

AWS SAA-C03 통합 가이드 - Part 4: 비용 최적화 (20%)

비용 절감 전략 - 아키텍처 설계의 마지막 퍼즐!

비용 최적화 핵심 원칙

비용 최적화 4대 원칙

1. 적절한 크기 조정 (Right Sizing): 과도한 프로비저닝 방지
2. 탄력성 (Elasticity): 사용하지 않을 때 리소스 축소/중지
3. 구매 옵션 최적화: 예약/Spot으로 할인 극대화
4. 데이터 전송 최소화: 리전 간, 인터넷 아웃바운드 비용 감소

비용 최적화 우선순위



1. 유휴/미사용 리소스 제거 (즉시 효과)
2. Reserved/Savings Plans (장기 할인)
3. Spot Instances (단기 워크로드)
4. 스토리지 계층화 (S3 Lifecycle)
5. 네트워크 최적화 (VPC Endpoint)

EC2 비용 전략

WHY

- 컴퓨팅은 AWS 비용의 **최대 축** (평균 40-60%)
- 워크로드 특성에 맞는 구매 방식으로 **최대 90% 절감 가능**

HOW

구매 옵션 비교

| 옵 셴 | 할 인 율 | 약 정 기 간 | 유 연 성 | 중 단 | 사 용 사 례 |
|-------------------------|-------|---------|-------|-----------|-----------------|
| On-Demand | 0% | 없음 | 최 고 | ✗ | 단기, 테스트, 스파이크 |
| Spot | ~90% | 없음 | 낮 음 | ✓ (2분 알림) | 배치, 빅데이터, CI/CD |
| Reserved (Standard) | ~75% | 1-3년 | 낮 음 | ✗ | 예측 가능 24/7 |
| Reserved (Convertible) | ~54% | 1-3년 | 중 간 | ✗ | 유연성 필요 시 |
| Savings Plans (Compute) | ~66% | 1-3년 | 최 고 | ✗ | 가장 권장 |
| Savings Plans (EC2) | ~72% | 1-3년 | 중 간 | ✗ | 패밀리 고정 가능 시 |
| Dedicated Hosts | 0% | 시간당/예약 | - | ✗ | BYOL (라이선스) |
| Dedicated Instances | 0% | 시간당 | - | ✗ | 규정 준수 |

Reserved Instances 상세

Standard RI

- 인스턴스 유형, AZ, OS 고정
- 최대 할인 (72-75%)
- 변경 불가
- 판매 가능 (Marketplace)

Convertible RI

- 인스턴스 패밀리 변경 가능
- 중간 할인 (54-66%)
- 유연성 높음
- 판매 불가

RI 범위

- Regional:** 리전 전체, AZ 유연성, 용량 예약 ✗
- Zonal:** 특정 AZ, 용량 예약 ✓

RI 수정 가능 항목

- Availability Zone
- 인스턴스 크기 (같은 패밀리 내)
- 네트워킹 유형

Savings Plans 상세

Compute Savings Plans

- EC2, Fargate, Lambda 모두 적용
- 리전, 인스턴스 패밀리, OS, Tenancy 자유
- 시간당 사용량 약정 (예: \$10/시간)
- 가장 유연, 권장

EC2 Instance Savings Plans

- 특정 인스턴스 패밀리 + 리전

- 크기, OS, Tenancy 자유
- 더 높은 할인 (Compute보다)

Savings Plans 특징

- 자동 적용 (가장 절감 효과 큰 것 우선)
- RI와 병행 가능
- Cost Explorer로 권장 확인

Spot Instances

작동 방식



1. Spot 요청 제출 (최대 가격 지정)
2. 현재 Spot 가격이 최대 가격 이하면 시작
3. Spot 가격 상승 또는 용량 부족 시:
 - 2분 전 중단 알림 (CloudWatch Events)
 - 인스턴스 중단 (Stop, Terminate, Hibernate)

