Introduction to Proteogenomics

Dr. Sanjeeva Srivastava
Dr. D. R. Mani
Department of Biosciences and Bioengineering
Indian Institute of Technology, Bombay
Broad Institute of MIT and Harvard

Lecture - 26
Workflow for Automated Data processing

Welcome to MOOC course on Introduction to Proteogenomics. Analysis of large data set is tedious and time consuming process. Data can be stored in large spaces available on the internet called Cloud Storage rather than using the local computers or clusters. Data can be analyzed using cloud computing. In today's lecture Dr. D.R. Mani will introduce you to the basic idea of cloud computing. This lecture will also provide you an idea of the workflows involved in automating data analysis and the processing steps. So, let us welcome Dr. Mani for today's session.

So, if you take a standard computer let us say a laptop or a desktop that you have and you connect hundreds or thousands of them together and you have an operating system that can use all of them efficiently and you have a network that is so fast that you can move data from one laptop to another without having to get a coffee break in the middle, then you basically have a supercomputer.

So, there were several types of supercomputers that were built in the last like several decades but of late they are all converging towards basically what you can call a rack of computers. So, you take a lot of computers basically motherboards, the boards that go into your computer, you stick them into a big rack connect them with a very fast network and then you have an operating system that can take jobs or tasks that are assigned by that or being issued by users and then efficiently spread them across all the computers.

So, that is basically a current day equivalent of a supercomputer. It is a lot cheaper than building a custom one, but it is still expensive because the networks and disks have to be really fast to keep up with a lot of the computing.

So, a lot of the major institutions and research centers started having their own computers. Many times they are called grid computing environments because you connect the computers in a big grid and then you have a job scheduler or an operating system that can dispatch jobs to the grid and you would be able to have a lot of users run many programs.

So, let us say you have Protigy that you saw the other day and you want like 50 people using it at once; obviously if you run it on one server, it is not going to work and, so you set it up on a computer. You give it like maybe 50 processors, so that every time a user submits a job it goes to a different processor. So, that is automatically taken care of by the user.

So, people started building systems like that. So, they are called local clusters or local grid computers. So, the broad had one with about 3000 nodes in them. So, it had 3000 processors and you could submit your genome alignment job and if it needed 5 processors, it will get 5 processors and it would run it on those but as people started doing this, it became more and more expensive to maintain them.

There are a lot of computing nodes you have, there is lot of hard disk and the other stuff you need to track and keep operational and you need to constantly keep upgrading as computers become faster and disk drives become faster and so, it is an expensive operation for each research institution to kind of maintain and have these are available.

So, the latest trend was there are some companies in the world that has so much computing that they can basically rent computing to anybody who wants to do any kind of computation. So, Google is one of them.

So, Google has an extensive infrastructure for parallel computing with a lot of storage because they crawl the entire internet and index everything that there is on it and so that requires a lot of computing and a lot of space and they built a very large computing system to deal with it but then they are not using it 100 percent of the time. So, they figured we have this idle now and then why do not we start kind of renting it to other people so they can use it.

So, Amazon did the same thing. Amazon has an extensive computing infrastructure that they have for their business. It started off as a computing environment for their business, but slowly as other people started using it and found it to be very useful and actually significantly cheaper than your own IT department maintaining your cluster or grid computer people started using those systems more and more and now they have their own infrastructure for just providing computing to people who want to use it.

So, these kind of systems where you have a lot of computing and disk space, lot of computing power and disk space available that you access through the internet is called a Cloud computer. So, it is the cloud because the internet is usually represented as a cloud. When you are drawing a network diagram of computers, you always draw the internet as a cloud because it there is no specific connections that you can.

So, if you connect your laptop to your neighbour's laptop, then you can say computer 1, computer 2 and draw a line between them, but the internet is a set of connections that basically are generated on demand and you do not know what goes where. So, you generally draw it as a cloud and so computing systems that you access using the internet are called cloud computing systems.

