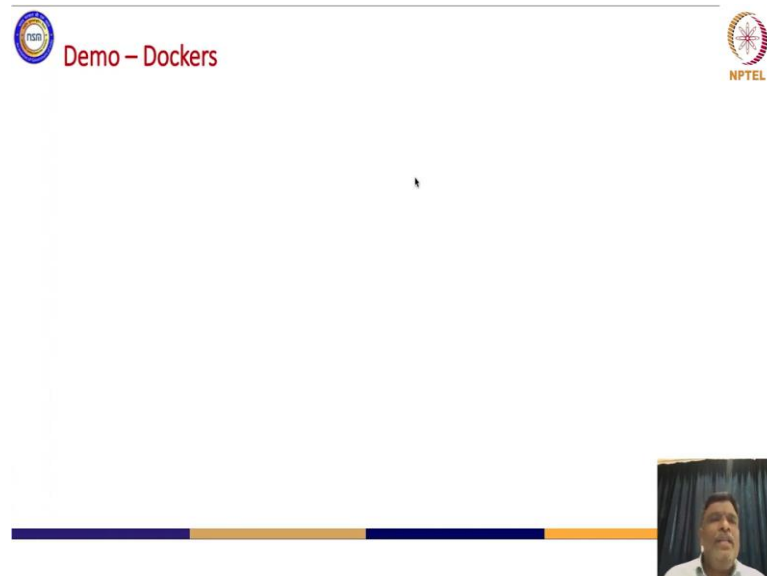


Applied Accelerated Artificial Intelligence
Prof. Satyadhyam Chickerur
School of Computer Science and Engineering
Indian Institute of Technology, Madras

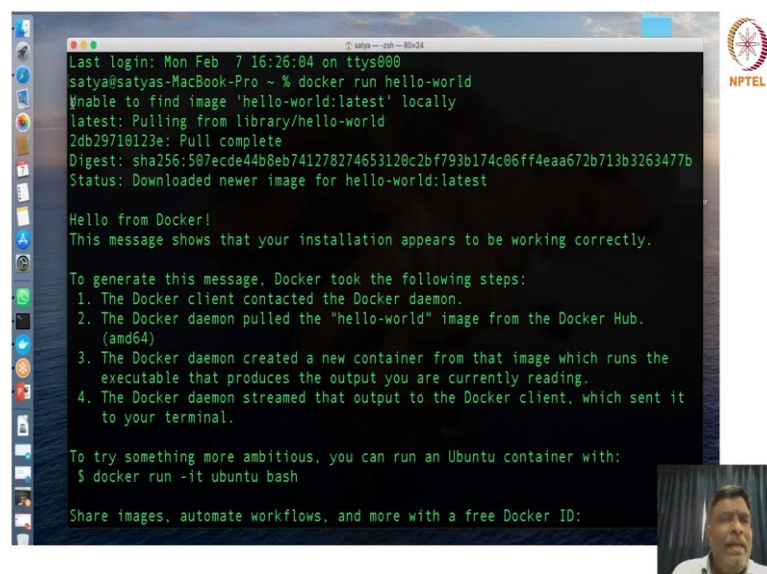
Lecture - 07
Introduction to Containers and IDE Dockers Part - 2

(Refer Slide Time: 00:33)



Let us try to do demo on Dockers. So, hands on Dockers if you had installed Dockers already. So, please try to open the command prompt ok.

(Refer Slide Time: 00:44)



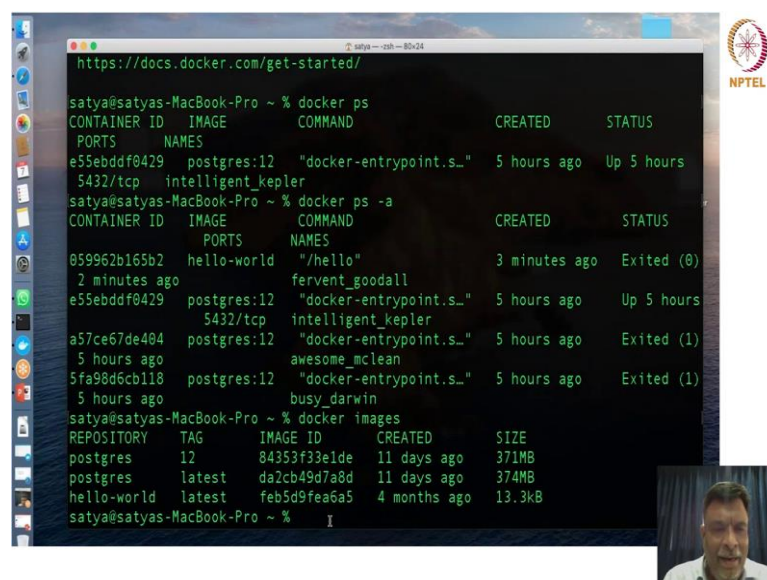
And, then let me start with the very basics of Dockers. So, I am going to now show you once the Docker engine is running in the background, you can start with working with Dockers. So, the first thing is a very very simple thing of how do you actually start the first hello-world container ok.

So, you can actually start with something like `docker run hello-world` right. So, try understanding this that when a Docker is run ok, if that particular Docker is not available in your cache, it will show that it is unable to find the image locally ok. Once it shows that it is unable to find the image locally, what effectively happens is it is going to pull ok, pull it from the library which is the latest library from the Docker hub.

And, once the pull is complete ok; once the pull is complete you will get that thing downloaded, that Docker will get downloaded ok. So, now, here you see that you have run a container, it is a Docker container of hello-world ok and it shows a message that Hello from Docker right.

So, what does it mean? It is pulling the container layer by layer ok, we will see in some of the other container as to how that layer by layer pulling happens. But, for the time being just understand that you pull it layer by layer ok.

(Refer Slide Time: 03:02)



```

satya@satyas-MacBook-Pro ~ % docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS
PORTS         NAMES
e55ebddf0429   postgres  "docker-entrypoint.s..." 5 hours ago Up 5 hours
5432/tcp      intelligent_kepler
satya@satyas-MacBook-Pro ~ % docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS
PORTS         NAMES
059962b165b2   hello-world  "/hello"                 3 minutes ago Exited (0)
2 minutes ago fervent_goodall
e55ebddf0429   postgres  "docker-entrypoint.s..." 5 hours ago Up 5 hours
5432/tcp      intelligent_kepler
a57ce67de404   postgres  "docker-entrypoint.s..." 5 hours ago Exited (1)
5 hours ago   awesome_mclean
5fa98d6cb118   postgres  "docker-entrypoint.s..." 5 hours ago Exited (1)
5 hours ago   busy_darwin
satya@satyas-MacBook-Pro ~ % docker images
REPOSITORY    TAG       IMAGE ID       CREATED      SIZE
postgres      latest   84353f33e1de   11 days ago  371MB
postgres      latest   da2cb49d7a8d   11 days ago  374MB
hello-world    latest   feb5d9fea6a5   4 months ago 13.3kB
satya@satyas-MacBook-Pro ~ %

```

And, then from that primary image what you have ok, it displays the output. The container stops its execution ok, it ceases to run after the output is displayed, that is hello

from the Docker. Now, what containers are running on system? How do I come to know? There is a command `docker ps` ok.

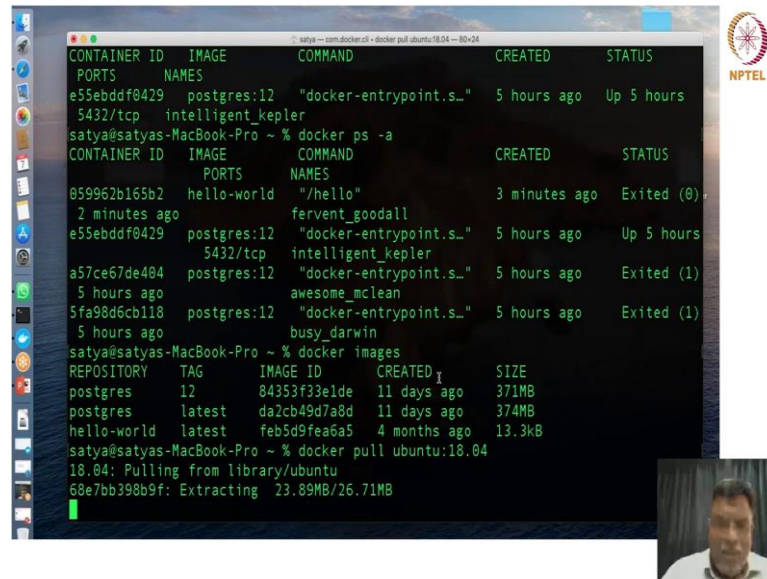
So, for the time being when you run it on your system, you would not see anything because none of your Docker containers are running. But, for my system some postgres thing is running because, I would be showing that to you as to how do you interface a Docker to a database right; for that reason I have kept it running ok, only with that intention. So, in your case you will not get anything ok, you will not get anything.

But, there is another command which will tell us as to how many images right are there in my cache locally ok. Before that there is another command which you can do and see as to how many containers right have run and stopped or are running ok with the command `docker ps -a`. So, you will see that in this hello-world /hello-world container ran 3 minutes ago ok and it stopped.

So, this is what you will get by using the command `docker ps -a`, it tells all the containers which are running which have run which exist in your cache right. So, then what are the images ok which are cached locally, how are you going to know? You can type `docker images` which will tell as to what is a docker image which is there in my cache right. So, hello-world is the docker image, there is another docker image which is postgres.

There is another docker image which is postgres 12 right. So, this gives us the information about three things: one is what is the TAG associated with it, what is the IMAGE ID, IMAGE ID is created actually and then when was this Docker CREATED and what is the SIZE of the Docker. This is the information which you get from Docker images right. Now, once you are able to do this, let us try to understand when we get all these Dockers right.

(Refer Slide Time: 06:23)

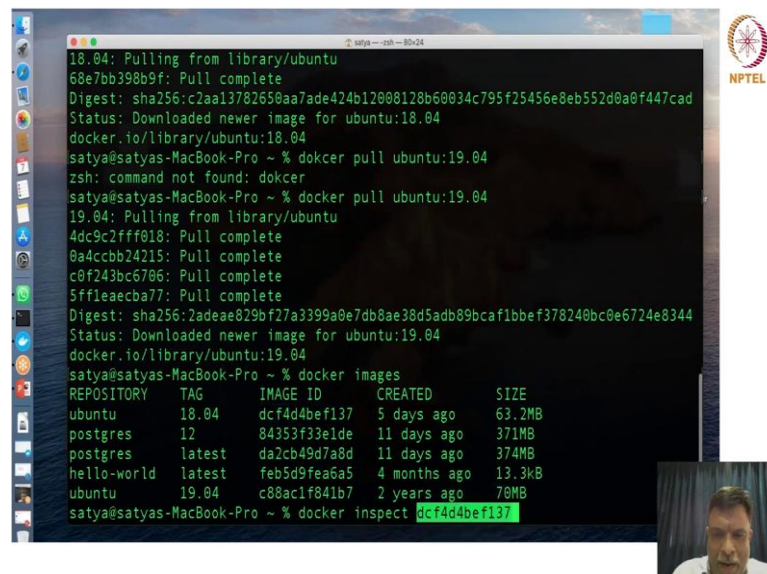


```
satya ~ -- ssh -- 80x24
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
PORTS         NAMES
e55ebddf0429   postgres:12    "docker-entrypoint.s..." 5 hours ago    Up 5 hours
5432/tcp      intelligent_kepler
satya@satyas-MacBook-Pro ~ % docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
PORTS         NAMES
059962b165b2   hello-world    "/hello"                 3 minutes ago  Exited (0)
2 minutes ago   fervent_goodall
e55ebddf0429   postgres:12    "docker-entrypoint.s..." 5 hours ago    Up 5 hours
5432/tcp      intelligent_kepler
a57ce67de404   postgres:12    "docker-entrypoint.s..." 5 hours ago    Exited (1)
5 hours ago    awesome_mclean
5fa98d6cb118   postgres:12    "docker-entrypoint.s..." 5 hours ago    Exited (1)
5 hours ago    busy_darwin
satya@satyas-MacBook-Pro ~ % docker images
REPOSITORY    TAG          IMAGE ID      CREATED      SIZE
postgres      12           84353f33e1de 11 days ago  371MB
postgres      latest       da2cb49d7a8d 11 days ago  374MB
hello-world    latest       feb5d9fea6a5 4 months ago 13.3kB
satya@satyas-MacBook-Pro ~ % docker pull ubuntu:18.04
18.04: Pulling from library/ubuntu
68e7bb398b9f: Extracting 23.89MB/26.71MB
```

You have just downloaded them, now we will try to run a Docker ok. So, the Docker is actually run from a Docker hub ok and then let us try to understand and see how do we start them, stop them ok and then try to understand ok some of the things. So, let us try to pull in a docker pull ok.

Let us say ubuntu 18.04, let us try to pull ok. So, we are pulling Docker which is ubuntu 18.04 right yeah. So, if you see this, it is downloading ok the ubuntu 18.04, it has pulled everything completely ok.

