Amazon

(AWS Certified Advanced Networking - Specialty ANS-C01)

AWS Certified Advanced Networking - Specialty ANS-C01

Total: **272 Questions**

Link:

Question: 1

A company is planning to create a service that requires encryption in transit. The traffic must not be decrypted between the client and the backend of the service. The company will implement the service by using the gRPC protocol over TCP port 443. The service will scale up to thousands of simultaneous connections. The backend of the service will be hosted on an Amazon Elastic Kubernetes Service (Amazon EKS) duster with the Kubernetes Cluster Autoscaler and the Horizontal Pod Autoscaler configured. The company needs to use mutual TLS for two-way authentication between the client and the backend.
Which solution will meet these requirements?

A. Install the AWS Load Balancer Controller for Kubernetes. Using that controller, configure a Network Load Balancer with a TCP listener on port 443 to forward traffic to the IP addresses of the backend service Pods.

B. Install the AWS Load Balancer Controller for Kubernetes. Using that controller, configure an Application Load Balancer with an HTTPS listener on port 443 to forward traffic to the IP addresses of the backend service Pods.

C. Create a target group. Add the EKS managed node group's Auto Scaling group as a target Create an Application Load Balancer with an HTTPS listener on port 443 to forward traffic to the target group.

D. Create a target group. Add the EKS managed node group's Auto Scaling group as a target. Create a Network Load Balancer with a TLS listener on port 443 to forward traffic to the target group.

Answer: A

Explanation:

Here's a detailed justification for why option A is the most suitable solution, and why the other options are less ideal:

Justification for Option A:

Option A leverages a Network Load Balancer (NLB) deployed via the AWS Load Balancer Controller for Kubernetes. This approach directly addresses the core requirements:

1. **Encryption in Transit (End-to-End):** An NLB operates at the transport layer (Layer 4), specifically TCP in this case. It forwards traffic based on IP addresses and ports without inspecting the content. This ensures that the TLS encryption, crucial for the gRPC protocol and mutual TLS, remains intact from the client to the backend pods. No decryption happens at the load balancer level.
2. **Mutual TLS:** Mutual TLS (mTLS) necessitates that both the client and the server (backend pods) authenticate each other using certificates. The NLB's role in this setup is to purely forward encrypted traffic, allowing the backend pods to handle the TLS handshake and certificate validation directly. 3. **Scalability:** NLBs are designed for high throughput and low latency, scaling to handle thousands of simultaneous connections efficiently. They automatically handle the load distribution across available backend pods. The Kubernetes Cluster Autoscaler and Horizontal Pod Autoscaler further enhance scalability by dynamically adjusting the number of pods and nodes as needed.
4. **gRPC over TCP Port 443:** The NLB can be configured with a TCP listener on port 443, which is the standard port for HTTPS/TLS traffic, accommodating the gRPC protocol.
5. **Kubernetes Integration:** The AWS Load Balancer Controller automates the provisioning and management of load balancers based on Kubernetes resources, simplifying deployment and maintenance. It dynamically updates the NLB's target group (backend pods) as the Kubernetes service scales.

Why other options are less suitable:

Option B (Application Load Balancer with HTTPS listener): An Application Load Balancer (ALB) terminates TLS connections. While it provides features like content-based routing, it would decrypt the traffic at the load balancer, violating the requirement that traffic must not be decrypted between the client and the backend. This breaks the end-to-end encryption needed for mTLS and is a fundamental flaw.

Option C (ALB with Auto Scaling Group): Similar to option B, an ALB with an HTTPS listener would terminate TLS at the load balancer level, decrypting the traffic and defeating the purpose of end-to-end encryption and

mutual TLS. Targeting the Auto Scaling group directly bypasses Kubernetes' service abstraction.

Option D (NLB with Auto Scaling Group and TLS Listener): While using an NLB avoids TLS termination, creating a TLS listener on the NLB means the NLB itself becomes responsible for TLS encryption. This isn't what the question asked for, as it changes the mutual TLS relationship, and the application is still meant to be doing this itself.

Authoritative Links:

AWS Load Balancer Controller:https://kubernetes-sigs.github.io/aws-load-balancer-controller/
Network Load Balancer:https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html
Mutual TLS:https://smallstep.com/hello-mtls/
gRPC:https://grpc.io/

In conclusion, option A is the only one that completely aligns with the problem requirements, providing end-to-end encryption, supporting mutual TLS, scaling efficiently, and integrating seamlessly with the Kubernetes environment.

Question: 2

A company is deploying a new application in the AWS Cloud. The company wants a highly available web server that will sit behind an Elastic Load Balancer. The load balancer will route requests to multiple target groups based on the URL in the request. All traffic must use HTTPS. TLS processing must be offloaded to the load balancer. The web server must know the user's IP address so that the company can keep accurate logs for security purposes. Which solution will meet these requirements?

A.Deploy an Application Load Balancer with an HTTPS listener. Use path-based routing rules to forward the traffic to the correct target group. Include the X-Forwarded-For request header with traffic to the targets.

B.Deploy an Application Load Balancer with an HTTPS listener for each domain. Use host-based routing rules to forward the traffic to the correct target group for each domain. Include the X-Forwarded-For request header with traffic to the targets.

C.Deploy a Network Load Balancer with a TLS listener. Use path-based routing rules to forward the traffic to the correct target group. Configure client IP address preservation for traffic to the targets.

D.Deploy a Network Load Balancer with a TLS listener for each domain. Use host-based routing rules to forward the traffic to the correct target group for each domain. Configure client IP address preservation for traffic to the targets.

Answer: A

Explanation:

The correct solution is A because it effectively addresses all the requirements outlined in the scenario.

High Availability & Web Server: An Application Load Balancer (ALB) inherently provides high availability by distributing traffic across multiple target groups (web servers) in different Availability Zones.

HTTPS & TLS Offloading: The ALB supports HTTPS listeners, enabling TLS termination at the load balancer. This offloads the CPU-intensive task of encryption/decryption from the web servers, improving their performance.

URL-Based Routing: ALB's path-based routing rules allow forwarding traffic to different target groups based on the URL in the request. This is crucial for directing requests to specific application components.

User IP Address: Including the X-Forwarded-For request header is the standard way to preserve the original client IP address when TLS is terminated at the load balancer. The ALB automatically adds this header, allowing the web servers to log the client's IP for security purposes.

Option B is less efficient because it suggests creating a separate HTTPS listener for each domain, which can complicate configuration and management if the number of domains increases. While host-based routing is a valid approach, it isn't strictly necessary when URL-based routing already satisfies the pathing requirement.

Options C and D involve a Network Load Balancer (NLB). While NLBs offer extremely high performance and TCP/UDP support, they are not ideal for HTTP/HTTPS traffic and URL-based routing. NLBs operate at Layer 4 (TCP/UDP), while ALBs operate at Layer 7 (Application Layer), enabling more sophisticated routing based on HTTP headers and paths. Moreover, NLBs do not automatically add the X-Forwarded-For header. Although NLBs can preserve the source IP address, using an ALB and X-Forwarded-For header is the best practice for HTTP(S) applications.

In summary, using an ALB with HTTPS, path-based routing, and the X-Forwarded-For header efficiently meets all of the company's requirements for high availability, secure communication, URL-based traffic distribution, and accurate logging.

Authoritative Links:

Application Load Balancers:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html
Network Load Balancers:
https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html
X-Forwarded-For Header:https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-For

Question: 3

A company has developed an application on AWS that will track inventory levels of vending machines and initiate the restocking process automatically. The company plans to integrate this application with vending machines and deploy the vending machines in several markets around the world. The application resides in a VPC in the us-east-1 Region. The application consists of an Amazon Elastic Container Service (Amazon ECS) cluster behind an Application Load Balancer (ALB). The communication from the vending machines to the application happens over HTTPS.
The company is planning to use an AWS Global Accelerator accelerator and configure static IP addresses of the accelerator in the vending machines for application endpoint access. The application must be accessible only through the accelerator and not through a direct connection over the internet to the ALB endpoint.
Which solution will meet these requirements?

A.Configure the ALB in a private subnet of the VPC. Attach an internet gateway without adding routes in the subnet route tables to point to the internet gateway. Configure the accelerator with endpoint groups that include the ALB endpoint. Configure the ALB's security group to only allow inbound traffic from the internet on the ALB listener port.

B.Configure the ALB in a private subnet of the VPC. Configure the accelerator with endpoint groups that include the ALB endpoint. Configure the ALB's security group to only allow inbound traffic from the internet on the ALB listener port.

C.Configure the ALB in a public subnet of the VPAttach an internet gateway. Add routes in the subnet route tables to point to the internet gateway. Configure the accelerator with endpoint groups that include the ALB endpoint. Configure the ALB's security group to only allow inbound traffic from the accelerator's IP addresses on the ALB listener port.

D.Configure the ALB in a private subnet of the VPC. Attach an internet gateway. Add routes in the subnet route tables to point to the internet gateway. Configure the accelerator with endpoint groups that include the ALB endpoint. Configure the ALB's security group to only allow inbound traffic from the accelerator's IP addresses on the ALB listener port.

Answer: A

Explanation:

Here's a detailed justification for why option A is the correct solution:

The core requirement is to ensure that the application is only accessible through the AWS Global Accelerator and not directly via the internet. This implies preventing direct public access to the Application Load Balancer (ALB).

Option A achieves this through the following steps:

1. **ALB in a Private Subnet:** Placing the ALB in a private subnet ensures that it does not have a direct public IP address. This makes it inaccessible directly from the internet.

2. **Internet Gateway Attachment without Routes:** Attaching an Internet Gateway (IGW) to the VPC is necessary for the Global Accelerator to communicate with resources within the VPC. However, by not adding routes in the subnet route tables to the IGW, you prevent resources in the subnet (including the ALB) from initiating outbound internet traffic. This configuration allows the Global Accelerator to initiate traffic to the ALB, but it prevents direct internet access.

3. **Accelerator Endpoint Groups:** Configuring the Global Accelerator with endpoint groups that include the ALB endpoint establishes the connection between the accelerator and the ALB. The Global Accelerator will route traffic to the ALB's private IP address via the VPC.

4. **ALB Security Group:** Configuring the ALB's security group to only allow inbound traffic from the internet on the ALB listener port (HTTPS - usually port 443) is crucial. Because it's a private subnet, this sounds counterintuitive, but it works as follows. The Global Accelerator acts as the entry point to the VPC. When the Global Accelerator sends traffic to the ALB, the source IP will be internet-routable but originating from an AWS service, not the direct client's IP. This ensures that only traffic originating from AWS and routed through the Global Accelerator can reach the ALB. Direct connections to the ALB are blocked.

Option B is incorrect because while the ALB is in a private subnet, allowing inbound traffic from the internet without controlling the source IPs will permit anyone on the internet to directly access the ALB, bypassing the Global Accelerator.

Option C is incorrect because placing the ALB in a public subnet defeats the entire purpose of preventing direct internet access. An ALB in a public subnet, with an internet gateway and appropriate routes, will always be directly accessible via the internet.

Option D is incorrect because configuring the ALB to only allow traffic from the accelerator's IP addresses does not work since the traffic won't originate directly from the accelerator's IPs, but rather it will come through internet routable IPs originating from AWS itself (as it originates from the edge location of the Global Accelerator).In addition, adding routes in the subnet route tables to the internet gateway would also allow resources in the subnet to initiate outbound internet traffic, contrary to the overall architecture.

Here are some helpful links to further understand AWS Global Accelerator:

AWS Global Accelerator:https://aws.amazon.com/global-accelerator/
AWS Global Accelerator Documentation:https://docs.aws.amazon.com/global-accelerator/ **Security Groups:**https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html

Question: 4

A global delivery company is modernizing its fleet management system. The company has several business units. Each business unit designs and maintains applications that are hosted in its own AWS account in separate application VPCs in the same AWS Region. Each business unit's applications are designed to get data from a central shared services VPC.
The company wants the network connectivity architecture to provide granular security controls. The architecture also must be able to scale as more business units consume data from the central shared services VPC in the future.
Which solution will meet these requirements in the MOST secure manner?

A.Create a central transit gateway. Create a VPC attachment to each application VPC. Provide full mesh connectivity between all the VPCs by using the transit gateway.

B.Create VPC peering connections between the central shared services VPC and each application VPC in each business unit's AWS account.

C.Create VPC endpoint services powered by AWS PrivateLink in the central shared services VPCreate VPC endpoints in each application VPC.

D.Create a central transit VPC with a VPN appliance from AWS Marketplace. Create a VPN attachment from each VPC to the transit VPC. Provide full mesh connectivity among all the VPCs.

Answer: C

Explanation:

The most secure and scalable solution is option C, using VPC endpoint services (powered by AWS PrivateLink). Here's why:

Security: PrivateLink establishes private connectivity between VPCs without exposing traffic to the public internet. This eliminates the need for internet gateways, NAT devices, or public IPs, significantly reducing the attack surface. VPC endpoints are accessed through private IP addresses within your VPC.

Granular Access Control: VPC endpoint services provide granular control over which services and resources within the central shared services VPC are accessible to each application VPC. This is achieved through security groups and endpoint policies, allowing you to restrict access to specific resources.

Scalability: AWS PrivateLink is highly scalable and can easily accommodate new business units and application VPCs as the company grows. Adding a new application VPC only involves creating a new VPC endpoint, without disrupting existing connections.

Simplified Network Management: PrivateLink simplifies network management by eliminating the need for complex routing configurations or VPN connections. This reduces operational overhead and potential for misconfigurations.

Let's look at why the other options are less suitable:

A (Transit Gateway): While Transit Gateway provides centralized connectivity, full mesh connectivity increases the potential attack surface. It also requires more complex route table management and doesn't offer the same level of granular security controls as PrivateLink.

B (VPC Peering): VPC peering creates direct connections between VPCs. For a large number of VPCs, this can become difficult to manage due to the number of peerings required (n*(n-1)/2, where n is the number of VPCs). Peering connections do not inherently offer the granular security control of PrivateLink.

D (Transit VPC with VPN): This solution is more complex, requires managing VPN appliances, and introduces potential performance bottlenecks. It also relies on VPN connections over the public internet (or Direct Connect) and is less secure than PrivateLink.

In summary, AWS PrivateLink provides the most secure, scalable, and manageable solution for connecting application VPCs to a central shared services VPC, with granular control over access to resources.

Relevant Documentation:

AWS PrivateLink:https://aws.amazon.com/privatelink/
VPC Endpoints:https://docs.aws.amazon.com/vpc/latest/privatelink/vpc-endpoints.html

Question: 5

A company uses a 4 Gbps AWS Direct Connect dedicated connection with a link aggregation group (LAG) bundle to connect to five VPCs that are deployed in the us-east-1 Region. Each VPC serves a different business unit and uses its own private VIF for connectivity to the on-premises environment. Users are reporting slowness when they access resources that are hosted on AWS.

A network engineer finds that there are sudden increases in throughput and that the Direct Connect connection becomes saturated at the same time for about an hour each business day. The company wants to know which business unit is causing the sudden increase in throughput. The network engineer must find out this information and implement a solution to resolve the problem.
Which solution will meet these requirements?

A.Review the Amazon CloudWatch metrics for VirtualInterfaceBpsEgress and VirtualInterfaceBpsIngress to determine which VIF is sending the highest throughput during the period in which slowness is observed. Create a new 10 Gbps dedicated connection. Shift traffic from the existing dedicated connection to the new dedicated connection.

B.Review the Amazon CloudWatch metrics for VirtualInterfaceBpsEgress and VirtualInterfaceBpsIngress to determine which VIF is sending the highest throughput during the period in which slowness is observed. Upgrade the bandwidth of the existing dedicated connection to 10 Gbps.

C.Review the Amazon CloudWatch metrics for ConnectionBpsIngress and ConnectionPpsEgress to determine which VIF is sending the highest throughput during the period in which slowness is observed. Upgrade the existing dedicated connection to a 5 Gbps hosted connection.

D.Review the Amazon CloudWatch metrics for ConnectionBpsIngress and ConnectionPpsEgress to determine which VIF is sending the highest throughput during the period in which slowness is observed. Create a new 10 Gbps dedicated connection. Shift traffic from the existing dedicated connection to the new dedicated connection.

Answer: A

Explanation:

The correct answer is A. Here's why:

Identify the Problem: The question states that the 4 Gbps Direct Connect connection is getting saturated, causing performance issues. The company wants to identify the business unit responsible for the throughput spikes.

Monitoring Direct Connect: To pinpoint the source of the high throughput, we need to monitor the traffic on each private Virtual Interface (VIF) associated with each VPC.

CloudWatch Metrics: Amazon CloudWatch provides metrics for monitoring AWS resources. VirtualInterfaceBpsEgress and VirtualInterfaceBpsIngress metrics show the bits per second sent and received on each VIF, respectively. By analyzing these metrics during the period of slowness, the network engineer can determine which VIF (and therefore which business unit's VPC) is consuming the most bandwidth.

Addressing the Saturation: After identifying the culprit VIF, creating a new 10 Gbps dedicated connection and shifting traffic alleviates the congestion by providing more bandwidth. This avoids upgrading the existing connection, which can be disruptive.

Why other options are incorrect:

B: Upgrading the bandwidth of the existing connection will work, but migrating traffic to a separate connection is a much cleaner approach for large traffic spikes.
C and D: ConnectionBpsIngress and ConnectionPpsEgress metrics provide information about the overall connection, not individual VIFs. This will not help in identifying the problematic business unit. Additionally, upgrading to a hosted connection (Option C) is not correct. Hosted connections typically have lower bandwidth options compared to dedicated connections.https://docs.aws.amazon.com/directconnect/latest/UserGuide/monitoring-cloudwatch.html

Question: 6

A software-as-a-service (SaaS) provider hosts its solution on Amazon EC2 instances within a VPC in the AWS Cloud. All of the provider's customers also have their environments in the AWS Cloud.

A recent design meeting revealed that the customers have IP address overlap with the provider's AWS deployment. The customers have stated that they will not share their internal IP addresses and that they do not want to connect to the provider's SaaS service over the internet.
Which combination of steps is part of a solution that meets these requirements? (Choose two.)

A.Deploy the SaaS service endpoint behind a Network Load Balancer.

B.Configure an endpoint service, and grant the customers permission to create a connection to the endpoint service.

C.Deploy the SaaS service endpoint behind an Application Load Balancer.

D.Configure a VPC peering connection to the customer VPCs. Route traffic through NAT gateways.

E.Deploy an AWS Transit Gateway, and connect the SaaS VPC to it. Share the transit gateway with the customers. Configure routing on the transit gateway.

Answer: AB

Explanation:

The chosen solution (A and B) leverages AWS PrivateLink to securely connect the SaaS provider's service to its customers' VPCs, even with overlapping IP addresses, without exposing traffic to the public internet.

A. Deploy the SaaS service endpoint behind a Network Load Balancer (NLB): NLBs are crucial for PrivateLink endpoints. The NLB serves as the front end for the SaaS service. It distributes incoming traffic from the customers' VPCs to the backend EC2 instances hosting the SaaS application. NLBs operate at Layer 4, making them suitable for handling TCP and UDP traffic efficiently.

B. Configure an endpoint service, and grant the customers permission to create a connection to the endpoint service: The endpoint service is the core of the PrivateLink setup. The SaaS provider creates an endpoint service in their VPC, associated with the NLB. The endpoint service enables customers to create VPC endpoints in their own VPCs, allowing them to connect privately to the SaaS application. The SaaS provider controls which customers can access the service by granting permissions on the endpoint service. This eliminates the need for VPC peering or NAT gateways.

Let's analyze why the other options are less suitable:

C. Deploy the SaaS service endpoint behind an Application Load Balancer (ALB): ALBs are compatible with PrivateLink, but NLBs are generally preferred for non-HTTP(S) workloads or when direct TCP/UDP traffic is required. The question doesn't specify HTTP(S) requirements, making NLB a reasonable default.

D. Configure a VPC peering connection to the customer VPCs. Route traffic through NAT gateways: VPC peering would not work due to overlapping IP addresses, which is the core problem. NAT gateways would only provide internet access, which is explicitly forbidden by the customers. VPC peering also requires extensive routing configurations across multiple VPCs.

E. Deploy an AWS Transit Gateway, and connect the SaaS VPC to it. Share the transit gateway with the customers. Configure routing on the transit gateway: While Transit Gateway provides centralized connectivity, it doesn't solve the IP address overlap problem directly. Sharing a TGW also grants the customers broader access than needed; they only need access to the specific SaaS service. It does not handle the underlying overlap in IP address space. Using Transit Gateway might also incur unnecessary costs and complexity compared to PrivateLink for this specific use case.

Therefore, using a Network Load Balancer with an Endpoint Service provides a direct, secure, and isolated connection between the SaaS provider and its customers, resolving the IP address conflict and meeting the customers' requirements for private connectivity.

Authoritative Links:

AWS PrivateLink: https://aws.amazon.com/privatelink/
AWS Network Load Balancer: https://aws.amazon.com/elasticloadbalancing/network-load-balancer/

Question: 7

A network engineer is designing the architecture for a healthcare company's workload that is moving to the AWS Cloud. All data to and from the on-premises environment must be encrypted in transit. All traffic also must be inspected in the cloud before the traffic is allowed to leave the cloud and travel to the on-premises environment or to the internet.
The company will expose components of the workload to the internet so that patients can reserve appointments.
The architecture must secure these components and protect them against DDoS attacks. The architecture also must provide protection against financial liability for services that scale out during a DDoS event.
Which combination of steps should the network engineer take to meet all these requirements for the workload?
(Choose three.)

A.Use Traffic Mirroring to copy all traffic to a fleet of traffic capture appliances.

B.Set up AWS WAF on all network components.

C.Configure an AWS Lambda function to create Deny rules in security groups to block malicious IP addresses.

D.Use AWS Direct Connect with MACsec support for connectivity to the cloud.

E.Use Gateway Load Balancers to insert third-party firewalls for inline traffic inspection.

F.Configure AWS Shield Advanced and ensure that it is configured on all public assets.

Answer: DEF

Explanation:

The correct answer is DEF. Let's break down why each choice is either correct or incorrect:

D. Use AWS Direct Connect with MACsec support for connectivity to the cloud: This ensures that all data transmitted between the on-premises environment and AWS is encrypted in transit. MACsec provides layer 2 encryption for Direct Connect connections, fulfilling the requirement for encrypted communication between the on-premises and AWS environments. This choice directly addresses the requirement for encryption in transit. https://aws.amazon.com/directconnect/

E. Use Gateway Load Balancers to insert third-party firewalls for inline traffic inspection: Gateway Load Balancers (GWLB) allow you to insert third-party virtual appliances, such as firewalls, intrusion detection and prevention systems, and deep packet inspection systems, inline into the network traffic flow. This fulfills the requirement for traffic inspection before traffic leaves the cloud. A GWLB distributes traffic to a fleet of these appliances, providing scalability and high availability. https://aws.amazon.com/gateway-load-balancer/

F. Configure AWS Shield Advanced and ensure that it is configured on all public assets: AWS Shield Advanced provides enhanced DDoS protection, including protection against financial liability for scaling out during a DDoS attack. This directly addresses the requirement to protect against DDoS attacks and limit financial exposure. It automatically detects and mitigates sophisticated DDoS attacks targeted at your AWS resources. https://aws.amazon.com/shield/

Now, let's address why the other options are less suitable:

A. Use Traffic Mirroring to copy all traffic to a fleet of traffic capture appliances: Traffic mirroring is useful for monitoring and analyzing network traffic, but it does not provide inline inspection or prevent malicious traffic. It's a passive observation tool, not an active security control. It would not directly help with the inspection requirement.

B. Set up AWS WAF on all network components: While AWS WAF is an important security tool for protecting web applications from common web exploits, it doesn't provide comprehensive traffic inspection for all traffic, particularly traffic not destined for web applications. Also, placing it on "all network components" is vague and possibly inefficient; it's more targeted to application entry points. WAF helps protecting web applications and APIs.

C. Configure an AWS Lambda function to create Deny rules in security groups to block malicious IP addresses: This approach is not scalable or efficient for dealing with large-scale DDoS attacks. Security groups have limitations, and Lambda functions may not be responsive enough to mitigate rapidly evolving attacks. Also, manually managing security group rules with Lambda for DDoS mitigation is not a recommended practice. AWS Shield Advanced provides a more robust and automated solution.

Question: 8

A retail company is running its service on AWS. The company's architecture includes Application Load Balancers (ALBs) in public subnets. The ALB target groups are configured to send traffic to backend Amazon EC2 instances in private subnets. These backend EC2 instances can call externally hosted services over the internet by using a NAT gateway.
The company has noticed in its billing that NAT gateway usage has increased significantly. A network engineer needs to find out the source of this increased usage.
Which options can the network engineer use to investigate the traffic through the NAT gateway? (Choose two.)

A.Enable VPC flow logs on the NAT gateway's elastic network interface. Publish the logs to a log group in Amazon CloudWatch Logs. Use CloudWatch Logs Insights to query and analyze the logs.

B.Enable NAT gateway access logs. Publish the logs to a log group in Amazon CloudWatch Logs. Use CloudWatch Logs Insights to query and analyze the logs.

C.Configure Traffic Mirroring on the NAT gateway's elastic network interface. Send the traffic to an additional EC2 instance. Use tools such as tcpdump and Wireshark to query and analyze the mirrored traffic.

D.Enable VPC flow logs on the NAT gateway's elastic network interface. Publish the logs to an Amazon S3 bucket. Create a custom table for the S3 bucket in Amazon Athena to describe the log structure. Use Athena to query and analyze the logs.

E.Enable NAT gateway access logs. Publish the logs to an Amazon S3 bucket. Create a custom table for the S3 bucket in Amazon Athena to describe the log structure. Use Athena to query and analyze the logs.

Answer: AD

Explanation:

The correct answer is AD. Here's why:

A. Enable VPC flow logs on the NAT gateway's elastic network interface. Publish the logs to a log group in Amazon CloudWatch Logs. Use CloudWatch Logs Insights to query and analyze the logs.

VPC Flow Logs capture information about the IP traffic going to, from, and within your VPC. By enabling VPC Flow Logs on the NAT gateway's Elastic Network Interface (ENI), you capture detailed information about all traffic passing through the NAT gateway. This includes the source and destination IP addresses, ports, the number of bytes, and the action taken (ACCEPT or REJECT). CloudWatch Logs Insights provides a powerful query language to analyze these logs. You can quickly identify the EC2 instances generating the most traffic, the destination services they are accessing, and the total data transfer volume. This allows you to pinpoint the source of increased NAT gateway usage.

D. Enable VPC flow logs on the NAT gateway's elastic network interface. Publish the logs to an Amazon S3 bucket. Create a custom table for the S3 bucket in Amazon Athena to describe the log structure. Use Athena to query and analyze the logs.

This option also uses VPC Flow Logs, providing the same granular traffic information as option A. Instead of CloudWatch Logs Insights, it utilizes Amazon S3 for storage and Amazon Athena for querying. Storing the flow logs in S3 provides a cost-effective solution for long-term storage and enables you to use other AWS services for further analysis. Athena allows you to run SQL queries on the data stored in S3, making it easy to analyze large datasets and identify patterns in the NAT gateway traffic. This is beneficial if you need to perform complex analysis or retain the logs for extended periods.

Why the other options are incorrect:

B & E. NAT gateway access logs: NAT Gateway access logs do not exist. VPC Flow Logs are the proper method for logging traffic that passes through the NAT gateway.

C. Traffic Mirroring: While Traffic Mirroring allows you to capture network traffic, it's less suitable for this scenario due to its complexity and performance implications. Setting up Traffic Mirroring involves configuring a traffic mirror source (NAT gateway ENI), a traffic mirror target (EC2 instance), and traffic mirror filters. The mirrored traffic duplicates all packets, which could impact the performance of the monitoring EC2 instance.

Also, analyzing the captured packets using tools like tcpdump or Wireshark can be time-consuming and requires more manual effort compared to analyzing VPC Flow Logs with CloudWatch Logs Insights or Athena.

Additionally, traffic mirroring is generally used for security and deep-packet inspection, not basic cost analysis, making it overkill for this scenario.

Authoritative links:

VPC Flow Logs: https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html
CloudWatch Logs Insights: https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AnalyzingLogData.html
Amazon Athena: https://aws.amazon.com/athena/
NAT Gateway: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html Traffic Mirroring: https://docs.aws.amazon.com/vpc/latest/mirroring/what-is-traffic-mirroring.html

Question: 9

A banking company is successfully operating its public mobile banking stack on AWS. The mobile banking stack is deployed in a VPC that includes private subnets and public subnets. The company is using IPv4 networking and has not deployed or supported IPv6 in the environment. The company has decided to adopt a third-party service provider's API and must integrate the API with the existing environment. The service provider's API requires the use of IPv6.
A network engineer must turn on IPv6 connectivity for the existing workload that is deployed in a private subnet. The company does not want to permit IPv6 traffic from the public internet and mandates that the company's servers must initiate all IPv6 connectivity. The network engineer turns on IPv6 in the VPC and in the private subnets.
Which solution will meet these requirements?

A.Create an internet gateway and a NAT gateway in the VPC. Add a route to the existing subnet route tables to point IPv6 traffic to the NAT gateway.

B.Create an internet gateway and a NAT instance in the VPC. Add a route to the existing subnet route tables to point IPv6 traffic to the NAT instance.

C.Create an egress-only Internet gateway in the VPAdd a route to the existing subnet route tables to point IPv6 traffic to the egress-only internet gateway.

D.Create an egress-only internet gateway in the VPC. Configure a security group that denies all inbound traffic. Associate the security group with the egress-only internet gateway.

Answer: C

Explanation:

The correct answer is C. Here's why:

The core requirement is to enable IPv6 connectivity from private subnets to an external IPv6-only API, ensuring no inbound IPv6 traffic from the public internet is allowed.

Egress-Only Internet Gateway (EIGW): An EIGW is designed precisely for this scenario. It allows instances in private subnets to initiate outbound IPv6 connections to the internet but prevents the internet from initiating IPv6 connections to those instances. This aligns with the requirement that the company's servers initiate all

IPv6 connectivity.

Route Table Configuration: By adding a route to the subnet route table that directs IPv6 traffic (::/0) to the EIGW, any IPv6 traffic originating from the instances in the subnet will be routed through the EIGW to the internet.

Why other options are incorrect:

A & B (IGW and NAT Gateway/Instance): These solutions are primarily for IPv4. While NAT Gateways support IPv4 NAT, they don't inherently handle IPv6 connectivity. IGW enables inbound and outbound internet communication, violating the requirement to block inbound internet IPv6 traffic. Also, NAT gateway does not support IPv6.

D (EIGW and Security Group): While using a security group to deny inbound traffic is good practice, associating a security group with an EIGW is not how EIGWs function. Security groups are associated with network interfaces of EC2 instances, not with gateways. The EIGW inherently blocks inbound traffic.

In summary: The EIGW provides the necessary outbound-only IPv6 connectivity for private subnets, fulfilling the company's requirement to integrate with the third-party IPv6 API while preventing inbound IPv6 traffic from the internet.

Supporting Links:

Egress-Only Internet Gateway:https://docs.aws.amazon.com/vpc/latest/userguide/egress-only-internet-gateway.html
Route Tables:https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Route_Tables.html

Question: 10

A company has deployed an AWS Network Firewall firewall into a VPC. A network engineer needs to implement a solution to deliver Network Firewall flow logs to the company's Amazon OpenSearch Service (Amazon Elasticsearch Service) cluster in the shortest possible time.
Which solution will meet these requirements?

A.Create an Amazon S3 bucket. Create an AWS Lambda function to load logs into the Amazon OpenSearch Service (Amazon Elasticsearch Service) cluster. Enable Amazon Simple Notification Service (Amazon SNS) notifications on the S3 bucket to invoke the Lambda function. Configure flow logs for the firewall. Set the S3 bucket as the destination.

B.Create an Amazon Kinesis Data Firehose delivery stream that includes the Amazon OpenSearch Service (Amazon Elasticsearch Service) cluster as the destination. Configure flow logs for the firewall Set the Kinesis Data Firehose delivery stream as the destination for the Network Firewall flow logs.

C.Configure flow logs for the firewall. Set the Amazon OpenSearch Service (Amazon Elasticsearch Service) cluster as the destination for the Network Firewall flow logs.

D.Create an Amazon Kinesis data stream that includes the Amazon OpenSearch Service (Amazon Elasticsearch Service) cluster as the destination. Configure flow logs for the firewall. Set the Kinesis data stream as the destination for the Network Firewall flow logs.

Answer: B

Explanation:

The correct answer is B because it provides the most direct and efficient method for delivering Network Firewall flow logs to an Amazon OpenSearch Service cluster.

Here's a detailed justification:

Kinesis Data Firehose for Real-time Ingestion: Kinesis Data Firehose is designed specifically for reliably streaming data to destinations like Amazon OpenSearch Service. It provides buffering, transformation, and loading capabilities, optimizing the data flow for analysis and storage.https://aws.amazon.com/kinesis/data-

firehose/

Direct Integration: Firehose offers direct integration with Amazon OpenSearch Service as a destination, eliminating the need for intermediary steps or custom code for data ingestion. This simplifies the solution and minimizes latency.

