How to solve Technical Debt in AI System Development with DevOps

Hasan Yasar

Technical Director, Adjunct Faculty Member

Software Engineering Institute | Carnegie Mellon University

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM21-0805

Outline

- Set the stage:
  - DevOps
  - AI System Development
- The problem: Technical Debt
- DevOps for AI

Modern SW Development: DevOps

DevOps is a set of principles and practices emphasizing collaboration and communication between software development teams and IT operations staff along with acquirers, suppliers, and other stakeholders in the lifecycle of a software system

DevSecOps is a model on integrating the software development and operational process considering security activities: requirements, design, coding, testing , delivery , deployment and incident response.

Mature DevOps practices are constantly testing, deploying and validating that software meets every requirement and allows for fast recovery in the event of a problem. As a result we can easily say,

"DevSecOps is DevOps done right"

DevOps has four Fundamental Principles

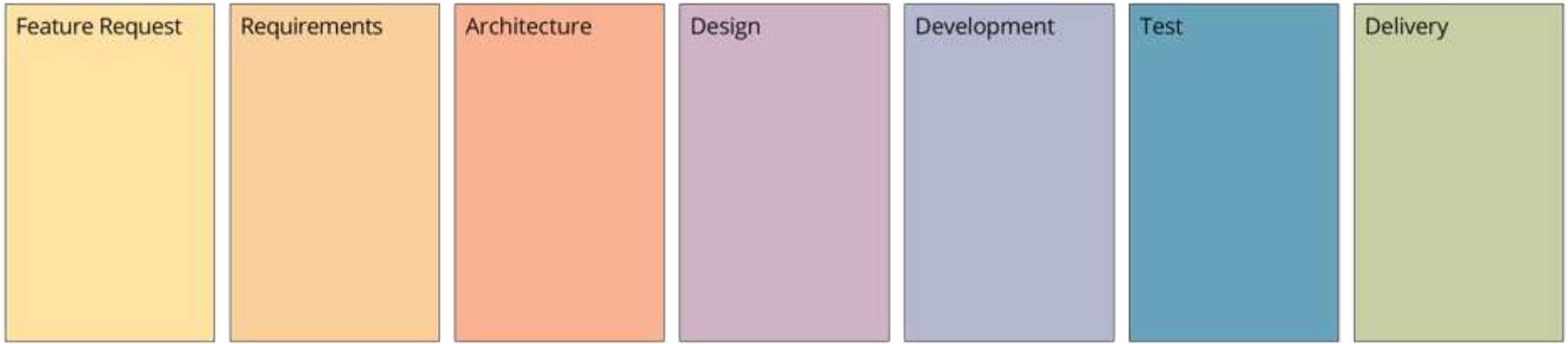
- **Collaboration:** between project team roles

- **Infrastructure as Code:** all assets are versioned, scripted, and shared where possible

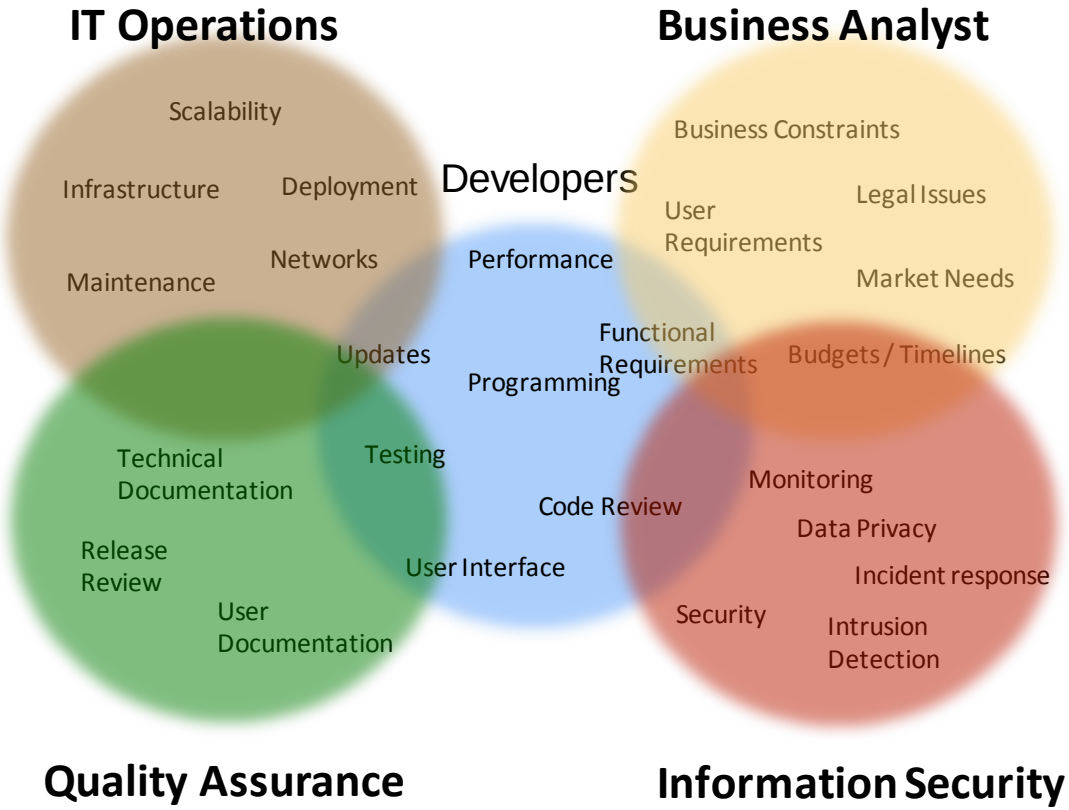
- **Automation:** deployment, testing, provisioning, any manual or human-error-prone process

- **Monitoring:** any metric in the development or operational spaces that can inform priorities, direction, and policy