(Refer Slide Time: 07:40)



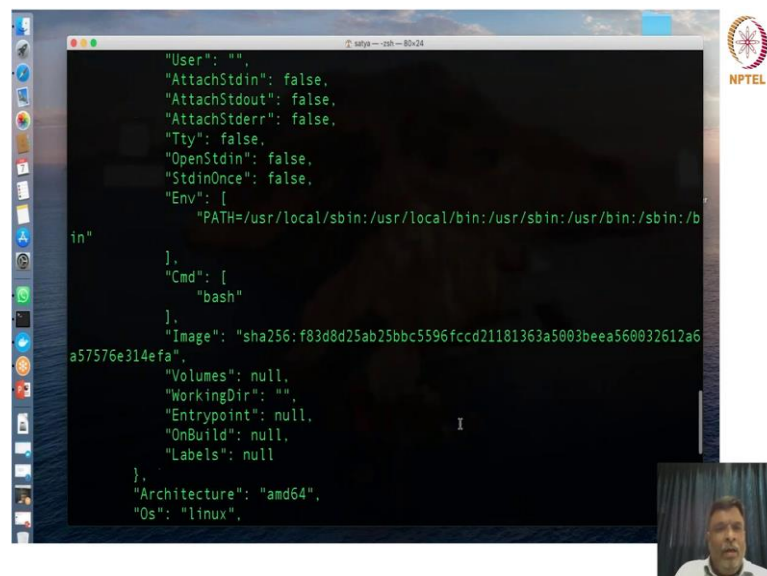
```
satya ~ -- ssh -- 80x24
18.04: Pulling from library/ubuntu
68e7bb398b9f: Pull complete
Digest: sha256:c2aa13782650aa7ade424b12008128b60034c795f25456e8eb552d0a0f447cad
Status: Downloaded newer image for ubuntu:18.04
docker.io/library/ubuntu:18.04
satya@satyas-MacBook-Pro ~ % docker pull ubuntu:19.04
zsh: command not found: dokcer
satya@satyas-MacBook-Pro ~ % docker pull ubuntu:19.04
19.04: Pulling from library/ubuntu
4dc9c2fff018: Pull complete
0a4ccbb24215: Pull complete
c0f243bc6706: Pull complete
5ff1eae6ba77: Pull complete
Digest: sha256:2adeae829bf27a3399a0e7db8ae38d5adb09bcafbbe378240bc0e6724e8344
Status: Downloaded newer image for ubuntu:19.04
docker.io/library/ubuntu:19.04
satya@satyas-MacBook-Pro ~ % docker images
REPOSITORY    TAG          IMAGE ID      CREATED      SIZE
ubuntu        18.04        dcf4d4bef137 5 days ago   63.2MB
postgres      12           84353f33e1de 11 days ago  371MB
postgres      latest       da2cb49d7a8d 11 days ago  374MB
hello-world    latest       feb5d9fea6a5 4 months ago 13.3kB
ubuntu        19.04        c88ac1f841b7 2 years ago  70MB
satya@satyas-MacBook-Pro ~ % docker inspect dcf4d4bef137
```

And, then let us say you want to work with ubuntu 19.04 or 20.04 or whatever. So, let us pull ubuntu 19.04 ok, in docker pull ubuntu:19.04 right. So, in your case if that would not be there in your library, it will show you ok something like this. These are layers. If you see here, this is one layer getting pulled, this is another getting layer getting pulled right.

So, you get now two images, two Dockers for ubuntu itself. So, let us try to understand ok and see whether those images are there or not. If you see here, now see you have ubuntu 18.04 which was created 5 days ago which has got 63.2 MBs as its size and then you have a ubuntu 19.04 which was created 2 years ago and it is 70 MB.

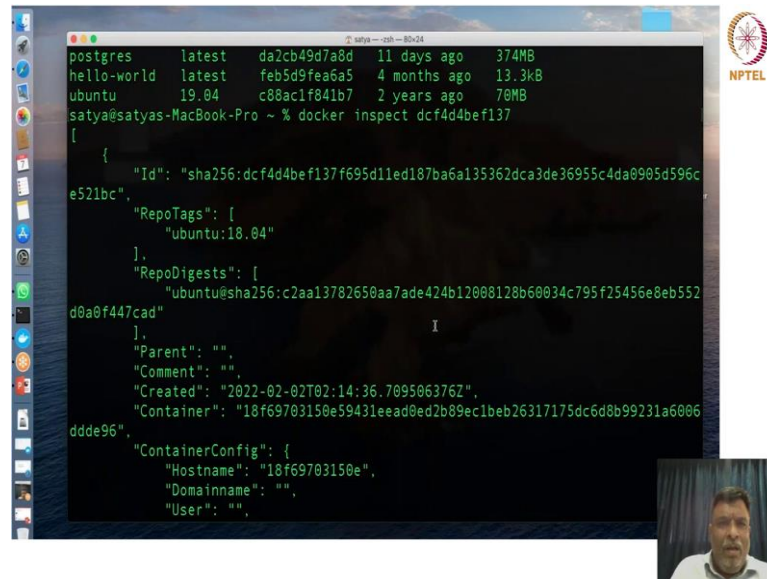
Now, see one of the things is from the security aspect, if you want to see which is the latest one whether somebody has tampered with that Docker ok or you feel that it is not correct from the size aspect right; those are all which could be related to this right. Now, for that reason if you want to see something like you suspect, there is something which is called as docker inspect. And, for docker inspect what are you going to do? You are going to actually take ok, the IMAGE ID ok of whichever container you would like to inspect right.

(Refer Slide Time: 10:02)



```
"User": "",
"AttachStdin": false,
"AttachStdout": false,
"AttachStderr": false,
"Tty": false,
"OpenStdin": false,
"StdinOnce": false,
"Env": [
  "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/b
in"
],
"Cmd": [
  "bash"
],
"Image": "sha256:f83d8d25ab25bbc5596fccd21181363a5003beea560032612a6
a57576e314efa",
"Volumes": null,
"WorkingDir": "",
"Entrypoint": null,
"OnBuild": null,
"Labels": null
},
"Architecture": "amd64",
"Os": "linux",
```

(Refer Slide Time: 10:09)

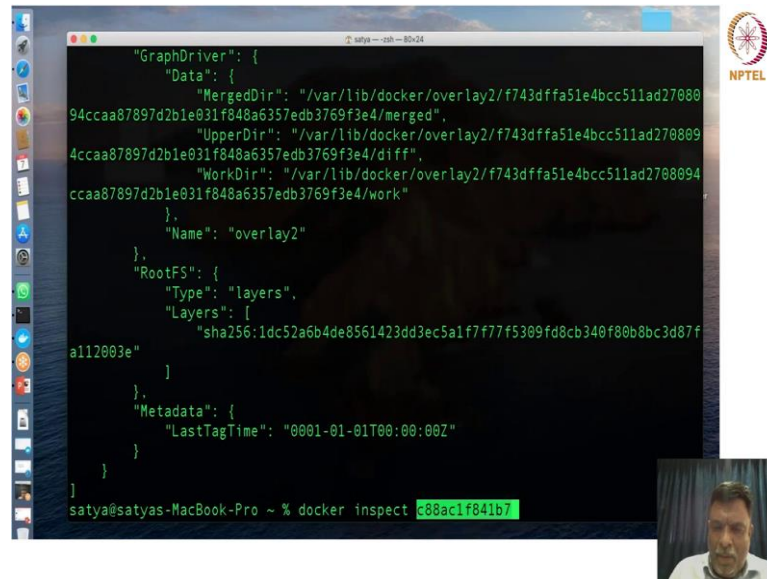


```
satya ~ % docker inspect dcf4d4bef137
[
  {
    "Id": "sha256:dcf4d4bef137f695d11ed187ba6a135362dca3de36955c4da0905d596c
e521bc",
    "RepoTags": [
      "ubuntu:18.04"
    ],
    "RepoDigests": [
      "ubuntu@sha256:c2aa13782650aa7ade424b12008128b60034c795f25456e8eb552
d0a0f447cad"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2022-02-02T02:14:36.709506376Z",
    "Container": "18f69703150e59431eead0ed2b89ec1beb26317175dc6d8b99231a6006
ddde96",
    "ContainerConfig": {
      "Hostname": "18f69703150e",
      "Domainname": "",
      "User": ""
    }
  }
]
```

So, I did this, you will come to know that this particular Docker image ok has got this checksum right; from which repository it was downloaded, what is the repo tag, when was it created right, when was it created ok. And then which was the host name, domain name so on and so forth.

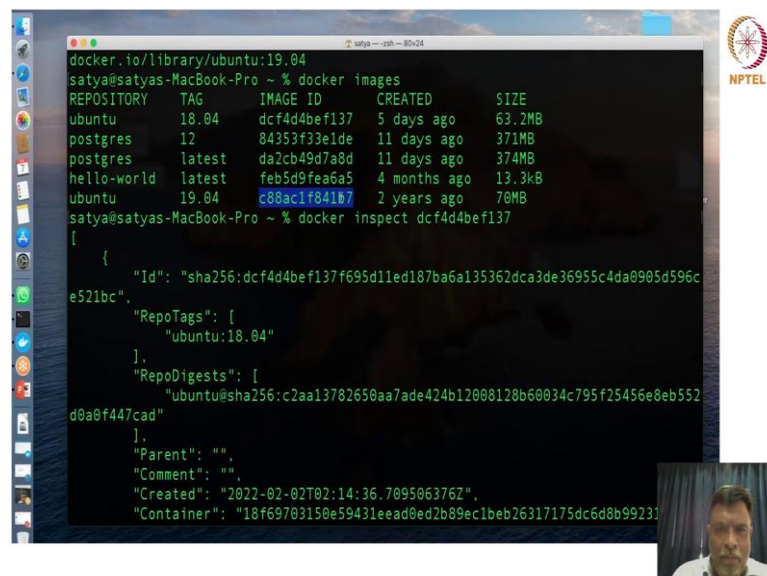
There are so many things which you can come to know from this right. We are not going into the details of each one of this right, but the idea is to make you familiarize right with what all you can do with Dockers right. So, once you inspect that is fine. Now, it will contain lot of things.

(Refer Slide Time: 10:51)



```
"GraphDriver": {
  "Data": {
    "MergedDir": "/var/lib/docker/overlay2/f743dffa51e4bcc511ad2708094ccaa87897d2b1e031f848a6357edb3769f3e4/merged",
    "UpperDir": "/var/lib/docker/overlay2/f743dffa51e4bcc511ad2708094ccaa87897d2b1e031f848a6357edb3769f3e4/diff",
    "WorkDir": "/var/lib/docker/overlay2/f743dffa51e4bcc511ad2708094ccaa87897d2b1e031f848a6357edb3769f3e4/work"
  },
  "Name": "overlay2"
},
"RootFS": {
  "Type": "layers",
  "Layers": [
    "sha256:1dc52a6b4de8561423dd3ec5a1f77f75309fd8cb340f80b8bc3d87fa112003e"
  ]
},
"Metadata": {
  "LastTagTime": "0001-01-01T00:00:00Z"
}
}
]
satya@satyas-MacBook-Pro ~ % docker inspect c88ac1f841b7
```

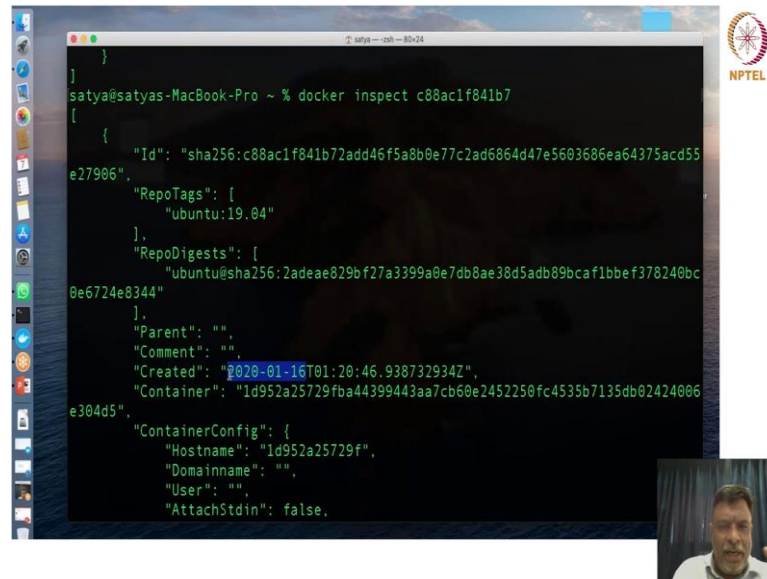
(Refer Slide Time: 11:02)



```
docker.io/library/ubuntu:19.04
satya@satyas-MacBook-Pro ~ % docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
ubuntu 18.04 dcf4d4bef137 5 days ago 63.2MB
postgres 12 84353f33e1de 11 days ago 371MB
postgres latest da2cb49d7a8d 11 days ago 374MB
hello-world latest feb5d9fea6a5 4 months ago 13.3kB
ubuntu 19.04 c88ac1f841b7 2 years ago 70MB
satya@satyas-MacBook-Pro ~ % docker inspect dcf4d4bef137
[
  {
    "Id": "sha256:dcf4d4bef137f695d11ed187ba6a135362dca3de36955c4da0905d596ce521bc",
    "RepoTags": [
      "ubuntu:18.04"
    ],
    "RepoDigests": [
      "ubuntu@sha256:c2aa13782650aa7ade424b12008128b60034c795f25456e8eb552d0a0f447cad"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2022-02-02T02:14:36.709506376Z",
    "Container": "18f69703150e59431eead0ed2b89ec1beb26317175dc6d8b99231"
  }
]
```

Similarly, let us try to actually inspect docker inspect let us say 19 04. So, here you will come to know that when was it created, was it a new creation ok, because of the date and time ok which is shown.

(Refer Slide Time: 11:21)

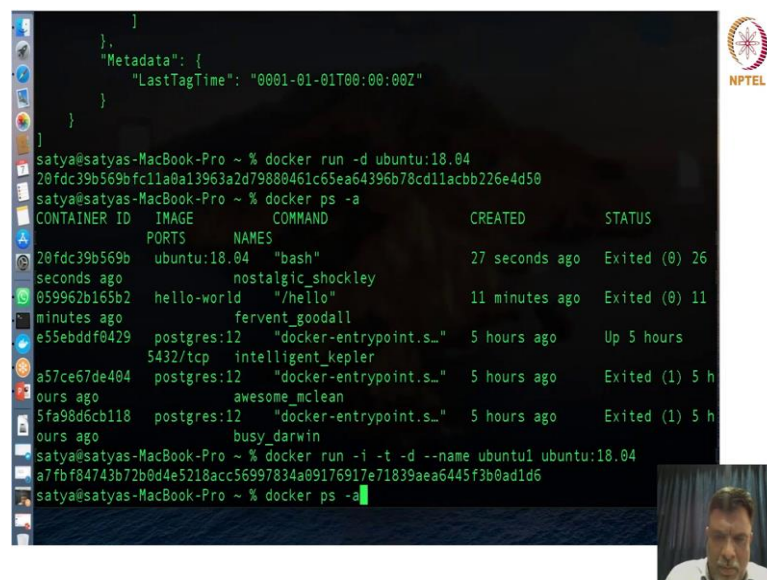


```
satya@satyas-MacBook-Pro ~ % docker inspect c88ac1f841b7
[
  {
    "Id": "sha256:c88ac1f841b72add46f5a8b0e77c2ad6864d47e5603686ea64375acd55e27906",
    "RepoTags": [
      "ubuntu:19.04"
    ],
    "RepoDigests": [
      "ubuntu@sha256:2adeae829bf27a3399a0e7db8ae38d5adb89bcacf1bbef378240bce0e6724e8344"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2020-01-16T01:20:46.938732934Z",
    "Container": "1d952a25729fba44399443aa7cb60e2452250fc4535b7135db02424006e304d5",
    "ContainerConfig": {
      "Hostname": "1d952a25729f",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,

```

So, see here it was created in the year 2020 whereas, 18.04 right I suppose was created very recently right; it is 2nd of February, 2022 right. So, this basically means that you can get lot of details about these Dockers right from this inspect command.

