쿠버네티스 배치 워크로드 스케줄링: Spark, Flink, Trino를 위한 실전 가이드

Apache YuniKorn, Volcano, Kueue 세 가지 주요 스케줄러가 쿠버네티스 배치 작업 관리의 표준으로 자리잡고 있다. 2024-2025년 기준, YuniKorn은 멀티 테넌시와 대규모 엔터프라이즈 환경에서 강점을 보이고(2000+ 노드에서 초당 610+ 할당), Kueue +3 Volcano는 HPC와 AI/ML 워크로드에서 업계 선도적 위치를 차지하며(60+ 프로덕션 배포), InfraCloud Kueue는 쿠버네티스 네이티브 접근으로 도입 장벽을 낮추고 있다(IBM Vela에서 90% 이상 GPU 활용률 달성). Red Hat ACM Digital Library 초기 단계 조직은 Kueue로 시작해 3-6개월 후 워크로드 패턴에 따라 YuniKorn이나 Volcano로 전환하는 단계적 접근이 효과적이다.

쿠버네티스에서 데이터 처리 워크로드를 운영할 때 리소스 스케줄링과 우선순위 관리는 필수적이다. Sched +2 기본 쿠버네티스 스케줄러는 단순한 워크로드에는 적합하지만, Spark나 Flink 같은 분산 처리 프레임워크는 갱 스케줄링(gang scheduling), 계층적 큐, 정교한 선점(preemption) 메커니즘이 필요하다. InfraCloud CNCF 본 보고서는 Apache YuniKorn, Volcano, Kueue를 중심으로 실제 도입 시 고려해야 할 기술적 세부사항과 의사결정 프레임워크를 제공한다.

세 가지 주요 솔루션 개요와 최신 동향

쿠버네티스 배치 스케줄링 생태계는 2024-2025년 들어 성숙기에 접어들었다. 세 가지 주요 솔루션이 각각 다른 철학과 강점을 가지고 시장을 형성하고 있다.

Apache YuniKorn은 2022년 3월 Apache 최상위 프로젝트로 졸업하며 엔터프라이즈급 안정성을 입증했다. Apache +2 최신 버전 v1.6.3(2025년 1월 v1.6.1 릴리스) Apache은 쿠버네티스 기본 스케줄러를 완전히 대체하는 아키텍처를 채택한다. GitHub +2 비동기 이벤트 기반 프레임워크로 설계되어 **2000개 노드 규모에서 초당 610건 이상의 리소스 할당**을 처리하며, Apache +2 ACM 연구에 따르면 기본 스케줄러 대비 워크플로 실행 시간을 최대 4.6배 단축하고 클러스터 활용률을 3배 향상시킨다. ACM Digital Library Alibaba, Apple, Cloudera, Lyft, Visa, Zillow 등 주요 기업들이 프로덕션에서 사용 중이며, Apache +2 YARN 경험이 있는 조직에게 친숙한 계층적 큐 모델을 제공한다.

Volcano는 CNCF Incubating 프로젝트로 HPC와 AI/ML 워크로드에 특화되어 있다. Medium GitHub 최신 v1.13.0(2025년)은 LeaderWorkerSet 통합으로 멀티 호스트 AI 추론을 지원하고, 네트워크 토폴로지 인식 스케줄링을 위한 HyperNode 자동 발견 기능을 추가했다. CNCF +2 기본 스케줄러와 공존하는 아키텍처로 **초당 1000개 pod 처리 능력**을 보유하며, Huawei Cloud Huawei, AWS, Apple, Baidu, Tencent 등 60개 이상 기업이 프로덕션에서 운영 중이다. CNCF +2 VolcanoJob이라는 커스텀 CRD를 통해 복잡한 작업 종속성과 워크플로 오케스트레이션을 지원하며, Alibaba Cloud Community GitHub NVIDIA DGX Cloud 사례에서 GPU 활용률을 90%까지 끌어올린 실적이 있다. NVIDIA Developer

Kueue는 Kubernetes SIG-Batch의 공식 프로젝트로 v0.14.1(2025년 10월)이 최신 버전이다. Kubernetes +4 쿠버네티스 네이티브 설계를 따라 기본 스케줄러를 대체하지 않고 작업 레벨에서 큐잉과 리소스 할당을 관리한다. Kueue +3 spec.suspend 필드를 활용해 작업을 일시 중지하거나 재개

하며, Kubernetes +2 **IBM Vela 슈퍼컴퓨터에서 OpenShift와 결합해 90% 이상 GPU 활용률**을 달성했다. Red Hat MultiKueue 기능으로 멀티 클러스터 작업 디스패칭을 지원하며, GitHub +3 Shopee, RedHat, CyberAgent, DaoCloud 등에서 채택했다. Medium 2025년 로드맵은 코스트 최적화 전략과 체크포인트 기반 협력적 선점에 중점을 둔다. GitHub

핵심 기능 비교: 갱 스케줄링부터 멀티 테넌시까지

배치 워크로드 스케줄링에서 가장 중요한 기능들을 심층 비교하면 각 솔루션의 차별점이 명확해진다.

