DataStax

DataStax Monday Learning

- From engineers to engineers
- Open-source only
- Hands-on experience
- Live communication with experts

DataStax

June 21st Led By Aleks and Zeke

Event Streaming Series: Ep. III

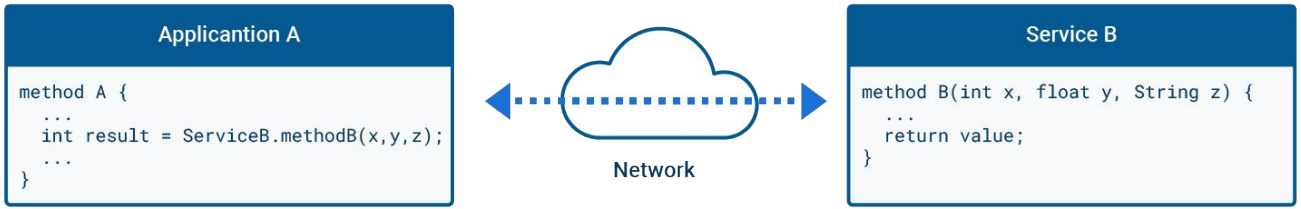
Building an Advanced Event Streaming System

DataStax

Why Messaging?

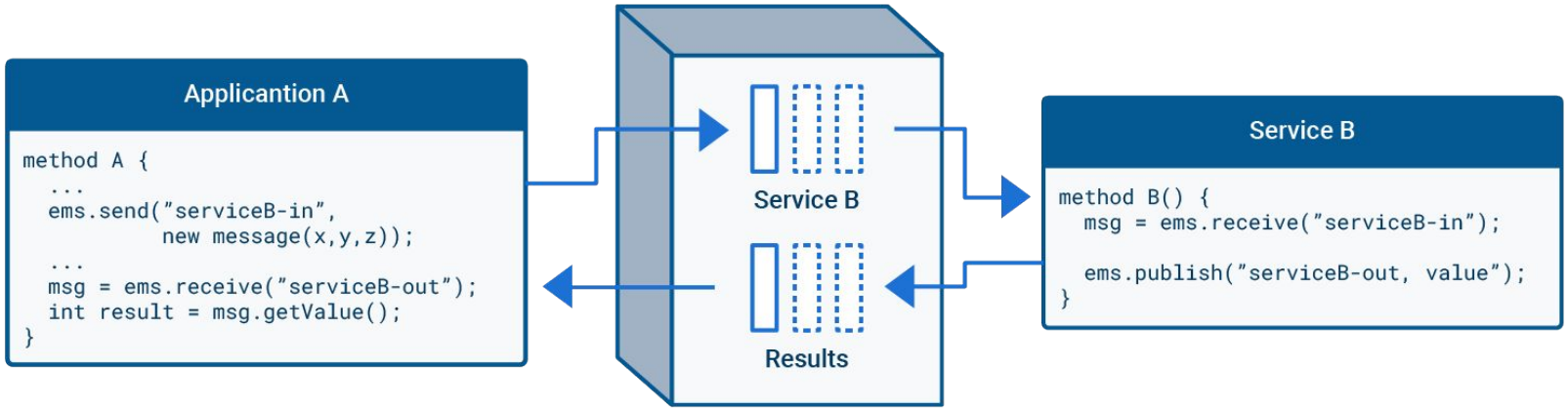
Remote Procedure Calls – Pain before Messaging Systems

- Remote Procedure Calls
  - Application A makes a request from service B
  - Service B responds

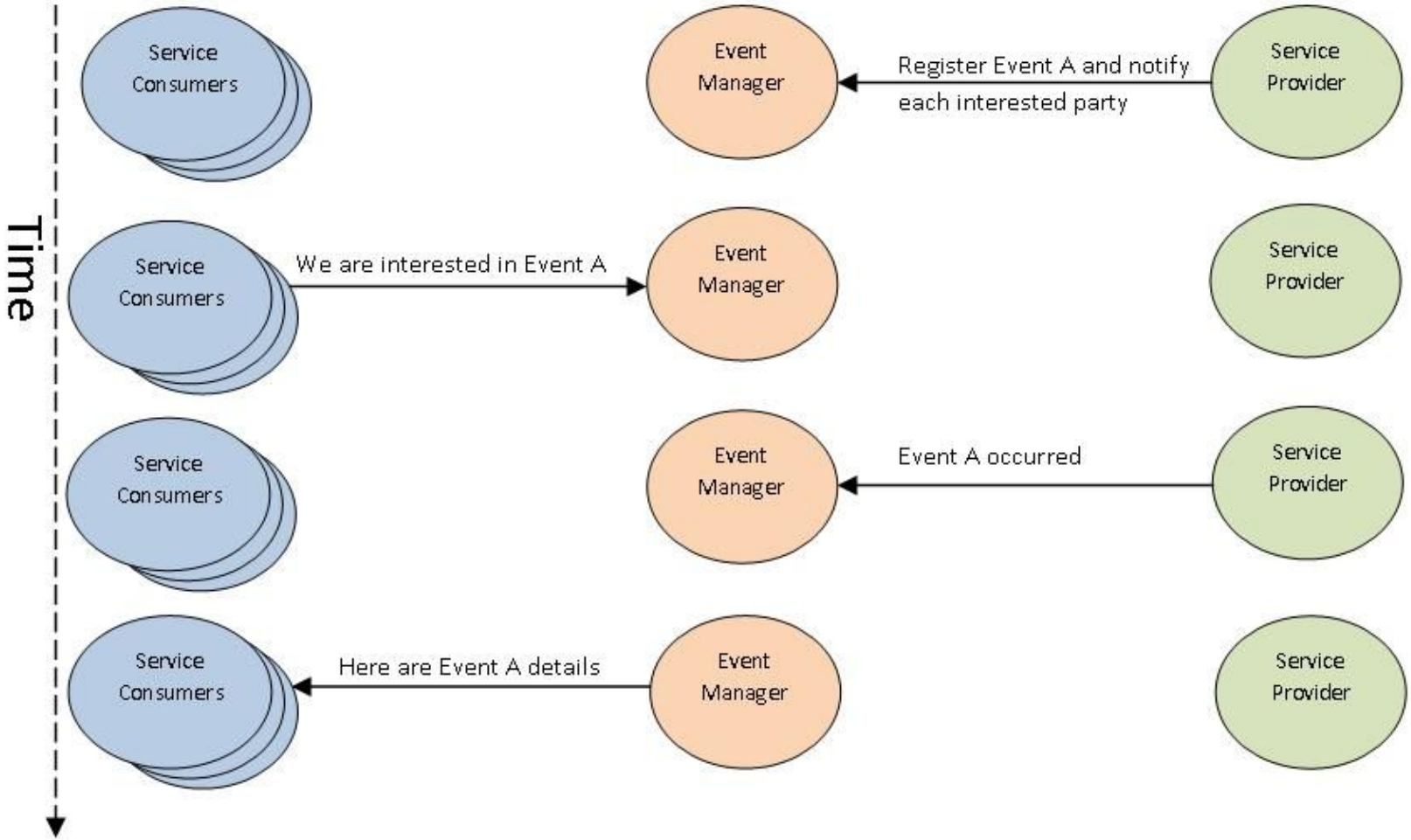


- Problems
  - Huge risk with networking
  - Needs service discovery
  - Point to Point communications
  - Slow (blocking) as you are waiting for one server to respond to another
  - If a receiver is not available, process is failed
  - One and Only One receiver

Message Systems solved RPC limits

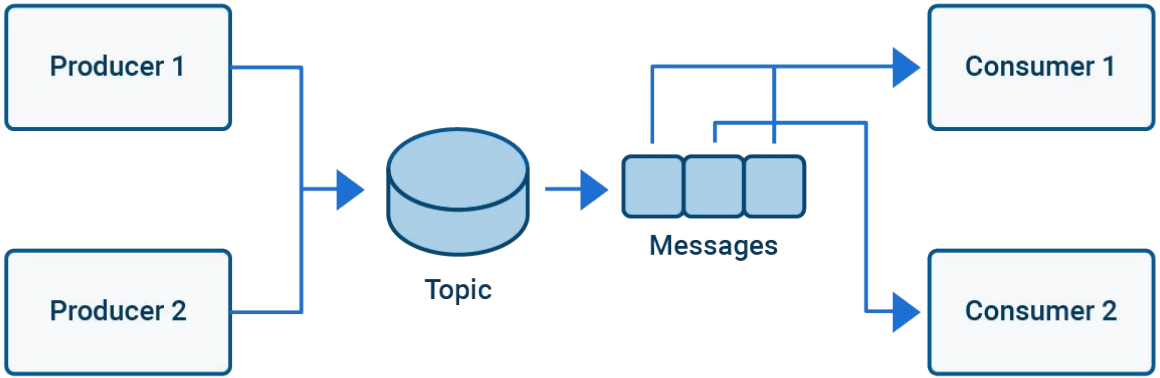


- Decouple message senders from the receivers by using a **messaging service**
- Persistent buffer to handle requests and data
- Async = no need to wait!
- Multiple Receivers!
- Perfect fit for distributed systems / microservices
- Message will be consumed by the next available service instance


