
AWS
re:Invent

OPN305

Building real-time applications using Apache Flink

Dr. Steffen Hausmann

Principal Specialist SA, Analytics

AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Apache Flink

STATEFUL COMPUTATIONS OVER DATA STREAMS



Diverse use cases

- Event-driven applications
- Streaming analytics and ETL
- Batch analytics



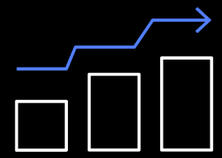
Processing guarantees

- Exactly once state consistency
- Event-time processing
- Late data handling



Expressive APIs

- SQL
- Table and DataStream API
- Complex event processing



Scale-out architecture

- Adapt to desired throughput
- Support for TBs of state



Strong community

- Vibrant open-source community
- Broad set of connectors

Running Apache Flink on AWS



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Running Apache Flink on AWS



YARN



Apache Mesos

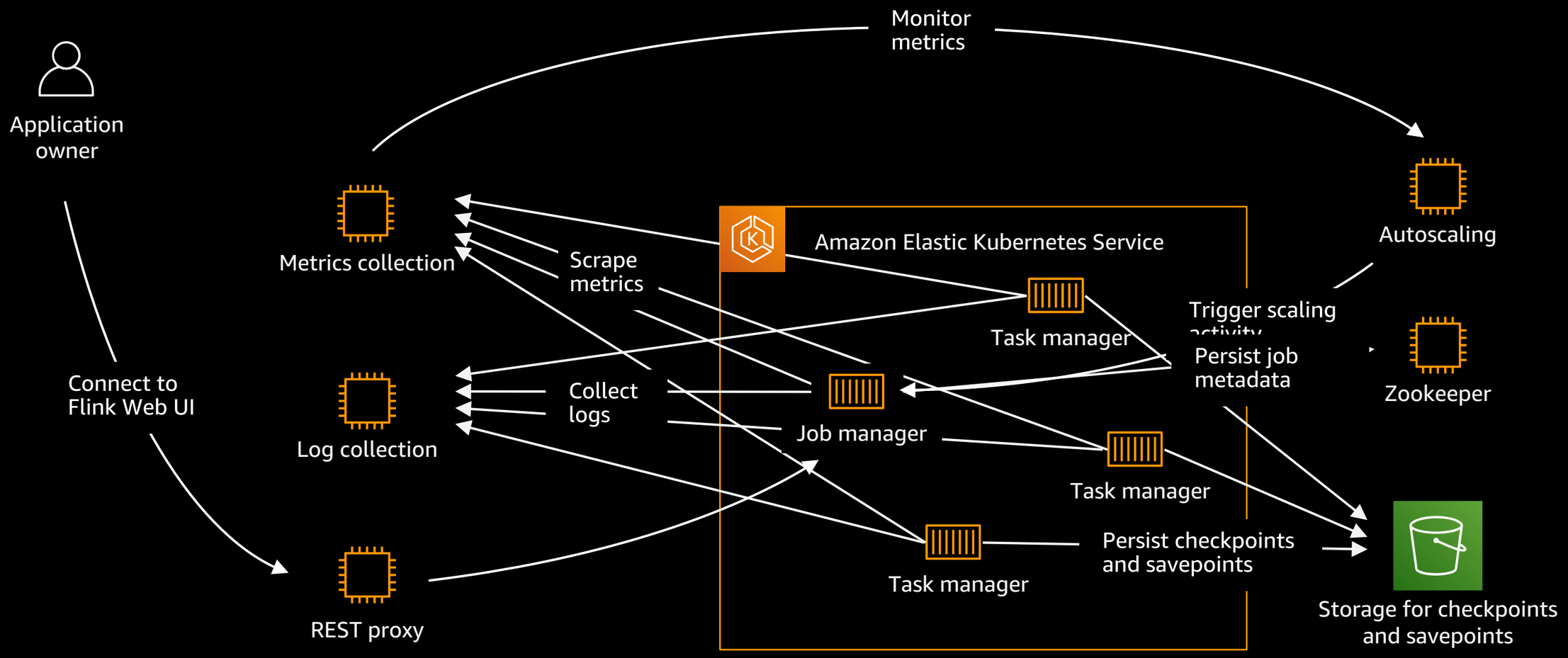


Kubernetes



Amazon Kinesis
Data Analytics

Running Apache Flink on Kubernetes

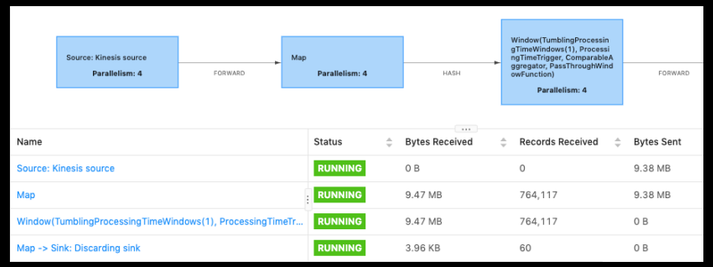


Amazon Kinesis Data Analytics under the hood

Application owner

Amazon CloudWatch

Amazon CloudWatch Logs



Flink Web UI

Amazon Kinesis Data Analytics

Amazon Elastic Kubernetes Service

Task manager

Job manager

Task manager

Task manager

Autoscaling

Zookeeper

Storage for checkpoints and savepoints

When to choose what?

Kinesis Data Analytics
Easy and powerful stream processing

- No management of infrastructure
- Monitoring and high availability built in
- Automatic scaling capabilities

Self-managed cluster
Flexibility and choice for your needs

- Full flexibility on instance types
- Optimize cost with reserved instances, savings plan, and spot instances
- Fine-grained control over cluster settings, debugging tools, and more

Building low latency applications with Apache Flink

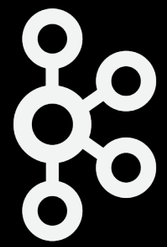


AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Streaming Architectures with Apache Flink