Managed Service: Kinesis Data Firehose is a fully managed service. AWS handles the underlying infrastructure and scaling, reducing operational overhead.

Alternative A is Inefficient: Option A involves S3, Lambda, and SNS, which introduces unnecessary complexity and potential latency compared to the direct Firehose approach. The Lambda function requires custom code to load the logs into OpenSearch, adding development and maintenance burden.

Option C is Incorrect: Network Firewall flow logs cannot be directly sent to Amazon OpenSearch Service. A data streaming solution is needed.

Option D is Incorrect: Kinesis Data Streams require more manual management compared to Firehose, including provisioning and scaling consumers to process the data before loading it to OpenSearch. Data Streams offer real time but at the cost of increased management. For just moving logs, data firehose is better suited.

In summary, utilizing Kinesis Data Firehose offers the quickest and most straightforward path to delivering Network Firewall flow logs to Amazon OpenSearch Service. It eliminates the need for custom code and intermediary services, simplifying the process and reducing latency.

Question: 11

A company is using custom DNS servers that run BIND for name resolution in its VPCs. The VPCs are deployed across multiple AWS accounts that are part of the same organization in AWS Organizations. All the VPCs are connected to a transit gateway. The BIND servers are running in a central VPC and are configured to forward all queries for an on-premises DNS domain to DNS servers that are hosted in an on-premises data center. To ensure that all the VPCs use the custom DNS servers, a network engineer has configured a VPC DHCP options set in all the VPCs that specifies the custom DNS servers to be used as domain name servers.
Multiple development teams in the company want to use Amazon Elastic File System (Amazon EFS). A development team has created a new EFS file system but cannot mount the file system to one of its Amazon EC2 instances. The network engineer discovers that the EC2 instance cannot resolve the IP address for the EFS mount point fs-33444567d.efs.us-east-1.amazonaws.com. The network engineer needs to implement a solution so that development teams throughout the organization can mount EFS file systems.
Which combination of steps will meet these requirements? (Choose two.)

A.Configure the BIND DNS servers in the central VPC to forward queries for efs.us-east-1.amazonaws.com to the Amazon provided DNS server (169.254.169.253).

B.Create an Amazon Route 53 Resolver outbound endpoint in the central VPC. Update all the VPC DHCP options sets to use AmazonProvidedDNS for name resolution.

C.Create an Amazon Route 53 Resolver inbound endpoint in the central VPUpdate all the VPC DHCP options sets to use the Route 53 Resolver inbound endpoint in the central VPC for name resolution.

D.Create an Amazon Route 53 Resolver rule to forward queries for the on-premises domain to the on-premises DNS servers. Share the rule with the organization by using AWS Resource Access Manager (AWS RAM). Associate the rule with all the VPCs.

E.Create an Amazon Route 53 private hosted zone for the efs.us-east-1.amazonaws.com domain. Associate the private hosted zone with the VPC where the EC2 instance is deployed. Create an A record for fs-33444567d.efs.us-east-1.amazonaws.com in the private hosted zone. Configure the A record to return the mount target of the EFS mount point.

Answer: BD

Explanation:

Here's a detailed justification for why options B and D are the correct choices and why the other options are incorrect for resolving the EFS mounting issue across multiple AWS accounts using a transit gateway and custom DNS.

Understanding the Problem:

The core issue is that EC2 instances in multiple VPCs cannot resolve the EFS mount point (fs-33444567d.efs.us-east-1.amazonaws.com). This resolution needs to occur across multiple AWS accounts.

Currently, the VPCs use custom DNS servers (BIND) that forward on-premises DNS queries but aren't configured for resolving AWS service endpoints like EFS.

Why B and D are correct:

B. Create an Amazon Route 53 Resolver outbound endpoint in the central VPC. Update all the VPC DHCP options sets to use AmazonProvidedDNS for name resolution.

This approach directly addresses the inability to resolve AWS service endpoints. A Route 53 Resolver outbound endpoint allows you to forward DNS queries from your VPC to DNS servers outside the VPC (in this case, to Amazon's DNS servers).

By setting the DHCP options set to AmazonProvidedDNS, you ensure that all VPCs now use the Amazon-provided DNS, which can resolve AWS service endpoints like those for EFS. This eliminates the need for custom BIND configuration for these AWS-specific zones.

https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver.html

D. Create an Amazon Route 53 Resolver rule to forward queries for the on-premises domain to the on-premises DNS servers. Share the rule with the organization by using AWS Resource Access Manager (AWS RAM). Associate the rule with all the VPCs.

Since the company still needs the VPCs to resolve queries for the on-premise domain, we can use the Route 53 Resolver to forward those queries from VPCs to the on-premise DNS servers. Using AWS RAM to share the rules allows the rule to be used across different AWS accounts within the organization. This will ensure that on-premise DNS queries are still resolved after the VPCs DHCP options sets have been updated to use the AmazonProvidedDNS.

https://docs.aws.amazon.com/ram/latest/userguide/what-is.html

Why the other options are incorrect:

A. Configure the BIND DNS servers in the central VPC to forward queries for efs.us-east-1.amazonaws.com to the Amazon provided DNS server (169.254.169.253).

While technically feasible, this approach is less scalable and maintainable than using Route 53 Resolver. You'd have to manage BIND configuration, and it introduces a single point of failure in the central VPC. Plus, forwarding to 169.254.169.253 directly is generally discouraged; using AmazonProvidedDNS is the AWS-recommended method. Also, it won't address the on-premise DNS resolutions.

C. Create an Amazon Route 53 Resolver inbound endpoint in the central VPC. Update all the VPC DHCP options sets to use the Route 53 Resolver inbound endpoint in the central VPC for name resolution.

Route 53 Resolver inbound endpoints are used to forward DNS queries from on-premises networks to AWS. This is the opposite of what's needed here; you want AWS resources to resolve AWS service endpoints.

E. Create an Amazon Route 53 private hosted zone for the efs.us-east-1.amazonaws.com domain. Associate the private hosted zone with the VPC where the EC2 instance is deployed. Create an A record for fs-33444567d.efs.us-east-1.amazonaws.com in the private hosted zone. Configure the A record to return the mount target of the EFS mount point.

This approach only addresses the resolution issue for one VPC. It doesn't scale to the requirement that development teams throughout the organization be able to mount EFS file systems. Also, manually creating A records and keeping them in sync is complex and error-prone, given that EFS mount targets can change. The AmazonProvidedDNS automatically resolves these.

In summary, the combination of using AmazonProvidedDNS (option B) for resolving AWS service endpoints like EFS and using Route 53 Resolver rules for on-premises DNS resolution, which is shared across the organization using AWS RAM (option D), provides the most scalable, manageable, and robust solution.

Question: 12

An ecommerce company is hosting a web application on Amazon EC2 instances to handle continuously changing customer demand. The EC2 instances are part of an Auto Scaling group. The company wants to implement a solution to distribute traffic from customers to the EC2 instances. The company must encrypt all traffic at all stages between the customers and the application servers. No decryption at intermediate points is allowed. Which solution will meet these requirements?

A.Create an Application Load Balancer (ALB). Add an HTTPS listener to the ALB. Configure the Auto Scaling group to register instances with the ALB's target group.

B.Create an Amazon CloudFront distribution. Configure the distribution with a custom SSL/TLS certificate. Set the Auto Scaling group as the distribution's origin.

C.Create a Network Load Balancer (NLB). Add a TCP listener to the NLB. Configure the Auto Scaling group to register instances with the NLB's target group.

D.Create a Gateway Load Balancer (GLB). Configure the Auto Scaling group to register instances with the GLB's target group.

Answer: C

Explanation:

The requirement is end-to-end encryption between customers and EC2 instances without decryption at intermediate points. This means the load balancer cannot decrypt and re-encrypt the traffic. An Application Load Balancer (ALB) terminates SSL/TLS at the load balancer, thus decrypting the traffic. Therefore, option A is incorrect. Amazon CloudFront is a CDN and also terminates SSL/TLS; hence, it won't meet the end-to-end encryption requirement without decryption. Therefore, option B is incorrect. A Gateway Load Balancer (GLB) is designed for deploying and managing virtual appliances, typically for deep packet inspection, intrusion detection and prevention, and other network security functions. While it doesn't inherently decrypt traffic, it's not the ideal choice for simply load balancing encrypted traffic directly to instances. Therefore, option D is incorrect. A Network Load Balancer (NLB) operates at Layer 4 (TCP/UDP) and can forward encrypted traffic directly to the backend EC2 instances without decryption. This fulfills the end-to-end encryption requirement. The EC2 instances would need to handle SSL/TLS termination themselves. Therefore, option C correctly uses an NLB with a TCP listener to forward the encrypted traffic to the instances in the Auto Scaling group, ensuring no intermediate decryption.

Authoritative links:

Network Load Balancer
Application Load Balancer
Gateway Load Balancer

Question: 13

A company has two on-premises data center locations. There is a company-managed router at each data center.

Each data center has a dedicated AWS Direct Connect connection to a Direct Connect gateway through a private virtual interface. The router for the first location is advertising 110 routes to the Direct Connect gateway by using BGP, and the router for the second location is advertising 60 routes to the Direct Connect gateway by using BGP. The Direct Connect gateway is attached to a company VPC through a virtual private gateway.
A network engineer receives reports that resources in the VPC are not reachable from various locations in either data center. The network engineer checks the VPC route table and sees that the routes from the first data center location are not being populated into the route table. The network engineer must resolve this issue in the most operationally efficient manner.
What should the network engineer do to meet these requirements?

A.Remove the Direct Connect gateway, and create a new private virtual interface from each company router to the virtual private gateway of the VPC.

B.Change the router configurations to summarize the advertised routes.

C.Open a support ticket to increase the quota on advertised routes to the VPC route table.

D.Create an AWS Transit Gateway. Attach the transit gateway to the VPC, and connect the Direct Connect gateway to the transit gateway.

Answer: B

Explanation:

The correct answer is B: Change the router configurations to summarize the advertised routes. Here's why:

The core problem is that the VPC route table is not being populated with routes from one of the on-premises locations, specifically the first data center advertising 110 routes. This points to a limit on the number of routes that a VPC route table can accept from a Direct Connect gateway. While AWS doesn't publicly document hard limits for propagated routes from a DXGW to a VPC route table, exceeding a reasonable number of routes is a common issue.

Summarizing routes is the most operationally efficient way to solve this. By aggregating multiple smaller, specific routes into fewer, larger, more general routes, the total number of routes advertised to the Direct Connect gateway and subsequently propagated to the VPC route table is reduced. This allows the essential reachability information to be conveyed without exceeding any potential route limits. This solution addresses the problem at the source of the route advertisements (the on-premises routers) without requiring
infrastructure changes or significant service disruptions.

Option A is incorrect because removing the Direct Connect gateway and creating private virtual interfaces directly to the VPG is not best practice and removes the central management capability the Direct Connect Gateway provides. This would require reconfiguring BGP sessions on the VPC side and create more configuration and management overhead.

Option C, opening a support ticket, might seem plausible if a hard quota was definitively the cause. However, summarizing routes is a better practice and more controllable solution, and you don't have confirmation of an actual quota limit, so it is premature to rely on AWS increasing quotas. Furthermore, even if AWS granted an increased quota, the underlying problem of overly granular routing isn't addressed.

Option D, using Transit Gateway, is a valid architectural choice for more complex scenarios and could solve the immediate problem. However, it introduces additional complexity and cost. For this relatively simple scenario, the most operationally efficient solution is route summarization. TGW should be considered when you need more advanced routing policies, shared services VPC connectivity, or more dynamic scaling than DXGW + VPC VPG provides.

Route summarization offers a targeted solution that directly addresses the route limits without impacting other parts of the network and is the most straightforward approach in this situation.

For further research, consult the AWS Direct Connect documentation and networking best practices, though explicit VPC route limits related to DXGWs aren't clearly documented.

AWS Direct Connect:https://aws.amazon.com/directconnect/
AWS Transit Gateway:https://aws.amazon.com/transit-gateway/

Question: 14

A company has expanded its network to the AWS Cloud by using a hybrid architecture with multiple AWS accounts. The company has set up a shared AWS account for the connection to its on-premises data centers and the company offices. The workloads consist of private web-based services for internal use. These services run in different AWS accounts. Office-based employees consume these services by using a DNS name in an on-premises DNS zone that is named example.internal.
The process to register a new service that runs on AWS requires a manual and complicated change request to the internal DNS. The process involves many teams.
The company wants to update the DNS registration process by giving the service creators access that will allow them to register their DNS records. A network engineer must design a solution that will achieve this goal. The solution must maximize cost-effectiveness and must require the least possible number of configuration changes.
Which combination of steps should the network engineer take to meet these requirements? (Choose three.)

A.Create a record for each service in its local private hosted zone (serviceA.account1.aws.example.internal). Provide this DNS record to the employees who need access.

B.Create an Amazon Route 53 Resolver inbound endpoint in the shared account VPC. Create a conditional forwarder for a domain named aws.example.internal on the on-premises DNS servers. Set the forwarding IP addresses to the inbound endpoint's IP addresses that were created.

C.Create an Amazon Route 53 Resolver rule to forward any queries made to onprem.example.internal to the on-premises DNS servers.

D.Create an Amazon Route 53 private hosted zone named aws.example.internal in the shared AWS account to resolve queries for this domain.

E.Launch two Amazon EC2 instances in the shared AWS account. Install BIND on each instance. Create a DNS conditional forwarder on each BIND server to forward queries for each subdomain under aws.example.internal to the appropriate private hosted zone in each AWS account. Create a conditional forwarder for a domain named aws.example.internal on the on-premises DNS servers. Set the forwarding IP addresses to the IP addresses of the BIND servers.

F.Create a private hosted zone in the shared AWS account for each account that runs the service. Configure the private hosted zone to contain aws.example.internal in the domain (account1.aws.example.internal). Associate the private hosted zone with the VPC that runs the service and the shared account VPC.

Answer: BDF

Explanation:

The correct solution, **BDF**, efficiently addresses the company's DNS registration challenges by leveraging Route 53 Resolver capabilities. Here's why:

B. Create an Amazon Route 53 Resolver inbound endpoint...: An inbound endpoint acts as a gateway for on-premises DNS servers to forward queries to Route 53 Resolver in the AWS shared services account. Creating a conditional forwarder on the on-premises DNS server for aws.example.internal and directing it to the inbound endpoint ensures that all requests for AWS services are properly routed to AWS. This avoids direct exposure of AWS infrastructure IP addresses to internal DNS infrastructure. (Source: https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver.html)

D. Create an Amazon Route 53 private hosted zone named aws.example.internal...: This establishes a central location for managing DNS records related to AWS-hosted services within the shared account. It's the root of the delegated AWS DNS namespace. This enables more granular control and allows for further sub-delegation of authority. This is a crucial part of the solution which allows the use of private hosted zones to resolve the names of the internal services.

F. Create a private hosted zone in the shared AWS account for each account that runs the service...: By creating separate private hosted zones (e.g., account1.aws.example.internal) for each AWS account, the service

creators gain the authority to manage their own DNS records. Associating these hosted zones with both the service VPC and the shared account VPC enables DNS resolution from both locations, ensuring internal employees can access the services using their desired DNS names.

Why other options are incorrect:

A: Directly providing the service-specific DNS records to employees lacks scalability and doesn't streamline the DNS registration process. It also doesn't address centralized management.
C: Creating a Resolver rule to forward onprem.example.internal to on-premises is already the default behavior. **E:** Launching EC2 instances with BIND adds unnecessary complexity, operational overhead, and costs. Route 53 Resolver offers a managed and highly available DNS solution. This is a legacy implementation and it will introduce more configuration changes.

In essence, the BDF combination establishes a streamlined and cost-effective DNS delegation model. Service teams get authority over their DNS records under aws.example.internal through private hosted zones, which reduces the manual change requests. Route 53 Resolver, via an inbound endpoint, connects the on-premises network to the delegated AWS DNS namespace. This provides the employees with access to services hosted on AWS.

Question: 15

A company has multiple AWS accounts. Each account contains one or more VPCs. A new security guideline requires the inspection of all traffic between VPCs.
The company has deployed a transit gateway that provides connectivity between all VPCs. The company also has deployed a shared services VPC with Amazon EC2 instances that include IDS services for stateful inspection. The EC2 instances are deployed across three Availability Zones. The company has set up VPC associations and routing on the transit gateway. The company has migrated a few test VPCs to the new solution for traffic inspection. Soon after the configuration of routing, the company receives reports of intermittent connections for traffic that crosses Availability Zones.
What should a network engineer do to resolve this issue?

A.Modify the transit gateway VPC attachment on the shared services VPC by enabling cross-Availability Zone load balancing.

B.Modify the transit gateway VPC attachment on the shared services VPC by enabling appliance mode support.

C.Modify the transit gateway by selecting VPN equal-cost multi-path (ECMP) routing support.

D.Modify the transit gateway by selecting multicast support.

Answer: B

Explanation:

The issue described points to asymmetric routing problems occurring due to the stateful nature of the IDS appliances in the shared services VPC. When traffic crosses Availability Zones, the return traffic might not be routed back to the same IDS instance that initially processed the outbound traffic, leading to connection failures because the stateful firewall in the IDS appliance doesn't recognize the return traffic as part of an established connection.

Option B, enabling appliance mode support on the transit gateway VPC attachment for the shared services VPC, addresses this asymmetric routing problem. Appliance mode ensures that traffic flows are symmetrical, meaning that traffic originating from a source will always return through the same appliance. This resolves the issue of return traffic going to a different IDS instance, which would cause the stateful inspection to fail. By enabling appliance mode, the transit gateway ensures that both forward and return traffic for a connection are routed through the same IDS appliance, maintaining connection state and preventing intermittent connection issues.

Option A, enabling cross-Availability Zone load balancing on the transit gateway VPC attachment is incorrect. Cross-AZ load balancing distributes traffic across instances in different AZs, but doesn't guarantee symmetric routing, which is required for stateful inspection.

Option C, enabling VPN ECMP routing, is not relevant in this scenario. ECMP is for distributing traffic across multiple VPN connections, not for managing traffic flow through IDS appliances in a shared services VPC.

Option D, enabling multicast support, is also not relevant as multicast is used for one-to-many communication, which isn't the problem here. The issue is with the routing of unicast traffic and stateful inspection.

Therefore, enabling appliance mode support ensures consistent routing and allows the stateful IDS appliances to properly function, resolving the intermittent connection issues.

Reference link: https://docs.aws.amazon.com/vpc/latest/tgw/tgw-appliance-mode.html

Question: 16

A company is using a NAT gateway to allow internet connectivity for private subnets in a VPC in the us-west-2 Region. After a security audit, the company needs to remove the NAT gateway.
In the private subnets, the company has resources that use the unified Amazon CloudWatch agent. A network engineer must create a solution to ensure that the unified CloudWatch agent continues to work after the removal of the NAT gateway.
Which combination of steps should the network engineer take to meet these requirements? (Choose three.)

A.Validate that private DNS is enabled on the VPC by setting the enableDnsHostnames VPC attribute and the enableDnsSupport VPC attribute to true.

B.Create a new security group with an entry to allow outbound traffic that uses the TCP protocol on port 443 to destination 0.0.0.0/0

C.Create a new security group with entries to allow inbound traffic that uses the TCP protocol on port 443 from the IP prefixes of the private subnets.

D.Create the following interface VPC endpoints in the VPC: com.amazonaws.us-west-2.logs and com.amazonaws.us-west-2.monitoring. Associate the new security group with the endpoint network interfaces.

E.Create the following interface VPC endpoint in the VPC: com.amazonaws.us-west-2.cloudwatch. Associate the new security group with the endpoint network interfaces.

F.Associate the VPC endpoint or endpoints with route tables that the private subnets use.

Answer: ACD

Explanation:

The correct combination of steps is **ACD**. Here's why:

A. Validate that private DNS is enabled on the VPC: CloudWatch agent communication relies on resolving DNS names for the AWS services. enableDnsHostnames and enableDnsSupport must be enabled at the VPC level to allow instances within the VPC to resolve AWS service endpoints through the Amazon DNS server. This allows the CloudWatch agent to find the proper endpoints even without direct internet access. Without this, the agent will fail to resolve the CloudWatch endpoint, and metrics will not be sent.
https://docs.aws.amazon.com/vpc/latest/userguide/vpc-dns.html

C. Create a new security group with entries to allow inbound traffic that uses the TCP protocol on port 443 from the IP prefixes of the private subnets: While not strictly necessary for the CloudWatch agent to send metrics, VPC endpoints (created in the next step) require a security group. The endpoint's security group needs to allow traffic from the resources using the endpoint. The most secure approach is to limit traffic only from the IP ranges of the private subnets. This ensures only resources within the private subnets can access the CloudWatch endpoints. This is not an outbound rule on the instances. It is an inbound rule on the VPC

endpoint.

D. Create the following interface VPC endpoints in the VPC: com.amazonaws.us-west-2.logs and com.amazonaws.us-west-2.monitoring. Associate the new security group with the endpoint network interfaces: Interface VPC endpoints provide private connectivity to AWS services without requiring internet gateways, NAT devices, or VPN connections. Creating endpoints for com.amazonaws.us-west-2.logs and com.amazonaws.us-west-2.monitoring allows the CloudWatch agent to securely send logs and metrics directly to CloudWatch over the AWS network. Associating the security group (created in step C) with the endpoint network interfaces controls the traffic flow to the endpoint. These are the correct service names.
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch-agent-vpc-endpoints.html

Why other options are incorrect:

B: Creating a new security group with an outbound rule to 0.0.0.0/0:443 would be applicable if the instances were still using the NAT Gateway. Since the goal is to remove the NAT gateway, this option is incorrect.
E:com.amazonaws.us-west-2.cloudwatch is not the correct endpoint. The required endpoints are com.amazonaws.us-west-2.logs and com.amazonaws.us-west-2.monitoring.
F: The VPC endpoints are associated with the subnets via route tables during the endpoint creation. There isn't a separate "associate with route tables" step that the engineer performs after endpoint creation. The wizard in the console or the CLI command will ask for which route table the endpoint will affect.

Question: 17

An international company provides early warning about tsunamis. The company plans to use IoT devices to monitor sea waves around the world. The data that is collected by the IoT devices must reach the company's infrastructure on AWS as quickly as possible. The company is using three operation centers around the world. Each operation center is connected to AWS through Its own AWS Direct Connect connection. Each operation center is connected to the internet through at least two upstream internet service providers.
The company has its own provider-independent (PI) address space. The IoT devices use TCP protocols for reliable transmission of the data they collect. The IoT devices have both landline and mobile internet connectivity. The infrastructure and the solution will be deployed in multiple AWS Regions. The company will use Amazon Route 53 for DNS services.
A network engineer needs to design connectivity between the IoT devices and the services that run in the AWS Cloud.
Which solution will meet these requirements with the HIGHEST availability?

A.Set up an Amazon CloudFront distribution with origin failover. Create an origin group for each Region where the solution is deployed.

B.Set up Route 53 latency-based routing. Add latency alias records. For the latency alias records, set the value of Evaluate Target Health to Yes.

C.Set up an accelerator in AWS Global Accelerator. Configure Regional endpoint groups and health checks.

D.Set up Bring Your Own IP (BYOIP) addresses. Use the same PI addresses for each Region where the solution is deployed.

Answer: C

Explanation:

The correct answer is **C: Set up an accelerator in AWS Global Accelerator. Configure Regional endpoint groups and health checks.**

Here's a detailed justification:

AWS Global Accelerator is specifically designed to improve the availability and performance of applications for global users. It achieves this by directing traffic to the optimal endpoint based on network conditions and endpoint health. In this scenario, the company has IoT devices scattered globally and wants the data to reach

its AWS infrastructure as quickly and reliably as possible across multiple regions.

Global Accelerator uses the AWS global network to optimize the path from the IoT devices to the nearest healthy endpoint in one of the company's AWS Regions. It accomplishes this via static anycast IPs that serve as the entry point for traffic. Because the company has operation centers connected via Direct Connect in different Regions, Global Accelerator can direct traffic to the closest operation center with healthy endpoints, even if one Region or Direct Connect link experiences issues. Endpoint groups are configured for each Region, allowing traffic to be dynamically routed to the healthy Region. Health checks are used to monitor the health of the application endpoints within each Region.

Option A, Amazon CloudFront with origin failover, is primarily designed for caching static and dynamic content closer to users for improved performance and is less suited for TCP-based real-time data ingestion from IoT devices. It's better suited for distributing content than reliably accepting and routing data.

Option B, Route 53 latency-based routing, makes routing decisions based on latency. While it's useful for routing users to the Region with the lowest latency, it doesn't provide the same level of fault tolerance and global traffic management as Global Accelerator, particularly in the presence of multiple network paths (landline and mobile). More over, R53 is a DNS service, which relies on resolution and propagation.

Option D, Bring Your Own IP (BYOIP) addresses, allows the company to use its own IP address range with AWS. While useful for IP address management and reputation, it doesn't directly address the requirement for high availability and optimized routing to the nearest healthy endpoint. BYOIP simply lets you use your own IP addresses in AWS, it doesn't inherently improve routing or availability. It would need to be combined with a routing solution, but Global Accelerator is a more direct and effective solution.Global Accelerator provides both a stable entry point and global routing, maximizing availability and minimizing latency.**Authoritative links:**

AWS Global Accelerator: https://aws.amazon.com/global-accelerator/
Route 53 Routing Policies: https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-policy.html Amazon CloudFront Origin Failover:
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/high_availability_origin_failover.html AWS BYOIP: https://aws.amazon.com/byoip/

Question: 18

A company is planning a migration of its critical workloads from an on-premises data center to Amazon EC2 instances. The plan includes a new 10 Gbps AWS Direct Connect dedicated connection from the on-premises data center to a VPC that is attached to a transit gateway. The migration must occur over encrypted paths between the on-premises data center and the AWS Cloud.
Which solution will meet these requirements while providing the HIGHEST throughput?

A.Configure a public VIF on the Direct Connect connection. Configure an AWS Site-to-Site VPN connection to the transit gateway as a VPN attachment.

B.Configure a transit VIF on the Direct Connect connection. Configure an IPsec VPN connection to an EC2 instance that is running third-party VPN software.

C.Configure MACsec for the Direct Connect connection. Configure a transit VIF to a Direct Connect gateway that is associated with the transit gateway.

D.Configure a public VIF on the Direct Connect connection. Configure two AWS Site-to-Site VPN connections to the transit gateway. Enable equal-cost multi-path (ECMP) routing.

Answer: C

Explanation:

The correct answer is C because it provides a high-throughput, encrypted connection between the on-premises data center and AWS, specifically tailored for the given scenario.

Here's a detailed justification:

The core requirement is encrypted traffic over a high-bandwidth connection between on-premises and AWS. MACsec provides hardware-based encryption at the data link layer (Layer 2) and is specifically designed for Direct Connect connections. This encryption method minimizes overhead and provides significantly higher throughput compared to IPsec VPNs, which operate at Layer 3.

Configuring a transit VIF is essential for connecting the Direct Connect connection to the transit gateway. The transit gateway acts as a central hub, allowing the on-premises network to connect to multiple VPCs within AWS through the Direct Connect connection. Associating the Direct Connect gateway with the transit gateway allows the propagation of routes between your on-premises network and the AWS environment, routed through the Transit Gateway.

Option A uses a public VIF and Site-to-Site VPN. While Site-to-Site VPN provides encryption, it introduces significant overhead due to IPsec encapsulation and software processing. This limits throughput and would not maximize the 10 Gbps Direct Connect connection. A public VIF isn't directly routed to the transit gateway but over public endpoints which defeats the purpose of using direct connect.

Option B proposes an IPsec VPN to an EC2 instance. This approach incurs considerable overhead, similar to Option A, due to IPsec and the limitations of the EC2 instance's processing capacity. Additionally, managing and scaling a VPN solution on EC2 instances is complex and may introduce availability concerns. The transit VIF to a transit gateway provides a more seamless and managed approach.

Option D tries to improve throughput using ECMP with two Site-to-Site VPN connections. While ECMP can increase aggregate bandwidth, the inherent overhead of IPsec remains, and it doesn't leverage the full potential of the Direct Connect connection and isn't as efficient as MACsec. Furthermore, managing two VPN connections adds operational complexity. A public VIF isn't directly routed to the transit gateway but over public endpoints which defeats the purpose of using direct connect.

In summary, MACsec on Direct Connect with a transit VIF provides the optimal balance of encryption and high throughput, directly addressing the stated requirements for migrating critical workloads.

Relevant documentation:

AWS Direct Connect MACsec: https://aws.amazon.com/directconnect/macsec/
AWS Transit Gateway: https://aws.amazon.com/transit-gateway/
AWS Direct Connect Gateways: https://docs.aws.amazon.com/directconnect/latest/UserGuide/direct-connect-gateways-intro.html

Question: 19

A network engineer must develop an AWS CloudFormation template that can create a virtual private gateway, a customer gateway, a VPN connection, and static routes in a route table. During testing of the template, the network engineer notes that the CloudFormation template has encountered an error and is rolling back. What should the network engineer do to resolve the error?

A.Change the order of resource creation in the CloudFormation template.

B.Add the DependsOn attribute to the resource declaration for the virtual private gateway. Specify the route table entry resource.

C.Add a wait condition in the template to wait for the creation of the virtual private gateway.

D.Add the DependsOn attribute to the resource declaration for the route table entry. Specify the virtual private gateway resource.

Answer: D

Explanation:

The CloudFormation template is failing because the route table entry (static route) is likely being created before the virtual private gateway (VGW) is fully provisioned. The route entry creation depends on the VGW being available. CloudFormation attempts to create resources in parallel, and without explicit dependencies, it might try to create the route before the VGW is ready, leading to an error and rollback.

Option D correctly addresses this dependency issue. By adding the DependsOn attribute to the route table entry resource declaration and specifying the VGW resource, we explicitly tell CloudFormation to create the VGW first. CloudFormation will then wait until the VGW is successfully created before proceeding to create the route table entry that depends on it. This enforces the correct order of operations and resolves the dependency conflict.

Option A is incorrect because simply changing the order of resource declarations in the template doesn't guarantee the order of resource creation. CloudFormation can still attempt to create resources in parallel unless explicit dependencies are defined.

Option B is incorrect because the route table entry depends on the VGW, not the other way around. Specifying the route table entry resource in the DependsOn attribute of the VGW would create a circular dependency, which is not what we need. The route table entry needs to know about the VGW, not the VGW needing to know about the route table entry.

Option C, adding a wait condition, is generally not recommended for dependencies between AWS resources created within the same CloudFormation stack. DependsOn is a cleaner and more appropriate solution for expressing resource dependencies. Wait conditions are better suited for situations where you need to wait for an external process or service to complete before proceeding.

Therefore, the most effective solution is to explicitly define the dependency between the route table entry and the VGW using the DependsOn attribute.

Refer to the following AWS documentation for further details on DependsOn attribute and CloudFormation resource dependencies:

AWS CloudFormation DependsOn Attribute
AWS CloudFormation Resource Creation Options (for understanding wait conditions and why they are not the ideal choice here)

Question: 20

A company operates its IT services through a multi-site hybrid infrastructure. The company deploys resources on AWS in the us-east-1 Region and in the eu-west-2 Region. The company also deploys resources in its own data centers that are located in the United States (US) and in the United Kingdom (UK). In both AWS Regions, the company uses a transit gateway to connect 15 VPCs to each other. The company has created a transit gateway peering connection between the two transit gateways. The VPC CIDR blocks do not overlap with each other or with IP addresses used within the data centers. The VPC CIDR prefixes can also be aggregated either on a Regional level or for the company's entire AWS environment.
The data centers are connected to each other by a private WAN connection. IP routing information is exchanged dynamically through Interior BGP (iBGP) sessions. The data centers maintain connectivity to AWS through one AWS Direct Connect connection in the US and one Direct Connect connection in the UK. Each Direct Connect connection is terminated on a Direct Connect gateway and is associated with a local transit gateway through a transit VIF.
Traffic follows the shortest geographical path from source to destination. For example, packets from the UK data center that are targeted to resources in eu-west-2 travel across the local Direct Connect connection. In cases of cross-Region data transfers, such as from the UK data center to VPCs in us-east-1, the private WAN connection must be used to minimize costs on AWS. A network engineer has configured each transit gateway association on the Direct Connect gateway to advertise VPC-specific CIDR IP prefixes only from the local Region. The routes toward the other Region must be learned through BGP from the routers in the other data center in the original, non-aggregated form.

The company recently experienced a problem with cross-Region data transfers because of issues with its private WAN connection. The network engineer needs to modify the routing setup to prevent similar interruptions in the future. The solution cannot modify the original traffic routing goal when the network is operating normally.
Which modifications will meet these requirements? (Choose two.)

