

Applied Accelerated Artificial Intelligence
Prof. Satyadhyan Chickerur
School of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 12

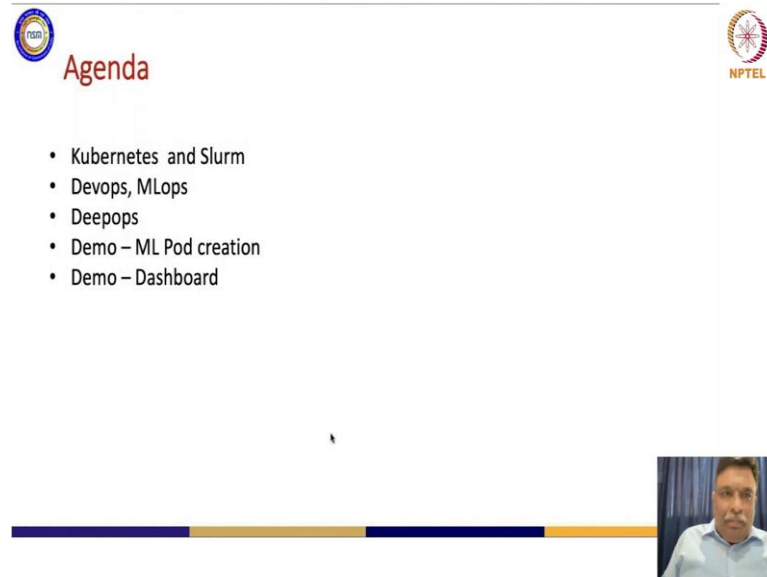
DeepOps: Deep Dive into Kubernetes with deployment of various AI based Services
Session II - Kubernetes Part - 1

Good evening everyone. So, today we would be covering the next part right of the previous session which we discussed and which we discussed was some demo on a small local cluster of our side right. We tried simulating a situation wherein we had a master node and then there were two compute nodes or two worker nodes right. This was what we did in the previous session.

Now, in this session what are we going to do is we are trying to understand that how do you deploy right, a machine learning program we will be taking an example of how basically you can develop that type of a program put it onto a Kubernetes cluster and then we will also try to see some other recent developments in monitoring Kubernetes cluster as a whole right.

We will try to see one or two very basic very very nice ok, dashboards which will help you to understand things much more better and as you go along this course you will understand that once you start developing your programs using this type of a approach which we have discussed in the previous class and then this class you will really appreciate as to how lot of work which you are supposed to do would be done by such systems right.

(Refer Slide Time: 02:10)



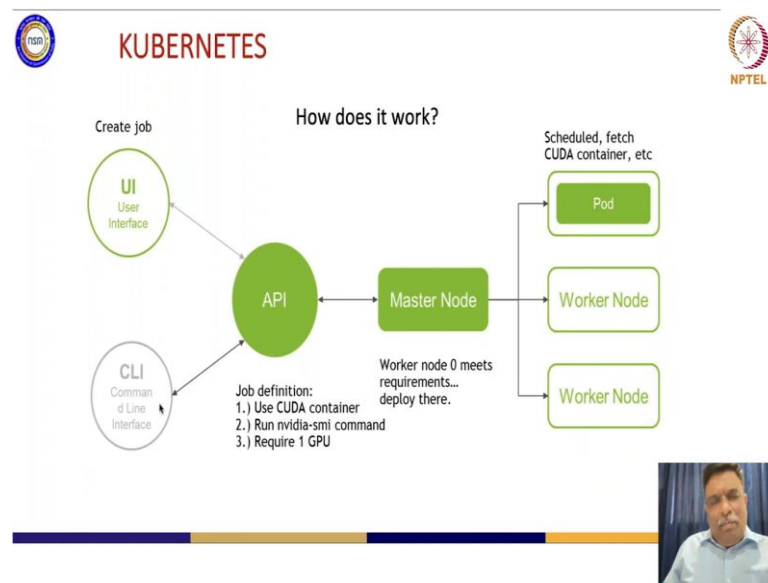
The slide features the IIT Bombay logo on the top left and the NPTEL logo on the top right. The word "Agenda" is written in a large, red, serif font. Below it, a bulleted list contains the following items: "Kubernetes and Slurm", "Devops, MLops", "Deepops", "Demo – ML Pod creation", and "Demo – Dashboard". At the bottom right of the slide, there is a small video feed showing a man with a mustache, wearing a light blue shirt, speaking. A horizontal bar with alternating blue and yellow segments is positioned below the video feed.