Ingestion



Processing











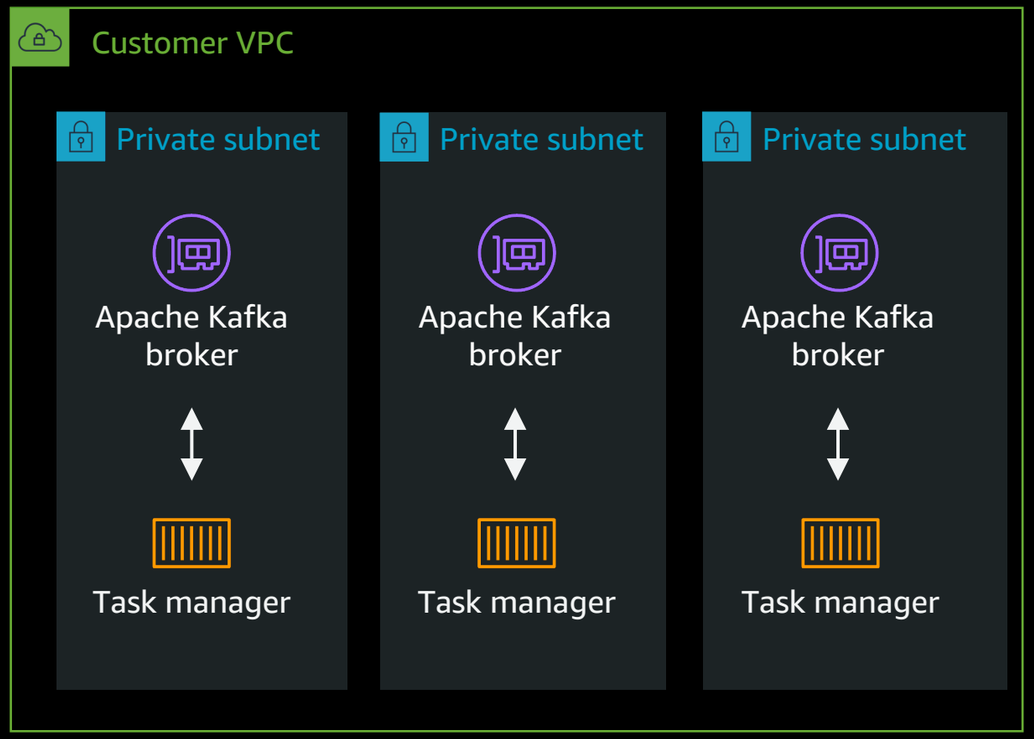




Consumption

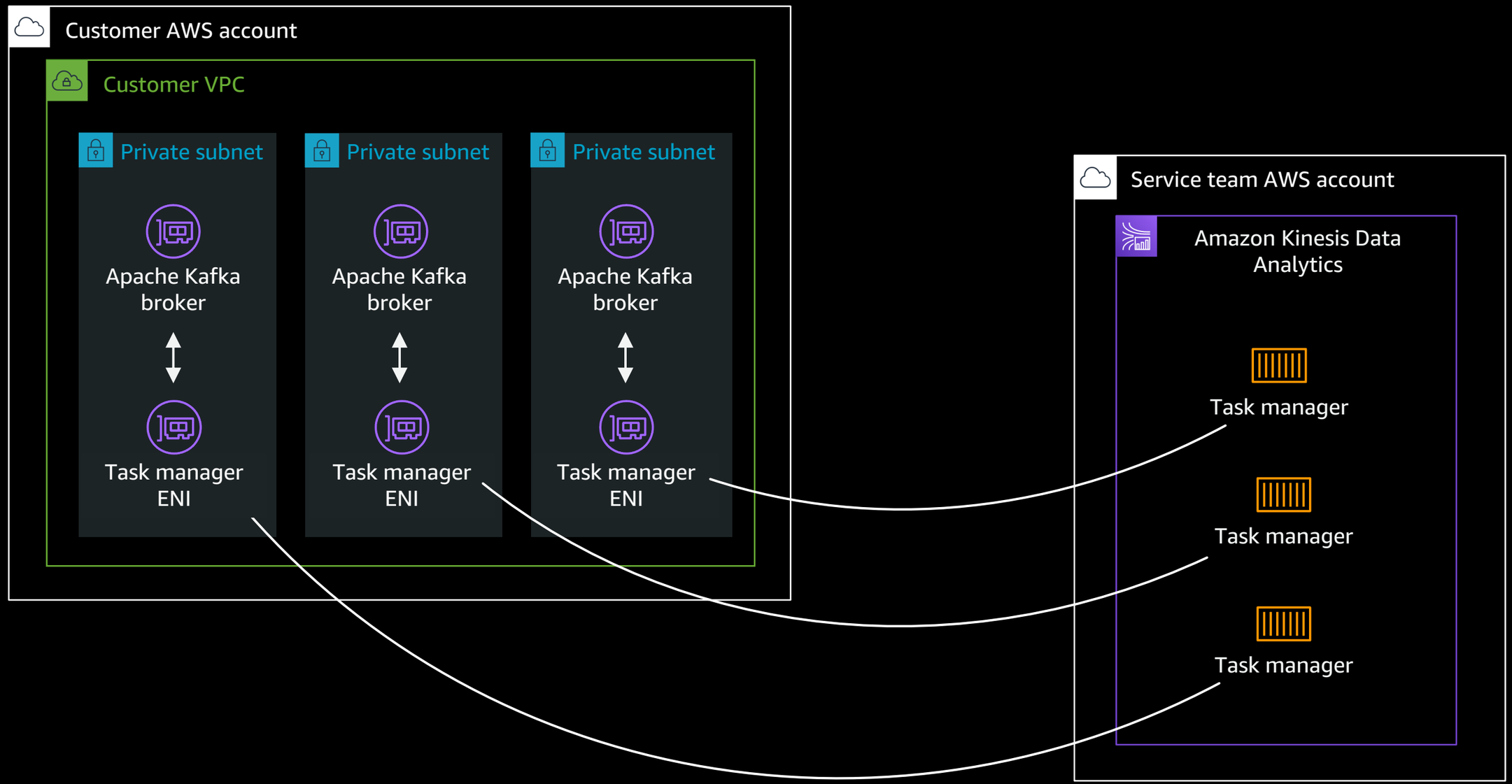
Accessing Apache Kafka clusters

ESTABLISHING NETWORK CONNECTIVITY



Accessing Apache Kafka clusters

ESTABLISHING NETWORK CONNECTIVITY



Authentication options for Amazon MSK

Client authentication (TSL)

- A Certificate Authority signs certificates of individual consumers
- Clients authenticate with signed certificates certificates

