Disclaimer: These slides are copyrighted and strictly for personal use only

- This document is reserved for people enrolled into the Amazon MSK Course by Stephane Maarek
- **Please do not share this document**, it is intended for personal use, thank you.
- If you've obtained these slides for free on a website that is not the course's website, please reach out to piracy@datacumulus.com. Thanks!

Amazon MSK

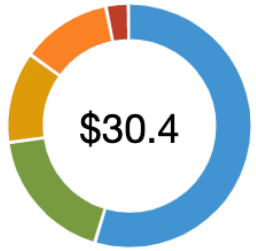
Welcome! We're starting in 5 minutes

- Course pre-requisites
  - MUST – Understand AWS (AWS Certified Developer course better)
  - MUST – Understand Apache Kafka (Kafka for Beginners course better)
  - SHOULD – Basic Linux & networking concepts
- Course plan
  - Getting started with Amazon MSK + installing admin tools
  - Deploying applications against Amazon MSK
  - Cluster Configuration and Operations
  - Cluster Security – TLS, ACL…
  - Cluster Monitoring – CloudWatch, Open Monitoring…
  - Stream Processing on Amazon MSK – Kafka Streams, Kinesis Data Analytics (Flink)…

Course Cost

Month-to-Date Spend by Service — Bill Details

The chart below shows the proportion of costs spent for each service you use.



MSK	$13.98
VPC	$4.61
EC2	$3.04
CertificateManager	$3.00
Other Services	$0.69
Tax	$5.07
Total	**$30.40**

About me – I'm Stephane!

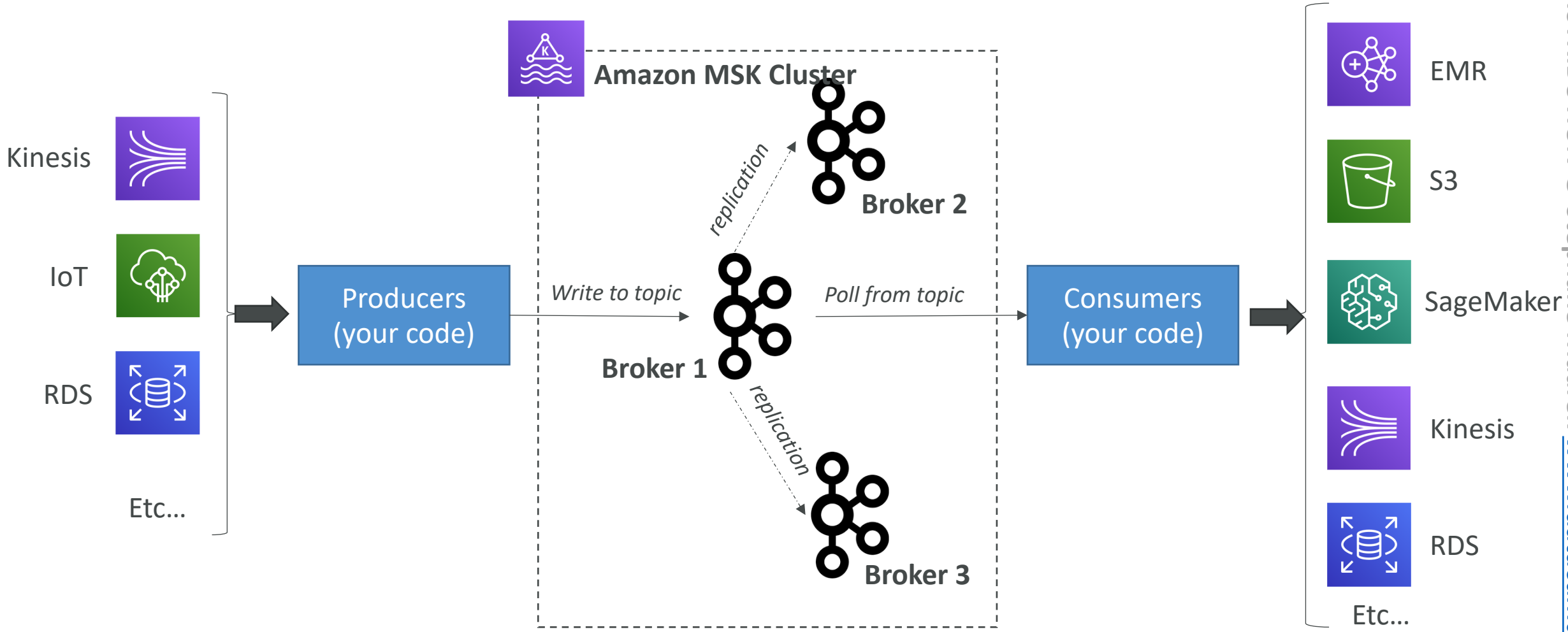
- 500,000+ students around the world on Apache Kafka & AWS
- Kafka Summit Program Committee since 2019
- Worked as in IT consultant and AWS Big Data Architect, Developer & SysOps
- Worked with AWS many years: built websites, apps, streaming platforms
- Got some help from **Krunal Vora** to put together this course!

- You can find me on
  - LinkedIn: https://www.linkedin.com/in/stephanemaarek
  - Instagram: https://www.instagram.com/stephanemaarek/
  - Twitter: https://twitter.com/stephanemaarek
  - Medium: https://medium.com/@stephane.maarek
  - GitHub: https://github.com/simplesteph

Amazon Managed Streaming for Apache Kafka (MSK)

