Professional PostgreSQL scheduling made easy

Pavlo Golub

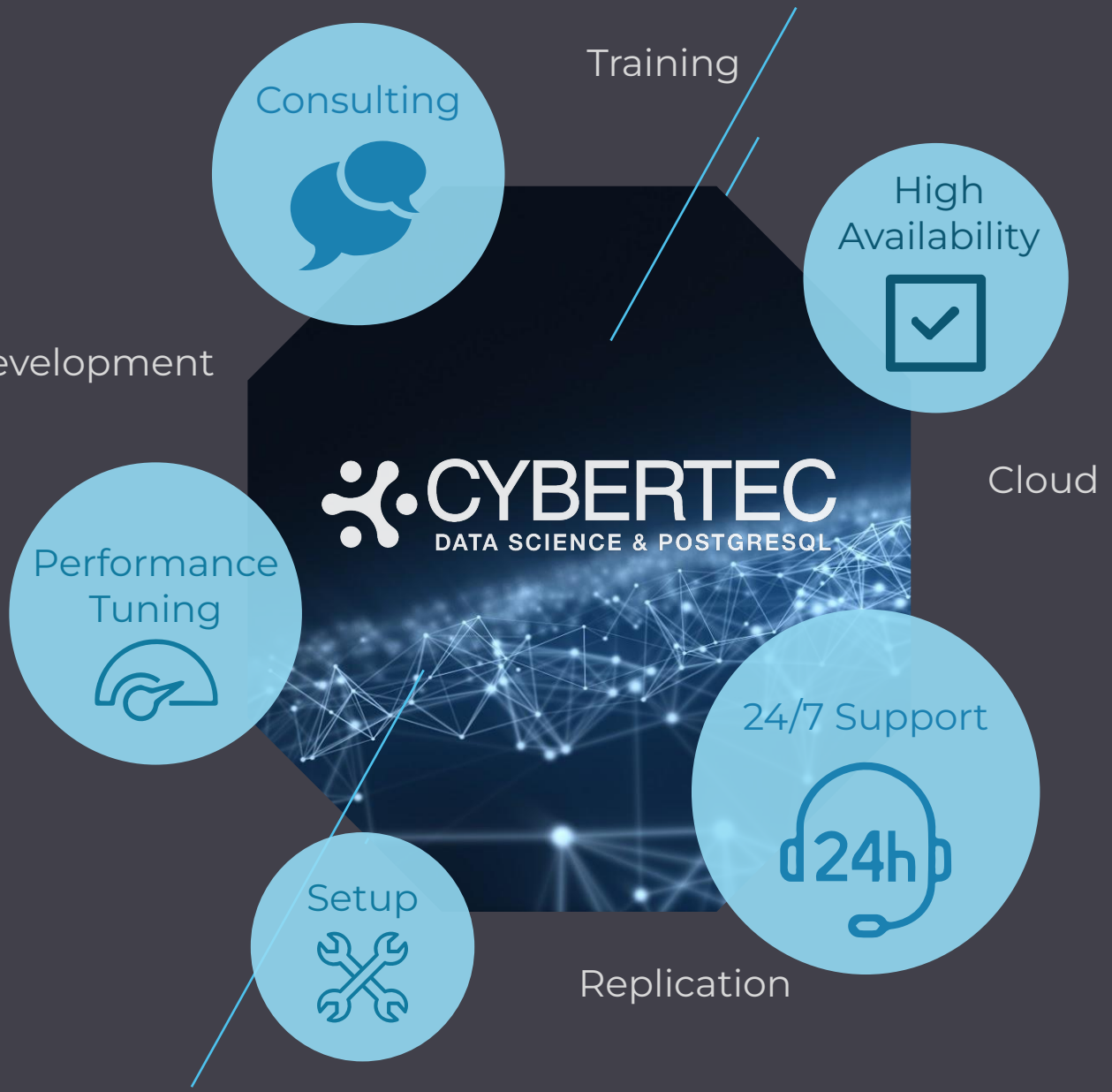
Senior Database Consultant

pavlo.golub@cybertec.at

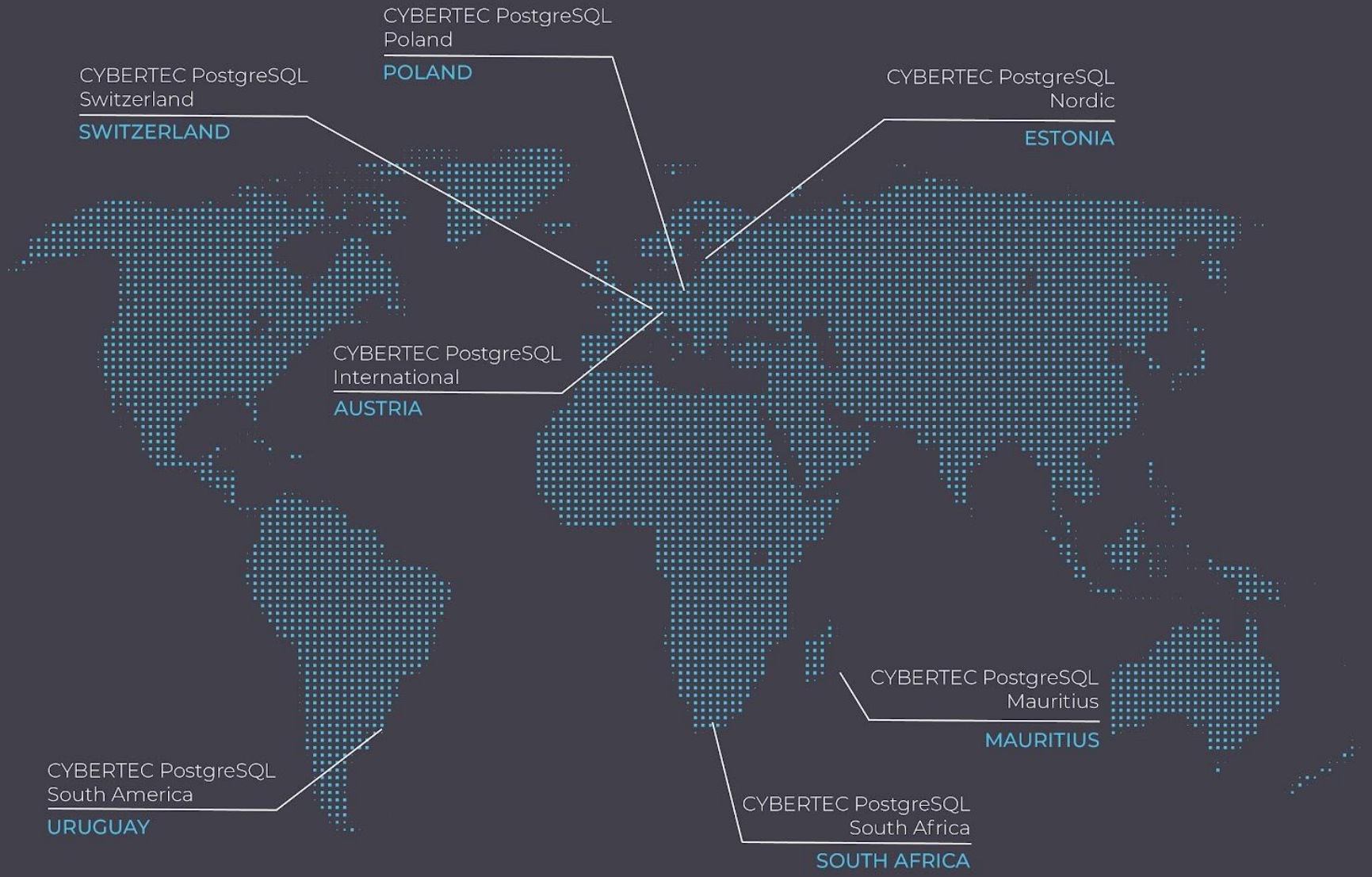
@PavloGolub

CYBERTEC
The PostgreSQL Database Company

PostgreSQL Database Services



CYBERTEC
DATA SCIENCE & POSTGRESQL



CYBERTEC
DATA SCIENCE & POSTGRESQL

CLIENT SECTORS

- ICT
- University
- Government
- Automotive
- Industry
- Trade
- Finance
- etc.

