

Enterprise-ready Trino at Bloomberg: *One Giant Leap Toward Data Mesh!*

Engineering

Bloomberg

Trino Summit 2022
November 10, 2022

Vishal Jadhav and Pablo Arteaga
Software Engineers, Analyst Client Engineering

TechAtBloomberg.com

Today's speakers

Pablo Arteaga is a Software Engineer with Bloomberg's Content Generation Platforms Engineering team, which is building a data mesh and the tooling around it to empower data owners to easily manage and share their datasets. He is passionate about creating scalable and reliable systems, and enjoys looking at the lower level details to get a full picture, particularly when troubleshooting networking & OS-level issues.

Vishal Jadhav is a Software Engineer with Bloomberg's Content Generation Platforms, he enjoys working on Trino and its ecosystem. Within Bloomberg, he created an Ansible-based Kubernetes deployment stack for Trino. Within the Trino open source community, he has contributed the OAuth-based authentication API and its implementation to Trino's Python client, and is now working on developing a load balancer for highly-available Trino. He likes to help where help is needed the most, leveraging his 25+ year of experience as coder and hacker.

Catalogs: Our team's main offering

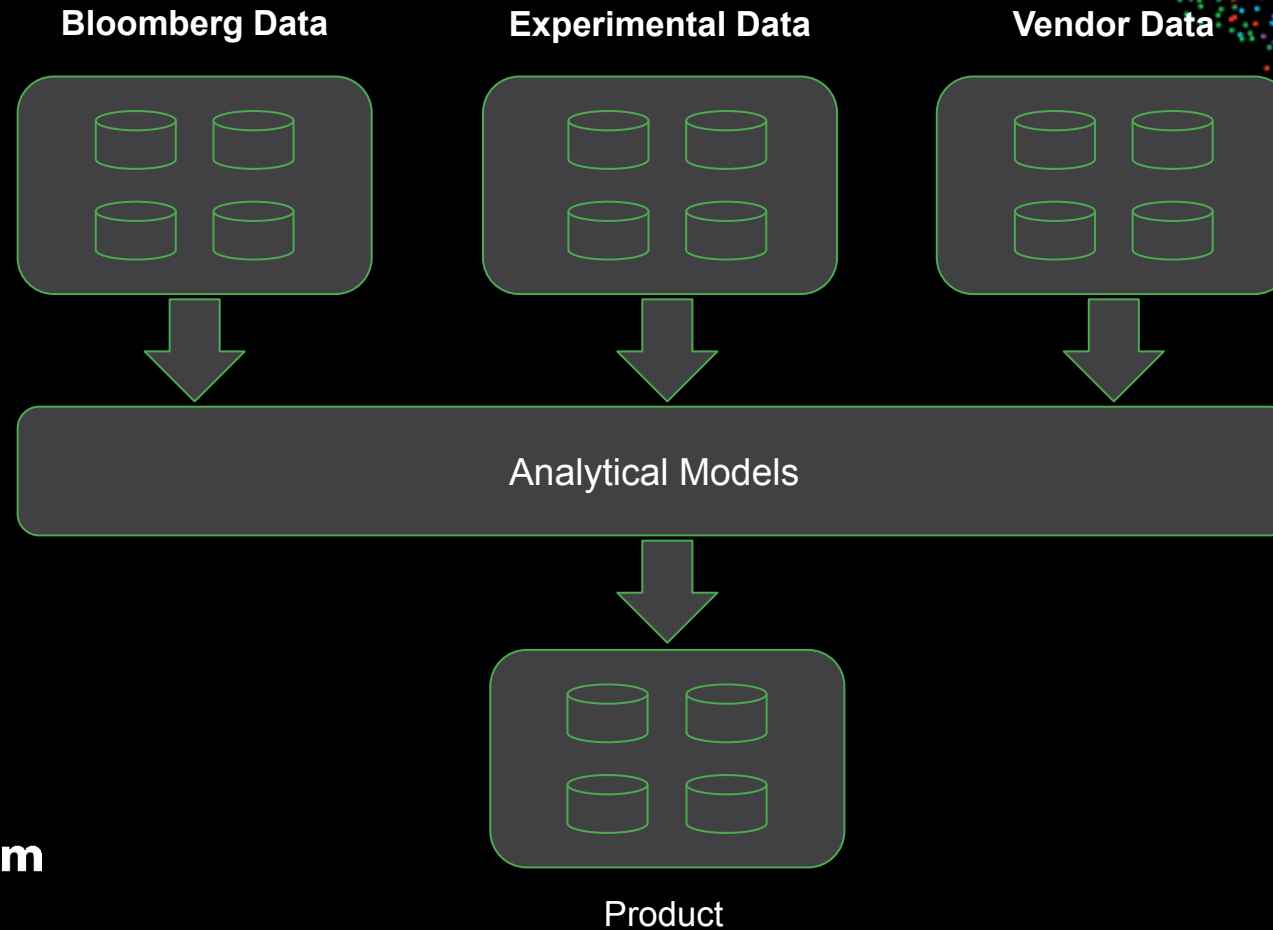
TechAtBloomberg.com

© 2022 Bloomberg Finance L.P. All rights reserved.

