이번 문서가 답하는 질문을 씁니다.
2. 답하지 않는 질문을 다섯 개 씁니다.
3. 각 질문의 evidence와 authority를 붙입니다.
4. Candidate·accepted·verified 상태를 구분합니다.

6. Stakeholder concern과 viewpoint 매핑하기

M13-01 · 03

한 장의 그림이 모든 이해관계자 질문을 답하지 못합니다

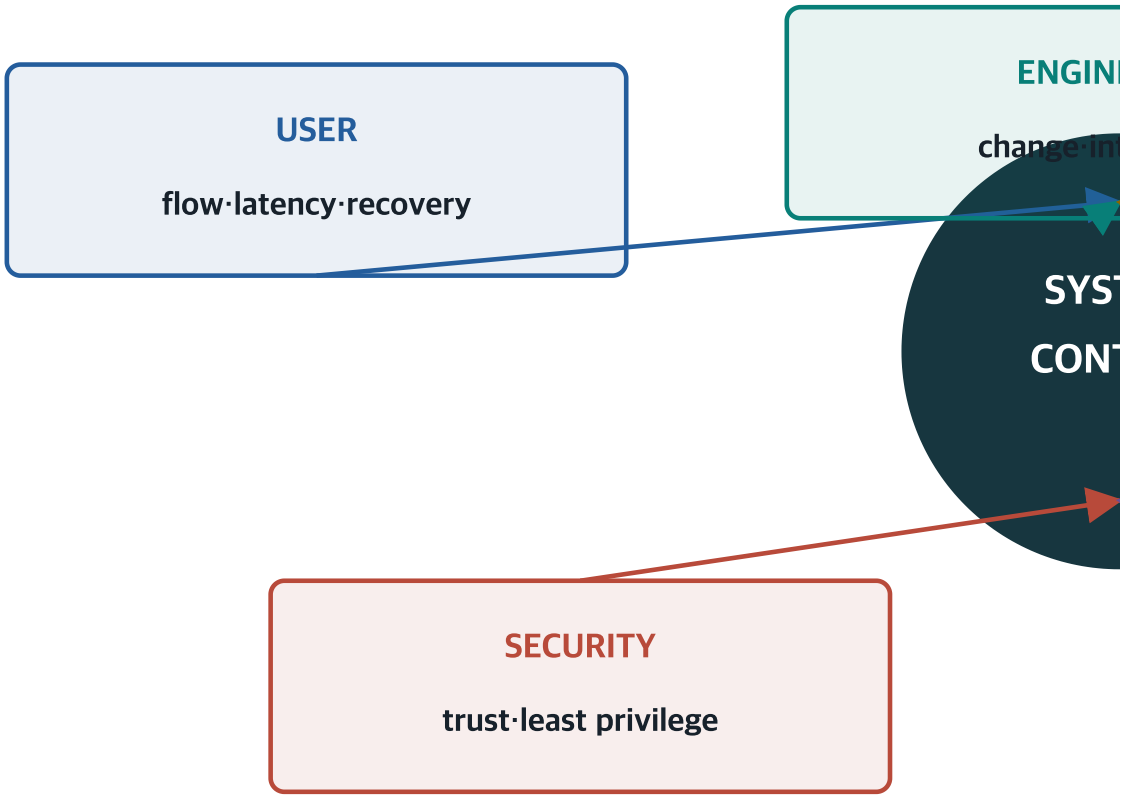
사용자·개발·운영·보안·재무 관점을 분리하고 필요한 view와 evidence를 연결합니다.



Viewpoint는 그림 장식이 아니라 특정 concern을 어떤 model과 규칙으로 설명할지 정하는 계약입니다.

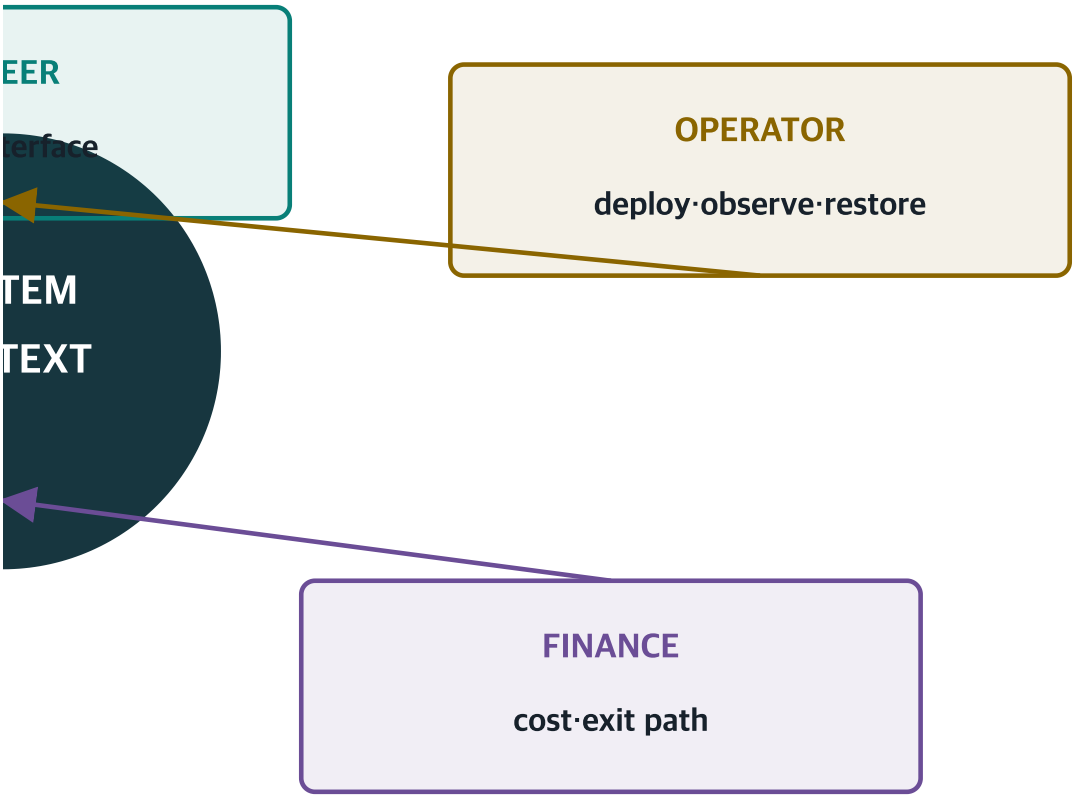
그림 3. 사용자·개발·운영·보안·재무의 concern을 필요한 view와 evidence에 연결합니다.

사용자·개발·운영·보안·재무 관점을 분리하고 필요한 view와 evidence를 연



입력 구성입니다

연결합니다.



핵심 원칙: 모든 사람에게 같은 그림을 보여 주는 것이 공통 이해는 아닙니다.

사용자는 recovery와 latency를, 운영자는 deploy·observe·restore를, 보안 담당자는 trust·least privilege를, 재무 담당자는 cost·exit path를 봅니다. 하나의 component 그림에 모든 정보를 넣으면 abstraction이 섞여 누구도 정확히 검토하기 어렵습니다.

관점	합성 사례	확인할 증거
User	flow·latency·recovery	context·quality view
Engineer	change·interface·ownership	container·module view
Operator	deploy·failure·restore	deployment·operations view
Security	trust·data·threat	security view
Finance	cost·managed responsibility·exit	cost/dependency view

흔한 오답: 한 장에 user journey, module, network subnet, DB table, cost를 모두 넣습니다.

고친 예: Concern별 view를 분리하고 stable element ID와 relationship label로 서로 연결합니다.

그림을 보며 4단계 연습

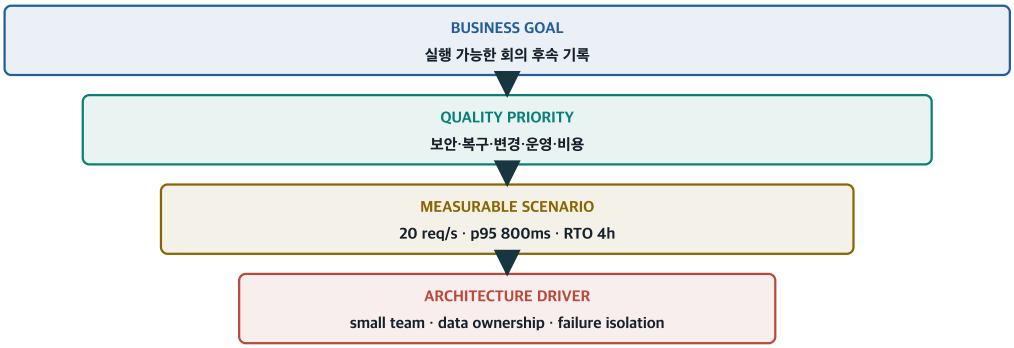
1. Stakeholder 다섯 명을 적습니다.
2. 각 concern 두 개를 씁니다.
3. Concern에 맞는 view와 evidence를 고릅니다.
4. 같은 ID가 여러 view에서 같은 뜻인지 확인합니다.

7. Goal을 architecture driver로 내리기

M13-01 · 04

품질 단어를 측정 가능한 아키텍처 동인으로 바꿉니다

Business goal에서 우선순위를 정하고 충돌과 response measure가 있는 시나리오로 내려갑니다.



'빠르고 안전하게'는 동인이 아닙니다. 누가 어떤 상황에서 무엇을 자극하고 어느 수치로 판정할지 써야 합니다.

그림 4. Goal에서 quality priority와 측정 시나리오를 거쳐 선택을 움직이는 architecture driver를 만듭니다.

Business goal에서 우선순위를 정하고 충돌과 response measure가 있는

BUSINES

실행 가능한 호

QUALITY

보안·복구·변

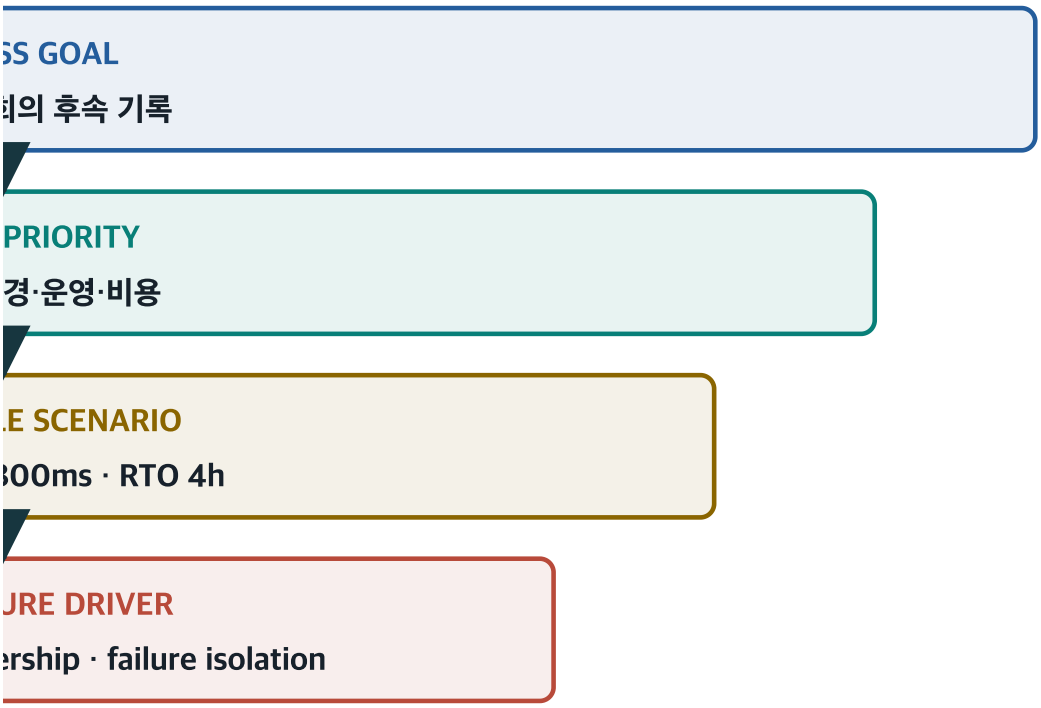
MEASURABL

20 req/s · p95 8

ARCHITECTU

small team · data owne

시나리오로 내려갑니다.


SS GOAL
의 후속 기록
PRIORITY
경·운영·비용
E SCENARIO
00ms · RTO 4h
URE DRIVER
rship · failure isolation

핵심 원칙: ‘빠르고 안전하게’는 목표일 뿐 아직 선택 기준이 아닙니다.

Driver는 business outcome과 quality conflict, hard constraint, data·integration·team·cost 조건을 함께 가져야 합니다. ‘cloud-native’, ‘AI-ready’, ‘scalable’ 같은 형용사는 response measure와 선택 영향이 없으면 driver가 아닙니다.

관점	합성 사례	확인할 증거
Goal	실행 가능한 후속 기록	PRD source
Priority	security·recovery·change	rank reason
Measure	p95 800ms·RTO4h·RPO1h	synthetic target
Constraint	2 engineer·no platform team	current boundary
Driver	low ops·owned writes·failure isolation	option criterion

흔한 오답: 확장성을 중요 품질로 표시하고 예상 사용자·peak·growth를 쓰지 않습니다.

고친 예: 20 organization·200 WAU·peak20 req/s와 성장 range를 합성 profile로 두고 option별 부담과 evidence를 비교합니다.

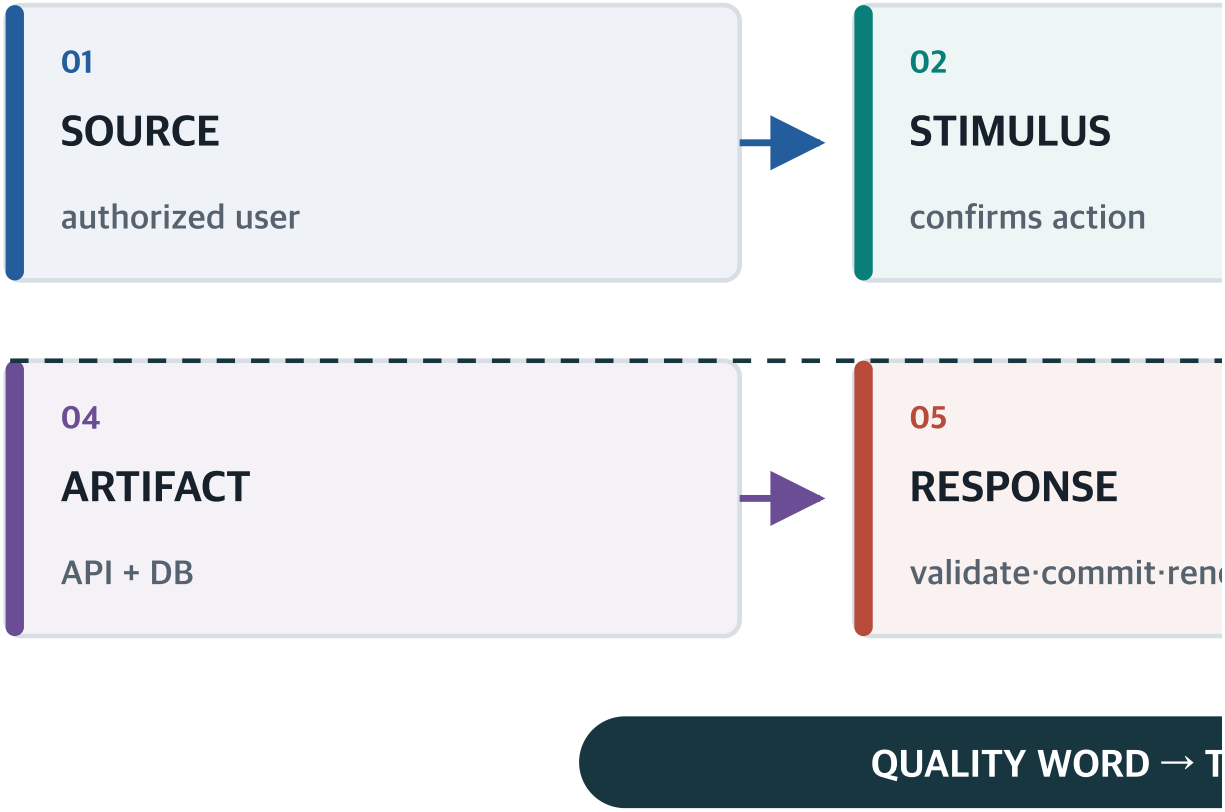
그림을 보며 4단계 연습

1. Goal 하나를 고릅니다.
2. Quality conflict 두 개를 찾습니다.
3. 숫자와 환경을 붙입니다.
4. Option 선택에 실제 영향을 주는 driver만 남깁니다.

8. 여섯 부분 품질 시나리오 쓰기

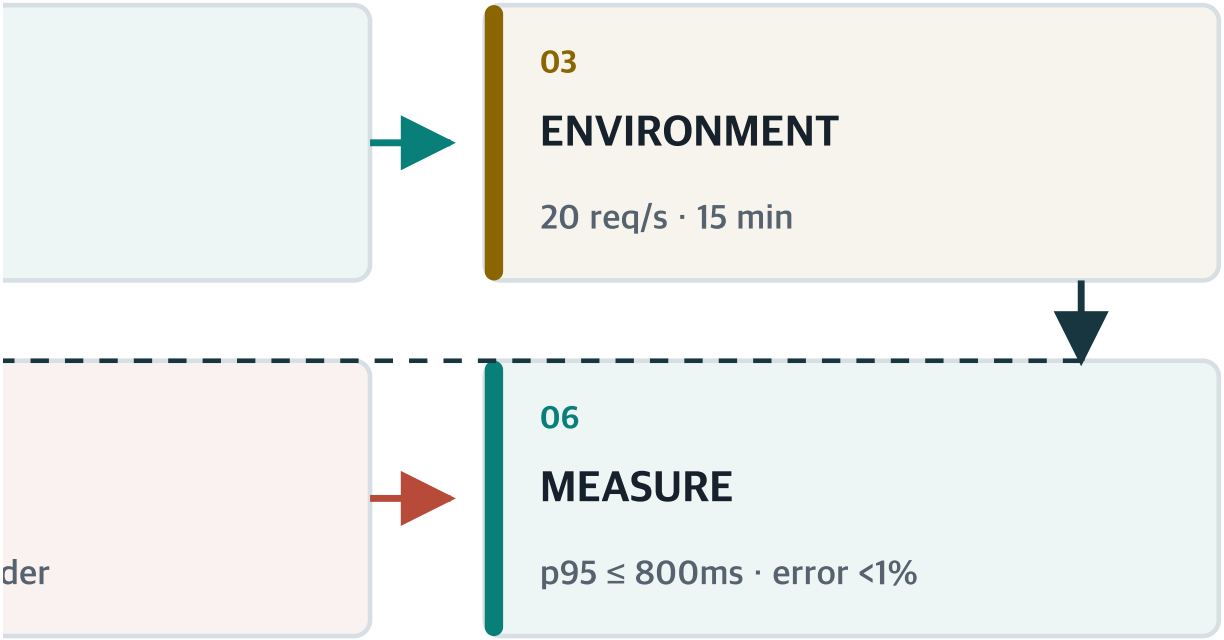
그림 5. 품질 단어를 source·stimulus·environment·artifact·response·measure로 바꾸면 test가 가능해집니다.