Time
Service Consumers
Event Manager
Register Event A and notify each interested party
Service Provider
Service Consumers
We are interested in Event A
Event Manager
Service Provider
Service Consumers
Event Manager
Event A occurred
Service Provider
Service Consumers
Here are Event A details
Event Manager
Service Provider

Queues

- Multiple Producers, One Consumer
- But you can have thousands in parallel
- Consumers share work
- Perfect fit for background processing
- Easy to scale
- Easy to distribute

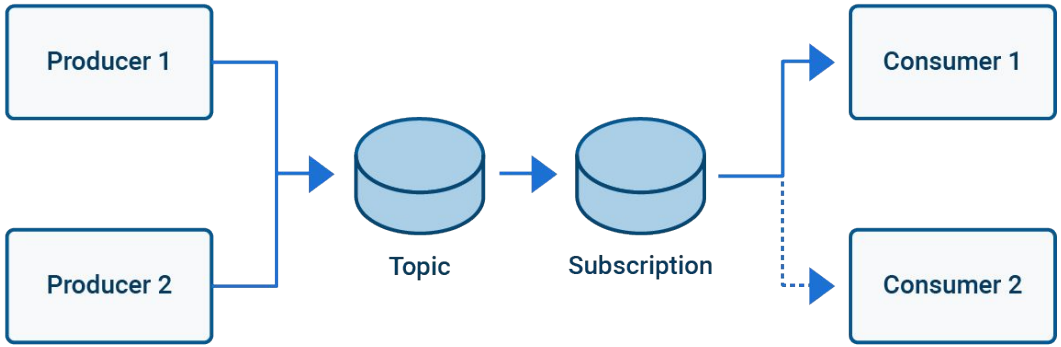


Use-Case: workers

Notifications

- Multiple Producers, Multiple Consumers
- Perfect fit for decentralized processing
- Easy to extend

Use-Case: Microservices



Pulsar Components and Ecosystem

Zookeeper

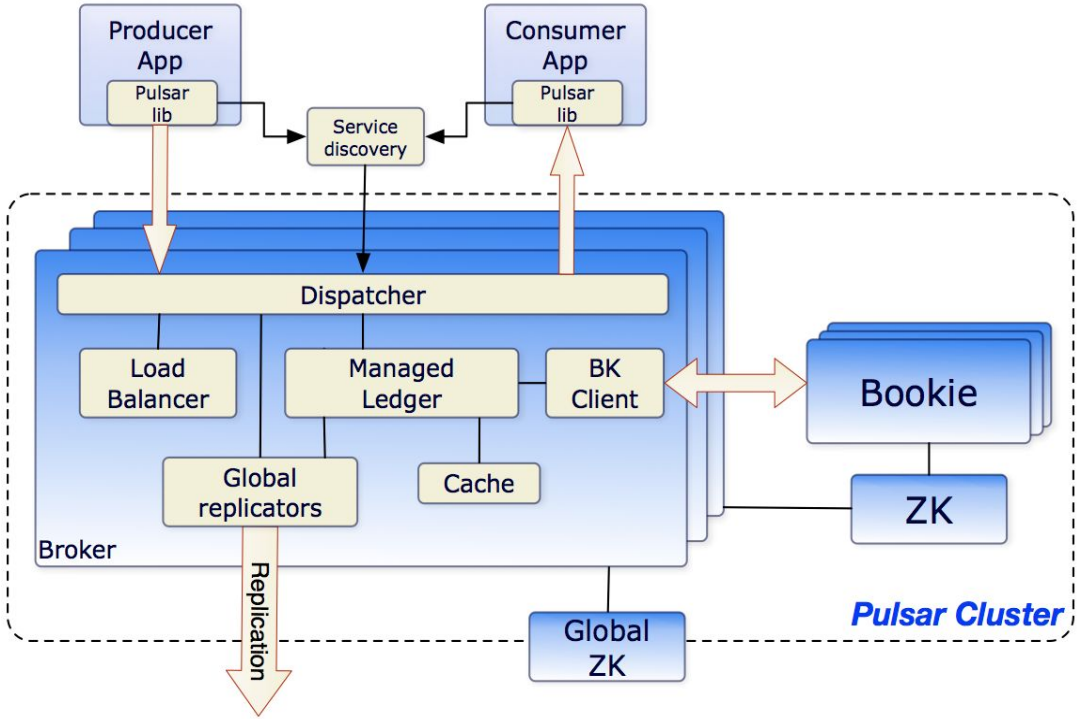
- For metadata storage, cluster configuration, and coordination

- Instance/Global level Zookeeper
  - Geo-replication
  - Configuration for tenants, namespaces, and other entities that need to be globally consistent
  - Optional

- Cluster level Zookeeper
  - Ownership metadata
  - Broker load reports
  - BookKeeper ledger metadata
  - … …

Broker

- Stateless
- Topic ownership
- Load Balancing
- Pulsar's "Brain"
  - HTTP Rest APIs for Admin tasks and topic lookup
  - TCP binary protocol for data transfers



Broker (Topic Ownership), cont'd

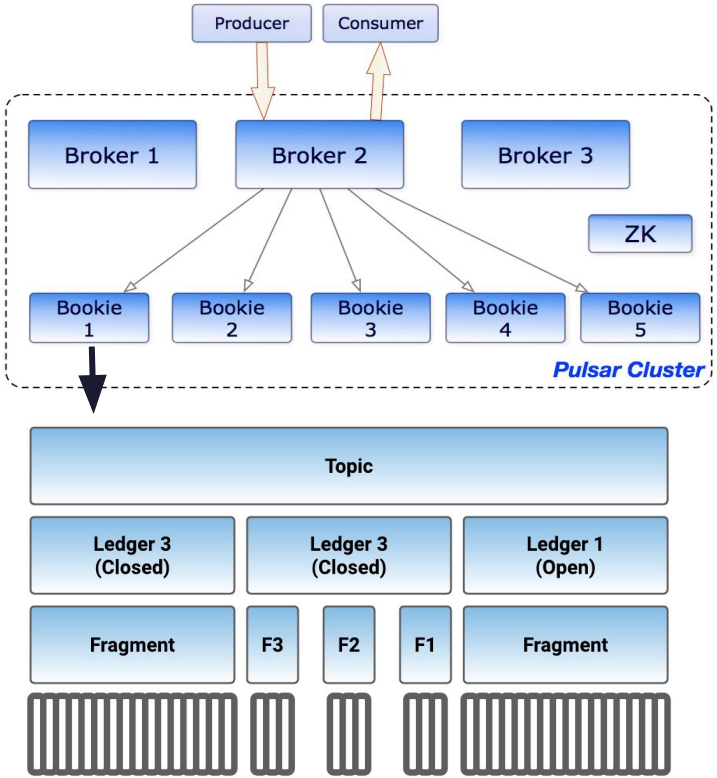
- Stateless broker makes possible of **dynamic assignment**
  - Topic ownership can change on the fly
  - Broker crash
  - Overloaded broker
- Ownership assignment granularity
  - Namespace **bundle** (subset of a namespace)
    - defaultNumberOfNamespaceBundles=4
  - Consistent hashing
    - C* ring

Broker (Load Balancing), cont'd

- Unload topics and bundles
  - Triggers topic re-assignment based on the current workload
  - Automatic or manual
- Bundle Split
  - Tunable thresholds
  - Automatically triggers topics unloading
- Automatic Load Shedding
  - Forces topics unloading from overloaded brokers
  - Enabled by default and can be disabled

