PolarDB国产轻量化数据库用户手册

纯内核部署 (单节点部署, 推荐)

形态简介

仅提供 `PolarDB-PG` 国产数据库内核rpm安装包, `PolarDB-PG` 本身是基于社区PostgreSQL版本14开发, 兼容社区PostgreSQL 14版本所有功能, 也与各开源PostgreSQL的第三方组件兼容。

系统安装

以rpm包 `PolarDB-2.0.14.16.3-20240708092447.alios7.x86_64.rpm` 为例, 使用如下方式安装即可:

```
rpm -ivh PolarDB-2.0.14.16.3-20240708092447.alios7.x86_64.rpm
```

或

```
yum install -y PolarDB-2.0.14.16.3-20240708092447.alios7.x86_64.rpm
```

初始化及启动

创建用户

需要创建一个用户供数据库运维使用。

```
useradd polardb
su -l polardb
```

手动创建数据库

初始化

`$DATA_PATH` 为数据库文件所在绝对路径, 根据实际情况初始化即可。

```
/u01/polardb_pg/bin/initdb -D $DATA_PATH
```

启动 & 停止

```
# 启动
/u01/polardb_pg/bin/pg_ctl -D $DATA_PATH -l logfile start

# 停止
/u01/polardb_pg/bin/pg_ctl stop -D $DATA_PATH

# 重启
/u01/polardb_pg/bin/pg_ctl restart -D $DATA_PATH
```

一键脚本

复制代码

```
#!/bin/bash
# init_polardb_pg_14.sh: init single node polardb pg 14
#
# Copyright (c) 2024, Alibaba Group Holding Limited
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

POLARDB_PG_BIN="/u01/polardb_pg/bin/"

## connection info
DEFAULT_HOST="/tmp"
DEFAULT_USER="postgres"

## args
### init port
INIT_PORT=5432
### init user
INIT_USER="kong"
### init password
INIT_PASS=""
### init data path
unset -v INIT_DATAPATH

function usage() {
```

```
    echo "Usage:"
    echo "    -d  init polardb pg data dir"
    echo "    -p  init polardb pg listen port"
    echo "    -U  init polardb pg user name"
    echo "    -P  init polardb pg user password"
    echo ""
    echo " You can also set db config in enviroment by name starting with
'POLARDB_PARAMS_' like this: "
    echo "    POLARDB_PARAMS_max_connections=512 sh $0 -d DATAPATH -p 5432 -U user -P
pass"
}

function initdb() {
    $POLARDB_PG_BIN/initdb -D $1
    $POLARDB_PG_BIN/pg_ctl -o "-F -p $2" -D $1 -l logfile start
}

function restart_db() {
    $POLARDB_PG_BIN/pg_ctl restart -mi -D $INIT_DATAPATH
}

function psql() {
    $POLARDB_PG_BIN/psql -h $DEFAULT_HOST -p $INIT_PORT -U $DEFAULT_USER -c "$1"
}

function set_db_single_config() {
   psql "ALTER SYSTEM SET $1=$2"
}

function set_db_configs() {
    set_db_single_config "listen_addresses" "\"*\""
    set_db_single_config "logging_collector" "on"

    env | while IFS= read -r line; do
        value=${line#*=}
        key=${line%%=*}

        if [[ $key == "POLARDB_PARAMS_"* ]]
        then
            set_db_single_config ${key:15} $value
        fi
    done

    psql "SELECT pg_reload_conf()"
}

function set_db_hba_config() {
    echo "host    all             all             0.0.0.0/0               md5" >>
$1/pg_hba.conf
```

```
    echo "host    all             all             ::0/0                   md5" >>
$1/pg_hba.conf
}

function create_db_user() {
    psql "CREATE USER $1 WITH SUPERUSER PASSWORD '$2'"
}

while getopts 'd:p:U:P:h' opt;
do
    case ${opt} in
        d)
            INIT_DATAPATH=${OPTARG};;
        p)
            INIT_PORT=${OPTARG};;
        P)
            INIT_PASS=${OPTARG};;
        U)
            INIT_USER=${OPTARG};;
        h)
            usage
            exit 0
            ;;
        *)
            usage
            exit 0
            ;;
    esac
done

if [ -z "$INIT_DATAPATH" ]; then
    echo "missing -d" >&2
    usage
    exit 1
fi

COMMAND="install"
if [[ $COMMAND =  "install" ]]
then
    initdb $INIT_DATAPATH $INIT_PORT
    create_db_user $INIT_USER $INIT_PASS
    set_db_configs
    set_db_hba_config $INIT_DATAPATH
    restart_db $INIT_DATAPATH
else
    usage
    exit -1
fi
```

