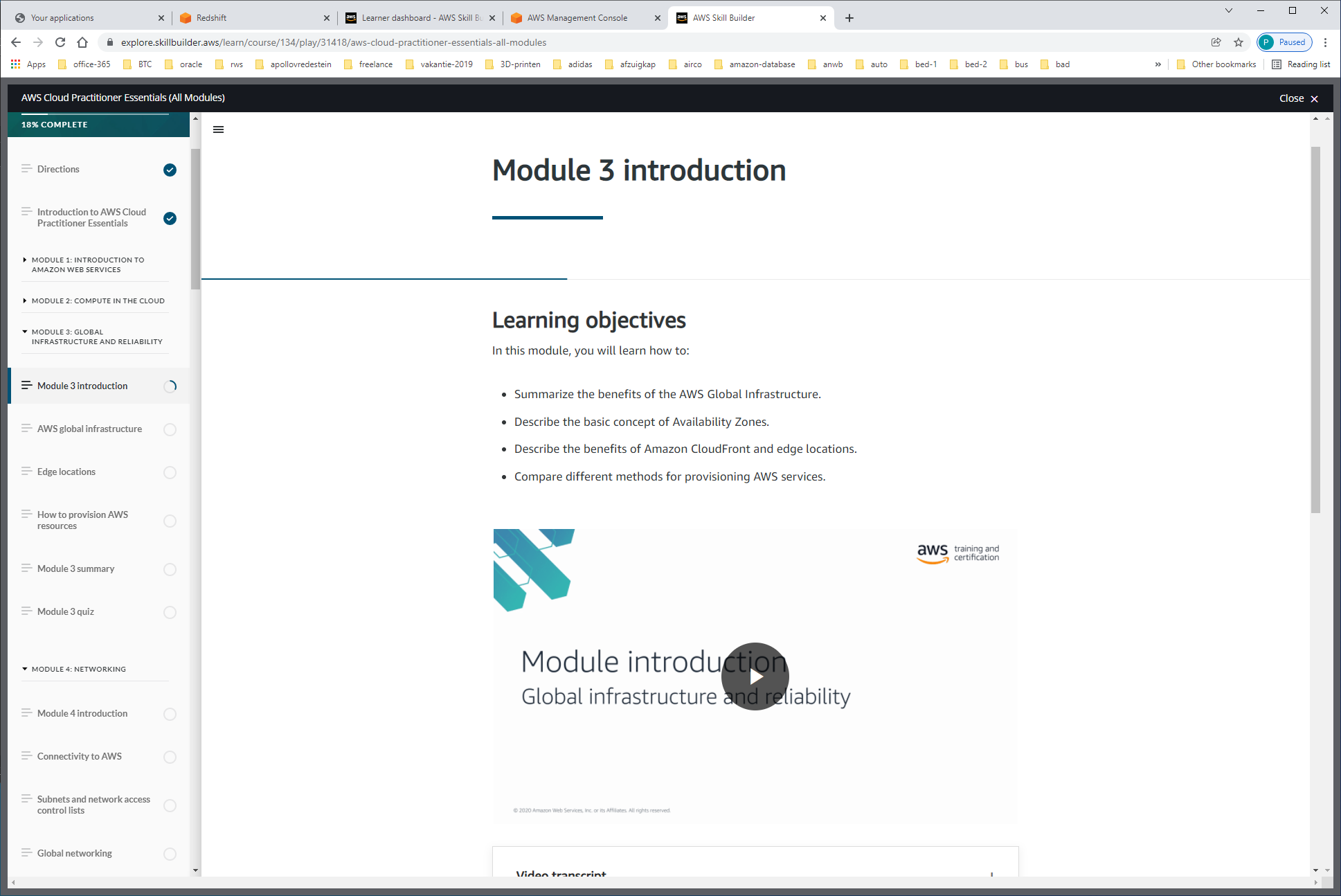
https://explore.skillbuilder.aws/learn/course/134/play/31418/aws-cloud-practitioner-essentials-all-modules