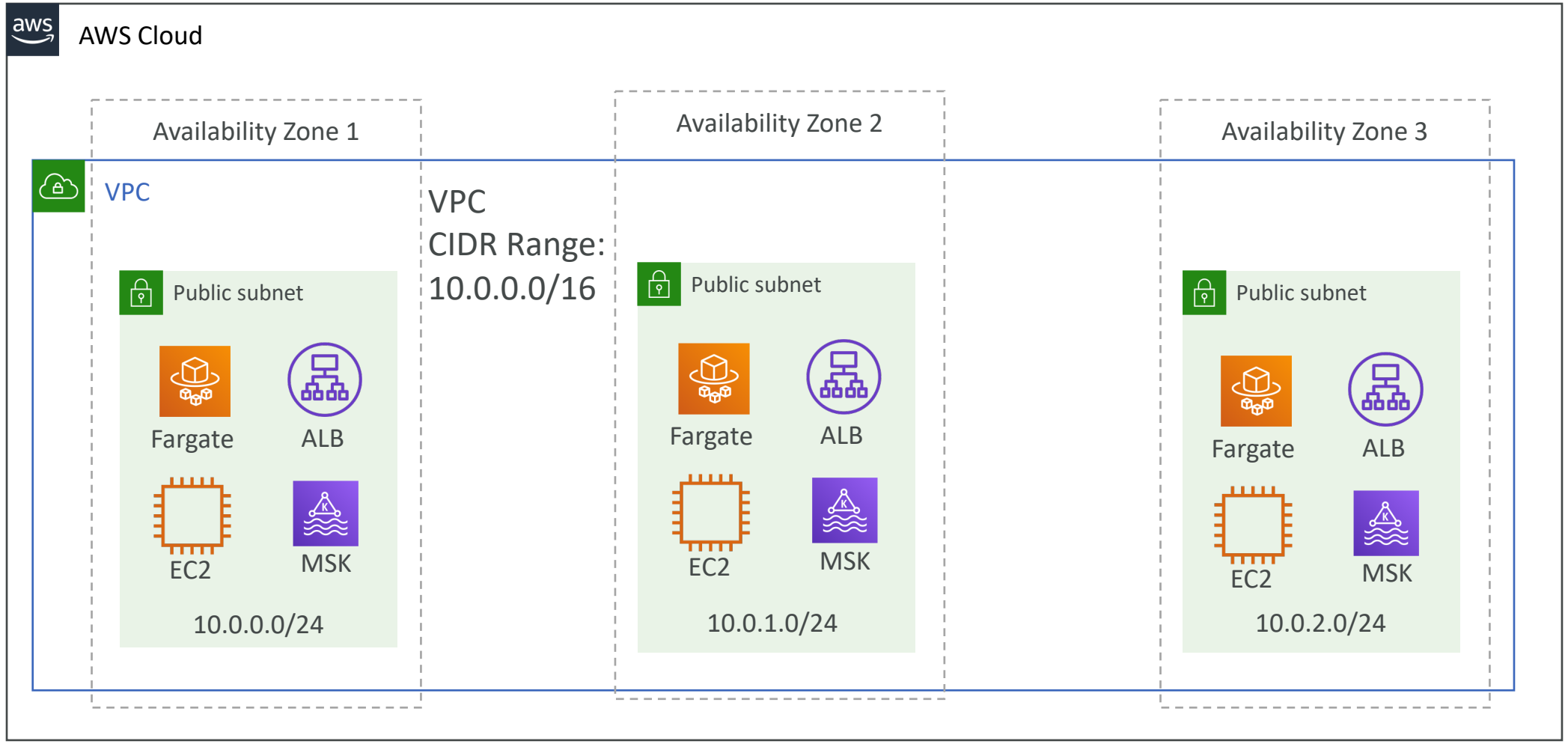
- Fully managed Apache Kafka on AWS
- Allow you to create, update, delete clusters (control plane)
- Amazon MSK creates & manages broker nodes & Zookeeper nodes for you
- Deploy the Amazon MSK cluster in your VPC, supports 2 or 3 AZ (HA)
- Automatic recovery from common Apache Kafka failures
- Can create custom configurations for your clusters
- Data is stored on EBS volumes

