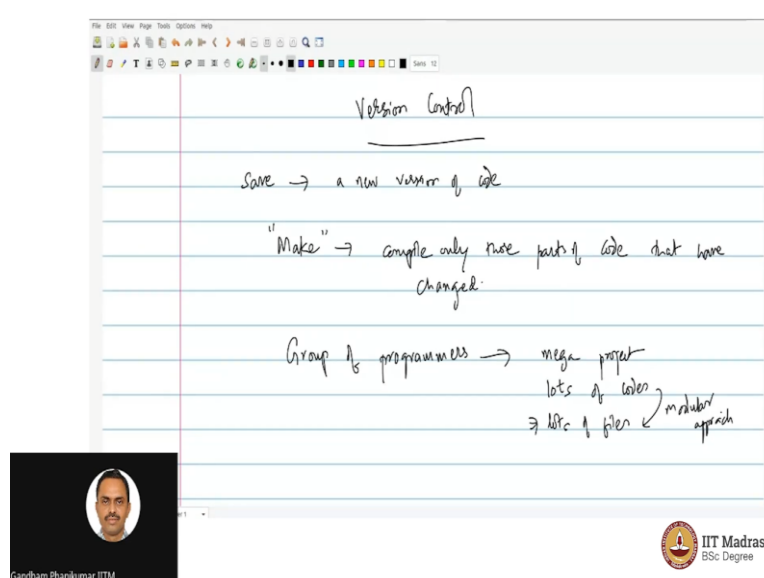


System Commands
Professor Gandham Phanikumar
Department of Metallurgical and Materials Engineering
Indian Institute of Technology, Madras
Version Control - Part 01

(Refer Slide Time: 00:14)



All right. So, let us get started. So, the concept that we are going to talk about now, is called Version Control. So, the idea of course is something that every programmer will have any requirement. Because, so, every programmer knows that each time you save, every save is, basically, effectively a new version of the code.

Now, one technology which the programmers have started using, whenever there are different versions, is the concept of make. So, but, that is, basically, to compile only those parts of code that have changed. So, that allows for at least that as you keep writing your code and modifying, you do not touch what has not been modified. So, it is compiled and that is it. You do not have to go back to look at it.

So, that way there is some discipline about going about programming. Now, but still, this is still in a regime where there is only one person. So, there are requirements then, when there is a group. So, let us say a group of programmers are working and they are working on a major project. So, you, basically, think of a mega project.

And of course, it entails that there will be lots of codes. And that we would like to also, would like to think of it as implying that lots of files. Because, you know that we are going to

have a modular approach towards making our program. So, if it is a modular approach then our code will be broken down. So, typically we would like to have one file per function that we are writing in the program. So, we are talking about that as a C programmer C plus plus program.

Then we would like to have in a C plus plus code, of course, you would like to have a class in one file and in a C language you would like to have each function as a separate file. So, that once you are done with the function and you do not have to go back to it. And why we would like to have a lot of files is, because, of course, there is a group of programmers. So, each programmer can volunteer to work on a particular function, which means it is a particular file. So, there is a tacit understanding that the rest of the programmers are not going to touch it at the same time, typically. And therefore, they are able to work together.

(Refer Slide Time: 03:20)

lots of files
⇒ lots of files ← modular approach

Each programmer has multiple versions of each file they worked on

Version control → trace back to a working version of code

Versions → # users × # files × # version → d/b to hold everything

Gandham Phanikumar IITM

IIT Madras BSc Degree

So, naturally, so, you will have a situation where each programmer has, let us say, multiple the versions of each file we worked on. So, there is not just one version per file, but, every file that they are saving, if they are saving 10 times a day, it means that there are 10 different versions of that file and over the programming duration and there will be a large number of versions.

And, why is the version control necessary? So, the version control, essentially, is to trace back to a working version of code. So, that is that is the idea, basically. So, you have made some changes. So, it seems to work fine and then you made one purchase everything broke.

So, naturally the first thing that you would like to do is, just go back to the version that worked fine and then continue developing by not making the same mistake but making maybe different mistakes, that is how life is. But, of course, we keep improving all this.

We have learned from our mistakes always and we improve. And that is how, we basically create new version. So, the ability to trace back to a working version of a code is must. Now, you can see that this entire scenario entails that the version are going to be like number of users into number of files into number of times you have got the versions.

So, that will be quite a large and you need a database to hold this info. So, because there are a lot of files and a lot of users and there are versions for every file and therefore there is a way by which you have to supposed to see, keep track of all these things. And therefore, we have a requirement to keep it in a good database.

(Refer Slide Time: 05:51)

SVN -> Centrally hosted & managed version system
GIT -> distributed version control system
storage system -> if it fails x
when it fails ✓
RAID

Gandham Phanikumar IITM

IIT Madras BSc Degree

Now, there are some methodologies that were followed in the past to go about this process and I would say there are two ways of handling. So, you can think of these major categories as for example. One is called SVN. So, source version controlling system and you can think of it as a centrally hosted and managed version system.

And of course, we also have GIT, which is, basically, distributed version control system. So, that is a major difference. So, the difference between these is, is it central or is it distributed? Of course, there are people who still use SVN, but there are two major categories that people use and there are pros and cons.

So, SVN allows for one master who actually keeps track of what is the version of the code that we are actually officially supporting. And therefore the master is then going on to essentially ensure that the same version is replicated by others. And therefore, whenever any person has any doubt about that code, you can just go to the master server and bring the version and then that there is a version that everybody is supposed to have. So, there is certain amount of simplicity and clarity that is there in this approach.

But, of course the programs have all to be saved on some storage systems and that the typically the solar systems, when we talk about their failures, typically, we do not save versus storage systems. You do not say, you know, if it fails, no, what we always say is, when it fails.

So, what does it mean? It means that the storage, there is no uncertainty that it will fail. It will definitely fail. Because, there are moving parts and there is some heat generated and therefore definitely there are going to be failures in the hardware. So, the problem, our analysis should never be, if it fails.

Because, the if is not, you know a possibility of probability because the probability is 100 percent. Because, it is only a matter of time and therefore you are supposed to ask, when it fails what would we do? So, in that situation, if you have a master server on which the code is stored then when it fails, you have a problem. You have a problem because the entire software is down. Nobody can actually access what is the master version of the source code.

Of course, people can take into account that the storages can be failing and therefore they may have multiple storages and they may have a way by which if one hard disk fails the others are also able to chip it. So, does anybody know? I mean this is something that I want to have a little bit interaction. We are going quite fine with the syllabus; we are going to finish quite soon.

So, I want to also sort of finish up some loose ends, as I go along. So, upon (9:06) we will come to the concept of branch, we will come to it little later, because the idea of a branch exists only in GIT, not in SVN. Because, in SVN, the concept, I mean the word called branch has no special meaning like the way we talk about it in GIT.

Because, it has a much more connotation. There is lot more to it; the word is a very loaded word. So, we will come to it in a moment. Now, I want to ask you, what is the technology

used by people around the world to ensure that if one hard disk fails, it does not actually fail the storage that you are still able to access the file? Can somebody volunteer and say what that could be? Anyone? I am sure in a large class like you almost 50 students around, definitely, there will be one person who has heard of it. Backup is a separate matter.

So, you can always take it by backup. But, when the server crashes, I mean, the storage box crashes then it is down. So, what I am saying is that if there is a hard disk, you see, if each hard disk is a separate hardware entity, it is a unit that you are going to plug in and give power. So, if that unit actually fails, you do not want the ability to access your file to be lost.

Backup is always something like, when you have burnt your hands, you go and apply some ointment. It is like that. The problem has already occurred. And then you go and see mitigate the problem to the extent by bringing up a version that was there earlier. That is something secondary. But, to even avoid hardware failures, what is the technology that has been in place for the storage systems across the world? And it is a standard actually. No? Nobody yet? So, I will give you a clue and I will ask you. So, what is this? Anybody, anybody knows what is RAID? Shobhan says yes. Okay. Now, go ahead and tell Shobhan, what do about read.

Shobhan (Student): RAID is the old versions but they still use it, right.

Professor Gandham Phanikumar: Yeah, of course, there are multiple versions of RAID. So, the number that follows RAID, RAID 5, RAID 6. So, that versions are always there, but the idea of RAID is something that I want you to be aware.

(Refer Slide Time: 11:57)

The image shows a screenshot of the Wikipedia page for RAID. The page title is "RAID" and the subtitle is "From Wikipedia, the free encyclopedia". The article text begins with a disambiguation note: "This article is about the data storage technology. For the police unit, see RAID (French Police unit). For other uses, see Raid (disambiguation)." The main text defines RAID as a data storage virtualization technology that combines multiple physical disk drive components into one or more logical units for the purposes of data redundancy, performance improvement, or both. It mentions that this was in contrast to the previous concept of highly reliable mainframe disk drives referred to as "single large expensive disk" (SLED). The article also states that data is distributed across the drives in one of several ways, referred to as RAID levels.

Overlaid on the bottom left of the screenshot is a video frame showing a man, identified as Gandham Phanikumar IITM, speaking. The IIT Madras logo is visible in the bottom right corner of the video frame.

Let us go ahead and then look at it quickly. Because, I do not want again to take too much time. So, I would say RAID for storage. So, this I will just tell you, what those numbers are? That is all. So, RAID is, basically, a way by which you would like to have a redundancy. So, that means you would like to have a copy of your file in multiple disks. So, that if one disk fails, you still have file available to you that was the idea basically.

And, obviously you should have multiple disks. Because, you are talking about if one disk actually has a problem, there is another disk that is responding to you. So, that means there must be multiple disks and therefore that is the idea of array. So, the concept of array is because you need multiple disks. And in terms of the second, third letter i. So, is it independent disks or inexpensive disks? Of course, it up to us, for us to interpret, but it is now, today RAID is actually almost like a word, you know, I mean people do not expand it.

(Refer Slide Time: 13:11)

RAID (^[1][reɪd](#); "Redundant Array of Inexpensive Disks"^[2] or "Redundant Array of Independent Disks"^[2]) is a data storage virtualization technology that combines multiple physical disk drive components into one or more logical units for the purposes of data redundancy, performance improvement, or both. This was in contrast to the previous concept of highly reliable mainframe disk drives referred to as "single large expensive disk" (SLED).^[3]^[4]

Data is distributed across the drives in one of several ways, referred to as RAID levels, depending on the required level of redundancy and performance. The different schemes, or data distribution layouts, are named by the word "RAID" followed by a number, for example RAID 0 or RAID 1. Each scheme, or RAID level, provides a different balance among the key goals: reliability, availability, performance, and capacity. RAID levels greater than RAID 0 provide protection against unrecoverable sector read errors, as well as against failures of whole physical drives.