A.Remove all the VPC CIDR prefixes from the list of subnets advertised through the local Direct Connect connection. Add the company's entire AWS environment aggregate route to the list of subnets advertised through the local Direct Connect connection.

B.Add the CIDR prefixes from the other Region VPCs and the local VPC CIDR blocks to the list of subnets advertised through the local Direct Connect connection. Configure data center routers to make routing decisions based on the BGP communities received.

C.Add the aggregate IP prefix for the other Region and the local VPC CIDR blocks to the list of subnets advertised through the local Direct Connect connection.

D.Add the aggregate IP prefix for the company's entire AWS environment and the local VPC CIDR blocks to the list of subnets advertised through the local Direct Connect connection.

E.Remove all the VPC CIDR prefixes from the list of subnets advertised through the local Direct Connect connection. Add both Regional aggregate IP prefixes to the list of subnets advertised through the Direct Connect connection on both sides of the network. Configure data center routers to make routing decisions based on the BGP communities received.

Answer: CE

Explanation:

Here's a breakdown of why options C and E are the correct choices for ensuring cross-Region data transfers continue during private WAN issues, without altering normal routing behavior:

Understanding the Requirements

The core goal is to maintain shortest-path routing when the private WAN is healthy and to failover to AWS infrastructure in case the WAN link fails. The solution must not impact the original traffic routing when the network is operating normally.

Why C is Correct

Adding the aggregate IP prefix for the other Region and the local VPC CIDR blocks to the list of subnets advertised through the local Direct Connect connection: This means the Direct Connect gateway (and hence, the local Direct Connect connection) will advertise aggregate routes for the VPCs in both the local region and the remote region. This enables the data center routers to learn routes to the remote region's VPCs via Direct Connect in addition to the private WAN.

Why E is Correct

Remove all the VPC CIDR prefixes from the list of subnets advertised through the local Direct Connect connection. This stops advertising specific VPC CIDRs via the local Direct Connect connection, which were originally limiting traffic to preferred paths.

Add both Regional aggregate IP prefixes to the list of subnets advertised through the Direct Connect connection on both sides of the network. This means the Direct Connect gateway (and hence, the local Direct Connect connection) will advertise the aggregate routes for the VPCs in both the local region and the remote region. This enables the data center routers to learn routes to the remote region's VPCs via Direct Connect in addition to the private WAN.

Configure data center routers to make routing decisions based on the BGP communities received. By leveraging BGP communities, the data center routers can be configured to prefer the routes learned via the private WAN (e.g., by assigning a higher local preference to routes received from the WAN). If the WAN fails, the routes learned via Direct Connect (with lower preference) will then be used. This ensures the traffic fails over to the AWS backbone without disrupting the original preferred path.

Why other options are Incorrect

A: Advertising only the entire AWS environment aggregate route through Direct Connect would mean that all traffic, even within the same region, could potentially be routed through the Direct Connect to the other region, which violates the shortest path objective during normal operations.

B: This approach doesn't solve the failover problem because data center routers might still prefer routes advertised from the local region via the Direct Connect.

D: Advertising the entire AWS environment aggregate route through Direct Connect would cause all traffic destined to AWS to route via the Direct Connect connection instead of the shorter path over the WAN. This removes the shortest path objective during normal operations.

Supporting Resources

AWS Transit Gateway:https://aws.amazon.com/transit-gateway/
AWS Direct Connect:https://aws.amazon.com/directconnect/
BGP Communities:https://datatracker.ietf.org/doc/html/rfc1997 (RFC defining BGP Communities)

Question: 21

A company's network engineer needs to design a new solution to help troubleshoot and detect network anomalies. The network engineer has configured Traffic Mirroring. However, the mirrored traffic is overwhelming the Amazon EC2 instance that is the traffic mirror target. The EC2 instance hosts tools that the company's security team uses to analyze the traffic. The network engineer needs to design a highly available solution that can scale to meet the demand of the mirrored traffic. Which solution will meet these requirements?

A.Deploy a Network Load Balancer (NLB) as the traffic mirror target. Behind the NLB. deploy a fleet of EC2 instances in an Auto Scaling group. Use Traffic Mirroring as necessary.

B.Deploy an Application Load Balancer (ALB) as the traffic mirror target. Behind the ALB, deploy a fleet of EC2 instances in an Auto Scaling group. Use Traffic Mirroring only during non-business hours.

C.Deploy a Gateway Load Balancer (GLB) as the traffic mirror target. Behind the GLB. deploy a fleet of EC2 instances in an Auto Scaling group. Use Traffic Mirroring as necessary.

D.Deploy an Application Load Balancer (ALB) with an HTTPS listener as the traffic mirror target. Behind the ALB. deploy a fleet of EC2 instances in an Auto Scaling group. Use Traffic Mirroring only during active events or business hours.

Answer: A

Explanation:

The correct answer is **C. Deploy a Gateway Load Balancer (GLB) as the traffic mirror target. Behind the GLB, deploy a fleet of EC2 instances in an Auto Scaling group. Use Traffic Mirroring as necessary.**

Here's why:

Traffic Mirroring and Load Balancers: Traffic Mirroring duplicates network traffic from EC2 instances and sends it to a target for inspection. Load balancers are used to distribute incoming traffic across multiple targets, providing scalability and high availability.

Gateway Load Balancer (GLB): The Gateway Load Balancer is specifically designed for virtual appliances like intrusion detection systems (IDS), intrusion prevention systems (IPS), firewalls, and deep packet inspection (DPI). These appliances often need to inspect all traffic flowing through a network, making the GLB the ideal choice for a traffic mirror target. The GLB uses the GENEVE protocol to forward the mirrored traffic along with metadata to the backend instances.

Auto Scaling Group: An Auto Scaling group ensures that the fleet of EC2 instances behind the GLB can scale up or down based on demand. This addresses the requirement for a solution that can handle varying amounts of mirrored traffic, ensuring high availability and preventing the analysis tools from being overwhelmed.

Why other options are incorrect:

Network Load Balancer (NLB): While NLBs can handle high throughput, they are designed for TCP, UDP, and TLS traffic. They lack the features and protocol support (GENEVE) necessary to efficiently handle mirrored traffic, especially traffic that includes additional metadata. While NLB could technically forward mirrored traffic, it's not optimized for this use case and would require more complex configurations to work effectively.

Application Load Balancer (ALB): ALBs operate at the application layer (Layer 7) and are designed for HTTP/HTTPS traffic. Mirrored traffic is typically raw network packets and would not be suitable for an ALB. Also, the suggestion to only use traffic mirroring during non-business hours or active events defeats the purpose of continuous network anomaly detection.

Solution Summary: By using a GLB, the mirrored traffic can be distributed across a fleet of EC2 instances in an Auto Scaling group, providing both scalability and high availability. This ensures that the security team's analysis tools can effectively process the mirrored traffic without being overwhelmed, enabling continuous network anomaly detection.

Supporting Links:

AWS Gateway Load Balancer:https://aws.amazon.com/elasticloadbalancing/gateway-load-balancer/
AWS Traffic Mirroring:https://docs.aws.amazon.com/vpc/latest/mirroring/traffic-mirroring-working.html
GENEVE Protocol:https://datatracker.ietf.org/doc/html/draft-ietf-geneve-06

Question: 22

A company uses a hybrid architecture and has an AWS Direct Connect connection between its on-premises data center and AWS. The company has production applications that run in the on-premises data center. The company also has production applications that run in a VPC. The applications that run in the on-premises data center need to communicate with the applications that run in the VPC. The company is using corp.example.com as the domain name for the on-premises resources and is using an Amazon Route 53 private hosted zone for aws.example.com to host the VPC resources.
The company is using an open-source recursive DNS resolver in a VPC subnet and is using a DNS resolver in the on-premises data center. The company's on-premises DNS resolver has a forwarder that directs requests for the aws.example.com domain name to the DNS resolver in the VPC. The DNS resolver in the VPC has a forwarder that directs requests for the corp.example.com domain name to the DNS resolver in the on-premises data center. The company has deckled to replace the open-source recursive DNS resolver with Amazon Route 53 Resolver endpoints.
Which combination of steps should a network engineer take to make this replacement? (Choose three.)

A.Create a Route 53 Resolver rule to forward aws.example.com domain queries to the IP addresses of the outbound endpoint.

B.Configure the on-premises DNS resolver to forward aws.example.com domain queries to the IP addresses of the inbound endpoint.

C.Create a Route 53 Resolver inbound endpoint and a Route 53 Resolver outbound endpoint.

D.Create a Route 53 Resolver rule to forward aws.example.com domain queries to the IP addresses of the inbound endpoint.

E.Create a Route 53 Resolver rule to forward corp.example.com domain queries to the IP address of the on-premises DNS resolver.

F.Configure the on-premises DNS resolver to forward aws.example.com queries to the IP addresses of the outbound endpoint.

Answer: BCE

Explanation:

The correct answer is BCE. Here's a breakdown of why:

C. Create a Route 53 Resolver inbound endpoint and a Route 53 Resolver outbound endpoint. This is the foundational step. Route 53 Resolver endpoints act as the bridge for DNS queries between your on-premises network and your VPC. The inbound endpoint allows on-premises resources to query Route 53, and the outbound endpoint allows VPC resources to query on-premises DNS. (https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver.html)

B. Configure the on-premises DNS resolver to forward aws.example.com domain queries to the IP addresses of the inbound endpoint. This ensures that when on-premises resources need to resolve names within the aws.example.com domain (hosted in the Route 53 private hosted zone), the queries are forwarded to the Route 53 Resolver inbound endpoint. This allows Route 53 to resolve those queries using the records in the private hosted zone.

E. Create a Route 53 Resolver rule to forward corp.example.com domain queries to the IP address of the on-premises DNS resolver. This step configures Route 53 to forward queries for the corp.example.com domain (used by on-premises resources) to the on-premises DNS resolver via the outbound endpoint. The rule ensures that the Route 53 Resolver (and thus, resources in the VPC) can resolve names within the on-premises domain.

Why other options are incorrect:

A & D: Forwarding aws.example.com to the outbound endpoint or to the inbound endpoint again would create a loop or break resolution. The on-premises DNS server should forward to the inbound endpoint. A Route 53 resolver rule forwards to on-premise DNS via the outbound endpoint.

F: Configuring the on-premises DNS resolver to forward to the outbound endpoint is incorrect. The on-premises DNS server should forward to the inbound endpoint to leverage the VPC-based resolvers.

In essence, the correct answer establishes bidirectional DNS resolution between the on-premises environment and the VPC, using Route 53 Resolver endpoints and rules to direct traffic to the correct destinations.

Question: 23

A government contractor is designing a multi-account environment with multiple VPCs for a customer. A network security policy requires all traffic between any two VPCs to be transparently inspected by a third-party appliance. The customer wants a solution that features AWS Transit Gateway. The setup must be highly available across multiple Availability Zones, and the solution needs to support automated failover. Furthermore, asymmetric routing is not supported by the inspection appliances.
Which combination of steps is part of a solution that meets these requirements? (Choose two.)

A.Deploy two clusters that consist of multiple appliances across multiple Availability Zones in a designated inspection VPC. Connect the inspection VPC to the transit gateway by using a VPC attachment. Create a target group, and register the appliances with the target group. Create a Network Load Balancer (NLB), and set it up to forward to the newly created target group. Configure a default route in the inspection VPCs transit gateway subnet toward the NLB.

B.Deploy two clusters that consist of multiple appliances across multiple Availability Zones in a designated inspection VPC. Connect the inspection VPC to the transit gateway by using a VPC attachment. Create a target group, and register the appliances with the target group. Create a Gateway Load Balancer, and set it up to forward to the newly created target group. Configure a default route in the inspection VPC's transit gateway subnet toward the Gateway Load Balancer endpoint.

C.Configure two route tables on the transit gateway. Associate one route table with all the attachments of the application VPCs. Associate the other route table with the inspection VPC's attachment. Propagate all VPC attachments into the inspection route table. Define a static default route in the application route table. Enable appliance mode on the attachment that connects the inspection VPC.

D.Configure two route tables on the transit gateway. Associate one route table with all the attachments of the application VPCs. Associate the other route table with the inspection VPCs attachment. Propagate all VPC

attachments into the application route table. Define a static default route in the inspection route table. Enable appliance mode on the attachment that connects the inspection VPC.

E.Configure one route table on the transit gateway. Associate the route table with all the VPCs. Propagate all VPC attachments into the route table. Define a static default route in the route table.

Answer: BC

Explanation:

The correct answer is BC. Here's a detailed explanation:

Choice B: Deploying Inspection Appliances with Gateway Load Balancer (GWLB)

This choice addresses the requirement for transparent inspection, high availability, automated failover, and handling asymmetric routing. The deployment of clusters of appliances across multiple AZs ensures high availability. The Gateway Load Balancer is designed specifically for virtual appliances and provides a transparent, scalable, and highly available way to direct traffic to these appliances for inspection. GWLB automatically handles appliance health checks and failover, fulfilling the automated failover requirement. Importantly, GWLB preserves source and destination IP addresses and ports, preventing asymmetric routing issues, as traffic both enters and exits through the same appliance. Configuring a default route in the inspection VPC's transit gateway subnet towards the Gateway Load Balancer endpoint ensures that all traffic destined for other VPCs is sent to the GWLB for inspection.

Choice C: Transit Gateway Route Table Configuration with Appliance Mode

This choice ensures that all inter-VPC traffic is routed through the inspection VPC, satisfying the network security policy requirement. By creating two route tables on the Transit Gateway (TGW), one for application VPCs and another for the inspection VPC, you can control the routing behavior. Associating the application VPC attachments with one route table and the inspection VPC attachment with the other allows you to isolate traffic flow. Propagating all VPC attachments into the inspection route table ensures the inspection VPC's appliances know about all possible destinations. Defining a static default route in the application route table pointing to the inspection VPC forces all inter-VPC traffic to pass through the inspection VPC. Enabling appliance mode on the inspection VPC attachment is crucial; it disables source/destination checks and ensures that the TGW forwards traffic to the correct appliance, handling asymmetric traffic correctly when combined with the GWLB setup.

Why other options are incorrect:

A: While NLB can distribute traffic, it's not designed for transparently inspecting network traffic like a GWLB. It can also cause asymmetric routing issues if not configured correctly.

D & E: These options incorrectly handle the route table configuration and lack the use of Gateway Load Balancer, which is crucial for appliance integration and handling asymmetric traffic.

Supporting Links:

AWS Transit Gateway:https://aws.amazon.com/transit-gateway/
Gateway Load Balancer:https://aws.amazon.com/elasticloadbalancing/gateway-load-balancer/
Transit Gateway Appliance Mode:https://docs.aws.amazon.com/transit-gateway/latest/userguide/configure-appliance.html

Question: 24

A company has deployed Amazon EC2 instances in private subnets in a VPC. The EC2 instances must initiate any requests that leave the VPC, including requests to the company's on-premises data center over an AWS Direct Connect connection. No resources outside the VPC can be allowed to open communications directly to the EC2 instances.

The on-premises data center's customer gateway is configured with a stateful firewall device that filters for incoming and outgoing requests to and from multiple VPCs. In addition, the company wants to use a single IP match rule to allow all the communications from the EC2 instances to its data center from a single IP address. Which solution will meet these requirements with the LEAST amount of operational overhead?

A.Create a VPN connection over the Direct Connect connection by using the on-premises firewall. Use the firewall to block all traffic from on premises to AWS. Allow a stateful connection from the EC2 instances to initiate the requests.

B.Configure the on-premises firewall to filter all requests from the on-premises network to the EC2 instances. Allow a stateful connection if the EC2 instances in the VPC initiate the traffic.

C.Deploy a NAT gateway into a private subnet in the VPC where the EC2 instances are deployed. Specify the NAT gateway type as private. Configure the on-premises firewall to allow connections from the IP address that is assigned to the NAT gateway.

D.Deploy a NAT instance into a private subnet in the VPC where the EC2 instances are deployed. Configure the on-premises firewall to allow connections from the IP address that is assigned to the NAT instance.

Answer: C

Explanation:

Here's a detailed justification for why option C is the correct answer:

The problem requires EC2 instances in private subnets to initiate outbound connections to an on-premises data center via Direct Connect, while preventing unsolicited inbound connections to the instances. Additionally, the company wants to use a single IP address for all outbound communication from the instances to the on-premises data center, and the solution must minimize operational overhead.

Option C, deploying a NAT gateway in a public subnet (a key correction, the question should have specified a public subnet for the NAT gateway to function) and configuring the on-premises firewall to allow connections from the NAT gateway's IP address, perfectly addresses these requirements. A NAT gateway allows instances in private subnets to initiate outbound traffic to the internet or, in this case, the Direct Connect connection to the on-premises data center. The NAT gateway performs Network Address Translation, replacing the private IP address of the EC2 instance with its own public IP address. This enables a single IP address for all outbound traffic from the EC2 instances to the on-premises data center, fulfilling the requirement for a single IP match rule on the firewall. Crucially, a NAT gateway only allows return traffic for connections initiated from within the VPC, preventing unsolicited inbound connections. A private NAT gateway does not exist.

Option A is incorrect because establishing a VPN over Direct Connect adds unnecessary complexity. Direct Connect already provides a dedicated network connection. Furthermore, relying on a VPN and blocking all traffic from on-premises to AWS is overly restrictive and complicates management.

Option B is insufficient because it only addresses the on-premises firewall configuration. It doesn't address how the EC2 instances will initiate outbound connections using a single IP address, and it doesn't provide a mechanism to prevent inbound connections.

Option D, deploying a NAT instance, is a valid approach but incurs more operational overhead than using a NAT gateway. NAT instances require manual patching, scaling, and management. A NAT gateway is a managed service, automatically scaling and patching itself. It also provides higher availability and throughput than a NAT instance.

Therefore, option C is the most efficient solution as it provides the required functionality with the least operational overhead by leveraging the managed NAT gateway service and presenting a single IP address to the on-premises firewall.

Supporting links:

AWS NAT Gateway:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html

AWS Direct Connect:https://aws.amazon.com/directconnect/

Question: 25

A global company operates all its non-production environments out of three AWS Regions: eu-west-1, us-east-1, and us-west-1. The company hosts all its production workloads in two on-premises data centers. The company has 60 AWS accounts and each account has two VPCs in each Region. Each VPC has a virtual private gateway where two VPN connections terminate for resilient connectivity to the data centers. The company has 360 VPN tunnels to each data center, resulting in high management overhead. The total VPN throughput for each Region is 500 Mbps.
The company wants to migrate the production environments to AWS. The company needs a solution that will simplify the network architecture and allow for future growth. The production environments will generate an additional 2 Gbps of traffic per Region back to the data centers. This traffic will increase over time.
Which solution will meet these requirements?

A.Set up an AWS Direct Connect connection from each data center to AWS in each Region. Create and attach private VIFs to a single Direct Connect gateway. Attach the Direct Connect gateway to all the VPCs. Remove the existing VPN connections that are attached directly to the virtual private gateways.

B.Create a single transit gateway with VPN connections from each data center. Share the transit gateway with each account by using AWS Resource Access Manager (AWS RAM). Attach the transit gateway to each VPC. Remove the existing VPN connections that are attached directly to the virtual private gateways.

C.Create a transit gateway in each Region with multiple newly commissioned VPN connections from each data center. Share the transit gateways with each account by using AWS Resource Access Manager (AWS RAM). In each Region, attach the transit gateway to each VPRemove the existing VPN connections that are attached directly to the virtual private gateways.

D.Peer all the VPCs in each Region to a new VPC in each Region that will function as a centralized transit VPC. Create new VPN connections from each data center to the transit VPCs. Terminate the original VPN connections that are attached to all the original VPCs. Retain the new VPN connection to the new transit VPC in each Region.

Answer: C

Explanation:

The correct answer is C. Here's why:

Why Option C is Correct:

Scalability and Future Growth: Transit Gateways (TGWs) are designed for connecting many VPCs and on-premises networks. Creating a TGW in each Region addresses the current need and allows for future expansion as the production environments are migrated.

Simplified Network Architecture: TGWs centralize routing, significantly reducing the complexity of managing numerous VPN connections. Each TGW serves as a hub, simplifying routing between VPCs and the data centers.

Increased Throughput: TGWs support higher bandwidth than individual VPN connections. Creating new VPN connections from each data center to each Region's TGW allows the 2 Gbps traffic requirement to be met and exceeded. The initial VPN connections only supported 500Mbps of traffic per region.

Resource Sharing: AWS Resource Access Manager (RAM) enables sharing the TGWs across multiple AWS accounts, reducing management overhead and ensuring consistent routing policies.

Regional Isolation: Deploying a TGW in each Region offers regional isolation. A failure in one Region's TGW will not affect connectivity in other Regions.

Why Other Options Are Incorrect:

Option A (Direct Connect): While Direct Connect provides high bandwidth and low latency, attaching private

VIFs to a single Direct Connect gateway and then attaching the Direct Connect gateway to all the VPCs is not the intended use case and will still require regional connections. Also, this option will only assist the production environments after they have been migrated. The company needs connectivity for both environments.

Option B (Single Transit Gateway): Creating a single TGW and sharing it across Regions using RAM is not possible. Transit Gateways are regional resources.

Option D (Transit VPC): Transit VPCs are an older solution and less scalable and manageable than Transit Gateways. They also introduce a single point of failure in each Region.

Supporting Concepts:

AWS Transit Gateway: A network transit hub that you can use to interconnect your virtual private clouds (VPC) and on-premises networks. https://aws.amazon.com/transit-gateway/
AWS Resource Access Manager (RAM): Enables you to share your AWS resources with any AWS account or within your organization in AWS Organizations. https://aws.amazon.com/ram/

By using Transit Gateways in each Region, the company achieves a scalable, manageable, and resilient solution for connecting its on-premises data centers to its AWS environments.

Question: 26

A company is building its website on AWS in a single VPC. The VPC has public subnets and private subnets in two Availability Zones. The website has static content such as images. The company is using Amazon S3 to store the content. The company has deployed a fleet of Amazon EC2 instances as web servers in a private subnet. The EC2 instances are in an Auto Scaling group behind an Application Load Balancer. The EC2 instances will serve traffic, and they must pull content from an S3 bucket to render the webpages. The company is using AWS Direct Connect with a public VIF for on-premises connectivity to the S3 bucket.
A network engineer notices that traffic between the EC2 instances and Amazon S3 is routing through a NAT gateway. As traffic increases, the company's costs are increasing. The network engineer needs to change the connectivity to reduce the NAT gateway costs that result from the traffic between the EC2 instances and Amazon S3.
Which solution will meet these requirements?

A.Create a Direct Connect private VIF. Migrate the traffic from the public VIF to the private VIF.

B.Create an AWS Site-to-Site VPN tunnel over the existing public VIF.

C.Implement interface VPC endpoints for Amazon S3. Update the VPC route table.

D.Implement gateway VPC endpoints for Amazon S3. Update the VPC route table.

Answer: D

Explanation:

The problem is that EC2 instances in private subnets are accessing S3 via a NAT gateway, incurring significant costs due to the data transfer. The goal is to reduce these costs while maintaining connectivity to S3.

Option A, creating a Direct Connect private VIF, doesn't solve the problem of NAT gateway usage. It merely shifts the traffic from the public VIF to a private VIF for on-premises access to AWS services, not from EC2 instances within the VPC. EC2 instances will still use the NAT gateway to reach the internet (and S3) if they don't have another route.

Option B, using a Site-to-Site VPN over the public VIF, only adds complexity and doesn't address the NAT gateway issue. It would still require the EC2 instances to route traffic through the VPN connection and then potentially still through the NAT gateway.

Option C, implementing interface VPC endpoints for S3, provides private connectivity to S3 but requires a Network Load Balancer (NLB) or Elastic Network Interface (ENI) within the VPC to function as a proxy. This is more complex and not the best solution for simple S3 access. Interface endpoints utilize PrivateLink, creating network interfaces within your subnets.

Option D, implementing gateway VPC endpoints for S3, directly addresses the issue. Gateway endpoints create a route within the VPC that directs traffic destined for S3 to a service prefix list, bypassing the NAT gateway. This creates a private connection to S3 without requiring public IPs or NAT gateways, significantly reducing costs. The EC2 instances can then pull content from S3 directly without incurring NAT gateway charges. This is the simplest and most cost-effective solution for private access to S3 from within a VPC. Gateway endpoints modify the route table to point traffic destined for S3 directly to S3.

Therefore, implementing gateway VPC endpoints for Amazon S3 and updating the VPC route table is the optimal solution for reducing NAT gateway costs.

Relevant documentation:

VPC Endpoints
Gateway VPC Endpoints
Interface VPC Endpoints

Question: 27

A company wants to improve visibility into its AWS environment. The AWS environment consists of multiple VPCs that are connected to a transit gateway. The transit gateway connects to an on-premises data center through an AWS Direct Connect gateway and a pair of redundant Direct Connect connections that use transit VIFs. The company must receive notification each time a new route is advertised to AWS from on premises over Direct Connect.
What should a network engineer do to meet these requirements?

A.Enable Amazon CloudWatch metrics on Direct Connect to track the received routes. Configure a CloudWatch alarm to send notifications when routes change.

B.Onboard Transit Gateway Network Manager to Amazon CloudWatch Logs Insights. Use Amazon EventBridge (Amazon CloudWatch Events) to send notifications when routes change.

C.Configure an AWS Lambda function to periodically check the routes on the Direct Connect gateway and to send notifications when routes change.

D.Enable Amazon CloudWatch Logs on the transit VIFs to track the received routes. Create a metric filter Set an alarm on the filter to send notifications when routes change.

Answer: B

Explanation:

The correct answer is B. Here's a detailed justification:

Why Option B is Correct:

Option B provides the most scalable, efficient, and near real-time solution for tracking route advertisements from on-premises over Direct Connect within a multi-VPC environment using a transit gateway. Transit Gateway Network Manager provides centralized visibility into your global network, including connections, traffic, and routes. Integrating it with CloudWatch Logs Insights allows you to query and analyze network data. EventBridge (formerly CloudWatch Events) is a serverless event bus that enables you to react to changes in your AWS environment in near real-time.

1. **Transit Gateway Network Manager:** Centralizes network monitoring for Transit Gateway-based networks, including Direct Connect attachments. It simplifies management and enhances visibility.

2. **CloudWatch Logs Insights:** Enables you to search, analyze, and visualize log data in CloudWatch Logs. Network Manager sends network events which you can log to CWL.
3. **EventBridge:** Serves as an event bus that reacts to changes in your AWS environment. With EventBridge, you can create rules to trigger actions based on specific events, such as changes in routing tables. These events would come from CWL after they are processed by CWL insights.

The combination allows the engineer to leverage the centralized network monitoring capabilities of Transit Gateway Network Manager with the log analysis power of CloudWatch Logs Insights and the automated notification functionality of EventBridge to receive immediate notifications when new routes are advertised from on-premises.

Why Other Options are Incorrect:

Option A: While CloudWatch metrics can provide aggregate information about Direct Connect, they lack the granularity to track individual route changes. This solution wouldn't provide near real-time notifications for specific route advertisements.

Option C: Using a Lambda function for periodic polling is inefficient and not scalable. Frequent polling consumes unnecessary resources and might miss rapid route changes. Furthermore, determining route changes requires significant logic within the Lambda function.

Option D: While it is possible to analyze logs on the transit VIF, it is far more complicated to extract route changes, and maintain. This method is not as straightforward, centralized, or manageable as leveraging Transit Gateway Network Manager. This option also does not integrate directly with EventBridge which makes reporting complex.

Authoritative Links:

Transit Gateway Network Manager:https://aws.amazon.com/transit-gateway/network-manager/
Amazon CloudWatch Logs Insights:https://aws.amazon.com/cloudwatch/features/logs-insights/
Amazon EventBridge:https://aws.amazon.com/eventbridge/

Question: 28

A software company offers a software-as-a-service (SaaS) accounting application that is hosted in the AWS Cloud The application requires connectivity to the company's on-premises network. The company has two redundant 10 GB AWS Direct Connect connections between AWS and its on-premises network to accommodate the growing demand for the application.
The company already has encryption between its on-premises network and the colocation. The company needs to encrypt traffic between AWS and the edge routers in the colocation within the next few months. The company must maintain its current bandwidth.
What should a network engineer do to meet these requirements with the LEAST operational overhead?

A.Deploy a new public VIF with encryption on the existing Direct Connect connections. Reroute traffic through the new public VIF.

B.Create a virtual private gateway Deploy new AWS Site-to-Site VPN connections from on premises to the virtual private gateway Reroute traffic from the Direct Connect private VIF to the new VPNs.

C.Deploy a new pair of 10 GB Direct Connect connections with MACsec. Configure MACsec on the edge routers. Reroute traffic to the new Direct Connect connections. Decommission the original Direct Connect connections

D.Deploy a new pair of 10 GB Direct Connect connections with MACsec. Deploy a new public VIF on the new Direct Connect connections. Deploy two AWS Site-to-Site VPN connections on top of the new public VIF. Reroute traffic from the existing private VIF to the new Site-to-Site connections. Decommission the original Direct Connect connections.

Answer: C

Explanation:

The best solution to encrypt traffic between AWS and the colocation edge routers while maintaining 10 Gbps bandwidth with minimal operational overhead is option C: deploy new Direct Connect connections with MACsec. Here's why:

MACsec on Direct Connect for Encryption: MACsec (Media Access Control Security) is a Layer 2 encryption protocol designed for point-to-point links like Direct Connect. It encrypts all traffic at the physical layer, ensuring data confidentiality. AWS supports MACsec on Direct Connect, allowing for secure communication between your on-premises network and AWS. https://aws.amazon.com/directconnect/features/

Maintaining Bandwidth: Option C involves deploying new 10 Gbps Direct Connect connections. This is crucial because it preserves the existing bandwidth capacity. Using new connections minimizes disruption and allows for a staged migration.

Minimal Operational Overhead: MACsec is implemented at the hardware level, minimizing performance impact and operational complexity. Once configured, the encryption and decryption processes are largely handled by the Direct Connect hardware.

Public VIF Inefficiency: Option A suggests a public VIF with encryption. Public VIFs are designed for accessing public AWS services over Direct Connect. They don't inherently provide encryption for all traffic between your on-premises network and AWS resources, requiring additional solutions such as VPN.

VPN Overhead: Option B and D suggest using Site-to-Site VPN over either Direct Connect or new Direct Connect. VPNs introduce significant overhead in terms of processing power for encryption and encapsulation. This added overhead will likely affect the overall network performance and reduce available bandwidth. VPN adds complexity compared to MACsec. The existing encryption is at the colocation edge routers, so there's no need to bring in additional complexity of Site-to-Site VPN tunnels.

Cost Efficiency: Although deploying new Direct Connect connections involves costs, it simplifies encryption and optimizes performance in the long run, offering a cost-effective solution compared to the operational overhead and bandwidth degradation associated with VPNs.

Therefore, deploying new Direct Connect connections with MACsec offers a balance between security, performance, and manageability, making it the most suitable solution with the least operational overhead.

Question: 29

A company hosts an application on Amazon EC2 instances behind an Application Load Balancer (ALB). The company recently experienced a network security breach. A network engineer must collect and analyze logs that include the client IP address, target IP address, target port, and user agent of each user that accesses the application.
What is the MOST operationally efficient solution that meets these requirements?

A.Configure the ALB to store logs in an Amazon S3 bucket. Download the files from Amazon S3, and use a spreadsheet application to analyze the logs.

B.Configure the ALB to push logs to Amazon Kinesis Data Streams. Use Amazon Kinesis Data Analytics to analyze the logs.

C.Configure Amazon Kinesis Data Streams to stream data from the ALB to Amazon OpenSearch Service (Amazon Elasticsearch Service). Use search operations in Amazon OpenSearch Service (Amazon Elasticsearch Service) to analyze the data.

D.Configure the ALB to store logs in an Amazon S3 bucket. Use Amazon Athena to analyze the logs in Amazon S3.

Answer: D

Explanation:

The most operationally efficient solution is **D. Configure the ALB to store logs in an Amazon S3 bucket. Use Amazon Athena to analyze the logs in Amazon S3.**

Here's why:

ALB Access Logs to S3: Application Load Balancers (ALBs) can directly log detailed access information to S3, including client IP, target IP, target port, and user agent. This meets the logging requirements of the problem statement. (See: https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-access-logs.html)
Amazon Athena for Analysis: Amazon Athena allows you to query data directly in S3 using standard SQL. It's serverless, meaning no infrastructure management, and you only pay for the queries you run. This simplifies the analysis of the logs. (https://aws.amazon.com/athena/)
Operationally Efficient: Athena eliminates the need to download large log files and analyze them locally with tools like spreadsheets (option A), which is less efficient for large datasets. It also avoids the complexity of setting up and managing streaming data pipelines like Kinesis Data Streams and OpenSearch Service (options B and C).

Cost Effective: Athena is cost-effective because you only pay for the queries you run. Other options might incur costs for continuous data streaming and storage, even when not actively analyzing the logs.

Scalability: Both S3 and Athena are highly scalable. S3 can store petabytes of log data, and Athena can query that data efficiently, scaling as needed.

