GOTC

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

OPEN SOURCE , OPEN WORLD

分布式数据库与存储

老树发新芽 —— 开箱即用的开源PostgreSQL发行版 Pigsty

冯若航 2021年08月01日

About Me

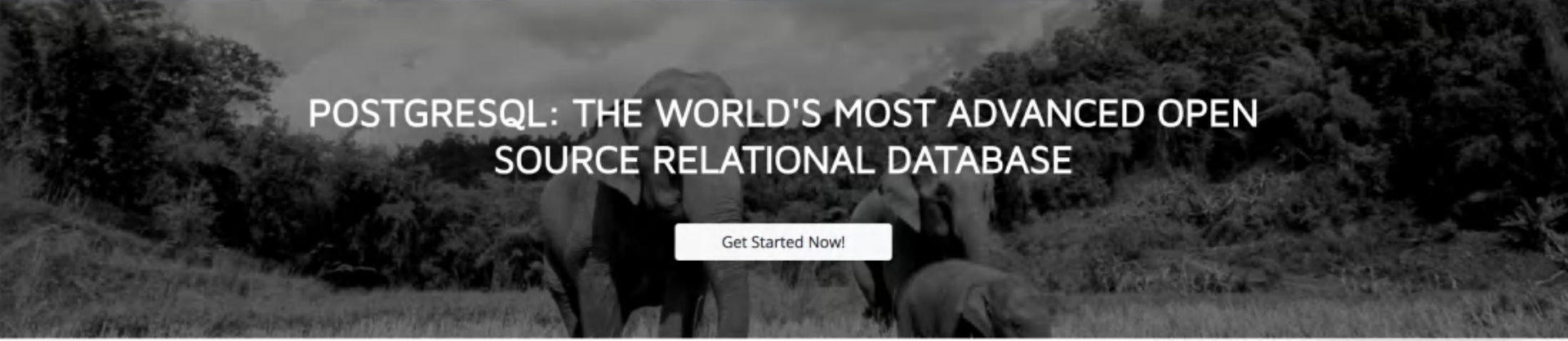
Ruohang Feng (Vonng)

https://vonng.com

rh@vonng.com

独立开源贡献者，Pigsty作者，《Designing Data-Intensive Application》《PG Internal》译者
PostgreSQL中国社区开源技术委员会委员，PostgreSQL专家，全栈开发者，曾任职于Alibaba，Apple，TanTan

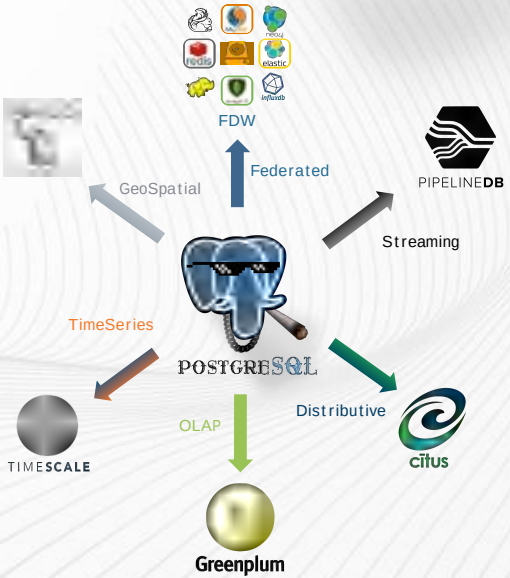
- http://demo.pigsty.cc



世界上最先进的开源关系型数据库

- 性能优异，功能强大
- 扩展丰富，协议友善
- 历史悠久，稳定可靠
- 代码清晰，定制灵活

一招鲜，吃遍天



一专多长的全栈数据库

成熟的应用可能会用到许许多多的数据组件：
缓存，OLTP，OLAP/批处理/数据仓库，流处理/消息队列，搜索索引，NoSQL/文档数据库，地理数据库，空间数据库，时序数据库，图数据库。

传统的架构选型可能会组合使用多种组件，典型如：Redis + MySQL + Greenplum/Hadoop + Kafka/Flink + ElasticSearch，
一套组合拳基本能应付大多数需求。

带来的新问题就是 异构系统集成

需编写大量重复繁琐搬砖代码，
把数据从A组件搬运到B组件。

“开源的Oracle”

去O扛旗者

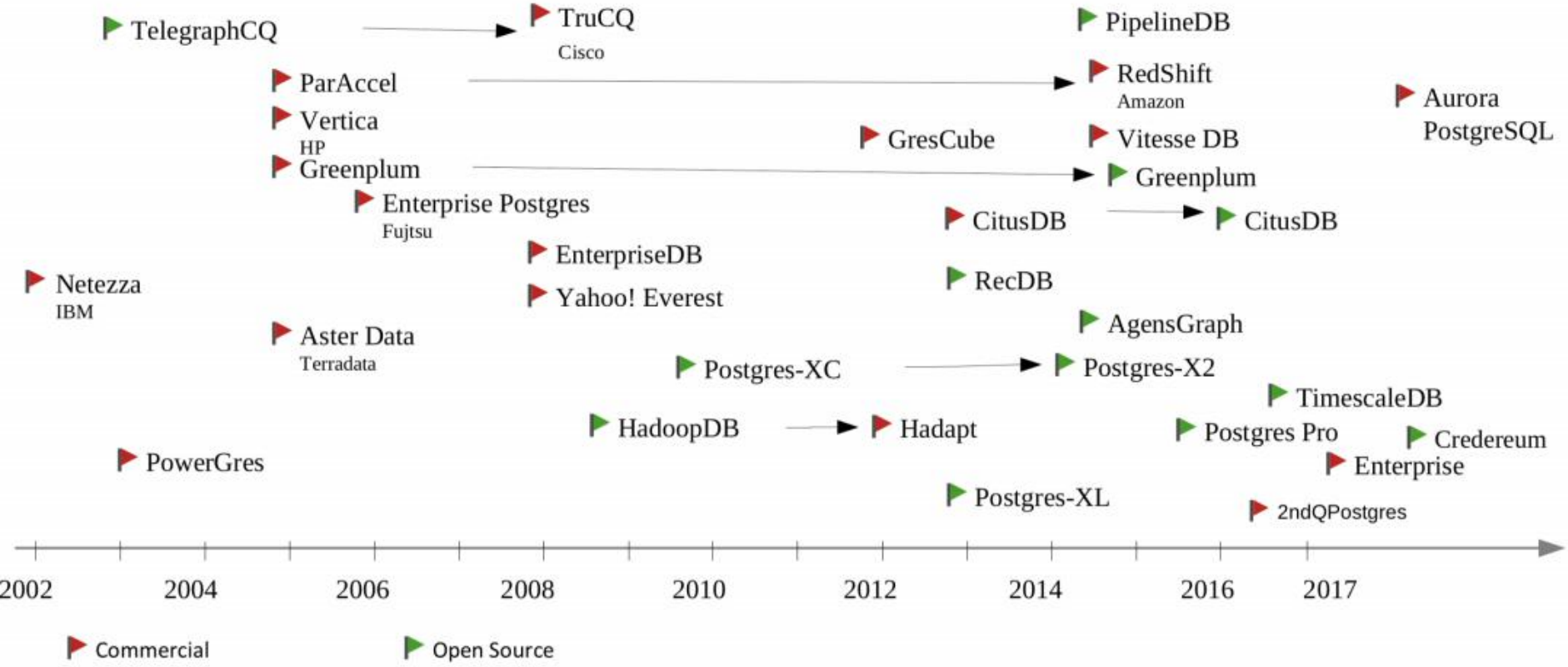
原生支持Oracle 八九成功能

唯一能对Oracle构成直接威胁的开源数据库

开放扩展的数据库

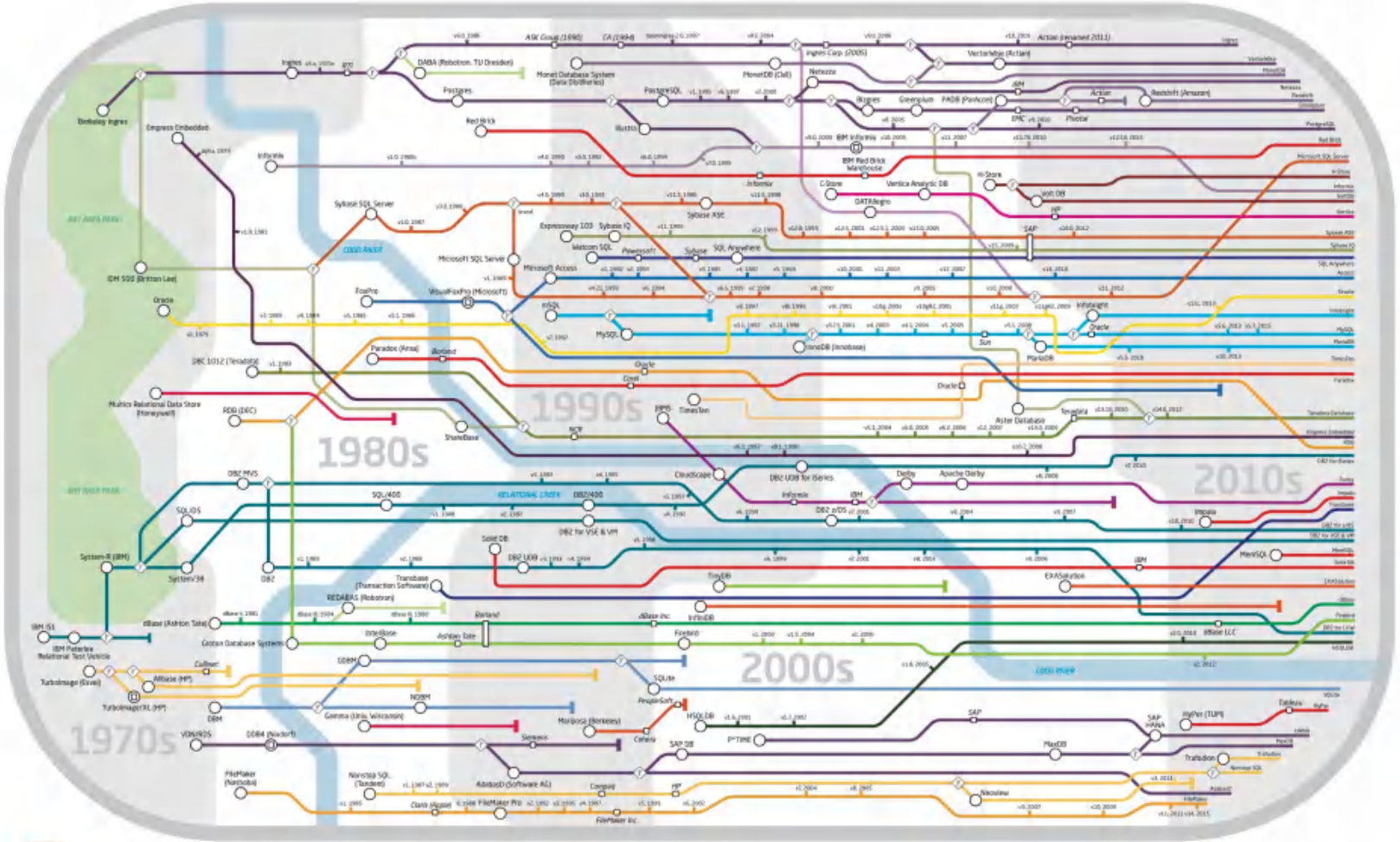
健康的社区，友善的协议，丰富的扩展，繁荣的生态

开支散叶，子孙满堂

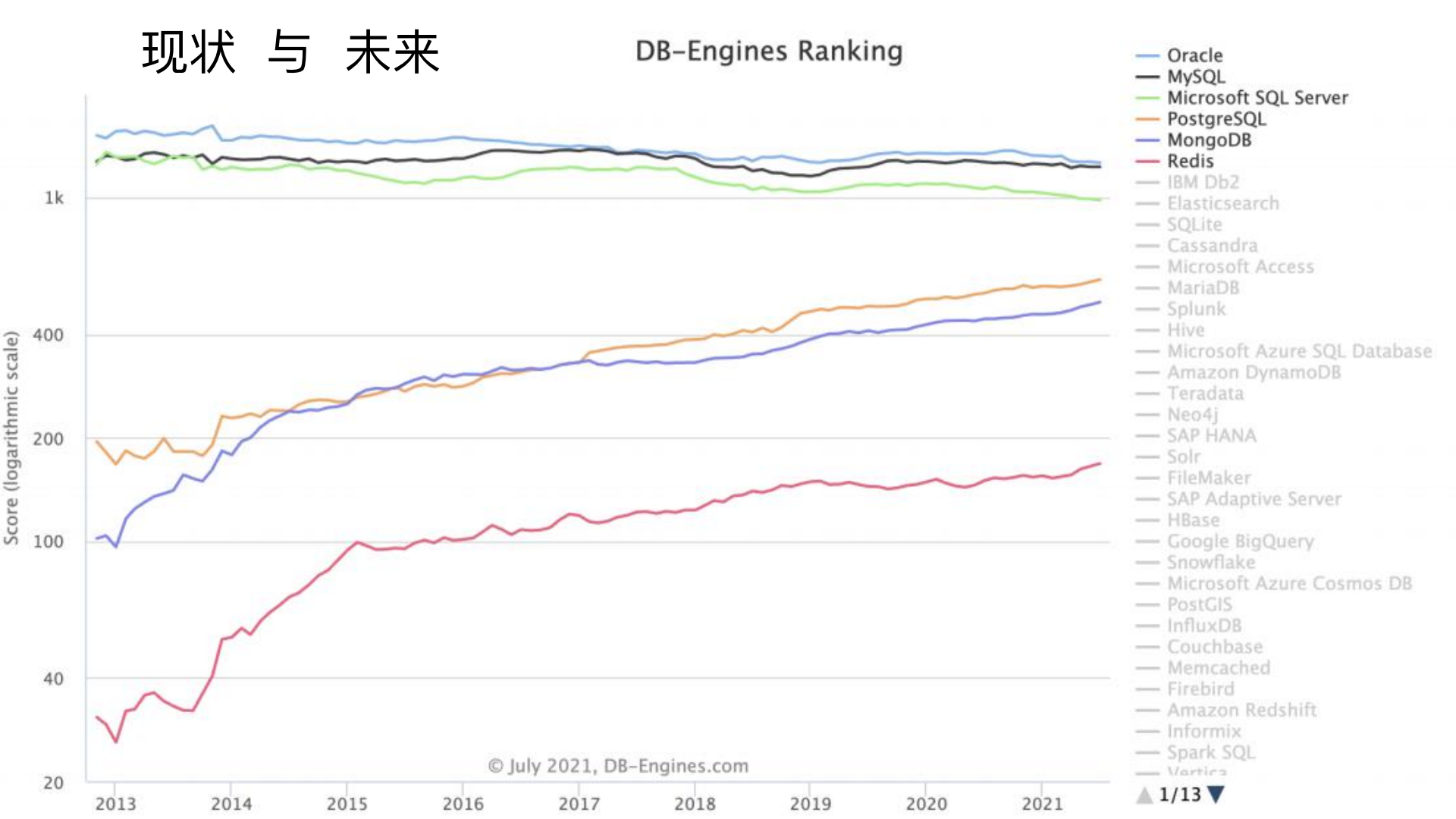


老树

祖宗级开源项目


Genealogy of Relational Database Management Systems
1970s
1980s
1990s
2000s
2010s

现状 与 未来


DB-Engines Ranking
Score (logarithmic scale)
1k
400
200
100
40
20
2013
2014
2015
2016
2017
2018
2019
2020
2021
© July 2021, DB-Engines.com
Oracle
MySQL
Microsoft SQL Server
PostgreSQL
MongoDB
Redis
IBM Db2
Elasticsearch
SQLite
Cassandra
Microsoft Access
MariaDB
Splunk
Hive
Microsoft Azure SQL Database
Amazon DynamoDB
Teradata
Neo4j
SAP HANA
Solr
FileMaker
SAP Adaptive Server
HBase
Google BigQuery
Snowflake
Microsoft Azure Cosmos DB
PostGIS
InfluxDB
Couchbase
Memcached
Firebird
Amazon Redshift
Informix
Spark SQL
1/13

Pigsty v1.0.0

Battery-Included Open Source PostgreSQL Distribution

Postgres in Graphic STYle

Ruohang Feng

Outline

What Where Who Why How

Monitoring

Provisioning

High Available

Sandbox

Analysis

Open Source


Lord of PostgreSQL
Fellowship of Pigsty

What is Pigsty

The Battery-Included Open Source PostgreSQL Distribution

PostgreSQL 数据库 发行版

开箱即用的开源PostgreSQL数据库发行版

打包最新PG内核：PostgreSQL 13.3/14-beta2，以及海量扩展插件（强力扩展：时序数据TimescaleDB 2.3，地理空间PostGIS 3.1，分布式Citus 10），全部开箱即用。
数据部署时带有可靠的中间件： Patroni 2.1, Pgbouncer1.15, Haproxy 2.2, Consul 1.10
打包有生产级的运行时环境： Grafana 8.0, Prometheus 2.28, Nginx 1.20, Ansible
无需互联网访问，在全新CentOS 7机器上一行命令10分钟完成所有安装流程，所有组件已为生产使用做好准备（不计Pigsty下载，约1GB）

监控系统 解决方案

全面专业的开源PostgreSQL监控解决方案

1000+监控指标 描述了从数据库，中间件，机器节点 以及负载均衡器的方方面面，从最顶层的全局概览到最小的单个数据库对象（表、索引、函数等）都一览无余。
带有了20余个精心设计的监控面板，将监控指标转化为多层次全方位的分析与洞察。所有图表都带有丰富的导航连接供下钻、上卷、横跳，可快速定位故障与性能问题
内置了基于Prometheus+AlertManager与Grafana的报警系统，两者等效且互为备份，预置常用数据库与机器报警规则，开箱即用。
可选装基于loki与promtail的实时日志收集方案，并提供用于分析PG CSV日志样本的应用 pglog

部署管控 解决方案

简单易用的开源PostgreSQL管控解决方案

声明式配置文件允许用户通过160+参数描述与定制数据库与基础设施运行时的方方面面，而无需操心具体的实现，整体提供类似Kubernetes的云原生使用体验。
自愈的高可用数据库集群 Pigsty部署的数据库集群可以自动从常见故障中恢复，集群成员对外表现等价，只要仍有任意实例存活，集群对外提供的各种服务就不会停止。
提供了一系列内置幂等剧本 用于部署、扩容、销毁PostgreSQL数据库集群与实例，管理业务数据库与用户，使用简单方便。提供自带的CMDB供扩展与定制，提供可选的CLI与GUI
内置服务发现机制。除用于提供高可用的DCS外，数据库与基础设施采用松耦合模式。数据库变动（扩缩容，上下线）自动适配，无需手工维护基础设施与周边系统。

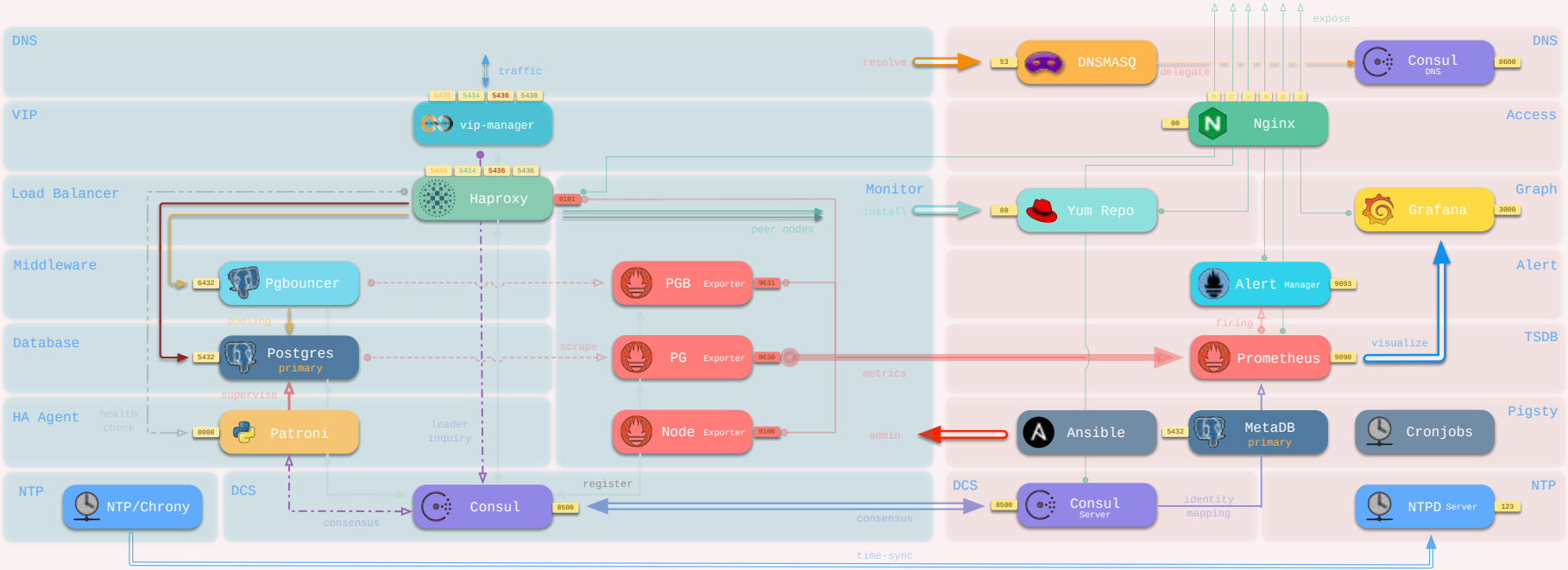
便捷全能 沙箱环境

一键拉起的全能PostgreSQL本地沙箱环境

Pigsty针对大规模生产环境的PostgreSQL管理而设计，但亦设计为可完整运行于本地1核1GB虚拟机中，便于保持开发、测试、预发、生产环境的一致性。
用户在自己的笔记本(Macbook)上可一键拉起功能完整的本地沙箱（由vagrant与virtualbox驱动），或通过简单的配置在自己的虚拟机或云ECS/VPS上完成部署。
用户可以在与实际生产环境（除规格外）完全一致的沙箱环境中学习探索PostgreSQL，进行各种实验，仿真，测试，开发，并从生产级监控系统中获取全面且实时的数据反馈。
用户可以使用内置的Postgres & Grafana & Echarts开发交互性数据应用与数据可视化作品。快速产出作品原型，并以标准的方式、在确定性的环境中分享，演示与交付。

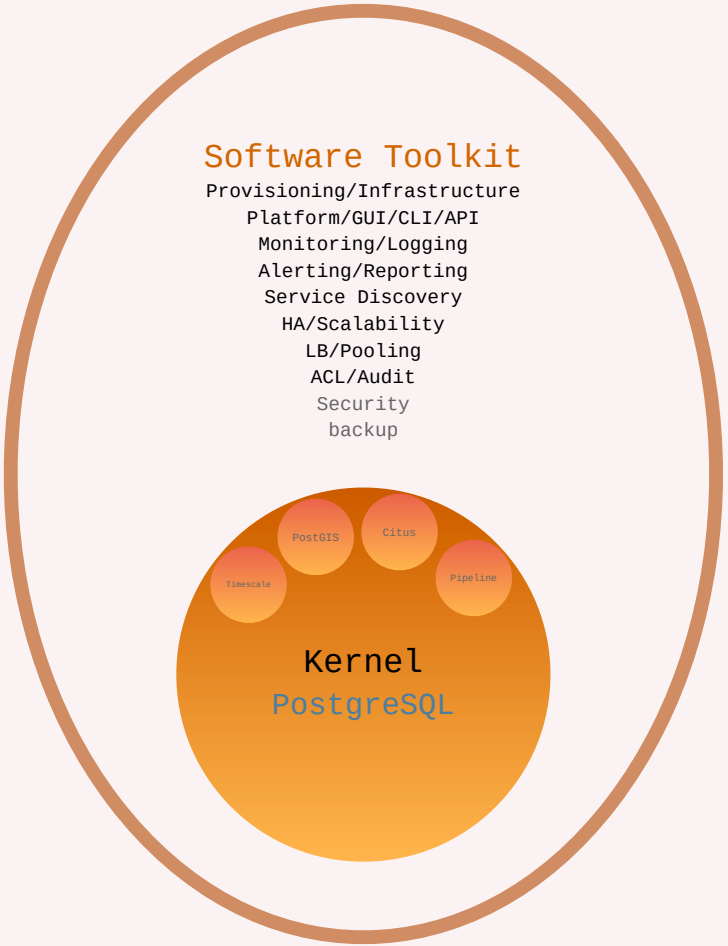
开箱即用

Pigsty将集群部署，扩容缩容，主从复制，故障切换，流量代理，连接池，服务发现，访问控制，监控系统，告警系统，日志采集等生产级成熟解决方案封装为发行版，一次性解决生产环境、个人使用PostgreSQL数据库的各类问题，真正做到 开箱即用。




pgsql
3.23%
21.2%
OK
v7新UI

发行版


Software Toolkit
Provisioning/Infrastructure
Platform/GUI/CLI/API
Monitoring/Logging
Alerting/Reporting
Service Discovery
HA/Scalability
LB/Pooling
ACL/Audit
Security
backup
PostGIS
Citus
Timescale
Pipeline
Kernel
PostgreSQL

Who wants Pigsty

互联网企业

中小型互联网企业 （日活千万量级，QPS百万量级，数据库集群数百量级）

体量可观：实例动辄数百个，QPS动辄数百万。
追求敏捷：集群实例变动频繁，对敏捷交付与快速伸缩有需求。
追求可用，对可用性有高要求，故障自愈，自动切换。
追求技术，希望通过数据驱动提高治理水平，偏爱开源。

传统企业

有信息化需求的各类传统企业

不想折腾，简单好用快速上手，易维护。
追求功能，地理,空间,时序,JSON,存储过程,物化视图,倒排索引等。
追求可靠，数据不丢不错不乱，备份容灾，有兜底。
自主可控，不想上云，但又想要类似云的管理体验

软件厂商

软件集成商，软件厂商

需要人才：用到PostgreSQL，却没有相关人才储备。
需求方案：需要成熟、完整的解决方案，比 `$ yum instsall && systemctl start` 更靠谱
追求成本：用最少的成本办最多的事

个人用户

DBA，OPS，DEV，数据研发，数据分析师，爱好者，学生

上手快： 一键拉起，配套齐全，开箱即用
门槛低： 1核1GB的底配虚拟机也可以跑起来。
一致性： 本地开发测试环境与生产环境保持一致
全功能：希望有数据处理与分析的整合环境： Python + SQL + 可视化

Where to run Pigsty

生产环境

管理节点：3-5　　64核 400GB PCI-E SSD x n

使用高规格物理机/虚拟机部署，使用3-5个管理节点控制监管几百套数据库集群，实际服务于线上业务。
一套Pigsty部署支持几百套数据库集群，几百万QPS，上千万日活不在话下。

测试/开发仿真环境

管理节点：1-3　　8核 16B SSD x n

与生产环境保持一致的Staging/UAT/预发环境
搭建公用的开发、测试环境，仿真线上环境

云虚拟机演示环境

管理节点：1　　2核 4GB x 1～4

使用云服务器部署，分享公开演示程序，快速服务小微数据业务。

个人电脑沙箱环境

管理节点：1　　1核 2GB x 1

运行于个人笔记本&台式机，Mac & PC 虚拟机中。
用于搭建Devbox，测试，实验，学习PostgreSQL
制作数据应用，交互式数据可视化应用

痛点！痒点！爽点！

功能?
ACID事务
地理空间?
时间序列?
全文检索?
存储过程?
JSON/XML
JOIN
可扩展性
扩展性

实施?
环境?
安装?
平台?
演示
部署?
管理?
维护?
自动化?

可靠性
高可用?
监控系统?
告警系统
可观测性
主从延迟
负载均衡
中间件
接入

伸缩性
分区分片
高并发
性能?
调度?
弹性伸缩?
参数调优
HTAP
读写分离
ETL
CDC

SQL变更?
数据迁移?
扩容缩容
用户管理
软件升级
插件管理
配置管理
恢复方案
备份方案
故障预案

易用性?
复杂度?
机器成本?
软件成本?
人力成本?
学习成本?
数据安全
风险?
审计
日志
访问控制
权限管理

生态
费用

是否开源
社区活力
协议
厂商支持
厂商锁定
项目生命力
人才储备
技能通用性
技能市场价值
技能贬值速度
投资人认可度?
炒作价值?
吹嘘价值?
审美?
谈资?

痛点?	从无到有	发行版，部署，解决方案
痒点?	从有到优	监控系统，高可用架构
爽点?	从优到易	开箱即用，全能沙箱
不可替代性?	从易到廉	开源免费

Installation Wizard

安装

Someone told me that each equation I included in the book would halve the sales.