갱 스케줄링은 Spark나 Flink 같은 분산 프레임워크에서 필수적이다. Ray +2 드라이버와 여러 executor 또는 TaskManager가 모두 동시에 스케줄되지 않으면 리소스 데드락이나 부분 실행으로 인한 낭비가 발생한다. CNCF YuniKorn은 TaskGroups 어노테이션을 통해 플레이스홀더 pod를 미리 생성해 리소스를 예약하는 방식으로 구현하며, 이는 클러스터 오토스케일링과 효율적으로 작동한다. Cloudera Blog +2 Volcano는 PodGroup CRD와 minAvailable 필드로 "전부 아니면 전무(all or nothing)" 원칙을 구현하며, Alibaba Cloud Community Ray 작업별로 태스크 수준의 최소 요구사항을 설정할 수 있어 더욱 세밀한 제어가 가능하다. Kubeflow +4 Kueue는 순수 갱 스케줄링을 제공하지 않지만, 코호트 기반 borrowing과 admission control로 유사한 효과를 낸다. Ray

우선순위 관리와 선점에서 세 솔루션 모두 강력한 기능을 제공하지만 접근법이 다르다. YuniKorn은 큐 우선순위와 작업 우선순위를 계층적으로 관리하며, 멀티 레벨 선점을 지원한다. Apache Volcano는 v1.10부터 큐 간 우선순위를 추가했고, CNCF 세 가지 리소스 모델(capability, deserved, guarantee)로 복잡한 리소스 공유 시나리오를 처리한다. **capacity는 절대 초과할 수 없는 상한선, deserved는 공정 분배 기준으로 초과 시 회수 가능, guarantee는 절대 보장되는 최소 리소스**를 의미한다. Volcano GitHub Kueue는 협력적 선점(cooperative preemption)을 2025년 로드맵으로 개발 중이며, 체크포인트를 지원하는 워크로드에 적합하도록 설계된다. SRE DevOps GitHub

멀티 테넌시와 리소스 격리는 부서별 권한 관리가 필요한 환경에서 핵심이다. YuniKorn은 YARN 스타일의 계층적 큐를 제공해 조직→부서→팀 구조를 자연스럽게 표현하며, apache DRF(Dominant Resource Fairness) 알고리즘으로 CPU, 메모리, GPU를 포괄하는 다차원 공정성을 보장한다. InfraCloud +3 Volcano는 큐 기반 격리와 v1.9부터 계층적 큐를 지원하며, CNCF +2 Ruitian Capital 사례에서 200개 노드에서 하루 10-30만 개 pod를 처리하며 네임스페이스별 PVC 권한과 CephFS 디렉토리 수준 접근 제어를 구현했다. Volcano Kueue는 ClusterQueue와 LocalQueue 개념으로 멀티 테넌시를 구현하며, ResourceFlavor를 통해 다양한 리소스 타입(온디맨드, 스팟, GPU 등)을 관리한다. Kubernetes +5

확장성과 성능에서 YuniKorn은 증분 노드 정렬 알고리즘으로 기본 스케줄러 대비 35-74배 빠른 성능을 보이며, Apache Volcano는 세션 기반 캐싱과 병렬 노드 스코어링으로 10,000개 이상 노드를 목표로 한다. Medium 실제 프로덕션에서 Volcano는 노드 필터링 캐시(predicate.CacheEnable)와 플러그인 가중치 설정으로 최적화가 가능하며, 배치 연산으로 API 서버 부하를 줄인다. Volcano Kueue는 기본 스케줄러에 의존하므로 스케줄링 레이턴시는 쿠버네티스 기본 성능을 따르지만, 작업 레벨 오버헤드가 최소화되어 실용적이다.

Spark, Flink, Trino 오퍼레이터 통합 실무

각 프레임워크와 스케줄러의 통합 방식을 이해하는 것이 성공적인 도입의 핵심이다.