AWS GLOBAL INFRASTRUCTURE

REGIONS

Afbeelding met tekst, schermafbeelding, monitor

Automatisch gegenereerde beschrijving

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

**Selecting a Region**

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

**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.

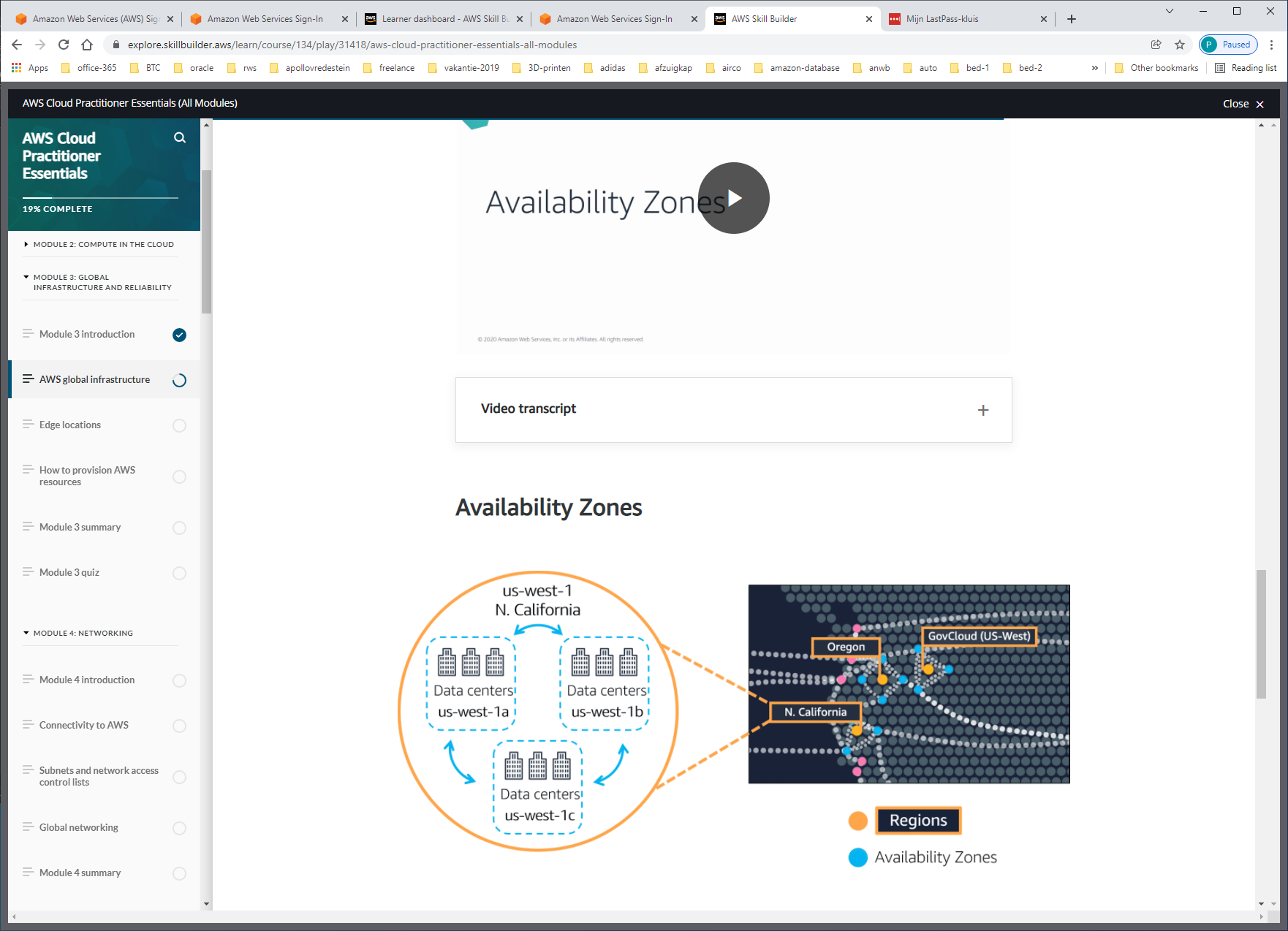
**Available services within a Region**

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.