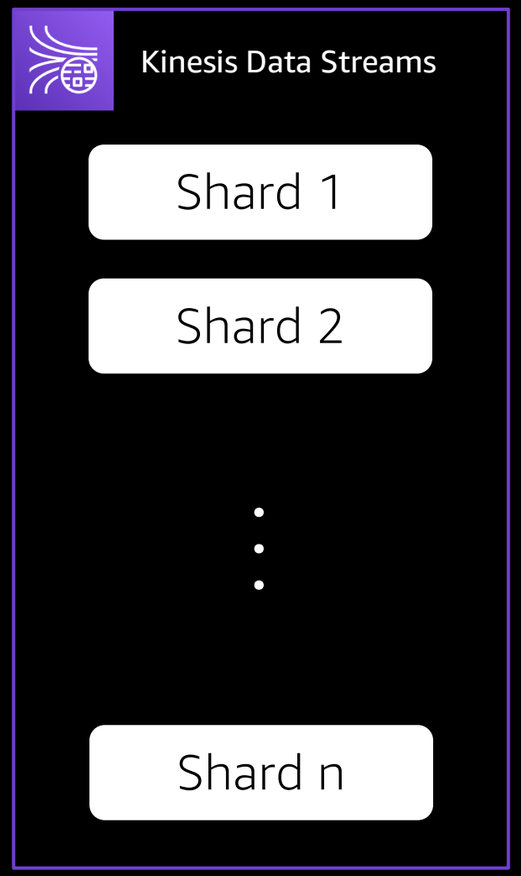
SASL/ SCRAM authentication

- Username/password are stored in AWS Secrets Manager and imported into the Kafka cluster
- Clients can authenticate with username/password

In the Flink application

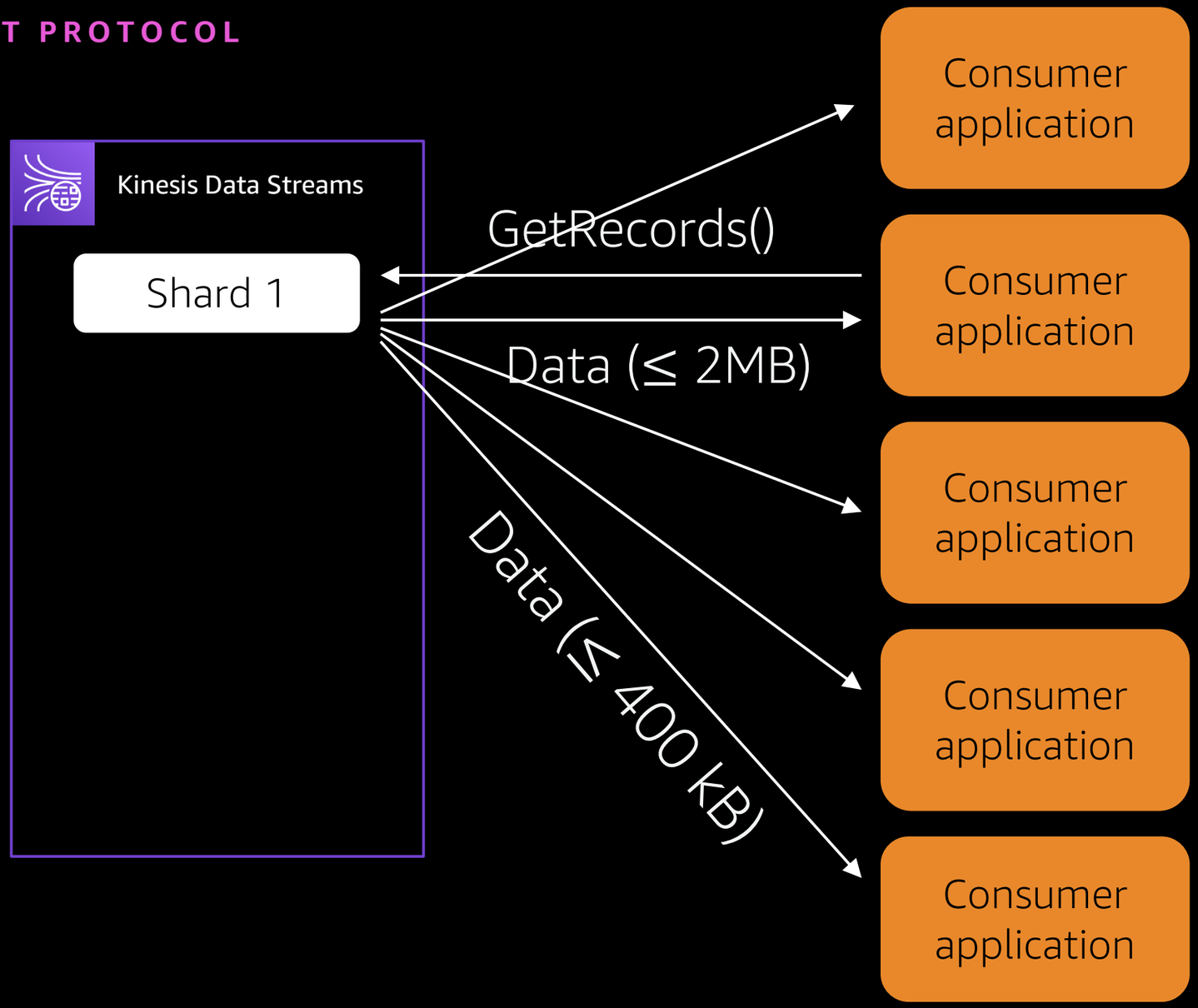
- Configure custom trust store for mutual authentication
- Do NOT bake credentials into your application code

