

Kubernetes from Basic to Advanced



kubernetes

Session #07

Other Workloads



kubernetes

Session Contents



- DaemonSets
- Jobs & CronJobs
- StatefulSets

DaemonSets



DaemonSets



- Daemonset is another controller that manages pods like Deployments or ReplicaSets
- DaemonSet ensures that all (or some) Nodes run a copy of a Pod
- As nodes are added to the cluster, Pods are added to them
- As nodes are removed from the cluster, those Pods are garbage collected
- Deleting a DaemonSet will clean up the Pods it created.

DaemonSets: Typical Uses



- Running a cluster storage daemon on every node
- Running a logs collection daemon on every node
- Running a node monitoring daemon on every node

Demo | DaemonSets



Jobs & CronJobs



Motivation



- When Kubernetes controller detects a container has failed, it will attempt to restart it based on the container's restartPolicy: Always, Never or OnFailure
- When controlled by a Deployment, containers in Pods are expected to run continuously. Thus, the only restart policy a Deployment supports is Always.
- However, what if you want to run a container that's expected to perform a finite task and then just stop? Just like an Init Container, but without creating any other container afterwards?

Job



- A Job is a Kubernetes controller that creates and run task-based workload.
- A Job creates one or more Pods and will continue to retry execution of the Pods until a specified number of them successfully terminate.
- As Pods successfully complete, the Job tracks the successful completions. When a specified number of successful completions is reached, the task (i.e., Job) is complete.
- Deleting a Job will clean up the Pods it created.
- Containers defined as part of a Job only support Never and OnFailure restart policies.
- To make changes you need to delete and recreate

Job Types



- Multiple Parallel Jobs: Run multiple jobs (pods) in parallel to make it faster (when possible) to process. This can be set using **parallelism** property
- Parallel Jobs with Fixed Completion Count: These jobs occur concurrently but run a set number of times before terminating successfully. By setting **completions** to a value greater than one, you trigger the formation of successful pods.
- Non-parallel Jobs: This specifies a job that executes single-handedly or independently. Only one successful pod is started, with additional pods forming in response to any startup failures. Once a pod terminates successfully, that specific job is complete.



CronJob



- A CronJob is a Kubernetes controller that creates and manages Jobs according to a cron schedule.
- CronJobs are useful for creating periodic and recurring tasks, like running backups or sending emails.
- CronJobs maintain a certain number of successful and failed jobs, so their logs can be examined.
- When a CronJob deletes a Job, that Job's Pods are deleted.
- When a CronJob is deleted, the Jobs it created (and their Pods) are deleted.



Demo | Jobs & CronJobs



StatefulSets



Motivation



- Using Deployments you need to build your application (or components) on a stateless approach
- Your pods can be deleted and recreated without having a consistency on names and IPs
- Additionally, some solutions to have an associated volume to be always available when running and not depending from a PVC defined on another resource
- For these needs, you should use a StatefulSet

StatefulSet



- StatefulSet is a pod controller designed to manage stateful applications
- Unlike Deployments, a StatefulSet maintain a sticky identity for each of its Pods
- Pods are created from the same spec but are not interchangeable: each Pod has a persistent identifier that it maintains across rescheduling
- Pods are created in order and are assigned sequential numbers, starting with 0...N-1.



StatefulSet



- StatefulSets define a Volume Claim Templates, which define parameters to use when dynamically creating PersistentVolumeClaims and PersistentVolumes for each Pod.
- Pod order and persistent volume claims are maintained when Pods are deleted.
- Replacement Pods are linked to the same PVCs.
- Deleting Pods in a StatefulSet (or the StatefulSet itself) will use by default retain reclaim policy
- Can be configured delete reclaim policy when scale down or delete the StatefulSet

Headless Service

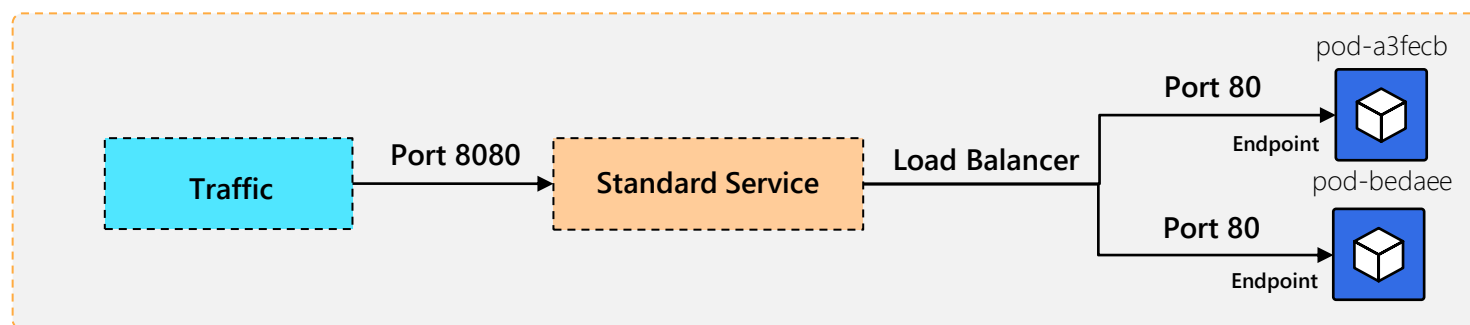


- A “headless” service (**clusterIP=None**) is required for StatefulSets.
- The service is referenced by the Pod definition and allows direct access to each without load balancing.
- Creates a DNS entry for each Pod instead of just for the Service.
- Pods is accessed individually through the headless service using DNS and the FQDN address.
 - Fully Qualified domain name: `{pod}.{service}.{namespace}.svc.cluster.local`
 - Example: `mysql.connect("mysql-02.db-service.micro.svc.cluster.local")`

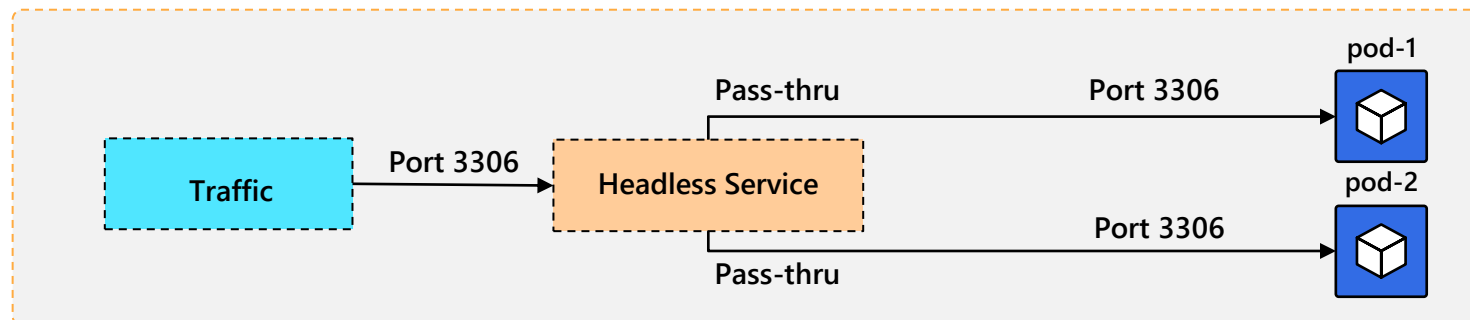
Headless Service



A standard Service load balances traffic across all Pods listed as Endpoints



A Headless Service doesn't have Endpoints. It routes traffic directly to specific Pods



Demo | StatefulSets



Questions?



kubernetes

Lab #06: Other workloads



kubernetes