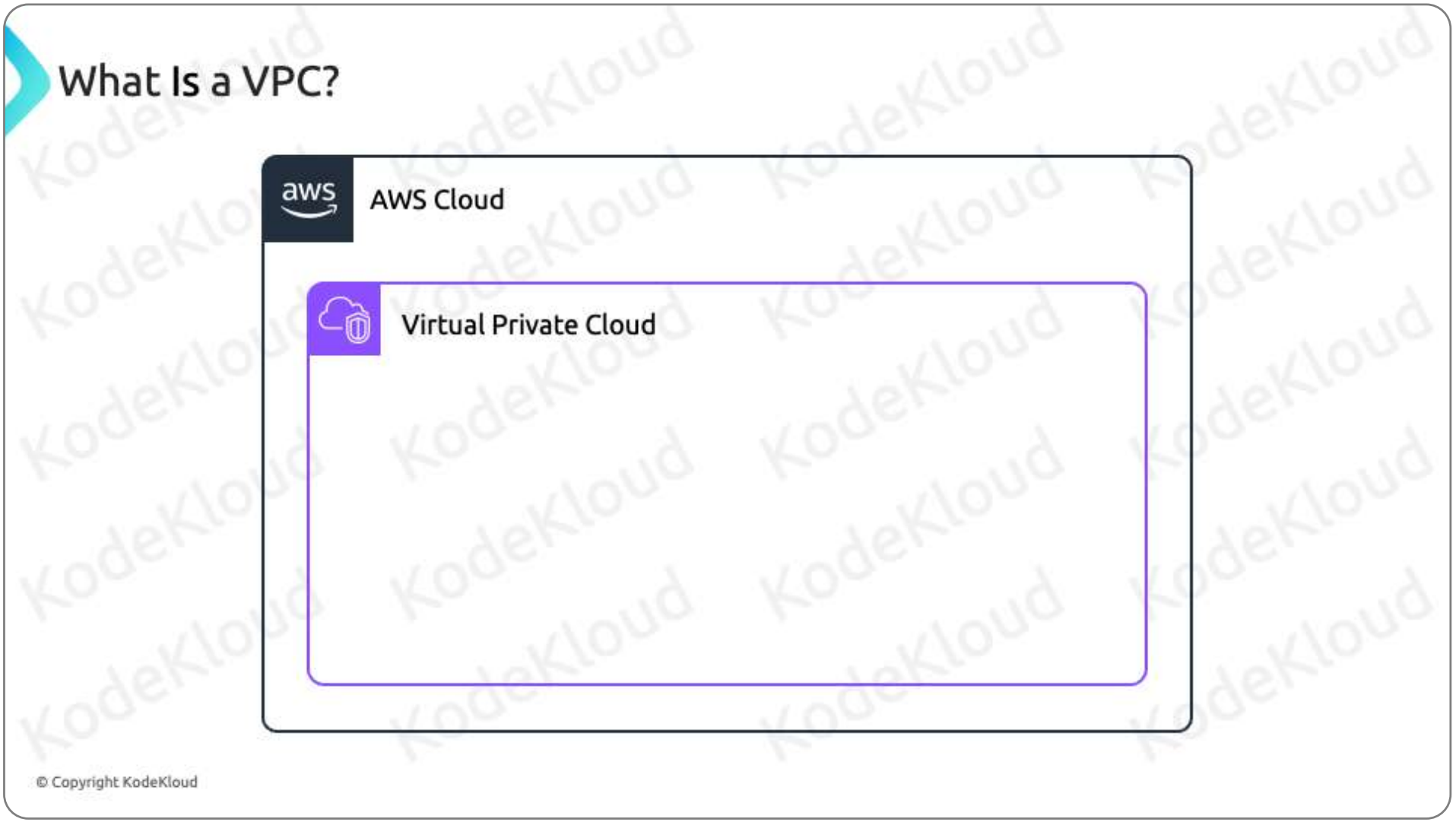
KodeKloud

Service Section

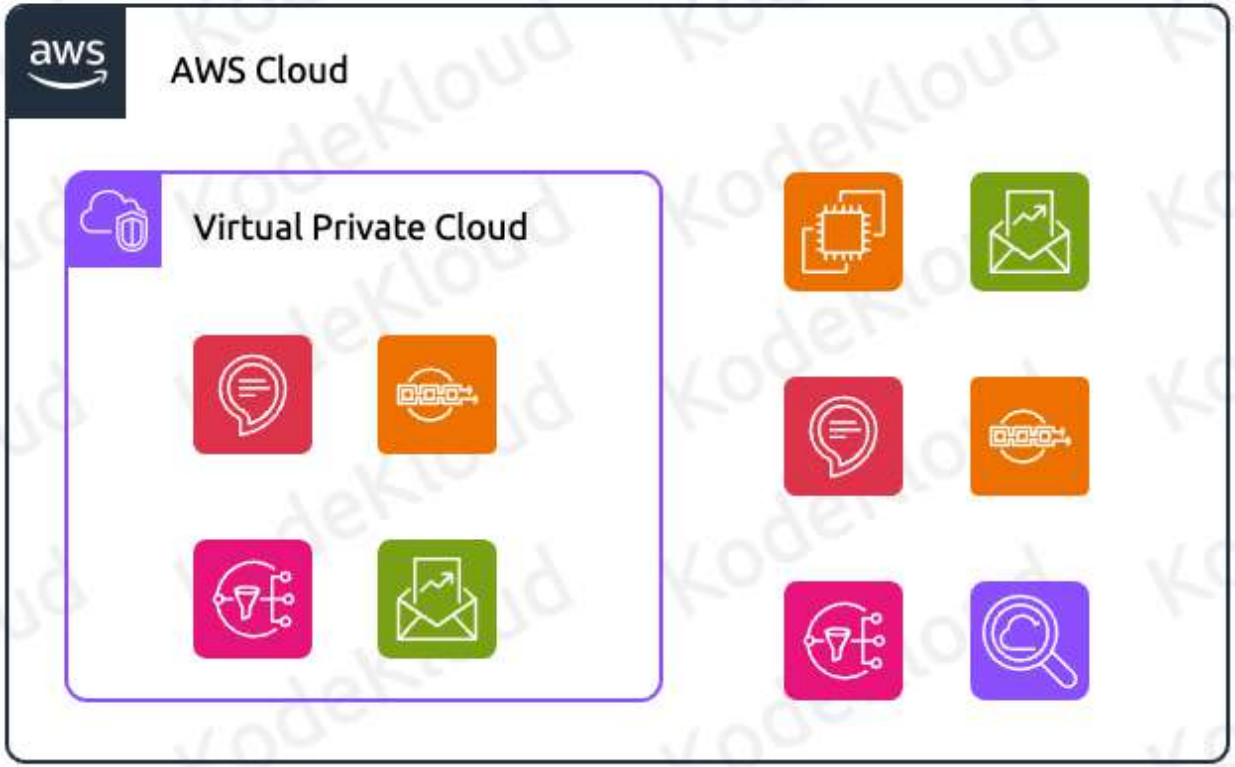
The Power of Core Networking

VPC Overview

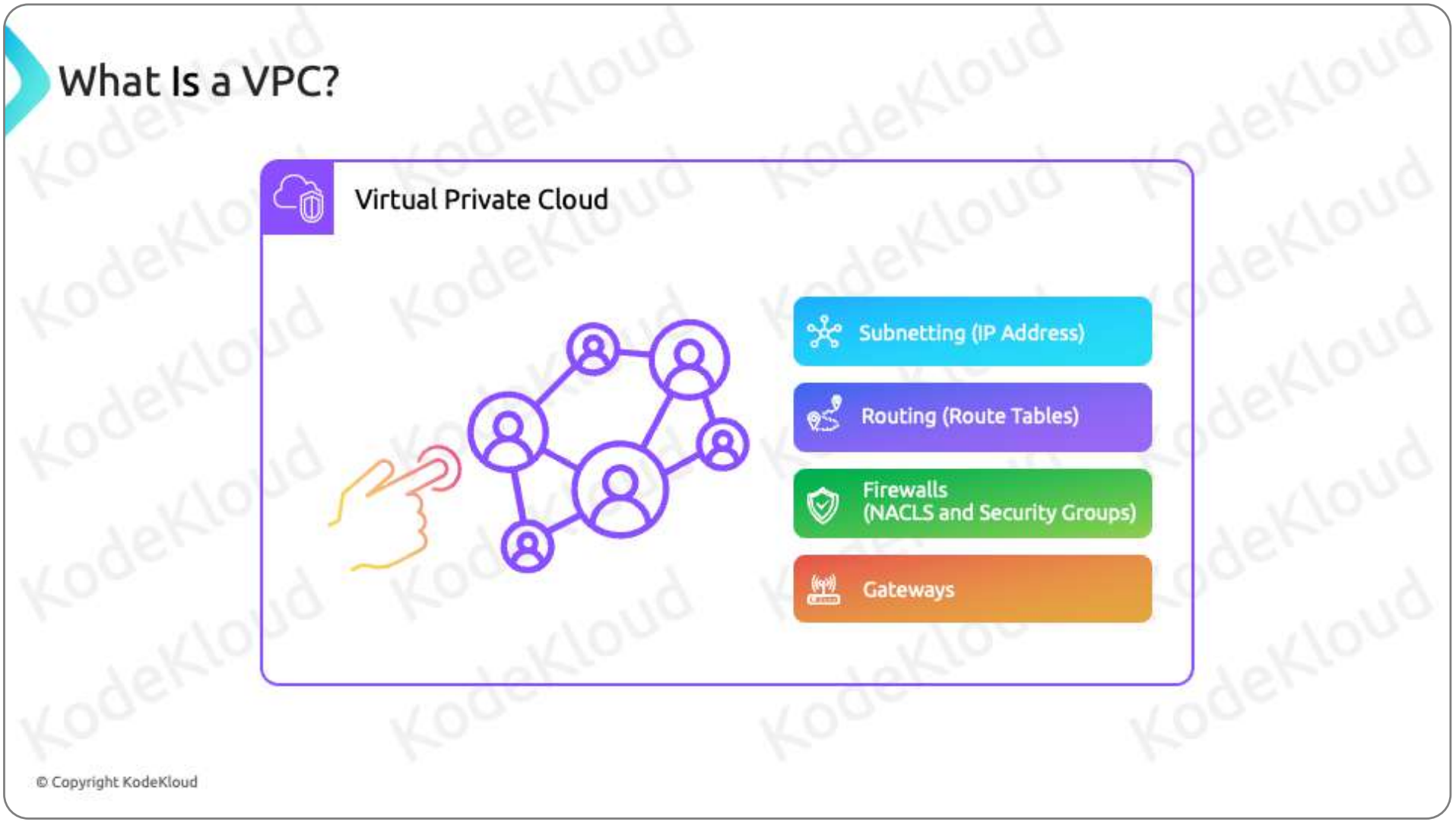


Virtual private cloud (VPC) is a secure, isolated network segment hosted within AWS.

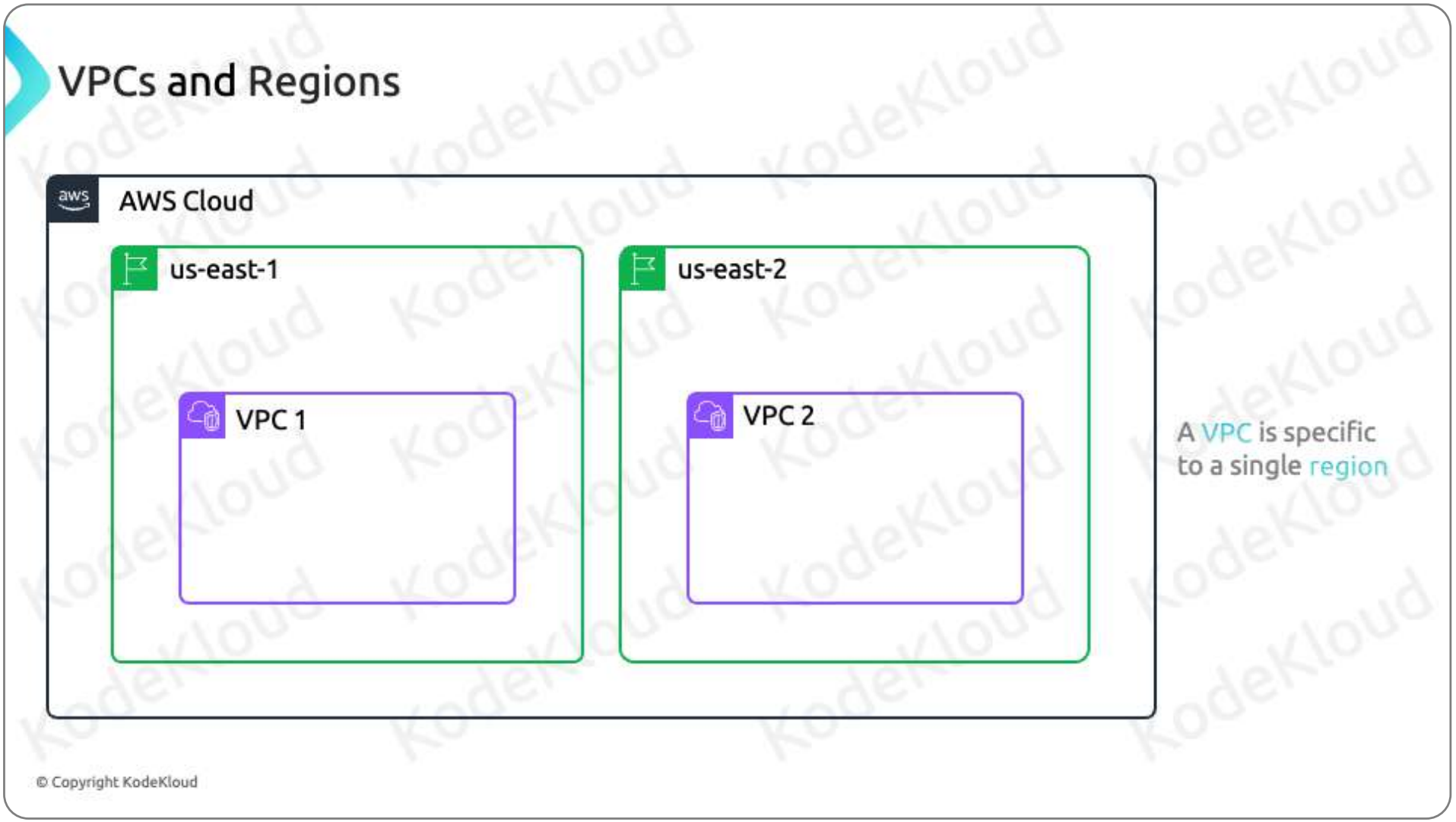
What Is a VPC?



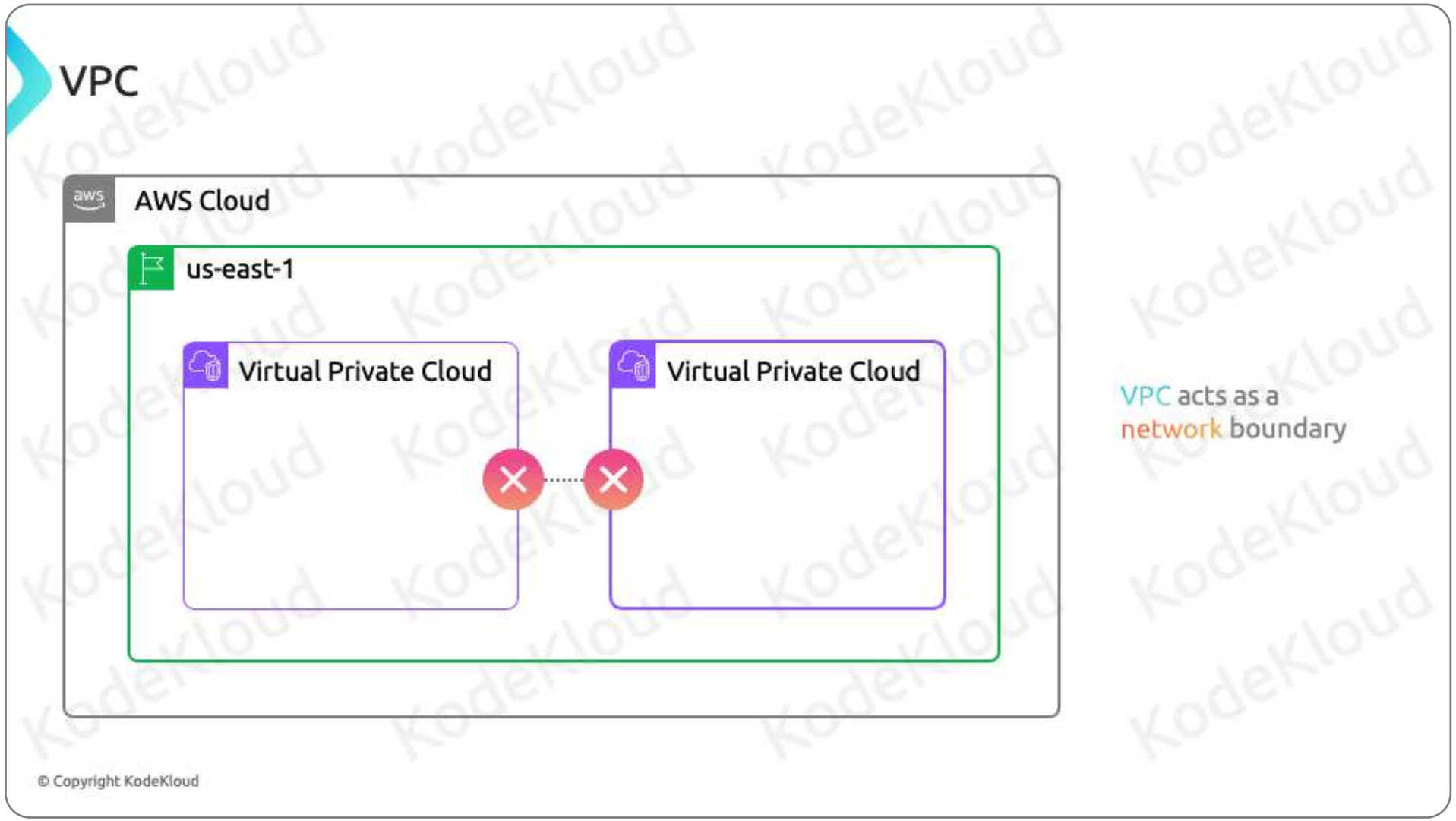
VPC isolates computing resources from other computing resources available in the cloud



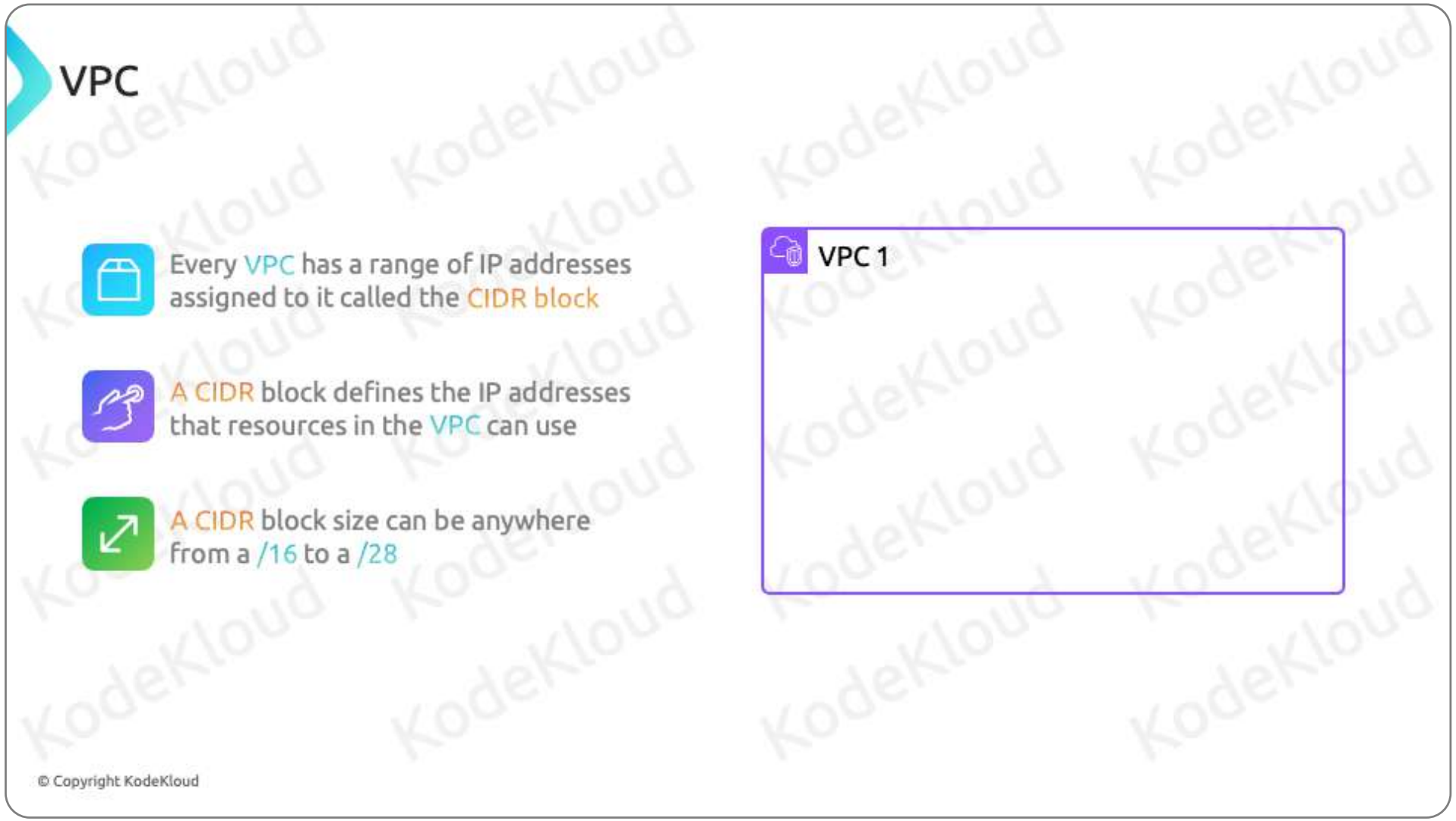
It gives the customer full control of the networking in the cloud



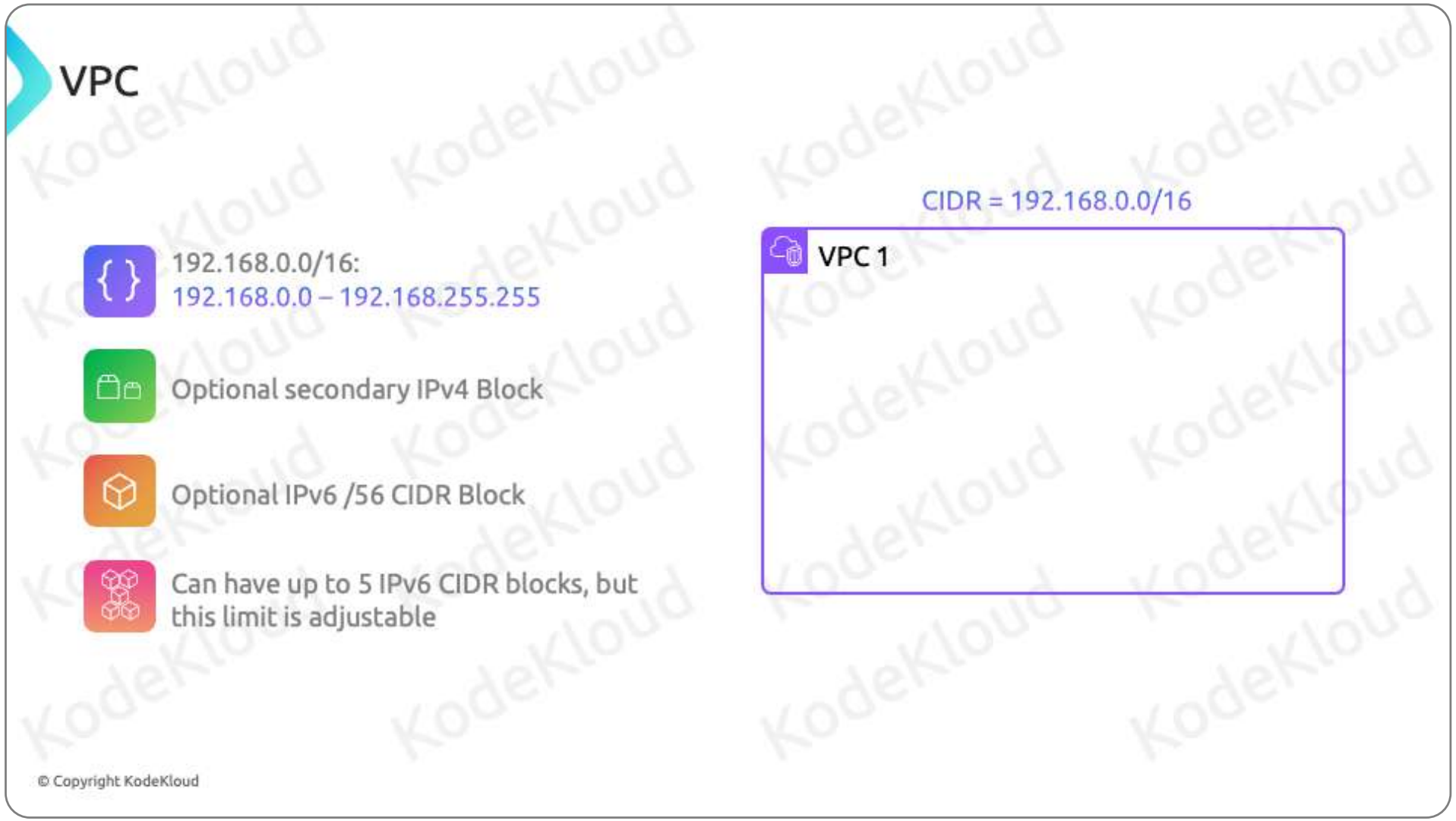
A VPC is specific to a single region



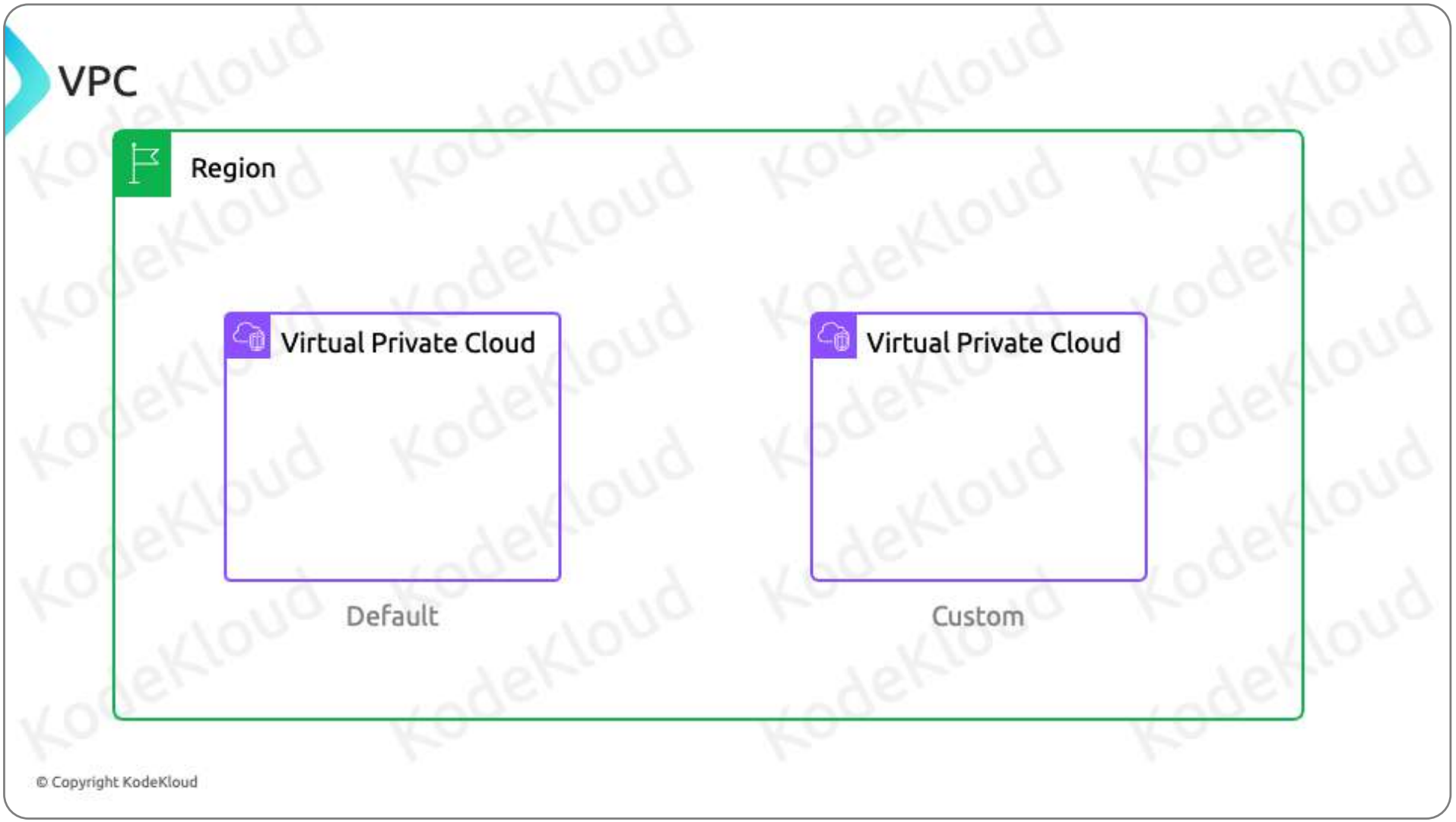
VPC acts as a network boundary



- Optional secondary ipv4 block
- Optional ipv6 /56 CIDR Bock
    - Can have up to 5 ipv6 CIDR blocks but this limit is adjustable

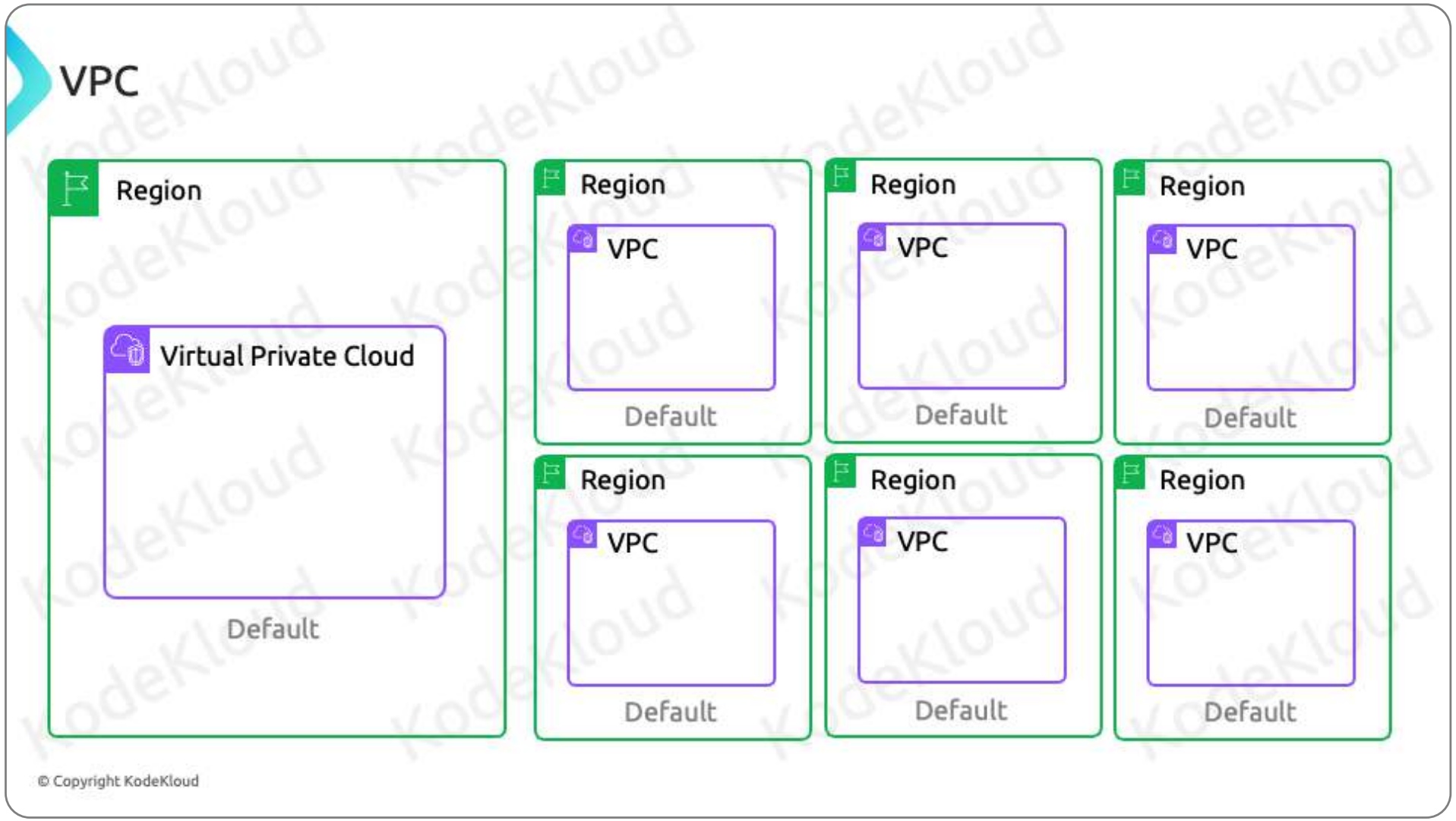


- Optional secondary ipv4 block
- Optional ipv6 /56 CIDR Bock
    - Can have up to 5 ipv6 CIDR blocks but this limit is adjustable

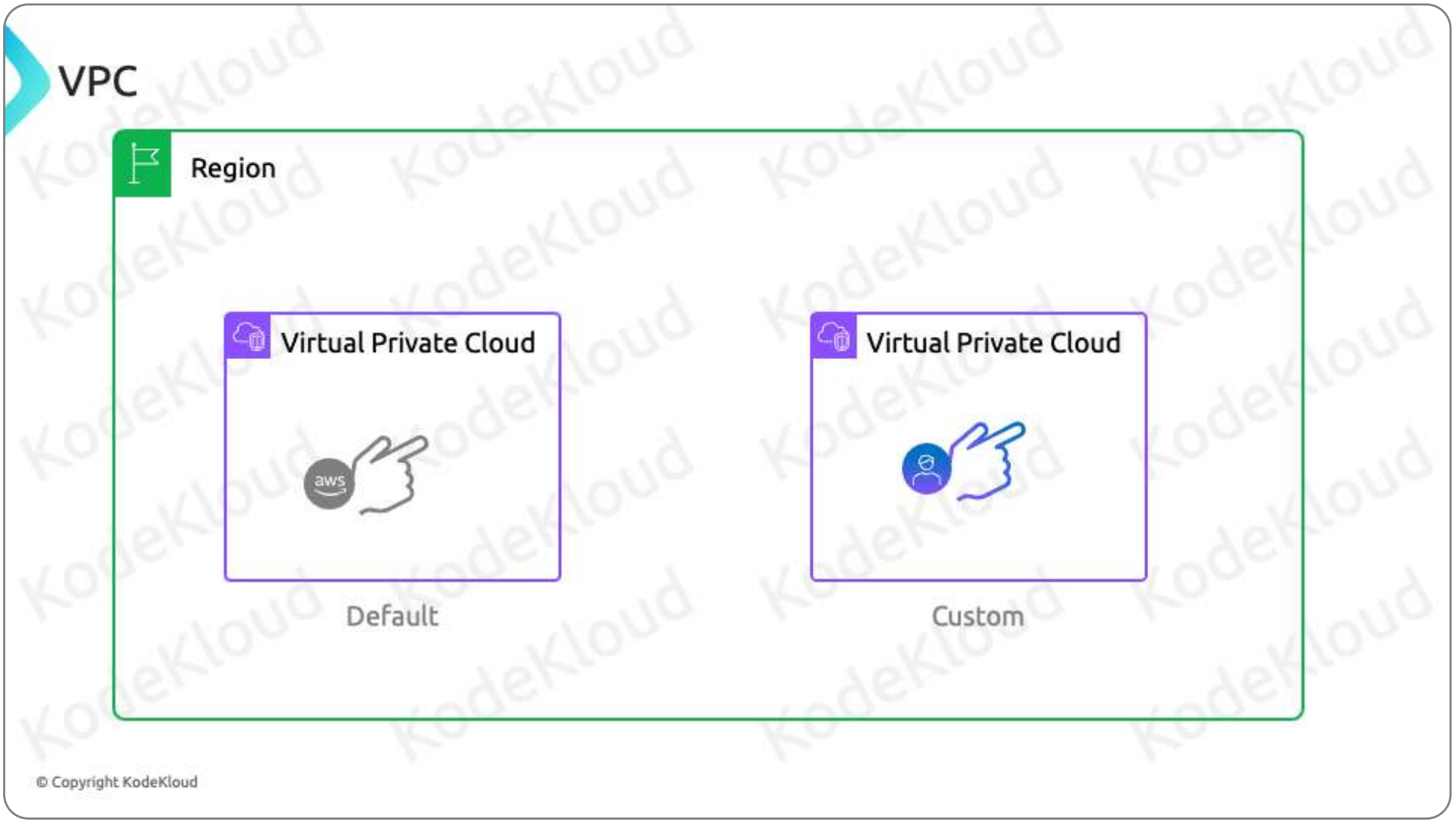


VPCs are of two types

- Default
- Custom



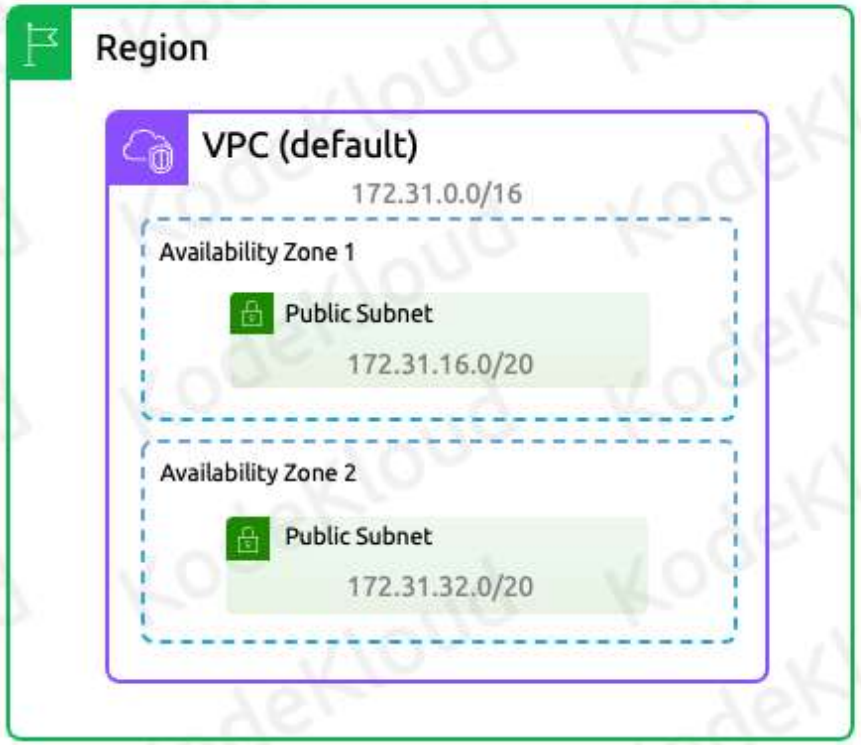
Every account has a default VPC in each region



- Default VPCs have configurations defined by AWS
- Custom VPCs allow you to create/modify all configurations associated with the VPC

VPC

- One default VPC per region
- /16 IPv4 CIDR block 172.31.0.0/16 (65,536 addresses)
- /20 default subnet in each Availability Zone (4,096 addresses)



Summary

01 VPC isolates computing resources from other computing resources available in the cloud

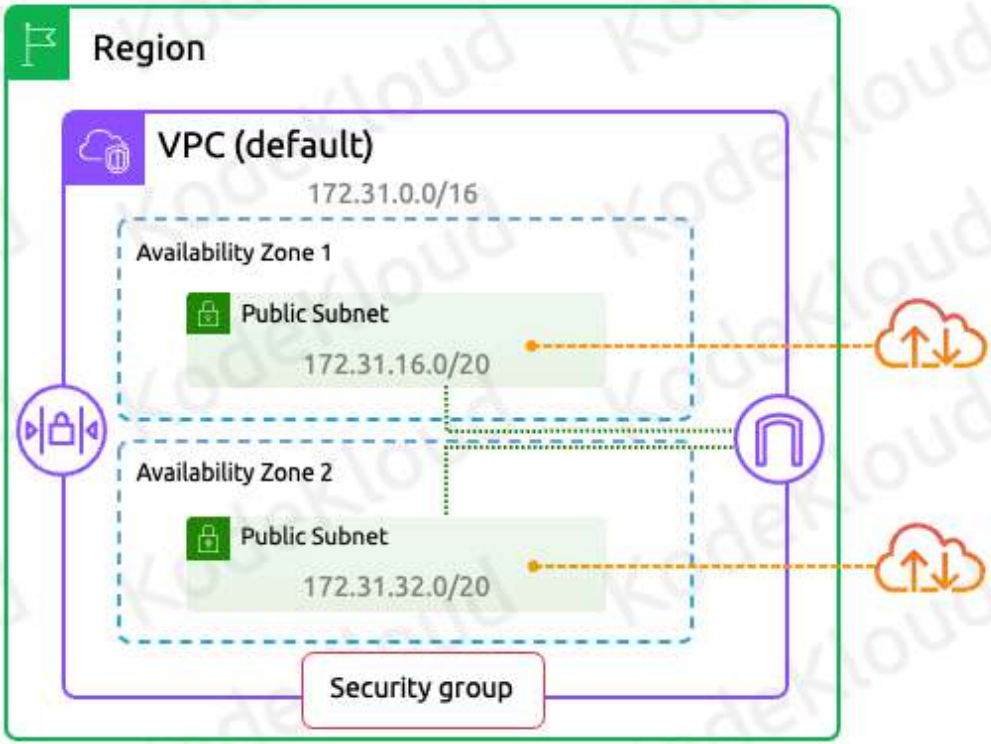
02 VPCs are isolated to a region

03 VPC CIDR block defines the IP addresses a VPC can use

04 VPCs can have optional secondary IPv4 CIDR block as well as IPv6 CIDR block

Default VPC

- Internet gateway attached to the VPC
- A route that points all traffic (0.0.0.0/0) to the internet gateway
- Devices in these default subnets will be accessible from the internet
- Default Security Group
- Default Network Access Control List



Summary

05 Every region has a Default VPC with default subnets, Security Groups, and NACLs

06 The CIDR block for the Default is 172.31.0.0/16

07 The Default VPC and its subnets have outbound access to the internet by default

08 One default subnet in each Availability Zone

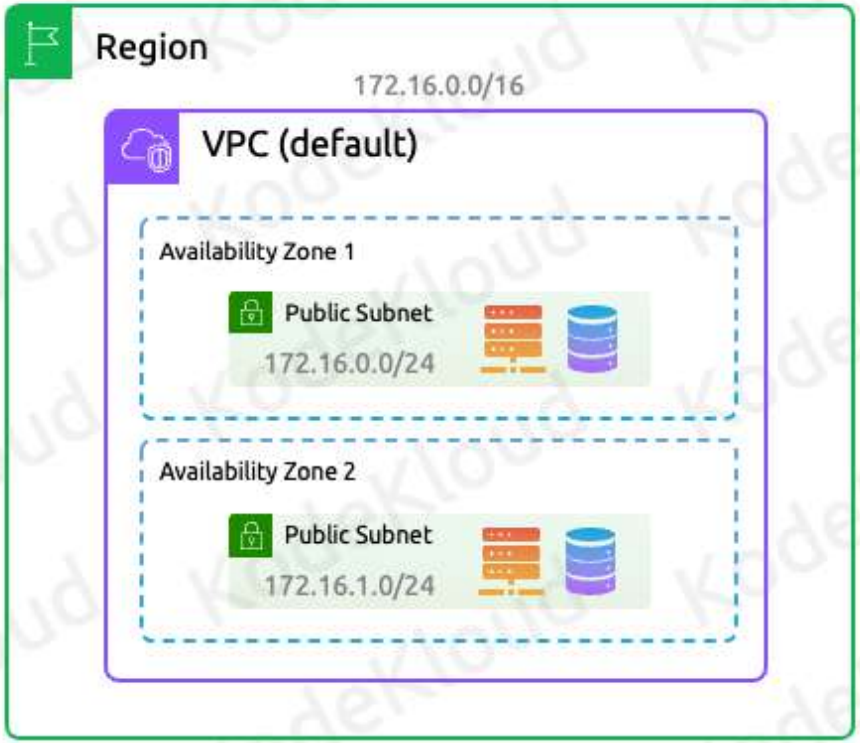
Service Section

The Power of Core Networking

Subnets

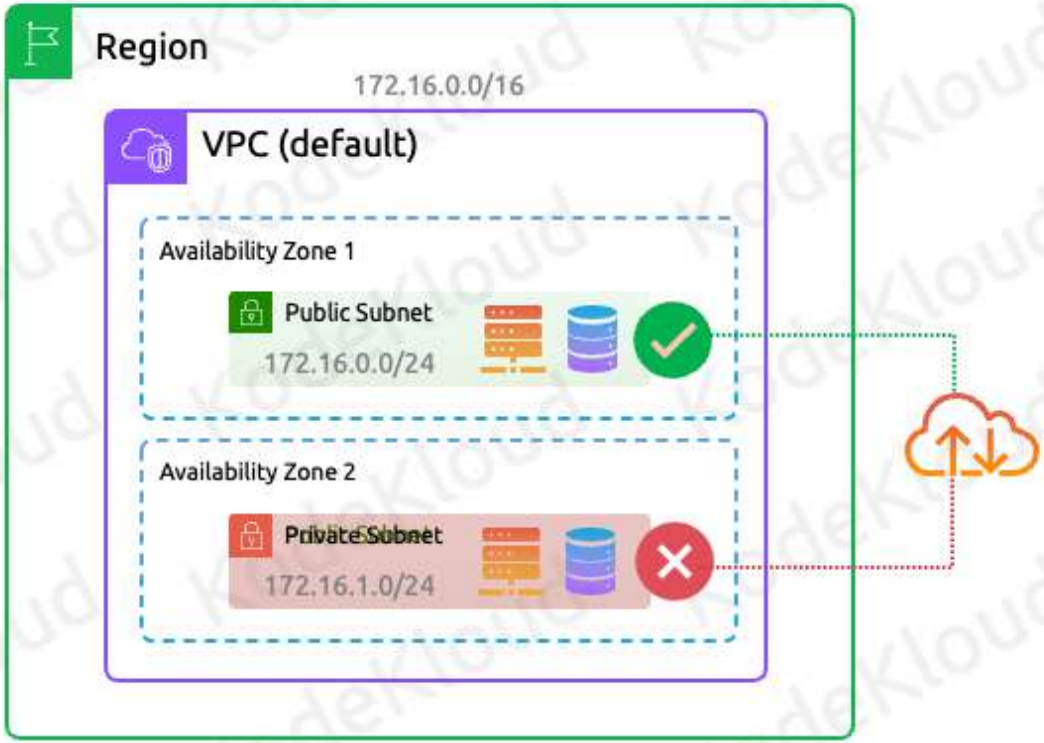
Subnets

- Subnets are groups of IP addresses in your VPC
- A subnet resides within a single Availability Zone



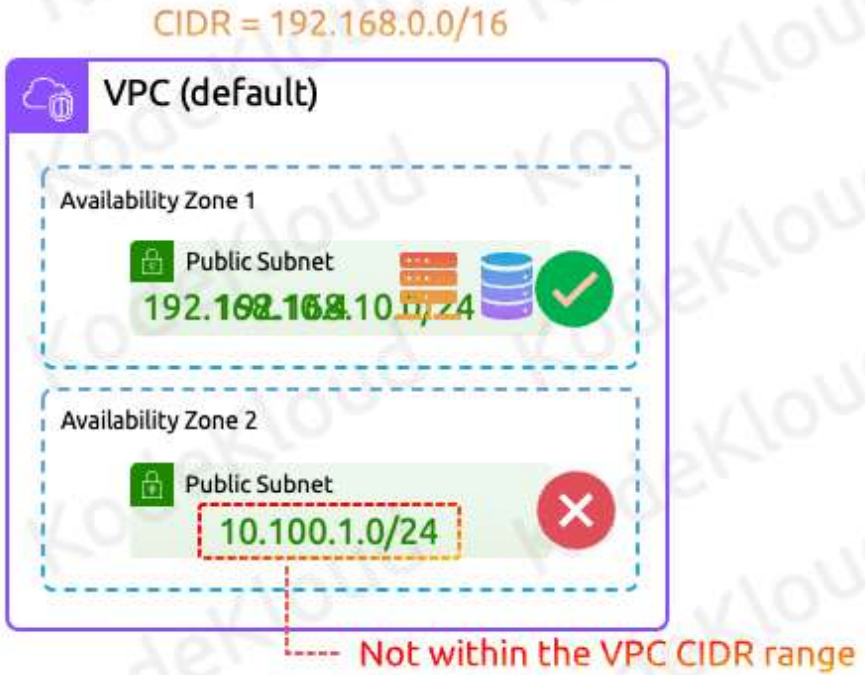
Subnets

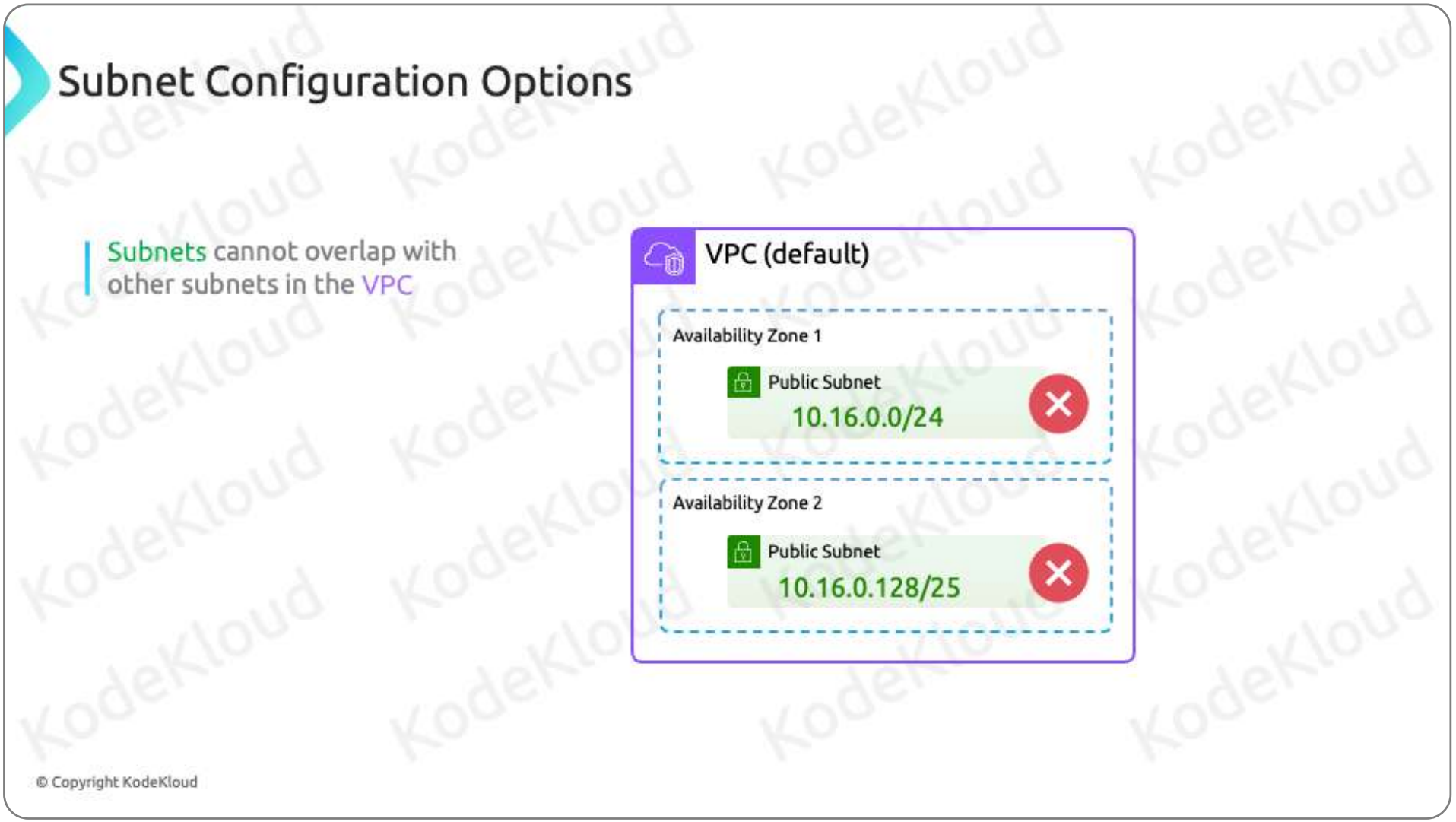
- Subnets are groups of IP addresses in your VPC
- A subnet resides within a single Availability Zone
- Subnets can be made public or private to allow external access to resources within them



