**Memcached**

[Memcached](http://memcached.org/) is a free and open-source **high-performance memory caching system**.

It’s typically **used to cache** database data, API calls or page rendering chunks in **RAM** to increase the application performance.

Through its **multi-thread architecture**, Memcached can also use several process kernels simultaneously.

Conventional databases store data on a hard drive or preferably a solid-state drive ([SSD](https://www.ionos.com/digitalguide/server/know-how/what-is-ssd/)). With Memcached, on the other hand, data is stored in memory so that it can be retrieved in **microseconds**, eliminating the delay caused by seek times when retrieving data.

While it is possible to store data for a prolonged period, most data are automatically deleted after a certain amount of time. This is because Memcached is **just a cache** and not a database in the traditional sense. The least requested data is thus deleted as soon as there is no more space for new data.

### **Simple Key/Value Store**

### **Logic Half in Client, Half in Server**

### **Servers are Disconnected From Each Other**

### **Complexity: O(1)**

### **Forgetting is a Feature**

### **Cache Invalidation**

**Memcached Docs:**

https://github.com/memcached/memcached/wiki

**Blogs:**

<https://www.ionos.com/digitalguide/hosting/technical-matters/what-is-memcached/>

<https://medium.com/swlh/what-is-memcached-d1498623db3b>

<https://memcached.org/blog>

Memcached is also described as an **in-memory key-value store**. Using the [TCP and IP](https://www.ionos.com/digitalguide/server/know-how/introduction-to-tcpip/) protocols, a connection is first established with the server. If a user wants to retrieve specific data, Memcached will check whether it is available in the cache. If it is not, the required data will be retrieved from the main memory. The client then provides a key value for the data in question which is generated by the software library. Using a **hashing algorithm**, the client then determines which Memcached server the data, which is in the form of character strings, is to be stored on. Below are the five most important characteristics of Memcached:

* Data is only sent to one server.
* Data is stored as key-value pairs.
* The various servers do not share data with each other.
* The server only stores data in the memory.
* If there is not enough space, the server will delete the oldest data.

| Advantages | Disadvantages |
| --- | --- |
| Extremely fast response times thanks to the in-memory key-value store | Only stores data temporarily and loses that data if a Memcached instance fails |
| Multi-thread architecture enables vertical scaling of computing capacity | Data cannot be viewed which makes debugging difficult |
| Sophisticated open-source solution with a publicly available datastore | Key length for values is limited to 250 characters (1 MB) |
| Easy to use and flexible in terms of application development | Lack of security features necessitates additional firewalls |
| Supports open data formats and most common clients and programming languages | Not redundant (i.e., no failover using mirroring or data backups) |

spec.terminationPolicy to **Halt** (which creates dormantdatabase when memcached is deleted and

keeps PVC, snapshots, Secrets intact) or remove this field (which default to Halt).

export LICENSE\_FILE= Path

Make Deploy-to-kind

Add Configure open-api

OOps manager

PVC baad e sob delete- halt

Governing service- headless service

Container init -> debug

# **Init Containers**

This page provides an overview of init containers: specialized containers that run before app containers in a [Pod](https://kubernetes.io/docs/concepts/workloads/pods/). Init containers can contain utilities or setup scripts not present in an app image.

You can specify init containers in the Pod specification alongside the containers array (which describes app containers).

In Kubernetes, a [sidecar container](https://kubernetes.io/docs/concepts/workloads/pods/sidecar-containers/) is a container that starts before the main application container and *continues to run*. This document is about init containers: containers that run to completion during Pod initialization.

Ping -> **telnet service\_name.namespace\_name.service Port\_number**

**kubectl exec -it ubuntu -n demo -- sh**

**apt-get update**

**apt-get install telnet**

**# telnet memcd-quickstart.demo.svc 11211**

Trying 10.96.217.212...

Connected to memcd-quickstart.demo.svc.cluster.local.

Escape character is '^]'.

**set my\_key 0 2592000 1**

**2**

STORED

**get my\_key**

VALUE my\_key 0 1

2

END

**quit**

Connection closed by foreign host.

**#**

**make gen ->** After changing Api-machinary’s [types.go/helper.go](http://types.go/helper.go)

**make gen fmt** -> For formatting

**make gen fmt lint**

**Memcached-Configuration-File**

<https://www.dragonflydb.io/faq/how-to-set-up-memcached>

sudo systemctl edit memcached.service

sudo systemctl status memcached.service

journalctl -xeu memcached.service

sudo systemctl restart memcached

**Custom-Configuration:**

kubectl apply -f mc-configuration.yaml

In memcached yaml,

**configSecret**:

**name**: mc-configuration

Must have this field.

kubectl port-forward -n Namespace\_Name Pod\_Name 11211

telnet 127.0.0.1 11211

**Vertical Scaling:**

IF Ops\_Request\_Phase\_Pending ->

* Update\_Memcached\_Ops\_Request

IF Ops\_Request\_Not\_SuccessFul ->

* Pause\_Memcached Database
* Waiting For Pause Request by Provisioner
* Update StatefulSet ->
* Update StatefulSet Resources
* Update Memcached Ops Request Condition
* Re-start Pods
* Check is\_Pod\_Resource\_Updated

Vertical Scaling Succeed ->

* Update Memcached
* Resume Memcached Database
* Update Memcached Ops Request Status to Success

**Restart:**

IF Ops\_Request\_Phase\_Pending ->

* Update\_Memcached\_Ops\_Request
* Restart Pods

IF Ops\_Request\_Not\_SuccessFul ->

* Update Memcached Ops Request Condition

**Reconfiguration:**

**STATS ->**

kubectl exec -it memcd-quickstart-0 -n demo -- sh

echo "stats settings" | nc localhost 11211

**Memcached.conf ->**

usr/config/memcached.conf

**Monitoring:**

<https://code.google.com/archive/p/memcache-top/>