Apache Spark 통합은 세 스케줄러 모두에서 프로덕션급 지원을 제공한다. Kubeflow Spark 3.3부터 Volcano와 YuniKorn 지원이 공식 포함되었으며(SPARK-36057), Apache JIRA CNCF Kubeflow Spark Operator는 Apache YuniKorn을 갱 스케줄링 성능을 위해 권장한다. Kubeflow Kubeflow YuniKorn과 Spark를 통합하면 드라이버/executor 생성 전에 플레이스홀더 pod로 업스케일링이 가능해 리소스 세분화를 방지하고, 클러스터 오토스케일링이 효율적으로 작동한다. Cloudera Blog Alibaba, Apple, Cloudera, Lyft, Visa 등이 이 조합을 프로덕션에서 사용 중이다. Apache +2 Volcano는 SparkApplication spec에 batchScheduler: "volcano" 설정과 PodGroup의 minResources 구성이 필요하며, Kubeflow Kubeflow AWS EMR on EKS에서 공식 튜토리얼이 제공된다. AWS Kueue는 Kubernetes Job API와 suspend 필드를 통해 통합되며, Medium 리소스 쿼터 관리와 작업 큐잉에 적합하다. Kubernetes

Flink Kubernetes Operator는 v1.10.0(2024년 10월)이 최신이며, FlinkDeployment와 FlinkSessionJob 커스텀 리소스를 관리한다. Decodable +2 Volcano와의 통합은 JobManager와 TaskManager에 동시에 schedulerName: volcano를 설정하며, 리소스 제약이 있는 클러스터에서 갱 스케줄링의 이점이 크다. GitHub Kubeflow GoogleCloudPlatform의 flink-on-k8s-operator(현재 deprecated)가 통합 패턴을 보여주며, Data on Kubernetes 커뮤니티에서 YuniKorn과 Flink의 멀티 테넌트 스트리밍 작업에 대한 논의가 활발하다. Flink 1.18+는 Adaptive Scheduler와 in-place rescaling을 지원하며, Reactive Mode로 동적 병렬성 조정이 가능해 스케줄러와 독립적으로 오토스케일링이 작동한다. Apache

Trino의 경우 2024년 10월 기준 공식 오퍼레이터가 존재하지 않으며, 2024년 10월 30일 커뮤니티 미팅에서 개발 논의가 시작되었다. Trino 현재는 Helm 차트를 통한 배포가 표준이며, Trino Trino Stackable Data Platform의 Trino Operator가 가장 성숙한 서드파티 솔루션이다. GitHub Trino는 쿼리 엔진으로 배치 처리 프레임워크가 아니므로 Trino 특수한 배치 스케줄러가 필수는 아니며, coordinator/worker 레벨에서 리소스 할당을 관리한다. Stackable Medium 실무에서는 Spark/Flink가 Iceberg나 Delta Lake에 데이터를 쓰고 Trino가 쿼리하는 패턴이 일반적이므로, Trino **스케줄러 선택은 Spark/Flink 요구사항에 따라 결정**하면 된다.

2024-2025 주요 트렌드와 커뮤니티 동향

쿠버네티스 배치 워크로드 생태계는 AI/ML 붐과 함께 급격한 성장을 경험하고 있다. H100 GPU가 중고 시장에서 4만 달러 이상에 거래되는 GPU 부족 현상이 효율적인 스케줄링과 활용률 최적화 수요를 폭발시켰다. Red Hat 분산 학습에서 갱 스케줄링이 필수가 되면서, 세 스케줄러 모두 AI/ML 워크로드 지원을 강화하고 있다.

YuniKorn의 2024-2025 로드맵은 안정성과 성능에 집중한다. v1.6.0(2024년 9월)에서 일부 오토스케일링 불안정성이 발견되어 v1.6.1(2025년 1월)에서 패치되었으며, Apache v1.5.0(2024년 3월)은 219개 JIRA를 해결하며 이벤트 스트리밍 지원(YUNIKORN-2115), 쿠버네티스 1.29 지원(1.24-1.29 클러스터 지원), Web UI 개선(노드 필터링, 활용률 차트, 할당 정렬), 큐 메트릭 레이블 지원을 추가