Subnets

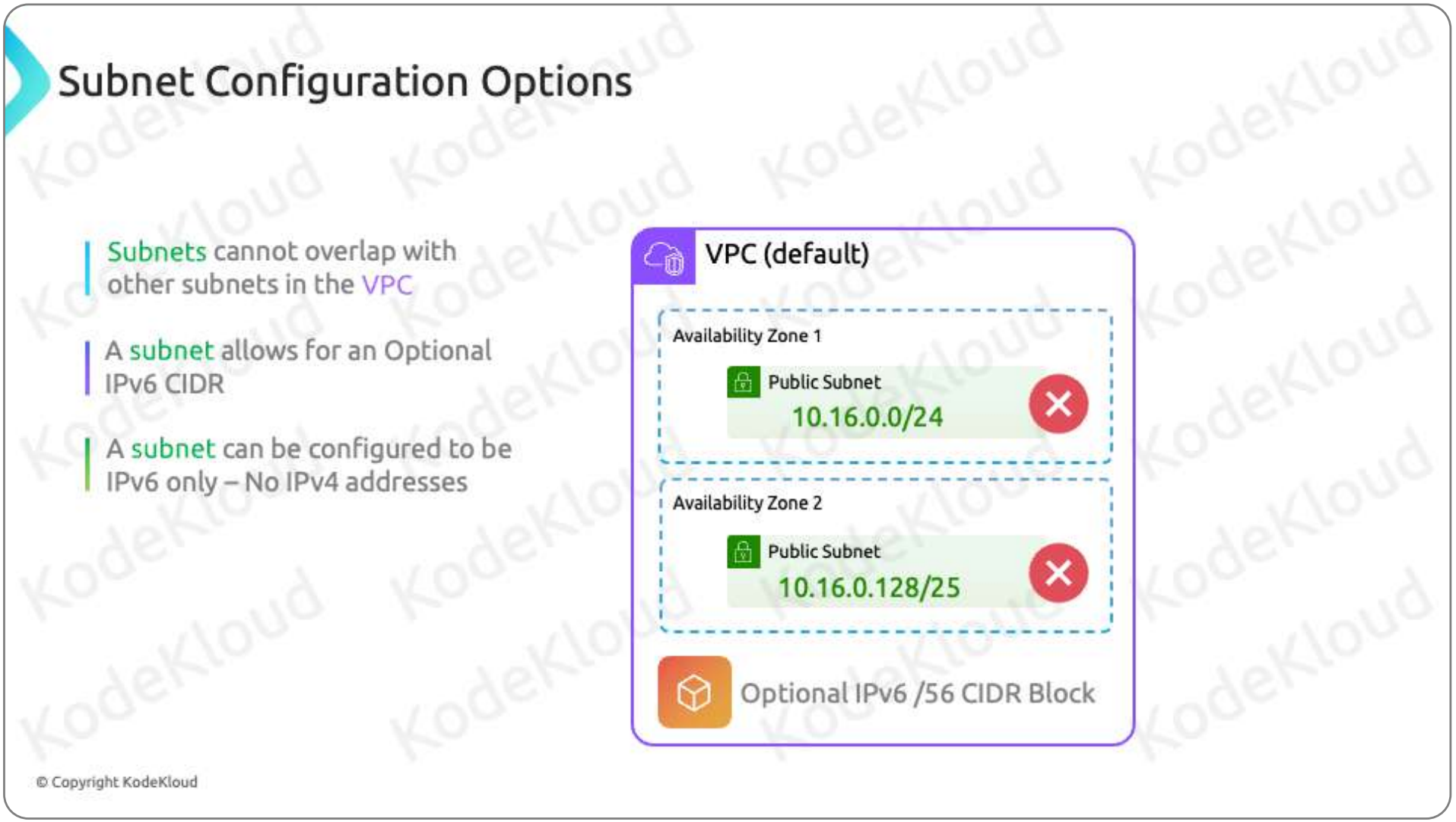
- Subnets within a VPC must be within the CIDR range
- A subnet block size must be between a /16 and a /28
- The first 4 IP addresses of a subnet are reserved and cannot be used
  - **192.168.10.0** (Network address)
  - **192.168.10.1 – 192.168.10.3** (for AWS)
    - 192.168.10.1 (VPC Router)
    - 192.168.10.2 (DNS)
    - 192.168.10.3 (for Future Use)
- The last IP address of a subnet (192.168.10.255) is reserved as the broadcast address





subnets Cannot overlap with other subnets in the VPC (example would be if subnet1 was 10.16.0.0/24 and subnet2 was 10.16.1.128/25 these overlap and would not be allowed)
Subnets allow for an Optional IPv6 CIDR – Only available if the corresponding VPC has an associated IPv6 CIDR block
Subnets can be configured to be IPv6 only – No ipv4 addresses
Subnets can communicate with other subnets in the VPC
Auto Assign Public IPv4/IPv6 ip address in addition to the private address [design team show a server automatically getting

an ip like 10.0.0.4]

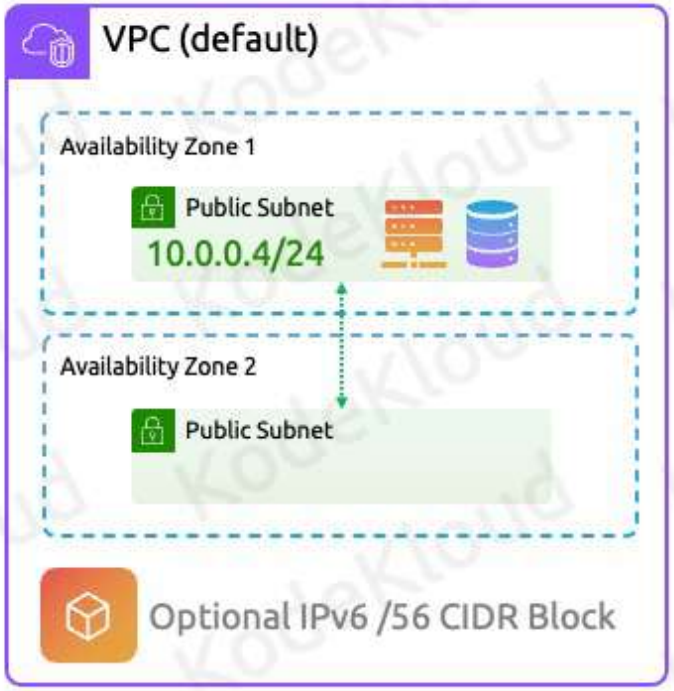


subnets Cannot overlap with other subnets in the VPC (example would be if subnet1 was 10.16.0.0/24 and subnet2 was 10.16.1.128/25 these overlap and would not be allowed)
Subnets allow for an Optional IPv6 CIDR – Only available if the corresponding VPC has an associated IPv6 CIDR block
Subnets can be configured to be IPv6 only – No ipv4 addresses
Subnets can communicate with other subnets in the VPC
Auto Assign Public IPv4/IPv6 ip address in addition to the private address [design team show a server automatically getting

an ip like 10.0.0.4]

Subnet Configuration Options

- Subnets cannot overlap with other subnets in the VPC
- A subnet allows for an Optional IPv6 CIDR
- A subnet can be configured to be IPv6 only – No IPv4 addresses
- Subnets can communicate with other subnets in the VPC
- Auto-assign Public IPv4/IPv6 IP address in addition to the private address



subnets Cannot overlap with other subnets in the VPC (example would be if subnet1 was 10.16.0.0/24 and subnet2 was 10.16.1.128/25 these overlap and would not be allowed)
Subnets allow for an Optional IPv6 CIDR – Only available if the corresponding VPC has an associated IPv6 CIDR block
Subnets can be configured to be IPv6 only – No ipv4 addresses
Subnets can communicate with other subnets in the VPC
Auto Assign Public IPv4/IPv6 ip address in addition to the private address [design team show a server automatically getting

an ip like 10.0.0.4]

Summary

01 Subnets are a range of IP addresses within a VPC

02 Subnets reside within a single Availability Zone

03 Subnets can be made public/private using Internet Gateways and Nat Gateways

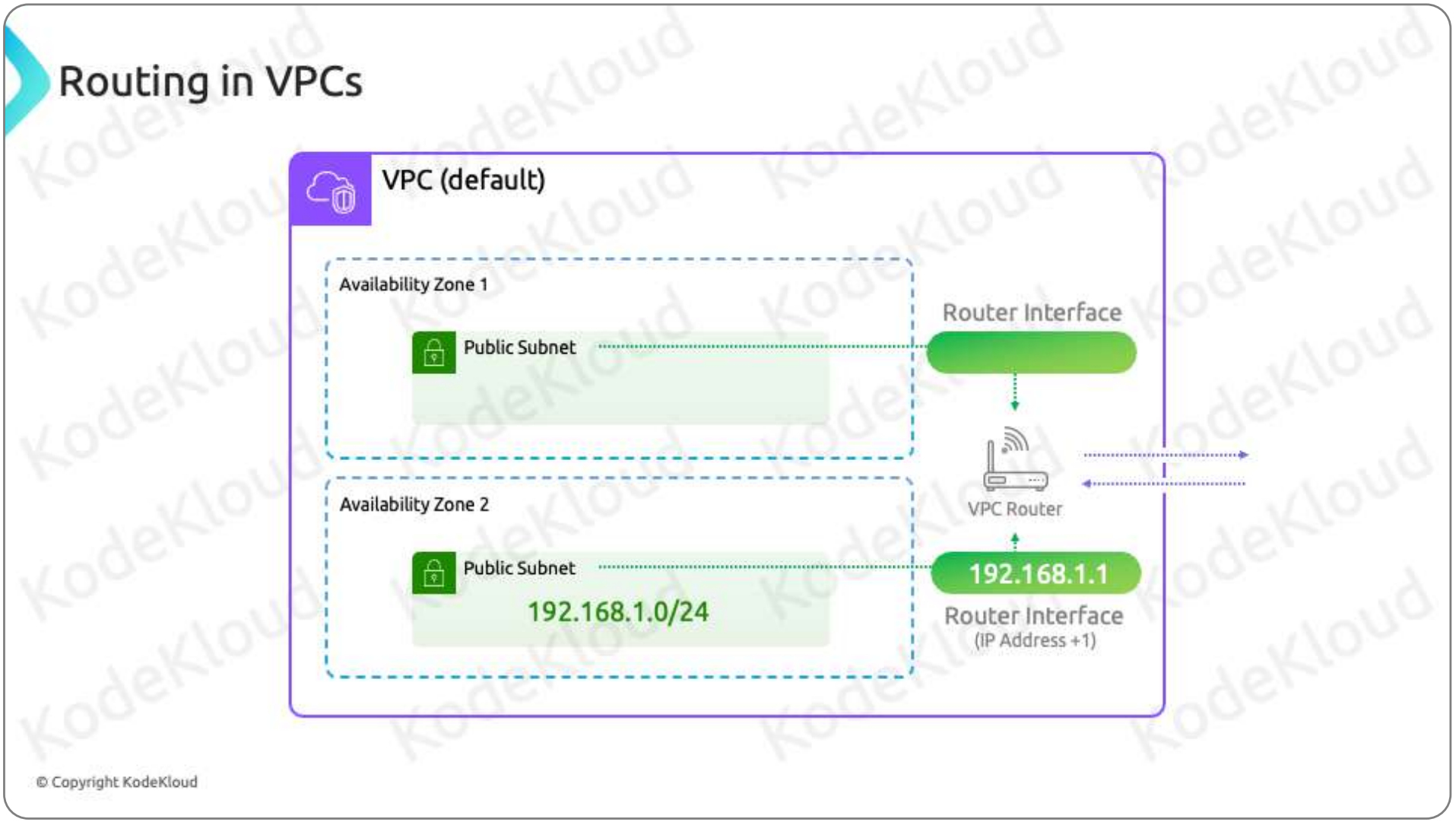
04 Subnets can be configured for IPv4 and/or IPv6

05 Subnets cannot overlap with other subnets in a VPC

Service Section

The Power of Core Networking

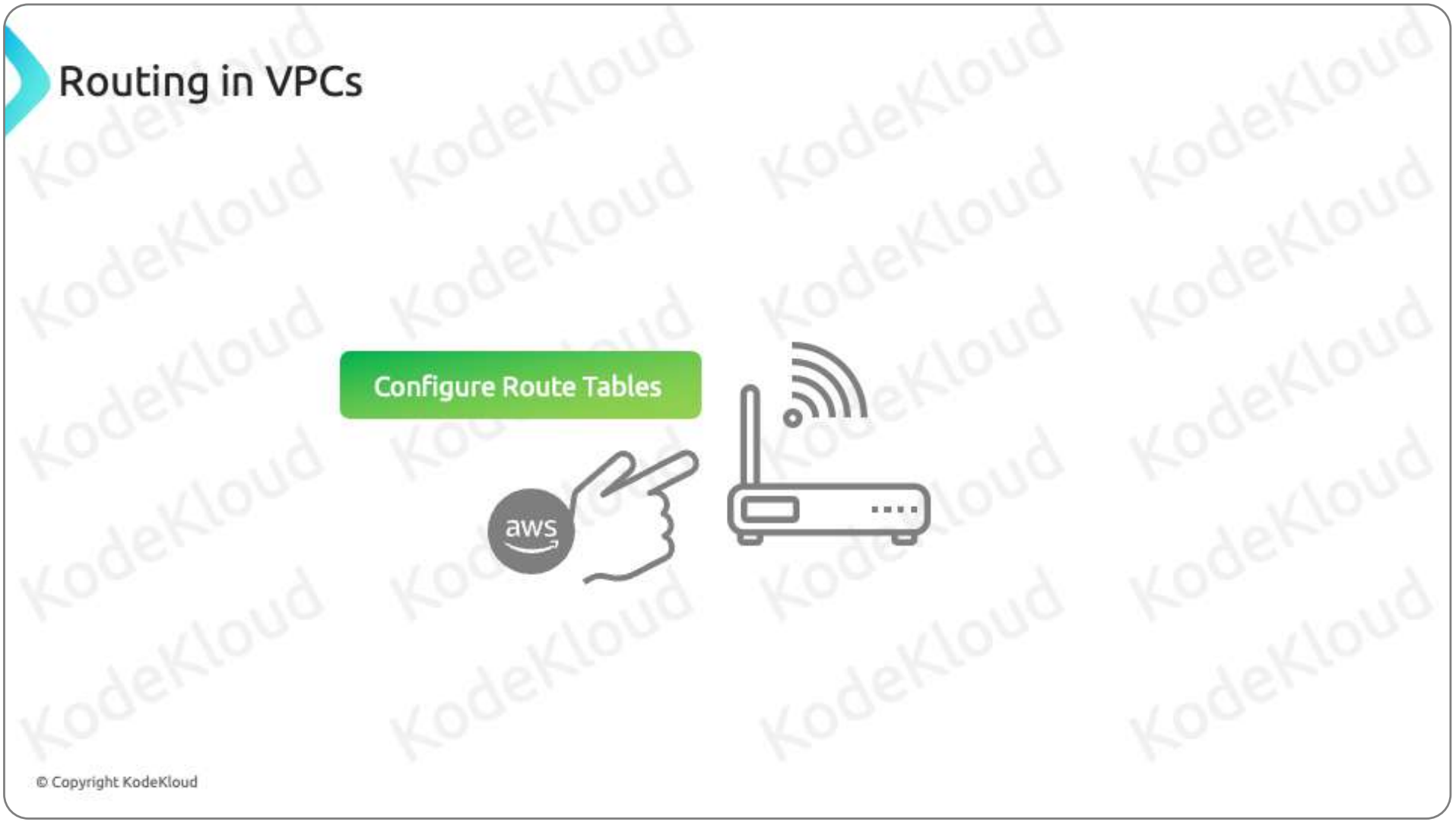
Routing VPCs



Every VPC has a VPC Router
The router has an interface in every subnet of the VPC and can be reachable form the network + 1 address of each subnet (example 192.168.1.0/24 the router would be 192.168.1.1)

The purpose of the router is to route traffic between subnets as well as in and out of the VPC.
Just like a physical router in a datacenter, AWS does allow you to control the router by configuring route tables which

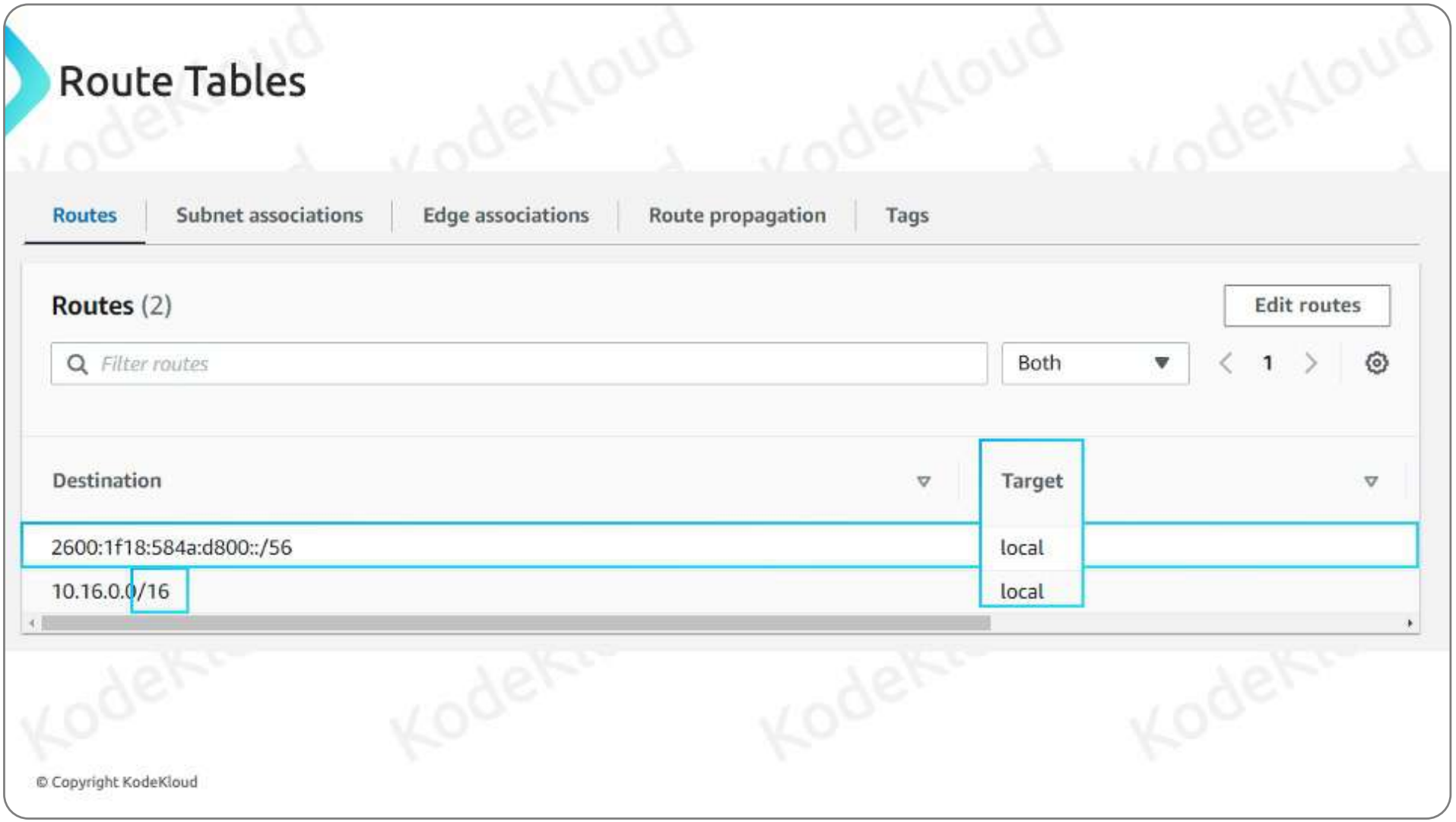
determine where network traffic will get sent



Every VPC has a VPC Router
The router has an interface in every subnet of the VPC and can be reachable form the network + 1 address of each subnet (example 192.168.1.0/24 the router would be 192.168.1.1)

The purpose of the router is to route traffic between subnets as well as in and out of the VPC.
Just like a physical router in a datacenter, AWS does allow you to control the router by configuring route tables which

determine where network traffic will get sent



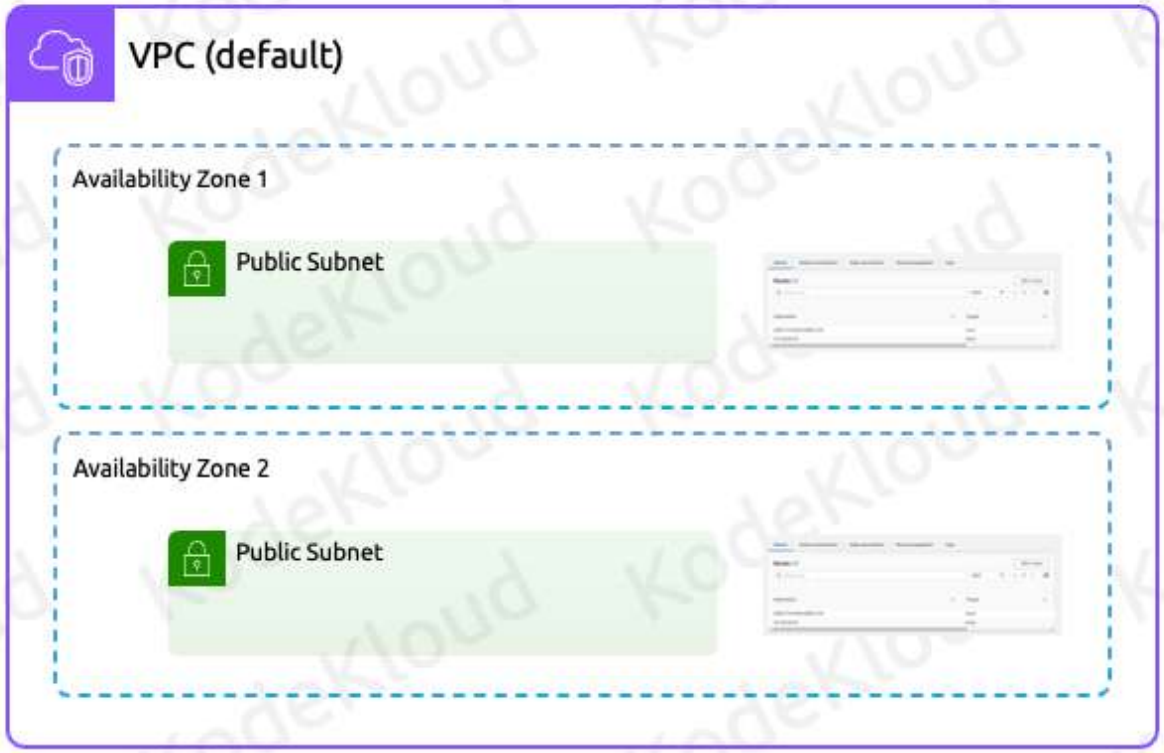
A route table is a set of rules the router uses to forward network traffic. Each rule is referred to as a route.
The way that the AWS router works is no different than any traditional router. The router will take a look at a packet leaving the subnet and it will check only the destination ip of the packet and find a matching route by checking the destination column.

Now a packet could match multiple destination, in this case the router will look for the destination that has the higher prefix

number which means it is more specific(/24 is better than /16 as it is longer) and select that.
Once a route is selected for the packet the VPC router will forward the network packet to the Target field of that specific column. All route tables have exactly one route by default which is the local route. This route will match all traffic from within the vpc as the destination prefix is the vpc CIDR. So any traffic originating within the VPC destined to another device in the same vpc will match this local route

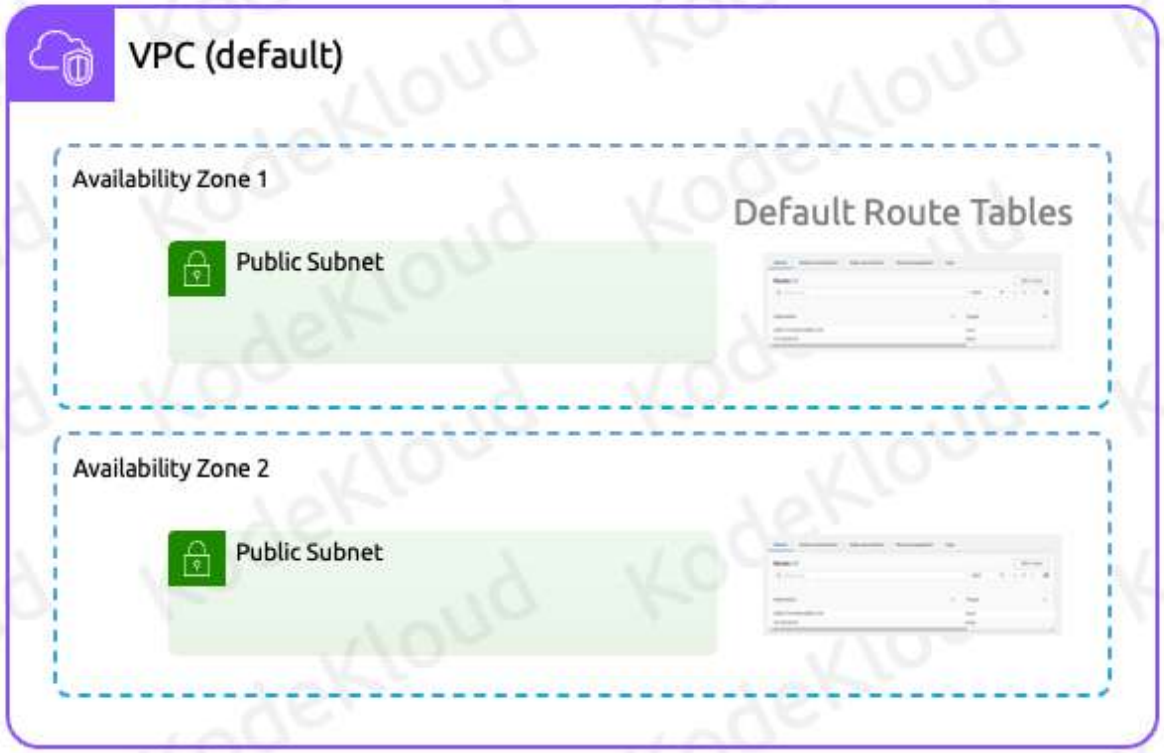
If vpc has ipv6 enabled then it will also have a local route for the ipv6 cidr block of the vpc

Route Tables



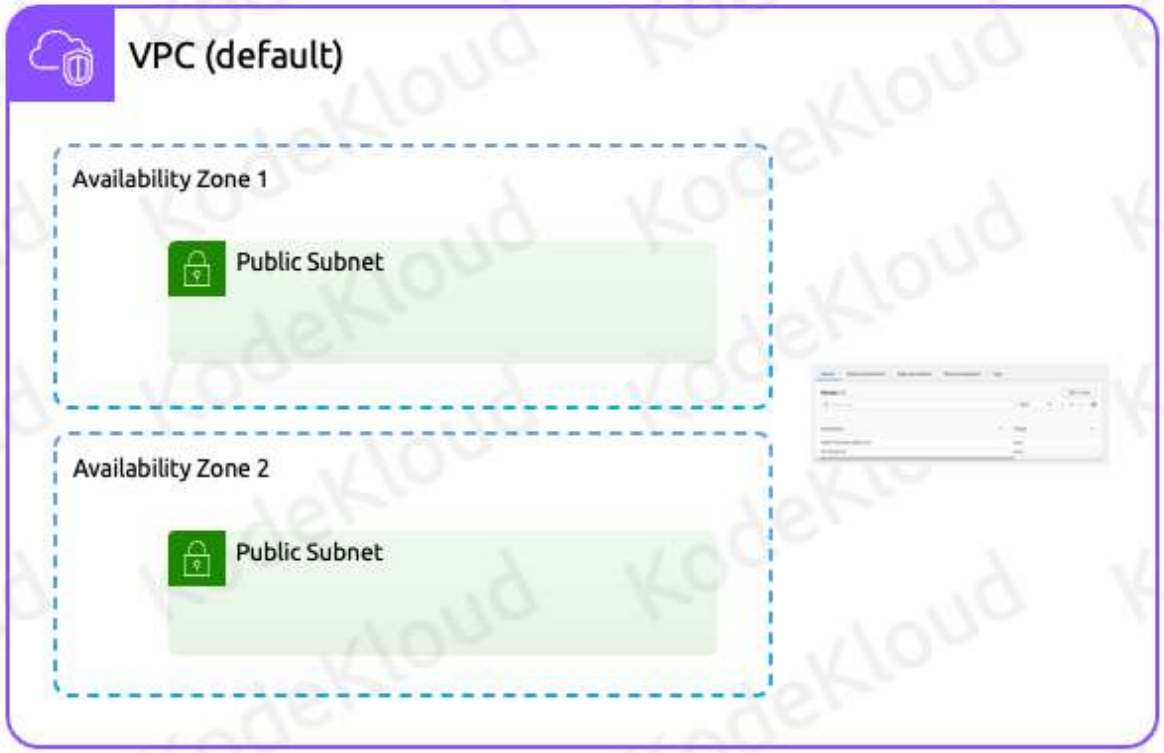
Every subnet is associated with one route-table
Packets leaving the subnet will follow the rules on the route-table associated with the sbunet
Each VPC has one default route-table
Multiple subnets can be associated with a single route-table

Route Tables



Every subnet is associated with one route-table
Packets leaving the subnet will follow the rules on the route-table associated with the sbunet
Each VPC has one default route-table
Multiple subnets can be associated with a single route-table

Route Tables



Every subnet is associated with one route-table
Packets leaving the subnet will follow the rules on the route-table associated with the sbunet
Each VPC has one default route-table
Multiple subnets can be associated with a single route-table

Summary

01 Every VPC has a router which is responsible for routing traffic between subnets as well as in and out of a VPC

02 The router has an interface in every subnet of the VPC and is reachable from the network + 1 address of each subnet

03 Route table is a set of rules the router uses to forward network traffic – each rule is referred to as a route

04 The router looks at the destination IP of a packet leaving the subnet and will find a matching route by checking the destination column

05 Once matched with a destination, the packet will be forwarded to the respective Target column of the route

Summary

06 All route tables have exactly one route by default, which is the local route (2 local routes will be created if IPv6 is enabled for VPC)

07 Every subnet is associated with one route table

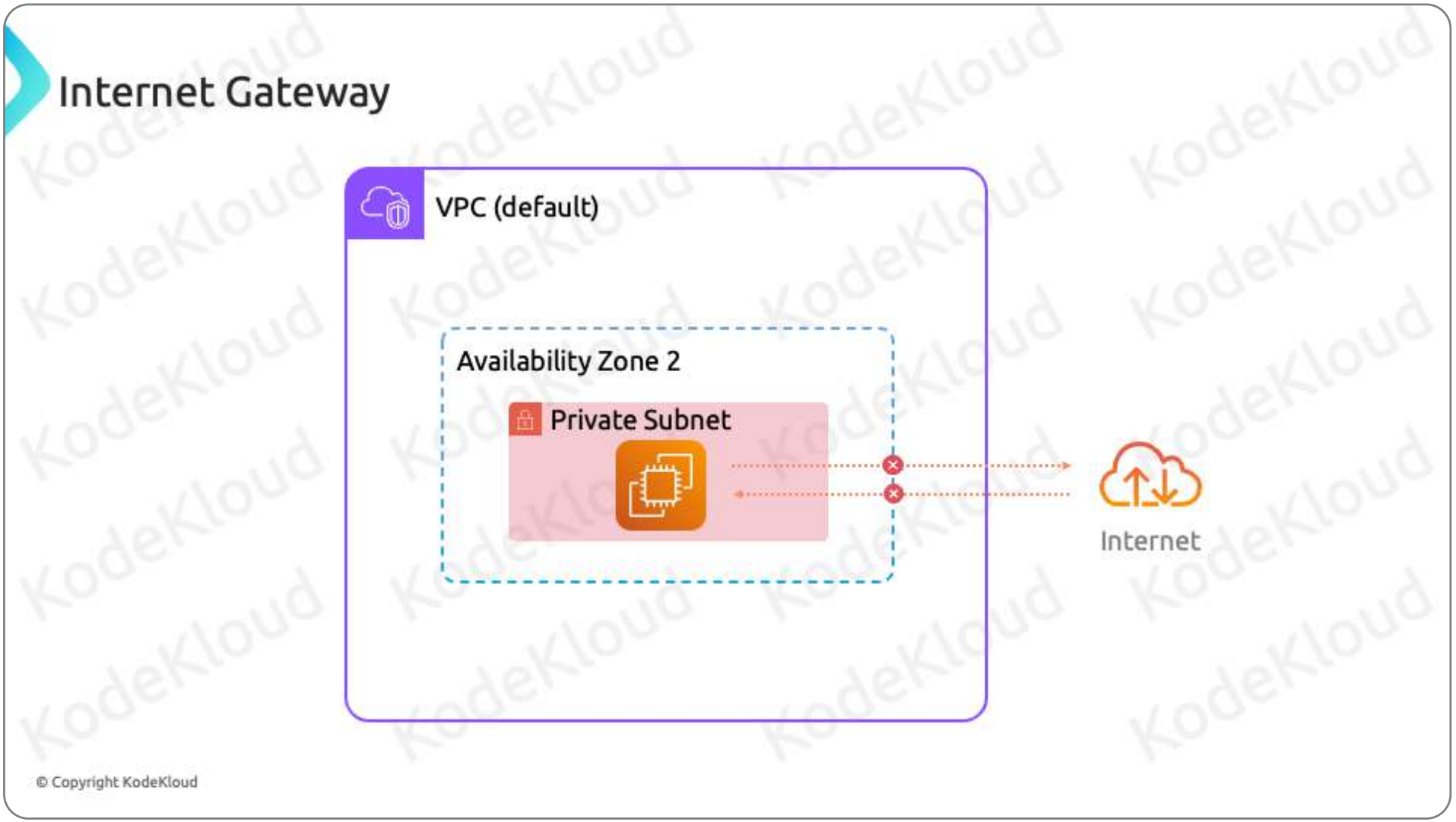
08 Each VPC has one default route table

09 Multiple subnets can be associated with a single route table

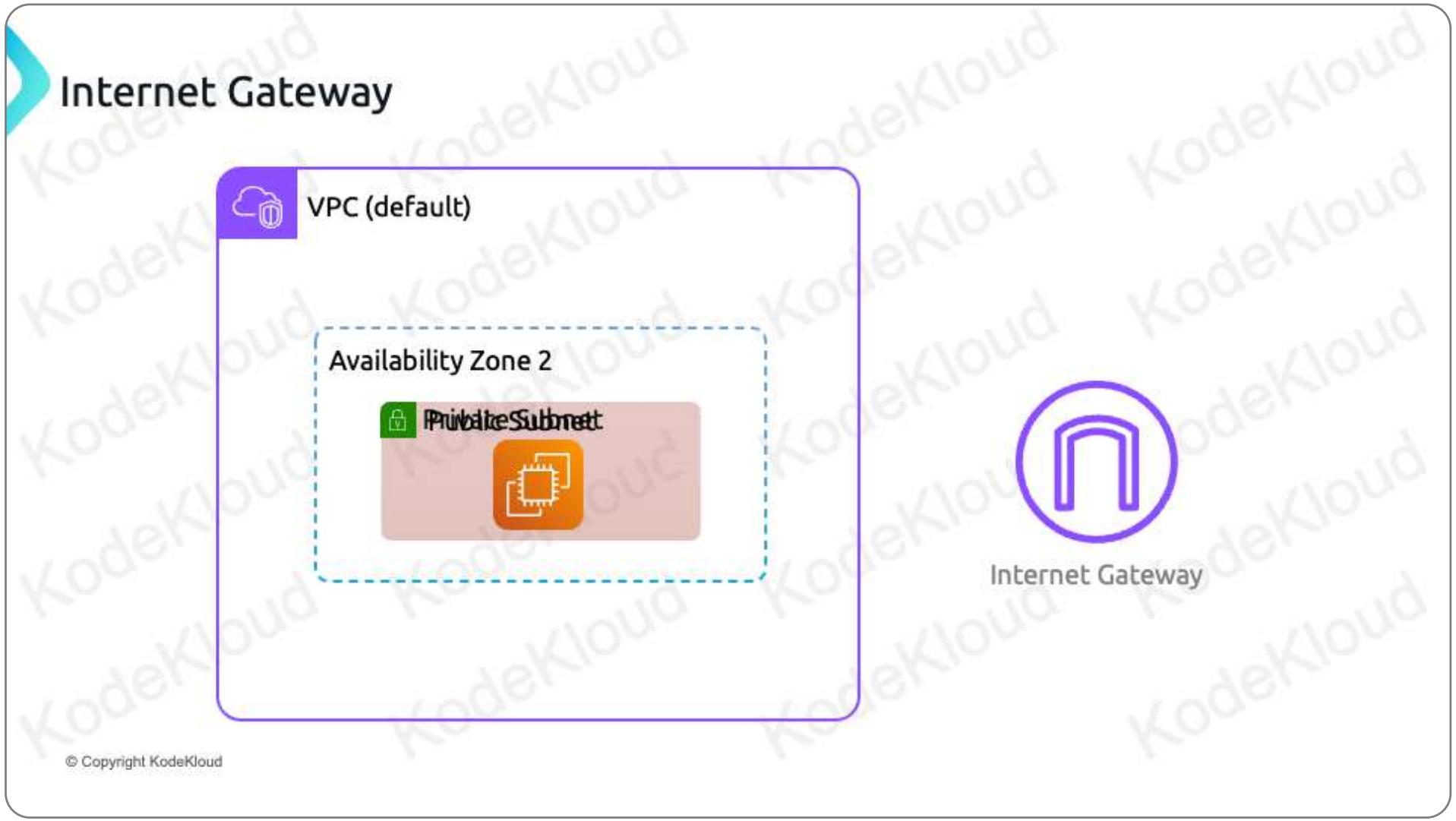
Service Section

The Power of Core Networking

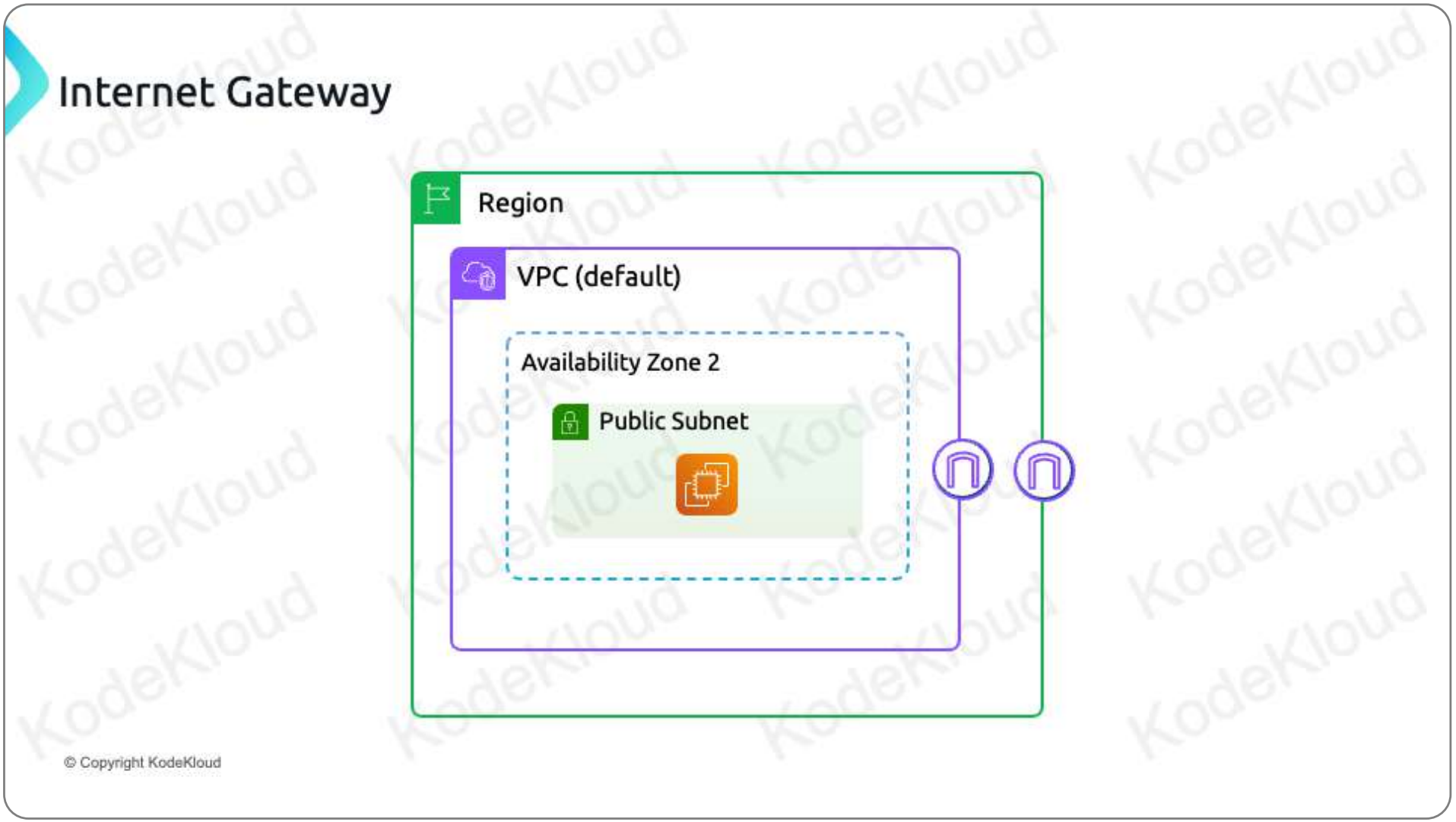
Internet Gateway



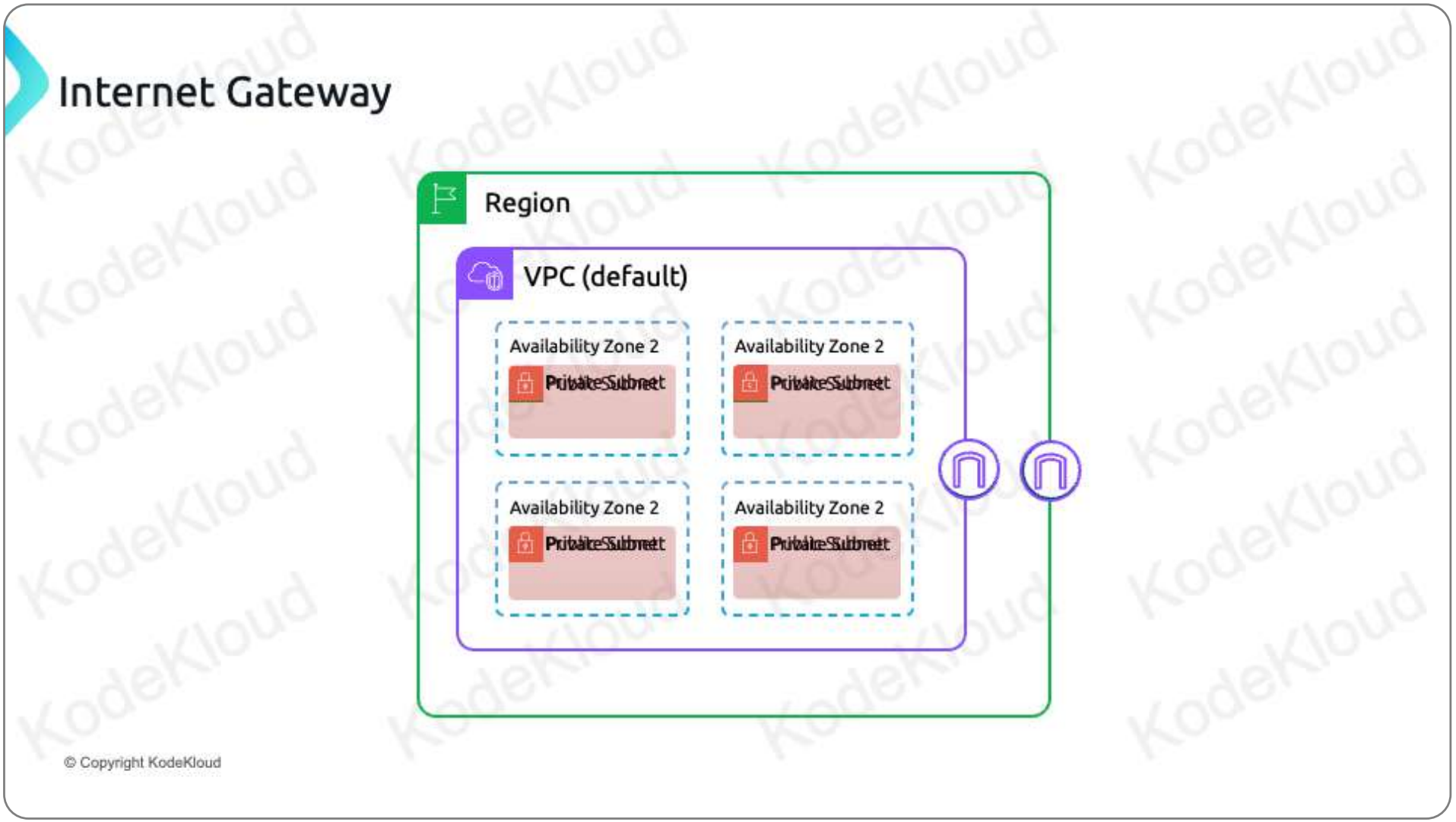
When you create a subnet, by default it will be a private subnet, which means devices in the subnet can't talk to the internet and vice versa.



To make a subnet public we need to make use of Internet Gateways(IGW)

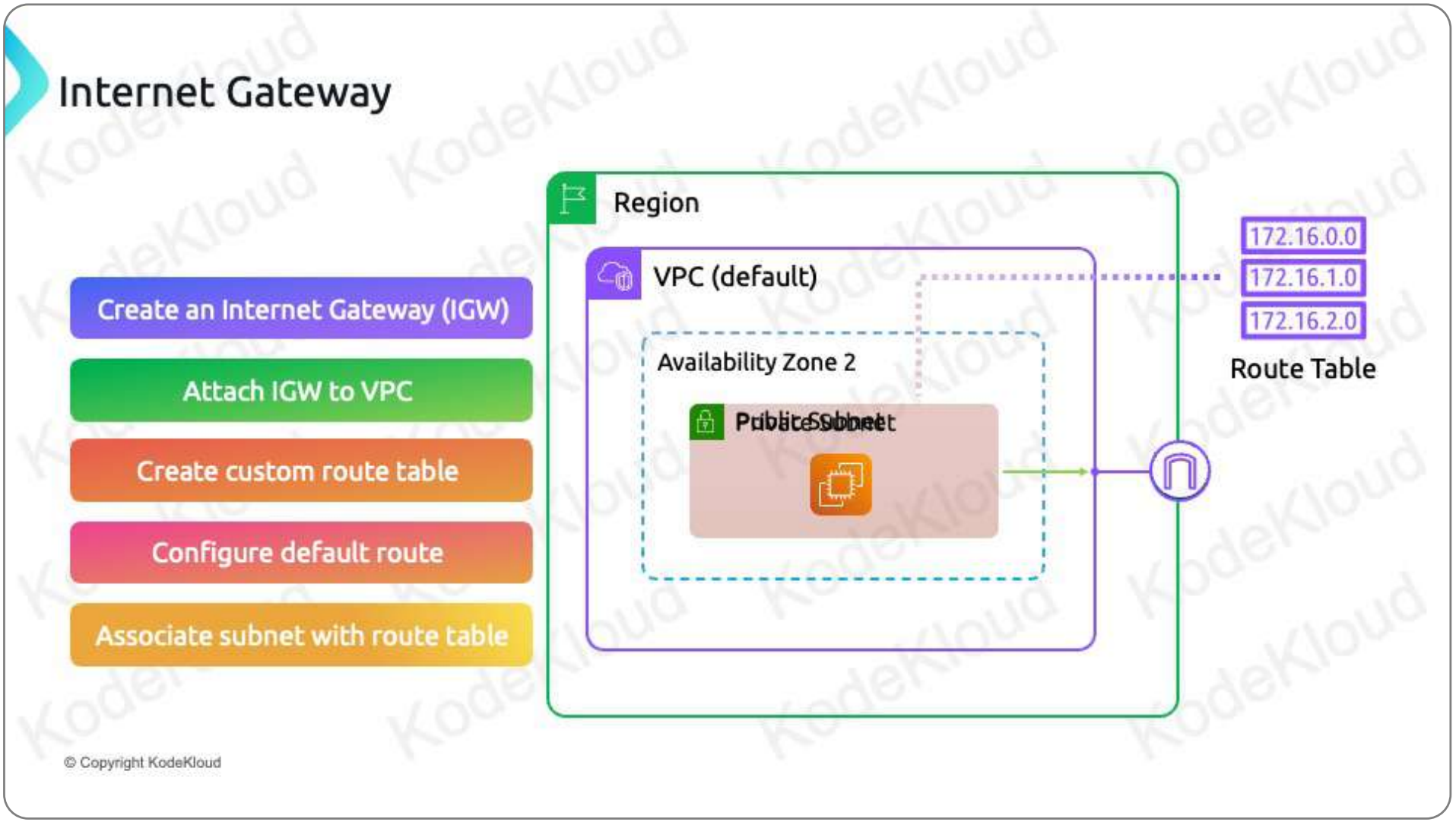


Internet Gateways are attached to a VPC and a region resilient which means they cover all availability zones for the region your vpc is in.

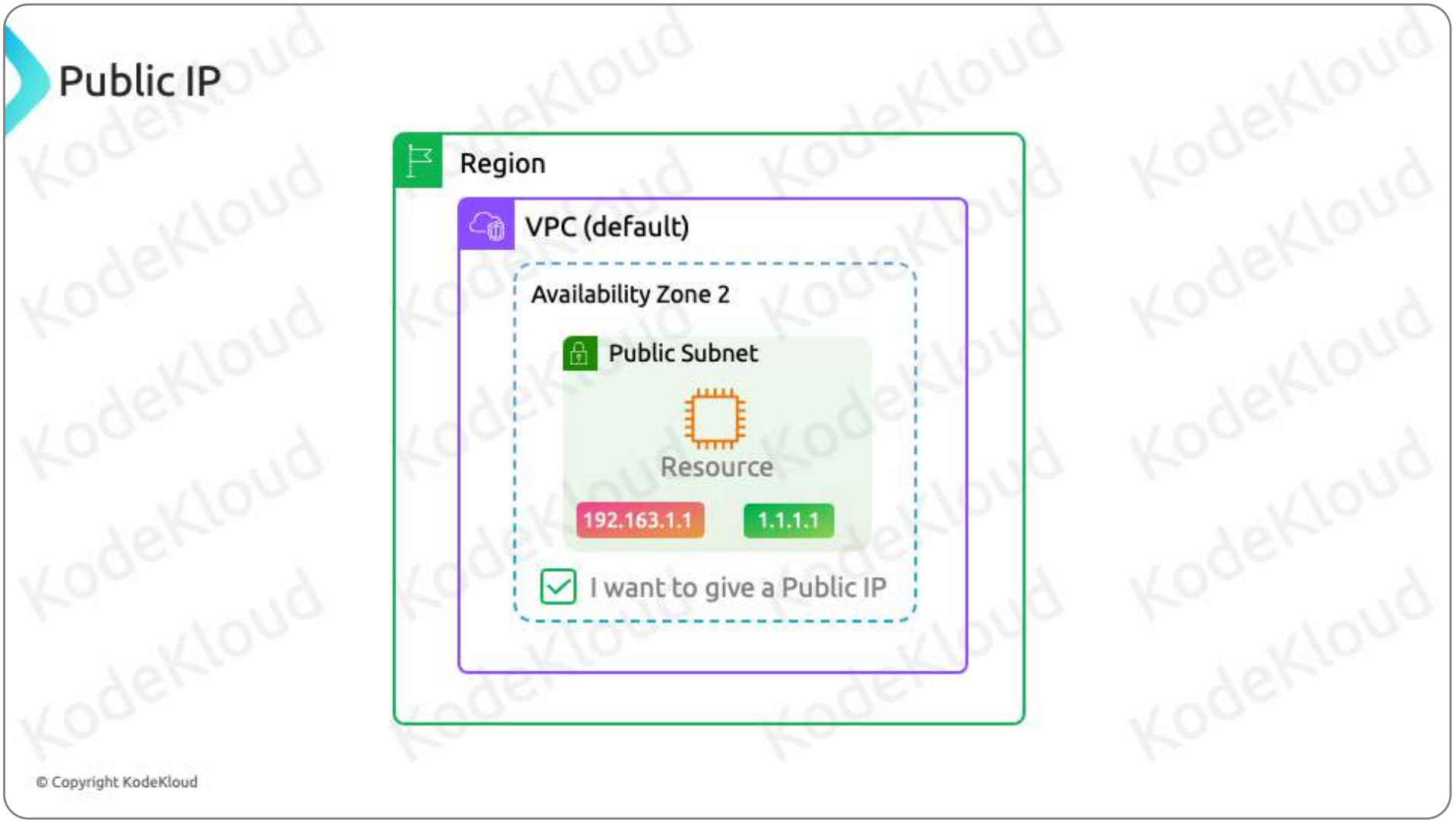


Internet Gateways are attached to a VPC and a region resilient which means they cover all availability zones for the region your vpc is in.

