**Module 3 introduction**

**Video Transcript (High availability, fault tolerance)**

I want to talk to you about high availability. Say you wanna grab a cup of coffee at the cafe, but today is not just a normal day in our city. Today, there is a parade coming to march down our street in celebration of all of the wonderful cloud migrations that have been successful. This parade is going to march right in front of our shop. This is great because who doesn't love to look at floats and balloons and have someone throw candy at them from the street? But this is also bad because while the parade is coming down the street, our customers who are driving to come get coffee will be diverted and won't be able to stop by the shop. This will drive sales down and also make customers unhappy.

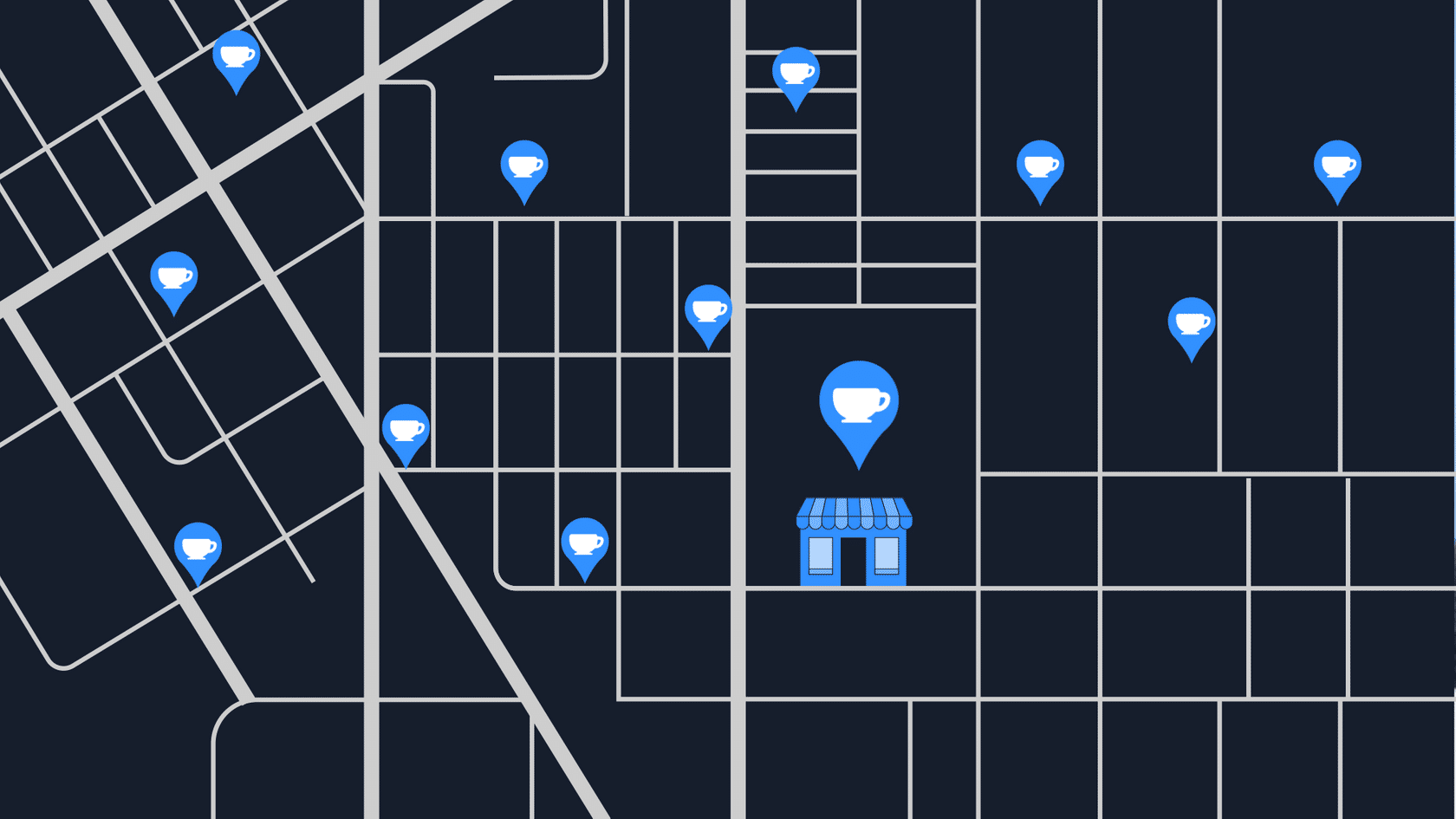
Luckily for us, we thought ahead. We thought long ago what would happen if for some reason our customers couldn't make it into the shop temporarily? Like if there was a parade or if there's a flood or some other event that blocks us from getting into the shop? Well, no matter the reason, we need to be highly available for our customers.

So, I'll let you in on a secret. This isn't our coffee shop's only location. The cafe is actually a chain and we have locations all around the city. That way, if a parade comes down the street, or if there was a power outage on our block, no worries. Customers can still get their coffee by visiting one of our shops just a few blocks away. We still make money and they get their coffee. All is well.

AWS has done a similar thing with how the AWS global infrastructure is set up. It's not good enough to have one giant data center where all of the resources are. If something were to happen to that data center, like a power outage or a natural disaster, everyone's applications would go down all at once. You need high availability and fault tolerance.

Turns out, it's not even good enough to have two giant data centers. Instead, AWS operates in all sorts of different areas around the world called Regions. We're going to talk about this in depth in upcoming videos. In the meantime, I'm gonna sit here and relax, because I know my business is highly available regardless of any parades blocking the street.

**Building a global footprint**



To understand the AWS global infrastructure, consider the coffee shop. If an event such as a parade, flood, or power outage impacts one location, customers can still get their coffee by visiting a different location only a few blocks away.

This is similar to how the AWS global infrastructure works.

# AWS global infrastructure

**Video transcript**

–

To understand the global infrastructure AWS, I wanna begin with your fundamental business need. You have an application that you have to run, or content you need stored, or data you need analyzed. Basically you have stuff that has to live and operate somewhere. Now historically, businesses had to run applications in their own data centers, 'cause they didn't have a choice. Once AWS became available, companies like yours could now run their applications in other data centers they didn't actually own.

But the conversation is so much more than that, 'cause I want you to understand a fundamental problem with any data center, doesn't matter who built it or who owns it. Events can happen that could cause you to lose connection to that building. If you run your own data center, you have to figure out how to answer the question of what will you do when disaster strikes your building. You could run a second data center, but real estate prices alone could stop you, much less all the duplicate expense of hardware, employees, electricity, heating and cooling, security. Most businesses simply end up just storing backups somewhere, and then hope for the disaster to never come. And hope is not a good business plan.

AWS answers the question of what happens when disaster strikes by building our data centers in large groups we call Regions, and here's how it's designed.

Throughout the globe, AWS builds Regions to be closest to where the business traffic demands. Locations like Paris, Tokyo, Sao Paulo, Dublin, Ohio. Inside each Region, we have multiple data centers that have all the compute, storage, and other services you need to run your applications. Each Region can be connected to each other Region through a high speed fiber network, controlled by AWS, a truly global operation from corner to corner if you need it to be. Now before we get into the architecture of how each Region is built, it's important to know that you, the business decision maker, gets to choose which Region you want to run out of. And each Region is isolated from every other Region in the sense that absolutely no data goes in or out of your environment in that Region without you explicitly granting permission for that data to be moved. This is a critical security conversation to have.

For example, you might have government compliance requirements that your financial information in Frankfurt cannot leave Germany. Well this is absolutely the way AWS operates outta the box. Any data stored in the Frankfurt Region never leaves the Frankfurt Region, or data in the London region never leaves London, or Sydney never leaves Sydney, unless you explicitly, with the right credentials and permissions, request the data be exported.

Regional data sovereignty is part of the critical design of AWS Regions. With data being subject to the local laws and statutes of the country where the Region lives. So with that understanding, that your data, your application, lives and runs in a Region, one of the first decisions you get to make is which Region do you pick? There's four business factors that go into choosing a Region.

Number one, compliance. Before any of the other factors, you must first look at your compliance requirements. Do you have a requirement your data must live in UK boundaries? Then you should choose the London Region, full stop. None of the rest of the options matter. Or let's say you must run inside Chinese borders. Well then, you should choose on of our Chinese Regions. Most businesses, though, are not governed by such strict regulations. So if you don't have a compliance or regulatory control that dictates your Region, then you can look at the other factors.

Number two, proximity. How close you are to your customer base is a major factor because speed of light, still the law of the universe. If most of your customers live in Singapore, consider running out of the Singapore Region. Now you certainly can run out of Virginia, but the time it takes for the information to be sent, or latency, between the US and Singapore is always going to be a factor. Now we may be developing quantum computing, but quantum networking, still some ways away. The time it takes light to travel around the world is always a consideration. Locating close to your customer base, usually the right call.

Number three, feature availability. Sometimes the closest Region may not have all the AWS features you want. Now here's one of the cool things about AWS. We are constantly innovating on behalf of our customers. Every year, AWS releases thousands of new features and products specifically to answer customer requests and needs. But sometimes those brand new services take a lot of new physical hardware that AWS has to build out to make the service operational. And sometimes that means we have to build the service out one Region at a time. So let's say your developers wanted to play with Amazon Braket, our new quantum computing platform. Well then, they have to run in the Regions that already have the hardware installed. Eventually, can we expect it to be in every Region? Well, yeah, that's a good expectation, but if you wanna use it today, then that might be your deciding factor.

Number four, pricing. Even when the hardware is equal one Region to the next, some locations are just more expensive to operate in, for example, Brazil. Now Brazil's tax structure is such that it costs AWS significantly more to operate the exact same services there than in many other countries. The exact same workload in Sao Paulo might be, let's say, 50% more expensive to run than out of Oregon in the United States. Price can be determined by many factors, so AWS has very transparent granular pricing that we'll continue to discuss in this class. But know that each Region has a different price sheet. So if budget is your primary concern, even if your customers live in Brazil, you might still wanna operate in a different country, again, if price is your primary motivation.

So four key factors to choose a Region: Compliance, proximity, feature availability, and pricing. When we come back, we'll look at the moving parts inside a Region.

**Selecting a Region**

When determining the right Region for your services, data, and applications, consider the following four business factors.

To learn more, select the **+**symbol next to each category.

**Compliance with data governance and legal requirements**

–

Depending on your company and location, you might need to run your data out of specific areas. For example, if your company requires all of its data to reside within the boundaries of the UK, you would choose the London Region.

Not all companies have location-specific data regulations, so you might need to focus more on the other three factors.

**Proximity to your customers**

–

Selecting a Region that is close to your customers will help you to get content to them faster. For example, your company is based in Washington, DC, and many of your customers live in Singapore. You might consider running your infrastructure in the Northern Virginia Region to be close to company headquarters, and run your applications from the Singapore Region.

**Available services within a Region**

–

Sometimes, the closest Region might not have all the features that you want to offer to customers. AWS is frequently innovating by creating new services and expanding on features within existing services. However, making new services available around the world sometimes requires AWS to build out physical hardware one Region at a time.

Suppose that your developers want to build an application that uses Amazon Braket (AWS quantum computing platform). As of this course, Amazon Braket is not yet available in every AWS Region around the world, so your developers would have to run it in one of the Regions that already offers it.

**Pricing**

–

Suppose that you are considering running applications in both the United States and Brazil. The way Brazil’s tax structure is set up, it might cost 50% more to run the same workload out of the São Paulo Region compared to the Oregon Region. You will learn in more detail that several factors determine pricing, but for now know that the cost of services can vary from Region to Region.

**Video transcript**

–

If a Region is where all of the pieces and parts of your application live, some of you might be thinking that we never actually solved the problem that we presented in the last video. Let me restate the problem. You don't want to run your application in a single building because a single building can fail for any number of unavoidable reasons.

You may be thinking, if my business needs to be disaster proof, then it can't run in just one location. Well, you're absolutely correct. AWS agrees with that statement and that's why our Regions are not one location. To begin with, AWS has data centers, lots of data centers, all around the world, and each Region is made up of multiple data centers.

AWS calls a single data center or a group of data centers, an Availability Zone or AZ. Each Availability Zone is one or more discrete data centers with redundant power, networking, and connectivity. When you launch an Amazon EC2 instance, it launches a virtual machine on a physical hardware that is installed in an Availability Zone. This means each AWS Region consists of multiple isolated and physically separate Availability Zones within a geographic Region.

But we don't build Availability Zones right next to each other because if a large scale incident were to occur, like a natural disaster, for example, you could lose connectivity to everything in that Availability Zone. The question what happens in case of a disaster matters and if you are familiar with disaster recovery planning, you might even have an idea of where we are going with this.

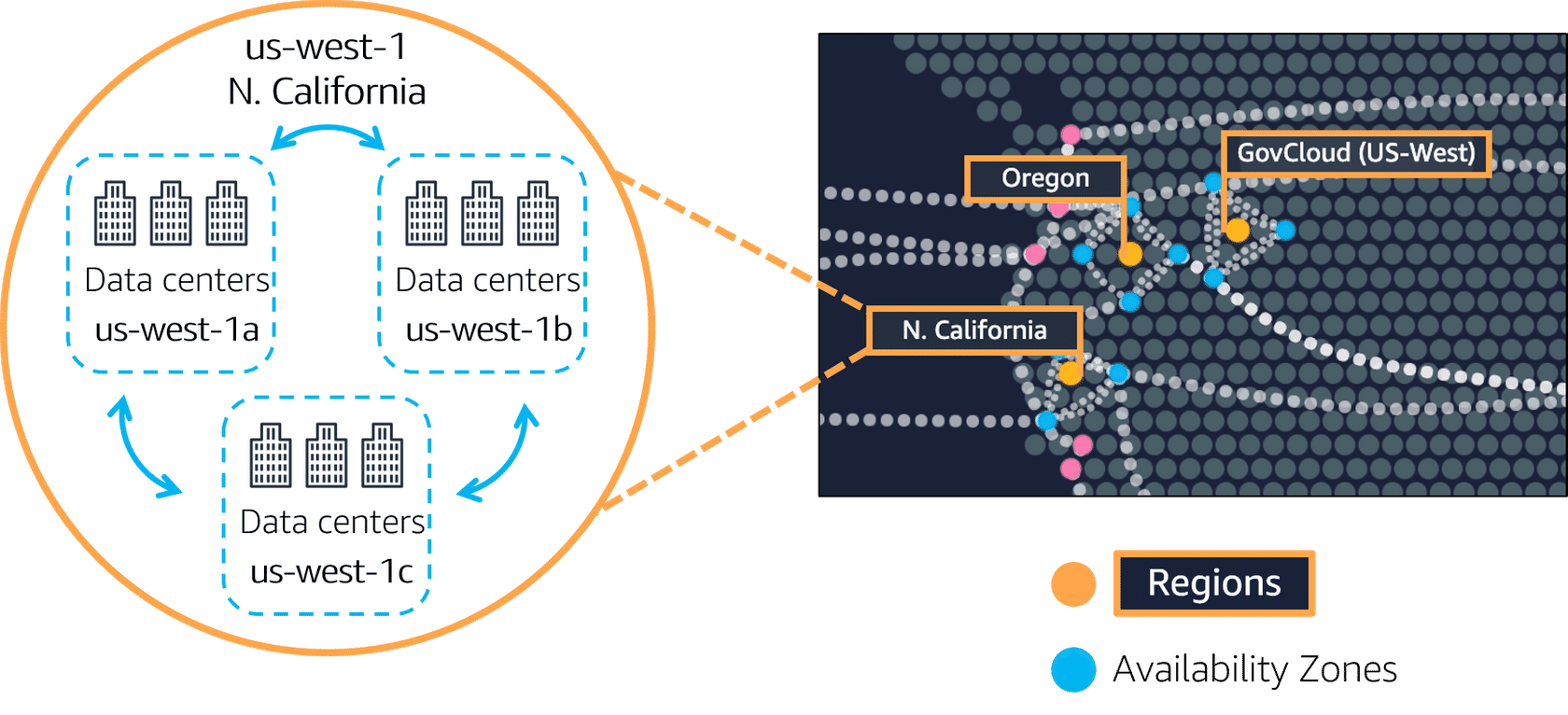
If you only run one EC2 instance, it only runs in one building, or one Availability Zone and a large scale disaster occurs, will your application still be able to run and serve your business? The obvious solution to this is to run multiple EC2 instances, just like we showed in the scaling example earlier. But the main thing is don't run them in the same building. Don't even run them in the same street, push them as far apart as you can before the speed of light tells you to stop if you still want low latency communication. Turns out that the speed of light will let us move these Availability Zones tens of miles apart from each other and still keep single digit millisecond latency between these Availability Zones. Now, if a disaster strikes, your application continues just fine because this disaster only knocked over some of your capacity, not all.

And as we saw in the last section, you can rapidly spin up more capacity in the remaining Availability Zones, thus allowing your business to continue operating without interruption. And as a best practice with AWS, we always recommend you run across at least two Availability Zones in a Region. This means redundantly deploying your infrastructure in two different AZs.

But there's more to Regions than just places to run EC2. Many of the AWS services run at the Region level, meaning they run synchronously across multiple AZs without any additional effort on your part. Take the ELB we talked about previously. This is actually a regional construct. It runs across all Availability Zones, communicating with the EC2 instances that are running in a specific Availability Zone. Regional services are by definition already highly available at no additional cost of effort on your part.

So as you plan for high availability, any service that is listed as a regionally scoped service, you'll already have that box checked. When we come back, let's look at going outside the Regions for additional solutions.

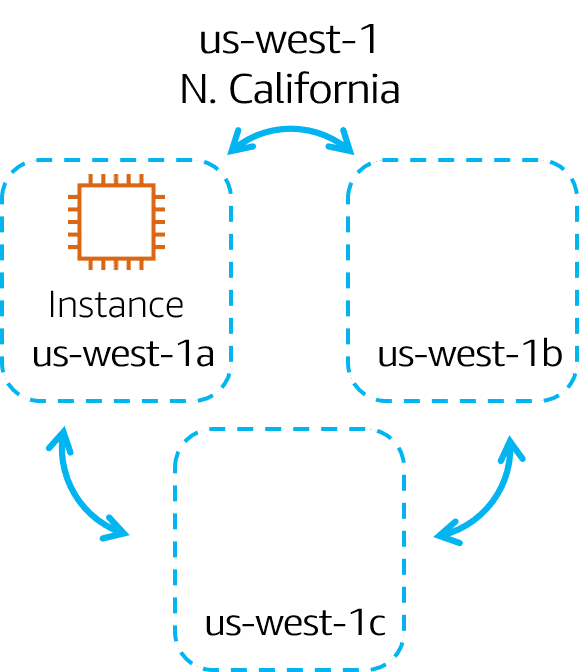
**Availability Zones**



An **Availability Zone** is a single data center or a group of data centers within a Region. Availability Zones are located tens of miles apart from each other. This is close enough to have low latency (the time between when content requested and received) between Availability Zones. However, if a disaster occurs in one part of the Region, they are distant enough to reduce the chance that multiple Availability Zones are affected.

**Step 1**

## Amazon EC2 instance in a single Availability Zone



Suppose that you’re running an application on a single Amazon EC2 instance in the Northern California Region. The instance is running in the us-west-1a Availability Zone. If us-west-1a were to fail, you would lose your instance.

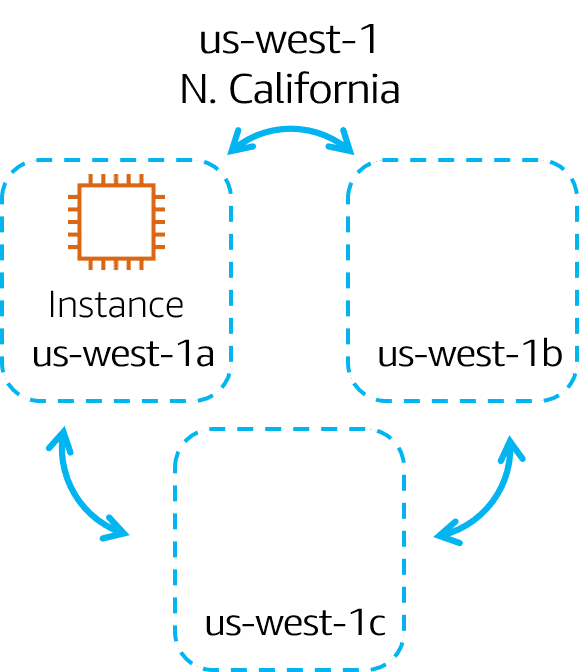
## Running Amazon EC2 instances in multiple Availability Zones

To review examples of running an Amazon EC2 instance in one or more Availability Zones, select **Start**.

START

**Step 1**

## Amazon EC2 instance in a single Availability Zone



Suppose that you’re running an application on a single Amazon EC2 instance in the Northern California Region. The instance is running in the us-west-1a Availability Zone. If us-west-1a were to fail, you would lose your instance.

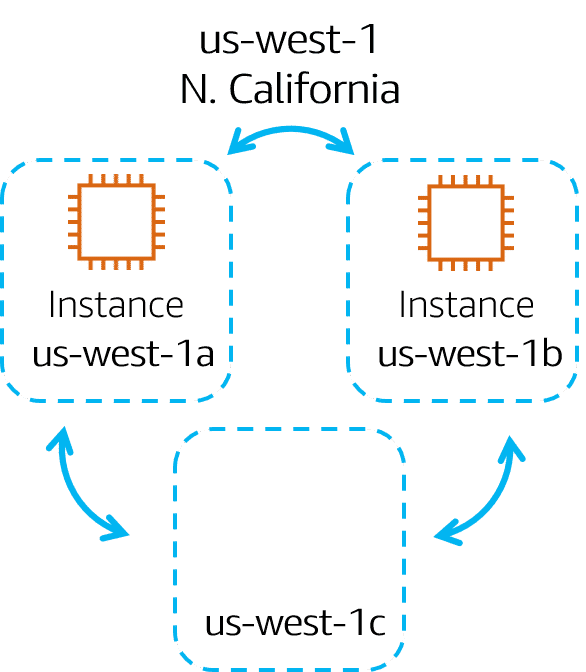
1

2

3

**Step 2**

## Amazon EC2 instances in multiple Availability Zones



A best practice is to run applications across at least two Availability Zones in a Region. In this example, you might choose to run a second Amazon EC2 instance in us-west-1b.

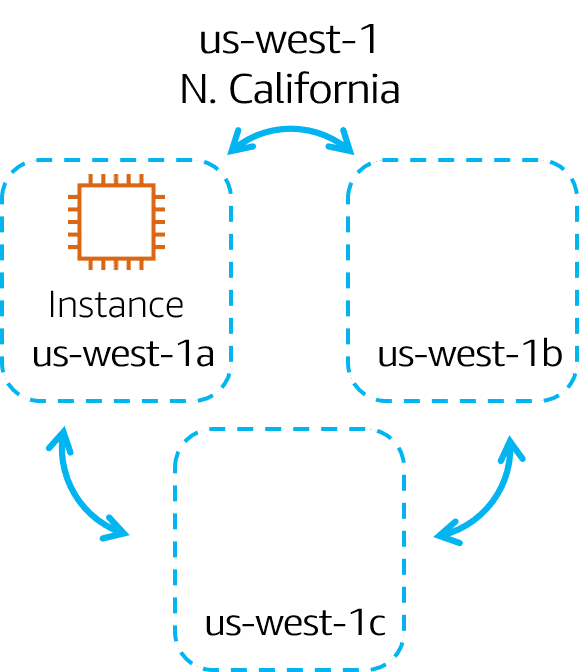
## Running Amazon EC2 instances in multiple Availability Zones

To review examples of running an Amazon EC2 instance in one or more Availability Zones, select **Start**.

START

**Step 1**

## Amazon EC2 instance in a single Availability Zone



Suppose that you’re running an application on a single Amazon EC2 instance in the Northern California Region. The instance is running in the us-west-1a Availability Zone. If us-west-1a were to fail, you would lose your instance.

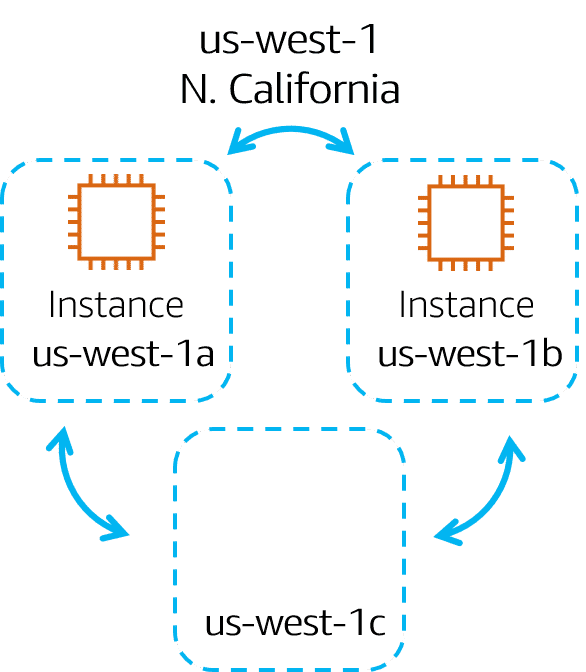
1

2

3

**Step 2**

## Amazon EC2 instances in multiple Availability Zones



A best practice is to run applications across at least two Availability Zones in a Region. In this example, you might choose to run a second Amazon EC2 instance in us-west-1b.

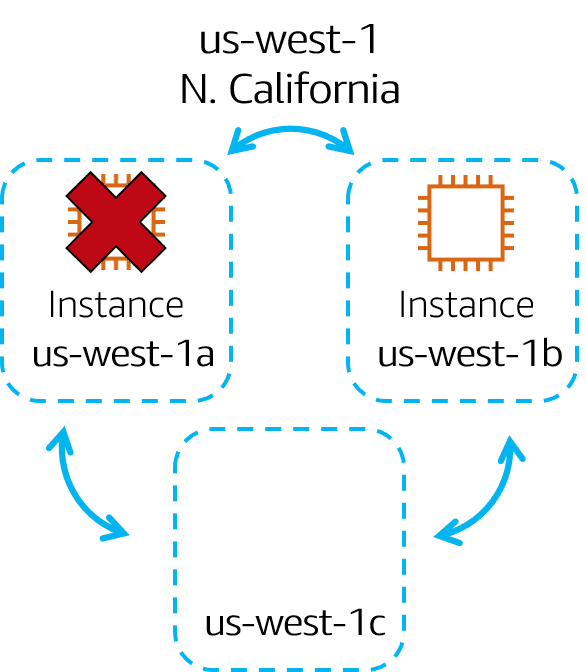
1

2

3

**Step 3**

## Availability Zone failure



If us-west-1a were to fail, your application would still be running in us-west-1b.

## Summary

Planning for failure and deploying applications across multiple Availability Zones is an important part of building a resilient and highly available architecture.

# Edge locations

Video Transcript

One of the great things about the AWS global infrastructure, is the way it's engineered to help you better serve your customers. Remember when selecting a Region, one of the major criteria was proximity to your customers, but what if you have customers all over the world or in cities that are not close to one of our Regions? Well, think about our coffee shop. If you have a good customer base in a new city, you can build a satellite store to service those customers.

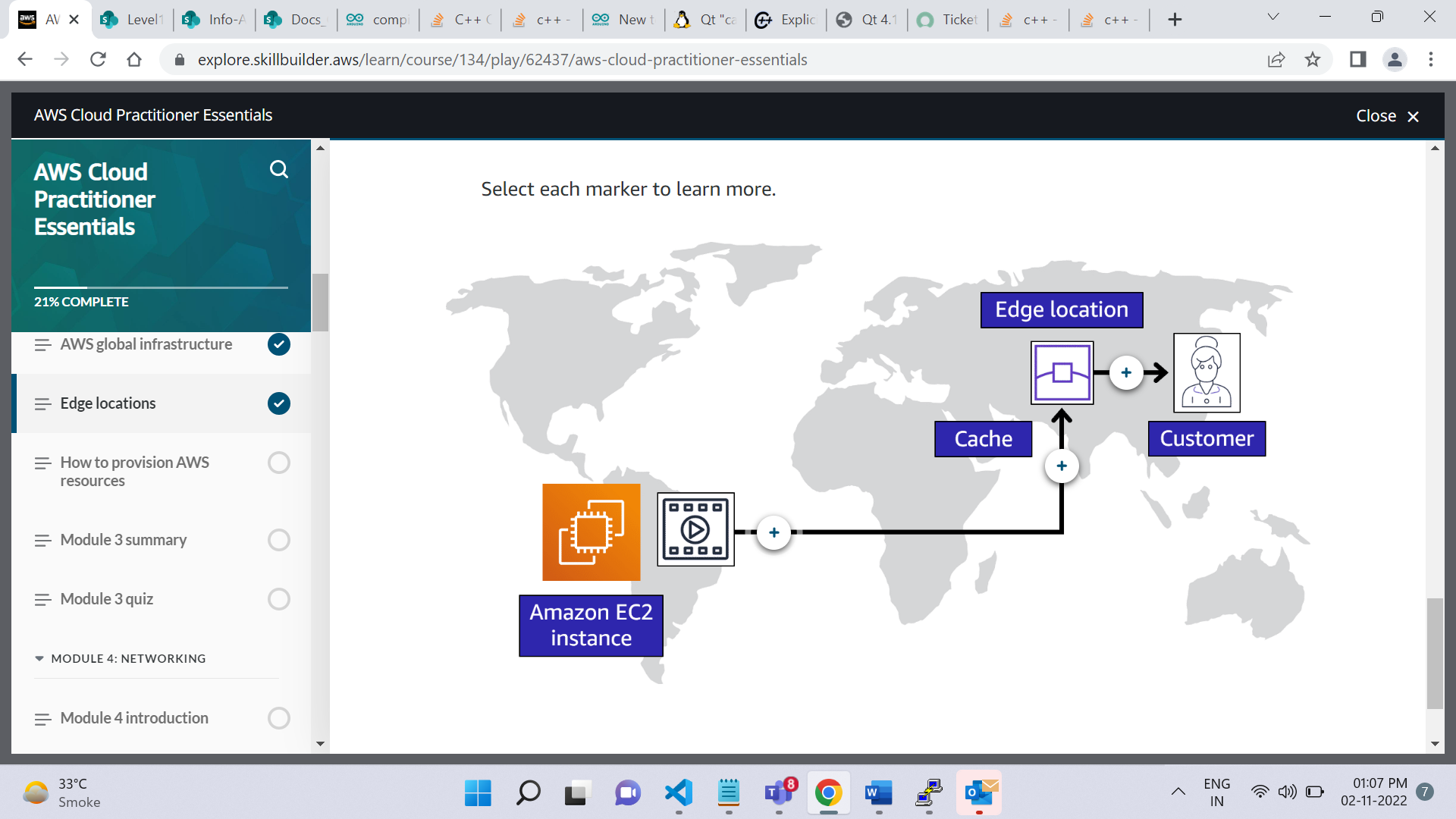
You don't need to build an entire new headquarters or from an IT perspective, if you have customers in Mumbai who need access to your data, but the data is hosted out of the Tokyo Region, rather than having all the Mumbai-based customers, send requests all the way to Tokyo, to access the data, just place a copy locally or cache a copy in Mumbai. Caching copies of data closer to the customers all around the world uses the concept of content delivery networks, or CDNs.

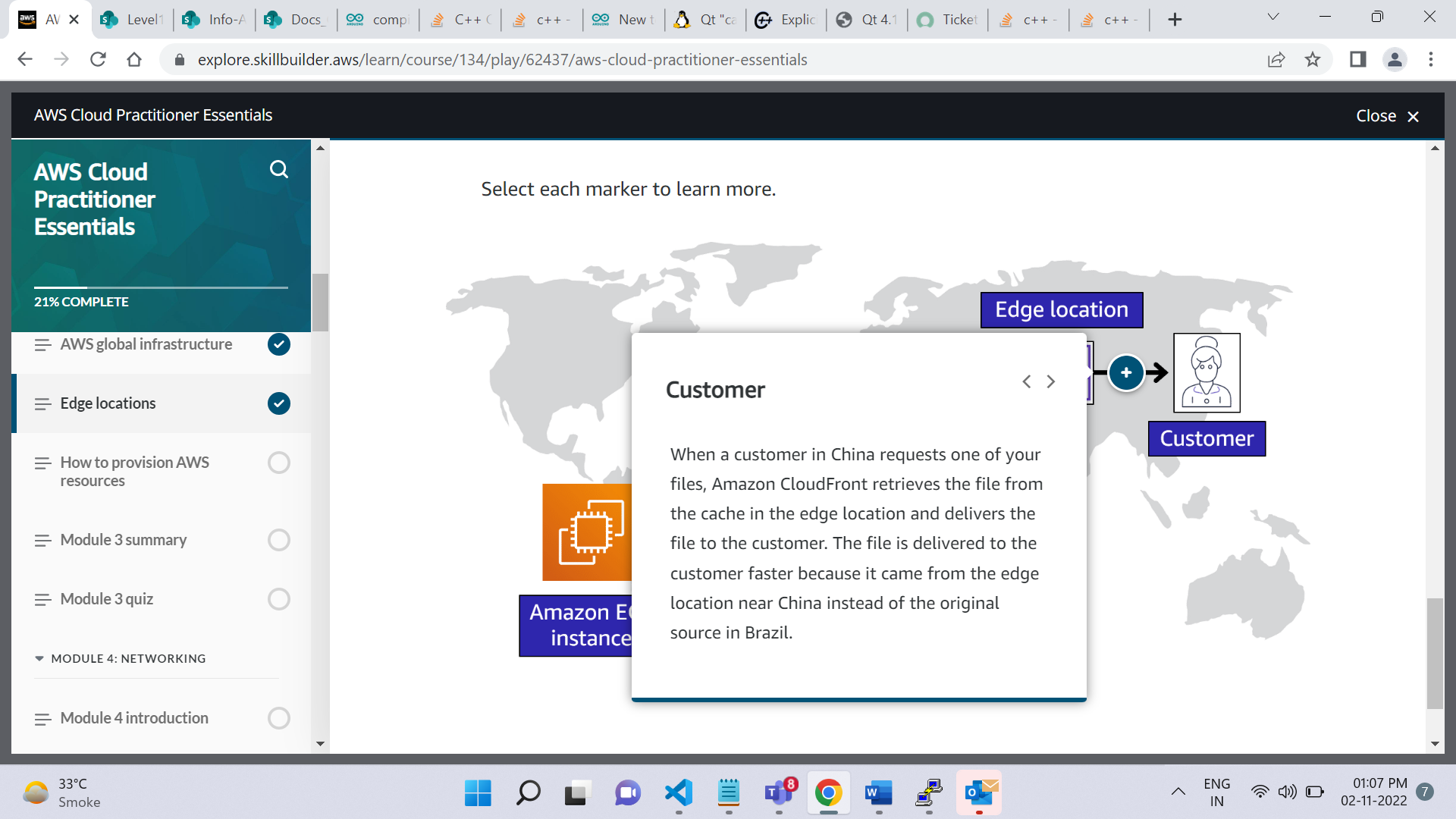
CDNs are commonly used, and on AWS, we call our CDN Amazon CloudFront. Amazon CloudFront is a service that helps deliver data, video, applications, and APIs to customers around the world with low latency and high transfer speeds. Amazon CloudFront uses what are called Edge locations, all around the world, to help accelerate communication with users, no matter where they are.

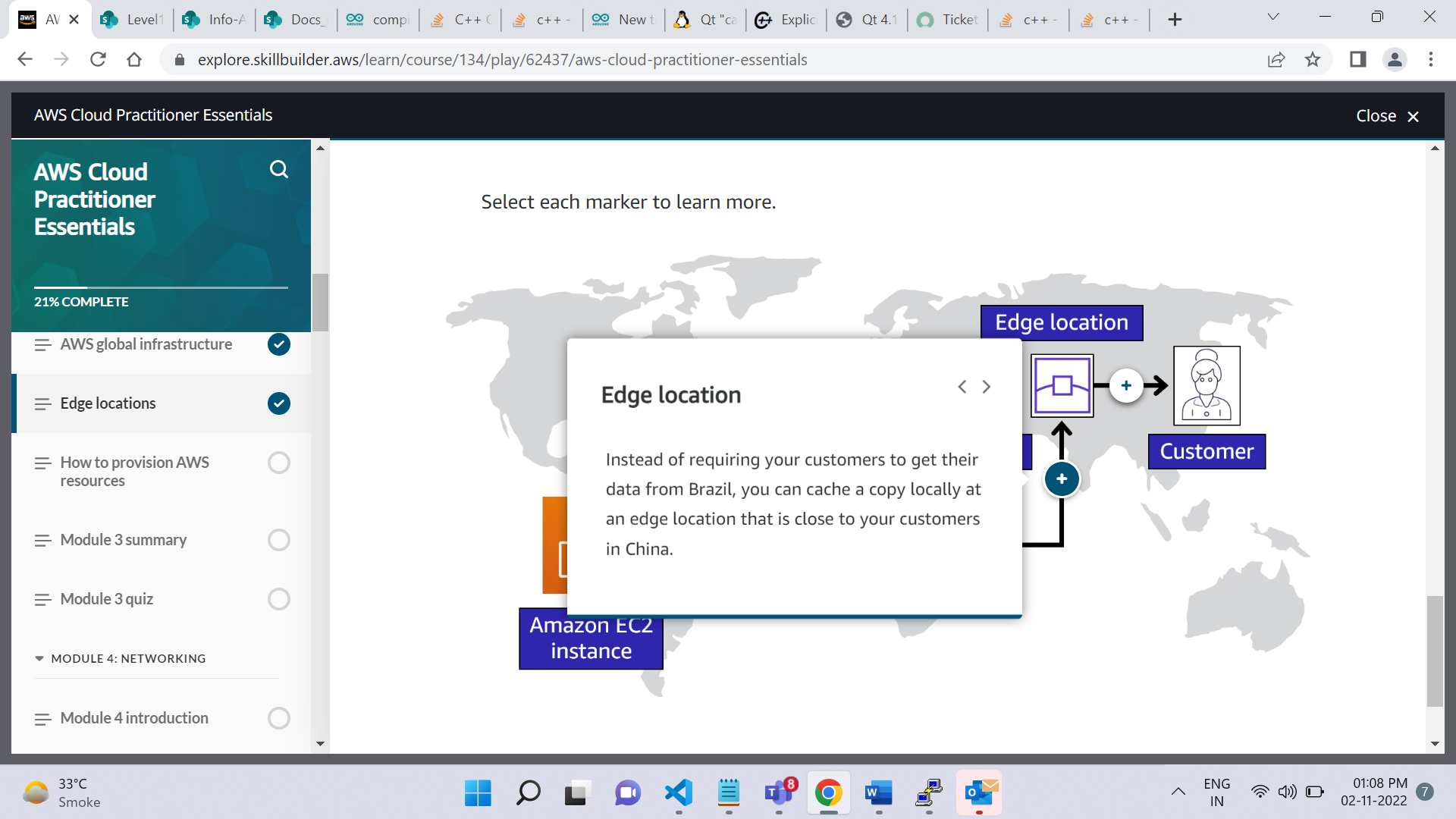
Edge locations are separate from Regions, so you can push content from inside a Region to a collection of Edge locations around the world, in order to accelerate communication and content delivery. AWS Edge locations, also run more than just CloudFront. They run a domain name service, or DNS, known as Amazon Route 53, helping direct customers to the correct web locations with reliably low latency.

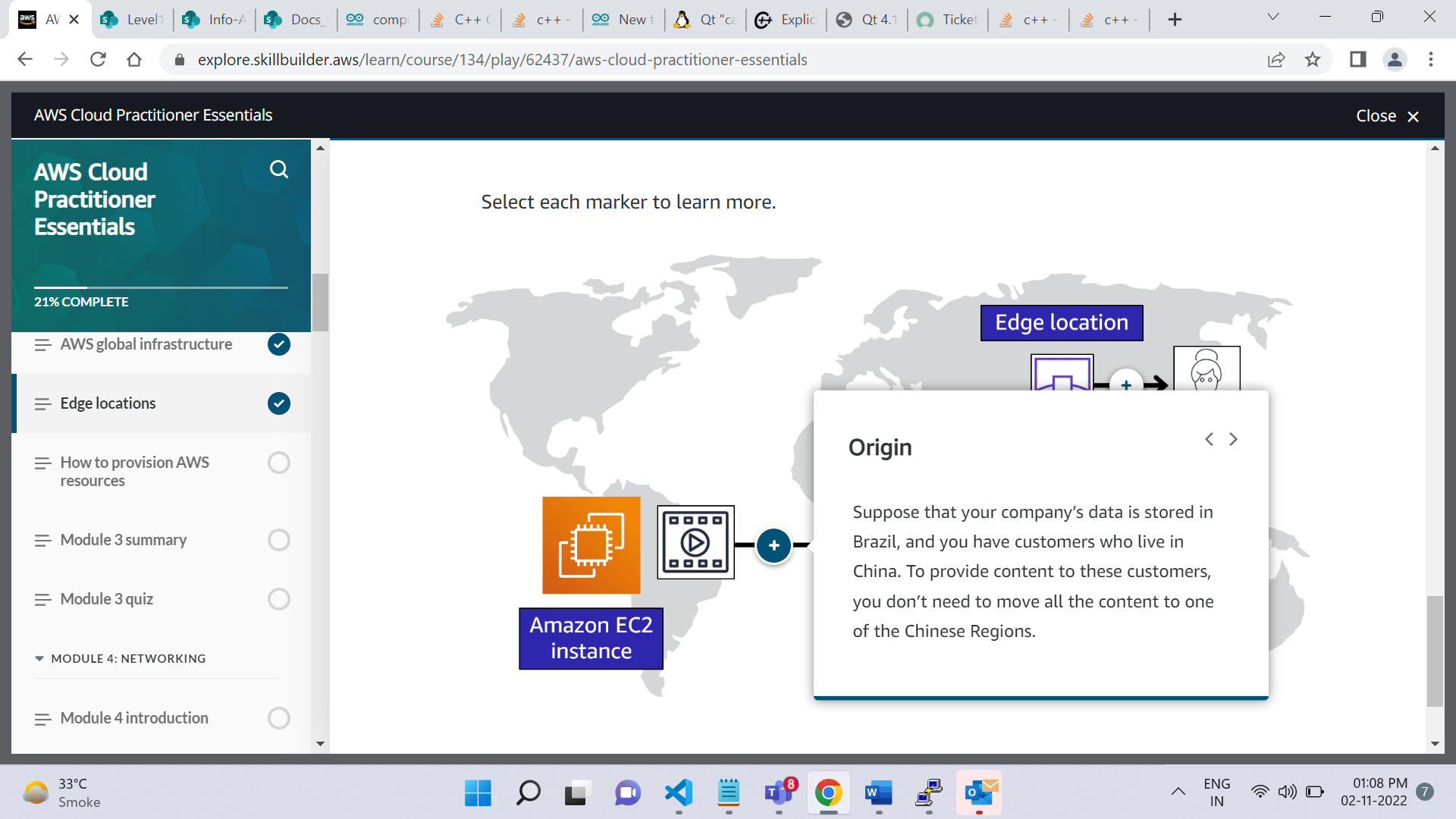
But what if your business wants to use, AWS services inside their own building? Well sure. AWS can do that for you. Introducing AWS Outposts, where AWS will basically install a fully operational mini Region, right inside your own data center. That's owned and operated by AWS, using 100% of AWS functionality, but isolated within your own building. It's not a solution most customers need, but if you have specific problems that can only be solved by staying in your own building, we understand, AWS Outposts can help.

All right, there is so much more that we can say about AWS global infrastructure, but let's keep it simple and stop here. So here's the key points. Number one, Regions are geographically isolated areas, where you can access services needed to run your enterprise. Number two, Regions contain Availability Zones, that allow you to run across physically separated buildings, tens of miles of separation, while keeping your application logically unified. Availability Zones help you solve high availability and disaster recovery scenarios, without any additional effort on your part, and number three, AWS Edge locations run Amazon CloudFront to help get content closer to your customers, no matter where they are in the world.









# How to provision AWS resources

**Video transcript**

–

We've been talking about a few different AWS resources as well as the AWS global infrastructure. You may be wondering, how do I actually interact with these services? And the answer is APIs. In AWS, everything is an API call. An API is an application programming interface. And what that means is, there are pre determined ways for you to interact with AWS services. And you can invoke or call these APIs to provision, configure, and manage your AWS resources.

For example, you can launch an EC2 instance or you can create an AWS Lambda function. Each of those would be different requests and different API calls to AWS. You can use the AWS Management Console, the AWS Command Line Interface, the AWS Software Development Kits, or various other tools like AWS CloudFormation, to create requests to send to AWS APIs to create and manage AWS resources.

First, let's talk about the AWS Management Console. The AWS Management Console is browser-based. Through the console, you can manage your AWS resources visually and in a way that is easy to digest. This is great for getting started and building your knowledge of the services. It's also useful for building out test environments or viewing AWS bills, viewing monitoring and working with other non technical resources. The AWS Management Console is most likely the first place you will go when you are learning about AWS.

However, once you are up and running in a production type environment, you don't want to rely on the point and click style that the console gives you to create and manage your AWS resources. For example, in order to create an Amazon EC2 Instance, you need to click through various screens, setting all the configurations you want, and then you launch your instance. If later, you wanted to launch another EC2 Instance, you would need to go back into the console and click through those screens again to get it up and running. By having humans do this sort of manual provisioning, you're opening yourself up to potential errors. It's pretty easy to forget to check a checkbox or misspell something when you are doing everything manually.

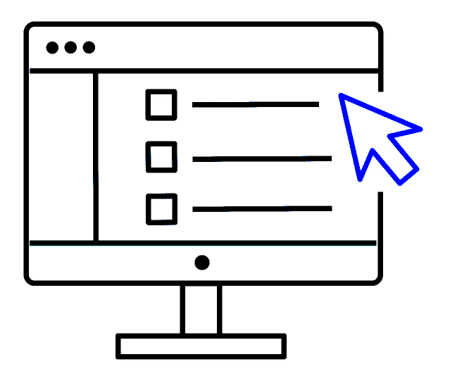
The answer to this problem is to use tools that allow you to script or program the API calls. One tool you can use is the AWS Command Line Interface or CLI. The CLI allows you to make API calls using the terminal on your machine. This is different than the visual navigation style of the Management Console. Writing commands using the CLI makes actions scriptable and repeatable. So, you can write and run your commands to launch an EC2 Instance. And if you want to launch another, you can just run the pre-written command again. This makes it less susceptible to human error. And you can have these scripts run automatically, like on a schedule or triggered by another process.