调用方式如下：

```
sh init_polardb_pg_14.sh -d $DATA_PATH
```

访问

使用如下命令即可访问数据库：

```
/u01/polardb_pg/bin/psql -h /tmp -p 5432 -U polardb postgres
```

License授权（购买后申请）

- License用于限定客户是否可以正常使用PolarDB数据库，License通过系统识别码与使用时长进行可用性控制，License过期后数据库在运行状态下可正常使用，但数据库无法再次重新启动；
- 不申请License安装包可以用于系统测试，但具有最大连接限制，影响生产环境中应用；启用License后限流解除；

获取system identifier

```
/u01/polardb_pg/bin/pg_controldata -D data/ | grep 'Database system identifier'
```

```
$/u01/polardb_pg/bin/pg_controldata -D data/ | grep 'Database system identifier'
Database system identifier:           7389817754324353002
```

获取License

- 客户通过阿里云前线同学（或云市场下单后通过售前支持钉钉）申请license文件，申请License需要提交 `system identifer` 与购买订单号；
- 产品经理确认购买订单后，根据客户生成License（内部操作）， license文件名为 `license.lic` ；

启用License

- 将生成的license.lic放到pg数据目录，即 `$DATA_PATH` ；
- 启动实例时会检测 `license.lic` 文件中信息，无license文件数据库会被限流，无法用于生产环境；

