Continuous Integration and Continuous Deployment (CI/CD) for ML

Streamlining ML Workflows from Code to Deployment

Lecturer: Vangelis Oden - Technology Lead (Kera)

Assistant: Natalija Mitic - AI/ML Engineer (Kera)

Agenda

- Automated Testing of ML Code and Models
- Why it Matters
- CI/CD Pipelines Specific to ML Workflows
- Github Actions for CI/CD Pipelines
- Integration with Cloud Platforms (AWS, GCP, Azure)
- Best Practices for Model Deployment
- Conclusion
- Q&A

Automated Testing of ML Code

Intuition:

Testing ML Code is different from traditional software due to the model's dynamic nature (data shifts, hyperparameters, etc.). Automated tests ensure the correctness of ML logic, data and results.

Let's understand automated testing..

Why it matters?

- **consistency & quality control** : automated tests ensure consistent performance and high code quality in dynamic ML environments.
- **bug prevention** : regular testing catches issues early, preventing costly errors in production.
- **model integrity** : ensure the model's accuracy and performance metrics are maintained even after changes in the code or data pipeline.

Automated Testing

Intuition:

Tests are a way for us to ensure that something works as intended.
We're incentivized to implement tests and discover sources of error as early in the development cycle as possible so that we can decrease downstream costs and wasted time.
Once we've designed our tests, we can automatically execute them every time we change or add to our codebase.

Automated Testing - Types

- unit tests : tests on individual components that each have a single responsibility (e.g. function that filters a list).
- integration tests : tests on the combined functionality of individual components (e.g. data processing).
- system tests : tests on the design of a system for expected outputs given inputs (e.g. training, inference, etc.). This can include - load tests, security tests, etc.
- acceptance tests : tests to verify that requirements have been met, usually referred to as User Acceptance Testing (UAT).
- regression tests : tests based on errors we've seen before to ensure new changes don't reintroduce them.

Automated Testing - How

We use a framework when composing tests - the Arrange Act Assert methodology

- arrange : set up the different inputs to test on
- act : apply the inputs on the component we want to test
- assert : confirm that we received the expected output

Automated Testing - What to test

When arranging our inputs and asserting our expected outputs, what are some aspects of our inputs and outputs that we should be testing for?

- inputs : data types, format, length, edge cases (min/max, small/large, etc.)
- outputs : data types, formats, exceptions, intermediary and final outputs

Tools : PyTest, SonarQube, etc.

Automated Testing - What to test

```
tests/
├── code/
│   ├── conftest.py
│   ├── test_data.py
│   ├── test_predict.py
│   ├── test_train.py
│   ├── test_tune.py
│   ├── test_utils.py
│   └── utils.py
├── data/
│   ├── conftest.py
│   └── test_dataset.py
└── models/
    ├── conftest.py
    └── test_behavioral.py
```