— Stephen Hawking

有人告诉我，安装脚本里每多一行命令，用户数量就会减半

How to get Pigsty

Get pigsty with three lines of code

准备

内核：Linux
架构：x86_64
版本：CentOS 7.8.2003 （或其他RHEL7等效版本）
用户：root （或免密码sudo用户）
规格：1核2G （推荐使用2核4G以上规格）

```
$ echo "$(uname -s) $(uname -m) $(cat /etc/redhat-release)"
Linux x86_64 CentOS Linux release 7.8.2003 (Core)
```

确认无误后即可开始下载配置安装

下载

源代码 （必选项）
pigsty.tgz ---下载解压至---> ~/pigsty

软件包 （可选项）
pkg.tgz ----------> /tmp/pkg.tgz

```
$ cd ~
$ git clone https://github.com/Vonng/pigsty
```

或使用curl下载

```
$ curl -SL https://github.com/Vonng/pigsty/releases/download/v1.0.0-beta1/pigsty.tgz | gzip -d | tar -xC ~
```

下载可选离线软件安装包，可加速安装，目标节点无互联网访问时为必选。离线软件包也可以在configure过程中下载

```
$ curl -SLO https://github.com/Vonng/pigsty/releases/download/v1.0.0-beta/pkg.tgz -o /tmp/pkg.tgz
```

配置

使用何种配置样板？（通过-m参数指定）
demo （沙箱默认，单节点），demo4 （4节点沙箱） , pg14 , tiny （常规默认），oltp ,…
是否下载软件包？ （如果/tmp/pkg.tgz不存在则会询问）
Y：现在从Github上下载pkg.tgz
N：后续安装时从原始上游源下载 （如果操作系统非CentOS 7.8请使用此项）
主要的IP地址？ （如果发现多块网卡与多个可用IP地址会询问）
输入：当前节点（管理节点）使用的IP地址
（访问该节点时使用的IP地址，但不要使用公网IP）

```
$ cd ~/pigsty
$ ./configure [-n|--non-interactive]        # 如果您已经确定了三个参数选项，可以使用 -n 跳过交互式向导
              [-m|--mode <config_mode>]     # 指定需要使用的配置文件模板：demo, demo4, pg14, tiny, oltp等
              [-d|--download]               # 如果/tmp/pkg.tgz不存在，就从Github下载离线安装包
              [-i|--ip <primary_ip>]        # 存在多个IP地址时，指定其中一个作为首要IP地址
```

安装

访问该节点的3000端口即可浏览Pigsty监控首页

```
http://10.10.10.10:3000              # 替换为您自己的IP地址
默认用户名与密码：admin:pigsty

# 使用域名将带来更好的使用体验，例如：http://g.pigsty
执行 sudo make dns 会将Pigsty相关域名写入/etc/hosts
10.10.10.10  meta pigsty g.pigsty
```

```
$ make install
```

Ansible剧本infra.yml将在meta分组上执行以完成安装

ansible将在configure过程中从离线软件包或本地可用的其他源中安装

Installation

download → configure → install


```
[vagrant@meta ~]$ curl -fsSL https://pigsty.cc/pigsty.tgz | gzip -d | tar -xC ~; cd ~/pigsty  # DOWNLOAD
[vagrant@meta pigsty]$ make config
./configure
configure pigsty begin
[ OK ] kernel = Linux
[ OK ] machine = x86_64
[ OK ] release = 7.8.2003 , perfect
[ OK ] sudo = vagrant ok
[ OK ] ssh = vagrant@127.0.0.1 ok
[WARN] Multiple IP address candidates found:
    (1) 10.0.2.15          inet 10.0.2.15/24 brd 10.0.2.255 scope global noprefixroute dynamic eth0
    (2) 10.10.10.10        inet 10.10.10.10/24 brd 10.10.10.255 scope global noprefixroute eth1
[ OK ] primary_ip = 10.10.10.10 (from demo)
[ OK ] admin = vagrant@10.10.10.10 ok
[ OK ] mode = demo (vagrant demo)
[ OK ] config = demo@10.10.10.10
[ IN ] Cache /tmp/pkg.tgz not exists, download? (y/n):
=> y
[ OK ] cache = download from internet
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 1031M  100 1031M    0     0  3338k      0  0:05:16  0:05:16 --:--:-- 7061k
[ OK ] repo = extract from /tmp/pkg.tgz
[ OK ] repo file = /etc/yum.repos.d/pigsty-local.repo
[ OK ] utils = install from local file repo
[ OK ] ansible : ansible 2.9.18
configure pigsty end. Use 'make install' to proceed
[vagrant@meta pigsty]$ make install
./infra.yml -l meta
```

https://pigsty.cc/zh/docs/quick-start/

Monitoring

You can’t manage what you don’t MEASURE.

— Peter F. Drucker

监控系统

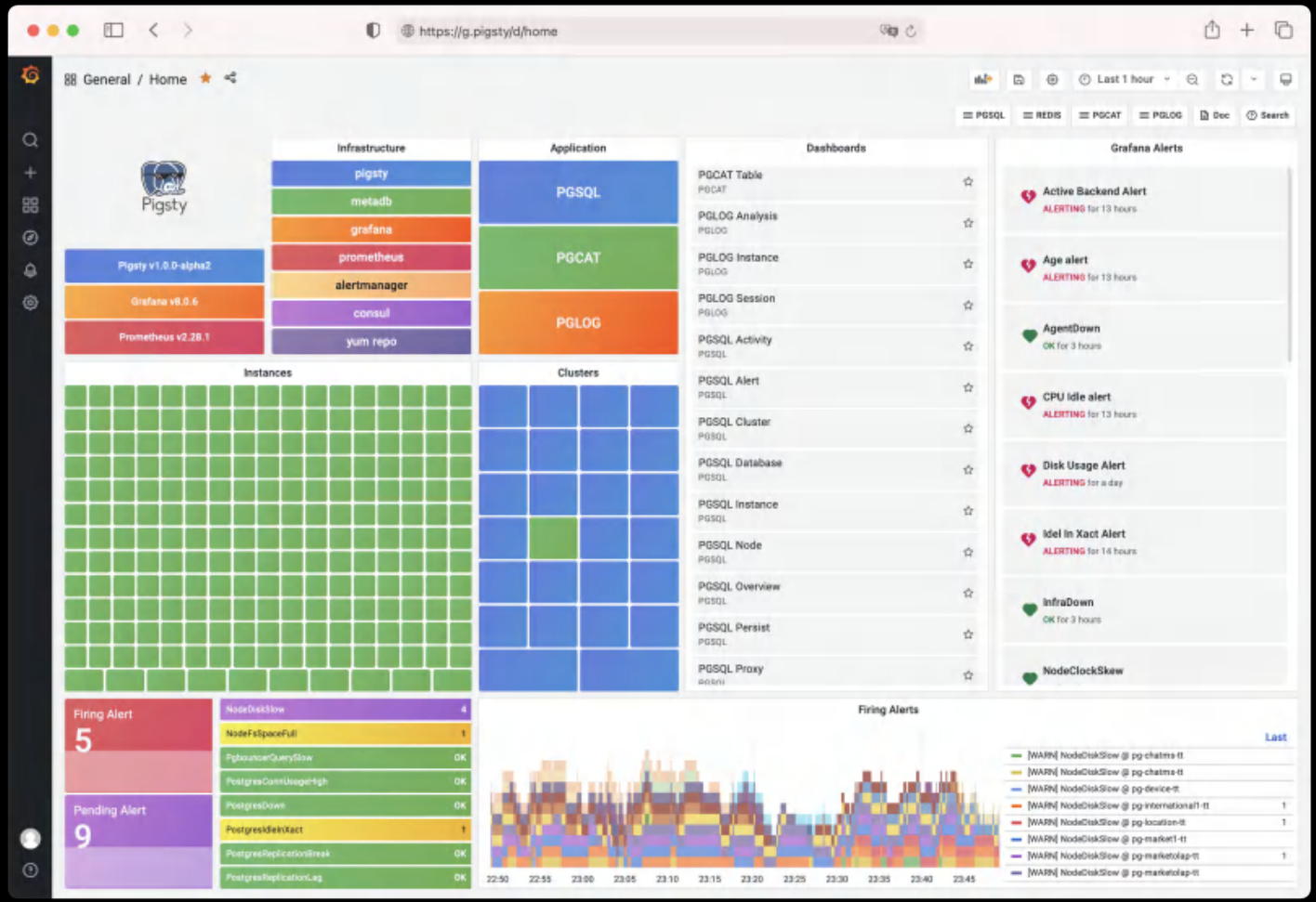
开源监控最佳实践

大规模集群管理

多层次，全覆盖

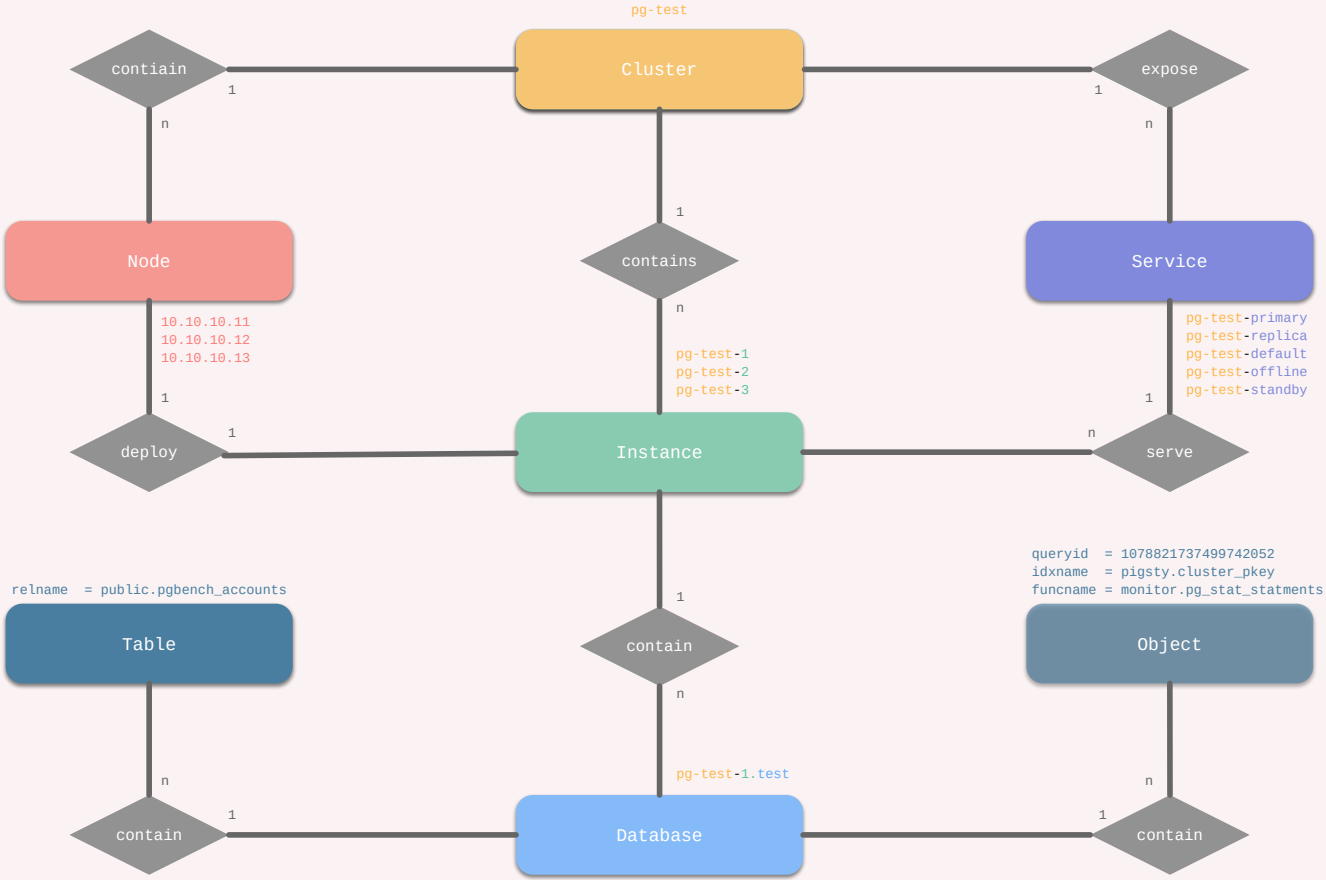
生产验证方案

最专业，最全面

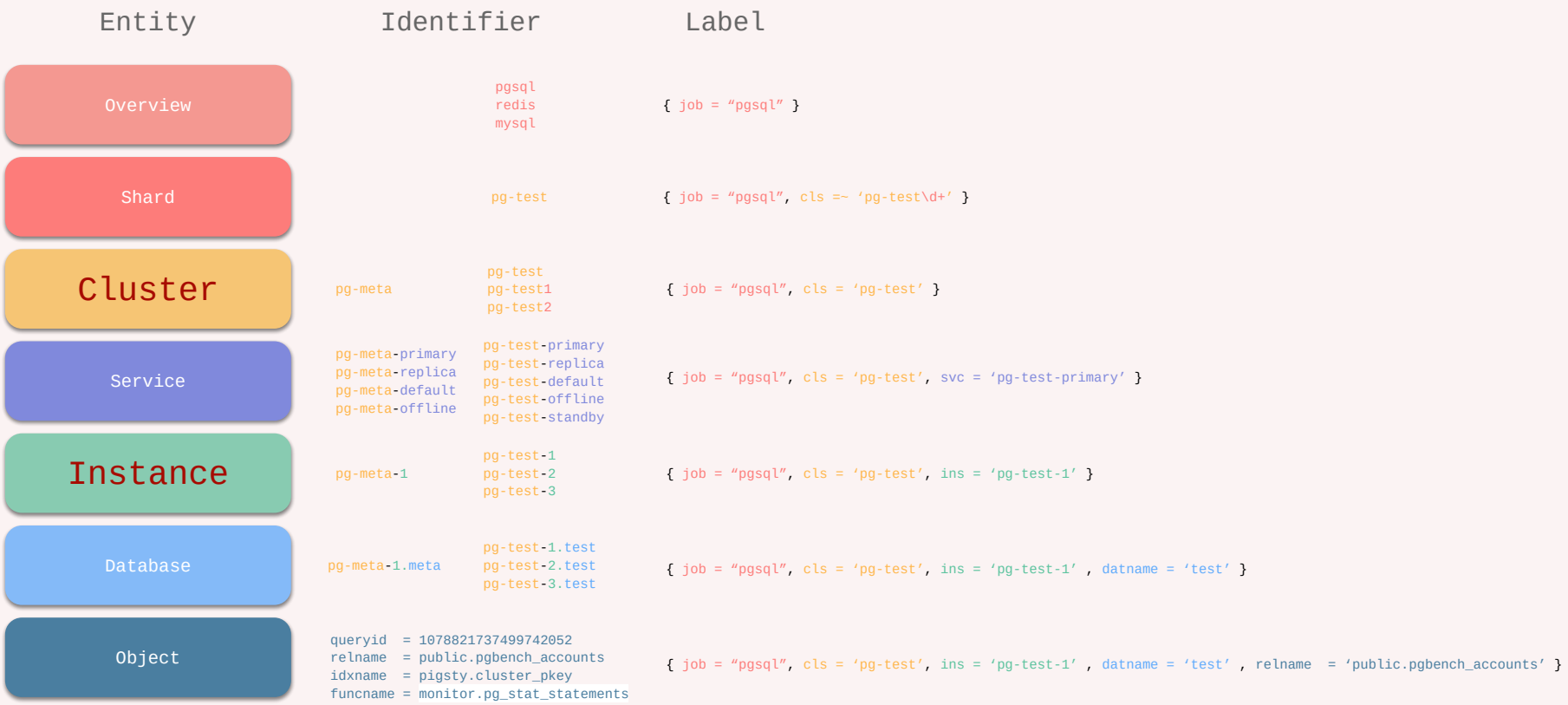


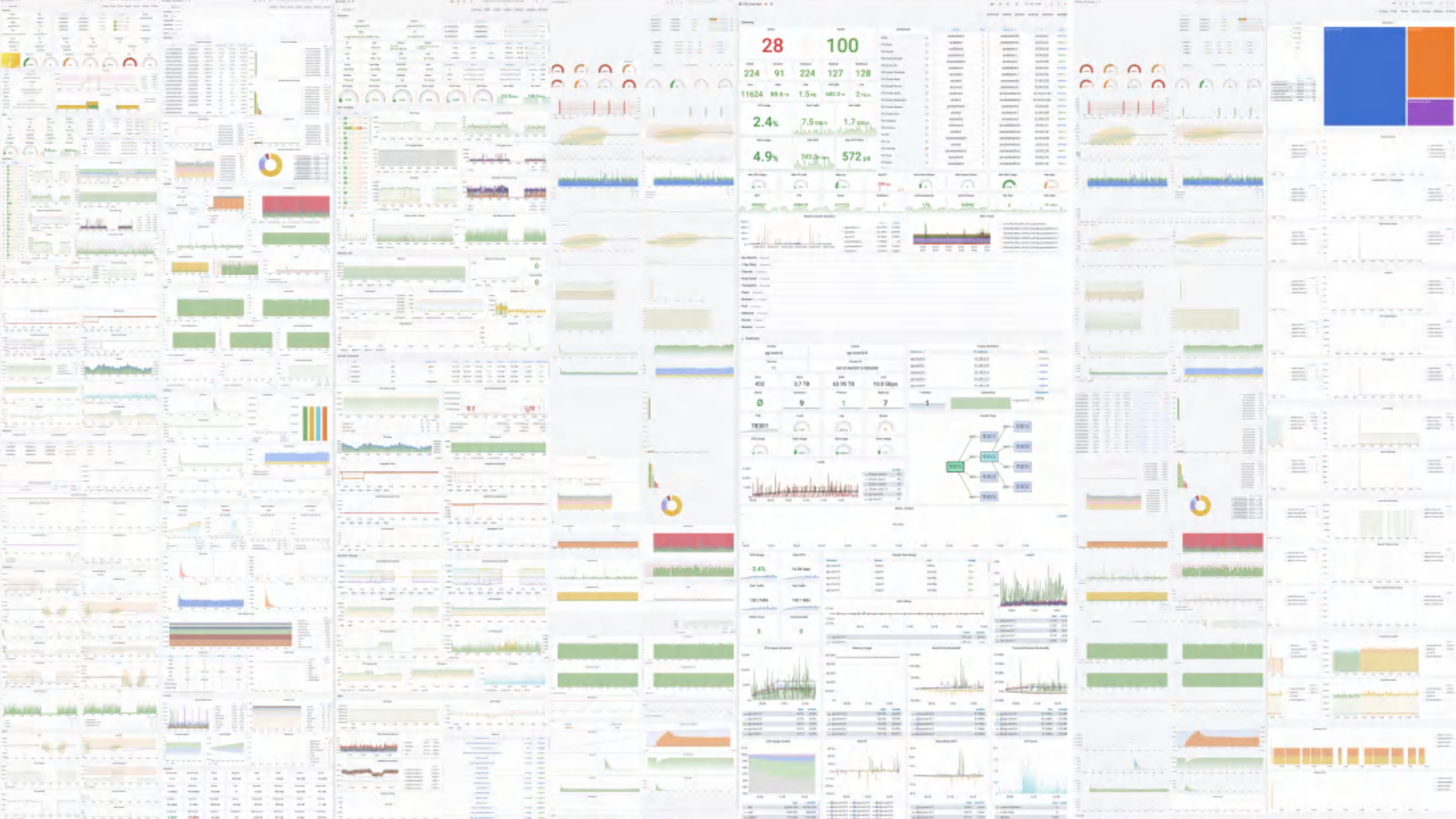
Domains

Entity
Relation
Identifier



Entities & Identifiers





Pigsty / PG Table Catalog

Last 1 hour

instance: DBCenter | relation: instance_all_query_info

Overview | Cluster | Service | Instance | Database

- Catalog

Namespace / Schema	Relation	Relation Kind
dba_daily_report	instance_all_query_info	(r) Table

Size	Rows	Page	Age	Live Tuple	Dead Tuple	Dead Ratio	Bloat Ratio
48.4 MB	51.0 K	5903	67955885	50976	0	0%	40.3%

Identifier	Value
Relation OID	2285126
Schema OID	17008
Reltype OID	2285128
Comptype OID	
Owner OID	16390
Access Method ID	2
Rel Filenode	2285126
Tablespace OID	
Toast Table OID	2285130

Features	Value
Has Index	TRUE
Is Partitioned	FALSE
Is Shared	FALSE
Has Rule	FALSE
Has Trigger	FALSE
Has Subclass	FALSE
Row level security	FALSE
Force RLS	FALSE
Is Populated	TRUE

Field	Value
Rel Pages	5903
Rel Bytes	48357376
Rel Tuples	50976
Rel All Visible	5903
Columns	14
Checks	0
Frozen XID	2182795092
Age	67955885
Min XID	1

Persist Feature	Value
Unlogged	FALSE
Temporary	FALSE

ACL

dbuser_dba=arwdDxt/dbuser_dba

dbuser_grafana=r/dbuser_dba

Options

No data to show

Tuple	Value
Seq Scan	3
Seq Scan Tuples	0
Index Scan	4273996
Index Scan Tuples	4222970
Inserted Tuples	50995
Deleted Tuples	0
Updated Tuples	4222926
Hot-Updated Tuples	1013160

Vacuum	Value
Vacuums	4579
Vacuum Count	0
Lost Vacuum	
Since Last Vacuum	
AutoVacuum Count	4579
Last AutoVacuum	2021-04-25 00:31:00
Since Last AutoVacu...	1 day 21:28:58
Inserts since Last Va...	0

Analyze	Value
Analyzes	687
Analyze Count	0
Last Analyze	
Last Analyze Elapse	
AutoAnalyze Count	687
Last AutoAnalyze	2021-04-01 13:01:35
Last AutoAnalyze Ela...	25 days 08:58:23
Modify Since Analyze	0

Blocks I/O	Value
Heap Blocks Read	198334
Heap Blocks Hit	82116341
Index Blocks Read	29864
Index Blocks Hit	37696619
Toast Blocks Read	13484
Toast Blocks Hit	11570860
Toast Index Blocks R...	291
Toast Index Blocks Hit	1087925

Attributes

Seq #	Name	Storage	Align	NOT NULL	Null Frac	Avg Width	N Distinct	Correlation
1	id	p	d	true	0	8	-1	0.565
2	instance_ip	x	i	false	0	12	203	-0.13
3	database_name	x	i	false	0	18	47	-0.813
4	queryid	p	d	false	0	8	-0.31	0.00699
5	query	x	i	false	0	405	-0.37	0.8758
6	calls	p	d	false	0	8	-0.64	0.8925

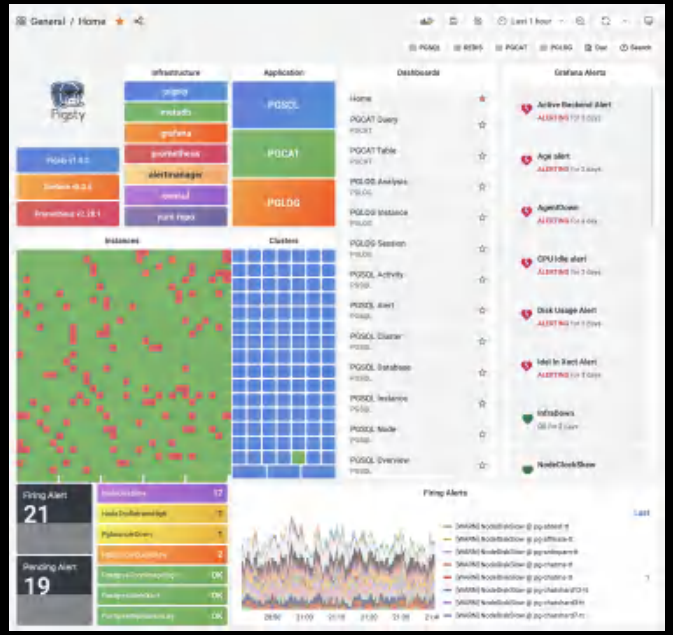
Indexes

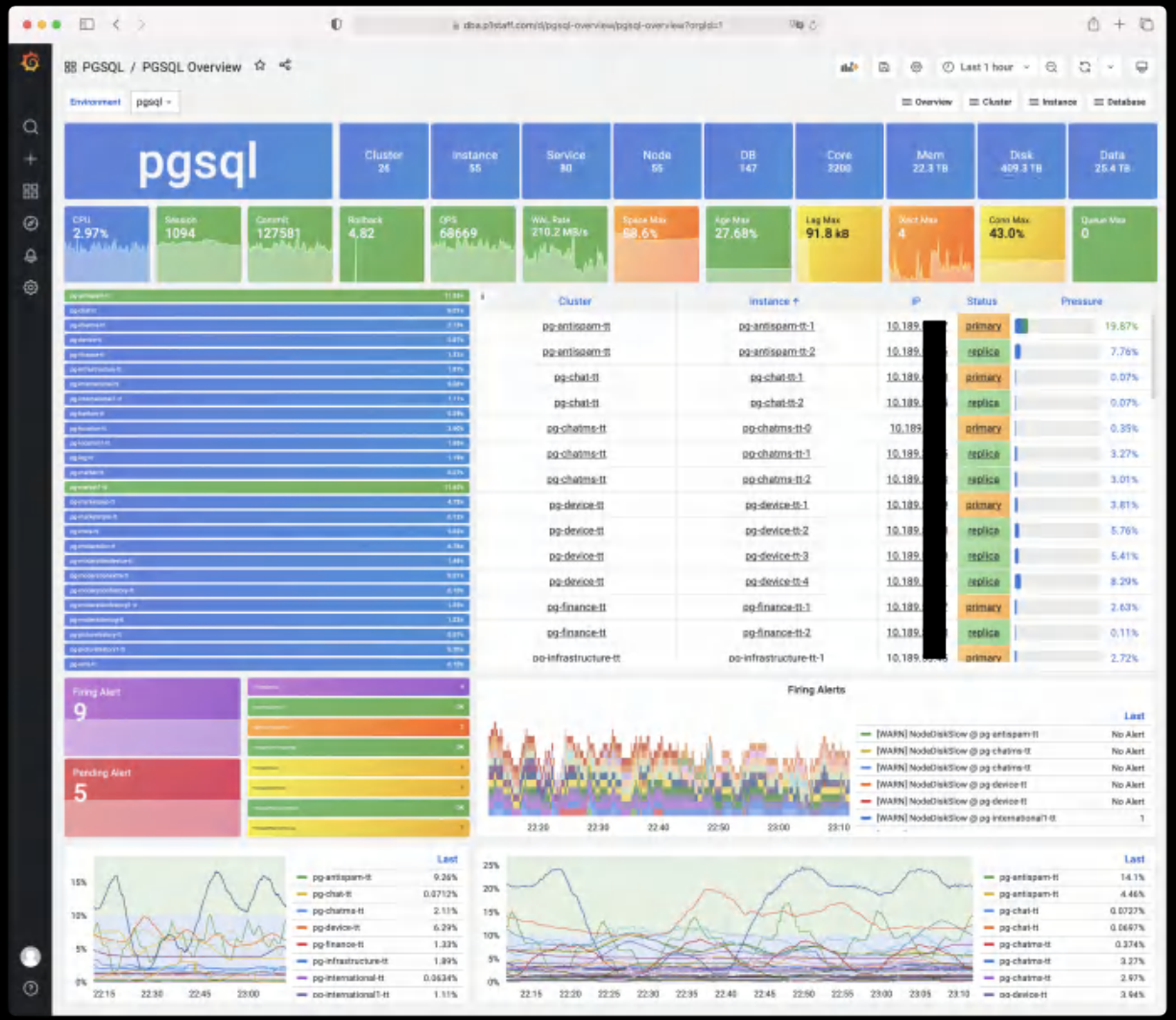
Schema	Index	Index ID	Table ID	Scan	Tup Read	Tup Fetch
dba_daily_r...	instance_all_query_info_pkey	2285406	2285126	0	0	0
dba_daily_r...	instance_all_query_info_create_date_idx	2285533	2285126	9	0	0
dba_daily_r...	instance_all_query_info_instance_ip_database_name_queryid_c_idx	2285534	2285126	4.27 Mil	7.98 Mil	4.22...