Contents [hide]
1 History
2 Overview
3 Standard levels

Gandham Phanikumar IITM

IIT Madras
BSc Degree

Now, there are different versions of it. So, here you see RAID 0, RAID 1, etcetera. So, each version has certain names.

(Refer Slide Time: 13:24)

The screenshot shows a presentation slide with text on the left and an image on the right. The text describes the Storage Networking Industry Association (SNIA) in the Common RAID Disk Drive Format (DDF) standard. It explains RAID 0 as striping without mirroring or parity, noting that its capacity is the sum of the drives in the set, but a single drive failure results in data loss. It also mentions that RAID 0 increases throughput by allowing concurrent reads and writes. RAID 1 is described as data mirroring without parity or striping, where data is written identically to two or more drives, creating a 'mirrored set'.

Storage Networking Industry Association (SNIA) in the Common RAID Disk Drive Format (DDF) standard:^{[16][17]}

RAID 0 consists of striping, but no mirroring or parity. Compared to a spanned volume, the capacity of a RAID 0 volume is the same; it is the sum of the capacities of the drives in the set. But because striping distributes the contents of each file among all drives in the set, the failure of any drive causes the entire RAID 0 volume and all files to be lost. In comparison, a spanned volume preserves the files on the unfailing drives. The benefit of RAID 0 is that the throughput of read and write operations to any file is multiplied by the number of drives because, unlike spanned volumes, reads and writes are done concurrently.^[14] The cost is increased vulnerability to drive failures—since any drive in a RAID 0 setup failing causes the entire volume to be lost, the average failure rate of the volume rises with the number of attached drives.

RAID 1 consists of data mirroring, without parity or striping. Data is written identically to two or more drives, thereby producing a "mirrored set" of drives. Thus, any read request can be

Storage servers with 24 hard disk drives each and built-in hardware RAID controllers supporting various RAID levels

Gandham Phanikumar IITM

IIT Madras BSc Degree

Now, the idea of RAID 0, you can just see the here. So, what the RAID 0 is, basically, to ensure that you have a large disk and you do not have one disc that can have that same capacity and therefore you would like to have a two discs that are kept one after other, therefore you have got, you know, ability to have a larger disc. Let us say, you wanted a 20 TB disk, but in the market there is only 10 TB disk. So, you can get two of them.

(Refer Slide Time: 13:55)

The screenshot shows a presentation slide with handwritten notes in blue ink. The notes define SVN as a centrally located & managed version system and GIT as a distributed version control system. They compare a storage system where a failure results in data loss (marked with an 'x') to a RAID system where a failure is handled (marked with a checkmark). A diagram shows RAID leading to an Array, which then leads to 'Speed up!' and 'Redundancy'. A note indicates that 6 GB/s can be achieved with 18 GB/s.

SVN → Centrally located & managed version system
GIT → distributed version control system

Storage System → if it fails x
when it fails ✓

RAID → Array → Speed up!
Redundancy

6 GB/s → 18 GB/s

Gandham Phanikumar IITM

IIT Madras BSc Degree

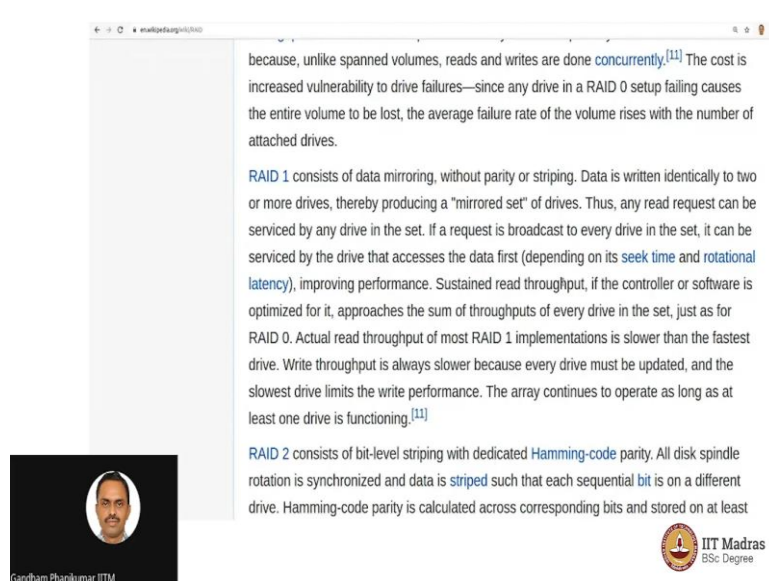
Now, one thing I want you to also remember is, whenever you have an array, it also means that speed up. So, whenever you have got any file that is actually stored over multiple disks and you are going to read it. So, each disk can respond for different parts of the file and

therefore you actually can get faster speed. So, this is not only about the redundancy, but it is also about the speed. So, both are necessary.

Now, what is the kind of speed we are talking about? So, for SATA disks, 6 GB per second is, what we are talking about. And let us say you would like to have jacked up, let us say, to 18 GB per second. So, the you cannot actually go and buy a faster hard drive because they just simply do not exist. So, the way to achieve is because you have a concept of RAID.

So, you can put multiple disks and therefore when you are reading a file, it can actually come from three different disks, three parts of the file can come from three different disks and therefore the effective speed that you are gaining from the system is actually 18 GBPS and not 6 GBPS and therefore you are actually getting the speed up. So, this is also an important feature of this technology.


(Refer Slide Time: 15:15)




because, unlike spanned volumes, reads and writes are done **concurrently**.^[11] The cost is increased vulnerability to drive failures—since any drive in a RAID 0 setup failing causes the entire volume to be lost, the average failure rate of the volume rises with the number of attached drives.

RAID 1 consists of data mirroring, without parity or striping. Data is written identically to two or more drives, thereby producing a “mirrored set” of drives. Thus, any read request can be serviced by any drive in the set. If a request is broadcast to every drive in the set, it can be serviced by the drive that accesses the data first (depending on its **seek time** and **rotational latency**), improving performance. Sustained read throughput, if the controller or software is optimized for it, approaches the sum of throughputs of every drive in the set, just as for RAID 0. Actual read throughput of most RAID 1 implementations is slower than the fastest drive. Write throughput is always slower because every drive must be updated, and the slowest drive limits the write performance. The array continues to operate as long as at least one drive is functioning.^[11]

RAID 2 consists of bit-level striping with dedicated **Hamming-code** parity. All disk spindle rotation is synchronized and data is **striped** such that each sequential **bit** is on a different drive. Hamming-code parity is calculated across corresponding bits and stored on at least



Gandham Phanikumar IITM



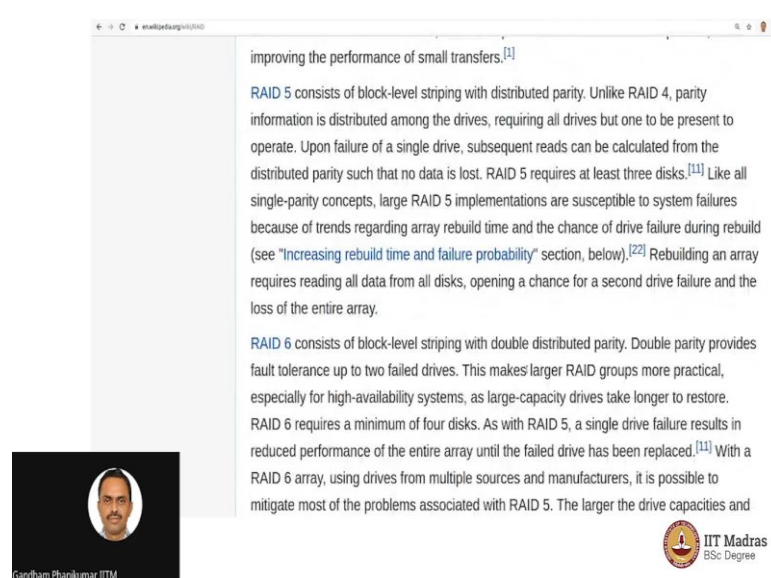
IIT Madras
BSc Degree

Now, as your colleague Shobhan was asking about the version old and old technology etc. So, here RAID 1 for example, is today a very continuously being practiced version of RAID and that is because you have got mirroring. Now, which means that in fact, in my computer I have got two hard disks of 500 GB each, which are basically in RAID 1, which means that if one of them goes off then other disk is there. So, I am free from the failure of hard disk.

Of course, the probability of a two disk simultaneously having a problem is very rare and to that extent, I am actually having safety. So, generally (())(16:02) is asking about the actual disk or virtual partition, they are actually on disks itself. So, once the disks are actually made

as RAID then you create a volume, logical volume and it is a logical volume that you see as a partition. So, think of the partition is as an abstraction level that is after RAID has been configured. So, it is not before but it is after.

(Refer Slide Time: 16:30)



improving the performance of small transfers.^[1]

RAID 5 consists of block-level striping with distributed parity. Unlike RAID 4, parity information is distributed among the drives, requiring all drives but one to be present to operate. Upon failure of a single drive, subsequent reads can be calculated from the distributed parity such that no data is lost. RAID 5 requires at least three disks.^[11] Like all single-parity concepts, large RAID 5 implementations are susceptible to system failures because of trends regarding array rebuild time and the chance of drive failure during rebuild (see "Increasing rebuild time and failure probability" section, below).^[22] Rebuilding an array requires reading all data from all disks, opening a chance for a second drive failure and the loss of the entire array.

RAID 6 consists of block-level striping with double distributed parity. Double parity provides fault tolerance up to two failed drives. This makes larger RAID groups more practical, especially for high-availability systems, as large-capacity drives take longer to restore. RAID 6 requires a minimum of four disks. As with RAID 5, a single drive failure results in reduced performance of the entire array until the failed drive has been replaced.^[11] With a RAID 6 array, using drives from multiple sources and manufacturers, it is possible to mitigate most of the problems associated with RAID 5. The larger the drive capacities and

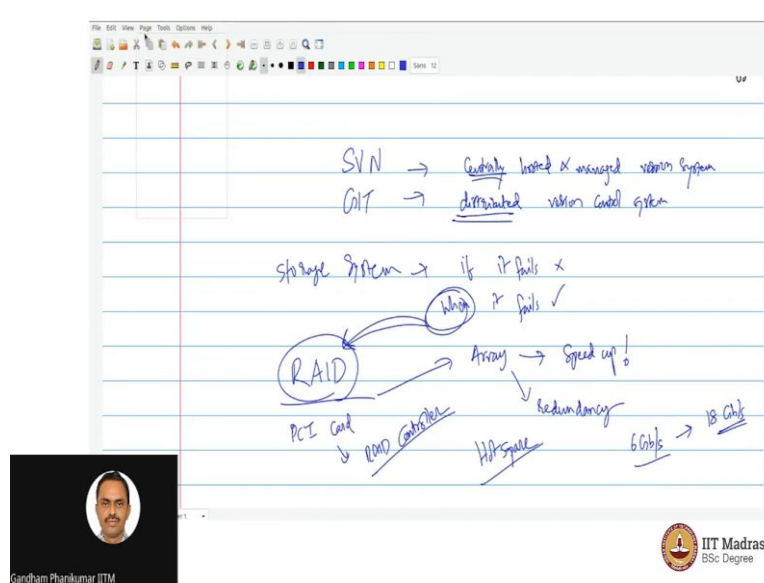
Gandham Phanikumar IITM

IIT Madras
BSc Degree

Now, there are some RAID options that you may be interested. So, today RAID 5 and RAID 6 are quite popular. So, RAID 5, which actually has both distributed as well as some amount-distributed parity as well as some striping. What it implies is that when you have RAID 5, when you do on three disks, it means that if one disk goes off, absolutely, there is no loss of data.

