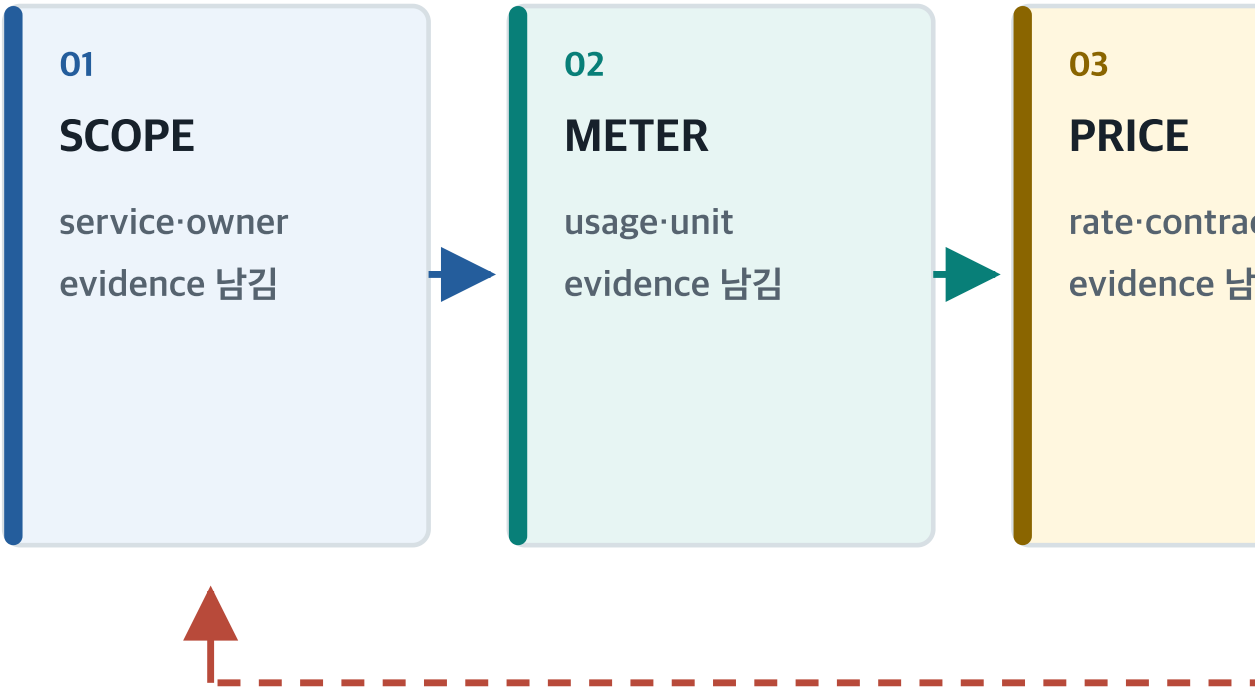
M11-04 · GIBALJA VISUAL MANUAL

클라우드·AI 비용과 라이선스 계산하기

한 문장 목표: 가격표 숫자를 외우지 않고 `scope → usage·meter → rate·contract → allocation → unit economics·TCO → Cost Gate` 를 24개 합성 시나리오로 연결해, 무엇을 알고 무엇이 아직 가정인지 설명할 수 있는 비용 모델을 만듭니다.

그림 1. 비용은 가격표를 한 번 읽는 작업이 아닙니다. Scope와 사용량을 정규화하고 rate·contract를 연결해 배부한 뒤, unit cost와 TCO가 다음 제품·architecture 결정을 바꾸고 actual·invoice가 다시 가정을 갱신합니다.

Scope·meter·rate·allocation을 연결하고 unit cost와 TCO가 다음 설계 결



[illegible]

[illegible]정을 바꿉니다.


ct
김
04
ALLOCATE
product·team
evidence 남김
05
VALUE
unit cost·TCO
evidence 남김

0. 한눈에 보는 두 번째 지도

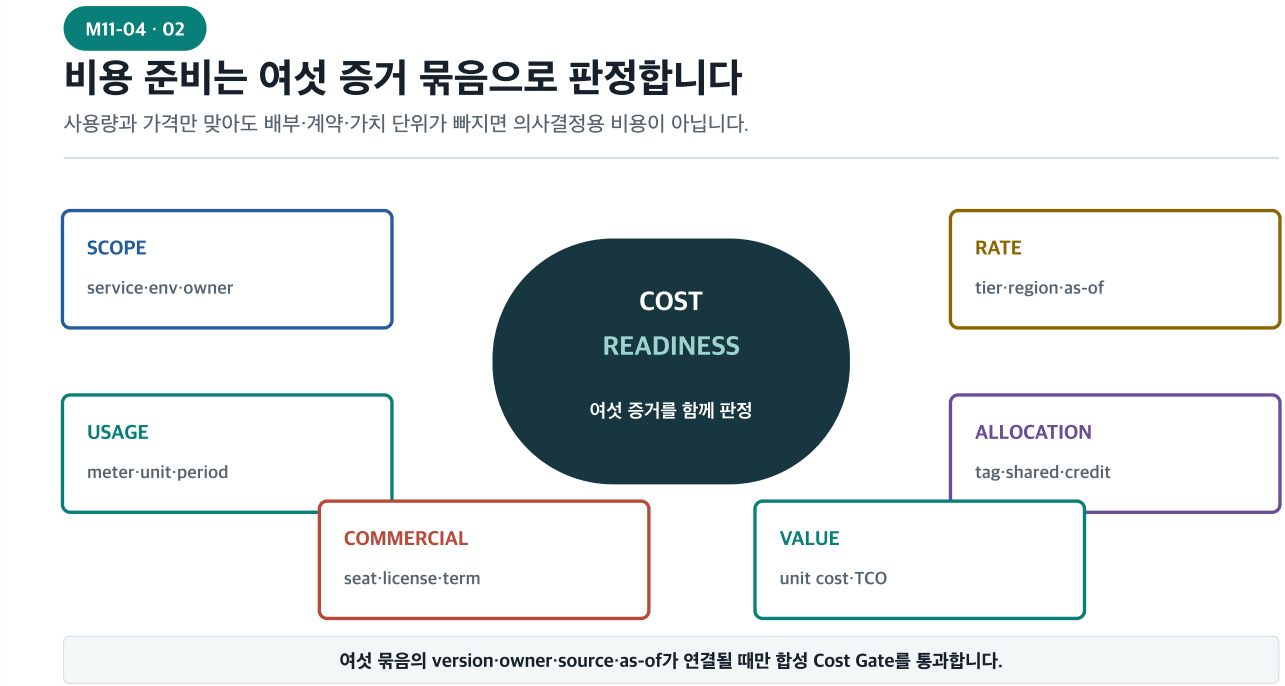


그림 2. Usage × rate가 맞아도 allocation·commercial term·business value가 빠지면 의사결정용 비용이 아닙니다. 여섯 증거의 version·source·as-of·owner가 연결돼야 합니다.

사용량과 가격만 맞아도 배부·계약·가치 단위가 빠지면 의사결정용 비용이

SCOPE

service·env·owner

USAGE

meter·unit·period

COMMERCIAL

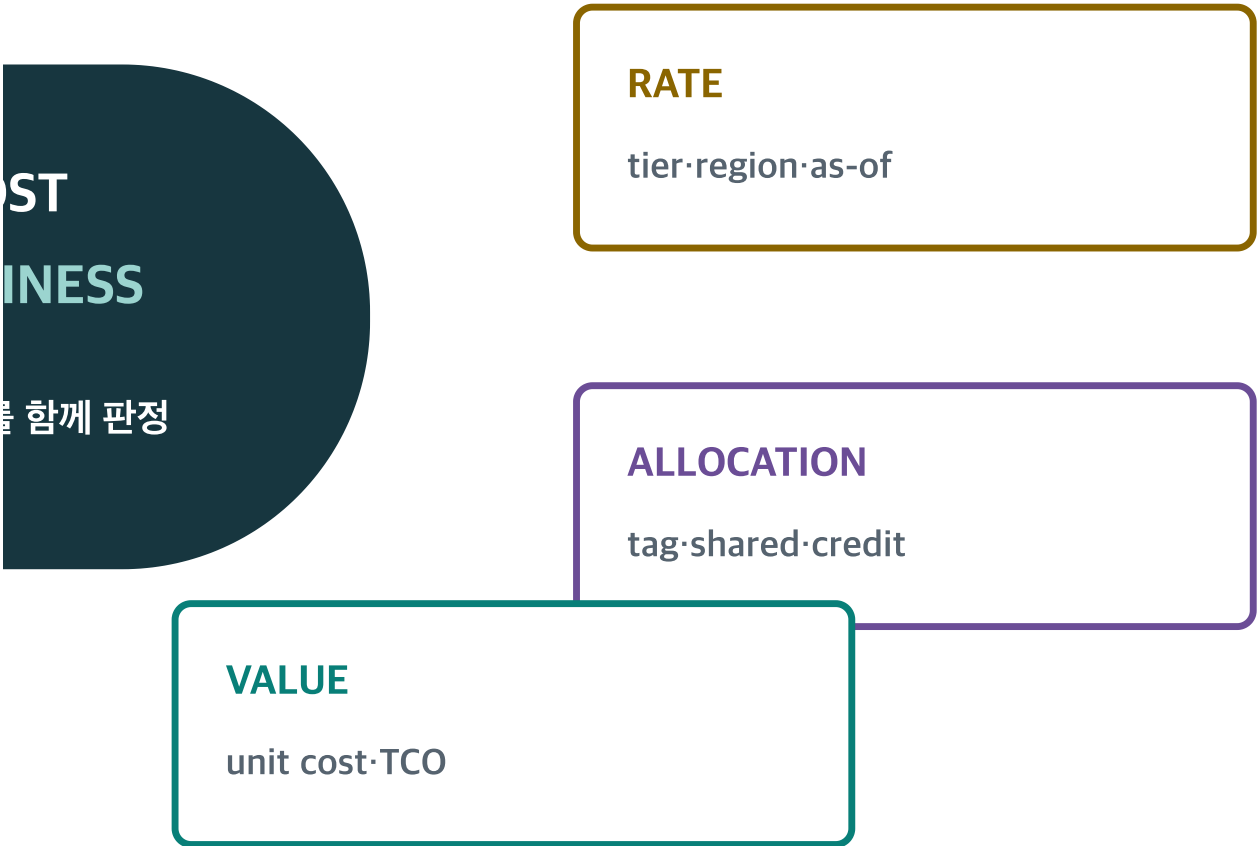
seat·license·term

CO

READ

여섯 증거를

아닙니다.


ST
INESS
를 함께 판정
RATE
tier·region·as-of
ALLOCATION
tag·shared·credit
VALUE
unit cost·TCO

난이도	그림 먼저	개념·계산	실습	셀프 테스트	최종 산출물
Level 2	40분	85분	95분	20분	월 비용·unit cost·12개월 TCO·Cost Gate

이 매뉴얼은 특정 provider의 영구 가격표가 아닙니다. Cloud·AI·SaaS 가격, model, region, tier, free allowance, discount, contract, FX, tax, license 조건은 바뀝니다. 본문의 모든 계산값은 재현 학습을 위한 허구의 고정값이며 실제 quote·invoice·tax·법률 의견·license clearance·구매·예산 승인·비용 보장을 대신하지 않습니다.

0.1 첫 두 장에서 기억할 열두 문장

```
Cost는 price 한 칸이 아니라 scope와 기간의 결과다.
Usage와 rate는 같은 unit이어야 한다.
숫자에는 source·as-of·owner가 필요하다.
List cost와 billed cost는 다를 수 있다.
Network는 GB보다 방향과 경계가 먼저다.
AI는 input·cached input·output·tool·retry를 나눠 센다.
가장 싼 model보다 cost per successful outcome을 본다.
Telemetry·backup·restore는 운영 신뢰성의 비용이다.
License는 seat 가격과 사용권·의무를 함께 본다.
Estimate·forecast·budget·actual·invoice는 다른 숫자다.
한 점 견적보다 low·base·high와 break-even을 본다.
TCO와 Cost Gate는 실제 권한자의 승인을 대신하지 않는다.
```

1. 이 PDF를 공부하는 방법

1.1 1회차 · 그림 16장만 읽기 · 40분

그림 제목과 caption만 읽고 다음 흐름을 소리 내어 설명합니다.

```
Cost-to-value 순환
→ 여섯 evidence 묶음
→ usage × rate + fixed + risk
→ cloud cost driver·pricing model
→ AI token·tool·quality-cost frontier
→ observability·backup·restore
→ license·entitlement·obligation
→ estimate·forecast·budget·actual·invoice
→ unit economics·TCO·sensitivity
→ 24 scenario·Cost Gate
```

각 그림에서 다음 문장을 완성합니다.

> 이 비용의 usage source는 _____이고, meter·unit은 _____이며, rate의 as-of는 _____이고, 누락 시 잘못 내릴 결정은 _____이다.

1.2 2회차 · 비용 계산 스튜디오 · 60분

실습 생성기를 실행합니다.

```
./02_Labs/G11_Deployment_Operations/L11-04_create-cost-license-practice.sh
```

세 model을 비교합니다.

Version	Pass	월 합계 USD	Decision
`list-price-only-v1`	6/24	2,809.10	`blocked_cost_blindness`
`usage-without-license-v2`	16/24	4,335.70	`blocked_commercial_readiness`
`verified-cost-model-v3`	24/24	6,141.23	`cost_ready`

1.3 3회차 · 내 서비스 TCO 표 · 120분

클라우드·AI 비용·라이선스·TCO 모델을 다음 순서로 작성합니다.

1. Scope·owner·period·currency·as-of
2. Rate card source·unit·region·tier·contract
3. Compute·storage·network usage
4. AI token·cache·embedding·tool·quality
5. Telemetry·backup·restore·support
6. Commercial·OSS license register
7. Shared cost·credit·discount·tax allocation
8. Estimate·forecast·budget·actual·invoice
9. Unit cost·TCO·sensitivity·break-even
10. 24 scenario·residual risk·Cost Gate

1.4 읽다가 막힐 때

비용·라이선스·TCO 용어집 300에서 지금 장과 같은 번호의 20개만 읽습니다. 300개를 먼저 외우지 않습니다.

---

2. “가격을 안다”를 다시 정의하기

2.1 여섯 문장은 서로 다르다

문장	실제로 아는 것	아직 모르는 것
`$0.08` 이다.	숫자 하나	무엇의 rate인지
`$0.08/vCPU-hour` 다.	Rate와 unit	Region·tier·as-of·contract
2,920 vCPU-hour다.	Usage	Provision·idle·source·period
월 compute가 `$233.60` 다.	Usage × rate	Shared·support·tax·TCO
Invoice가 `$233.60` 이다.	Billed charge 일부	Product value·allocation
Cost per success가 `$0.005196` 다.	Value unit과 fully allocated cost	Quality 정의·미래 변화

COST READINESS 비용 준비는

무엇을 어느 기간에 얼마나 썼고 어떤 rate·contract가 적용됐으며 누구의 가치 단위에 귀속되는지 재현하고, 불확실성과 의무를 드러낸 상태

입니다.

2.2 Price·rate·usage·charge·cost

용어	질문	예시
Price	이 상품 묶음의 표시 금액은	월 subscription `$250`
Rate	단위 하나 가격은	`$0.08/vCPU-hour`
Usage	기간에 몇 단위를 썼나	`2,920 vCPU-hour/month`

용어	질문	예시
Charge	어떤 사용·구매·조정 항목인가	Compute usage line
Cost	Scope 안 경제적 부담은	Usage·fixed·shared·risk 합계
Spend	기간 전체 발생·지급 합계는	Monthly technology spend

2.3 최소 비용 계약

```
scope + period + usage source + meter + unit
+ rate + region + tier + currency + as-of
+ contract + allocation + owner + assumption
= reproducible cost evidence
```

2.4 대표 실패 여섯 가지

실패 유형	겉으로 보이는 증상	실제 위험
범위·배부 공백	총액은 있음	Product·team·environment owner 불명
계량·요율 오류	계산식은 있음	Unit·tier·region·rounding 불일치
AI usage drift	Request는 비슷함	Output·tool·retry·model 변화
운영 비용 누락	Cloud는 쌈	Telemetry·backup·support가 사라짐
License 의무 공백	Seat 가격은 앎	Entitlement·overage·notice·source 모름
Forecast 거버넌스 실패	Budget은 있음	Actual·invoice 차이와 reforecast owner 없음

2.5 M11-03에서 받는 handoff

M11-04는 운영 설계를 다시 만들지 않습니다. M11-03에서 다음 사용량을 받습니다.

- Log ingestion GB/day
- Log index·archive GB-month·retention
- Active metric series·sample
- Trace spans/day·sampling·retention
- Dashboard·alert·on-call seats
- Backup protected GB·copy·tier·retention

- Restore read·compute·egress·request per drill
- Support·incident tool·extra region

비용 때문에 telemetry·backup·restore를 자동 삭제하지 않습니다. 먼저 어떤 reliability evidence가 필요한지 확인하고 sampling·aggregation·tiering·retention·architecture를 조정합니다.

3. 한 줄 계산식의 네 층

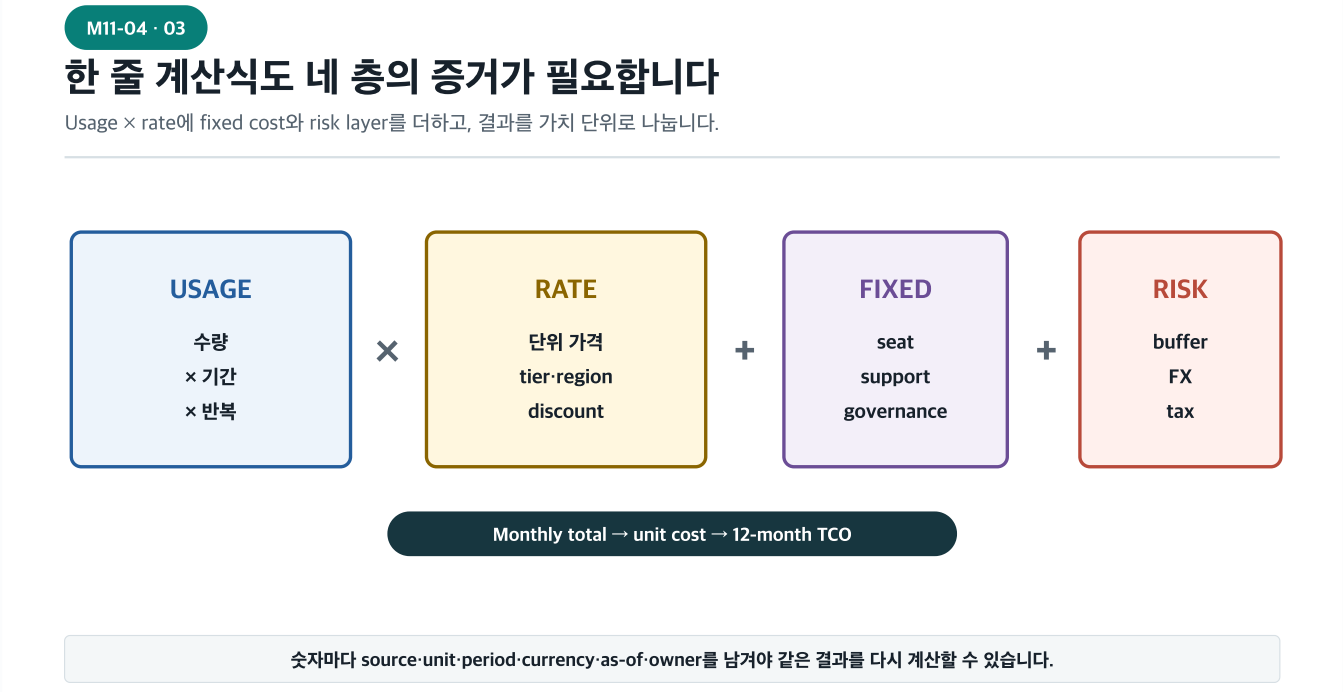
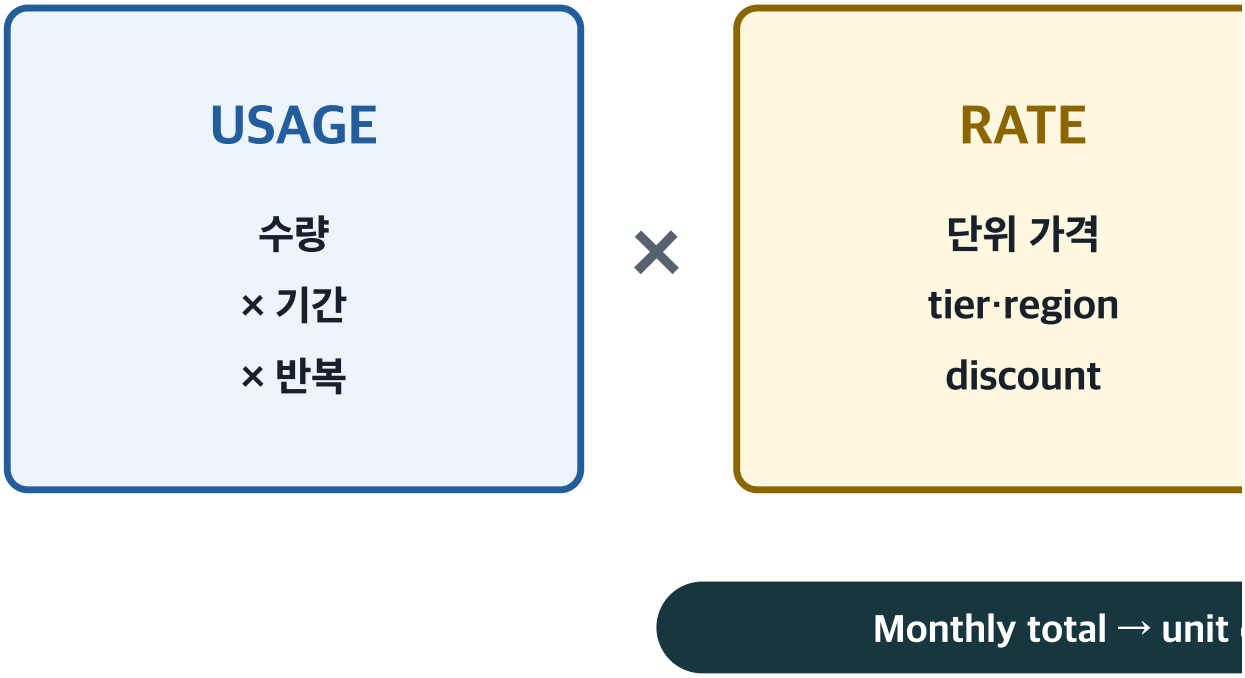


