Nicole Gallo
8/12/2025
D602 - QBN1 Task 1: Business Case Analysis

Create your business case summary by doing the following:

A. Describe the objectives of an MLOps deployment architecture.

The objectives of an MLOps deployment architecture, what the MLOps architecture must achieve, are as follows:

- **Unify the model lifecycle**
  - Track data, code, experiments, models, and promotion stages end-to-end, ensuring every change is documented and reproducible.
- **Support multiple languages**
  - Enable consistent development and deployment for Python, R, and Julia models through standardized environments.
- **Reproducibility**
  - Pin environments, parameters, and dataset versions so results can be reproduced exactly on any system (GeeksforGeeks, n.d.).
- **Two serving modes**
  - Provide both batch processing for scheduled reports and API endpoints for real-time or near-real-time access to predictions.
- **Central, secure artifact storage**
  - Store all artifacts (code, datasets, models, documentation) in a secure, centralized repository with appropriate access controls (IBM, n.d.).
- **Governance & access control**
  - Implement role-based permissions and maintain an audit trail of changes for accountability and compliance (IBM, n.d.).
- **Monitoring & alerting**
  - Track model performance metrics and system health; set automated alerts for drift, failures, or performance degradation (GeeksforGeeks, n.d.).
- **Cost-aware scalability**
  - Start with minimal infrastructure to meet current needs but design for seamless scaling as model usage grows.
- **Collaboration & reuse**
  - Provide templates, shared tools, and best practices to enable consistent work across teams and reduce redundant effort.

B. Identify constraints to implementing an MLOps solution.

The constraints to implement an MLOps solution are as follows:

- **Heterogeneous tech stack**

    - Models exist in Python, R, and Julia without a unified runtime or environment standard, creating integration challenges.
- **Siloed teams**
    - Different departments (e.g., Marketing vs. Procurement) own their own models and processes, limiting collaboration.
- **Inconsistent deployment**
    - Only one model is deployed via an internal API; others are run manually on local machines, often on a quarterly basis.
- **Manual parameter and data updates**
    - Inputs are entered directly into the code or imported via static text files, increasing the risk of errors (GeeksforGeeks, n.d.).
- **Weak storage and lineage**
    - Model files, datasets, and code are stored together in OneDrive without version control or clear lineage tracking.
- **No dedicated MLOps role**
    - Maintenance, tracking, and quality control responsibilities are split across analysts, with only a small current-year budget available.
- **Skeptical leadership**
    - Some senior leaders question the return on investment for MLOps initiatives, potentially limiting support.
- **Intellectual property protection**
    - Expanding into new markets increases the urgency to secure models and datasets as trade secrets (IBM, n.d.).

C. Identify *all* functional and non-functional requirements for the MLOps solution you propose.

All the function and non-functional requirements for the MLOps solution are as follows:

Functional

- **Source control for code/config**
    - Use Git (instead of OneDrive) as the authoritative system of record for all ML code, configuration files, and scripts. This ensures version history, branching, and rollbacks are possible.
- **Data versioning**
    - Implement data version control (or similar) to track datasets alongside code so each model version is linked to its exact input data. Enables reproducibility and rollback to prior datasets (GeeksforGeeks, n.d.).
- **Experiment tracking**
    - Use MLflow to log parameters, metrics, and artifacts for all experiments, ensuring transparency in how models were trained and evaluated.
- **Model registry**
    - Maintain a registry with lifecycle stages (staging/production) and approval gates to control promotions and deployments.
- **Artifact store**

    - Central repository for datasets, trained models, evaluation reports, and supporting files, with proper metadata and lineage tracking.
- **Training/retraining pipelines**
    - Create automated, schedulable, and parameterized workflows to refresh models with updated data or parameters.
- **CI/CD for ML**
    - Integrate continuous integration (tests, linting) and continuous delivery (container builds, deployment triggers) into the ML lifecycle.
- **Multi-language packaging**
    - Containerize environments so Python, R, and Julia models can be deployed consistently.
- **Serving**
    - Support both batch jobs (for reports) and HTTP APIs (for real-time or near-real-time predictions).
- **Monitoring**
    - Track latency, error rates, and data/model drift; set up alerting to respond quickly to performance degradation.
- **RBAC & audit logging**
    - Implement role-based access control and audit logs to track who accessed or changed models and data (IBM, n.d.).
- **Backups & disaster recovery**
    - Regular backups and tested restoration processes for repositories, artifacts, and data stores.

Non-Functional:

- **Security & compliance**
    - Apply encryption in transit and at rest, enforce least-privilege access, and ensure compliance with relevant data regulations (IBM, n.d.).
- **Reliability & availability**
    - Establish SLAs/SLOs (service level agreements, service level objectives) for critical services like model registry, storage, and APIs; design for minimal downtime.
- **Scalability & performance**
    - Enable elastic scaling of compute/storage; define and monitor response time targets for both batch and API predictions.
- **Portability**
    - Use containerization and avoid vendor-specific lock-in to ensure models can run in multiple environments.
- **Maintainability**
    - Standardize templates, code style, documentation, and runbooks to speed onboarding and reduce errors.
- **Cost visibility**
    - Implement resource tagging, monitoring, and phased rollout to manage operational costs.
- **Usability & adoption**

- Provide easy-to-use command line commands (CLI) and notebook examples so analysts across departments can adopt the system quickly.

D. Sources

Other than the WGU internal resources for this course, I've also gained more understanding for terms and concepts using the following external resources:

- GeeksforGeeks. (n.d.). MLOps challenges. https://www.geeksforgeeks.org/machine-learning/mlops-challenges/
- IBM. (n.d.). Role-based access control (RBAC). https://www.ibm.com/think/topics/rbac