Python-Public,Protected,Private Members

Public Members:

All members in a Python class are **public** by default. Any member can be accessed from outside the class environment.

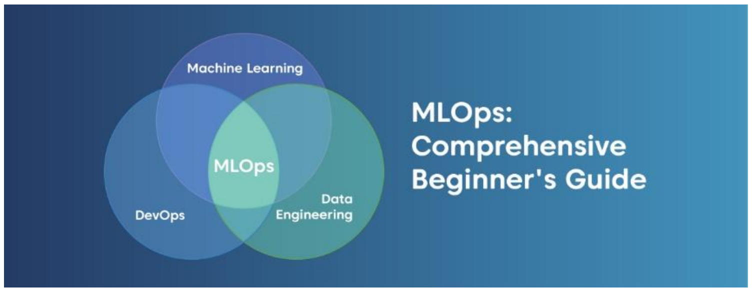
Protected Members:

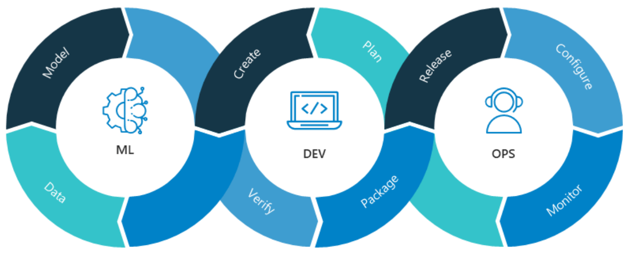
Python's convention to make an instance variable **protected** is to add a prefix _ (single underscore) to it. This effectively prevents it from being accessed unless it is from within a sub-class.

Private Members:

The double underscore __ prefixed to a variable makes it **private**. It gives a strong suggestion not to touch it from outside the class. Any attempt to do so will result in an AttributeError

MLOps – 5 Steps you Need to Know to Implement a Live Project





the 5 important steps required to implement a successful ML project.

1. Team Integration

Planning and assembling the right team is the first step. You need one or more machine learning engineers (ML engineers), data engineers, and DevOps engineers. The number of people required will be based on the complexity of the project. Data engineers are required for manipulating data from various sources, data scientists for the modeling part, and the DevOps team for the regular development and testing.

2. ETL step

This step is aimed more at the machine learning part than at the DevOps part. For the modeling, extract the data from all the sources and create a pipeline to ensure that the data extraction will be seamless even in the system.

Work through the data to identify any nulls, exceptions, or useless values. You need to identify the ideal replacement mechanism for it. It could be anything ranging from replacing them with the average, median, mode, etc. In some cases, you could also remove the entire row.

Create an automated system, wherein the data will flow from the various sources into your system with all the necessary transformations in a seamless manner.

3. Version Control

Version control is followed quite strictly with respect to DevOps. In the same manner, it is important to have strict version control in place for the ML model as well. You can use a Git repository for this as well.

So why is versioning important for ML modeling? Well, each time you run the model, you may be changing the various parameters. This in turn will give different results. If say in the future you wish to go back to the previous set of parameters, versioning will help. another important point is that versing can help you study the various changes done to the model and how it has evolved over the years/months. To help with the pipelining activity a lot of companies rely on end-to-end solutions such as KubeFlow or MLFlow.

4. Testing

Now, this step can get you thinking. A DevOps project has unit testing, integration testing, etc. In an ML project what exactly constitutes testing?

Well, in simple terms, the model validations are what is considered testing for an ML model. So the testing phase should ideally have 2 steps – one for model validation and the other for data validation.

Model validations include checking for the bad rates, accuracy, ROC, area under the curve, Population Stability Index (PSI), Characteristic Stability Index (CSI), and so on. The parameters for validation ned to be set based on the models being used and the use cases. The validation parameters will vary based on the models being used. For example, if you are using an unsupervised ML model like k-means clustering then accuracy, F-measure, precision, and recall are good validation parameters. On the other hand, if you are using logistic regression or linear regression models, then PSI, CSI, GINI are good validators.

So what does data validation do? ML models are oddly specific in a certain way. If the input changes, then the output will also vary. As long as the model is able to make sense of the input, it will give out sensible predictions. So the data validations are done to ensure that the input from the various sources does not change the format. They should be present in the format required for processing the ML model. This is also an important step in the lifecycle of MLOps.