Security Incident Response: Athena allows for rapid analysis of logs during a security incident, enabling faster identification of malicious activity and aiding in incident response.

Option A is inefficient for large datasets. Options B and C involve more complex setup and ongoing management of streaming data services and search clusters, which is not the most operationally efficient approach for the given requirements. The problem statement emphasizes "MOST operationally efficient," which Athena fulfills due to its serverless nature and direct SQL querying capabilities on S3 log data.

Question: 30

A media company is implementing a news website for a global audience. The website uses Amazon CloudFront as its content delivery network. The backend runs on Amazon EC2 Windows instances behind an Application Load Balancer (ALB). The instances are part of an Auto Scaling group. The company's customers access the website by using service example com as the CloudFront custom domain name. The CloudFront origin points to an ALB that uses service-alb.example.com as the domain name.
The company’s security policy requires the traffic to be encrypted in transit at all times between the users and the backend.
Which combination of changes must the company make to meet this security requirement? (Choose three.)

A.Create a self-signed certificate for service.example.com. Import the certificate into AWS Certificate Manager (ACM). Configure CloudFront to use this imported SSL/TLS certificate. Change the default behavior to redirect HTTP to HTTPS.

B.Create a certificate for service.example.com by using AWS Certificate Manager (ACM). Configure CloudFront to use this custom SSL/TLS certificate. Change the default behavior to redirect HTTP to HTTPS.

C.Create a certificate with any domain name by using AWS Certificate Manager (ACM) for the EC2 instances. Configure the backend to use this certificate for its HTTPS listener. Specify the instance target type during the creation of a new target group that uses the HTTPS protocol for its targets. Attach the existing Auto Scaling group to this new target group.

D.Create a public certificate from a third-party certificate provider with any domain name for the EC2 instances. Configure the backend to use this certificate for its HTTPS listener. Specify the instance target type during the creation of a new target group that uses the HTTPS protocol for its targets. Attach the existing Auto Scaling group to this new target group.

E.Create a certificate for service-alb.example.com by using AWS Certificate Manager (ACM). On the ALB add a new HTTPS listener that uses the new target group and the service-alb.example.com ACM certificate. Modify the CloudFront origin to use the HTTPS protocol only. Delete the HTTP listener on the ALB.

F.Create a self-signed certificate for service-alb.example.com. Import the certificate into AWS Certificate Manager (ACM). On the ALB add a new HTTPS listener that uses the new target group and the imported service-alb.example.com ACM certificate. Modify the CloudFront origin to use the HTTPS protocol only. Delete the HTTP listener on the ALB.

Answer: BDE

Explanation:

The correct answer is BDE. Here's why:

B: Create a certificate for service.example.com by using AWS Certificate Manager (ACM). Configure CloudFront to use this custom SSL/TLS certificate. Change the default behavior to redirect HTTP to HTTPS. This is necessary to ensure secure communication between the users and CloudFront. ACM provides free SSL/TLS certificates that are trusted by browsers. By associating the certificate for the custom domain service.example.com with CloudFront, you establish an HTTPS connection. Redirecting HTTP to HTTPS enforces encryption for all requests. Reference:
https://docs.aws.amazon.com/cloudfront/latest/DeveloperGuide/using-https-cloudfront-to-viewers.html

D: Create a public certificate from a third-party certificate provider with any domain name for the EC2 instances. Configure the backend to use this certificate for its HTTPS listener. Specify the instance target type during the creation of a new target group that uses the HTTPS protocol for its targets. Attach the existing Auto Scaling group to this new target group. We require the communication between CloudFront and ALB to be encrypted as well, it will do that by adding a new HTTPS listener to the ALB with certificate.

E: Create a certificate for service-alb.example.com by using AWS Certificate Manager (ACM). On the ALB add a new HTTPS listener that uses the new target group and the service-alb.example.com ACM certificate. Modify the CloudFront origin to use the HTTPS protocol only. Delete the HTTP listener on the ALB. This step ensures secure communication between CloudFront and the ALB. Creating an ACM certificate for the ALB's domain (service-alb.example.com) and configuring an HTTPS listener on the ALB enables HTTPS.
Modifying the CloudFront origin to use HTTPS ensures that CloudFront only sends encrypted traffic to the ALB. Finally, removing the HTTP listener prevents unencrypted traffic to the ALB. Reference:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-listeners.html

Options A and F are incorrect because self-signed certificates are not trusted by browsers and will result in security warnings. Option C is incorrect because you cannot use the same ACM certificate created with "any" domain name for the EC2 instance to secure the ALB communication.

Question: 31

A company is hosting an application on Amazon EC2 instances behind a Network Load Balancer (NLB). A solutions architect added EC2 instances in a second Availability Zone to improve the availability of the application. The solutions architect added the instances to the NLB target group.
The company's operations team notices that traffic is being routed only to the instances in the first Availability Zone.
What is the MOST operationally efficient solution to resolve this issue?

A.Enable the new Availability Zone on the NLB
B.Create a new NLB for the instances in the second Availability Zone
C.Enable proxy protocol on the NLB
D.Create a new target group with the instances in both Availability Zones

Answer: A

Explanation:

The most operationally efficient solution is to enable the new Availability Zone on the Network Load Balancer (NLB).

Here's why: NLBs are designed to distribute traffic across multiple Availability Zones for high availability.
Simply adding instances in a new AZ to the target group doesn't automatically mean the NLB will start routing traffic there. You need to explicitly enable that AZ for the NLB itself. Enabling the AZ allows the NLB to create cross-zone load balancing and health checks to the instances in the newly added AZ.

Option B, creating a new NLB, is inefficient because it introduces unnecessary complexity and cost. You would then need to manage two separate NLBs. Option C, enabling proxy protocol, addresses the issue of preserving client IP addresses but does not affect cross-zone load balancing; it is related but does not solve the actual routing problem. Option D, creating a new target group with instances in both AZs, is unnecessary. The instances are already in a target group, and the NLB can distribute traffic among instances in a single target group across multiple AZs if those AZs are enabled for the NLB. Modifying or recreating the target group does not address the underlying problem of whether the NLB is aware of the new AZ or not.

Enabling the AZ for the existing NLB is the simplest and most direct solution. It leverages the existing infrastructure without introducing new components or management overhead.

Therefore, enabling the new Availability Zone on the NLB is the most operationally efficient solution.

Relevant documentation for further research:

Network Load Balancer Availability Zones
Add or remove Availability Zones for your load balancer

Question: 32

A network engineer needs to set up an Amazon EC2 Auto Scaling group to run a Linux-based network appliance in a highly available architecture. The network engineer is configuring the new launch template for the Auto Scaling group.
In addition to the primary network interface the network appliance requires a second network interface that will be used exclusively by the application to exchange traffic with hosts over the internet. The company has set up a Bring Your Own IP (BYOIP) pool that includes an Elastic IP address that should be used as the public IP address for the second network interface.
How can the network engineer implement the required architecture?

A.Configure the two network interfaces in the launch template. Define the primary network interface to be created in one of the private subnets. For the second network interface, select one of the public subnets. Choose the BYOIP pool ID as the source of public IP addresses.

B.Configure the primary network interface in a private subnet in the launch template. Use the user data option to run a cloud-init script after boot to attach the second network interface from a subnet with auto-assign public IP addressing enabled.

C.Create an AWS Lambda function to run as a lifecycle hook of the Auto Scaling group when an instance is launching. In the Lambda function, assign a network interface to an AWS Global Accelerator endpoint.

D.During creation of the Auto Scaling group, select subnets for the primary network interface. Use the user data option to run a cloud-init script to allocate a second network interface and to associate an Elastic IP address from the BYOIP pool.

Answer: D

Explanation:

The correct answer is **D** because it provides the most flexible and manageable way to configure the second network interface and associate the BYOIP Elastic IP address with the EC2 instances in the Auto Scaling group. Here's a detailed justification:

Auto Scaling Groups and Launch Templates/Configurations: Auto Scaling groups use launch templates or configurations to define the instances they create. While launch templates can define one network interface, they aren't easily modifiable post-launch for advanced configuration.
https://docs.aws.amazon.com/autoscaling/ec2/userguide/LaunchTemplates.html

User Data and Cloud-Init: The user data field allows you to specify scripts that are executed when an instance starts. Cloud-init is a widely used package that handles early initialization of cloud instances. This makes it suitable for configuring the second network interface.
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html

BYOIP and Elastic IP Addresses: BYOIP allows bringing your own IP addresses to AWS and associating them with Elastic IP addresses. You can then associate these Elastic IPs with your EC2 instances. Answer D uses user data to allocate the ENI and then associate the EIP.
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-byoip.html

Why other options are less ideal:

A: Configuring both interfaces in the launch template won't allow associating a specific BYOIP Elastic IP as a public IP for the second ENI, giving you less control.

B: While technically feasible, automatically assigning a public IP through subnet settings doesn't utilize the BYOIP pool which is a key requirement.

C: While Lifecycle hooks and Lambda can configure resources, this approach adds significant complexity and is unnecessary for a simple task like assigning a second ENI and associating a BYOIP EIP. Using Lambda to connect a network interface to a Global Accelerator endpoint is not a direct solution for assigning a BYOIP to the second network interface of an EC2 instance within an autoscaling group. This approach adds more complexity than is needed.

In summary, option D provides a programmatic yet manageable way to attach the second network interface and associate the required BYOIP Elastic IP address during the instance launch process. It uses user data and cloud-init, industry-standard tools for instance configuration, adhering to best practices for managing resources in an Auto Scaling environment.

Question: 33

A company delivers applications over the internet. An Amazon Route 53 public hosted zone is the authoritative DNS service for the company and its internet applications, all of which are offered from the same domain name.
A network engineer is working on a new version of one of the applications. All the application's components are hosted in the AWS Cloud. The application has a three-tier design. The front end is delivered through Amazon EC2 instances that are deployed in public subnets with Elastic IP addresses assigned. The backend components are deployed in private subnets from RFC1918.
Components of the application need to be able to access other components of the application within the application's VPC by using the same host names as the host names that are used over the public internet. The network engineer also needs to accommodate future DNS changes, such as the introduction of new host names or the retirement of DNS entries.
Which combination of steps will meet these requirements? (Choose three.)

A.Add a geoproximity routing policy in Route 53.

B.Create a Route 53 private hosted zone for the same domain name Associate the application’s VPC with the new private hosted zone.

C.Enable DNS hostnames for the application's VPC.

D.Create entries in the private hosted zone for each name in the public hosted zone by using the corresponding private IP addresses.

E.Create an Amazon EventBridge (Amazon CloudWatch Events) rule that runs when AWS CloudTrail logs a Route 53 API call to the public hosted zone. Create an AWS Lambda function as the target of the rule. Configure the function to use the event information to update the private hosted zone.

F.Add the private IP addresses in the existing Route 53 public hosted zone.

Answer: BCD

Explanation:

The correct solution is BCD. Here's a detailed justification:

B. Create a Route 53 private hosted zone for the same domain name and associate the application's VPC with the new private hosted zone. This is essential for providing internal DNS resolution that is distinct from the public DNS. By creating a private hosted zone with the same domain name as the public zone and associating it with the application's VPC, we ensure that internal traffic within the VPC will query this private zone first. Route 53 will then resolve names to private IP addresses within the VPC.
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/hosted-zones-private.html

C. Enable DNS hostnames for the application's VPC. Enabling DNS hostnames ensures that EC2 instances within the VPC are automatically assigned private DNS hostnames, which are required for resolution within the VPC. If this isn't enabled, instances would not get the internal DNS entries required for resolution.
https://docs.aws.amazon.com/vpc/latest/userguide/vpc-dns.html

D. Create entries in the private hosted zone for each name in the public hosted zone by using the corresponding private IP addresses. This step ensures that when components inside the VPC query the same hostname used externally, they resolve to the internal (private) IP addresses of the application components.
This allows the application to use the same hostnames internally as it does externally, simplifying configuration and making the solution more flexible.

Option A (geoproximity routing) is not relevant because the requirement focuses on internal name resolution within the VPC. Geoproximity policies are used to route traffic to different endpoints based on geographical location.

Option E (EventBridge and Lambda) is an overly complex and potentially unreliable way to keep the private and public hosted zones synchronized. This could lead to latency and potential errors and would require intricate monitoring and management.

Option F (Adding private IPs to the public hosted zone) would expose the internal infrastructure to the public internet, which is a security risk and violates best practices. Public DNS should only contain records for publicly accessible resources. Therefore, BCD is the only combination that meets the stated requirements and follows best practices.

Question: 34

A company is deploying an application. The application is implemented in a series of containers in an Amazon Elastic Container Service (Amazon ECS) cluster. The company will use the Fargate launch type for its tasks. The containers will run workloads that require connectivity initiated over an SSL connection. Traffic must be able to flow to the application from other AWS accounts over private connectivity. The application must scale in a manageable way as more consumers use the application.
Which solution will meet these requirements?

A.Choose a Gateway Load Balancer (GLB) as the type of load balancer for the ECS service. Create a lifecycle hook to add new tasks to the target group from Amazon ECS as required to handle scaling. Specify the GLB in the service definition. Create a VPC peer for external AWS accounts. Update the route tables so that the AWS accounts can reach the GLB.

B.Choose an Application Load Balancer (ALB) as the type of load balancer for the ECS service. Create path-based routing rules to allow the application to target the containers that are registered in the target group. Specify the ALB in the service definition. Create a VPC endpoint service for the ALB Share the VPC endpoint service with other AWS accounts.

C.Choose an Application Load Balancer (ALB) as the type of load balancer for the ECS service. Create path-based routing rules to allow the application to target the containers that are registered in the target group. Specify the ALB in the service definition. Create a VPC peer for the external AWS accounts. Update the route tables so that the AWS accounts can reach the ALB.

D.Choose a Network Load Balancer (NLB) as the type of load balancer for the ECS service. Specify the NLB in the service definition. Create a VPC endpoint service for the NLB. Share the VPC endpoint service with other AWS accounts.

Answer: D

Explanation:

Here's a detailed justification for why option D is the correct answer:

The requirements highlight several critical aspects: SSL connectivity, private connectivity from other AWS accounts, scalability, and containerization using ECS with Fargate. Let's break down why each component of option D addresses these needs.

Network Load Balancer (NLB): NLBs are designed for high performance and handle TCP, UDP, and TLS (SSL) traffic efficiently. They provide static IP addresses per Availability Zone, which are essential for creating a VPC endpoint service. https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html **VPC Endpoint Service:** A VPC endpoint service enables private connectivity between VPCs, even across different AWS accounts. It allows other AWS accounts to access your application without exposing it to the public internet. This directly fulfills the requirement for private connectivity from other accounts.

https://docs.aws.amazon.com/vpc/latest/privatelink/endpoint-services.html
Sharing the VPC Endpoint Service: Once the VPC endpoint service is created for the NLB, it can be shared with specific AWS accounts. These accounts can then create VPC endpoints within their VPCs to privately connect to the NLB, and consequently, the ECS Fargate tasks behind it.

Why other options are less suitable:

Option A (Gateway Load Balancer): Gateway Load Balancers are primarily used for network virtual appliances (like firewalls or intrusion detection systems), not directly for application traffic routing like SSL-based application requests. They are also more complex to configure for simple application scaling. **Options B and C (Application Load Balancer):** While ALBs support SSL termination and path-based routing, they don't natively provide the simplest or most efficient method for cross-account private connectivity compared to VPC endpoint services. Using VPC peering with ALBs requires more configuration and management of route tables in each account. Additionally, ALBs are designed for HTTP/HTTPS traffic and might not be the best choice if the application requires direct TCP or TLS connections. Also, sharing an ALB using a VPC endpoint service is not as straightforward as with an NLB.

In summary, option D provides the most straightforward, scalable, and secure solution for enabling private SSL-based connectivity to an ECS Fargate application from other AWS accounts, using an NLB and VPC endpoint service. This approach minimizes complexity and leverages AWS's managed services effectively.

Question: 35

A company's development team has created a new product recommendation web service. The web service is hosted in a VPC with a CIDR block of 192.168.224.0/19. The company has deployed the web service on Amazon EC2 instances and has configured an Auto Scaling group as the target of a Network Load Balancer (NLB).
The company wants to perform testing to determine whether users who receive product recommendations spend more money than users who do not receive product recommendations. The company has a big sales event in 5 days and needs to integrate its existing production environment with the recommendation engine by then. The existing production environment is hosted in a VPC with a CIDR block of 192.168.128 0/17.
A network engineer must integrate the systems by designing a solution that results in the least possible disruption to the existing environments.

Which solution will meet these requirements?

A.Create a VPC peering connection between the web service VPC and the existing production VPC. Add a routing rule to the appropriate route table to allow data to flow to 192.168.224.0/19 from the existing production environment and to flow to 192.168.128.0/17 from the web service environment. Configure the relevant security groups and ACLs to allow the systems to communicate.

B.Ask the development team of the web service to redeploy the web service into the production VPC and integrate the systems there.

C.Create a VPC endpoint service. Associate the VPC endpoint service with the NLB for the web service. Create an interface VPC endpoint for the web service in the existing production VPC.

D.Create a transit gateway in the existing production environment. Create attachments to the production VPC and the web service VPC. Configure appropriate routing rules in the transit gateway and VPC route tables for 192.168.224.0/19 and 192.168.128.0/17. Configure the relevant security groups and ACLs to allow the systems to communicate.

Answer: C

Explanation:

The best solution is **C. Create a VPC endpoint service. Associate the VPC endpoint service with the NLB for the web service. Create an interface VPC endpoint for the web service in the existing production VPC.**

Here's why:

Least Disruption: VPC endpoints, specifically interface endpoints using PrivateLink, offer the least disruptive integration method. They allow private access to services in another VPC without exposing the traffic to the public internet or requiring complex routing configurations. This minimizes changes to existing infrastructure and reduces potential for errors.

Scalability and Availability: Using the NLB with the VPC endpoint service ensures that the production environment can scale its access to the recommendation engine as needed. The NLB distributes traffic across multiple instances of the web service, providing high availability.

Security: Traffic between the production VPC and the web service VPC remains within the AWS network, improving security by avoiding public internet exposure. Security groups and network ACLs can be configured to further restrict access.

Speed of Implementation: Setting up a VPC endpoint service and interface endpoint is relatively straightforward, allowing for quick integration within the 5-day deadline.

Why other options are not ideal:

A (VPC Peering): While possible, VPC peering has limitations with overlapping CIDR blocks (which is not the case here, but should be checked). Also, peering requires careful route table management and can become complex with larger networks. It also has the potential risk of exposing more resources than necessary between VPCs.

B (Redeploying): Redeploying the web service into the production VPC is the most disruptive option, requiring significant code changes, testing, and potential downtime. It's also not a clean separation of concerns.

D (Transit Gateway): Transit Gateway is more suitable for connecting multiple VPCs and on-premises networks. It's an overkill solution for connecting just two VPCs, especially considering the need for minimal disruption and a quick turnaround. It would also be more expensive to implement than VPC Endpoints.

Supporting Concepts and Links:

AWS PrivateLink:https://aws.amazon.com/privatelink/
VPC Endpoints:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html
Network Load Balancer (NLB):
https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html

Question: 36

A network engineer needs to update a company's hybrid network to support IPv6 for the upcoming release of a new application. The application is hosted in a VPC in the AWS Cloud. The company's current AWS infrastructure includes VPCs that are connected by a transit gateway. The transit gateway is connected to the on-premises network by AWS Direct Connect and AWS Site-to-Site VPN. The company's on-premises devices have been updated to support the new IPv6 requirements.
The company has enabled IPv6 for the existing VPC by assigning a new IPv6 CIDR block to the VPC and by assigning IPv6 to the subnets for dual-stack support. The company has launched new Amazon EC2 instances for the new application in the updated subnets.
When updating the hybrid network to support IPv6 the network engineer must avoid making any changes to the current infrastructure. The network engineer also must block direct access to the instances' new IPv6 addresses from the internet. However, the network engineer must allow outbound internet access from the instances. What is the MOST operationally efficient solution that meets these requirements?

A.Update the Direct Connect transit VIF and configure BGP peering with the AWS assigned IPv6 peering address. Create a new VPN connection that supports IPv6 connectivity. Add an egress-only internet gateway. Update any affected VPC security groups and route tables to provide connectivity within the VPC and between the VPC and the on-premises devices

B.Update the Direct Connect transit VIF and configure BGP peering with the AWS assigned IPv6 peering address. Update the existing VPN connection to support IPv6 connectivity. Add an egress-only internet gateway. Update any affected VPC security groups and route tables to provide connectivity within the VPC and between the VPC and the on-premises devices.

C.Create a Direct Connect transit VIF and configure BGP peering with the AWS assigned IPv6 peering address. Create a new VPN connection that supports IPv6 connectivity. Add an egress-only internet gateway. Update any affected VPC security groups and route tables to provide connectivity within the VPC and between the VPC and the on-premises devices.

D.Create a Direct Connect transit VIF and configure BGP peering with the AWS assigned IPv6 peering address. Create a new VPN connection that supports IPv6 connectivity. Add a NAT gateway. Update any affected VPC security groups and route tables to provide connectivity within the VPC and between the VPC and the on-premises devices.

Answer: A

Explanation:

The most operationally efficient solution is **A**. Here's why:

Direct Connect and IPv6: To extend IPv6 connectivity over Direct Connect, the existing transit virtual interface (VIF) should be updated for IPv6 BGP peering. This allows the exchange of IPv6 routes between AWS and the on-premises network via the Transit Gateway. Using the AWS assigned IPv6 peering address is required for setting up BGP for IPv6 on Direct Connect.
Site-to-Site VPN and IPv6: Since a connection is already in place to the on-premise environment and IPv6 support is required a new VPN connection that supports IPv6 connectivity needs to be created.
Egress-Only Internet Gateway: To prevent direct inbound access to the EC2 instances' IPv6 addresses from the internet while allowing outbound internet access, an egress-only internet gateway is the correct choice. This gateway allows instances within the VPC to initiate outbound IPv6 traffic to the internet but prevents the internet from initiating IPv6 connections to the instances.
Security Groups and Route Tables: Security groups control inbound and outbound traffic at the instance level, and route tables determine the path that network traffic takes. These need to be updated to allow IPv6 traffic between the VPC subnets, on-premises network (via Transit Gateway), and the egress-only internet gateway.

Options B, C and D are incorrect because:

Option B suggests updating the VPN connection instead of creating a new one. IPv4 and IPv6 VPN connections are supported simultaneously.

Option C proposes creating a Direct Connect transit VIF from scratch. The question states that avoiding changes to current infrastructure is necessary so reusing the current VIF is best.
Option D proposes creating a NAT gateway. This is for IPv4 traffic, not IPv6, and thus doesn't fit the scenario.

Supporting Links:

AWS Direct Connect and IPv6:https://docs.aws.amazon.com/directconnect/latest/UserGuide/direct-connect-ipv6.html
Egress-Only Internet Gateway:https://docs.aws.amazon.com/vpc/latest/userguide/egress-only-internet-gateway.html
AWS Site-to-Site VPN and IPv6:https://docs.aws.amazon.com/vpn/latest/s2svpn/VPNS2S_VPNProtocol.html

Question: 37

A network engineer must provide additional safeguards to protect encrypted data at Application Load Balancers (ALBs) through the use of a unique random session key.
What should the network engineer do to meet this requirement?

A.Change the ALB security policy to a policy that supports TLS 1.2 protocol only
B.Use AWS Key Management Service (AWS KMS) to encrypt session keys
C.Associate an AWS WAF web ACL with the ALBs. and create a security rule to enforce forward secrecy (FS)
D.Change the ALB security policy to a policy that supports forward secrecy (FS)

Answer: D

Explanation:

The requirement is to protect encrypted data at Application Load Balancers (ALBs) using a unique random session key, adding an extra layer of security. Forward secrecy (FS) achieves this.

Option D, changing the ALB security policy to one that supports forward secrecy (FS), directly addresses this. FS ensures that even if a private key is compromised, past session keys remain secure because they weren't derived from the private key itself. Instead, each session uses a unique, randomly generated session key. TLS cipher suites that implement protocols such as Diffie-Hellman Ephemeral (DHE) and Elliptic-Curve Diffie-Hellman Ephemeral (ECDHE) provide FS. By selecting an ALB security policy that includes these cipher suites, the ALB will negotiate connections using FS, generating a new session key for each session.

Option A, focusing only on TLS 1.2, is not sufficient. While using TLS 1.2 is good security practice, TLS 1.2 itself doesn't automatically guarantee FS. The specific cipher suites used within TLS 1.2 determine whether FS is enabled.

Option B, using AWS KMS to encrypt session keys, isn't applicable for ALB session keys used for TLS connections. AWS KMS is helpful for encrypting data at rest or when managing encryption keys for services like S3 or EBS, but it's not directly involved in the ephemeral session key exchange in TLS with ALBs.

Option C, associating an AWS WAF web ACL with the ALB and creating a security rule to enforce FS is incorrect. AWS WAF protects web applications from common web exploits, but it doesn't directly enforce the forward secrecy mechanism at the TLS layer. WAF operates at a higher level (application layer), inspecting traffic based on rules. While WAF can add security, it does not replace the need for configuring the ALB with a security policy that supports FS.

Therefore, enabling forward secrecy on the ALB directly addresses the requirement of generating unique, random session keys for each session, ensuring that past communications remain secure even if the server's private key is compromised in the future.

Supporting Links:

AWS Load Balancer Security Policies:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/create-https-listener.html#security-policies
Forward Secrecy: https://en.wikipedia.org/wiki/Forward_secrecy

Question: 38

A company has deployed a software-defined WAN (SD-WAN) solution to interconnect all of its offices. The company is migrating workloads to AWS and needs to extend its SD-WAN solution to support connectivity to these workloads.
A network engineer plans to deploy AWS Transit Gateway Connect and two SD-WAN virtual appliances to provide this connectivity. According to company policies, only a single SD-WAN virtual appliance can handle traffic from AWS workloads at a given time.
How should the network engineer configure routing to meet these requirements?

A.Add a static default route in the transit gateway route table to point to the secondary SD-WAN virtual appliance. Add routes that are more specific to point to the primary SD-WAN virtual appliance.

B.Configure the BGP community tag 7224:7300 on the primary SD-WAN virtual appliance for BGP routes toward the transit gateway.

C.Configure the AS_PATH prepend attribute on the secondary SD-WAN virtual appliance for BGP routes toward the transit gateway.

D.Disable equal-cost multi-path (ECMP) routing on the transit gateway for Transit Gateway Connect.

Answer: C

Explanation:

The correct answer is C. Here's why:

The requirement is for only one SD-WAN appliance to actively handle traffic from AWS workloads at a time. This implies an active/passive failover scenario. We need a mechanism to ensure the primary appliance is preferred for routing traffic.

Option C uses the AS_PATH prepend attribute of BGP on the secondary appliance. AS_PATH prepend works by artificially lengthening the AS_PATH attribute of the BGP route advertised by the secondary appliance.

BGP prefers routes with shorter AS_PATH lengths. Therefore, the Transit Gateway will always prefer the route advertised by the primary appliance (which has a shorter, or default, AS_PATH) unless the primary fails.

When the primary appliance is unavailable, the Transit Gateway will choose the route with the longer AS_PATH from the secondary appliance, thus initiating failover.

Option A is incorrect because static routes are generally less dynamic than BGP and would require manual intervention to failover. Adding a default route to the secondary appliance and more specific routes to the primary appliance would likely create routing conflicts or require complex route management.

Option B suggesting a BGP community tag (7224:7300) is relevant to Transit Gateway route tables, not for influencing route selection when multiple routes are available. BGP communities are often used for traffic engineering, filtering, or tagging routes, but not as the primary mechanism for active/passive failover based on path preference.

Option D, disabling ECMP, does not achieve the desired result. ECMP spreads traffic across equal-cost paths. The goal is not to load balance between appliances, but to ensure one appliance actively handles the traffic with a seamless failover to the second.

In summary, the AS_PATH prepend on the secondary appliance provides a standard, BGP-based, and dynamic

way to implement an active/passive failover mechanism for SD-WAN connectivity to AWS workloads via Transit Gateway.

Relevant links for more information:

AWS Transit Gateway:https://aws.amazon.com/transit-gateway/
BGP Path Attributes:https://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/16045-bgp-attributes.html (Cisco Doc, but explains the concepts)
AS_PATH Prepending:https://networklessons.com/bgp/bgp-as-path-prepend-configuration

Question: 39

A company is planning to deploy many software-defined WAN (SD-WAN) sites. The company is using AWS Transit Gateway and has deployed a transit gateway in the required AWS Region. A network engineer needs to deploy the SD-WAN hub virtual appliance into a VPC that is connected to the transit gateway. The solution must support at least 5 Gbps of throughput from the SD-WAN hub virtual appliance to other VPCs that are attached to the transit gateway.
Which solution will meet these requirements?

A.Create a new VPC for the SD-WAN hub virtual appliance. Create two IPsec VPN connections between the SD-WAN hub virtual appliance and the transit gateway. Configure BGP over the IPsec VPN connections

B.Assign a new CIDR block to the transit gateway. Create a new VPC for the SD-WAN hub virtual appliance. Attach the new VPC to the transit gateway with a VPC attachment. Add a transit gateway Connect attachment. Create a Connect peer and specify the GRE and BGP parameters. Create a route in the appropriate VPC for the SD-WAN hub virtual appliance to route to the transit gateway.

C.Create a new VPC for the SD-WAN hub virtual appliance. Attach the new VPC to the transit gateway with a VPC attachment. Create two IPsec VPN connections between the SD-WAN hub virtual appliance and the transit gateway. Configure BGP over the IPsec VPN connections.

D.Assign a new CIDR block to the transit gateway. Create a new VPC for the SD-WAN hub virtual appliance. Attach the new VPC to the transit gateway with a VPC attachment. Add a transit gateway Connect attachment. Create a Connect peer and specify the VXLAN and BGP parameters. Create a route in the appropriate VPC for the SD-WAN hub virtual appliance to route to the transit gateway.

Answer: B

Explanation:

The correct answer is B. Here's why:

Transit Gateway Connect: AWS Transit Gateway Connect enables you to use GRE/VXLAN tunnels for higher bandwidth and better performance compared to IPsec VPNs. It is specifically designed for integrating SD-WAN appliances with Transit Gateway.

https://aws.amazon.com/blogs/networking-and-content-delivery/aws-transit-gateway-connect-sd-wan-connectivity-and-performance-enhancements/

CIDR block for Transit Gateway: While not strictly necessary for the connectivity itself, assigning a new CIDR block to the Transit Gateway helps with better route management and avoiding overlapping CIDRs between your VPCs and SD-WAN network. This becomes more important with a large number of SD-WAN sites.

https://docs.aws.amazon.com/transit-gateway/latest/userguide/how-transit-gateways-work.html

New VPC: Isolating the SD-WAN hub virtual appliance in its own VPC promotes better security and network management.

VPC Attachment: Attaching the VPC to the Transit Gateway allows connectivity to other VPCs connected to the Transit Gateway.

Connect Attachment and Peer: The Transit Gateway Connect attachment and Connect Peer configuration, using GRE and BGP, establishes the tunnel and routing required for SD-WAN to function correctly. GRE encapsulation allows non-IP protocols to be carried over IP networks, which may be necessary for some SD-WAN implementations. BGP ensures dynamic routing between the SD-WAN network and the AWS environment.

Routing: Adding a route in the VPC route table for the SD-WAN appliance, directing traffic destined for other connected VPCs to the Transit Gateway, is essential for directing traffic to the SD-WAN hub and onward.

Why other options are incorrect:

A and C (IPsec VPN): IPsec VPNs have a lower throughput limit compared to Transit Gateway Connect, generally topping out around 1.25 Gbps per VPN connection. While you could use multiple VPN connections in parallel, this adds complexity and is not the optimal solution for 5 Gbps requirement.

D (VXLAN): While VXLAN is an encapsulation protocol, GRE is more commonly used and compatible for most SD-WAN appliance integrations with AWS Transit Gateway. Choosing VXLAN without a specific requirement is less suitable.

In summary, Transit Gateway Connect offers a high-bandwidth, efficient way to integrate SD-WAN appliances with AWS Transit Gateway using GRE tunnels and BGP for routing. The setup also includes the required VPC attachment, CIDR assignment and the correct route configurations to facilitate SD-WAN functionality.

Question: 40

A company is deploying a new application on AWS. The application uses dynamic multicasting. The company has five VPCs that are all attached to a transit gateway Amazon EC2 instances in each VPC need to be able to register dynamically to receive a multicast transmission.
How should a network engineer configure the AWS resources to meet these requirements?

A.Create a static source multicast domain within the transit gateway. Associate the VPCs and applicable subnets with the multicast domain. Register the multicast senders' network interface with the multicast domain. Adjust the network ACLs to allow UDP traffic from the source to all receivers and to allow UDP traffic that is sent to the multicast group address.

B.Create a static source multicast domain within the transit gateway. Associate the VPCs and applicable subnets with the multicast domain. Register the multicast senders' network interface with the multicast domain. Adjust the network ACLs to allow TCP traffic from the source to all receivers and to allow TCP traffic that is sent to the multicast group address.