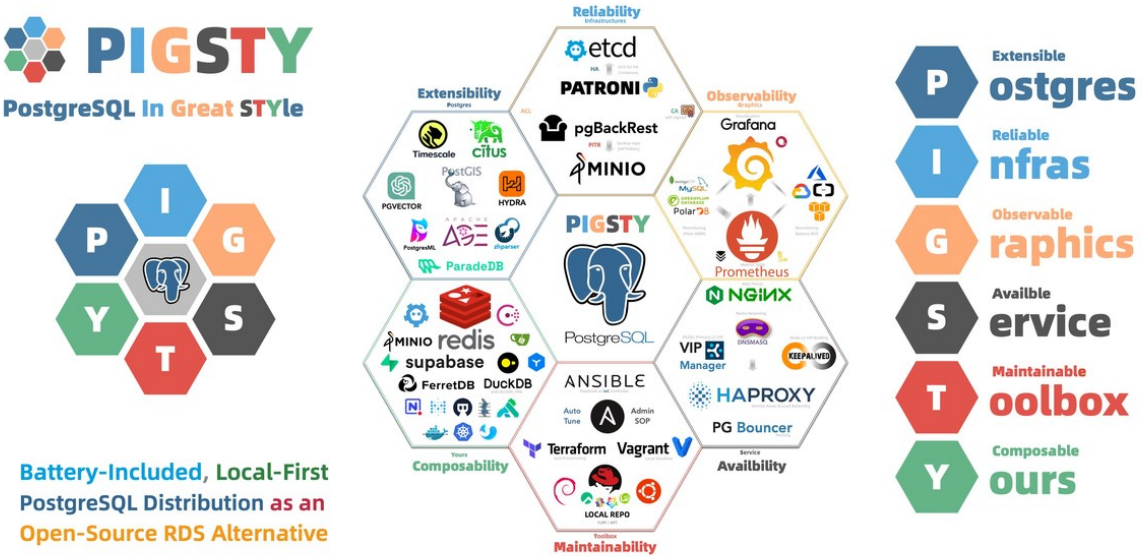
系统对接方案

兼容社区PostgreSQL 14版本所有功能，也与各开源PostgreSQL的第三方组件兼容。

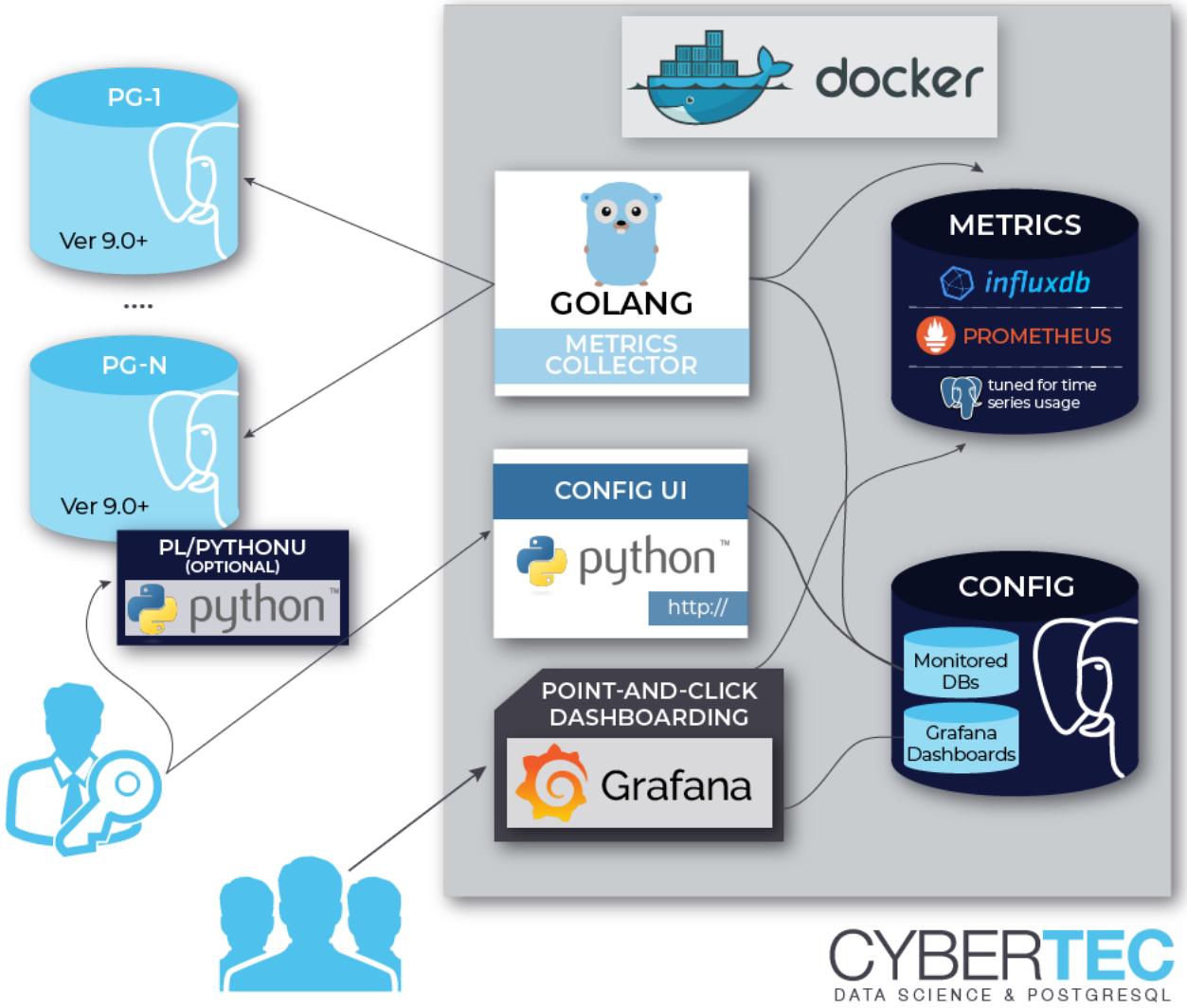
监控

开源社区贡献了不少监控组件

- pigsty：国内PG社区大V冯若航的创业项目，不仅包含了监控内容本身，还提供了高可用组件、备份组件，组件生态相对完善。



- pgwatch2: 社区知名的监控工具，包含了prometheus和grafana, 以易部署、易维护著称。支持自建的pg环境, 支持自定义监控采集SQL.



备份

原生PG即提供了附带的备份组件, 既包含了物理备份工具和逻辑备份工具。

原生备份能力

原生PostgreSQL提供两个命令行工具, 分别支持物理数据备份及逻辑数据备份:

- pg_basebackup: PostgreSQL提供的原生物理数据备份工具, 可以将数据目录进行整体备份, 外加WAL备份后, 可在恢复时精确到秒级恢复;
- pg_dump: PostgreSQL提供的原生逻辑备份工具, 可以将数据dump成SQL文件, 并可导入不同版本的自建PostgreSQL数据库;