Source·stimulus·environment·artifact·response·measure를 한 문장과 e



[illegible]

evidence plan으로 묶습니다.



TESTABLE SCENARIO

핵심 원칙: 품질 요구는 측정 가능한 이야기로 써야 architecture를 움직입니다.

SEI QAW는 architecture가 만들어지기 전 stakeholder의 중요한 quality attribute를 scenario로 발견·우선순위화·정교화합니다. 합성 QA-01은 authorized coordinator가 peak 20 req/s 환경에서 action을 confirm할 때 API+DB가 p95 800ms 안에 commit·render하도록 씁니다.

관점	합성 사례	확인할 증거
Source	authorized coordinator	누가
Stimulus	create/confirm action	무엇이 일어남
Environment	20 req/s·15 min	어떤 조건
Artifact	Application API+DB	무엇이 영향
Response·measure	commit·render / p95≤800ms	어떻게 판정

흔한 오답: 성능이 좋아야 하고 장애가 없어야 한다고 씁니다.

고친 예: Peak load·duration·operation mix·p95·error target·evidence path와 수치의 합성 경계를 씁니다.

그림을 보며 4단계 연습

1. 품질 단어 하나를 고릅니다.
2. 여섯 부분을 모두 채웁니다.
3. Measure를 수치 또는 관찰 조건으로 씁니다.
4. Test·drill·owner를 연결합니다.

9. 같은 경계에서 네 architecture option 비교하기

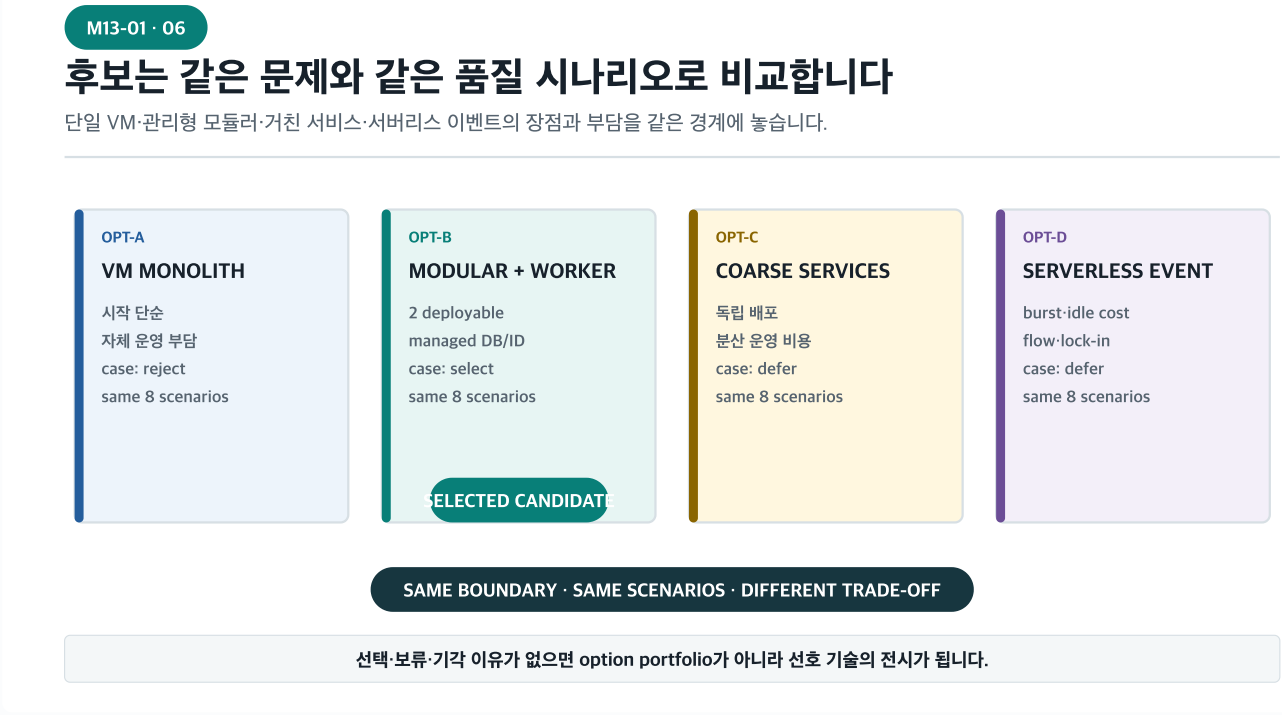


그림 6. 네 후보는 같은 user boundary와 같은 8개 quality scenario를 만족하는 방식으로 비교합니다.

단일 VM·관리형 모듈러·거친 서비스·서버리스 이벤트의 장점과 부담을 같은

OPT-A

VM MONOLITH

시작 단순

자체 운영 부담

case: reject

same 8 scenarios

OPT-B

MODULAR + WORKER

2 deployable

managed DB/ID

case: select

same 8 scenarios

SELECTED CANDIDATE

SAME BOUNDARY · SAME SCEN

은 경계에 놓습니다.

OPT-C

COARSE SERVICES

독립 배포
분산 운영 비용
case: defer
same 8 scenarios

OPT-D

SERVERLESS EVENT

burst·idle cost
flow·lock-in
case: defer
same 8 scenarios

ARIOS · DIFFERENT TRADE-OFF

핵심 원칙: 후보의 이름보다 책임·배포·데이터·실패·운영 비용의 차이를 비교합니다.

OPT-A는 시작은 단순하지만 self-managed operation이 큽니다. OPT-B는 local transaction과 module 경계를 유지하며 managed dependency로 부담을 낮춥니다. OPT-C는 독립 deployment를 얻지만 distributed complexity가 생깁니다. OPT-D는 burst와 idle cost에 유리할 수 있으나 trace·transaction·lock-in을 봐야 합니다.

관점	합성 사례	확인할 증거
OPT-A	1 deploy·self-managed DB	operation reject
OPT-B	API+worker·managed DB/ID	selected candidate
OPT-C	3~4 services·owned DB	defer until trigger
OPT-D	functions·events·managed queue	selective defer
Rule	same boundary·same scenarios	evidence comparable

흔한 오답: Monolith는 낡고 microservices는 확장 가능하다는 이름표로 점수를 줍니다.

고친 예: 각 option의 deploy unit·transaction·failure mode·team ownership·observability·TCO·exit path를 같은 표에 씁니다.

그림을 보며 4단계 연습

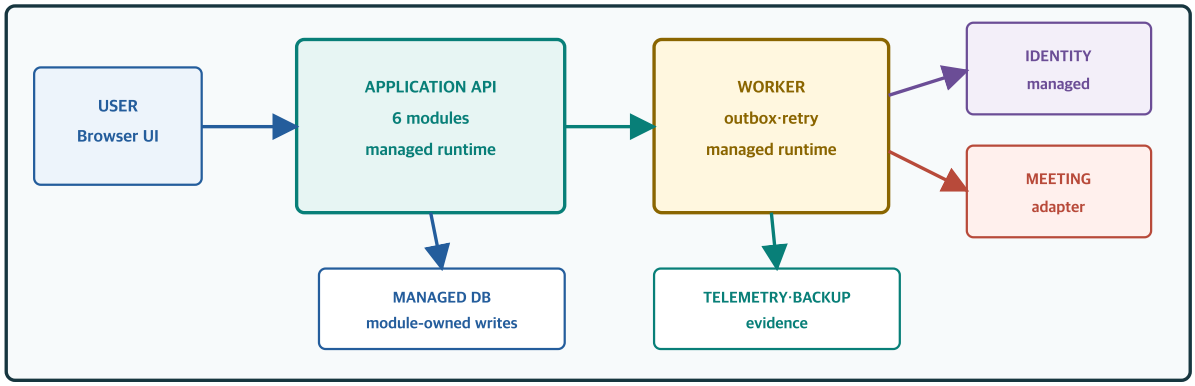
1. 실질적 후보 네 개를 만듭니다.
2. Hard constraint를 위반한 non-option을 지웁니다.
3. 같은 quality scenario를 적용합니다.
4. Selected·deferred·rejected 이유를 씁니다.

10. 선택 후보의 context와 container 그리기

M13-01 · 07

선택 후보는 두 배포 단위와 관리형 기반으로 작게 시작합니다

Browser·API·worker·identity·DB·adapter·telemetry·backup의 책임과 관계를 한 abstraction level로 표시합니다.

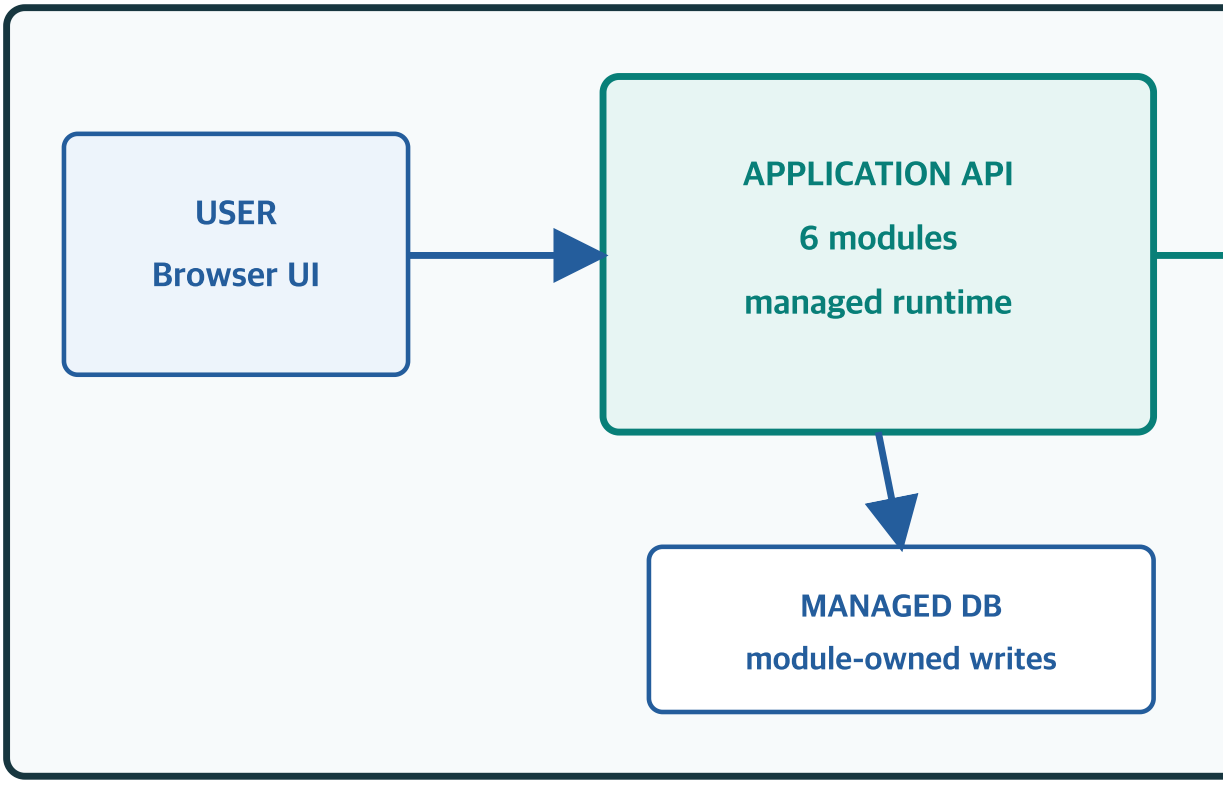


이 그림은 구조 후보입니다. 실제 provider·region·instance·capacity·security configuration은 별도 검증과 승인이 필요합니다.

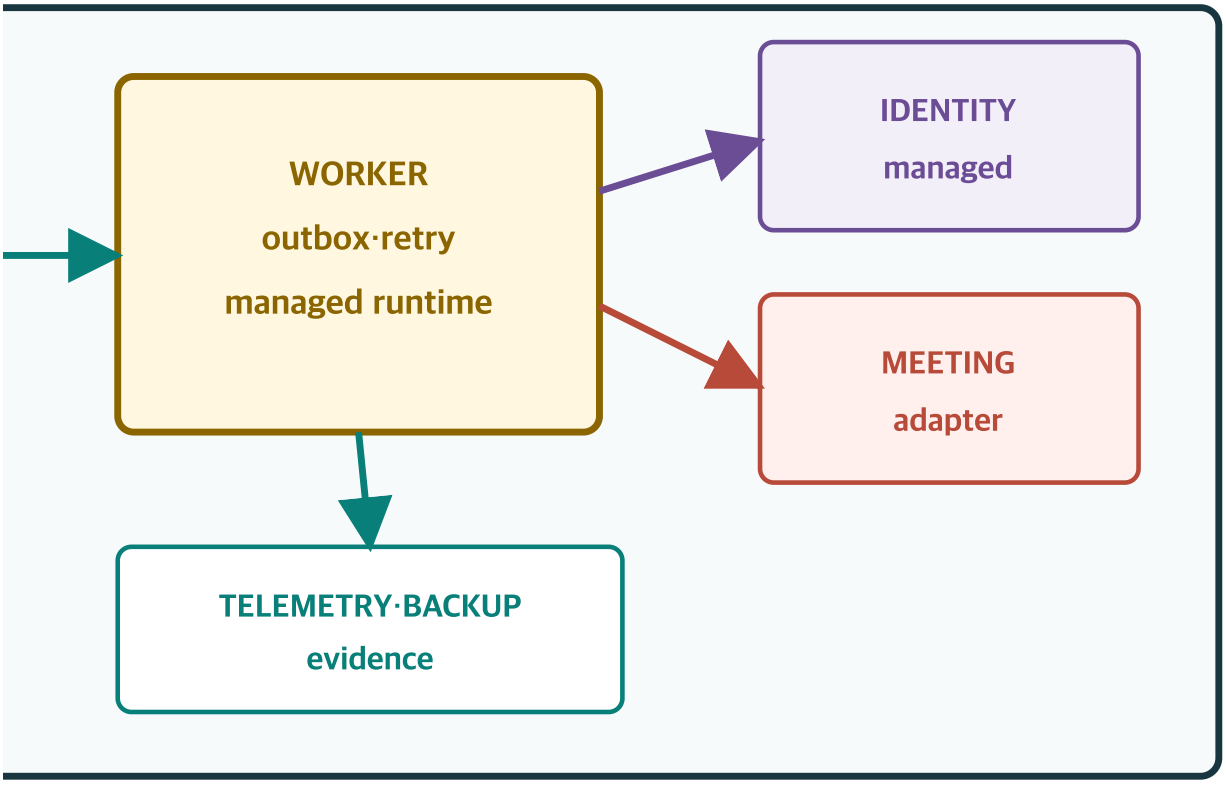
그림 7. 선택 후보는 Browser·API·worker와 managed identity·DB·telemetry·backup·external adapter로 구성됩니다.

런타임 구성과 프로세스 경계

Browser·API·worker·identity·DB·adapter·telemetry·backup의 책임과 권



관계를 한 abstraction level로 표시합니다.



핵심 원칙: 좋은 그림은 한 abstraction level에서 element type·responsibility·relationship을 설명합니다.

C4 model은 system context·container·component·code의 계층을 제공하며, 공식 guidance는 필요한 가치가 있는 view만 만들고 제목·scope·legend·element type·relationship label을 명확히 하도록 권합니다. 합성 사례는 context와 container view만으로 핵심 질문을 답합니다.

관점	합성 사례	확인할 증거
Browser	screen state·recovery	HTTPS→API
API	6 modules·authz·transaction	deployable1
Worker	outbox·retry·integration	deployable2
Managed	identity·DB·telemetry·backup	shared responsibility
External	meeting provider adapter	timeout boundary

흔한 오답: 한 화살표에 '연결됨'이라고만 쓰고 protocol·purpose·direction을 생략합니다.

고친 예: 모든 element에 name·type·responsibility를, 모든 relation에 intent·direction과 필요한 protocol을 씁니다.

그림을 보며 4단계 연습

1. System context를 먼저 그립니다.
2. Container type과 responsibility를 씁니다.
3. Relationship label을 동사로 씁니다.
4. Legend와 scope가 그림만 읽어도 보이는지 확인합니다.

