Lecture 16 - Contents

An overview of the main sections in this lecture.

This outline is for guidance. Navigate the slides with the left/right arrow keys.

Lecture 16:

Medical MLOps: Production-Ready AI Systems

Ho-min Park

homin.park@ghent.ac.kr

powersimmani@gmail.com

MLOps in Healthcare: Overview

Medical MLOps: Production deployment and lifecycle management of ML models in **healthcare environments**

Traditional DevOps

- Code versioning
- CI/CD pipelines
- Infrastructure automation
- Deployment strategies

Medical MLOps

- Code + Data + Model versioning
- Continuous training pipelines
- Model monitoring & drift detection
- Regulatory compliance tracking

Healthcare-Specific Requirements

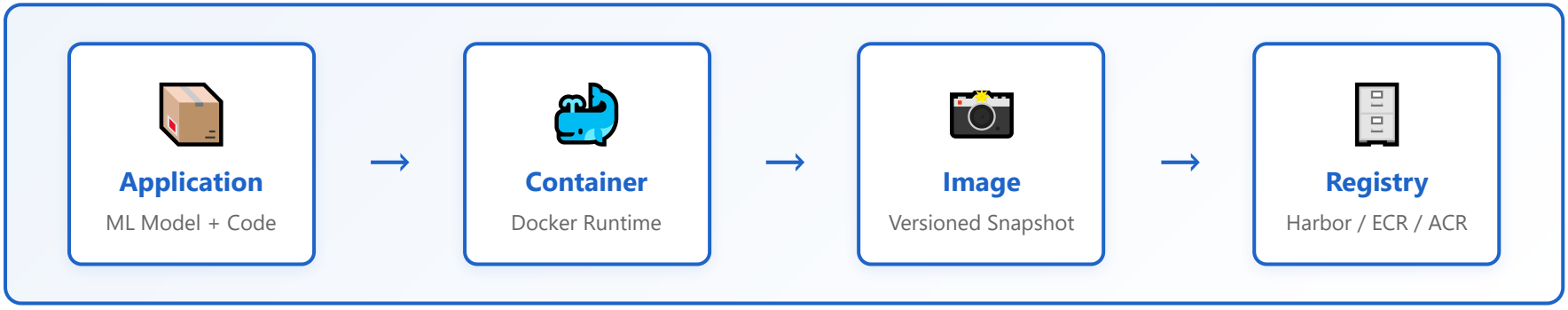
HIPAA Compliance	FDA Validation	Audit Trails
Explainability	Safety Monitoring	Version Control

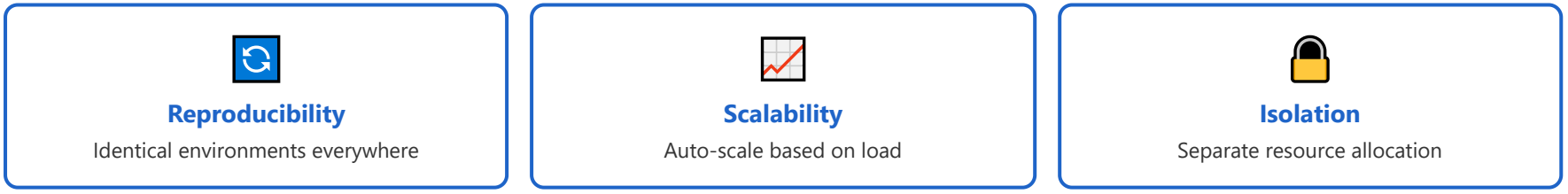
Part 1

Infrastructure and Architecture

Container Orchestration

Container orchestration automates the deployment, management, scaling, and networking of containers





