CONTAINERIZED MACHINE LEARNING: AN INTRO TO ML IN CONTAINERS

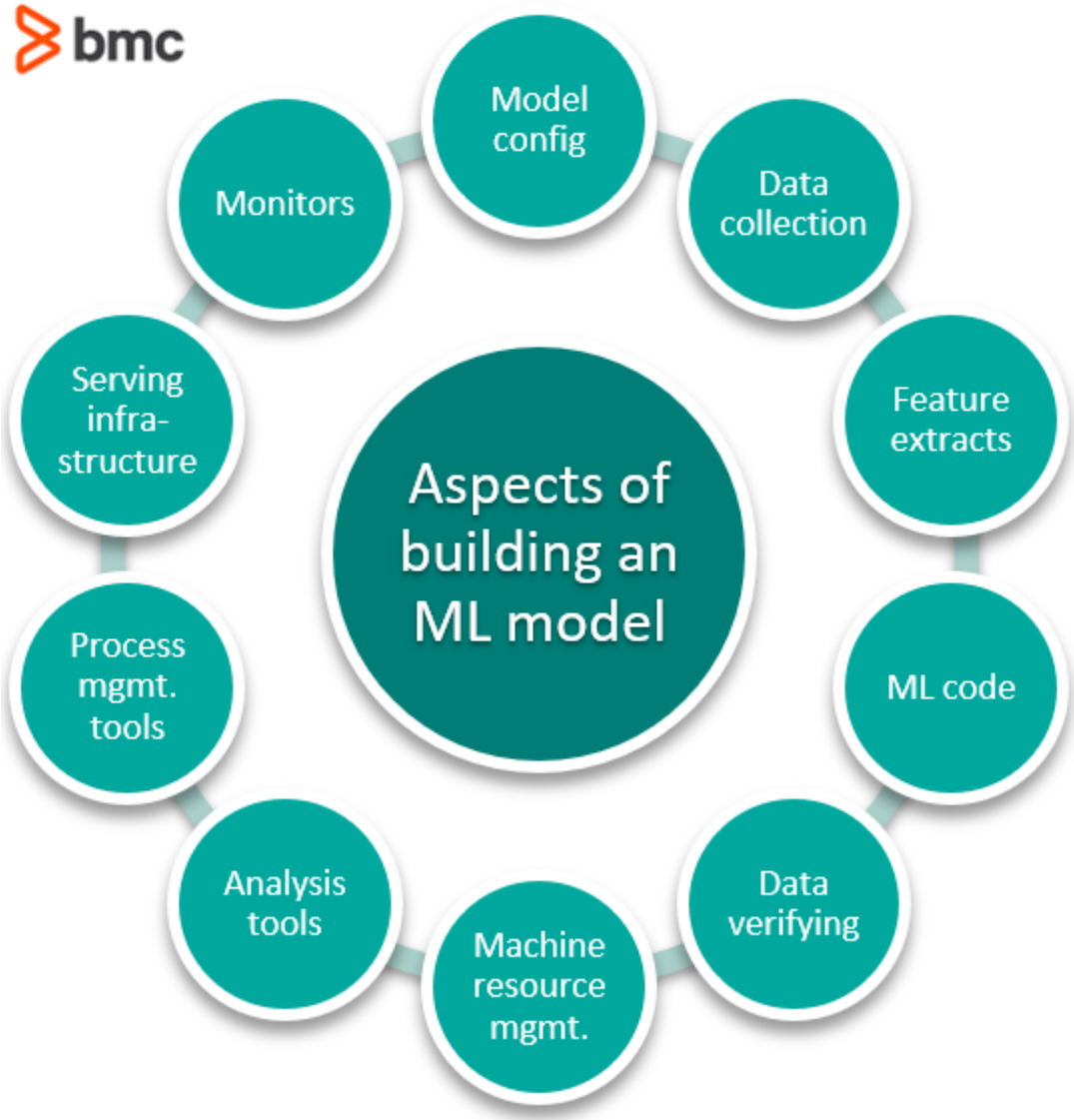
Machine learning models can be resource heavy. They require a good amount of processing power to predict, validate, and recalibrate, millions of times over. GPUs are currently being used to do handle this computing. New types of chips are being created specifically to handle this new load of processes.

One emerging trend—containerizing ML models—seeks to solve this problem. Let's take a look.

Putting ML models in containers

Training an ML model on your own machine, without containerizing, can:

1. Slow down your machine or even make it unusable.
2. Hog local resources for a long time.
3. Waste time on repetitive tasks. If your machine fails during training, training times can start all over.
4. Limit your resources to only what's on your machine.



