中国PostgreSQL培训认证课程之
服务管理

主要内容

1. 自动服务管理
2. 命令行服务管理
3. 集群服务配置管理

学习目标

1. 掌握自动服务管理、命令行服务管理
2. 熟悉集群服务配置管理方式

相关概念

- 数据库服务（集簇服务\集群服务\实例）
  - 多数操作系统上，PostgreSQL 是作为一种服务（或者叫守护进程）安装的
  - 多个 PostgreSQL 服务可以运行于同一台物理服务器上
  - 它们的侦听端口不能重复，也不能共享同一个数据存储目录
  - 多数情况下，在一台物理服务器上只会安装一个服务

- DATABASE
  - 每个 PostgreSQL 服务可以包含多个独立的 database

自动服务管理（Linux）

- 发行版安装的数据库，在安装完成之后会自动添加数据库服务
- 编译安装的数据库添加自启动服务

  1）将“源码包/contrib/start-scripts”目录下的脚本重命名后放到/etc/init.d目录下

  2）通过下面的命令添加到服务中

  chkconfig --add ***ScriptName***

  相应的，可通过下述命令删除

  chkconfig --del ***ScriptName***

  注：在 Centos 7中 systemctl 是设置系统服务（service）的命令，它融合之前service和chkconfig的功能于一体。

命令行服务管理

- 通过pg_ctl命令来实现数据库的启动、关闭、加载管理
- 在线帮助 pg_ctl --help
- 主要选项
  - start：启动
  - stop：关闭
  - reload：重新加载
  - restart：重启

命令行服务管理

- pg_ctl stop关闭数据库时，可以通过“-m”参数定义关闭方式
  - smart：缺省模式，等待所有客户端断开
  - fast：　强制模式，对未断开的客户端进行回滚，类似oracle中的immediate选项
  - immediate：强制模式，但不回滚，重启时需要自动恢复

