#_____Section 3_____
Welcome to the first section of this course
in which I will introduce you the Cloud
and the Cloud Computing.
So this section is not going to be hands-on,
it's just some theory and some slight,
but hopefully it will put some context into
why the Cloud is useful and how it works.
So let's go back to the very basics.
How do websites work?
Well we have a server hosted somewhere,
and we, as a web browser,
want to get access to that server to visualize a websites.
What we are going to do as a client is use a network.
A network between ourselves and the server,
and the client will find the network and will use network
to route the packets, the data into the server,
then the server will reply to us,
and we will get the response, and we can view a website.
Obviously that is very simplified,
but that gives you an idea.
Now for the clients to find the server
and the server to find the clients,
you need to have IP addresses.
So a clients have IP addresses
and a server also have an IP address.
And so the idea is that when you use an IP address,
you can send a request to wherever you want
to the server you want,
and the server can know how to find you back.
This is very similar to when you are
writing some letters to your friend.
For example, you would write a letter,
and that would be your data, and you would be the client,
then when you send the letter you put it in your mailbox,
and then the network will be the network of the post office,
then the post office will use network
and the address you put on the letter
to route your letter to the destination,
which is, in this case, the server,
and then if your correspondent wants to reply you back,
they can use the address you put on the back of the envelope
to write you back, and again,
use the same network to get the letter back to you.
So servers are just like the network of your mail.
Hopefully that's a good analogy.
So what is in a server?

Well a server is going to contain a CPU,
and a CPU is a little piece
that will be doing some computations,
it will be very helpful
to do some calculations and find results,
and then, your server also needs RAM, or memory.
This is going to be very, very fast memory,
which will allow us to store information
and retrieve it very quickly.
So when we have a CPU and a memory bar, what do we get?
Well we get a brain.
Think of your brain.
When you are thinking, you are actually making computations,
very complicated ones, but they are computations,
but then you need to retain some information,
and again, we have memories
and these memories are in our brain,
so if we think of the CPU and the RAM together,
they sort of look like a brain.
Now we also need to have
some more long-term storage of data.
Obviously it's still in our brain as humans,
but in computers, we have included
some special storage to store data, for example, files,
and then if we want to store the data
in a more structured way, we're going to use a database,
and a database is going to be data formatted in a way
that we can easily search it and query it.
Finally in the server,
we're also going to have some networking aspect.
So there's going to be the routers, switch, DNS servers,
and don't worry, all these terms,
we'll be seeing them later on in this course.
So in the server, we an aspect of compute, memory, storage,
maybe your server sometimes is a database,
and we have a networking aspect.
All these things are gonna super important going forward
because the cloud is going to be giving these things
for us on demand.
So if we just want to define
a little bit of IT terminology before we get started,
the network is a bunch of cables, routers, and servers
that are going to be connected with each other,
and the router is a specific device
that will forward the data packets
between computer in the networks, and they will know
where to send your packets on the internet,
just like your post delivery service.

Now when we have a packet and it arrives as a destination,
there's a switch, and the switch
will send the packet to the correct clients on your network.
So if we put all these things together, it looks like this.
Our client will send the data to a router,
the router will find it's way all the way to a switch,
and the switch will know to which computer
in your network to send the data to.
So why do I introduce all these things?
Well, let's go back to traditional IT.
When people used to start websites or companies before,
they used to do it in their home or their garage,
and so they would literally go to the store,
buy a server, and they put the server in their home.
You may have seen TV shows,
you may have read some documentation on the internet
that describes on how Google was made.
You know, Google was started in a garage.
Now, as your website grows, you need to add
more and more servers to serve that demand,
and so your home starts to be filled with servers.
So this bad right, but your company is getting bigger,
you're generating some money,
so you're going to move to your own office,
and you decide to allocate a special room
which is going to be called a data center.
In a data center, you're going to have, again, your servers,
and you're going to be able to scale them
by adding and purchasing more and more servers.
Now this worked, and this worked for so many years,
but there are a few problems with this approach.
Number one is that when you have a data center
or your own home, you're going to have to pay your rent,
then you're going to have to add power supply,
cooling, and maintenance
because it does require some electricity
to run your servers, it does require some cooling
because the servers do get hot,
and sometimes they break down,
so you need someone to do the maintenance.
On top of it, if you want to add or replace servers,
it will take a lot of time because you have to order them,
and then you have to hook them up in your center.
Scaling is limited.
If tomorrow you're getting 10 times bigger,
you're going to need 10 times more servers,
but you may not have the time or the space to do so.
You also need to hire a team

that is going to be there all the time,
24/7 to monitor the infrastructure
in case something goes wrong.
And what if there is a disaster,
what if there is an earthquake,
what if there's a power shutdown, or even a fire?
That would be bad, right?
So can we externalize all this?
And the answer is yes, and that will be the cloud.
So I will see you in the next lecture
to discuss a little bit more about the cloud.

So now let's talk about cloud computing.
So what is cloud computing?
The definition is as such,
cloud computing is the on-demand delivery of compute power,
database storage, application, and other IT resources.
The very important keyword here is on-demand,
you get it when you need it.
And then through a cloud service platform,
you're going to get a pay-as-you-go pricing.
That means that you're only going to pay
for what you requested when you requested it
and as you're using it, when you're done using it,
you're not going to pay anymore.
This is a big shift, right?
Then this is cloud computing.
So we can provision exactly the right type
and size of computing resources you need.
Do you need a big server? We have that for you.
Do you want a small one? We have that too.
Do you want 10? Yes.
Do you want two tomorrow? Of course.
The cloud really allows you to adapt
to the type and size you need.
Then you can access all these resources,
not with 24-hour notice, not with two hours notice,
but instantly, you don't need to order things in advance.
When you want a server, and you'll see this in this course,
you'll have it within seconds.
Then the cloud will also give you a really nice interface
so you can easily access your servers, your storage,
databases, and a set of application services.
Something about the cloud, but in specific AWS,
which is Amazon Web Services owns and maintains
the network-connected hardware required
for these application services while you provision and use
what you need via a web application.

So with this interface,
we'll make all these things a reality.
Now, let's go back to our traditional IT.
So we're changing.
We have our office or our garage,
but now instead of building our own data center
we're going to use the cloud, and in the cloud,
which is also a data center, is just not our data center,
we're going to have servers one, two, three, as we need
and as we go and we're just going to pay for exactly
what we're using.
So you have actually been using the cloud without
even knowing it because it is omnipresent,
but not necessarily visible.
So if you use a web client such as Gmail,
well, for example, it's an email cloud service
and you're going to pay only for the emails you stored.
You're not provisioning servers when you use Gmail,
you just use it.
Maybe you've stored some data on the cloud,
maybe through Dropbox, Google Drive, Google Photos,
iCloud, I don't know.
But with Dropbox, for example, it's a cloud store service,
you're going to put your files on Dropbox.
And originally, fun fact, Dropbox was built on AWS.
So we've been using a cloud storage service as well
without knowing it.
And Netflix, it's huge.
It is built entirely on AWS and it provides you
a cloud service, which is to get video on-demand.
Now, obviously these cloud services are very different
from AWS, but we'll learn what it goes behind these services
and how AWS can help you build
these kinds of cloud services.
So let's go one step further.
There are different kinds of clouds out there.
The first one is called a private cloud and the provider is,
could be Rackspace.
This is cloud services used by a single organization,
they're not exposed to the public,
so you get your own private cloud,
your own private data center,
it's just managed by someone else.
You still have complete control over it
and you have more security for a sensitive application,
which may need some specific business needs.
This is out of scope for this course,
but still good to mention.

Now the public cloud is more interesting.
So three famous cloud providers that are public,
are Microsoft Azure, Google Cloud, and Amazon Web Services
that we'll be learning in this course, obviously.
So in this case, the cloud resources own and operated
by a third party cloud service provider
and they're delivered over the Internet
and we'll see the six advantages of using cloud computing.
So in this instance, that means that from AWS,
we'll be able to request what we need when we want it.
And then lastly, which is also important for the exam
is the concept of a hybrid cloud.
So with hybrid, we're actually getting the mix
of private and public.
We're going to keep some servers on premises and we'll
extend some of the capabilities we need into the cloud.
That means that we'll have a hybrid of our own
infrastructure and the AWS cloud.
We'll have control over sensitive assets
in your private infrastructure,
but we'll have the flexibility and the cost effectiveness
of using the public cloud.
Now, five characteristics of cloud computing.
The first one is that it's fully on-demand and self service.
Users, and we'll see this in this course,
we will be able to provision resources
and use them without having anyone from AWS intervene.
Then we'll be having access to a broad network,
the resources will be available over the network,
and it can be accessed in diverse ways
as we'll see in this course.
It'll be multi-tenancy and we'll have resource pooling.
So that means that not just us, but other customers
from AWS can share the same infrastructure
and applications while still having security and privacy.
And then these multiple customers are getting serviced
from the same physical resources.
So here, me, you, and other customers,
we're going to share this entire data center of the cloud.
This gives us rapid elasticity and scalability.
That means that we can automatically and quickly acquire
and dispose resources when we need.
And that means that we can quickly
and easily scale based on demand.
And that is a major advantage of the cloud.
Finally, it's a measured service, so the usage is going
to be measured and we're going to pay exactly
for what we have used.

This is a big shift from on premises.
Now, six advantages.
We're going to trade capital expenses
for operational expenses, so CAPEX or OPEX.
That means that you don't own hardware,
you're going to pay on-demand and that will reduce
your total cost of ownership, your TCO,
and your operational expense.
That means that you don't buy the hardware in advance,
you're just going to rent it from AWS.
Then we're going to benefit from massive economies of scale.
The price is because we are using AWS, not just us,
but other customers and so many people are using it,
then the prices will be reduced by AWS over time because AWS
will be more efficient at running due to its large scale.
We also need to stop guessing capacity.
Before we had to plan and buy servers in advance
and hope that it would meet the capacity,
but now we can actually scale automatically
based on the actual measured usage for our application.
And because everything's on-demand,
we have increased speed and agility.
We can create, operate and do stuff right away,
no blockers for us to be efficient.
And finally, we have a huge cost that we don't need
to have anymore, which is we can stop spending money running
and maintaining data centers.
And this allows a team of say five people
to create a global application in minutes,
thanks to leveraging this AWS global infrastructure
that is going to be worldwide.
Okay. So the problems we've just solved by using the cloud
is that we're more flexible, we're more cost effective,
we are more scalable because we can add resources
as we need to go along,
we're elastic, we can scale out and scale-in when needed,
we also have high availability and fault tolerance
because we don't really on the one data center,
we rely on the fleet of data centers all around the world.
We're more agile, we can rapidly develop,
test and launch software applications,
and although this make the cloud a really no brainer.
So that's it, just for an introduction of how the cloud
is going to be effective.
Now in the next lecture,
we're going to view one step further,
what are the different types of cloud computing.

As we will see in this course,
there are different types of Cloud Computing,
and it is important for us to be able to recognize them.
The first one is called Infrastructure as a Service or IaaS.
This is to provide the building blocks for cloud IT.
With this IaaS, we're going to provide networking,
computers and data storage space in its raw form.
And using these building blocks deals building Legos,
we're going to be given a very high level of flexibility,
and we can easily understand
how we can migrate from traditional
on-premises IT to the cloud.
That is the first service we'll see in this course,
which is going to be easy too.
Then we're going to get Platform as a Service.
In this, we're going to remove the need
for your organization to manage
the underlying infrastructure.
And you can just focus on the deployment
and management of your applications.
And then one step even further,
is Software as a Service or SAS.
This is a completed product
that is going to be run and managed by the service provider.
So if you want to compare all these things,
well, let's take an example.
On-premise is you're going to manage everything.
So your applications, your data,
your runtime, your middleware,
the operating system, virtualization, servers,
storage and networking.
And that's a lot.
With IaaS, Infrastructure as a Service,
we're going to manage the application,
the data, the runtime, the middleware,
and the OS, but all the virtualization,
servers, storage and networking,
are going to be managed by others.
And in our case, AWS.
With Platform as a Service,
we manage even less.
So everything from the runtime
to the networking is managed by AWS.
And the only thing we care about
when we use a Platform as a Service
is our application and our data.
And finally, well if you're using Software as a Service,
everything is going to be managed by AWS.
So how does it translate?
Well, with IaaS, we can use Amazon EC2 on AWS.

We have other services such as Google Cloud,
Azure, Rackspace, Digital Ocean, and Linode,
which will provide us
Cloud Computing Infrastructure as a Service.
Platform as a Service
also exists on AWS with Elastic Beanstalk
and we'll see all these services obviously in this course.
And outside of AWS,
we have Heroku, Google App Engine and Windows Azure.
For Software as a Service,
we'll also have this on AWS,
that represents many services of AWS for example,
recognition when we want to do some Machine Learning,
but we've been using it as well
in the real world with Google Apps such as Gmail,
or Dropbox or zoom for your meetings.
So the clouds have different flavors.
But one thing is common
is that the pricing is very different from what you know.
AWS has three pricing fundamentals
and it will follow the pay-as-you-go pricing model.
So for the compute, and that represents various services,
where you're going to pay
for the exact compute time.
For the storage, we're going to pay
for the exact amount of data stored in the cloud.
And for the networking, we're going to only pay
when the data leaves the cloud.
Any data that goes into the cloud is free.
And this solves the expensive issue of traditional IT.
Because now we only pay exactly what we need
and so we have huge cost savings ahead of us.
So that's it for this lecture.
In the next lecture,
we'll be having a deeper dive on AWS

Now let's look at the history
of the AWS cloud.
So it was launched in 2002
internally at amazon.com.
Because they realized
that the IT departments could be externalized.
So their Amazon infrastructure was one
of their core strength
and they said, "you know what
"maybe we can do IT for someone else,
"for other people."
So they launched their first offering publicly
which was SQS in 2004.
In 2006, they expanded their offering

and they relaunched with the availability of
SQS, S3, and EC2.
Don't worry, we'll see all these services
in this course.
Then they expanded and said, "you know what?
"We don't have to be just in America.
"We could be in Europe."
And then fast forward to today,
we have so many applications that used to run
or are still running on AWS,
such as Dropbox, Netflix, Airbnb, or even the NASA.
Now, let's look at where AWS is today.
If we look at the Gartner magic quadrants,
which sort of ranks the cloud providers,
as we can see AWS is on the top right corner,
which is a leader.
It's able to execute really well.
And it has a really great completeness of vision.
It is followed closely by Microsoft and Google.
But still
in 2019, AWS had $35 billion in annual revenue,
which is huge and accounted for 47%
of the market in 2019.
With Microsoft being second with 22%.
So by learning AWS
you are learning a tool that is widely used.
It is a pioneer and leader
of the AWS Cloud Markets for the ninth consecutive year.
And it has over 1 million active users.
So what can you build on AWS?
Well, pretty much everything.
AWS will enable you to build sophisticated
and scalable applications
and they are applicable to diverse set of industries.
Every company has a use case for the cloud.
So Netflix, McDonald's, 21st century Fox, Activision,
they're all using the cloud.
And use cases can include just
transferring your enterprise IT
or using the cloud as a backup and storage,
or doing some big data analytics.
You can also host a website
or create a backend
for your mobile and your social applications.
Or you could have your entire
gaming servers running on the clouds.
The applications are endless.
Now AWS is global.
And this is where we are going to learn a bit more specifics
about how it works.

So we have AWS regions,
we have availability zones, data centers,
edge locations, and points of presence.
And all of these can be represented on the map right here.
So let's go on this website
to have a quick look at it.
So this is a cool map,
because on this website
we can see how AWS is global.
So if I click on it,
I can, you know,
scroll the world and see what is happening.
So we can see that AWS has multiple regions
and they're in orange and they're all around the world.
For example, Paris, in Spain, in Ohio,
in Sao Paulo, Cape Town, Mumbai, and everywhere else.
So AWS truly is a global service.
On top of it,
each region are going to be connected through the network.
So these are the network reconnecting the regions
and this is a private network of AWS.
And then within each region, for example,
if I really scroll into the Cape Town region,
we can see that we have blue dots.
And each blue dots will be availability zones
that we'll be describing in the next slide.
So as we can see,
what I want to get you out of this,
is that the AWS is truly is global.
And we can leverage the infrastructure of a cloud
provider to make ourselves, our application global.
The first important concept in AWS are regions.
So regions are all around the world
and we saw it on the map from before
the regions have a name,
it could be us-east-1, eu-west-3,
and we can see the mapping of the name of the region
to their code on the console that we'll see in a minute.
Now a region, what is it?
It's truly,
well, it's going to be a cluster of data centers.
So many different data centers.
Look at it (indistinct) for example,
Ohio or Singapore or Sydney or Tokyo.
When we use AWS services,
most services are going to be linked
and scoped to a specific region.
That means that if we use a service
in one region and we try to use it in another region,
it will be like a new time of using the service.

Now, a question that may come up
in the exam is how do you choose an AWS region?
So say you're launching a new application.
Where should you do it?
Should you do it in America, in Europe
in South America, or in Australia?
Well, the answer is, of course it depends.
But let's look at some factors that may impact your choice
of a AWS region.
The first one is compliance.
So sometimes governments want the data to be local
to the country you're deploying the application in.
For example, France,
data in France may have to stay
in France and therefore you should launch your application
in the French region.
Then, there is also a concept of latency.
So if most of your users are going to be in America,
it makes a lot of sense to
deploy your application in America,
close to your users,
because they will have a reduced latency.
If you deploy your application
in Australia and your users are in America,
they will have a lot of lag at using your application.
Then, also not all regions have all services.
Okay?
Some regions do not have services.
And so obviously if you're
leveraging a service with your application,
you need to make sure
that the region you're deploying into
is available and does have that service.
And finally,
pricing.
So, pricing does vary from region to region
and you need to consult the applicant,
the services, pricing, (indistinct)
to see what the differences are between the regions.
But this could be obviously a factor that could
impact your deployment
of an application into a specific region.
Now, availability zones are what actually
are going into the region.
So each region will have many availability zones.
Usually three, the minimum is three, and the max is six.
But, really the usual is three.
So, let's take the Sydney region as an example.
The senior region code is ap-southeast-2.
So, we can have two,

have three availability zones in
Sydney, ap-southeast-2a,
ap-southeast-2b,
and ap-southeast-2c.
Now, each of these availability zones
are going to be one or more,
just create data centers that will have redundant power,
networking, and connectivity.
That means that in
southeast-2a,
I can have two data centers maybe, as well two
in 2b and two in 2c.
But it could be one, it could be three, it could be four.
We don't really know.
AWS doesn't tell us that.
But, what we know
is that these availability zones are
separate from each other
so that they will be isolated from disasters.
So, if something happens to ap-southeast-2a,
we know that it is designed not to cascade
into ap-southeast-2b, or ap-southeast-2c.
So they're really isolated from disasters.
And then these data centers,
these availability zones,
they are connected with high bandwidth,
ultra low latency networking and therefore
altogether being linked together,
it will form a region.
Okay.
Next to, the only thing we need to know
about AWS for the global infrastructure is the points
of presence or edge locations.
We will see them in details
in the global section of this course,
but you should know that AWS has more than 400 points
of presence in 90 cities across 40 countries.
And this will be very helpful when we deliver content
to the end users with the lowest latency possible.
And this is what you see on this map.
Now again, I'm going quickly
over this because we will see this at the,
about the middle of this course.
So, now how about we just play
around and do a tour of the console.
We'll see that AWS has global services such
as IAM, Route 53, CloudFront, and WAF,
but we'll see that also
most AWS services are going to be regions scoped,
such as Amazon EC2, Elastic Beanstalk,

Lambda, and Rekognition.
Finally, to know if a service is available in your region,
there is a region table you should check out right here.

So welcome to the AWS Console Home.
And in this page, you can do a lot of things.
So first of all, let's have a look
at the top right corner of your screen,
right now, this is what's called the regions selector,
and right now I am in Northern Virginia,
US east one.
But it is advised for this course to
choose a region that is geographically close to you.
So because I'm in Europe, I'm actually close to Ireland.
So I can choose EU west one.
But if you are in other regions of the world, for example
if you're in Africa and you're close to Capetown
then choose this.
Of course you don't have to physically be
in that region to use that region, okay?
You can for example
choose whatever region makes sense for you.
So choose whatever is closest
and this will give you the lowest amount of latency.
Next in the console console, you will see a list
of recently visited services
and it should be empty for you.
And I just tried that one
so it's showing one right here for me.
On the bottom you get some information about AWS.
You get the health issues, if need be
and cost and usage info for your accounts
as well as tutorials to build a solution and so on.
So this webpage is actually changing a lot over timeless
changed a lot for the past two years.
And so it may not look exactly the same as you
and in case it looks very, very, very different.
I will rerecord this lecture.
Okay, so once we have this
we need to look at services of AWS.
And for this two options,
the first one is to go on the top left click on services
and you can either look at services
by alphabetical order, as you can see
there are a lot of services on AWS,
or by category for example, for compute
you will have all these services and so on.
But don't worry over time,
we will learn these services
and we don't have to navigate that page.

Another thing I really like is the search bar.
So you can actually type a service for example, route 53
and then it gives you search results.
So it gives you four services that match this query, okay?
And then within these services,
we can also have a look at features
and 13 features match them.
So we can directly jump into the domain names
of the route 53 service and it's a good thing.
We can also look at blogs, knowledge articles
documentations and so on.
So this is quite cool.
Let's go into route 53 now to have a look at this console.
So this one is very special because
on the top right hand side, it says global.
That means that this console
does not require a region selection.
And that is more of the exception than the rule,
but some services in AWS are where
it's called global services and no matter where you are
you're going to get the same view.
But if you switch services and you go, for example
to the EC two service,
this time on the top right hand side,
as you can see it says Ireland
because I chose the Ireland region.
And so based if I run this console in Ireland
or say in another region, for example, in Canada,
well my view is going to be different
in terms of the resources that I will see.
So that's why it's important for you to remain
within the same region for the entire duration
of this tutorial and this course.
The other thing you can look at
is called the AWS global infrastructure
that you can find on Google.
And this gives you a lot
of information around your services.
And one thing that is very important to look at
is the AWS regional services.
And it gives you the services list by region.
And so this is table,
and so for example, if in the course,
I talk about a service and do a hands on,
but it doesn't seem to be in your region,
you can check here and find the availability of services.
So for example, we can check a look,
have a look for Cape town
and see the services that are available in this region.
And if you don't see a service

maybe you need to switch a region
in of course the console,
to have access to it
because not all the services from AWS are in every region.
So that's it for this lecture,
I hope you liked it
and I will see you in the next lecture.

Hi, so I just won't talk to you
about the UI changes that you may see
between my videos and AWS.
So just to explain things to you,
AWS is doing a huge transition of their interface
to a new guidelines for design.
And so in that regards, sometimes the UI will look new
and this is what I call a new type of UI
and it looks like pretty square and so on,
but sometimes you will have the old kind of UIs,
which looks like this.
And so AWS is doing a transition for all their services,
of all their UIs and it's not all at the same time.
It happens over one year or two years
and it's still happening today.
So my goal is obviously to have videos
as updated as possible, but it is literally impossible
to keep up with all the changes
until they're done with them.
So what I'm doing is that I try my best to keep my videos
up to date if they are breaking changes,
but if I believe that the changes are minor enough
that you can find your way around the console,
I will not update the video just yet, but trust me,
I do listen to feedback
and whenever a video is outdated,
I will go ahead and update it.
Okay.
Another thing you can do is that if you really are lost,
you can switch a toggle in most of the services
that have a new experience and switch back to the old UI.
If you do switch back to the old UI like here
as you can see, you will see the old UI sometimes
like you see in my videos, but really I just wanted
to give you a little warning just to explain to you
the course is not outdated.
It is my passion and my duty to make sure
that you pass your exam and learn a lot in this course
and so trust me, I am monitoring every single day
for updates and changes that are breaking my course
and if they do break my course,
I promise that I will update my videos, but please, please,

please forgive me in advance
if the video doesn't look as updated as the UI in AWS,
it is very difficult right now
to keep up with their changes.
So thank you so much and I wish you a very happy learning.

Now, one last touch before get started
with this course.
I will be referring to
the Shared Responsibility Model quite a lot.
And you will see this diagram at the end of this course.
This is what define what is your responsibility
versus AWS when using the cloud
and there is a shared responsibility.
You as a customer, you're responsible
for the security in the cloud.
So whatever you use in the cloud,
however you configure it is your entire responsibility.
That includes security, your data, your operating system,
your network and firewall configuration, et cetera.
And AWS is going to be responsible
for the security of the cloud.
So all the infrastructure, all the hardware,
all the software, all their own internal security,
they are responsible of.
And this is why we have shared responsibility.
In the Certified Cloud Practitioner exam,
you will be asked some questions around finding
what is your responsibility
and what is the responsibility of AWS.
So I just wanted to give you a quick overview of this
and in some sections I will be referring back
to this diagram and this model to tell you
what is your responsibility
and what is the responsibility of AWS.
Finally, when you use AWS,
you are agreeing to their Acceptable Use Policy,
which you can find right here
and I think it's pretty obvious
that you cannot do any illegal,
harmful of offensive use or content,
you cannot do any security violation,
you cannot abuse a network
and you cannot do email or other types of messages abuse.
All of that makes sense but it's good to specify it.
Now , I hope you're excited.
We're gonna get started with this course
and actually get to use the cloud.
So, I will see you in the next section.

#_____Section 4 _____#

Welcome to the first deep dive
on an iterator service.
The first one is called IAM.
So IAM stands for identity and access management.
It is a global service because in IAM,
we are going to create our users and assign them to group.
So we've already used IAM without knowing,
when we created an account, we created a root accounts,
and has been created by default.
This is the root user of our accounts.
And the only things you should use it for is
to set up your account as we'll do it right now.
But then you shouldn't use that account anymore,
or even share it.
What you should be doing instead, is create users.
So you will create users in IAM,
and one user represents one person within your organization.
And the users can be grouped together if it makes sense.
So let's take an example we have an organization
with six people.
You have Alice, Bob, Charles, David, Edward
and Fred so all these people are in your organization.
Now Alice, Bob, and Charles they work together.
They're all developers.
So we're going to create a group called
the group developers who regrouping Alice,
Bob and Charles.
And it turns out that David and Edward also work together.
So we're going to create an operations group.
Now we have two groups within IAM.
Now groups can only contain users, not other groups.
So this is something very important to understand.
Groups only contain users.
Now, some users don't have to belong to a group.
For example, Fred right here is alone,
he does not correspond to any group.
That is not best practice.
But it is something you can do in AWS.
And also, a user can belong to multiple groups.
That means that for example, if you know that Charles
and David worked together,
and they're part of your audit team,
you can create a third group with Charles and David.
And as you can see, now, in this example,
Charles and David are part of two different groups.
So this is the possible configurations for IAM.

So why do we create users and why do we create groups?
Well, because we want to allow them to use our AWS accounts
and to allow them to do so,
we have to give them permissions.
So users or groups can be assigned
what's called a JSON document.
I'll show you right now what it means called a policy,
an IAM policy.
So it looks just like this.
So you don't have to be a programmer.
This is not programming.
This is just describing in, I think plain English,
what a user is allowed to do or what a group
and all the users within that group are allowed to do.
So in this example, we can see that we allow people
to use the EC2 to service and do describe on it,
to use the elastic load balancing service
and to describe on it and to use CloudWatch.
Now we'll see what EC2 elastic load balancing
and CloudWatch mean, but through this JSON document
that looks just like this.
We are allowing our users to use some services in AWS.
So these policies will help us define permissions
of our users.
And so in AWS, you don't allow everyone to do everything
that would be catastrophic,
because a new user could basically launch so many services
and they will cost you a lot of money
or would be valid for security.
So in AWS, you apply a principle called
the least privilege principle.
So you don't give more permissions than a user needs.
Okay, so if a user just needs access to three services,
just create a permission for that user.
So now we have seen an overview IAM.
Let's go in the next lecture
to practice creating users and groups.

So let's go ahead
and practice using the IAM service to create users in AWS.
So in the search bar,
I just type IAM and I go into the IAM console.
So upon arriving on the IAM Dashboard,
we have some security recommendations
that we can for now not care about.
And what I want to draw your attention to
is that on the left hand side, we go to users.

So this is where we're creating to create users for IAM,
but first, let's notice something.
If you go to the top right corner and click on Global,
you can see that the region selection is not active.
That means that IAM as an entire service is a global service
and therefore there is no region to be selected.
When you create a user in IAM,
it will be available everywhere,
but some other consoles we'll see in this course
will be region-specific.
So just something to notice.
Okay, so now we have users, and why do we create users?
Well, we create users,
because right now, we are using what's called the root user.
So if you click on this,
you see there's just the account ID available to you.
So when you have it,
that means you're using the root account
and it's not best practice to use the root account.
So therefore, we want to create users such as admin users
that will allow us to use our accounts more safely.
So for this, let's go ahead and create a user,
and I will provide a username, for example, Stephane.
So of course I want to provide myself access
to the management console,
so I'm going to do this,
and we have the option to use identity center,
which is recommended,
or to create an IAM user.
I will choose the second option because it is more simple,
and from an exam perspective,
this is the one you need to know about.
But don't worry,
this does not affect how your course is going to go.
Okay, so we create an IAM user,
and now we have to set the password.
So if this was a user that was not me,
I would leave it as auto-generated password,
and I would leave this
so that the user must change this password
at the next sign-in,
but because it is me,
I'm just going to enter a custom password
and I'm going to untick this
because I don't need to change my password
at the next login.
So let's click on next.
Next, we have to add permissions to this user,

so we can add it directly or we can get started with groups.
So let's create a group, and we're going to create a group.
The group name is going to be admin
and the policy name is going to be administrator access.
So now that this is done,
we can add the user into the admin group.
So let's click on next,
and we can review everything right now.
So we have the username, the permissions on the group,
and we have tags, and tags are everywhere in AWS.
They're optional, but they allow you to give metadata
to many of your resources.
For example, I could say
that the department of Stephane is engineering.
This is not something I'm going to do everywhere
on the course,
but I want show you once how
you can add tags to resources in AWS.
Okay, so now the user is created successfully.
So now we can email signing instructions or download
CSV files and then we can log in with this user.
But first, let's return to the user list
and have a look at everything.
So here is my user lists, here is me
and we also have groups.
So if I go to the left hand side, user groups,
we have admins.
So let's observe admins.
So admins has one user in it named Stephane.
And if you look at permissions of admins
you see that there is administrator access attached
to the admin group.
Now if I go to my user, Stephane in here,
we can look at permission policies
and see it also has administrative access
but this one has not been attached directly.
It has been attached via the group admin.
So that means that Stephane inherited any permissions
of the group admin it is in.
And this is why we put users in groups.
It is a bit more simple to manage permissions this way.
So now let's go back to our dashboard
and we want to sign in with our user, Stephane.
So first what we can do is look at our AWS accounts
and it has an account ID and a Sign-in URL.
Now you can customize this Sign-in URL very easily
by creating what's called an account alias.
So it could be aws-stephane-v3 and then Create alias

so whatever alias until someone hasn't created it,
so it has to be unique.
For example, v5 is available.
So now using this alias can simplify my signing URL.
Now to sign in using my Stephane accounts
we could use the same browser
or we could create a new browser window in private mode.
And the benefit of doing this is
that we can have two windows side by side using AWS.
So if you don't do this, that's fine, but if you log
in using the Stephane account on the right hand side window
then you will be disconnected on the left hand side,
this is the only difference.
So to use two accounts at the same time,
the route on the left and my account on the right
what I'm doing as a trick is that I'm using a private window
on my web browser, and Chrome has this feature,
Firefox as this feature, Safari as this feature, and so on.
So by pasting the signing URL, as you can see,
I get the sign in and as an IAM user
and to get to this page, we can go back to one.
And as you can see, when you do a sign in on AWS,
you have the root user sign in or the IAM user sign in.
So to get back to this, we go to IAM user.
We enter either the account ID or the account alias
that I can copy in here, and then we are taken to this page.
So the IAM user name is going to be Stephane
and the password is going to be whatever you have set
from before, then you sign in.
So now the cool thing is that if I look
at the top right hand side, IAM logged in using my IAM user.
So it says the account ID and the IAM user.
But if I look on the top right hand side of here
it just says the account id
which shows me it's the root accounts.
So here we are, we have the root accounts logged
in on the left hand side through a normal window
and we have the IAM user logged in on the right hand side
through a private window.
Please make sure not to lose your root account
logins and your admin login.
Otherwise, you will be in deep trouble with your
account and you'll have to contact AWS for support.
And currently I cannot help you with this.
Now from a course perspective,
I recommend you use your IAM user and not your root user
but this is just a normal recommendation.
Sometimes you'll see me using root

sometimes I'm using IAM user.
But when you have to use roots or when you have
to use an IAM user, I will let you know in the course.
Don't worry about that.
Now for the rest of this section,
please keep these two windows open and I will
see you in the next lecture.