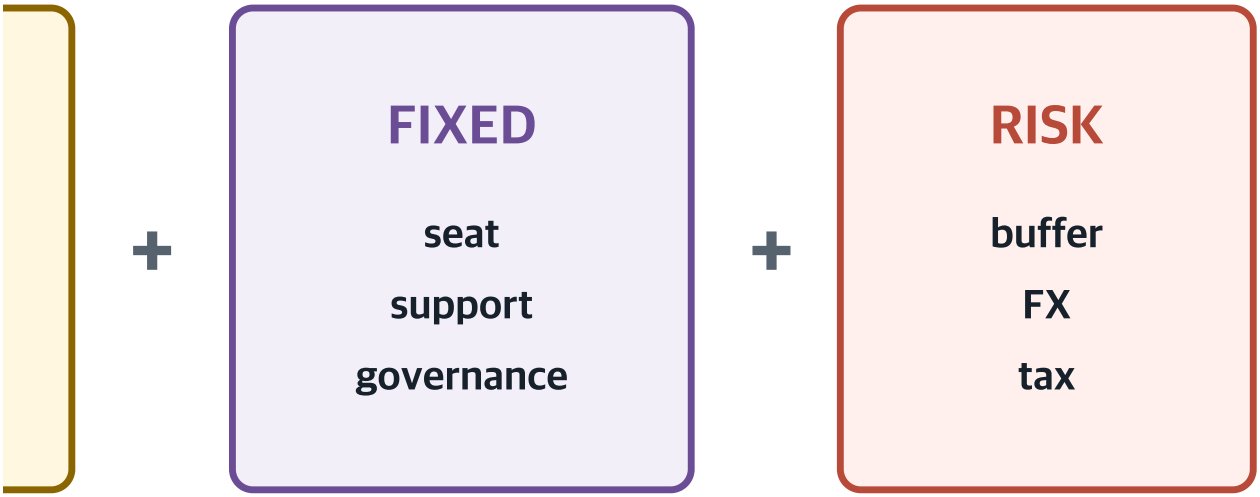
그림 3. Usage × rate는 시작입니다. Seat·support·governance 같은 fixed cost와 contingency·FX·tax 같은 risk layer를 더하고, business unit으로 나눠야 의사결정에 가까워집니다.

단일 레코드 사용량 계산 파이프라인

Usage × rate에 fixed cost와 risk layer를 더하고, 결과를 가치 단위로 나눕



니다.



cost → 12-month TCO

3.1 Usage를 만든다

Usage는 “대충 많음”이 아니라 기간과 반복을 가진 수량입니다.

```
compute = instance count × vCPU × runtime hours
memory = instance count × GB × runtime hours
storage = time-weighted average GB × month
AI input = requests × input token/request
tool calls = requests × calls/request × agent iterations
```

3.2 Rate를 붙인다

Rate에는 숫자 외에 조건이 있습니다.

필드	왜 필요한가
Provider·service·SKU	같은 이름 아래 규격이 다름
Unit	Usage와 같은 차원이어야 함
Region·zone	위치별 rate·transfer가 다름
Tier·band	구간별 marginal rate가 다름
Pricing model	On-demand·commitment·subscription 차이
Currency	Pricing과 billing 통화가 다를 수 있음
As-of	가격이 바뀔 수 있음
Source·snapshot	재검과 감사가 가능해야 함
Contract	List와 negotiated·effective가 다름

3.3 Fixed cost를 더한다

Usage가 0에 가까워도 남을 수 있습니다.

- Subscription minimum
- Purchased seats
- Vendor support·maintenance
- On-call·incident tool
- Platform base fee

- License minimum·true-up
- Compliance·legal·governance
- Dedicated connectivity

3.4 Risk layer를 분리한다

Risk layer를 숨겨 subtotal과 섞지 않습니다.

Layer	예시	원칙
Contingency	사용량·설계 불확실성 7%	조직 policy·risk 기반
FX	USD→KRW 1,400	Source·as-of 고정
Tax	예시 10%	세무 판단 별도
Variance tolerance	Rounding·late data	Invoice reconciliation 규칙

> 본문의 7%, 1,400, 10%는 허구의 학습 값입니다.

3.5 Value unit으로 나눈다

```
successful outcomes = eligible requests × success rate
cost per success = fully allocated monthly cost / successful outcomes
```

Resource unit과 business unit을 구분합니다.

Unit	예시	답하는 질문
Resource unit	Cost/token·GB·vCPU-hour	기술 효율이 좋아졌나
Business unit	Cost/user·transaction·success	가치 생산이 지속 가능한가

FinOps Unit Economics도 resource efficiency unit과 business unit을 구분하고, 성숙도가 높아지면 fully loaded cost를 business outcome에 연결하도록 설명합니다.

4. Cost Evidence Bundle 만들기

4.1 Scope

```
service + environment + region + period + currency + owner
```

Scope에 반드시 답합니다.

- 어떤 user journey와 service인가.
- Development·test·production 중 어디인가.
- 어떤 region·zone·billing account인가.
- 어느 billing·usage period인가.
- 어떤 direct·shared·people·risk 비용을 포함하는가.
- 무엇을 왜 제외했는가.

4.2 Usage

Usage	좋은 evidence	나쁜 evidence
Compute	vCPU-hour·GB-hour by service·env	Instance 수만 있음
Storage	평균 GB-month·request·tier	Peak GB만 있음
Network	Source→destination GB	전체 traffic GB
AI	Model version별 token·tool·retry	Request 수만 있음
Telemetry	GB·series-day·span·retention	Dashboard 수만 있음
License	Purchased·assigned·active·overage	Seat price만 있음

4.3 Rate

Rate card는 작은 database처럼 관리합니다.

```
rate_id
provider · service · SKU
meter · unit
region · tier · pricing model
list · contracted · effective rate
currency · as-of · source
```

4.4 Allocation

총액을 나누는 규칙도 versioned evidence입니다.

Cost	Driver 예시	주의
Shared cluster	CPU·memory request	Request가 value와 같은지는 확인
Observability	Log GB·series	공통 signal은 별도 정책
Support	Direct cost 비율	작은 팀 과부담 가능
SaaS	Active user	Purchased minimum은 별도
Governance	Even·revenue·headcount	목적에 맞는 driver 필요

4.5 Commercial·license

Price와 entitlement를 분리하지 않습니다.

```
product · vendor · metric
purchased · assigned · active · minimum
quota · overage · support · SLA
term · renewal · true-up · true-down
BYOL · marketplace · restriction
```

4.6 Value

Value evidence는 “저렴함”이 아니라 다음을 포함합니다.

- Business unit 정의
- Quality·success eligibility
- Cost per unit
- Target·budget·variance
- Low·base·high range

- Break-even·exit condition
- Owner·action·revalidation

5. Cloud 비용 driver 지도

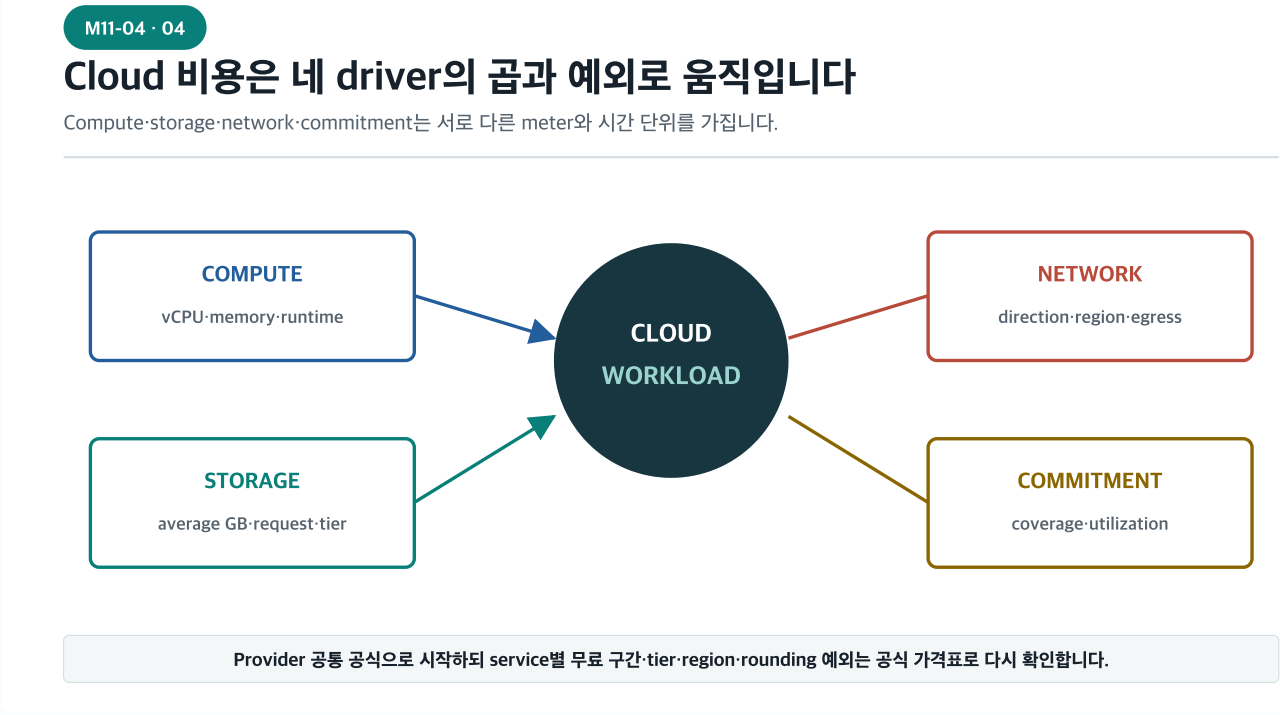
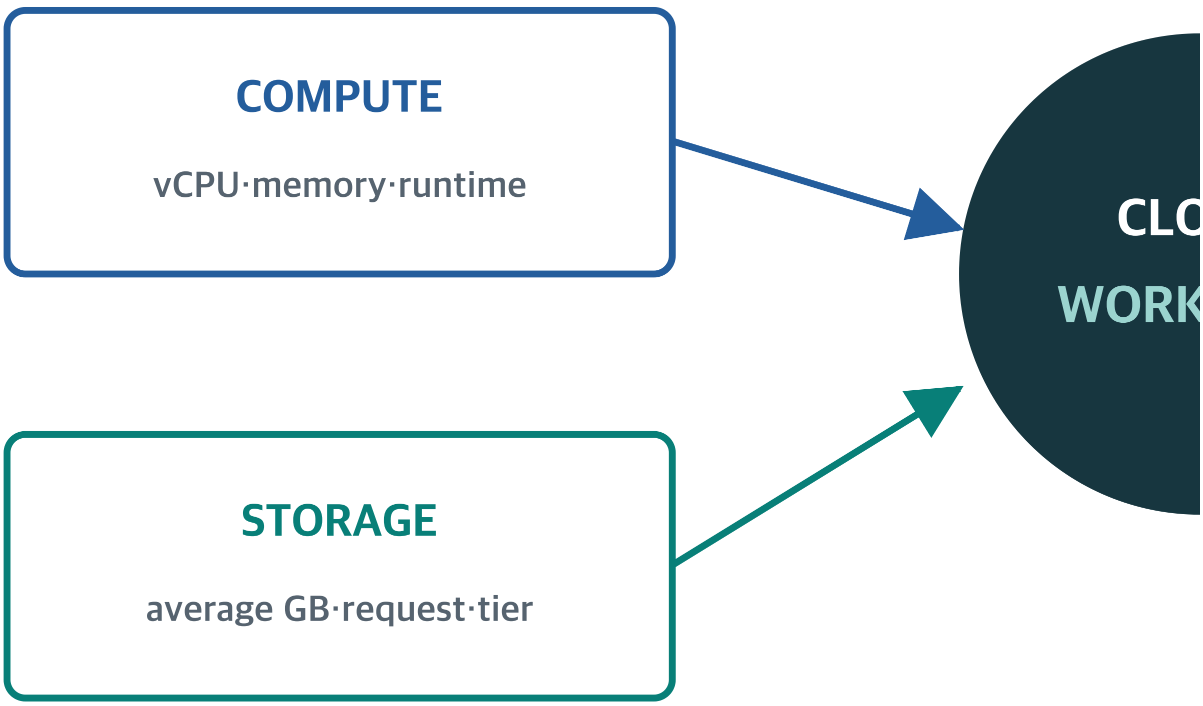
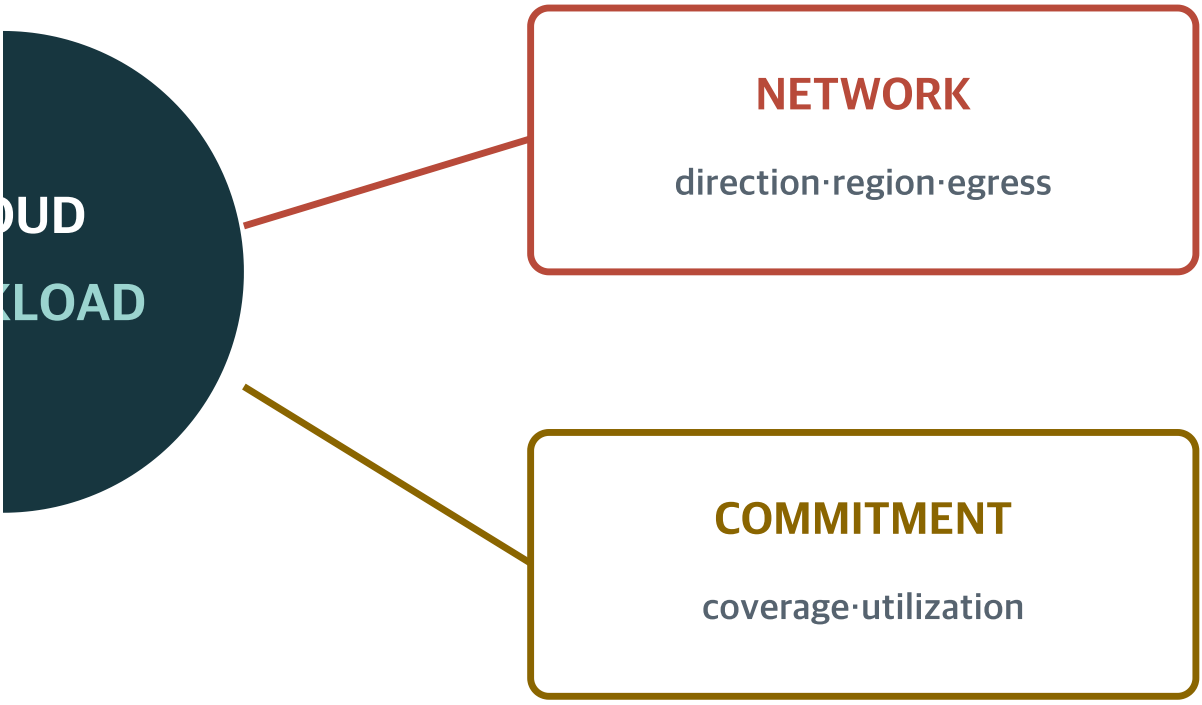


그림 4. Compute·storage·network·commitment는 서로 다른 meter와 예외를 가집니다. Provider-neutral 공식으로 시작하고 service-specific pricing으로 다시 검증합니다.

Compute·storage·network·commitment는 서로 다른 meter와 시간 단우



를 가집니다.


UD
LOAD
NETWORK
direction·region·egress
COMMITMENT
coverage·utilization

5.1 Compute

합성 workload:

```
4 vCPU × 730 hours = 2,920 vCPU-hour
16 GB × 730 hours = 11,680 GB-hour
```

합성 rate:

```
compute = $0.08 / vCPU-hour
memory = $0.01 / GB-hour
```

계산:

```
2,920 × 0.08 = $233.60
11,680 × 0.01 = $116.80
compute subtotal = $350.40
```

하지만 실제 model은 다음을 더 봅니다.

질문	Evidence
Autoscaling minimum은	Configuration·runtime count
Idle은 얼마인가	Provisioned–value-producing usage
Peak가 반복되는가	Time series·seasonality
Serverless rounding은	Duration·memory·minimum unit
GPU가 공유되는가	Allocation·utilization
Spot 중단을 견디는가	Workload checkpoint·fallback

5.2 Storage

```
500 GB-month × $0.025/GB-month = $12.50
```

“500GB”만으로 부족합니다.

- Time-weighted average인지 peak인지
- Object·block·file 중 무엇인지
- Hot·cool·archive tier
- Read·write·list request
- Retrieval·early deletion
- Snapshot·replication·backup 중복
- IOPS·throughput provision
- Retention·lifecycle

5.3 Network

```
300 GB egress × $0.09/GB = $27.00
```

Network cost evidence:

```
source → destination
direction
source region·zone
destination region·zone
internet·CDN·NAT·load balancer·private path
GB + processing charge + hourly charge
```

> Network는 “300GB”가 아니라 **어디에서 어디로 어떤 경계를 넘었는가**가 먼저입니다. Ingress가 무료라는 일반화도 service·path·processing charge의 예외를 가립니다.

5.4 Provider calculator를 쓰는 법

AWS Pricing과 AWS Pricing Calculator 같은 공식 도구는 architecture assumption을 빠르게 계산하는 데 유용합니다. 그러나 AWS calculator 문서도 결과가 estimate이며 실제 usage·currency·tax·discount·third-party license·support 등에 따라 달라질 수 있음을 설명합니다.

안전한 사용 순서:

1. Scope와 user journey를 먼저 적습니다.
2. Service·region·configuration을 선택합니다.

3. Usage와 pricing model assumption을 적습니다.
4. Tax·support·license 포함 여부를 확인합니다.
5. Estimate URL·export·as-of를 저장합니다.
6. Actual·invoice와 나중에 reconciliation합니다.

