EXAM ASSOCIATE CLOUD ENGINEER TOPIC 1 QUESTION 60 DISCUSSION

Actual exam question from Google's Associate Cloud Engineer
Question #: 60
Topic #: 1
[All Associate Cloud Engineer Questions]

You have an application running in Google Kubernetes Engine (GKE) with cluster autoscaling enabled. The application exposes a TCP endpoint. There are several replicas of this application. You have a Compute Engine instance in the same region, but in another Virtual Private Cloud (VPC), called gce-network, that has no overlapping IP ranges with the first VPC. This instance needs to connect to the application on GKE. You want to minimize effort. What should you do?

A. 1. In GKE, create a Service of type LoadBalancer that uses the application's Pods as backend. 2. Set the service's externalTrafficPolicy to Cluster. 3. Configure the Compute Engine instance to use the address of the load balancer that has been created.

B. 1. In GKE, create a Service of type NodePort that uses the application's Pods as backend. 2. Create a Compute Engine instance called proxy with 2 network interfaces, one in each VPC. 3. Use iptables on this instance to forward traffic from gce-network to the GKE nodes. 4. Configure the Compute Engine instance to use the address of proxy in gce-network as endpoint.

C. 1. In GKE, create a Service of type LoadBalancer that uses the application's Pods as backend. 2. Add an annotation to this service: cloud.google.com/load-balancer-type: Internal 3. Peer the two VPCs together. 4. Configure the Compute Engine instance to use the address of the load balancer that has been created.

D. 1. In GKE, create a Service of type LoadBalancer that uses the application's Pods as backend. 2. Add a Cloud Armor Security Policy to the load balancer that whitelists the internal IPs of the MIG's instances. 3. Configure the Compute Engine instance to use the address of the load balancer that has been created.

Show Suggested Answer

by juancambb at *May 17, 2020, 6:23 p.m.*

Comments

Type your comment...

Submit

someoneinthecloud Highly Voted 4 years, 2 months ago

I believe it's A. It's never mentioned in the question that traffic cannot go through the Internet but it's mentioned that effort should be minimized. A requires a lot less effort than C to accomplish the same (no VPC peering, per example).

upvoted 61 times

ArtistS 1 year ago

A,C are ok for me. But this is a exam. Why the question mention the same region, no overlapping IP ranges means they suggest you to use VPC rather than public traffic. I 99% sure, if there is an official explaniation, there would be A is not correct there is a risk or error prone, sth like this.

upvoted 3 times

pgb54 2 years, 7 months ago

Totally agree. I had the same thought and looked through the question for any indication that the traffic must be private.

upvoted 2 times

ShakthiGCP 3 years, 7 months ago

Ans: A . This sounds correct and avoids unnecessary steps in C. C is also correct but compared to it, A is much easier to achieve. Go over Kubernetes Loadbalancer concepts to get more details. Initially i was thinking C is the Answer. but after putting some time on K8's Network - changed my mind to A.

upvoted 13 times

AmitKM 4 years, 2 months ago

Yeah, I feel the same. Nowhere does it say that the traffic has to be internal. But it does say "minimal effort" which I feel is option A.

upvoted 10 times

juancambb Highly Voted 4 years, 5 months ago

i think C is better solution, the solution A pass trafic trought public internet, also C by internal network and the "no overlap ips" in the statament suggest that.

upvoted 46 times

subha.elumalai Most Recent 5 months ago

Correct Answer: A

upvoted 1 times

DWT33004 6 months, 1 week ago

Selected Answer: A

Here's why Option A might be preferred over Option C:

Simplicity: Option A requires creating a LoadBalancer service in GKE and configuring the Compute Engine instance to use the load balancer's address. This is a straightforward setup and does not involve additional networking configurations.

Reduced Complexity: Peering two VPCs involves setting up and managing VPC peering configurations, which can be complex, especially if there are overlapping IP ranges. It also requires additional permissions and coordination between different teams.

Direct Connectivity: Option A provides direct connectivity between the Compute Engine instance and the application running in GKE through the load balancer. Peering VPCs might introduce additional network hops, potentially impacting latency and network performance.

Scalability and Flexibility: Using a LoadBalancer service in GKE allows for scalability and flexibility, as the load balancer can automatically scale to handle increased traffic and can be easily configured to adapt to changing requirements.

upvoted 3 times

edoo 8 months, 3 weeks ago

Selected Answer: C

Not A, exposing the service with an external LoadBalancer (externalTrafficPolicy set to Cluster) and not peering VPCs or

Not A, exposing the service with an external LoadBalancer (externalTrafficPolicy set to Cluster) and not peering VPCs is using an internal load balancer unnecessarily exposes the service to the internet, which is not required for inter-VPC communication and could lead to security concerns.
All the details in the question are pushing to answer C.

upvoted 2 times

ovokpus 1 year ago

Selected Answer: C

Option A suggests creating an external LoadBalancer. This is not the most efficient method because you're exposing your GKE application to the internet just to allow communication between two internal resources.