11. Module 책임과 data ownership 고정하기

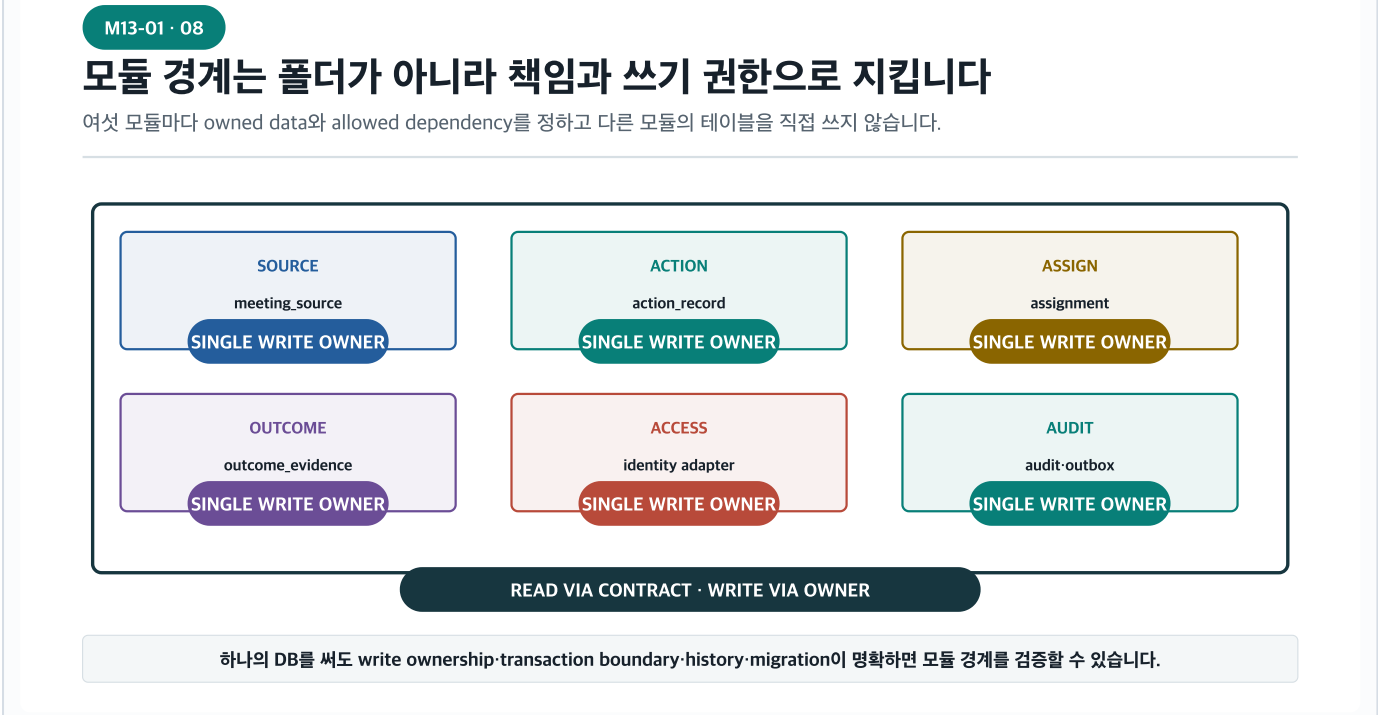
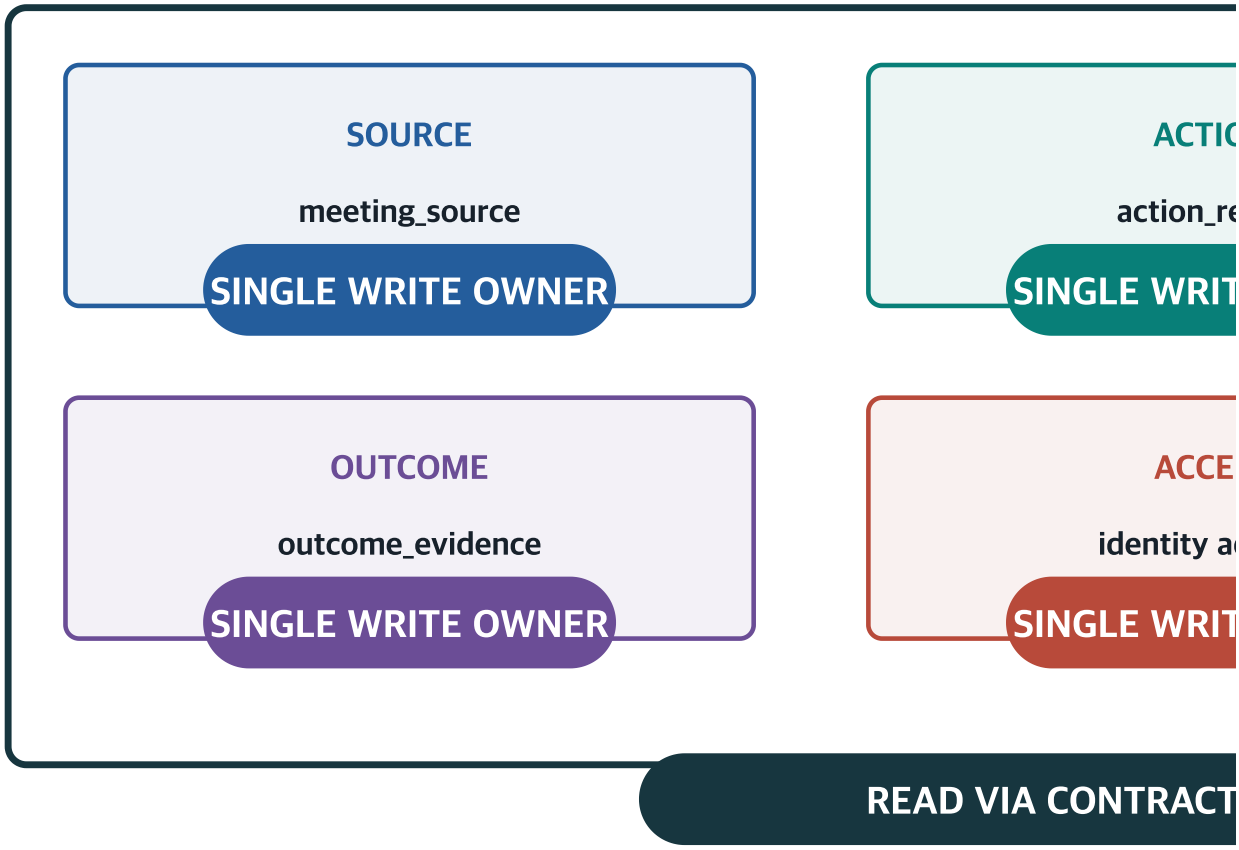
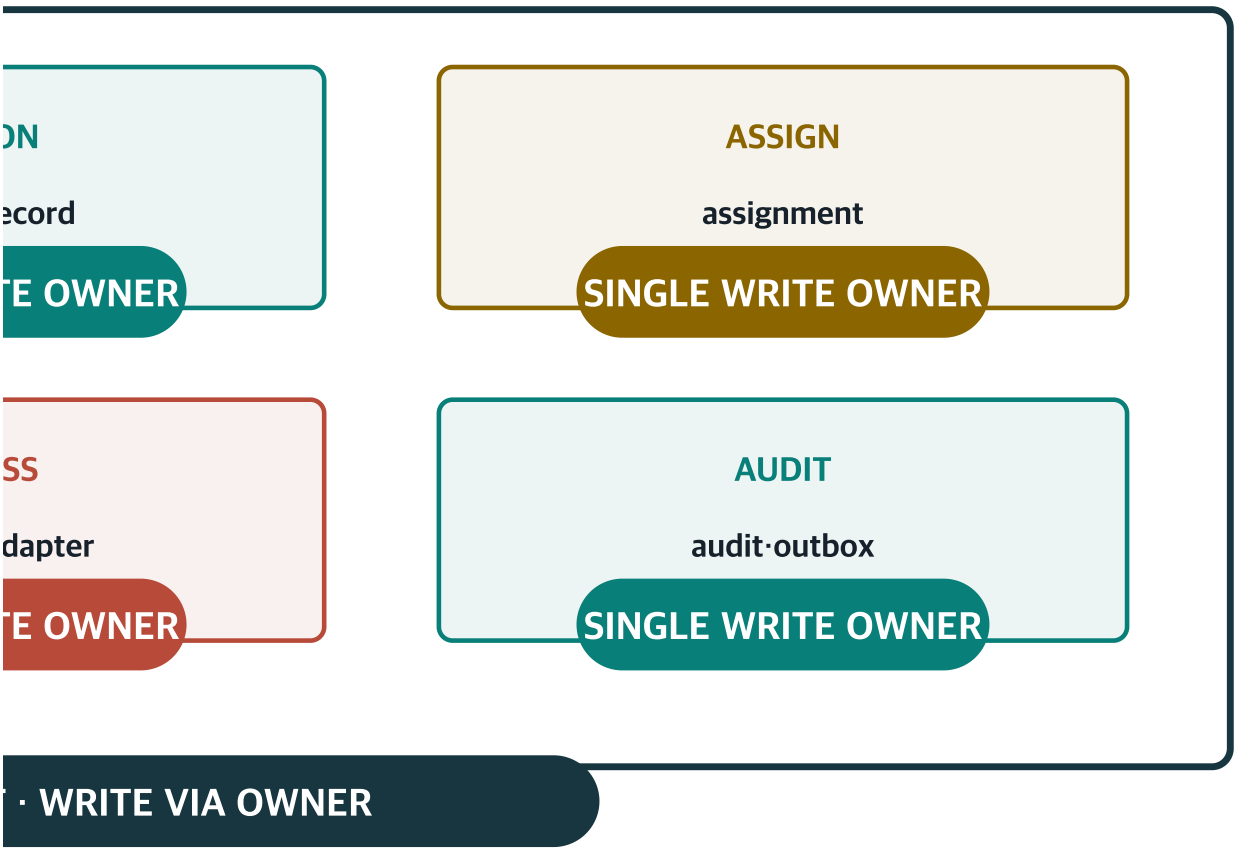


그림 8. 여섯 module은 business responsibility와 single write owner를 갖고 다른 data는 contract로 읽습니다.

여섯 모듈마다 owned data와 allowed dependency를 정하고 다른 모듈으



| 테이블을 직접 쓰지 않습니다.



핵심 원칙: Module은 폴더가 아니라 변경 책임·공개 계약·쓰기 권한의 경계입니다.

한 DB를 사용해도 Action module만 action_record를 쓰고 Assignment module만 confirmation을 쓰도록 강제할 수 있습니다. 직접 cross-module table write를 허용하면 local transaction의 단순성을 얻고도 coupling과 migration risk는 그대로 남습니다.

관점	합성 사례	확인할 증거
Meeting Source	meeting_source	import cursor
Action Record	action_record·version	core invariant
Assignment	assignment_confirmation	actor confirm
Outcome	outcome_evidence	completion state
Access·Audit	identity adapter·audit·outbox	boundary evidence

흔한 오답: Module 이름만 나누고 모든 module이 같은 table을 직접 수정합니다.

고친 예: Owned data·public operation·event·allowed dependency·transaction boundary를 쓰고 architecture test로 forbidden write를 막습니다.

그림을 보며 4단계 연습

1. Business responsibility 여섯 개를 씁니다.
2. 각 data의 write owner를 하나 고릅니다.
3. Public read/write contract를 씁니다.
4. Forbidden dependency와 검증 rule을 붙입니다.

12. Local transaction과 외부 실패 분리하기

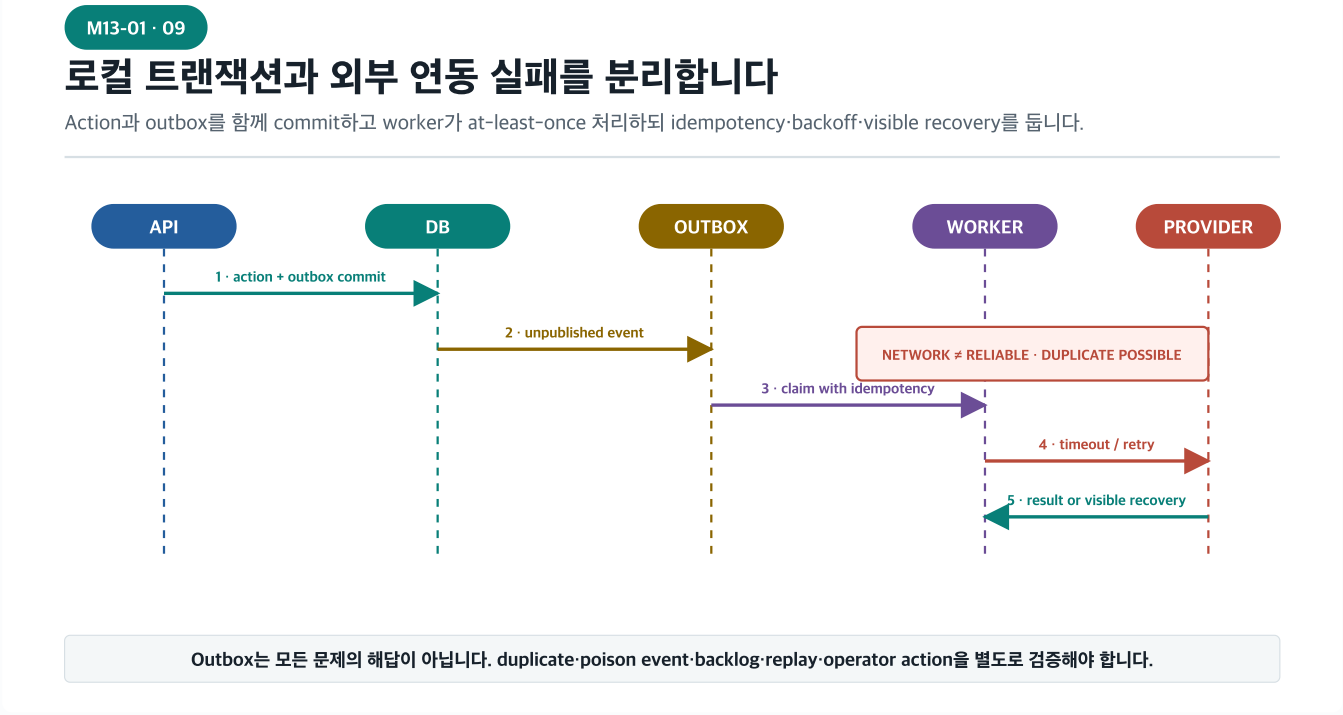
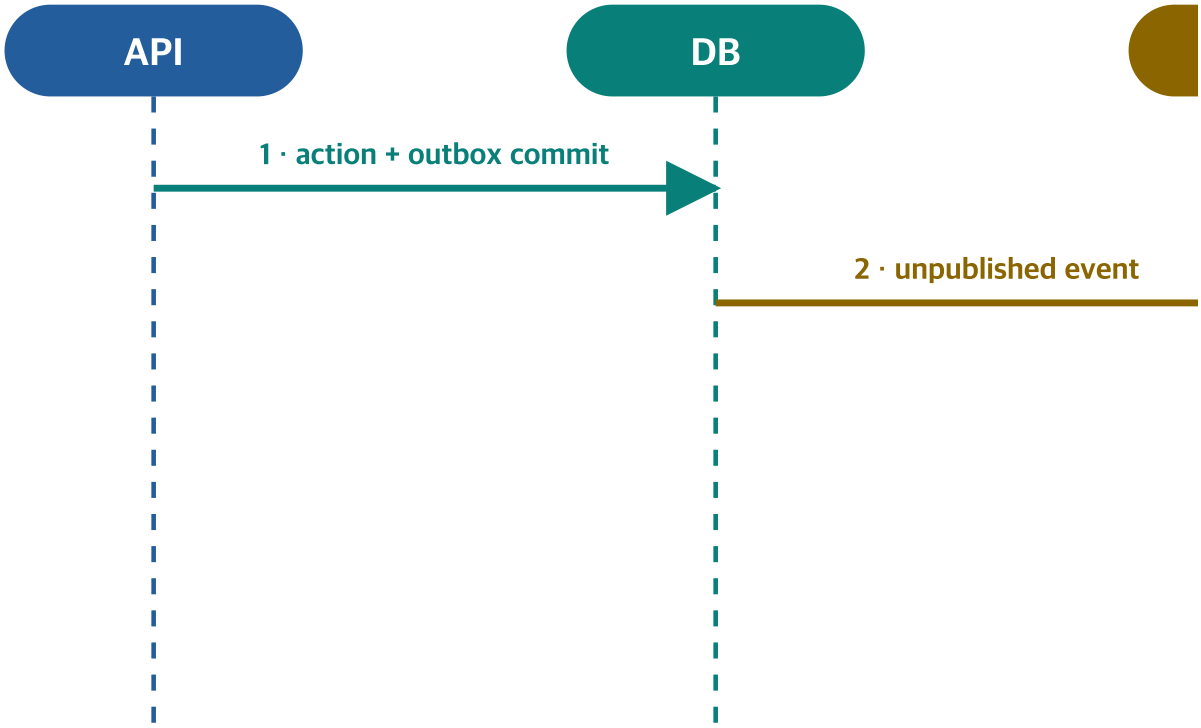


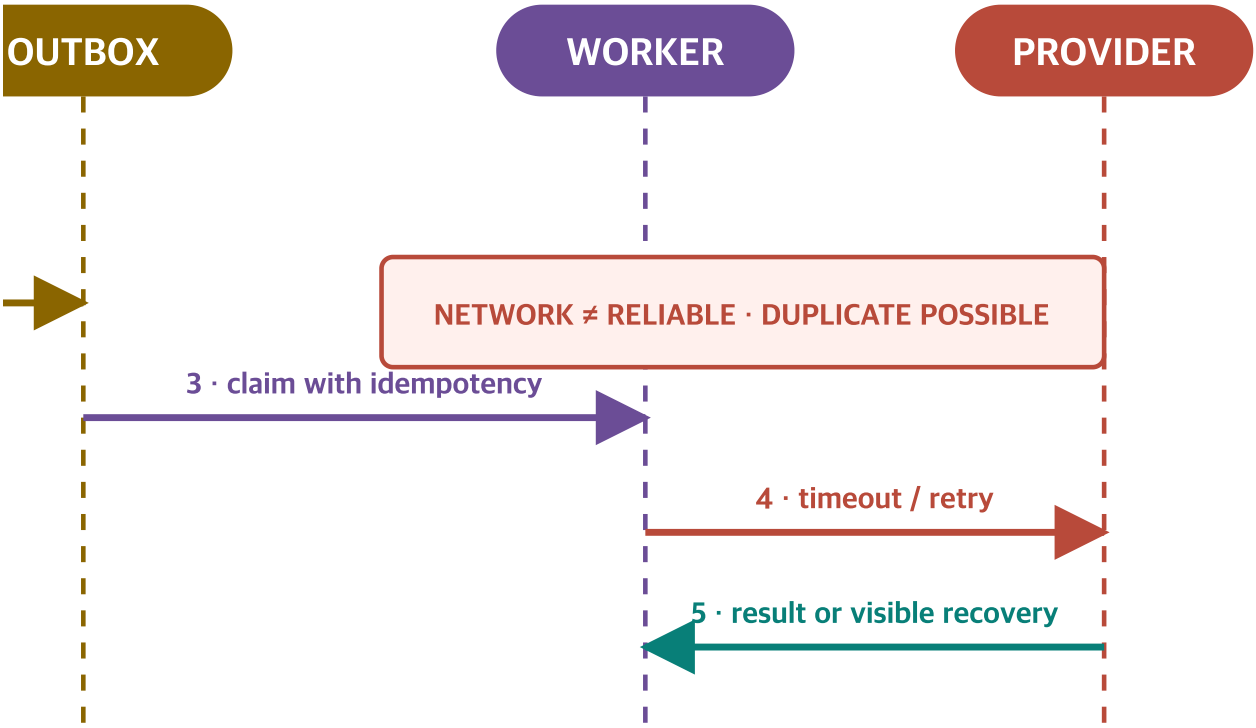
그림 9. Action과 outbox를 한 transaction에 commit한 뒤 worker가 외부 provider를 재시도합니다.