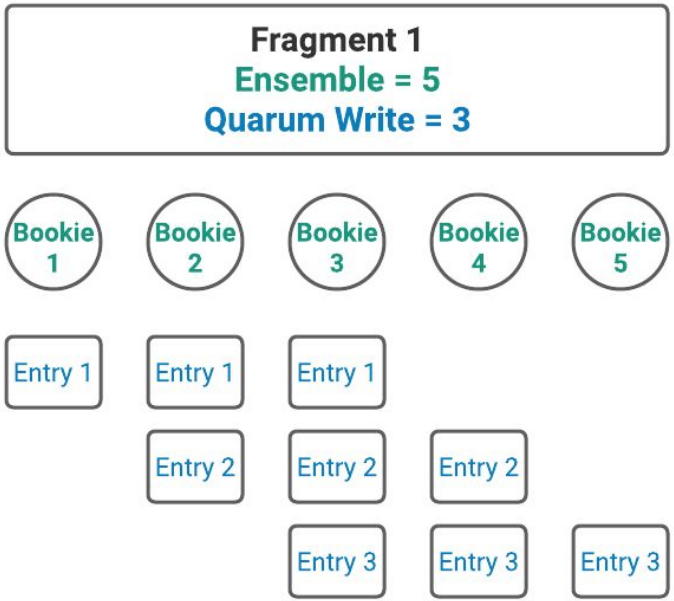
Bookkeeper (Bookie)

- Managed ledger (Topic)
  - Logical abstraction
  - A stream of ledgers
- A Ledger is created when:
  - A new topic is created, or
  - Roll-over occurs (size/time limit, ownership change)
- Append only
  - Read only after closed
- Fragments and Entries

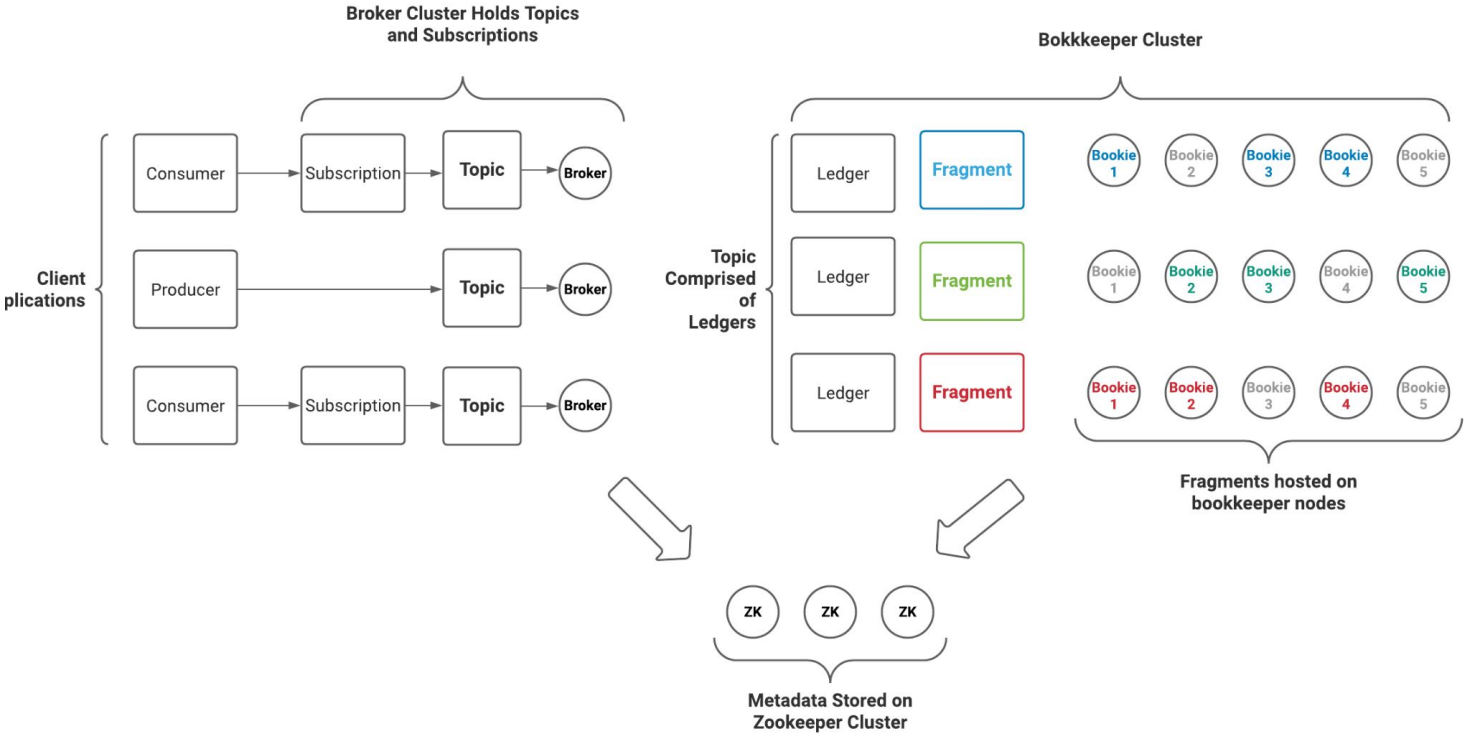


Bookkeeper (Bookie), cont'd

- Key Configuration
  - Ensemble Size (E)
    - The size of the pool of Bookies available for writes
  - Write Quorum Size (Qw)
    - The number of actual Bookies that Pulsar will write an entry to
  - Ack Quorum Size (Qa)
    - The number of Bookies that must acknowledge the write
- Rack-awareness
- WAL Journal
  - Seperate Journal and Ledger disks