If a VPC has no IGW attached to It then all subnets within the VPC are private.
A VPC can have up to 1 IGW attached to it and an IGW can only be attached to 1 VPC



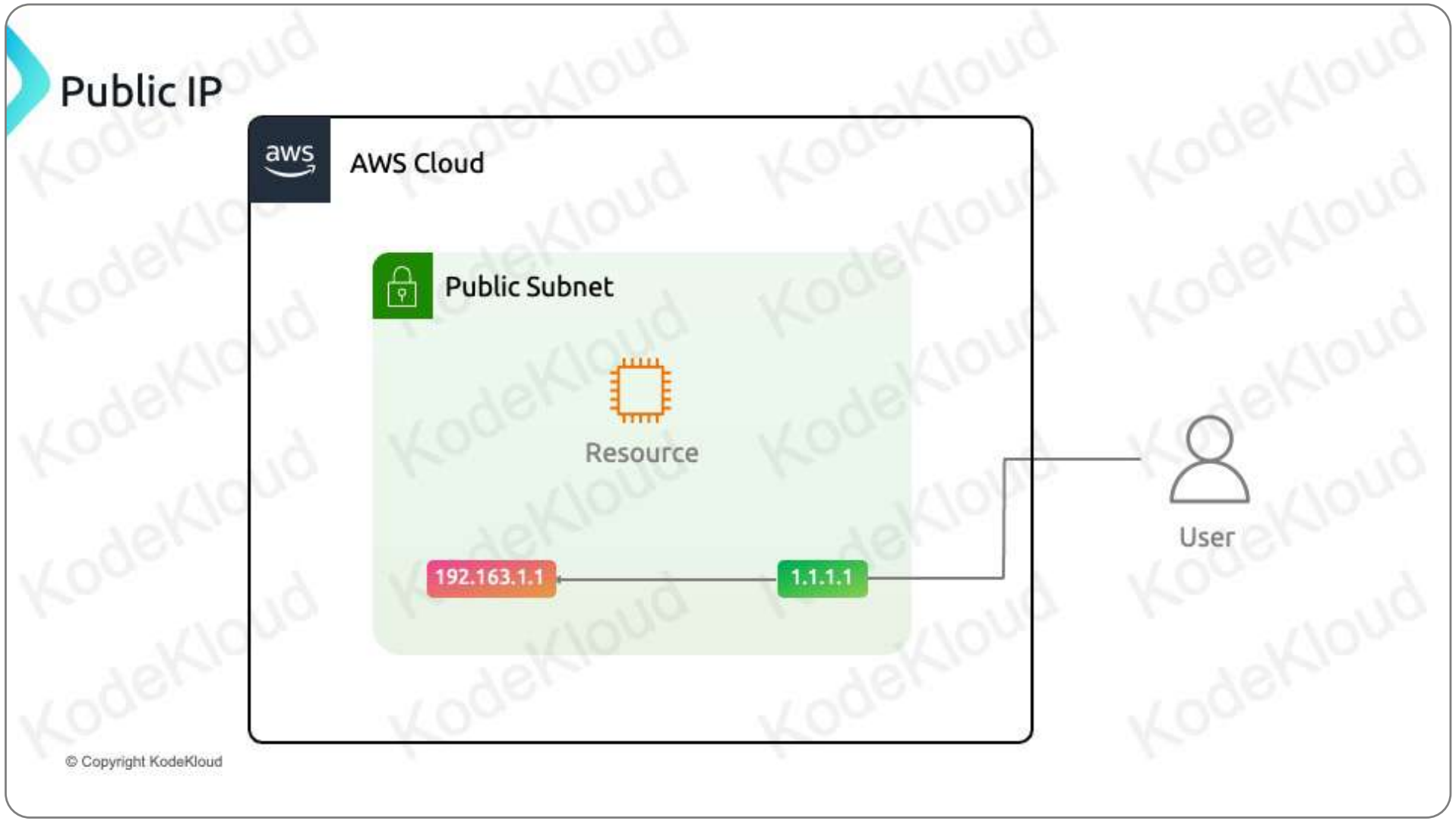
To make a subnet public we have to do the following:
1. create an IGW
2. Attach IGW to VPC
3. Create a custom route-table
4. Configure default route pointing to IGW
5. Associate subnet with route-table



When you deploy a resource onto a public subnet, you still have to select the checkbox that says you want to give it a public IP. By default all resources will be given a private IP that will allow them to communicate internally within AWS. When you click the checkbox to give it a public IP, the server will have both a private IP on the interface and a public ip on that same interface.

And so traffic destined to the public IP on the server map to the private IP on that interface

[design team] Have an ec2 instance with a private ip of 192.168.1.1. Then show that when the user enabled the public-ip checkbox the server also gets a public ip of 1.1.1.1. Then show when users send a request to 1.1.1.1, once the request arrives in aws it then gets mapped to 192.168.1.1 the private IP. I gave a very rough sketch of what I’m trying to go for, but you can change it however you like



When you deploy a resource onto a public subnet, you still have to select the checkbox that says you want to give it a public IP. By default all resources will be given a private IP that will allow them to communicate internally within AWS. When you click the checkbox to give it a public IP, the server will have both a private IP on the interface and a public ip on that same interface.

And so traffic destined to the public IP on the server map to the private IP on that interface

[design team] Have an ec2 instance with a private ip of 192.168.1.1. Then show that when the user enabled the public-ip checkbox the server also gets a public ip of 1.1.1.1. Then show when users send a request to 1.1.1.1, once the request arrives in aws it then gets mapped to 192.168.1.1 the private IP. I gave a very rough sketch of what I’m trying to go for, but you can change it however you like

Summary

01 Enable resources to connect to internet

02 Internet Gateways attach to a VPC and are region resilient

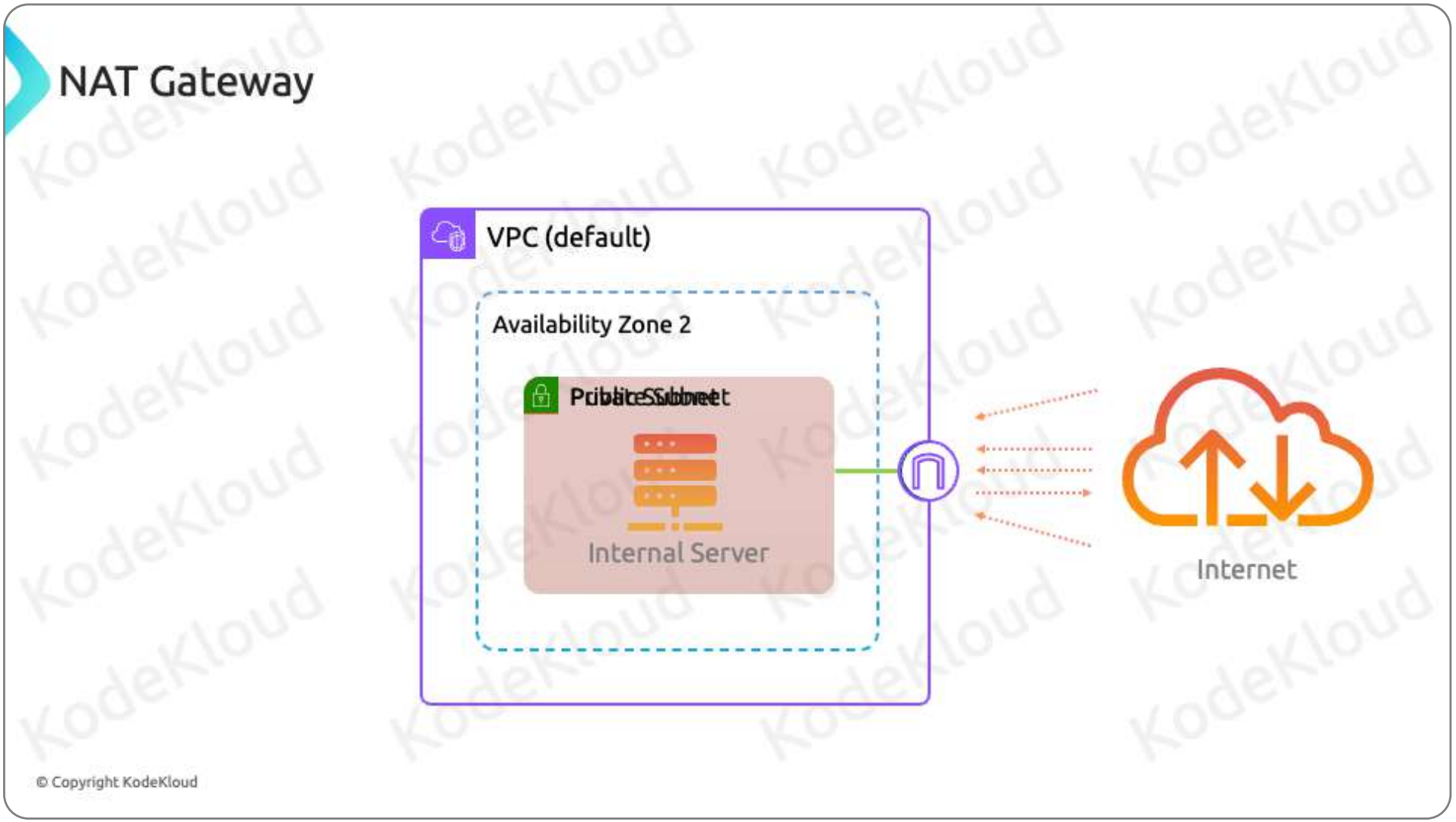
03 VPC can only have max 1 internet gateway, and an internet gateway can only be attached to max 1 VPC

04 A subnet is made public once a default route points to the internet gateway in the VPC

Service Section

The Power of Core Networking

NAT Gateway



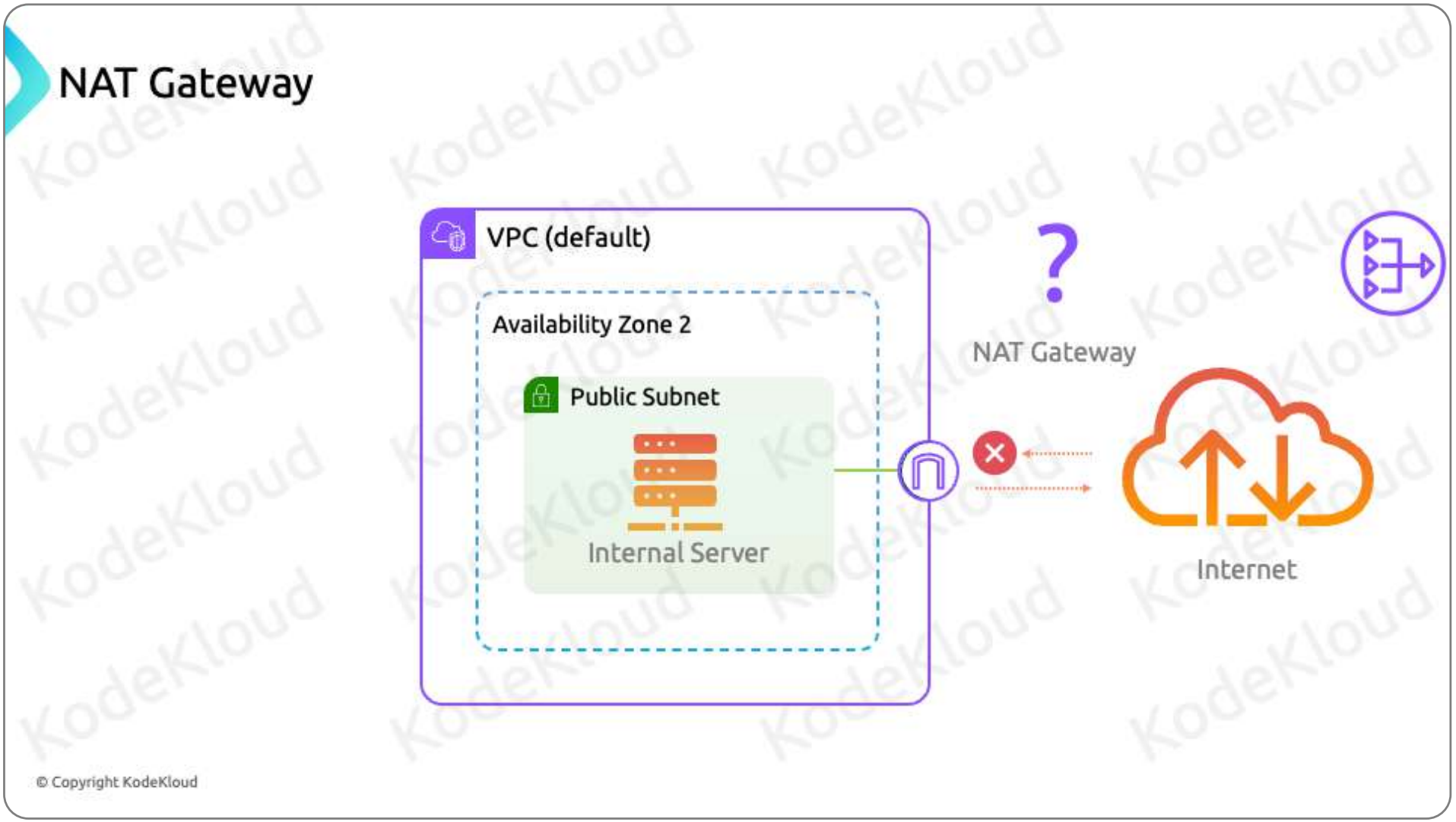
Now let's say that we have a server in a private subnet and we need to give it access to access to the internet so it can download updates and security patches. Now we know that we can turn the subnet into a public subnet by attaching a transit gateway to the VPC and setting up a default route pointing to it.

The issue with this is that now the server is in a public subnet which means its open to the entire internet so any device can talk to it. What do we do if this is server is meant to be an "internal" server that shouldn't be accessible from the internet

but it should still be able to initiate a connection to the internet for updates.

How do make it so that we can give a private resource/server only outgoing access to the internet?

Well AWS has NAT gateways to solve this issue



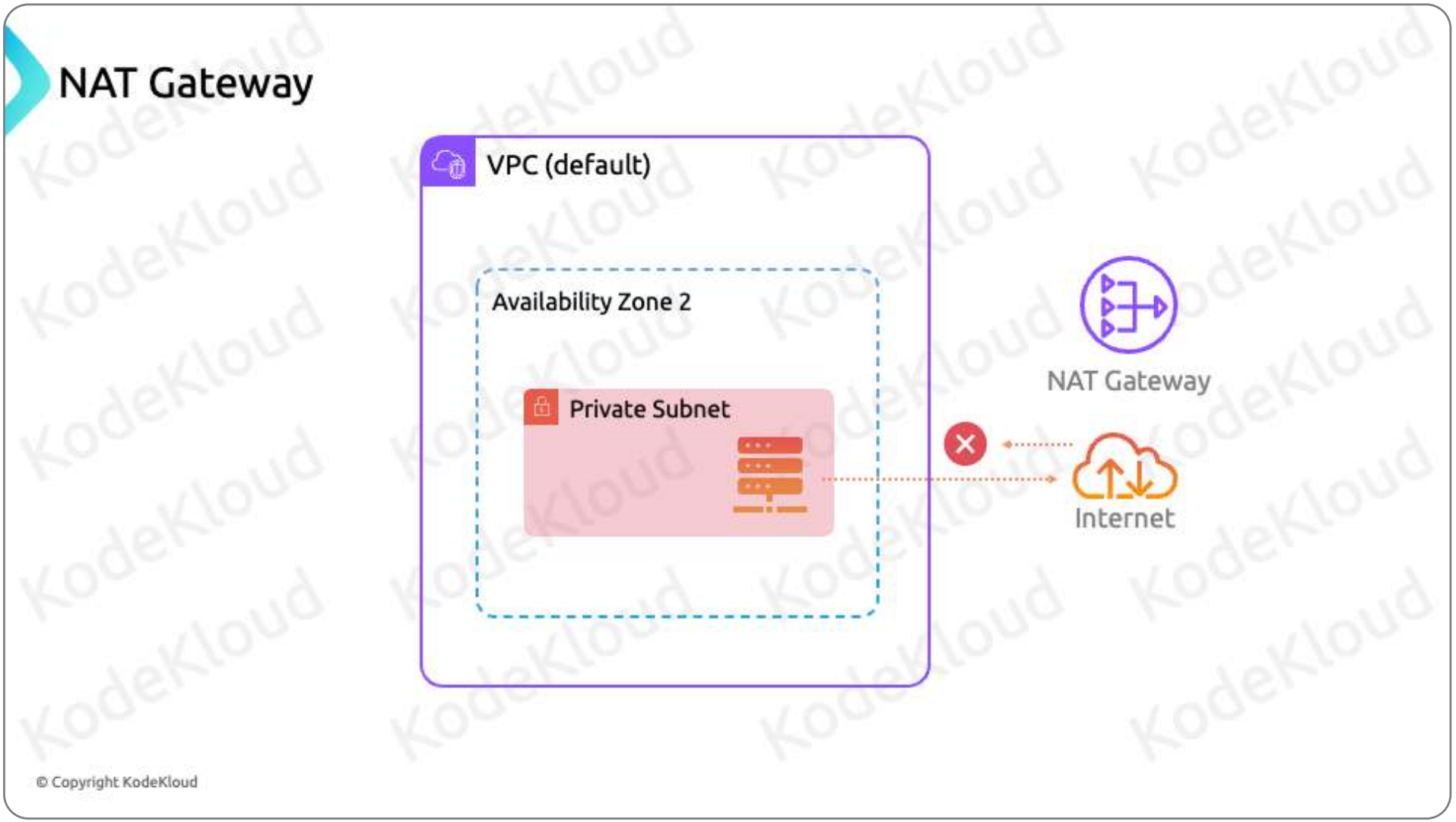
Now let's say that we have a server in a private subnet and we need to give it access to access to the internet so it can download updates and security patches. Now we know that we can turn the subnet into a public subnet by attaching a transit gateway to the VPC and setting up a default route pointing to it.

The issue with this is that now the server is in a public subnet which means its open to the entire internet so any device can talk to it. What do we do if this is server is meant to be an "internal" server that shouldn't be accessible from the internet

but it should still be able to initiate a connection to the internet for updates.

How do make it so that we can give a private resource/server only outgoing access to the internet?

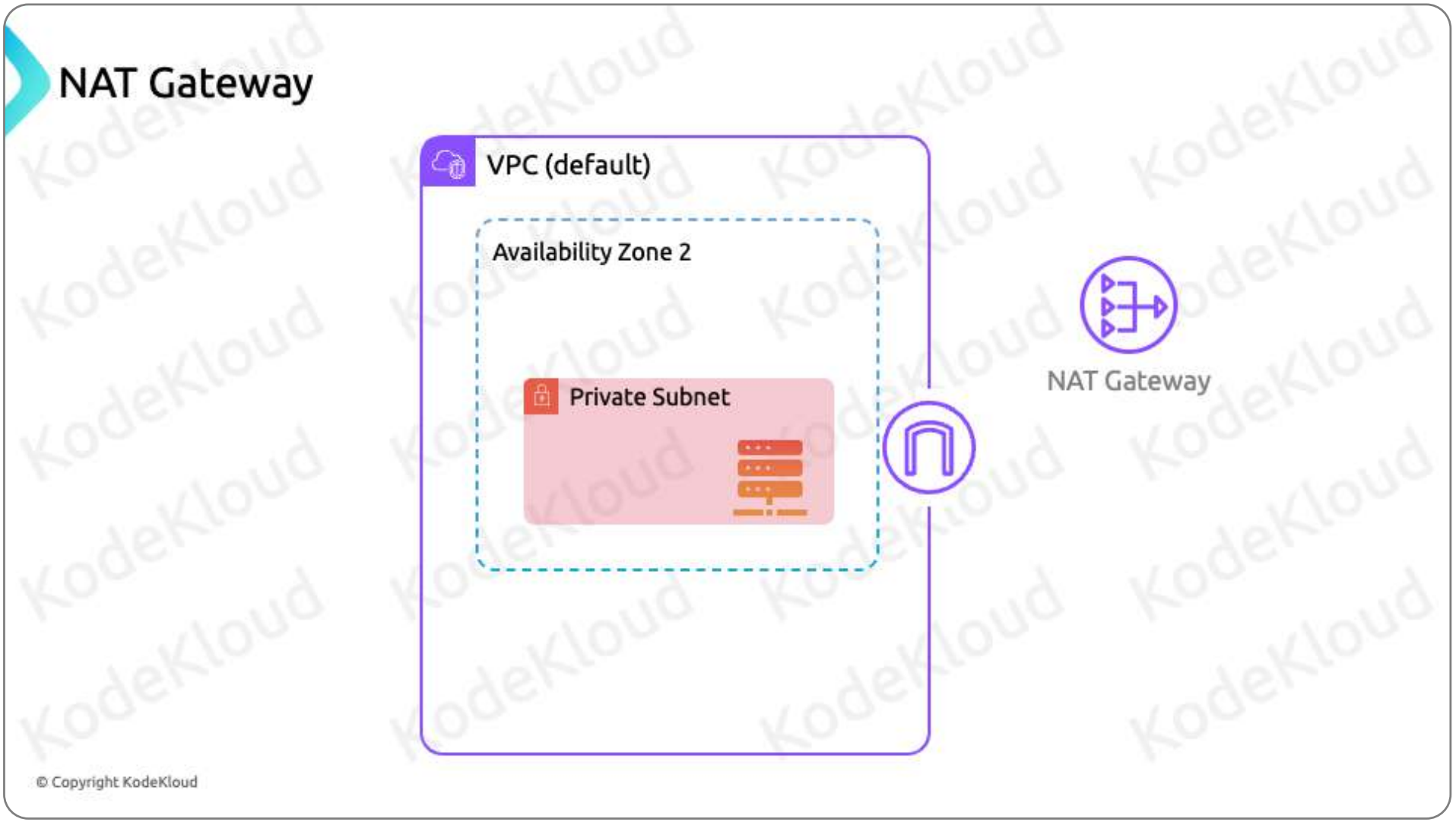
Well AWS has NAT gateways to solve this issue



So let’s say that we have a server running on a private subnet
And let’s say that we need to grant it internet access so it can download updates form the internet
But this server should still be a private server so it should only have outgoing internet access. To accomplish this let’s utilize a NAT gateway.

For NAT Gateways to work we still need a Internet Gateway.

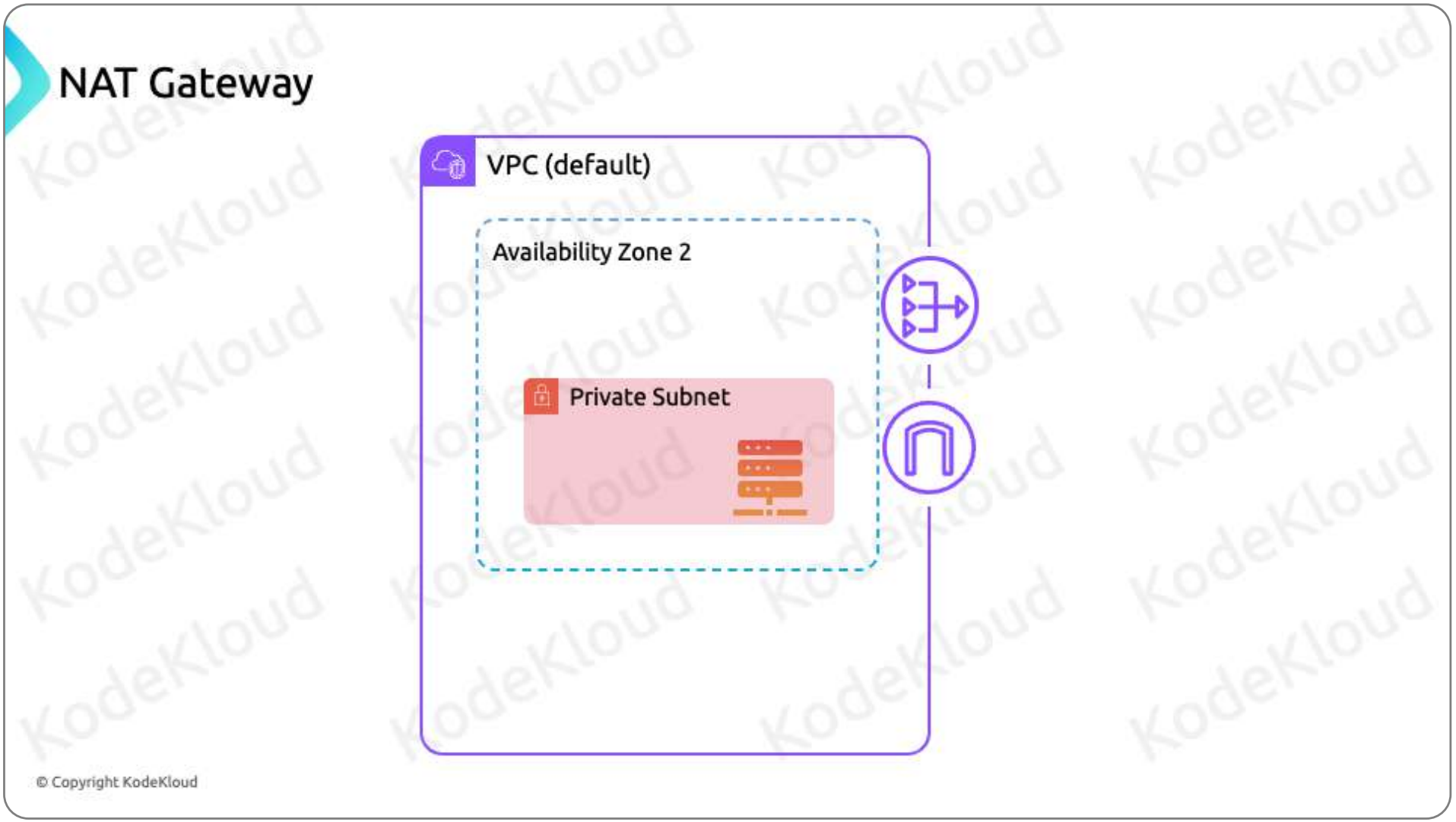
1. We need to create a NAT Gateway and attach it to the VPC
2. We need to create a public subnet which means adding a default route pointing to the internet gateway
3. We then will deploy the NAT gateway onto the public Subnet. The NAT Gateway is on a public subnet so it is a public device that has full access to the internet and the internet can talk to it
4. So now what we need to do is setup routing so that devices in the private subnet will have a default route(0.0.0.0/0) that points to the NAT Gateway
5. So now the packet flow goes:
   Private subnet server to nat gateway
   Nat gateway to the internet Gateway
   Internet Gateway to the internet
   Return direction:
   Internet to internet gateway
   Internet gateway to nat gateway
   Nat gateway to private subnet



So let's say that we have a server running on a private subnet
And let's say that we need to grant it internet access so it can download updates form the internet
But this server should still be a private server so it should only have outgoing internet access. To accomplish this let's utilize a NAT gateway.

For NAT Gateways to work we still need a Internet Gateway.

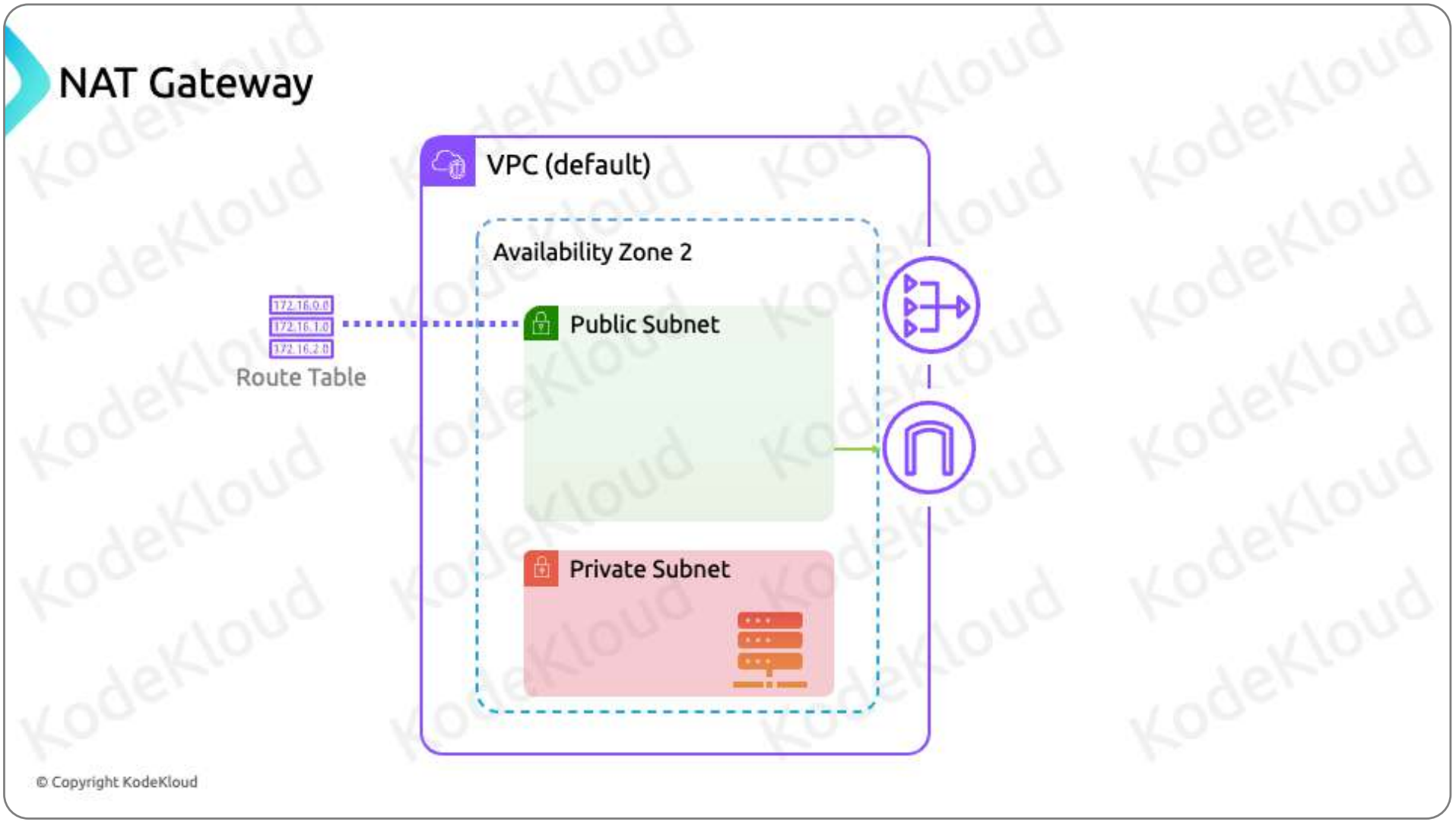
1. We need to create a NAT Gateway and attach it to the VPC
2. We need to create a public subnet which means adding a default route pointing to the internet gateway
3. We then will deploy the NAT gateway onto the public Subnet. The NAT Gateway is on a public subnet so it is a public device that has full access to the internet and the internet can talk to it
4. So now what we need to do is setup routing so that devices in the private subnet will have a default route(0.0.0.0/0) that points to the NAT Gateway
5. So now the packet flow goes:
   Private subnet server to nat gateway
   Nat gateway to the internet Gateway
   Internet Gateway to the internet
   Return direction:
   Internet to internet gateway
   Internet gateway to nat gateway
   Nat gateway to private subnet



So let’s say that we have a server running on a private subnet
And let’s say that we need to grant it internet access so it can download updates form the internet
But this server should still be a private server so it should only have outgoing internet access. To accomplish this let’s utilize a NAT gateway.

For NAT Gateways to work we still need a Internet Gateway.

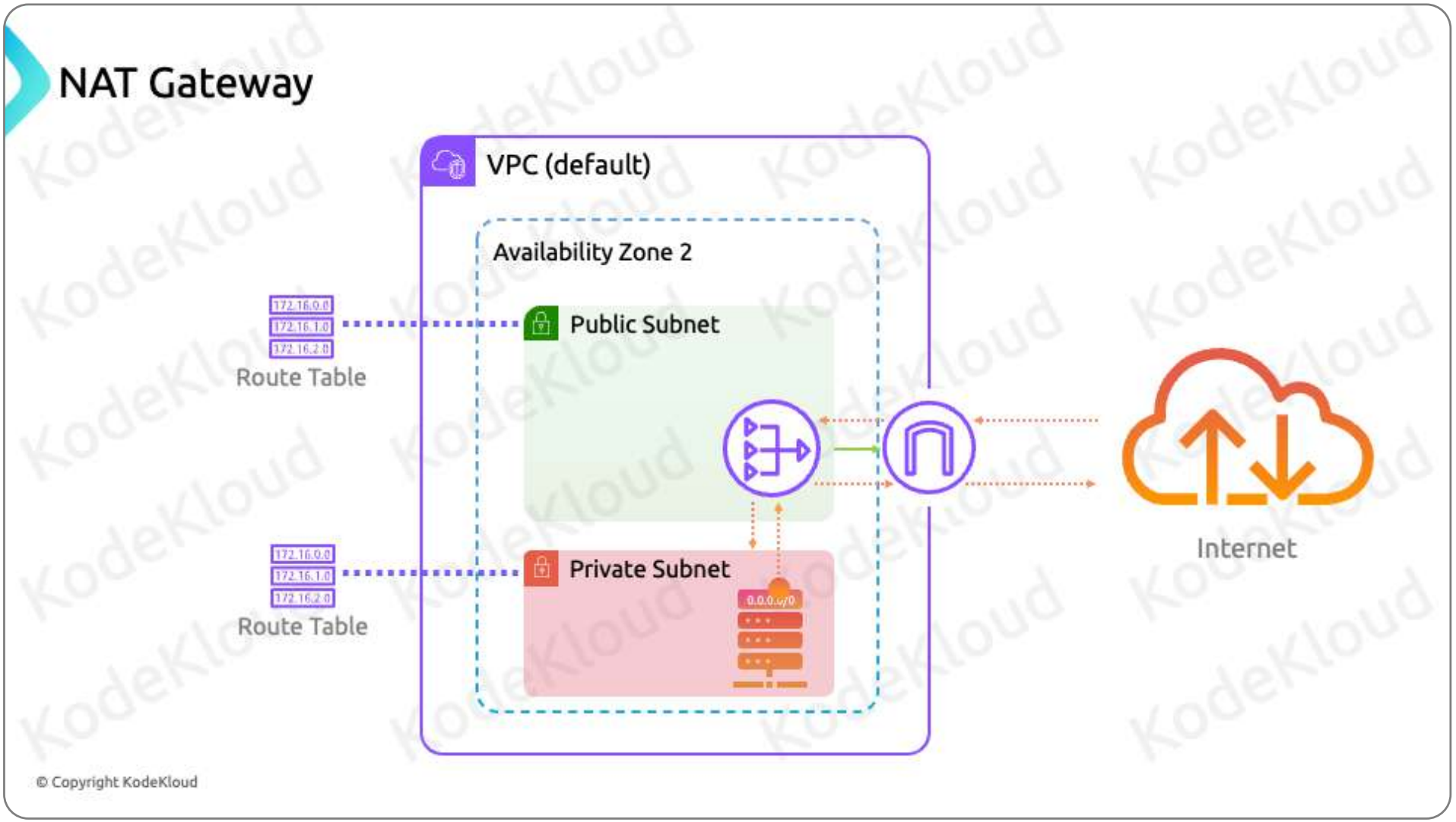
1. We need to create a NAT Gateway and attach it to the VPC
2. We need to create a public subnet which means adding a default route pointing to the internet gateway
3. We then will deploy the NAT gateway onto the public Subnet. The NAT Gateway is on a public subnet so it is a public device that has full access to the internet and the internet can talk to it
4. So now what we need to do is setup routing so that devices in the private subnet will have a default route(0.0.0.0/0) that points to the NAT Gateway
5. So now the packet flow goes:
   Private subnet server to nat gateway
   Nat gateway to the internet Gateway
   Internet Gateway to the internet
   Return direction:
   Internet to internet gateway
   Internet gateway to nat gateway
   Nat gateway to private subnet



So let’s say that we have a server running on a private subnet
And let’s say that we need to grant it internet access so it can download updates form the internet
But this server should still be a private server so it should only have outgoing internet access. To accomplish this let’s utilize a NAT gateway.

For NAT Gateways to work we still need a Internet Gateway.

1. We need to create a NAT Gateway and attach it to the VPC
2. We need to create a public subnet which means adding a default route pointing to the internet gateway
3. We then will deploy the NAT gateway onto the public Subnet. The NAT Gateway is on a public subnet so it is a public device that has full access to the internet and the internet can talk to it
4. So now what we need to do is setup routing so that devices in the private subnet will have a default route(0.0.0.0/0) that points to the NAT Gateway
5. So now the packet flow goes:
   Private subnet server to nat gateway
   Nat gateway to the internet Gateway
   Internet Gateway to the internet
   Return direction:
   Internet to internet gateway
   Internet gateway to nat gateway
   Nat gateway to private subnet

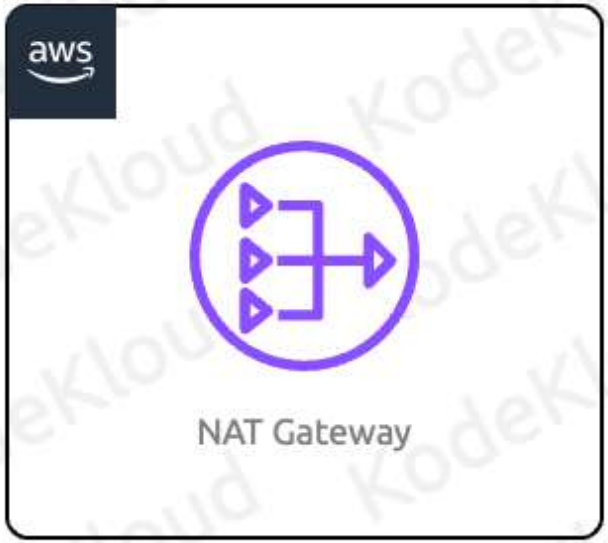


So let’s say that we have a server running on a private subnet
And let’s say that we need to grant it internet access so it can download updates form the internet
But this server should still be a private server so it should only have outgoing internet access. To accomplish this let’s utilize a NAT gateway.

For NAT Gateways to work we still need a Internet Gateway.

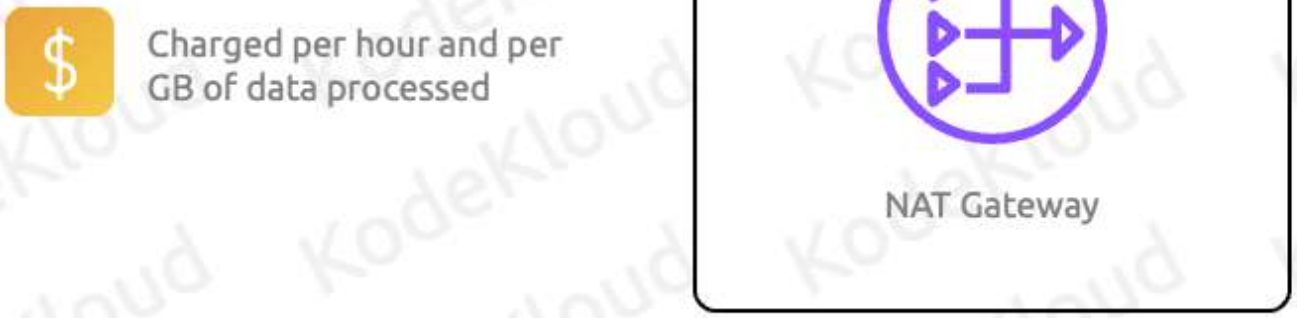
1. We need to create a NAT Gateway and attach it to the VPC
2. We need to create a public subnet which means adding a default route pointing to the internet gateway
3. We then will deploy the NAT gateway onto the public Subnet. The NAT Gateway is on a public subnet so it is a public device that has full access to the internet and the internet can talk to it
4. So now what we need to do is setup routing so that devices in the private subnet will have a default route(0.0.0.0/0) that points to the NAT Gateway
5. So now the packet flow goes:
   Private subnet server to nat gateway
   Nat gateway to the internet Gateway
   Internet Gateway to the internet
   Return direction:
   Internet to internet gateway
   Internet gateway to nat gateway
   Nat gateway to private subnet

NAT Gateway

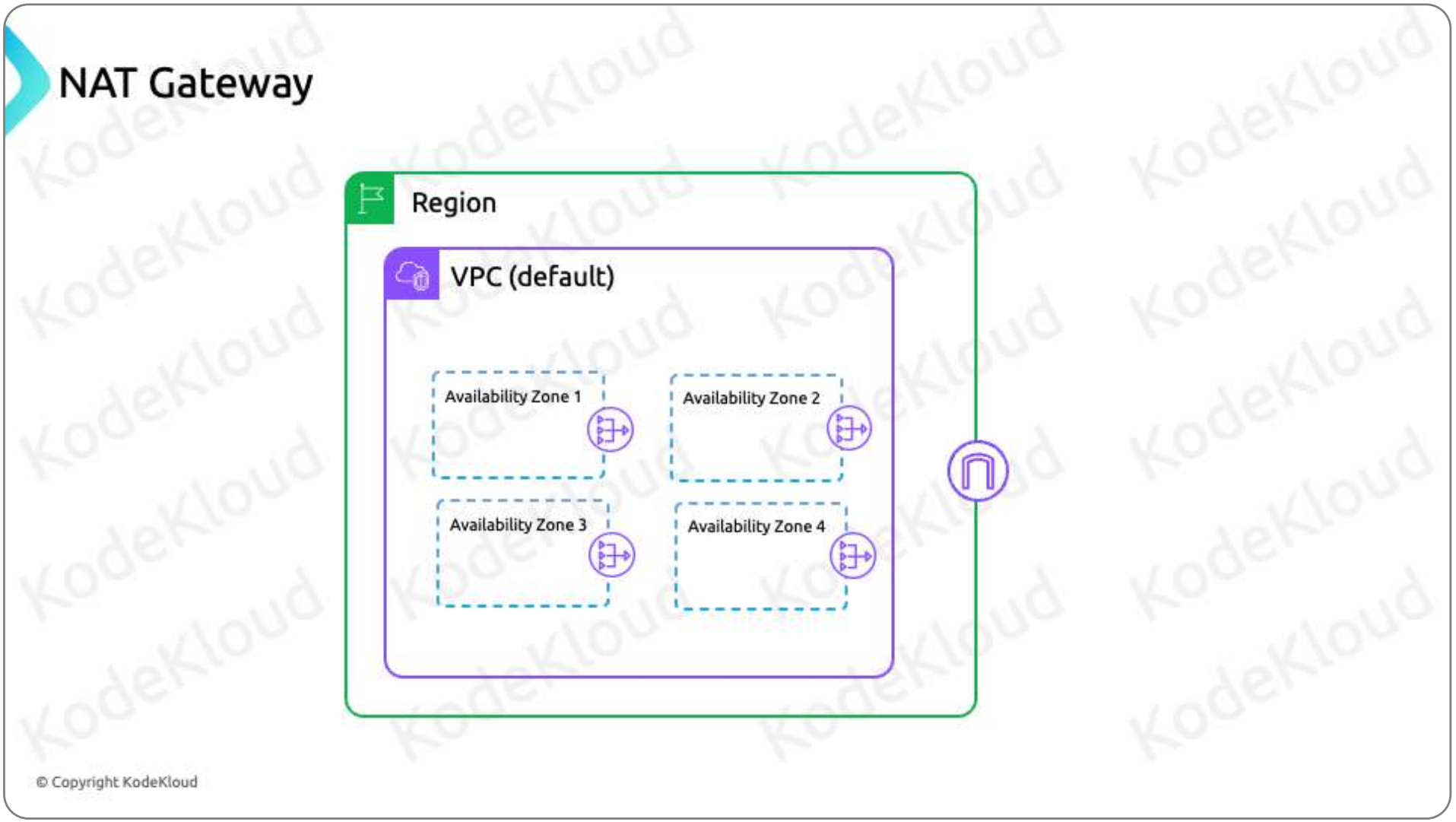


Just like Internet Gateways, NAT gateways are a managed service so you deploy them and AWS handles the rest
You are charged for every hour the NAT Gateways is runnign as well as each Gigabyte of data that is processed
Nat Gateways are not region resilient like Internet Gateways.
They are only AZ resilient. That means you will need a NAT Gateway in each AZ of a region
Private subnets in each availability zone will need to have a default route 0.0.0.0/0 pointing to the NAT Gateway in that specific AZ

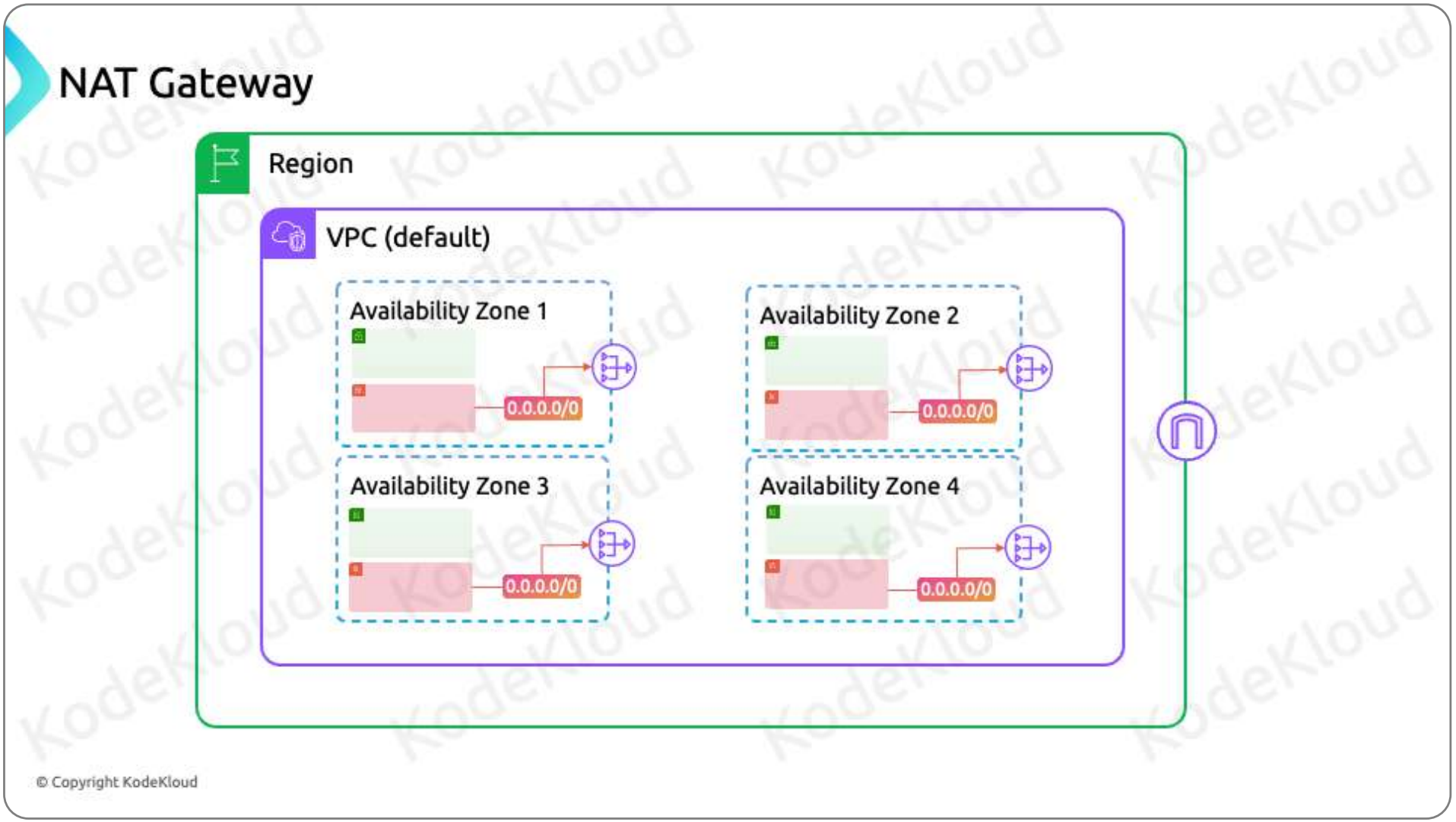
NAT Gateway



Just like Internet Gateways, NAT gateways are a managed service so you deploy them and AWS handles the rest
You are charged for every hour the NAT Gateways is runnign as well as each Gigabyte of data that is processed
Nat Gateways are not region resilient like Internet Gateways.
They are only AZ resilient. That means you will need a NAT Gateway in each AZ of a region Private subnets in each availability zone will need to have a default route 0.0.0.0/0 pointing to the NAT Gateway in that specific AZ



Nat Gateways are not region resilient like Internet Gateways.
They are only AZ resilient. That means you will need a NAT Gateway in each AZ of a region Private subnets in each availability zone will need to have a default route 0.0.0.0/0 pointing to the NAT Gateway in that specific AZ



Nat Gateways are not region resilient like Internet Gateways.
They are only AZ resilient. That means you will need a NAT Gateway in each AZ of a Private subnets in each availability zone will need to have a default route 0.0.0.0/0 pointing to the NAT Gateway in that specific AZ

Summary

01 NAT Gateways allow subnets to talk to the internet, but connections must be initiated from within the VPC

02 NAT Gateways are deployed onto public Subnets so that they have a public IP and internet access

03 Uses Elastic IPs

04 AZ-reliant service; need 1 NAT Gateway in each AZ

Summary

05 Route table for private subnets should point to NAT Gateway

06 Managed by AWS

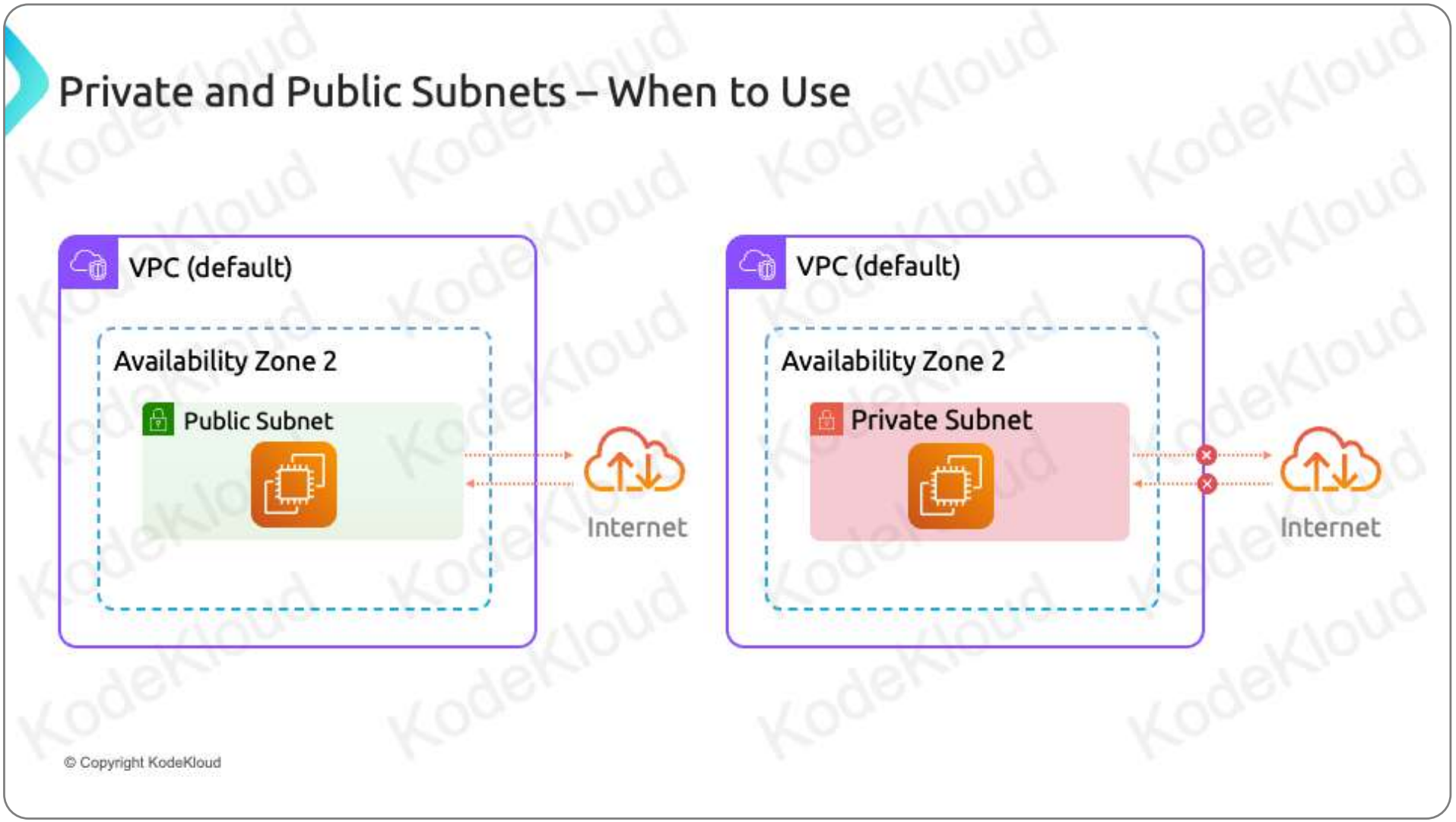
07 Charged for each hour that NAT gateway is available and for each Gigabyte of data that it processes

08 A NAT gateway supports 5 Gbps of bandwidth and automatically scales up to 100 Gbps

Service Section

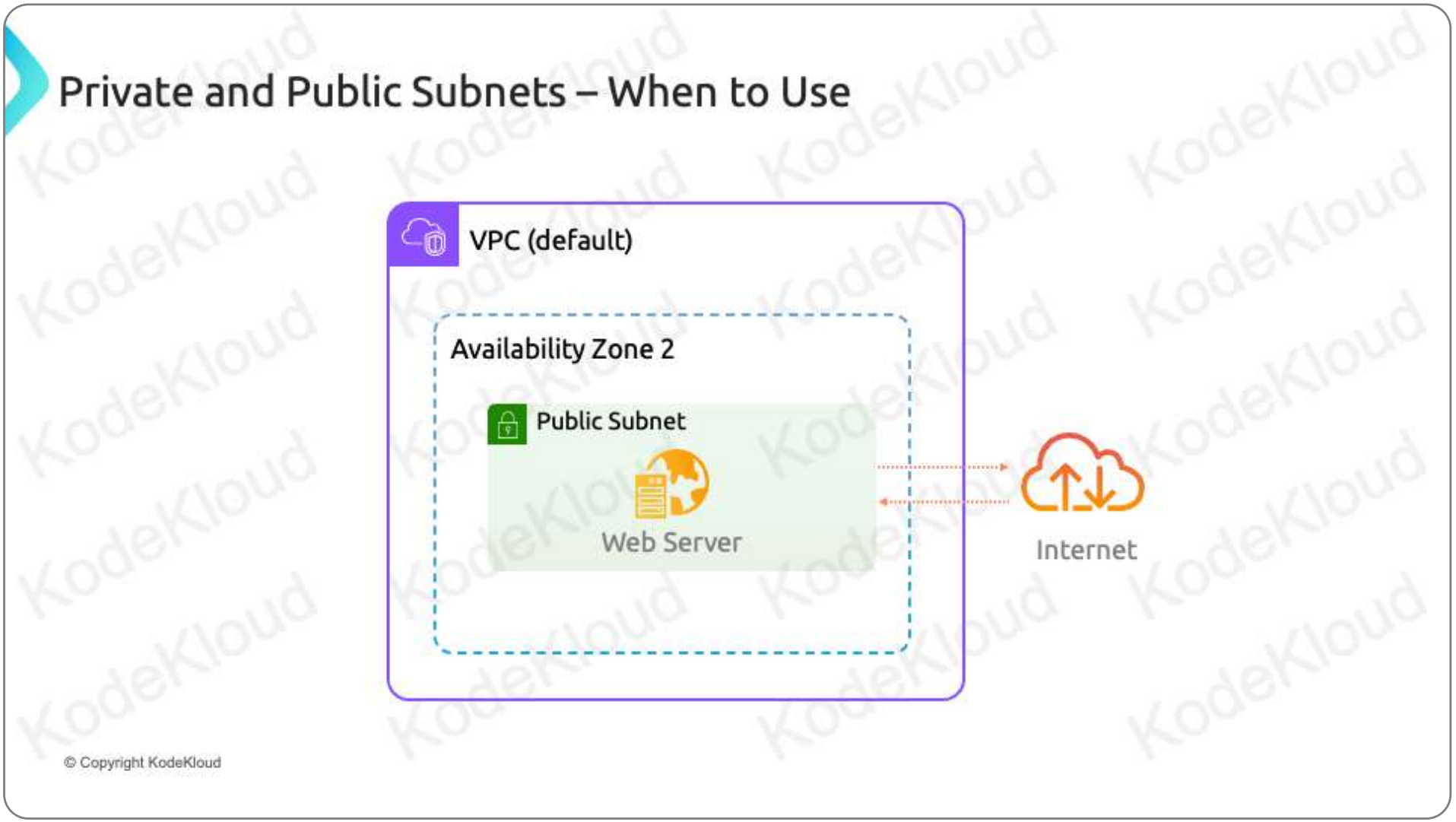
The Power of Core Networking

Private and Public Subnets



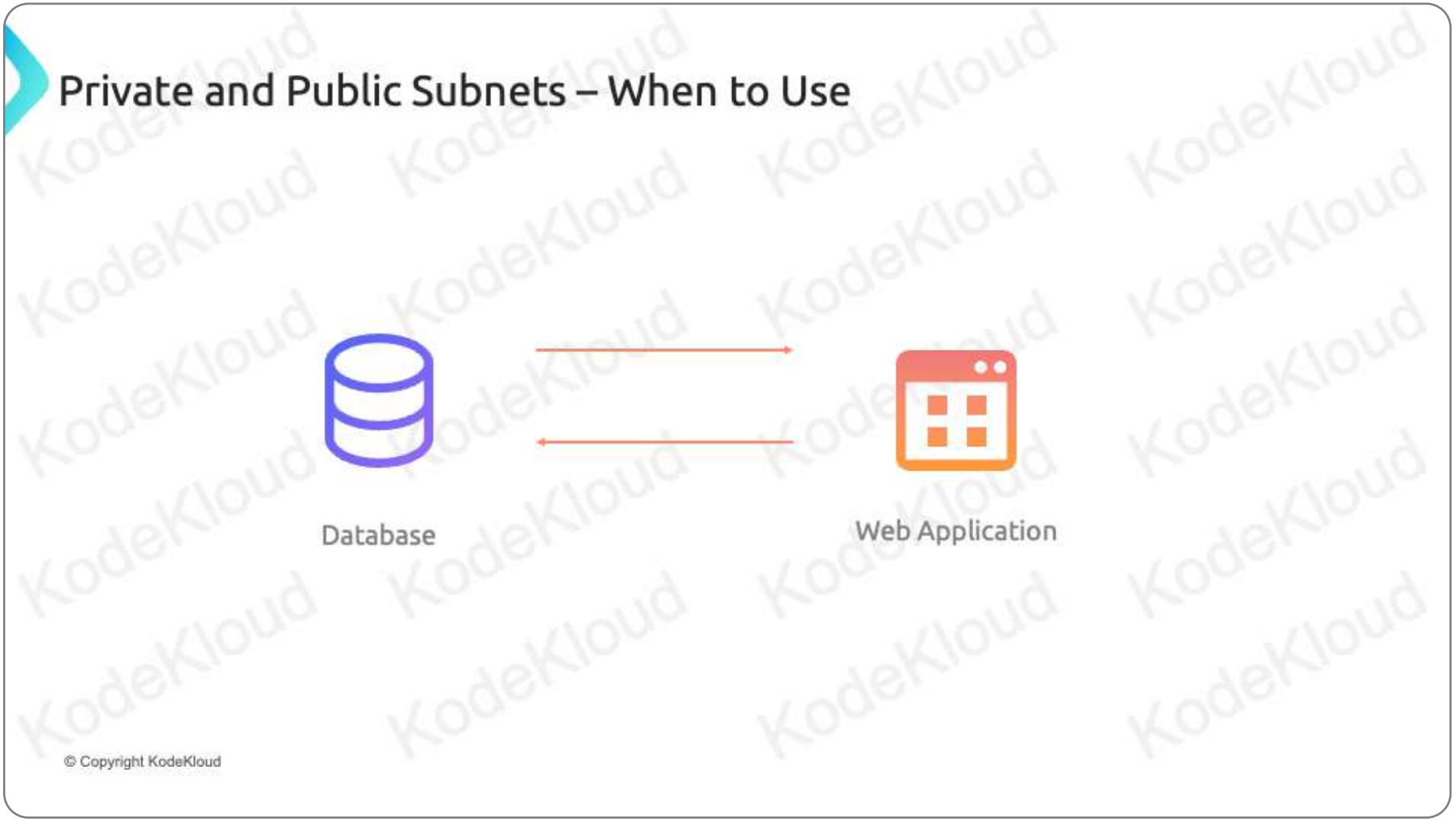
To determine whether a subnet should be public or private you need to ask yourself should devices on the internet be able to interact with resources deployed on the subnet. If yes then it should be public, if not make it private.