Automation is very important to having a successful and predictable cloud deployment over time. Another way to interact with AWS is through the AWS Software Development Kits or SDKs. The SDKs allow you to interact with AWS resources through various programming languages. This makes it easy for developers to create programs that use AWS without using the low level APIs, as well as avoiding that manual resource creation that we just talked about. More on that in a bit.

**Ways to interact with AWS services**

To learn about each category, select each tab.

The **AWS Management Console** is a (browser)web-based interface for accessing and managing AWS services. You can quickly access recently used services and search for other services by name, keyword, or acronym. The console includes wizards and automated workflows that can simplify the process of completing tasks.  
  
You can also use the AWS Console mobile application to perform tasks such as monitoring resources, viewing alarms, and accessing billing information. Multiple identities can stay logged into the AWS Console mobile app at the same time.



**AWS Command Line Interface (AWS CLI)**

To save time when making API requests, you can use the **AWS Command Line Interface (AWS CLI)**. AWS CLI enables you to control multiple AWS services directly from the command line within one tool. AWS CLI is available for users on Windows, macOS, and Linux.

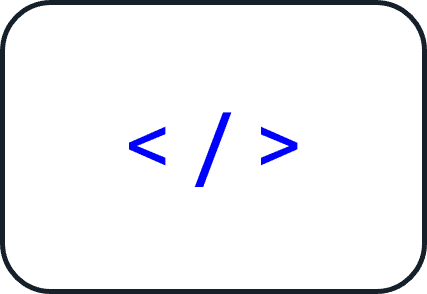
By using AWS CLI, you can automate the actions that your services and applications perform through scripts. For example, you can use commands to launch an Amazon EC2 instance, connect an Amazon EC2 instance to a specific Auto Scaling group, and more.



**software development kits (SDKs)**.

Another option for accessing and managing AWS services is the **software development kits (SDKs)**. SDKs make it easier for you to use AWS services through an API designed for your programming language or platform. SDKs enable you to use AWS services with your existing applications or create entirely new applications that will run on AWS.

To help you get started with using SDKs, AWS provides documentation and sample code for each supported programming language. Supported programming languages include C++, Java, .NET, and more.



**Video transcript**

–

All right, let's continue to talk about how to interact with AWS. You have the AWS Management Console, the CLI, and the SDKs, which are all sort of do it yourself ways to provision and manage your AWS environment. If you want to provision a resource, you can log into the console and point and click, you can write some commands, or you can write some programs to do it. There are also other ways you can manage your AWS environment using managed tools like AWS Elastic Beanstalk, and AWS CloudFormation.

AWS Elastic Beanstalk is a service that helps you provision Amazon EC2-based environments. Instead of clicking around the console or writing multiple commands to build out your network, EC2 instances, scaling and Elastic Load Balancers, you can instead provide your application code and desired configurations to the AWS Elastic Beanstalk service, which then takes that information and builds out your environment for you. AWS Elastic Beanstalk also makes it easy to save environment configurations, so they can be deployed again easily. AWS Elastic Beanstalk gives you the convenience of not having to provision and manage all of these pieces separately, while still giving you the visibility and control of the underlying resources. You get to focus on your business application, not the infrastructure.

Another service you can use to help create automated and repeatable deployments is AWS CloudFormation. AWS CloudFormation is an infrastructure as code tool that allows you to define a wide variety of AWS resources in a declarative way using JSON or YAML text-based documents called CloudFormation templates. A declarative format like this allows you to define what you want to build without specifying the details of exactly how to build it. CloudFormation lets you define what you want and the CloudFormation engine will worry about the details on calling APIs to get everything built out.

It also isn't just limited to EC2-based solutions. CloudFormation supports many different AWS resources from storage, databases, analytics, machine learning, and more. Once you define your resources in a CloudFormation template, CloudFormation will parse the template and begin provisioning all the resources you defined in parallel. CloudFormation manages all the calls to the backend AWS APIs for you. You can run the same CloudFormation template in multiple accounts or multiple regions, and it will create identical environments across them. There is less room for human error as it is a totally automated process.

To recap, the AWS Management Console is great for learning and providing a visual for the user. The AWS Management Console is a manual tool. So right off the bat, it isn't a great option for automation. You can instead use the CLI to script your interactions with AWS using the terminal. You can use the SDKs to write programs to interact with AWS for you or you can use manage tools like AWS Elastic Beanstalk or AWS CloudFormation

**AWS Elastic Beanstalk**

With **AWS Elastic Beanstalk**, you provide code and configuration settings, and Elastic Beanstalk deploys the resources necessary to perform the following tasks:

* Adjust capacity
* Load balancing
* Automatic scaling
* Application health monitoring

**AWS CloudFormation**

With **AWS CloudFormation**, you can treat your infrastructure as code. This means that you can build an environment by writing lines of code instead of using the AWS Management Console to individually provision resources.

AWS CloudFormation provisions your resources in a safe, repeatable manner, enabling you to frequently build your infrastructure and applications without having to perform manual actions. It determines the right operations to perform when managing your stack and rolls back changes automatically if it detects errors.

# Module 3 summary

In Module 3, you learned about the following concepts:

* AWS Regions and Availability Zones
* Edge locations and Amazon CloudFront
* The AWS Management Console, AWS CLI, and SDKs
* AWS Elastic Beanstalk
* AWS CloudFormation
* **Video transcript**
* –
* Awesome stuff, I mean, we covered a lot of ground in here. And I don't mean that just because we talked about AWS global infrastructure.
* But seriously, we covered how logical clusters of data centers make up Availability Zones, Availability Zones in turn make up Regions, and those are spread globally. You then choose what Regions and Availability Zones you want to operate out of and as a best practice, you should always deploy infrastructure across at least two Availability Zones. And some AWS services like Elastic Load Balancing, Amazon SQS, and Amazon SNS already do this for you.
* We also talked about Edge locations and how you can deploy content there to speed up delivery to your customers. We even touched upon edge devices like AWS Outposts which allow you to run AWS infrastructure right in your own data center.
* Another thing we discussed was how to provision AWS resources through various options, such as the AWS Management Console, the SDK, CLI, AWS Elastic Beanstalk, and AWS CloudFormation, where you learned how you can set up your infrastructure as code.
* I hope you learned how globally available AWS is and how easy it is to get started with provisioning resources.