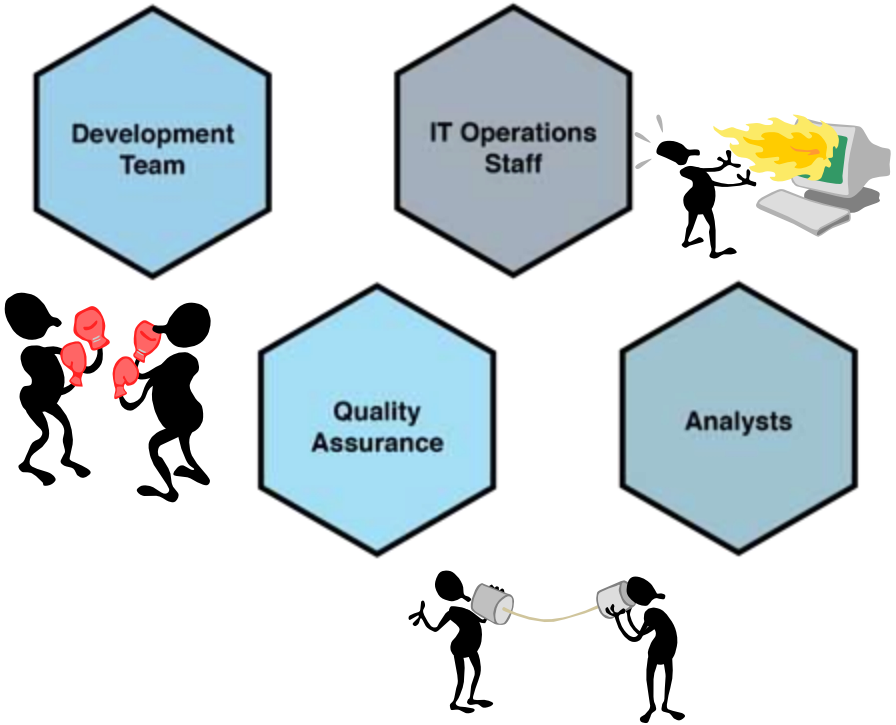
SW Development Phases



Collaboration: *Many stakeholders*

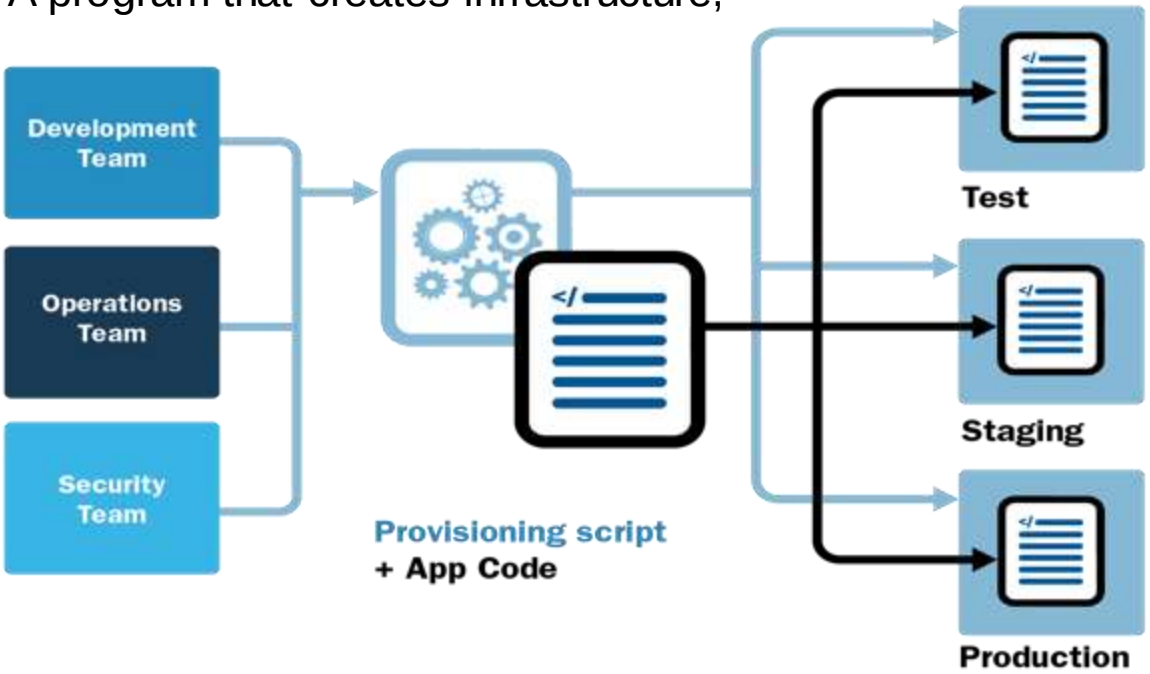


Collaboration: *Silos Inhibit Collaboration and poor communication*



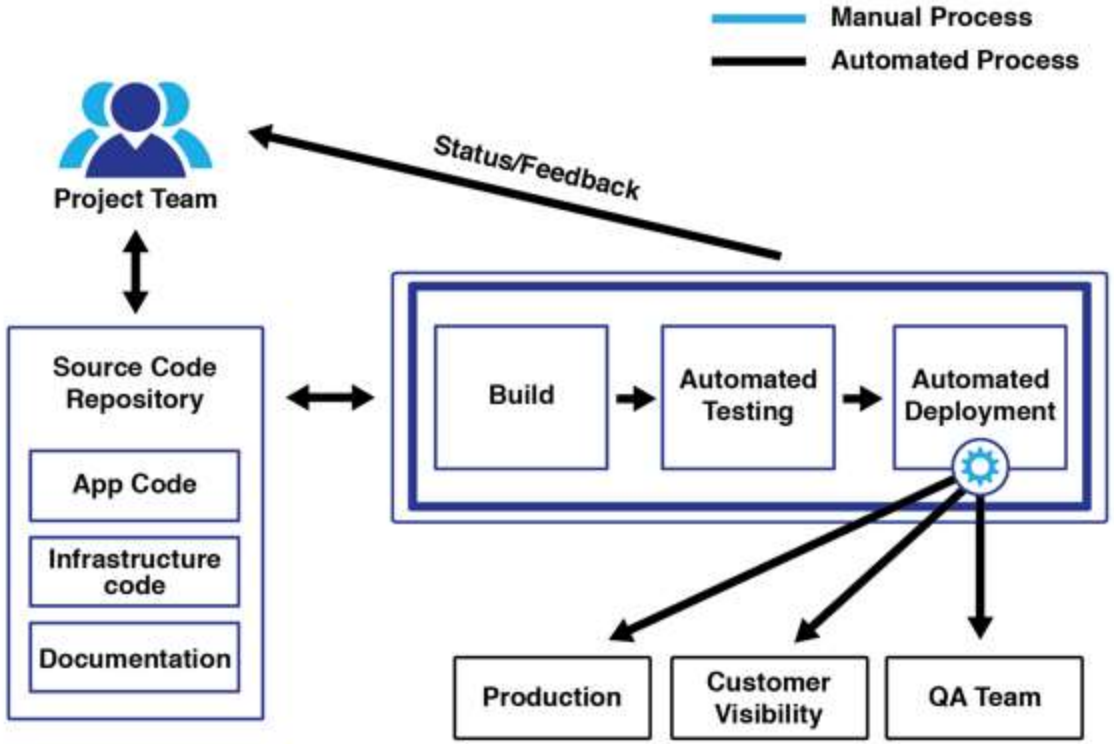
Infrastructure as Code (IaC)

A program that creates infrastructure,



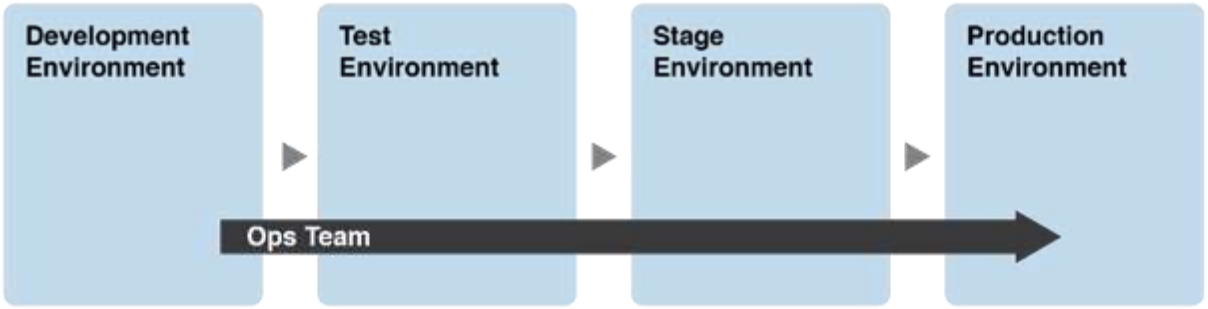
A concretely defined description of the environment is good material for conversation between team members.

Automation : *Continuous Integration (CI)*



Continuous integration is a process that continually merges a system's artifacts, including source code updates and configuration items from all stakeholders on a team, into a shared mainline to build and test the developed system.

Automation : *Continuous Delivery / Deployment (CD)*

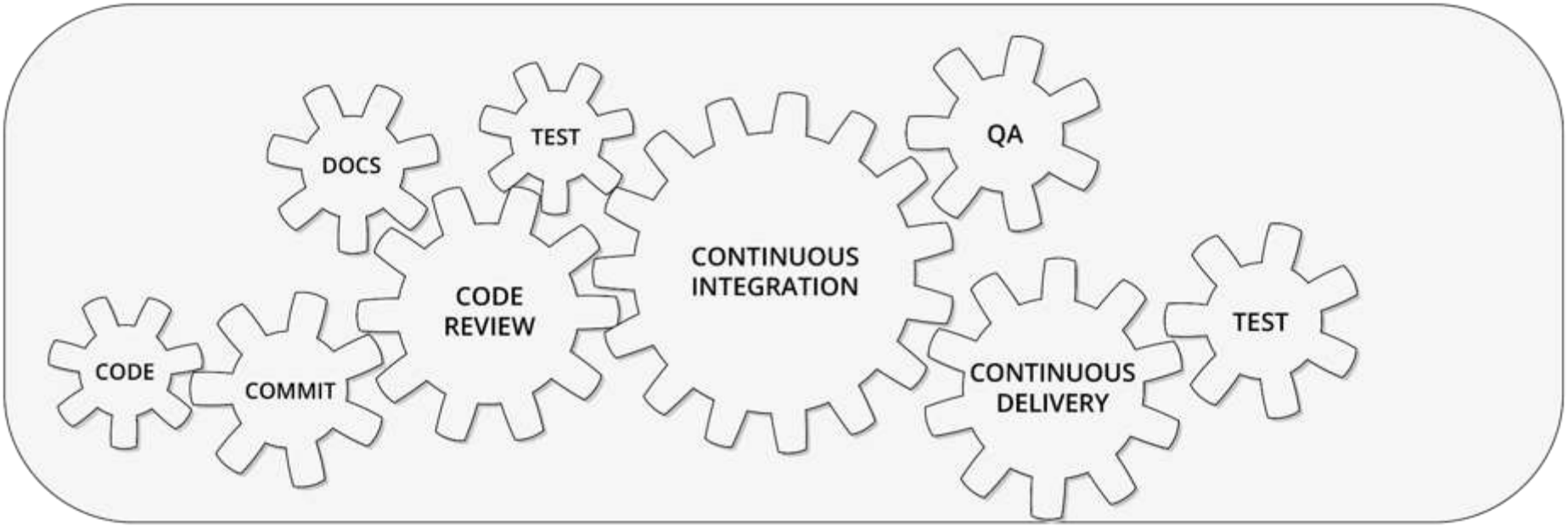


Shift Left Operational Concerns Enforced by Continuous Delivery with parity across various environment

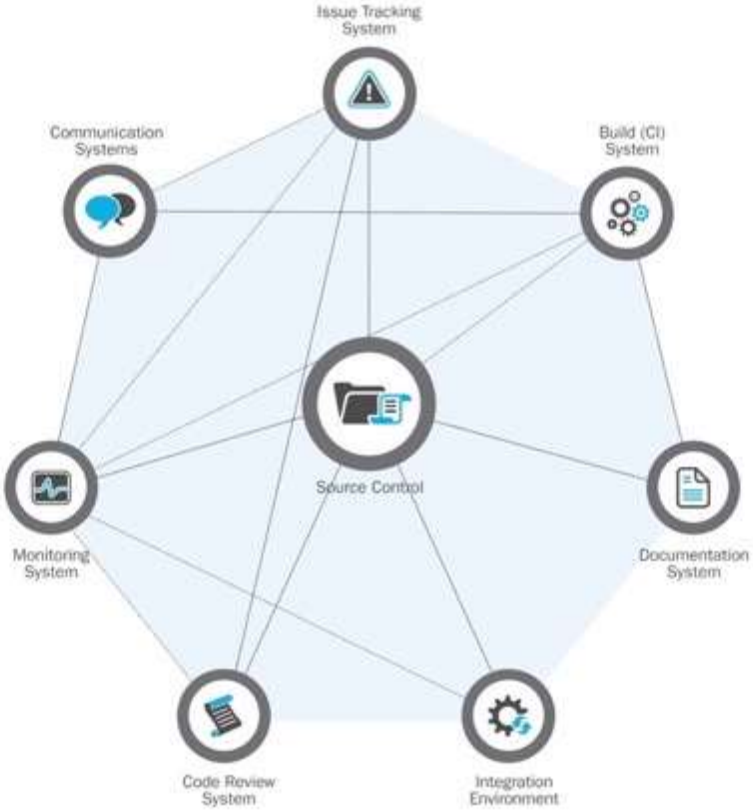
Continuous delivery is a software engineering practice that allows for frequent releases of new software to staging or various test environments through the use of automated testing.

Continuous deployment is the automated process of deploying changes to production by verifying intended features and validations to minimize risk.