(Refer Slide Time: 11:48)



```
satya@satyas-MacBook-Pro ~ % docker run -d ubuntu:18.04
20fdc39b569bfc11a0a13963a2d79880461c65ea64396b78cd1lacbb226e4d50
satya@satyas-MacBook-Pro ~ % docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS
20fdc39b569b   ubuntu:18.04   "bash"                  27 seconds ago   Exited (0) 26
seconds ago
059962b165b2   hello-world   "/hello"                11 minutes ago   Exited (0) 11
minutes ago
e55ebddf0429   postgres:12   "docker-entrypoint.s..." 5 hours ago     Up 5 hours
5432/tcp      intelligent_kepler
a57ce67de404   postgres:12   "docker-entrypoint.s..." 5 hours ago     Exited (1) 5 h
ours ago
5fa98d6cb118   postgres:12   "docker-entrypoint.s..." 5 hours ago     Exited (1) 5 h
ours ago
busy_darwin
satya@satyas-MacBook-Pro ~ % docker run -i -t -d --name ubuntu1 ubuntu:18.04
a7fbf84743b72b0d4e5218acc56997834a09176917e71839aea6445f3b0ad1d6
satya@satyas-MacBook-Pro ~ % docker ps -a
```

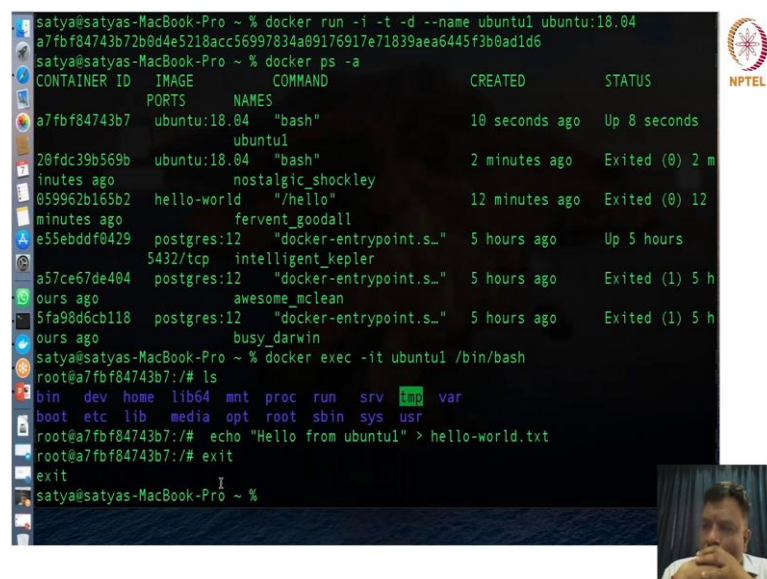
So, now let us try to understand and run the command. So, we will try to run the command, in the sense we will try to run the docker now ok using the run command that is what I meant. So, you run the docker using the run command. So, docker run -d ubuntu:18.04, it has run right.

So, but this tells the Docker to run the command in the daemon mode, in the background right because I have used -d. So, it is a daemon background command right. If you basically omit this flag like before omitting, let us try to understand and see and check the status of the docker. So, if you see here ubuntu 18.04 ok, the batch shell is running right; let me just enlarge it and show you ok.

So, yeah; so, this particular thing is actually trying to show you, returns the container ID actually; once you run it ok, it is going to return the container ID for you ok and some Dockers will not return anything to you ok. So, now, I showed you this psa command. So, now, the idea is when we are trying to run the Docker ok, we need to start using it right. It is not that it just runs in the background. How do I use it?

So, for that you have to make docker run in the interactive mode -i -t, because I am trying to work with a terminal, a pseudo terminal handle and then d, the name is let us say ubuntu1 ok and then ubuntu:18.04 right.

(Refer Slide Time: 14:30)



```
satya@satyas-MacBook-Pro ~ % docker run -i -t -d --name ubuntu1 ubuntu:18.04
a7fbf84743b72b0d4e5218acc56997834a09176917e71839aea6445f3b0ad1d6
satya@satyas-MacBook-Pro ~ % docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
a7fbf84743b7	ubuntu:18.04	"bash"	10 seconds ago	Up 8 seconds
20fdc39b569b	ubuntu:18.04	"bash"	2 minutes ago	Exited (0) 2 m
059962b165b2	hello-world	"/hello"	12 minutes ago	Exited (0) 12
e55ebddf0429	postgres:12	"docker-entrypoint.s..."	5 hours ago	Up 5 hours
a57ce67de404	postgres:12	"docker-entrypoint.s..."	5 hours ago	Exited (1) 5 h
5fa98d6cb118	postgres:12	"docker-entrypoint.s..."	5 hours ago	Exited (1) 5 h
		busy_darwin		

```
satya@satyas-MacBook-Pro ~ % docker exec -it ubuntu1 /bin/bash
root@a7fbf84743b7:/# ls
bin  dev  home  lib64  mnt  proc  run  srv  var
boot  etc  lib   media  opt  root  sbin  sys  usr
root@a7fbf84743b7:/# echo "Hello from ubuntu1" > hello-world.txt
root@a7fbf84743b7:/# exit
exit
satya@satyas-MacBook-Pro ~ %
```

Now, if you see ok; so, here we can see a new instance of ubuntu 18.04 which is termed as ubuntu1 which is created 10 seconds ago and it is running right. So, now we know that its running and now I have to let us say work on it right. So, I need to run the command inside the container ok using the docker exec command right. So, we need to run docker exec ok and then I need to do it in interactive mode, I want to run it on ubuntu1 instance.

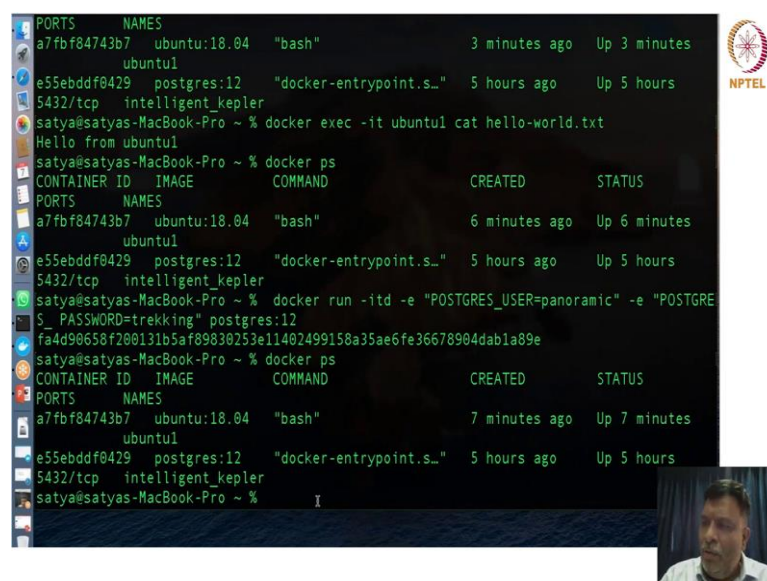
With what do I want to run? I want to run the shell right, the batch shell now. So, my ubuntu container is there, it is actually running ubuntu which is up by 8 seconds, but I want to work on the shell now. So, what I do? I run this, I get into ubuntu as rule. So, now, let me just check and show you this. So, if you see this, this is my ubuntu 18.04 ok. So, it is now running ok. So, now, we will get the root prompt here ok.

So, I suppose people would have got it. Now, how do I use something right? I have now created a Docker, it is running, I have now come to the shell; now I want to let us say do something and maybe use it right. So, let me just do some small thing like echo "Hello from ubuntu1" ok and now I save it in text file right; so, let first write. So, I save it in a text file right. Now, we can actually exit; that means, my work is done, I have saved something in the hello-world text file, that text file tells that it is hello from ubuntu1.

So, I exit this container right. So, once I exit this container, what thing I will get right? I have exited the container right. So, now, let us try to maybe create another container called ubuntu 2. So, let me just go to the next thing and let us try to see you know. So, if you see this docker ps now right, I have done some operation, I have exited out of this.

But, it is up since 3 minutes right. So, what I am going to do is I will just show you ok instead of Docker execute again to access a shell inside for our container use it to display the output of the hello-world text file ok.

(Refer Slide Time: 18:16)



```

PORTS      NAMES
a7fbf84743b7  ubuntu:18.04  "bash"          3 minutes ago  Up 3 minutes
ubuntu1
e55ebddf0429  postgres:12   "docker-entrypoint.s..." 5 hours ago    Up 5 hours
5432/tcp    intelligent_kepler
satya@satyas-MacBook-Pro ~ % docker exec -it ubuntu1 cat hello-world.txt
Hello from ubuntu1
satya@satyas-MacBook-Pro ~ % docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS
a7fbf84743b7   ubuntu:18.04  "bash"                6 minutes ago  Up 6 minutes
ubuntu1
e55ebddf0429   postgres:12   "docker-entrypoint.s..." 5 hours ago    Up 5 hours
5432/tcp      intelligent_kepler
satya@satyas-MacBook-Pro ~ % docker run -itd -e "POSTGRES_USER=panoramic" -e "POSTGRES
PASSWORD=trekking" postgres:12
fa4d90658f200131b5af89830253e11402499158a35ae6fe36678904dab1a89e
satya@satyas-MacBook-Pro ~ % docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS
a7fbf84743b7   ubuntu:18.04  "bash"                7 minutes ago  Up 7 minutes
ubuntu1
e55ebddf0429   postgres:12   "docker-entrypoint.s..." 5 hours ago    Up 5 hours
5432/tcp      intelligent_kepler
satya@satyas-MacBook-Pro ~ %
I

```

We can directly use a cat command on it and I will show you that how do we do it. We do `docker exec -it ubuntu1 cat hello-world.txt`. Do this, my text file I had saved Hello from ubuntu1. So, this saves it right and I am able to get it. So, this is how actually you are going to use your Docker containers ok and then you can start your Docker, stop your Docker anytime you like ok.

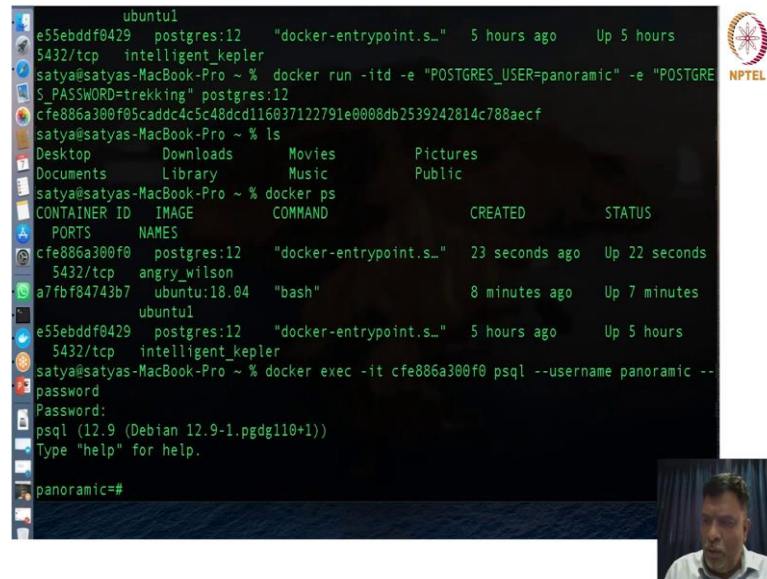
So, now, let me just show you another thing as to how do you use it, you can prune it. I was just trying to think that if I can give example ok of trying to attach some what to say command to a Docker container which is running something right. So, that is something which is called as Docker attach ok. So, Docker attach I would not execute now; I actually have done it, but I will just show you some other things wherein you can connect it with a sql.

So, for that reason I am just trying to tell you that Docker attach command attaches any primary process inside the container that is our dash file right `/bin /bash` shell ok to the container instance right. So, that could also be done. So, now let us try to do something like access to ok postgres sql container and let us try to do something wherein we can actually try to see how we can do things.

So, let me just show you `docker ps`. So, we have this postgres 12 which we have kept ok. Now, I will try to directly run that command which is there which I have done. So, I have pulled the postgres ok. Now, in that there is a user by name panoramic and then that PASSWORD is trekking.

So, I will run this ok. So, I have a postgres sql installed which I showed you and that particular thing actually has got a table right and we will try to see. So, now, this is running right, if it is running healthy; we will try to see `docker ps` it is running ok, the other one was also running. So, once we logged in using that we can actually see ok the Docker thing.

(Refer Slide Time: 21:35)



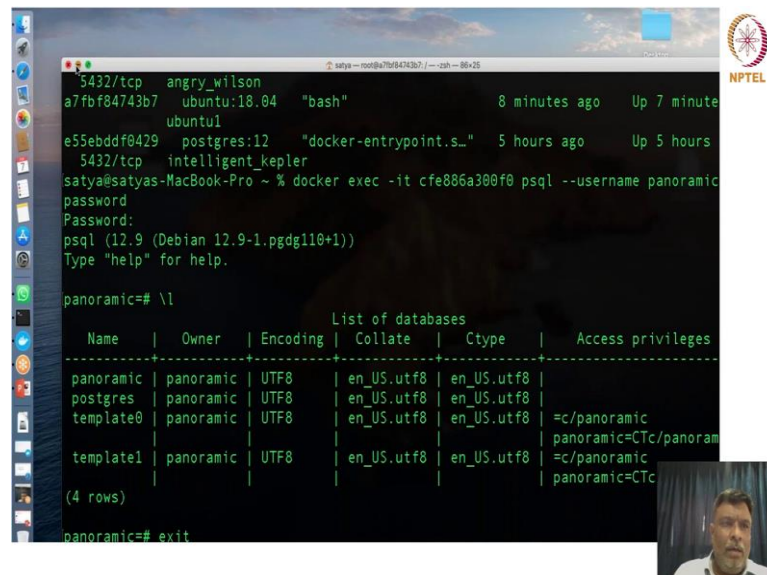
The terminal shows a user on a MacBook-Pro running Docker commands. It lists several containers, including 'postgres:12' and 'angry_wilson'. The user then runs 'docker exec' to enter a PostgreSQL container, providing a username and password. The prompt changes to 'psql (12.9 (Debian 12.9-1.pgdg110+1))' and the user is at the 'panoramic=#' prompt.