Azure Cost Management와 Google Cloud Billing reports도 비용 분석과 reporting을 제공하지만, account·offer·credit·currency·data latency를 확인해야 합니다.

6. 가격 모델은 위험의 모양이 다르다

그림 5. 할인율 하나로 비교하지 않습니다. On-demand는 단가 변동, commitment는 미사용, tier는 경계값, seat·license는 minimum·true-up·exit 위험을 가집니다.

[illegible]

On-demand·commitment·tier·seat 계약은 각각 다른 forecast와 exit 조

01

ON-DEMAND

유연성 높음

사용량 × list

변동 대응

단가 높을 수 있음

02

COMMIT

예측 가능

기간·금액 약정

coverage 관리

미사용 위험

건을 요구합니다.

03

TIERED

구간별 단가

marginal rate

무료 구간

경계값 주의

04

SEAT·LICENSE

계약 제약

minimum·true-up

entitlement

갱신 선행

6.1 On-demand

장점:

- 낮은 초기 commitment
- 빠른 scale up·down
- 실험과 변동 수요에 적합

위험:

- 고정된 수요에서는 높은 unit rate
- Usage drift가 바로 spend drift가 됨
- Budget 예측이 어려울 수 있음

6.2 Commitment

핵심 metric:

```
coverage = eligible usage covered / total eligible usage
utilization = consumed commitment / purchased commitment
```

Coverage만 높고 utilization이 낮으면 과다 구매일 수 있습니다. 먼저 workload 안정성·architecture 계획·exit horizon을 봅니다.

6.3 Tiered·graduated

예시:

```
0–100 units: $1.00
101–500 units: $0.80
501+ units: $0.60
```

전체 사용량에 하나의 rate를 적용하는 volume 방식인지, 각 구간을 따로 합산하는 graduated 방식인지 확인합니다.

6.4 Seat·license

Seat model의 수량은 하나가 아닙니다.

수량	의미
Purchased	계약상 구매
Assigned	계정에 할당
Active	기간 중 활동
Concurrent	같은 순간 접속
Minimum	사용과 무관한 하한
Overage	포함 범위 초과

6.5 Break-even 질문

```
at what usage does commitment TCO < on-demand TCO?
```

단, TCO에는 선불금·미사용·migration·운영·exit·reliability 위험도 포함합니다.

7. AI 비용은 요청 하나 안에서 갈라진다

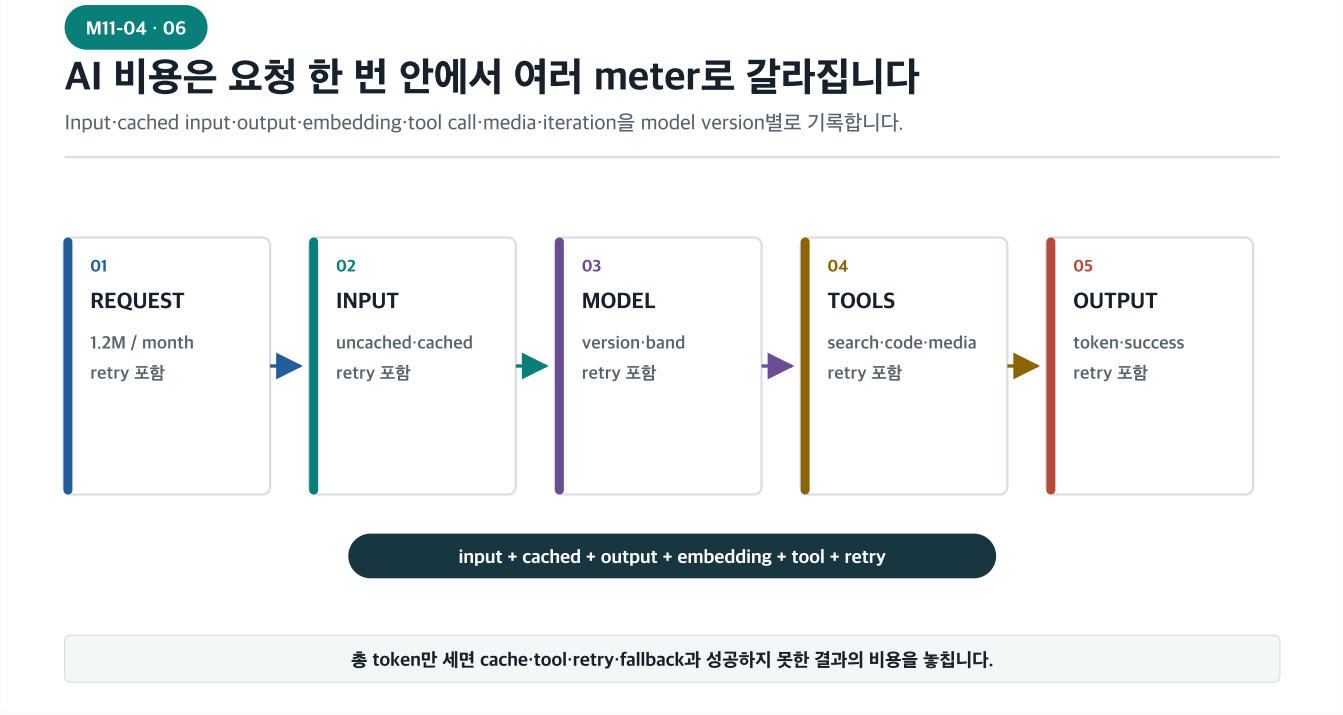
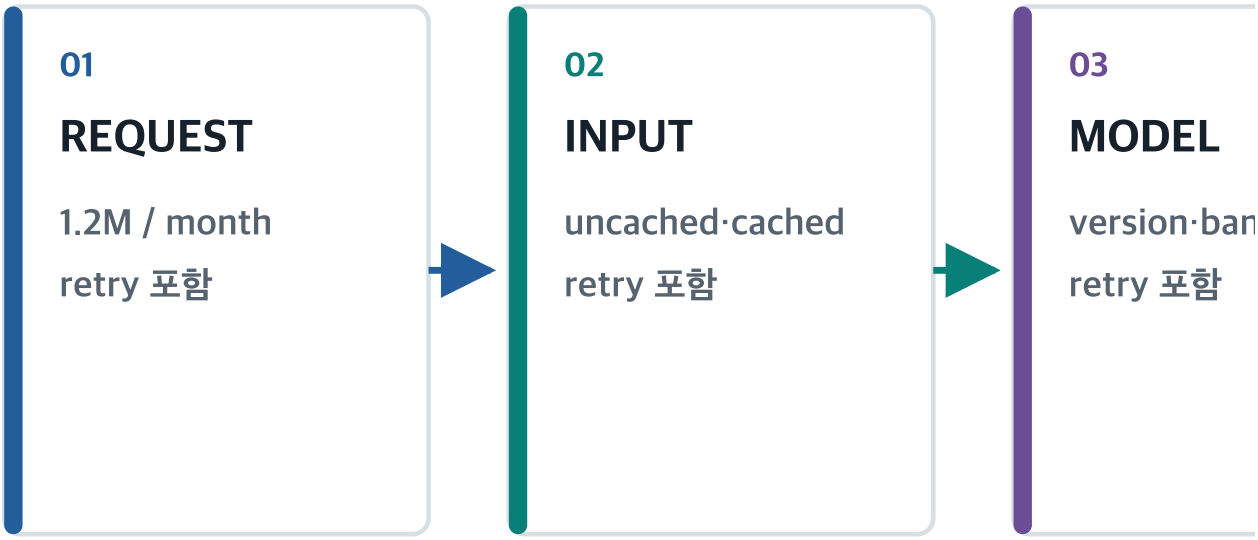


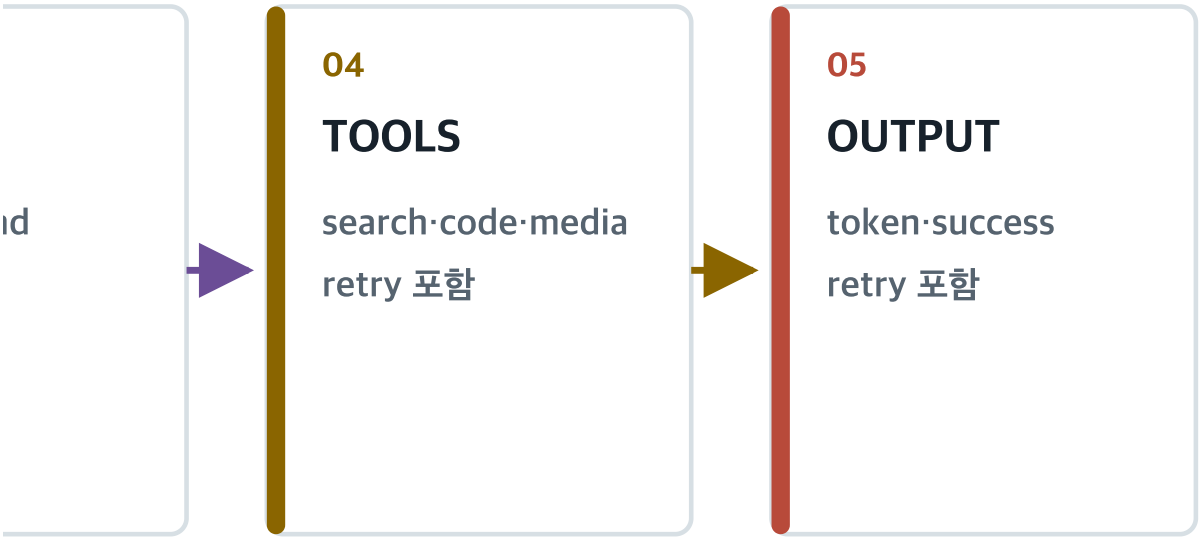
그림 6. Request 수만 세면 부족합니다. Input·cached input·output·embedding·tool·media·iteration·retry를 정확한 model version과 pricing band에 연결합니다.

Input·cached input·output·embedding·tool call·media·iteration을 mo



input + cached + output +

del version별로 기록합니다.



embedding + tool + retry

7.1 합성 workload

```
monthly requests = 1,200,000
input token/request = 800
cached input share = 40%
output token/request = 220
embedding = 80M token/month
tool calls/request = 0.08
```

모든 숫자는 허구의 학습 값입니다.

7.2 Input·cached input·output

```
total input = 1,200,000 × 800 = 960M
cached input = 960M × 0.40 = 384M
uncached input = 960M − 384M = 576M
output = 1,200,000 × 220 = 264M
```

합성 rate:

Meter	Usage	Rate	Cost
Uncached input	576M	$1.20/M	$691.20
Cached input	384M	$0.30/M	$115.20
Output	264M	$4.80/M	$1,267.20

AI input cost는 “960M × input rate” 하나가 아닙니다. Cache eligibility·prefix stability·minimum length·expiration 같은 provider 조건을 확인해야 합니다. OpenAI Prompt Caching 공식 가이드는 cache 사용과 관찰 방법을 설명합니다. 실제 적용 시 model별 최신 조건과 rate를 같은 기준일로 확인합니다.

7.3 Model·version·context band

`latest` 같은 alias는 행동과 가격이 바뀔 수 있습니다.

비용 evidence:

- Exact model identifier·version
- Input·cached input·output rate

- Context window와 long-context threshold
- Reasoning·multimodal meter
- Batch·priority·realtime pricing mode
- Tool별 별도 charge
- Rate as-of·source URL

OpenAI model 비교와 각 model page는 input·cached input·output 등 pricing dimension을 보여 줍니다. 이 매뉴얼은 특정 현재 model 가격을 실습 rate로 고정하지 않습니다. Gemini API pricing도 model·context·feature별 dimension을 공식 문서로 다시 확인해야 합니다.

7.4 Embedding·RAG

Embedding 비용만 세면 부족합니다.

```
source documents
→ parse·chunk
→ embedding
→ vector storage·index
→ query embedding
→ retrieval
→ reranking
→ prompt context
→ generation
```

Driver	질문
Source volume	문서·chunk가 몇 개인가
Embedding dimension	Vector 한 개가 얼마나 큰가
Reindex	Model·chunk·source 변경 주기는
Vector storage	Replication·metadata·index overhead는
Query	사용자 request당 검색 횟수는
Reranking	후보 몇 개를 어떤 model로 재평가하는가
Context	Retrieved token이 input을 얼마나 늘리는가

합성 값:

```
80M embedding token × $0.08/M = $6.40
```

작아 보여도 full reindex·growth·vector storage·query·generation context까지 합쳐야 합니다.

7.5 Tool·agent iteration

```
1,200,000 requests × 0.08 call/request = 96,000 calls
96,000 × $0.004/call = $384.00
```

Agent는 한 request 안에서 tool을 반복할 수 있습니다.

통제	비용 효과	품질·안전 효과
Iteration cap	무한 loop 차단	불완전 종료 가능
Tool allowlist	불필요 call 감소	기능 제한
Cache	반복 search·lookup 감소	Stale data 위험
Stop condition	성공 뒤 추가 call 차단	Oracle 필요
Timeout·retry cap	폭주 제한	일시 실패 허용
Human escalation	비싼 loop 중단	Labor·latency 추가

7.6 Retry·fallback·failure cost

성공한 request만 보면 실패 비용이 사라집니다.

```
total model calls
= initial calls
+ retry calls
+ fallback calls
+ validation calls
+ repair calls
```

기록할 것:

- Timeout·rate-limit·server error retry
- Invalid schema repair

- Low-quality regenerate
- Safety rejection handling
- Fallback model
- Human review
- Failed outcome denominator

7.7 Batch·비동기

OpenAI Batch 공식 가이드는 비동기 batch 작업 흐름을 설명합니다. Batch는 즉시 응답이 필요 없는 작업에서 다른 가격·throughput tradeoff를 가질 수 있지만, deadline·failure handling·data residency·model availability를 확인해야 합니다.

비교 질문:

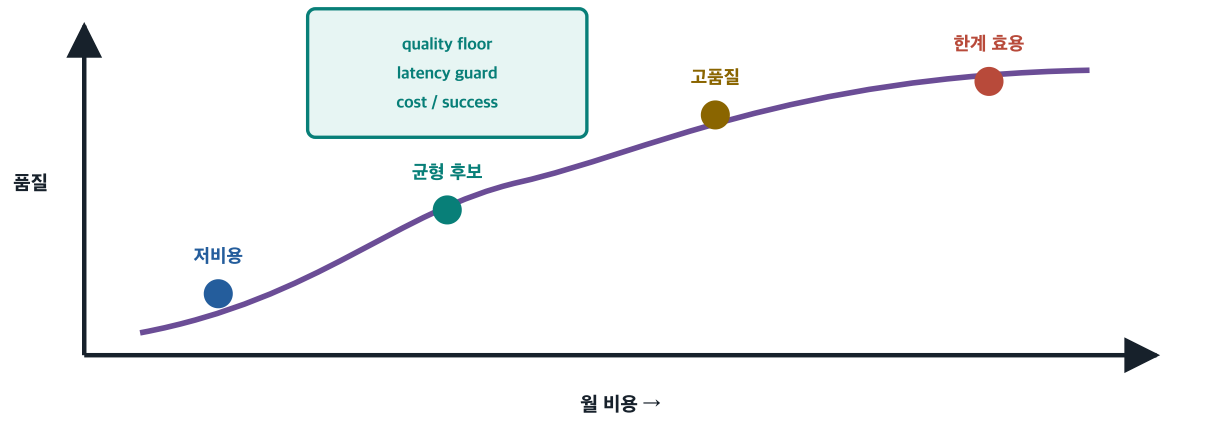
Mode	Latency	Rate	Queue·failure	적합 workload
Realtime	짧음	보통 높음	즉시 retry·fallback	User interaction
Batch	길 수 있음	할인 가능	Job lifecycle·partial failure	Offline evaluation·enrichment

8. 가장 싼 model보다 품질·지연·비용 경계를 본다

M11-04 · 07

가장 싼 model이 아니라 품질·지연·비용 경계의 후보를 고릅니다

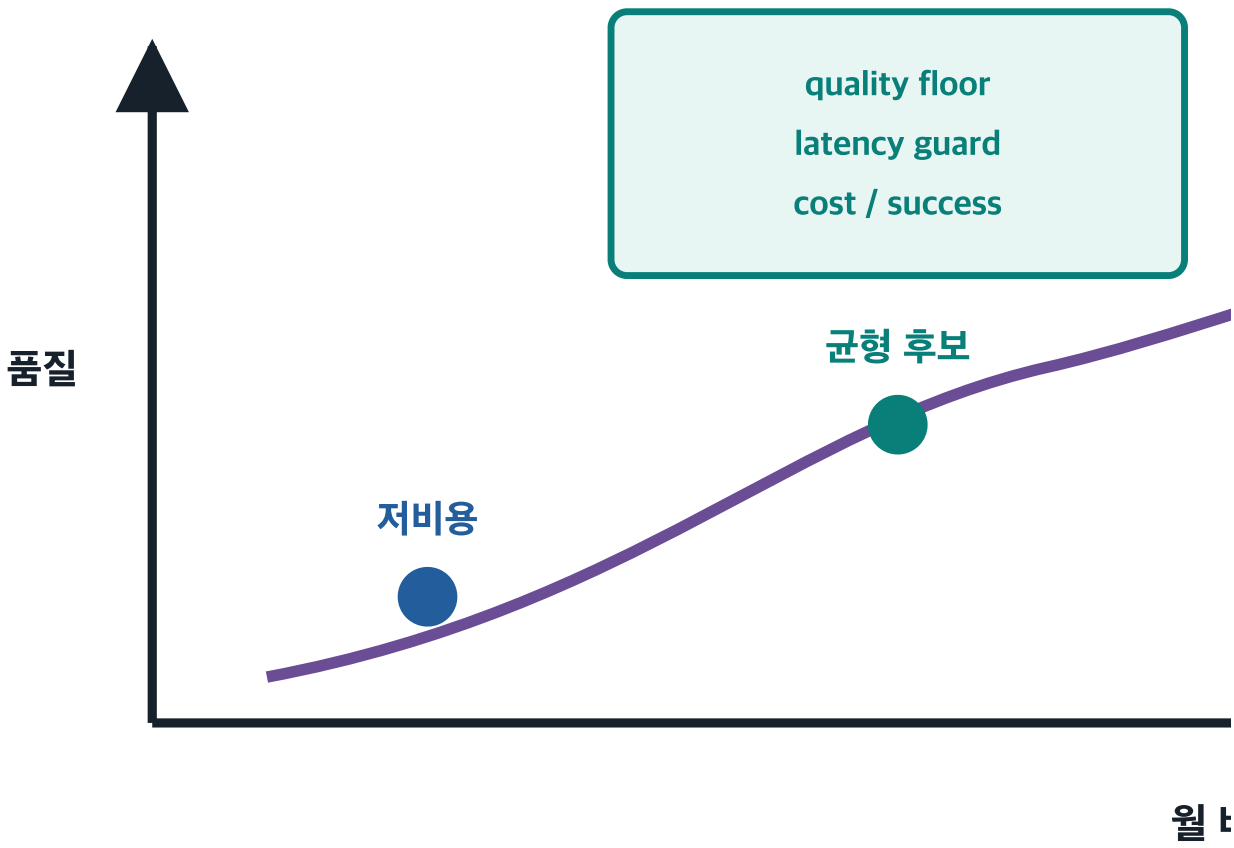
Model·prompt·context·tool을 바꾸며 성공 결과당 비용과 품질 기준을 동시에 봅니다.