CYBERTEC
DATA SCIENCE & POSTGRESQL

pg_timetable: Today's agenda

- Different levels of database scheduling
- PostgreSQL scheduling approaches
- PostgreSQL scheduling tools available
- pg_timetable: Why it is so cool ;)

Why use a scheduler

- Maintenance
- Data Import / Export
- Backup / Restore
- Analytical Processing
- Monitoring
- External Actions

Different levels of scheduling

- Built-in Schedulers
- System Schedulers
- PostgreSQL land

Built-in Schedulers

- Microsoft SQL
- Oracle
- MySQL (MariaDB)
- DB2

Built-in Scheduler in PostgreSQL

> Many people say it's not necessary, and probably some hackers would oppose it; but mainly I think we just haven't agreed (or even discussed) what the design of such a scheduler would look like. For example, do we want it to be able to just connect and run queries and stuff, or do we want something more elaborate able to start programs such as running pg_dump? What if the program crashes -- should it cause the server to restart? And so on. It's not a trivial problem.
>
> Alvaro Herrera

SYSTEM SCHEDULING

Tools available:

- cron, anacron, etc.
- Windows Task Scheduler
- Google Cloud Tasks, Amazon Scheduled Tasks
- Kubernetes CronJob

Cons: They don't know anything about databases.

SCHEDULERS IN “PostgreSQL land”

- pgAgent
- jpgAgent
- pg_cron
- pgBucket (runseven)
- pgAutomator (discontinued?)
- maybe more?

pgAgent

- The oldest one!
- Was a part of pgAdmin, now distributed independently
- Written in C++
- Stores configuration in the database
- SQL and SHELL tasks
- https://github.com/postgres/pgagent

jpgAgent

- pgAgent compatible
- Written in Java
- Minimizes the pain of switching for existing pgAgent users
- Provides more stable and feature rich agent implementation
- SQL and SHELL tasks, with partial email task support
- Parallel task execution
- Can kill running jobs
- Supports job and task timeout
- https://github.com/GoSimpleLLC/jpgAgent

pg_cron

- Very old
- Implemented as PostgreSQL background worker
- Written in C
- Uses libpq to open a new connection to the databases
- SQL only tasks
- Jobs are executed locally with permissions of the current user
- Superusers may update sys table to allow remote execution
  - Need to use .pgpass to authenticate with the remote server
- https://github.com/citusdata/pg_cron/

pgBucket (runseven)

- Under active development
- Written in C++
- Uses dedicated configuration file
- SQL and SHELL tasks
- Special cascaded/event tasks
- Auto job disable
- https://bitbucket.org/dineshopenscg/pgbucket/

HOW SCHEDULERS PROLIFERATE:

(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)

SITUATION: THERE ARE 14 COMPETING SCHEDULERS

14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL SCHEDULER THAT COVERS EVERYONE'S USE CASES.

SOON:

SITUATION: THERE ARE 15 COMPETING SCHEDULERS

pg_timetable: CREATING THE **ULTIMATE** SCHEDULER

- Main design principles
- Architecture
- Features
- Demo

MAIN PRINCIPLES

- 1-minute setup
  - Docker image (“cloud ready”)
  - One binary written in Go
- Non-invasive
  - No extensions or superuser needed for base functionality
  - Schema auto deployment
- Huge number of jobs
- Cross platform support
- SQL, PROGRAM and BUILT-IN tasks available

Supported Environments

Cloud Service	Supported	PostgreSQL Version	Supported	OS	Supported
Alibaba Cloud	✅	16 (devel)	✅	Linux	✅
Amazon RDS	✅	15 (current)	✅	Darwin	✅
Amazon Aurora	✅	14	✅	Windows	✅
Azure	✅	13	✅	FreeBSD*	✅
Citus Cloud	✅	12	✅	NetBSD*	✅
Crunchy Bridge	✅	11	✅	OpenBSD*	✅
DigitalOcean	✅	10	✅	Solaris*	✅
Google Cloud	✅				
Heroku	✅				
Supabase	✅				

