



Welcome

# Cloud Networking with AWS

 **Develop**Intelligence

A PLURALSIGHT COMPANY

Hello



## About me...



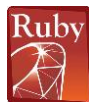
- 25+ years in the industry
- 20+ years in teaching
- Certified Cloud architect
- Passionate about learning
- Also, passionate about Reese's Cups!



## Why study this subject?

- Cloud is everywhere and it's not going away
- As with many topics in technology, there are multiple options and multiple dimensions to those options
- Understanding how to securely configure a network topology in the Cloud to support your workloads is critical

## We teach over 400 technology topics



**You experience our impact on a daily basis!**





## My pledge to you

### I will...

- Make this interactive
- Ask you questions
- Ensure everyone can speak
- Use an on-screen timer



## Objectives

**At the end of this course you will be able to:**

- Speak to some specific networking concepts for the cloud and AWS
- Describe the high-level purpose and function of key AWS networking services
- Create simple instances of AWS networking services and components



# Agenda

- Application Hosting
- Amazon VPC & Subnets
- Amazon API Gateway
- Amazon CloudFront
- Application Load Balancers in AWS
- Hub-and-Spoke Network Topology





## How we're going to work together

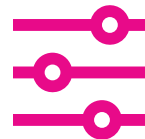
- Lecture & slides
- Class discussions
- Demos
- Hands-on labs

# The Cloud – Application Hosting



## What Do We Mean by Hosting?

- The target infrastructure and runtime platform that will be employed for deployment and execution of an application or system
- Can include compute (CPU and server resources), storage, network, data and operating system



## What Are the Hosting Options with Cloud?

- IaaS
- PaaS
- Serverless / FaaS
- SaaS
- Containers

What do they all mean?



## Infrastructure-as-a-Service (IaaS)

- Involves the building out (and management) of virtual instances of:
  - Compute
  - Network
  - Storage
- Akin to spinning up a server (physical or virtual) in your location or data center complete with disks and required network connectivity
- The difference is in the where – instead of in your data center, it is created in a data center managed by one of the public Cloud providers
- Your organization is responsible for patching the OS, ensuring all appropriate security updates are applied and that the right controls are in place to govern interaction between this set of components and other infrastructure



## Platform-as-a-Service (PaaS)

- Involves leveraging managed services from a public Cloud provider
- With this model, an enterprise can focus on management of their application and data vs. focusing on management of the underlying infrastructure
- Patching and security of the infrastructure used to back the managed services falls to the CSP (Cloud Service Provider)
- Many managed services support automatic scale up or down depending on demand to help ensure sufficient capacity is in place
- Part of what is often termed the “Shared Responsibility Model”



## Serverless / Functions-as-a-Service (FaaS)

- Also represents a type of managed service provided by the CSP
- Cost structure is usually consumption-based (i.e., you only pay for what you use)
- Supports many different coding paradigms (C#/.NET, NodeJS, Python, etc.)
- Typically, with Serverless (and PaaS), the consumer is only concerned with the application code and data – elements of the CSP’s “backbone” used to support are managed by the CSP
- Includes more sophisticated automated scaling capabilities – built for Internet scale



## Software-as-a-Service (SaaS)

- Subscription-based application services
- Licensed for utilization over the Internet / online rather than for download and install on a server or client machine
- Fully-hosted and fully-managed by a 3<sup>rd</sup> party
- Of those discussed, often the cheapest option for service consumers
- However, also offers minimal (or no) control, outside of exposed configuration capabilities