There is no loss of data. And you can also get the speed up because the file is coming from three disks instead of one disk. So, today we have got RAID 5 and RAID 6 very popular to have these things. Higher versions of RAID are also available. And there are much larger concepts of it when you go to huge storages like petaflop, petabyte storages etcetera, where they have got you know Hadoop and distributed storages etcetera.

(Refer Slide Time: 17:31)

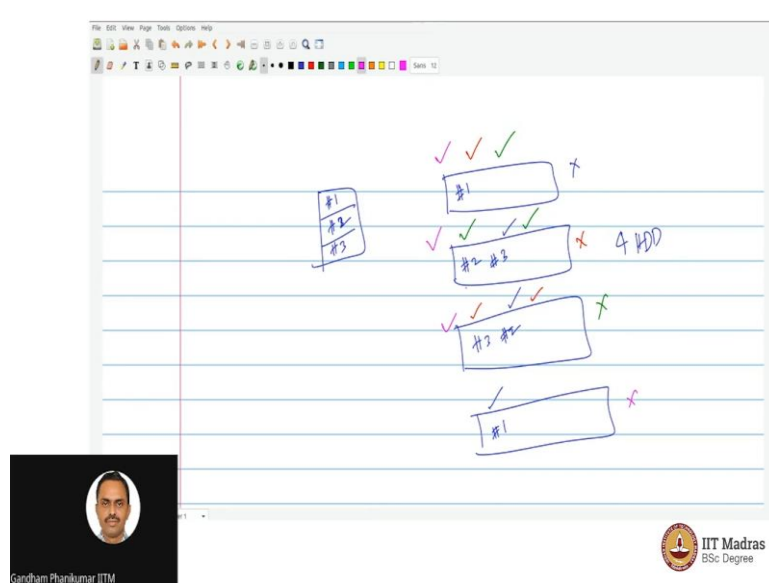


And, what I want you to know today is that when it comes to this, what, when it fails is sort of addressed by RAID, that is what I wanted you to have heard of it. It is address by RAID. So, when you buy a workstation, just check, does it contain a RAID controller? So, there is a card, there is a PCI card, which is called as RAID controller. And this controller will have a particular speed and that speed actually tells you how fast you can get the data and. So, and also it will tell, it will have a capacity of, how many disks it can actually talk to.

So, for example, my workstation is a small workstation for my office. So, it has about eight disks that it can talk to. So, I have populated all of the eight and I have got what is called as a hot spare. Hot spare is basically a disc which we keep plugged in but it is not being used. So, if any of the existing disks fail then automatically the hot sphere will be put to use and I will have an amber light on the failed disk. So, I can safely take it out and the hot spare actually be.

Now, configured in automatically, so that it is participating in the shared storage of the files. So, we cannot plug out the disks randomly, we have, once we configure them in the RAID, we have to not touch them. And whenever there is an error, we have to immediately replace the disk. But, if you do, that advantage would be that your storage can actually live for very long. Because, as the disks fail, you can keep replacing and your data is never getting lost. Because, think of it like this.

(Refer Slide Time: 19:18)



So, it is like this you have got a file. So, if you make it as two parts, three parts and let us say you have got the storage in this manner, let us say. So, four hard disks. So, I will put the part 1 here, part 2 here, part 3 here and what I will do, I will put part 1 here also and I put part 2 here also and I put part 3 here also. Now, you see that I have got, if let us say, this fellow goes off, then I have got 1 here, 2 here, 3 here. And let us say, if this fellow goes then I have got 1 here, 2 here and 3 here.

And let us say, I have got this fellow going, I have got 1 here, I have got 2 here, I have got a 3 here and then let us say, if this fellow goes, then I have got 1 here, 2 here, 3 here. So, you can see that I mean this is a very simple, you know, illustration to show you that if I just divide it by three parts and arrange them in this manner, if one of the four hard disks fail, I have no problem at all, because I have got portions of it in other hard disks. So, I can recoup.

So, if one of them fail I can replace it with a fresh hard disk and then the system will automatically know which part actually has been replaced and it will go on to write that part alone, so that the RAID configuration is back to normal. So, which means that the hard disks are actually live, they are being written all the time and the computer is analysing, which part of which file is in which hard disk and ensuring that that particular hard disk is alive. And there are at least two copies of every chunk. I would not say two copies of every file, but two copies of every chunk.

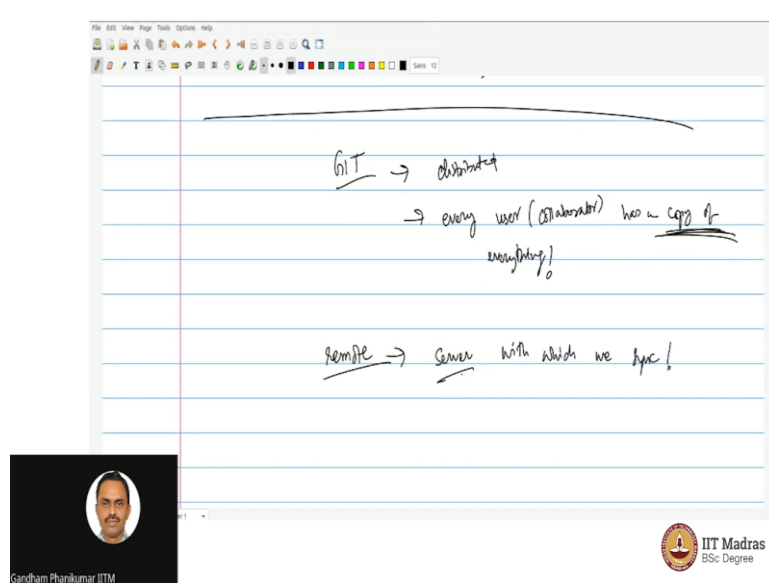
(Refer Slide Time: 21:09)

The screenshot shows a presentation slide with a white background and blue horizontal lines. At the top, there is a menu bar with options: File, Edit, View, Page, Tools, Options, Help. Below the menu bar is a toolbar with various icons. The main content area contains handwritten text in black ink. The text reads: "In a RAID: Usable space < actual space". Below this, there is a calculation: "26 TB" is written on the left, and "4 x 8TB" is written on the right, with a vertical line between them. Below "4 x 8TB", "32 TB" is written. In the bottom left corner, there is a small circular video feed showing a man with a beard and glasses. Below the video feed, the text "Gandham Phanikumar IITM" is visible. In the bottom right corner, there is a logo for "IIT Madras BSc Degree".

So, which means that which actually means that in a RAID, RAID config usable space is less than actual space. So, for example, if you have got a 4 into 8 TB disks, you have 32 TB disk space. But, when you configure as a RAID 5, you may have about 26 TB or so. So, the remaining data is lost because you are actually repeating some portions. So, about one third of what you have stored is actually repeated and therefore you cannot have that but that comes with a positive thing that one hard disk crashes and you have no problem.

So, much about RAID and go on to read about it for your own benefit. I do not need to talk about that. So, next we will go and say, now, this hard disk failure is not a problem, we have controlled it, but what happens when the server itself is hacked and actually the data loss. So, therefore SVN has that still problem, because, it may actually still keep the data but if somebody hacks in and wipes off then you have a problem.

(Refer Slide Time: 22:25)



So, for that the GIT actually has a very good idea because it is distributed. So, which means that if the master, I mean, if at all, if you want to call something as a master, if the master server is actually suddenly disappeared from the face of the earth, nothing is lost because what this implies is that every user or I would say collaborator has a copy of everything.

Now, obviously program codes are not very large. So, therefore it does not actually take so much of space we are talking about, maybe few megabytes or maybe 100 mb of your programming codes and therefore it does not matter if everybody has a copy. But, it gives a redundancy and therefore if any user wants to go on, independently, they can actually start from where they were and then start a new thread and a new version of that particular program. And you can think of it as a fork or a new branch.

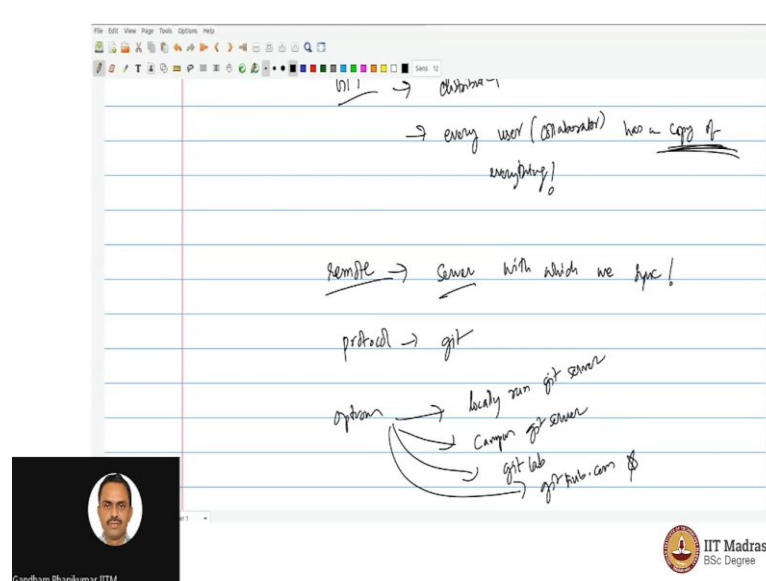
So, both are having technical differences, we will come to that in a moment when we actually use. So, a user can actually go on to have their own version of developing things independently and therefore that also is more democratic. So, it is that way more preferred way of going about.

Now, it also does not really require a server. The GIT system does not require a server because when you are alone, you yourself can use GIT and you can just simply have the version control within your own machine. And you are the only person and that is all it is. So, that is also possible. But, it is a good idea to also have some server.