Spot 가격

- 실시간 변동 (수요/공급)
- 일반적으로 On-Demand의 10-20%
- 리전, AZ, 인스턴스 유형별 다름

Spot Fleet

- 여러 인스턴스 유형 조합
- 목표 용량 지정
- 할당 전략:
 - **lowestPrice**: 최저 가격 (비용 최적화)
 - **diversified**: 여러 풀 분산 (가용성)
 - **capacityOptimized**: 용량 최적화 (중단 최소화, 권장)
 - **priceCapacityOptimized**: 가격+용량 균형

Spot 사용 사례

- 배치 처리
- 빅데이터 분석 (EMR, Spark)
- CI/CD 빌드
- 컨테이너 워크로드
- 웹 서버 (ASG MixedInstancesPolicy)
- 데이터베이스
- 중단 불가능한 작업

Spot Instance Interruption

- CloudWatch Events로 2분 전 알림
- Lambda로 자동 처리 (백업, 재시작)

ASG Mixed Instances Policy

On-Demand + Spot 혼합



```
json
{
  "OnDemandBaseCapacity": 2,
  "OnDemandPercentageAboveBaseCapacity": 30,
  "SpotAllocationStrategy": "capacity-optimized"
}
```

- Base: 최소 On-Demand 인스턴스 수
- Percentage: Base 초과분의 On-Demand 비율
- 나머지: Spot

예시



Desired Capacity: 10
Base: 2 (On-Demand)
Percentage: 30%
→ On-Demand: 2 (Base) + 2.4 (30% of 8) = 4-5개
→ Spot: 5-6개

비용 절감 전략

1. 인스턴스 스케줄링

- AWS Instance Scheduler
- EventBridge + Lambda
- 비업무 시간 중지 (예: 주말, 야간)
- 개발/테스트 환경 적용

2. Right Sizing

- Compute Optimizer 권장 확인

- CloudWatch 메트릭 분석 (CPU, 메모리)
- 과도한 프로비저닝 방지
- 작은 인스턴스로 시작 → 확장

3. 세대 업그레이드

- 최신 세대가 저렴 + 고성능
- 예: t3 > t2, c6i > c5

4. ARM 기반 (Graviton)

- Graviton3: 최대 40% 저렴 + 60% 고성능
- Graviton2: 최대 20% 저렴
- 지원: Amazon Linux 2, Ubuntu, etc.

✚ SCENARIO

| 상 황 | 정 답 | 이 유 |
|----------------------|------------------------------|-------------|
| 예측 가능한 24/7 웹 서비스 | Compute Savings Plans | 최고 유연성 + 할인 |
| 배치 처리, 중단 허용 | Spot Instances | 최대 90% 절감 |
| 인스턴스 종류 변경 가능성 | Convertible RI 또는 Compute SP | 유연성 |
| 특정 소프트웨어 라이선스 (BYOL) | Dedicated Hosts | 물리 서버 고정 |
| 단기 프로젝트 (3개월) | On-Demand | 약정 없음 |
| 웹 서버 (고가용성 + 비용 절감) | ASG Mixed (On-Demand + Spot) | 안정성 + 절감 |
| 개발 환경 (주말 사용 안 함) | 인스턴스 스케줄링 | 비업무 시간 중지 |

🎯 시험 키워드 매칭

- "예측 가능" → Savings Plans / RI
- "중단 허용" → Spot
- "유연성" → Compute Savings Plans
- "최대 할인" → Spot (90%), Standard RI (75%)
- "라이선스" → Dedicated Hosts

🚫 TRAP

- ❌ Spot을 DB로 사용 (중단 불가)
- ❌ Savings Plans = RI (Savings Plans가 더 유연)
- ❌ RI 구매 후 즉시 변경 가능 (Standard는 제한)
- ❌ Spot은 항상 중단 (가격/용량에 따라 다름)