Option C suggests creating an internal LoadBalancer, which is the right approach. By using an internal LoadBalancer, the service is only exposed within the Google Cloud environment and won't be accessible from the internet. Peering the two VPCs ensures the two resources can communicate across the VPCs.

upvoted 4 times

ekta25 1 year ago

C. 1. In GKE, create a Service of type LoadBalancer that uses the application's Pods as backend. 2. Add an annotation to this service: cloud.google.com/load-balancer-type: Internal 3. Peer the two VPCs together. 4. Configure the Compute Engine instance to use the address of the load balancer that has been created.

upvoted 2 times

SinghAnc 1 year ago

Selected Answer: C

Correct Answer is C
Option A suggests setting the service's externalTrafficPolicy to Cluster. While this is a valid configuration, it's not directly related to the scenario described.

In the given scenario, the goal is to connect a Compute Engine instance from a different VPC to the application running in GKE. This involves networking configurations, peering the VPCs, and potentially setting up a LoadBalancer.

Setting the externalTrafficPolicy to Cluster primarily affects how traffic is balanced across Pods within the cluster, but it doesn't directly address the requirement of connecting an external instance from a different VPC.

upvoted 3 times

RobAlt 1 year, 1 month ago

Selected Answer: A

Minimal effort is the point.

upvoted 2 times

ExamsFR 1 year, 3 months ago

Selected Answer: C

Option C.

upvoted 1 times

KerolesKhalil 1 year, 4 months ago

Selected Answer: C

A is not correct
Because the GKE cluster and the instance are not in the same vpc , so without vpc peering traffic can't be established .
C is the correct answer.
Traffic still internal not exposed to internet , as they mentioned creating internal tcp loadbalancer not public one and created vpc peering . so no additional steps are needed.

upvoted 3 times

tempdir 1 year, 4 months ago

C is also correct but as the question states, minima effort. In A you dont need vpc peering since you will be using loadbalancer to expose the application, therefore, traffic can still be establishd b/n the two.

upvoted 1 times

sana_sree 1 year, 4 months ago

Selected Answer: C

Correct is C
please refer
https://www.youtube.com/watch?v=qx8PEmxKYzg

upvoted 3 times

DrLegendgun 1 year, 6 months ago

Selected Answer: C

The answer is C as two VPC needed to Peer first

upvoted 1 times

upvoted 1 times

esqandares 1 year, 6 months ago

Selected Answer: A

same region, but in another Virtual Private Cloud (VPC), called gce-network, that has no overlapping IP ranges with the first VPC.... need to read question again and again

upvoted 1 times

raselsys 1 year, 7 months ago

peering is lot easier effort but with dependent on not having overlapping IPs and that was clearly stated on the question. So C without any doubt is the correct answer here IMO.

upvoted 1 times

Buruguduystunstugudunstuy 1 year, 8 months ago

Selected Answer: A

Answer A is the correct solution.

In Answer A, we can create a Service of the type LoadBalancer in GKE that uses the application's Pods as a backend. This will create a Google Cloud load balancer with an external IP address that can be used to connect to the application. We can set the service's externalTrafficPolicy to Cluster to ensure that traffic is routed only to the nodes running the application. Then we can configure the Compute Engine instance to use the address of the load balancer that has been created.

upvoted 6 times

Buruguduystunstugudunstuy 1 year, 8 months ago

Answer B is not recommended because it requires the creation of an additional instance called a proxy, and the use of iptables to forward traffic from gce-network to the GKE nodes. This solution introduces additional complexity and potential points of failure.

Answer C is not recommended because it requires the peering of two VPCs. This solution is also more complex and requires additional configuration.

Answer D is not recommended because it involves using Cloud Armor to whitelist the internal IPs of the MIG's instances. This solution introduces additional complexity and potential security risks.

Therefore, Answer A is the most straightforward and least complex solution to connect the Compute Engine instance to the application running on GKE.

upvoted 6 times

Shivangi30 1 year, 3 months ago

externalTrafficPolicy is supported for internal LoadBalancer Services (via the TCP/UDP load balancer), but load balancing behavior depends on where traffic originates from and the configured traffic policy. Hence Answer is C as per link: https://cloud.google.com/kubernetes-engine/docs/how-to/service-parameters#externalTrafficPolicy

upvoted 2 times

antivrillee 1 year, 7 months ago

Option C requires less effort compared to option A.

In option A, you need to set the service's externalTrafficPolicy to Cluster, which means that the traffic will be load balanced across all nodes in the cluster, including those outside of the VPC network. You will also need to configure the Compute Engine instance to use the address of the load balancer that has been created.

In option C, you only need to add an annotation to the service with the value of "Internal", which will create an internal load balancer that is only accessible from within the VPC network. You will also need to peer the two VPCs together and configure the Compute Engine instance to use the address of the load balancer that has been created.

Therefore, option C requires less effort as it involves fewer steps and less configuration.

upvoted 2 times

Bobbybash 1 year, 8 months ago

Selected Answer: A

A is correct
Option A is the best solution to minimize effort. In GKE, creating a Service of type LoadBalancer that uses the application's Pods as backend and setting the service's externalTrafficPolicy to Cluster will expose the TCP endpoint of the application with a public IP address. Then, configuring the Compute Engine instance to use the address of the load balancer that has been created will allow it to connect to the application on GKE. Option B requires creating a separate instance as a proxy and using iptables to forward traffic, which adds unnecessary complexity. Option C involves peering the two VPCs together, which may not be desirable or feasible in all cases. Option D adds additional complexity by adding a Cloud Armor Security Policy to the load balancer.

upvoted 3 times

Load full discussion...