An example of an applicated that should be on a public subnet would be a web server. Since users on the internet need to interact with the website, the webserver needs to be on a public subnet.

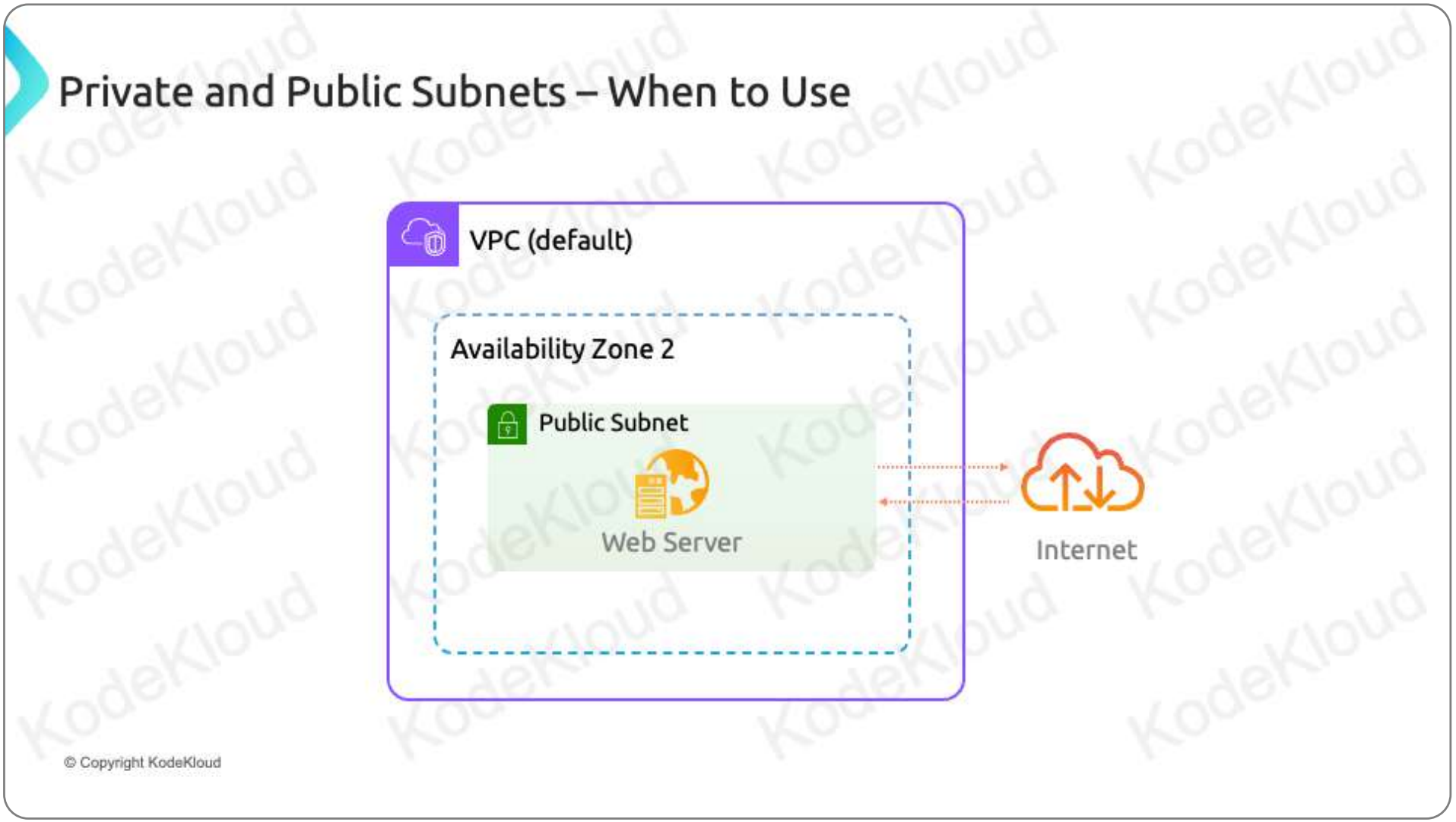


To determine whether a subnet should be public or private you need to ask yourself should devices on the internet be able to interact with resources deployed on the subnet. If yes then it should be public, if not make it private.

An example of an applicated that should be on a public subnet would be a web server. Since users on the internet need to interact with the website, the webserver needs to be on a public subnet.

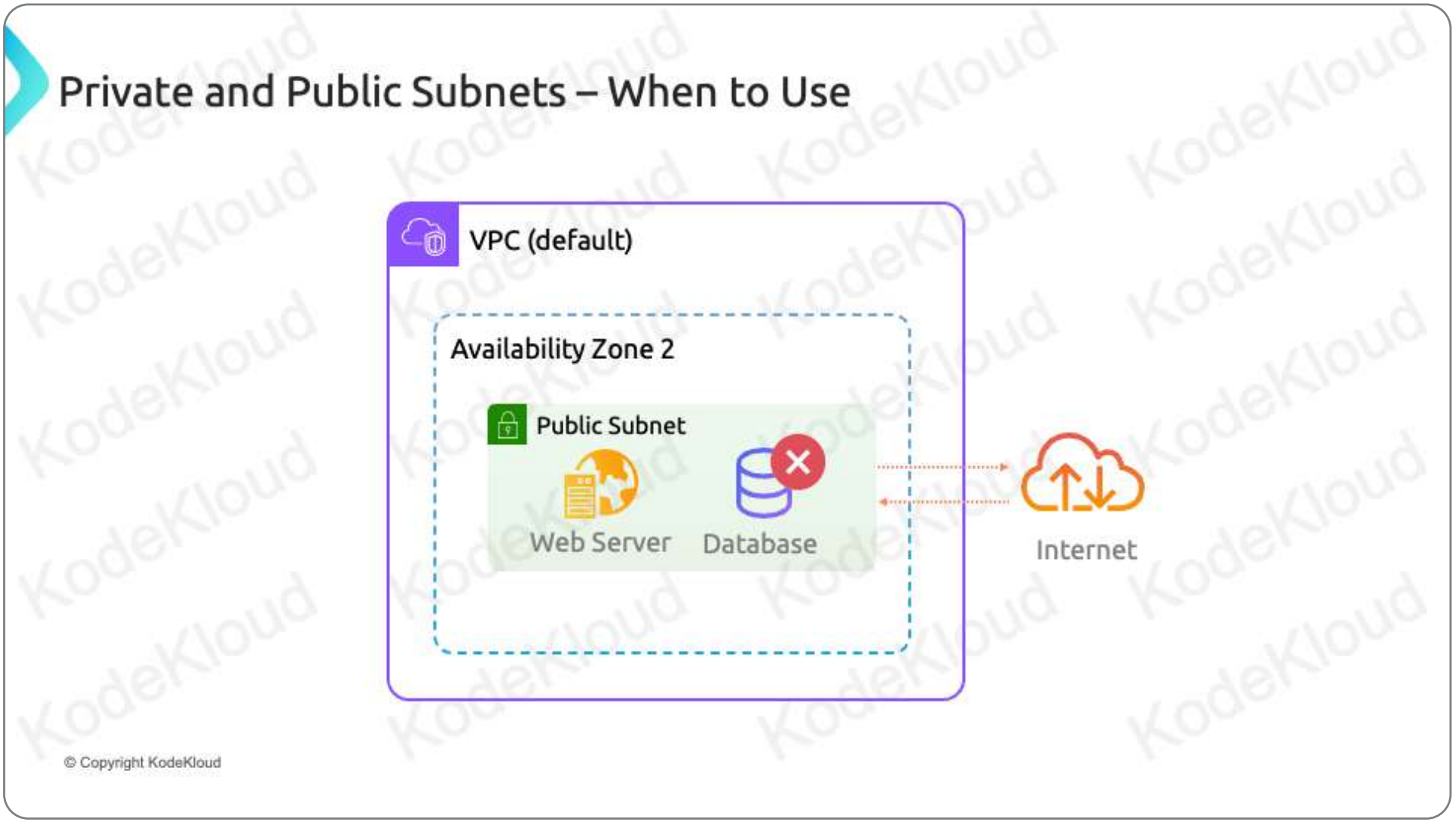


Now let's say we have a web application that has to talk to a database. In this case the web server once again needs to be on a public subnet so that users on the internet can access the website.



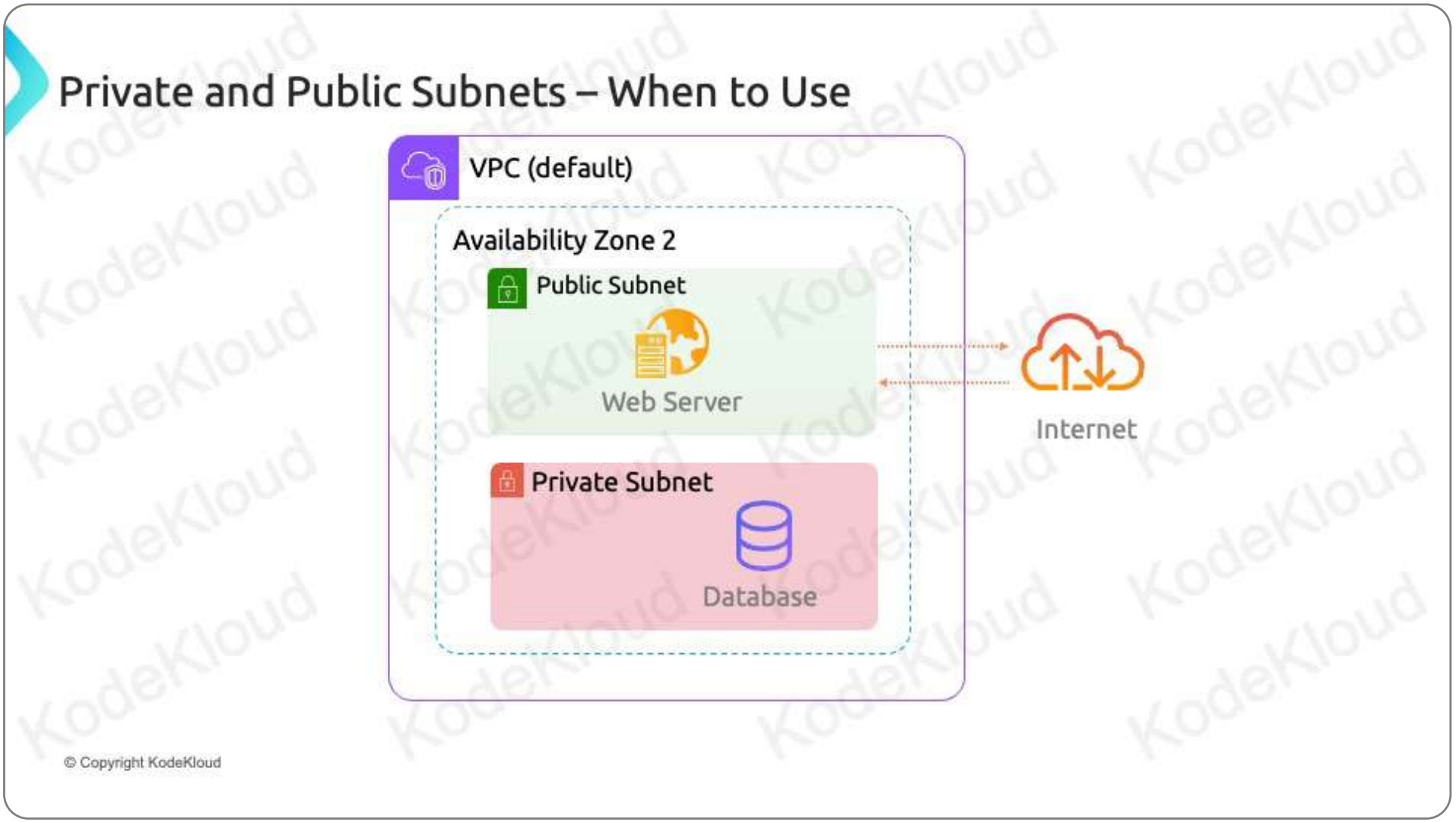
Now let's say we have a web application that has to talk to a database. In this case the web server once again needs to be on a public subnet so that users on the internet can access the website.

But what about the Database? Normally you don't want to expose the database to your users or the internet as it has sensitive information. From the database perspective, only the web server should be able to talk to it. So the database would be a good example of a resource that should only be deployed on a private subnet.



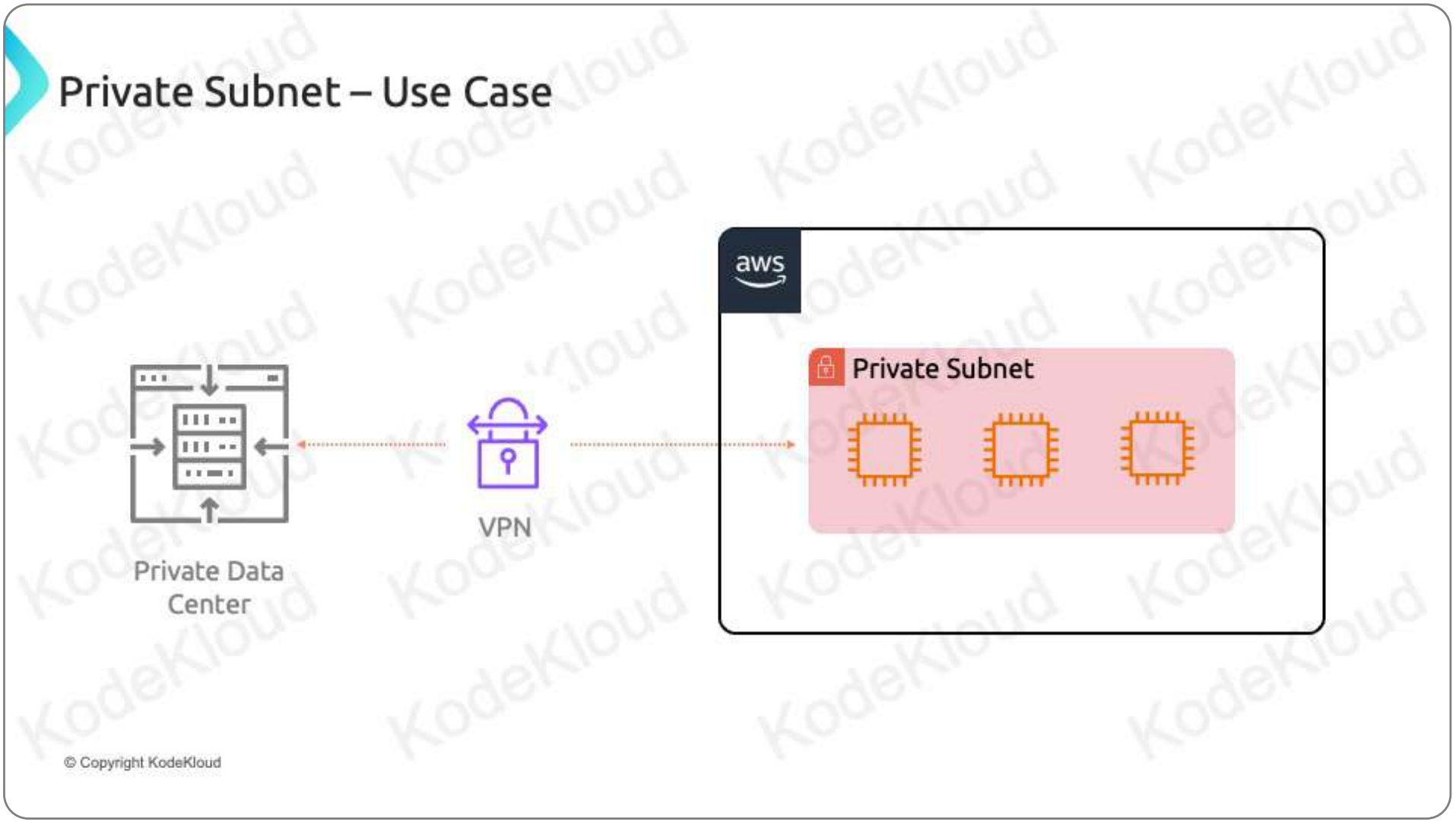
Now let's say we have a web application that has to talk to a database. In this case the web server once again needs to be on a public subnet so that users on the internet can access the website.

But what about the Database? Normally you don't want to expose the database to your users or the internet as it has sensitive information. From the database perspective, only the web server should be able to talk to it. So the database would be a good example of a resource that should only be deployed on a private subnet.



Now let's say we have a web application that has to talk to a database. In this case the web server once again needs to be on a public subnet so that users on the internet can access the website.

But what about the Database? Normally you don't want to expose the database to your users or the internet as it has sensitive information. From the database perspective, only the web server should be able to talk to it. So the database would be a good example of a resource that should only be deployed on a private subnet.



A common use case for the cloud is to have AWS to act as an extension to your private datacenter, then you can deploy all your resources in a private subnet and use a VPN to connect it to your datacenter as all of these resources don't need to be private.

Resources in public subnets are accessible to and from the internet

Resources in private subnets are not accessible from the internet

Gateways determine whether a subnet is public or private

Service Section

The Power of Edge Networking

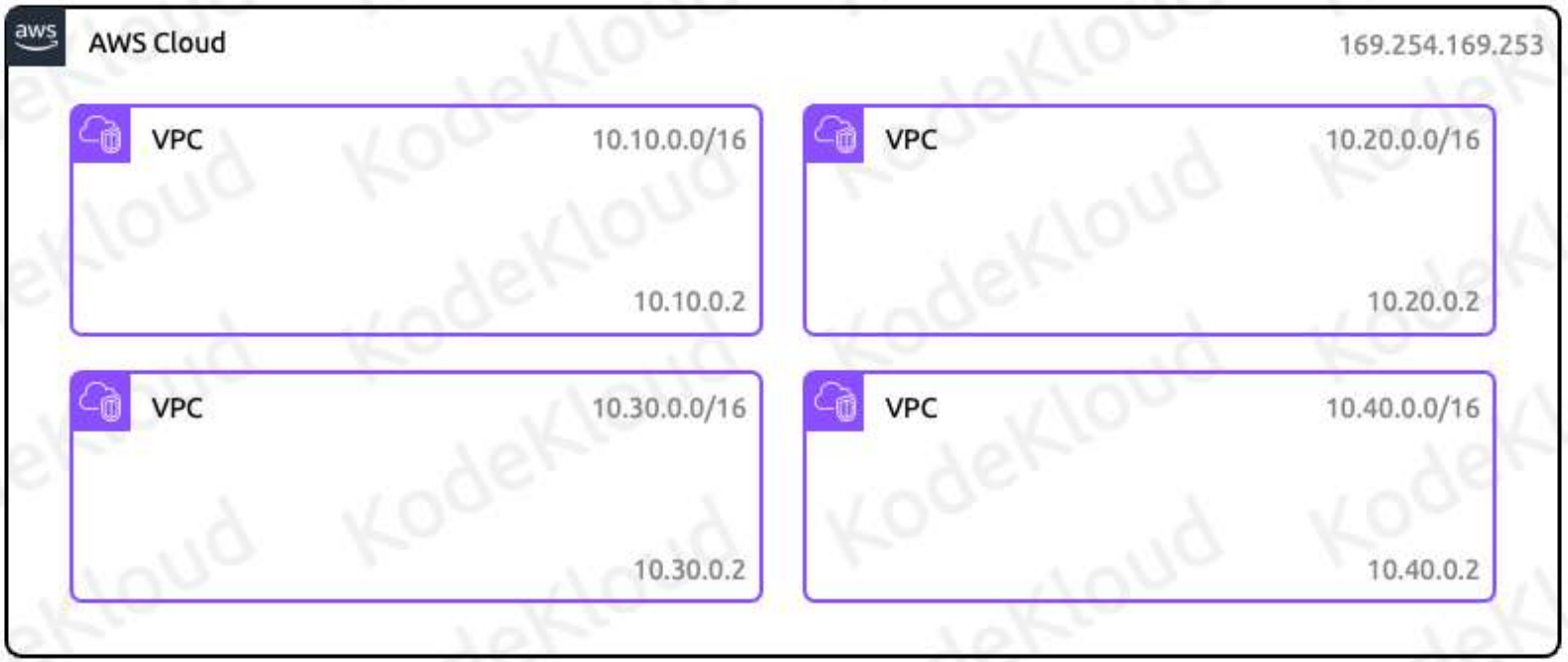
DNS in VPC

When resource like an ec2 instance gets deployed to a subnet it will be given a private IP and that is both for private and public subnets. By default all private addresses assigned to an EC2 instance will get a domain name.

So as an example let's say we have a subnet with a address block of 10.0.100.0/24. The ec2 instance will get an IP from the block, lets say it gets 10.0.100.10. AWS will also give you a dns entry that will make to the 10.0.100.10 address that will look something like ip-10-0-100-10.ec2.internal. The ip address is usually embedded in the dns name to make it easier to identify.

So for other services in aws they could communicate with the ec2 instance using the ip address or the dns name.

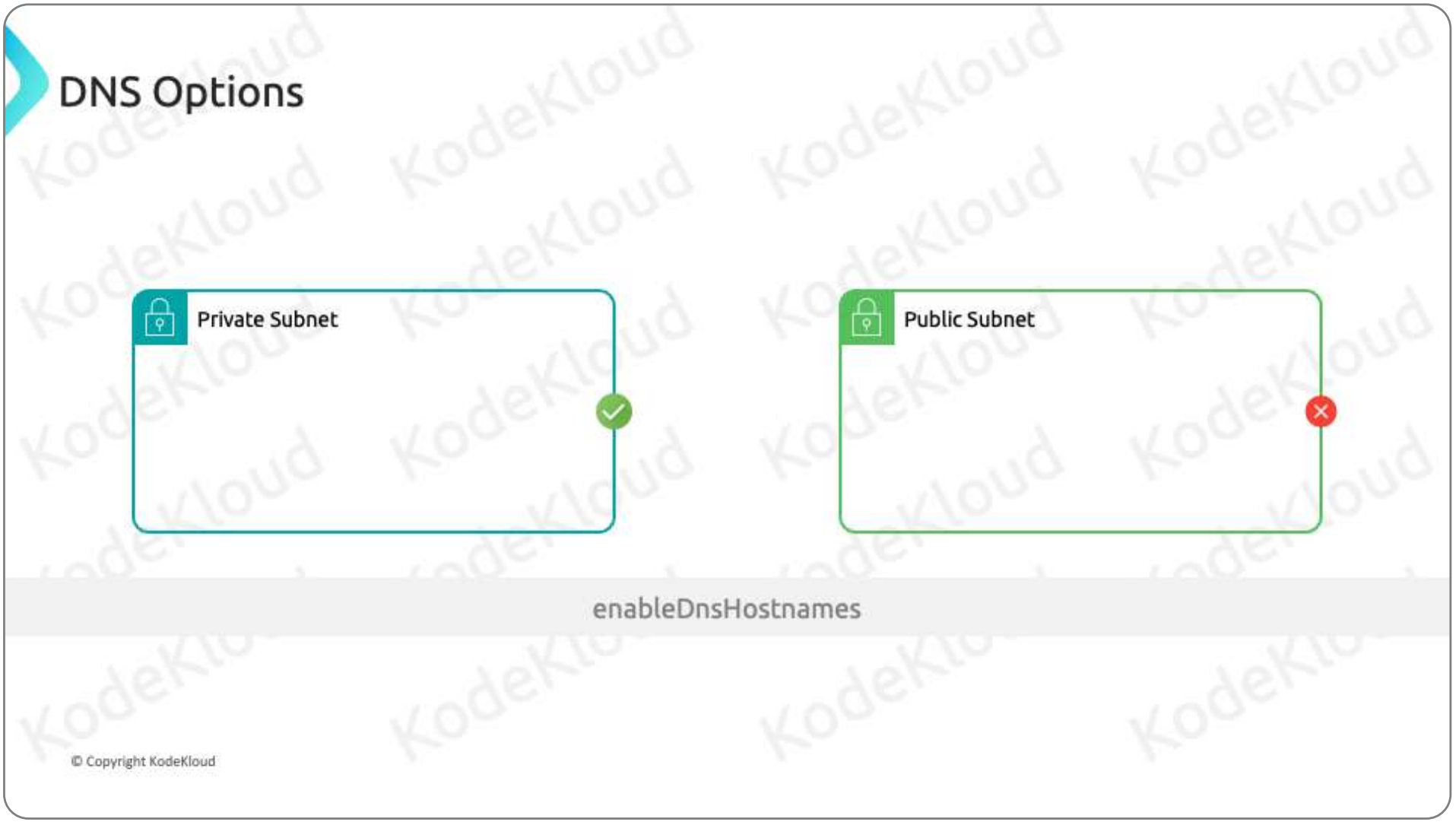
DNS in VPCs



To communicate using the DNS entries for a server we must query the DNS servers. AWS gives access to their DNS server which runs on 169.254.169.253.

The DNS server can also be reached at the second IP address of the VPC.

So if the VPC has a CIDR block of 10.0.0.0/16 then the second ip 10.0.0.2 will act as the DNS server.



When you create a custom VPC there are two knobs the affect.

By default only private IPs get a DNS entry and public ones do not. If you want public IP addresses to also get a domain name you have to make sure that the enableDnsHostnames knob is enabled.

enableDNsHostnames - Determines whether the VPC supports assigning public DNS hostnames to instances with public IP addresses.
The default for this attribute is false unless the VPC is a default VPC.

Then to actually enable DNS resolution in the VPC so we can utilize the AWS DNS servers we have to make sure that we enable the enableDnsSupport knob

enableDnsSupport - Determines whether the VPC supports DNS resolution through the Amazon provided DNS server (169.254.169.253).
If this attribute is true, queries to the Amazon provided DNS server succeed. For more information, see Amazon DNS server.
The default for this attribute is true.

The AWS DNS server also resolves pubic DNS entries as well not just aws internal IPs

Summary

01 Device private IPs will automatically be assigned a DNS entry

02 AWS DNS server can be accessed on the second IP of the VPC CIDR block as well – 169.254.169.253

03 **enableDnsHostnames** – Determines whether the VPC supports assigning public DNS hostnames to instances with public IP addresses

04 **enableDnsSupport** – Determines whether the VPC supports DNS resolution through the Amazon provided DNS server

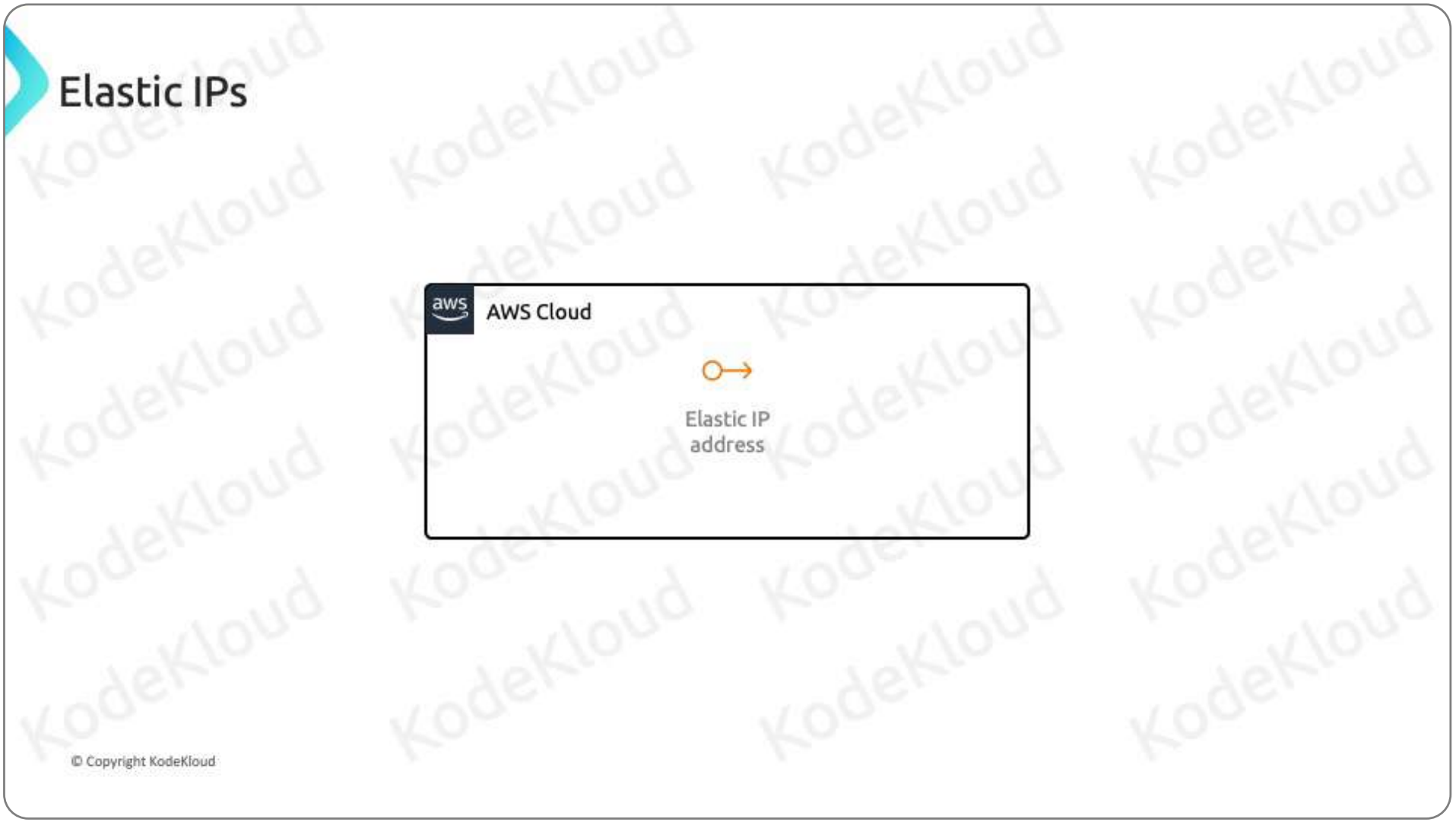
Service Section

The Power of Core Networking

Elastic IP

When a server (EC2) instance gets deployed to a public subnet, it is given a public IP. The thing about this Public IP is that it is not reserved for our instance. In fact if we shut down and reboot our server, you'll see that the public IP will change and our old address might be used by another customer.

Now, this can obviously be a problem if our users are going to a specific IP and then our server reboots. Then all of a sudden, it will be at a new IP and they won't know about it.



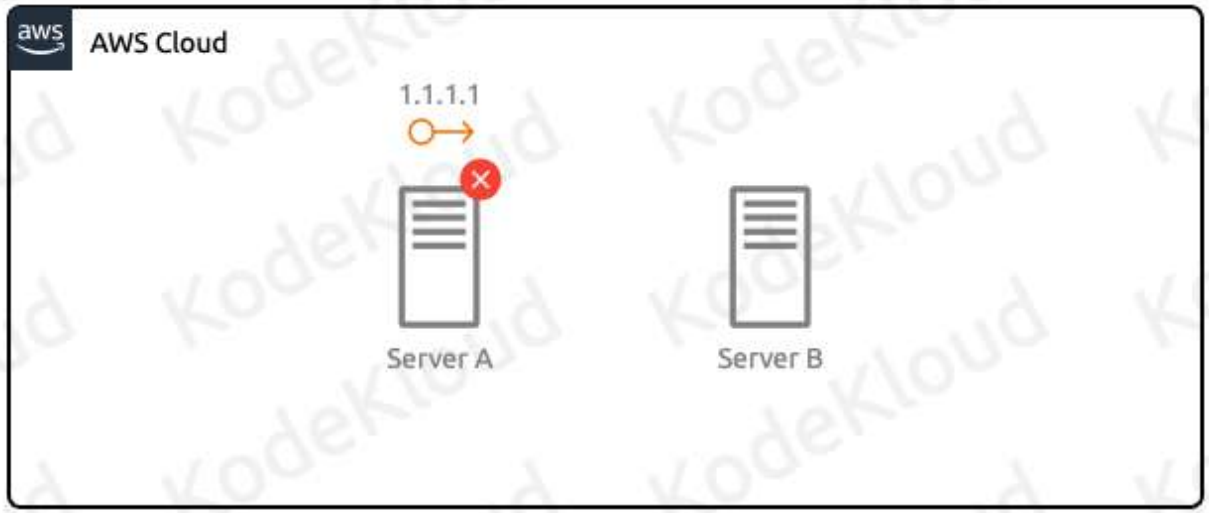
That is why AWS has what's called Elastic IP addresses.

Elastic IP addresses is a static IPv4 address. When you allocate an Elastic IP address to your account, it is yours until you release it.

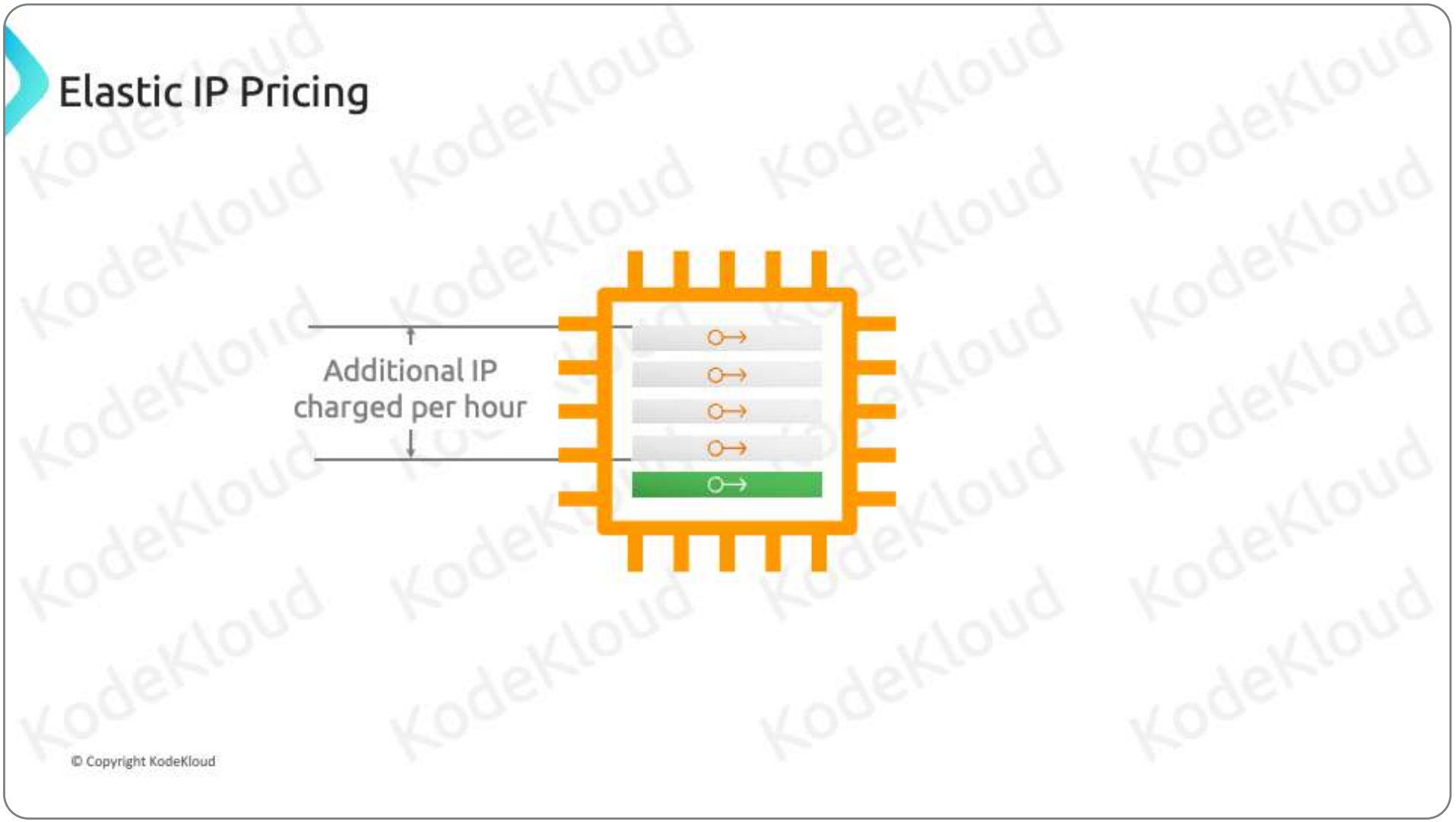
We can then associate the elastic IP with an EC2 instance so that even if it reboots, it will always have the static (elastic IP)

because that is reserved for our account.

Elastic IPs



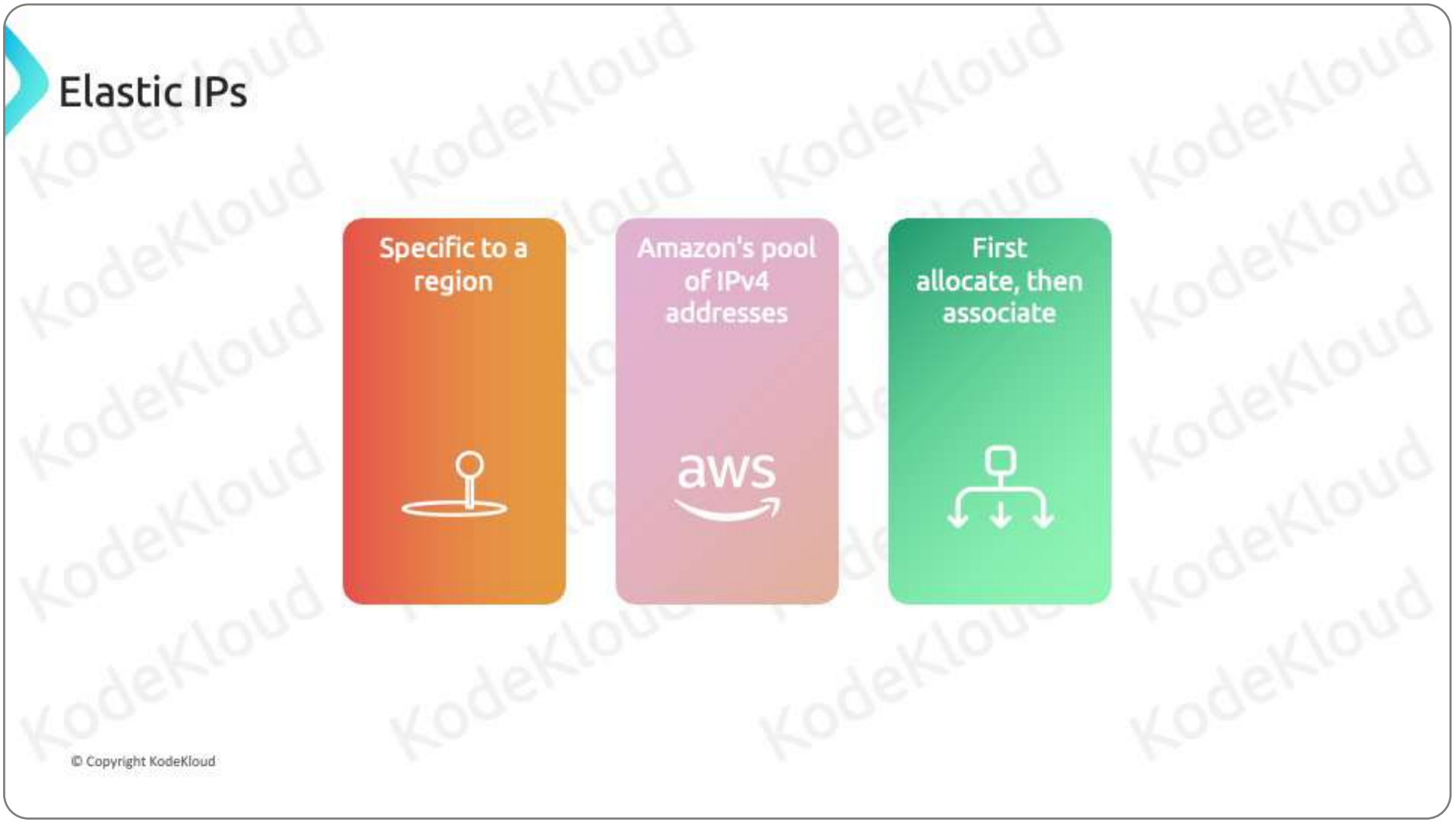
The Elastic IP can be moved from one server to another. So if the elastic IP is associated with server A and server A has to go down for maintenance, we can associate the Elastic IP with server B so that our users are not impacted during the downtime.



You can have one Elastic IP address associated with a running instance at no charge.

If you associate additional elastic Ips with that instance you will be charged for each additional IP per hour.

Elastic IPs that are not associated with an instance incur a small hourly charge



- Elastic Ips are specific to a region, cannot be moved to a different region
- An Elastic IP address comes from Amazon's pool of IPv4 addresses, or from a custom IPv4 address pool that you have brought to your AWS account
- To use an Elastic IP address, you first allocate one to your account, and then associate it with your instance or a network interface.