📁 S3 비용 절감

💡 WHY

- 데이터 규모가 커질수록 스토리지 비용 지배적
- 접근 패턴 기반 티어링으로 최대 95% 절감

Storage Class 비교

| Class | 접근 빈도 | 검색 시간 | 최소 보관 | 최소 객체 크기 | GB당 비용 / 월 | 사용 사례 |
|----------------------|-------|---------|-------|----------|----------------|--------------|
| Standard | 자주 | 즉시 | 없음 | 없음 | \$0.023 | 핫 데이터 |
| Intelligent-Tiering | 모름 | 즉시 | 없음 | 128KB | \$0.023 + 모니터링 | 예측 불가 |
| Standard-IA | 드물게 | 즉시 | 30일 | 128KB | \$0.0125 | 백업 |
| One Zone-IA | 드물게 | 즉시 | 30일 | 128KB | \$0.01 | 재생성 가능 |
| Glacier Instant | 분기별 | 밀리초 | 90일 | 128KB | \$0.004 | 아카이브 (즉시 필요) |
| Glacier Flexible | 년 단위 | 분~12시간 | 90일 | 없음 | \$0.0036 | 아카이브 |
| Glacier Deep Archive | 7년+ | 12~48시간 | 180일 | 없음 | \$0.00099 | 장기 보관 |

비용 구조

Storage Class별 비용 요소



- Standard:
- 스토리지: \$0.023/GB
 - GET/PUT: \$0.0004/1,000 (무시 가능)

- Standard-IA:
- 스토리지: \$0.0125/GB (46% 절감)
 - GET: \$0.001/1,000 (2.5배)
 - 검색: \$0.01/GB (추가!)

- Glacier Deep Archive:
- 스토리지: \$0.00099/GB (96% 절감)
 - 검색: \$0.02/GB
 - 복원 시간: 12시간 (Standard)

주의사항

- IA/Glacier는 검색 비용 발생
- 자주 읽으면 오히려 비용 증가
- 최소 보관 기간 미달 시 조기 삭제 비용

Intelligent-Tiering

자동 계층 이동



Frequent Access (30일)

↓ (30일 미접근)

Infrequent Access (90일)

↓ (90일 미접근)

Archive Instant Access (180일)

↓ (180일 미접근)

Archive Access (선택적)

↓

Deep Archive Access (선택적)

비용

- 스토리지: 계층별 비용
- 모니터링: \$0.0025/1,000 objects
- 검색: 없음 (Frequent Access만)

사용 시기

- 접근 패턴 불명확
- 데이터 수명 주기 변동
- 자동화 선호

Lifecycle Rules

Rule 구성



json

```
{
  "Rules": [
    {
      "Id": "Archive old data",
      "Status": "Enabled",
      "Transitions": [
        {
          "Days": 30,
          "StorageClass": "STANDARD_IA"
        },
        {
          "Days": 90,
          "StorageClass": "GLACIER_IR"
        },
        {
          "Days": 365,
          "StorageClass": "DEEP_ARCHIVE"
        }
      ],
      "Expiration": {
        "Days": 2555
      }
    }
  ]
}
```

Versioning + Lifecycle



json

```
{
  "NoncurrentVersionTransitions": [
    {
      "NoncurrentDays": 30,
      "StorageClass": "GLACIER"
    }
  ],
  "NoncurrentVersionExpiration": {
    "NoncurrentDays": 90
  }
}
```

S3 비용 최적화 팁

1. 객체 크기 최적화

- 128KB 미만은 Standard 사용 (IA 최소 크기)
- 멀티파트 업로드 (>100MB)