- Kubernetes and Slurm
- Devops, MLops
- Deepops
- Demo – ML Pod creation
- Demo – Dashboard

So, let us start with the agenda for the day, we would be discussing a bit of Kubernetes and SLURM in a manner that we will try to just have one slide of trying to understand how they both are linked, try to link it with something which is called as DevOps, which links with MLops, then DeepOps and then we will give a demo of how you can create your own machine learning program right, create it as a pod and then put it onto some cluster for execution right.

So, we will do a prediction analysis of a placement this thing scenario right and we will share the links also in the slack for you to actually work on that. And then there would be a demo of a very recent very well what to say well designed GUI ok for managing your cluster.

(Refer Slide Time: 03:20)

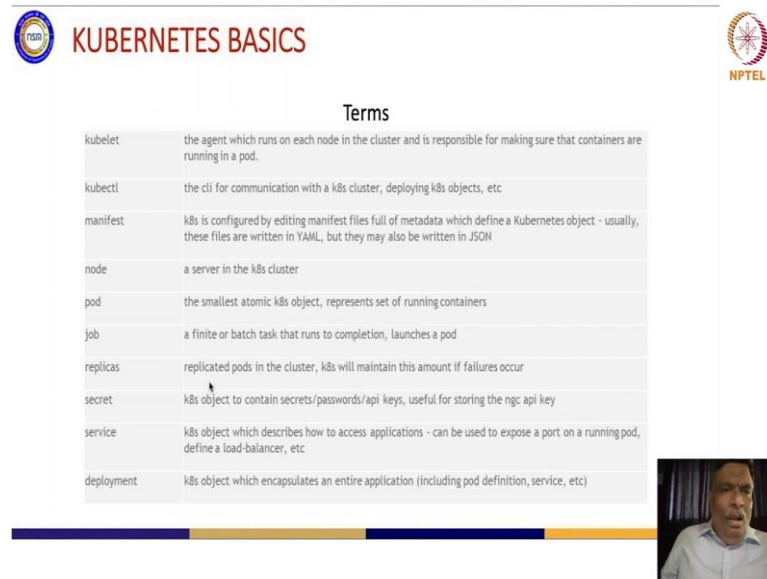


So, let us try to understand this we saw this right Kubernetes we had to create a job right we had to create a job one minute sorry we had to create a job ok and then through the API you send it to the master node and then reduce this thing, so that no it is good for me one second yeah so, yes.

So, the idea is that we have this API ok, which actually helps us to create a job which basically is to be controlled by a master node and which actually has to run on one of these worker nodes as a pod right. Now, here this worker node can actually have a GPU it may not have a GPU right. So, in case it has a GPU you would basically have to install the appropriate CUDA drivers and all that stuff right.

So, based on whether that has a GPU, it does not have a GPU, you will have to set in the environment for the worker node right, but effective idea is you can create this using this a job and run it on a worker node as a pod. So, this is what we are going to actually do today ok. And then you have a command line interface also of course, it is basically terminal which you generally use so, we will have that as well. So, this is how basically the Kubernetes is set up right.