[illegible]

Action과 outbox를 함께 commit하고 worker가 at-least-once 처리하되



idempotency·backoff·visible recovery를 둡니다.



핵심 원칙: 분산 호출은 정상 응답보다 timeout·duplicate·replay·operator recovery를 먼저 그립니다.

DB commit과 external call을 한 번에 원자적으로 만들 수 없으면 dual-write gap이 생깁니다. 합성 후보는 local business record와 outbox를 함께 commit하고 worker가 at-least-once 처리합니다. Idempotency는 중복 effect를 줄이지만 backlog·poison message·ordering은 별도 통제가 필요합니다.

관점	합성 사례	확인할 증거
Commit	action+outbox atomic	local invariant
Delivery	at-least-once	duplicate possible
Consumer	idempotency key	effect suppression
Failure	timeout·backoff·dead letter	bounded retry
Recovery	visible status·replay·reconcile	operator evidence

흔한 오답: Message queue를 쓰면 exactly-once이고 장애가 사라진다고 씁니다.

고친 예: Delivery semantics와 end-to-end duplicate boundary를 밝히고 retry budget·DLQ·replay·reconciliation을 설계합니다.

그림을 보며 4단계 연습

1. Local invariant를 찾습니다.
2. External side effect를 분리합니다.
3. Duplicate key와 retry policy를 씁니다.
4. Poison·backlog·replay 운영 절차를 씁니다.

13. Security·privacy를 trust boundary에 넣기

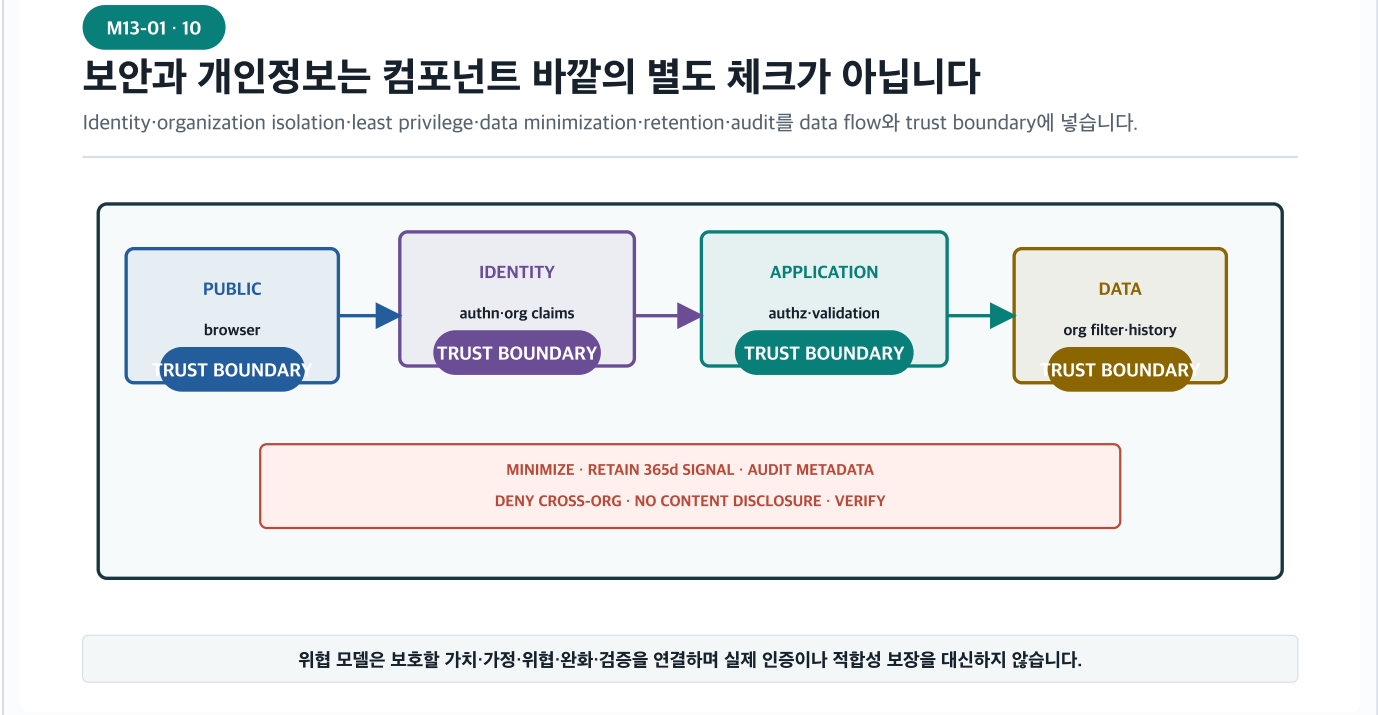
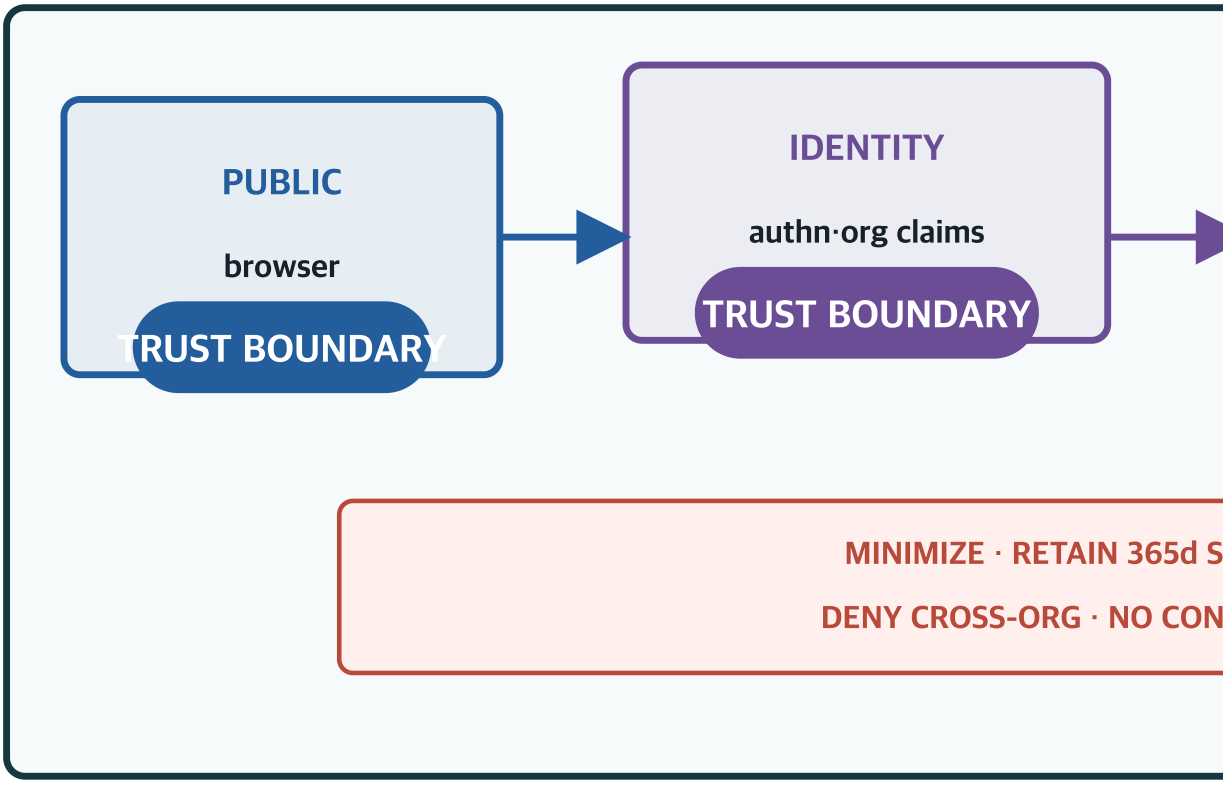
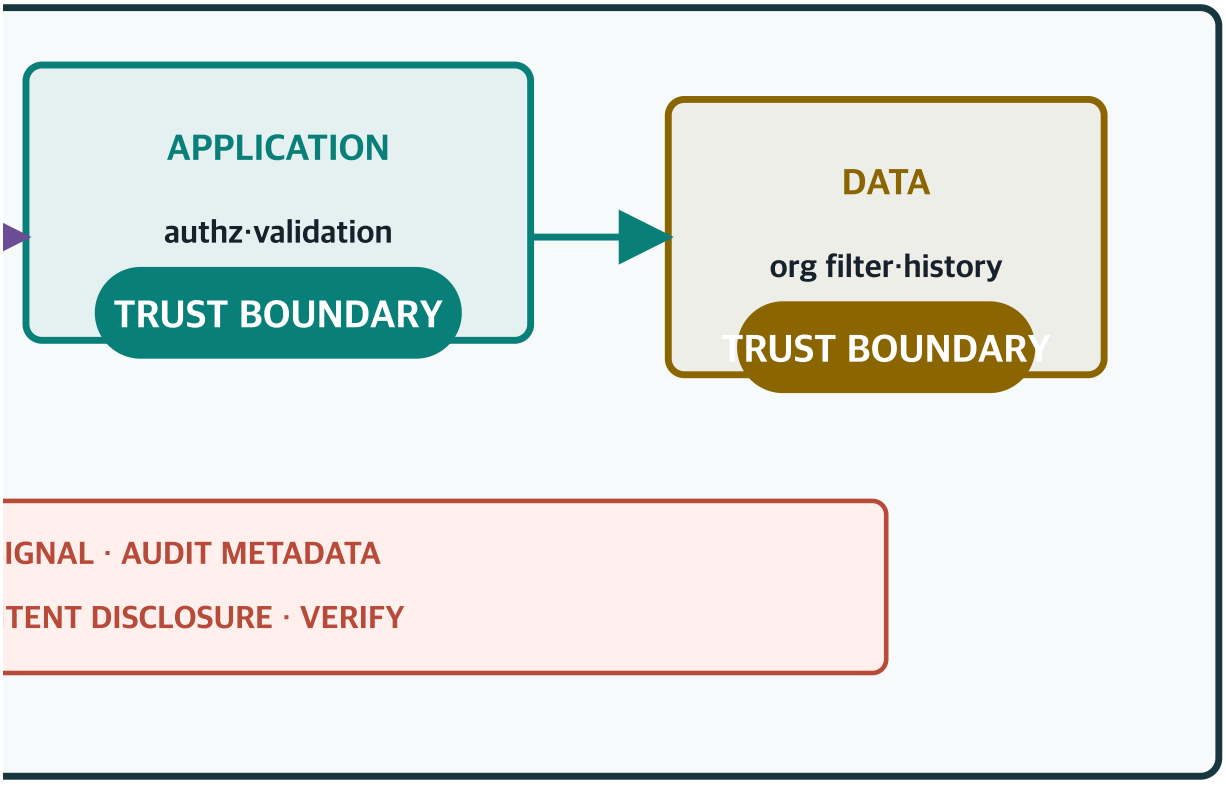


그림 10. Public·identity·application·data 경계에서 authn·authz·organization isolation·minimization을 검토합니다.

Identity·organization isolation·least privilege·data minimization·reten



tion·audit를 data flow와 trust boundary에 넣습니다.



핵심 원칙: 보안은 architecture 이후에 붙이는 제품이 아니라 경계·data·operation에 내장된 결정입니다.

NIST SP 800-160은 trustworthy secure system engineering을 lifecycle 전체의 engineering problem으로 다룹니다. OWASP threat modeling은 보호할 가치·가정·위협·완화·검증을 system context로 구조화합니다. 합성 사례는 cross-org access 0건, raw meeting content 최소화, retention 365일 신호를 검증 backlog로 둡니다.

관점	합성 사례	확인할 증거
Identity	authn·org claims	managed boundary
Authorization	deny default·resource scope	API
Data	minimize·retain·dispose	owned lifecycle
Audit	safe metadata·no payload	investigation
Evidence	cross-org tests·threat review	not certification

흔한 오답: TLS와 managed identity를 쓰므로 security와 privacy가 완료됐다고 표시합니다.

고친 예: Trust boundary·asset·threat·mitigation·verification·residual risk를 data flow와 operation별로 연결합니다.

그림을 보며 4단계 연습

1. Trust boundary를 표시합니다.
2. 보호할 asset을 씁니다.
3. Threat와 mitigation을 연결합니다.
4. Test evidence와 certification boundary를 분리합니다.

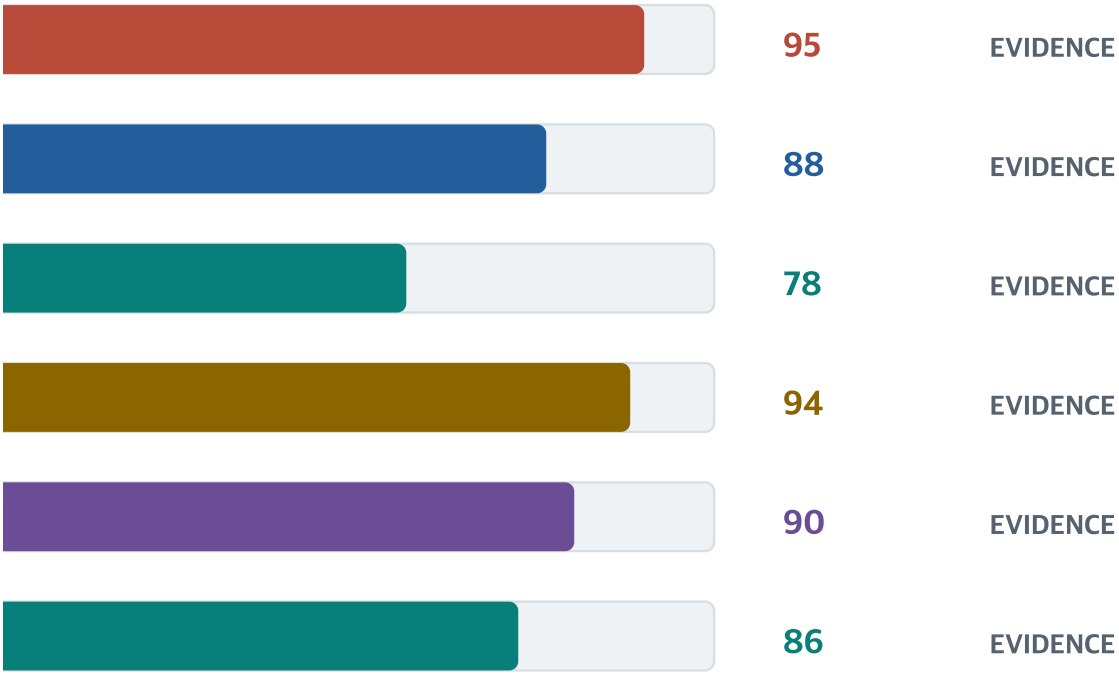
14. Quality·operations·cost 균형 맞추기

그림 11. Security·reliability·performance·operability·cost·changeability를 evidence confidence와 함께 봅니다.

[illegible]

Security·reliability·performance·operability·cost·changeability를 우선순

순위와 evidence confidence로 함께 봅니다.



NAME TRADE-OFFS

핵심 원칙: 잘 설계된 system은 한 품질을 최대로 만든 system이 아니라 우선순위와 trade-off가 드러난 system입니다.