```
ubuntu1
e55ebddf0429 postgres:12 "docker-entrypoint.s..." 5 hours ago Up 5 hours
5432/tcp intelligent_kepler
satya@satyas-MacBook-Pro ~ % docker run -itd -e "POSTGRES_USER=panoramic" -e "POSTGRES_PASSWORD=trekking" postgres:12
cfe886a300f0 postgres:12 "docker-entrypoint.s..." 23 seconds ago Up 22 seconds
satya@satyas-MacBook-Pro ~ % ls
Desktop Downloads Movies Pictures
Documents Library Music Public
satya@satyas-MacBook-Pro ~ % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS
cfe886a300f0 postgres:12 "docker-entrypoint.s..." 23 seconds ago Up 22 seconds
5432/tcp angry_wilson
a7fbf84743b7 ubuntu:18.04 "bash" 8 minutes ago Up 7 minutes
ubuntu1
e55ebddf0429 postgres:12 "docker-entrypoint.s..." 5 hours ago Up 5 hours
5432/tcp intelligent_kepler
satya@satyas-MacBook-Pro ~ % docker exec -it cfe886a300f0 psql --username panoramic --password
Password:
psql (12.9 (Debian 12.9-1.pgdg110+1))
Type "help" for help.

panoramic=#
```

So, I will run the Docker, docker exec; I already have it so, maybe I will show you that yeah. So, yes so, now, if this is running.. So, I will have to change this container ID with this container ID right and then run this, I should type the Password, I have come to this ps sql prompt.

(Refer Slide Time: 23:13)



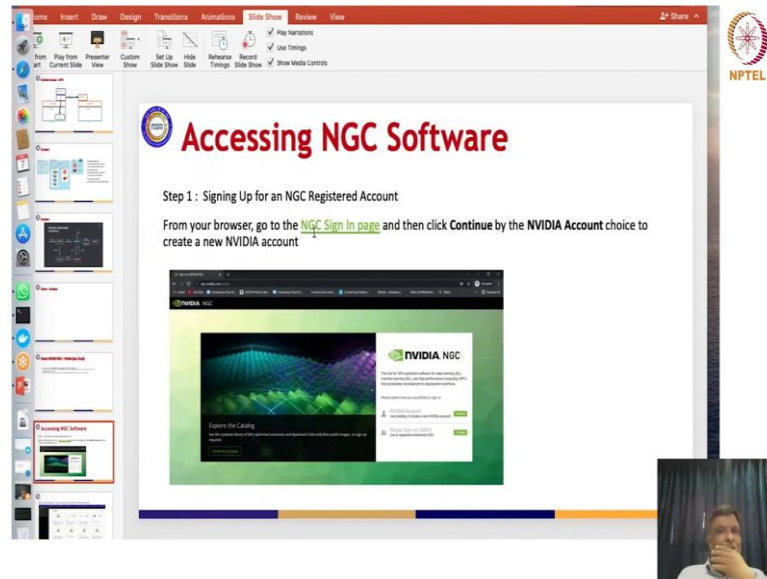
The terminal shows the user running the '\l' command in the PostgreSQL prompt, which lists the databases: 'panoramic', 'postgres', 'template0', and 'template1'. The output includes details like owner, encoding, collate, ctype, and access privileges.

```
panoramic=# \l
List of databases
Name | Owner | Encoding | Collate | Ctype | Access privileges
-----+-----+-----+-----+-----+-----
panoramic | panoramic | UTF8 | en_US.utf8 | en_US.utf8 | 
postgres | panoramic | UTF8 | en_US.utf8 | en_US.utf8 | 
template0 | panoramic | UTF8 | en_US.utf8 | en_US.utf8 | =c/panoramic
template1 | panoramic | UTF8 | en_US.utf8 | en_US.utf8 | =c/panoramic
(4 rows)

panoramic=# exit
```

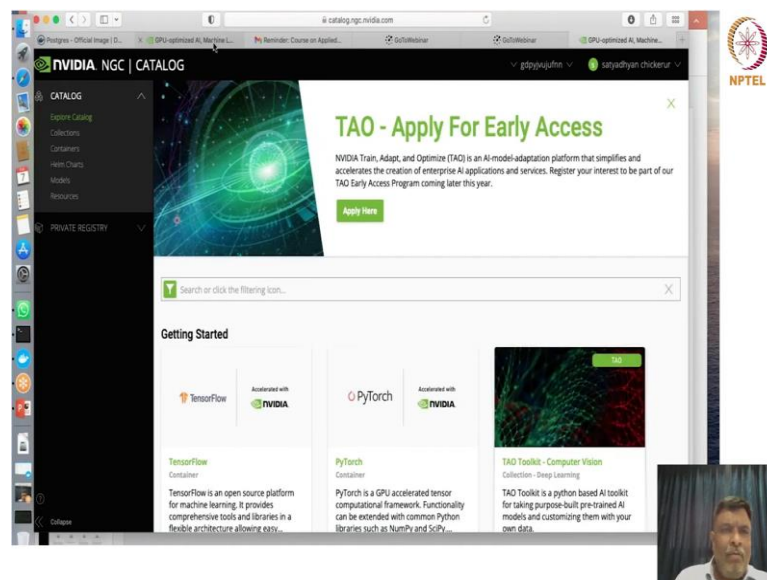
And, if I l, these are the list of databases which is there in my postgres sql right.

(Refer Slide Time: 23:37)

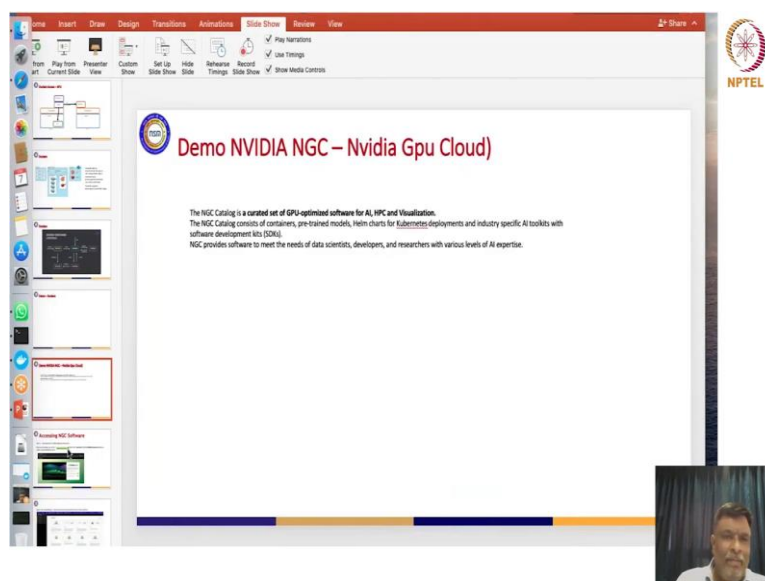


So, I just try to show you two things. Now, let me just go to the basics of NGC, as to how do you download right effective Dockers from the NGC cloud. So, all of you can actually visit this site ok, whenever you can visit.

(Refer Slide Time: 23:54)

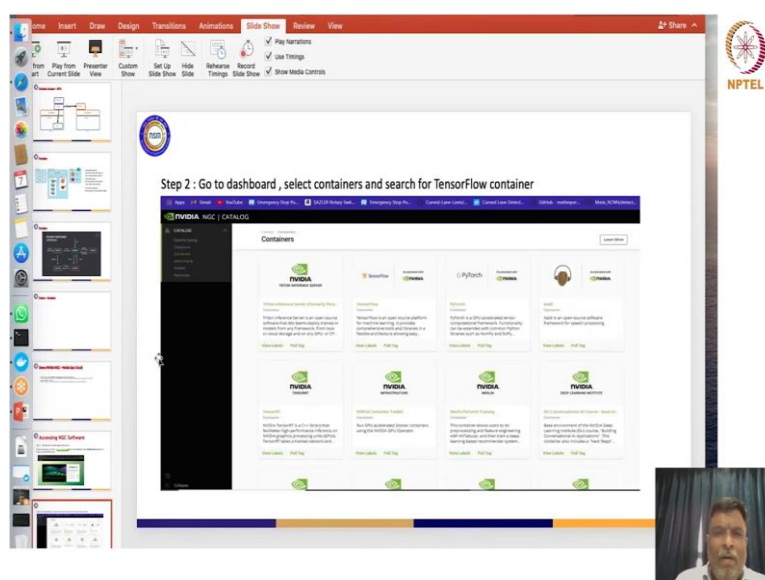


(Refer Slide Time: 24:04)



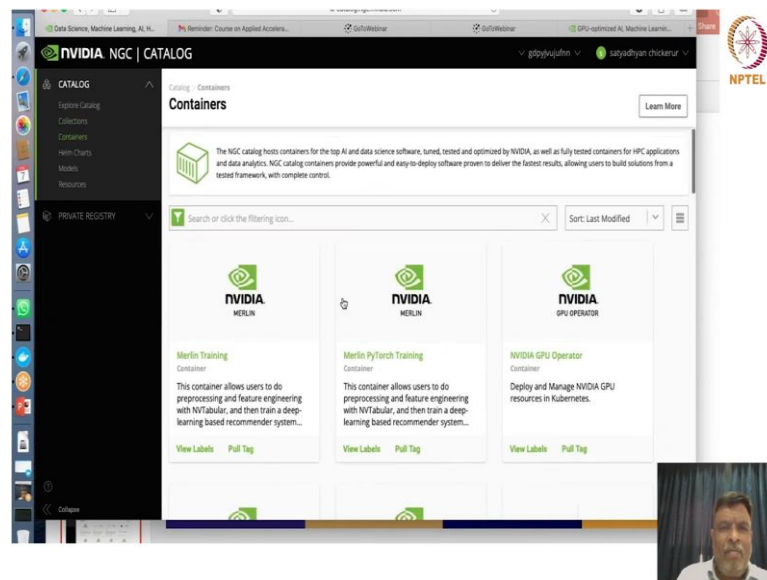
So, what you are going to do is I will have to show you the very starting thing which is there in the PPT because I have logged in. So, you will actually get ok this sign in page ok. These NGC Dockers are very optimized to run on the NVIDIA GPUs which are connected to your laptop, your workstation or your cluster right; any cluster which has NVIDIA GPUs, these Dockers are optimized right.

(Refer Slide Time: 24:31)

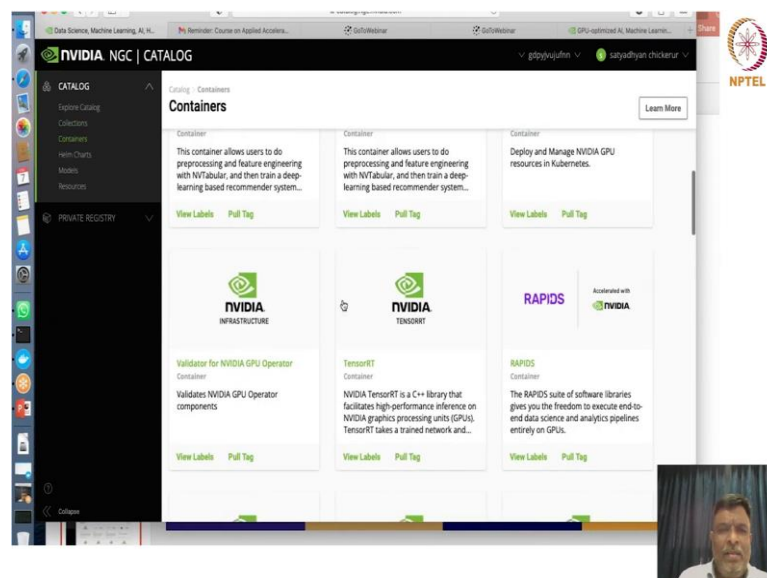


So, once you sign up and log in, you will come across this type of a catalogue which has got lot of containers right; TensorFlow containers, PyTorch containers.

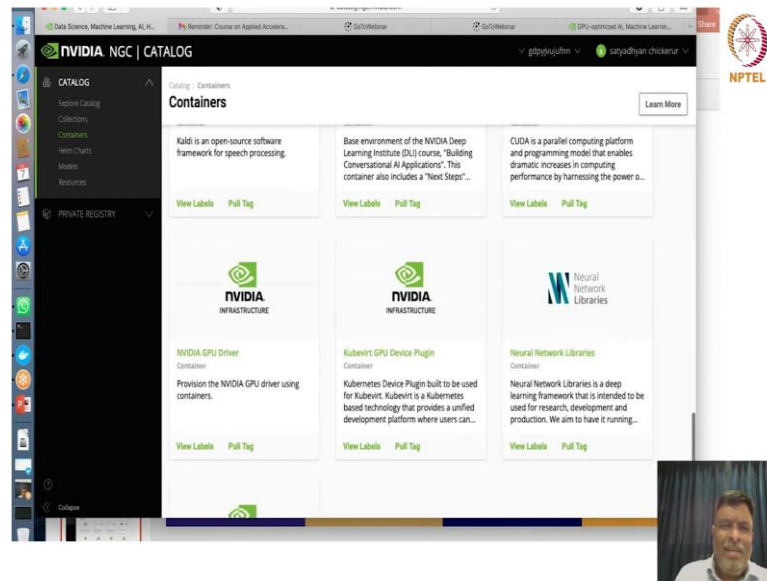
(Refer Slide Time: 24:47)



(Refer Slide Time: 24:50)

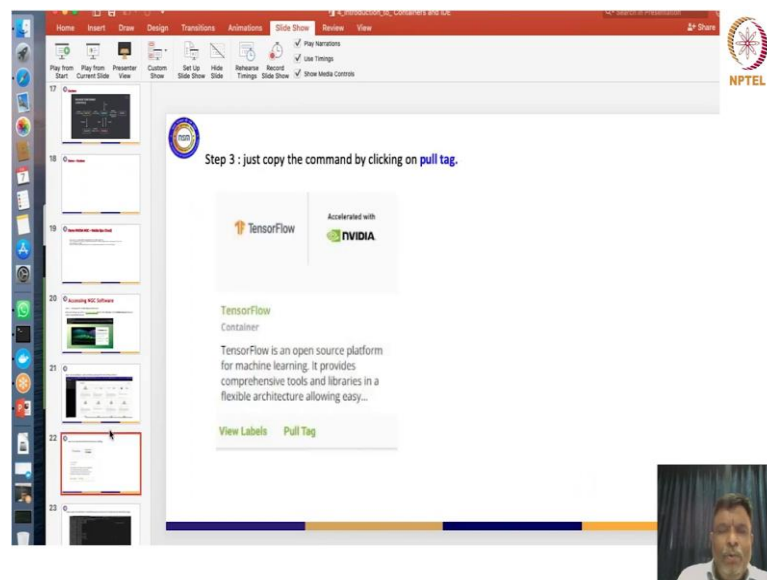


(Refer Slide Time: 24:55)



So, many of them right, if you see this. So, I will show you in detail right, if you explore the collections you have containers right, lot of containers ok, TensorRT. So, many of them right, there are plenty of them ok. All optimized to run.

(Refer Slide Time: 25:05)



(Refer Slide Time: 25:07)

Step 4: paste the command in command prompt and execute it to download the TensorFlow image.

