KUBERNETES: SERVICES, DEPLOYMENTS, STORAGE**Introduction & Learning Objectives**

KUBERNETES: SERVICES, DEPLOYMENTS, STORAGE

### Knowledge Check

Congratulations on completing Chapter 12 - Kubernetes: Services, Deployments, Storage. Take this quiz to check your understanding of the concepts you've learned about so far.

KUBERNETES: SERVICES, DEPLOYMENTS, STORAGEIntroduction & Learning Objectives KUBERNETES: SERVICES, DEPLOYMENTS, STORAGE Knowledge Check Congratulations on completing Chapter 12 - Kubernetes: Services, Deployments, Storage. Take this quiz to check your understanding of the concepts you've learned about so far. Question 12.1 Which of the following is the default type of Service? A. NodePort B. LoadBalancer C. ClusterIP D. ExternalIP

ChatGPT

C. ClusterIP

The default type of Service in Kubernetes is **ClusterIP**. When you create a Service without specifying a type, it will be assigned a ClusterIP. A ClusterIP service makes the service only reachable from within the cluster and is not exposed to the external network. It provides a stable virtual IP address for the Service within the cluster. Other Pods and Services within the cluster can access the Service using this ClusterIP.

The other Service types are:

* **NodePort**: Exposes the Service on a static port on each Node's IP. It opens a high port on all the Nodes and forwards traffic to the Service.
* **LoadBalancer**: Creates an external load balancer (in cloud environments) that forwards traffic to the Service. It is primarily used in cloud environments with load balancer support.
* **ExternalIP**: Maps the Service to an externally accessible IP address. It is not supported on all cloud providers and usually requires manual configuration.

You can specify the type of Service you want when creating the Service using the **type** field in the Service manifest. For example:

yamlCopy code

apiVersion: v1 kind: Service metadata: name: my-service spec: selector: app: MyApp ports: - protocol: TCP port: 80 targetPort: 8080 type: NodePort



Question 12.2 Service acts as a load balancer for your application. True or False? A. True B. False

ChatGPT

A. True

Yes, that's true. A Service in Kubernetes acts as a load balancer for your application. When you create a Service, it sets up a stable virtual IP address (ClusterIP) that can be used to access the application or microservice running inside the cluster. The Service acts as a single entry point to access multiple replicas (Pods) of the application.

When multiple replicas of the application are deployed as part of a Deployment or ReplicaSet, the Service automatically load balances the incoming traffic across these replicas. It distributes requests among the available Pods, ensuring that each replica receives a fair share of the traffic. This load balancing feature helps in distributing the workload evenly and providing high availability for the application.

The type of Service you choose (ClusterIP, NodePort, LoadBalancer, or ExternalIP) determines the level of exposure and accessibility of your application to the outside world. For example, the **LoadBalancer** type of Service can create an external load balancer in cloud environments to handle incoming traffic from the internet.

In summary, a Service in Kubernetes provides load balancing capabilities, which is essential for running applications in a scalable and highly available manner.



Question 12.3 What's the range of NodePorts? A. 30000-32767 B. 30000-32657 C. 30000-33000 D. 30000-32667

ChatGPT

A. 30000-32767

In Kubernetes, the NodePort range is 30000-32767. When you create a Service of type **NodePort**, it exposes the Service on a static port in this range on each Node in the cluster. Any external traffic that reaches the Node's IP on that specific port will be forwarded to the Service, and Kubernetes will handle the internal routing to the appropriate Pods.

You can specify the desired NodePort in the Service definition, or if you omit it, Kubernetes will automatically allocate a free NodePort from the allowed range. For example:

yamlCopy code

apiVersion: v1 kind: Service metadata: name: my-service spec: selector: app: MyApp ports: - protocol: TCP port: 80 targetPort: 8080 nodePort: 31000 type: NodePort

In this example, the Service **my-service** will be accessible on port 31000 on each Node in the cluster, and external traffic arriving at any Node's IP on port 31000 will be directed to the Service. If you don't specify the **nodePort** field, Kubernetes will automatically assign a free NodePort from the range 30000-32767.



Question 12.4 Service selects pods based on the \_\_\_\_\_\_\_\_\_. Fill in the blank. A. Labels of the pods B. Service name C. Service type D. None of the above

ChatGPT

A. Labels of the pods

In Kubernetes, a Service selects pods to include in its load balancing and routing based on the labels associated with those pods. When you create a Service, you define a set of labels to specify which pods should be part of the Service. These labels act as selectors that link the Service to the pods that match the specified labels.

For example, when defining a Service, you might use the **selector** field to specify the labels to identify the pods:

yamlCopy code

apiVersion: v1 kind: Service metadata: name: my-service spec: selector: app: MyApp environment: production ports: - protocol: TCP port: 80 targetPort: 8080

In this example, the Service **my-service** will include all pods that have both the **app=MyApp** and **environment=production** labels. These labels must be set on the pods for them to be part of the Service. The Service will then load balance incoming traffic across all pods that match the specified labels, ensuring high availability and scalability for the application.



Question 12.5 Which of the following service types expose application to the outside world? Select all answers that apply. A. ClusterIP B. ExternalIP C. NodePort D. LoadBalancer

ChatGPT

The service types that expose applications to the outside world are:

B. ExternalIP C. NodePort D. LoadBalancer

Explanation:

* B. ExternalIP: This type allows you to manually specify an external IP address to be used by the Service. It is useful when you have a specific external IP address that you want to use to expose your Service.
* C. NodePort: This type exposes the Service on a static port on each Node in the cluster. Any external traffic that reaches the Node's IP on that specific port will be forwarded to the Service.
* D. LoadBalancer: This type provisions an external load balancer (usually provided by the cloud provider) to distribute traffic to the Service. It automatically creates a cloud load balancer and maps the Service to it, allowing external access to the application.

A. ClusterIP is not used to expose the application to the outside world. It provides an internal IP address that is only accessible within the cluster. It allows other services within the cluster to access the Service but does not expose the Service to external traffic.



Question 12.6 NodePort is accessible from \_\_\_\_\_\_\_\_\_. Fill in the blank. A. All the nodes in the cluster B. The localhost C. Selected nodes in the cluster D. The master machine

ChatGPT

A. All the nodes in the cluster

NodePort is a Kubernetes Service type that exposes the Service on a static port on each Node in the cluster. This means that the Service is accessible from all the nodes in the cluster on the designated NodePort. Any external traffic that reaches the Node's IP on that specific port (NodePort) will be forwarded to the Service, allowing access to the application running in the Service from any node in the cluster.



Question 12.7 Which of the following are the deployment strategy types? Select all answers that apply. A. Rolling Update B. Recreate C. Blue Green D. Canary E. All of the above

ChatGPT

The deployment strategy types in Kubernetes are:

A. Rolling Update B. Recreate C. Blue Green D. Canary

E. All of the above

Explanation:

* A. Rolling Update: This strategy gradually replaces instances of the old version of the application with instances of the new version, ensuring high availability during the update process.
* B. Recreate: This strategy stops all instances of the old version and creates new instances of the new version all at once. It does not provide high availability during the update process.
* C. Blue Green: This strategy deploys a new version of the application alongside the old version. Once the new version is ready and tested, the traffic is switched from the old version to the new version all at once.
* D. Canary: This strategy gradually shifts traffic from the old version to the new version in a controlled manner, allowing a subset of users to access the new version and monitor its performance before fully deploying it.

So, the correct answer is E. All of the above.



Question 12.8 Which of the following is the default Deployment strategy? A. Recreate B. Blue Green C. Rolling Update D. Canary

ChatGPT

C. Rolling Update

In Kubernetes, the default Deployment strategy is Rolling Update. This strategy gradually updates the Pods in the Deployment to the new version while maintaining high availability. It ensures that there is no downtime during the update process by gradually replacing instances of the old version with instances of the new version. This way, users and applications continue to access the service during the update process.



Question 12.9 During the Rolling Update, maxSurge is set to 3 and maxUnavailable is set to 2. What will be the batch size of the pods that get updated? A. 2 B. 3 C. 4 D. 5

ChatGPT

The batch size during a Rolling Update is determined by the sum of **maxSurge** and **maxUnavailable**.

In this case:

* **maxSurge** is set to 3, which means Kubernetes can create up to 3 additional replicas (pods) beyond the desired number of replicas during the update.
* **maxUnavailable** is set to 2, which means Kubernetes can take down up to 2 replicas (pods) during the update.

To calculate the batch size, we add **maxSurge** and **maxUnavailable**: 3 + 2 = 5.

So, the batch size of the pods that get updated during the Rolling Update will be 5.



Question 12.10 Which of the following commands should be used to roll back to a previous version of Deployment? A. kubectl rollback deploy/<name> --revision=<number> B. kubectl rollout undo deploy/<name> --revision=<number> C. kubectl rollout undo deploy/<name> --to-revision=<number> D. None of the above

ChatGPT

C. kubectl rollout undo deploy/<name> --to-revision=<number>

To roll back to a previous version of a Deployment in Kubernetes, you can use the **kubectl rollout undo** command with the **--to-revision** flag followed by the revision number of the desired version. The correct command would be:

phpCopy code

kubectl rollout undo deploy/<name> --to-revision=<number>

# Question 12.7

Which of the following are the deployment strategy types? Select all answers that apply.

Incorrect Answer

* **A.** Rolling Update

Correct Answer

* **B.** Recreate

Correct Answer

* **C.** Blue Green
* **D.** Canary
* **E.** All of the above

Your Answer:

Incorrect

# Question 12.5

Which of the following service types expose application to the outside world? Select all answers that apply.

Incorrect Answer

* **A.** ClusterIP
* **B.** ExternalIP

Your Answer:

Correct

* **C.** NodePort

Correct Answer

* **D.** LoadBalancer

Correct Answer