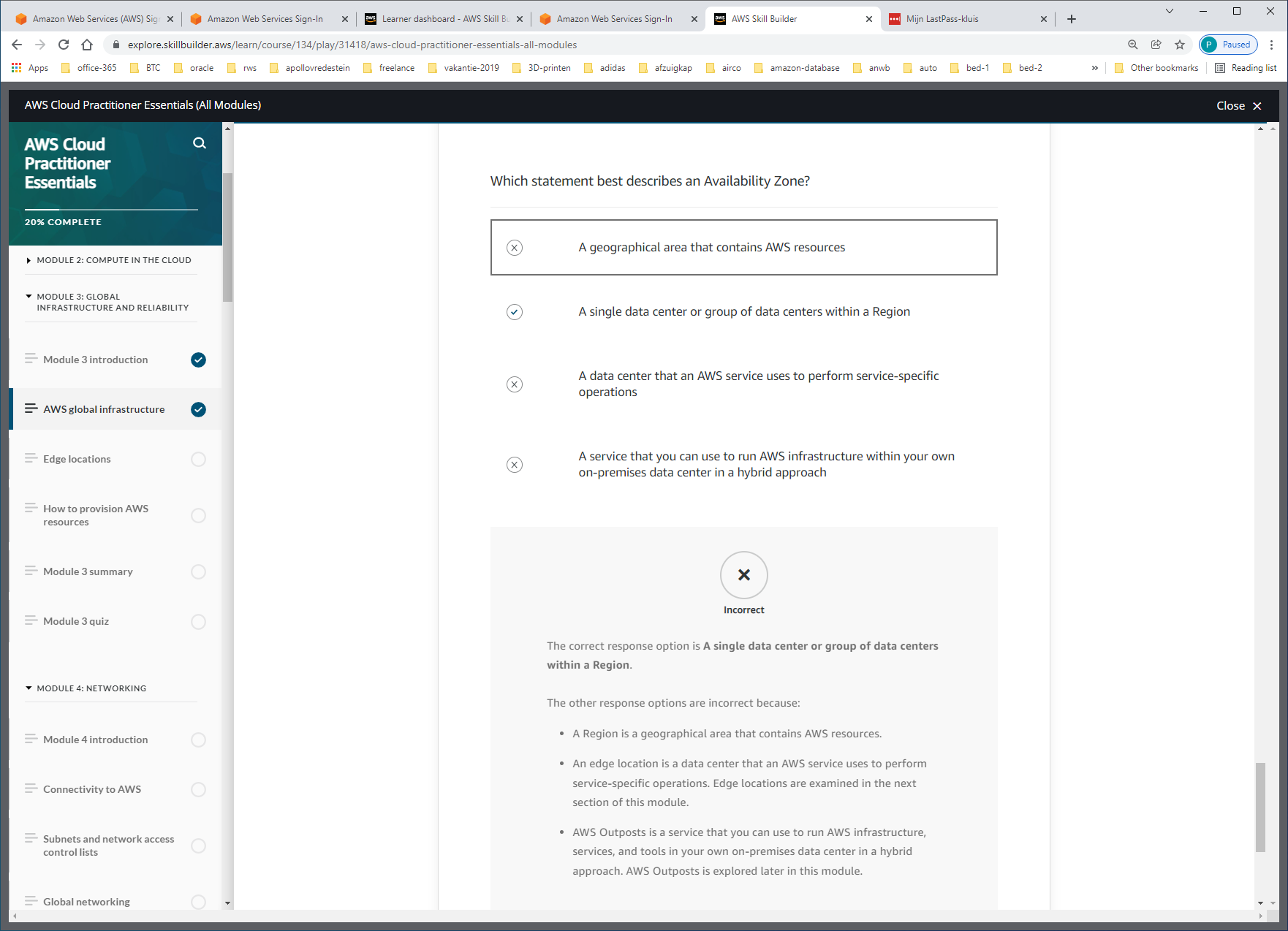


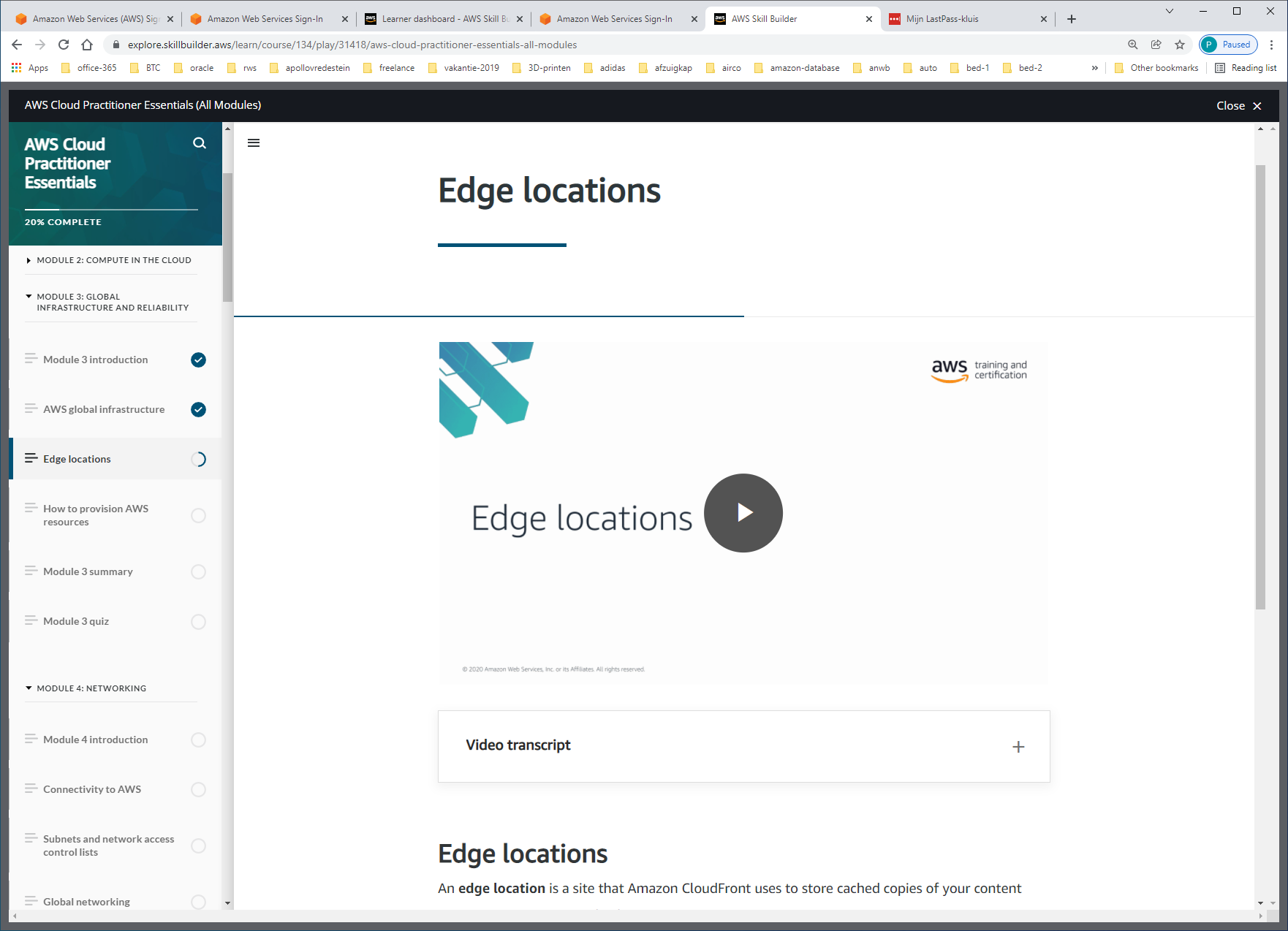
An **Availability Zone** is a single data center or a group of data centers within a Region. Availability Zones are located tens of miles apart from each other. This is close enough to have low latency (the time between when content requested and received) between Availability Zones. However, if a disaster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

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.