Key Features

- Reproducibility
- Scalability
- Isolation
- Portability

Kubernetes for Medical Deployments

Kubernetes (K8s) orchestrates containerized ML workloads across clusters

Core Components

Pods
Smallest unit containing containers

Services
Expose pods with load balancing

Deployments
Manage replicas & rolling updates

Configuration

ConfigMaps
Store configuration data

Secrets
Manage sensitive credentials

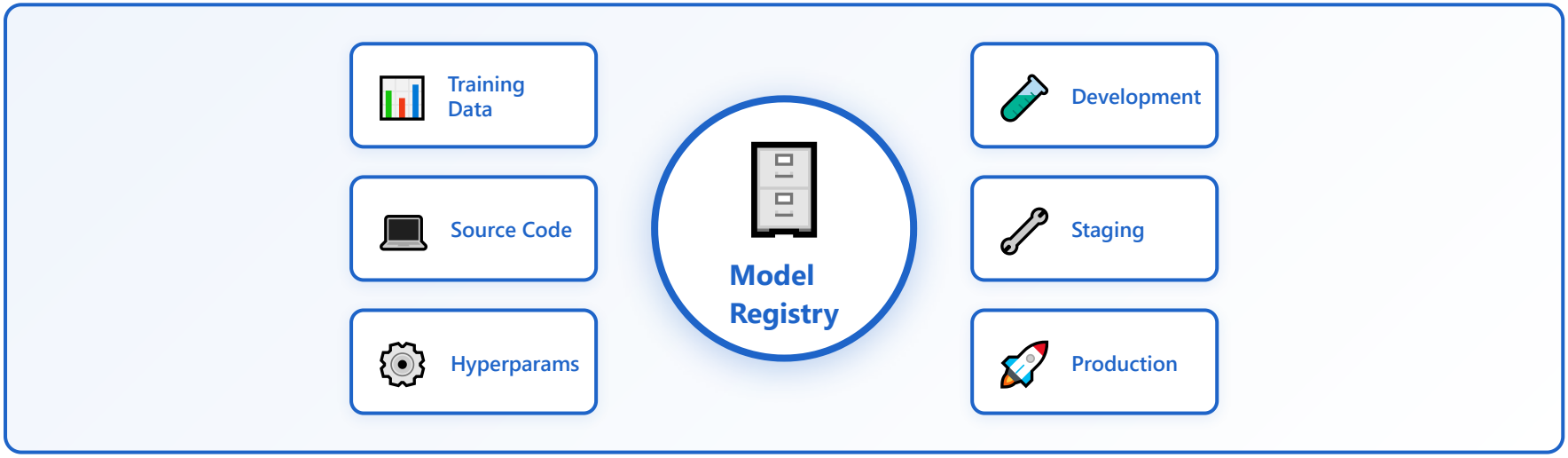
Volumes
Persistent data storage

Scaling Features

- Horizontal Pod Autoscaling (HPA)
- Resource requests/limits
- Node affinity for GPU scheduling

Model Registry

Centralized repository for storing, versioning, and managing ML models



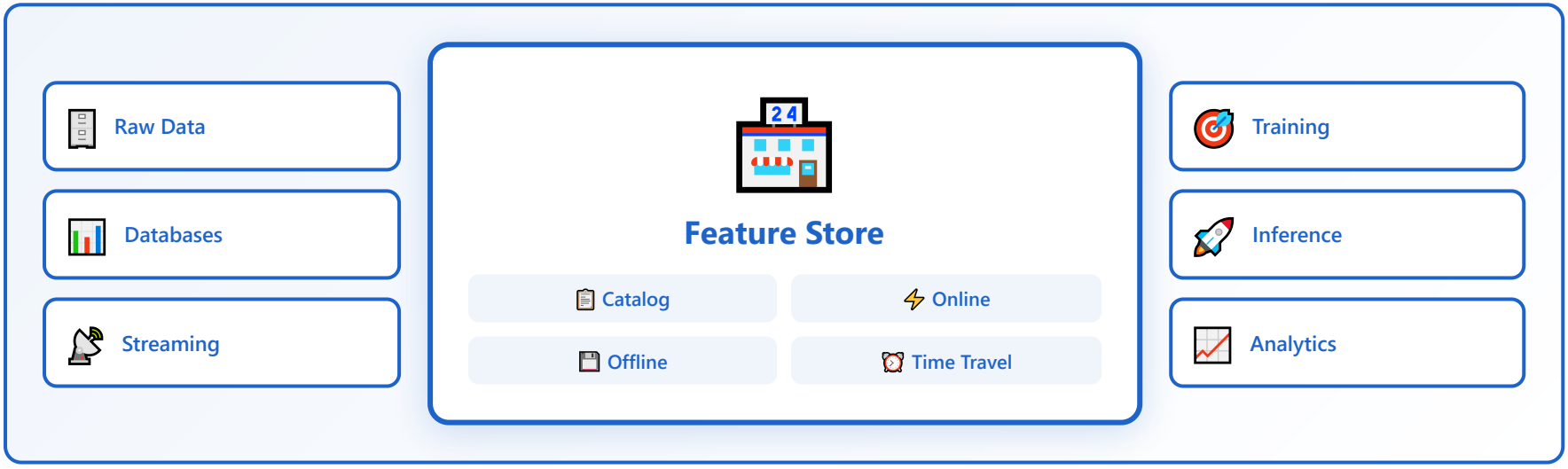
Versioning | **Lineage** | **Stages** | **Access Control**