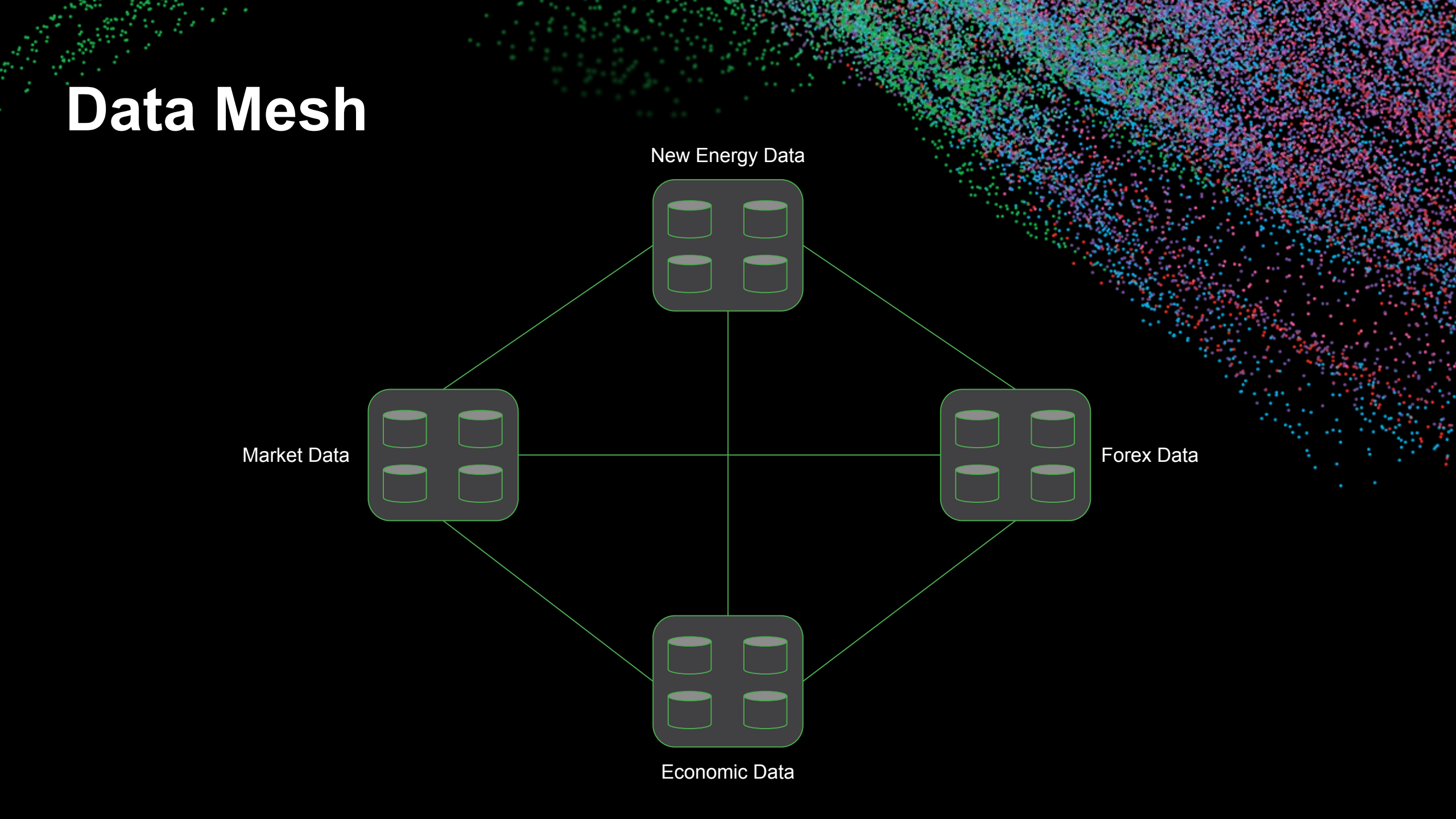