(Refer Slide Time: 05:18)



The slide is titled "KUBERNETES BASICS" and features a table of terms. In the top right corner, there is a small video inset showing a man speaking. The table lists the following terms and their definitions:

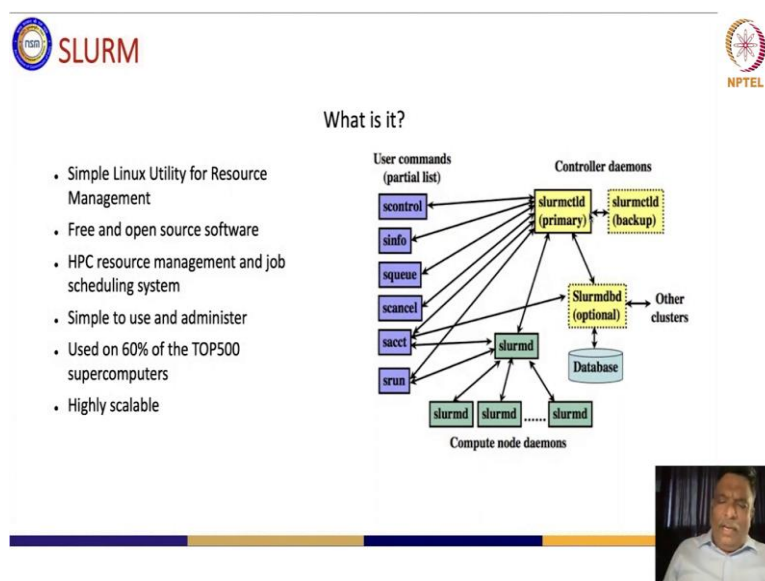
Terms	
kubelet	the agent which runs on each node in the cluster and is responsible for making sure that containers are running in a pod.
kubect	the cli for communication with a k8s cluster, deploying k8s objects, etc
manifest	k8s is configured by editing manifest files full of metadata which define a Kubernetes object - usually, these files are written in YAML, but they may also be written in JSON
node	a server in the k8s cluster
pod	the smallest atomic k8s object, represents set of running containers
job	a finite or batch task that runs to completion, launches a pod
replicas	replicated pods in the cluster, k8s will maintain this amount if failures occur
secret	k8s object to contain secrets/passwords/api keys, useful for storing the nginx api key
service	k8s object which describes how to access applications - can be used to expose a port on a running pod, define a load-balancer, etc
deployment	k8s object which encapsulates an entire application (including pod definition, service, etc)

And then we saw certain basic terminologies just to have a recapitulation of what we actually talked about. So, kubelet is a agent which runs on each node in the cluster and is responsible for making sure that the containers are running in a pod right. So, that is basically a kubelet ok. And then you have this kubectl this is the cli for communication with a k8 cluster or whatever for deploying right.

Then manifest is basically the configuration as a full metadata right, which defines a Kubernetes object - usually, these files are written in YAML, we will see and edit a YAML file as well. And then there is a node which we know which is basically a server in the k8 cluster, pod is the smallest atomic object on that particular node, job is a finite or a batch task that runs to completion which launches a pod.

Replicas we saw are replicated pods in the cluster and k8 objects to maintain api keys, passwords and all this we saw what is a secret thing right. And then we have service and then deployment are k8 objects which encapsulate an entire application right which includes a pod definition, service this that everything. So, this is what is a general basic Kubernetes terminology right. And we will see few of them today right.


(Refer Slide Time: 07:04)




And then, let us try to understand SLURM. SLURM basically is Simple Linux Utility for Resource Management this you saw in one of the previous classes right and what it does? It basically again does know the same thing of scheduling and, but it is basically used for the HPC resource management and job scheduling, but when you see Kubernetes, it does so many things automatically right, this also does automatically, but this basically is a older concept right than the existing Kubernetes concept.

So, that is how it is right, because initially people used to work on high performance computing systems and there were basically all high compute requirements. So, only few people would be given the chance to work on it. So, you had to have a scheduling type of mechanism. So, this is how it actually started.