AWS·Azure·Google의 Well-Architected guidance는 security·reliability·performance·operations·cost와 지속 개선을 공통적으로 다룹니다. 특정 cloud 제품을 정답으로 가져오기보다 현재 workload에 relevant한 질문을 고르고 서로 충돌하는 영향과 team ability를 기록합니다.

관점	합성 사례	확인할 증거
Security	org isolation·least privilege	test
Reliability	restart·RTO·RPO	drill
Performance	20rps·p95 800ms	load
Operability	2 deploy·diagnose 30m	tabletop
Cost·change	USD600 signal·module impact	review

흔한 오답: 모든 pillar에 최고 점수를 주고 trade-off가 없다고 씁니다.

고친 예: Priority·evidence source·confidence·known weakness·residual risk와 다른 축의 손실을 함께 씁니다.

그림을 보며 4단계 연습

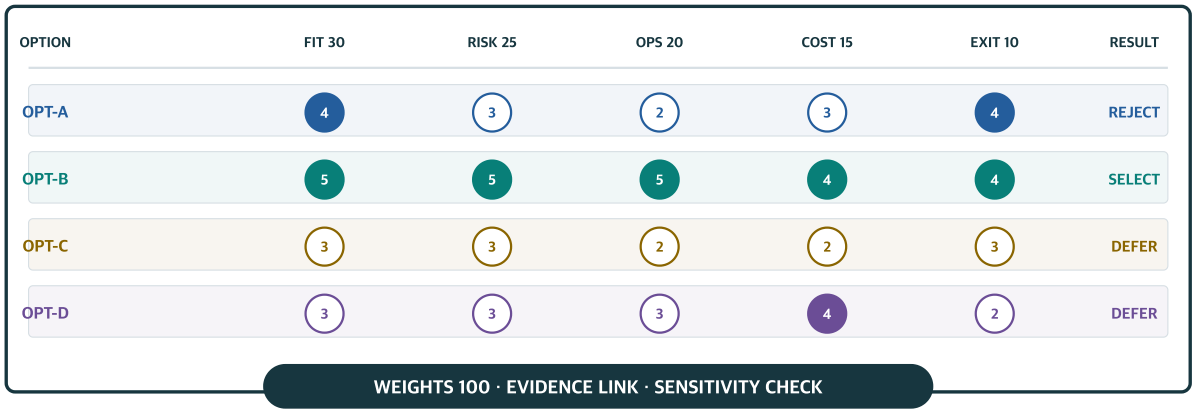
1. 중요한 품질 여섯 개를 고릅니다.
2. 각 measure와 evidence를 붙입니다.
3. 한 축을 높일 때 낮아지는 축을 찾습니다.
4. 현재 팀이 운영할 수 있는지 재검토합니다.

15. Option matrix와 sensitivity 검토하기

M13-01 · 12

가중치와 점수보다 근거와 민감도 검토가 중요합니다

같은 기준으로 후보를 평가하고 점수의 출처·confidence·가중치 변화에 따른 순위 변화를 확인합니다.



OPT-B가 가장 높아도 자동 승인되지 않습니다. hard constraint·residual risk·검증 backlog가 Gate를 통과해야 합니다.

그림 12. Fit·risk·operations·cost·exit에 weight를 주되 score 근거와 sensitivity를 함께 검토합니다.

같은 기준으로 후보를 평가하고 점수의 출처·confidence·가중치 변화에 따

OPTION	FIT 30	RISK 25
OPT-A	4	3
OPT-B	5	5
OPT-C	3	3
OPT-D	3	3

WEIGHTS 100 · EVIDENCE

른 순위 변화를 확인합니다.

OPS 20	COST 15	EXIT 10	RESULT
2	3	4	REJECT
5	4	4	SELECT
2	2	3	DEFER
3	4	2	DEFER

LINK · SENSITIVITY CHECK

핵심 원칙: 점수는 대화를 구조화하지만 decision authority를 대신하지 않습니다.

Weighted score는 복잡한 정보를 한눈에 보게 하지만 weight를 조금 바꾸면 순위가 달라질 수 있습니다. Hard constraint를 먼저 적용하고, score마다 evidence link와 confidence를 붙이며, 가중치 변화·unknown·residual risk를 sensitivity review로 확인합니다.

관점	합성 사례	확인할 증거
Fit	30	business+quality
Risk	25	failure+security
Operations	20	team+deploy
Cost	15	TCO signal
Exit	10	lock-in+reversibility

흔한 오답: OPT-B의 weighted score가 가장 높으므로 자동 승인합니다.

고친 예: Hard constraint·evidence confidence·sensitivity·residual risk·decision authority를 별도 Gate로 둡니다.

그림을 보며 4단계 연습

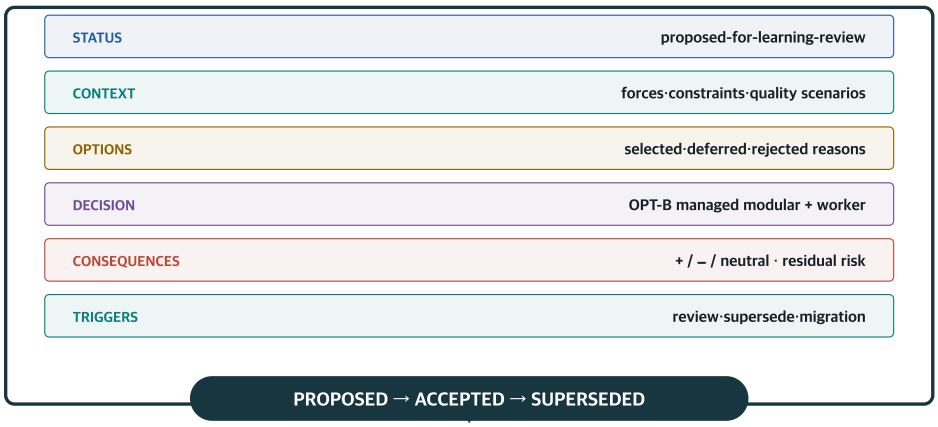
1. Criteria를 driver에서 도출합니다.
2. Weight 합을 100으로 맞춥니다.
3. Score마다 evidence를 연결합니다.
4. Weight ±10 변화와 hard constraint를 검토합니다.

16. ADR로 맥락·선택·결과 보존하기

M13-01 · 13

ADR은 선택 결과보다 당시의 힘과 결과를 보존합니다

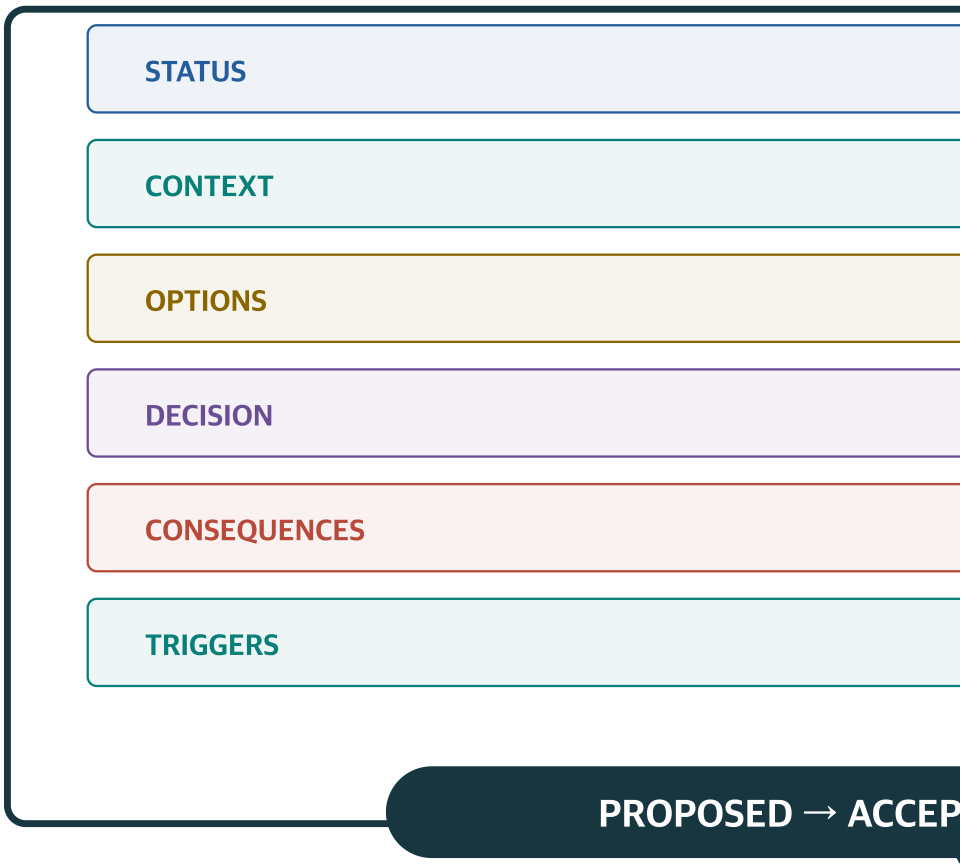
Context·options·decision·status·긍정/부정/중립 consequence·trigger를 짧고 versioned하게 기록합니다.



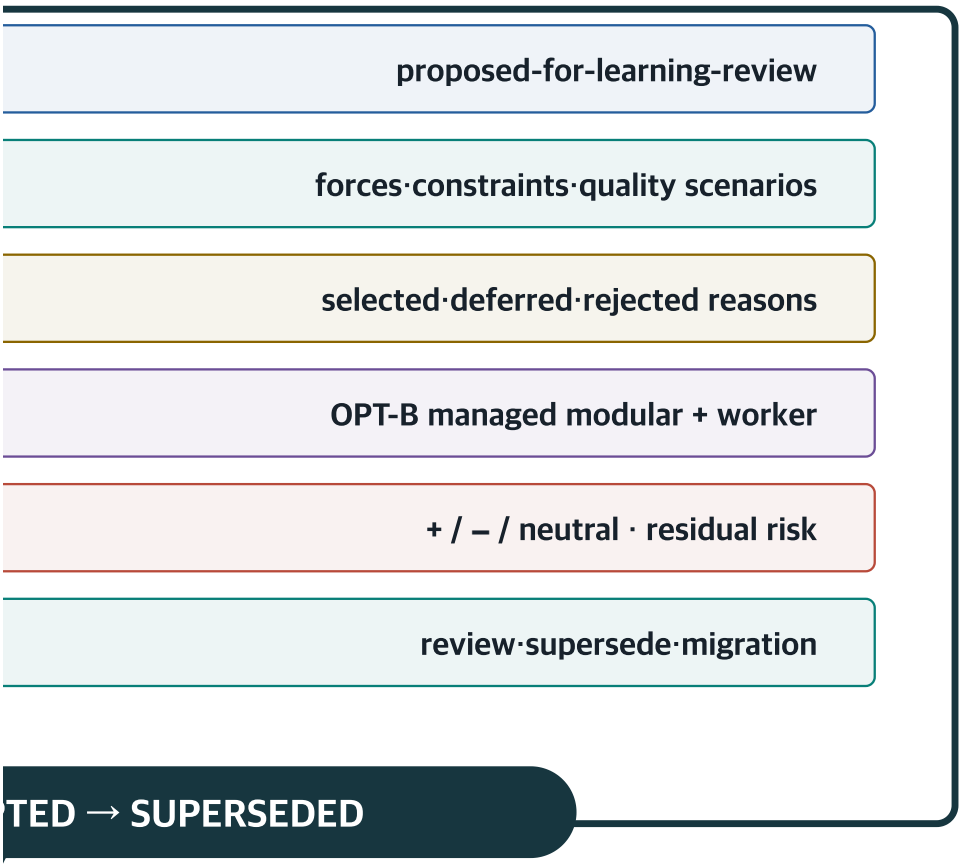
결정이 바뀌면 과거 ADR을 지우지 않고 superseded로 남겨 future team이 이유와 전환 비용을 이해하게 합니다.

그림 13. ADR은 status·context·option·decision·consequence·trigger를 짧고 versioned하게 보존합니다.

Context·options·decision·status·긍정/부정/중립 consequence·trigger를



짧고 versioned하게 기록합니다.



핵심 원칙: ADR은 future team에게 '무엇'보다 '왜'와 '무엇을 감수했는지'를 남깁니다.

Michael Nygard의 ADR 제안은 architecturally significant decision을 작은 text record로 보존하고 context·decision·status·consequence를 기록합니다. 결정이 바뀌면 과거를 삭제하지 않고 superseded link를 남겨 당시의 force와 전환 시점을 이해하게 합니다.

관점	합성 사례	확인할 증거
Status	proposed-for-learning-review	not accepted
Context	goal·driver·constraint·risk	value-neutral
Options	A/B/C/D	reason
Decision	OPT-B	active sentence
Consequences	+/–/neutral·trigger	future context

흔한 오답: ADR에 '모듈러 모놀리스를 사용한다' 한 줄만 씁니다.

고친 예: 왜 지금 OPT-B인지, 왜 다른 후보는 defer/reject인지, 무엇을 잃고 어떤 evidence와 trigger로 다시 볼지 씁니다.

그림을 보며 4단계 연습

1. Decision title을 noun phrase로 씁니다.
2. Context의 tension을 중립적으로 씁니다.
3. Decision을 능동 문장으로 씁니다.
4. 긍정·부정·중립 consequence와 trigger를 씁니다.