Apache Kafka at a high level

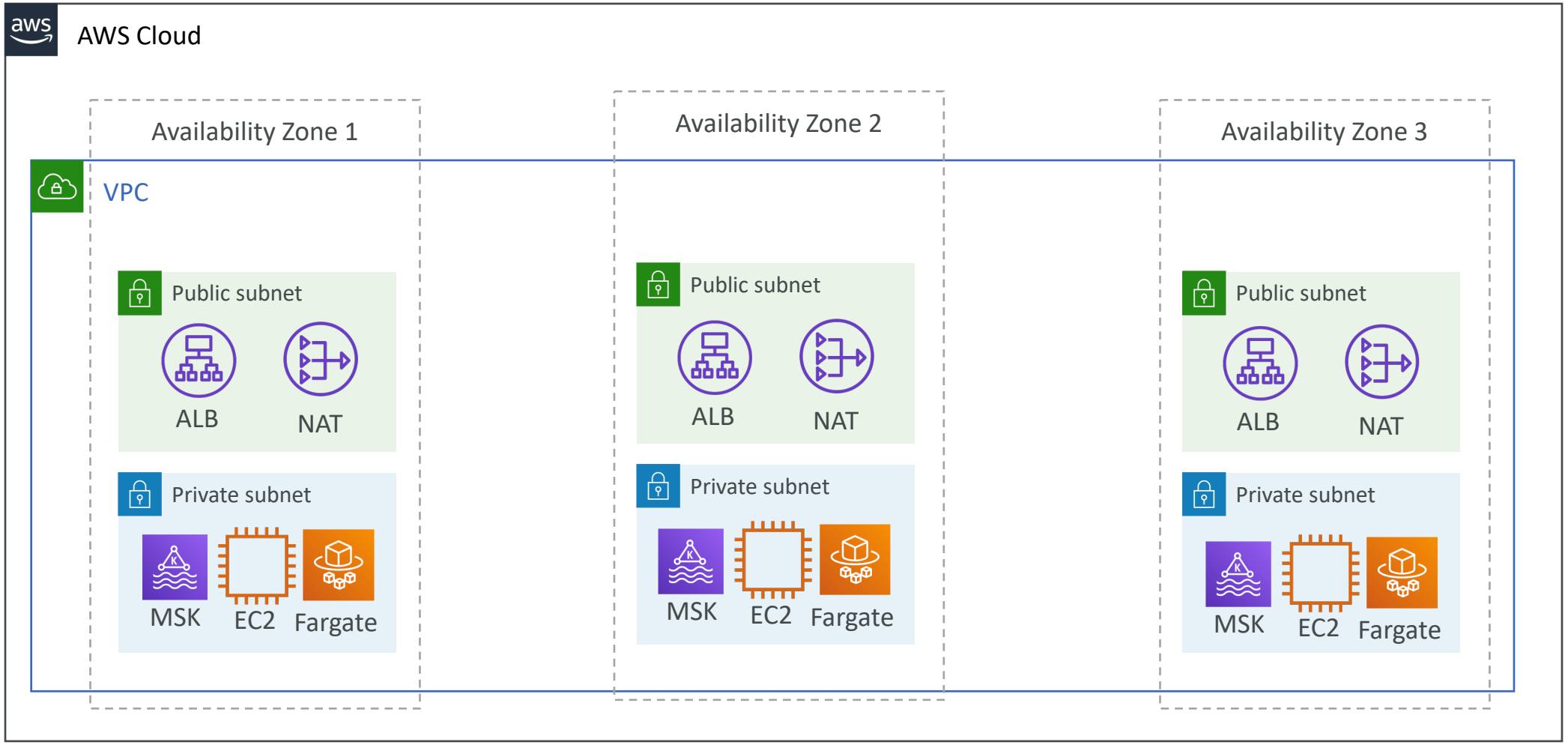


Setting up and accessing Amazon MSK and Administration Tools

