**BUILDING CLOUD NATIVE AND MULTICLOUD APPLICATIONS**

### Welcome!

This course is intended for those interested in learning about migrating, modernizing, building and delivering applications in cloud native and multicloud environments.  People who need to make decisions about application migration and enterprise modernization will benefit from the course.  It is also of interest to anyone and everyone who wants to learn about cloud native and multicloud environments whether they are C-level executives or if they want to work on cloud native development projects, on legacy modernization projects, on DevOps engineering projects or as an IT/cloud administrator.

Ce cours est destiné à ceux qui souhaitent apprendre à migrer, moderniser, créer et fournir des applications dans des environnements natifs et multicloud. Les personnes qui doivent prendre des décisions concernant la migration des applications et la modernisation des entreprises bénéficieront de ce cours. Il est également intéressant pour tous ceux qui veulent en savoir plus sur les environnements natifs et multicloud, qu'ils soient cadres supérieurs ou qu'ils souhaitent travailler sur des projets de développement natifs, sur des projets de modernisation des applications patrimoniales, sur des projets d'ingénierie DevOps ou en tant qu'administrateur IT/cloud.

### About this course

**Learning Objectives**

**In this course you will learn about:**

* + - Migrating existing images to cloud
    - Modernizing applications
    - Using cloud native practices
    - Establishing best practices for continuous integration and continuous delivery
    - Managing multiple cloud infrastructures, applications and middleware

Ce cours vous permettra de vous familiariser avec :

* La migration d'images existantes vers le nuage
* Modernisation des applications
* Utiliser les pratiques natives du nuage
* Établir les meilleures pratiques pour une intégration continue et une prestation continue
* Gestion de plusieurs infrastructures, applications et intergiciels en nuage

### Module 1 - Getting to the Cloud

### Learning Objectives

In this lesson you will learn about:

* + - Creating cloud-native applications.
    - Eliminating capital expenses by moving VMs to the cloud.
    - Migrating to the cloud using the lift-and-shift strategy.
    - Making decisions about data storage for cloud applications.

Dans cette leçon, vous en apprendrez plus sur :

* La création d'applications natives du nuage.
* Éliminer les dépenses d'investissement en déplaçant les VM vers le cloud.
* Migrer vers le cloud en utilisant la stratégie "lift-and-shift".
* Prendre des décisions concernant le stockage des données pour les applications dans le nuage.

#### [Lesson 1 - What do we mean by cloud native and multicloud?](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/407a9f86565c44189740699636b4fb85/12eab34ec218468995e4d06566ef4a32/)

## Lesson Introduction - What do we mean by cloud native and multicloud?

This lesson contains video content, as follows:

- **What are cloud native apps?** provides an introduction to the various layers of cloud native applications and explains the some of the differences between monolithic applications and cloud native applications.

- **What is Multicloud? How Do You Manage It?**introduces concepts related to using multiple clouds that will be explored throughout the course.

Cette leçon contient du contenu vidéo, comme suit :

- Que sont les applications natives du nuage ? présente les différentes couches des applications natives du nuage et explique certaines des différences entre les applications monolithiques et les applications natives du nuage.

- Qu'est-ce que le Multicloud ? Comment le gérer ? présente les concepts liés à l'utilisation de plusieurs nuages qui seront explorés tout au long du cours.

### What is Cloud Native?

Hi, I'm Andrea Crawford and I'm with IBM Cloud.

So, today we're going to talk about cloud native applications.

In the heritage world we have our lumpy,

monolithic applications,

and in the new world we have our microservices

living on the cloud.

If we take a look at this diagram here,

we see we have cloud infrastructure.

This is your private, your public,

and your enterprise infrastructure.

Cloud-native applications apply

to hybrid and multicloud situations.

We also have our scheduling and orchestration layer.

This layer is all about control planes like our Kubernetes.

We also have our application and data services layer.

This layer is all about backing services

and being able to integrate our application code

with existing services that may be available on other clouds,

or even on-premises.

We have our application run times,

these are what were traditionally,

or conventionally, known as middleware.

And over here, well,

that's where we have our cloud native applications.

This is the sweet spot right up here.

So, our application code

is actually designed, built and delivered

very differently for cloud-native

than it would be for conventional,

monolithic, lumpy applications over here.

So, let's talk a little bit about

why cloud native applications can actually leverage benefits

like enabling innovation, business agility,

- and, most importantly from a technology perspective,

the commoditization of the solution stack over here.

So, as time has progressed

and technologies have matured and emerged,

a lot of the services

are actually being re-factored lower down in this stack.

This means that core services

are starting to have a lower center of gravity,

freeing up innovation at this level over here.

So, what are our use-cases for

when to build a cloud-native app?

Star everything.

Everything that lives in the cloud should have

a cloud-native app design and approach.

This means our application code

needs to be instrumented with things like

standardized logging, standardize events,

and being able to match those logging and events to

a standard catalog

that multiple microservices and cloud-native apps can use.

The last thing we want to do is have our development squads

have to figure out what their log and event messages should be.

Let's standardize that

because we want to be able to commoditize that as well.

We also need to have things like distributed tracing.

When we get over into the microservices world over here,

we have a lot of moving parts.

This means we're going to need to leverage services

core to the system like load balancing,

service discovery, and routing.

These are the kinds of things that are commoditized

in this layer here with things like Istio,

and with the emergence of newer projects like Knative.

If you read the tea leaves, I think we're going to find

these types of technologies

embedded into this control plane layer here.

But they're still more innovation to come.

There are other technologies like Tekton,

which actually address CI pipelines

specifically for cloud-native applications

that leverage Docker and Kubernetes.

And so, if we were to recognize the benefits for cloud-native apps

and to sum it all up:

we are all about enterprise and engineering at scale.

Thank you for watching this video.

If you would like to see more videos like this,

you can subscribe to the IBM Cloud YouTube channel.

If you have any questions or comments

drop a line below.

### What is Multicloud? How Do You Manage It?

Hi, everyone.

My name is Sai Vennam, and I'm with the IBM Cloud team.

Today let's talk about multicloud.

Multicloud is a cloud computing approach made up of 2 or more cloud environments.

Let's get started.

So, we've got a public environment here as well as a private cloud.

There's an important distinction to be made here:

multicloud is not exactly the same thing as hybrid cloud.

Hybrid cloud implies that your workloads are working together across multiple clouds

- so interoperability and portability of your workloads.

Multicloud doesn't have that same requirement,

but we are seeing that a lot of enterprise users and customers

are using multi and hybrid cloud strategy together.

I'd say that the growth of containers and Kubernetes technology

has really enabled the growth of multicloud.

So, we're seeing almost every big cloud provider out there providing managed Kubernetes,

as well as it being used on-premises in the private sector

to help modernize legacy applications.

So, why would you want to use multicloud and the strategy for your cloud computing?

Well, there's a number of reasons.

For one, let's say that you want to hit that big

three-nines number as far as availability goes.

Multicloud strategy can help you hit that by ensuring that, you know, it's one thing

when the clouds of one of your application workloads stops to work you still have another

cloud that's supporting your applications.

In addition, you can enable better user experience.

So, let's say you have users on different sides of the globe.

By routing them to the cloud that's nearest to them, you can ensure lower latency and

better user experiences.

Finally, what about specific integrations that only work on a particular clouds?

Say you have some sensitive firewall data that you don't want to put in the public cloud,

but you need to build integrations for.

By taking advantage of a multicloud approach, you can build workloads on the private side

that can take advantage of that private, sensitive, customer data.

I'd say there's 3 major ways and things that you want to tackle when taking on a multicloud

strategies, and I'm gonna introduce these as three pillars.

The first one is gonna be automation.

So, to do this I, let me introduce kind two different types of users: so, we've got Ops,

as well as Devs.

So, operations engineers are going to want to be able to spin up new clusters as well

as manage and, kind of, see the different clusters that have been created.

So, they're gonna want to take advantage of a unified, kind of, dashboard to do something

like that.

So, we'll have the control plane here.

Let's say the ops engineers want to spin up two Kubernetes clusters.

So, they'll essentially figure out that configuration, and then go to the control panel, and tell

it to spin up those clusters.

At that point you know they can specify where they want those clusters to be created.

Let's say they spin up two.

Kubernetes clusters, one in each of the clouds.

On, the other side with devs, let's say that they've built out a new application, a Docker

container, and they're ready to push it.

So, they've got a container built, and they wanted to deploy it the same way to both of

the clouds but you know it can be quite difficult if they didn't have a control plane to figure

out the different authentications and configurations of the unique different clouds and kind of

deploy that both times.

In addition, this really gets in the way of traditional kind of CICD and DevOps workflows.

So, by taking advantage of that control plane, they can spin up that container in both of

the clouds with a single command.

Say that it's application 'A'.

So, I think that really hits this requirement for automation with multicloud.

The next kind of requirement that I want to hit on here is visibility.

Visibility is very important for working with multiple clouds because, you know, they each

have their own unique dashboards but to have a single unified way of managing those things

is very important.

So, for operations engineers, by taking advantage of a single control dashboard, they can have

access to a unified kind of approach and see all the different, you know, clusters that

they've spun up and even drill down deeper and see things like pods and deployments in

applications across multiple clouds.

The last thing I want to touch on, the last pillar, is governance.

These days regulatory and compliance policies are becoming increasingly strict and they

differ from geography to geography.

Let's say the operations team has a compliance policy that they want to push out across all

of these clouds.

So, first they'll have to, you know, write out that compliance policy, and then, you

know, if they didn't have access to a control plane, this can be a very kind of painstaking,

or rather time-consuming, approach.

So, by taking advantage of the control plane they can push that compliance policy across

multiple clusters with a single command and ensure those security policies for compliance

are being met.

I'd say that these are the 3 major things that you want to tackle when taking on a multicloud

strategy.

Multicloud has been a growing focus for us here at IBM.

If you want to learn more about our multicloud management strategy, check out some of the

links in the description below.

If you like this video, or have any questions, be sure to drop a comment and subscribe for

more videos like this in the future.

Thank you.

#### [Lesson 2 - Moving VMs and Containers between Clouds](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/407a9f86565c44189740699636b4fb85/b33a4cf71efb4f688374e4dbab6219b4/)

## Lesson Introduction - Moving VMs and Containers between Clouds

This lesson contains video sessions, as follows:

- **Virtualization Explained** provides an introduction to virtualization and explains the different types of hypervisors that can be important to your cloud computing strategy.

- **Modernize with VMWare** illustrates the idea of moving from legacy hardware and software using migration and modernization tools.

- **Deploy OpenShift on VMWare** shows how to move an application without changing anything to gain the tactical benefits and the elasticity of moving to the cloud.  
  
NOTE:  For additional information about containers, see Module 2.

Cette leçon contient des sessions vidéo, comme suit :

- Virtualisation expliquée fournit une introduction à la virtualisation et explique les différents types d'hyperviseurs qui peuvent être importants pour votre stratégie de cloud computing.

- Moderniser avec VMWare illustre l'idée de passer du matériel et des logiciels existants à l'aide d'outils de migration et de modernisation.

- Deployer OpenShift sur VMWare montre comment déplacer une application sans rien changer pour bénéficier des avantages tactiques et de l'élasticité du passage au cloud.

NOTE : Pour plus d'informations sur les conteneurs, voir le module 2.

## Moving VMs and Containers between Clouds

### Virtualization Explained

Hi, my name is Kaleigh Bovey with the IBM Cloud team, and today we're going to be talking

about virtualization.

As you know virtualization, is a fairly old technology, but it's still super relevant

to building your cloud computing strategy today.

So, first off: What is virtualization?

Simply put, virtualization is the process of creating a software-based (or virtual)

version of something, whether that be compute, storage, networking, servers, or applications.

And what makes virtualization feasible is something called a hypervisor.

So, we're going to write that here.

And, what a hypervisor is, is it's simply a piece of software that runs above the physical

server or host.

There are a couple different types of hypervisors out there, and what they do is, essentially,

pool the resources from the physical server and allocate them to your virtual environments.

There are two main types of hypervisors out there - one being Type 1.

Very simple to remember.

And two - you guessed it - Type 2.

So, let's start with Type 1.

A Type 1 hypervisor is a hypervisor that is installed directly on top of the physical

server.

They're also called bare metal hypervisors.

So we'll write that up here, just so you can remember.

These are the most frequently used types of hypervisors, and they're the most secure,

they lower the latency, and these are the ones that you'll see in the market the most.

Some examples would be VMware ESXi, Miscrosoft Hyper-V, or open source KVM.

The other type of hypervisor is a Type 2 hypervisor, over here.

And what makes these different is that there is a layer of host OS that sits between the

physical server and hypervisor.

So, by that nature, they are also called hosted.

These are a lot less frequent.

They're mostly used for end-user virtualization, and you might see some of the market that

are called Oracle VirtualBox or VMware Workstation.

Again, they are a lot less frequent, they're a bit more - they have a higher latency than

a Type 1 hypervisor.

Building virtual machines (VMs) So, once you have your hypervisor installed,

you can build virtual environments or virtual machines or, simply put, VMs.

So, let's spin up some environments.

So, what makes a VM, a VM?

A VM is simply a software-based computer.

They run like a physical computer, they have an operating system and applications, and

they're completely independent of one another.

But, you can run multiple of them on a hypervisor.

And the hypervisor manages the resources that are allocated to these virtual environments

from the physical server.

So, because they're independent, you can run different operating systems on different virtual

machines.

So, you could run Windows here or Linux here or Unix here, for example.

Because they're independent, they're also extremely portable.

You can move a virtual machine from one hypervisor to another hypervisor on a completely different

machine almost instantaneously, which gives you a lot of flexibility and a lot of portability

within your environment.

So, looking at all of this - this is the core of virtualization as a process.

So, let's talk about a couple key benefits that you want to take away from this.

One - cost savings.

When you think about this and the fact that you can run multiple virtual environments

from one piece of infrastructure, it means that you can drastically reduce your physical

infrastructure footprint.

This is consolidation at its core, and the fact that you don't have to maintain nearly

as many servers, run as much electricity, save on maintenance costs means that you save

on your bottom line at the end of the day.

Number two would be agility and speed.

So, like I said, spinning up a virtual machine is relatively easy and quick - a lot more

simple than provisioning an entire new environment for your developers.

If they say they want to spin up a new environment so that they can run a dev-test scenario - whatever

it might be - virtualization makes that process a lot simpler and quicker.

And three - it lowers your down time.

So, let's say that this host goes out unexpectedly.

The fact that you can move virtual machines from one hypervisor to another on a different

physical server means that you have a great backup plan in place, right?

So, if this host goes down, you can simply move your VMs very quickly to another hypervisor

on a machine that was working.

So, with this - this is really virtualization today.

And like I said at the beginning, virtualization is a technology that's a few decades old at

this point, but it's still super critical to understand for your cloud computing strategy

today.

Thanks for watching as we discussed the basics of virtualization.

Make sure to subscribe below and give us a big thumbs up if you liked this content.

### Modernize with VMWare

We've gotten a few things already sketched out on the board, but let's

actually start with the stack that we need to get started with, you know

modernizing with VMware. Yeah, and IBM Cloud, kind of our basic or fundamental

VMware software stack it is quite simple, right. You have your bare metal

infrastructure where your environment runs on, so bare mental infra. And

then on top we have your vSphere and so we actually do automated deployments of

6.5, 6.7, and any future releases from a VMware perspective. We have NSX which is

the VMware networking tool, and sitting on top of that kind of the brain of the

whole operations is vCenter, right. And when we look at this

this is just the very basic fundamental stack there's tons of more components

that you can add. For example, you can add vSan, which is the storage option, as

well as any kind of day two operations on top of the stack. Gotcha. So, for an end

user who's coming in and they want to get started they could just go directly

with vCenter and IBM will kind of automate and take care of the rest of these

components, get them all installed? Absolutely, everything we do is from an

automation perspective too and we understand people have different

requirements, right. I talked about vSAN earlier, you know customers

prefer other types of storage compared to vSAN or they want to use other

networking tools so they don't actually even need to use NSX, it's just an option

and kind of what we do from a stack perspective. I see. Ok, so we've got a

number of things already sketched out here like I mentioned and we want to

show kind of and really break down how we modernize from let's say legacy

architecture infrastructure to something like VMware and IBM Cloud. So, we've got

some things sketched out and can you kind of talk through what we have already? Yeah,

absolutely. So, I think one of the things we were talking about earlier today was

modernizing means a lot, right. And one specific use case I want to look at is

modernizing from legacy hardware and legacy software, and calling out really

VMware HDX. So, HDX is a migration, hybridity modernization tool, whatever

acronym you want to put with that. And so it allows customers to actually migrate

from vSphere 5.1 which can be running on any hardware you want, and this is an

old software stack. And so, the nice thing that it allows you

to do is actually create a connection where you can actually go from old

hardware in 5.1 into the newest hardware in 6.7 and whatever VMware is going to

be releasing in the future. And how it does this it actually makes a

kind of a loosely coupled network which allows you to essentially just stretch

your network. And that's the really cool thing is they're kind of stretching your

L2 network it allows you to kind of create these connections where you can

bring your own IP, you can bring your MAC addresses, and the nice thing since

you're bringing those backwards and forth it really provides us true

hybridity or multicloud approach. Right. So, essentially all the connections that

the applications may have had previously, so IP address dependencies and those

kinds of things you don't really have to worry about refactoring or breaking all

those down since you're bringing your own IP, since you're kind of having that

that model, those applications continue to work with each other even though now

you know parts of them are on the cloud. Absolutely, and one kind of story

from one of our customers that I love to talk about is you know we had a customer

running on vSphere 5.0, on old hardware, they were in a datacenter in

California and they were looking to try VMware on IBM Cloud and they

actually moved that application from California all the way to Washington D.C.

So you're talking across the country and so obviously with that they increased

latency compared to the other VMs in their application, but that application

actually ran better in IBM Cloud you know even though it was in Washington D.C.

because it's running on a latest software and hardware version compared

to what the customer was running on-prem. So, the advantages in the compute

basically outweighed that the disadvantage of having latency across

the country. Exactly. Wow, okay. And so that's just kind of one of the use cases

we really like to look at because what HCX allows you to do it cuts this notion

of hey I'm in a monolithic, or in you know legacy app, you know hardware

applications, I can't move. HCX eliminates that by really providing

that backwards compatibility where you can take everything you have existing

today and move it into your future state. Right. So, in this kind of example we can

probably say this layer right here, or at least HCX is enabling that, excuse me,

that translation. So kind of moving forward, so let's say we

have that that hybrid model on-prem and on the cloud, what if we wanted to start

moving some of these assets, right. So we want to move those database pieces and

put them over here, is that is that something that HCX would help me do? Yeah,

absolutely. So the beautiful thing about HCX is it allows you to move when

you choose, right. I can group individual VMs, I can do a live migration, I can do

you know planned migration, and it ultimately depends on what the customer

is looking for, but really the thing that we see with HCX and you know any

migration tool it's a lot of it is analysis paralysis, and with what HCX

allows you to do is it allows people to feel comfortable because you can start

moving one VM, two VM, and start building up the comfort of moving you know back

and forth. Right, definitely. So, you know with HCX I can run this for

you know, let's say however long to make sure it's stable, we're not having any

downtime that you know critical business workloads are continuing to run and then

once we've kind of already we can start duplicating some of these assets, right.

So maybe run some of those DB pieces over here in the cloud, and I would guess

eventually even think about phasing this portion out. Absolutely, yeah, and it's all

a matter of what the customer is looking for but if you think about from a data

consolidation or a data evacuation use case that's exactly what we see with HCX,

you take what you have existing from an IP parameter and the nice thing you can

move to the cloud, if it doesn't work you can move it back, but if it moves it runs

better you can actually shut down your own data center and save you know a

significant amount of cost, but also get you to where you want to be, right, cloud,

multicloud environment. Right, exactly. So, Jordan thank you for breaking this

down for me and our viewers. If you have any questions, please drop us a line

below. If you want to see more videos like this in the future, please like and

subscribe. And don't forget you can always get started in the cloud at no

cost by signing up for a free IBM Cloud account.

### Deploy OpenShift on VMWare

Matt has sketched out a few things already and we're gonna be talking about

you know taking advantage of VMware on the cloud, but a little bit of a focus on

lift and shift, as well as transforming. So, Matt real quick what do we have

So, Matt real quick what do we have graphed out over here? So right here is

we're kind of picking up from where we left in our last webcast that we did

with Jordan where we talked about more just lifting and shifting the

application, re-hosting it if you will. Where we're more concerned about how do

we get the application and the VM without changing anything about it into

the cloud as quickly as possible, and extending that VM and all the network

policy and everything underneath it into the cloud for consistent operations with

on-premise and in the cloud. Right, so last time you know we saw that with

things like HCX it was fairly easy to re-host capabilities from a VMware

running on-prem to something running in the cloud. Right. So, that's what we

have here right? So right here you have an on-premise right, that's what that side

is suppose to represent. And then after lift and shift you have your in the state called

cloud hosted, like what we talked about. Where it's just more moving the

application without changing anything about it just to get the tactical

benefits, and the elasticity of moving to the cloud. And you know what we're

talking about back at the at the desk is that by re-hosting first and in having

your application live in the cloud brings a host of advantages in terms of

the app modernization, because right now we're just focused on getting on the

latest and greatest infrastructure that the cloud has to offer. Right. And then

after that taking advantage of the automation and the and the scalability

and the standardization of the cloud to layer on more services like OpenShift,

and like any other kubernetes services out there. So, that's that's kind of this

you know this time frame that we have laid out here right so every step of the

way you know you're starting on-prem, you're taking that next step to cloud

hosted and now what's that next step? Like from that last

kind of webcast that we did we understood there was a lot of

capabilities helping us here, but you know at the end of the day customers

want to take advantage of higher value services. Right. And capabilities that you

can get in something like IBM Cloud. Exactly, they want of like we're talking

what they want to bring start introducing cloud-native services and

then just as an organization get more adapted to the

the value of being in the cloud, right, and the elasticity of that. So at that stage

we are now at what we call the, I'm going to try making a nice square here, at

cloud-enabled is what we call it. And in this phase we are now taking

advantage of being on the the cloud network right because now you're in the

IBM Cloud, or the cloud in general whatever club that might be, you're in

there private network, you're taking advantage of the services that are on

the the private endpoints and such as you know object storage or analytic

services that might not require changing the application too much, you can still

keep it in a VM. Kind of like plug and play services. Exactly, and you're

not you're not refactoring the application just yet. So, from the

application level you're enriching it as much as you can from the infrastructure

level you're taking advantage of the the consumption models that the cloud has to

offer and using things as needed. So basically your VMs are continuing

to run in a cloud environment but now you're kind of reaching out and taking

advantage of some of these cloud propositions, those cloud services. Exactly, to the extent that it can with the application in it's probably

most likely it's in a three-tier architecture, right. You have your your web tier, your

app tier, and your database tier. Right. And as much as you can integrate

services and that kind of architecture you're taking advantage

of the cloud enabled stage. Right, so now it's less about just lift

and shift and more about transform. Right Maybe you have to get in there sometimes

change some things around, call some third party APIs, you know, yeah that

kind of thing. Exactly, and then after you know from an

operational perspective you're more adapted and ready to take the next

step. This is where we introduce OpenShift and Kubernetes, you know you

have this VMware estate that now lives in the cloud, you have the VMs

there, you have your legacy applications living on top of them. Now you can start

to think about re-platforming, or even re-hosting, or not re-hosting, you can

start rearchitecting or refactoring the application. So, here you'll see that

we'll have the VM here, and then you'll see that we have we're gonna use

a container right here. So this is what we're going to use to to label a

container. And this is still kind of you know, you're not fully cloud-native

yet, so we're gonna call this cloud labeled as well. And at this stage you

know...You've got them side by side like VM's working alongside containers in the

environment. Assuming all the networking pieces are in place below and you have

this common network fabric you know and that's the value that NSX from

VMware brings is that you can have this common network fabric across containers

and VM's. Now you're able to start to thinking about, all right I want

to keep my databases of VM because that's stateful, right. Right. And I

have these stateless components of my application that don't really

rely on storage as much and can be containerized and re-platformed and

that's what we're doing here and it's kind of like this hybrid world where the

application is living as containers and also as a VM too. Right. And so, this you

know assuming this an old monolithic, heritage application that

there are some some benefits that the app tier and the web tier can

get out of being a container. Exactly, so let's say in that VM like you mentioned

we had all three tiers, what if we break apart a front-end that can really

take advantage of things like cloud auto scaling capabilities and the fact that

you can distribute that cloud application across data centers, you know

all over the world where IBM Cloud has data centers. Basically allowing that

front-end that's not you know stateful, it's a stateless application to take

better advantage of cloud-native concepts right away. Right.

Whereas kind of those legacy apps as VMS where you have to start thinking about

refactoring if you wanted to make them cloud-native, kind of changing that

that's stateful layer. Those might take some more time. Right. So, that's that

halfway step, that modernization step. Exactly, this is the in-between phase and

really this is this this part of the story here is really what the value of

lift, shift, and transform is that customers are doing this at their own

pace, it's not a sushi roll of a solution that they need to take a bite of the

whole thing altogether. Whether there's pieces of that sushi they don't like,

right. So, they're able to take it step-by-step here and in a way that

makes sense to them that fits their strategic goals, right. And you know

there's some other numbers out there, there's 50 million VMs that are

still out there in the world today, right. And a lot of most business

applications still run on VM. So, I think it's it's wise to realize that the

future will still have VMs in it and we'll probably see scenarios like this

where we have databases living in VM still, but you

have all the that client interfacing parts of the application, all

containerized and able to introduce new functionality to it. Right. I mean

there's still some capabilities out there that were written you know in like

the 70s and 80s that are still chugging along that no one wants to go in there

and change those out. Exactly, and like we talked about back there that you know

they're the people that actually wrote the teams that wrote those applications

they're long gone and it's it's lost in translation. So, I'm

guessing, you know you hinted at where you're going with this last phase here,

the end of all of this modernization phase we're gonna have..? Cloud-native, heeyyy. (laughter)

So, by this point, you know we've kind of decoupled the application from

virtual machines all together, at least the parts that that were that are you

know in in containers now. So, we don't really need to rely on hypervisors as

much anymore. Now that the application is abstracted enough that we

can start taking advantage of higher value services like OpenShift as a

Service. So, you don't really have to rely on the the actual hardware anymore, you

can take advantage of services like the IBM Cloud provides that make OpenShift

more of like an endpoint where you're only bursting as much

OpenShift capacity that you need for your application to run and to

develop with it. So, let me draw this out here. So, you have your application here

and it's running now and let me label it cloud-native. Cloud-native, that's right. So, this

is kind of when, you know you've phased out some of those legacy

applications. Right. Many of your assets aren't required to run as a VM anymore.

Right, so and then you're taking advantage of, let me get my green marker

here, of services like our managed Kubernetes service here, and also now

that you're at this cloud-native development stage you can start taking

advantage of other microservices and you yourself have turned your

application into microservices too. Right. So that it's just working, you

know, where you've achieved this cloud-native development state. Right. So, there

we're taking advantage of you know, we have machine learning capabilities, higher

value AI, those capabilities that really enable you, like kind of like we

mentioned to create more engaging applications, applications that are

connecting to your end users just a little bit better. Right. So Matt this was

really helpful to be able to actually sketch this out and show how, you know,

how every step of the way with VMware solution,s with

VMware solutions on IBM Cloud, you're supported kind of every phase of that

modernization path. Right. If you have any questions please drop us a line below. If

you want to see more videos like this in the future, please like and subscribe. And

don't forget you can always get started in the cloud at no cost by signing up

for a free IBM Cloud account.

#### [Lesson 3 - Infrastructure as Code](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/407a9f86565c44189740699636b4fb85/f1b8543b5f68457e864e15a4477c2109/)

## Lesson Introduction - Infrastructure as Code

This lesson contains video material, as follows:

- **What is Infrastructure as Code?** explains why it is crucial to automate your infrastructure as code to handle applications that can be deployed into production up to hundreds of times per day.

Cette leçon contient du matériel vidéo, comme suit :

- Qu'est-ce que l'infrastructure sous forme de code ? explique pourquoi il est crucial d'automatiser votre infrastructure sous forme de code pour traiter des applications qui peuvent être mises en production jusqu'à des centaines de fois par jour.

## Infrastructure as Code

### What is Infrastructure as Code?

Hi everyone, my name is Sai Vennam and I'm with the IBM Cloud team.

Today let's talk about Infrastructure as Code.

These days it's increasingly crucial to automate your infrastructure as applications can be

deployed into production up to hundreds of times per day.

In addition, infrastructure is fleeting and can be provisioned or deprovision in response

to load.

Let's start with an example.

Let's say you're building out an application and you've chosen a public cloud.

Now, the first thing you decided to do is build your application on Kubernetes.

So, we'll have a Kubernetes application stack.

Now we don't actually have to dive deeper into Kubernetes because it isolates the hardware

from the application layer.

So, we don't actually have to go in deeper and it'll manage that for us.

Next, let's say that after a week of development we've decided to bring in a VM that holds

a legacy application that we have not modernized just yet.

So, we'll bring on a VM and now to actually connect up those dots will need a VPC.

So, there we go, we have a basic infrastructure in place.

Now, let's say that I've developed this, it's great, all the infrastructure details are

documented now I'm ready to move it into a test phase.

Now, I know that for best practice what I should do is create a whole new environment

that mimics my dev environment.

To do so I'll go back to my documentation and start following the steps to spin up that

infrastructure.

But let's say that maybe I forgot to document one of the config switches that I've changed,

or maybe the platform is different in how it handles provisioning infrastructure regardless

the application and has don't work the same way in the new environment.

I decide okay we need to fix this problem and to never have this problem in the future

again we need to take advantage of infrastructure as code.

Let's talk about the first approach to infrastructure automation, it's going to be imperative.

Now, this is kind of intuitive for most people because an imperative approach allows you

to define step-by-step how to get your infrastructure into a certain state.

So, in general an imperative approach would use something like a CLI along with maybe

a bash script.

So, for example in this in this case we could do something like CLI Create Kubernetes and

then we would define some maybe additional commands to customize that Kubernetes deployment.

We'll do the same thing for the VM, as well as the VPC.

So, an imperative approach has an advantage.

It allows you to really define step-by-step of how your infrastructure is provisioned

and that generally comes with more power as well because you're using the cloud tools

and doing it in a kind of step-by-step process.

But at the same time this can come with complexity.

For example, if you wanted to tear down your VMs, or your environments rather, or let's

say you wanted to scale it up or down, you'd have to write custom scripts.

It's not handled for you in an imperative approach, so this generally doesn't scale

well.

Another approach to Infrastructure Automation is going to be Declarative, and this is actually

my favorite approach.

Now, a declarative approach would be something like Terraform and this is what it's basically

allows you to do is to define the final state of your infrastructure and then it let's a

provider handle the rest.

So, instead of defining every step you just defined the final state.

So, in this example maybe you would do something like define a Kubernetes resource and a VM

resource, as well as a VPC resource.

Another great thing about this is it's generally managed through just simple config maps.

So, if you wanted to do something like define a host you could do that, maybe a domain,

or maybe even the subnets.

So, in general a declarative approach allows you to more easily manage the configuration

and is my preferred approach for automating infrastructure.

Let's take this simple example, if you ran the imperative script multiple times you would

actually end up with a multiple environments and in addition, let's say one of the steps

halfway through failed you would have to add error handling to tear down the steps that

did succeed.

Now with the declarative approach no matter how many times you run the script you end

up with the exact same infrastructure.

So, you could do it the first time, provision your environment, and then maybe run it again

later on to ensure that your environment hasn't changed.

So, I'd say this is very important to can understand the different approaches infrastructures

code but in general I do prefer a good declarative approach.

Next let's talk about DevOps.

Now we all understand how important about DevOps is.

When developing an application, you'll first write some code, you'll want test that it

actually works and then you want to push it into production.

And then you want to make sure that all of that is kind of always working and you can

repeat those processes.

Now, I know there's teams out there that have a perfect agile DevOps flow but because they're

working with legacy infrastructure they have to open a ticket every time they want to get

a new VM and that's kind of just due to the infrastructure that they're running on.

Now that really holds them back.

Now with Infrastructure as Code when it supported it allows you to treat your infrastructure

with the same level of quality that you treat your code.

So, this includes things like versioning so essentially you want to make sure that any

time infrastructure changes you're tracking that and is generally a best approach for

automation.

The last thing I want to talk about is immutable verses mutable infrastructure.

Now breaking that down, an immutable infrastructure is one that can't be changed, it can't be

mutated.

Now at first impression that might sound like a bad thing but let's break this down by seeing

an example with a mutable approach to infrastructure architecture.

So, we have our example here and let's say that we're building out the app and we decide

we need a database.

So, to do so we'll run a script for a dev environment, this brings up that database

within our VPC.

Now all of this is working great, so we say, "Hey let's just run that script across all

of the environments that we have."

Now let's say ninety-nine percent a time that works fine but some of the times it fails

you're in a weird limbo state, so let's break that down.

So, we're going from version one to version two.

Now we have infrastructure code in place to bring up version one, but now we ran this

custom script to move it from version one to version two.

What we essentially have right now is something called configuration drift, or environment

drift.

Our existing environment is no longer matching what we have in our automation.

Now the problem is to kind of help debug those problem situations you would have to wipe

out the entire environment and then redeploy version one and then run those scripts.

Now that might seem okay the first couple times to do it but when moving to scale it

becomes incredibly hard to maintain.

So with an immutable approach to infrastructure and infrastructure automation every time you

want to make a change to the infrastructure you bring up a brand new environment alongside

the old one and then once you verify that they're both working you can bring down the

older version.

Now it's a little expensive because you can imagine that you're running both environments

at the same time, but in general it's the best practice to ensure that your infrastructure

can move to scale.

Thanks for joining me for this quick overview of infrastructure as code.

If you want to learn more about this, or DevOps check out the link below.

Remember you can always get started with a free account on IBM Cloud.

Thank you.

#### [Lesson 4 - Data Considerations](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/407a9f86565c44189740699636b4fb85/af5d5c59cc9c45c79cd7d63039903807/)

## Lesson Introduction - Data Considerations

This lesson contains video sessions, as follows:

- **Cloud Migration** provides information about getting workloads off their enterprise infrastructure and onto the cloud.

- **Big Data Explained** helps you understand the reasons why enterprises are so interested in moving workloads to the cloud.

- **Data Migration Examples** introduces serverless technology and how it is applied to big data analytics.

Cette leçon contient des sessions vidéo, comme suit :

- La migration vers le cloud fournit des informations sur la façon de transférer les charges de travail de l'infrastructure de l'entreprise vers le cloud.

- Big Data Explained vous aide à comprendre les raisons pour lesquelles les entreprises sont si intéressées par le transfert des charges de travail vers le cloud.

- Exemples de migration des données présente la technologie sans serveur et la façon dont elle est appliquée à l'analyse des grandes données.

## Data Considerations

### Cloud Migration

101 – Cloud Migration audio transcript

Hi, I'm Andrea Crawford with IBM Cloud and we're going talk about cloud migration.

Take a look at this picture here.

We have enterprise infrastructure.

This is typically where a lot of our compute workloads are bare metal, enterprise core

systems mainframe, on- premises data center.

We have private cloud over here, also on-premise but cloud capabilities in full effect.

And then we have public cloud, this is off-premise cloud capabilities.

Now many of our enterprises are still struggling to get workloads of their enterprise infrastructure

and on to the cloud.

So, when we migrate enterprise workloads to private or public, we call that migration.

And this is all about understanding enterprise workloads and their characteristics in terms

of whether they should land on-premise cloud, or off-premise public cloud.

We also have modernize.

There could be some workloads over here that we might be able to re-factor if you will.

Think about a mainframe monolithic code, if we could put some API's on top of that to

expose core business functions, we might be able to modernize functions in a way where

those API’s might be able to live here.

And then we have our third use case build native.

For those workloads that live over here if we have an opportunity to do things over in

a way where we could design cloud native apps to live either here, or here we would be able

to infuse qualities like correlation ID's to track microservices and where traffic flows

we will be able to leverage a lot of the core functions for load balancing and service management

here in the clouds.

The clouds offer some core capabilities within the platform so that we can free up some of

the logistics and the application layer so that we can be more innovative which brings

me to the benefits.

So, moving, migrating, modernizing applications to the cloud is cost effective.

So, we can lower our costs specifically around public, around OpEx and being able to let

our cloud providers take care of a lot of the management of those workloads.

Another benefit is scalability because cloud heavily leverages virtualization

and the ability to do things like Cloud Foundry and Kubernetes, we can leverage the core services

in those particular models, but we also have the flexibility to bare metal, and even virtual

machines.

Another benefit is security.

One of the great things about cloud is we don't have to worry about a lot of the core

services that we would have had to worry about over here.

A lot of our cloud providers nowadays actually address a lot of the security concerns where

you have compliance regulation, data concerns, these are services that can be provided here.

We also have as our last benefit, accelerated adoption.

So, by leveraging the services in the cloud, like service routing, service discovery, and

load balancing, we don't have to worry about that at the application level.

So, we really start to understand why enterprises are really chomping at the bit to move workloads

from here to here, or here.

And that is cloud migration.

Thanks for watching this video.

Be sure to like and subscribe, if you have any questions or comments drop a line below.

### Big Data Explained

Hello, this is just Torsten Steinbach, Architect here at IBM for Data and Analytics in the

Cloud, and today I'm going to talk to you about serverless technology and how it is

applied to big data analytics.

When we look at big data in the past few decades, we can see that there has been a traditional

form factor of big data systems that has been used for many decades already and this is

the form factor of a data warehouse.

So, this is a highly integrated system, highly optimized for handing big data queries, big

data analytics in a very efficient manner.

Nevertheless, we had around the year 2000, Hadoop coming up and being adopted very rapidly,

and gaining a lot of popularity in this now widely adopted industry.

Even though there was already big data analytics, so why is that Hadoop came up?

So, this is because it brought in addition to this integrated system more openness to

a table.

More openness, in terms of the type of data that it could handle, bring your own data

formats, the types of analytics, analytics libraries, and languages that can be supported.

And also, the flexibility in terms of the hardware, the deployment options that you

can have.

You can bring your custom hardware or even have heterogeneous hardware.

So, that's why Hadoop basically gained a lot of traction and is now widely adopted.

Today, however we are seeing a trend that basically results in yet another form factor

of doing big data analytics, and this trend is driven by actually one thing that is happening

which is era with the rise of cloud.

And another thing to actually goes hand in hand a little bit with the rise of cloud is

the consumption behavior of many people of end users to be more oriented on the sharing

economy.

So, people are using more and more just ride shares instead of just renting a car and not

to speak of buying a car just to get around.

Or they are just going with Airbnb to sleep a night somewhere.

So, this consumer behavior is also applied now to a tee and this term serverless is actually

explained as this, serverless is in fact the sharing economy for a team.

And it is it is enabled by cloud.

And it is in fact the most consequent usage model of cloud, serverless.

And many of you have heard from serverless, and probably most of will associate with a

thing called function as a service with serverless.

Many of you may think it's synonymous which is not exactly true, but that is what basically

many people think of and function service is I have my code that I need to run my business

logic but I don't provision dedicated systems, dedicated hardware or not yet not even dedicated

software I'm just sending it to service and saying please run it for me.

Run it for me maybe that many times.

So, how to scale out and it's all done adhoc.

It's basically hiding the fact that there is a server.

That's why it's called serverless.

Now, as I said this is what many people think of when they hear the term serverless, but

serverless is more than just function as a service.

Especially when we now look at it back out here again which is big data and analytics.

The problem with big data analytics is that we are talking about state.

State has to keep data safely and durable and be reliable so I can be able to access

it anytime I want it.

And that's what these systems provide.

But now in the cloud we have new options.

We can actually extract the storage of data itself as a cloud service on its own.

And that's also what's happening on the cloud and it is basically cloud-native storage of

object storage.

And object storage is basically serverless storage because you do not provision disk

volumes, you do not configure disk volumes and you just bring your data and the system

figures out how to store it and basically how to distribute it to make it highly available

and so on.

It's highly abstracted, you just have a rest API where you upload and download your data

and you can come with kilobytes of data going up to terabytes of data in the same organizational

unit.

And to think about why it is serverless, is also that it's a pay as you go consumption

model.

You just don't use it as you go, you also to pay as you go.

Which means you're just paying for the gigabytes if you're storing at this point right now

and if you just told us you will be paying us in a very elastic completely seamless the

elastic my.

Now, maybe we talk about big data analytics, it's not just about storage of data but also

how can we analyze this data and process this data.

And this exactly now what we are seeing as well driven by cloud, we are seeing additional

services that are made available around object storage such as sequel as a service or allow

it will allow you to run SQL.

Basically, all the data in object storage and just be built for this one SQL, depending

on how big the SQL was in terms of data it had to scan and you do not pay for database

that is provisioned and standing around, just a single SQL and that's it.

And there are other things that basically play into this like for instance messaging

as a service, where you are just paying by the number of messages being processed and

then eventually stored to the object storage.

So there's a series of these services basically coming up, and in combination they are providing

this new form factor of a big data and analytics system that is augmenting and actually complementing

the existing form factors because even though there are more established and older there

are still a point for using them.

They have their sweet spots in terms of their own performance characteristics and response

time guarantees, but on the other side there are maybe cost effectiveness benefits here.

So, depending if your business model and requirements you may use this or this or the combination

of those things.

So, I hope this helps to put in perspective a serverless play into big data analytics

and how it basically generates a whole new form factor with big data and analytics systems.

Thank you very much.

### Data Migration Examples

Hi there, my name is Katie Morgan and I'm with the IBM Cloud team

and today I'm going to be talking about data migration.

So, before you can actually start using the cloud,

you'll have to first figure out

how you're actually going to get your data to the cloud.

In my experience, there are 3 primary factors that you should be considering

when you're looking at data transfer methods.

The first being the type of workload that you're moving,

and the second is how much data are you moving.

Thirdly, how quickly do you need the transfer to occur?

So, for large scale data migrations,

and by large, I mean terabytes to petabytes worth of data,

cloud providers will typically provide you with a portfolio of options

such as products, services

- that enable you to move your data from point A to point B.

And most of these portfolios span two primary categories:

offline transfer and online transfer.

For offline transfer, which is great if you're in a remote location

or if you're in a place where high-speed connections just

are unavailable or are just cost prohibitive to you.

Offline transfer options are great because they leverage

portable storage devices to move your data from point A to point B.

The first being a customer-owned device.

And what that looks like is you sending in your own piece of hardware

whether it's a USB stick, external hard drive, CD, DVD, or something like that,

to a cloud provider's data center for connection.

And once that device is mounted, depending on the cloud provider,

either you will remotely control that data transfer,

or they will initiate the transfer on your behalf.

Once the transfer is complete, they'll ship the device back to you,

or some providers actually offer to destroy the device

on your behalf if that's not something that you're interested in.

So, not a hard and fast rule,

but we often recommend a customer-owned device transfer method

for workloads that are 10 terabytes or less in size.

Again, not as strict rule, but a good rule of thumb to go by.

And for workloads that exceed that ten-terabyte capacity,

we'll often point people towards provider-owned device offline transfer options.

And what that really looks like is your cloud provider

shipping you a large capacity portable storage device

to your location for you to put your data on to it

and then immediately send back to the cloud provider's data center.

Once it gets back to that cloud provider,

they're going to immediately offload your data from that device

and into your target caught environment.

Once the transfer is complete, absolutely go free and access your data

while the cloud provider will securely wipe that device of your data

and immediately to return the device to inventory for reuse for the next customer.

So, similar to the customer-owned device,

we use this as the standard benchmark for capacities

when using a provider-owned device,

and that's really tens of terabytes to hundreds.

It depends on the cloud provider that you're working with.

Some of the devices actually span from single terabytes in capacity

all the way up to a petabyte scale,

it just depends on who you're working with and what you're trying to do.

And finally, if you're really not looking for an offline transfer,

you want to transfer data over the network,

or you're really looking for that high speed technology,

that's when you want to consider an online transfer option.

You can write custom applications using high-speed transfer libraries

or spin up a high-speed transfer client at your location

and connect it to the cloud provider's high-speed server cluster.

Something to consider with online transfer, as well as offline,

as I'm sure you can tell your network connections and speed significantly impact

all of these options but especially the online transfer.

If you're thinking that your transfer time is really going to creep up

into that week-long or plus duration for a migration,

you might want to consider a combination of any of these offerings

or really an offline transfer.

The longer that you spend migrating using over-the-network options,

the longer that it will take and the higher the cost, typically.

So, if you're looking to drive down costs you definitely want to keep that in mind.

And then, finally, just a couple of things that you should

probably consider with some of these offerings:

with the customer-owned device, definitely look at your cloud provider's web page.

They'll do a good job of outlining any hardware specifications or requirements

so that you are

able to send a device that's actually compatible with what they're looking for.

For the provider-owned device area,

you definitely want to look at their web pages and see

any features and benefits that the varying devices and capacities will offer.

The size of your workload will really

determine what capacity you're looking for in terms of device.

And then extra bells and whistles like GPS tracking or edge computing,

definitely look and see if any of those peak your interest

and see if the device models match.

Thanks for watching this video on data migration.

If you have any questions feel free to drop us a line below,

and if you like this video and want to see more

then "like" and subscribe to our channel

and be on the lookout for more videos just like this.

#### [Lesson 5 - Data Lakes and Object Storage](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/407a9f86565c44189740699636b4fb85/401346823cb34394b6a32de0c1f2f2a1/)

## Lesson Introduction - Data Lakes and Object Storage

This lesson contains video sessions, as follows:

- **What is a data lake?** explains what data lakes are and how you use them.  It also provides insights into the kind of things you ought to be thinking about as you set up a data lake to power your applications and create more intelligent user experiences.

- **Data lakes in the cloud** covers the persistence of data and the data itself in the data lake that is persisted in object storage.

- **Using SQL to combine databases and object storage** shows how you can surround your data warehouse with recent technology that is especially useful in the cloud.

Cette leçon contient des sessions vidéo, comme suit :

- Qu'est-ce qu'un lac de données ? explique ce que sont les lacs de données et comment les utiliser. Elle donne également un aperçu du genre de choses auxquelles vous devriez penser lorsque vous créez un lac de données pour alimenter vos applications et créer des expériences utilisateur plus intelligentes.

- Les lacs de données dans le nuage couvrent la persistance des données et les données elles-mêmes dans le lac de données qui est persistant dans le stockage des objets.

- L'utilisation de SQL pour combiner les bases de données et le stockage d'objets montre comment vous pouvez entourer votre entrepôt de données d'une technologie récente particulièrement utile dans le nuage.

## Data Lakes and Object Storage

### What is a Data Lake?

Hi everyone, my name's Adam Kocoloski with IBM Cloud and I'm here to talk to you today

about data lakes, what they are, how you use one, and the kind of things you ought to be

thinking about as you set one up to power your applications and create more intelligent

experiences for users.

So data lakes exist because we're all awash with data and we've got systems of record,

we've got systems of engagement, we've got streaming data, we've got batch data internal

external data, and it's really a combination of these different kinds of data sources that

leads us to get powerful insights about what our users are doing about the way the world

is working around us and leads us to develop more intelligent applications.

Data Lakes start by collecting all those different types of data sources through a common ingestion

framework and that ingestion framework is something that typically wants to be able

to support a diverse array of different types of data, and it wants to kind of standardize

and centralize all that stuff into a common storage repository.

That's not always required but typically you don't want to be analyzing the source data

directly you want to be able to take a copy of it so that you've got the flexibility to

do the kind of things you need to do with that data.

And speaking of that, the data typically doesn't common a form where you can use it right out

of the box.

There's a lot of data cleansing and data preparation that's required right.

There is often times the ability to, or the requirement to create new features, something

we call feature extraction, combinations of different types of data that need to be pulled

together in order to create the right sort of bits of information to analyze.

And once you sort of cleanse that data, prep the data, model the right kind of features

for your analysis, then you get to the fun part which is actually going in and doing

the machine learning model training and doing your advanced analytics.

And each of these steps is typically creating new derived data sources doing data sets that

tie back to the original one.

And that relationship is a really important thing to capture, because let's say there

was a problem with one of your data sources, you know there was a correction that needed

to be made.

You need to understand how that flows through the entire pipeline of more refined data sets

and models that you're producing so that you can go back and correct it.

And that's what this governance stuff comes into play.

This is something that's really you know infused at every step of the journey.

It means collecting meta data, you know data about your data, you know the right kinds

of information about the tables in your data sets and how they relate to one another.

It means being able to enforce policies so that as an organization we use the data the

way it's meant to be used, the way it's intended to be used, the way it's acceptable to be

used to drive the business forward.

That's really something that can't be bolted on after the fact that something has to be

present throughout the entire life cycle.

If we stop here, we haven't really changed anything.

It's only by getting these insights that were producing in this data lake back out into

the real world that were able to you know deliver on the business promise of these data

lakes that that we're all investing in and that's where this apply step comes in.

This can take a few different forms.

You might be you know building simply dashboards That are helping business executives make

smarter decisions about where to take the business forward with new projects to invest

in.

Or you might be building smarter applications that are able to make intelligent recommendations

to the users of those apps based on you know historical purchased data.

Increasingly we're also seeing a lot of process automation where an intelligent model can

smooth over some typically manual business processes and create a more intelligent experience

and based on the sort of rich data driven understanding of the problem at hand.

And really this whole process iterates back, right.

Those more intelligent applications, they end up generating new data and the cycle continues.

And so that in a nutshell at a very high level is what a day lake does.

Some of you may have heard us talk about "the ladder to AI", the "AI ladder", and we talk

about that - we talk about collecting data.

We talk about organizing data.

We talk about analyzing.

And we talk about infusing.

And really those four steps on this ladder are things that you can see represented throughout

this data lake environment.

Clearly over here we're doing a lot of collection of these individual sources of data.

This data preparation and feature extraction step into governed fashion is absolutely what

we mean by the organizing of data.

ML model training is a key example of data analysis.

And we talk about infusing the insights from the data lake into the applications, that's

really this last step here.

And so, there is very much a clear linkage between climbing this AI ladder and a data

lake as a vehicle that can help you make that journey.

Thanks for watching.

If you have any questions or comments, please drop us a line below.

If you enjoyed this content, please consider liking or subscribing thank you.

### Data Lakes in the Cloud

Hello, this is Torsten Steinbach, an architect at IBM for Data and Analytics in the cloud

and I'm going to talk to you about data lakes in the cloud.

The center of a date a lake in the cloud is the data persistency itself.

So, we talk about persistency of data, and the data itself in the data lake in the cloud

is persisted in object storage.

But we don't just persist the data itself, we also persist information about the data,

which is on one side about indexes.

So, we need to index the data so that we can make use of this data in the cloud, data lake

efficiently.

And we also need to store metadata about the data in the catalog.

So, this is our persistency of the data lake and now the question is how do we get this

data into the data lake?

So, there are different types of data that we can ingest, so we need to talk about ingestion

of data, and we can have a situation that some of your data that is already persistent

in databases.

So, these can be relational databases and can also be other operational databases, no

sequel database and so on.

And then we get this data into a data lake.

There are actually two fundamental economic mechanisms.

One is basically an ETL, which stands for extra contractual transform load and this

is done in a bench fashion.

In a typical mechanism to do ETL is using sequel and since we're talking about cloud

data lakes as a sequel as a service now.

But there's also an addition if you combine those things, the mechanism of replication

which is basically more of the change feeds or after you may have batched ETL the initial

data set we talk about will be replicated with all of the changes that come in after

this initial batch.

Next, we may have data that is not persistent yet at all which is generated as we are speaking

here for instance from devices.

So, we may have things like IoT devices driving cars and the like.

And they are actually producing a lot of IoT messages all the time continuously and they

also need to basically stream in to the date lake.

So, here we're talking about streaming mechanism.

In a very similar manner, we are taught that we have data that is originated from applications

that are running in the cloud or services that are used by your applications.

They're all producing logs and that's very valuable information, especially if you're

talking about operational optimizations and getting business insights of your user behavior

for these kind of things.

This is very important data that we need to get hold of.

So, locks also need a streaming mechanism to basically get streamed and stored in objects

storage.

And finally, you may have a situation that you do already have data sitting around in

local discs.

So, you may have local discs maybe on your own machine.

You may have even a local data lake, a classical data lake, not in the cloud and typically

these are hadoop clusters that you have on premise in your enterprise, or it can be as

simple as used very frequently just as local shares that are used in your team and your

enterprise to store certain data.

And if you want to basically get them to a data lake you also need a mechanism, and it's

basically an upload mechanism.

So, a data lake needs to provide you an efficient mechanism to upload data from ground to cloud,

this means from on premises into object storage in cloud

Now, the next thing we need to do when a bunch of data is here is process it.

This is especially important if you're talking about data that hasn't gone through an initial

processing, like for instance device data, application data, this is pretty raw data

that has a very raw format that is very volatile that has very different structures changing

schema and sometimes it doesn't have a real structure which can be binary data, let's

say images that are being taken by a device's cameras and I need to extract features from

it.

So, we're talking about feature extraction from this data to this data.

But even if you already have no structure extracted it might still need a lot of cleansing,

you may have to basically normalize it to certain units, you may have to load it up

to certain time boundaries to get rid of null values in these kind of things.

So, there's a lot of things that you need to do about transformation, you need to transform

the data.

Once you have transformed the data, basically you now have the data that you can potentially

now use for other analytics but one additional thing is advisable that you should do with

this data is you could create an index for this data so we will know more about the data

and can get proficient, performance analytics.

And finally, you should also leverage this data and need to tell a data lake this by

cataloging the data.

So, there are multiple steps that often when we talk about the pipeline of data transformations

that need to be done here.

Now the question is what do we use here?

And there are actually two processes, two mechanisms, two services or types of services

that are especially suited for this type of processing.

One is function as a service and the other one is sequel as a service again.

So, with SQL and function as a service you can do this whole range of things here, you

can basically create indexes through SQL DDLs, it also can create tables through SQL DDLs,

you can transform data when you can use functions with custom libraries and custom code to do

future extractions from the format of the data that you need to process.

Once we have gone through this pipeline the question is what's next now?

So, we have prepared, we have processed all of this data and we have probably cataloged

it, so we know of what data we have.

Now it comes to the point that we really harvest all of this work by basically generating insights.

So, generating insights is on one side the whole group of business intelligence which

consists of things like doing reporting for creating dashboards

and that's what's typically often referred to as BI.

And one option that is possible now is to simply directly do basically BI against this

data in a data lake.

But actually, it turns out that this is especially useful for an option for batch ETL options

like creating reports in a batch function.

Because when it comes to more interactive requirements you need basically sitting in

front of the screen and you need to refresh it in a sub second.

Let's see dashboard here, there is actually another very important mechanism that is very

well established and it is part of this whole data lake ecosystem and this is a data warehouse.

So, data warehouse or a database, maybe more and more general, is highly optimized and

has a lot of mechanisms for giving you low latency and also guaranteed response times

for your queries.

So, the question is how do that?

Now, we obviously need to move this data one step further after it has gone through all

of the data preparation in the data lake with an ETL again.

And it happens to be again that SQL as a service is a useful mechanism because we already use

it to ETL data into the data lake.

Now we can also use it to ETL data out of this data lake into a data warehouse so that

it's now in this - I would say more traditional, establish - stack off doing BI that can be

used by your BI tools, reporting tools, dashboarding tools to do interactive BI with performance

and response time SLAs.

So, that's one end to end flow now, but very oppositely insights there is more than just

doing reporting and dashboarding right.

So, there's a whole domain of tools and frameworks out there for more advanced types of analytics

such as machine learning, or simply using data signs tools framework that now you basically

do also analytics and artificial intelligence against the data that we've prepared here

in the catalog.

And machine learning tools and data science tools, basically they all have very strong

support for accessing data in an object storage.

So that's why this is a good fit basically let them connect directly here to this data

lake.

Now, that is the end-to-end process basically getting from your data with the help of a

data lake into insights.

One of the big problems that is there today is for people to do that to prove and explain

how they got to his insight?

How can you trust this insight?

How can you reproduce this insight?

So, one of the key things that need to be part of this picture is data governance.

So, data governance in this context has two main things that we need to take care of.

One is we need to be able to track the lineage of your data because you've seen the data

is traveling from different sources, from preparation into some insights in the form

of a report.

And you need to be able to track back where did this report come from, why is it looking

like this, what's the data that basically produced it?

And the other things are you need to be able to enforce what a data lake can actually be

able to enforce, policies, governance policies.

Who is able to access what, who is able to see personal information and can I access

it directly or only in an anonymized masked form.

So, these are all governance rules, and there are governance services available also in

the cloud that basically a data lake needs to apply with and use in order to track all

of this.

So, we almost done with this overall Data Lake introduction, but there is just one more

thing that I want to highlight and this is since we're talking about in the cloud how

can I deployed my entire pipeline of traveling data for this whole infrastructure and how

can automate that.

And here basically function as a service places a special role because function as a service

has a lot of mechanisms that can that I can use to schedule and automate things like for

instance batch ETL step, like basically generating a report.

So, this is the final thing that we need in our data lake in order to automate and operationalize

my entire data and analytics using a data lake.

Thank you very much for your attention this has been my introduction into the data lake.

### Using SQL to Combine Databases and Object Storage

Hello, this is Torsten Steinbach, Architect at IBM for Data and Analytics in the Cloud

and I'm going to talk to you today about the SQL sandwich.

This is actually about databases and object storage.

So, as you can see in the image, we're putting it together as a nice and tasty sandwich.

So, at the center of the sandwich we have our data warehouse.

So, this is a very, well established mechanism of storing and analyzing data, big data but

we're surrounding it now with more recent technology and it is especially popular in

the cloud and object storage.

We have it as the bonds basically for a sandwich at the top and at the bottom.

Now, what's the purpose of having all of these components?

What we have actually at the top is our initial place where we are landing and storing all

of our raw data.

So, this might be things like log messages from applications, IoT messages from devices

that are just coming in and in a pretty raw format, we're able to just store them in the

object storage for very little money in a highly elastic manner.

And in the data warehouse however we want to have high quality data.

Because the data warehouse is a much more sophisticated but also much more expensive

component to operate a host and to purchase.

And at the bottom again, why do we have another object storage at the bottom?

Well, this is where we have our archived data.

Now, all of these components basically makes sense in an end-to-end big data analytics

use case because we have new data arriving and being stored in the object storage, and

here in object storage basically we are going to explore the data.

So, we try to find out what's actually in the data, and we prep the data, we prepare

the data, cleanse it, make it higher quality and more curated and we also conduct to some

extent batch analytics directly at the data that is stored in the object storage.

In a data warehouse however we are basically able to do interactive analytics that requires

certain SLAs for latencies of poor performance response times and so on.

And why are you having this archived data again, well this is basically because you

do not want to keep all of the data for years around in a data warehouse because as I told

you this is the more expensive thing to run with all of these components.

So, it makes sense for the data that is not hot anymore and is not required for your daily

business to archive it off again into an object storage.

So, for instance you can still run things like compliance reports that you're required

to do to be compliant with certain regulations.

So, these are the different types of analytics and some things that you want to do with the

data at the different stages of its life cycle basically.

Of course, a real sandwich should not be this dry so you will have some sauce in there.

So, what is the source basically that glues these things together.

It is basically an ETL mechanism.

It allows you basically to read data from here and transform it so that it can be read

here and same from here to here.

So, basically data is traveling this way.

Now finally you see a sandwich is often something that you do not eat into pieces right, you

do not just eat the patty or just the bun.

You eat it as a whole thing.

So, we consume it as a whole thing and it's also the same analogy that holds here that

while it makes sense to prepare a system that can serve these different types of workloads

in the most efficient way and cost effective way possible, it's of course hard if you always

have to think about do I have to go here, here, or here for this query.

It should be automatically figured out by the system and for that reason we are putting

this all into a nice box, putting it all together.

And we are using federation as a mechanism on top of all of that basically virtualizes

the location of the data depending on its age or state off the pipeline that it is in.

Now finally it is called a SQL sandwich, why is it called a SQL sandwich?

Well because SQL is the essential thing for all of these things that you can see here.

SQL is used for basically doing these analytics, these are SQL queries that we are running

here.

SQL is also used to do detail.

And finally, SQL is also used in order to federate those things together, It's a SQL

federation.

Okay, that's our SQL sandwich and I hope this helps you to put in perspective the roles

of object storage and data warehouses and how these two things can be put together into

this nice and tasty format so that you can get the most out of the technology in combination.

Thank you very much.

#### [Lab - Getting to the Cloud](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/407a9f86565c44189740699636b4fb85/d82ba5edac4f40efa334fff96b944b34/)

# Lab Instructions

To gain hands-on experience with the concepts mentioned in this section, click the following tutorial link:

* + Build a data lake using object storage  
    <https://cloud.ibm.com/docs/solution-tutorials?topic=solution-tutorials-smart-data-lake>

## Module Summary

# Summary

#### In this lesson you learned about:

* + Creating cloud-native applications.
  + Eliminating capital expenses by moving VMs to the cloud.
  + Migrating to the cloud using the lift-and-shift strategy.
  + Making decisions about data storage for cloud applications.

Résumé

Dans cette leçon, vous avez appris :

* La création d'applications natives du nuage.
* Éliminer les dépenses d'investissement en déplaçant les VM vers le cloud.
* Migrer vers le cloud en utilisant la stratégie du "lift-and-shift".
* Prendre des décisions concernant le stockage des données pour les applications dans le nuage.

# For more information

For additional information, you may be interested in the following materials:

* Cloud computing: A complete guide  
  <http://ibm.com/cloud/learn/cloud-computing>
* Getting started with VMWare solutions   
  <https://cloud.ibm.com/docs/vmwaresolutions>
* Cloud databases on IBM Cloud  
  <https://www.ibm.com/cloud/databases>
* IBM Cloud Learn Hub: Databases  
  <https://www.ibm.com/cloud/learn/database>
* IBM Cloud Catalog: Databases  
  <https://cloud.ibm.com/catalog?category=databases>

### Module 2 - Application Modernization

### Learning Objectives

In this lesson you will learn about:

* + - Breaking apart application monoliths into smaller packages suggested by the programming model.
    - Re-architecting applications into containerized microservices that communicate through a service mesh.
    - Exposing critical elements of applications as an API for other applications to use.
    - Making code changes and adding new value and capabilities to workloads.

Dans cette leçon, vous en apprendrez plus sur :

* Décomposer les monolithes d'application en petits paquets suggérés par le modèle de programmation.
* La ré-architecture des applications en micro-services conteneurisés qui communiquent à travers un maillage de services.
* Exposer les éléments critiques des applications en tant qu'API pour que d'autres applications puissent les utiliser.
* Apporter des modifications au code et ajouter de la valeur et des capacités nouvelles aux charges de travail.

#### [Lesson 1 - Preparation for Modernization](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/76d637cbe8024e509dc445df847e6c3a/2d204ac4fa3143048a998da7e53702d7/)

## Lesson Introduction - Application Modernization Preparation

This lesson contains video sessions, as follows:

- **Application Modernization**discusses three types of transformations that are being adopted by enterprises.

- **Client Success Story**tells how a global bank overcame obstacles to deliver a secure, scalable, reliable cloud application that is used by all tellers at all branches.

- **Analyze Application Readiness** introduces a tool to analyze existing applications and display information about the readiness for moving to the cloud.

Cette leçon contient des sessions vidéo, comme suit :

- La modernisation des applications aborde trois types de transformations qui sont adoptées par les entreprises.

- Client Success Story raconte comment une banque mondiale a surmonté des obstacles pour fournir une application en nuage sécurisée, évolutive et fiable, utilisée par tous les guichetiers de toutes les succursales.

- Analyser l'état de préparation des applications présente un outil permettant d'analyser les applications existantes et d'afficher des informations sur l'état de préparation pour le passage au cloud.

## Application Modernization Preparation

### Application Modernization

Hi, I'm Eric Minick with IBM Cloud,

and I want to talk about application modernization

and 3 huge transformations

that have been going on together.

So, we've got 3 things going on. They're interrelated.

This is what we're seeing:

this change in how we're doing architecture, infrastructure,

and our ways of working - how we deliver.

And if we go back in time a little bit,

we saw applications that were very monolithic,

they were running on physical servers,

and we used waterfall-style development,

where we'd have long plans and we'd say

"OK, this is going to be our planning phase, or development phase, or testing phase",

and we could plan out a year as a project.

And that's really what we've gone away from.

So, if we look at how most organizations are working today,

architecturally, they've got some sort of distributed architecture.

It's usually related to like a Service-Oriented Architecture (SOA),

the big buzz words from a few years ago,

but some sort of distributed architecture.

We have a bunch of web services,

they're talking to each other. We've got some databases on the back end

and then some front ends that kind of go through all that.

On an infrastructure level,

they're running on some sort of a virtual machine, right?

So, we said,

"We could probably do better than having to order a new server

every time we have a new service".

"Let's virtualize this stuff."

"And we need a little more density along the way."

And, from a way of working: agile development.

Pretty normal.

And then trying to figure out a little bit of what happens downstream.

So, this takes us up to where a lot of teams are today,

but not really where they're going.

So, if we look at the next phase:

we're taking another pass at this service-oriented architecture

and really shrinking the sizes of the services,

taking advantage of the more dynamic infrastructure we have.

And we're calling these now "microservices", right?

Microservices.

So, we had a microservice architecture.

So, very small, very focused services,

moving away from a lot of the heavyweight

XML-based communication we saw on SOA

towards more REST-based communication, things like that.

But, same idea: let's keep breaking into smaller and smaller pieces,

we have more independence of what we ship,

more rigor in saying:

"this service needs to be independent from another service

so I can change these things by themselves."

On the infrastructure side: Cloud.

Cloud is pretty popular.

This could be public cloud.

This could also be private cloud.

I am painting with a very broad brush when I say "cloud" here.

And then, from a delivery and a way of working,

we could say that DevOps is really key

- and I would include in this approaches like Site Reliability Engineering (SRE),

- more the ways of working that we have today.

Now, that's fine and interesting,

but what do these things have to do with each other?

And I'd argue that what we're really seeing is

modernization and how the applications are delivered

and how they're built.

What they are.

And, while you could walk into any large enterprise today

and you'll find someone who says:

"We are going through a cloud transformation."

You'll often find someone who says:

"Yes. I'm in charge of leading the DevOps transformation".

And you'll walk into an enterprise architecture and say:

"Yes. We are pushing microservice architectures."

Individuals think that they're going through

3 separate transformations, but they are really tied.

If I'm doing microservices,

and I have new microservices all the time,

- and in order to get a new microservice up and running,

I'm over here and I have to order a new physical server

and then rack and stack it a couple of months later,

I'm not going to get any time-to-market benefits.

The resilience benefits that I'm going to normally look for

from microservices are going to be modest at best.

Microservices want cloud infrastructure.

You want to be able to say:

"I've got a new microservice."

"Let me put it in a container and just run that container right now,

and scale that dynamically."

Similarly, cloud really likes running microservices.

The benefits of being able to dynamically scale are really cool

when you have a lot of small things that you might need few of,

or a lot of.

It's not as interesting when I've got a monolith

that isn't even distributed.

How do I scale that?

Do I get a bigger cloud server?

And then all of this is kind of baking-in this idea of speed and resiliency.

And DevOps brings that together, right?

The developers, who have always wanted speed.

The operations people,

who have always wanted that resiliency.

They're going to be programming that cloud, right?

The programmable infrastructure that cloud provides

needs operations people who understand resiliency,

but bring some of that development skill in.

And, to really take advantage of these new infrastructures,

the new architectures,

you need these new ways of working.

And you also are going to say:

"If this is going to give me time-to-market benefits,

I can't be back here"

and say: "Yeah, no, we got a one-year project plan that we're just going to execute."

I need to be able to be more agile and adapt

in my planning and my responsiveness to the business.

I need to better wire-up my applications,

so they can be more easily monitored and more resilient.

We have to have the application in a way

that it knows when one of these services is failing

and we can spin up another.

So, this is really, for me, fascinating

- that you walk into these organizations everywhere,

and they're undergoing these 3 different transformations.

But they're always doing them together

and when they don't, it doesn't quite work.

So, you've got these 3 transformations going on at once.

And you'll hear us talk a lot about "application modernization".

See that written across the top here?

And, when I think of application modernization,

I think it's just this.

It's this transformation right here.

Going from these monoliths, or service-oriented architectures,

to microservices, adopting cloud,

modernizing our ways of working towards DevOps and SRE.

That's "App Mod" (Application Modernization).

It's a really exciting time

and it's really great when you're able to go after it in a holistic fashion.

Thank you.

If you have questions, please drop us a line.

If you want to see more videos like this in the future,

be sure to like and subscribe.

### Client Success Story

We were asked to help a global bank customer

that had spent two years developing an application

meant to be used by their tellers across their bank branches.

That thousands of business users across hundreds of branches

worldwide, were seeing performance problems.

When the customer has a problem, even though it's in code that they wrote

that's still on top of our product

so we certainly jump in and help.

There were a lot of issues, and they escalated to our team, the SWAT Team

to come in and give an assist.

Digital Business Automation SWAT Team is a little like a real SWAT team

we're a quick response, reactive team that brings deep technical skills across all of our products

to solve the customer's issue as quickly as possible.

Any customer that's experiencing a difficult issue has access to the SWAT Team.

What we’re looking for is to really dig down into the problem

and get real, hands-on information to see

where we can help and what we can do to best help the customer.

You start when the customer starts, and you don't finish until the customer is done for the day.

So, it might be a 12-hour or a 16-hour day, it's whatever it takes.

A problem like this, it's important to look at the entirety of the application.

We want to look at the hardware it's running on

we want to look at the operating system they're using

the database, and how they've developed their UI.

And it was important for our team to come in and we were able to get them down to five seconds.

One of the great things about working in IBM Support is every customer situation brings a new problem.

And I know, based on our experience, that

we're going to solve that problem and we're going to get the customer to where they want to be.

### Analyze Application Readiness

hello I'm Scott Johnston an offering

manager for websphere application server

in this video I will demonstrate new

functionality introduced to the

traditional WebSphere application server

administrative console this new

functionality analyzes your enterprise

applications and shows you how ready

they are for running your WebSphere

Liberty a lightweight composable

production-ready application server for

both on-premises workloads and cloud

deployment to start login to the admin

console and navigate to applications

then to application type to get to your

web seer enterprise applications there

are six applications in this cell two of

which already have Liberty reports and

four that do not let's run scans against

the four applications which do not have

reports by selecting each of them and

then choosing the run scanner option in

the drop down list under the analyze

button a scan request gets submitted for

each of the applications in the

background the websphere migration

toolkit for application binaries scans

the application files and generates an

application migration report you can

click the refresh icon in the Liberty

report column header to get an updated

view of the scanners progress when the

scan of an application is complete a

summary of the results is displayed

within the enterprise applications

collection table applications that are

ready for websphere liberty of a green

circular icon with a checkmark

applications that might need some

updates before being deployed to Liberty

have one or more other icons when the

scanner detects api's that are not

available in Liberty or there are

behavioral changes that can break the

application they are labeled severe

issues in flag with a red circular icon

with a backslash mark if the scanner

detects api's or behavioral changes that

need further evaluation because they

might break the application they are

labeled warnings and flagged with a

yellow triangular icon with an

exclamation mark deprecated IP eyes or

minor behavioral changes they should not

affect most applications are labeled

informational and flagged with a blue

square icon with a lowercase I click the

report link associated with an

application to view the full application

migration report directly within the

admin console once opened you can

download the report for offline

evaluation by clicking the Save report

button at

top of the report is the technology

evaluation summary which indicates which

IBM platforms are good fit with the

technologies used by the application the

second part is the migration rules

severity summary this shows an overview

of how many rules were flagged and the

number of times each was encountered and

the last part is the application

migration details which is divided into

four sections the first section

technology evaluation provides a table

of technologies that were detected in

their matching IBM platforms

technologies that are not supported or

linked to analysis rules listed in the

second section the detailed migration

analysis where all the flagged rules are

listed along with the rules specific

health information the detailed

migration analysis even identifies the

file names and the line numbers where

the rule infractions were detected the

inventory section provides a high-level

view of the content and structure of the

application

the final section the Liberty feature

list contains a list of all the required

features for the application in the

Liberty server XML configuration file

the application scanner can be

configured to include and exclude

specific packages from its analysis

select the configure scanner option to

access the configuration panel when

package names are provided in the

include packages field only the

specified packages will be analyzed or

leave it empty to scan all packages the

scanner is configured to exclude Java EE

and some third-party packages by default

additionally you can select the target

version of Java SE that you expect to

use as the runtime for websphere liberty

the default is set to IBM Java 8 but

options are available for other versions

of Java also if you're running the

scanner on an extremely large

application then you might want to

specify the max heap size of memory that

gets allocated to the scanners JVM

lastly you can delete reports by

selecting one or more applications that

already have reports and then choosing

the delete report option also note that

when applications are either uninstalled

or updated the report also gets deleted

so there's no mismatch between the

information in the report and the

version of the application which is

deployed in the admin console for

instance after the issues that were

identified by the analysis get fixed and

the application is redeployed the old

report gets deleted simply run the

scanner

again to get a new report for the

application in this example the issues

and warnings that were found in the

sample default application were resolved

and it is now Liberty ready thank you

for joining me for this demonstration of

the new Liberty readiness application

analysis functionality that's now

available in the websphere application

server administrative console

[Music]

#### [Lesson 2 - Containerization](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/76d637cbe8024e509dc445df847e6c3a/dc5181709f164e49bfa9490292612bc7/)

## Lesson Introduction - Containerization

This lesson contains video sessions, as follows:

- **Containerization Explained** describes how containers provide a standardized way to package and ship software and can be used to deploy and run distributed applications without launching an entire virtual machine.

- **Container Orchestration Explained** highlights the benefits of container orchestration and shows why container orchestration was necessary in the first place.

Cette leçon contient des sessions vidéo, comme suit :

- Containerization Explained décrit comment les conteneurs fournissent une méthode standardisée pour emballer et expédier des logiciels et peuvent être utilisés pour déployer et exécuter des applications distribuées sans lancer une machine virtuelle entière.

- Container Orchestration Explained met en évidence les avantages de l'orchestration des conteneurs et montre pourquoi l'orchestration des conteneurs était nécessaire au départ.

## Containerization

### Containerization Explained

Hi everyone, my name is Sai Vennam,

and I'm a developer advocate with IBM.

Today, I want to talk about containerization.

Whenever I mention containers,

most people tend to default to something like Docker

- or even Kubernetes these days,

but container technology has actually been around for quite some time.

It was actually back in 2008

that the Linux kernel introduced C-groups, or "control groups",

which paved the way for all the different container technologies we see today.

So, that includes Docker, but also things like Cloud Foundry,

as well as Rocket and other container runtimes out there.

Let's get started with an example.

We'll say that I'm a developer,

and I've created a Node.js application,

and I want to push it into production.

We'll take 2 different form factors to kind of explain

the advantages of containerization.

First, we'll talk about VMs

and then we'll talk about containers.

So, first things first,

let's introduce some of the things that we've got here.

So, we've got the hardware itself,

which is the big box.

We've got the host operating system,

as well as a hypervisor.

The hypervisor is actually what allows us to spin up VMs.

Let's take a look at the shared pool of resources

with the host OS and hypervisor.

We can assume that some of these resources have already been consumed.

Next, let's go ahead and take this .js application and push it in.

And to do that, I need a Linux VM.

So, let's go ahead and sketch out that Linux VM.

And in this VM, there's a few things to note here.

So, we've got another operating system in addition to the host OS

it's going to be the guest OS.

As well as some binaries and libraries.

So, that's one of the things about Linux VMs:

even though we're working with the really lightweight application,

to create that Linux VM,

we have to put that guest OS in there and a set of binaries and libraries.

And so, that really bloats it out.

In fact, I think the smallest Node.js VM that I've seen out there is over 400 MB.

Whereas the Node.js runtime app itself would be under 15.

So, we've got that,

and we go ahead and let's push that .js application into it,

and just by doing that alone,

we're going to consume a set of resources.

Next, let's think about scaling this out.

So, we'll create 2 additional copies of it.

And you'll notice that even though it's the exact same application,

we have to use and deploy that separate guest OS and libraries every time.

And so, we'll do that 3 times.

By doing that,

essentially, we can assume that for this particular hardware,

we've consumed all of the resources.

There's another thing that I haven't mentioned here:

this .js application, I developed on my MacBook.

So, when I pushed it into production to get it going in the VM,

I noticed that there were some issues and incompatibilities.

This is the kind of foundation for a big "he said, she said" issue,

where things might be working on your local machine

and work great, but when you try to push it into production,

things start to break.

This really gets in the way of doing Agile DevOps and continuous integration and delivery.

That's solved when you use something like containers.

There's a 3-step process when doing anything container-related

and pushing or creating containers.

And, it almost always starts with, first, some sort of a manifest.

So, something that describes the container itself.

So, in the Docker world, this would be something like a Dockerfile,

and in Cloud Foundry, this would be a manifest YAML.

Next, what you'll do is create the actual image itself.

So, for the image,

if you're working with something like Docker,

that would be a Docker image.

If you're working with Rocket, it would be an ACI (or, "Application Container Image").

So, regardless of the different containerization technologies,

this process stays the same.

The last thing you end up with is an actual container itself,

which contains all of the runtimes and libraries and binaries needed

to run an application.

That application runs on a very similar setup to the VMs,

but what we've got on this side is,

again, a host operating system.

The difference here is, instead of a hypervisor,

we're going to have something like a runtime engine.

So, if you're using Docker, this would be the Docker Engine,

and, you know, different containerization technologies would have a different engine.

Regardless, it's something that runs those containers.

Again, we've got this shared pool of resources;

so we can assume that that alone consumes some set of resources.

Next, let's think about actually containerizing this technology.

So, we talked about the 3-step process:

we create a Dockerfile,

we build out the image,

we push it to a registry, and we have our container,

and we can start pushing this out as containers.

The great thing is, these going to be much more lightweight.

So, deploying out multiple containers

- since you don't have to worry about a guest OS this time,

you really just have the libraries as well as the application itself.

So, we scale that out 3 times,

and because we don't have to duplicate all of those

operating system dependencies and create bloated VMs,

we actually will use less resources.

So, let's use a different color here.

And, scaling that out 3 times,

we still have a good amount of resources left.

Next, let's say that my coworker decides,

"Hey for this .js application, let's take advantage of a third-party,

let's say, a cognitive API to do something

like image recognition."

So, let's say that we've got our third-party service,

and we want to access that using maybe a Python application.

So, he's created that service that accesses the third-party APIs,

and with our Node.js application, we want to access that Python application

to then access that service.

If we wanted to do this in VMs, I'm really tempted to basically create a VM

out of both the .js application

and the Python application

because that would allow me to continue to use the VMs that I have.

But that's not truly cloud-native, right?

Because if I wanted to scale out the .js but not the Python app,

I wouldn't be able to if they're running in the same VM.

So, to do it in a truly cloud-native way,

I would have to free up some of these resources

- basically, get rid of one of these VMs,

and then deploy the Python application in it instead.

And, that's not ideal.

But with the container-based approach, what we can do is simply say,

since we're modular, we can say,

"OK, just deploy one copy of the Python application".

So, we'll go ahead and do that in a different color here.

And that consumes a little bit more resources.

Then, with those remaining resources,

the great thing about container technology,

that actually becomes shared between all the processes running.

In fact, another advantage:

if these container processes aren't actually utilizing the CPU or memory,

all of those shared resources become accessible

for the other containers running within that hardware.

So, with container-based technology,

we can truly take advantage of cloud-native-based architectures.

So, we talked about things like portability of the containers;

talked about how it's easier to scale them out;

and then, overall, with this three-step process and the way we push containers,

it allows for more Agile DevOps and continuous integration and delivery.

Thanks for tuning in for this broad overview of container-based technology.

As always, we're looking for feedback so definitely drop a comment below,

and be sure to subscribe to stay tuned for more videos in the future.

Thank you.

### Container Orchestration Explained

Hi everyone, my name is Sai Vennam,

and I'm with the IBM Cloud team.

Today, we want to talk about container orchestration.

I know that in the past, we've talked about containerization technology

- as well as dived into Kubernetes as an orchestration platform.

But, let's take a step back,

and talk about why container orchestration was necessary in the first place.

We'll start with an example.

Let's say that we've got 3 different microservices

that have already been containerized.

We've got the frontend,

we'll have the backend,

as well as a database access service.

These 3 services will be working together,

and are also exposed to end users,

so they can access that application.

The developer has a very focused look at this layout.

So, they're thinking about the end user,

the end user accessing that frontend application,

that frontend, which relies on the backend,

which may, in turn, store things using the database service.

The developer is focused entirely on this layer.

Underneath it, we've got an orchestration layer.

So, we can call that a master,

and I'm thinking about Kubernetes right now,

where you would have something like a master node

that manages the various applications

running on your computer resources.

But, again, a developer has a very singular focused look at this layout

and they're really only looking at this stack right here.

They're thinking about the specific containers

and what's happening within them.

Within those containers, there are a few key things.

So, there's going to be the application itself,

there's also going to be things like the operating system,

as well as dependencies.

And there are going to be a number of other things that you define,

but all of those things are contained within those containers themselves.

An operations team has a much larger view of the world.

They're looking at the entire stack.

So, an operations team:

there's a number of things that they need to focus on,

but we'll use this side to kind of explain how they work with

deploying an application that is made up of multiple services.

So, first, we'll talk about deploying.

So, taking a look here,

it's very similar to over here, but the key difference is

these are no longer containers,

but the actual computing resources.

This can be things like VMs (Virtual Machines)

or, in the Kubernetes world, we call these "worker nodes".

So, each one of these would be an actual

computing worker node.

So, you know, it could be something like

4 vCPUs (virtual CPUs) with 8 GB of RAM

per each one of these different boxes that we have laid out here.

The first thing you would use an orchestration platform to do

is something simple - just deploying an application.

Let's say that we start with a single node.

And, again, here we've got the master.

On that single node, we'll deploy 3 different microservices

- one instance each.

So, we'll start with the front end,

we'll have the backend,

as well as the database access service.

Already, let's assume that

we've consumed a good bit of the compute resources

that are available on that worker node.

So, we realize - let's add additional worker nodes to our master

and start scheduling out and scaling our application.

So, that's the next piece of the puzzle.

The next thing an orchestration platform cares about

is scaling an application out.

So, let's say that we want to scale out the frontend twice.

The backend, we'll scale it out 3 times.

And the database access service,

let's say we scale this one out 3 times as well.

An orchestration platform will schedule out our different

microservices and containers to make sure that

we utilize the computer resource in the best possible way.

One of the key things that an orchestration platform does is scheduling.

Next, we need to talk about

networking and how we enable

other people to access those services.

That's the third thing that we can do with an orchestration platform.

So, that includes creating things

like services that represent each of our individual containers.

The problem is: without having

something like an orchestration platform take care of this for you

- you would have to create your own load balancers.

In addition, you would have to manage your own services

and service discovery, as well.

So, by that, basically I mean

that if these services need to talk to one another,

they're not going to try to find the IP addresses of each different container

and resolve those and see if they're running.

That's something the orchestration platform needs to do

- is handle that system around it.

So, with this, we have the ability

to expose singular points of access

for each of those services.

And again, very similarly, an end user

might access that frontend application

- so the orchestration platform would expose that service to the world,

while keeping these services internal

- where the frontend can access the backend,

and the backend can access that database.

Let's say that that's the third thing

that an orchestration platform will do for you.

The last thing I want to highlight here is insight.

Insight is very important

when working with an application in production.

So, developers are focused on the applications themselves,

but let's say that one of these pods accidentally goes down.

What the orchestration platform will do

is it will rapidly bring up another one,

and bring it within the purview of that service.

It will do that for you automatically.

In addition, an orchestration platform has a number of pluggable points

where you can use key open source technologies

- things like Prometheus and Istio

- to plug in directly into the platform

and expose capabilities that let you do things like a logging,

analytics, and there's even a cool one,

something that I want to sketch out here,

- the ability to see the entire service mesh.

Many times, you might want to

lay out all of the different microservices that you have

and see how they communicate with one another.

In this example, it's fairly straightforward,

but let's go through the exercise anyway.

So, we've got our end user;

and the end user would likely be accessing the frontend application.

And, we've got the two other services as well:

the database, as well as the backend.

In this particular example, I'll admit,

we have a very simple service mesh

- we've only got three services.

But seeing at how they communicate with one another

can still be very valuable.

So, the user accesses the frontend,

the frontend accesses the backend,

and we expect the backend to access the database.

But, let's say the operations team finds that,

oh actually, sometimes the frontend

is directly accessing the database service.

They can see how often, as well.

With things like a service mesh,

you get insight into things like the operations per second.

Let's say that every time

- or let's say there are 5 operations per second hitting the frontend,

maybe 8 that go to the backend,

maybe 3 that go per second to the database service,

but then .5 requests per second

going from the frontend to the database service.

The operations team has identified,

by taking a look at the requests

and tracing them through the different services,

that here's where the issue is.

This is a simple example

about how you can use something out like Istio and Kiali

(which is a key service-meshing capability)

to gain insight into running services.

Orchestration platforms have a number of capabilities

that they need to support,

and this why operations teams

and these roles that we're seeing pop up

- things like SREs (Site Reliability Engineers)

- and we're seeing the growth of those roles because

there are a lot of things that they need to concern themselves with

when running an application in production.

Developers see a very singular view of the world,

where they're focusing on the things within the containers themselves.

Thanks for joining me for this quick overview

of container orchestration technology.

If you like this video please be sure to drop a comment below

or leave us any feedback and we'll get back to you.

Be sure to subscribe,

and stay tuned for more videos in the future.

Thank you.

#### [Lesson 3 - Refactoring](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/76d637cbe8024e509dc445df847e6c3a/164f711c50564bd1ade654e0c6b3e637/)

## Refactoring

now with this migration process that

we're talking about taking enterprise

applications over to the cloud you can

see with some of these options that were

proposing is that there's going to be

cases where it doesn't make sense to

take the application and rewrite it

often from scratch right there are going

to be times where you can do exactly

what we are showing here the

transformation advisor is to analyze

some key dependencies and take

applications that are ready to run in a

cloud and containerize them and move

them directly to a cloud and it looks a

bit like this this is the real platform

sort of a like a sample kind of scenario

and again we take like the before on the

left hand side you have a websphere

application server environment and the

app is running inside a websphere

deployed also in the data center you'll

have middleware db2 mq Oracle other

kinds of databases are available and

this process of just replac forming

takes that application and puts it into

a container with Liberty runtime and

then deploys that into a cloud and

kubernetes environment an application is

running it still has now injected

through its environment the credentials

and other resources to be able to go

back and talk to the deployed middleware

that's available and in this scenario

you'd have you know a containerized

environment running in a private cloud

and the application is talking to the

middleware and you know this is a very

effective and quick way to get cloud

scale in the enterprise but before we

talk a lot about these kinds of

fundamental concepts I think we should

explain them here so let's start by

talking about containers now probably

many of you have heard about containers

and a run docker on your machine or seen

or heard about people doing it but one

thing it's important to remember about

containers and docker

is that the containers and the

containerization and isolation of

processes is not a very new idea this

has been something that's been around

for a long time and I've got some

examples here of how things have evolved

over time and it's important to note

that what's really significant here with

docker and why docker containers have

become so so widely used in a part of a

growing ecosystem is the overall

ecosystem itself it's not just a

platform it's also an ecosystem and it's

an ecosystem built upon a large library

of available container images that are

from many different providers so and at

a fundamental level what's happening

here is that I am running a single

process and that single process is

surrounded by the assets that it needs

to run and that's a container image and

it's much more efficient way to deliver

applications than installing an

application on top of a full operating

system image it's running on top of a

hypervisor that's running on top of a VM

and cloud environment and let's see why

so what is going on is that a container

is isolation that's done at the process

level and when you look at and this is

not just a Linux phenomenon there are

other operating systems as well there's

windows containers as well think of it

this way is that at the platform level

in the kernel instead of having a

namespace that all of the processes on

that kernel can see I create a unique

namespace and and those namespaces cover

a number of different dimensions within

the kernel they'll cover a process ID

space they'll cover user ID space I'll

cover networking services naming

services IPC and also file system

supports right and so this allows a

single process to run in a way in which

it doesn't see anything else around it

and it's still within a host and that

host is up and running and this is one

of those key differences between VMs and

containers and that is that the

container is just a

single-process is nice lated space when

I want to start a container what I do is

I create a namespace and then I set up I

can use C groups to actually set limits

on what's going on with resources in

terms of CPU and memory and then I fire

up a single process in that process that

runs for the lifetime of the container

so comparing that to the ends there's a

few things that that make it rather

effective to consider in this cloud

model so on the left hand side I've got

a virtual machine environment and in

typical virtual machine environment I've

got my hardware on top of that I'll have

a hypervisor

and then I'll have multiple virtual

machines on one hypervisor

within that virtual machine I've got

this rather sort of monolithic image

that's composed of an operating system

I've got libraries and then I could have

one or more applications all run and

that's why I have an OVA file which

would have all of these you know items

on it in order for us to be able to

start an application I have to pick on

the hardware pick on the hypervisor

let's say that's already running right

for me to start an application for a VM

I've got to load it into a hypervisor

I've got to mount that source OVA then

I've got to boot that operating system

operating system has to come all the way

up all the environments come up at that

point I can start an application shift

over to the right-hand side the key

thing that's different here about the

container is that within the container I

am my my isolation layer you can think

about it's like going back into the

depth of the slide right so each

container he has got a thin and we'll

use the term layer and I'll talk a bit

more about later a thin layer of

resources that are needed by that

application there are probably some

operating specific elements that I need

maybe it's like a base operating system

it could be a full operating system like

Ubuntu but then there's also a container

image is very poplar called Alpine which

is a stripped-down version of an

operating system with the oils I might

see need some libraries or runtime so

for example Liberty runtime would fall

into category

these supporting pieces and then I'll

have my my application so when you think

of this starting up what will happen is

is that this image container image which

is composed of a number of layers to

start the application I would map these

image layers into a read/write and into

it into a stack it's sort of like I

think it'd like transposing a slide into

a file system and then that's mounted by

that process it's my out but all that

has to happen here in this case if I

already have these image layers sitting

on that base operating system in a

repository the only thing that I need to

do to be able to start up that

applications instead of starting an

operating system and all these other

components all I do is I start my my

library and my app like for example I

start Liberty and I start my application

and it's much less it has to start so

when I'm spinning up a container versus

spinning up a VM I have fewer things to

start and so I can very quickly

instantiate new applications or

horizontal application copies

#### [Lesson 4 - Hybrid Cloud Architecture](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/76d637cbe8024e509dc445df847e6c3a/15b3ccc4c6304b0595d7230a72773498/)

## Lesson Introduction - Hybrid Cloud Architecture

This lesson contains video sessions, as follows:

- **Multicloud versus hybrid cloud** discusses the differences between multicloud and hybrid cloud environments and explains some of the tools that can help you implement your cloud solution.

- **Hybrid Cloud Architecture Part 1: Connectivity** helps you develop your hybrid cloud strategy for connecting various parts of your environment including private cloud and public cloud.

- **Hybrid Cloud Architecture Part 2: Modernize**helps you develop your strategy for moving legacy or monolithic applications to the cloud.

- **Hybrid Cloud Architecture Part 3: Security** explains how to develop your strategy for increasing security and reducing network vulnerabilities.

Cette leçon contient des sessions vidéo, comme suit :

- Multicloud versus nuage hybride aborde les différences entre les environnements multicloud et nuage hybride et explique certains des outils qui peuvent vous aider à mettre en œuvre votre solution de nuage.

- Architecture de nuage hybride - Partie 1 : La connectivité vous aide à développer votre stratégie de nuage hybride pour connecter différentes parties de votre environnement, y compris le nuage privé et le nuage public.

- L'architecture de cloud hybride, partie 2 : la modernisation vous aide à développer votre stratégie pour déplacer les applications héritées ou monolithiques vers le cloud.

- Architecture de cloud hybride - Partie 3 : Sécurité explique comment développer votre stratégie pour accroître la sécurité et réduire les vulnérabilités du réseau.

## Hybrid Cloud Architecture

Let me ask you a few questions, but I can hit you first with an easy one:

What's the difference between multi-cloud and hybrid cloud?

It's a really good question, Dan.

So, hybrid cloud traditionally refers to

running some part of your workload on-premise

and some part in a public cloud.

But usually that means one and only one public cloud,

possibly tied or locked to that particular vendor.

Multi-cloud means having the flexibility to run in one or more different public clouds,

perhaps switching between them, depending on cost or other reasons.

Well, considering acquisitions are happening so much today,

you could have a multi-cloud simply by the very fact that you acquired someone, right?

Right.

Speaking of acquisitions, we've had one at IBM.

The big news for IBM was the acquisition of Red Hat.

How has that affected IBM's cloud strategy going forward?

I think the single biggest thing is

that weare leveraging the OpenShift Kubernetes platform

as a delivery vehicle for all of our software

via our IBM Cloud Pak strategy.

So, all of our Cloud Paks, including the Cloud Pak for Multicloud Management,

are delivered on top of OpenShift.

Our Cloud Pak for Applications actually allows customers

to build applications to run on top of OpenShift.

So, that's really the biggest piece.

We have lots of other bits where we are bringing components of Red Hat together

with both our management portfolio and other portfolios.

But the biggest is OpenShift, I'd say.

Red Hat is a big fan of open source, so is IBM.

It has open source tools like monitoring and management.

I know they're big fans like we mentioned,

but what role does open source play in the Cloud Pak for Multicloud Management?

We've seen what Red Hat's done with open source, as you mentioned, right?

They've done a great job of providing enterprise grade support

and updates around an open source set of communities.

IBM has had a similar relationship with open source

and we look to continue that with the Cloud Pak for Multicloud Management.

So, leveraging both the open source capabilities that come in from Red Hat

and open source communities such as

Kubernetes, that are outside of that,

and Prometheus and Grafana, ELK, other sorts of things like that;

we'll look to expand our open source involvement and really

follow Red Hat's lead there.

Really, they've had the lead in this particular space,

but have a whole big stack that is based upon open source

and then the enterprise support on top of it.

We're going to be working more and more with Red Hat as the years go on, right?

Arguably though, development shops have multiple clouds platforms -- it's the new norm.

Why wouldn't customers also use multiple vendor tools to manage and maintain their environments?

What we find is,

if you use multiple clouds, you end up with multiple tool sets from each of those individual clouds.

And if you stick with just those tool sets from the different cloud vendors,

you end up with multiple panes of glass and multiple places that need to go and check.

So, in the case of something like security,

it's really easy for something to slip through the cracks.

And your operationss team is going to be upset by that!

Absolutely!

So, having a single control plane

that sits in front of those cloud tools

really gives you that visibility

and governance end-to-end across your entire environment.

So, you have multiple cloud platforms and you have a single view on it.

And that doesn't eliminate the need for those tools on those multiple cloud providers.

You still may drill down to those.

But having some layer that's above that, that is that single control point,

really is critical.

Let's go down one level.

We know that Kubernetes manages containers.

VMware and OpenStack, manages VMs.

Does the Cloud Pak for Multicloud Management help manage these non-container workloads too?

One of the things that we found as we've discussed with clients since we've been on this multi

cloud management journey for the last couple of years,

is that while cloud native architectures with containers are the hot new thing,

it represents a pretty small percentage of

what customers have in their environments today.

Virtual machines make up the vast majority of that.

I hear something something like 80%.

It's probably even more than that, 80-90 percent.

And then there's some bare metal stuff, mainframes, et cetera.

The ability to have your multi-cloud management platform handle all of those resources,

even as your modernization journey might take you to more containers

in the next three, five, ten years,

is a really critical thing.

And it's a design point for us with the Cloud Pak for Multicloud Management.

Meet the client where they are today in terms of their infrastructure,

and be able to grow with them as they modernize

and maybe change that blend in the future.

They've made a huge investment; you're wanting to preserve that investment.

Absolutely.

And if you look outside of our Cloud Pak for Multicloud Management

to someplace like our Cloud Pak for Applications, we can also help them

modernize that application layer that's running on top of it as well.

So, we really are looking at the Cloud Pak for Multicloud Management

as helping them modernize that management layer,

while at the same time

the development teams are modernizing that applications layer.

Speaking of development teams,

you know we've talked a lot about the Operations teams

and we've talked about the Development team,

is the multi-cloud compliance and security, is that a concern for development,

for operations, or both of them have to worry about this now?

I think the answer is both,

but I think traditionally the answer has been

it's primarily been an operations problem

and security has been something that has been a bolt-on or an afterthought

once you've pushed the application into production.

Or the developer has to worry about it.

Well, when they have to fix the bugs!

Of course, right?

You're seeing an emerging practice in the industry called DevSecOps,

which really, if you think about it, brings that together:

development, security, operations.

And really what that means is

you're injecting security and policy and compliance practices

into the development lifecycle from the beginning.

It's just an extension of DevOps and Agile practices.

So, DevOps would have you inject things like monitoring from day one.

DevSecOps says, "Well, in addition to adding monitoring

and management from day one, you need to add security too."

So, those two teams have to work together.

Absolutely, absolutely.

And again, I think

you'll probably see a new term come out at some point, which is to have a security-focused SRE.

Because again, that's very similar, those practices and bringing those things together

and not have them be separate silos.

So, how does the multicloud Cloud Pak for Multicloud Management

make it easier to enforce those policies

ensuring that safe and compliant environment?

I mean, that's about making the operations team more efficient.

Absolutely, right.

And also providing something that the development team can develop to them.

So, we've picked up some concepts from Kubernetes,

and that's the notion of a declarative model

where you write out, in code or in a text document,

what you want the compliance to be in the form of a policy.

That's very Kubernetes like, isn't it?

And the system will enforce that for you.

So, what that means is

our developers can write that as a piece of code with their application.

They can check it into source control,

it can go over to the operations team,

the operations team can pull that out and deploy it into production.

So, it's now completely a part of that DevSecOps process

because you have that declarative model.

Again, building off of what Kubernetes has done,

but really stretching that out to those non-container resources as well.

So, if I'm adopting this cloud model here,

how does security and compliance become critical to that deployment success?

Well, I would say

you want to start right away with security when you adopt.

It's a critical component.

I would say that traditionally speaking,

we've always had it as at the end of the process.

Once we get to production, let's worry about,

let's put intrusion detection and vulnerability detection

and all that stuff on it.

There's trip wire systems on the production environment.

But what we're realizing is,

particularlyin this world where we're bringing in more

open source components and using public cloud services,

that you really need to be,

as soon as you start developing these components,

you need to start worrying about security of the system,

checking the security of the system,

even if it's in your development builds...

Like with the microservices, won't it be a component that you can then put a wrapper around?

Absolutely. It absolutely is.

And that microservice, you might develop and think it's only going to run on premise,

but then the company may decide, "Oh, I'm going to run it on a public cloud".

You have that flexibility.

But you still need to have that ability to have security,

which is why you need to be thinking about that

from when you first start writing that design document, right?

Even before you even start writing code,

security needs to be part of that process.

You can test it independently, component by component.

Yes, absolutely.

Got it.

So, can you talk about how IBM Cloud Pak for Management

integrates with a customer's existing tools and processes,

whether it's from IBM or another vendor?

When we walk into a client site,

they've got management in place;

they're running applications,

they're running infrastructure, their business is going on, right?

Cause there's, there's no green fields in IT where nobody's doing anything yet.

There's stuff that's running, and they have to manage it somehow.

So, these can be things like,

everyone's got a service desk somewhere, something like a ServiceNow

where they're getting tickets in.

Often there's some level of monitoring and log management that exists in that environment,

whether it's an open source tool or a third-party tool or an IBM tool to solve that problem.

We can integrate those into the Cloud Pak for Multicloud Management

so that single control plane can get events and data and incidents

from all of those different tools.

And you have the flexibility to choose,

in some cases, the IBM tool or stick with your existing tool,

in the case that there might be an overlap between those two things.

We really don't want to present the Cloud Pak for Multicloud Management as

a monolith that you must consume the whole thing.

It's really a set of modular capabilities

that are meant to plug in with what you already have

and build that control plane between everything.

Well, thanks Matt for taking the time to talk with us,

we look forward to hearing more from you in our next presentation.

Absolutely, thank you so much.

### Hybrid Cloud Architecture Part 1: Connectivity

Hi everyone, my name is Sai Vennam, and I'm a developer advocate with IBM.

Today, I want to start the first part of the Hybrid Cloud Architecture series

with a deep dive into Connectivity.

Connectivity is an important concern when you're starting with your hybrid cloud strategy,

and that's why I want to start with it first.

By establishing connectivity, we can then start thinking about

other requirements and then move on to the other parts of the series.

There are 3 major parts about connectivity that I want to hit with this video

- starting with, very simply,

how do you actually connect private and public cloud environments?

Next, I'll be moving on to the service mesh

- essentially enabling your applications and microservices

to work with one another as one singular mesh.

And, we'll close it off by talking about some integration tools

that we have available to make it easier

to connect up your services and third-party services and that kind of thing.

To better explain and set the stage for the rest of the topics,

I want to introduce a Stock Trader sample application

that we'll be kind of revisiting with this architecture.

So, let's get started.

Over here, we have a consumer application - whether it's a mobile app,

a web browser, whatever it might be

- and whenever a user accesses the Stock Trader application,

they'll be hitting the private cloud endpoint.

At this point, they'll be fed into the Kubernetes cluster that we have here.

And within this Kubernetes cluster,

we have a number of services. The first service that they're going to hit

is the Trader. So this will be the front end of the application. So there will be kind

of an exposure from the Trader to the outside of that cluster.

The Trader, in turn, goes and creates Portfolios. So this, essentially, is the reason why people

use Stock Trader - to create these portfolios to manage their investments and their trades

and that kind of thing. This Portfolios app then, in turn, has a couple

of services that it takes advantage of which it actually pulls from the public cloud. One

- it actually needs to get the price of a stock, and to do that, we have a service in

the public cloud which we'll call Get, which actually goes off to the Investors Exchange

API (IEX) to access the current stock price. So, it'll take advantage of that, and then

to kind of feed that data back, we have an egress set up - external API request - that

allows the Portfolio app to work directly with the service that we have in the public

cloud. Another service that we have that backs this

Stock Trader application is the MQ service, which is, essentially, a message queuing capability.

And we're going to be using that to keep track of the loyalty levels that a user has when

working with their portfolios. So, various commissions would be changed based on how

long that they've kept a particular stock within their portfolio.

And the same thing here; so, in addition to Portfolios working with the public cloud,

the MQ service is also going to be accessing the public cloud. However, the MQ service

isn't concerned with getting stock prices; instead, it wants to notify users whenever

there is a change in their loyalty or in their portfolio. And to do that, we're actually

going to take advantage of serverless capabilities using Cloud Functions, which, in turn, will

go and send a message to the user using a Slack integration.

So, this kind of sets the stage for the various parts that we have within the Stock Trader

application. And actually, before I forget, there's one more piece - to actually persist

the data for the Portfolios, we have a dedicated database service that's hosted in the private

cloud outside of the cluster that the Portfolios application will be using to persist the data.

This kind of sets the stage for us to jump into the very first piece that I want to talk

about, which is Connect. So, although we've laid out the architecture here, we haven't

really talked about how these applications are able to work with the public cloud services,

although you know that, generally, a private cloud is going to be behind a firewall - it's

going to be in a restricted network. There's one very easy way to expose services

from a private cloud to a public cloud and that is by taking advantage of a VPN tunnel

- that's one of the easiest ways to get started. An IPsec VPN tunnel essentially exposes a

subnet of IPs that can be exposed from the private cloud and public cloud, enabling those

connections to happen. So, we'll create that VPN tunnel between the two environments.

And one key thing to note here is that this is all happening over public internet. So

this has some caveats; although it was very easy to set up, the problem is that when you're

working over the public internet, you can be affected by variability in the amount of

time the request takes to travel between the private and public cloud. In addition, with

VPN, you know you're not going to get the best bandwidth capabilities out there because

you're going over the public internet flows. So there's an alternative to VPNs, and that's

taking advantage of Direct Link capabilities to create entirely private connections between

the private and public cloud. This is made possible by taking advantage of a PoP - which

is a point of presence - generally provided by a public cloud and enables completely private

connections to that private cloud. And this kind of always exists.

To enable your existing architecture to fit into this, you'll need to work with your network

service provider and create a direct connection for all connections coming out of your private

cloud - maybe you have a WAN (wide area network) - to make sure that all of these connections

flow privately. And this way, you never have to actually take advantage of and you're never

actually using the internet for this connection. It's all private and, in addition, the big

advantage of that is you get much higher bandwidth capabilities.

There is one thing though I want to mention - once you have a Direct Link like this setup,

it's also possible to have a failover, which in case this doesn't work, it'll kind of fall

back and use the VPN over internet. By using those two in conjunction, I'd say that's probably

the best way to connect up your networks from a private and public cloud environment.

Next, I want to talk about the service mesh. There's a great project out there that you

might be familiar with - it's completely open source, it's called Istio, and it was created

by a number of industry leaders like Google, IBM, Lyft, and a number of other leaders out

there. And, you know, what we've noticed here - we are taking advantage of Kubernetes on

our on a private cloud, and let's say that we're also using Kubernetes on the public

cloud, although we only have one service in there so far, we'll get around to creating

some additional ones later on. So, what we have is two different clusters

in different environments - we want to make sure that they get managed in an easy way

so that your operations teams don't have to concern themselves with working with multiple

environments, multiple clusters, which can lead to an increased kind of load and can

be quite difficult to manage. So, a service mesh - generally, in the context

of Istio, you manage the services within a single cluster. But there's been new developments

in Istio that allow you to connect up multiple clusters together and have the services behave

as one singular cluster - one mesh across multiple environments.

To better explain this, let's say that Stock Trader wants to create a new version of Trader

- so we've got v1 here and we want to make v2. And this time, we want to host it on the

public cloud - let's say because we want to have the front end of the application geographically

closer to where most of our customers are. So we'll create another Trader application,

and this one is going to be v2. Let's say that all this traffic coming right

now - so, 100% of traffic flowing into the applications - with Istio, what we're going

to essentially set up is a gateway right here. And this gateway has a number of policies

that are kind of enabled and set up by a control plane - so we'll make an Istio control plane.

So we've got Istio here, and that's essentially enabling us to create policies for this ingress

gateway - so all requests that are flowing in.

And now, let's say that we want 50% of traffic to flow through to v1. And then we want 50%

traffic to flow to v2. Very simply, once we have the service mesh set up, all we have

to do is create a policy in Istio that tells the gateway to route the other traffic, and

that will actually go ahead and take advantage of the VPN or the Direct Link connection we

have to move 50% of all traffic to this version of the Trader application.

So, very simply, taking advantage of Istio, our existing connection policies, as well

as the control plane, we were able to create a policy that allows us to route a certain

percentage of traffic to the new version of the application. This is very useful when

you start thinking about creating new versions of your app and rolling them out to your users.

The last thing I want to mention - along with Istio, you also get a number of awesome analytics

and metrics (tracing, management capabilities), so all of those health management capabilities

that Istio offers, they're not limited to the cluster itself. They will actually manage

request the flow between all of your services that your Istio mesh is connected to. Essentially,

this gives your operations team a single point of management for all of your services across

your environments. And the last thing I want to touch on is Integration.

So, there's a lot of things out there that are repeated quite often. That means that

customers are kind of constantly doing these things, so IBM has created a suite of tools

to make integration with the services easier. For example, let's imagine that you have a

set of user data that's stored in Salesforce. You've already taken advantage of this data

- this account data - in your on-prem, private cloud application, but you want to start reusing

those capabilities in the public cloud. So, in the public cloud, maybe there's certain

network or network challenges that change how it's implemented; you can take advantage

of these integration tools to very quickly move that data between Salesforce and your

public cloud microservice applications, taking advantage of some of these tools. This is

made possible through connectors that, you know, not only connect up Salesforce but a

lot of other services out there - things that we notice that our customers are doing a quite

often. Another integration tool I want to talk about

is an API gateway. We're noticing more and more that this is something that's really

important to the overall hybrid cloud architecture, especially when you're working with third-party

services. So here, we actually have a number of them

with Salesforce, Slack, and the Investor's Exchange. Let's say that one of our engineers

has a bug that accidentally hits the Investor's Exchange way too many times and they're throttling

us, which ends up bogging down the whole system. To prevent that from happening, or just to

be more secure about how we're accessing third-party services, what we can do is create a gateway

that essentially sits in between the public cloud and those third-party services and allows

you to do things like manage rate limits, create authentications - things like OAuth

or maybe even basic keys - to really restrict how your public cloud services as well as

users are accessing those third-party services. So, notice that API gateways, in addition

to that suite of tools that I mentioned, are a core part of connecting up your cloud services

to third-party services as well as some of the things that you have going in your private cloud.

I'd say that these 3 topics are the main things you want to think about

when figuring out connectivity with your hybrid cloud architecture.

As always, we're checking comments for feedback or ideas for new videos,

so feel free to drop a comment below

- and definitely subscribe, because the next part of this series

we'll be talking about strategies to to modernize legacy applications.

Thank you.

### Hybrid Cloud Architecture Part 2: Modernization

Hi everyone, my name is Sai Vennam, and I'm a developer advocate with IBM.

Today, we're going to start with Part 2 of the hybrid cloud architecture series

- strategies to modernize legacy or monolithic applications.

In Part 1 of this series, we talked about hybrid cloud connectivity,

and we used a similar sample application - Stock Trader.

This time around, we're taking a step back in time

to when Stock Trader was still a monolithic application

running on-premises, on VMs.

But the architecture is mostly the same

- it's using an SOA (or service-oriented architecture)

that's actually a precursor to microservices-based architectures.

So, very simply, within the monolith itself,

you can imagine this is something like a Java-EE-based application.

We've got the frontend - the UI application - that works with the Portfolio

- which basically manages your different portfolios and keeps track of the stock prices.

So, to get those stock prices, it actually hits another service,

which goes out to the Investor's Exchange public REST API endpoint.

All of that data and the Portfolio information is stored in an on-premises database.

And then we've also got a couple of services here.

So, we've got the loyalty service, which keeps track of

loyalty with specific stocks that you might have in your portfolio,

and then notifies users, as well, on that loyalty whenever it changes

by taking advantage of a message queue service

which notifies the user through something like email.

That's a very simple overview of the architecture,

and this is something that's worked quite well for that fictional company - Stock Trader.

It's worked well for them, and they've seen growth and expanded,

and potentially, you know, maybe they become an international company.

So, what they've noticed is that certain users that are using this application are seeing

increased latency. So, the architects on the Stock Trader side decided, you know, it's

time to get rid of the monolith - it's time to start deconstructing it and taking advantage

of the public cloud. So, let's talk about how they can do that.

The first step of the process to deconstructing is going to be to identify the piece that

we want to break out of the monolith. So, a couple ideas we can throw out here; for

example, we don't want to move the Portfolio service to the public cloud because, you know,

it's tied so deeply to the other services - in fact, it's actually also talking to the

loyalty service. So, you know, if we move that portion, there'd be a lot of unnecessary

network hops - probably making the issue even worse for our users.

Probably the best portion to break out is going to be the UI, or the frontend. That

allows us to put the frontend in multiple geographic locations. Just a quick clarification,

the UI is not only a frontend component, but also the backend for that frontend, which

kind of makes calls to all these other backend services to render data.

So, yeah, I think UI is a great piece to start with - it allows us to start small and set

us up for better deconstruction in the future. So, the first thing we've done, we've identified

the resource. The second thing we want to do is refactor.

So, we can't simply just move that portion out of the monolith into the cloud, and there's

a lot of reasons for that, but the main one being that communication between these services

doesn't do well on public internet. It's software-based calls - it's based within the SOA architecture

within the Java platform. We need to take advantage of something like

REST - something that performs well over the public internet. So, the first thing we need

to do is create glue code; essentially, we need to create endpoints said the UI can access

that Portfolio. In addition, we have to expose REST API endpoints on the Portfolio on the

Loyalty side so that the UI can access that itself.

This is essentially what we call glue code because it allows us to kind of keep that

same pathway of communication between services, but we enable it to work over the public internet.

So that's the second step - we refactor it. And once we've done that, we can actually

go ahead and deploy that into the public cloud. So, the third step is going to be deploy.

So, we take that UI and we put it in the public cloud. And basically, what we need to do is

expose a point for access of it. And, you know, we got the same thing over here where

the UI is exposed from the monolith. The legacy API flow - you know, when a user

traditionally hit this application - comes from their browsers and hits that monolith

application. This continues to work great - we've verified it, that the glue code that

we put in place isn't breaking anything. And then, here's the important step - we want

to make sure that that new API flow, the one that's directly accessing the UI in the public

cloud, continues to work. Now, a good strategy here is to initially

maybe do 10% of your user traffic goes to the public cloud UI, with the remaining percent

goes to the on-prem. This allows you to, kind of, make sure to catch issues in production,

make sure a lot of your users aren't affected. Eventually, you catch all the errors, you

make sure the public cloud is error-free -

that's when you deprecate the older UI portion,

you know, just get rid of the whole thing,

and take advantage of the UI on the public cloud side.

So, the last step is repeat.

Once you have successfully broken out a part of the monolith into the public cloud,

we can start thinking about the next things that we want to break out.

So, so far, we've moved the UI to the public cloud, and let's say that things are going

great. All the international users are getting better response times when they're accessing

the applications. Things are going well, and we may not have a need to further our deconstruction

of the monolith. I think this is something very important to think about. Doing this

refactorization and converting services into microservices - it's an expensive approach,

and until you feel the need to do so, it might make sense to keep your monolith as it.

But, regardless, let's say that this application is continuing to grow, and we have a new bottleneck

- that's gonna be the Stock Price.

So, with all these portfolios and all of these different

users using it, you know, we don't need to scale out these other portions as much.

But maybe the Stock Price, you know, where we're heading that so many times, we're using the

Investor's Exchange to get those stock prices, and we want to scale that out.

Unfortunately, with our monolithic architecture, if you want to scale Stock Price out, we've

got to scale everything out. And then, on-premises, we don't have enough resources to do that.

So, our users are getting bad experiences again as our user base grow, and we want to

move that Stock Price to take advantage of the public cloud scalability.

But, let's say that we don't have time, right? So, users are already having a bad experience;

we don't have time to refactor that Stock Price out and create a microservice. That's

where we can take advantage of "lift and shift". Essentially, take this entire monolith and

move it over to the public cloud. So, you can imagine, you know, let's take

that entire monolith with Stock Trader. But, you know, although we have the whole portion

here, the piece that we really want to scale out is the Stock Price. So, although this

is the entire monolith, we want to only use the getting the stock price portion. So, inside

here, we have a smaller portion to "Get" stock prices.

So, now that we've effectively lifted and shifted the entire model it out to the public

cloud, we can start taking advantage of scalability. We can maybe scale it out eight times, as

an example of taking advantage of public cloud resources.

Now, I understand that this is not the best approach, but with the limitations and since

we needed to go to market, by containerizing the monolith and moving it to the public cloud,

we can really start taking advantage of the resources there and then start thinking about

the next step of our modernization process. One thing I want to mention here - so, my

next line here, "innovate and improve" - you always want to find ways to improve your application.

What we've noticed here is that the UI is in the public cloud, but kind of the communication

channel for it, essentially the UI is always kind of going back to the monolith to take

advantage of the Portfolio, Loyalty, and whatever other services might be there, right?

So, the first thing that we notice is that the UI hits the Portfolio, which then has

to come back out here to hit the monolith in the public cloud to get the stock prices.

That, in turn, goes back to the Portfolio and then back out to the UI. That's a lot

of unnecessary network hops. We can always innovate and improve throughout this process

of modernization. Why don't we get the stock prices directly

in the UI and then offload all of the database storage activities asynchronously? Well, that's

one easy way to innovate and improve on our existing architecture. Let's refactor these

applications so that the UI is talking directly to the monolith in the public cloud to get

those stock prices. That's one example of always innovating and always improving when

doing this kind of migration. Another thing - so we talked about how using

the whole monolith in the public cloud is not the best approach, but it allowed us speed

with go to market. Let's take advantage of some new technology - say serverless, for

example. So, we want to factor out this one portion right here and take advantage of serverless

capabilities to get stock prices. So, by using the serverless platform on a

cloud, we can then take Functions as a Service, which in turn goes and hits the Investor's

Exchange public APIs. So, we'll take advantage of the IEX public APIs using serverless, and

then what we want to do again - so, same four-step process - we've identified the piece (Get),

we've refactored into a serverless action, we've deployed into production, and then what

we want to do is test it there, right? So, legacy and new API flows. So, again, this

is the legacy flow. And then the new API flow is going to hit that serverless action directly.

Once we verified that this flow works well, we can entirely cut out that monolithic architecture

that we pulled into the public cloud for simply just stock prices.

So, again, we talked about how we take advantage of this four-step process to break out individual

portions of a monolithic architecture and move it into a public cloud. These three things

- deconstruction of monoliths, lifting and shifting, plus always innovating and improving

are going to set you up for success when you're modernizing your monolithic applications.

As always, we checking for feedback, so drop a comment below.

In the next part of the Hybrid Cloud Architecture series we'll be talking about security.

So, definitely subscribe and stay tuned.

If you want more information about what we talked about today

check the related information in the links below.

Thank you.

### Hybrid Cloud Architecture Part 3: Security

Hi everyone, my name is Sai Vennam, and I'm a developer advocate with IBM.

Today, I want to talk about security with hybrid cloud architectures.

This is going to be Part 3 of the hybrid cloud architecture series.

Security is a nuanced topic, but to kind of help me explain,

I'm going to start with two major concepts:

north-south network traffic vs. east-west network traffic.

When I walked into the office today, I had to pull out my badge

and scan to get into the building.

This is something called perimeter security,

and it's a core part of north-south network traffic.

Essentially, what that refers to is any traffic that's traveling from end-user applications

to your data centers or public or private cloud environments.

Let's take a step back and kind of explain these pieces here.

So, we talked about this in the previous videos, but what we've got here

is the Stock Trader monolith,

which is going to be on an on-premises data center.

We've got a couple of services here -

maybe something to help us talk to the cloud

and maybe a data store as well.

So, we mentioned perimeter security,

and that's something you, honestly, take as a given with data centers

- that you have that firewall

sitting in front of that data center giving you a private network

for your actual data center and the applications and workloads running within it.

This made security a lot easier to tackle when working with monolithic applications,

but it did put the onus of security

on the application - the enterprise application developer.

The main thing here to actually secure these endpoints was to make sure that all the capabilities

that this monolith exposes (those API endpoints) were secured. And to do that, we could take

advantage of something like an API gateway. So, traditionally what we would see is a gateway

that's set up in front of that on-prem application with key capabilities exposed that may be

required by that frontend to render the application. And potentially the same for a mobile app

as well. That, I think, helps tackle security with north-south network traffic on the on-premises side.

Let's shift gears here for a second and talk

about the public cloud side or even potentially a private cloud.

I'll talk about the different components here later in the video but let's start with this

piece right here, which is the Kubernetes worker. Within the Kubernetes worker, we can

assume that we have a couple of services that we need to actually render the Stock Trader

application, whether it's mobile or in a web app. We have a couple of services and can

assume they talk to one another. So, what happens when an end user actually

accesses the application? Well, one, they'll actually have to hit that endpoint that becomes

available, at which point they will enter the public cloud. At that layer, we get things

like denial of service protection and other things that the cloud provider offers you

to make sure that those requests are maybe authenticated or, you know, they're they're

coming in in a safe manner. The next thing that happens, that request

will get forwarded to your actual Kubernetes worker node with the capabilities that it

exposes. So, at that level, we have a couple of options for securing those endpoints.

Let's say, you know, we want to hit this first microservice running in a Kubernetes worker

- there's two ways that we can kind of configure security policies. The first is going to be

at Layer 3, which is, if you're familiar, it's things like IPs and ports - basically,

it allows you to configure policies for any network interface. That's gonna be done with

things like Calico or the native Kubernetes API policies. So, that handles the Layer 3

security level. The other option we have here is to use something

like Istio for Layer 7 network policies and routing for security. Together, with those

two capabilities, we can cover everywhere from Layer 3 to Layer 7 network security policies.

So, the request comes in and, you know, granted that it passes those policies, it gets forwarded

to your worker and whatever services it might hit. So, this is the ingress application flow.

And then, for external requests that a service might make (for egress calls), the same can

be configured in Istio or Calico, going everywhere from Layer 3 to Layer 7.

So that kind of talks about north-south traffic - ingress and egress - communication with

the clients as well as a data center or a public/private cloud environment. So, that

tackles north-south network flows. Next, let's talk about east-west. So these

are going to be, essentially, communication happening between services running on-premises

or in your public/private cloud environments.

So, for east-west, going back to my analogy

- I badged into my building, they let me into the perimeter, but to actually get to my floor

where I work every day, I have to badge again. That's going to be on the third floor of the

building, right? So, I go up to the third floor, and I'm forced to actually scan my

badge again. If I try to enter the fourth floor, I actually wouldn't be allowed to enter

as I'm not on the design team. So, essentially, what that refers to is a

concept called segmentation. So, within the actual building or an application infrastructure

- maybe a public cloud environment

- we want to create segments of what users are allowed to access,

what admins allowed access, what processes are allowed to access

when talking to one another.

So, at that level, we actually call this in

Kubernetes environments, we call that micro-segmentation.

In the customer-managed environment, what

that would look like is, essentially, setting up - using something like Istio

- TLS between all requests going between microservices.

The thing about encryption - it's one of those things that

you want to encrypt any requests as early as possible

and decrypt as late as possible.

So, with traditional Kubernetes microservices architectures,

you want to make sure that all of those requests are being encrypted at the earliest level possible.

That kind of handles microservice-to-microservice architecture, but we didn't really need to

consider that with the monolith because, again, as we mentioned, monoliths would be using

RPC or remote procedure calls, software-based calls, which remove the requirement of talking

over a network so we wouldn't actually have to take advantage of TLS.

But, you can imagine that you do want to make sure

that the network calls made to the database would be secured TLS.

The next concept I want introduce is what

we have sketched out here on the cloud-managed side of our cloud.

So, what we've got here is the Kubernetes master node.

And one thing to remember here is that when you're working

with a managed Kubernetes service,

the master node is actually going to be managed by the cloud provider.

So, whereas you control the worker nodes, the master is completely managed

and houses a very important piece of the architecture - the etcd data store.

So, in the Kubernetes world, the etcd data store is something that you want to be really

careful about protecting because that has all the information about your services, your

deployments, and all of the Kubernetes API resources. So, securing the etcd is going

to be very important; it's paramount to your security architecture.

And to secure that, the cloud provider, traditionally, will have a kind of a three-phase process.

So what we'll have is everything from - we'll start with step one, which is authentication,

so TLS. Next, we've got RBAC, which is Kubernetes role-based access control for authorization.

And then, finally, over here, the last piece of the puzzle is gonna be the admission controller,

which is a Kubernetes concept that, you know, once you've made it through the authentication

and authorization, there's another level of security to make sure that those API requests

are mutated or, you know, massaged and made sure that they're in the right format to access that data.

So, they'll access that etcd data, and to

send that back to your worker node, where your application pods need to request that

information or, you know, pass information to it, there's an open VPN server. And there's

also going to be a client, as well. But, that's going out enable you to basically access that

etcd data store and return data back into the Kubernetes worker.

So that kind of covers the pattern of how Kubernetes is set up in a cloud provider service,

with the master node being managed and the worker node being able to, kind of work, with

that master node in a secure fashion to make sure your assets are protected at all times.

The other thing I want to mention here - that etcd data store is going to be backed up in

a cloud object storage capability to make sure that, you know, worst case scenario,

you do have those assets in a secure place.

So, I think that covers north-south network traffic as well as east-west,

where we talked about network traffic coming in from clients

or, at least, network traffic going between services

in your data center and in your private or public cloud environments.

The last thing I want to talk about is a concept called DevSecOps.

You'll notice here that it's DevOps with the word "security" right in the middle,

and, essentially, it's a way to ensure that security is something that you think about from the ground up

when you start architecting the application

all the way until you move into production.

And that's something you want to take advantage of

to make sure that you don't have any issues when moving to production.

You don't want to architect an application the incorrect way

and then realize you have to go back and rework all of that.

So, thinking about security from the beginning is going to be an important thing.

When working with a cloud-provided Kubernetes service

there's something that makes it a little bit easier

to make sure your flows are secure.

One consideration you want to have here is to make sure that your CI workflow

- that DevOps flow -

has security embedded within it and is automated.

So, you can imagine, maybe you have your favorite

code repository holding your application code

- your Docker files, whatever they might be.

We're going to automate that process

and make sure that only the developers who building that could have access to that Git repo.

Next, you want to make sure you have a trusted

signer to make sure that that code, when it gets pushed into a registry, will go ahead

and sign it as a trusted image to something that's available with the cloud manage registry.

So, we'll push that image into the registry.

Once there, there's a capability called Vulnerability Advisor

that's going to scan that image and make sure that if there's any issues or any vulnerabilities that are detected

- everywhere from the base operating system

to the runtime that you're using

- that if a vulnerability is detected, you'll be made aware of it.

Once it passes that vulnerability assessment,

you can tie that in to build that image and push it directly into Kubernetes.

At that stage, you can use something like an admission controller

(which we talked about in the Kubernetes master)

to make sure that that image is, again,

secure and without vulnerabilities. Finally, there's a live-scanning capability

to allow you to scan your images running in production to make sure that there are no vulnerabilities in there.

So, DevSecOps is a very important concept

that ensures that, from the ground up, you're managing security when doing DevOps.

Thanks for tuning into part 3 of the Hybrid Cloud Architecture series on security.

If you haven't already,

be sure to check out the introduction, part 1, and part2 - the links will be below.

As always, feel free to subscribe if you want to see more of these videos in the future.

Thank you.

#### [Module Summary](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/76d637cbe8024e509dc445df847e6c3a/a426f5831d1d48a895817ae2f33e3422/)

# Summary

In this lesson you learned about:

* + Breaking apart application monoliths into smaller packages suggested by the programming model.
  + Re-architecting applications into containerized microservices that communicate through a service mesh.
  + Exposing critical elements of applications as an API for other applications to use.
  + Making code changes and adding new value and capabilities to workloads.

Dans cette leçon, vous avez appris :

o Décomposer les monolithes d'application en plus petits paquets suggérés par le modèle de programmation.

o Ré-architecture des applications en micro-services conteneurisés qui communiquent à travers un maillage de services.

o Exposer les éléments critiques des applications en tant qu'API pour que d'autres applications puissent les utiliser.

o Modifier le code et ajouter de la valeur et des capacités nouvelles aux charges de travail.

# For more information

For additional information, you may be interested in the following materials:

* Cloud-Native Applications  
  <https://www.ibm.com/cloud/learn/cloud-native>
* Increase agility with application modernization  
  <https://www.ibm.com/services/cloud/modernize-applications>

### Module 3 - Cloud-Native Practices

### Learning Objectives

In this lesson you will learn about:

* + - Leveraging cloud development best practices.
    - Using frameworks including containers, microservices and serverless.
    - Automating application delivering with CI/CD.

Dans cette leçon, vous en apprendrez plus sur :

* Tirer parti des meilleures pratiques en matière de développement du cloud.
* Utiliser des cadres de travail comprenant des conteneurs, des micro-services et du "sans serveur".
* Automatiser la livraison d'applications avec CI/CD.

#### [Lesson 1 - DevOps Tools](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/bd64ccdf56ad4ea1afe870e26d583038/155e435d066a4a22a2c7a2994308e709/)

## Lesson Introduction - DevOps Tools

This lesson contains video sessions, as follows:

- **What is DevOps?** explains the set of practices that combine software development and IT operations to shorten the systems development life cycle and to provide continuous delivery with high software quality.

- **Terraform Explained**provides information about a tool for automating and managing infrastructure and platform services.

- **What is Tekton?**explains how to address specific problems around continuous integration and continuous delivery.

Cette leçon contient des sessions vidéo, comme suit :

- Qu'est-ce que DevOps ? explique l'ensemble des pratiques qui combinent le développement de logiciels et les opérations informatiques afin de raccourcir le cycle de vie du développement des systèmes et d'assurer une livraison continue avec une haute qualité logicielle.

- Terraform Explained fournit des informations sur un outil d'automatisation et de gestion des services d'infrastructure et de plate-forme.

- Qu'est-ce que Tekton ? explique comment résoudre des problèmes spécifiques liés à l'intégration continue et à la livraison continue.

## DevOps Tools

### What is DevOps

Hi, I'm Andrea Crawford with IBM Cloud.

We're going to talk today about DevOps.

As the name implies,

DevOps is about bringing together

2 traditionally contentious groups within IT organizations:

application development, which has primarily been measured on

the number of changes, features, and defects fixed

that are able to be pushed out into production,

whereas IT operations,

and their success, is primarily measured on system stability, availability.

So, we have one group with pushing more changes,

and another group that saying don't push any changes.

So, let the fireworks begin!

DevOps transformation

is an important part to our client's transformation strategies in general.

Particularly because of 2 main benefits.

One of those being velocity.

Velocity in terms of how quickly products or applications

can be piped through this release pipeline.

The second benefit: quality.

It's not enough just to be quick in terms of

how fast applications are delivered,

but we also must be cognizant of the quality of what gets delivered on the

other end of that pipeline.

Digital reputations are at stake here, so quality and velocity

both need to be balanced.

The war between application development and IT operations really has one big loser and

that is the business, because you see the benefits are really going to manifest themselves

in terms of business agility through are being responsive to consumer demands, to changing

market conditions, and basically improving overall time to market of what an enterprise

can deliver to its consumers.

If you were to think of application delivery as a supply chain, or a pipeline, we really

have this notion of ideation, user stories, what should be delivered in terms of adding

value to the business and its consumers.

So, we have ideation over here. This is also on in the form of a user stories,

work items, and the like.

Then we have the coding piece. This is all about programming the idea from the user story.

And then we have Build. This piece here includes unit test cases, packaging of the code into

executables, and run times that will actually be then deployed on some sort of run time

environment, be it bare metal, virtualized environment, or the cloud.

And then we have Manage, and this is all about addressing what happens to an application

once it lands in its runtime environment. And this is primarily focused on production,

but it can also be management in terms of what you have in dev, test, pre-prod,

and the like.

And, then to round out the supply chain here, we've got Learn - and this is all about continuous

improvement. So, not only do we need to understand are we getting faster as we deliver applications,

but is the quality improving, too? So, we want to make sure that our benefits are being

achieved throughout the pipeline here.

So, if we were to identify the use-cases for how to increase velocity and quality in the

pipeline there are really 3 primary use-cases. The first is optimizing the core. By core

systems, I mean we those systems that are deep within the enterprise. These tend to

be monolithic, legacy, big systems of record. Typically, these applications are very difficult

to modernize, re-factor, and the like. DevOps for the core often looks like retrofitting

automation in situ, you so that we can optimize what we've got.

The second use case is about unlocking the legacy. So, this is all about addressing culture,

ways of working, and bringing development and operations together in new ways, integrated

multi-disciplinary teams, and this amps up any kind of automation that we could have

done in the core.

And then the last: unleashing the digital. So, with the maturation of cloud-native apps,

12-factor, Kubernetes, and Docker, were all about continuous delivery from ideation all

the way through to continuous improvement and rounding back to the beginning.

So, what we learned that we can do better we feed it back into the pipeline. This is all about

modern applications delivered by modern teams on modern platforms. So, if we were to sum

all of this up, DevOps is all about people, process, and tools for the benefit of

business agility.

Thanks for watching this video. If you have any questions or comments be sure to

drop a line below. If you want to see more videos like this in the future,

be sure to like and subscribe.

### Terraform Explained

Hello everyone, my name is Sai Vennam

and I'm with the IBM Cloud team.

Today we're going to be talking about Terraform.

Terraform is an open-source tool

originally developed by HashiCorp,

that enables you to automate and manage your infrastructure

and platform and services as well.

It does all of this using a declarative language

and that's the first thing we're going to focus on

when we go through our example

to go through Terraform.

So, I generally like to start with an example here.

So, say you're driving in your car,

trying to get from point A to point B.

Generally you would follow a set of instructions, right?

So, going from point A to point B,

you have to take a left turn,

get on the highway, take this exit

- you end up at your destination.

Now, that's kind of an imperative approach to automation.

The way Terraform does it:

imagine you called a taxi or a rideshare service

and told them exactly where you wanted to go - point B.

The car, or the driver, then takes care of the rest

- you don't have to worry about every step of the way to get there

and that's the approach Terraform takes.

That's why it's so effective

at managing your infrastructure.

Today we'll start with an example.

Say you've got 3 different resources

that you're trying to spin up as part of your infrastructure.

We'll say you've got a VM,

we'll also say you've got a Kubernetes cluster,

and say that they're networked together using a VPC

- or "Virtual Private Cloud".

So, current state: nothings there.

Desired state is this set of infrastructure.

So, in Terraform there are going to be 3 major phases,

and we'll go step-by-step.

The first phase:

you actually have to create the the terraform file.

So, we'll start with that -

where you actually write or code up that Terraform file.

Now, in this Terraform file we're going to have

3 major resources, right?

So, we'll start with the VM

and a set of arguments to support that.

Things like the name, and networking, data center,

that kind of thing.

We'll have the Kubernetes cluster,

a set of arguments for that,

and then finally we've also got the VPC.

It's probably going to refer

to some of the network settings of the VM and the Kubernetes cluster,

but it's going to network those together

and so it has arguments to support that.

So, that's your coding phase.

Say you've got a Terraform file ("TF" file),

with these things defined.

Next up, we've got the "plan" phase.

Now, this is an actual Terraform command.

So, in the TF CLI, or "Command-Line Interface",

you can run "terraform plan",

and what it's going to do is it's going to compare

the desired state to what actually exists.

So, on day-zero it'll notice that you have

none of these resources,

so it'll say, "Hey, you have to create all of them".

So, create the VM,

create the cluster,

as well as create the VPC.

So, it has a plan -

and it tells you, the user, this is what it's going to do.

Next, if everything looks good,

you'll do the "apply" phase.

This is another Terraform CLI command.

In the apply phase you can actually

take those resources and spin them up.

So, that's exactly what's going to happen:

Terraform is going to work against the cloud providers

using real APIs - your API token -

to spin up these infrastructure resources

and it's going to output some interesting or

auto-generated variables along the way.

So, for example, maybe the Kubernetes dashboard URL,

or maybe even a URL to access your application,

but, regardless,

it's going to output a number of these output variables.

So, that's generally how the terraform workflow goes

to get you from point A to point B

in a situation where you had nothing

and now you have something.

So, that's what I wanted to start with.

Next, I want to go into the fact that Terraform

has a strong open community

and it's pluggable by design.

So, by "pluggable" essentially what I mean here

is the fact that it's made up of

modules and resources where

you put in input variables and output comes out,

as well as the community is out there,

and cloud providers are out there,

building and supporting things called "cloud providers",

or "providers" in the Terraform world,

to enable you to connect up to any cloud

and automate infrastructure there.

So, in this process,

we actually made something called a Terraform module.

A Terraform module is a way of grouping together some terraform automation.

So, we've got a terraform module here.

It actually takes a set of inputs

and also creates some output

and as part of every module,

- well, you can define this, but,

in general, when you're working with cloud providers,

and using Terraform,

you'll also define what's called a "provider".

Now, a provider can be a number of things,

but in our case, we are using it as a cloud provider

to connect up to a given cloud.

So, that's the first thing that a provider can do is

connect you up to IaaS providers

- IBM Cloud, or AWS, or Azure -

it enables you to connect up to some infrastructure provider,

spin up things like VMs,

or that kind of thing - maybe bare metal.

You can also use a provider, in the Terraform sense,

to spin up platforms as well.

So, it enables you to manage, for example,

Cloud Foundry running in the cloud.

Finally, you can even manage SaaS offerings.

Things like CloudFlare, or other software services,

can actually be managed by Terraform.

So, although Terraform is considered to be

an infrastructure automation tool,

it's expanded its role

to support other types of providers as well.

So, essentially, we've got our flow here,

our module, inputs, and outputs,

and so we've got a kind of pipeline

to be able to spin up resources.

The last advantage I want to mention here with Terraform

is the fact that it enables you to essentially

have the best practices for DevOps.

So, we'll say "DevOps first".

Now, let's take an example for this.

Now, in our example here,

we did a day-one kind of deployment,

- nothing to something.

But let's say we're going back and iterating on this.

So, we've got our VPC,

we've got our VM,

as well as Kubernetes.

Now let's say we're iterating on this and we

want to create something new on top of this infrastructure.

Let's say we want a load balancer.

So, that's our desired state.

So, we'll do our three phases,

starting with the code phase.

So, we'll say here that we want a load balancer,

and we'll have set of arguments to support that.

When we get to the plan phase,

Terraform is going to realize,

"hey we actually already have the VM, Kubernetes, and VPC",

so it checks the current state of the world

and realizes that all we need is that new load balancer.

So, it will say, "hey, let's just add the load balancer this time around",

and, as a user, you'll confirm that make sure it looks good,

go to the apply phase, go ahead and apply it,

maybe some more config variables will come out

for this new capability that you added.

So, essentially, with Terraform

you have a DevOps-first approach,

one of the key advantages to this

is that it enables you to avoid potential "config drift" -

that's when the configuration that defines your infrastructure

actually doesn't match what's actually there.

So, as long as all changes to your infrastructure

go through the Terraform file and Terraform pipeline,

you can essentially eliminate the risk of configuration drift.

In addition, since we've set up

a module where we can take input variables and have a final state,

what we can essentially do is recreate this

- maybe switch up the environment variables a bit

and then create a whole other environment

that looks just like this

but this time we can make one for Test.

And maybe this one was Development,

and we can do that again for maybe Production.

It's another advantage of taking a Terraform

approach to infrastructure automation:

it enables you to put DevOps first.

Thanks for joining us for this quick overview of Terraform.

If you enjoyed this video

be sure to check out our other video on infrastructure as code.

As always, if you like this video, or have any comments,

be sure to drop a "like" or a comment below.

Stay tuned and subscribe for more videos like this in the future.

Thank you.

### What is Tekton?

Hi, my name's Matthew Perrins. I'm part of the IBM Cloud team and today I'm going

to give you an introduction to Tekton. So, what is Tekton? Tekton started life

as part of the Knative Project as they were trying to solve specific problems

around how they do continuous integration and continuous delivery of

their software components and was later donated to the Continuous

Delivery Foundation where we're now being able to use it and consume it

across multiple Kubernetes environments and platforms. So, what I want to do today is

give you an overview of the base concepts that underpin what Tekton is on

explain about how you run it inside a Kubernetes environment and how it

helps you automate and deliver your software components for running inside

that platform. And then I want to give a brief explanation of how developers

would consume it and work with it and how that separation of concerns may go

forward. So, base level component of a Tekton continuous integration or

delivery pipeline is something called a task. And a task is an automation task,

yeah an automation tasks that you need to create that will help you build, test,

deploy, manage, check the health of a particular piece of software. Once you

create a number of tasks you can actually isolate them and reuse them

very easily and Tekton one of the real benefits is that it allows you to

separate your tasks that are then consumed by your developers and I'll

explain that in a little bit more detail as we move forward. So, once you've got a

base element task you can create something called a pipeline and a

pipeline can be made up about any number of tasks

so you can customize that and tailor it for the specific software needs that you

have. Once you've defined a pipeline you can trigger a pipeline from something

called a pipeline run, and you create multiple runs in a software delivery

project but it will continue to run the exact same definition of the pipeline

that you defined with the exact same definition of the task that you define.

Now one of the next pieces of key information to make this sequence run

smoothly is you need to provide some data for that execution so that you can

tailor and customize it. Anything from a build number to supplying a unique git

repository or supplying a Docker image and this is called

Pipeline Resource Res. So, now we built these base components for a Tekton

Pipeline what we need to do is then start to think about how we prepare a

developer to consume them. So, what you would typically do is you'd register your

common tasks into your Kubernetes environment inside the Tekton Execution

Engine that you will install inside Kubernetes, and those tasks can be commonly

shared across different applications or different development teams. So, it allows

a level of management and control that can be owned by the operation, the

operations side of your DevOps organization. So, Tekton itself runs

inside the Kubernetes cluster and it associates very cleanly to a lot of the

Kubernetes concepts like pods and services and CRD, CRC, YAML definitions. So,

once you've got your pipelines defined or the concepts about how you're going

to use your pipelines and once you've actually installed and got Tekton

running inside your Kubernetes environment, what we need to do is then talk about

how a developer would start using that in the creation

and delivery of their projects. So, as I mentioned you can have different Tekton

pipeline definitions and we'll call that pipeline YAML and you can define that

file inside your individual application and the relationship if you remember is

a pipeline can run any number of tasks and these tasks have been already

registered and defined inside the Tekton environment. So, you can have task 1, task

2, task 3, and keep going, and within each task you can have a

number of steps and again you can have any number of steps depending on the

particular problem you're trying to solve.

So once you've defined your pipeline definition inside your application then

the developer can push their change to their Git Repository and this will allow

them to iterate multiple times on code changes or definition changes or feature

and function improvements. And what happens is this triggers a webhook which

will then trigger a type a Tekton execution which will then marry the two

elements together, it will marry their pipeline definition that comes from the

application and it will take the tasks that you've defined inside your Tekton

environment and this could result and we hope it does successful delivery of your

applications into specific namespaces, whether it be a dev, a test, or a pre-prod.

So, you've actually packaged up your code into a K8, a K8 image registry and from

there you've now got a piece of packaged software that you can then roll out and

deploy to other environments. So, I hope you enjoyed that walkthrough and

you've got some information about the basic concepts of what Tekton

is, how applications can use it, and how you run it on top of the Kubernetes

platform. Thank you if you have any questions please drop us a line below. If

you want to see more videos like this in the future, please like and subscribe and

don't forget you can always get started on the cloud at no cost by signing up

for a free IBM Cloud account.

#### [Lesson 2 - Continuous Integration / Continuous Delivery (CI/CD)](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/bd64ccdf56ad4ea1afe870e26d583038/bb610b284aed4d31aa756196ceeab64f/)

## Lesson Introduction - Continuous Integration / Continuous Delivery

This lesson contains video sessions, as follows:

- **What is Continuous Integration?**dispels myths about the widely misunderstood development practice of continuous integration.

- **What is Continuous Delivery?** explains the practice of continuous delivery.

- **CI/CD Pipelines for Microservices**helps you prepare to test and publish your microservices.

- **CI/CD and DevOps**explainsthe CI/CD portion of DevOps.

Cette leçon contient des sessions vidéo, comme suit :

- Qu'est-ce que l'intégration continue ? dissipe les mythes sur la pratique largement méconnue de l'intégration continue en matière de développement.

- Qu'est-ce que la prestation continue ? explique la pratique de la prestation continue.

- CI/CD Pipelines for Microservices vous aide à vous préparer à tester et à publier vos microservices.

- CI/CD et DevOps explique la partie CI/CD de DevOps.

## Continuous Integration / Continuous Delivery

### Continuous Integration

Hi, I'm Eric Minick with IBM Cloud.

I want to tell you a little bit about continuous integration today.

This is a development practice

that just about everybody thinks they're doing,

but it's widely misunderstood.

So, I want to start by talking about the bad old days before we had it,

what problem we're trying to solve,

and what this does for us and what it's really about.

So, let's start with our old school approach

where we've got a couple of developers,

they're probably on a team of 30 or 50,

we got Alice and Bob and they're working on their own features,

and they're going to keep separate on their own features for a long time.

They're writing some bits of new code here.

Maybe they're deleting a line of code here or there, what have you.

And their plan is that they're going to get their features to work against the code base,

then in a couple of weeks, or months, later

they're going to integrate their work together.

They're going to merge their work together.

They're going to say:

"My code, your code, it's got to come together, it's all got to work".

Now, in this world,

you could have a situation where Alice & Bob, they changed the same line of code here, right?

And maybe Alice changed this code, but then Bob deleted it.

And so, this gets really awkward, really fast when you just try to algorithmically combine it.

You get what's known as a merge conflict.

This gets worse because it might be that some code up here,

interferes with some code down here,

they just don't work together and you've got bugs.

And this will happen here within just one file,

but in the real world,

it's not just two developers working on the same file

it's thousands of files, or millions, in an application.

There's tens of developers involved

and they're doing this over weeks and weeks.

And so, reconciling all of these problems, it's big mess.

It's called "merge hell".

And so, they say, "What do we do to avoid merge hell? This is how we got here".

Well, we start, and we'll have Alice writing some code again, right?

Alice writes her code.

Cool. As soon as she has something that works even a little bit, her feature is not done,

but she's got code that works, it doesn't break things.

She submits that in the source control, Bob pulls down the same code

before he starts working, right?

He always is working off the latest.

Alice is always submitting hers in.

Bob makes his change right?

Maybe he deletes a line of code, great.

Alice comes back, she's working another part of a feature, she's working on another feature,

she grabs Bob's code. Cool.

And now they're working on these things together.

And the likelihood that they're both working on the same thing, at the same time,

and that they conflict is pretty small.

And if they do hit that, it's all on code they've worked on in the last day, right?

they can sit together, they can reconcile this thing, it's no big deal.

And so, the core principle that we're seeing here,

one of my favorite things in software development, because it is so counterintuitive,

is that if it hurts, do it often and it won't hurt so much.

And that helps us get rid of the big, big bogey here, right?

Which is, we're going to not have merge hell.

We don't like merge hell. Get rid of that.

OK, so that's fix number one.

Now, this does create a new problem though, right?

They have a whole bunch of people constantly checking in code into the code base.

Things are going to break, right?

It's not going to compile.

There's going to be bugs that weren't there yesterday.

And you have this kind of, "continuous integration led to continuously broken" problem.

And that's sad. So, what do we do?

Well, we put some automation in, right?

And it's always watching the code.

Keeping an eye on it,

and whenever a change happens,

It says, "OK, let me grab that code. I'm going to build it."

"And then I'm going to test it."

"Alright, I'm going to run my unit tests".

So, if there's any change, build it, test it.

If anything in here fails, then we email the team.

We let Alice, Bob, the whole team know, say

"Hey, there were 2 new changes since last time I did a build,

looks like both Alice and Bob touched this thing.

It's broken now. They need to go fix the code base."

Good. So, now we've got a safety net that enables continuous integration. Good.

The other thing that we're getting though, is that we always have this build. Right?

So, whenever the test team wants to go test it in more depth,

to run a heavier level of tests than the unit testing that we're doing here,

they've got a build they know compiles -

and it's the newest stuff, right?

So, that's our other key benefit coming out of continuous integration

is that we always have a testable build.

So, there you go.

Continuous integration was addressing the old way of infrequent integration,

saying, "If it hurts, we're going to do it continuously, we're going to do it a lot."

That avoids merge hell, keeps our developers productive,

and it gives us a constantly testable build.

So that's continuous integration in a nutshell.

Thank you.

If you have questions, please drop us a line.

If you want to see more videos like this in the future, be sure to like and subscribe.

### Continuous Delivery

Hi, I'm Eric Minick with IBM Cloud.

I want to talk a little bit about continuous delivery,

which is all about "how do I quickly get code into production?".

Continuous delivery as a term goes all the way back to the agile manifesto.

The first principle of that says,

"What's really important is that we satisfy our customers

through the early and continuous delivery of valuable software."

So, if we're in a place where we've got valuable changes to the code

(and if our changes aren't valuable, we probably shouldn't be making them)

but if we've got valuable changes to our code

we want to get in production, how do we do that?

That's what continuous delivery is all about.

Well, the first step here is that we've got code, not software,

so we need to turn it into software.

That's our build process, right? Good.

Now we've got software, we've got production, and we could just deploy the software into production.

Is that what we're going to do? No.

What we're really going to do is a whole bunch of testing

to make sure that what we have actually belongs out there.

So, most companies I work with have anywhere from 2 to 5 test environments.

They'll have an environment named something like "QA", right,

and another environment named something like "Performance" or "Stage."

Well, we can use those. Alright.

So, we take our software, we put it in the QA environment, we put it in the staging environment,

and we put it out in production.

Alright, and we should probably do some testing too.

So, this is kind of the backbone or the path to production in continuous delivery.

When you look at a continuous delivery tool, or you look at a continuous delivery pipeline,

you'll often see this kind of perspective: "I have builds and I put them out in production."

Now, what are the activities here? What are the key behaviors?

So the first one is this migration that we see, right,

so we can look at that migration, this migration, this migration,

and we say "Okay. This requires an automatic deployment, right?"

Auto deploy. This is where we see tools like application release automation,

continuous delivery pipelines, very focused on "how do my builds move through the environments?",

"what's the order of the environments?", "How do I manage and govern that?",

"Are there rules for when I need to move from here to here to here?" That kind of content.

The next side of this is it's not just about moving the bits, right, if we're just moving things around

the only thing that we've tested is that we can actually deploy our software.

That's valuable, but we probably have all these environments because

you know down in QA, in staging, we're doing some automated testing.

Right. Supporting a continuous testing kind of approach as part of our overall continuous delivery.

And then we're probably also using some tooling down here that we care about.

This would be our continuous integration server that does our builds, right?

The idea of a build manager or a continuous integration server have all come together.

So, here are the key pieces.

Final idea, typically in continuous delivery,

this transition here, moving from stage to prod,

is going to have another level of governance on it.

You might be in a fully automated situation where code commit comes in to Github.

There is a trigger that tells the build server "new code's here". It automatically does a build.

It triggers a deployment to QA, which automatically triggers our tests.

If those pass, we automatically trigger out to stage, which automatically triggers more tests.

That's great. It's continuous. It's fast.

When we're down here, when we're looking at this move to prod,

almost always were looking at some sort of a human getting involved.

Historically, that was your change approval board, your CAB.

Ideally if we're automating all of our testing, we're getting all the data,

we're able to automate the CAB out of existence and

just say "Hey look all the test passed, let's make a go/no-go decision here,

button's clicked, we moved to prod."

There may be some approvals out to a change management system, you might still have the CAB,

so there's something going on here with some decision making typically.

That's the basic idea of continuous delivery.

You have something valuable in code.

You want that in production, but you don't want to break things.

So what do we do?

Take our code, make it software in a build process.

Now we want to make sure we're not breaking, so we're going to test our deployment process, moving to QA.

We're then going to test the software with automated testing.

We're going to use the same deployment process to stage,

same deployment process out to production after doing more tests, more checks, getting approvals.

That allows us to get those changes into production quickly and safely.

And that's continuous delivery.

Thank you.

If you have questions, please drop us a line.

If you want to see more videos like this in the future, be sure to like and subscribe.

## CI/CD Pipelines and Tools

### CI/CD Pipelines for Microservices

as we saw with the demonstration before

as we start to look at you know multiple

components that are in different

repositories and the microservices

architecture where I have one component

per repository I need to have a

continuous delivery methodology that

embraces and supports this and this

diagram that we have here is something

as from our IBM cloud innovate used to

be called the cloud garage method but

it's a cycle right where we include into

continuous delivery you know

understanding about what's going on

allowing our teams to be able to

collaborate effectively to test and

shift left when we are testing a talk a

little about shift left testing and the

implications here in a moment

being able to once testing is past being

able to continuously release and deploy

monitoring and looking at feedback from

users and using that to optimize what we

do as as we move forward for design of

an application so the key point here is

if you're going to move into this new

world of enterprise applications on a

cloud environment it needs to be done

automation automation first in fact you

need to build these processes and

procedures first before you embark on

the journey because it's essential for

moving forward and when a few a few tips

around what we're doing here and the

examples that we're showing right that's

it when you think of a micro services

application it's going to have

individual components each component

will have its own repository and each

component may have different

requirements or baled and deploy for

example with a micro services

architecture I now have a polyglot style

in the old monoliths I had to have the

same programming language top to bottom

now I can have many different

programming languages so I may have

different build needs each particular

component will need its own deployment

scripts so in the example we're showing

here or you have a Jenkins file at

Jenkins file is part of the code and

check it in to the source code

management system and automate your

deployments using that so

that you can get microservices up as

quickly as possible

the other part of processing micro

services and you think about it from a

testing perspective is that you have

some new opportunities for testing with

micro services not only do you have the

capability to do unit tests during the

build phase and also look at code

quality through static code analysis is

that into a staging environment we need

to do into end testing and integration

testing so if I have three components

that are live and I've added a new

version of one of the components I also

need to set up tests to make sure that

my Indian environment is working

correctly so it's another part of

effectively managing and governance from

a micro services perspective that's

changed

previously with monoliths we had much

more complex testing environments and

you really couldn't test them till it

was all together in one place here with

the micro services I can unit test and

then also test in the end before

deploying to production the last thing

that I want to touch on here is there's

some really good and guidance to think

about this overall journey and the 12

factor a methodology was created by sand

developers originally from Heroku who

observed the key characteristics of

applications that were successful and

running in a cloud environment and we

don't have time in this session to go

through each of these item by item but

there's a few that I'll call out one of

the ones is something that David

mentioned earlier which would regards to

configuration a traditional Enterprise

application will often have property

files you know all over the place and

that was one of the things that made

deploying these tricky from environment

to environment successful cloud

environment deployments are based upon a

different approach the code is one

version of code without any

configuration inside and then the

configuration that's needed for a

particular deployment whether it's QA or

integration testing or production those

configuration attributes are pulled in

through the environment something else

so we're talking about the difference of

environments is if you take a look at

item number 10 which is that

paraty it's really important in this

model to have a consistent view of what

is used for development and for staging

and for production because small

differences will sort of create this it

works on my machine but then it doesn't

work in production so it's very key when

you're going to this cloud model to keep

and use the capability is to have the

same technology stack across a

deployment and the last one I like to

call out it's really important is item

number 11 which is logs so logging is

something that you do differently in the

cloud environment

we no longer log to local files instead

since these containers are ephemeral and

will often have temporary storage

locally what you need to do is you need

to log to an event stream and the

easiest way to do this is to write

unbuffered the vents out to standard out

and then the container orchestration

system will capture that and either

bring it into a local environment or you

can also send this out to one of many

log and lot of analysis services so this

is our last diagram that we've got in

terms of the journey so we talked about

replay repackage Andry platform and

repackage and this last example is

refactor and in particular we talked

about a methodology that we've seen very

successful with clients something called

the strangler pattern and a strangler

pattern draws from the sense of if

you've ever seen one of these creeping

vines that grows up inside of a tree and

around the tree and eventually there's

so much mine that there's hardly any

tree left to be seen this is an approach

that sort of draws on that and what it

does is we start with the starting

monolith and we look at a logical module

biological module basis so for example

here I've got a war file which I've

taken that war file and I've refactored

that war file into a web app plus some

micro services and what I will do is the

original app traffic is now going to be

proxied into instead of running on the

war file it'll be proxied over into the

cloud environment and this is the way to

do a piecewise approach where you take a

chunk

functionality and you bring it over to

the cloud environment implemented as a

set of micro services and then route the

traffic for that part of the request in

the application over to that component

and then you complete this step for the

other logical components and eventually

you'll have migrated the application

over over time and there won't be any

parts of that original ear file left the

service the application in less

### CI/CD and DevOps

Hi everyone, my name is Sai Vennam

and I'm with the IBM Cloud team.

Welcome to another episode of Cloud Lab.

Today I'm joined by Matt Perrins.

Matt, do you want to introduce yourself?

My name is Matt Perrins, I'm one of the lead engineers

in the IBM Garage for Cloud,

and I'm focused at helping clients

drive innovation with IBM Cloud and OpenShift.

OK, so for our viewers that are hearing about the

Garage for the first time, can you quickly explain what the the garage team does?

Yeah, what we've optimized to do is

help customers deliver first-of-a-kind solutions using IBM Cloud

and we drive this through what we call a 6-week MVPs

where we go into a design thinking workshop,

we rapidly extract the needs for the business,

and then we turn it into a very agile, quick execution for cloud-native delivery.

As a developer, you know, I think that's great,

with container technology, the fact that

you can build once and run anywhere.

So, for IT leaders, where does that decision come up

when deciding to move forward with Kubernetes?

How do they decide how to move forward with cloud providers being an option

versus running something on-premises and the different options out there?

So, we're seeing absolutely the the advent of multi-cloud.

Every client I visit has got more than one cloud - if not two.

They're struggling with the dilemma of how to modernize legacy applications on-premises,

and then there also have certain business models that work better in the public cloud.

So, they as an IT leader, they want to make architectural decisions

around technology that will future-proof them

from either a vendor lock-in or future-proof them for their ability to scale, and

the the component of building a microservice into a container and then orchestrating it

in a common managed platform like Kubernetes

and then making the decision to either run it

locally on your local environments or out in a public cloud

or put it closer to a piece of business

function that may be on a specific cloud that you need to interlock with.

So, it's giving IT leaders a very much

common playing field for delivering software.

How do we make this easier?

Especially as a company like IBM, we need to really think about our customers

and the fact that they do have so many choices

- so how are we making it a little bit easier for them to get started?

So, I'd say that this is one of the key motivations for IBM's

acquisition of Red Hat and the provisioning of OpenShift.

One of the real value propositions of OpenShift

is that it's making Kubernetes development a lot easier

and making it more straightforward.

I'll use the analogy of a chef with a really sharp knife.

If you think of raw vanilla, Kubernetes is a very sharp knife

- you need to be highly trained to know how to use it

to make the right incision, to make the right execution,

and what OpenShift is doing is is making it simpler for

for mere mortals to approach that development platform

and it's smoothing off a lot of the rough edges or sharp edges.

So, you're less likely to hurt yourself when your ...

Maybe adding a handle to the sharp knife.

Yeah, putting a handle on it, put it in a case, having a safe place to store it ...

I know we can keep elaborating on that analogy but

You know, if you don't know what you're doing

and you come up to Kubernetes, you can -

you can get lost, or find it hard to work with.

And OpenShift's job

is to make the developer experience as smooth and as easy as possible,

and they've done a really good job of embedding the tools and

bringing the lifecycle around how you build cloud-native applications onto that platform

I think we want to drill into one of these topics, so I think a

core part of doing cloud-native is

DevOps, and figuring out how to how to

get through that DevOps lifecycle.

So, I know that, Matt, you work quite a bit

especially I know you work with

a lot of customers through the garage

process and I know that DevOps is a big

concern for a lot of these customers so

you talked a little bit about why that's

such a core part of cloud-native and

what you're doing to kind of help

alleviate some of those concerns when

you're prototyping some of these capabilities?

Right, so, as we talked previously about

breaking software into smaller components and packaging them up

one of the key things you need is discipline,

and a discipline in consistency.

I think the other aspect is less humans touching things.

So automation, validation, testing.

So, one of the things DevOps really brings to this cloud-native

development lifecycle is it allows to

put a robust repeatable process so that

when you make a code change you know the

thing that you arrive at is consistently

the same every time and that through

that process of DevOps you're validating

that it's code quality is high enough,

it's tested, it scales, it does all the things you need.

So, when you take that component

and you put it out ready for customers

to use you know that it's offering the

flexibility that you need.

Right, exactly.

So, for that dev team or

the ops team instead of them having to

from-scratch figure out that entire

DevOps pipeline and tool chain flow they

can go with something like where with

OpenShift, some of those capabilities

are already embedded?

Yeah.

But then you have that experience around it to make it a little bit easier.

That's correct.

So, I know that we could talk and talk

about DevOps for the next hour but instead

let's actually go to the cloud lab and sketch out some of these capabilities.

I want to start with laying out an example

cloud-native application that I've architected

and I know how to build it out.

So, let's start with the front-end.

We'll call this the UI portion here.

Below that we've got the BFF ("Back-end For Front-end").

So, this is serving the API's for that UI to serve up information.

So, the UI accesses the BFF

and that, in turn, is going to access

the microservice or the back-end layer.

So, in here let's say "back end".

Now, obviously for higher value services -

let's say that back-end

goes out to something like AI capabilities,

and in addition, maybe a database.

So, Matt as the expert, I'm going to hand this off to you.

This is the application architecture that I want.

How do I start migrating this over to a cloud-native approach

and what are the DevOps considerations

that I need to take into account?

Ok. So, you've already laid out some of the separation of concerns.

You've got a component that is focused on delivering a user experience,

which, again, can be containerized and packaged.

You've then maybe got a back-end for front-end which is serving

UI-friendly APIs

and abstracting and orchestrating across a number of back-end.

So, you've got your 3 logical points.

So, moving forward, what you typically do is take

this component and start to break it into a pipeline

that will enable you to offer some discipline

around how you build, deploy, and test.

So, what we typically do here is we're going to use DevOps

and we're going to create a pipeline,

and this pipeline is going to consist of a number of stages

that will take us through

the lifecycle of building and packaging this component.

So, typically the first step is to clone the code

from your source code management, which is typically Git

or some kind of Git-based technology, GitHub, GitLab,

and then the next step is to build the app.

So, "Build App".

In this portion, when you're actually building out the application,

you have considerations for a Node.js app,

you have things like NPM,

Java, you have to figure out the build process for that.

So, the pipeline is kind of configured

to build each one of these components

based on the programming language?

Right. So, typically you have one pipeline per component

and, as you correctly stated,

if you're building a UI and it's got React in it,

you're going to use a web pack to build the UI TypeScript code,

package that into a form

that will then be package-ready for run.

So, there are steps

- and, again, with a Spring app,

a Spring Boot app, you'll package it using Maven or Gradle,

and we know that Node.js you'd use NPM and various other steps.

So, this part of the pipeline

is about packaging the source code in the way that it's needed

to then be run.

But then, typically, at this point the next step is to to run a set of tests.

So, you run a set of unit tests against the code,

you validate code coverage.

And then this enables you to determine

whether any code changes that have been made in the pipeline are valid.

And again, these steps are sequentially moving along,

but if any one of these fails it will stop the build,

you'll be informed of it as a developer

and then you'll go back and fix the code

or fix the test.

So, just to clarify at this level we're going to do

unit tests, so tests within kind of the app context.

Not really considering

connections between the different components.

Yeah. Today we're not going to cover that the integration story

or performance testing,

but typically when you're building a pipeline you need to

test the code that you've written

using various techniques.

Typically, you can use test-driven development which is a concept we use in the Garage.

So, you write the test first and then create the code to validate that.

You can use other frameworks,

most of the major programming models have good test frameworks around them,

whether it's Java, Node, or other languages.

So, next step:

again, one of the key things to try and drive for

is to get to a point of continuous delivery.

This is a continuous integration pipeline,

but if you fail the test then that's going to prevent

this package of code moving into a test environment.

So, another common technique we use is code scanning,

or vulnerability scanning, or security scanning.

So, what we do here is we're looking for vulnerabilities,

we're looking for test coverage,

we're looking for quality gates.

So, if your code isn't a good enough quality,

from a code analysis perspective,

we could actually stop the build and say we're not going to move

this microservice further along the build process.

Right. So, if we were building out this

- let's say the BFF application was

a container-based application running in IKS (IBM Cloud Kubernetes Service),

We have some capabilities to allow you to test for that scanning, right?

It's the Vulnerability Advisor.

So, would that exist in this phase then?

So, you tested the code, then you...

Yeah. Again, I'm lumping in one or two different stages here,

you can do vulnerability scan, you can do code scan,

it's kind of a common technique to make sure.

The good thing about vulnerability scanning is you're validating

that there's no security holes in the Docker image,

or the container image as you build it.

Got it. OK.

So, now that we've got up to the scanning phase,

what's our next phase - where are we going?

The next step is

to take the application that we built and tested and scanned,

and now we're gonna build it into an image.

So, we call it a "build image".

So, what this is doing is using the tools

to package up the code that we built and put it inside a container.

And once we've built the image

we then store that image out in an image registry

with a tagged version that goes with it.

Right. So, I guess I got ahead of that right there

- so, that's where we would actually do that vulnerability scanning:

once we've tested the code itself,

done some some scanning at that level,

once we build the image then, something like vulnerability advisors ...

Right. So, you could have that as another stage,

but, again, if the vulnerability is poor

then you could prevent this moving forward

and that will inform the developers

to either upgrade the level of base images they're using

or fix a number of the packages that they've included in it.

So, basically every step of the way

- if anything fails you're notified of that

and you can go back and fix that.

Right - and at the next stage, now you have an image, and the next thing is to deploy it.

So, what we're looking to do is to take that image and deploy it inside an OpenShift managed platform

so it will move the container from the image registry and deploy it.

And there are a number of different techniques for deployment that's are used.

Some developers are using Helm,

but the more modern approach is to use operators,

so there's a life cycle around that component when it gets deployed.

So, and then this deploy -

let's say I have a a Kubernetes environment -

so you would deploy an application,

let's say the BFF application,

into that Kubernetes environment, right?. Yep.

OK, and I'm guessing at this phase this is still part of the developer flow,

- would this be the development environment that you're pushing into, or the test environment?

So, typically a continuous integration flow

builds and packages the code up for the development environment.

When we talk in a few seconds we'll more talk a bit more about

how we move that package of code from the container registry

out into a test environment.

Got it, so right here, like that. Yep.

So, the final step is to validate the health.

So, what you're really asking here is,

"Is the container running?"

- is it sending back operational information

such that you can determine that it's healthy enough

to validate that, not only that the tests have run,

but actually it started,

and it's communicating with its dependent services,

and it's going to operate in the way that you'd expect it to.

Of course, yeah. So, this is where you

connect it up to the different components

and make sure they're all working together seamlessly.

This is where you would probably

find issues with integration,

or how the teams are connecting up with each other,

API contracts, and those kind of things,

those issues will start to bubble up in this space.

Yes, and again, the health input is important

because you can hook that into operational tools

like Sysdig and LogDNA and

other monitoring that will give you

a better feel of the current state of your applications as they run.

So, this has got us as far through the development cycle.

The next step is to -

and, again, introduce - this is starting to be common in the industry,

is to use a technique called GitOps

where you would now say

I've got my application, I built it, I packaged it, I've tested it,

I've validated it.

What I'm now going to do is update a Git repo

with the build number, the tagged version,

and the reference point to the image registry.

And then GitOps can then trigger off a deployment of that image

out into a test environment

with all the other components that go with it,

and there are a number of GitOps tools out in the market

and one of the ones we use in the Garage is Argo CD,

which allows you to monitor a webhook of a Git repo

and then it will pull the image,

it will pull the deployment reference, and then package it and deploy it

ready for use and testing.

So, basically the same quality that developers have been doing

forever with SCMs to manage different versions of their code,

now operations team are taking advantage of that same approach

to basically operationalize the deployment

of these actual images, containers, applications.

Absolutely, and it comes back to a point we made earlier,

that this is about discipline and repeatability.

There's no humans hurt in this process as you go through it,

and the less humans touching these steps the better.

Again, one of the things we often do with clients is

we'll work with them

and we'll discover that there's some human process in the middle

and that really slows down your ability to execute.

So, it's about automation, discipline, and repeatability,

and if you can get to this point

and prove that this code is good enough to run in production,

you can then start to move towards

that golden milestone of delivering continuous delivery.

Right. So, once you've automated all of this,

that's when you can truly say you have CI/CD.

That's that's when you can finally get to that level.

OK, so, honestly Matt, this was a great overview of all the concepts we've discussed already.

Let's head back and actually dive in a little

bit and see how some of these

capabilities look in IBM Cloud or

potentially even OpenShift. Yeah, great.

All right, so that was great to see how

all of those concepts are laid out in

the cloud lab, but let's take it a step

further and see how some of these

capabilities actually look.

OK, so what I've got here is a managed OpenShift

environment running on the IBM Cloud

I've come into the OpenShift

console and from here I can navigate to

my application console where I can

manage my different projects and my

different namespaces so the way I've set

this up is I've got a development

namespace or project and within there I

have a number of build pipelines that

are managing my source code and taking

me through the different stages from

build, test, scan, deployment and

health check, which we discussed earlier and so

taking a look at this it looks like if I

had to guess for each of the individual

applications that are running in the dev

environment you've got kind of their own

pipelines built out and it even shows

you how long each step takes and that

kind of thing, source repository,

- OK, so this is kind of a representation

all of the back-end pipeline logic that's going on

Do you want to a drill in little bit deeper and show show me what this is powered by?

Yeah, so again Red Hat is predominant in using

open source open source components.

Red Hat is a very popular continuous integration technology.

They've put a very nice interface,

which is what I meant earlier when we talked about

making things easier and approachable.

If I actually drill into a particular build

I can actually see the detail - when it

completed, how long it took, and then if I

jump into a log I'm now dropping into

the individual Jenkins environment and

I'm now working with the native Jenkins that's installed.

So, clearly, there's a tight integration between the Jenkins capability

as well as OpenShift that's that's allowing us to

have OpenShift show us this broader view of

the pipeline but then if we wanted to at

the end of the day we could dive in

deeper and we see you have the Jenkins

capability loaded here.

So, this is kind of the raw, open source pipeline tool

that's that's powering OpenShift.

Yeah, and I can drop into the logging, I can go and look at

- if there's a failure I can go and look at the different environments and

and again one of the nice benefits of

Jenkins is it has a rich ecosystem of

plugins and it's been extended to

support Kubernetes to do deployment and

management in that environment.

That's great to see, so I know that

for this particular example we're

using Jenkins to deploy

Kubernetes or OpenShift-based applications

- what if we wanted to deploy something entirely different?

Say we wanted to work with Cloud Foundry applications

or a serverless function that we wanted to deploy

- is that something that we can also leverage,

the OpenShift DevOps pipelines to do?

The pipeline's are are optimized in the OpenShift environment to work with

containers that run on OpenShift.

You could extend them to manage serverless technology, or you could build a

Cloud Foundry and then target it

to a Cloud Foundry environment if you

wanted to but there are other DevOps

tools in the family around the IBM Cloud

that are more optimized for doing that.

All right, that was a great demo I think

it was really good for us to be able to

see how some of these components

actually play out in the OpenShift offering in IBM Cloud.

I want to thank you Matt for joining us today for this cloud-native webinar.

if you want to learn more or get started yourself

check out some of the related links that we have attached

and be sure to stay tuned for more videos like this in the future.

Thank you.

#### [Lab - Cloud-Native Practices](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/bd64ccdf56ad4ea1afe870e26d583038/e206de186b7a4519a63f01aab09bf60c/)

# Lab Instructions

If you would like to gain hands-on experience with the cloud native practices mentioned in this section, click the following tutorial link:

* Cloud Native Bootcamp  
  <https://cloudnative101.dev/>
* Comparing Workflows  
  <https://www.atlassian.com/git/tutorials/comparing-workflows>
* Getting started with microservices using Spring Boot and Cloudant  
  <https://developer.ibm.com/tutorials/getting-started-with-microservices-using-spring-boot-and-cloudant/>

#### [Module Summary](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/bd64ccdf56ad4ea1afe870e26d583038/e02c88538f244f64a8eb45b0ab6e3d1c/)

# Summary

In this lesson you learned about:

* + Leveraging cloud development best practices.
  + Using frameworks including containers, microservices and serverless.
  + Automating application delivering with CI/CD.

Dans cette leçon, vous avez appris :

o Tirer parti des meilleures pratiques en matière de développement du cloud.

o Utiliser des cadres de travail comprenant des conteneurs, des micro-services et des systèmes sans serveur.

o Automatiser la livraison des applications avec CI/CD.

# For more information

For additional information, you may be interested in...

* Best practices for CI/CD pipelines  
  <https://developer.ibm.com/technologies/containers/articles/ibm-best-practices-for-the-kubernetes-cicd-secure-container-image-pipeline/>
* IBM Cloud Learn Hub: DevOps  
  <https://www.ibm.com/cloud/learn/devops-a-complete-guide>
* DevOps toolchains  
  <https://www.ibm.com/cloud/architecture/toolchains/>
* IBM Cloud continuous delivery  
  <https://www.ibm.com/garage/method/practices/deliver/tool_continuous_delivery/>

### Module 4 - Multicloud Management

#### [Learning Objectives](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/89227024130b43f684d95376901b65c8/f739d53a78f34915b8e0c4e2996ca5a9/)

#### In this lesson you will learn about:

* + - Using tools to manage applications.
    - Observing multiple applications.
    - Managing security and compliance.

Dans cette leçon, vous en apprendrez plus sur :

* L'utilisation des outils de gestion des demandes.
* L'observation de multiples applications.
* Gérer la sécurité et la conformité.

#### [Lesson 1 - Application Management](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/89227024130b43f684d95376901b65c8/aed4761a25e64d57af95da56a15ac609/)

## Lesson Introduction - Application Management

This lesson contains two sections, as follows:

**Managing Applications**

The ***Managing Applications***section contains video sessions, as follows:  
- **Provision and deploy to multiple cloud providers**explains how to benefit from the IBM Multicloud Manager.

**- Simplify application monitoring with SRE Golden Signals** discusses how you can simplify monitoring for complex modern applications.

***Cloud Paks***

The ***Cloud Paks***section contains video sessions, as follows:

**- Cloud Paks Explained** helps you understand how a packaged, containerized solution can help you create cloud native applications and modernize existing workloads.

**- Why Cloud Paks on IBM Cloud**helps you understand some of the tools that can help you overcome roadblocks as you move to the cloud.   
**- Simplify Modernization with IBM Cloud Paks for Applications**illustrates how this packaged solution accelerates the process of building of cloud-native applications by leveraging built-in developer tools and processes including support for microservices and serverless computing.

Cette leçon comprend deux sections, comme suit :

Gestion des demandes

La section Gestion des candidatures contient des sessions vidéo, comme suit :

- Provisionner et déployer vers plusieurs fournisseurs de cloud explique comment bénéficier du Multicloud Manager d'IBM.

- Simplifier la surveillance des applications avec les Signaux d'or de SRE explique comment simplifier la surveillance des applications modernes complexes.

Cloud Paks

La section "Cloud Paks" contient des sessions vidéo, comme suit :

- Cloud Paks Expliqué vous aide à comprendre comment une solution packagée et conteneurisée peut vous aider à créer des applications natives dans le nuage et à moderniser les charges de travail existantes.

- Pourquoi les Cloud Paks sur IBM Cloud vous aide à comprendre certains des outils qui peuvent vous aider à surmonter les obstacles lors de votre passage au cloud.

- Simplifier la modernisation avec IBM Cloud Paks for Applications illustre comment cette solution packagée accélère le processus de création d'applications natives du cloud en tirant parti des outils et processus de développement intégrés, notamment la prise en charge des micro-services et de l'informatique sans serveur.

## Managing Applications

### Provision and deploy to multiple cloud providers

Hey everybody, welcome to this demonstration.

Today we're gonna be showing you

how we can leverage IBM's multicloud management platform

to both provision infrastructure as code

through a terraform service composer that IBM provides as a capability on this platform

as well as how we would manage cloud-native deployments of Kubernetes Helm charts

to multiple cloud service providers

and so you can see here I have a multicloud management cluster

where I'm managing an IBM federal cluster, a Microsoft Azure IKS instance, an AWS ek

instance, and 4 other IBM cloud private clusters.

This really gives us a single control plane that

allows us to have a perspective across all of these clusters

and we can seen real time resource utilization.

we can filter on this information based on the specific cluster that we want to manage

and, for example, I can click on Azure here

I can see details about compute utilization,

the number of pods that are running, you know, whether this cluster is compliant

because we can build compliancy policies into these clusters the number of pods

and the resource utilization whether they're running whether something has

failed and really just giving that administrative dashboard of what's going

on across all of our clusters and I have multiple clusters running here so one of

my clusters is running the service composer capability and you're gonna see

that here that is this MOU MCM MOU cluster that I have running in the

Toronto data center and I'm gonna jump over to that catalog here on on mo and

you're gonna see that I've got my catalog pulled up for this cluster and

there's a bunch of different catalog things that come with IBM cloud private

anything with a blue checkmark is called our certified cloud packs those are

really cloud packs that are built for us for the full staff to to include

metering monitoring logging patching and upgrading features things like that what

I'm gonna jump to is first we're gonna show you that traditional be in

placement so I've created a service called mean and this is going to get

provisioned to AWS so one of the differentiators about ICP is we can

provision infrastructure as code and manage infrastructure as

what's from the same catalogue that we manage our kubernetes deployments and so

this service gets designed and developed by our service composer and so you can

see here this this service that we have was created and published and then it

becomes available in our catalog but in order to create these we'll create a

service and just show you how this gets done really quick and so let's say I

just want to call this my AWS stack and then I'm gonna pick a category we can

add categories to kind of filter out what types of service this is I've got a

cloud services category we'll select that I'll hit create and then you know

if I want to give it a fancy icon I can put a short description in here and then

a long description an AWS staff that only gets angry if you feed it after

midnight right because this is a mean stack so we can jump over to our

composition and then this is where we would actually design what the flow of

this service is that we want to create and make available in our catalog so I'm

gonna jump down to AWS here and we're gonna scroll down until we see that mean

stack that we want and so I'm just click and drag and I'll put that on a process

diagram so from there basically the parameters that you would have to fill

in manually you can pre prescribe those and so if we look over here you can see

that there are specific parameters that you would have to prescribe in order to

provision this means that what do you want to call the database instance what

do you want to call the node.js instance what's our public SSH key that we're

going to pass those types of parameters and those would be preset for an end

user that just wants to get access to the VM the other thing you could do is

introduce rest hook calls you could introduce a decision and then based on

that decision we're gonna send an email notification if Cass or if they'll that

type of stuff and so very powerful service composer for what it means to

provision your infrastructure and adding that automation into not just

provisioning the infrastructure resources but the

business processes that go along with it so all of the source code gets generated

for you you can import existing source code for for terraform if you have them

we would save this and then publish it when we do that that's when this service

becomes available in our catalog and that allows an end user to come in

provision this resource so let's show an example of that I'm gonna select mean

AWS and I'm just gonna select you can create plans like hey maybe this is the

T - medium or in five extra-large t-shirt size for AWS those types of

plans that can pass parameters into the service when we do provisioning I'm

gonna click configure and then to an end user all they have to do is give it a

name and then select a name space not collect services and then they'll click

install so that's gonna fire off a process and what it's gonna do is it's

gonna kick off that service which is gonna provision an instance in ec2 so

I'll come in here and I'll click on launch that instance and we can see that

instance is now in progress of coming online if I jump over my ec2 environment

here you're gonna be able to see if I hit refresh and now you can see that

there's two new instances coming online the powerful thing is is that IBM cloud

private is gonna manage that layer of abstraction for the end user when they

want to provision this resource you'll have operations management teams and IT

administrators that will help in the development and management of these

infrastructure instances but to an end-user they want to be able to consume

that service that's been created through the service composer and so while that's

coming online we'll go ahead and check out some of the other features of IBM

cloud private for managing traditional infrastructure so I'm gonna jump back

over here and we're gonna just take a look at our monitoring capabilities now

IBM cloud private comes with installed with an out staff

Prometheus for performance monitoring and grow fauna and this dashboard

basically what I've done is I've created a cloud watch database for graph on ax

that will allow me to monitor my AWS ec2 instances and you

actually see that those new instances that are coming online are now available

to be performance monitored and so we can start to take a look at what types

of performance monitoring we could see from a single dashboard across a region

or across your ec2 instances so how would we assess some of the costs

associated with these instances and so we can do that through IBM's cloud

private cloud brokerage solution called cost and asset manager and this solution

also running on ICP gives us the ability to assess some of the cost of our

resources this is across all three clouds that were integrated with here

from AWS to IBM cloud and Microsoft Azure and we can filter on those we can

get a timeline of our cost of those infrastructure resources we can

highlight infrastructure resources and pass this billing information to a

billing tool for example by exporting a CSV the other things that we can do is

filter so we can filter on you know a date of all of our databases and we can

see the cost of what our databases are or network services we can see what the

cost of our network service is over a time line the other thing we can do is a

heat map and this is really interesting because this gives me the ability to

take a look at my AWS account for example where am i spending money and

what is the utilization as it relates to that cost and so in this case you can

see an example that I've got an ec2 instance running out there that's

costing me 135 dollars a month but it's only being utilized five and a half

percent of the time so I can either reduce or descale the compute resources

on that or I can build in auto scaling policies that could allow us to take

much more effective use of that instance because if I look over here I've got

another instance that's actually being used 98% of the time but it's only

costing me $14 a month so that's a pretty dramatic gap between those two

things for me to start to investigate might be helps to be able to have some

of the cost savings the other thing we can look at is policies so policies we

create to you know signal what the age of an asset is and this is showing us

you know when this asset was issued what the age of that asset is and then how

much that asset is actually costing us and all of this information is gonna

feed into our insights capability and so if I jump over here to insights we can

now see using Watson we can start to make in black Watson help us make

intelligent decisions on what we are to do with these instances that are either

under underutilized or outdated and so for example I've got assets that it's

recommending hey we could repurpose or refund these assets and if we do then we

could save as much as two thousand five hundred and twenty nine dollars a month

there's other examples that if I were to delete an asset we could delete assets

and save six hundred and sixty six dollars a month so Watson's gonna start

to build some intelligence on what types of signals that's getting for your

infrastructure resources and make recommendations based on those signals

ultimately saving our clients cost and money saving our clients money if we

jump back over we can go to our cloud automation manager we can check our ec2

instances we can see those are online and we'll jump back over to cam and we

can see that it's active and so now that the instance is running we can get the

details about that instances from an operational management perspective or an

IT administrator and that's just going to show them what was provisioned what

were the characteristics of that provisioning exercise and then how am I

going to actually access it so all of this information could be passed into an

email that could take the IP address pass that into the knee mail and it'll

send that to the developer for whoever requested that instance and then he

could pop that into the browser and now he's ready to go and start doing some

development with a simple stack so really interesting capability that

allows us to manage infrastructure from any cloud service provider from a

centralized control plane and simplify the end-user experience

through a self-service catalog and of course the last thing I can do certainly

is I can come to my brokerage service and I can just simply remove and delete

that instance so when I'm done with it as long as I got the right role based

permissions I can then deep provision it and then

that no longer is costing me money because I don't need to keep it up and

running we'll give it a couple seconds here and you'll actually see that it is

gonna start shutting those instances down okay so let's jump over to the

other use case from a cloud native deployment perspective we can now start

to look at leveraging this multi cloud manager in a single catalogue to deploy

applications and I can leverage one Hjelm chart that can deploy to any

kubernetes service so in this example I'm actually going to show how we can

deploy a simple node.js app to three different cloud service providers at the

same time and so I'm gonna come down we're gonna select version 1.2 because I

want later show you how some of the patching and updating works we'll click

configure I'm gonna give it a name I'll select the name space will click MCM MOU

which is the target cluster we were using earlier and accept the license and

what's really interesting about MCM is now I have the ability to not just do a

local install to the MCM cluster but I can remotely install into any other

cluster that I'm connected to and so I'm gonna select AWS I'll select Microsoft

Azure and then we'll select MOU those are three different cloud service

providers clusters these are public cloud service services for kubernetes

and this is a private kubernetes cloud service I'll click install so that's

going to deploy our nodejs app out to all three cloud service providers if I

jump over here to my terminal you can see that I have a kubernetes connection

to my ec2 instance I have the three nodes there and we can do a COO

and that's targeting my MC MMO namespace where I deployed that nodejs app I'm

gonna click on that and there's our demo node one and you can see that it's

running it's available and it's been up for 31 seconds

we can also jump over here to a juror since I've got a cluster running out

there we will get the same type of information how many nodes do we have we

should have two and then we'll do a and go get that information and there it is

it's also running it's been up for 69 seconds so I'm gonna jump back over to

our cluster that was MC mo where we deployed that third to that third cloud

private cloud cluster and we're gonna take a look at our house early PSA's and

we're gonna see that we deployed that nodejs to this MCM cluster and there it

is but now we're going to demonstrate some of the patching and upgrading

features of ICP and so we've gotten a 1.20 the year and a new available

version was released 1.21 I'm gonna step into this and take a look at this

application that we deployed one of the first things I noticed is I have a new

revert vailable version the second thing I notice is that I actually had an error

and so in this use case we tried to deploy a version that didn't work and so

what I can do to resolve this issue is the developer told me he released a

patch that then I can upgrade to the newest patched version and it's gonna

fix the issue and so I'll click upgrade I'll select the version that I want and

then we'll click upgrade and what that's gonna do is that's gonna roll out the

update and it's gonna start creating the container and it's gonna track for me

what release is currently running and what the previous release was if

something fails here and I had a running application before I could just hit

rollback and go to the last known good working version and so there's some

there's some capabilities for a delivery pipeline here that can be leveraged

right out of the box additionally what we can do since IBM cloud private

deploys and Hulk stack is we can seamlessly traverse a link

to our Cabana log files as it relates to that specific pod and so I'm gonna

traverse to that log file and then we can start to filter on what type of log

information are we interested in about this pod do we know the pod IP what

namespace it's a part of are there any tags associated with it

and then finally what is the actual relative log information in context to

that so really being able to consistently view log information in

context to a pod and the ability of course to search on all other of the log

information that might be available as part of your kubernetes clusters very

powerful capability we can see that that demo app is running it's available if

you wanted to scale it out which auto scaling policies can be built into a

home chart but we can manually do it here and say oh I want to provision

three instances of this application to make it highly available the other thing

we can do is come over here to our web terminal and connect right to a terminal

using the web interface you can see our nodes here we'll get pods same thing and

now we see our demo app running there we go and so we just took a node.js

application and we deployed it successfully demonstrating some patching

up great features of ICP to three different cloud service

providers at the same time the powerful thing about multi cloud manager is it's

not going to be hindered based on the kubernetes service that you're using

from any third-party kubernetes service or cloud service providers kubernetes

service giving you that central level of control and wrapping your arms around

managing multiple kubernetes clusters the last thing we'll do is take a look

at our topology diagram the topology diagram is really powerful for your IT

administrators to give you that architectural representation of all of

the relationships between you know your deployments your pods the containers the

hosts for the nodes that those deployments and pods are

on the service and we're you know for example a access to the Internet has

been established and so you can filter on these here on the diagram and show

different types of information and this is a real-time diagram so you'll

actually see it moving and updating as things are coming online or being

removed and deleted and we can take a snapshot of this and put it into an

architectural diagram that represents a specific cluster and the applications

running out there so just a very powerful tool of giving you real-time

insight of all of the relationships between the application services and

what's running where and so this concludes this demonstration I really

hope you enjoyed it and I hope you have a great day thank you

### Simplify application monitoring with SRE Golden Signals

today I'd like to talk a little about

the site reliability or sre discipline

and how we can apply it to simplifying

monitoring for complex modern

applications this will help us identify

root causes more quickly and drastically

reduce the mean time to recovery so that

we can maintain the end-user performance

that we want for our applications so

first let's take a look at what happens

before we've applied these sre

principles to our monitoring so let's

say that I'm the owner of an application

and I've gotten an alert that says that

I'm having a latency issue now my

application is really critical for this

business and so I need to find the root

cause quickly but because I'm part of

this complex micro service topology it

can be really difficult to figure out

where exactly the root cause is coming

from and to make things more complex all

of my dependencies could be based on

different technologies so let's say one

is built on nodejs one is a db2 database

another is written in Swift and so on

now all of these have different metrics

that are typically monitored and I may

not be an expert in any of these

different technologies so it may be

difficult for me personally to go in and

figure out what the problem is so I

would have to call in a expert for each

of these technologies now as you can

imagine this is time consuming for

everyone to go through their service

figure out if there is a problem or if I

need to keep going downstream and all

the while my users are still

experiencing this latency issue now what

if there was a better way this is what

we can learn from the SRA discipline

which tells us that there's really only

four key performance indicators that we

need to monitor not all the different

metrics for each technology and we call

these golden signals

so the golden signals are latency which

is the time it takes to service a

request errors which is a view of the

request error rate traffic which is the

demand placed on the system and

saturation which is our utilization

versus max capacity now let's go back to

our initial example and see how this

would work applying the golden signals

so my service will call it service a we

know we have a latency issue now we know

that latency is typically a symptom and

if we examine the service let's say

we're not seeing any of the causes so we

know we have to keep looking downstream

but we don't want to go back to this

complicated micro service topology and

try and figure it all out

so some APM tools can help you out with

this by identifying only the services

that are one hop away from my service in

question so let's say we have services B

C and D that are connected to my service

a that's having the problem now no

matter what technology these services

are built on all we need to do is go

look at the golden signals so let's say

we look at the golden signals for

service B and everything looks fine so

we know service B is not the problem and

let's say service C same scenario we

don't see any issues so we can eliminate

that as the problem now service D let's

say that we're seeing an issue with our

saturation which is trending upwards so

right there after only a few minutes

we've identified service D is likely our

root cause so now instead of having to

pull in the experts for each of these

different services now we can go

directly to service D and let them know

that we've identified that they're

likely a cause of this issue that we're

having and they can go about fixing it

and what's even better is if they're

using golden signals to monitor their

service

it's very likely they've already

identified this and are already working

on the fix so as you can see this

process drastically improves the time

that it takes to go through this complex

topology and many different technologies

to figure out where your root causes and

identify exactly how to fix it so when

you're identifying an APM tool to use

make sure that it offers the ability to

use these golden signals and this one

hop dependency view so that you can

quickly identify the root causes and get

your service restored as quickly as

possible

thanks for watching this video on

simplifying monitoring for modern

applications

you

## Cloud Paks

Hi everyone, my name is Sai Vennam

and I'm with the IBM Cloud team.

Did you know that, according to a recent study,

only 20% of enterprise applications

have moved to the cloud?

Let's talk about why exactly that is.

Well, we know that the container technology

Is crucial for making that transition to the cloud,

but even with businesses adopting it

they're still running into a number of roadblocks.

Say, for example, that a business is creating a new cloud-native application

but the pace at which it is progressing

is much faster than the legacy data layer can scale.

Now, this is just one use case.

IBM has identified key challenges

preventing businesses from moving to the cloud

and created what we are calling IBM Cloud Paks.

Now, let's start building out the stack that supports Cloud Paks.

It starts with Red Hat OpenShift.

Now OpenShift is their flagship container platform

and, essentially,

it's based on container and Kubernetes technology.

So, with Cloud Paks, what we've done is we've taken IBM middleware packaged it up, but more

importantly optimized it to run directly on top of OpenShift. This means it runs best

in an OpenShift or container-based environment.

So up here, we have Cloud Packs. Now since Cloud Packs are supported on OpenShift, now

we have to think about what OpenShift is support, and the great thing is OpenShift is actually

supported anywhere. So, any major public cloud provider or maybe on-prem or a private cloud.

The experience for getting Cloud Paks on IBM Cloud is actually very streamlined and we

will have a future video that goes through that entire experience.

Now, we've built out the ground work here but the last layer that we haven't mentioned

is gonna be the actual applications or the business logic that actually leverages those

Cloud Paks. That's where the user and the customer comes in to actually build out.

Now there's three key tenants to Cloud Paks that I want to outline.

So, for one are they can run anywhere. We talked about this, regardless of where you're

OpenShift platforms are running, so on-prem or public or private clouds, or even running

on IBM Cloud you can run those Cloud Paks in a fully supported fashion anywhere.

Next, we have the idea that they're open as well as secure.

Now we mentioned that their built and optimized for OpenShift which is based on open source

technology and principles. This allows you to avoid vendor lockin and basically run these

applications and logic anywhere.

In addition, it's secure from the ground up, so essentially, we have security patches as

well as updates the platform regardless of whether it's the operating system OpenShift

or the entire set IBM certified middleware through the Cloud Paks.

Now, the last thing that we have here is the fact that they're consumable. Now Cloud Paks

are consumable in the sense that you can not only pay for what you need but also when you need it.

In addition, since we package them up into you these five clear use cases that we have

here, they essentially allow you to consume this particular solution that you need that

is mitigating or preventing you from making the transition to the cloud.

Now we'll start by actually going through each of these different Cloud Paks.

At the top we have Cloud Paks for Apps. Cloud Paks for Apps essentially allows you to, whether

you're creating new cloud-native assets or modernizing maybe an existing application,

it allows you to get started on the right foot. So, creating a new microservice you

have to think maybe wanted to run on any cloud environment in addition taking advantage of

certified and secure run times and capabilities. So, we have the tools to let you do that.

In addition, so you have legacy infrastructure or application workloads, particularly those

based on WebSphere there are prescribed approaches to modernize those.

Next, we have Cloud Paks for Data. Now data is crucial to any application and over the

years and maybe decades businesses have accrued lots and lots of data, but the problem is

the majority of that data is not cataloged or maybe even unaccessible, but businesses

know that if they were to leverage all of that data they can really get new analytics

and insights. Now Cloud Pak for Data gives you a single pane allowing you to connect

up all of those data sources and then starting infusing AI and machine learning to derive

new insights from those capabilities. That's the power of this Cloud Pak.

Next, we have Cloud Pak for Automation. Now every business kind of has these everyday

mundane tasks, things that can be automated. So for example invoices that need to be approved

or documents that need to make sure that they're following the right guidelines. Now, this

Cloud Pak allows you to automate those everyday processes allowing your employees to focus

on those higher value tasks.

Next, we have Cloud Pak for Integration. Now, in previous videos we've talked about this

emerging pattern of applications, essentially applications are that are basically hybrid

cloud or following a hybrid cloud architecture, so apps that are running legacy on-prem as

well as new apps running in the cloud.

Now, those desperate environments to kind of get them to work holistically you may take

advantage of something like APIs and API management, or maybe you're taking advantage of a faster

protocol something like messages with Kafka. Regardless when it comes to integrating these

different solutions and having them work with each other that's where Cloud Pak for Integration comes in.

Last, but not least, we have the Cloud Pak for Multicloud Management. Now, the future

is not just hybrid but also multicloud. This means maybe you're taking advantage of multiple

clouds and multiple public cloud providers. Now in this case there's three major things

to think about, how do you cover all of those environments, so things like security and

compliance policies.

How do you automate deployment of assets to those environments? And finally, how to get

visibility things like metrics and analytics on those apps or just to make sure that they're

healthy. That's where this final Cloud Pak Multicloud Management comes then.

Now bring all of this together we have IBM Cloud Paks, essentially allowing you to reduce

your speed for development as well as your operational expenses.

Thanks for joining me for this quick overview of IBM Cloud Paks. As always you can get started

quickly by creating a free account on IBM Cloud.

If you have any questions or simply like the video, be sure to drop a like or comment below.

Stay tuned for more videos like this in the future.

Thank you.

### Why Cloud Paks on IBM Cloud?

Hi, everyone. My name is Sai Vennam

and I'm with the IBM Cloud team.

Today, I'm excited to show off the all new experience

for using Cloud Paks on IBM Cloud.

Let's start with introducing Cloud Paks.

Essentially, Cloud Paks are containerized solutions

created to help users not only create new cloud-native applications,

but also modernize existing workloads.

Each IBM Cloud Pak includes IBM middleware

and common services for development and operations,

and all of this is on top of a common integration layer, OpenShift.

Cloud Paks are optimized to run on OpenShift,

this means that Cloud Pak is supported wherever OpenShift is supported.

On IBM Cloud, Cloud Paks support a streamlined deployment

directly into OpenShift clusters.

Red Hat OpenShift on IBM Cloud is a fully managed service.

This means a number of things,

like automated cluster provisioning,

multi-zone regions for high availability,

and integrations for logging and monitoring.

First we'll need an OpenShift cluster,

this is straightforward using our managed OpenShift service,

an experience that I've detailed in our other videos.

Let's take a look at my IBM Cloud dashboard.

Here we can see that we've got an OpenShift cluster created and ready to go.

Next, we'll need to push a Cloud Pak into this cluster, to do so we're going

to take advantage of IBM Cloud Schematics, a completely free Terraform

as a Service infrastructure automation tool in IBM Cloud. Yep, that means you can

use the community favorite open-source Terraform to automate IBM Cloud

infrastructure. It starts with defining the

infrastructure that you want a provision in a terraform file. Then you're able to

verify that automation by generating a plan. Finally, you apply that plan to

start provisioning that define infrastructure and software. Schematics

helps you manage this automation moving forward allowing you to track those

resources once you've created them and even make updates, and the best thing

about Schematics we can use it to install Cloud Packs. Let's see how this

experience looks by first navigating into the catalog. The catalog and IBM

Cloud is actually core to the developer experience. Here we've got various

compute options, as well as services. In addition, we have the all new easy-to-use

Terraform templates, as well as Cloud Paks which we'll start with now. We'll

start our demo with Cloud Pak for Data. This Cloud Pak allows you to manage

multiple sources of data and then use AI to extract new insights. First, we'll need

to verify that that OpenShift cluster that I showed you

meets the sizing requirement. It does, so then we'll customize that Schematics

workspace and then run a pre-install script to set up our cluster.

We can even configure some of these variables

and then start the install. Schematics manages our Terraformed based install of

that Cloud Pak through every step of that install phase. Once it's completed

we can access our Cloud Pak for Data dashboard. That's it, you've seen how easy

it is to deploy a Cloud Pack into IBM Cloud. Let's summarize with Red Hat Open

Shift on IBM cloud you keep that familiar OpenShift experience while

we manage the infrastructure, security, and scale. With support for Cloud

Paks that experience is further streamlined, allowing you to install and

start using Cloud Paks with the click of a button. Start building on the next

generation multicloud platform today, leveraging IBM Cloud Paks running on

Red Hat's OpenShift on IBM Cloud.

### Simplify Modernization with IBM Cloud Pak for Applications

I want to share an interesting statistic which I learned just recently from IDC.

Over 500 million applications are going to be built in the next five years, and

that equals the number that have been built over the last 40 years. That

statistics amazing to me, and what that speaks to is the fact that you are

communicating and engaging with your customers through your applications. And

that's why it's so critical that you are able to have the tool set and the

technologies available to you that allow you to get to those applications quickly

no matter where you're at today. So, it's important as you're going through this

journey that you realize that you may have traditional IT, you may have

applications that are running in a three-tier architecture, and they may run

that way for a long time, mission-critical providing a lot of

money for you. But you know that the destination is really cloud native, microservices,

and being able to have applications that are developed in that

manner. So, in this journey that you're on you

will have technology and applications that are in a more traditional

environment and you'll be targeting a newer cloud native world, and you're

going to be anywhere along that continuum. So, what you need is, you need a

set of technologies and the Cloud Pak for Applications allow you to be

wherever you're at in that journey. And that's what's so important is that you

have the flexibility to adapt and to modernize and refactor, and to take on

new things when the time is right for you. So, the Cloud Pak for Applications

has three use cases and those use cases are one, run existing applications where

they are today. You know, don't touch them, don't mess

with them, just run them. They're making a lot of money for me and they're

mission-critical. And that's going to be around for a long

period of time. The second use case is modernized those existing applications

that you have, refactor them. What parts of those can be split into components

that can be microservices and then to be stitched into that broader

application. The third use case is building apps brand-new, from a

microservices, cloud native perspective. And so what you want is you want an environment

that allows you to do all three of those use cases. And

allows you to write, mix your environment, and at the right time move your

applications or create new. So, anyhow, a lot of great things in the Cloud Pak for

Apps and you have a lot of opportunities to explore and to learn more about it.

I invite you if you have questions, please drop us a line below. If you

want to see more videos like this in the future, you can like and subscribe to

this. And don't forget you can always get started on the cloud at no cost by

signing up for a free IBM Cloud account.

#### [Lesson 2 - Observing Multiple Applications](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/89227024130b43f684d95376901b65c8/dcd3e22eb99e4cc9b56ff109f7f904c5/)

## Lesson Introduction - Observing Multiple Applications

This lesson contains video sessions, as follows:

- **Observability Explained** helps you harness and drive new insights as your applications grow in complexity.

- **Modernizing Log Management**explains how LogDNA provides visibility into your infrastructure and helps developers and engineers monitor and debug production issues.

Cette leçon contient des sessions vidéo, comme suit :

- Observability Explained vous aide à exploiter et à susciter de nouvelles idées à mesure que vos applications deviennent plus complexes.

- Modernisation de la gestion des logs explique comment LogDNA fournit une visibilité sur votre infrastructure et aide les développeurs et les ingénieurs à surveiller et à déboguer les problèmes de production.

## Observing Multiple Applications

### Observability Explained

As your applications grow in complexity how do you harness and drive new

insights from all the chaos? And is observability just a buzzword, or is it

something that you actually need to think about? Spoiler alert, it is. My name

is Sai Vennam and I'm with the IBM Cloud team, and today I'm joined with a special

guest. Hi there, I'm Laura Santamaría and I am a Developer Advocate with LogDNA.

If you don't know LogDNA is a core part of our observability story on IBM Cloud,

but today we're gonna be talking about observability, so let's start with

definition. So observability is a property of your systems that helps you

understand what's going on with them, monitor what they're doing, and be able

to get the information you need to troubleshoot. So the way we see it

there's three major tears of observability and let's go through those now.

We're gonna start out with my favorite which is logging. In addition to logging

we additionally have metrics, so that's just all of your analytics around all of

the data that you're gathering and finally...we've got monitoring. Now

monitoring is essentially putting up a magnifying glass to your systems and

getting new insights from what's actually running there. Today we're gonna

be starting with an example,

in the bottom left corner we have sketched out

a few of the different infrastructure pieces so we'll start with today. Can we

explain what those are? Sure, we have a public cloud, it can be any of them. And then

you have on-prem, and then let's say we actually have some user data, maybe this

is a tablet or a cell phone. So all of those infrastructure pieces are creating

and generating data and what I'm kind of gonna focus on here is the personas that

are going to consume them. So we've got that Dev persona,

we've got Ops,

and finally we have Security. So, all of this data flowing in is kind of a lot, I

want to have some way of filtering it down for my specific user personas to be

able to understand it. So let's start with developers,

what do developers care about? I actually want to back up here for a moment though

because let's talk about all the different levels that logging can come

from. So we have three different levels that we can think about so you have your

Operating System, you have Kubernetes or any other type of platform, so I'm

picking kubernetes. That's my favorite. And then finally your application. So your

operating system and kubernetes all send really good logs and you can use a lot

of that data pretty much as this, or add in some of your own but applications is

really where you need to spend some time. So you're devs need to create a proper

event stream and this really goes by the garbage in, garbage out system where you

really need to put in good work and get some good data on the side of the

application so that you get good logs out. Right, exactly, so the great

developers out there on kubernetes and the operating Systems they've

instrumented their platforms but the application that's up to you as a

developer to make sure the instrumentation is in place. Absolutely,

and when you think about it, let's say that we have an operating system here

and I'm gonna say that's an operating system, and then we have kubernetes

running on it. And then you actually have your app running on top of kubernetes.

And all of these are to each sending data. So we have three different levels

of data all coming out and trying to come towards the dev that wants some

information. Right, so it looks like they're all coming into this central

area here. That's right. We can talk about this is our aggregator.

So our aggregator takes in all of this data and puts it all into one place so

we can work with it. That's right, but kind of

coming back to the the problem here a developer might not care about all of

the information flowing in, how do we drive just the pieces that they care

about like we mentioned? Maybe they instrumented their specific application,

how do we drive that to them? Absolutely, so an aggregator often has filters. So in

this case let's say the dev is just asking for data about debugging and just

some information there, and your data, your filter can actually set up a

dashboard or some other way of accessing all of that data that the dev can take a

look at just the pieces that they need. That's a core part of a observability

solution, this aggregator not only does it collect the data but it needs to

externalize it, expose it, so my developers can access it and drive new

insights. So let's say we solved that part of the puzzle,

what do operators care about? What are the operations teams? What are they

looking for out of these systems? So an operations team might need to know more

about degradation of its system, or if a pod is falling over, maybe your database

filled up and you need to know more information about how you can fix it.

The ops teams is going to be getting data from all of these different systems

and filtering it out to yet another dashboard or another interface of some

sort and getting that data just what they need. Right, so potentially they may

not care as much about specific application level logs but they'll be

looking to kubernetes to say hey what was the CPU usage, do we need to set up

some horizontal pod auto scalers to make sure that we don't hit those limits.

Finally, kind of probably see where I'm going here with the last piece of the

puzzle with security, they probably have a dashboard that's created for them as

well. So a security team let's say they're using a third-party tool as most

security teams generally, do they identify a threat ID, or maybe a customer

ID and they want to dive in deeper to a potential threat that's been identified.

So they put that information in the aggregator and they can identify and

make kind of sense of all the chaos to identify exactly what that specific

security analyst might be looking for. But I want to pose an interesting

question here, it's not always about going to the system and identifying

what's there, many times security advisors need to

know what's happening the second it happens and they can't just sit there

and stare at logs all day, right. Absolutely, this is where monitoring

comes in, this is really a two-way street. We have automated alerts that can go out

and tell all of these different groups about specific things that they're

interested in, specific events that they want to know about. So let's say that you

have a system that's been accessed and it's not supposed to be frankly that

system is going to figure it out long before a human is and that's what an

alert is for an ops team doesn't want to find out that there's a degradation of

service when their user does, they need to know ahead of time. So a good

observability solution should have the ability to externalize the data and then

additionally set up a learning on top of that. So our dev team may be their most

comfortable in Slack, so they set up a chat bot so that particular exceptions

when they're thrown they're able to know when they happen. Your ops team may be

they were using something like a paging system so that you know in the middle of

the night if something goes down they get alert and they can start looking

into it right away. And then finally for our security teams, kind of as I

mentioned, they're generally using you know maybe third party tools or custom

dashboards they can set up custom alerting so they can know exactly when

something goes down. And to be honest this is your new norm, you're going to

have multiple clouds, you're going to have on-premise systems, you're going to

have data coming directly in from your users. You need to be able to understand

what's going on and really this is what observability is all about. Thanks for

joining us for this quick overview of observability, also thank you so much for

joining us today Laura. Absolutely, my pleasure. If you have any questions

please drop us a line below. If you want to see more videos like this in the

future, please like and subscribe. And don't forget you can always get started

on the cloud at no cost by signing up for a free IBM Cloud account.

### Modernizing Log Management

So, I am Jason McGee and I am CTO for IBM Cloud platform. I'm Chris Nguyen, I'm

CEO and Co-founder of LogDNA. So great to be here with you Chris, I think we just

got a chance to talk about our partnership and what we're doing. Why

don't you kick off with you know tell us a little bit about LogDNA and what you

guys do, and then we'll talk about kind of our partnership together. Absolutely,

thanks for having me this is a super fun moment. LogDNA, we help developers and

engineers, DevOps teams monitoring and debug their production issues. We aggregate all

the logs, whether it's from servers or OS and platforms and we put into a powerful

platform that gives you full visibility into your infrastructure

which is pretty fun. That's great, yeah so, you know at IBM Cloud we're

trying to build this platform for developers, and logging and

observability are so important for developer productivity, and so you and I

started work together a few years ago to kind of combine the best of what you

guys were doing and bring it to the IBM Cloud platform. It's been really

interesting to me when you tell me a little bit about like the growth that

you guys have seen over the last year. It's been awesome, like we,

I love being a customer of IKS and we see the scale and the

flexibility that it made perfect sense when we met two years ago, your world and

our world goes hand-in-hand. Right. And it's nice, we we were big

believers in multicloud, we're big believers in Kubernetes, and it's nice

that the bets we made two years ago actually paid off, and it's actually the

future what we think about together. Right. I think what was nice is the

simplicity of IBM Cloud match our simplicity of what we think about simple

is also powerful from a usability perspective, from a scale, from a search,

and it's very seamless. That's the best part, is the key to developers, and IBM Cloud

with the power of LogDNA, so it's a great win-win scenario. Yeah, I mean, so

let's talk about, kind of you as a client of IBM Cloud, you know one of the

things that attracted me to LogDNA was really two things. One was this

incredible user experience that you guys have built around developer need for

understanding their environments, and two, especially two or three years ago, very

forward-looking on, kubernetes as a platform for running things.

So, you know what's worked well for you on IBM Cloud? What, do you know, what do you like

about it? I think simplicity to scale. So when we

think about scale we we do a lot, and again I don't know full information but

we do a lot, and we have a pretty lean team that could manage that and scale.

So, I think nowadays with the power of kubernetes you don't have to maintain it

as much, right. So, there's like, I think you mentioned that 70/30 rule,

in the past you know 70% is is spent on maintaining infrastructure, it's flipped

now, right, and we love that. So it allowed us to deploy LogDNA on your

infrastructure, pretty efficiently, and from a customer perspective it's easy to

use and it matches with the best out there, so we love it and we're super

proud to be to call ourselves customers. Yeah, and one of the things I think is

really interesting is that they take logging as a use case. It's not maybe what

people typically think of with kubernetes, you know, it's very

data-intensive, lots of information, high ingest rates on logs coming in, a lot of

processing, not where people really started with kubernetes. Any struggles

they're like..? No. Do you feel that like, you know, I'm doing something on the edge or

you know, did it..? I think we always had the foresight of the volume of days

growing exponentially. So when you think about that, we had to think about

building a platform that's able to ingest that well and store and search

it. The second is because of the complexity, we knew the complexity

because of DevOps, containers, kubernetes, it's gonna be complex and the data is going to be

fragmented. So how do we make it simple, and I think our user experience is

always magical, and it should always feel that way. Right. And the third, was like

one size does not always fit all, so how do you capture the right platform for

developers, versus DevOps, versus security, and that is hard to do, and it was a lot

of foresight we did, I think we just hit our four-year birthday and we

had that first foresight of like it should be a beautiful product and less

is more. So you're up and running with kubernetes with two command lines which

is pretty quick. Right. And you tested it. Yeah. Which is awesome and that's how it

should always feel like, you have other things to focus on your priorities, and us

handling the infrastructure monitoring aspect, observability, should be our core

focus, that's the benefit of IBM customers. Yeah,

so let's talk about our partnership a little bit, so logging it plays an

important role for IBM. You know you are the embedded logging solution within our

cloud. Yes. And you know I think the you and I have done a lot of work together

to try to make that a seamless experience for our users, our joint users.

You know one of the ways that I tried to push our partnership is IBM

being a client of yours as well. Correct. We have this kind of joint relationship

which I think is really powerful, and we use LogDNA tremendously within the

cloud team, and in other parts of IBM to run our own systems. Yes.

Right. And I think that's taught us a lot about you know how to operate at scale,

you've pushed kubernetes service hard which has been good as well. Talk

to me a little bit about kind of the partnership angle and kind of how its

benefited you guys. It's such a win-win scenario, So one, we're customers of IBM

Cloud. Second, we get to embed OEM, our product to

your customer base. Right. And your customers happens in two folds, it's

your customer base that you have but also IBM is their own customer. Right. And it's

such an honor when we see scale and that's the word that resonates

a lot, and the reason why you have scale is customers that hit Watson uses

us, Weather Channel uses us and that's a lot of logs, right,

and it's being able to handle that infra which just talks about so much of like

enterprise ready product. Yeah, right. Not just for IBM, but for IBM's customers as

well and just it doesn't get more confidence there versus the other

options that were out there. Yeah, one of my personal favorite moments is when I

was at the Masters last year and walking through the Operations Center that IBM

runs to run that event and and seeing the team there on the ground who I'd

never met using LogDNA using IBM Cloud and kind of leveraging that is how they

ran that event and all on their own, self service, they just consumed the platform.

That's amazing, that's what we're seeing a lot of, just internal teams using it just

because it's easy if you're up and running within minutes. Yeah. And that's

how it should always be, but also the data that you're capturing and the

value of sending alerts. It's a lot of flexibility that's we think for first

and foremost of our product and it jointly collaborates pretty

well. Yeah, so what's next? What do is a big thing that you think

you guys are focused on next that we can go do together? I think when we talk

about like OpenShift is a great opportunit,y we talked about Cloud

Paks is a great opportunity, and it's nice that the technology, our technology is

aligned, super aligned with with your future path and it comes down to execution. I

think being a lean team we get to focus on what matters most, but most

importantly our core DNA has always been developers, and it's what do developers

want, how do they get up to speed, and how do you just get them hooked so that we

can land and expand over time. Yeah. That's a new movement that we both share together

which is amazing. Yeah, I love the lean team. I remember when we first

met and you know I have this really small focused team and

the amount that, I

think we all lose sight sometimes of you know with cloud, with public cloud, with

technologies like kubernetes, but with all these things we were so used to

talking about we lose sight of like how much people are able to accomplish

with small teams using those platforms. It was awesome, a great communication between

the teams. It was just a vote of confidence both ways because the fact

that IBM was forward thinking to say who do we partner with, and also the fact

that we were able to showcase our technology from scale as one perspective,

flexible deployment, but I also love that we're fully compliant across the board.

Yeah. It's PCI, GDPR, SOC 2, HIPAA, we were ready. Yeah. And then it doesn't

get better alignment then having the right overlap. Especially with IBM's

kind of focused customer base absolutely the industries that we're in like that

was really critical that that layer of the platform which can contain a

lot of sensitive data met those requirements and that was really

key for us. Right. And I think we get to collaborate of what our future, our

UI/UX looks like. Right. I look forward to how do you create predictive graphs, so

instead of teams walking in setting up their own graphs if we know it's kubernetes

here's a five graphs you need to know and where do you want

the alerts being sent. So, we always think about building those magical moments

and this should feel magical when developers use us. Yeah, well look

Chris, I really appreciate the partnership that we've had. I'm excited

for what we're doing. I think LogDNA has really proven

to be a powerful platform for our customers and I'm excited for

us to continue to collaborate together. Well thanks for collaborating and we're just

getting started, that's the best part. Great, and for everybody

else you know go check out LogDNA on logdna.com, or come to IBM Cloud

and check out the offerings that we have there. Thank you very much. Thank you.

#### [Lesson 3 - Security and Compliance Management](https://courses.cognitiveclass.ai/courses/course-v1:IBMDeveloperSkillsNetwork+CC0250EN+2020_T3/courseware/89227024130b43f684d95376901b65c8/2f87e39cdfab4c7e81c2d887eea60c57/)