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.

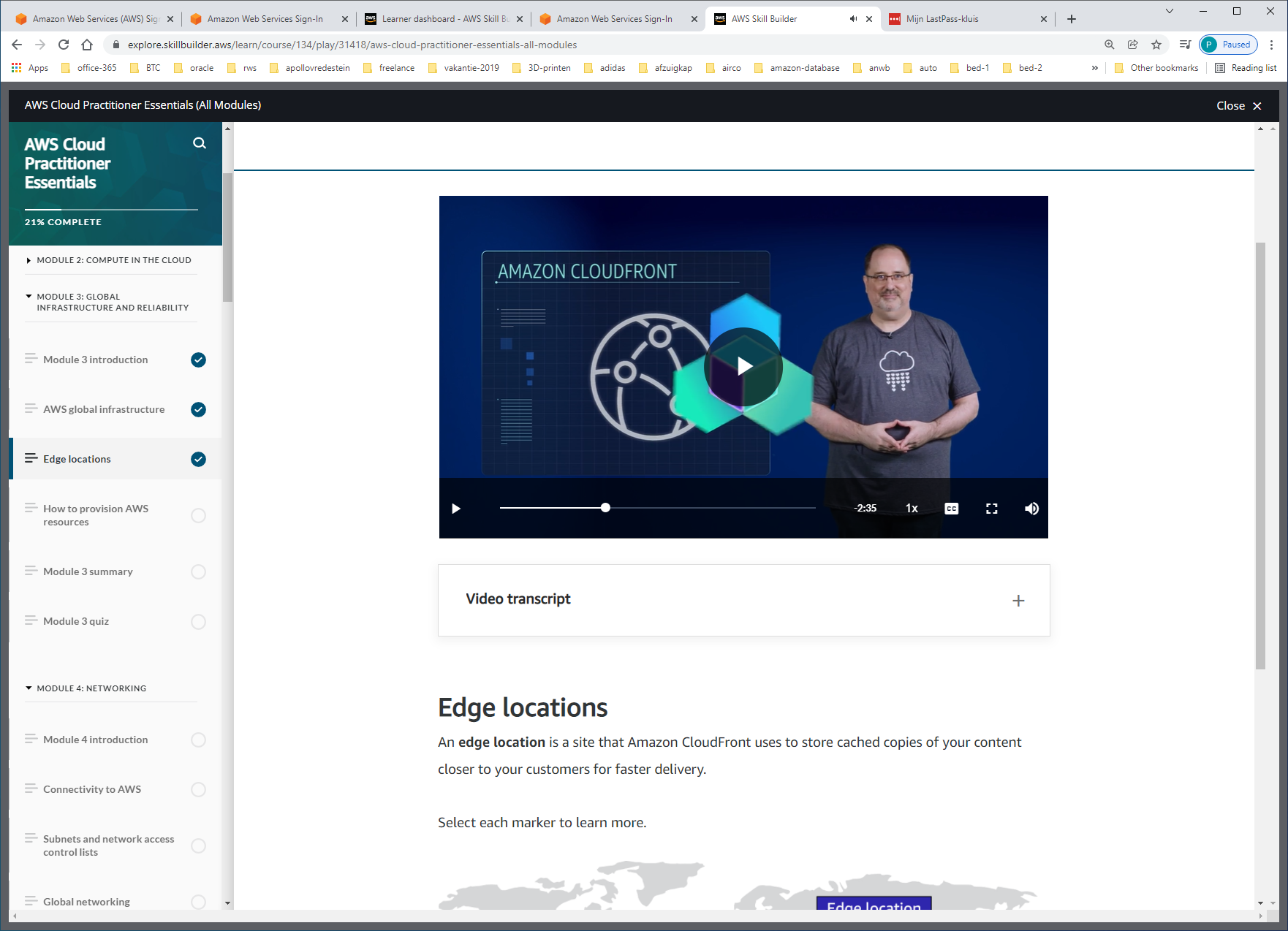
But there's more to Regions than just places to run EC2. Many of the AWS services run at the Region level, meaning they run synchronously across multiple AZs without any additional effort on your part. Take the ELB we talked about previously. This is actually a regional construct. It runs across all 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.





