OVERSIGHT:

Building An Asset Inventory Data Pipeline

PHIL MEYERSON, SOFTWARE ENGINEER

EMBEDDED FLIGHT SYSTEMS, INC.
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Agenda

- Oversight Technical Add-On
  - Purpose
  - Asset Inventory Data Pipeline
  - Add-On Feature Review
  - Demo
- Wrap-Up, Acknowledgements, References

Disclaimer

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Goddard Space Flight Center.

Splunk is a registered trademark of Splunk, Inc.

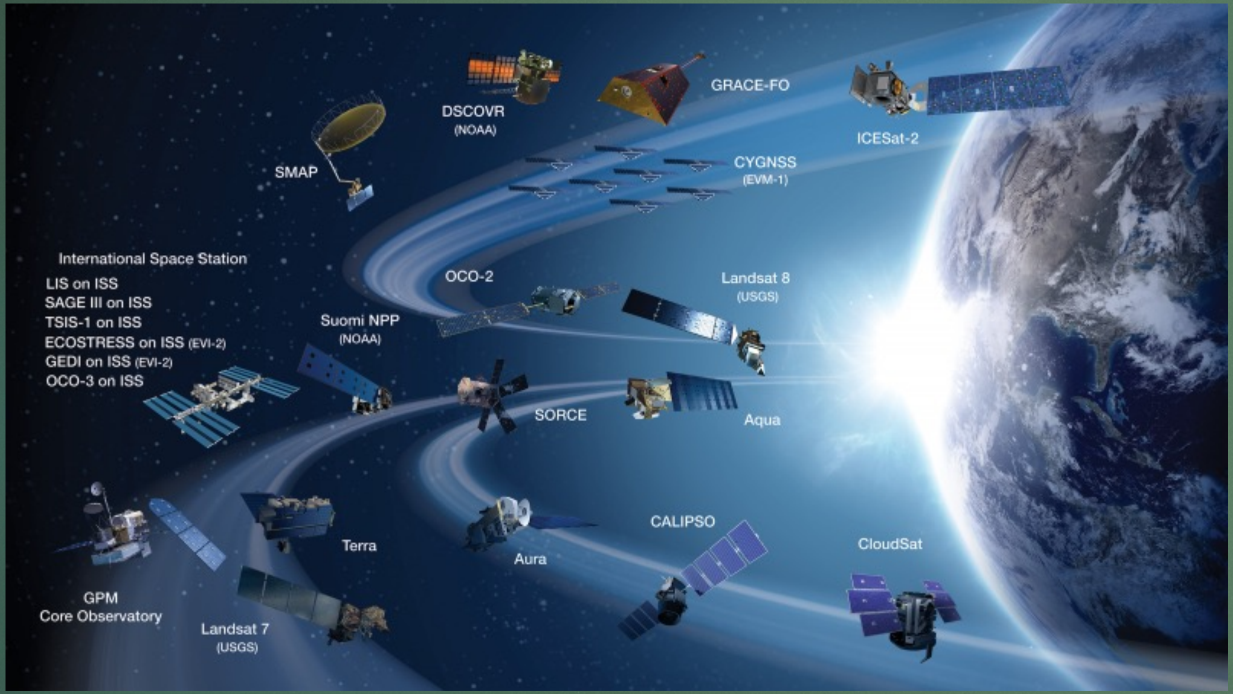
Phil Meyerson

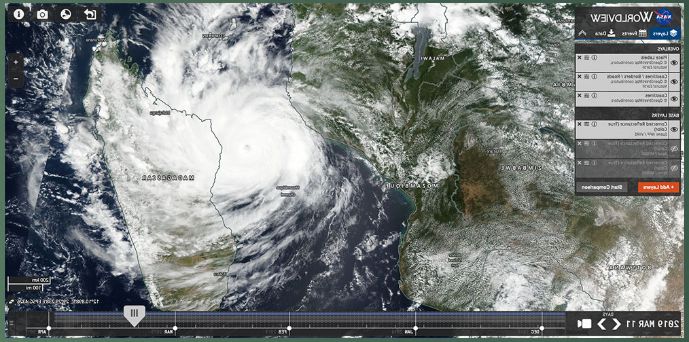
- Software Engineer for Embedded Flight Systems, Inc.
- Supporting NASA Earth Science Data and Information System (ESDIS) with custom SIEM apps and analytic products
- 10+ years in enterprise IT; last as security analyst
- Splunk Certified Enterprise Admin

- Music: jazz/rock/ska/folk
- Mountains and Museums > beach
- DC Cherry Blossom Festival!

- Github.com/pmeyerson
- @pmeyerson
- https://linkedin.com/in/philmeyerson
- Baltimore Splunk User Group (pre-pandemic ☹)

Earth Science Projects Division > EOSDIS > ESDIS





https://espd.gsfc.nasa.gov/projects.html
https://earthdata.nasa.gov

Oversight Technical Add-On

Goals

- How many assets do we have on our network?
- How long have these assets been present?
- Are we receiving all the sourcetypes we expect from a given asset?

Method

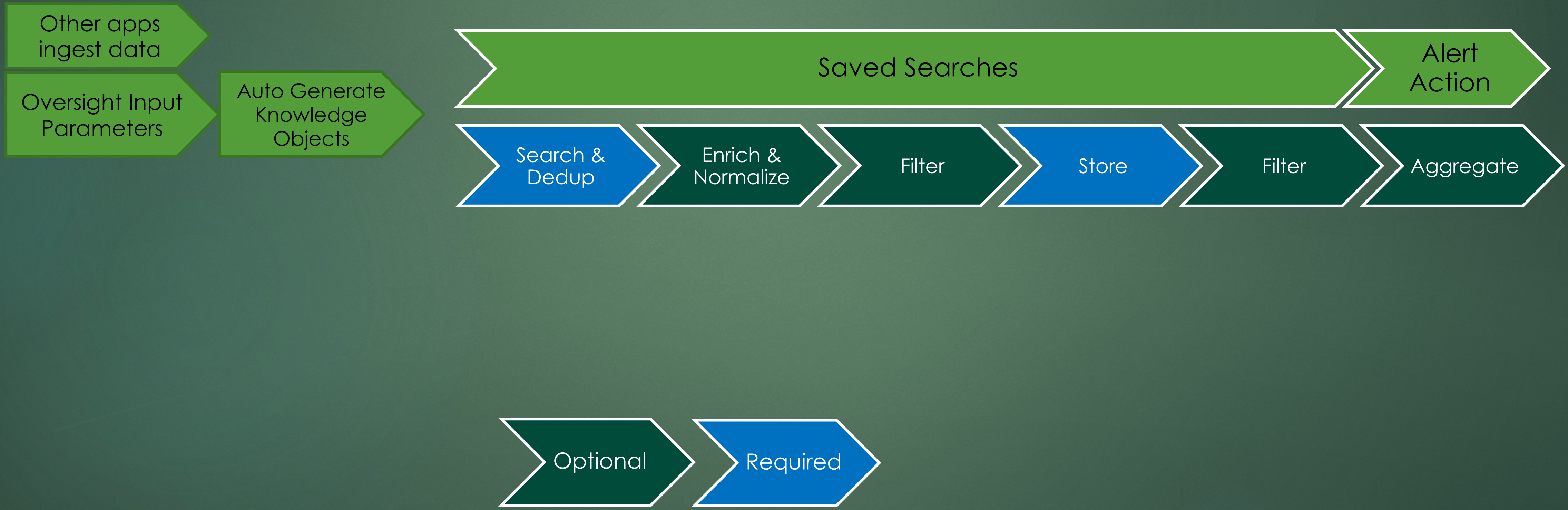
- Setup a saved search for each sourcetype to be monitored

- Search executes on schedule and updates lookup tables
  - One detailed lookup for that sourcetype
  - A summary lookup listing assets seen across all sourcetpes

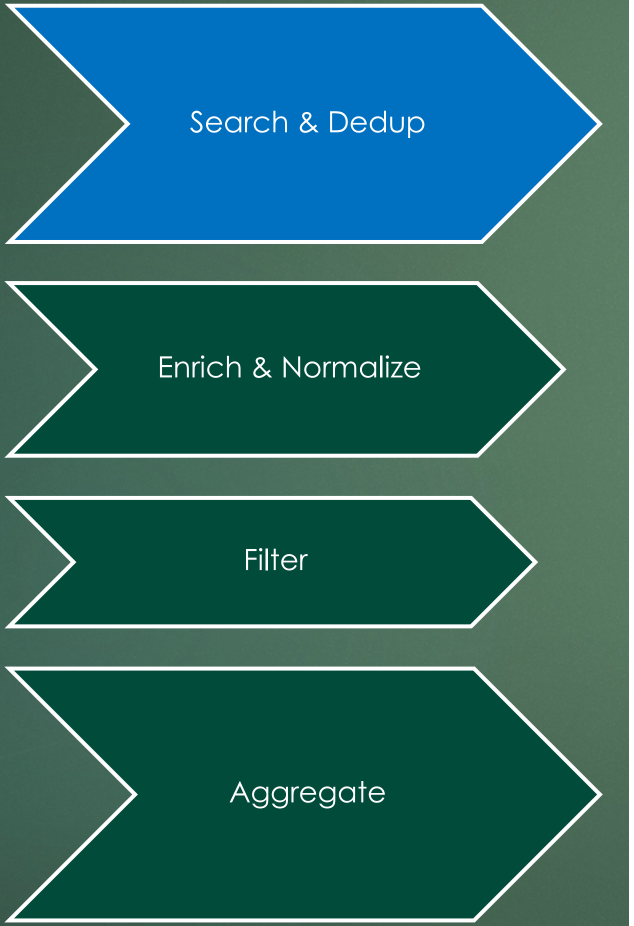
- Provide flexible system using automation to add/modify search jobs as needed

Asset Inventory Data Pipeline

Sarah Dietrich, EFSI



Oversight Input Parameters

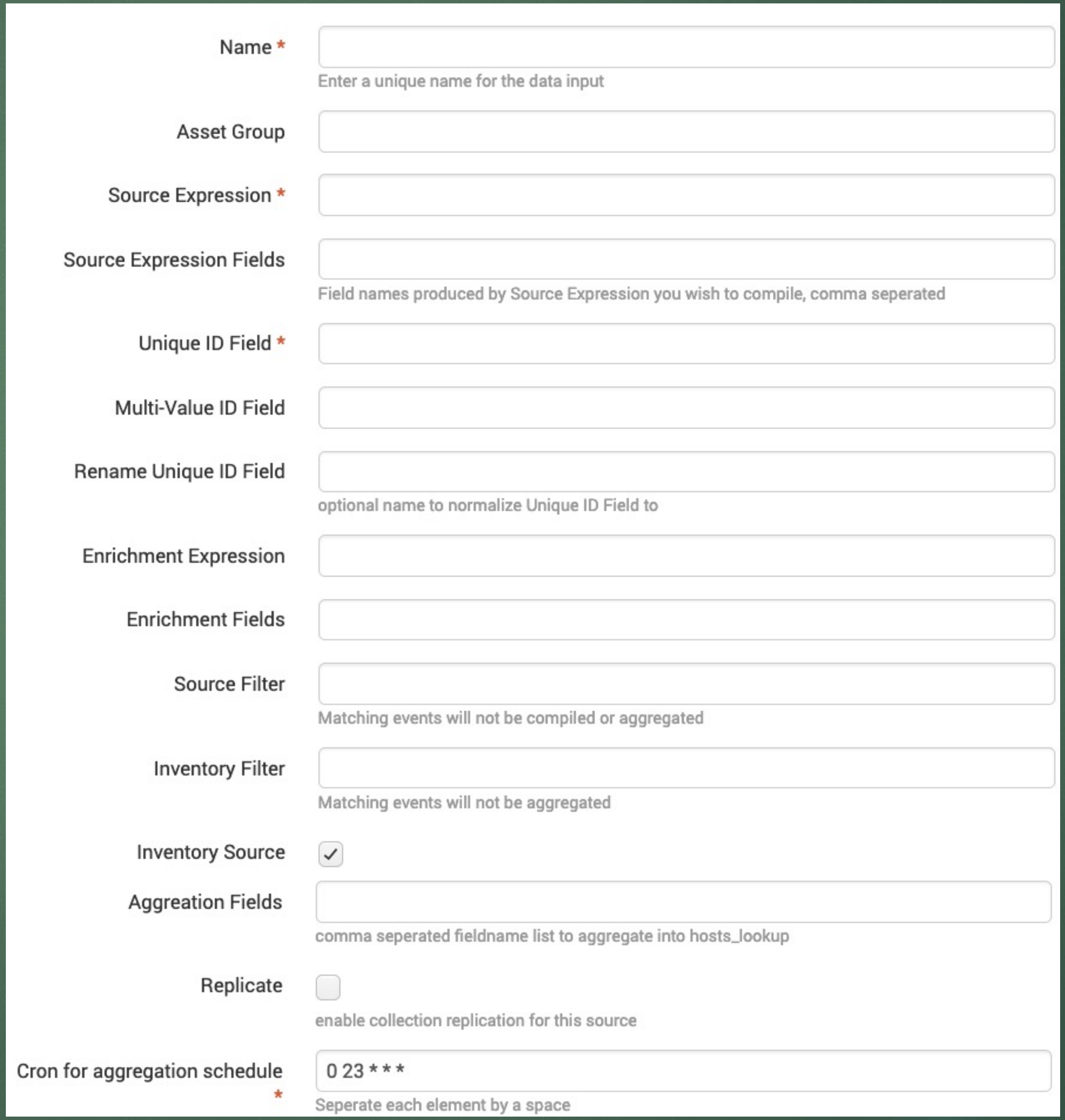


Search & Dedup

Enrich & Normalize

Filter

Aggregate



Name *

Enter a unique name for the data input

Asset Group

Source Expression *

Source Expression Fields

Field names produced by Source Expression you wish to compile, comma seperated

Unique ID Field *

Multi-Value ID Field

Rename Unique ID Field

optional name to normalize Unique ID Field to

Enrichment Expression

Enrichment Fields

Source Filter

Matching events will not be compiled or aggregated

Inventory Filter

Matching events will not be aggregated

Inventory Source ☑

Aggreation Fields

comma seperated fieldname list to aggregate into hosts_lookup

Replicate □

enable collection replication for this source

Cron for aggregation schedule *: 0 23 * * *

Seperate each element by a space

Feature Overview

- Generate Knowledge Objects and Lookups
- Collate events from disparate sources which share common identifiers (IP Addresses, etc.)
- Track when each type of event was last observed, per asset
- Expire assets by a user-defined age limit or search expression
- Provide normalized asset inventory datasets

Data Source Requirements

- Each data source needs a unique identifier:

✅

```
1. Mar 01 03:22:01 status=CLIENT_CONNECT server=antivirus_server_1
   ClientID=10.2.1.1 d

2. Mar 01 08:22:01 sourcetype=login username=jdoe host=10.2.1.1
```

- Unique identifier consistent across all data sources for an asset:

🚫

```
1. Mar 01 03:22:01 sourcetype=antivirus status=CLIENT_CONNECT
    server=antivirus_server_1 ClientID=10.2.1.1

2. Mar 01 08:22:01 sourcetype=login username=jdoe
    host=system1.test.co
```

DEMOS

Wrap Up

Lessons Learned

- Adopt naming conventions, at least for inputs in the same `asset_group`
- Macros are awesome for enrichment and data normalization
- Set your metdata.conf so your users can share dashboard content within the app

Acknowledgements

- Sarah Dietrich, EFSI,
  Cloud Network & Security Architect for NASA Earth Observing Systems,
  Oversight original software author and 1.0 developer
- Tim Meader, CACI, the best Splunk Administrator
- Don Anderson, KBRWyle, Technical Lead, EOS Networks
- ESDIS Project and Customers for support of this work
- Baltimore Splunk User Group, answers.splunk.com, splunk-usergroups.slack.com
- "Measuring Data Quality for Ongoing Improvement: A Data Quality Assessment Framework", Laura Sebastian-Coleman, Morgan Kaufmann Publishers, 2012.
- Docs.splunk.com, answers.splunk.com, splunk-usersgroups.slack.com, github.com/splunk

Definitions[3]

Splunk Knowledge Objects: A user-defined entity that enriches the existing data in Splunk Enterprise. Examples include macros, lookups, and saved searches.

Splunk Lookup: A knowledge object that provides data enrichment by mapping a select value in an event to a field in another data source, and appending the matched results to the original event.

Splunk (KV Store) Collection: The container for a set of data in an App Key Value Store, similar to a database table where each record has a unique key.

Splunk Saved Search: A search that a user makes available for later use. Saved searches are knowledge objects.

References

1. https://www.axonius.com/blog/the-toyota-camry-of-cybersecurity-axonius-wins-rsac-2019-innovation-sandbox-to-solve-the-asset-management-challenge

2. “Building an Asset Inventory Framework with Splunk”.pdf, Phil Meyerson, https://github.com/pmeyerson/presentations

3. ”Splexicon: the Splunk glossary”, https://docs.splunk.com/Splexicon

Thank you!

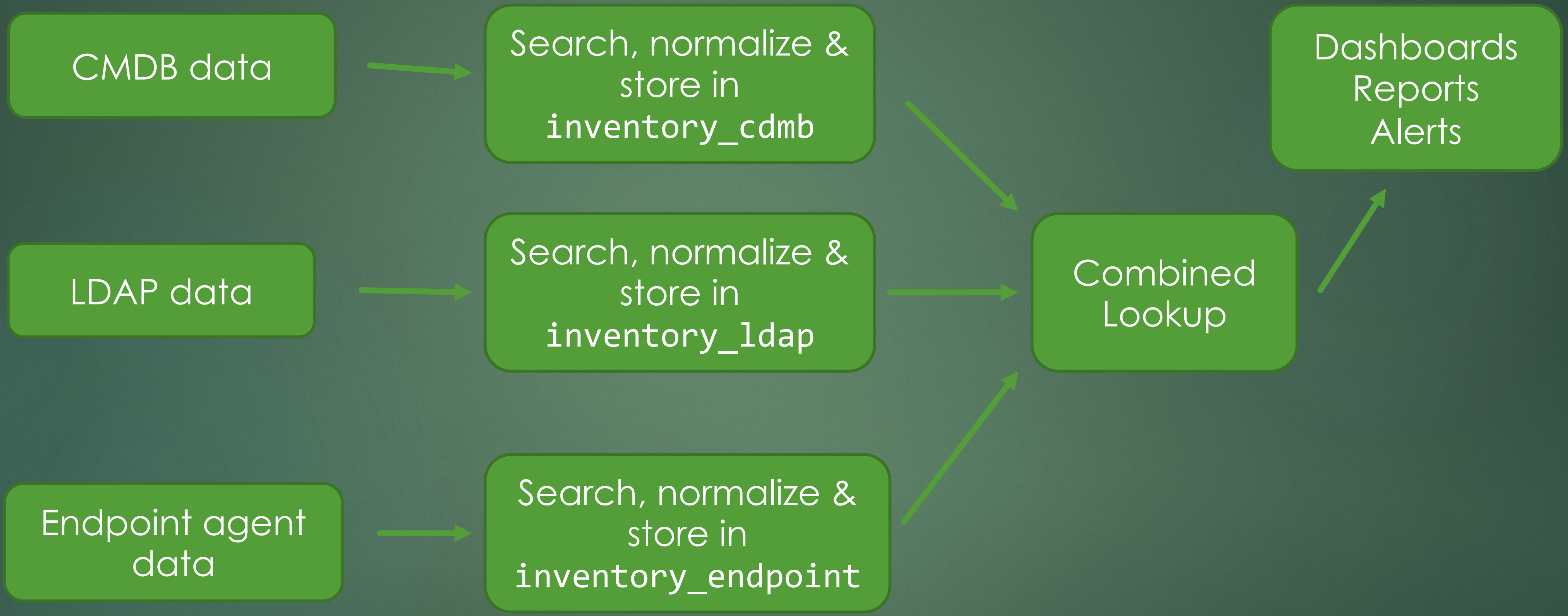
Additional Content

INSPIRATION

Challenge Accepted

"…NEVER KNOWING how many servers there are, virtual machines, endpoint devices, …. it's one of these fundamental problems in security that for some reason is really obvious, and many of us have lived with this pain, but nobody's really solved yet."[1]

Precursor – Prescriptive and Manual


CMDB data
LDAP data
Endpoint agent data
Search, normalize & store in inventory_cdmb
Search, normalize & store in inventory_ldap
Search, normalize & store in inventory_endpoint
Combined Lookup
Dashboards
Reports
Alerts

"Building an Asset Inventory Framework with Splunk Enterprise"[2]

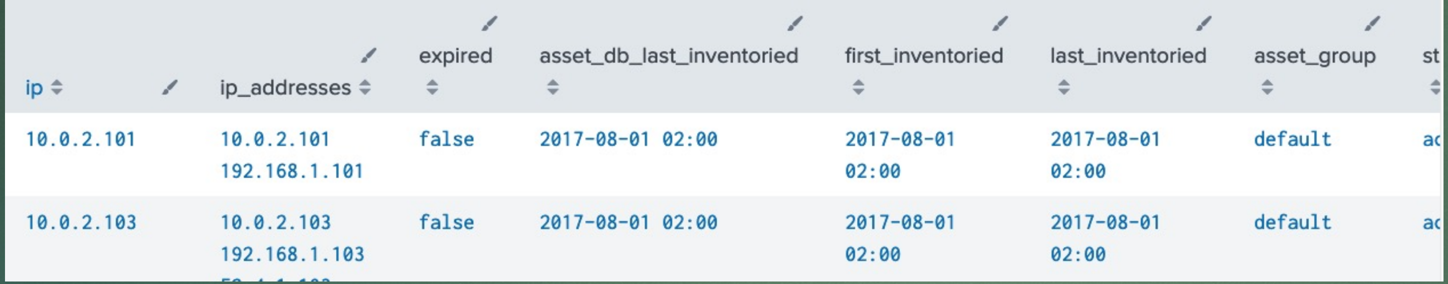
Generated Knowledge Objects

```
`asset_db_source`
|`eval_last_inventoried`
| `asset_db_enrichment_expression`
| `dedup(ip)`
| `set_key(ip)
| `set_not_expired`
| table `asset_db_fields`
| `outputlookup(asset_db_lookup)`
| `asset_db_inventory_filter`
| sendalert update_inventory
param.source_name=asset_db
```

```
collections.conf:
…

transforms.conf:
…

macros.conf:
…

savedsearches.con:
…
```

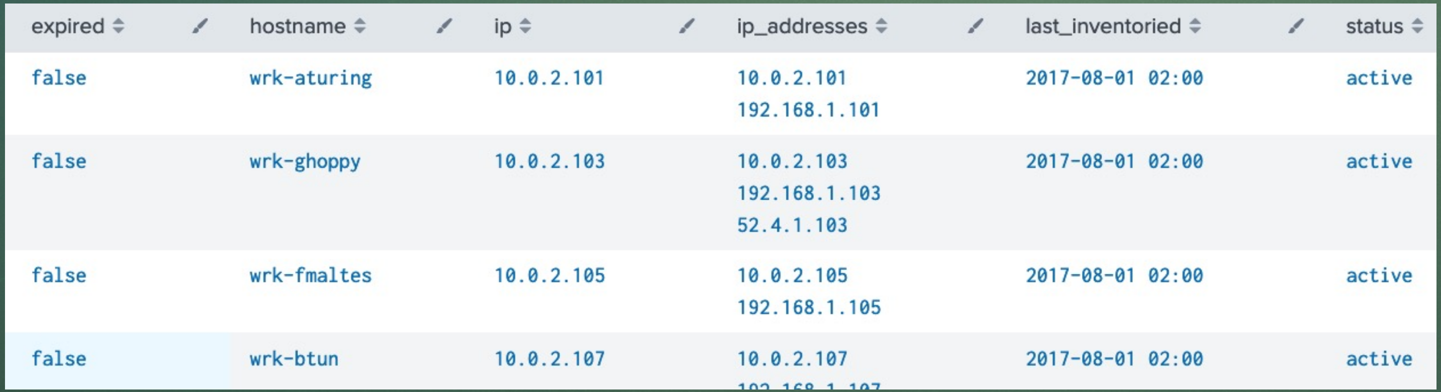
Example Lookup Content

hosts_lookup



ip	ip_addresses	expired	asset_db_last_inventoried	first_inventoried	last_inventoried	asset_group	st
10.0.2.101	10.0.2.101 192.168.1.101	false	2017-08-01 02:00	2017-08-01 02:00	2017-08-01 02:00	default	ac
10.0.2.103	10.0.2.103 192.168.1.103	false	2017-08-01 02:00	2017-08-01 02:00	2017-08-01 02:00	default	ac

asset_db_lookup



expired	hostname	ip	ip_addresses	last_inventoried	status
false	wrk-aturing	10.0.2.101	10.0.2.101 192.168.1.101	2017-08-01 02:00	active
false	wrk-ghoppy	10.0.2.103	10.0.2.103 192.168.1.103 52.4.1.103	2017-08-01 02:00	active
false	wrk-fmaltes	10.0.2.105	10.0.2.105 192.168.1.105	2017-08-01 02:00	active
false	wrk-btun	10.0.2.107	10.0.2.107	2017-08-01 02:00	active

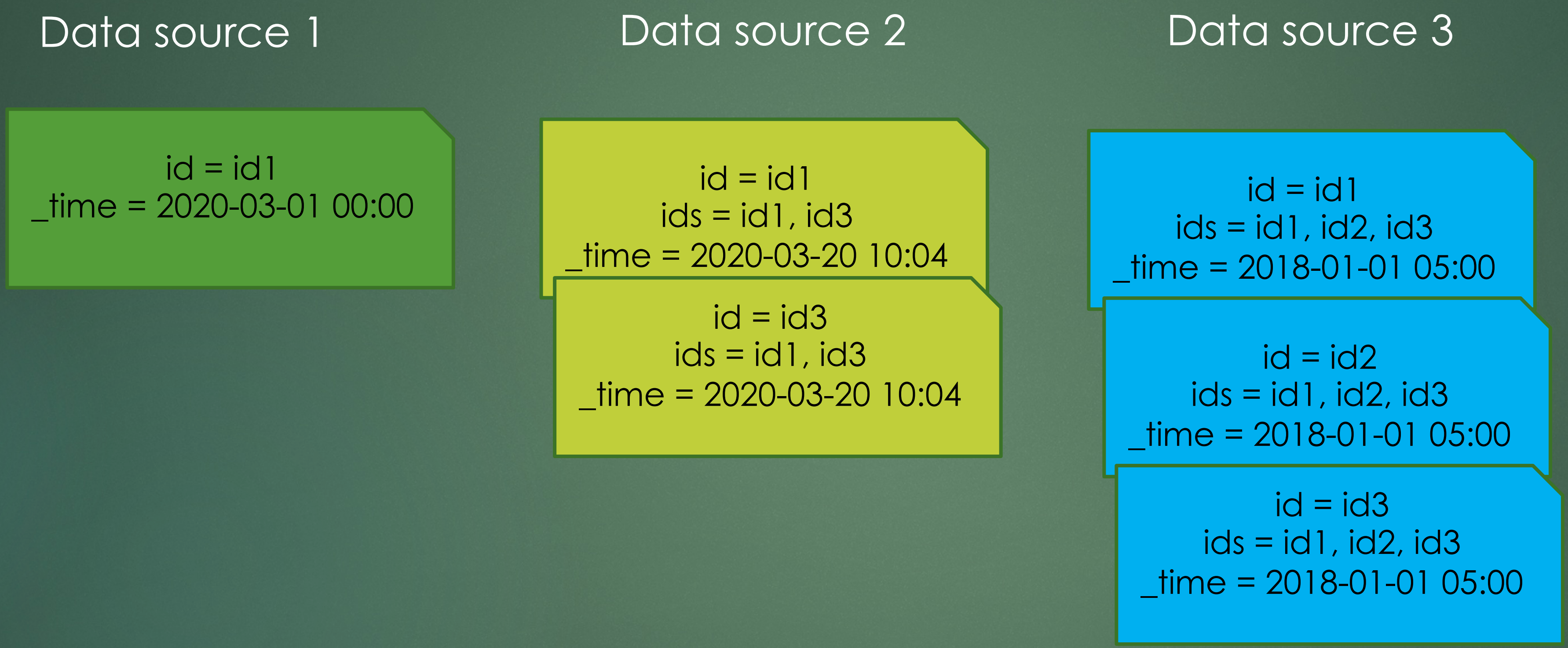
… source_N_lookup

Data Model Notes

- `asset_group`
  - Logical grouping of inputs
  - Pivot point for dashboards and gap analysis.
  - Each data source maps to one asset group
- `expired`
  - Initially “`false`”
  - Alert Action sets value to timestamp on expiration
  - Used as filter in lookup definition
- `Multi-Value ID Field`
  - Provides common field to `dedup` unique asset records
  - Multi-Value field
  - i.e.: ip_addresses
- Customizable via App Configuration

Multi-ID Collation

Multi-ID Collation



Data source 1

id = id1
_time = 2020-03-01 00:00

Data source 2

id = id1
ids = id1, id3
_time = 2020-03-20 10:04

id = id3
ids = id1, id3
_time = 2020-03-20 10:04

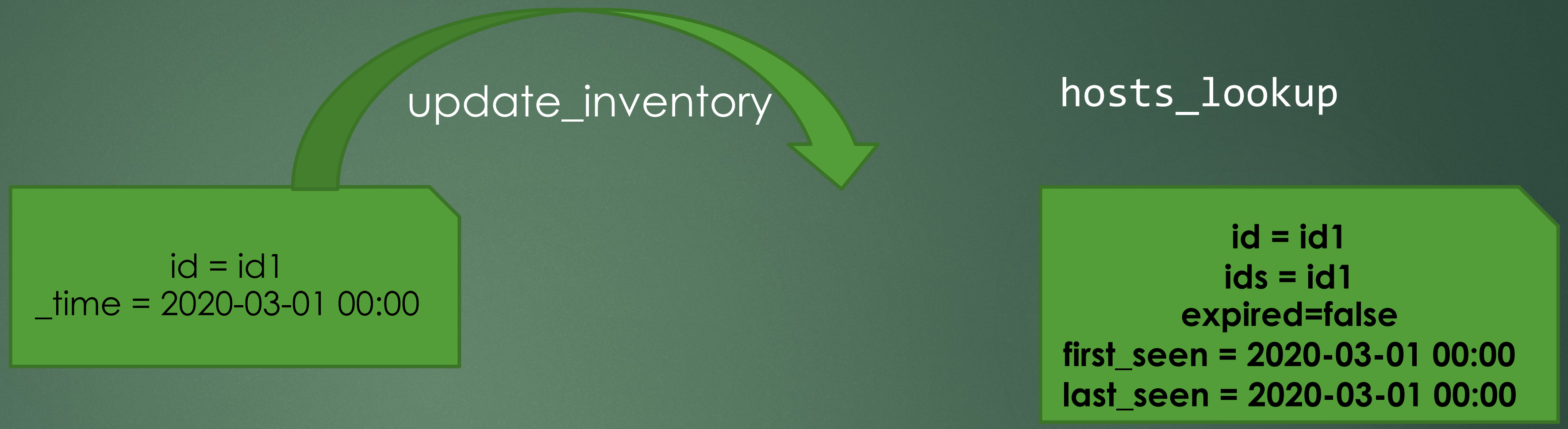
Data source 3

id = id1
ids = id1, id2, id3
_time = 2018-01-01 05:00

id = id2
ids = id1, id2, id3
_time = 2018-01-01 05:00

id = id3
ids = id1, id2, id3
_time = 2018-01-01 05:00

Multi-ID Collation



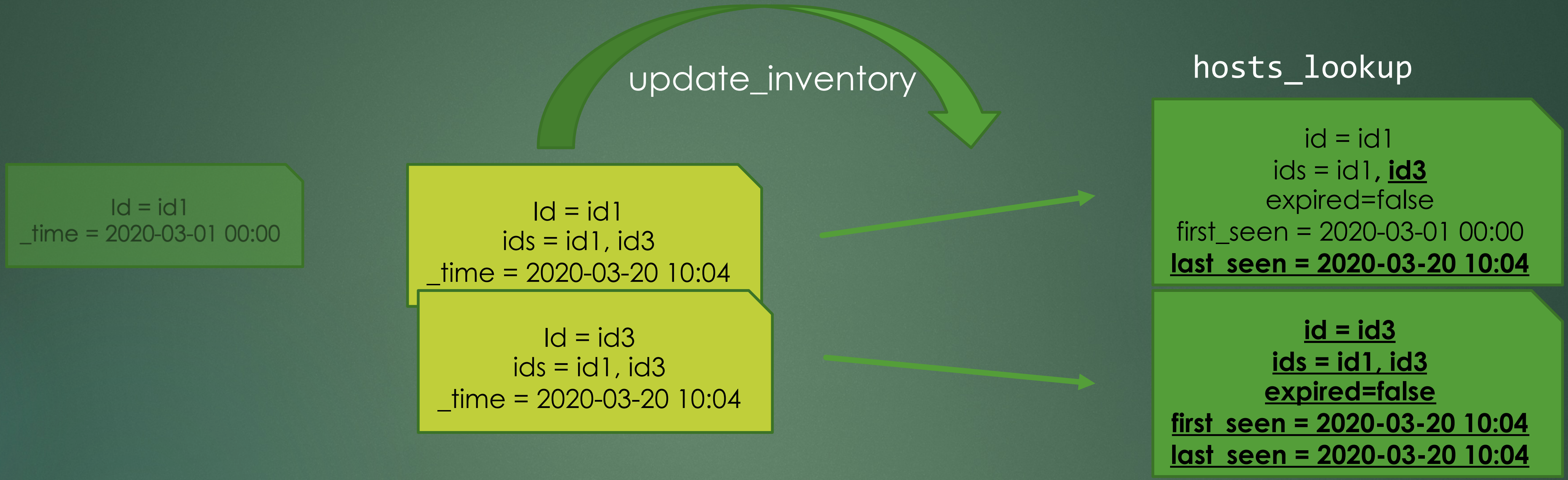
update_inventory

id = id1
_time = 2020-03-01 00:00

hosts_lookup

id = id1
ids = id1
expired=false
first_seen = 2020-03-01 00:00
last_seen = 2020-03-01 00:00

Multi-ID Collation



Id = id1
_time = 2020-03-01 00:00

update_inventory

Id = id1
ids = id1, id3
_time = 2020-03-20 10:04

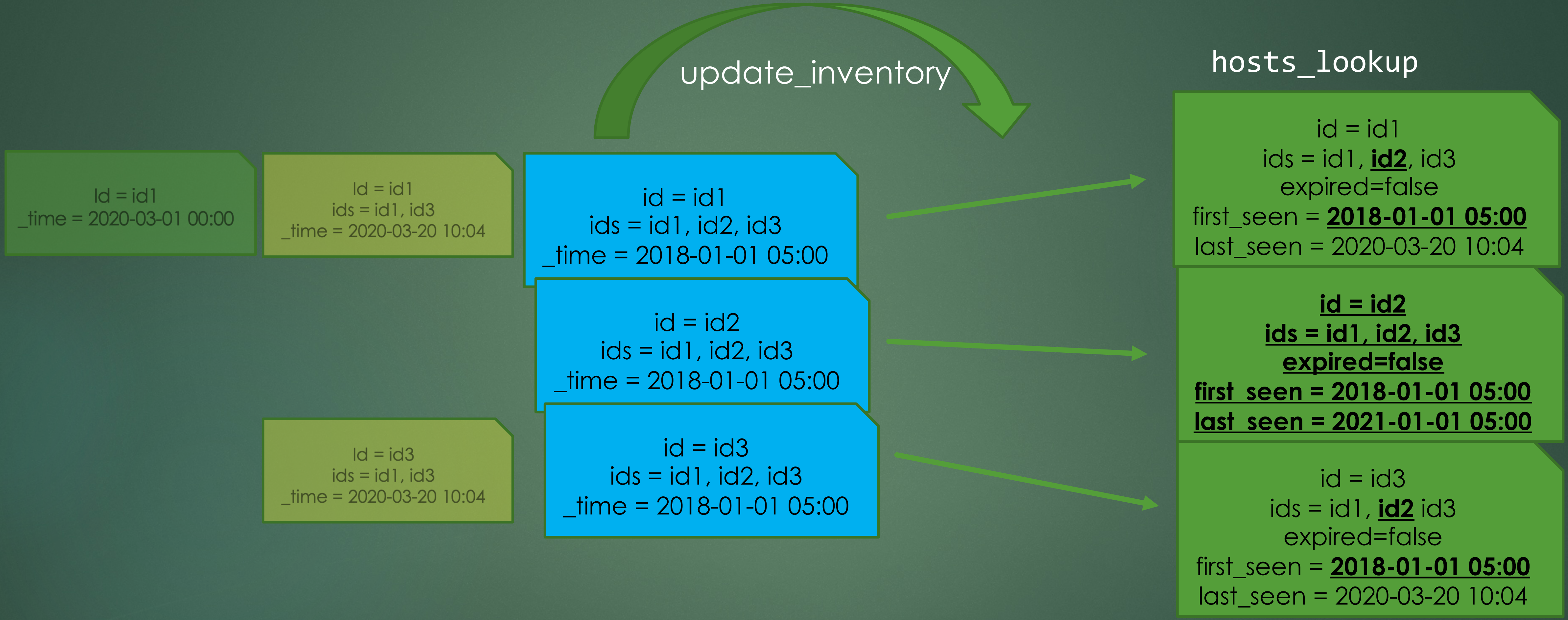
Id = id3
ids = id1, id3
_time = 2020-03-20 10:04

hosts_lookup

id = id1
ids = id1, **id3**
expired=false
first_seen = 2020-03-01 00:00
last_seen = 2020-03-20 10:04

id = id3
ids = id1, id3
expired=false
first_seen = 2020-03-20 10:04
last_seen = 2020-03-20 10:04

Multi-ID Collation



update_inventory

hosts_lookup

Id = id1
_time = 2020-03-01 00:00

Id = id1
ids = id1, id3
_time = 2020-03-20 10:04

Id = id3
ids = id1, id3
_time = 2020-03-20 10:04

id = id1
ids = id1, id2, id3
_time = 2018-01-01 05:00

id = id2
ids = id1, id2, id3
_time = 2018-01-01 05:00

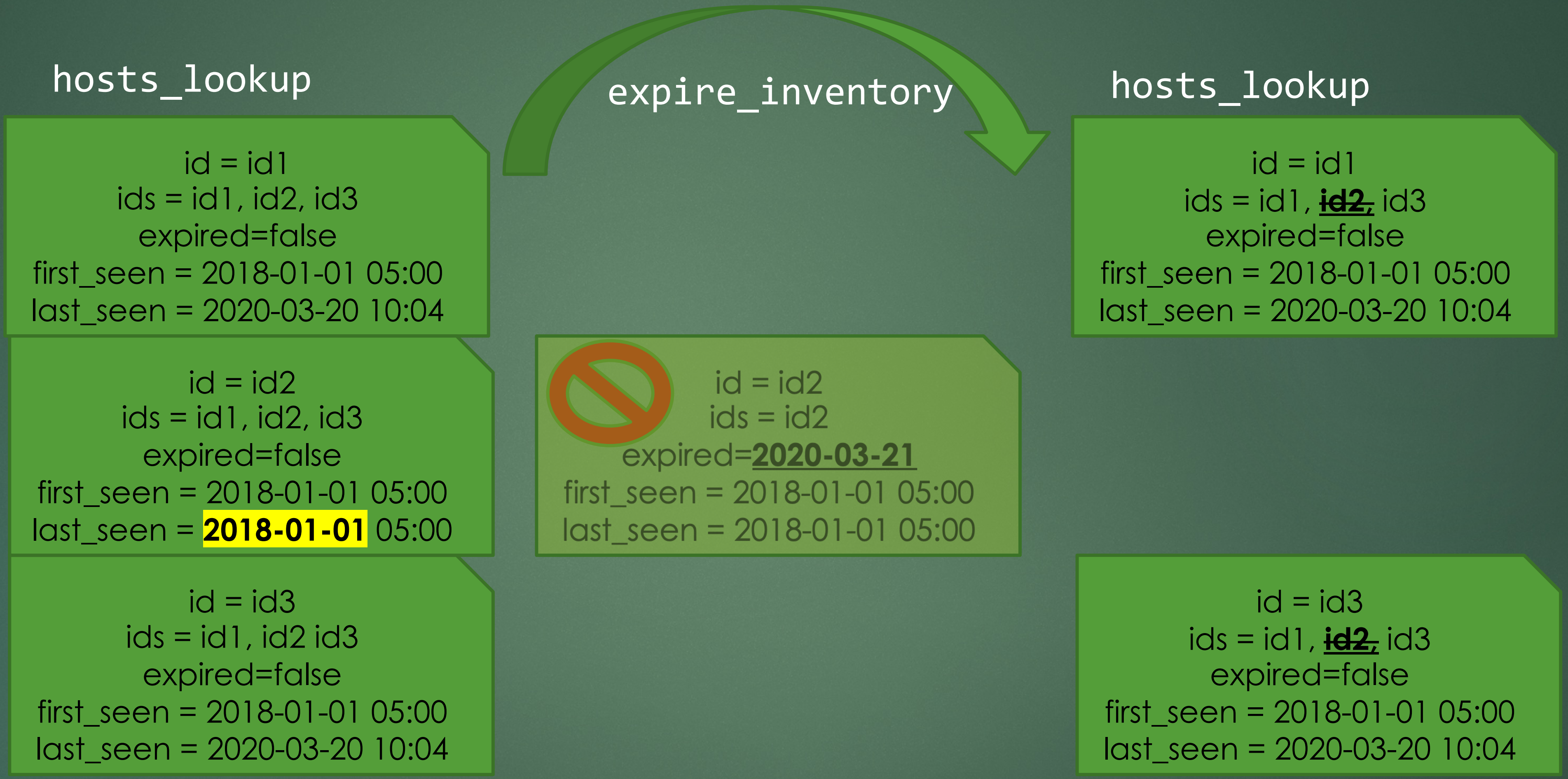
id = id3
ids = id1, id2, id3
_time = 2018-01-01 05:00

id = id1
ids = id1, **id2**, id3
expired=false
first_seen = **2018-01-01 05:00**
last_seen = 2020-03-20 10:04

id = id2
ids = id1, id2, id3
expired=false
first_seen = 2018-01-01 05:00
last_seen = 2021-01-01 05:00

id = id3
ids = id1, **id2** id3
expired=false
first_seen = **2018-01-01 05:00**
last_seen = 2020-03-20 10:04

Multi-ID Collation



hosts_lookup

id = id1
ids = id1, id2, id3
expired=false
first_seen = 2018-01-01 05:00
last_seen = 2020-03-20 10:04

id = id2
ids = id1, id2, id3
expired=false
first_seen = 2018-01-01 05:00
last_seen = **2018-01-01** 05:00

id = id3
ids = id1, id2 id3
expired=false
first_seen = 2018-01-01 05:00
last_seen = 2020-03-20 10:04

expire_inventory

id = id2
ids = id2
expired=**2020-03-21**
first_seen = 2018-01-01 05:00
last_seen = 2018-01-01 05:00

hosts_lookup

id = id1
ids = id1, ~~**id2,**~~ id3
expired=false
first_seen = 2018-01-01 05:00
last_seen = 2020-03-20 10:04

id = id3
ids = id1, ~~**id2,**~~ id3
expired=false
first_seen = 2018-01-01 05:00
last_seen = 2020-03-20 10:04

Oversight Evaluation

+

Easy to configure improvements to data model

More control over record aggregation

Macro names clarify intent of query expressions

−

Complexity moves to custom code

Oversight of Oversight

- Daily ingestion rates per source:

```
index=_internal sourcetype=scheduler savedsearch_name=*hosts app=TA-oversight
| stats count result_count by savedsearch_name
```

- Data ingested today:

```
| inputlookup hosts_lookup
| search mvindex(split(last_seen, “ “),0) = strftime(now(), “%Y-%m-%d”)
| stats count by asset_group
```

- Data Freshness per data source
  - When was the telemetry last ingested (raw event `_time`)
  - When was the telemetry last processed (source lookups `last_seen`)
- Evaluate per-field data quality by values

See[3] for more thoughts on data quality

Evaluation

+	−
Quick to spin up... very fast payoff	Lots of overhead to add a new data source
No custom code to maintain, test, or debug	No historical information retained
Easy data analysis – all fields available in combined lookup	Difficult to iterate data model improvements
Easy to fix any search issues and repopulate lookups if needed	Difficult to understand if a given source is expected to map to a particular asset