Relation Size


pgsql
3.23%
21.2%
OK
v7新UI

PGSQL Overview





PGSQL Alert

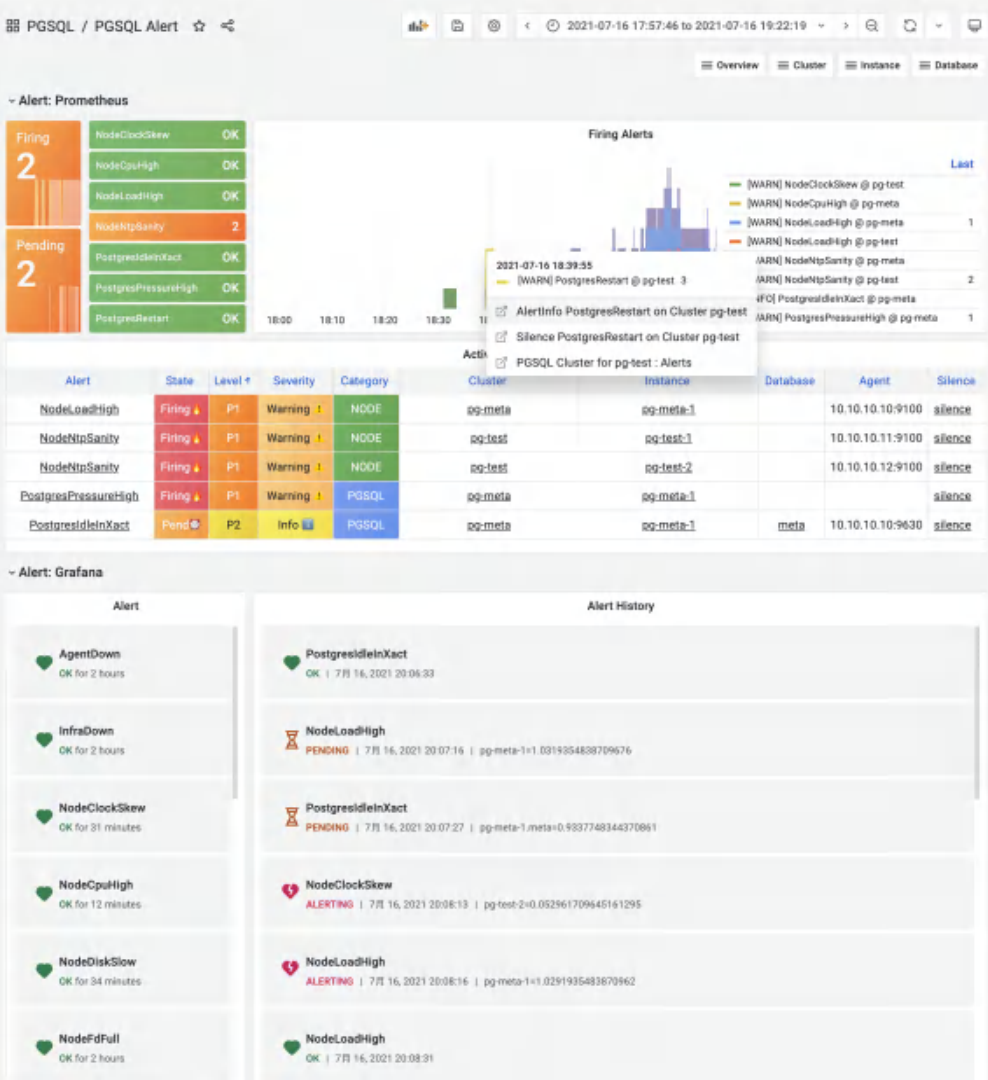
Prometheus

AlertManager

Grafana

等效实现

互为补充



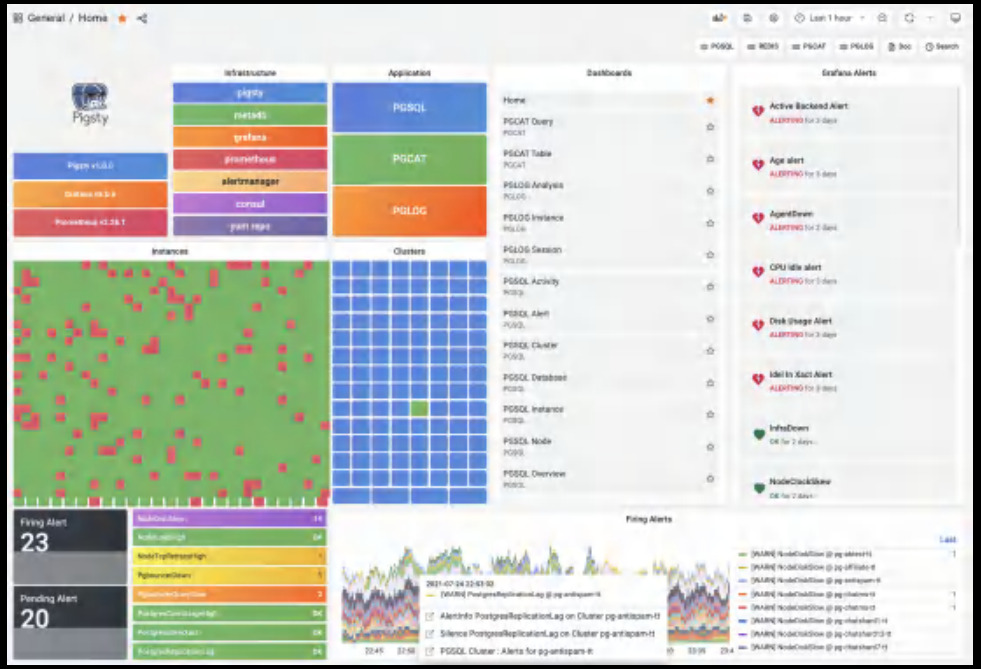
交互式图表

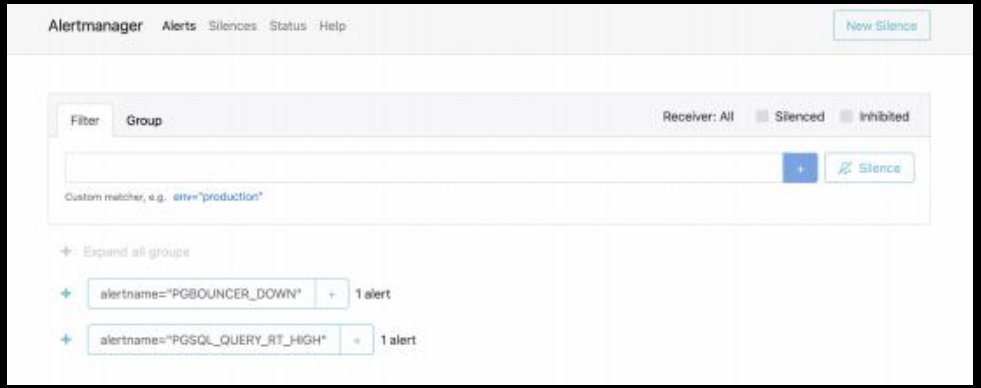
点击跳转

查看详情

屏蔽告警

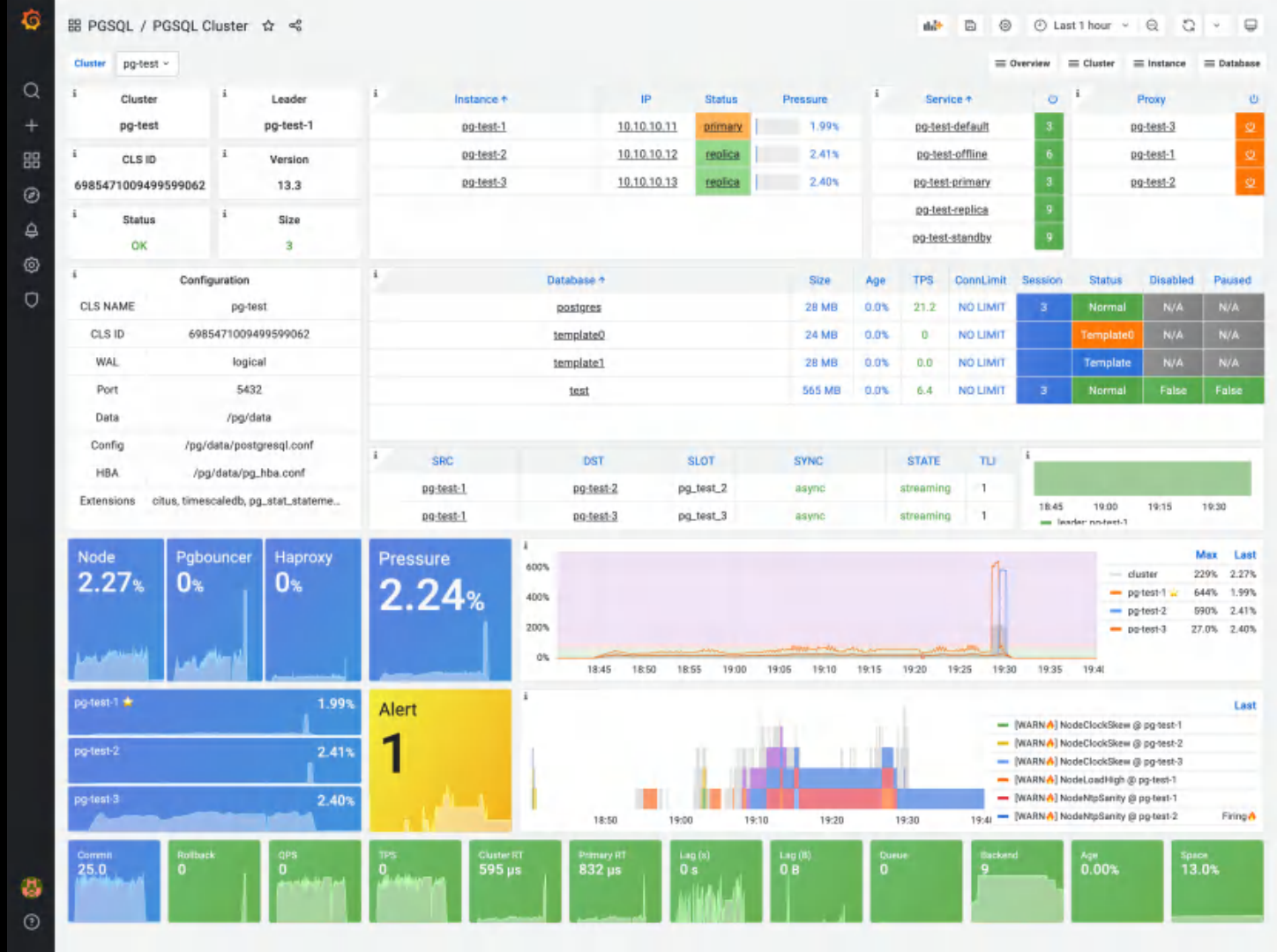
下钻至具体集群/实例





PGSQL Cluster

cluster-wide insights



PGSQL Activity

Replication

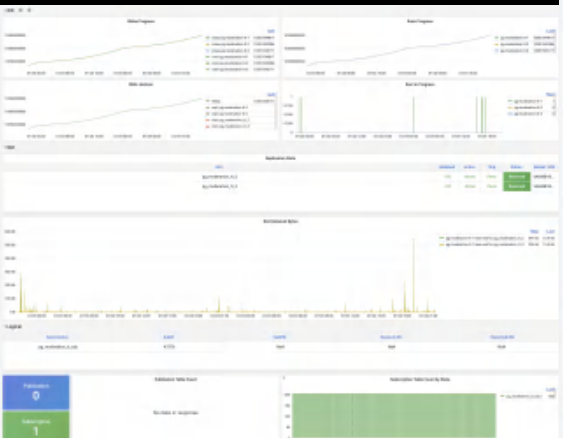
跟踪复制进度

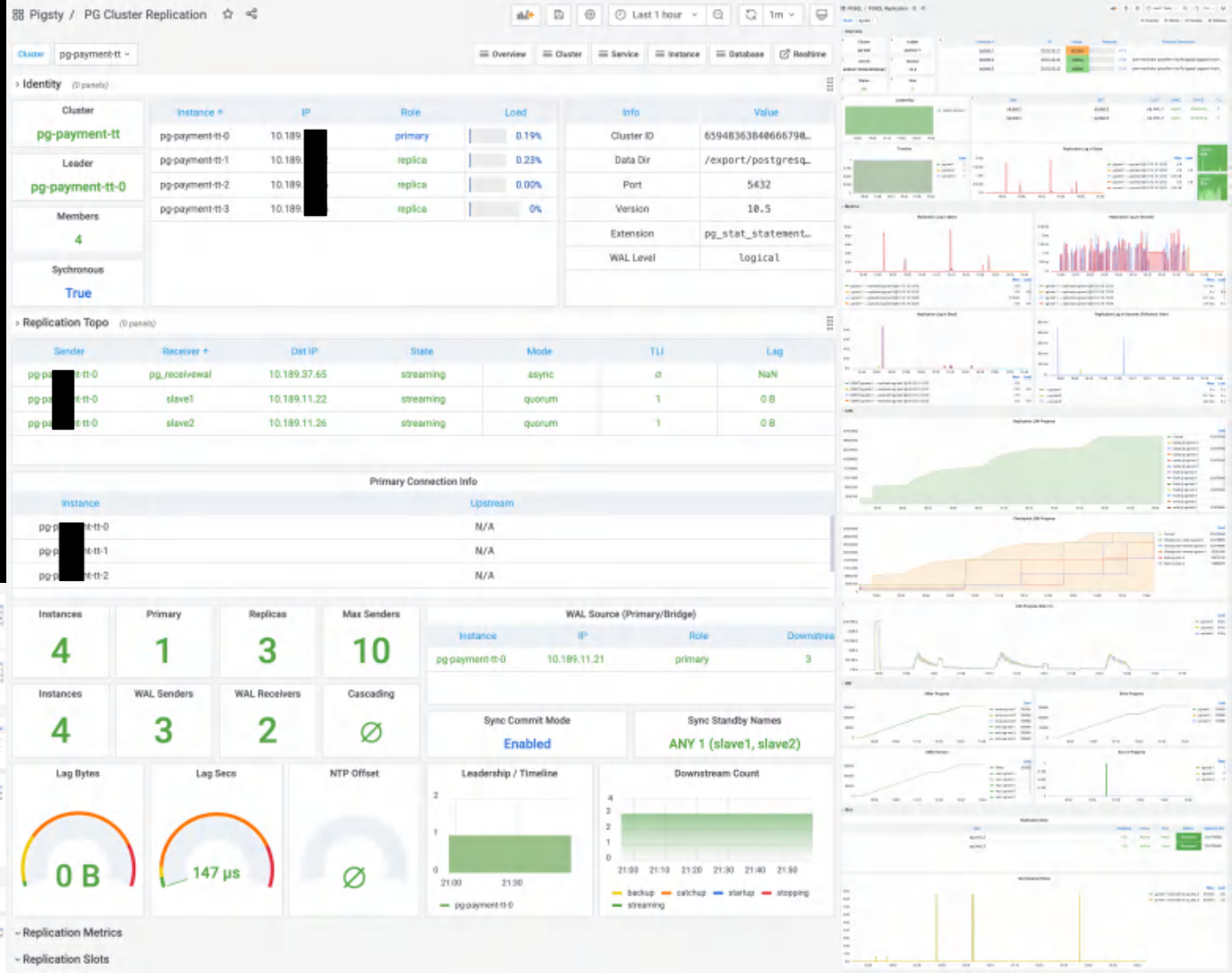
监控复制延迟

定位复制延迟问题

逻辑复制进度

同步复制状态





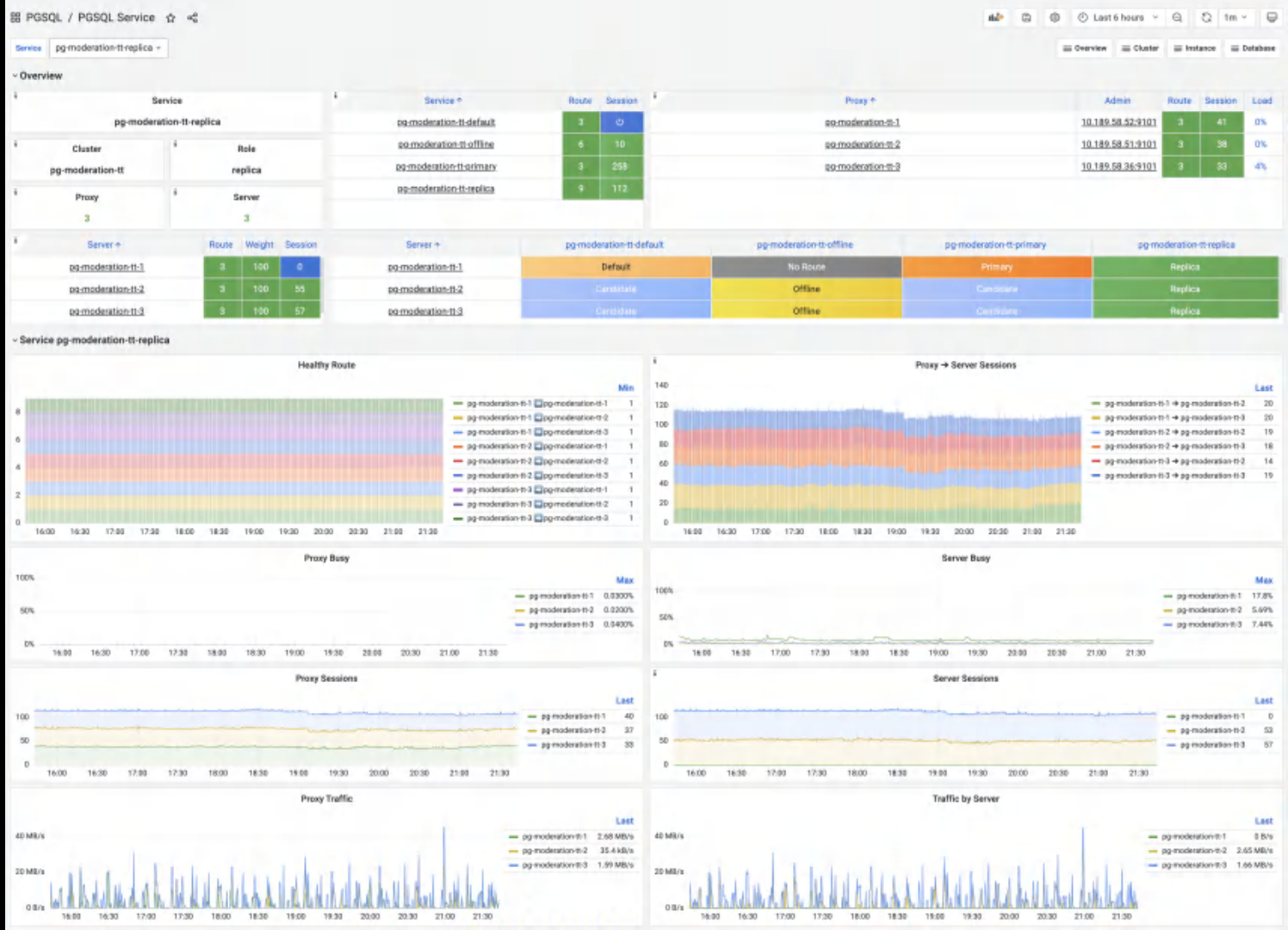
Service

Load Balancer

Proxy Instance

Traffic Control

.....

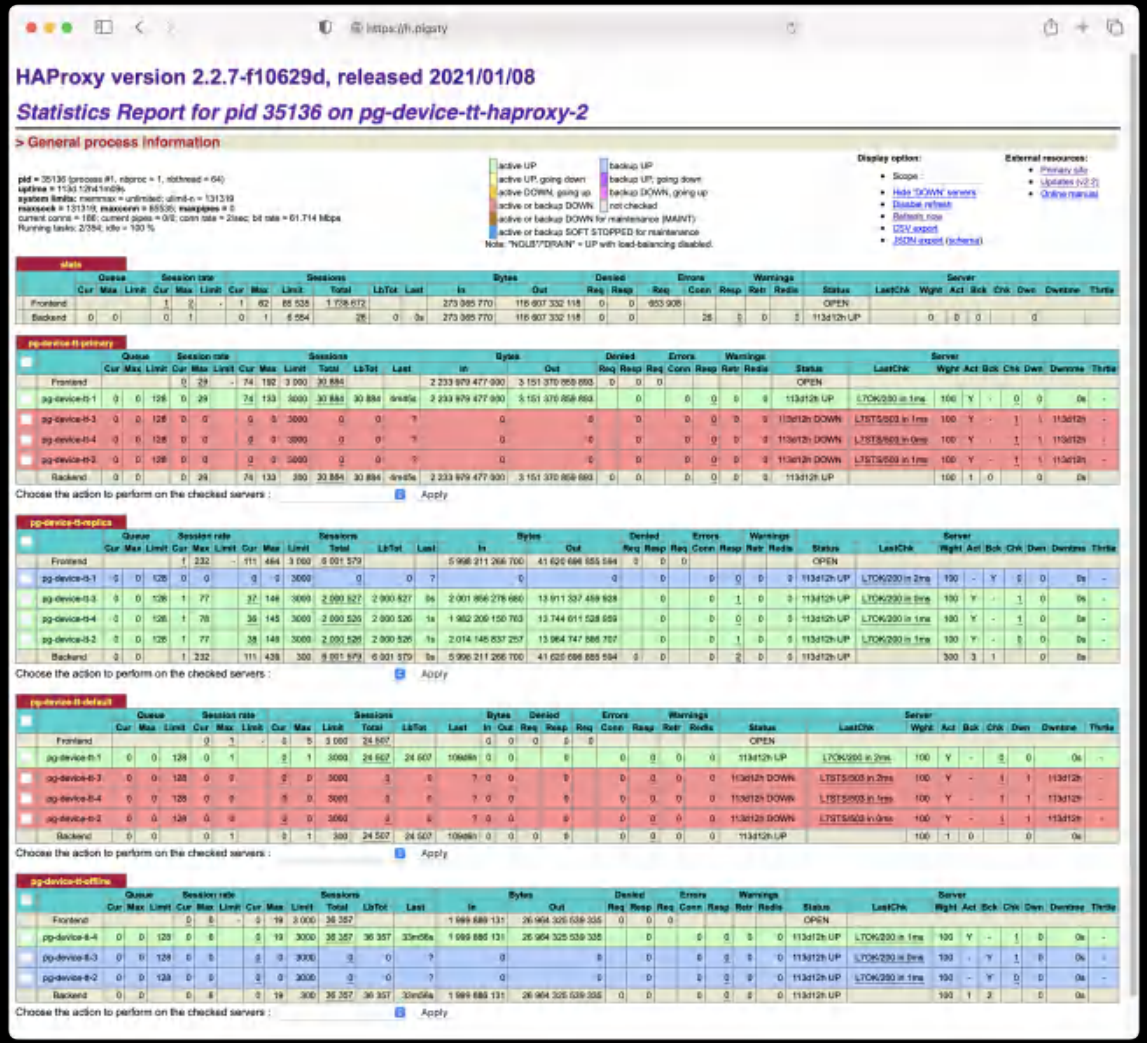


HAProxy Admin

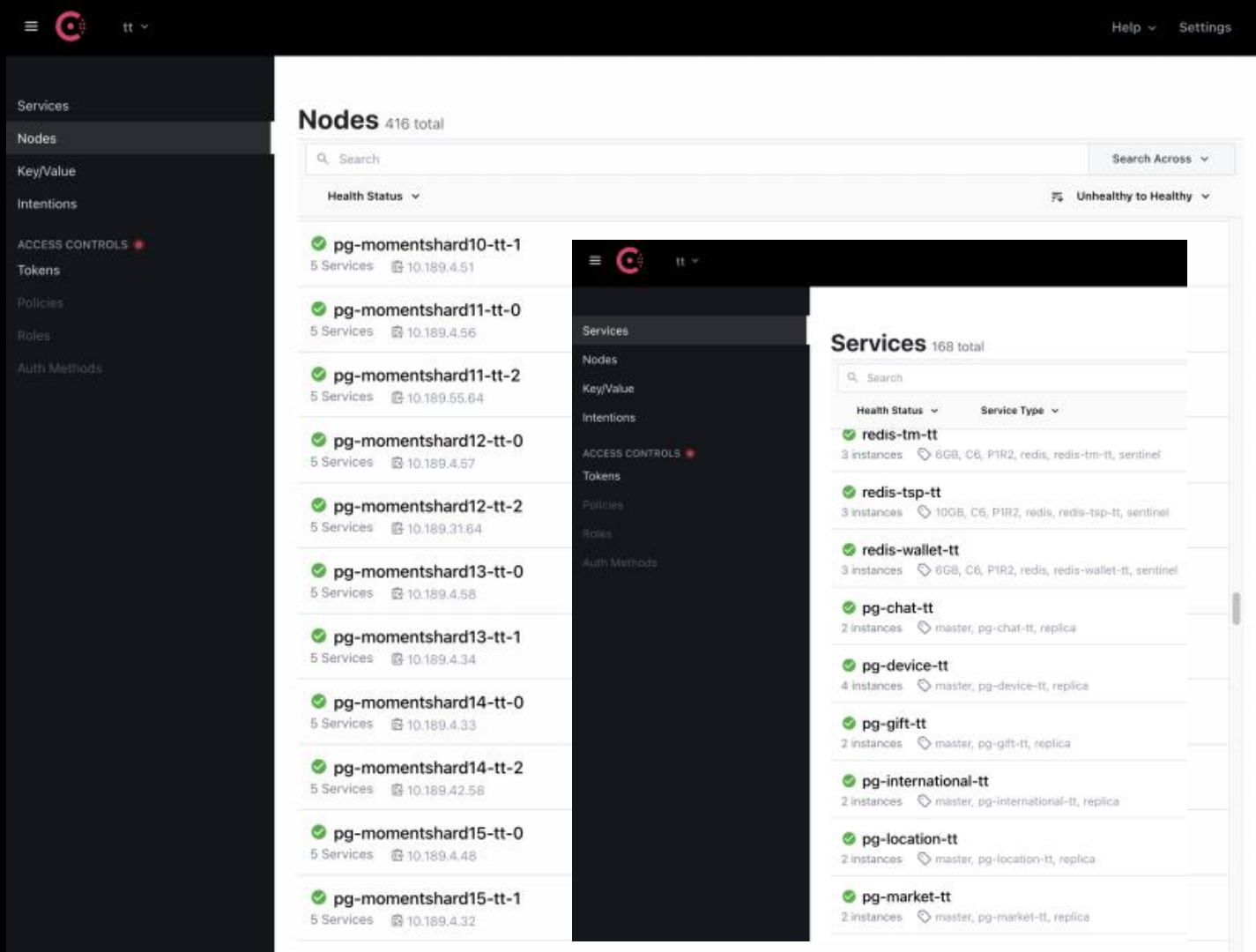
点击管理图标

直接从监控面板跳转

至流量管理面板



Service Registry With Consul



PGSQL / PGSQL Instance

PG Instance

Last 1 hour | 1m

Instance: pg-test-3

Overview | Cluster | Instance | Database

Cluster	Instance
pg-test	pg-test-3

Status	Node IP
Follower	10.10.10.13

Node Name
node-3

Instance	IP	Status	Pressure
pg-test-1	10.10.10.11	primary	1.99%
pg-test-2	10.10.10.12	replica	2.19%
pg-test-3	10.10.10.13	replica	2.13%

Service	
pg-test-default	
pg-test-offline	3
pg-test-primary	
pg-test-replica	3
pg-test-standby	3

Configuration	
Version	13.3
CLS	pg-test
CLS ID	6985471009499599062
WAL	logical
Port	5432
Data	/pg/data
Config	/pg/data/postgresql.conf

Database	Size	Age	TPS	PG CONN	ConnLimit	Session	PGB CONN	Disabled	Paused
test	188 MB	0.0%	0.7	0%	NO LIMIT	1	1%	False	False
postgres	9 MB	0.0%	7.1		NO LIMIT	1	2%	N/A	N/A
template1	9 MB	0.0%	0		NO LIMIT		0%	N/A	N/A
template0	8 MB	0.0%	0		NO LIMIT		0%	N/A	N/A