(Refer Slide Time: 08:05)



CUSTOMER NEEDS



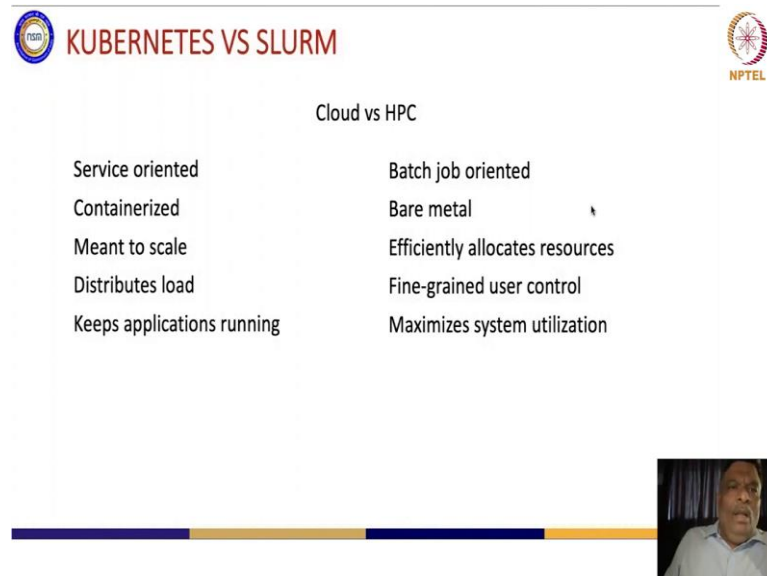
Basic: Kubernetes	Advanced: SLURM
Share nodes, schedule jobs for GPUs on a node (current best solution: Excel spreadsheet)	Multi-node jobs
Covers data permissions and security (LDAP, file permissions)	Job dependencies, workflows, DAGs
Adds analytics and monitoring (important also for justification of purchase)	Advanced reservations
	Intelligent scheduling (not just FIFO)
	Other HPC-like scheduling functionality



Now, if you see from the client's perspective or the customer needs perspective you have got two things right; one is the Kubernetes way of implementing thing, another one is SLURM way of implementing things. So, Kubernetes if you see you can share the node, schedule the jobs for GPU on a node right, something like that and then you can give permission, security, you can do analytics and monitoring this is what we will see today also.

And then these advanced thing like SLURM is for multi node jobs you have DAGs, we will discuss this in our future classes, what about reservation ticketing and then there is a scheduling right and certain other functionality. So, this is how actually it is a bit differentiated right.

(Refer Slide Time: 08:58)



KUBERNETES VS SLURM

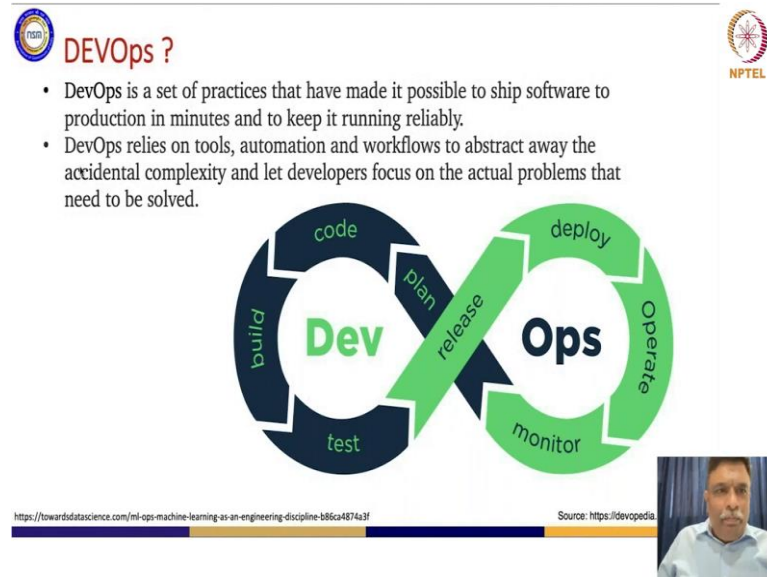
Cloud vs HPC

Service oriented	Batch job oriented
Containerized	Bare metal
Meant to scale	Efficiently allocates resources
Distributes load	Fine-grained user control
Keeps applications running	Maximizes system utilization

Now, coming down to how the implementation happens right. So, Kubernetes basically is used for a cloud like situation. So, it is service oriented, it is containerized, it can be scaled and it does load balancing on its own and it will ensure that your quality of service is maintained and it keeps the applications running.