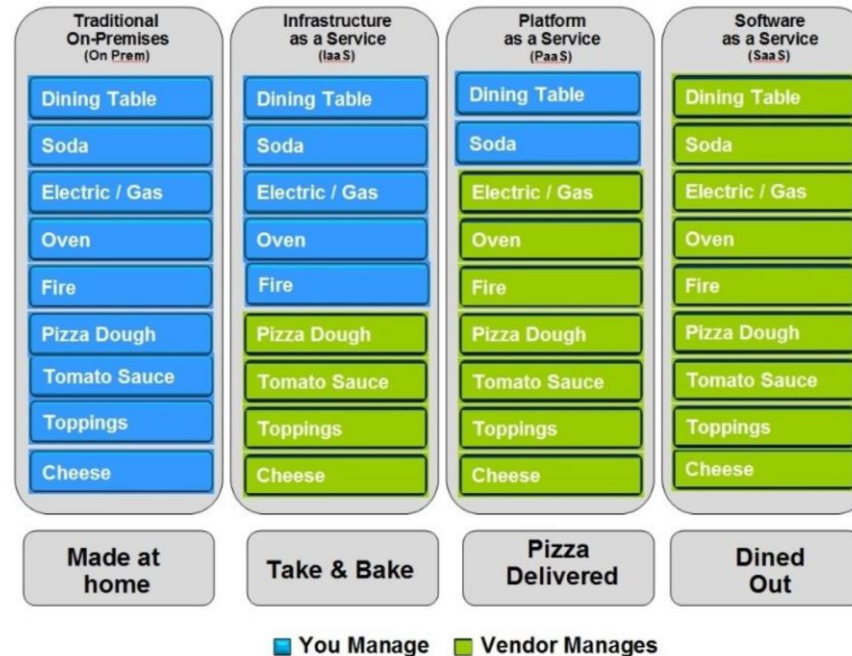


# Pizza-as-a-Service

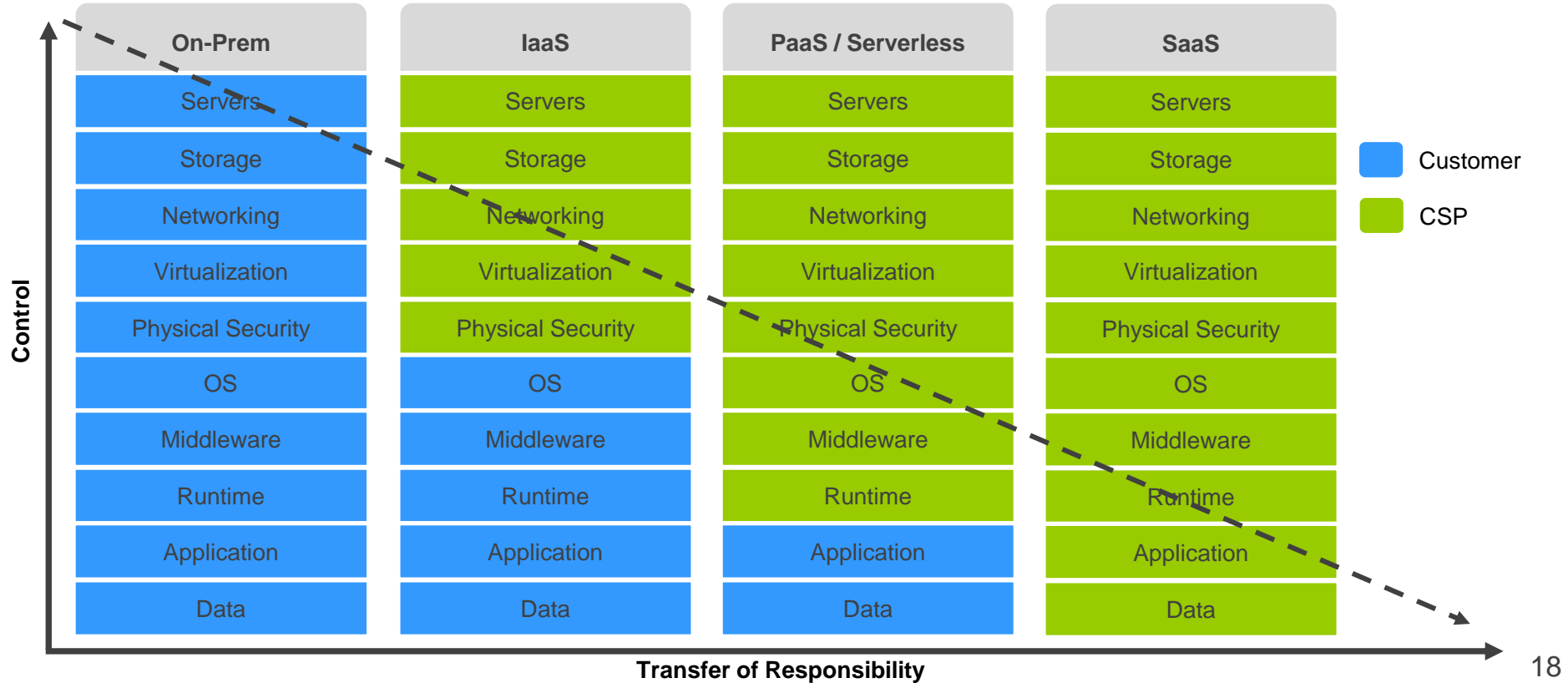
From a LinkedIn post by Albert Barron from IBM (<https://www.linkedin.com/pulse/20140730172610-9679881-pizza-as-a-service/>)



