Vertex

Synapse Bootcamp

Module 11

Building Queries in Storm

v0.4 - May 2024

Objectives

- Learn some helpful Storm commands to aid your analysis
- Learn strategies for building Storm queries
- Understand how Storm's operating concepts apply to queries
- Know which Synapse tools can help you:
  - Create Storm queries
  - Save Storm queries
  - Share Storm queries

Storm Commands

Storm Operations

Operation	Meaning	Common Storm Operator	UI Equivalent
Lift	Select data (nodes) from Synapse	Query bar - Storm	Query bar - Lookup / Text Search `query` and copy menu options
Pivot	Move between nodes that share the same **property value**	-> or <- *	Explore button `pivot` menu option
Traverse	Move between nodes that are linked by an **edge**	-(*)> or <(*)-	Explore button
Filter	Include / exclude a subset of nodes	+ or -	n/a (column filters; `query` / `select` menu options)
Run	Execute a Storm command	| <command>	Node Action
Modify / Edit	Modify or delete properties Add or remove tags Add nodes	[ ] or [( )]	Inline property edit; delete menu option Add / remove tags menu options Lookup or Auto Add / Add Node

Storm Commands

- Storm commands take some action
  - Often acts on the nodes in your query
  - E.g., Power-Ups / Node Actions run Storm commands that enrich data
- Synapse includes a broad range of built-in commands
  - "Anything you can do in the UI, you can do in Storm"
  - Admin tasks:
    - Manage users, roles, permissions, automation, views, layers…
- Some Storm commands are particularly useful for analysis
- Use the pipe character (|) to switch between a Storm operation and a Storm command

Useful Storm Commands

Storm Command	Purpose
`uniq`	Deduplicate ("unique") a set of results
`limit`	Return only the number of results specified
`max`	Display a node with the **highest** value for the specified property or tag
`min`	Display a node with the **lowest** value for the specified property or tag
`count`	Count the total number of nodes returned and display the tally in the Console Tool

Storm Commands - Demo

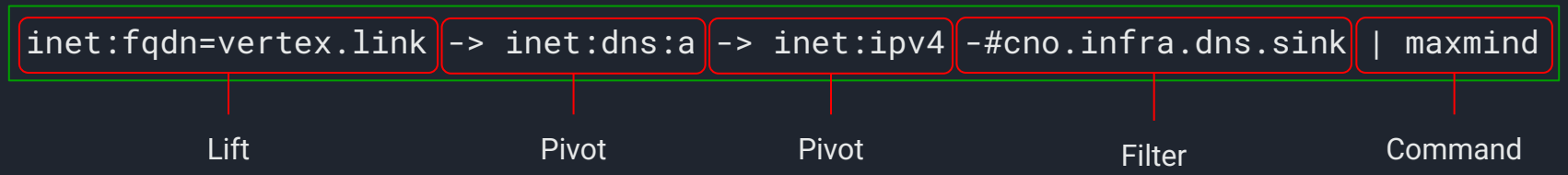
Additional Storm Commands

Storm Command	Purpose
`diff`	Display differences between your forked view and the underlying view
`merge`	Merge some or all data from a forked view to the underlying view
`gen.*`	Create (generate) deconflictable guid-based nodes from user input
`tee`	Perform multiple Storm queries and combine the results
`intersect`	Perform a pivot on multiple nodes and return the results in common
`scrape`	Extract and create (and optionally link) common nodes from text properties
`reverse`	Return the results of a lift operation in reverse-indexed order
`delnode`	Delete a node or nodes
`wget`	Retrieve the content of a URL