**Edge locations**

An **edge location** is a site that Amazon CloudFront uses to store cached copies of your content closer to your customers for faster delivery.

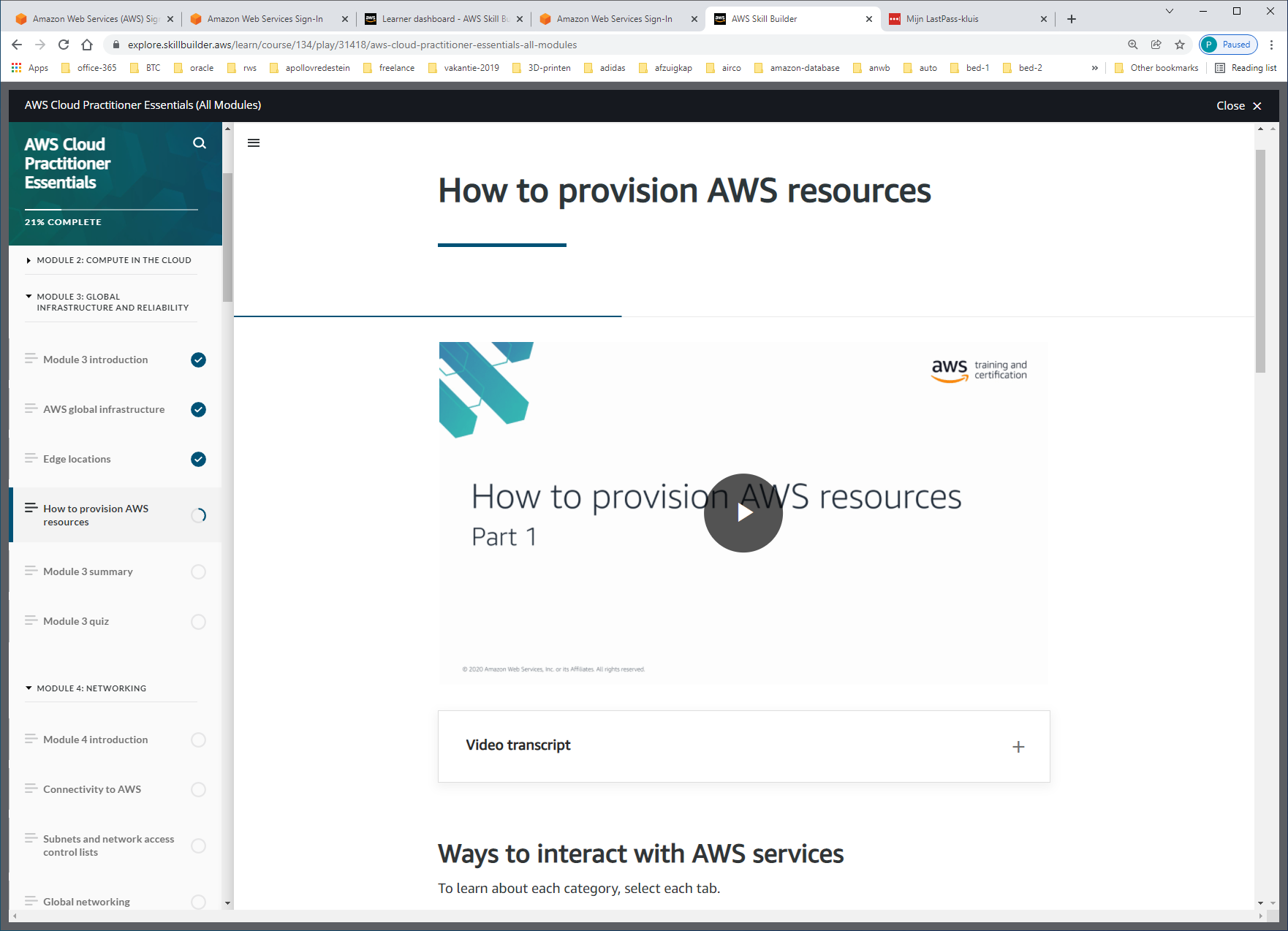


CDNs are commonly used, and on AWS, we call our CDN Amazon CloudFront.

**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.