Aliveness

Node UP	PG UP	postgres	pgbouncer	node	haproxy
1.73 hour	1.13 hour	UP	UP	UP	UP

Replication

recovery	sync	children	pub	sub	
Yes	No	0	0	0	10.10.10.11 Parent

Node 1.72%

Commit 7.78

RT 538 µs

QPS 0

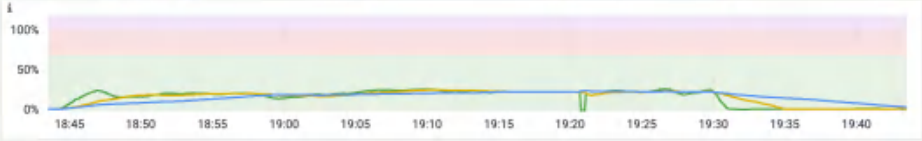
Queue 0

Lag 0 s

Space 12.6%

Pgbouncer 0%

Pressure 2.13%

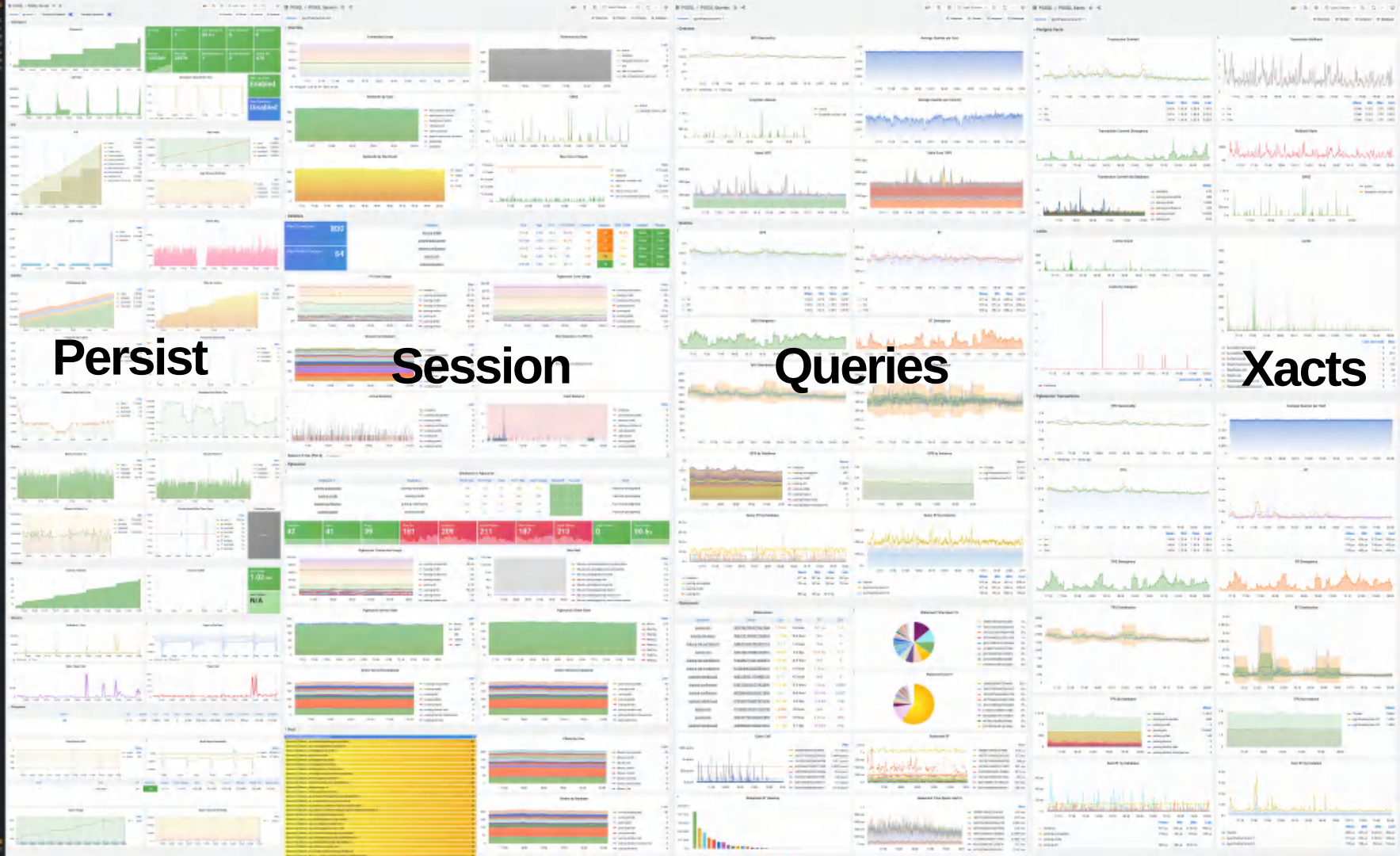


Haproxy 0%

Alert 0

Node

Proxy


Persist
Session
Queries
Xacts

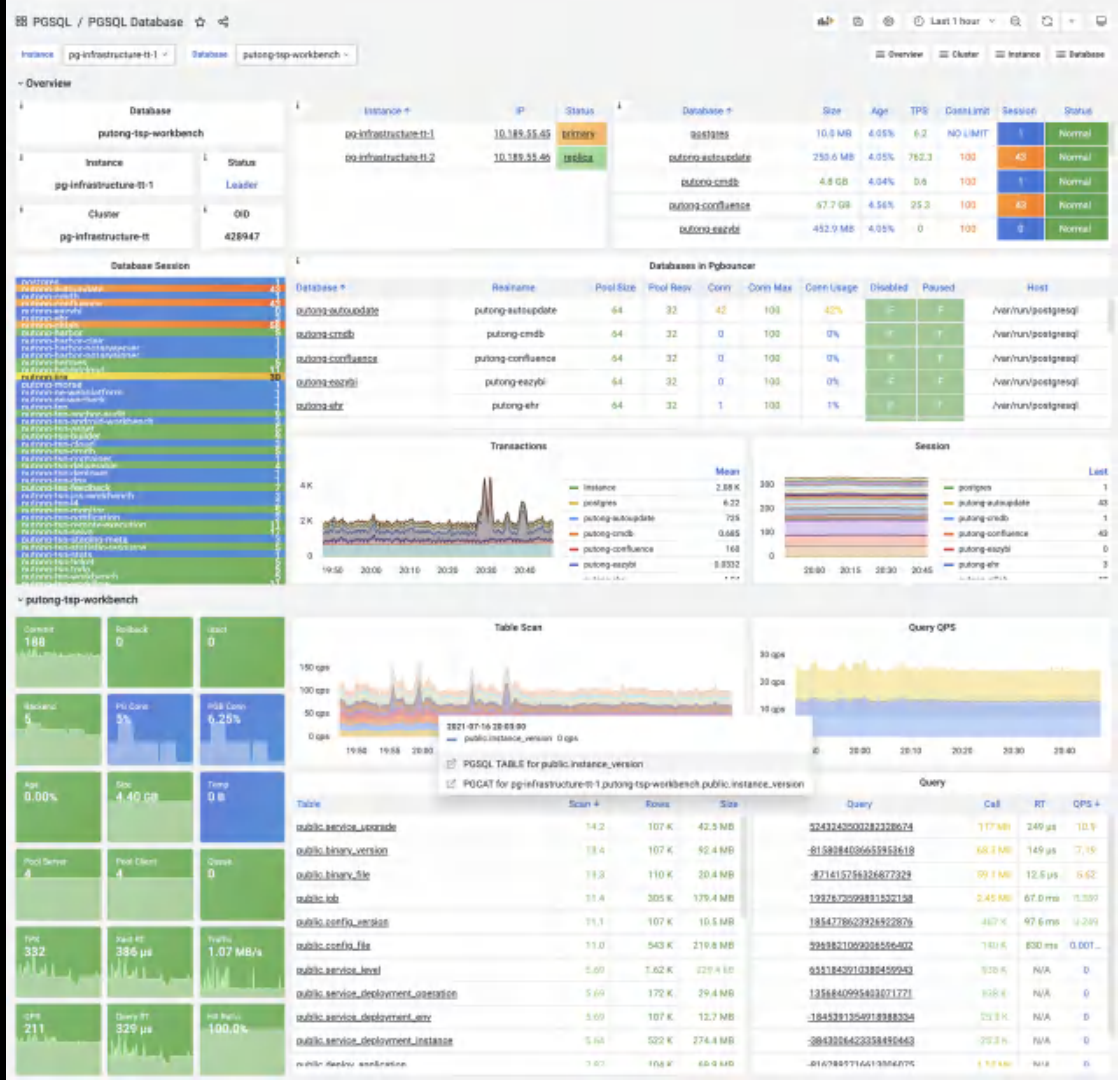
PGSQL Database

数据库级别指标

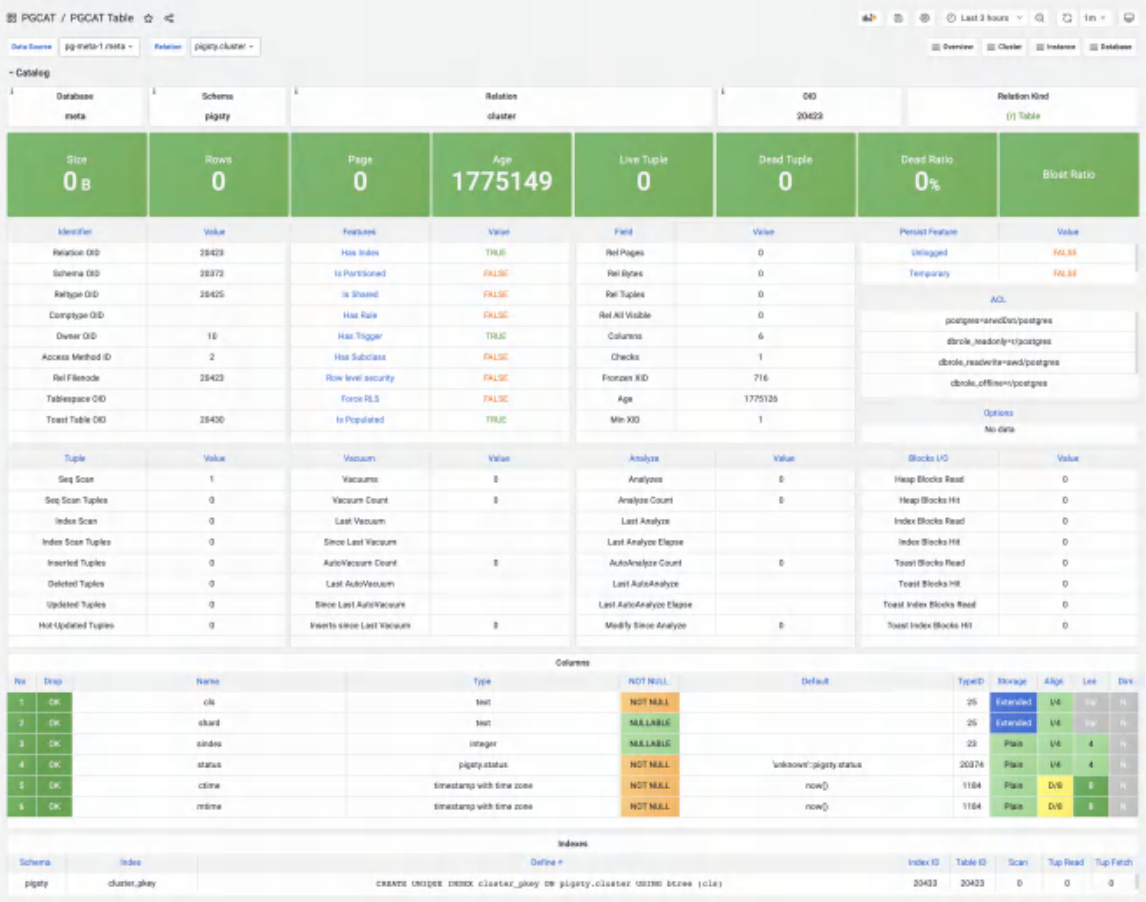
快速访问

每一张表

每一个查询



PGSQL Table


PGCAT / PGCAT Table
Catalog
Database
meta
Schema
pigsty
Relation
cluster
OID
20423
Relation Kind
Size
0 B
Rows
0
Page
0
Age
1775149
Live Tuple
0
Dead Tuple
0
Dead Ratio
0%
Bloat Ratio
Columns
Indexes

PGSQL Query

PGCAT / PGCAT Query

Data Source: pg-sms-tt-1.putong-sms | Query: -3288231039287227845 | Last 1 hour

Overview | Cluster | Instance | Database

Cluster	Instance	Database
pg-sms-tt	pg-sms-tt-1	putong-sms

Query Identifier

-3288231039287227845

Query String

```
SELECT CURRENT_CATALOG AS datname, schemaname || $1 || indexrelname AS idxname, schemaname || $2 || relname AS
relname ,idx_scan, idx_tup_read, idx_tup_fetch, idx_blks_read, idx_blks_hit
  FROM pg_stat_user_indexes psui, LATERAL (SELECT idx_blks_read, idx_blks_hit FROM pg_statio_user_indexes psio
WHERE psio.indexrelid = psui.indexrelid) p2
  ORDER BY idx_tup_read DESC LIMIT $3
```

Users

{dbuser_monitor}

Exec Time	Value
μ	10.9 ms
σ	1.06 ms
min	0.862 ms
max	12.5 ms
sum	4.58 min

Plan Time	Value
μ	
σ	
min	
max	
sum	

Metric	Value
plans	
calls	25113
rows	1356102
shared_blks_hit	18222435
shared_blks_read	
shared_blks_dirtied	
shared_blks_written	
local_blks_hit	
local_blks_read	
local_blks_dirtied	
local_blks_written	
temp_blks_read	
temp_blks_written	
blk_read_time	
blk_write_time	
wal_records	
wal_fpi	
wal_bytes	

QPS

RT

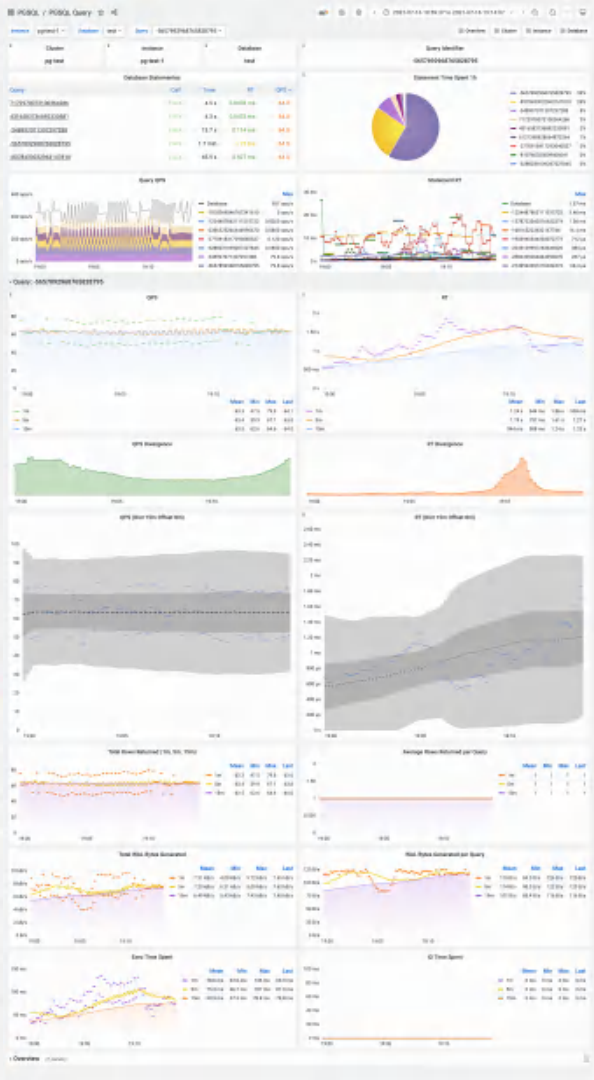
Statements Stat for -3288231039287227845

User	calls	rows	plans	total_plan_time	min_plan_time	max_plan_time	mean_plan_time	stddev_plan_tim	total_exec_time	min_exec_time	max_exec_time	mean_exec_time	st
dbuser_monitor	25.1 K	1.36 Mil	0	0 ms	0 ms	0 ms	0 ms	0 ms	4.6 min	0.9 ms	12.5 ms	10.9 ms	

Overview

Statements

Query	SQL	calls	rows	total_exec	min_exec_	max_exec	mean_exe	blk_read_t	blk_write_	wal_records	wal_fpi	wal_bytes
7226155749781812196	SELECT current_catalog, current_user, …	1.54 Mil	1.54 Mil	1.3 min	0.0 ms	0.2 ms	0.1 ms	0.0 ms	0 ms	1.0	1.0	4.5 kB
5769961677013624247	SELECT datid, datname, numbackends, xa…	679 K	679 K	1.9 min	0.1 ms	10.5 ms	0.2 ms	0 ms	0 ms	0	0	0 B
-1903945346310072773	SELECT datname, pid, relid::RegClass A…	679 K	0	15.3 s	0.0 ms	0.3 ms	0.0 ms	0 ms	0 ms	0	0	0 B
5845664379928311205	select schemaname,relname,indexrelname…	121	4.24 K	1.3 s	0.7 ms	11.0 ms	10.4 ms	0 ms	0 ms	0	0	0 B
2342127242629226027	SELECT count(*) AS cnt FROM yay.secret…	10	10	9.6 s	852.9 ms	1.3 s	957.2 ms	895.1 ms	0 ms	10.0 K	5.8 K	47.4 MB
-2260398699914946894	SELECT query FROM monitor.pg_stat_stat…	5	5	2.3 ms	0.4 ms	0.5 ms	0.5 ms	0 ms	0 ms	0	0	0 B
7071073393163095576	SELECT queryid::TEXT AS qid, * FROM mo…	4	40	2.8 ms	0.4 ms	0.8 ms	0.7 ms	0 ms	0 ms	0	0	0 B
-9428308577950747751	SELECT c2.relname, i.indisprimary, i.i…	2	2	14.7 ms	0.3 ms	14.3 ms	7.3 ms	0 ms	0 ms	0	0	0 B
2376857088910362203	SELECT * FROM pg_sequences	2	32	1.0 ms	0.2 ms	0.8 ms	0.5 ms	0 ms	0 ms	0	0	0 B
-2941126072218693949	SELECT count(*) AS cnt FROM sms_promot…	2	2	51.5 s	25.6 s	25.9 s	25.7 s	16.7 s	0 ms	0	0	0 B

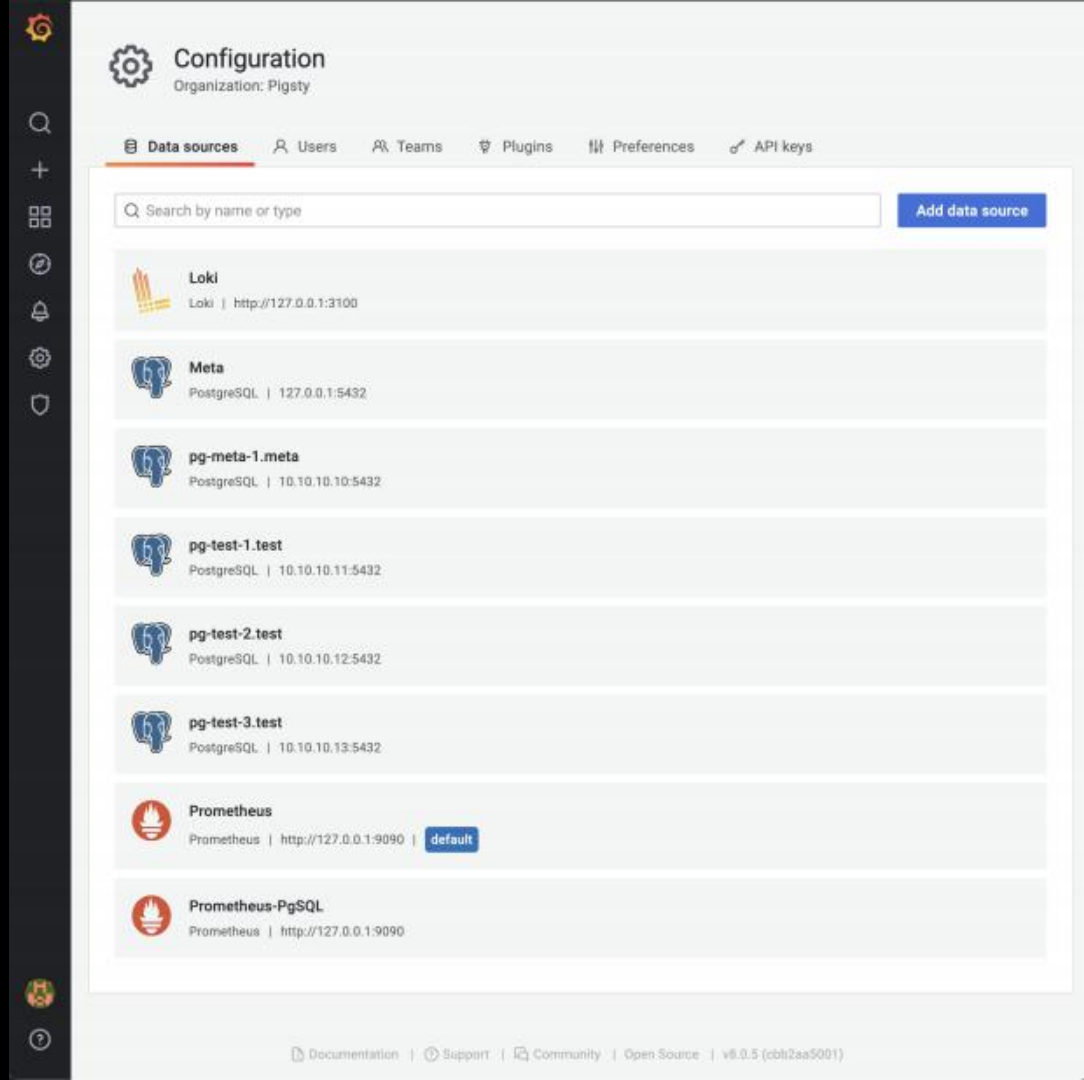


PGCAT

自动为新数据库注册数据源

通过Grafana直接浏览系统目录

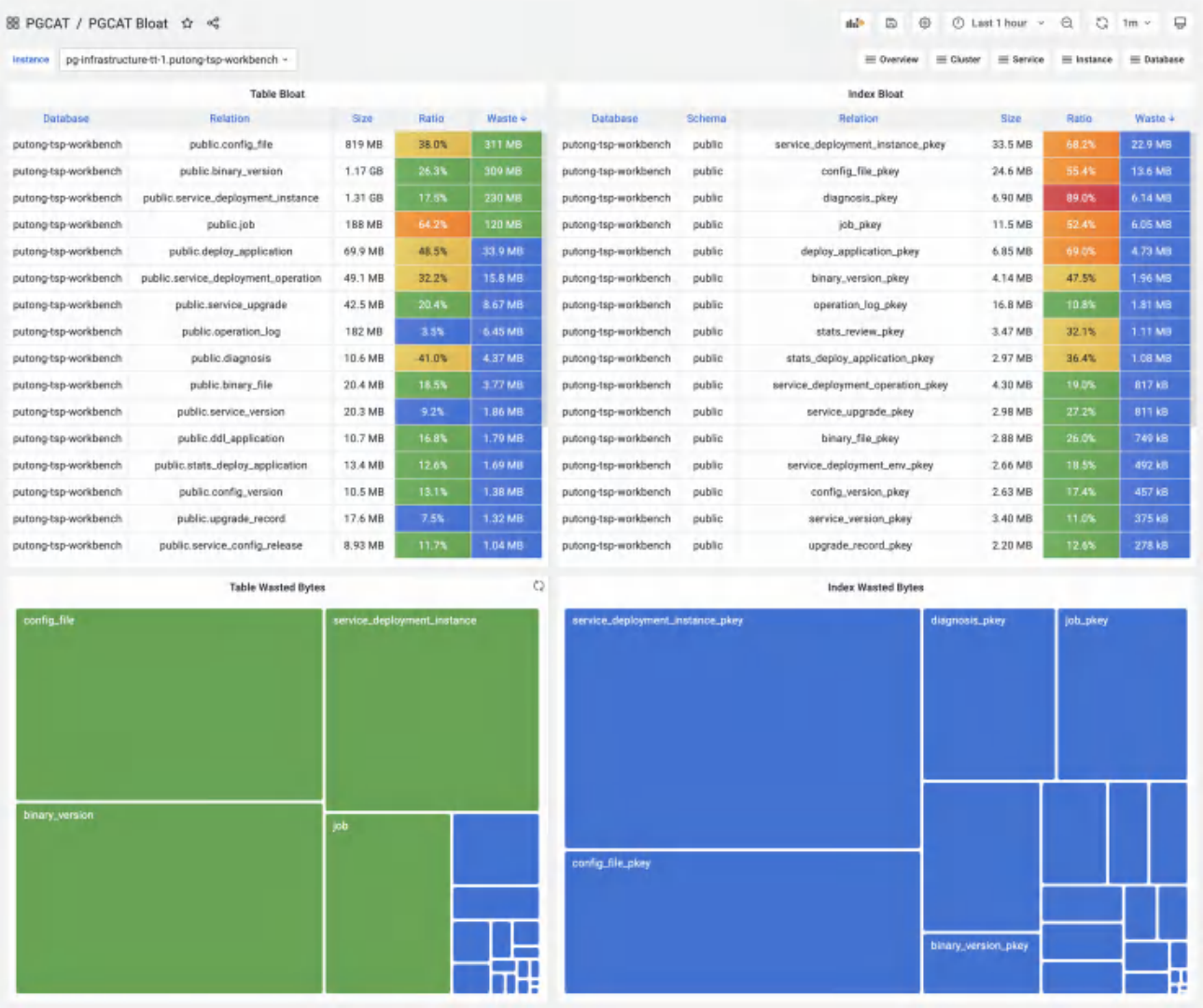
带来无限新可能性



PGCAT Bloat

表与索引膨胀总览

从Catalog获取信息



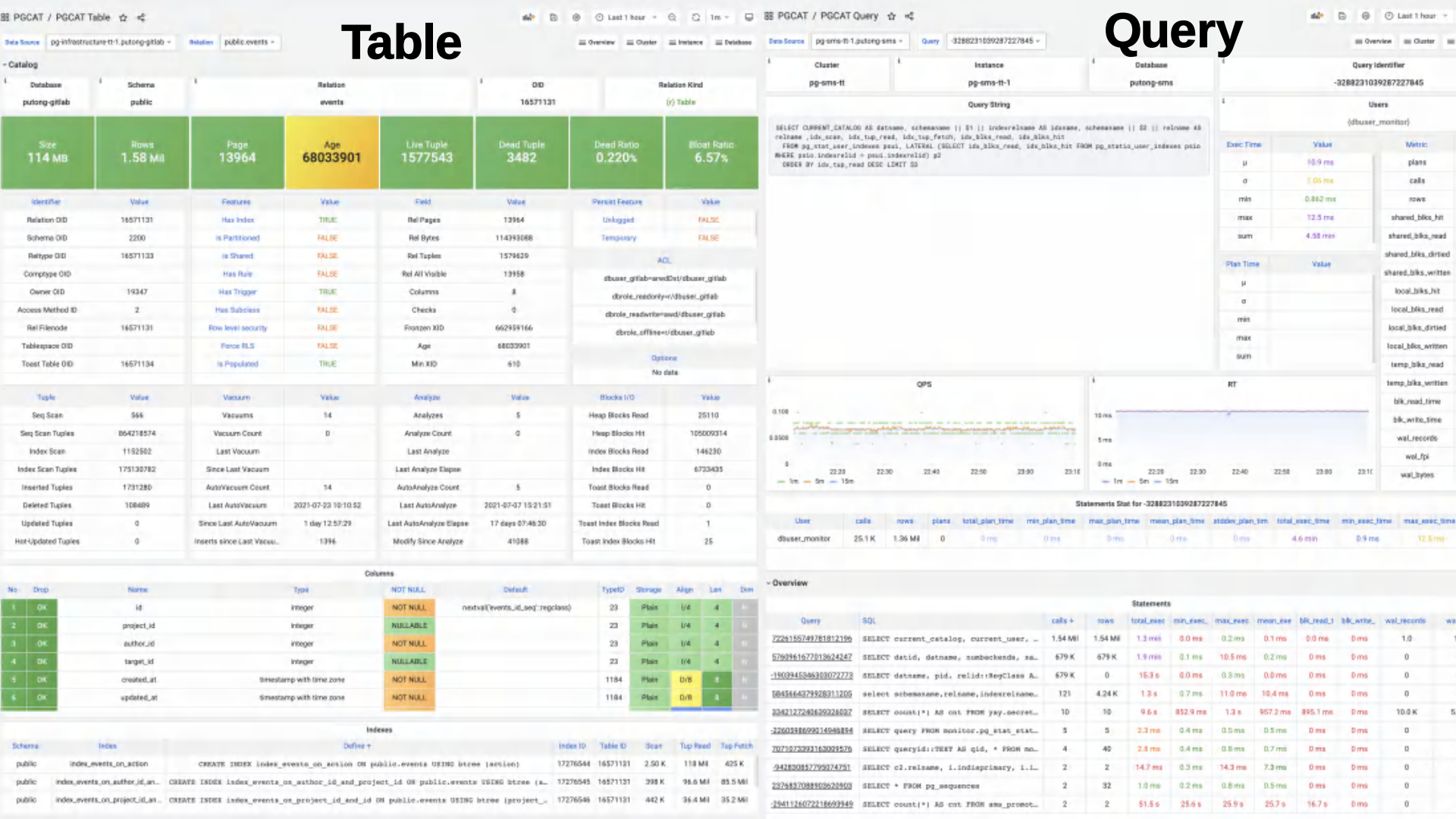


Table
Query
PGCAT / PGCAT Table
PGCAT / PGCAT Query
Catalog
Database
putong-gitlab
Schema
public
Relation
events
OID
16571131
Relation Kind
(r) Table
Size
114 MB
Rows
1.58 Mil
Page
13964
Age
68033901
Live Tuple
1577543
Dead Tuple
3482
Dead Ratio
0.220%
Bloat Ratio
6.57%
Identifier
Value
Relation OID
16571131
Schema OID
2200
Reltype OID
16571133
Comptype OID
Owner OID
19347
Access Method ID
2
Rel Filenode
16571131
Tablespace OID
Toast Table OID
16571134
Features
Value
Has Index
TRUE
Is Partitioned
FALSE
Is Shared
FALSE
Has Rule
FALSE
Has Trigger
TRUE
Has Subclass
FALSE
Row level security
FALSE
Force RLS
FALSE
Is Populated
TRUE
Field
Value
Rel Pages
13964
Rel Bytes
114393088
Rel Tuples
1579629
Rel All Visible
13958
Columns
8
Checks
0
Frozen XID
662959166
Age
68033901
Min XID
610
Persist Feature
Value
Unlogged
FALSE
Temporary
FALSE
ACL
dbuser_gitlab=arwdDxt/dbuser_gitlab
dbrole_readonly=r/dbuser_gitlab
dbrole_readwrite=awd/dbuser_gitlab
dbrole_offline=r/dbuser_gitlab
Options
No data
Tuple
Value
Seq Scan
566
Seq Scan Tuples
864218574
Index Scan
1152502
Index Scan Tuples
175130782
Inserted Tuples
1731280
Deleted Tuples
108489
Updated Tuples
0
Hot-Updated Tuples
0
Vacuum
Value
Vacuums
14
Vacuum Count
0
Last Vacuum
Since Last Vacuum
AutoVacuum Count
14
Last AutoVacuum
2021-07-23 10:10:52
Since Last AutoVacuum
1 day 12:57:29
Inserts since Last Vacuu...
1396
Analyze
Value
Analyzes
5
Analyze Count
0
Last Analyze
Last Analyze Elapse
AutoAnalyze Count
5
Last AutoAnalyze
2021-07-07 15:21:51
Last AutoAnalyze Elapse
17 days 07:46:30
Modify Since Analyze
41088
Blocks I/O
Value
Heap Blocks Read
25110
Heap Blocks Hit
105009314
Index Blocks Read
146230
Index Blocks Hit
6733435
Toast Blocks Read
0
Toast Blocks Hit
0
Toast Index Blocks Read
1
Toast Index Blocks Hit
25
Columns
No
Drop
Name
Type
NOT NULL
Default
TypeID
Storage
Align
Len
Dim
1 OK id integer NOT NULL nextval('events_id_seq'::regclass) 23 Plain i/4 4
2 OK project_id integer NULLABLE 23 Plain i/4 4
3 OK author_id integer NOT NULL 23 Plain i/4 4
4 OK target_id integer NULLABLE 23 Plain i/4 4
5 OK created_at timestamp with time zone NOT NULL 1184 Plain d/8 8
6 OK updated_at timestamp with time zone NOT NULL 1184 Plain d/8 8
Indexes
Schema
Index
Define
Index ID
Table ID
Scan
Tup Read
Tup Fetch
public index_events_on_action CREATE INDEX index_events_on_action ON public.events USING btree (action) 17276544 16571131 2.50 K 118 Mil 425 K
public index_events_on_author_id_an... CREATE INDEX index_events_on_author_id_and_project_id ON public.events USING btree (a... 17276545 16571131 398 K 96.6 Mil 85.5 Mil
public index_events_on_project_id_an... CREATE INDEX index_events_on_project_id_and_id ON public.events USING btree (project_... 17276546 16571131 442 K 36.4 Mil 35.2 Mil
Data Source
pg-infrastructure-tt-1.putong-gitlab
Relation
public.events
Overview
Cluster
Instance
Database
Last 1 hour
1m
Data Source
pg-sms-tt-1.putong-sms
Query
-3288231039287227845
Cluster
pg-sms-tt
Instance
pg-sms-tt-1
Database
putong-sms
Query Identifier
-3288231039287227845
Query String
SELECT CURRENT_CATALOG AS datname, schemaname || $1 || indexrelname AS idxname, schemaname || $2 || relname AS relname , idx_scan, idx_tup_read, idx_tup_fetch, idx_blks_read, idx_blks_hit
FROM pg_stat_user_indexes psai, LATERAL (SELECT idx_blks_read, idx_blks_hit FROM pg_statio_user_indexes psio WHERE psio.indexrelid = psai.indexrelid) p2
ORDER BY idx_tup_read DESC LIMIT $3
Users
{dbuser_monitor}
Exec Time
Value
μ
10.9 ms
σ
1.06 ms
min
0.862 ms
max
12.5 ms
sum
4.58 min
Metric
plans
calls
rows
shared_blks_hit
shared_blks_read
shared_blks_dirtied
shared_blks_written
local_blks_hit
local_blks_read
local_blks_dirtied
local_blks_written
temp_blks_read
temp_blks_written
blk_read_time
blk_write_time
wal_records
wal_fpi
wal_bytes
Plan Time
Value
μ
σ
min
max
sum
QPS
RT
Statements Stat for -3288231039287227845
User
calls
rows
plans
total_plan_time
min_plan_time
max_plan_time
mean_plan_time
stddev_plan_tim
total_exec_time
min_exec_time
max_exec_time
dbuser_monitor 25.1 K 1.36 Mil 0 0 ms 0 ms 0 ms 0 ms 0 ms 4.6 min 0.9 ms 12.5 ms
Overview
Statements
Query
SQL
calls +
rows
total_exec
min_exec_
max_exec
mean_exe
blk_read_t
blk_write_
wal_records
7226155749781812199 SELECT current_catalog, current_user, ... 1.54 Mil 1.54 Mil 1.3 min 0.0 ms 0.2 ms 0.1 ms 0.0 ms 0 ms 1.0
5760961677018624247 SELECT datid, datname, numbackends, xa... 679 K 679 K 1.9 min 0.1 ms 10.5 ms 0.2 ms 0 ms 0 ms 0
-1903945346303072773 SELECT datname, pid, relid::RegClass A... 679 K 0 15.3 s 0.0 ms 0.3 ms 0.0 ms 0 ms 0 ms 0
5845664379928311205 select schemaname,relname,indexrelname... 121 4.24 K 1.3 s 0.7 ms 11.0 ms 10.4 ms 0 ms 0 ms 0
8242127240162932682037 SELECT count(*) AS cnt FROM yay.secret... 10 10 9.6 s 852.9 ms 1.3 s 957.2 ms 895.1 ms 0 ms 10.0 K
-2260398869991494884 SELECT query FROM monitor.pg_stat_stat... 5 5 2.3 ms 0.4 ms 0.5 ms 0.5 ms 0 ms 0 ms 0
7071073393163099576 SELECT queryid::TEXT AS qid, * FROM mo... 4 40 2.8 ms 0.4 ms 0.8 ms 0.7 ms 0 ms 0 ms 0
-9428308577959174751 SELECT c2.relname, i.indisprimary, i.i... 2 2 14.7 ms 0.3 ms 14.3 ms 7.3 ms 0 ms 0 ms 0
2376837088859628503 SELECT * FROM pg_sequences 2 32 1.0 ms 0.2 ms 0.8 ms 0.5 ms 0 ms 0 ms 0
-2941126072218699949 SELECT count(*) AS cnt FROM sms_promot... 2 2 51.5 s 25.6 s 25.9 s 25.7 s 16.7 s 0 ms 0