Consuming from Amazon Kinesis Data Streams



Consuming from Amazon Kinesis Data Streams

DEFAULT PROTOCOL

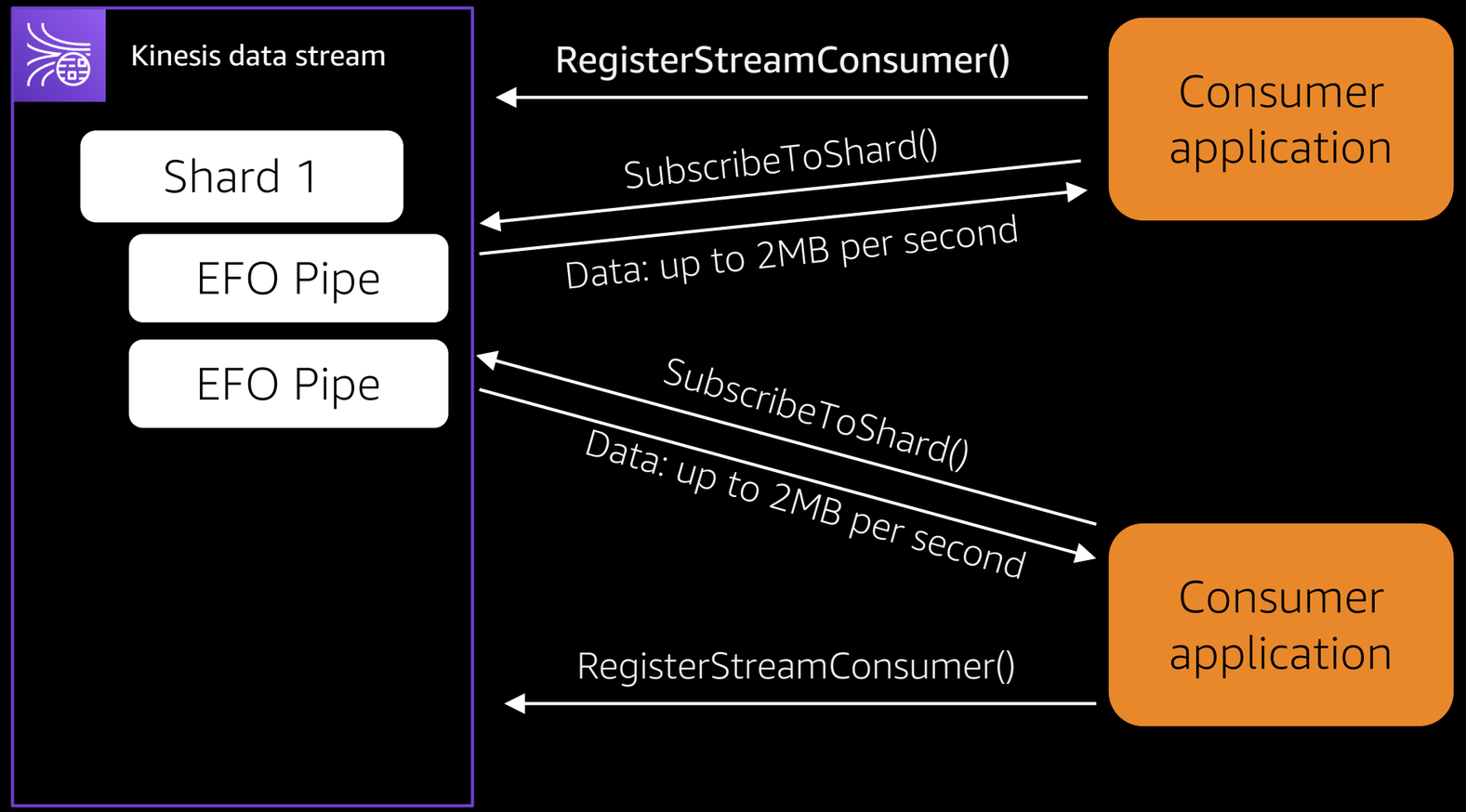


With only one consumer application,
records can be retrieved every 200 ms

With more consumer applications,
propagation delay increases and
throughput per consumer decreases

Consuming from Amazon Kinesis Data Streams

ENHANCED FAN OUT (EFO)



Each consumer application gets dedicated 2MB per second egress, per shard

Amazon Kinesis Data Streams best practices

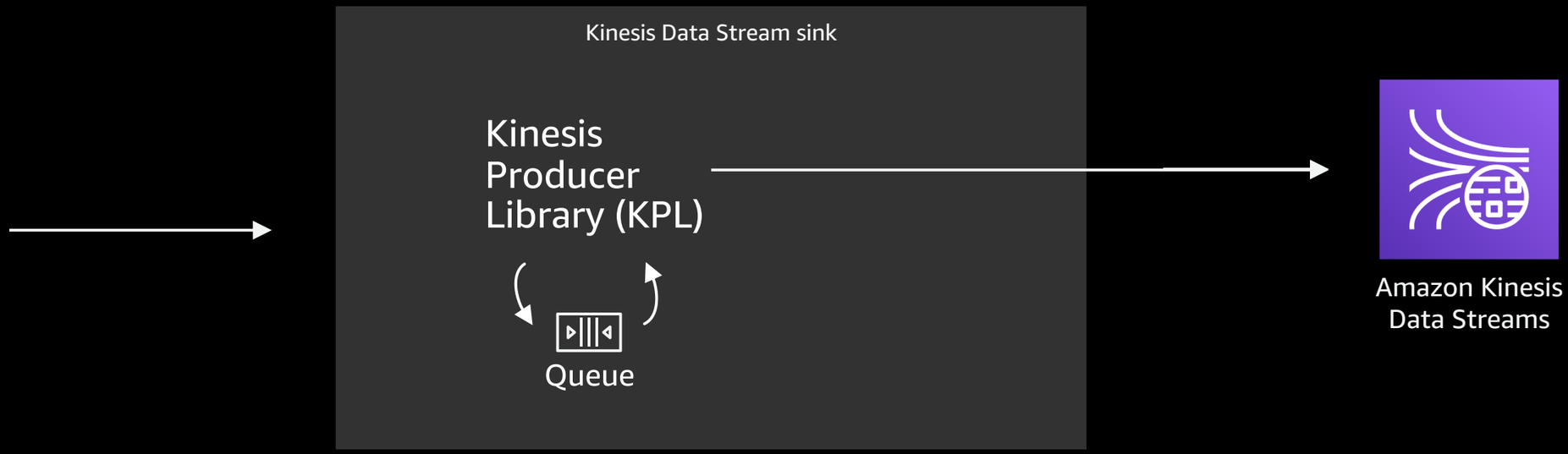
Fine tune GetRecords TPS and throughput for the classic protocol