Now, the GIT system is open source, which means that you have got the GIT system enabled for all operating systems and you also have some free software, a few free servers that are allowing you to use it as a one of the synchronizing remote spots. So, in GIT system, there is some concept of remote. So, that means that it is basically some server with which we, basically, sync.

Of course, there is no need to do that of course. But when we want to sync because we would like to together maintain some master version, to that extent only we want to sync up, but each user has a copy. So, that is the idea. So, it is still not as, if the remote machine is down, you do not have access to the master version. That is not true. So, you have got the version with you, at least as on the last synchronization that you have done. So, by remote we mean a server with which we try to synchronize.

(Refer Slide Time: 25:18)



Now, there is a way by which we connect and the protocol is what is called as a GIT protocol and that is now, because it is open source, it is now adopted by people across the globe and therefore there are many many things that actually use the concept of GIT. But, do totally different things, you know, very unlike a programmer also. That is also fine, there is no problem. Because, it is actually a protocol by which you are actually exchanging information with a remote. And also keeping the version control. There is only two aspects of it that we need to, basically, think of.

Now, you can have many many different options of using it. So, one, of course, is locally run GIT server. You may have, for example, a campus GIT server. You may have a GIT lab,

some remotely hosted GIT server or you may even have github.com. Now, I prefer that you start getting used to the github.com for multiple reasons.

One reason is that it actually is a cloud and which means that you do not have to worry about how it is storing files and who is paying for it etcetera, because it is, basically, taken care by Microsoft and the files are all meant for public good of a whole world because it contains lot of free software that people are actually using to share and all that is actually going to be there and they also have taken a copy of GitHub as on last year and kept it in arctic vault, an underground storage system.

So that even if there was a huge bomb attack on the data centres in America, you do not still lose your files because there is one copy of it somewhere kept aside. So, for the future of humanity and, so there are all those kinds of things that can be done by a company and that is what is the advantage. So, starting to learn to use GIT with github.com has some specific advantages.

(Refer Slide Time: 27:35)

Aug 11 to ? **Two Factor Auth!**

one more way

- App → OTP → enter
- App → QR → Super
- SMS → OTP → enter
- Email → OTP → enter

Set up later → enter

Phone #

IIT Madras BSc Degree

Gardham Phankumar IITM

Now, there is other things that I want to also tell you why you should learn is for this that now, github.com insists from august 11th onwards I think, I think the concept of two factor authentication. So, have you heard of two-factor Auth? Anybody wants to tell what is two-factor authentication? You may have done it already for google account, because it is a good habit actually. So, that you do not lose your password easily or lose access to your account easily.

So, anybody wants to tell, what is two-factor authentication? Raghwan may be knowing it. So, Shobha says you need another verification to log into the account, apart from password. Yeah, that is about it. Succinctly, that is all it is. So, there is must be one more way. So, what does it mean by one more way.

So, this is our, you may have an app and it will be telling you some OTPs and that you need to enter or you may have an app that will ask you and you swipe and confirm and are you may actually have an SMS that is coming to you as an OTP and that you enter or you may have an email which actually contains the OTP and that you enter or you may have a set of codes which you actually use it to enter and then authenticate.

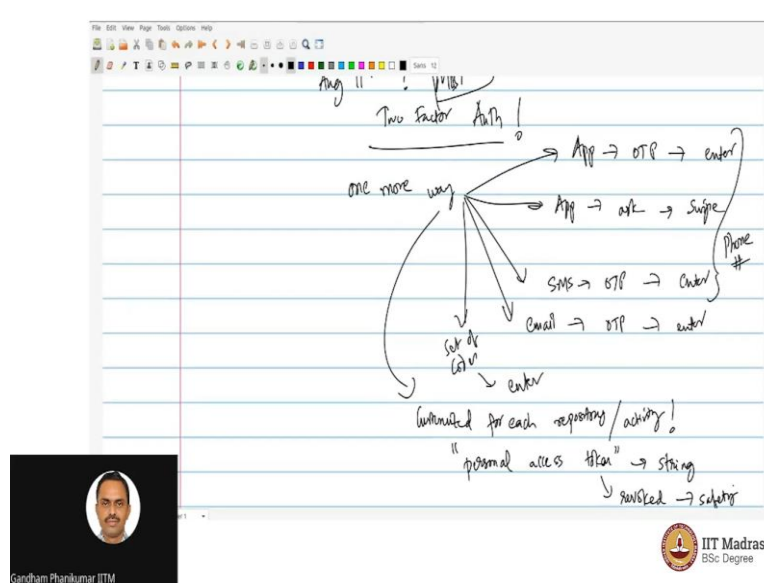
So, set of codes which are somewhat like recovery etcetera and whenever you say an app and an OTP and all that so, most of these things, for example, up to here, you have a phone number involved, so, which means that there is a certain way by which we are trying to see, who is it, who is the person, who is actually using this account. And that person is identified by the phone number uniquely because you are sending something and somebody is checking that number and entering and it is it is being done in a way that you know requires some amount of human action.

So, preferably, so that is one way by which you can actually ensure. So, you do not lose both. So, you do not give away your password and also you hand over your phone to somebody. So, anybody who knows your password and has your phone is, basically, you basically. It is as good as you and therefore that is like total loss of identity and we do not do that.

So, we have only one of them losing at a time and the immediate other one is blocked. As soon as you lose a phone, you go and go to the app and in the website and then tell that you know my phone is lost or you just simply change the phone number to something else and therefore the person who happens to gain access to your phone, does not have actually the ability to mimic who you are. You can understand the things on the internet. So, these are the ideas.

So, the two-factor authentication is a must. What we mean by a must is from august onwards, it is a must. So, by that we mean that you simply cannot have only the password to actually authenticate, you must have two factor authentication enabled and you should use it while doing some actions on the command line and that is good.

(Refer Slide Time: 31:14)

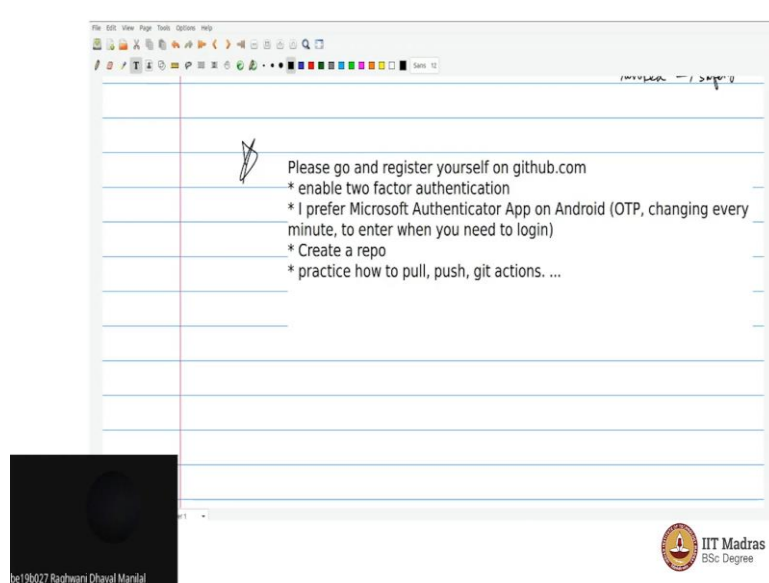


One of the things of all these is that you also have it the second factor authentication is sort of customized for each, I would say, repository or activity. So, why I say activities? Because, in Google, for example, if you are going to have an account, for each machine on which you are going to have the authentication happening, for example, you are configuring thunderbird on your machine and you are going to talk to your Google email account then you can actually have a what is called Personal Access Token PAT.

So, you can have your token, which is basically a long string. So, you say, basically, a string and you can use that string as a password and you can actually have that specifically for certain activities, in the case of Gmail, or for repositories in the case of GitHub. So, that you do not actually have the same thing for multiple repositories and therefore if you lose you still do not need to worry about it much, it does not cause so much of damage.

Of course, you can immediately go and you know, block access that is always there, you can always revoke. So, these tokens can be revoked in a click. And that is what is actually gives you safety. So, if you lose something does not matter. You can always revoke the privileges and go ahead.

(Refer Slide Time: 33:00)



So, I want you to know about this and go ahead and actually log into GitHub and start doing it. So, the activities I would like you to do are, I would say, login, please, go and register yourself on github.com. And of course, next step I would like you to do is that enable two factor authentication and of course, you need some way of enabling and I would say for my side I would say prefer a Microsoft authenticator app on android and that will give you those OTPs changing every minute to enter then when you need to login.

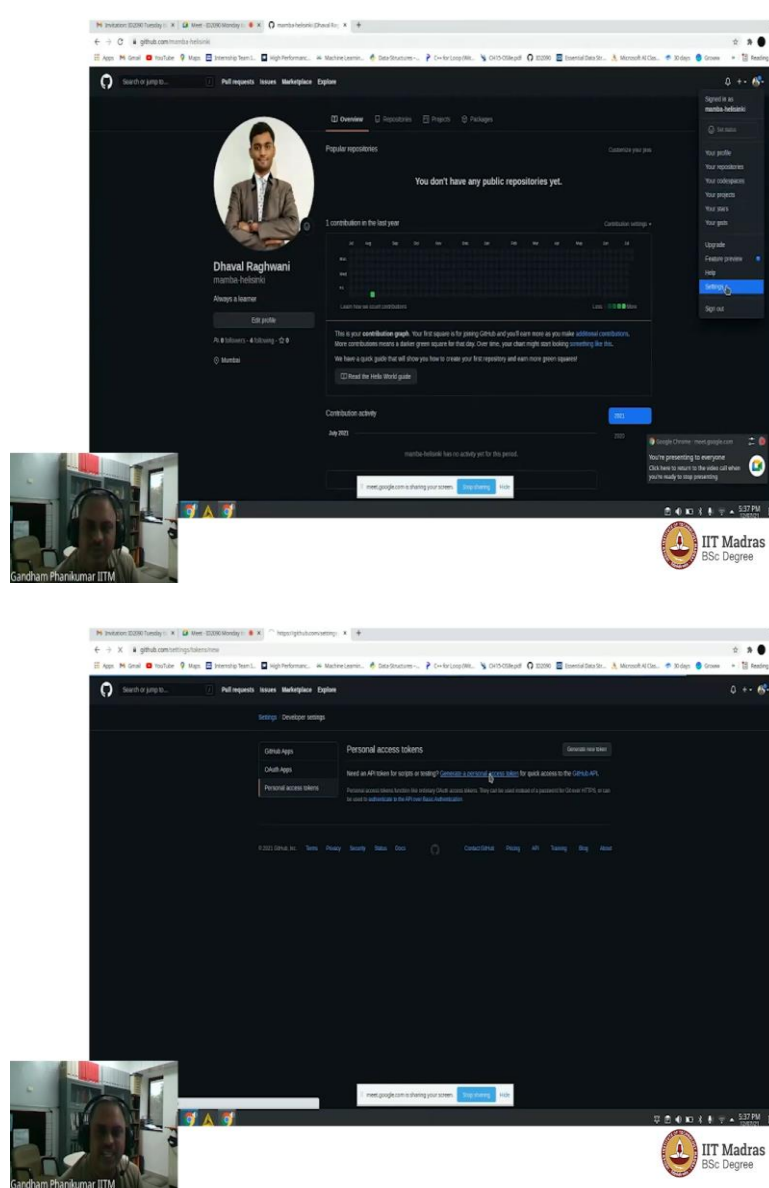
So, I will do the demo of it in a moment. And of course, once you do that, of course, you need to create a repo and you need to practice how to pull, push and do all the GIT actions. So, we will do that. We will do all of these in a moment. So, I think, there are not too many students, who have been using it. So, if somebody is actually started to use GIT then I would like them to share their screen.