Whereas in case of HPC when you talk right it is basically a batch type work right. So, batch type job oriented things, you run it on a bare metal, you can also allocate resources and the user control right, can be fine - grained to certain specific levels right and it will also ensure that system utilization is maximum. So, these are certain broad differences right, as to how people try to do all of this.

(Refer Slide Time: 09:57)



Now, let us try to understand, how is this link to DevOps? Now, what is DevOps? DevOps is a set of practices that have made it possible to ship software to production in minutes and to keep it running reliably. That basically means you have two things right one is the development of your software and another one is the operation of what you are going to maintain right. So, there are two different aspects; so, development and operations. So, when you mix both of them it becomes DevOps.

So, if you develop a software you are developing it, once you develop it, it has to be operational right and that is what is DevOps right. So, what does DevOps actually do? So, if you see the development portion of it you have to plan, code, build, test ok. So, you have to plan, code, build and test, then once you do this for your software you release it and once you release it you have to deploy it, operate it, monitor it.

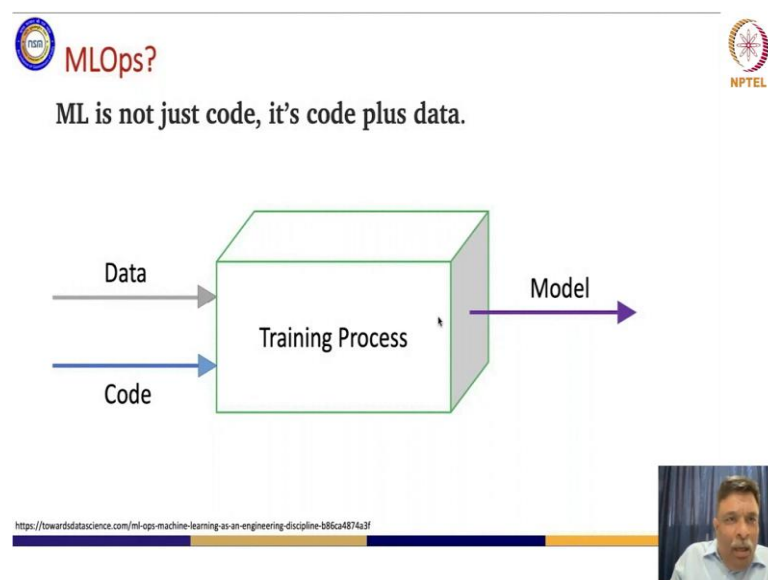
So, this portion is the operational part of your project implementation, whereas, this is what is the development part of your project implementation phase. Now, when we say DevOps, DevOps is going to actually help us coordinate between these two phases ok and when you say they are actually coordinating between these two phases DevOps is going to rely on tools, automation and workflows to abstract away the accidental complexity.

So, the idea is this that it ensures that automation and workflow ok is actually implemented or you are able to implement it without any complexity and let the

developers focus on the actual problem that needs to be solved. So, till now you have seen of course, it may be very overwhelming in the beginning itself to understand about Kubernetes or Dockers and all of this, but once you actually implement these type of algorithms in the days to come, you will really appreciate as to how Docker is very useful for your own use or how Kubernetes is of very easy thing right for maintaining so many people trying to share a single cluster.

So, gradually you will actually appreciate all of this in the days to come. So, that is the reason why you should know actually how this particular thing right or the coordination between the DevOps people who are actually trying to manage things and people who develop ok need to be actually streamlined and understood ok.

(Refer Slide Time: 13:18)



So, let us try to now come to the next version of DevOps which is for the people who actually do machine learning development and deep learning development right. So, we call it as MLOps, when you say MLOps it came from actually DevOps, because you are trying to develop your software or develop your project, at the same time you will have to implement it, deploy it, see that it is operational and see that people are using it right.