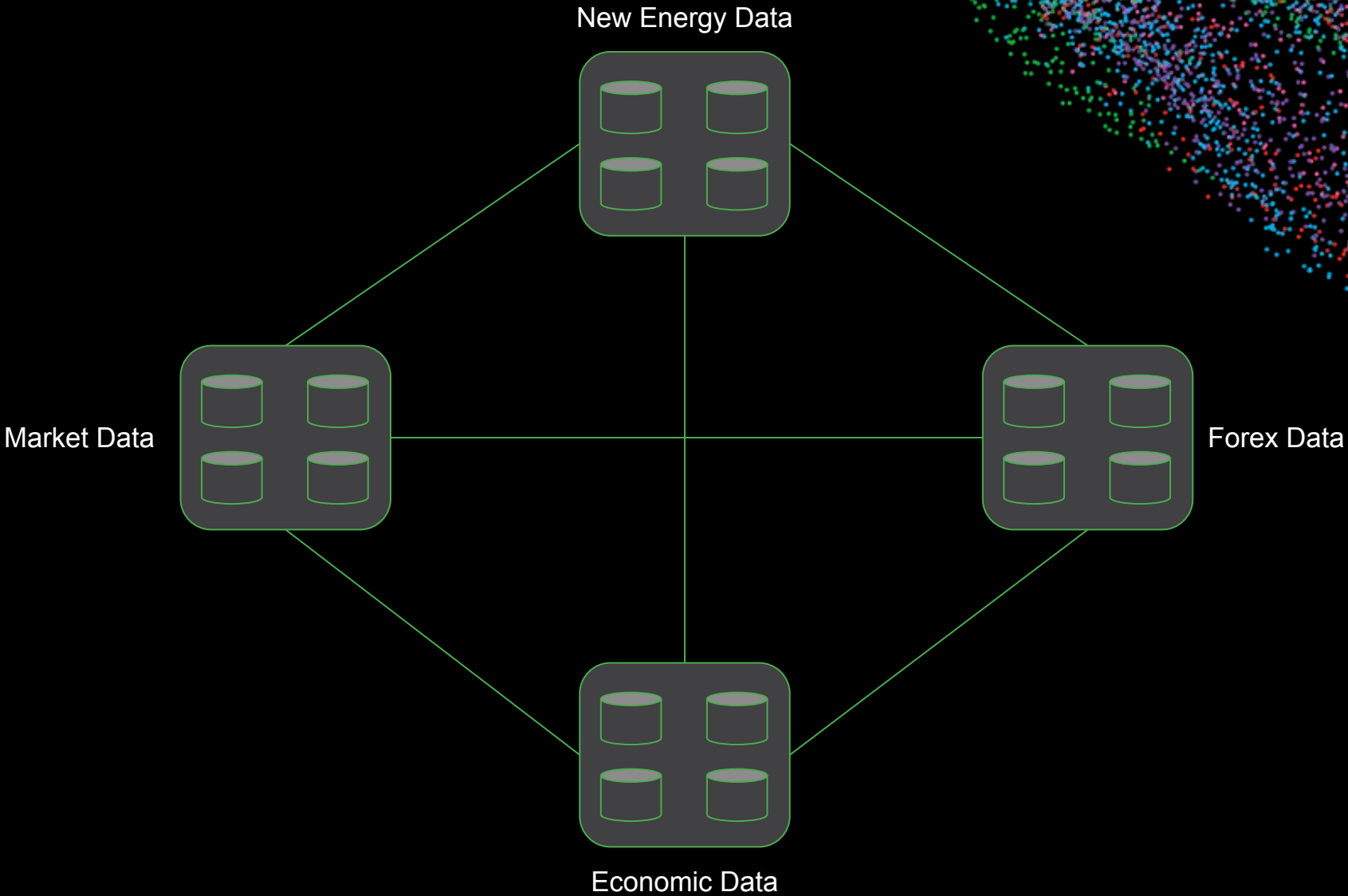
Bloomberg