2. Incomplete Multipart 정리

- Lifecycle Rule로 자동 삭제
- 7일 후 정리 권장

3. Requester Pays

- 요청자에게 비용 전가
- 대용량 데이터 공유 시

4. S3 Select / Glacier Select

- SQL로 필터링
- 네트워크 전송 감소 (최대 80%)
- 비용 절감

5. S3 Batch Operations

- 대량 객체 일괄 처리
- 복사, 태그, Lifecycle 등

SCENARIO

| 상황 | 정답 | 이유 |
|-------------------|---------------------------------|----------------|
| 30일 자주 접근, 그 후 보관 | Standard → IA → Glacier | Lifecycle Rule |
| 접근 패턴 불명확 | Intelligent-Tiering | 자동 최적화 |
| 7년 규정 준수 보관 | Glacier Deep Archive | 최저 비용 (96% 절감) |
| 즉시 복원 필요한 아카이브 | Glacier Instant Retrieval | 밀리초 복원 |
| 재생성 가능한 데이터 | One Zone-IA | 단일 AZ, 저렴 |
| 로그 (30일 후 삭제) | Standard + Lifecycle Expiration | 자동 삭제 |

시험 키워드 매칭

- "접근 패턴 모름" → Intelligent-Tiering
- "장기 보관" → Glacier Deep Archive
- "즉시 복원" → Glacier Instant
- "비용 최소화" → Lifecycle + Deep Archive

TRAP

- ✗ Standard-IA에 자주 읽으면 비용 증가 (검색 비용)
- ✗ Intelligent-Tiering은 완전 무료 (모니터링 비용 존재)
- ✗ Glacier는 즉시 복원 (Instant만 가능, Flexible은 시간 소요)
- ✗ 128KB 미만 객체는 IA 비용 효율 낮음

Serverless 비용 고려

WHY

- 사용량 기반 과금 = **유휴 비용 0**
- 하지만 호출 수·실행 시간·메모리가 비용 결정
- 잘못 설계 시 EC2보다 비쌀 수 있음

HOW

Lambda 비용 구조

비용 요소

1. **요청 비용**: \$0.20 / 100만 요청
2. **실행 비용**: GB-초당 \$0.0000166667
3. **Provisioned Concurrency**: ACU 시간당 (선택적)

계산 예시



시나리오:

- 메모리: 512MB = 0.5GB
- 실행 시간: 200ms = 0.2초
- 요청 수: 100만/월

비용:

- 요청: 100만 × \$0.20 = \$0.20
- 실행: 100만 × 0.5GB × 0.2초 × \$0.0000166667 = \$1.67
- 총: \$1.87/월

비용 최적화

- 메모리 증가 = CPU 증가 = 실행 시간 감소
- 최적 메모리 찾기 (AWS Lambda Power Tuning)
- 불필요한 라이브러리 제거
- 연결 재사용 (DB, HTTP)

Lambda vs EC2 비교



Lambda (간헐적):

- 1시간/일 실행 = \$1-2/월
- 유휴 비용 0

EC2 t3.micro (24/7):

- On-Demand = \$7.5/월
- Reserved (1년) = \$4.5/월

결론: 50% 이하 사용 시 Lambda 유리

API Gateway 비용

| 유형 | 요청 비용 | 캐싱 | 데이터 전송 |
|-----------|----------------------|----|-----------|
| REST API | \$3.50/100만 | ✓ | \$0.09/GB |
| HTTP API | \$1.00/100만 (71% 절감) | ✗ | \$0.09/GB |
| WebSocket | \$1.00/100만 + 연결 시간 | ✗ | \$0.09/GB |

비용 절감

- HTTP API 우선 사용 (REST 기능 불필요 시)
- 캐싱으로 백엔드 호출 감소
- Throttling으로 과도한 요청 차단

Fargate 비용

비용 구조

- vCPU: \$0.04048/시간
- 메모리: \$0.004445/GB-시간

계산 예시



0.5 vCPU, 1GB 메모리, 24시간 실행:

- vCPU: $0.5 \times \$0.04048 \times 24 = \0.49
- 메모리: $1 \times \$0.004445 \times 24 = \0.11
- 총: \$0.60/일 = \$18/월

Fargate vs EC2



Fargate (작은 컨테이너):

- 0.5 vCPU, 1GB = \$18/월

EC2 t3.micro (1 vCPU, 1GB):

- On-Demand = \$7.5/월
- Reserved = \$4.5/월

결론:

- 항상 실행 = EC2 유리
- 간헐적 = Fargate 유리
- 관리 오버헤드 = Fargate 유리

DynamoDB 비용

On-Demand vs Provisioned

| 항목 | On-Demand | Provisioned + Auto Scaling |
|-------|-------------|------------------------------|
| WCU | \$1.25/100만 | \$0.00065/시간 (약 \$0.47/100만) |
| RCU | \$0.25/100만 | \$0.00013/시간 (약 \$0.09/100만) |
| 예측 | 불필요 | 필요 |
| 사용 사례 | 예측 불가 | 예측 가능 (62% 절감) |

비용 최적화

- 예측 가능하면 Provisioned + Auto Scaling

- TTL로 자동 삭제 (무료)
- GSI 최소화 (별도 RCU/WCU)
- Batch Operations 사용

SCENARIO

| 상 황 | 정 답 | 이 유 |
|---------------|-------------------------------------|-------------|
| 하루 몇 번 실행 | Lambda | 유 휴 비 용 0 |
| HTTP API, 저비용 | API Gateway HTTP API | 71% 절 감 |
| 24/7 컨테이너 | EC2 (Reserved) 또는 Fargate Spot | 비 용 비 교 필 요 |
| 예 측 가 능 한 | DynamoDB Provisioned + Auto Scaling | 62% 절 감 |
| 예 측 불 가 | DynamoDB On-Demand | 유 연 성 |

시험 키워드 매칭

- "간헐적" → Lambda, Fargate
- "저비용 API" → HTTP API
- "예측 가능" → Provisioned
- "예측 불가" → On-Demand

TRAP

- ❌ Lambda가 항상 저렴 (고동시성 장시간은 비쌀 수 있음)
- ❌ API Gateway REST 무조건 사용 (HTTP API로 충분)
- ❌ Provisioned Concurrency는 무료 (추가 비용 발생)

비용 관리 도구

WHY

- 모니터링 없으면 **비용 폭증** 위험
- 권장사항으로 절감 포인트 발견
- 예산 초과 사전 방지

HOW

Cost Explorer

기능

- 비용 추세 분석 (일/주/월)
- 서비스별, 리전별, 태그별 분석
- 예측 (최대 12개월)
- 필터링, 그룹화

권장 리포트

- 월별 비용
- 서비스별 비용
- 태그별 비용 (프로젝트/팀/환경)
- Reserved Instance 사용률
- Savings Plans 사용률

Savings Plans 권장

- 과거 사용 패턴 분석
- 최적 약정 금액 제시
- 예상 절감액 계산

AWS Budgets

예산 유형

- **Cost Budget:** 비용 기반
- **Usage Budget:** 사용량 기반 (예: GB, 시간)
- **RI Utilization:** RI 사용률
- **RI Coverage:** RI 커버리지

알림 설정

- 실제 비용 (Actual)
- 예측 비용 (Forecasted)
- 임계값 (예: 80%, 100%, 120%)
- SNS, Email, Chatbot

예산 액션

- IAM Policy 적용 (권한 제한)
- SCP 적용 (조직 차원)
- EC2/RDS 중지 (Lambda 연동)

Trusted Advisor

5가지 카테고리

- 1. **비용 최적화**
 - 유휴 RDS
 - 미연결 EIP
 - 낮은 활용도 EC2
 - 최적화되지 않은 EBS
 - RI 권장
- 2. **성능**
 - 높은 활용도 EC2
 - CloudFront 최적화

- EBS Provisioned IOPS

3. 보안

- 열린 보안 그룹
- MFA 비활성화
- S3 퍼블릭 액세스

4. 내결함성

- EBS 스냅샷
- RDS 백업
- Multi-AZ

5. 서비스 제한

- VPC 제한
- EIP 제한

지원 플랜별 기능

- **Basic/Developer:** 7개 핵심 체크 (무료)
- **Business/Enterprise:** 모든 체크 (115+)
- **Business/Enterprise:** CloudWatch Events 통합