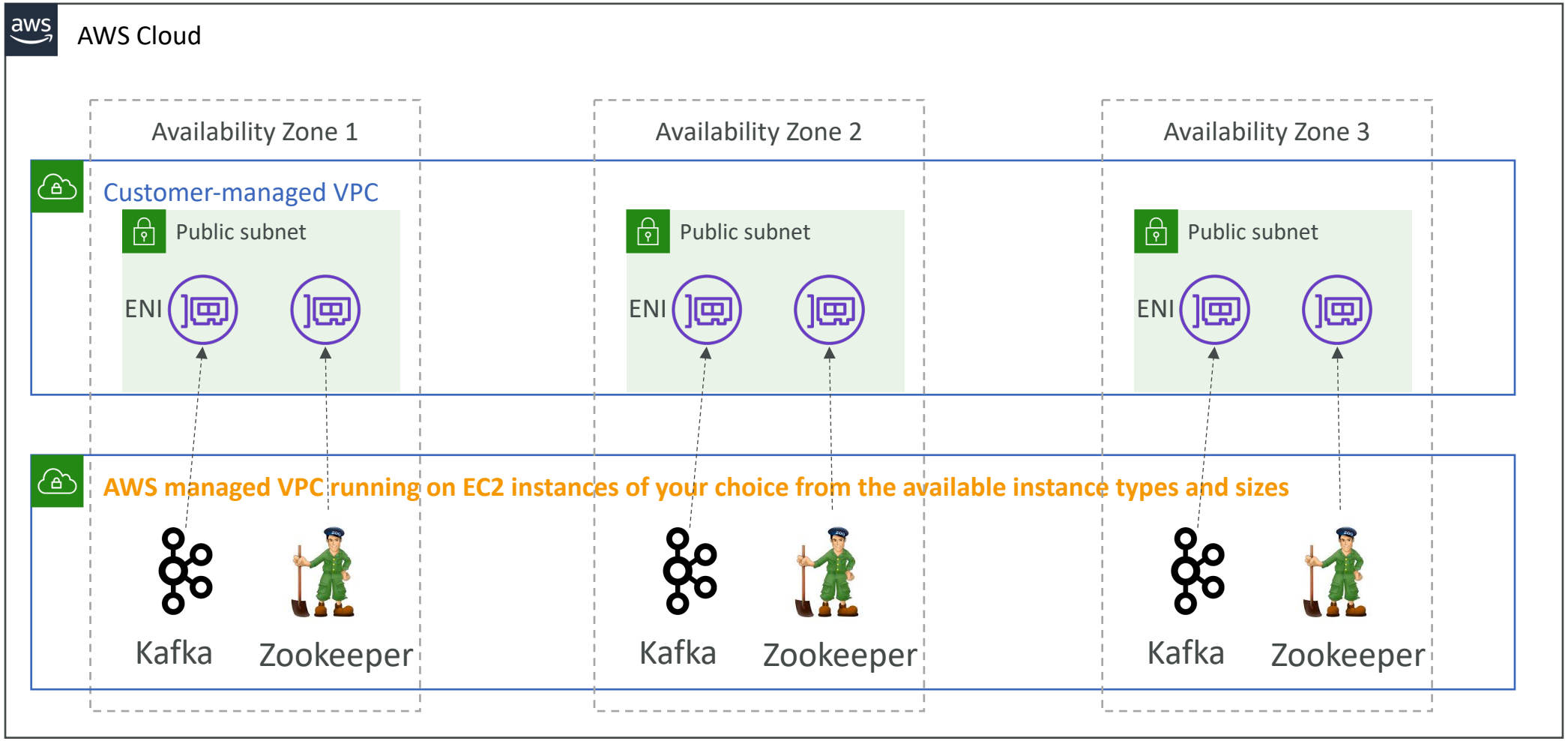
VPC Creation – Our course



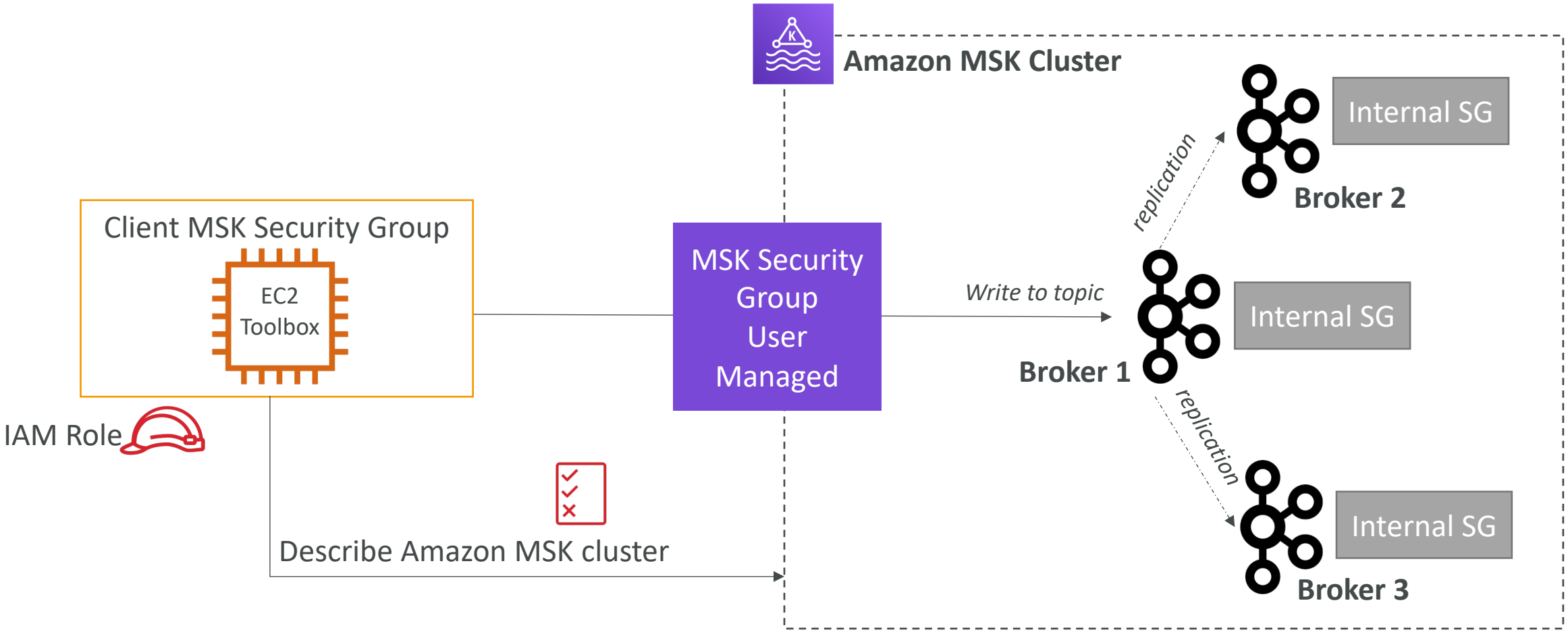
VPC Architecture – More secure – More $$