Engineering

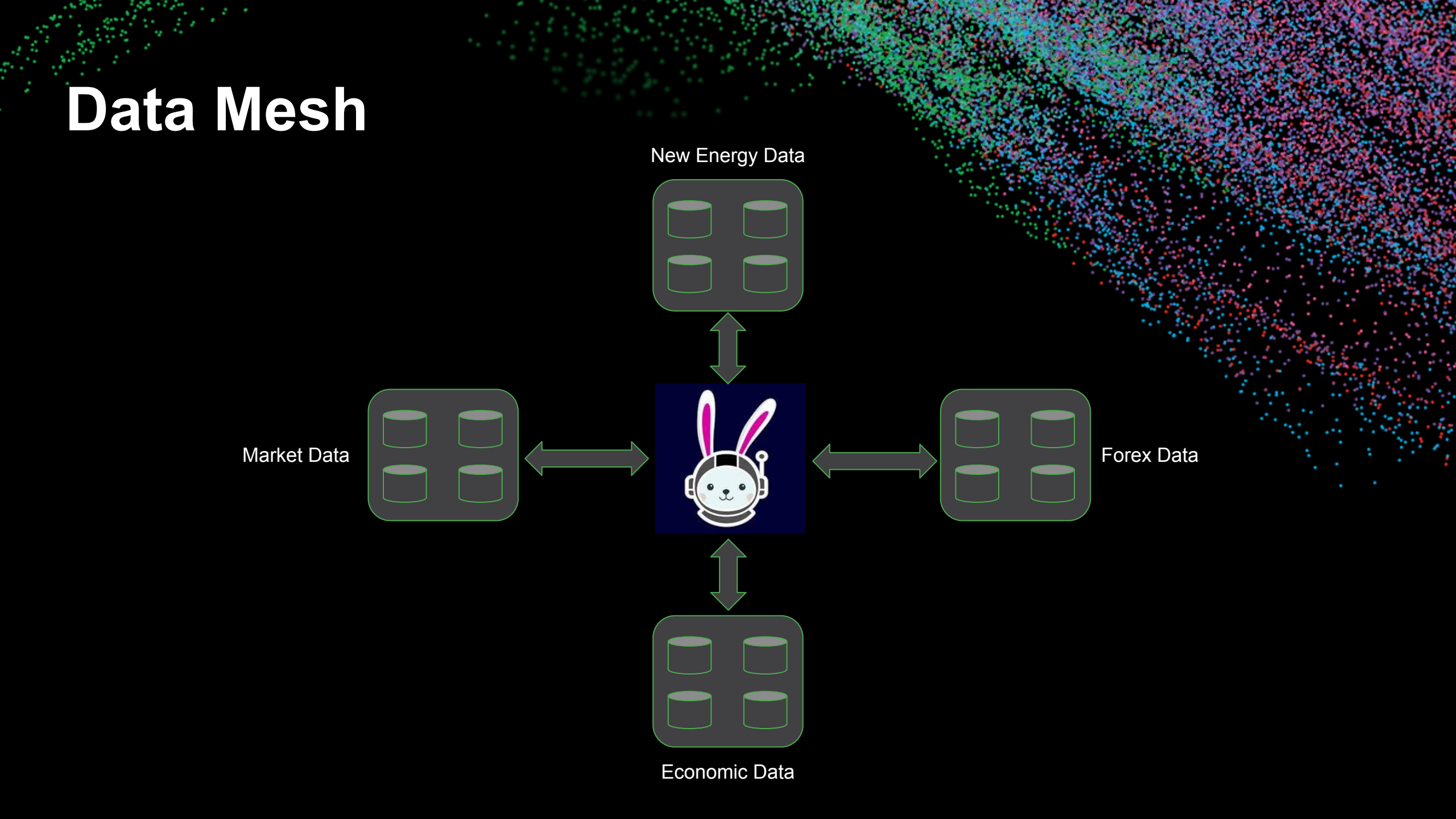
