



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

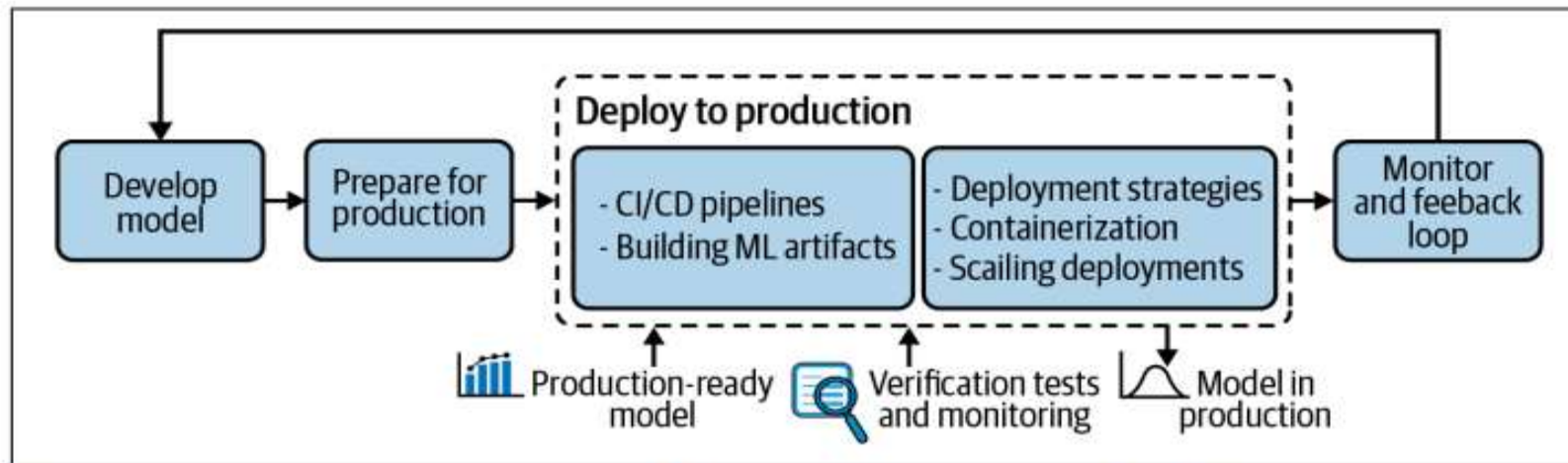
Deploying to Production

Pravin Y Pawar

Adapted from “Introducing MLOps”
By Mark Treveil and Dataiku Team

Deploying to Production

- Business leaders view rapid deployment of systems into production as key to maximizing business value
 - only true if deployment can be done smoothly and at low risk
- Lets dive into the concepts and considerations when deploying machine learning models to production
 - that impact and drive—the way MLOps deployment processes are built



Deployment to production highlighted in the larger context of the ML project life cycle

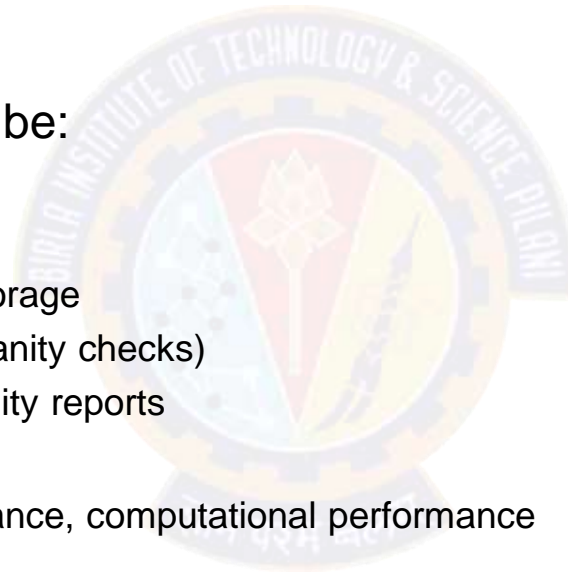
CI/CD Pipelines

A common acronym for continuous integration and continuous delivery (or put more simply, deployment)

- Forms a modern philosophy of agile software development and a set of practices and tools
 - to release applications more often and faster, while also better controlling quality and risk
- Ideas are decades old and already used to various extents by software engineers
 - different people and organizations use certain terms in very different ways
- Essential to keep in mind that
 - these concepts should be tools to serve the purpose of delivering quality fast
 - first step is always to identify the specific risks present at the organization
- **CI/CD methodology should be adapted based on the needs of the team and the nature of the business.**

CI/CD for ML

- CI/CD concept apply just as well to machine learning systems
 - are a critical part of MLOps strategy
- An example of such pipeline could be:
 1. Build the model
 - a. Build the model artifacts
 - b. Send the artifacts to long-term storage
 - c. Run basic checks (smoke tests/sanity checks)
 - d. Generate fairness and explainability reports
 2. Deploy to a test environment
 - a. Run tests to validate ML performance, computational performance
 - b. Validate manually
 3. Deploy to production environment
 - a. Deploy the model as canary
 - b. Fully deploy the model



CI/CD for ML(2)

- Many scenarios are possible, depend on the application,
 - the risks from which the system should be protected
 - and the way the organization chooses to operate
- Generally, an incremental approach to building a CI/CD pipeline is preferred:
 - a simple or even naïve workflow on which a team can iterate
 - often much better than starting with complex infrastructure from scratch
- A starting project does not have the infrastructure requirements of a tech giant
 - can be hard to know up front which challenges deployments will present
- There are common tools and best practices,
 - but there is no one-size-fits-all CI/CD methodology
 - means the best path forward is starting from a simple (but fully functional) CI/CD workflow
 - then introducing additional or more sophisticated steps along the way as quality or scaling challenges appear