```
docker pull tensorflow/tensorflow:2.2.0-gpu
```

The terminal output shows the image being pulled from the Docker registry.

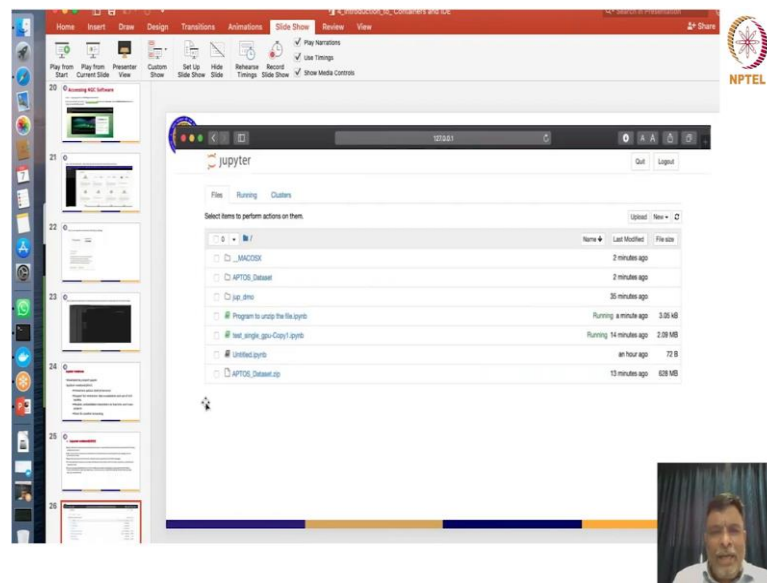
So, now, once you are able to do this ok; once you are able to do this, you can pull a TensorFlow Docker ok and we have pulled it. It goes like this, you have got a TensorFlow thing and then you can run a Jupyter Notebook.

(Refer Slide Time: 25:17)

Jupyter notebook

- Developed by project jupyter.
- Python notebook(2011)
 - Interactive python shell at terminal.
 - Support for interactive data visualization and use of GUI toolkits.
 - Flexible, embeddable interpreters to load into one's own projects.
 - Tools for parallel computing.

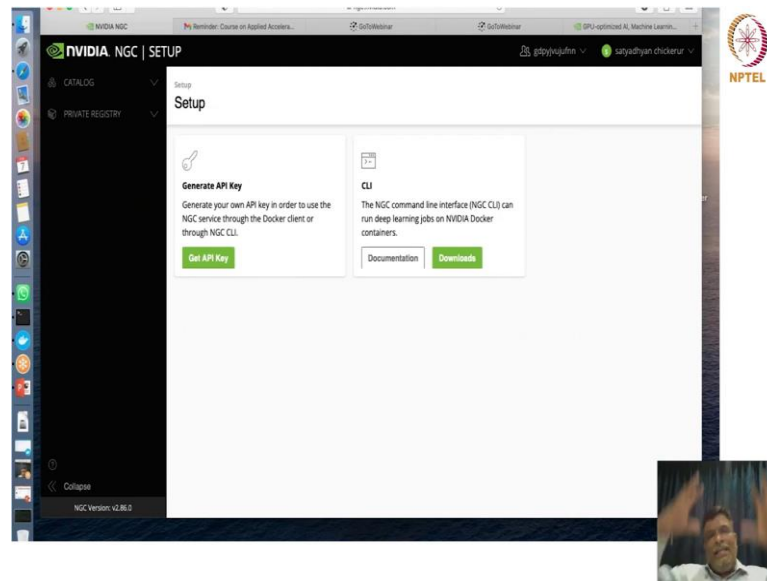
(Refer Slide Time: 25:26)



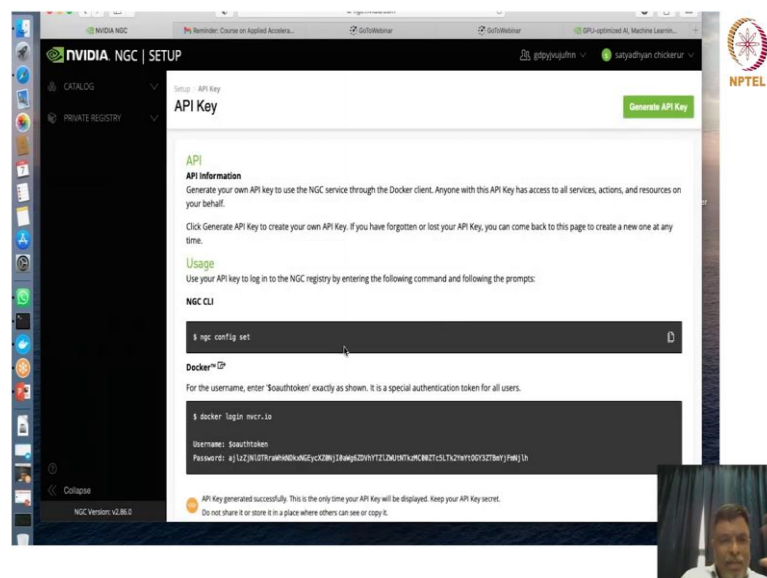
So, I just thought like one or two slides on Jupyter Notebook, not much on it. So, you will get a Jupyter Notebook like this which you can run on it. And, then we will show you how do we do it right. Because, now the idea is there is a Docker container which is running on my DGX, wherein it gives me a link to a Jupyter Notebook, but that is not having a GUI ok.

It actually has to be forwarded right, that information has to be forwarded to my client system from where I am working on, because I need to access the Jupyter Notebook right. So, that particular step we will show you, but once you are able to register ok; one of the things which you should do is from the command prompt ok, from the command prompt before you access these containers, there is a step which I will show you.

(Refer Slide Time: 26:30)

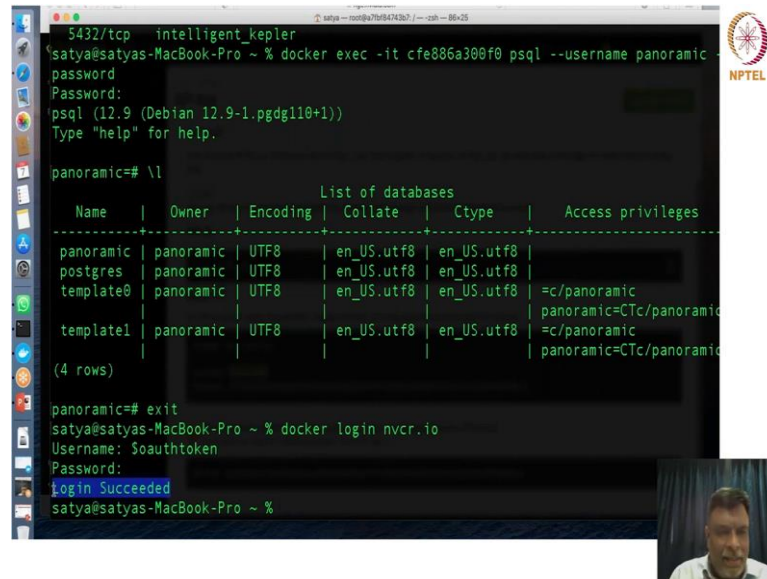


(Refer Slide Time: 26:47)



The setup if you go yeah, but there are two things right. This is a cloud, in the cloud there are lot of containers available. So, to access the container which is there in this NGC cloud, there is this API key which you need to generate ok. And, from this Docker hub sorry Docker prompts from where you are trying to access your Docker ok or you are to going to download the Dockers, you need to actually do this docker login nvcr.io ok.

(Refer Slide Time: 27:08)



The image shows a terminal window on a macOS system. The user is logged in as 'satya' on a 'satya-MacBook-Pro'. They run the command 'docker exec -it cfe886a300f0 psql --username panoramic password'. The terminal shows the psql prompt, followed by the user entering 'panoramic=# \l'. This command lists the databases in the container. The output is a table with columns: Name, Owner, Encoding, Collate, Ctype, and Access privileges. The table lists four databases: panoramic, postgres, template0, and template1. The user then enters 'exit' to leave the psql container. Finally, they run 'docker login nvcr.io' and provide a username and password, resulting in a 'Login Succeeded' message.

```
5432/tcp intelligent_kepler
satya@satya-MacBook-Pro ~ % docker exec -it cfe886a300f0 psql --username panoramic
password
Password:
psql (12.9 (Debian 12.9-1.pgdg110+1))
Type "help" for help.

panoramic=# \l

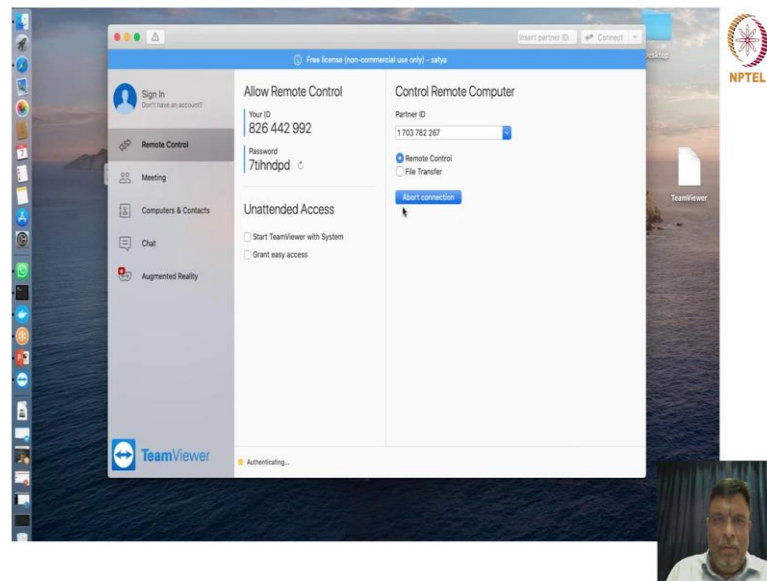
              List of databases
  Name | Owner  | Encoding | Collate | Ctype  | Access privileges
-----+-----+-----+-----+-----+-----
 panoramic | panoramic | UTF8    | en_US.utf8 | en_US.utf8 | 
 postgres | panoramic | UTF8    | en_US.utf8 | en_US.utf8 | 
 template0 | panoramic | UTF8    | en_US.utf8 | en_US.utf8 | =c/panoramic
 template1 | panoramic | UTF8    | en_US.utf8 | en_US.utf8 | panoramic=Ctc/panoramic
              (4 rows)