Summary

01 Public IPs are not static and, if an EC2 instance goes down, then it will get a new public IP

02 Elastic IPs are static IPv4 addresses that do not change

03 To use an Elastic IP address, you first allocate one to your account, and then associate it with your instance or a network interface

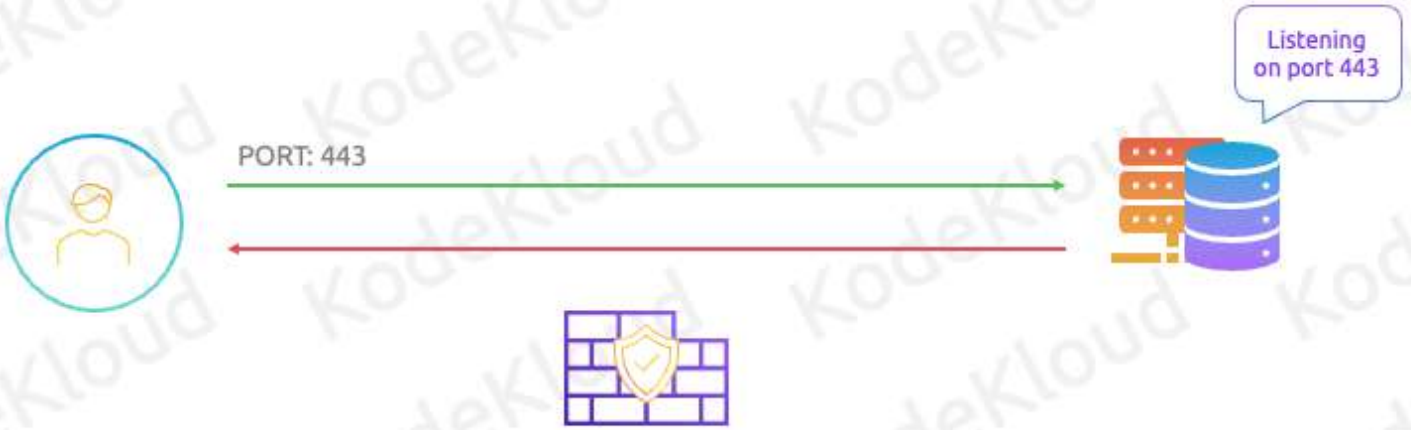
04 Elastic IPs are region specific and cannot be moved to a different region

Service Section

The Power of Edge Networking

Security Groups and NACLs

Stateless Firewalls



Firewalls monitor traffic and only allow traffic permitted by a set of predefined rules

Stateless Firewalls

Firewall rules are broken down into inbound and outbound rules

IP/Port	Action
443	Allow
1024-65535	Allow

PORT: 443 Inbound

Listening on port 443

Outbound

IP/Port	Action
1024–65535	Allow
80	Allow

Stateless firewalls must be configured to allow both inbound & outbound traffic

PORT: 80

PORT: 80

Stateful Firewalls

Stateful firewalls are intelligent enough to understand which request and response are part of the same connection

IP/Port	Action
443	Allow
1024-65535	Allow

PORT: 443 Inbound

I know this a response to the original request

Listening on port 443

Outbound

PORT: 80

If a request is permitted, the response is automatically permitted as well in a stateful firewall

IP/Port	Action
1024–65535	Allow
80	Allow

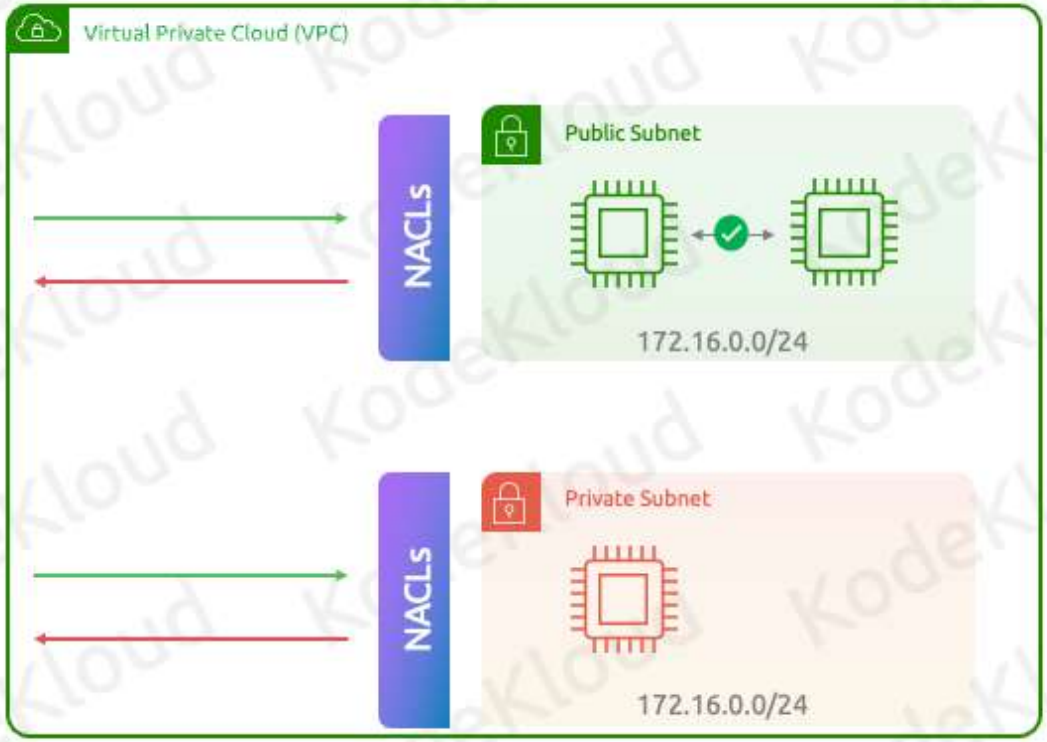
PORT: 80

Network Access Control List (NACL)

NACLs filter traffic entering and leaving a subnet

NACLs do not filter traffic within a subnet

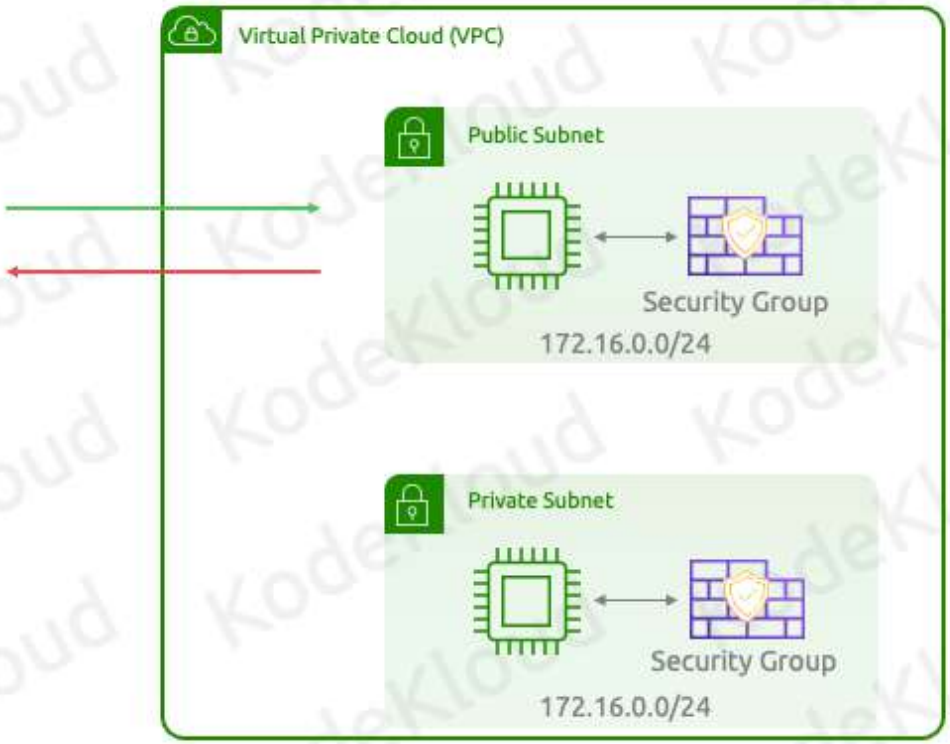
NACLs are stateless firewalls, so rules must be set for both inbound and outbound traffic



Security Groups

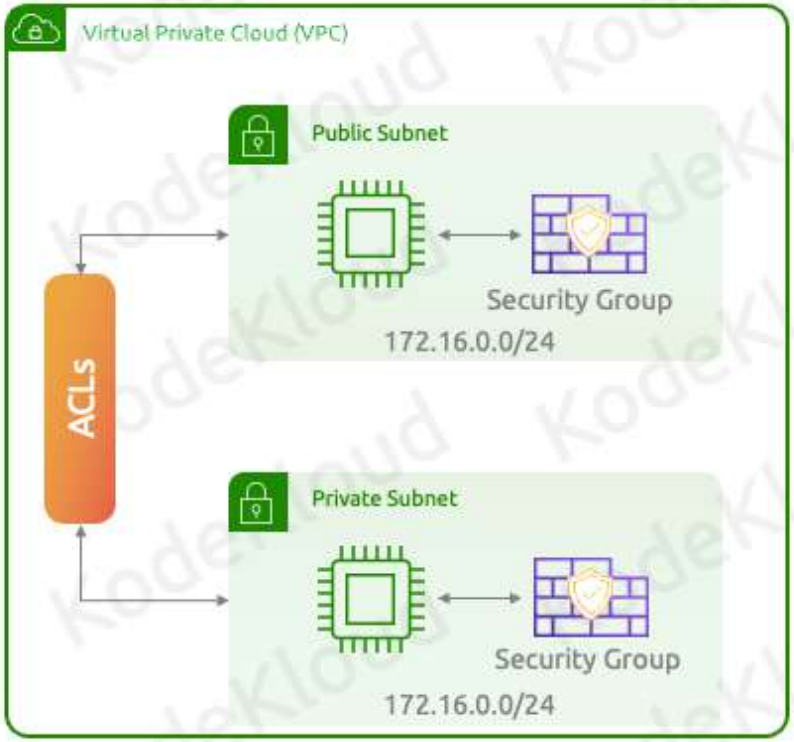
Security Groups act as firewalls for individual resources (EC2, LB, RDS)

Security Groups are stateful, so only the request needs to be allowed

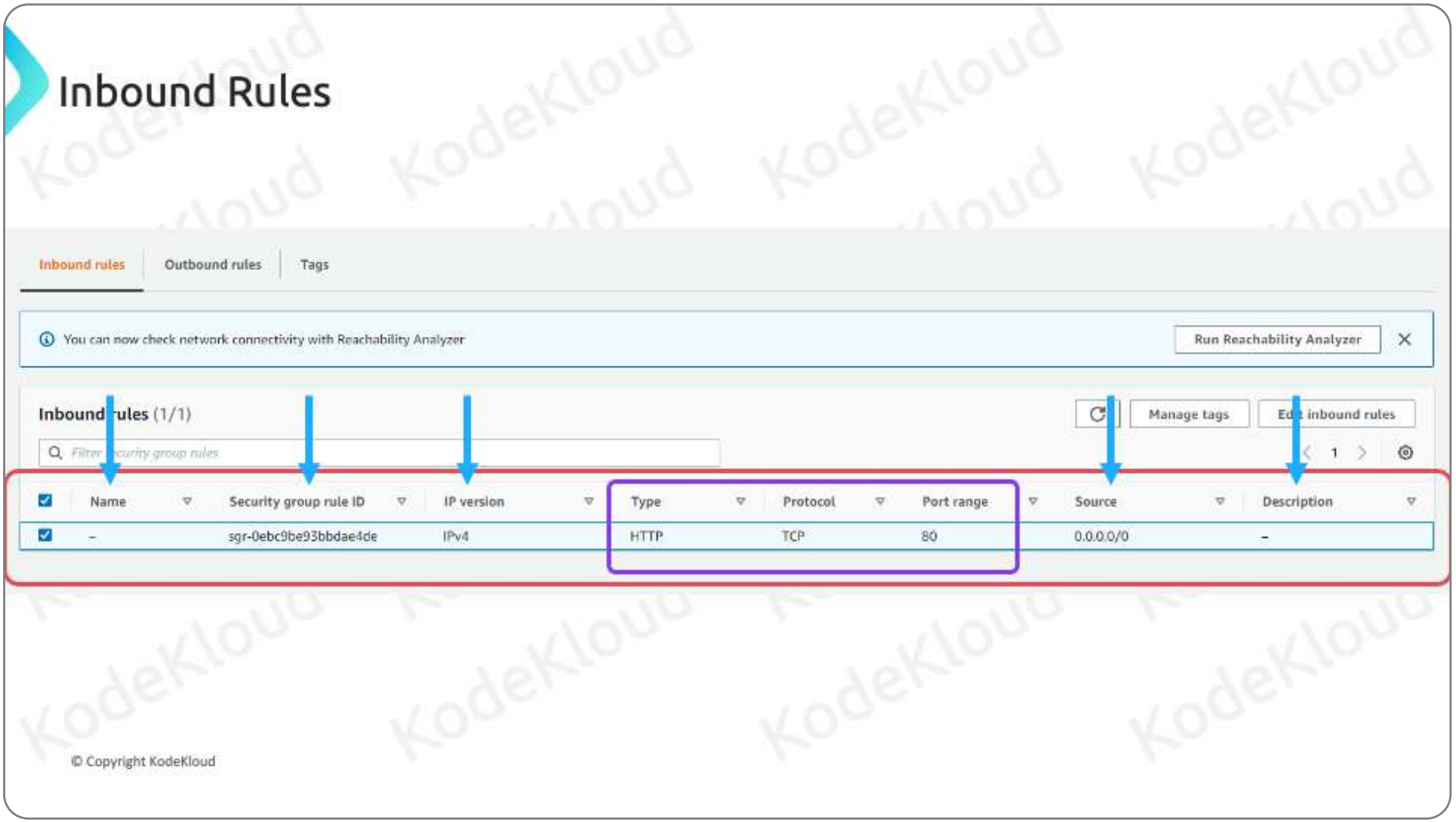


NACLs vs Security Groups

- NACLs monitor traffic entering and leaving a subnet
- NACLs are stateless firewalls
  - Traffic must be permitted in both ingress and egress directions
- Security Groups act as personal firewalls for individual resources
- Security Groups are stateful
  - Only the direction of the request needs to be permitted
  - The response will automatically be permitted as well



Configuring Security Group Rules



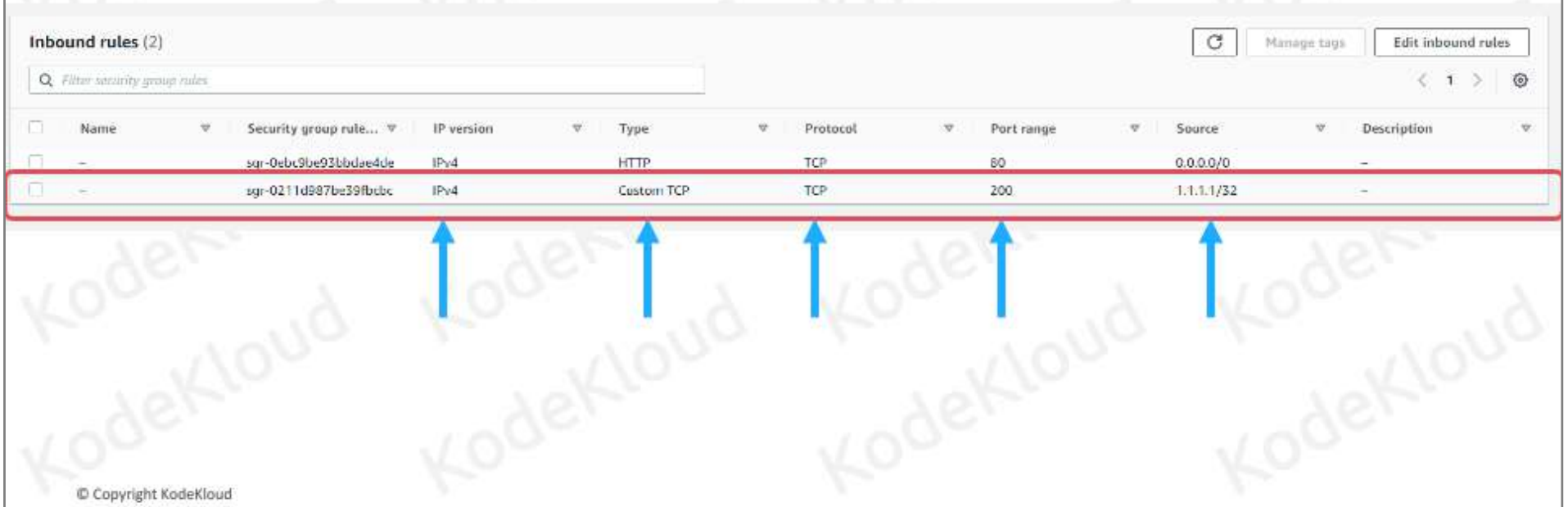
Under the security group we can take a look at the inbound and outbound rules. Here we are taking a look at the Inbound rules. So this is what is allowed inbound to a specific resource.

We can see here there is one rule.

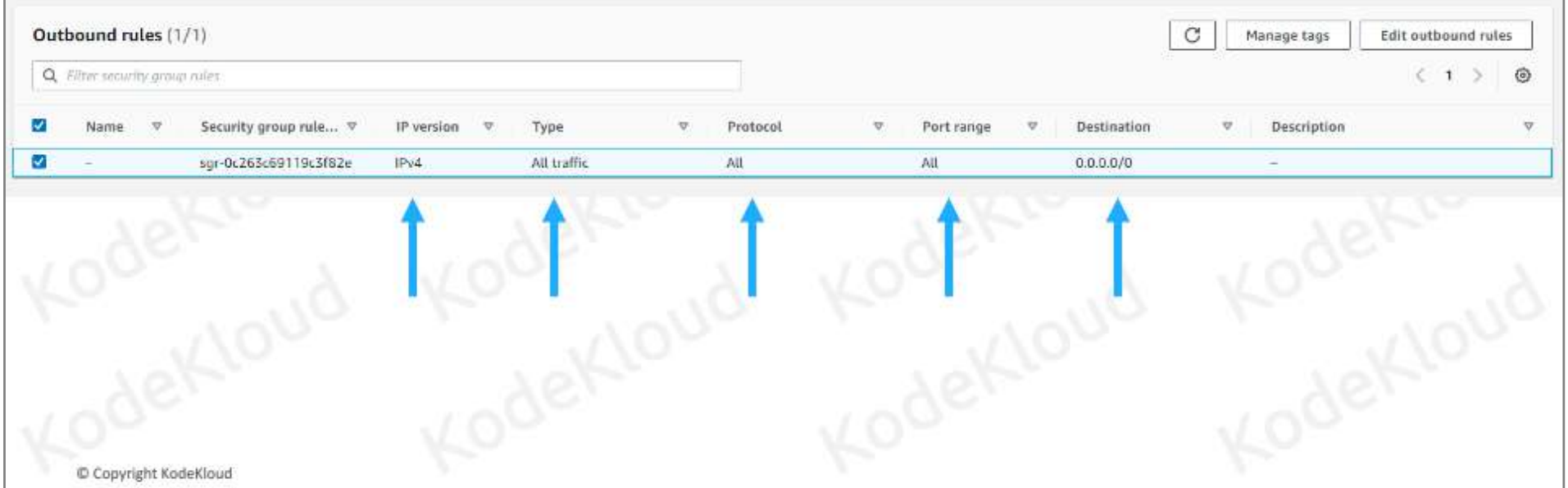
Let’s go over the columns

- Name – we can give the rule a name
- Rule id
- Ip version
- http/protocol/protocol/port range
- Source IP
- Description

Inbound Rules



Outbound Rules

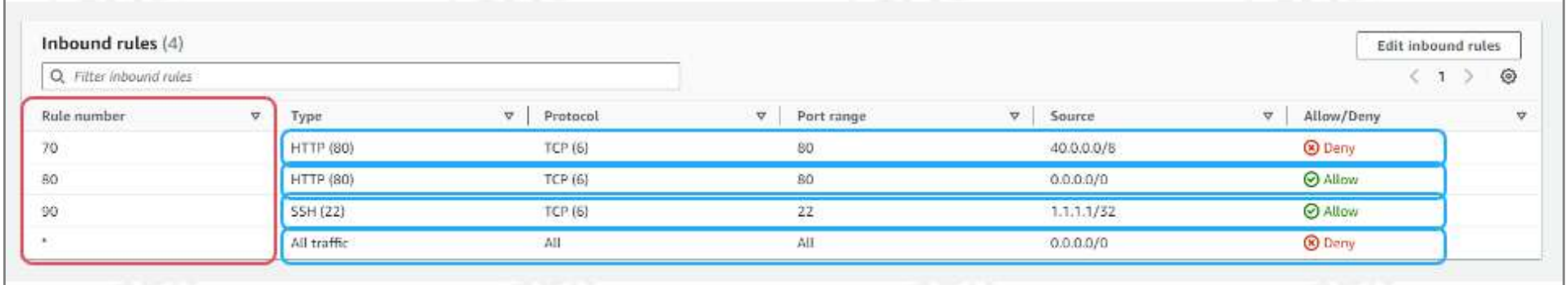


Security groups, when there are no rules, block everything.

- When you add a security group rule, it allows a certain type of traffic
- All security group rules "allow" traffic; there is no "deny" option for security groups

NACL rules can either allow or deny traffic.

NACL Rules



Multiple Security Groups

- You can assign multiple security groups to a single resource
- The rules for both security groups get merged

Security group: web

Port	IP
80	0.0.0.0/0
443	0.0.0.0/16

Security group: mgmt

Port	IP
3389	0.0.0.0/0
22	50.1.0.0/16

More than one security group can get applied to a single resource. When this happens rules from both security groups get merged into one.

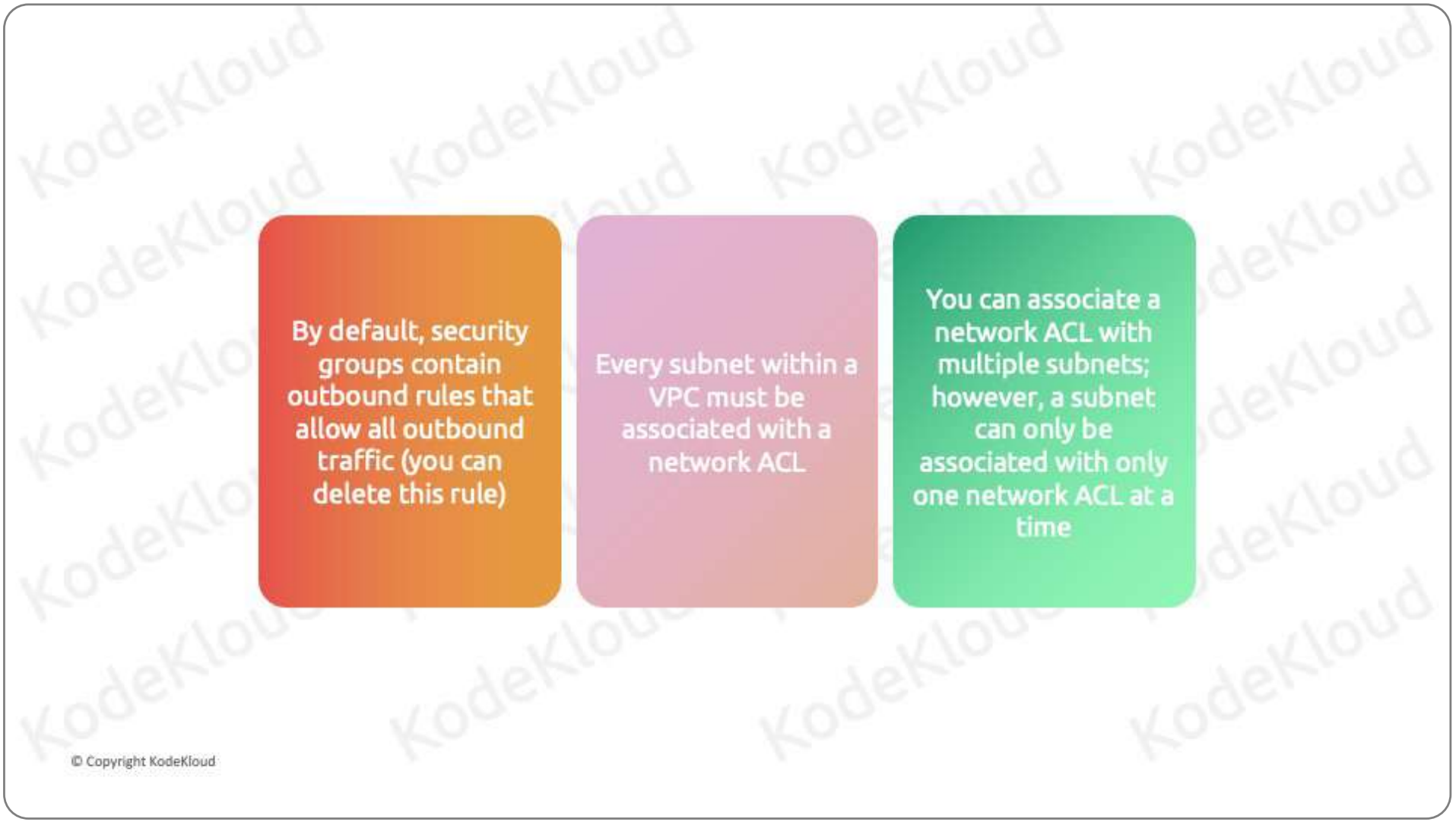
Multiple Security Groups

- You can assign multiple security groups to a single resource
- The rules for both security groups get merged

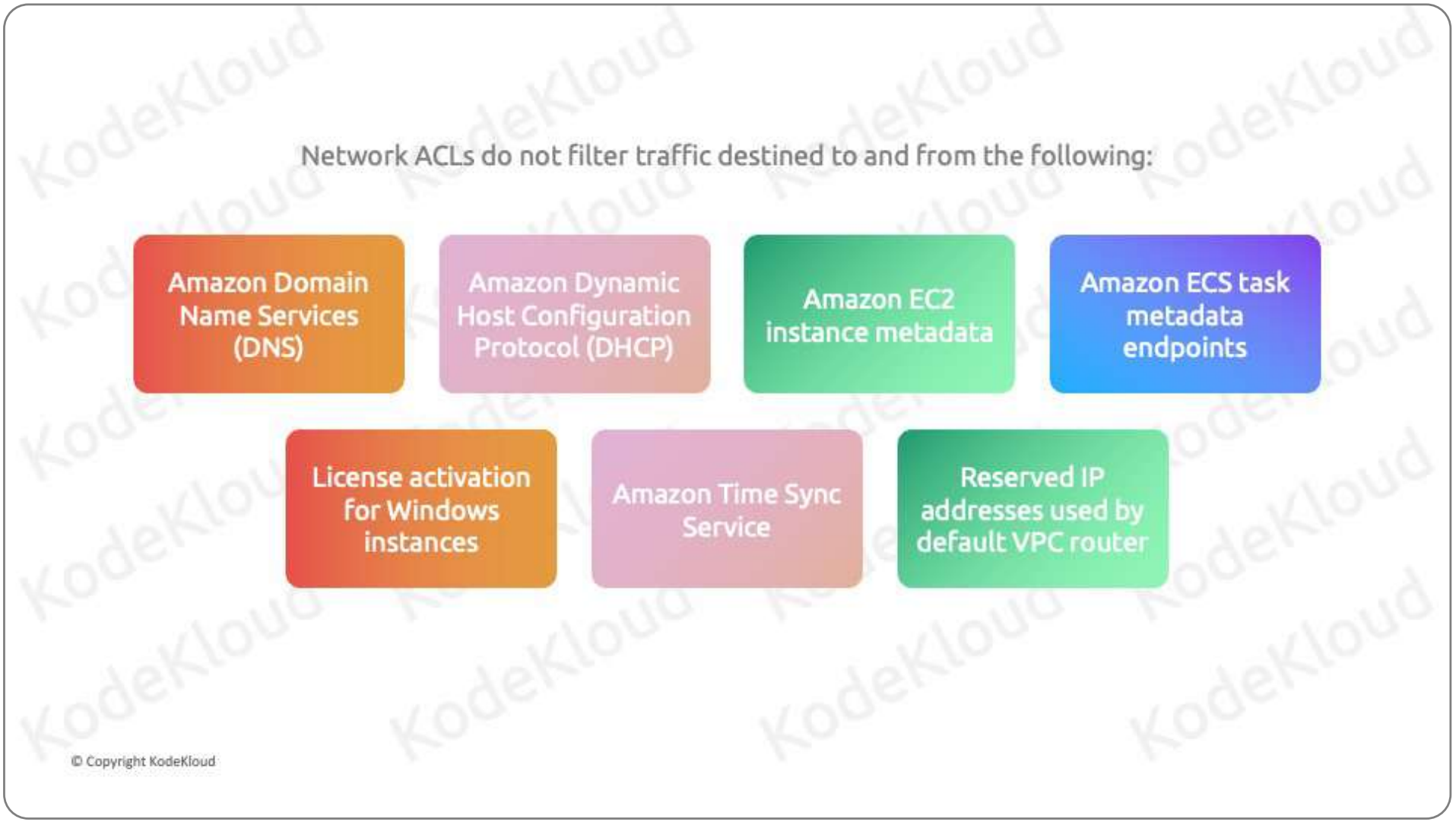
Security group: web + mgmt

Port	IP
80	0.0.0.0/0
443	0.0.0.0/16
3389	0.0.0.0/0
22	50.1.0.0/16

More than one security group can get applied to a single resource. When this happens rules from both security groups get merged into one.



- By default security groups contain outbound rules that allow all outbound traffic (you can delete this rule)
- Every subnet within a VPC must be associated with a network ACL
- You can associate a network ACL with multiple subnets, however a subnet can only be associated with only one network ACL at a time



Network ACLs do not filter traffic destined to and from the following:

- Amazon Domain Name Services (DNS)
- Amazon Dynamic Host Configuration Protocol (DHCP)
- Amazon EC2 instance metadata
- Amazon ECS task metadata endpoints
- License activation for Windows instances

Amazon Time Sync Service
Reserved IP addresses used by the default VPC router

Summary

01 Stateless firewalls require traffic to be explicitly permitted inbound and outbound

02 Stateful firewalls are intelligent firewalls that track requests and allow response

03 Network ACLs filter traffic entering and leaving a subnet

04 Network ACLs are stateless firewalls

Summary

05 Security Groups act as firewalls for individual resources such as EC2, NICs, and other network objects

06 Security Groups are stateful firewalls

07 Security Group rules only "allow" traffic, whereas NACL rules have the option to "allow" or "deny"

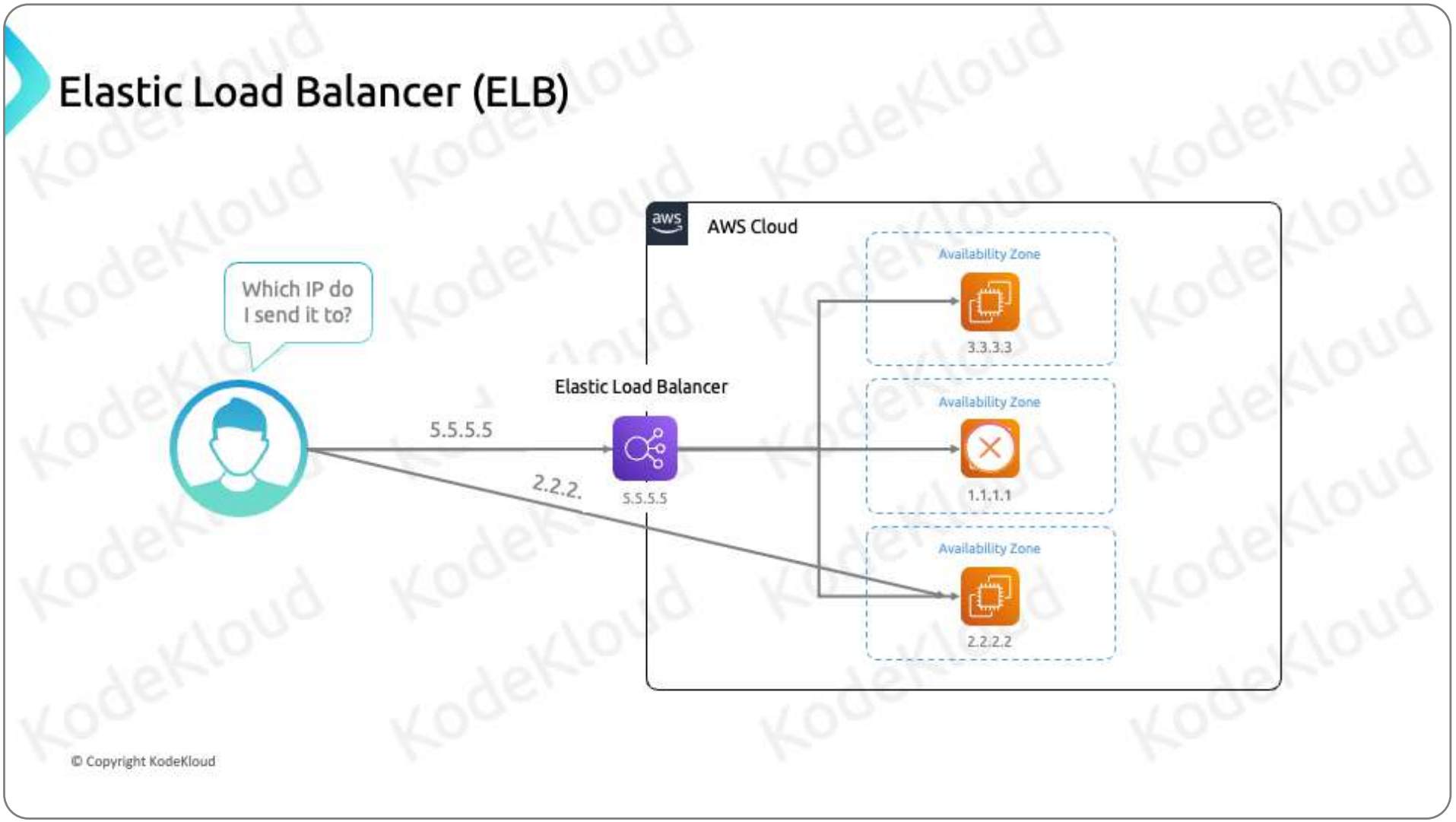
08 Multiple Security Groups applied to a resource will have their rules merged

Summary

09 Every subnet within a VPC must be associated with a network ACL

10 You can associate a network ACL with multiple subnets; however, a subnet can only be associated with only one network ACL at a time

Welcome to the section on Cloud Computing.



Let's say that we have a web application deployed on an ec2 instance. The EC2 instance has an ip of 1.1.1.1 and our end users can access the applications by sending a request to that ip(1.1.1.1) If the EC2 instance were to go down then our application would be completely down for our users.

So what would we do to get around this issue? Well we would deploy our application on several EC2 instances for redundancy. So now we have 3 servers runnign the app. And Ideally we would have them deployed on different AZ zones so

that our application can tolerate an AZ failure

There’s just one problem, each server is going to have a different public IP(1.1.1.1, 2.2.2.2, 3.3.3.3). So what IP does the user send requests to?

Would we give the end users just one IP like the 1.1.1.1? Well what happens if that server goes down? We’d have to tell them to send it to a different IP. End users shouldn’t have to worry about this.

So a better solution than having to give all of your server Ips to the end user would be to have a separate device sitting infront of all of our servers. The user would send requests to this device and it would then load-balance the requests to the available ec2 instances. This device is referred to as a load-balancer. The benefit of doing this that the only IP the end users would need to know about would be the IP of the load balancer

Load Balancers in AWS – Types

01

Classic Load Balancer

02

Application Load Balancer

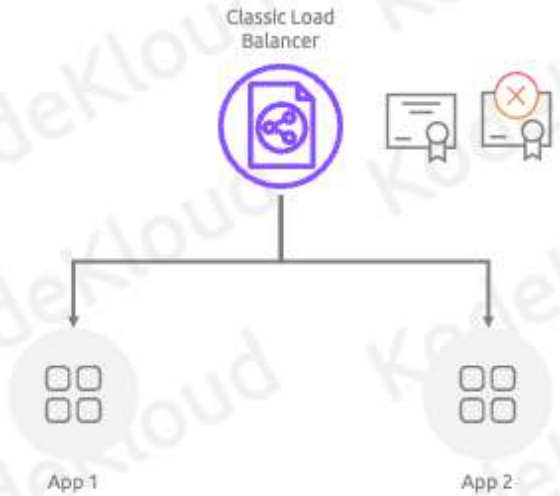
03

Network Load Balancer

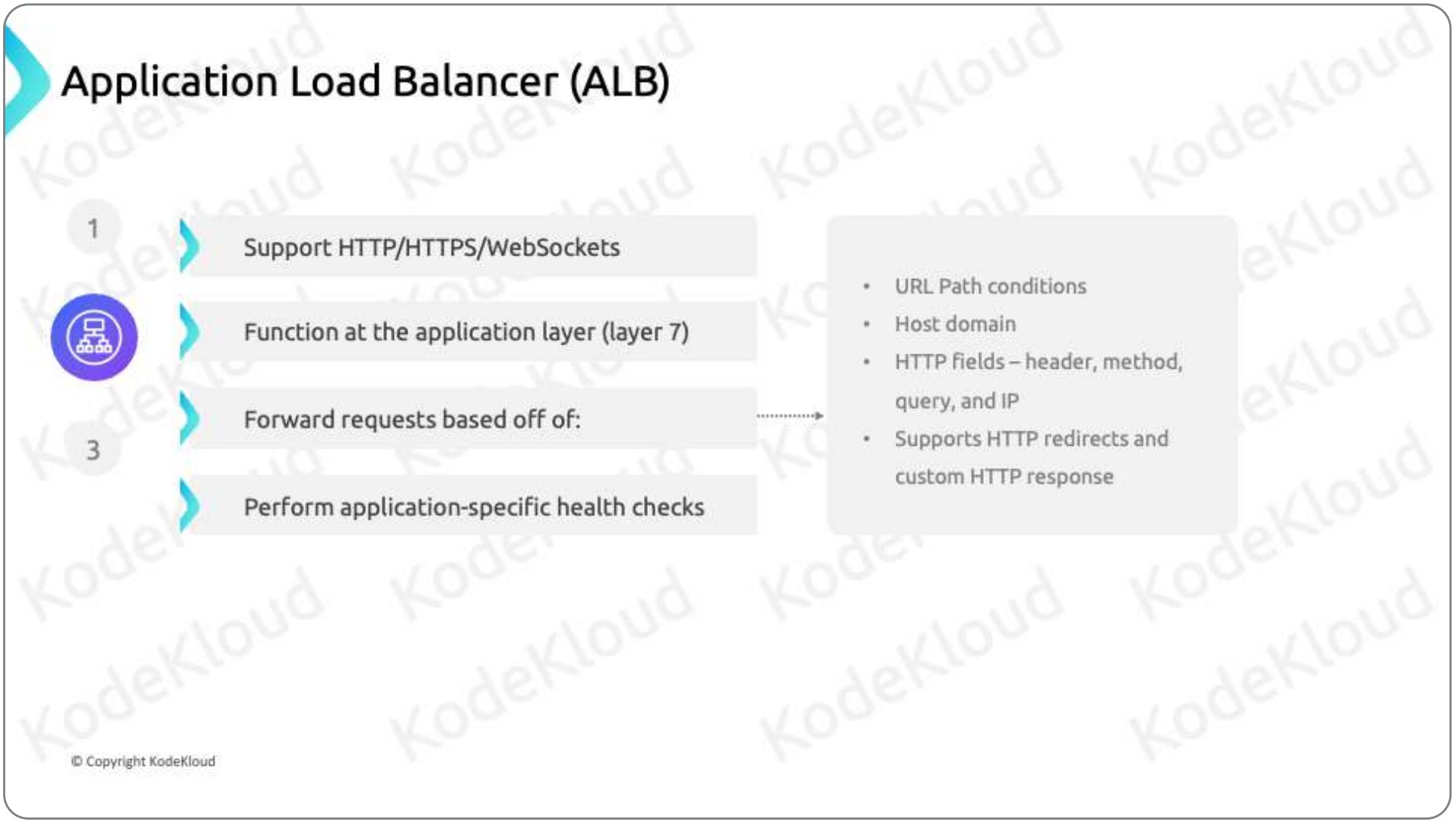
Classic Load Balancer (CLB)

2 First load balancer introduced by AWS

3 Not recommended to use

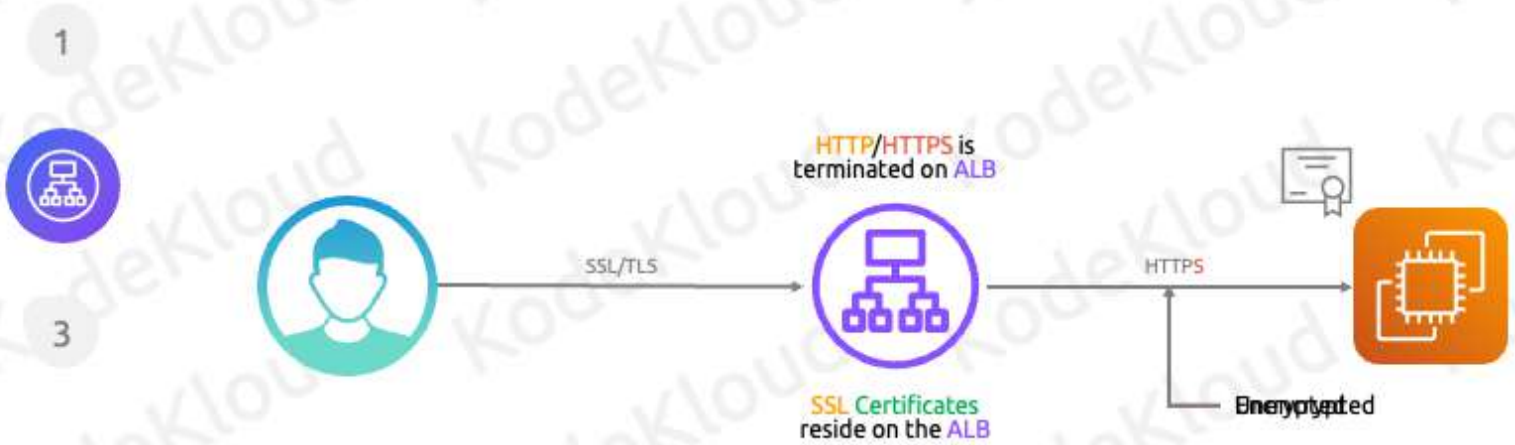


Has limited features
Only allows 1 SSL certificate per Classic Load Balancer



Can inspect all of the layer 7 protocol information

Application Load Balancer (ALB)

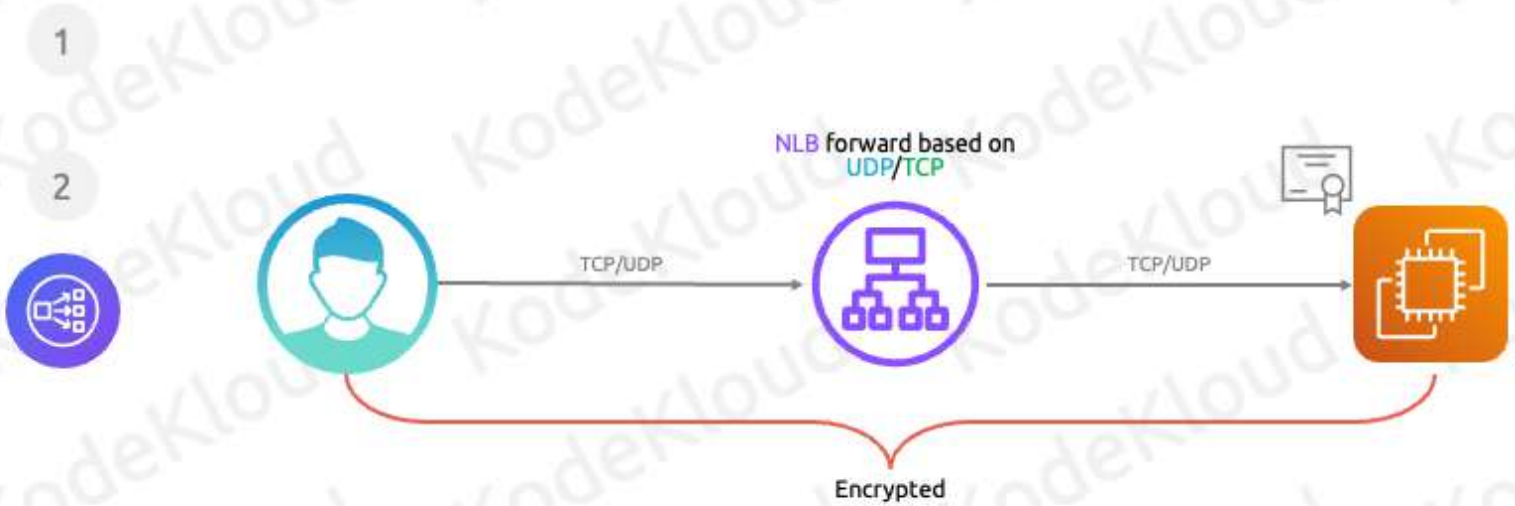


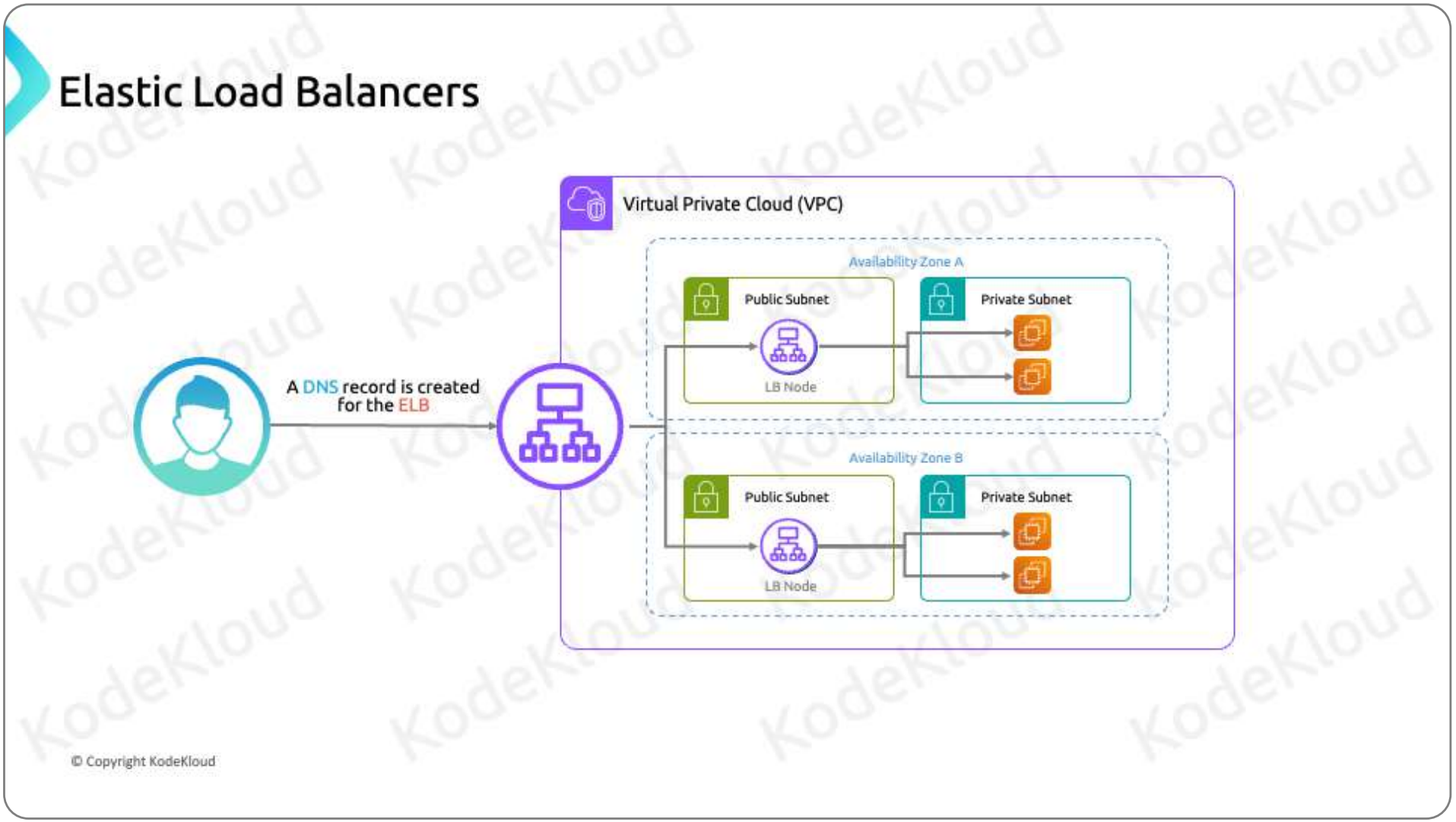
Network Load Balancer (NLB)

1. Load balance traffic based on TCP/UDP (layer 4)
2. Meant for applications that don't use HTTP/HTTPS

- Faster than Application load balancers
- Health checks are only basic ICMP/TCP connections
- NLB forwards TCP connections to instances

Network Load Balancer (NLB)





Let’s say we have a VPC and we have 2 availability zones

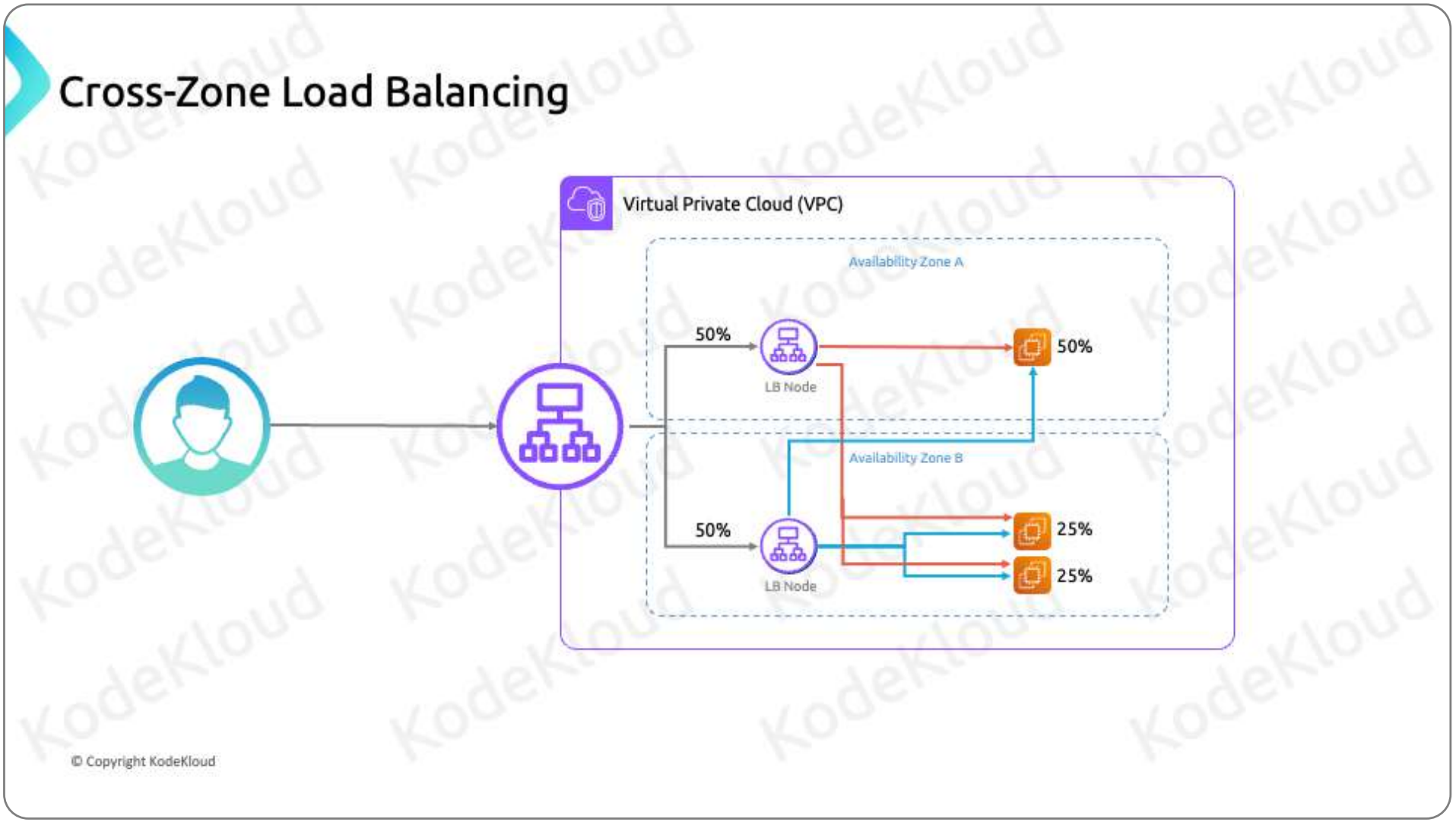
Now when we deploy a load balancer we have to specify what availability zones the load balancer should load balance traffic to. More specifically you are specifying subnets that you want to deploy your load balancer

Now what happens when we deploy load-balancers to an a subnet is that AWS will deploy a lb node to each specificied

subnet.

So in the aws console when you see a single load balancer object, it is actually made up of multiple load-balancer nodes which live within the subnets you pick.

A DNS record is created for ELB which resolves to all of the load balancer nodes associated with that elastic load balancer at which point then can then forward it to an isntance



Let’s say we have a load balancer deployed across 2 Azs that means we’ll have lb nodes deployed in each AZ.

When a user sends a request to the DNS name for the Elastic loadbalancer, and the DNS entry will load balance equally across all of the load balancer nodes.

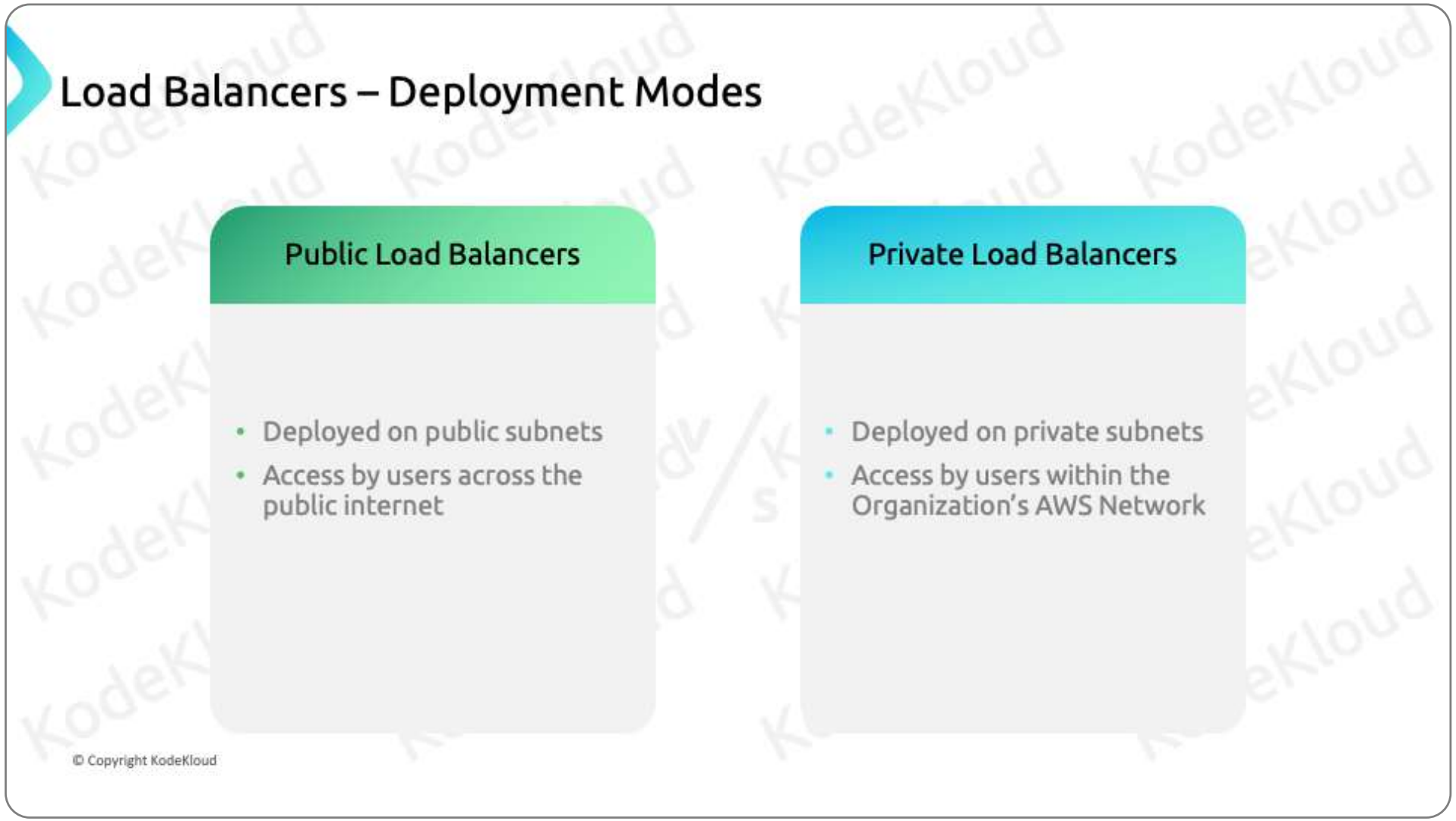
We have 1 LB node in each AZ so 2 total LB nodes which means each nodes gets 50%. Now initially load balancer nodes

could only distribute connections to instances in the same AZ.

To fix this a feature called Cross-zone LB was created. The idea is simple, it allows load balancer nodes to distribute connections equally across instances in all AZ.

This feature is now enabled by default.