했다. Apache 커뮤니티는 2024년 4분기 175개 이상의 새 JIRA와 183개 PR로 높은 활동성을 보인다. Apache Apache

Volcano는 v1.13.0에서 보안 강화와 함께 LeaderWorkerSet 통합으로 멀티 호스트 AI/ML 추론을 지원하고, 레이블 기반 HyperNode 자동 발견으로 InfiniBand와 RoCE 같은 네트워크 토폴로지를 인식한다. CronVolcanoJob 지원으로 스케줄된 배치 작업이 가능해졌으며, GitHub v1.12는 UFM 자동 발견으로 네트워크 토폴로지 인식 스케줄링을 강화했다. CNCF Incubating 상태이며 GitHub KubeCon 2024 China에서 여러 발표가 있었고, 인터넷/클라우드, 금융, 제조 분야에서 광범위하게 채택되었다. CNCF GitHub

Kueue의 2025년 우선순위는 MultiKueue 사용자 경험 개선(멀티 클러스터 작업 디스패칭), flavor 할당 전략(비용 최소화 vs borrowing 최소화), 체크포인트를 지원하는 워크로드를 위한 협력적 선점에 있다. GitHub github TensorFlow, PyTorch, RayJob, RayCluster, Kubeflow와의 네이티브 통합을 제공하며, Medium +4 IBM Vela 사례는 OpenShift에서 Kueue로 90% 이상 GPU 활용률을 달성한 성공 스토리다. Red Hat

KubeCon + CloudNativeCon Europe 2025(2025년 4월)에서 Wei Huang(Apple)과 Shiming Zhang(DaoCloud)이 "Kueue, Volcano, YuniKorn의 비교 분석" 세션을 진행할 예정이며, 설계 트레이드오프와 사용 사례 적합성에 초점을 맞춘다. Sched Sessionize Data on Kubernetes 커뮤니티는 데이터 워크로드에 Apache YuniKorn이나 Volcano가 "기본 요구사항"이 되고 있다고 보고하며, Karpenter와 커스텀 스케줄러의 통합이 커뮤니티에서 요청되고 있다. GitHub GitHub

초기 단계 도입을 위한 의사결정 프레임워크

배치 워크로드가 적고 향후 성장을 예상하는 조직은 단계적 접근이 효과적이다. 복잡도와 기능 사이의 균형을 고려한 전략적 선택이 필요하다.

1-3개월: 시작 단계에서는 Kueue를 권장한다. 쿠버네티스 네이티브 설계로 도입 리스크가 낮고, InfraCloud DEV Community Helm이 아닌 단순 kubectl apply로 설치 가능하며(kubectl apply --server-side -f https://github.com/kubernetes-sigs/kueue/releases/download/v0.14.1/manifests.yaml), SRE DevOps +2 ClusterQueue와 LocalQueue 정의만으로 시작할 수 있다. github +2 기본 스케줄러를 대체하지 않아 롤백이 쉽고, 리소스 활용 패턴과 작업 큐잉 패턴을 모니터링하며 학습할 수 있다.

3-9개월: 확장 단계에서 워크로드 특성이 명확해지면 평가를 시작한다. Spark 작업이 많으면 YuniKorn이나 Volcano를 평가하고, ML/AI 학습 워크로드가 주를 이루면 Volcano로, 다양한 워크로드 타입이 혼재하면 YuniKorn으로 기울어진다. 개발/스테이징 환경에서 파일럿 테스트를 실행하며, 갱 스케줄링이 실제로 얼마나 필요한지, 멀티 테넌시 요구사항이 얼마나 복잡한지 데이터 기반으로 판단한다.

9개월 이상: 프로덕션 단계에서 증거를 바탕으로 고급 스케줄러에 커밋한다. 마이그레이션은 신중하게 계획하며, 기존 스케줄러와 병렬로 테스트하고, 마이그레이션 전후 메트릭을 수립해 ROI를 측정한다. YuniKorn은 스케줄러 교체이므로 더 높은 커밋먼트가 필요하고, Volcano는 VolcanoJob CRD로 작업 스펙 변경이 필요하지만 기본 스케줄러와 공존이 가능하다.

선택 기준을 명확히 하면 의사결정이 쉬워진다. YuniKorn을 선택해야 하는 경우는 강력한 멀티 테넌시가 필요하고, 배치와 서비스 워크로드의 공정성이 중요하며, YARN 마이그레이션이 있고, 2000+ 노드 규모가 입증된 성능이 필요하고, 학습에 투자할 리소스가 있을 때다. InfraCloud Volcano는 HPC, AI/ML 학습, 과학 컴퓨팅 워크로드가 주를 이루고, Rafay InfraCloud 복잡한 작업 종속성과 워크플로 오케스트레이션이 필요하며, 갱 스케줄링이 필수적이고, Spark/Flink/PyTorch/TensorFlow 프레임워크를 네이티브로 사용하며, 작업 우선순위, 백필, 토폴로지 인식 같은 고급 기능이 필요할 때 선택한다. InfraCloud Kueue는 기존 쿠버네티스 설정에 최소 변경을 원하고, 단순한 작업 큐잉과 리소스 쿼터가 목적이며, 쿠버네티스 네이티브 접근을 선호하고, InfraCloud 초기 단계에서 단순한 배치 요구사항이 있으며, 멀티 클러스터 작업 디스패칭이 우선순위고, 낮은 학습 곡선이 중요할 때 적합하다. Rafay +3

