Milestone 2 - Submission Checklist

MLOps Course - Module 3

Pre-Submission Verification

Use this checklist to verify your Milestone 2 submission is complete before the deadline. Work through each section systematically.

Deliverables Checklist

Part 1: Multi-Stage Docker Image (4 points)

- ☐ `Dockerfile` exists with multi-stage build (builder + runtime stages)
- ☐ `app/` directory contains inference script (e.g., `app.py`)
- ☐ `app/requirements.txt` or `app/pyproject.toml` with pinned dependencies
- ☐ `docker-compose.yaml` present (optional but recommended)
- ☐ Registry verification screenshot or link saved/documented

Part 2: CI/CD Pipeline (4 points)

- ☐ `.github/workflows/build.yml` exists and is valid YAML
- ☐ Workflow includes test job with pytest
- ☐ Workflow includes build job for Docker image
- ☐ Workflow includes registry authentication step
- ☐ Workflow includes publish step with semantic version tags
- ☐ `README.md` contains CI/CD status badge
- ☐ `README.md` includes image pull/run instructions
- ☐ `tests/test_app.py` exists with unit tests

Part 3: Operations Runbook (2 points)

- ☐ `RUNBOOK.md` exists with all required sections:
  - ☐ Dependency pinning strategy documented
  - ☐ Image optimization with before/after size metrics
  - ☐ Security considerations explained
  - ☐ CI/CD workflow step-by-step explanation
  - ☐ Versioning strategy (semantic versioning)
  - ☐ Troubleshooting guide with common issues

Submission Requirements

- ☐ All files pushed to `module3/milestone2/` directory
- ☐ Final commit tagged as `m2-submission`
- ☐ CI/CD pipeline shows green (passing) status
- ☐ Image accessible in course container registry

Rubric Evidence Map

Use this table to verify you have evidence for each graded criterion:

Criteria	Points	Evidence Location	Verification
Multi-stage Build	2	`Dockerfile` lines showing `FROM ... as builder` and second `FROM`	[ ] Verified
Runtime Environment	1	`requirements.txt` with pinned versions (e.g., `flask==2.3.2`)	[ ] Verified
Registry Integration	1	Screenshot or registry URL with `v1.0.0` style tag	[ ] Verified
Pipeline Functionality	2	`.github/workflows/build.yml` with test→build→push jobs	[ ] Verified
Test Integration	1	`tests/test_app.py` + workflow showing `pytest` step	[ ] Verified
Authentication & Versioning	1	Workflow secrets usage + semantic version in image tag	[ ] Verified
Runbook Quality	1	`RUNBOOK.md` with all 6 required sections	[ ] Verified
Project Organization	0.5	Clean directory structure under `module3/milestone2/`	[ ] Verified
README & Instructions	0.5	`README.md` with badge + quick start guide	[ ] Verified

---

Common Pitfalls

** Docker Optimization Pitfalls**

- Installing dev dependencies in runtime image (use multi-stage to avoid this)
- Not ordering Dockerfile commands from least to most frequently changing

- Using full base images instead of `-slim` or Alpine variants
- Missing `.dockerignore` causing bloated images

** CI/CD Pitfalls**

- Forgetting to authenticate before pushing to registry
- Tests passing locally but failing in CI due to environment differences
- Hardcoded credentials instead of using GitHub Secrets
- Push step running even when tests fail (missing `needs: test`)

** Documentation Pitfalls**

- Outdated documentation that doesn't match current implementation
- Missing exact commands (only descriptions)
- Assuming reader familiarity with your setup
- No CI badge in README

Automated Sanity Checks

Run these commands from your `module3/milestone2/` directory before submitting:

File Existence Checks

```
# Check all required files exist
echo "=== Checking required files ==="
for file in Dockerfile README.md RUNBOOK.md .github/workflows/build.yml; do
  if [ -f "$file" ]; then
    echo "✓ $file exists"
  else
    echo "✗ MISSING: $file"
  fi
done

# Check app directory
if [ -d "app" ]; then
  echo "✓ app/ directory exists"
  ls -la app/
else
  echo "✗ MISSING: app/ directory"
fi

# Check tests directory
if [ -d "tests" ]; then
  echo "✓ tests/ directory exists"
  ls -la tests/
else
  echo "✗ MISSING: tests/ directory"
fi
```

Dockerfile Validation

```
# Verify multi-stage build structure
echo "=== Checking Dockerfile structure ==="
if grep -q "FROM.*as builder" Dockerfile 2>/dev/null || grep -q "FROM.*AS
    builder" Dockerfile 2>/dev/null; then
  echo "✓ Builder stage found"
else
  echo "✗ Missing builder stage (check 'FROM ... as builder')"
fi

# Count FROM statements (should be at least 2 for multi-stage)
FROM_COUNT=$(grep -c "^FROM" Dockerfile 2>/dev/null || echo 0)
if [ "$FROM_COUNT" -ge 2 ]; then
  echo "✓ Multi-stage build detected ($FROM_COUNT stages)"
else
  echo "✗ Not a multi-stage build (only $FROM_COUNT FROM statement)"
fi
```

