轻松上手：使用 Docker Compose 部署 TiDB 的简易指南

首页推荐

原创 shunwah M 2025-04-27 189

> 作者：ShunWah
>
> 在运维管理领域，我拥有多年深厚的专业积累，兼具坚实的理论基础与广泛的实践经验。精通运维自动化流程，对于OceanBase、MySQL等多种数据库的部署与运维，具备从初始部署到后期维护的全链条管理能力。拥有OceanBase的OBCA和OBCP认证、OpenGauss社区认证结业证书，以及崖山DBCA、亚信AntDBCA、翰高HDCA、GBase 8a | 8c | 8s、Galaxybase的GBCA、Neo4j的Graph Data Science Certification、NebulaGraph的NGCI & NGCP、东方通TongTech TCPE等多项权威认证。
> 在OceanBase & 墨天轮的技术征文大赛中，多次荣获一、二、三等奖。同时，在OpenGauss第五届、第六届、第七届技术征文大赛，TiDB社区专栏征文大赛，金仓数据库有奖征文活动，以及YashanDB「产品体验官」征文等活动中，我也屡获殊荣。此外，我还活跃于墨天轮、CSDN、ITPUB等技术平台，经常发布原创技术文章，并多次被首页推荐。

引言

TiDB 是一个开源的分布式关系型数据库，兼容 MySQL 协议。它结合了传统关系型数据库的强一致性和 NoSQL 数据库的可扩展性。本文将详细介绍如何在 CentOS 7 上使用 Docker Compose 部署一个完整的 TiDB 集群，包括 PD (Placement Driver)、TiKV (Storage Engine) 和 TiDB (SQL Layer)，并进行基本的数据库操作和性能优化。

一、部署 TiDB

准备工作
首先确保你的 CentOS 7 系统已经安装了 Docker 和 Docker Compose。如果没有，请先安装它们：

1. 系统环境准备

确保你的开发环境满足 TiDB 的运行要求。TiDB 支持多种操作系统，包括Linux、macOS和Windows。本文以 x86 架构的 CentOS Linux 7.9 镜像作为环境介绍如何使用 docker容器化部署 TiDB v7.5 数据库。

shunwah M LV.5 关注

133 文章 99 粉丝 181K+ 浏览量

获得了 1266 次点赞
内容获得 407 次评论
获得了 166 次收藏

热门文章

部署反向代理神器 Nginx Proxy Manager 配置阿里云ssl证书
2022-10-27 15542浏览

第一次如何通过OceanBase初级OBCA、中级OBCP认证考试
2022-06-24 10388浏览

Windows工具DBeaver连接OceanBase数据库访问MySQL和Oracle租户
2022-03-09 7502浏览

「更易用的OceanBase」Docker 部署 OceanBase 4.0 数据库 快速体验增删改查
2022-11-25 4257浏览

OceanBase 单节点手动部署OB文档
2022-02-13 4021浏览

最新文章

【金仓数据库产品体验官】从零实测：金仓数据库MySQL兼容深度探秘
1天前 37浏览

探索 OceanBase 租户克隆：一分钟搞定租户复制
2025-07-28 84浏览

金仓数据库KingbaseES 兼容SQL Server高级特性评测
2025-07-21 80浏览

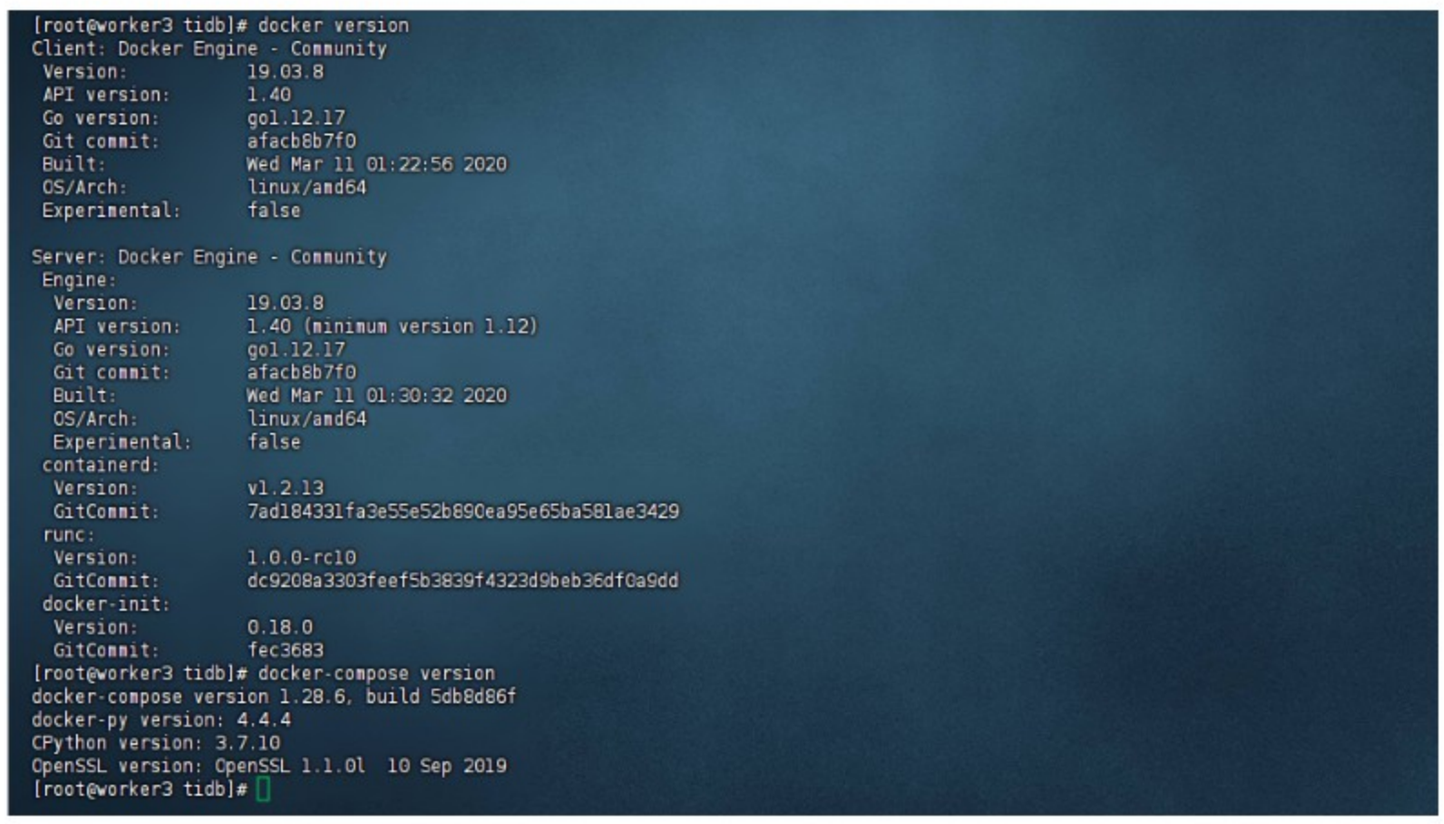
MySQL 8.0.40：字符集革命、窗口函数效能与DDL原子性实践
2025-07-15 141浏览

【金仓数据库产品体验官】金仓数据库（SQL Server兼容版）部署与核心特性...
2025-07-02 167浏览

```
[root@worker3 tidb]# cat /etc/redhat-release
CentOS Linux release 7.9.2009 (Core)
[root@worker3 tidb]# free -h
              total        used        free      shared  buff/cache   available
Mem:            27G        1.9G         19G        672M        5.8G         24G
Swap:          8.0G          0B        8.0G
[root@worker3 tidb]#
```

2. docker 容器环境准备

```
[root@worker3 tidb]# docker version
Client: Docker Engine - Community
 Version:           19.03.8
 API version:       1.40
 Go version:        go1.12.17
 Git commit:        afacb8b7f0
 Built:             Wed Mar 11 01:22:56 2020
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.8
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.17
  Git commit:       afacb8b7f0
  Built:            Wed Mar 11 01:30:32 2020
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          v1.2.13
  GitCommit:        7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc:
  Version:          1.0.0-rc10
  GitCommit:        dc9208a3303feef5b3839f4323d9beb36df0a9dd
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
[root@worker3 tidb]# docker-compose version
docker-compose version 1.28.6, build 5db8d86f
docker-py version: 4.4.4
CPython version: 3.7.10
OpenSSL version: OpenSSL 1.1.0l  10 Sep 2019
[root@worker3 tidb]#
```

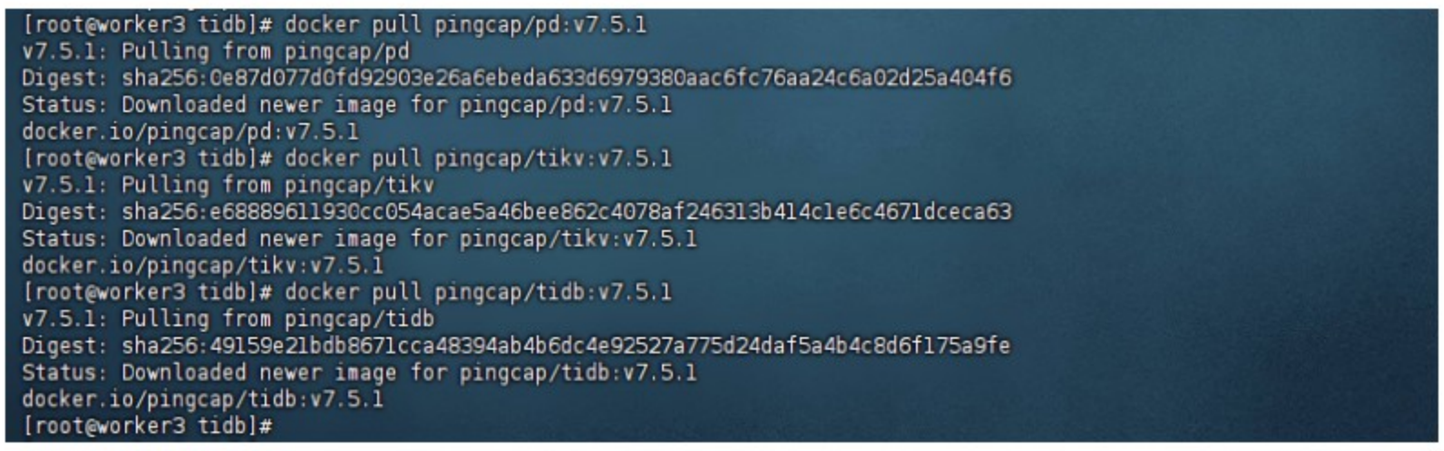


目录

3、Docker 部署 TiDB 拉取镜像：

通过 Docker 部署 TiDB 适合测试或开发环境，可快速启动基础集群。步骤如下：

```
[root@worker3 tidb]# docker pull pingcap/pd:v7.5.1
v7.5.1: Pulling from pingcap/pd
Digest: sha256:0e87d077d0fd92903e26a6ebeda633d6979380aac6fc76aa24c6a02d25a404f6
Status: Downloaded newer image for pingcap/pd:v7.5.1
docker.io/pingcap/pd:v7.5.1
[root@worker3 tidb]# docker pull pingcap/tikv:v7.5.1
v7.5.1: Pulling from pingcap/tikv
Digest: sha256:e68889611930cc054acae5a46bee862c4078af246313b414c1e6c4671dceca63
Status: Downloaded newer image for pingcap/tikv:v7.5.1
docker.io/pingcap/tikv:v7.5.1
[root@worker3 tidb]# docker pull pingcap/tidb:v7.5.1
v7.5.1: Pulling from pingcap/tidb
Digest: sha256:49159e21bdb8671cca48394ab4b6dc4e92527a775d24daf5a4b4c8d6f175a9fe
Status: Downloaded newer image for pingcap/tidb:v7.5.1
docker.io/pingcap/tidb:v7.5.1
[root@worker3 tidb]#
```

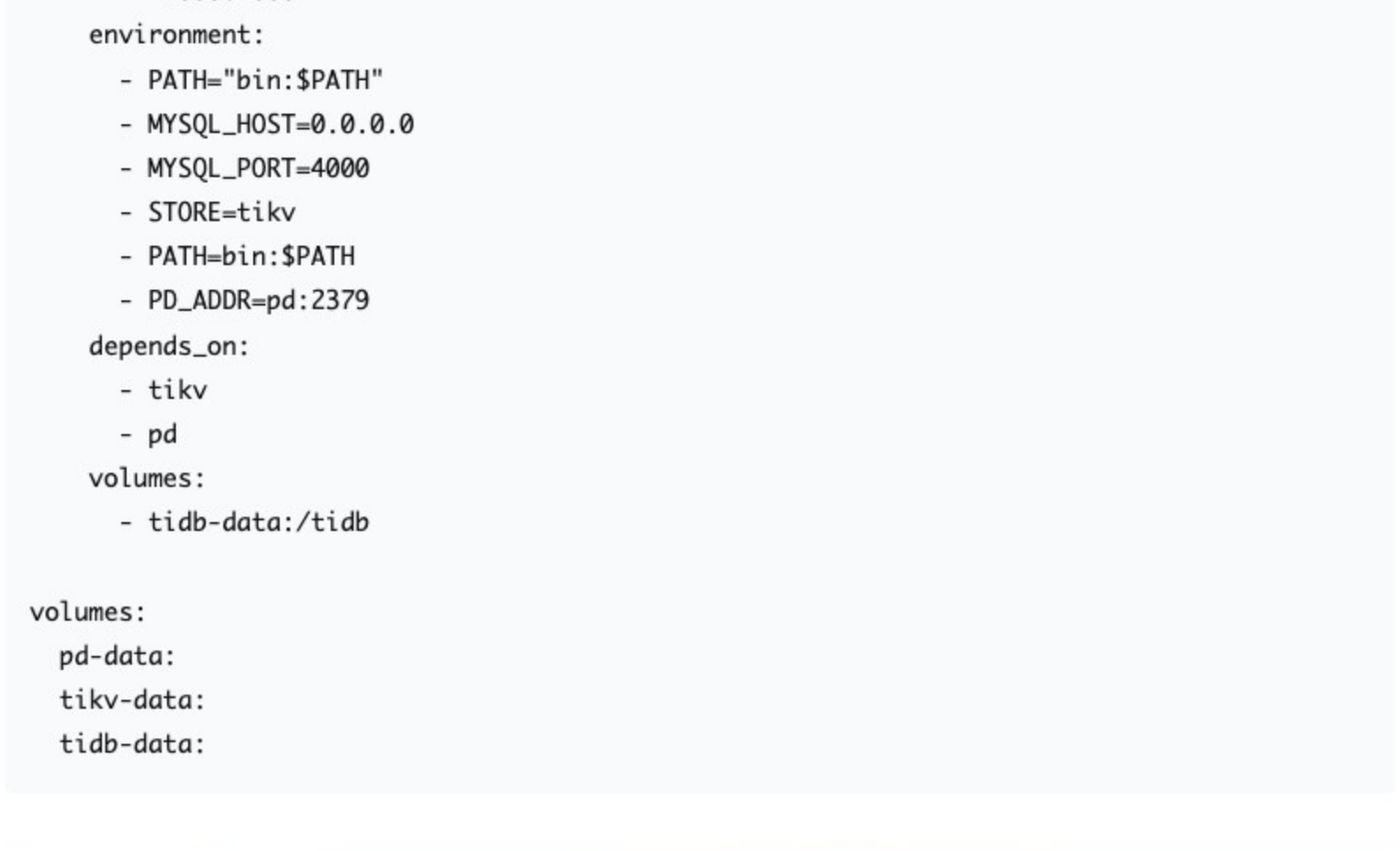


4、创建 docker-compose.yml 文件：

Docker Compose 更适合需要简化配置、快速部署和管理多个容器的场景。接下来，我们将为 TiDB 的各个组件（PD、TiKV、TiDB）编写 docker-compose.yml 文件。步骤如下：

```
[root@worker3 tidb]# vim docker-compose.yml

version: '3'
services:
  pd:
    image: pingcap/pd:latest
    container_name: tidb_pd
    cap_add:
      - SYS_ADMIN
    security_opt:
      - seccomp:unconfined
    privileged: true
    ports:
      - "2379:2379"
      - "2380:2380"
    environment:
      - PD_SERVER_NAME=pd
      - INITIAL_CLUSTER=pd=http://pd:2380
      - ETCD_INITIAL_CLUSTER_STATE=new
    volumes:
      - pd-data:/pd

  tikv:
    image: pingcap/tikv:latest
    container_name: tidb_tikv
    cap_add:
      - SYS_ADMIN
    security_opt:
      - seccomp:unconfined
    privileged: true
    ports:
      - "20160:20160"
    environment:
      - PD_ADDR=pd:2379
```

```
    depends_on:
      - pd
    volumes:
      - tikv-data:/tikv

  tidb:
    image: pingcap/tidb:latest
    container_name: tidb_server
    cap_add:
      - SYS_ADMIN
    security_opt:
      - seccomp:unconfined
    privileged: true
    ports:
      - "4000:4000"
    environment:
      - PATH="bin:$PATH"
      - MYSQL_HOST=0.0.0.0
      - MYSQL_PORT=4000
      - STORE=tikv
      - PATH=bin:$PATH
      - PD_ADDR=pd:2379
    depends_on:
      - tikv
      - pd
    volumes:
      - tidb-data:/tidb

volumes:
  pd-data:
  tikv-data:
  tidb-data:
```



5、启动 TiDB 集群

保存上述配置后，在相同目录下运行以下命令启动 TiDB 集群：

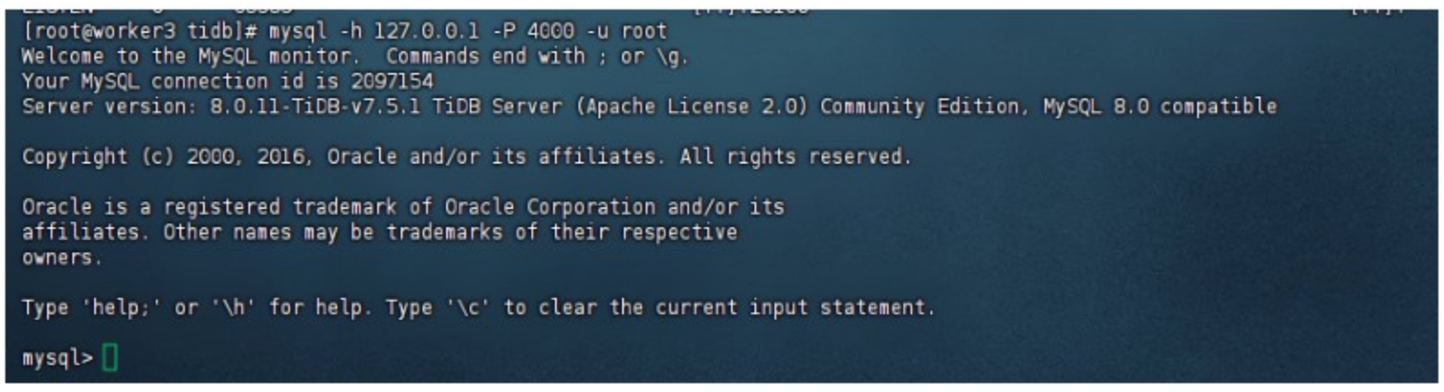
```
[root@worker3 tidb]# docker-compose up -d
Starting tidb_pd ... done
Starting tidb_tikv ... done
Starting tidb_server ... done
[root@worker3 tidb]#
```

二、数据库使用优化操作

1、创建数据库和表

连接到 TiDB 实例并执行 SQL 命令：

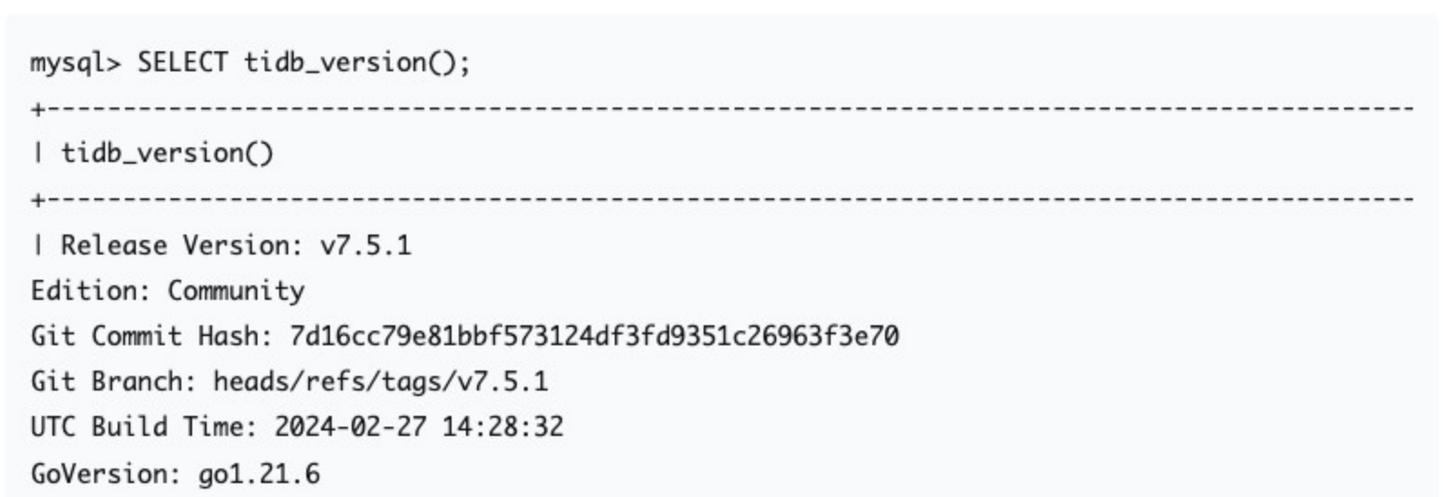
```
[root@worker3 tidb]# mysql -h 127.0.0.1 -P 4000 -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2097154
Server version: 8.0.11-TiDB-v7.5.1 TiDB Server (Apache License 2.0) Community Edition, MySQ

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```



2、查看数据库版本

在 MySQL 客户端中：查看版本 Version: v7.5.1

```
mysql> SELECT tidb_version();
+--------------------------------------------------------------------------------------------------
| tidb_version()
+--------------------------------------------------------------------------------------------------
| Release Version: v7.5.1
Edition: Community
Git Commit Hash: 7d16cc79e81bbf573124df3fd9351c26963f3e70
Git Branch: heads/refs/tags/v7.5.1
UTC Build Time: 2024-02-27 14:28:32
GoVersion: go1.21.6
Race Enabled: false
Check Table Before Drop: false
Store: unistore |
+--------------------------------------------------------------------------------------------------
1 row in set (0.00 sec)

mysql>
```



3、创建数据库

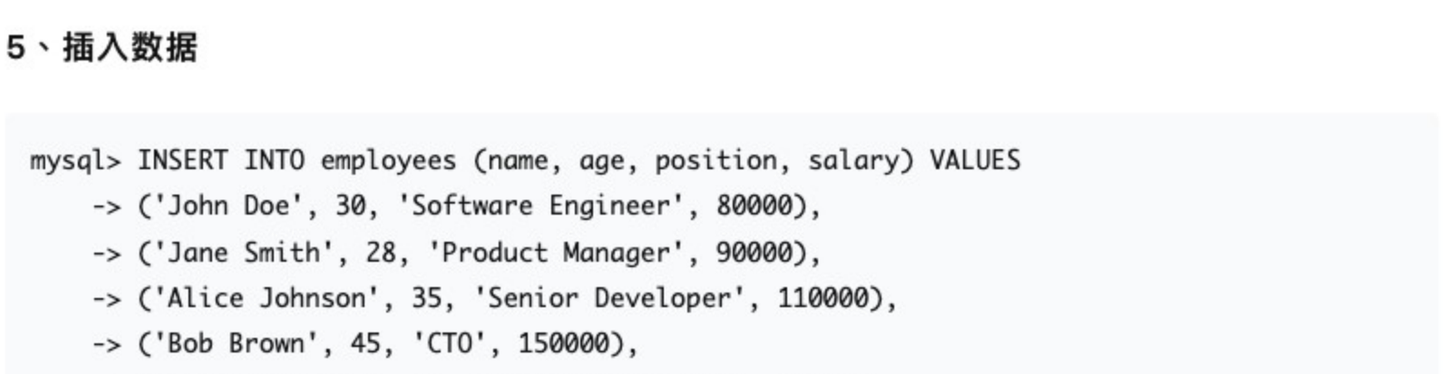
```
mysql> CREATE DATABASE testdb;
Query OK, 0 rows affected (0.02 sec)

mysql> USE testdb;
Database changed
mysql>
```

4、创建表

```
mysql> CREATE TABLE employees (
    ->     id INT PRIMARY KEY AUTO_INCREMENT,
    ->     name VARCHAR(255) NOT NULL,
    ->     age INT,
    ->     position VARCHAR(255),
    ->     salary DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
```

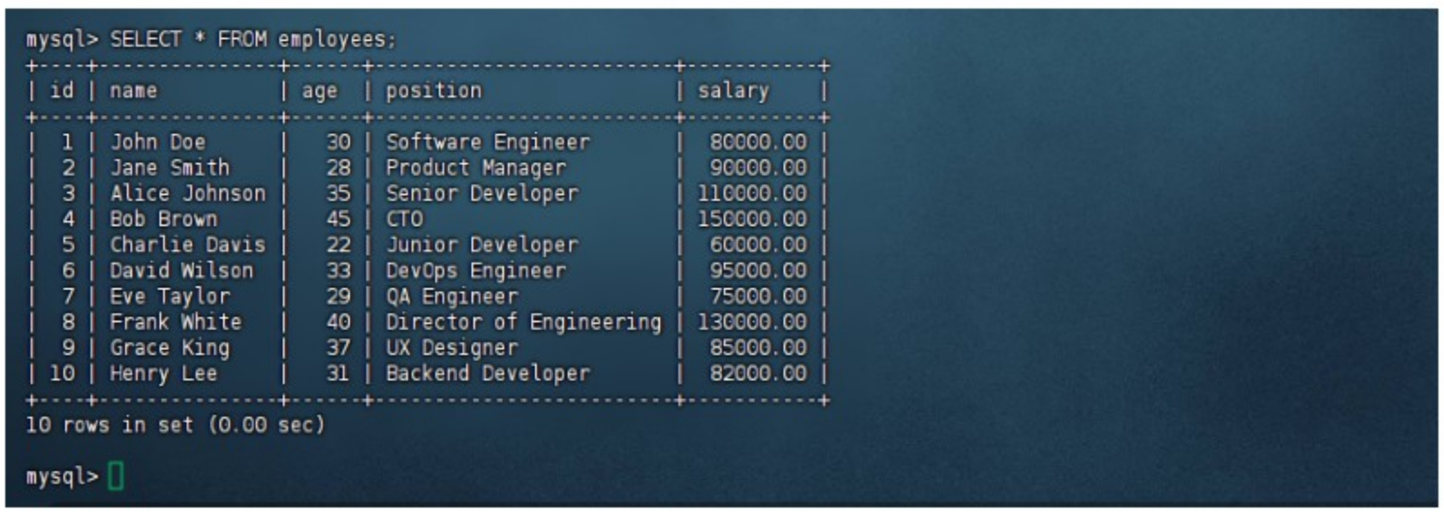
5、插入数据

```
mysql> INSERT INTO employees (name, age, position, salary) VALUES
    -> ('John Doe', 30, 'Software Engineer', 80000),
    -> ('Jane Smith', 28, 'Product Manager', 90000),
    -> ('Alice Johnson', 35, 'Senior Developer', 110000),
    -> ('Bob Brown', 45, 'CTO', 150000),
    -> ('Charlie Davis', 22, 'Junior Developer', 60000),
    -> ('David Wilson', 33, 'DevOps Engineer', 95000),
    -> ('Eve Taylor', 29, 'QA Engineer', 75000),
    -> ('Frank White', 40, 'Director of Engineering', 130000),
    -> ('Grace King', 37, 'UX Designer', 85000),
    -> ('Henry Lee', 31, 'Backend Developer', 82000);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
```



6、查询所有员工信息

```
mysql> SELECT * FROM employees;
+----+---------------+------+------------------------+-----------+
| id | name          | age  | position               | salary    |
+----+---------------+------+------------------------+-----------+
|  1 | John Doe      |   30 | Software Engineer      |  80000.00 |
|  2 | Jane Smith    |   28 | Product Manager        |  90000.00 |
|  3 | Alice Johnson |   35 | Senior Developer       | 110000.00 |
|  4 | Bob Brown     |   45 | CTO                    | 150000.00 |
|  5 | Charlie Davis |   22 | Junior Developer       |  60000.00 |
|  6 | David Wilson  |   33 | DevOps Engineer        |  95000.00 |
|  7 | Eve Taylor    |   29 | QA Engineer            |  75000.00 |
|  8 | Frank White   |   40 | Director of Engineering | 130000.00 |
|  9 | Grace King    |   37 | UX Designer            |  85000.00 |
| 10 | Henry Lee     |   31 | Backend Developer      |  82000.00 |
+----+---------------+------+------------------------+-----------+
10 rows in set (0.00 sec)

mysql>
```

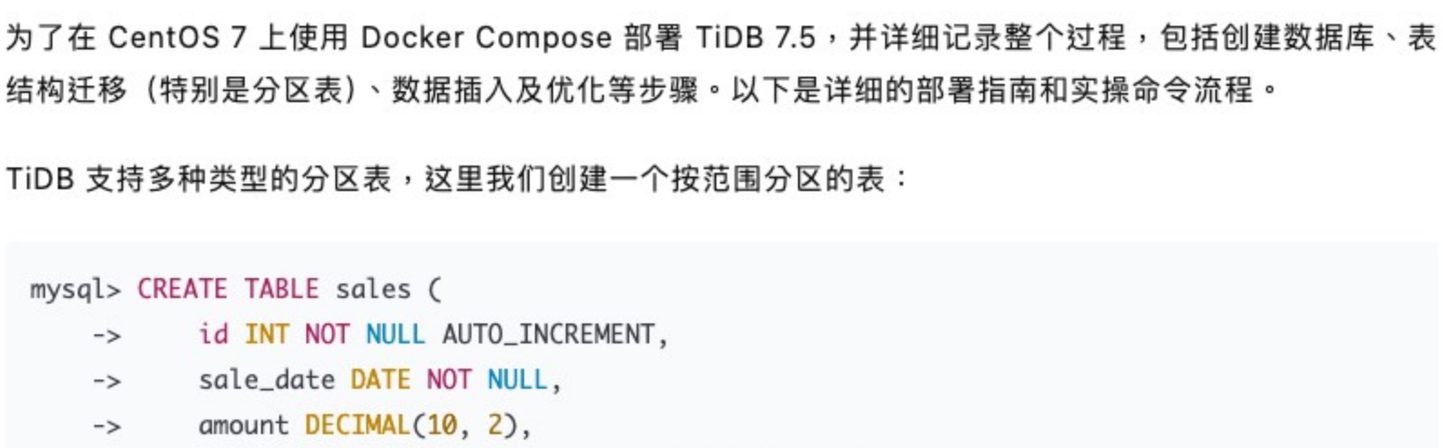


7、创建分区表

为了在 CentOS 7 上使用 Docker Compose 部署 TiDB 7.5，并详细记录整个过程，包括创建数据库、表结构迁移（特别是分区表）、数据插入及优化等步骤。以下是详细的部署指南和实操命令流程。

TiDB 支持多种类型的分区表，这里我们创建一个按范围分区的表：

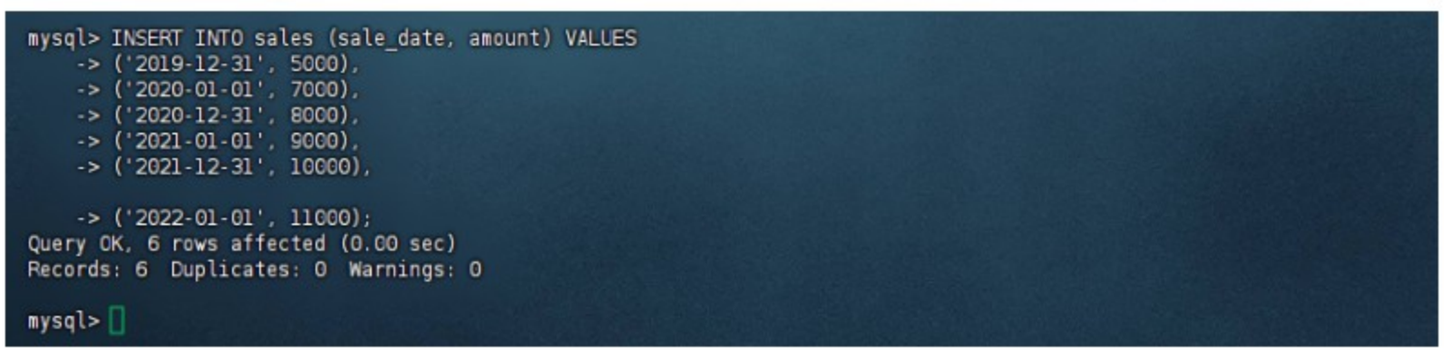
```
mysql> CREATE TABLE sales (
    ->     id INT NOT NULL AUTO_INCREMENT,
    ->     sale_date DATE NOT NULL,
    ->     amount DECIMAL(10, 2),
    ->     PRIMARY KEY (id, sale_date) -- 主键包含分区列
    -> ) PARTITION BY RANGE (YEAR(sale_date)) (
    ->     PARTITION p0 VALUES LESS THAN (2020),
    ->     PARTITION p1 VALUES LESS THAN (2021),
    ->     PARTITION p2 VALUES LESS THAN (2022),
    ->     PARTITION p3 VALUES LESS THAN MAXVALUE
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
```



让主键包含分区函数中使用的列（即 sale_date），从而满足 TiDB 的约束条件。

8、 插入数据

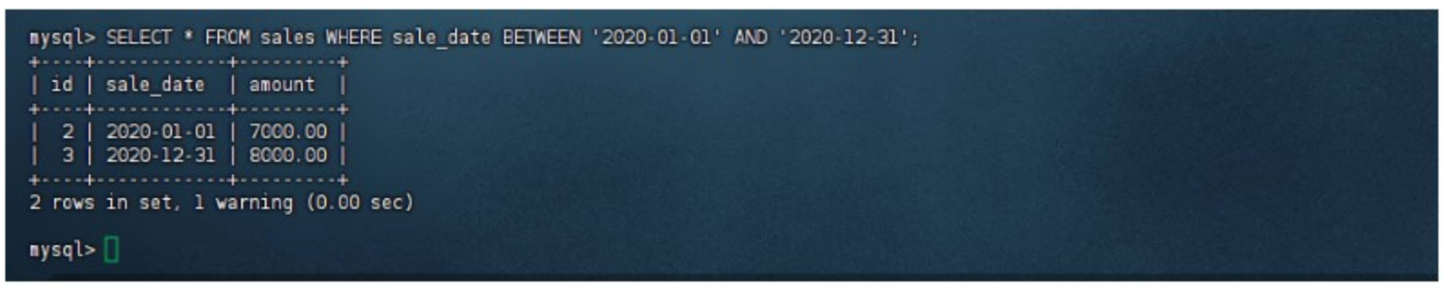
```
mysql> INSERT INTO sales (sale_date, amount) VALUES
    -> ('2019-12-31', 5000),
    -> ('2020-01-01', 7000),
    -> ('2020-12-31', 8000),
    -> ('2021-01-01', 9000),
    -> ('2021-12-31', 10000),

    -> ('2022-01-01', 11000);
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql>
```



9、查询 2020 年的数据

SELECT * FROM sales WHERE sale_date BETWEEN ‘2020-01-01’ AND ‘2020-12-31’;

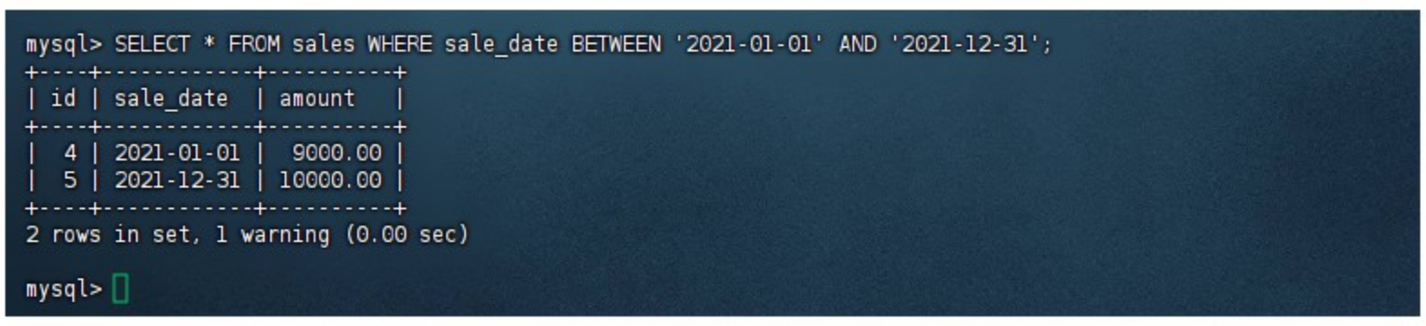
```
mysql> SELECT * FROM sales WHERE sale_date BETWEEN '2020-01-01' AND '2020-12-31';
+----+------------+---------+
| id | sale_date  | amount  |
+----+------------+---------+
|  2 | 2020-01-01 | 7000.00 |
|  3 | 2020-12-31 | 8000.00 |
+----+------------+---------+
2 rows in set, 1 warning (0.00 sec)

mysql>
```



10、查询 2021 年的数据

SELECT * FROM sales WHERE sale_date BETWEEN ‘2021-01-01’ AND ‘2021-12-31’;

```
mysql> SELECT * FROM sales WHERE sale_date BETWEEN '2021-01-01' AND '2021-12-31';
+----+------------+----------+
| id | sale_date  | amount   |
+----+------------+----------+
|  4 | 2021-01-01 |  9000.00 |
|  5 | 2021-12-31 | 10000.00 |
+----+------------+----------+
2 rows in set, 1 warning (0.00 sec)

mysql>
```



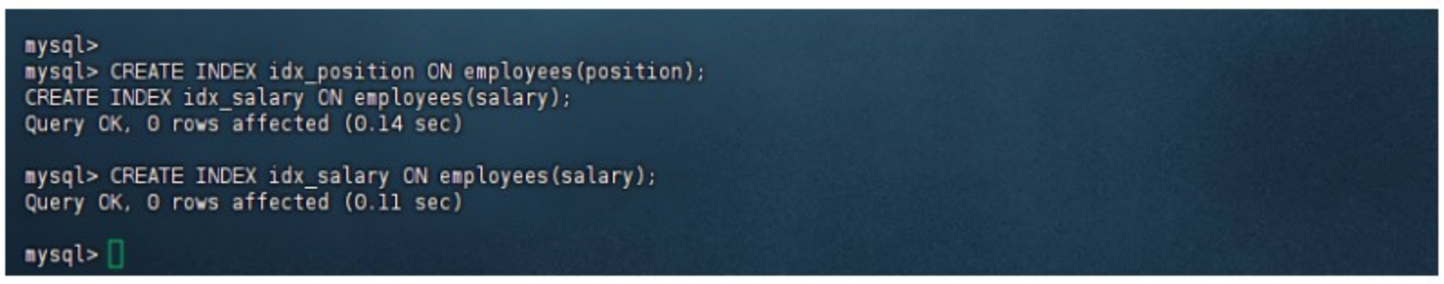
三、数据库优化实战

1、普通表的优化与分析

我们可以对表进行索引优化以提高查询效率：

1.1 在经常用于搜索的列上添加索引

```
mysql> CREATE INDEX idx_position ON employees(position);
CREATE INDEX idx_salary ON employees(salary);
Query OK, 0 rows affected (0.14 sec)

mysql> CREATE INDEX idx_salary ON employees(salary);
Query OK, 0 rows affected (0.11 sec)

mysql>
```



1.2 分析查询性能

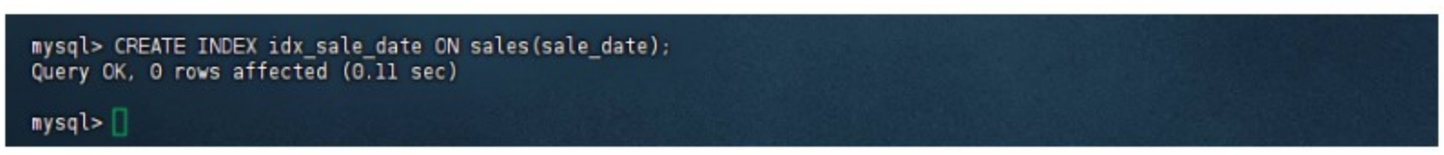
```
mysql> EXPLAIN SELECT * FROM employees WHERE position = 'Software Engineer';
+-------------------------------+---------+-----------+-----------------------------------
| id                            | estRows | task      | access object
+-------------------------------+---------+-----------+-----------------------------------
| IndexLookUp_10                | 0.01    | root      |
| ├─IndexRangeScan_8(Build)     | 0.01    | cop[tikv] | table:employees, index:idx_positior
| └─TableRowIDScan_9(Probe)     | 0.01    | cop[tikv] | table:employees
+-------------------------------+---------+-----------+-----------------------------------
3 rows in set (0.00 sec)

mysql>
```

2、分区表的优化与分析

我们可以对表进行索引优化以提高查询效率：
确保 sale_date 列上有索引，以便加速范围查询。

2.1 在经常用于搜索的列上添加索引

```
mysql> CREATE INDEX idx_sale_date ON sales(sale_date);
Query OK, 0 rows affected (0.11 sec)

mysql>
```



2.2 分析查询性能

TiDB 支持分区裁剪（Partition Pruning），在查询时会自动跳过无关分区，提升查询效率。例如：

```
mysql> EXPLAIN SELECT * FROM sales WHERE sale_date BETWEEN '2020-01-01' AND '2020-12-31';
+-------------------------+---------+-----------+---------------+--------------------------
| id                      | estRows | task      | access object | operator info
+-------------------------+---------+-----------+---------------+--------------------------
| TableReader_7           | 2.00    | root      | partition:p1  | data:Selection_6
| └─Selection_6           | 2.00    | cop[tikv] |               | ge(testdb.sales.sale_date
|   └─TableFullScan_5     | 6.00    | cop[tikv] | table:sales   | keep order:false
+-------------------------+---------+-----------+---------------+--------------------------
3 rows in set (0.01 sec)

mysql>
```

2.3 查看执行计划，确认是否只扫描了相关的分区。

```
mysql> EXPLAIN SELECT * FROM sales WHERE sale_date BETWEEN '2020-01-01' AND '2020-12-31';
+-------------------------+---------+-----------+---------------+--------------------------------------------------------------------------------------------------
-------------------------------------+
| id                      | estRows | task      | access object | operator info
                                     |
+-------------------------+---------+-----------+---------------+--------------------------------------------------------------------------------------------------
-------------------------------------+
| TableReader_7           | 2.00    | root      | partition:p1  | data:Selection_6
                                     |
| └─Selection_6           | 2.00    | cop[tikv] |               | ge(testdb.sales.sale_date, 2020-01-01 00:00:00.000000), le(testdb.sales.
sale_date, 2020-12-31 00:00:00.000000) |
|   └─TableFullScan_5     | 6.00    | cop[tikv] | table:sales   | keep order:false
                                     |
+-------------------------+---------+-----------+---------------+--------------------------------------------------------------------------------------------------
-------------------------------------+
3 rows in set (0.01 sec)

mysql>
```

避免过度分区：
分区数量不宜过多，否则可能增加元数据管理的开销。通常建议根据业务需求合理划分分区范围。
在 TiDB 中创建分区表时，必须确保分区列与主键或唯一索引的关系符合其约束条件。
通过对比未加索引前后的查询性能，可以看到索引显著提升了查询速度。此外，TiDB 提供了分布式事务支持，使得大规模数据处理更加高效可靠。特别是在处理大量数据时，分区表可以有效提升查询性能和管理便利性。

总结

本篇征文介绍了如何在 CentOS 7 上利用 Docker Compose 快速搭建 TiDB 7.5 集群，并展示了基本的数据库操作如创建数据库、表以及数据插入。特别地，我们演示了如何创建分区表并对其进行优化。通过这些实践，可以更好地理解 TiDB 的功能特性及其应用场景，为后续深入学习和应用打下坚实基础。希望这篇教程能帮助更多开发者顺利部署并使用 TiDB 进行开发，享受其带来的高效和灵活性。同时，通过实际操作中的对比分析，进一步认识到了优化数据库设计的重要性。

关键点总结

1. **环境准备**：确保 Docker 和 Docker Compose 已正确安装。
2. **集群部署**：使用 Docker Compose 配置和启动 PD、TiKV 和 TiDB 服务。
3. **数据库操作**：创建数据库、表，插入数据，并进行查询。
4. **性能优化**：通过调整 TiDB 的启动参数来优化性能。
5. **对比测试**：通过插入数据和查询来验证 TiDB 的性能。

通过这些步骤，您可以快速上手 TiDB，并在实际业务中应用其强大的功能。

—— 仅供参考。如果有更多具体的问题或需要进一步的帮助，请随时告知。

docker tidb docker-compose tidb数据库 tidb第四届征文-运维开发之旅

最后修改时间：2025-04-28 11:18:26

评论

分享你的看法，一起交流吧～

相关阅读

开赛 | KWDB 核心贡献挑战赛，30万奖金池等你来瓜分！

KaiwuDB 283次阅读 2025-07-25 10:07:45

【GaussDB】构建一个GaussDB的Docker镜像

DarkAthena 273次阅读 2025-07-26 17:54:22

IDC 2024年中国金融行业分布式事务型数据库市场份额：腾讯云第一、阿里云第二、OceanBase第三

通讯员 206次阅读 2025-08-01 15:34:10

Greenplum、OceanBase、Oracle 和 TiDB 这四种数据库的核心架构区别

陈耀斌 178次阅读 2025-07-17 14:21:52

TEM on Cloud 试用指南：值得 DBA 花时间吗？

Lucifer三思而后行 118次阅读 2025-07-27 16:16:54

数据库之路-第5期-超强的运维管理平台，TEM on 腾讯云安装 + TiDB 集群实践

悟空聊架构 86次阅读 2025-07-28 23:47:19

数据库之路-第 6期 Mac M1 部署 TiDB

悟空聊架构 82次阅读 2025-07-30 11:53:05

为可观测性注入LLM ：结合 ClickStack、OpenTelemetry 与 MCP

ClickHouseInc 69次阅读 2025-07-24 10:00:35

扣子开源了，白捡一百多万（完整部署步骤）

数据库平台优化 68次阅读 2025-07-28 12:24:13

Up！使用 Navicat Premium 连接平凯数据库（TiDB 企业版）敏捷模式

严少安 54次阅读 2025-07-22 14:39:34