AI Operations for infrastructure management, monitoring, and cluster orchestration/job scheduling focuses on keeping GPU-heavy AI platforms reliable, observable, and efficiently utilized at scale. It combines classic SRE/DevOps practices with AI-specific needs such as GPU pooling, mixed training/inference workloads, and data-driven monitoring.

Infrastructure management and monitoring

AI infrastructure management is about provisioning, configuring, and governing compute, storage, and networking so training and inference can run predictably and cost-effectively. Monitoring adds continuous measurement of health, performance, and cost across these layers.

Key goals and concepts

- Reliability and availability: Maintain SLAs around uptime, latency, and job completion, even as workloads spike or models grow.
- Capacity and cost control: Right-size GPU/CPU, storage, and network capacity, using autoscaling, spot/reserved instances, and workload placement to reduce waste.
- Security and governance: Enforce access control, auditability, and policy for data, models, and clusters (e.g., isolation by team or environment).
- Observability over time: Track metrics, logs, and traces that describe both system behavior (GPU utilization, I/O, memory) and model behavior (accuracy, drift).

Core components of infrastructure

- Compute: GPU/CPU clusters on-prem, cloud, or hybrid; often exposed through Kubernetes, Slurm, or cloud ML platforms.
- Storage: High-throughput object and file storage for datasets, checkpoints, and artifacts; data versioning layers for reproducibility.
- Networking: High-bandwidth, low-latency fabrics (e.g., InfiniBand, 100G+ Ethernet) for distributed training and data pipelines.
- Control plane: APIs and services for provisioning (Terraform, cloud IaC), identity, secrets, and configuration management.

Monitoring and observability practices

- Metrics:
    - System: GPU utilization, GPU memory, CPU, RAM, I/O, network throughput, node health.

    - Workload: job queue depth, training throughput, time-to-train, failed jobs, autoscaling events.
    - Model: accuracy, precision/recall, latency, throughput, data drift, fairness indicators.
- Tools: Prometheus and Grafana are common for infrastructure metrics and dashboards; ELK/Opensearch for logs; ML-specific tools like Evidently for model monitoring.
- Alerting and SLOs: Threshold-based and anomaly alerts (e.g., GPU idle while jobs queued, accuracy below baseline) tied to incident response runbooks.
- Auditing and compliance: Retain logs, configuration history, and model versions to support reproducibility, governance, and regulatory checks.

Operational workflows

- Provisioning and change management: Use IaC and GitOps to create/modify clusters, node pools, and storage in a controlled, reviewable way.
- Continuous monitoring: Always-on collection of metrics and logs, with dashboards per environment (dev, staging, prod) and workload type (training vs inference).
- Incident response: On-call rotations, playbooks for node failures, degraded GPUs, storage hot-spots, or model performance regressions.
- Cost and efficiency reviews: Regular reports on GPU utilization, idle capacity, and per-project spend, feeding back into scheduling and capacity planning.

Cluster orchestration and job scheduling

Cluster orchestration and job scheduling decide where and when AI workloads run across shared compute, coordinating containers, GPUs, and dependencies. The aim is to maximize resource utilization while honoring priorities, SLAs, and hardware constraints.

Core concepts

- Orchestration vs scheduling:
    - Orchestration manages lifecycle of services/containers (deploy, scale, restart, roll out/roll back).
    - Scheduling decides which node gets which job or pod, based on policies, constraints, and current load.
- Workload types:
    - Batch/distributed training (long-running, GPU-intensive, may use MPI or Horovod).

    - Online inference services (latency-sensitive, horizontally scalable).
    - Data/ETL and feature engineering jobs, often pipeline-oriented.
- Policies and constraints: Priorities, quotas per team, GPU type and count, locality to data, preemption rules, and fairness.

Common platforms and tools

- Kubernetes:
    - Provides container orchestration, autoscaling, rolling updates, and service discovery for AI microservices and many training workloads.
    - GPU scheduling via device plugins and node selectors; operators (e.g., for distributed training frameworks) encapsulate job logic.
- Slurm (HPC scheduler):
    - Classic batch scheduler used for large-scale AI/ML and HPC training; organizes jobs into queues and partitions, with a central controller allocating resources.
    - Supports advanced scheduling policies, backfilling, and tight control over GPU/CPU allocation for long-running jobs.
- Hybrid models (Slurm on Kubernetes, SUNK):
    - Integrations such as SUNK let Kubernetes pods map onto Slurm jobs, enabling Slurm-managed training to share nodes with Kubernetes-managed inference.
    - This leverages strengths of both systems while simplifying cluster management and improving resource utilization.

How orchestration and scheduling work together

- Submission and description: Users submit jobs as manifests (Kubernetes) or batch scripts (Slurm) describing resource needs (GPU count, memory, time limits) and dependencies.
- Scheduling decisions: The scheduler examines current node states and queues, applies policies (priority, fairness, preemption), and binds jobs or pods to specific nodes.
- Execution and lifecycle: Orchestration handles container creation, retries, health checks, and scaling, while the scheduler continues to optimize placement as cluster state changes.
- Feedback loop to monitoring: Scheduler metrics (queue times, utilization, failed placements) feed back into dashboards and capacity planning.

Best-practice patterns

- Separate training and inference classes: Use node pools or partitions tuned for training (large GPUs, longer jobs) vs inference (smaller, latency-optimized instances).
- Priority and preemption: Allow critical inference or high-priority research jobs to preempt low-priority or opportunistic training workloads.
- Autoscaling and burst to cloud: Scale node groups based on queue depth and utilization; offload surplus training to cloud GPUs during peaks.
- Pipeline-aware scheduling: Align data preprocessing, training, evaluation, and deployment steps via workflow engines (e.g., MLOps platforms) that integrate with the underlying scheduler.

Orchestration vs scheduling tools

The table below highlights typical roles of key technologies in AI operations.

Aspect	**Kubernetes role**	**Slurm role**	**Hybrid SUNK-style role**
Primary focus	Container orchestration for services and jobs	Batch and HPC-style job scheduling	Bridge Kubernetes pods to Slurm jobs
Typical workloads	Inference APIs, microservices, some training	Large distributed training, research batches	Training on Slurm, inference on Kubernetes
Scheduling control	Pod scheduling with policies, autoscaling	Queue/partition-based GPU/CPU allocation	Coordinate placements across both systems
Strengths	Rolling updates, service discovery, ecosystem	Mature HPC features, advanced policies	Shared compute pool, simplified management

When preferred	Production inference, ML platforms, cloud-native apps	Massive training jobs, on-prem GPU clusters	Mixed AI/HPC environments needing both

Linking to MLOps and continuous delivery

AI operations for infrastructure and scheduling is tightly linked to MLOps practices that automate the AI lifecycle. Together they ensure models move from experiment to reliable production service with continuous monitoring and feedback.

Key integrations

- CI/CD and deployment: Pipelines build, test, and deploy containers or model artifacts into orchestrated clusters with versioning and rollbacks.
- Model registries and serving: Registries track model versions; serving platforms run on Kubernetes or similar to expose models behind stable endpoints.
- Continuous training and evaluation: Scheduled retraining jobs and evaluation pipelines run on the same clusters, triggered by data drift or performance degradation.
- Governance and compliance: Policies for who can deploy what, promotion gates based on monitoring metrics, and audit trails across infrastructure, jobs, and models.

If you share your target audience (e.g., executives vs engineers) and time horizon (today vs 3–5 years out), the material can be expanded into a structured multi-chapter section for your larger report.