命令行服务管理

- 关于pg_ctl命令的几个常用参数解释如下：

  -D：数据库data目录的路径

  -w：等待启动或关闭完成，等待是关闭的默认选项，但不是启动的默认选项

  -t： 等待时间

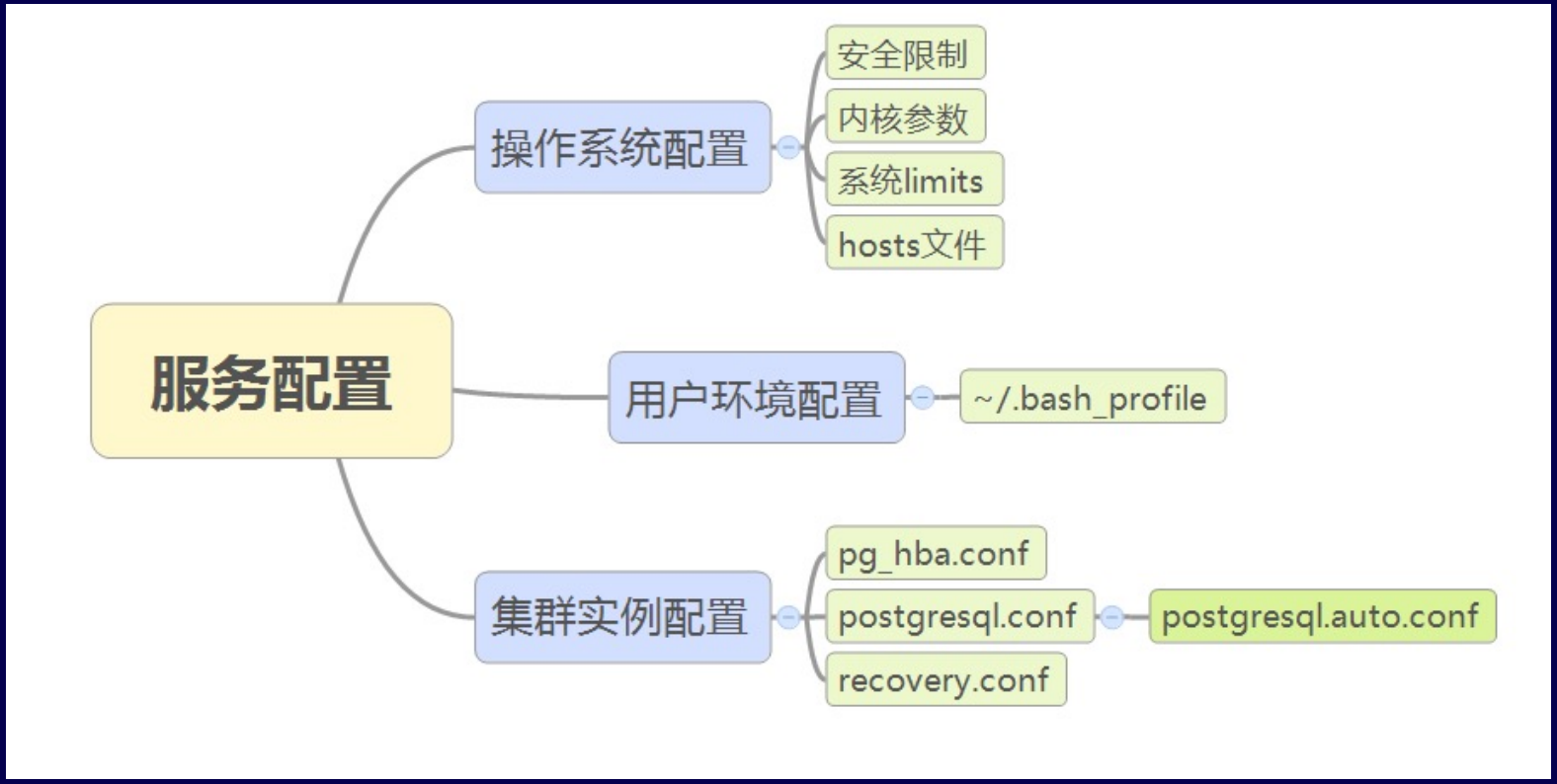
  -W：不等待命令成功完成

  -l：指定服务器日志记录文件

命令行服务管理

```
pg_ctl start   [-w] [-t SECS] [-D DATADIR] [-s] [-l FILENAME] [-o "OPTIONS"]
pg_ctl stop    [-W] [-t SECS] [-D DATADIR] [-s] [-m SHUTDOWN-MODE]
pg_ctl restart [-w] [-t SECS] [-D DATADIR] [-s] [-m SHUTDOWN-MODE] [-o "OPTIONS"]
pg_ctl reload  [-D DATADIR] [-s]
pg_ctl status  [-D DATADIR]
pg_ctl promote [-D DATADIR] [-s]
pg_ctl logrotate  [-D DATADIR] [-s]
pg_ctl kill  SIGNALNAME PID
```

服务配置


服务配置
操作系统配置
安全限制
内核参数
系统limits
hosts文件
用户环境配置
~/.bash_profile
集群实例配置
pg_hba.conf
postgresql.conf
postgresql.auto.conf
recovery.conf

操作系统配置

- Linux系统内核调整

```
vi /etc/sysctl.conf
---------------------------------
#add
kernel.shmmax = 68719476736（默认）          #定义了单个共享内存段的最大大小，以bytes为单位
kernel.shmall = 4294967296（默认）           #定义了共享内存页的总页数
kernel.shmmni = 4096                         #整个系统共享内存段的最大数目
kernel.sem = 50100 64128000 50100 1280       #每个信号对象集的最大信号对象数；系统范围内
                                             #最大信号对象数；每个信号对象支持的最大操作数；
                                             #系统范围内最大信号对象集数
fs.file-max = 7672460                        #文件句柄的最大数量。
                                             #文件句柄设置表示在linux系统中可以打开的文件数量
net.ipv4.ip_local_port_range = 9000 65000    #应用程序可使用的IPv4端口范围
net.core.rmem_default = 1048576              #套接字接收缓冲区大小的缺省值
net.core.wmem_default = 262144               #套接字发送缓冲区大小的缺省值
net.core.wmem_max = 1048576                  #套接字发送缓冲区大小的最大值
---------------------------------
sysctl -p                    #使配置生效
```

操作系统配置

- Linux资源限制调整

```
vi /etc/security/limits.conf
#add
* soft nofile 131072                    #当前系统生效的打开文件的数目

* hard nofile 131072                    #系统中所能设定的打开文件的最大值
* soft nproc  131072                    #当前系统生效的进程的数目

* hard nproc  131072                    #系统中所能设定的进程数目的最大值
* soft core   unlimited                 #当前系统生效的内核文件的大小
* hard core   unlimited                 #系统中所能设定的内核文件的最大值
* soft memlock 50000000                 #当前系统生效的锁定内存地址空间的大小

* hard memlock 50000000                 #系统中所能设定的的锁定内存地址空间最大值
```