Dependency Pinning Check

```
# Check for pinned dependencies
echo "=== Checking dependency pinning ==="
if [ -f "app/requirements.txt" ]; then
  UNPINNED=$(grep -E "^[a-zA-Z]" app/requirements.txt | grep -v "==" | grep
      -v ">=" | head -5)
  if [ -z "$UNPINNED" ]; then
    echo "✓ Dependencies appear pinned"
  else
    echo " Potentially unpinned dependencies found:"
    echo "$UNPINNED"
  fi
elif [ -f "app/pyproject.toml" ]; then
  echo " Using pyproject.toml - manually verify pinning"
else
  echo "✗ No requirements.txt or pyproject.toml in app/"
fi
```

Docker Build Test

```
# Test Docker build locally
echo "=== Testing Docker build ==="
docker build -t milestone2-test:local . && echo "✓ Docker build successful"
    || echo "✗ Docker build failed"
```

```
# Check image size
docker images milestone2-test:local --format "Image size: {{.Size}}"
```

Test Execution

```
# Run tests locally
echo "=== Running tests ==="
pip install pytest -q
pytest tests/ -v && echo "✓ All tests passed" || echo "✗ Tests failed"
```

GitHub Actions Workflow Validation

```
# Validate workflow YAML syntax
echo "=== Validating workflow YAML ==="
if command -v python3 &> /dev/null; then
  python3 -c "import yaml;
      yaml.safe_load(open('.github/workflows/build.yml'))" 2>/dev/null && \
    echo "✓ Workflow YAML is valid" || echo "✗ Workflow YAML has syntax
       errors"
else
  echo "□ Python not available - manually check YAML syntax"
fi

# Check for required workflow components
echo "=== Checking workflow components ==="
WORKFLOW=".github/workflows/build.yml"
grep -q "pytest" "$WORKFLOW" && echo "✓ pytest step found" || echo "✗
   Missing pytest step"
grep -q "docker.*build" "$WORKFLOW" && echo "✓ Docker build step found" ||
   echo "✗ Missing Docker build"
grep -q "docker.*push" "$WORKFLOW" && echo "✓ Docker push step found" ||
   echo "✗ Missing Docker push"
grep -q "secrets\." "$WORKFLOW" && echo "✓ Secrets usage found" || echo "
   No secrets referenced"
```

README Validation

```
# Check README contains required elements
echo "=== Checking README ==="
if grep -q "badge" README.md 2>/dev/null || grep -q "!\[" README.md
   2>/dev/null; then
  echo "✓ Badge or image found in README"
else
  echo "✗ No CI badge found in README"
fi
```

```
if grep -q "docker pull\|docker run" README.md 2>/dev/null; then
  echo "✓ Docker pull/run instructions found"
else
  echo " Missing docker pull/run instructions"
fi
```

Git Tag Check

```
# Check for submission tag
echo "=== Checking Git tags ==="
if git tag | grep -q "m2-submission"; then
  echo "✓ m2-submission tag exists"
else
  echo "✗ Missing m2-submission tag (run: git tag m2-submission && git push
      --tags)"
fi
```

Self-Assessment Questions

Before submitting, honestly answer these questions:

Reproducibility

- ☐ **Can someone clone my repo and build the Docker image in under 5 minutes?**
- ☐ Have I tested building from a clean clone (not just my local cache)?
- ☐ Are all dependencies pinned to specific versions?

CI/CD Pipeline

- ☐ **Does my CI badge show green on the main branch?**
- ☐ If I push a commit with failing tests, does the pipeline correctly fail?
- ☐ Does the workflow only push images after tests pass?

Docker Quality

- ☐ Is my final image size reasonable (<500MB for most ML services)?
- ☐ Have I excluded unnecessary files using `.dockerignore`?
- ☐ Does my container run as a non-root user?

Documentation

- ☐ Could a team member understand my system from RUNBOOK.md alone?
- ☐ Are all commands in my documentation copy-pasteable?
- ☐ Have I documented what to do when common issues occur?

Security

- ☐ Are there any hardcoded credentials in my repository?
- ☐ Have I used GitHub Secrets for registry authentication?
- ☐ Does my Docker image use a minimal attack surface?

Versioning

- ☐ Does my registry image use semantic versioning (e.g., `v1.0.0`)?
- ☐ Is my submission tagged as `m2-submission` in Git?

Quick Reference: Expected Repository Structure

```
module3/milestone2/
├── .github/
│   └── workflows/
│       └── build.yml          # CI/CD pipeline
├── app/
│   ├── app.py                 # Main inference script
│   └── requirements.txt       # Pinned dependencies
├── tests/
│   └── test_app.py            # Unit tests
├── .dockerignore              # Files to exclude from build
├── Dockerfile                 # Multi-stage build
├── docker-compose.yaml        # (Optional) Local development
├── README.md                  # Project docs with CI badge
└── RUNBOOK.md                 # Operations documentation
```

Final Submission Checklist

Complete this final checklist immediately before submitting:

- ☐ All automated sanity checks pass
- ☐ All self-assessment questions answered “yes”
- ☐ CI/CD pipeline shows green status
- ☐ `m2-submission` tag pushed to remote
- ☐ Image accessible in course registry with semantic version tag
- ☐ Reviewed rubric evidence map - all criteria verified