Tools & Platforms

MLflow Model Registry | SageMaker Model Registry | Weights & Biases Registry

Feature Stores

Centralized platform for storing, managing, and serving ML features



Key Benefits

- ✓ Training-serving consistency
- Feature reuse across models
- Reduces redundant computations

Data Versioning

Track and manage versions of datasets used for ML training and evaluation

Large Medical Images

CT/MRI scans are gigabytes per patient

HIPAA Compliance

Encrypted storage with audit logs

Reproducibility

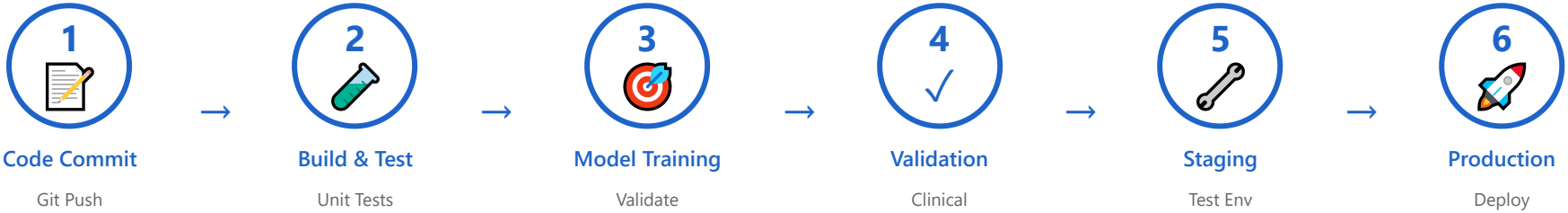
Exact dataset for each model version

🛠 Tools & Platforms

- DVC (Data Version Control)
- Git-LFS
- Pachyderm
- Delta Lake

CI/CD Pipelines for ML

Automated pipelines for Continuous Integration and Continuous Deployment of ML models



Trigger
GitHub, GitLab, Bitbucket webhooks

Testing
pytest, unittest, Docker build

Metrics
Check performance regression

Safety
Ensure thresholds met

Integration
PACS/EHR system testing

Strategy
Canary/Blue-Green rollout

Tools & Platforms

Jenkins | GitHub Actions | GitLab CI | Azure DevOps | CircleCI

Infrastructure as Code (IaC)

Define and provision infrastructure through code instead of manual processes

Reproducibility
Same infra every time

Version Control
Track changes in Git

Documentation
Code is docs

Disaster Recovery
Rebuild quickly

Tools & Platforms

Terraform | AWS CloudFormation | Ansible | Pulumi

Example

```
terraform apply → Provision GPU cluster + networking + storage
```

Part 2

Monitoring and Observability

Model Performance Tracking

Continuously monitor ML model performance in production to detect degradation

Classification Metrics

Track accuracy, precision, recall, F1-score

Monitor diagnostic accuracy

Inference Latency

P50, P95, P99 response times

Real-time diagnosis < 2s

Throughput

Requests/sec, batch processing rate

Handle peak clinic hours

Error Rates

Failed predictions, timeout errors

Alert if errors > 1%

Calibration

Predicted probabilities vs actual outcomes

80% predictions = 80% accurate?

Fairness Metrics

Performance across demographic groups

Equal accuracy across groups

When to Alert

- Accuracy drops > 5%
- Latency exceeds SLA
- Error rate spikes
- Calibration drift

Data Drift Detection

Data Drift: When production data distribution **differs from training data**

Covariate Shift

`P(X) changes`

Input distribution changes

> *Example: New imaging equipment with different resolution/contrast*

Label Shift

`P(Y) changes`

Output distribution changes

> *Example: Seasonal disease prevalence changes*

Concept Drift

`P(Y|X) changes`

Relationship between input and output changes

> *Example: Treatment protocols evolve, outcomes change*

Feature Drift

`Individual features change`

Specific feature distributions shift

> *Example: Patient demographics shift over time*

Detection Methods

- KS Test (Kolmogorov-Smirnov)
- Chi-Square Test
- PSI (Population Stability Index)
- Wasserstein Distance
- KL Divergence

Logging Frameworks

Structured logging captures detailed information about model predictions and system behavior