So, that I can actually navigate and show, where the options are. So, I just simply now stop presenting and I will ask one of you, who has a GitHub account to see. So, can you share anybody who has a, let me just see, anybody volunteer to show there, you know, ability to log into GIT, github.com? I do not expect every one of you to do it, obviously, because then I do not need to have this session.

But, I am sure that in 50 students there will be at least one, who would have already started to use GIT. So, anybody who has used GIT? Raghavani is, good, go ahead, go ahead. I thought we also discussed that we would like to have you do that. So, go ahead. So, share your screen, browser, yeah.

Raghavani (Student): I have just created the account. I do not know how to use it.

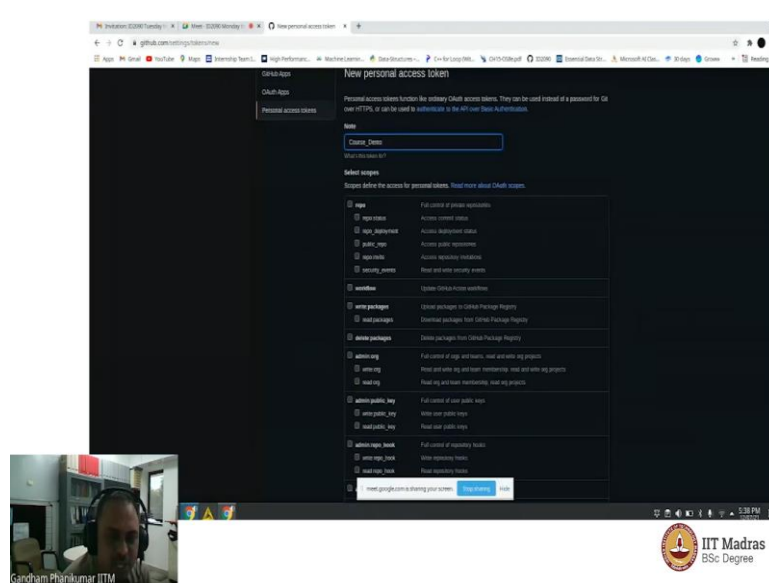
(Refer Slide Time: 35:26)



Professor Gandham Phanikumar: No problem, no problem, that is what I would actually guide you. So, on the right-hand side top, there is your photo and that when you click on it, there are a bunch of things that are coming up. So, go down to settings. Now, on the left, you can scroll down and developer settings, just click on the developer settings.

So, then personal access tokens is there, right. Click on it. So, now, we need to basically create a new token and with that you can actually go ahead and use it. So, you can click on that to generate a new token. There is an icon there, right side, right top. So, obviously, it is asking you to log in once more or confirm who you are.

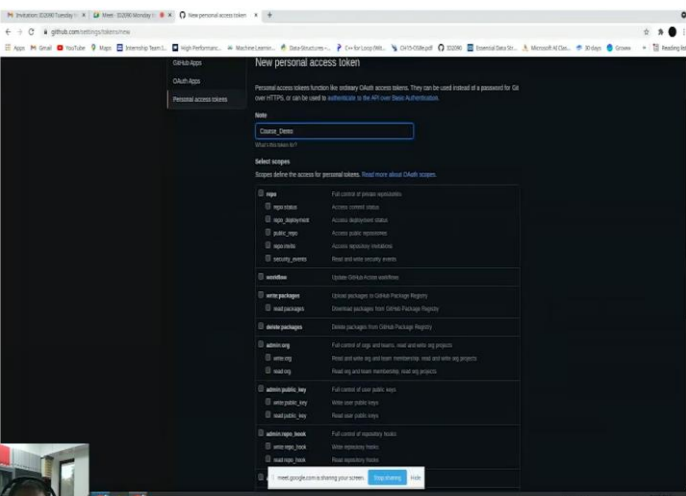
(Refer Slide Time: 36:32)



So, now you need to describe what is this token for, let us say. So, you can say a course demo or something like that. Let us say course demo. And then you have to tell, what are the actions for which you are going to use this token. Now, you see this is a place, where you are controlling, what actions are possible for each of these tokens. So, the token that you are giving, you do not want to do anything about the administration or deleting etcetera, nothing.

So, just click on the repo, the very first box. And obviously, it has got a lot of actions, five of them. So, even, you can even remove that repo invite from there. So, you do not need that because you are not going to invite others as of now. Keep it, keep it. Fine, leave it. Now, you have come down and you can, this is sufficient. There are a lot of actions that are possible. And you can say generate token. Click on generate token.

(Refer Slide Time: 37:34)



New personal access token

Personal access tokens function like ordinary GitHub access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Name

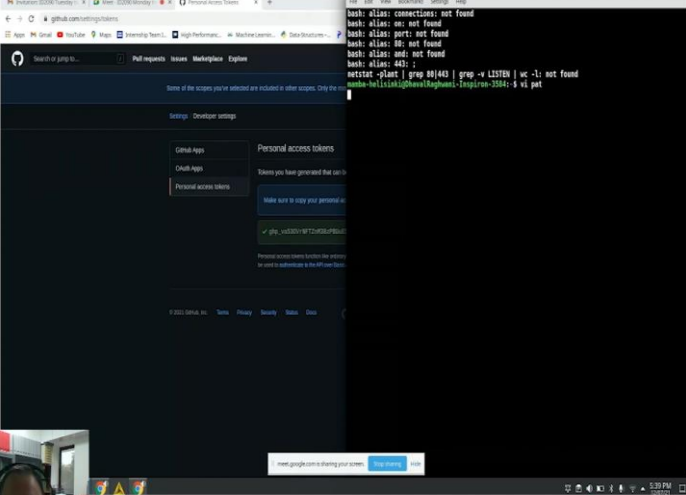
Course_Demo

Select scopes

Scopes define the access for personal tokens. Read more about GitHub scopes.

- ☒ repo: Full control of private repositories
 - ☐ repo:status: Access commit status
 - ☐ repo:commit_status: Access commit status
 - ☐ repo:pull_request: Access pull request
 - ☐ repo:read: Access repository metadata
 - ☐ security_events: Read and write security events
- ☐ workflow: Update GitHub Actions workflow
- ☐ admin:package: Manage packages in GitHub Package Registry
- ☐ read:package: Download packages from GitHub Package Registry
- ☐ delete:package: Delete packages from GitHub Package Registry
- ☐ admin:org: Full control of org permissions, and org-wide org projects
 - ☐ org:org: Read and write org and team membership, read and write org projects
 - ☐ read:org: Read org and team membership, read org projects
- ☐ admin:public_key: Full control of user public keys
 - ☐ user:public_key: Write user public keys
 - ☐ read:public_key: Read user public keys
- ☐ admin:repo_hook: Full control of repository hooks
 - ☐ user:repo_hook: Write repository hooks
 - ☐ read:repo_hook: Read repository hooks

Generate token

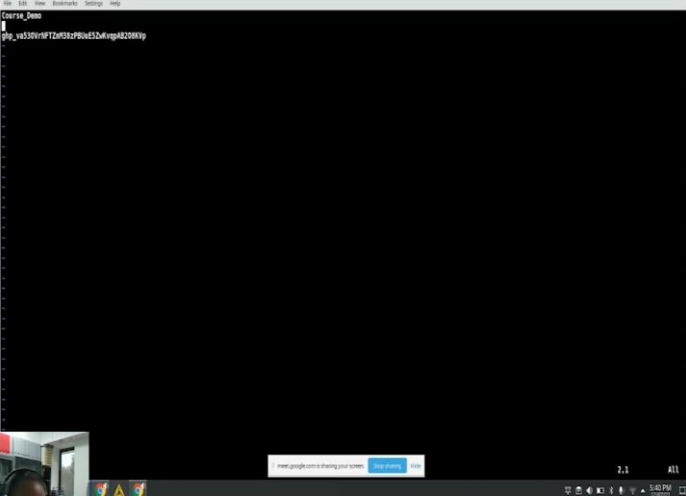


Personal access tokens

Tokens you have generated that can be used to authenticate to the API over Basic Authentication.

Make sure to copy your personal access token.

ghp_v4330r1wF7Zw03zP9u6E3ZuXvg4200K9



```
Course_Demo
ghp_v4330r1wF7Zw03zP9u6E3ZuXvg4200K9
```

```
bash: ghp_v4330r1wF7Zw03zP9u6E3ZuXvg4200K9: command not found
```

Now, be careful. Now, everybody is seeing this token. So, it does not matter. We are already learning. So, if you click on that the keyboard icon, which is, if you click on it, it will copy, copy it and immediately open a text file and save it to the text file. So, that you have you do not forget it at this moment.