PGLOG Analysis

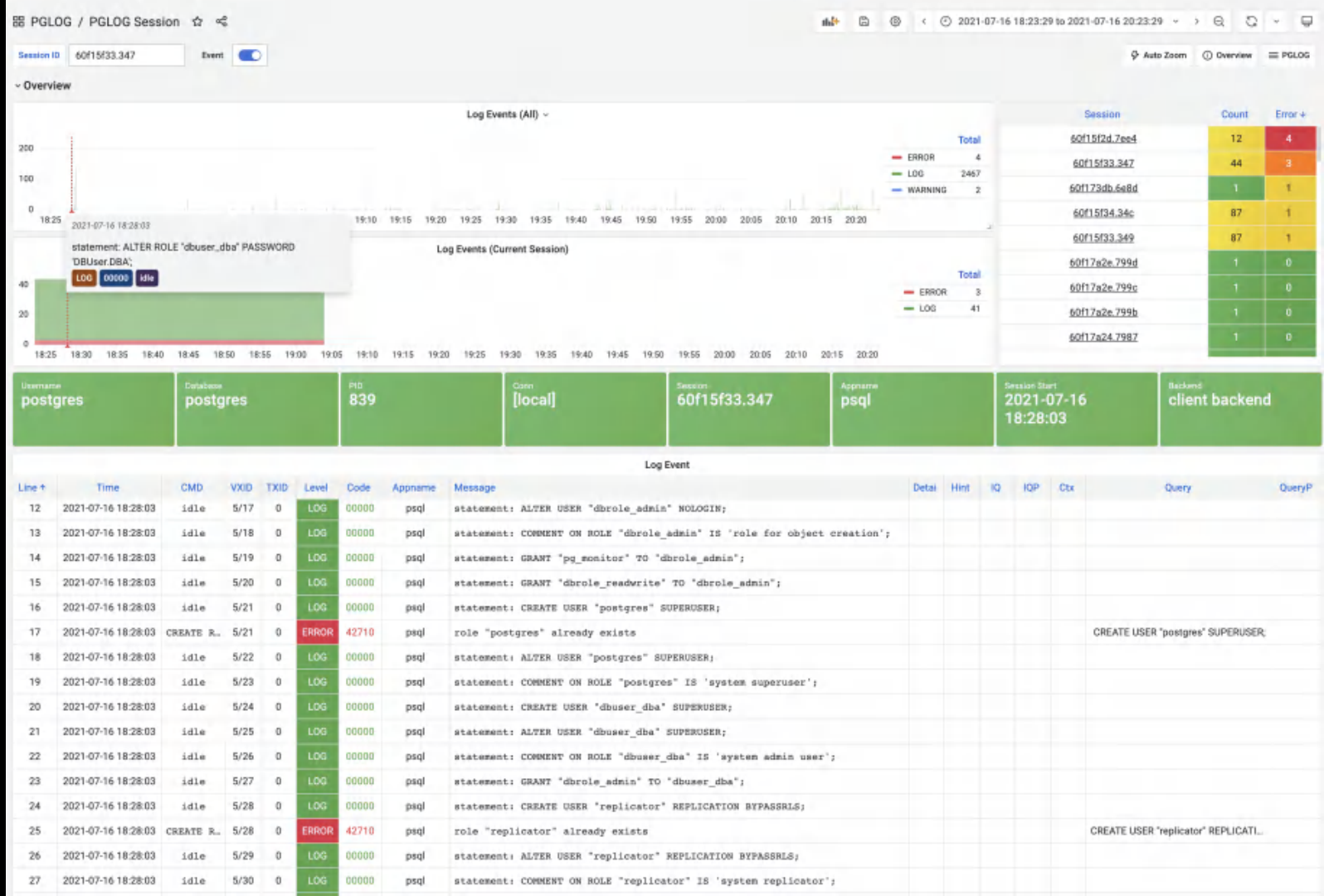
Run on meta node as admin user to fetch and pour log into sample table: `pglog.sample`

```
catlog | pglog                    # get local (meta0b) today's log
catlog node-1 '2021-07-15' | pglog  # get node-1's csvlog @ 2021-07-15
```

Time	Username	Database	PID	Conn	Session	Line	CMD	Session Start	VXID	TXID	Level	Code	Appname	Backend	Message	Detail	Hint	IO
2021-07-16 19:55:59	dbuser_meta	meta	26916	[local]	60f17140.6924	34	UPDATE waiting	2021-07-16 19:45:04	17/197953	1274644	LOG	00000	pgbench - 10.10.10.10:53834	client backend	process 26916 still waiting for ShareLock on transaction 1274646 after 73.935 ms	Pro…		
2021-07-16 19:55:59	dbuser_meta	meta	27278	[local]	60f17113.6a8e	23	UPDATE waiting	2021-07-16 19:48:03	11/190758	1274645	LOG	00000	pgbench - 10.10.10.10:53856	client backend	process 27278 still waiting for ExclusiveLock on tuple (214,145) of relation 21553 of database 19344 after 74.262 ms	Pro…		
2021-07-16 19:55:59	dbuser_meta	meta	26916	[local]	60f17140.6924	35	UPDATE waiting	2021-07-16 19:45:04	17/197953	1274644	LOG	00000	pgbench - 10.10.10.10:53834	client backend	process 26916 acquired ShareLock on transaction 1274646 after 75.095 ms			
2021-07-16 19:55:59	dbuser_meta	meta	27278	[local]	60f17113.6a8e	24	UPDATE waiting	2021-07-16 19:48:03	11/190758	1274645	LOG	00000	pgbench - 10.10.10.10:53856	client backend	process 27278 acquired ExclusiveLock on tuple (214,145) of relation 21553 of database 19344 after 96.935 ms			
2021-07-16 19:55:59	dbuser_meta	meta	27278	[local]	60f17113.6a8e	25	UPDATE	2021-07-16 19:48:03	11/190758	1274645	LOG	00000	pgbench - 10.10.10.10:53856	client backend	duration: 124.700 ms statement: UPDATE pgbench_branches SET bbalance = bbalance + -1775 WHERE bid = 7;			
2021-07-16 19:55:59	dbuser_meta	meta	25823	[local]	60f17109.649f	28	UPDATE waiting	2021-07-16 19:44:09	6/39657	1274663	LOG	00000	pgbench - 10.10.10.10:53868	client backend	process 25823 still waiting for ShareLock on transaction 1274666 after 50.073 ms	Pro…		
2021-07-16 19:56:00	dbuser_meta	meta	25823	[local]	60f17109.649f	29	UPDATE waiting	2021-07-16 19:44:09	6/39657	1274663	LOG	00000	pgbench - 10.10.10.10:53868	client backend	process 25823 acquired ShareLock on transaction 1274666 after 71.853 ms			
2021-07-16 19:56:03	dbuser_meta	meta	25823	[local]	60f17109.649f	30	UPDATE waiting	2021-07-16 19:44:09	6/39766	1275701	LOG	00000	pgbench - 10.10.10.10:53866	client backend	process 25823 still waiting for ShareLock on transaction 1275696 after 50.584 ms	Pro…		
2021-07-16 19:56:03	dbuser_meta	meta	25823	[local]	60f17109.649f	31	UPDATE waiting	2021-07-16 19:44:09	6/39766	1275701	LOG	00000	pgbench - 10.10.10.10:53866	client backend	process 25823 acquired ShareLock on transaction 1275696 after 74.148 ms			
2021-07-16 19:56:05			28276		60f173ce.6e74	1	null	2021-07-16 19:55:58	19/1523	0	LOG	00000		autovacuum …	automatic vacuum of table "meta.public.pgbench_accounts": index scans: 1 pages: 0 removed, 17443 remain, 0 skipped due to pins, 0 skipped froz…			
2021-07-16 19:56:05	dbuser_meta	meta	27443	[local]	60f17259.6b33	21	UPDATE waiting	2021-07-16 19:49:45	15/206925	1276475	LOG	00000	pgbench - 10.10.10.10:53864	client backend	process 27443 still waiting for ShareLock on transaction 1276461 after 50.408 ms	Pro…		
2021-07-16 19:56:05	dbuser_meta	meta	27443	[local]	60f17259.6b33	22	UPDATE waiting	2021-07-16 19:49:45	15/206925	1276475	LOG	00000	pgbench - 10.10.10.10:53864	client backend	process 27443 acquired ShareLock on transaction 1276461 after 51.177 ms			
2021-07-16 19:56:11	dbuser_monit…	postgres	28301	[local]	60f173db.6e8d	1	SELECT	2021-07-16 19:56:11	5/7317	0	ERROR	57014		client backend	canceling statement due to user request			
2021-07-16 19:56:13	dbuser_meta	meta	27643	[local]	60f172ca.6bfb	11	UPDATE waiting	2021-07-16 19:51:38	12/195977	1278645	LOG	00000	pgbench - 10.10.10.10:53932	client backend	process 27643 still waiting for ShareLock on transaction 1278644 after 50.239 ms	Pro…		
2021-07-16 19:56:13	dbuser_meta	meta	27643	[local]	60f172ca.6bfb	12	UPDATE waiting	2021-07-16 19:51:38	12/195977	1278645	LOG	00000	pgbench - 10.10.10.10:53932	client backend	process 27643 acquired ShareLock on transaction 1278644 after 53.496 ms			
2021-07-16 19:56:25	dbuser_meta	meta	27038	[local]	60f17188.699e	23	UPDATE waiting	2021-07-16 19:46:16	9/209098	1282595	LOG	00000	pgbench - 10.10.10.10:54072	client backend	process 27038 still waiting for ShareLock on transaction 1282593 after 50.165 ms	Pro…		
2021-07-16 19:56:25	dbuser_meta	meta	27038	[local]	60f17188.699e	24	UPDATE waiting	2021-07-16 19:46:16	9/209098	1282595	LOG	00000	pgbench - 10.10.10.10:54072	client backend	process 27038 acquired ShareLock on transaction 1282593 after 52.482 ms			

PGLOG Session

- 单条连接详情
- Annotation



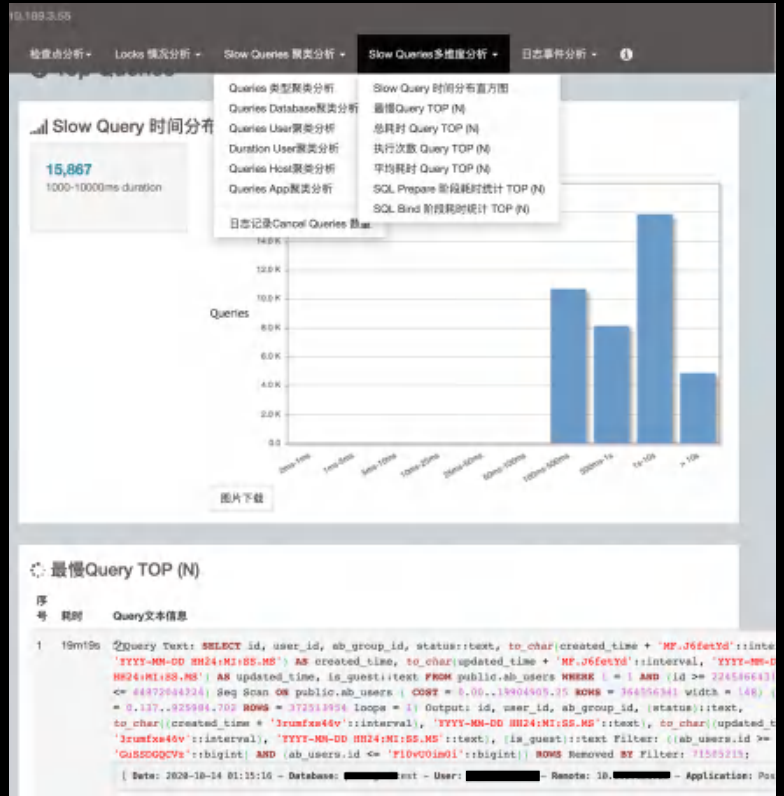
Line	Time	CMD	VXID	TXID	Level	Code	Appname	Message	Detai	Hint	IQ	IQP	Ctx	Query	QueryP
12	2021-07-16 18:28:03	idle	5/17	0	LOG	00000	psql	statement: ALTER USER "dbrole_admin" NOLOGIN;							
13	2021-07-16 18:28:03	idle	5/18	0	LOG	00000	psql	statement: COMMENT ON ROLE "dbrole_admin" IS 'role for object creation';							
14	2021-07-16 18:28:03	idle	5/19	0	LOG	00000	psql	statement: GRANT "pg_monitor" TO "dbrole_admin";							
15	2021-07-16 18:28:03	idle	5/20	0	LOG	00000	psql	statement: GRANT "dbrole_readwrite" TO "dbrole_admin";							
16	2021-07-16 18:28:03	idle	5/21	0	LOG	00000	psql	statement: CREATE USER "postgres" SUPERUSER;							
17	2021-07-16 18:28:03	CREATE R...	5/21	0	ERROR	42710	psql	role "postgres" already exists						CREATE USER "postgres" SUPERUSER;	
18	2021-07-16 18:28:03	idle	5/22	0	LOG	00000	psql	statement: ALTER USER "postgres" SUPERUSER;							
19	2021-07-16 18:28:03	idle	5/23	0	LOG	00000	psql	statement: COMMENT ON ROLE "postgres" IS 'system superuser';							
20	2021-07-16 18:28:03	idle	5/24	0	LOG	00000	psql	statement: CREATE USER "dbuser_dba" SUPERUSER;							
21	2021-07-16 18:28:03	idle	5/25	0	LOG	00000	psql	statement: ALTER USER "dbuser_dba" SUPERUSER;							
22	2021-07-16 18:28:03	idle	5/26	0	LOG	00000	psql	statement: COMMENT ON ROLE "dbuser_dba" IS 'system admin user';							
23	2021-07-16 18:28:03	idle	5/27	0	LOG	00000	psql	statement: GRANT "dbrole_admin" TO "dbuser_dba";							
24	2021-07-16 18:28:03	idle	5/28	0	LOG	00000	psql	statement: CREATE USER "replicator" REPLICATION BYPASSRLS;							
25	2021-07-16 18:28:03	CREATE R...	5/28	0	ERROR	42710	psql	role "replicator" already exists						CREATE USER "replicator" REPLICATI...	
26	2021-07-16 18:28:03	idle	5/29	0	LOG	00000	psql	statement: ALTER USER "replicator" REPLICATION BYPASSRLS;							
27	2021-07-16 18:28:03	idle	5/30	0	LOG	00000	psql	statement: COMMENT ON ROLE "replicator" IS 'system replicator';							

PG Instance Log

- Loki + Promtail
- Realtime log metrics
- Full-text search

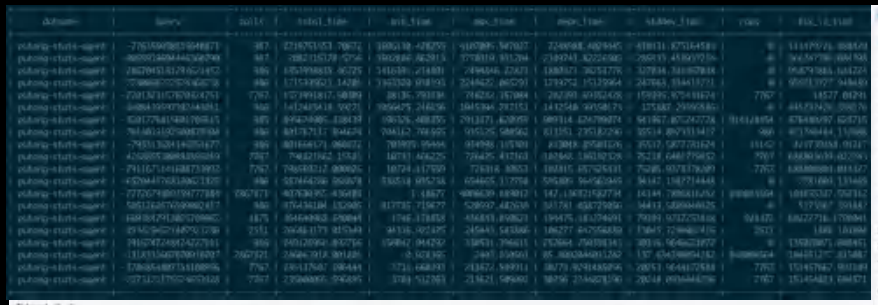
Log Events

Summary based on pgbadger

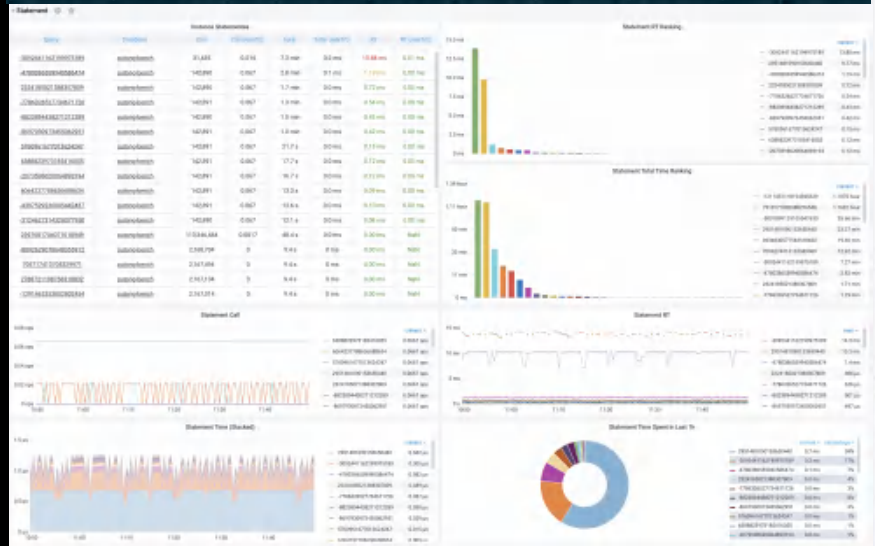


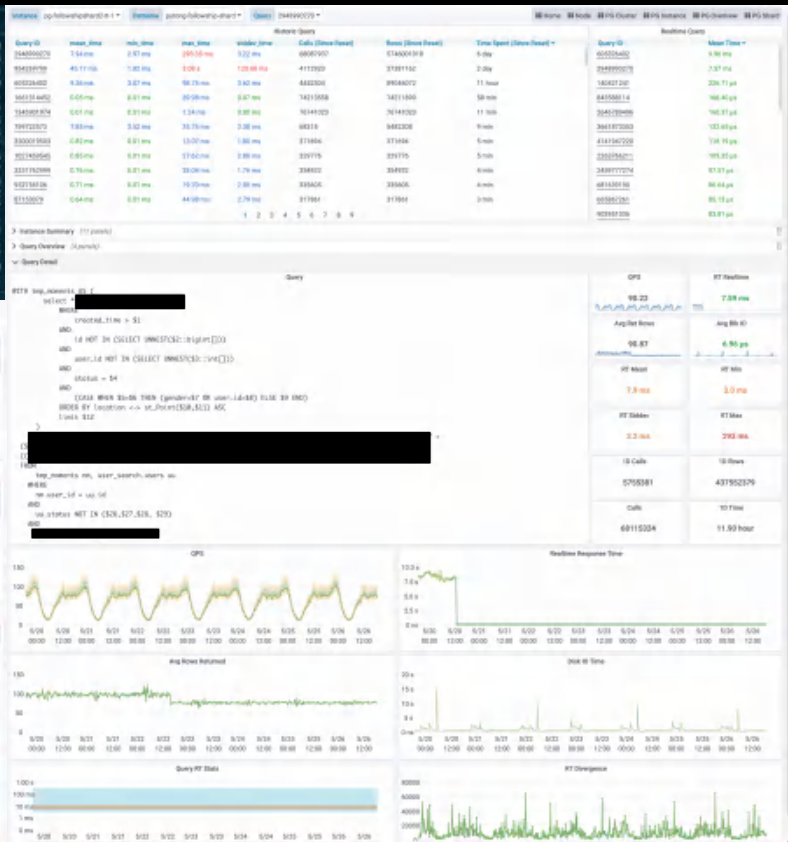
Observability

Original Statements Metrics



Query Statements Dashboards





Postgres Observability

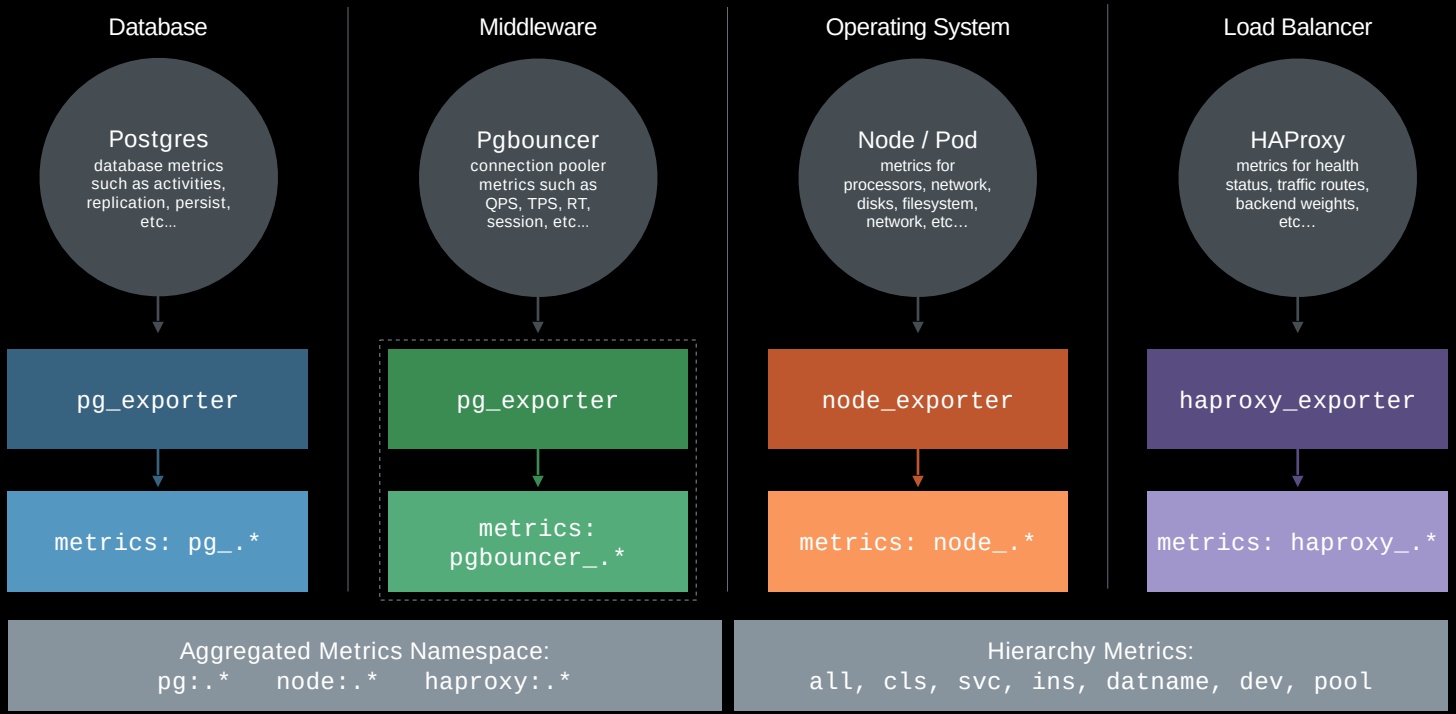
PG 14

new metrics are

READY!

Metrics Source

metrics data flow

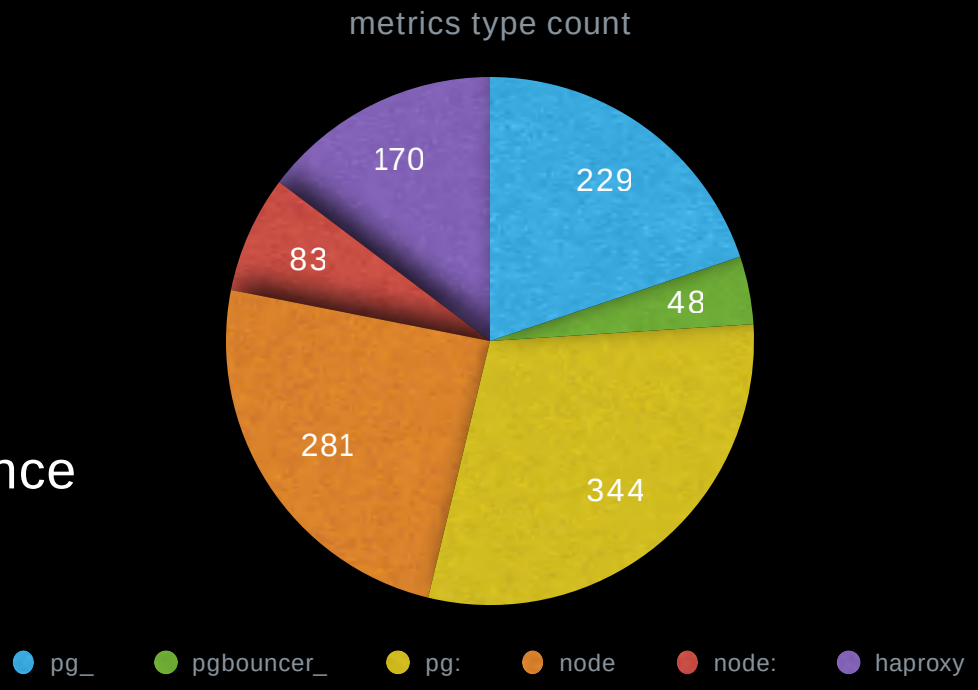


Metrics Ratio

metric type for a cluster

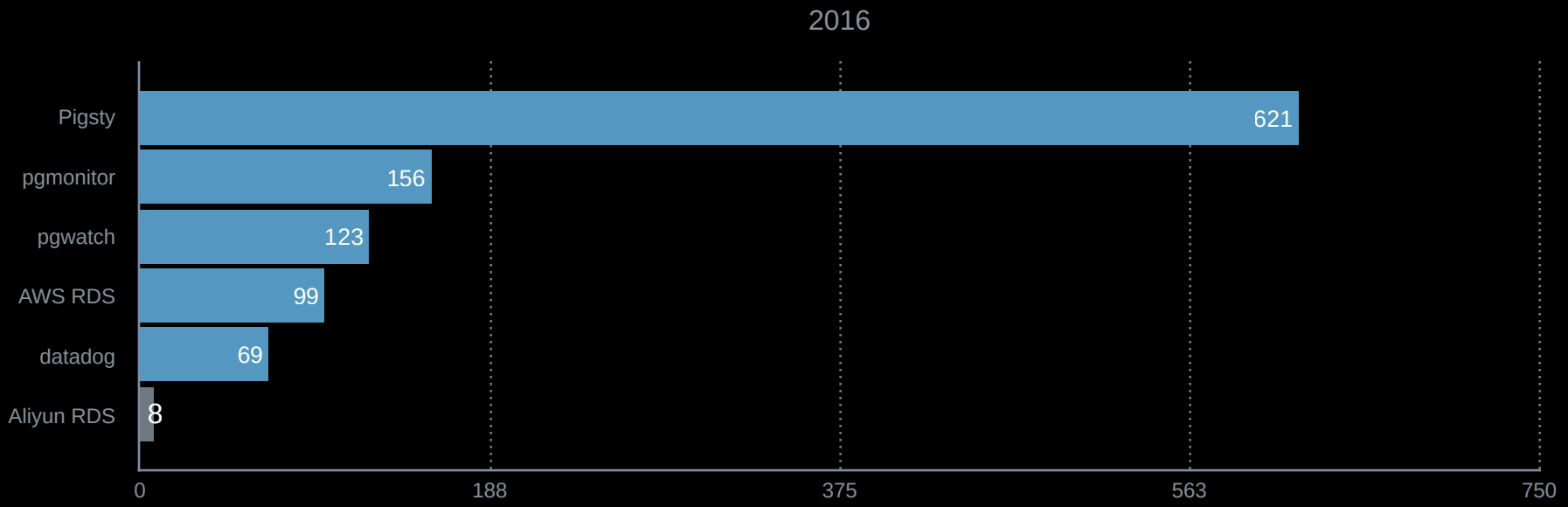
1155 metrics per env

3k~10k+ time-series per instance



Metrics Coverage

metrics **numbers** comparing to others



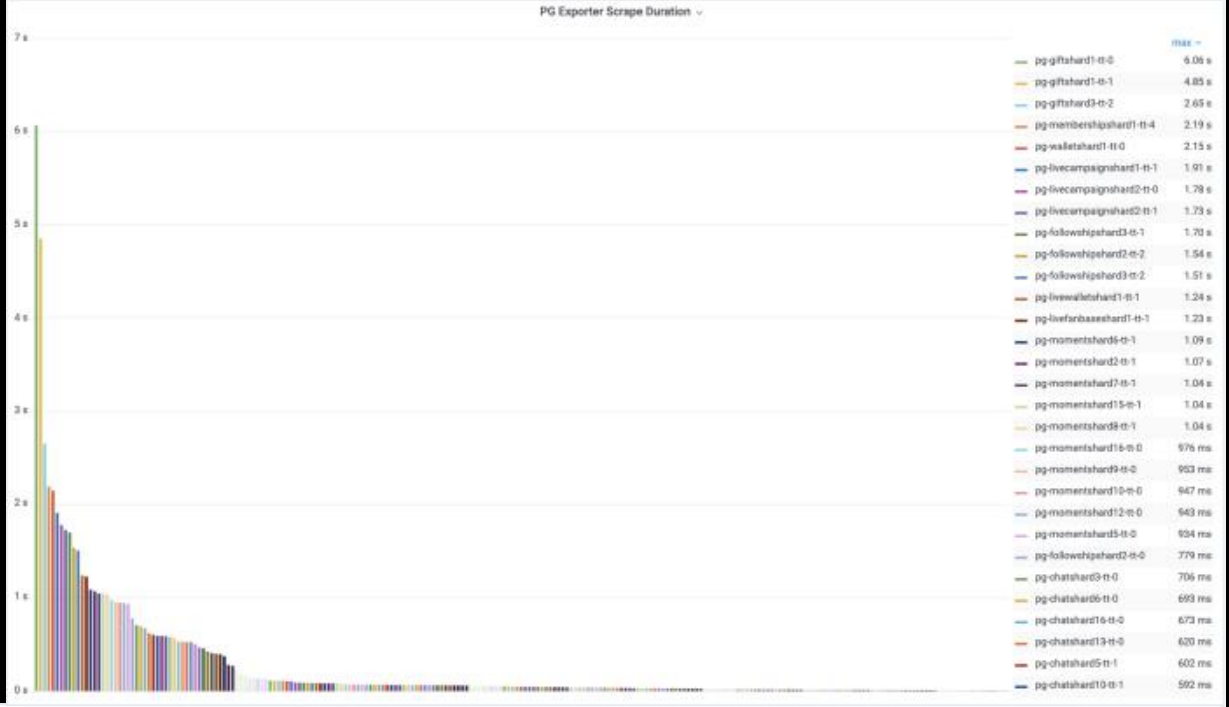
metrics related to database are counted (i.e: node metrics are excluded)

Overhead

Typical Scrape Duration: 10~100ms

Typical Scrape Count: 1500 ~ 4500

Typical Scrape Interval: 15s



Provisioning

Clearly, we must break away from the sequential and not limit the computers. We must state definitions and provide for priorities and descriptions of data. We must state relation- ships, not procedures. —Grace Murray Hopper, *Management and the Computer of the Future* (1962)

ONE Node TO RULE

THEM ALL

Interface

Clearly, we must break away from the sequential and not limit the computers. We must state definitions and provide for priorities and descriptions of data. We must state relation- ships, not procedures.