Okay, so now let's discuss, IAM policies in depth.
So let's imagine we have a group of developers,
Alice, Bob and Charles, and we,
attach a policy at the group level.
In that case, the policy will get applied
to every single member of the group
so both Alice, Bob, and Charles
they will all get access and inherit this policy.
Now, if you have a second group with operations
with a different policy,
David and Edward will have a different policy
than the group of developers.
If Fred is a user,
it has the possibility not to belong to a group.
And we have the possibility to create what's called
an inline policy which has a policy
that's only attached to a user.
So that user could or could not belong to a group
you can have inline policies for whatever user you want.
And finally, if Charles and David both belong
to the audit team and you attach a policy to the audit team
as well, Charles and David will also inherit
that policy from the audit team.
So in this case, Charles has a policy from developers
and a policy from audit team.
And David has a policy from audit team
and a policy from the operations team.
That should make a lot of sense
when we get into the hands-on.
Now, in terms of the policy structure,
you just need to know at a high level how it works,
as well as how it is named.
So this is something you will see quite a lot in AWS,
so get familiar with this structure
this is adjacent documents.
And so an IAM policy structure, consists of a
version number, so usually it's 2012-10-17,
this is the policy language version.
And ID which is how to identify that policy,
this is optional.
And then more statements,
and statements can be one or multiple ones,

and a statement has some very important parts.
So the Sid is a statement ID, which is an identifier
for the statement, which is optional as well,
so on the right hand side is the number one.
The effect of the policy itself, so it is whether or not
the statement allows or denies access to certain API,
so in the right hand side, this says allow,
but you can see deny as well.
The principle consists of which accounts, user or role
which, to which this policy will be applied to.
So in this example, it's applied to the root accounts
of your AWS accounts.
Action is the list of API calls that will be
either denied or allowed based on the effect.
And the resource is a list of resources,
to which the actions will be applied to.
So in this example, it is a bucket,
but it could be many different things.
And finally in, not represented here
but there's a condition to which when
this statement should be applied or not,
and this is not representative here because it is optional.
So going into the exam, you need to make sure
that you really understand the effect, the principle,
the action and resource, but don't worry,
you will see those along the way in the course
so you should be confident with them
by the end of the course.
That's it for this lecture, I hope you liked it.
And I will see you in the next lecture.

Okay, so now let's discuss, IAM policies in depth.
So let's imagine we have a group of developers,
Alice, Bob and Charles, and we,
attach a policy at the group level.
In that case, the policy will get applied
to every single member of the group
so both Alice, Bob, and Charles
they will all get access and inherit this policy.
Now, if you have a second group with operations
with a different policy,
David and Edward will have a different policy
than the group of developers.
If Fred is a user,
it has the possibility not to belong to a group.
And we have the possibility to create what's called
an inline policy which has a policy
that's only attached to a user.
So that user could or could not belong to a group
you can have inline policies for whatever user you want.

And finally, if Charles and David both belong
to the audit team and you attach a policy to the audit team
as well, Charles and David will also inherit
that policy from the audit team.
So in this case, Charles has a policy from developers
and a policy from audit team.
And David has a policy from audit team
and a policy from the operations team.
That should make a lot of sense
when we get into the hands-on.
Now, in terms of the policy structure,
you just need to know at a high level how it works,
as well as how it is named.
So this is something you will see quite a lot in AWS,
so get familiar with this structure
this is adjacent documents.
And so an IAM policy structure, consists of a
version number, so usually it's 2012-10-17,
this is the policy language version.
And ID which is how to identify that policy,
this is optional.
And then more statements,
and statements can be one or multiple ones,
and a statement has some very important parts.
So the Sid is a statement ID, which is an identifier
for the statement, which is optional as well,
so on the right hand side is the number one.
The effect of the policy itself, so it is whether or not
the statement allows or denies access to certain API,
so in the right hand side, this says allow,
but you can see deny as well.
The principle consists of which accounts, user or role
which, to which this policy will be applied to.
So in this example, it's applied to the root accounts
of your AWS accounts.
Action is the list of API calls that will be
either denied or allowed based on the effect.
And the resource is a list of resources,
to which the actions will be applied to.
So in this example, it is a bucket,
but it could be many different things.
And finally in, not represented here
but there's a condition to which when
this statement should be applied or not,
and this is not representative here because it is optional.
So going into the exam, you need to make sure
that you really understand the effect, the principle,
the action and resource, but don't worry,
you will see those along the way in the course
so you should be confident with them

by the end of the course.
That's it for this lecture, I hope you liked it.
And I will see you in the next lecture.

So now let's have a look
at IAM policies in depth.
So first of all, let's go into users.
And as you can see, the user Stephane
is part of the admin group,
and therefore, has administrator access permissions to AWS.
That means that if I use my user Stephane
to go into the IAM console, so now I'm using my user,
and then I go to the left-hand side and click on users,
as you can see, I can see my user Stephane,
which is right here.
So my user Stephane has permission to do anything
because it's an administrator.
But what I'm going to do is that I'm going to the groups
of admins and then I'm going to remove my user Stephane
from that group.
So by removing the user, which I've done right now,
then Stephane loses its permissions on the right-hand side.
How do we make sure of this?
Well, let's refresh this page.
And as you can see, now I see zero users
and I get an access denied and it said
that I don't have the permission to do iamListUsers.
And so therefore, because I removed my Stephane user
from the admin group, then I've lost permissions to look
at users on the right-hand side.
So let's try to fix this.
So let's go into IAM
and we're gonna go under users, find Stephane in here.
And right now, as you can see,
Stephane has zero permission policies
but let's add permissions.
So we can add permissions directly
or create an inline policy.
So let's add permissions, and this is going to be easier.
And so again, we could add the user back to a group.
That's not what we want.
Or we could attach policies directly to my user.
And so the policy I'm going to attach is going
to be IAMReadOnlyAccess.
So this will allow my user Stephane
to read anything on IAM, which is what we want.
So let's add this permission
and now this policy has been added.
So back in here, let's refresh this page.
And as you can see now, I can finally do my API call again

and look at the Stephane user in my users category.
So I can view users, I can view user groups, such as admin
but can I create a group?
Let's try to create the developer group
and then create this group.
And as you can see, I cannot create it
because I'm not allowed to actually create a group.
I'm only given the read-only access on IAM.
And so therefore, because I have read-only access,
I cannot create groups.
So this shows you that you can only permission users
for what they're supposed to do.
And of course, if I wanted to give access to create groups
on the right-hand side,
I will need to attach a bigger permission sets,
such as the IAM full access.
So next, let's do something.
So next, I'm going to go into the left-hand side
under user groups, and I'm going to create a group.
So this group is going to be called developers.
And then I'm going to add the user Stephane in this group
and I'm going to attach whatever policy I can find,
for example, AlexaForBusiness
but it doesn't really matter.
Just attach the first policy you can
and let's create this group.
Okay, so this has been added.
And finally, let's go into the admin group.
And again, we're going to add users
and re-add Stephane into this group.
So now if we go back to the Stephane user, so let's go
into IAM and look at the users and look at Stephane,
I'm going to shut down this message on right-hand side.
So if we look at Stephane as the user, as we can see,
we have three permission policies attached to my user.
We have the administrator access that has been inherited
from the group admin.
We have this AlexaForBusiness managed policy
that has been attached via the group developers.
And finally, IAMReadOnlyAccess
that has been attached directly.
And so as you can see,
I inherited different permissions based
on how it's been attached.
So now let's look at policies in detail.
So on the left-hand side, let's look at policies.
And first, let's have a look
at this AdministratorAccess policy.
So if we look at it, it's the permission
that gave us administrator access to everything.

And if you look at the permissions defined
in this policy as a summary, as you can see,
this allows all the services in AWS.
And this number can change over time.
It doesn't matter.
The course will be up to date.
So all these services, for example, App Mesh
or Alexa for Business or Amplify, they all have full access.
So how is this permission defined?
Well, if you click on JSON,
this is the JSON form of this policy, and we can see
that here we have allow Action, star and resource, star.
So star in AWS means anything.
So it means we allow any action on any resource.
And of course, allowing any action
on any resource is exactly the same thing
as giving administrator access to someone.
So this is how it's been defined.
If we have a look at another policy, for example,
the IAMReadOnlyAccess that we saw from before.
So if we look at it, we see
that IAM is authorized with Full: List and Limited: Read.
And if I click on it, you can actually have a look
at all the API calls that has been allowed
as part of this policy, which is very handy.
But if we look at how this has been actually defined,
let's click on JSON.
And here we have the JSON document that shows
how this has been defined.
So the effect is allow, and then we list out the API calls
that are being allowed.
So we have this one, this one, and then we have Get*.
So when you have Get*, it says that anything that starts
with Get and then has something after is authorized.
For example, get users or get groups.
Same for list.
So we have a List*.
So list users or list groups.
So by using a star,
we encompass and group many API calls together.
So all this is allowed on Resource*.
And so therefore, that summarizes
what the read-only IAM access policy is made of.
So this is very handy.
You can also create your own policy.
So let's create a policy
and we have a visual editor or a JSON editor.
So if you have JSON, you can just very simply edit this
and create your JSON document with this builder,
which is very handy.

Or you can use the visual editor.
And for example, let's say IAM,
we wanna create stuff for IAM.
And what action do we wanna authorize?
Well, we want to authorize ListUsers.
So we're going to take this and GetUser.
So just two API calls.
And as we can see, we have selected one out of 38 in list
and one out of 32 in read.
And then what do we want to authorize this on?
So on all resources or only specific resources?
So this is a very simple one
but as you can see, this builder is very handy.
And when you click on next,
you can have a look and say MyIAMPermissions.
And then we create this policy.
And if we have a look at the policy we created,
we can have a look at the corresponding JSON
and see that indeed through the visual editor,
we allowed iam:ListUsers and iam:getUser on Resource*.
And then this policy, we can attach to groups
or to users and so on.
So this is how you manage permissions in AWS.
So now to just wrap up this hands-on,
let's go to user groups
and we're going to delete the developers group
because we don't need it.
And then I'm going to go into my Stephane user
and I'm going to just remove this IAMReadOnlyAccess
that had attached directly.
So now Stephane only belongs to the group admin
and it has administrator access.
So of course, if I go back
to my IAM console in here and I just look at users,
as you can see, yes, everything is showing fine.
So it is working correctly.
Okay, so that's it for this lecture.
I hope you liked it and I will see you in the next lecture.

Now that we have created users in groups,
it is time for us to protect these users
in groups from being compromised.
So for this, we have two defense mechanisms.
The first one is to define
what's called a password policy.
Why?
Well, because the stronger the password you use,
the more security for your accounts.
So in AWS you can set up a password policy
with different options.

The first one is you can set a minimum
password length and you can
require specific character types.
For example, you may want
to have an uppercase letter, lowercase letter,
number, non alphanumeric characters,
for example, a question mark and so on.
Then you can allow or not IAM users
to change their own passwords,
or you can require users to change their password
after some time to make your password expired,
for example, to say every 90 days,
users have to change their passwords.
Finally, you can also prevent password reuse
so that users, when they change their passwords,
don't change it to the one
that they already have or change it
to the one they had before.
So this is great.
A password policy really is helpful
against brute force attacks on your account.
But there's a second defense mechanism
that you need to know going into the exam.
And this is the Multi-Factor Authentication or MFA.
It is possible, you already used it
on some websites, but on AWS it's a must
and it's very recommended to use it.
So users have access to your account
and they can possibly do a lot of things,
especially if they're administrators.
They can change configuration,
delete resources and other things.
So you absolutely want to protect
at least your root account and hopefully,
all your IAM users.
So how do we protect them on top of the password?
Well, you use an MFA device.
So what is MFA?
MFA is using the combination of a password
that you know and a security device that you own.
And these two things together
have a much greater security
than just a password.
So for example, let's take Alice.
Alice knows her password,
but she also has an MFA generating token.
And by using these things together
while logging in, she's going to be able
to do a successful login on MFA.
So the benefit of MFA is that even if Alice

has lost her password
because it's stolen or it's hacked,
the account will not be compromised
because the hacker will need to also get a hold
of the physical device of Alice,
that could be her phone, for example,
to do a login.
Obviously that is much less likely.
So what are the MFA devices option in AWS?
And you need to know them going into the exam,
but don't worry, they're quite simple.
The first one is a virtual MFA device.
This is what we'll be using in the hands-on.
And so you can use Google Authenticator,
which is just working on one phone at a time,
or using Authy.
So for Authy, you have support
for multiple tokens on a single device.
So that means that with the virtual MFA device,
you can have your root account,
your IAM user, another account, another IAM user.
It's up to you.
You can have as many users and accounts
as you want on your virtual MFA device,
which make it a very easy solution to use.
Now we have another thing
called a Universal 2nd Factor
or U2F Security Key,
and that is a physical device.
For example, a YubiKey by Yubico
and Yubico is a third party to AWS.
This is not AWS that provides it.
This is a third party.
And we use a physical device
because maybe it's super easy,
you put it on your key fobs
and you're good to go.
So this YubiKey supports multiple root
and IAM users using a single security key.
So you don't need as many keys as users,
otherwise there will be a nightmare.
Then you have other options.
You have a hardware key fob MFA device,
for example, this one provided by Gemalto,
which is also a third party to AWS.
And finally, if you are using the cloud
of the government in the US, the AWS GovCloud,
then you have a special key fob
that looks like this
that is provided by SurePassID,

which is also a third party.
So that's it.
We've seen the theory
on how to protect your account,
but let's go in the next lecture
to implement that.
So I will see you in the next lecture.

So we are going to first,
define a password policy.
For this, click on account settings on the left-hand side.
You will find password policy and you can edit it.
So here, we can use the IAM default password policy,
which composes of these kind of requirements,
or we can customize the password policy
and force a password minimum length.
We can also require uppercase letter, lowercase letter,
a number, a non-alphanumeric character.
We can also turn on password expiration to turn on,
for example, expire after 90 days,
or that a password expiration
requires administrative resets, or we can allow the users
to change their own password
or we can prevent password reuse.
So this password process can be edited directly
from the IAM console, and that's the first part of security.
The second part
is around setting multi-factor authentication
for your root account.
So if you click on the account name
and then click on security credentials, if you are logged in
with the root user,
you will see my security credentials root user.
Now, there is a way for you
to actually protect your root user,
which is the most important account in your AWS account,
and you can protect it by using multi-factor authentication.
Now, just so you know, I'm going to do it
and demonstrate how it works in front of you,
but I've had students who locked themselves out
of their accounts because they lost access
to their multi-factor authentication device.
As such, if you think you are running the risk
of losing your iPhone or whatever, do not do this, okay?
Just keep your phone with you, just watch my video.
It will be good enough if you want to practice along
with me, you can as well.
And you can also delete the MFA device after activating it.
Okay, but let's go ahead and assign an MFA device.
So I will call this one my iPhone

because this is what I have, but you can name it
whatever you want.
Then you can select the type of MFA device.
So it could be an authenticator app,
which is something I'm going to use,
but also it can be a security key or a hardware TOTP token.
So I'm going to use an authenticator app
because it will be virtual.
And now we go into the setup of the app.
So there's a list of compatible applications right here.
You can find here for Android
and for iOS that we know work well with AWS.
And as such, I'm going to use the Twilio
of the Authenticator, which is an app I like.
So what I have to do then is actually launch the app
on my phone and then you click on show QR code.
So when you should a QR code, you need
to scan this QR code directly on your phone.
So for this, you add an account,
you scan the QR code right here,
and once scanned, it will add the account
and start naming it.
So we'll just save this, this looks good.
And then we get access to MFA code.
So there is first, the first MFA code, so 301935.
So this is a code generated by my iPhone in real-time.
And this code is going to change over time.
And the reason why these two codes are asked by AWS
is that it wants to make sure that the MFA device
is set up correctly and that the codes are accurate.
So the second code is 792843.
And, of course, there will be difference for your device.
And once these two codes are entered, you click on add MFA.
And as you can see, we can reach there up
to eight MFA devices currently, and you can scroll down
and see them right here on the list.
So the multi-factor authentication, MFA, one,
it's called my iPhone that's been created right now.
So if you wanted to remove it, you can remove it and so on.
But so how do we use MFA?
Well now, if I log out of AWS
and I log back in, so I'm going
to use my router account and my password.
Now after doing a successful login,
I have the MFA code to enter.
And so I open my app
and enter the code that I see and press submit.
And this way IAM logged in.
And this is perfect because well, we had an extra level
of security on our account.

So that's it for this lecture.
I hope you liked it, and I will see you in the next lecture.

So we have seen how to access AWS
using the Management console,
which is the Web interface that we've done
so far in this course, but there are, actually,
three different options to access AWS.
So the first one is a Management console,
as we've seen, and is protected by your username,
password, maybe multifactor authentication.
Then there is the CLI, Command Line Interface,
and this is something we will set up on our computer,
and this is protected by access keys,
and access keys our credentials we're going to download
in a few seconds that will allow us
to access AWS from our terminal.
Then, finally, there is the SDK,
the AWS Software Development Kit,
which is used whenever you want to call APIs from AWS
from within your application code.
And again, these will be protected
by the exact same access keys.
So how do we generate access keys?
Well we will do this through the Management console,
and users are responsible for their own access keys,
and access keys, from the user perspective,
there are secret, just like a password,
so if you generate your own access keys
do not share them with your colleagues,
because they can generate their own access keys as well.
So really make sure that you treat your access key ID
just like your username,
and your secret access key just like your password,
you do not share them with other people.
So when you go into the Management console,
you get access key as there's a button
to create access keys, and then it gives you the right
to download it in the very second.
And so, for example, here's a fake access key ID
and a fake secret access key,
and these, when loaded into my Command Line Interface,
would allow me to access the AWS API,
and we'll do this in the hands-on in a second.
So again, remember, I want to make sure
that you don't have any security issues
while doing this course or at work,
do not share your access keys, they are private to you.
So if you're new to the Cloud, and programming and so on,
or IT, then you might not know what's a CLI.

So CLI stands for Command Line Interface,
and the AWS CLI is a tool that allows you to interact
with the AWS services using commands
from your command-line shell.
So whenever you see some code where you type a command line,
and then it returns a result, for example,
aws, s3, cp, and so on, this is what we call the CLI.
And we are using the AWS CLI
because we start every command by the word AWS.
Now with this CLI, you get direct access
to the public APIs of your AWS services
which is going to be very helpful in this course.
And, then, using the CLI you can develop scripts
to manage your resources and automate some of your tasks.
The CLI is open-source,
you can find all the source code on GitHub,
and it is an alternative to using
the AWS Management console.
I know that some people, actually,
do not even use the Management console,
they only use the CLI, for example.
So what's the SDK now?
SDK stands for Software Development Kit,
and this is a set of library,
this is going to be language specific,
so you're going to have an SDK
for different programming languages,
and similarly, it will allow you to access and manage
your AWS services and APIs programmatically,
but this time the SDK is not something that you use
within your terminal, it is something that you embed
within your application that you have to code.
So your application will have the AWS SDK
from within them.
It supports many different programming languages,
such as JavaScript, Python, PHP.NET,
Ruby, Java, Go, Node.js, C++,
all of that's our programming languages.
There's also the mobile SDK,
if you're using Android or iOS,
and the IoT, so Internet of Things device SDK
in case you're using some thermal sensors
or bike locks that are connected, all these kinds of things.
So to give you an example of what you can build
with the SDK, well the AWS CLI that we're going to be using
in this course is actually built
on the AWS SDK for Python named Boto.
So that's it for this lecture.
Now in then the next lecture, we're going to practice
setting up the CLI and dealing with access keys,

so I will see you in the next lecture.

Okay so we are going to install
the aws command line on Windows.
So for this we do aws CLI install windows on Google.
And this will give us the list link
and we want to install the aws-cli version 2 on Windows.
Okay this is the latest and so we'll up to dates.
It doesn't change much versus version 1,
it is just some improved performance and capability,
but the API's exactly the same
and there's also an improved installer.
So I'm going to scroll down,
in here and to Install on Windows
where you can just simply
install it using the MSI installer,
so I just click on this link to download the MSI installer.
then I'm going to run the installer
so it should very very simple.
Now the installer is starting,
I click on Next.
I accept the terms of the license,
click on Next,
Install and then we wait for the installer to be done.
Yes, I want to allow whatever this is doing.
Okay so the installer is now complete.
I click on Finish
and now I can go ahead and open a command line,
so I'll do command line,
Command Prompt here we go on Windows,
and to be sure that it's fully installed
I just type aws--version
and press enter,
and if you get a result like this:
aws-cli, a version that starts with a 2,
then Python, Windows,
that means that your aws-cli
is now properly installed on Windows
and you're good to go.
Finally just note that if you want to upgrade your aws-cli
then you just need to re-download that MSI installer
and just re-run the install
and it will be automatically upgraded.
But as soon as you have this output you're good to go
and you can follow me in the next lecture.
So see you in the next lecture.
So let's install the AWS CLI on mac
and for this we're just going to go on Google
and make sure to choose a link
installing the AWS CLI version 2 on macOS.

And then we're just going to follow the process,
so we'll scroll down
and see what they say.
And here is how to install it.
So we can just download a pkg file
and this will be a graphical installer.
So you download the pkg file.
Then you click on continue, continue, continue
and you agree.
Then you say, "Okay, install for all the users
"on this computer."
Click on continue
and then click on install,
and this goes ahead and installs the CLI on mac.
So we wait for everything to be done,
the files are being written.
Okay, the installation is now successful
and we'll move the installer to trash.
And now to fill this out,
you open a terminal on mac.
So you just go ahead and type, for example, "terminal",
and this will give you a terminal app.
Mine is called iterm on mac,
which is a free terminal that you can use.
And then you just type aws -- version
and if everything is going well,
then it should give you back the version
of the AWS executable.
So let's wait for a little bit.
And we get the answer,
AWS CLI 2.0.10,
so that means that everything has been installed correctly.
So that's it for this lecture.
In case of issues,
please have a look at this guide.
It will have the answer for you.
And that's it for me.
I will see you in the next lecture.

Okay, so let's proceed
with installing the AWS CLI on Linux.
So for this I just google it, and I will choose installing
the AWS CLI Version 2 on Linux because
this is the latest one,
and I'm going to scroll down.
And to install the CLI it goes installing.
We just have to run these three commands.
So the first one is to get a Zip file.
So I copy this, go into a terminal,

and then I will paste it.
And here we go.
So this has been pasted, and it is downloading currently
the installer.
The next thing we have to do is to unzip it.
So I copy this command and paste it,
and this will unzip my installer.
Great.
And the last thing is to run the installer as root,
so I'll do sudo and then install.
This should prompt me for my password,
which I enter right now,
and then the installation proceed.
Now it says you can run,
"user/local/bin/AWS minus minus version,"
or very simply, "AWS minus minus version"
if your user local bin is in your path,
and there we go.
The AWS CLI has been installed as you can see.
It says AWS CLI/2.
and then you are getting a different version based on
when you do this.
And then you get Python and Linux and Botocore,
so you are good to go.
When this works we can run any command on the AWS CLI
and you can go ahead, with the rest of the lectures.
If you got ant issues please read this
and it will tell you what's going on and that it,
I will see you in the next lecture.

Let me show you how to create access keys.
So I'm gonna click on my username, Stephane,
and I'm gonna go under Security credentials.
I will scroll down and I will create an access key.
As you can see,
there are some selection you need to do,
and based on the selection I'm doing,
for example, I want access key for the CLI,
AWS is going to have an alternative recommended.
For example, for the CLI,
it's better to use CloudShell
which I will show you in the next lecture,
so don't worry about it.
Or to use the CLI V2 and an authentication
through the IAM Identity Center,
which I will not show you
because it's a bit more complicated.
So based on what you wanna do,
if it's local code application running outside of AWS

or in AWS and so on,
you will have some recommendation in the bottom.
For now, we're going to use the CLI
and we'll use these access keys
and we'll click here to say
"I understand the above recommendation,"
and I still want to create an access key
because it is important for you to understand
how they are, how they work, what they are, and so on.
So let's create this access key.
And now, this is the only time
you'll be able to have access to the access key
and the secret access key.
So I will go back now
to a previous version of that lecture
where you see the old UI,
but don't worry nothing changes from that point on.
The first thing I have to do is to configure my AWS CLI.
So I'm going to type: aws configure.
And then I am greeted with entering my access key id.
Very nice, I can just enter this one, and press Enter.
And then I'm greeted with entering my secret access key,
which I will enter right here as well.
The default region name,
so this is a region that is close to you.
I will choose eu-west-1
because I will be doing all my tutorials eu-west-1,
but you will choose your own region
and you can enter your own region name.
The region name, by the way, you can get directly
from this drop down right here.
It shows you the name of the region
as well as the region code.
So for me, I'm going to use my eu-west-1.
I'll press Enter, and in the default output format,
I'll just press Enter as well.
So now my AWS CLI is configured,
and so we can have a look at how it works.
We can do: aws iam list-users,
and press Enter.
And this will list all the users in my account.
And as we can see,
the user I have right now is called Stephane,
here is the UserId, here is the ARN,
when it was created,
and when the password was last used.
Which is very similar to what I would get
if I were to go into this UI right here.
So the Management Console and the CLI
do provide similar kind of information.

Next, I want to show you what happens
if we remove permissions from our users.
So I'm gonna go to admins
and I'm going to remove the Stephane user
from the group admin.
And so again, if I go back to my user, Stephane,
it doesn't have any permissions.
And I did this, obviously, with my root account,
not the other accounts.
So now if I go into my UI
and obviously refresh this page,
I'm going to get an error saying that, yes,
I do not have the permissions to do this.
But let's try to do the same thing with the CLI.
So we're going to do an "iam list-user" call,
and we get no response because actually it was being denied.
So, the CLI permissions
are obviously going to be the exact same
as the permissions you get from the IAM console.
So, the takeaway from this lecture
is that you can access AWS using the Management Console
or using access key and secret access key
that you can configure, and then use into the CLI.
So hope you liked it,
and I will see you in the next lecture.
And obviously, do not forget
to add your user back into the group,
otherwise that would be horrible.
So I'm gonna go into Groups, admins,
and I'm going to add my Stephane user back into my group,
and now I am an administrator again.
So that's it, I will see you in the next lecture.

Okay.
So I would like to talk to you about an alternative
to using the terminal to issue commands against AWS.
And this is using cloud shell.
So cloud shell is this icon right here on the top
right corner of your screen.
And if you don't see it, just make sure you check
out the clutch shell availability regions
because it's not available everywhere.
And so if you go to the clash of FAC, you can see
that there are some regions that's not available.
So let's have a look right now with the regions.
Here we go. Question three.
Right now that's, I'm recording.
It's only available in one of these regions.
So by the way, I would recommend if you want to follow

along to just use one of these regions, then
so we can use cloud shell, but if you don't use cloud shell
in this hands-on, that is completely fine.
If the terminal was working for you, do not worry.
You're good to go.
Okay. So we have cloud shell in here and within cloud shell
you could take a minute maybe to launch your environment.
You can issue commands.
For example, you can issue the AWS commands.
So as you can see is installed, if I do
either of us management is version, as we can see
I'm on version two.one right now using cloud shell.
So cloud shell is basically a terminal in the cloud of AWS.
That's free to use.
Okay. So the cool thing about cloud shell is
that whenever you are using the CLI, so for example
it was am list users.
This is going to return for you an API call
as if the credentials being used, where the credentials
of the accounts of you using the cloud right now
which is why the API calls are working.
And by default, you can specify any kind
of region you want to do.
The API call using the management is region arguments, but
in cloud shell, the default region is going to
be the region you're currently in
logged in right now in cloud shell.
So this is another thing that's good to know.
Okay. Other things that you should know
about cloud shell is that you have a full repository.
So for example, right now, as we can see
we have zero files within cloud shell.
But if you just do
echo
tests
into demo dot TXT, this is going to create a text
file that contains the word texts tests.
And so it turns out that
if you happen to restart your cloud shell
then this file will stick.
So all the files you are creating
within your cultural environment, for example
this demo that's TXT are going to stay.
And the other cool thing you can do
about cloud shell is that you can configure it.
So you can say what font size you want, smallest
medium and large.
And so on the tech, the theme you want, so light or dark
if he wants safe based or nuts.
So resist like a bigger cloud shell for me right now.

And also you have the possibility to download
and upload files.
So for example, if I want you to get the full path
to my file, so did this demo let's six.
I can just copy it right now.
Action and download foul, and then do demo dot TXT.
And this will go ahead and download the file for me.
And alternatively, you could upload your own files
into your cultural environments.
So I want to show you these handy options
because for me, they are lifesavers.
Okay.
And finally, if you wanted more tabs into this environment
you could have a new tab.
You can split into column example, and there you go.
You have two terminals
into cloud shell connected at the same time.
So really that once show you the power
of cloud shell in this hands-on again, you're doing it.
You know, all the commands that data just wants to show you.
If you're a power user, then you can do these commands
and how they would work with cloud shell.
So the bottom line for this lecture again, is number one
cloud shell is only available in some regions.
So maybe try to choose one
of the regions where cloud shell is available.
If you want to use it, if you don't want to use cloud shell
or cloud shell is not working for you.
This is completely fine.
As long as you use the terminal, the way we configured it
from before, this will work just fine.
And you'll be fine in the course to either use cloud shell
or your terminal to perform the commands
with the CLA against AWS.
Okay. And also remember that I really like the upload
and download feature of cloud shell
choose to upload files and download files from it.
Okay. So that's it for this lecture.
I hope you liked it.
And I will see you in the next lecture.

So we have to talk about
the last component of IAM,
which is called IAM Roles.
So some AWS services that we'll be launching
throughout this course will need to perform actions
on our behalf, on our account, okay?
And for this to do these actions,
they're just like users,
they will need some kind of permissions.

So we need to assign permissions to AWS services
and to do so,
we're going to create what's called an IAM Role.
So these IAM role will be just like a user,
but they are intended to be used not by physical people,
but instead they will be used by AWS services.
So what does that mean?
It's a bit confusing.
So for example,
we are going to create throughout this course,
an EC2 Instance.
An EC2 Instance is just like a virtual server,
and we'll see this in the next section.
But so this EC2 Instance may want to
perform some actions on AWS and to do so,
we need to give permissions to our EC2 Instance.
To do so, we're going to create an IAM Role and together
they're going to make one entity.
And together, once the EC2 Instance is trying
to access some information from AWS,
then it will use the IAM Role.
And if the permission assigned to the IAM Role is correct,
then we're going to get access to the call
we're trying to make.
So some common roles include
what I just showed you, EC2 Instance roles,
but also other things that perform actions against AWS
we'll see in this course.
For example, Lambda Function Roles or CloudFormation.
So I know this is a high level of review.
In the next lecture we'll be creating a role,
but we won't be using it yet until the next section,
but let's go ahead and create a role.
I will see you in the next lecture.

So let's practice using roles.
So on the left hand side, you click on roles,
and you can see that some roles may have already
been created for your accounts.
Could be two, could be more. It doesn't matter.
But what we're going to do is that we're going to
create our own role in here.
So a role is a way to give AWS entities permissions
to do stuff on AWS.
As you can see, you have different kind of roles.
You can create actually five of them right now.
But the one that you need to know about for this hands-on
and for the exam is going to be a role for an AWS service.
So let's choose this one, and then we need to choose
for which service we want this role to apply to.

So as you can see, if you click on it,
you have commonly used services, such as EC2 and Lambda,
or a role for pretty much every service on AWS.
So it's a very common thing to know in AWS,
and that's why we learn about it.
So we are going to create a role for an EC2 instance
when we get to the EC2 section.
And so we choose EC2, and the use case is just EC2.
We disregard any of these.
So click on next, and now that we create a role
for an EC2 instance, we need to attach a policy.
So I'm going to attach the IAM read only access
to allow my EC2 instance to read whatever is in IAM.
Let's click on next.
Next do meet to enter a demo, a role name,
so DemoRoleForEC2 is going to be my role name,
and then we select the trusted entities.
So this is saying, hey, this role can be assumed
by the EC2 service, and this is what defines it
as a role for Amazon EC2.
We are verifying the permissions,
yes, it has IAM read only access,
and we create this role.
So now my role is created.
As you can see, it appears in my role lists.
And we can verify that the permissions
are correct for this role.
Now, we cannot use this role just yet
because we need to get to the EC2 section,
but we will use it when we get to it.
In the meantime, you've seen how to create a role
for Amazon EC2 and how to attach correct permissions to it.
So that's it for this lecture.
I hope you liked it, and I will see you in the next lecture.

We are nearing the end of the section,
but first let's talk about the kind of security tools
we have in IAM.
So we can create an IAM Credentials Report
and this is at your account-level.
This report will contain all your accounts users
and the status of their various credentials.
We'll be actually generating it right now
and having a look at it.
The second security tool we're gonna use in IAM
is called IAM Access Advisor.
This one is at the user-level
and the Access Advisor is going to show
the service permissions granted to a user
and when those services were last accessed.

This will be very helpful because we are talking already
about the principle of least privilege,
and so using this tool, we're able to see which permissions
are not used and reduce the permission a user can get
to be inline with the principle of least privilege.
So I will see you in the next lecture
to show you how to use the security tools.

So let's generate a credentials report.
For this, on the left hand side,
I just click on Credential report
and then Download credential report
which is going to create a CSV file.
Now this CSV, because I'm using a training account,
is not fascinating,
but as we can see we have two rows in it.
We have my root account and my account name, stephane.
We can see when the user was created,
if the password was enabled,
when the password was last used and last changed,
when is the next rotation to be expected
if we do enable password rotation?
Is MFA active?
So we can see it's active from my root accounts
but it is not active for my Stefane accounts.
Then access keys, are they generated or not?
Yes, they're created for my Stefane account,
but not for my root account,
and when they were last rotated, last used and so on,
you can get more information about other access keys
and certificates and so on.
So this report is extremely helpful
if you want to look at some users
that haven't been changing the password,
or using it, or their account,
it could be giving you a great way to find which users
that deserve your attention from a security standpoint.
Now let's have a look at IAM Access Advisor.
For this, I'm gonna go to my user, stephane,
and then on the right hand side, I click on Access Advisor.
And Access Advisor is going to show me
which services were accessed by my user and when.
So as you can see, organizations, health,
identity and access managers with IAM Service,
EC2, Resource Explorer, were all accessed by my user.
So I use my user to access these things
by clicking in the UI, but some services were not accessed,
for example, Alexa for Business
or AWS App2Container and so on.
So using Access and Advisor, you can actually have a look

at whether or not the user has the correct permissions.
And it turns out that maybe based on this access,
have 37 pages of this,
maybe the user needs access only to a few services
but not all of them.
And this UI allows you to drill down.
On top of it, if a user accesses a specific service
for example Amazon EC2, we are told
that this is the administrator access
that granted access to this service.
So to summarize, Access Advisor becomes very helpful
when you need to do granular user access permissions on AWS.
So that's it for this lecture.
I hope you liked it and I will see you in the next lecture.

So here are some general guidelines
on IAM and best practices, 'cause I don't want you
if you go to use AWS to make some mistakes.
So do not use a root account
except when you set up your AWS account.
So by now you should have two accounts, a root account
and your own personal accounts.
And remember, one AWS user is equal to one physical user.
So if a friend of yours wants to use AWS,
do not give them your credentials,
instead, create another user for them.
You can assign user to groups
and assign permission to groups to make sure
that security is managed at the group level
and should create a strong password policy.
Also, if you can use and enforce the use
of multi-factor authentication or MFA to really guarantee
that your account is going to be safe or safer from hackers.
Then you should create and use roles whenever
you're giving permissions to AWS services,
and that includes easy two instances
which are virtual servers.
If you were to use AWS programmatically or using the CLI,
so the CLI or some SDK, you must generate access keys,
and these access keys are just like passwords,
they're very secret.
So just keep them for yourself.
To edit the permissions of your account,
you can use the IAM credentials reports
or the IAM Access advisor feature.
Finally, never, ever, ever share your IAM users
and access keys, I am very serious about it.
So that's it,
we are nearing the end of this section.
You know everything about IAM.

I will see you in the next lecture.

Throughout the CCP exam, you are going
to get a lot of questions on something called
Shared Responsibility Model.
And this is to ensure that you know
what AWS is responsible for,
and what you are responsible for.
Now, I want to include some bits of information
within some sections to give you an idea of how
the shared responsibility model works for AWs.
So AWS is responsible for everything
that they do, for example, their infrastructure
and their global network security,
the configuration and vulnerabilty analysis
of the services they offer,
and any sign of compliance
that they are responsible for.
But regarding IAM, you are responsible for a lot
of things that AWS will not do for you.
You are responsible for creating your own users,
your groups, your roles, your policies,
the management of these policies,
and the monitoring of that.
You are responsible for enabling MFA
on all accounts and enforcing this, not AWS.
You are also responsible to make sure
that the keys are rotated often.
You need to make sure that you use the IAM tools
to apply the appropriate permissions,
and again, you are responsible
for analyzing the access patterns
and review the permissions in your accounts,
not AWS.
So this is a very simple example,
but an obvious one. But AWS is responsible
for all the infrastructure,
and you are responsible for how you use
that infrastructure.
This is just one of these lectures
on shared responsibility.
I hope you liked it,
and I will see you in the next lecture.

So here is a summary for IAM.
We've seen IAM users and they should be mapped
to an actual physical user within your company.
And this user will have a password for the AWS console.
Now we can group these users into groups
and therefore users only.

We can attach policies
or share JSON documents that outline the permission
for users or groups.
And we can also create roles
and these roles will be identities, but this time
for maybe EC2 instances or other AWS services.
We assume that for security
we can enable multi-factor authentication so MFA
and also set a password policy for our users.
We can use the CLI to manage your services
using the command line
or the SDK to manage your AWS services,
using a programming language.
Finally, we can create access keys to access AWS
using the CLI or the SDK.
And finally, we can audit our IAM usage
by creating an IAM credentials report
and also using the IAM access advisor service.
So that's it for this lecture.
I hope you liked it.
And I will see you in the next lecture.

#_____Section 5
# EC2 E;astic Compute cloud

So, we are going to make sure
to set up a budget and an alarm for that budget
for this course in order for us
not to spend any money or too much money.
So, therefore we need to go into the billing console.
So, you can click on the top right of your screen
and then click on Billing and Cost Management.
So, as you can see here, I get a lot of access denied.
This is because I'm logged in as an IAM user, Stephane,
from my accounts.
And I had administrative access,
but even though I have administrative access,
I cannot access my billing data.
So to fix this, you can go into your root account.
So here, I'm in my root account as you can see.
It just says the name of my account.
It doesn't say IAM user.
And then, you click on it and you go to Accounts.
So from Accounts, you're going to scroll down.
And then scrolling down, you will find the IAM user
and role access to billing information.
As you can see right now it is deactivated.
So, we need to just activate IAM access

and this will allow IAM users to access
the billing information if they are administrators.
So back into my billing console now,
and I refresh this page.
And I refresh again and it can take a bit of time.
As you can see now, I see my data.
So, except this for the forecast
because there is a data unavailable exception,
so insufficient amount of historical data.
Except for this, we can see all my cost information in here.
So, now let me show you what a billing page looks like
for an account that I'm actually using and have some costs.
So as we can see, we get some information
around the month-to-date's cost.
We get the total forecasted cost
for the current month and so on,
and last month's total cost.
So, from this we can get a few information
such as the cost breakdown by month.
So, this is when you start seeing some cost.
And then we can have a look at bills.
So if you look at bills,
say you have any cost for this tutorial.
For this course, let's go into December, 2023.
So you will find at the bottom of it, charges by service.
And so you'll see the number of active services.
Right now I have 28.
And for example,
if I look at the Elastic Compute Cloud, so EC2,
I see I have $43 of cost in EU Ireland.
And it turns out that here is the breakdown of my cost.
So, there is some Amazon Elastic Compute NatGateway,
which is costing me $35.
And there is some EBS cost,
there is some Elastic IP cost, and so on.
And so you can get a lot of information
out of just this bill.
So, in case you see any kind of cost for your accounts,
remember to go into bills,
go to the month you're interested into,
and then scroll down to get charged by service
where you're going to get a lot of information
around how every service is being used,
and how you are billed for service,
which will allow you to break down your bill very easily.
Next, you can go into free tier on the left hand side.
So, AWs does have a free tier.
And you're going to be able to see the current usage

and the forecasted usage,
as well as again, what the free tier is.
And then you will see whether or not
you're going to pass the free tier usage.
So if you do pass it as a forecast, it goes into the red,
then you are going to be billed,
so make sure you turn off anything that is turned on
and potentially costing you money.
So, this is a very, very helpful dashboard.
Okay, so now let's go ahead and set up a budget.
So, on the left hand side you click on Budgets.
And here you can create a budget that will alert you
whenever you reach your thresholds.
So, let's create a budget.
And we're going to use a template simplified.
And the first one is going to be a zero spend budget.
So as soon as we reach 1 cent, we're going to get an alert.
So this is very helpful.
So, the budget name is My Zero Spend Budget.
Here you add your email.
So, I put here stephane@example.com
and then create the budget.
So whenever I will spend 1 cent,
I will have this budget send me an email.
You can also use another template for a monthly cost budget.
And here we're saying, "Okay, we want to have
a monthly cost budget of, for example, $10."
And saying, "Hey, I want to spend
no more than $10 per month on this course."
And then, the email recipients
are going to be again, stephane@example.com.
And by the way, if you follow this course closely,
you should not spend any dollars as you,
when things can cost you money.
But if you're careful, you should not spend any money.
Regardless, it's still good to set up a budget
just to make sure in case you do mistakes
that you don't have a big bill coming to your way.
So now for this $10 budget,
I'm actually going to reach an email
when my actual spend reaches 85% and my,
when my actual spend is going to reach 100%.
And if my forecasted spend is expected to also reach 100%.
So, very helpful.
I can have three emails I can receive from this.
And I'm going to create this budget.
So as you can see,
while my zero spend budget has already been exceeded,

because while I've spent some money this month,
so I'm getting an email right away at this address.
So, with these budgets and access to the free tier
on the left hand side to explore,
as well as accessing your bills breakdown
on the left hand side,
you should be able to debug any kind of costing issue
and billing issue you have on this course.
And this is a skill that is going to be necessary
for you when using AWS.
All right, so that's it for this lecture.
I hope you liked it.
And I will see you in the next lecture.

Been on EC2
in which we will create our first website on AWS.
So what is Amazon EC2?
Well, EC2 is one of the most popular of AWS' offering.
It is definitely used everywhere.
And what is it?
Well, it stands for Elastic Compute Cloud
and this is the way to do Infrastructure as a Service
on AWS.
So EC2 is not just one service.
It's composed of many things at a high level.
So you can rent virtual machines on EC2,
they're called EC2 instances.
You can store data on virtual drives or EBS volumes.
You can distribute load across machines,
Elastic Load Balancer.
You can scale services using an auto-scaling group or ASG,
and all these things, do not worry,
we will see in depth during this course.
Knowing how to use EC2 in AWS is fundamental
to understand how the cloud works.
Because as I said from before,
the cloud is to be able to rent these compute
whenever you need, on demand, and EC2 is just that.
So, EC2, what can we choose for our instances
so there're virtual server that were rent from AWS?
So what Operating System can we choose
for our EC2 instances?
Three options:
Linux, and it's going to be the most popular,
Windows or even Mac OS.
How much compute power and cores you want
on this virtual machine?
So how much CPU?
Then you need to choose how much random access memory

or RAM you want,
and how much storage space.
So for example,
do you want storage that is going to be attached
through the network
and we'll see about it with EBS or EFS,
or do you want it to be hardware attached?
In this case, it will be an EC2 instance store.
And we have a whole section on storage,
so don't worry about it.
And then finally,
the type of network you want to attach to your EC2 instance.
So, do you want a network card that is going to be fast?
What kind of public IP do you want?
And finally, we need to handle the firewall rules
of our EC2 instance, and that is the security group.
And I live, finally, finally,
there's the Bootstrap script to configure the instance
at first launch, which is called the EC2 User Data.
So we have lots and lots of options
and as you'll see in the hands-on,
even more options at other certification levels
that you need to know in EC2 instances,
but at a core of it what you need to remember is that
you can choose pretty much how you want your visual machine
to be and you can rent it from AWS.
And that is the power of the cloud.
You can do this by just in the blink of the eye, really.
So it is possible to bootstrap our instances
using the EC2 User Data script.
So what does bootstrapping mean?
Well, bootstrapping means launching commands when
the machine starts.
So, that script is only run once and when it first starts,
and then will never be run again.
So the EC2 User Data has a very specific purpose.
It is to automate boot tasks, hence the name bootstrapping.
So what tasks do you want to automate?
Usually, when you boot your instance
while you want to install updates, install software,
download common files from the Internet,
or anything you can think of really,
anything you can think of.
So it could be whatever you want,
but just know that
the more you add into your User Data script,
the more your instant has to do at boot time.
Simple, right?
By the way,
the EC2 User Data scripts runs with a root user.

So any command you have will have the pseudo rights.
Okay?
What type of instances do we get for EC2?
And this is an example.
I have hundreds and hundreds of EC2 instance types,
but, here are five for you.
So the first one is a t2 micro, very very simple.
It has one vCPU, one gigabyte of memory.
The storage is only for EBS,
and it has a low to moderate network performance.
But as soon as you increase the instance type
like for example if you stay in the same family,
so we stay in the t2 family but we go to t2.xlarge,
now we have access to four vCPU
16 megabytes of RAM,
gigabytes of RAM, sorry,
and network performance of moderate.
If we go to complete different new levels,
so c5d.4xlarge, which is a very complicated name,
you get 16 vCPU, so 16 cores,
you get 32 gigabytes of memory, so a lot more,
you get some storage that is attached to your EC2 instance,
this is where it says 400 NVMe SSD.
Now the network is going to get really good
up to 10 gigabytes,
as well as the bandwidth to talk to network storage.
And so, as you can see,
if you go to r5.16xlarge or m5.8xlarge,
again you have different characteristics.
So, the idea with this is that
you choose the kind of instance that fits best
your application,
and you can use that on the cloud on demand.
Okay?
Now, for this instance, for our course,
t2 micro is going to be part of the AWS free tier.
You can get up to 750 hours per month of t2 micro
which represents basically running that instance
continuously for a month.
And so this is what we'll be using in the hands-on
that comes in the next lecture.
So this was a short introduction to EC2.
Don't worry, it's going to get very very practical
very soon.
I will see you in the next lecture.

Welcome.
So in this lecture we are going
to launch our first EC2 instance running Amazon Linux.
So for this we'll be launching our first EC2 instance,

which is well a visual server
and we'll use the console for this.
We'll get a high level approach
to all the various parameters you have
when launching an EC2 instance,
and you'll see there are many,
but we'll learn the most important ones.
And then we will launch a web server directly
on the EC2 instance using a piece of code we will pass
to the EC2 instance that is called the user data.
Finally, we'll learn how to start, stop
and terminate our instance.
So let's get started and launch our first EC2 instance.
For this, I'm gonna go
into the EC2 Console, then I will click on Instances
and then click on Launch Instances.
So in there I'm able to launch my first EC2 instance
and to do so I need to add a name and tags.
So the name is going to be My First Instance
and that is an app that is the name tag.
And if you wanted to add additional tags
to tag your instance differently,
then you could click there, but you don't need
to click on this using just name
as My First Instance is good enough.
Next, you need to choose a base image for your EC2 instance.
This is the operating system of your instance.
As you can see, there's a full catalog that you can search
from, but we're going to use the ones
from the quick start that are very, very helpful.
And the one we'll be using is the Amazon Linux,
which is provided by AWS.
So in it I will choose the Amazon Linux 2 AMI.
And as you can see that one is free tier eligible.
So we'll just leave it as is.
So this gives me Amazon Linux 2
and the architecture I will choose is 64-bits x86.
So everything left pretty much as a default
and we'll see in this section and more
and the other ones, you can create your own AMIs
and you can found them in here, okay?
But currently we're just going to use the ones provided
by AWS as quick starts.
Next, we need to choose an instance type.
And so instance types are going to differ based
on the number of CPUs they have, the amount
of memory they have and how much they cost.
As you can see right now I have a t2.micro selected.
This one is free tier eligible, so it will be free
to launch one of them during an entire month

if we leave it running.
So this is what we'll be using.
But in here you could scroll down
and look at other types of instances.
For example, t1.micro is also free tier eligible,
but that's older generation.
And as you can see, you have a bunch
of instances right here available to you.
Some of them are going
to be free tier eligible, some of them will not.
And by default, the one
that's gonna be free tier eligible is a t2.micro.
So we'll be using that one a lot.
If you wanted
to compare the instance types, you would just click on
that link and it shows you all the type of instances in here
as well as how much memory they have and so on.
So right now we'll be using a t2.micro.
Okay next, a key pair to log into your instance.
So this is necessary if we use the SSH utility
to access our instance and we will be using the SSH utility
in this course.
Therefore, it is required for us to create a key pair.
So as we can see right now there is no key pair
and we could proceed without a key pair,
but for now we won't do this.
So let's go ahead and create a new key pair
and the name is going to be EC2 Tutorial.
Then you need to choose a key pair type.
So we'll be using the RSA encrypted, okay, this is good.
And then the key pair format.
So if you have Mac or Linux
or Windows 10, then you can use the .pem format.
If you have Windows less than version 10,
for example Windows 7
or Windows 8, then you can do our shortcut
and directly use a ppk, which is going to be used for PuTTY
and PuTTY is how you do SSH on windows 7
and Windows 8.
So remember anything else but Windows 7
and Windows 8, choose .ppm else use ppk.
Okay, I should be clear enough, I'm going
to create this key pair and it is downloaded
for me directly.
So now it is selected automatically here.
Next we have to go into network settings.
So for now, I will not touch anything.
My instance is going to get a public IP
and then we need to connect to our instance.
And so for this, there's going

to be a security group attached
to our instance, which is going to control the traffic from
and to our instance, and therefore we can add rules.
And the first security group created will be called
launch-wizard-1.
So created by the console directly
and we can define multiple rules.
So the first rule we want to have is
to allow SSH traffic from anywhere.
So we leave it as this and this will create a rule
in our security group to allow SSH traffic,
but we also want to allow HTTP traffic from the internet.
So we'll tick that box and this is
because we're going to launch a web server
on our EC2 instance, so we need it as well.
As we're now going to use HTTPS
for now, we don't need to tick the second box.
Let's configure the storage
so then we can configure the storage.
And as we can see we have eight gigabits gp2 root volume
that will leave it as is, okay?
Because in the free tier we can get up to 30 gigabytes
of EBS general purpose SSD storage.
So this is good. And we only have one volume necessary.
If you go into advanced, you could configure them
and see a little bit more information, okay?
And the one important thing
to note in here is the delete on termination.
By default, it is unable to yes, I just did advanced
to show you that one detail, okay?
That means that once we terminate our EC2 instance, then
that volume is also going to be deleted.
Okay, so we leave everything as is
and we'll be get back into the simple mode.
Okay?
Next, for advanced details,
this is where it gets interesting.
So I will skip spot, I will skip IAM instance profile.
Don't worry, I will go
over them once we need to explore them.
I will skip all of that. So let's scroll down.
Let's scroll down, let's scroll down all the way
to the bottom.
And at the bottom there is user data.
User data is when we pass a script.
So some comments to our EC2 instance
to execute on the first launch of our EC2 instance
and only the first launch.
And therefore on the first launch, we want
to be able to pass these command right here.

So for this, you go into your code, you go
to the EC2 fundamentals,
and then the EC2 user data set file, you copy entirely this,
so all of it.
And then you paste it here. So you paste everything.
And that means that this script is going
to be executed when the instance is first started
and only once, okay?
In the whole lifecycle of the instance.
And what it's going to do is that it's going
to update a few things, then install the HTTPD web server
on the machine and then write a file, an HTML file.
That will be our web server.
And so you don't need to know code or know these commands.
Okay?
This is provided to you to illustrate a few things
on this lecture.
So finally for summary, we want to start one instance.
This is great and we can review everything we have here.
It all looks good. We are very happy.
And as you can see in the free tier, we get a first year
of 750 hours of 2.micro, which is running it for one month.
So that's every month.
And if you don't have a t.2micro
in your region, then it's going to be a t3.micro, okay?
And then also we get 30 gigabytes of EBS storage and so on.
So let's launch this instance and the instance is going
to be launched.
Let's go to View all Instances, Refresh.
And now my instance is in pending state.
So it's gonna take about 10, 15 seconds
for the instance to come up.
And this is the whole power of the cloud.
Thanks to the cloud, I am able to create an instance or 100
of them very quickly in less than 10 seconds
without me owning any single server.
So that is extremely powerful
and we just scratched the surface of the power of the cloud,
obviously because the course is just getting started,
but you can get a feeling of the advances
and the speed we can have on the cloud thanks to this.
So as you can see now my instance is running
and right now I wanna show you a few things, okay?
The first one is
that the instance name is My First Instance,
and there's an instance ID which is just
a unique identifier for my instance.
There is a public IPv4 address.
This is what we're going to use to access our EC2 instance.
Or there is a private IPv4 address, which is how

to access that instance internally
on the AWS network, which is private.
The instance site is running and we get some information
around host name, private DNS, which instance that we have.
So t2.micro as well as if you scroll
down the AMI we're using, which is Amazon Linux 2,
and the key pair we're using, which is EC2 tutorial, okay?
So you can have a look at a few details in here.
You have more information.
For example, on security, we get some information
on the security group, which was created called
launch-wizard-1 with these inbound rules.
So port 22 accessible from everywhere
and port 80 accessible from everywhere.
So you should have something similar, okay?
If you don't start over because you probably missed a step.
And the rule allowing all communication
outwards, which allows the instance to access the internet.
For storage we saw that, yes, we created one volume
of eight gigabytes, so we're good to go.
So now let's have a look
at the web server running on my instance.
And for this you go on public IPv4 address, you copy this
or you click on Open Address.
And as you can see, it doesn't work.
Or if you click on it, copy
and then paste it, you press Enter, it's going to work.
So it depends on the web browsers you have and so on. Okay?
But the reason it doesn't work here is that
in the URL you need to make sure
that you're using the HTTP protocol,
so http:// and then the IP.
Because if you use HTTPS, this is not going to work.
It's going to give you an infinite loading screen
which was happening right here.
So please make sure to use http:// and then the IP address.
And you're going to get this screen.
And in programming, when you do something
for the first time, you usually say Hello World.
So this web server is saying,
Hello World from and this IP right here,
which is not the public IP, this IP right here,
172-31-33-135 actually corresponds
to the private IPv4 address.
So this is something that I program myself.
So we use the public IP address to access it,
but we have the private IP address in here
and we have the hello world.
And if you go too fast, you're going to get no messages.
So if you go too fast, just wait five minutes, get back

to it, refresh this page and you'll see it.
Okay, so cool, we have a web server running, this is great.
Now let's explore a few options.
So we have an EC2 instance and it's running,
but if we don't need it, we can go to instance states
and then click on Stop Instance.
And in the cloud you can start
and stop instances just as you wish.
And why would you stop an instance?
Well, the longer you leave it running, the more you're going
to pay of course.
But if you decide
to stop an instance, then AWS will not bill you for it.
The instance state is kept
because you have a volume attached to it,
but at least you're not paying for it.
So we can see right now, well the instance is
in a stopping state, and if we try
to refresh this page, it's going to,
of course it's not going to work
because well you don't have the server running anymore.
So you can see it gets
to some like infinite loading experience.
Okay?
So my instance is now stopped and if I wanted
to actually I could get rid of it.
And in the cloud it's very common
to start instances and then get rid of them.
We're very quickly just to try it out
because this is the cloud and we can do whatever we want.
So we can do in instance state and then terminate instance.
If we do so, we're going to get a warning message
and don't click on Terminate
because I want to keep this instance with me, okay?
But this is how we would get rid of it.
So I cancel this, but what I'm going to do now is I'm going
to start my instance again.
So I go to Instance State and then Start Instance.
And now as you can see, the state is pending,
so it is getting started and I just wait for it
to be started in the green state
and I will show you something very interesting.
Okay, so my instance is now running, and if I go here
and stop the refresh and try again to refresh,
as you can see, it still goes into an infinite loop.
Well, you may say while the server is running, Stefan.
So why is it not displaying the message now?
It is displaying here,
but like from the old one, of course.
So here the IP start with 54, right?

But here, if you click on here, now the public IP start
with 3.250.
So the public IP actually has changed.
So if you stop an instance and then you start it later
on, then AWS will maybe change its public IPv4.
So therefore you need
to copy the new IPv4, make sure to use HTP.
And voila, we have access back to our EC2 instance.
But one thing that has not changed is the private IPv4,
the private IP will always stay the same,
but the public IPv4 may change.
Okay? So well, so that's it for this hands-on.
We have seen quite a lot of things.
We've launched our first EC2 instance,
which is very exciting.
Our first web server in the cloud.
We've had to look at some of the power of the cloud.
You're just using some API calls
to stop an instance, start instance, and so on.
So I hope you liked it and I will see you
in the next lecture.

Okay, so now let's talk
about EC2 Instance Types.
So there are different types of EC2 instances
that you can use for different use cases
and they have different types of optimization.
And let's go check out this link
and we'll see we have for now,
seven different types of EC2 instances.
So this website on the AWS website
is what we're interested into.
And as we can see, we have different types of instances.
We have general purpose, compute, optimized,
memory optimize and so on.
And so for each type of instance
we have different families.
And so as we can see this website is going to
be the reference for us to look at Institute Instance types
and know their costs and other specificity.
What I'm going to do though, is just walk you
through a high-level overview of how they work in AWS.
AWS will have the following naming convention.
For example, we'll be talking
about an M five dot two X large type of instance.
What does that mean?
Well, M is going to be called the Instance Class.
Okay. And this is going to be, for example, in this case
a general purpose type of instance,
five is generation of the instance.

So as AWS improves the hardware over time
if we release a new generation of hardware, and so
after M five, if they improve the M type of instance class
then we'll go to M six
and then finally the two X large represented the size
within the instance class.
So, it starts as small and then large
and then two X large four X large and so on.
So it represents the size of the instance,
and the more the size, the more the memory
the more the CPU is going to have on your instance.
So from an exam perspective, what do you need to know?
Well, we'll talk about a few different types
of instance types.
So you have a general purpose
and these are great for diversity of workloads
such as web servers or code repositories.
They will have a good balance
between compute, memory, networking.
And so in this course
we'll be using general purpose instances.
We'll be using the T2 micro
which is the free tier, general purpose type of instance.
On the website that I just showed you
you will see all the different types
of instance that our general purpose
and this is going to evolve over time,
So I won't update this slide.
But you can always refer back
to the AWS website to check what the instances are
in the general purpose type of family.
Then we have compute optimized,
and these are instances are great,
and optimized for compute intensive tasks.
So what requires a high level of processor?
Well, if, for example, it could be
if you're batch processing some data
if you're doing media transcoding
if you need a high-performance web servers
if you're doing high performance, computing is called HPC.
If you're doing machine learning
or if you have a dedicated gaming server.
So all these things are tasks that require a very good CPU
very good compute side.
And so Ec2,
instances do have this kind of particularity
and for now all the computer optimized instances
in EC2, are of the C names.
So C5, C6, and so on.
Next, we have some EC2 instances that are memory optimized

and there are going to be
have a really fast performance for the type
of workloads that will process large datasets in memory.
So memory means RAM.
And so the use cases are this is going to
be high performance for relational or
non-relational databases are mostly going to be
in memory databases,
distributed web-scale cache store.
So for our elastic cache, for example
in memory databases that are optimized
for business intelligence or BI.
And applications performing real-time processing
of big unstructured data.
So in terms of the names for the memory optimized instances
there's going to be the R series because R stands for RAM
but there's also going to be X one high memory and Z one,
but again, you don't have to remember the name
of the instances, but good to know at a high level.
Okay. And finally we'll have storage optimized instances.
They're great when you are accessing a lot
of data sets on the local storage.
And so the use cases for storage optimized instances
are going to be high-frequency online
transactional processing, so OLTP systems.
Relational and NoSQL databases.
And we'll see those in details.
When we get to the database sections.
Cache for in-memory databases, for example,
Reddit's data warehousing application
distributed file systems
and the search optimized instances in AWS
will start with an I, a G, or H one.
Okay. But again, don't have to remember this.
I'm just giving you examples.
So what does it mean?
Let's compare a few instance types.
So for example, for t2.micro
we have one VCPU and one memory, one gigabytes of memory.
And if you look for example, at r5.16xlarge
we have 16 VCPU and 512 gigabytes of memory.
So we can see there's a lot of more emphasis on the memory.
If we compare it to example, to a c5d.4Xlarge
we can see we have 16 VCPU and 32 gigabytes of memory.
So less memory, more CPU
and so on different network performance
different EBS bandwidth and so on.
So just to give you a point of comparison,
and because we're using t2.micro in this course
it is part of the AWS feature.

So we get up to 750 hours per month of t2.micro.
And if you want a website to compare all the
EC2 Institute instances together,
there's one that I really like,
it's called instituteinstance.info,
and I'll show it to you right now.
So I am on the instituteinstances.info website
and as we can see,
we have a list of all the instances available in AWS.
So really, a lot.
We also get some information around
the Linux on demand cost and an X reserves cost,
and so on, so some cost information.
We get information around the memory the number of VCPU.
We can order by name, we can search it.
So it's, it's quite handy.
And I really like this website.
And if you go and use AWS
you probably will use this website as well.
So that's it for this lecture.
I hope you liked it.
And I will see you in the next lecture.

Let's talk about these firewalls
around our EC2 instances.
So we briefly configured one in the previous lecture,
but security groups, yet again,
are going to be fundamental into doing network security
in the AWS cloud.
They will control how the traffic is allowed into
and out of your EC2 instances.
Security groups are going to be very easy.
They only contain allow rules.
So we can say what is allowed to go in and to go out.
And security groups can have rules that reference
either by IP addresses, so where your computer is from,
or by other security groups.
So as we'll see, security groups can reference each other.
So here, let's take an example.
We are on our computer, so we are on the public internet,
and we're trying to access our EC2 instance
from our computer.
We are going to create a security group
around our EC2 instance,
that is the firewall that is around it.
And then this security group is going to have rules.
And these rules are going to say
whether or not some inbound traffic,
so from the outside into the EC2 instance is allowed.
And also if the EC2 instance

can perform some outbound traffic,
so to talk from where it is into the internet.
Now let's do a deeper dive, right?
Security groups are a firewall on our EC2 instances,
and they're going to really get
and regulate access to ports.
They're going to see the authorized IP ranges.
Would it be on IPv4 or IPv6?
These are the two kinds of IP on the internet.
This is going to control the inbound network,
so from the outside to the instance,
and the outbound network from the instance to the outside.
And when we look at security group rules,
they will look just like this.
So there will be the type, the protocol,
so TCP, the port allowing it,
so where the traffic can go through on the instance,
and the source, which represents an IP address range.
And 0.0.0.0/0 means everything.
And this here means just one IP.
Now let's look at a diagram, right?
So we have our EC2 instance,
and it has one security group attached to it
that has inbound rules and outbound rules,
so I've separated them onto this diagram.
So our computer is going to be authorized on, say, port 22,
so the traffic can go through from our computer
to the EC2 instance.
But someone else's computer that's not using my IP address
because they don't live where I live,
then if they try to access our EC2 instance,
they will not get through it
because the firewall is going to block it,
and it will be a timeout.
Then for the outbound rules, by default,
our EC2 instance for any security group
is going to be by default allowing any traffic out of it.
So our EC2 instance,
if it tries to access a website and initiate a connection,
it is going to be allowed by the security group.
So this is the basics of how the firewall works.
Now, good to know,
what do you need to know with security groups?
Well, they can be attached to multiple instances, OK?
There's not a one-to-one relationship
between security group and instances,
and actually an instance
can have multiple security groups too.
Security groups are locked down
to your region/VPC combination, OK?

So if you switch to another region,
you have to create a new security group,
or if you create another VPC,
and we'll see what VPCs are in the later lecture,
well, you have to recreate the security groups.
The security groups live outside the EC2.
So as I said, if the traffic is blocked,
the EC2 instance won't even see it, OK?
It's not like an application running on EC2.
It's really a firewall outside your EC2 instance.
To be honest, and that's just an advice to you
from developer to developer,
but it's good to maintain one separate security group
just for SSH access.
Usually SSH access is the most complicated thing,
and you really want to make sure that one is done correctly.
So I usually separate my security group
for SSH access separately.
If your application is not accessible, so timeout,
so we saw this in the last lecture,
then it is a security group issue, OK?
So if you try to connect to any port
and your computer just hangs and waits and waits,
that's probably a security group issue.
But if you receive a connection refused error,
you actually get a response saying connection refused,
then the security group actually worked,
the traffic went through, and the application was errored
or it wasn't launched or something like this.
So this is what you would get
if you get a connection refused.
By default, all inbound traffic is blocked
and all outbound traffic is authorized, OK?
Now there is a small advanced feature
that I really, really like,
and I think it's perfect if you start using load balancers,
and we'll see this in the next lecture as well,
which is how to reference security groups
from other security groups.
So let me explain things.
So we have an EC2 instance, and it has a security group,
what I call group number one.
And the inbound rules is basically saying,
I'm authorizing security group number one inbound
and security group number two.
So why would we even do this?
Well, if we launch another EC2 instance
and it has security group two attached to it,
well, by using the security group run rule
that we just set up, we basically allow our EC2 instance

to go connect straight through on the port we decided
onto our first EC2 instance.
Similarly, if we have another EC2 instance
with a security group one attached,
well, we've also authorized this one to communicate
straight back to our instances.
And so regardless of the IP of our EC2 instances,
because they have the right security group attached to them,
they're able to communicate straight through
to other instances.
And that's awesome because it doesn't make you think
about IPs all the time.
And if you have another EC2 instance,
maybe with security group number three attached to it,
well, because group number three wasn't authorized
in the inbound rules of security group number one,
then it's being denied and things don't work.
So that's a bit of an advanced feature,
but we'll see it when we'll deal with load balancers
'cause it's quite a common pattern.
I just want you to know about it.
Again, just remember this diagram.
And by now you should be really, really good
at security groups and understand them correctly.
Now, going into the exam, what ports do you need to know?
Well, we need to know something called SSH or secure shell.
And we're going to see this in the very next lectures.
This is the port 22.
And this allows you to log into
an EC2 instance on Linux.
You have port 21 for FTP or file transfer protocol,
which is used to upload files into a file share.
And you have SFTP, which is also using port 22.
Why?
Well, because we're going to upload files,
but this time using SSH,
because it's going to be a secure file transfer protocol.
Then we have port 80 for HTTP.
And we've been using it in the previous lecture.
This is to access unsecured websites.
And you've seen this whenever you go on the internet
and you enter HTTP colon slash slash,
and then the address of the website.
And you've seen most likely a lot more like this.
You've seen HTTPS, which is to access secured websites,
which are the standard nowadays.
And for HTTPS, it is port 443.
Finally, the last port you need to remember is 3389
for RDP or the remote desktop protocol,
which is the port that's used

to log into a Windows instance.
OK, so 22 is SSH for Linux instance,
but 3389 is RDP for a Windows instance.
Now, this is all the theory about security groups.
I will see you in the next lecture for some practice.

So we've launched our EC2 instance
and now let's have a look at security groups.
So we have a short idea of security groups
by just clicking on security in here.
And we get some overview
of the security groups attached to our instance
as well as the inbound rules and the outbound rules.
But what I will do is
that I will just access the more complete page
of security groups from the left hand side menu.
So under networking and security,
you click on security group.
And we can see so far
that we have two security groups
in our console so far.
So the default security group that is created by default
as well as the launch wizard one
which is the first security group
that was created when we created our EC2 instance.
And so a security group has an ID.
So an identifier, just like an EC2 instance has an ID.
And then we can check the inbound rules.
So the inbound rules are the rules that allows connectivity
from the outside into the EC2 instance.
And as we can see, we have two inbound rules in here.
And the first one is of type SSH,
which allows port 22 in our instance.
And let me just click on edit inbound rules to see better.
So set first one as SSH on port 22 from anywhere.
So 000/0 is anywhere.
And the second one is HTTP
from port 80, again, anywhere.
So this rule right here is what allowed us
to access our web servers.
So if you go back to the EC2 console,
go to our instance
and
we were doing this IPv4 address.
Okay, so we were opening it as an HTTP website.
This worked thanks to this rule, port 80.
Let's verify this.
So if we delete this rule on port 80 and save the rules,
as we can see now we only have port 22.
So if I go back to this and refresh my page,

now as we can see,
there is an infinite loading screen right here
on the top of my screen,
which shows that well,
indeed I don't have access to my EC2 instance.
So here is a very important tip for you.
Any time you see a timeout,
okay, this is a timeout
because it keeps on trying to connect
but it doesn't succeed
and then it will eventually fail, called a timeout.
So if you see a timeout when trying to establish any kind
of connection into your EC2 instances,
for example, if you try to SSH into it,
but there's a timeout,
or if you try to do an HTTP query,
but there's a timeout,
or if you try to do anything with it
and there is a timeout,
this is 100% the cause
of an EC2 security group.
Okay, so in that case,
go to your security group rules
and make sure that they are correct,
because if they're not correct,
then you will get a timeout.
So to fix this, we can add back a rule.
We will do
HTTP,
which allows to get port 80
in here automatically.
And then from anywhere IPv6, IPv4, excuse me, right here,
which allows this block right here.
We save the rule.
Now the rule is done.
If I go back to my page and refresh
as you can see, now it is fully working.
So this inbound rule really did the trick.
But we could add any sort of inbound rule.
So we could define the port or the port range
that we want to.
So we could say, for example, any port we want,
for example 443, which is HTTPS
or choose directly from a dropdown here
as a little shortcut the type of protocol you want.
For example, HTTPS is 443 automatically.
And then you can define where you want to allow from.
So you have different CIDR blocks
and we don't need them right now,
or security groups or prefix list,

but we'll get to see them later on,
okay, in this course.
For now, just know that you could have
either a custom CIDR anywhere which adds this blog
or if you want to, can select my IP
to only allow access to your IP.
But just be aware that if your IP changes,
then you will get a timeout
and will not be able to access your EC2 instance.
Finally, one last bit of information.
So we can have a look at outbound rules.
So we allow all traffic on IPv4 to anywhere.
So this allows our EC2 instance
to get full internet connectivity anywhere.
And something you should know,
so we have two security groups right here
default and launch wizard,
and an EC2 instance
can have many security groups attached to it.
So it can attach one but two or three
if you want maybe five security groups
and the rules will just add on to each other.
And also this security group we have created from default
so for example, this launch wizard one can be attached
to other EC2 instances.
Okay, so you can attach
as many security groups as you want
as well as as many EC2 instances you want
to one security group.
That's it for this lecture.
I hope you liked it.
And I will see you in the next lecture.

You're getting to one of the trickier bits
of running in the Cloud, which is how do you connect
inside of your servers
to perform some maintenance or action.
So for this, for Linux servers,
we can use SSH to do a secure shell into our servers.
So based on the operating system you have on your computer,
you have different ways of achieving it.
So I have separated Mac, Linux,
Windows before version 10 and Windows after version 10.
So the SSH is a command line interface utility
that can be used on Mac or Linux
as well as Windows over version 10.
Then if you have Windows less than version 10,
you can use something called putty.
Putty will exceed the exact same thing as SSH.
So when I say you should SSH, if you're on Windows,

you can use putty.
And putty is valid for any version of Windows.
They do the exact same thing, they allow you to
use the SSH protocol to connect into your EC2 instances.
And then finally, there's something new
called the EC2 Instance Connect,
which is going to use your web browser.
So not a terminal nut putty your web browser
to connect to your EC2 instance.
And I like it a lot because it is valid
for Mac, Linux, Windows, all versions.
The cool thing about EC2 Instance Connect is that it works,
but it only works for now with Amazon NX2
And this is why I've been using
Amazon NX2 in this tutorial.
So now what should you do?
If you are on Mac or Linux,
then please watch the SSH lecture on Mac/Linux.
If you're on Windows,
then you can either watch the Putty Lecture
or if you have Windows 10, then I have created an SSH
on Windows 10 lecture as well.
Regardless, I am going to personally use
in the future lectures, EC2 Instance Connect.
So if you wanna have a look and play with it,
I find it really simple.
You don't need to install anything
or use the command line interface
if you're not familiar with it.
So this could be very handy for all of you.
Nonetheless, SSH is in my experience
and I've taught hundreds of thousands of students
what caused the most troubles in this course.
So if you get a problem with SSH
we can re-watch the lecture, you may have missed something,
maybe a security group rule, maybe you command,
maybe a typo, I don't know you.
There's also a troubleshooting guide
that I've put together after these lectures so have a look.
I would recommend your try, EC2 Instance Connect
as well as sometimes fixes all problems.
And if none of these method works, sorry
if one method works, then you're good to go.
You don't need to have them all working.
If one works, you're good to go.
And if no method works, that's completely okay.
This course is just introductory
and he won't use SSH much and you'll be fine.
So that's it that just for the introduction.
Now find your right lecture and it will see you

in the next lecture.

All right.
So now we're going to add SSH into our EC2 Instance
using our Linux or our Mac computer.
And you may say, what the hell is SSH?
What are you talking about Stephane?
Well, SSH is one of the most important function
when you deal with Amazon Cloud.
It basically allows you
to control a remote machine or server,
all using your terminal or your command line.
So how does that look like with the diagram?
Well, we have our EC2 machine,
and we launched Amazon Linux 2 on it
and our machine has a public IP.
Now we want to access that machine.
And so for this, I don't know if you remember
but we have a security group
and on it we allowed the Port 22 of SSH.
So what's going to happen is at our computer,
so my laptop for you, for me, or whatever for you,
then we'll access over the web.
Through that Port 22,
it will access the EC2 machine.
Basically, our command line interface is going to be
just as if we were inside that machine.
So let's get started.
Okay.
So we are going to SSH into our instance.
So remember that PEM file you've downloaded
called EC2 Tutorial.pem?
Please make sure to remove the space in it
if you have a space, even if you have a PPK file.
Please rename it and remove the space from it.
So EC2Tutorial.pem is removed for me.
And then, you go ahead and place it in a directory you like.
So for me, I took my file and I pasted it,
and I placed it in a folder called aws-course. Okay?
So this is the first step to making sure you are ready.
So next, what I'm going to do is that I am going to go
in my EC2 instance overview page
and find my first instance.
So here we have my first instance
and we're going to SSH into it.
So we're going to open a remote terminal into it.
And for this, I need to get the public IPv4 address,
so I can copy this, and I will use it later.
The other thing I need to do
is to look at the security of my instance.

So again, if you did everything with me,
then your security groups have this rule in it
called Port 22 which is the SSH port
from anywhere by 0.0.0.0/0. Okay?
So if you have that rule, then you're good to go.
If not, please click on the security group
and add the missing rule.
Next, I need to try to do an SSH.
So first of all, ssh ec2-user@
and then the IP you have.
So the reason we do ssh ec2-user
is because the Amazon Linux 2 AMI
has one user already set up for us
and that user is named ec2-user.
Then we have at,
to say that we want to access that user
on the specific server.
And then, we have the IP right here.
This is the public IP of our EC2 instance.
So we try this.
So we do SSH and then we're going to get
a too many authentication failure.
So that means that we don't authenticate
into our EC2 instances.
Well, that makes sense
because we haven't specified the key
that we downloaded from before yet. Okay?
You may get another kind of error
but right now this is the one I get.
So for this,
we need to reference the file we just downloaded
called EC2Tutorial.pem into our command.
So make sure again, there is no space.
And then, you need to make sure your terminal
is exactly where your file is.
So if I do ls right here to list the files in my folder
and I'm sorry if this is too advanced for you
but I have to cover the grounds for everyone.
So if I do ls, as you see right now,
it says EC2Tutorial.pem
that's because I placed my command line
in the correct directory on my computer.
Okay.
So for this, if you were not in the correct directory,
for example if I was line level up, so I do cd ..,
which puts me one level up,
then I do ls.
Of course, I don't see my EC2Tutorial.pem.
Okay?
So to do so, what you can do is just check where you are.

So pwd is where you are.
So, I'm in user stephanemaarek
and I know that I placed my folder aws-course
within my home.
So right under user stephanemareek, there is aws-course.
So for this, then I know that I can do ls or ll
just to confirm that my folder exists.
As you can see right here is my aws-course folder,
so this is good.
So what I do is I will do cd and then aws-course
which now puts me in the directory of my aws-course.
So if I do pwd, I am in the correct directory.
And if I do ls, I can see my EC2Tutorial.pem file.
The reason we have to do this is that
because now in the next command.
So the SSH command,
you do ssh -i then you specify the EC2Tutorial.pem file
and that will not work
if you're not in the correct directory.
So please make sure to get there
and if you're missing the bit of the Linux here,
please try to go online,
but I should be good with what I showed you.
And then, ec2-user@
and we reference the public IP of our instance.
So this one right here, we reference it.
Press enter,
and now we get another kind of error
which is saying that we have an unprotected key file
and we need to change the permissions for it.
So for that reason, we'll have to enter another command
and that command is chmod.
So chmod 0400, and then we pass in the file itself
so EC2Tutorial.pem.
So I clear my screen
and then I'm going to try the exact same command as before.
So I press enter and I am logged into my machine.
So you may have seen a screen
where they prompt you for yes/no
to trust the instance as well,
just enter yes if you do get that screen.
So as you can see,
now I have done the SSH into my instance
and now it says ec2-use at this IP,
which means that now all the commands are issued
are going to be issued directly
from the Amazon Linux 2 AMI EC2 instance
that I've just launched from before.
So let's try a few commands.
For example, if you do whoami

then it says ec2-user
or I can ping google.com
and we see the google.com is responding to our pings.
So we can launch some commands directly
from the Amazon Linux 2 AMI,
and I did control C to stop that command,
Now to exit the instance itself, you can either type exit
and I think this should work,
or you do control G and then you will close the connection
into the EC2 instance.
And if you ever want to get back into it
remember this command,
ssh -I EC2Tutorial.pem
if you are in the correct directory.
Please make sure to do so.
As well as ec2-user@
and then the public IP of your instance.
Remember that if you stop and then start your instance
then the public IP can change.
So make sure to change that part as well.
All right.
So that's it for this lecture.
I hope you liked it, and I will see you in the next lecture.

Okay.
So we are going to learn how
to SSH into our EC2 Instance using Windows.
And for this, we used to say, what is SSH?
Well, SSH to me is one of the most important function,
especially when you deal with Amazon cloud.
It will basically allow you to control a machine remotely
all using the command line.
Okay.
And so what does it look like?
Well, basically we have our EC2 machine,
and it's running Amazon Linux 2, and it has a public IP.
And I don't know if you remember,
but we had an SSH security group on it.
And basically we allowed SSH on port 22 to any IP,
which basically allows our Windows machine to connect
over the internet directly into the machine
and control it using the command line.
So we'll see how to do the requirements for parameterizing
basically our Windows.
And so we'll use PuTTY to do SSH.
So this is a free tool available online.
And as you can see,
it's a little bit tricky to use the first time,
but we'll get used to it,
and we'll learn how to SSH into Windows,

into Linux using PuTTY.
So let's get started.
Okay.
So we are going to SSH
into our EC2 Instance and I'm running on Windows.
And for this, I assume that you have Windows 7,
or Windows 8, or an older version Window.
If you have Windows 10,
there is an alternative in the next lecture, both work.
Okay.
So let's try for, even if you're on Windows 10
you can do this technique.
So for this, you would go and download PuTTY.
So PuTTY is a free SSH client for Windows.
So download PuTTY.
And then I will choose, for example,
the 64 bits installer, the first one.
Then I go ahead.
I perform the install of PuTTY.
So next, next, yes.
And yes, I want to install PuTTY.
Perfect.
So PuTTY is installed.
And I have to install PuTTY.
So, we have two things here on PuTTY.
We have the PuTTY app, as well as PuTTYgen.
So let's first open PuTTYgen.
And in case you did not download your file
in the PPK format, you can actually generate the PPK format
directly from here.
So let me show you how it's done.
So, I need to go ahead and load my file.
So I click on load, and then find where my file is.
So for me, it's on my desktop.
And you can see, I see nothing,
but if I go to the bottom right, and show all files
I will find my EC2 tutorial in the PEM format.
So I can select it.
It says, okay, you have successfully imported this.
And then you can save it as a private key.
So click on save private key.
And said you wanna have a pathways, you say,
if you don't wanna have path, you say yes.
And then you save it on your desktop.
So EC2tutorial.PPK
Now your file is saved
and you have converted successfully
a PEM file into the PPK format.
If you have done this already, then you're good to go.
Next, we need to set up PuTTY

to access our EC2 Instance.
So, this time you open the PuTTY app,
and here we have to enter a host name
or an IP address of where we are trying to connect.
So this we know it's My First Instance.
You copy the public IPv4 address.
You paste it.
And it's SSH.
You're going to save this under EC2 Instance.
And then you click on save, but we're not done just yet.
Okay.
We need to specify the key itself.
So let's specify the key in here.
So, you have it saved under EC2 Instance.
I double clicked.
So as you can see, I have to accept this.
So if we accept because we trust the host,
we say, yes, accept.
But we still have the login as prompt and it will not work.
So if I do, for example, EC2 user, it says okay,
I cannot authenticate right now.
So for this, we go back into PuTTY,
and we're going to fix things.
So click on EC2 Instance and load this profile.
The first thing I'm going to add is in the host name.
I have EC2 minus user at the IP.
So the IP is where we access our server,
and the EC2 user is a user already created for us
on Amazon Linux 2.
So I can click on save again.
So we're done in here.
This will be saved.
And then for the key, we need to go into the SSH.
You click the plus, you have the Auth,
you click the plus, excuse me, no need to click the plus.
So you just click on Auth,
and then you need to find a private key file
for authentication.
Click on browse, go to your desktop.
And then you find the EC2 tutorial, that PPK file,
you have just generated using Puttygen.
Or, if you had downloaded it already
from the AWS console, that works as well.
So this file is good.
Don't click on open just yet, go back to session,
and then click on save, to save this profile.
This way you don't have to redo all these steps
all at once, over again.
So click on open.
And now it says, okay, you're authenticating

using the EC2 user.
And we have this file we just opened.
And so now we are into our Amazon Linux 2 AMI.
So we have successfully performed the SSH using PuTTY.
Okay.
So if in the course I refer to SSH, just for you,
just that means you should PuTTY into the Instance,
at least once.
And now what I can do, who am I?
And you find that I am EC2 user.
Or ping google.com and start running some commands.
So to stop this,
just do control C, and it will stop the command.
And then if you want to just exit this,
you can just close this, exit this session,
and you're good to go.
And let's check if you go back into PuTTY.
So click back on PuTTY, and you load the EC2 Instance.
Hopefully all you're seeing there is save.
So as you can see, the top settings are saved,
and my SSH Auth also has saved.
And therefore, if I click on open, then yes,
I have access directly into my EC2 Instance.
So we've successfully performed SSH
on this Windows 7 or Windows 8, through PuTTY.
And I will see you in the next lecture
in case you have Windows 10,
to show you how to SSH using Windows.

So on Windows 10,
we can use the SSH command,
so I opened Windows PowerShell
and I typed SSH and if you see this,
that means that it's available.
Otherwise, you can also use the command prompt
and do SSH and if you see this,
that means the command is available.
If you don't see the SSH command on your Windows,
that means that you don't have it and therefore,
you must use the patching method I
just showed in the previous lecture.
In my instance, I'm just going to use PowerShell
to do these exercises, okay?
So, next what I have to do
is to actually run this SSH command.
So the first thing I have to do is to be
in the directory of where my pem file is.
So right now, I'm in C:\users\stephanemaarek and I do ls,
and as you can see, well, I don't have my pem file
because it's under, for me, under desktop.

So I do cd.\Desktop to just change directory,
I clear my screen then I do ls again
and I can see my EC2Tutorial.pem file
which is the one I downloaded,
as well as the ppk file but this is not relevant,
if you don't have it, it is fine,
this is only if you want to use PuTTY.
The only file that is of relevance for us
is the EC2 tutorial.pem file.
Okay, so we need to make sure that
on the security group of course we have
the port 22 open for SSH, which we do
and next we need to enter our SSH command.
So for this, it is very similar
to the one we have on Mac, so we do SSH minus I
then you pass in the name of the tutorial file.
So I did to get this, I did EC2 and then I press tab
and I get auto suggested the EC2Tutorial.pem file.
I press tab again, I can switch between ppk and pem.
Okay so, by pressing tab, I get the autocomplete of this,
so I do SSH minus I or you can just type this,
EC2Tutorial.pem and then when to type EC2 minus user at
and at, well, the public IP of my EC2 instance
which is right here, so I copy and paste it.
So now this command is saying,
please enter this IP using this user,
the EC2 user which is the one we have
because we use Amazon next to using
this key right here, so let's press enter.
And it says, well, the authenticity
of the host cannot be trusted,
do you want to continue?
You say, yes and we are in the machine.
So sometimes you will get permission issues.
So sometimes the permissions will not be happy
and I will show you how to fix them.
So first let's exit this and clear the screen.
So in case you get a permission issue
when running this command,
what you have to do is to find your keys,
so for me, it's in my desktop.
You right click on your pem file,
you do properties, you go to the security tab
and this is where we're going to change permissions.
To do so, we're going to do advanced
and the first thing you need to make sure of
is that the owner of this file is yourself.
So it's working for me, but you can just click on change
and then for object types, you can find yourself,
locations make sure it's on your computer

and then type the object name.
So it would be for me, Stefan
but I'm already an owner so just type your name
and then you can be an owner of this file.
The owner is also indicated in here,
in your app permission entries.
The second thing you have to do
is just remove these entities,
so system and administrator don't need to have access to it
and we need to disable inheritance.
So first let's disable inheritance, for this,
we're going to remove all inherited permissions
from this object.
And then in here, I just select the principal
so myself, I just go enter Stephan Maarek.
So as we can see, I did Stephan Maarek in here,
check names and this entered my principal name
and we are going to give myself full control over this,
press okay.
So now the owner is myself and the principal
that owns that file is myself as well.
We do okay and okay,
so if I right click again on this file
and do properties under security now,
I only see myself, Stefan Maarek with full permission.
And then if you have that
and you do this command again
for sure you will not have any permission issues
and you will not be prompted with a yes,
no question ever again.
And if you wanted to, you could try this command,
so let's exit this and we can, for example,
open the command prompt, go to my desktop.
Again, if you don't go to desktop, it will not work
and then paste in this command and it will work,
as well you can do an SSH from the command prompt.
So we've successfully seen how to SSH
onto our EC2 instance directly from Windows to exit,
you can just type exit or do Control + D
and we're good to go.
And now we can get started with this course,
I hope you liked it.
And I will see you in the next lecture.

I want to show you an alternative to SSH
that I found a lot easier
which is the EC2 Instance Connect.
So for this, you click on My First Instance
and then you click on the top, it says connect.
You have multiple options including

the SSH client we saw from before.
But one option I wanna show you is the EC2 Instance Connect.
So this allows us to do a browser based SSH session
into our EC2 Instance.
For this we verified the public IP address, which is good.
The username is provided by default,
which is EC2 user because it get guessed
by AWS that we are using Amazon and X2
and therefore EC2 User is the right username,
but if you wanted to, you could override it,
but it doesn't work unless you enter EC2 users.
So we'll leave it as is for now,
and then, as you see there's no SSH key option,
because, well, when we do connect to it,
it's going to upload for us a temporary SSH key
and establish a connection this way.
So with this methodology,
we don't even need to manage SSH keys which I found lovely.
So you click on Connect and it's going to open a new tab.
And very quickly you are into your Amazon Linux 2 AMI
and you can start running some commands
such as whoami or ping google.com.
And as you can see, everything is working.
So the cool thing about it is that,
well, your session is in the browser
instead of using a different command line interface
such as terminal and so on.
And so you do ping google.com or do
any kind of commands you want really on it
without using the SSH utility beforehand.
But in this course, if I say use SSH,
you have either the option to use your own terminal mssh
or to use for example, party, or to use the SSH command
on windows or to use, regardless of you here
on windows, Linux, or Mac, the EC2 Instance Connect.
Now this is relying on the SSH behind the scenes.
So if I go, for example, to my Instance
look at the security group and I want to edit the rules.
So I click on my security group in here,
the inbound rules I'm going to edit them
and I'm going to remove the SSH inbound rules.
So I delete it and save the rules
and then go back to my EC2 Instances,
and I close this one and I'll try to establish
a new EC2 Instance Connect.
So let's connect, as you can see, this is not working
because while there's a problem connected to your Instance.
The first thing is that you need to open the port 22.
So back into my launch wizard, I can fix this.
I will edit the inbound rule, add the SSH rule

from anywhere IPV4, save the rule.
Let's go back in here and just in case,
if it doesn't work for you,
sometimes it's because you're using IPV6, excuse me.
So therefore you need to do from anywhere IPV6 as well.
So you need to add these two entries
for your EC2 Instance Connect
to work sometimes, depends on your setup.
So we're good to go.
And now if we try to connect to the Instance itself,
so let's try to connect in here.
Voila, we are into the Instance.
Okay.
So there was a quick demo of EC2 Instance Connect.
I will use it a lot in this course.
I hope you liked it, and I will see you in the next lecture.

Okay. So let's practice using IAM roles
for our EC2 Instance.
So at first, I'm going to connect to my EC2 Instance.
You can SSH, or you can use EC2 Instance Connect
if you wanted to.
I will use EC2 Instance Connect
because it's just going to be in my web browser
and a little bit simpler.
So back into my instance
with EC2 Instance Connect right here.
And we are in our Instance.
So as we can see,
we are ec2-user@ and the private IP.
So regardless if you're using EC2 Instance Connects
or SSH through your terminal, or whatever,
through PuTTY.
Then if you see this, we are at the same stage, okay?
So now you can just do some Linux commands.
For example, ping Google,
and you can get some information out of Google.
And I will do Control + C to go out of it
or issue any kind of Linux commands you want.
Okay, you don't need to know the next command
going into the exam,
but this is just a Linux terminal available
to you right now in the cloud.
So we'll type clear to clear the screen
and next we have to run some IAM commands.
So the cool thing is that's the Amazon Linux AMI
we're using right now comes with the aid of a CLI.
And so, as you can see, it is installed.
So what we can do is start using some commands.

For example, aws iam list users.
And if we do so, it says unable to look at credentials.
You can configure credentials by using aws configure.
So we could indeed run aws configure
to configure the credentials and specify an Access ID
a Secret Access key, and a region name.
But this is a really, really, really bad idea.
And the reason is that if we run aws configure
and enter our personal details onto this EC2 Instance,
then anyone else in our accounts
could again connect to our EC2 Instance.
For example, using EC2 Instance Connect
and retrieve the value of these credentials in our instance,
which is not what we want.
This is something that's really, really bad.
And so as a rule of thumb, never, ever, ever
enter your IAM APA key.
So the Access Key ID and the Secret Access key
into an EC2 Instance.
This is horrible and if you see someone doing it,
please show them this video.
Instead, what we have to do is use IAM rules.
So if you remember,
when we were in the management console and we were in IAM,
we had created an IAM role.
So let's go back into the Roles.
We had this demo role for EC2
that had one policy attached called IAMReadOnlyAccess.
So we are going to attach this role
onto our EC2 Instance to provide it with credentials.
Okay, so how do we do this?
For this, we can go into Security.
And as you can see,
there is no IAM Role right now onto our instance.
So what we can do is go back to our Instances,
Action, Security, and then Modify IAM role.
Here we have to choose an IAM role.
So we have DemoRoleForEC2 and click on Save
to attach this IAM role into our Instance.
So if you go back to Security,
now the IAM role attached to Instance
DemoRoleForEC2.
So the effect of this is that now
if we do aws iam list users and press Enter,
where we are getting a response around the users from IAM.
So as we can see, we did not run the command aws configure.
We just attached an IAM role and ran this command.
And it works.
And as a proof, if we go into this role
and detach this permission, so now it's gone,

and run the command again, we're getting an access denied.
So the role is really linked now to the EC2 Instance.
And this is how we provide AWS credentials
to our EC2 Instances only,
only through IAM roles, okay?
So if we go back to IAM
and we attach a policy to this role
and go back to IAMReadOnlyAccess,
attach this policy and then rerun the command,
we get an access denied
because sometimes it can take a little bit of time
to propagate the changes from IAM into AWS.
But if we run it one more time,
we're getting the output we expect,
which is what we want.
So this is very important
for you to understand this,
use IAM roles for your EC2 Instances.
So this is hopefully good for you.
I hope you like this hands-on
and I will see you in the next lecture.

Hi, and welcome to this lecture
on EC2 Instances Purchasing Options.
So we've been using so far, on-demand EC2 instances.
They allow us to run instances on demands,
that means they're good for short workloads,
we get predictable pricing,
and we're going to pay by the second.
But if you have different kind of workloads,
you can optimize your discounts and your pricing
by specifying it to AWS.
For example, you can use reserved instances
and you have one year or three years term,
and they're meant for long workloads.
So if you know you're going to run a database
for a long time, then a reserved instance is great.
And if you want to have a flexible instance type,
so for example, you want to change
the instance type over time,
then convertible reserved instances are for you.
And by the way, don't worry,
I'm going to do a deep dive in all of those over time.
Okay, next we have savings plan
and savings plan are one and three years term,
and they're more modern
because instead of committing to a specific instance type,
you commit to a specific amount of usage in dollars,
and there again, for long workloads.
Spot instances instead are meant

for very short workloads,
they're very, very cheap,
but at any time you can lose these instances
and that makes them less reliable.
Dedicated host allows you to book an entire physical server
and control instance placements.
And dedicated instances means that
no other customers will share your hardware.
Finally, capacity reservations allow you to reserve capacity
in a specific AZ for any duration.
Now let's look at EC2 on-demand.
So you're going to pay for what you use.
So that means that if you're using Linux or Windows,
you're going to be getting a billing per second
after the first minute,
or for all the other operating systems,
you're going to get a billing per hour.
It has the highest cost but no upfront payments
and no long-term commitments.
That means it's definitely recommended for a short-term
and uninterrupted workload
where you cannot predict how the application will behave.
Now for reserved instances,
and by the way, the numbers I show you can change over time,
so I will not update this video every time,
but it gives you an idea of what the numbers can be.
So the reserved instances give you a 72% discounts
compared to on-demand.
And you reserve a specific instance attributes.
For example, the instance type, the region,
the tenancy, and the OS.
You specify a reservation period one year or three years,
to get even more discounts,
and whether or not you wanna pay upfront,
partially upfront, or not upfront.
And all upfront of course gives you
the maximum amount of discounts.
In terms of the scope,
do you want the scope to be
into a specific region or a zone?
That means reserve capacity in a specific AZ.
And so you would use reserved instances
for the steady-state usage applications,
think for example, for a database.
And you can buy or sell your reserved instances
in a marketplace if you don't need them anymore.
And there is a specific kind of reserved instances
called the convertible reserved instance,
which is allowing you to change the instance type,
the instance family, the operating system,

the scope and the tenancy.
And because you have more flexibility,
well you get a bit less discounts
you get up to 66% discounts.
So that's for reserved instances.
And now you have the EC2 savings plans
which is to allow you to get a discount
based on long-term usage,
which is the same 70% as reserved instances.
But instead, you're going to say,
"I want to spend $10 per hour for the next 1, 2, 3 years."
And any usage beyond the savings plan
is going to be billed at the on-demand price.
So with savings plans,
you're locked to a specific instance family and region.
For example, you say,
"I want to have M5 type of instance family in us-east-1."
But you're flexible across the instance size.
So you can have m5.xlarge, m5.2xlarge and so on.
The OS, so you can switch
between Linux and Windows and so on.
And the tenancy, you can switch
between host, dedicated and default.
Now, for spot instances,
they have the most aggressive discounts,
so up to 90% discounts compared to on-demand,
but they are instances you can lose at any point of time
because you define a max price
you're willing to pay for your spot instances.
And if the spot price goes over it,
then you're going to lose it.
So they're the most cost-efficient instances in AWS
and they're going to be very helpful
if you have a workload that is resilient to failure.
So what could they be?
Well, it could be batch jobs, data analysis,
image processing, any kind of distributed workloads,
or workloads that have a flexible start and end time.
They are not suited for critical jobs or databases
and the exam will test you on that.
Next we have dedicated hosts.
So you get an actual physical server
with EC2 instances capacity
fully dedicated to your use case.
And you want to have dedicated hosts
in the use case of, you have compliance requirements
or you need to use your existing
server-bound software licenses that has billing
based on a per-socket, per-core, per VM software licenses.
This is in these kind of use cases

that you need to access the physical server
and get a dedicated host.
So for dedicated hosts, you get on-demand,
and you're gonna pay per second,
or you can reserve them for one or three years.
They're the most expensive option of AWS
because you actually reserve a physical server.
And so again, a use case is when you have
a software that comes with a licensing model
that is bring your own license.
Or if you have a company that has strong regulatory
or compliance needs.
We also have dedicated instances,
and there are instances that runs on hardware
that's dedicated to you,
which is different from the physical server.
But you may share the hardware with other instances
in the same accounts
and you have no control over instance placements.
So there's a difference between dedicated instances
and hosts, that is here.
At the exam, honestly, it doesn't trick you
into one or the other,
but remember that dedicated instances
mean that you have your own instance on your own hardware,
whereas dedicated host,
you get access to the physical server itself
and it gives you visibility into the lower level hardware.
Next, we have capacity reservations for EC2.
So you can reserve on-demand instances
in a specific AZ for any duration.
And then you get access to that capacity
whenever you need it.
You have no time commitment so you can reserve capacity
or cancel your reservation at any time.
And no billing discounts.
The only purpose is to reserve capacity.
So if you want to get billing discounts,
you need to combine it with regional reserved instances
or your savings plan.
And you're charged at the on-demand rates,
whether or not you run instances.
So that means that your reserved capacity,
you're going to be billed for it,
and you know for sure that if you want to launch instances
they're going to be available,
but if you don't launch them,
you're still going to get charged.
So they're very suitable
for short-term uninterrupted workloads

that need to be in a specific AZ.
So it gets difficult to understand
which purchasing option is right for me,
especially if you are a beginner.
But let me give you a summary.
The first one is on-demand.
So we'll take a resort as an analogy.
And with on-demand,
you have a resort and you come in whenever you like
and whenever you like, you pay the full price.
For reserved, well, you like to plan ahead
and you know you're going to stay a very long time
in your resort, one, two, three years,
and then you're going to get a good discount
because we know you're going to stay long time.
Savings plan is saying,
"Hey, I know for sure that in my resort
I'm going to spend a specific amount.
So I'm going to spend maybe $300 per month
every month for the next 12 months."
And therefore, you may wanna change the room type over time.
So king, suite, sea view, and so on.
But the savings plan is saying,
"Hey, you're to commit
to a specific spending in your hotel."
Spot instances are whenever
the hotel runs very last-minute discounts
because they have empty rooms and they wanna attract people.
So they get empty rooms
and people bid on getting this empty room.
And so you get very, very discounts.
But in this specific resort,
well, you can get kicked out of at any time
if someone is willing to pay more
for your room than what you did.
But I don't wanna stay in such a resort.
Dedicated host is saying,
"Hey, I want to book the entire building of the resort."
So you get your own hardware, your own resort.
And then capacity reservation is saying,
"I'm going to book a room,
I'm not even sure if I don't stay in,
but I know that if I want to stay in, I will have it."
And you will pay full price
for booking that room nonetheless.
In terms of price comparison,
I've just put together this table to give you
one example at one point of time.
So I took an m4.large in us-east-1,
and the on-demand price is 10 cents,

but then the spot price is going to be up to,
for example, 61% off in my example.
Then if you want to reserve your instance,
as you can see, you have different pricing.
If you wanted to have for one year, for three years,
and pay no upfront or all upfront.
Same for the EC2 savings plan
we see that it is the same as a reserved instance discounts.
We also get reserved convertible instances
and we have dedicated host,
which is at the on-demand price.
The dedicated host reservation which is up to 70% off
because you reserve your host.
And capacity reservation again at the on-demand price.
So the exam will ask you to just know
which type of instance is the right one
based on your workloads.
And by now, you should have some good hints,
and don't worry, we will practice this over time.
All right, that's it for this lecture,
I hope you liked it.
And I will see you in the next lecture.

Now again,
one more reminder into the Shared Responsibility Model
and how this applies for EC2.
So AWS is going to be responsible for all data centers,
their infrastructure and the security of them,
then AWS is going to be responsible
for making sure you have isolation on the physical host,
if you're getting,
for example, a dedicated host or replacing faulty hardware
if one of their servers is failing,
or making sure they are still compliant with the regulations
that they have agreed to.
But you as a user,
you're responsible for the security in the cloud.
So that means that you define your own security group rules.
And this allows you or other people to access
or not your EC2 instances.
You own the entire virtual machine
inside of your EC2 instance.
So that means that your operating system will be Windows
or Linux, all the patches and the updates.
You have to do them not AWS.
AWS will give you the virtual machine.
It's up to you to handle it the way you want.
That means that all the software
and the utilities that are installed on this EC2 instance,
also are yours to be responsible for.

Finally, understanding how to assign AM Roles
and make sure the permissions are correct.
And finally, making sure data is secure on your instance
is also very important for your EC2 instances.
So that's it.
Hope that again puts a bit of perspective
on to the shared responsibility model for EC2
and I will see you in the next lecture.

Let's do a summary
on what we learned for EC2.
So we have created EC2 instances,
and they were composed of an AMI,
which was defining the operating system.
Then we defined an instance size where we defined
how much CPU power we want and how much RAM we want.
We described the storage for our EC2 instance.
We defined the firewall on our EC2 instance
with the security groups.
And finally, a bootstrap script called the EC2 User Data
that was started when the EC2 instance was started.
So the security groups are attached to EC2 instances,
and they are a firewall outside of your instance.
And you can define rules to allow which ports
and which IP can access your EC2 instance.
For EC2 user data, this was a script that we launched
at the first start of an instance that we used to set up
our EC2 instance to be a web server and say, "Hello, world."
SSH was our way for us to start a terminal
from our computers into our EC2 instances
to start issuing commands on port 22.
And once we did it, we were able to leverage
an EC2 instance role that was similar to an IAM role
to have our EC2 instance issue commands against IAM.
In terms of purchasing options,
you have multiple options you need to know for the exam.
You have on-demand, spot instances,
reserved instances for standard or convertible,
dedicated host, dedicated instance, and that's it.
So that's it for this lecture.
I hope you liked it, and I will see you in the next lecture.

###----------Section 6 _____
EC2 Instance storage

Welcome to this section
where we will at look the different storage options
for EC2 instances.

So first, the most important ones
are going to be EBS Volumes, so let's define what they are.
An EBS Volume stands for Elastic Block Store.
It's a network drive that you can attach to your instances
while they run,
and we've been using them without even knowing.
So this EBS Volumes allow us to persist data,
even after the instance is terminated.
And so that's the whole purpose,
we can recreate an instance
and mount to the same EBS Volume from before
and we'll get back our data.
That is very helpful.
So these EBS Volumes, at the CCP level,
can only be mounted to one instance at a time, okay?
And when you create an EBS Volume,
it is bound to a specific availability zone.
That means that you cannot have an EBS Volume
in created, for example, us-east-1a
we'll see this in the diagram in a second.
So how do you think of EBS Volumes?
Well, you can think of them as network USB sticks.
So, it's a USB stick that you can take from a computer
and put it in another computer
but you actually don't physically put it in a computer.
It's attached through the network.
The feature gives us 30 GBs of free EBS storage
of type General Purpose or SSD or Magnetic per month.
And in this course, we'll be using this
with the GP2 to GP3 Volumes.
Now let's look at it.
So EBS Volumes are network drivers
that is not a physical drive, okay?
So to communicate between the instance and the EBS Volume,
it will be using the network.
And because the network is used,
there may be a bit of latency from one computer
to reach to another server.
Now, EBS Volumes, because they are a network drive
they can be detached from an EC2 instance
and attached to another one very quickly.
And that's makes it super handy
when you want to do failovers for example.
EBS Volumes are locked to a specific availability zones,
that means that, as I said, if it's created in us-east-1a
it cannot be attached to us-east-1b
but, we will see in this section that if we do a snapshot,
then we are able to move a volume across
from different availability zones.
And finally, it's a volume,

so you have to provision capacity in advance.
So you need to say how many GBs you want in advance
and the IOPS, which is I/O operations per seconds,
and you're basically defining how you want
your EBS Volume to perform.
You're going to get billed for that provision capacity
and you can increase the capacity over time
if you want you to have better performance or more size.
So, as a diagram, how does it look like?
Well, we have us-east-1a with one EC2 instance
and we can attach, for example
one EBS Volume to the EC2 instance.
If we create another EC2 instance,
as I said an EBS Volume can not be attached to two instances
at a time at the Certified Cloud Practitioner level.
And therefore, what I wanna say is that
this other EC2 instance needs to have
its own EBS Volume attached to it,
but it is a very possible for us to have two EBS Volumes
attached to one instance
think of it as two network USB sticks into one machine
that makes a lot of sense.
Now EBS Volumes are linked to an availability zone.
So as we can see, all this diagram has been so far using
us-east-1a.
So if you want it to have other EBS Volumes in an other AZ
then you would need to create this separately
in the other availability zone.
So just same way that's your EC2 instances
are bound to an AZ, so are the EBS Volumes.
Finally, it is possible for us to create EBS Volumes
and leave them unattached
they don't need to be necessarily attached
to an ECG instance, they can be attached on demand
and that makes it very, very powerful.
Finally, when we go ahead and create EBS Volumes
through EC2 instances, there is this thing
called a Deletes on Termination attribute
and this can come up in the exam so,
if you look at this when we create
an EBS Volume in the console, when we create an EC2 instance
there is the second to last column
called Delete on Termination.
And by default, it is ticked for the Root Volume
and not ticked for a new EBS Volume.
So this controls the EBS behavior
when an EC2 instance is being terminated.
So by default, as we can see, the root EBS Volume
is deleted alongside the instance being terminated.
So it's enabled

and the default any other attached EBS Volume is not deleted
because it's disabled by default.
But obviously as we can see in this UI, we can control
if you want to enable or disable delete on termination.
And so use case right, would be for example,
if you want to preserve the root volume,
when an instance is terminated,
for example, to save some data
then you can disable delete on termination
for the root volume, and you'll be good to go
and it could be an exam scenario at the exam.
So I hope you liked it.
And I will see you in the next lecture.

**About EBS Multi-Attach**

**About EBS Multi-Attach**

While I say that in the previous lecture that EBS volumes cannot be attached to multiple instances, I know it is not true for io1 and io2 volume types: this is called the EBS Multi-Attach feature.

**From an AWS Cloud Practitioner exam perspective this out of scope for the exam.**
In order to keep the course simple and accessible, I have left out this feature from the course.

If you are curious to learn about EBS Multi-Attach, you will find it in the AWS Certified Solutions Architect Associate course, or in the AWS documentation.

**Happy learning!**

So let's have a look
at the EBS volumes attached to our instance.
So if you click on instance
and then you go to the storage tab in here,
you find that there is a root device
and there's a block device on it.
As you can see, we got one volume
of eight gigabytes currently attached
into our EC2 instance.
So what I can do is I can click on this volume
and it will take me into the volumes interface of AWS.
And we can see that, yes, indeed,
our volume exists and it's there.
It's in use as shown here
and it's attached to an instance right here.
So we have a different kind of console here
and to access it, you can just go on the left hand side
and click on volumes.
So as we can see, now we have one EBS of eight gigabytes

and what I can do is I can create a second volume.
So let me create a volume
and I will have many options to choose from,
GP2, GP3 and so on
but I will just use GP2 of type of size two gigabytes.
And then for availability zone,
I can choose the same one where my EC2 instance is.
So for this, I'm gonna go into my EC2 instance, in here
and I will find which availability zone it is on.
So I scroll down,
and it is going to be in the networking one.
So I scroll down in the networking
and here, availability zone, it says eu-west-1b.
Therefore, the volume I will create
is going to be in eu-west-1b
because the EBS volumes are bound by specific AZ.
So that's good.
I will have it done and create this volume.
And now my volume is created.
And what I can do is if I can click on it,
this one is currently not attached.
Okay, so it's been creating
so let me refresh this to see if it's created.
Okay, it's available and it's not attached yet.
So therefore, because it's available,
what I can do is action
and then I can attach the volume
and we need to find an instance.
So we have one running right here.
And so we're going to attach this volume to my instance,
click on attach volume and voila,
our instance now has two EBS volumes attached to it.
How do we know?
Well, I can refresh this page,
go to storage on my EC2 console, scroll down.
As you can see now for block devices,
I have two block devices.
I have the one of eight gigabytes
and the one of two gigabytes.
To actually use this new block device,
it's a bit more complicated
and out of scope for this course,
but you can go to format EBS volume attach EC2,
and you should find something like, yes,
make an Amazon EBS volume available to use on Linux
and this gives you instructions on how to do it
but again, this is out of scope for this.
So now if I go into my volumes and I create a volume,
I can create a volume of two gigabytes of GP2
but this time the AZ is going to be eu-west-1a

and not eu-west-1b.
So it's gonna be a different AZ
than the one of my EC2 instance.
And the reason I do so is to show you
that right now, we have three GP2 volumes.
So let me refresh this.
So the last one is available and it's a different AZ,
so eu-west-1a.
And if I do actions and then attach volume,
as you can see, I cannot attach it to my EC2 instance
because my EC2 instance is in eu-west-1b.
And so therefore, we can see that the EBS volumes
indeed are bound by a specific availability zone.
Lastly, what I can do is that I can take this volume,
do action, delete volume, and it's gone.
And that really shows you the power of the cloud.
I can just request volumes, delete volumes
right on the go in the matters of seconds.
Okay, so we have our two EBS volumes in here
and what I wanna show you now is a cool behavior.
So what happens if I terminate my instance?
Well remember, and I will show you again,
this one root volume of eight gigabytes
has the delete on termination attribute.
So how do we know?
Well, if I go into my storage
and then go to my block devices, into this table right here,
and scroll all the way right,
you see the first one has delete on termination yes
and the second one no.
So why this one is yes?
Well, I don't know if you remember,
but when you go through the process
of launching an instance, okay?
And then you scroll down to the storage,
in here, if you click on advanced,
you can see the fact
that it is your roots of eight gigabytes
and by default, this delete on termination attribute is yes,
which makes sense, but you could set it to no
if you wanted to keep the root
after terminating your instance.
So this explains why we see from before,
the yes in this table.
And so therefore, if I go ahead and terminate my instance
which I will, so it says successfully terminated
so it's going to really remove it from here.
I can go back into my EBS volumes.
I can refresh them.
And what's going to happen is that this one

soon is going to be available
so because it's going to be detached from my EC2 instance
and then it's going to be terminated.
So I'm going to pause until this is done.
And here we go.
So my eight gigabyte volume has now disappeared.
Only my two gigabyte volume is left
and if I go to my EC2 console,
well, it says that my first instance has been terminated.
So that's it for this lecture.
I hope you liked it
and I will see you in the next lecture.

So now let's talk about EBS snapshots.
So you can take your EBS volumes and make a snapshot,
which is also called a backup,
at any point of time that you wanted to.
The idea is that you will be able
to back up the state of it,
and even if the EBS volume is terminated later on,
you could restore it from that backup.
Now to make a backup, it is not necessary
to detach the volume prior to doing the backup,
but it is recommended just to make sure
that everything is clean on your EBS volume.
And why do we do snapshots?
Well, we can obviously restore them
but we can also copy snapshots
across availability zones or regions,
and the idea is that you would be able to transfer
some of your data in a different region on AWS
to leverage the global infrastructure.
So if we look at US-EAST-1A,
and we want to transfer an EBS volume to US-EAST-1B,
the way we do it is that we would have the EBS volume
attached to the EC2 instance, and then we would snapshot it.
So maybe we would stop the EC2 instance ahead of time
to make sure the snapshot is clean
or we could just do it on the fly.
It is really up to you and based on how you have programmed
your EC2 instance.
Now the EBS snapshots exist in your region.
And that EBS snapshot can be used
to restore a new EBS volume in another availability zone.
And then now that this is done,
we can attach the new EBS volume
to an EC2 instance in US-EAST-1B.
And there we go.
We have successfully and effectively transferred
an EBS volume through a snapshot across AZ.

So some features need to know about for EBS Snapshots
is the first one, an EBS snapshot archive.
So it allows you to move your snapshots
to another storage tier called an archive tier,
and that tier is 75% cheaper.
So your snapshot is going to be moved,
you know, manually or whatever, however you set
to the archive tier.
But if you have it in the archive,
it takes you between 24 to 72 hours
to restore from the archive.
So these are for snapshot
that are not very important to you
to restore it in a rush, okay,
but you wanna save some cost on them.
The second feature is around recycle bin for EBS snapshots.
So by default, when you delete snapshots, they're gone.
But you can set up a recycle bin,
and the recycle bin will have all the snapshots
that are deleted.
And then after a while, maybe you can specify
from one day to one year,
the snapshots are gone from the bin.
So on deleting the snapshots, it would go into recycle bin.
And you may have, for example, one year
to recover the snapshots, which allows you
to protect yourself against accidental deletion.
Okay.
That's it for this lecture.
I hope you liked it
and I will see you in the next lecture.

So we have this two gigabyte,
GP2 EBS Volume available to us,
and we can take a snapshot from it.
So if we do Actions, we can Create a snapshot.
So we can add a Description, for example, DemoSnapshots,
and then click on Create snapshots.
So then on the left hand side menu,
instead of Volumes, you can click on Snapshots.
And Snapshots shows you a list
of all the snapshots you have.
And as you can see, we have one right here.
It is Completed.
It is 100% Available.
And we get some information around the snapshot itself.
For first of all, what we can do is that we can Copy
this snapshot into other region.
So if you right click and do Copy Snapshots,
then as you can see, you can copy the snapshot

into any destination region that you want.
And this can come in very handy,
in case you want for example,
to have a Disaster Recovery Strategy,
to make sure data is backed up in another region of AWS.
So I won't do that, I won't do so, but you get the idea.
Another thing I can do is take the Snapshot,
and I can recreate a Volume from it.
So Action, Create volume from snapshots.
And we choose a two gigabytes GP2 Volume.
And then the target AZ doesn't have to remain eu-west-1a,
it can be, for example, eu-west-1b.
And we can also encrypt these volumes if you wanted to
and add some Tags.
And so as we can see, when we click on Create volume,
what's going to happen is that if we go back
into our Volumes,
well, now we have two Volumes, as you can see.
And one of them,
this one was restored through a Snapshots, okay?
And it is an eu-west-1b.
So the idea is that thanks to Snapshots,
we can quote unquote copy EBS volumes
across different Availability Zones.
And that's very handy.
If we look again at Snapshots, we can have a look
at what's called the Recycle Bin.
So the Recycle Bin is a way for you to protect
your EBS Snapshots from accidental deletion,
as well as your Amazon Machine Images.
So we can Create a Retention Rule and name it
DemoRetentionRule.
I will select EBS Snapshots.
I will Apply to all resources and retain them
during one day.
And for Rule Lock Setting, I will leave this unlocked,
so that I can delete this rule whenever I want to.
And then click on Create Retention Rule.
And now on the Resources, we can see if we have Resources
in the Recycle Bin.
So let's go back into our Snapshots in the EC2 Console.
So I'm going to go into EC2.
Here we go, Snapshots.
Little shortcuts.
And what I'm going to do is take the Snapshots,
and first, before I delete it,
I wanna show you the Storage Tiers.
So right now it is a Standard Storage Tier, okay?
But you can move the Storage Tier,
by Archiving a snapshots,

and so I need to move the Storage Tier
into another Pricing Level.
But if you want to restore it,
of course, you will have to wait 24 to 72 hours.
So just to show you.
But anyway, let's go ahead and delete the snapshot.
So we'll delete the Snapshots.
As you can see, it's gone.
And before, it used to completely delete the snapshots,
and you couldn't recover it, okay?
As you can see, it's gone.
But now thanks to the Recycle Bin,
well, if I refresh here my Resources,
I can now see that my Snapshot has appeared here.
And what I can do is click on it and Recover it.
Yes, Recover Resources, and voila, it's back
into my Snapshots on my EC2 console.
Okay, that was pretty awesome, right?
Okay, that's it for EBS Snapshots.
I hope you liked it, and I will see you in the next lecture.

Now let's talk about
what powers our EC2 instances which is an AMI,
so AMI stands for Amazon machine image
and they represent a customization of an EC2 instance.
So you can use AMIs you created by AWS
or you can customize it into your own and what is in an AMI?
Well we have our own software configuration,
we can define and set up the operating system,
we can set up any monitoring tool
and if we create our own AMI
we're going to get a faster boot time and configuration time
because all the software that we want to install
onto our EC2 instance
is going to be prepackaged through the AMI.
So we have to build our own AMIs
and they can be built for a specific region
and then they can be copied across region
if we wanted to use it
and leverage the AWS global infrastructure.
So we can launch EC2 instances from different kind of AMIs.
What we've been doing so far in this course
is to use a public AMI and these are provided by AWS.
So the Amazon Linux 2 AMI is a very popular AMI for AWS
and it was provided by AWS themselves
but we can create our own AMI
then therefore you have to make and maintain them yourself
there are tools obviously to automate this
but this is a task that you have to do as a cloud user
or finally you can launch an EC2 instance

from an AWS Marketplace AMI
which is an AMI that has been made by someone else
and potentially sold by someone else
so it is quite common to have vendors on AWS
to create their own AMIs or their own software
with a nice configuration and so on
and then they will sell it through the marketplace AMI
for you to buy it and to save some time.
And even you as a user you could create a business
of selling AMIs on the AWS marketplace,
this is something that some businesses do.
Okay so the AMI process from an EC2 instance,
how does it work?
Well we'll start an EC2 instance and we'll customize it.
Then we will stop the instance
to make sure the data integrity is correct,
then we can build an AMI from it
so this will also create EBS snapshots behind the scenes
and then finally we can launch instances from other AMIs
so this is what we'll be doing
in the demo and the next lecture.
So we have US-EAST-1A,
and we can create the same instance as US-EAST-1B,
so the process is we launch the instance in US-EAST-1A,
we're going to customize it
then we're going to create an AMI from it
this will be our custom AMI.
And then in US-EAST-1B
we will be able to launch from that AMI
and we'll effectively create a copy of our EC2 instance.
So I hope you're excited
and I will see you in the next lecture.

So let's go ahead and practice using AMX.
For this, I'm going to launch an instance,
and I'm going to go into this new experience.
So I will scroll down.
We'll choose Amazon Linux 2.
I will scroll down again.
We'll use the G2 micro.
I will select the easy to draw as my key pair
but it doesn't really matter.
Then I will scroll down.
I would edit the network settings
and I will select an existing security group
being my launch wizard one from before.
Then we'll have the same storage and for advanced details,
I will scroll down and I will go for user data,
but this time you're going to copy everything,

but the last line.
So, in the first four lines, we actually install HTTPD
which is the Apache web server.
In the last line, we create an index file
but right now we don't want to create an index file
want to do everything but that to create an AMI out of it.
So we just copy everything.
Remember, the last line is the system CTL command.
And then we launch our instance.
So our instance is launched and what's going to happen is
that the instance is going to launch that's right here.
And then it's going to run
through the EC two, using the script
and it's going to install the Apache web server.
So if I'm too quick
and I try to open right now, this public IPV four
and try to enter into it, as you can see, it doesn't work.
I'll get a connection, error, refused and so on.
I need to make sure I'm also using the HTTP protocol,
of course, but it, it just won't work, right?
Because you need to give it a bit of time.
Even if it says 'running', you need to give it time
for the EC two user data script to run for the first time.
So this might take a minute.
This might take two minutes.
Okay. But don't be in a rush, just wait for it to be done.
And then at some point when you refresh
you will see the screen.
So let me pause the video.
And it took about two minutes, but now I have my test page
and this is the basic page from the Apache web server.
And therefore we are good to go.
So what we're going to do out of it is we're going to
create an AMI because we want to just save the state
of our EC two instance and reuse that.
So I right click and I will do image and templates
and then I will create an image.
So we'll call this one 'demo image'
and we're going to create our own AMI.
And then we have these settings right here.
We'll just leave it as is.
Go here and click on create image.
So now what's going to happen is that
an AMI is going to be created.
So if I go to the left-hand side and click on AMI's,
as we can see here, my demo AMI is registered.
And right now the status is pending
because it is being created.
So what I need to do is to be a little bit patient and
for it to go into the created state.

So my Amazon AMI is now created.
And what I can do now is I can launch instances
from this AMI by clicking here.
Or if you are in the instance creation page,
we can launch instances.
And this one is called 'From AMI'.
And in here in the quick start
we get access to the ones we know
but you can also go into the 'my AMI's tab'.
Okay. And you have the ones owned by me
and you can choose the demo image that you just created
from before.
So in that case, you scroll down, you select
to key pair or not, just up to you.
Network settings, again,
we edit them and we just select the existing launchers
in one security group.
And what I'm going to do is that in the advanced,
at the very bottom, I'm going to enter some user data.
So I copy the first three lines, okay.
Which are starting with the hash.
And then the last line, which is the echo on the new line.
And so what we do is that we only just write a new file.
So we don't need to reinstall HTTPD
because this AMI already contains HTTPD
which is why we will speed up in the boot up time.
This is why you would create an AMI.
So we launch an instance and now the instance is launched.
So I can just click on it.
This is the one from the AMI.
I need to wait for it to be fully created.
My instance from my AMI is now running.
I take the public IP address
and then I see the 'Hello World' from this.
And as you can see, this was much quicker
because we didn't have to install HTTPD again.
So this is the whole power of AMI's.
And you can imagine, well, it could be just more than that.
It could be security software,
a lot of prerequisite software, and so on,
you install it, it maybe it takes two,
three minutes to do it.
Then you package it as an AMI,
and then you just start from the AMI,
maybe do some end customization at the end
but you have a much faster boot-up.
And you're good to go.
So that's it for this demo.
Now to get back from this
what you do is that you take your two instances

and you can terminate them.
And that's it.
I will see you in the next lecture.

Okay, so let's talk about
a new service that I really like
and that does come up in the exam now.
It is called EC2 Image Builder.
It is used to automate the creation
of virtual machines or container images.
Quickly, what does that mean for the exam?
That means that you're gonna be able with EC2 Image Builder
to automate the creation, maintain,
validate and test AMIs for EC2 instances.
So let's have a look at diagram
to see how that works in detail.
So we have the EC2 Image Builder service
and it is automatically, when it's going to run,
it is going to create an EC2 instance
called a Builder EC2 instance,
and that EC2 instance is going to build components
and customize the software, for example,
install Java, update the CLI,
update the software system, maybe install firewalls,
whatever you define to happen on that EC2 instance,
maybe install your application.
And then once this is done,
then an AMI is going to be created out of that EC2 instance,
but all of this is obviously automated.
Then the AMI is created, but we want to validate it.
So EC2 Image Builder will automatically create
a test EC2 instance from that AMI
and going to run a bunch of tests
that you are defining in advance.
And if you don't wanna run any tests,
obviously you can just skip that test,
but the test can be asking,
is the AMI working, is it secure?
Is my application running correctly?
All these kinds of things.
And then once the AMI is tested,
then the AMI is going to be distributed.
So while EC2 Image Builder is original service,
it is possible for you to take that AMI
and distribute it to multiple regions,
therefor allowing your application
and your workflow to be truly global.
Next, EC2 Image Builder can be run on a schedule.
So you can define a weekly schedule
or you can say you can run whenever packages are updated

or you can run it manually, et cetera, et cetera.
And it is a free service,
so you're only going to pay for the underlying resources.
What does that mean?
Well, that means that if you create
an EC2 instance during this process
and EC2 Image Builder will create these EC2 instances,
then you're going to pay for these EC2 instances
and when the AMI is created and distributed,
you're going to pay for the storage of that AMI
wherever it has been created
and wherever it has been distributed,
but that should make sense, right?
So I will see you in the next lecture.

So we've seen a way to attach a network drive
onto our EC2 Instances
but they have limited performance
and I said with cause because it's a really good performance
but sometimes you want something even higher performance
and that is going to be a hardware disk
attached onto you EC2 Instance.
So the EC2 Instance is a virtual machine
but it is obviously attached to a real hardware server.
And some of these servers do have disk space
that is attached directly you know,
with a physical connection onto the server.
And so a special type of EC2 Instance can leverage
something called an EC2 Instance Store,
which is the name of the hardware,
the hard drive attached to the physical server.
So what we do with EC2 Instance Store,
we use them for better I/O performance.
We also make sure that they have good through put
and so on, so they're a great choice
when you want to have extremely high disk performance.
But the caveat is that if you stop
or you terminate your EC2 Instance,
that has an Instance Store,
then the storage will be lost.
And therefore it's called an ephemeral storage
so that means that the EC2 Instance Store
can now be used as a durable long term place
to store your data.
So what is a good use case for it then?
Well if you have a buffer, a cache,
you want to have scratch data or temporary content,
this would be a great place to do these things
but not for long term storage.

For long term storage, EBS for example
is a great use case.
Finally, in case the on the line server
of the EC2 Instance does fail,
then you'll risk to have a dear loss
because the hardware attached to the EC2 Instance
will fail as well.
So if you do decide to use an EC2 Instance Store,
then it is your entire responsibility
to make sure that you back it up
and that you replicate it correctly based on your needs.
So what I mean by better performance,
this is just an example to illustrate it,
don't need to know it.
But if you look at for example,
the Instance size of I3 dot something,
there is an Instance Store attached
to these kinds of instances
and if you look at the Read IOPS
and the Write IOPS
which correspond to how many I/O operations
we can do per second.
Then you can see at some of these random Read IOPS
and Write IOPS can reach 3.3 million or 1.4 million,
for the most performant one.
And to put this in comparison
with an EBS volume of type BP2 for example,
you can reach thirty two thousand IOPS.
So this is a lot more.
But again, it's just to illustrate my point
from an exam perspective anytime you see very
high performance hardware attached volume
for your EC2 Instances, think local EC2 Instance Store.
That's it.
I will see you in the next lecture.

Now, here's a third type of storage
you can attach onto an EC2 instance.
And this is a network file system or NFS.
So, EFS stands for elastic file system
and it is a managed network file system.
And the idea and the benefits of it
is that it can be mounted to hundreds
of EC2 instances at a time.
So, before we had an EBS volume attached
to only one EC2 instance at a time.
But with an EFS drive, you can mount it
onto hundreds of EC2 instances.
So, that makes it a shared network file system

or shared NFS.
So, EFS works only with your Linux EC2 instances.
And on top of it, it works
across multiple availability zones.
So, it is possible for an instance
in one AZ to be attaching the same EFS volume
as the instance in another AZ.
Now, EFS is highly available, scalable, pretty expensive.
It's about three times the price of a gp2 EBS volume,
but you pay per use and you don't plan for capacity.
So, that means that if you store 20 gigabytes of data
onto your EFS drive then you're only going to pay
for these 20 gigabytes.
So, if you look at the diagram, this is very easy.
We have an EFS file system with a security group,
and then we have EC2 instances
in various availability zones connected to it.
So, we have EC2 instances in us-east-1a,
EC2 instances in us-east-1b,
as well as EC2 instances in us-east-1c.
And they're all connected to the same EFS File system.
So now, let's outline the exact differences
between EBS and EFS.
So, with EBS, say we had EC2 instance
in one AZ and another one.
Then the EBS volume can only be attached
to one instance in one specific AZ.
And the EBS volumes are bound
to specific availability zones.
But if we wanted to move over the EBS volume
from one AZ to another, we could create a snapshot,
it would create an EBS snapshot
and then restore that snapshot into a new availability zone.
And effectively we would've moved the EBS volume over.
But this is a copy, this is not an in-sync replica.
This is a copy, and that would mean
that this drive can now be used by another EC2 instance.
EFS is a network file system.
That means that whatever is on the EFS drive
is shared by everything that is mounted to it.
So, what does that mean?
Say we have many instances in Availability Zone 1 on one
or many instances as well on Availability Zone 2.
At the same time, all these instances
can mount the same EFS drive, okay,
using a mount target, and they will all see the same files.
So, that makes it a shared file system.
So, hopefully that makes it a very easy way
to understand the difference between EBS and EFS.
And this is something that the exam will test you on.

There is a storage class you need to know for EFS
and it's called EFS infrequent access or EFS-IA.
So, storage class is going to be cost-optimized
for files that you don't access very often.
So, for example, you don't access these files every day.
And this storage class will give you
upto 92% lower cost for storing the data
compared to the other surge class,
which is called EFS standard.
And if you enable EFS-IA, then EFS
will automatically move your files to EFS-IA,
based on the last times they were accessed
and something called a lifecycle policy.
So, let's take an example.
We have our EFS file system.
We have three files in EFS standard
and maybe a fourth file in EFS standard.
This one hasn't been accessed for 60 days,
that means that no one has read this file.
And no one has been writing to this file either, right?
So, if you define your lifecycle policy
and you enable EFS-IA, then for example, you can say,
hey, after 60 days, please move this file
to a different storage class which is called EFS-IA,
which is going to have some cost saving.
And this is going to be done automatically.
Now, next time you access this file,
it's going to be be put back into EFS standard
because it's more accessed more often, right?
So, the idea is that this is a cost saving optimization.
And from an application perspective,
there is really no drawbacks.
It's very transparent to the applications,
that means that your applications don't even need
to know where the file is if it's an EFS standard,
or EFS-IA, it will access all these files the same way.
It's just behind the scenes,
EFS will do some cost optimizations.
So, something to need to know, going to your exam.
I hope you liked it and I will see you in the next lecture.

As usual, shared responsibility is important
going into the certified cloud practitioner exam.
And so we need to understand where is the responsibility
for AWS and yours regarding EC2 storage.
So AWS, of course, is responsible for their infrastructure,
but also because in the technical specification of EBS, EFS,
the tell you the data is replicated across many hardware,
it is AWS responsibility to perform that replication.

So that if one day some hardware is not working,
you as a customer is not impacted.
Also, anytime an EBS drive would fail,
or one part of it would fail.
It is obviously AWS responsibility
to replace that faulty hardware.
And finally, because we're talking about data storage,
it is their responsibility to ensuring that their employees
cannot access your data.
Now what is your responsibility as a customer?
Well, setting up any backup or snapshot procedures
and guidelines is very important to ensure
that you don't lose your data.
Setting up data encryption is another layer of protection
to ensure that people cannot have access to your data,
would it be AWS or other customers of AWS,
even though obviously,
security would be in place for it not to happen,
but it would be an interesting second layer of security.
Any data you set,
obviously, on the drive is your responsibility.
Anything you write to that disk is your own responsibility.
And if you're using an EC2 instance store,
you need to understand the risks
that are associated with it,
which is that you can lose the drive if somehow
there's a faulty hardware,
or that if you stop or terminate the EC2 instance
that has an instant store,
then the data will be lost.
So again, it would be responsibility
to back it up in the first place.
Okay, that's it, I hope that was helpful
and I will see you in the next lecture.

Okay, so now let's talk about Amazon FSx,
which is a managed service to get third-party
high-performance file systems on AWS.
So in case you don't wanna use EFS or S3,
and you want something else, then you can use FSX
to manage these file systems.
So you have three offerings today.
You have FSx for Lustre, FSx for Windows File Server,
and FSx for NetApp ONTAP.
And they can add file systems over time
to the FSx service, I'm pretty sure,
but they will not update this video
unless you need to know about the other ones.
So the two most important ones are going to be
FSx for Lustre, and FSx for Windows File Server,

and this is what is covered in this lesson.
The first one is Amazon FSx for Windows File Server,
which is a fully managed, highly reliable and scalable
Windows native shared file system
built on Windows File Server.
So this is meant for Windows instances.
So the way you do it is that deploy the FSx
usually across two availability zones,
and then there is support
for all the Windows native protocols,
such as the SMB protocol and Windows NTFS,
which allows you to mount this file system
onto your Windows machines.
And so if you look at your corporate data center
and you have a Windows Client, for example,
over SMB, it's able to access the Windows file server.
But also if you had EC2 instances that are Windows based,
and then they could also might as well
access this Windows file server.
So Amazon FSx is the way to deploy this Windows file server
that leverages the SMB protocol and Windows NTFS.
Because this is also a Microsoft type of offering,
there is integration with Microsoft Active Directory
for user security and Windows file server in Amazon FSx
can be accessed from AWS directly,
or as you can see on this diagram
from your on-premises infrastructure.
Now, the second flavor of Amazon FSx you have in AWS
is called Amazon FSx for Lustre.
And you need to remember that this says to have
a fully managed, high-performance and scalable file storage
for high performance computing.
So whenever you see storage for HPC,
so high-performance computing, thinks FSx for Lustre.
Why, well Lustre, there is derived from the name
Linux and cluster, so put together it's Lustre.
And so imagine cluster like processing
this kind of things, maybe it's a way for you to remember
what Amazon FSx for Lustre is.
This allows you to run a lot of use cases
for high performance computing, such as machine learning,
analytics, video processing, financial modeling,
and it scales to extremely high traffics
in terms of hundreds of gigabytes per second
of data exchanged, millions of IO operations per second,
sub milliseconds latency,
so it's really a high-performance file system.
So the way it works is that Amazon FSx for Lustre
can be connected either to your corporate data center
or to your compute instances directly within AWS.

And then in the backend, Amazon FSx for Lustre
is actually storing your data,
possibly onto an Amazon S3 bucket.
So that's it, that's all you need to know for Amazon FSx
there is no easy hands-on to do it.
So we'll skip that, but just remember the two flavors
of Amazon FSx going into the exam,
and you should be good to go.
So that's it, I will see you in the next lecture.

So let's summarize what we have learned in this section.
The first one around the EBS volumes,
they are network drives that are attached
to one EC2 instance at a time.
They're mapped to a specific Availability Zone
and we can use the EBS snapshot feature to do backups
or to be able to transfer the data from one EBS volume
across to an other Availability Zone.
We've seen about AMIs.
There are ready to use EC2 instances images, with
the customizations we want.
And if we want to automate the process of building an AMI
we can use the EC2 image builder service to
automatically build tests and distribute AMIs.
We've seen EC2 instance store, which is a way
for us to have a very high performance hardware disc
this time, attached to our EC2 instance.
But it is lost
in case our instances stopped or terminated.
If we wanted to have a network file system,
we would use the EFS service.
With EFS service,
we can get a NFS that we can attach
to hundreds of instances within the region.
So there is no AZ luck,
now it's at the region level.
And if you want to cost-optimize our EFS service,
we could use EFS-IA,
which allows us to move the files that are infrequently used
into a lower cost tier.
Finally, we've seen about FSx, FSx for Windows,
which is a way for us to get a network file system
for Windows servers
and FSx for Luster, which is
for us to perform high performance computing.
So HPC on a Linux file system.
So to say for this section, I hope you liked it
and that I will see you in the next section.

So let's do a quick cleanup just to make sure

we have a clean slate and that we don't overpay anything
even though we are in the free tier.
So if you go to the EC2 Dashboard, you will see all
the resources running in your region.
So I have two running instances.
I'm going to open this in new tab.
Four volumes and then four key pairs.
You can leave it as is.
It doesn't cost you any money, no matter what.
Snapshots, I have two so I will open this in new tab.
And then finally, security group.
As well, it don't cost you any money so I'm going
to leave them as is.
So for our EC2 instances, you can take them all,
right-click, and then terminate them.
See, this is very easy.
This is the cloud.
Then for the volumes, we have some root EBS volumes
that are going to be deleted as we terminate our instances.
But for these two EBS volumes, I'm going to delete them
as well manually, okay.
And finally, for the snapshots, I can delete
the snapshot right away.
So I can take these two things, do action,
delete the snapshots, yes.
And then it says this one is not being able to be deleted
because it is in use by an AMI.
So I will say okay, go to AMI on the left-hand side.
I'm going to delete the AMI so I can just deregister it.
I say yes, I want to be deregistering this AMI.
And back into our snapshots, now I'm able to get this one,
delete it, and say yes. And now we should be good to go.
So if I look at instances, all terminated.
If I look at EBS volumes, none of them.
And snapshots, empty, and AMI, empty.
Okay, that's it for the cleanup.
I'm going to refresh this and as we can see,
we are good to go.
And I will see you in the next lecture.

#_____Section 7 _____#

Welcome to this session
on Elastic Load Balancing and Auto Scaling Groups.
This is a section where we really see the power
of the AWS cloud and the cloud in general,
and see how these new paradigms we saw
really help us shine
and make our application scales seamlessly.
So let's discuss the concept

of Scalability and High Availability.
So if your applications can scale,
that means that it can handle greater loads by adapting.
And there are two kinds of scalability in the cloud.
There is vertical scalability and horizontal scalability,
also called elasticity.
So the scalability is going to be linked,
but different to high availability.
And these things are going to be discussed in this lecture.
So let's do a deep dive.
And we'll be using a call center as an example.
So imagine, we have a call center and we just receive calls.
Now let's see what it means to be scalable in that case.
So first, let's deal with vertical scalability.
In AWS, when you are vertically scalable,
that means that you can increase the size of the instance.
So for our call center, say we have a junior operator
and say we were able to upgrade that operator,
we would get a senior operator.
And for example, the senior operator
can handle a lot more calls than the junior operator
because it's more experienced.
So this would be what vertical scalability looks like
in a call center.
If we could upgrade obviously,
a junior operator into a senior operator.
So in AWS, for example,
say your application runs on the t2.micro,
and to do a vertical scalability for that application,
that means that now we run our application on a t2.large.
So we've changed the size of our EC2 instance.
And vertical scalability is very common
when you have a non-distributed system,
for example, a database.
If you want to give more performance to your database,
you would just increase the size of your database.
But usually with vertical scalability,
there is a limit to how much you can vertically scale
and that is a limit of the hardware.
Even though nowadays
these limits can be very, very, very high.
Okay, next is horizontal scalability.
So that means that
now instead of increasing the size of your EC2 instance,
you increase the number of instances
or systems for your application.
So back into our call center example,
we have an operator,

and we want to do horizontal scalability for that operator,
that means we will add another operator.
And if we need to handle more calls,
we will add another operator, and so on.
So maybe we can scale horizontally from one operator
all the way to six operators.
So when you have horizontal scaling,
that implies as you can see on the right hand side,
that you need to have a distributed system.
And for call center, that makes sense.
You don't need these people to be talking constantly.
For a call center, each of the individual operators
can take calls on their own.
In AWS, or for web applications,
so this is going to be very common,
so if you have a web application or a modern application,
you usually design it with horizontal scalability in mind.
And it's super easy on AWS to scale,
thanks to Amazon EC2 and auto scaling groups,
and we'll see this in the section.
So now let's talk about High Availability.
And it goes hand in hand with horizontal scaling.
High availability means
that you are running your application and system
in at least two availability zones on AWS.
But for our call center, what does it mean?
That means that we have a call center in New York,
and maybe a second call center in San Francisco.
And somehow, if one of these call centers is down,
for example, say there's a power outage in New York,
then we can still take calls in San Francisco.
And so we are highly available.
Obviously, San Francisco will be more busy,
but we are at least surviving the disaster of a power outage
in one of our buildings.
So in AWS you use two availability zones, obviously.
And the goal of it
is to usually survive a data center loss, a disaster.
And in AWS, it could be an earthquake,
that could be a power outage that could a lot of things.
Okay, so to summarize,
High Availability and Scalability for EC2.
If we have vertical scaling,
that means that we're increasing the instance size.
So we can scale up if we're increasing it
or scale down if you're decreasing it.
So we can go all the way
from a T2.nano of 0.5 gigabytes of RAM,

and one vCPU, all the way to,
and obviously they can change over time,
a u-12tb1.metal, which is a very scary name,
but has 12.3 terabytes of RAM, and 448 vCPUs.
That is for vertical scalability.
Now for horizontal,
this is when you increase the number of instances,
it's called scaling out,
or when you decrease the number of instances,
it's called scaling in.
And for this, we'll be using an auto scaling group
and a load balancer.
This is the topic of this section,
and then when we have high availability,
that means that we run the instances of the same application
across multiple availability zones,
and this will be again leveraged by an auto scaling group
in multi AZ mode.
And a load balancer in multi AZ.
One last word.
So the exam will as could you figure out
is this scalability, is it elasticity, is it agility?
And so I just wanna give you some formal definitions,
so to summarize.
Scalability is the ability for a system
to accommodate a larger load
by making the hardware stronger or scaling up,
or by adding nodes scaling out.
This is when your application can scale.
Now, elasticity is something a bit more cloud native.
This is once a system is actually scalable,
so you can either scale up or scale it out.
Elasticity means that there will be some sort
of auto scaling in it,
so that the system can scale
based on the load that it's receiving.
And in this case, we're going to pay per use,
we're going to match the demand we're receiving
with a number of servers,
and obviously, we're going to pay just the right amount
so we will optimize cost.
So in AWS, elasticity is a key concept.
And agility is absolutely not related to scalability
or elasticities, it is a distractor.
But just to remind you what it means,
agility means that the new IT resources
are only a click away,
which means that you can reduce the time

to make these resources available to your developers
from weeks to just minutes.
And so your organization is more agile,
it can iterate more quickly
and you are going faster, okay?
So that's it for this section on introduction.
I hope you liked it
and I will see you in the next lecture.

So let's see, the first service
that will allow us to be more elastic
on AWS, this is called elastic load balancing.
So a load balancer is a server that will forward
the internet traffic down to multiple servers downstream.
And for then there will be EC2 instances.
They're also called the backend EC2 instances.
So elastic load balancing is something
that is managed by AWS.
So we have a load balancer
and this is what we will be publicly exposing for our users.
And behind that load balancer,
we will have multiple EC2 instances,
maybe three in that case.
And then we have our first user talking
to our load balancer, okay?
And the load balancer will be directing the traffic
to one of these EC2 instances.
And the EC2 instance will reply back
with something and the user will get the response.
But now if a second user comes in,
then they will get the reply from another EC2 instance.
And if a third user comes in
as you can expect, it will be replying
from another EC2 instance.
And so the load balancer, the more users we have,
the more it will balance the load
across multiple EC2 instances
and that will allow us to scale better our backend.
So why would you use one?
I think it's clear.
You can spread the load
across multiple downstream instances.
You can expose a single point of access,
DNS host name, for your application.
You can seamlessly handle the failures
of downstream instances.
So we do regular health checks

on them and if one of them is failing,
then the load balancer will not direct
traffic to that instance.
So we can hide the failure
of an instance using a load balancer.
We can also provide SSL termination,
so HTTPS for your websites very easily.
And you are able to use a load balancer
across multiple availability zones
which was making your application highly available.
Okay, let's keep on going.
So the ELB is a manager load balancer,
so you don't need to be provisioning servers.
AWS will do it for you
and AWS will guarantee that it will be working.
AWS will take care of all the upgrades, maintenance,
and high availability of that elastic load balancer.
And the only thing we have to do is to configure
a few things for the behavior of that load balancer.
It's obviously less expensive
if you want to set up your own load balancer on EC2.
It is definitely possible, but in the end,
there will be a lot more effort on your end
for maintenance, integration, maintaining and taking care
of the operating system, upgrades, et cetera, et cetera.
So there are four kinds of load balancers offered
by AWS and I need you to know the differences between them.
So the first one is the application load balancer,
which is for HTTP or HTTPS-only traffic
which is called a Layer 7 type
of load balancer because it's HTTP and HTTPS.
Next, we have the network load balancer.
It's ultra high performance.
So look for that keyword.
It allows for TCP and UDP actually.
And it's Layer 4.
So it's Layer 4 because it's lower level, so TCP and UDP.
Then we have the gateway load balancer.
It's Layer 3.
I will show you the differences in the next slide.
And then finally, we have the classic load balancer
but it's being retired in 2023,
so it's not going to appear at the exam anymore I feel.
But if you wanna know, it was a Layer 4
and Layer 7 type of load balancer of older generation.
And it's been replaced by the ALB,
the application balancer
and the NLB, the network load balancer.

So if you have a look at the differences between the ALB,
the NLB, and the gateway load balancer, also GWLB,
then what you'll need to look
at for the exam are these kind of keywords.
So if you see HTTP, HTTPS or gRPC protocol,
it means it's Layer 7 and it's the ALB.
Also, anytime you need HTTP routing features,
this will be requested.
For a static DNS as well,
this would be very helpful if you wanna have a static URL.
So the architecture is very simple.
The users access your load balancers
on one of the protocols I just mentioned
and then the load balancer routes traffic
to the downstream EC2 instances.
For example, if you've chosen the targets
to be EC2 instances.
For the network load balancer,
it is Layer 4.
So TCP and UDP protocol, and it's very high performance.
We're talking millions of requests per second.
It gives you a static IP this time, so not a static URL,
but a static IP through the use of elastic IP
which are IPs that you own that you can move around.
So this NLB gives you a static IP
and the architecture is the exact same as the ALB.
The traffic is being sent
to the NLB on the TCP and UDP protocol, and then sent,
forwarded to downstream targets.
For example, E2 instances.
Now, the gateway load balancer is using the GENEVE protocol
on the IP packets themselves.
So it's Layer 3.
And the use case you need to look
at for the exam is to route traffic
to firewalls that you manage on EC2 instances,
so that you can do, for example, classic firewall
or intrusion detection or deep packet inspection.
And the architecture, it is a little bit more complicated.
So the gateway load balancer doesn't balance
the load to your application.
It actually balances the load of the traffic
to the virtual appliances that you run
on EC2 instances so that you can analyze the traffic
or perform firewall operations.
That's why it's called
third-party security virtual appliances.
And they can be, many of them address

represented one on this diagram.
And so therefore, the traffic, when it goes
to the gateway load balancer, first sends the traffic
to these EC2 instances that will analyze the traffic.
The traffic will be sent back afterwards
to the gateway load balancer
and then forwarded back to the applications.
So the gateway load balancer here is
in the middle to allow us to inspect the IP packets themself
and to perform firewall features
or intrusion detection or deep packet inspection.
Okay, so if you understood this, you know the differences
between the load balancers and you'll be good for the exam.
And I will see you in the next lecture

So we are going to practice launching
a load balancer, but first,
we need to send traffic to something.
So first we're going to launch EC2 Instances.
So I'm gonna go into launch instances
and I will launch two instances.
So on the right hand side I can say two instances
and the name's going to be My First Instance.
We'll rename the second one when it comes to it.
We're going to use Amazon Linux 2 on this architecture.
We're going to use a t2.micro,
and then we are going to proceed
without a key pair because we don't need SSH capability.
We can use EC2 Instance Connect if we ever need to.
Then for network settings
we can select an existing security group
and we will use the Launch Wizard 1 security group
which allowed us to do HTTP traffic
and SSH traffic into our EC2 instance.
So that's perfect.
We're going to use the basic storage
and for advanced details, I will scroll down
and I will add some EC2 user data,
and to do so I'm going to copy what I have here
and paste it here.
So this will just launch the EC2 instances
the same way we've launched them before
using this EC2 user data script.
So let's launch our two instances
and now we're going to view all instances.
So I'm going to rename the second one
My Second Instance and save.

So let's wait for these instances to be ready.
So my EC2 instances are now ready.
I'm going to copy the first IPv4 address and paste it,
and I will visit the URL
and as you can see, I get a hello world from my instance
so this is great.
And then I'm gonna go to my second instance right here.
I will copy again the IPv4 and then paste it, press enter,
and I get a hello world again.
So as you can see, two instances give us two hello worlds,
and the last part is changing.
And so what we'd like to do is to have only one URL
to access these two EC2 instances
and balance the load between them.
So for this of course, we're going to use a load balancer.
So let's scroll down and look at load balancers.
And here you can create a load balancer.
So we have different load balancer types,
and in this demo we're going to only look
at the application balancer, but you need to
understand the difference between the ALB,
the network load balancer, and the gateway load balancer.
So for the application of balancer,
you can see here it is for HTTP
and HTTPS kind of traffic.
For the network load balancer it's going to be on the TCP
and UDP protocol or TLS over TCP.
And this is something you going to use
when you need ultra high performance.
That means millions of requests per second
while maintaining ultra low latency.
So this is a very high performance load balancer, this one.
And then finally the gateway load balancer right here,
as you can see, it's used for security,
for intrusion detection, for firewalls and so on.
So it's to analyze the network traffic.
When it goes to the classic load balancer,
by the time you watch this video, this may be gone
because the classic load balancer is going away
and so therefore I'm not going to discuss it and touch it.
Okay, so let's focus on creating
the application load balancer.
So I'm going to call this one DemoALB.
And if you wanted to read about how load balancing works,
you can read it here,
but hopefully the previous lecture was enough for you.
So this scheme is internet facing
and the address type is IPv4.

For network mapping, we we need to decide where
to deploy the load balancer and how many availability zones.
So let's deploy it in all of them.
Great, and then we need to assign a security group
to our load balancer.
So it turns out that I'm going to create
a new security group for it
and we need to only allow HTTP traffic.
So I'll call it demo-sg-load-balancer.
Allow HTTP into load balancer, into ALB,
and the inbound rules
is going to allow all HTTP from anywhere.
Okay, and the outbound rules are fine.
Let's create this security group.
So it is now created
and I can go back in here, refresh this page,
choose my demo-sg-load-balancer
and remove the default security group
so that I'm only left with one security group.
So let's scroll down.
And we are under listeners and routing.
And so we need to route the traffic
from SHTTP on port 80 to a target group.
And a target group is nothing more
than a group of my EC2 instances that were created.
So for this we need to create a target group.
So let's click here to create one.
And the basic configuration tells us that we want
to group instances together,
but you can see you have other options.
So we want to group instances together
and I'll call this one demo-tg-alb.
The protocol is HTTP on port 80.
You have different options, but based on the option
you choose, it's going to be a target group
for a different kind of load balancer.
So we'll keep it as HTTP on port 80, the version
of HTTP is 1 so we'll keep it as 1.
The health check is good.
And then let's click on next.
And then we need to register our targets.
So we're going to register both EC2 instances on port 80,
and let's include them as spending below.
So now my instances are registered
and let's create this target group.
So it's created, and now I need to refresh my page
and actually I had created one before
so the one I wanna use is demo-tg-alb.

So this target group is created
and it's linked to the listener
on my load balancer on port 80.
So now I'm good to go
and I can go ahead and create my load balancer.
So I'm going to click on view load balancer,
and I'm back into this page
where I can have a look at my load balancer.
And right now it is in the provisioning space
so we need to wait until it is provisioned.
So my ALB is now active, it's ready.
And as you can see, there's a DNS name available for me.
So I'm going to copy this, paste it in a new tab
and through the application load balancer
I'm able to get a hello world.
But the cool thing about it is
that if I refresh this page
and keep on refreshing it,
then as you can see the target is changing.
That's because my application load balancer
is actually redirecting between both my EC2 instances,
which is very cool.
And that's the proof
that load balancing is actually happening.
How do we know?
Well, if we go to our target group, this one,
and we look at the targets of my target group,
as you can see they're both healthy.
That means that the application load balancer
through the target group is going to send traffic
to both of them, one after the other.
And the target group is very smart
because if I take my first instance for example
and I stop it,
through this, what we're doing
is that we're stopping our two instance
and so therefore it's going to be unhealthy
because it cannot respond anymore to the traffic coming in.
And so if I go in my target group,
maybe I'm too fast, let's see,
and refresh, so I will wait about 30 seconds.
And now as you can see
the first instance is unused because it's in stopped states.
And so therefore, if I go back
to my applicational balancer
and refresh, the only response I'm getting
for this instance is that one instance
that is still up and running.

This is the power of using load balancers
because they know when the targets are healthy
or not healthy.
And so this instance is stopped,
but of course if I recover it, if I start it again
then it's going to boot up
and is going to create the service behind the scenes.
And so let's wait for the instance to be started
and hopefully we'll see it again
as being healthy in our target group.
The instance is now up
and we are in the initial health status as you can see
and now we are in a healthy status.
So the instance was deemed healthy
and so therefore if I go back
to my application balancer and refresh, as you can see now,
the hello world is coming from both instances.
So that's it, we've practiced load balancer,
we created one as well as two targets in the target group.
I hope you liked it, and I will see you in the next lecture.

Okay, so now new have an application
that can be load balanced through a load balancer,
but how do we create, automatically
these servers in the back end.
For this we can use an auto scaling group.
So why?
Well in real life you're load on the websites
can change over time.
So for example, say your users are doing shopping,
they're most likely doing shopping during the day
and not at night.
So you expect more load during the day
and less load during the night.
So in the cloud we know we can create
and get rid of servers very quickly
and so the goal of an auto scaling group is to scale out.
That means add EC2 instances to match an increased load
or scale in, that means remove EC2 instances
to match a decreased load.
With this we can ensure that we have,
also as well, a minimum,
and a maximum number of machines
running at any point of time
and once the auto scaling group does create,
or remove EC2 instances we can make sure
that these instances will be registered,

or de registered into our load balancer.
So these two things work hand in hand.
Finally, in case one of our servers becomes unhealthy,
maybe there's an application bug,
then the auto scaling group can detect it
and say, yeah, you know what,
I don't need an unhealthy instance.
I'm going to de register it.
I'm going to terminate it
and replace it by a new healthy one.
So with an auto scaling group we get a lot of benefits
and another benefit we get,
is that we have huge cost savings,
because we are only running all the time
at the optimal capacity
and that is one of
the guiding principle of the cloud, elasticity.
So if we look at our autos scaling group in AWS.
This is it.
We have a minimum size, maybe it's one EC2 instance.
Then there is a setting called desired capacity,
which is also usually the actual size
of your auto scaling group
and then finally,
you can define a maximum size of you auto scaling group
and automatically your ASG, auto scaling group,
can scale out as needed
or scale in as needed by adding EC2 instances over time
and it work hand in hand with a load balancer.
So that means that if we have our auto scaling group,
for example, with one EC2 instance,
web traffic can be coming in through our load balancer,
which will be redirecting the traffic
directly into your EC2 instance
and as our auto scaling group scales out
by adding EC2 instances,
the load balancer will have them registered
and will send traffic to them as well.
So as we add on more and EC2 instances,
the load balancer distributes more
and more of the traffic,
all the way to the maximum size of your auto scaling group
if it scales all that point.
So that's it for this lecture.
In the next lecture
we will be reproducing that very same set up
with an auto scaling group, multiple EC2 instance,
and a load balancer.

So I hope that was helpful
and I will see you in the next lecture.

So before we go ahead
and practice creating an auto scaling group,
you need to take your first two instances
and we're actually going to terminate them.
Okay, so now this is done,
we can go ahead and create an auto scaling group.
For this, on the bottom left, click on Auto Scaling Group
and we will create an auto scaling group.
So I'll call this one DemoASG,
and we need to create a launch template.
So currently we have none.
So let's create a launch template,
and I will call this one DemoLaunchTemplate.
And this template is being used to tell
to the ASG how to create EC2 instances within it.
So this will look very, very similar
to what we have when we create EC2 instances.
As you can see here, I can choose, for example,
a quick start Amazon Linux for getting Amazon Linux 2
as the base of my EC2 instance.
Then we have an instant type that we can include,
for example, t2.micro.
For key pair, we will not include it in the launch templates
or we can just say that, no, we don't need one,
so this is good enough.
For subnets, so we'll not include this
in the launch templates.
For security group, we can select
a security group that's already existing.
For example, my launch-wizard-1,
under advanced network configuration
we don't need to do anything.
For EBS volumes for storage we don't need to do anything.
And then for advanced details
we want these instances to start with some user data.
And so we scroll all the way down
and here we copy and paste the user data.
Okay, so let's create this launch template.
As you can see, thanks to this launch template
we launch EC2 instances just like before.
So let's refresh this
and then click on the DemoLaunchTemplate of version 1.
So here it describes what is going to happen, the type
of instance we're going to have, the security groups
and so on.
So let's click on next.
Next we need to choose where to launch these instances.

So we have our VPC,
and then we can select multiple
availability zones and subnets.
So we select three of them.
And for instance type requirements, we can use the one
from the launch template or if you wanted to override them
but we don't need to actually.
So let's click on next.
Next we have load balancers.
So we want to attach to an existing load balancer
and it's going to be a application balancer.
And to do this, we're going to tell the ASG,
the auto scanning group, that every instance created should
be registered within my demo target group for my ALB.
So therefore all these instances will be
under the target group,
and then the load balancer will be able
to direct traffic to them.
So the health check can be EC2 and also ELB.
So we're good.
And then we can click on next.
Now here comes the scaling of the auto scaling group.
So the desired capacity is how many instances you want
at any time.
For example, we want two EC2 instances
to have some sort of load balancing.
The minimum capacity is one,
meaning we want at least one instance,
and the maximum for them is we want at most four instances.
But the desired is what matters the most
because this is the actual capacity that we're going to get.
Okay, so then do we want scaling policies?
This is too advanced, but you of course can set
scaling policies on a auto scaling group.
That's the whole point of it
which allows you to resize your ASG on demand.
If there is much more demand than is going to
have more instances, if there is less requests,
less demand than is going to be less instances.
So click on next.
Next, we don't need notifications, we don't need tags.
We can review everything.
It looks good.
And let's create our auto scaling group.
So now our auto scaling group is being created
and as you can see, the state is updating capacity
because we have zero instances in our ASG, but we want to,
so I can click on it to get a bit more details.
So let's go under activity.
And in here we have two activity history.

That is we are launching two new EC2 instances
because well, the desired capacity went from zero to two.
And so if we have a look
under the instance management tab, as you can see now,
2 EC2 instances are independent states.
So if I go under EC2 and look at my EC2 instances,
in that UI, we also see that two instances are running
and these have been created by my auto scaling group.
So the benefit is that now they are fully managed
by my auto scaling group,
and let's go see for example, in my target group as well.
So if I go to my target group on the left hand side
and look at my demo-tg-alb right here, scroll down.
As you can see now we have two total targets
and these are the targets created
by our auto scanning group.
So again, thanks to the integration that we've defined
between the auto scanning group
and the load balancer, we are able to have automatically
these new EC2 instances registered as targets
in our target group.
So currently they're unhealthy,
this is because the instance hasn't started all the way yet
so let's wait a little bit until they become healthy.
So to speed up the check from unhealthy to healthy,
you can go under health checks of your target group,
and here you can edit these settings.
And under other settings, we can say
that the healthy threshold is going to be 2
and the interval is going to be 5 seconds.
This is going to make the thing much quicker.
So of course the timeout needs to be 2 seconds,
something less than the interval itself.
So let's save our changes.
And now the health check change settings have changed.
Let's see, if I go back into my targets
and refresh, now both my instances are healthy.
We just made the health check happen faster and more often.
So now both my instances are healthy,
and so therefore if I go under my load balancer right here
and I look at the DNS name
and open it in a new tab, I get a hello world
from both my instances.
And this is cool because these two instances were created
by the auto scaling group.
So because now we have an auto scaling group
we can actually do some cool stuff.
So if we take one of these instances, for example
and we can for example terminate it,
so I'm going to click on it

and I'm under the instance itself
I will do instance date and then terminate instance.
Now it's been successfully terminated.
So what's going to happen is that
because this instance is being shutting down
and terminated, well, my auto scaling group
is going to detect that, guess what, one of these instance
is not in service anymore, it's being terminated.
And so therefore, because we have an auto scaling group
with a desired capacity of two instances,
automatically a new instance should appear.
So let's observe this behavior
by having here the activity history.
And as you can see,
in progress was terminating EC2 instance.
And so an instance was taken out of service
because, well, it's been terminated.
And then we have a new activity saying,
hey, an instance was launched in response
to an unhealthy instance needing to be replaced.
So it's very cool because the auto scaling group
can automatically detect unhealthy instances
and create new one for replacements.
So if we go here now, there's one instance in pending state
which is being started, one instance being terminated
and one instance in service.
And this is the whole power of auto scaling groups.
Of course you can go to the next level
but for now we know enough
which is around automatic scaling to actually
define scaling policies to automatically increase
or decrease the desired capacity over time,
based on our load and so on.
But here you've seen the basics
and the major features of auto scaling groups,
and you could play around
by editing the desired capacity yourself
to set it to one, for example, to terminate instances
and only keep one of them, or to set it to four
and see the auto scaling group creates multiple instances
that will be registered with our load balancer.
And so therefore the traffic is going to be distributed
between four instances.
So I hope you liked it
and I will see you in the next lecture.

Okay, so we've seen how Auto Scaling Groups works
but let's have a look at the different Scaling Strategy
for your Auto Scaling Groups.
So the first one is to do Manual Scaling.

Which is when we update the size
of a Auto Scaling Group manually.
And this is when for example,
we changed the capacity from one to two,
or back from two to one.
Then we can define some Scaling Strategies such as
Dynamic Scaling to respond
to changing demands automatically.
So we have different type
of scaling policies within Dynamic Scaling,
we have the Simple and the Step Scaling
which is the idea is
that whenever a CloudWatch alarm is triggered for example,
you say whenever the average CPU utilization
of all my EC2 instance goes over 70% for five minutes,
then add two units to capacity to my ASG.
Or when another alarm for example,
say whenever the CPU utilization is less
than 30% for 10 minutes.
Then remove one unit of capacity in my ASG.
This would be Simple or Step Scaling
because we define the trigger,
and then we define how many units we add or remove.
Then we have Target Tracking Scaling,
which is a very easy way of defining a scaling policy.
The example is to say,
hey I want the average CPU utilization
of all the EC2 instances in my ASG
to stay at around 40% on average
and then the ASG will scale automatically
to make sure that you stay around that target of 40%.
And we have also Scheduled Scaling.
So this is when we know
that changes are going to happen ahead of time.
So we anticipate a scaling based on known users pattern.
And for example we're saying,
hey we know that's on Friday at 5:00 PM.
People are going to do sports betting maybe who knows,
before the soccer game,
and so please increase the minimum capacity
to 10 EC2 instances in my ASG
at 5pm on Friday.
This could be a Scheduled Scaling
and there's one last type of scaling.
that is definitely appearing on the exam
which is called Predictive Scaling.
So this one uses Machine Learning
to predict future traffic ahead of time
so there's some algorithms,
they will look at the past traffic patterns,

and it will forecast what will happen
to traffic based on the past patterns.
And so the idea is that
it's called predictive because we predict
what the load will be over time,
and maybe the load is just on a daily basis
it peaks for three hours.
So this is the kind of things that
Predictive Scaling will pick up, okay.
And it will automatically provision
the right number of EC2 instances
in advance to match that predicted period.
So this is what the graphs you see on the right hand side.
This is very helpful
when you have time-based patterns
and you just want to have an easy,
without any intervention type of scaling
trust strategies
that is powered by Machine Learning,
then that would be Predictive Scaling.
So that's it for this lecture.
I hope you liked it.
And remember the strategies.
I will see you in the next lecture.

So, we are going to clean up our instances
and so on, but if you try to go in instances
and actually terminate these two instances,
this will not work, because if you do so,
then the auto-scaling group will recreate them.
So, the strategy here is to actually go
under the auto-scaling group,
and we're going to delete the auto-scaling group altogether,
so just type 'Delete' in here to confirm the deletion of it,
and then the next thing we have to delete
is the low balancer,
so find your application balancer action and then delete.
Confirm to agree and you're good to go and finally,
you may be wondering, well,
should I delete my target group?
Well, we don't have to
because the target group don't cost you any money
and the target group is going to be empty
because we have deleted the auto-scaling group
and we have deleted the low balancer.
That's it.
When the ASG is going to be gone,
then your EC2 instances, that's your ASG managers,
will also be gone, so, you'll be fully clean.
So, that's it.

We will remain within the free tier for this course.
I hope you liked it and I will see you in the next lecture.

Okay, so let's summarize the section
on the ELB and the ASG.
So first, we discussed the concept of high availability.
Scalability would be vertical and horizontal,
elasticity and agility in the Cloud.
It is very important for you to understand
to which concept corresponds to which features.
For example, high availability
means that you are having your applications
across multiple availability zone.
Vertical scaling means that you're increasing the size
of an instance.
And horizontal scaling,
is that you are increasing the number of instances.
Elasticity is the ability to scale up and down
based on demand.
And agility is a concept of the Cloud that is going
to be able to make you work faster,
because you can create and delete resources
very, very quickly.
Now, our load balancers, or ELB,
are allowing us to distribute traffic
across backend EC2 instances, and they can be spread out
across multiple availability zones.
We support health checks to make sure
that the backend EC2 instances are indeed healthy.
And we have four kinds of load bouncers.
The classic one is old and retired.
We have the application balancer for HTTP type of workload
at the layer seven.
Network load balancer for very high performance
and TCP level load balancing, layer four.
And finally, the gateway load balancer, layer three,
to route the network itself, and make it go by, for example,
through some virtual appliances.
We have auto scaling groups that allow us
to implement elasticity for our application,
therefore spreading our load across multiple AZ
and scaling accordingly.
So we scale these EC2 instances
based on the demand on your system,
and we can replace unhealthy instances.
There's a tight integration between the ASG and the ELB.
So this is why they are a great combination,
and together, we achieve high availability, scalability,
elasticity, and agility in the Cloud.
All right, that's it.

I hope you liked it and I will see you in the next lecture.

#_____Section 8 _____

Amazon S3

Welcome to this section on Amazon S3.
So this section is very important
because Amazon S3 is one of the main building blocks of AWS
and the way it's advertised is
that it's infinitely scaling storage.
So as a matter of fact,
a lot of the web relies on Amazon S3.
For example, many websites use Amazon S3 as a backbone
and many AWS services will also use Amazon S3
for integrations as well.
So in this section, we'll have a step-by-step approach
to Amazon S3 to learn the main features.
So there are so many use cases
for Amazon S3 because at its core is S storage.
So imagine S3 is used for backup and storage.
It could be for your files
it could be for your discs, and so on.
For disaster recovery purposes.
For example, you will move your data to another region.
In case a region goes down,
then your data is backed up somewhere else.
It's for archival purposes.
So you can archive files in Amazon S3
and retrieve it at a later stage
for much, much cheaper.
For hybrid cloud storage.
So in case you have storage on premises,
but you won't expand it into the cloud,
you can use Amazon S3 for this.
To host applications, to host media such as video files,
images, and so on.
To have a data lake, so to store a lot of data
and to perform big data analytics,
for delivering software updates,
for hosting static websites, and so on.
And two use cases is that the Nasdaq stores
seven years of data into the S3 Glacier share service,
which is like the archival service of Amazon S3.
And Sysco runs analytics on its data
and gains business insights from Amazon S3.
So Amazon S3 stores files into buckets.
And buckets can be seen as top level directories.
And actually, the files

in these S3 buckets are called objects.
And these buckets, they are created in your account
and they must have a globally-unique name.
That means that the name must be unique
across all the regions you have it in your accounts,
but also all the accounts that exist out there on AWS.
So this is the only thing
that must be globally unique in AWS.
And the buckets are defined at the region level.
So even though the name of the bucket is unique
across all regions and all the accounts,
the buckets must be defined in a specific AWS regions.
So three looks like a global service,
but the buckets are actually created in a region,
and that is a common mistake for beginners.
So there is a naming convention for S3 buckets.
You don't remember it, but it's good to know.
So bucket names must have no uppercase, no underscore.
They must be between three and 63 characters long.
They must not be an IP.
They must start with a lowercase number or lowercase letter.
And then there are some prefix restrictions.
So as long as you use letters and numbers and hyphens,
you're good to go.
Okay, so now let's talk about objects.
So these objects, they're files
and they have what's called a key.
And an Amazon S3 object key is the full path of your file.
So if you look at my bucket,
this is the top level directory.
Then the key of my file at TXT is my file dot TXT.
But in case you want to nest it in what we call folders,
then the key is going to be the full path.
So my folder one slash another folder slash my file dot TXT.
Therefore, the key is composed
of a prefix and then an object name.
So we can, for example, decompose the path from before
into the prefix, which is my folder one and another folder,
and the object name, which is my file dot TXT.
So Amazon S3 does not have a concept of directories per se,
although when you look in the console, the UI,
you will think otherwise
and you will actually create directories.
But anything and everything in Amazon S3 is actually a key.
And keys are just very, very long names
that contain slashes and keys are made
of a prefix and an object name.
Okay, so the objects then, what are they?
Well, their values are the content of the body.
So you can upload a file,

you can upload whatever you want into Amazon history.
So the max object size is five terabytes.
So this is 5,000 gigabytes.
And if you upload a file that is very big
and if that file is greater than five gigabytes,
so a big file, okay, then you must use
the multi-part upload to upload that file
into several parts.
So if you have a file of five terabytes,
then you must upload at least 1,000 parts of five gigabytes.
Now, the object can also have metadata,
their list of key and value pairs, and that could be set
by the system or set by the user to indicate some elements
about the file, some metadata.
Their tags, for example, their Unicode key
and value pairs up to 10,
they're very useful for security and life cycles
and sometimes the object will have a version ID
if you have enabled versioning.
So that's it for an introduction to Amazon S3.
I'm sure you're curious about how that works,
so let's go in the console to get started.

So here I am in Amazon S3,
and I'm going to go ahead and create a bucket.
Now you will notice here that there's a region selected,
which is Europe, Stockholm, eu-north-1.
And this is because I have the region selection in here.
So choose the region you want to create your bucket in,
and we'll see that Amazon S3 still has a view
over all your buckets across all regions.
Now there's a bucket type that you may or may not see.
So if you're in some regions where it's available,
you will see general purpose or directory new.
And over time, it will be in more regions.
If you don't see it in your region, this is fine.
The option you should choose
if you see it is general purpose.
And if you don't see this option,
it will be automatically general purpose.
So don't touch anything
and don't feel alarmed if you don't see these options.
Directory buckets are for a specific type of use case
that is not covered at the exam,
so I will not be talking about it.
So if you see the screen, choose general purpose.
If you don't see the screen,
everything is fine, do not worry.
Okay, so you choose a region,
and next you need to choose a bucket name.

And so if you enter a bucket name that is already taken,
for example, tests,
and you try all the way down to create your bucket,
you're going to get an error saying
that the bucket with the same name already exists.
So your bucket name must be unique across all regions
and all accounts ever created in AWS.
This is why I name my buckets
with something very, very personal.
For example, it could be Stephane and then demo, S3,
and I usually add version number, v5,
because I've been creating this video many, many times over
as the interface changes.
So, stephane-demo-s3-v5 should be available,
and they should have no errors.
But if someone already took it,
then I will need to change the name.
Okay, so next for object ownership.
Right now you have ACL disabled.
This is recommended.
This is a security setting.
Don't worry about it. We'll leave it as a default.
Now for blocking public access to this bucket,
again, we'll leave this enabled.
So we'll block all public access,
and we want to have maximum security in our bucket,
so only us can upload files to it.
Next for bucket versioning.
So we want to disable bucket versioning right now,
and we'll see later on how to enable it.
No tags are needed.
And for default encryption,
I'm going to use server-side encryption
with Amazon S3 managed key.
So all my objects are going to be encrypted,
and I will choose the first option.
We'll talk about encryption later on.
And bucket key, I will enable it.
So we'll leave, as you can see, all the settings as default.
The only thing we have set, really, is the bucket name.
So I'll go ahead and create my bucket.
And now it has been successfully created.
And you will see here in this UI all your buckets.
If you have directory enabled,
you will see also directory buckets,
right now I have none.
But your general purpose buckets are here.
Right now, you should see one bucket
if you just created this course.
For me, I have 33

because I've been using my account quite a lot.
And this will deploy buckets for all AWS regions,
not just the one you're in right now, but all regions.
As you can see, I have Ireland, London.
I scroll down, I get us-east-1, Frankfurt, and so on.
So all your buckets are going to be displayed here,
and you can do a little search.
For example, stephane-demo,
and here is my bucket.
So I'm going to click on it and have a look at it inside.
And now in my bucket,
I would like to start uploading objects
because currently you have zero objects.
So let's click on upload, and then we can add files.
And navigate into your code, go into the S3 folder,
and then you will find a coffee.jpg file.
So choose this coffee.jpg file.
As you can see, it's an image jpg,
it has 100 kilobytes in size.
And then the destination is S3 stephane demo,
which is my bucket.
Okay, so let's upload this file.
We're done.
So I can close this on the right-hand side.
And now back into my S3 bucket.
I can see the coffee.jpg file is under my objects.
So I can do is now click on it
and have more details around that file.
So now that we are in the object page,
we can have a look at a bunch of overviews.
So a bunch of properties where it's been uploaded, the size,
the type, and there's an object URL here.
We'll be playing it in a moment.
So how do we do this?
So now we want to open this object
and see if we can open it.
We can view it because we have uploaded it
onto our Amazon S3 buckets.
Therefore, I'm going to click on open.
And if I do click on open, as you can see,
I can see my coffee.jpg file.
So this is the one I have uploaded,
and it is on the internet.
Awesome, right?
But if I go back to my overview
and click on this object url over here.
So I copy it, I paste it, and I enter it.
As you can see, I get an access denied.
And this access denied
tells me that I cannot access my object

using what's called the public URL.
So you can see here, this public URL is not working,
but this URL is working.
So what's the difference?
Well, this URL right here,
if you have a look at it, the beginning is exactly the same,
but then the rest is a very, very complicated and long URL,
because it's called an S3 pre-signed URL.
Why?
Well, because this URL contains actually a signature
that verifies that I am the one making the request,
and therefore it has my credentials in it.
And so because my credentials are encoded in this URL,
then Amazon S3 says,
"Well, Stephane is allowed to view his own object,
therefore I will display it."
So this public URL does not work,
but this pre-signed URL with my own credentials works,
and of course, this URL is only for me.
So we'll see how to make that object public later on,
so that the public URL will function as well.
So let's go back into our bucket, the stephane-demo-s3.
And I have one object, but I can create a folder.
And this folder name may be called images.
So we scroll down and create this folder.
So now I have the images folder in my bucket.
I can click on it,
and within it, I can upload again a file.
And this time, I will upload the beach.jpg file into,
as you can see, the destination is my images folder
within my S3 buckets.
So let's upload this.
Close this.
As we can see now, we have the beach.jpg object
within the images folder.
And if I go one level up, we can see the folder here.
So this looks just like,
you know, the cloud storage service you used to know
such as Google Drive or Dropbox,
or whatever you want.
Here, we have something very similar
in terms of the user experience on Amazon S3.
So, of course, I can go to images
and I can delete this folder entirely.
So this will delete everything within the folder.
And to delete things,
I just type permanently delete into the text inputs,
delete my objects, and I'm good to go.
So that's it for this lecture.
We've seen how we can upload objects into Amazon S3.

We've seen how we can open them in two different ways,
creating folders, deleting folders, and so on.
So I hope you liked it,
and I will see you in the next lecture.

So now let's talk about Amazon S3-Security.
The first part is User-Based.
So as a user you can have IAM policies that you
and this IAM policy is going to authorize
which API calls should be allowed for a specific IAM user.
You can also have Resource-Based Security.
So this is new.
We can use what's called S3 Bucket policies
and there are bucket wide rules
that you can assign directly from the S3 console.
And this will allow, for example, a specific user to come in
or allow a user from another account,
this is called cross-account
to access your S3 Buckets.
This is also how we'll make our S3 Buckets public
as I will show you in a minute.
Next, you have the Object Access Control List or ACL,
they're finer grain security and they can be disabled.
And if you need to go
at the Bucket level, you can have Buckets ACL
which is way less common also can be disabled.
And the most common way now to do security
on an Amazon S3 Bucket is to use Bucket policies.
So in which situation can an IAM principle
can access an S3 object?
Well, if the IAM permissions allow it
or if the resource policies allows it
and that there is no explicit deny in the action,
then the IAM principle can access the S3 object
on the specified API call.
So we'll have a look at these use cases in a minute.
Finally, another way to do security on Amazon S3
is to encrypt the objects using encryption keys.
So what does S3 Bucket policy actually look like?
Because this is the focus of S3-Security for us.
So they are JSON based policies and they look like this.
So this is JSON documents and it's quite easy to read.
So the first thing is that you have a resource block
and the resource tells the policy
what buckets and object this policy applies on.
And in here we can see that this applies to every object
within the example Bucket,
this is what the star is for.
Next we have the effect.
So Allow or Deny, and what do we Allow or Deny?

Well, we Deny actions.
So we have a set of APIs we can either Allow or Deny
and in this example, the action we Allow is GetObject.
So this allows anyone thanks to the principle,
the principle presents the account or the user
to apply the policy to so principle is star.
So, here we allow anyone with a star to GetObject,
so to retrieve an object from my example Bucket
with a start, that means any object within it.
So therefore this S3 Bucket, is setting public reads
on all objects inside my Buckets.
So we can use an S3 Bucket policy
to grant public access to the Bucket
as the one shown on the right hand side
or to force objects to be encrypted at upload
or to grant access to another account.
So let's have a look at the situation.
So here is a Bucket Policy for Public Access.
So we have a user, it's on the worldwide web
it's a website visitor
and he wants to access files within our S3 Buckets.
So we'll attach an S3 Bucket policy
that allows public access.
This is the one you've seen in the previous slide.
And once this Bucket policy is attached to the S3 Bucket
then we can access any objects within it.
That's what we'll see in the hands-on.
But another way to do it
is that if you have a user within your account,
so it's an IAM user
and that user wants to access Amazon S3,
then we can assign IAM permissions
to that user through a policy.
And therefore because the policy allows access
to the S3 Buckets
then the user can access our S3 Buckets right now.
If we have an EC2 instance and want to give access
from the EC2 instance into the S3 Buckets,
we've seen that IAM users are not appropriate.
We need to use IAM roles instead.
So we create an EC2 instance role
with the correct IAM permissions
and that EC2 instance will be able to access
the Amazon S3 Buckets.
More advanced,
if we want to allow Cross-Account Access,
then we must use the Bucket Policy.
So we have an IAM user in another AWS account
and we create an S3 Bucket policy
that allows Cross-Account Access for that specific IAM user

and therefore the IAM user will be able to make API calls
into our S3 Buckets.
Other security settings you need to know about
is that there is the Bucket settings
for Block Public Access.
So this is what we set to own when we created the Buckets
and these settings were invented by AWS
as a extra layer of security to prevent company data leaks.
So even though you would set an S3 Bucket policy
that would make it public,
if these settings are enabled,
the Bucket will never be public.
So this is to prevent data leaks.
So if you know that your Bucket should never be public,
then leave these settings on
and you have this level of security
against people who would set the wrong S3 Bucket policy.
And if you know that none
of your S3 Buckets ever should be public,
then you can set this at the account level.
Okay, so that's it for S3-Security.
Now let's go in the hands-on to practice.

So, let's go ahead and make a bucket policy
so that we can access this coffee file from the public URL.
So to do so, let's go under the Permissions tabs.
And the first thing we have to do is to allow public access
from the bucket setting
because right now, everything is blocked.
So we edit this, and we're going to untick this,
and therefore, we will allow public access.
But again, this is something you would disable only
and only if you know you want to set a public bucket policy.
So this is dangerous action.
So we say yes because, of course,
if you sets data, real data of your company
on an S3 bucket and you make this public,
you have data leaks, and that can never be good.
So now, under permissions
or review, the access that objects can be public.
So that's the first step.
Next, we scroll down, and we look at Bucket policy.
So currently, we have none, and we wanna create one
so that we make our entire buckets public.
So the first thing you can do is look
at the policies example, and this is the documentation.
And it will show you a lot of use cases
on the right hand side that will show you
what's the appropriate and corresponding Bucket policy.
But for us, we're going to use the Policy Generator.

So here is the AWS Policy Generator,
and we're going to create an S3 bucket policy.
So let's select the right type.
We'll allow, and then the Principal is going to be a start
because we want to allow anyone
on the Amazon S3 Service to perform.
And because we read the objects on our bucket
we want to perform a GetObject.
So here it is. We want to allow, GetObject
and the Amazon Resource Name must be the bucket name
with a "/", and then with a "*".
So let's have a look first.
So back into our S3 buckets, we have the buckets ARN here,
the Amazon Resource Name here.
So we copy it, we paste it into the Amazon Resource Name
and this is not over.
We add a "/", and then we add a "*".
And the reason why we do this is
that this action the GetObject action right here applies
to objects within your buckets and therefore objects within
your buckets are after "/" and their "*"
to represent these objects.
So let's add this statements,
and then let's generate this policy.
And this policy is what we copy into here.
And this is a public S3 policy.
So that means that GetObjects are allowed from anyone
on any objects of this buckets.
Okay, that's good.
So let's save these changes.
And there is a space here. So let's remove this.
Perfect. Save these changes. Now that works.
And now, as you can see under permissions
of review the access is now set to public,
and we get a little warning that unless you want this
to be really accessible from the internet,
then don't do this.
Okay, so now it's publicly accessible.
We can see it really from everywhere.
So if we go back into our coffee.jpg file,
this is the public URL, okay? The full public URL.
And I'm going to refresh this page.
And now, as you can see, the coffee.jpg file appears
before my screen, and therefore, that object is now public.
And any objects uploaded
onto our Amazon S3 buckets is now going to
be accessible using the public URL.
Okay. So that's it for this lecture.
We've had an overview of Bucket policies,
and as well as a sneak peek into the Policy Generator.

I hope you liked it, and I will see you in the next lecture.

So now let's talk about how Amazon S3
can be used to create aesthetic websites.
So S3 can host static websites
and have them accessible on the internet
and the website URL will be depending
on the AWS region where you create this,
either this or that.
And they look very, very similar as you can tell.
But the only difference is that here, you have a dash
and here, you have a dot.
It doesn't really matter for you to remember this
but just so you know, here it is.
So we have an extra bucket
and it will contain files, maybe HTML files, maybe images
and then we're going to enable this to be compatible
with hosting a website.
So this is what it will look like with the corresponding URL
and then the user will access our S3 buckets.
But this will not work
if we don't have public reads enabled on our S3 buckets.
And this is why in the first place,
in the last lecture we learned about S3 bucket policies.
So if you have a 403 forbidden error
after enabling your S3 bucket for reads,
then that means that your bucket is not public.
Therefore, you must attach an S3 bucket policy
that allows it to be public.
So that's it for this short lecture,
now let's go into the hands on to practice this.

Okay, so let's enable our bucket
to be a website.
But first, I'm going to upload one more file here,
I'm going to upload my beach.jpg file into our buckets.
Okay, this is done, so we have two files into our buckets,
and let's go into Properties, scroll down,
and all the way down
you will find the static website hosting.
So click on Edit,
and here we will enable static website hosting.
We want to host a static website
and we need to specify an index document.
So I will say, index.html
and we will have to upload that file.
This is the default or homepage of the websites.
As you can see here, we also have a little warning sign
saying, hey, by the way

if you want to enable this as a website endpoint
you must make all your content publicly readable.
And we've done this in the previous lecture, so that's good.
Okay, so let's go and save this.
So this is saved, and go back into our objects.
And the one thing that's missing here
is that index, that HTML file.
So let's go to upload this,
we Add files and then I will click on index.html
and then upload, close it.
It's created, so now back into Properties,
let's scroll all the way down,
and you see now that's under static website hosting,
we have a bucket website endpoint.
So I copy this URL, paste it,
and I get, I love coffee.
Hello world!
And my coffee.jpg file.
So this working, so this is my index at HTML file,
and if I right click on this and open this image in new tab,
we have the public URL of our coffee.jpg.
So this is working,
and by the way if you wanted to change the coffee for beach,
well under beach.jpg we can see as well
this beach image right here.
So that means that everything is working,
our S3 bucket is enabled for static website hosting,
and thanks to the S3 bucket policy being public,
we can see all these files.
So we're good to go, I hope you like this lecture
and I will see you in the next lecture.
Autoscroll

## Course content

So, now let's talk about versioning
in Amazon S3.
Because we've seen how to create a website,
but it would be nice to be able to update it in a safe way.
So, you can version your files in Amazon S3,
and this is a setting you have to enable
at the bucket level.
So, we have my bucket and it's enabled with versioning.
So, whenever a user uploads a file,
it's going to create a version of that file
at the selected key.
And then should we re-upload the same key,
should we overwrite that file,
then instead it's going to create a version two,
and then a version three, and so on.
So, therefore, it is best practice to version your buckets.
Why?
Well, the first thing is that it protects

against unintended deletes.
So, for example, if you delete a file version,
actually you just add a delete marker
and therefore you can restore versions
that were previously there instead.
You can also easily roll back to a previous version.
So, if you want to go back to what happened two days ago,
you can take a file and roll it back.
So, there are some notes you need to be aware of.
First of all, that any file that is not versioned
prior to enabling versioning will have the version null.
And also that if you suspend versioning,
it does not delete the previous version,
so, it's a safe operation.
Okay, so now let's go into the console
and have a look at how we can use versioning.

So now let's play with S3 versioning.
And so first you need to go into the properties
and then we have a bucket versioning setting.
We're going to edit this and we're going to enable it.
And this is to enable bucket versioning.
And therefore any files we override now
is just going to add versions into our buckets.
So this is good.
Let's go into our objects
and say we want to update our website.
So let's go back, find the website URL.
It is right here.
Okay, so we have "I love coffee"
but let's say we want to write, "I really love coffee"
so therefore let's go back into here, our file.
And I'm going to edit it
and say I really love coffee.
I've saved it.
And now I upload this file again.
So I'm going to add a file and it will be my index.html.
And now we have updated content in that file.
So if I upload it.
It's successful.
So now it's been overwritten.
If I go into my first webpage and refresh this,
I get "I REALLY love coffee."
But what did happen in the back?
Well, if we go here
and we have here this toggle "Show versions"
we're going to show the actual version ID with the files.
And so we can notice a few things.
Number one, the files that we had uploaded before

such as the beach.jpg and the coffee.jpg
have a null version id.
That's because they were uploaded
before we had enabled versioning.
But this file index.html as you can see has two versions.
One has version ID null, which is the file we had uploaded
before enabling versioning.
But the file we uploaded just right now has a version ID.
And therefore, by updating this file
and uploading it into our S3 bucket,
we have created a new version ID.
So this is something you can only see if you
enable this toggle.
So now thanks to versioning
what we can do is we can actually roll-back our page.
So we have, "I REALLY love coffee"
but we wanna go back to "I love coffee."
So how do we do this while we click on
this specific version ID
Okay, so make sure that "Show versions" is enabled
and then we're going to delete.
And here we have to delete a specific version ID
of our object.
And therefore when we delete a specific version ID
of our object, it's called a permanent delete.
So it's a destructive operation, it cannot be undone.
And so if we're sure, we type permanently delete
in this text box and click on delete objects.
And now if you go back, as we can see,
we are back with the previous date of our bucket.
And therefore if I refresh this page,
I just get "I love coffee."
But what if we disable "Show versions"?
So now we just have our objects
and I'm going to take this coffee.jpg file,
and I will delete it itself.
So this time we don't actually delete
the underlying version ID,
we delete by adding a delete marker.
And so it doesn't actually delete the underlying object
as we'll see.
Let's just type delete this time.
It's not permanently delete, it's just delete
and we delete the object.
Perfect.
So now if we have a look at it from within our bucket,
it looks like the coffee.jpg file is done.
But if we click on "Show versions"

we see that we have a delete marker on our coffee.jpg file
with this version ID.
And the real, the previous, at least, coffee.jpg file
is still in our buckets
but it's being overwritten at least right now
from a version perspective by a delete marker.
So back into our webpage, if I refresh this page
and I refresh it by forcing a refresh,
by doing a Command + Shift + R to force a refresh
then we see that the image is not available.
So how do we get back this image?
And if I just try to, for example,
open this image in the new tab, it doesn't work.
I get a "404 Not Found."
So instead to get back the previous objects
I can click on this delete marker
and I'm going to delete the delete marker.
So I will permanently delete this delete marker.
And the effect of that is that it will restore
the previous version of my object
which is this one from before.
So back into my webpage, if I refresh it now
we can see that the coffee.jpg file is there.
So you can play around with versioning
you can add as many file version as you want.
You can start deleting them and view what happens.
But I hope you like this lecture
and I will see you in the next lecture.
Autoscroll

Now let's talk about Amazon S3 Replication,
and there are two flavors of it.
So CRR is for cross-region replication
and SRR is for same-region replication.
The idea is that we have an S3 Bucket in one region
and a target S3 Bucket in another region,
and we want to set up asynchronous replication
between these two buckets.
So, to do so, we first must enable Versioning
in the source and the destination buckets.
And if we do CRR, so cross-region replication,
the two regions must be different.
If we do SRR, the two regions are the same.
Now, it's possible for you to have these buckets
in different AWS accounts
and copying happens asynchronously.
So the replication mechanism happens behind the scenes,
in the background.

And to make replication work,
you must give proper IAM permissions
to the S3 service so that it has the permission
to read and write from specified buckets.
So the use cases for replication are manyfold.
The first one is that if you use cross-region replication,
this can be helpful for compliance
or for providing lower latency access to your data
because it's in another region,
or to replicate data across accounts.
For SRR, or same-region replication,
this can be very helpful to aggregate logs
across multiple S3 Buckets
or to perform a live replication
between a production and test accounts,
so, you have your own test environment.
Okay, so that's it about replication.
I will see you in the next lecture for some practice.

So let's practice replication
on Amazon S3.
For this, we're going to create a new bucket.
I'll call it S3 Stephane bucket origin V2,
and I will set it in one region that I want,
for example, EU west one.
This will be my origin buckets
and then data will be replicated
from this bucket to another bucket.
So the thing I need to do of course,
is to enable versioning
because replication only works
if versioning is enabled.
So I will create this bucket,
and then open this bucket in a new tab
and I will create a second bucket
and this will be my target bucket.
So I will do S3 Stephane bucket replica V2.
And this time, the region can be either the same,
for example, if you wanted to do same region replication
or something completely different,
for example if you wanted the US,
you could do US east one
to replicate from Europe to the US.
Okay, so let's scroll down
and let's again, enable bucket versioning
on the target buckets.
And we're good to go.
So now we have our primary buckets
and our secondary buckets.
What I'm going to do is that in the origin bucket,

I'm going to upload a file
so I will add a file of my beach,
for example, beach.jpeg.
Okay, this is done and we can close this.
So now this has one file,
but of course, this does not get replicated yet
because we haven't set up replication yet
so let's go ahead and do this.
So on the origin bucket,
what I need to do is to go under management,
scroll down and there are replication rules,
currently, zero.
So let's go ahead and create our first replication rule.
So I'll call this one demo replication rule.
Okay, perfect.
We'll set it as enabled.
For the source bucket,
we'll leave it as is
and in terms of rule scope,
we'll apply it to all objects in the buckets.
Now for the destination,
we can specify a bucket in this account
or in other accounts,
and we'll choose one in this account
and the bucket name is my target bucket.
So I need to actually enter the name.
So I'll take this bucket right here, copy and paste it.
Okay, and as you can see,
the destination region was identified
as being US east one,
so therefore, this is a cross region replication.
Okay, now for IAM role,
we need to actually go
and create a new role for this.
So there's the option.
And then we can have a look at some settings,
but for now, it doesn't really matter for us.
Let's just save this.
So we get a prompt right here,
which says, do you want to replicate existing objects?
So it turns out that when you do enable replication,
it will only replicate objects
from the moment you set it,
so for newer uploads.
So if you wanted to replicate the previous objects
from the source of the destination bucket,
you could use something called a batch operation,
an S3 batch operation to do so
and you would need to say yes, replicate existing objects,
but this is separate from the replication feature itself.

Therefore, I'm going to say no,
do not replicate existing objects
and we're good to go.
So now let's have a look.
So we have this management
with a replication rule that is ready.
And now what I'm going to do is check in my replica bucket.
Of course, if I refresh now,
we see that the objects haven't been replicated.
So I'm going to do is now upload a new file,
for example, upload the coffee.jpeg file,
upload it.
We're done.
So here is the coffee.jpeg file.
Let's show the version.
So this is coffee.jpeg and the version is GBK.
Okay, perfect.
Now, if we go in my target bucket and refresh this,
it's gonna take maybe five seconds.
And this took about 10 seconds on the first replication,
but we can see that my coffee.jpeg
has been added into my replica bucket.
And if I show the versions,
we can see the version ID
of my coffee.jpeg is the exact same of the origin bucket.
So the versions IDs are replicated, which is great.
And if I wanted to port the beach.jpeg,
I would need to upload a new version of that file,
so let's upload beach.jpeg again.
Now this has been uploaded.
We can look at the version.
So there is a new version right here
of DK2, of my beach.jpeg file.
And now if I go here and refresh,
and this took a bit of time, but as you can see,
you can find the DK2 version of that file.
So that's it for S3 bucket replication.
I hope you liked it
and I will see you in the next lecture.

Okay, so let's discuss
different storage classes we have for Amazon S three.
The first one is Amazon S three Standard-General Purpose.
Then we have Amazon S three-Infrequent Access.
Then we have Amazon S three One Zone-Infrequent Access.
Then we have Glacier Instant Retrieval,
Glacier Flexible Retrieval, Glacier Deep Archive,
and then finally, the Amazon S three Intelligent Tiering.
So we'll learn about all these classes in depth

in this lecture, but you have to know them for the exam.
Then when you create an object in Amazon S three,
you can choose its class,
you can also modify its storage class manually,
or as we'll see as well,
you can use Amazon S3 Lifecycle configurations
to move objects automatically
between all these storage classes.
So first, before we go into the classes,
let's define the concept of durability and availability.
So durability represents how many times an object
is going to be lost by Amazon S three.
And so Amazon S three has a very high durability.
It's called 11 nines.
So nine nine point and then nine times nine percent.
And that means that on average,
if you store 10 million objects on Amazon S three,
you can expect to lose a single object
once every 10,000 years.
So it's quite durable.
And the durability is the same
for all storage classes in Amazon S three.
Availability represents how readily a service is.
And so this depends on the storage class.
For example, S three Standard has a 99.99% availability.
That means that about 53 minutes a year,
the service is not going to be available.
That means that you'll get some errors
when you deal with the service.
So you need to take that into account
when you develop your applications.
Okay.
So S3 standard has 99.99 availability.
It's going to be used for frequently accessed data.
This is the kind of storage you use by default,
and it has low latency and high throughputs.
It can sustain two concurrent facility failures
on the side of AWS and the use cases for it
is going to be big data analytics,
mobile and gaming application,
as well as content distribution.
Next, we have S three infrequent access.
So this is data that is going to be as the name indicates,
less frequently accessed,
but requires rapid access when needed.
It's going to be lower cost than S three Standard,
but you will have a cost on retrieval.
So the S three Standard-IA is 99.9% availability,
so a bit less available.
And the use case for it is going to be Disaster Recovery

and backups.
And Amazon S three One Zone-Infrequent access, One Zone-IA.
ESC has high durability, okay, within a single AZ only,
and the data is going to be lost
if the AZ is somewhat destroyed.
As well as durability, it's even lower.
So it's 99.5% availability.
And so the use cases of S three One Zone-IA
is to store secondary copy of backups
of maybe on-premises data, or data you can recreate.
Next we have the Glacier Storage Classes.
So Glacier is, as the name it gets very cold,
so it's low cost object storage
meant for archiving and backup.
And the pricing is that you're going to pay for the storage
plus pay for a retrieval cost.
In your three classes of storage within Glacier,
you have the Amazon S three Glacier Instant Retrieval.
And this gives you milliseconds retrieval
which is great for example,
for data that's accessed once a quarter,
and the minimum store duration is 90 days.
So this is backup,
but you need to access it within milliseconds.
Then we have the Glacier Flexible Retrieval.
It used to be called Amazon S three Glacier
but then they renamed things as they added more tier.
So the Amazon Glacier Flexible Retrieval
has three flexibility.
So you have expedited where you get the data back
between one and five minutes.
You have standard to get the data back
between three to five hours, or bulk, which is free,
where you get data back between five to 12 hours.
And the minimum storage duration as well is 90 days.
So here, instance means you retrieve data instantly
and flexible means that you're willing
to wait up to for example, 12 hours to retrieve your data.
And then we have Glacier Deep Archive
which is meant for long term storage.
So we have two tiers of retrieval as well.
We have Standard of 12 hours and Bulk of 48 hours.
So you may be ready to wait a lot of time
to retrieve data,
but it's going to give you the lowest cost,
and as well, the minimum storage duration is 180 days.
So as you know, that's a lot of storage classes
and there's one last called S three Intelligent- Tiering,
which is going to allow you to move objects
between excess tiers based on usage patterns.

And for this, you're going to incur a small
monthly monitoring fee, and auto tiering fee.
And there are no retrieval charges
in S three Intelligent- Tiering.
So there is the frequent access tier that's automatic
the default tier.
Then we have the Infrequent Access tier
for objects not accessed for example, for 30 days.
Then you have the Archive Instant Access tier,
automatic as well for objects not accessed over 90 days.
And then the Archive Access tier that's optional.
And you can configure it from 90 days to 700 plus days.
And then you have the Deep Archive Access tier
also optional, that you can configure for objects
that haven't been accessed between 180 days
to 700 plus days.
Okay.
So S three Intelligent-Tiering is really
to allow you to just sit back and relax
while S three moves objects for you.
So if you compare all the storage classes
you don't need to remember these numbers,
but it's just for you to make sense of what they are.
So you get durability of 11 nine's everywhere.
Then as availability goes down, the less zones you have,
of course.
It just shows you like for example
the minimum storage duration chart and so on.
So take some time to look at this diagram on your own.
You should understand it,
but you should not remember it for sure.
So if we look at some pricing, for example
in the us-east-one,
so this is the kind of pricing you would have
for all the storage classes.
And again, you're not supposed to remember everything.
But it's good for you to have a look at it
on your own time, just to make sure you understand.
Because if you understand what the classes name are,
then you should be able to make sense of these classes.
Okay?
So that's it for the lecture.
I hope you liked it.
And I will see you in the next lecture.

So let's create a new bucket industry
and call it "s3-storage-classes-demos-2022."
Okay. Then I create into any kinda region,
and I will go ahead and just create this bucket.

So back in my bucket, I can go ahead
and upload a object, and click on add files.
I will choose my coffee.JPEG.
And let's have a look at the options.
So we can look at the properties
of that object,
and under storage class, I get the wide range
of storage class that are for AWS objects.
So we have S3 standard, okay,
and we get the design four column.
How many AZ's we have,
as well as some other (indistinct)
the minimum storage duration,
minimum billable object size,
and monitoring and auto-tiering fees.
So let's have a look at all of them.
So we have standard, which is the basic ones by default.
Then we get intelligent tiering,
in case we don't know our data patterns,
and therefore we want AWS
to perform the data tiering for us.
Standard-IA, if we want data to be infrequently accessed,
but still with low latency.
One-Zone-IA,
which is that you can recreate this data,
and it's going to be stored in one AZ only.
And therefore you can
run the risk of losing the object,
if the AZ is destroyed.
Then we have three glacier levels.
So we have Glacier Instant Retrieval,
Glacier Flexible Retrieval,
or a Glacier Deep Archive,
and it tells you exactly
what are the conditions in here.
And finally, Reduced Redundancy,
which is a deprecated type of storage tier,
and therefore I did not describe it in the course.
So what if we go with standard IA
for example, and create an object there.
So we're going there,
and then we're going to say upload.
Back in our bucket.
So now this object has the storage class, Standard-IA,
as it is shown here.
But what I can do is
that I can also change the storage class
if I wanted to.
So I can go into properties and scroll down,
and we can actually edit the storage class

to do something different.
So we can move it for example,
to One-Zone-IA,
in which case this object is going
to be stored in one zone only.
So let's save these changes.
And now my object has successfully been edited
and therefore the object class has changed.
So if we scroll down, now we are in One-Zone-IA.
And again, you could edit it,
and go, for example, for Glacier-Instant-Retrieval.
And now it's going to be archived,
or you can go for Intelligent-Tiering,
and it could be automatically set
to the right tier based on our patterns, and so on.
So you can see there's a lot of power using storage classes.
And finally,
I want to show you how we can automate moving these objects
between the different storage classes.
So let's go back
into our buckets, and there under management,
you can create lifecycle rules.
And you can create a rule,
and we'll call this one "DemoRule."
And then you're going to say,
"hey," apply to all objects in the buckets.
Yeah, sure.
And then we can say, okay
move current versions between storage classes.
And you're saying, hey, you go
to Standard-IA after, for example, 30 days.
And then you go to Intelligent-Tiering after 60 days.
And then you would go to Glacier-Flexible-Retrieval
after 180 days,
and so on.
So you get some transitions.
And in here you can review
all the transitions you have done.
So it is possible
for you to automate moving objects between tiers.
Okay. So that's it,
we've seen everything we need to know about storage classes.
I hope you liked it.
And I will see you in the next lecture.

So you might have one question
on S3 encryption at the exam,
so here is a high level review of what that means.
The first one is server-side encryption,

so it is by default whenever you create a bucket
or whenever you upload an object, it will be encrypted.
What is server-side encryption?
Well, the user uploads an object into Amazon S3,
and then that object when it arrives in the bucket
is going to be encrypted by Amazon S3 for security purposes.
The idea is that the server is doing the encryption,
and therefore we call this server-side encryption.
On the opposite, we have client-side encryption.
This is when the user will actually take the file,
will encrypt it before uploading it,
so the lock is done by the user,
and then put it in the bucket.
And that's called client-side encryption.
And both models exist in AWS,
but by default you should know
that server-side encryption is always on.
All right, that's it.
I hope you liked it and I will see you in the next lecture.

So, just a quick lecture
on IAM Access Analyzer for Amazon S3,
which is something that can come up
in the exam in one question.
So, what is this?
Well, this is a monitoring feature
for your Amazon S3 buckets to ensure
that only the intended people have access
to your S3 buckets.
So, how does that work?
Well, it's going to analyze your Bucket Policies,
your S3 ACLs, your S3 Access Point Policies,
and so on, and is going to surface to you
which buckets are going to be publicly accessible,
which buckets have been shared
with other AWS accounts and so on.
And the idea is that you can review this and say, okay,
this is normal, this is expected, or this looks a bit
as a security issue because I did not intend
to share this bucket with these people,
and therefore, you can take action.
And this is powered by IAM Access Analyzer,
which allows you to find out resources in your account
that are shared with other entities.
So, I hope that makes sense.
I hope you like this lecture
and I will see you in the next lecture.

As always, it's good to see

what is the shared responsibility model for Amazon S3.
So AWS is going to be responsible
for all the infrastructure.
That includes all the things specific to S3
so that their ability, availability,
the fact that it can actually sustain
current losses of two facilities.
As well as, I will see their own internal configuration
and vulnerability analysis.
And their own compliance validation
internally within their infrastructure.
And you as a user of Amazon S3,
you're supposed to set up correctly S3 Versioning
to make sure you set up the right S3 Bucket Policy
so that the data is protected within your buckets.
You need to make sure that if you want verification
you set it up yourself.
Logging and monitoring is optional
so you have to enable it yourself.
Making sure that you are using the most optimal
cost storage cloud that is going to be
the most cost friendly is also your responsibility.
And finally, if you wanted to encrypt your data
onto your Amazon S3 bucket,
that is up to you as well, okay?
So here we can really see what is the difference
between the responsibility of
AWS and yourself in this Amazon S3 service.
All right, that's it.
I will see you in the next lecture.

So now let's talk about the AWS Snow Family.
So it represents a highly-secure portable devices
that has two use cases within AWS.
Either it's used to collect and process data at the edge
or to migrate data in and out of AWS.
So we have two use cases, data migration,
and for this we have three different types of devices
within the Snow Family.
We have the Snowcone, the Snowball Edge,
and the Snowmobile.
And for the second use case, edge computing,
we have the Snowcone and the Snowball Edge.
So we'll first tackle the data migration subject
and then the edge computing.
So why do we want to do data migration
with the AWS Snow Family?
Well, if we look at the time it takes
to transfer a lot of data over the network,
it can take a lot of time.

So for example,
if we want to transfer a hundred terabytes
over a one gigabits per second network line,
it will take us 12 days to achieve it.
Okay?
And so obviously if we do a petabyte,
it will take forever and so on.
So as we can see, sometimes we just wanna do that
to get to AWS fast.
And the challenges that sometimes,
on top of having a small network transfer,
you have limited connectivity,
limited bantwidth, bandwidth, sorry.
Transferring data over the network
may cost you some money, okay?
It's not free to use a network.
It could be that also the bandwidth is shared.
For example, if you download a video from AWS
and you download 10 terabytes of data, you know,
maybe you're going to block your entire office
because you're maximizing the bandwidth within your office.
And then maybe the connection is not stable enough,
so you have to retry, and so on.
So all these reasons make a case for Snow Family.
So the Snow Family are offline devices that allow you
to perform data migrations.
So AWS will send you an actual physical device, you know,
by the post office, and then you load your data onto it
and then you send it back to AWS.
So the rule of thumb is that if it takes more than a week
to transfer data over the network,
then you should use a Snowball device, for example.
So to really explain how that works, let's take an example.
If you wanted to directly upload a file into Amazon S3,
we have the client sends the data into Amazon S3.
Very easy, right?
But with the Snow Family, for example,
with a Snowball device,
the client request a Snowball device,
we receive it via the post, okay?
AWS will deliver the device to us,
we load the data directly onto the devices locally,
and then we ship back the device to AWS
into an AWS facility.
Then they will take the device
and they will plug it into their own infrastructure
and then the data will be imported or exported,
based on what you want to do,
to an Amazon S3 bucket and you're good to go.
So really, it is a way to transfer data to AWS,

but through the physical route, not the network route.
Okay, so now, what sort of devices do we have?
We have the Snowball Edge.
And Snowball Edge is a huge box, as you can see,
and it is going to be used to move terabytes
or petabytes of data in and out of AWS.
It's going to be an alternative
to moving data over the network, that we've seen.
We're going to pay per data transfer job,
and the interface within the Snowball Edge
is going to provide a block storage
or Amazon S3-compatible object storage.
So we have two flavors for the Snowball Edge.
We have the Snowball Edge Storage Optimized,
which is going to give us 80 terabytes of HDD
or 210 terabytes of NVME capacity,
which works for block volume
or S3-compatible object storage.
And we have the Snowball Edge Compute Optimized of space
of 42 terabytes or 28 terabytes.
So really, if we want to have more storage,
obviously we want to get the flavor
that is called Snowball Edge Storage Optimized.
So the use case for a Snowball Edge for data transfers
so far is to do a large data cloud migration
to decommission a data center
or maybe to do disaster recovery
by backing up your data into AWS.
Next, we have the Snowcone.
So the Snowcone is a very small portable device
and it's rugged, it's secure,
it can withstand a harsh environment,
and it's meant for environments
where you have little amount of data.
So it's light, it's like 2.1 kilograms,
and you can put it on a drone if you wanted to.
And it's going to be used
for edge computing, storage, and data transfer.
You have two flavors of it.
You have the Snowcone,
which comes with eight terabytes of HDD storage,
and the Snowcone SSD of 14 terabytes of SSD storage
if you need a faster disk.
You will use the Snowcone
where the Snowball does not fit, for example,
in a space-constrained environment,
and you must provide your own batteries and cables.
Now, to send back the data to AWS, you have two options.
So either you send the data back offline by shipping it,
or you can connect this device

after it has captured some data into a data center,
for example, whatever that has an internet connection,
and then use the AWS DataSync service
to send the data back to AWS.
And then Snowmobile is an actual truck.
So when they announced it,
they actually took a truck on stage to show
that it was an actual truck that is going to transfer data.
And so with the Snowmobile,
you can transfer exabytes of data.
So one exabyte is 1000 petabytes is 1 million terabytes,
and each Snowmobile will have 100 petabytes of capacity.
So if you wanted to reach one exabyte of data,
you need to order 10 Snowmobiles.
It's high security, it's temperature controlled,
there's GPS, 24/7 video surveillance,
so it's quite a secure way to transfer your data.
And it's a better use case than Snowball
if you start transferring more than 10 petabytes of data.
So as a summary, for data migration, we have three options.
We have Snowcone, Snowball Edge, Snowmobile,
and each come with different storage capacities.
So 8 terabytes up to 210 terabytes,
and 100 petabytes.
The migration size that's recommended by AWS
is a Snowcone is up to 24 terabytes.
For Snowball Edge, it's up to petabytes,
and it's offline because you have to send it back to AWS.
And for Snowmobile,
the use case is up to exabytes of data.
DataSync agent is pre-installed on a Snowcone
because you can plug it to a network
and have DataSync send data over the network as well.
Okay, so how do we use a Snow Family device?
Well, you request a device from the console for delivery,
and we'll see this in the hands-on.
Then we install the Snowball client,
or we use AWS OpsHub
that we'll see in this lecture on your servers.
Then we connect the Snowball to the servers
and start copying the files in the clients.
Then we ship back the device when we're ready.
It will go straight to the right AWS facility,
thanks to an E Ink marker,
and the data will be loaded onto an S3 bucket,
and then the Snowball will be completely wiped according
to the highest security measures.
So that's for the data migration,
and that was originally one and the only use case
for Snowball devices.

But the second use case now
for the Snow Family is called edge computing.
And so edge computing is when you process data
while it's being created at an edge location.
So what is an edge location?
Well, an edge location is anything
that really doesn't have internet
or that is far away from a cloud.
So for example, if you have a truck on the road
or if you have a ship on the sea,
or a mining station underground,
all these things can be called edge locations
because they can produce data,
but they may not necessarily have internet connectivity.
So either limited connectivity or no internet access
or no access to computing power.
And so you may still want
to run computation, data processing at these locations.
And for this, we need edge computing.
And so to do edge computing,
we can order a Snowball Edge device or a Snowcone
and have it embedded into these edge locations
and start doing edge computing.
So the use cases of edge computing is to preprocess data,
do machine learning at the edge,
so without it going back to the cloud,
transcode media streams in advance,
and eventually if need be,
if you need to transfer the data back into AWS,
you can ship back the device
for your Snowcone or your Snowball Edge.
So really you start processing the data very, very close
to where it's being created
and then you ship it back to AWS.
So for edge computing, what do we have?
We have the Snowcone and the Snowcone SSD,
which give you two CPUs, four gigabytes of memory,
wired or wireless access,
and you have USB-C to power the thing
or an optional battery.
Then we have a Snowball Edge devices,
which are compute optimized,
which gives you about 104 vCPUs, 416 gigabytes of RAM,
an optional GPU, if you wanted to do some video processing
or some machine learning,
and then you get either 28 terabytes of NVME
or 42 terabytes of HDD usable storage.
And storage clustering is available for up to 16 nodes
to increase the total size of your storage.
Now we have Snowball Edge Storage Optimized,

which give you about 40 vCPU, 80 gigabytes of RAM,
and 80 terabytes of storage,
or up to 104 vCPUs, 416 gigabytes of RAM,
or 210 terabytes of storage.
So all these Snowcone devices can run EC2 instance
and Lambda function.
And for Lambda function, it's lever
is something called the AWS IoT Greengrass service.
And for all of them,
because you can have them on-site for a long time,
actually rent them,
you have long-term deployment options, for example,
from one or three years discounted pricing.
Finally, for the Snow Family there is Snow, sorry, OpsHub.
So historically, when you were using these devices,
you needed a CLI, so command line interface tool,
to deal with them and it was very, very difficult.
And so AWS recognizes that,
and so they've created OpsHub,
which is a software that you install
on your computer or laptop,
so it's not something you use on the cloud,
it's something you have to download on your computer.
And then once it's connected,
it's going to give you a graphical interface to connect
to your Snow devices and configure them and use them,
which is very, very handy.
So this is how to do unlocking and configuring single
or clustered devices, transferring files,
launching and managing instances,
so EC2 instances running on Snow Family devices,
monitor device metrics,
and launch compatible AWS services on your devices,
for example, EC2 instances, AWS DataSync,
or a Network File System.
So that's it for the Snow Family.
I hope you liked it and I will see you in the next lecture.

So let's have a look at the AWS
Snow Family Service.
So from the console you can actually order
a Snow family device and you have to name your job,
for example, demo job.
So we need to choose a job type, so we can import data
into Amazon S3, which is the most common use case.
We can also export data from Amazon S3.
You can order a Snow device just for local compute
and storage only.
This gives you some kind of device

that server that can runs in a remote place
without being connected to the internet.
Or you can import virtual tapes
into the AWS storage gateway.
So we'll choose the first option, import into Amazon S3
and here we have several options for Snow devices.
So we have Snowcone, Snowcone SSD,
Snowball Edge Storage Optimized with 80 terabytes.
And then we have the compute optimized
and the compute optimized with GPUs.
So it's possible that AWS will add options over time
and may not rerecord this video.
So, that's fine.
I always cover what's at the exam, so, don't worry.
And so, as you can see, we have several options
and we seen the previous lecture,
the use case for each option
so, I won't go over them.
But this is a nice way to know where you're getting.
Okay, so let's say we go, for example
with a compute optimized Snowball Edge.
Then we need to choose a pricing option.
So we have on demand pricing, monthly pricing
or then commitment for one or three year period.
Then we have the storage types.
So it's an S3 data transfer
and then what AMI do we want to use on it?
So, this is a compute type of instance.
So we can actually load an AMI for example
this Amazon Linux 2 first, no family,
but you could create your own AMI to have your own
compute needs.
Then where do you want to load the data onto?
So, what S3 buckets are available to you?
I have this one right here
but you can create your own S3 buckets.
Next, we have several features and options we can use.
So we have IoT Greengrass for Snow to have IoT capability
on your Snow device, but it's not needed.
And then we can do remote device management
by using OpsHub or the Snowball Clients.
So I'll click on next
then what type of encryption do we want
on our Snowball device.
So, we have this encryption key right here where
you can create your own KMS key and encrypt it with that.
Then you need to grant access of course to Amazon S3 and SNS
to publish actions and to send data.
So you need to create a service role.
And then you need to add an address

so where you want to ship it, which country, which states
and so on, as well as the shipping speed
and the SNS notification for your job status changes.
And then you'll get the job summary.
So that's it, you've seen all the options for Snow devices
and the Snow family.
Remember, it's most commonly used to send data
into Amazon S3 and out of Amazon S3.
And this gives you remote compute capabilities.
We've seen also the different kind of devices
which are Snowcone, Snowball, and Snowmobile.
All right, that's it.
I hope you liked it and I will see you in the next lecture.

Let's talk about Snowball Edge pricing.
So you're going to pay for usage of the device,
as well as data transfer out of AWS.
So if you take data from AWS onto your Snowball Edge
and then you receive it, then you're going to pay for this.
But instead, if you put data onto your Snowball Edge
and then put that data onto Amazon S3,
it's going to be free.
$0 per gigabytes.
So putting data in Amazon S3 is free.
Next, for the pricing of the device itself,
so there's two levels.
You have on-demand,
where you have a one-time service fee per job,
so you're going to pay something,
but as part of the service fee,
you're going to get 10 days of usage
for the 80 terabytes Snowball Edge Storage Optimized
or 15 days for the 210 terabytes.
And then the shipping days are not included
towards the 10 or 15 days limit.
So shipping from it was to you or from you to AWS is free.
And then just days of usage is counted
per days after 10 or 15 days.
Next, you have committed upfront
where you pay in advance for monthly, one year,
or three years usage of the Snowball Edge device.
This is to do edge computing.
And you get up to 62% of discounted pricing
because you are committing upfront to a long-term usage.
So that's it.
From an exam perspective,
I think we need to remember
is that everything you have to pay for,
except data into Amazon S3,

which costs you $0 per gigabytes.
Okay, that's it for this lecture, I hope you liked it.
And I will see you in the next lecture.

So we've seen Amazon S3
as a standalone service, but it is possible for you
to use it in a hybrid cloud type of setting.
So AWS wants you to bridge
between your on-premises environment to AWS
and that's called a hybrid cloud.
So part of your infrastructure is going to be on-premises
and the rest is going to be on the cloud.
Why?
Well, maybe because sometimes you have created,
for example, your on-premises infrastructure first
and you're doing a migration but it could be long
so you can do a long cloud migration
or for security requirements or compliance requirements,
maybe it's your strategy to have part of it on the cloud
and part of it on-premises.
You have really different use cases
for trying to have two different ways of doing IT.
Some of it on-premises and some of it on the cloud.
And Amazon S3, it's a proprietary storage technology,
so this is not something like the EFS service
or NFS protocol we saw from before
that we can use directly on two servers on-premises.
For exposing S3 data on-premises,
you have to use something called a Storage Gateway.
So if you wanted to summarize the storage options on AWS,
the block storage would be EBS
or an EC2 instance store.
The file storage would be a network file system
so Amazon EFS,
and object storage would be Amazon S3 or Glacier.
And where does the Storage Gateway fit in all this?
Well, the Storage Gateway is going to be bridging
your on-premises data and cloud data in AWS.
So your hybrid storage will allow your on-premises systems
to seamlessly use the cloud
to extend the storage capability.
So this can be used for disaster recovery,
backup and restore or tiered storage.
There are different types of Storage Gateway.
There is a File Gateway, there is a Volume Gateway
and a Tape Gateway.
Now you don't need to know all these types
going into the exam, okay?
What you need to know is that the Storage Gateway

allows you to bridge whatever happens on-premises
directly into the AWS Cloud.
So under the scene,
the Storage Gateway will be using Amazon EBS,
Amazon S3, and Glacier behind the scenes, okay?
So from a certified cloud petitioner perspective,
the Storage Gateway is a way for you
to bridge your file systems and your storage on-premises
into the cloud to leverage best of both worlds, okay?
For if you were to do
the AWS Certified Solutions Architect Associate course,
you would need to understand Storage Gateway
a little bit better, but I will save you this right now
for the Certified Cloud Practitioner.
Okay, that's it.
I hope you liked it
and I will see you in the next lecture.

So now let's summarize
what we've learned about Amazon S3.
And I know that's a lot.
So first of all, we've learned
about the difference between Buckets and Objects.
So we know that Buckets must have a global unique name,
and is tied to a specific region
and Objects live within these Buckets.
For S3 security, we've seen that we can attach IAM policies
to users or to roles.
We've seen that we can also use S3 Bucket Policies.
For example, we granted public access
into an S3 Bucket, and we can set
up S3 Encryption to protect some files.
We can enable websites on an S3 bucket.
This is the idea to host a static website on Amazon S3.
You first need to make sure
that the Bucket, of course, is going to be public.
And then we can statically host some files.
We have S3 Versioning.
This is to have multiple versions for a file to prevent
against accidental deletes and to be able to roll
back to previous versions if we need to.
We've seen there are two kinds of S3 Replication.
You have the same-region replication
and the cross-region replication.
And for them to work, you must enable versioning beforehand.
We've seen different S3 storage classes.
We have standard, infrequent access,
one zone infrequent access, intelligent tiering
and we have three different classes for the Glacier

for (indistinct) purposes.
We've seen what the Snow Family is.
So there are physical devices that are used
to import data onto Amazon S3.
So you have like the Snowmobile, you have the Snowcone,
you have the Snowballs and so on.
And if you use a Snowcone or a Snowball edge device
then you can do edge computing on your data.
OpsHubs is a way to get a desktop application
to manage your Snow Family devices and add data onto them.
And finally, we've seen a way to extend on-premises storage
onto Amazon S3, that is using the AWS storage gateway.
So not so big lessons
but hopefully now you know all the specificity
of Amazon S3 that will allow you to answer all
the questions you need at the exam.
That's it.
I will see you in the next section.