So, you can open a vi editor or you can text file you can just open vi editor and save it. So, normally I would actually call that as a PAT, the files called PAT is a personal access token file, basically, in which I can just have that thing. So, vi pat and then you put the, put I and then. Now, just go above one, shift O, you press shift O,

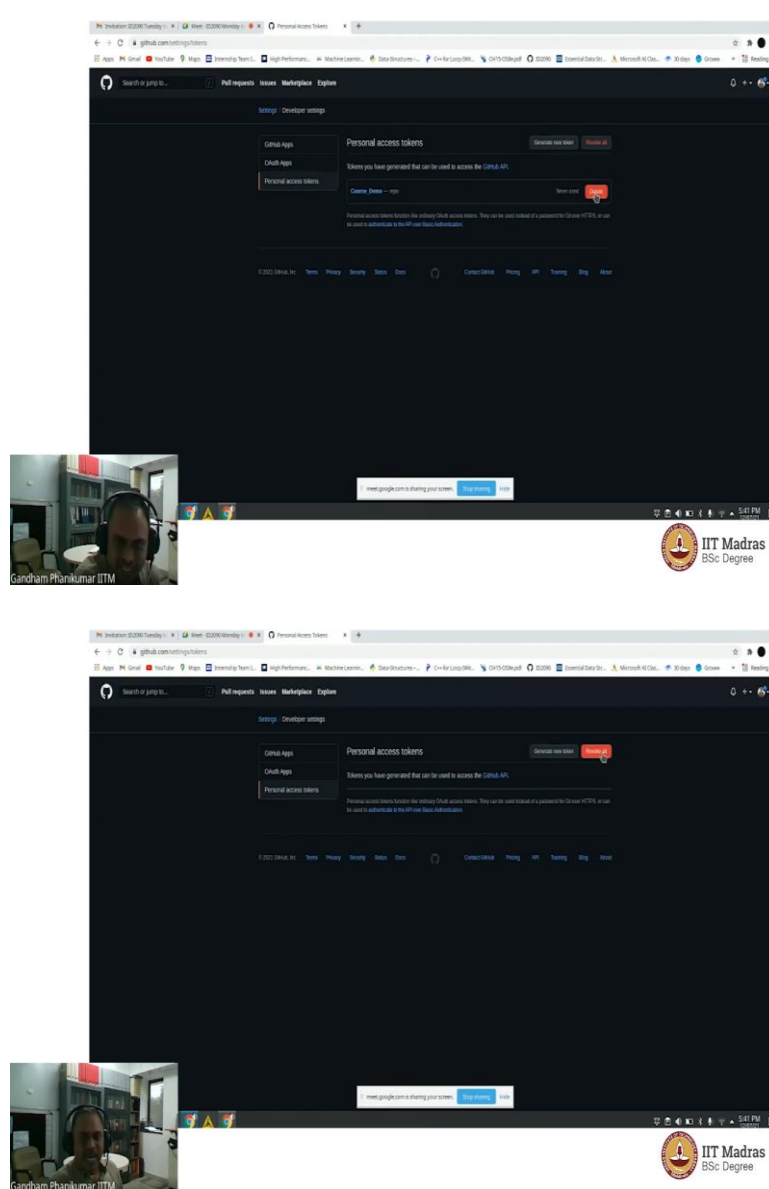
Raghavani (Student): shift.

Professor Gandham Phanikumar: O, shift capital O. No, no, while you are in the vi editor, shift O. Do not insert, just escape. Now, you write what was this meant for. So, you said, no course demo.

Raghavani (Student): Yes

Professor Gandham Phanikumar: Write that name there. So, you understand that this token was for that particular thing and that is just for you to get used to it. Later on, there are technologies to enable these tokens to be stored by a ring, a key ring, and then you do not need to actually see them at all. You can have them populated directly by the key ring to that GIT directly. You do not need to worry right now. So, you try to save it. Now, you have got to your desktop.

(Refer Slide Time: 39:14)

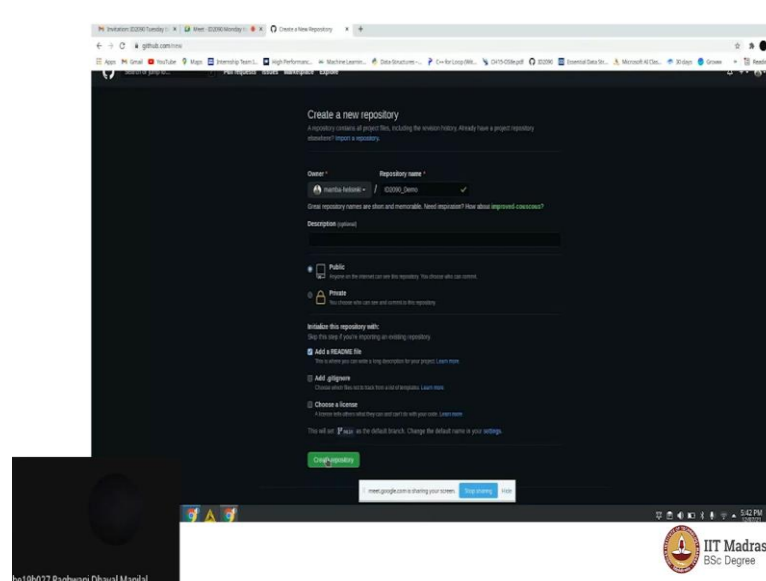


So, you can just go to the browser again and then what you can do is that you have, now, what happens is if you click anywhere else then you cannot see the token. So, you can just click on; let us say, GitHub apps, just click on GitHub apps. Now, come back again personal access tokens.

Now, you see that you have got a token there and by default, you cannot see it and of course, you can delete it. You can delete that, if you want. So, that is one way by which you can have ability to just revoke it. Now, let us say, you have got a severe compromise on something, where your GitHub passwords are stored and that laptop has been stolen or something, what you would normally do is, come here and say revoke all.

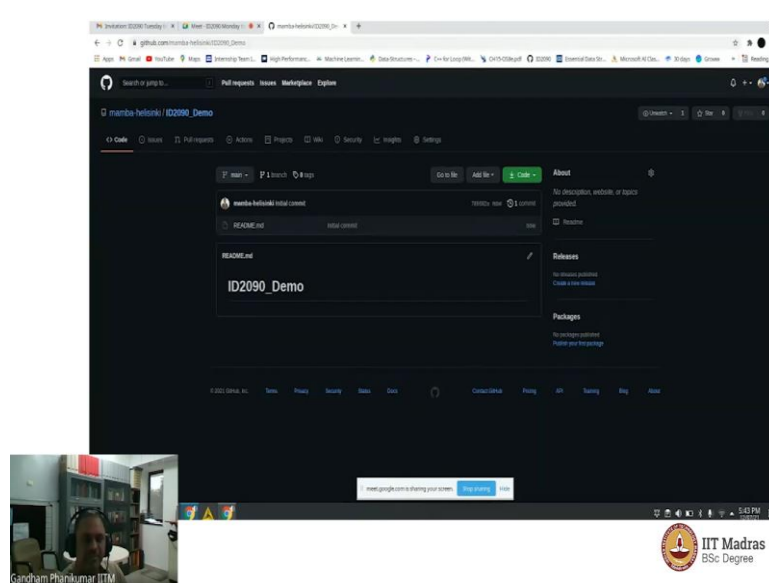
Because, you do not want your permissions on any machine at all. So, therefore nobody can use your GitHub access at all. That is one advantage. And of course, you can then create the personal access tokens as you like and keep using them. Now, that long string is what you are going to use it as a password, basically. You do not give your own password. You give that long string as a password. That is a, that is the thing that I want you to know. Now, once you have that idea coming up.

(Refer Slide Time: 40:35)



Now, next thing you are going to do is that we have to create a repository and start working with it. So, click on your repositories, go ahead and you can create a new repository. So, there is a button there, new, and give a name that your first part of the link is same. So, just call it ID2090 demo or something, because you are going to do something with this course as an assignment also. So, just keep it as a demo and leave it public. Leave it public. And let us say, add a readme file and create repository, yeah.

(Refer Slide Time: 41:21)



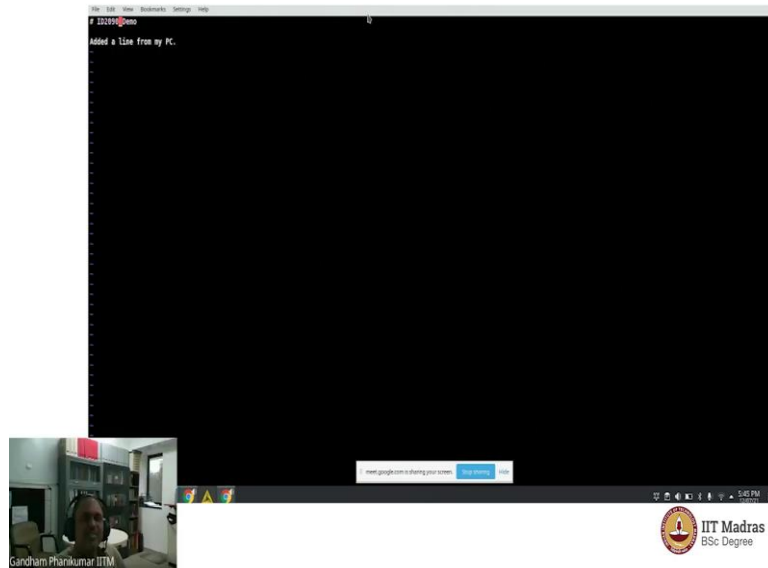
So, the now, repo is created. Now, do not have to do anything. Now, what you should do is, find a way by which you can replicate this onto your computer with the GIT command and start to edit. So, what you do is you go to the command line and choose a directory in which you would like to do these actions.

(Refer Slide Time: 42:00)

```

C:\Users\Bharat> git clone https://github.com/namba-helinski/ID2090_Demo.git
Cloning into 'ID2090_Demo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3) done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3) done.
namba-helinski@BharatRaghuani-Inspiron-3541: ~ % cd /ID2090_Demo/
namba-helinski@BharatRaghuani-Inspiron-3541: ~ % ls
README.md
namba-helinski@BharatRaghuani-Inspiron-3541: ~ % cd README.md
namba-helinski@BharatRaghuani-Inspiron-3541: ~ % git init
Initialized empty Git repository in /home/namba-helinski/ID2090_Demo/.git/
namba-helinski@BharatRaghuani-Inspiron-3541: ~ % cd git
namba-helinski@BharatRaghuani-Inspiron-3541: ~ % git ls -la
total 32
drwxr-xr-x 8 namba-helinski namba-helinski 4096 Jul 12 17:45
drwxr-xr-x 3 namba-helinski namba-helinski 4096 Jul 12 17:45
-rw-r--r-- 1 namba-helinski namba-helinski 4096 Jul 12 17:44 README
-rw-r--r-- 1 namba-helinski namba-helinski 271 Jul 12 17:43 config
-rw-r--r-- 1 namba-helinski namba-helinski 73 Jul 12 17:44 description
-rw-r--r-- 1 namba-helinski namba-helinski 21 Jul 12 17:44 HEAD
drwxr-xr-x 1 namba-helinski namba-helinski 4096 Jul 12 17:44 refs
-rw-r--r-- 1 namba-helinski namba-helinski 137 Jul 12 17:44 index
drwxr-xr-x 2 namba-helinski namba-helinski 4096 Jul 12 17:44 info
drwxr-xr-x 3 namba-helinski namba-helinski 4096 Jul 12 17:44 logs
drwxr-xr-x 4 namba-helinski namba-helinski 4096 Jul 12 17:44 objects
-rw-r--r-- 1 namba-helinski namba-helinski 112 Jul 12 17:44 packed-refs
drwxr-xr-x 5 namba-helinski namba-helinski 4096 Jul 12 17:44 refs
namba-helinski@BharatRaghuani-Inspiron-3541: ~ % cd ..
~
~*
[!] Stopped
cd..
namba-helinski@BharatRaghuani-Inspiron-3541: ~ % cd /ID2090_Demo/.git of ..
namba-helinski@BharatRaghuani-Inspiron-3541: ~ % cd /ID2090_Demo/.git/
[!] Terminated
cd.. |wd: ~/ID2090_Demo/.git/
(wd now: ~/ID2090_Demo)
namba-helinski@BharatRaghuani-Inspiron-3541: ~ % cd git
namba-helinski@BharatRaghuani-Inspiron-3541: ~ % git remote add master https://github.com/namba-helinski/ID2090_Demo
namba-helinski@BharatRaghuani-Inspiron-3541: ~ % git config --global user.name "namba-helinski"
namba-helinski@BharatRaghuani-Inspiron-3541: ~ % git config --global user.email "raghuaniidhar@gmail.com"
namba-helinski@BharatRaghuani-Inspiron-3541: ~ %

```



So, for that what you do? Wait, wait. What you do is first you make a clone of that particular repo. You are already making it as a public repo. So, it does not actually ask a password. So, I can, you can type GIT clone and then give the URL, space, give a space and give the URL of that particular GIT ID. You can go to the browser and actually copy it from the browser. Yeah, come here and then paste it and then in the end, put a dot GIT because that would be the normally, dot GIT. Demo.git. Yeah.