Cost Anomaly Detection

기능

- AI/ML 기반 이상 비용 탐지
- 서비스, 계정, 태그별 모니터링
- 이상 패턴 자동 감지
- SNS 알림

사용 사례

- 갑작스런 비용 증가 탐지
- 실수로 리소스 생성
- 공격 또는 오용

Cost Allocation Tags

태그 전략



필수 태그:

- Project: project-a, project-b
- Environment: dev, staging, prod
- Team: engineering, marketing
- CostCenter: cc-1234

활성화

1. 리소스에 태그 부여
2. Cost Allocation Tags 활성화 (Billing Console)
3. 24시간 후 Cost Explorer에서 분석

태그 정책 (Organizations)

- 조직 전체 태그 표준화
- 필수 태그 강제

Cost and Usage Report (CUR)

기능

- 가장 상세한 비용 데이터
- 시간별 사용량
- S3 저장 (Parquet, CSV)
- Athena, QuickSight로 분석

사용 사례

- 커스텀 대시보드
- 부서별 Chargeback
- 세밀한 비용 분석

 SCENARIO

| 상황 | 정답 | 이유 |
|-------------|--------------------------------------|-------------|
| 월 비용 급증 탐지 | Cost Anomaly Detection | AI 기반 자동 탐지 |
| 프로젝트별 비용 할당 | Cost Allocation Tags + Cost Explorer | 태그 기반 분석 |
| 예산 초과 알림 | AWS Budgets | 임계값 알림 |
| 비용 절감 권장 | Trusted Advisor | RI/유휴 리소스 |
| 유휴 리소스 탐지 | Trusted Advisor | 자동 스캔 |
| 세밀한 비용 분석 | Cost and Usage Report + Athena | 시간별 데이터 |

 시험 키워드 매칭

- "비용 분석" → Cost Explorer
- "예산 알림" → Budgets
- "절감 권장" → Trusted Advisor

- "이상 탐지" → Cost Anomaly Detection
- "태그 기반" → Cost Allocation Tags

TRAP

- ✖

 Budgets만으로 자동 차단 (알림만, 액션은 추가 설정)
- ✖

 태그 없이 세부 분석 가능 (태그 필수)
- ✖

 Trusted Advisor는 모든 플랜에서 전체 기능 (Business+ 필요)

데이터 전송 & 네트워크 비용

WHY

- 데이터 전송은 숨겨진 비용
- 리전 간, 인터넷 아웃바운드 비용 누적

HOW

데이터 전송 비용 구조

| 경로 | 비용 |
|----------------------|--------------------|
| 인터넷 → AWS (Inbound) | 무료 |
| AWS → 인터넷 (Outbound) | \$0.09/GB (첫 10TB) |
| 리전 간 전송 | \$0.02/GB |
| AZ 간 전송 (Public IP) | \$0.01/GB |
| AZ 간 전송 (Private IP) | \$0.01/GB |
| 동일 AZ (Private IP) | 무료 |
| S3 → CloudFront | 무료 |
| CloudFront → 인터넷 | \$0.085/GB (더 저렴) |

비용 절감 전략

1. VPC Endpoint 사용



나쁜 예:
Private Subnet → NAT Gateway (\$0.045/시간 + \$0.045/GB) → S3

좋은 예:
Private Subnet → Gateway Endpoint (무료) → S3

절감액: NAT Gateway 비용 전체 제거

2. CloudFront 사용



나쁜 예:

S3 → 인터넷 (\$0.09/GB)

좋은 예:

S3 → CloudFront (무료) → 인터넷 (\$0.085/GB)

추가 효과:

- 캐싱으로 오리진 요청 감소
- 빠른 응답 속도

3. Direct Connect



대량 전송 (월 수백 TB):

- 인터넷: $\$0.09/\text{GB} \times 500\text{TB} = \$45,000$
- Direct Connect: 포트 비용 + $\$0.02/\text{GB} = \text{약 } \$10,000\text{-}15,000$