17. Fitness evidence와 evolution trigger 만들기

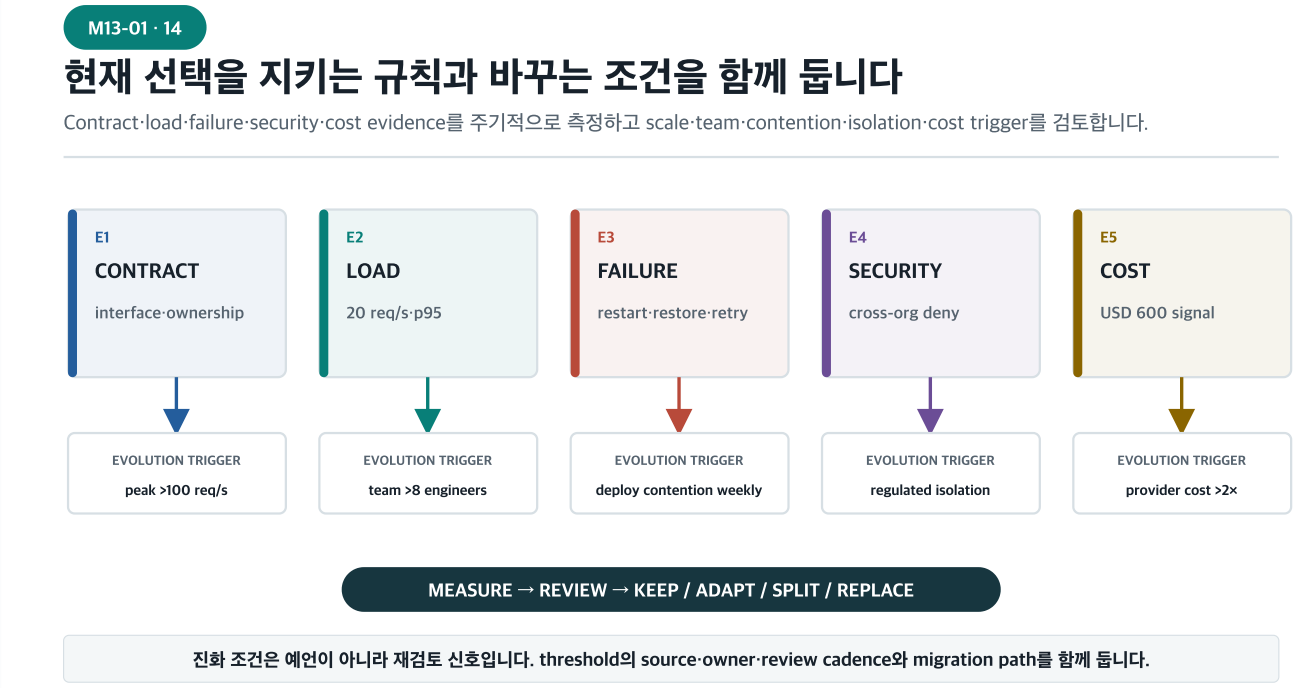
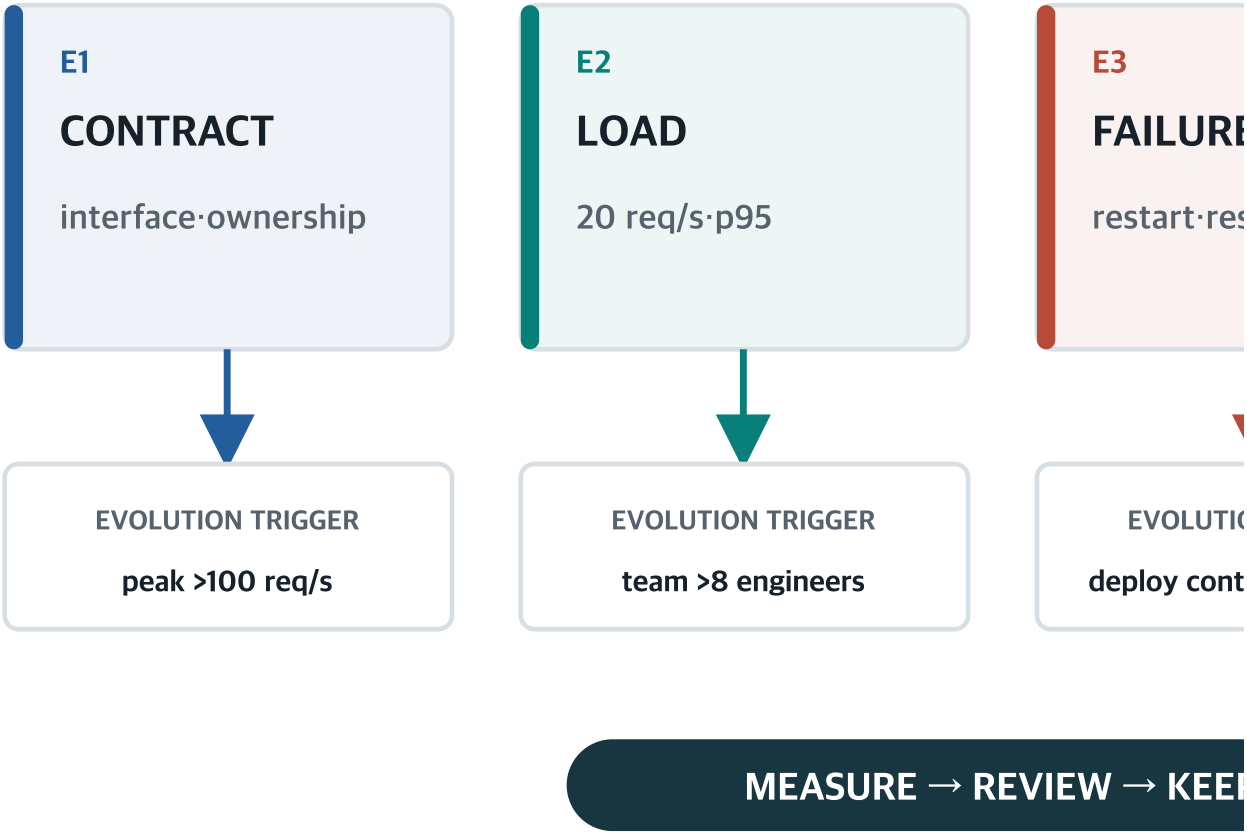
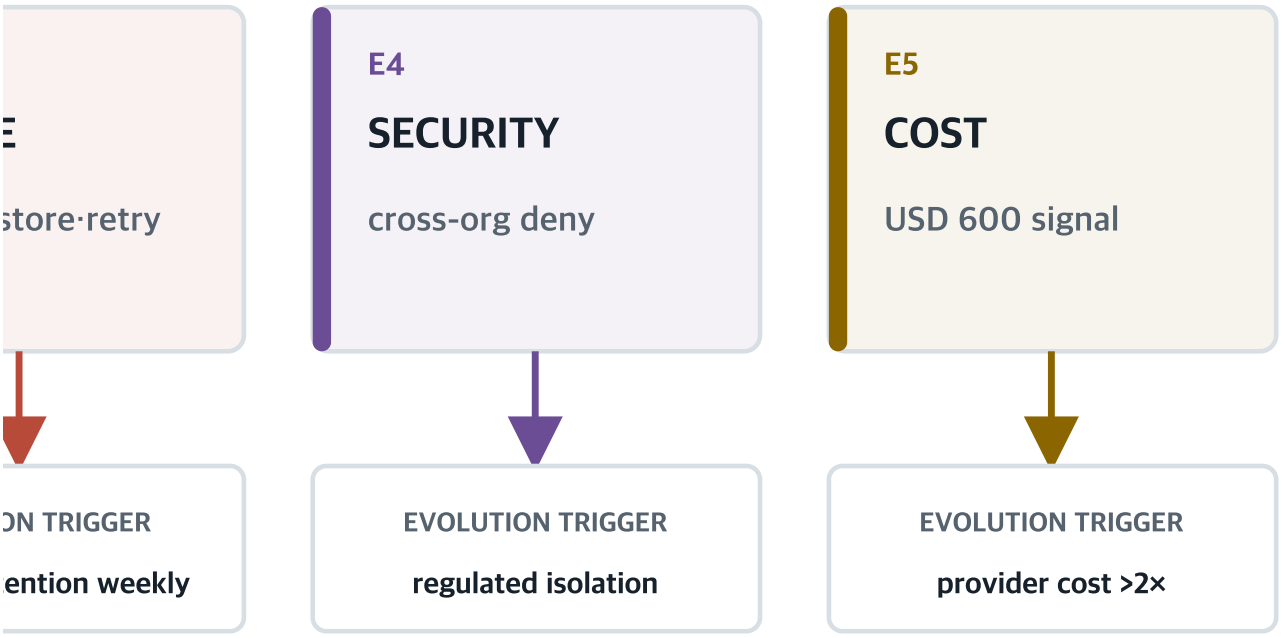


그림 14. 현재 선택을 지키는 evidence와 architecture를 다시 볼 trigger를 한 쌍으로 관리합니다.

Contract·load·failure·security·cost evidence를 주기적으로 측정하고 sca



ıle·team·contention·isolation·cost trigger를 검토합니다.



P / ADAPT / SPLIT / REPLACE

핵심 원칙: Architecture는 한 번 고르고 끝나는 그림이 아니라 변화에 맞춰 검증되는 decision system입니다.

Contract test는 boundary를, load test는 performance를, failure/restore drill은 recovery를, security test는 isolation을, cost review는 value를 확인합니다. Peak>100rps, team>8, weekly deploy contention, regulated isolation, provider cost>2× 같은 신호가 나타나면 split·replace·adapt를 재검토합니다.

관점	합성 사례	확인할 증거
Contract	interface·ownership	continuous
Load	20rps·p95	before release
Failure	restart·restore·retry	quarterly drill
Security	cross-org deny	change+review
Cost	unit·monthly signal	monthly review

흔한 오답: 언젠가 사용자가 많아지면 microservices로 바꾼다고 씁니다.

고친 예: Scale·team·contention·isolation·cost threshold와 source·owner·review cadence·migration path를 씁니다.

그림을 보며 4단계 연습

1. Architecture claim 다섯 개를 고릅니다.
2. 각 claim의 fitness evidence를 정합니다.
3. Keep/adapt/split/replace trigger를 씁니다.
4. Owner·cadence·migration path를 붙입니다.

18. Architecture Decision Gate와 M13-02 handoff 닫기

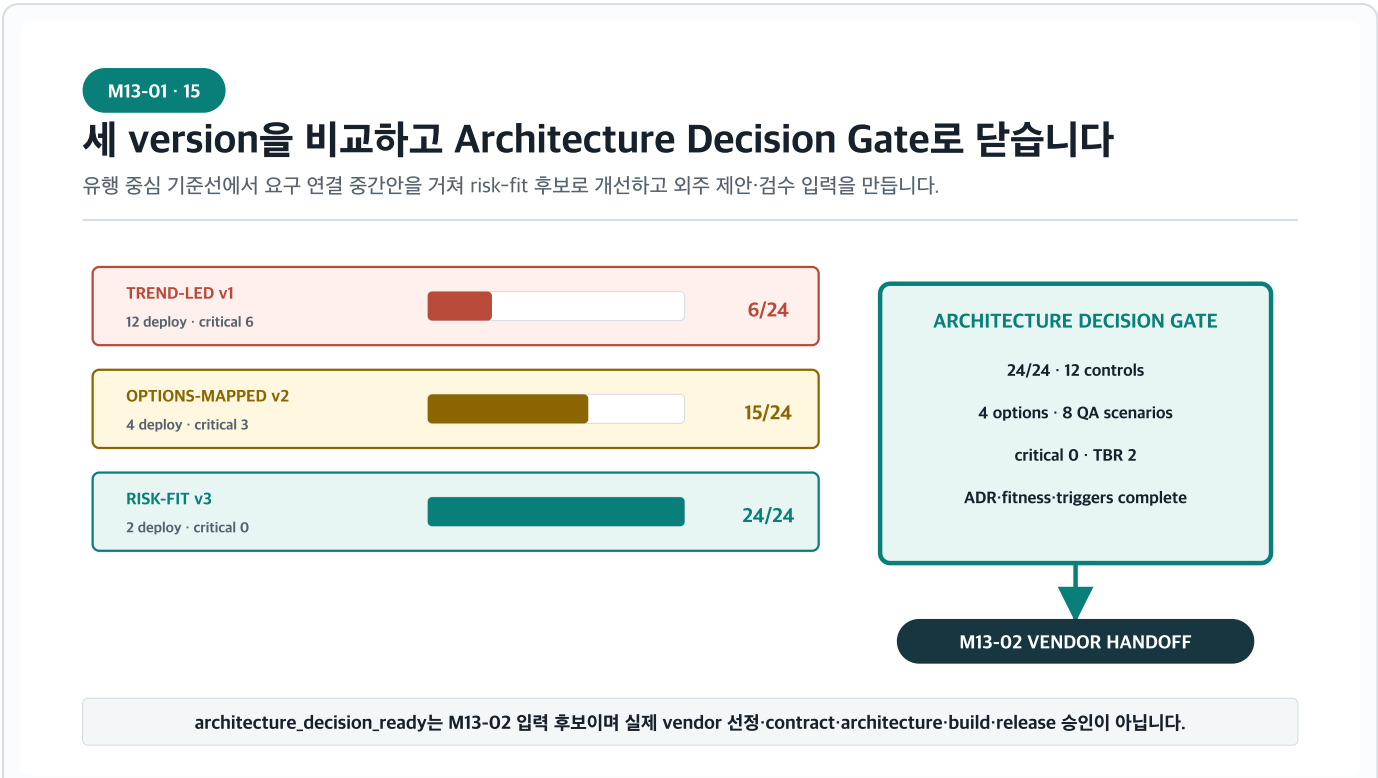
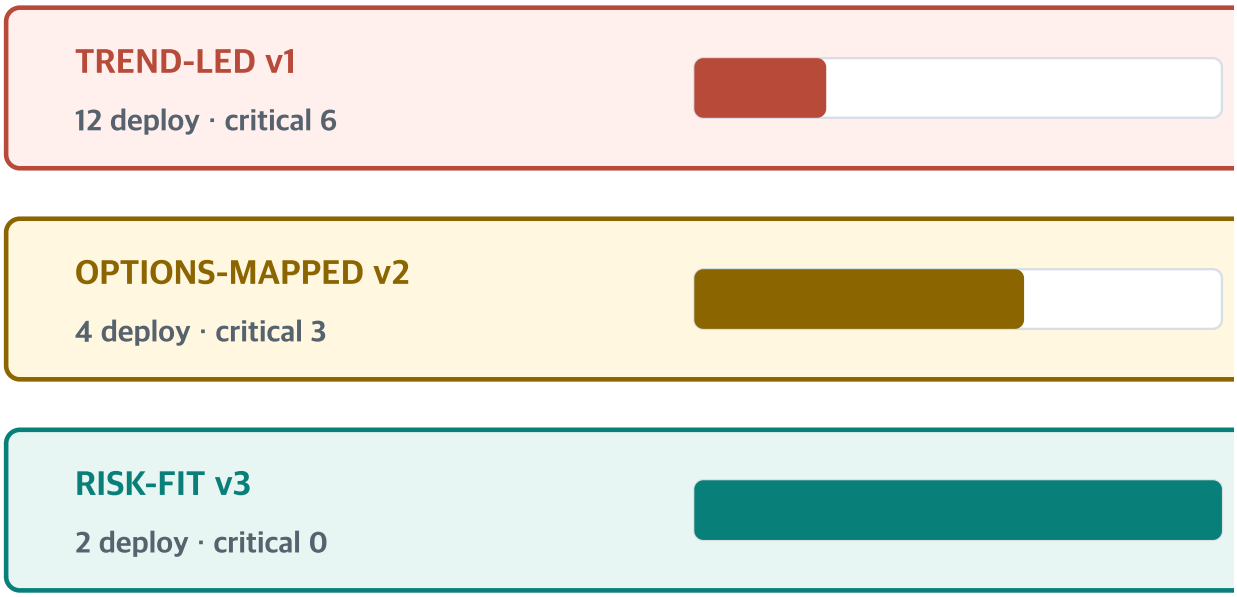


그림 15. 6/24→15/24→24/24로 개선하고 critical risk 0·TBR 2·ADR·fitness·trigger를 Gate에 남깁니다.

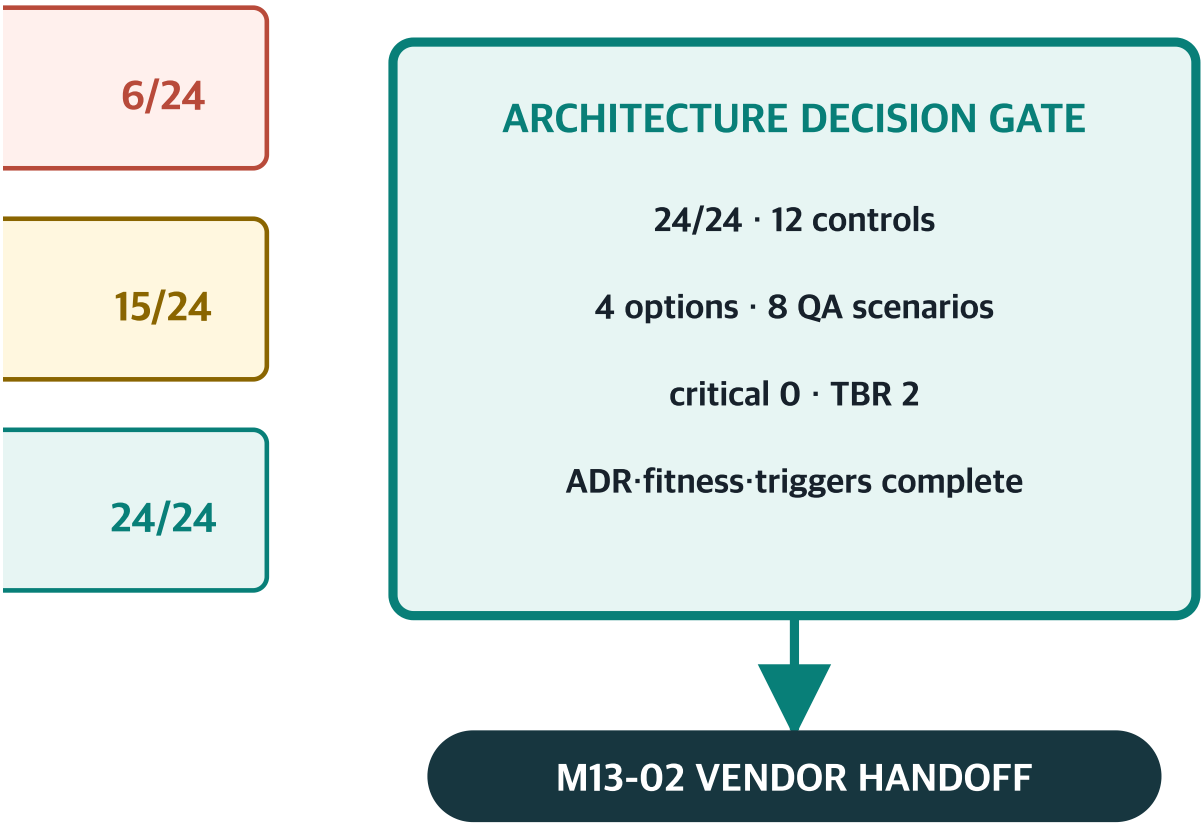
3개 Version을 나란히 Architecture

유행 중심 기준선에서 요구 연결 중간안을 거쳐 risk-fit 후보로 개선하고 외



Decision Gate 입력 구성

주 제안·검수 입력을 만듭니다.



핵심 원칙: Gate는 확신의 말투를 versioned evidence·residual risk·authority로 바꿉니다.

최종 합성 후보는 24 scenario, 12 controls, 네 lane 각 6개, 4 options, 8 quality scenarios, 6 modules, 8 components, 2 deployables, open critical risk 0, TBR 2를 요구합니다. M13-02에는 외주사가 같은 기준으로 제안하고 구현 evidence를 제출할 수 있는 묶음을 넘깁니다.

관점	합성 사례	확인할 증거
Trend-led v1	6/24	critical6·12 deploy
Mapped v2	15/24	critical3·4 deploy
Risk-fit v3	24/24	critical0·2 deploy
Gate	ADR·fitness·trigger	TBR2
Handoff	RFP·acceptance·inspection	M13-02

흔한 오답: architecture_decision_ready이므로 vendor 선정과 개발 착수를 승인합니다.

고친 예: Decision candidate와 unresolved TBR·evidence backlog·approval boundary를 M13-02 제안 요청·검수 입력으로 넘깁니다.

그림을 보며 4단계 연습

1. 세 variant를 실행합니다.
2. Fixed·regressed와 critical risk를 비교합니다.
3. Gate evidence와 open TBR을 기록합니다.
4. M13-02 vendor response·inspection item을 만듭니다.