Structured Logs

JSON format with timestamps, request IDs, user IDs, predictions

Log Levels

DEBUG, INFO, WARNING, ERROR, CRITICAL

Centralized Collection

Aggregate logs from distributed services

Search & Analysis

Query logs for debugging and auditing

🛠 Tools & Platforms

- ELK Stack (Elasticsearch, Logstash, Kibana)
- Splunk
- Datadog
- CloudWatch Logs

Alert Configuration

Define thresholds and rules to notify teams when issues occur

Threshold Alerts

Trigger when metric exceeds/falls below threshold

Anomaly Detection

ML-based alerts for unusual patterns

Alert Routing

Send to appropriate team (ops, data science, clinical)

Escalation Policies

If not acknowledged in X minutes, escalate

🛠 Tools & Platforms

- PagerDuty
- Opsgenie
- Prometheus Alertmanager
- AWS SNS

SLA Management

Service Level Agreements define expected performance and availability

Availability SLA

99.9% uptime = 43 min downtime/month

Latency SLA

P95 inference time < 500ms

Accuracy SLA

Maintain > 95% diagnostic accuracy

Response Time SLA

Acknowledge incidents within 15 minutes

🛠 Tools & Platforms

- Status Pages
- SLA Dashboards
- Automated Reporting

Cost Optimization

Monitor and optimize infrastructure costs without sacrificing performance

Right-sizing

Match instance types to actual resource usage

Auto-scaling

Scale down during low-traffic periods

Spot Instances

Use for batch training jobs (up to 90% savings)

Model Optimization

Quantization, pruning to reduce compute needs

🛠 Tools & Platforms

- AWS Cost Explorer
- GCP Cost Management
- Kubecost
- FinOps practices

Part 3

Compliance and Governance

Audit Logging

Comprehensive logging of all actions for regulatory compliance and security

Who

User ID, role, authentication method

What

Action performed (access, modify, delete, predict)

When

Timestamp with timezone

Where

IP address, geographic location, device

Result

Success/failure, error messages

Data

What patient data was accessed/modified

🛠 Tools & Platforms

- CloudTrail
- Azure Monitor
- Audit Trail Systems

Access Control

Role-Based Access Control (RBAC) ensures only authorized users access resources

Principle of Least Privilege

Grant minimum permissions needed

Role Definitions

Data Scientist, ML Engineer, Clinician, Auditor

Multi-Factor Authentication

Require MFA for production access

Session Management

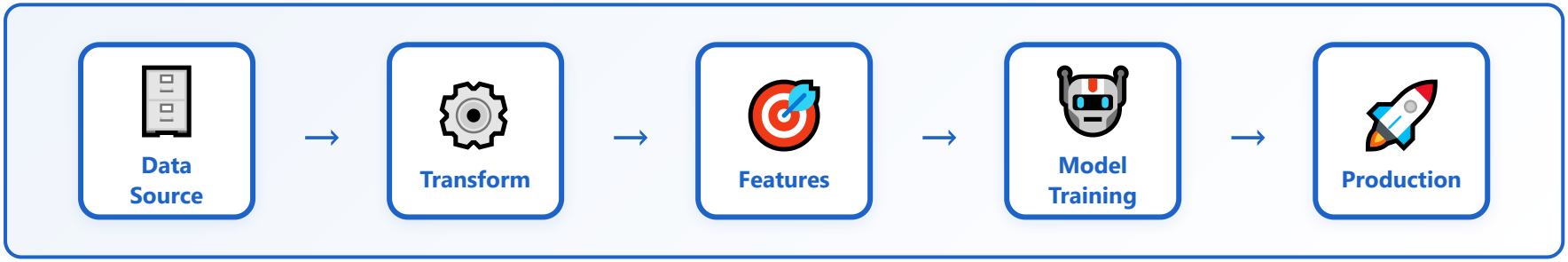
Timeout inactive sessions, logging

🛠 Tools & Platforms

AWS IAM | Azure AD | OAuth 2.0 | SAML

Data Lineage

Track the complete history and flow of data through ML pipelines



Data Provenance

Where did training data originate?

Transformation Chain

All preprocessing and feature engineering steps

Model Lineage

Which data version trained which model version?

Impact Analysis

If data source changes, which models are affected?

Tools & Platforms

MLflow Tracking | DataHub | Apache Atlas | DVC

Model Documentation

Comprehensive documentation including Model Cards for transparency

Model Details

Architecture, training data, hyperparameters

Intended Use

Clinical scenarios where model should be used

Performance

Metrics across demographic groups

Limitations

Known failure modes, out-of-scope cases

Ethical Considerations

Potential biases, fairness concerns

Maintenance

Retraining schedule, monitoring plan

🛠 Tools & Platforms

- Model Cards
- Datasheets for Datasets
- Internal Documentation

Disaster Recovery

Plan and procedures to recover from system failures or disasters

Primary Site

- Production Database
- Live ML Services
- Active Users

FAILOVER

Backup Site

- Replicated Data
- Standby Services
- Ready to Activate

RPO (Recovery Point Objective)

Maximum acceptable data loss (e.g., 1 hour)

RTO (Recovery Time Objective)

Maximum acceptable downtime (e.g., 4 hours)

Tools & Platforms

Multi-region Deployment | Automated Failover | DR Runbooks

Backup Strategies

Regular backups ensure data and models can be restored

3-2-1 Rule

3 copies, 2 different media, 1 offsite

Model Backups

All model versions with metadata

Data Backups

Training data, feature stores, databases

Configuration Backups

Infrastructure code, config files

Automated Testing

Verify backups can be restored

🛠 Tools & Platforms

- AWS S3 + Glacier
- Azure Backup
- Automated Backup Scripts

Security Scanning

Continuous security scanning to identify and fix vulnerabilities

Container Scanning

Scan Docker images for CVEs

Dependency Scanning

Check for vulnerable Python packages

Secret Scanning

Detect exposed API keys, passwords

Penetration Testing

Simulate attacks to find weaknesses

🛠 Tools & Platforms

- Snyk
- Trivy
- GitHub Dependabot
- OWASP ZAP

Performance Optimization

Optimize model inference speed and resource efficiency

Model Quantization

INT8 inference for 4x speedup

Batching

Process multiple requests together

Caching

Cache frequent predictions or features

GPU Optimization

Use TensorRT, ONNX Runtime

Profiling

Identify bottlenecks in pipeline

🛠 Tools & Platforms

- NVIDIA TensorRT
- ONNX Runtime
- PyTorch JIT
- TF Lite

Case Study: Hospital MLOps Deployment

Real-world example of deploying diagnostic imaging AI in hospital

1. **Infrastructure**
   K8s, registry, monitoring
2. **Integration**
   PACS, HL7/FHIR
3. **Validation**
   Clinical study
4. **Production**
   Gradual rollout
5. **Monitoring**
   Track performance

Timeline

6 months from start to production

Team Size

8 engineers + 4 radiologists

Accuracy

96.5% diagnostic accuracy

Results Achieved

30%
Faster Diagnosis

99.5%
Uptime

FDA
Approved

Hands-on: Building an MLOps Pipeline

Step-by-step guide to implement a complete MLOps pipeline

Step 1

Setup: Docker, K8s, MLflow, Prometheus

Step 2

Model Training: Version data, train, log to MLflow

Step 3

CI/CD: GitHub Actions pipeline for testing & deployment

Step 4

Deployment: Deploy to K8s with Helm charts

Step 5

Monitoring: Setup Grafana dashboards, alerts

Step 6

Testing: Send test requests, verify monitoring

🛠 Tools & Platforms

GitHub repo with complete code examples

MLOps Best Practices Checklist

Comprehensive checklist for production ML systems in healthcare

Infrastructure

✓ Containerized, ✓ Auto-scaling, ✓ Multi-region

Versioning

✓ Model registry, ✓ Data versioning, ✓ Feature store

CI/CD

✓ Automated testing, ✓ Gradual rollout, ✓ Rollback plan

Monitoring

✓ Performance tracking, ✓ Drift detection, ✓ Alerts

Compliance

✓ Audit logs, ✓ Access control, ✓ Documentation

Security

✓ Encryption, ✓ Vulnerability scanning, ✓ Backups

🛠 Tools & Platforms

Maturity Model: Level 0 (Manual) → Level 4 (Full MLOps)

Thank You!

🎯 Key Takeaways: Medical MLOps

- ✓ Infrastructure: Containers & K8s
- ✓ Versioning: Models, Data, Features
- ✓ CI/CD: Automated pipelines
- ✓ Monitoring: Performance & Drift
- ✓ Compliance: Audit & Governance
- ✓ Security: Access control & Backups

📖 Additional Resources

- MLOps: Continuous delivery and automation pipelines in ML (Google)
- Practitioners guide to MLOps (Google Cloud)
- ML Engineering for Production (MLOps) Specialization (Coursera)

Ho-min Park

homin.park@ghent.ac.kr | powersimmani@gmail.com