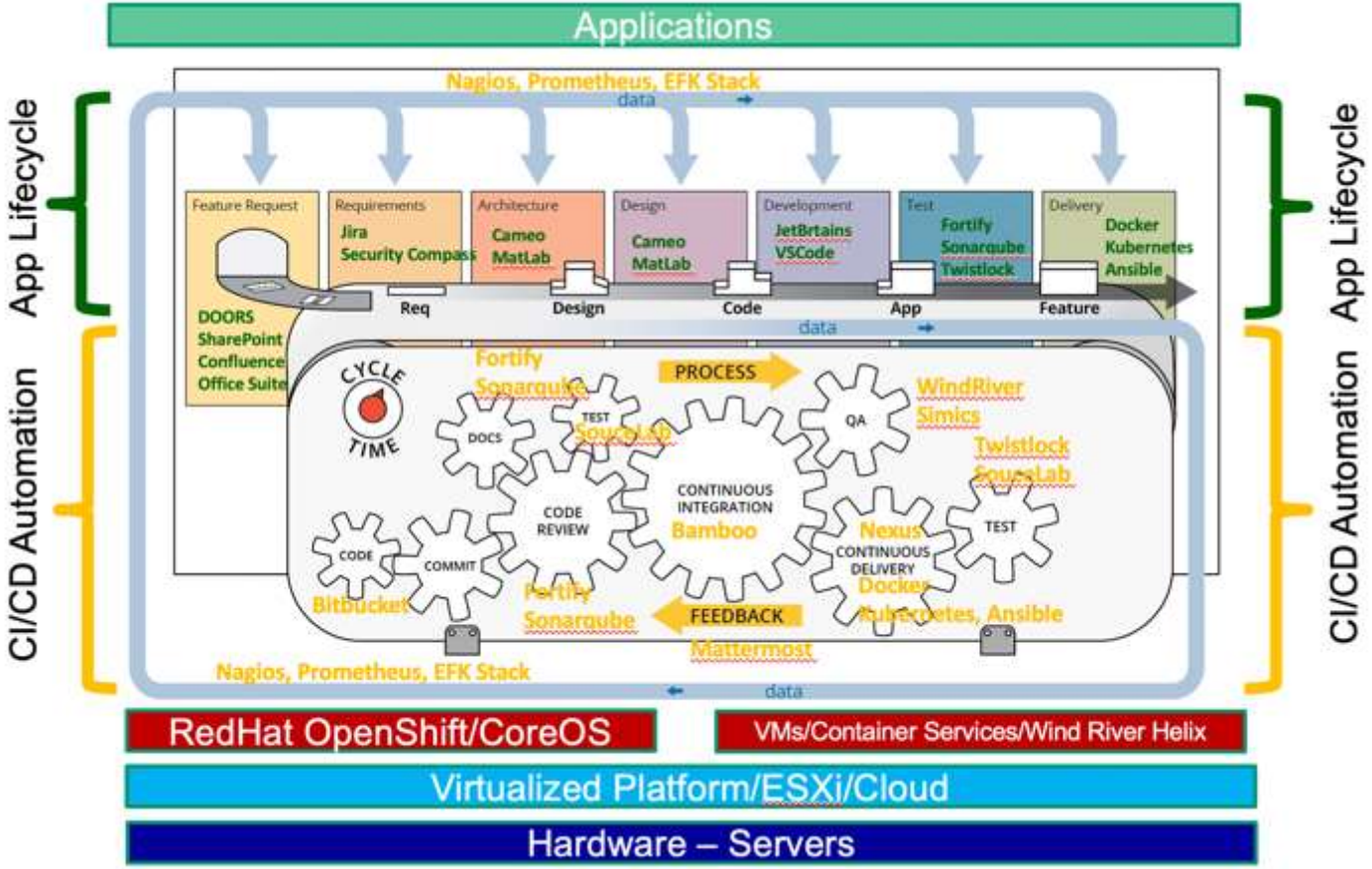
Automation with IaC, CI, CD



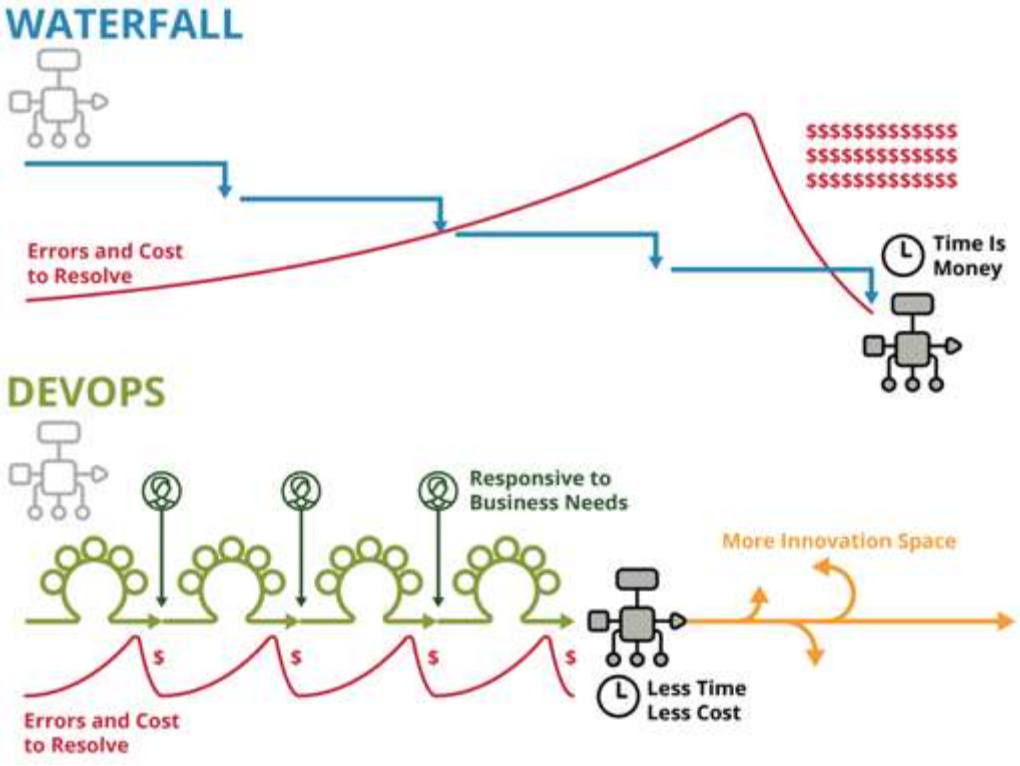
Integrated Development Pipeline - General



Exemplary DevOps tool stack

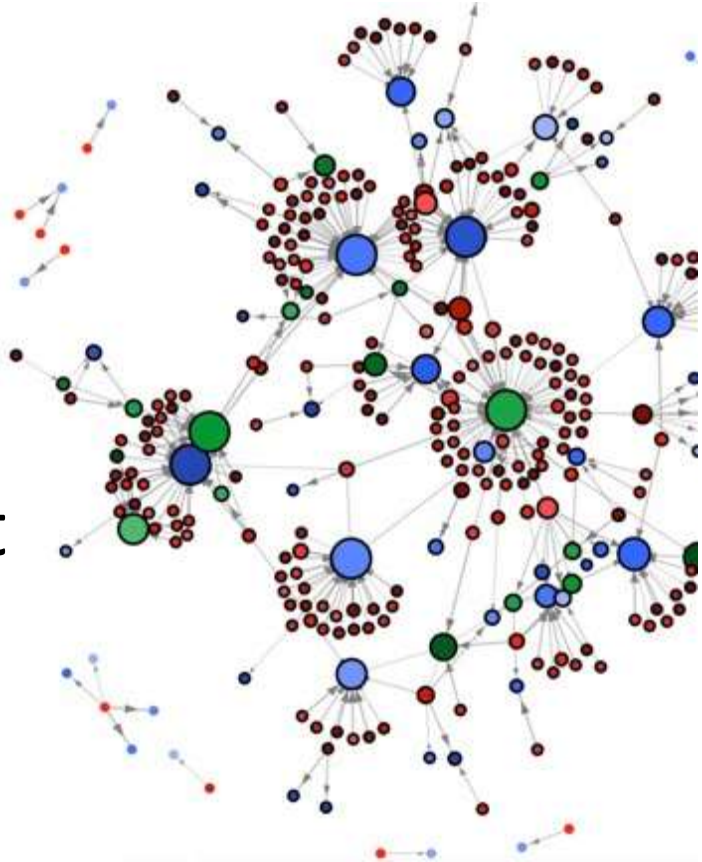


Key Benefits of DevOps

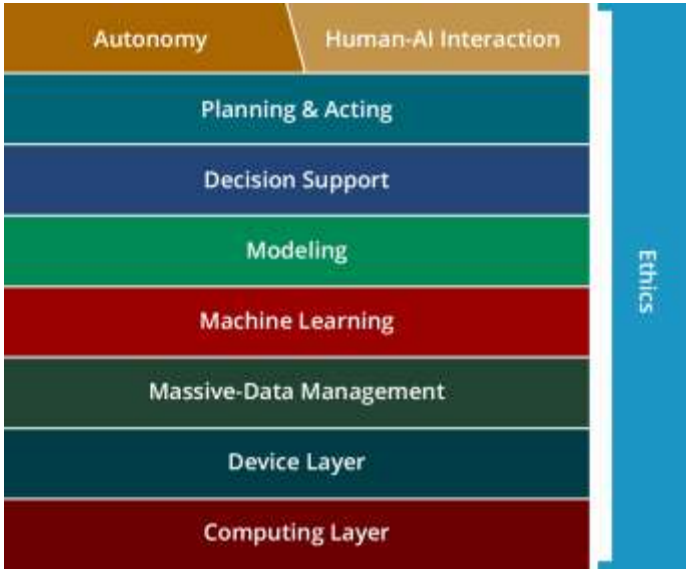


- Reduced errors during deployment
- Reduced time to deploy and resolve discovered errors
- **Repeatable** steps
- **Continuous availability** of pipeline and application
- Increased innovation time
- **Responsiveness** to business needs
- **Traceability** throughout the application lifecycle
- Increased stability and quality
- **Continuous feedback**

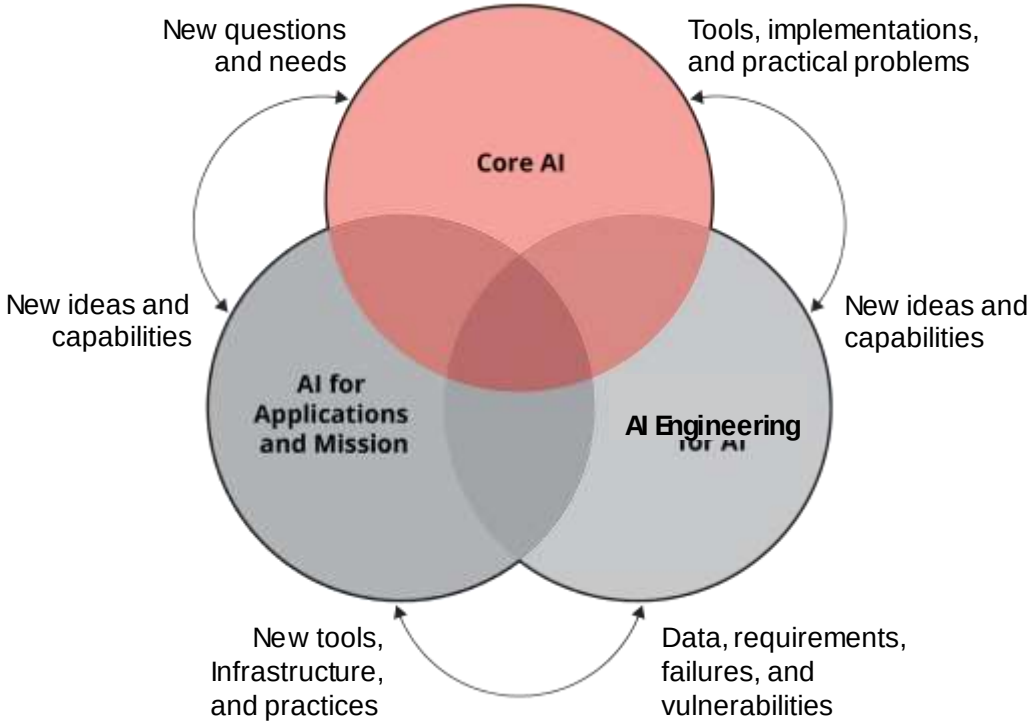
AI System Development



What is AI systems?



CMU AI Stack



Why AI Engineering?

Traditional software and system engineering have a lot to offer in building reliable AI systems, but there are some differences and some gaps.

Many modern AI systems are built using machine learning.