There are two or three very big ones. One is Amazon obviously; Amazon system is called AWS, Google has one which is called the Google Cloud Computing platform I think, Microsoft has one called Azure I think but Google and Microsoft are the sorry Google and Amazon are the really big ones that pretty much everybody uses.

So, the concept there is they have a large parallel computing system that they that Google maintains and when you want to do some computation, you basically rent whatever you want.

So, you say I want 10 computing nodes to run my job for 5 hours. You get that, you finish your job, you pay whatever it costs like a few dollars and then you are done. You do not need to pay them anything else. So, you do not need to maintain it, you do not need to upgrade it, if things go wrong you do not need to fix it.

Google will do everything for you and they will maintain a large enough set of nodes that you can basically get very large computers on demand and. So, when these things started becoming available a lot of kind of genomic, proteomic and proteogenomic computing started to move over to the cloud.

So, there is a big difference in kind of how you think about analysis now. So, previously you had all your data on your laptop. People you wrote algorithms, you got the algorithms to your computer. So, your data was fixed, you brought your algorithms to wherever the data was and you ran them. Now the algorithms are in the cloud. So, you have to take your data to the cloud. So, it is kind of a paradigm shift in the sense that you now kind of move your data to the cloud and then do the analysis there.

So, there is costs for storing data. There is cost for getting data out. The most expensive part of cloud computing is if you want to get data out of the cloud because they want it to be there, so you can they can charge you for storage and they can charge you for computing, but if you take it out then there is the cost of taking it out as many times more than actually doing the computation.

But that that is how they have set up the infrastructure but the bottom line is you can now have algorithms that are deployed on the cloud. It could be what you have developed many times. Other people make their algorithms available, you take your data, you put it there and then you analyze it and you get results, you download the results and then you can look at them. So, that is kind of the basic 5 minute overview of what cloud computing is and how you use it.

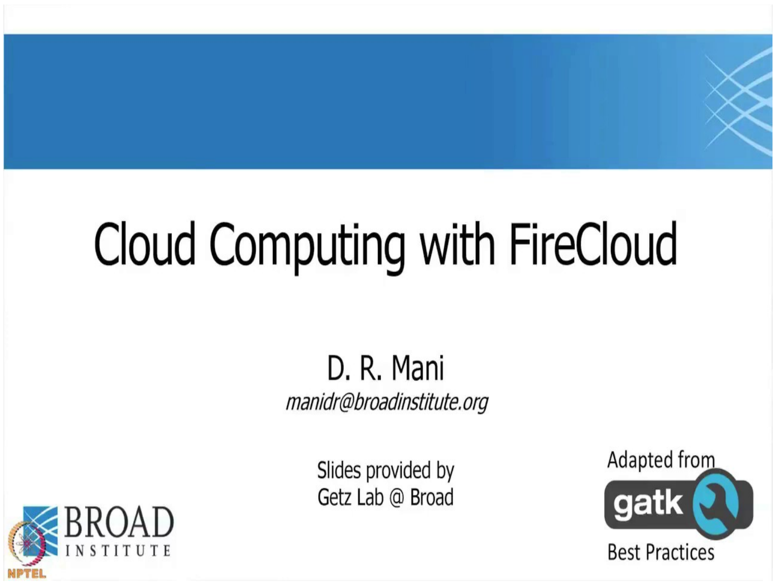
So, some of the advantages here other than the maintenance and cost I mentioned is that a lot of these are set up to provide a lot of parallelism. So, if you want like a machine learning algorithm that is very compute intensive, then you can recruit 100 or 200 or a 1000 nodes, do your; runs very fast or however long it takes and then release the nodes. So, it is on demand, it is very flexible and it provides a lot of kind of flexibility in terms of how you use computing and what algorithms you use.

So, what I am going to go through here is at the broad we have this platform called Fire Cloud. I will go through what FireCloud is and how it is deployed on the Google cloud. So, the cloud computing platform that FireCloud is written on is the Google cloud computing platform.