So if you get a question about unequal distribution across your isntances on the exam doublechekc on cross zone toggle



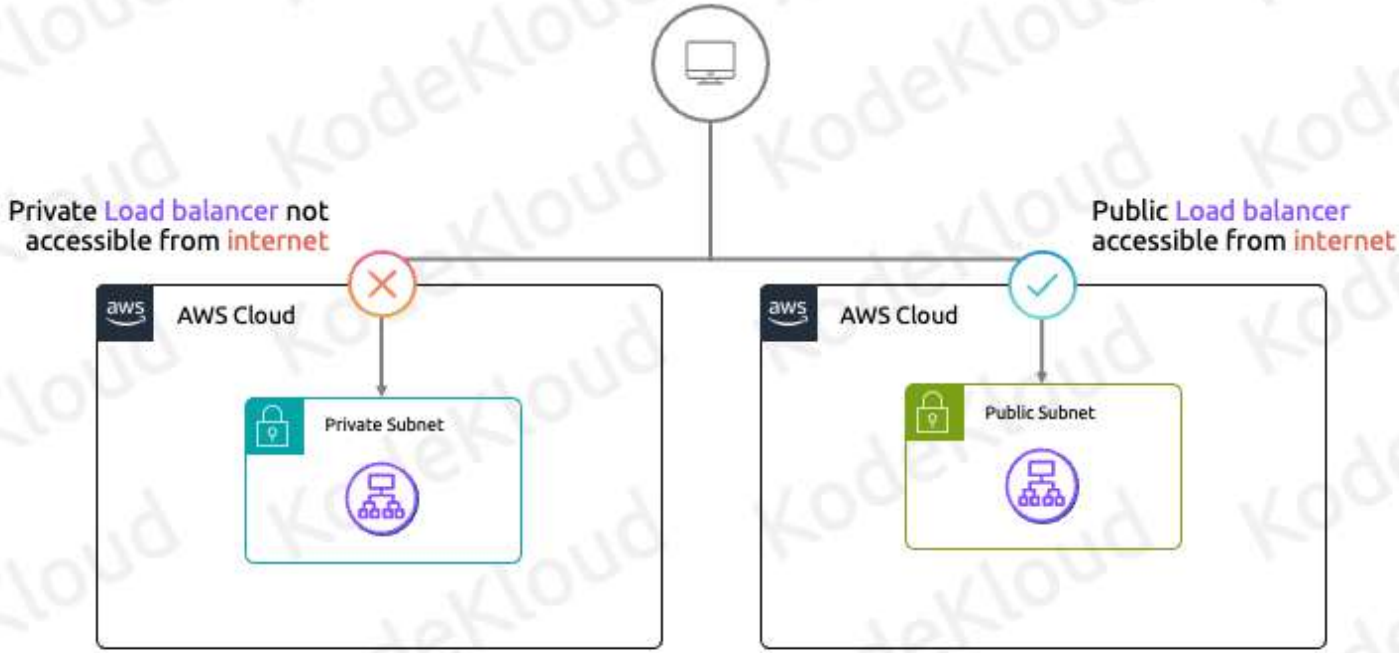
Load balancers can be deployed in a similar fashion to EC2 instances

- Public
- Private

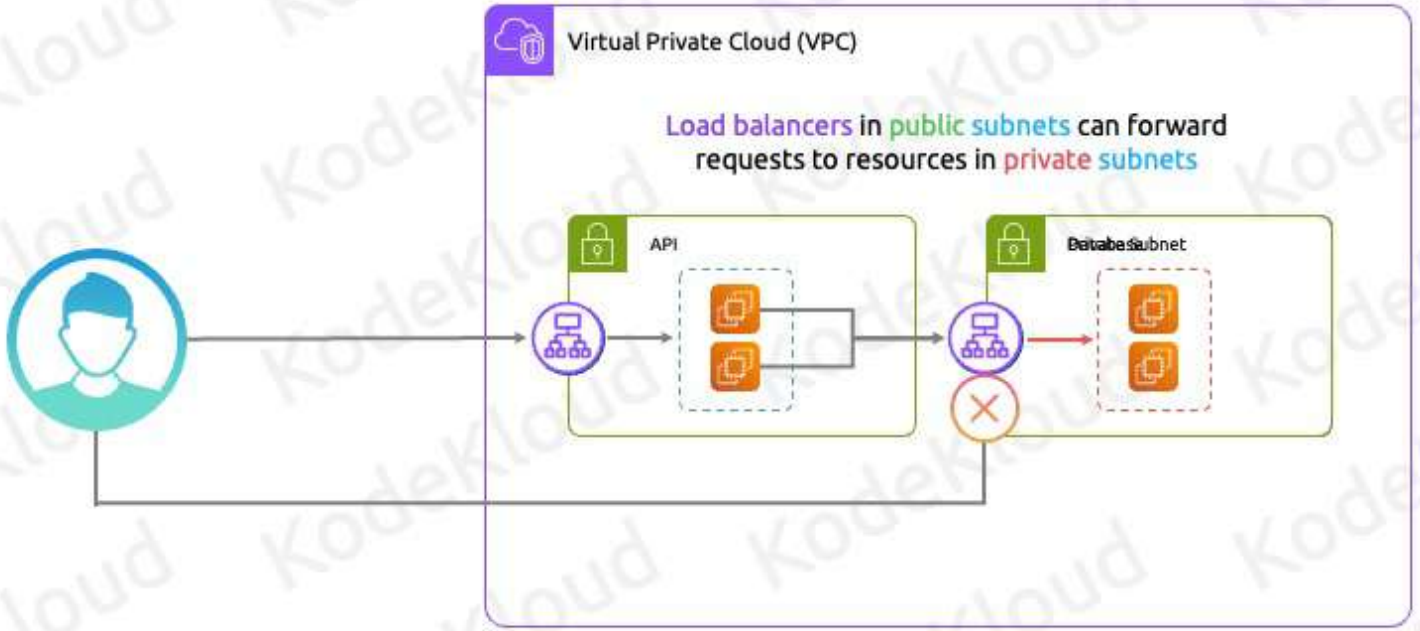
Public load-balancers are load balancers deployed on public subnets so it is public facing and the internet can access it

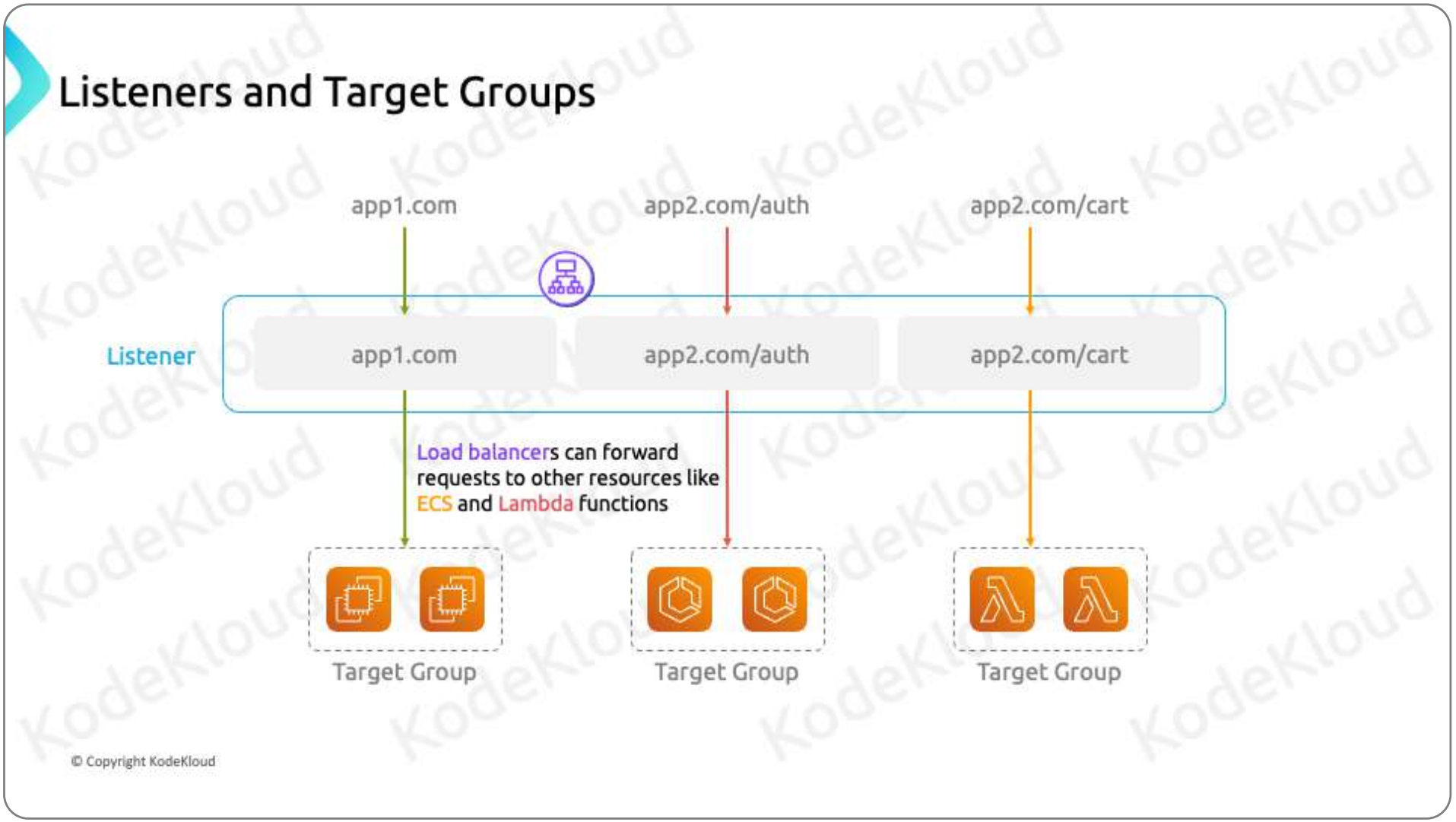
Private load balancers are load balancers deployed on private subnets and can only be accessed from within AWS

Private vs Public Subnets



ELB Architecture





A listener is a process that checks for connection requests, using the protocol and port that you configure.

Target groups route requests to one or more registered targets, such as EC2 instances, using the protocol and port number that you specify. You can register a target with multiple target groups. You can configure health checks on a per target group basis. Health checks are performed on all targets registered to a target group that is specified in a listener rule for your load balancer.

Summary

01 ELB auto-distributes incoming traffic across multiple targets in multiple Availability Zones

02 Distributes traffic to EC2 instances, IPs, and Lambda functions

03 Three types of ELB: Classic, Application, Network

04 CLBs are outdated and should be avoided

Summary

05 ALB functions on application layer and forwards HTTP/HTTPS traffic

06 NLB can forward non-HTTP/HTTPS traffic

07 NLBs are faster than ALBs

08 HTTP/HTTPS is always terminated on ALB and not Network Load Balancer

Summary

09 Cross-Zone Load Balancing equally distributes traffic across all instances in all AZs

10 A listener matches traffic based on rules and handles connection requests

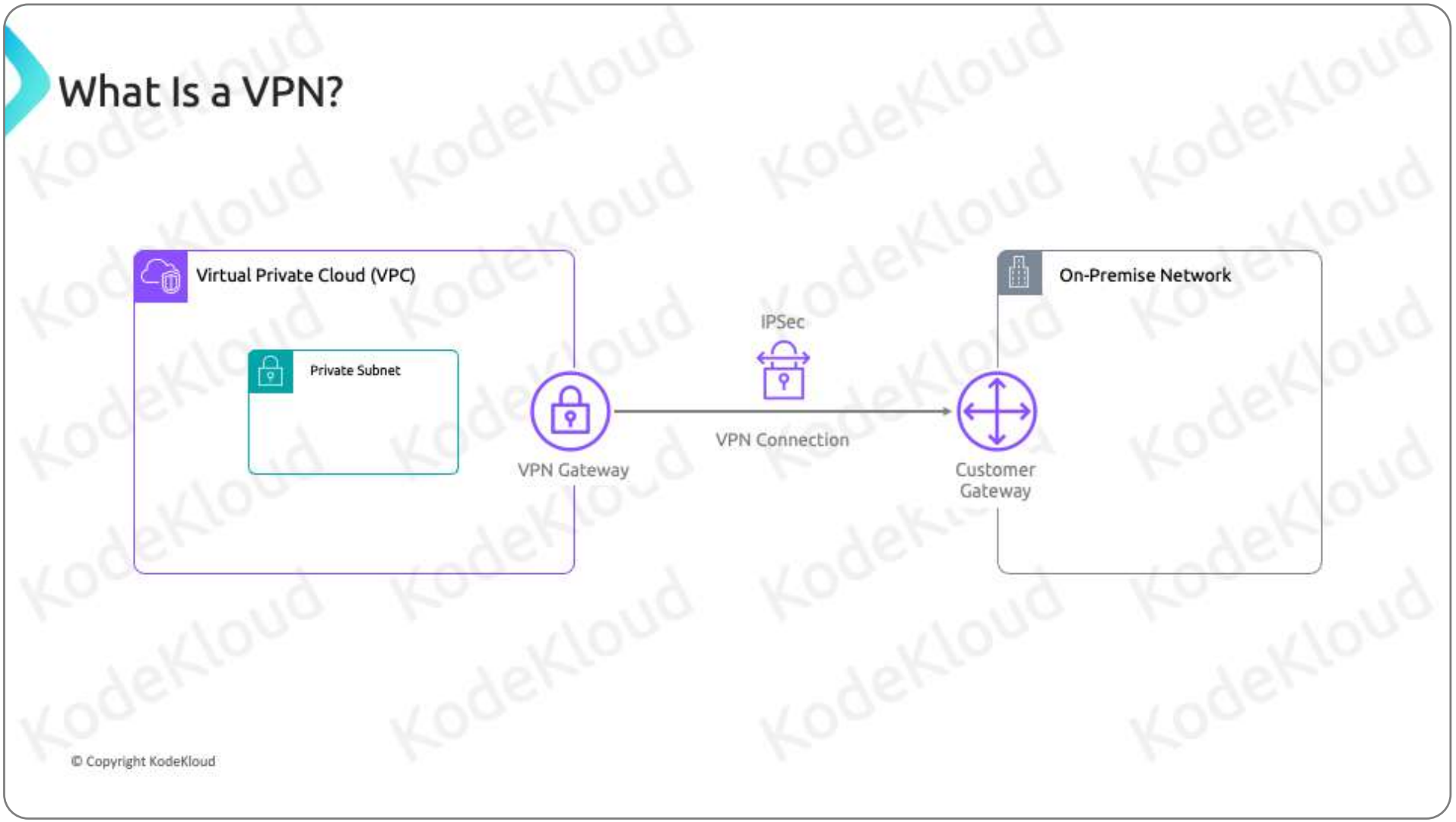
11 Target groups route requests to registered EC2 instances based on specified protocol and port

Service Section

The Power of Edge Networking

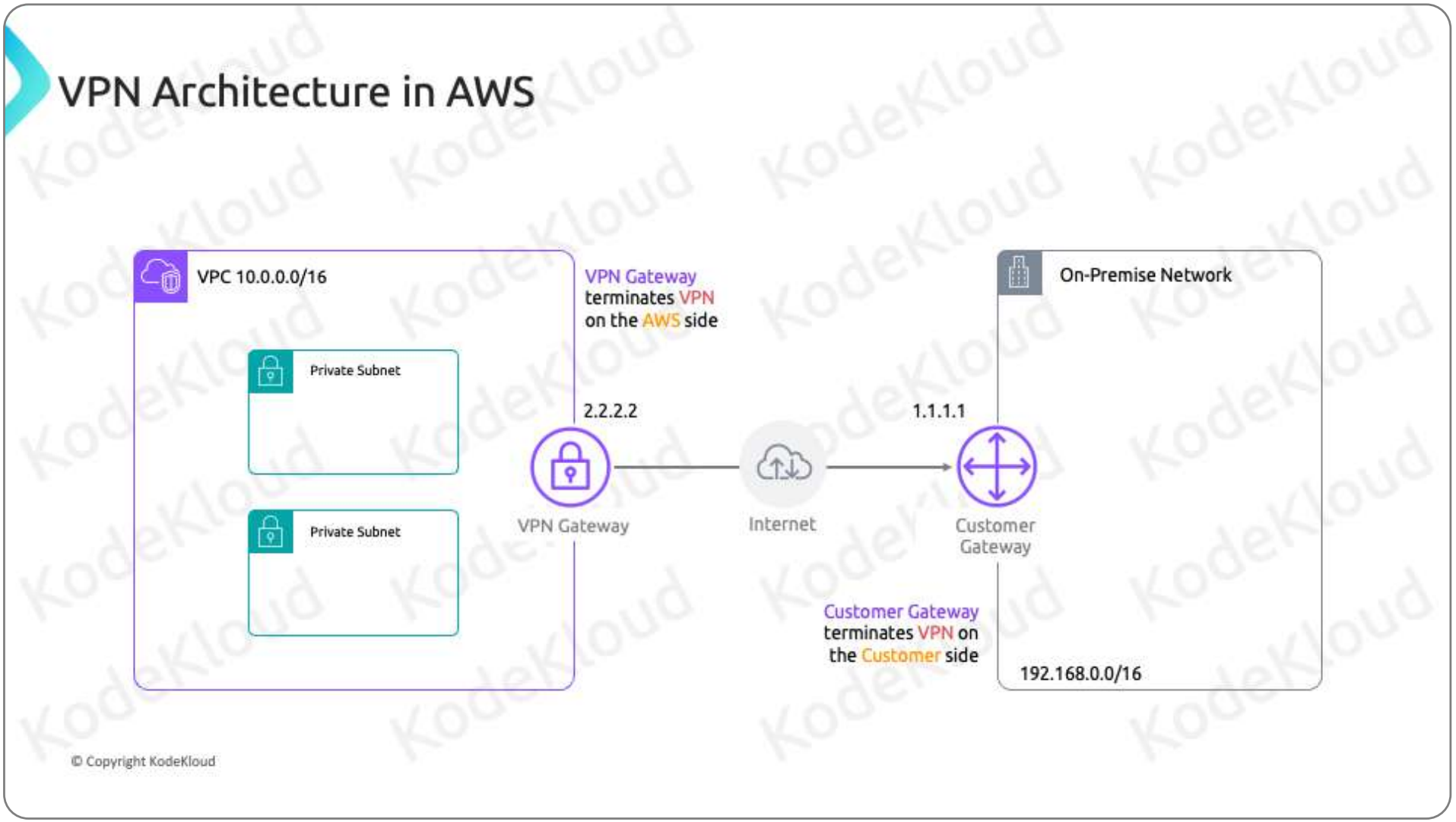
VPN

What Is a VPN?



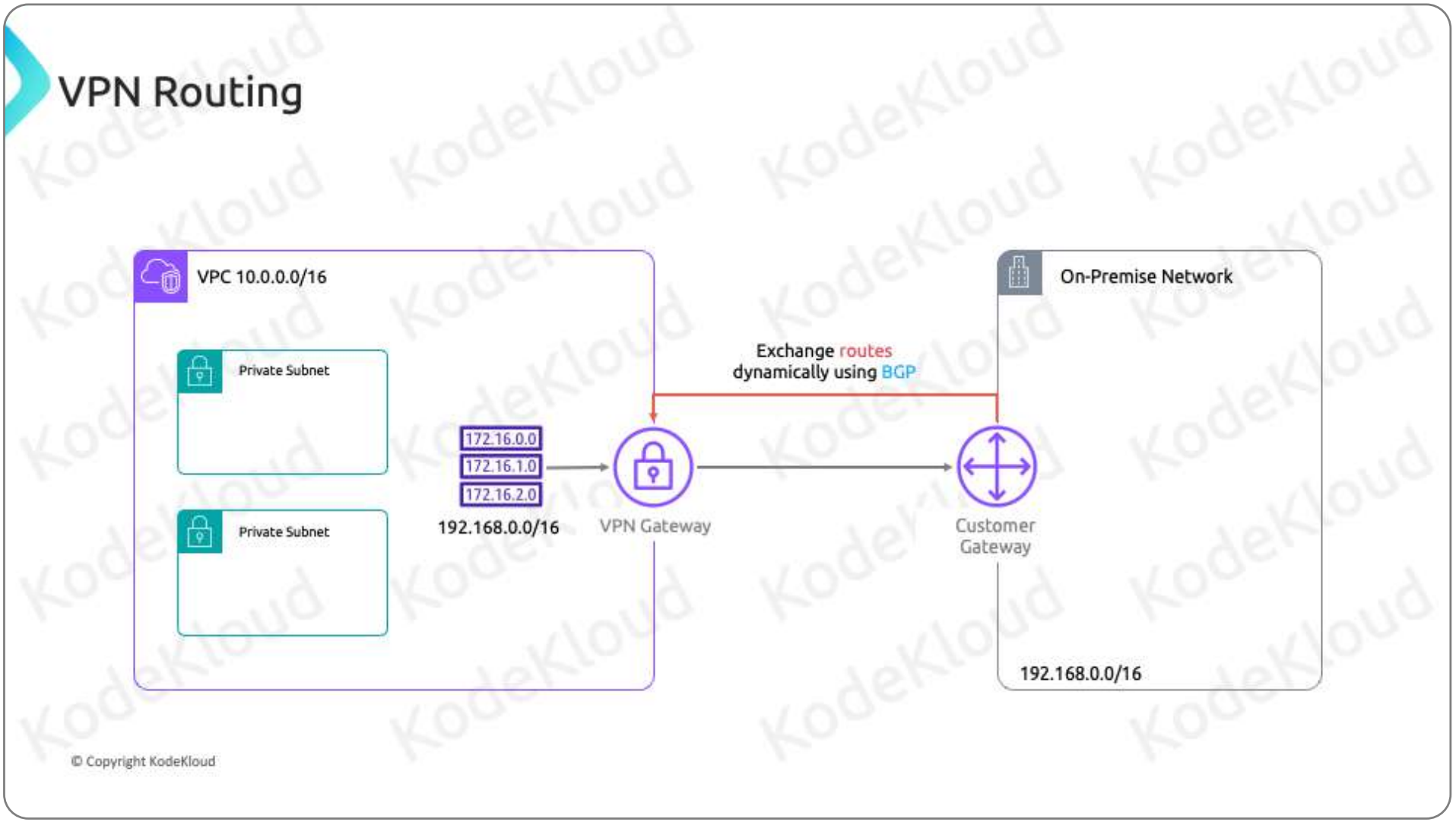
By default devices in a VPC more specifically devices in a private subnet cannot communicate on-premise network.

VPN connections connect resources in VPCs to on-premise network using encrypted IPSec tunnels over the public internet



By default devices in a VPC more specifically devices in a private subnet cannot communicate on-premise network.

VPN connections connect resources in VPCs to on-premise network using encrypted IPSec tunnels over the public internet



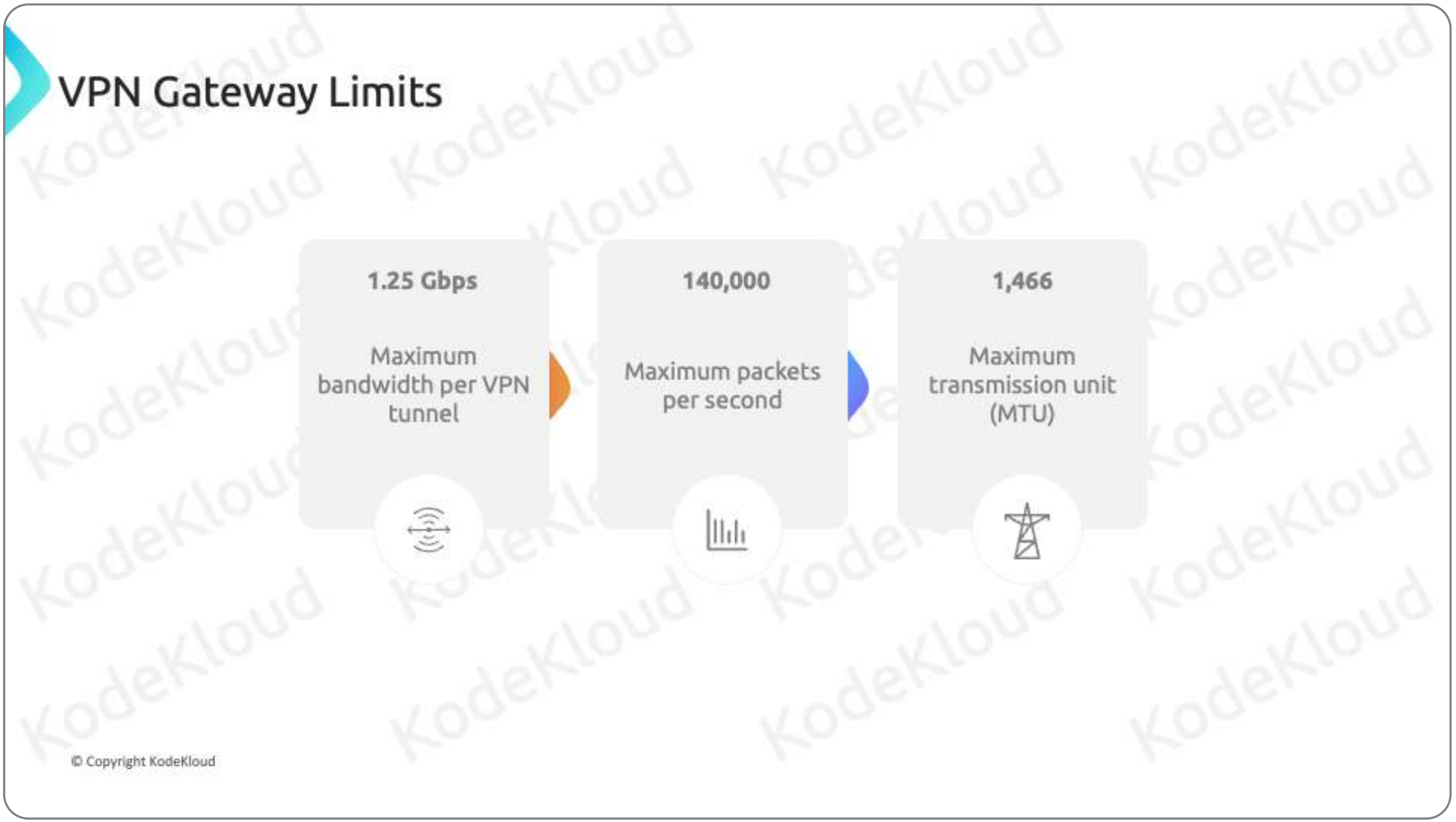
Just because we have a VPN between AWS and on-Prem how does the VPC know how to get to the on premis network 192.168.0.0/16?

Well we have two options:

1. Static – manually define a route in the AWS routing table for the 192.168.0.0/16 pointing to the vpn gateway

2. Dynamic – we use a dynamic routing protocol that will automatically exchange routes for the networks

- You are charged for each VPN connection hour that your VPN connection is provisioned and available
- You are charged for data transfer out from Amazon EC2 to the internet



- Maximum bandwidth per VPN tunnel is up to 1.25 Gbps
- Maximum packets per second - 140,000
- Use ecmp across multip VPN tunnels to increase bandwidth
- Maximum transmission unit (MTU) – 1466

Summary

01 Connect VPCs to on-premise data centers

02 A virtual private gateway is the Amazon-side endpoint for your VPN, attachable to one VPC

03 Customer Gateway is your side's physical device or software in a VPN connection

04 Traverses over the public internet

Summary

05 On-premise network can be set statically in a route table or dynamically exchanged via BGP

06 Charged per hour for every VPN connection as well as egress data out

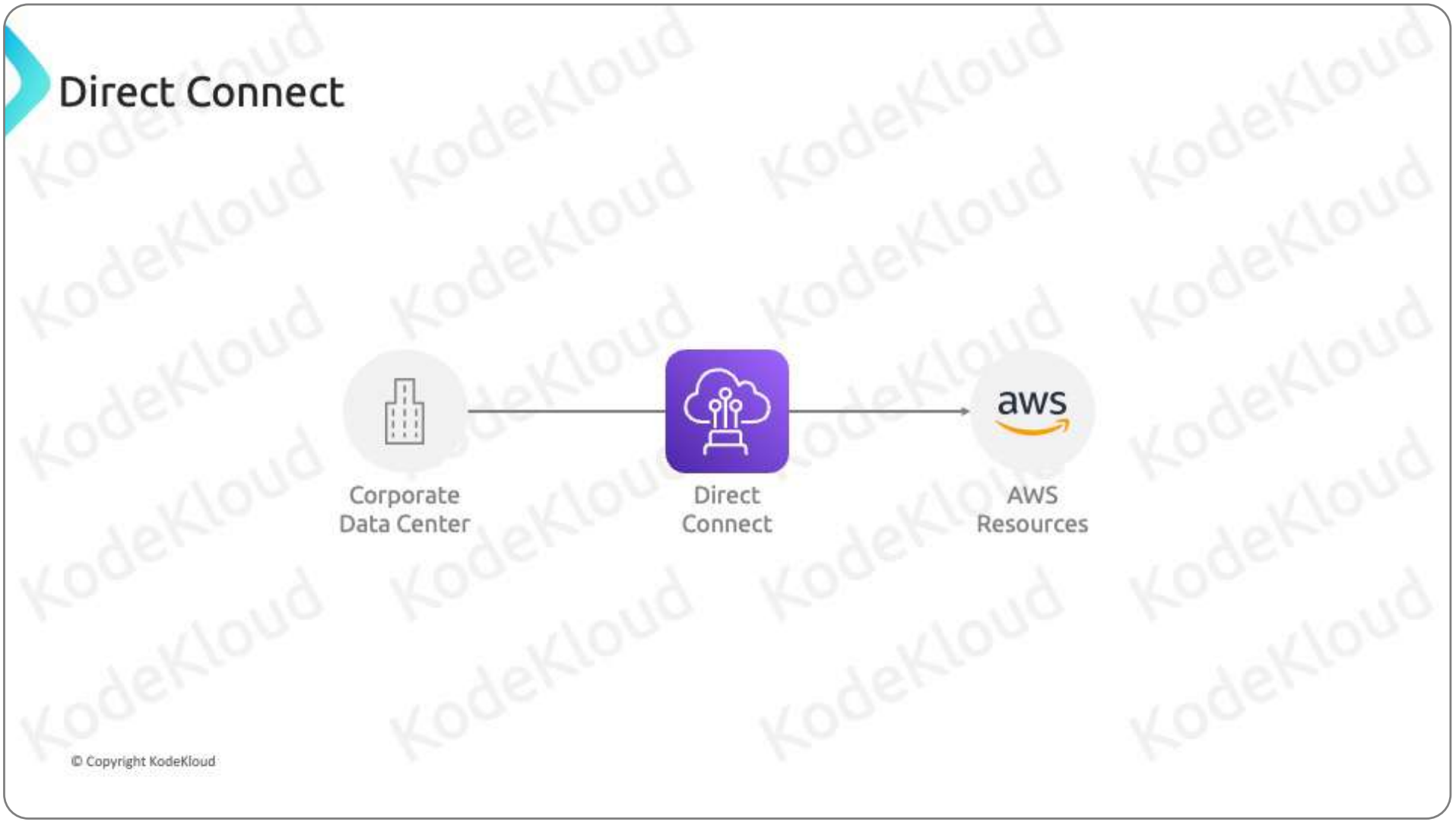
07 Max bandwidth 1.25 Gbps

Service Section

The Power of Edge Networking

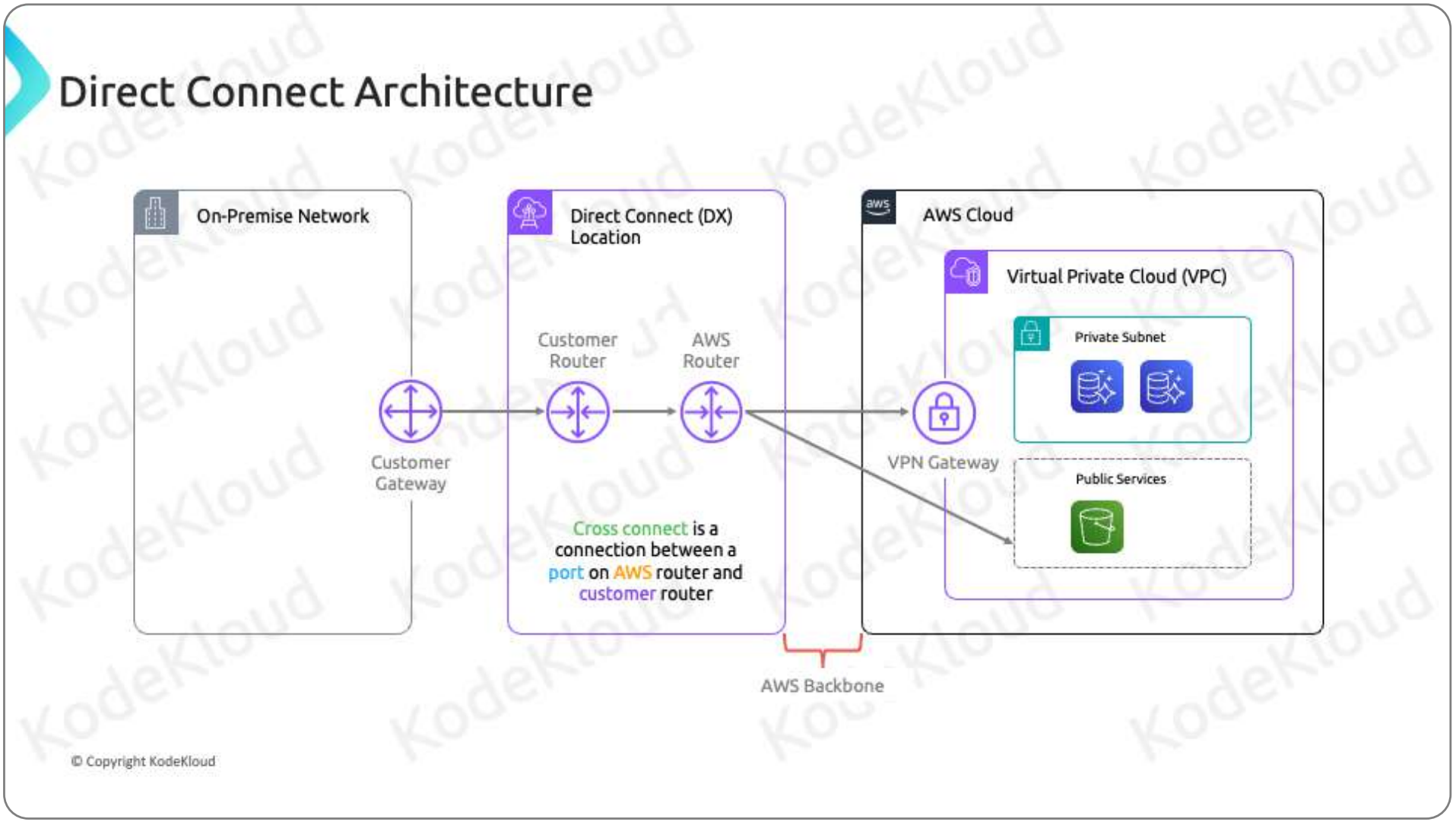
Direct Connect

What Is Direct Connect?



Direct connect is a physical connection into an AWS region. Think of this as an alternative to using a VPN.
Low consistent latency and high speeds can get 1 or 10 or 100gb speeds

VPNs go over the internet which can be unstable and experience congestion
Whereas with direct connect you have a direct connection into AWS



So let’s talk a look at the direct connect architecture.

There’s 3 main components:

You have your physical data center or corporate office. There you will have an edge router or firewall which can be the same router you use for internet connection or you can have a dedicate router for direct connect. This is up to you and your

organization on how you want to set that part up.

In the middle we have a Direct Connect Location. Which is really nothing more than a location where we can connect to an Amazon router. The direct connect location isn't usually owned by amazon. It's usually a regional data center where AWS rents space and so they have routers in there in which we can connect to.

In the Direct connect location you as the customer will have a router installed in the location. This is referred to as the customer router.

On the other side we have the AWS Direct Connect router. This is a router owned by amazon.

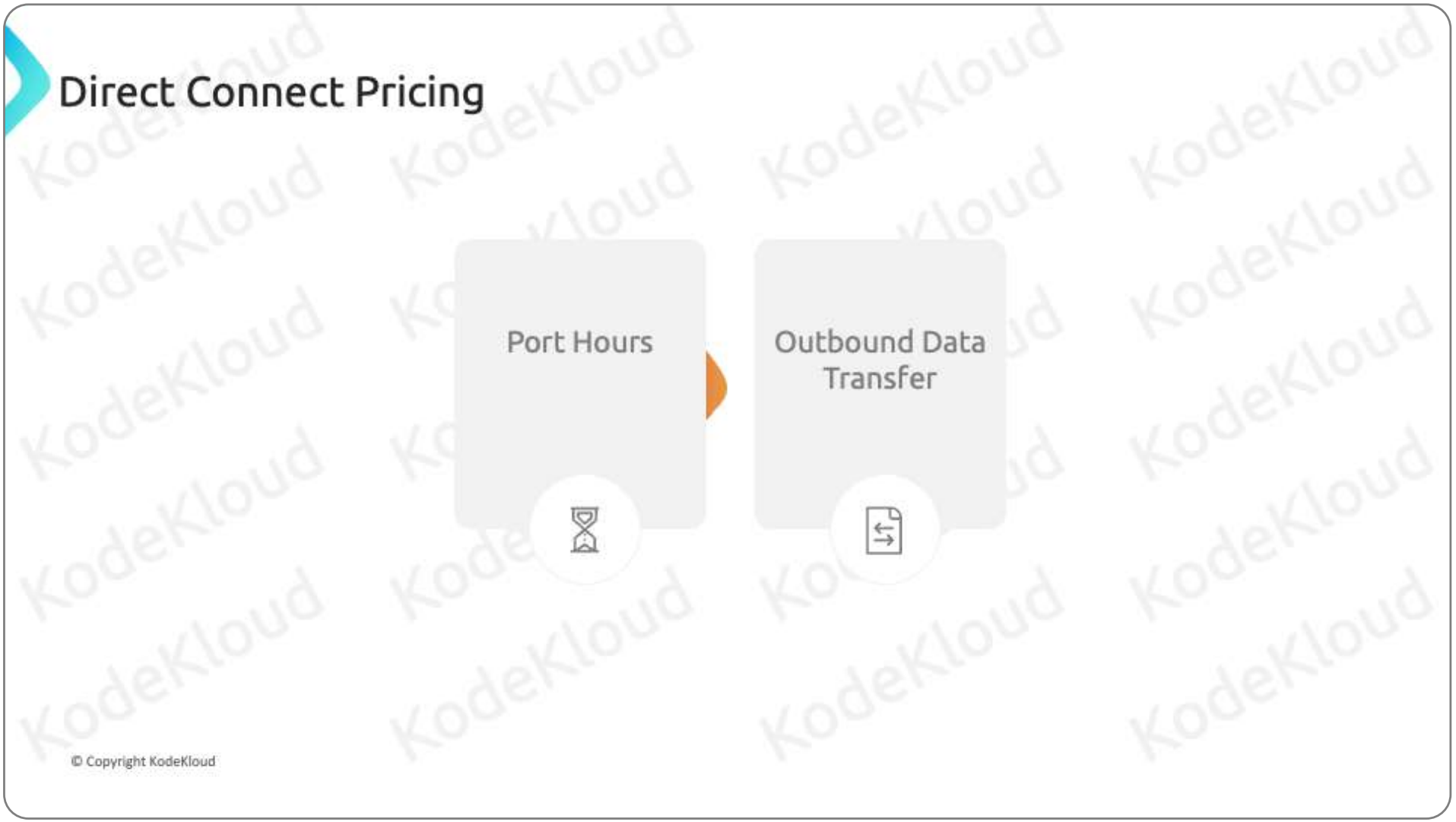
Now what you are purchasing when you order direct connect is a port on the AWS direct connect router.

So you then connect a port on the customer router to the port on the direct connect router to connect to AWS network.

Cross connect is a connection between a port on AWS router and the customer router

And we from the corporate datacenter we get a connection to the customer router in dx location which goes to the aws router which now has a direct connection to aws. So we can go directly to Aws public services

TO access private VPCs the aws router has which attach to a VPC over virtual private gateway



AWS Direct Connect has two billing elements:
port hours
outbound data transfer.

Summary

01 Directly links on-premises with AWS without internet routing like a VPN

02 Offers greater throughput and a more secure, stable connection than VPN over the internet

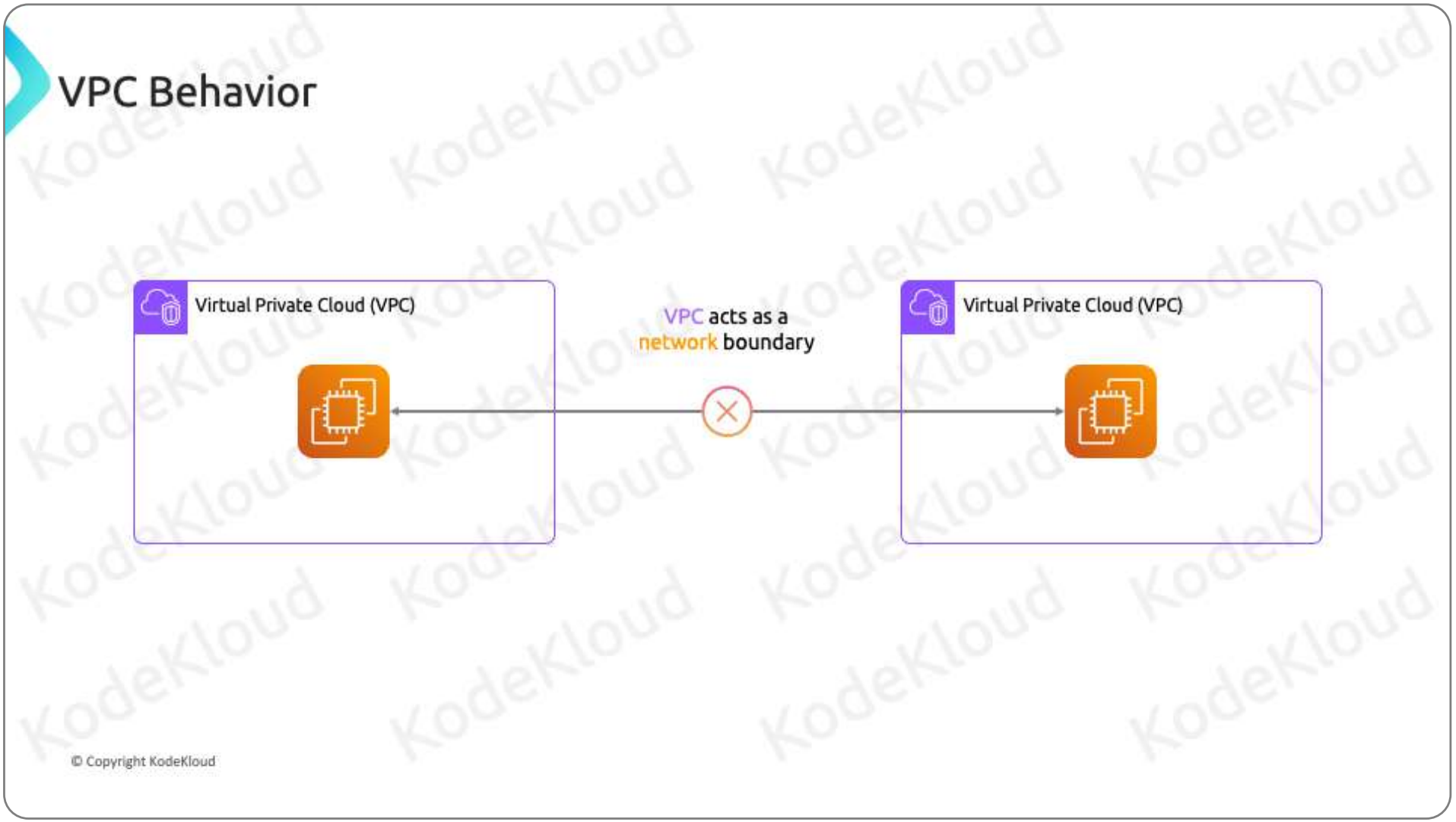
03 Charges per Port hour and Outbound data transfer

Service Section

The Power of Core Networking

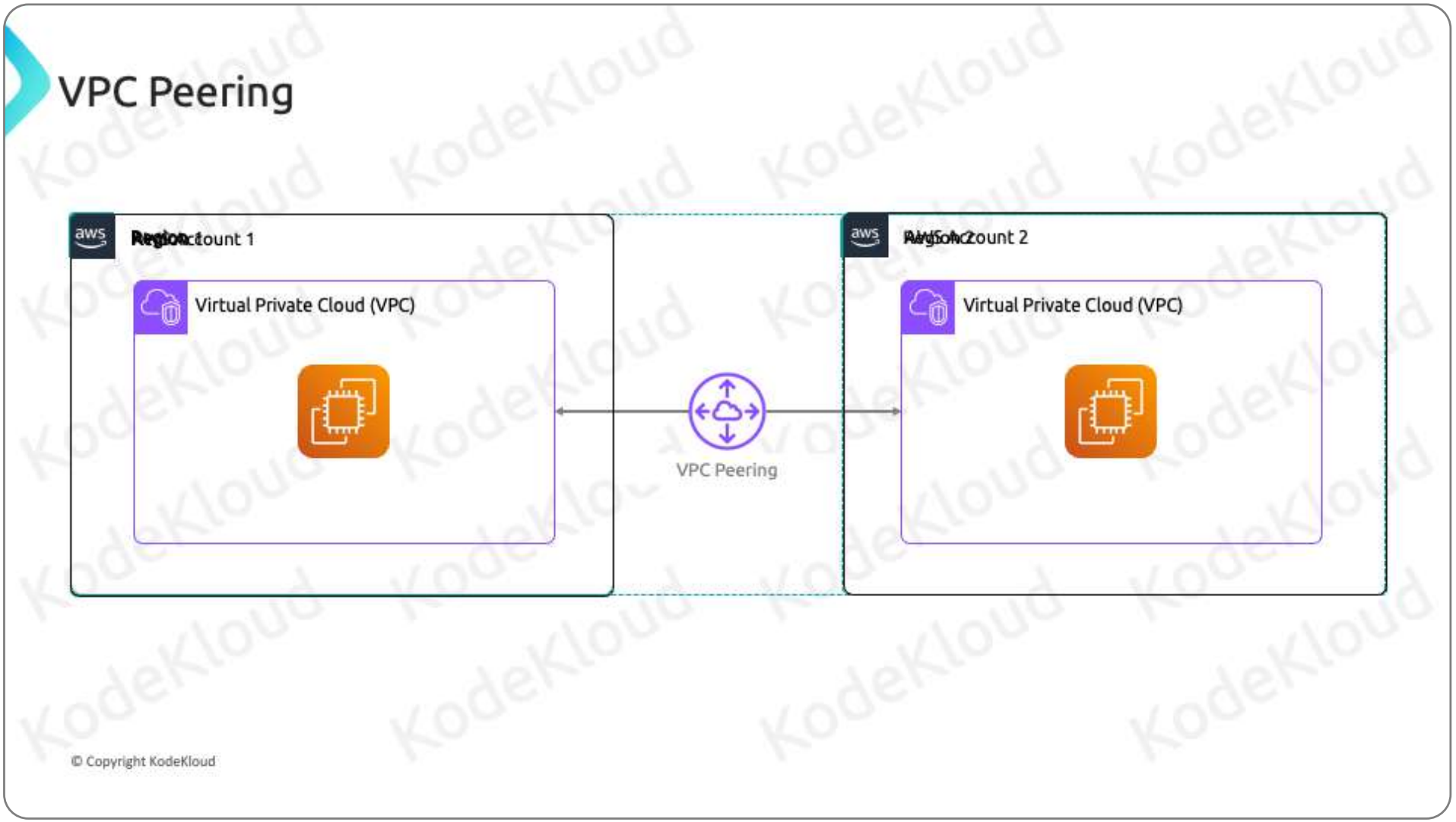
VPC Peering

What we learned in Cloud Practitioner



By default, resources in one VPC can talk to resources in other VPCs

What Is VPC Peering?

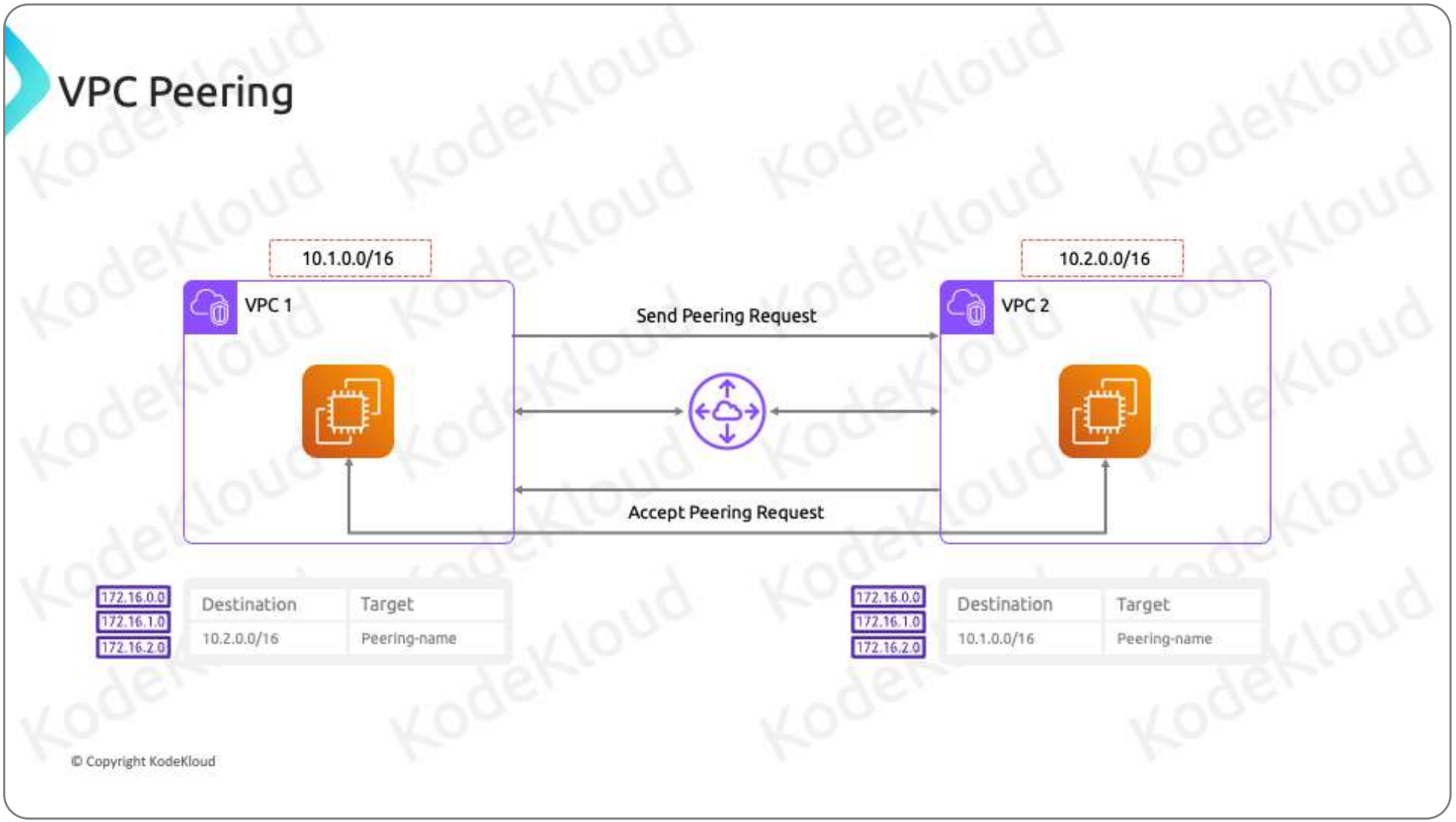


Network connection between two VPCs that routes traffic between them
Instances in either VPC can communicate with each other as if they are within the same network
VPC Peering connections can take place between:

- VPCs in the same regions
- VPCs in different regions
- VPCs in different AWS accounts

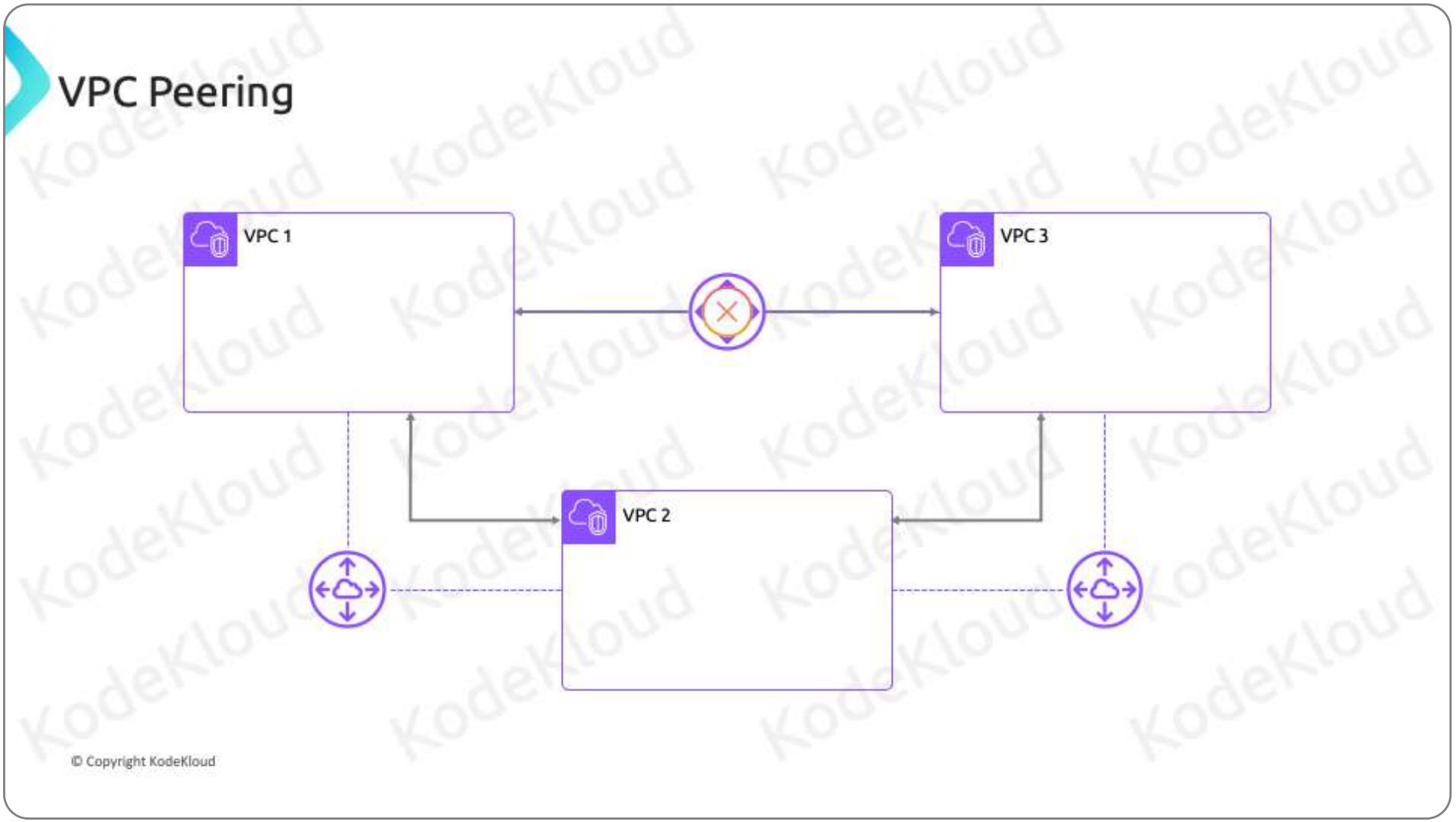
VPC Peering Pricing

How does it work?



The owner of the requester vpc sends a request to the owner of the acceptor VPC to create the VPC Peering connection.

Obviously if both VPCs are owned by you then you sent the request yourself and also accept it. However if oen of the VPCs is on another AWS account then you would be sending a request to the person that owns the other VPC



Let’s say that we have 3 vpcs VP1,2,3

Now we setup a peering between VPC1 and 2. and we setup a peering between VPC 2 and 3

You would think that would mean vpc 1 and 3 could communicate through vpc 2 however that is not the case

Transtive VPCs are not supported

So VPC 1 and 3 would need to have a peering as well

So every VPC that needs to talk to another vpc would need to have a direct peering.

Now there are other features like transtit gateway that can help you accomplish this but we'll talk about that later in the coruse

Summary

01 Network connection between two VPCs that routes traffic between them

02 VPC Peering connects VPCs in the same/different regions and AWS accounts

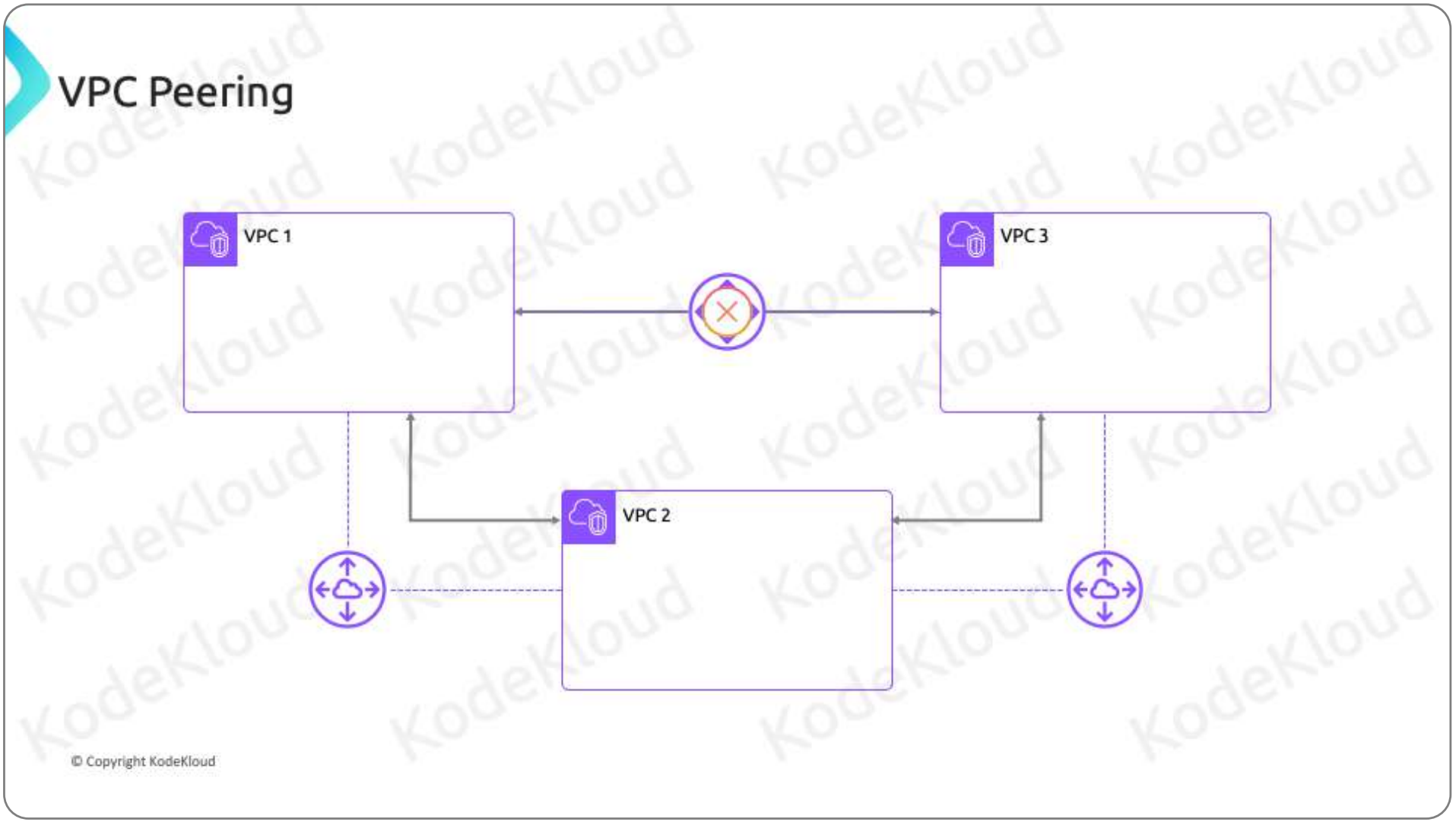
03 No charge for creating a VPC Peering; data transfer that crosses Availability Zones is chargeable

04 VPC peering is not transitive

Service Section

The Power of Transit Networks

Transit Gateway



Let’s say that we have 3 vpcs VP1,2,3

Now we setup a peering between VPC1 and 2. and we setup a peering between VPC 2 and 3

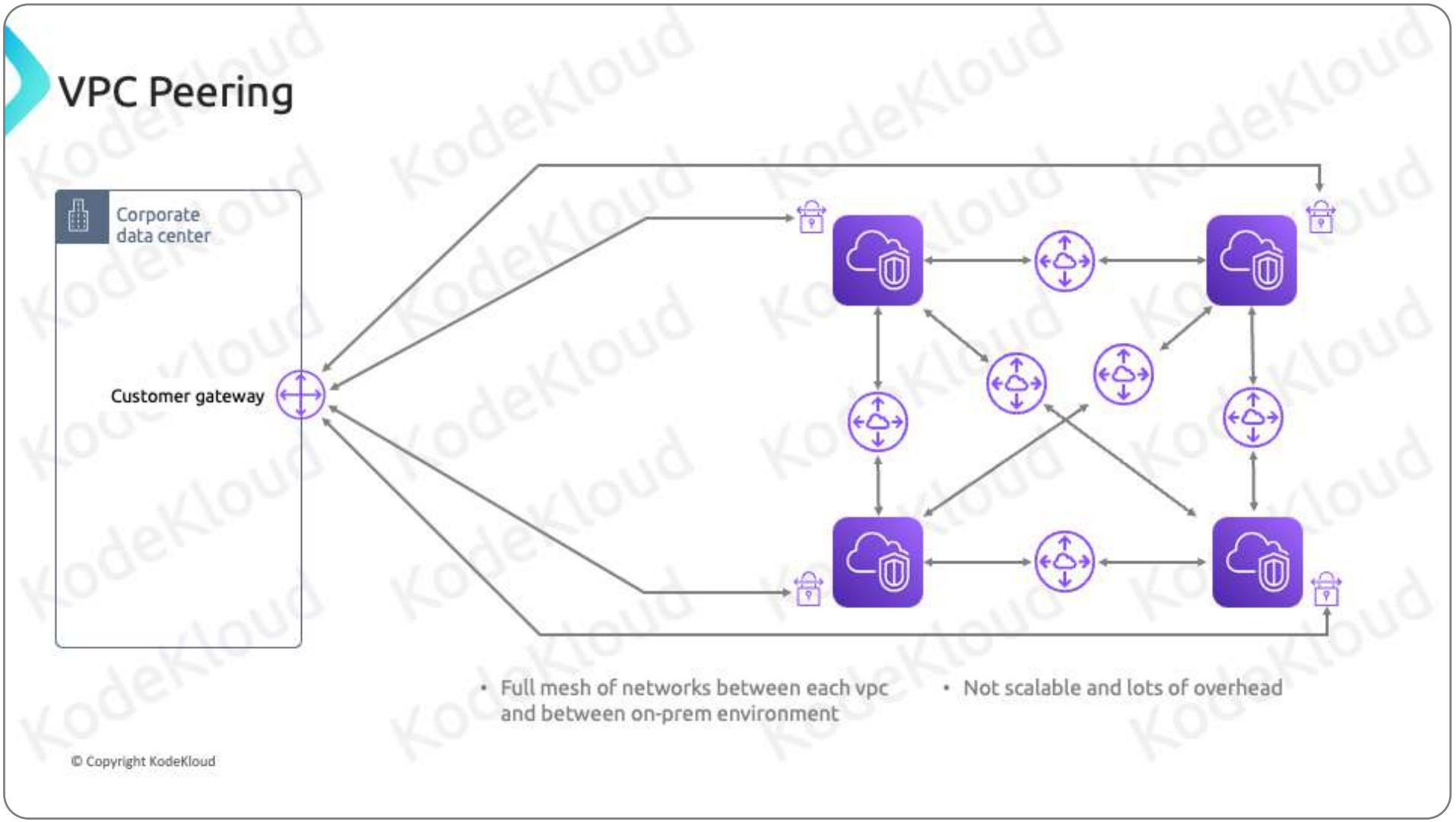
You would think that would mean vpc 1 and 3 could communicate through vpc 2 however that is not the case

Transtive VPCs are not supported

So VPC 1 and 3 would need to have a peering as well

So every VPC that needs to talk to another vpc would need to have a direct peering.

Now there are other features like transtit gateway that can help you accomplish this but we'll talk about that later in the coruse

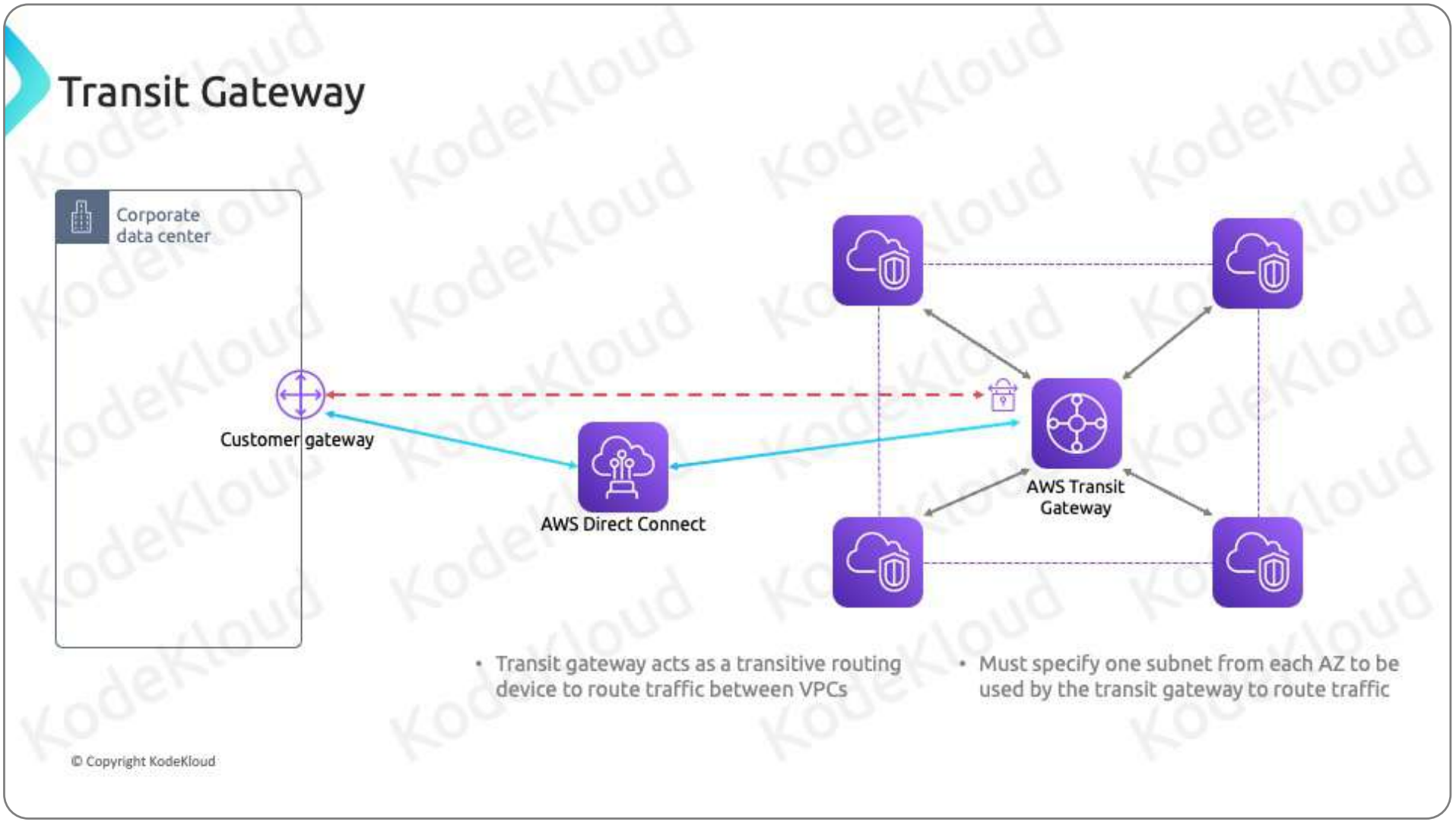


So this means that if we have 4 vpcs they need to be in a full mesh of vpc peerings. This would result in 6 total vpc peerings. [design team] animate in the vpc peerings.

In addition if we have an onprem environment and we need to have the connected to each of the VPCs then we would need a vpn connection between the Customer Gateway and each of the VPCs

[design team] please animate a connection between the customer gateway router and the little lock icon of each VPC

[design team] animate text in



Transit gateways help address this issue.

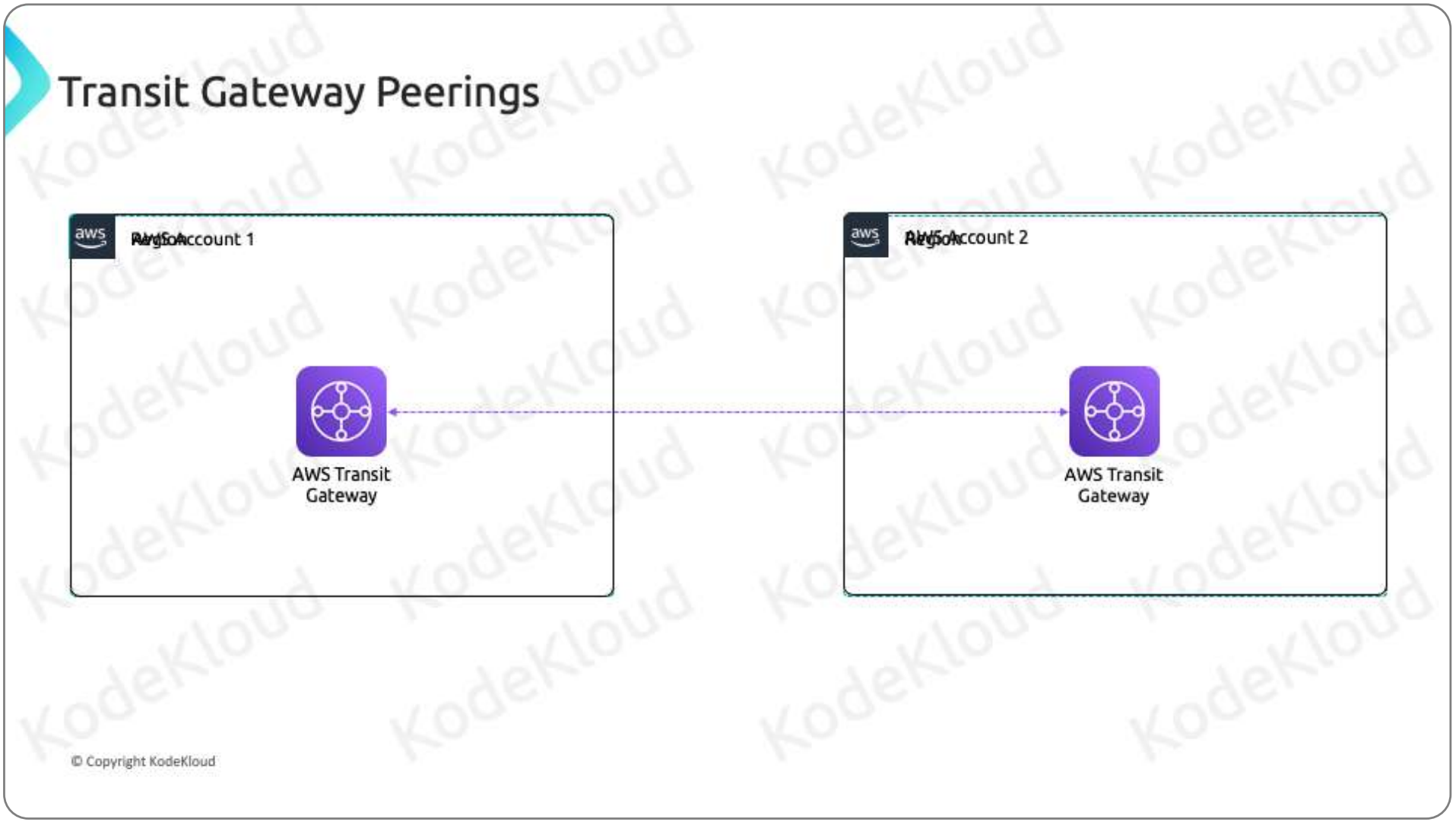
Transit gateway acts as a router to route traffic between VPCs. And what makes it great is that it can act as a transtive router. So we no longer need a full mush of peerings. Instead we can configure each vpc to have a connection to the transit gateway and the TG handles the rest.

[design team] animate in the transit gateway and the connections to each vpc. If you want maybe add an animation showing that all VPCs can talk to each other through the transit gateway

In addition we can also terminate the VPN tunnels form the Customer gateway to the transit gateway. So we don’t need a vpn connection to each and every vpc. Thus simplifying the solution

[design team] animate in the connection from Customer gateway to the lock icon

You can also connect a transit gateway to a direct connect if you prefer physical connections
[design team] animate in the direct connect connection between customer gateway and transit gateway



Transit Gateways can also peer with other transit gateways in other regions as well as other AWS accounts

Summary

01 Transit Gateway simplify networking between VPCs and On-Premise environments

02 Allow for transitive routing

03 Must specify one subnet from each AZ to be used by the transit gateway to route traffic

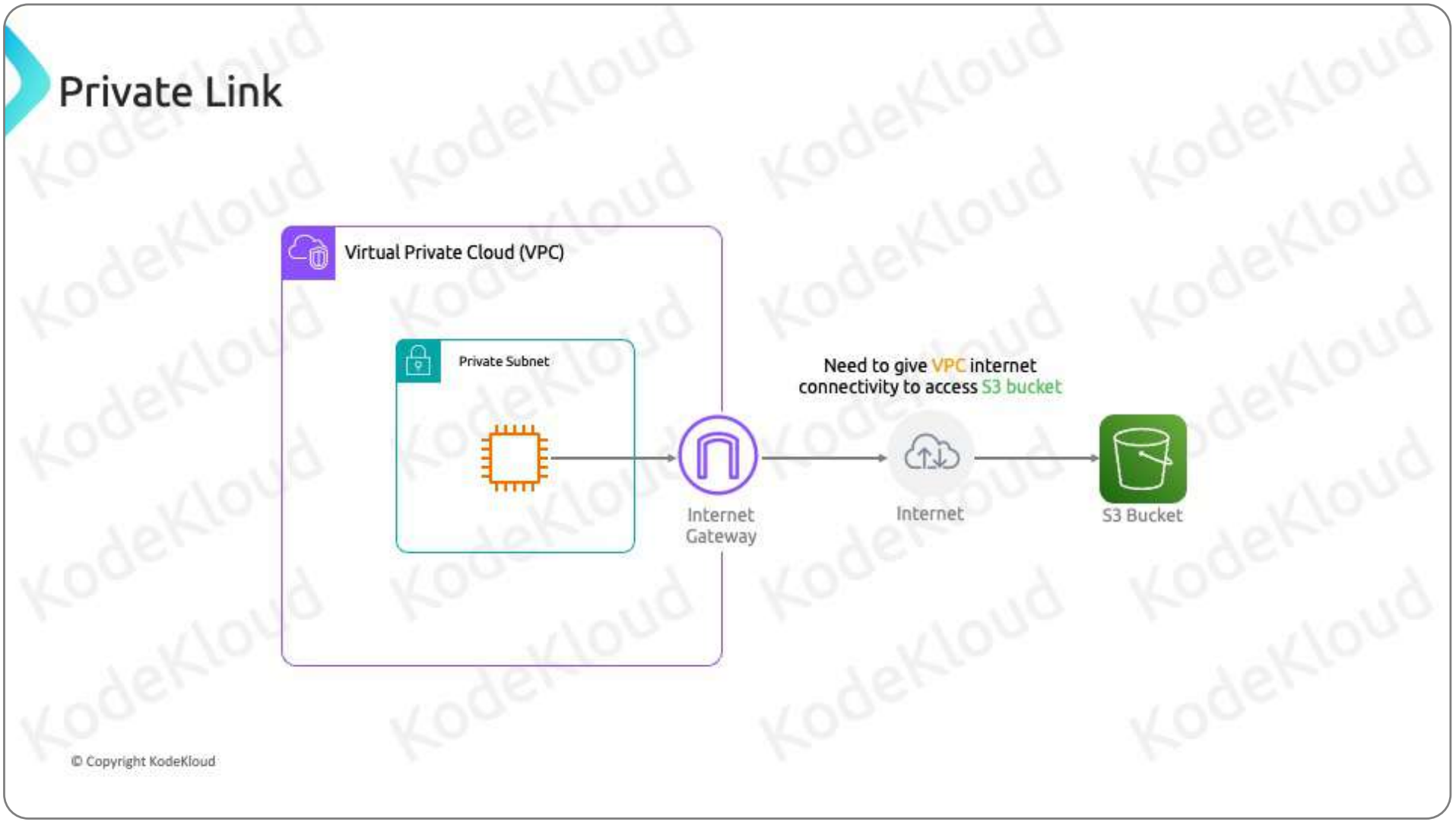
04 Can peer with other Transit Gateways in different regions or AWS accounts

Service Section

The Power of Transit Networking

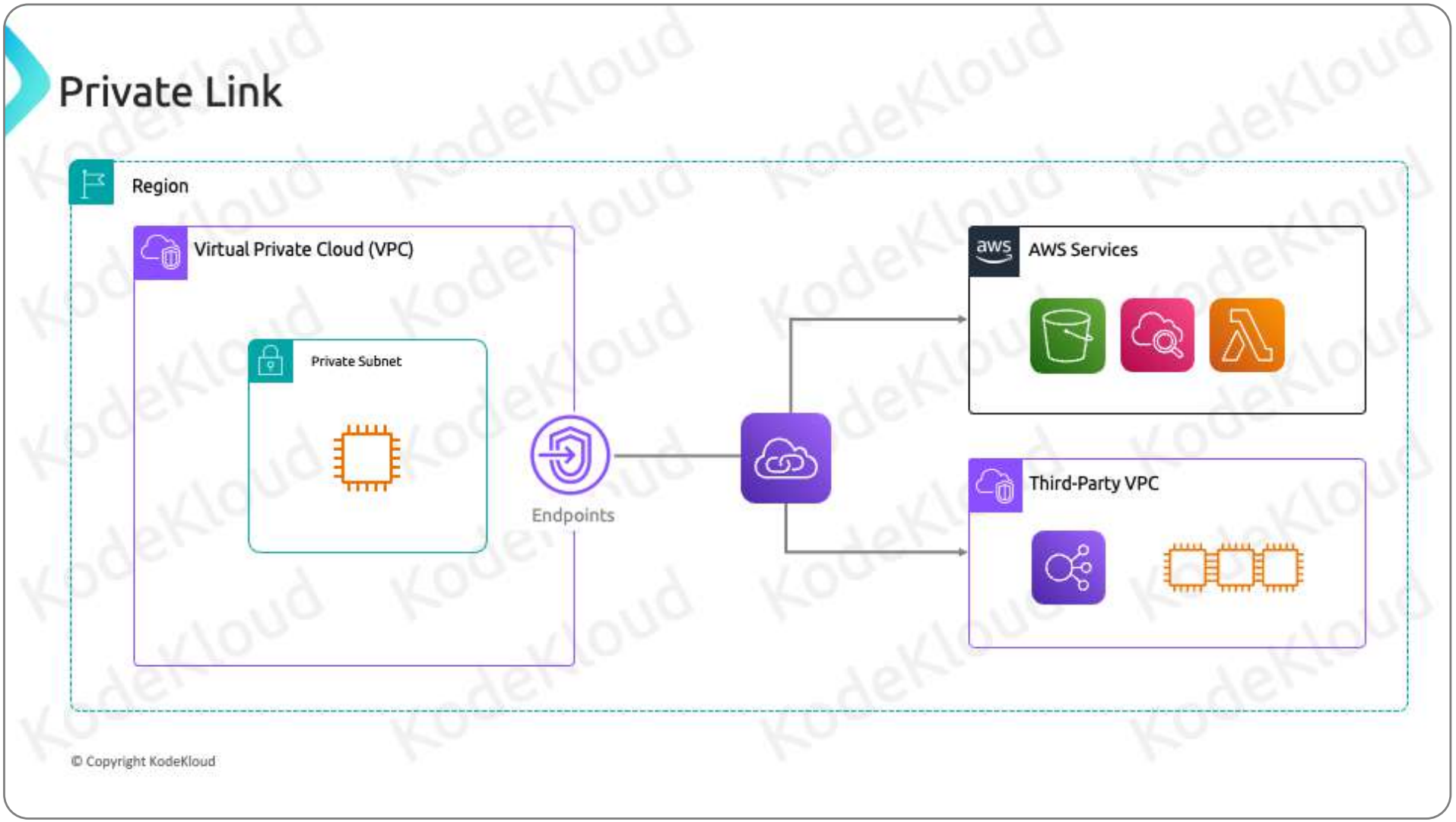
Private Link

What Is Private Link?



Let’s say that we have an EC2 instance deployed in a private subnet. If this ec2 instance needs to access an S3 bucket then it will need access to the internet. So we either need to create a Internet gateway or nat gateway to give it access so it can access the URL of the S3 bucket.

This creates a bunch of problems because now we have to secure our VPC since it has internet connectivity to avoid any ddos attacs or any other security attacks.



Private Links were created to solve this issue as it allows you route traffic between VPCs and services over AWS network. So we can give our VPC direct access to public services like S3 so traffic doesn't have to go through the internet.

Alternatively, use AWS PrivateLink to allow the resources in your VPC to connect to services in other VPCs using private IP addresses, as if those services were hosted directly in your VPC

So if 3rd party company was providing a service and you needed access to it you could create a private link to the vPC that service runs in to give your direct access as if it were hosted directly in your VPC

Summary

01 Allows resources in our VPC to connect to services as if they were in the same VPC

02 Used to connect to public AWS services (S3, CloudWatch) or to other VPCs in AWS

03 VPC endpoints facilitate communication between VPC instances and services

Service Section

The Power of Edge Networking

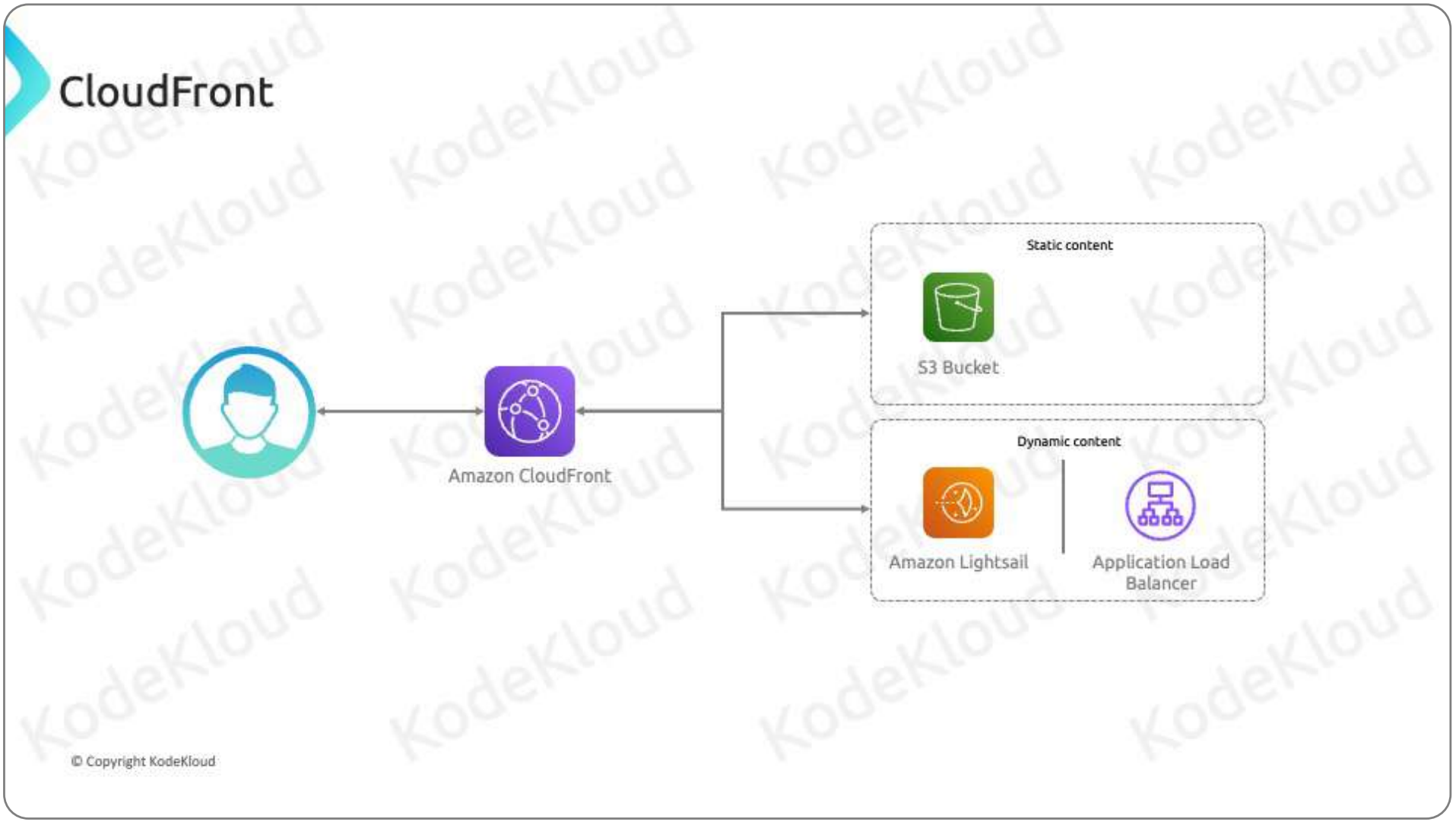
Amazon CloudFront

What we learned in Cloud Practitioner

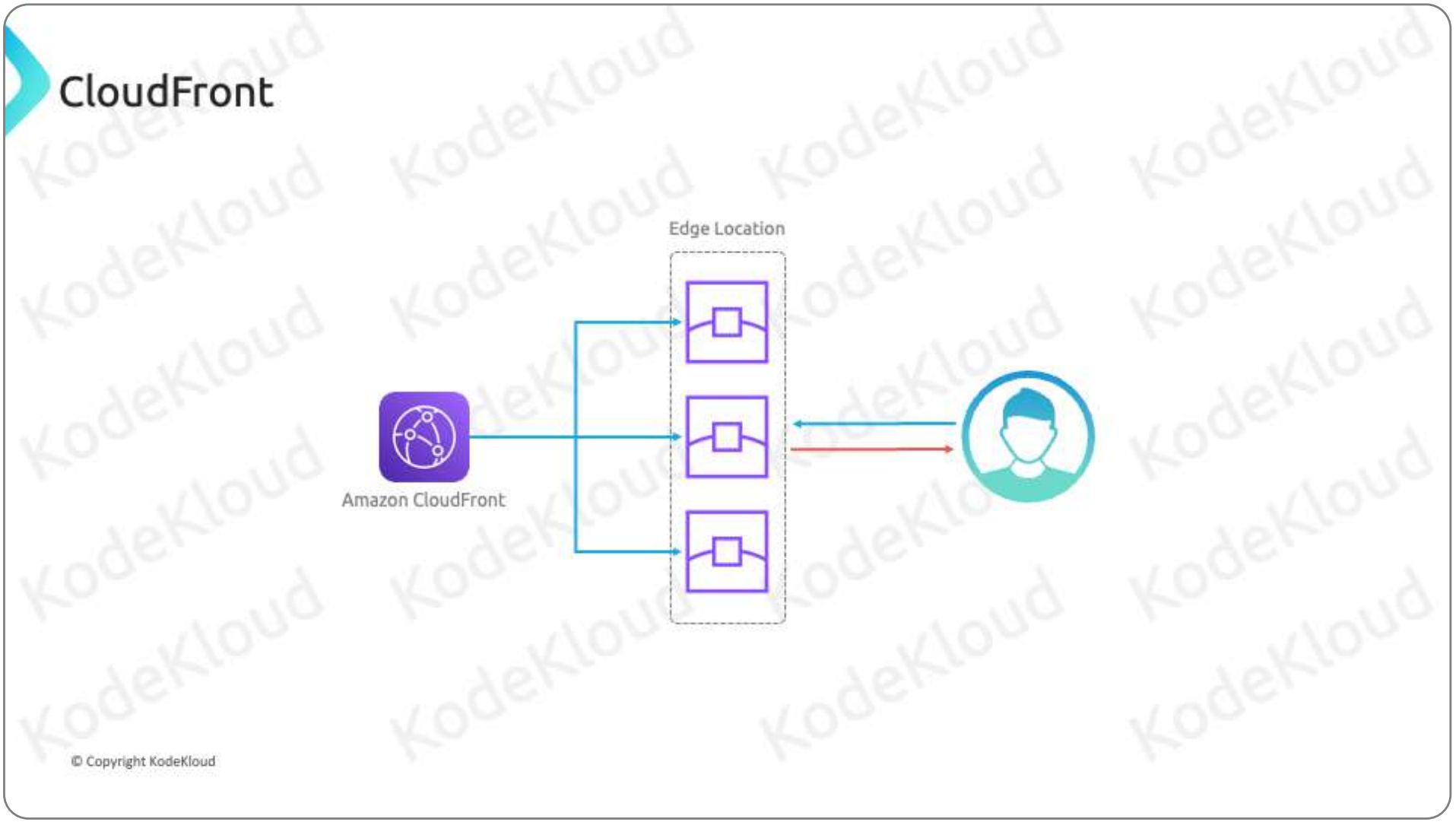
First let's have a quick reminder remember this is all available in the original a WS cloud partitioner course available on code cloud so just making sure that we're clear this is a

Edge Locations are much smaller regions that perform CDN services. Since they are smaller, there are more of them across the globe, allowing your resources to be closer to customers

What Is CloudFront?

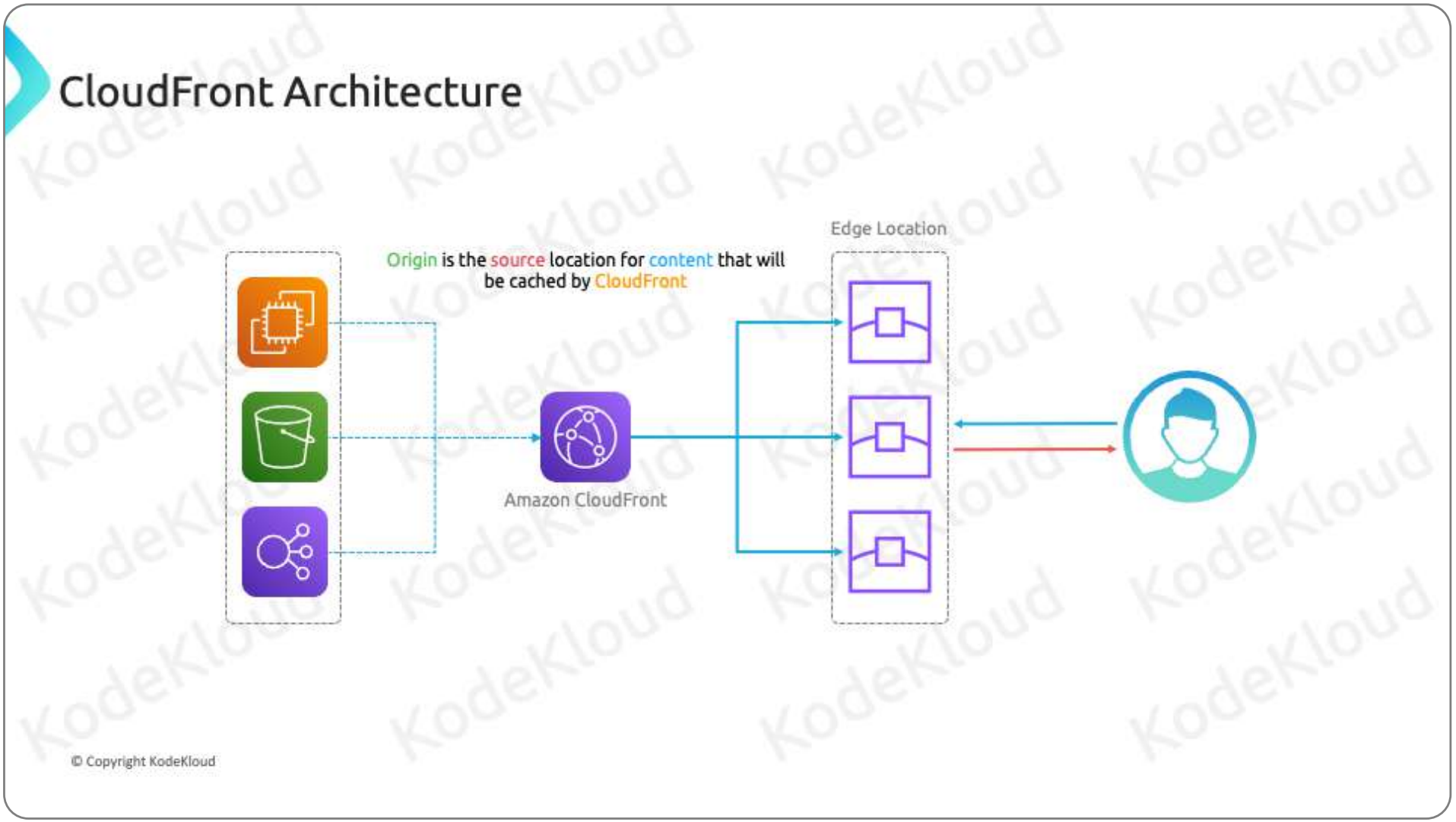


Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users.



CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the request is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.

CloudFront – Basic Components and Features



So let's talk about cloudfront architecture now. With cloudfront we first have to specify the origin.

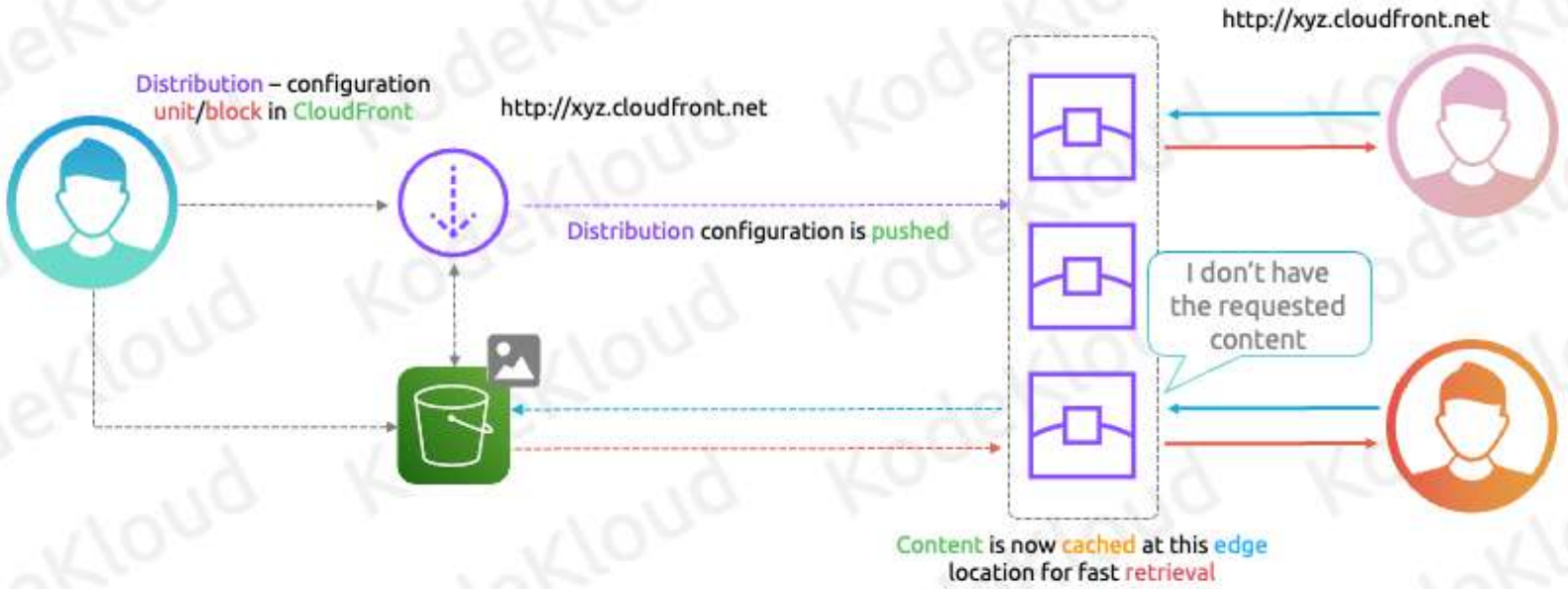
The origin is the sourc elocation of your content that will eventually get cached on the cloudfront edge location. So you are telling cloudfront where you can find the content.

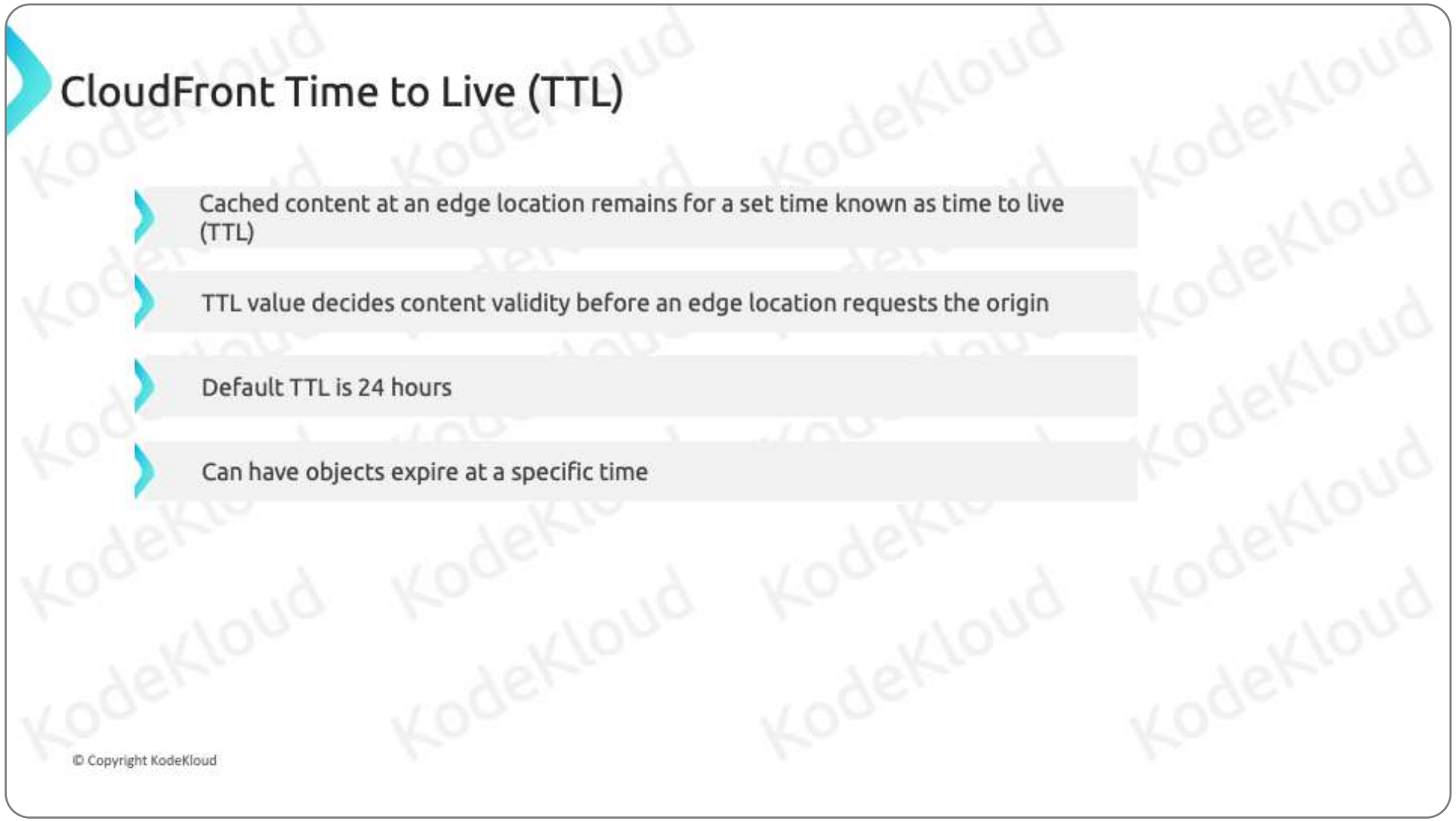
The origin can come from a variety of places like an S3 bucket or a custom origin(Elastic load balancer, http server running

on EC2.

Cloudfront will then push the content from the origin to its edge locations and users can get the content form the edge location instead of going all the way to the origin

CloudFront Architecture





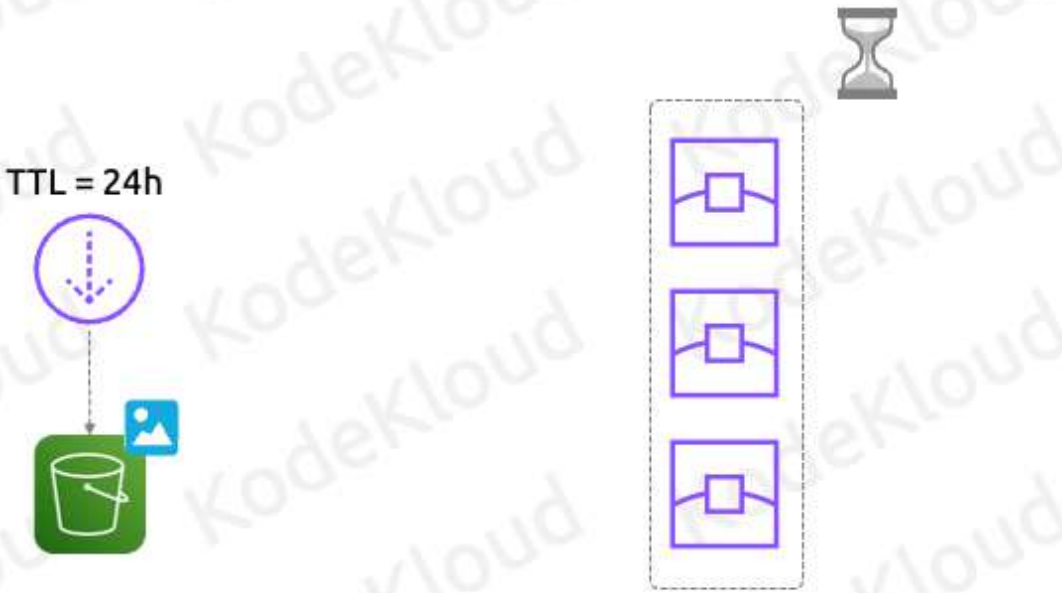
Content cached at an edge location remains cached for a specified period of time – TTL
The TTL value is used to determine how long content should remain valid before an edge location has to forward a request to the origin
Default TTL is 24 hours
- Can be customized per distribution
- Can also set per object TTL values

Origin Header: Cache-Control max-age (seconds)
Origin Header: Cache-Control s-maxage (seconds)

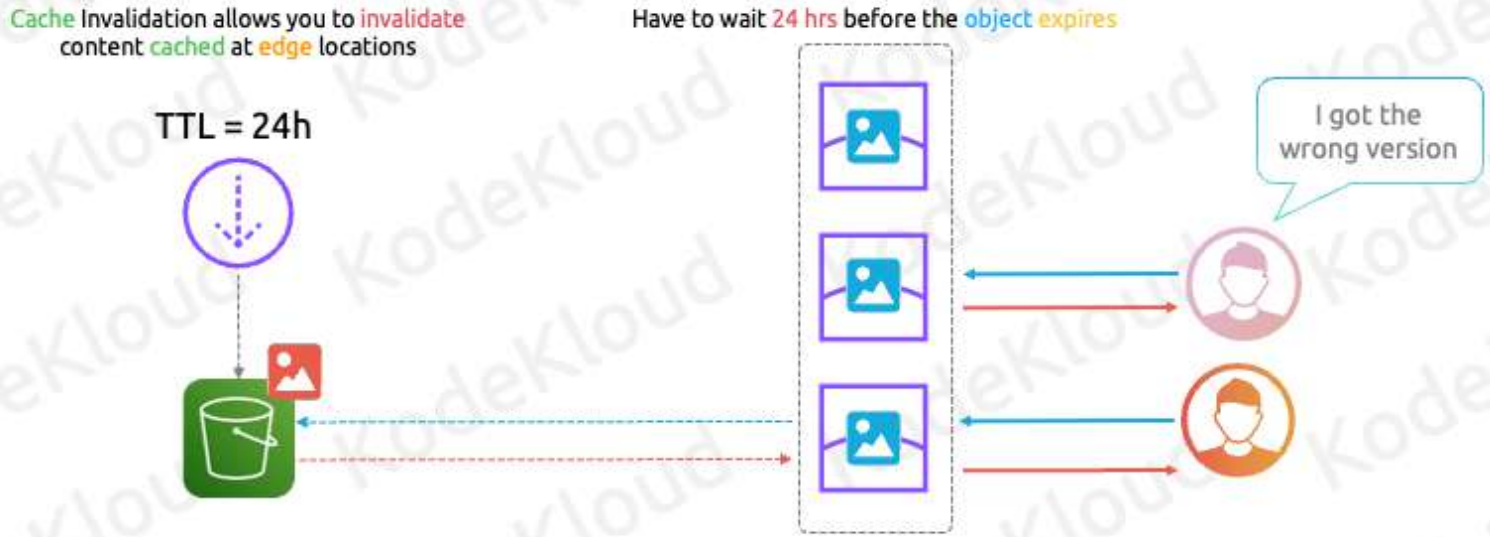
Can have objects expire at a specific time

Origin Header: Expires (Date/Time)

Time to Live (TTL)



Invalidation



Cache Invalidations

- Invalidations are performed on a distribution
- You can invalidate all objects in a distribution, a specific folder, or a specific object
- /* – Entire distribution
- /file.txt – Individual file
- /images/* – All objects in images directory

CloudFront – Basic Integrations With Other Services

SSL/TLS and ACM

CloudWatch

Automatically publishes operational metrics for distributions

Can enable extra metrics for an additional cost

CloudFront – Basic Use Cases

Use Cases

01

Static websites

02

Video on demand

03

Streaming

Summary

01 Delivers content via a worldwide network of data centers called edge locations

02 Helps get content close to customers

03 Origin is the content source for CloudFront edge locations

04 Distribution is a unit of configuration for CloudFront

Summary

01 CloudFront distributions get a default domain name- http://xyz.cloudfront.net

02 TTL value decides content validity before an edge location requests the origin

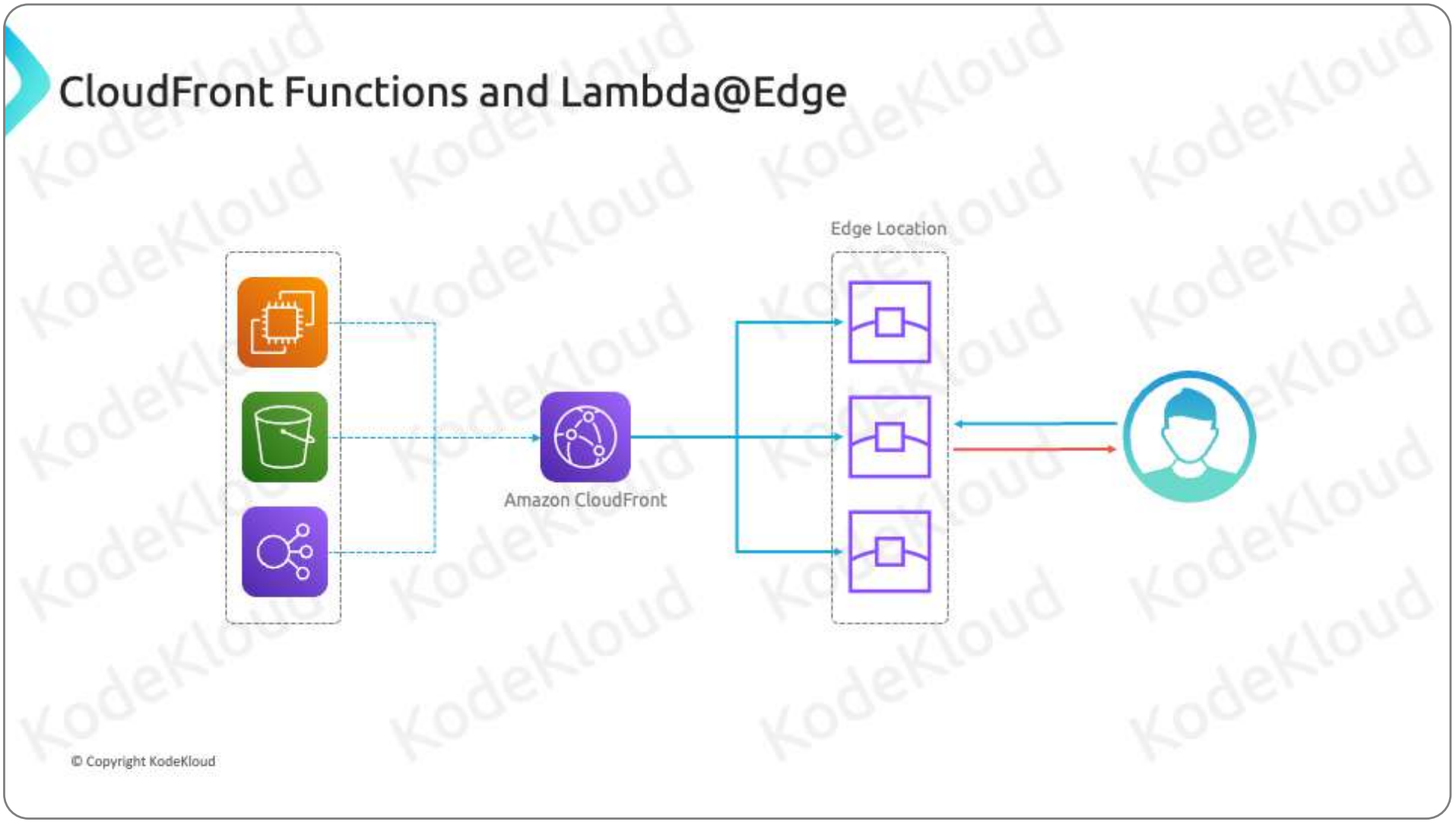
03 Default TTL is 24 hours

04 Cache Invalidation – Invalidates objects at edge locations before their TTL expires

Service Section

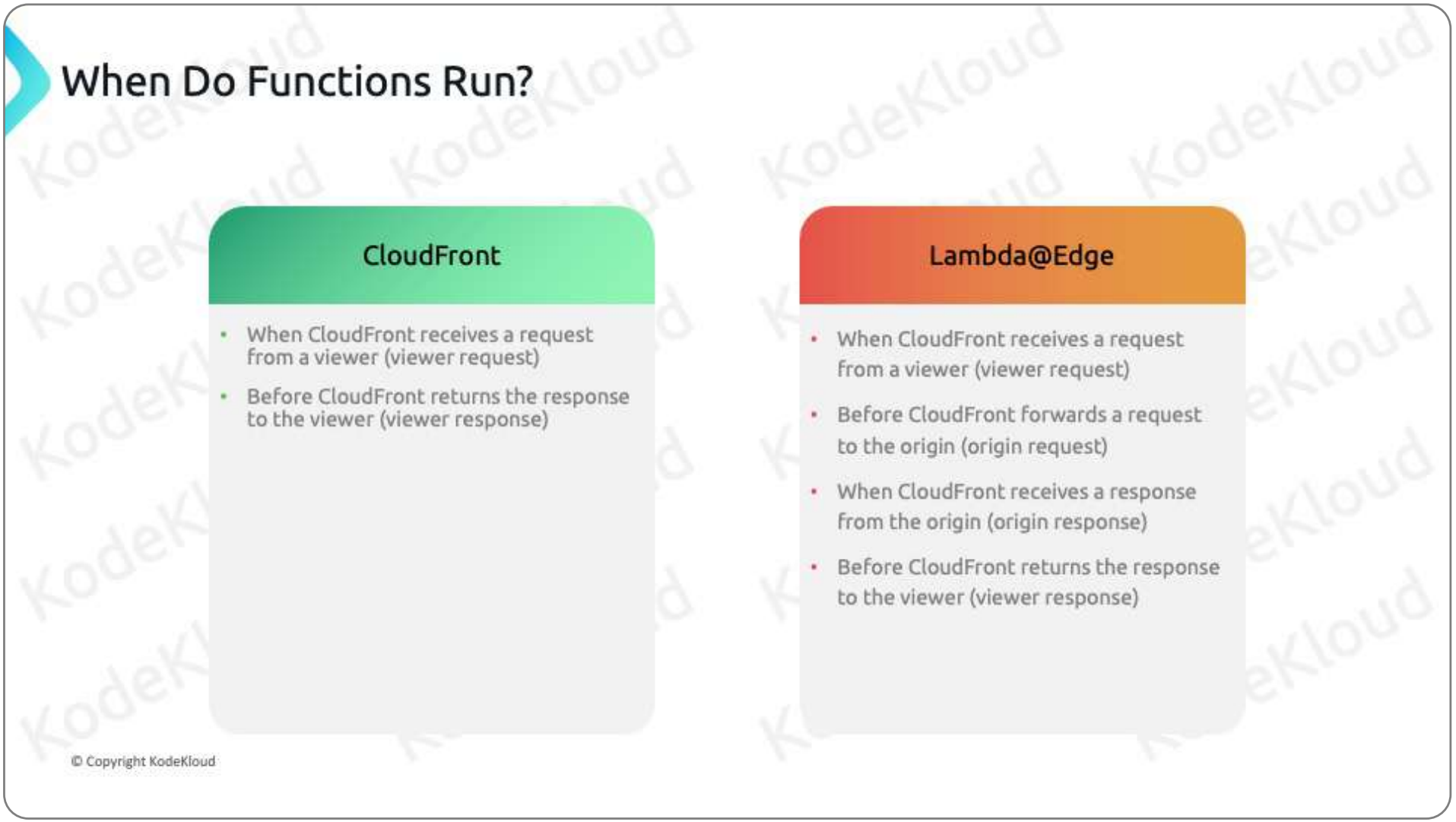
The Power of Edge Networking

CloudFront Functions and Lambda@Edge



Edge locations are nothing more that caches to store/and cache data. With cloudfront functions and lambda@edge we can now write lightweight functions at these edge locations.