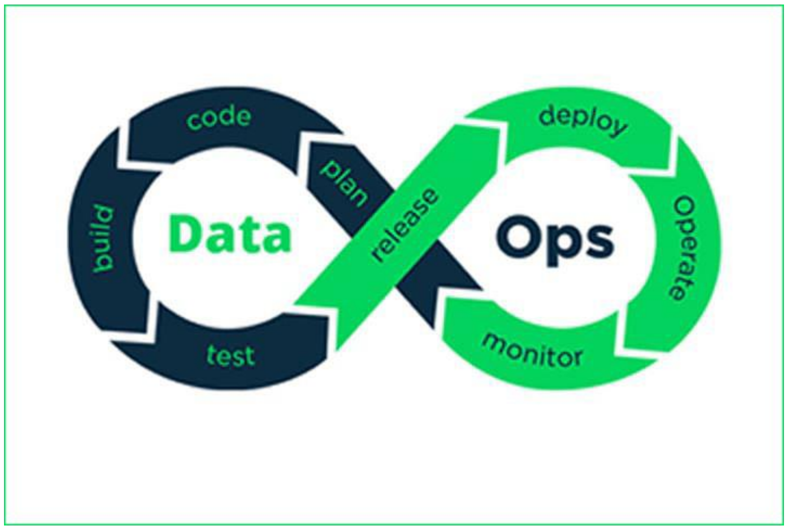
5. Monitoring

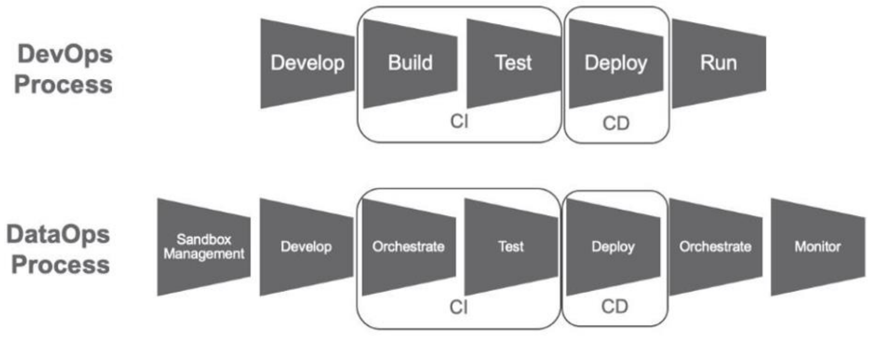
So, once your project is properly integrated and has gone live, the work doesn’t end. In a regular DevOps project, once the project goes live, the work of the developer is completed unless there are further enhancements. In an MLOps project, it is important to periodically monitor the performance of the

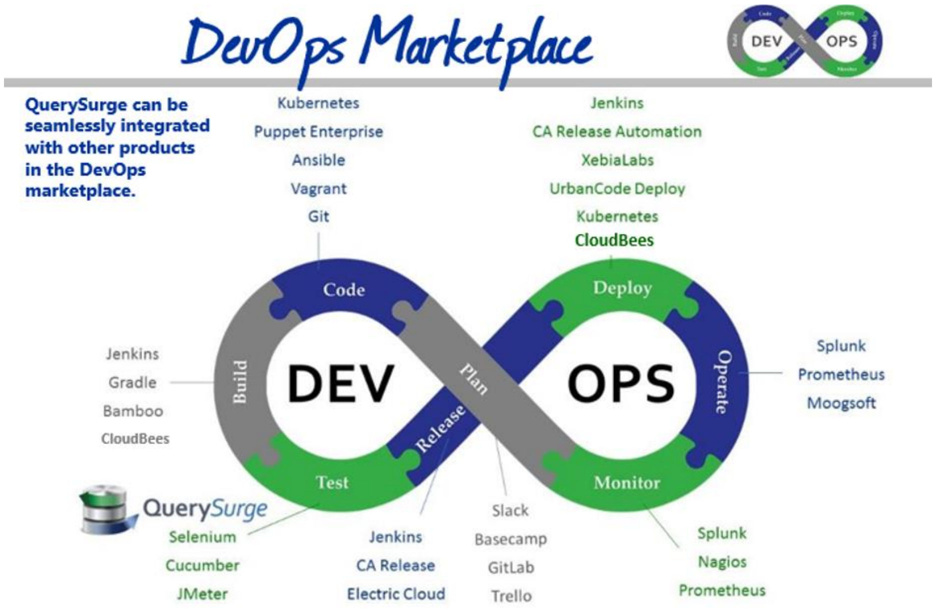
ML model. In any ML modeling, periodic monitoring ad validation is an integral component.

The periodicity of monitoring will depend on the model used. However, it has to be regularly validated against the live data using the same validation parameters used to ensure that the model is working as expected. If at any point you find that the validation parameters are not working properly, you will need to rework the modeling activity. This could either be a minor exercise where a small level of fine-tuning might suffice or a major activity where the model parameters need to be completely reworked.

You can even set up automated reporting to enable easier tracking of these parameters. A periodic reporting can help you notice even the minor deviations in the performance metrics which can be adjusted through a little bit of fine-tuning.







differences between DataOps vs DevOps are:

- **Quality:** DataOps ensures usage of high quality data for high quality outputs; DevOps delivers a quality product.
- **Collaboration:** DataOps works with business users, application developers, and IT operations; DevOps works with engineering and development teams.
- **Cycle Times:** DataOps strives to build a continuous data pipeline so business users become self-sufficient; DevOps strives for shorter release cycles to meet business demands.

- **Operations:** DataOps is constantly addressing new and changing data challenges involving many sources and needs; DevOps runs repeatable, highly similar cycles.