panoramic=# exit
satya@satya-MacBook-Pro ~ % docker login nvcr.io
Username: $oauthtoken
Password:
Login Succeeded
satya@satya-MacBook-Pro ~ %
```

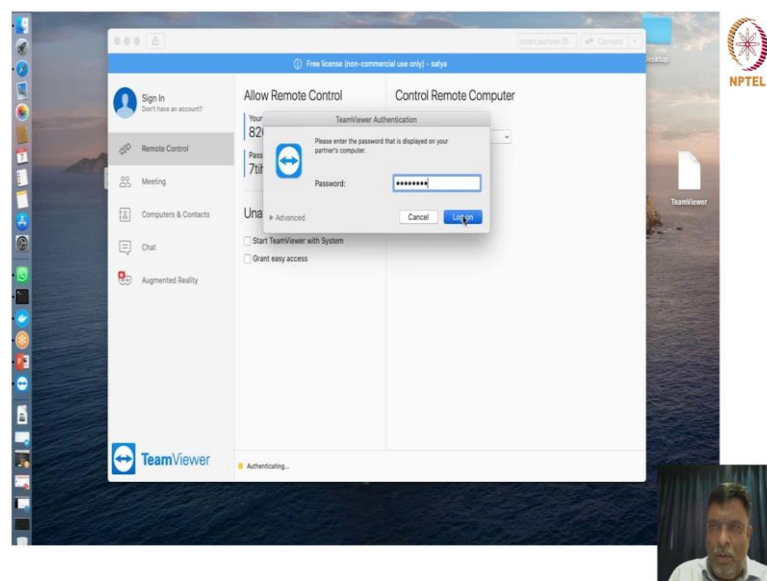
So, you are going to do this your username, Username is the same for all of us. So, it is dollar o authenticate stuff paste it, then it will ask for the Password and this password will be your API key right. So, you will have to save it somewhere or you can regenerate the password whenever you want. So, what I will do is I paste it and then I get this message that Login Succeeded. The once this login gets succeeded, you can access any Docker container from this collection.

But, if you are not logged in ok, you cannot download the containers. So, here doing this we have downloaded the PyTorch container, a TensorFlow container right. We have downloaded lot of them, but today we are going to show you the TensorFlow container right. So, let me just show you that using our DGX system. So, I am accessing that through, can you give me a sec yes.

(Refer Slide Time: 28:43)

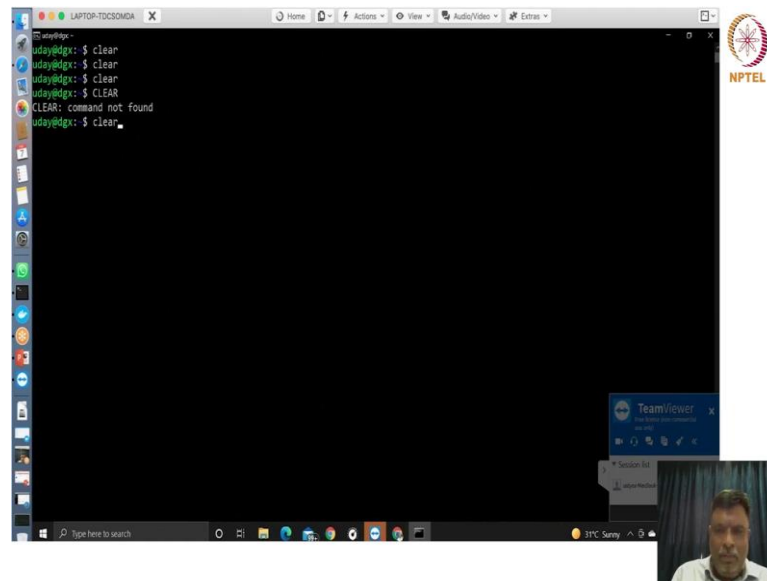


(Refer Slide Time: 28:59)



So, what is the ID? 1 703 782 267, Connect, Password is 1mi543t4. So, Log on ok.

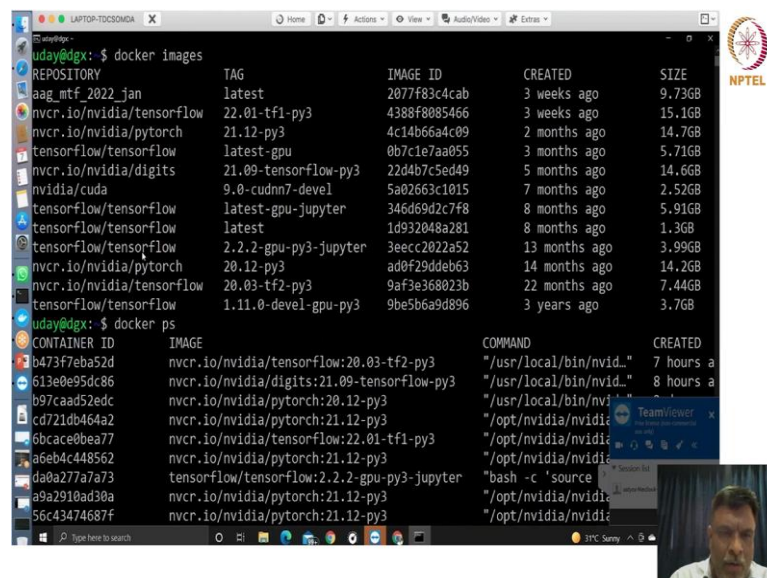
(Refer Slide Time: 29:14)



So, 1 minute let me increase the size of it yeah, 1 minute. So, still the size is ok. So, 1 minute let me just increase the size of it yeah; suppose there is some issue, yeah suppose is it visible?

Student: Yeah, it is visible.

(Refer Slide Time: 29:56)

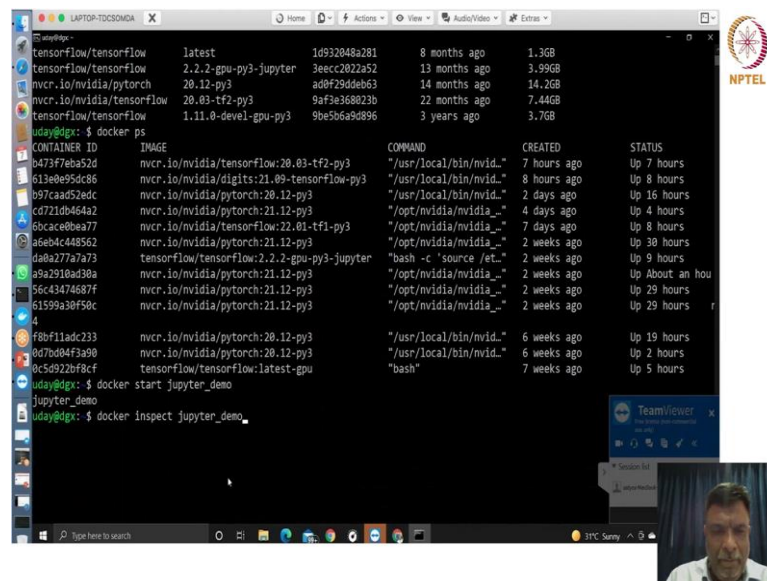


So yeah so, now, we will show you yeah docker images. There so many docker images in the repository and we have downloaded from nvcr.io right. These are docker images,

some of the docker images we have actually renamed them so whatever, but the idea is that these are the docker images.

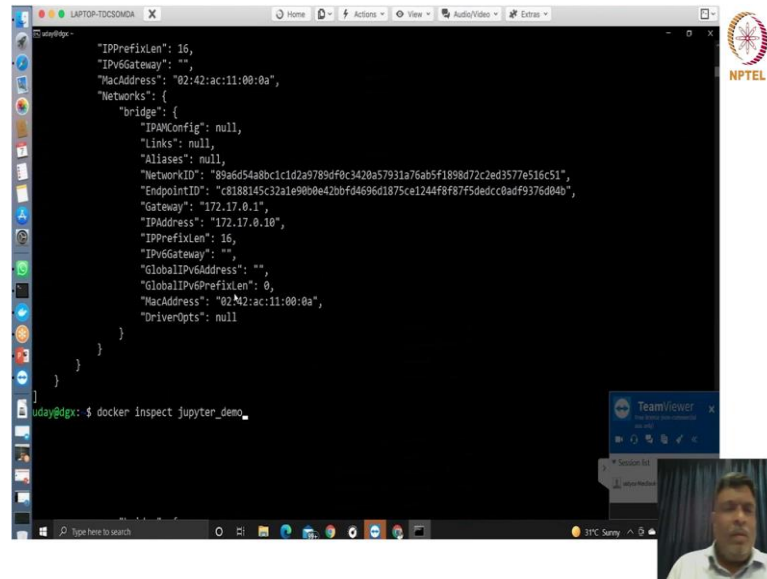
So, now, we have downloaded a TensorFlow Docker. We are just trying to show you using TensorFlow and Jupyter Notebook, how can you run some workload on the DGX or on the cluster right, using remote access, using you know client server type of a methodology right, yes.

(Refer Slide Time: 30:48)



So, now these are all the dockers which are there, those are images, these are dockers which we are trying to run. So, these when were they created, what is the status right of all the dockers on our DGX system yeah. So, now we are going to actually show you. So, now, this demo has started. So, let us try to inspect the Jupyter demo yes.

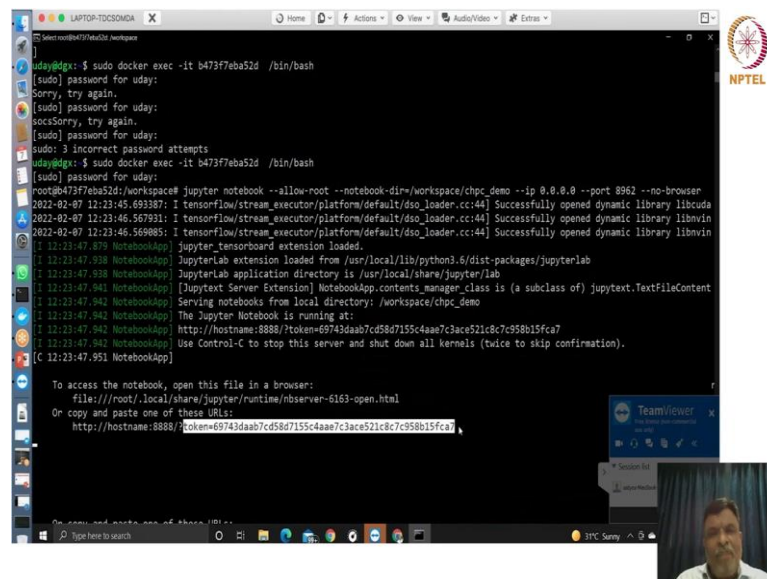
(Refer Slide Time: 31:39)



```
"IPPrefixlen": 16,
"IPv6Gateway": "",
"MacAddress": "02:42:ac:11:00:0a",
"Networks": {
  "bridge": {
    "IPv4Config": null,
    "Links": null,
    "Aliases": null,
    "NetworkID": "89a6d54a0b1c1d2a789d9fc3420a57931a76ab5f1898d72c2ed35776516c51",
    "EndpointID": "c8188145c32a1e9800e42b0f64696d1875ce1244f8f87f5dedcc8adf9376d04b",
    "Gateway": "172.17.0.1",
    "IPAddress": "172.17.0.10",
    "IPPrefixlen": 16,
    "IPv6Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6Prefixlen": 0,
    "MacAddress": "02:42:ac:11:00:0a",
    "DriverOpts": null
  }
}
}

uday@dgx: $ docker inspect jupyter_demo
```

(Refer Slide Time: 32:03)



```
uday@dgx: $ sudo docker exec -it b473f7eba52d /bin/bash
[sudo] password for uday:
Sorry, try again.
[sudo] password for uday:
Sorry, try again.
[sudo] password for uday:
sudo: 3 incorrect password attempts
uday@dgx: $ sudo docker exec -it b473f7eba52d /bin/bash
[sudo] password for uday:
root@b473f7eba52d:/workspace# jupyter notebook --allow-root --notebook-dir=/workspace/chp_demo --ip 0.0.0.0 --port 8962 --no-browser
2022-02-07 12:23:45.693387: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudart
2022-02-07 12:23:46.569931: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libnvin
2022-02-07 12:23:46.569985: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libnvin
[I 12:23:47.879 NotebookApp] JupyterLab extension loaded.
[I 12:23:47.938 NotebookApp] JupyterLab extension loaded from /usr/local/lib/python3.6/dist-packages/jupyterlab
[I 12:23:47.938 NotebookApp] JupyterLab application directory is /usr/local/share/jupyter/lab
[I 12:23:47.941 NotebookApp] [Jupyter Server Extension] NotebookApp.contents_manager_class is (a subclass of) jupyter.TextFileContent
[I 12:23:47.942 NotebookApp] Serving notebooks from local directory: /workspace/chp_demo
[I 12:23:47.942 NotebookApp] The Jupyter Notebook is running at:
[I 12:23:47.942 NotebookApp] http://hostname:8888/?token=69743daab7cd58d7155c4aa7c3ace521c8c7c958b15fca7
[I 12:23:47.951 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).