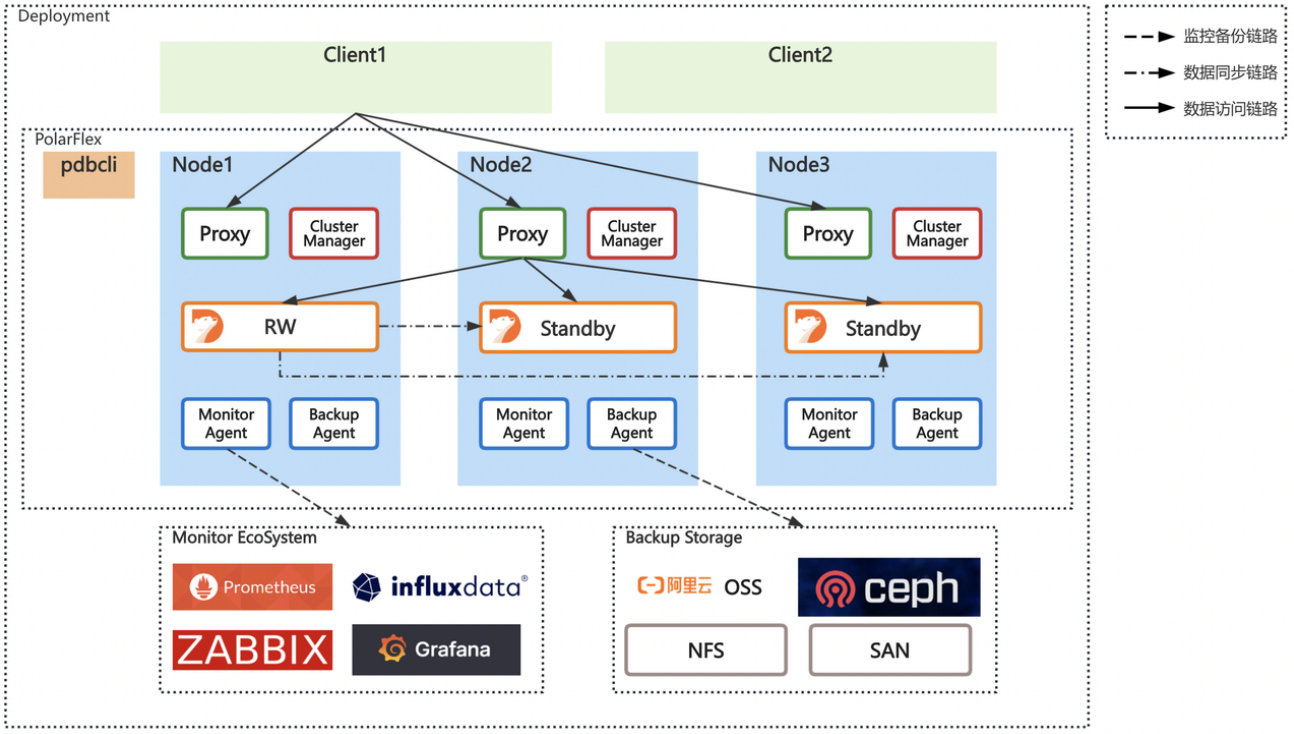
社区备份工具

- pg_probackup: PostgrePro公司提供的增强型备份工具，提供块级增量备份等企业级备份功能。

软件栈部署 (主备形态)

形态简介

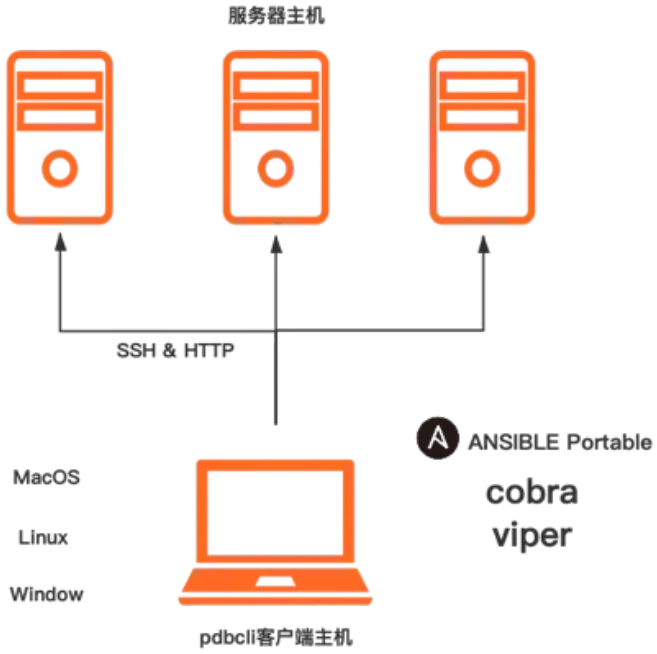
除了内核rpm包以外, PolarDB国产轻量化产品还支持完整的软件栈部署形态 `PolarFlex` , 提供一写多读、高可用、监控、备份等基本能力，范围以下图 `PolarFlex` 标注的虚线框为准:



`PolarFlex` 包含如下核心组件:

- `pdbcli` ： 安装部署的客户端组件，作为整个 `PolarFlex` 集群的管理客户端，可进行数据库实例部署、添加Standby节点、高可用切换、版本升级等操作.
- `Proxy` ： 数据库代理组件, 客户端在访问数据库时直接访问代理组件, 可自动实现数据库集群的读写分离功能,
- `Cluster Manager` ： 数据库集群的高可用组件, 在主节点出现宕机或者挂掉时, 可即时选择可用备节点进行切换, 亦可手动进行切换。高可用组件本身也使用三节点形态部署, 其利用raft协议保障自身的高可用。
- `Monitor Agent` ： 监控采集组件，会采集数据库、Proxy、主机等组件的监控数据，提供prometheus、influxdb或者zabbix等开源监控数据存储组件的对接能力
- `Backup Agent` : 备份组件, 相对于社区的备份组件 `pg_basebackup` ， `backup agent` 支持更多的备份介质, 同时提供并行备份以及增量备份的能力.