## Pizza as a Service



## Side-by-Side Comparison

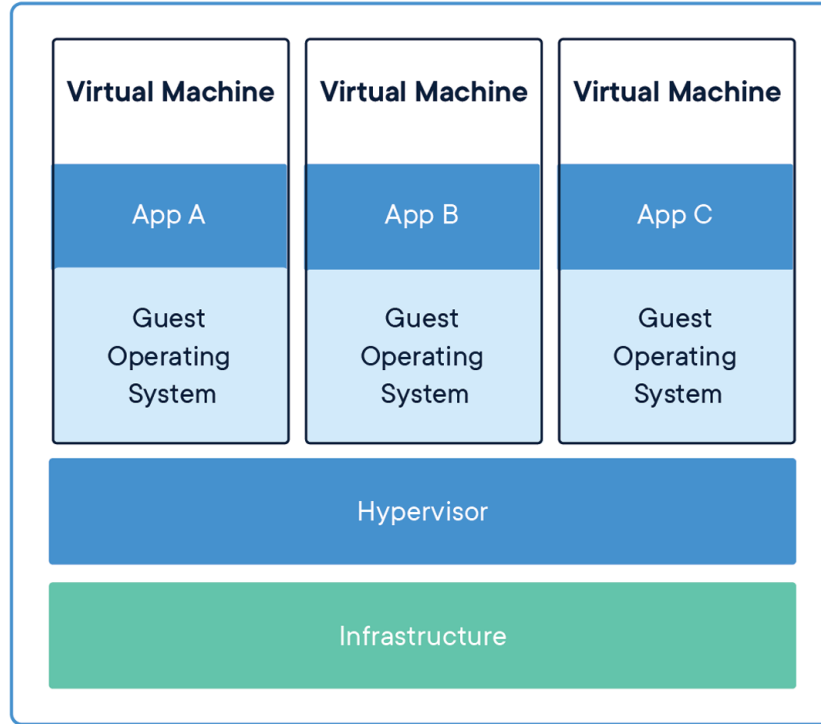




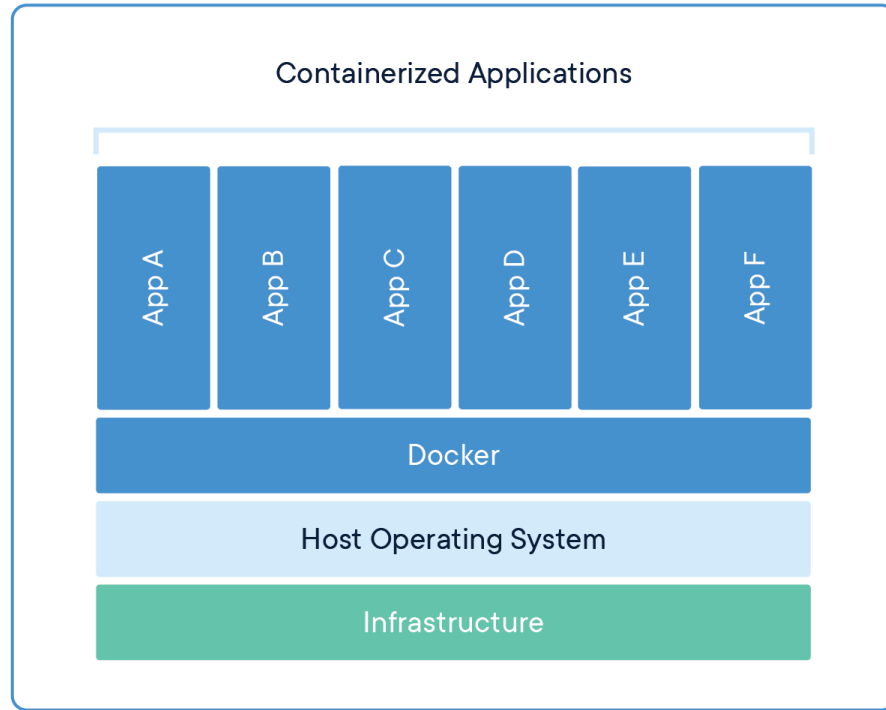
# Containers

- Form of virtualization at the app packaging level (like virtual machines at the server level)
- Isolated from one another at the OS process layer (vs VM's which are isolated at the hardware abstraction layer)
- Images represent the packaging up of an application and its dependencies as a complete, deployable unit of execution (code, runtime and configuration)

# Virtual Machines



# Containers





# Containers

- A platform (e.g., Docker) running on a system can be used to dynamically create containers (executable instances of the app) from the defined image
- Typically, much, much smaller than a VM which makes them lightweight, quickly deployable and quick to “boot up”
- An orchestration engine (e.g., Kubernetes) might be used to coordinate multiple instances of the same container (or a “pod” of containers) to enable the servicing of more concurrent requests (scalability)



## Which One is Better?

- The answer is “it depends”
- It depends on the type of application
- It depends on the enterprise
- It depends on the skillset and expertise within the organization
- It depends on whether you have budget and opportunity to modernize an application environment (in some cases)
- The best option might be a combination of multiple approaches – right tool for the right job



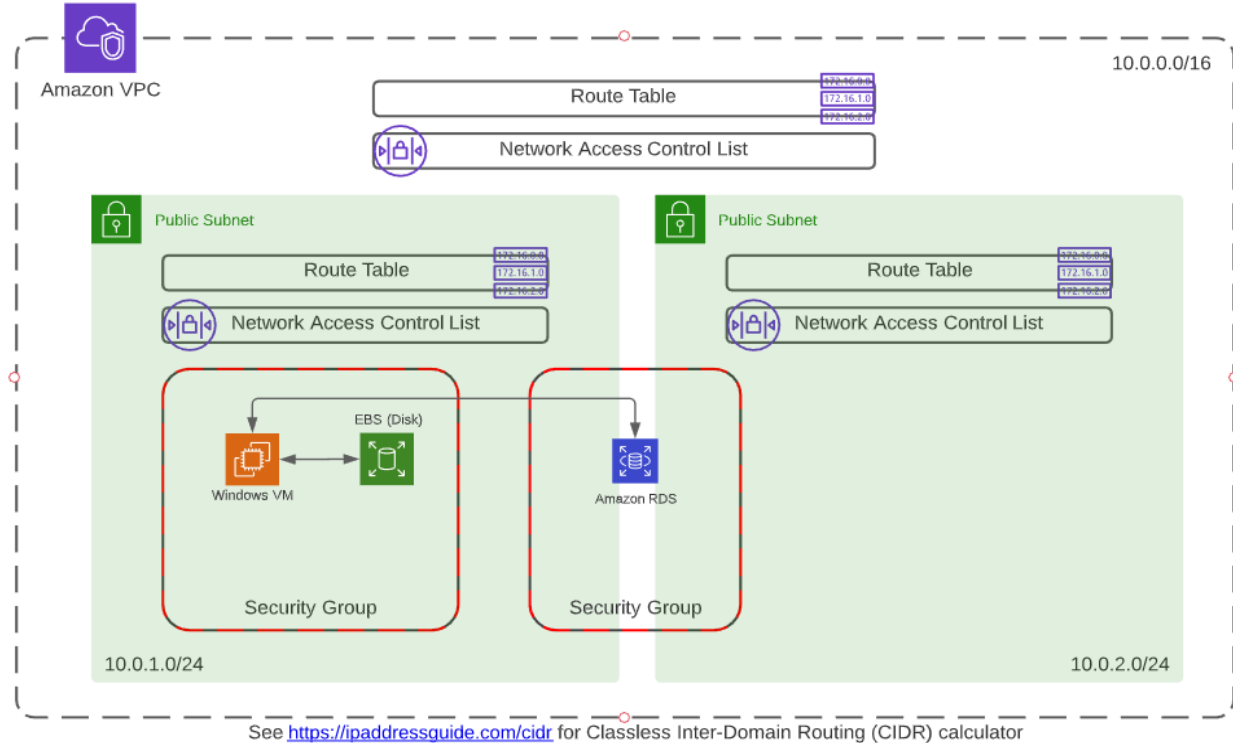
## So, What Does This Have To Do With Networking?