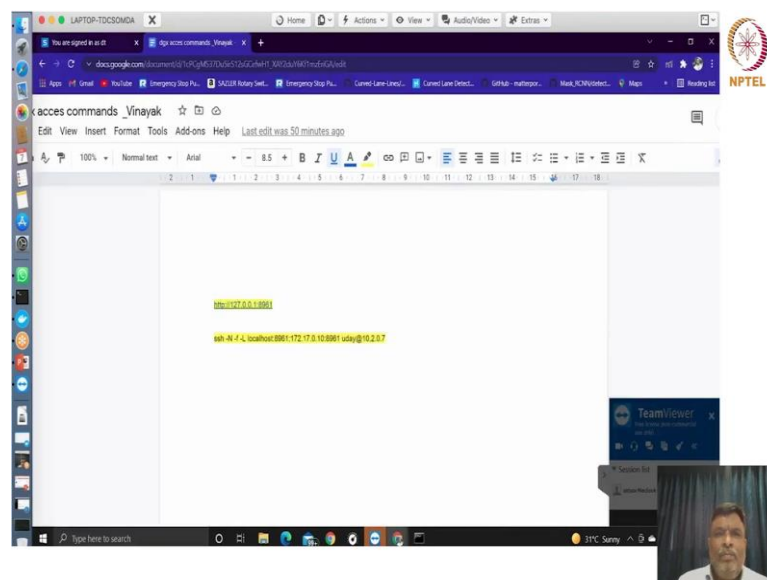
To access the notebook, open this file in a browser:
file:///root/.local/share/jupyter/runtime/nbserver-6163-open.html
Or copy and paste one of these URLs:
http://hostname:8888/?token=69743daab7cd58d7155c4aa7c3ace521c8c7c958b15fca7
```

So, this has got lot of information, we have got the IP Address ok, IP viz IP Address, the Gateway and all of that which would be useful for us. We will run this now, typing off the password, what happened? Yes, ok. So, this is the workspace of our container yes. So, this is how we try to run our Jupyter Notebook in that container. One of the things which we should remember here is this particular port number right which we are trying to change is because the ports number or the port would be in use right.

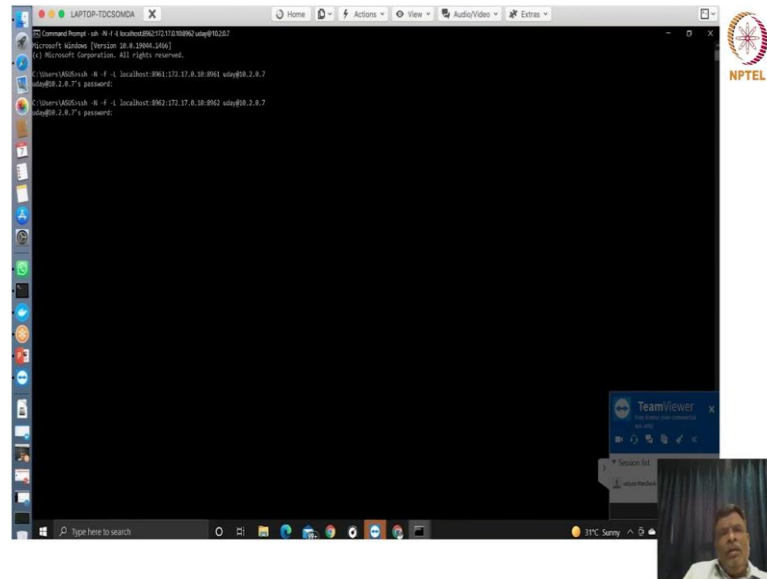
So, we will have to change the port number as you do experimentation, as you try working on this you will come to know right that this is one thing which will be of use to you that you need to go on changing the port numbers, because the earlier port number would have been used. So, it cannot be connected right.

So, this is one thing which you should keep in mind, yes. So, we will run this yeah. So, now, we have run this and now we have a Jupyter Notebook running right on the server, on the DGX server. But, we cannot view it right, because we will have to open it in Jupyter Notebook.

(Refer Slide Time: 33:49)

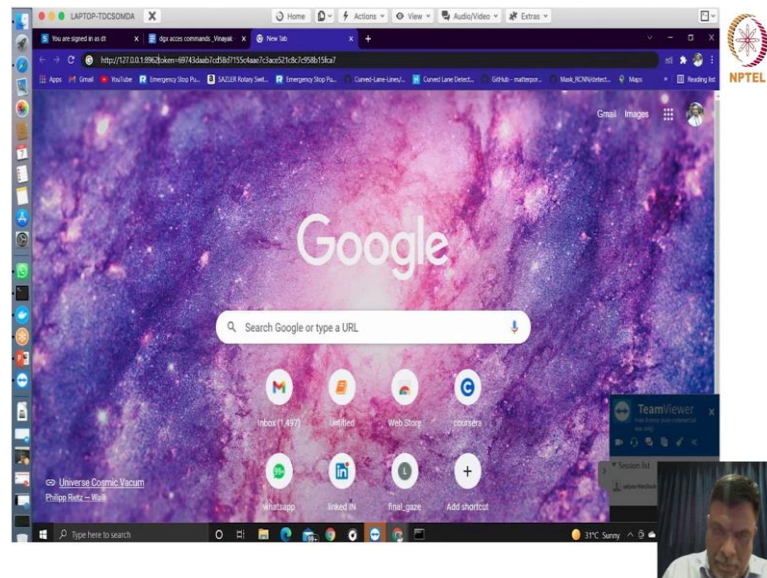


(Refer Slide Time: 34:02)



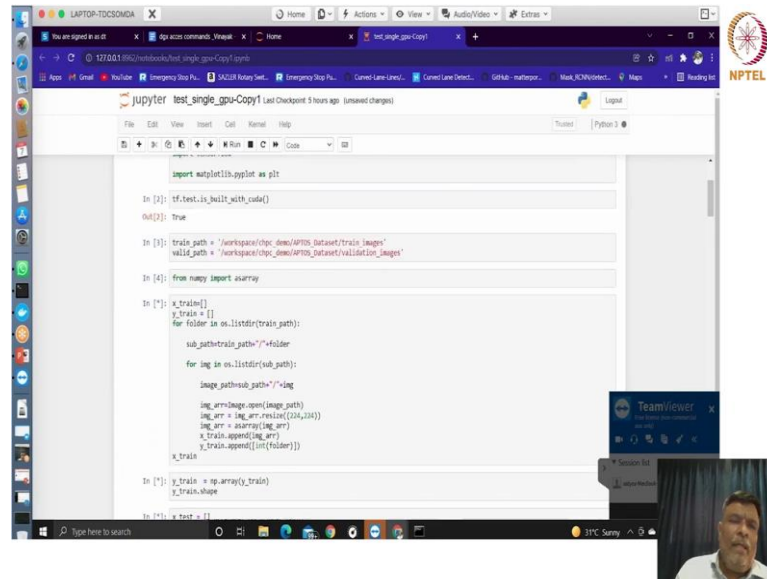
So, what we do is we try to open it using the IP address of our local system, we try to do it. This is called as port forwarding and we are trying to do a port forwarding from local host to the IP address of the DGX along with the username and this thing ok. And, then we would now see it on the Jupyter Notebook of our local host yes.

(Refer Slide Time: 35:02)



So, this is the Jupyter Notebook which is actually running on the DGX and we are having it available for us to use on our client right.

(Refer Slide Time: 36:01)



A screenshot of a Jupyter Notebook interface titled 'test_single_gpu-Copy1'. The notebook is running on a remote system, as indicated by the 'You are signed in as it' message at the top. The code in the notebook is as follows:

```
import matplotlib.pyplot as plt

In [2]: tf.test.is_built_with_cuda()
Out[2]: True

In [3]: train_path = '/workspace/dgx_demo/TFDS/dataset/train_images'
        valid_path = '/workspace/dgx_demo/TFDS/dataset/valid_test_images'

In [4]: from numpy import ndarray

In [5]: x_train = []
        y_train = []
        for folder in os.listdir(train_path):
            sub_path_train_path = folder
            for img in os.listdir(sub_path_train_path):
                image_path = sub_path_train_path + "/" + img
                img_arr = img_arr.imread(image_path)
                img_arr = img_arr.resize((224, 224))
                img_arr = ndarray(img_arr)
                x_train.append(img_arr)
                y_train.append(int(folder))

In [6]: y_train = np.array(y_train)
        x_train.shape
```

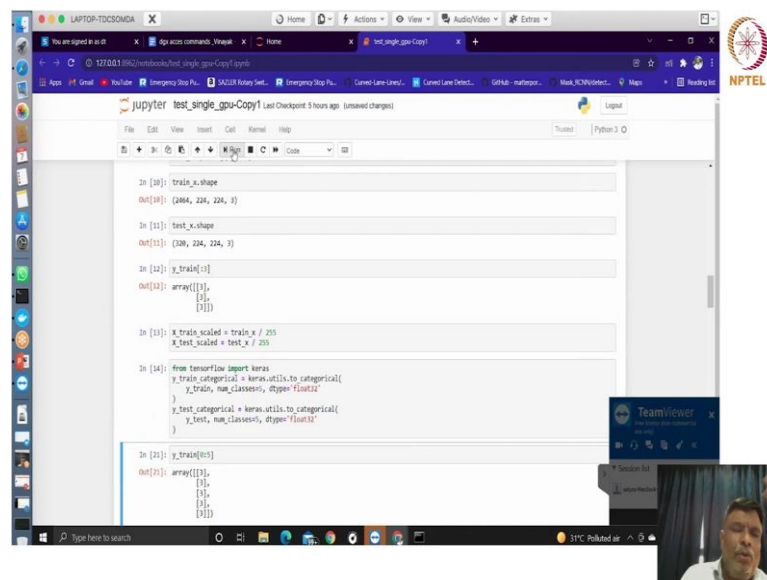
The output of the last cell is:

```
Out[6]: (1000, 224, 224, 3)
```

The notebook interface includes a file explorer on the left, a toolbar with icons for file operations, and a 'Run' button. A small video feed of a person is visible in the bottom right corner.

And, then its a very simple image classification type of a very small program, just to show you that using this type of a methodology using a Jupyter Notebook right, you can actually train your model on the DGX or on the cluster ok.

(Refer Slide Time: 36:36)



A screenshot of a Jupyter Notebook interface titled 'test_single_gpu-Copy1'. The notebook is running on a remote system, as indicated by the 'You are signed in as it' message at the top. The code in the notebook is as follows:

```
In [10]: train_x.shape
Out[10]: (1000, 224, 224, 3)

In [11]: test_x.shape
Out[11]: (100, 224, 224, 3)

In [12]: y_train[0]
Out[12]: array([[3],
               [3],
               [3]])

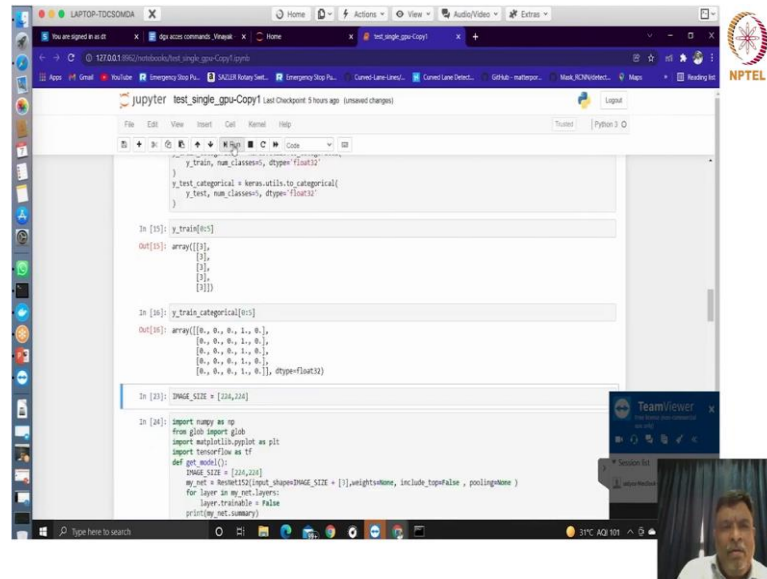
In [13]: x_train_scaled = train_x / 255
        x_test_scaled = test_x / 255

In [14]: from tensorflow.keras
        y_train_categorical = keras.utils.to_categorical(
            y_train, num_classes=10)
        y_test_categorical = keras.utils.to_categorical(
            y_test, num_classes=10, dtype='float32')

In [15]: y_train[0:3]
Out[15]: array([[3],
               [3],
               [3]])
```

The notebook interface includes a file explorer on the left, a toolbar with icons for file operations, and a 'Run' button. A small video feed of a person is visible in the bottom right corner.

(Refer Slide Time: 36:48)



The screenshot shows a Jupyter Notebook titled 'test_single_gpu-Copy1'. The code in the notebook includes:

```
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras import layers, models

# Data preprocessing
train_data_gen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    fill_mode='nearest',
    horizontal_flip=True,
    vertical_flip=True)

val_data_gen = ImageDataGenerator(rescale=1./255)

train_data_gen.flow_from_directory(
    'data/train',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical')

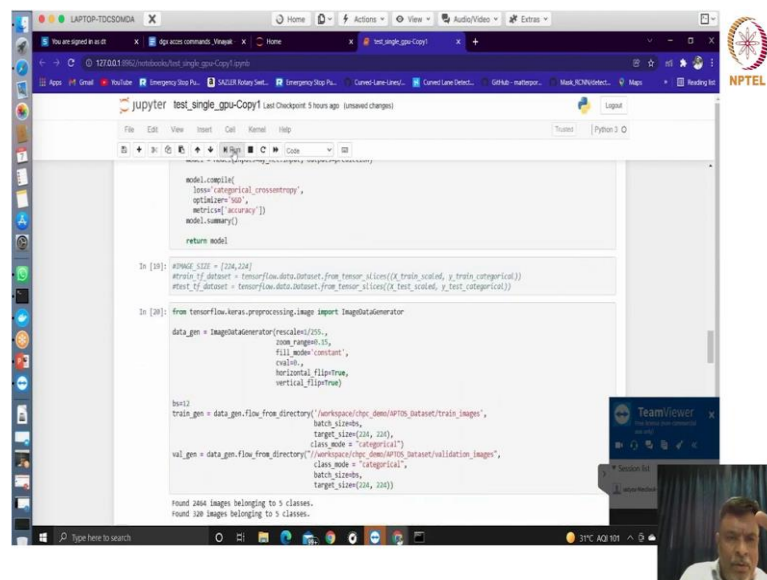
val_data_gen.flow_from_directory(
    'data/val',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical')

# Model building
model = VGG16(weights='imagenet', include_top=False)
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

The output of the code shows the shape of the training and validation data, and the summary of the VGG16 model.

And, then I am not going into the details of the program, that you will see when you go into your further classes. But, I just wanted to show you how you can use Docker for running ok any of these machine learning, deep learning programs ok.

(Refer Slide Time: 36:59)



The screenshot shows a Jupyter Notebook titled 'test_single_gpu-Copy1'. The code in the notebook includes:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras import layers, models

# Data preprocessing
train_data_gen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    fill_mode='nearest',
    horizontal_flip=True,
    vertical_flip=True)

val_data_gen = ImageDataGenerator(rescale=1./255)

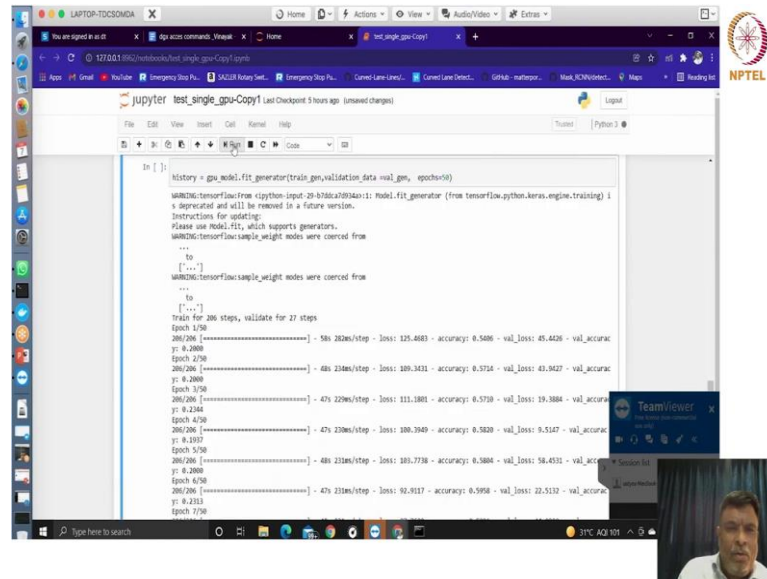
train_data_gen.flow_from_directory(
    'data/train',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical')

val_data_gen.flow_from_directory(
    'data/val',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical')

# Model building
model = VGG16(weights='imagenet', include_top=False)
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

The output of the code shows the shape of the training and validation data, and the summary of the VGG16 model.