절감: 60-70%

4. 리전 간 전송 최소화

- 동일 리전 아키텍처 설계
- 필요 시에만 교차 리전
- S3 Transfer Acceleration (업로드)

5. Private IP 사용

- AZ 간 전송 시 Private IP 사용
- Public IP는 인터넷 경유 (비용 발생)

VPC Endpoint 비용

Gateway Endpoint (S3, DynamoDB)

- 무료 (데이터 전송 비용만)
- NAT Gateway 대체

Interface Endpoint (기타 서비스)

- $\$0.01/\text{시간} = \$7.2/\text{월}$
- 데이터 전송: $\$0.01/\text{GB}$

- NAT Gateway보다 저렴 (대량 전송 시)

NAT Gateway 비용 비교



NAT Gateway:
- 시간당: \$0.045 = \$32.4/월
- 데이터: \$0.045/GB

Interface Endpoint:
- 시간당: \$0.01 = \$7.2/월
- 데이터: \$0.01/GB

절감: 약 75%

SCENARIO

| 상황 | 정답 | 이유 |
|-------------------------|---------------------|----------------|
| Private Subnet S3 접근 비용 | Gateway Endpoint 무료 | |
| 글로벌 콘텐츠 제공 | CloudFront | 오리진 요청 감소 + 저렴 |
| 대량 온프레미스 전송 | Direct Connect | 60-70% 절감 |
| 리전 간 전송 최소화 | 동일 리전 설계 | 리전 간 비용 제거 |
| AZ 간 통신 | Private IP | Public IP는 비용 |

시험 키워드 매칭

- "데이터 전송 비용" → CloudFront, VPC Endpoint
- "NAT Gateway 비용" → Gateway Endpoint
- "대량 전송" → Direct Connect

TRAP

- ❌ NAT Gateway로 S3 접근 (Gateway Endpoint 무료)
- ❌ CloudFront는 모든 비용 해결 (구성 필요)
- ❌ Direct Connect는 소량 전송에 유리 (대량에만)

비용 최적화 체크리스트

즉시 실행 (Quick Wins)

- ☐ 유휴 리소스 식별 및 삭제
- ☐ 미연결 EIP 제거
- ☐ 미사용 EBS 볼륨 삭제
- ☐ 오래된 스냅샷 삭제

- ☐ Cost Anomaly Detection 활성화

✓ 단기 (1개월)

- ☐ Reserved/Savings Plans 권장 확인
- ☐ S3 Lifecycle 정책 설정
- ☐ Budgets 알림 설정
- ☐ Cost Allocation Tags 적용
- ☐ Right Sizing (Compute Optimizer)

✓ 중기 (3개월)

- ☐ Reserved/Savings Plans 구매
- ☐ Spot Instances 도입 (배치/CI/CD)
- ☐ VPC Gateway Endpoint 구성
- ☐ 인스턴스 스케줄링 (개발/테스트)
- ☐ CloudFront 도입

✓ 장기 (6개월+)

- ☐ 아키텍처 재설계 (서버리스)
- ☐ Graviton 인스턴스 전환
- ☐ Multi-AZ 최적화
- ☐ Direct Connect (대량 전송)
- ☐ Cost and Usage Report 분석

비용 최적화 의사결정 트리



워크로드 특성?

└─ 예측 가능?

| └─ Yes → Savings Plans (Compute) ← 권장

| └─ No → On-Demand (단기), Spot (중단 허용)

|

└─ 실행 시간?

| └─ 24/7 → Reserved/Savings Plans

| └─ 간헐적 → Lambda/Fargate

| └─ 비업무 시간 미사용 → 스케줄링

|

└─ 중단 허용?

└─ Yes → Spot (90% 절감)

└─ No → On-Demand/Reserved

 **비용 최적화는 지속적인 프로세스입니다! 매월 Trusted Advisor와 Cost Explorer를 확인하세요.**

다음: [Part 5 - 시험 전략 & 키워드 총정리](#)