- The option (or options) selected will dictate the amount of network configuration required
- For some of the options, you can take more of a “managed” approach to networking, allowing AWS to handle the configuration for you
- Or you can drive and define network configuration at lower levels of granularity and control
- Regardless of the approach taken, be aware of how your network is configured, how it is secured, and how it is monitored in the Cloud



# PaaS – Create an RDS Instance in AWS

*Demo*



# AWS (Amazon Web Services)



## AWS (Amazon Web Services)

- Management console accessible at <https://aws.amazon.com>
- Identity & Access Management handled via the IAM service
- Users, groups, roles, and policies
- Unit of access & activity is the “account”

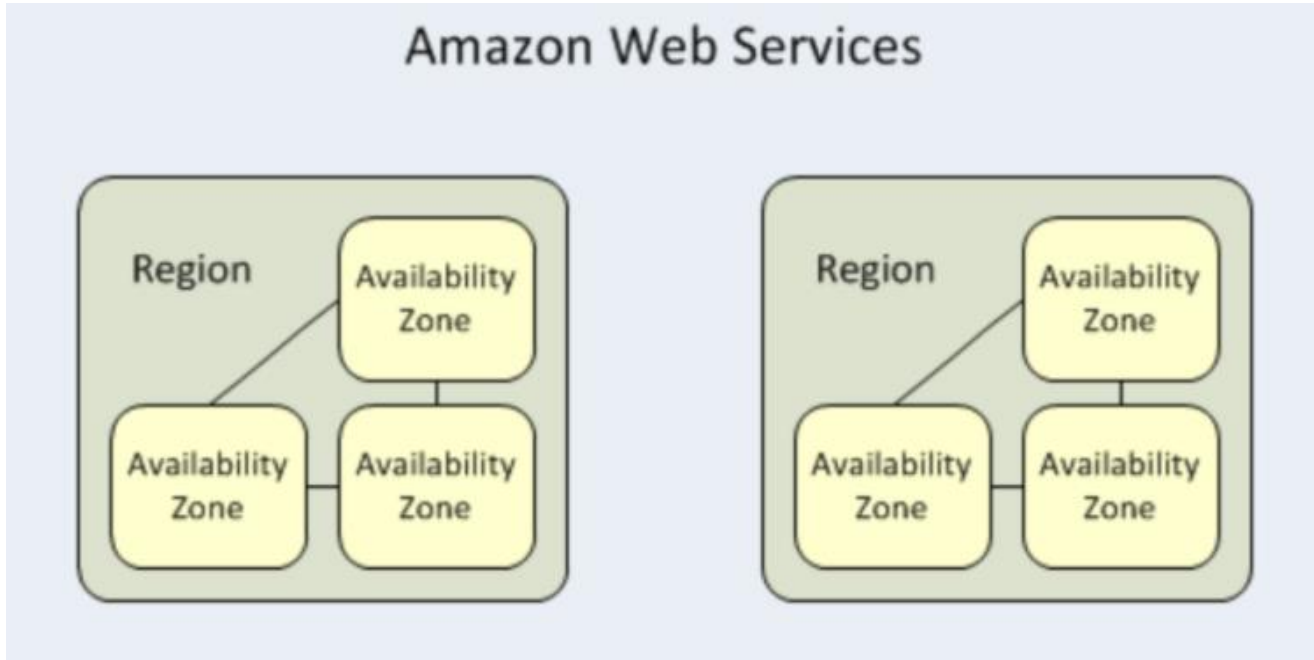


## AWS (Amazon Web Services)

- Uses concept of “region” for geographical location of resources
- Multiple geographies and multiple availability zones within geography

# AWS (Amazon Web Services)

- See [https://aws.amazon.com/about-aws/global-infrastructure/regions\\_az/](https://aws.amazon.com/about-aws/global-infrastructure/regions_az/) for additional detail





## Lab00 - Setup

# Amazon VPC



## Amazon VPC

- VPC stands for Virtual Private Cloud
- Supports creation of a virtual network sectioned off within AWS's infrastructure
- Company can use as their own for hosting Cloud workloads
- Multiple ways to create & configure, including via Management Console, CLI, and Infrastructure-as-Code (IaC)



# Amazon VPC

- As part of VPC creation, able to further sub-divide using subnets
- Subnets can be public or private (i.e., intended for access via the wider Internet or intended only for backend integration between your Cloud-hosted resources)
- You define the hierarchy and associate IPv4 or IPv6 configuration for addressing using CIDR (Classless Inter-Domain Routing → <https://www.geeksforgeeks.org/classless-inter-domain-routing-cidr/>)
- Also, able to configure other types of network components to support and manage connectivity