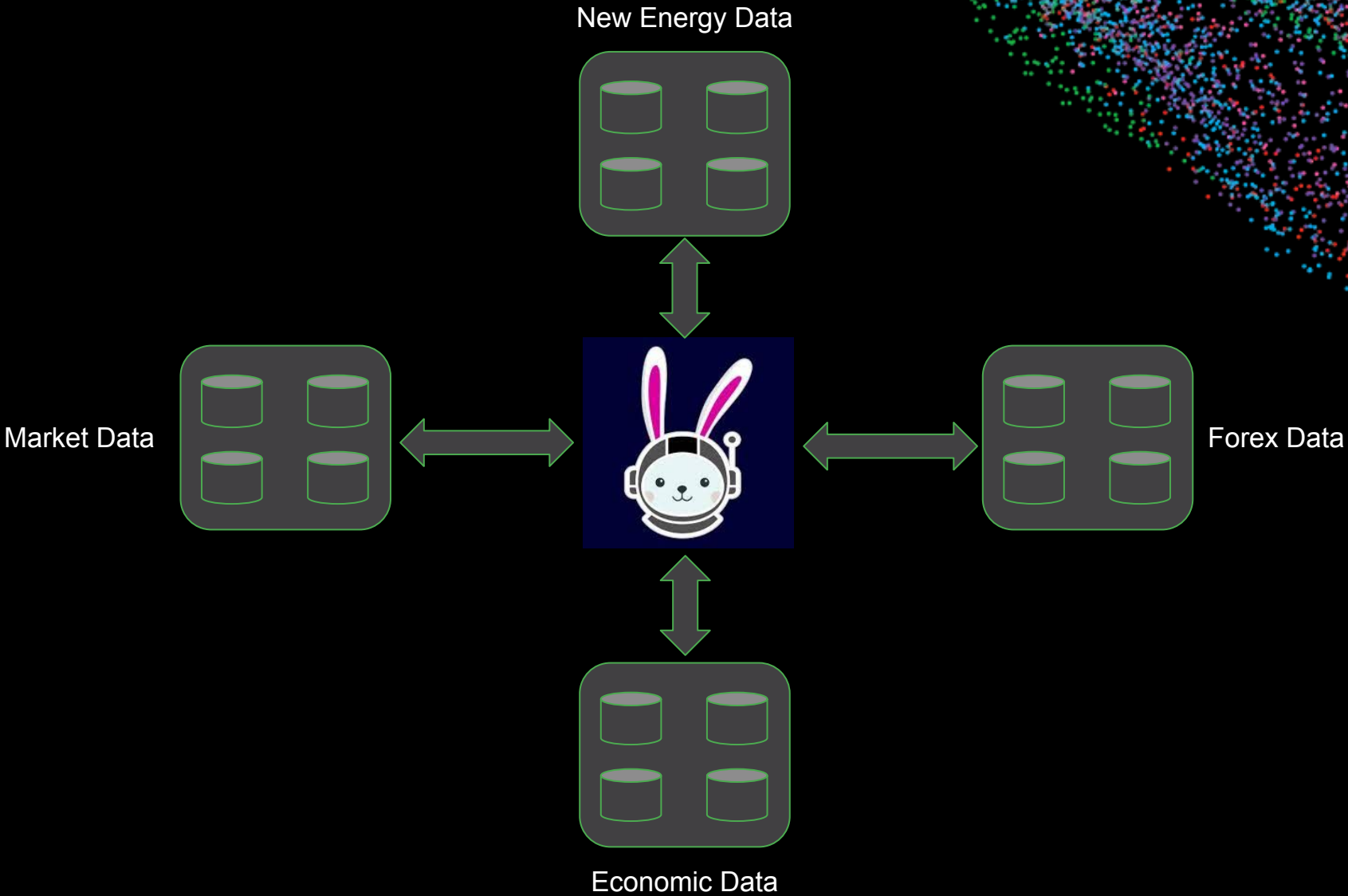
A Research Analyst's Work