So, it also has got two things; the development and the operational aspect of it and since it is related to machine learning MLOps came into the picture. Now, there is a very very basic vast difference right, between the way in which the ML projects are managed and operated vis a vis a general software or a general software project for that matter. Why is

it different? Machine learning project or a machine learning software it is not just a code, we should understand that it is code plus data ok.

Now, this basically means that you have got a situation like this whether where you are having this data also, the program or application code also, then you do a training process or do whatever ok on to some model or whatever you do. But you have got the software code as well as the data to be used with it, but in case of a general software development you generally work with code right.

So, this is one of the very very major differences between the DevOps thing and how it has actually gone to the next level of MLOps. And when we talk of DeepOps it is still more complicated in a sense that when a project is developed to maintain it, to operate it, to operationalize it, it is at the next level of MLOps right.

So, in all these three cases Kubernetes is the major central block which ensures that you can operate, you can deploy, you can schedule, you can coordinate, you can orchestrate all of this very very successfully. So, it supports all of it right.

(Refer Slide Time: 16:09)

MLOps?

While code is carefully written in a restricted development environment. Data comes from the real world, It never stops changing, and you can't control how it will change.

Data

Code

Code and data evolve independently. We can think of them as separate planes with a common time dimension.

The challenge of an ML process is to create a bridge between these two planes in a controlled way.

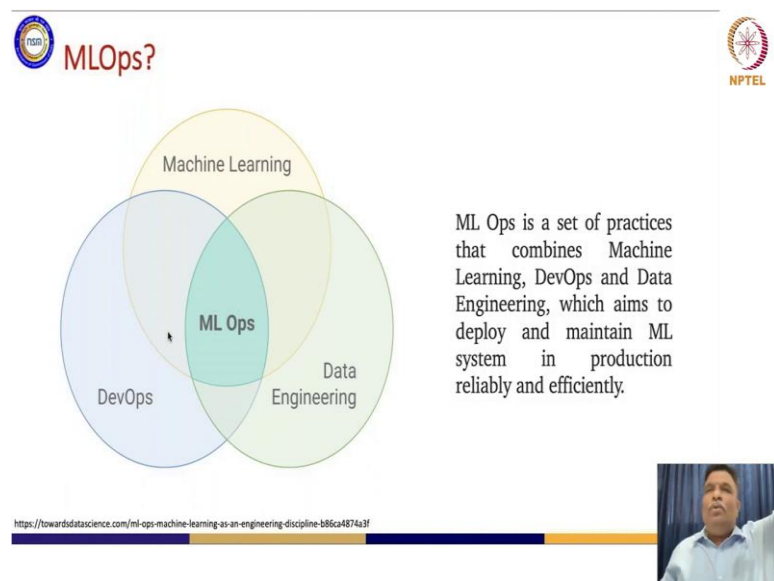
The slide features the IITM logo on the top left, the NPTEL logo on the top right, and a small video feed of a man speaking in the bottom right corner. A decorative bar with blue and yellow segments is located at the bottom of the slide content area.

So, now let us go a bit of a detail into it as to when we say the data is there, the code is there. So, when we say that code is carefully written while the code is carefully written in a restricted development environment. Data generally comes from the real world right.

So, let us say you develop a model, you have written your code, you have created your model, you have created your software application, you would have thought that data would be of this, this type, it should have these, these features, it will have these, these characteristics ok and it is from this type of a modality everything fine. But when it starts coming from the real world right it will be a bit of a changed scenario or it will never stop changing and we do not have control or you do not have control on how it is going to change.

So, technically speaking the data and the code in case of MLOps is going to evolve independently right. So, we think of them as a separate plane with a common time dimension to some extent. Now, when you say that there is a ML process which needs to be maintained so, the challenge of a ML process is to create a bridge between these two planes such that it actually synchronizes and orchestrates the things very very effectively. So, this is how MLOps actually needs to be understood right.

(Refer Slide Time: 18:03)




So, we can again come up to a basic definition or a basic understanding of MLOps, as a set of practices that combine machine learning, data engineering or data processing and then this is what is DevOps. So, MLOps is a set of practice that actually combines machine learning, data engineering and DevOps which aims to deploy and maintain machine learning systems in production reliably and efficiently.


