



**Getting started with Milvus - the world's  
most popular open-source vector  
database.**

@Milvus.io

## Getting Started with Vector Database - Milvus Quickstart

Hey there - welcome back to Milvus codelabs. In the previous tutorial, we provided a brief introduction to Milvus, Milvus' history, as well as the primary differences between Milvus 1.x and Milvus 2.x. We also took a quick tour of the architecture of Milvus 2.x and helped shine some light on how Milvus' architecture allows it to implement all of the required features of vector databases.

### Let's get started

If you haven't read the previous tutorials (unstructured data, vector databases, Milvus introduction), I recommend you go ahead and read them. If you have, great. Let's get started with Milvus!

We offer two different modes of deployment: standalone and cluster. In Milvus standalone, all nodes - coordinators, worker nodes, and forward-facing proxies - are deployed as a single instance. For persistent data and metadata, Milvus standalone relies on [MinIO](#) and [etcd](#), respectively. In future releases, we hope to eliminate these two third-party dependencies, allowing everything to run in a single process and removing the need to install third-party dependencies.

Milvus cluster is our full-fledged version of Milvus, complete with separate instances/pods for all eight microservice components along with three third-party dependencies: [MinIO](#), [etcd](#), and [Pulsar](#) (Pulsar serves as the log broker and provides log pub/sub services). If you haven't gotten the chance to take a look at the Milvus overview from the previous slide, please do so! It'll help clarify what each of these third party dependencies is used for and why we've included them in Milvus cluster.

### Milvus standalone (docker-compose)

Milvus standalone is meant to be super easy to install. In this section, we'll go over how [docker-compose](#) can be used to install Milvus. You can view the recommended prerequisites [here](#).

Let's first download the [docker-compose.yml](#) configuration file needed for the standalone installation. If you're on any Debian-based Linux (including Ubuntu), you can use the following command:

```
1 $ wget https://github.com/milvus-io/milvus/releases/download/v2.0.2/
   milvus-standalone-docker-compose.yml -O docker-compose.yml
```

```
1 Resolving objects.githubusercontent.com (objects.githubusercontent.
   com)... 185.199.108.133, 185.199.111.133, 185.199.109.133, ...
```

Getting started with Milvus - the world's most popular open-source vector database.

---

```
2    Connecting to objects.githubusercontent.com (objects.
    githubusercontent.com)|185.199.108.133|:443... connected.
3    HTTP request sent, awaiting response... 200 OK
4    Length: 1303 (1.3K) [application/octet-stream]
5    Saving to: docker-compose.yml
6
7    docker-compose.yml  100%[=====>]   1.27K  --.-KB/s
    in 0s
8
9    2022-06-29 13:58:49 (113 MB/s) - docker-compose.yml saved
    [1303/1303]
```

Alternatively, if you're on MacOS, make sure you have Docker Desktop installed first. I recommend using `brew`:

```
1 % brew install --cask docker
```

You can then follow this up with the command below:

```
1 % curl https://github.com/milvus-io/milvus/releases/download/v2.0.2/
    milvus-standalone-docker-compose.yml -o docker-compose.yml
```

With everything ready, we can now spin up our Milvus standalone instance:

```
1 $ docker-compose up -d
```

Docker Compose is now in the Docker CLI, try `docker compose up`

```
1    Creating milvus-etcd    ... done
2    Creating milvus-minio   ... done
3    Creating milvus-standalone ... done
```

Now, we can check on the status of our containers

```
1 $ docker ps -a
```

|   | CONTAINER ID             | IMAGE                                    | CREATED        | STATUS                  | COMMAND           |
|---|--------------------------|--|----------------|-------------------------|-------------------|
| 2 | 711d54ab15c7             | milvusdb/milvus:v2.0.2                   | 42 seconds ago | Up 40 seconds           | "/tini --         |
|   |                          | milvus run."                             |                |                         |                   |
|   | 0.0.0.0:19530->19530/tcp |  |                |                         | milvus-standalone |
| 3 | 0d85f4927864             | minio/minio:RELEASE.2020-12-03T00-03-10Z | 42 seconds ago | Up 40 seconds (healthy) | "/usr/bin         |
|   |                          | /docker-ent."                            |                |                         | 9000/             |
|   |                          | tcp                                      |                |                         | milvus-minio      |
| 4 | 99de39278b35             | quay.io/coreos/etcd:v3.5.0               | 42 seconds ago | Up 40 seconds           | "etcd -           |
|   |                          | advertise-cli."                          |                |                         |                   |
|   | 2379-2380/tcp            |  |                |                         | milvus-etcd       |

Here's a quick rundown of what each of the containers are doing. `milvus-standalone` is the com-

Getting started with Milvus - the world's most popular open-source vector database.

---

piled/compressed version of Milvus, mean to run on a single machine.

To stop Milvus standalone, run:

```
1 $ docker-compose down
```

And that's it for Milvus standalone! Easy, right?

## Milvus standalone (apt)

We also provide a handy `apt` package for Debian-based distributions. Simply run:

```
1 $ sudo apt install software-properties-common
2 $ sudo add-apt-repository ppa:milvusdb/milvus
3 $ sudo apt update
4 $ sudo apt install milvus
```

Once that's done, you're good to go. You can check the status of the running services with:

```
1 $ sudo systemctl status milvus
2 $ sudo systemctl status milvus-etcd
3 $ sudo systemctl status milvus-minio
```

## Milvus cluster

From the previous tutorial, we know that Milvus is composed of four primary components: the access layer, coordinator service, worker nodes, and object storage. Requests are sent to a cluster of proxies in the access layer, which then forwards the requests to either the coordinator layer or a streaming service for vector data. The stateful coordinator nodes within the coordinator service manage and control all of the stateless worker nodes, allowing for easy horizontal scaling. Object storage is accomplished via S3 or any "S3-like" storage layer, allowing Milvus to be run both in the cloud and on-premises via MinIO.

Milvus' remaining third-party dependencies, Pulsar/Kafka and etcd, are also distributed and cloud-native, allowing the entirety of Milvus to run via Kubernetes as an orchestration engine. Using Kubernetes is a no-brainer for nearly all distributed applications, as it provides out-of-the-box support for application deployment, maintenance, and scaling. We recommend deploying Milvus as a Kubernetes application via Helm:

```
1 % helm repo add milvus https://milvus-io.github.io/milvus-helm/
```

"milvus" has been added to your repositories

Now, let's grab the latest Milvus chart from the `milvus-io/milvus-helm` repository.

Getting started with Milvus - the world's most popular open-source vector database.

---

```
1 % helm repo update
```

```
1      Hang tight while we grab the latest from your chart repositories...
2      ...Successfully got an update from the "milvus" chart repository
3      Update Complete. Happy Helming
```

Great. Now that we've gotten all of the dependencies out of the way, let's install Milvus (cluster)!

```
1 % helm install my-release milvus/milvus
```

```
1      W0629 16:01:00.674407    21803 warnings.go:70 policy/v1beta1
      PodDisruptionBudget is deprecated in v1.21+, unavailable in v1
      .25+; use policy/v1 PodDisruptionBudget
2      W0629 16:01:00.676536    21803 warnings.go:70 policy/v1beta1
      PodDisruptionBudget is deprecated in v1.21+, unavailable in v1
      .25+; use policy/v1 PodDisruptionBudget
3      W0629 16:01:00.678594    21803 warnings.go:70 policy/v1beta1
      PodDisruptionBudget is deprecated in v1.21+, unavailable in v1
      .25+; use policy/v1 PodDisruptionBudget
4      W0629 16:01:00.680671    21803 warnings.go:70 policy/v1beta1
      PodDisruptionBudget is deprecated in v1.21+, unavailable in v1
      .25+; use policy/v1 PodDisruptionBudget
5      W0629 16:01:00.808448    21803 warnings.go:70 policy/v1beta1
      PodDisruptionBudget is deprecated in v1.21+, unavailable in v1
      .25+; use policy/v1 PodDisruptionBudget
6      W0629 16:01:00.809339    21803 warnings.go:70 policy/v1beta1
      PodDisruptionBudget is deprecated in v1.21+, unavailable in v1
      .25+; use policy/v1 PodDisruptionBudget
7      W0629 16:01:00.809344    21803 warnings.go:70 policy/v1beta1
      PodDisruptionBudget is deprecated in v1.21+, unavailable in v1
      .25+; use policy/v1 PodDisruptionBudget
8      W0629 16:01:00.809594    21803 warnings.go:70 policy/v1beta1
      PodDisruptionBudget is deprecated in v1.21+, unavailable in v1
      .25+; use policy/v1 PodDisruptionBudget
9      NAME: my-release
10     LAST DEPLOYED: Wed Jun 29 16:01:00 2022
11     NAMESPACE: default
12     STATUS: deployed
13     REVISION: 1
14     TEST SUITE: None
```

With this done, we can now see the pods that are up and running via `kubectl`:

```
1 $ kubectl get pods
```

| 1 | NAME                       | READY | STATUS    |
|---|----------------------------|-------|-----------|
|   | RESTARTS AGE               |       |           |
| 2 | my-release-etcd-0<br>2m23s | 1/1   | Running 0 |

|    |  |     |           |   |
|----|--|-----|-----------|---|
| 3  | my-release-etcd-1<br>2m23s                               | 1/1 | Running   | 0 |
| 4  | my-release-etcd-2<br>2m23s                               | 1/1 | Running   | 0 |
| 5  | my-release-milvus-datacoord-6fd4bd885c-gkzwx<br>2m23s    | 1/1 | Running   | 0 |
| 6  | my-release-milvus-datanode-68cb87dcbd-4khpm<br>2m23s     | 1/1 | Running   | 0 |
| 7  | my-release-milvus-indexcoord-5bfcf6bdd8-nmh5l<br>2m23s   | 1/1 | Running   | 0 |
| 8  | my-release-milvus-indexnode-5c5f7b5bd9-l8hjg<br>2m24s    | 1/1 | Running   | 0 |
| 9  | my-release-milvus-proxy-6bd7f5587-ds2xv<br>2m24s         | 1/1 | Running   | 0 |
| 10 | my-release-milvus-querycoord-579cd79455-xht5n<br>2m24s   | 1/1 | Running   | 0 |
| 11 | my-release-milvus-querynode-5cd8fff495-k6gtg<br>2m24s    | 1/1 | Running   | 0 |
| 12 | my-release-milvus-rootcoord-7fb9488465-dmbbj<br>2m23s    | 1/1 | Running   | 0 |
| 13 | my-release-minio-0<br>2m23s                              | 1/1 | Running   | 0 |
| 14 | my-release-minio-1<br>2m23s                              | 1/1 | Running   | 0 |
| 15 | my-release-minio-2<br>2m23s                              | 1/1 | Running   | 0 |
| 16 | my-release-minio-3<br>2m23s                              | 1/1 | Running   | 0 |
| 17 | my-release-pulsar-autorecovery-86f5dbdf77-lchpc<br>2m24s | 1/1 | Running   | 0 |
| 18 | my-release-pulsar-bookkeeper-0<br>2m23s                  | 1/1 | Running   | 0 |
| 19 | my-release-pulsar-bookkeeper-1<br>98s                    | 1/1 | Running   | 0 |
| 20 | my-release-pulsar-broker-556ff89d4c-2m29m<br>2m23s       | 1/1 | Running   | 0 |
| 21 | my-release-pulsar-proxy-6fbd75db75-nhg4v<br>2m23s        | 1/1 | Running   | 0 |
| 22 | my-release-pulsar-zookeeper-0<br>2m23s                   | 1/1 | Running   | 0 |
| 23 | my-release-pulsar-zookeeper-metadata-98zbr<br>2m24s      | 1/1 | Completed | 0 |

That's it! You now have Milvus installed directly on your on-premises cluster. Check out our next tutorial to see how to create a collection within Milvus and begin inserting and querying embeddings.

If you're interested in running Milvus on cloud infrastructure check out the Milvus standalone on AWS Marketplace.

## Wrapping up

In this tutorial, we took a look at how to install the standalone version of Milvus (via [docker-compose](#)) and the cluster version of Milvus (via [helm](#)). The standalone version is suitable for testing purposes, while the cluster version is suitable for internal clusters or on-premises deployments. In the next tutorial, we'll look at basic Milvus operations: connecting to a Milvus server, creating a collection (equivalent to a table in relational databases), creating a partition within the collection, inserting embedding vector data, and conducting a vector search.

See you in the next couple of tutorials.