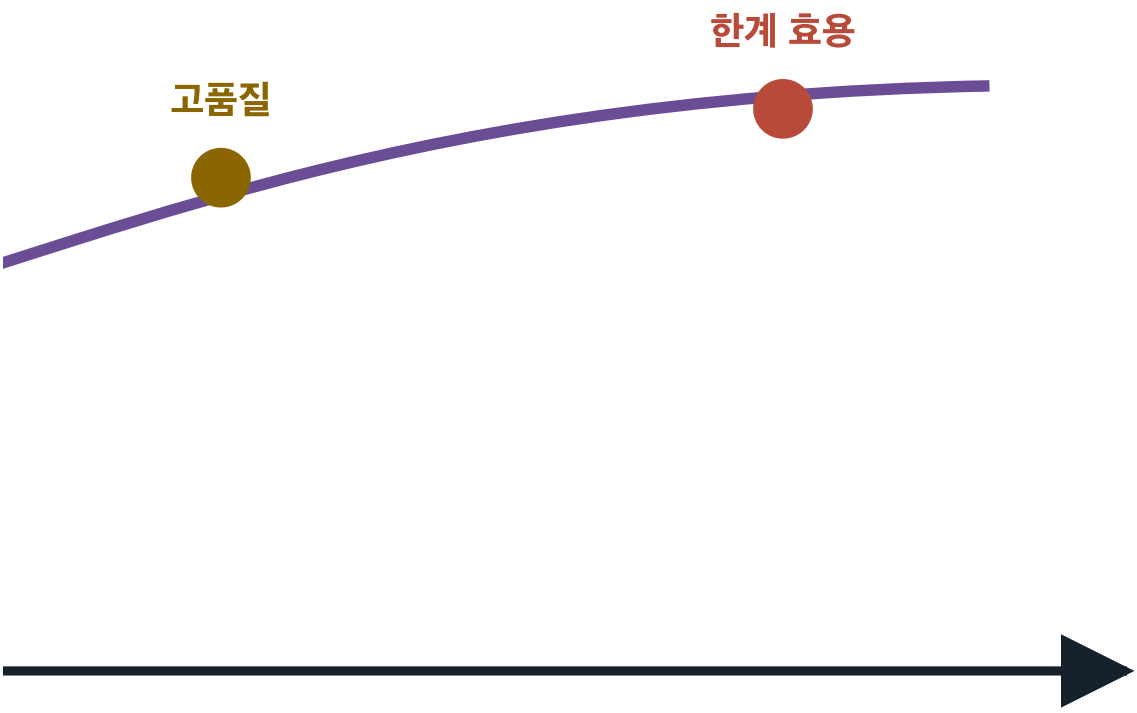
품질 floor 아래의 절감과 retry·fallback을 늘리는 절감은 실제 unit economics를 악화시킬 수 있습니다.

그림 7. Model·prompt·context·tool을 바꾸며 quality floor·latency objective·cost per success를 동시에 비교합니다. 값싼 실패는 절감이 아닙니다.

Model·prompt·context·tool을 바꾸며 성공 결과당 비용과 품질 기준을 동



·시에 봅니다.



8.1 Success denominator를 먼저 정한다

나쁜 분모:

```
all API responses
```

더 나은 분모:

```
eligible request 중
schema valid
+ grounded evidence present
+ policy passed
+ task-specific quality threshold passed
+ latency objective met
```

8.2 합성 cost per success

```
eligible requests = 1,200,000
success rate = 98.5%
successful outcomes = 1,182,000
fully loaded monthly total = $6,141.23
cost per success = $6,141.23 / 1,182,000 = $0.005196
```

8.3 세 후보 비교

후보	월 model cost	Quality	Success	p95 latency	Cost/success	판정
Low-cost	낮음	Floor 미달	낮음	빠름	실패 때문에 상승	Reject
Balanced	중간	Floor 통과	높음	목표 안	최소 유효	Candidate
High-quality	높음	조금 향상	비슷	느림	한계 효용	Review

8.4 비용 절감 실험의 순서

1. Success·quality·latency oracle을 고정합니다.
2. Baseline prompt·model·context·tool을 version으로 고정합니다.
3. 한 번에 한 driver만 바꿉니다.

4. Token·tool·retry와 quality를 함께 측정합니다.
5. Cost/request와 cost/success를 같이 계산합니다.
6. Regression·safety·user harm을 확인합니다.
7. Candidate를 점진 적용하고 revalidation합니다.

8.5 흔한 착시

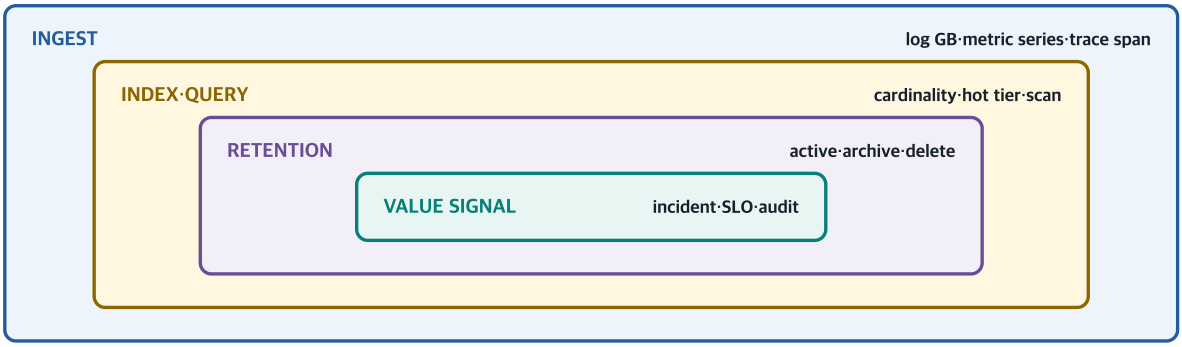
착시	왜 틀리는가
Input rate가 낮으니 싸다	Output·tool·retry가 큼
Request 수가 같으니 비용도 같다	Token length·model·iteration drift
Cache hit가 높으니 항상 좋다	Stale·privacy·eligibility 조건
Quality 평균이 같으니 좋다	Critical slice·tail failure 숨김
Cost/request가 낮다	Success denominator가 나빠질 수 있음

9. Observability 비용은 reliability evidence의 비용이다

M11-04 · 08

Observability 비용은 cardinality와 retention이 키웁니다

Log·metric·trace마다 수집·index·query·보존 meter와 가치 있는 signal의 기준이 다릅니다.

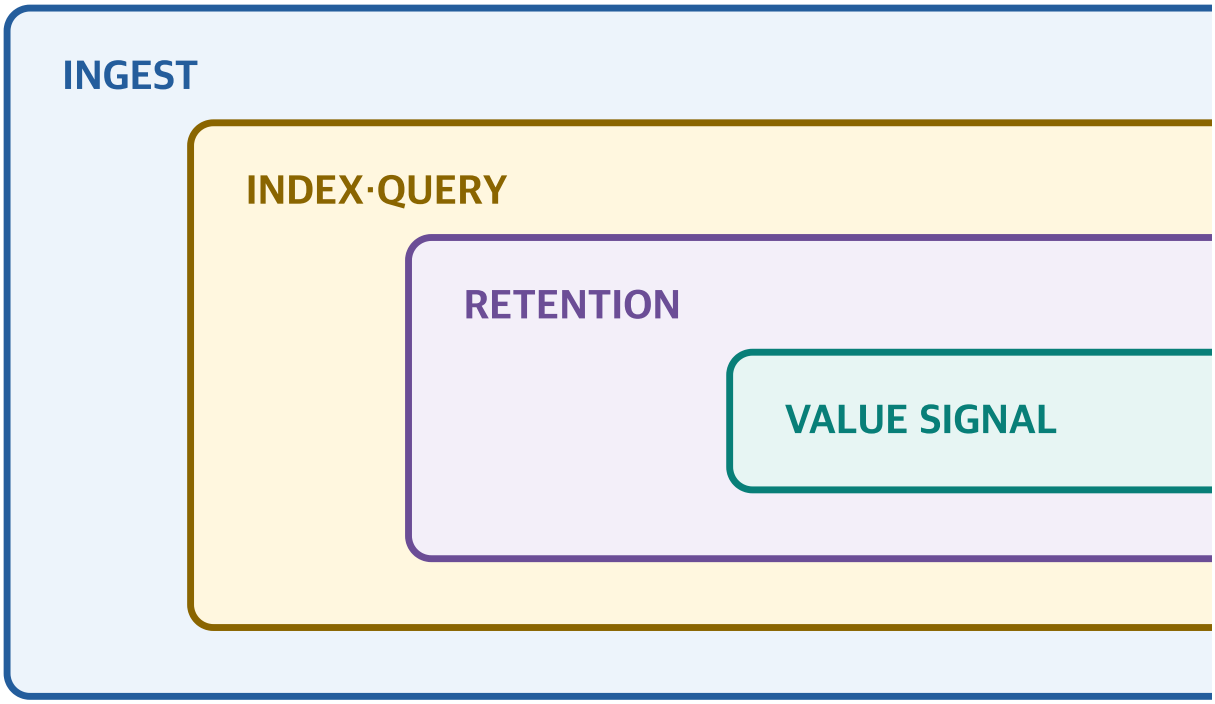


신뢰성을 해치며 지우지 말고 sampling·aggregation·tiering·retention을 evidence와 함께 조정합니다.

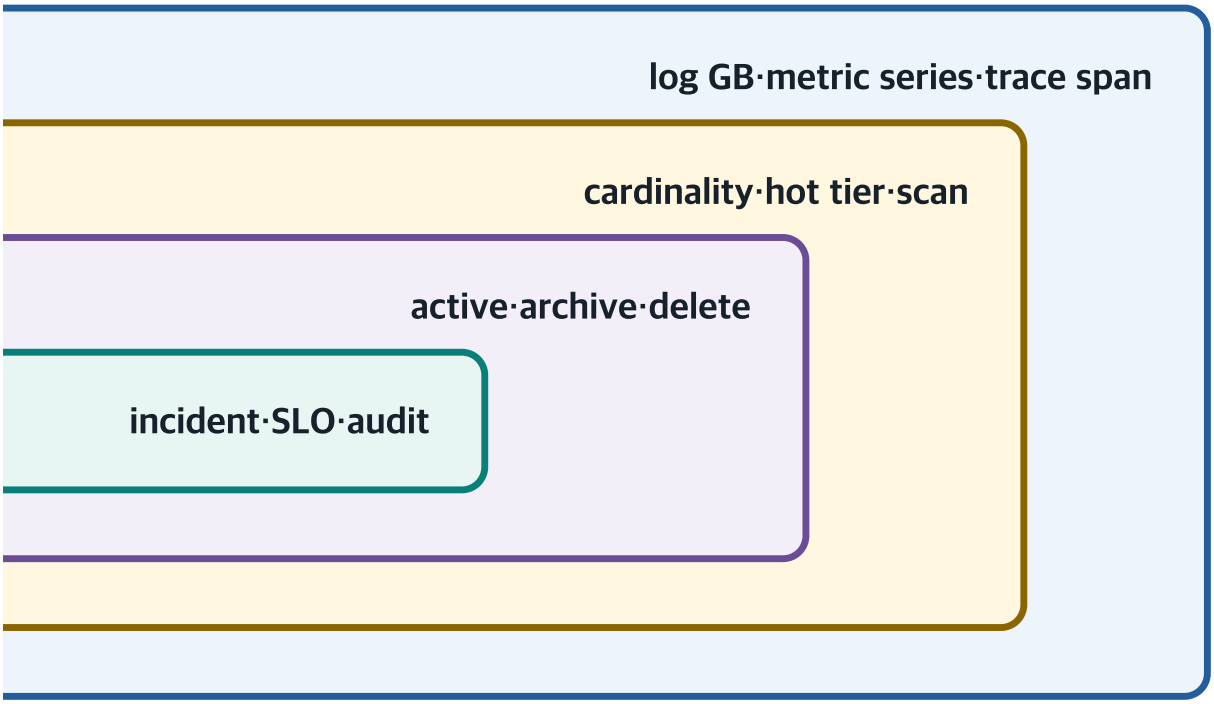
그림 8. Observability는 ingest만이 아닙니다. Index·query·cardinality·hot retention이 비용을 키울 수 있고, 마지막에는 incident·SLO·audit에 실제 쓰이는 value signal을 남겨야 합니다.

observability 비용은 cardinality에서

Log·metric·trace마다 수집·index·query·보존 meter와 가치 있는 signal으



| 기준이 다릅니다.


log GB·metric series·trace span
cardinality·hot tier·scan
active·archive·delete
incident·SLO·audit

9.1 Log

합성 값:

```
12 GB/day × 30 days = 360 GB/month
ingest: 360 × $0.45 = $162.00
archive: 360 × $0.02 = $7.20
```

실제 driver:

- Ingest GB
- Index GB·field
- Hot retention
- Archive GB-month
- Query scan GB·request
- Export·egress
- Access seats
- Compliance retention

9.2 Metric

```
25,000 active series × 30 days = 750,000 series-day
750,000 × $0.0008 = $600.00
```

이 합성 예시에서는 metric 비용이 log보다 큽니다. High-cardinality label 하나가 많은 series를 만들 수 있기 때문입니다.

```
series ≈ metric names × unique label combinations
```

비용을 줄이는 순서:

1. 사용하지 않는 metric·dashboard·alert를 찾습니다.
2. User ID·request ID 같은 unbounded label을 제거합니다.
3. Recording rule·aggregation을 검토합니다.
4. Scrape interval과 retention을 purpose별로 조정합니다.
5. SLI·security·capacity signal regression을 확인합니다.

9.3 Trace

```
18M spans × $0.20/M = $3.60
```

Trace가 싸 보이는 것은 합성 rate 때문일 뿐입니다. 실제로는 다음이 중요합니다.

- Head·tail sampling
- Error·slow trace retention
- Span attribute cardinality
- Payload·sensitive data exclusion
- Query·analysis volume
- Cross-service context propagation

9.4 Support·on-call

```
support fixed = $500/month
10 on-call seats × $15 = $150/month
operations subtotal = $1,422.80
```

Seat 수를 줄이기 전에 다음을 묻습니다.

- 누가 alert를 받고 ack하는가.
- Incident Commander·communications·scribe가 필요한가.
- Handoff와 training에 몇 명이 필요한가.
- Business hour와 24×7 coverage 차이는 무엇인가.
- Vendor support를 제거하면 MTTR·risk가 어떻게 바뀌는가.

9.5 Reliability를 해치지 않는 절감

나쁜 절감	더 나은 접근
모든 log 7일로 축소	Purpose·class별 hot·archive·delete
Trace sampling 극단 축소	Error·slow·critical journey tail sampling
Metric label 무차별 제거	SLI·capacity·debug query 기준 정리
Alert tool seat 즉시 삭제	Role·coverage·license position 검토

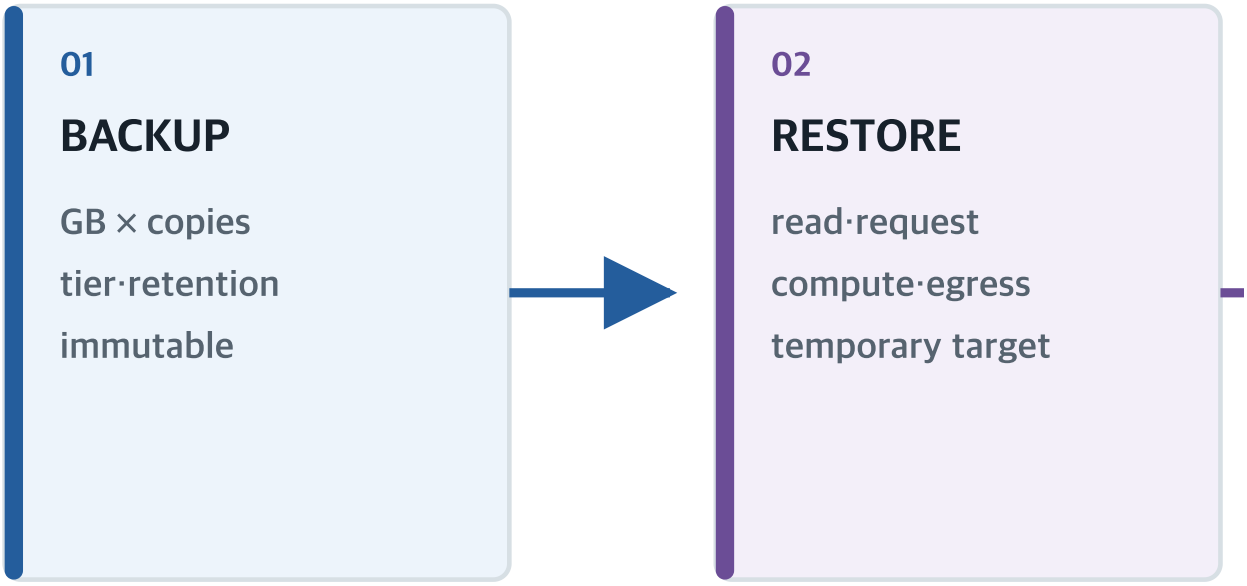
나쁜 절감	더 나은 접근
Monitoring 자체 삭제	Pipeline health·query canary 유지

10. Backup 비용은 restore evidence까지 포함한다

그림 9. Backup GB만 계산하면 restore read·compute·egress·temporary target·labor·validation·cleanup이 사라집니다. RPO·RTO·region 위험도 함께 봅니다.

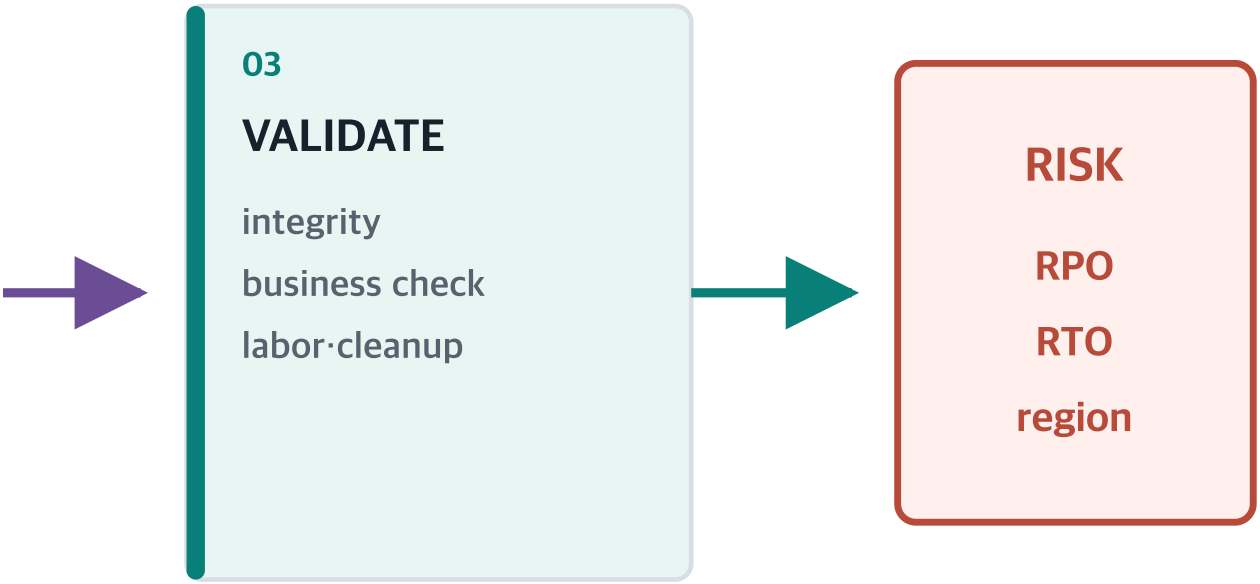
Backup 비용은 복원에서 다른 숫자입니다

Copy·tier·retention 뒤에 restore read·compute·egress·labor·검증·clea



[illegible]

nup 비용이 있습니다.