These functions can perform a variety of different operations like manipulating requests and responses that flow through cloudfront, perform basic authentication/authorization as well as generate responses at the ddge, so that we don't' need to has a server.



Cloudfront:

- When CloudFront receives a request from a viewer (viewer request)
- Before CloudFront returns the response to the viewer (viewer response)

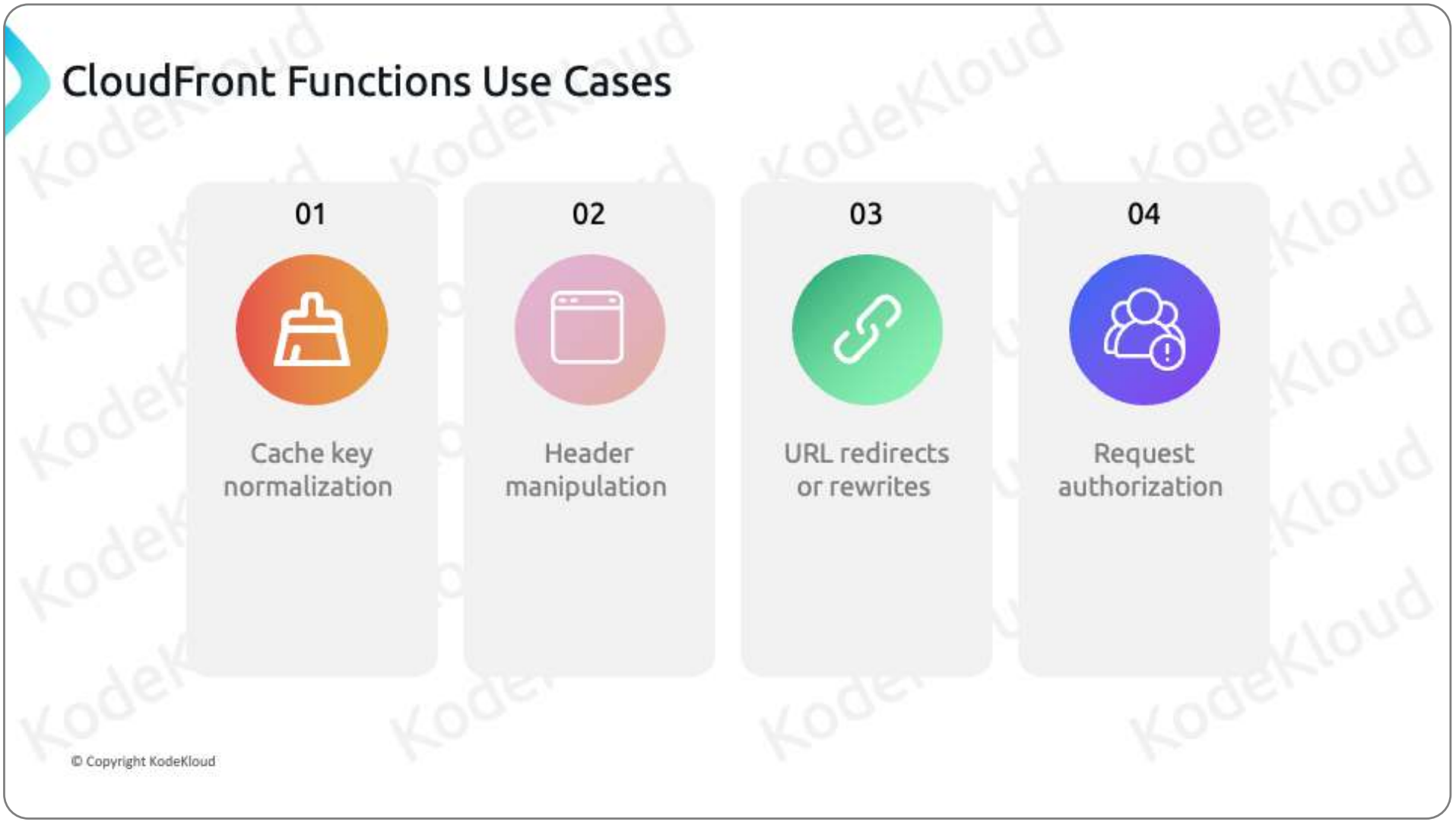
Lambda@Edge

When CloudFront receives a request from a viewer (viewer request)

Before CloudFront forwards a request to the origin (origin request)

When CloudFront receives a response from the origin (origin response)

Before CloudFront returns the response to the viewer (viewer response)

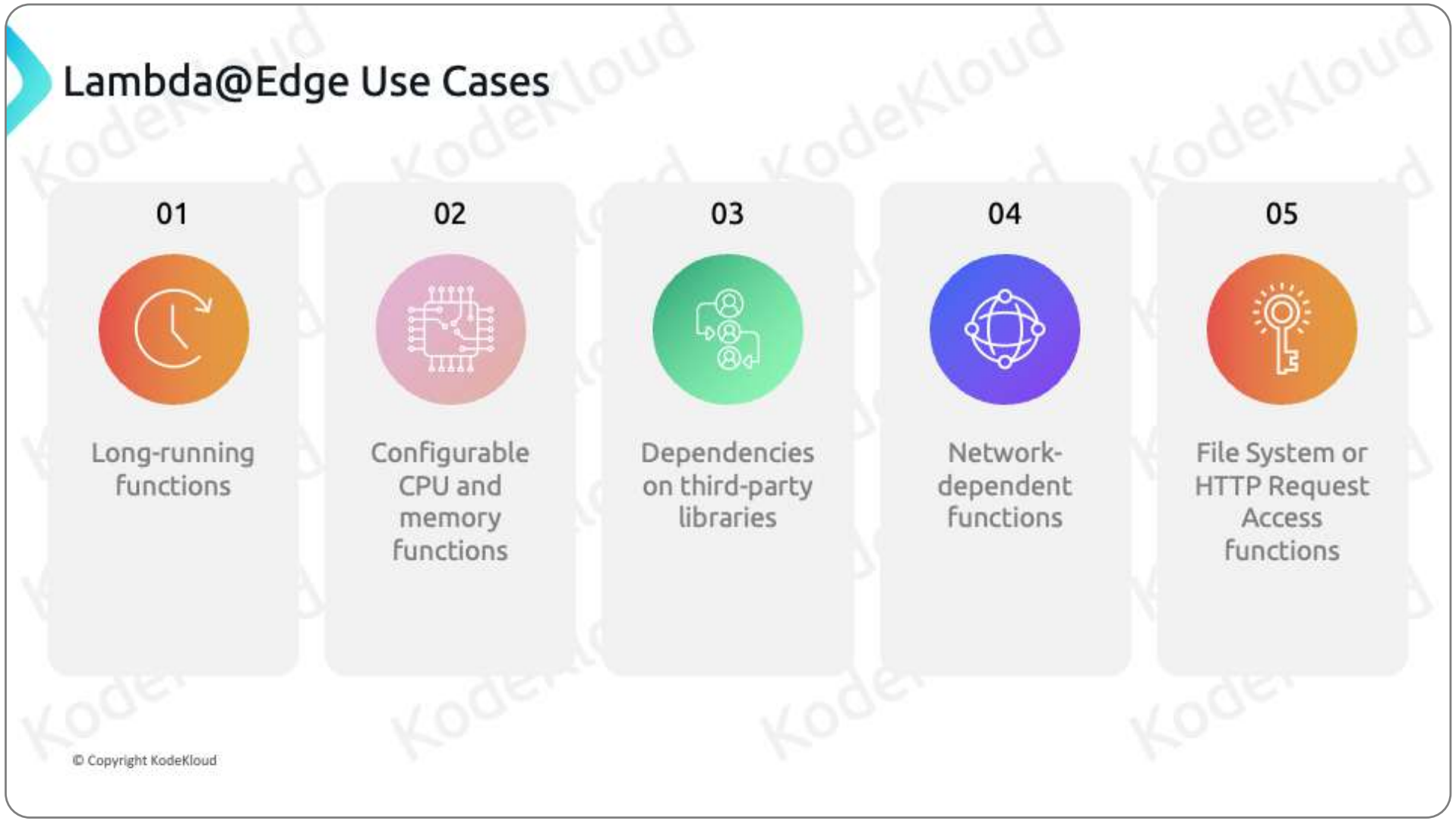


CloudFront Functions is ideal for lightweight, short-running functions for use cases like the following:

- **Cache key normalization** – You can transform HTTP request attributes (headers, query strings, cookies, and even the URL path) to create an optimal cache key, which can improve your cache hit ratio.
- **Header manipulation** – You can insert, modify, or delete HTTP headers in the request or response. For example, you can add a True-Client-IP header to every request.
- **URL redirects or rewrites** – You can redirect viewers to other pages based on information in the request, or rewrite all requests from

one path to another.

- **Request authorization** – You can validate hashed authorization tokens, such as JSON web tokens (JWT), by inspecting authorization headers or other request metadata.



Lambda@Edge is a good fit for the following scenarios:

- Functions that take several milliseconds or more to complete.
- Functions that require adjustable CPU or memory.
- Functions that depend on third-party libraries (including the AWS SDK, for integration with other AWS services).
- Functions that require network access to use external services for processing.
- Functions that require file system access or access to the body of HTTP requests.

CloudFront Functions and Lambda@Edge

	CloudFront Functions	Lambda@Edge
Programming Languages	JavaScript (ECMAScript 5.1 complaint)	Node.js and Python
Event Sources	• Viewer request • Viewer response	• Viewer request • Viewer response • Origin request • Origin response
Scale	10,000,000 requests per second or more	Up to 10,000 requests per second per Region
Function duration	Sub millisecond	Up to 5 seconds (Viewer request and viewer response) Up to 30 seconds (Origin request and origin response)
Maximum memory	2 MB	128-3,008 MB
Maximum size of the function code and included libraries	10 KB	1 MB (Viewer request and viewer response) 50 MB (Origin request and origin response)
Network access	No	Yes
File system access	No	Yes
Access to the request body	No	Yes
Access to geolocation and device data	Yes	No (viewer request) Yes (origin request, origin response, and viewer response)
Can build and test entirely within CloudFront	Yes	No
Function Logging and metrics	Yes	Yes
Pricing	Free tier available; charged per request	No free tier; charged per request and function duration

Summary

01 Allow you to run code at edge locations as close to users as possible

02 Can be used to manipulate requests/responses that flow through CloudFront

03 CloudFront functions are great for header manipulations, URL redirects, authorization

04 Lambda@Edge: Long runtime, Network/File access

Service Section

The Power of Edge Networking

Global Accelerator

Let’s say that we have an application deployed in a region in NA. When a user in NA sends a request to your application, the request will traverse through the internet through their ISP and arrive at your application which is a relatively short distance away from the user.

Now when usuers are all the away across the globe, their requests once again have to traverse over the internet all the way across the globe to arrive at your application.

A lot of times with the internet, traffic haas to traverse through multiple routers, and sometimes traffic may not take the most optimal path as well as there could be a drop in network packets resulting in a degraradation in performance for users that are far away from the NA region.

This is where global accelerator was created to address this issue. With global accelerator the main idea is to get users as quickly onto the AWS global network by routing them to the closest edge location so that they can most optimally reach your application regardless of what region its located in.

Traffic travels over the well-monitored, congestion-free, redundant AWS global network to the endpoint. By maximizing the time that traffic is on the AWS network, Global Accelerator ensures that traffic is always routed over the optimum network path.

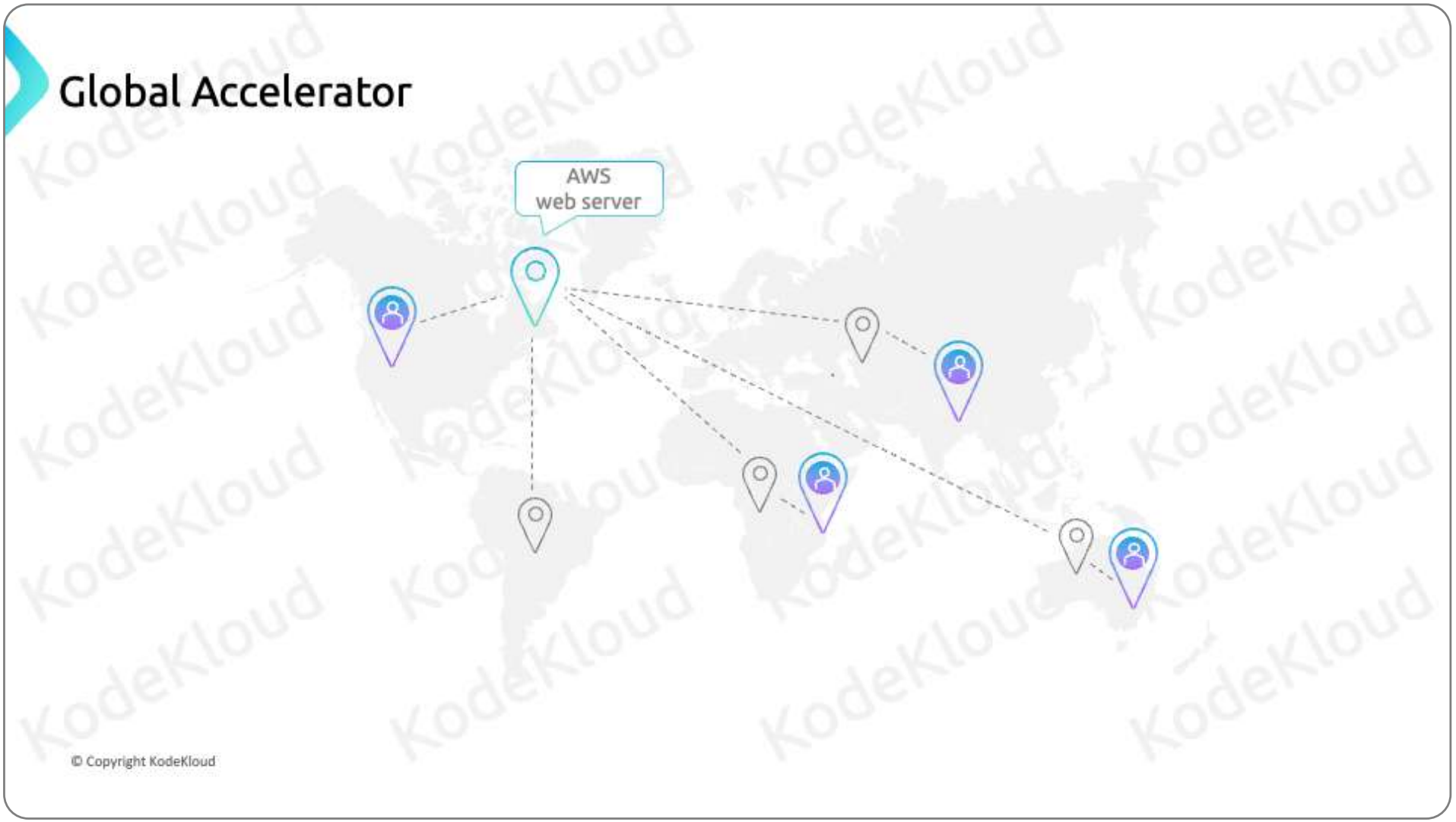
The way that this works is that Global accelerator will provide you an IP address. This is the address users will use to reach your application. So when navigates to this IP it will route to the closest AWS edge location at which point it will get forwarded to your pplication over the AWS backbone

[design team] I've included slides from the cloudfront lecture that is kind of similar to give you a starting point. Please check the two slides below this one.

The main idea is for users that are far away from the app, When they don't use the feature then they have to go over the internet and show that they may take a less than optimal path in a congested network.

When they use the feature, they send a request to the edge location, and then from the edge location it

traverses AWS' special backbone network to make the request as quick and efficient as possible.



Let’s say that we have an application deployed in a region in NA. When a user in NA sends a request to your application, the request will traverse through the internet through their ISP and arrive at your application which is a relatively short distance away from the user.

Now when usuers are all the away across the globe, their requests once again have to traverse over the internet all the way across the globe to arrive at your application.

A lot of times with the internet, traffic haas to traverse through multiple routers, and sometimes traffic may not take the most optimal path as well as there could be a drop in network packets resulting in a degraradation in performance for users that are far away from the NA region.

This is where global accelerator was created to address this issue. With global accelerator the main idea is to get users as quickly onto the AWS global network by routing them to the closest edge location so that they can most optimally reach your application regardless of what region its located in.

Traffic travels over the well-monitored, congestion-free, redundant AWS global network to the endpoint. By maximizing the time that traffic is on the AWS network, Global Accelerator ensures that traffic is always routed over the optimum network path.

The way that this works is that Global accelerator will provide you an IP address. This is the address users will use to reach your application. So when navigates to this IP it will route to the closest AWS edge location at which point it will get forwarded to your pplication over the AWS backbone

[design team] I've included slides from the cloudfront lecture that is kind of similar to give you a starting point. Please check the two slides below this one.

The main idea is for users that are far away from the app, When they don't use the feature then they have to go over the internet and show that they may take a less than optimal path in a congested network.

When they use the feature, they send a request to the edge location, and then from the edge location it

traverses AWS' special backbone network to make the request as quick and efficient as possible.

Let’s say that we have an application deployed in a region in NA. When a user in NA sends a request to your application, the request will traverse through the internet through their ISP and arrive at your application which is a relatively short distance away from the user.

Now when usuers are all the away across the globe, their requests once again have to traverse over the internet all the way across the globe to arrive at your application.

A lot of times with the internet, traffic haas to traverse through multiple routers, and sometimes traffic may not take the most optimal path as well as there could be a drop in network packets resulting in a degraradation in performance for users that are far away from the NA region.

This is where global accelerator was created to address this issue. With global accelerator the main idea is to get users as quickly onto the AWS global network by routing them to the closest edge location so that they can most optimally reach your application regardless of what region its located in.

Traffic travels over the well-monitored, congestion-free, redundant AWS global network to the endpoint. By maximizing the time that traffic is on the AWS network, Global Accelerator ensures that traffic is always routed over the optimum network path.

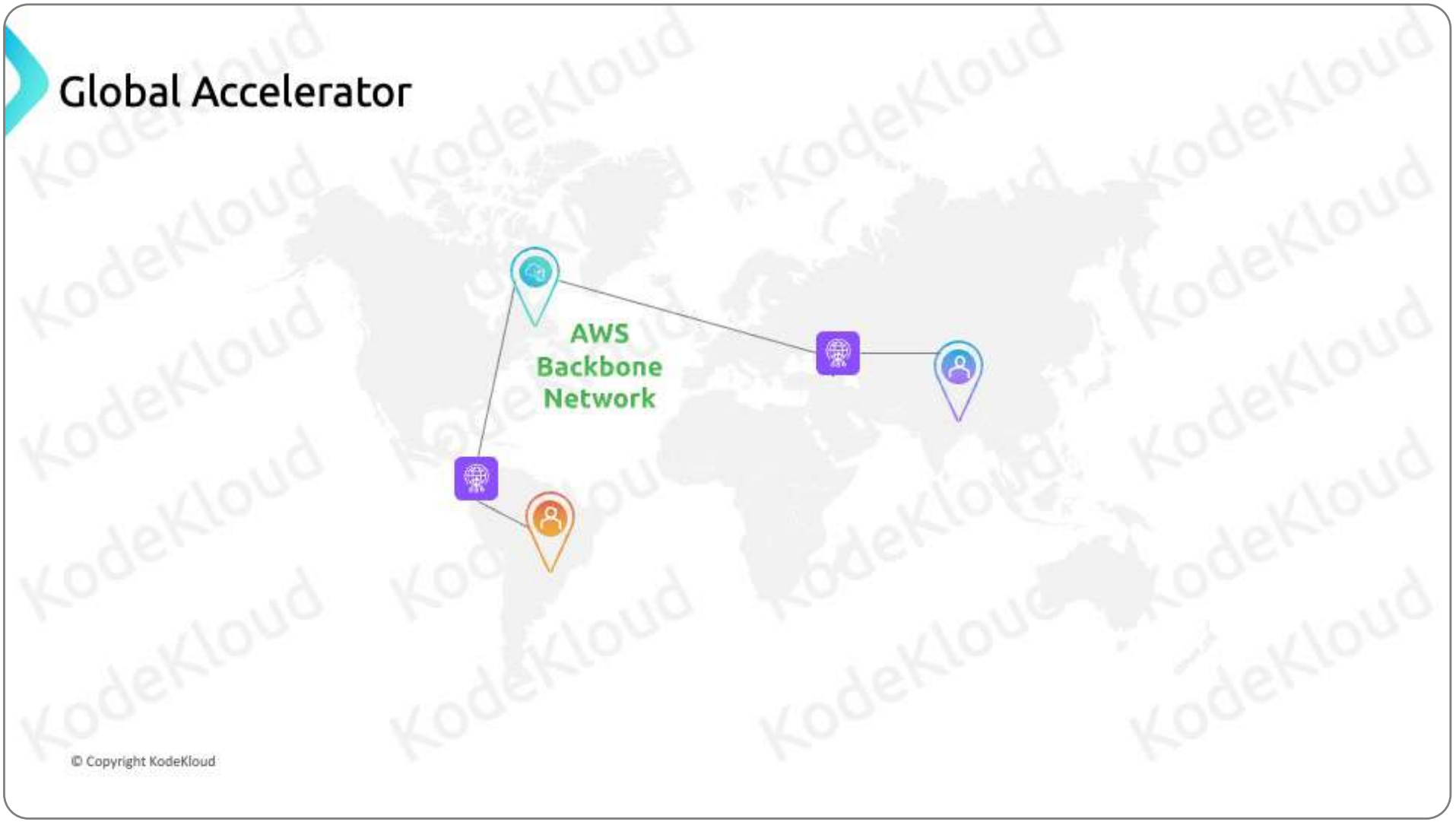
The way that this works is that Global accelerator will provide you an IP address. This is the address users will use to reach your application. So when navigates to this IP it will route to the closest AWS edge location at which point it will get forwarded to your pplication over the AWS backbone

[design team] I've included slides from the cloudfront lecture that is kind of similar to give you a starting point. Please check the two slides below this one.

The main idea is for users that are far away from the app, When they don't use the feature then they have to go over the internet and show that they may take a less than optimal path in a congested network.

When they use the feature, they send a request to the edge location, and then from the edge location it

traverses AWS' special backbone network to make the request as quick and efficient as possible.



Let’s say that we have an application deployed in a region in NA. When a user in NA sends a request to your application, the request will traverse through the internet through their ISP and arrive at your application which is a relatively short distance away from the user.

Now when usuers are all the away across the globe, their requests once again have to traverse over the internet all the way across the globe to arrive at your application.

A lot of times with the internet, traffic haas to traverse through multiple routers, and sometimes traffic may not take the most optimal path as well as there could be a drop in network packets resulting in a degraradation in performance for users that are far away from the NA region.

This is where global accelerator was created to address this issue. With global accelerator the main idea is to get users as quickly onto the AWS global network by routing them to the closest edge location so that they can most optimally reach your application regardless of what region its located in.

Traffic travels over the well-monitored, congestion-free, redundant AWS global network to the endpoint. By maximizing the time that traffic is on the AWS network, Global Accelerator ensures that traffic is always routed over the optimum network path.

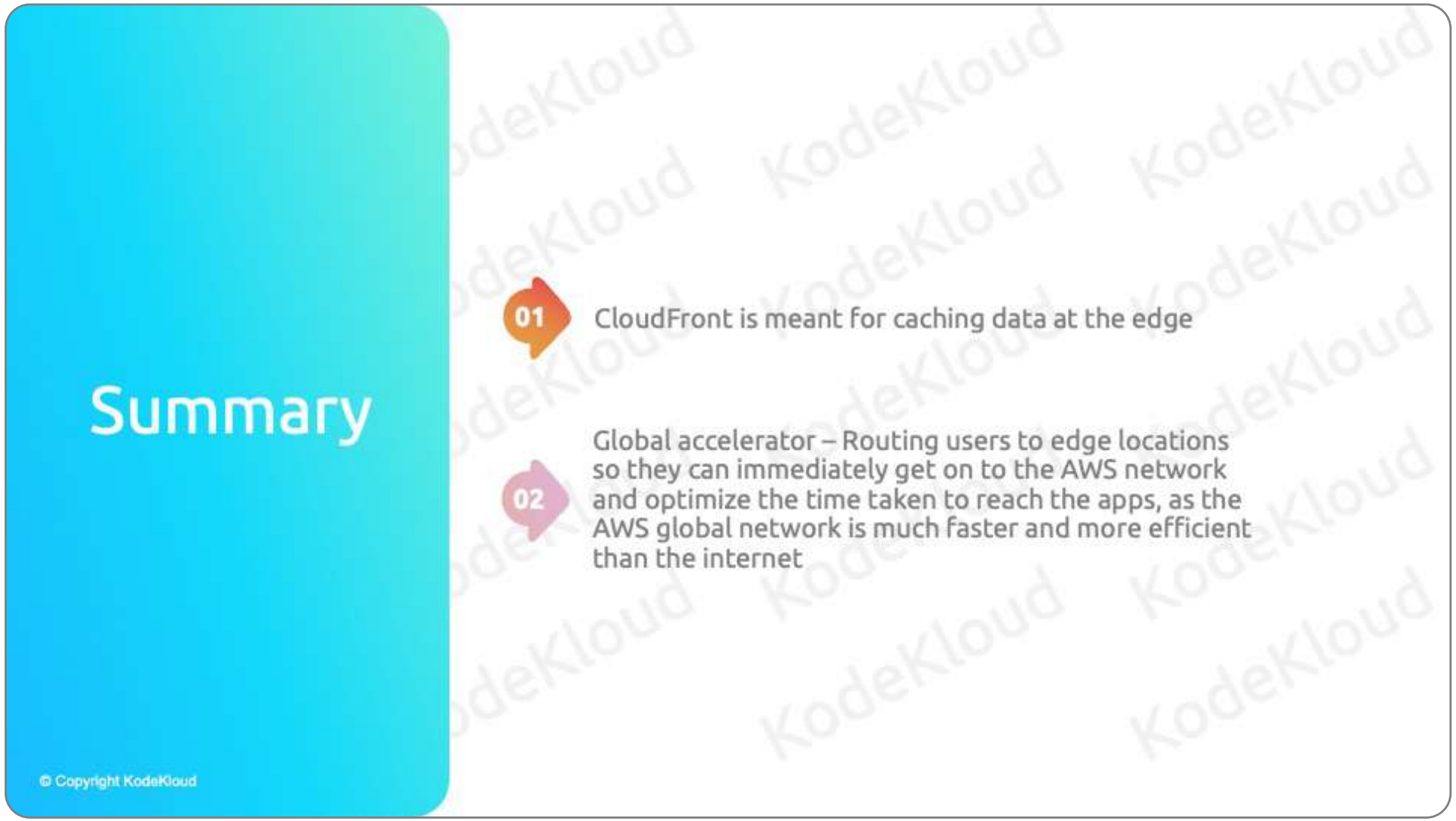
The way that this works is that Global accelerator will provide you an IP address. This is the address users will use to reach your application. So when navigates to this IP it will route to the closest AWS edge location at which point it will get forwarded to your pplication over the AWS backbone

[design team] I've included slides from the cloudfront lecture that is kind of similar to give you a starting point. Please check the two slides below this one.

The main idea is for users that are far away from the app, When they don't use the feature then they have to go over the internet and show that they may take a less than optimal path in a congested network.

When they use the feature, they send a request to the edge location, and then from the edge location it

traverses AWS' special backbone network to make the request as quick and efficient as possible.



The global accelerator sounds a lot like cloudfront, how is it any different.

Cloudfront is all about caching data as close to the end users as possible. So if you have a website, or media files, you can push them out to cloudfront edge locations and users can access the media as close to them as possible. Primarily used for HTTP/HTTPS applications

With the global accelerator the idea is not to cache data close to the end user, it’s all about getting users request(to any type of application) as quickly as possible onto the aws global network so they can traverse the redundant, congestion-free network. We want to maximize the time that traffic on the aws network as the global accelerator ensure traffic is routed over the optimum network path.

So for the exam remember this:
Cloudfront is meant for caching data at the edge
Global accelerator – routing users to edge locations so they can immediately get onto the aws network and optimize the time it takes to reach the apps as the aws global network is much faster and more efficient then going over the internet.

Service Section

The Power of Core Networking

Route 53

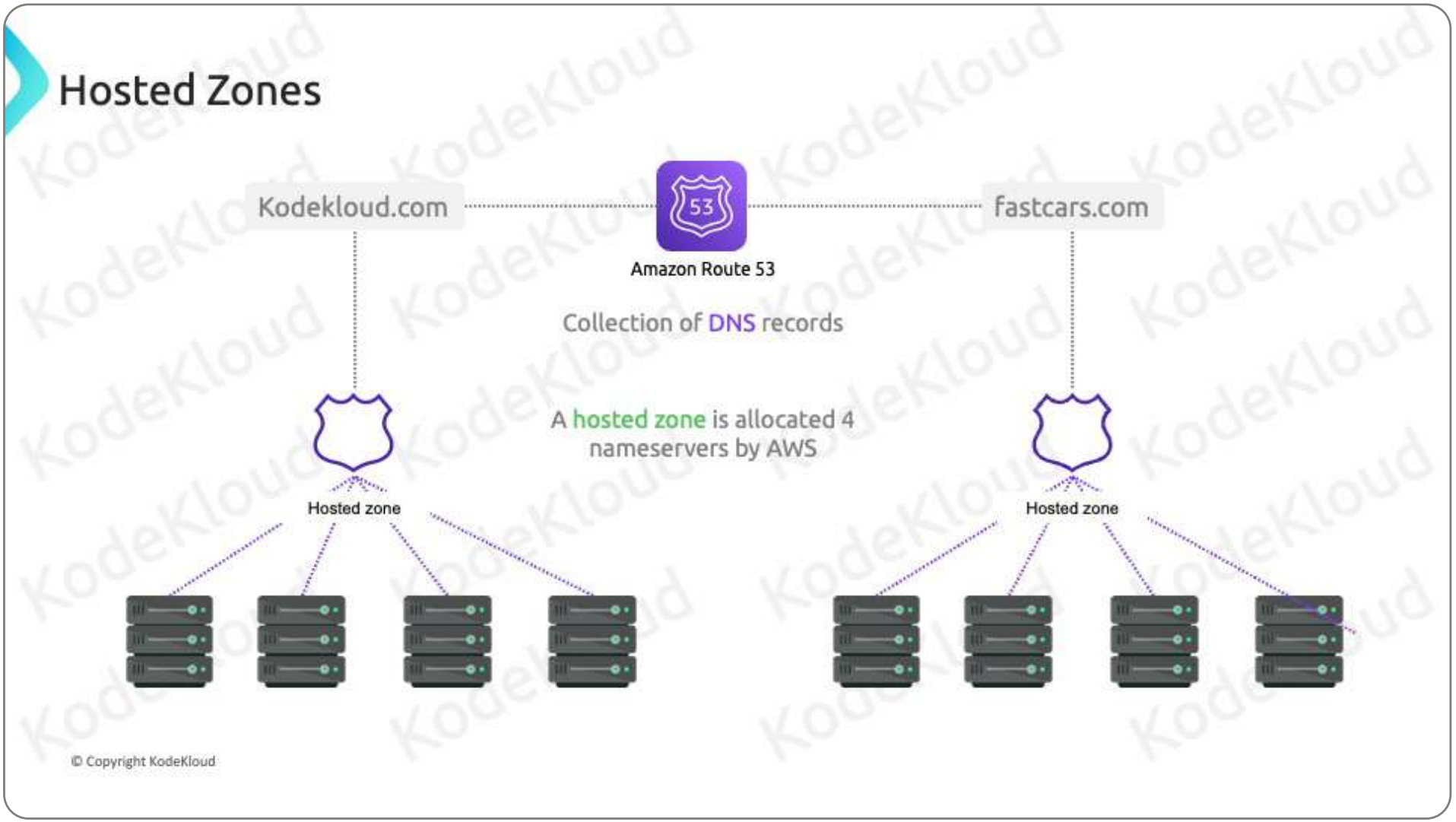
Route 53

An AWS managed DNS service

Acts as a domain registrar – purchase domains similar to sites like Godaddy and Namescheap

I want to set up a website called www.kodekloud.com

A global service – not specific to a region



Now when you either purchase a domain from route53 or you configure route53 to manage a domain purchased from some place else

Each hosted zone is allocated 4 nameservers to hsot this hosted zone

Summary

01 AWS managed DNS service

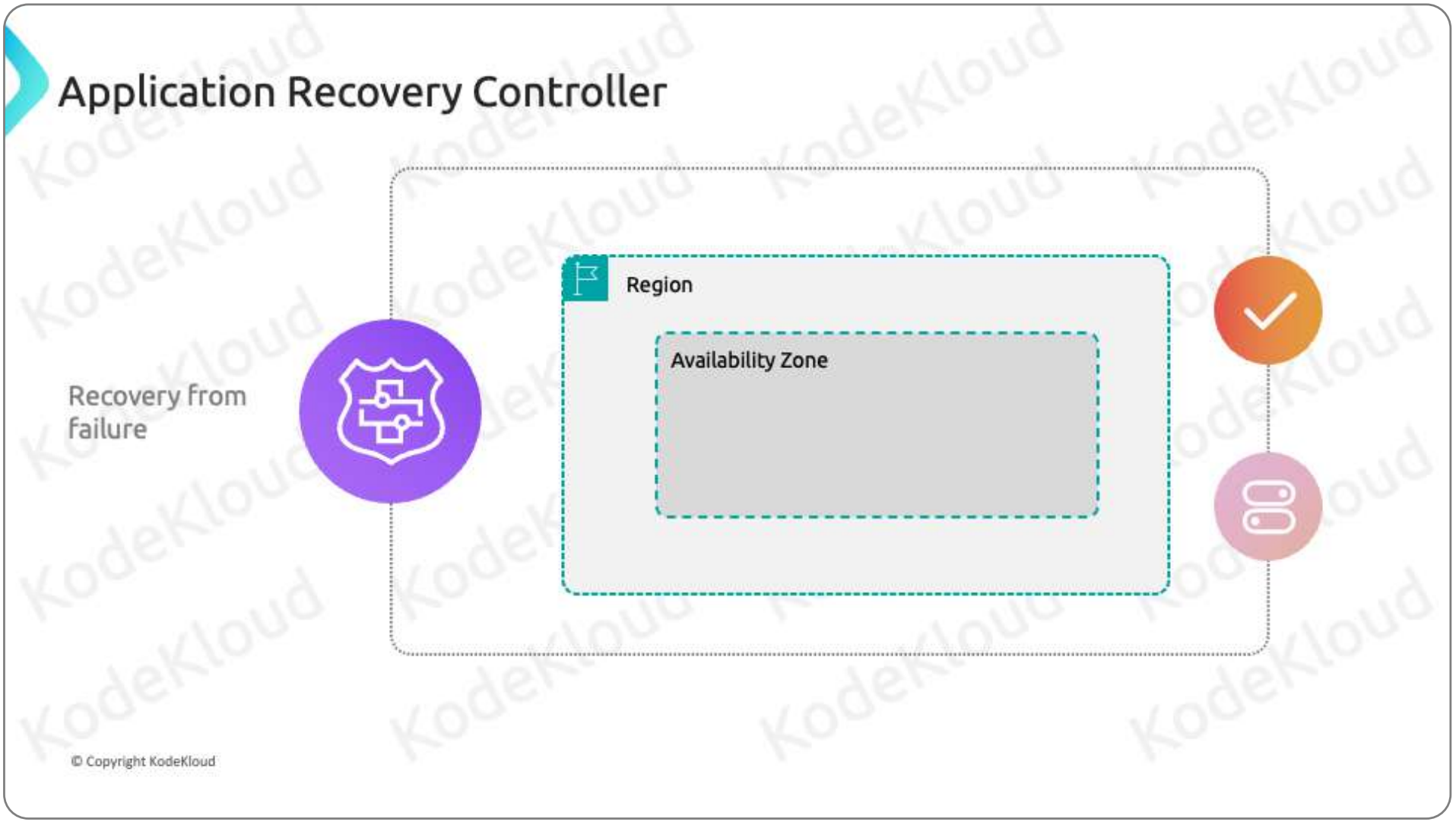
02 Route 53 is a global service

03 Hosted zones is a collection of DNS records

Service Section

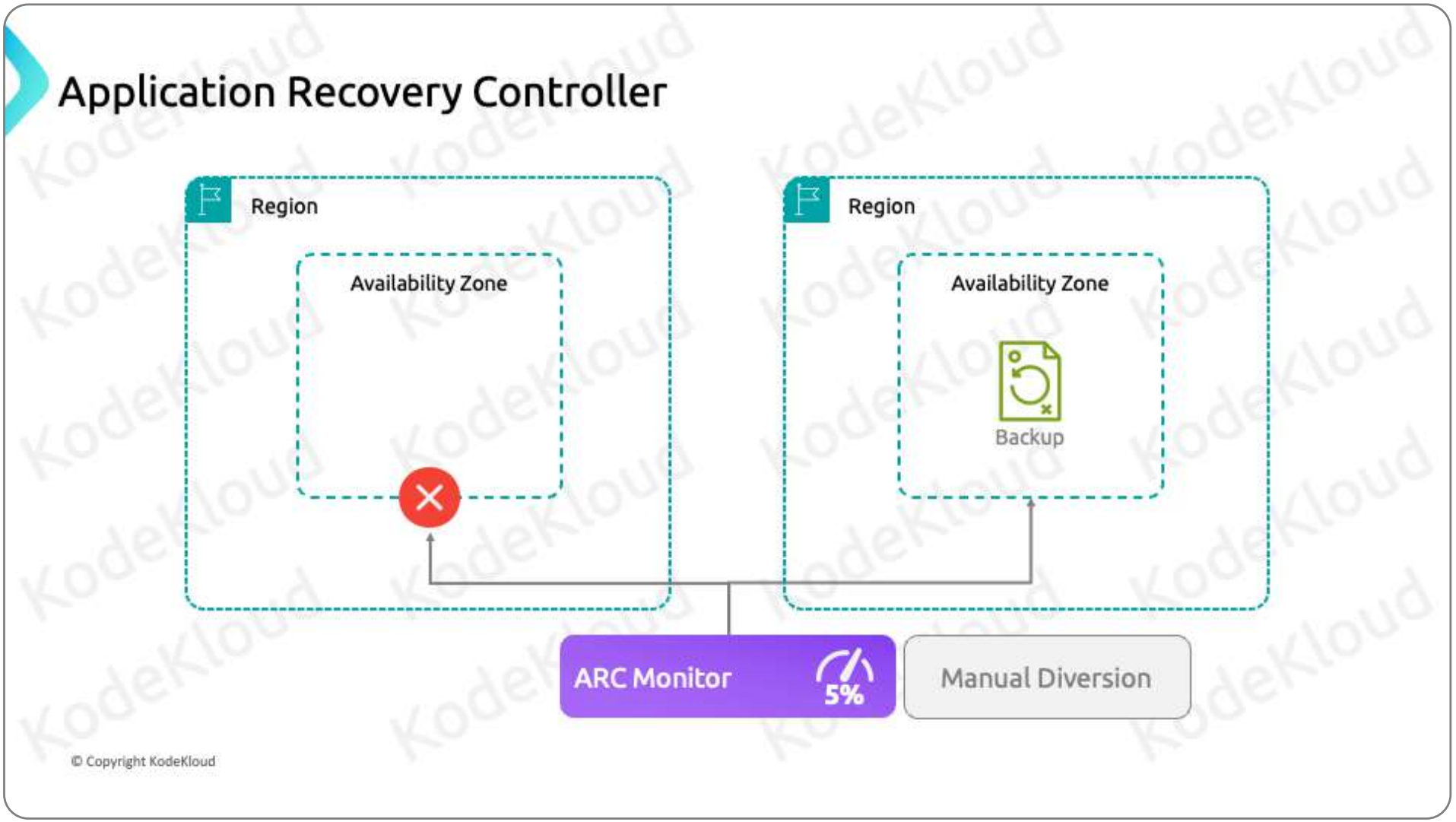
The Power of Core Networking

Route53 Application Recovery Controller



- Amazon Route 53 Application Recovery Controller is a service that continually monitors your applications ability to recover form failrues, and to control application recovery across multiple AWS Regions, and Availability zones.

Route 53 ARC's capabilities make application recovery simpler and more reliable by eliminating manual steps required by traditional tools and processes.

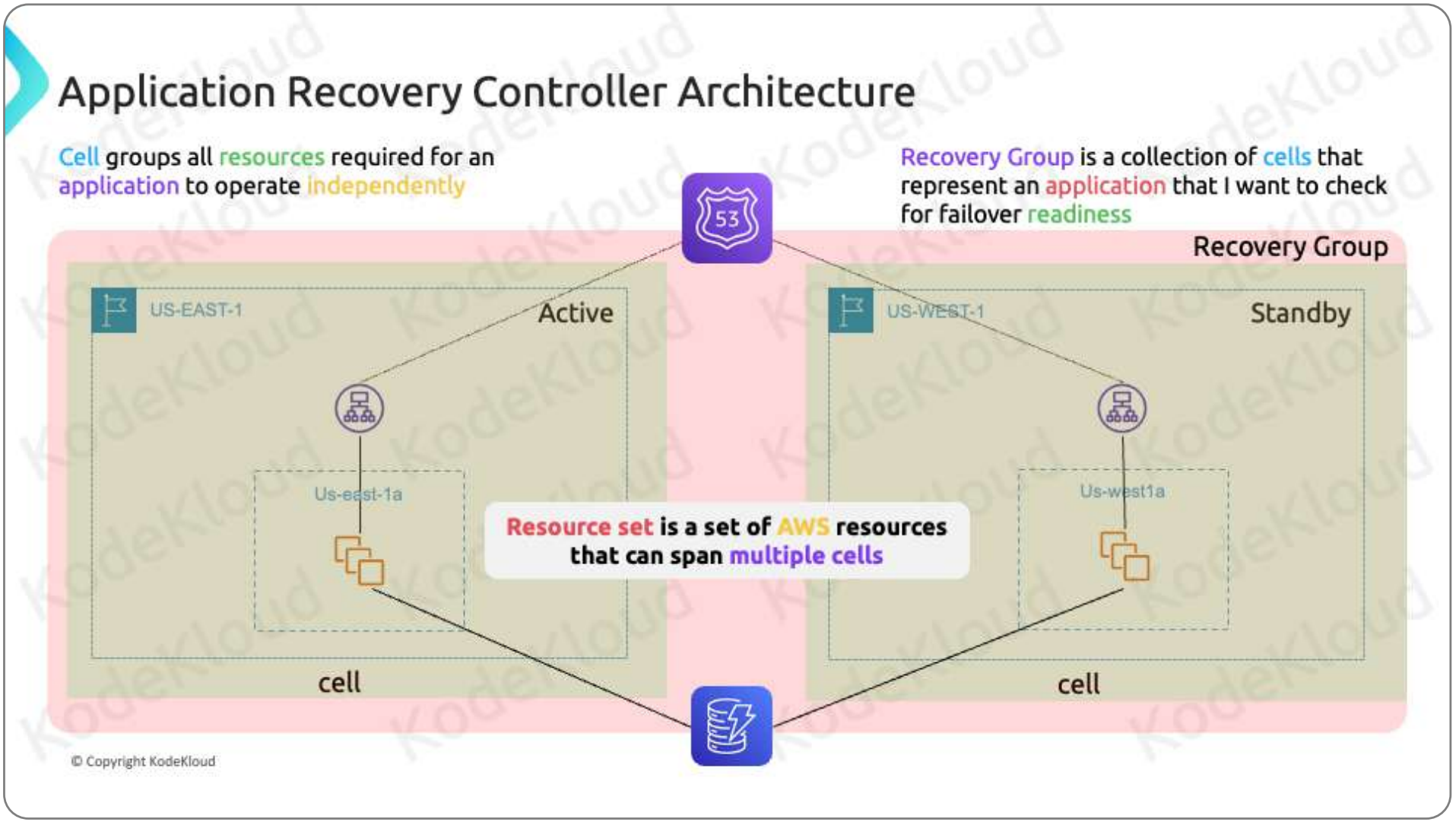


So if you have an application that is deployed in one region, you can have a backup or standby deployment of your application in another region. And what application recovery controller would do is make sure that the recovery environment is up and running, scaled and configured properly and continuously monitor it.

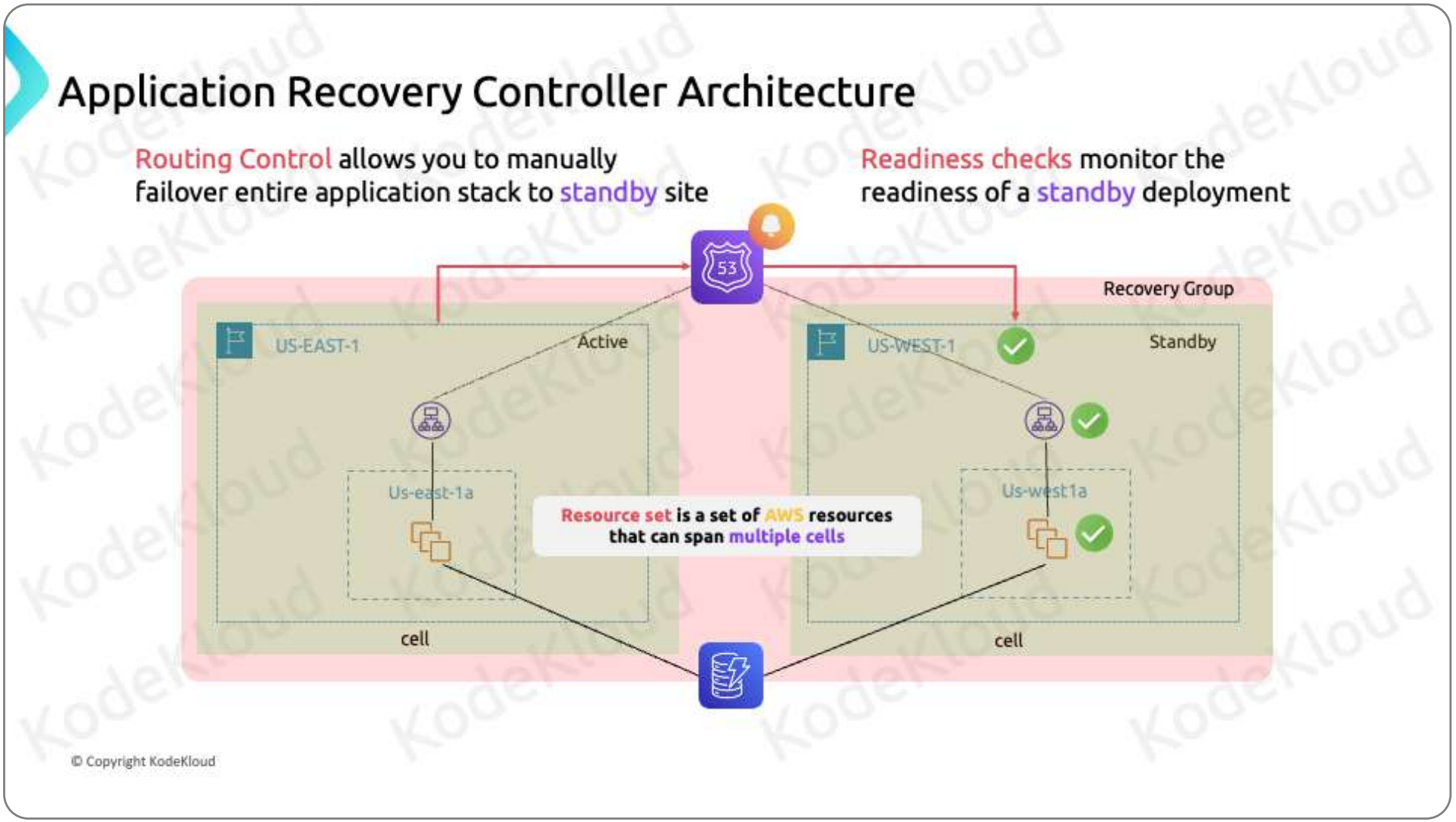
Then based on predefined configuration we can have ARC monitor the primary deployment and when application performance degrades due to a certain metric like 5% increase error rate or increased latency then ARC can divert traffic to

the standby deployment.

In addition, ARC can be used to manually divert traffic away from a deployment during planned maintenance.



The readiness check status for each Regional and zonal cell, as well as the status for each resource set, is shown in Figure 3. The readiness check for each cell has evaluated to Ready across all the different readiness rules applied for the NLB, Auto Scaling group, and Aurora DB Cluster in both Regions. Because the statuses have evaluated to Ready, this helps confirm that the configurations are matching and that the runtime checks for these resources are passing.



Routing control allows you to failover between the active/standby deployments either manually or through automated health checks by updating route53 to point to the load-balancer at the other site.

D[design team] show that route53 gets updated to send traffic from active site to then going to standby site with an animation

Summary

01 Amazon Route 53 Application Recovery Controller is a service that continually monitors your applications' ability to recover from failures

02 Route 53 ARC's capabilities make application recovery simpler and more reliable by eliminating manual steps

03 Cells are silos that contain an application's independent unit of failover. They group all AWS resources required for your application to operate independently

04 Recovery Group is a collection of cells that represent an application or group of applications that you want to check for failover readiness

Summary

05 Resource sets are resources that span multiple cells

06 A readiness check validates a set of AWS resources' readiness to be failed over to

KodeKloud