Creating an Amazon MSK Cluster



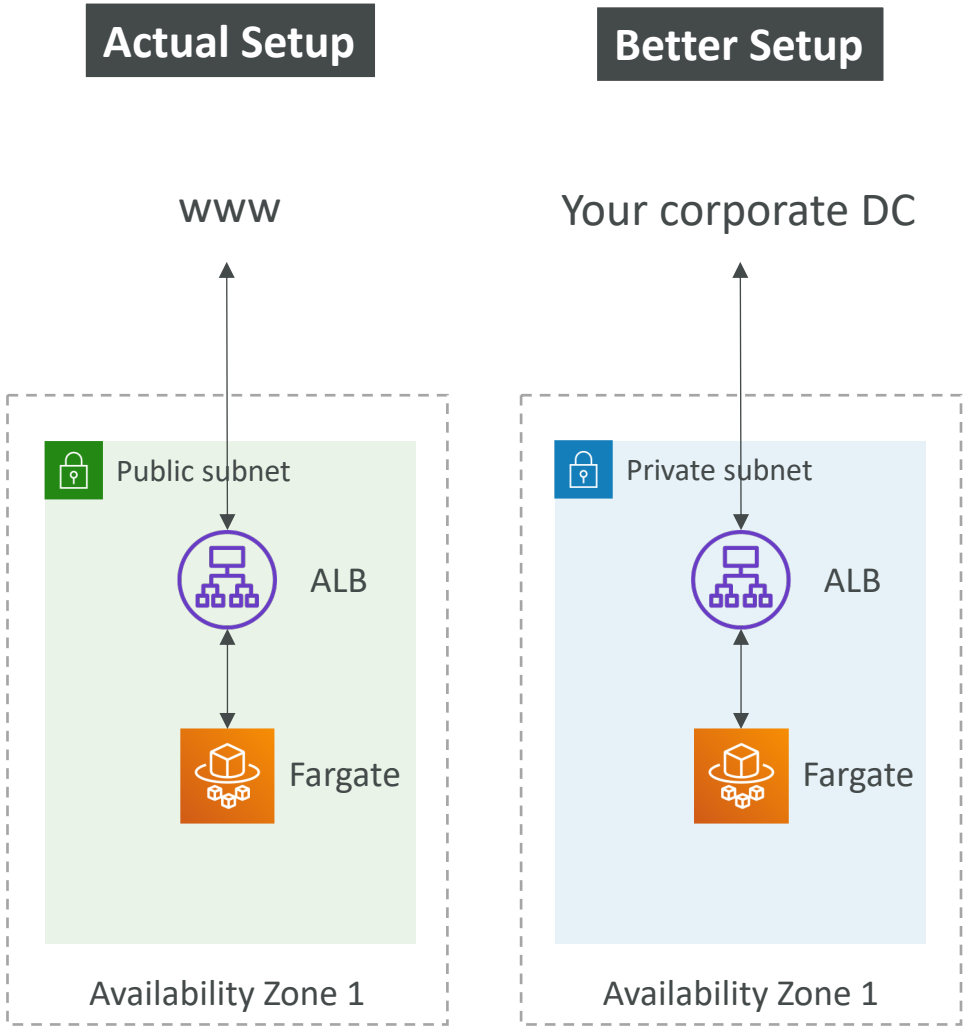
Amazon MSK Security Groups



Setting up AKHQ

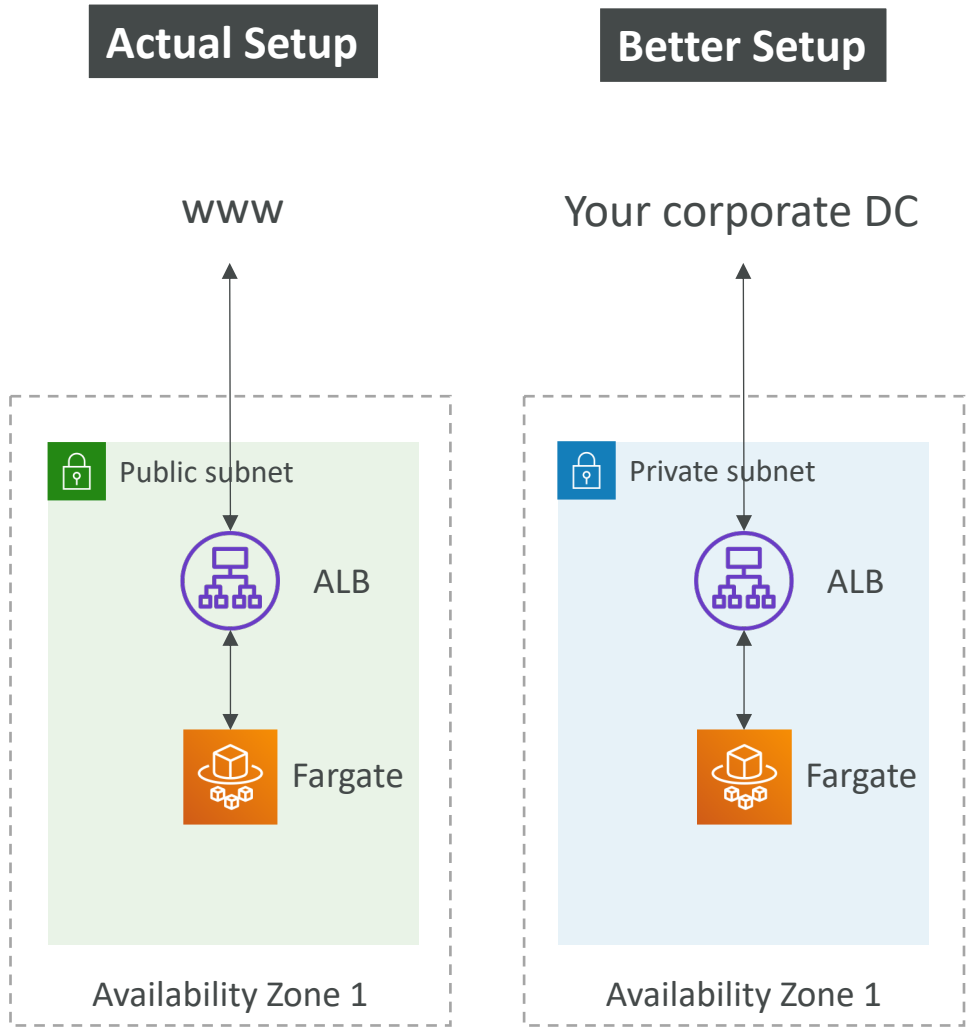
- https://github.com/tchiotludo/akhq

- Open-source Administration for:
  - Apache Kafka
  - Confluent Schema Registry
  - Kafka Connect
  - Kafka Security



Setting up ZooNavigator

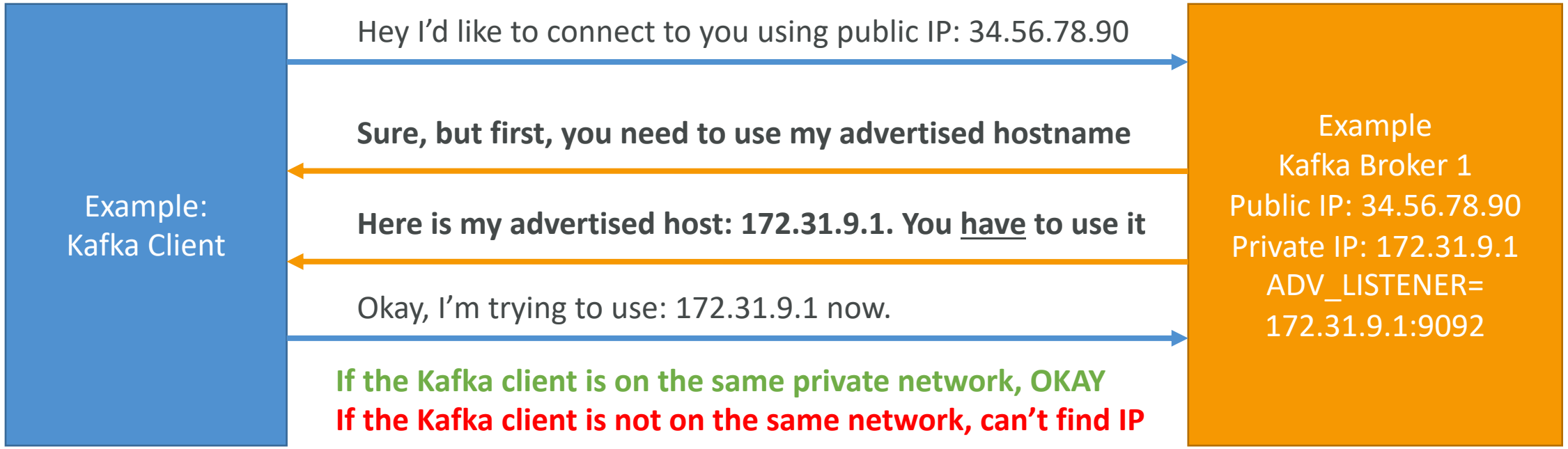
- https://github.com/elkozmon/zoonavigator
- Open-source Administration Tool for Zookeeper
- Can connect to multiple Zookeeper clusters (need to only deploy once)