系统部署



pdbcli安装

- 下载名为 `polarflex-${version}-${build-date}.tar.gz` 的安装包，并将其发送至安装主机上。
- 登录安装主机，以 `version 2.2.2.3` 为例，执行以下命令创建工作目录：

```
version=2.2.2.3
mkdir -p polarflex-${version}
tar -C polarflex-${version}/ -xf polarflex-${version}-${build-date}.tar.gz
```

- 执行以下 `install.sh` 命令进入工作目录并开始安装，该命令需要 `root` 权限

```
cd polarflex-${version}/
./scripts/install.sh
```

- 安装完成后，执行如下命令确认安装的版本，版本无误即表明安装正确

```
pdbcli version
```

前置操作

所有机器打通ssh免密

```
# 生成ssh key
ssh-keygen -t rsa

# 复制公钥到所有部署机器上
ssh-copy-id -i ~/.ssh/id_rsa.pub root@$HOST
```

安装polarflex软件栈

复制代码

```
pdbcli install cluster
```

部署集群

准备配置文件

`config.yaml` 包括主机存储和配置参数。可以以 `pdbcli-${version}/` 文件夹下的 `config.yaml` 文件为模板，按照注释根据实际情况进行编辑。内容如下所示:

复制代码

```
# 说明 完成请根据已有的config.yaml文件中的注释信息，结合实际部署环境的情况，修改配置文件
# 中的内容。修改完成后，可执行 pdbcli validate 验证配置是否有问题。关于 pdbcli validate ，
# 具体可参见验证配置文件。

# 注意 以下配置文件示例展示了如何搭建一个由三台主机构建的集群，主机IP分别为10.XX.XX.1、
# 10.0.0.2、10.0.0.3。请您根据实际情况进行配置。

## config.yaml文件中的内容如下，请根据实际情况和需求进行配置。
all:

  ## 填写所有主机的信息，包括主机名称、IP地址等。需确保各目标主机建立对pdbcli主机的ssh信任关系。
  hosts:

    ## 主机名称按照 hostNN 来命名，例如host01。
    host01:

      ## 提供ansible_host指定主机的IP地址
      ansible_host: 10.0.0.1

      ## 存储设备名称
      # storage_device: vdc

    host02:
      ansible_host: 10.0.0.2

    host03:
      ansible_host: 10.0.0.3
```

```
vars:
    ## 如下为具体的集群配置。
    ## 设置PolarDB O引擎数据库引擎的安装根目录，默认为:/var/local/polardb
    # polardb_data_root_dir: /var/local/polardb
    ## 设置PolarDB O引擎 CM(Cluster Manager)的安装根目录，默认为:/var/local/polardb
    # polardb_cm_root_dir: /var/local/polardb_cluster_manager
    ## 【必填】设置cluster_id，即集群ID。该设置会影响安装文件夹。
    ## 警告:在数据库集群创建后，请勿修改该ID。
    cluster_id: mycluster
    ## 设置主库节点。如果未指定，则默认指定all.children.db[0]作为主库。
    primary_db_host: host01
    ## 【必填】设置external_storage_path，即外部存储路径。
    ## 填写绝对路径。若未分配或为空，则默认使用$PGDATA目录。
    ## 数据库集群创建后，请勿修改该外部存储路径。
    external_storage_path: /mnt/mycluster
    ## 系统参数设置
    ## 开启或关闭firewalld防火墙服务，默认为false，即关闭状态。如果开启，则还需要手动打开服务对应端口.
    firewalld_enabled: false

    ## 数据库参数设置
    ## 设置监听端口，默认值为1521。
    # polardb_port: 1521

    ## 设置最大连接数，默认为2048。
    # polardb_max_connections: 2048
    ## PolarDB replication账户设置
    ## 警告:数据库集群创建后，请勿修改该账户设置。
    ## 设置PolarDB replication用户名，默认为user_rep。
    # polardb_rep_username: user_rep
    ## 设置PolarDB replication密码，默认为pgsql。
    # polardb_rep_password: pgsql
    ## CM参数设置
    ## CM服务 HTTP 监听端口，默认为5500。
    # cm_service_port: 5500## CM服务 HTTPS 监听端口，默认为5501。
    # cm_tls_service_port: 5501
    ## CM consensus服务监听端口，默认为5502。
    # cm_consensus_port: 5502
    ## CM是否开启TLS，默认不开启，即false。当前版本暂不支持开启。
    # cm_tls_enabled: false
    ## proxy参数设置
    ## PolarDB Proxy工作并发数，默认为2。
    # polardb_proxy_concurrency: 2
    ## PolarDB Proxy服务端口，默认为12366。
    # polardb_proxy_port: 12366
    ## PolarDB Proxy管理服务端口，默认为12367。
    # polardb_proxy_admin_port: 12367
    ## RW_TYPE为1时有效。主节点是否参与读请求的负载均衡。默认为true。如果设置为false，则读请求不发往主库。
```

```
    # polardb_proxy_master_accept_ready: true

    ## 是否开启事务拆分。默认为true。如果设置为false，事务所有请求路由到主库;如果设置为true，则事务
中写之前的读请求可以路由到只读库。写之后的读还是路由到主库。

    # polardb_proxy_trx_split: true

    ## 是否开启会话一致性。默认为true。如果设置为false，不保证会话内读写一致性;如果设置为true，保证
会话一致性。效果为同一个连接内，读请求一定能读到这个连接之前写入的数据。

    # polardb_proxy_casual_reads: true
    ## agent参数设置## Node Driver服务端口，默认为12355。
    # ue_node_driver_service_port: 12355
    ## -- 以下内容请勿改动 --

    ## ansible varsansible_group_priority: 99
    # uncomment following if you are using CentOS7 or RHEL7
ansible_python_interpreter: /usr/bin/python2.7
    ## -- 以上内容请勿改动 --

## 为主机列表中的主机设置不同的角色。
children:
    ## db:数据库集群主机的分组
    ## cm:Cluster Manager集群管理主机的分组
    ## proxy:Proxy集群主机的分组
    db:
        hosts:
              ## db host可包含以下选项:

#【必填】polardb_polar_hostid, 保证主机索引唯一
        # external_storage_path:
        # polardb_proxy_aux_db_readonly:enable readonly for aux db instance
        # polardb_dma_node_type:logger or learner
            host01:

        ## 【必填】polardb_polar_hostid, 保证主机索引唯一
                polardb_polar_hostid: 1
            host02:

        ## 【必填】polardb_polar_hostid, 保证主机索引唯一
                polardb_polar_hostid: 2

            # host03:

        vars:
        ## polardb_custom_params为用户自定义参数。格式为: '<key> = <value>'。等号前后必须各有一个空格。value若为字符串类型，则必须使用单引号。
```

```
        ## 例如:polar_datadir = '/1739656-1/data'
        ## 如果无需配置自定义参数，可留空(polardb_custom_params: [])，或者使用注释符号将其屏蔽。
            # polardb_custom_params: []
            # - archive_mode = off
            # - archive_command = ''

    ## CM节点分组。将要安装CM服务的节点列在此处。当前仅支持配置为1个或3个节点作为CM节点。
    cm:
        ## 若此处CM节点分组包含三个主机，则会配置为三节点高可用模式。
        hosts:
            host01:
            host02:
            host03:

        var:
    ## proxy节点分组。将要安装proxy服务的节点列在此处。
    proxy:
      ## proxy节点分组中，建议包含至少两个节点以保障高可用及负载均衡。
        hosts:
            host02:
            host03:
        var:
```

创建数据库集群和组件

```
pdbcli create cluster
```

- 完整的 `pdbcli` 使用帮助, 可见附录 `pdbcli使用指南`

数据库使用

License授权（购买后申请）

仅有数据库内核需要授权, 授权方式如 `单节点形态` 所示，但需要每台机器都单独申请授权并启用；

数据库访问

数据库提供代理节点 `Proxy` , 其作为建议的接入层, 可实现对数据库failover自动恢复的能力、以及提供一写多读的能力.

连接方式

使用jdbc可以使用如下连接串, 可将多个Proxy的endpoint均配置其中, 确保单个Proxy挂掉客户端访问不受影响:

```
jdbc:postgresql://proxy1_host:proxy1_port,proxy2_host:proxy2_port,proxy3_host:proxy3_port/database
```

访问方式

Proxy在提供读写分离能力的基础上, 提供了两种不同的一致性:

- `强一致性` : 所有请求均路由到数据库RW节点
- `会话一致性` : 支持读路由到Standby节点, 单个连接内根据位点保证一致性, 不同连接不作保证, 可充分利用Standby的资源提供只读的能力.

两种不同的一致性需求分别对应两个endpoint.

系统对接方案

高可用

`ClusterManager` 作为数据库集群的高可用组件, 在主节点出现宕机或者挂掉时, 可即时选择可用备节点进行切换, 亦可手动进行切换。高可用组件本身也使用三节点形态部署, 其利用raft协议保障自身的高可用。

Failover

- 正常情况下, 高可用组件 `ClusterManager` 会定期对数据库进行3s一次的探活操作, 针对RW节点进行可写探活, 针对Standby节点进行只读探活.
- 一旦探活出现异常, 在探活超时的场景下, 会结合 `Monitor Agent` 采集的数据, 判断是否因为CPU打满所致, 如果是则不会优先尝试切换.
- 如果是其它探活失败的场景, 连续三次失败则会优先考虑切换操作， 选择其中一个处于同步状态的 `Standby` 提升为 `RW` 节点

Switchover

使用pdbcli进行手动的切换, 先列出所有可切换的Standby节点

```
pdbcli failover --list
Cluster Status:
{
  "phase": "RunningPhase",
  "master":
  {
    "endpoint": "10.0.0.1:1521",
    "phase": "RUNNING",
    "start_at": "2020-09-24 23:32:07",
    "sync_status": "SYNC"
  },
  "standby":
  [
    {
      "endpoint": "10.0.0.2:1521",
```

```
      "phase": "RUNNING",
      "start_at": "2020-09-24 23:32:10",
      "sync_status": "SYNC"
    },
    {
      "endpoint": "10.0.0.3:1521",
      "phase": "RUNNING",
      "start_at": "2020-09-24 23:32:10",
      "sync_status": "SYNC"
    }
  ]
}
```


之后可选择Standby的endpoint进行切换

复制代码

```
$ pdbcli failover --target 10.0.0.2
```

监控

prometheus对接

PolarFlex已经提供exporter, 监听9974端口, `prometheus` 中可进行如下配置, 完整的监控指标可见附件 `PolarFlex监控组件用户手册`

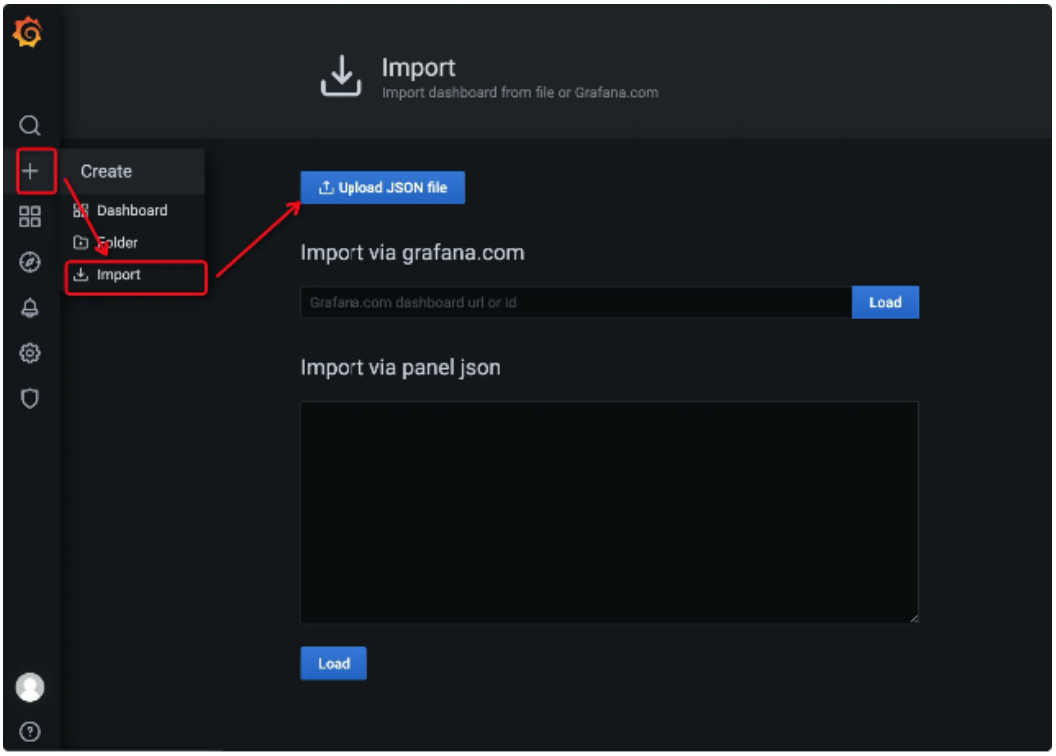
复制代码

```
scrape_configs:
- job_name: 'polardb_o'

  static_configs:
  - targets: ['0.0.0.0:9974'] # UE默认监听9974作为exporter端口

  honor_labels: true

  scrape_interval: 20s
  scrape_timeout: 20s
```