—Grace Murray Hopper, *Management and the Computer of the Future* (1962)

我们必须跳出电脑指令序列的窠臼。 叙述定义、描述元数据、梳理关系，而不是编写过程。

—— Grace Murray Hopper，未来的计算机及其管理（1962）

管控

数据库是管理数据的软件，管控系统是管理数据库的软件

DBMS，包括数据库内核，和配套软件。

云厂商卖开源PG，卖的不是内核，而是 数据库管控能力

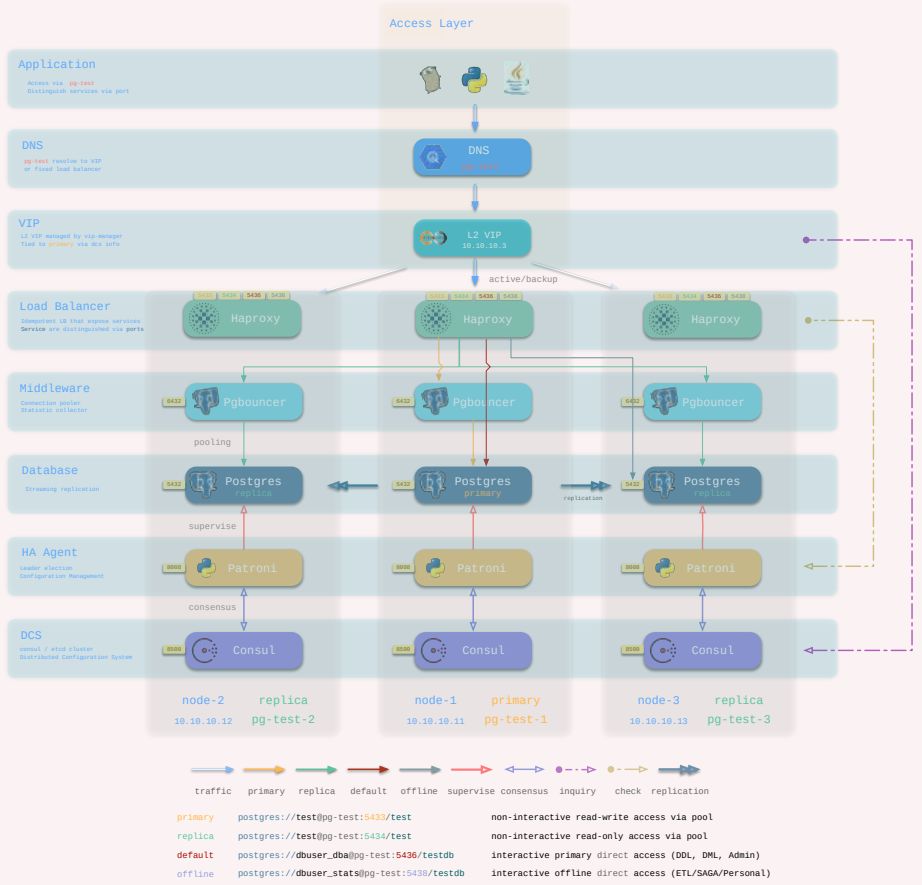
Provisioning

定义一套集群，只要几行代码

```yaml
pg-test:    # 定义一个新的 pg-test 集群
  hosts:    # 包含有一主两从三个实例，部署在三个节点上
    10.10.10.11: { pg_seq: 1, pg_role: primary }
    10.10.10.12: { pg_seq: 2, pg_role: replica }
    10.10.10.13: { pg_seq: 3, pg_role: replica }
  vars:     # 集群名称，使用的VIP地址，业务用户与数据库
    pg_cluster: pg-test
    vip_address: 10.10.10.2
    pg_users: [ { name: test } ]
    pg_databases: [ { name: test } ]
```

从配置到实现，只要一行命令

```
pgsql.yml -l pg-test      # 创建出该集群
```



Interface

Idempotent Ansible Playbooks

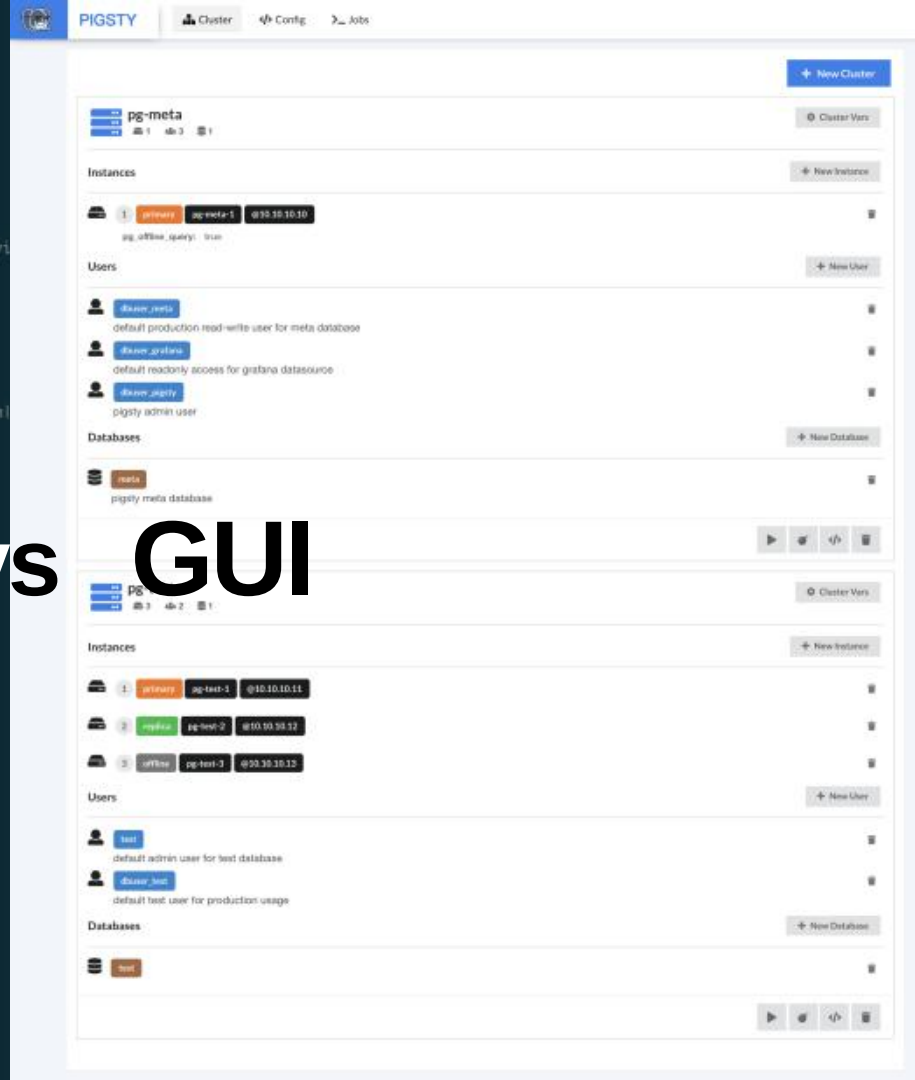
```
infra.yml                 # 部署/修改基础设施
pgsql.yml                 # 部署/修改数据库集群/实例
pgsql-remove.yml          # 下线数据库集群/实例
pgsql-createuser.yml      # 创建/修改业务角色&用户
pgsql-createdb.yml        # 创建/修改业务数据库
```

```
./infra.yml --tags=environ                      # 重新在管理节点上配置环境
./infra.yml --tags=repo -e repo_rebuild=true    # 强制重新创建本地源
./infra.yml --tags=repo_upstream                # 加入上游YumRepo
./infra.yml --tags=prometheus                   # 重新创建Prometheus
./infra.yml --tags=nginx_config,nginx_restart   # 重新生成Nginx配置文件

# 基础设施初始化
./pgsql.yml --tags=infra        # 完成基础设施的初始化，包括机器节点初始化与DCS部署

./pgsql.yml --tags=node         # 完成机器节点的初始化，通常不会影响运行中数据库实例
./pgsql.yml --tags=dcs          # 完成DCS：consul/etcd的初始化
./pgsql.yml --tags=dcs -e dcs_exists_action # 完成consul/etcd的初始化，强制抹除

# 数据库初始化
./pgsql.yml --tags=pgsql        # 完成数据库部署：数据库、监控、服务

./pgsql.yml --tags=postgres     # 完成数据库部署
./pgsql.yml --tags=monitor      # 完成监控的部署
./pgsql.yml --tags=service      # 完成负载均衡的部署，（Haproxy & VIP）
./pgsql.yml --tags=register     # 将服务注册至基础设施
```

```
#-----------------------------------------------------------------
- name: Infra Init      # init infra on common database node
  become: yes
  hosts: all
  gather_facts: no
  tags: infra
  roles:

    - role: node        # init common database node
      tags: node

    - role: consul      # init dcs:consul clients
      tags: [ dcs , consul ]

#-----------------------------------------------------------------
- name: Pgsql Init      # init postgres cluster/instance
  become: yes
  hosts: all
  gather_facts: no
  tags: pgsql
  roles:

    - role: postgres    # init postgres pgbouncer patroni
      tags: postgres

    - role: monitor     # init monitor exporters
      tags: monitor

    - role: service     # init service , lb , vip
      tags: service

    - role: register    # register cluster/instance to infra
      tags: register

#-----------------------------------------------------------------
```

```
[05-23 16:53:03 CST] 24180 vonng@vonng-mac:~/pigsty  master [+!?]
$ ./pigsty

NAME
    pigsty -- Pigsty Command-Line Interface v0.9

SYNOPSIS

    meta              setup meta nodes           init|fetch|repo|cache|ansible
    node              setup database nodes       init|tune|dcs|remove|ping|bash|ssh|admin
    pgsql             setup postgres clusters    init|node|dcs|postgres|template|business|monitor|servi
    infra             setup infrastructure       init|ca|dns|prometheus|grafana|loki|haproxy|target
    clean             clean pgsql clusters       all|service|monitor|postgres|dcs
    config            mange pigsty config file   init|edit|info|dump|path
    serve             run pigsty API server      init|start|stop|restart|reload|status
    demo              setup local demo           init|up|new|clean|start|dns
    log               watch system log           query|postgres|patroni|pgbouncer|message
    pg                pg operational tasks       user|db|svc|hba|log|psql|deploy|backup|restore|vacuum|

EXAMPLES

    1. infra summary
       pigsty infra

    2. pgsql clusters summary
       pigsty pgsql

    3. pigsty nodes summary
       pigsty node

    4. create pgsql cluster 'pg-test'
       pigsty pgsql init -l pg-test

    5. add new instance 10.10.10.13 of cluster 'pg-test'
       pigsty pgsql init -l 10.10.10.13

    6. remove cluster 'pg-test'
       pigsty clean -l pg-test

    7. create user dbuser_vonng on cluster 'pg-test'
       pigsty pg user dbuser_vonng -l pg-test

    8. create database test2 on cluster 'pg-test'
       pigsty pg db test -l pg-test
```

CLI vs GUI



Completeness

160+配置参数，全面定制数据库与基础设施 （可以不用，不能没有）

配置

Pigsty提供的配置参数与定制选项

Pigsty采用声明式配置：用户配置描述状态，而Pigsty负责将真实组件调整至所期待的状态。

Pigsty配置文件遵循Ansible规则，采用YAML格式，详见配置文件 。

Pigsty包含了168个配置项，分为十类五级，详见配置项。

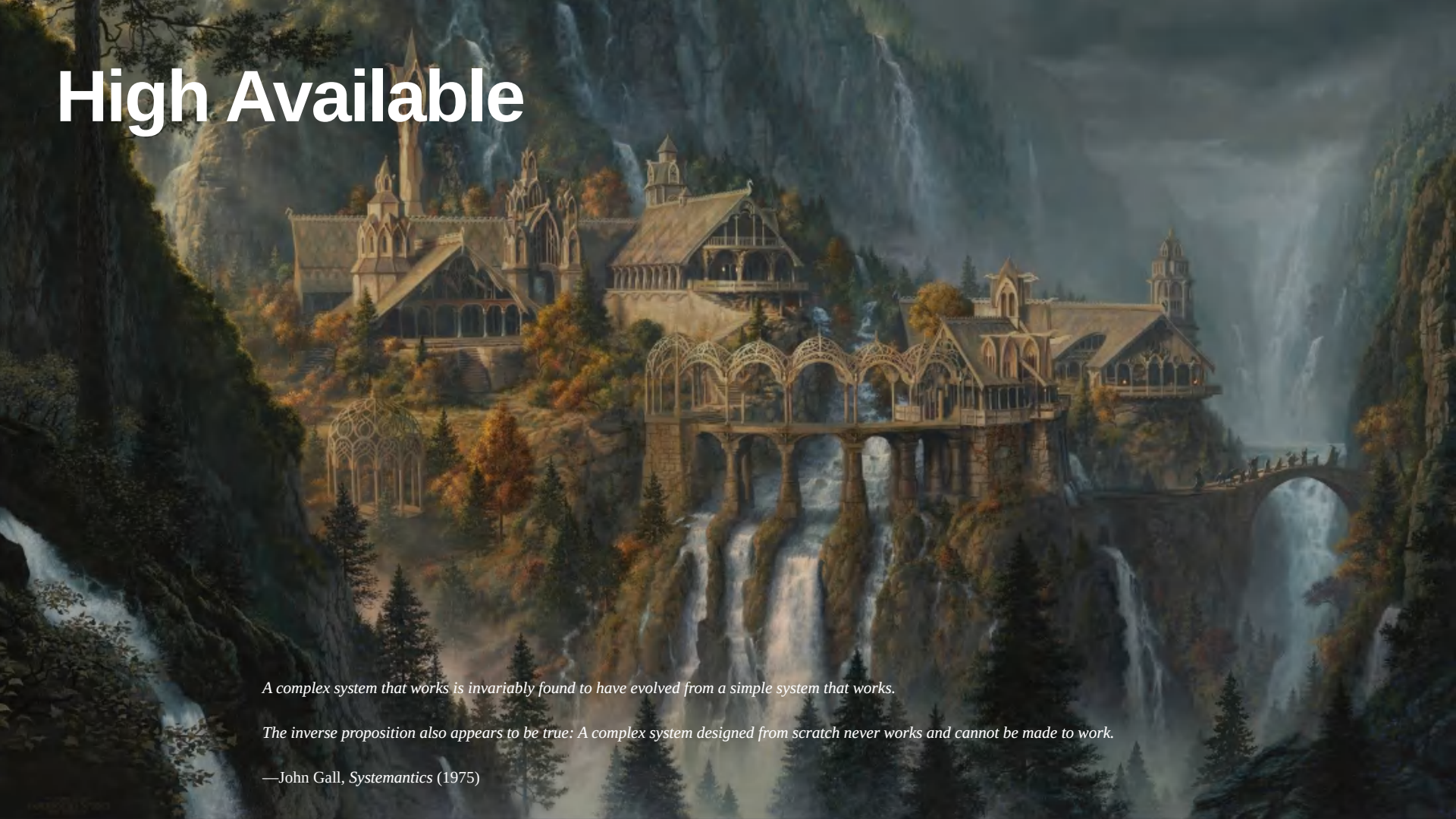
绝大多数配置参数无需修改，可直接使用默认值：定义新数据库集群只有三个必选身份参数。

No	类目	英文	大类	数量	功能
1	连接参数	connect	基础设施	1	代理服务器配置，管理对象的连接信息
2	本地仓库	repo	基础设施	11	定制本地Yum源，离线安装包
3	节点供给	node	基础设施	30	在普通节点上配置基础设施
4	基础设施	meta	基础设施	23	在元节点上安装启用基础设施服务
5	元数据库	dcs	基础设施	8	在所有节点上配置DCS服务（consul/etcd）
6	PG安装	pg-install	数据库	11	安装PostgreSQL数据库
7	PG供给	pg-provision	数据库	30	拉起PostgreSQL数据库集群
8	PG模板	pg-template	数据库	19	定制PostgreSQL数据库内容
9	监控系统	monitor	数据库	18	安装Pigsty数据库监控系统
10	服务供给	service	数据库	17	通过Haproxy或VIP对外暴露数据库服务

```
#-----------------------------------------------------------------
# POSTGRES TEMPLATE
#-----------------------------------------------------------------
# - template - #
pg_init: pg-init                        # init script for cluster template

# - system roles - #
pg_replication_username: replicator     # system replication user
pg_replication_password: DBUser.Replicator  # system replication password
pg_monitor_username: dbuser_monitor     # system monitor user
pg_monitor_password: DBUser.Monitor     # system monitor password
pg_admin_username: dbuser_admin         # system admin user
pg_admin_password: DBUser.Admin         # system admin password

# - default roles - #
# check http://pigsty.cc/zh/docs/concepts/provision/acl/ for more detail
pg_default_roles:

  # common production readonly user
  - name: dbrole_readonly               # production read-only roles
    login: false
    comment: role for global readonly access

  # common production read-write user
  - name: dbrole_readwrite              # production read-write roles
    login: false
    roles: [dbrole_readonly]            # read-write includes read-only access
    comment: role for global read-write access

  # offline have same privileges as readonly, but with limited hba access on offline instance
  # for the purpose of running slow queries, interactive queries and perform ETL tasks
  - name: dbrole_offline
    login: false
    comment: role for restricted read-only access (offline instance)

  # admin have the privileges to issue DDL changes
  - name: dbrole_admin
    login: false
    bypassrls: true
    comment: role for object creation
    roles: [dbrole_readwrite,pg_monitor,pg_signal_backend]

  # dbsu, name is designated by `pg_dbsu`. It's not recommend to set password for dbsu
  - name: postgres
    superuser: true
    comment: system superuser

  # default replication user, name is designated by `pg_replication_username`, and password is
  - name: replicator
    replication: true
    roles: [pg_monitor, dbrole_readonly]
    comment: system replicator

  # default replication user, name is designated by `pg_monitor_username`, and password is set
  - name: dbuser_monitor
    connlimit: 16
    comment: system monitor user
    roles: [pg_monitor, dbrole_readonly]

  # default admin user, name is designated by `pg_admin_username`, and password is set by `pg_
  - name: dbuser_admin
    bypassrls: true
    comment: system admin user
    roles: [dbrole_admin]

  # default stats user, for ETL and slow queries
  - name: dbuser_stats
    password: DBUser.Stats
    comment: business offline user for offline queries and ETL
    roles: [dbrole_offline]
```

```
# - privileges - #
# object created by dbsu and admin will have their privileges properly set
pg_default_privileges:
  - GRANT USAGE                         ON SCHEMAS   TO dbrole_readonly
  - GRANT SELECT                        ON TABLES    TO dbrole_readonly
  - GRANT SELECT                        ON SEQUENCES TO dbrole_readonly
  - GRANT EXECUTE                       ON FUNCTIONS TO dbrole_readonly
  - GRANT USAGE                         ON SCHEMAS   TO dbrole_offline
  - GRANT SELECT                        ON TABLES    TO dbrole_offline
  - GRANT SELECT                        ON SEQUENCES TO dbrole_offline
  - GRANT EXECUTE                       ON FUNCTIONS TO dbrole_offline
  - GRANT INSERT, UPDATE, DELETE        ON TABLES    TO dbrole_readwrite
  - GRANT USAGE,  UPDATE                ON SEQUENCES TO dbrole_readwrite
  - GRANT TRUNCATE, REFERENCES, TRIGGER ON TABLES    TO dbrole_admin
  - GRANT CREATE                        ON SCHEMAS   TO dbrole_admin

# - schemas - #
pg_default_schemas: [monitor]           # default schemas to be created

# - extension - #
pg_default_extensions:                  # default extensions to be created
  - { name: 'pg_stat_statements',  schema: 'monitor' }
  - { name: 'pgstattuple',         schema: 'monitor' }
  - { name: 'pg_qualstats',        schema: 'monitor' }
  - { name: 'pg_buffercache',      schema: 'monitor' }
  - { name: 'pageinspect',         schema: 'monitor' }
  - { name: 'pg_prewarm',          schema: 'monitor' }
  - { name: 'pg_visibility',       schema: 'monitor' }
  - { name: 'pg_freespacemap',     schema: 'monitor' }
  - { name: 'pg_repack',           schema: 'monitor' }
  - name: postgres_fdw
  - name: file_fdw
  - name: btree_gist
  - name: btree_gin
  - name: pg_trgm
  - name: intagg
  - name: intarray

# - hba - #
pg_offline_query: false                 # set to true to enable offline query on instance
pg_hba_rules:                           # postgres host-based authentication rules
  - title: allow meta node password access
    role: common
    rules:
      - host    all     all                     10.10.10.10/32      md5

  - title: allow intranet admin password access
    role: common
    rules:
      - host    all     +dbrole_admin           10.0.0.0/8          md5
      - host    all     +dbrole_admin           172.16.0.0/12       md5
      - host    all     +dbrole_admin           192.168.0.0/16      md5

  - title: allow intranet password access
    role: common
    rules:
      - host    all     all                     10.0.0.0/8          md5
      - host    all     all                     172.16.0.0/12       md5
      - host    all     all                     192.168.0.0/16      md5

  - title: allow local read/write (local production user via pgbouncer)
    role: common
    rules:
      - local   all     +dbrole_readonly                            md5
      - host    all     +dbrole_readonly        127.0.0.1/32        md5

  - title: allow offline query (ETL,SAGA,Interactive) on offline instance
    role: offline
    rules:
      - host    all     +dbrole_offline           10.0.0.0/8        md5
      - host    all     +dbrole_offline           172.16.0.0/12     md5
      - host    all     +dbrole_offline           192.168.0.0/16    md5

pg_hba_rules_extra: []                  # extra hba rules (for cluster/instance overwrite)

pgbouncer_hba_rules:                    # pgbouncer host-based authentication rules
  - title: local password access
    role: common
    rules:
      - local  all          all                             md5
      - host   all          all                 127.0.0.1/32  md5

  - title: intranet password access
    role: common
    rules:
      - host   all          all                 10.0.0.0/8    md5
      - host   all          all                 172.16.0.0/12 md5
      - host   all          all                 192.168.0.0/16 md5

pgbouncer_hba_rules_extra: []           # extra pgbouncer hba rules (for cluster/instance overwrite)
```

High Available



A complex system that works is invariably found to have evolved from a simple system that works.

The inverse proposition also appears to be true: A complex system designed from scratch never works and cannot be made to work.

—John Gall, *Systemantics* (1975)

Architecture

A complex system that works is invariably found to have evolved from a simple system that works.

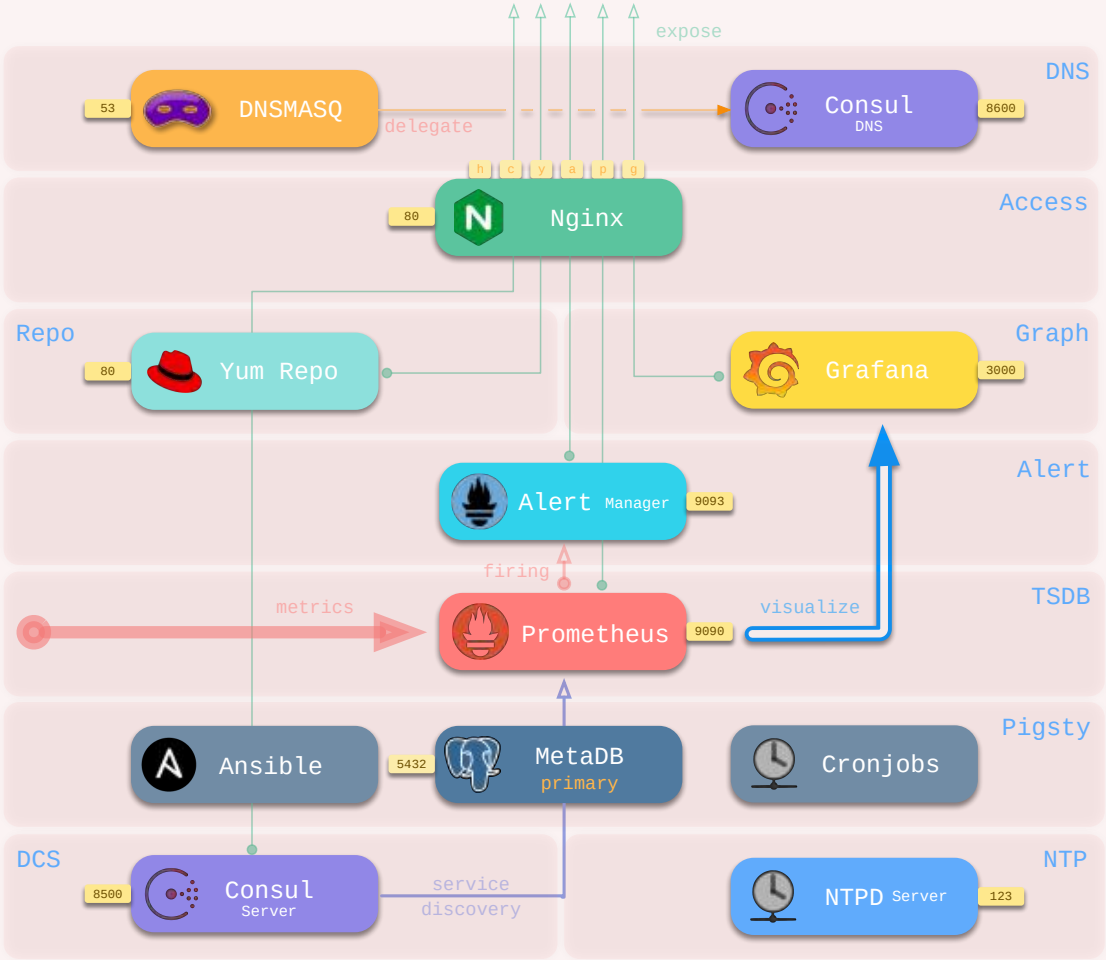
The inverse proposition also appears to be true: A complex system designed from scratch never works and cannot be made to work.