Traditional Software
- Analytical
- Explicit instructions given by programmer
- Reducible and decomposable
- Deterministic

Machine Learning
- Empirical
- Behavior learned from data or experience
- Opaque (and lots of math)
- Unpredictable

"Teaching, not micromanaging" *– Peter Norvig*

An AI Engineering Framework

AI Technologies and Components

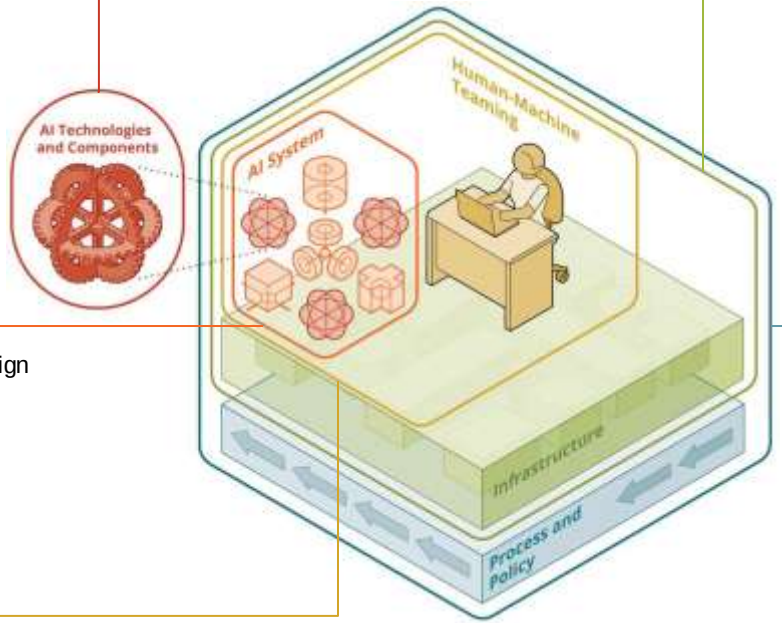
- Knowledge representation
- Modeling and abstraction
- Algorithms
- Scalability and performance
- Robust component design
- Component V&V
- Novelty and Uncertainty

AI System

- Architecture, analysis and design
- Model-based engineering
- Virtual integration
- Robustness and resiliency
- Build security-in Secure AI
- System V&V

Human-Machine Teaming

- Interpretability and explainability
- Human-machine trust
- Machine-human trust
- Co-learning
- Communicating uncertainty
- Data Presentation



Infrastructure

- Data, data management, and data pipelines
- Scalability, performance, and evaluation
- Computational resources and considerations (CSWaP)
- Processing placement and deployment, data locality
- Platform choices and transferability
- Tooling
- Sensors and actuators
- Future architectures

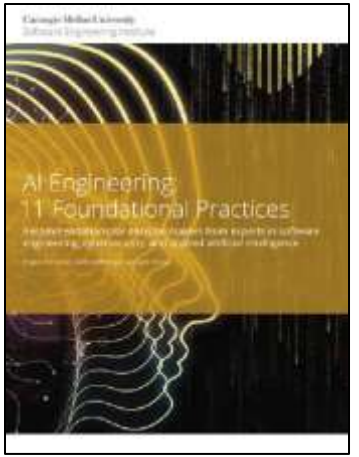
Process and Policy

- AI system acquisition
- AI-ready modern software practices
- ML lifecycle management
- Data lifecycle management
- Standards
- Benchmarks
- Economic considerations
- AI-centric Threat Model
- Risk and resiliency
- Security Coordination
- Ethics
- Bias
- Privacy

Scalable ● Robust and Secure ● Human-centered

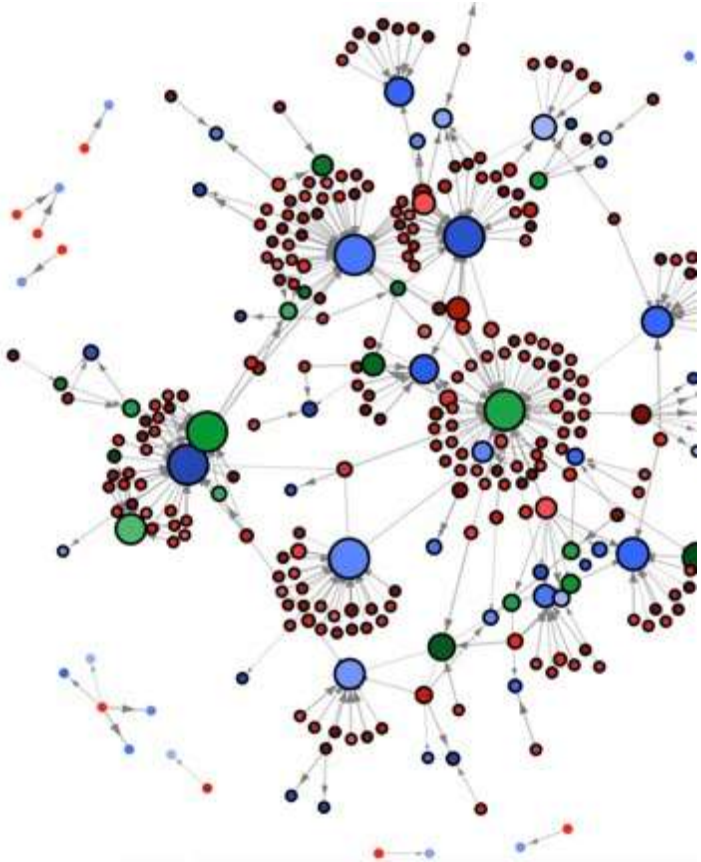
AI Engineering: 11 Foundational Practices

1. Remember that AI is typically built with software.
2. Ensure you need AI for your problem.
3. Build the right team.
4. Do not let data eat your lunch.
5. Choose algorithms based on what you need.
6. Define checkpoints for recovery, traceability, and decision justification.
7. Incorporate user feedback for validation and evolution.
8. Develop a threshold for comfort with ambiguity.
9. Implement loosely coupled solutions.
10. Do not underestimate time and resources.
11. Account for organizational and societal values.



Adapted from:
Ipek Ozkaya, Andrew Mellinger, and Angela Horneman
CMU Software Engineering Institute

The Problem: Technical debt



The hidden technical debts in ML Systems *

- **Boundary erosion**:
  - *it is difficult to enforce strict abstraction boundaries for machine learning systems*
  - *The real world does not fit into tidy encapsulation*
- **Entanglement:** *entangling them and making isolation of improvements impossible*
- **Hidden feedback loops:** *Two systems influence each other indirectly through the world.*
- **Undeclared consumers:** *Expensive at best and dangerous at worst*
- **Data dependencies:** *Input signals are unstable, meaning that they qualitatively or quantitatively change behavior over time*
- **Configuration issues:** *treating configurations an afterthought*
- **ML System Anti-patterns:** *Glue code, pipeline jungles, dead code paths*

*https://papers.nips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf

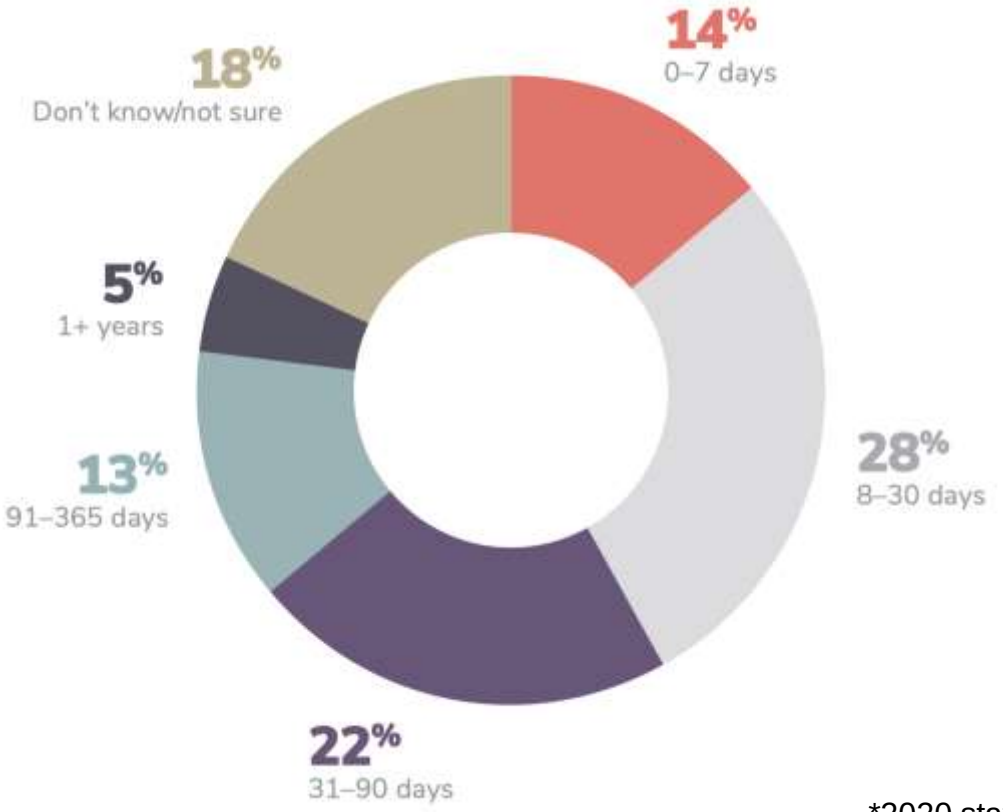
Other areas of technical dept

- **Data Testing Debt:** *Testing of input data is critical to a well-functioning system*
- **Reproducibility Debt**: *Real world testing is different than the scientific experimentation*
- **Process Management Debt**: *single model designed , but mature systems may have dozens or hundreds of models running simultaneously*
- **Cultural Debt**: *There is a hard line between ML research and engineering, but this can be counter-productive for long-term system health*

- **Unable to deploy models**:*50-90% of the models that data science teams have spent months developing, testing, and verifying don't make the leap from the data science team into operations **

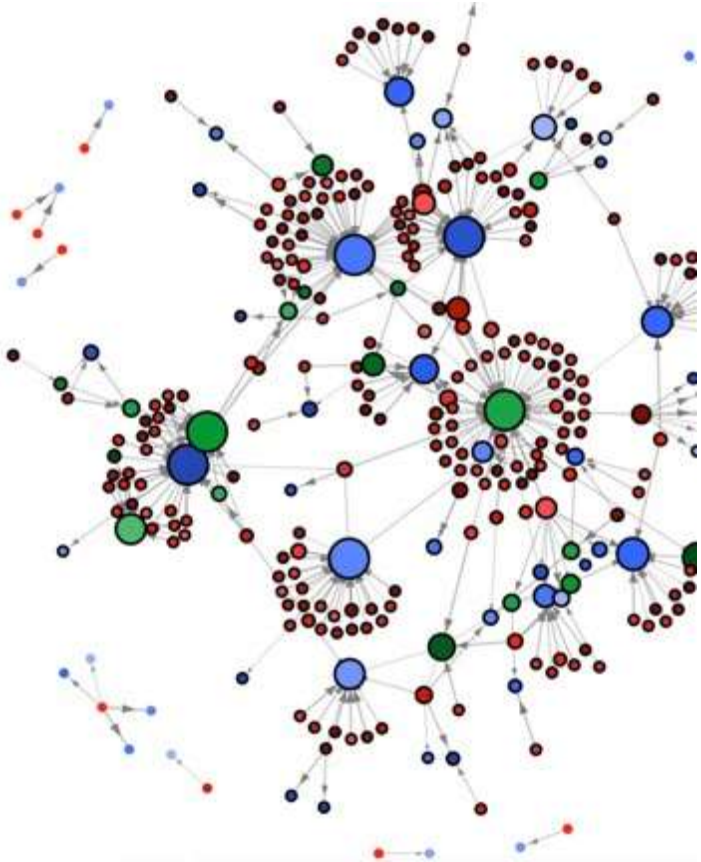
*https://gritdaily.com/why-60-percent-of-machine-learning-projects-are-never-implemented/