C.Create an Internet Group Management Protocol (IGMP) multicast domain within the transit gateway. Associate the VPCs and applicable subnets with the multicast domain. Register the multicast senders' network interface with the multicast domain. Adjust the network ACLs to allow UDP traffic from the source to all receivers and to allow UDP traffic that is sent to the multicast group address.

D.Create an Internet Group Management Protocol (IGMP) multicast domain within the transit gateway. Associate the VPCs and applicable subnets with the multicast domain. Register the multicast senders' network interface with the multicast domain. Adjust the network ACLs to allow TCP traffic from the source to all receivers and to allow TCP traffic that is sent to the multicast group address.

Answer: C

Explanation:

The correct answer is C because it leverages IGMP for dynamic multicast group membership and uses UDP for multicast traffic. Let's break down why.

The problem statement indicates that instances need to dynamically register to receive multicast transmissions.
IGMP (Internet Group Management Protocol) is a protocol used to manage multicast group memberships.
Receivers use IGMP to inform a multicast router (in this case, the transit gateway) that they

want to join a specific multicast group. This allows the router to forward multicast traffic only to the VPCs/subnets containing interested receivers. A "static source multicast domain" doesn't support this dynamic registration.

Therefore, the transit gateway needs to be configured with an IGMP multicast domain to handle dynamic group membership. VPCs and their relevant subnets must be associated with this multicast domain. Additionally, the multicast senders' network interfaces need to be registered within the multicast domain. This informs the transit gateway which source is authorized to send multicast traffic.

Multicast generally uses UDP (User Datagram Protocol) rather than TCP (Transmission Control Protocol). UDP is connectionless, making it more suitable for the one-to-many nature of multicast, where reliability is often sacrificed for speed and efficiency. Thus, network ACLs need to be adjusted to permit UDP traffic between the source and all receivers and traffic sent to the multicast group address. By opening UDP at the network ACL level, this guarantees traffic will traverse.

In summary, IGMP is the key to dynamic multicast, UDP is standard for multicast traffic, and network ACLs act as a firewall to restrict unwanted traffic. The other options are incorrect because they either use static multicast domains which don't support dynamic registration, or specify TCP, which is not commonly used for multicast.

Reference links:

AWS Transit Gateway Multicast: https://docs.aws.amazon.com/vpc/latest/tgw/tgw-multicast.html IGMP: https://en.wikipedia.org/wiki/Internet_Group_Management_Protocol
Network ACLs: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html

Question: 41

A company is creating new features for its ecommerce website. These features will use several microservices that are accessed through different paths. The microservices will run on Amazon Elastic Container Service (Amazon ECS). The company requires the use of HTTPS for all of its public websites. The application requires the customer's source IP addresses. A network engineer must implement a load balancing strategy that meets these requirements.
Which combination of actions should the network engineer take to accomplish this goal? (Choose two.)

A.Use a Network Load Balancer
B.Retrieve client IP addresses by using the X-Forwarded-For header
C.Use AWS App Mesh load balancing
D.Retrieve client IP addresses by using the X-IP-Source header
E.Use an Application Load Balancer.

Answer: BE

Explanation:

The correct answer is BE. Here's why:

E. Use an Application Load Balancer (ALB): ALBs are ideal for HTTP and HTTPS traffic. They operate at the application layer (Layer 7), which allows them to make routing decisions based on the content of the request (like path or host headers). This is perfectly suited for routing requests to different microservices based on the URL path, as mentioned in the question. ALBs also provide native HTTPS support, fulfilling the requirement for secure connections.

B. Retrieve client IP addresses by using the X-Forwarded-For header: When an ALB terminates SSL/TLS connections, the original client IP address is not directly available to the backend servers (ECS containers in

this case). The ALB adds the client IP address to the X-Forwarded-For header in the request it forwards to the backend. The microservices can then read this header to determine the client's original IP address.

Why other options are incorrect:

A. Use a Network Load Balancer (NLB): NLBs operate at the transport layer (Layer 4), which means they forward traffic based on IP address and port. They don't have the application-level awareness to route based on URL path. While they preserve the source IP, using an ALB is more appropriate given the need for path-based routing and HTTPS termination.

C. Use AWS App Mesh load balancing: AWS App Mesh is a service mesh that provides application-level networking. While it can be used with ECS, it primarily focuses on managing service-to-service communication within the mesh. Using App Mesh alone does not provide the public HTTPS termination and source IP preservation needed directly from a public facing load balancer in this scenario.

D. Retrieve client IP addresses by using the X-IP-Source header: There is no standard or common header named X-IP-Source used for transmitting the client IP address. The X-Forwarded-For header is the widely accepted and implemented standard.

Supporting Documentation:

Application Load Balancer:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html
X-Forwarded-For Header:https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-For **AWS App Mesh:**https://aws.amazon.com/app-mesh/

In conclusion, an ALB offers the required HTTPS support and path-based routing, while the X-Forwarded-For header provides the client's original IP address to the microservices, fulfilling all the stated requirements.

Question: 42

A company is migrating its containerized application to AWS. For the architecture the company will have an ingress VPC with a Network Load Balancer (NLB) to distribute the traffic to front-end pods in an Amazon Elastic Kubernetes Service (Amazon EKS) cluster. The front end of the application will determine which user is requesting access and will send traffic to 1 of 10 services VPCs. Each services VPC will include an NLB that distributes traffic to the services pods in an EKS cluster. The company is concerned about overall cost. User traffic will be responsible for more than 10 TB of data transfer from the ingress VPC to services VPCs every month. A network engineer needs to recommend how to design the communication between the VPCs.
Which solution will meet these requirements at the LOWEST cost?

A.Create a transit gateway. Peer each VPC to the transit gateway. Use zonal DNS names for the NLB in the services VPCs to minimize cross-AZ traffic from the ingress VPC to the services VPCs.

B.Create an AWS PrivateLink endpoint in every Availability Zone in the ingress VPC. Each PrivateLink endpoint will point to the zonal DNS entry of the NLB in the services VPCs.

C.Create a VPC peering connection between the ingress VPC and each of the 10 services VPCs. Use zonal DNS names for the NLB in the services VPCs to minimize cross-AZ traffic from the ingress VPC to the services VPCs.

D.Create a transit gateway. Peer each VPC to the transit gateway. Turn off cross-AZ load balancing on the transit gateway. Use Regional DNS names for the NLB in the services VPCs.

Answer: C

Explanation:

The most cost-effective solution for high data transfer between the ingress VPC and the 10 services VPCs is VPC peering with zonal NLB DNS names.

Explanation:

Cost Considerations: The primary concern is minimizing costs associated with high data transfer (10+ TB/month). AWS charges for data transfer between Availability Zones (AZs) and across VPCs.

VPC Peering: VPC peering provides a direct, private network connection between VPCs. Data transfer within a VPC peering connection is generally cheaper than routing through a Transit Gateway or using AWS PrivateLink due to the reduced complexity and infrastructure overhead.

Zonal DNS Names: NLBs have both regional and zonal DNS names. Using zonal DNS names for the service NLBs ensures that traffic from the ingress VPC is routed to an NLB in the same Availability Zone, if possible. This significantly reduces or eliminates data transfer charges between Availability Zones.

Why not Transit Gateway? A Transit Gateway introduces additional data processing and associated costs. While Transit Gateway simplifies management for a larger number of VPCs (dozens or hundreds), it's not the most cost-effective option for only 10 VPCs, especially with the high data transfer volume.

Why not PrivateLink? PrivateLink creates VPC endpoints within the ingress VPC, which then connect to Network Load Balancers (NLBs) in the service VPCs. While it offers enhanced security, PrivateLink adds cost complexity and typically doesn't provide a significant cost advantage over VPC peering for this specific scenario.

Why not Transit Gateway with disabled cross-AZ Load Balancing? The Transit Gateway itself still carries cost overhead, and while turning off Cross-AZ Load Balancing may reduce some costs, it forces traffic into single AZs leading to availability challenges.

In summary, VPC peering with zonal NLB DNS names provides a direct, lower-cost path for data transfer, while minimizing or eliminating cross-AZ data transfer charges, addressing the problem statement's concerns.

Supporting Links:

VPC Peering: https://docs.aws.amazon.com/vpc/latest/peering/what-is-vpc-peering.html
Network Load Balancer DNS Records:
https://docs.aws.amazon.com/elasticloadbalancing/latest/network/network-load-balancers.html#availability-zones
AWS Pricing: https://aws.amazon.com/pricing/ (check the data transfer section)
Transit Gateway: https://aws.amazon.com/transit-gateway/
AWS PrivateLink: https://aws.amazon.com/privatelink/

Question: 43

A company has stateful security appliances that are deployed to multiple Availability Zones in a centralized shared services VPC. The AWS environment includes a transit gateway that is attached to application VPCs and the shared services VPC. The application VPCs have workloads that are deployed in private subnets across multiple Availability Zones. The stateful appliances in the shared services VPC inspect all east west (VPC-to-VPC) traffic.
Users report that inter-VPC traffic to different Availability Zones is dropping. A network engineer verified this claim by issuing Internet Control Message Protocol (ICMP) pings between workloads in different Availability Zones across the application VPCs. The network engineer has ruled out security groups, stateful device configurations and network ACLs as the cause of the dropped traffic.
What is causing the traffic to drop?

A.The stateful appliances and the transit gateway attachments are deployed in a separate subnet in the shared services VPC.

B.Appliance mode is not enabled on the transit gateway attachment to the shared services VPC.

C.The stateful appliances and the transit gateway attachments are deployed in the same subnet in the shared services VPC.

D.Appliance mode is not enabled on the transit gateway attachment to the application VPCs.

Answer: B

Explanation:

Here's a detailed justification for why option B is the correct answer:

The problem describes a scenario where inter-VPC traffic between Availability Zones is being dropped when it traverses stateful security appliances in a shared services VPC via a transit gateway. The key element causing this issue is the lack of "appliance mode" enabled on the transit gateway attachment to the shared services VPC, where the security appliances reside.

Without appliance mode, the transit gateway's default behavior might cause asymmetric routing. Asymmetric routing occurs when the request and response traffic flow through different paths. For example, a packet might enter the security appliance from AZ-A, but the return traffic might attempt to bypass the appliance and go directly back through AZ-B due to the transit gateway's path selection. Stateful appliances require traffic to flow symmetrically; if they only see one direction of a flow, they will drop the packets because they cannot maintain the state of the connection.

Appliance mode, when enabled, ensures that traffic flows symmetrically through the security appliances. It forces the transit gateway to route return traffic for a given flow back through the same appliance instance that initially inspected the traffic, regardless of the destination Availability Zone. This prevents the stateful appliance from dropping packets due to incomplete flow information.

Option A is incorrect because the subnet configuration of the appliances and transit gateway attachments doesn't directly cause asymmetric routing. The subnet merely provides a logical network segment for the resources.

Option C is incorrect because deploying appliances and attachments in the same subnet doesn't inherently cause asymmetric routing. The key issue is the traffic path selection of the transit gateway.

Option D is incorrect because appliance mode on the application VPC attachments isn't relevant to the issue. The problem focuses on traffic inspection by appliances in the shared services VPC. The symmetric routing issue occurs when the transit gateway fails to ensure traffic returns to the same appliance it initially passed through in the shared services VPC.

In summary, enabling appliance mode on the transit gateway attachment to the shared services VPC ensures traffic symmetry, allowing the stateful appliances to correctly inspect and forward traffic between VPCs in different Availability Zones.

Supporting Links:

AWS Transit Gateway Appliance Mode: https://docs.aws.amazon.com/vpc/latest/tgw/tgw-appliance-mode.html
AWS Transit Gateway Routing: https://docs.aws.amazon.com/vpc/latest/tgw/tgw-routing.html

Question: 44

A company has hundreds of Amazon EC2 instances that are running in two production VPCs across all Availability Zones in the us-east-1 Region. The production VPCs are named
VPC A and VPC B.
A new security regulation requires all traffic between production VPCs to be inspected before the traffic is routed to its final destination. The company deploys a new shared VPC that contains a stateful firewall appliance and a transit gateway with a VPC attachment across all VPCs to route traffic between VPC A and VPC B through the firewall appliance for inspection.
During testing, the company notices that the transit gateway is dropping the traffic whenever the traffic is between two Availability Zones.
What should a network engineer do to fix this issue with the LEAST management overhead?

A.In the shared VPC, replace the VPC attachment with a VPN attachment. Create a VPN tunnel between the transit gateway and the firewall appliance. Configure BGP.

B.Enable transit gateway appliance mode on the VPC attachment in VPC A and VPC B.

C.Enable transit gateway appliance mode on the VPC attachment in the shared VPC.

D.In the shared VPC, configure one VPC peering connection to VPC A and another VPC peering connection to VPC B.

Answer: C

Explanation:

The correct answer is C: Enable transit gateway appliance mode on the VPC attachment in the shared VPC.

Justification:

The problem describes a scenario where traffic between VPC A and VPC B is being dropped when it traverses Availability Zones via the transit gateway. This is happening because the stateful firewall appliance needs to maintain session affinity (ensuring traffic for a specific session always goes to the same appliance instance). Without appliance mode enabled, the transit gateway might route return traffic from the firewall appliance to a different Availability Zone, causing the stateful firewall to drop the traffic because it doesn't recognize the existing session.

Appliance mode on the transit gateway VPC attachment is designed specifically for scenarios where you have network appliances like firewalls. Enabling appliance mode ensures that traffic entering and exiting the VPC attachment is always routed through the same Availability Zone, thus preserving session affinity. This prevents asymmetrical routing, where the initial request and response take different paths, which is crucial for stateful firewalls.

Option A is incorrect because replacing the VPC attachment with a VPN attachment introduces unnecessary complexity and management overhead. VPNs are better suited for connecting to on-premises networks or remote sites, not for internal VPC traffic within a region. Configuring BGP also adds to the management burden.

Option B is incorrect because enabling appliance mode on the VPC attachments in VPC A and VPC B is not the appropriate solution. Appliance mode needs to be enabled on the transit gateway's VPC attachment within the shared VPC where the firewall appliance resides. This tells the transit gateway to route traffic through the firewall appliance's AZ consistently.

Option D is incorrect because VPC peering connections do not work with Transit Gateways to provide a central routing point, and is not designed to facilitate routing through the security appliance. This would negate the purpose of the transit gateway and firewall appliance architecture.

Enabling appliance mode on the shared VPC's transit gateway attachment provides the simplest and most direct solution to ensure traffic for a specific session remains within the same Availability Zone, allowing the stateful firewall to function correctly and resolve the dropped traffic issue. This approach minimizes management overhead while satisfying the requirement for traffic inspection.

Authoritative Links:

AWS Transit Gateway Appliance Mode:https://docs.aws.amazon.com/transit-gateway/latest/userguide/tgw-appliance-mode.html
AWS Transit Gateway Routing:https://docs.aws.amazon.com/transit-gateway/latest/userguide/tgw-routing.html

Question: 45

A company has deployed a critical application on a fleet of Amazon EC2 instances behind an Application Load Balancer. The application must always be reachable on port 443 from the public internet. The application recently had an outage that resulted from an incorrect change to the EC2 security group.
A network engineer needs to automate a way to verify the network connectivity between the public internet and

the EC2 instances whenever a change is made to the security group. The solution also must notify the network engineer when the change affects the connection.
Which solution will meet these requirements?

A.Enable VPC Flow Logs on the elastic network interface of each EC2 instance to capture REJECT traffic on port 443. Publish the flow log records to a log group in Amazon CloudWatch Logs. Create a CloudWatch Logs metric filter for the log group for rejected traffic. Create an alarm to notify the network engineer.

B.Enable VPC Flow Logs on the elastic network interface of each EC2 instance to capture all traffic on port 443. Publish the flow log records to a log group in Amazon CloudWatch Logs. Create a CloudWatch Logs metric filter for the log group for all traffic. Create an alarm to notify the network engineer

C.Create a VPC Reachability Analyzer path on port 443. Specify the security group as the source. Specify the EC2 instances as the destination. Create an Amazon Simple Notification Service (Amazon SNS) topic to notify the network engineer when a change to the security group affects the connection. Create an AWS Lambda function to start Reachability Analyzer and to publish a message to the SNS topic in case the analyses fail Create an Amazon EventBridge (Amazon CloudWatch Events) rule to invoke the Lambda function when a change to the security group occurs.

D.Create a VPC Reachability Analyzer path on port 443. Specify the internet gateway of the VPC as the source. Specify the EC2 instances as the destination. Create an Amazon Simple Notification Service (Amazon SNS) topic to notify the network engineer when a change to the security group affects the connection. Create an AWS Lambda function to start Reachability Analyzer and to publish a message to the SNS topic in case the analyses fail. Create an Amazon EventBridge (Amazon CloudWatch Events) rule to invoke the Lambda function when a change to the security group occurs.

Answer: D

Explanation:

The correct answer is D because it provides a complete and automated solution for verifying network connectivity from the public internet to the EC2 instances after security group changes. Here's a breakdown of why it's the best choice:

1. **VPC Reachability Analyzer:** VPC Reachability Analyzer is a network diagnostics tool that helps you analyze and troubleshoot network reachability between two endpoints in your VPC. By creating a path on port 443, you can specifically test the connectivity required for the application.
   [https://docs.aws.amazon.com/vpc/latest/reachability/what-is-reachability-analyzer.html]

2. **Source and Destination:** Specifying the internet gateway as the source accurately simulates traffic originating from the public internet. The EC2 instances are the intended destination of this traffic, ensuring the test reflects the real-world scenario.

3. **Automation and Notification:** The solution uses several services to achieve automation and notification upon failure:

   Amazon SNS Topic: The SNS topic facilitates notification to the network engineer when the reachability analysis fails.

   AWS Lambda Function: The Lambda function is the key to automation. It starts the Reachability Analyzer and publishes a message to the SNS topic if connectivity fails.

   Amazon EventBridge (CloudWatch Events) Rule: This rule triggers the Lambda function whenever there's a change to the security group. This allows you to react instantly when your critical security configurations are modified.

4. **Comprehensive Solution:** This approach is comprehensive because it proactively verifies connectivity after security group changes. It leverages the combination of tools to test network paths and provide automatic alerts, allowing the network engineer to immediately investigate and rectify the connectivity issues caused by the modifications.

Why other options are not the best:

Option A & B (VPC Flow Logs): While VPC Flow Logs can capture rejected traffic, they are reactive. They only

detect failures after they happen. They don't proactively verify connectivity after a security group change. It also doesn't address simulating traffic from the internet. Furthermore, filtering all traffic (option B) is inefficient and generates unnecessary logs. [https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html]
Option C (Security Group as Source): Using the security group as the source in the Reachability Analyzer isn't appropriate to test connectivity from the internet. The source should represent where the traffic is originating from, which is the public internet. The security group is an access control mechanism, not a source of network traffic.

Question: 46

A security team is performing an audit of a company's AWS deployment. The security team is concerned that two applications might be accessing resources that should be blocked by network ACLs and security groups. The applications are deployed across two Amazon Elastic Kubernetes Service (Amazon EKS) clusters that use the Amazon VPC Container Network Interface (CNI) plugin for Kubernetes. The clusters are in separate subnets within the same VPC and have a Cluster Autoscaler configured.
The security team needs to determine which POD IP addresses are communicating with which services throughout the VPC. The security team wants to limit the number of flow logs and wants to examine the traffic from only the two applications.
Which solution will meet these requirements with the LEAST operational overhead?

A.Create VPC flow logs in the default format. Create a filter to gather flow logs only from the EKS nodes. Include the srcaddr field and the dstaddr field in the flow logs.

B.Create VPC flow logs in a custom format. Set the EKS nodes as the resource Include the pkt-srcaddr field and the pkt-dstaddr field in the flow logs.

C.Create VPC flow logs in a custom format. Set the application subnets as resources. Include the pkt-srcaddr field and the pkt-dstaddr field in the flow logs.

D.Create VPC flow logs in a custom format. Create a filter to gather flow logs only from the EKS nodes. Include the pkt-srcaddr field and the pkt-dstaddr field in the flow logs.

Answer: C

Explanation:

The correct answer is **C**. Here's a detailed justification:

The goal is to identify the pod IP addresses involved in communication between two specific applications across two EKS clusters within the same VPC, while minimizing operational overhead and limiting flow logs to only traffic relevant to the applications.

Option A is incorrect because creating VPC flow logs in the default format and filtering by EKS nodes will capture traffic not related to the applications of interest. This increases the volume of logs and adds operational overhead by requiring filtering of the relevant flows. The default VPC flow log format might not provide the necessary level of packet-level detail (specifically pod IP addresses).

Option B is incorrect because while using a custom format and focusing on EKS nodes is better than logging everything, it still misses the core requirement: filtering logs specifically for the application subnets, not just the nodes. This is because traffic flows from different PODs within those subnets. Additionally, VPC Flow Logs do not associate directly with EKS nodes as a resource.

Option D is incorrect because while using a custom format and filtering EKS nodes helps reducing overhead, it is still more effective to specify subnets instead.

Option C is the most appropriate solution because:

1. **Targets Application Traffic:** Setting the application subnets as the resource for VPC flow logs ensures that only traffic originating from or destined to those subnets (where the applications reside)

is captured. This directly addresses the requirement of focusing only on the two applications' traffic.

2. **Includes Pod IP Addresses:** The pkt-srcaddr and pkt-dstaddr fields in the custom format provide the source and destination IP addresses at the packet level. When using the Amazon VPC CNI plugin for Kubernetes, these IP addresses will be the pod IP addresses. This allows the security team to determine which specific pods are communicating with each other and with other services.

3. **Minimizes Operational Overhead:** By focusing on the application subnets, the volume of flow logs is minimized, reducing storage costs and the effort required to analyze the logs. There is no additional filtering required.

Therefore, creating VPC flow logs in a custom format with the application subnets as the resource and including the pkt-srcaddr and pkt-dstaddr fields is the most efficient and effective way to meet the security team's requirements.

Supporting resources:

VPC Flow Logs:https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html
VPC Flow Log Record Syntax:https://docs.aws.amazon.com/vpc/latest/userguide/flow-log-records.html
Amazon VPC CNI plugin for Kubernetes:https://docs.aws.amazon.com/eks/latest/userguide/pod-networking.html

Question: 47

A data analytics company has a 100-node high performance computing (HPC) cluster. The HPC cluster is for parallel data processing and is hosted in a VPC in the AWS Cloud. As part of the data processing workflow, the HPC cluster needs to perform several DNS queries to resolve and connect to Amazon RDS databases, Amazon S3 buckets, and on-premises data stores that are accessible through AWS Direct Connect. The HPC cluster can increase in size by five to seven times during the company's peak event at the end of the year.
The company is using two Amazon EC2 instances as primary DNS servers for the VPC. The EC2 instances are configured to forward queries to the default VPC resolver for Amazon Route 53 hosted domains and to the on-premises DNS servers for other on-premises hosted domain names. The company notices job failures and finds that DNS queries from the HPC cluster nodes failed when the nodes tried to resolve RDS and S3 bucket endpoints.
Which architectural change should a network engineer implement to provide the DNS service in the MOST scalable way?

A.Scale out the DNS service by adding two additional EC2 instances in the VPC. Reconfigure half of the HPC cluster nodes to use these new DNS servers. Plan to scale out by adding additional EC2 instance-based DNS servers in the future as the HPC cluster size grows.

B.Scale up the existing EC2 instances that the company is using as DNS servers. Change the instance size to the largest possible instance size to accommodate the current DNS load and the anticipated load in the future.

C.Create Route 53 Resolver outbound endpoints. Create Route 53 Resolver rules to forward queries to on-premises DNS servers for on premises hosted domain names. Reconfigure the HPC cluster nodes to use the default VPC resolver instead of the EC2 instance-based DNS servers. Terminate the EC2 instances.

D.Create Route 53 Resolver inbound endpoints. Create rules on the on-premises DNS servers to forward queries to the default VPC resolver. Reconfigure the HPC cluster nodes to forward all DNS queries to the on-premises DNS servers. Terminate the EC2 instances.

Answer: C

Explanation:

The best architectural change to address the DNS resolution issues and scalability requirements is **C. Create Route 53 Resolver outbound endpoints. Create Route 53 Resolver rules to forward queries to on-premises DNS servers for on premises hosted domain names. Reconfigure the HPC cluster nodes to use the default VPC resolver instead of the EC2 instance-based DNS servers. Terminate the EC2 instances.**

Here's why:

Scalability and Availability: Route 53 Resolver is a highly scalable and available managed DNS service. It automatically scales to handle a large number of DNS queries, eliminating the need for manual scaling and management as the HPC cluster grows. This contrasts with solutions A and B, which involve scaling EC2 instances, a less scalable and more management-intensive approach.

Integration with AWS Services: Route 53 Resolver integrates seamlessly with other AWS services like RDS and S3. The default VPC resolver handles DNS resolution for public AWS service endpoints. Using the VPC resolver, the HPC nodes will consistently and reliably resolve DNS for RDS and S3.

Hybrid DNS Resolution: Option C allows for hybrid DNS resolution. Route 53 Resolver outbound endpoints enable the VPC to forward DNS queries to on-premises DNS servers via Direct Connect for on-premises hosted domain names. This ensures the HPC cluster can resolve both AWS and on-premises resources.

Reduced Management Overhead: By using Route 53 Resolver, the company can eliminate the need to manage and maintain EC2 instance-based DNS servers, reducing operational overhead. This simplifies the DNS infrastructure and frees up resources for other tasks.

Reliability: The Amazon Route 53 Resolver is highly reliable because it is a managed service by AWS, which offers a fully managed DNS infrastructure for both on-premises and cloud resources. This service scales automatically to serve large volumes of DNS queries without any need of manual intervention.

Options A and B are less desirable because they involve managing EC2 instances for DNS, which adds complexity and operational overhead. Option D is less efficient because it requires forwarding all DNS queries to the on-premises DNS servers, which can introduce latency and increase the load on the on-premises infrastructure.

Authoritative Links:

Route 53 Resolver
How Route 53 Resolver Works

Question: 48

A company's network engineer is designing an active-passive connection to AWS from two on-premises data centers. The company has set up AWS Direct Connect connections between the on-premises data centers and AWS. From each location, the company is using a transit VIF that connects to a Direct Connect gateway that is associated with a transit gateway.
The network engineer must ensure that traffic from AWS to the data centers is routed first to the primary data center. The traffic should be routed to the failover data center only in the case of an outage.
Which solution will meet these requirements?

A.Set the BGP community tag for all prefixes from the primary data center to 7224:7100. Set the BGP community tag for all prefixes from the failover data center to 7224:7300

B.Set the BGP community tag for all prefixes from the primary data center to 7224:7300. Set the BGP community tag for all prefixes from the failover data center to 7224:7100

C.Set the BGP community tag for all prefixes from the primary data center to 7224:9300. Set the BGP community tag for all prefixes from the failover data center to 7224:9100

D.Set the BGP community tag for all prefixes from the primary data center to 7224:9100. Set the BGP community tag for all prefixes from the failover data center to 7224:9300

Answer: B

Explanation:

The correct answer is B. Here's why:

AWS uses specific BGP community tags to influence routing decisions when using Direct Connect and Transit Gateways. To achieve active-passive routing, we need to prioritize the primary data center over the failover data center for traffic originating from AWS. This is achieved using the 7224:7300 and 7224:7100 BGP

community tags.

Specifically, the 7224:7300 BGP community tag influences the Transit Gateway to prefer the path advertising prefixes with this tag. Conversely, 7224:7100 indicates a lower preference.

Therefore, by setting the BGP community tag for prefixes advertised from the primary data center to 7224:7300, we tell the Transit Gateway to prefer this path when routing traffic back to the on-premises networks. Setting the BGP community tag for prefixes advertised from the failover data center to 7224:7100 makes the Transit Gateway use this path only when the primary path is unavailable. This ensures that traffic flows to the primary data center under normal circumstances and fails over to the failover data center only in case of an outage.

Options C and D use the 7224:9100 and 7224:9300 BGP community tags, which are used to control prefix advertisement to specific AWS Regions and are irrelevant to achieving active/passive routing to on-premises data centers. Option A incorrectly assigns tags which would make the failover data center the preferred path.

Refer to the AWS documentation on Direct Connect routing policies for more details:
https://docs.aws.amazon.com/directconnect/latest/UserGuide/routing-policies.html

Question: 49

A real estate company is building an internal application so that real estate agents can upload photos and videos of various properties. The application will store these photos and videos in an Amazon S3 bucket as objects and will use Amazon DynamoDB to store corresponding metadata. The S3 bucket will be configured to publish all PUT events for new object uploads to an Amazon Simple Queue Service (Amazon SQS) queue.
A compute cluster of Amazon EC2 instances will poll the SQS queue to find out about newly uploaded objects. The cluster will retrieve new objects, perform proprietary image and video recognition and classification update metadata in DynamoDB and replace the objects with new watermarked objects. The company does not want public IP addresses on the EC2 instances.
Which networking design solution will meet these requirements MOST cost-effectively as application usage increases?

A.Place the EC2 instances in a public subnet. Disable the Auto-assign Public IP option while launching the EC2 instances. Create an internet gateway. Attach the internet gateway to the VPC. In the public subnet's route table, add a default route that points to the internet gateway.

B.Place the EC2 instances in a private subnet. Create a NAT gateway in a public subnet in the same Availability Zone. Create an internet gateway. Attach the internet gateway to the VPC. In the public subnet's route table, add a default route that points to the internet gateway

C.Place the EC2 instances in a private subnet. Create an interface VPC endpoint for Amazon SQS. Create gateway VPC endpoints for Amazon S3 and DynamoDB.

D.Place the EC2 instances in a private subnet. Create a gateway VPC endpoint for Amazon SQS. Create interface VPC endpoints for Amazon S3 and DynamoDB.

Answer: C

Explanation:

The correct solution (C) offers the most cost-effective and secure networking design as application usage increases.

Option C places the EC2 instances in a private subnet, ensuring they don't have public IP addresses, which aligns with the security requirement. It then leverages VPC endpoints for accessing AWS services. Gateway VPC endpoints are used for S3 and DynamoDB. Gateway endpoints are free and provide reliable connectivity to S3 and DynamoDB. https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html

An interface VPC endpoint for SQS is used because the EC2 instances need to poll SQS. Gateway endpoints

don't support SQS. Interface endpoints use PrivateLink, allowing private connectivity to SQS over the AWS network.

Option A is incorrect because it places instances in a public subnet, even with public IP assignment disabled. This doesn't completely isolate them, and using an Internet Gateway adds unnecessary complexity.

Option B uses a NAT Gateway. While it provides outbound internet access, it's significantly more expensive than VPC endpoints and unnecessary for accessing S3, DynamoDB, and SQS.

Option D is incorrect because it swaps the endpoint types for SQS and S3/DynamoDB. Gateway endpoints do not work for SQS, while they are the most cost-effective solution for S3 and DynamoDB.

Therefore, option C provides a secure, private, and cost-effective solution by utilizing VPC endpoints for accessing AWS services without traversing the public internet or incurring NAT Gateway charges.

Question: 50

A company has an AWS Direct Connect connection between its on-premises data center in the United States (US) and workloads in the us-east-1 Region. The connection uses a transit VIF to connect the data center to a transit gateway in us-east-1.
The company is opening a new office in Europe with a new on-premises data center in England. A Direct Connect connection will connect the new data center with some workloads that are running in a single VPC in the eu-west-2 Region. The company needs to connect the US data center and us-east-1 with the Europe data center and eu-west-2. A network engineer must establish full connectivity between the data centers and Regions with the lowest possible latency.
How should the network engineer design the network architecture to meet these requirements?

A.Connect the VPC in eu-west-2 with the Europe data center by using a Direct Connect gateway and a private VIF. Associate the transit gateway in us-east-1 with the same Direct Connect gateway. Enable SiteLink for the transit VIF and the private VIF.

B.Connect the VPC in eu-west-2 to a new transit gateway. Connect the Europe data center to the new transit gateway by using a Direct Connect gateway and a new transit VIF. Associate the transit gateway in us-east-1 with the same Direct Connect gateway. Enable SiteLink for both transit VIFs. Peer the two transit gateways.

C.Connect the VPC in eu-west-2 to a new transit gateway. Connect the Europe data center to the new transit gateway by using a Direct Connect gateway and a new transit VIF. Create a new Direct Connect gateway. Associate the transit gateway in us-east-1 with the new Direct Connect gateway. Enable SiteLink for both transit VIFs. Peer the two transit gateways.

D.Connect the VPC in eu-west-2 with the Europe data center by using a Direct Connect gateway and a private VIF. Create a new Direct Connect gateway. Associate the transit gateway in us-east-1 with the new Direct Connect gateway. Enable SiteLink for the transit VIF and the private VIF.