03
VALIDATE
integrity
business check
labor·cleanup
RISK
RPO
RTO
region

10.1 Backup storage

합성 값:

```
protected data = 800 GB
copies = 2
protected GB-month = 1,600
rate = $0.03/GB-month
backup cost = $48.00/month
```

실제 driver:

- Full·incremental·log backup
- Compression·deduplication
- Daily·weekly·monthly tier
- Immutability minimum period
- Cross-account·cross-region copy
- Key·identity material protection
- Catalog·checksum·monitoring

10.2 Restore drill

합성 값:

```
restore egress: 60GB × $0.05 = $3.00
restore compute fixed = $8.00
recovery subtotal = $59.00/month
```

아직 빠진 비용:

- Engineer labor
- Temporary isolated environment
- Database·queue·identity dependency
- Integrity·business validation
- Cleanup·access revocation
- Failed drill rework
- Extra-region standing capacity

10.3 RPO·RTO와 비용

목표 변화	일반적인 비용 영향	위험
RPO 단축	Backup·replication 빈도 증가	Write·network·storage 증가
RTO 단축	Warm·hot standby 증가	Idle·license·region 증가
Retention 연장	Storage 증가	Privacy·legal·discovery 범위 증가
Copy 증가	Storage·transfer 증가	관리·key·삭제 복잡성 증가
Drill 증가	Compute·labor 증가	준비도 evidence 향상

10.4 “Backup이 비싸다”의 올바른 답

```
어떤 data class의 어떤 RPO·RTO·retention·threat를 위해
어떤 copy·tier·region·immutability·drill을 유지하는가?
```

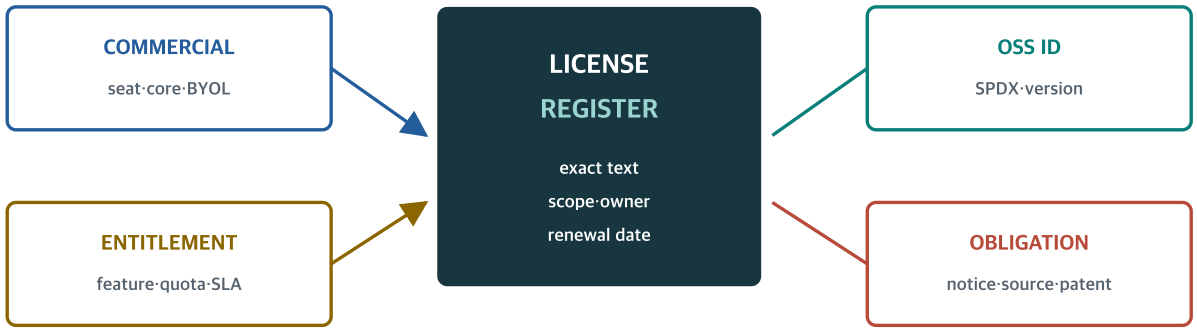
답 없이 copy를 지우면 비용을 줄인 것이 아니라 residual risk를 숨긴 것입니다.

11. License는 가격과 사용권·의무를 함께 본다

M11-04 · 10

라이선스는 가격과 사용권·의무를 같은 register에서 봅니다

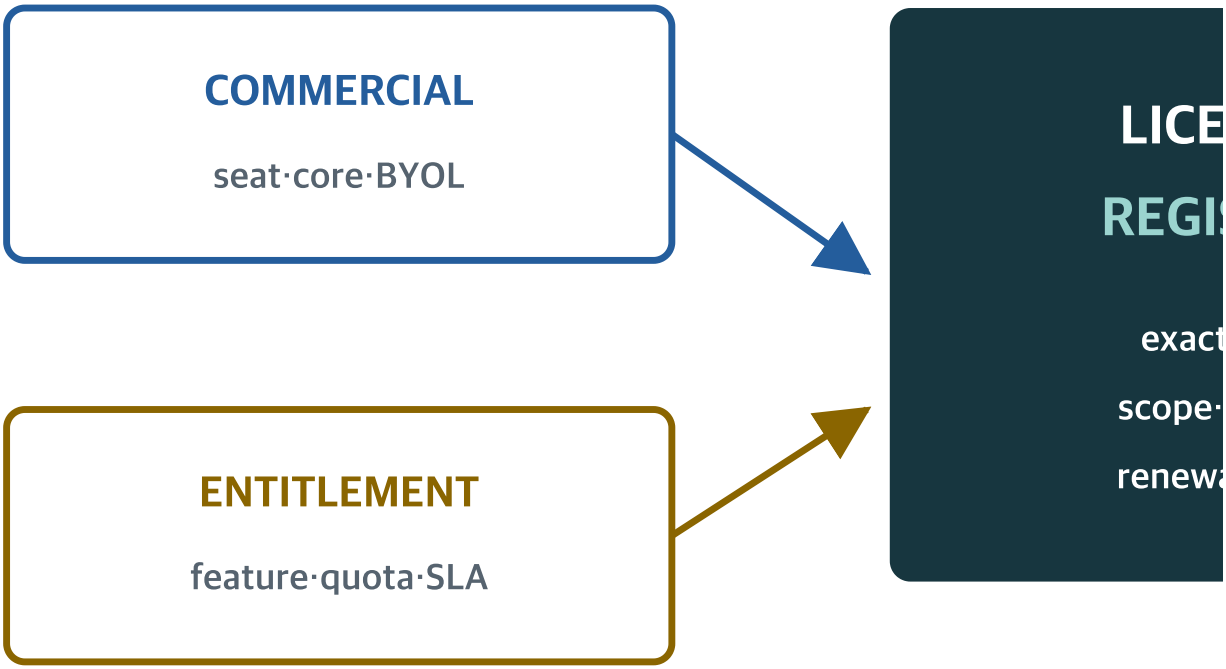
Commercial seat·entitlement·BYOL과 OSS identifier·notice·source·patent 조건을 분리해 기록합니다.



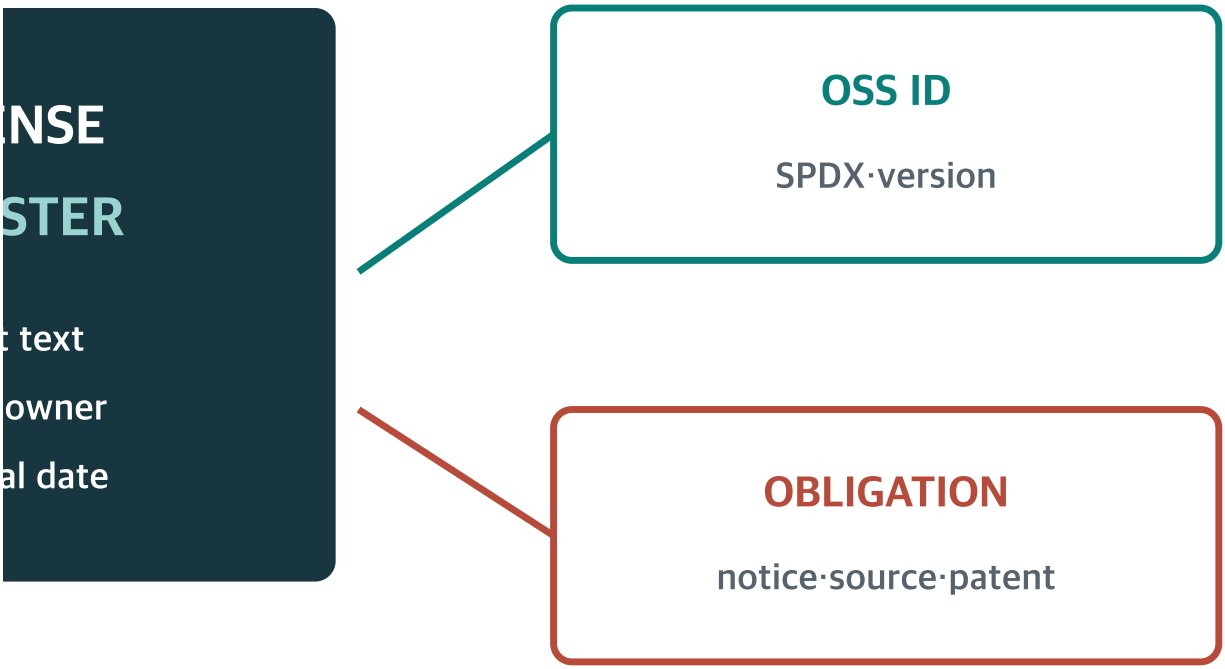
이 도표는 운영 체크입니다. 실제 배포 판단은 정확한 license text와 계약을 법무·구매가 검토합니다.

그림 10. Commercial license는 seat·core·BYOL과 entitlement·quota·SLA를, OSS는 identifier와 notice·source·patent 조건을 같은 register에서 추적합니다.

Commercial seat·entitlement·BYOL과 OSS identifier·notice·source·pa



tent 조건을 분리해 기록합니다.



11.1 Commercial·SaaS

FinOps Licensing & SaaS capability는 license와 SaaS가 seat·user·core·storage 같은 meter와 secondary usage charge를 가질 수 있고, minimum·overage·true-up·BYOL·marketplace·renewal·ITAM·SAM·procurement·legal 협업이 중요하다고 설명합니다.

Commercial register:

필드	질문
Metric	Seat·active user·core·API·storage 중 무엇인가
Purchased	몇 단위를 계약했나
Assigned·active	실제 position은
Minimum	사용이 적어도 내는 비용은
Overage	초과 rate·threshold는
Entitlement	어떤 feature·version·support를 쓸 수 있나
Term	Start·end·renewal notice는
True-up·down	늘리거나 줄일 수 있는 시점은
BYOL	기존 권리를 cloud에 적용할 수 있나
Marketplace	기존 계약과 중복 구매인가

합성 비용:

```
18 SaaS seats × $24 = $432/month
commercial platform fixed = $250/month
```

11.2 Shelfware·overdeployment

```
shelfware = purchased entitlement − needed entitlement
overdeployment = actual use − allowed entitlement
```

Shelfware는 낭비, overdeployment는 추가 charge와 compliance risk가 될 수 있습니다. Renewal 직전이 아니라 충분한 lead time 전에 usage와 architecture를 최적화합니다.

11.3 OSS는 “무료”와 다르다

Open source는 license가 없다는 뜻이 아닙니다. OSI Approved Licenses는 Open Source Definition과 review를 통과한 license 목록을 제공합니다. SPDX License List는 standard short identifier·full name·license text·permanent URL로 license를 일관되게 식별하도록 돕습니다.

하지만 SPDX ID는 법률 결론이 아닙니다.

```
component + exact version + exact license text
+ how used·modified·combined·distributed·network-served
+ notice·source·patent·trademark obligation
+ legal review
= deployable decision
```

11.4 OSS register

필드	이유
Component·version	License가 version별로 달라질 수 있음
Direct·transitive	Build 결과에 포함되는 경로 추적
SPDX expression	AND·OR·WITH 관계 표현
Exact text source	Identifier 오류·custom term 확인
Modified	Source·notice 조건 영향
Distributed	의무 trigger 가능
Network service	Network copyleft 검토
Notice·source	Release artifact 준비
Patent·trademark	별도 권리·제한
Legal owner·due	책임과 release gate

11.5 License category를 판결문처럼 쓰지 않는다

분류	학습용 관찰	실제 행동
Permissive	일반적으로 조건이 적음	Notice·copyright·patent 확인
Weak copyleft	File·module 범위 가능	결합·수정·배포 검토

분류	학습용 관찰	실제 행동
Strong copyleft	넓은 source 의무 가능	Derivative·distribution 법무 검토
Network copyleft	Network 제공 조건 가능	Hosted use trigger 검토
No license·custom	기본 사용권 불명확	사용 중단 또는 권리 확인

> 이 장은 운영 checklist이지 법률 자문이 아닙니다. 실제 배포·판매·network 제공은 정확한 license text와 사실관계를 법무가 검토합니다.

11.6 Governance cost

합성 값:

```
annual OSS review effort = $2,400
monthly amortized = $200
```

Governance를 0으로 적는다고 의무가 사라지지 않습니다. SBOM·notice bundle·source delivery·review·labor를 TCO에 포함합니다.

---

12. 다섯 cost state를 섞지 않는다

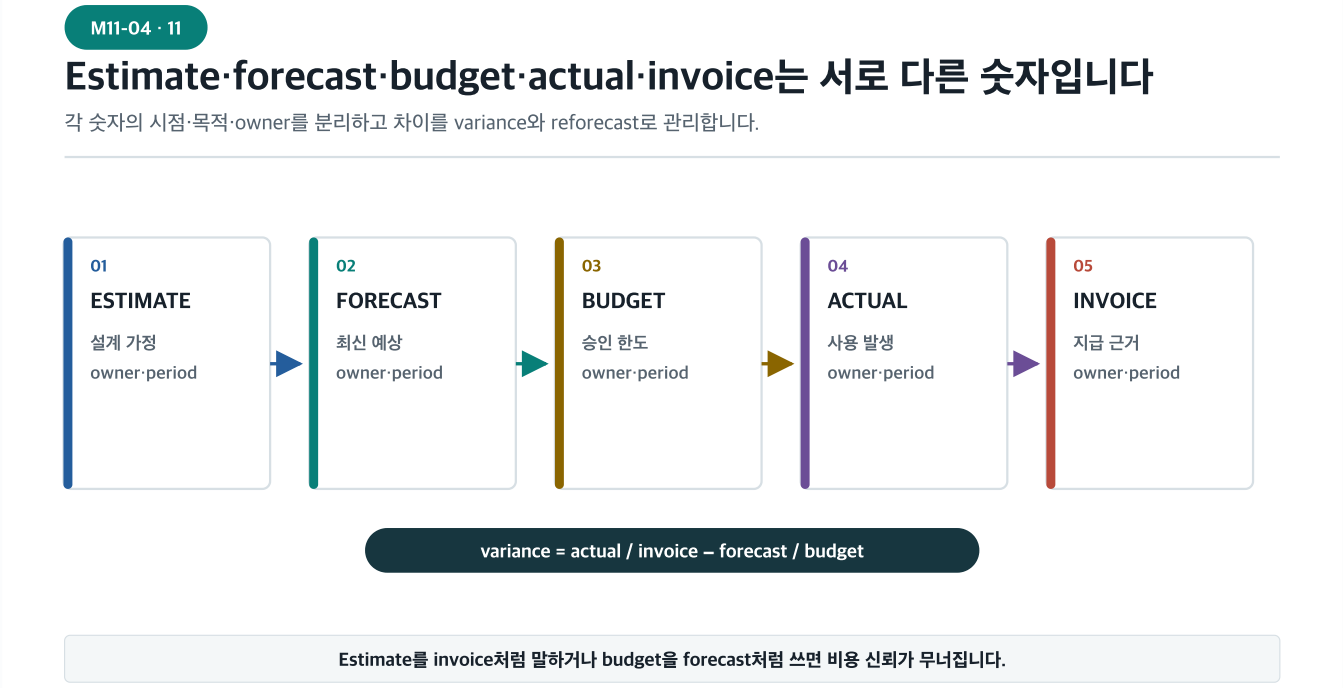


그림 11. Estimate는 설계 가정, forecast는 최신 예상, budget은 승인 한도, actual은 발생 비용, invoice는 지급 근거입니다. 시점과 owner를 분리해야 variance를 설명할 수 있습니다.

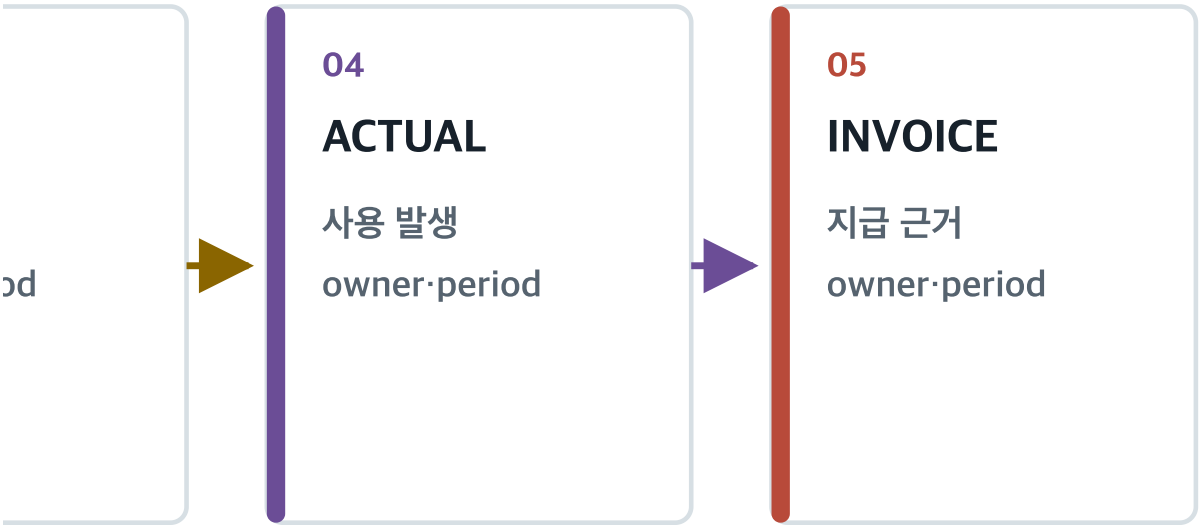
Estimate·forecast·budget·actual·fi

각 숫자의 시점·목적·owner를 분리하고 차이를 variance와 reforecast로 관

variance = actual / invo

[illegible]

[illegible]리합니다.



[illegible]ice – forecast / budget

12.1 Estimate

Architecture·usage·rate 가정으로 만든 사전 계산입니다.

Evidence:

- Calculator export·worksheet
- Assumption list
- Rate as-of
- Low·base·high
- Excluded cost

12.2 Forecast

최신 actual·trend·seasonality·launch plan·contract change를 반영합니다.

```
forecast = current run rate
+ planned growth
+ known change
+ seasonality
+ risk range
```

12.3 Budget

Budget은 예측이 아니라 승인·계획 한도입니다. Budget이 `$5,000`이라고 actual도 `$5,000`이 되는 것은 아닙니다.

12.4 Actual

Provider cost and usage data 또는 내부 원가 자료에서 이미 발생한 비용입니다. Data latency·correction·credit timing을 확인합니다.

12.5 Invoice

Invoice는 지급 근거입니다. Usage period와 billing period, tax·credit·adjustment, invoice issuer가 actual reporting과 다를 수 있습니다.

FOCUS Specification v1.4는 provider-neutral billing data schema를 정의하고 billed·effective·list·contracted cost, billing period, charge period, invoice detail, contract commitment, allocation 등 비용 분석에 필요한 공통 개념을 제공합니다.

12.6 Variance

```
forecast variance = actual − forecast
budget variance = actual − budget
invoice variance = invoice − reconciled actual
```

Variance log:

필드	질문
Period	언제의 차이인가
Category	어떤 cost driver인가
Expected·actual	같은 currency·scope인가
Root cause	Usage·rate·allocation·timing 중 무엇인가
Action	Fix·reforecast·accept 중 무엇인가
Owner·due	누가 언제 닫는가

13. Allocation은 비용을 책임과 가치에 연결한다

13.1 Direct·shared·unallocated

분류	의미	행동
Direct cost	한 product·service에 직접 귀속	그대로 배부
Shared cost	여러 대상이 공동 사용	Rule과 driver로 배부
Unallocated cost	Metadata·rule 부족	Threshold와 개선 owner

13.2 Tag coverage

```
tag coverage = cost with required tags / eligible cost
```

필수 key 예시:

- Service
- Environment
- Product
- Team
- Cost center
- Owner
- Data class
- License mode

Tag가 있다고 정확한 것은 아닙니다. 허용 값·대소문자·변경·상속·resource lifecycle을 관리합니다.

13.3 Shared cost driver

Driver	장점	위험
Even split	단순	사용량 차이 무시
Direct cost ratio	계산 쉬움	Shared value와 다를 수 있음
CPU·memory usage	기술 사용 반영	Business value와 차이
Request·transaction	활동 반영	Quality·복잡도 무시
Active user	SaaS에 적합	Minimum·shared account 무시
Revenue	가치 반영	초기 제품 불리 가능

Rule은 한 번 정하고 끝나지 않습니다.

```
rule_id + version + source + driver + 대상 + 비율 + owner + as-of
```

13.4 Credit·discount·tax

Credit을 한 team에 임의 귀속하면 unit economics가 왜곡됩니다.

- 어떤 purchase·usage에 의해 생긴 credit인가.
- Organization-wide benefit인가.
- Commitment discount를 누가 구매했나.

- Effective cost에 이미 amortized됐나.
- Tax는 usage cost와 별도인가.
- Refund·correction은 어느 period에 반영할까.

13.5 Showback·chargeback

방식	목적	필요한 신뢰 수준
Showback	사용과 비용 가시화	설명 가능·반복 가능
Chargeback	내부 비용 책임 이전	감사 가능·분쟁 처리·승인 필요

Allocation은 처벌 도구가 아니라 좋은 결정을 위한 feedback이어야 합니다.