Model deployment timeline

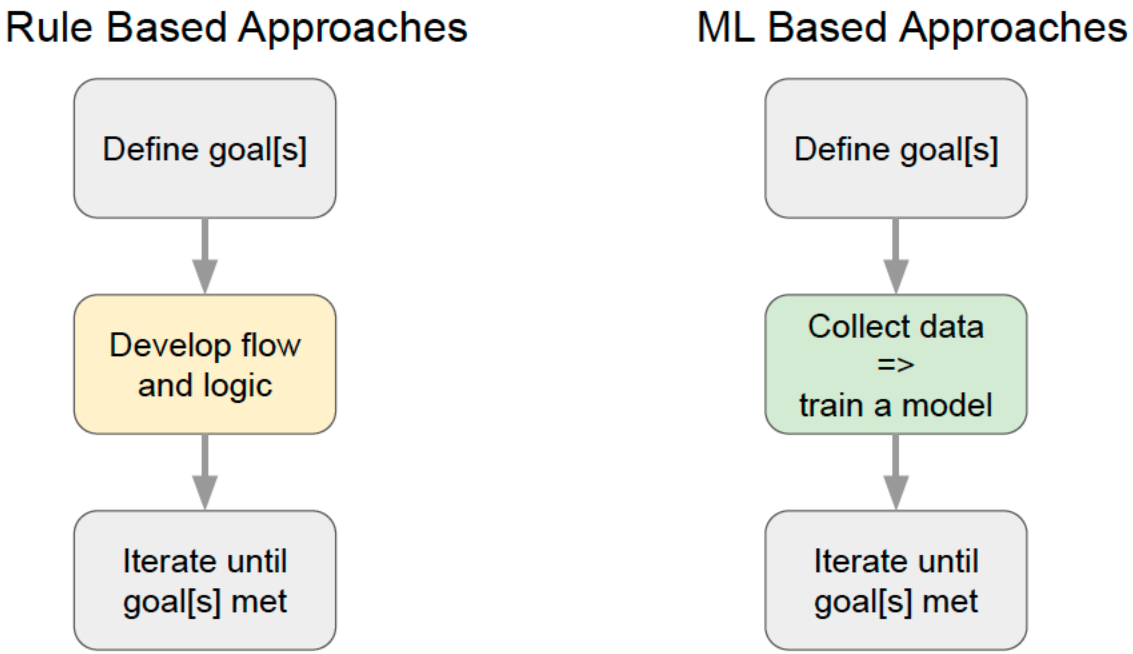


*2020 state of enterprise machine learning

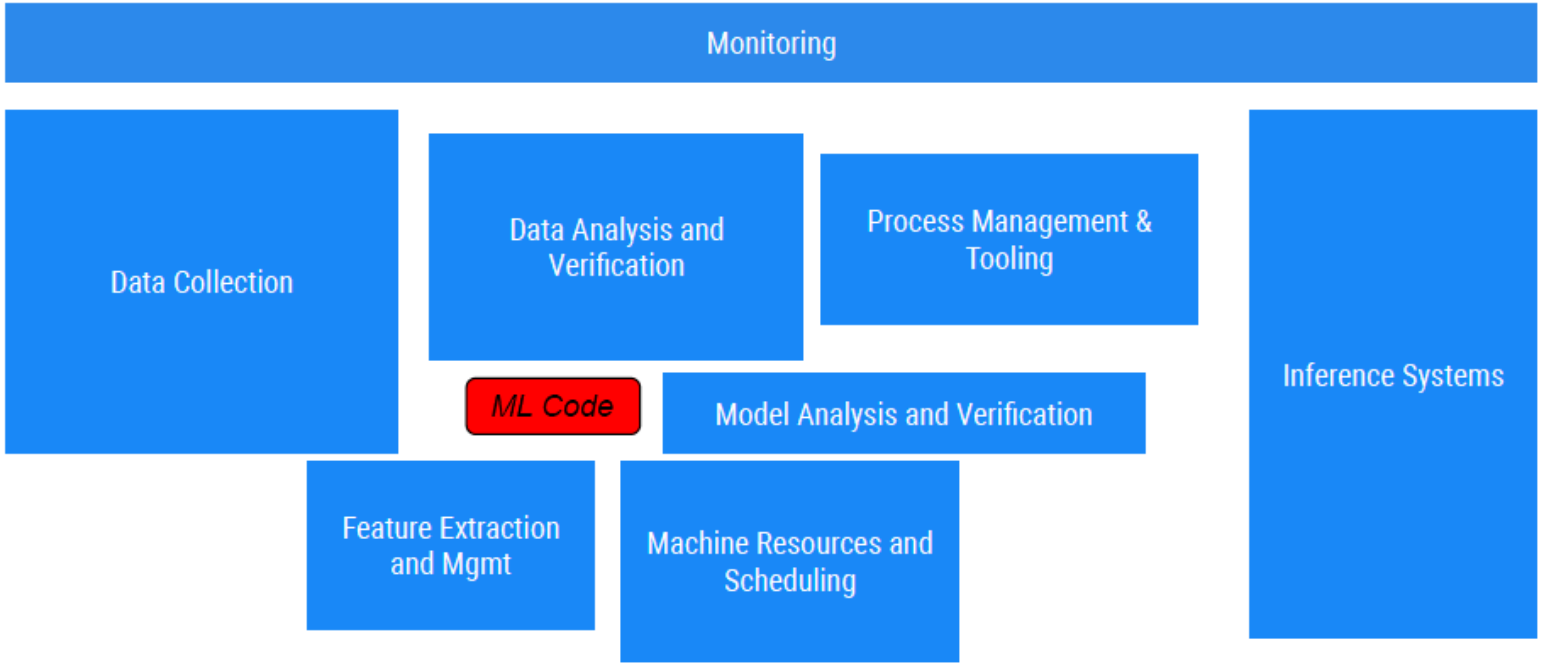
DevOps for AI



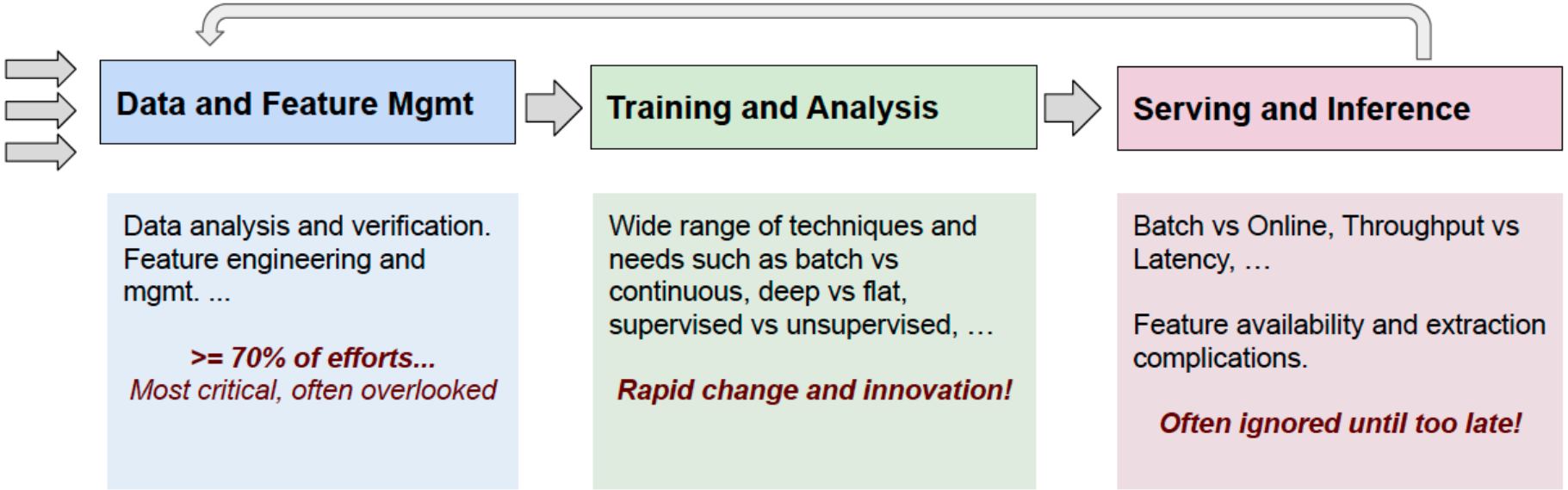
Traditional vs ML Based Approaches?