Raghavani (Student): Demo dot

Professor Gandham Phanikumar: Dot git. That is the name of the repo, basically. Now, enter and then see what happens. So, it is already creating a directory, that is why I stopped you from creating directory. It is already created a directory. Now, go to the directory.

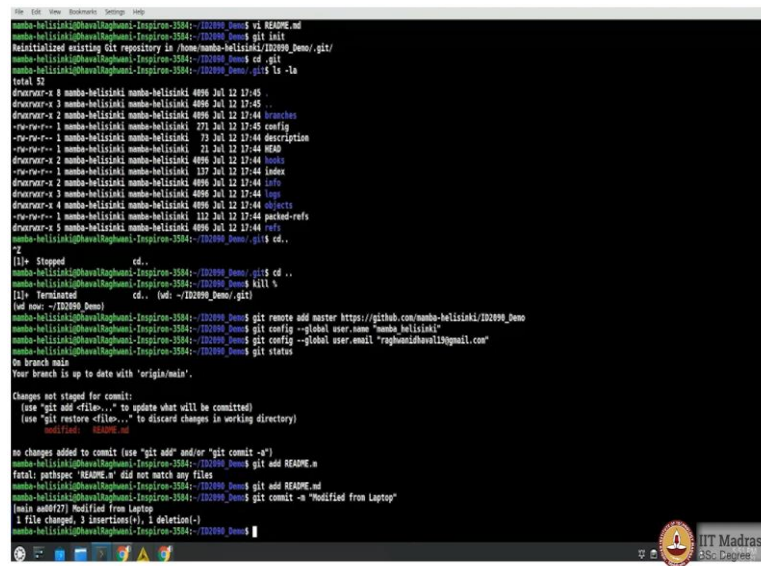
Raghavani (Student): Oh.

Professor Gandham Phanikumar: Go to there that is cd. The ID20, whatever. Yeah. Enter. ls. Put ls. And now, you see that there is a file that is there. Now, you have basically replicated the repository that was created on the github.com. Now, you would like to edit it and push these changes to the remote and therefore you are able to basically start working.

So, go ahead and actually, go ahead and edit it. Just a vi readme.md. And just below that just press O. Added a line, you can say added a line from my PC. Just to, yeah. Now, save it and come out. Now, what you need to do is, you have to, basically, configure this directory, so that the GIT understands that it is not any other directory, it is actually going to talk to the GIT server.

So, what you should do is, type a command GIT init. GIT init. No, no. GIT space init, i n i t, init, initiation. Yeah. Enter. Now, it has initialized. What it implies by saying initialized is it actually has created a directory called dot GIT. Go and see, what is in the dot GIT. Go to dot GIT. cd dot GIT. And put ls minus la.

(Refer Slide Time: 44:19)

A terminal window showing the execution of 'git init' and 'ls -la' commands. The output of 'ls -la' lists the contents of the '.git' directory, including files like 'COMMIT_EDITMSG', 'config', 'description', 'HEAD', 'hooks', 'index', 'logs', 'objects', 'packed-refs', and 'refs'. The terminal also shows the output of 'git status' and 'git config' commands.

```
nmba-helisi@havalaghami-Inspiron-3541: ~/Documents
nmba-helisi@havalaghami-Inspiron-3541:~/Documents$ git init
Initialized empty Git repository in /home/nmba-helisi/Documents/.git/
nmba-helisi@havalaghami-Inspiron-3541:~/Documents$ cd .git
nmba-helisi@havalaghami-Inspiron-3541:~/Documents/.git$ ls -la
total 52
drwxr-xr-x 8 nmba-helisi nmba-helisi 4096 Jul 12 17:45 .
drwxr-xr-x 3 nmba-helisi nmba-helisi 4096 Jul 12 17:45 ..
drwxr-xr-x 2 nmba-helisi nmba-helisi 4096 Jul 12 17:44 branches
-rw-rw-r-- 1 nmba-helisi nmba-helisi 271 Jul 12 17:45 config
-rw-rw-r-- 1 nmba-helisi nmba-helisi 73 Jul 12 17:44 description
-rw-rw-r-- 1 nmba-helisi nmba-helisi 21 Jul 12 17:44 HEAD
drwxr-xr-x 2 nmba-helisi nmba-helisi 4096 Jul 12 17:44 hooks
-rw-rw-r-- 1 nmba-helisi nmba-helisi 137 Jul 12 17:44 index
drwxr-xr-x 2 nmba-helisi nmba-helisi 4096 Jul 12 17:44 logs
drwxr-xr-x 3 nmba-helisi nmba-helisi 4096 Jul 12 17:44 logs
drwxr-xr-x 4 nmba-helisi nmba-helisi 4096 Jul 12 17:44 objects
-rw-rw-r-- 1 nmba-helisi nmba-helisi 112 Jul 12 17:44 packed-refs
drwxr-xr-x 5 nmba-helisi nmba-helisi 4096 Jul 12 17:44 refs
nmba-helisi@havalaghami-Inspiron-3541:~/Documents/.git$ cd ..
~
nmba-helisi@havalaghami-Inspiron-3541:~/Documents$ cd ..
nmba-helisi@havalaghami-Inspiron-3541:~/Documents$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
nmba-helisi@havalaghami-Inspiron-3541:~/Documents$ git add README.md
fatal: pathspec 'README.md' did not match any files
nmba-helisi@havalaghami-Inspiron-3541:~/Documents$ git config --global user.name "nmba-helisi"
nmba-helisi@havalaghami-Inspiron-3541:~/Documents$ git config --global user.email "nmba-helisi@gmail.com"
nmba-helisi@havalaghami-Inspiron-3541:~/Documents$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
nmba-helisi@havalaghami-Inspiron-3541:~/Documents$ git add README.md
fatal: pathspec 'README.md' did not match any files
nmba-helisi@havalaghami-Inspiron-3541:~/Documents$ git commit -m "Modified from Laptop"
[main 6a0f277] Modified from Laptop
1 file changed, 3 insertions(+), 1 deletion(-)
nmba-helisi@havalaghami-Inspiron-3541:~/Documents$
```

And you see that it has created a whole bunch of things. It has created a whole bunch of things, which are called as hooks, logs, objects, references, index and what (())(44:28). And, so, these are all branches. These are all, basically, the paraphernalia that GIT will be requiring to understand which version you are working etcetera.

So, do not watch anything. Just come back. cd dot dot cd space dot dot. So, we do not always say. Do not. Kill percentage. Kill space percent, enter, yeah. So, I just killed the, see, when you put control z, it actually makes the program what was running, become a background job and that is not a good idea. So, you, I just made you kill the background job. So, when you say, kill percentage sign, it means kill the background job, that is what is the meaning of that command anyway.

Now, this one requires you to, basically, start working with your GIT account. So, you have to, you have to tell, what was your remote machine and start doing some things on it. So, what you do here is, tell that you have got that there is a remote server for it. So, GIT space remote space, space, remote space, add and then space and then you have to give a name for this.

So, let us say, master. Let us say, master. And space and master is the name for the remote machine, remote location of this particular directory and you have to give a URL. So, you already have the URL. So, you can type that URL and enter. Now, it understands that there is a remote location, that you have configured. Now, how are you going to talk to it? You should tell that. So, for that you should write GIT config. GIT space config minus minus global minus minus 2 minus s. minus minus global, minus minus global.

Raghavani (Student): minus minus

Professor Gandham Phanikumar: Global, global, space, user dot name, space, and in quotes, you give your user name. Your username on the GitHub. Enter. Now, you do not need to keep password because you are going to give it on the on the flight, but you should also configure the email because the authentication can take place using either username or email.

So, it is a good idea to tell that. So, again type GIT config minus minus global user dot email. User dot email and give the user dot email. Not name. No, no. Just go back, go back, a little bit, yeah. Now, email and then in space and then in the quotes you put the email which you were actually configuring your GitHub account, whatever email that you had while registering. So, that was the email that you were supposed to give there. Gmail dot com, whatever that email, which you have configured, basically, enter.

And what happens is that from now on, whenever you do any action with respect to the remote server, it will actually check as, if it is supposed to be done by mamba Helsinki and then it will check, whether something has to be done or not. So, now let us say GIT status. So, it will check, what is happening.

Now, you see that it has recognized that there is something that has changed. And you have changed your file. So, you need to, basically, ensure that these changes are calculated to the server also. So, what we do is, we have to, basically, before we push the changes to the server, what we do is, that we have to note down what is it that I am going to do, like a message for the commit. So, you try to GIT add and then readme dot md. Just press r and then the tab, it will work. Enter, read dot md md.

Now, it has staged. It is becoming ready for you to push that file to the remote server, basically. Now, you have to also tell what have you done. So, you have to tell by giving a

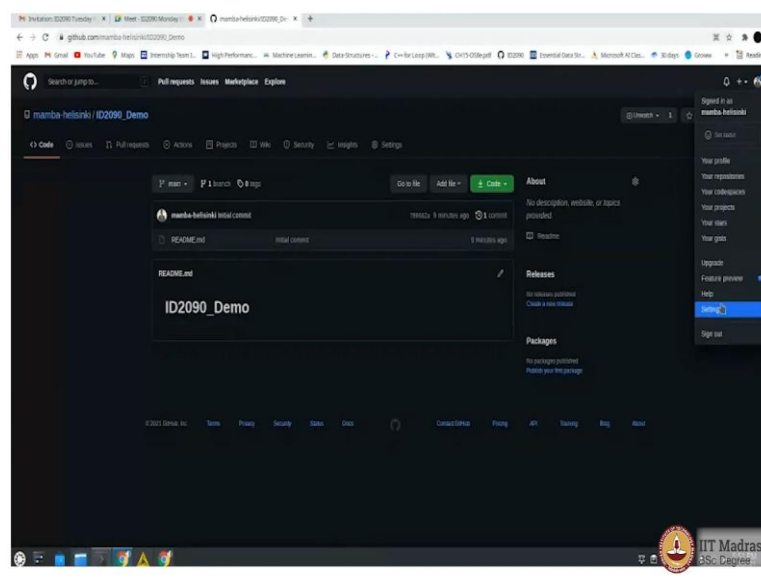
comment. Can you just hide the Google Meet thing that is on the screen? Yeah, good Now, GIT commit, GIT commit, minus m.