## Additional Networking Components

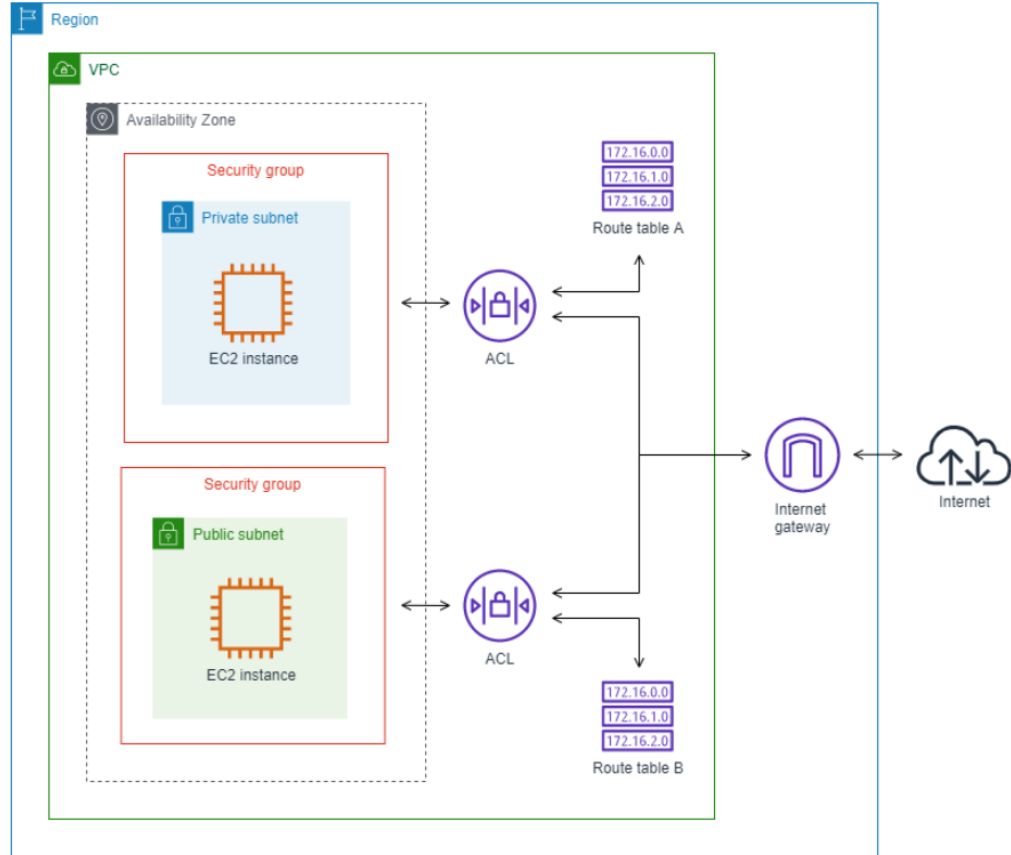
- Network ACLs – Access Control Lists that provide an additional (optional) layer of security at the subnet level
- NAT Gateways – defines Network Address Translation; managed service that allows EC2 instances to send outbound traffic to internet but prevent inbound
- Route Tables – defines a set of rules (routes) used to direct traffic to and from your subnet or gateway
- Security Groups – provides a virtual firewall at the instance level; uses 5 tuple-like rules for managing connectivity between resources

## Security Groups vs. Network ACLs

Security group	Network ACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
Is stateful: Return traffic is automatically allowed, regardless of any rules	Is stateless: Return traffic must be explicitly allowed by rules
We evaluate all rules before deciding whether to allow traffic	We process rules in order, starting with the lowest numbered rule, when deciding whether to allow traffic
Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on	Automatically applies to all instances in the subnets that it's associated with (therefore, it provides an additional layer of defense if the security group rules are too permissive)

Source: [https://docs.aws.amazon.com/vpc/latest/userguide/VPC\\_Security.html](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Security.html)

# Layers of Security



Source: [https://docs.aws.amazon.com/vpc/latest/userguide/VPC\\_Security.html](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Security.html)



## Default VPCs

- From the start, a default VPC is provided in each AWS region for your account
- Comes with a public subnet (default) in each Availability Zone in the region, an internet gateway, and DNS resolution
- Can be used for quickly standing up simple workloads
- For more complex workloads, you will likely want to create components specific to your workload to ensure better isolation and security

## Default VPCs

- You can delete the default VPC and default subnets in a region
- If you do, you'll be forced to define VPCs and subnets (and supporting components) as part of any workloads you build out for run in that region/Availability Zone
- If you delete, you cannot restore and you cannot designate a non-default to be the new default
- Instead, you can create a default VPC and default subnets as needed (<https://docs.aws.amazon.com/vpc/latest/userguide/default-vpc.html#create-default-vpc> and <https://docs.aws.amazon.com/vpc/latest/userguide/default-vpc.html#create-default-subnet>)



## Elastic IP Address

- Static, public IPv4 address that provides a layer of addressing abstraction for your Cloud resources
- Can be associated with a specific instance or a network interface in one of your account's VPCs
- Can be re-associated with a different resource (e.g., on failure) to provide the appearance of uninterrupted network access to your workloads



## VPC Peering

- You can connect two different VPCs enabling routing of private traffic between their resources
- Peered VPCs look like they are part of the same “network”
- Supports peering between your own VPCs, between your VPC and a VPC in another account, or between your VPC and VPC in another region
- One VPC submits a request for peering to a target VPC – once the target VPC accepts, peering is established
- Route tables, security groups, and Network ACLs are updated to enable traffic between the peered VPCs





## Lab01 – VPCs and Subnets



## Lab02 – VPC Peering

# Amazon API Gateway

# Capabilities Provided by an API Gateway



Reduce coupling between API consumer & API producer



Aggregate & translate backends to simplify consumption

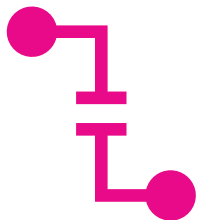


Detect & help mitigate security threats



Promote observability, traceability, & visibility

## Reduce Coupling Between API Consumer & Producer



- API Gateway provides an intermediary between a consumer & one or more data producers
- Helps to reduce coupling as consumer only needs to know about the gateway, not all the underlying API endpoints used to service the request

## Aggregate & Translate Backends to Simplify Consumption



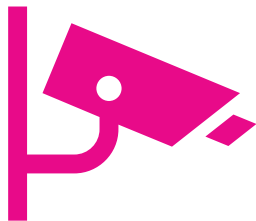
- API Gateway can be used to aggregate results from multiple data sources into a single, unified contract
- Able to support various types of translation between consumer & producer (if required)
- Can include protocol translation, message format translation, etc.