Data Mesh

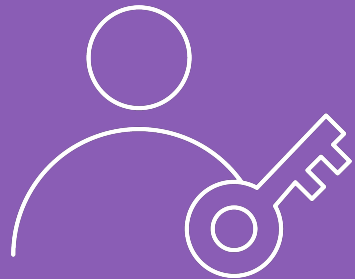


Data Mesh



Catalogs: Our team's main offering

- Catalogs allow **data owners** to **expose their datasets** and **control access** to them
- Catalogs must be:



Access-controlled

Data owners want to control access **themselves**



Highly available

Stakeholders demand **no downtime**, **high throughput** and **fast response times**



Auditable

Knowing **who** accessed **what** and **when** to empower diagnostics & traceability

Easier said than done

High availability is tricky: **Trino coordinator is a single point of failure**

Plenty of scenarios could result in downtime:

- Cluster **upgrades**
- Compute / network / infrastructure **outages**
- Heavy jobs causing **noisy neighbor** issues
- Different stakeholders have different expectations: latency, throughput, etc.
- Trino coordinator **crashes**

Increasing reliance on the **data mesh** creates higher levels of responsibility on us, as the maintainers of our Trino-as-a-service offering



Trino Load Balancer to the rescue

The solution is to add an extra level of indirection... for a change

Trino Load Balancer to the rescue

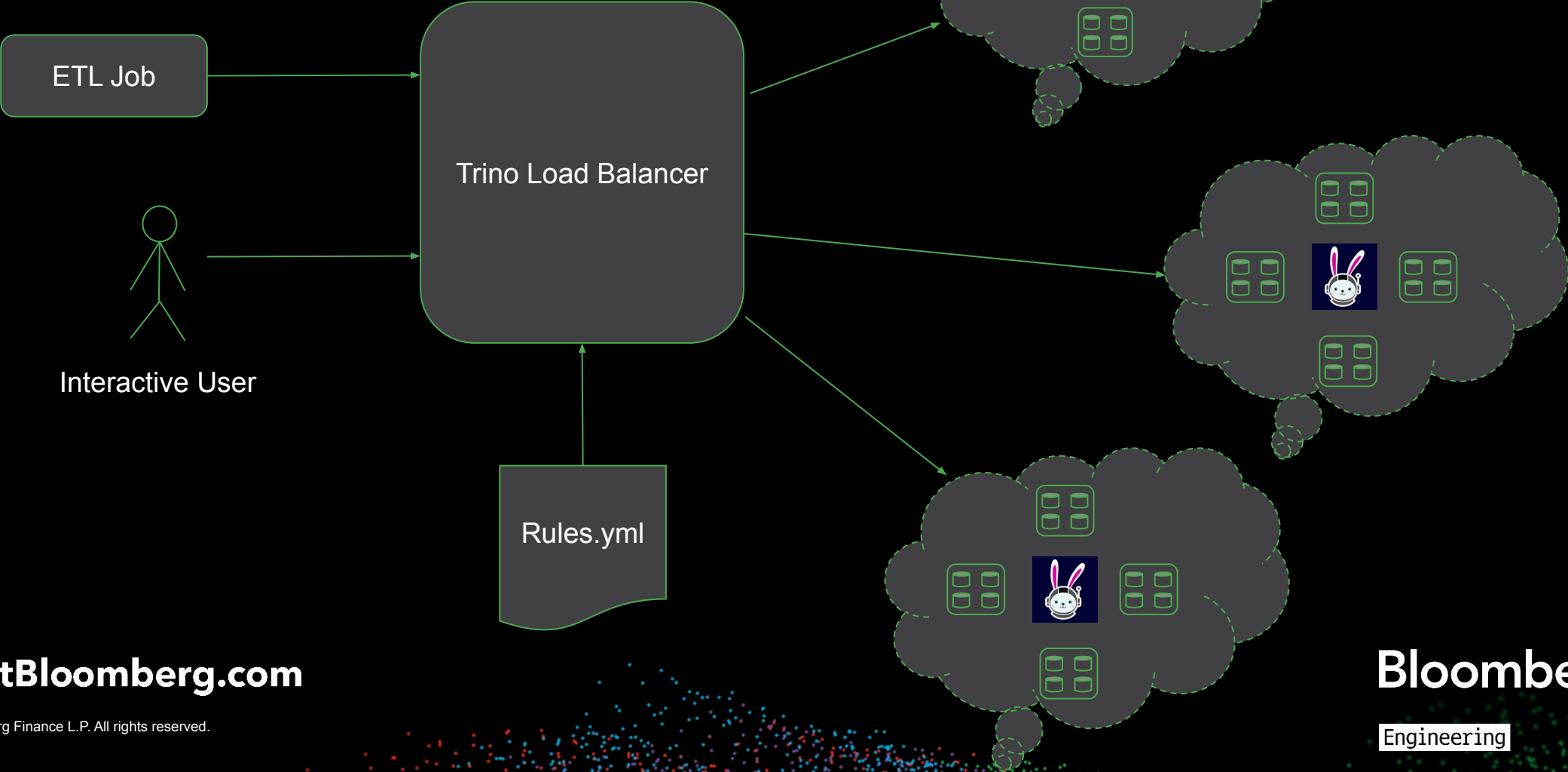
Trino Load Balancer - fork of the open source *presto-gateway*:

- **Retains privacy:** each user only sees their own queries on the LB homepage
- Authentication/Authorization using OAuth sign-on
- TCP connectivity rule
- Easy navigation to the cluster running a query from the Load Balancer homepage

It keeps all the great features of *presto-gateway*:

- **Insight** into the state of each cluster - outstanding queries & where they are running
- Configurable routing based on:
 - Source user
 - Catalog
 - Health of each cluster & environment circumstances (network/CPU load, congestion, ...)