Given that, the ML process benefits from containerization in the same way developing any application or microservice benefits from utilizing cloud resources:

- Less server down-time
- Third-parties provision necessary resources
- Easier teamwork

How to build and deploy ML models with containers

Containerizing your ML workflow requires putting your ML models in a container (Docker is sufficient), then deploying it on a machine. Kubernetes is the most modern container orchestration tool, and all the major cloud providers offer it.

Adopting a containerized ML workflow does not require adopting a whole lot of new technologies or processes for developers. The AI applications can be written in most common languages and built right alongside an enterprise's current CI/CD environment.

Data lakes: pros and cons

"Use volumes for persistent data. The data that needs to be preserved after a container is destroyed must be written to a volume." - *Redhat Design Principles*

Containers are created to run a process, then they are destroyed. Containers do not store any data, and should not store any, in their design. That's why external persistent volumes need to be used. Resources that offer distributed storage systems can be included in the ML workflow, such as:

- Hadoop
- GCP
- Azure
- AWS

A data lake is a repository to store data in its raw form. It can store all types of data from structured to unstructured data. Data can be stored in the Hadoop distributed file system (HDFS) and accessed by ML containers for training.

Here's how using a data lake can work:

1. The data lake provides a set of training and testing samples to an ML model at the time of training.
2. The container runs its training on those samples and output its results back to storage.
3. The container saves checkpoints of its ML model to the outside data source, not on the container, so new instances of the container can pick up training where one left off.

The trouble is that orchestrating this process can be very difficult to execute. So, we look to Kubernetes and cloud architectures to make it easier.

Kubeflow

Kubeflow is a tool launched in 2018 by Google to help orchestrate the ML modelling process. Where Architects have Revit, ML engineers have Kubeflow. Kubeflow allows ML engineers to see just where their models are at in their training process, in terms of sequence, and also exposes the TensorFlow Tensorboard tool to show how well the model is performing. So Kubeflow abides by the Redhat High Observability Principle (HOP).

Kubeflow is a great tool, so I'll summarize the key benefits:

1. Great documentation
2. Multi-cloud framework
3. Monitoring tools
4. Workflow management
5. Model deployment

Containers support multiple languages

Data science teams often use several languages. Containers let them continue to do just that. Resources will use whatever container is passed to them, so the programming language can be any that a developer is already used to. ML containers can support Julia, Python, R, Go, Java, JavaScript, etc.

The major Machine Learning libraries are written in Python and JavaScript, so data science and MLOps teams will only need to evolve from using their own, local resources to packaging their setup in a container to be distributed on whatever hosting platform they wish.

Google Search is the developer's best friend. Developers share some ML container templates, so others don't have to start from scratch:

- ML Container Templates
- Build a Docker Container with Your Machine Learning Model

Machine learning workflows

Finally, at the core of the ML workflow are notebooks. The idea for notebooks is to make the process from training, testing, and deploying a model as effortless as possible.

Jupyter notebooks

Jupyter notebooks have opened the door for people who do not know how to code to gradually learn code by running and reading code. The notebook lets data scientists conduct research around a ML model's performance and make minor tweaks to it without having to know a whole lot of code.

The notebook has opened the doors to coding and data science research to an illiterate population. I mean that in the most respectful way. The notebook has granted jobs to people who otherwise would not have been qualified, not because they weren't intelligent and couldn't understand the data, but because they simply couldn't code.

For the ML industry, these notebooks are where much of the work, research, and fine-tuning is done. To turn these notebooks into production-ready models takes an extra step, but taking the extra step is cheap compared to decreasing the size of the labor force capable of doing the R&D. Good ML models need a lot of R&D.

Kubeflow

Kubeflow supports Jupyter Notebook integration. It will use the kf-fairing library for both training directly from the notebook or deploying a model from the notebook. Kubeflow includes services to spawn and manage Jupyter notebooks, and even custom resources to containerize a Jupyter notebook so it is prepped to run on the Kubernetes infrastructure.

GitHub Actions

Then, to make this seamlessly integrate with your CI/CD infrastructure and GitHub, GitHub launched GitHub Actions in 2019 to help automate the CI/CD process.

To get the most recent software, a container might pull from a GitHub repo to then execute its ML tasks. An action can be used to deploy a new model architecture on a new commit or an approved code review.

Start containerizing ML

Containerized ML takes just a little bit of work to jump in and setup. Once the leap is taken, ML models can be trained faster and turned into APIs. Their creation can be better monitored, and the R&D phase can be versioned and better documented.

Additional resources

For more on this topic, explore these resources:

- BMC Machine Learning & Big Data Blog
- Guide to Machine Learning with Keras and TensorFlow
- Machine Learning: Hype vs Reality
- Interpretability vs Explainability: The Black Box of Machine Learning
- What Is Human in The Loop (HITL) Machine Learning?