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.

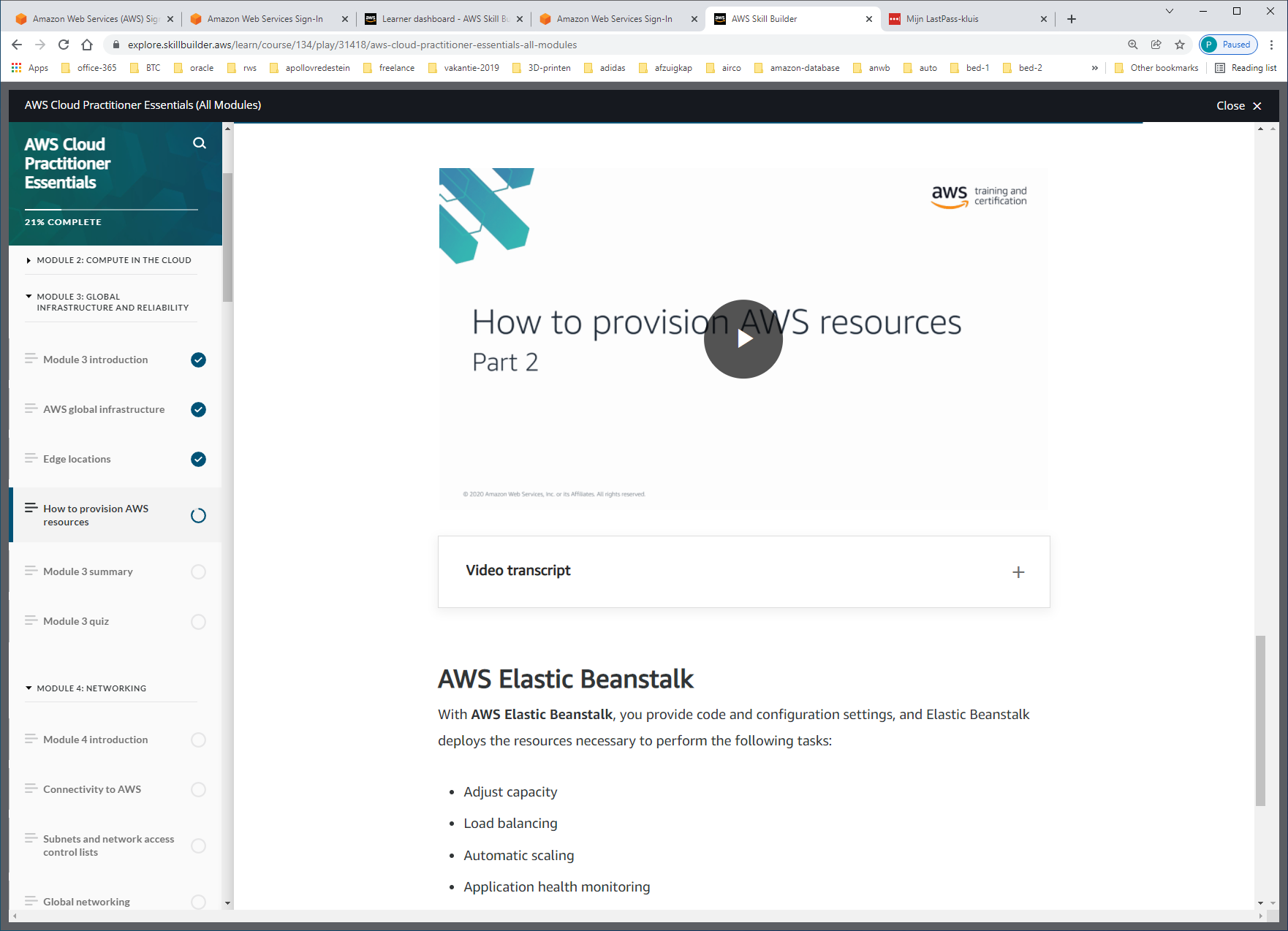
You can use **the AWS Management Console**, the **AWS Command Line Interface**, the **AWS Software Development Kits**, or various other tools like **AWS CloudFormation**, to create requests to send to AWS APIs to create and manage AWS resources.