## Detect & Help Mitigate Security Threats



- API Gateway can provide first line of defense for incoming requests
- Can include specific security protections like TLS termination, AuthN/AuthZ, & access management at the IP level
- Can provide helpful features like request throttling to protect against DDoS

## Promote Observability, Traceability, & Visibility



- API Gateway can provide detail crucial to measuring & tracking metrics & Key Performance Indicators (KPIs) relative to API usage & consumption
- Can be used to automatically inject correlation IDs (e.g., GUIDs) for end-to-end traceability
- Detail gleaned from monitoring can help inform proper security configuration





## API Gateway in AWS

- Provides a managed service for hosting, running, and securing multiple types of APIs
- Supports stateful (WebSockets) and stateless (HTTP and REST)
- Also, can be used with Lambda / serverless as a trigger for your hosted function
- Provides the previously discussed capabilities to APIs you're hosting in AWS



## API Gateway in AWS

- Supports granular role-based security for securing access
- Supports API key as a mechanism for associating a call to a caller and controlling caller's access

# Architecture of API Gateway in AWS



Source: <https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html#api-gateway-overview-aws-backbone>



## API Gateway in AWS

- Supports both REST and HTTP APIs
- Both are RESTful implementations, but the REST API in API Gateway supports more features than the HTTP API
- Choosing between the two →  
<https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-vs-rest.html>



## Demo – API Gateway with Lambda



## Lab03 – API Gateway with Lambda

# Amazon CloudFront



## Amazon CloudFront

- Amazon CloudFront provides a CDN (Content Delivery Network)
- Provides a worldwide network of edge locations for enabling lowest latency access to static and dynamic web assets (e.g., .html, .css, .js, and images)
- When user requests content, if content already available at the edge location with lowest latency, CloudFront serves from there
- If content is not already available there, CloudFront retrieves from a user-defined origin and caches it for future access



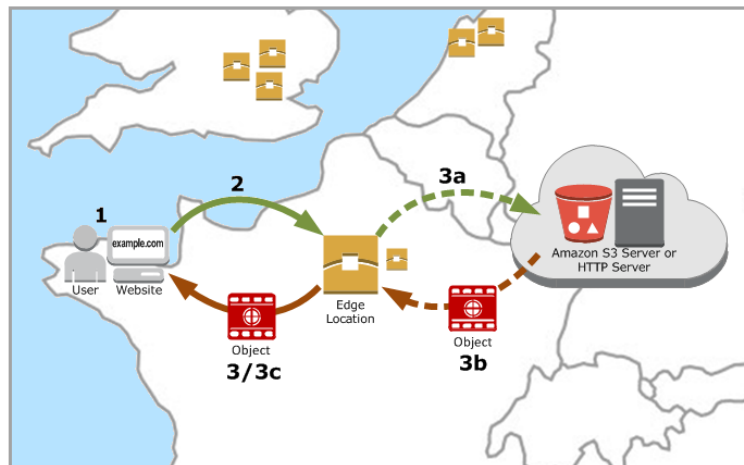
# Amazon CloudFront

## How CloudFront delivers content to your users

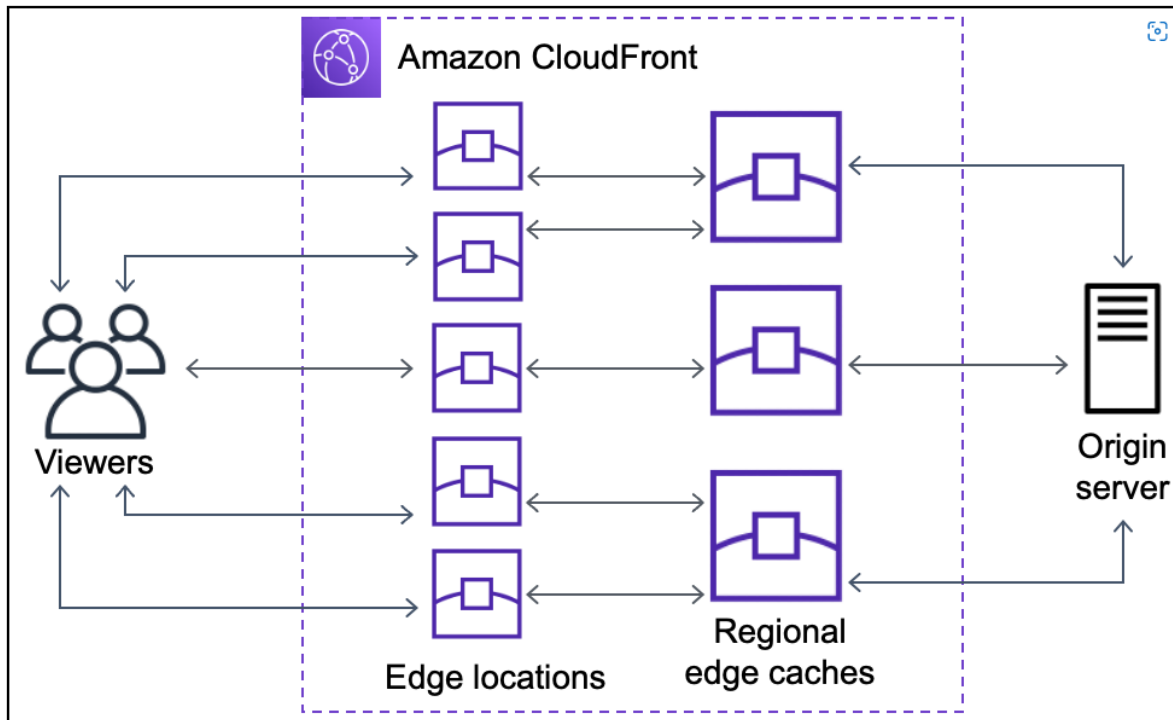
After you configure CloudFront to deliver your content, here's what happens when users request your objects:

1. A user accesses your website or application and sends a request for an object, such as an image file or an HTML file.
2. DNS routes the request to the CloudFront POP (edge location) that can best serve the request—typically the nearest CloudFront POP in terms of latency—and routes the request to that edge location.
3. CloudFront checks its cache for the requested object. If the object is in the cache, CloudFront returns it to the user. If the object is *not* in the cache, CloudFront does the following:
  - a. CloudFront compares the request with the specifications in your distribution and forwards the request to your origin server for the corresponding object—for example, to your Amazon S3 bucket or your HTTP server.
  - b. The origin server sends the object back to the edge location.
  - c. As soon as the first byte arrives from the origin, CloudFront begins to forward the object to the user. CloudFront also adds the object to the cache for the next time someone requests it.