Comparison table

Feature\Product	pg_timetable	pg_cron	pgAgent	jpgAgent	pgbucket
Architecture					
Year	2019	2016	2008	2016	2015
Implementation	standalone	bgworker	standalone	standalone	standalone
Language	Go	C	C++	Java	C++
Can operate w\o extension	✓	✗	✗	✗	✓
Jobs meta stored in	database	database	database	database	file
Remote Database Execution	✓	✗	✓	✓	✓
Cross Platform	✓	✓	✓	✓	✗

Comparison table

Feature\Product	pg_timetable	pg_cron	pgAgent	jpgAgent	pgbucket
Functionality					
SQL tasks	✓	✓	✓	✓	✓
Program/Shell tasks	✓	✗	✓	✓	✓
Built-in tasks	✓	✗	✗	✗	✗
Parallel Jobs	✓	✓	✓	✓	✓
Parallel Jobs Limit	✓	?	?	?	✓
Concurrency protection	✓	✓	?	?	?
Task Parameters	✓	✗	✗	✗	✗
Arbitrary Role	✓	✗	✓	✓	✓
On Success Task	✓	✗	✓	?	✓
On Error Task	✗	✗	✗	?	✓

Comparison table

Feature\Product	pg_timetable	pg_cron	pgAgent	jpgAgent	pgbucket
Scheduling					
Standard Cron	✓	✓	✗	✗	✓
Interval	✓	✗	✗	✗	✗
On Reboot	✓	✗	✗	✗	✗
Start Manually	✓	✗	✗	✗	✓
Kill Running Job	✓	✗	✗	✓	✓
Job Timeout	✓	✗	✗	✓	✗
Task Timeout	✓	✗	✗	✗	✗
Disable Job	✓	?	✓	✓	✓
Auto Job Disable	✗	✗	✗	✗	✓
Self-Destructive Jobs	✓	✗	✗	✗	?

BUILDING BLOCKS: TASKS AND CHAINS

- A task is the most basic building block
  - Tasks can take parameters
  - e.g. “Download data”, “Aggregate data”, etc
- A chain is a sequence of tasks
  - Arrange tasks in a large sequence of things

BUILDING BLOCKS: Tasks and chains

Start Transaction

Download data

Aggregate

Delete file

Commit

DESIGN: Which features are necessary?

- Cron-style scheduling
  - People are used to that
  - Necessary for simple things
- Ability for more complex flows
  - By adding “chains”
- Enhanced logs
  - Workflow log and task execution log
  - Database side log means that a GUI can be written
  - Not true for text logs

DESIGN: Which features are necessary?

- Concurrency implemented using light weight goroutines
  - Efficiency does matter in case of xxx.xxx jobs
- Fully database driven configuration
  - Backups are easy and centralized
  - GUIs can be produced easily
  - Easy to search, modify
  - Simplified versioning

DESIGN: Which features are necessary?

- Concurrency protection
  - Make sure that identical jobs cannot run concurrently
  - Example: Ensure that only one backup is running, etc.
- Optionally ignore errors
- Optional exclusive execution

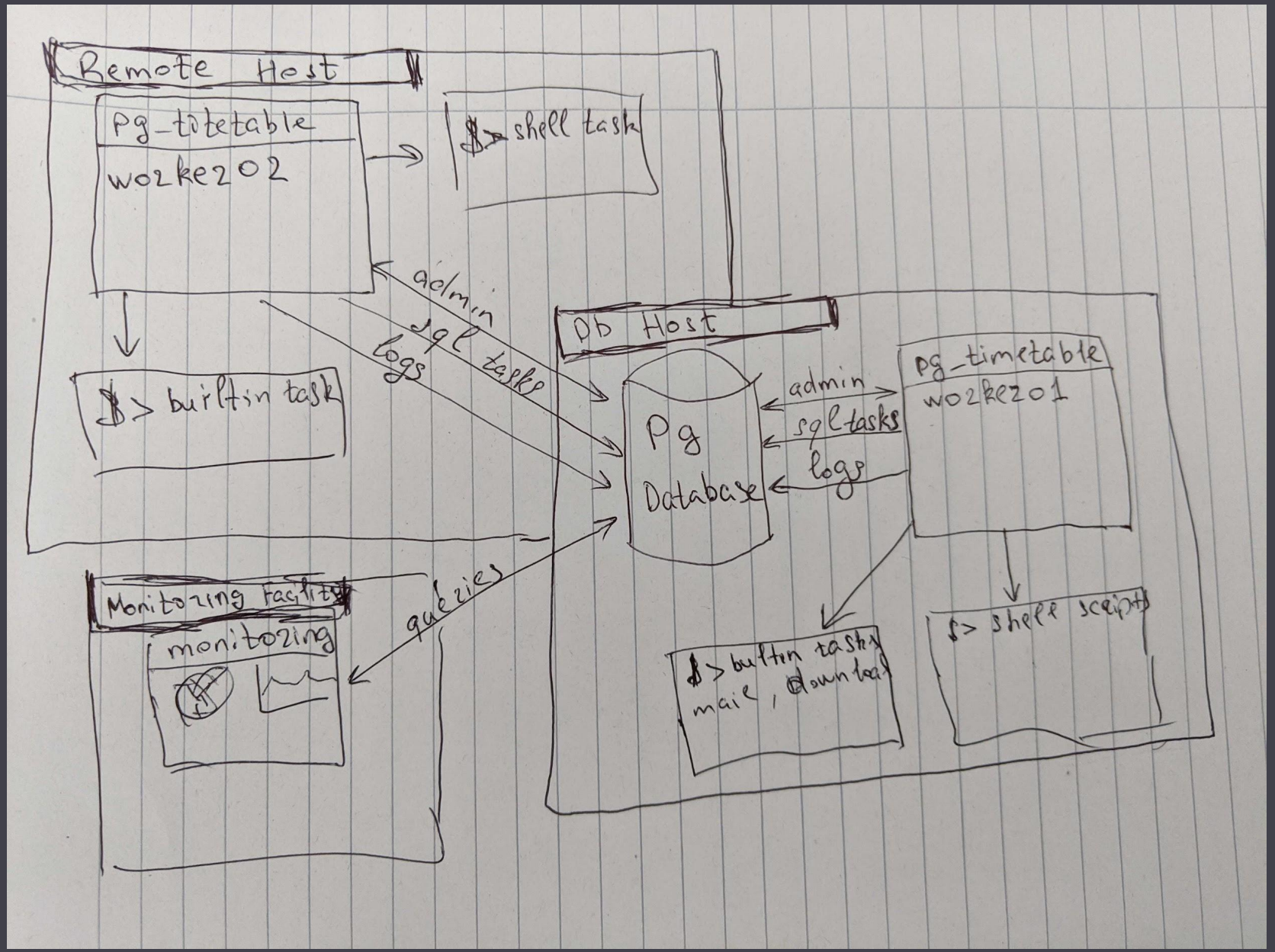
DESIGN: Which features are necessary?

- Self-destructive chains
  - Basically for asynchronous execution
  - Try to execute one and kill it when done
  - Otherwise try again

This is super important for GUI applications
They can do async execution using only INSERT

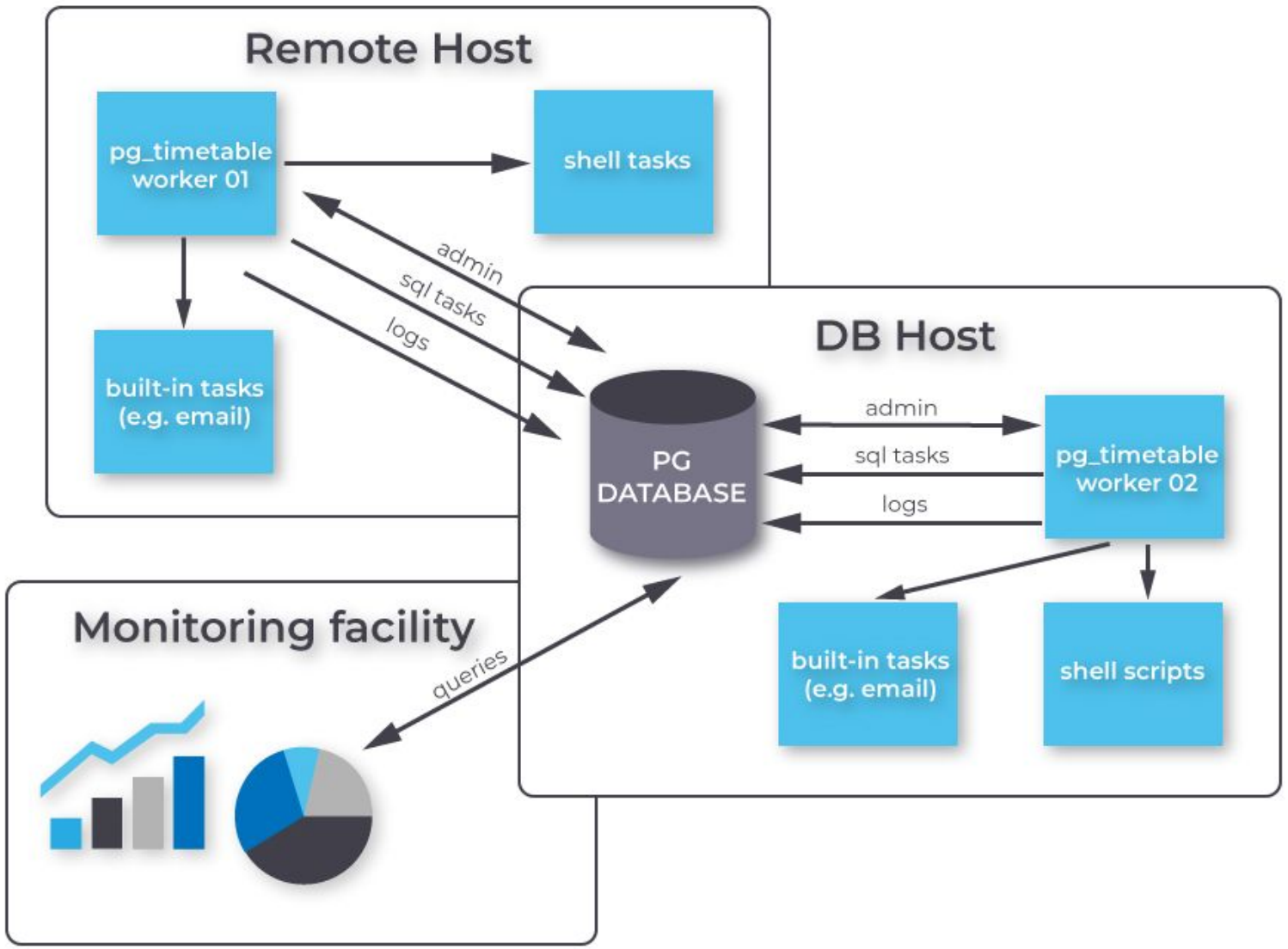
ARCHITECTURE AND COMPONENTS

- Workers (Golang)
- Config database (PostgreSQL)
- Optional target databases
- Optional monitoring
  - pgwatch2
  - psql
  - Anything you want ...
    - Everything is in tables



PG Timetable

CYBERTEC
DATA SCIENCE & POSTGRESQL



TODO

- ✅ Task / Chain abortion
- ✅ Asynchronous chain execution
- ⏳ OnError Chain / Task
- ✅ Support interval scheduling, e.g. `'@interval(00:00:10)'`
- ✅ Collect client messages for tasks, e.g. `'RAISE NOTICE foo'`
- ✅ Tool for debugging standalone tasks
- ✅ Graphical User Interface
- ⏳ BGW implementation

Getting started

```
$ pg_timetable -c loader postgresql://scheduler@localhost/timetable
2022-04-07 13:04:45.578 [INFO] Database connection established
2022-04-07 13:04:45.580 [INFO] Executing script: DDL
2022-04-07 13:04:45.760 [INFO] Schema file executed: DDL
2022-04-07 13:04:45.760 [INFO] Executing script: JSON Schema
2022-04-07 13:04:45.764 [INFO] Schema file executed: JSON Schema
2022-04-07 13:04:45.764 [INFO] Executing script: Cron Functions
2022-04-07 13:04:45.768 [INFO] Schema file executed: Cron Functions
2022-04-07 13:04:45.769 [INFO] Executing script: Job Functions
2022-04-07 13:04:45.785 [INFO] Schema file executed: Job Functions
2022-04-07 13:04:45.786 [INFO] Configuration schema created...
2022-04-07 13:04:45.792 [INFO] Accepting asynchronous chains execution requests...
2022-04-07 13:04:45.794 [INFO] [count:0] Retrieve scheduled chains to run @reboot
2022-04-07 13:04:45.796 [INFO] [count:0] Retrieve interval chains to run
2022-04-07 13:04:45.974 [INFO] [count:0] Retrieve scheduled chains to run
...
```

Getting started: session

```
$ pg_timetable -c loader postgresql://scheduler@localhost/timetable
2022-04-07 13:04:45.578 [INFO] Database connection established
2022-04-07 13:04:45.580 [INFO] Executing script: DDL
2022-04-07 13:04:45.760 [INFO] Schema file executed: DDL
2022-04-07 13:04:45.760 [INFO] Executing script: JSON Schema
2022-04-07 13:04:45.764 [INFO] Schema file executed: JSON Schema
2022-04-07 13:04:45.764 [INFO] Executing script: Cron Functions
2022-04-07 13:04:45.768 [INFO] Schema file executed: Cron Functions
2022-04-07 13:04:45.769 [INFO] Executing script: Job Functions
2022-04-07 13:04:45.785 [INFO] Schema file executed: Job Functions
2022-04-07 13:04:45.786 [INFO] Configuration schema created...
2022-04-07 13:04:45.792 [INFO] Accepting asynchronous chains execution requests...
2022-04-07 13:04:45.794 [INFO] [count:0] Retrieve scheduled chains to run @reboot
2022-04-07 13:04:45.796 [INFO] [count:0] Retrieve interval chains to run
2022-04-07 13:04:45.974 [INFO] [count:0] Retrieve scheduled chains to run
...
```

Getting started: new schema

```
$ pg_timetable -c loader postgresql://scheduler@localhost/timetable
2022-04-07 13:04:45.578 [INFO] Database connection established
2022-04-07 13:04:45.580 [INFO] Executing script: DDL
2022-04-07 13:04:45.760 [INFO] Schema file executed: DDL
2022-04-07 13:04:45.760 [INFO] Executing script: JSON Schema
2022-04-07 13:04:45.764 [INFO] Schema file executed: JSON Schema
2022-04-07 13:04:45.764 [INFO] Executing script: Cron Functions
2022-04-07 13:04:45.768 [INFO] Schema file executed: Cron Functions
2022-04-07 13:04:45.769 [INFO] Executing script: Job Functions
2022-04-07 13:04:45.785 [INFO] Schema file executed: Job Functions
2022-04-07 13:04:45.786 [INFO] Configuration schema created...
2022-04-07 13:04:45.792 [INFO] Accepting asynchronous chains execution requests...
2022-04-07 13:04:45.794 [INFO] [count:0] Retrieve scheduled chains to run @reboot
2022-04-07 13:04:45.796 [INFO] [count:0] Retrieve interval chains to run
2022-04-07 13:04:45.974 [INFO] [count:0] Retrieve scheduled chains to run
...
```

Getting started: workflow

```
$ pg_timetable -c loader postgresql://scheduler@localhost/timetable
2022-04-07 13:04:45.578 [INFO] Database connection established
2022-04-07 13:04:45.580 [INFO] Executing script: DDL
2022-04-07 13:04:45.760 [INFO] Schema file executed: DDL
2022-04-07 13:04:45.760 [INFO] Executing script: JSON Schema
2022-04-07 13:04:45.764 [INFO] Schema file executed: JSON Schema
2022-04-07 13:04:45.764 [INFO] Executing script: Cron Functions
2022-04-07 13:04:45.768 [INFO] Schema file executed: Cron Functions
2022-04-07 13:04:45.769 [INFO] Executing script: Job Functions
2022-04-07 13:04:45.785 [INFO] Schema file executed: Job Functions
2022-04-07 13:04:45.786 [INFO] Configuration schema created...
2022-04-07 13:04:45.792 [INFO] Accepting asynchronous chains execution requests...
2022-04-07 13:04:45.794 [INFO] [count:0] Retrieve scheduled chains to run @reboot
2022-04-07 13:04:45.796 [INFO] [count:0] Retrieve interval chains to run
2022-04-07 13:04:45.974 [INFO] [count:0] Retrieve scheduled chains to run
...
```

