Intro to ML-ops

From Your PC to the Cloud

Sveinung Myhre

ReLU NTNU

September 30, 2025

Outline

1. Introduction
2. Docker Containers
3. Kubernetes
4. GPU Glossary
5. Hands-on Coding
6. Questions & Discussion

Welcome

Welcome to this brief introduction to ML-ops!

Today's mission:
"Acquire the knowledge to put a model from your PC into the cloud"

Disclaimer

Assumptions:

- You know Git basics
- You're familiar with basic Linux commands
- You know what a REST API looks like

If not, that's OK! These are concepts you'll become familiar with throughout your tech career.

Mixed Experience Levels

For some: this will be elementary
For others: entirely new
Everyone: please come with questions!

Plan of the Day

Goal: Acquire the knowledge to put a model from your PC into the cloud

Topics we'll cover:

- Typical ML-ops pipeline overview
- Docker & Containerization
- Kubernetes & Orchestration
- Cloud Deployment strategies
- Hands-on coding session

Typical ML-ops Pipeline

ML-ops workflow: Development → Testing → Deployment → Monitoring

```
Code → Container → Orchestration → Cloud → Production
```

Today we focus on: CI/CD components

- Docker containers
- Kubernetes orchestration
- Auto-scaling clusters
- Reverse proxy architecture

Why I'm Teaching This: Docker

What is Docker?

- Containerization platform
- Package applications with dependencies
- Consistent environments across machines

Why is it important?

- Eliminates "works on my machine" problems
- Essential for modern cloud deployments
- Industry standard for ML model deployment

Evolution: Machine → VM → Container

Physical Machine

- Heavy resource usage
- OS dependency
- Difficult scaling

Virtual Machine

- Full OS overhead
- Better isolation
- Still resource heavy

Container

- Shared kernel
- Lightweight
- Fast startup

Contrast: Docker containers vs Python virtual environments

- venv: Python packages only
- Docker: Entire runtime environment

Essential Docker Commands

Most important commands to know:

- `docker build -t myapp .` - Build image
- `docker run myapp` - Run container
- `docker ps` - List running containers
- `docker images` - List available images
- `docker pull ubuntu` - Download image
- `docker stop <container-id>` - Stop container

Key files:

- `Dockerfile` - Build instructions
- `.dockerignore` - Files to exclude

Docker Registry Services

Container registries store and distribute Docker images

Popular services:

- **Docker Hub** - Free public registry
- **AWS ECR** - Amazon's container registry
- **Google Container Registry**
- **GitHub Container Registry**

Analogy: Docker Hub is to Docker as GitHub is to Git

- Git/Docker: Version control tools
- GitHub/Docker Hub: Cloud hosting services

Hands-on Approach

Time Constraint Strategy

Instead of traditional pair-programming:

Our approach:

1. **Run the code** - Execute provided examples
2. **Read what it does** - Understand the implementation
3. **Explain to each other** - Discuss with peers
4. **Ask AI assistants** - Use ChatGPT/LLMs for clarification

Goal: Maximum learning in minimal time!

Why I'm Teaching This: Kubernetes

What is Kubernetes?

- Container orchestration platform
- Manages containerized applications at scale
- Handles deployment, scaling, and operations

Why is it important?

- Industry standard for container orchestration
- Enables auto-scaling and self-healing
- Essential for production ML deployments

The Scaling Problem

Scenario: Joe and all his friends request a prediction simultaneously

Traditional approach:

- Create a queue?
- Serve each request sequentially?
- Users wait... and wait... and wait...

The Solution

Scale up the number of computers!
But who manages this scaling? **Kubernetes!**

Kubernetes in Action

What Kubernetes provides:

- **Auto-scaling:** Automatically add/remove containers
- **Load balancing:** Distribute requests across instances
- **Self-healing:** Restart failed containers
- **Rolling updates:** Deploy new versions without downtime
- **Service discovery:** Apps find each other automatically

Architecture components:

- Pods (smallest deployable units)
- Services (network access)
- Deployments (manage replicas)
- Ingress (reverse proxy/load balancer)

GPU + Kubernetes: The Future

Why GPUs + Kubernetes matter:

Massive investments:

- $500B on OpenAI Stargate project alone
- McKinsey: $5.2 trillion needed for AI data centers by 2030
- OpenAI scaling to 7,500+ Kubernetes nodes

Career implications:

- Growing demand for cloud infrastructure consultants
- Kubernetes + GPU expertise is highly valued
- Critical skill for modern ML engineering

Source:
`https://openai.com/index/scaling-kubernetes-to-7500-nodes/`

GPU Glossary

Essential GPU terminology for ML:

- **CUDA:** NVIDIA's parallel computing platform
- **Tensor Cores:** Specialized AI acceleration units
- **VRAM:** Video memory for model parameters
- **FP16/FP32:** Floating-point precision levels
- **Multi-GPU:** Training across multiple GPUs
- **GPU Memory:** Often the bottleneck in ML workloads

Excellent resource: Charles Frye's GPU Glossary
`https://modal.com/gpu-glossary`

Credit: Charles Frye for this comprehensive resource

Cloud Deployment Strategy

The complete journey: PC → Container → Orchestration → Cloud

Key components we've covered:

- **Docker:** Containerize your ML model
- **Kubernetes:** Orchestrate at scale
- **Cloud platforms:** AWS, GCP, Azure
- **Auto-scaling:** Handle variable loads

Architecture pattern:

- Load balancer/Reverse proxy
- Multiple container instances
- Auto-scaling based on demand
- Health checks and self-healing

Deployment Architecture

Typical production setup:

```
User Request → Load Balancer → Kubernetes Cluster → ML Model Pods
```

Components:

- **Reverse Proxy:** Nginx, Traefik, or cloud load balancer
- **Ingress Controller:** Routes traffic to services
- **Service Mesh:** Advanced traffic management (optional)
- **Monitoring:** Prometheus, Grafana for observability

Scaling triggers:

- CPU/Memory utilization
- Request queue length
- Custom metrics (inference time, etc.)

Cloud Provider Options

Major cloud platforms for ML deployment:

AWS

- EKS (Kubernetes)
- SageMaker
- EC2 with GPUs
- Lambda (serverless)

Google Cloud

- GKE (Kubernetes)
- Vertex AI
- Compute Engine
- Cloud Run

Azure

- AKS (Kubernetes)
- Azure ML
- Virtual Machines
- Container Instances

Key considerations:

- GPU availability and pricing
- Regional data requirements
- Integration with existing systems

It's Coding Time!

Let's put theory into practice

What we'll implement:

- Containerize an ML model with Docker
- Set up basic Kubernetes deployment
- Test scaling and load balancing
- Deploy to cloud (time permitting)

Coding Session Structure

Our hands-on approach:

1. **Run the provided code examples**
2. **Read and understand what each part does**
3. **Discuss with your neighbor**
4. **Ask questions** - to me, each other, or AI assistants
5. **Experiment** - modify and see what happens

Resources Available

- Code examples in the repository
- Setup scripts for different operating systems
- ChatGPT/Claude for quick explanations
- Peer collaboration encouraged!

Ready to Build?

Let's transform your ML model
from a local script
into a cloud-ready application!

Questions before we start coding?

What We've Covered

Today's journey:

- **Introduction:** The ML-ops pipeline overview
- **Docker:** Containerization fundamentals
- **Kubernetes:** Orchestration and scaling
- **Cloud Deployment:** Production-ready architecture
- **Hands-on Coding:** From theory to practice

Mission accomplished?

"Acquire the knowledge to put a model from your PC into the cloud"

You now have the foundation!

The building blocks are in place. Practice and iteration will make you proficient.

Follow-up Resources

Continue learning:

- **Docker:** Official documentation and tutorials
- **Kubernetes:** `https://kubernetes.io/docs/tutorials/`
- **GPU Glossary:** `https://modal.com/gpu-glossary` (Charles Frye)
- **Cloud platforms:** Provider-specific ML documentation
- **Practice:** Deploy your own models using today's concepts

Next steps:

- Try the setup scripts for your OS
- Experiment with the provided code examples
- Join ML-ops communities and forums
- Consider cloud certifications (AWS, GCP, Azure)

Thank You!

Questions & Discussion

Contact:
sveinung.myhre@example.com

ReLU NTNU
September 30, 2025

References I