So, ultimately you are going to ensure the terms which people talk in terms of cloud now that I should have this this much quality of service, I should have this much bandwidth, I should have these much resources, I should have this much memory, I should have these many CPUs, I should have these many compute elements or whatever right.

So, you need to ensure that your machine learning application ok satisfies all that because you are talking of production environment reliability and efficiency. So, we have now come down to a level wherein we can now appreciate that how it has evolved from DevOps to MLOps.


(Refer Slide Time: 19:51)



MLOps?



- A data pipeline is a series of transformations that are applied to data between its source and a destination.
- ML models always require some type of data transformation, which is usually achieved through scripts or cells in a notebook, making them hard to manage and run reliably.
- Using proper data pipelines provides many advantages in code reuse, run time visibility, management and scalability.
- ML transformation can be interpreted as data transformation. It becomes natural to include specific ML steps in the data pipeline itself turning it into ML pipeline. Most versions have 2 pipeline one for training other for serving.
- The ML Pipeline is a pure code artifact, independent from specific data instances



Now, let us try to go a bit into what a MLOp is. A data pipeline as we see here which is in the next diagram. So, there will be machine learning pipeline right. So, a data pipeline also is a basic pipeline which is a series of transformations that are applied to data between its source and a destination. So, you can give your data as an input to one of the sequence of modules through which the data has to pass it has to come to a destination. So, in between that whatever is that pipeline that basically becomes a data pipeline.

Now, ML models require some type of data transformations and it depends on your application though right, but it requires some type of data transformation. Now this type of data transformation either you write in some scripts or you write some cells in a notebook Jupyter Notebook or whatever, which actually makes them hard to manage and run reliably.

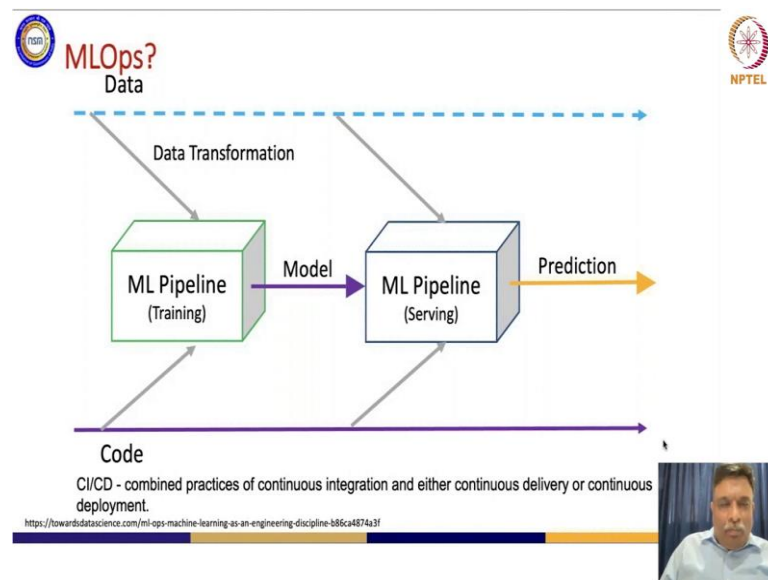
Please understand this that we know that our machine learning model will require certain types of data transformation and which we generally do by writing scripts very fast or we put it in a Jupyter Notebook and there will be so many cells which will do that, but gradually in the long run how do you manage them and run those things reliably is very very important.

So, using proper data pipeline provides many advantages in code reuse, though runtime visibility, management and scalability are also actually improved right, but how do you design a proper data pipeline and then how do you ensure that this ML transformation and this data pipeline right ok are actually coordinated ok. So, we will see, what does it mean?.

So, most versions have two pipelines one for training and other for serving. So, this basically means what, now we are talking of our application which actually has to do some task like classification, object detection, recognition, and some number of applications it can do, it will have two separate pipelines right one for training and one for serving ok. So, when we say it is for serving that basically means people will use it right.