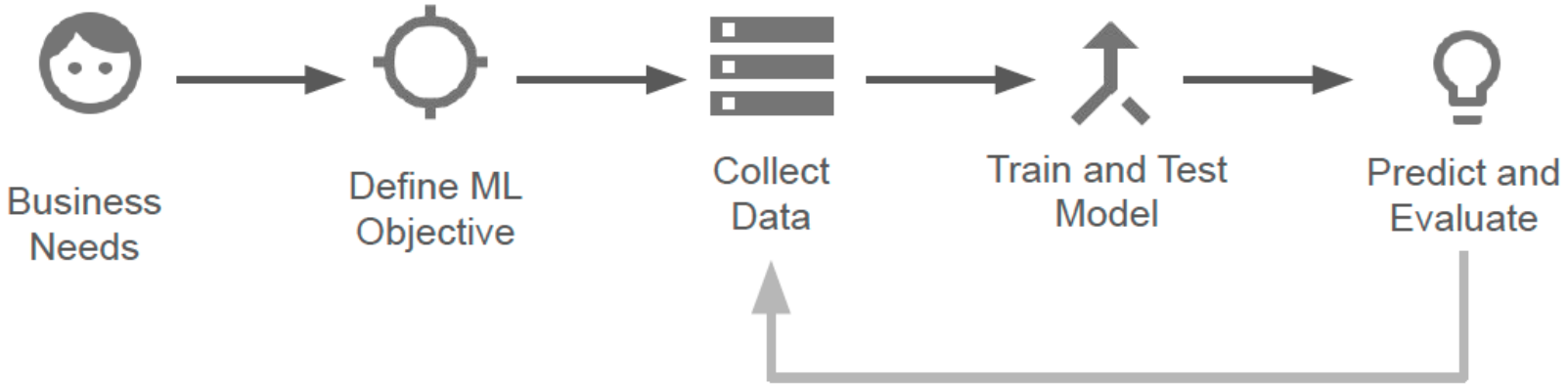
Code in AI System



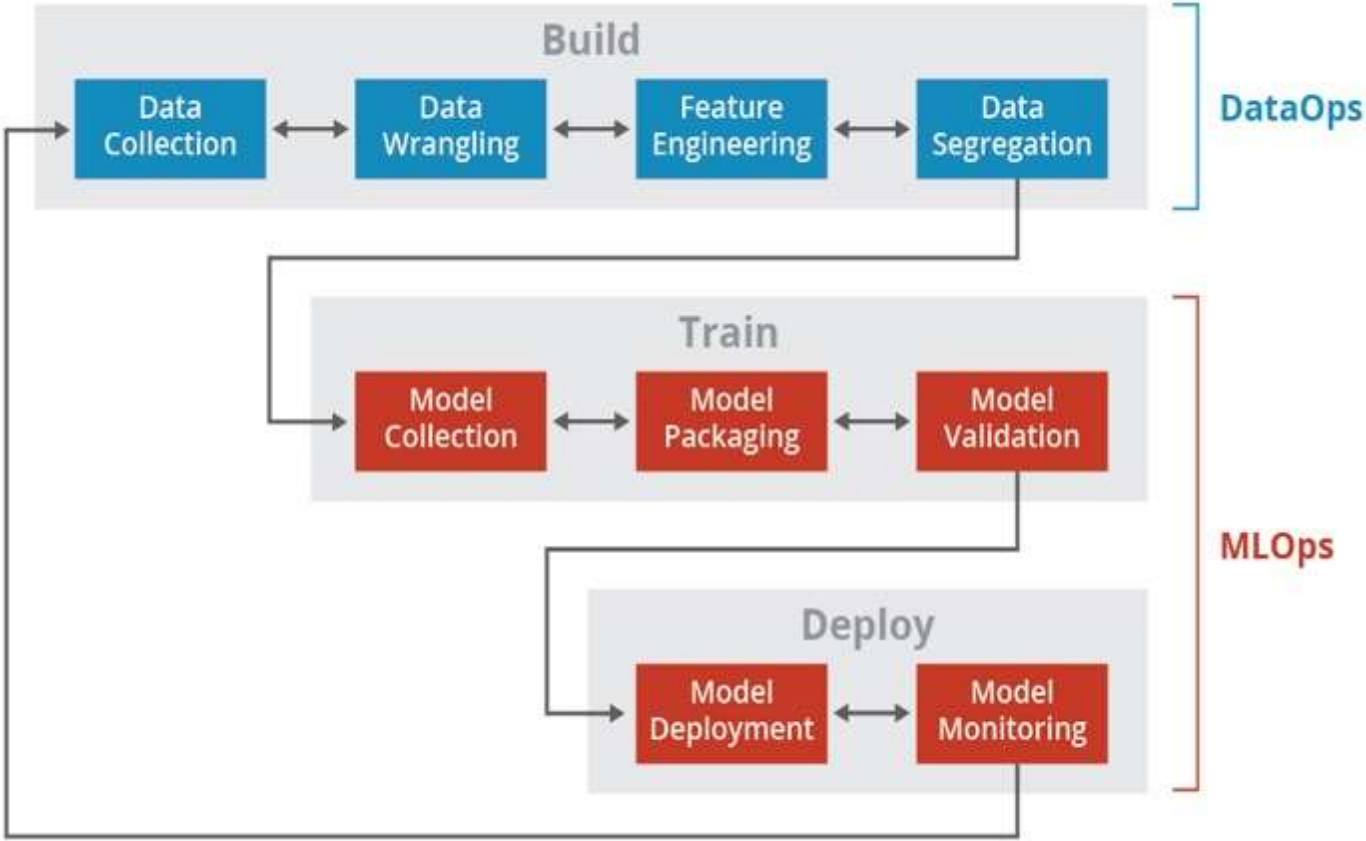
More formally…



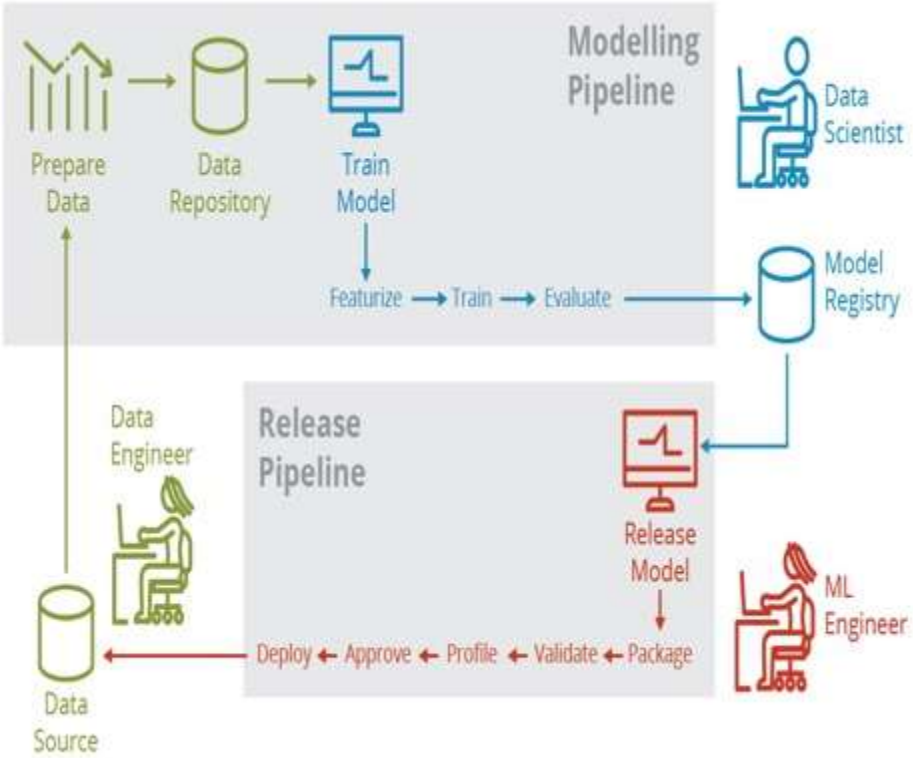
ML Development Cycle



DataOps and MLOps exist



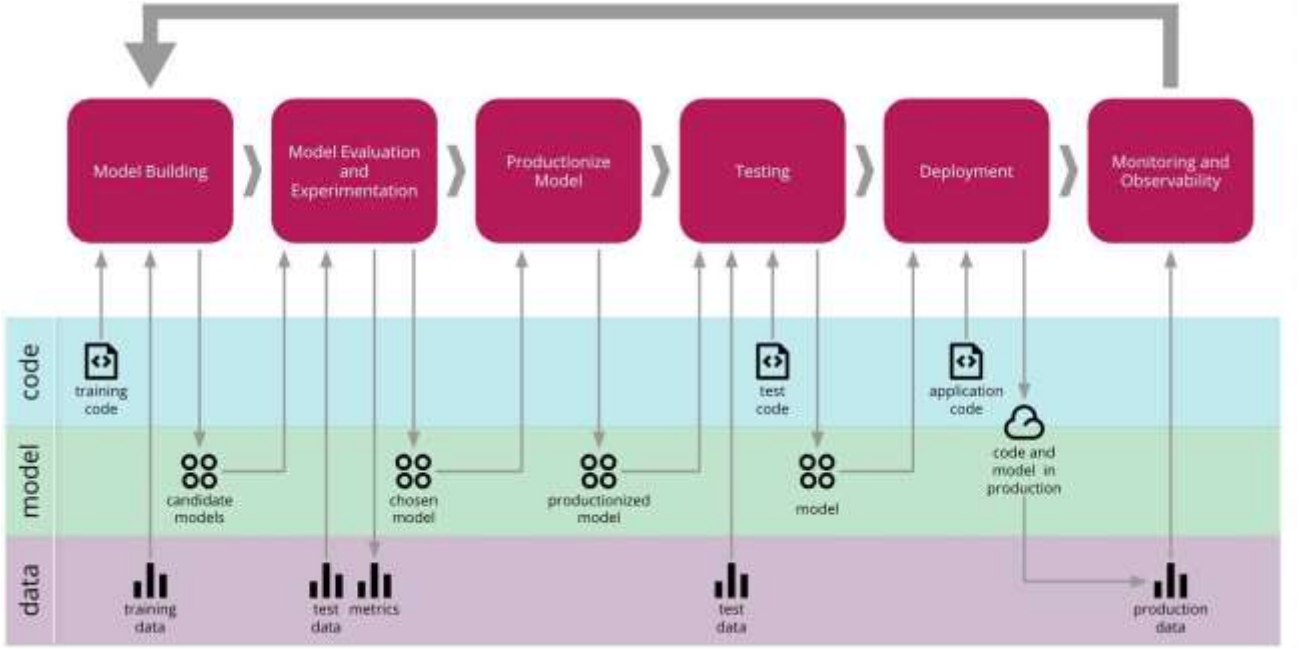
DevOps for AI



Using DevOps concepts and methodologies in every aspect of ML and AI enabled software systems.

- Data curation
- Training data
- Model creation, storage
- Deployment
- Monitoring
- Re-training

How an ML Model *Actually* Comes into Existence and Is Used (in real-world deployed systems)

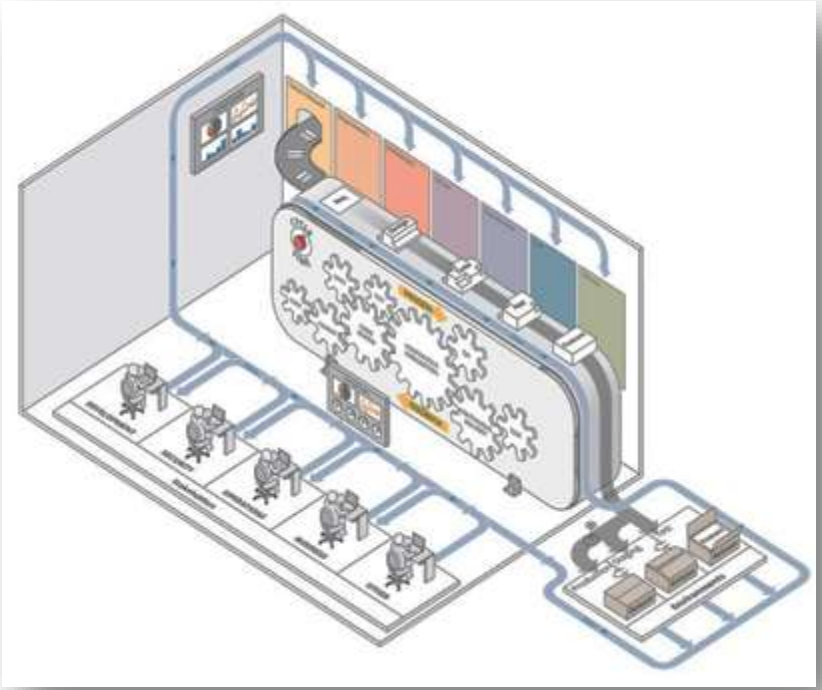


Source: [Continuous Delivery for Machine Learning, Martin Fowler]