Put all Together

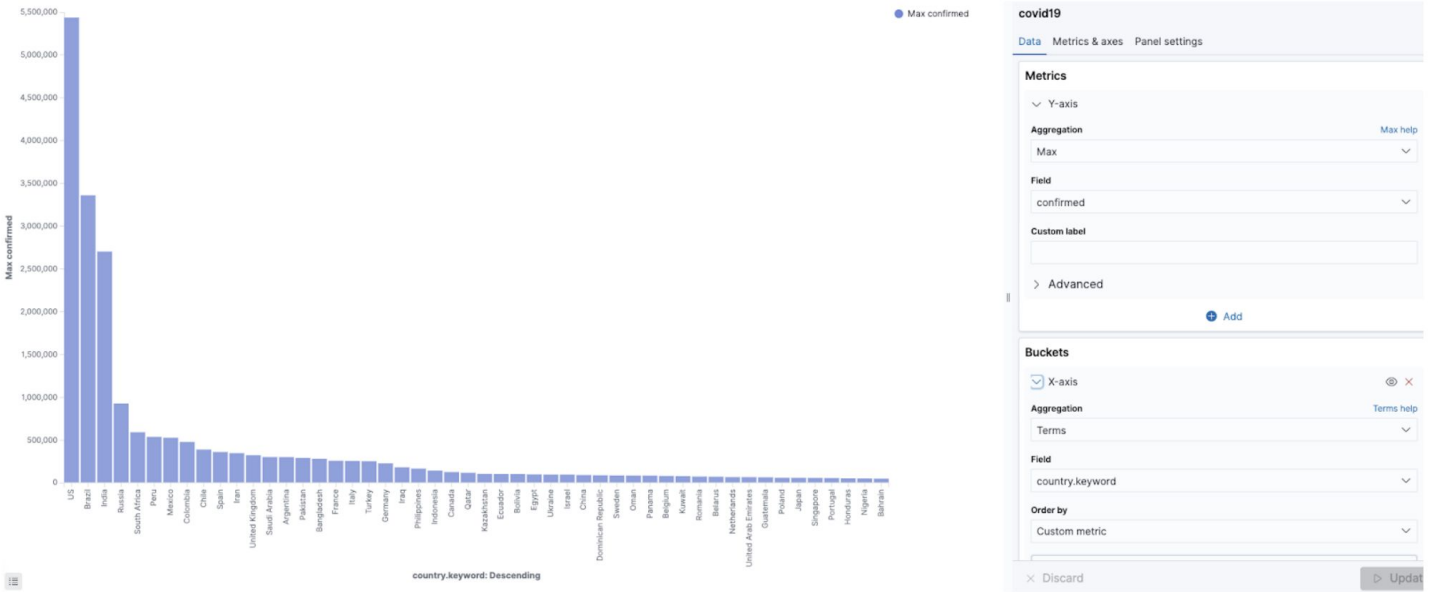


COVID19
Pulsar Ecosystem Application

Example

Covid19 Data Parsing

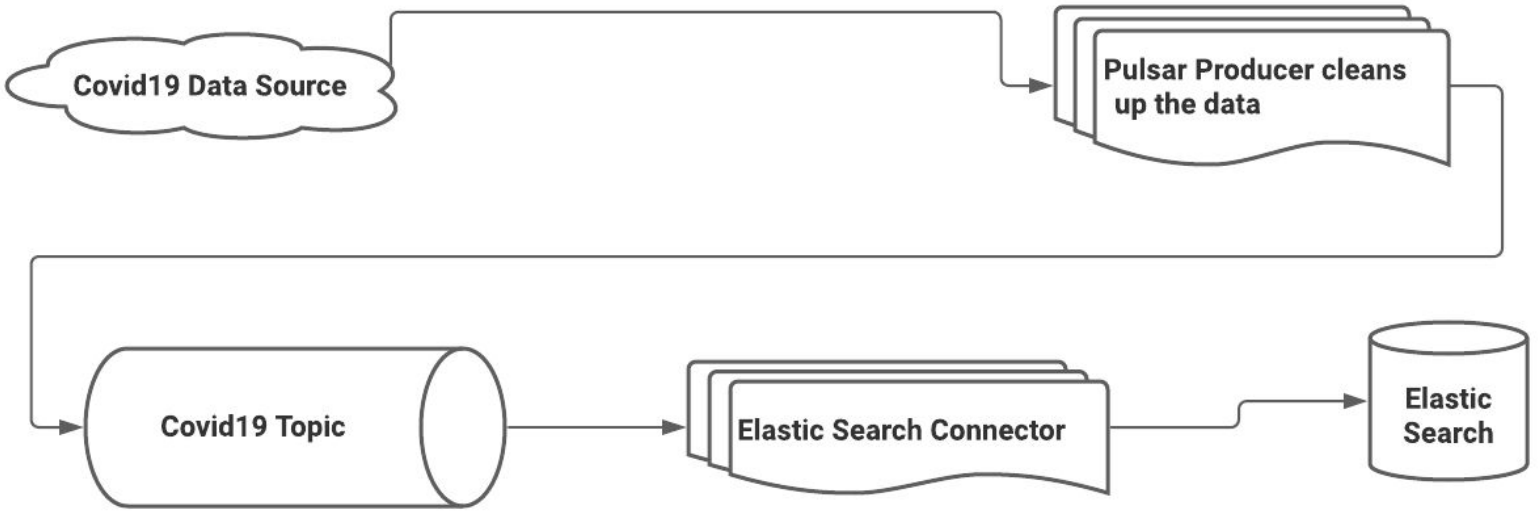
- Huge data set - +100k records
- Updated once a day
- REST API



Application Architecture

https://github.com/meticulo3366/covid19-with-pulsar

- Huge data set - +100k records
- Updated once a day
- REST API



Pulsar Producer Code

```
import json
import datetime
import urllib.request, json
import pulsar
from pulsar.schema import *

class Covid19(Record):
    date = String()
    confirmed = Integer()
    deaths = Integer()
    recovered = Integer()
    country = String()

with urllib.request.urlopen("https://pomber.github.io/covid19/timeseries.json") as url:
    covid = json.loads(url.read().decode())

client = pulsar.Client('pulsar://pulsar:6650')

producer = client.create_producer(topic='covid19',schema=AvroSchema(Covid19))

count=0
for key in covid:
    print("Processing all entries for -> "+ key)
    for i in covid[key]:
        # we need to ensure our data format is correct to merge the data streams
        # i.e. -> “date”:“2020-01-02" != “date”:“2020-1-2",
        record_date = datetime.datetime.strptime(i['date'], '%Y-%m-%d')
        record_date = record_date.strftime('%Y-%m-%d')
        i['date'] = record_date
        i['country'] = key
        #we are sending our records to a new topic called covid19US
        record = Covid19(date=i['date'],country=i['country'],confirmed=i['confirmed'],deaths=i['deaths'],recovered=i['recovered'] )
        producer.send( partition_key=record_date, content=record )
        count+=1

print( str(count) + " Covid19 Records Loaded into Apache Pulsar!")
client.close()
```

Pulsar Connector to Write into ElasticSearch

Configure the Connector

```
configs:
  ...
  topics: sensor-data
  ...
  topic:
    sensor-data:
      iot:
        temperatures_by_sensor:
          mapping: 'sensor=value.sensorId,timestamp=value.timestamp,value=value.value'
  ...
```

Enable the Connector

```
pulsar-admin sinks create \
      --name sensor-data-sink \
      --sink-type cassandra-enhanced \
      --inputs persistent://public/default/sensor-data \
      --sink-config-file /root/sensor-data-sink.yml
```

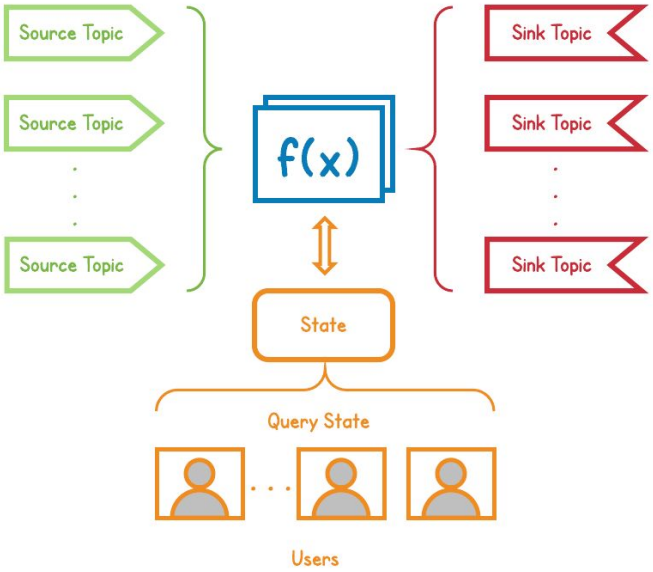
DEMO TIME!

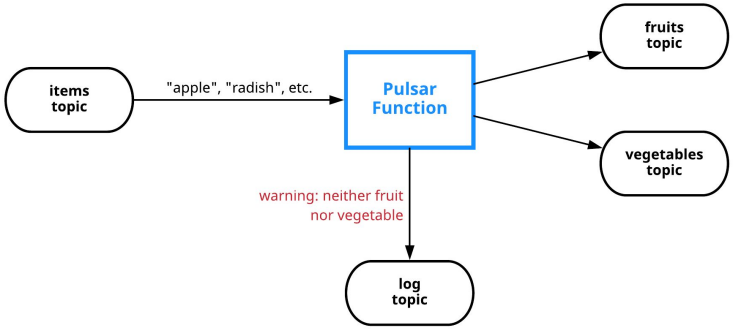
🤓 💻 👇

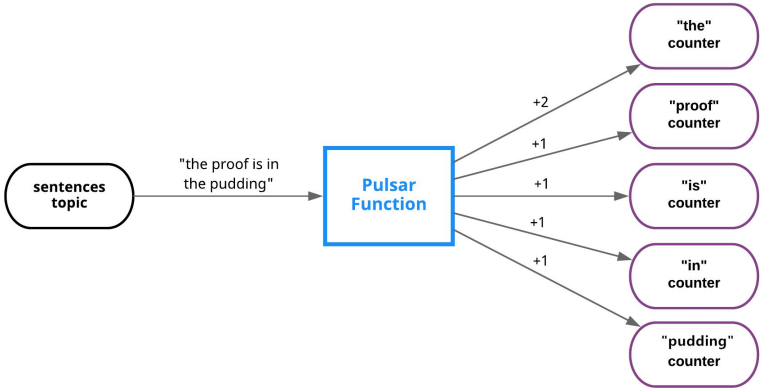
Covid Data w/ Pulsar

Pulsar Functions

Pulsar Functions







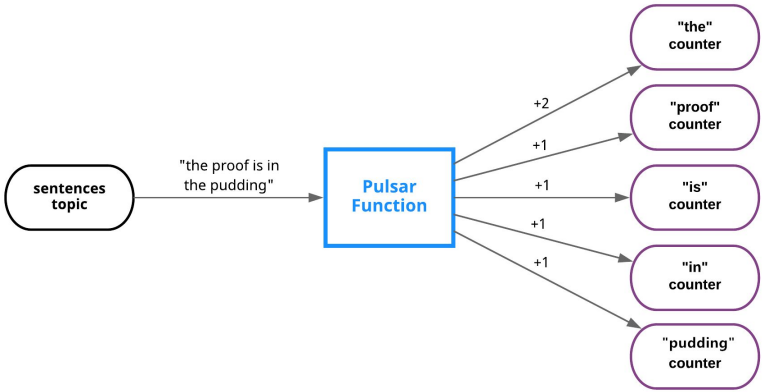
Run Serverless Lambda functions on the pulsar