19. 12개 Architecture Decision Control

ID	통제	최소 evidence
CTRL-01	M12-04 linked package handoff	Architecture input manifest
CTRL-02	Stakeholder·context·system boundary	Context and stakeholder map
CTRL-03	Architecture driver·quality scenario	Prioritized quality scenarios
CTRL-04	Constraint·assumption·TBR	Constraint and TBR register
CTRL-05	Comparable option portfolio	Architecture option cards
CTRL-06	Module·component·interface boundary	Context and container views
CTRL-07	Data ownership·consistency·transaction	Data ownership map
CTRL-08	Integration·async·failure handling	Failure path and sequence map
CTRL-09	Security·privacy·trust boundary	Trust and threat map
CTRL-10	Reliability·performance·operability·cost	Quality and operations scorecard
CTRL-11	Trade-off·ADR·reversibility	ADR-001 and trade-off matrix
CTRL-12	Fitness evidence·evolution·M13-02 handoff	Architecture Decision Gate

20. 24개 합성 시나리오 포트폴리오

ID	Lane	시나리오	위험	Source → Evidence	Controls
SC-01	context-drivers	M12-04 linked package를 source version으로 인수	false-architecture-approval	M12-04 evidence pack → architecture input manifest	CTRL-01, CTRL-02
SC-02	context-drivers	결정 범위와 승인 경계 분리	false-architecture-approval	architecture question → decision boundary	CTRL-01, CTRL-02, CTRL-12

ID	Lane	시나리오	위험	Source → Evidence	Controls
SC-03	context-drivers	이해관계자와 관점 지도	trend-driven-choice	stakeholder concerns → viewpoint map	CTRL-02, CTRL-03
SC-04	context-drivers	시스템·외부 의존·trust boundary	failure-blindness	product boundary → context view	CTRL-02, CTRL-06, CTRL-09
SC-05	context-drivers	비즈니스 목표에서 architecture driver 우선순위	trend-driven-choice	product goal and constraints → driver register	CTRL-01, CTRL-03
SC-06	context-drivers	측정 가능한 8개 품질 시나리오	trend-driven-choice	quality concerns → quality scenario portfolio	CTRL-03, CTRL-10
SC-07	quality-risk	Constraint·assumption·TBR 구분	false-architecture-approval	unknowns and limits → assumption register	CTRL-04
SC-08	quality-risk	보안·개인정보·조직 격리 위험	shared-data-coupling	trust and data flow → threat and mitigation map	CTRL-09, CTRL-07
SC-09	quality-risk	장애·복구·RTO·RPO 시나리오	failure-blindness	failure modes → recovery evidence plan	CTRL-08, CTRL-10
SC-10	quality-risk	부하·latency·capacity profile	trend-driven-choice	scale assumptions → performance evidence plan	CTRL-03, CTRL-10
SC-11	quality-risk	팀 운영 역량과 비용 신호	operability-overload	team and usage profile → operations cost scorecard	CTRL-04, CTRL-10
SC-12	quality-risk	위험·mitigation·evidence·residual risk	failure-blindness	risk register → evidence backlog	CTRL-08, CTRL-09, CTRL-10, CTRL-12
SC-13	structure-data-integration	같은 기준으로 4개 option 비교	trend-driven-choice	architecture drivers → option portfolio	CTRL-05, CTRL-03
SC-14	structure-data-integration	6개 module과 8개 component 책임	shared-data-coupling	domain responsibilities → container and module view	CTRL-06
SC-15	structure-data-integration	Interface·protocol·version·timeout·idempotency	distributed-complexity	component relationships → interface contract	CTRL-06, CTRL-08

ID	Lane	시나리오	위험	Source → Evidence	Controls
SC-16	structure-data-integration	Data write ownership·transaction·consistency	shared-data-coupling	five data domains → data ownership map	CTRL-07
SC-17	structure-data-integration	외부 연동·outbox·중복·복구	failure-blindness	meeting provider and committed action → worker and outbox flow	CTRL-08, CTRL-07
SC-18	structure-data-integration	배포·failure isolation·관측 연결	operability-overload	two deployables → failure and telemetry map	CTRL-06, CTRL-08, CTRL-10
SC-19	decision-evolution-governance	가중치·증거·trade-off matrix	trend-driven-choice	four options → trade-off matrix	CTRL-05, CTRL-10, CTRL-11
SC-20	decision-evolution-governance	Build-vs-buy·managed dependency·exit path	operability-overload	capability choices → dependency register	CTRL-05, CTRL-11
SC-21	decision-evolution-governance	ADR에 context·option·decision·consequence 기록	false-architecture-approval	trade-off evidence → ADR-001	CTRL-11
SC-22	decision-evolution-governance	Fitness evidence portfolio와 검증 backlog	failure-blindness	quality scenarios and risks → fitness evidence portfolio	CTRL-03, CTRL-09, CTRL-10, CTRL-12
SC-23	decision-evolution-governance	진화 trigger·reversibility·review cadence	distributed-complexity	measured change signals → evolution plan	CTRL-11, CTRL-12
SC-24	decision-evolution-governance	Architecture Decision Gate·M13-02 handoff	false-architecture-approval	architecture evidence pack → decision gate and vendor handoff	CTRL-01, CTRL-09, CTRL-10, CTRL-11, CTRL-12

네 lane은 각각 6개 scenario를 가집니다: `context-drivers`, `quality-risk`, `structure-data-integration`, `decision-evolution-governance`.

21. 8개 Quality Scenario 한눈표

ID	속성	Source·stimulus	Environment·artifact	Response measure
QA-01	performance	authorized coordinator · creates or confirms an action	20 req/s for 15 minutes · Application API and DB	p95 <= 800 ms; error rate < 1% in synthetic test
QA-02	reliability	runtime · API process terminates	normal business hours · Application API	service recovery <= 10 minutes in drill
QA-03	recoverability	operator · logical data corruption is declared	restore exercise · Managed Relational DB	RTO <= 4 h; RPO <= 1 h in synthetic drill
QA-04	security	user from another organization · requests an action record	authenticated session · API authorization boundary	0 cross-org records exposed in test portfolio
QA-05	integration resilience	meeting provider · times out for 30 minutes	source import · Meeting Adapter and Worker	0 duplicate records; recovery after provider resumes
QA-06	operability	product engineer · one of six known failure modes occurs	support window · Telemetry and runbook	diagnosis <= 30 minutes in tabletop drill
QA-07	cost	usage profile · synthetic monthly load is applied	20 organizations and 200 WAU · runtime, DB, identity, telemetry	signal <= USD 600/month; alert at 80%; not budget approval
QA-08	modifiability	product team · adds one optional outcome field	backward-compatible change · Outcome module, API, schema, UI	impact map complete; no cross-module direct data write

22. 4개 Option과 선택 상태

ID	후보	형태	이익	부담	상태·이유
OPT-A	단일 VM 계층형 모놀리스	한 배포 단위·자체 운영 DB	이해와 시작이 단순함	백업·패치·복구 운영 책임이 큼	rejected-for-case · 작은 팀에 자체 운영 부담이 과도함
OPT-B	관리형 모듈러 모놀리스 +워커	API와 worker 두 배포 단위·관리형 DB/identity	트랜잭션 단순성·명확한 모듈 경계·낮은 운영 부담	한 runtime의 배포 결합과 장기 확장 경계	selected-candidate · 현재 scale·팀·품질 시나리오에 가장 균형적

ID	후보	형태	이익	부담	상태·이유
OP T-C	거친 단위 서비스 분리	3~4 service·서비스별 write ownership	독립 배포·격리·선택적 확장	분산 transaction·관측·배포·계약 운영	deferred · 독립 확장과 팀 ownership trigger가 아직 없음
OP T-D	함수·이벤트 중심 서버리스	요청 함수·event·managed queue/store	낮은 유휴 비용·burst 대응	흐름 추적·transaction·cold start·provider coupling	deferred · 핵심 편집 흐름보다 비동기 일부에 선택 적용 가능

23. 6개 Module·8개 Component

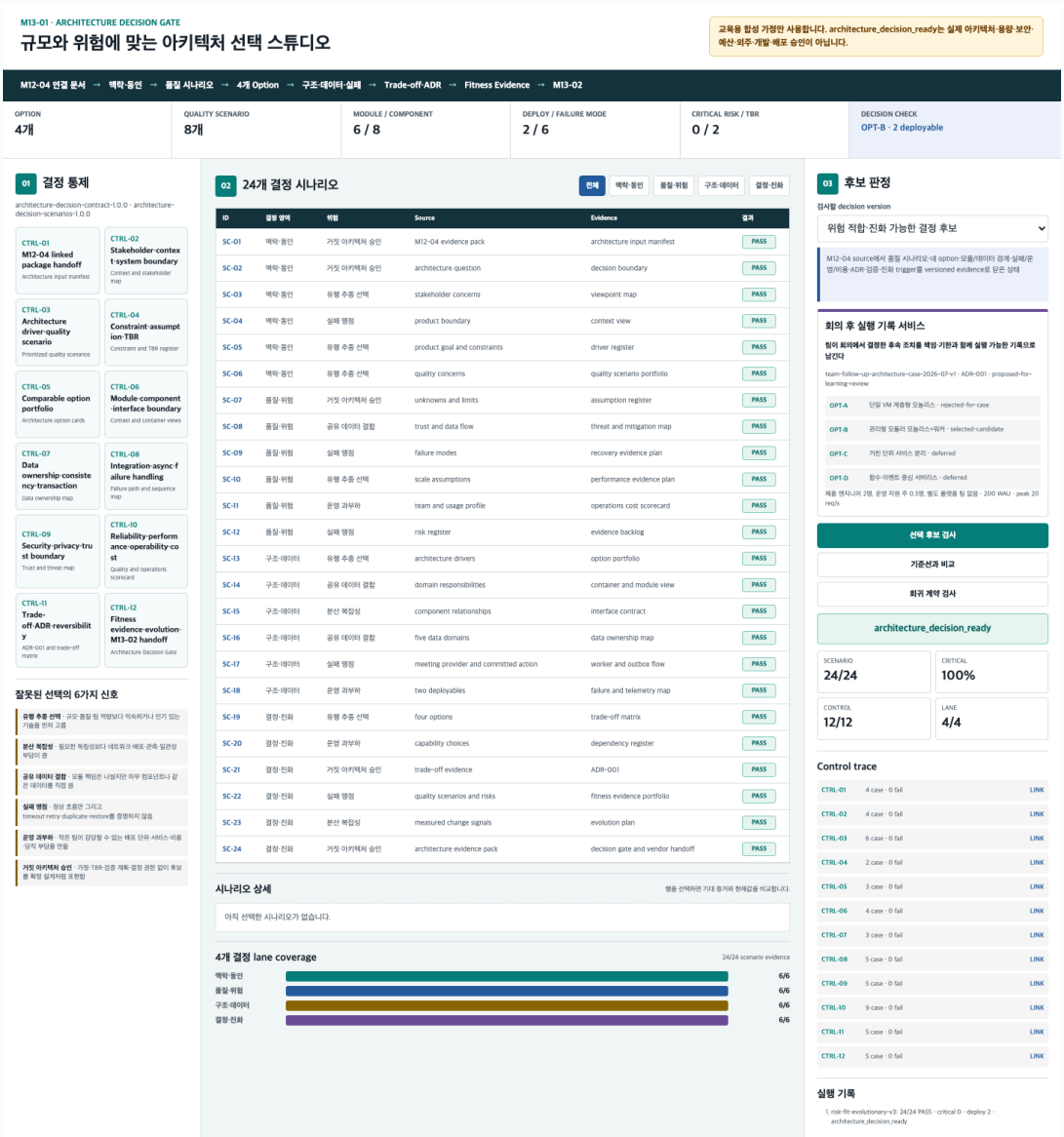
Module	책임	Write owner
MOD-01 Meeting Source	source reference·import cursor	meeting_source
MOD-02 Action Record	action text·status·version	action_record·record_version
MOD-03 Assignment	assignee proposal·confirmation	assignment_confirmation
MOD-04 Outcome Evidence	completion outcome·evidence reference	outcome_evidence
MOD-05 Organization Access	organization context·role adapter	no local credentials
MOD-06 Audit and Change	audit event·outbox·change evidence	audit_event·outbox_event

Component	Type	Responsibility
CMP-01 Browser UI	client	screen state·accessible recovery
CMP-02 Application API	managed runtime	six modules·authorization·transactions
CMP-03 Background Worker	managed runtime	outbox·retry·integration jobs
CMP-04 Managed Identity	managed dependency	authentication·organization claims
CMP-05 Managed Relational DB	managed data	module-owned tables·backup·restore input
CMP-06 Meeting Adapter	external boundary	source metadata import·timeout isolation
CMP-07 Telemetry	managed operations	metrics·logs·traces·alerts

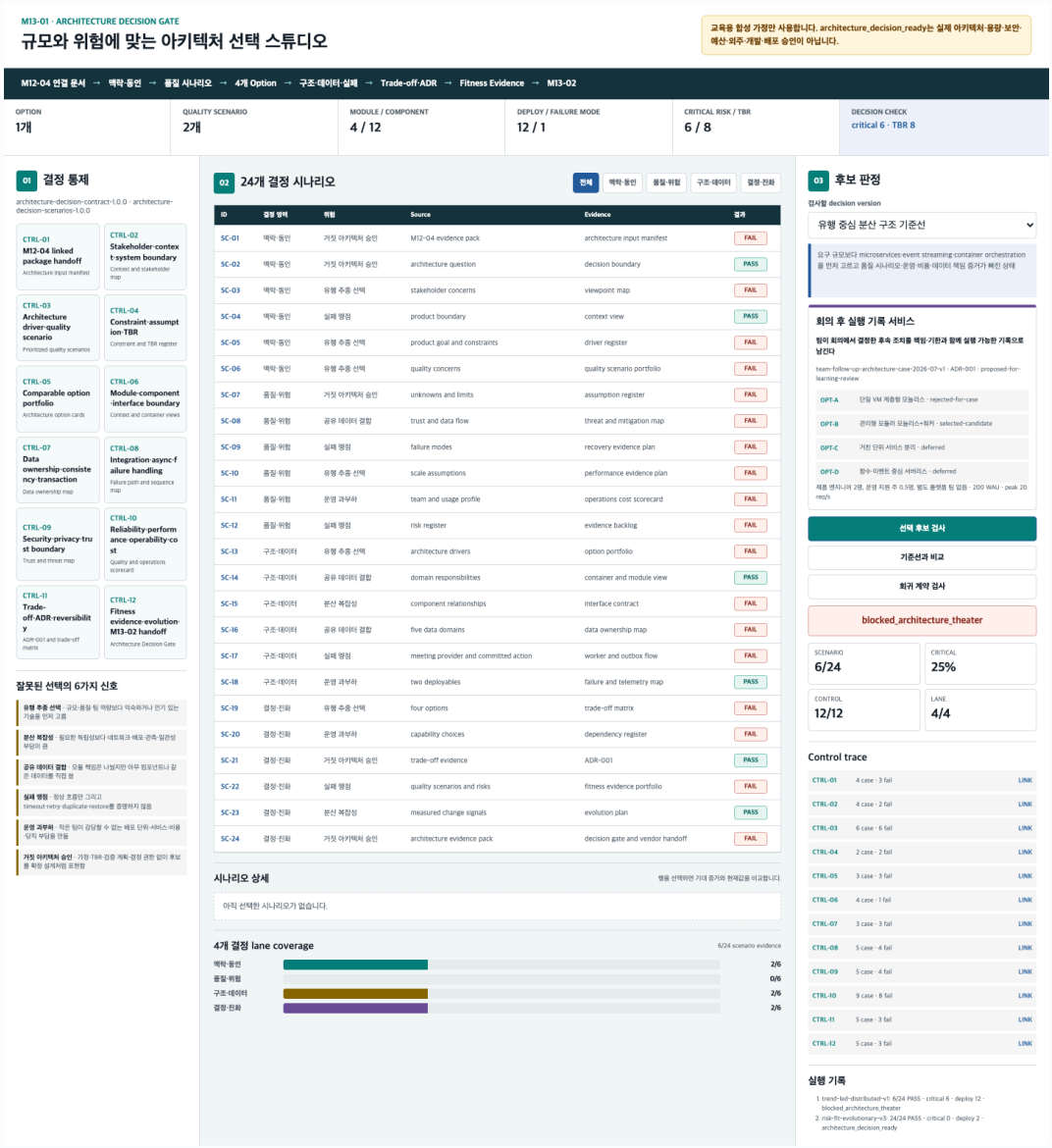
Component	Type	Responsibility
CMP-08 Backup and Restore	managed operations	encrypted backup·restore evidence

24. 실습 스튜디오: 세 decision version 실행

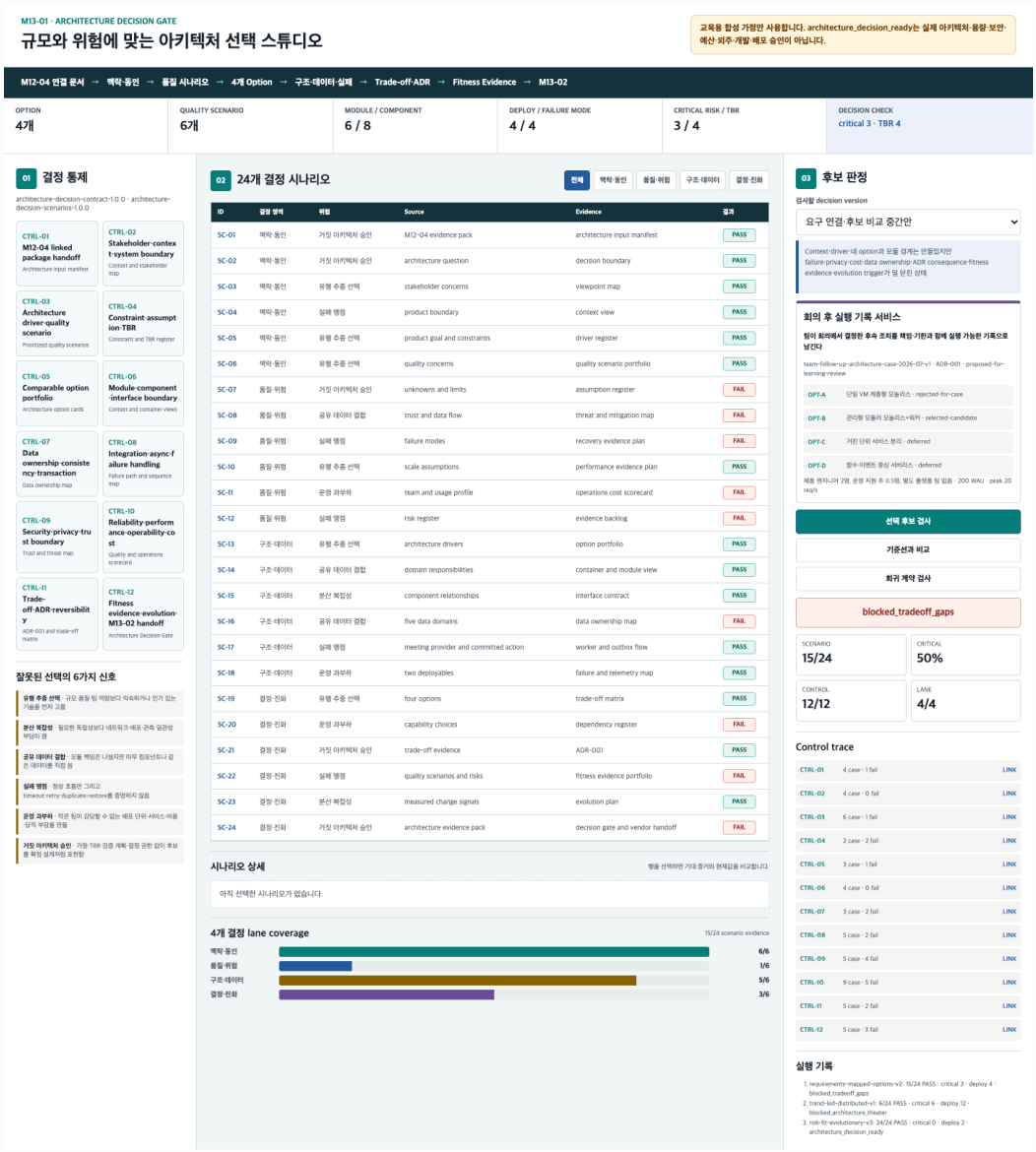
실습은 loopback-only·standard-library-only이며 외부 network나 production resource를 사용하지 않습니다. 각 variant는 같은 24 scenario를 실행해 expected와 actual의 mismatch를 계산합니다.