Connecting to MSK from your laptop – AWS Client VPN & Conduktor

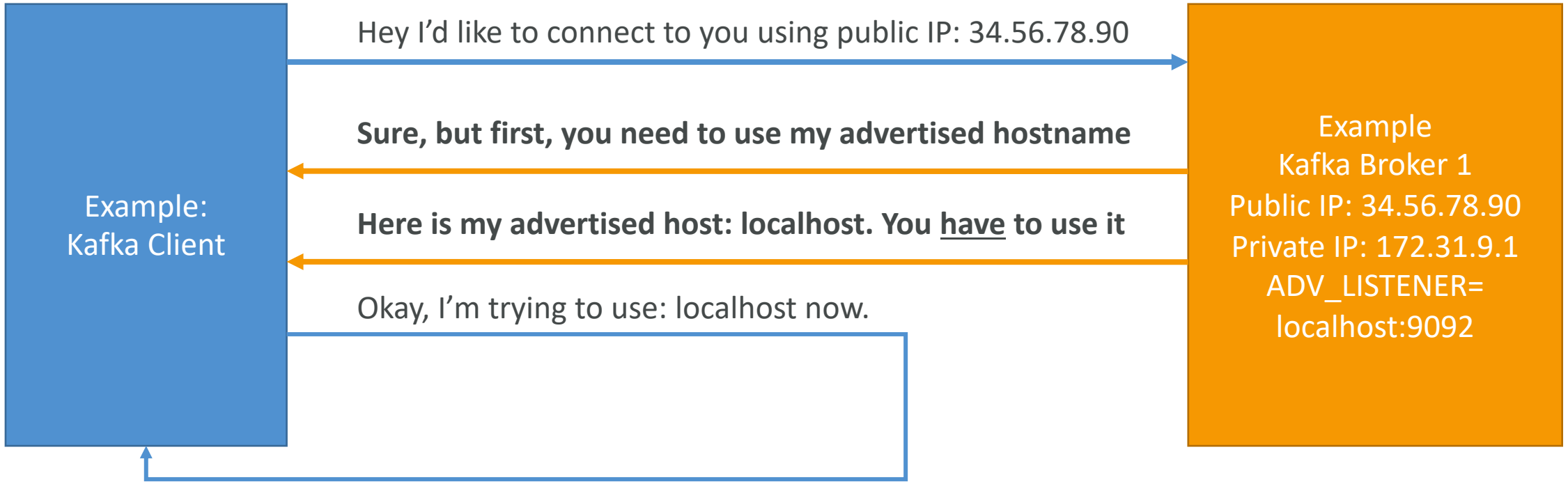
Understanding communications between Client and Kafka

- Advertised listeners is the most important setting of Kafka



But what if I put localhost? It works on my machine!

- Advertised hostname is the most important setting of Kafka

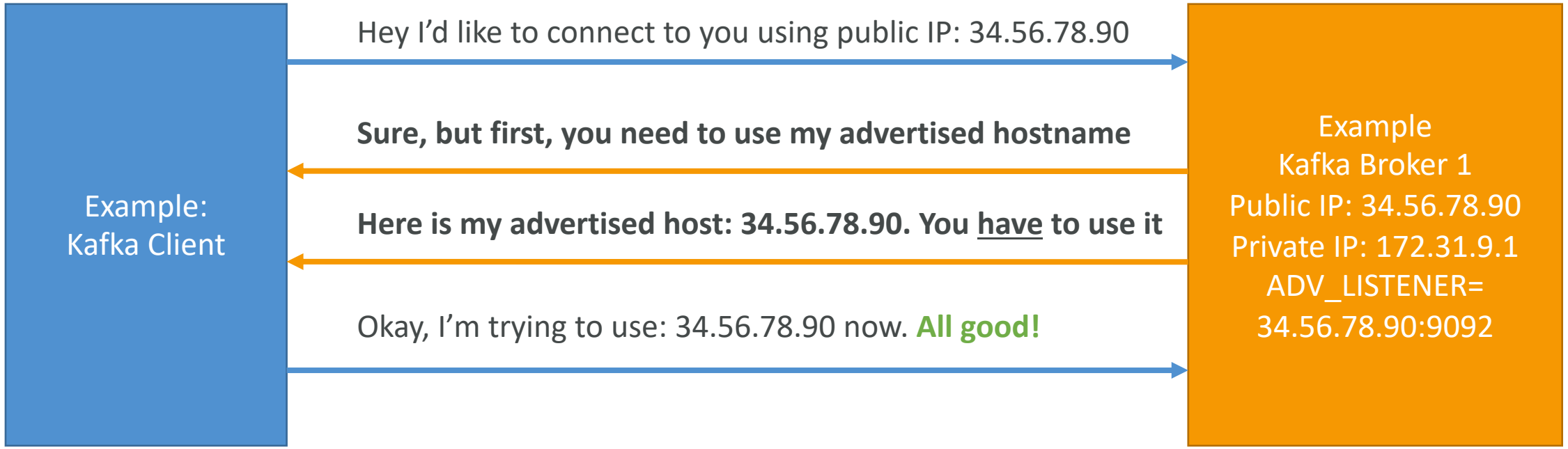


If the Kafka client is on the machine and you have 1 broker, OKAY

If the Kafka client is not on the same machine or you have 2 or more broker, NOT OKAY

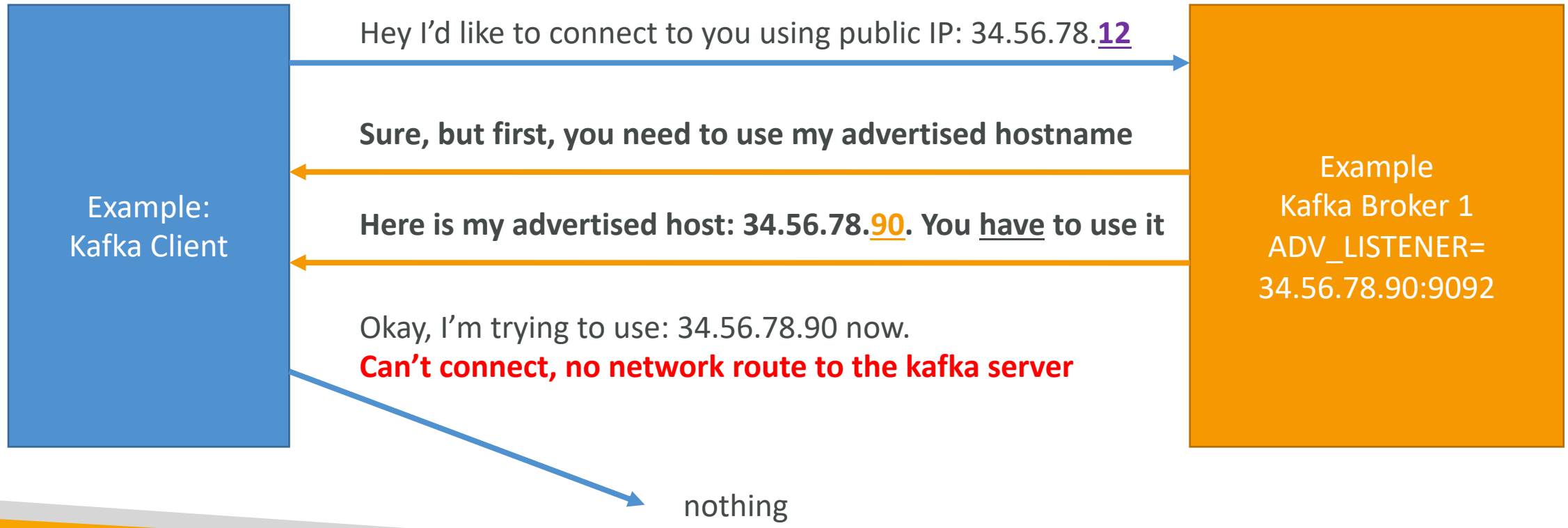
What if I use the public IP?

- Advertised hostname is the most important setting of Kafka



What if I use the public IP…
But after a reboot Kafka public IP changed!

- Assume the IP of your server has changed:
  - FROM 34.56.78.90 => TO 34.56.78.12



What about advertised.listeners for Amazon MSK?

```
listeners=CLIENT://b-1.msk-demo-cluster.rxt4pb.c2.kafka.eu-west-
2.amazonaws.com:9092,REPLICATION://…:9093
listener.security.protocol.map=CLIENT:PLAINTEXT,CLIENT_SECURE:SSL,REPLI
CATION:PLAINTEXT,REPLICATION_SECURE:SSL
```

- You can't change the listeners settings

- Kafka brokers on MSK have a public DNS that resolve to a private IP

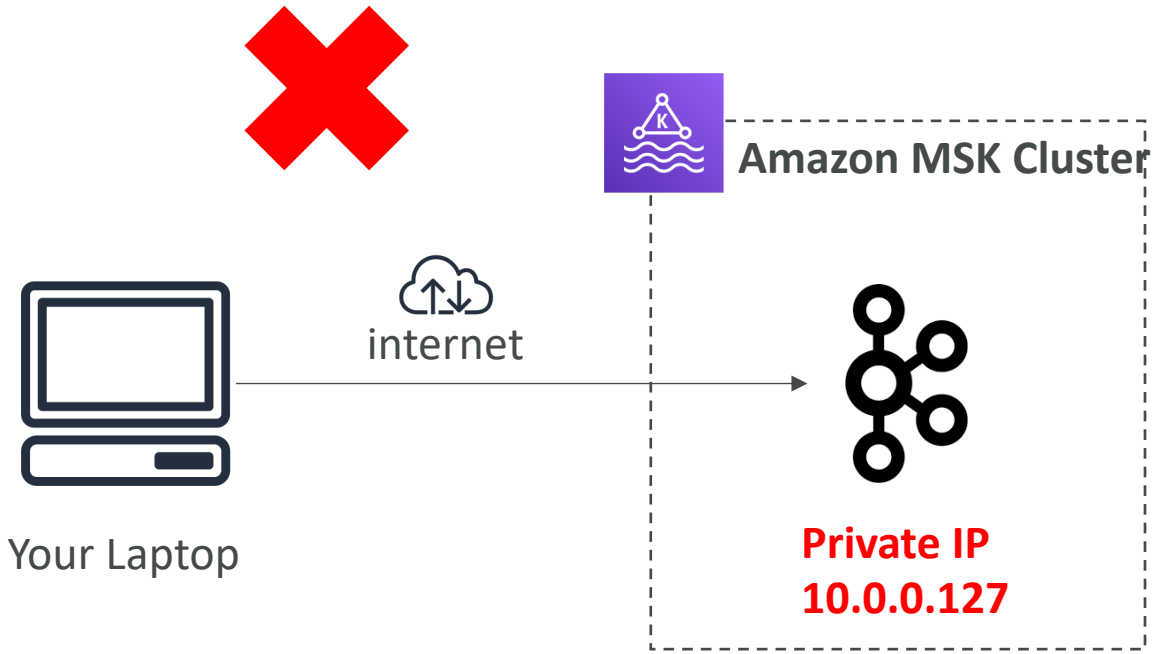
Connecting from your laptop to Amazon MSK

- Find the bootstrap servers hostname using *get-bootstrap-brokers*
- Apache Kafka & Zookeeper DNS in Amazon MSK resolve to a private IP