For processing topics in real time for

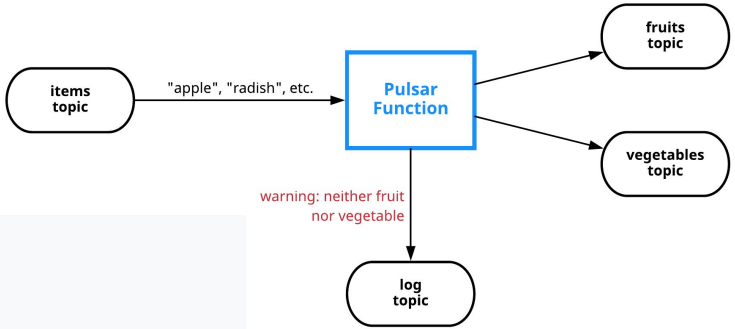
event driven workflows

Pulsar Functions

```
package org.example.functions;

import org.apache.pulsar.functions.api.Context;
import org.apache.pulsar.functions.api.Function;

import java.util.Arrays;

public class WordCountFunction implements Function<String, Void> {
    // This function is invoked every time a message
    // is published to the input topic
    @Override
    public Void process(String input, Context context) throws Exception
{
        Arrays.asList(input.split(" ")).forEach(word -> {
            String counterKey = word.toLowerCase();
            context.incrCounter(counterKey, 1);
        });
        return null;
    }
}
```



Pulsar Functions




```
from pulsar import Function

class RoutingFunction(Function):
    def __init__(self):
        self.fruits_topic = "persistent://public/default/fruits"
        self.vegetables_topic = "persistent://public/default/vegetables"

    def is_fruit(item):
        return item in [b"apple", b"orange", b"pear", b"other fruits..."]

    def is_vegetable(item):
        return item in [b"carrot", b"lettuce", b"radish", b"other vegetables..."]

    def process(self, item, context):
        if self.is_fruit(item):
            context.publish(self.fruits_topic, item)
        elif self.is_vegetable(item):
            context.publish(self.vegetables_topic, item)
        else:
            warning = "The item {0} is neither a fruit nor a vegetable".format(item)
            context.get_logger().warn(warning)
```

DS

LAB TIME!

🤓 💻 👇

dtsx.io/ess-ep3-lab

Select ONLY this one →

Pulsar & Cassandra

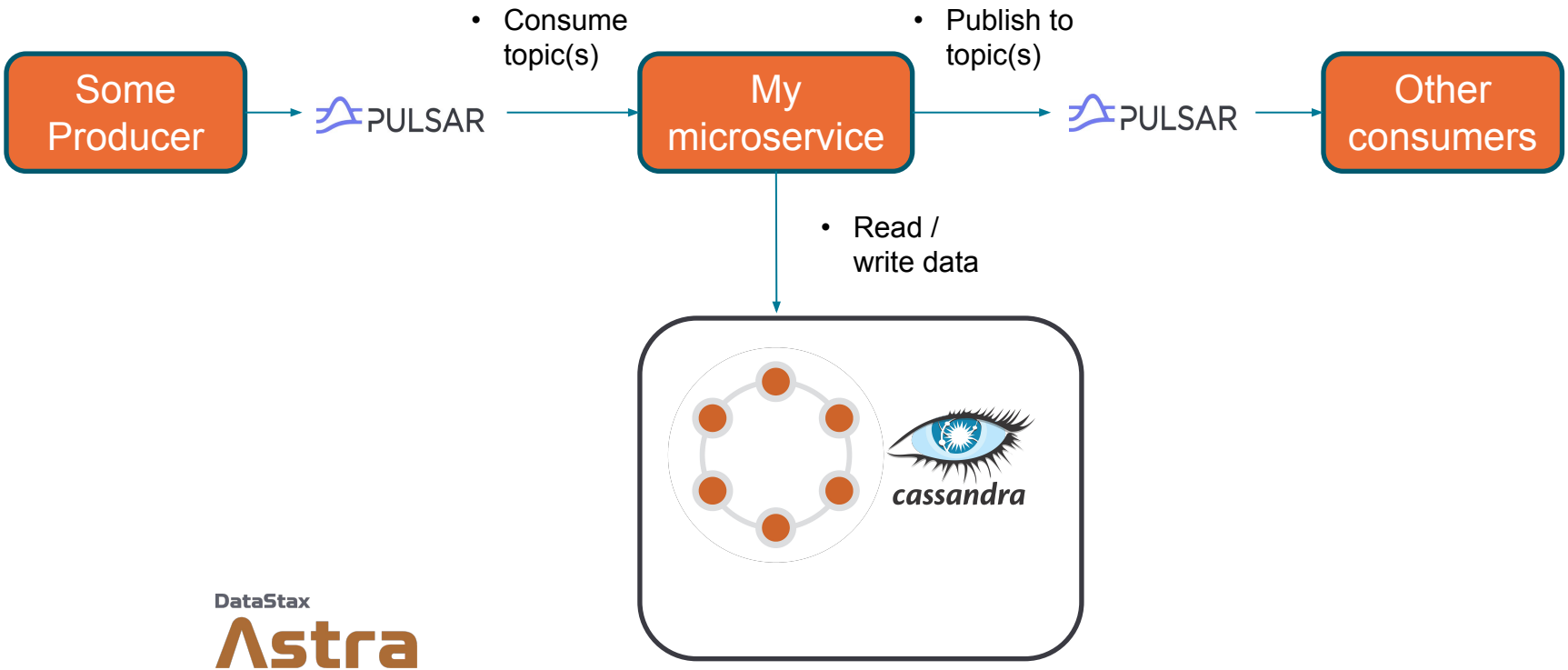
Similarities and Differences

Cassandra + Pulsar – Similarities and Distinctives

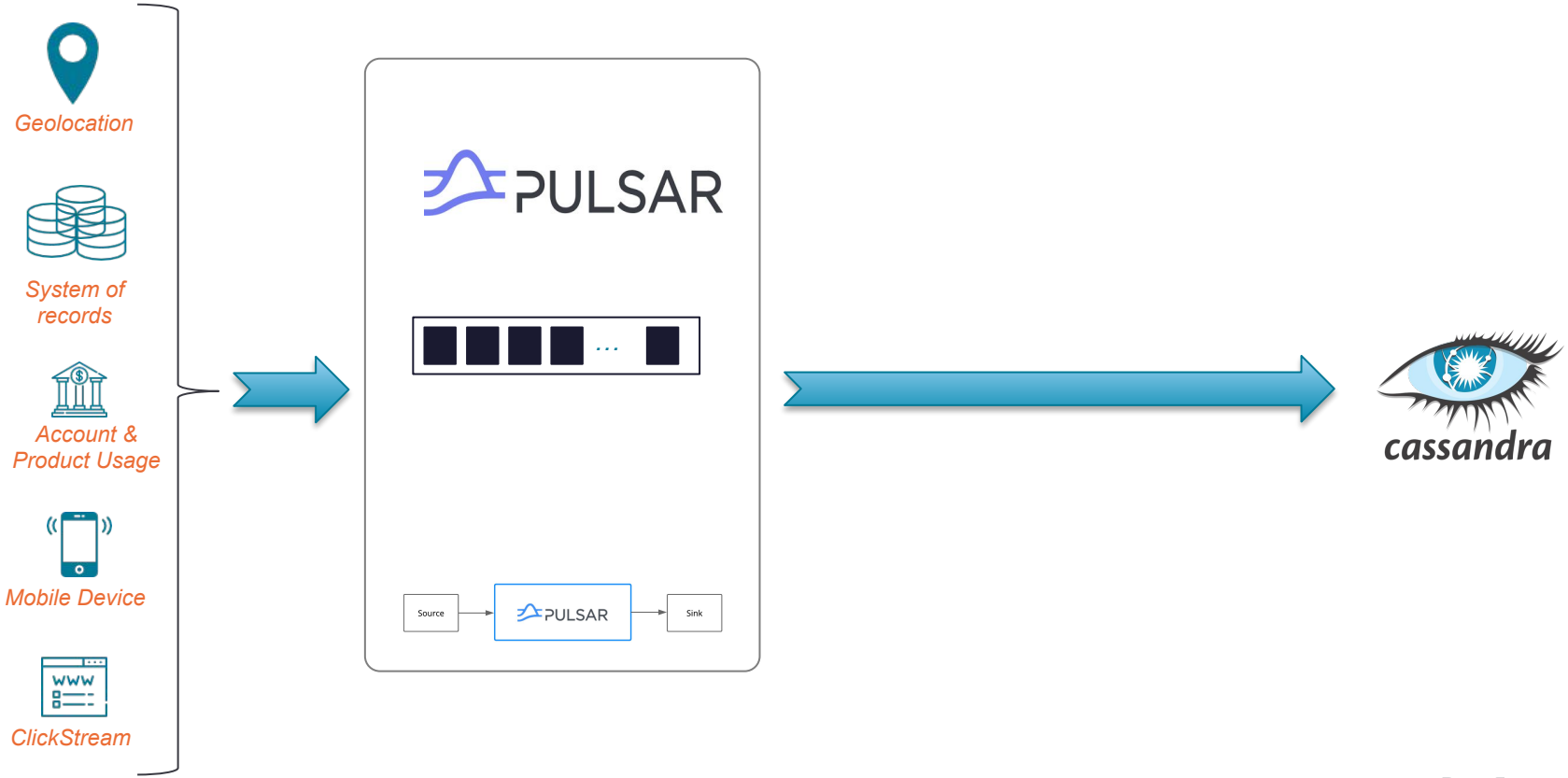
- Concepts in common
  - Distributed Systems
  - Partitioning / Hashing
  - Replication
    - Slight differences in implementation
  - Multi-DC
  - Log-structured
  - TTL / retention