Answer: B

Explanation:

Here's a detailed justification for why option B is the most suitable solution, and why the other options are less ideal, focusing on the lowest latency requirement:

Option B: Correct Solution

Option B proposes connecting the eu-west-2 VPC to a new transit gateway, connecting the Europe data center to this new transit gateway using a Direct Connect gateway and a new transit VIF, associating the transit gateway in us-east-1 with the same Direct Connect gateway, enabling SiteLink for both transit VIFs, and peering the two transit gateways.

Regional Separation & Lowest Latency: Connecting the EU data center to a transit gateway in the EU region (using a Direct Connect gateway and a transit VIF) ensures traffic stays within Europe for communication with

EU resources. This minimizes latency compared to routing traffic back to the US.

Transit Gateway for VPC connectivity: Using a transit gateway allows simple connections between the VPC in eu-west-2 and the Europe data center. This ensures seamless connectivity between these resources.

Direct Connect Gateway Sharing: The crux of this design is that both transit gateways (us-east-1 and eu-west-2) associate with the same Direct Connect gateway. SiteLink is then enabled on both transit VIFs (US and EU).

SiteLink for Direct Connectivity: SiteLink allows direct communication between the on-premises locations connected to the same Direct Connect Gateway, bypassing the need to traverse the AWS backbone for on-premises to on-premises communication. This is crucial for minimizing latency between the US and Europe data centers.

Transit Gateway Peering: Peering the transit gateways enables connectivity between the US and EU regions while using the Direct Connect SiteLink for lowest latency between on-premise locations.

Why other options are not ideal:

Option A: Using a private VIF instead of a transit VIF for the Europe data center would prevent seamless and easy integration with other AWS services in the eu-west-2 Region without more complex routing configurations. Furthermore, without a separate transit gateway in the EU, traffic from the EU data center to the us-east-1 region and US data center would have to go through the US East region.

Option C: Creates a new Direct Connect gateway which adds cost and complexity. Also since it's a new Direct Connect gateway, there is no SiteLink possible.

Option D: Similar to Option A, this uses a private VIF, which has drawbacks for scalability. Also, creating a new Direct Connect gateway does not enable SiteLink.

Authoritative Links:

AWS Direct Connect Gateways:https://docs.aws.amazon.com/directconnect/latest/UserGuide/direct-connect-gateways-intro.html
AWS Transit Gateway:https://docs.aws.amazon.com/vpc/latest/tgw/what-is-transit-gateway.html **AWS Direct Connect SiteLink:**https://aws.amazon.com/about-aws/whats-new/2023/11/aws-direct-connect-sitelink/

Question: 51

A network engineer has deployed an Amazon EC2 instance in a private subnet in a VPC. The VPC has no public subnet. The EC2 instance hosts application code that sends messages to an Amazon Simple Queue Service (Amazon SQS) queue. The subnet has the default network ACL with no modification applied. The EC2 instance has the default security group with no modification applied.
The SQS queue is not receiving messages.
Which of the following are possible causes of this problem? (Choose two.)

A.The EC2 instance is not attached to an IAM role that allows write operations to Amazon SQS.
B.The security group is blocking traffic to the IP address range used by Amazon SQS
C.There is no interface VPC endpoint configured for Amazon SQS
D.The network ACL is blocking return traffic from Amazon SQS
E.There is no route configured in the subnet route table for the IP address range used by Amazon SQS

Answer: AC

Explanation:

Here's a detailed justification for why options A and C are the correct answers, and why the others are incorrect, in the context of the scenario provided.

Why A is correct: The EC2 instance is not attached to an IAM role that allows write operations to Amazon

SQS.

The EC2 instance needs permission to interact with the SQS queue. This is managed through IAM roles. If the EC2 instance doesn't have an IAM role attached, or if the attached role lacks the necessary permissions (specifically, sqs:SendMessage or sqs:* for broader access) to write to the SQS queue, the application code will be unable to send messages, causing the reported problem. IAM roles are the standard and secure way to grant permissions to EC2 instances to access AWS services. Without the appropriate permissions, the application code will likely encounter an "Access Denied" error when attempting to interact with the SQS queue.

Why C is correct: There is no interface VPC endpoint configured for Amazon SQS.

Since the EC2 instance resides in a private subnet without a public subnet, it doesn't have direct access to the internet. To communicate with SQS, which is a public AWS service, a mechanism for private connectivity is needed. An Interface VPC Endpoint for SQS provides this connectivity. It allows traffic to flow from the EC2 instance to SQS without traversing the internet. Without a VPC Endpoint, the EC2 instance would need a NAT Gateway or a NAT instance to access SQS, and even with a NAT Gateway, the proper routing would be necessary. Because the question doesn't mention any NAT setup, the absence of a VPC endpoint directly prevents the EC2 instance from reaching the SQS service.

Why B is incorrect: The security group is blocking traffic to the IP address range used by Amazon SQS.

The default security group allows all outbound traffic. Unless modified, the security group will not block the EC2 instance from sending traffic. While a restrictive security group could block this traffic, the problem description explicitly states the default security group is in use, eliminating this as a likely cause.

Why D is incorrect: The network ACL is blocking return traffic from Amazon SQS.

The default network ACL allows all inbound and outbound traffic. Unless modified, the network ACL will not block the EC2 instance from receiving responses from SQS or from sending the initial request. While a restrictive network ACL could cause issues, the problem description explicitly states the default network ACL is in use, eliminating this as a likely cause.

Why E is incorrect: There is no route configured in the subnet route table for the IP address range used by Amazon SQS

This is related to option C but less direct. Without a VPC Endpoint, a route to a NAT Gateway (or NAT instance) would be required for the private subnet to access the internet and thus SQS. However, the primary issue is the lack of a secure private connection mechanism. The explicit mention of no VPC Endpoint makes option C a more accurate and direct cause. Routing problems would manifest if a VPC endpoint was not used and the NAT configuration was incorrect, but the endpoint absence is the primary problem.

Supporting Links:

IAM Roles for EC2:https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_ec2.html **VPC Endpoints:**https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html **Security Groups:**https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-groups.html **Network ACLs:**https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html

Question: 52

A network engineer needs to standardize a company's approach to centralizing and managing interface VPC endpoints for private communication with AWS services. The company uses AWS Transit Gateway for inter-VPC connectivity between AWS accounts through a hub-and-spoke model. The company's network services team must manage all Amazon Route 53 zones and interface endpoints within a shared services AWS account. The company wants to use this centralized model to provide AWS resources with access to AWS Key Management Service (AWS

KMS) without sending traffic over the public internet.
What should the network engineer do to meet these requirements?

A.In the shared services account, create an interface endpoint for AWS KMS. Modify the interface endpoint by disabling the private DNS name. Create a private hosted zone in the shared services account with an alias record that points to the interface endpoint. Associate the private hosted zone with the spoke VPCs in each AWS account.

B.In the shared services account, create an interface endpoint for AWS KMS. Modify the interface endpoint by disabling the private DNS name. Create a private hosted zone in each spoke AWS account with an alias record that points to the interface endpoint. Associate each private hosted zone with the shared services AWS account.

C.In each spoke AWS account, create an interface endpoint for AWS KMS. Modify each interface endpoint by disabling the private DNS name. Create a private hosted zone in each spoke AWS account with an alias record that points to each interface endpoint. Associate each private hosted zone with the shared services AWS account.

D.In each spoke AWS account, create an interface endpoint for AWS KMS. Modify each interface endpoint by disabling the private DNS name. Create a private hosted zone in the shared services account with an alias record that points to each interface endpoint. Associate the private hosted zone with the spoke VPCs in each AWS account.

Answer: A

Explanation:

The correct answer is A because it aligns with the principles of centralized management, private connectivity, and DNS resolution within the AWS environment using Transit Gateway and Route 53.

Here's a breakdown:

Centralized Interface Endpoint: Creating the KMS interface endpoint in the shared services account centralizes management, which is a key requirement. This allows the network services team to control access to KMS from a single location.

Private Connectivity: Using an interface endpoint ensures that traffic to KMS does not traverse the public internet, meeting the requirement for secure private communication.

Disabling Private DNS Name: Disabling the private DNS name on the interface endpoint prevents automatic resolution within the VPC where the endpoint is created. This is necessary because we want to manage DNS centrally in the shared services account.

Private Hosted Zone in Shared Services: Creating a private hosted zone in the shared services account provides a central DNS management point for the KMS endpoint.

Alias Record: The alias record within the private hosted zone maps the standard KMS endpoint name (e.g., kms.us-east-1.amazonaws.com) to the interface endpoint's DNS name or IP addresses.

Associating with Spoke VPCs: Associating the private hosted zone with the spoke VPCs allows resources in those VPCs to resolve the KMS endpoint name to the private IP addresses of the interface endpoint. This is how the spoke VPCs will access KMS through the interface endpoint.

Transit Gateway: Although not explicitly used in DNS resolution, the Transit Gateway provides the underlying network connectivity between the spoke VPCs and the shared services VPC, ensuring that traffic to the interface endpoint stays within the AWS network.

Options B, C, and D are incorrect for the following reasons:

B: Creating private hosted zones in each spoke account defeats the purpose of centralized management. **C:** Creating interface endpoints in each spoke account also defeats the purpose of centralized management and increases operational overhead.

D: Creating interface endpoints in each spoke and centralizing the DNS zone in the shared service account is not suitable as it requires to keep track of all endpoints, as the DNS needs to map to the individual IPs of each created interface endpoint, which defeats the purpose of the centralized management.

Authoritative Links:

AWS PrivateLink:https://aws.amazon.com/privatelink/
Interface VPC Endpoints:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints-interface.html
Amazon Route 53 Private Hosted Zones:
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/private-hosted-zones.html
AWS Transit Gateway:https://aws.amazon.com/transit-gateway/

Question: 53

A development team is building a new web application in the AWS Cloud. The main company domain, example.com, is currently hosted in an Amazon Route 53 public hosted zone in one of the company's production AWS accounts.
The developers want to test the web application in the company's staging AWS account by using publicly resolvable subdomains under the example.com domain with the ability to create and delete DNS records as needed. Developers have full access to Route 53 hosted zones within the staging account, but they are prohibited from accessing resources in any of the production AWS accounts.
Which combination of steps should a network engineer take to allow the developers to create records under the example com domain? (Choose two.)

A.Create a public hosted zone for example com in the staging account

B.Create a staging example.com NS record in the example.com domain. Populate the value with the name servers from the staging.example.com domain. Set the routing policy type to simple routing.

C.Create a private hosted zone for staging example com in the staging account.

D.Create an example com NS record in the staging example.com domain. Populate the value with the name servers from the example.com domain. Set the routing policy type to simple routing.

E.Create a public hosted zone for staging.example.com in the staging account.

Answer: BE

Explanation:

Let's break down why options B and E are the correct choices to enable subdomain delegation from the production AWS account to the staging AWS account for DNS management, focusing on Route 53 and DNS delegation concepts.

Option E: "Create a public hosted zone for staging.example.com in the staging account" is crucial. The staging account needs its own dedicated hosted zone to manage DNS records specifically for the staging.example.com subdomain. This isolates the staging environment's DNS configuration from the production environment and allows the developers in the staging account to have full control. Without this, the staging environment would not be able to manage its DNS records.

Option B: "Create a staging example.com NS record in the example.com domain. Populate the value with the name servers from the staging.example.com domain. Set the routing policy type to simple routing." This step creates a delegation record within the production example.com hosted zone that points to the authoritative name servers for the staging.example.com hosted zone created in the staging account. An NS record is specifically used for delegating a subdomain to a different set of name servers (and therefore, a different hosted zone). The "simple routing" policy is suitable as the primary purpose is delegation. This delegation is what allows DNS queries for staging.example.com and its subdomains to be correctly resolved by the name servers in the staging account.

Option A is incorrect because creating another public hosted zone for example.com in the staging account would lead to DNS conflicts. There can only be one authoritative hosted zone for a domain (in this case, in the production account). Option C is incorrect because a public hosted zone is needed for publicly resolvable subdomains as stated in the problem. A private hosted zone would only resolve within specified VPCs. Option D is reversed; the NS record needs to be in the parent domain (example.com) pointing to the name servers of the subdomain (staging.example.com).

In summary, to delegate control of a subdomain, you must create a hosted zone for the subdomain, then create an NS record for that subdomain in the parent zone, pointing to the subdomain's name servers.

Supporting resources:

AWS Route 53 Documentation: https://docs.aws.amazon.com/route53/
Delegating Subdomains: https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/dns-configuring-dns-delegation.html
NS Record: https://en.wikipedia.org/wiki/NS_record

Question: 54

A company plans to deploy a two-tier web application to a new VPC in a single AWS Region. The company has configured the VPC with an internet gateway and four subnets. Two of the subnets are public and have default routes that point to the internet gateway. Two of the subnets are private and share a route table that does not have a default route.
The application will run on a set of Amazon EC2 instances that will be deployed behind an external Application Load Balancer. The EC2 instances must not be directly accessible from the internet. The application will use an Amazon S3 bucket in the same Region to store data. The application will invoke S3 GET API operations and S3 PUT API operations from the EC2 instances. A network engineer must design a VPC architecture that minimizes data transfer cost.
Which solution will meet these requirements?

A.Deploy the EC2 instances in the public subnets. Create an S3 interface endpoint in the VPC. Modify the application configuration to use the S3 endpoint-specific DNS hostname.

B.Deploy the EC2 instances in the private subnets. Create a NAT gateway in the VPC. Create default routes in the private subnets to the NAT gateway. Connect to Amazon S3 by using the NAT gateway.

C.Deploy the EC2 instances in the private subnets. Create an S3 gateway endpoint in the VPSpecify die route table of the private subnets during endpoint creation to create routes to Amazon S3.

D.Deploy the EC2 instances in the private subnets. Create an S3 interface endpoint in the VPC. Modify the application configuration to use the S3 endpoint-specific DNS hostname.

Answer: C

Explanation:

The correct answer is C because it provides the most cost-effective and secure way for EC2 instances in private subnets to access S3, aligning with the requirements of the scenario.

Here's a detailed justification:

EC2 instances in private subnets: Deploying EC2 instances in private subnets ensures they are not directly accessible from the internet, fulfilling the security requirement.

S3 Gateway Endpoint: A Gateway Endpoint for S3 is the most cost-effective solution because it's free. Interface Endpoints incur data processing charges.

Gateway Endpoint Functionality: Gateway Endpoints work by adding a route to the specified route table. Traffic destined for S3 is routed directly from the private subnets to S3 over the AWS backbone network, bypassing the internet. This fulfills the requirement of minimizing data transfer costs, as traffic to S3 doesn't go through a NAT gateway which incurs costs.

Route Table Association: Specifying the route table of the private subnets during endpoint creation ensures that the route to S3 is automatically added to those route tables. EC2 instances will then automatically use this route to communicate with S3.

Why other options are incorrect:

A: Placing instances in public subnets violates the requirement that they should not be directly accessible from the Internet.
B: Using a NAT Gateway to access S3 is more expensive than using a Gateway Endpoint as NAT Gateways charge for data processed.
D: While an S3 interface endpoint provides secure access to S3, it incurs data processing charges. A Gateway Endpoint is more cost-effective for S3 access within the same region.

In summary, answer C provides a solution that satisfies all requirements by deploying EC2 instances in private subnets, offering private connectivity to S3 through a gateway endpoint, and optimizing cost by utilizing the free Gateway Endpoint for S3.

Relevant links for further research:

VPC Endpoints
Gateway Endpoints
Interface Endpoints

Question: 55

A company has two AWS accounts one for Production and one for Connectivity. A network engineer needs to connect the Production account VPC to a transit gateway in the Connectivity account. The feature to auto accept shared attachments is not enabled on the transit gateway.
Which set of steps should the network engineer follow in each AWS account to meet these requirements?

A.1. In the Production account: Create a resource share in AWS Resource Access Manager for the transit gateway. Provide the Connectivity account ID. Enable the feature to allow external accounts
2. In the Connectivity account: Accept the resource.
3. In the Connectivity account: Create an attachment to the VPC subnets.
4. In the Production account: Accept the attachment. Associate a route table with the attachment.

B.1. In the Production account: Create a resource share in AWS Resource Access Manager for the VPC subnets. Provide the Connectivity account ID. Enable the feature to allow external accounts.
2. In the Connectivity account: Accept the resource.
3. In the Production account: Create an attachment on the transit gateway to the VPC subnets. 4. In the Connectivity account: Accept the attachment. Associate a route table with the attachment.

C.1. In the Connectivity account: Create a resource share in AWS Resource Access Manager for the VPC subnets. Provide the Production account ID. Enable the feature to allow external accounts.
2. In the Production account: Accept the resource.
3. In the Connectivity account: Create an attachment on the transit gateway to the VPC subnets. 4. In the Production account: Accept the attachment. Associate a route table with the attachment.

D.1. In the Connectivity account: Create a resource share in AWS Resource Access Manager for the transit gateway. Provide the Production account ID Enable the feature to allow external accounts.
2. In the Production account: Accept the resource.
3. In the Production account: Create an attachment to the VPC subnets.
4. In the Connectivity account: Accept the attachment. Associate a route table with the attachment.

Answer: D

Explanation:

The correct answer is D because it outlines the necessary steps for connecting a Production account VPC to a Transit Gateway in a Connectivity account without auto-accept enabled, using AWS Resource Access Manager (RAM).

Here's a breakdown of why answer D is correct and why the other options are incorrect:

Step 1 (Connectivity Account): The Connectivity account owns the Transit Gateway. Therefore, the resource share needs to be created here, sharing the Transit Gateway with the Production account using RAM. Enabling the feature to allow external accounts is essential when sharing resources across AWS accounts.

https://docs.aws.amazon.com/ram/latest/userguide/getting-started-sharing.html

Step 2 (Production Account): The Production account needs to accept the resource share invitation to gain access to the shared Transit Gateway.

Step 3 (Production Account): The Production account creates the Transit Gateway attachment from its VPC subnets to the Transit Gateway that was shared. Since auto-accept is disabled, an explicit attachment needs to be made. The attachment originates from the VPC wanting the connection.
https://docs.aws.amazon.com/vpc/latest/tgw/tgw-attachments.html

Step 4 (Connectivity Account): Because auto-accept is disabled, the Connectivity account (the Transit Gateway owner) must explicitly accept the attachment request initiated by the Production account. Finally, the Transit Gateway route table must be updated to route traffic through the attachment.

Why other options are incorrect:

Option A: Incorrect because the Production account cannot share the transit gateway. The Connectivity account, which is where the Transit Gateway exists, is responsible for sharing the resource. Also, the association with the attachment comes from the Connectivity account, not Production, at the beginning.

Option B: Incorrect because it attempts to share VPC subnets using RAM, which isn't directly how Transit Gateway attachments work across accounts. The Production account cannot directly create the attachment on the Transit Gateway. Also, the attachment is initiated from the VPC.

Option C: Incorrect because it attempts to share VPC subnets using RAM. Further, it incorrectly assigns the Transit Gateway attachment creation to the Connectivity account.

In summary, option D accurately describes the process of sharing the Transit Gateway from the Connectivity account to the Production account, creating the attachment from the Production VPC to the Transit Gateway, and then accepting that attachment on the Connectivity account side. This approach leverages AWS RAM for secure cross-account resource sharing and adheres to the requirement of manual attachment acceptance.

Question: 56

A company is running multiple workloads on Amazon EC2 instances in public subnets. In a recent incident, an attacker exploited an application vulnerability on one of the EC2 instances to gain access to the instance. The company fixed the application and launched a replacement EC2 instance that contains the updated application. The attacker used the compromised application to spread malware over the internet. The company became aware of the compromise through a notification from AWS. The company needs the ability to identify when an application that is deployed on an EC2 instance is spreading malware.
Which solution will meet this requirement with the LEAST operational effort?

A.Use Amazon GuardDuty to analyze traffic patterns by inspecting DNS requests and VPC flow logs.

B.Use Amazon GuardDuty to deploy AWS managed decoy systems that are equipped with the most recent malware signatures.

C.Set up a Gateway Load Balancer. Run an intrusion detection system (IDS) appliance from AWS Marketplace on Amazon EC2 for traffic inspection.

D.Configure Amazon Inspector to perform deep packet inspection of outgoing traffic.

Answer: A

Explanation:

The best solution is A: Use Amazon GuardDuty to analyze traffic patterns by inspecting DNS requests and VPC flow logs.

Here's why:

GuardDuty is designed for threat detection: GuardDuty is a threat detection service that continuously monitors your AWS accounts and workloads for malicious activity and anomalous behavior. It leverages machine learning, anomaly detection, and integrated threat intelligence feeds.

Network Traffic Analysis: GuardDuty analyzes VPC Flow Logs, DNS logs, and CloudTrail event logs to identify suspicious patterns. By inspecting DNS requests and VPC flow logs, GuardDuty can detect communication with known malicious domains or unusual traffic patterns indicative of malware spreading.

Least Operational Effort: GuardDuty is a managed service, meaning AWS handles the underlying infrastructure and maintenance. This significantly reduces the operational overhead compared to setting up and managing your own IDS solution. You simply enable GuardDuty, and it starts monitoring your environment.

No agents required: GuardDuty doesn't require installing agents on EC2 instances, simplifying deployment and reducing the performance impact on your workloads.

Decoy Systems are not required: While option B uses GuardDuty with decoy systems, this approach is generally used to attract and trap attackers, providing insights into their techniques. It does not directly help identify malware spreading from EC2 instances.

Alternatives have higher operational overhead: Option C, using a Gateway Load Balancer and an IDS appliance, involves significantly more setup and management. You need to select, configure, and maintain the IDS appliance, as well as manage the Gateway Load Balancer. Option D, using Amazon Inspector for deep packet inspection, is not primarily designed for real-time network traffic analysis for malware spreading.

Cost-Effective: GuardDuty is generally a cost-effective solution for threat detection, especially when considering the reduced operational overhead compared to self-managed solutions.

Authoritative Links:

Amazon GuardDuty: https://aws.amazon.com/guardduty/
VPC Flow Logs: https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html

Question: 57

A company deploys a new web application on Amazon EC2 instances. The application runs in private subnets in three Availability Zones behind an Application Load Balancer (ALB). Security auditors require encryption of all connections. The company uses Amazon Route 53 for DNS and uses AWS Certificate Manager (ACM) to automate SSL/TLS certificate provisioning. SSL/TLS connections are terminated on the ALB.
The company tests the application with a single EC2 instance and does not observe any problems. However, after production deployment, users report that they can log in but that they cannot use the application. Every new web request restarts the login process.
What should a network engineer do to resolve this issue?

A.Modify the ALB listener configuration. Edit the rule that forwards traffic to the target group. Change the rule to enable group-level stickiness. Set the duration to the maximum application session length.

B.Replace the ALB with a Network Load Balancer. Create a TLS listener. Create a new target group with the protocol type set to TLS Register the EC2 instances. Modify the target group configuration by enabling the stickiness attribute.

C.Modify the ALB target group configuration by enabling the stickiness attribute. Use an application-based cookie. Set the duration to the maximum application session length.

D.Remove the ALB. Create an Amazon Route 53 rule with a failover routing policy for the application name. Configure ACM to issue certificates for each EC2 instance.

Answer: C

Explanation:

The correct answer is **C. Modify the ALB target group configuration by enabling the stickiness attribute. Use an application-based cookie. Set the duration to the maximum application session length.**

Here's why:

The problem described suggests a session management issue. Users can log in, but every subsequent request restarts the login process. This indicates that the user's session is not being maintained across multiple requests. In a load-balanced environment like this, a user's requests can be routed to different EC2 instances behind the ALB. If each instance treats the user as a new user with each request, the session is lost.

Application Load Balancers (ALBs) offer a feature called "stickiness" or "session affinity" to address this problem. Enabling stickiness ensures that all requests from a specific user are routed to the same EC2 instance for the duration of the session.

Option C directly addresses this by suggesting enabling stickiness at the target group level. By configuring an application-based cookie, the ALB inserts a cookie in the user's browser after the first request. This cookie contains information about the instance that initially served the request. Subsequent requests from the same user will include this cookie, and the ALB will use it to route those requests to the same instance, maintaining session continuity. Setting the duration to the maximum application session length ensures the stickiness persists for the entire duration the user interacts with the application.

Why the other options are incorrect:

A. Modify the ALB listener configuration... enable group-level stickiness: While this option also aims to enable stickiness, "group-level stickiness" is not a standard term associated with ALBs. The ALB stickiness is configured at the target group level, as described in option C. Using target group stickiness with an application-based cookie is the standard and more direct approach.

B. Replace the ALB with a Network Load Balancer: NLBs operate at layer 4 (TCP), while ALBs operate at layer 7 (application layer). NLBs are designed for high throughput and low latency, but they don't inherently handle application-level session stickiness based on cookies. While NLBs support source IP based stickiness which is a very coarse mechanism, converting the ALB and associated configurations to NLB won't resolve the actual application's need for managing user sessions using cookies. Introducing TLS at the NLB level doesn't address the root cause either; the fundamental problem lies in session management.

D. Remove the ALB. Create an Amazon Route 53 rule...: This is not the correct solution. Removing the load balancer removes the benefits of traffic distribution and high availability. Configuring Route 53 failover routing policies is for disaster recovery scenarios, not session management. ACM certificates are already being used with the ALB for SSL/TLS termination, so issuing certificates to individual instances is not necessary and makes management more complex.

Supporting Documentation:

Application Load Balancers - Target Groups - Attributes (Stickiness):
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/tutorial-get-started-application-load-balancer.html#modify-target-group-attributes

Question: 58

A company recently migrated its Amazon EC2 instances to VPC private subnets to satisfy a security compliance requirement. The EC2 instances now use a NAT gateway for internet access. After the migration, some long-running database queries from private EC2 instances to a publicly accessible third-party database no longer receive responses. The database query logs reveal that the queries successfully completed after 7 minutes but that the client EC2 instances never received the response.
Which configuration change should a network engineer implement to resolve this issue?

A.Configure the NAT gateway timeout to allow connections for up to 600 seconds.

B.Enable enhanced networking on the client EC2 instances.

C.Enable TCP keepalive on the client EC2 instances with a value of less than 300 seconds.

D.Close idle TCP connections through the NAT gateway.

Answer: C

Explanation:

Here's a detailed justification for why option C is the correct answer:

The problem describes a scenario where long-running database queries are completing successfully on the server side, but the client EC2 instances in private subnets, using a NAT gateway for internet access, are not receiving the responses. This suggests an idle timeout issue with the connection tracking performed by the NAT gateway. NAT gateways, by default, have TCP connection tracking. If a TCP connection remains idle for a certain period (the idle timeout), the NAT gateway will drop the connection and remove its state from the connection tracking table.

The default idle timeout for a TCP connection through an AWS NAT gateway is 350 seconds (approximately 5 minutes 50 seconds). Since the database queries are completing successfully in 7 minutes (420 seconds), the connections are being dropped by the NAT gateway before the responses can be delivered back to the client. The client instances don't know the connection has been dropped and keep waiting.

Option C, enabling TCP keepalive on the client EC2 instances with a value of less than 300 seconds, is the correct solution because it sends periodic TCP keepalive packets. These packets are small probes sent by the client to the server (or vice versa) to keep the connection active and prevent the NAT gateway from considering the connection idle. By setting the keepalive interval to less than 300 seconds, the NAT gateway will receive these packets and reset the idle timeout counter, ensuring that the connection remains active for the duration of the 7-minute database query.

Option A is incorrect because you cannot configure the NAT gateway timeout. It's a managed service and the timeout values are fixed. While increasing the timeout would solve the problem, it's not a configurable option.

Option B, enabling enhanced networking, improves network performance (higher throughput, lower latency, and lower jitter). While beneficial overall, it doesn't address the specific problem of the NAT gateway dropping idle connections.

Option D, closing idle TCP connections through the NAT gateway, is the opposite of what is needed. It would exacerbate the problem by prematurely terminating connections.

In summary, TCP keepalive mechanisms provide a way to maintain the connection alive by sending periodic probes. Since NAT gateway timeouts cannot be modified, client-side TCP keepalive configurations are crucial for long-running connections through the gateway.

Authoritative Links:

AWS NAT Gateway:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html (Specifically, review information about connection timeouts.)

TCP Keepalive HOWTO:http://tldp.org/HOWTO/TCP-Keepalive-HOWTO/index.html (General explanation of TCP Keepalive)

Question: 59

A company uses AWS Direct Connect to connect its corporate network to multiple VPCs in the same AWS account

and the same AWS Region. Each VPC uses its own private VIF and its own virtual LAN on the Direct Connect connection. The company has grown and will soon surpass the limit of VPCs and private VIFs for each connection. What is the MOST scalable way to add VPCs with on-premises connectivity?

A.Provision a new Direct Connect connection to handle the additional VPCs. Use the new connection to connect additional VPCs.

B.Create virtual private gateways for each VPC that is over the service quota. Use AWS Site-to-Site VPN to connect the virtual private gateways to the corporate network.

C.Create a Direct Connect gateway, and add virtual private gateway associations to the VPCs. Configure a private VIF to connect to the corporate network.

D.Create a transit gateway, and attach the VPCs. Create a Direct Connect gateway, and associate it with the transit gateway. Create a transit VIF to the Direct Connect gateway.

Answer: D

Explanation:

The question asks for the most scalable way to connect an on-premises network to multiple VPCs in the same region, given that the current Direct Connect setup is hitting limits. Let's analyze why the chosen answer (D) is the best option:

Option D: This solution leverages Transit Gateway (TGW) and Direct Connect Gateway (DXGW). TGW acts as a central hub, allowing VPCs to connect to it. The DXGW then connects the TGW to the Direct Connect connection. This avoids the limitations of one private VIF per VPC. As more VPCs are needed, they can be attached to the TGW without exhausting Direct Connect resources. A single transit VIF is used, regardless of the number of connected VPCs. This approach allows for transitive routing between VPCs and the on-premises network.

Option A: Provisioning a new Direct Connect connection only delays the problem. The company will eventually hit the limits again as they add more VPCs. It's not a scalable solution.

Option B: Using Site-to-Site VPNs would create a complex mesh of VPN connections. Each VPC would need its own VPN connection, which would be hard to manage and potentially less performant than Direct Connect. **Option C:** Creating a Direct Connect gateway is a good start, but it only allows you to associate VPCs with the Direct Connect connection through Virtual Private Gateways (VGWs). Although a DXGW can be associated with multiple VGWs, this option does not provide scalable transitive connectivity between VPCs. Each VPC still needs its own VGW.

Therefore, option D is the most scalable solution because it uses Transit Gateway to act as a hub for VPCs, and Direct Connect Gateway to efficiently connect the TGW to the Direct Connect connection, minimizing the number of virtual interfaces required and centralizing the routing.

Supporting links:

AWS Transit Gateway: https://aws.amazon.com/transit-gateway/
AWS Direct Connect Gateway: https://aws.amazon.com/directconnect/dx-gateway/

Question: 60

A network engineer is designing a hybrid architecture that uses a 1 Gbps AWS Direct Connect connection between the company's data center and two AWS Regions: us-east-1 and eu-west-1. The VPCs in us-east-1 are connected by a transit gateway and need to access several on-premises databases. According to company policy, only one VPC in eu-west-1 can be connected to one on-premises server. The on-premises network segments the traffic between the databases and the server. How should the network engineer set up the Direct Connect connection to meet these requirements?

A.Create one hosted connection. Use a transit VIF to connect to the transit gateway in us-east-1. Use a private VIF to connect to the VPC in eu-west-1. Use one Direct. Connect gateway for both VIFs to route from the Direct Connect locations to the corresponding AWS Region along the path that has the lowest latency.

B.Create one hosted connection. Use a transit VIF to connect to the transit gateway in us-east-1. Use a private VIF to connect to the VPC in eu-west-1. Use two Direct Connect gateways, one for each VIF, to route from the Direct Connect locations to the corresponding AWS Region along the path that has the lowest latency.

C.Create one dedicated connection. Use a transit VIF to connect to the transit gateway in us-east-1. Use a private VIF to connect to the VPC in eu-west-1. Use one Direct Connect gateway for both VIFs to route from the Direct Connect locations to the corresponding AWS Region along the path that has the lowest latency.

D.Create one dedicated connection. Use a transit VIF to connect to the transit gateway in us-east-1. Use a private VIF to connect to the VPC in eu-west-1. Use two Direct Connect gateways, one for each VIF, to route from the Direct Connect locations to the corresponding AWS Region along the path that has the lowest latency.

Answer: D

Explanation:

The correct answer is **D**, which involves a dedicated connection, a transit VIF for us-east-1, a private VIF for eu-west-1, and two Direct Connect gateways. Let's break down why:

1. **Dedicated Connection:** The question doesn't explicitly state a need or lack of need for a dedicated vs. hosted connection. However, it implies the organization already has an on-premises datacenter that needs secure, consistent connectivity. A dedicated connection provides more control and consistent performance compared to a hosted connection which can be impacted by the service provider.

2. **Transit VIF for us-east-1:** The requirement to connect to a transit gateway (TGW) in us-east-1 necessitates using a transit VIF. Transit VIFs are specifically designed for connecting to transit gateways, enabling connectivity between multiple VPCs connected to the TGW and the on-premises network.

3. **Private VIF for eu-west-1:** Since only one VPC in eu-west-1 needs to connect to one on-premises server, a private VIF is the appropriate choice. Private VIFs establish direct connectivity between the Direct Connect connection and a specific VPC.

4. **Two Direct Connect Gateways:** This is the crucial distinction. The requirement to connect to two separate AWS Regions necessitates two Direct Connect Gateways (DXGW). A single DXGW can only be associated with VIFs in a single AWS Region. The DXGWs enable routing between the Direct Connect connection and the corresponding AWS Regions (us-east-1 and eu-west-1), allowing the traffic to reach the transit gateway and the VPC in eu-west-1.

5. **Lowest Latency Routing:** Each DXGW independently handles traffic routing. While the question mentions lowest latency, the DXGW automatically uses BGP routing to determine the best path to the associated AWS Region, so latency is automatically a consideration. There's no configuration needed to explicitly set this.

In summary, using a dedicated connection, a transit VIF to the us-east-1 TGW, a private VIF to eu-west-1, and separate DXGWs for each region is essential for meeting the requirements for connecting to two distinct AWS Regions and enabling connectivity between the on-premises network and the VPCs in those regions while adhering to the one-to-one VPC to on-premises server restriction in eu-west-1.

Authoritative Links:

AWS Direct Connect Gateways: https://docs.aws.amazon.com/directconnect/latest/UserGuide/direct-connect-gateways-intro.html
AWS Direct Connect Virtual Interfaces: https://docs.aws.amazon.com/directconnect/latest/UserGuide/WorkingWithVirtualInterfaces.html AWS Transit Gateway: https://docs.aws.amazon.com/vpc/latest/tgw/what-is-transit-gateway.html

Question: 61

A company has deployed an application in a VPC that uses a NAT gateway for outbound traffic to the internet. A network engineer notices a large quantity of suspicious network traffic that is traveling from the VPC over the internet to IP addresses that are included on a deny list. The network engineer must implement a solution to determine which AWS resources are generating the suspicious traffic. The solution must minimize cost and administrative overhead.
Which solution will meet these requirements?

A.Launch an Amazon EC2 instance in the VPC. Use Traffic Mirroring by specifying the NAT gateway as the source and the EC2 instance as the destination. Analyze the captured traffic by using open-source tools to identify the AWS resources that are generating the suspicious traffic.

B.Use VPC flow logs. Launch a security information and event management (SIEM) solution in the VPC. Configure the SIEM solution to ingest the VPC flow logs. Run queries on the SIEM solution to identify the AWS resources that are generating the suspicious traffic.

C.Use VPC flow logs. Publish the flow logs to a log group in Amazon CloudWatch Logs. Use CloudWatch Logs Insights to query the flow logs to identify the AWS resources that are generating the suspicious traffic.

D.Configure the VPC to stream the network traffic directly to an Amazon Kinesis data stream. Send the data from the Kinesis data stream to an Amazon Kinesis Data Firehose delivery stream to store the data in Amazon S3. Use Amazon Athena to query the data to identify the AWS resources that are generating the suspicious traffic.

Answer: C

Explanation:

The best solution for identifying the source of suspicious traffic from a VPC, minimizing cost and administrative overhead, is to utilize VPC Flow Logs published to CloudWatch Logs and queried via CloudWatch Logs Insights.

Here's why:

VPC Flow Logs: Capture information about the IP traffic going to and from network interfaces in a VPC. This includes source and destination IP addresses, ports, and the action taken (ACCEPT or REJECT).
(https://docs.aws.amazon.com/vpc/latest/userguide/vpc-flow-logs.html)
Cost Efficiency: VPC Flow Logs incur costs based on the data published to CloudWatch Logs. CloudWatch Logs Insights charges based on the amount of data scanned during queries, making it a pay-as-you-go model.

Administrative Overhead: Setting up VPC Flow Logs and configuring CloudWatch Logs Insights queries is relatively straightforward, requiring minimal administrative effort.

CloudWatch Logs Integration: Publishing to CloudWatch Logs provides a centralized location for log data and seamless integration with CloudWatch Logs Insights.

CloudWatch Logs Insights: Provides a purpose-built query language to analyze log data. This allows the network engineer to quickly filter and identify AWS resources generating traffic to the deny-listed IP addresses, using source IP addresses and network interface IDs.
(https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AnalyzingLogData.html)

The other options are less suitable:

A (Traffic Mirroring): Traffic Mirroring is powerful but more complex to set up and requires deploying and managing an EC2 instance. This adds administrative overhead and EC2 instance costs.

B (SIEM Solution): A SIEM solution is comprehensive but typically more expensive and requires significant setup and configuration, increasing both cost and administrative overhead.

D (Kinesis/Athena): This solution involves multiple services (Kinesis Data Streams, Kinesis Data Firehose, S3, Athena) and data transformation. This increases complexity, management overhead, and overall cost compared to using CloudWatch Logs and CloudWatch Logs Insights. Specifically, Kinesis stream is for real time ingestion, which is overkill for this usecase.

Question: 62

A company has its production VPC (VPC-A) in the eu-west-1 Region in Account 1. VPC-A is attached to a transit gateway (TGW-A) that is connected to an on-premises data center in Dublin, Ireland, by an AWS Direct Connect transit VIF that is configured for an AWS Direct Connect gateway. The company also has a staging VPC (VPC-B) that is attached to another transit gateway (TGW-B) in the eu-west-2 Region in Account 2.
A network engineer must implement connectivity between VPC-B and the on-premises data center in Dublin.
Which solutions will meet these requirements? (Choose two.)

A.Configure inter-Region VPC peering between VPC-A and VPC-B. Add the required VPC peering routes. Add the VPC-B CIDR block in the allowed prefixes on the Direct Connect gateway association.

B.Associate TGW-B with the Direct Connect gateway. Advertise the VPC-B CIDR block under the allowed prefixes.

C.Configure another transit VIF on the Direct Connect connection and associate TGW-B. Advertise the VPC-B CIDR block under the allowed prefixes.

D.Configure inter-Region transit gateway peering between TGW-A and TGW-B. Add the peering routes in the transit gateway route tables. Add both the VPC-A and the VPC-B CIDR block under the allowed prefix list in the Direct Connect gateway association.

E.Configure an AWS Site-to-Site VPN connection over the transit VIF to TGW-B as a VPN attachment.

Answer: BD

Explanation:

The correct answer is BD. Here's why:

B: Associate TGW-B with the Direct Connect gateway. Advertise the VPC-B CIDR block under the allowed prefixes. This solution directly extends the Direct Connect connection's reach to TGW-B. By associating TGW-B with the Direct Connect gateway and advertising the VPC-B CIDR, traffic from the on-premises data center can be routed to VPC-B through the Direct Connect connection and the transit gateway. This is efficient as it leverages the existing Direct Connect infrastructure.
https://docs.aws.amazon.com/directconnect/latest/UserGuide/associate-tgw.html

D: Configure inter-Region transit gateway peering between TGW-A and TGW-B. Add the peering routes in the transit gateway route tables. Add both the VPC-A and the VPC-B CIDR block under the allowed prefix list in the Direct Connect gateway association. This option establishes a connection between TGW-A and TGW-B, enabling routing between the on-premises data center (connected to TGW-A via Direct Connect) and VPC-B. Crucially, the Direct Connect gateway must allow the VPC-B CIDR so that traffic destined for VPC-B can be routed through the Direct Connect connection. This ensures a complete path from on-premises to VPC-B. Adding VPC-A to the allowed prefix list is necessary as TGW-A and TGW-B must communicate.
https://docs.aws.amazon.com/vpc/latest/tgw/tgw-peering.html

Let's analyze the incorrect answers:

A: Configure inter-Region VPC peering between VPC-A and VPC-B. Add the required VPC peering routes. Add the VPC-B CIDR block in the allowed prefixes on the Direct Connect gateway association. VPC peering doesn't automatically provide transitive routing. Traffic from on-premises to VPC-B wouldn't traverse VPC-A using VPC peering alone. It requires a TGW or VPN to extend the connection through VPC-A to VPC-B.

C: Configure another transit VIF on the Direct Connect connection and associate TGW-B. Advertise the VPC-B CIDR block under the allowed prefixes. While technically possible, adding another transit VIF is generally unnecessary when you can associate TGW-B directly with the existing Direct Connect gateway, as in option B. Adding another VIF increases complexity without significant benefit.

E: Configure an AWS Site-to-Site VPN connection over the transit VIF to TGW-B as a VPN attachment. This solution introduces unnecessary complexity. A VPN over Direct Connect would add another layer of overhead and isn't the most direct way to achieve connectivity. The other options leverage the existing Direct Connect

and transit gateway infrastructure more efficiently.

Question: 63

A company's network engineer is designing a hybrid DNS solution for an AWS Cloud workload. Individual teams want to manage their own DNS hostnames for their applications in their development environment. The solution must integrate the application-specific hostnames with the centrally managed DNS hostnames from the on-premises network and must provide bidirectional name resolution. The solution also must minimize management overhead.
Which combination of steps should the network engineer take to meet these requirements? (Choose three.)

A.Use an Amazon Route 53 Resolver inbound endpoint.
B.Modify the DHCP options set by setting a custom DNS server value.
C.Use an Amazon Route 53 Resolver outbound endpoint.
D.Create DNS proxy servers.
E.Create Amazon Route 53 private hosted zones.
F.Set up a zone transfer between Amazon Route 53 and the on-premises DNS.

Answer: ACE

Explanation:

The requirement is a hybrid DNS solution that allows teams to manage their application-specific hostnames in the development environment while integrating with the centrally managed on-premises DNS. The solution needs bidirectional name resolution and minimized overhead.

A. Use an Amazon Route 53 Resolver inbound endpoint: This is correct. An inbound endpoint enables on-premises DNS servers to forward queries for your AWS private hosted zones to Route 53. This provides the "bidirectional name resolution" by allowing on-premises systems to resolve names hosted in AWS.
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver.html

C. Use an Amazon Route 53 Resolver outbound endpoint: This is also correct. An outbound endpoint allows your VPC resources to query your on-premises DNS servers. This provides the other direction of "bidirectional name resolution" by allowing AWS systems to resolve names hosted on-premises. By forwarding unresolved queries from your VPC to the on-premises DNS infrastructure, it allows seamless integration between the AWS environment and the existing on-premises DNS zones.
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver.html

E. Create Amazon Route 53 private hosted zones: This option is crucial. Private hosted zones are used to host DNS records within a VPC. By creating these zones, the individual teams can manage their application-specific hostnames for their applications in the development environment. The integration with inbound and outbound resolvers then ensures that these hostnames are resolvable both from within AWS and from the on-premises network. This also satisfies the requirement to let individual teams manage their own DNS records.
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/private-hosted-zones.html

Why other options are incorrect:

B. Modify the DHCP options set by setting a custom DNS server value: While DHCP options configure DNS servers for EC2 instances, this would override the default AmazonProvidedDNS and not allow resolution of public AWS services. It also doesn't directly facilitate bidirectional name resolution between on-premises and AWS.
D. Create DNS proxy servers: While functional, setting up and maintaining DNS proxy servers adds significant overhead compared to using Route 53 Resolver endpoints, which are managed services. Using Route 53 resolvers minimizes management overhead, which is a specific requirement.

F. Set up a zone transfer between Amazon Route 53 and the on-premises DNS: Zone transfers are complex and can be less efficient than using resolver endpoints for bidirectional name resolution. Resolver endpoints provide a simpler, more scalable, and managed solution. Zone transfers also generally involve transferring the entire zone, making granular team-based DNS delegation more difficult.

Question: 64

A company hosts a web application on Amazon EC2 instances behind an Application Load Balancer (ALB). The ALB is the origin in an Amazon CloudFront distribution. The company wants to implement a custom authentication system that will provide a token for its authenticated customers.
The web application must ensure that the GET/POST requests come from authenticated customers before it delivers the content. A network engineer must design a solution that gives the web application the ability to identify authorized customers.
What is the MOST operationally efficient solution that meets these requirements?

A.Use the ALB to inspect the authorized token inside the GET/POST request payload. Use an AWS Lambda function to insert a customized header to inform the web application of an authenticated customer request.

B.Integrate AWS WAF with the ALB to inspect the authorized token inside the GET/POST request payload. Configure the ALB listener to insert a customized header to inform the web application of an authenticated customer request.

C.Use an AWS Lambda@Edge function to inspect the authorized token inside the GET/POST request payload. Use the Lambda@Edge function also to insert a customized header to inform the web application of an authenticated customer request.

D.Set up an EC2 instance that has a third-party packet inspection tool to inspect the authorized token inside the GET/POST request payload. Configure the tool to insert a customized header to inform the web application of an authenticated customer request.

Answer: C

Explanation:

The most operationally efficient solution for implementing a custom authentication system with token verification for a web application behind CloudFront and ALB is option C, utilizing Lambda@Edge.

Lambda@Edge allows you to execute code at AWS edge locations, which are closer to the users. This provides lower latency compared to routing requests to the origin (ALB) for authentication. In this scenario, the Lambda@Edge function can inspect the authorized token within the GET/POST request payload as it reaches the CloudFront edge location.

Upon successful token validation, the Lambda@Edge function inserts a custom header into the request before forwarding it to the origin. This custom header informs the web application that the request originates from an authenticated user.

This approach is operationally efficient because:

1. **Lower Latency:** Edge processing reduces the round trip time needed for authentication.
2. **Scalability:** Lambda@Edge automatically scales with request volume.
3. **Security:** The token is validated before the request reaches the origin, protecting the web application from unauthorized requests.
4. **Reduced Origin Load:** Only authenticated requests reach the web application, reducing the load on the EC2 instances.
5. **Cost-Effective:** Lambda@Edge is billed only for the compute time consumed.

Option A is less efficient. Using a Lambda function with ALB requires sending traffic to the ALB first, then invoking the Lambda function, adding latency. ALB is not designed for payload inspection.

Option B, integrating AWS WAF with ALB for token inspection, could work for basic token validation rules and

could have scalability challenges. While WAF can filter based on request parameters, it's typically used for security rules, not full-blown authentication and header manipulation.

Option D, using an EC2 instance with a third-party packet inspection tool, is complex and difficult to manage. It requires maintaining the EC2 instance, ensuring high availability, and configuring the packet inspection tool, which is not operationally efficient. It also adds latency and introduces a potential single point of failure.

Therefore, Lambda@Edge is the most operationally efficient and suitable solution for this scenario. It offers low latency, scalability, and security while minimizing operational overhead.

Relevant Documentation:

AWS Lambda@Edge: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/lambda-at-the-edge.html
Amazon CloudFront: https://aws.amazon.com/cloudfront/

Question: 65

A company has created three VPCs: a production VPC, a nonproduction VPC, and a shared services VPC. The production VPC and the nonproduction VPC must each have communication with the shared services VPC. There must be no communication between the production VPC and the nonproduction VPC. A transit gateway is deployed to facilitate communication between VPCs.
Which route table configurations on the transit gateway will meet these requirements?

A.Configure a route table with the production and nonproduction VPC attachments associated with propagated routes for only the shared services VPC. Create an additional route table with only the shared services VPC attachment associated with propagated routes from the production and nonproduction VPCs.

B.Configure a route table with the production and nonproduction VPC attachments associated with propagated routes for each VPC. Create an additional route table with only the shared services VPC attachment associated with propagated routes from each VPC.

C.Configure a route table with all the VPC attachments associated with propagated routes for only the shared services VPCreate an additional route table with only the shared services VPC attachment associated with propagated routes from the production and nonproduction VPCs.

D.Configure a route table with the production and nonproduction VPC attachments associated with propagated routes disabled. Create an additional route table with only the shared services VPC attachment associated with propagated routes from the production and nonproduction VPCs.

Answer: A

Explanation:

The correct answer is A because it effectively isolates traffic between the production and non-production VPCs while allowing both to communicate with the shared services VPC. Let's break down why.

Transit Gateway Route Tables: Transit Gateways use route tables to determine where to forward traffic. Attachments (in this case, VPCs) are associated with route tables, and routes within these tables dictate the next hop for traffic based on its destination. Route propagation enables a route table to automatically learn routes from attached resources (VPCs, VPNs, Direct Connect).

Option A implements two route tables:

Route Table 1 (Production/Non-Production): This route table is associated with the production and non-production VPC attachments. It's configured to propagate routes only from the shared services VPC. This means that the production and non-production VPCs will learn how to reach the shared services VPC (based on the shared services VPC's CIDR block), but they won't learn about each other's CIDR blocks. This prevents direct communication between production and non-production environments.

Route Table 2 (Shared Services): This route table is associated with the shared services VPC attachment. It's configured to propagate routes from both the production and non-production VPCs. This allows the shared services VPC to learn how to reach resources in both the production and non-production VPCs.

The effect is a hub-and-spoke model. Production and Non-Production VPCs are spokes that only know about the hub (Shared Services). The Shared Services VPC knows about both spokes, but the spokes are isolated from each other.

Options B, C and D fail to achieve the necessary isolation:

Option B: Propagating routes from all VPCs into the primary route table would allow production and non-production VPCs to learn about each other, violating the requirement.

Option C: The same issue as option B occurs. By propagating shared service VPC routes to the first route table with all VPC attachments, it effectively creates a single network spanning all VPCs.

Option D: Disabling route propagation on the first route table would isolate Production and Non-Production; however, since there's no mechanism defined that tells the Production and Non-Production VPCs how to reach the Shared Services VPC, communication with Shared Services would fail for Production and Non-Production.

Therefore, only option A provides the required isolation between the production and non-production VPCs while still enabling communication with the shared services VPC through a well-defined routing mechanism.

Authoritative Links:

AWS Transit Gateway Documentation: https://docs.aws.amazon.com/vpc/latest/tgw/what-is-transit-gateway.html
Transit Gateway Route Tables: https://docs.aws.amazon.com/vpc/latest/tgw/tgw-route-tables.html Transit Gateway Route Propagation: https://docs.aws.amazon.com/vpc/latest/tgw/tgw-route-tables.html#tgw-route-propagation

Question: 66

A company is using an AWS Site-to-Site VPN connection from the company's on-premises data center to a virtual private gateway in the AWS Cloud Because of congestion, the company is experiencing availability and performance issues as traffic travels across the internet before the traffic reaches AWS. A network engineer must reduce these issues for the connection as quickly as possible with minimum administration effort.
Which solution will meet these requirements?

A.Edit the existing Site-to-Site VPN connection by enabling acceleration. Stop and start the VPN service on the customer gateway for the new setting to take effect.

B.Configure a transit gateway in the same AWS Region as the existing virtual private gateway. Create a new accelerated Site-to-Site VPN connection. Connect the new connection to the transit gateway by using a VPN attachment. Update the customer gateway device to use the new Site to Site VPN connection. Delete the existing Site-to-Site VPN connection

C.Create a new accelerated Site-to-Site VPN connection. Connect the new Site-to-Site VPN connection to the existing virtual private gateway. Update the customer gateway device to use the new Site-to-Site VPN connection. Delete the existing Site-to-Site VPN connection.

D.Create a new AWS Direct Connect connection with a private VIF between the on-premises data center and the AWS Cloud. Update the customer gateway device to use the new Direct Connect connection. Delete the existing Site-to-Site VPN connection.

Answer: B

Explanation:

The most efficient solution to improve Site-to-Site VPN performance and availability with minimal administrative overhead involves utilizing AWS Transit Gateway with VPN acceleration.

Option B is correct because it leverages AWS Transit Gateway and accelerated Site-to-Site VPN. Transit Gateway acts as a central hub, simplifying network management and routing. By creating a new accelerated Site-to-Site VPN connection and attaching it to the Transit Gateway via a VPN attachment, you benefit from AWS Global Accelerator, which optimizes the network path between your on-premises data center and AWS.

Updating the customer gateway device to use the new connection and deleting the old one ensures traffic utilizes the optimized path. This solution addresses the congestion issues with minimal administrative effort.

Option A is incorrect because you cannot simply enable acceleration on an existing Site-to-Site VPN connection. Acceleration requires a new VPN connection to be established with the acceleration feature enabled from the start.

Option C is incorrect because it suggests connecting the new accelerated VPN directly to the existing virtual private gateway (VGW). While this uses VPN acceleration, it doesn't address the scalable routing and simplified management benefits that Transit Gateway provides, and using a VGW for multiple VPNs can still create management complexities. Transit Gateway makes the solution more scalable and easier to manage.

Option D, while a solid long-term solution, is not the quickest and requires significant initial setup and cost. Establishing a Direct Connect connection takes time, involving physical infrastructure deployment, and may not be feasible as a quick fix for immediate congestion issues.https://aws.amazon.com/vpn/features/https://aws.amazon.com/transit-gateway/https://docs.aws.amazon.com/vpn/latest/s2svpn/vpn-acceleration.html

Question: 67

An Australian ecommerce company hosts all of its services in the AWS Cloud and wants to expand its customer base to the United States (US). The company is targeting the western US for the expansion.
The company's existing AWS architecture consists of four AWS accounts with multiple VPCs deployed in the ap-southeast-2 Region. All VPCs are attached to a transit gateway in ap-southeast-2. There are dedicated VPCs for each application service. The company also has VPCs for centralized security features such as proxies, firewalls, and logging.
The company plans to duplicate the infrastructure from ap-southeast-2 to the us-west-1 Region. A network engineer must establish connectivity between the various applications in the two Regions. The solution must maximize bandwidth, minimize latency and minimize operational overhead.
Which solution will meet these requirements?

A.Create VPN attachments between the two transit gateways. Configure the VPN attachments to use BGP routing between the two transit gateways.

B.Peer the transit gateways in each Region. Configure routing between the two transit gateways for each Region's IP addresses.

C.Create a VPN server in a VPC in each Region. Update the routing to point to the VPN servers for the IP addresses in alternate Regions.

D.Attach the VPCs in us-west-1 to the transit gateway in ap-southeast-2.

Answer: B

Explanation:

The best solution for connecting the Australian ecommerce company's infrastructure in ap-southeast-2 to us-west-1 while maximizing bandwidth, minimizing latency, and minimizing operational overhead is to peer the transit gateways in each Region (Option B).

Transit Gateway peering allows for direct, high-bandwidth connectivity between transit gateways in different AWS Regions. This peering connection creates a fully meshed network between the peered transit gateways. Routing is then configured between the transit gateways so that each transit gateway learns the IP address ranges associated with the VPCs connected to the other transit gateway. This avoids the overhead and

latency associated with VPN connections.

VPN attachments (Option A) introduce additional latency and overhead due to encryption and encapsulation. Furthermore, the bandwidth is limited by the VPN connection's capacity. Attaching VPCs in us-west-1 to the transit gateway in ap-southeast-2 (Option D) would not be a viable solution because resources in different regions need to be physically in those regions. This also does not minimize latency. A VPN server in each region (Option C) suffers from the same latency and bandwidth limitations as VPN attachments.

Transit Gateway peering is the most efficient and scalable method to connect resources across Regions using the AWS backbone network, optimizing performance and reducing operational complexity.

Here are some authoritative links for further research:

AWS Transit Gateway Peering Attachments:https://docs.aws.amazon.com/vpc/latest/tgw/tgw-peering.html
AWS Transit Gateway:https://aws.amazon.com/transit-gateway/

Question: 68

An IoT company sells hardware sensor modules that periodically send out temperature, humidity, pressure, and location data through the MQTT messaging protocol. The hardware sensor modules send this data to the company's on-premises MQTT brokers that run on Linux servers behind a load balancer. The hardware sensor modules have been hardcoded with public IP addresses to reach the brokers.
The company is growing and is acquiring customers across the world. The existing solution can no longer scale and is introducing additional latency because of the company's global presence. As a result, the company decides to migrate its entire infrastructure from on premises to the AWS Cloud. The company needs to migrate without reconfiguring the hardware sensor modules that are already deployed across the world. The solution also must minimize latency.
The company migrates the MQTT brokers to run on Amazon EC2 instances.
What should the company do next to meet these requirements?

A.Place the EC2 instances behind a Network Load Balancer (NLB). Configure TCP listeners. Use Bring Your Own IP (BYOIP) from the on-premises network with the NLB.

B.Place the EC2 instances behind a Network Load Balancer (NLB). Configure TCP listeners. Create an AWS Global Accelerator accelerator in front of the NLUse Bring Your Own IP (BYOIP) from the on-premises network with Global Accelerator.

C.Place the EC2 instances behind an Application Load Balancer (ALB). Configure TCP listeners. Create an AWS Global Accelerator accelerator in front of the ALB. Use Bring Your Own IP (BYOIP) from the on-premises network with Global Accelerator

D.Place the EC2 instances behind an Amazon CloudFront distribution. Use Bring Your Own IP (BYOIP) from the on-premises network with CloudFront.

Answer: B

Explanation:

Here's a detailed justification for why option B is the best solution, along with explanations of why the other options are not ideal:

The primary goal is to migrate the MQTT brokers to AWS without reconfiguring the already deployed hardware sensor modules that are hardcoded with public IP addresses. The solution must also minimize latency globally.

Option B: Place the EC2 instances behind a Network Load Balancer (NLB). Configure TCP listeners. Create an AWS Global Accelerator accelerator in front of the NLB. Use Bring Your Own IP (BYOIP) from the on-premises network with Global Accelerator.

This is the most suitable option because it addresses all requirements.

NLB with TCP Listeners: NLBs are designed for handling TCP traffic, which is the protocol used by MQTT. Configuring TCP listeners allows the NLB to forward traffic to the MQTT broker EC2 instances.

AWS Global Accelerator: This service is designed to minimize latency for globally distributed clients. It uses the AWS global network to route traffic to the nearest edge location, reducing the distance and improving performance. Global Accelerator then routes traffic to the NLB.

Bring Your Own IP (BYOIP) with Global Accelerator: BYOIP enables the company to bring their existing public IP address range to AWS and associate them with the Global Accelerator. Since the sensor modules are hardcoded with specific public IPs, using BYOIP ensures that no changes are needed on the sensor module side during the migration. Traffic sent to these IP addresses will be routed via Global Accelerator to the nearest edge location and then routed to the NLB.

Option A: Place the EC2 instances behind a Network Load Balancer (NLB). Configure TCP listeners. Use Bring Your Own IP (BYOIP) from the on-premises network with the NLB.

This option fulfills the hardcoded IP requirement by using BYOIP with the NLB. However, it lacks the global latency optimization provided by AWS Global Accelerator. The NLB exists in a single region, so clients from distant locations will experience higher latency compared to Option B.

Option C: Place the EC2 instances behind an Application Load Balancer (ALB). Configure TCP listeners. Create an AWS Global Accelerator accelerator in front of the ALB. Use Bring Your Own IP (BYOIP) from the on-premises network with Global Accelerator.

This option is incorrect because Application Load Balancers (ALBs) primarily operate at Layer 7 (HTTP/HTTPS). MQTT typically runs over TCP (Layer 4). While ALBs can forward TCP traffic, they're not optimized for it and aren't the best choice for this MQTT use case.

Option D: Place the EC2 instances behind an Amazon CloudFront distribution. Use Bring Your Own IP (BYOIP) from the on-premises network with CloudFront.

CloudFront is a Content Delivery Network (CDN) primarily used for caching static and dynamic content. It's not designed for handling persistent connections like those used by MQTT. While CloudFront can accelerate content delivery, it's not the appropriate solution for a real-time messaging protocol.
Additionally, CloudFront is designed for HTTP/HTTPS traffic, not raw TCP.

In conclusion, Option B provides the best solution because it combines the use of NLB for TCP traffic management, Global Accelerator for latency optimization, and BYOIP for maintaining the existing hardcoded IP addresses, allowing for a seamless migration without requiring any changes on the sensor modules.

Authoritative Links:

AWS Global Accelerator:https://aws.amazon.com/global-accelerator/
Network Load Balancer (NLB):https://aws.amazon.com/elasticloadbalancing/network-load-balancer/ **Bring Your Own IP (BYOIP):**https://docs.aws.amazon.com/global-accelerator/latest/dg/using-byoip.html

Question: 69

A company has deployed a web application on AWS. The web application uses an Application Load Balancer (ALB) across multiple Availability Zones. The targets of the ALB are AWS Lambda functions. The web application also uses Amazon CloudWatch metrics for monitoring.
Users report that parts of the web application are not loading properly. A network engineer needs to troubleshoot the problem. The network engineer enables access logging for the ALB.
What should the network engineer do next to determine which errors the ALB is receiving?

A.Send the logs to Amazon CloudWatch Logs. Review the ALB logs in CloudWatch Insights to determine which error messages the ALB is receiving.

B.Configure the Amazon S3 bucket destination. Use Amazon Athena to determine which error messages the

ALB is receiving.

C.Configure the Amazon S3 bucket destination. After Amazon CloudWatch Logs pulls the ALB logs from the S3 bucket automatically, review the logs in CloudWatch Logs to determine which error messages the ALB is receiving.

D.Send the logs to Amazon CloudWatch Logs. Use the Amazon Athena CloudWatch Connector to determine which error messages the ALB is receiving.

Answer: B

Explanation:

The correct answer is B. Here's why:

Application Load Balancer (ALB) access logs are stored in Amazon S3 buckets. To efficiently analyze these logs for errors, you need a querying tool. Amazon Athena is a serverless, interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL.

Option A is incorrect because while CloudWatch Logs can ingest logs, it's not designed for efficient analysis of large-scale, structured data like ALB access logs directly without considerable pre-processing and complex filtering. CloudWatch Logs Insights is better suited for real-time operational data rather than historical analysis of access logs.

Option C is incorrect because CloudWatch Logs doesn't automatically pull logs from S3. You would need to configure a mechanism for this transfer, which is not an efficient approach compared to directly querying the data in S3 with Athena.

Option D is incorrect because while Athena can connect to CloudWatch Logs via a connector, it's not the direct and efficient method for analyzing ALB access logs, which are already stored in S3. Using Athena directly on the S3 bucket where the ALB logs are stored is more efficient and cost-effective.

Therefore, the most efficient workflow is to configure the ALB to send access logs to an S3 bucket and then use Amazon Athena to query those logs to identify error messages. Athena allows you to run SQL queries directly against the log files in S3, making it easy to identify specific errors like HTTP 500 errors or other issues reflected in the logs.

Here are some helpful links:

Application Load Balancer Access Logs
Amazon Athena
Analyzing Application Load Balancer logs with Amazon Athena

Question: 70

A company is planning to use Amazon S3 to archive financial data. The data is currently stored in an on-premises data center. The company uses AWS Direct Connect with a Direct Connect gateway and a transit gateway to connect to the on-premises data center. The data cannot be transported over the public internet and must be encrypted in transit.
Which solution will meet these requirements?

A.Create a Direct Connect public VIF. Set up an IPsec VPN connection over the public VIF to access Amazon S3. Use HTTPS for communication.

B.Create an IPsec VPN connection over the transit VIF. Create a VPC and attach the VPC to the transit gateway. In the VPC, provision an interface VPC endpoint for Amazon S3. Use HTTPS for communication.

C.Create a VPC and attach the VPC to the transit gateway. In the VPC, provision an interface VPC endpoint for Amazon S3. Use HTTPS for communication.

D.Create a Direct Connect public VIF. Set up an IPsec VPN connection over the public VIF to the transit gateway. Create an attachment for Amazon S3. Use HTTPS for communication.

Answer: B

Explanation:

The correct solution is B because it leverages AWS Direct Connect's private connectivity along with secure tunneling and interface VPC endpoints to meet the stated requirements of private data transfer and encryption in transit.

Here's a breakdown of why option B is the best choice:

1. **Private Connectivity:** The question explicitly states that data cannot be transported over the public internet. Option B utilizes the existing AWS Direct Connect connection through the transit VIF, ensuring data remains on the AWS private network.

2. **Encryption in Transit:** Option B establishes an IPsec VPN connection over the transit VIF. This creates an encrypted tunnel for the data as it travels between the on-premises data center and AWS. HTTPS is also specified for S3 communication adding another layer of encryption.

3. **Access to S3:** Creating a VPC, attaching it to the Transit Gateway, and then provisioning an interface VPC endpoint for S3 provides a private connection to S3 within the AWS network, bypassing the public internet. This private connectivity to S3 ensures data transfer occurs securely and privately within the AWS environment. Interface VPC Endpoints use AWS PrivateLink to provide private connectivity.

Why other options are incorrect:

Option A & D: Both options A and D propose creating a public VIF. This contradicts the explicit requirement that data cannot be transported over the public internet. Public VIFs inherently route traffic over the internet.

Additionally, Option D attempts to create an "attachment" for Amazon S3 on a public VIF which isn't a valid configuration.

Option C: Option C lacks encryption in transit. While it establishes private connectivity to S3 using a VPC endpoint, it doesn't secure the data's journey from the on-premises data center to AWS. The data needs to be encrypted while traversing the Direct Connect link, which requires an IPsec VPN.

In summary, option B offers the most complete solution by combining the private connectivity of Direct Connect, the encryption of an IPsec VPN, and the private S3 access provided by an interface VPC endpoint, fulfilling all stated requirements.

Supporting Links:

AWS Direct Connect:https://aws.amazon.com/directconnect/
AWS Transit Gateway:https://aws.amazon.com/transit-gateway/
AWS VPN:https://aws.amazon.com/vpn/
VPC Endpoints:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html **AWS PrivateLink:**https://aws.amazon.com/privatelink/

Question: 71

A company is using Amazon Route 53 Resolver DNS Firewall in a VPC to block all domains except domains that are on an approved list. The company is concerned that if DNS Firewall is unresponsive, resources in the VPC might be affected if the network cannot resolve any DNS queries. To maintain application service level agreements, the company needs DNS queries to continue to resolve even if Route 53 Resolver does not receive a response from DNS Firewall.
Which change should a network engineer implement to meet these requirements?

A.Update the DNS Firewall VPC configuration to disable fail open for the VPC.

B.Update the DNS Firewall VPC configuration to enable fail open for the VPC.

C.Create a new DHCP options set with parameter dns_firewall_fail_open=false. Associate the new DHCP options set with the VPC.

D.Create a new DHCP options set with parameter dns_firewall_fail_open=true. Associate the new DHCP options set with the VPC.

Answer: B

Explanation:

The question requires ensuring DNS resolution continues even if Route 53 Resolver DNS Firewall is unresponsive. This implies a need for a fail-safe mechanism.

Option B suggests enabling "fail open" for the DNS Firewall VPC configuration. Fail open, in the context of DNS Firewall, means that if the firewall is unable to process DNS queries (e.g., due to an outage), it allows all DNS queries to pass through without being inspected. This ensures that applications can continue to resolve DNS and function normally, even if the DNS Firewall is unavailable. This directly addresses the requirement of maintaining application service levels despite DNS Firewall unresponsiveness.

Option A suggests disabling fail open, which is the opposite of what's needed. Disabling fail open would cause DNS resolution to fail if the firewall is unresponsive, thus violating the requirements.

Options C and D suggest using DHCP options sets to configure the "dns_firewall_fail_open" parameter. However, the dns_firewall_fail_open is not a valid DHCP option. Route 53 Resolver DNS Firewall's fail-open behavior is configured directly within the VPC's DNS Firewall configuration, not through DHCP.

Therefore, the only option that achieves the stated goal is to enable fail open at the VPC level, ensuring DNS resolution continues when the DNS Firewall is unavailable.

https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver-dns-firewall-configuring.htmlhttps://aws.amazon.com/blogs/networking-and-content-delivery/protecting-your-dns-traffic-using-route-53-resolver-dns-firewall/

Question: 72

A company is migrating an existing application to a new AWS account. The company will deploy the application in a single AWS Region by using one VPC and multiple Availability Zones. The application will run on Amazon EC2 instances. Each Availability Zone will have several EC2 instances. The EC2 instances will be deployed in private subnets.

The company's clients will connect to the application by using a web browser with the HTTPS protocol. Inbound connections must be distributed across the Availability Zones and EC2 instances. All connections from the same client session must be connected to the same EC2 instance. The company must provide end-to-end encryption for all connections between the clients and the application by using the application SSL certificate.

Which solution will meet these requirements?

A.Create a Network Load Balancer. Create a target group. Set the protocol to TCP and the port to 443 for the target group. Turn on session affinity (sticky sessions). Register the EC2 instances as targets. Create a listener. Set the protocol to TCP and the port to 443 for the listener. Deploy SSL certificates to the EC2 instances.

B.Create an Application Load Balancer. Create a target group. Set the protocol to HTTP and the port to 80 for the target group. Turn on session affinity (sticky sessions) with an application-based cookie policy. Register the EC2 instances as targets. Create an HTTPS listener. Set the default action to forward to the target group. Use AWS Certificate Manager (ACM) to create a certificate for the listener.

C.Create a Network Load Balancer. Create a target group. Set the protocol to TLS and the port to 443 for the target group. Turn on session affinity (sticky sessions). Register the EC2 instances as targets. Create a listener. Set the protocol to TLS and the port to 443 for the listener. Use AWS Certificate Manager (ACM) to create a

certificate for the application.

D.Create an Application Load Balancer. Create a target group. Set the protocol to HTTPS and the port to 443 for the target group. Turn on session affinity (sticky sessions) with an application-based cookie policy. Register the EC2 instances as targets. Create an HTTP listener. Set the port to 443 for the listener. Set the default action to forward to the target group.

Answer: A

Explanation:

The correct answer is A. Here's why:

The requirements specify HTTPS protocol with end-to-end encryption using the application's SSL certificate, distribution across Availability Zones, and session affinity (sticky sessions).

Network Load Balancer (NLB): NLBs operate at the transport layer (Layer 4) and are ideal for TCP/TLS traffic. This makes them suitable for handling the HTTPS traffic while providing high performance and low latency. NLBs also support TLS termination.

TCP/TLS Protocol: NLBs can forward traffic using TCP or TLS protocols. Since we need to encrypt all traffic between the clients and the EC2 instances, TLS should be implemented either on the NLB or on the backend instances (end-to-end). The answer suggests implementing TLS on the backend instances.

Session Affinity (Sticky Sessions): NLBs support session affinity based on IP addresses, which ensures that all connections from the same client IP address are routed to the same EC2 instance.

End-to-End Encryption: Deploying the SSL certificates directly on the EC2 instances ensures end-to-end encryption between the clients and the application.

Why other options are incorrect:

Option B (ALB with HTTP): An Application Load Balancer (ALB) with HTTP protocol would terminate the SSL connection at the ALB and send unencrypted traffic to the EC2 instances, failing to meet the end-to-end encryption requirement.

Option C (NLB with TLS and ACM): This option suggests using ACM for the NLB, but doesn't say to also add certificates to the EC2 instances. This would terminate the TLS connection at the NLB, and traffic between the NLB and EC2 instance would be unencrypted.

Option D (ALB with HTTPS listener and HTTP target): This configuration would result in the ALB terminating the HTTPS connection and forwarding unencrypted HTTP traffic to the EC2 instances, violating the end-to-end encryption requirement. Also the ports don't match between the listener and target group.

Supporting documentation:

Network Load Balancer
Application Load Balancer
Elastic Load Balancing Listener Configuration

Question: 73

A company is developing an application in which IoT devices will report measurements to the AWS Cloud. The application will have millions of end users. The company observes that the IoT devices cannot support DNS resolution. The company needs to implement an Amazon EC2 Auto Scaling solution so that the IoT devices can connect to an application endpoint without using DNS.

Which solution will meet these requirements MOST cost-effectively?

A.Use an Application Load Balancer (ALB)-type target group for a Network Load Balancer (NLB). Create an EC2

Auto Scaling group. Attach the Auto Scaling group to the ALB. Set up the IoT devices to connect to the IP addresses of the NLB.

B.Use an AWS Global Accelerator accelerator with an Application Load Balancer (ALB) endpoint. Create an EC2 Auto Scaling group. Attach the Auto Scaling group to the ALSet up the IoT devices to connect to the IP addresses of the accelerator.

C.Use a Network Load Balancer (NLB). Create an EC2 Auto Scaling group. Attach the Auto Scaling group to the NLB. Set up the IoT devices to connect to the IP addresses of the NLB.

D.Use an AWS Global Accelerator accelerator with a Network Load Balancer (NLB) endpoint. Create an EC2 Auto Scaling group. Attach the Auto Scaling group to the NLB. Set up the IoT devices to connect to the IP addresses of the accelerator.

Answer: C

Explanation:

The scenario requires a cost-effective solution where IoT devices, incapable of DNS resolution, can connect to an application endpoint without DNS. We need a load balancing solution that provides static IP addresses and integrates well with EC2 Auto Scaling.

Option C utilizes a Network Load Balancer (NLB) directly connected to an EC2 Auto Scaling group. NLBs provide static IP addresses per Availability Zone, addressing the core requirement of no DNS resolution needed. The Auto Scaling group dynamically scales the EC2 instances based on demand. This approach is straightforward and cost-effective because it leverages the core functionality of NLBs without introducing additional services. NLBs are designed for high throughput and low latency, making them suitable for handling the high volume of connections from millions of IoT devices.

Option A uses an Application Load Balancer (ALB) behind an NLB. This introduces unnecessary complexity and cost. ALBs are typically used for HTTP/HTTPS traffic and layer-7 routing, which are not relevant to the stated problem concerning IoT devices lacking DNS capabilities.

Option B uses AWS Global Accelerator with an ALB endpoint. Global Accelerator improves performance by routing traffic over AWS's global network, but is not needed simply to provide fixed IPs for devices that can't resolve DNS. It would increase complexity and cost without providing specific advantages for IoT devices with DNS limitations.

Option D uses AWS Global Accelerator with an NLB endpoint. While Global Accelerator provides static IP addresses, it is more expensive than using the NLB's static IPs directly. Global Accelerator is typically used for applications needing global performance improvements, which are not specified here. Thus, it introduces unnecessary cost and complexity.

Therefore, option C is the most cost-effective and straightforward solution because it leverages NLB's static IP addresses and Auto Scaling group integration to meet the stated requirements without introducing unnecessary complexity or services.

Further Research:

Network Load Balancer (NLB):
https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html
EC2 Auto Scaling:https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-benefits.html

Question: 74

A company has deployed a new web application on Amazon EC2 instances behind an Application Load Balancer (ALB). The instances are in an Amazon EC2 Auto Scaling group. Enterprise customers from around the world will use the application. Employees of these enterprise customers will connect to the application over HTTPS from office locations.

The company must configure firewalls to allow outbound traffic to only approved IP addresses. The employees of the enterprise customers must be able to access the application with the least amount of latency.

Which change should a network engineer make in the infrastructure to meet these requirements?

A.Create a new Network Load Balancer (NLB). Add the ALB as a target of the NLB.
B.Create a new Amazon CloudFront distribution. Set the ALB as the distribution's origin.
C.Create a new accelerator in AWS Global Accelerator. Add the ALB as an accelerator endpoint.
D.Create a new Amazon Route 53 hosted zone. Create a new record to route traffic to the ALB.

Answer: C

Explanation:

The correct answer is C: Create a new accelerator in AWS Global Accelerator. Add the ALB as an accelerator endpoint.

Here's why this is the best solution, and why the others aren't ideal:

AWS Global Accelerator: Global Accelerator uses the AWS global network to route traffic to the closest healthy endpoint based on the user's location. This significantly reduces latency for global users, as traffic doesn't have to traverse long distances over the public internet. Crucially, Global Accelerator provides static IP addresses that customers can whitelist in their firewalls, meeting the security requirement. Global Accelerator offers connection optimization and failover capabilities, ensuring high availability and performance. https://aws.amazon.com/global-accelerator/

Why other options are incorrect:

A. Network Load Balancer (NLB): While NLBs can handle high traffic volumes, they don't inherently reduce latency for globally distributed users. They primarily operate within a single AWS Region. Using an NLB in front of an ALB adds complexity without addressing the latency or static IP address requirements.

B. Amazon CloudFront: CloudFront caches content at edge locations, reducing latency for repeated requests. However, it's primarily designed for static or cacheable content. The web application, likely generating dynamic content, wouldn't benefit as much from CloudFront. Also, CloudFront's IP address ranges change, making it difficult to whitelist them in firewalls.

D. Amazon Route 53 hosted zone: Route 53 is a DNS service. While it can route traffic to the ALB, it doesn't address the latency concerns for global users or provide static IP addresses for whitelisting. The routing strategies in Route 53 (like latency-based routing) aren't as optimized for performance across the AWS global network as Global Accelerator.

In summary, Global Accelerator is the only service that effectively addresses both the low latency requirement for a global user base and the need for static IP addresses for firewall whitelisting. It leverages the AWS global network for optimized routing, making it the most suitable solution.

Question: 75

A company has hundreds of VPCs on AWS. All the VPCs access the public endpoints of Amazon S3 and AWS Systems Manager through NAT gateways. All the traffic from the VPCs to Amazon S3 and Systems Manager travels through the NAT gateways. The company's network engineer must centralize access to these services and must eliminate the need to use public endpoints.

Which solution will meet these requirements with the LEAST operational overhead?

A.Create a central egress VPC that has private NAT gateways. Connect all the VPCs to the central egress VPC

by using AWS Transit Gateway. Use the private NAT gateways to connect to Amazon S3 and Systems Manager by using private IP addresses.

B.Create a central shared services VPC. In the central shared services VPC, create interface VPC endpoints for Amazon S3 and Systems Manager to access. Ensure that private DNS is turned off. Connect all the VPCs to the central shared services VPC by using AWS Transit Gateway. Create an Amazon Route 53 forwarding rule for each interface VPC endpoint. Associate the forwarding rules with all the VPCs. Forward DNS queries to the interface VPC endpoints in the shared services VPC.

C.Create a central shared services VPIn the central shared services VPC, create interface VPC endpoints for Amazon S3 and Systems Manager to access. Ensure that private DNS is turned off. Connect all the VPCs to the central shared services VPC by using AWS Transit Gateway. Create an Amazon Route 53 private hosted zone with a full service endpoint name for Amazon S3 and Systems Manager. Associate the private hosted zones with all the VPCs. Create an alias record in each private hosted zone with the full AWS service endpoint pointing to the interface VPC endpoint in the shared services VPC.

D.Create a central shared services VPC. In the central shared services VPC, create interface VPC endpoints for Amazon S3 and Systems Manager to access. Connect all the VPCs to the central shared services VPC by using AWS Transit Gateway. Ensure that private DNS is turned on for the interface VPC endpoints and that the transit gateway is created with DNS support turned on.

Answer: C

Explanation:

The correct answer is C because it provides the most streamlined approach to centralizing access to S3 and Systems Manager while minimizing operational overhead. Here's a detailed justification:

Centralized Access: Option C establishes a shared services VPC containing interface VPC endpoints for S3 and Systems Manager. This concentrates access in one location, eliminating the need for individual NAT gateways in each VPC.

Private Connectivity: Using interface VPC endpoints ensures all traffic to S3 and Systems Manager remains within the AWS network, bypassing the public internet and eliminating the need for NAT gateways.

Least Operational Overhead: Creating an Amazon Route 53 private hosted zone and associating it with all VPCs automates DNS resolution for S3 and Systems Manager. This ensures that when services in any of the VPCs attempt to access S3 or Systems Manager, they will automatically resolve to the interface VPC endpoints within the shared services VPC. This simplifies the network configuration and maintenance compared to other options.

Private DNS Zone for resolution: Private DNS hosted zones provide DNS resolution only for the VPCs that it is associated with.

Alias records in route 53: Alias records map to VPC endpoints which handles the internal routing to services in AWS.

Why other options are wrong:

Option A: Using private NAT gateways in a central egress VPC does not eliminate the need to route traffic through public endpoints and does not leverage the benefits of interface VPC endpoints.

Option B: Turning off Private DNS and implementing custom Route 53 forwarding rules introduces unnecessary complexity. Interface endpoints automatically handle DNS resolution with Private DNS enabled, while the Transit Gateway has an option for turning on DNS support.

Option D: It enables Private DNS name resolution, which may cause DNS conflicts as the same DNS names might be used in other VPCs.

Here are some helpful resources:

AWS PrivateLink:https://aws.amazon.com/privatelink/
VPC Endpoints:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html
AWS Transit Gateway:https://aws.amazon.com/transit-gateway/
Route 53 Private Hosted Zones:https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/private-hosted-zones.html

Question: 76

A company manages resources across VPCs in multiple AWS Regions. The company needs to connect to the resources by using its internal domain name. A network engineer needs to apply the aws.example.com DNS suffix to all resources.

What must the network engineer do to meet this requirement?

A.Create an Amazon Route 53 private hosted zone for aws.example.com in each Region that has resources. Associate the private hosted zone with that Region's VPC. In the appropriate private hosted zone, create DNS records for the resources in each Region.

B.Create one Amazon Route 53 private hosted zone for aws.example.com. Configure the private hosted zone to allow zone transfers with every VPC.

C.Create one Amazon Route 53 private hosted zone for example.com. Create a single resource record for aws.example.com in the private hosted zone. Apply a multivalue answer routing policy to the record. Add all VPC resources as separate values in the routing policy.

D.Create one Amazon Route 53 private hosted zone for aws.example.com. Associate the private hosted zone with every VPC that has resources. In the private hosted zone, create DNS records for all resources.

Answer: D

Explanation:

The correct answer is D. Here's why:

The requirement is to resolve resources across multiple VPCs and Regions using the aws.example.com DNS suffix. This necessitates a centralized DNS management solution that can span these different network environments. Amazon Route 53 private hosted zones are designed for this purpose.

Option D proposes creating a single private hosted zone for aws.example.com and associating it with all relevant VPCs. This is crucial because it allows all VPCs, regardless of Region, to query the same DNS records. Within this hosted zone, the network engineer creates the necessary DNS records (A, CNAME, etc.) pointing to the resources in each VPC. This way, any resource within any of the associated VPCs can resolve resources using the aws.example.com suffix.

Option A is incorrect because creating separate private hosted zones in each Region defeats the purpose of a centralized DNS solution. While it would work within a single Region, cross-Region resolution would become overly complex. It also adds operational overhead because the DNS entries need to be managed across multiple zones.

Option B is incorrect because Route 53 doesn't natively support zone transfers to VPCs. Zone transfers are typically used between DNS servers. Sharing DNS information with VPCs is done using Route 53's VPC association feature, not zone transfers.

Option C is incorrect because using a multivalue answer routing policy alone doesn't solve the problem. While multivalue answer routing is a valid routing policy within Route 53, it doesn't simplify the DNS configuration across multiple VPCs. A single record for aws.example.com with multiple values doesn't address the need to create records for the specific resources within each VPC. Plus it uses example.com which fails to create DNS suffix of aws.example.com

In summary, option D offers the most straightforward, scalable, and manageable approach to resolving resources across multiple VPCs and Regions using a consistent DNS suffix. It leverages the core capabilities of Route 53 private hosted zones for centralized DNS management.

Further research:

Amazon Route 53 Private Hosted Zones

Associating Hosted Zones with Your VPCs

Question: 77

An insurance company is planning the migration of workloads from its on-premises data center to the AWS Cloud. The company requires end-to-end domain name resolution. Bi-directional DNS resolution between AWS and the existing on-premises environments must be established. The workloads will be migrated into multiple VPCs. The workloads also have dependencies on each other, and not all the workloads will be migrated at the same time.

Which solution meets these requirements?

A.Configure a private hosted zone for each application VPC, and create the requisite records. Create a set of Amazon Route 53 Resolver inbound and outbound endpoints in an egress VPC. Define Route 53 Resolver rules to forward requests for the on-premises domains to the on-premises DNS resolver. Associate the application VPC private hosted zones with the egress VPC, and share the Route 53 Resolver rules with the application accounts by using AWS Resource Access Manager. Configure the on-premises DNS servers to forward the cloud domains to the Route 53 inbound endpoints.

B.Configure a public hosted zone for each application VPC, and create the requisite records. Create a set of Amazon Route 53 Resolver inbound and outbound endpoints in an egress VPC. Define Route 53 Resolver rules to forward requests for the on-premises domains to the on-premises DNS resolver. Associate the application VPC private hosted zones with the egress VPC. and share the Route 53 Resolver rules with the application accounts by using AWS Resource Access Manager. Configure the on-premises DNS servers to forward the cloud domains to the Route 53 inbound endpoints.

C.Configure a private hosted zone for each application VPC, and create the requisite records. Create a set of Amazon Route 53 Resolver inbound and outbound endpoints in an egress VPDefine Route 53 Resolver rules to forward requests for the on-premises domains to the on-premises DNS resolver. Associate the application VPC private hosted zones with the egress VPand share the Route 53 Resolver rules with the application accounts by using AWS Resource Access Manager. Configure the on-premises DNS servers to forward the cloud domains to the Route 53 outbound endpoints.

D.Configure a private hosted zone for each application VPC, and create the requisite records. Create a set of Amazon Route 53 Resolver inbound and outbound endpoints in an egress VPC. Define Route 53 Resolver rules to forward requests for the on-premises domains to the on-premises DNS resolver. Associate the Route 53 outbound rules with the application VPCs, and share the private hosted zones with the application accounts by using AWS Resource Access Manager. Configure the on-premises DNS servers to forward the cloud domains to the Route 53 inbound endpoints.

Answer: A

Explanation:

Here's a detailed justification for why option A is the correct solution, along with supporting concepts and links:

The question specifies a requirement for bi-directional DNS resolution between AWS and on-premises environments during a workload migration. The solution must support multiple VPCs and dependencies between workloads, even as they are migrated incrementally.

Option A correctly addresses these requirements by leveraging Amazon Route 53 Resolver inbound and outbound endpoints. Here's a breakdown:

1. **Private Hosted Zones:** Using private hosted zones for each application VPC ensures internal DNS resolution within each VPC is isolated and managed independently. This aligns with the "multiple VPCs" requirement and allows for specific DNS records related to the migrated applications.
2. **Route 53 Resolver Endpoints (Inbound & Outbound):** Route 53 Resolver inbound endpoints provide a target IP address for the on-premises DNS servers to forward DNS queries for AWS-based domains. Conversely, outbound endpoints allow AWS workloads to query the on-premises DNS servers.
3. **Egress VPC:** Centralizing the Route 53 Resolver endpoints in an egress VPC simplifies management and security. Instead of creating and managing endpoints in every VPC, a single, well-controlled

egress point handles DNS traffic.

4. **Route 53 Resolver Rules:** Resolver rules define how DNS queries are routed. Specifically, a rule is created to forward requests for on-premises domains to the on-premises DNS resolver.
5. **Resource Access Manager (RAM):** RAM allows sharing the Route 53 Resolver rules with the application accounts. Sharing the resolver rules is crucial for centralizing management and ensuring consistent routing policies.
6. **On-Premises DNS Forwarding:** Configuring the on-premises DNS servers to forward requests for cloud domains to the Route 53 inbound endpoints establishes the bi-directional DNS resolution.

In contrast, here's why the other options are not as suitable:

Option B: Using public hosted zones is incorrect because the requirement is for internal, private DNS resolution between AWS and on-premises environments. Public hosted zones are for publicly accessible domains.

Option C: Configuring the on-premises DNS servers to forward to Route 53 outbound endpoints is backwards. On-premises DNS servers need to forward to inbound endpoints.

Option D: Associating Route 53 outbound rules with application VPCs, instead of sharing the hosted zones for AWS to On-Prem resolution, is also not the intended method.

Supporting Concepts & Links:

Amazon Route 53 Resolver:https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver.html **Private Hosted Zones:**https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/private-hosted-zones.html
AWS Resource Access Manager (RAM):https://aws.amazon.com/ram/
Hybrid DNS:https://aws.amazon.com/blogs/networking-and-content-delivery/hybrid-cloud-dns-resolution-using-amazon-route-53-resolver/

In summary, option A provides a comprehensive and well-architected solution for bi-directional DNS resolution in a hybrid cloud environment with multiple VPCs, aligning with the best practices for workload migration and AWS networking.

Question: 78

A global company runs business applications in the us-east-1 Region inside a VPC. One of the company's regional offices in London uses a virtual private gateway for an AWS Site-to-Site VPN connection tom the VPC. The company has configured a transit gateway and has set up peering between the VPC and other VPCs that various departments in the company use.

Employees at the London office are experiencing latency issues when they connect to the business applications.

What should a network engineer do to reduce this latency?

A.Create a new Site-to-Site VPN connection. Set the transit gateway as the target gateway. Enable acceleration on the new Site-to-Site VPN connection. Update the VPN device in the London office with the new connection details.

B.Modify the existing Site-to-Site VPN connection by setting the transit gateway as the target gateway. Enable acceleration on the existing Site-to-Site VPN connection.

C.Create a new transit gateway in the eu-west-2 (London) Region. Peer the new transit gateway with the existing transit gateway. Modify the existing Site-to-Site VPN connection by setting the new transit gateway as the target gateway.

D.Create a new AWS Global Accelerator standard accelerator that has an endpoint of the Site-to-Site VPN connection. Update the VPN device in the London office with the new connection details.

Answer: A

Explanation:

The correct answer is **A**. Here's why:

The core issue is latency between the London office and the AWS VPC in us-east-1. The existing connection is a standard Site-to-Site VPN through a virtual private gateway. Virtual private gateways do not offer acceleration options. To improve performance, we want to leverage AWS's global network for faster routing. AWS Site-to-Site VPN connections now support acceleration.

Option A proposes creating a new Site-to-Site VPN connection and associating it with the transit gateway. Crucially, it suggests enabling acceleration on this new connection. Acceleration leverages AWS's global network to optimize the path between the VPN endpoint in London and the transit gateway, significantly reducing latency. Updating the London office's VPN device ensures traffic uses the new, accelerated connection.

Option B is incorrect because you can't simply enable acceleration on an existing VPN connection that isn't already using a Transit Gateway as its endpoint. The underlying infrastructure needs to be re-provisioned to support acceleration.

Option C involves creating a new transit gateway in London and peering it. While this might seem helpful, it adds unnecessary complexity and cost. The latency problem is between London and us-east-1, not within London. Also, this would not solve the issue, as the initial leg of the traffic would still traverse across the regular internet.

Option D proposes using AWS Global Accelerator. While Global Accelerator is a good service, it doesn't directly integrate with Site-to-Site VPN connections in the way required for this scenario. Global Accelerator directs traffic to specified endpoints (e.g., EC2 instances, Network Load Balancers), but can't be configured as an endpoint for a Site-to-Site VPN connection directly targeting a transit gateway. The Site-to-Site VPN solution using acceleration is a better fit for this low latency use case.

Therefore, Option A provides the most direct and efficient solution to reduce latency by utilizing AWS's network acceleration capabilities for Site-to-Site VPN connections through a transit gateway.

Further reading:

AWS Site-to-Site VPN:https://aws.amazon.com/vpn/
AWS Transit Gateway:https://aws.amazon.com/transit-gateway/
Accelerated Site-to-Site VPN Connections:https://aws.amazon.com/about-aws/whats-new/2021/03/aws-site-to-site-vpn-accelerated-vpn-connections/

Question: 79

A company has a hybrid cloud environment. The company’s data center is connected to the AWS Cloud by an AWS Direct Connect connection. The AWS environment includes VPCs that are connected together in a hub-and-spoke model by a transit gateway. The AWS environment has a transit VIF with a Direct Connect gateway for on-premises connectivity.

The company has a hybrid DNS model. The company has configured Amazon Route 53 Resolver endpoints in the hub VPC to allow bidirectional DNS traffic flow. The company is running a backend application in one of the VPCs.

The company uses a message-oriented architecture and employs Amazon Simple Queue Service (Amazon SQS) to receive messages from other applications over a private network. A network engineer wants to use an interface VPC endpoint for Amazon SQS for this architecture. Client services must be able to access the endpoint service from on premises and from multiple VPCs within the company's AWS infrastructure.

Which combination of steps should the network engineer take to ensure that the client applications can resolve DNS for the interface endpoint? (Choose three.)

A.Create the interface endpoint for Amazon SQS with the option for private DNS names turned on.

B.Create the interface endpoint for Amazon SQS with the option for private DNS names turned off.

C.Manually create a private hosted zone for sqs.us-east-1.amazonaws.com. Add necessary records that point to the interface endpoint. Associate the private hosted zones with other VPCs.

D.Use the automatically created private hosted zone for sqs.us-east-1.amazonaws.com with previously created necessary records that point to the interface endpoint. Associate the private hosted zones with other VPCs.

E.Access the SQS endpoint by using the public DNS name sqs.us-east-1 amazonaws.com in VPCs and on premises.

F.Access the SQS endpoint by using the private DNS name of the interface endpoint .sqs.us-east-1.vpce.amazonaws.com in VPCs and on premises.

Answer: BCF

Explanation:

Here's a breakdown of why the correct answer is BCF, along with justifications and relevant links:

B. Create the interface endpoint for Amazon SQS with the option for private DNS names turned off.

When creating a VPC endpoint, enabling private DNS does not automatically propagate the endpoint's DNS records to your existing DNS infrastructure, especially across VPCs and on-premises. With Private DNS name turned off, AWS will create the endpoint and associate it to the VPC with a regional Endpoint DNS (vpce) which will be used for your DNS configurations. To use Private DNS, additional steps are needed that is beyond the scope of the question.

C. Manually create a private hosted zone for sqs.us-east-1.amazonaws.com. Add necessary records that point to the interface endpoint. Associate the private hosted zones with other VPCs.

Since the requirement is to have on-premises and multiple VPCs resolve the endpoint, a private hosted zone that mirrors the SQS domain is essential. This allows you to override the public DNS resolution with the private IP addresses of the interface endpoint within your AWS network. You then need to associate the private hosted zone with all the VPCs that need to access SQS via the endpoint. This is a common pattern for hybrid cloud DNS resolution.

F. Access the SQS endpoint by using the private DNS name of the interface endpoint .sqs.us-east-1.vpce.amazonaws.com in VPCs and on premises.

Since private DNS names are turned off, the automatically created VPC endpoint DNS names, which end in .vpce.amazonaws.com, must be used for resolving the endpoint from both within VPCs and from on-premises after the appropriate DNS forwarding/resolution is set up. If private hosted zones are configured correctly, the queries will resolve to the private IP addresses of the SQS interface endpoint.

Why other options are incorrect:

A: Enabling private DNS names on the endpoint, while simplifying resolution within the VPC where the endpoint is created, doesn't solve the cross-VPC or on-premises resolution problem without additional configuration.

D: Assuming that using the automatically created private hosted zone will solve the problem without manual intervention is incorrect. The hosted zone needs manual association.

E: Using the public DNS name defeats the purpose of using the private interface endpoint, as it would route traffic over the public internet, which the scenario is designed to avoid.

Supporting Concepts:

Interface VPC Endpoints: Allow you to connect to AWS services privately from your VPC, without exposing your traffic to the public internet. (https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html) **Private Hosted Zones:** Let you create private DNS records that are only visible within your specified VPCs.

(https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/private-hosted-zones.html)
Route 53 Resolver: Facilitates hybrid cloud DNS by enabling conditional forwarding between your on-premises DNS servers and Route 53.
(https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver.html)

By creating a private hosted zone, adding the interface endpoint record, associating the private hosted zones and turning off the private DNS name option, the network engineer ensures that client applications, both on-premises and in different VPCs, can resolve the SQS endpoint privately.

Question: 80

A company's network engineer builds and tests network designs for VPCs in a development account. The company needs to monitor the changes that are made to network resources and must ensure strict compliance with network security policies. The company also needs access to the historical configurations of network resources.

Which solution will meet these requirements?

A.Create an Amazon EventBridge (Amazon CloudWatch Events) rule with a custom pattern to monitor the account for changes. Configure the rule to invoke an AWS Lambda function to identify noncompliant resources. Update an Amazon DynamoDB table with the changes that are identified.

B.Create custom metrics from Amazon CloudWatch logs. Use the metrics to invoke an AWS Lambda function to identify noncompliant resources. Update an Amazon DynamoDB table with the changes that are identified.

C.Record the current state of network resources by using AWS Config. Create rules that reflect the desired configuration settings. Set remediation for noncompliant resources.

D.Record the current state of network resources by using AWS Systems Manager Inventory. Use Systems Manager State Manager to enforce the desired configuration settings and to carry out remediation for noncompliant resources.

Answer: C

Explanation:

The correct answer is **C: Record the current state of network resources by using AWS Config. Create rules that reflect the desired configuration settings. Set remediation for noncompliant resources.**

Here's a detailed justification:

AWS Config continuously monitors and records the configuration of your AWS resources and allows you to automate the evaluation of recorded configurations against desired configurations. This is precisely what the company requires: monitoring changes to network resources and ensuring compliance with network security policies. Config provides historical configuration data and allows you to create Config Rules to check if resources are compliant.

Option A, using EventBridge and Lambda, can detect changes but requires writing custom code to analyze these changes against compliance rules. This is less efficient and more complex than using Config. While EventBridge can trigger actions based on events, it doesn't inherently maintain historical configuration data. Also, managing DynamoDB for storing these changes adds unnecessary overhead.

Option B, using CloudWatch logs and Lambda, is not designed for configuration management. CloudWatch logs primarily focus on monitoring application and system logs. Extracting configuration changes from logs and comparing them against rules would be extremely complex and inefficient.

Option D, using AWS Systems Manager Inventory and State Manager, is primarily intended for managing EC2 instances and their software configurations, rather than general network resource configurations. It can collect information about instances, but it isn't the best tool for overall network configuration compliance like VPCs and network security groups.

Config offers pre-built and custom rules for compliance checking. Remediation actions can be automated within Config to automatically fix non-compliant resources. This directly addresses both the monitoring and compliance requirements in a managed and scalable way.

In summary, AWS Config provides the comprehensive, managed solution best suited for continuous monitoring, historical tracking, and compliance enforcement for network resources, which aligns perfectly with the company's requirements.

Relevant links:

AWS Config: https://aws.amazon.com/config/
AWS Config Rules: https://docs.aws.amazon.com/config/latest/developerguide/config-rules.html