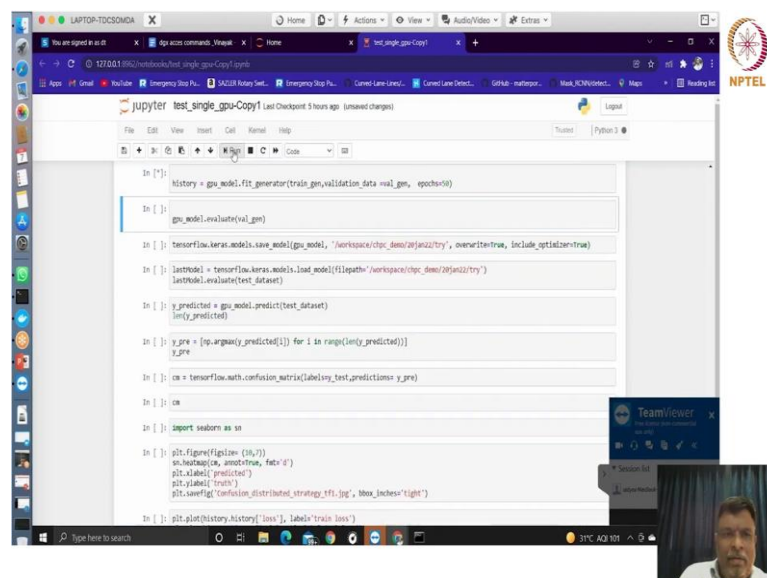
(Refer Slide Time: 37:04)



```
In [ ]: history = gsu_model.fit_generator(train_gen, validation_data=val_gen, epochs=10)

WARNING:tensorflow:From <ipython-input-20-878da28934a1>:1: Model.fit_generator (from tensorflow.python.keras.engine.training) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
WARNING:tensorflow:sample_weight modes were coerced from
...
to
...
WARNING:tensorflow:sample_weight modes were coerced from
...
to
...
Epoch 1/10
200/200 [-----] - 58s 282ms/step - loss: 125.4083 - accuracy: 0.5408 - val_loss: 45.4428 - val_accuracy:
y: 8.2000
epoch 2/10
200/200 [-----] - 48s 234ms/step - loss: 889.3431 - accuracy: 0.5716 - val_loss: 43.9427 - val_accuracy:
y: 8.2000
epoch 3/10
200/200 [-----] - 47s 229ms/step - loss: 111.1801 - accuracy: 0.5718 - val_loss: 19.3884 - val_accuracy:
y: 8.2344
epoch 4/10
200/200 [-----] - 47s 229ms/step - loss: 100.3949 - accuracy: 0.5820 - val_loss: 9.5147 - val_accuracy:
y: 8.1937
epoch 5/10
200/200 [-----] - 48s 231ms/step - loss: 103.7738 - accuracy: 0.5884 - val_loss: 58.4531 - val_accuracy:
y: 8.2000
epoch 6/10
200/200 [-----] - 47s 231ms/step - loss: 92.9117 - accuracy: 0.5998 - val_loss: 22.5512 - val_accuracy:
y: 8.2313
epoch 7/10
```

(Refer Slide Time: 37:09)



```
In [ ]: history = gsu_model.fit_generator(train_gen, validation_data=val_gen, epochs=10)

In [ ]: gsu_model.evaluate(val_gen)

In [ ]: tensorflow.keras.models.save_model(gsu_model, '/workspace/chp_demo/ai/jazz/try', overwrite=True, include_optimizer=True)

In [ ]: test_model = tensorflow.keras.models.load_model('/workspace/chp_demo/ai/jazz/try')
test_model.evaluate(test_dataset)

In [ ]: y_predicted = gsu_model.predict(test_dataset)
len(y_predicted)

In [ ]: y_pre = [np.argmax(y_predicted[i]) for i in range(len(y_predicted))]
x_pre

In [ ]: cm = tensorflow.math.confusion_matrix(labels=y_test_predictions, y_pre)

In [ ]: cm

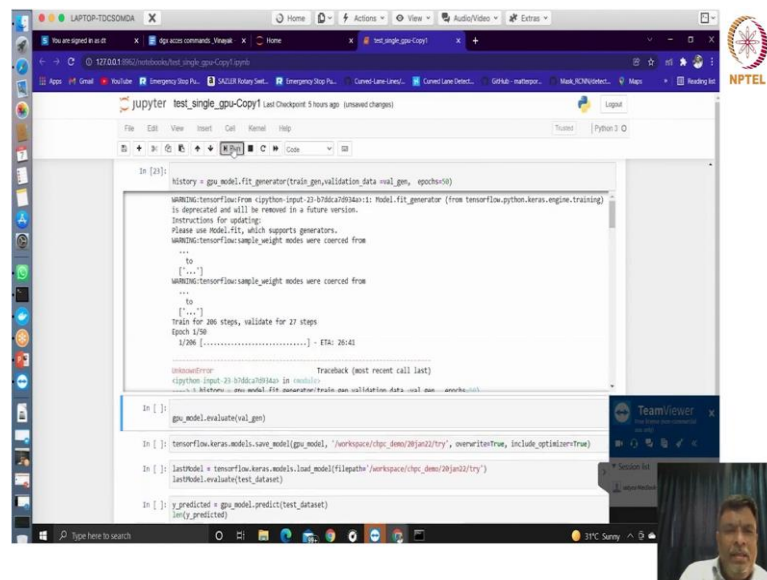
In [ ]: import seaborn as sns

In [ ]: plt.figure(figsize=(10,7))
sns.heatmap(cm, annot=True, fmt='f')
plt.xlabel('predicted')
plt.ylabel('truth')
plt.savefig('confusion_distribution_strategy_xfi.jpg', bbox_inches='tight')

In [ ]: plt.plot(history.history['loss'], label='train loss')
```

So, we are trying to use TensorFlow with Keras, preprocessing. So, we get the gpu model also right and then we try to do the validation or so many epochs and then so on and so forth right.

(Refer Slide Time: 37:24)



```
In [2]: history = gpu_model.fit_generator(train_gen, validation_data=val_gen, epochs=20)

WARNING:tensorflow:From c:\python-input-23-b78da7894a\1: Model.fit_generator (from tensorflow.python.keras.engine.training)
is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
WARNING:tensorflow:sample_weight modes were coerced from:
...
to:
[...]
WARNING:tensorflow:sample_weight modes were coerced from:
...
to:
[...]
Train for 200 steps, validate for 27 steps
Epoch 1/20
1/206 [...] - ETA: 26.43s

UnboundError: c:\python-input-23-b78da7894a\1 in module:
...
x: tf.Tensor<gpu_model_fit_generator/train_gen validation_data=...>

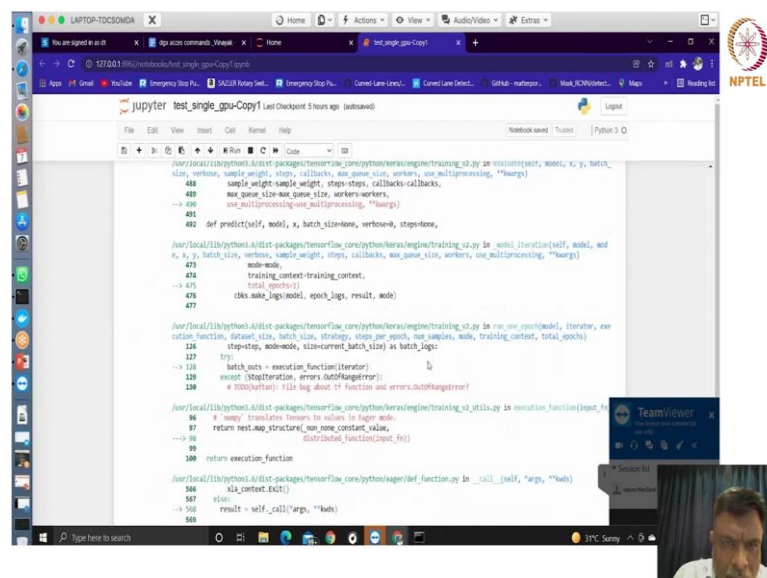
In [ ]: gpu_model.evaluate(val_gen)

In [ ]: tensorflow.keras.models.save_model(gpu_model, "/workspace/chp_demo/20j20/try", overwrite=True, include_optimizer=True)

In [ ]: test_model = tensorflow.keras.models.load_model("/workspace/chp_demo/20j20/try")
test_model.evaluate(test_dataset)

In [ ]: y_predicted = gpu_model.predict(test_dataset)
len(y_predicted)
```

(Refer Slide Time: 37:34)

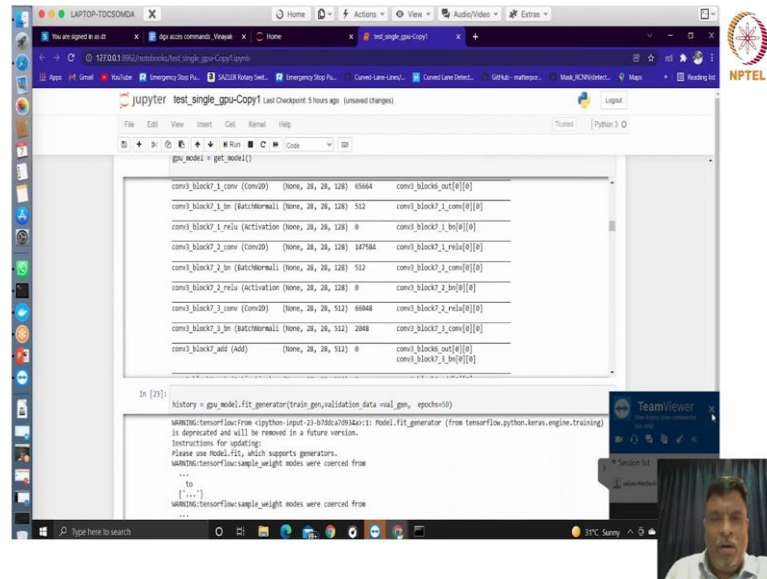


```
#!/usr/local/lib/python3.8/dist-packages/tensorflow/python/keras/engine/training_v2.py in _run_session(self, model, x, y, batch_size, verbose, sample_weight, steps, callbacks, max_queue_size, workers, use_multiprocessing, **kwargs)
488 sample_weight=sample_weight, steps=steps, callbacks=callbacks,
489 max_queue_size=max_queue_size, workers=workers,
490 use_multiprocessing=use_multiprocessing, **kwargs)
491
492 def predict(self, model, x, batch_size=None, verbose=0, steps=None,
493             x, y, batch_size, verbose, sample_weight, steps, callbacks, max_queue_size, workers, use_multiprocessing, **kwargs)
494 model=model,
495 training_context=training_context,
496 total_epochs=1,
497 ckps_save_logs(model, epoch_logs, result, mode)
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
```

A screenshot of a TeamViewer session. The top bar is blue with the TeamViewer logo and text. Below it, a 'Session list' sidebar is visible on the left. The main area shows a video call of a man with a mustache, wearing a white shirt, against a dark background. The bottom status bar shows 'C. Sunny' and weather icons.



(Refer Slide Time: 39:34)



The screenshot shows a Jupyter Notebook window titled 'test_single_gpu-Copy1'. The notebook has a menu bar (File, Edit, Insert, Cell, Kernel, Help) and a toolbar with icons for running, saving, and other actions. The main area displays a table of model layers:

Layer Name	Layer Type	Kernel Size	Stride	Output Shape
conv1_block7_1_conv	conv2d	(None, 28, 28, 128)	65664	conv1_block7_out[0][0]
conv1_block7_1_bn	BatchNormalization	(None, 28, 28, 128)	512	conv1_block7_1_bn[0][0]
conv1_block7_1_relu	Activation	(None, 28, 28, 128)	0	conv1_block7_1_bn[0][0]
conv1_block7_2_conv	conv2d	(None, 28, 28, 128)	147584	conv1_block7_2_relu[0][0]
conv1_block7_2_bn	BatchNormalization	(None, 28, 28, 128)	512	conv1_block7_2_conv[0][0]
conv1_block7_2_relu	Activation	(None, 28, 28, 128)	0	conv1_block7_2_bn[0][0]
conv1_block7_3_conv	conv2d	(None, 28, 28, 512)	60848	conv1_block7_3_relu[0][0]
conv1_block7_3_bn	BatchNormalization	(None, 28, 28, 512)	2048	conv1_block7_3_conv[0][0]
conv1_block7_3_relu	Activation	(None, 28, 28, 512)	0	conv1_block7_3_bn[0][0]

Below the table, a code cell is shown with the following text:

```
In [21]: history = gsu_model.fit_generator(train_gen, validation_data=val_gen, epochs=50)
```

Below the code cell, a warning message is displayed:

```
WARNING:tensorflow:From c:\python-input-2\3-7\6da70934a11: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
WARNING:tensorflow:sample_weight modes were coerced from
...
to
['...']
WARNING:tensorflow:sample_weight modes were coerced from
...
```

So, I think that we can run it like this and then we will get some results, ok. So, I suppose yes; so, this is how if you run ok, you will be able to get the output on your machine learning models yeah. So, I suppose I am done; so yes.