- `SHARD_GETRECORDS_INTERVAL_MILLIS`
- `SHARD_USE_ADAPTIVE_READS`
- `DEFAULT_SHARD_GETRECORDS_MAX`

Consider enhanced fan-out for use cases with

- more than 2 consumer applications
- Low latency requirements

Producing into Amazon Kinesis Data Streams



Enable backpressure of the Kinesis sink

- `sink.setQueueLimit(number of shards * 100 kB / record size);`

Best practices for building Apache Flink applications

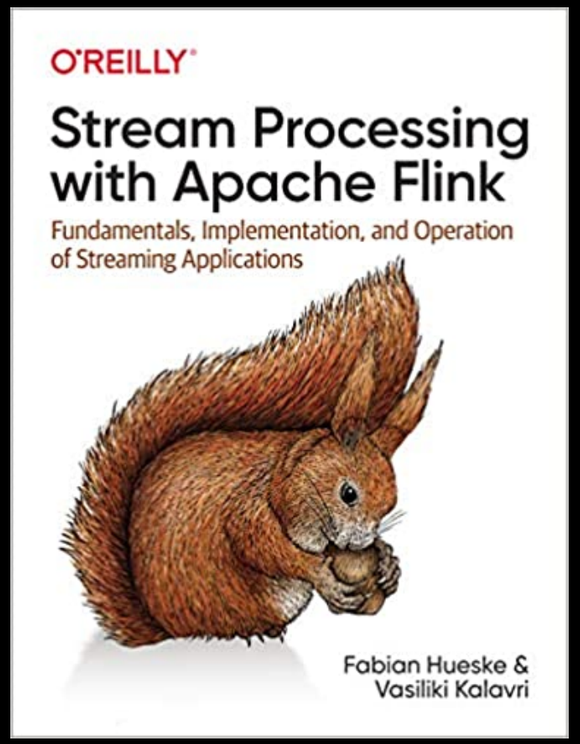


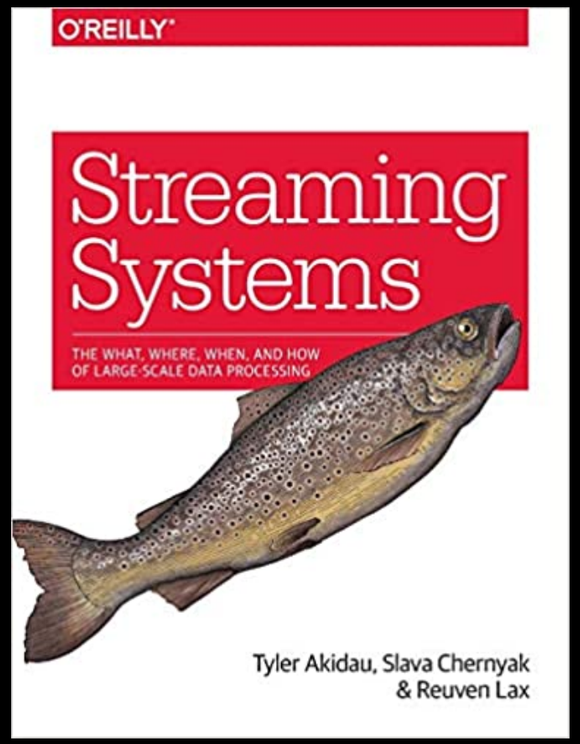
AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

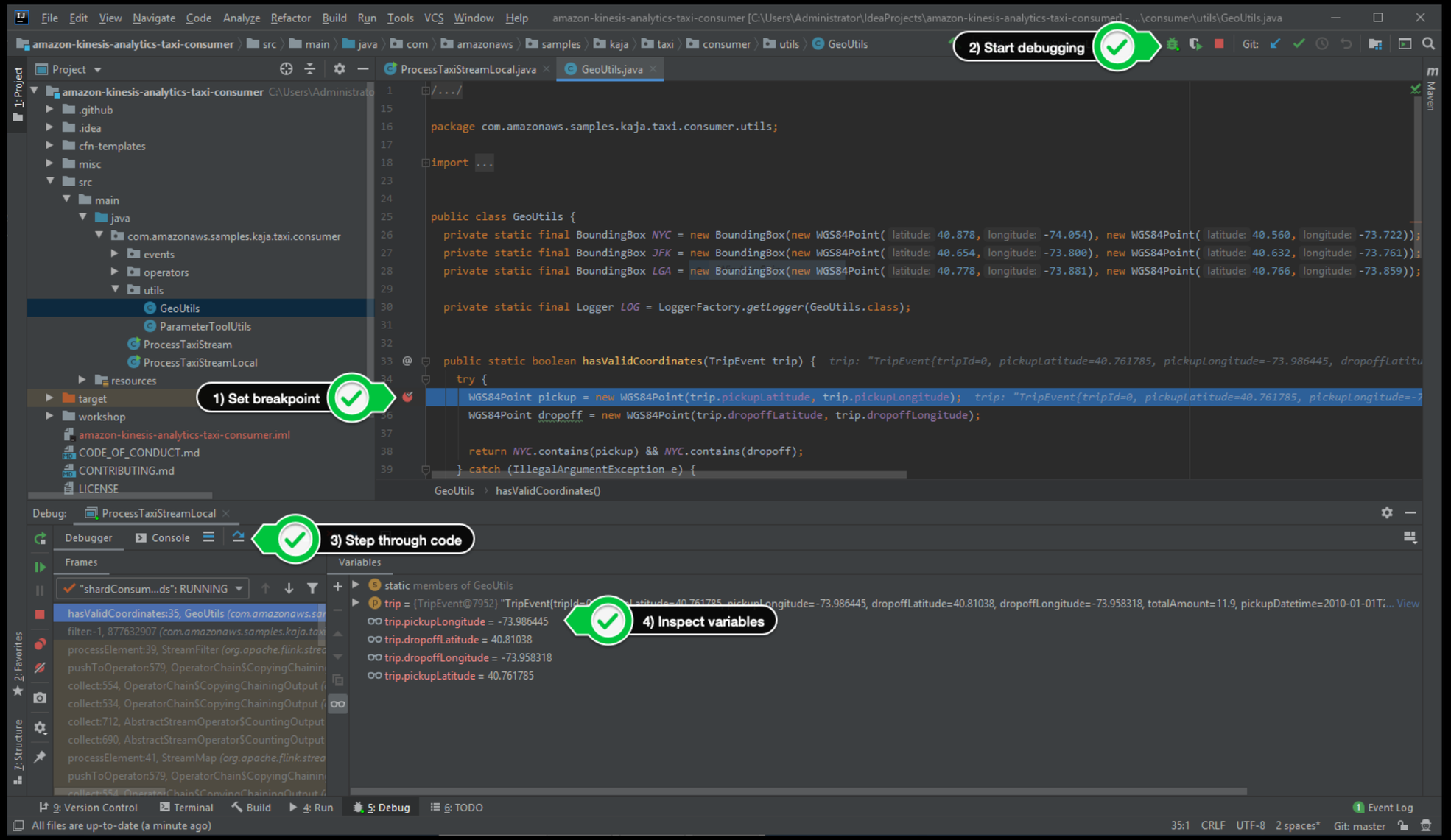
Keep it simple

Flink has very powerful high-level APIs, use them to your advantage



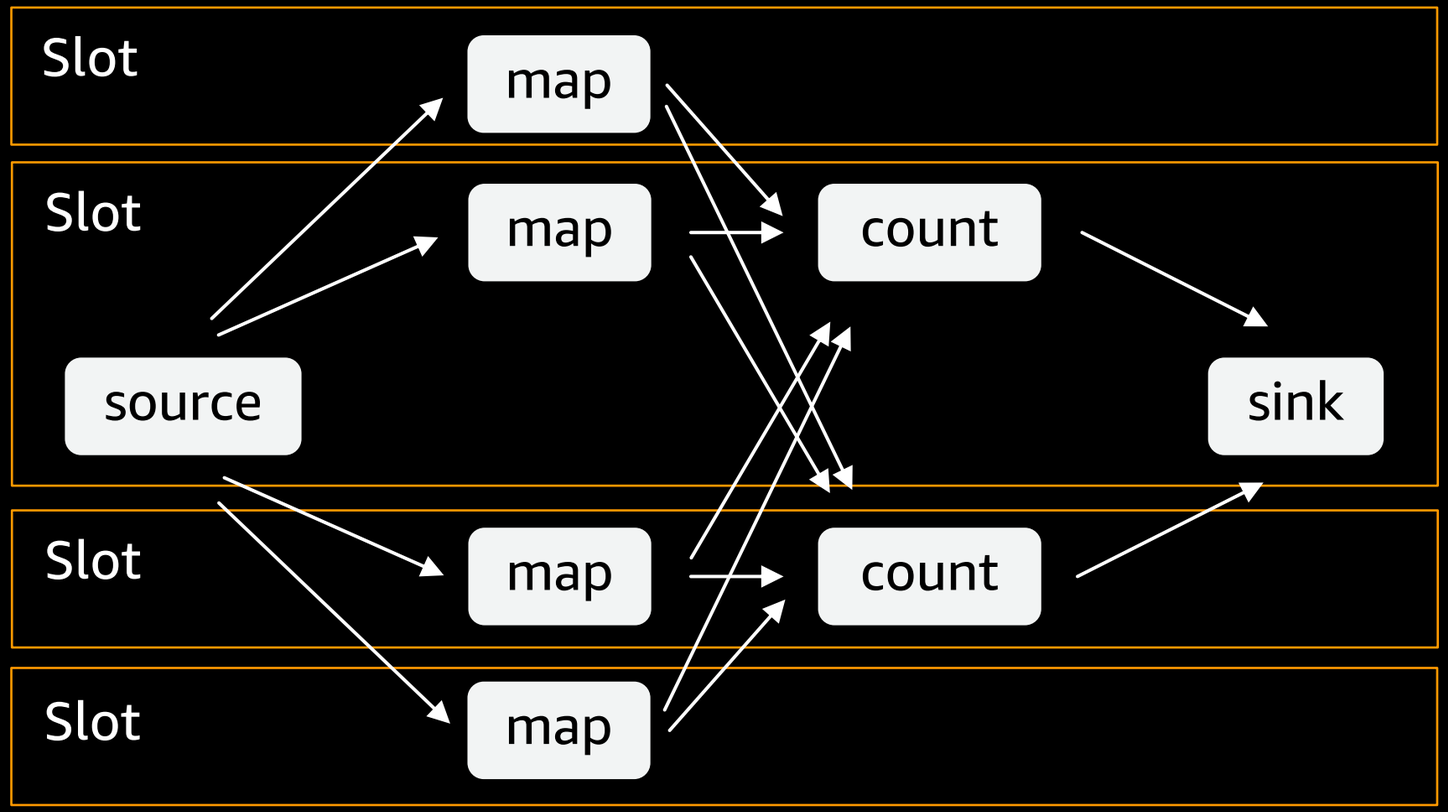


Develop locally, deploy remotely



Structure of a Flink application

A STREAMING WORD COUNT EXAMPLE



Scaling operators with slow IO

KDA scales in terms of Kinesis Processing Units (KPUs)

- 1 vCPU, 4 GB mem, 50 GB disk

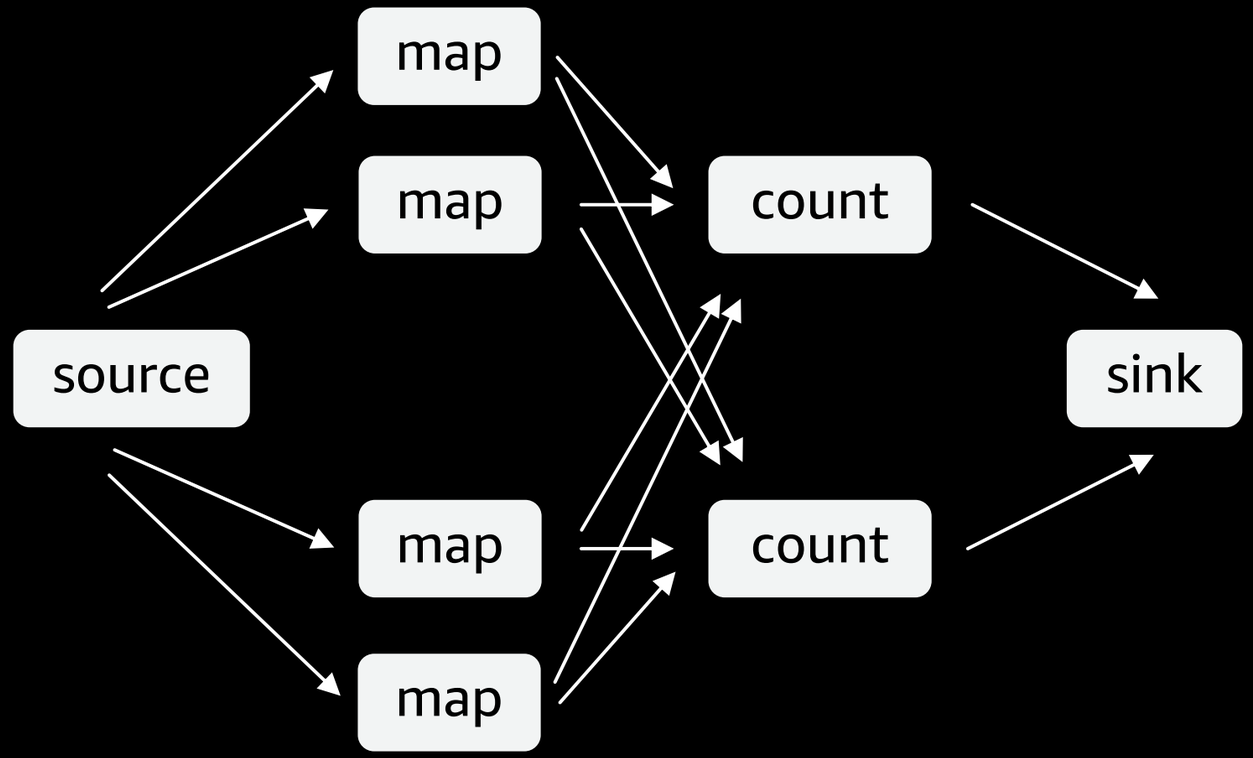


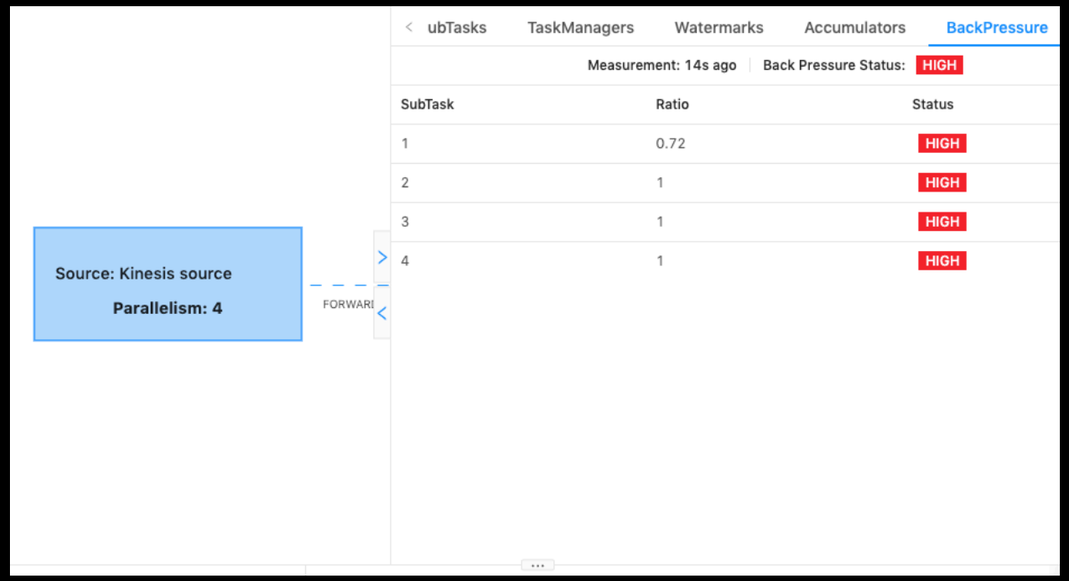
1 parallelismPerKPU



2 parallelismPerKPU

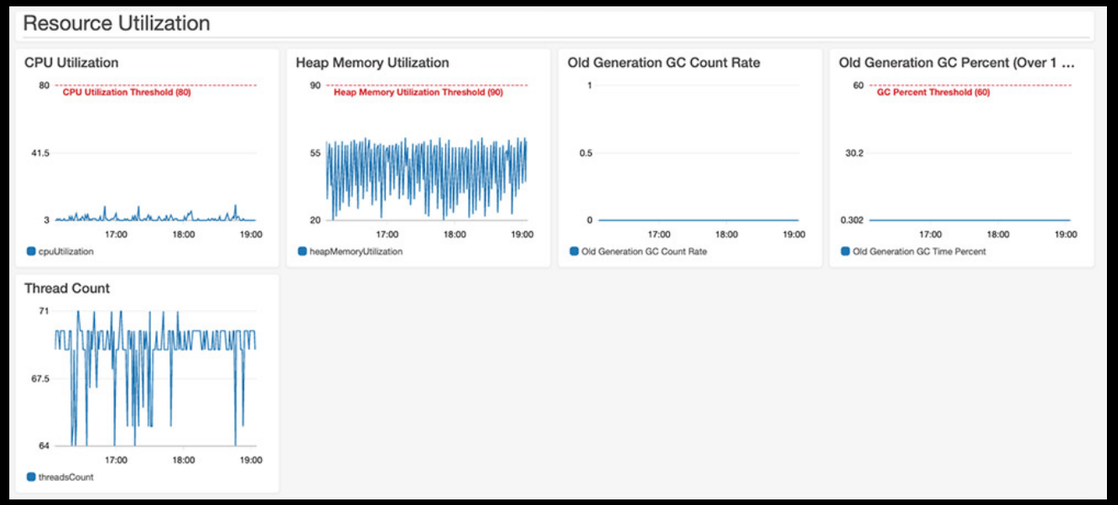
Identifying slow operators in the Flink web UI



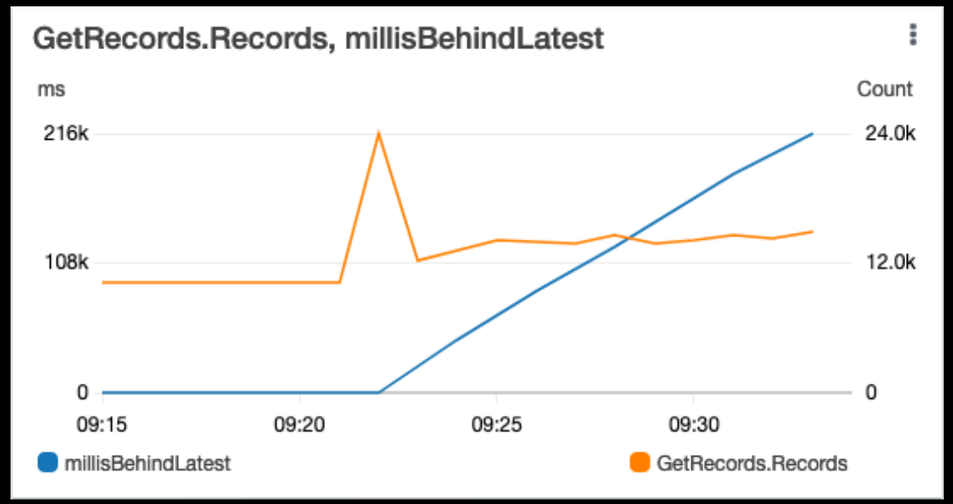


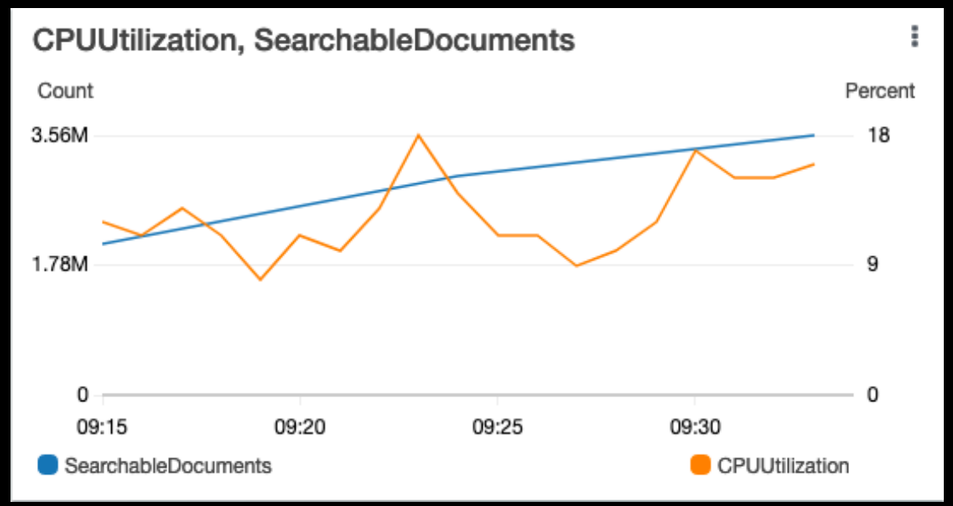
Monitoring Flink applications

Monitor the Flink application



Monitor the entire architecture





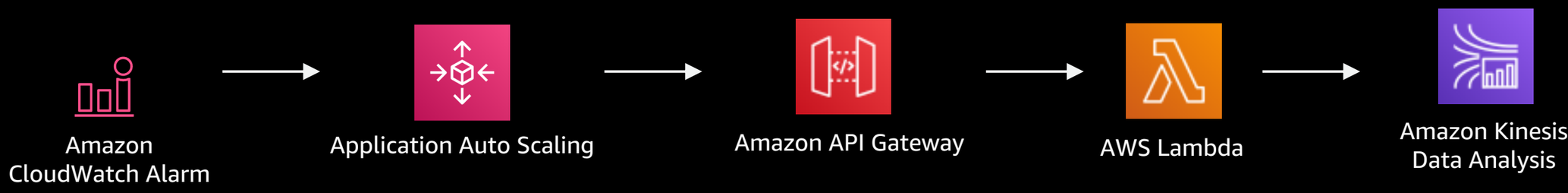
Scaling Flink applications with KDA

Native autoscaling

- Scale out when CPU usage remains above 75% for 15 minutes
- Scale in over a few hours after a drop in CPU usage

Custom autoscaling

- For non CPU bound use cases, scale based on your own alarms



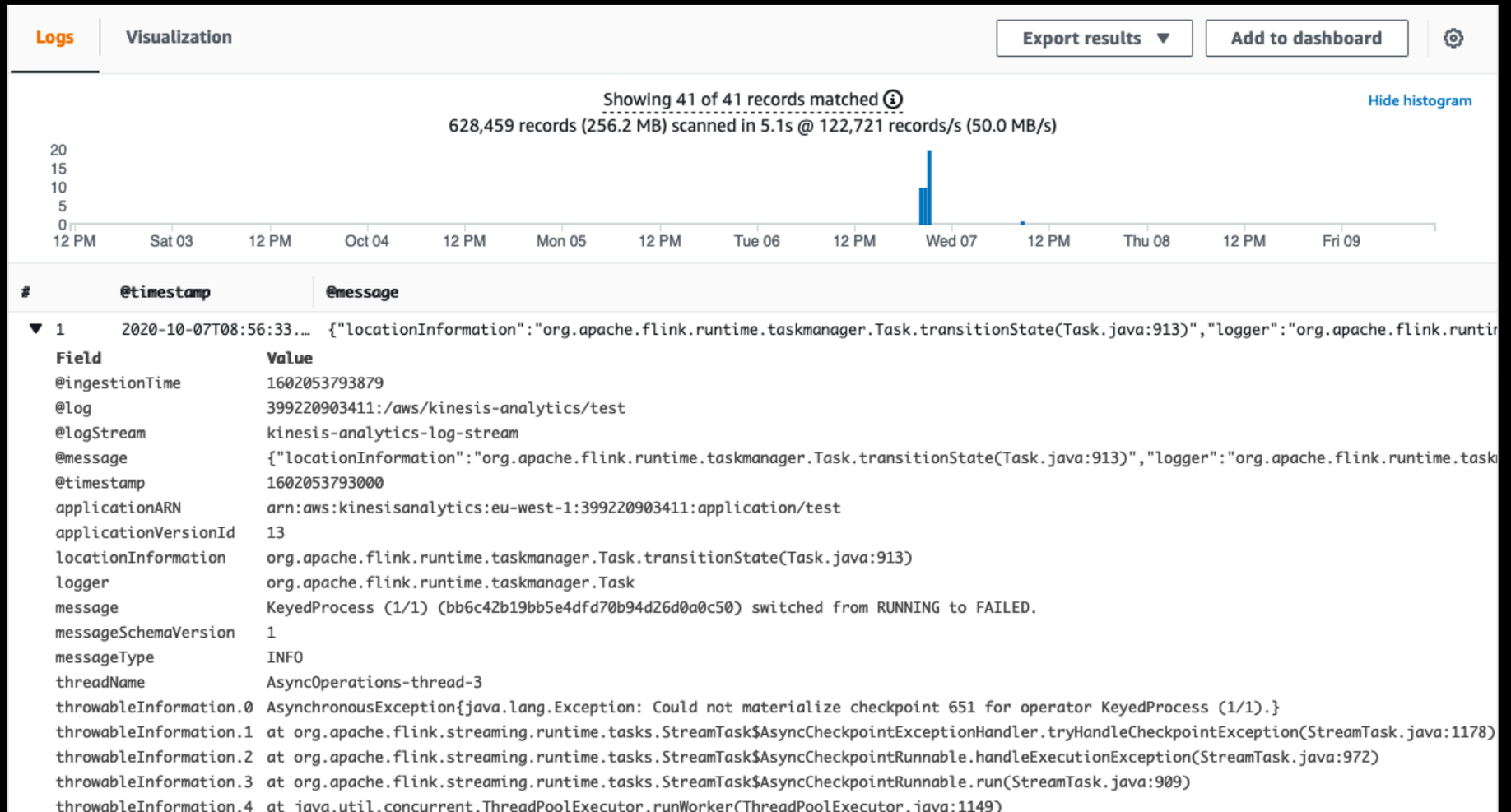
Inspecting logs with CloudWatch logs insights



Inspecting logs with CloudWatch logs insights



Inspecting logs with CloudWatch logs insights



Open-source contributions

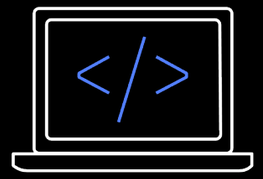


© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Open-source contributions



Team is dedicated to contribute back



EFO source and Kinesis Flink SQL Connector



Become an active part of the community



Increase visibility of the open source project



Supporting community events

References

Deployment to Kubernetes

- ci.apache.org/projects/flink/flink-docs-stable/ops/deployment/kubernetes.html

Accessing Apache Kafka clusters

- amzn.to/2SBhKn1

Authentication options for Amazon MSK

- amzn.to/3deXoty

Keep it simple

- amzn.to/30PvaQP
- amzn.to/3lwJg1k

Identifying slow operators in the Flink Web UI

- ci.apache.org/projects/flink/flink-docs-stable/monitoring/back_pressure.html

Scaling Flink applications with KDA

- amzn.to/36N4B2t

Monitoring Flink applications

- amzn.to/36PNgpU

Inspecting logs with CloudWatch Logs Insights

- amzn.to/2SFVt7D

Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.