---

14. Unit Economics와 Fully Loaded TCO

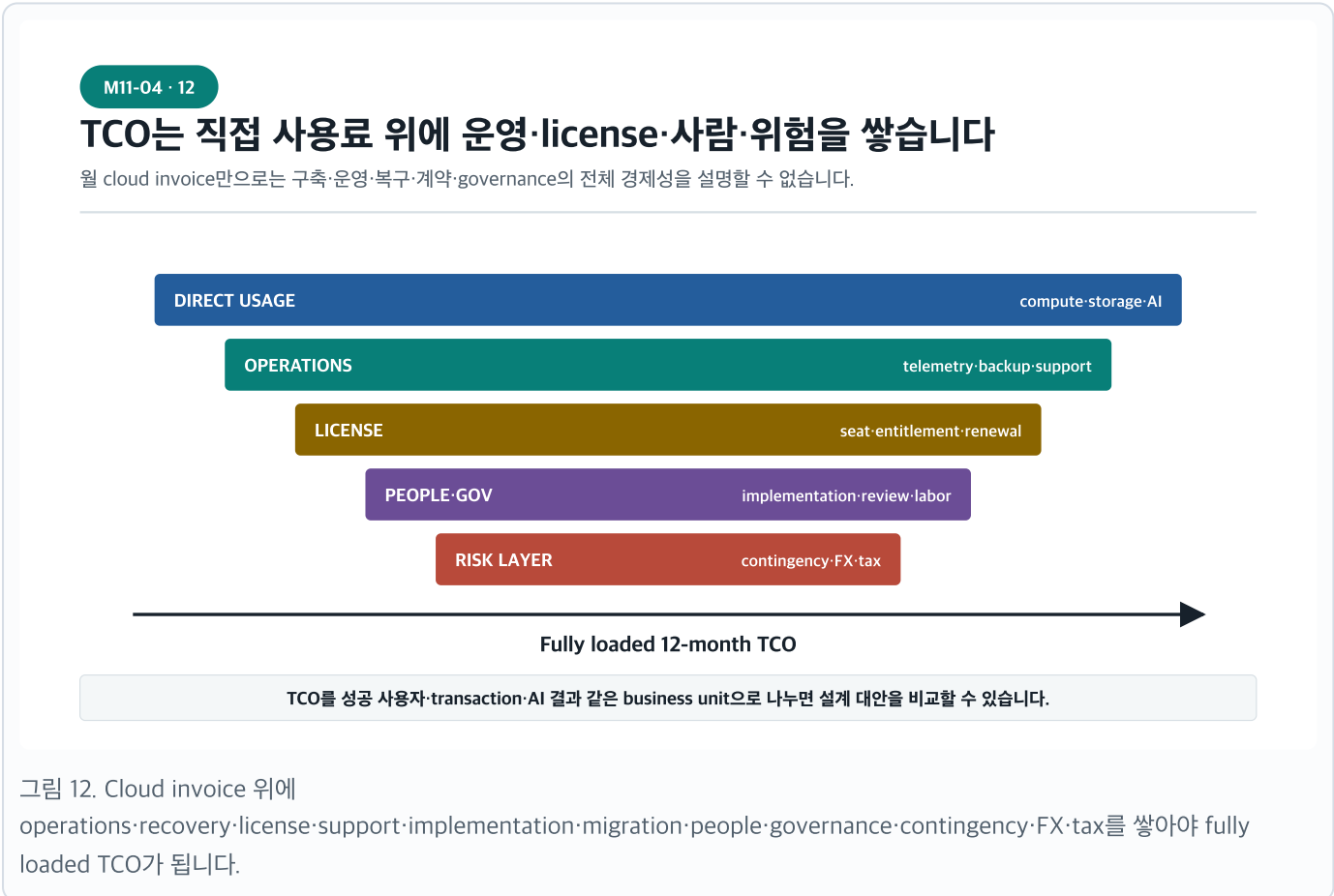
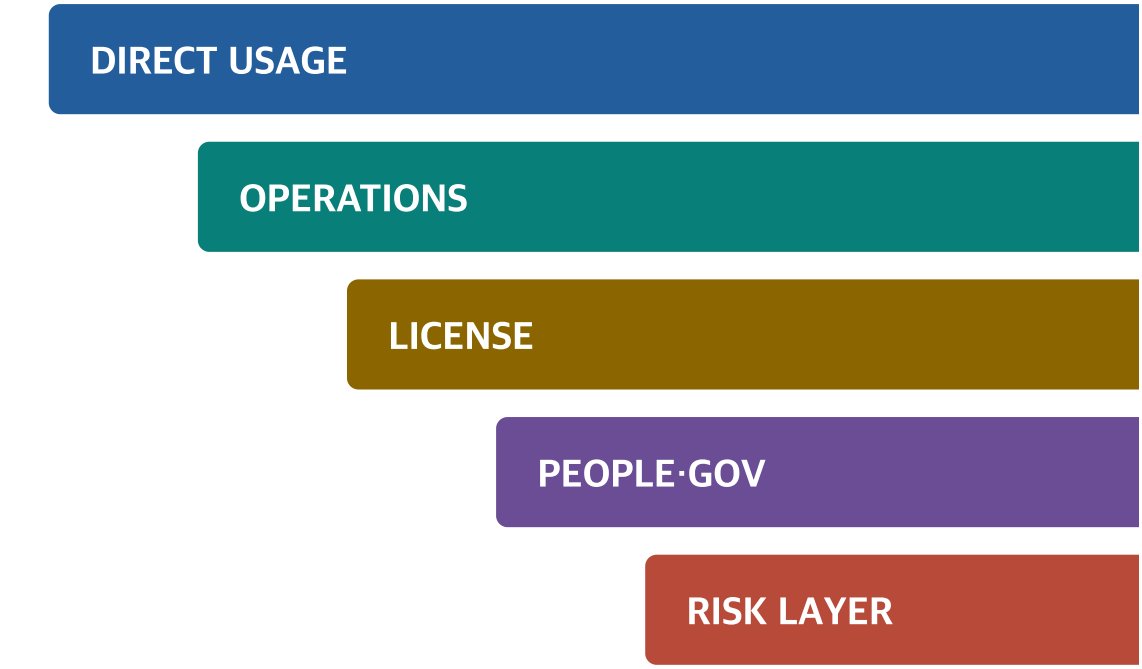


그림 12. Cloud invoice 위에 operations·recovery·license·support·implementation·migration·people·governance·contingency·FX·tax를 쌓아야 fully loaded TCO가 됩니다.

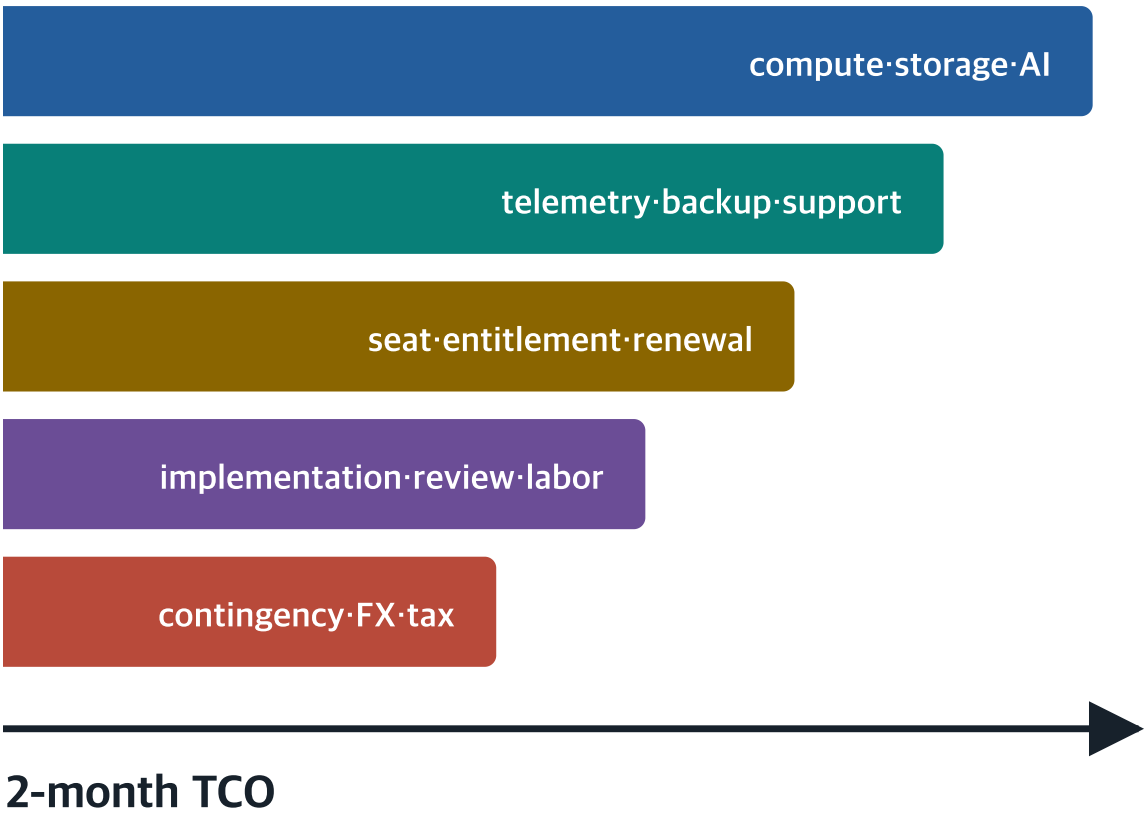
월 cloud invoice만으로는 구축·운영·복구·계약·governance의 전체 경제성



Fully loaded 1

[illegible]

[illegible]을 설명할 수 없습니다.


compute·storage·AI
telemetry·backup·support
seat·entitlement·renewal
implementation·review·labor
contingency·FX·tax
2-month TCO

14.1 합성 월 subtotal

Category	월 비용 USD	비율 관찰
Cloud	389.90	7.5% of subtotal
AI	2,464.00	47.2%
Operations	1,422.80	27.3%
Recovery	59.00	1.1%
Commercial	682.00	13.1%
Governance	200.00	3.8%
Subtotal	**5,217.70**	**100%**

이 예시에서 cloud compute만 최적화해도 전체 subtotal에 미치는 영향은 제한적일 수 있습니다. AI output·tool, metric cardinality, support, license가 큰 driver입니다.

14.2 Contingency·tax·FX

```
contingency = 5,217.70 × 7% = 365.24
pre-tax = 5,217.70 + 365.24 = 5,582.94
tax = 5,582.94 × 10% = 558.29
monthly total = 6,141.23 USD
reporting conversion = 6,141.23 × 1,400 = 8,597,726 KRW
```

모두 허구의 고정 학습 값입니다.

14.3 12개월 TCO

```
recurring 12 months ≈ $73,694.79
implementation one-time = $6,000
fully loaded 12-month TCO = $79,694.79
```

Engine은 component별 cent rounding을 거치므로 단순 표시값 곱셈과 몇 cent 차이가 날 수 있습니다. 실제 모델은 rounding 순서와 invoice tolerance를 문서화합니다.

14.4 TCO layer

Layer	포함 예시
Direct usage	Cloud·AI·data transfer
Operations	Telemetry·support·on-call·incident
Recovery	Backup·restore·drill·extra region
Commercial	SaaS·license·maintenance
Shared	Platform·security·network·governance
Implementation	Build·integration·test·training
Migration	Data·user·workflow·cutover
Exit	Export·termination·decommission
Risk	Contingency·FX·tax·residual risk

14.5 Resource unit에서 business unit으로

```
cost / token → model efficiency
cost / request → API efficiency
cost / success → quality-adjusted efficiency
cost / active user → product economics
cost / transaction → business economics
```

좋은 unit은 행동을 바꿉니다.

Unit	바꿀 수 있는 결정
Cost/token	Prompt·context·model·cache
Cost/request	Tool·retry·routing
Cost/success	Quality·fallback·human review
Cost/user	Seat·journey·retention
Cost/transaction	Pricing·product·architecture

14.6 분모 품질

분모가 커 보이게 실패 request를 모두 성공으로 세면 unit cost는 거짓으로 낮아집니다.

```
eligible
+ completed
+ quality passed
+ policy passed
+ latency passed
= successful business unit
```

15. 한 점 견적 대신 sensitivity와 break-even

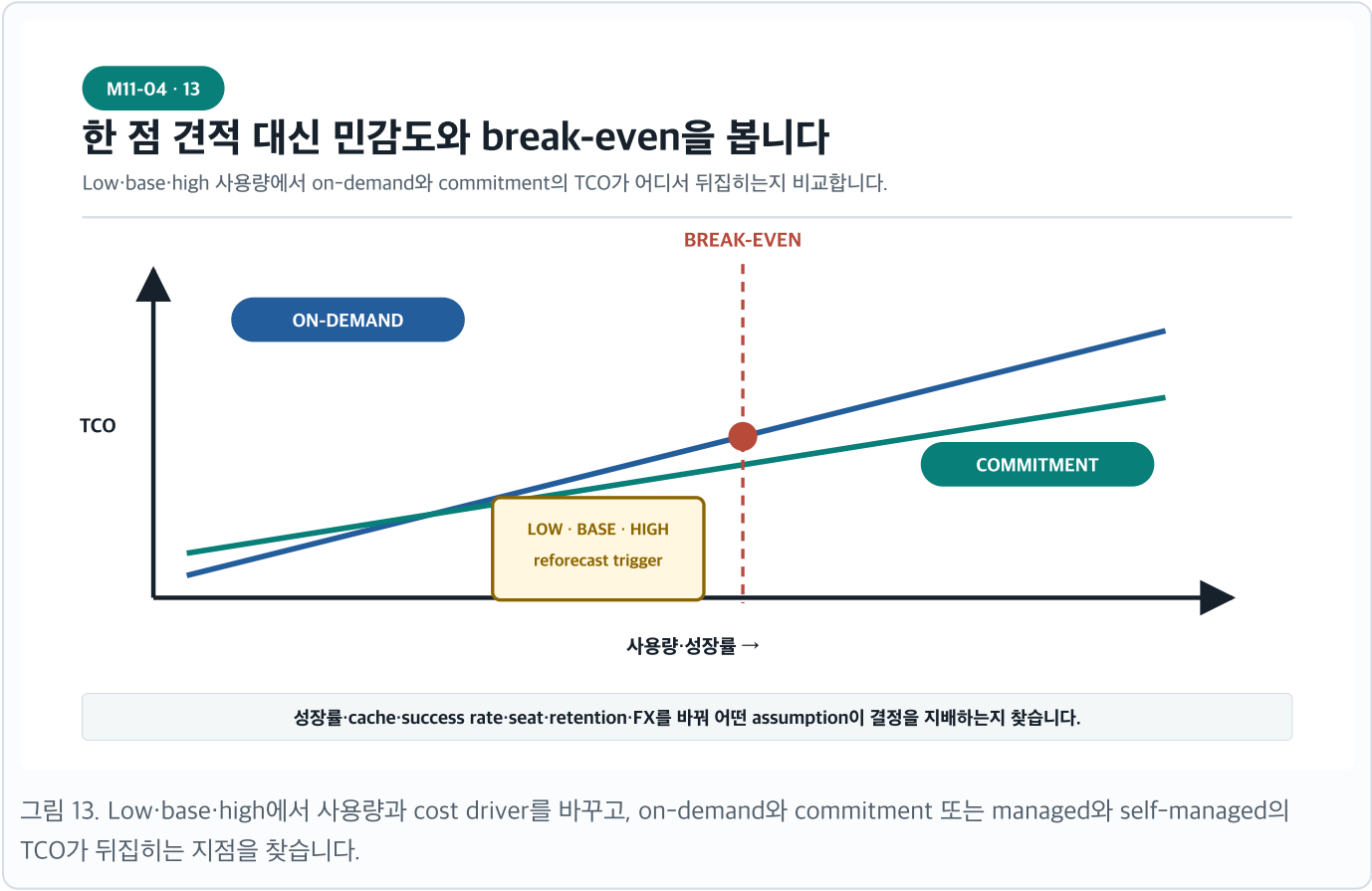
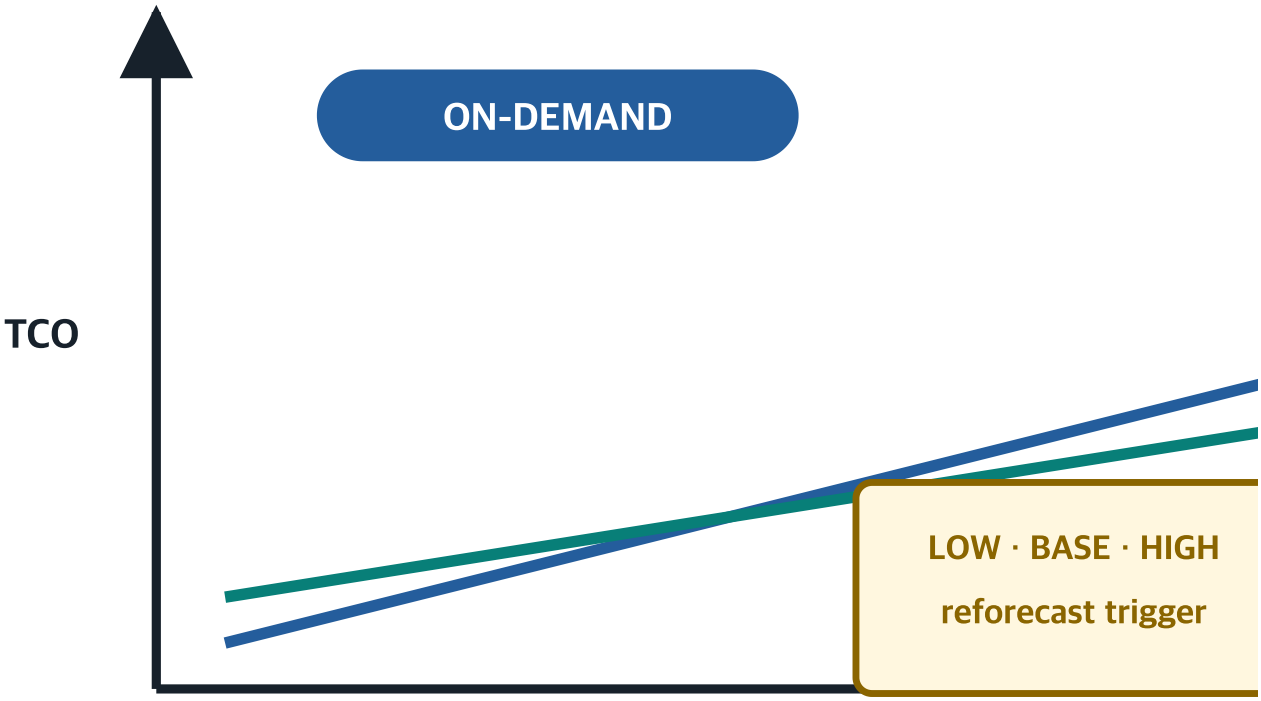


그림 13. Low·base·high에서 사용량과 cost driver를 바꾸고, on-demand와 commitment 또는 managed와 self-managed의 TCO가 뒤집히는 지점을 찾습니다.

[illegible]

Low·base·high 사용량에서 on-demand와 commitment의 TCO가 어디서



사용량··

뒤집히는지 비교합니다.

15.1 바꿔 볼 assumption

Assumption	Low	Base	High
Monthly requests	0.6M	1.2M	2.4M
Input/request	500	800	1,400
Output/request	120	220	420
Cache share	60%	40%	20%
Tool calls/request	0.03	0.08	0.20
Success rate	99%	98.5%	94%
Log GB/day	6	12	30
Metric series	12K	25K	80K
Active seats	12	18	30

위 표도 방법 예시일 뿐 실제 forecast가 아닙니다.

15.2 Tornado 질문

한 번에 하나씩 바꿉니다.

```
monthly total sensitivity
= Δrequests
+ Δinput length
+ Δoutput length
+ Δcache share
+ Δtool iteration
+ Δsuccess rate
+ Δtelemetry volume
+ Δseat
+ ΔFX
```

어느 입력이 결과를 가장 많이 바꾸는지 순서를 매깁니다.