CI/CD Pipelines Specific to ML Workflows

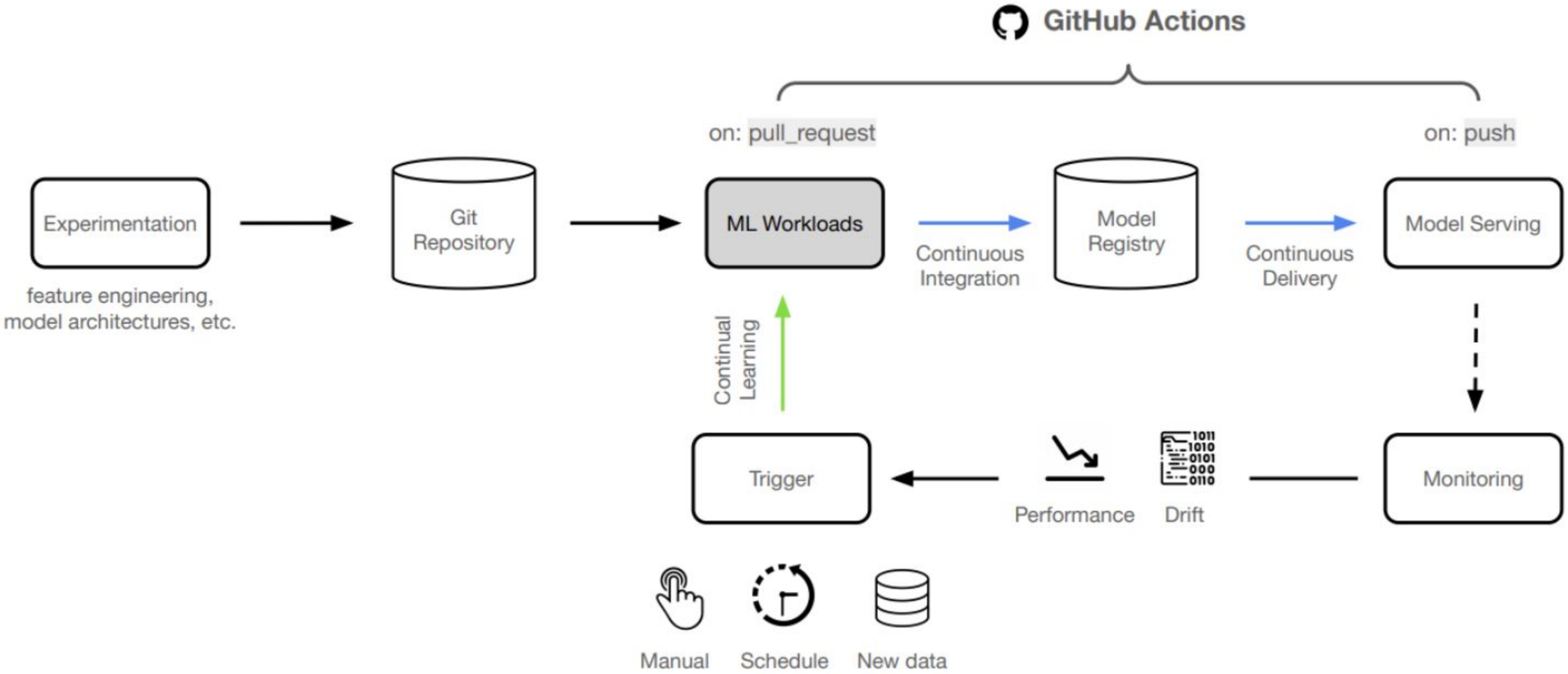
Intuition:

While traditional CI/CD focuses on code, in ML CI/CD pipelines models, data and sometimes experiments also need to be integrated into the pipeline, hence the concept of CI/CT2/CD.

Why ML-Specific CI/CD Pipelines Matter

- **handling data & model complexity** : automating model retraining and data validation is essential for managing the complexity of ML workflows.
- **faster iteration** : automated pipelines allow faster experimentation and feedback cycles, enabling data scientists to focus on improving models instead of manual deployments.
- **consistency** : ensures every change - whether in the code, data, or model - follows the same workflow, reducing the risk of errors in production.

CI/CD Pipelines Specific to ML Workflows



GitHub Actions for CI/CD Pipelines

Intuition :

GitHub Actions is a flexible tool to automate CI/CD workflows directly in GitHub. It allows you to automate code builds, tests, and model deployment processes.

Why it matters:

- Triggers on events like code commits or pull requests.
- Customizable workflows to run automated tests, model training, and deployment.
- Seamless integration with Docker, cloud services, and other CI/CD tools.

GitHub Actions for CI/CD Pipelines

yaml Copy

```yaml
name: ML CI/CD Pipeline
on:
  push:
    branches:
      - main
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Set up Python
        uses: actions/setup-python@v2
      - name: Install dependencies
        run: pip install -r requirements.txt
      - name: Run Tests
        run: pytest
      - name: Train Model
        run: python train.py
      - name: Deploy Model
        run: python deploy.py
```