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.

 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.

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.



**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 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.

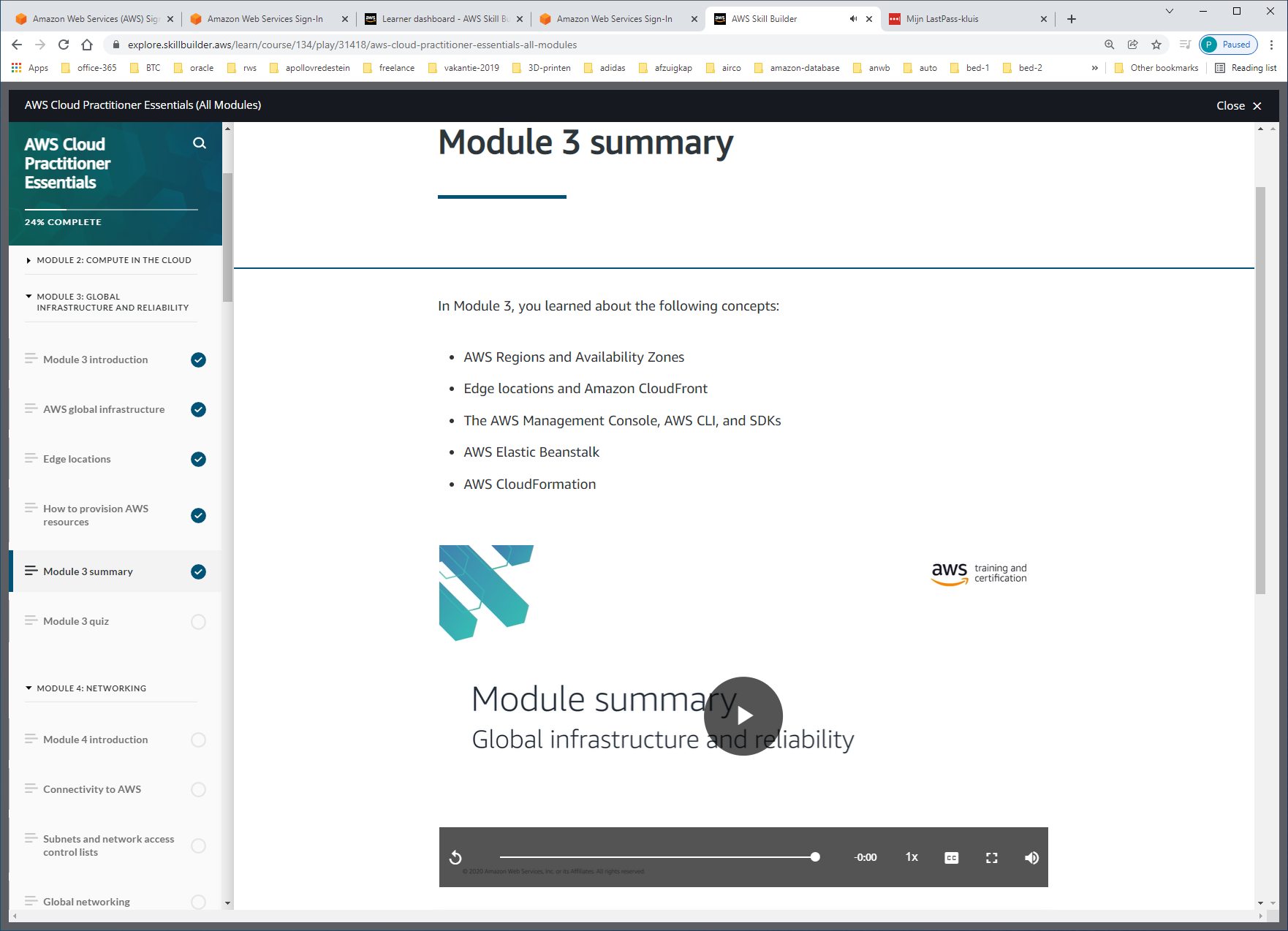
AWS Elastic Beanstalk gives you the convenience of not having to provision and manage all of these pieces separately, while still giving you the visibility and control of the underlying resources. You get to focus on your business application, not the infrastructure.