So, we will have to give people to use it and that is the testing phase of our application that is what we say right. So, we want people to use it so that is where serving comes right. So, the ML pipeline is pure code artifact, independent from specific data instances. So, ML pipeline does not have any interference with the data pipeline whereas, data pipeline is totally separate from the ML pipeline and then you assume it that you have two types of pipeline right one is for training and another one is serving right.

(Refer Slide Time: 23:36)




So, this is how it is, you take the data, you transform it appropriately right. So, there will be a ML pipeline, there will be a model which will be trained using this type of a training mechanism, then you have a code which will actually be actually training this developing this trained model right training model for that matter.


And then the once this is done you again giving the data to the serving pipeline and then you do the prediction. So, this CI/CD ok combined practice of continuous integration and continuous delivery or continuous deployment. So, try understanding that, as we go along this particular axis you should actually go on doing so many things right.

You should have continuous integration, you should have continuous delivery, you should have continuous deployment, all of this go hand in hand, but you should ensure that everything happens so seamlessly smooth right, that we will try to actually synchronize things right and then do basically a very very good efficient ok, system for synchronizing and orchestrating all of them ok.

(Refer Slide Time: 25:03)




DevOps – MLOps



Practice	DevOps	Data Engineering	ML Ops
Version control	Code version control	Code version control Data lineage	Code version control + Data versioning + Model versioning (linked for reproducibility)
Pipeline	n/a	Data pipeline/ETL	Training ML Pipeline, Serving ML Pipeline
Behavior validation	Unit tests	Unit tests	Model validation
CI/CD	Deploys code to production	Deploys code to data pipeline	Deploys code to production + training ML pipeline
Data validation	n/a	Format and business validation	Statistical validation
Monitoring	SLO-based	SLO-based	SLO + differential monitoring, statistical sliced monitoring

<https://towardsdatascience.com/ml-ops-machine-learning-as-an-engineering-discipline-b86ca4874a3f>



So, let us try to actually understand a bit of a similarity ok between the DevOps and MLOps and the differences between both of them. So, see DevOps was developed to maintain version control, maintain good pipelines, to understand behavior validation, CI/CD continuous deployment ok, then data validation for some matter and monitoring.

So, then, DevOps and MLOps actually are in a sense that you have this code version control, code version control and data lineage, code version control plus data versioning plus model versioning. So, since when people started developing this code version control on DevOps, because at that time it was only DevOps ok.

Then you had this data lineage also linked to code version control you went and mixed both of them to get a ML of type of a thing which is code version control plus data versioning plus model versioning and you should ensure actually that it basically meant ok that you are going to link DevOps with data engineering plus adding this model based versioning thing into it right.

So, when you talked of this pipeline initially DevOps did not have any concept of pipelines because you only had to ensure version control and know something like that. But when you talk of data engineering you surely had a data pipeline the extract ETL ok. So, when you had this ETL and data pipeline in data engineering, the MLOps was bit modified into training ML pipeline and serving ML pipeline. Then, behavioural

validations, you had unit test, but in case of MLOps; obviously, you will have to have a model validations.

And when you talk of CI/CD ok, deploys code to production, deploys code to data pipeline, deploys code to production plus training ML pipeline. So, these are certain differences that how do you actually try to do all of this. Then data validation right, what type of data needs to be validated? In case of DevOps there was no data. So, there was no question of data validation as such, but when it came down to data engineering you had this format and business validation and when you talked of MLOps you had this statistical validation right.

And when you talked of monitoring all of these operations right in case of DevOps in case of data engineering in case of MLOps right you use something called as service level objective right or service level agreements or whatever right.

So, in that case monitoring was done on service level agreement types, here also, here also, then in case of MLOps right there are service level objectives, service level agreements, then there is something called as differential monitoring and statistical slice monitoring this basically is a advanced version of trying to monitor things right.

We are not going into the details of any of this, but trying to link the DevOps with MLOps right along with the new pipeline which got introduced because of data engineering right.