- Cassandra excels at…
  - High volume, write intensive data storage workloads at scale
  - Suitable as a system of record
  - High performance searching via DSE
- Pulsar excels at…
  - Streaming data to/from services and legacy data sources
  - Acting upon changes in data from multiple sources (aka pipelines)

Pattern 1: Cassandra + Pulsar in Microservices



Pattern 2: Event Streaming



https://pulsar.apache.org/docs/en/io-overview/

Pulsar IO Connectors

- **Sources**
  - feed data from external systems into Pulsar.
- **Sinks**
  - feed data from Pulsar into external systems
- **When creating a connector, you can set the processing guarantee with the following semantics:**
  - ATLEAST_ONCE
  - ATMOST_ONCE
  - EFFECTIVELY_ONCE

Why a Pulsar Connector with Cassandra ?

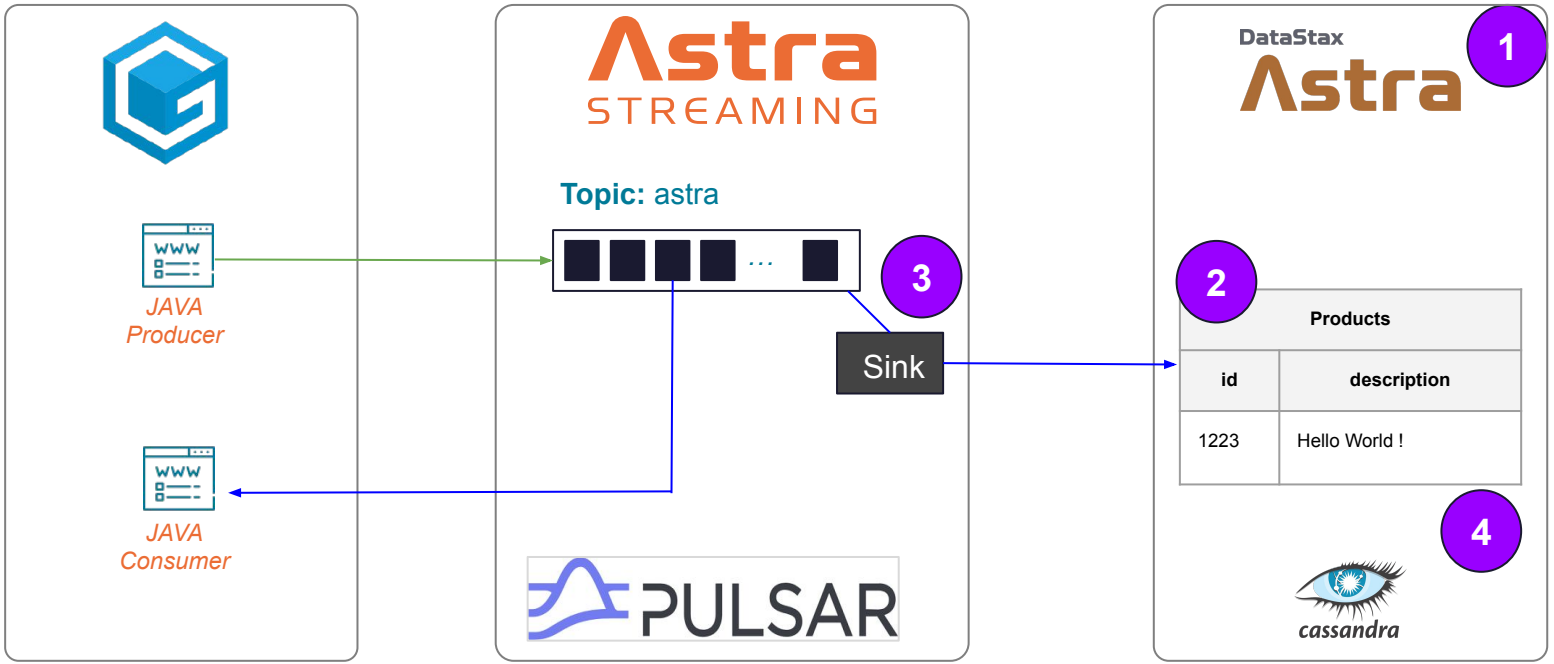
Spark Streaming = PULL

- Computation framework
- Enable advanced transformations
- Mode Pull with a dedicated runtime (poll)

Pulsar IO Connector = PUSH

- No extra runtime to consume
- Simple Mappings

Next Lab


Astra
STREAMING
Topic: astra
JAVA
Producer
JAVA
Consumer
Sink
PULSAR
DataStax
Astra
1
2
Products
id
description
1223
Hello World !
3
4
cassandra

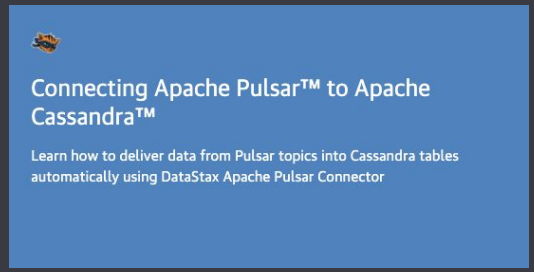
DS

LAB TIME!