So, I will kind of go through how it is done and how you can create pipelines for analyzing all kinds of data using FireCloud and so, when you use FireCloud , you are essentially using the Google cloud computing platform and I will just at the end of it, I will spend a few minutes to show kind of a hand demo where we have put a lot of the proteogenomics tools on FireCloud the I called it a demo or not a hands on because as of now the workspaces and algorithms are not accessible to everyone.

They will be soon, but right now they are kind of private because we are still developing and the broad wants to ah make sure we know what kind of license we are going to use and things like that before we make it public but it should be available to the public very soon and I will just give you a demo and it is kind of a teaser for what you can do in the near future.

(Refer Slide Time: 11:03)



Cloud Computing with FireCloud

D. R. Mani
manidr@broadinstitute.org

Slides provided by
Getz Lab @ Broad

Adapted from
gatk
Best Practices

BROAD INSTITUTE

NPTEL

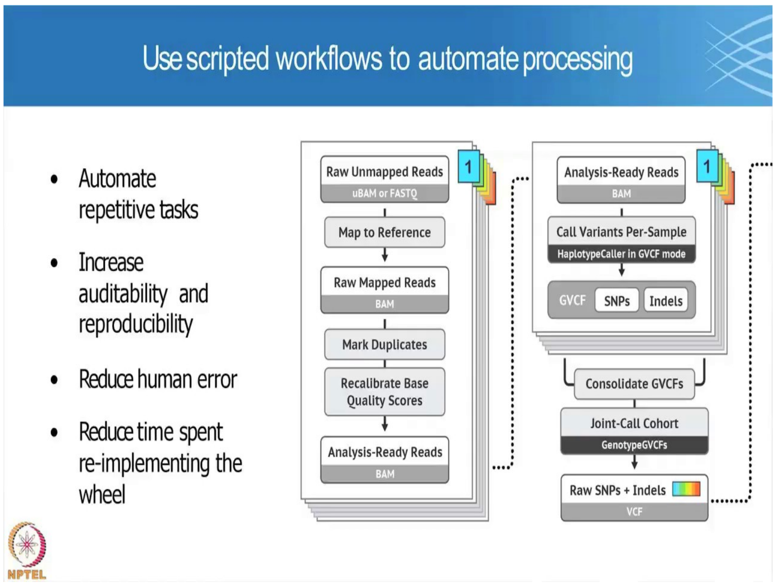
I want to kind of acknowledge that the set of slides you see was created by people in Gaddy Getz's Group at the Broad Institute. So, he does a lot of genomic data analysis and FireCloud the primary use for FireCloud when it started was for genomic data analysis. So, they created these slides and I am just modifying or adapting them for this presentation. So, I want to kind of acknowledge that these are their slides.

So, first of all when you are doing a lot of analysis, so you are going to write a paper that does a lot of proteogenomic analysis. For example, that uses proteomics, phospho, proteomics, whole genome, whole-exome, RNA-Seq you are combining all the data, you are doing a lot of analysis, you can try to do it in excel but at the end of it you will be so confused on what you did that you cannot write the method section and if you did nobody can replicate it.

So, usually when you do large scale big projects like that which involve a lot of analysis with many different pieces, you generally write code. So, you write a script that says this is my initial data set. I did this filtering and then I did this imputation and then I did this test and then I visualized it with the heat map or something like that.

So, there are many different like paths analysis, paths for different types of data. There is a lot of analysis you do and one easy way to kind of document all of that is to write code. When you write code, what the algorithms are doing and what data processing you are doing is extremely clear.

(Refer Slide Time: 12:43)



Use scripted workflows to automate processing

- Automate repetitive tasks
- Increase auditability and reproducibility
- Reduce human error
- Reduce time spent re-implementing the wheel

So, you generally write code and the so here script basically means is code. In in many different languages when you have code that is interpreted and you write line by line. It is called a script ah. If you write a big chunk of code and then run it in a compiled language like Java or C, then it is called code but if you write R you would call an R program a script. So, when I say script I mean like a program.

So, the advantage of writing something like that is that in the in the breast cancer project for example, what happened was we did some set of analysis. People looked at all the results and said; oh, this data has to be pre-processed in a different way; what we did now is it is not quite right and so, you go and make a change. Now you have to re-run and all the analysis you did because the basic data changed.

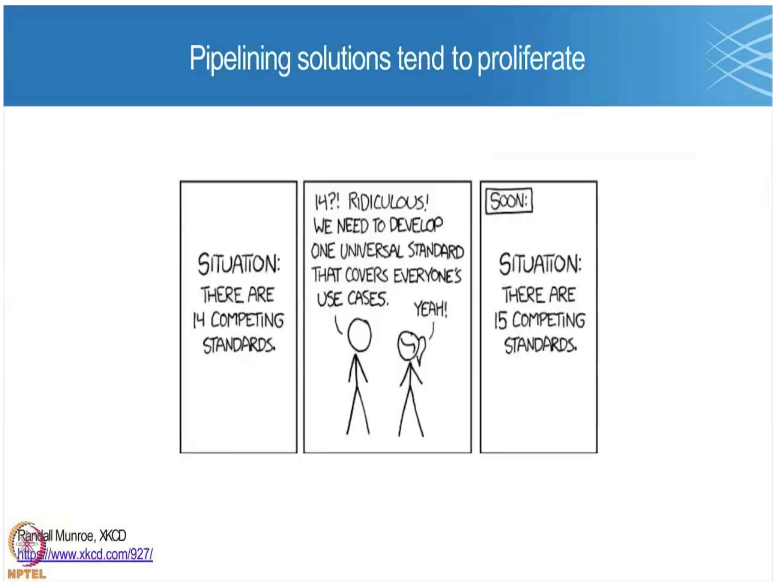
So, you have to repeatedly run your analysis many times and also if you make it general enough, you can run it on other types of cancers too. Whatever analysis we did for the breast cancer, you could have done it for the other cancers also. So, that is kind of the concept behind writing a script, so that you can automate your processing.

So, if you have a script all you need or you need to specify what your inputs are and then the whole thing will run and all the analysis will get done. So, it helps you automate repetitive tasks. So, you do not need to keep doing it and if people do repetitive things, they generally tend to make mistakes because that the brain is not tuned to repeating the same thing over and over again.

You can increase reproducibility and someone else can see what you did without having to kind of bug you and you trying to remember what was done and so you can reduce human error and you can re-implement. Sorry, you can reduce time spent on re-implementing.

So, suppose you have an analysis method that you have in a big pipeline, you can give it to someone else and the ovarian cancer group can use your pipeline and do the same analysis on their data without having to rewrite everything again. So, that is kind of the advantages of using automated scripts for a data analysis.

(Refer Slide Time: 14:53)



Pipelining solutions tend to proliferate

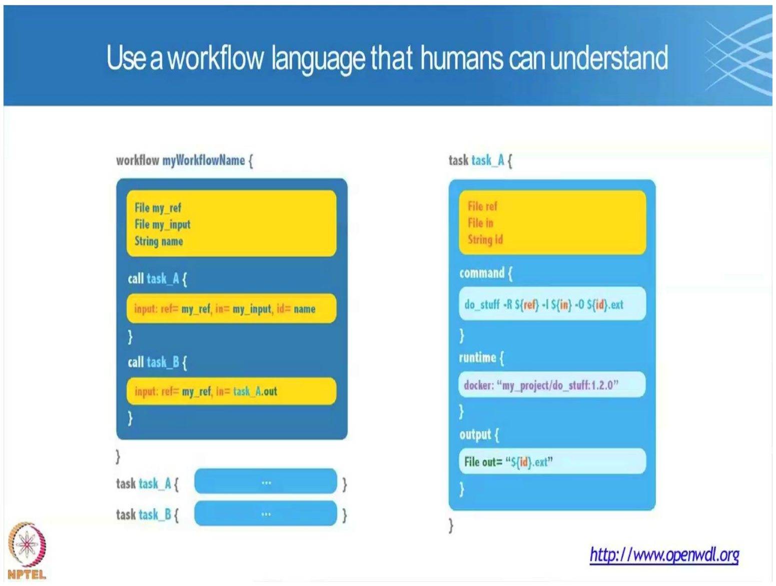
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

Randall Munroe, XKCD
https://www.xkcd.com/927/

So, there are a lot of ways to take a script and make it into a pipeline, so that this is like a humorous thing where the situation is that there are 14 competing standards. So, there are let us say 14 competing pipelines and then they discuss they say you have 14. That is ridiculous. We need one that kind of combines the features from all of them and so now we have 15 competing standards.

So, there are already many tools for writing pipelines and kind of using them but I guess the point I am trying to make is that we have one more and we think ours is better like everybody else.

(Refer Slide Time: 15:39)



Use a workflow language that humans can understand

```
workflow myWorkflowName {
    File my_ref
    File my_input
    String name

    call task_A {
        input: ref= my_ref, in= my_input, id= name
    }
    call task_B {
        input: ref= my_ref, in= task_A.out
    }
}
task task_A {   ...   }
task task_B {   ...   }
```

```
task task_A {
    File ref
    File in
    String id

    command {
        do_stuff -R ${ref} -I ${in} -O ${id}.ext
    }
    runtime {
        docker: "my_project/do_stuff:1.2.0"
    }
    output {
        File out= "${id}.ext"
    }
}
```

http://www.openwdl.org

So, the way scripts are written in this FireCloud environment is basically in a human readable form. So, many times you there are environments where you can graphically write a script. So, there will be a box for doing a data filtering, there will be a box for doing data normalization. So, you layout all these boxes and then connect them with arrows and that is your pipeline.

So, there are tools that do that ah, but the problem with those is if you want to rerun them with a different data set for example, or you want to change your threshold from 5 to 3 like you did in your R program yesterday, you have to open the thing, you have to click it, you have to go to the right box and make the change. So, it is not prone to higher levels of automation.Suppose you wanted to loop through 5 different thresholds like we did yesterday, then you would have to go to the user interface, make the changes 5 times and then rerun it. Would not it be nice if you could just do that in yet another program that actually calls your pipeline?

So, in order to go automate things like that you want your pipeline to be defined in a way that is also automatable and so, here what we do is we use a language called WDL. It is pronounced widdle and it stands for Workflow Description Language. What that does is there is a definition of a workflow.

So, there are some input variables that you can give when you call the workflow and then there are several tasks that the workflow executes. So, it starts with tasks A, does whatever is needed for task A and then goes on to task B.

The nice thing about this is the output of task A can now become inputs for task B. So, that is how you chain your things together. If you had a task C, the output of task A could go to task C and skip task B. In that case you know the task A and B can be executed in parallel because they do not have any dependencies.

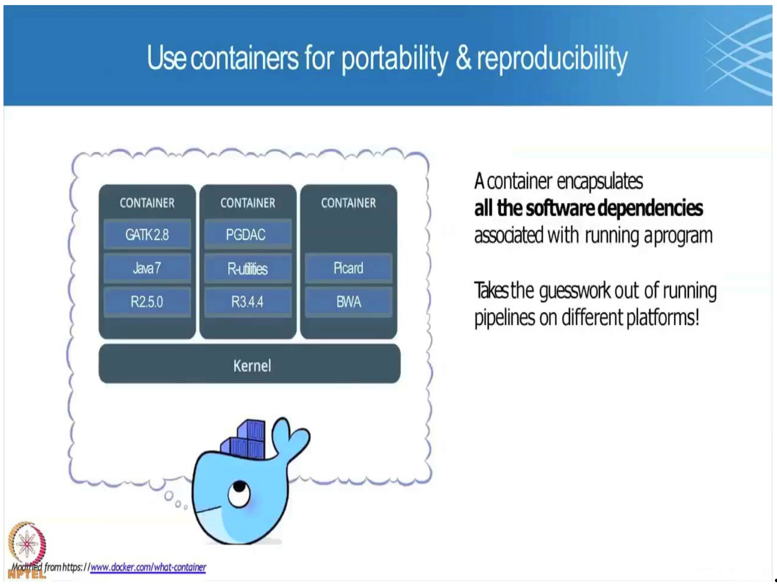
So, what the language does is it says let me draw out my graph. If things are not dependent on the previous thing being run, I can run them in parallel. So, if you have task A, B and C and C requires input which is the output of task A, then it is going to run A and B in parallel and as soon as A is done, it will start C. It won't wait for B to stop. So, it provides some kind of inherent parallelism to run things which are not dependent on each other separately and so, it speeds up your pipeline execution too.

So, once you have defined your workflow you have to define what your tasks are and so, here is an example of how a task would look. So, task again has inputs, there is a command. So, remember a lot of these things are based on tools that were written for the command line and in order to put it into ah into WDL and use it in FireCloud, you have to be able to run your algorithm or task or method using command line invocation.

So, you say what command you need to use in order to run task A. The other thing here which says where do I run this task is the cool is another kind of recent technological concept that that is now permeating how people write software. How many of you have heard of Dockers? Couple of them, ok.

So, I will I think there are more slides that kind of go into that in more detail but basically where the thing is run is in a Docker and then it creates some output. So, once the output is there, other tasks that are run afterwards can use that output. So, that is kind of the overall concept of how a workflow is written in FireCloud.

(Refer Slide Time: 19:23)



Use containers for portability & reproducibility

A container encapsulates **all the software dependencies** associated with running a program

Takes the guesswork out of running pipelines on different platforms!

Modified from https://www.docker.com/what-container

So, this is the thing about Dockers.

So, Dockers are basically conceptually containers. So, what you do is there is this kind of environment called Docker. So, you have to download the software environment, install it on your computer and then the concept is that suppose so all of you saw how much we struggled with getting the versions and stop write for the protigy hands on because we started with 3.5. We did not work. We went back to 3.4 and then to 3.5 and every time we made a change you had to go and make the change in your computer and pretty soon it got annoying that you had to reinstall so many times.

A nicer way to do that would have been to create a Docker. So, here what we do is conceptually we create like a container that says this container is going to contain all the software that I want to run my program and it will contain everything in the right versions. So, you start a Docker and then you say I need the R version 3. You put in R version 3.5 and then you say I need these 15 libraries. So, you put in all the 15 libraries, you need then let us say Java to run something else and a specific version of Java you can put in that. So, you put all those and then you create a new container.

Now, instead of asking you to download Protigy on your computer and install R and this and that, what we are going would have done if we had a Docker is to say download the Docker. So, just like you can put code on GitHub, you can put Docker containers that you have created on a repository called Docker Hub. So, what we would have done in the ideal case is create a container for Protigy, put it on Docker Hub and say everybody in the class go to Docker Hub and download this container.

If you had done that you would not have to worry about versions. It will run on your computer because it has been set up to run the correct way. All you needed to do was download the Docker and give your data as input. So, that is kind of the concept here. In in a lot of the cloud computing frameworks what people do is use Dockers to make it platform independent.

So, you have a Docker that runs on the Google cloud, you can take the Docker and run it on Amazon cloud or you can take the Docker and run it on your laptop. It will run anywhere ah. Docker cum containers can be executed and that is most operating systems in computers.

So, what we are trying to show here is that there is a common operating system and on the operating system you can create containers. So, this one is for our proteogenomic data analysis containers, this is for GATK to do genome analysis, this is for some other alignment and so forth.

So, you can create dockers on a common kernel and this kernel will be made available by the by the or the company or the institution that supports the Docker platform. So, they will make a common kernel available. You can modify the kernel by adding stuff that is specific for your application and then you create a container and now that is your Docker for that application.

And so in this workflow what you are doing in the runtime is you are you take the command and you execute that command in a Docker that contains the algorithm that you have written. So, this makes it very easy to kind of encapsulate things and if you want to replace your algorithm with a new version, all you have to do is replace the Docker with a new version or update your Docker and then you will have the new version in there.

Any questions so far? Ok yeah.

Student: Sir, can we go back at some point of time and add things to the same Docker or does it freeze after we put something once?