15.3 Break-even

On-demand와 commitment:

```
on-demand TCO(u) = variable rate × usage u
commitment TCO(u) = fixed commitment + uncovered usage + unused commitment
```

Managed SaaS와 self-managed:

```
SaaS TCO = subscription + overage + integration + governance
self-managed TCO = infrastructure + license + operations + recovery + people
```

15.4 Reforecast trigger

- Usage가 base 대비 20% 이상 변함
- Model·version·pricing band 변경
- Context·tool·agent 설계 변경
- Quality·success rate floor 미달
- Contract·discount·seat·overage 변경
- FX·tax 정책 변경
- Retention·backup·SLO 변경
- New region·market·compliance scope

Threshold는 조직과 risk에 맞게 정합니다.

15.5 False precision 피하기

```
$6,141.23 = synthetic deterministic evidence
≠ actual future invoice guarantee
```

두 자리 소수는 계산 재현용이지 미래 정확성의 증거가 아닙니다. Range·assumption·confidence·as-of를 함께 보여 줍니다.

---

16. 비용 계산 스튜디오 읽기

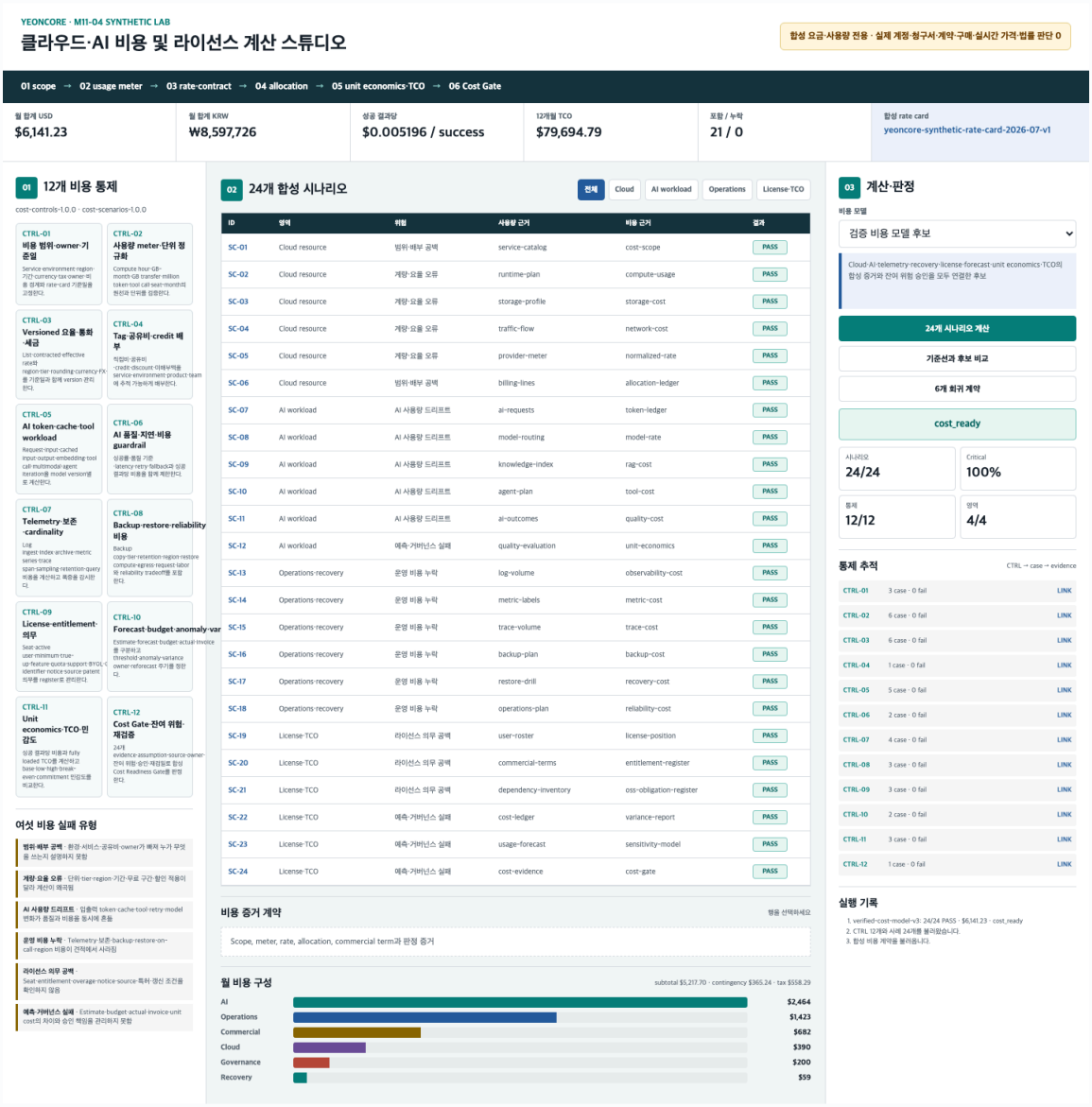


그림 14. Desktop 화면은 위에서 total·unit cost·TCO를, 왼쪽에서 12 control, 가운데에서 24 scenario, 오른쪽에서 decision·coverage, 아래에서 category driver를 한 번에 비교합니다.

16.1 세 모델

`list-price-only-v1`

```
6/24 pass
18 fail
monthly total $2,809.10
blocked_cost_blindness
```

포함:

- Cloud core
- AI input·output list calculation

주요 누락:

- Storage·network pricing detail
- Allocation
- Cache·embedding·tool·retry
- Metric·trace·backup·restore·support
- Entitlement·OSS
- Sensitivity·TCO·risk layer

`usage-without-license-v2`

```
16/24 pass
8 fail
monthly total $4,335.70
blocked_commercial_readiness
```

남은 실패:

```
SC-04 network direction
SC-08 model pricing band
SC-10 tool·agent
SC-16 backup
SC-20 entitlement
SC-21 OSS obligation
SC-23 sensitivity
SC-24 fully loaded Gate
```

`verified-cost-model-v3`

```
24/24 pass
critical 100%
12/12 controls
4/4 lanes
21 included / 0 omitted
cost_ready
```

16.2 화면을 읽는 순서

1. Rate card version·as-of를 확인합니다.
2. Included·omitted component를 확인합니다.
3. Decision과 pass·critical·coverage를 봅니다.
4. Category bar에서 큰 driver를 찾습니다.
5. FAIL scenario를 선택해 expected·actual mismatch를 봅니다.
6. Control을 눌러 linked scenario와 orphan을 봅니다.
7. Baseline→candidate fixed·regressed를 비교합니다.
8. Regression 6/6을 확인합니다.

16.3 Mobile 학습

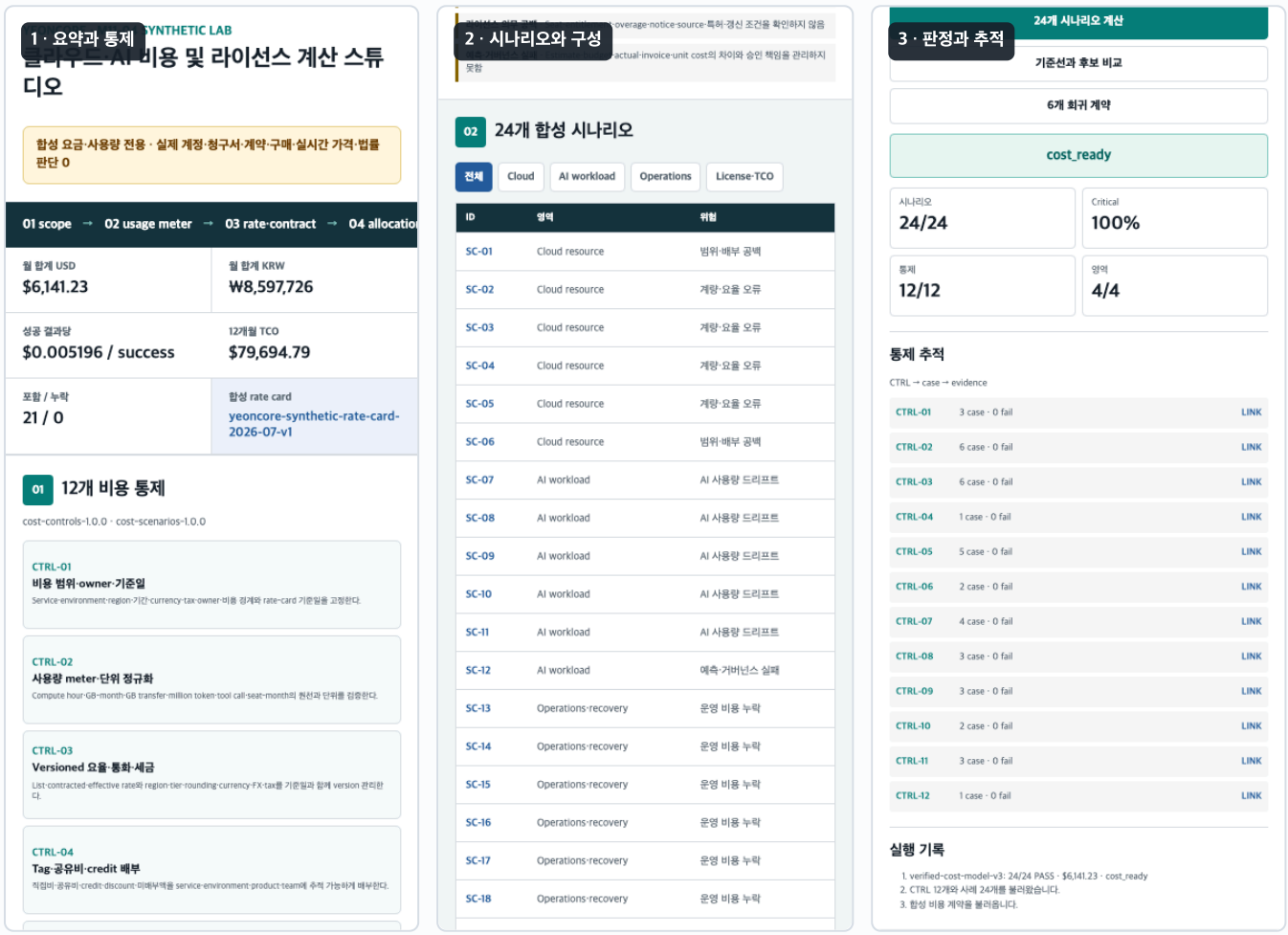


그림 15. 같은 390px 실제 화면을 세 구간으로 확대했습니다. 왼쪽부터 total·controls, scenarios·breakdown, decision·trace 순으로 읽습니다. Table은 내부에서만 가로 이동합니다.

16.4 실습 안전 경계

- Synthetic data only

- Standard library only
- Loopback `127.0.0.1` only
- External network 0
- Cloud·billing account 0
- Real invoice·contract 0
- Payment·purchase 0
- Live price·FX·tax 0
- Production resource 0
- Metadata-only trace

`cost_ready`는 합성 학습 판정입니다.

---

17. 12개 Cost Readiness Control

Control	질문	최소 evidence
CTRL-01	Scope·owner·as-of가 있는가	Scope register
CTRL-02	Meter·unit이 정규화됐는가	Usage ledger
CTRL-03	Rate·currency·tax가 versioned인가	Rate card
CTRL-04	Tag·shared·credit가 배부됐는가	Allocation ledger
CTRL-05	AI token·cache·tool이 있는가	AI workload profile
CTRL-06	Quality·latency·cost guardrail이 있는가	Cost/success evaluation
CTRL-07	Telemetry·retention·cardinality가 있는가	Observability cost model
CTRL-08	Backup·restore·reliability가 있는가	Recovery cost model
CTRL-09	Entitlement·OSS obligation이 있는가	License register
CTRL-10	Forecast·budget·variance가 있는가	Reconciliation report
CTRL-11	Unit economics·TCO·sensitivity가 있는가	TCO model
CTRL-12	Gate·residual risk·revalidation이 있는가	Decision record

17.1 Control은 비용 cap 하나가 아니다

나쁜 control:

```
월 $5,000을 넘지 않는다.
```

더 나은 control:

```
owner + scope + usage signal + threshold + action
+ quality·reliability floor + evidence + revalidation
```

17.2 Control coverage

Scenario가 control을 실제로 exercise해야 합니다.

```
CTRL-05
→ SC-07 token
→ SC-08 model rate
→ SC-09 RAG
→ SC-10 tool
→ SC-11 retry
→ evidence·actual·mismatch
```

---

18. 24개 Scenario를 네 lane으로 검증하기

18.1 Cloud resource · 6개

ID	Scenario	Expected evidence	대표 실패
SC-01	Service·environment·owner·scope·as-of	Cost scope	Owner·as-of 없음
SC-02	Compute runtime·autoscale·idle	Normalized vCPU·GB-hour	Provision만 있음
SC-03	Storage average·tier·request·retention	Storage cost	Peak GB만 있음
SC-04	Network direction·region·egress	Flow cost	전체 traffic GB

ID	Scenario	Expected evidence	대표 실패
SC-05	Meter·unit·tier·free·rounding	Rate-meter match	Unit mismatch
SC-06	Tag·shared·credit·discount·tax	Allocation ledger	총액만 있음

18.2 AI workload · 6개

ID	Scenario	Expected evidence	대표 실패
SC-07	Request·input·cached·output	Token ledger	Request만 셈
SC-08	Model·version·context·media·reasoning	Pinned rate	`latest` alias
SC-09	Embedding·vector·reindex·query	RAG cost	Embedding만 셈
SC-10	Tool·search·code·media·iteration	Tool cost	Agent loop 누락
SC-11	Retry·failure·fallback·guardrail	Failure cost	성공 call만 셈
SC-12	Quality·latency·success·unit cost	Value unit	Cost/request만 봄

18.3 Operations·recovery · 6개

ID	Scenario	Expected evidence	대표 실패
SC-13	Log ingest·archive·retention·query	Log cost	Ingest만 셈
SC-14	Metric series·cardinality·sample	Metric cost	Label explosion
SC-15	Trace sampling·span·retention	Trace cost	Error trace 소실
SC-16	Backup copy·tier·retention	Backup cost	원본 GB만 셈
SC-17	Restore compute·egress·labor·cleanup	Drill cost	Job success만 봄
SC-18	On-call·support·region·SLO tradeoff	Reliability cost	신뢰성 삭제

18.4 License·TCO · 6개

ID	Scenario	Expected evidence	대표 실패
SC-19	Seat·active·minimum·true-up	License position	Purchased=active 가정

ID	Scenario	Expected evidence	대표 실패
SC-20	Entitlement·quota·overage·support	Entitlement register	가격만 봄
SC-21	OSS ID·notice·source·patent	Obligation register	분류로 법률 결론
SC-22	Estimate·forecast·budget·actual·invoice	Reconciliation	State 혼용
SC-23	Low·base·high·break-even·commitment	Sensitivity	한 점 견적
SC-24	Fully loaded TCO·unit cost·Gate	Decision record	누락을 0으로 처리

18.5 PASS의 의미

Scenario PASS는 expected field와 actual evidence가 같은 contract를 만족했다는 뜻입니다. 실제 가격·계약·법률·세금이 옳다는 보장이 아닙니다.

18.6 Critical 100%

Critical scenario는 하나라도 실패하면 candidate를 block합니다.

- Network direction
- Meter·rate match
- Allocation
- Model version·tool·quality
- Backup·restore·reliability
- Entitlement·OSS
- Sensitivity·TCO·residual risk

19. Cost Readiness Gate

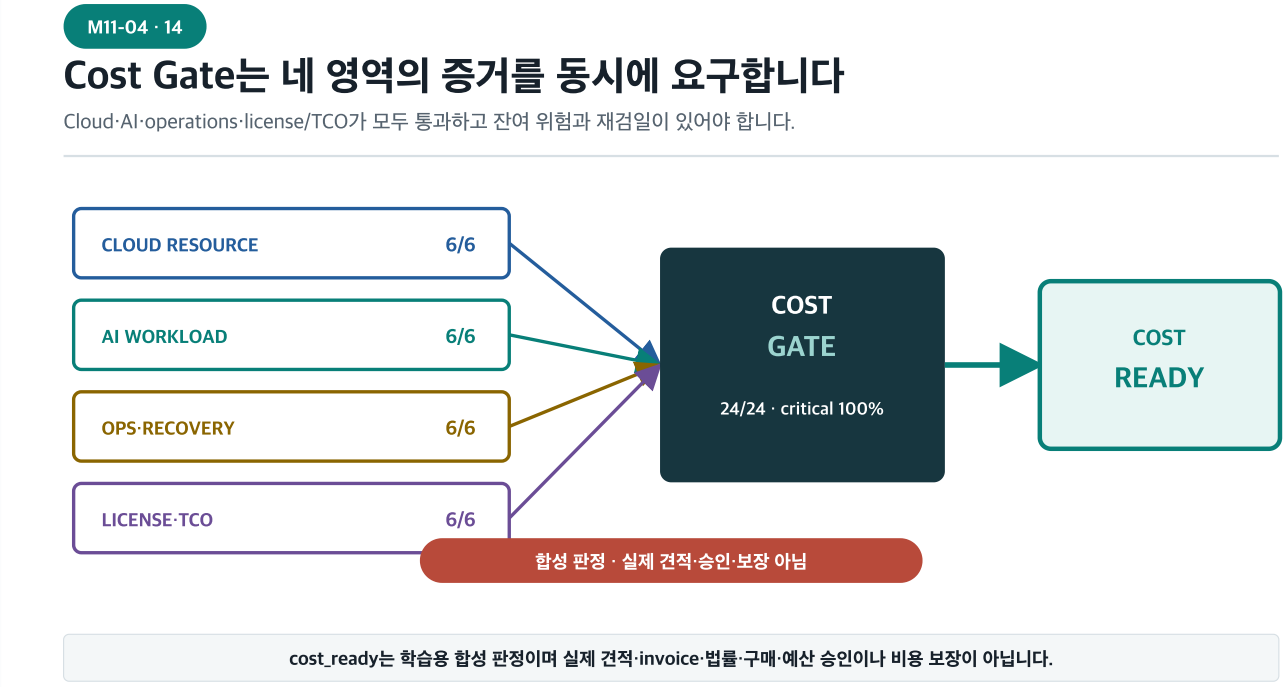
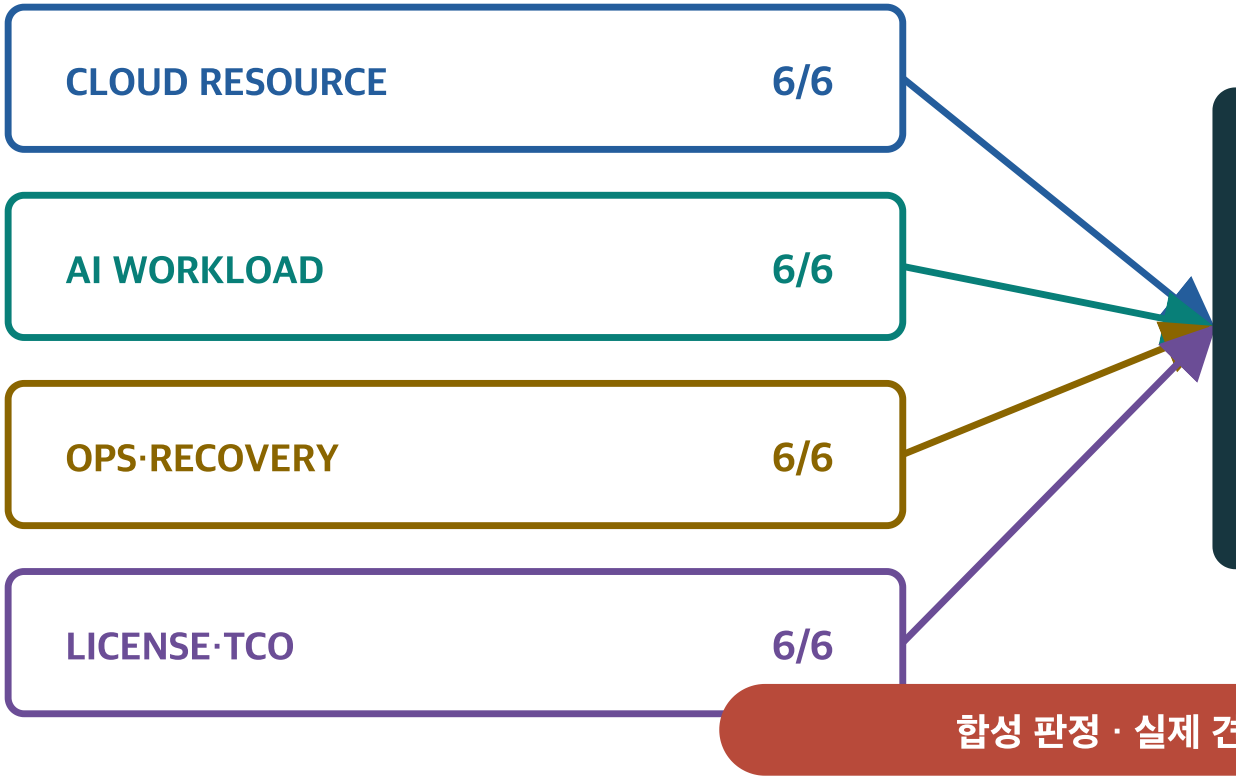
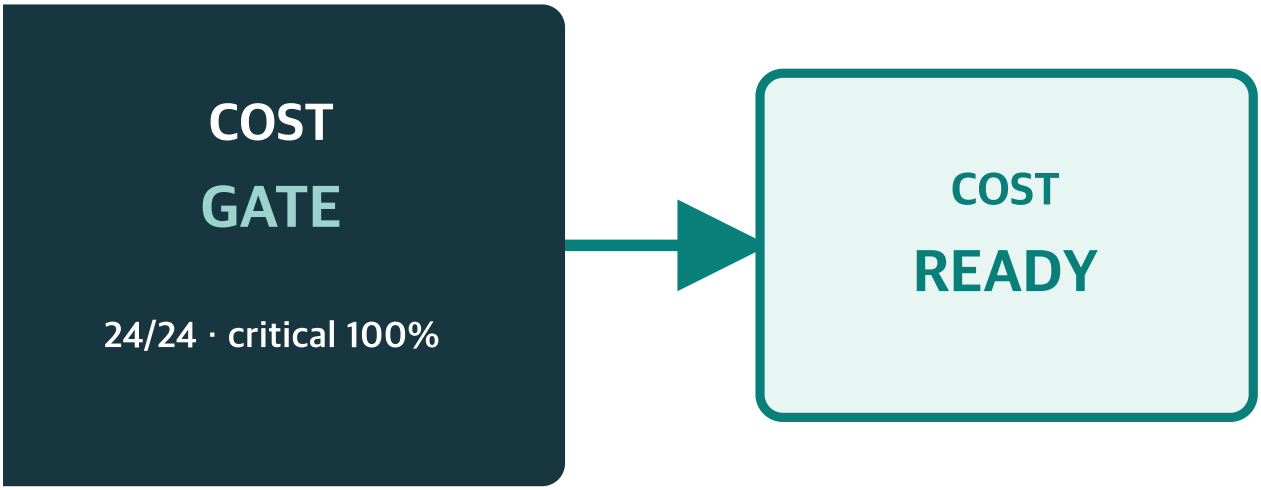