**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 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.

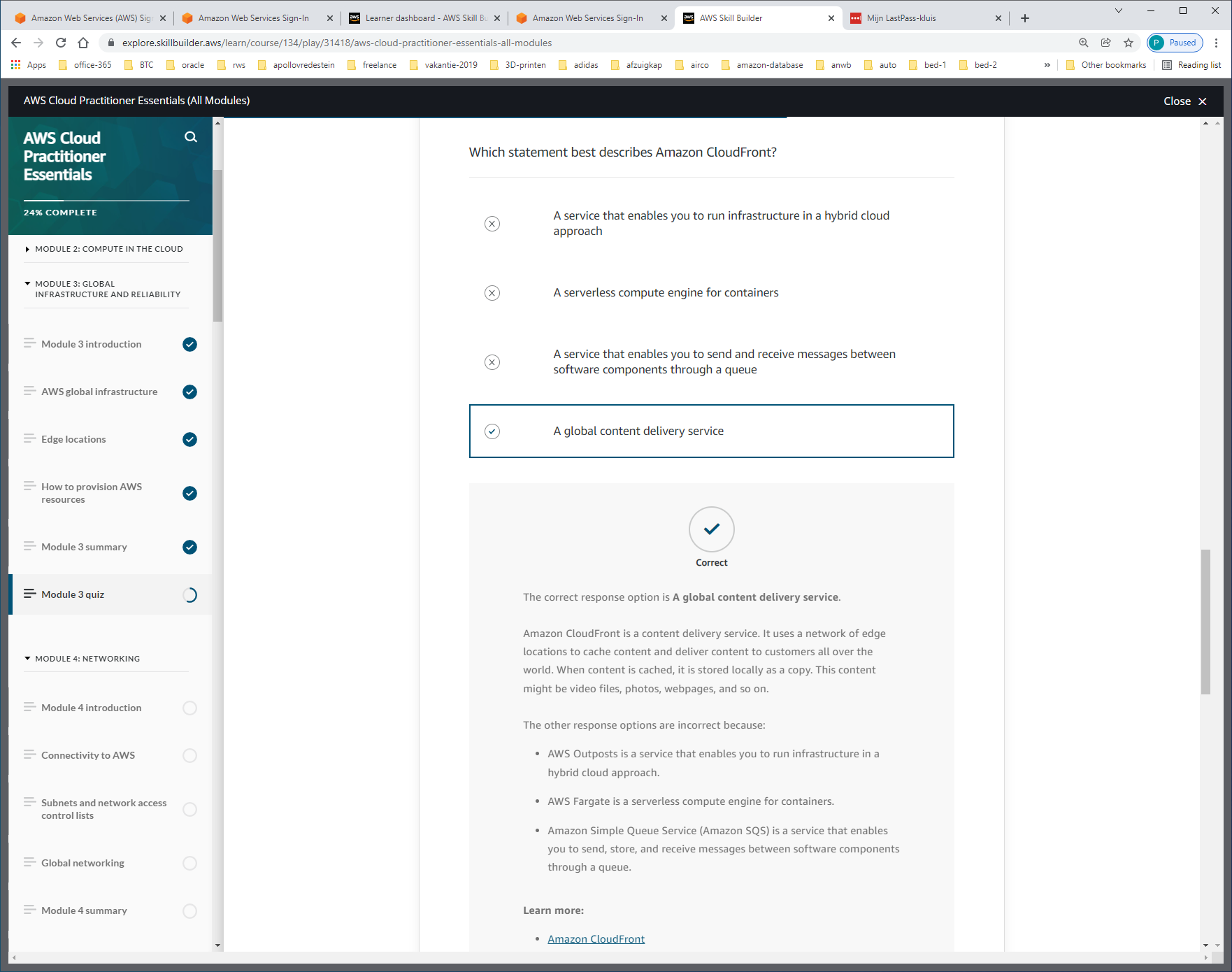


Afbeelding met tekst, schermafbeelding, computer, monitor

Automatisch gegenereerde beschrijving

Afbeelding met tekst, computer, schermafbeelding

Automatisch gegenereerde beschrijving



Afbeelding met tekst, schermafbeelding, monitor

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermafbeelding, monitor

Automatisch gegenereerde beschrijving