Storm Concepts Review

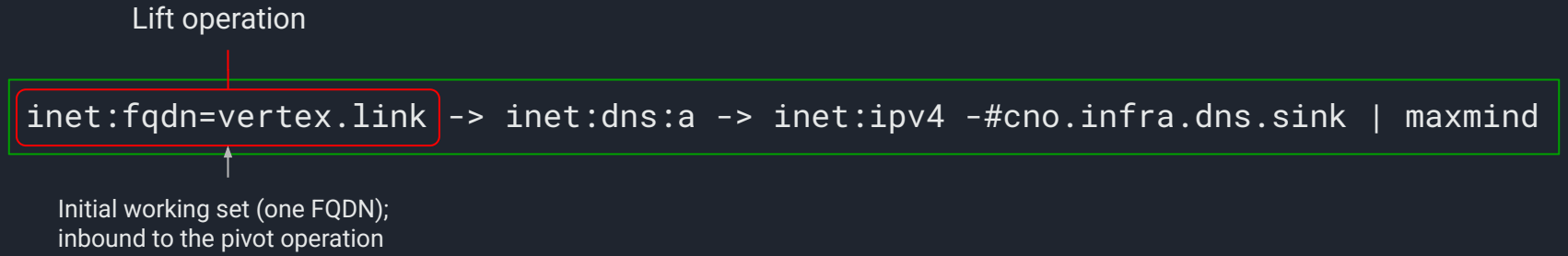
Operations and Operation Chaining

- A Storm query is comprised of individual **operations**
- Queries typically start with a **lift** operation
- Storm operations can be **chained** together to form longer queries
- A chain of Storm operations act as a **pipeline** through which nodes pass
- Nodes pass through the query pipeline **individually**

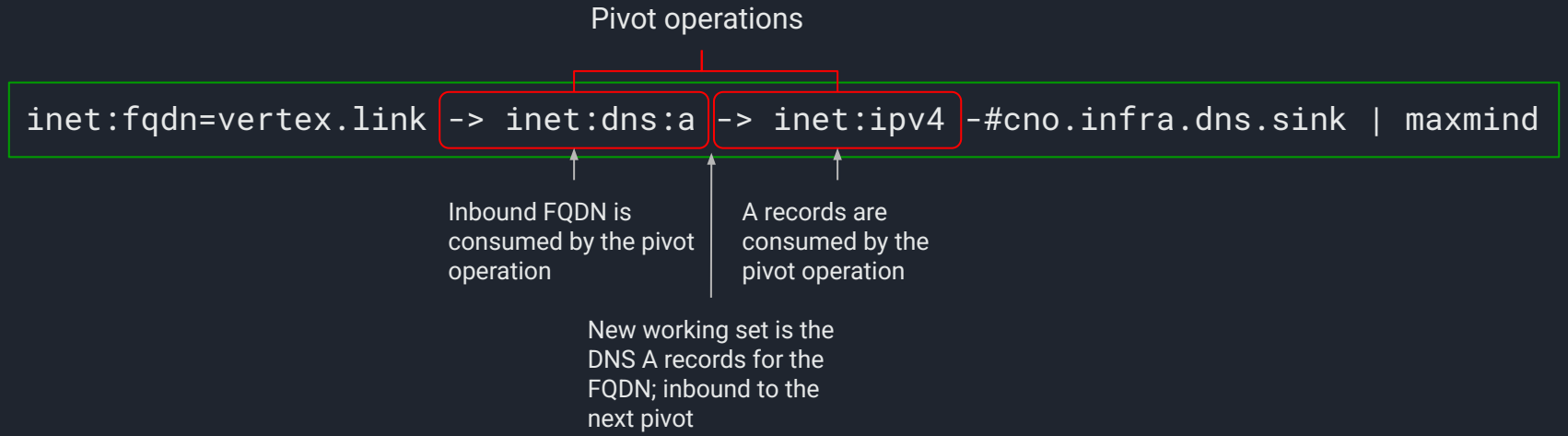


The Storm Pipeline and Working Set

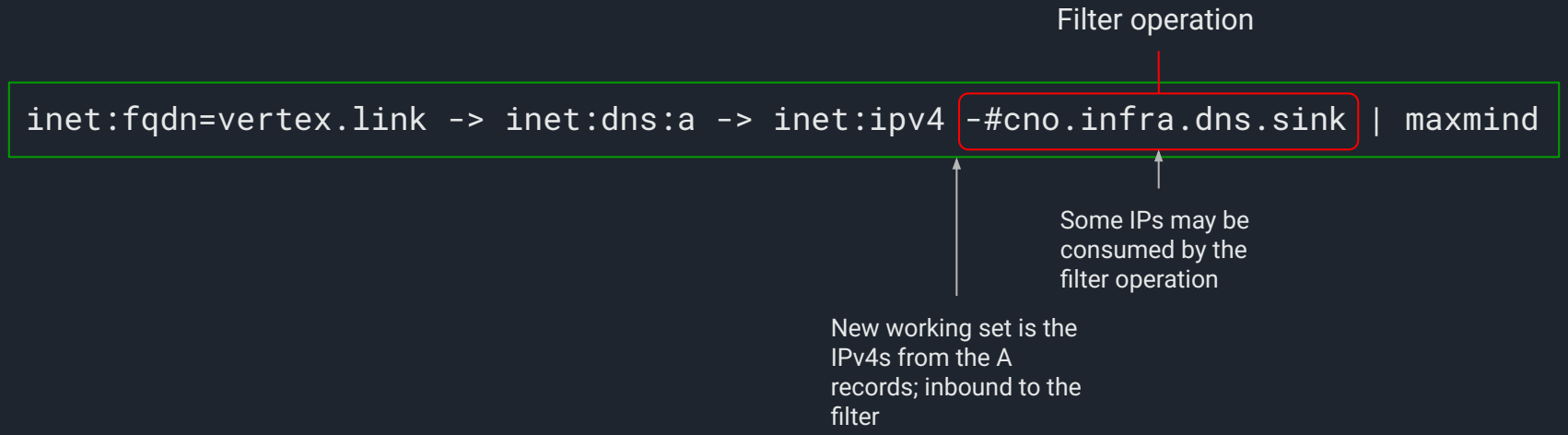
- Because Storm acts as a pipeline, Storm operations are **linear**
  - Nodes (**working set**) start at one end
  - Nodes are **consumed** as they pass through operations from left to right
  - Final working set (**result set**) is the result of the chain of operations



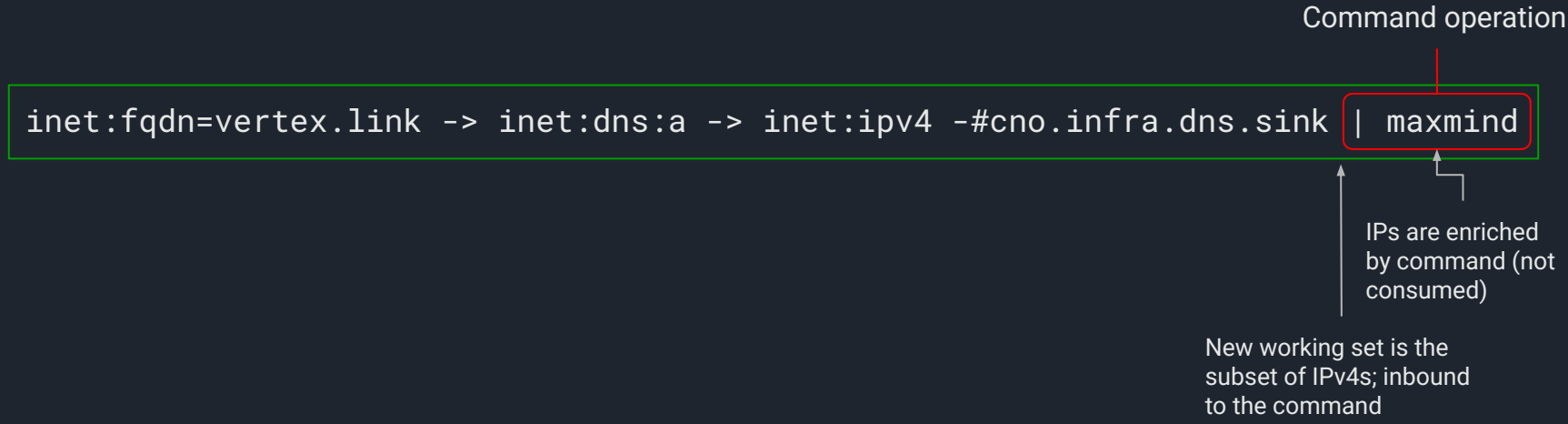
The Storm Pipeline and Working Set



The Storm Pipeline and Working Set



The Storm Pipeline and Working Set



Building Queries Demo

Building Storm Queries

- All queries are built from individual operations
- "Where do I want to go?"
  - Use Storm and the UI to **explore** the data
  - View results, decide on next operation, repeat
- "How do I get from A to D?"
  - Use Storm to answer **specific** questions
  - Given the model / data, what steps do I take?

Protip: With Storm (and Synapse) you can ask (and answer) a question many different ways - there's no single "right" query! (Though some queries may be more efficient than others.)

Storm Resources

Saving / Accessing Storm

- Bookmarks
  - Save and easily run Storm queries
- Custom Node Actions
  - Encode Storm to operate on nodes
- Storm Editor
  - Development environment to create, test, and save queries
- Automation
  - Save **any** Storm to run on a schedule, on demand, or when an event occurs
  - Leverage more advanced Storm features for extra awesomeness

Storm Resources Demo

Storm Tips

"All queries are equal, but some queries are more equal than others." - George Orwell

Lifting "All the Things"

- What is wrong with these queries?

```
inet:ipv4 +:asn=4808
```

```
inet:dns:a +:fqdn=vertex.link
```

```
hash:md5 +#rep.mandiant.apt1
```

Protip: The last example is such a common error that Synapse automatically fixes it for you.

Lift the "Smallest" Thing First

- Which query do you think is most efficient?

```
ou:org +:loc^=us +:name~=vertex
```

```
ou:org:loc^=us +:name~=vertex
```

```
ou:org:name~=vertex +:loc^=us
```

Know When to Uniq

- A pivot goes from **each** inbound node to **each** target for that node

Without uniq:

```
inet:fqdn:zone=scanmalware.info -> inet:dns:a -> inet:ipv4
...
complete. 255 nodes in 6556 ms (39/sec).
```

With uniq:

```
inet:fqdn:zone=scanmalware.info -> inet:dns:a -> inet:ipv4 | uniq
...
complete. 19 nodes in 5650 ms (3/sec).
```

There's not much performance difference between these two queries.
However, if you pivot again, the first query will have to do ~13X the work of the second one.

Know When to Filter

- Pivoting through data we don't care about adds processing overhead and can muddy results

```
inet:dns:a:fqdn=todayusa.org -> inet:ipv4 -> inet:dns:a -> inet:fqdn
```

```
inet:dns:a:fqdn=todayusa.org -> inet:ipv4 -#cno.infra.dns +:type=unicast
  -> inet:dns:a -> inet:fqdn
```

Impact of No Filter / Uniq

"Show me all the FQDNs that resolve to the same IPv4 addresses that all the APT1 FQDNs resolve to"

`inet:fqdn#rep.feye.apt1`	`-> inet:dns:a`	`-> inet:ipv4`	`-> inet:dns:a`	`-> inet:fqdn`	`| uniq`
2,071	10,142	10,142	5,860,567	5,860,418	3,100

Protip: If Synapse is "taking a long time" to load results, the **`count`** command can help troubleshoot!

Synapse UI and Storm

Synapse UI	Storm
Helpful and intuitive	Takes a bit of practice, but learn as you go!
Limitations displaying large data sets	Navigate through as much data as you need
Subset of query and navigational tools	All the power!

Summary

- Create queries using the Storm **operations** as your "building blocks"
- As you're learning, build **step by step**
  - Review results
  - Decide on next operation
  - Use same method to troubleshoot
- Keep Storm's **operating concepts** in mind
  - "What's in my current working set?"
- Use helpful Synapse features to work with Storm
  - Bookmarks, Node Actions, Storm Editor...

Protip: You won't break Synapse by running queries. It's okay to make mistakes (write "non-optimal" queries) - that's how we all learn!