# ☁️ From Code to Cloud

**Generative AI Bootcamp – Week 1, Day 5, Session 1**

*November 21, 2025*

# 🎯 Learning Objectives

- Understand the end-to-end journey: **code → container → cloud**
- Review the **MLOps workflow** and how it supports generative AI applications
- Explore practical deployment paths using **Docker**, **Cloud Run**, and **Vertex AI**

# 🏗️ From Local Code to Deployed AI Service

1. **Develop locally**: write, test, and refactor Python code
2. **Package**: environment consistency and code reusability
3. **Containerize**: build and run anywhere
4. **Deploy and monitor**: scale using cloud infrastructure

# 🧩 MLOps Lifecycle Overview

1. **Development**
2. **Packaging** → consistency and reusability
3. **Containerization** → platform abstraction and scalability
4. **Deployment**
5. **Monitoring & Maintenance**

Each phase introduces automation and reliability to AI delivery.

# 🧱 Development Phase

- Version control with **Git & GitHub**
- Code testing using **PyTest**
- Continuous Integration (CI) for automatic validation
- Style & quality checks with **ruff / flake8**, **black**, **mypy**

# 📦 Packaging & Containerization

- Package code into Python distributions (`pyproject.toml`)
- Use **Docker** to containerize environments
- Build container environment using `requirements.txt`

# 🐳 Docker Essentials Recap

- **Image** = blueprint of environment
- **Container** = running instance
- **Dockerfile** defines build steps
- **Registry** stores and shares images (e.g., Docker Hub, Artifact Registry)

# ☁️ Deployment Targets

| Cloud Platform | Deployment Option | Description |
|---|---|---|
| **Vertex AI** | Cloud Run / Model Garden | Serverless deployment for AI microservices |
| **AWS** | ECS / Lambda | Container orchestration / serverless runtime |
| **Watson X** | AI Workbench | Deployment and monitoring for foundation models |

# ⚙️ Continuous Deployment (CD)

- CI pipeline triggers build & test
- Docker image pushed to registry
- CD step deploys new version automatically
- Use GitHub Actions or Cloud Build for automation

# 🩺 Monitoring & Logging

- Log structured data using `logging` and `rich`
- Track latency, errors, and uptime
- Use Cloud Monitoring / CloudWatch, or Prometheus + Grafana for metrics visualization
- Integrate alerts for model drift or service failure

# 🧠 MLOps in Generative AI Context

- Each component (LLM, vector DB, API) becomes a microservice
- CI/CD ensures coordinated updates
- Containers ensure consistent environments across the stack

## 🌐 Example: Deploying a FastAPI AI Service

1. Build Docker image: `docker build -t ai-service .`
2. Test locally: `docker run -p 8000:8000 ai-service`
3. Push to registry: `docker push gcr.io/.../ai-service`
4. Deploy to Cloud Run: `gcloud run deploy ai-service --image ...`

## 🧩 Integration with Vertex AI & Watson X

- Use APIs to connect to model endpoints
- Deploy services as managed containers
- Combine with **BigQuery** or **FAISS** for RAG-based architectures

# ⚖️ Trade-offs & Design Decisions

| Approach | Pros | Cons |
|---|---|---|
| Docker + Cloud Run | Fast, scalable, serverless | Cold starts |
| Vertex AI Pipelines | Integrated AI ecosystem | Higher setup complexity |
| On-prem (Docker Compose) | Control, cost-effective | Maintenance burden |

## 🧰 Best Practices for Cloud-ready AI Apps

- Externalize configs (`.env`, secrets)
- Write stateless microservices
- Log everything in structured JSON
- Version data and models explicitly

# 💡 Key Takeaways

- MLOps bridges development and deployment
- Containers = reproducibility and scalability
- CI/CD automates quality and rollout
- Cloud platforms abstract infrastructure for speed and reliability