操作系统配置

- 系统防火墙配置（或者直接关闭防火墙）

```
vi /etc/sysconfig/iptables-config
----------------------------------------
-A INPUT -i lo -j ACCEPT
#允许源IP
-A INPUT -s 192.168.0.0/16 -j ACCEPT

#允许源IP访问目标端口
-A INPUT -s 192.168.1.0/24 -m state --state NEW -m tcp -p tcp --dport 1922 -j ACCEPT

#允许任意IP访问目标端口
-A INPUT -p tcp -m state --state NEW -m tcp -p tcp --dport 5432 -j ACCEPT
```

用户环境配置（.bash_profile）

- 当安装完成后，用户如果想直接用命令行进行操作，需要进行环境变量的设置
- 否则直接执行psql、pg_ctl等操作，会提示命令无法找到
- 以下操作过程结束后，就可以直接使用psql连接数据库了

```
# su -postgres
vi  ~/.bash_profile
--------------------------
export PGHOME=/data/postgres/9.5.7
export PATH=$PGHOME/bin:$PATH
export PGDATA=$PGHOME/data
export LD_LIBRARY_PATH=$PGHOME/lib
--------------------------
source ~/.bash_profile
```

集群实例配置

文件名	主要作用
pg_hba.conf	客户端访问认证文件
postgresql.auto.conf	保存ALTER SYSTEM修改后的参数 不要手动修改它 优先级较高
postgresql.conf	主要配置文件
recovery.conf（12版本之前）	基于wal日志恢复的配置文件
pg_ident.conf	配置哪些操作系统用户可以映射为数据库用户

pg_hba.conf

- 客户端认证配置配置文件pg_hba.conf作用
  - 哪些主机可以连接数据库实例
  - 哪个数据库用户可以使用它
  - 允许这个用户使用哪些数据库
  - 客户端使用什么连接方式和认证方式

pg_hba.conf-格式

该配置文件有5个参数，分别为：
TYPE（主机类型）、DATABASE（数据库名）、USER（用户名）、ADDRESS（IP地址和掩码）、METHOD（加密方法）

# TYPE	DATABASE	USER	ADDRESS	METHOD
local	database	user	auth-method	[auth-options]
host	database	user	address	auth-method [auth-options]
hostssl	database	user	address	auth-method [auth-options]
hostnossl	database	user	address	auth-method [auth-options]
host	database	user	IP-address IP-mask	auth-method [auth-options]
hostssl	database	user	IP-address IP-mask	auth-method [auth-options]
hostnossl	database	user	IP-address IP-mask	auth-method [auth-options]

pg_hba.conf-主机类型TYPE

- local匹配使用Unix域套接字的连接
  - 如果没有TYPE为local的条目则不允许通过Unix域套接字连接
- host匹配使用 TCP/IP建立的连接，同时匹配SSL和非SSL连接
  - 缺省安装只监听本地环回地址localhost的连接，不允许使用TCP/IP远程连接
  - 启用远程连接需要修改postgresql.conf中的listen_addresses参数
- hostssl匹配必须是使用SSL的TCP/IP连接
  - 客户端和服务器端都安装OpenSSL
  - 编译PostgreSQL的时候指定configure参数--with-openssl打开SSL支持
  - 在postgresql.conf中配置ssl = on
- hostnossl只匹配使用非SSL的TCP/IP连接

pg_hba.conf-认证方式

- trust
  - 无条件地允许连接。
  - 允许任何可以与PostgreSQL数据库服务器连接的用户身份登入
  - 不需要口令或者其他任何认证。
- reject
  - 无条件拒绝连接。常用于从一个组中“过滤出”特定主机
  - 例如一个reject行可以阻塞特定的主机连接,而后面一行允许特定网络中的其余主机进行连接
- md5和password口令认证
  - md5认证方式为双重md5加密，password指明文密码
  - 不能在非信任网络使用password方式。
- peer
  - 从操作系统获得客户端的操作系统用户,并且检查它是否匹配被请求的数据库用户名
  - 只对本地连接可用
- 其它认证方式
  - https://www.postgresql.org/docs/current/static/auth-methods.html