그림 16. 네 lane 6개씩, 전체·critical·control·lane coverage, versioned rate card, fully loaded TCO, license review, residual risk를 동시에 만족할 때만 합성 `cost_ready`를 반환합니다.

Cloud·AI·operations·license/TCO가 모두 통과하고 잔여 위험과 재검일이



있어야 합니다.



적·승인·보장 아님

19.1 Gate threshold

Check	Threshold
Scenario pass	24/24
Critical pass	100%
Lane coverage	4/4
Control coverage	12/12
Versioned rate card	Yes
Cost scope complete	Yes
Omitted component	0
License review complete	Yes
Fully loaded TCO	Yes
Orphan control·scenario	0
Residual risk owner	100%
Revalidation date	Present

19.2 세 decision

Decision	의미	다음 행동
`blocked_cost_blindness`	Usage·meter·rate·allocation·ops evidence 부족	Scope·usage·rate 수정
`blocked_commercial_readiness`	Entitlement·license·sensitivity·TCO 부족	Finance·procurement·legal review
`cost_ready`	합성 contract 통과	실제 공식 자료·승인 절차로 이동

19.3 Residual risk

대응	언제 쓰는가	Evidence
Accept	영향·확률·범위가 허용됨	Named approver·expiry

대응	언제 쓰는가	Evidence
Mitigate	통제로 낮출 수 있음	Action·owner·due·verification
Transfer	Contract·insurance·vendor로 일부 이전	Exact term·limit
Avoid	Risk를 만드는 선택을 하지 않음	Architecture·scope decision

19.4 Cost Gate가 승인하지 않는 것

- Actual provider quote
- Payable invoice
- Tax calculation
- Legal opinion
- OSS clearance
- Commercial compliance
- Procurement approval
- Budget approval
- Production deployment
- Cost guarantee

---

20. 실제 service에 적용하는 순서

20.1 Crawl · 보이게 만들기

1. Scope·owner·as-of를 정합니다.
2. Top 10 cost category를 찾습니다.
3. Usage와 rate unit을 맞춥니다.
4. Required tags와 unallocated cost를 봅니다.
5. Estimate·actual·invoice를 분리합니다.
6. Seat·entitlement·renewal을 inventory합니다.

20.2 Walk · 가치에 연결하기

1. Shared cost rule을 version화합니다.
2. AI token·tool·quality를 연결합니다.
3. Telemetry·recovery cost를 TCO에 넣습니다.
4. Business unit과 cost per success를 계산합니다.
5. Low·base·high forecast를 운영합니다.
6. Budget·anomaly·variance owner를 둡니다.

20.3 Run · 의사결정을 자동화하기

1. FOCUS-compatible data를 검토합니다.
2. Contract·license·SaaS 자료와 billing을 연결합니다.
3. Rate·usage·allocation freshness를 감시합니다.
4. Cost·quality·reliability guardrail을 delivery에 연결합니다.
5. Renewal 전에 optimization과 forecast를 완료합니다.
6. TCO·unit economics로 product portfolio를 비교합니다.

20.4 공식 자료 적용 checklist

- [ ] Provider pricing page와 calculator as-of가 같다.
- [ ] Organization contract·private offer가 반영됐다.
- [ ] Region·tier·meter·rounding·free allowance가 맞다.
- [ ] Pricing·billing·reporting currency가 구분됐다.
- [ ] FX·tax source와 권한자가 있다.
- [ ] Actual·invoice reconciliation이 있다.
- [ ] License exact text와 contract를 법무·구매가 검토했다.
- [ ] Model·rate·policy 변경 시 revalidation한다.

---

21. 흔한 실패와 교정

실패 1 · Current price를 영구 상수로 저장

증상: Source·as-of 없이 rate를 code에 고정합니다.

교정: Versioned rate card, as-of, source snapshot, revalidation date를 둡니다.

실패 2 · 월 730시간을 모든 사용량에 적용

증상: Autoscaling·serverless·stop schedule을 무시합니다.

교정: 실제 runtime distribution과 provisioned·consumed·idle을 나눕니다. 730은 calculator convention일 수 있을 뿐 실제 usage가 아닙니다.

실패 3 · Storage peak를 GB-month로 사용

증상: 평균 용량과 request·retrieval을 무시합니다.

교정: Time-weighted average·tier·operation·retention을 계산합니다.

실패 4 · Network 전체 GB 하나

증상: Source·destination·region·direction이 없습니다.

교정: Flow ledger를 만듭니다.

실패 5 · AI request 수만 셈

증상: Token length·cache·output·tool·retry drift를 놓칩니다.

교정: Model version별 workload profile과 cost/success를 둡니다.

실패 6 · Cheapest model 선택

증상: Quality·failure·human review cost가 늘어납니다.

교정: Quality floor·latency·success denominator와 frontier를 비교합니다.

실패 7 · Log만 줄임

증상: Metric cardinality·query·retention·support가 더 클 수 있습니다.

교정: Signal별 cost driver와 incident·SLO value를 연결합니다.

실패 8 · Backup GB만 계산

증상: Restore·validation·labor·RPO·RTO가 사라집니다.

교정: Recovery TCO와 drill evidence를 포함합니다.

실패 9 · Purchased seat=active user

증상: Shelfware·overage·minimum·true-up을 놓칩니다.

교정: License position과 renewal lead time을 관리합니다.

실패 10 · "MIT라서 괜찮다"

증상: Exact version·text·notice·patent·distribution 사실관계를 생략합니다.

교정: SPDX ID는 식별에 쓰고 법률 결론은 exact text와 사용 방식으로 법무가 검토합니다.

실패 11 · Budget=forecast

증상: 승인 한도를 실제 예상처럼 보고 variance를 늦게 발견합니다.

교정: 다섯 cost state와 owner를 분리합니다.

실패 12 · Unknown=0

증상: TCO가 낮아 보이지만 불확실성이 숨습니다.

교정: Assumption·range·owner·due·reforecast trigger로 남깁니다.

---

22. M12-01로 넘기는 handoff

다음 매뉴얼은 **사용자 문제와 기대 결과 정의하기**입니다. M11-04는 solution을 확정하지 않고 경제적 제약과 uncertainty를 넘깁니다.

Handoff	제품 발견 질문
Business unit	사용자가 실제로 얻는 결과는 무엇인가
Current cost/unit	지금 문제를 해결하는 비용은
Target cost/unit	지속 가능한 범위는
Quality floor	싸지만 실패하는 결과를 어떻게 제외할까
Latency objective	사용자가 기다릴 수 있는 시간은
Budget range	Low·base·high 실험 범위는

Handoff	제품 발견 질문
Break-even	어느 사용량에서 architecture가 바뀌나
Commercial constraint	Minimum·term·seat가 무엇을 제한하나
License constraint	Distribution·network·notice·source 조건은
Reliability floor	줄이면 안 되는 운영 evidence는
Largest uncertainty	어떤 사용자 행동을 먼저 검증할까

좋은 handoff:

```
성공 결과당 목표 비용은 ___이고 quality floor는 ___이다.
월 사용량 ___에서 architecture break-even이 생긴다.
가장 큰 uncertainty는 사용자의 ___ 행동이므로 다음 discovery에서 검증한다.
```

---

23. 셀프 테스트 30

질문 1

Price와 rate의 차이는 무엇입니까?

질문 2

Usage와 rate를 곱하기 전에 반드시 확인할 것은 무엇입니까?

질문 3

As-of가 없는 rate가 위험한 이유는 무엇입니까?

질문 4

List cost·contracted cost·effective cost·billed cost를 구분하십시오.

질문 5

Compute 비용에서 provisioned·consumed·idle을 나누는 이유는 무엇입니까?

질문 6

Storage 500GB라는 정보만으로 월 비용을 계산할 수 없는 이유를 세 가지 쓰십시오.

질문 7

Network cost에서 source·destination·direction이 필요한 이유는 무엇입니까?

질문 8

Commitment coverage와 utilization의 차이는 무엇입니까?

질문 9

Tiered와 graduated pricing을 혼동하면 어떤 오류가 생깁니까?

질문 10

AI request 수가 같아도 비용이 달라지는 driver를 다섯 가지 쓰십시오.

질문 11

Cached input과 uncached input을 분리하는 이유는 무엇입니까?

질문 12

RAG 비용에서 embedding 외에 계산할 항목은 무엇입니까?

질문 13

Agent iteration cap이 비용과 품질에 주는 tradeoff는 무엇입니까?

질문 14

Cost/request보다 cost/success가 나은 경우는 언제입니까?

질문 15

Cheapest model이 최적이 아닐 수 있는 이유는 무엇입니까?

질문 16

Observability cost에서 cardinality가 중요한 이유는 무엇입니까?

질문 17

Telemetry 절감 시 유지해야 할 reliability evidence는 무엇입니까?

질문 18

Backup storage 비용 외에 recovery TCO에 포함할 항목을 다섯 가지 쓰십시오.

질문 19

RPO를 짧게 하면 일반적으로 어떤 비용이 증가합니까?

질문 20

Purchased seat·assigned seat·active user가 다른 이유는 무엇입니까?

질문 21

Entitlement와 price를 함께 봐야 하는 이유는 무엇입니까?

질문 22

SPDX license identifier가 법률 판단을 대신하지 못하는 이유는 무엇입니까?

질문 23

OSS register에 exact version과 distribution 사실을 기록하는 이유는 무엇입니까?

질문 24

Estimate·forecast·budget·actual·invoice의 역할을 한 문장씩 설명하십시오.

질문 25

Shared cost allocation rule에 version과 owner가 필요한 이유는 무엇입니까?

질문 26

Resource unit과 business unit의 차이는 무엇입니까?

질문 27

Fully loaded TCO가 cloud invoice보다 큰 이유는 무엇입니까?

질문 28

Low·base·high sensitivity가 한 점 estimate보다 나은 이유는 무엇입니까?

질문 29

`cost_ready`가 실제 budget approval이 아닌 이유는 무엇입니까?

질문 30

M12-01에 비용표 전체보다 먼저 넘겨야 할 다섯 가지는 무엇입니까?

24. 모범 답안

답 1

Price는 상품이나 묶음의 표시 금액이고, rate는 vCPU-hour·GB-month·million token 같은 단위 하나에 적용하는 가격입니다.

답 2

Scope·period·source와 usage unit·rate unit이 같은지 확인해야 합니다.

답 3

가격·tier·model·region·계약 조건이 바뀌므로 언제 유효한 숫자인지 재현하고 재검할 수 없기 때문입니다.

답 4

List는 공개 할인 전, contracted는 계약 unit price, effective는 할인·선불 상각 반영, billed는 payable invoice와 맞는 최종 charge입니다.

답 5

확보한 용량과 실제 가치 생산 사용량을 구분해 idle cost와 autoscaling 기회를 찾기 위해서입니다.

답 6

평균인지 peak인지, storage class·tier가 무엇인지, read·write·retrieval·replication·retention 조건이 무엇인지 모르기 때문입니다.

답 7

Internet·inter-region·intra-region·cross-zone과 processing path에 따라 rate가 다르기 때문입니다.

답 8

Coverage는 eligible usage 중 할인이 적용된 비율이고, utilization은 구매한 commitment가 실제로 소진된 비율입니다.

답 9

전체 사용량에 하나의 rate를 적용할지 각 구간 rate를 합산할지 달라져 total cost가 잘못됩니다.

답 10

Input length, cached share, output length, exact model·context band, tool calls, retries, fallback, multimodal usage 등이 있습니다.

답 11

Cache 조건을 만족한 input은 다른 rate가 적용될 수 있고, cache eligibility·staleness·privacy도 별도 통제가 필요하기 때문입니다.

답 12

Chunking, vector dimension, storage·index, reindex, query embedding, retrieval, reranking, generation context를 계산합니다.

답 13

Cap은 무한 tool loop와 비용 폭주를 막지만 너무 낮으면 필요한 작업 전에 종료돼 success rate가 낮아질 수 있습니다.

답 14

API response는 반환됐지만 quality·policy·latency를 통과하지 못하는 실패가 의미 있을 때 cost/success가 더 적절합니다.

답 15

낮은 quality가 retry·fallback·human review와 사용자 실패를 늘려 fully loaded cost per success를 높일 수 있습니다.

답 16

Metric label의 고유 조합마다 series가 생겨 작은 label 변화가 series-day와 storage·query 비용을 크게 늘릴 수 있습니다.

답 17

사용자 중심 SLI, critical journey, security·audit, capacity, error·slow trace, pipeline health·query canary를 유지합니다.

답 18

Restore read, compute, egress, temporary environment, engineer labor, integrity·business validation, cleanup, failed drill rework가 있습니다.

답 19

Backup·log shipping·replication 빈도와 network·storage·write processing 비용이 일반적으로 증가합니다.

답 20

구매 수량, 계정 할당, 기간 중 실제 활동 조건이 서로 다르고 minimum·shared account·concurrency 조건도 있기 때문입니다.

답 21

같은 가격이라도 허용 feature·version·quota·support·BYOL·restriction이 달라 실제 가치와 compliance risk가 다르기 때문입니다.

답 22

Identifier는 표준 식별 도구일 뿐 exact text·version·수정·결합·배포·network 제공 사실에 대한 법률 해석을 하지 않기 때문입니다.

답 23

License와 의무 trigger가 component version과 실제 사용·배포 방식에 따라 달라질 수 있기 때문입니다.

답 24

Estimate는 설계 가정, forecast는 최신 미래 예상, budget은 승인 한도, actual은 발생 비용, invoice는 지급 근거입니다.

답 25

누가 어떤 driver로 비용을 나눴는지 반복 계산하고 변경·분쟁·감사 때 설명하기 위해서입니다.

답 26

Resource unit은 token·GB·vCPU-hour 같은 기술 효율 분모이고 business unit은 user·transaction·successful outcome 같은 가치 분모입니다.

답 27

Operations·recovery·license·support·shared platform·implementation·migration·people·governance·risk layer가 invoice 밖에 있을 수 있기 때문입니다.

답 28

불확실성 범위와 결과를 지배하는 assumption, budget risk, architecture break-even을 보여 주기 때문입니다.

답 29

`cost_ready`는 합성 contract의 scenario·coverage·evidence 통과일 뿐 actual quote·tax·legal·procurement·finance 권한자의 승인이 아니기 때문입니다.

답 30

Business unit, target cost/unit, quality·latency floor, low·base·high budget, break-even, commercial·license·reliability constraint, largest uncertainty 중 핵심 다섯 가지를 넘깁니다.

---

25. 최종 제출 Checklist

- [ ] Scope·owner·period·currency·as-of가 있다.
- [ ] 17개 이상의 실제 적용 rate source를 register로 만들었다.
- [ ] Usage와 rate unit을 맞췄다.
- [ ] Compute·storage·network flow를 계산했다.
- [ ] AI input·cache·output·embedding·tool·retry를 계산했다.
- [ ] Quality·latency·cost per success를 연결했다.
- [ ] Telemetry·retention·cardinality를 계산했다.
- [ ] Backup·restore·labor·reliability를 포함했다.
- [ ] Commercial entitlement·renewal을 확인했다.
- [ ] OSS dependency·exact text·obligation을 법무에 넘겼다.
- [ ] Shared cost·credit·discount·tax를 분리했다.
- [ ] 다섯 cost state와 variance owner가 있다.
- [ ] Low·base·high·break-even이 있다.
- [ ] Fully loaded 12-month TCO가 있다.
- [ ] 24 scenario·12 control·4 lane이 연결됐다.
- [ ] Residual risk·owner·due·revalidation이 있다.
- [ ] 실제 quote·invoice·tax·legal·procurement·budget 승인 경계를 적었다.
- [ ] M12-01 handoff가 있다.

26. 요약

```
scope
→ usage·meter·unit
→ rate·region·tier·contract·as-of
→ allocation·credit·discount·tax
→ cloud·AI·operations·recovery·license
→ estimate·forecast·budget·actual·invoice
→ unit economics·TCO·sensitivity
→ 24 scenario·Cost Gate
→ product discovery handoff
```

핵심은 다음 한 문장입니다.

> **비용을 안다는 것은 총액을 기억하는 일이 아니라, 숫자의 경계·단위·근거·의무·가치·불확실성을 다시 계산하고 책임질 수 있다는 뜻입니다.**

27. 공식 참고 자료

FinOps·billing data

- FinOps Framework
- FinOps Unit Economics
- FinOps Forecasting
- FinOps Planning & Estimating
- FinOps Reporting & Analytics
- FinOps Licensing & SaaS
- FinOps Governance, Policy & Risk
- FOCUS Specification v1.4

Cloud pricing·cost management

- AWS Pricing
- AWS Pricing Calculator
- AWS Pricing Calculator documentation
- AWS pricing key principles
- Azure Cost Management overview
- Azure cost allocation
- Google Cloud Billing reports
- Google Cloud cost report analysis

AI billing·pricing dimensions

- OpenAI model comparison
- OpenAI Prompt Caching
- OpenAI Batch
- Gemini API billing
- Gemini API pricing

License identification

- SPDX License List
- OSI Approved Licenses

> 공식 자료도 계속 바뀝니다. 실제 적용 직전에 선택한 provider·service·model·region·plan·contract·license의 최신 문서를 같은 as-of로 다시 확인합니다.