Integration with Cloud Platforms

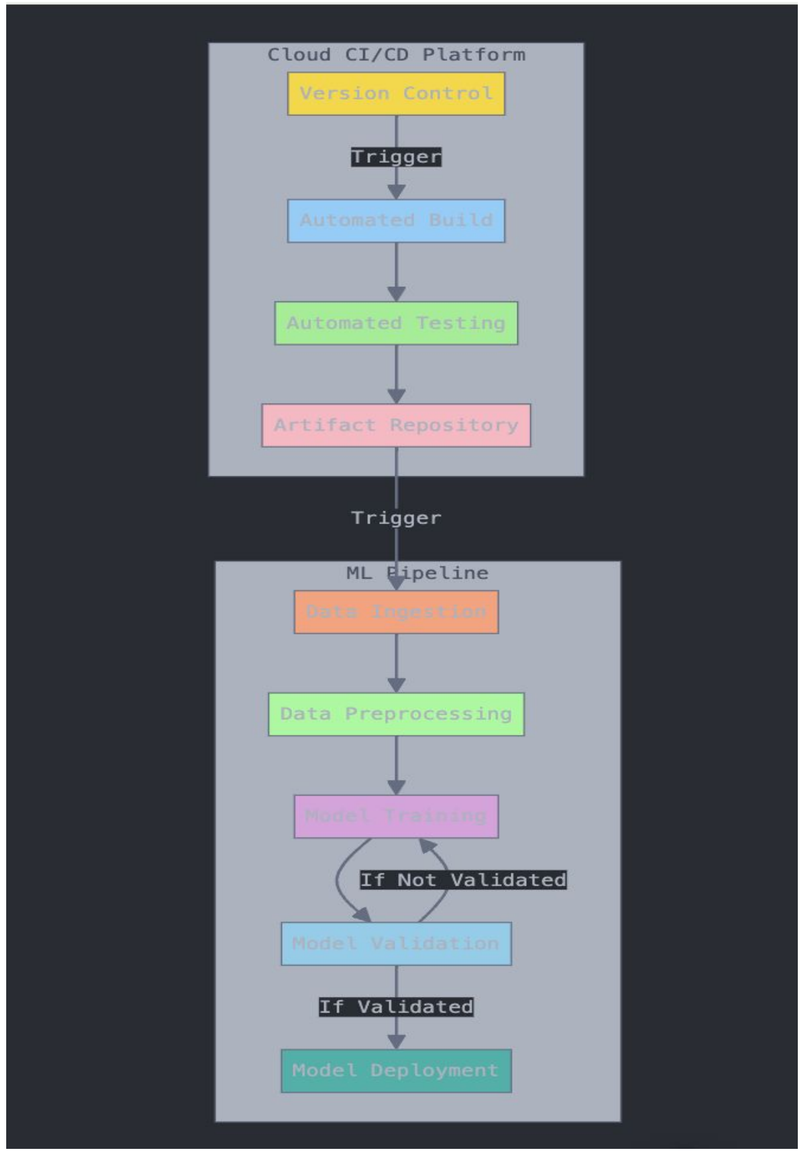
Intuition:

Cloud platforms provide scalable infrastructure to deploy ML models and manage CI/CD. They support automated workflows, serverless models, and large-scale training.

Key Services:

- **AWS :** Sagemaker for model training and deployment, CodePipeline for CI/CD.
- **GCP :** AI Platform for model training and deployment, Cloud Build for CI/CD.
- **Azure :** Machine learning for model training, Azure pipelines for CI/CD workflows.

Integration with Cloud Platforms



Why Cloud Platform Integration Matters

- **scalability :** cloud platforms offer the computing power needed for large-scale ML training and deployment, including handling large datasets.
- **managed infrastructure :** provides managed services for CI/CD pipelines, reducing the need to manage infrastructure.
- **seamless integration :** pre-built integration with common ML tools and frameworks makes it easier to deploy models and manage pipelines efficiently.

Best Practices for CI/CD in ML

- **version control :** track changes to both code and data to ensure reproducibility
- **automated testing :** use unit, integration, and performance tests to catch issues early
- **model monitoring :** continuously monitor models in production for performance degradation or data drift
- **data validation :** validate data before retraining models to avoid “garbage in, garbage out”
- **automated retraining :** automate model retraining when new data is available or performance drops below a threshold.

Conclusion

- CI/CD pipelines for ML automate complex workflows, ensuring faster, more reliable deployments.
- Tools like GitHub Actions simplify automation, while cloud platforms offer scalability and robust infrastructure.
- Adopting best practices ensures consistent quality, reproducibility, and quicker time-to-market for ML models.

Q&A

Thank You!