Important considerations

- Data must be prepared before model training
- Model release requires operationalization
- Post-deployment monitoring should record all real-world data serving as input to the deployed model
- Team members include
  - Data engineers, Data scientists
  - ML engineers, DevOps
  - Developers
- Model performance
- Deployment strategies
- Model storage and sharing

Necessary DevOps Factory additions



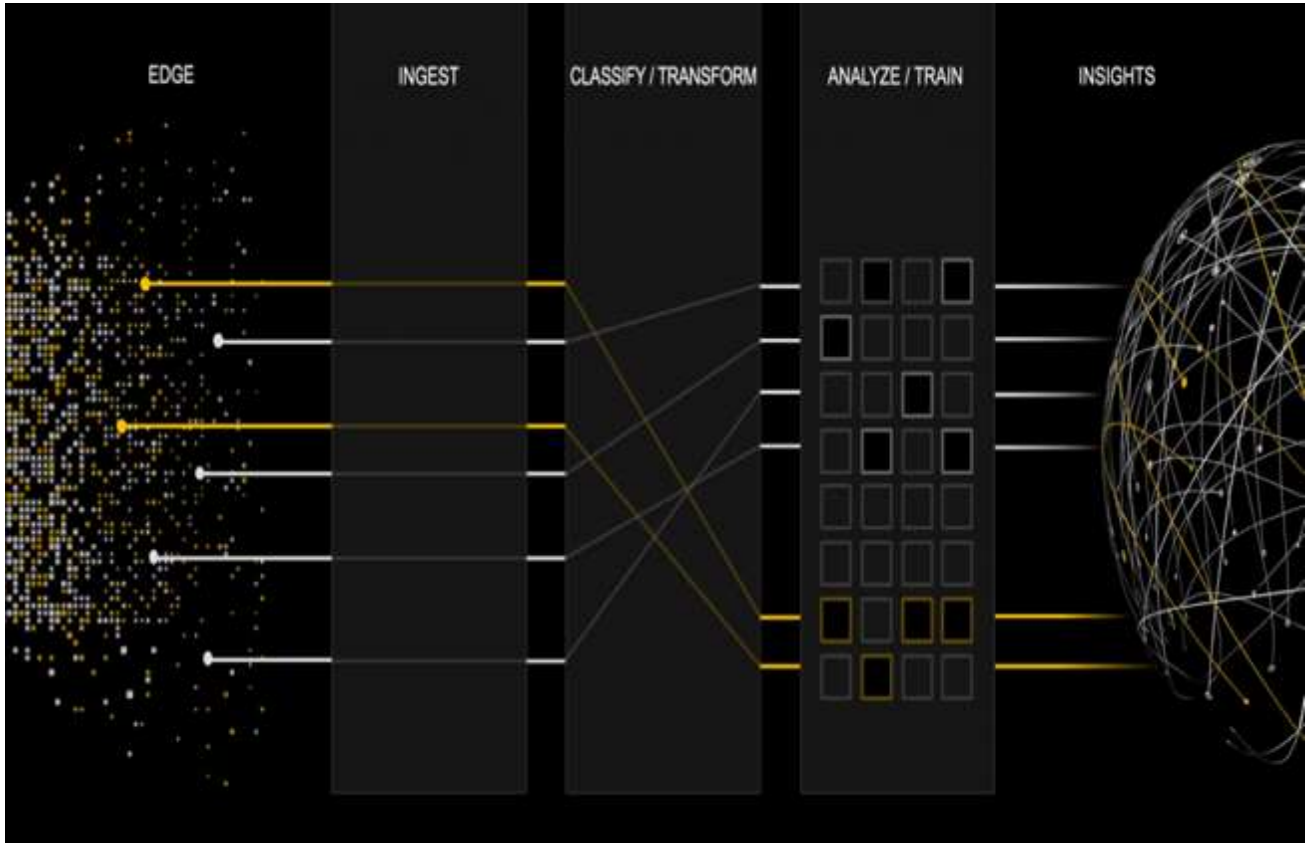
1. Embrace an MLOps culture to facilitate an ML- driven factory
2. Establish a cultural focus on data-driven development to facilitate ML model creation
3. Include data scientists and data engineers in software development teams
4. Automate model deployment via continuous delivery/deployment
5. Establish continuous feedback including model monitoring like *model inputs*, *model outputs and decisions*, *user action and rewards* and *model fairness.*

DevOps for data curation

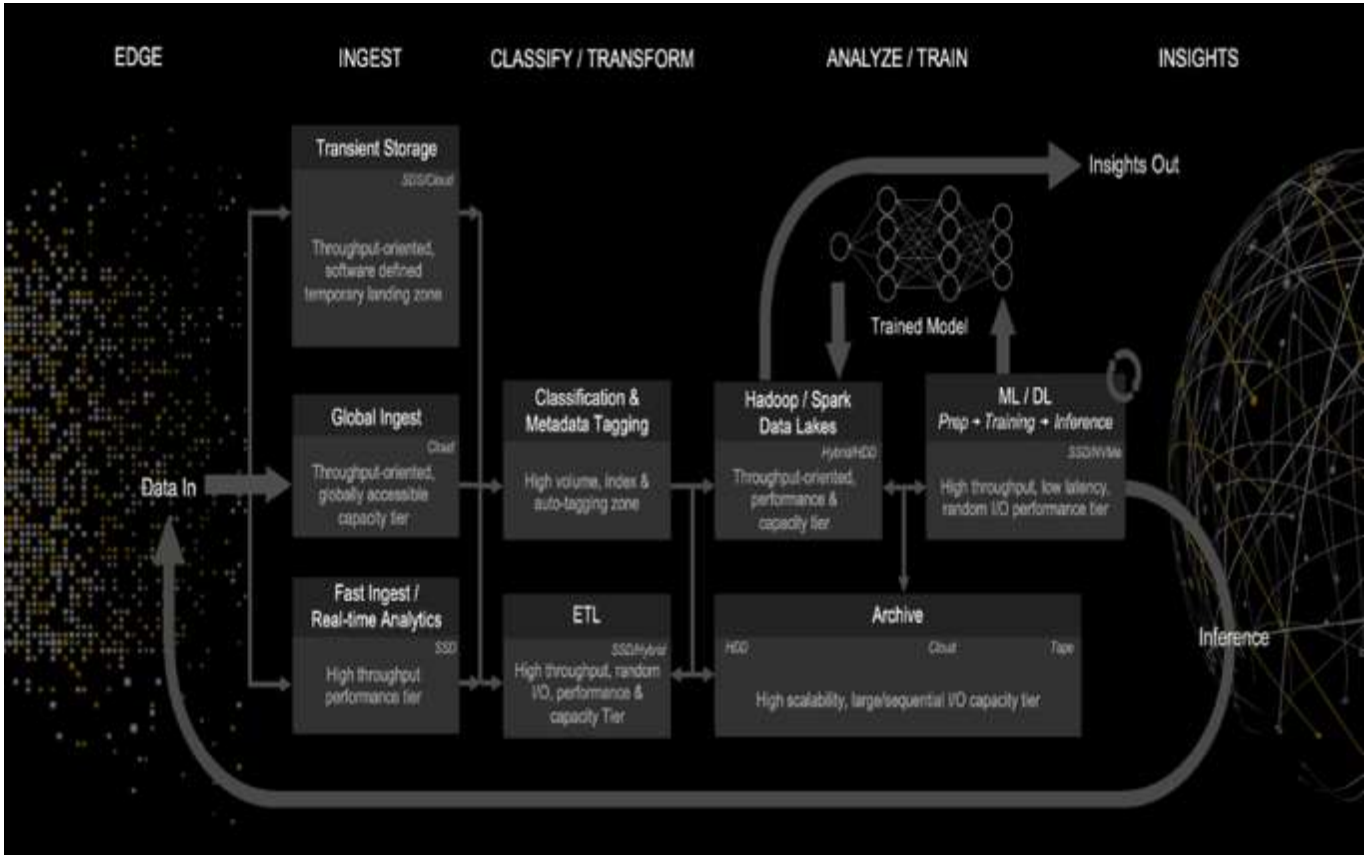
Data is a common critical element of an AI system

- The general process for data processing:
  - Develop business cases
  - Ingest
  - Classify/transform/analyze
  - Insights
  - Availability of Data for DS
  - Validating Data

Data curation



Data curation - workflow



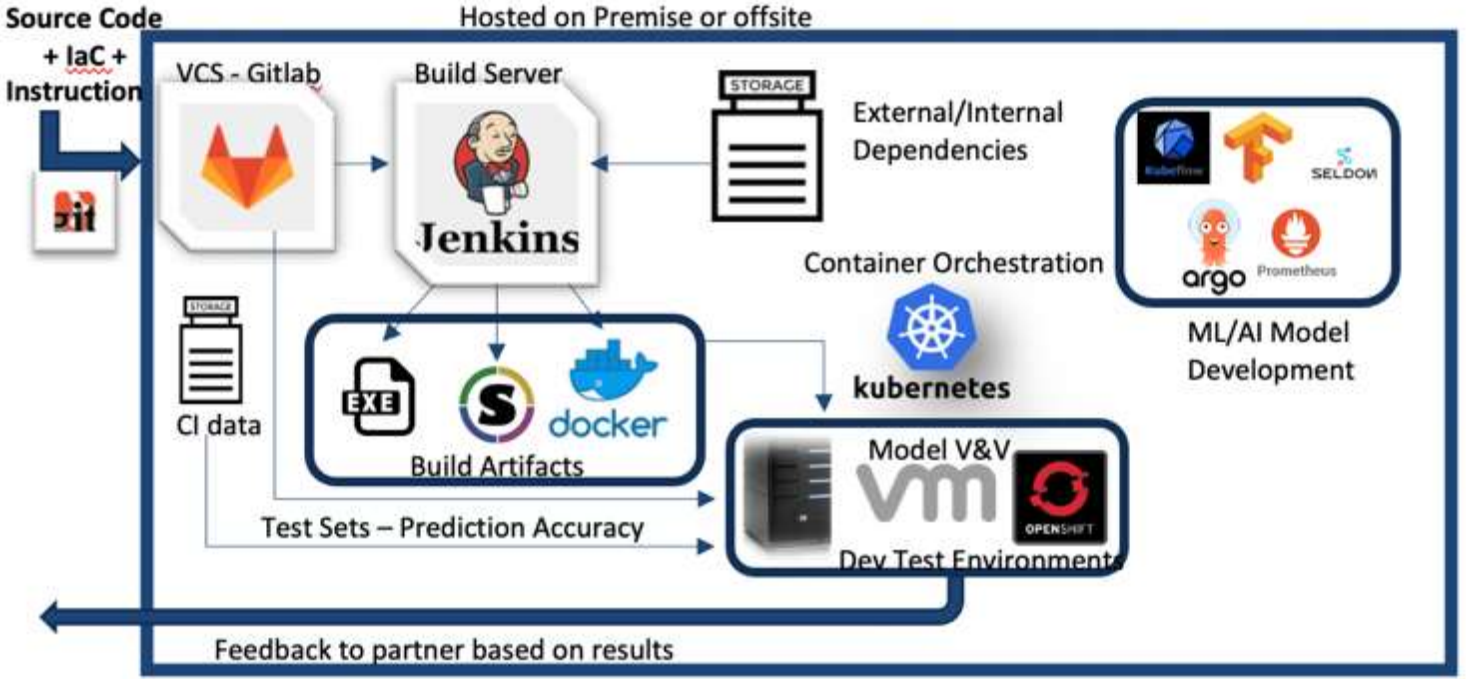
Monitoring deployed AI systems

- Include a return loop of data to the starting point of the pipeline
- Archive all ingress data to the model for future training
- Record and analyze the model's output for functionality and integrity
- Determine if a model requires modification or re-training
- Model inputs: what data, predictions or recommendations
- Model outputs and decisions
- Model interpretability outputs
- Example: using EFK stack for monitoring and observability
  - Elasticsearch: an open source search engine.
  - FluentD: an open source data collector for unified logging layer.
  - Kibana: an open source web UI that makes it easy to explore and visualize the data indexed by Elasticsearch.