실습 화면 1. Candidate는 24/24, critical 100%, 12 controls, 네 lane을 통과하고 OPT-B·2 deployable을 표시합니다.

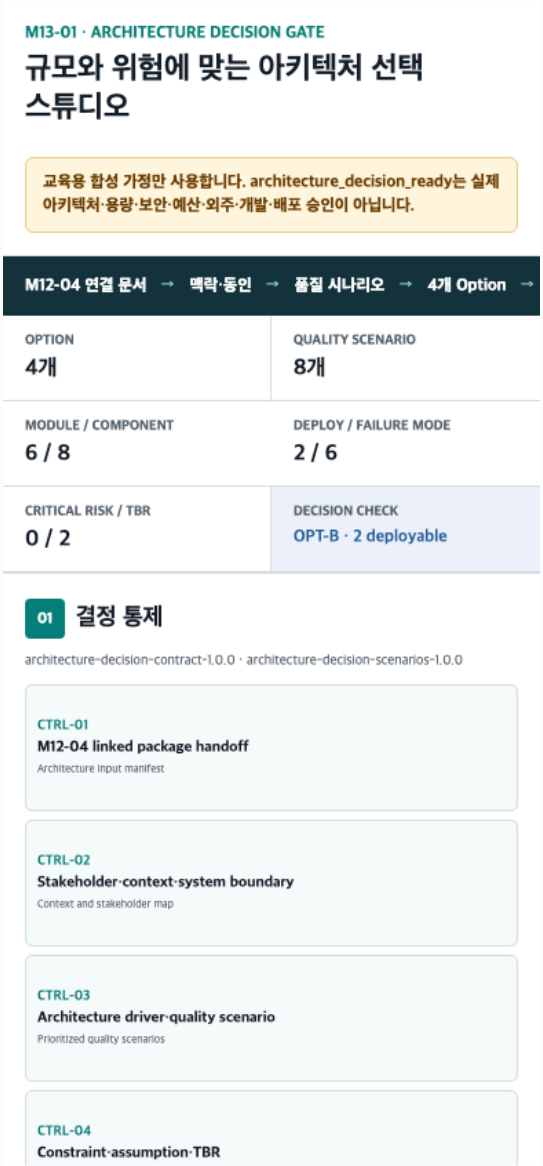


실습 화면 2. Baseline은 6/24, critical risk 6, 12 deployable로 architecture theater를 드러냅니다.



실습 화면 3. 중간안은 option과 module은 만들었지만 failure·privacy·cost·evolution evidence가 열려 있습니다.

실습 화면 4. Scenario row를 선택하면 boundary·source→evidence·expected oracle·actual을 나란히 비교합니다.



실습 화면 5. 모바일에서도 approval boundary와 option·quality·risk summary가 먼저 보입니다.

실습 화면 6. 좁은 화면에서도 네 lane filter와 scenario 결과를 가로 스크롤로 검토합니다.

Version	Pass	Fail	주요 상태	판정
trend-led-distributed-v1	6	18	1 option·2 QA·12 deploy·critical6	blocked_architecture_theater
requirements-mapped-options-v2	15	9	4 option·6 QA·4 deploy·critical3	blocked_tradeoff_gaps
risk-fit-evolutionary-v3	24	0	4 option·8 QA·2 deploy·critical0	architecture_decision_ready

25. 90분 실습 워크북

시간	행동	남길 evidence	통과 질문
0~10분	M12-04 source·boundary	input manifest	무엇이 authoritative한가
10~20분	Stakeholder·context	context/viewpoint map	누구의 concern을 답하나
20~35분	8 quality scenarios	six-part portfolio	측정 가능한가
35~50분	4 options	option cards	같은 경계로 비교했나
50~65분	module·data·failure·trust	views+failure path	ownership과 recovery가 보이나
65~75분	trade-off·sensitivity	matrix+evidence	점수가 자동 결정이 아닌가
75~85분	ADR·fitness·trigger	ADR-001+backlog	왜·무엇을 감수했나
85~90분	Gate·M13-02 handoff	decision+residual risk	후보와 승인을 구분하나

```
Architecture question + answered/unanswered boundary:
M12-04 source version + linked IDs:
Stakeholder concern → viewpoint → evidence:
Goal → quality scenario → architecture driver:
Option A/B/C/D + selected/deferred/rejected reason:
Module → interface → data write owner:
Failure mode → mitigation → test/drill:
Trade-off + sensitivity + residual risk:
ADR status + consequence + evolution trigger:
Gate result + TBR + M13-02 handoff:
```

26. Architecture Decision Package 템플릿

별도 템플릿 `03_Templates/T13-01_architecture-decision-package.md`를 사용합니다. 최소 15개 section은 input·context·drivers·quality scenario·option·views·data·failure·security/privacy·operations/cost·matrix·ADR·fitness·Gate·handoff입니다.

27. 셀프 테스트 30

1. Architecture와 architecture description의 차이는 무엇인가요?

정답: Architecture는 system의 환경 안에서 요소·관계와 진화 원리를 보는 개념이고, description은 stakeholder가 검토할 수 있도록 view·model·rationale로 표현한 산출물입니다.

2. Viewpoint와 view를 구분해 보세요.

정답: Viewpoint는 어떤 concern을 어떤 독자·model kind·표현 규칙으로 다룰지 정한 관례이고, view는 그 규칙을 특정 system에 적용한 실제 표현입니다.

3. Architecture driver는 어떤 정보에서 나오나요?

정답: Business·product goal, 우선 quality scenario, hard constraint, external dependency, team·scale·data·risk에서 나옵니다. 기술 이름 자체는 driver가 아닙니다.

4. Decision boundary를 먼저 쓰는 이유는 무엇인가요?

정답: 현재 evidence가 답하는 질문과 capacity·security·budget·vendor·build·release처럼 별도 검증·권한이 필요한 질문을 구분해 후보를 승인본으로 오해하지 않게 하기 위해서입니다.

5. Quality scenario의 여섯 부분을 순서 없이 모두 쓰세요.

정답: Source, stimulus, environment, artifact, response, response measure입니다.

6. '빠르고 안정적이어야 한다'가 architecture 입력으로 부족한 이유는 무엇인가요?

정답: 누가 어떤 상황에서 무엇을 자극하고 system이 어떻게 반응하며 어떤 수치로 합격하는지 없어 option 비교와 시험이 불가능하기 때문입니다.

7. Constraint·assumption·TBR의 차이는 무엇인가요?

정답: Constraint는 현재 범위에서 지켜야 할 조건, assumption은 검증 가능한 믿음, TBR은 owner·due·authority를 가진 미결정 사항입니다.

8. 합성 scale profile의 핵심 수치를 쓰세요.

정답: 20개 조직, 주간 활성 200명, peak 20 req/s, 분당 background job 30개, 연간 record 250,000개입니다. 실제 forecast나 capacity 보장이 아닙니다.

9. 합성 사례의 네 architecture option은 무엇인가요?

정답: 단일 VM 계층형 모놀리스, 관리형 모듈러 모놀리스+worker, 거친 단위 서비스, 함수·event 중심 serverless입니다.

10. OPT-B가 선택 후보가 된 현재 이유는 무엇인가요?

정답: 작은 팀·낮은 초기 scale에서 local transaction과 명확한 module/data ownership을 유지하면서 managed DB·identity·runtime으로 운영 부담을 줄이고, worker로 외부 연동 실패를 분리하기 때문입니다.

11. Microservices가 항상 확장성 있는 정답이 아닌 이유는 무엇인가요?

정답: Network failure·latency·contract version·distributed transaction·observability·deployment·ownership·cost가 추가되며, 독립 scale·isolation·team ownership trigger가 없으면 부담이 이익보다 클 수 있습니다.

12. Modular monolith의 module boundary를 무엇으로 증명하나요?

정답: Responsibility·public interface·allowed dependency·single write owner·transaction boundary와 architecture test로 증명합니다. 폴더 이름만으로는 부족합니다.

13. Single write owner가 필요한 이유는 무엇인가요?

정답: 여러 module이 같은 data를 직접 쓰면 invariant·history·migration·authorization의 authoritative rule이 갈리고 변경 결합이 커지기 때문입니다.

14. Strong consistency와 eventual consistency의 선택 기준은 무엇인가요?

정답: 같은 transaction에서 반드시 지켜야 하는 invariant와 사용자에게 즉시 보여야 하는 state는 strong 쪽을 검토하고, 지연을 허용하는 projection·notification·integration은 eventual을 검토합니다.

15. Outbox pattern이 줄이는 위험과 남기는 위험을 쓰세요.

정답: Business data와 발행 record의 dual-write 불일치를 줄입니다. Duplicate·backlog·poison message·replay·ordering·operator recovery는 여전히 남습니다.

16. At-least-once delivery에서 idempotency가 필요한 이유는 무엇인가요?

정답: 같은 message가 반복 전달될 수 있으므로 logical operation ID나 unique constraint로 중복 side effect를 막아야 하기 때문입니다.

17. Fault·error state·failure를 구분해 보세요.

정답: Fault는 원인이 될 수 있는 결함, error state는 내부 상태가 잘못된 단계, failure는 외부에 약속한 service를 제공하지 못한 사건입니다.

18. RTO와 RPO의 차이는 무엇인가요?

정답: RTO는 service 복구 목표 시간이고 RPO는 복구 시 허용 가능한 data 손실 시간 범위입니다.

19. Trust boundary에서 확인할 최소 항목은 무엇인가요?

정답: Identity, authentication, authorization, organization isolation, least privilege, input validation, data minimization, retention, audit, encryption, threat와 verification입니다.

20. Privacy를 data encryption만으로 설명할 수 없는 이유는 무엇인가요?

정답: 수집 목적·최소화·권한·공유·보존·파기·audit·사용자 권리와 data flow 전체를 다뤄야 하기 때문입니다.

21. 합성 performance scenario QA-01의 measure는 무엇인가요?

정답: 20 req/s를 15분 적용할 때 create/confirm의 p95가 800ms 이하이고 합성 시험 오류율이 1% 미만인 것입니다.

22. 작은 팀의 operability evidence 예를 쓰세요.

정답: 두 배포 단위, 알려진 여섯 failure mode, metric·log·trace, runbook, 30분 이내 진단 tabletop, backup·restore drill과 owner입니다.

23. USD 600/month 수치가 budget approval가 아닌 이유는 무엇인가요?

정답: 합성 workload에서 option을 비교하는 cost signal일 뿐 실제 provider 견적·세금·지원·인력·계약·forecast·재무 권한을 포함하지 않기 때문입니다.

24. Weighted score가 decision을 자동화하지 못하는 이유는 무엇인가요?

정답: Weight·score·evidence confidence가 가정에 민감하고 hard constraint·residual risk·authority·unmeasured consequence를 숫자 하나가 대신할 수 없기 때문입니다.

25. ADR의 핵심 구성 요소를 쓰세요.

정답: Title, status, context, considered options, decision, positive·negative·neutral consequences, evidence, residual risk, review·supersede trigger입니다.

26. ADR consequence에 단점도 기록해야 하는 이유는 무엇인가요?

정답: 선택이 만든 새 제약·운영 비용·lock-in·미래 migration 부담을 future team이 이해하고 context 변화 때 decision을 다시 볼 수 있게 하기 위해서입니다.

27. Fitness evidence portfolio의 다섯 묶음은 무엇인가요?

정답: Contract/architecture test, load test, failure·restore drill, security/privacy test, cost·operations review입니다.

28. 합성 사례의 evolution trigger 예를 세 개 쓰세요.

정답: Peak가 100 req/s를 지속 초과, engineer가 8명 이상으로 늘어 독립 ownership이 생김, 배포 contention이 매주 발생, 규제상 격리 필요, provider cost가 2배를 넘는 경우 중 세 개입니다.

29. 세 decision version의 pass 수와 판정을 쓰세요.

정답: trend-led-distributed-v1은 6/24·blocked_architecture_theater, requirements-mapped-options-v2는 15/24·blocked_tradeoff_gaps, risk-fit-evolutionary-v3는 24/24·architecture_decision_ready입니다.

30. Architecture Decision Gate가 뜻하지 않는 것과 M13-02에 넘길 것을 쓰세요.

정답: 실제 architecture·capacity·availability·security/privacy certification·budget·vendor·contract·build·release 승인을 뜻하지 않습니다. M13-02에는 context, quality scenarios, option matrix, views, data/interface boundary, ADR, evidence backlog, residual risk, acceptance·inspection criteria를 넘깁니다.

28. 공식 자료와 적용 경계

자료	이 장에서 가져온 원리	적용 경계
ISO/IEC/IEEE 42010:2022	architecture description·stakeholder·viewpoint·model	표준 전문을 대체하지 않음
ISO/IEC/IEEE 42020:2019	architecture governance·management·architecting process	조직 tailoring 필요
ISO/IEC 25010:2023	ICT/software product quality model	품질 우선순위는 context에서 결정
SEI QAW	architecture 전 quality scenario 발견·정교화	공식 workshop 전체를 축약한 학습 적용
SEI ATAM	risk·sensitivity·tradeoff·risk theme	정식 ATAM 평가 인증 아님
C4 model · review checklist	context/container hierarchy·diagram clarity	필요한 view만 사용
Michael Nygard ADR	small modular decision record·status·consequence	팀의 status·template tailoring 필요

자료	이 장에서 가져온 원리	적용 경계
NIST SP 800-160 Vol.1 Rev.1	trustworthy secure system engineering	security certification 아님
OWASP Threat Modeling	asset·assumption·threat·mitigation·validation	실제 threat assessment 별도
AWS Well-Architected	operations·security·reliability·performance·cost·sustainability	AWS product 선택 강제 아님
Azure Well-Architected	quality pillars·tradeoff·iterative review	Azure workload용 guidance임
Google Cloud Well-Architected	design for change·document architecture·six pillars	Google Cloud context를 일반화할 때 주의
FinOps Framework · Architecting & Workload Placement	cost awareness·business value·placement tradeoff	실제 budget·forecast는 finance authority 필요

29. 용어집 300 학습 순서

`04_Glossary/GLOSSARY_architecture_decision.md`에는 15개 묶음·300개 unique term이 있습니다.

묶음	핵심 질문
01~03	Architecture·context·quality를 구분하는가
04~06	Scenario·style·module/interface를 설명하는가
07~10	분산·data·failure·trust boundary를 설명하는가
11~13	Performance·operations·cost를 측정하는가
14~15	Trade-off·ADR·fitness·trigger·handoff를 연결하는가

30. 다음 단계 · M13-02 handoff

M13-02에서는 이 decision package를 이용해 외주 개발사가 같은 context·quality scenario·boundary·evidence를 이해하고 제안하도록 요청하며, 결과를 화면 완성도만이 아니라 contract·data·failure·security·operations·test evidence로 검수합니다.

```
M12-04 linked product documents
→ M13-01 context + quality scenarios + options
→ selected architecture candidate + ADR
→ module + interface + data + failure + trust boundaries
→ fitness evidence backlog + residual risk + evolution trigger
→ M13-02 request for proposal + response matrix + inspection plan
```

Handoff item	M13-01 source	M13-02 use
Context·scope	input manifest·context view	제안 전제·제외
Quality scenarios	QA-01~08	수용·성능·복구 질문
Option decision	matrix·ADR-001	대안·변경 제안 근거
Structure·data	container·module·ownership	구현 경계·migration
Failure·security	failure/trust views	위험·test evidence
Operations·cost	deploy·telemetry·TCO signal	운영·견적 분해
Evidence·trigger	backlog·Gate·TBR	검수·인계·변경 통제

마지막 확인: architecture decision package는 답을 고정하는 문서가 아니라 현재 맥락에서 가장 적합한 후보와 그 약점·검증·변경 조건을 함께 보존하는 문서입니다.