🤓 💻 👇

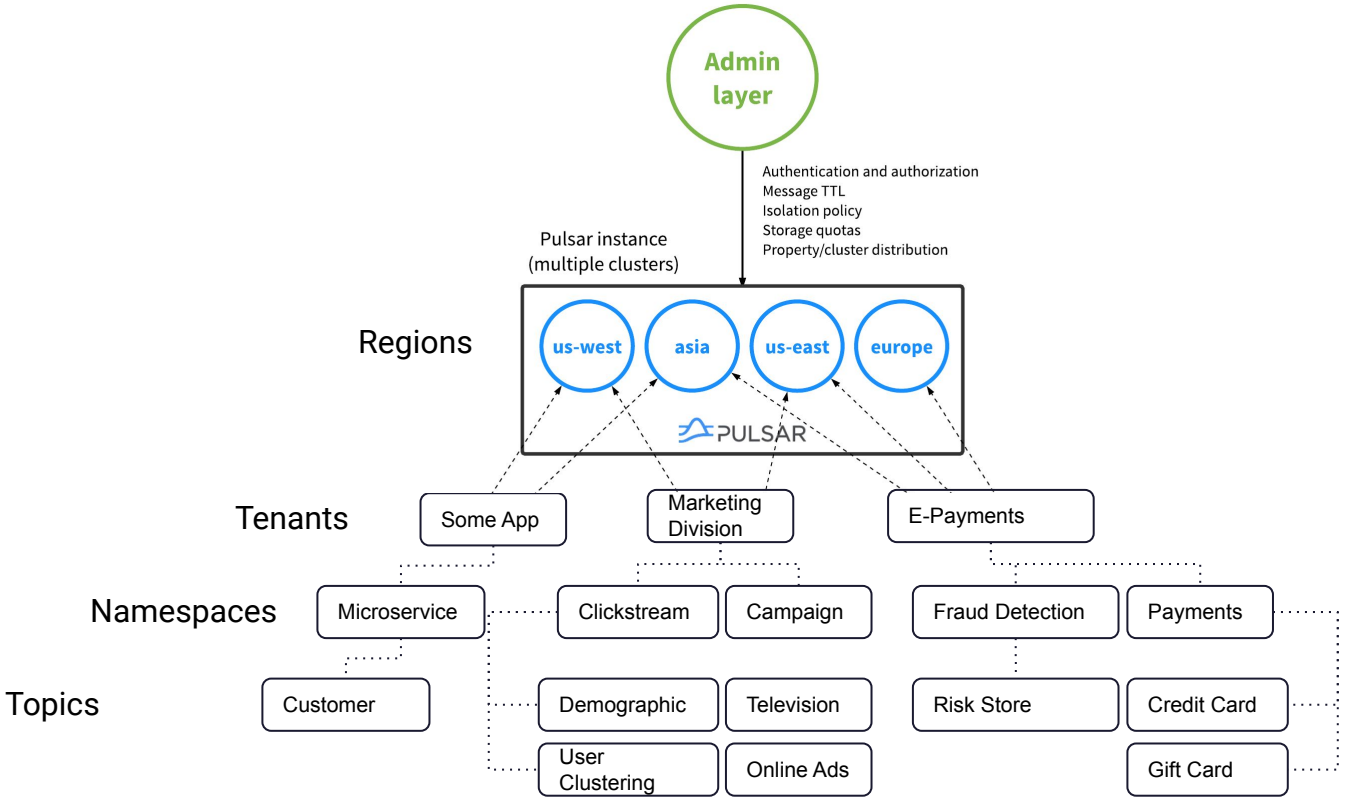
dtsx.io/ess-ep3-lab

Select ONLY this one →



Multi-Tenancy in Pulsar

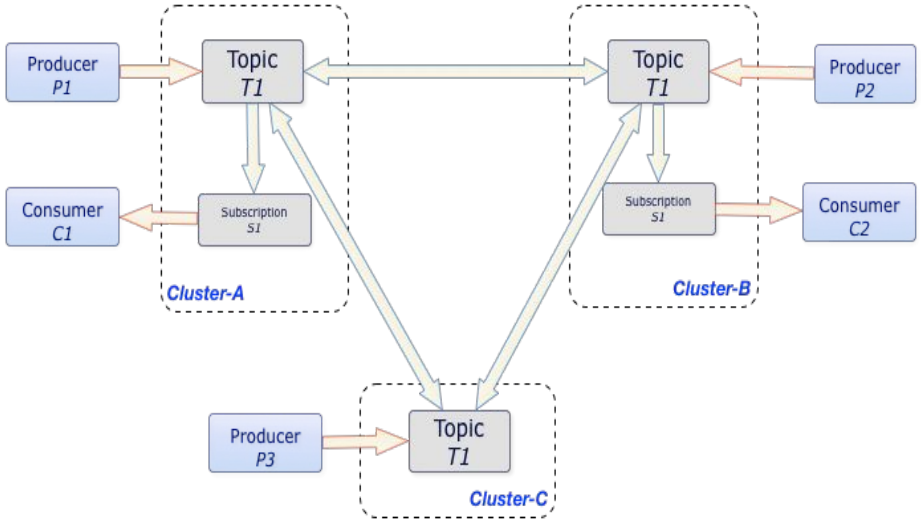
Multi-Tenancy



Geo-replication in Depth

Geo-Replication

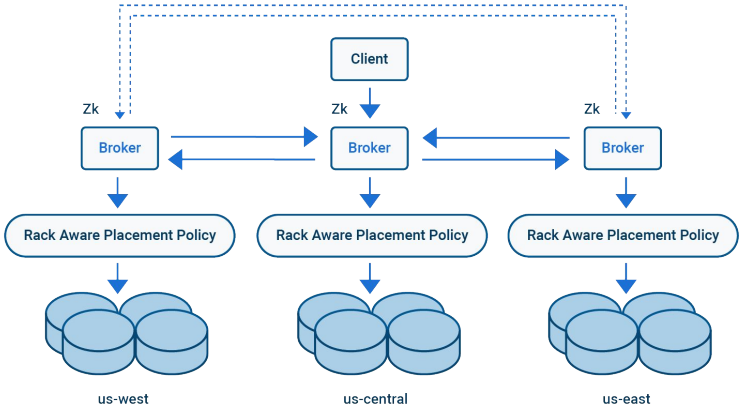
- Per-tenant basis
  - Only when a tenant has access to both clusters
- Enabled at namespace-level
  - Assign clusters to namespace
- Replicated Subscription
  - Keep subscription state in sync
  - Consumer failover to another cluster and resume from the failure point
  - Sub-second delay and possible duplicates

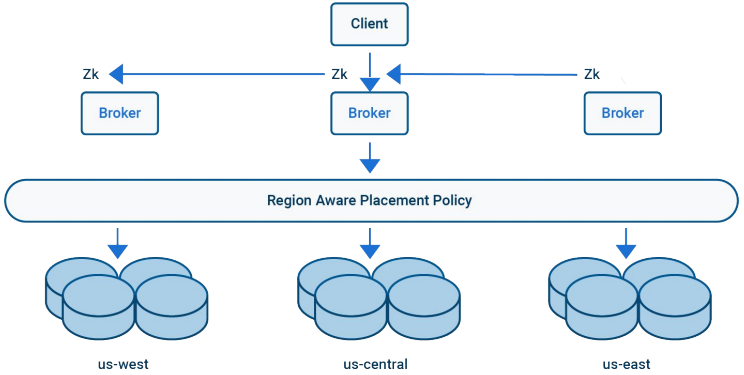


Geo-Replication, cont'd

- Geo-replication
  - Local acknowledgement
- Synchronous Geo-replication
  - Global acknowledgement
  - Rack-awareness Policy
- Selective Geo-replication per message

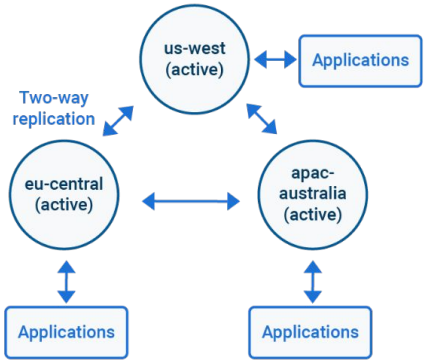
```
List<String> restrictDatacenters =
Lists.newArrayList("apac-australia");

Message message = MessageBuilder.create()
        …
        .setReplicationClusters(restrictDatacenters)
        .build();

producer.send(message);
```

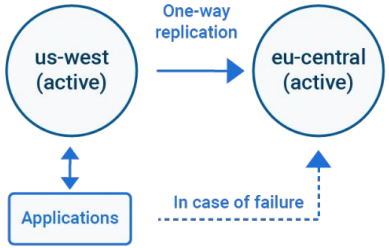


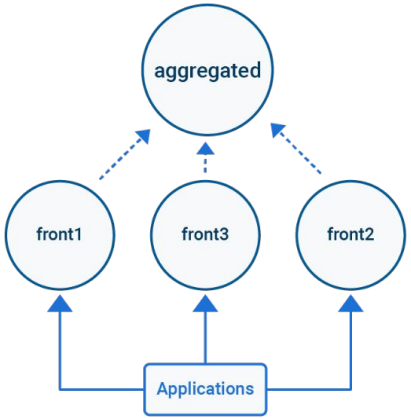


Geo-Replication, cont'd

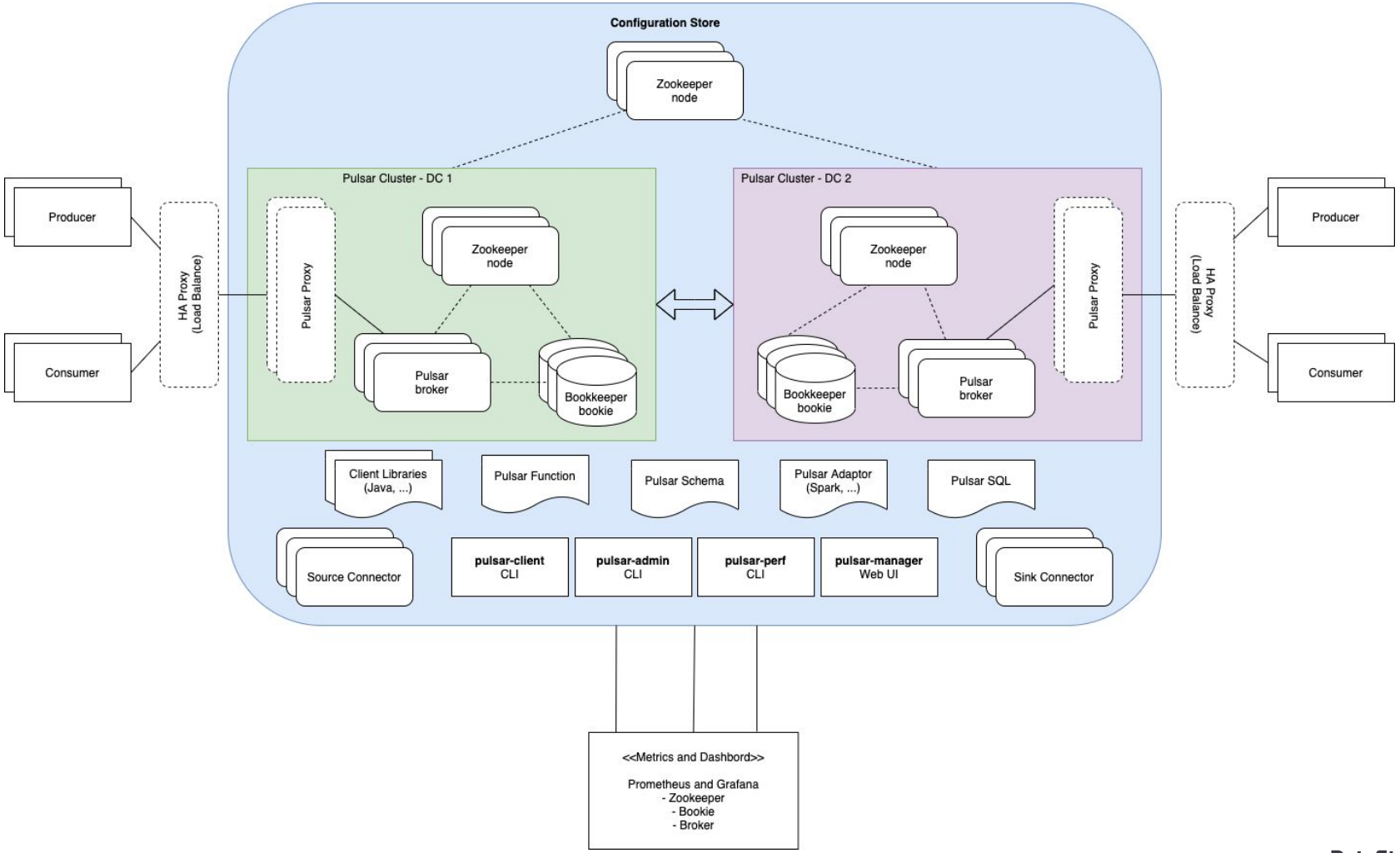
- Asynchronous Geo-replication pattern
  - Active-active
  - Active-standby
  - Aggregated replication
    - Edge computing
- Monitor replication statistics
  - Replication backlog
- Throttle replication