운영 복잡도와 실제 프로덕션 고려사항

스케줄러 선택에서 기능만큼 중요한 것이 운영 복잡도다. 학습 곡선, 설치 난이도, 지속적인 운영 오버헤드를 현실적으로 평가해야 한다.

설치 난이도는 세 솔루션 모두 낮은 편이다. YuniKorn은 `helm install yunikorn yunikorn/yunikorn -n yunikorn --create-namespace`로 2-3분 내 설치되지만, Medium Apache 기본 스케줄러를 대체하므로 클러스터 전체에 영향을 미친다. Volcano는 `helm install volcano volcano-sh/volcano -n volcano-system --create-namespace`로 설치하며, Medium scheduler, controllers, admission 세 개 디플로이먼트와 admission webhook 서비스, 인증서 설정을 위한 init job이 생성된다. Medium +3 Kueue는 단일 YAML 적용으로 가장 간단하지만, ClusterQueue와 ResourceFlavor 설정 이해가 필요하다. SRE DevOps GitHub

학습 곡선에서 차이가 크다. Kueue는 쿠버네티스 개념만 알면 1-2주 내 숙달 가능하며, 공식 문서가 간결하고 예제가 풍부하다. YuniKorn은 중간 난이도로 YARN 경험이 있으면 2-3주, 없으면 3-4주 정도 필요하며, 계층적 큐 구조와 비동기 이벤트 모델 이해가 관건이다. Volcano가 가장 복잡해 4-6주 학습이 권장되며, 3단계 리소스 모델(capacity/deserved/guarantee) 이해, 액션과 플러그인 설정, 선점 정책 튜닝, 스케줄링 결정 디버깅에 시간이 필요하다. Ruitian Capital 사례는 1주차에 기본 VolcanoJob과 갱 스케줄링, 2주차에 큐 관리, 3주차에 프레임워크 통합, 4주차에 고급 기능, 그 이후 프로덕션 튜닝으로 온보딩을 구조화했다.

운영 오버헤드도 고려해야 한다. YuniKorn은 큐 설정 ConfigMap 관리, 할당 로그 모니터링, 주기적인 큐 쿼터 검토가 필요하고, Web UI로 실시간 모니터링이 가능하다. Apache apache Volcano는 VolcanoJob과 PodGroup CRD 라이프사이클 관리, 작업 실패 조사(job phase가 Inqueue에서 멈추는 선점 버그 #2034, #3186 같은 알려진 이슈), 버전 업그레이드(다운타임 최소)가 있으며, Prometheus 메트릭이 내장되어 Grafana 대시보드 활용이 가능하다. Kueue는 가장 가벼우며, ClusterQueue와 LocalQueue 정의 업데이트, 작업 suspend/resume 상태 모니터링, 코호트 설정 조정 정도가 전부다.

알려진 제한사항을 이해하는 것이 실패를 방지한다. Volcano는 선점 버그(고우선순위 작업이 Inqueue에 갇힘, vGPU 선점 실패), GPU 조각화 문제(#3948, 분산된 유휴 GPU), 큐 검증 이슈(guarantee가 allocatable을 초과하면 스케줄러 패닉 #3127)가 있으며, etcd 1.5MB 요청 크기 제한이 10만+ pod 클러스터에 영향을 줄 수 있다. YuniKorn은 v1.6.0에서 오토스케일링 불안정성이 보고되

었으나 v1.6.1에서 해결되었다. Kueue는 갱 스케줄링 부재와 제한적인 작업 종속성 지원이 한계다. 세 솔루션 모두 50개 미만 소규모 클러스터나 순수 스트리밍 처리, Windows 환경에는 과도하거나 부적합하다.

한국 커뮤니티와 리소스 현황

한국어 리소스와 국내 채택 현황에 대한 조사 결과, 2024-2025년 기준으로 **공개된 한국어 문서나 커뮤니티 리소스는 제한적**이다. YuniKorn, Volcano, Kueue 모두 영어 공식 문서가 주를 이루며, 한국 기업의 구체적인 채택 사례는 공개 자료에서 확인되지 않았다.

