



operating system analytics and host intrusion detection at scale

mike arpaia / facebook



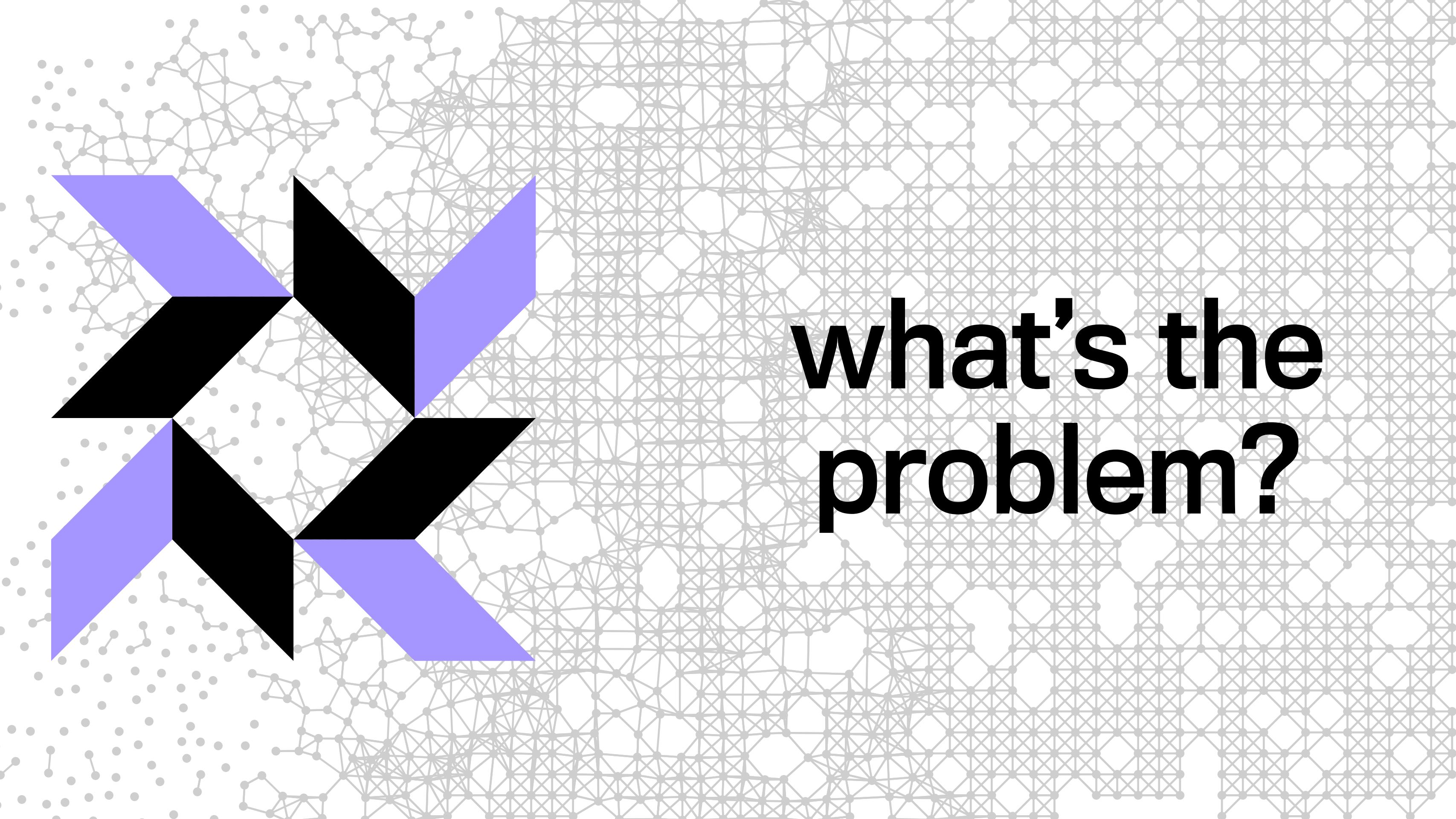
mimeframe / facebook



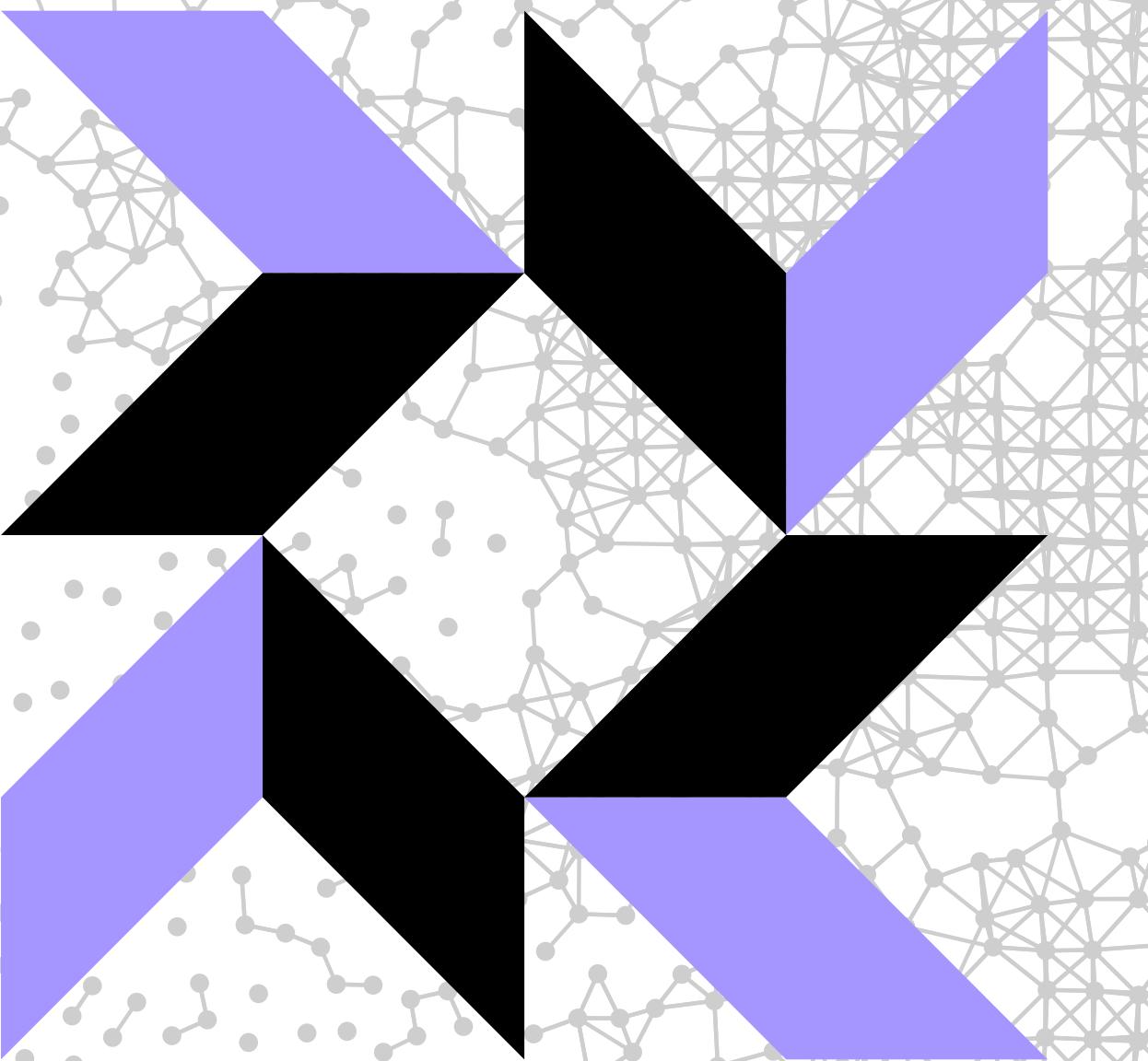
ted reed / facebook



javier marcos / facebook



**what's the
problem?**



we're all trying to catch attackers

it's a hard problem

- insider threats
 - espionage
- external threats
 - APT
 - hacktivists
 - mass malware
 - the list is endless

we're all deploying tools

we need help

- many of us have too many vendor products
- often times, a product solves too narrow of a use-case
- new use-case? new vendor
 - mo' money and mo' problems

we live in a windows centric world but, times are changing

- more OS X laptops
- most production infrastructure runs on Linux
- few are successfully instrumenting their OS X and Linux hosts
 - how would we solve that problem?



**desired
properties**

simple

performant and reliable

easy to integrate

flexible

simple

- no complex coding for users
- low-level details should be abstracted
- easy to use, deploy and maintain

performant and reliable

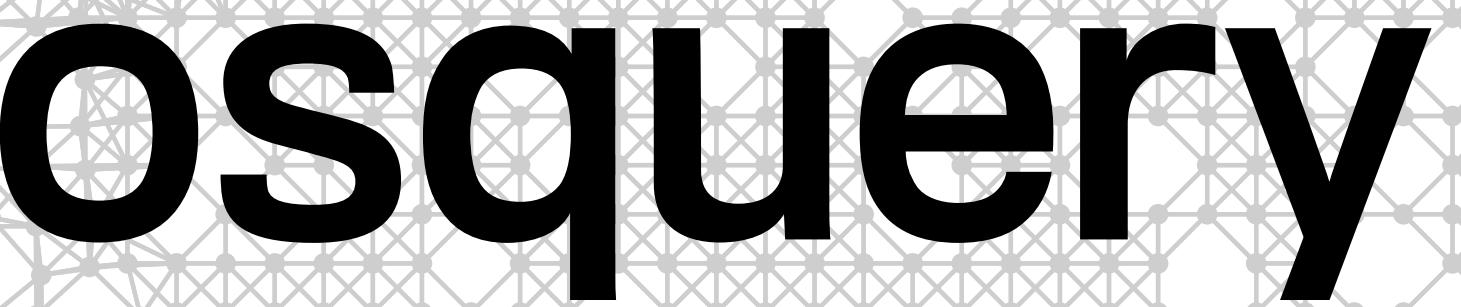
- host degradation is unacceptable
 - sane resource utilization over time
- company services should not be impacted
- extensive logging and metrics

easy to integrate

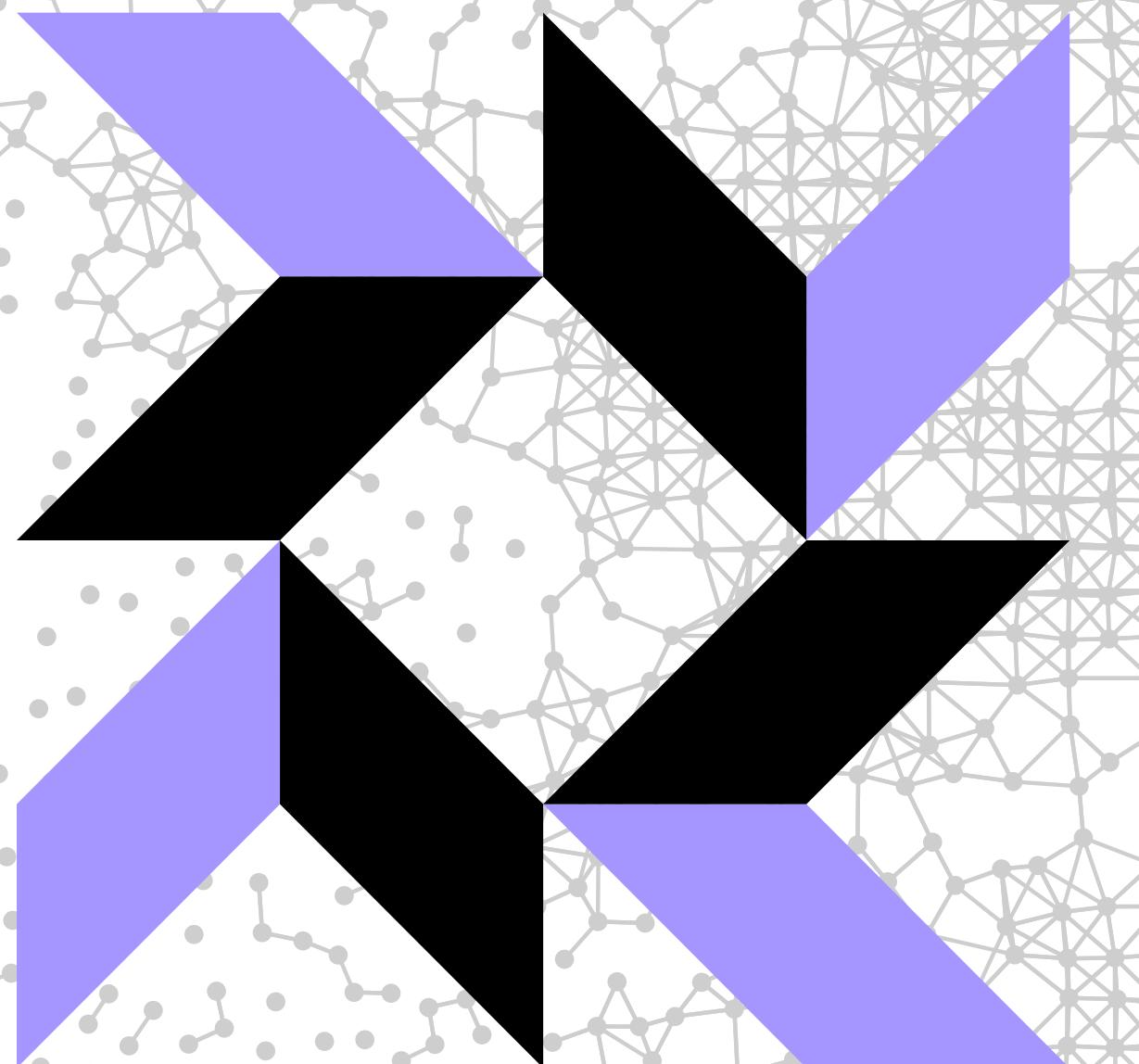
- every company has existing infrastructure
 - distributed configurations
 - real-time logging
 - data warehousing
- you should have the option to use existing infrastructure to help power your host instrumentation

flexible

- host instrumentation can help solve many problem domains
 - intrusion detection
 - vulnerability management
 - reliability
 - compliance
 - < insert domain here >
- having a single solution reduces cognitive overhead and time spent



osquery

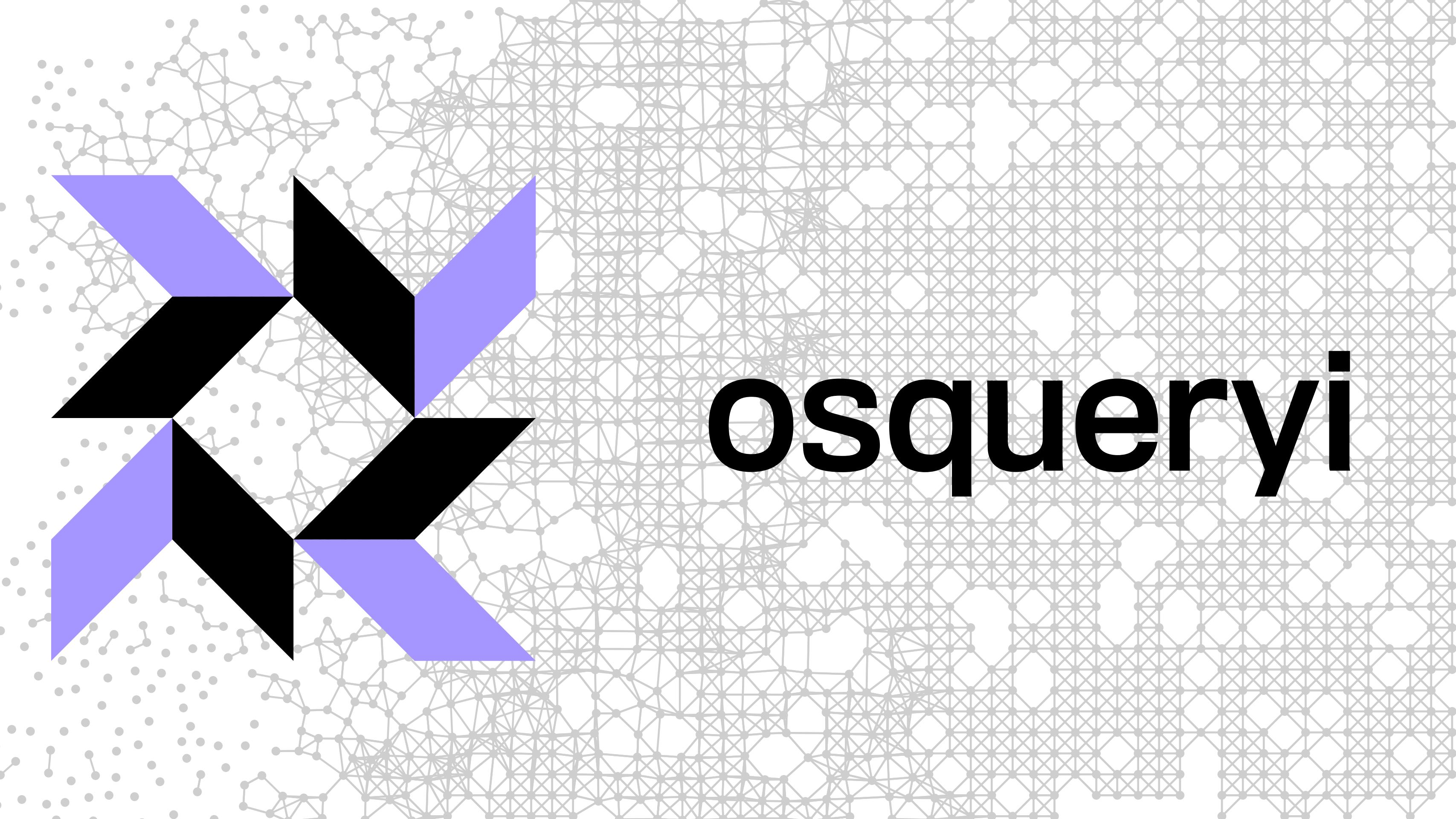


osquery

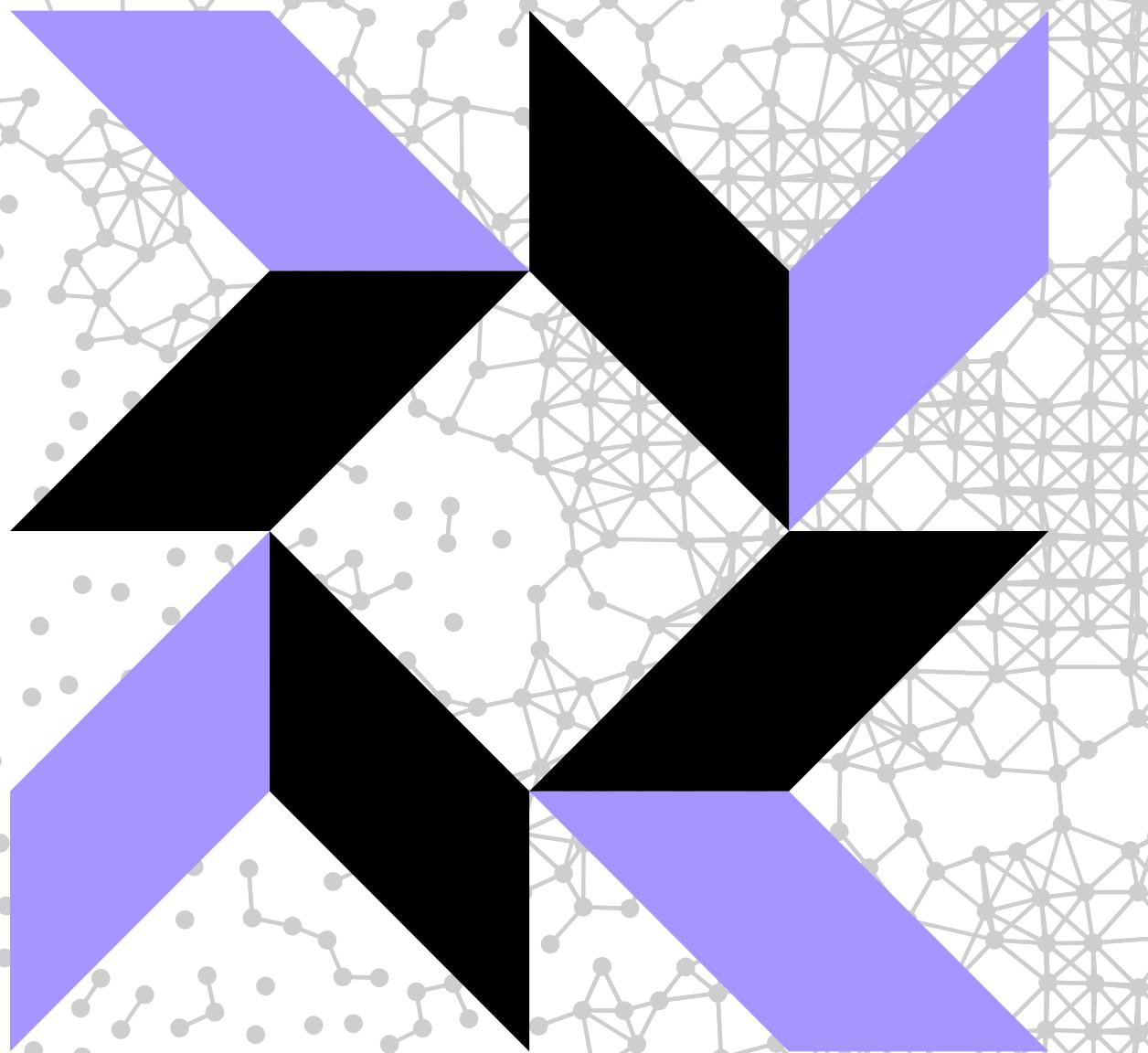
SQL for your infrastructure

use SQL queries to explore OS state

- running processes
- loaded kernel modules
- active network connections
- route table
- firewall settings
- installed software
- and more



osquery



1. osqueryi (osqueryi)

```
[marpaia-mbp] ~ osqueryi
osquery - being built, with love, at Facebook
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
osquery> SELECT name, program || program_arguments AS executable
...> FROM launchd
...> WHERE (run_at_load = 'true' AND keep_alive = 'true')
...> AND (program != '' OR program_arguments != '')
...> AND executable LIKE '/System/%';

+-----+
| name          | executable
+-----+
| com.apple.backupd-auto.plist | /System/Library/CoreServices/backupd.bundle/Contents/Resources/backupd-helper -launchd
| com.apple.logind.plist      | /System/Library/CoreServices/logind
| com.apple.mtmd.plist        | /System/Library/CoreServices/backupd.bundle/Contents/Resources/mtmd
| com.apple.mtmfs.plist       | /System/Library/CoreServices/backupd.bundle/Contents/Resources/mtmfs --tcp --resvport --listen localhost --oneshot --noportmap --nobrowse
| com.apple.revisiond.plist   | /System/Library/PrivateFrameworks/GenerationalStorage.framework/Versions/A/Support/revisiond
| com.apple.usbmuxd.plist     | /System/Library/PrivateFrameworks/MobileDevice.framework/Versions/A/Resources/usbmuxd -launchd
| com.apple.diagnostics_agent.plist | /System/Library/CoreServices/diagnostics_agent
+-----+
osquery>
```

LaunchDaemons which run a binary at boot

```
1. marpaia@marpaia-mbp: ~ (zsh)
~ (zsh) ⌘1
[marpaia-mbp] ~ echo "SELECT path, pid FROM processes LIMIT 10;" | osqueryi

+-----+-----+
| path | pid |
+-----+-----+
| /usr/local/bin/osqueryi | 9850 |
| /System/Library/PrivateFrameworks/Heimdal.framework/Helpers/kcm | 9825 |
| /System/Library/Frameworks/Python.framework/Versions/2.7/Resources/Python.app/Contents/MacOS/Python | 9774 |
| /bin/sleep | 9541 |
| /bin/zsh | 9348 |
| /System/Library/Frameworks/OpenGL.framework/Versions/A/Libraries/CVMCompiler | 9344 |
| /System/Library/Frameworks/OpenGL.framework/Versions/A/Libraries/CVMCompiler | 9343 |
| /usr/libexec/opendirectoryd | 8968 |
| /Applications/Google Chrome.app/Contents/Versions/38.0.2125.104/Google Chrome Helper.app/Contents/MacOS/Google Chrome Helper | 8615 |
| /usr/sbin/spindump | 8557 |
+-----+-----+
[marpaia-mbp] ~
```

running processes

[marpaia-mbp] ~ osqueryi

osquery - being built, with love, at Facebook

Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.

```
osquery> SELECT DISTINCT
...>   process.name,
...>   process.pid,
...>   process.parent,
...>   process.on_disk,
...>   process.phys_footprint,
...>   listening.port,
...>   listening.protocol,
...>   listening.address
...> FROM processes AS process
...> JOIN listening_ports AS listening
...> ON process.pid = listening.pid;
```

name	pid	parent	on_disk	phys_footprint	port	protocol	address
TextMate	8133	1	1	54149120	52698	6	0000:0000:0000:0000:0000:0000:0001:
SpotifyWebHelp	569	1	1	5505024	4370	6	127.0.0.1
SpotifyWebHelp	569	1	1	5505024	4380	6	127.0.0.1
2BUA8C4S2C.com.	557	1	1	25423872	6258	6	127.0.0.1
2BUA8C4S2C.com.	557	1	1	25423872	6258	6	0000:0000:0000:0000:0000:0000:0001:
2BUA8C4S2C.com.	557	1	1	25423872	6263	6	127.0.0.1
2BUA8C4S2C.com.	557	1	1	25423872	6263	6	0000:0000:0000:0000:0000:0000:0001:
ARDAgent	471	1	1	7933952	3283	17	0000:0000:0000:0000:0000:0000:0000:
ARDAgent	471	1	1	7933952	3283	17	0.0.0.0
SystemUIServer	458	1	1	23773184	49366	17	0.0.0.0
Spotify	450	1	1	234070016	4381	6	127.0.0.1
Spotify	450	1	1	234070016	4371	6	127.0.0.1
Spotify	450	1	1	234070016	8099	6	127.0.0.1
Spotify	450	1	1	234070016	57621	17	0.0.0.0
Spotify	450	1	1	234070016	57621	6	0.0.0.0

```
osquery> █
```

processes listening on ports

many tables are available

more tables are being written every day

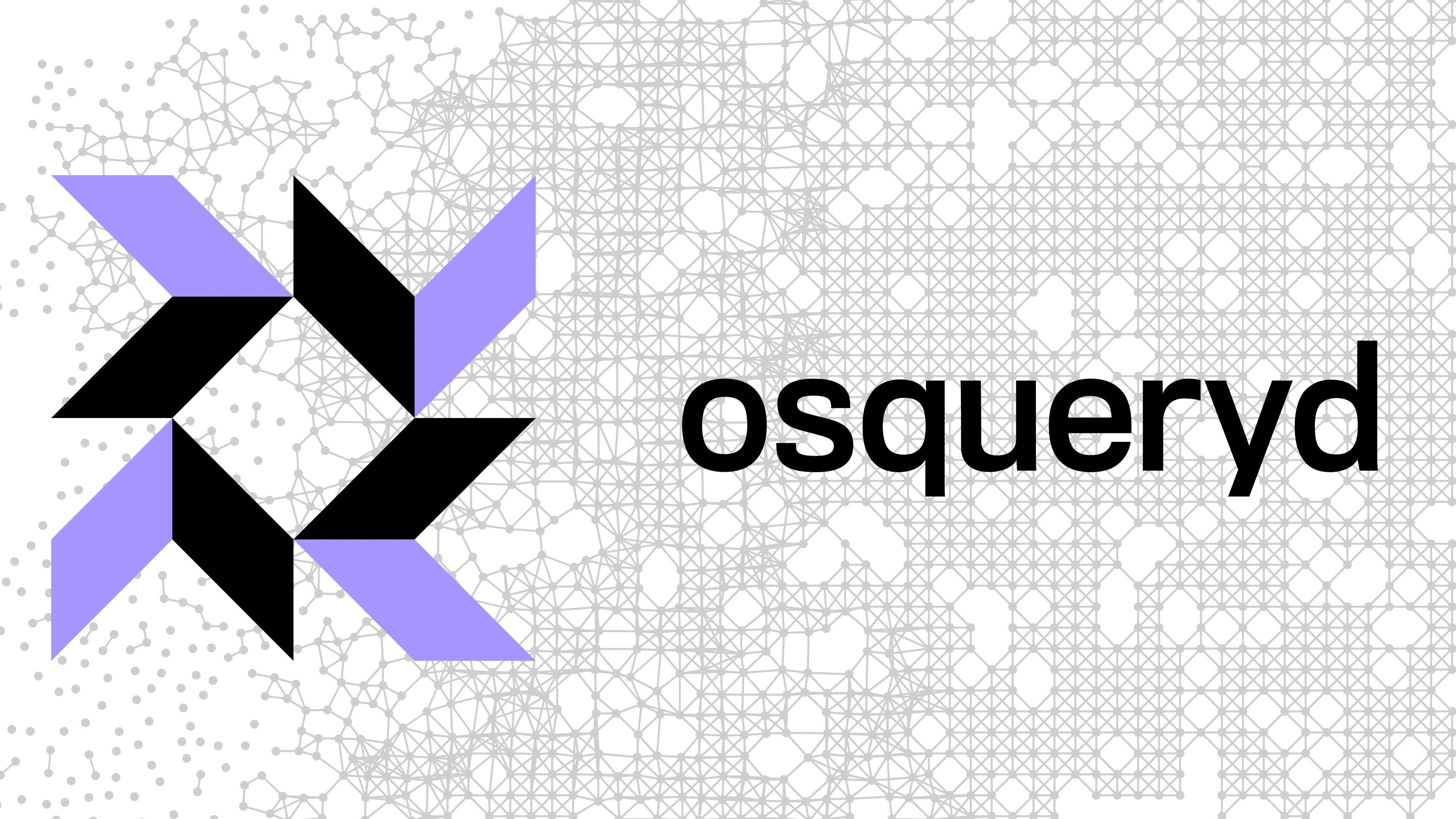
- alf
- alf_exceptions
- alf_explicit_auths
- alf_services
- apps
- ca_certs
- cpuid
- etc_hosts
- groups
- homebrew_packages
- interface_addresses
- interface_details
- kextstat
- last
- launchd
- listening_ports
- nvram
- osx_version
- passwd_changes
- processes
- routes
- suid_bin
- time
- users

use simple tables, together

osquery enables complex analysis by allowing users to join and aggregate across several simple tables

- simple tables have many advantages
 - easier to write
 - easier to maintain
 - can be used in many contexts





osqueryd

osqueryd

daemon for low-level host monitoring

know how the results of a query change over time

- schedule a query on your hosts via a config
- the daemon takes care of periodically executing your queries
 - buffers results to disk and generates a log of state changes
 - logs results for aggregation and analytics

host event pub/sub stream

event-based operating system introspection

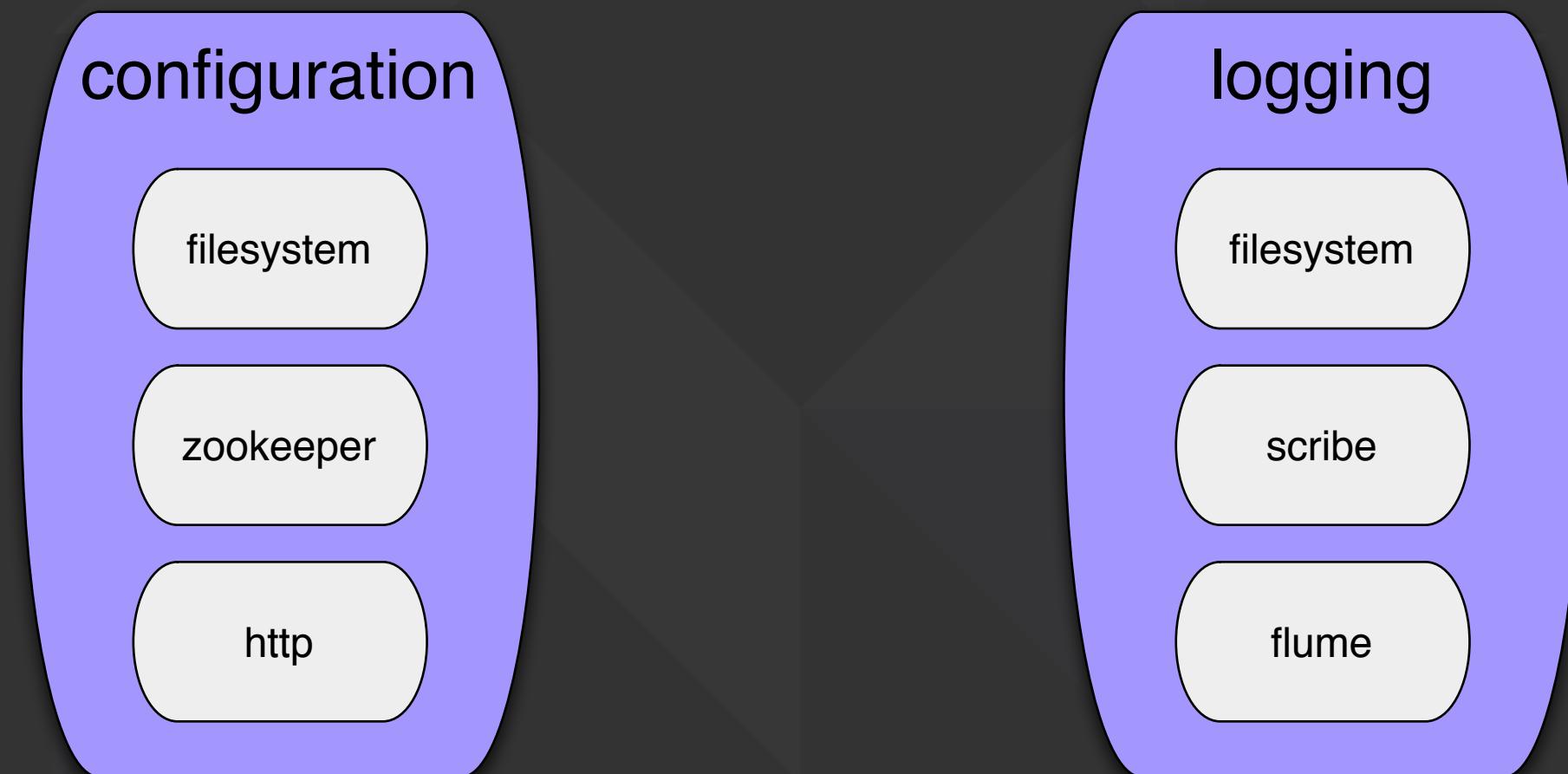
subscribe to key OS events to create dynamically growing tables

- subscribe to “publishers”
 - filesystem changes (inotify, FSEvents)
 - network setting changes (SCNetwork)
- query the history of your host, as it evolves

plugin system

for config distribution and data infrastructure

- simple plugin API
- specify your plugins at runtime with a command-line flag



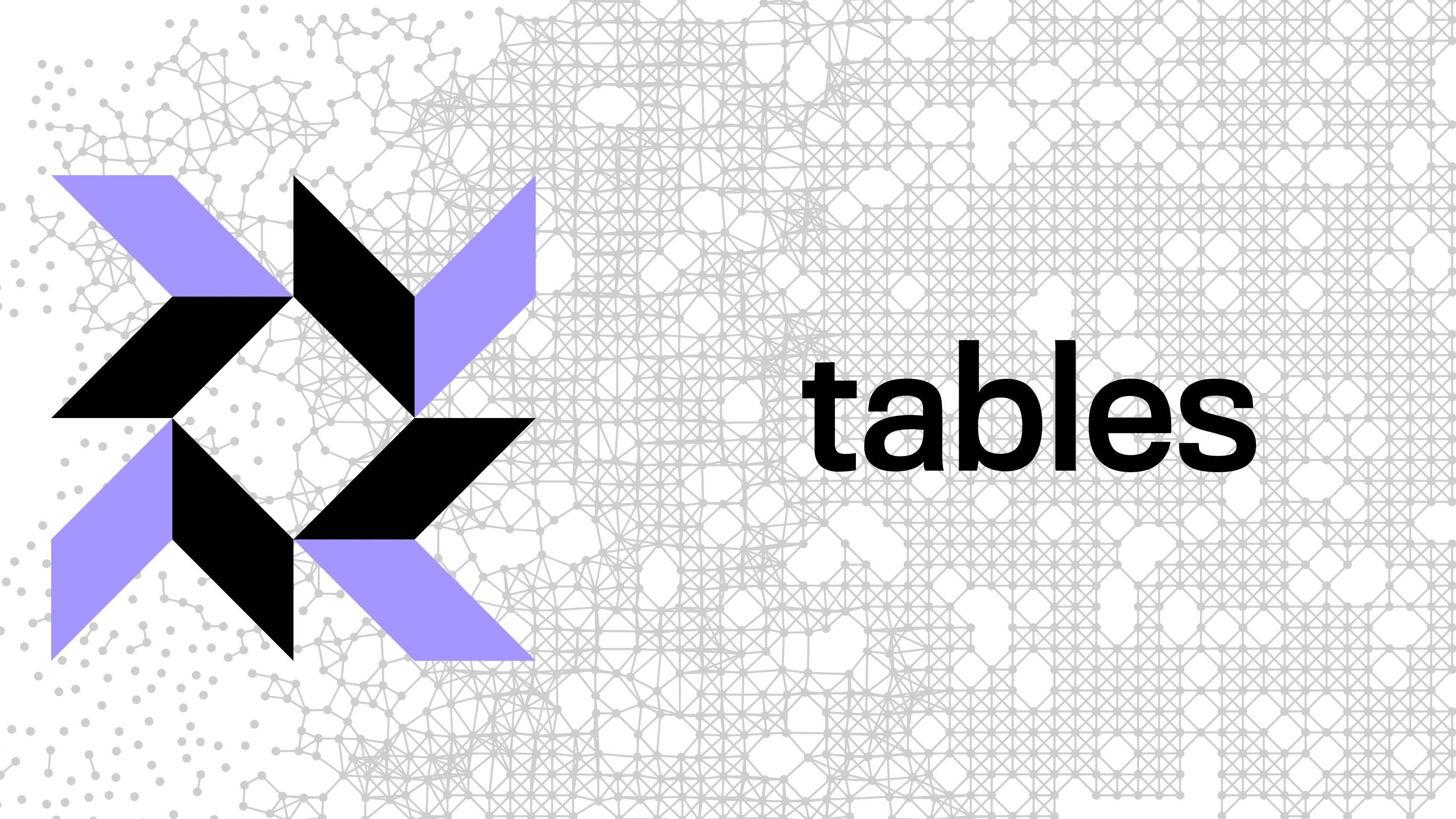
```
#include "osquery/logger/plugin.h"
#include <glog/logging.h>

namespace osquery {

class GlogPlugin : public LoggerPlugin {
public:
    Status logString(const std::string& message) {
        LOG(INFO) << message;
        return Status(0, "OK");
    }
};

REGISTER_LOGGER_PLUGIN("glog", std::make_shared<osquery::GlogPlugin>());
}
```

registering a glog plugin



tables

creating tables is easy

easily define what your tables “look like” in Python and use C++ to implement what a full-table scan would return

- the Python is used to generate faster C++ code transparently
- you write a single C++ function which implements a full-table scan

2. marpaia@marpaia-mbp: ~ (zsh)

~ (zsh)⌘1

```
[marpaia-mbp] ~ echo "SELECT * FROM time;" | osqueryi
```

hour	minutes	seconds
0	58	30

```
[marpaia-mbp] ~ █
```

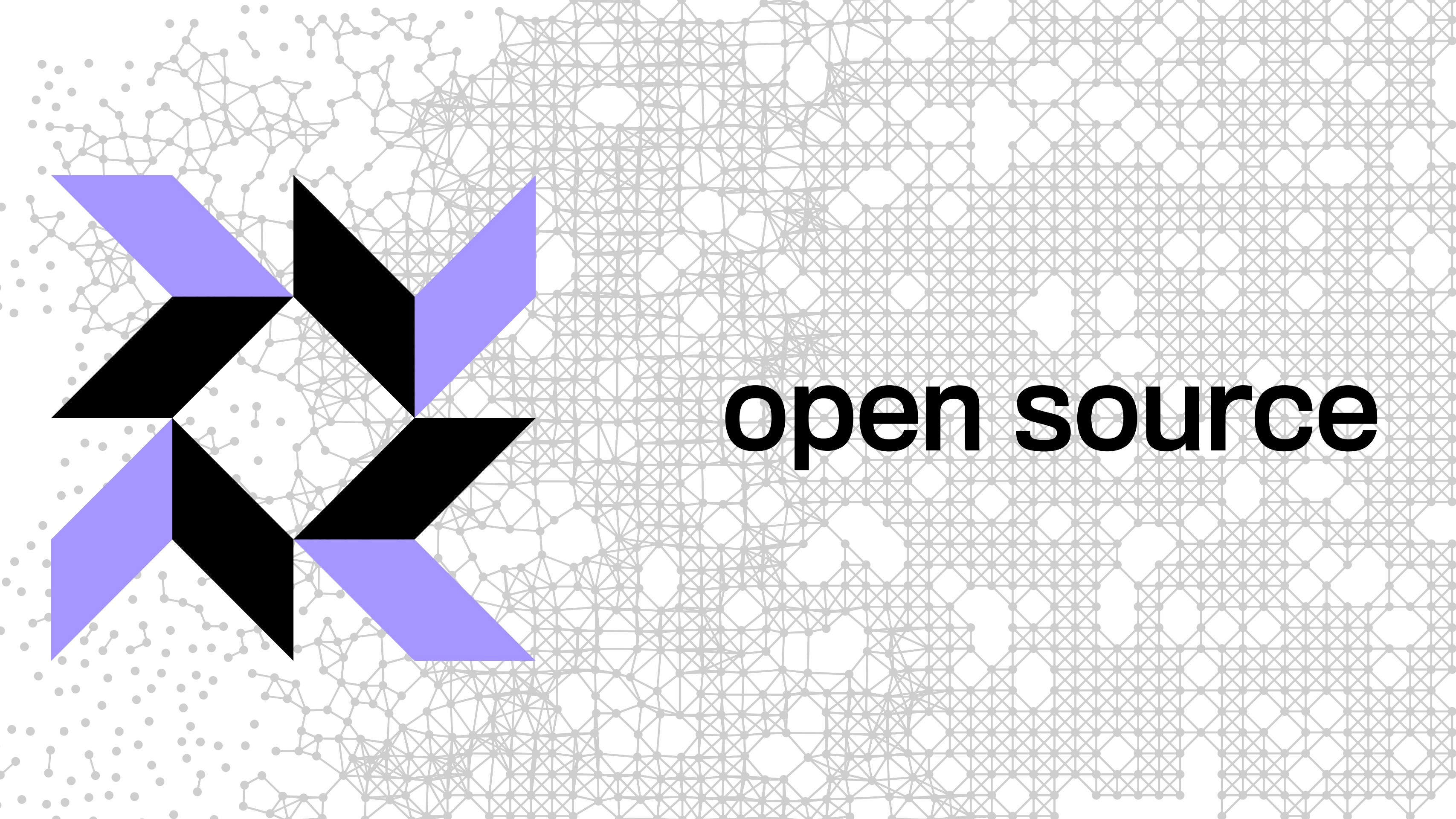
```
table_name("time")
schema([
    Column("hour", INTEGER),
    Column("minutes", INTEGER),
    Column("seconds", INTEGER),
])
implementation("time@genTime")
```

```
namespace osquery {
namespace tables {

QueryData genTime() {
    QueryData results;
    struct tm* now = localtime(time(0));

    Row r;
    r["hour"] = INTEGER(now->tm_hour);
    r["minutes"] = INTEGER(now->tm_min);
    r["seconds"] = INTEGER(now->tm_sec);
    results.push_back(r);

    return results;
}
}
}
```



open source

work on osquery with us

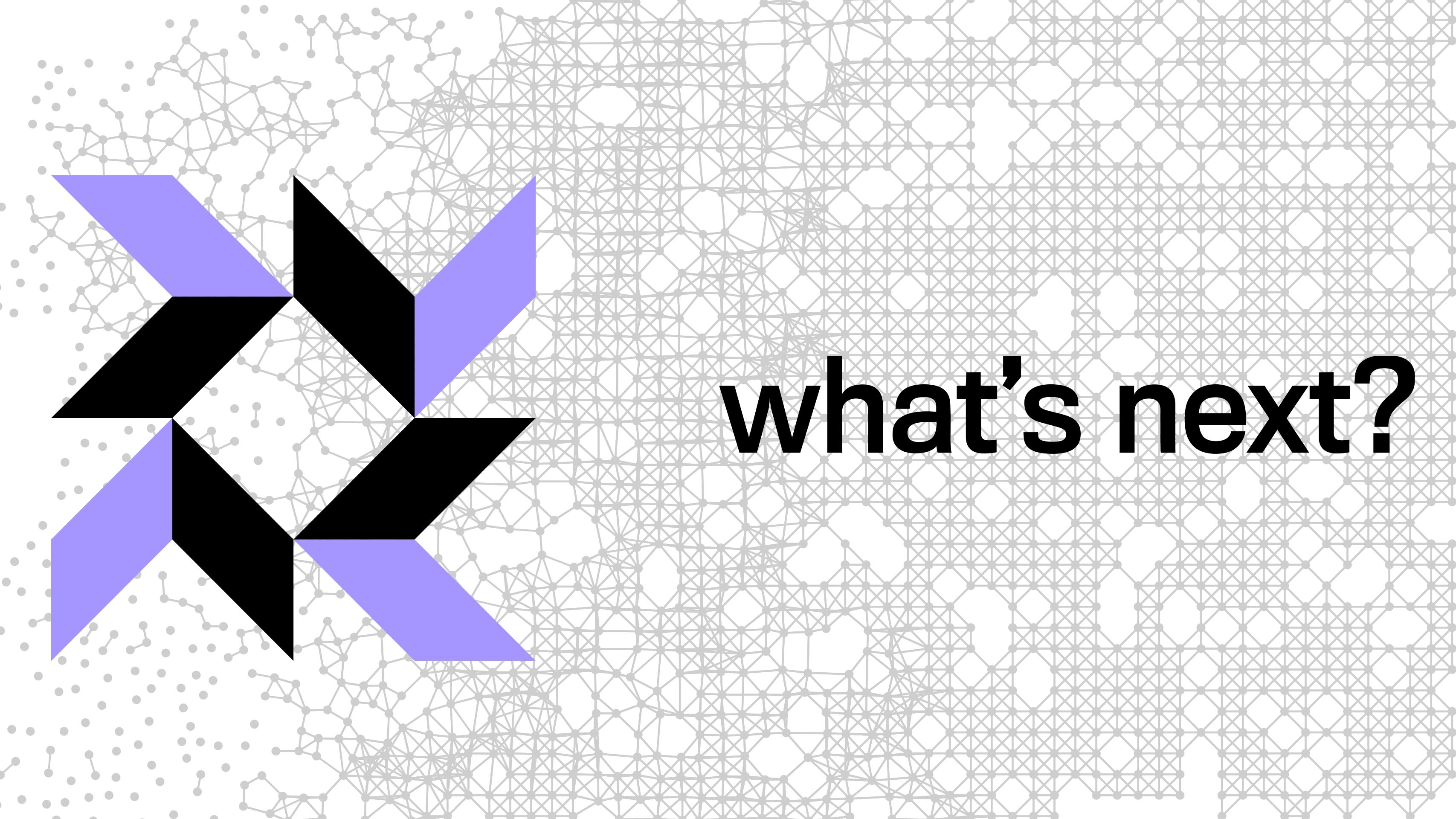
all development happens in the open, on GitHub

the problem that osquery solves isn't unique to facebook

- <https://github.com/facebook/osquery>
- <http://osquery.io>

this journey is 1% finished: get involved

- we're excited to take on future challenges in the open
- let's build together



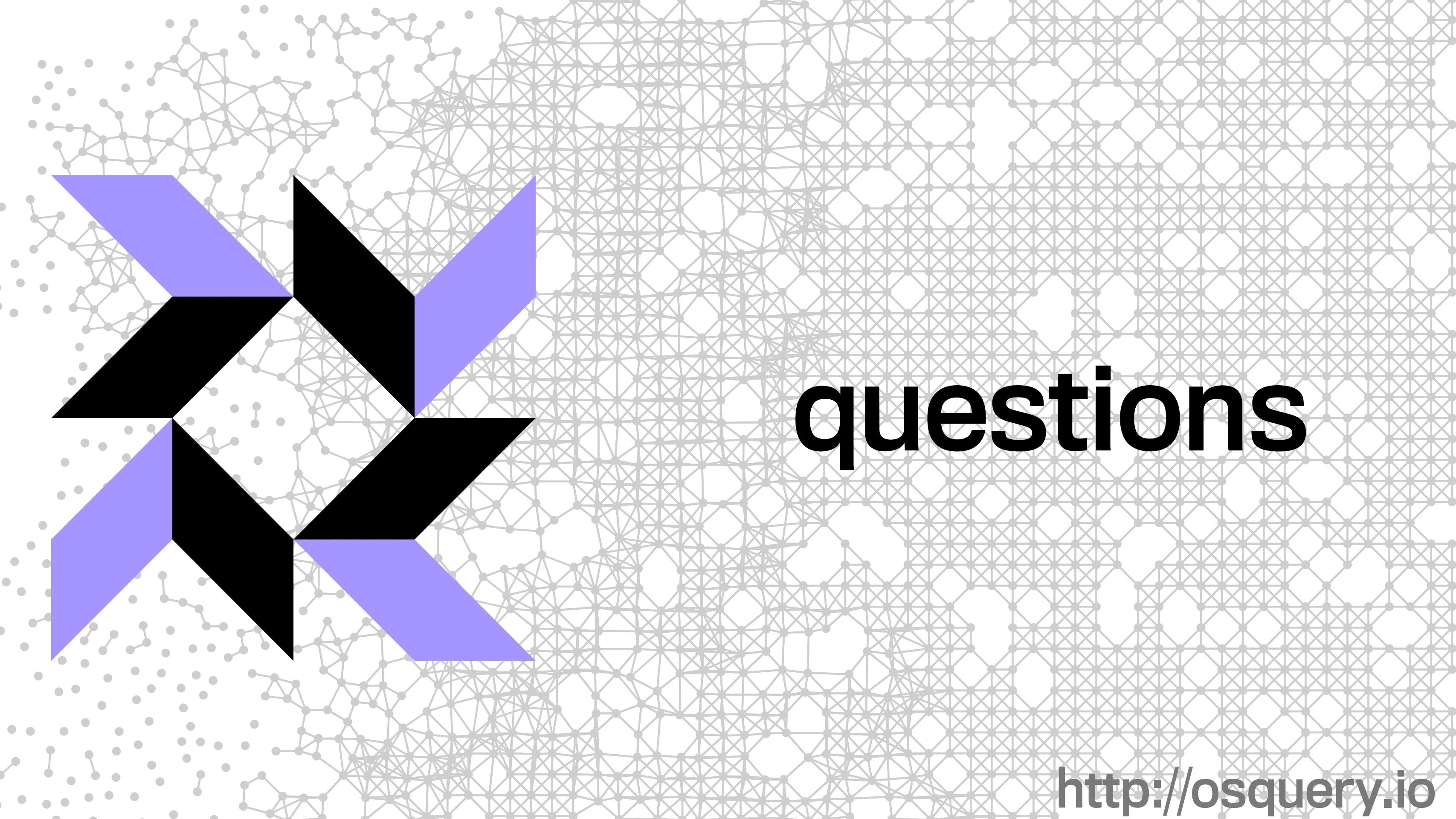
what's next?

what we're working on

contribute and help us build an awesome tool

there's a team of great engineers at facebook that are actively working on making osquery awesome for everyone

- more tables
- ad-hoc remote queries
- kernel modules for lower-level behavior monitoring
- deep systems visibility



questions

<http://osquery.io>