Trino Load Balancer in action

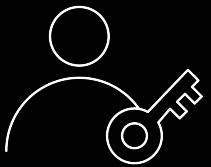


Data security



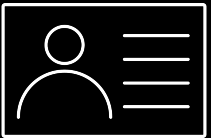
Authorization

- Integration with Apache Ranger
- Granular rules: from catalog-level down to row-level access, depending on use case



Administration of policies

- **Data owners** define access policies using granular rules
- Policies are flexible



Integration with Enterprise directories via LDAP



Traceability: query audit log enables analysis of usage patterns

Demo / screenshots

About the setup for this demo:

- LDAP is our source of truth:
 - 2 users: *limiteduser* and *superuser*
 - Both users are part of group *trino-users*
- Ranger manages policies:
 - Ranger syncs with LDAP through Ranger-Usersync
 - Trino fetches policies from Ranger using the Ranger plugin framework
- Trino authenticates users through LDAP

Demo / screenshots

Example policies:

- As our starting point, we want to:
 - Allow users in *trino-users* to **describe** all schemas
 - Deny **all other access** for all tables by default
- We also want to:
 - Allow *superuser* to query **any** catalog, schema, and table
 - Allow *limiteduser* to read from *tpcds.sf1.call_center* (built-in sample data), but **nothing else**

Service Name *

Display Name

Description

Active Status Enabled Disabled

Select Tag Service

Config Properties :

Username *

Password

jdbc.driverClassName *

jdbc.url *


































Add New Configurations

Name	Value
------	-------

List of Policies : trinoservice

Search for your policy...

Add New Policy

Policy ID ▲	Policy Name	Policy Labels	Status	Audit Logging	Roles	Groups	Users	Action
1	all - trinouser	--	Enabled	Enabled	--	trino-users	--	  
2	all - catalog	--	Enabled	Enabled	--	trino-users	--	  
3	all - function	--	Enabled	Enabled	--	trino-users	--	  
4	all - catalog, sessionproperty	--	Enabled	Enabled	--	trino-users	--	  
5	all - catalog, schema, procedure	--	Enabled	Enabled	--	trino-users	--	  
6	all - catalog, schema, table	--	Enabled	Enabled	--	trino-users	--	  
7	all - systemproperty	--	Enabled	Enabled	--	trino-users	--	  
8	all - catalog, schema, table, column	--	Enabled	Enabled	--	--	superuser	  
9	all - catalog, schema	--	Enabled	Enabled	--	trino-users	--	  
10	allow limiteduser to tpcds.sf1.call_center	--	Enabled	Enabled	--	trino-users	--	  
11	Allow information schemas	--	Enabled	Enabled	--	trino-users	--	  

Edit Policy

Policy Details:

Policy Type **Access**

[Add Validity Period](#)

Policy ID **11**

Policy Name *

Enabled Normal

Policy Label

catalog * **Include**

schema * **Include**

table * **Include**

column * **Include**

Description

Audit Logging **Yes**

column * Include

Description

Audit Logging Yes

- add/edit permissions
- Select
 - Insert
 - Create
 - Drop
 - Delete
 - Use
 - Alter
 - Grant
 - Revoke
 - Show
 - Impersonate
 - All
 - execute
 - Select/Deselect All

Allow Conditions:

Select Role	Select Group	Select User	Permissions	Delegate Admin
<input type="text" value="Select Roles"/>	<input type="text" value="x trino-users"/>	<input type="text" value="Select Users"/>	<input checked="" type="checkbox"/> Select <input checked="" type="checkbox"/> Use <input checked="" type="checkbox"/> Show <input checked="" type="checkbox"/> [x]	<input type="checkbox"/> [x]

+

Exclude from Allow Conditions:

Select Role	Select Group	Select User	Permissions	Delegate Admin
<input type="text" value="Select Roles"/>	<input type="text" value="Select Groups"/>	<input type="text" value="Select Users"/>	Add Permissions +	<input type="checkbox"/> [x]

+



Policy Label Policy Label

catalog * x tpcds Include

schema * x sf1 Include

table * x call_center Include

column * x * Include

Description

Audit Logging Yes

Allow Conditions:

Select Role	Select Group	Select User	Permissions	Grant Admin
Select Roles	x trino-users	Select Users	Select Use Show	

add/edit permissions

- Select
- Insert
- Create
- Drop
- Delete
- Use
- Alter
- Grant
- Revoke
- Show
- Impersonate
- All
- execute
- Select/Deselect All

hide ▲

```
Superuser < N/A >
* <Superuser> Script-3 x * <Limiteduser> Script
-- Running as superuser

• -- Showing schemas works
SHOW SCHEMAS from tpcds;

• -- We can query what tables exist
SHOW TABLES from tpcds.sf1;

• -- Selecting from call_center
select * from tpcds.sf1.call_center limit 2;
• -- Selecting from catalog_page
select * from tpcds.sf1.catalog_page limit 2;

• -- Selecting from another schema in the catalog works since
  -- superuser is entitled to everything
select * from tpcds.sf100.catalog_page limit 2;
```

```

-- Running as limiteduser

•-- Showing schemas works
SHOW SCHEMAS from tpcds;

•-- We can query what tables exist
SHOW TABLES from tpcds.sf1;

•-- Selecting from call_center works - limiteduser is entitled
select * from tpcds.sf1.call_center limit 2;

•-- Selecting from another table for which we
  -- are not permissioned fails
select * from tpcds.sf1.catalog_page limit 2;
•-- But we can see its schema
describe tpcds.sf1.catalog_page;

•-- Selecting from another schema in the catalog does not work
select * from tpcds.sf100.catalog_page limit 2;

```

Trino Load Balancer

Queries Clusters Editor

Started at : 11/9/2022, 2:51:33 PM

Query details [history size = 6]

Search:

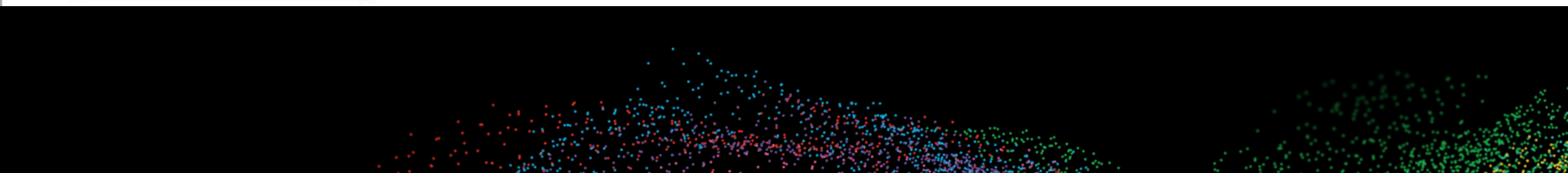
Show entries

queryId	routedTo	user	source	queryText	submissionTime
20221109_225325_00002_vn44t	http://trino-region2:8080	root	trino-cli	show schemas from system	11/9/2022, 2:53:25 PM
20221109_225324_00006_47k2w	http://trino-region1:8080	root	trino-cli	show schemas from system	11/9/2022, 2:53:24 PM
20221109_225313_00005_47k2w	http://trino-region1:8080	root	trino-cli	show catalogs	11/9/2022, 2:53:13 PM
20221109_225312_00004_47k2w	http://trino-region1:8080	root	trino-cli	show catalogs	11/9/2022, 2:53:12 PM
20221109_225311_00003_47k2w	http://trino-region1:8080	root	trino-cli	show catalogs	11/9/2022, 2:53:11 PM
20221109_225307_00002_47k2w	http://trino-region1:8080	root	trino-cli	show catalogs	11/9/2022, 2:53:07 PM

Showing 1 to 6 of 6 entries

Previous Next

Query history distribution



Trino Load Balancer

Queries

Clusters

Editor

Started at : 11/9/2022, 2:51:33 PM

All backends:

Name	Url	Group	Active	Queued	Running
region1	http://localhost:8081	region1	true	0	0
region2	http://localhost:8082	region1	true	0	0

Trino Load Balancer

Queries

Clusters

Editor

!!Admin ONLY!!

Override the Database (Optional):

Select config type

Select entity to edit

✚ ✚ ↶ ↷ 🔍 ▼ ▲

- object {5}
 - name : region1
 - proxyTo : <http://trino-region1:8080>
 - active : true
 - routingGroup : region1
 - externalUrl : <http://localhost:8081>

```
{  
  "name": "region1",  
  "proxyTo": "http://trino-region1:8080",  
  "active": true,  
  "routingGroup": "region1",  
  "externalUrl": "http://localhost:8081"  
}
```


Putting it all together

- Trino Load Balancer allows us to **maintain high availability**
- Federated access policies allow data owners to retain control over their datasets
- Data security as a first-level priority: authentication, traceability

All of this using well-known, open source components:

- Apache Ranger
- Trino
- Trino Load Balancer

Thank you!

<https://www.bloomberg.com/careers>

Contact us:

Vishal Jadhav (vjadhav@bloomberg.net)

Pablo Arteaga (parteagagonz@bloomberg.net)

Engineering

Bloomberg

TechAtBloomberg.com

© 2022 Bloomberg Finance L.P. All rights reserved.