—John Gall, *Systemantics* (1975)

Infrastructure

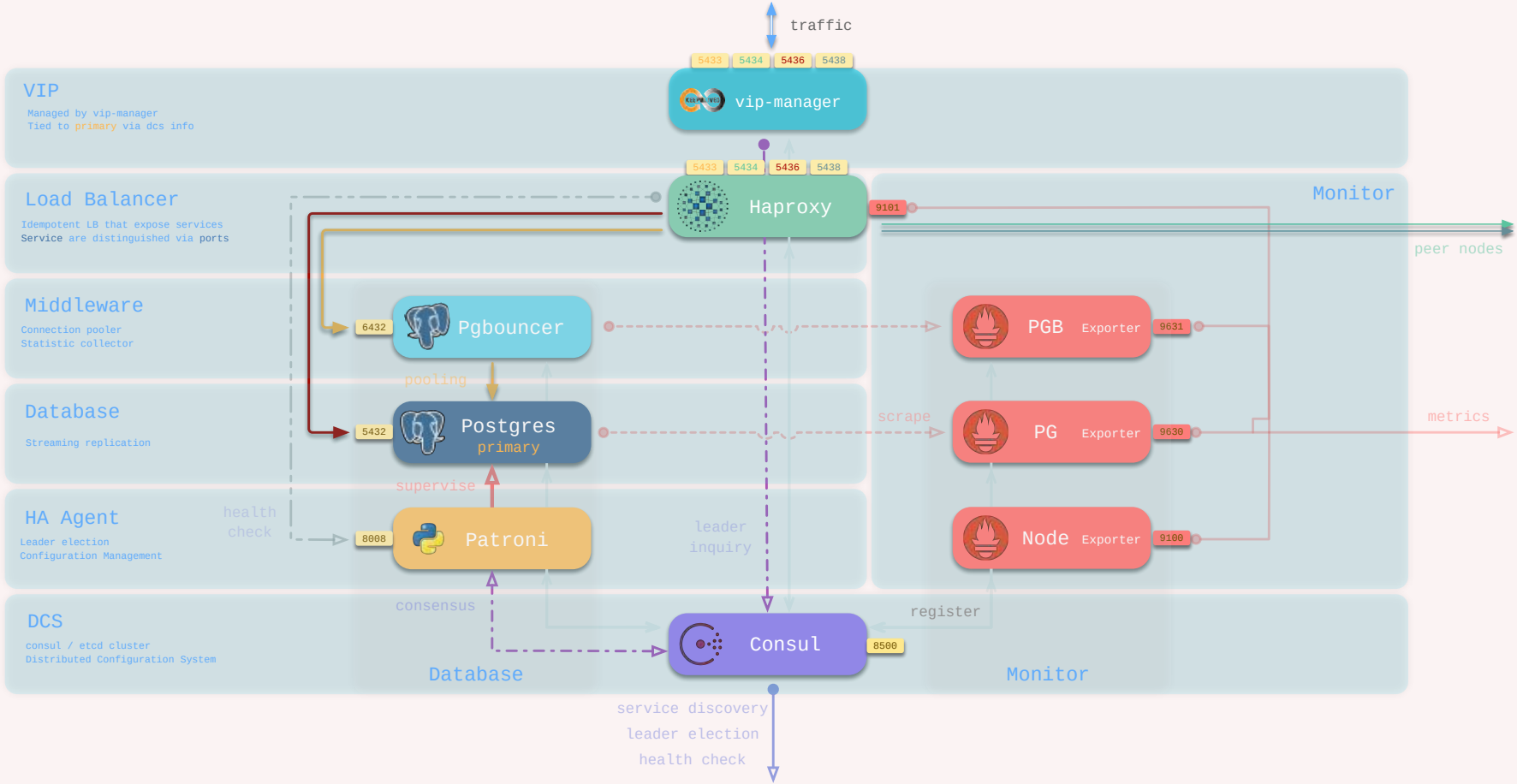
组件	端口	默认域名	说明
Grafana	3000	g.pigsty	Pigsty监控系统图形界面
Prometheus	9090	p.pigsty	监控时序数据库
AlertManager	9093	a.pigsty	报警聚合管理组件
Consul	8500	c.pigsty	分布式配置管理，服务发现
Consul DNS	8600	-	Consul提供的DNS服务
Nginx	80	pigsty	所有服务的入口代理
Yum Repo	80	yum.pigsty	本地Yum源
Haproxy Index	80	h.pigsty	所有Haproxy管理界面的访问代理
NTP	123	n.pigsty	环境统一使用的NTP时间服务器
Dnsmasq	53	-	环境统一使用的DNS域名解析服务器
Loki	3100	-	实时日志收集基础设施（选装）

Versatile Application Runtime



Database Node

以最小复杂度实现必须功能

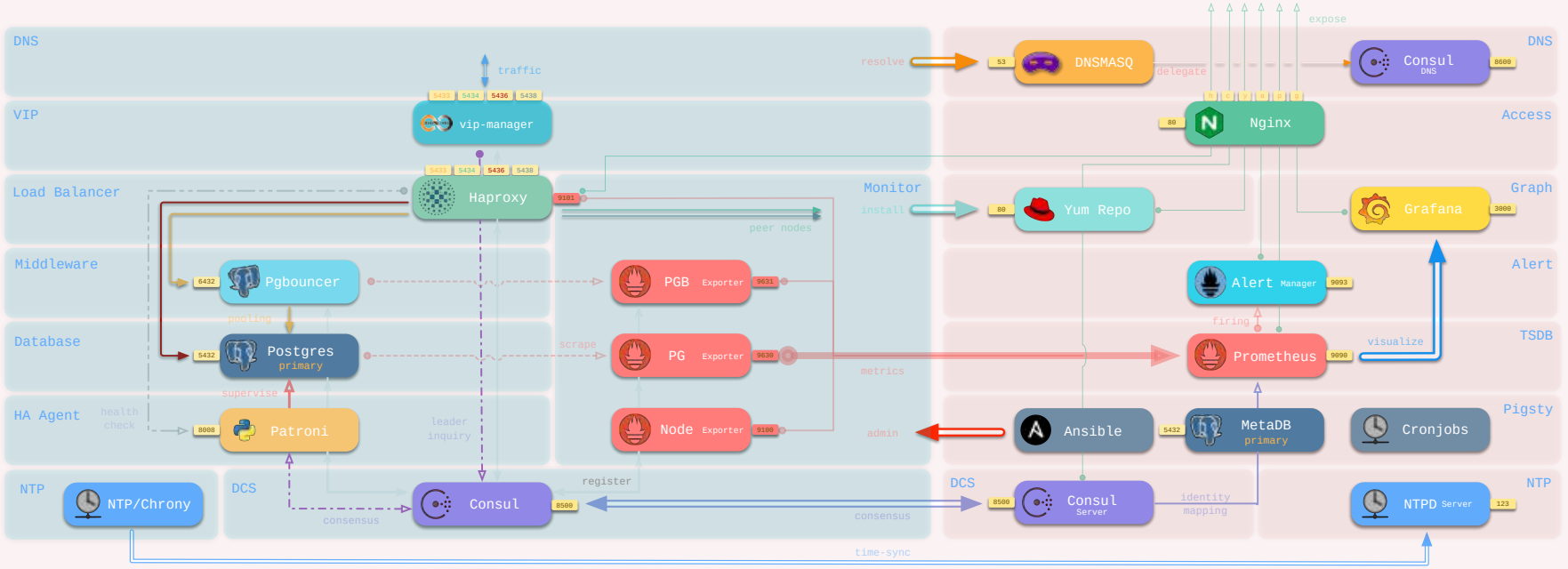


Infrastructure & Node

数据库集群与基础设施松耦合，通过注册的方式关联
基础设施升级&故障不会影响数据库集群

（除高可用使用的DCS外，维护模式下数据库集群亦免受DCS影响）

组件	端口	默认域名	说明
Grafana	3000	g.pigsty	Pigsty监控系统图形界面
Prometheus	9090	p.pigsty	监控时序数据库
AlertManager	9093	a.pigsty	报警聚合管理组件
Consul	8500	c.pigsty	分布式配置管理，服务发现
Consul DNS	8600	-	Consul提供的DNS服务
Nginx	80	pigsty	所有服务的入口代理
Yum Repo	80	yum.pigsty	本地Yum源
Haproxy Index	80	h.pigsty	所有Haproxy管理界面的访问代理
NTP	123	n.pigsty	环境统一使用的NTP时间服务器
Dnsmasq	53	-	环境统一使用的DNS域名解析服务器
Loki	3100	-	实时日志收集基础设施（选装）



Database Cluster

基于DCS

Fencing & Softdog

自动故障切换

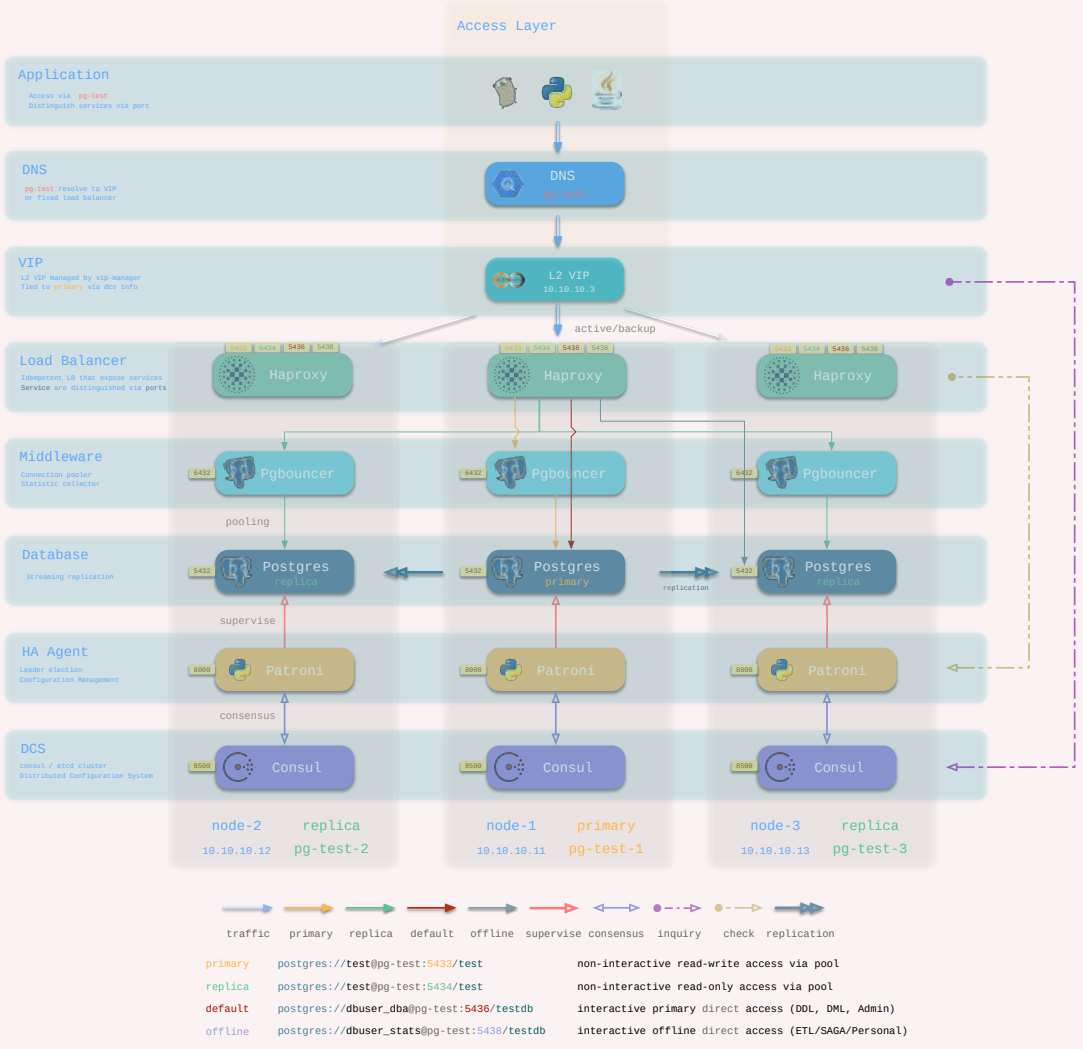
流量路由自动修复

实例提供幂等的服务

集群可自动从常见故障中恢复，
实例成员对外表现等价，
只要仍有任意实例存活，

集群对外提供的各种**服务**就不会停止。

从库故障基本无影响。
主库故障切换时，读写流量影响约15秒



Service

使用配置文件定义新的服务

采用NodePort的方式对外提供服务

任何实例都对外暴露完整服务，通过不同端口区分

从外部用户看来，集群中每一个实例都是相同的。

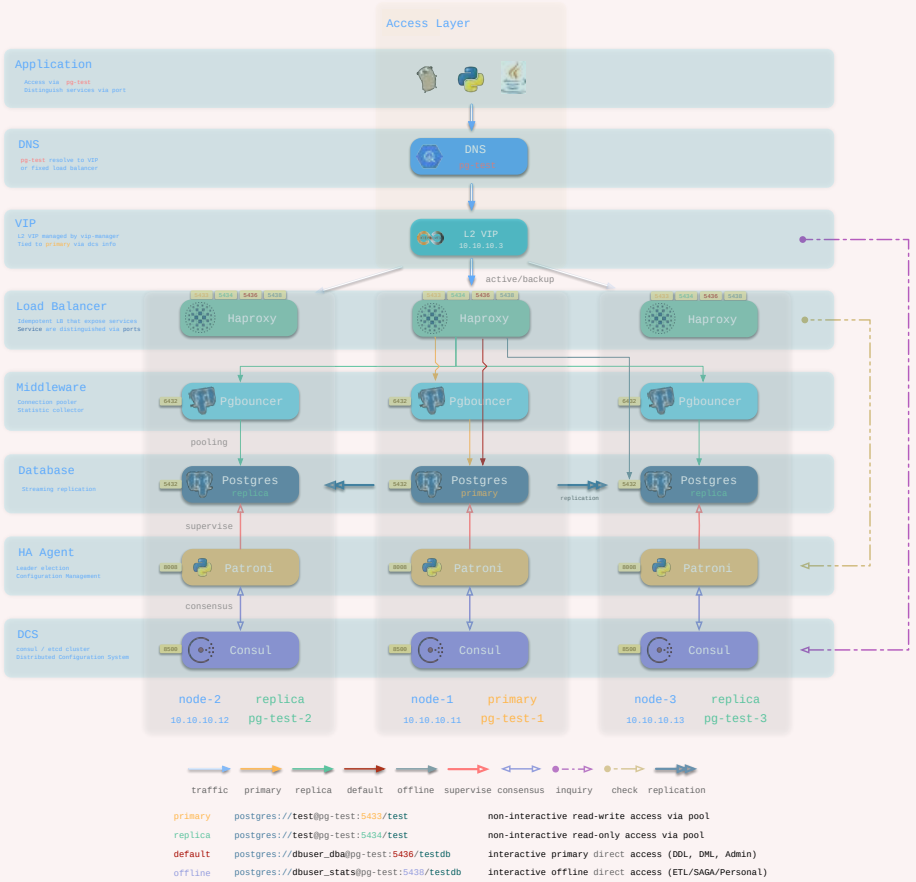
传统数据库，带来 **分布式数据库** 的体验

服务	端口	用途	说明
primary	5433	生产读写	通过**连接池**连接至集群主库
replica	5434	生产只读	通过**连接池**连接至集群从库
default	5436	管理	直接连接至集群主库
offline	5438	ETL/个人用户	直接连接至集群可用的离线实例

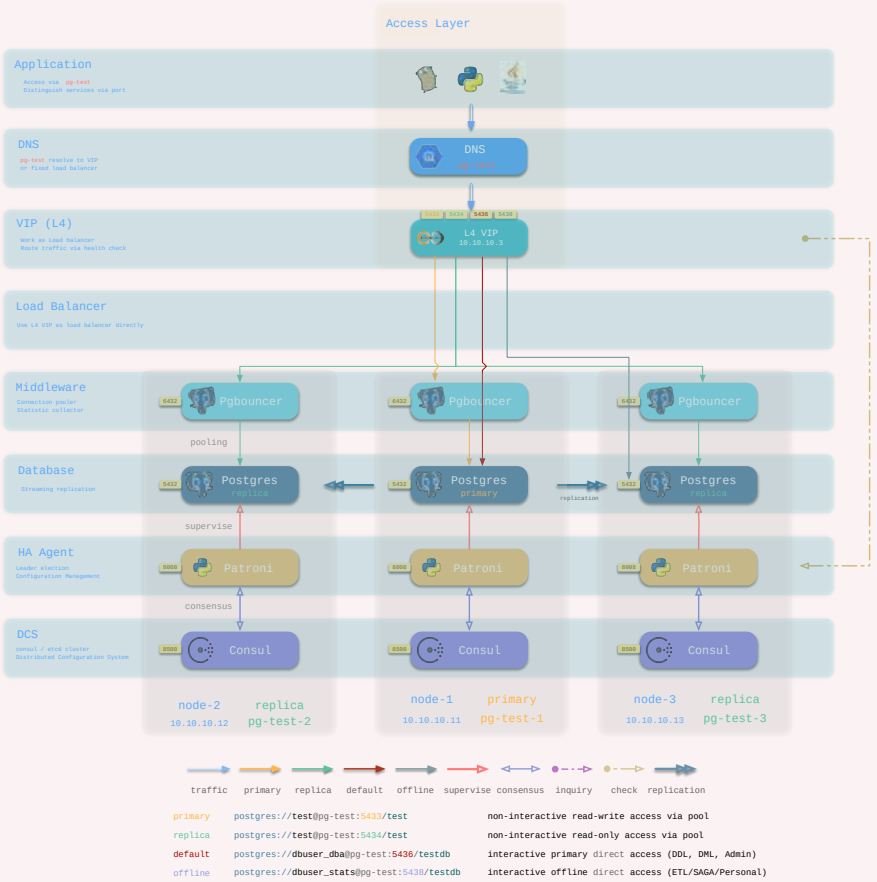
服务	端口	说明	样例
primary	5433	只有生产用户可以连接	postgres://test@pg-test:5433/test
replica	5434	只有生产用户可以连接	postgres://test@pg-test:5434/test
default	5436	管理员与DML执行者可以连接	postgres://dbuser_admin@pg-test:5436/test
offline	5438	ETL/STATS个人用户可以连接	postgres://dbuser_stats@pg-test-tt:5438/test postgres://dbp_vonng@pg-test:5438/test

```
# standby service will route {ip|name}:5435 to sync replica's pgbouncer (5435->6432 standby)
- name: standby                   # required, service name, the actual svc name will be prefixed with `pg_cluster`, e.g: pg-meta-standby
  src_ip: "*"                     # required, service bind ip address, `*` for all ip, `vip` for cluster `vip_address`
  src_port: 5435                  # required, service exposed port (work as kubernetes service node port mode)
  dst_port: postgres              # optional, destination port, postgres|pgbouncer|<port_number>   , pgbouncer(6432) by default
  check_method: http              # optional, health check method: http is the only available method for now
  check_port: patroni             # optional, health check port: patroni|pg_exporter|<port_number> , patroni(8008) by default
  check_url: /read-only?lag=0     # optional, health check url path, / by default
  check_code: 200                 # optional, health check expected http code, 200 by default
  selector: "[]"                  # required, JMESPath to filter inventory ()
  selector_backup: "[? pg_role == `primary`]"  # primary used as backup server for standby service (will not work because /sync for )
  haproxy:                        # optional, adhoc parameters for haproxy service provider (vip_l4 is another service provider)
    maxconn: 3000                 # optional, max allowed front-end connection
    balance: roundrobin           # optional, haproxy load balance algorithm (roundrobin by default, other: leastconn)
    default_server_options: 'inter 3s fastinter 1s downinter 5s rise 3 fall 3 on-marked-down shutdown-sessions slowstart 30s maxconn 3000 maxqueue
128 weight 100'
```

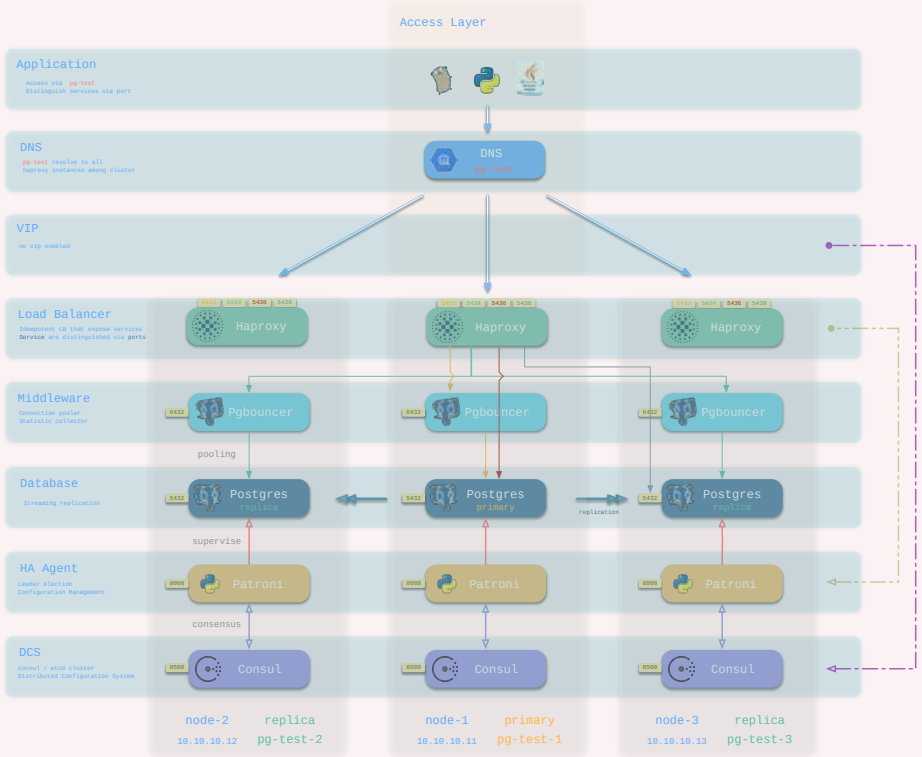
DNS + L2 VIP + HAProxy

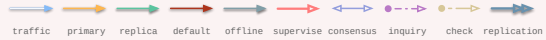


DNS + L4 VIP



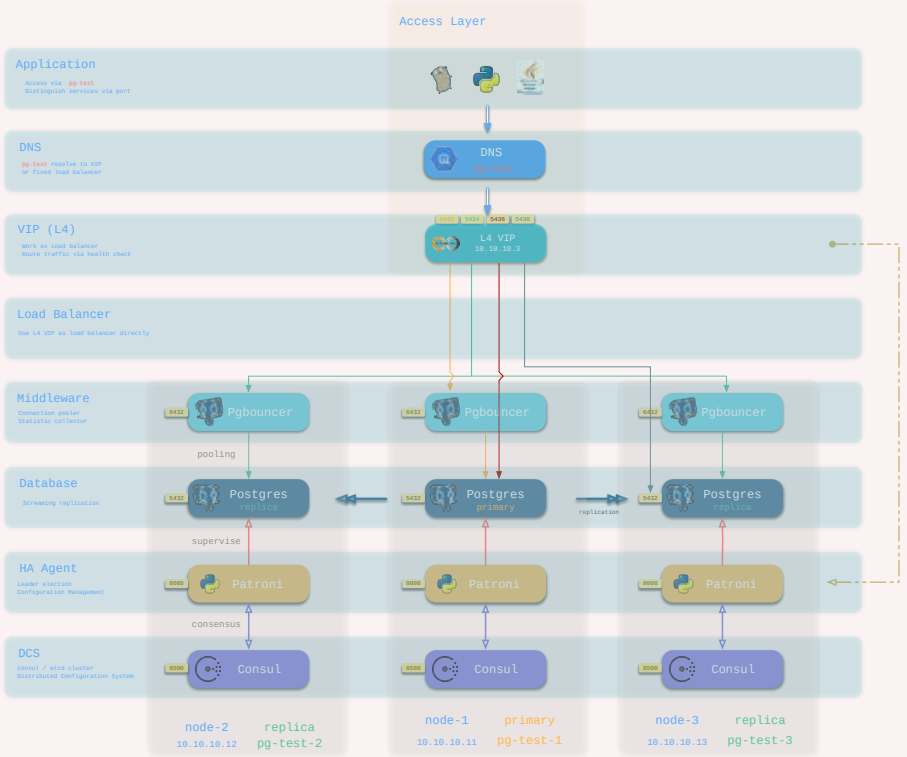
DNS + HAProxy

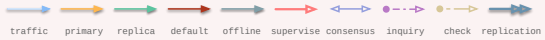




primary	postgres://test@pg-test:5433/test	non-interactive read-write access via pool
replica	postgres://test@pg-test:5434/test	non-interactive read-only access via pool
default	postgres://dbuser_dba@pg-test:5436/testdb	interactive primary direct access (DDL, DML, Admin)
offline	postgres://dbuser_stats@pg-test:5438/testdb	interactive offline direct access (ETL/SAGA/Personal)

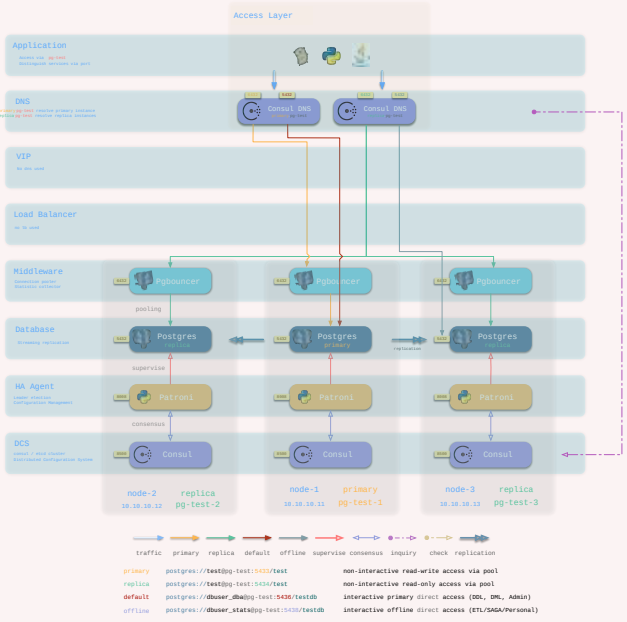
DNS + L4 VIP



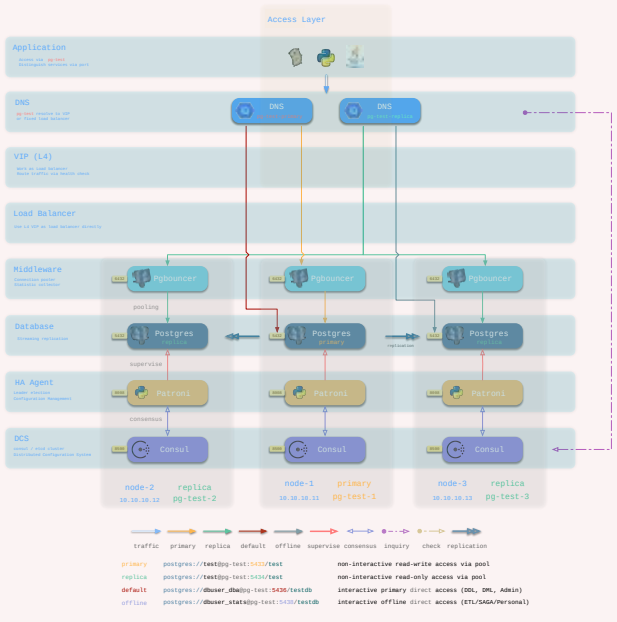


primary	postgres://test@pg-test:5433/test	non-interactive read-write access via pool
replica	postgres://test@pg-test:5434/test	non-interactive read-only access via pool
default	postgres://dbuser_dba@pg-test:5436/testdb	interactive primary direct access (DDL, DML, Admin)
offline	postgres://dbuser_stats@pg-test:5438/testdb	interactive offline direct access (ETL/SAGA/Personal)