한국 커뮤니티 참여 방법으로는 CNCF Korean Community Chapter를 통한 쿠버네티스 스케줄링 토픽 탐색, KubeCon + CloudNativeCon의 한국어 번역 트랙 활용, Naver, Kakao, Samsung 같은 쿠버네티스를 적극 활용하는 국내 기업의 기술 블로그(tech.kakao.com, d2.naver.com) 모니터링, Kubernetes Korea Slack 채널이나 한국 DevOps 커뮤니티(Facebook, KakaoTalk 그룹) 참여가 있다.

이는 **한국어 콘텐츠 기여 기회**를 의미한다. Volcano, YuniKorn, Kueue 공식 문서에 한국어 번역 기여, 세 솔루션을 비교하는 한국어 블로그 포스트 작성, 국내 쿠버네티스 밋업에서 실전 경험 공유 등이 커뮤니티에 가치를 더할 것이다. 한국어 리소스 부족은 사용 부족을 의미하지 않으며, 오히려 선도 조직이 지식을 공유할 기회로 봐야 한다.

최종 권장사항과 실행 로드맵

쿠버네티스 배치 스케줄링은 더 이상 선택이 아닌 필수다. Apache Spark, Flink, Trino를 운영하는 조직은 워크로드 특성과 조직 성숙도에 맞는 스케줄러를 전략적으로 선택해야 한다.

즉시 실행 가능한 로드맵은 다음과 같다. 현재 매우 초기 단계라면 Kueue를 1-2주 내 도입해 리소스 쿼터와 기본 큐잉을 설정하고, 3개월간 작업 패턴, 리소스 활용률, 큐 대기 시간을 모니터링한다. Spark 작업이 주를 이루고 갱 스케줄링 필요성이 확인되면 4-6개월 차에 YuniKorn 또는 Volcano 파일럿을 개발 클러스터에서 시작하며, Spark Operator 통합, 갱 스케줄링 효과, 리소스 활용률 변화를 측정한다. 6-9개월 차에 부서별 큐 설정, 우선순위 정책, 선점 규칙을 스테이징 환경에서 검증하고, 9-12개월 차에 프로덕션 마이그레이션을 수행하며 blue-green 전략으로 리스크를 관리한다.

의사결정 체크리스트로 명확성을 확보한다. Spark/Flink에서 갱 스케줄링이 필수인가(Yes → YuniKorn/Volcano)? 3개 이상 부서/팀이 리소스를 공유하는가(Yes → YuniKorn/Volcano)? HPC나 AI/ML 학습 워크로드가 주를 이루는가(Yes → Volcano)? 운영 복잡도를 최소화하고 싶은가(Yes → Kueue)? 클러스터 규모가 1000+ 노드를 목표로 하는가(Yes → YuniKorn)? 6개월 내 단순하게 시작하고 싶은가(Yes → Kueue로 시작)?

성공 지표 정의가 ROI를 입증한다. 스케줄러 도입 전후로 작업 완료 시간(평균, p95, p99), 클러스터 리소스 활용률(CPU, 메모리, GPU), 큐 대기 시간, 작업 실패율, 리소스 충돌/데드락 발생 빈도를 측정한다. YuniKorn과 Kueue 사례에서 보듯 3배 활용률 증가와 90% GPU 활용률은 달성 가능한 목표다.

2025년 쿠버네티스 배치 스케줄링 생태계는 성숙하고 프로덕션 검증되었다. YuniKorn은 대규모 멀티 테넌트 엔터프라이즈에 최적이고, Volcano는 HPC와 AI/ML 워크로드의 사실상 표준이며,

Kueue는 쿠버네티스 네이티브로 도입 장벽을 낮춘다. 초기 단계 조직은 Kueue로 시작해 워크로드가 복잡해지면 YuniKorn이나 Volcano로 진화하는 전략이 리스크를 최소화하며 장기적 성공을 보장한다.

쿠버네티스에서 데이터 플랫폼을 구축하는 여정에서 올바른 스케줄러 선택은 단순히 기술 결정이 아니라 향후 3-5년 확장성과 운영 효율성을 결정하는 전략적 투자다. 지금 작은 규모라도 미래 성장을 고려한 아키텍처 기반을 다지는 것이 현명하며, 이 보고서가 제시한 프레임워크와 실전 인사이트가 그 여정의 나침반이 되기를 바란다.