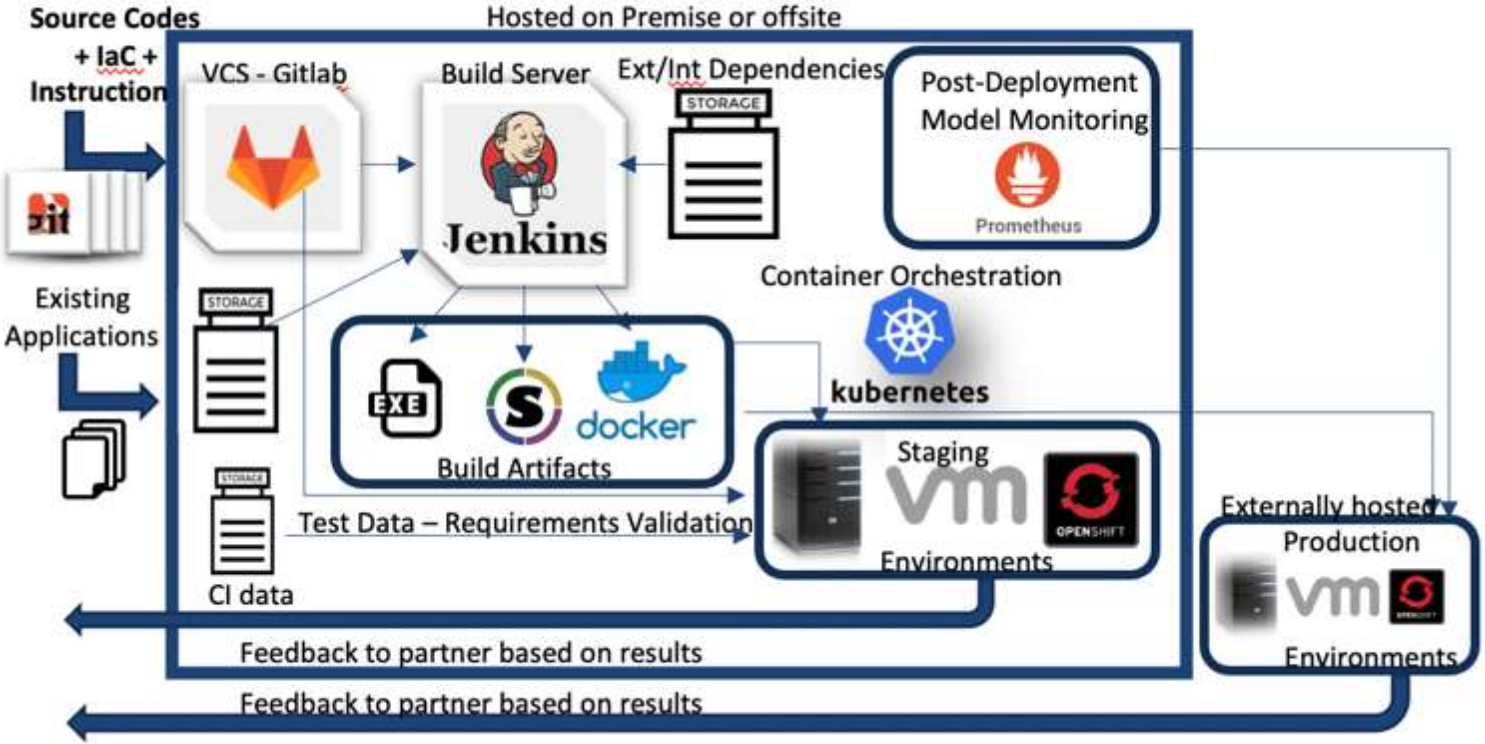
Additional guidance for an AI/ML Pipeline

- Capable of ingesting multiple data types
- Data maintained and versioned
  - Data Version Control (dvc.org)
- Real-time monitoring
- Responsive to changing conditions discovered during monitoring
- Traceability
- Language standardization

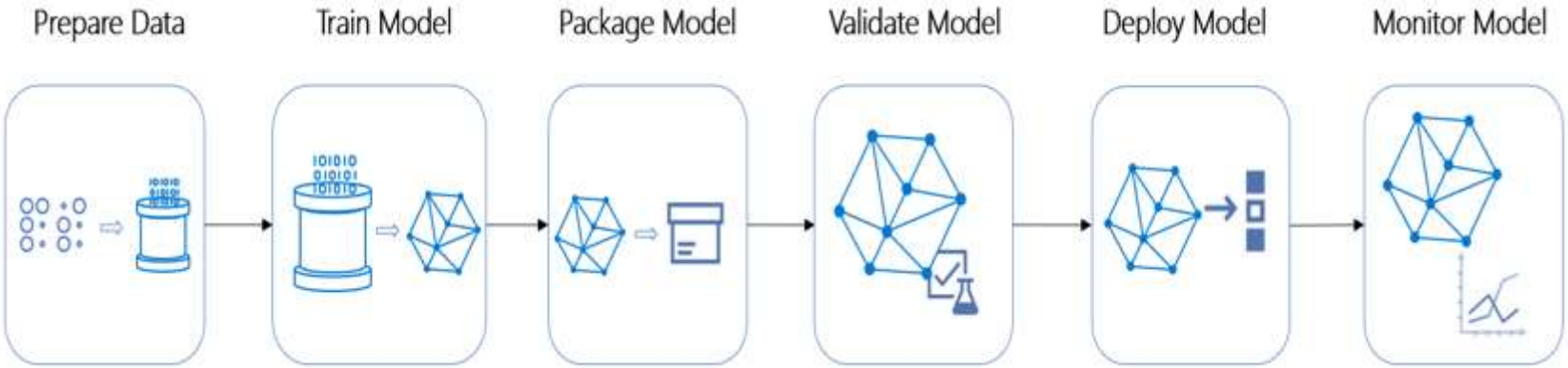
Exemplary AI+DevOps implementation scenarios



Exemplary AI+DevSecOps implementation scenarios



Summary



- Use DevOps to build, deploy, and monitor systems so that a pathway exists to take action on a ML/AI enabled system.
- These ‘actions’ could improve model performance, system security, and many other possibilities

Takeaway

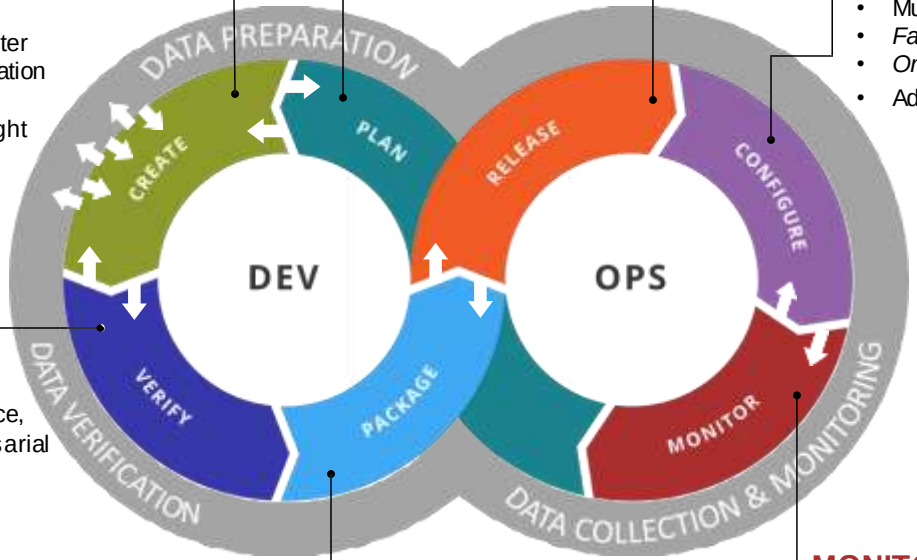
AI+DevOps

CREATE
- Model Selection
- *Training Time (Blocker)*
- Build "ilities" in
- Iterative Parameter Tuning Configuration & Management
- Scalable Oversight
- Regularization

PLAN
- *Metrics*: Performance, Robustness,…
- Architecture Decisions
- Data Preparation

RELEASE
- *Deployment Models*
- Compute Infrastructure
- Mission Risk

CONFIGURE
- Multiple Algorithms
- *Fallback/Rollback*
- *Online Learning*
- Adaptation



VERIFY
- Inspection & Introspection
- Testing: Performance, Robustness, Adversarial
- Security & Privacy
- *Push Back*

PACKAGE
- Integration into larger complex system
- *Algorithmic Agility* (multiple algorithms)
- Architecture Decisions
- Data Preparation

MONITOR
- Adversarial Attack
- Concept Drift
- Detection and Prevention
- "Safe" Operating Boundaries

Major Considerations:

1. Data is necessary throughout the AI/ML DevOps process; data dependencies are hard to track, assess and analyze
2. Emergent behavior in deployment model and from online adaptation
3. Modeling and training can block the speed of creation
4. Verification can push the DevOps cycle backwards at multiple stages

For more information…

DevOps: https://www.sei.cmu.edu/go/devops

DevOps Blog: https://insights.sei.cmu.edu/devops

Webinar : https://www.sei.cmu.edu/publications/webinars/index.cfm

Podcast : https://www.sei.cmu.edu/publications/podcasts/index.cfm

Thank You

Hasan Yasar
Technical Director, Adjunct Faculty Member
Continuous Deployment of Capability
hyasar@sei.cmu.edu
@securelifecycle