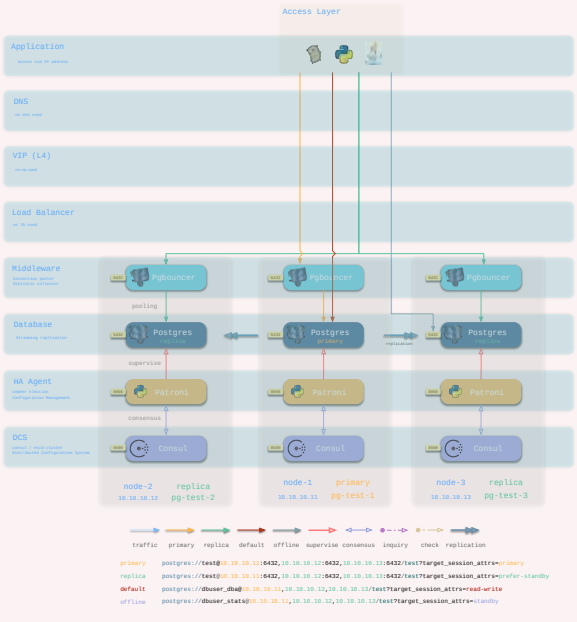
Consul DNS SD



Pure DNS



IP + Smart Client



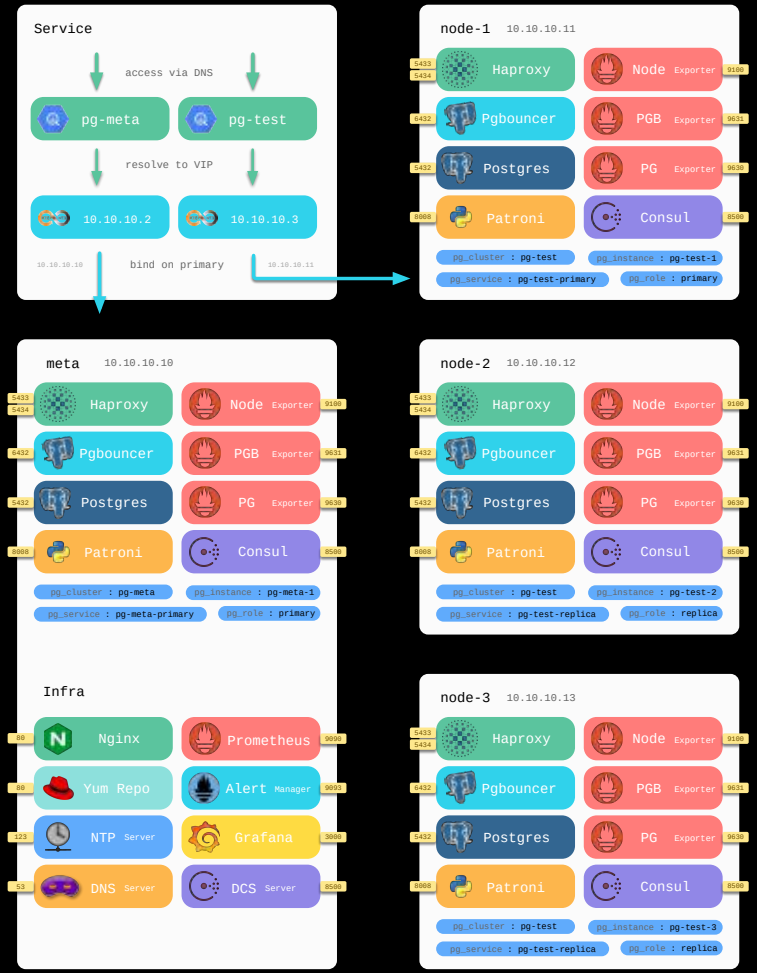
Sandbox

本地运行，一键拉起，低配置无压力

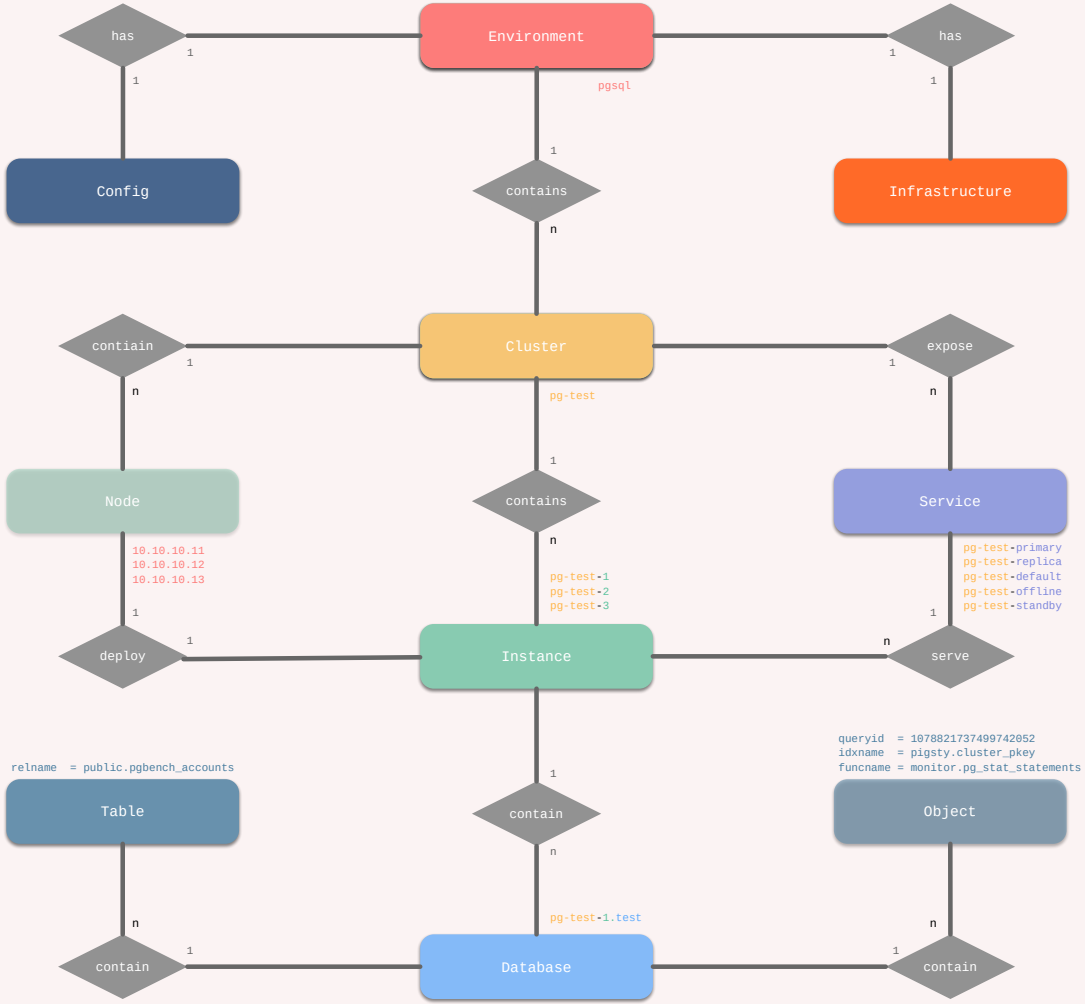
Jupyterlab，Grafana，Echarts
方便易用，助力数据分析

确定性的环境，便于交付演示案例

Just like Minikube for Kubernetes



Domain Extended



Analytic

数据分析

与可视化

*The best vision is **insight**.*

— Malcolm Forbes

Pigsty for Analysis

专注于数据处理、分析与可视化

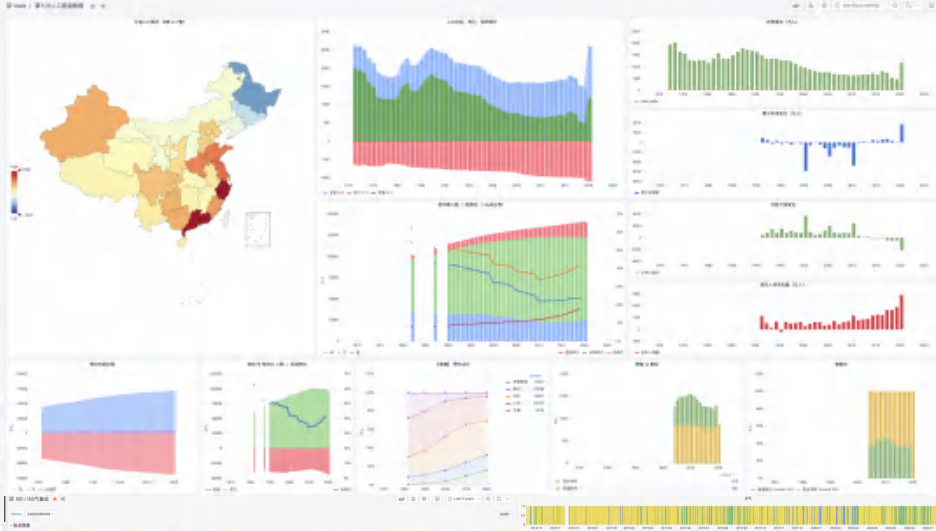
使用Grafana承载”前端”与”后端"的主体功能。

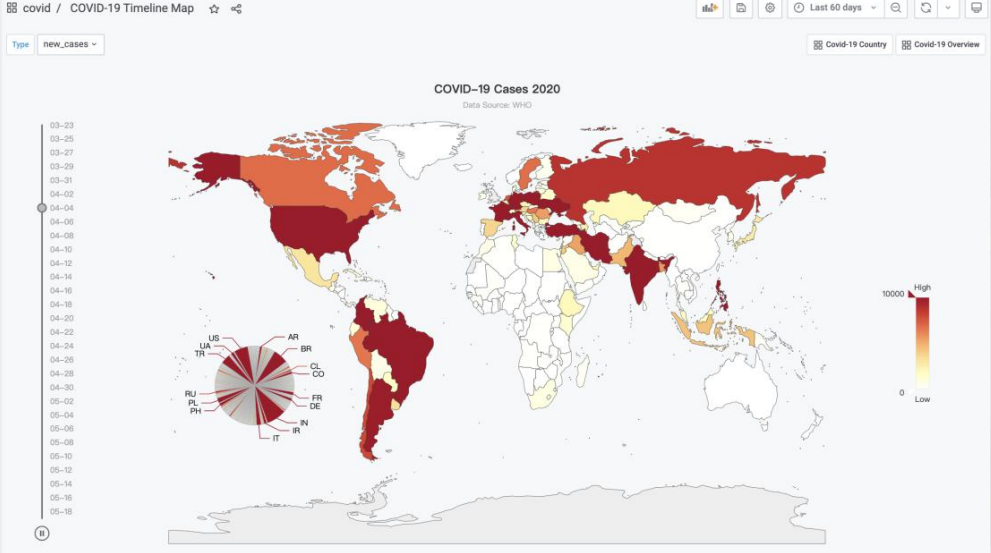
使用PostgreSQL作为主要数据库，存储数据并实现业务逻辑

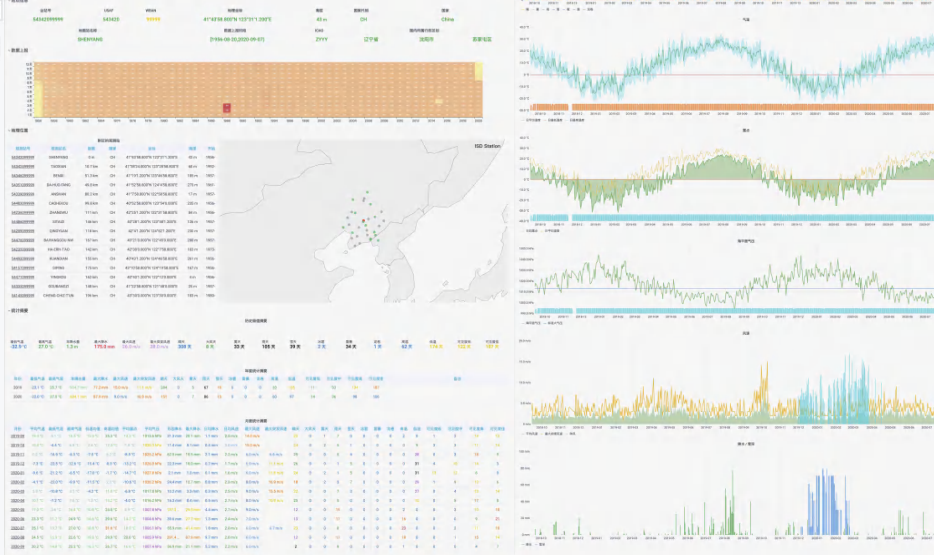
使用Echarts与Grafana实现数据可视化与交互功能

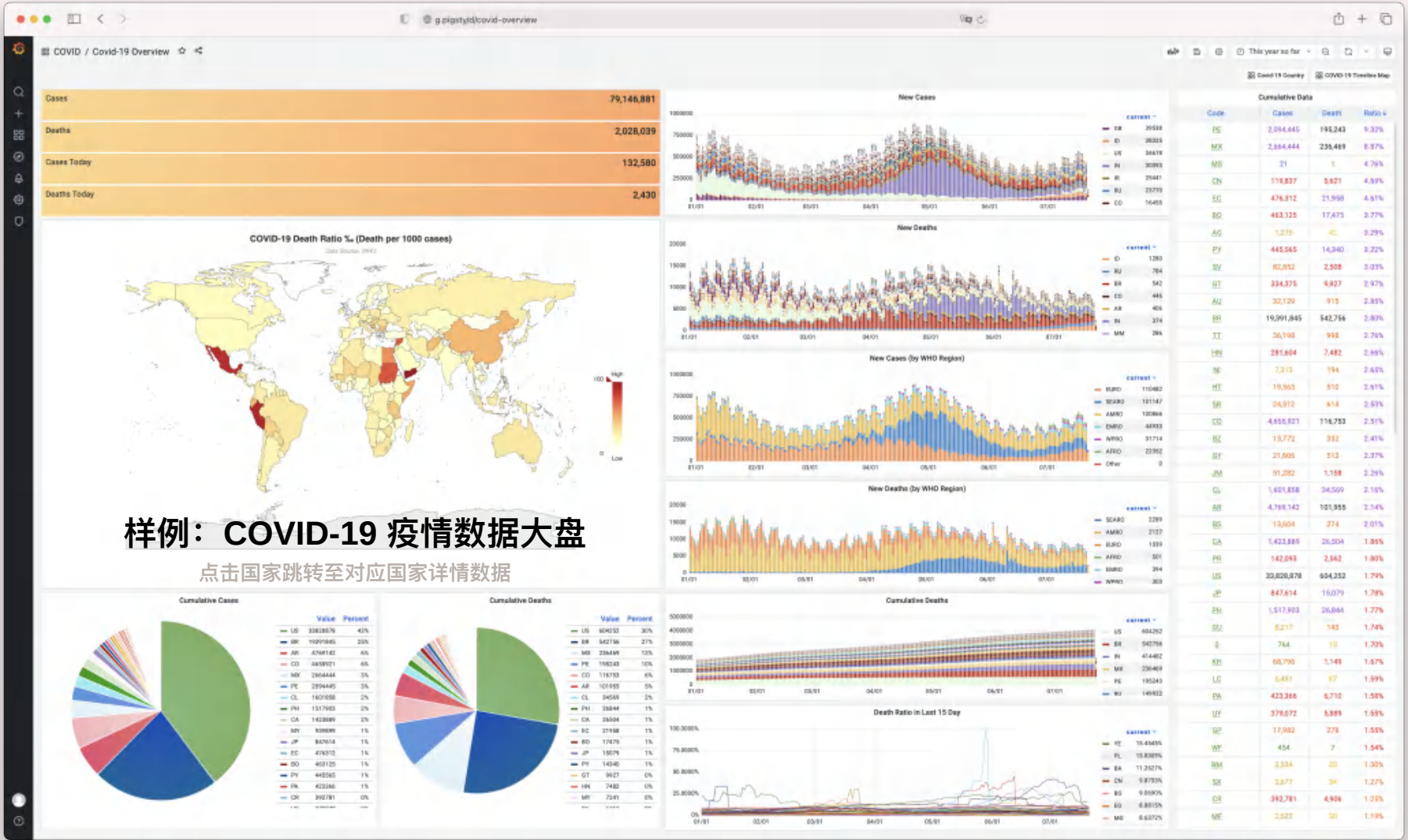
采用统一的格式，可以在Pigsty环境中一键安装运行

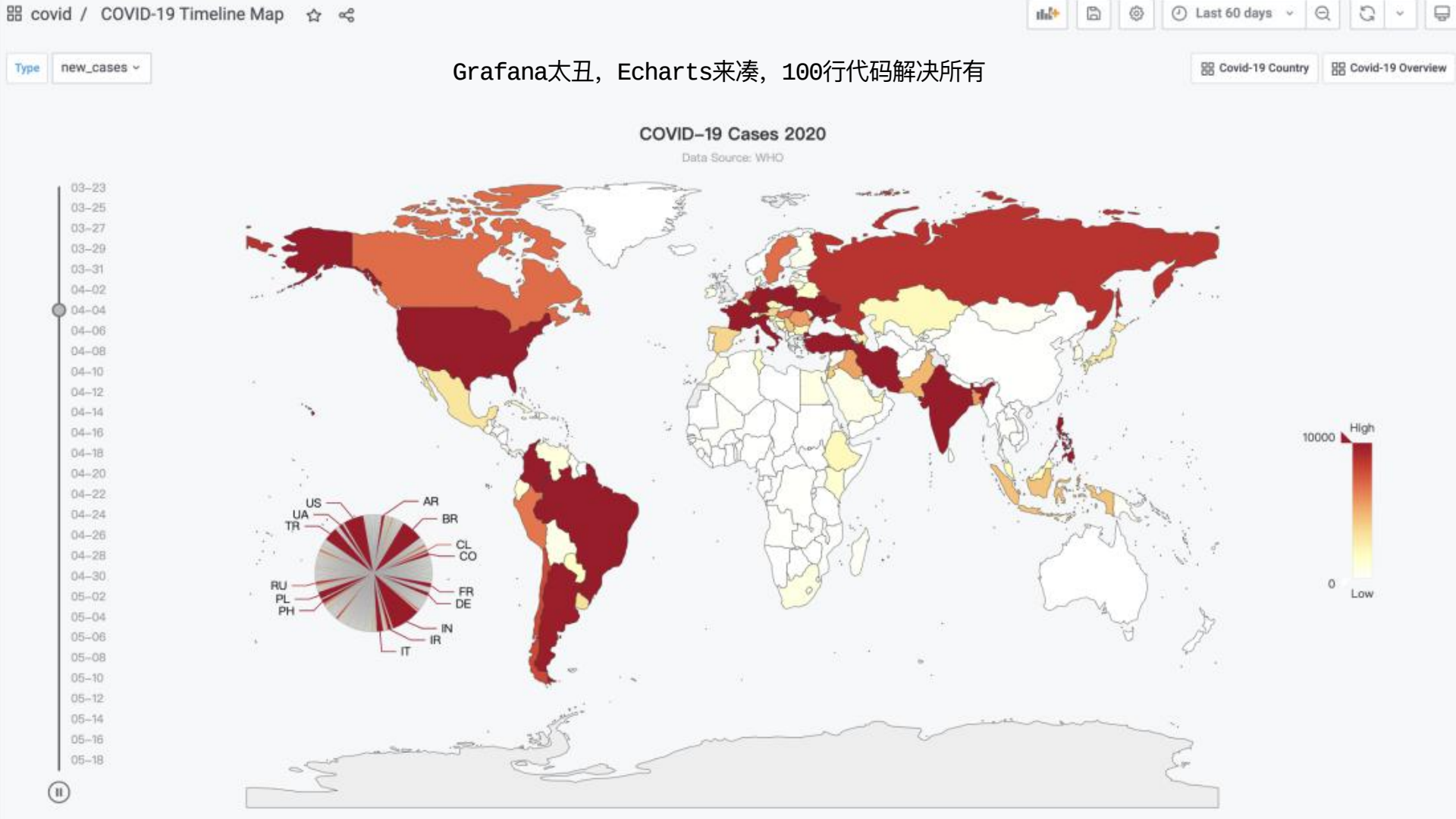

29760
29760
3,143,212,385
3,143,212,385
154,669,550
9,696,835,000




covid / COVID-19 Timeline Map
Last 60 days
Type
new_cases
Covid-19 Country
Covid-19 Overview
COVID–19 Cases 2020
Data Source: WHO
10000
High
0
Low
US
UA
TR
AR
BR
CL
CO
RU
PL
PH
FR
DE
IN
IR
IT




COVID / Covid-19 Overview
Cases 79,146,881
Deaths 2,028,039
Cases Today 132,580
Deaths Today 2,430
COVID-19 Death Ratio ‰ (Death per 1000 cases)
New Cases
New Deaths
New Cases (by WHO Region)
New Deaths (by WHO Region)
Cumulative Data
Cumulative Cases
Cumulative Deaths
Cumulative Deaths
Death Ratio in Last 15 Day
样例：COVID-19 疫情数据大盘
点击国家跳转至对应国家详情数据


covid / COVID-19 Timeline Map
Last 60 days
Type
new_cases
Grafana太丑，Echarts来凑，100行代码解决所有
Covid-19 Country
Covid-19 Overview
COVID-19 Cases 2020
Data Source: WHO
03-23
03-25
03-27
03-29
03-31
04-02
04-04
04-06
04-08
04-10
04-12
04-14
04-16
04-18
04-20
04-22
04-24
04-26
04-28
04-30
05-02
05-04
05-06
05-08
05-10
05-12
05-14
05-16
05-18
US
UA
TR
AR
BR
CL
CO
RU
PL
PH
FR
DE
IN
IR
IT
High
10000
0
Low

样例：ISD

全球地表气象站
历史数据查询

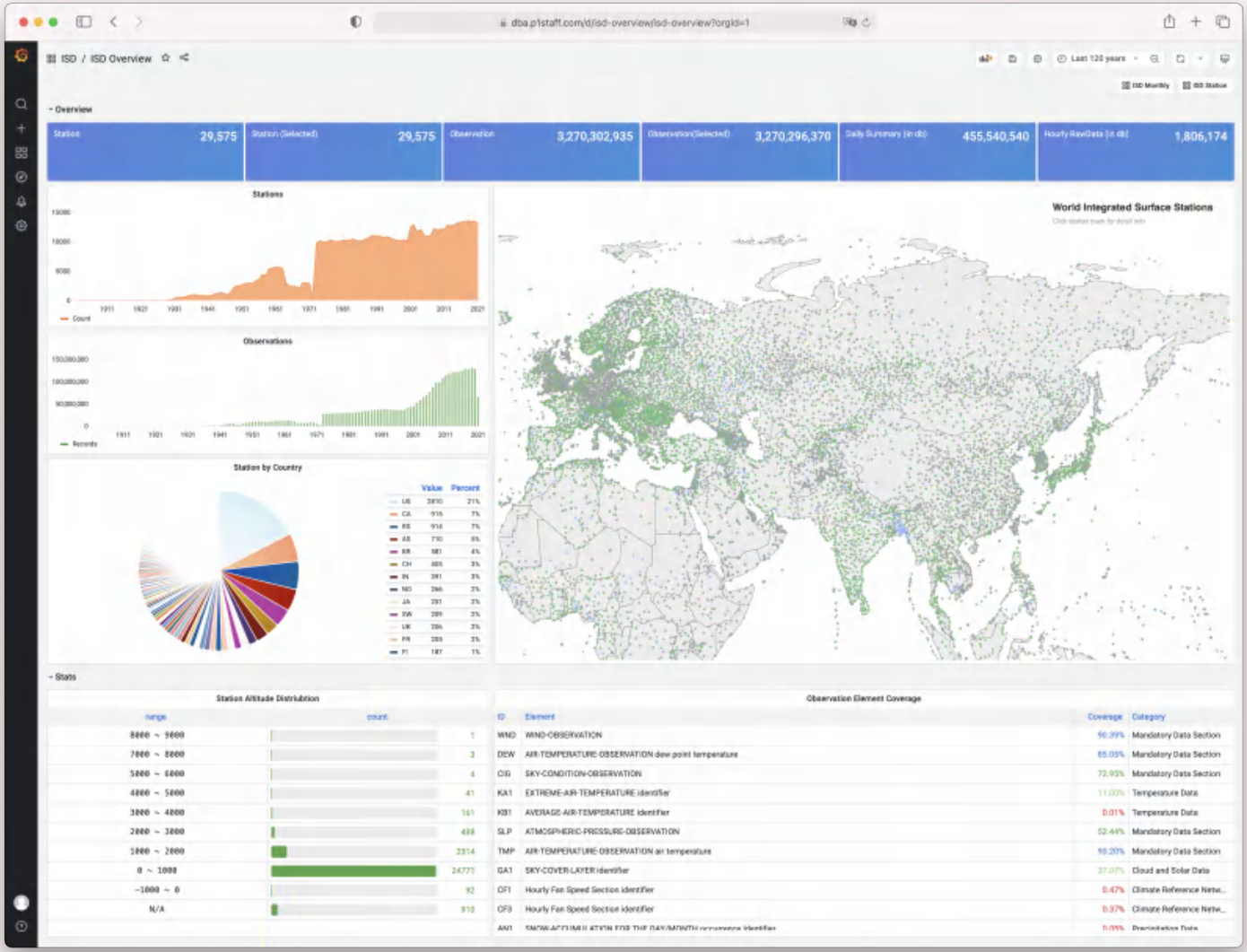
3万个地表气象站

120年来小时级

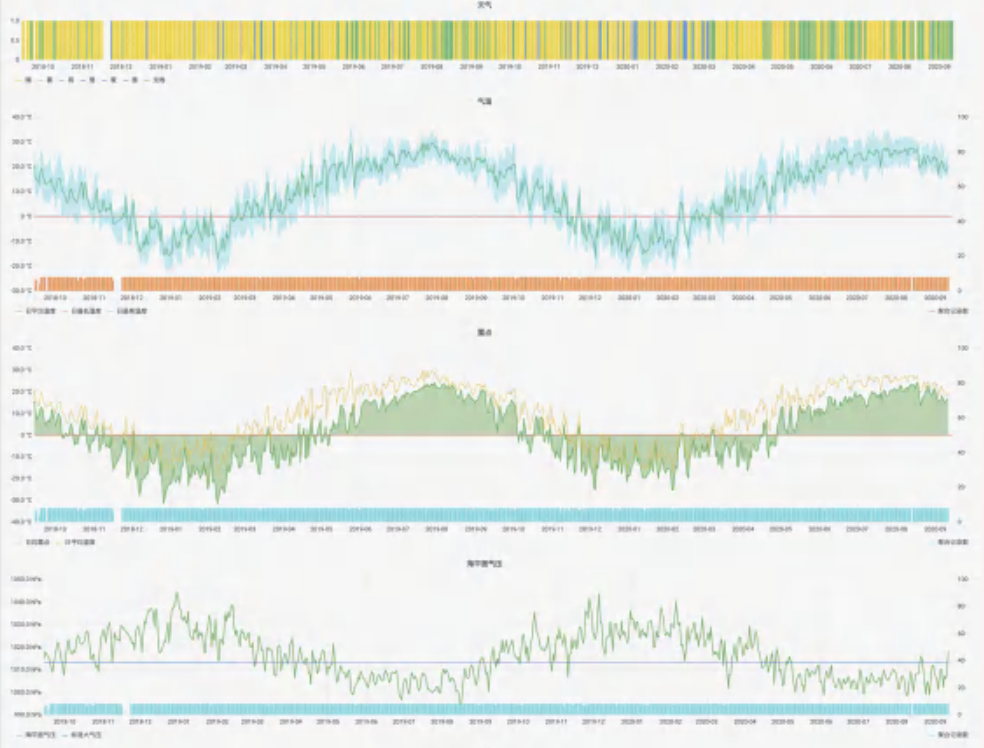
原始气象观测数据

点击气象站跳图标，

跳转至单气象站详情



单气象站历史数据摘要查询（年、月、日）



单个气象站原始观测数据（小时级）

More

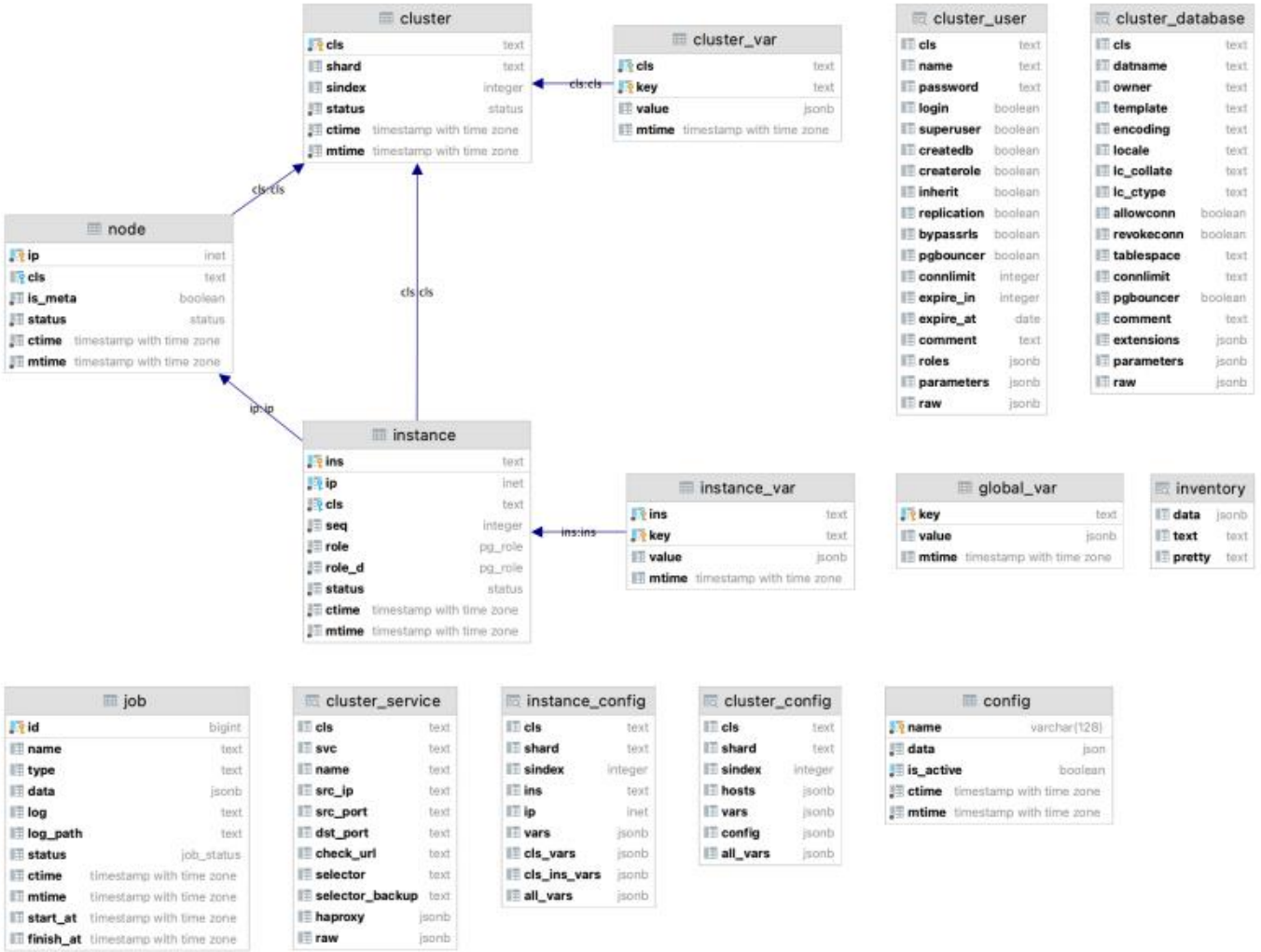
自带应用：Pigsty CMDB

使用PG管理PG

将静态配置文件

替换为动态CMDB查询

从而与外部系统与平台集成



More

Beta应用：Redis

使用同样的基础设施
部署、监控Redis

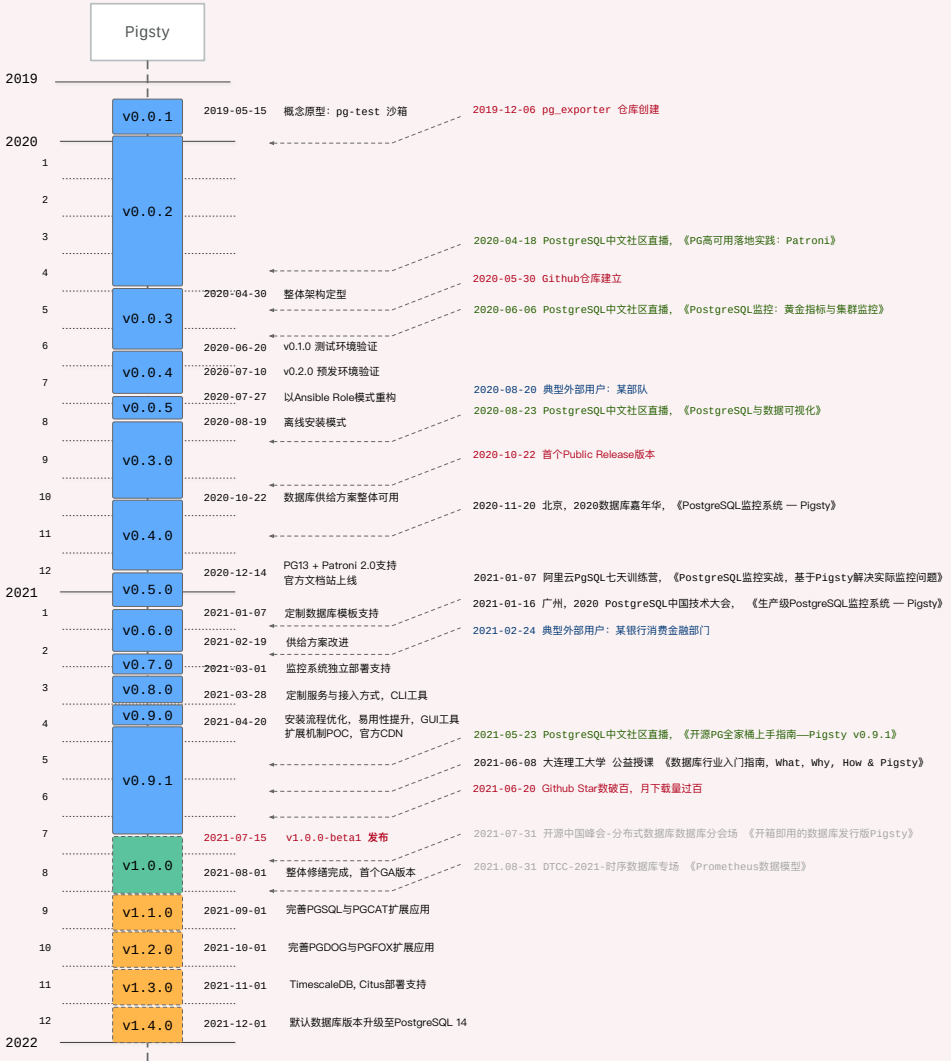
Open Source

*A system cannot be **successful** if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.*

— Donald Knuth

Pigsty Timeline

2021-07-31 v1.0.0 GA



Next Episode of Pigsty

更多功能 平台化

- 原生支持 部署Citus高可用水平分片集群组
- 支持使用 etcd 替代完整的 consul 功能
- 添加离线从库，延时从库部署支持
- 集中式备份/恢复方案
- 安全性强化：严格模式
- 巡检、日报、报警分析
- 管理平台 & CMDB
- 集成模式变更方案
- 集成数据迁移方案

监控系统 应用案例

- 更多、更全面、更专业的监控面板
- 更多、更实用的专题应用
- 集成pgbager日志分析报告
- 集成PEV慢查询可视化分析工具
- 集成黑盒监控 blackbox_exporter
- 集成更多日志衍生指标：mtail
- 完善的PostgreSQL系统目录浏览器：PGCAT

系统支持 云原生改造

- 支持ARM架构，支持更多Linux发行版：SUSE（✅），Debian，Ubuntu
- 容器化方案，Kubernetes 支持，Kubernetes Operator

管理更多 开源数据库

- 支持部署、监控更多种类的开源数据库：Redis（✅），MongoDB，MySQL，……

Open Source

第一时间获取反馈，了解用户的痛点、痒点与爽点，解决**真正的问题**

让用户能用**好数据库**，**用好**数据库

降低门槛，扩大用户群，促进PostgreSQL生态繁荣发展。

提供一种“自主可控”的选择，让中小企业，个人用户更有**自由选择**的底气

GOTC

THANKS

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE