



Exploration, Monitoring and Security with osquery



Zach Wasserman
Cofounder & Principal Engineer

Preparing for Interactivity

- Please download and install osquery so that you can follow along with the examples in this presentation.
- osquery.io/downloads (Mac pkg is ~12MB)
- Note Homebrew support is limited. Please download and install the package `osqueryi`.

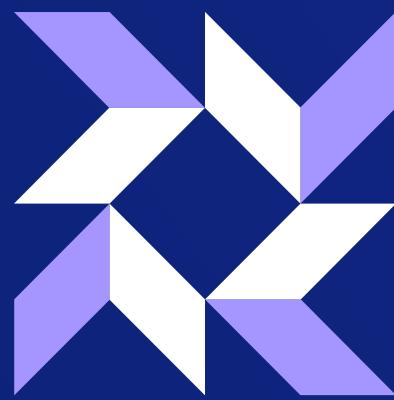
Who Am I?

- Core osquery contributor since its inception in 2014.
- Cofounder & Principal Engineer at Kolide.
- Currently building open-source (Kolide Fleet, Kolide Launcher) and commercial (Kolide Cloud) software in the osquery ecosystem

The Problem

The Problem

- Sysadmins and security folks have a huge number of sources for the data relevant to their operations and decision-making.
- How can we reliably access this data to get an understanding of the system state in the present moment, and as it changes over time?



Introducing Osquery

Introducing Osquery

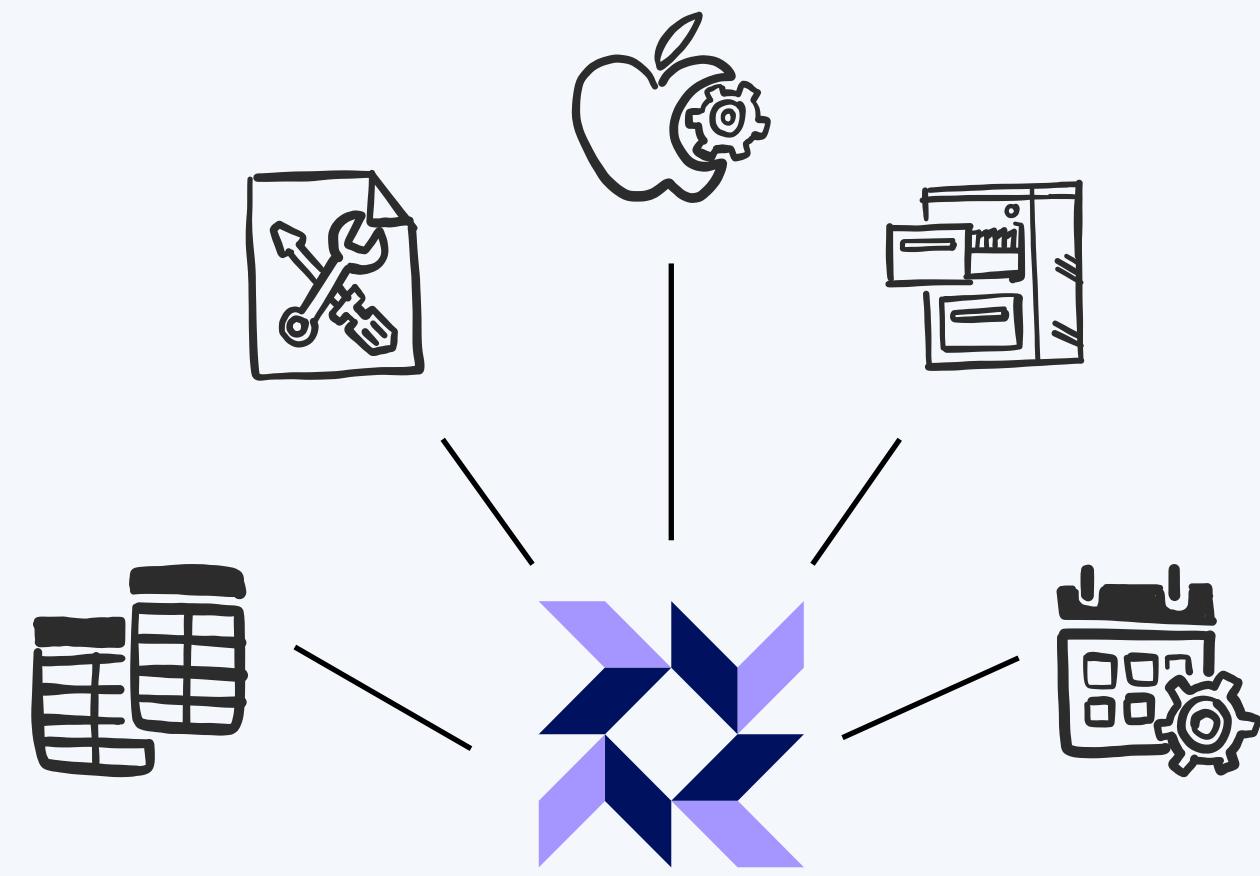
- Open-sourced by Facebook in 2014. Still supported by a core team at FB.
- 4,486+ commits, 231+ contributors, 12,707+ stars on Github
- Apache 2.0 License
- osquery.io

Osquery Goals

- First class support for macOS/Linux
- Enable non-developers to access and aggregate data across disparate sources
- Performance/reliability to deploy across corporate and production infrastructure

Unify disparate sources of information

- Flat files (/etc/hosts, /etc/crontab, ~/.ssh/known_hosts, etc.)
- SQLite files (/var/db/SystemPolicy [GateKeeper configuration], etc.)
- System APIs (Apple System Log, Keychain, SMC, CoreFoundation, etc.)
- Application APIs (Docker, Carbon Black, etc.)
- Event-based APIs (FSEvents, OpenBSM, etc.)
- Filesystem (Shared folders, file hashes, permissions, etc.)
- Plists (/Library/Managed\ Installs/* [Munki data], etc.)
- ... And more ...



The Power of SQL



osquery> SELECT * FROM...

account_policy_data	docker_container_mounts	keychain_items	prometheus_metrics
acpi_tables	docker_container_networks	known_hosts	python_packages
ad_config	docker_container_ports	last	quicklook_cache
alf	docker_container_processes	launchd	routes
alf_exceptions	docker_container_stats	launchd_overrides	safari_extensions
alf_explicit_auths	docker_containers	listening_ports	sandboxes
alf_services	docker_image_labels	load_average	shared_folders
app_schemes	docker_images	logged_in_users	sharing_preferences
apps	docker_info	magic	shell_history
apt_sources	docker_network_labels	managed_policies	signature
arp_cache	docker_networks	mdfind	sip_config
asl	docker_version	memory_devices	smbios_tables
augeas	docker_volume_labels	mounts	smc_keys
authorization_mechanisms	docker_volumes	nfs_shares	startup_items
authorizations	etc_hosts	nvram	sudoers
authorized_keys	etc_protocols	opera_extensions	suid_bin
block_devices	etc_services	os_version	system_controls
browser_plugins	event_taps	osquery_events	system_info
carbon_black_info	extended_attributes	osquery_extensions	temperature_sensors
carves	fan_speed_sensors	osquery_flags	time
certificates	file	osquery_info	time_machine_backups
chrome_extensions	file_events	osquery_packs	time_machine_destinations
cpu_time	firefox_addons	osquery_registry	uptime
cpuid	gatekeeper	osquery_schedule	usb_devices
crashes	gatekeeper_approved_apps	package_bom	user_events
crontab	groups	package_install_history	user_groups
cups_destinations	hardware_events	package_receipts	user_interaction_events
cups_jobs	hash	pci_devices	user_ssh_keys
curl	homebrew_packages	platform_info	users
curl_certificate	intel_me_info	plist	virtual_memory_info
device_file	interface_addresses	power_sensors	wifi_networks
device_firmware	interface_details	preferences	wifi_status
device_hash	iokit_devicetree	process_envs	wifi_survey
device_partitions	iokit_registry	process_events	xprotect_entries
disk_encryption	kernel_extensions	process_memory_map	xprotect_meta
disk_events	kernel_info	process_open_files	xprotect_reports
dns_resolvers	kernel_panic	process_open_sockets	yara
docker_container_labels	keychain_acls	processes	yara_events

The Power of SQL

- `select * from hosts; -- /etc/hosts`
- `select * from smc_keys; -- SMC`
- `select * from keychain_items; -- Keychain`
- `select * from file_events; -- FSEvents`
- `select * from hash where path = '/bin/bash'; -- File hashes`

```
osquery> SELECT u.username, g.gid, g.groupname FROM users u JOIN user_groups ug USING (uid) JOIN groups g ON ug.gid = g.gid WHERE uid > 500;
```

2. sudo osqueryi (osqueryi)

```
osquery> SELECT u.username, g.gid, g.groupname
...> FROM users u JOIN user_groups ug USING (uid) JOIN groups g ON ug.gid = g.gid
...> WHERE uid > 500;
+-----+-----+
| username | gid | groupname
+-----+-----+
| zwass    | 20  | staff
| zwass    | 501 | access_bpf
| zwass    | 12  | everyone
| zwass    | 61  | localaccounts
| zwass    | 79  | _appserverusr
| zwass    | 80  | admin
| zwass    | 81  | _appserveradm
| zwass    | 98  | _lpadmin
| zwass    | 702 | 2
| zwass    | 33  | _appstore
| zwass    | 100 | _lpoperator
| zwass    | 204 | _developer
| zwass    | 250 | _analyticsusers
| zwass    | 395 | com.apple.access_ftp
| zwass    | 398 | com.apple.access_screensharing
| zwass    | 399 | com.apple.access_ssh
| zwass    | 701 | 1
+-----+-----+
osquery> |
```

 HEROKU

facebook

UBER

NETFLIX

okta

 Dropbox

GitHub

DUO

Who's using osquery?

 airbnb

stripe

 Palantir

 ATLASSIAN

 CISCO

 KOLIDE

slack

 Square

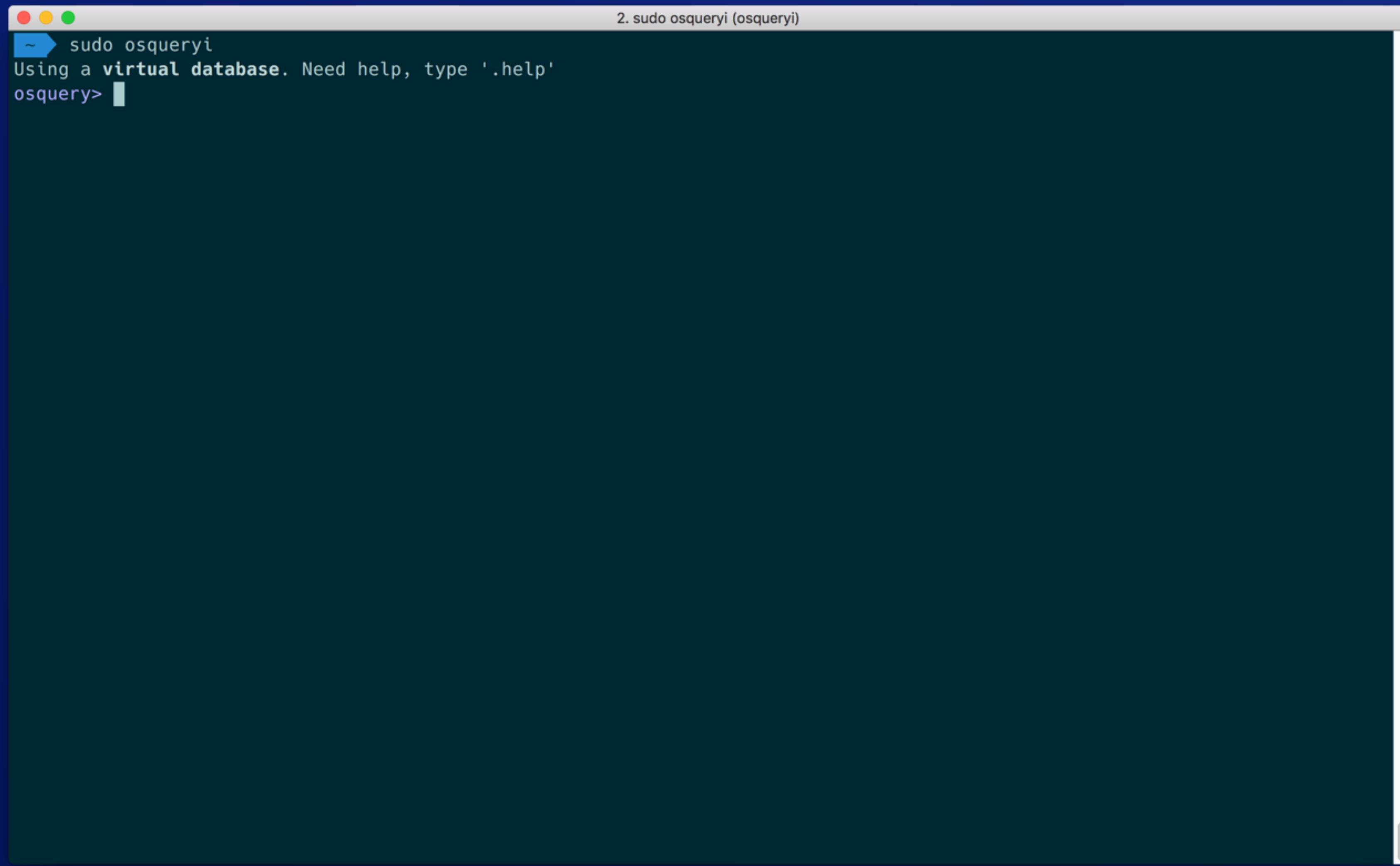
Digging In



osqueryi

- CLI and interactive shell for executing queries and viewing results
- Use this as a part of scripts, or for manual exploration
- After iterating on and understanding queries in osqueryi, evolve them to create monitoring via osqueryd (more later)

osqueryi



A screenshot of a terminal window titled "2. sudo osqueryi (osqueryi)". The window has a dark green background and white text. It displays the following command and its output:

```
~ ➔ sudo osqueryi
Using a virtual database. Need help, type '.help'
osquery> █
```

osqueryi

```
2. sudo osqueryi (osqueryi)
~ ➔ sudo osqueryi
Using a virtual database. Need help, type '.help'
osquery> SELECT * FROM time;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| weekday | year | month | day | hour | minutes | seconds | timezone | local_time | unix_time |
| datetime           | iso_8601           |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Friday   | 2018  | 6     | 8     | 2     | 48      | 11      | UTC      | 1528426091 | PDT          |
| 02:48:11 2018 UTC | 2018-06-08T02:48:11Z | 2018-06-08T02:48:11Z |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
osquery> .mode line
osquery> SELECT * FROM time;
weekday = Friday
year = 2018
month = 6
day = 8
hour = 2
minutes = 48
seconds = 31
timezone = UTC
local_time = 1528426111
local_timezone = PDT
unix_time = 1528426111
timestamp = Fri Jun 8 02:48:31 2018 UTC
datetime = 2018-06-08T02:48:31Z
iso_8601 = 2018-06-08T02:48:31Z
osquery>
```

osqueryi

The image shows a split-screen interface. On the left, a terminal window displays the output of the osqueryi command-line interface. On the right, a web browser window displays the osquery Schema documentation.

Terminal Output (osqueryi):

```
~ sudo osqueryi
Using a virtual database
osquery> SELECT * FROM
+-----+-----+
| weekday | year | month |
+-----+-----+
| Friday | 2018 | 6
02:48:11 2018 UTC | 20
+-----+-----+
osquery> .mode line
osquery> SELECT * FROM
    weekday = Friday
    year = 2018
    month = 6
    day = 8
    hour = 2
    minutes = 48
    seconds = 31
    timezone = UTC
    local_time = 152842
local_timezone = PDT
    unix_time = 152842
    timestamp = Fri Jun 08 02:48:11 2018
    datetime = 2018-06-08T02:48:11Z
    iso_8601 = 2018-06-08T02:48:11Z
osquery>
```

Web Browser (osquery Schema Documentation):

The web page title is "osquery | Schema". The header includes links for HOME, SCHEMA (which is selected), BLOG, DOCS, GITHUB, and DOWNLOADS. A dropdown menu shows "Osquery Version: 3.2.6 (current)".

The main content area is titled "154 Tables". It features a sidebar with a list of table names, including account_policy_data, acpi_tables, ad_config, alf, alf_exceptions, alf_explicit_auths, alf_services, app_schemes, apps, apt_sources, arp_cache, asl, augeas, authorization_mechanisms, authorizations, authorized_keys, block_devices, browser_plugins, carbon_black_info, carves, certificates, chrome_extensions, cpu_time, cpuid, crashes, crontab, and many others.

Two tables are detailed:

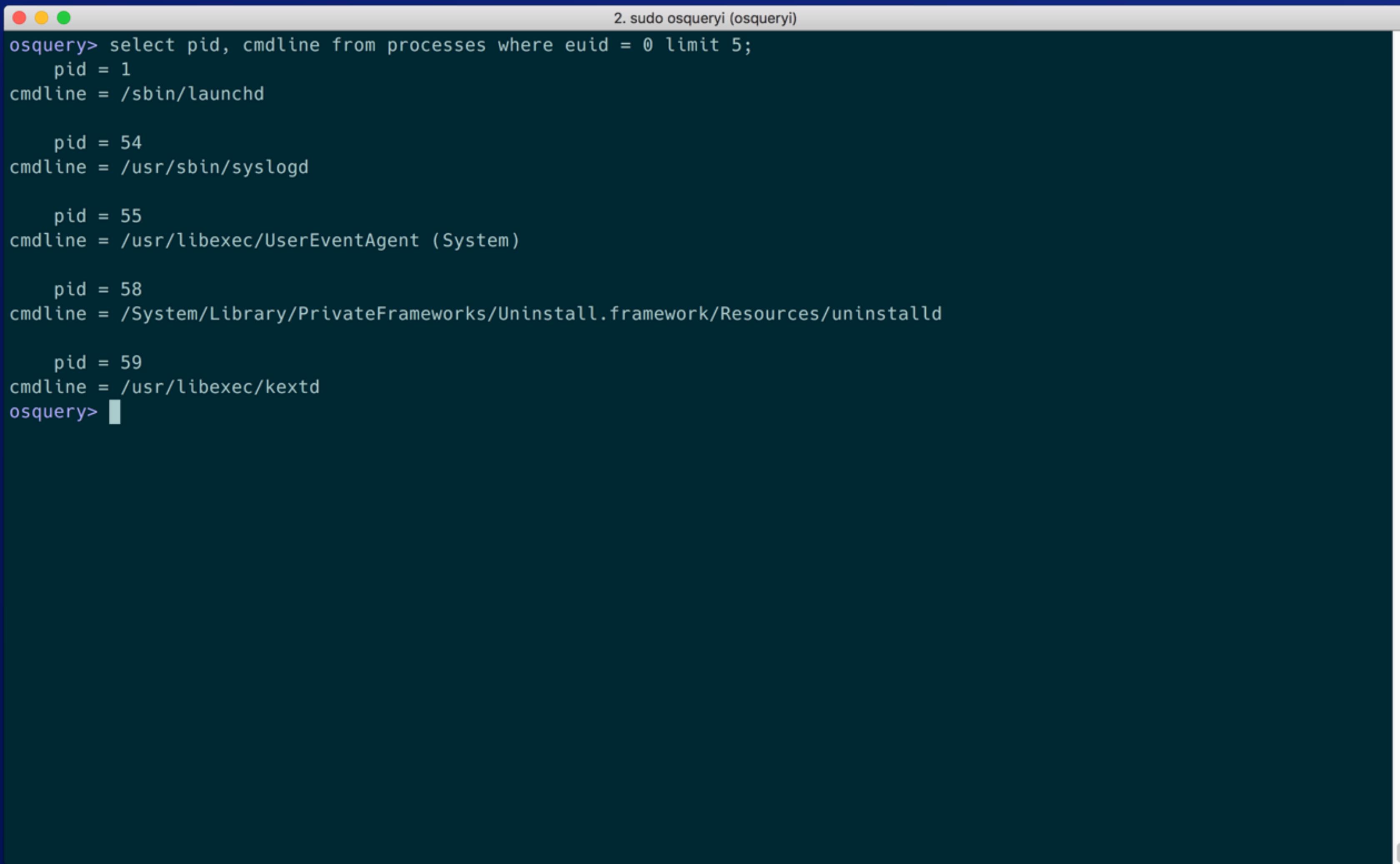
account_policy_data (Mac OS X):

COLUMN	TYPE	DESCRIPTION
uid	BIGINT	User ID
creation_time	DOUBLE	When the account was first created
failed_login_count	BIGINT	The number of times the user failed to login with the correct password. Resets after a correct password is entered
failed_login_timestamp	DOUBLE	The time of the last failed login attempt. Resets after a correct password is entered
password_last_set_time	DOUBLE	The time the password was last changed

acpi_tables (Firmware ACPI functional table common metadata and content):

COLUMN	TYPE	DESCRIPTION
name	TEXT	ACPI table name
size	INTEGER	Size of compiled table data
md5	TEXT	MD5 hash of table content

osqueryi



A terminal window titled "2. sudo osqueryi (osqueryi)" displaying a query result. The query is:

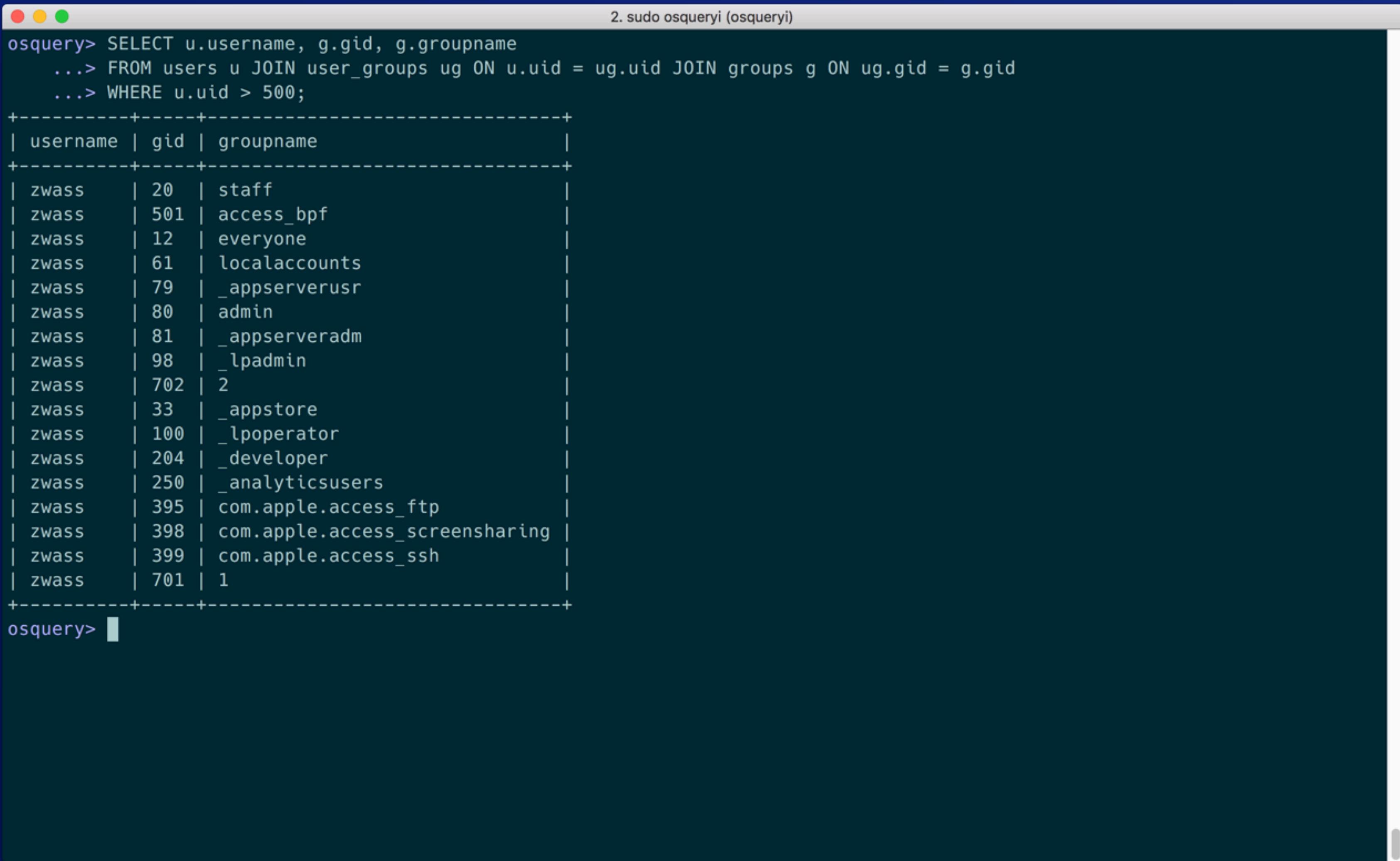
```
osquery> select pid, cmdline from processes where euid = 0 limit 5;
```

The output shows five processes:

- pid = 1
cmdline = /sbin/launchd
- pid = 54
cmdline = /usr/sbin/syslogd
- pid = 55
cmdline = /usr/libexec/UserEventAgent (System)
- pid = 58
cmdline = /System/Library/PrivateFrameworks/Uninstall.framework/Resources/uninstallld
- pid = 59
cmdline = /usr/libexec/kextd

osquery> █

osqueryi

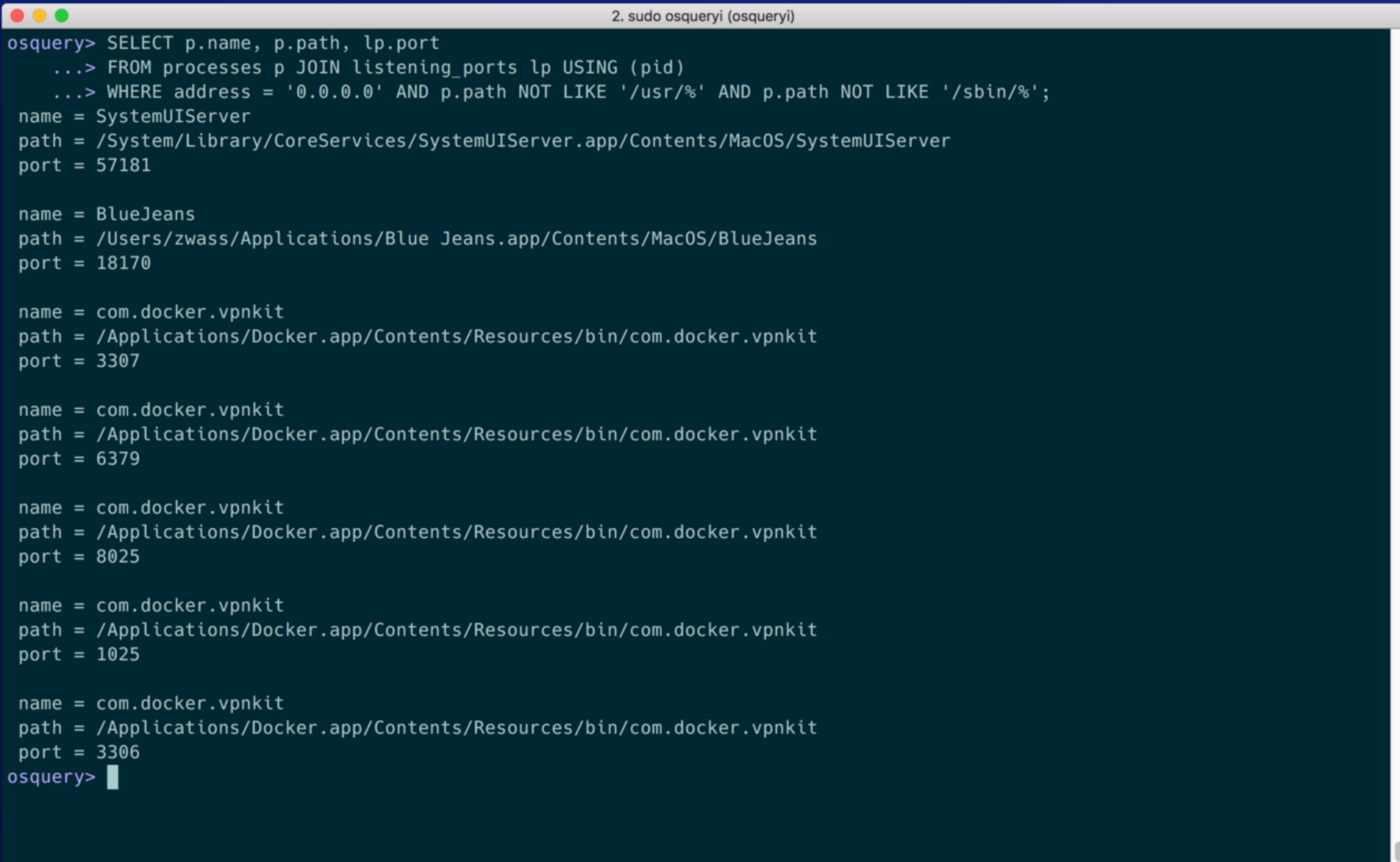


A screenshot of a terminal window titled "2. sudo osqueryi (osqueryi)". The window displays the output of an SQL query run against the osquery database. The query selects the username, group ID (gid), and group name for users whose uid is greater than 500. The results are presented in a tabular format with three columns: username, gid, and groupname. The data shows various system accounts and their corresponding groups.

username	gid	groupname
zwass	20	staff
zwass	501	access_bpf
zwass	12	everyone
zwass	61	localaccounts
zwass	79	_appserverusr
zwass	80	admin
zwass	81	_appserveradm
zwass	98	_lpadmin
zwass	702	2
zwass	33	_appstore
zwass	100	_lpoperator
zwass	204	_developer
zwass	250	_analyticsusers
zwass	395	com.apple.access_ftp
zwass	398	com.apple.access_screensharing
zwass	399	com.apple.access_ssh
zwass	701	1

How can we interactively investigate system activity using osqueryi?

osqueryi



A screenshot of a terminal window titled "2. sudo osqueryi (osqueryi)". The window displays a series of database queries and their results. The queries are as follows:

```
osquery> SELECT p.name, p.path, lp.port
...> FROM processes p JOIN listening_ports lp USING (pid)
...> WHERE address = '0.0.0.0' AND p.path NOT LIKE '/usr/%' AND p.path NOT LIKE '/sbin/%';
name = SystemUIServer
path = /System/Library/CoreServices/SystemUIServer.app/Contents/MacOS/SystemUIServer
port = 57181

name = BlueJeans
path = /Users/zwass/Applications/Blue Jeans.app/Contents/MacOS/BlueJeans
port = 18170

name = com.docker.vpnkit
path = /Applications/Docker.app/Contents/Resources/bin/com.docker.vpnkit
port = 3307

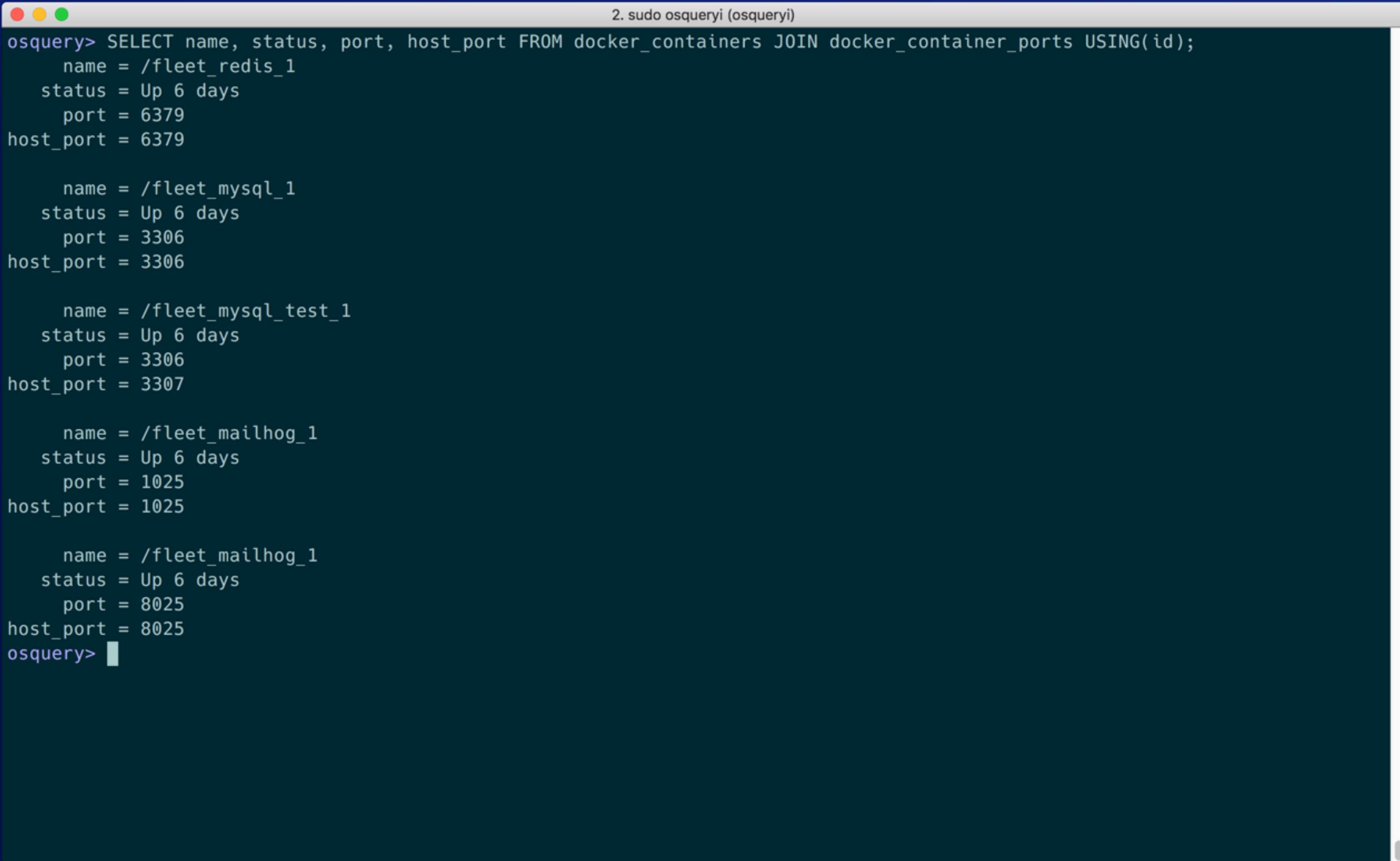
name = com.docker.vpnkit
path = /Applications/Docker.app/Contents/Resources/bin/com.docker.vpnkit
port = 6379

name = com.docker.vpnkit
path = /Applications/Docker.app/Contents/Resources/bin/com.docker.vpnkit
port = 8025

name = com.docker.vpnkit
path = /Applications/Docker.app/Contents/Resources/bin/com.docker.vpnkit
port = 1025

name = com.docker.vpnkit
path = /Applications/Docker.app/Contents/Resources/bin/com.docker.vpnkit
port = 3306
osquery>
```

osqueryi



2. sudo osqueryi (osqueryi)

```
osquery> SELECT name, status, port, host_port FROM docker_containers JOIN docker_container_ports USING(id);
    name = /fleet_redis_1
    status = Up 6 days
    port = 6379
host_port = 6379

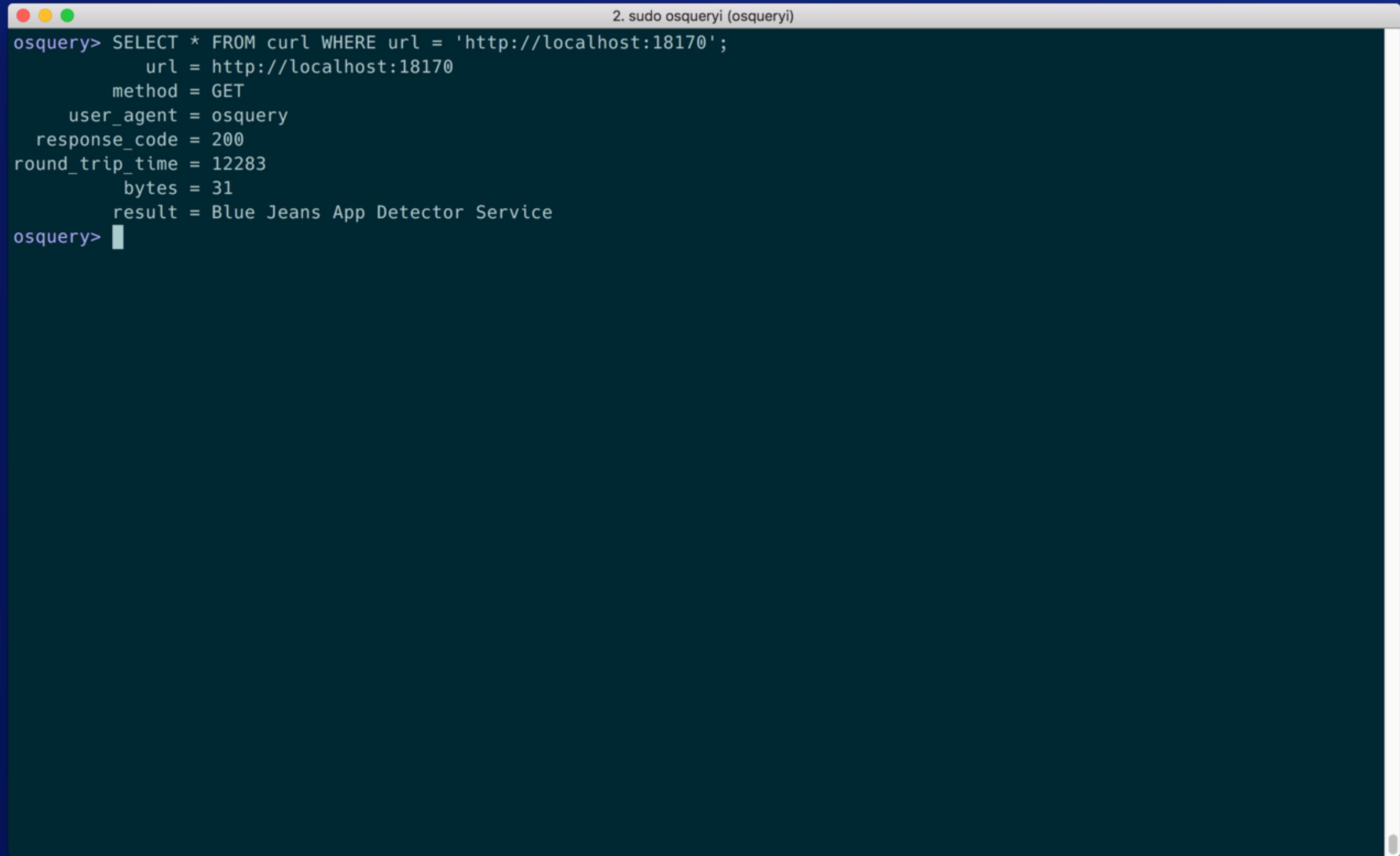
    name = /fleet_mysql_1
    status = Up 6 days
    port = 3306
host_port = 3306

    name = /fleet_mysql_test_1
    status = Up 6 days
    port = 3306
host_port = 3307

    name = /fleet_mailhog_1
    status = Up 6 days
    port = 1025
host_port = 1025

    name = /fleet_mailhog_1
    status = Up 6 days
    port = 8025
host_port = 8025
osquery> |
```

osqueryi



A screenshot of a terminal window titled "2. sudo osqueryi (osqueryi)". The window shows the output of a SQL query run against the osquery database. The query is:

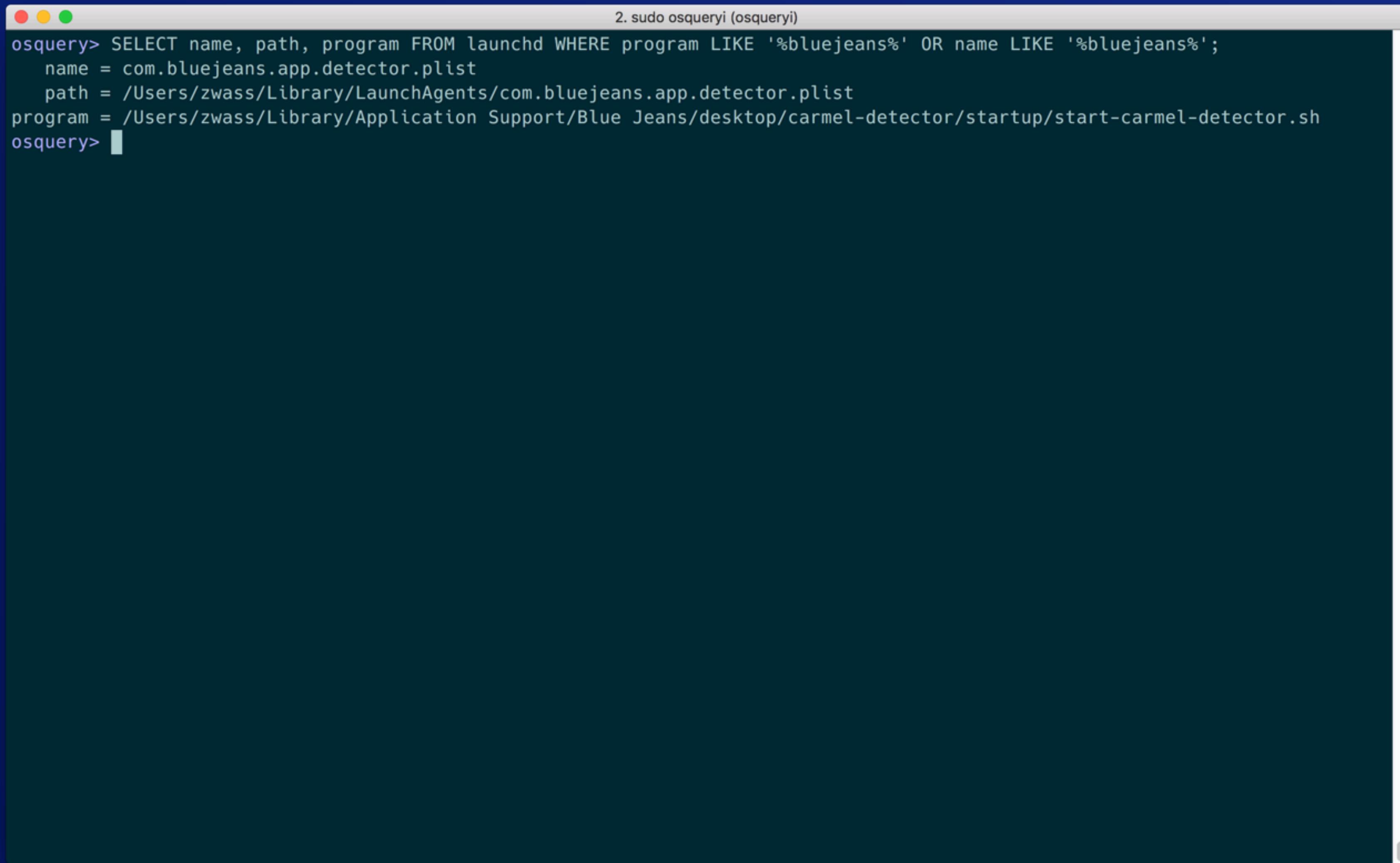
```
osquery> SELECT * FROM curl WHERE url = 'http://localhost:18170';
```

The results are as follows:

Column	Value
url	http://localhost:18170
method	GET
user_agent	osquery
response_code	200
round_trip_time	12283
bytes	31
result	Blue Jeans App Detector Service

The terminal prompt "osquery>" is visible at the bottom left.

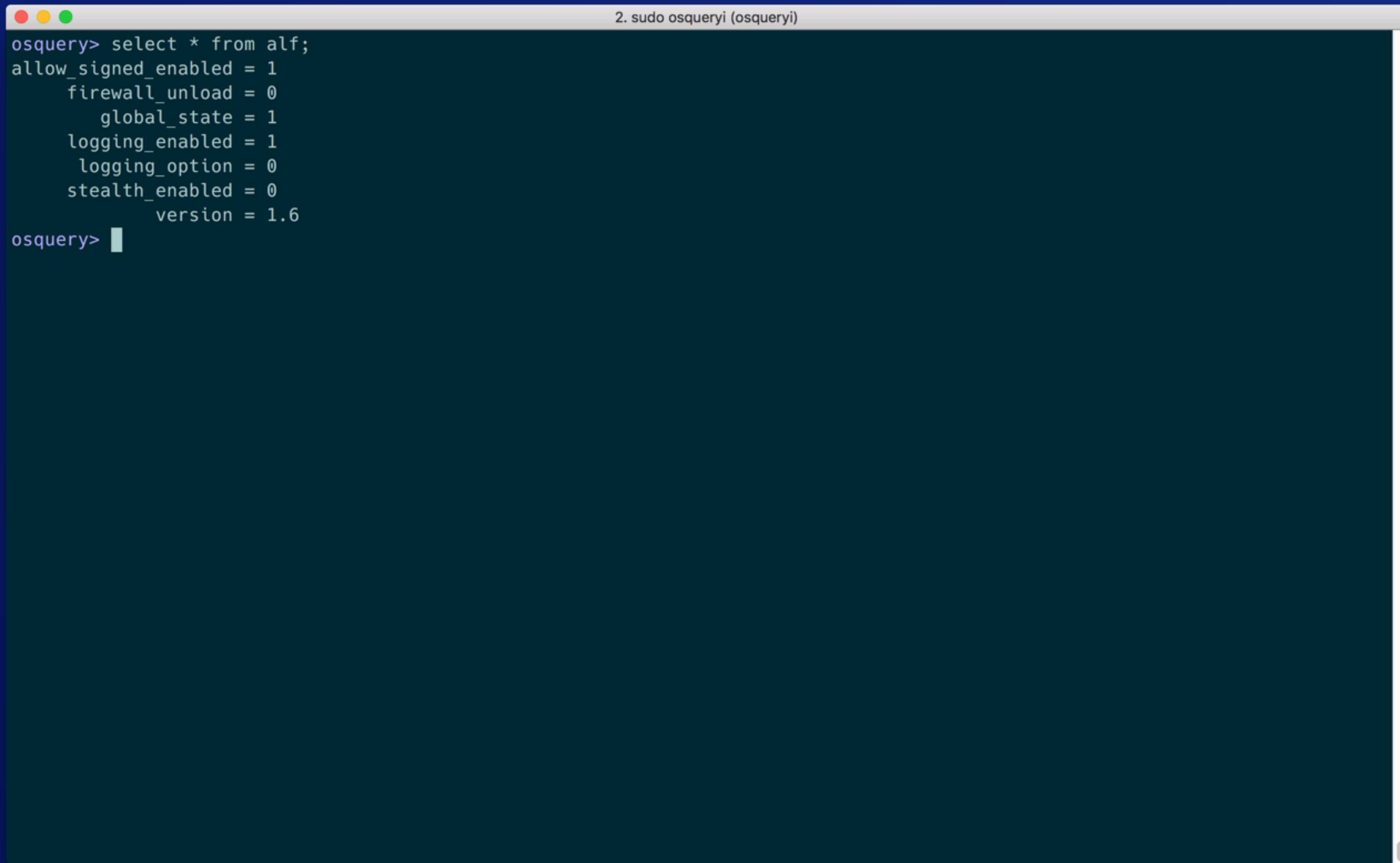
osqueryi



A screenshot of a terminal window titled "2. sudo osqueryi (osqueryi)". The window displays the following SQL query results:

```
osquery> SELECT name, path, program FROM launchd WHERE program LIKE '%bluejeans%' OR name LIKE '%bluejeans%';
  name = com.bluejeans.app.detector.plist
  path = /Users/zwass/Library/LaunchAgents/com.bluejeans.app.detector.plist
program = /Users/zwass/Library/Application Support/Blue Jeans/desktop/carmel-detector/startup/start-carmel-detector.sh
osquery>
```

osqueryi

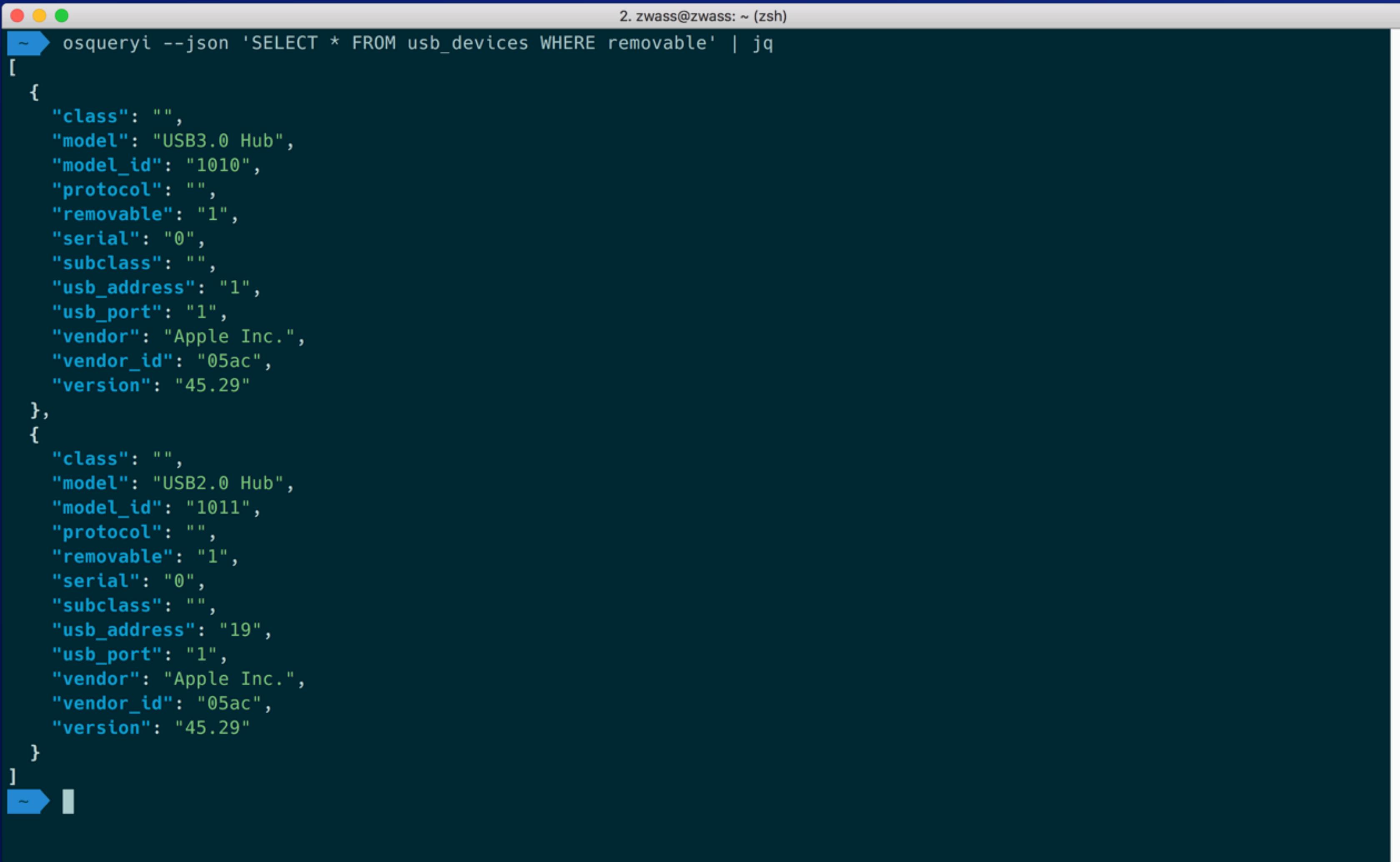


A screenshot of a terminal window titled "2. sudo osqueryi (osqueryi)". The window has a dark background and light-colored text. It displays the following command and its results:

```
osquery> select * from alf;
allow_signed_enabled = 1
firewall_unload = 0
global_state = 1
logging_enabled = 1
logging_option = 0
stealth_enabled = 0
version = 1.6
osquery>
```

Get structured output for scripting

osqueryi

A screenshot of a terminal window titled "2. zwass@zwass: ~ (zsh)". The window contains the following command and its JSON output:

```
osqueryi --json 'SELECT * FROM usb_devices WHERE removable' | jq
```

```
[  
  {  
    "class": "",  
    "model": "USB3.0 Hub",  
    "model_id": "1010",  
    "protocol": "",  
    "removable": "1",  
    "serial": "0",  
    "subclass": "",  
    "usb_address": "1",  
    "usb_port": "1",  
    "vendor": "Apple Inc.",  
    "vendor_id": "05ac",  
    "version": "45.29"  
  },  
  {  
    "class": "",  
    "model": "USB2.0 Hub",  
    "model_id": "1011",  
    "protocol": "",  
    "removable": "1",  
    "serial": "0",  
    "subclass": "",  
    "usb_address": "19",  
    "usb_port": "1",  
    "vendor": "Apple Inc.",  
    "vendor_id": "05ac",  
    "version": "45.29"  
  }]  
~
```

The terminal has a dark blue background and white text. The command is run in the background, indicated by a blue progress bar at the bottom.

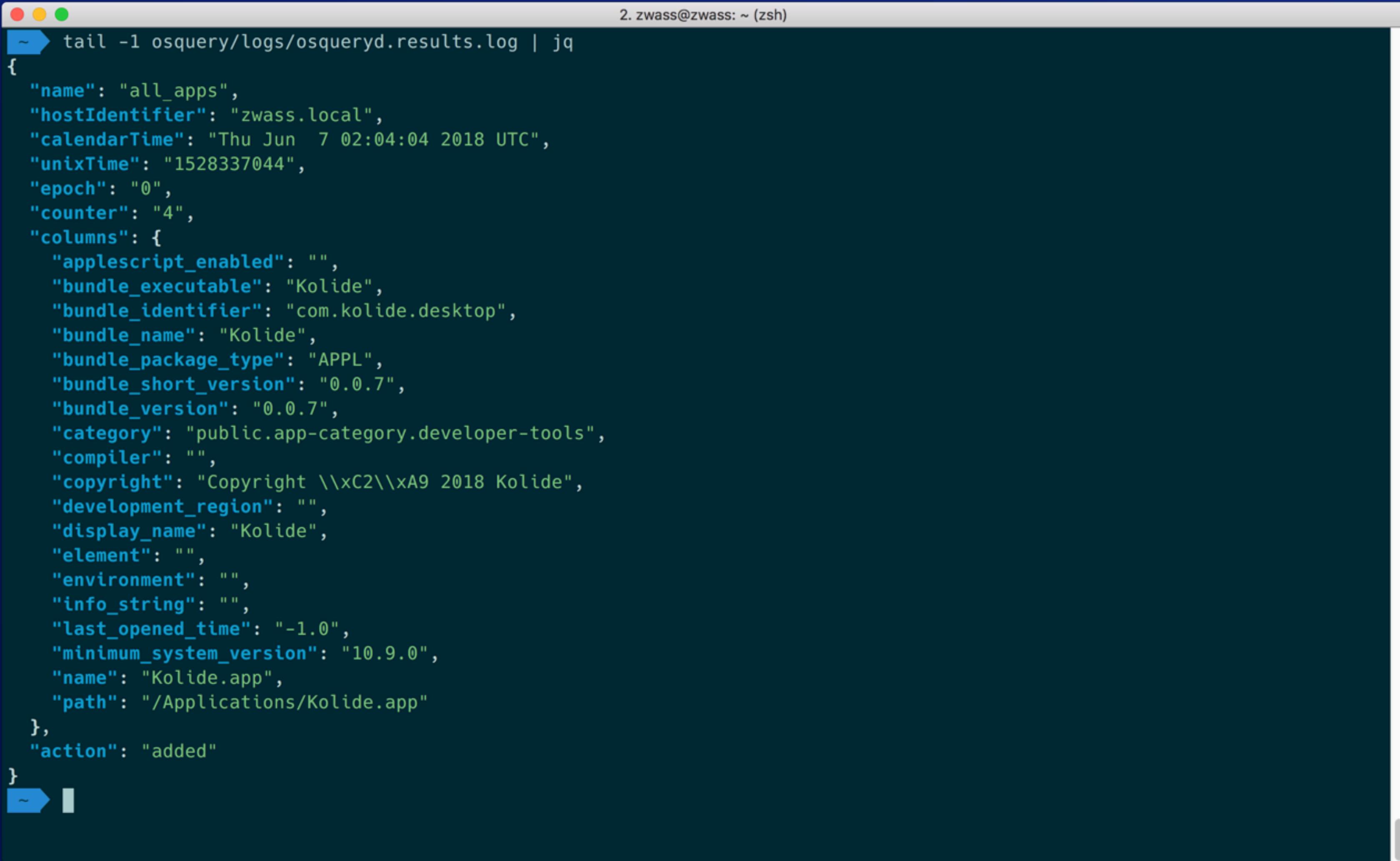
osqueryd

- Schedule queries for continuous results
- Differential engine to see how state changes over time
- Event-based tables ensure that data is not lost even when queries run on an interval

osqueryd

```
{  
  "schedule": {  
    "all_apps": {  
      "query": "SELECT * FROM apps",  
      "interval": 60  
    }  
  }  
}
```

osqueryd

A screenshot of a macOS terminal window titled "2. zwass@zwass: ~ (zsh)". The window contains a command-line session. The command entered is "tail -1 osquery/logs/osqueryd.results.log | jq". The output is a JSON object representing an event from the osqueryd log. The object has several fields, including "name": "all_apps", "hostIdentifier": "zwass.local", and a "columns" field which is itself a complex JSON structure describing an application bundle.

```
tail -1 osquery/logs/osqueryd.results.log | jq
{
  "name": "all_apps",
  "hostIdentifier": "zwass.local",
  "calendarTime": "Thu Jun 7 02:04:04 2018 UTC",
  "unixTime": "1528337044",
  "epoch": "0",
  "counter": "4",
  "columns": {
    "applescript_enabled": "",
    "bundle_executable": "Kolide",
    "bundle_identifier": "com.kolide.desktop",
    "bundle_name": "Kolide",
    "bundle_package_type": "APPL",
    "bundle_short_version": "0.0.7",
    "bundle_version": "0.0.7",
    "category": "public.app-category.developer-tools",
    "compiler": "",
    "copyright": "Copyright \u00c2\u00a9 2018 Kolide",
    "development_region": "",
    "display_name": "Kolide",
    "element": "",
    "environment": "",
    "info_string": "",
    "last_opened_time": "-1.0",
    "minimum_system_version": "10.9.0",
    "name": "Kolide.app",
    "path": "/Applications/Kolide.app"
  },
  "action": "added"
}
```

osqueryd

```
{  
  "schedule": {  
    "hardware_events": {  
      "query": "SELECT * FROM hardware_events",  
      "interval": 60  
    }  
  }  
}
```

osqueryd

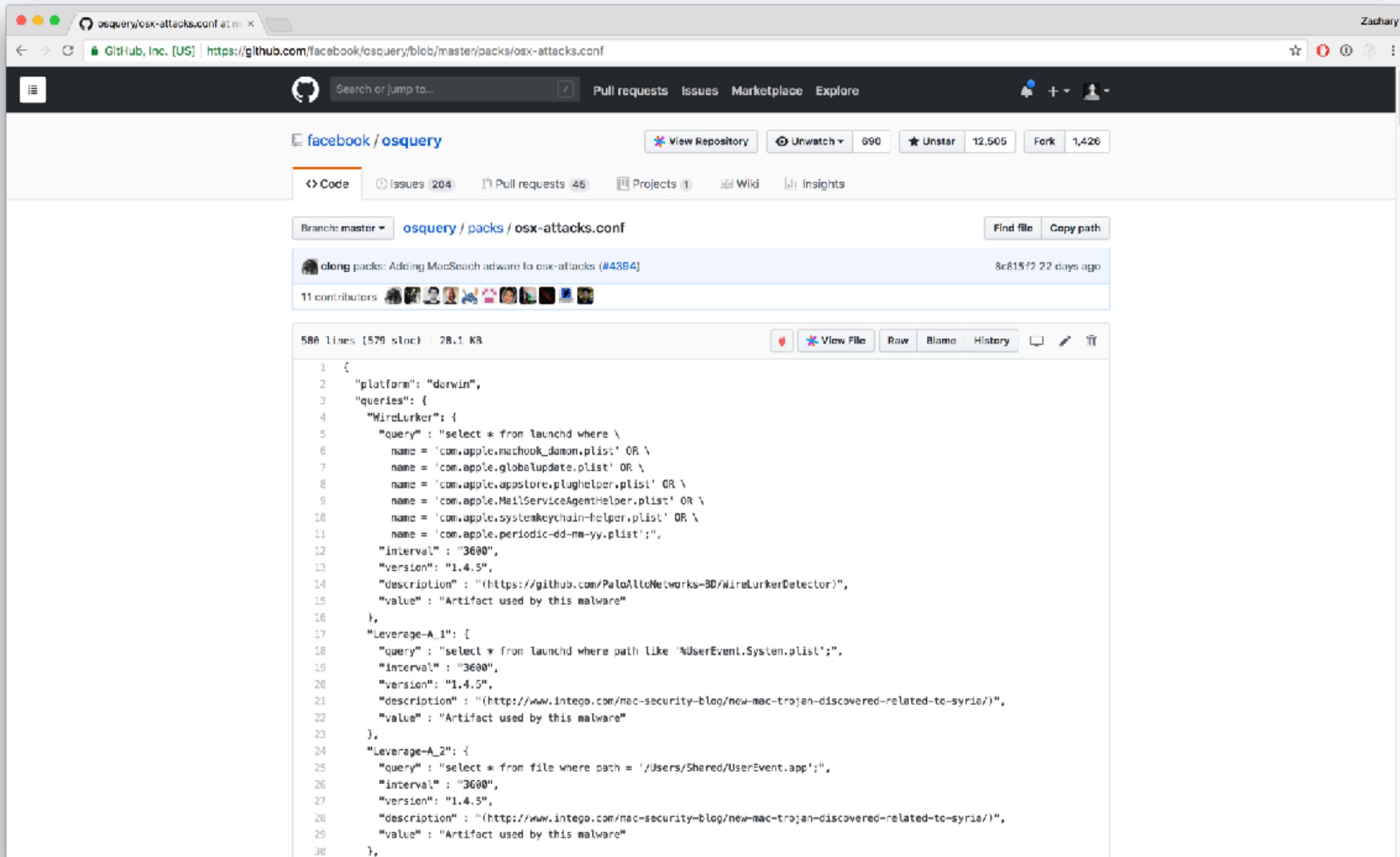
```
2. zwass@zwass: ~ (zsh)
~ ➤ osqueryd --config_path=osquery.conf --pidfile=osquery.pid --database_path=osquery/db --logger_path=osquery/logs
I0606 18:51:41.172075 2892845952 events.cpp:825] Event publisher not enabled: openbsm: Publisher disabled via configuration
I0606 18:51:41.172937 2892845952 events.cpp:825] Event publisher not enabled: scnetwork: Publisher not used
I0606 18:51:41.172960 2892845952 events.cpp:825] Event publisher not enabled: event_tapping: Publisher disabled via configuration
I0606 18:52:00.226517 112529408 scheduler.cpp:75] Executing scheduled query hardware_events: SELECT * FROM hardware_events
^C
✖ ➤ head -1 osquery/logs/osqueryd.results.log | jq
{
  "name": "hardware_events",
  "hostIdentifier": "zwass.local",
  "calendarTime": "Thu Jun  7 01:50:56 2018 UTC",
  "unixTime": "1528336256",
  "epoch": "0",
  "counter": "0",
  "columns": {
    "action": "attach",
    "driver": "IOUSBDeviceUserClientV2",
    "model": "USB3.0 Hub",
    "model_id": "1010",
    "path": "1:1",
    "revision": "",
    "serial": "0",
    "time": "1528336227",
    "type": "IOUSBDevice",
    "vendor": "Apple Inc.",
    "vendor_id": "05ac"
  },
  "action": "added"
}
```

What to do with all this power?



What to do with all this power?

Check out the community-sourced query packs



The screenshot shows a GitHub browser interface for the repository `facebook/osquery`. The specific file displayed is `osquery/packs/osx-attacks.conf`. The code listing contains several JSON objects representing OS queries. One notable object is for "WireLurker", which includes a query to select from launchd where certain plist names are present, an interval of 3600, and a version of 1.4.5. Another object is for "Leverage-A_1" and "Leverage-A_2", both of which include queries to select from launchd and file paths related to UserEvent.plist and UserEvent.app respectively, with intervals of 3600 and versions of 1.4.5.

```
1  {
2    "platform": "darwin",
3    "queries": {
4      "WireLurker": {
5        "query": "select * from launchd where \
6          name = 'com.apple.muchhook_damon.plist' OR \
7          name = 'com.apple.globalupdate.plist' OR \
8          name = 'com.apple.appstore.pluginhelper.plist' OR \
9          name = 'com.apple.MailServiceAgentHelper.plist' OR \
10         name = 'com.apple.systemkeychain-helper.plist' OR \
11         name = 'com.apple.periodic-dd-mm-yy.plist';",
12        "interval": "3600",
13        "version": "1.4.5",
14        "description": "(https://github.com/PaloAltoNetworks-8D/WireLurkerDetector)",
15        "value": "Artifact used by this malware"
16      },
17      "Leverage-A_1": {
18        "query": "select * from launchd where path like '%UserEvent.System.plist';",
19        "interval": "3600",
20        "version": "1.4.5",
21        "description": "(http://www.intego.com/mac-security-blog/new-mac-trojan-discovered-related-to-syria/)",
22        "value": "Artifact used by this malware"
23      },
24      "Leverage-A_2": {
25        "query": "select * from file where path = '/Users/Shared/UserEvent.app';",
26        "interval": "3600",
27        "version": "1.4.5",
28        "description": "(http://www.intego.com/mac-security-blog/new-mac-trojan-discovered-related-to-syria/)",
29        "value": "Artifact used by this malware"
30      }
31    }
32  }
```

What to do with all this power?

Build custom tables for your organization

```
package main

import (
    "context"
    "log"
    "os"

    "github.com/kolide/osquery-go"
    "github.com/kolide/osquery-go/plugin/table"
)

func main() {
    if len(os.Args) != 2 {
        log.Fatalf('Usage: %s SOCKET_PATH', os.Args[0])
    }

    server, err := osquery.NewExtensionManagerServer("foobar", os.Args[1])
    if err != nil {
        log.Fatalf("Error creating extension: %s\n", err)
    }

    server.RegisterPlugin(table.NewPlugin("foobar", FoobarColumns(), FoobarGenerate))
    if err := server.Run(); err != nil {
        log.Fatalln(err)
    }
}

func FoobarColumns() []table.ColumnDefinition {
    return []table.ColumnDefinition{
        table.TextColumn("foo"),
        table.TextColumn("baz"),
    }
}

func FoobarGenerate(ctx context.Context, queryContext table.QueryContext) ([]map[string]string, error) {
    return []map[string]string{
        {
            "foo": "bar",
            "baz": "baz",
        },
        {
            "foo": "bar",
            "baz": "baz",
        },
    }, nil
}

#!/usr/bin/env python

import osquery

@osquery.register_plugin
class MyTablePlugin(osquery.TablePlugin):
    def name(self):
        return "foobar"

    def columns(self):
        return [
            osquery.TableColumn(name="foo", type=osquery.STRING),
            osquery.TableColumn(name="baz", type=osquery.STRING),
        ]

    def generate(self, context):
        query_data = []

        for _ in range(2):
            row = {}
            row["foo"] = "bar"
            row["baz"] = "baz"
            query_data.append(row)

        return query_data

    if __name__ == "__main__":
        osquery.start_extension(name="my-awesome-extension", version="1.0.0")
```

What to do with all this power?

Implement a central management server

The screenshot shows the Kolide web application interface. On the left, there's a sidebar with icons for HOSTS, QUERY (selected), PACKS, CONFIG, and HELP. The main area has tabs for 'Manage Queries' and 'New Query'. The 'New Query' tab is active, displaying a code editor with the following SQL query:

```
1 SELECT
2     u.username,
3     g.groupname,
4     u.description,
5     u.shell
6 FROM
7     users u JOIN groups g
8 WHERE
9     u.gid = g.gid;
```

Below the query editor are sections for 'Select Targets' (listing four host IDs) and 'Query Title' (set to 'User Information'). There's also a 'Description' field containing the text 'Some brief information about the user account on a system.' At the bottom of this section are 'SAVE' and 'RUN QUERY' buttons.

On the right side, there's a sidebar titled 'Choose a Table' with 'USERS' selected. It shows 'Local system users' and 'OS Availability' (All Platforms). Below this is a 'Columns' section listing fields: uid (big int), gd (big int), uid_signed (bit int), gd_signed (big int), and username (text). A 'SHOW' link is at the bottom of this list.

At the bottom of the main content area, there's a table titled '4 of 4 Hosts Returning 84 Records' with columns for hostname, description, groupname, shell, and username. The data shows four hosts, each with root as the username and root as the groupname and shell. The table includes an 'EXPORT' button.

What to do with all this power?

Check out Kolide Launcher

Key Osquery Improvements



Always Up to Date

Take advantage of emerging osquery capabilities faster with Kolide Launcher's auto-updater. Using **The Update Framework (TUF)** you can securely update osquery with ease.



Easy Packaging & Deployment

Deploying osquery and configuring it to communicate with a management server can be complicated. The Launcher includes a tool to help create Launcher packages for your organization.



gRPC Remote API

The Launcher includes a set of gRPC plugins for remote communication with a gRPC server. An implementation of the gRPC server is included with the Kolide Fleet osquery fleet manager.

[Learn More →](#)

[Learn More →](#)

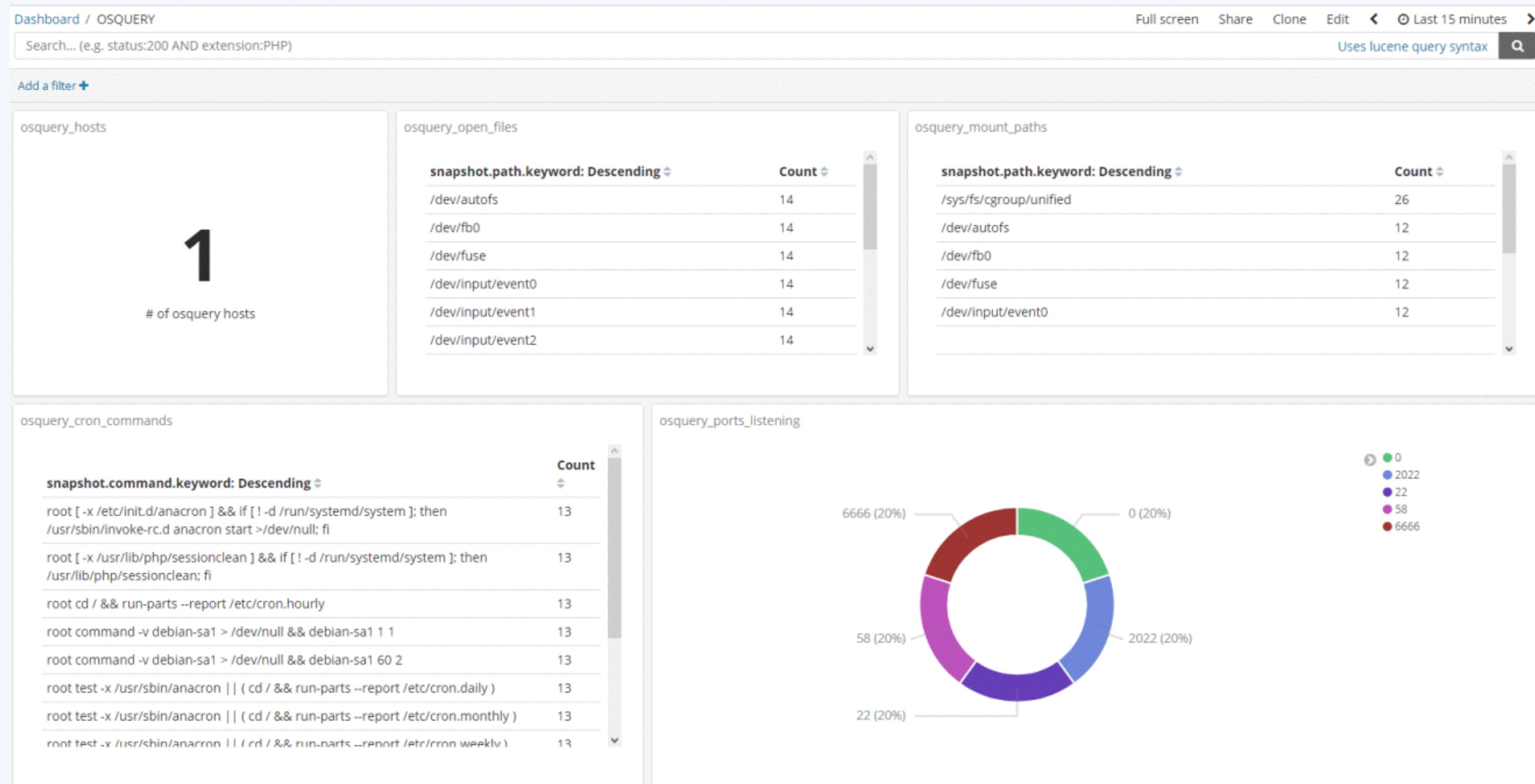
[Learn More →](#)

 KOLIDE

<https://kolide.com/launcher>

What to do with all this power?

Push logs to ELK stack for dashboards, alerting and archiving



What to do with all this power?

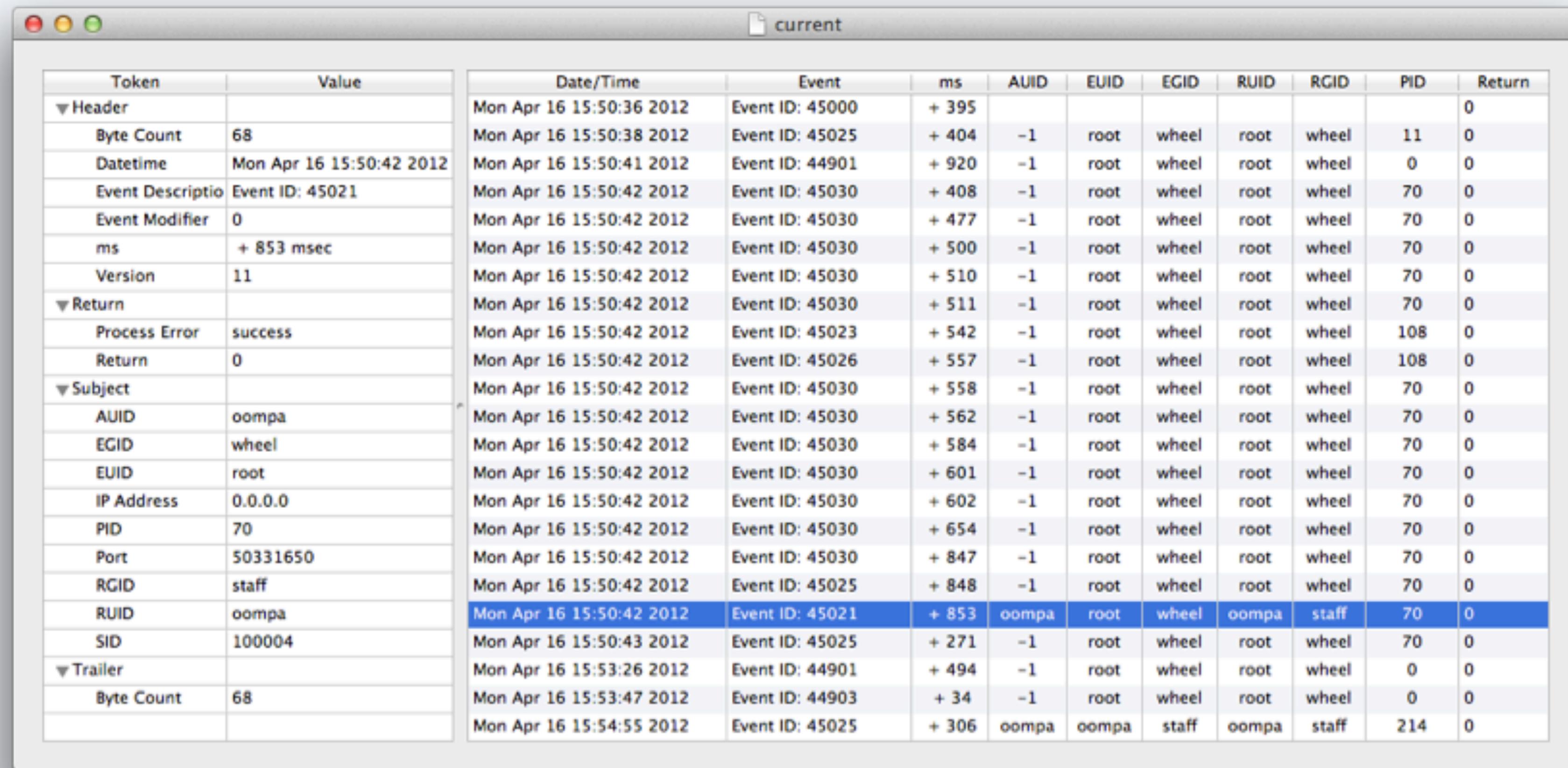
Conditionally install software using Munki



http://bit.ly/osquery_munki_groob

What to do with all this power?

Process/Socket Auditing, File Integrity Monitoring



The screenshot shows a Mac OS X application window titled "current". The window contains two tables: a configuration table on the left and a log table on the right.

Configuration Table (Left):

Token	Value
▼ Header	
Byte Count	68
Datetime	Mon Apr 16 15:50:42 2012
Event Description	Event ID: 45021
Event Modifier	0
ms	+ 853 msec
Version	11
▼ Return	
Process Error	success
Return	0
▼ Subject	
AUID	oompa
EGID	wheel
EUID	root
IP Address	0.0.0.0
PID	70
Port	50331650
RGID	staff
RUID	oompa
SID	100004
▼ Trailer	
Byte Count	68

Log Table (Right):

Date/Time	Event	ms	AUID	EUID	EGID	RUID	RGID	PID	Return
Mon Apr 16 15:50:36 2012	Event ID: 45000	+ 395	-1	root	wheel	root	wheel	11	0
Mon Apr 16 15:50:38 2012	Event ID: 45025	+ 404	-1	root	wheel	root	wheel	11	0
Mon Apr 16 15:50:41 2012	Event ID: 44901	+ 920	-1	root	wheel	root	wheel	0	0
Mon Apr 16 15:50:42 2012	Event ID: 45030	+ 408	-1	root	wheel	root	wheel	70	0
Mon Apr 16 15:50:42 2012	Event ID: 45030	+ 477	-1	root	wheel	root	wheel	70	0
Mon Apr 16 15:50:42 2012	Event ID: 45030	+ 500	-1	root	wheel	root	wheel	70	0
Mon Apr 16 15:50:42 2012	Event ID: 45030	+ 510	-1	root	wheel	root	wheel	70	0
Mon Apr 16 15:50:42 2012	Event ID: 45030	+ 511	-1	root	wheel	root	wheel	70	0
Mon Apr 16 15:50:42 2012	Event ID: 45023	+ 542	-1	root	wheel	root	wheel	108	0
Mon Apr 16 15:50:42 2012	Event ID: 45026	+ 557	-1	root	wheel	root	wheel	108	0
Mon Apr 16 15:50:42 2012	Event ID: 45030	+ 558	-1	root	wheel	root	wheel	70	0
Mon Apr 16 15:50:42 2012	Event ID: 45030	+ 562	-1	root	wheel	root	wheel	70	0
Mon Apr 16 15:50:42 2012	Event ID: 45030	+ 584	-1	root	wheel	root	wheel	70	0
Mon Apr 16 15:50:42 2012	Event ID: 45030	+ 601	-1	root	wheel	root	wheel	70	0
Mon Apr 16 15:50:42 2012	Event ID: 45030	+ 602	-1	root	wheel	root	wheel	70	0
Mon Apr 16 15:50:42 2012	Event ID: 45030	+ 654	-1	root	wheel	root	wheel	70	0
Mon Apr 16 15:50:42 2012	Event ID: 45030	+ 847	-1	root	wheel	root	wheel	70	0
Mon Apr 16 15:50:42 2012	Event ID: 45025	+ 848	-1	root	wheel	root	wheel	70	0
Mon Apr 16 15:50:42 2012	Event ID: 45021	+ 853	oompa	root	wheel	oompa	staff	70	0
Mon Apr 16 15:50:43 2012	Event ID: 45025	+ 271	-1	root	wheel	root	wheel	70	0
Mon Apr 16 15:53:26 2012	Event ID: 44901	+ 494	-1	root	wheel	root	wheel	0	0
Mon Apr 16 15:53:47 2012	Event ID: 44903	+ 34	-1	root	wheel	root	wheel	0	0
Mon Apr 16 15:54:55 2012	Event ID: 45025	+ 306	oompa	oompa	staff	oompa	staff	214	0

http://bit.ly/advanced_osquery_clong

What to do with all this power?

Get Inspired with ideas from QueryCon

The screenshot shows the QueryCon 2018 video library interface. At the top, there's a large thumbnail for a video titled "Nick Anderson (Facebook) - Evolving Our Open Source Community". Below it, a grid of smaller thumbnails represents other sessions:

- Teddy Reed (Facebook)**: The Scary Parts of Osquery
- Scott Lundgren (Carbon Black)**: An Outsider's Journey to Join the Osquery Community
- Mitchell Grenier (Facebook)**: Catching Everything with Osquery Events
- Milan Shah (UpTycs)**: Osquery and Docker Containers
- Mike Myers (Trail of Bits)**: Extensions Skunkworks: Unconventional Uses for Osquery
- Michael Lynn (Facebook)**: Fact Finding
- Lauren Pearl (Trail of Bits)**: Three Super Features That Could Transform Osquery
- Chris Long (Palantir)**: Practical System Auditing with Osquery
- Ben Hughes (Stripe)**: Osquery, He Knows Me
- Mike Arpaia (Kolide)**: fleetctl - Instrumenting Dynamic Environments

Nick Anderson (Facebook) - Evolving Our Open Source Community

An exploration of how the osquery community has grown, problems we faced/still face, and the directions we're hoping to push the community forward. Over the past two years the osquery community has increased dramatically. Not only have the number of contributions to the agent dramatically bumped, so has the number of platforms we support and richness of discussions in our community around best leveraging osquery. Nick takes us on a tour of where osquery has been and where it is headed.

<https://querycon.io/#videos>

What to do with all this power?

Kolide Cloud

Online Devices (right now)

12



Platforms

- macOS (15)
- Linux (0)
- Windows (0)

Compliance

- Compliant (6)
- Non-Compliant (9)

On Corp

- On Corp (2)
- Off Corp (10)

Munki Versions

VERSION	LAST 90 DAYS	NOW	HIGH	LOW
3.11.3447		14	14	1

macOS Builds

VERSION	LAST 90 DAYS	NOW	HIGH	LOW
17F77		10	13	10
17G65		3	3	3
17E202		2	4	2

Open Alerts

Ngrok Tunnels
8
ON 2 DEVICES

Plain-text 1Password Recovery Kits
6
ON 5 DEVICES

Unencrypted SSH Keys
6
ON 6 DEVICES

Unencrypted Time Machine Backups
2
ON 1 DEVICE

Plain-text GitHub 2FA Codes
5
ON 1 DEVICE

Daily Device Totals

Enrolled Online Has Alert

16 13 12

↑100% ↑100% ↑100%

Plain-text 1Password Recovery Kits
6
ON 5 DEVICES

Unencrypted SSH Keys
6
ON 6 DEVICES

Unencrypted Time Machine Backups
2
ON 1 DEVICE

Plain-text GitHub 2FA Codes
5
ON 1 DEVICE

no change last 28 days

no change last 28 days

no change last 28 days

↓74% last 28 days

Recently Enrolled Devices

scottsdale.local



MacBook Pro
(15-inch, 2017)
C02WB598HTD5
macOS 10.13.5
enrolled: 17 days ago

dover.local



MacBook Pro
(15-inch, 2017)
C02WB66LGHTD5
macOS 10.13.4
enrolled: 17 days ago

ccoy



MacBook Pro
(13-inch, 2017, Two Thunderbolt 3 ports)
C02W316YHV29
macOS 10.13.5
enrolled: 21 days ago

- ✓ Screen Sharing Disabled
- ✓ Internet Sharing Disabled
- ✓ Gatekeeper Enabled
- ✓ FileVault Enabled
- ✓ File Sharing Disabled
- ✓ SIP Configured
- ✗ System Firewall Enabled
- ✓ System Firewall Enabled
- ✓ Bluetooth Sharing Disabled
- ✓ Remote Management Disabled
- ✓ Remote Login Disabled

- ✓ Screen Sharing Disabled
- ✓ Internet Sharing Disabled
- ✓ Gatekeeper Enabled
- ✓ FileVault Enabled
- ✓ File Sharing Disabled
- ✓ SIP Configured
- ✓ System Firewall Enabled
- ✓ System Firewall Enabled
- ✓ Bluetooth Sharing Disabled
- ✓ Remote Management Disabled
- ✓ Remote Login Disabled

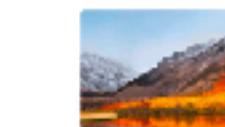
- ✓ Screen Sharing Disabled
- ✓ Internet Sharing Disabled
- ✓ Gatekeeper Enabled
- ✓ FileVault Enabled
- ✓ File Sharing Disabled
- ✓ SIP Configured
- ✓ System Firewall Enabled
- ✓ System Firewall Enabled
- ✓ Bluetooth Sharing Disabled
- ✓ Remote Management Disabled
- ✓ Remote Login Disabled

macOS Version Statistics



1

Mojave



16

High Sierra



Sierra



El Capitan



Yosemite & Older

10.14

10.13.5 (beta)

10.12.6

10.11.6

10.13.5

10.12.5

10.11.5

8

10.12.4

10.11.4

10.13.3

10.12.3

10.11.3

10.13.2

10.12.2

10.11.2

10.13.1

10.12.1

10.11.1

<https://kolide.com>

Two paths with osquery

Configuration management,
Log aggregation,
Alerting,
etc.

Direct to insights
with commercial tools
(Kolide Cloud)

Build it Yourself
with OSS tools
(Kolide Fleet, Zentral)

Two paths with osquery @ PSUMacAdmins

- The commercial straight-to-insights approach
- “How Software Engineers Ruin Macs and Could Ruin You” - Jason Meller (Kolide)
- Thursday July 12, 9:00-10:15am
- The DIY open-source approach with Kolide’s OSS tools
- “Using osquery via Fleet for client/server visibility” - Lucas Hall (Saturna Capital)
- Thursday July 12, 3:15-4:30pm

Join us in osquery Slack

bit.ly/osquery_slack

StackOverflow: #osquery

Thank you!

 **zach** @ kolide.com

 [github.com / zwass](https://github.com/zwass)

 **zwass** @ osquery Slack

 [twitter.com / thezachw](https://twitter.com/thezachw)