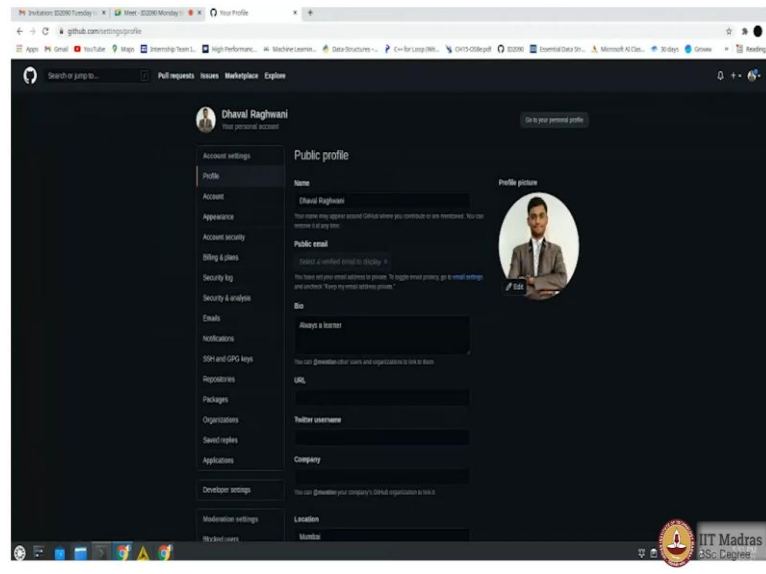
Now, in space, give a space and in the quotations, you write a message, which is helpful, you can say, you can say, modified from laptop, something like that, some message to say, what have you done. So, that is a like a message for the changes you have done. You can say sometimes fix some error or something like that. So, enter.

Now, what happens is that it is now, noted down, what files have to be pushed and it has noted on, what changes have been done. So, it is ready to perform. Now, comes the important part. Now, when you try to push, because the server is actually configured with you, it has to authenticate against you. So, it needs a personal access token. So, go ahead and create it and note it down and keep it ready.

Raghavani (Student): Again.

(Refer Slide Time: 50:15)



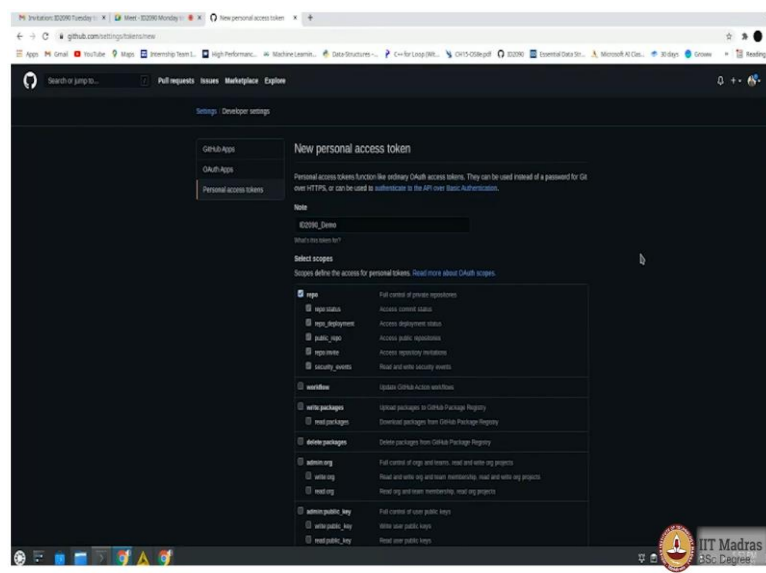


Professor Gandham Phanikumar: Yeah, yeah. Do not delete it. You keep it. Because, your GitHub will work as long as you have it.

Raghavani (Student): Should i crate it now?

Professor Gandham Phanikumar: Yeah, yeah. Create it and leave it like that. Because, once you delete it. It means that you do not have access to it again. Create one.

(Refer Slide Time: 50:35)



The screenshot shows a Windows 11 desktop environment. A web browser window is open, displaying a black page. The address bar of the browser shows the URL `http://10.0.0.1:8080/...`. The Windows taskbar is visible at the bottom, featuring the Start button, several application icons, and the system tray. The system tray includes a clock showing 11:11 AM on 11/11/2023, and the IIT Madras logo and text.

```

$ cd ..
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -m")

mamba-helisi@rhava@phampi-Inspiron-1564:~/D2090_Demo$ git add README.md
fatal: pathspec 'README.md' did not match any files

mamba-helisi@rhava@phampi-Inspiron-1564:~/D2090_Demo$ git add README.md
main aa0f27 Modified from Laptop

1 file changed, 3 insertions(+), 1 deletion(-)

mamba-helisi@rhava@phampi-Inspiron-1564:~/D2090_Demo$ vi ../pat
mamba-helisi@rhava@phampi-Inspiron-1564:~/D2090_Demo$ cat ../pat
Course_Demo
phy_MUjig@Mahl18VlanayCUDPRFVGkicJky

mamba-helisi@rhava@phampi-Inspiron-1564:~/D2090_Demo$ git push
Username for 'https://github.com': mamba-helisi
Password for 'https://github.com': mamba-helisi
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 301 bytes | 301.0 KiB/s, done.
total 3 (delta 0), reused 0 (delta 0), pushed 3
To https://github.com:mamba-helisi/D2090_Demo.git
78682ca..aa0f27 main -> main

mamba-helisi@rhava@phampi-Inspiron-1564:~/D2090_Demo$ 

```

Raghavani (Student): the same name or different

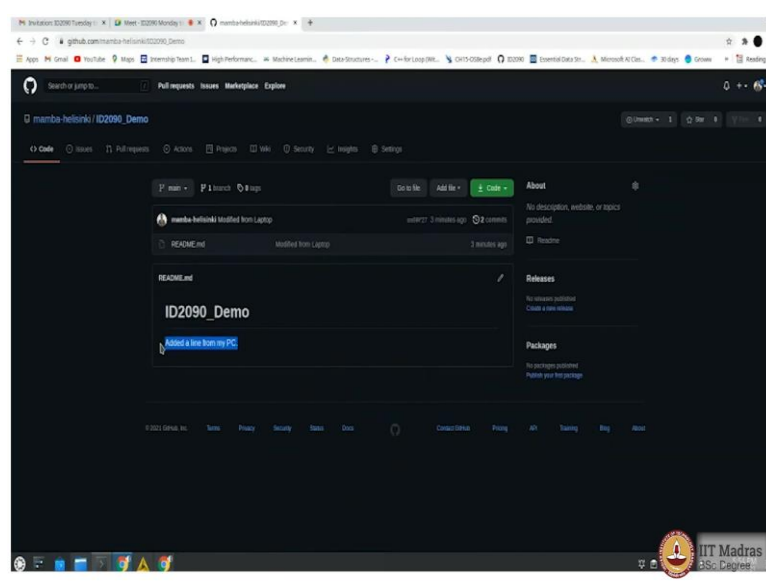
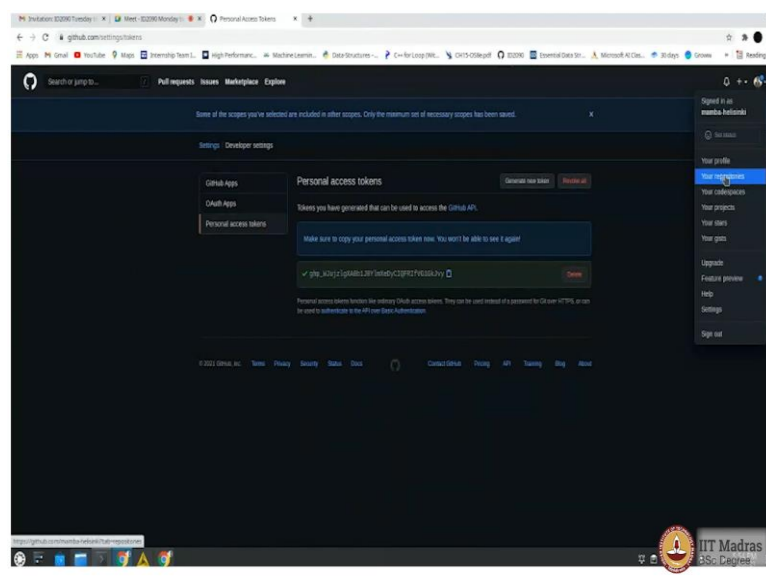
Professor Gandham Phanikumar: Yeah, you can use the same name. Not a problem. Yeah. Come down. Create token. And copy it and come here. And to be safe, just type vi dot dot by pat, v i dot v i dot dot by pat, by pat. No, no, dot dot slash p a t. What I am doing is, I am not asking you to store the password in the GitHub itself that is very crazy then everybody in the world will know your password.

So, I am storing one level up. So, it is not available to others. So, enter and here you just paste it. I and paste it. And paste it, paste it here and then delete the other one. So, you do not actually get confused. Yeah, save it and come out. Yeah, you have to escape. Escape. And save.

Now, I will tell you something for the beginners, a bit easy to do. So, when you are actually going to do the push command, it will ask for password and you need it readily. So, you can say cat dot dot by pat. So, enter it and it is there on the screen. So, you are ready to copy paste it.

So, now, you say GIT push and it knows where to push. So, you just type enter and see what happens. You give the user name and the password. What you do is you give this particular string, which is ghp whatever. Double click it on it and then paste it. Enter, yeah. So, it has actually accepted your authentication. It has pushed the content.

(Refer Slide Time: 52:42)



Now, what you do is, you go to the browser and go to your repository and you see that line has come up. So, what you did on your machine has now appeared on the server. Now, I will illustrate in the next session, on how I can make a change and you can approve it. And we will also do it with each other. So, that multiple people can actually go and make changes. So, that all of us can collaborately author some document. I will do that. So, next tomorrow's session, I will show you how to do that. So, for now, this demo I hope was useful. So, people can actually see what we are doing.