Deploying and Operating

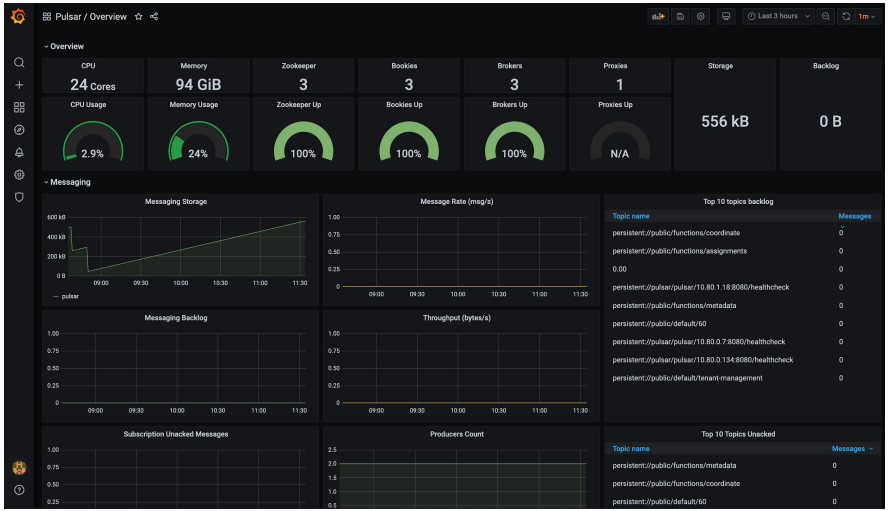

Configuration Store
Zookeeper node
Pulsar Cluster - DC 1
Pulsar Cluster - DC 2
Producer
Consumer
HA Proxy (Load Balance)
Pulsar Proxy
Zookeeper node
Pulsar broker
Bookkeeper bookie
Client Libraries (Java, ...)
Pulsar Function
Pulsar Schema
Pulsar Adaptor (Spark, ...)
Pulsar SQL
Source Connector
pulsar-client CLI
pulsar-admin CLI
pulsar-perf CLI
pulsar-manager Web UI
Sink Connector
<<Metrics and Dashbord>>
Prometheus and Grafana
- Zookeeper
- Bookie
- Broker

Pulsar Features, cont'd

- Security
  - Pluggable Authentication
    - Certificate based authentication
    - JWT token based authentication
    - Authenz
    - Kerberos
    - OAuth 2.0
  - Pluggable Authorization
    - Role based ("role token")
    - Authenz
    - Internal
  - TLS encryption
    - Traffic encryption
      - Between clients (producers/consumers) and Pulsar
      - Among Pulsar components
    - Message encryption (outside Pulsar)
      - End-to-end encryption
      - Volume based encryption

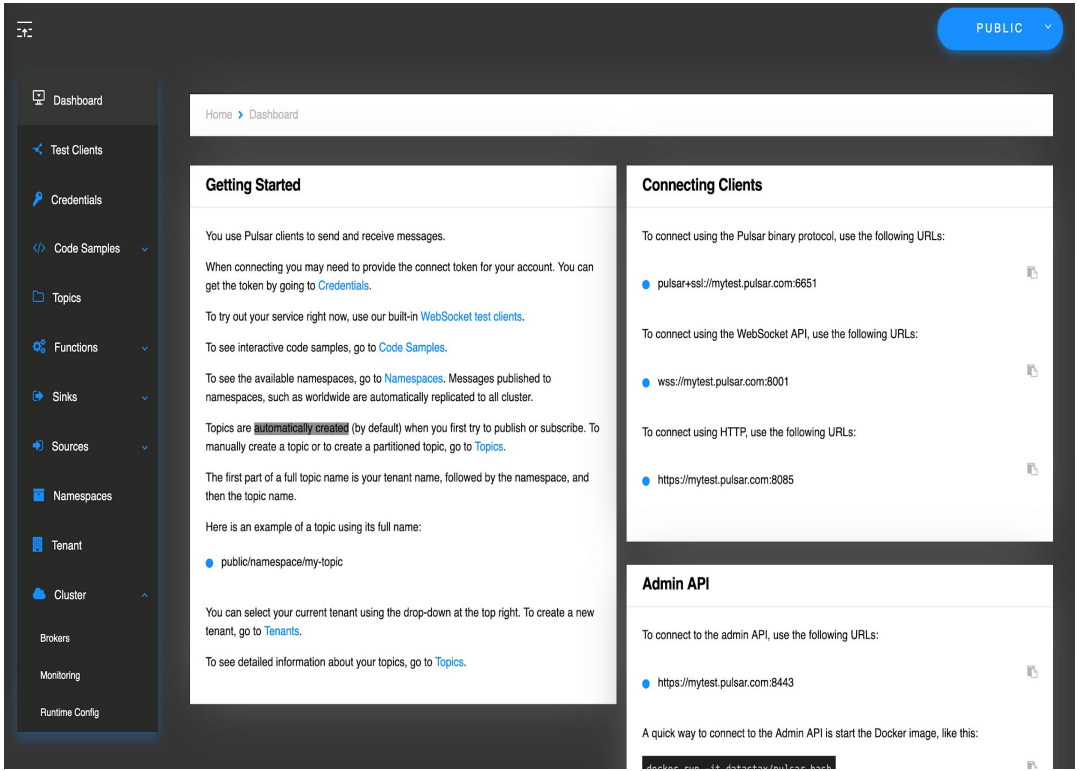
Pulsar Features, cont'd

- Metrics Monitoring
  - Expose metrics in **prometheus** format
    - "/metrics" at different ports (e.g. 8080 for broker, 8000 for zookeeper, etc.)
  - Pulsar Helm chart (for K8s) has embedded Prometheus and Grafana components
    - Built-in dashboards
  - Prometheus Alert-manager



Pulsar Features, cont'd

- Pulsar Admin Console (DataStax)
- Pulsar Manager
  - Web based Pulsar cluster management tool
  - Able to manage multiple environments
- Admin CLI tools
  - Pulsar-admin
  - Pulsar-client
  - Pulsar-perf
  - … …



DEMO TIME!

🤓 💻 👇

Astra Streaming Helm Charts

QUIZ TIME!

DataStax

SUBSCRIBE NOW: dtsx.io/cassandra-cert-2206

TOMORROW:

June 22nd

Cassandra Certification Workshop

Led By David and David

DataStax

SUBSCRIBE NOW: dtsx.io/spring-2306

THIS WEEK:

23rd / 24th of June

Build Spring microservices with a NoSQL DB

Led By Cedrick

DataStax

Cassandra Day India

Mark the date: July the 10th

One day, two tracks, over 12 hours of pure data knowledge with the best NoSQL experts and guest speakers. Boost your career and reach the new level with DataStax Developers!

DataStax

Cassandra Day India

Call for Papers is open: dtsx.io/cdindia-cfp

THANK YOU!