pg_hba.conf-配置示例

vi $PGDATA/pg_hba.conf

```
# TYPE  DATABASE USER     ADDRESS  METHOD
  local      all   all                                trust    # 服务端本地用户可信登录
  host       replication replica  0.0.0.0/0     md5   # 流复制用户密码验证登录
  host       all postgres          0.0.0.0/0     reject   # 拒绝超级用户从网络登录
  host       all all               0.0.0.0/0     md5    # 其它用户密码验证登陆（不太安
  全）
```

local 备注:
Linux下进程通讯方式有很多,比较典型的有套接字,平时比较常用的套接字是基于TCP/IP协议的,适用于两台不同主机上两个进程间通信, 通信之前需要指定IP地址.
同一台主机上两个进程间通信用套接字,还需要指定ip地址,有点过于繁琐. 这个时候就需要用到UNIX Domain Socket, 简称UDS,

postgresql.conf

配置文件结构

#注释

key = value

支持的参数值类型

布尔、整数、浮点数、字符串、枚举

include指令（允许嵌套）

备注：
如果配置值中包含数字，则需要用单引号括起。
如果参数值本身包含单引号，我们可以写两个单引号(推荐方法)或用反斜扛包围

postgresql.conf-连接

listen_addresses = '*'　　#（关联配置文件pg_hba.conf）

#指定服务器在哪些 TCP/IP 地址上监听客户端连接。
值的形式是一个逗号分隔的主机名和/或数字 IP 地址列表。

特殊项*对应监听所有可用 IP 接口

0.0.0.0允许监听所有 IPv4 地址
::允许监听所有 IPv6 地址
如果列表为空，服务器将根本不会监听任何 IP 接口，在这种情况中只能使用 Unix 域套接字来连接它。
默认值是localhost，它只允许建立本地 TCP/IP "环回"连接。
这能帮助在不安全网络接口上阻止重复的恶意连接请求。
这个参数只能在服务器启动时设置。

拓展阅读： https://blog.csdn.net/pg_hgdb/article/details/78717974

postgresql.conf-连接

port = 5866

#服务器监听的 TCP 端口；默认是 5432 。
请注意服务器会同一个端口号监听所有的 IP 地址。
这个参数只能在服务器启动时设置。

max_connections = 100

#决定数据库的最大并发连接数。
默认值通常是 100 个连接，但是如果内核设置不支持（initdb时决定），可能会比这个 数少。
这个参数只能在服务器启动时设置。（ cpu 1核 50个链接 ）

postgresql.conf-内存

shared_buffers

- 它表示数据缓冲区中的数据块的个数，每个数据块的大小是8KB。
- 数据缓冲区位于数据库的共享内存中，它越大越好，不能小于128KB。
- 这个参数只有在启动数据库时，才能被设置。
- 默认值是128MB。
- 推荐值：1/4 主机物理内存

wal_buffers

- 用于还未写入磁盘的 WAL 数据的共享内存量。
- 默认值 -1 表示将该参数值设置为 shared_buffers 的 1/32 的大小（ 大约 3%），但是不小于64kB 也不大于一个WAL段的大小（通常为 16MB）。
- 如果自动的选择太大或太小可以手工设置该值，但是任何小于 32kB 的正值都将被当作 32kB。
- 这个参数只能在服务器启动时设置。
- 事务日志缓冲区位于数据库的共享内存中。
- 推荐值：min(2047MB, shared_buffers/32) = 512MB

postgresql.conf-内存

work_mem

- 指定在写到临时磁盘文件之前用于内部排序操作和哈希表的内存量。
- ORDER BY， DISTINCT 和合并连接（ merge joins ） 都会用到排序操作。
- 默认值为 4 兆字节（ 4MB ）。
- 推荐值：work_mem = (输入内存数量- shared_buffers)/(连接数 * 3 ）* 1024 (单位是KB);

maintenance_work_mem

- 它决定数据库的维护操作使用的内存空间的大小。
- 数据库的维护操作包括VACUUM、CREATE INDEX和ALTER TABLE ADD FOREIGN KEY等操作。
- 值如果比较大，通常可以缩短VACUUM数据库和从dump文件中恢复数据库需要的时间。
- maintenance_work_mem存放在每个数据库进程的私有内存中，而不是存放在数据库的共享内存中。这个参数可以在任何时候被设置。