Building ML Artifacts

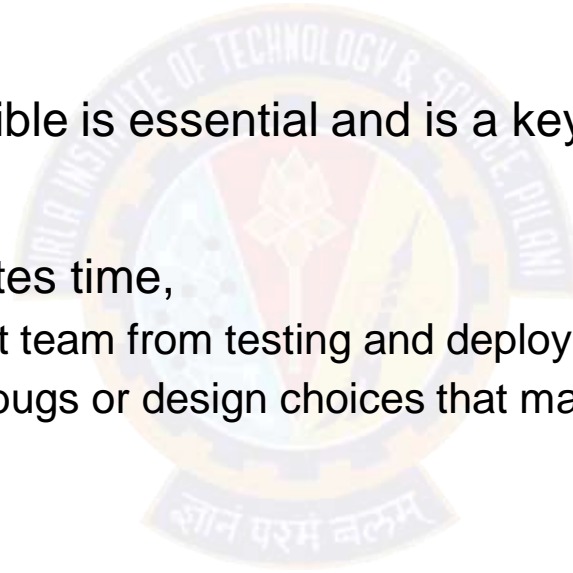
- The goal of a continuous integration pipeline is
 - to avoid unnecessary effort in merging the work from several contributors
 - also to detect bugs or development conflicts as soon as possible
- The very first step is using centralized version control systems
 - unfortunately, working for weeks on code stored only on a laptop is still quite common
- The most common version control system is Git, an open source software
 - majority of software engineers across the world already use Git,
 - increasingly being adopted in scientific computing and data science
- Git allows for
 - maintaining a clear history of changes,
 - safe rollback to a previous version of the code,
 - multiple contributors to work on their own branches of the project before merging to the main branch, etc
- Git is appropriate for code, but not designed
 - to store other types of assets common in data science workflows,
 - such as large binary files (for example, trained model weights),
 - or to version the data itself

ML Artifact

- Once code and data is in a centralized repository,
 - a testable and deployable bundle of the project must be built
 - are usually called **artifacts** in the context of CI/CD
- Each of the following elements needs to be bundled into an artifact
 - that goes through a testing pipeline and is made available for deployment to production:
 - Code for the model and its preprocessing
 - Hyperparameters and configuration
 - Training and validation data
 - Trained model in its runnable form
 - An environment including libraries with specific versions, environment variables, etc.
 - Documentation
 - Code and data for testing scenarios

The Testing Pipeline

- Testing pipeline can validate a wide variety of properties of the model contained in the artifact
 - good tests should make it as easy as possible to diagnose the source issue when they fail
- Automating tests as much as possible is essential and is a key component of efficient MLOps
- A lack of automation or speed wastes time,
 - also discourages the development team from testing and deploying often,
 - which can delay the discovery of bugs or design choices that make it impossible to deploy to production



Deployment concepts

- Integration
 - Process of merging a contribution to a central repository and performing more or less complex tests
 - typically merging a Git feature branch to the main branch
- Delivery
 - Same as used in the continuous delivery (CD) part of CI/CD,
 - Process of building a fully packaged and validated version of the model ready to be deployed to production
- Deployment
 - Process of running a new model version on a target infrastructure
 - Fully automated deployment is not always practical or desirable
- Release
 - In principle, release is yet another step, directing production workload to model
 - deploying a model version (even to the production infrastructure) does not necessarily mean that the production workload is directed to the new version
 - multiple versions of a model can run at the same time on the production infrastructure

Categories of Model Inferences

Two ways to approach model deployment

- Batch scoring,
 - where whole datasets are processed using a model, such as in daily scheduled jobs
- Real-time scoring,
 - where one or a small number of records are scored,
 - such as when an ad is displayed on a website and a user session is scored by models to decide what to display
- In both cases, multiple instances of the model can be deployed
 - to increase throughput and potentially lower latency

Considerations When Sending Models to Production

- When sending a new model version to production, first consideration is often **to avoid downtime**,
 - in particular for real-time scoring
- Blue-green or red-black— deployment
 - basic idea is that rather than shutting down the system, upgrading it, and then putting it back online,
 - a new system can be set up next to the stable one
 - and when it's functional, the workload can be directed to the newly deployed version
 - and if it remains healthy, the old one is shut down
- Canary deployment
 - idea is that the stable version of the model is kept in production,
 - but a certain percentage of the workload is redirected to the new model, and results are monitored
 - usually implemented for real-time scoring, but a version of it could also be considered for batch

Maintenance in Production

Once a model is released, it must be maintained

- At a high level, there are three maintenance measures:
 - Resource monitoring
 - Health check
 - ML metrics monitoring
- Resource monitoring
 - Just as for any application running on a server, collecting IT metrics such as CPU, memory, disk, or network usage
 - can be useful to detect and troubleshoot issues
- Health check
 - Need to check if the model is indeed online and to analyze its latency
 - simply queries the model at a fixed interval (on the order of one minute) and logs the results
- ML metrics monitoring
 - about analyzing the accuracy of the model and comparing it to another version or detecting when it is going stale
 - may require heavy computation, this is typically lower frequency
- Finally, when a malfunction is detected, a **rollback** to a previous version may be necessary
 - critical to have the rollback procedure ready and as automated as possible;
 - testing it regularly can make sure it is indeed functional



Thank You!

In our next session: