

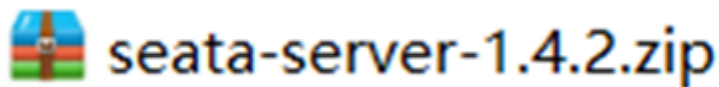
seata的部署和集成

一、部署Seata的tc-server

1. 下载

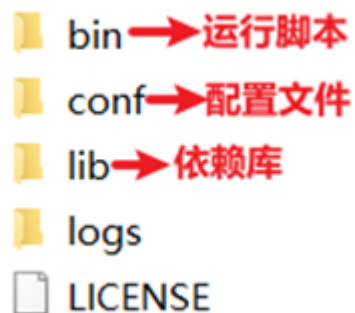
首先我们要下载seata-server包，地址在<http://seata.io/zh-cn/blog/download.html>

当然，课前资料也准备好了：



2. 解压

在非中文目录解压缩这个zip包，其目录结构如下：



3. 修改配置

修改conf目录下的registry.conf文件：

Lesson > seata > conf >

名称

- logback
- META-INF
- file.conf
- file.conf.example
- logback.xml
- README.md
- README-zh.md
- registry.conf

内容如下:

```
1 registry {
2   # tc服务的注册中心类，这里选择nacos，也可以是eureka、zookeeper
  等
3   type = "nacos"
4
5   nacos {
6     # seata tc 服务注册到 nacos的服务名称，可以自定义
7     application = "seata-tc-server"
8     serverAddr = "127.0.0.1:8848"
9     group = "DEFAULT_GROUP"
10    namespace = ""
11    cluster = "SH"
12    username = "nacos"
13    password = "nacos"
14  }
15 }
16
17 config {
18   # 读取tc服务端的配置文件的方式，这里是从nacos配置中心读取，这样
  如果tc是集群，可以共享配置
19   type = "nacos"
20   # 配置nacos地址等信息
21   nacos {
22     serverAddr = "127.0.0.1:8848"
23     namespace = ""
24     group = "SEATA_GROUP"
25     username = "nacos"
26     password = "nacos"
27     dataId = "seataServer.properties"
28   }
29 }
```

4. 在nacos添加配置

特别注意，为了让tc服务的集群可以共享配置，我们选择了nacos作为统一配置中心。因此服务端配置文件seataServer.properties文件需要在nacos中配好。

格式如下：

新建配置

* Data ID: seataServer.properties

* Group: DEFAULT_GROUP

[更多高级选项](#)

描述:

配置格式: ☐ TEXT ☐ JSON ☐ XML ☐ YAML ☐ HTML ☒ Properties

* 配置内容: ? : 1

配置内容如下：

```
1 # 数据存储方式，db代表数据库
2 store.mode=db
3 store.db.datasource=druid
4 store.db.dbType=mysql
5 store.db.driverClassName=com.mysql.jdbc.Driver
6 store.db.url=jdbc:mysql://127.0.0.1:3306/seata?
  useUnicode=true&rewriteBatchedStatements=true
7 store.db.user=root
8 store.db.password=123
9 store.db.minConn=5
10 store.db.maxConn=30
11 store.db.globalTable=global_table
12 store.db.branchTable=branch_table
13 store.db.queryLimit=100
14 store.db.lockTable=lock_table
15 store.db.maxWait=5000
16 # 事务、日志等配置
17 server.recovery.committingRetryPeriod=1000
18 server.recovery.asyncCommittingRetryPeriod=1000
19 server.recovery.rollbackingRetryPeriod=1000
20 server.recovery.timeoutRetryPeriod=1000
```

```

21 server.maxCommitRetryTimeout=-1
22 server.maxRollbackRetryTimeout=-1
23 server.rollbackRetryTimeoutUnlockEnable=false
24 server.undo.logSaveDays=7
25 server.undo.logDeletePeriod=86400000
26
27 # 客户端与服务端传输方式
28 transport.serialization=seata
29 transport.compressor=none
30 # 关闭metrics功能，提高性能
31 metrics.enabled=false
32 metrics.registryType=compact
33 metrics.exporterList=prometheus
34 metrics.exporterPrometheusPort=9898

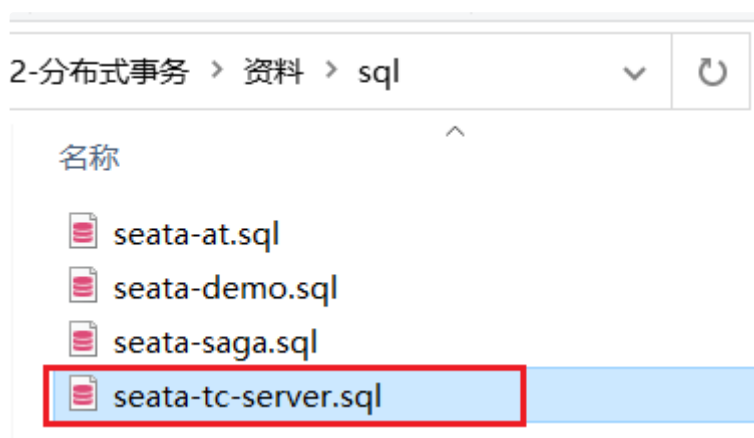
```

其中的数据库地址、用户名、密码都需要修改成你自己的数据库信息。

5. 创建数据库表

特别注意：tc服务在管理分布式事务时，需要记录事务相关数据到数据库中，你需要提前创建好这些表。

新建一个名为seata的数据库，运行课前资料提供的sql文件：



这些表主要记录全局事务、分支事务、全局锁信息：

```

1
2 SET NAMES utf8mb4;
3 SET FOREIGN_KEY_CHECKS = 0;
4
5 -- -----
6 -- 分支事务表
7 -- -----
8 DROP TABLE IF EXISTS `branch_table`;
9 CREATE TABLE `branch_table` (
10   `branch_id` bigint(20) NOT NULL,
11   `xid` varchar(128) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,

```

```

12     `transaction_id` bigint(20) NULL DEFAULT NULL,
13     `resource_group_id` varchar(32) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL,
14     `resource_id` varchar(256) CHARACTER SET utf8 COLLATE utf8_general_ci
NULL DEFAULT NULL,
15     `branch_type` varchar(8) CHARACTER SET utf8 COLLATE utf8_general_ci
NULL DEFAULT NULL,
16     `status` tinyint(4) NULL DEFAULT NULL,
17     `client_id` varchar(64) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL,
18     `application_data` varchar(2000) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL,
19     `gmt_create` datetime(6) NULL DEFAULT NULL,
20     `gmt_modified` datetime(6) NULL DEFAULT NULL,
21     PRIMARY KEY (`branch_id`) USING BTREE,
22     INDEX `idx_xid` (`xid`) USING BTREE
23 ) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8_general_ci
ROW_FORMAT = Compact;
24
25 -- -----
26 -- 全局事务表
27 -- -----
28 DROP TABLE IF EXISTS `global_table`;
29 CREATE TABLE `global_table` (
30     `xid` varchar(128) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
31     `transaction_id` bigint(20) NULL DEFAULT NULL,
32     `status` tinyint(4) NOT NULL,
33     `application_id` varchar(32) CHARACTER SET utf8 COLLATE utf8_general_ci
NULL DEFAULT NULL,
34     `transaction_service_group` varchar(32) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL,
35     `transaction_name` varchar(128) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL,
36     `timeout` int(11) NULL DEFAULT NULL,
37     `begin_time` bigint(20) NULL DEFAULT NULL,
38     `application_data` varchar(2000) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL,
39     `gmt_create` datetime NULL DEFAULT NULL,
40     `gmt_modified` datetime NULL DEFAULT NULL,
41     PRIMARY KEY (`xid`) USING BTREE,
42     INDEX `idx_gmt_modified_status` (`gmt_modified`, `status`) USING BTREE,
43     INDEX `idx_transaction_id` (`transaction_id`) USING BTREE
44 ) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8_general_ci
ROW_FORMAT = Compact;
45
46 SET FOREIGN_KEY_CHECKS = 1;


```


6. 启动TC服务

进入bin目录，运行其中的seata-server.bat即可：

D:) > lesson > seata > bin

名称

 seata-server.bat

 seata-server.sh

启动成功后，seata-server应该已经注册到nacos注册中心了。

打开浏览器，访问nacos地址：<http://localhost:8848>，然后进入服务列表页面，可以看到seata-tc-server的信息：

NACOS 1.4.1

配置管理

服务管理

服务列表

订阅者列表

权限控制

public

服务列表 | public

服务名称

分组名称

隐藏空服务: ☒

查询

服务名	分组名称	集群数目	实例数	健康实例数
seata-tc-server	DEFAULT_GROUP	1	1	1

二、微服务集成seata

1. 引入依赖

首先，我们需要在微服务中引入seata依赖：

```
1 <dependency>
2   <groupId>com.alibaba.cloud</groupId>
3   <artifactId>spring-cloud-starter-alibaba-seata</artifactId>
4   <exclusions>
5     <!-- 版本较低，1.3.0，因此排除 -->
6     <exclusion>
7       <artifactId>seata-spring-boot-starter</artifactId>
8       <groupId>io.seata</groupId>
9     </exclusion>
10  </exclusions>
11 </dependency>
12 <!-- seata starter 采用1.4.2版本 -->
13 <dependency>
14   <groupId>io.seata</groupId>
15   <artifactId>seata-spring-boot-starter</artifactId>
16   <version>${seata.version}</version>
17 </dependency>
```

2. 修改配置文件

需要修改application.yml文件，添加一些配置：

```
1 seata:
2   registry: # TC服务注册中心的配置，微服务根据这些信息去注册中心获取tc服务地址
3     # 参考tc服务自己的registry.conf中的配置
4     type: nacos
5     nacos: # tc
6       server-addr: 127.0.0.1:8848
7       namespace: ""
8       group: DEFAULT_GROUP
9       application: seata-tc-server # tc服务在nacos中的服务名称
10      cluster: SH
11    tx-service-group: seata-demo # 事务组，根据这个获取tc服务的cluster名称
12    service:
13      vgroup-mapping: # 事务组与TC服务cluster的映射关系
14      seata-demo: SH
```

三、TC服务的高可用和异地容灾

1. 模拟异地容灾的TC集群

计划启动两台seata的tc服务节点：

节点名称	ip地址	端口号	集群名称
seata	127.0.0.1	8091	SH
seata2	127.0.0.1	8092	HZ

之前我们已经启动了一台seata服务，端口是8091，集群名为SH。

现在，将seata目录复制一份，起名为seata2

修改seata2/conf/registry.conf内容如下：

```
1 registry {
2   # tc服务的注册中心类，这里选择nacos，也可以是eureka、zookeeper等
3   type = "nacos"
4
5   nacos {
6     # seata tc 服务注册到 nacos的服务名称，可以自定义
```

```

7     application = "seata-tc-server"
8     serverAddr = "127.0.0.1:8848"
9     group = "DEFAULT_GROUP"
10    namespace = ""
11    cluster = "HZ"
12    username = "nacos"
13    password = "nacos"
14  }
15 }
16
17 config {
18   # 读取tc服务端的配置文件的方式，这里是从nacos配置中心读取，这样
   # 如果tc是集群，可以共享配置
19   type = "nacos"
20   # 配置nacos地址等信息
21   nacos {
22     serverAddr = "127.0.0.1:8848"
23     namespace = ""
24     group = "SEATA_GROUP"
25     username = "nacos"
26     password = "nacos"
27     dataId = "seataServer.properties"
28   }
29 }

```

进入seata2/bin目录，然后运行命令：

```
1 seata-server.bat -p 8092
```

打开nacos控制台，查看服务列表：

服务名	分组名称	集群数目	实例数	健康实例数
seata-tc-server	DEFAULT_GROUP	2	2	2

点进详情查看：

集群: HZ					
元数据过滤 <input type="text" value="key"/> <input type="text" value="value"/> <button>添加过滤</button>					
IP	端口	临时实例	权重	健康状态	元数据
192.168.150.1	8092	true	1	true	

集群: SH					
元数据过滤 <input type="text" value="key"/> <input type="text" value="value"/> <button>添加过滤</button>					
IP	端口	临时实例	权重	健康状态	元数据
192.168.150.1	8091	true	1	true	

2. 将事务组映射配置到nacos

接下来，我们需要将tx-service-group与cluster的映射关系都配置到nacos配置中心。

新建一个配置：

新建配置

* Data ID:

client.properties

* Group:

SEATA_GROUP

更多高级选项

描述:

配置格式:

☐ TEXT ☐ JSON ☐ XML ☐ YAML ☐ HTML ☒ Properties

* 配置内容: ? :

1

配置的内容如下：

```
1 # 事务组映射关系
2 service.vgroupMapping.seata-demo=SH
3
4 service.enableDegrade=false
5 service.disableGlobalTransaction=false
```

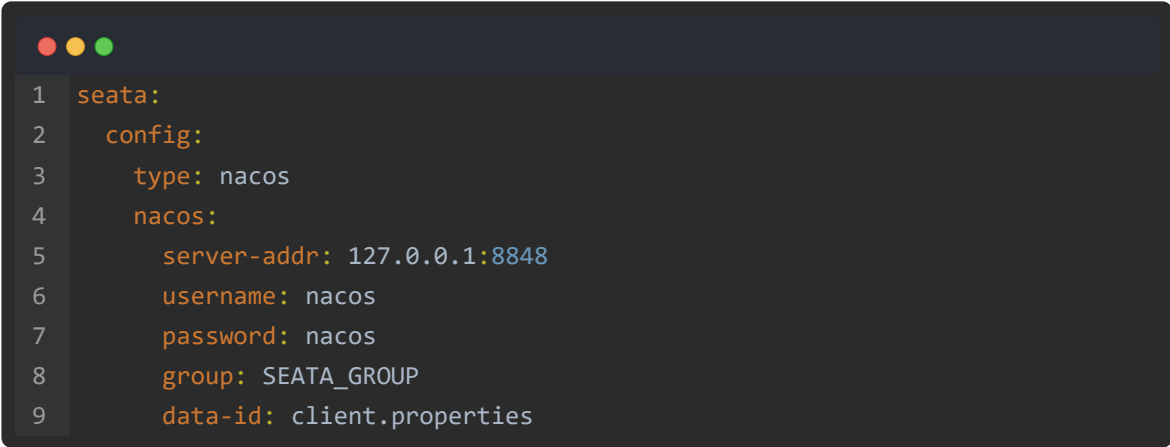
```

6 # 与TC服务的通信配置
7 transport.type=TCP
8 transport.server=NIO
9 transport.heartbeat=true
10 transport.enableClientBatchSendRequest=false
11 transport.threadFactory.bossThreadPrefix=NettyBoss
12 transport.threadFactory.workerThreadPrefix=NettyServerNIOWorker
13 transport.threadFactory.serverExecutorThreadPrefix=NettyServerBizHandler
14 transport.threadFactory.shareBossWorker=false
15 transport.threadFactory.clientSelectorThreadPrefix=NettyClientSelector
16 transport.threadFactory.clientSelectorThreadSize=1
17 transport.threadFactory.clientWorkerThreadPrefix=NettyClientWorkerThread
18 transport.threadFactory.bossThreadSize=1
19 transport.threadFactory.workerThreadSize=default
20 transport.shutdown.wait=3
21 # RM配置
22 client.rm.asyncCommitBufferLimit=10000
23 client.rm.lock.retryInterval=10
24 client.rm.lock.retryTimes=30
25 client.rm.lock.retryPolicyBranchRollbackOnConflict=true
26 client.rm.reportRetryCount=5
27 client.rm.tableMetaCheckEnable=false
28 client.rm.tableMetaCheckerInterval=60000
29 client.rm.sqlParserType=druid
30 client.rm.reportSuccessEnable=false
31 client.rm.sagaBranchRegisterEnable=false
32 # TM配置
33 client.tm.commitRetryCount=5
34 client.tm.rollbackRetryCount=5
35 client.tm.defaultGlobalTransactionTimeout=60000
36 client.tm.degradeCheck=false
37 client.tm.degradeCheckAllowTimes=10
38 client.tm.degradeCheckPeriod=2000
39
40 # undo日志配置
41 client.undo.dataValidation=true
42 client.undo.logSerialization=jackson
43 client.undo.onlyCareUpdateColumns=true
44 client.undo.logTable=undo_log
45 client.undo.compress.enable=true
46 client.undo.compress.type=zip
47 client.undo.compress.threshold=64k
48 client.log.exceptionRate=100

```

3. 微服务读取nacos配置

接下来，需要修改每一个微服务的application.yml文件，让微服务读取nacos中的client.properties文件：



```
1 seata:
2   config:
3     type: nacos
4     nacos:
5       server-addr: 127.0.0.1:8848
6       username: nacos
7       password: nacos
8       group: SEATA_GROUP
9       data-id: client.properties
```

重启微服务，现在微服务到底是连接tc的SH集群，还是tc的HZ集群，都统一由nacos的client.properties来决定了。