postgresql.conf-优化建议

random_page_cost = 2.5

规划器对一次非顺序获取磁盘页面的代价估计。 默认值是 4.0。 高端存储或者ssd可以适当调小该参数。

autovacuum_max_workers = 10

指定能同时运行的autovacuum进程的最大数量， 适当调大， 避免vacuum不及时导致表膨胀

checkpoint_completion_target = 0.7

增加checkpoint_completion_target来降低检查点的I/O负载， 默认0.5

archive_timeout =1800

强制服务器来周期性地切换到一个新的 WAL 段文件

archive_command = 'test ! -f /paic/dba/pgbackup/${PGNAME}/archlog/%f && pxz -2 < %p >/paic/dba/pgbackup/${PGNAME}/archlog/%f'

pxz压缩归档日志， 64M的归档， 压缩时间可以在0.5s内， 压缩比一般可在1:3左右

配置参数级别

- 系统级别 /集群服务级别/实例级别（全局）
- 用户/角色级别
- 用户/角色 + 数据库级别
- 会话级别

参数设置级别	参数存储位置
cluster	postgresql.conf or postgresql.auto.conf
db	pg_db_role_setting
role	pg_db_role_setting
db 和 role的组合	pg_db_role_setting

配置修改-全局

- 以下几种方式可实现全局配置
  - 通过linux命令(vim,echo,sed)修改配置文件
  - 启动时设置（不推荐，除非进入单用户模式）
  - psql -c configparameter=newvalue
  - 通过ALTER SYSTEM命令修改全局配置（针对postgresql.auto.conf）

```
\h alter system
命令：      ALTER SYSTEM
描述：      更改服务器的配置参数

语法：
ALTER SYSTEM SET 配置参数 { TO | = } { 值 | '值' | DEFAULT }
ALTER SYSTEM RESET 配置参数
ALTER SYSTEM RESET ALL
```

配置修改-非全局

配置生效级别	修改方式
Database	ALTER DATABASE name SET configparameter { TO | = } { value | DEFAULT } ALTER DATABASE name RESET configuration
Session	通过SET命令设置当前Session的配置 SET configparameter { TO | = } { value | 'value' | DEFAULT } SET configparameter TO DEFAULT; 更新pg_settings视图 UPDATE pg_settings SET setting = new_value WHERE name = 'configparameter'; 使用set_config函数更新会话配置 SELECT set_config('configparameter',new_value,false);
user/Role	ALTER ROLE name [IN DATABASE database_name] SET configparameter { TO | = } { value | DEFAULT } 重置/取消这些参数的设置 ALTER ROLE name [IN DATABASE database_name] RESET configparameter ALTER ROLE name [IN DATABASE database_name] RESET ALL;

配置生效说明

select distinct(context),name from pg_settings;

Context（上下文）	生效方式
sihup	给服务器发送HUP信号会使服务器重新加载postgresql.conf配置，可以立即生效
postmaster	只有服务重启才能生效
internal	编译期间的设置，只有重新编译才能生效。
backend	与sighup类似，但是不影响正在运行的会话，只在新会话中生效
superuser	使用superuser(如postgres)才能更改，不用重新加载所有配置即可生效
user	单个会话用户可以在任意时间做修改，只会影响该会话

配置生效方法

- 使配置生效的几种方法

1）用超级用户运行

SELECT pg_reload_conf();

2）使用pg_ctl命令触发SIGHUP信号

pg_ctl reload

3）用UNIX的kill手动发起HUP信号

ps -ef|grep -i postmaster|grep -v grep|xargs kill -HUP

4）重启数据库服务

pg_ctl restart

备注：
主服务器进程每次收到 SIGHUP 信号后都会重新读取这个配置文件。
同时主服务器进程也将这个信号广播给所有正在运行的子服务器进程，这样现有会话也能得到新值。

配置查看方法

- 通过 SHOW 命令查看特定系统设置
  - SHOW 命令的输出结果会自动根据数值大小选择合适的单位的所有设置
  - SHOW ALL 命令，其输出结果会针对每个设置选用合适的单位

```
--配置查看示例
show all;                    #查看所有数据库参数的值
show shared_buffers;   #查看某个参数的当前值(可查看当前会话值)
select current_setting('shared_buffers');
```