*POP = Point of Presence*



# Amazon CloudFront



- Regional edge caches provide caching for less “popular” resources
- Support a larger cache than the regular POP edge locations



## Amazon CloudFront - Distributions

- Used to define key details about the content that should be distributed via CloudFront
- Includes content origin, access control for the assets, and whether you want access to be via HTTPS
- Also, includes cache key used to uniquely identify each asset (and manage how cached)
- Can define settings for how requests are made to your origin (e.g., HTTP headers, cookies, query strings)
- Can define any required restrictions for access to content from a given geography



## Amazon CloudFront - Origins

- Available origin types include an S3 bucket, an ALB (Application Load Balancer), or a URL for a Lambda function
- Could also include a URL for custom HTTP server (e.g., hosted on EC2)
- Able to use CloudFront origin groups to provide high availability – primary origin plus secondary origin for failover
- Supports the addition of CNAMEs (alternate domain names) for use of custom domain to access CloudFront distribution



## Amazon CloudFront – Cache Management

- With CloudFront, key is correctly balancing performance and asset “staleness”
- Default cache expiration is 24 hours
- Additionally, cache preference is given to more popular assets, potentially ejecting less popular at the POP (regional edge cache can still hold reference)
- Able to control caching against a path pattern using Minimum TTL, Maximum TTL, and Default TTL settings in CloudFront



## Caching Content Based on Query String Params

- On a distribution in CloudFront, able to configure to forward query strings to the origin
- Able to cache based on all or a subset of those parameters
- Could be useful, for example, for caching content for a page based on a language query string parameter



## Caching Content Based on Cookies

- Typically, CloudFront does not consider the cookie content when determining whether or not to serve cached content
- Able to configure forward of cookies (some or all) on request to origin for caching different versions based on cookie values
- Could be useful, for example, for caching content based on a cookie that identifies the country of origin for request



## Caching Content Based on Request Headers

- CloudFront allows forward of headers on request to origin
- Can configure caching of separate versions based on request header values
- Could be useful, for example, for caching content based on user's device type, their location, and the language the viewer is using (among other things)





## Lab04 – CloudFront

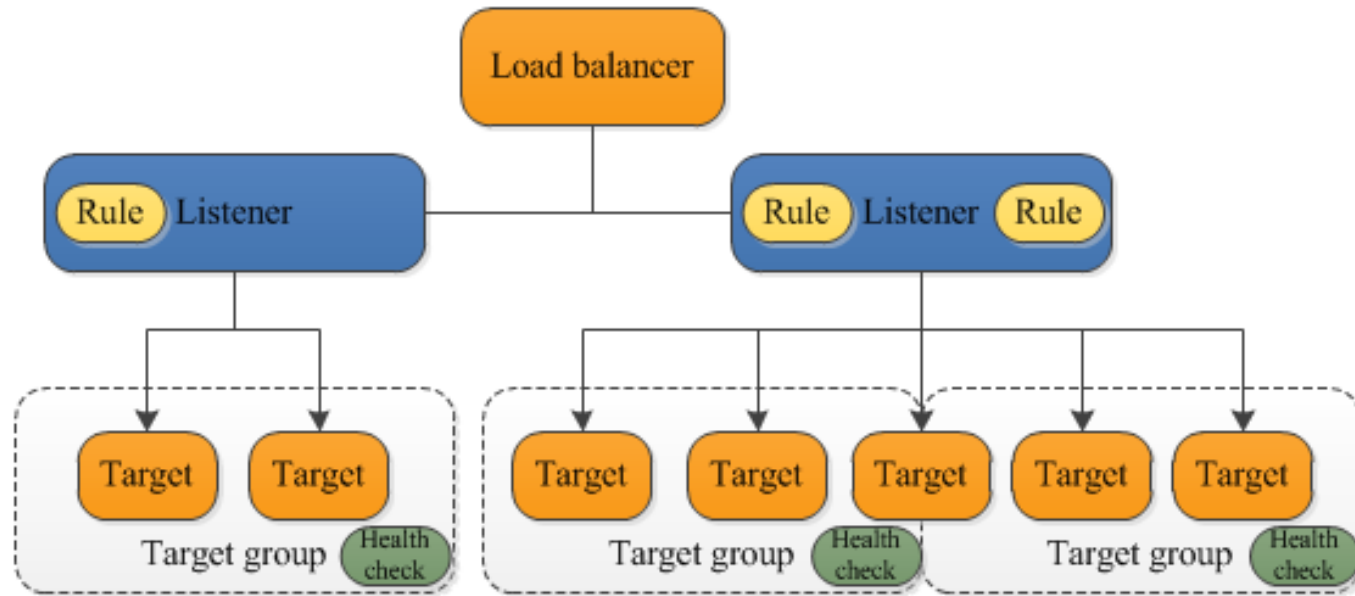
# Application Load Balancers (ALB)



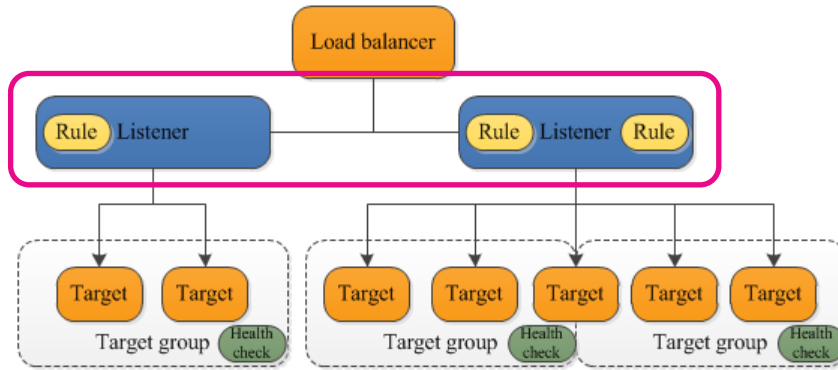
## Application Load Balancers (ALB)

- Represents one type of Elastic Load Balancer (ELB) in AWS
- Provides routing and load balancing at the 7<sup>th</sup> layer of the OSI (Open Systems Interconnection) model
- AWS also supports Network Load Balancers (NLB) which operate at the 4<sup>th</sup> level of OSI – closer to “bare metal” with improved performance

## Application Load Balancers (ALB)

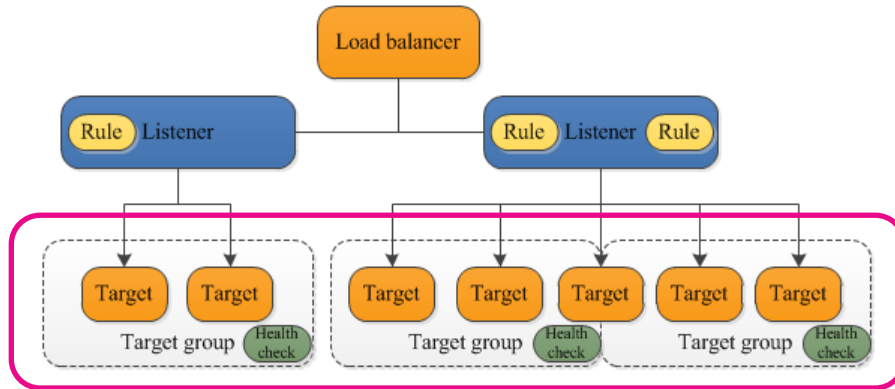


# Application Load Balancers (ALB) – Listener Rules



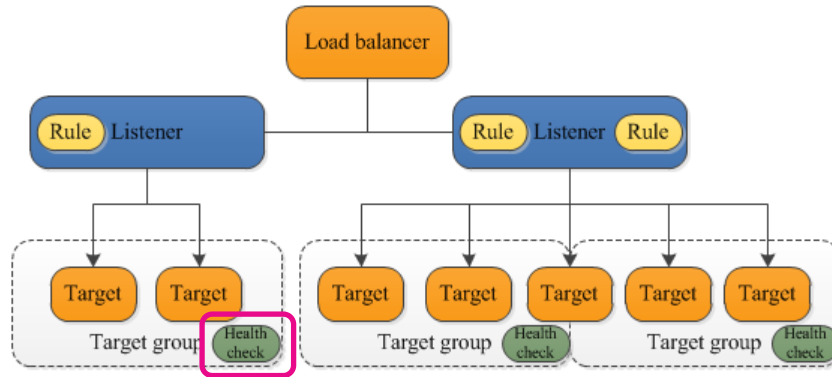
- Defines a configurable set of rules that can be evaluated in priority order
- Helps with determining which rule to apply – layer of abstraction to support intelligent routing
- Rule used to select a target for action

# Application Load Balancers (ALB) – Targets



- Defined in target groups – sets of targets that are configured to handle a given rule
- Multiple, potential targets is opaque to the caller – caller hits an endpoint, and routing gets the request to a target that can serve
- Able to configure routing algorithm for targets/target groups – round robin is default

# Application Load Balancers (ALB) – Health Checks




- Configurable monitors used to monitor and confirm the health of viable targets
- Load balancer uses to ensure that requests are only sent to “healthy” targets
- This routing, again, is transparent to the caller



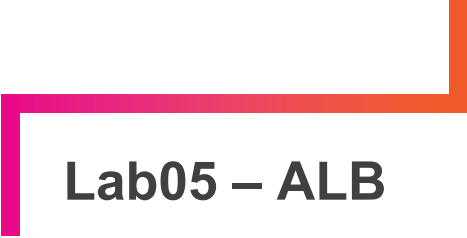
## ALB vs. NLB

<https://medium.com/awesome-cloud/aws-difference-between-application-load-balancer-and-network-load-balancer-cb8b6cd296a4>





## Demo – ALB



## Lab05 – ALB

# Hub-and-Spoke Network Topology



## Demo – Hub-and-Spoke

<https://docs.aws.amazon.com/whitepapers/latest/building-scalable-secure-multi-vpc-network-infrastructure/vpc-to-vpc-connectivity.html>

<https://learn.acloud.guru/handson/e16daea9-15ae-41fe-8ae6-3c67a609a291>

# Case Study – For Discussion

## Case Study – Discussion

Scenario: you would like to “sell” to the business a modernization effort to migrate a monolithic, mainframe-based application used to manage a critical sales & distribution process within the company to a microservices architecture with significant leverage of APIs. Included in the target goals would be the transition of the workflow from the data center to Cloud and Cloud native technologies (where possible).

Currently, the application leverages a mainframe-based relational data store to store and serve all data within the large and tightly-coupled workflow. There are several legacy technologies in play and the platform upon which the application is deployed is several versions behind latest. Key reports that the business needs for critical decisioning exist but cannot be delivered in real-time. They are requested in batch and sent as a follow-up, sometimes 1 or 2 days later.

Also, the business would like to be able to leverage the different dimensions of data housed within this system for Machine Learning purposes but, at present, the data is difficult to extract and utilize. The current mainframe system supports external customer access through a Web portal, internal access for account & order management (including remote sales associates utilizing a mobile device), and a handful of API endpoints that are used by external partners to enable authorized third-party sales.

Let’s discuss:

- Identify and describe the key networking requirements that will be needed to guide this solution
- Think about and discuss specific AWS networking components & services that you might use to satisfy those requirements



## Resources

<https://docs.aws.amazon.com/whitepapers/latest/aws-overview/networking-services.html>



# Thank you!

If you have additional questions,  
please reach out to me at:  
(email address)