Schema: tables

```
$ psql -d timetable
psql (14.1)

timetable=> \dt timetable.*
           List of relations
  Schema   |   Name          | Type  |   Owner
-----------+-----------------+-------+-----------
 timetable | active_session | table | scheduler
 timetable | chain          | table | scheduler
 timetable | execution_log  | table | scheduler
 timetable | log            | table | scheduler
 timetable | migration      | table | scheduler
 timetable | parameter      | table | scheduler
 timetable | run_status     | table | scheduler
 timetable | task           | table | scheduler
(8 rows)
```

EXAMPLE: Adding a chain

Download file from the internet

↳ Ignore the errors

Remove accents

Clean table

Import new data from processed file

CYBERTEC
DATA SCIENCE & POSTGRESQL

Adding a chain

```
timetable=# \i samples/Download.sql
NOTICE:  Step 1 completed. DownloadFile task added
NOTICE:  Step 2 completed. Unacent task added
NOTICE:  relation "location" already exists, skipping
NOTICE:  Step 3 completed. Import task added
DO
timetable=#
```

TESTING A CHAIN

- Session start
- Check for tasks
- Check if chain can be executed
- Execute chain task by task
  - Ignore errors if needed
- Check if chain is finished
- Commit chain transaction

```
2022-04-07 07:26:12.702 [INFO] Database connection established
2022-04-07 07:26:12.717 [INFO] Accepting asynchronous chains execution requests...
2022-04-07 07:26:12.727 [INFO] [count:0] Retrieve scheduled chains to run @reboot
2022-04-07 07:26:12.737 [INFO] [count:0] Retrieve interval chains to run
2022-04-07 07:26:13.151 [INFO] [count:501] Retrieve scheduled chains to run
2022-04-07 07:26:13.159 [INFO] [chain:12] Starting chain
2022-04-07 07:26:13.162 [INFO] [chain:8] Starting chain
2022-04-07 07:26:13.171 [INFO] [chain:8] [task:10] Starting task
2022-04-07 07:26:13.171 [INFO] [chain:12] [task:14] Starting task
2022-04-07 07:26:13.198 [INFO] [chain:12] [task:14] Task executed successfully
2022-04-07 07:26:13.276 [INFO] [chain:8] [task:10] Task executed successfully
2022-04-07 07:26:13.628 [INFO] [chain:2] Starting chain
2022-04-07 07:26:13.635 [INFO] [chain:2] [task:2] Starting task
2022-04-07 07:26:13.658 [INFO] [chain:10] Starting chain
2022-04-07 07:26:13.667 [INFO] [chain:10] [task:12] Starting task
2022-04-07 07:26:13.783 [INFO] [chain:9] Starting chain
2022-04-07 07:26:13.790 [INFO] [chain:9] [task:11] Starting task
2022-04-07 07:26:13.795 [INFO] [chain:7] Starting chain
2022-04-07 07:26:13.796 [INFO] [chain:16] Starting chain
2022-04-07 07:26:13.802 [INFO] [chain:7] [task:9] Starting task
2022-04-07 07:26:13.803 [INFO] [chain:16] [task:18] Starting task
2022-04-07 07:26:13.803 [INFO] [chain:12] Chain executed successfully
2022-04-07 07:26:13.811 [INFO] [chain:18] Starting chain
2022-04-07 07:26:13.816 [INFO] [chain:18] [task:20] Starting task
2022-04-07 07:26:13.828 [INFO] [chain:11] Starting chain
2022-04-07 07:26:13.836 [INFO] [chain:11] [task:13] Starting task
2022-04-07 07:26:13.870 [INFO] [chain:4] Starting chain
...
```

Improvement ideas?

User input very much appreciated!

github.com/cybertec-postgresql/pg_timetable

#StandWithUkraine

Thanks

Don't be a stranger:

https://www.cybertec-postgresql.com/en/blog/

HOW CAN WE HELP?

- Send money to right orgs
- Hire Ukrainian people
- Stop business in russia
- Suppot Ukrainians
- Spread the truth

CYBERTEC
DATA SCIENCE & POSTGRESQL