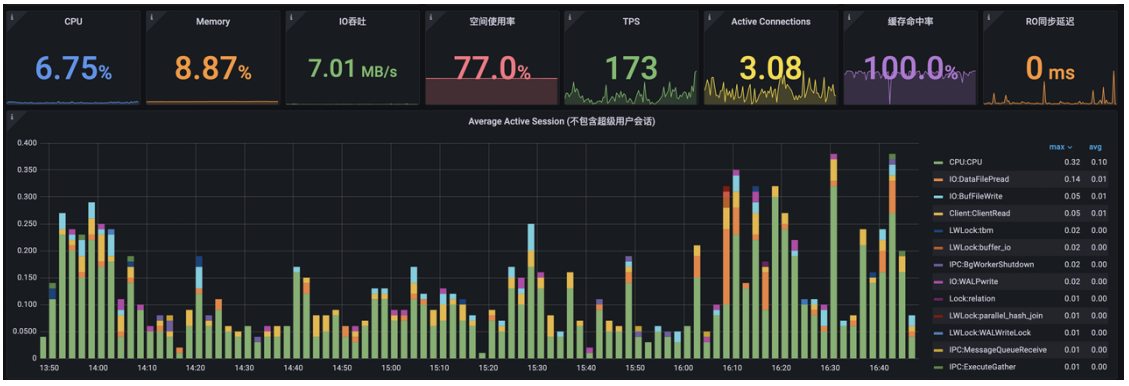
grafana dashboard

polardb_o.json
121.9KB

- 在grafana中可使用如下方式进行导入



- 样例效果如下:



告警配置指南

建议的告警指标、验证方式及处理方式如下:

告警指标	指标说明	单位	告警阈值	测试验证方式	建议处理方式
polar_cpu_user + polar_cpu_sys	cpu使用率	百分比	>= 70	1. 执行以下所有测试前需要先执行 pgbench -i -s 1000 2. 给数据库压力, $CONNECTION设置为CPU核数的2倍 pgbench -c $CONNECTION -j	

				$CONNECTION -M prepared -n -r -P 1 -T 10000	
polar_cgroup_mem_usage	内存使用率	百分比	>= 80	1. $CONNECTION设置为128 pgbench -c $CONNECTION -j $CONNECTION -M prepared -n -r -P 1 -T 10000 2. 告警阈值建议调整至5%	
polar_size_usage	磁盘空间使用率	百分比	>= 80	1. 执行pgbench -i -s 10000 2. 告警阈值调至10%	
polar_inodes_usage	文件系统inode使用率	百分比	>= 80	1. 同上 2. 告警阈值调至5%	
polar_connection_ratio	连接使用率	百分比	>= 90	1. psql连入数据库, SHOW polar_max_non_super_conns; 该值为连接使用率的分母 2. $CONNECTION设置为 polar_max_non_super_conns - 1 pgbench -c $CONNECTION -j $CONNECTION -M prepared -n -r -P 1 -T 10000	
polar_db_age	数据库年龄	xids	>= 1573741824	1. psql连入数据库 a. CREATE TABLE test_swell_time(id int);	查看是否存在长事务

				b. BEGIN; c. INSERT INTO test_swell_time VALUES(1); 2. 同时给数据库压力 pgbench -c $CONNECTION -j $CONNECTION -M prepared -n -r -P 1 -T 10000 2. 告警阈值调整至128 * 1000	
polar_inactive_slots	非活跃复制槽	个	> 0	不建议验证	确定slots是否有用, 无用则尽快删除
polar_replay_latency_in_mb	最大同步延迟	MB	>= 128	不建议验证	联系polardb同学
polar_active_connections	活跃连接	个	>= CPU核数 * 5	1. $CONNECTION调整至CPU核数*5 pgbench -c $CONNECTION -j $CONNECTION -M prepared -n -r -P 1 -T 10000	降低并发

polar_swell_time	当前最长事务持续时间	秒	>= 600	1. psql连入数据库
				a. CREATE TABLE test_swell_time(id int);
				b. BEGIN;
				c. INSERT INTO test_swell_time VALUES(1);
				
				2. 告警阈值调整至30s, 30s后即应开始报警
					尽快结束事务

polar_checkpoint_latency_in_mb	checkpoint延迟	MB	>=128	不建议验证	联系polardb同学

备份

详细可见附件 `PolarFlex备份组件用户手册`

附录

完整文档如下（购买后可通过阿里云销售同学或者云市场售前钉钉索取完整文档）：

pdbcli使用指南

阿里云 PolarDB 统一运维操作入口pdbcli使用指南 20240708.pdf
1.1MB

PolarFlex监控组件用户手册

PolarFlex监控组件用户手册.pdf
5.8MB

PolarFlex备份组件用户手册

PolarDB-O纯软版本备份工具操作指南（PolarDB Flex-2.2-20240715）.pdf
1.3MB