配置查看方法

- 查询 pg_settings 视图可以很方便地检查当前设置
  - 它本质上是SHOW和SET命令的可替换接口
  - 它还提供了SHOW不能提供的关于每一个参数的一些实现，例如最大值和最小值

```
--配置查看参考SELECT name, setting FROM pg_settings WHERE category = 'File Locations';
select name,context,unit,boot_val,setting,reset_val from pg_settings where name
in('listen_addresses','max_connections',
'shared_buffers','effective_cache_size','work_mem','maintenance_work_mem')
order by context, name;
SELECT name,setting FROM pg_settings where name ~  'xxx' ;
SELECT current_setting(name);
```

配置查看方法

pg_settings参数查看示例：

```
postgres=# select name,context,unit,boot_val,reset_val from pg_settings where name in('listen_addresses','max_c
onnections', 'shared_buffers','effective_cache_size','work_mem','maintenance_work_mem')
postgres-# ORDER BY context, name;
         name          |  context   | unit | boot_val  | reset_val
-----------------------+------------+------+-----------+-----------
 listen_addresses      | postmaster |      | localhost | *
 max_connections       | postmaster |      | 100       | 100
 shared_buffers        | postmaster | 8kB  | 1024      | 16384
 effective_cache_size  | user       | 8kB  | 524288    | 524288
 maintenance_work_mem  | user       | kB   | 65536     | 65536
 work_mem              | user       | kB   | 4096      | 4096
(6 rows)
```

备注：
setting 是指当前设置；boot_val 是指默认设置；reset_val 是指重新启动服务器或重新加载设置之后的新设置。
在 postgresql.conf 中修改了设置后，一定要记得查看一下 setting 和 reset_val 并确保二者是一致的，否则说明设置并未生效，需要重新启动服务器或者重新加载设置。

配置示例-用户角色级修改参数

```
--查询某参数在某用户级别的设置
create user test password 'test';
alter role test set log_min_duration_statement = 100;

--方法一：查询pg_user表
select * from pg_user where usename='test';

--方法二：查询pg_db_role_setting表
select * from pg_db_role_setting where setrole in (select usesysid from pg_user where usename in
('test'))
 order by setrole,setdatabase;

--参数在某用户下针对数据库级别的设置
alter role test in database postgres set client_min_messages='warning';

select * from pg_db_role_setting where setrole in (select usesysid from pg_user where usename in
('test')) order by setrole,setdatabase;
```

配置示例-postgresql.auto.conf

```
#运行日志相关配置变更
alter system set logging_collector = on;
alter system set log_destination = 'csvlog';
alter system set log_directory = 'log';
alter system set log_filename ='postgresql-%Y-%m-%d_%H%M%S.log';
alter system set log_rotation_age = '1d';                    #每天生成一个新的日志文件
alter system set log_rotation_size = 0;                      #不限制单个日志文件大小
alter system set log_truncate_on_rotation = on;              #覆盖同名文件
                                                             #只保留7天日志，循环覆盖
alter system set log_hostname = on;
alter system set log_line_prefix = '%m';          #控制每条日志信息的前缀格式，默认值是空串
alter system set log_statement = 'ddl';
```

配置示例-postgresql.auto.conf

```
#WAL和归档参数变更
alter system set wal_level = archive;
alter system set archive_mode = on;
alter system set archive_command = ''test ! -f /mnt/server/archivedir/%f && cp
%p /mnt/server/archivedir/%f'';
```

建议：此处/ssd/archive建议将归档日志放到数据库的定时备份的路径内，便于数据库异机恢复时最大可能的恢复数据

```
#参数修改完成后重新启动数据库
pg_ctl -m fast stop
pg_ctl start
```

学习要点

1. 自动服务主要是解决数据库服务器重启后不用手动去启动数据库；
2. 通常在生产环境中会涉及更为复杂的设置，主要是指借助高可用集群软件来管理数据库服务的启停；
3. 手动服务管理主要是根据运维事件的需要进行人工的数据库服务控制。
4. 服务配置面向操作系统的资源管理配置和集群服务的初始化配置，主要通过手动编辑配置文件来完成；
5. 配置不是一成不变的，特别是集群服务的配置或根据问题处理或者性能调优的目的做出相应的修改；
6. 配置的生效方式有所不同，同时涉及到配置的管控范围。

学习作业

1.实现服务的自启动;

2.使用命令pg_ctl对服务进行手动管控;

3.根据本机资源情况面向主要的配置参数进行修改，并检查其是否生效(修改服务端口).


CHINA
POSTGRESQL
ASSOCIATION
THANKS
PostgreSQL是世界上最强大开源对象关系数据库