DevOps - Module 8: Docker For Beginners

DevOps

Module 8: Docker For Beginners

Video: Docker For DevOps

Process all together.

If I say technically, uh,

to perform all these activities like from the

after, the coding is building, releasing, going

with sonar queue, all these activities, right?

If I say you do multiple times practice

and all this, also it'll take around two hours easily

for you to implement all of that entire circle.

I'm talking from build to that.

I'm saying, is there any way I can make it much faster?

Because what was DevOps

expecting for you in the first place?

You see right now, I'll take this off.

According to our declaration will go,

so whatever we completed, I'll just make it in orange so

that we know that we are done with all these activities.

Now the goal is how

to get the same thing

at a fast service,

is what I'm trying to say.

That's what DevOps is also expecting, right?

C, understanding application, how to do the bills,

deployments release is one part of the story.

How do you make it more efficient

and faster is another part.

Understanding is pretty, which you should already know.
Getting it done faster is your actual roles
and responsibilities in terms of DevOps.
And you cannot get the things done faster if you don't know
how they work manually in the first place.
Isn't it or not? Now you're clear
how the things are being done in manually.
Now we are going to focus, how am I going to make it faster?
So the outcome also added.
One second,
take this image for reference.
Is that what we did? We had a
complex setup process to set a backend print
and database connect with different systems
in a traditional based approach, meaning using a server,
you did it and almost it took to us to do all of that
in a traditional way.
And already we discussed also, if there is anything complex,
you can make it much simpler and faster.
If you use dog example, sonar queue. Last class
Nexus. I
didn't go through practic. What you say? Hectic setup.
Mm-hmm. I just used Docker to replace all of it.
Then why shouldn't I consider the same
approach for my application?
Also what the point is what I'm saying
right now.
The thing is that complex process for Sonar Cube Nexus,
it's a widely used solution by everyone in the world.
So they already implemented and given it,
but how does someone know, uh, things like Docker know
that I'm going to build LMS application
or someone has CRM application.
Someone has an e-commerce application,

they can't predict it, right?

So people who are building those applications,

if they follow Docker approach,

it becomes more faster and efficient.

Am I clear the techniques that you applied for Sonar Queue?

Why don't you apply the same

technique for your entire project?

That's the idea and that's what we are going to do.

If you understood Docker perfectly,

whatever the work we did last two, three sessions,

the same work, you'll do it within two, three minutes.

That's the outcome of Docker. Simple.

Why we use Docker right now in DevOps

or what is the outcome of Docker is simply you

maximize all the things like in terms of efficiency,

fastness, reliability,

all those we are going to get with Docker.

Traditionally, we used to work on servers, right?

Once we move into Docker,

we have a concept called as containers.

The technology is called containerization. What is it?

How it works, why we use it even in the first place,

we'll try to go understand.

So technically speaking,

you're applying containerization techniques on your project

so that you can run your projects at a faster pace.

That's the outcome. By end of Docker,

you literally run your LMS application in two minutes.

Clear everyone what I'm trying to say.

Video: Containerization

In terms of definition what it is

and then I'll start working.
Yes, as I said, containerization is one of the practices
that you implement in DevOps, which provides
a lightweight and efficient way of running the applications,
which will promote faster deployments
and as well as portability.
Technically speaking, they're saying the outcomes
of containerize, if use containerization,
you'll get lightweight environments.
They'll be more efficient than
managing them via traditional approach.
They're going to be faster as well.
And then yes, they're going to be portable.
I'll tell you, lightweight means pretty much you know it.
So assume that you have some uh,
application which is going with, let's say
it's not the best exam.
We have AUN two operating system, right?
What was the size of Ubuntu operating system
in our initial sessions
around like 1.7 gb, something like that, right?
The same thing here. I'll get for ATM B.
How and why You learn it's very lightweight.
An example I'm telling. I'll see all those practically.
Also, once we start efficient, I said, right.
When I say efficient, simple as I reduce the cost
from 1.7 to ATM B, it's very lightweight.
So deployments can be done faster, simple.
If you apply some common sense, let's take a uh, stone, uh,
which is like 50 kg and take five kg stone.
Which one can you throw farther, obviously, right?
The one which is much smaller, easy to handle for.
That's the same technique we are going to apply here.
Those are all techniques. Technically speaking we use

techniques to em.
Same thing like your computers.
They were very big in size when they were
earlier introduced, right?
Eighties and nineties you talk about. But now what?
Our computers are very, very small.
The same concept we're trying to apply. That's it.
Technically speaking, that's what we do.
How it works means because of a software called Docker.
That's what we are learning and implementing.
So lightweight, efficient, fast portability.
So when do we say some device is portable?
You can carry it and what else?
Uh, basically the actual term of portable it refers
to it produces the same output wherever you, that's
what generally portable applications means.
Uh, here. Uh, for example,
your US B sticks are the best examples for portable devices.
I'll take that pen drive. I have some file in it.
I can use it in my mobile if I want.
I can keep it in Mac laptops or I can keep it in Windows
or in Linux, anywhere I can use it, right?
That's what containers are.
Portal meaning if I build my application on a container
anywhere in the world, I can run irrespective
of environments and platforms.
I'll showcase all these outcomes in the future.
Now I'm explaining you the concept,
but practicality, you'll see once you learn the concept.
But these are all the outcomes of containerization.
Now we discuss what it is,
but how we get, we need to understand, right?
It's a practice. You need a tool version.
Controlling is a practice. GI is a tool source.

Code management is a practice. GitHub is a tool.
Static code analysis is a practice. Sonar Q is a tool.
Same way, containerization is a practice.
We follow it to implement it, we need a tool.
Docker is what values, right?
So what Docker says about themselves, let's try
to understand once how they project.
They say we are a containerization platform,
which has becoming very popular in the recent years due
to its ability to simplify the deployment process,
which in turn leads to faster workloads
of that's what they're saying.
We simplify all this workflow, isn't it?
I reduce literally hours of our time
by using Docker in terms of sonar cunet,
that's the same thing we are going to apply
for our project, right?
And I'll take one simple example.
Now the concept is basically container.
Say when I
say container in simple terms, this is what I can say it.
That's what we use a container, we hold some goods, uh,
something like objects in that.
And I pretty much sure, like
whenever I use this term called container,
you might most probably get this thing in your
mind, correct?
And that's exactly what we are going
to do, but it's very small.
So here to make you a little bit more comfortable
with all these concepts, right?
Just to have, collect a few images so
that you can understand this outcome.
Actually why we use RY industries

using docker is the same purpose.
If I have something in a container, let's say
I can ship them over water,
just try to understand the context.
I can ship them via our roads
over trains.
I can ship them and as well as over the flights,
I can ship maybe,
uh, do we have any other means of transportation, left
air, water, land, everything you covered, that's
how you go ahead and keep something, right?
That's the same thing. Try to understand from IT perspective
now what are the ecosystems where we can run things?
Anything pretty much
tell me if anything else can be there apart from this,
a virtual machine or a physical server, clouds server,
maybe your own personal laptop platforms like uh,
Mac, windows, Linux, that's where you learn things, right?
Technically speaking, what we run applications obvious.
So what they're saying, if you have an application
inside a container, it becomes platform
and you can run anywhere in the world you want.
That's the idea. That's how you can connect
how Docker basically works.
And if you see that official site also they,
they basically say the same thing.
Literally.
I'll tell you what this is later on.
Now AI is the trend. So basically they
replaced application with ai.
I mean like guess even machine learning models, LLMs,
everything can be deployed via Docker.
But not only like what the traditional application we went
with it, the latest and trendy thing like going

and working with all these L LMS and all of it, right?
You can literally use Docker for, we are using it
for our own part, which is why going with it typical
application architecture.
Got it.
So Like I said, right
in our perspective, this is what,
or this is why we are using doggo,
is it clear everyone?
So a container is an object that is basically going
to hold our applications moving forward.
In a context point of view,
there's a symbol of dock that will
these boxes you're seeing, those are containers
which will hold your applications
and once I have it in the container,
I can run anywhere in the world.
Like
there is no need for you
to rebuild across different platforms and all of it.
We just build it once.
Keeping the container, you ship the container,
the product will be delivered.
Exactly. That would be the thing.
Um, if sometimes I have built the application on Linux
platform, I try to run the same application on Mac is
or maybe Windows platform, I'll not get the exact results.
There might be issues in terms of platform dependence.
In Docker you don't have any such thing.
So earlier the problem was rebuilding, considering the,
what is it, target environment, what are its dependencies.
You need to take care of all those.
But now once Docker came,
you literally took away all those things

and that's why it became operationally more faster
to run the things because of tools like this.
Actually DevOps is nothing but literally using these tools
end of the day earlier it was not there, right?
There was no term called DevOps.
Only once these tools got familiar, these things gave the
outcomes that they're looking for.
Then they introduced all these tools
as a part of DevOps ecosystem.
Okay. So I hope now you understood the context,
like why I showed those images
and how it's connected with our applications.
Right. Okay.

Video: Virtualization

The com what they're saying.
That's exactly what we're trying. Develop it faster.
Run anywhere. Run anywhere means what?
There are no operational issues, not technically speaking.
So operations became more faster
and development pace is also more accelerated
because of the techniques that we are going to use.
That's exactly what we are going to perform.
So once I learned Docker, the traditional things
that we used like Nexus,
all these becomes like uh, not so good anymore.
Like I would say what you learned to
where transitioning now there will be a lot
of drastic difference.
Like basically you're moving from
legacy systems to new systems.
Simple and straightforward. So if the things are there in

legacy systems, still you are the person who need
to implement all of it and make it more efficient.
That's what is expected from you.
Or if it's already in the new system, you're supposed
to manage and maintain it.
They will only do outcomes mostly.
Now every application is containerized almost 80
to 90% of workloads.
So most probably when you go there, you might need
to maintain and then you need
to troubleshoot those environments.
That's what your work, if you want to maintain
and troubleshoot, obviously you need to know how it works
and that's what our goal is.
So this way what we are taking the whole
system moving there, right?
You obviously understand the troubleshooting part also
issues that good.
Yeah.
You see in terms of wiki, they're saying Docker is a
platform as a service fraud.
You know there are products in ias model infrastructure
as service model, platform
as a service model, software as a service model.
Docker comes under platform as a service model which
uses OS level virtualization to deliver the softwares
and packages called US container
To understand containers, right?
You need to understand a little bit about uh, the legacy.
Like before containerization, the popular thing that we used
to work on is virtualization.
So we basically are moving from
virtualization to containerization.
If you understand virtualization, it becomes a bit easier

to understand how containerization works
and why we are moving towards containerization.
So virtualization refers to this,
I'm pretty sure you might know it,
this outcome virtualization
involves creating a virtual version
of an actual thing, meaning what?
In context of our things you have in hardware,
you're utilizing that hardware in a better way
by creating multiple missions out of that hardware.
The cloud is best example for virtualization.
That's what you did in the cloud.
You went in the last session I created four missions.
That's all because of virtualization.
Uh, but the thing is, what's the outcome of virtualization?
Is VMware a virtual mission that's bit heavyweight.
So the reason why we came into continuation is also all
because of how applications evolved in the first place.
There would not have been any containerization techniques
if there was no application
development happened in the first place.
They changed the way applications are being built
that basically resulted in containerization.
So we need to also understand bit
how the legacy systems used to work
and why actually things evolved into this manner.
So further that
I'll also give you containers definition,
but it makes bit confusion right now.
So it says that containerization involves what?
Not creating a virtual version.
It says packaging an
application earlier.
You also package an application. Agree or not artifact.

Don't you agree? Let's go back to what we discussed Once,
these are packages only little,
but these packages, they're not portable.
If I give this package
and ask you to run in Windows, it'll not run.
If I give you the package and ask you to run on Linux
or maybe Windows are macros, not sure
and directly, I cannot say I will
install this package and use.
You cannot. Oh, I need no jist
to run the backend package to run the front end package.
I need engine X. If it is not there in my system,
do you think can I host the front end?
That's anxiety we don't want,
which are having some dependencies
and again, set up the dependencies.
To start, I want a whole application as a package,
not artifact as a package.
I hope you can differentiate.
So that is what exactly we are trying to refer packaging
and application and also what
everything that is required
to run an application should be there.
That to what? A lightweight
environment which is more portable.
That's what we are trying.
Virtualization is not containerization.
I hope you understood. Virtualization is specific to
what going through hardware only.
Virtualization entire concept revolves around
how you effectively use hardware.
Containerization. On the other hand, it's more about
how you package applications not the hard.
Yeah, everyone I hope understood the logic, right?

Simple example.

Don't worry about all the things here you're saying.

I want you to only focus on

what is very important right now.

Portion question is what is the advantage of uh,

using the traditional os uh, based

and rather than having the containers,

There is no advantage.

Technically yes. That's why we are

moving to containerization tool

And is it secured?

It is secure. If not, Google runs all their apps.

Right now you're using Gmail. Yahoo, sorry, what do you say?

Google drive documents. What? I'm using maps.

They're all containerized actually.

So it'll be faster performance will be good than the

traditional OS

and infrastructure event using containers or

Yes, correct. Those

are all advantages.

I said they're portable, lightweight, they're faster.

Yes, you get all of those.

It is open source only. Again, open source, right?

Uh, Docker,

See dock.

The things are like very complicated to understand

what is open source and all that.

Docker is open source,

but later on you'll understand to run them.

Also, we need some proprietary tools which will eventually

make sense once we go ahead

Out of the context.

Last question is what is the difference

between the Azure Docker service

and uh, AWS Docker service than the normal docker?
The uh, open source one, I mean the where we can download
and do it, what is the major difference between these two?
I'll tell you simple thing. I'm taking the same example.
What is it you said about three things, right?
One is traditional, the other one is
AWS, the other one is Azure.
Just say I went, I bought the milk,
I made the curd by myself.
That is traditional. I directly went to some supermarket.
I bought Heritage curd A WSI
bought Jesse Curd.
That is Azure. That's the same. Okay?
All the three are same. All the three outcomes are same.
They're just, the way you work is different. Okay?
So basically the underlying concept is always same.
Someone branded AWS and they're calling it my service.
Someone branded Azure. They're calling it the,
Okay, Got it.
That's why I said traditionally,
once you understand the other things are
just all about wrappers.
AWS wrapper, Azure wrapper, Google Wrap, et cetera.
Okay? Right? Yes.
Oh, now as I was saying right here,
the entire focus is application end of right.
If you talk about all of these things,
literally I'm not bothered about all these things
unless my application does not work.
If anything goes wrong on the application,
then I'll start looking into the underlying.
If my application is working, I might not go ahead
and look into these things in the first place.
Now to run this application, this is all mandatory agree

or not in context of our application.

Let's take, I want to run my frontend application.

LMS frontend. I want to run or login application. Frontend.

You did not. I want to run that first thing.

Hardware is agree. Uh, you need it.

Agree or not processing, right?

CPU driving or ramp for, uh, memory allocation.

All that is required. Yeah, so we use cloud.

So someone is going

and giving me the software, which can divide the hardware

that is called hypervisor, VMware,

hyper v, Zen Hypervisor.

These are all softwares.

We are not bothered about it

because when you purchase the cloud

or when you opted for a cloud,

they're handling it on behalf of them.

So it was hidden away from us when I click,

but behind the scenes hypervisor are the ones

who are actually allocating the hardware

on the given hardware.

You have installed some operating system, whereas what

that we generally call it as guest operating system.

On top of the guest operating system, you install binaries

and libraries, which are required

to make this application point in the context

of our application.

These binaries and libraries are nothing, but

those are what you can Azure, binaries

and libraries that are required

for their application to work.

It's not application. This one you're seeing now

like LMS Friend

and if I want to run, then don't you think I need all of it?

That's what I'm talking about. Are you clear? No.
The different layers in terms of application hardware.
Then on top of it, operating system,
why binaries and libraries?
What's the application Now if you see
for one application,
assume like a scale, which is quite big,
the same application they're saying
I will give you on a smaller scale.
Obviously I'm gonna take
it, agree or not.
Am I clear is what I'm trying to say?
I'm just saying let's assume out of context
you have some software.
I'm saying that software is five GB to run it
the same software some other company developed.
I'm saying I'll give you 10 50 mb, what would you prefer?
Lightweight, less cost, easy to manage. That's the same.
Literally that's what we are trying to do.
That's why I said the other one towards the other one.
Two minutes is because of this context
of here everybody.
So here, if you see right, the comparison perspective end
of the day to run this application.
Also hardware is required. That is always there.
So technically what if you want to run the containers?
Also, you need hardware,
but you don't need multiple operating systems to go with.
You have only your guest operating system,
our traditional operating system, whatever it is here.
Also for us, what? Same thing.
I'm not going to change the context as you already continue,
but here I'm not gonna install hypervisors.
I'm going to install, um, container engine.

Technically this container engine is capable
of creating containers, which is nothing
but what Here it is a software called Docker.
That's all. Now Dock
is platform independent,
meaning you can install on, uh, macro on Linux,
Ubuntu, on whatever the platforms you want.
Then, uh, if I say, actually
one second, I
technically this is called vm.
A virtual machine isn't here.
This is called container.
So in the context,
which is more lightweight, that's what we are.
All you need is a container engine like Docker
to be installed wherever you want to use this application.
You want it on your laptop,
you can install Docker in your laptop.
Yes, I want to run it on AWS platform.
You can install docker there.
You want to run it on Oracle platform,
you can install it there.
You have your own physical, uh, data center.
Everything is already configured, installed Docker,
and when I create a container,
I'm not gonna bother about underlying things.
The container itself is application.
You got the idea. The container itself is
app, no installation, nothing.
Bringing and placing, it'll work.
That's why they said we are going to package an application
as a container moving forward
and use it in any platform.
You want to basically go ahead and work clear.

Good sofa. Now actually why it came into the picture.
Let's have an idea because this is also important.
You might get the backstory of how things changed.
This part clear everyone. So I hope you understood.

Video: Monolithic Architecture

Basically it's a style to develop apps
as our architecture.
Same way, monolithic architecture
name itself is saying single thing.
So generally, if I take the dictionary definition
of monolithic, this is what it means.
Monolithic call general. It's a big stone. You can assume.
Now when I'm talking about this style of architecture,
this is what we used to do in the legacy applications.
Understand the style of developing the app.
What is the design style?
Your entire application is built as one single
indivisible unit.
That's why it's quite large.
Generally monolithic applications will be big.
Uh, if I have a monolithic application,
we have like a lot of challenges in terms
of operational work.
I'll take one simple example.
Let's take one real time app itself.
You see guys, I was searching for
Uber application monolithic architecture
And it's visible, right?
That green, you can assume
that like a monolithic in this.
Do you see I have multiple services like functionalities,

you organization building,
passenger management, notifications,
payments, driver related
management, trip management.
These are all different, different set of services,
which makes up Uber application.
Let's see the challenges in context of monolithic.
So let's say I have this payment system.
Uh, currently it does not have UPA payment.
So I want to upgrade UPA payment.
I'm changing only one part of the system, right?
Am I attaching other services?
But to make this new payment work, I need
to rebuild the entire system And eppo,
are you clear what I'm trying to say?
If I want the system to work when I make changes,
whole application needs to be rebuilt.
Not just the payment, the whole app.
That is problem number, like a lot of problems.
But I'll take few things to explain,
which are very prominent.
Second problem, let's say
we have this passenger management.
Uh, let's say a lot of people are going in
searching for the cabs.
Generally that's what happened.
But on the contrast, people who book will be quite less.
They might go ahead and search
and compare across platforms
to the do the booking, isn't it?
So I'm getting more number of requests to this one
as a result to accommodate those needs which are not coming
to other services I need to scale.
My actual goal is to scale only this,

but I cannot scale the whole system again,
you only said right, it's one big unit.
You cannot divide it.
Scaling issues, application building
and redeployment issues.
And also, let's say right now six, we have six services.
We have, we are adding couple of two services for, again,
this size will become much bigger
as a result.
What the startup time stop time for your app is more high,
which will also lead
to operational inefficiencies when we are going
and trying to maintain application.
These are some of the notable issues.
If you go with monolithic architectures,
clear.

Video: Microservices Architecture

Rest, APIs, rest services.
It all is connected to the concept called microservices.
Let's understand from the context here.
Pretty much you can easily get an idea
itself is saying right, micro, which means what?
Smiles, uh, small basically now itself, you can understand
how containers are going to be connected.
Small. I said containers. Also what? Lightweight, right?
Which means obviously it should be small, right?
So when I'm going with microservices, you see
what is the context here?
Here, what I'm trying to say, you take
and existing are a legacy application, which is quite big.
You cut that down into smaller complex.

Each component is called as a microservice.
Don't worry, as a DevOps near it is not your responsibility.
Who will do it? Devs. They rewrite their entire application.
How Uber did. Now,
Uber, the same thing.
What I showed you earlier, this thing,
this entire big app, it was cut down into what
this all circle search signal, this things,
there are individual applications.
You separate all the functionality. Simple.
We will not do it. Developers will go ahead
and write the code for it as another.
They'll use rest endpoint and everything.
It's basically called US web services
in technological context.
If we say it, they're all web services.
Outcome is same. There is no difference The way app works
and everything, but especially
to manage operationally things in efficient way.
They came up with this part and also tell like what, why,
and all this context came.
Now let's discuss the same problems we had earlier.
Now I want to update payment rate.
Now I don't need to rebuild the entire system.
Only that service you rebuild and readable.
Are you clear earlier?
It is hard coded in all of it,
but now it's not.
If I want to scale, only scale this one,
no need to scale the entire system.
You're adding a couple of new services,
it'll not impact the start and stop time of application
because it itself is stated as different apps.
So this is the way we develop the apps.

Another thing comes to this, the end thing boils down to
you are only saying that we are building what applications,
small micro applications.
Why in the world should I go ahead
and keep big infrastructure for small applications?
Does it make sense? So I'm saying if my symbol like this,
how can I keep it in stock?
Okay, let's say you have five MB file,
you're purchasing five TB harvest.
Does it make sense? That's how we used to do earlier.
Five TB was there, but we are not effectively using.
We are just like going in that context.
Now what you have cut them down into smaller, right?
So why you need a big system
to hold a small application was a question here
that five TB is nothing
but yet traditional virtual machines.
Now I'm saying based upon this micro sized application,
I want a micro sized environment.

Video: Need of Containerization

Clear.
So why containers actually came into existence in the first
placements because applications changed.
They evolved over time.
So earlier when containers were not there, we used
to host microservices, ments in VMs only,
plus it literally, if I go 10 years back
where my microservices are, there means in VMs
wasting a lot of resources.
Then they came up, is there any way we can still shrink down
and manage this more effectively means containers.

And that's basically where Docker capitalized the market.
They are the ones who first came up with this concept.
And one more thing, containers are not new.
Actually containers came long back to be specific,
but they were not so easy to manage.
It was all grappling line, external modules, C groups.
If you go deep down into Linx, right? Containers.
Containers is nothing but a technology that came from Linx.
Again, boils down to all Linx.
Again, if I just go back to history,
the docker was introduced when 2030
plus Linx Caners came,
but they were lot help, meaning you need to go completely
inside the kernel, understand it, see groups
and all these concepts used to be there.
You need to work. People thought like this is very complex
to manage the things.
That's when Docker came up with a wrapper.
They said, we you, you use our software.
We'll take away all that complexity.
You just go ahead and run some simple,
simple syntaxes inscription scripts.
They're absolute done. So containers are not you.
They way they were there from long time.
But the thing is,
the way Docker revolutionized it
became completely different.
And if you see the technology also, right?
It says now if, if you see the definition,
it is almost like docker definition.
See they're saying the same thing, right?
Linux containers are nothing,
but it's an OS level virtualization method
for running multiple isolated Linux systems

on a control host using single Linx current.

See the docker part I discussed

OS level virtualization.

Basically Docker, they're saying the same thing.

This is the control host.

Earlier, there was no docker software

directly using Linx kernel level only.

We can perform containers, but very complicated.

Then later on on that kernel,

we got this software called Docker,

which did all the simplification you did and they picked up

and that's when everyone started using containers from there

on and that's how the industry standards were set.

Clear everyone. So I hope you understood the entire thing.

So technically now moving forward, our work is

to do the same thing.

Take your applications and try to run them as containers.

Now what I did in your last class, I went

and I built my entire LMS application,

frontend backend database into one piece.

I'll separate them. Frontend container,

backend container database contains small pieces.

If one goes down, no problem.

I can still get it from other container

and I can still run the things and it'll not take two hours.

One, two minutes is sufficient. That's the idea arch.

Good, clear everyone from the context point of view.

So what's containerization technology

and how it is connected

with application architectures and all of it.

And for same reference, what I'll do, I'll keep that image

what I just discussed in case you have any other doubts.

Also later on you just verify this, right?

And next I think pretty much covered everything, right? Yes.

Now the thing is we need to differentiate
as a DevOps engineer, what is my work in containerization?
Prospect and developers also do,
but it's a different context.
So anything you take the low level layer, VMs,
physical machines, cloud infrastructure or cloud servers
or your laptop, I'm least bothered about it.
All I need is hardware on top of it.
Some operating system, your choice, windows, macros,
Linux, whatever you want.
You go ahead Ubuntu or what do you say? Amazon, Linux.
Red hat. Your choice only thing, what you need is
container runtime.
Docker, you're creating the containers.
Now this application part is not your response.
It'll be handled by what?
Meaning developing that microservice is their work.
Once a develop, will the developer go ahead
and deploy the thing?
We already saw it in our previous sessions.
Deployment is taken care by ops generally,
but are we doing the deployment in traditional approach?
Now you are responsible for deploying containers.
That's it. That's your work.
Are you clear? Same code is written for microservice
building and deploying is your response.
Same thing, building, deploying.
Releasing is your work,
but not in the traditional way, in the containerized way.
We need to do it simple and straightforward.
Further, you need to learn doc.
Now I hope everything is connected
and the main point to go ahead with this,
docker is not the end goal.

So here Docker will give you an outcome,
something similar to what you did in build
what is a build outcome in the traditional approach.
Once the build is in what you get,
which is called as what artifact.
That artifact also you get here in Docker,
technical context they call it as images.
They call software artifact.
Here that artifact is termed as docker image.
So context is same, but the way we work is different.
They also did the build.
It also will do the build,
but not in context of applications, in context
of Docker will do, who will build, not the developer.
You guys who did,
how did you do the build for LMS application in the last
class after receiving the code?
Only build you got code. Right?
Now also we are going to get code.
When I build, I'm not going to get traditional artifact.
I get docker artifact.
The docker artifact. Once you get, you need to run it right,
obviously where you need to run it in a container.
So obviously who should create a container?
If that container failed, who should troubleshoot it?
You have artifact.
Once the artifact is received,
if I did not host on web server, it'll not work.
Agree or not. I just got the dis folder that was artifact.
Technically in terms of front end in the last session,
if I just keep that this folder in Nexus, is there any use
it needs to run somewhere.
Where did you run on some machine.
Now this image, what you got is

relevant are almost similar to this folder.
That image you need to run.
Where we run means inside
contains that.
What? Like there is a concept called docker architecture,
which we are going to discuss tomorrow.
You'll understand all this connecting.
Uh, but here the end goal is this work.
You're learning docker,
but docker is like an intermediate step where you end
and you start picking up from Kubernetes next.
So here the end game is Kubernetes.
If you want to work with Kubernetes,
if you don't know Docker, it's a waste
of going to Kubernetes.
Simple. You don't know how to do the build,
but you have a system writing.
What's the use?
Kubernetes is a system which will help you run
containerized workloads.
Here, Kubernetes is a system which is used
for running what workloads.
Containerized workloads.
If there is nothing container,
what you'll learn, that's a context.
If I say in simple terms, Docker output
is input to Kubernetes.
These tools, they go hand in hand.
If you don't understand Docker,
Kubernetes will never make sense.
Your main goal is Kubernetes.
Now Docker mostly everyone knows I like everyone knows in a
sense in the context of people who are working like
to get it.

Next level, you learn Kubernetes.

In Kubernetes, Docker is input symbol.

So that's where Docker and Kubernetes, they go hand in hand.

If you did not understand Docker, the conceptually things

that happen are, I would say foundations

and Kubernetes will all have is

how Question why we are doing it.

What is the need? All the, you need

to understand from Docker, that's what we are doing, right?

So don't assume that we are learning the Docker, right?

So it's going to fix something. No, not yet.

There are multiple goals actually.

Fastness you get with Docker, reliability, you get with ERs.

This is what you're focusing.

See if it is fast and if it is not relatable,

what is the use said?

Yes, I can go like 300 kilometers per hour, but it'll crash.

Doesn't make sense. That thing, if you want

to make it stable and reliable, Kubernetes is us.

That's project three for us.

Actually, your entire codes, half

of your codes is project two.

End, Docker and Kubernetes.

Technically, almost like one month you'll spend in this two

tools zone because that's the importance of the tools.

So that's why when you're working on Docker, make sure

that it is going to connect with Kubernetes in the future.

In that context, we detail.

Video: Docker Architecture

Working with applications called as microservices

to deploy.

These microservices containers are
what we actually use to implement it.
We are learning simple and straightforward.
You have microservices.
Uh, your company goes
with microservice development approach.
A hundred percent containers with it. That's the whole idea.
Okay? Now let's understand internal of it, how it works.
What are the components you need to be aware
of architecture you need to understand.
Once you are good with it, then we can practically learn,
implement, right?
So let's say what are the Docker architecture components?
First,
these are primary things in terms of docker,
these are all the components you must a hundred percent know
if you want to work with Docker.
One is called Docker demon.
Our docker service also can client our uh, container engine.
Anything is fine. Docker engine, docker service,
container engine, docker, demon,
they're all one and the same.
Then along with the Docker client, CLI
and Docker registry,
which is a location on internet called hub docker.com,
Docker images, Docker contains if you know what they're
and how they're connected, you are good
with the overall flow of Docker.
Then learning the syntax as command is as such. Got it.
Okay. Let's try to understand the thing first.
So, uh,
so technically what was the goal of
learning Docker was I said you want
to make the things faster, isn't it?

And also efficient, portable,

lightweight, correct.

These are all the highlights of Docker I would say.

Now let's say I want to set up some web application.

Let's simply log application in your initial sessions

with deployed login app.

So what is required, if I want

to set up a login application from the scratch,

this thing, you don't have anything.

You have been just given one cloud account

or some physical server

and I ask you to host login application.

So how do you start the process?

You create a server, then next

you install EngineX.

You start in the next service.

You go to GitHub that the code deploy

and do all these activities, isn't it?

I'm saying I want login application itself

as a running entity,

okay?

Or no? Okay, don't.

If you're getting confused, let's think like this.

Let's not go to the big context.

Let's say I want Web Server as a software.

Simple. If you want to host any web application,

what is mandatory web server?

It may be login app, LMS app, X, Y, Z app.

You need a web server. If I want to set up the web server,

don't you need to have this setup?

Ready, started, all the things running.

I'm saying let's not do in the

traditional way what you learn.

We are moving away from the system.

Now we want to make the things faster. Now don't do it.
What we learned in the previous sessions
that was kind of traditional.
So I'm gonna say I want a web server
as a software in that regard.
Docker says we already went
and collected, we did the analysis
and we saw what were the most widely used
entities when you build applications.
Understood what I'm saying?
We already did research in the market
and we got to know there are few things that everyone needs
and everyone works on.
So we took all of those entities and we stored in one place.
The place we call it as a registry.
Generally the meaning are if I say definition wise,
register, what do you store all the names,
IDs, and this stuff, right?
Same way here. It's called Docker registry.
A docker registry is a place where we store
the most widely used entities to build applications
or anything related to it.
That's called Docker registry, which is going
with a location called hub do docker.com.
This is called Docker Registry.
Now I need a web application
to go out and work with further.
Ingenix is something I did right now, I'm no longer going
to traditionally download, install and all this stuff.
I'll go to this search.
Is that what you need?
I got Nginx,
you know it.
nGenx is a web server and all that part.

Now I don't need to do anything.

I just need to run this particular software.

This entity is called as image

In registry we store images.

These are the functionalities that you need.

A regular is what I'm saying.

You no longer need to install

and do all the things what you did earlier.

I'll just go to this docker registry

and then I'll search for what I need.

I'll get there.

And you see here is it

has been downloaded 1 billion times

and almost you see in the last week how much?

98 months, 98 months.

Almost a one pro time close to that part it was downloaded

and installed across different systems

because that's the trend.

Now we will follow the same trend.

Understood the logic is what I'm trying to say. This is X.

Now let's say I want a database called

Postgres for my application.

So again, I need to install, agree or not.

No, I'm not gonna do that any longer.

I'll use Postgres as a software.

I want a backend system that needs to run on no js.

Maybe my technology stack is different. I want Python.

I need a different database like my

SQl need

a different web server

like Apache.

So now I no longer go ahead

and do the setup start process and all of it.

Like traditional Docker said, we already have a registry,

the most widely used softwares, we all collected set up.
Everything is organized as a package
and we call the package as an image.
Now it's no longer ingen X website, ingen X image,
no longer Postgres database, Postgres image,
node JS image, Java image, Python image.
All these images are stored in one place.
It's called registry, which is available
for everyone on the internet.
It's open source.
Are you clearly right? So strictly clear. Image is clear.
Now understand this point
image is like a template.
Think once. If I have nGenx,
couldn't I use nGenx for multiple applications?
Used it for login app, used it for LMS app.
In the future some other app might also come. You'll use it.
Is it like hardcoded to one specific application? No.
You can universally use it for multiple web applications.
No. So that's why it's a template.
A template which I can use for multiple apps.
Now this template is not running, it is stored.
If I want to see engine X
application, you cannot see it here.
It's a template. This template needs to be executed.
It's a simple thing. What a web server it is.
It takes a request, it processes the request.
We'll give you the response back. Agree.
Now to do this processing,
don't I need A-C-P-U-A ramp to store your request?
Maybe some data to present when you want to see something.
Now hardware is required
to run some operation technically like
a web server operation, right?

So this thing, I need to run it so
that it can consume CPU ramp storage.
Now that consuming entity is called container.
You have an image which is a template.
You run that image or you run that template.
The outcome is contained
to run a container.
You need image images
are available in register.
Now you're able to connect.
You have a images are there, images are not running,
images are templates.
To run that template or run that image, we need a container.
Container will utilize CPU RAM storage network.
So eventually where the container should run on a system
or a server on a laptop.
Your choice physical system or a virtual system
or your laptop, which contains hardware
to run this container, you need one software.
Now that is D one.
Alright, image template. You want webs application.
For my LMS style, I need an image called Ingen X.
When I run that engine XI said when I run
to run, I need someone software.
Now that is Docker demon or Docker service.
Are it clear? Ah, now again, one more component comes.
So now you got image clear container, clear register.
Clear demon is what creates
to create you need to give request.
The request is CLL
Docker client.
Are you clear? Docker client is where you give the request.
What request I want to run LMS obligation is a request.
You have given the request. Now

who will take the request from client?
You have given the request. Who will take the request?
Demon will take the request demon.
After taking the request, it'll look what you want.
I want in Gen X demon will go to
register in the registry.
What is the template is there to bring the template.
It'll run the template.
That running entity is called container
are everywhere.
How all these things are gonna connect, right?
I have drawn one diagram also explained,
but I hope you understood the concept.
Uh, now here a system which is
capable of running containers.
That system we call it as docker host,
it takes some random machine if it can run containers.
That machine in generic we call it as what? A docker host.
Now when do you call a system?
A docker host means if it is capable
of running the containers.
Now so what and what software should be there means
docker client Docker deal.
Every docker host will contain Docker demon
Docker client Docker registry is not there
in the host registry is on the internet.
How you have GitHub Docker hub,
GitHub contains what code.
Docker hub contains uh, docker images which are capable
of running the softwares clear.
All good everyone. Now let's connect all these in one
picture and try to understand
my docker hub, which is also called Docker registry
and say I got all these images.

Docker host is nothing

but you install Docker on a system

that will be called as Docker host.

Once you install Docker host, you'll get Docker

CLI client

and also Docker service,

which is basically call us Docker Demon.

Okay? You've got Docker client,

you've got Docker service via client.

I said you'll give the request, isn't it?

I'll show you one simple example how the syntax was.

I'm not talking about the syntax, but what how it works.

I have given one command. You say first step

Docker container run.

See, name itself or command itself is quite self explained.

You're saying run AER with the talker.

The name of the container is

LMS Front end.

You already did the work. So

to run the frontend engine access mandate you, so I said

image

what?

I said I want to run a container using nGenx.

That ingen X there is image that is replaceable.

It can be nGenx, can be post this node, Java, Python,

my skill X, Y, Z.

As soon as you run this command, it is executed

by A-Y-C-L-I.

It goes to docker service.

Client will connect with talker service or a demon.

This is the heart of the system.

Everything it will take there.

So what the system does, as soon as you're done

with the work like executing the command, it'll first look in your system itself.

Do I have this image? See the third step?

First step you run. Second it went to docker service.

It is looking in the system. So this empty box, et cetera.

This is called Docker cache.

Temporary place.

You can see, it'll see if there is any image

with the name you asked.

Do I have? Because it's a brand new system.

There is no image not found. Right?

Next fourth step.

You see

it was not there.

So docker service, what it did, it went into register.

Your demon is configured in such a way

that if something is not found, it'll look in registry.

If it is not found in the registry, also it'll say not form.

Simple, clear.

It went to Docker release,

did it found the image, fifth step.

It took that image

and what it did, it downloaded that image

and it kept in your cache

per saving in the future bandwidth.

You don't need to again go back and download.

Now what we have in image, this is not application

and this is what image which holds what?

How to run Engine X.

Once downloaded that image, see the six step,

it'll create the container.

That container will have your NX in the running,

meaning you don't need to install directly.

Hit that container, it will load.

Welcome to engineers

are clear.

Every next time you see what I said,

why not?

I want to set up RA application. It also has a front end.

Do I need a web server when I run it?

Same first, second, third.

Do we have engineers directly continue

both content engineers

but both serving different applications.

That's how Docker works.

We are going to reuse lot of things

which is very, very efficient.

That's why they're going to be super

fast, super lightweight.

So that is individual.

No, once I do it for Inex,

I'm never gonna do it once again in the life

it's already cached.

It just uses the cached item from there and run the things.

Now clear everyone, demon, clear client registry

images, continuous from online.

All good. Anything further I need

to discuss on this part here?

Any confusion? It's confusing regarding Damon only.

Can you just brief about that? Docker Damon.

Okay, so Docker Damon.

Basically what it does means, as I was saying name itself,

you saying generally, do you know like

what is demon called us?

They the other word actually demons work in the background.

Demon process we call, Nope, not heard of it.

Okay, in Linux generally you'll get it.

Demon services are something that run behind the scenes.

Some server, I'll just take one random server.

Uh, there is one server called HCLI.

It's just like your regular open server only.

I just log in and I'll tell you

it says secure Shell server is running.

You see, what did I say?

T stands for T-S-S-H-D

meaning S, SH.

Background service.

He's the one who is responsible for taking your input.

Agree or not. S, SHI explained you

and you see where it is coming from.

Ss h dot service dot D.

The D stands for D, meaning your web server,

your H server, Postgres server.

They're all background demons. Singular entity.

Now technically I'm saying

when I say Docker container runs something, the Docker demon

from the background is taking your request and doing that.

So why we call it as demand?

Because it'll do all the work, creating the container,

starting the containers, stopping the containers,

removing the containers, building the images.

Everything is going to be taken care by demon,

which you can't see in the front,

but it does all of these activities in the background.

So any activity that is running in the background,

we call it as a demon process.

Docker, demon, whenever you do

anything on the client, right?

Demon says, I'm ready. Give it. I'll take care.

So here, container creation, image downloading,

who is doing only with the

Docker client, you're giving input.

Work is carried out by Demon

from, uh, sorry Kim.

You understood? I hope you got the point now.

Yeah, yeah, yeah.

So technically here,

Docker Demon is nothing but a software only.

We are going to install Docker.

Now at that point of time,

the demon will be installed on the system

to do all these operations and he'll not be seen.

I'll show you. When I use the now, we cannot see obviously

because Docker is not there.

No, like this.

I can check. Later on

it says Docker demon or Container.

Engine or Docker service is running like that.

He's the one who will do all the things for you.

I said not bringing the image, downloading,

keeping in the cache, running the container.

Who will do it? Not client Demon will do it. Right.

Okay, so same thing

for your reference, I'm gonna keep this image.

Just zoom in, you'll see all those right? Okay.

And a quick explanation about this. Just read it.

I'm pretty much sure now you'll easily understand.

Yeah. Docker demon, Docker client, Docker Register.

Video: Docker Images & Containers

That's image.

You see what it is saying? Image is it lightweight,

standalone, executable package,

which includes everything.

It has everything that is needed

to run the desired software.
Okay? What I'm, maybe it might be a little bit confusing,
but once you do the practical,
now you'll understand actually this is what image means.
Simple. Next time owners, I'm not going to do the coding
and all this building and everything.
I'll bring the image down. Image contains everything.
Clear the idea.
See I did the sonar queue.
How much steps were there to install Sonar cube.
But did I do any of it? Agree or not? Did I do any of it?
I just ran one command because
a sonar cube image is there.
Now check from the definition context.
A docker sonar cube image
has a lightweight, standalone
executable package, which requires everything to run.
Technically, if I want
to use Sonar Cube, I need to do all the work.
What I did. Get the code, build it,
post it on a server, run it.
But did I do any of the thing?
If you're still in the confusion mode,
Just for reference,
this is official source Code of Sonar.
It's an open source software. So it is very good.
If I give this code and say, run sonar queue, will it work?
Of course not. It is source code.
Now what do you need to do? You need to go
through all the process again,
build, you see all these things.
Then you'll have it. That was Spotify, five minutes of work.
But did I do it? I just went to Docker registry.
I downloaded Sonar Grant.

Now clear. Now what I'm saying, the same thing.
You're going to apply
for every project in the future if you learn Docker.
Now my project is no longer building going
and doing all this traditional approach.
I want a ready made image that can work
whenever I want, wherever I want.
I hope someone tried Sonar Cuban, Azure.
Also, you'll get the same outcome maybe.
I'm not sure whether it did or not. Same Universal.
So same context. You apply for all applications.
Now clear everyone, right?
So now you've got the clarity with image
and where these images are available.
Exactly. All the images that you're looking for are
where they're available in a registry like Hub dot.
From these images we created container, isn't it?
Now, let's see the definition of container
technical definition.
A container is Docker isn't isolated.
Environment that allows you to run
application are a service
with all the dependencies it needs.
Did you get the point? SonarCube is there.
We needed so many things actually
technically to run Sonar cube.
Now we need Java, uh, grad build software.
We need all these things.
We need to install, run, and go through the process.
Then finally we get sonar,
but it did not do any of this secret.
That's what there's a container is an isolated
environment in which your software runs.
To create this container, we need an image,

and image acts like a template.

Now, to make it much easier, uh, maybe, I'm not sure,

you know, like programming context, classes and objects.

If in case you don't know, fine.

I'll just take a simple example to help you understand.

If you know a classes image

object is contained in programming context.

If you know object oriented programming languages,

we have a concept called like objects and images.

Sorry, classes. Classes will act like blueprint

objects are runtime entities.

That's exactly what images containers are.

So to make it easier, the same example only I took anyway

and I said no already.

So what? You need to create a container.

We needed docker image must without image,

you cannot create container blindly.

You can go with it. Okay?

Now take this image, which, sorry, like a picture

to help you understand all these things put together.

See that car, it's a blueprint.

When I say car in your mind you'll get a picture, right?

Contains like four tires.

It has some shape, it has some glass.

It's a template, something that

clicks in your mind when you think about a car.

But there can be a number of different cars,

but they have all car functionality.

Those are contains.

Video: Docker Setup

Two medium instance are B two s in Azure with around 15 gigabytes of storage

and security groups.

You see we are adding a lot of ports.

I'll tell you why later on.

As of now, just not bother about it.

But we are not adding one R range earlier. Only. What?

One port? No, like 20

to 80, 80, 85, 4, 3, 2.

Like that. You went with some specific.

Now here you need to add a range of books.

Everything from 32,000 to 61,000.

Everything from 80 82, 90, 90 and 80 22 as usual.

These two things you are aware, I don't need

to give explanation on 80 and 22.

Why we use this, let's not discuss about it.

As of now we have docker networking concepts.

At that point of time we'll discuss, okay?

Uh, and then some explanation like predefined

range, user defined range.

But I said no, not to worry about

that in much context light.

Just go ahead and prepare the system like this.

This is

my Docker host where I'm going to run the contains.

Can I install Docker in my laptop?

Yes, Docker desktop is available for Windows,

Macs, linings, desktops.

Also, I can install same commands. Everything is same.

Zero difference, but technically it does not make sense

installing in Labro because in real time we work from

con uh, server zone.

Understand what I'm saying In the desktop also,

I'll show you one example in my system.

So I'm going with as usual, open two,

Version 22.
I'm going with Medium,
which gives me the configuration I need key pair,
edit and in Azure also,
you already know how to add the codes.
Now just go ahead and add in that.
Instead of 80 80, add the range, that's it here, SSH
80 and along with it,
this entire range of,
and along with this range
and some storage.
This thing also, you can do it in Isha like I showed you.
Now in the box, bottom box instead of one code,
remove update the range.
And I'm gonna say what? Launch?
No, I cannot call it technically Docker host reason
because we don't have any Docker.
Now you can install with app command also,
but thing is Docker has given official script.
You don't need to use this
thing directly from script itself.
You can install the package. I said no, there is one.
Get do docker.com script from there and here.
Docker Info is something that we use in order to figure out
what client demo and everything is done.
But as of now, I will not be able to say it
because there is no such thing.
Pretty straightforward.
The installation I think I already showed you in Sonar,
was the same command, uh, status Docker like anyway,
you'll not see because Docker is not installed.
See what you're saying? Docker service, Docker Demon is not
for you.
See what I'm doing? I'm downloading the file call, get

Docker s sh from this website.

If I go there, you'll find the Docker script.

I

said no.

Engine Demon, they're all one on the same.

And then what I'm doing executing the

script, are you curious?

I'll just talk copy the script.

You see it? It says both are ready.

And if you just scroll up a little bit here, what

and what you're seeing,

can you see two different entities or not

client server.

Server is nothing but demon technique.

Both are ready now in your system.

Or you can also you Docker.

Say Docker client is working,

but you see server, what is saying

you need pseudo access for that

because Docker demon is what background process.

Client is what is typical software.

Anyone can use client but demon behind the scenes

or I said no anyone also verify like this pseudo command.

If I go, you

see Docker server now you see it's showing me

all the containers image.

That's the reason why I said it's the main process.

Clear everyone what the point is like this.

I can go ahead and check. Now here

you see what it is saying.

Dock application container in chip.

Video: First Container

You can see it.
It says our Docker client is, sorry,
Docker Demon is active and running.
Now, technically speaking, what?
It's a system which is providing
hardware to run the containers.
That's the meaning actually, Docker host is nothing
but a machine, which is providing
hardware to run the containers.
To run the continuous, we need what mandate is images.
Uh, now if I try to go
and see what images I have, these are the instructions.
Docker image list, like how do you sell list, command,
docker, image list, same permission.
Then I, if I use pseudo, obviously it's gonna work
empty reason because it's a brand new system.
But every time I said now in real world it might not be
possible that everyone will get pseudo access, right?
So Docker says, we have a group called Docker,
like you remember pseudo, uh,
pseudo Psda has basically like a group you can add
and you'll get the permissions.
Same way Docker has a group.
If you add any user to that group,
they can work with containers.
No need route. Only Docker group
because I only want to work with containers.
That's generally the correct
pseudo rep.
Docker
clear what this command is.
Same command we learned in, uh,
permissions and all that part.
Now I'm checking first group called Docker,

and do I have any user there As of now?

No. No. I think you remember this command.

Also

add to the group,

group user.

Now what we did, we added Ubuntu user

to Docker group to reflect this changes.

I log out once logging back.

That's how you fixation.

Remember this point in the future, any tools, software,

anything you want to use with Docker,

add the to Docker group.

Your issues will be fixed as of now.

Any images in the system? No,

no containers also

unless stands for listener.

Basically, if you say this way,

it'll only show you running containers.

All containers. The ones which are also stopped, it'll show.

But you see there are no stopped containers.

Also images will not have any such thing

because images not running entity.

It's just a template. So there is no start

and stop for images, containers start and stop will be there

because they're going to run.

Okay, good. Uh, now I want to go ahead

and start running a container.

And further I need an image.

Uh, the best way to know if everything is working.

Uh, the architecture demon client registry,

everything connected and working as it,

the architecture we discussed.

Now, I want

to see if the system is capable of running all of it.

Docker has already for the same purpose,

Like every programming language they have given

one Hello World app.

Generally you, you know why we run? Hello?

World Apps like Java, Python, anyone

had any idea why we run?

Hello World? I think that's the first thing you do

after installing the Java or Python, right?

Meaning what? It works or not. You verify, isn't it?

Technically that's the same thing.

This hello world image has given to help you understand

that your docker is working as expected their program.

Here it is, image clear,

it's in Hello, world image.

Same thing. Syntax goes like this.

Now you're learning the new things. If I wanna create a

container, what is the syntax?

Docker container run.

That's the syntax. Which image

I'm running.

Hello? World image. Because I want to test

if everything is working as expected

or when I run it, see what happens.

Do I have any image right now? No. Right?

See the statement Step by step.

Anyway, I'll tell you.

Agree. Did we have any

like, you know it, terminology downloading

and final status

download it, meaning what image should be there, right?

Uh, if you wanna confirm, I can confirm that for you.

How do you check images? Are there,

did you get the image?

Because he said No.

Next. Hello,
how you write a program?
They say word. Hello, word, sorry, hello from Docker.
And you see what this says.
That's the purpose of this image
and this remaining four points, if you read, that's
what I explained you architecture, just one C,
the entire mechanism, what we discussed,
it was embedded into that image.
Okay? So if I'm seeing this message,
meaning it should come from a container, agree
or not, is it?
So
that was the container which showed you the message
that you read on the screen.
Now what I do,
I'm saying again, run hello faster.
When I ran what it said,
do you have the image now?
Now is it saying again, downloading again, pulling
images already there, it just run the contain.
So now how many containers will I have?
Two containers, Images only.
That's the whole idea behind this one.
Engine X image. A hundred different applications.
Yeah, that you'll distinguish once we start really setting
up the apps and all of it.
Okay, clear everyone.
Now, I hope you understand the point is this is
how Docker actually works.
A very simple way to understand now our work is toward
utilize these things, what Docker says we are good at
and apply it on top of your applications.
When I say your applications, every app that you're going

to learn are, you'll implement in your projects.
It might be an e-commerce project,
it might be some LMS project
and HRMS application or a CRM system.
Your choice, not C like it's being used
in AI and all of it, right?
They're all models and all
these things are nothing but images.
Non technical. They're all dock.

Video: Container Interactive Mode

Try to represent them in terms of contains.
Instead we just went with some hello world application,
which uh, gave us a confirmation
that your setup was working properly.
And also you got to know
how the docker architecture also works
and all the client, all the components are also very clear.
So today what we'll do, we'll start that machine
And then slowly we'll work towards
how containers can be created in terms
of application point of step by step.
We'll go ahead.
So here what I do, I'll open two different sessions to work
and get the concept of Docker in a bit easier way.
Connect with two sessions in one session, you'll work
with your host in another session.
We can continue with the containers.
Okay, so this is one session
or if not what not to get confused whether I'll create
two different windows itself.
This is one window.

I'm also working with another way, right?
Same CD downloads. CD downloads.
I'm gonna connect with two sessions,
same server, two session.
Yeah, same server two times.
It is the same server. I opened two different
sessions already.
I showed you in the last session here I'll show you the
confirmation docker image list.
In the last session we see we had this hello world image,
which we word and also there was a couple
of containers we created from this images.
How can I check them? Its docker
container list.
It'll not show anything because list will only
show you running containers.
If you want to see all the containers, dash a, so when I go
with dash A, it goes with all the containers.
You can see it. This is for first container
second back now.
Now this is what Simple. Hello?
Now I want to actually show you the difference
between working with a container and the host system.
Host means what? Your regular server
container is a different entity.
Correct? And you guys remember I said uh,
open operating system was around close to 1.7 gb.
Like it isn't it? When we started in initial part,
which is our version, Ubuntu 22,
I went to Docker registering, I'm gonna search again
for a new image
like how you work with Ubuntu on system.
Now I want to work with Ubuntu on containers.
I selected that Google image.

Uh, now once I scroll down,

you see here it says something called like tag.

You remember GitHub tax versions

one.one, two.one, one,

which the same thing here also same concept.

Docker tag we call, which are nothing

but what versions.

O2 20 O2, 22

O2 24 got versions we use 22.0 for for our system.

Agree or not, I want to do the same thing here.

Dock container,

run over.

If I do like this, basically what it is,

it'll go ahead and start creating the container.

Uh, what you say by using something called like default, uh,

what do you call tag?

A default tag is nothing

but it's represented

as latest like main branch.

You have latest tag we call it.

If I don't give tag, it'll take the latest tag.

So I want to run a container with some specific tag.

Then you need to say colon and the tag.

I'm saying run the container using Ubuntu 20 two.zero.

Let's see what happens when I run it.

This was syntax we used in the last session.

Docker container and hello word.

Now I'm saying docker container and open door.

Open image is not there. No. So it does the same process.

Unable to find going and downloading. Done.

Now I'll check you

got a wound image list again,

the container is not shown hyphen,

now you can say 15 seconds.

Image of 1 22 0.04.
Some id, some random name is generated
and you see created 15 seconds ago
and then it immediately exited.
I'm starting, I'm trying to create a container.
It's starting but immediately
after one second it's getting exited,
meaning basically it is getting stored while it's happening.
Containers have something called modes,
running modes, run mode.
We call it interactive mode, detached to mode server.
I'll understand if you don't specify
mode, it'll always exit.
Now I want to say I want
to run a container in the mode plus mode.
I'll show you, which is called interactive mode.
If you want to run a container in interactive syntax chain
container run and also earlier Yona
containers are getting some random names.
These are names, right?
Or if you're getting confused now I can little bit stretch
this Tom so that you can understand what machines.
That's the name of the container
which is randomly coming right now.
I want to name the container by myself. You can name it.
I'll call this container as C one container one
hyphen ID interactive.
That's what we are saying Ari.
So I'm saying run a container with a name called seven,
run it in the interactive mode.
Interactive mode is also called foreground mode.
Foreground background two modes.
Interactive means foreground. Detach means background.
So now I'm running the container in work mode,

interactive mode, which means basically foreground mode.
When I execute this particular,
and let's see what happens this time,
see earlier I'm on my Ubuntu system now.
Now I'm no longer in Ubuntu.
You see it is saying what route eight sum ID zero.
So 0 5 9, 0 4. I'll come back to the other session.
Now say I did not give hyphen eight, meaning what it is.
Running our hyphen A.
Also if you want to give status is what? This three R exit.
So like this. If you want to run the container,
you need to specify the mode.
If you specify interactive mode, you see
it'll directly take you inside the container.
This is actually a container of type open, how I can verify
2 0 4, agree
or not the same thing what you ran on the host.
That's exactly what's happening inside the container.
I hope you got the logic is what I'm trying to say.
Yes, in the interactive,
if you run a container in the interactive mode, you will be
inside the container and now I'll distinguish.
This is the host system.
This is host, this is container,
but both are giving the same output.
You think this is also
to is also, but if you see the size
towards the size of Ubuntu here, like close
to a MB, let's say,
but your original system is taking 1.7
gb, something like that, right?
I said one statement. Docker containers are lightweight.
That's where they're faster.
This is the point, but you might say,

what is difference then from a regular
full fledged guest operating system to this one?
What is the difference? Now let's try to understand.
You wrote this command, I showed you
this in your Linux sessions.
Initial what this command does
to check the process correct?
To check the processes in your system.
What are all the processes that are running?
We go ahead and relay with this.
There are so many processes running.
See your container process, bash, process.
S, s, H, process, uh, S, SHD, Docker, D, crs,
CR service S, SM service, Python three.
Lot of service are running because it's a
full-fledged machine, right?
That's why it took 1.7 G of same movement,
22.04.
Now the same thing. I'll come back where
inside the container
I execute the same command.
Where inside container and how many processes are there?
Only two. One is bash the shell.
Second one is the PS command, which I execute.
So inside the containers you don't have
everything like a full-fledged system.
Am I clear case? What I'm trying to say?
There's a major difference in terms of working with
what you say containers and working with your host systems.
I can also verify like this, this is a full system.
Do I have SSH full load?
Mm. Do I have SS command?
Yes. Do I have GI command?
Everything is there now I'll jump here.

The very bare system. That's the idea, right?
What was the idea? Don't keep everything, keep only
what is required for your microservice.
These things kept in the perception of working
with monolithic.
This thing is kept in the perception of working
with Microsoft.
Now you got the points. It'll have bare bot,
bare things like what if you want app, it'll be there
to set up and go ahead what you want on this bare system.
Clear what I'm trying to say from online everywhere.
So we do have a container.
It does have the things, but not like a full-fledged system.
You take this minimal systems and start working. Yes, yes.
If I want to restart the container
or I will do, we'll learn like there are commands.
Yes, we'll see like all those things. Yeah.
Did you understand the concept right now,
the differentiation, what I'm trying to say is
how containers are different from traditional way we used
to do the work is something I'm trying to explain, right?
Okay, now let's go ahead.
Now check the status of the containers right now.
Started, sorry, created six minutes ago.
Status is still up seven minute.
Now I'm inside the container.
Now I'm saying exit where
inside a container which was started in the
interactive exited.
That is what interactive modes.
When you run a container in the interactive mode name
itself, you say till you are connected with
that container interactively
and doing some actions, it'll be alive.

Once you disconnect from it, it's gone.

That is called interactive.

Interactive mode is helpful when you initially start,

you don't have much awareness on how containers work.

You try the things in interactive.

Video: Container Detached Mode

Isn't it?

Assume you have a login application service running,

you're out of the container,

your login app service stopped, it's not correct, right?

You want that service to always run.

Always run where in the background

that more is called detached.

Clear is. So you need

to start the container in the detached mode,

not the interactive, interactive mode.

Also, you can start for what?

Troubleshooting for this purpose you initially go through

or if something you're working with does not work.

I'll show you more examples in the future,

but actual perspective

or in the real time perspective,

you will always run the containers in the background mode

or also it is called as detached.

Let's try to create a container. The detached mode.

Now same.

There is no difference. You said it number for interactive.

Here you'll say dt. That's it. Nothing much.

This is how I created the first container.

You cannot see go with C two C one

because already C one is there.

So I'm creating C two

and I said what that dash dt,

which means basically detached.

Now you see what happens when run a container detached mode.

It's not interactive know,

so it is running in the background.

In interactive mode,

you'll directly get into interactive shell of the container.

As you said dt, it'll not be interactive.

It is always running in the bad wall.

How I can verify, I will come back to this shell.

Now you understood why I'm going with two shells

so I can easily go ahead and verify like this.

It'll be always up and running

unless you explicitly stop this particular country.

Are you clear? Guess what I'm trying to say?

So another container is running but how do I connect

or how do I communicate with the container is the thing Now

agree or not because here both are what?

In the host. Now you got separate syntax.

So interactive mode there is not much in new, nothing

to do directly or inside the container run the commands.

But when you go with the container in the background mode

or detached mode, you got a special command.

That's what means container execute meaning on the container

you can execute which container.

CO. If I say C one

saying you cannot

execute on a stop the container.

You can only execute on which containers

Running container it is running.

This is not coming from my system,

it is coming from container.

How I can verify another ways

is the same command you executed earlier
in the interactive mode.
Now I'm able to get the same outcome from the container
which is also running in the detached.
Now this container will always run
because you started in the background.
Uh, if you don't want it, you can shut it down.
Yes, there are different container command.
Second I'm gonna see now got the clarity,
interactive mode, detached mode.
But most of the time you always work with detached mode,
but it does not mean you'll
never work with interactive mode.
Interactive also use especially for troubleshooting.
And initially first time when you learn
or try something, you might go
and do it in the interactive mode.
I'll show you like later on.
Other scenarios also within private.
Mm-hmm That's the ax.
Like you can check docker commandment, sorry,
like they have given you different things.
Now, same way now we are clear,
our container is still running.
As you can see, uh, there is a container that is running
and there are few containers which are stopped.
You can do container management,
docker container remove.
How do you remove a VM? Same.
Either you can use name acceptable
or you can also use IDs.
Here is what I'm trying to say.
You see what it said?
It remove this two,

but you got an error response when trying to remove
what C two and what it is saying.
So running containers cannot be deleted.
You need to stop or you need to forcefully delete.
Uh, now I want to stop this container.
C two is running. I wanna stop it.
What happened? Stop.
Now it's not mandatory
that every time you create a new ER
you can start the whole content.
It is stop. Now if I delete it'll be gone,
but I don't want to delete.
I want to start it back Once again.
Here is so meaning what?
It's, isn't it like just like VMs I can create,
I can stop, I can start again.
Just literally like your VMs,
but what small sized s
And you see it has an operating system.
It is also having files, folders, which I did not show you.
Maybe I can show you here.
Onset list me rule.
Just like how you used to see in your typical host system.
I'm seeing the same thing even
inside there containers as well.
Now Claus in terms of how the things are working now.
Now same way every container also has a lot
of other properties as well.
But I said eventually you'll slowly get to know about them.
I have this particular container running now I want
to know the characteristics of this content.
Docker container.
Inspect is the command use.
Inspect the container.

You want to work with which container.

Okay, that's the container.

I want to go ahead and understand the properties technically

like what this container is about, how it works.

That part I'm talking about

INS inspect

container ID need.

Both are acceptable.

It has given me lots of information.

A container has an IP address like

how your system has an IP address.

A container also has an IP address.

A container also has a network.

See network settings. Uh, they'll also have, like I said,

images volumes, right?

Then uh, you can see mounds.

Then uh, you can see the options of uh, name

which platform, uh,

when it was created, what's the status

properties of the container.

You can verify like how you have instance details.

Select id,

IP address, right?

Uh, then network.

If I just talk about CPUs like that same

how you have instance details.

These are container details.

Later on you'll see like other things,

water volumes, why we use other things.

Also with the help of this inspect command

and you'll figure it out clear, right?

I hope you got the point. Is uh, now technically what?

Later on this will be something you might do.

I do. I need to close.

Let it be

This container is what?

C two right? C two.

Now what I'll do docker container.

The C two

with,

I renamed it.

I call it as alza

as I was seeing it.

Docker container

I'm creating.

You got an idea.

This is how we will be working

and I said one definition.

Docker containers are isolated environments.

Each container gets its own environment.

Clear what I'm trying to say.

So I

said inspect C two, sorry, C two.

We remote it now. LMS.

Mm-hmm Here, uh,

you can see there is something called like IP address.

Filter

filter, IP address

clear is what I'm trying to say.

It is giving me the IP address of that particular system.

Uh, I would say exact word

check for other apps.

It has its own IP address,

like how you have individual VMs

with individual IP addresses.

The four instances set up we did

for the production lab, dev lab

for instances four I three

containers, three different types.
That's why they said containers are isolated environments.
Now here it's so each container has its own environment.
In that environment your respective applications will run
and how VMs can communicate with each other.
Containers can also communicate with each other.
They have their own networking also.
And how for your systems you can create your own
networks for containers.
Also, you can create your own networks.
Literally, that's what I'm saying.
Whatever you did in your big scale systems,
everything is shrink down but you get the same out
because of this technology called containerization.
And you see how fast I was able to create in matter
of seconds everything was getting created
for creating a host, uh, installing some os.
It's gonna take a lot of time,
but here, instantly, within matter of minutes
and seconds, I was able to go out and get everything done.
That was the idea to even begin with this technology.
The hospital now, like what's the definitions we discussed?
Now it all makes sense.

Video: Container EXEC

Execute container
name command.
This command will be executed in the background mode
so it just gets executed under.
But that's not the case.
I want to get inside the container
and I want to see what's happening.

I want to clear on which

was possible in the interactive mode.

Now, once you start creating a container in the background

mode, you can interactively connect with it,

but it will not stop the container.

Even if you exit, you

start the container background mode.

Work in the interactive mode is what I'm saying.

How it can be done.

Execute

interactive interactively.

Where

on this?

Okay, execute interactively on this particular

container with what?

How can you execute something?

What you need to execute

something A

There's the purpose of bash.

What was bash a shell It send interface

between the system and you it.

That's what we are trying. When I say this

command, let's see what happens.

Now see the

ID three B seven one.

Now I'm inside this particular contain container.

Play around whatever you want.

I want get here. GI is not there. No,

this commands you're already familiar with.

Let's, but it's not like a host.

We call it as a process technical.

Every container behind the scenes, it's a Linux process.

No, I cannot because

No, you'll, that's why when I try

to go with that, right, it'll not work.

The reason because I said it works on a full-fledged

operating system here,

it's not a full fledged operating system.

Got it? Mm-hmm. So now I went and in store where?

On the LMS app container.

But the same thing

here.

I hope you understood the point

because you did that work.

Where? In LMS. So only there the reflection.

That's what I was saying. It's just like a,

an isolated environment.

You've got address, networking commands,

everything you can go out and work.

Uh, now an interesting thing,

so I have this three containers, isn't it?

And you see I started this in,

I started it in the background mode,

worked in interactive mode.

I exit and see what happens with the container.

Nothing, it's still in the background.

That's why we start in the background mode

and do the work in interactive mode.

That is the general way of working.

Now you want to troubleshoot anything, right?

Just go to the interactive mode

and then do the work Now, now I have this three containers.

End of the day there's three containers

where they're getting this processing.

Obviously they're doing the work now they're downloading,

they can calculate something.

When I have some program, it can take an input request.

So obviously they need some kind of allocation.

That's a command. You can see,

uh, if not I'll run here.

These two are quite memory usage is very less.

9, 9, 8 kilobytes.

Like literally one and B out of 3.8

because we have four gigabytes of this thing.

You see 15 reason

because we started installing some softwares

and see network io

earlier our was how much?

Uh, some a m means, right?

One second. Yes.

Okay, now you see here it says right,

like 1, 9, 5, 1 9, 5 ments.

I did the update. I downloaded the gate.

I was doing all the operations. Now it is taking it.

And containers are, like I said,

now it's quite different from what we used for.

These are basically called US

volatile measures, just like ramp.

This is also like ram. Generally. DISC is what?

Permanent storage. No, according to the knowledge.

We know know the difference, right?

Like when I'm talking about ram

and hard disk, anything in the RAM is volatile.

Once you restart the system, everything is wiped over there.

It's not the same with disc.

Hard disk always be there

and all this data you stored will be there.

Now it's not the same with contains.

I did GI and everything where on this LMS app,

I took the disc of the storage

and everything we are seeing here.

So let's exit control C.

You can see to exit from here, I'll type control C,
cancel all the context gone ly.
What I do,
uh, ments forcefully, you can remove a container.
If it is running again, I'm not doing stop
and remove two commands.
This thing,
remove it.
Do I have it
gone?
I'm creating what again, again, MS app I'm creating.
Same thing. I deleted. I'm recreating.
Is it running? I'll get back again.
Check the stats
again.
Everything is back to scratch. Zero.
No memory, like 15 MB earlier. Nothing.
I do control
CLMS
app.
I'm going back.
That's how it works. So
not like traditional systems.
There are differences. Little bit, same
few things, little bit different.
You need to understand those things
and then you need to start working towards these
applications.

Video: Container Nginx Setup

So I'll do what we did, what we did in technical terms.
If you guys remember, we launched when Amazon server, uh,

uh, I would say Azure server.

Then you went

and you installed something like an EngineX web server

by using app command and all of it.

Then you used GI to get the code

and then you deployed it into a location called document,

which was like via www htm and you went with that.

Now that's the same thing I want to do,

but in terms of containers, let's try to see what is that,

which the thing that we learned in the previous sessions,

the what you say, our initial class verified right to do,

what would be the problem

and how we try to do the things in terms of Docker will try

to understand because the traditional approach is not gonna

work in this particular place.

Docker works a little bit different. So here

I dt, meaning start in the background mode.

Name of the container I call it,

and the images UB 2 22 odd zero

four are here.

I'm creating a container called web using Ubuntu.

Got. Now we know a container will be running

uh, by seven seconds.

Now I want to know whether

a web server like nGenx is present or not.

I want to go ahead and figure it out here. I agree or not.

So I'll do the same. You remember in the last

session I have given you this comment.

This

will help you get inside container isn't that was

what the purpose of the command was.

No, I'm inside this particular, uh, container as expected.

Now the thing is I want to see if this engine accessory

or if not directly

see what it is saying.

Nothing. No, There

is no SS command, so I'll not be able to figure it first.

What? In order to verify I need some tools.

App update the system.

The application name

is IP route two.

This will give you SS command, use

SS found in this package.

Nothing. Not even SSH. Nothing is running.

It's completely empty as,

but there's the Bash

and the PS command, which I executed.

Anything related to nGenx, nothing.

So now what if I want to run the web application over here,

I need web server typically.

If not, I'll never be able to access. Okay?

This is one way or if not, I said no,

you can take the IP address,

confirm nothing is running.

What I'll do as usual,

isn't it the same thing that you guys also did in the uh,

previous sessions You went and you install INE X,

you install it

Neither.

I'm seeing this running on port 80 80,

nor I can see anything running with respect

to engine X process, right?

Yes. Framework, we are inside the container, correct?

Right. Okay. Reason because in the normal systems we have

system process, system day, which will automatically start

when things are installed.

But here it's not a fulfilled system, so it does not help.

So you need to explicitly hold and run it.

How means there's a command service.

Now it says starting engine X service.

Now it is functional.

So let's check the PS command also engineer.

Next process is also working correct? Right now.

Next is function. Pretty much. Sure.

Oh, I will go here, I'll refresh

install, but it's not working.

Okay, fine. Uh, we know all the things are working now,

so I'll do exit from here.

You remember we had this particular command.

Inspect the code web. Let's see what it says.

We got some network information.

We got an IP address.

See, this IP address is a private 1 1 72 17.

It's a private range. So if I take a private range

and access from internet, there is no use.

They will never work. Private ips are not meant

to resolve over public addresses or public network.

So there is no use. What I do for confirmation,

local browser, girl, I'm saying the ip,

this IP addresses off container.

It

is working, but where it is working, uh, inside the system,

but I'm not able to access outside.

Are you clear? Is what I'm trying to say.

So basically Docker, by default, it will only

keep the network within the system, not outside the system.

How? Because there there's a concept called docker

networking, which we need to understand.

Okay, clear. What I'm trying to say.

Concept was clear revenue.

So you can connect with the previous sessions.

We took one sum open two, then we installed nGenx, then we went and uh, started it.

It's functioning, but what we only inside the system

but not outside the system.

So this is one way of creating the container.

I'll tell you the problem statement.

With this approach, we have this entire setup in which

container web er, isn't it?

This was the container response.

Okay,

go on, right?

I'm recreating it.

We have the new container.

I'm inspecting the container. Do I have the IPA again?

Right?

So everything I did was gone.

So every time I create the container, I need

to redo all the work without the problem.

So if I want next time onwards the same thing to run,

then I need to make sure

this container always has engineer X.

Uh, that's where you start working with the actual images.

If I go with Ubuntu image,

Ubuntu will never contain engineers.

You always need to set it up

but instead see my work approach.

Now

I'm going back to my Docker registry.

I don't want Ubuntu.

Now my requirement is not open to my requirement is what?

A website.

Okay, so I'm taking word directly, Ingen x immediate itself.

Let's give it a try.

This of no use, no, I mean like it can be useful,

but again, I need to do the work.

Once again,

a different name.

I'm given

this time I changed the image from Ubuntu two.

Ingenix. Ingenix is not there.

So it's going and downloading the image for the first time.

And we have two containers, WEBC, er,

and as well as the old one, previous one.

This one is one with open two

and this one is created with HX.

Did I need to install Engine X?

Once again, no nGenx image already

has nGenx soft.

When I create the container using that engine X image,

a running instance of the nGenx cable.

Next time also if I,

I remove,

maybe I'll remove that old one.

Also, we have only this

container,

some IP address,

which was not possible when I was doing with

and I understood how to work with the containers.

I can rely on Ubuntu and I can get the things done,

but it is not so sufficient every time I need

to set up the things manually by myself.

But now what nGenx image already has this software, we,

so I just said run the container.

I got this clear everyone.

So first problem statement is fixed.

Next time onwards, I don't need

to set up once again web server from the scratch.

I can directly go ahead and rely on that image. I'll get it.

Video: Docker Networking Basics

So technically

this is what is actually happening.

This is your system.

This network is called Host Network.

This is host Docker

host in the Docker host.

It is getting the network using host network,

which is technically

AWS means VPC.

Azure means Vnet.

I think I discussed the same thing in the

initial sessions, isn't it?

A network layer is called VPCR vnet

and this is connected to internet

and your server is part of this network.

So it got access clear. This was what happened.

Now, as soon as you make this mission as Docker host

in your system there will be three

different networks created.

Okay? Three different networks. One is called Host Network.

The next one is called Bridge Network.

Another one is called a non network, no network.

Okay? These are three different network drivers you can,

which you get when you install Docker.

Now by default,

whenever you create any container, it is going to be part

of something called as Bridge Network.

Name itself is saying Bridge.

This is what Docker network, which is typed bridge.

Any container you create out of the box will be part

of Bridge Network.

The web container I created now that is part

of Bridge Network.
This Bridge network has access only till the host
by default
are clear, is within the host, not outside this
a technically what I'm trying to say, meaning I'm the user,
I am requesting the container, I'm getting the response.
This is okay, I am here.
I'm requesting like this.
That will not work. Clear everything what I'm trying to say.
Hmm. So now internal I can access,
but external I was not able to access.
The goal is what I want.
External lasts because I want my application
to be seen by users.
Now in that regard, how we work, we'll try to figure it
where you can see all this information
means, let me go ahead.
Context clear every, so whatever the networks
that you create, by default, they're only within the host
but not accessible outside the host.
Let me clear this off.
Currently we have containers.
Now what I know will help you understand in a better way,
I'll also get rid of this container.
Now, currently no containers,
you can go ahead and verify the list of networks
that you have inside your system with the help
of this command,
okay?
Those are the networks, bridge, network, host,
network, none.
Network, default is Bridge, then Host, then none.
None means no network host also, I'll show you an example
how to use Now.

Uh, you can go ahead and check

Docker Network.

Inspect which network

earlier.

Yes, we install in the next

Check.

So right now I'm saying curl no route

because there is no content, right?

Yeah. Now as I was saying, Docker network

inspect which network, either the name is fine or IDs.

Also

See why your containers are getting the IP addresses in

one 70 to 70.

Because of this range, it's already pretty different.

Here you see containers are empty

because we have no containers as of now empty.

Same with post here.

Also nothing.

Now I can see it, no containers.

If containers are there it'll show you. Same way, none.

Obviously none. We did not create any containers.

So no containers.

Now I'll go ahead and create one more container like

how I did in the last part.

I'll not give any name like big thing I'll call it

as web one, okay?

I'm creating a container called web one using the image

called US Internets.

Now let's see when I read how it is gonna work it up,

we have one container coming up.

So uh,

web one, see the network part,

it was same id, if not, I can confirm it for you.

So now this container is coming in which network?

This network, I can also go ahead and do like this one.
Inspect bridge network. Every, there were no containers,
it was like empty braces.
So By default, any container you create,
it is part of Bridge Network.
So Bridge Network has character name it, lip Sync Bridge.
So it can actually connect your system, meaning your host
to the external world and as well as Connect.
That's why we call it a Bridge tech.
But can I access them right now? No.
The reason because to enable that option, we need to go
with some changes in the command, some additional options
and all these things we need to give, which we didn't give.
So it is working only intranet meaning
only within the system.
So that's why still if I take this IP address,
it works as I did noted in the system earlier,
within the system it is working
but outside the system it's never gonna work.
Uh, now I can also do like this.
This is what Bridge Network.
I said there is one thing called network
like along with Bridge Network.
There is also host network. Post network means you're saying
I don't want to use a Docker network, even I want
to use the network of the system itself.
System network is connected to internet. Agree or not?
Are you here? What I'm trying to say?
See now internet X is working right?
Technically agree inside a country. Now what I know,
but can I say code 80 here?
No. Why not? Because your container
is not in this network.
Your container is in the Bridge Network.

Are you clear? Now I'm saying I wanna create a container
which should use host network so
that I can access host system.
Are you good? See, right now
in this network I want to create the system.
Uh, the system container.
I'm creating a container part web two.
If I create web two also, it'll be part of each network.
If I don't define this network, if you want, again,
I can confirm
Web one Web, isn't it Web one,
container web two.
Both are part of Bridge Network.
But I don't want because Bridge Network will
isolate only inside the system.
This is the network. I want
to use system and see the difference.
And once again for your confirmation,
nothing on the host I
I'm creating Web3 container,
but I don't want it in the Bridge network.
This is the command in which
network you want to launch.
We have host network. I'm saying launch this
container called Web3.
Inside the host. Meaning what? Technical area.
I showed you that big network.
Now you are launching the container
in the network technically.
Alright, clear. Let's run it.
Created it.
Can you see Web3? Because we did not
create it in the Bridge Network.
Inspect Web3 container.

What is the network? And you see there is no id.

Why not thing

once it's not giving any idea.

But for both Web one, web two I got I

but for Web3 I didn't get reason

because was I said in which network it is

coming post.

Uh, exactly. Post address itself is generate

earlier.

Did it work?

I did not install EngineX in my system.

Is it if X is installed,

do I see Ionix?

Neither documentary is there, neither Nix software is there.

But see is interesting.

What 80. But I never install Ionix out

via Docker Host network.

Uh, what is say Driver we are routing the traffic

is one 5%.

You have the container, your output,

everything is coming from container route

but it is coming from Post.

But it's like what you did.

But in terms of docket, this is how it works.

Uh, this is good but only up to

minimal part only we can use Force Network.

Now it is running now good things are working fine for us.

I was able to access that, right?

This is Web3 container. Okay, let's rename

elements.

Now I want to run login application.

Same using Inex.

Any difference from what we wrote earlier?

No, just I change the name

and same I'm saying the network should be post network on.

Is it there? No.

What happened? Check the status.

It it worked for my LMS

but once I started going with second time,

it's not working isn't it?

But did it happen with my web

and web two when I'm going with Bridge Network,

I did not find any difficulty

when I'm going with Bridge Network.

But in post network, once I started

creating it is not function.

Uh, now under try

to understand like why it might have failed.

Try to run it failed and let it work.

Give a thought why it might have

failed thing once.

Mm-hmm.

Only one can use it at a time. On what context?

Yes, that is correct answer. But on what context?

One at a time. What you can use.

One close.

Okay, if not it failed right now

you want to know why it fail.

First thing I said always the better option is to go with

if something new, something you're not able to figure out.

Always use this technique. What I already thought

the e-commerce er I want to create.

That's very interactive itself.

You don't know why it's failing, right?

I want to see it in the interactive.

Then you might get some information why it is failing

already saying addresses already in use.

What is the meaning that port 80,

what you were using for web server?

It says it's already occupied.

The rule is like this, A port can only be used

by one service at a.

If SSH is already allocated to 22,

I cannot allocate 22 for something else.

I already allocated 84 wat

web server called uh, what was that name?

LM ser.

Now I cannot go ahead

and again use the same port for this particular file.

Only one port can be used

by one service data already X is there in that one, right?

It occupied. So you cannot create

other containers which goes with the same code.

If I delete that one, this will work.

But wouldn't it be a disadvantage

because what was the idea we discussed when we were working

with Docker

one image multiple apps,

but with host network I can only run one.

So host network is fine if you want to run only one service

specific to that particular port at a time,

but that's not how the things work.

I said I can go ahead and host multiple services now in

that particular regard, I cannot go ahead

and rely on the host network.

Host network is convenient for running one thing,

but once I scale to multiple applications, then in

that scenario it's not going to be a feasible option for

understood the problem statement.

So that's why.

Video: Docker None Network

Test container.

Dash, dash network equal

to none.

Same engine example,

running none.

Network also same way.

So bridge network and as well as none network.

Yes, I'm able to create multiple containers

and in the non network also,

if you check the characteristics,

no IP address, nothing and it says none.

Uh, difference here is quite simple

from e-commerce container login container is there.

Inspect login card. I got some IP

execute on

E-commerce container.

Curl this ip.

Did you understand what I did?

I had two containers from one container.

I was talking to another container

that 1 72 7 0 threes of which address?

Login. And I'm executing

In e-comm.

So two different containers. Are they communicating?

Yes, same thing.

Now tell me

I want to talk to this guy.

How?

Not possible. Simply that's what non network is.

If you want containers never to communicate with each other.

The network mode we go is none.

Bridge network by default and communicate.

That's why we call it as bridge network
are clear is none.
Network has no connectivity. Right?
And also maybe I can show you this example.
I'm here. Agree or not In your next one, this is,
what are you saying?
Fail. Fail. Fail. It didn't update.
I'm sorry, nix is already installed now. So let's say GI
I cannot, if I want to install, I need to update the system.
But is it telling me,
because you only said no network.
None. That's what, not network. No access to network at all.
If you never want a container to communicate
from internet are within the network.
We go with this. Now you might ask
why we need this kind of things.
One simple. Maybe you just want to calculate, you want
to do some build in that particular scenario.
For temporary work, you can use this particular candidates.
No network, nothing. Just you want
to test something quickly.
Go ahead, do it.

Video: Docker Port Mapping - Expose

Today we'll discuss about a concept called port mapping
and port exposing.
If you understand this particular concept,
then you can go ahead and figure out how to work
with, uh, what is it?
Uh, containers in the bridge network.
But still I'll be able to access this.
Okay, so it says

port mapping and exposing crucial concepts.

Insert docket, networking, exposing both.

I know definition wise they look quite lengthy,

but I try to simplify it first.

Explain, then we'll read this,

then it'll be much easier for you to understand.

So there are two things now exposing both

mapping two different concepts.

So Azure Douglas, we have a system here.

Docker post in this, you're creating a container.

When you're creating a container.

Container will be created from an image, agree

or without image.

Can I create a container? No image is a template.

That point is also clear. Everything agree.

So if you are creating a container,

it comes from an image, isn't it?

That is for sure. Now I'm saying

based upon the image,

they will expose the port

If I'm going with an image called as n engine X,

meaning technically this is EngineX.

Now on which app, on which port this application runs

generally EngineX works on which?

Which port or 80, right? That is called exposed port.

Hmm. If I'm going

and working with something like Postgres

Post works on which foot?

This is exposed quote.

Exposed quote means you are telling the person who is going

to use this image.

Your application will be made available on this spot.

Where it will be present means inside the image

there is source code, which we call it as a docker file.

Docker image is created using something
called like docker file.
Actually, uh, in the docker file you have instruction.
What instruction means? Expose That.
Expose is responsible for
making your application run on that specific portal.
Is it clear everyone what I'm trying to say? So technically
here you see what it is saying.
Docker file links for docker file. I'll open one of them.
This docker file is source code for your image.
If I scroll down,
this is the reason why your
container is working on port eight.
Same bit.
Got the logic base. Every now
we understood exposed code.
Now this tells that a person who wants to use this image,
this is where application is available inside,
but someone should send the input.
No. Can he directly come communicate with containers?
Not possible because we already discussed
your network is isolated in terms
of docker from your host network, correct?
But you already know host network
I can use, but what's the problem?
Only one. So now on the system level,
we need to open some port
and why are that I need to send the request.
Are you clear port mapping? I'm saying this thing.
Where am I keeping on the system level?
Some port, let's say
on the system level, I'm opening
port mapping is a technique in which
you are allocating one port on the system

to redirect the traffic to a specific port on a cutting.
If this guy hits this,
you're saying send it to this contain.
If this guy hits the traffic on port 80 80,
you're telling hit this.
So mapping is nothing
but taking the exposed port inside a container,
making it available to the outside world
by using a PO on the system on your host
clear technique, everyone.
This is what we are trying to do,
but according to the docker
definition and all of it, right?
Looks quite clumsy, but try, try to read this ones both.
Now you should able to understand with the concept
that I explained clear,
I hope understood the logic is what I'm trying to say.
Uh, now I'll demonstrate, we'll go ahead.
We'll try to create a container without port mapping.
With port mapping using the predefined, with port mapping,
using the user define, you'll get the clarity.
The uh, Ravi, uh, the upper one,
hyen p uh, that's defined.
Uh, that's a by by default value or
Yeah, by default value it is defined already.
Yes. We don't need to do anything.
It just comes up in the system.
Okay? The second one is, uh, the customized, I think
the the range which is defined we can use from,
Yeah, based upon your choice you
can go ahead and update it.
It can be from 3000 to 5,000.
I selected from 80 80 to 90 90.
Okay. Okay.

Okay. Now let's see guys,

how I can go ahead by matching all of it.

Currently we don't have any container

small.

I take W one web one like that.

Same using which image engine X

running you see on port 80, this is internal.

You cannot access it externally. Is it possible

what I said dash capital B.

Now what it does, I said now range starts from 30 to 7 68.

From there it'll pick it up

per your idea.

I'll say that's my P of the system.

This will not work anywhere.

We are pretty much sure about it

because I already added this firewall.

Nothing is coming.

The only difference from this one

to this one is I added only capital P here

from outside world.

When you request on 30 to 7 68,

we'll send it to 80.

On this T,

this is how externally you can access.

So we map

the code 30 to 7 68 to the exposed code on the container.

Eight.

Uh, now this is what predefined.

So if you just keep on going,

it'll keep on incrementing the port

7 68, 7 69.

It just goes like, now can I run multiple apps?

No issues. Each port one specific app,

like in your last session you said like I want

to run three different applications.
Now it's possible I in each
container you host what you wants. Yes,
One single link page
Your choice, it's up to same multiple you can define.
So same way this is predefined. No, I want to go with my own
small P.
When I send the traffic on 80 80, send it to 80
post code container code.
So are we only one container for one port?
Yes. See now I can
request all these different applications.
One app, another app,
but other things will not work
because you never port map that thing like this.
You can take your own system ports, route the traffic
to the container, wherever
what the idea is, how the things will work.
This is how Bridge Network can help you go ahead
and set up multiple apps and start working with the things.
Claire, is your problem solve what we had earlier?
So that's why I said bridge is what we go ahead
and relay when we are going and working with the containers.
Don't worry you learned it, but in the future it'll be
replaced in the KU and it is later on.
But for timing, yes we need that network to go out
and access our applications.
Clear Everyone Fine. I hope you understood.
Okay, this is default bridge network, which is good,
but default bridge network.
If you want to communicate in this way, it'll work out
from external in intranet.
We are going and discussing. Now, uh, in that scenario,
by default it only supports

IP based communication,

but we need name based communication,

not now in the Kubernetes.

Very required actually technically speaking,

but further now you need to understand the concept

again in Kubernetes.

I cannot come back to all these sessions back

and do this work.

So I'm explaining now what is it, how it works.

The same concept in the Kubernetes. It will help.

We are talking about name based communication.

Name based is nothing but domain based.

You have some name rather than relying on an IP address.

We are using names to communicate

for going with that name based communication.

You need to work with some additional things.

By default, the default bridge network you

get, it'll not support.

That's why we create our own custom

networks in that networks.

We launch our containers

and then we can do the communication using the

names clear everywhere.

I hope you got the logic is what I'm trying to say.

I'll demonstrate. I'll just go ahead.

But this concept clear external access everywhere.

Still any confusion. So you got two choices.

Either you can go with that.

Would you say custom and whatever you want are predefined.

Generally custom is what you prefer

because whatever the port you want, you'll add

and you can just go with it, right?

Okay. Okay. I'll keep the same syntax

as well in the document what we just discussed right now.

And yes, I'm gonna get rid of all these things.

Let me write the same taxes for all the three.

No port mapping. Port mapping

with prefin port mapping with user here.

Just check the same tax. This is what we discussed.

I already said it's a network which is automatically created

and if you don't specify any network by default,

containers will launch in this network.

Containers on the same bridge network.

They can communicate with each other using IP addresses

or each container gets an IP address from

that private IP range defined by the docker,

which we already saw

for containers on the default bridge network.

DNS resolution is not available

or will not work out of the box.

Containers can resolve each other.

Oh sorry, cannot resolve each other by names.

This is something will be used in Kubernetes.

Its name is called as cluster IP service in the Kubernetes.

So right now we need it. We are checking out.

But just remember this point is very,

very help important in terms of Kubernetes later on

to enable name based communications

because in the future IP address cannot be

predicted dynamic.

From time to time they change. We can never rely on I.

So that's why we rely on names.

So further that you need to understand

how name based communications work.

So further we'll go ahead and check out the things.

Pretty simple. I'll show you.

How am I going to help you understand the concept?

We have no containers right now, correct?

Let's say something like C I'm grading A called C one.

Same using

XX is both

grading.

One more container called C two using open tool.

Well both CC two running.

Let's say of word C one

I command,

I'm looking for it's not there, fine,

I can quickly install it.

Look,

is IP of NXC

Are we able to communicate?

So I hope you got the point.

Using Ubuntu container, I'm communicating

with nGenx container using ip.

What is this? C one

name of the container.

Which container engine X container

cannot.

It does not know names. It does not communicate with names.

I want enable it because this is the concept

that you need in the future.

What I'm saying, I want the communication

to be done via what names?

Not via ips ip.

It's already there out of the box but names, it's not there.

So I want to make it happen.

Now in that particular regard, default bridge network,

whatever you get in the system, it does not deployment.

When you create your own custom networks, they support

name based ation.

Good. Know what I'm trying to say.

Let's see, Alex, it

and for your information I'll also, so this way,
C one
DS names, no, it does not support technical.
Not only this, any container.
Any container you launch in the default
based network, it does not.
So, okay,
what I said, now I want to go ahead and enable this network.
Now then in that scenario, what you need to do, you need
to work with Custom bridge networks.
Let me show you.

Video: Docker Custom Network

Good.
So what you don't get, you'll be able to get the things
with custom networks.
Okay? So here and also whatever the command I went with.
Now I'll show you guys the same commands.
I'll write for your reference here anyway.
See I said curl, see one fail. But with ip, does it worked?
Now we are going with the custom bridge networks.
Not pretty simple, nothing.
But end of the day we are creating our own
network and getting the things.
So let me go ahead and start creating my own network.
Now you
already remember this command, right?
I can create my own network here.
I think dash slash help give you the commands.
Uh, you can create your network with a subnet.
The subnet is required in order to create one network.
So here, if not fine, set open network,

dash, dash subnet.

My range. Earlier it was going with 1 72 something.

Now in this range, I wanted to go

with hyphen D means driver.

If you don't specify, it'll be always default fund D.

Uh what? Bridge

network, name of the network.

Maybe something like my network.

I'm creating my own network.

Uh, Ravi.

Yeah, if we don't specify there

by default we'll also go into the bridge only, right?

Yes. Because we need to say post network

or non networking.

Need to specify. That's fine.

Okay. Okay. Now

our network call.

My network is created my network.

Now in this network, when I launch the containers,

now they will have the capability to communicate via needs.

A quick example actually

if I go like this, in which network it'll be created

default network, but we don't want default network.

Now where we want in my network,

simple dash dash network is a command.

And then you need to specify in which network you want

to launch, which network I want to launch.

That's it.

Now, it'll not launch in the default network.

It'll launch in the my network, which you created

even C two C3,

the newly launched C3 container where it is coming

see the IP address range.

But the advantage that you need to look at this.

So now the communication can happen over container ID
or you can also talk with the needs.
Are you clear? Is what I'm trying to say? Good.
Oh, now what? Same thing like again, I can just go ahead
and again, uh, what you say recreate all of those things
by C one C again in this network,
but instead of recreating, you can switch,
you can launch same way like how I did again,
install call and all of it.
Why install already? It's there.
So now you see what I'm going to do?
These two containers, they're already part of which network.
Default network
disconnect.
Uh, what you say the network,
which your container is in which network by default
bridge network, which container?
Now they're no longer part of Bridge Networks.
Then I want them to be connected
with the new network we created
which network.
Now I said connect both this particular containers on the
network called my network.
Now if you want, again, you can verify,
inspect, save,
and it got that ip.
And also the thing is what BNS names are enabled both
for C and as well as for C two.
You can see the same thing.
Uh, now I'll repeat the same story.
I mean the same thing
C one did it work?
It was saying no route to the host.
Now I'm doing the same command just

after enabling it in there.

Customer network.

Now clear is what I'm trying to say.

Now the communication is happening over names.

This concept is very important Later on in terms

of Kubernetes, that's what I discussed.

But you know, also got to know, okay, via containers.

I can also not only rely on IP addresses,

but I can also rely on names and IDs too.

Okay? Clearance everyone.

So communication via names

and communication via IBS would both be went

and we verified this concept.

Make sure you remember later on we are going to start this,

this, right?

Okay. Now I'll add the same things what I did right now.

Same commands, how

to create the custom network disconnect connect.

And I said no, no. Now this thing is working.

Now here it says in Kubernetes,

these concepts are required in cluster IP service

in Kubernetes network.

So again, I'm not going to discuss at that point of time

how name based communication

and all this works behind the scenes.

This is what is going to happen. Okay?

If name based communication is happening,

some custom networkers better keep that in mind.

Okay, now, now concept layer.

So internal communication with containers.

V verified container to container.

Container to host verified container to outset.

Also verify IP based communication.

Verified name based communication is also verified. Perfect.

These are all the things you should be
aware in terms of network.
Uh, now good, I can see the things.
Now the main idea is what I'm creating all these containers.
Now end of the day, all these containers will be accessed
by customers with respective application.
Agree or not. But with that we get some problem stream.
I'll show you don't break. So let's say I will
exit from these things.
I will go back and I'll just remove this container
base we don't need right now.
Remove forcefully C one, C two.
This is just for helping you to understand the concept.
Now I'll do as usual what I did earlier.
You see what I'm doing right?
What I did there.
Does it make sense when I send the request on 80 from the
system, send it to 80 on this container with the login app?
I said agree or not
working.
Yes, loading fine. But do I need this application?
What is actual application? I need login app.
No, I want to go ahead and set up the login app.
Fine, I'll do it. Do we have the commands and all of it?
Yes, right,
I'm here.
Okay, fine.
Agree.
What's happening?
It's still loading. Welcome to next one.
But already the code is also correct.
It was implemented in the proper place.
Why it is not working? Because in containers it is not
exactly same as VMs.

This is the approach in vm in container.

It is not in container.

You need to read that images,

which image we are using right now

for every image like this.

You see what they're saying? Static HTML directory is what?

That's the location. I'll take it.

That's where their pages are coming from

and it clear is what I'm trying to say.

Now, if I try to clone to this location,

it'll obviously not work

because clone will only work in an empty direct.

What I'll do

remove all the HTML pages

because there are only two files now

after cleaning up, is it cloning

work?

Understood concept. Like what I want to tell.

Don't think that whatever you did in VMs,

it'll work exactly same in containers mode.

You need to use the image description.

You need to read about the image.

Then you need to start fine. Perfectly good. I did this. I.

Video: Docker Data Persistency

Just assume this is some important container with me

and I'm going with container.

Okay.

Here what I do,

I have executed this in this, uh, what do you say?

Important container. We got bash access. Now you see list.

I'm seeing this set of directories linked.

I'm creating one folder called as important data,

and I'll just use some type command,

right?

So I assume this is some zip file or whatever it is,

but it's very, uh, data.

It's very important data that you do.

You do not want to lose base, like kept it in the container.

It was generated inside the country.

Agree or not, it's not coming from the host.

It was generated inside the country.

Like,

now, will the data exist?

I can never access the data if I once

again create this container.

Can I access the data? No.

Can you see any important container here or important data?

Go Meaning what? Data is temporary inside the containers.

That's why containers are called as non-persistent. Okay?

By default, containers are non-persistent.

Now, I want to make the containers persistent.

Then I said the concept we use is called as volumes.

The same example I, I'll just keep here.

Same example what I just showed you, important data

and all of it.