Oracle® NoSQL Database

Quick Start to KVLite

Release 26.1
E87605-39
April 2026

Oracle NoSQL Database Quick Start to KVLite, Release 26.1

E87605-39

Contents

List of Tables

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

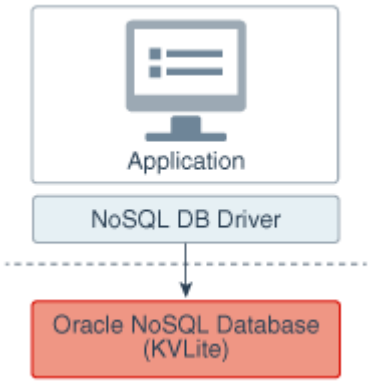
Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Introduction

The Oracle NoSQL Database is a scalable, distributed NoSQL database, designed to provide highly reliable, flexible and available data management across a configurable set of storage nodes. It consists of two parts - a NoSQL DB Driver and a collection of storage nodes called the data store. The NoSQL DB Driver is an intelligent driver that transparently handles all the core operations of Oracle NoSQL Database, and the data store consists of storage nodes.

KVLite is a simplified version of the Oracle NoSQL Database. It provides a single storage node, single shard store, that is not replicated. It runs in a single process without requiring any administrative interface. You configure, start, and stop KVLite using a command line interface.



Note: KVLite is intended for use by application developers who want to develop and unit test their Oracle NoSQL Database applications. It can be used as a development platform for developers to get familiar with Oracle NoSQL APIs, and test different ways of interacting with these APIs. KVLite runs on a single system. It is not intended for production deployment, or for performance measurements.

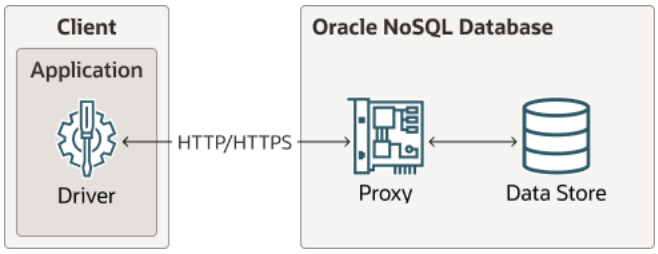
Also, KVLite is secure by default. If you want to run KVLite in non-secure mode, you will have to explicitly provide parameters to disable security while installing KVLite as demonstrated in this guide.

This Quick Start guide demonstrates how to perform the following tasks:

1. About the Oracle NoSQL Database Proxy
2. Install KVLite
3. Start KVLite
4. Verify your Installation
5. Stop and Restart KVLite

About the Oracle NoSQL Database Proxy

The Oracle NoSQL Database Proxy is a middle-tier component that lets the Oracle NoSQL Database SDK communicate with the Oracle NoSQL Database (kvlite configuration).



The Oracle NoSQL Database drivers are available in various programming languages that are used in the client application. The Oracle NoSQL Database Proxy is a server that accepts requests from the client application and processes them using the Oracle NoSQL Database. The JAR file (httpproxy.jar) for the Oracle NoSQL Database Proxy is included in the Enterprise Edition distribution and the Community Edition distribution of Oracle NoSQL Database that you downloaded. You can download the JAR for the Oracle NoSQL Database Proxy from the Oracle Technology Network.

After you start KVLite, you must run the following command to start up the proxy.

- For a non-secure kvlite:

```
java -jar lib/httpproxy.jar \
-storeName <kvstore_name> \
-helperHosts <kvstore_helper_host> \
[-hostname <proxy_host>] \
[-httpPort <proxy_http_port>]
```

- For a secure kvlite:

```
java -jar lib/httpproxy.jar \
-storeName <kvstore_name> \
-helperHosts <kvstore_helper_host> \
[-hostname <proxy_host>] \
[-httpsPort <proxy_https_port>] \
-storeSecurityFile proxy/proxy.login \
-sslCertificate certificate.pem \
-sslPrivateKey key-pkcs8.pem \
-sslPrivateKeyPass <privatekey_password> \
[-verbose true]
```

For more information, see Oracle NoSQL Database Proxy.

Install KVLite

KVLite is bundled with the Oracle NoSQL Database software. To install KVLite, perform the following:

1. Download the tar.gz or .zip file (depending on your operating system) from Oracle NoSQL GitHub repository.
2. In a Linux operating system, use `gunzip` and `tar` commands to extract the `.tar.gz` package (or use `unzip` command if you downloaded the `.zip` package). Oracle NoSQL Database version 25.1.13 Community Edition is used in this example. The actual package names and directory names will change, depending upon the release version you are using, and whether you are using Community Edition (CE) or Enterprise Edition (EE).

   Also, make sure you meet the following requirements to run KVLite:

   - Install Java version 17 in your machine.
   - Maintain a minimum disk space of 5GB.

   ```
   $ gunzip kv-ce-25.1.13.tar.gz
   $ tar xvf kv-ce-25.1.13.tar
   ```

3. Set the $KVHOME variable to the directory path where you extracted the files. For example,

   ```
   export $KVHOME=$HOME/nosql/kv-25.1.13
   ```

Start KVLite

Perform the following steps to start a KVLite instance:

You could start KVLite in secure mode (the default option) or non-secure mode. If you start KVLite in a secure mode and you are using a to access it, configure and start a secure proxy. Similarly, if you start KVLite in non-secure mode, configure and start a non secure proxy.

Start KVLite in secure mode:

You can start KVLite in secure mode using one of the following options:

- Open a terminal and navigate to `$KVHOME`, the directory where you have installed Oracle NoSQL Database distribution.

  ```
  $ cd $KVHOME$
  java -Xmx64m -Xms64m -jar lib/kvstore.jar kvlite [-storagedirsizegb N]
  ```

  The `storagedirsizegb` is an optional parameter. This parameter can be used to control the size of the storage directory for a new store instance. The value of N is in GB and must be >= 1. By default, the created store has a size of 10GB, and this flag can be used to override that. The size cannot be changed after the instance has been created and use of the flag on an existing instance is ignored.

- To start KVLite using the script:

  1. Download and extract the ZIP file (start_securekvlite.zip) into your `$KVHOME` directory. The ZIP file contains the script required to start KVLite in secure mode.
  2. The script (`start_kvlite.sh`) contains the following environment variables with default values to start KVLite. You can run the script as-is or set any of these variables before running it to override the defaults.

Table Environment Variables for KVLite

Name of the variable	Optional	Description	Sample values
HOSTNAME	Yes	Identifies a host name associated with the node on which the script to configure kvlite is run.	localhost
KV_PORT	Yes	The TCP/IP port on which KVLite should be contacted. Sometimes referred to as the registry port.	5000
KV_ADMIN_WEB_PORT	Yes	The TCP/IP port on which the admin service should be started.	-1
KV_STORAGESIZE	Yes	The size of the storage directory in GB.	10

Table (Cont.) Environment Variables for KVLite

Name of the variable	Optional	Description	Sample values
RESTORE_SNAPSHOT	Yes	The full name of the snapshot of the store you want to restore. For example, `260311-053335 -mySnapshot`	null, there will be no restore

> **Note**
>
> On macOS, the default `KV_PORT` value (5000), is not available. Set `KV_PORT` to an unused port before starting KVLite.

3. Invoke the script (`start_kvlite.sh`) to start KVLite in secure mode.

   ```
   $/bin/bash start_kvlite.sh
   ```

   You get an output as shown below:

   ```
   Waiting for kvstore to start...
   Waiting for kvstore to start...
   Generated password for user admin:*********
   User login file: kvroot/security/user.security
   Created new kvlite store with args:
   -root kvroot -store kvstore -host localhost -port 5000 -admin-web-port
   -1 -secure-config enable
   ```

   Where `kvstore` is the name of the store, `localhost` is the name of the local host, and `kvroot` is the directory where Oracle NoSQL Database data is placed. It takes about 10 - 60 seconds before this message is issued, depending on the speed of your system.

- To ensure KVLite is running, see Verify your Installation.

To start Proxy in secure mode

The proxy is required only for SDK-based clients (such as Java, Python, Node.js, C#, Rust, or Go SDKs). It is not required when using the Java Direct Driver.

You can start proxy in secure mode using one of the following two options:

- To manually start a proxy in secure mode, follow the steps in Using the Proxy in a secure data store.
- To start proxy using the script, first configure the proxy, then start it:
  1. Open a new terminal and navigate to `$KVHOME` directory.
  2. Invoke the script (`setup-http-proxy-sec.sh`) to configure the following:
     - Create a user (`proxy_user`) as the proxy needs an identity to connect to the secure data store.

- Create a new password file to store the credentials needed to login as the proxy user.
- Create a login file `proxy.login` for the proxy user.
- Create self-signed certificates that can be used to securely connect to the Oracle NoSQL Database Proxy.

For more information, see .Using the Proxy in a secure data store

> **Note**
>
> If you have started KVLite with a nondefault value for `KV_PORT`, ensure that you set the same port value before configuring the proxy.

```
$/bin/bash setup-http-proxy-sec.sh
```

You get an output as shown below:

```
Creating password
Creating USER proxy_user
Dec 30, 2025 6:48:23 AM org.jline.utils.Log logr
WARNING: Unable to create a system terminal, creating a dumb terminal
(enable debug logging for more information)
sql-> Statement completed successfully
sql-> Creating proxy secfiles
Created
Secret created
Creating certificate
Generating a RSA private key
.....................................................................
.....................................................................
.....................................+++
.....................................+++
writing new private key to 'kvroot/proxy/key.pem'
-----
Certificate was added to keystore
```

3. Open a new terminal and navigate to `$KVHOME` directory.
4. The script, `start_proxy.sh`, contains the following environment variables with default values to start proxy. You can run the script as-is or set any of these variables before running the script to override the defaults.

> **Note**
>
> If you have started KVLite with a non-default value for `KV_PORT`, ensure that you set the same port value before starting the proxy.

Table Environment Variables for Proxy

Name of the variable	Optional	Description	Sample values
HOSTNAME	Yes	Identifies a host name associated with the node on which the script to configure kvlite is run.	localhost
KV_PORT	Yes	The TCP/IP port on which KVLite should be contacted. Sometimes referred to as the registry port.	5000
KV_PROXY_PORT	Yes	The TCP/IP port on which proxy should be contacted.	Non-Secure proxy: Use 80 if root privilege is there, else 8080 Secure proxy: Use 443 if root privilege is there, else 8443

5. Invoke the script `start_proxy.sh`

```
$/bin/bash start_proxy.sh
```

You get an output as shown below:

```
Starting Proxy
Proxy creating SSL channel
Proxy started:
async=false
helperHosts=localhost:5000
httpPort=0
httpsPort=8443
idleReadTimeout=0
kvConsistency=NONE_REQUIRED
kvDurability=COMMIT_NO_SYNC
kvRequestTimeout=-1
monitorStatsEnabled=false
numAcceptThreads=3
numRequestThreads=32
proxyType=KVPROXY
sslCertificate=kvroot/proxy/certificate.pem
sslPrivateKey=kvroot/proxy/key-pkcs8.pem
sslPrivateKeyPass=iTO6aUCnh9XdsgkxFig=
sslProtocols=TLSv1.2,TLSv1.1,TLSv1
storeName=kvstore
storeSecurityFile=kvroot/proxy/proxy.login
verbose=true
proxyVersion=null
kvclientVersion=25.3.21
```

Start KVLite in non-secure mode:

You can start KVLite in non-secure mode using one of the following options:

- Open a terminal and navigate to `$KVHOME`, the directory where you have installed Oracle NoSQL Database distribution.

  ```
  $ cd $KVHOME
  $ java -jar lib/kvstore.jar kvlite [-storagedirsizegb N ] -secure-config
  disable
  ```

  The `storagedirsizegb` is an optional parameter. This parameter can be used to control the size of the storage directory for a new store instance. The value of N is in GB and must be >= 1. By default, the created store has a size of 10 GB, and this flag must be used to override that. The size cannot be changed after the instance has been created and use of the flag on an existing instance is ignored.

- To start KVLite using the script:

  1. Download and extract the ZIP file (start_nonsecurekvlite.zip) into your $KVHOME directory. The ZIP file contains the script required to start KVLite in non-secure mode. The script, `start_nonsecure_kvlite.sh`, contains the environment variables with default values to start KVLite. You can run the script as-is or set any of these variables before running it to override the defaults. For details on the environment variables, see the table - Environment Variables for KVLite.
  2. Invoke the script (`start_nonsecure_kvlite.sh`) to start KVLite in non-secure mode.

     ```
     $/bin/bash start_nonsecure_kvlite.sh
     ```

     You get an output as shown below:

     ```
     Created new kvlite store with args: -root kvroot -store kvstore -host
     localhost
     -port 5000 -admin-web-port 5999 - secure-config disable
     ```

- To ensure KVLite is running, see Verify your Installation.

To start Proxy in a non-secure mode

The proxy is required only for SDK-based clients (such as Java, Python, Node.js, C#, Rust, or Go SDKs). It is not required when using the Java Direct Driver.

You can start proxy in non-secure mode using one of the two following options:

- To manually start a proxy in non-secure mode, follow the steps from Using the Proxy in a non-secure data store.
- Use the script (`start_nonsecure_proxy.sh`) to start the proxy for a non-secure data store. It contains the environment variables with default values to start a non-secure proxy. You can run the script as-is or set any of the variables before running the script to override the defaults. For details on the environment variables, see the table - Environment Variables for Proxy.

  ```
  $/bin/bash start_nonsecure_proxy.sh
  ```

  You get an output as shown below:

  ```
  Starting Proxy
  Proxy started:
  async=false
  ```

```
helperHosts=localhost:5000
httpPort=8080
httpsPort=0
idleReadTimeout=0
kvConsistency=NONE_REQUIRED
kvDurability=COMMIT_NO_SYNC
kvRequestTimeout=-1
monitorStatsEnabled=false
numAcceptThreads=3
numRequestThreads=32
proxyType=KVPROXY
sslProtocols=TLSv1.2,TLSv1.1,TLSv1
storeName=kvstore
verbose=true
proxyVersion=null
kvclientVersion=25.1.13
```

For a complete list of command line options that can be provided with the kvlite utility, see kvlite.

To connect KVLite with the SQL Shell, see Running the SQL Shell.

Verify your Installation

You can verify your installation and ensure that KVLite is running.

- Start a new shell and run the following command:

  ```
  $ jps -m
  ```

  Your list of processes running will include the kvlite(kvstore.jar) and the proxy that you configured and started. You get an output similar to the one shown below.

  ```
  3523439 Jps -m
  3500313 httpproxy.jar -helperHosts localhost:5000 -storeName kvstore -
  httpPort 8080 -verbose true
  3499946 kvstore.jar kvlite -secure-config disable -root kvroot -host
  localhost -port 5000 -admin-web-port 5999 -harange 5010,5020 -servicerange
  5021,5049 -storagedirsizegb 10
  ```

  You can also ping your KVLite instance to see if the KVLite is configured and started successfully.

  For secure KVLite:

  ```
  java -Xmx64m -Xms64m -jar lib/kvstore.jar ping -host localhost \
  -port 5000 -security kvroot/security/user.security
  ```

  For non-secure KVLlite:

  ```
  java -Xmx64m -Xms64m -jar lib/kvstore.jar ping -host localhost -port 5000
  ```

  You get an output similar to the one shown below:

  ```
  Pinging components of store kvstore based upon topology sequence #14
  10 partitions and 1 storage nodes
  Time: 2025-04-01 05:23:45 UTC   Version: 25.1.13
  Shard Status: healthy: 1 writable-degraded: 0 read-only: 0 offline: 0
  total: 1
  Admin Status: healthy
  Zone [name=KVLite id=zn1 type=PRIMARY allowArbiters=false
  masterAffinity=false]
  RN Status: online: 1 read-only: 0 offline: 0
  Storage Node [sn1] on localhost: 5000
  Zone: [name=KVLite id=zn1 type=PRIMARY allowArbiters=false
  masterAffinity=false]
  Status: RUNNING   Ver: 25.1.13 2025-04-01 05:24:19 UTC
  Build id: e0c93c1f1395 Edition: Enterprise    isMasterBalanced: true
  serviceStartTime: 2025-04-01 05:24:21 UTC
  Admin [admin1] Status: RUNNING,MASTER  serviceStartTime: 2025-04-01
  05:24:23 UTC
  stateChangeTime: 2025-04-01 05:25:04 UTC   availableStorageSize: 2 GB
  Rep Node [rg1-rn1] Status: RUNNING,MASTER sequenceNumber: 85 haPort: 5011
  availableStorageSize: 9 GB storageType: HD serviceStartTime: 2025-04-01
  ```

```
05:24:05 UTC
stateChangeTime: 2025-04-01 05:24:07 UTC
```

Stop and Restart KVLite

To stop and restart KVLite, perform the following steps:

To stop KVLite, use `Ctrl C (^C)` from within the shell where KVLite is running.

To restart the process, run the KVLite utility without any command line options. Do this even if you provided non-standard options when you first started KVLite. This is because KVLite remembers information such as the port value and the store name in between run times. You cannot change these values by using the command line options.

```
$ java -Xmx64m -Xms64m -jar KVHOME/lib/kvstore.jar kvlite
```

If you want to start over with different options than you initially specified, delete the `KVROOT` directory (`./kvroot`), and then rerun the KVLite utility with the options you need. Refer to Start KVLite.

Note: If you decide to start over, all your previous data will be lost.

Quick Start to KVLite in a Container

You can run Oracle NoSQL Database in a container using the Oracle NoSQL Database container image.

Prerequisite

Install a containerization tool. In this book, we have made use of Docker as an example.

> **Note**
>
> A full description of Docker is beyond the scope of this documentation. This chapter assumes prior knowledge of Docker.

The container image that we will be using is a simplified version of the Oracle NoSQL Database called KVLite. KVLite provides a single storage node, single shard store, that is not replicated. It runs in a single process without requiring any administrative interface.

> **Note**
>
> - KVLite is not intended for production deployment or performance measurements. We recommend testing with data that is not considered sensitive in nature. In other words, do not test with sensitive information such as usernames, passwords, credit card information, medication information, etc.
> - There are two Oracle NoSQL container images available in the GitHub container registry, one using a secure configuration and the other using a non-secure configuration. The primary difference is in the way KVLite is accessed. We recommend using the secure setup, although additional steps are needed during setup. One advantage of using the secure setup is that it gives you exposure to what is needed to setup a secure KVStore.

We will cover the following topics in this chapter:

- Start KVLite in a Container
- Connect to Database Using SDK Driver
- Connect to Database Using Oracle NoSQL Visual Studio Code Extension
- Connect to Database Using Oracle NoSQL IntelliJ Plugin

For details on more advanced scenarios, see the documentation in Github for connecting to non-secure and secure Oracle NoSQL Database in a container.

Start KVLite in a Container

This section explains how to start KVLite in a container.

1. Pull the Oracle NoSQL Database Container Image

The steps outlined below use the Oracle NoSQL Database Community Edition. You can pull the image directly from the GitHub Container Registry. Here, we are pulling the non-secure image.

```
docker pull ghcr.io/oracle/nosql:latest-ce
docker tag ghcr.io/oracle/nosql:latest-ce oracle/nosql:ce
```

2. Run Oracle NoSQL Database in a Container

   You must provide a name and a hostname.

   ```
   docker run -d --name=kvlite --hostname=kvlite --env KV_PROXY_PORT=8080 -p
   8080:8080 oracle/nosql:ce
   ```

   Output:

   KVLite is started in a container and the ID of the container is displayed.

   ```
   19001d44b56aa9a53c75cf0904298c4e0e3013df287e4a8f83ebff2e1b0ac172
   ```

   By default, the KVLite store that is created has a size of 10 GB. You can use `--env KV_STORAGESIZE=N` to set a new value, where N is in gigabytes and must have a value greater than 1.

3. Check the Status of the Container

   To check the status of the container, run the following command:

   ```
   docker ps
   ```

   Output:

   The details of the container are displayed.

   ```
   CONTAINER ID   IMAGE              COMMAND
   CREATED           STATUS            PORTS
   NAMES
   19001d44b56a   oracle/nosql:ce    "bash -c ./start-kvl…"  6 minutes ago
   Up 6 minutes 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp  kvlite
   ```

4. Validate your Deployment

   You can ping the KVLite store instance using the following command

   ```
   docker exec -ti kvlite java -jar lib/kvstore.jar ping -host kvlite -port
   5000
   ```

   Output:

   ```
   Pinging components of store kvstore based upon topology sequence #14
   10 partitions and 1 storage nodes
   Time: 2025-04-28 09:57:53 UTC   Version: 24.4.9
   Shard Status: healthy: 1 writable-degraded: 0 read-only: 0 offline: 0
   total: 1
   Admin Status: healthy
   Zone [name=KVLite id=zn1 type=PRIMARY allowArbiters=false
   ```

```
masterAffinity=false]   RN Status: online: 1 read-only: 0 offline: 0
Storage Node [sn1] on kvlite: 5000    Zone: [name=KVLite id=zn1
type=PRIMARY allowArbiters=false masterAffinity=false]    Status:
RUNNING   Ver: 24.4.9 2024-11-21 17:06:06 UTC  Build id: 95fa28ea4441
Edition: Community    isMasterBalanced: true      serviceStartTime:
2025-04-28 09:43:36 UTC
        Admin [admin1]          Status: RUNNING,MASTER  serviceStartTime:
2025-04-28 09:43:38 UTC       stateChangeTime: 2025-04-28 09:43:38
UTC        availableStorageSize: 2 GB
        Rep Node [rg1-rn1]      Status: RUNNING,MASTER sequenceNumber: 84
haPort: 5011 availableStorageSize: 9 GB storageType: HD
serviceStartTime: 2025-04-28 09:43:39 UTC stateChangeTime: 2025-04-28
09:43:39 UTC
```

5. Query the Database

   You can start the SQL shell to query the database and create few tables.

   ```
   docker exec -ti kvlite  java -jar lib/sql.jar -helper-hosts kvlite:5000 -
   store kvstore
   ```

   The SQL prompt is displayed.

   ```
   sql->
   ```

   Enter the SQL query to create a table named `ticket`.

   ```
   sql->create table if not exists ticket(ticketNo LONG, confNo STRING,
   PRIMARY KEY(ticketNo))
   ```

   Output:

   The table is created successfully

   ```
   Statement completed successfully
   ```

   Enter the query to show the list of tables

   ```
   sql->show tables
   ```

   Output:

   The table `ticket` that you just created is listed.

   ```
   tables
     SYS$IndexStatsLease
     SYS$MRTableAgentStat
     SYS$MRTableInfo
     SYS$MRTableInitCheckpoint
     SYS$PartitionStatsLease
     SYS$SGAttributesTable
     SYS$StreamRequest
     SYS$StreamResponse
     SYS$TableMetadata
   ```

```
SYS$TableStatsIndex
SYS$TableStatsPartition
SYS$TopologyHistory
ticket
```

Connect to Database Using SDK Driver

This section describes how you can connect to KVLite in a container using SDK driver.

The Oracle NoSQL Database SDK drivers can be used to access either the Oracle NoSQL Database cloud service, cloud simulation, or an on-premise installation via the Oracle NoSQL Database Proxy. For more details, see About Oracle NoSQL Database SDK drivers.

The Oracle NoSQL Database drivers are available for various programming languages. Here is a snippet from a Python program showing the connection to the KVLite instance that we are running inside a container.

```
def get_connection_onprem():

   kvstore_endpoint ='http://localhost:8080'
   provider = StoreAccessTokenProvider()
   return NoSQLHandle(NoSQLHandleConfig(kvstore_endpoint, provider))
```

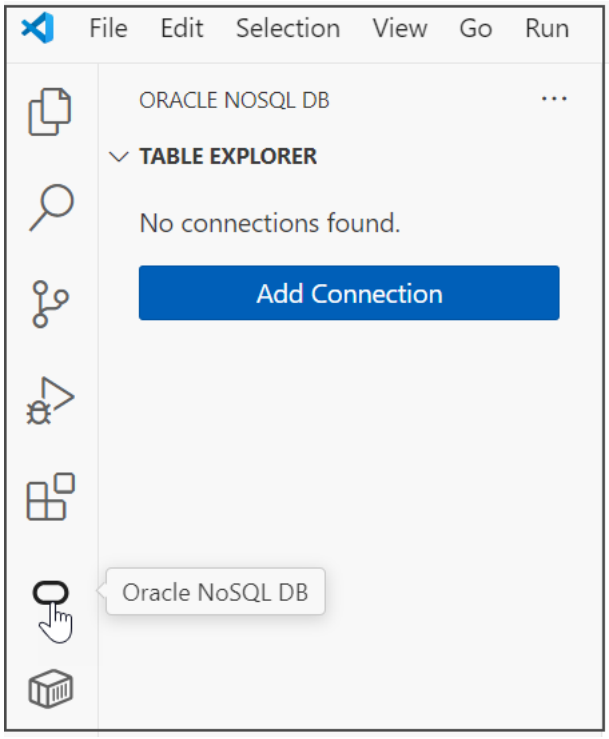
If you are using an external host, provide the name of the host machine instead of `localhost`.

Connect to Database Using Oracle NoSQL Visual Studio Code Extension

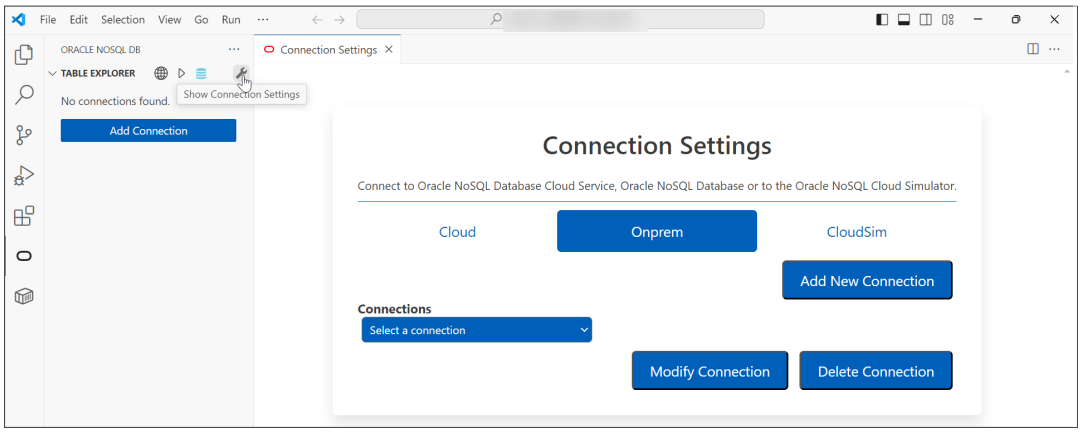
This section describes how you can connect to KVLite in a container using Oracle Database Visual Studio Code Extension.

Oracle NoSQL Database Visual Studio (VS) Code extension provides multiple ways to connect to an Oracle NoSQL Database. You can connect to Oracle NoSQL Database cloud service, cloud simulation, or an on-premise installation. For more details, see About Oracle NoSQL Database Visual Studio Code Extension. In the below case, we will connect to the KVLite instance that we are running inside a container by creating an on-premise connection.

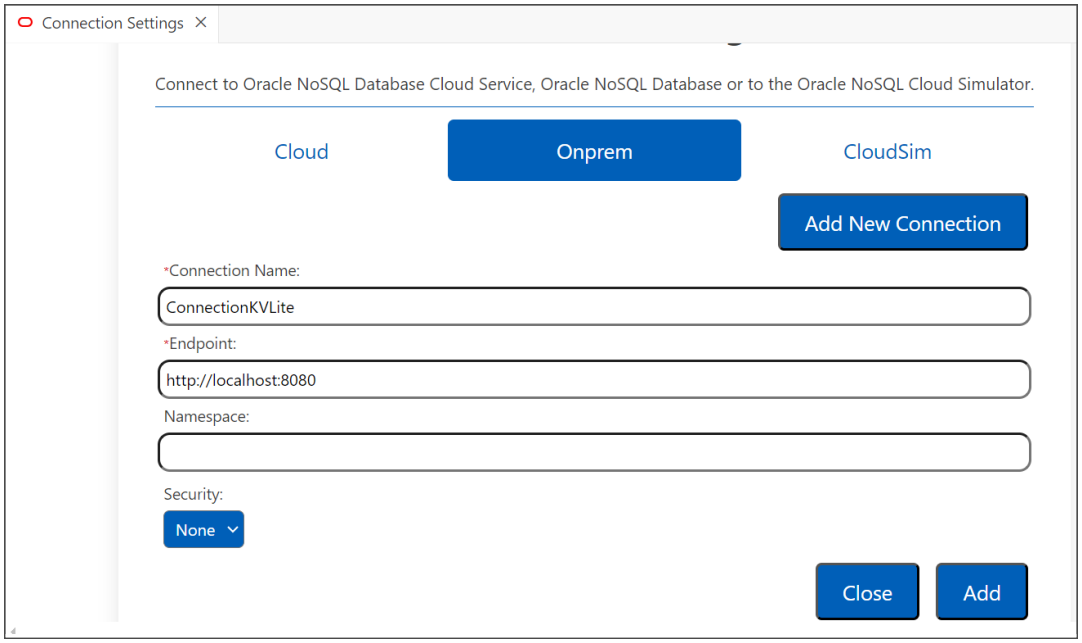
1. Open the **Oracle NoSQL DB** extension from the left panel in Visual Studio Code.



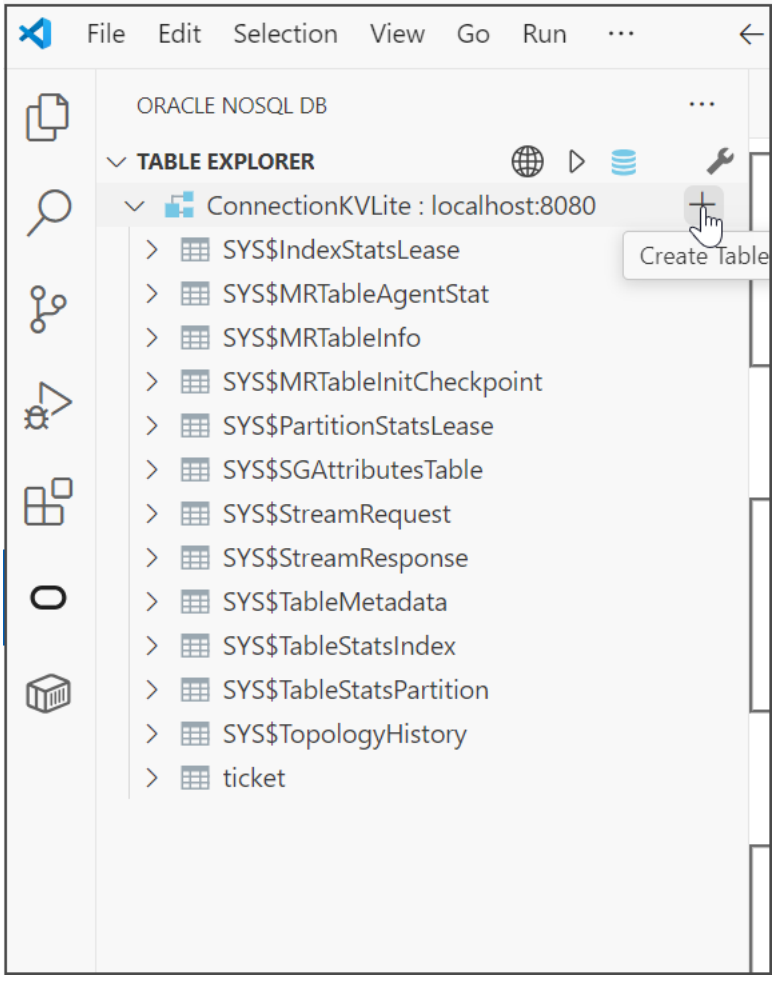
2. Choose **Show Connection Settings** page from the Command Palette.



3. Choose **Onprem**. Choose **Add New Connection**. Provide a **Connection Name** and provide the **Endpoint** as `http://localhost:8080` (if you are using an external host - provide the name of the host machine). Choose **Add**.



4. You will now be able to view the connection in the left panel. Choose the Plus icon or right-click on the database connection name and choose **Create Table**.



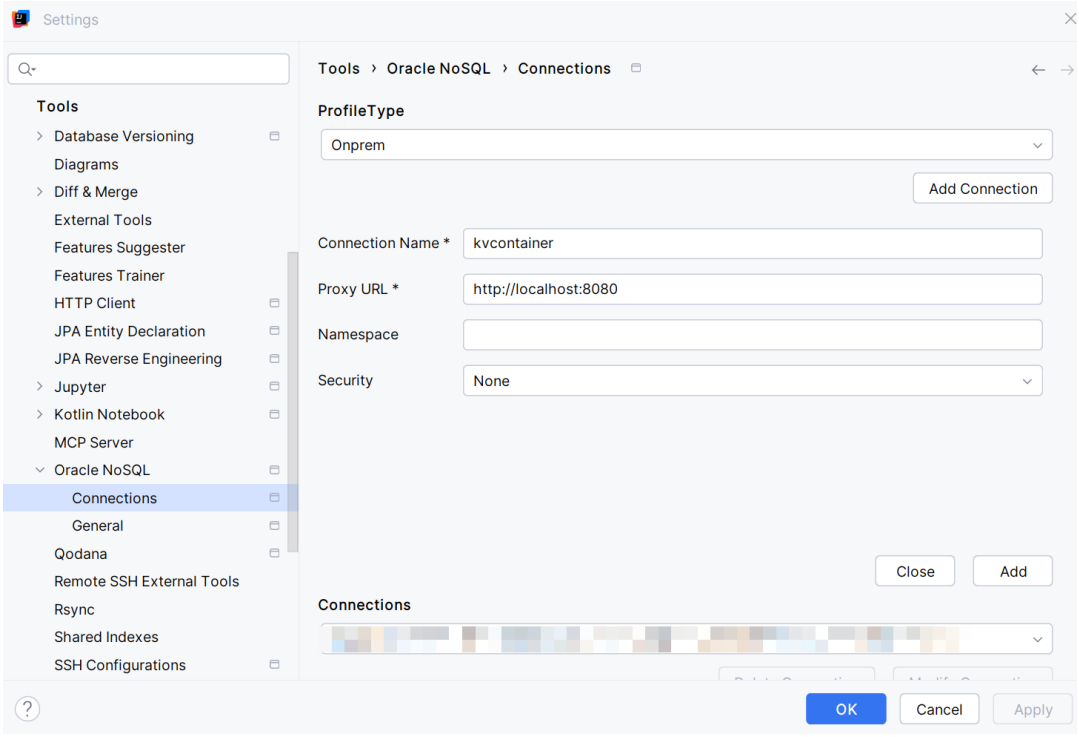
. For more details on creating and managing tables, see Managing Tables Using Visual Studio Code Extension

Connect to Database Using Oracle NoSQL IntelliJ Plugin

This section describes how you can connect to KVLite in container using Oracle NoSQL IntelliJ plugin.

Oracle NoSQL Database IntelliJ Plugin provides multiple ways to connect to an Oracle NoSQL Database. You can connect to Oracle NoSQL Database cloud service, cloud simulation, or an on-premise installation. For more details, see About IntelliJ Plugin. In the below case, we will connect to the KVLite instance that we are running inside a container by creating an on-premise connection.

1. Choose the task icon in the **Schema Explorer** window to open the **Settings** dialog.
2. Expand **Tools> Oracle NoSQL** in the **Settings** dialog, and choose **Connections**.
3. Select **Onprem** from the drop-down menu.
4. Choose **Add Connection**. Enter the connection parameters and select **OK**. Enter the proxy URL as `http://localhost:8080` (if you are using an external host - provide the name of the host machine).



5. Go to **Schema Explorer** and view the connection. You can right-click on the connection and choose **Create Table** and perform various table operations. For more details on this, see Managing Tables Using the IntelliJ Plugin.