No, the Dockers are plug. You can make changes, but the good thing is when you make a change you create a new version and so let us say you published your paper with version 1 of this Docker, now you realize that I can actually do this much faster by changing some computation that I do. So, you do that and you create version 2 of your Docker.

Now, both versions can be kept around you can say I will keep my version 1 around just in case somebody who is looking at the paper wants to run what I did, but I for most users I want to use my new version because it runs much faster. So, you can keep a version of this method that uses the old Docker and create a new version that uses the new Docker or you can say I do not care it is the same thing. It just runs faster. I am just going to replace it, then you can replace the Docker with your new version.

So, you can do either way and Dockers are not like written in stone. You can modify them, you can create newer versions and you can take what you have and add more things if you want. Any other questions? Yeah.

Student: If we add the like in container if we add the one version and then second version if we download it?

Yeah.

Student: Are you going to give a command as to which of the two is executable?

So, when you download the Docker you have to specify what version you want. I think the default is if you do not specify any version, you will get the latest version, but if you want the older version you have to specify which version you want. So, just like in code, right. So, you can have code that is version 1.

So, windows for example you have windows 10.5 or Mac OS 10.5.

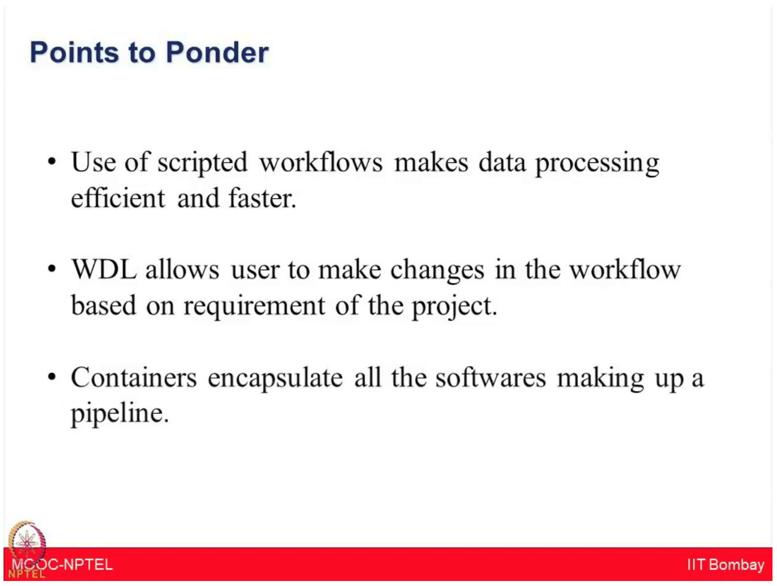
But you could have a older version which was 10.3. For some specific applications you may want to do that for example in the lab tour yesterday we saw Peptide Sequencing machine that was running off a Mac that is about 20 years old. I am sure that cannot run the new version of the operating system. So, you would need a old version. So, you could have dependencies like that. Did that answer your question or?

Student: Generally in the new version already you have the old version if you read and then something extra is there. So, there is no need to add.

So, in the Docker if you just want to add something extra, you can add something extra, but it is still a new Docker because you modified the old one. So, it will be considered a new version.

So, if you think the change was simple enough that you do not need the old version anymore, you can throw it away and just replace it with the new version.

(Refer Slide Time: 25:57)



Points to Ponder

- Use of scripted workflows makes data processing efficient and faster.
- WDL allows user to make changes in the workflow based on requirement of the project.
- Containers encapsulate all the softwares making up a pipeline.

MOOC-NPTEL IIT Bombay

In today's lecture you were introduced to the terminology used to describe automated data processing work flows, key advantages of automated data processing involved reduction of error time and cost. We are also introduced to the concept of Dockers which contain all the relevant versions of softwares and files needed for data analysis and lastly you learnt about how the workflow definition language can be used to provide parallelism and faster speed in cloud computing. The next lecture we will focus on FireCloud, a cloud computing platform from the Broad Institute of MIT and Harvard.

Thank you.