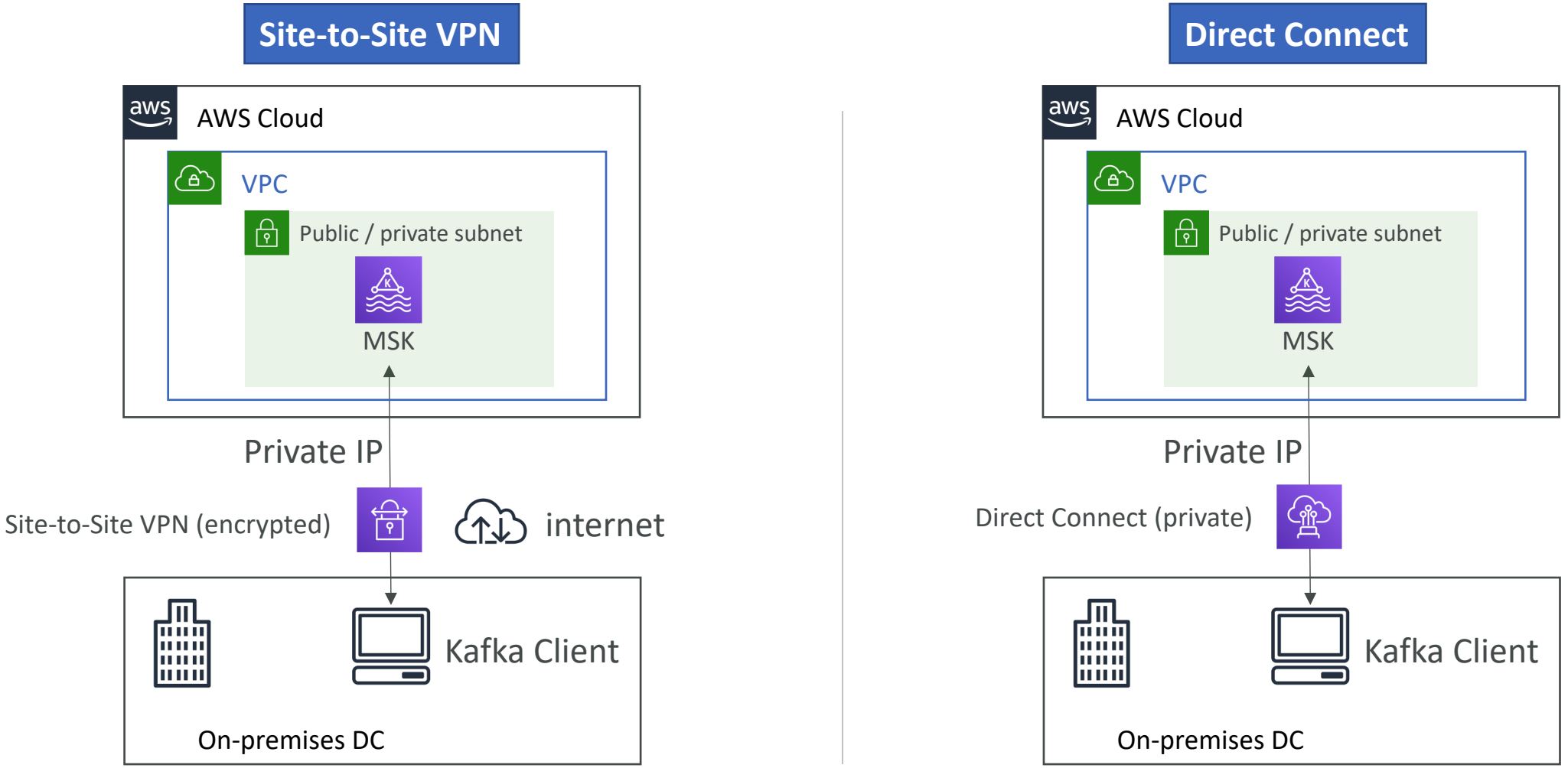
- Example for Kafka:
  - **Public DNS:** b-1.msk-demo-cluster.rxt4pb.c2.kafka.eu-west-2.amazonaws.com
  - => A record in **10.0.0.127**
- Example for Zookeeper:
  - **Public DNS:** z-2.msk-demo-cluster.rxt4pb.c2.kafka.eu-west-2.amazonaws.com
  - => A record in **10.0.1.78**

- Can only connect to Amazon MSK through the VPC or a private connection

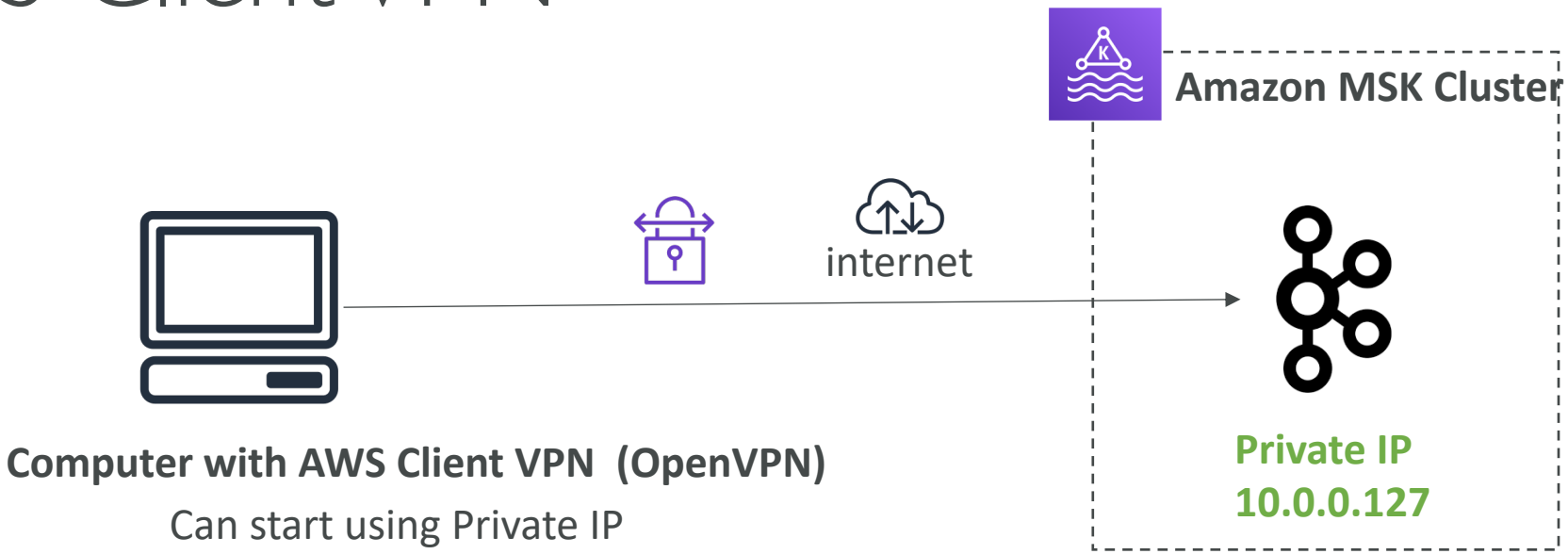
Impossible connection configuration



Connecting from your laptop to Amazon MSK



AWS Client VPN



AWS Client VPN pricing

Region: US East (Ohio)

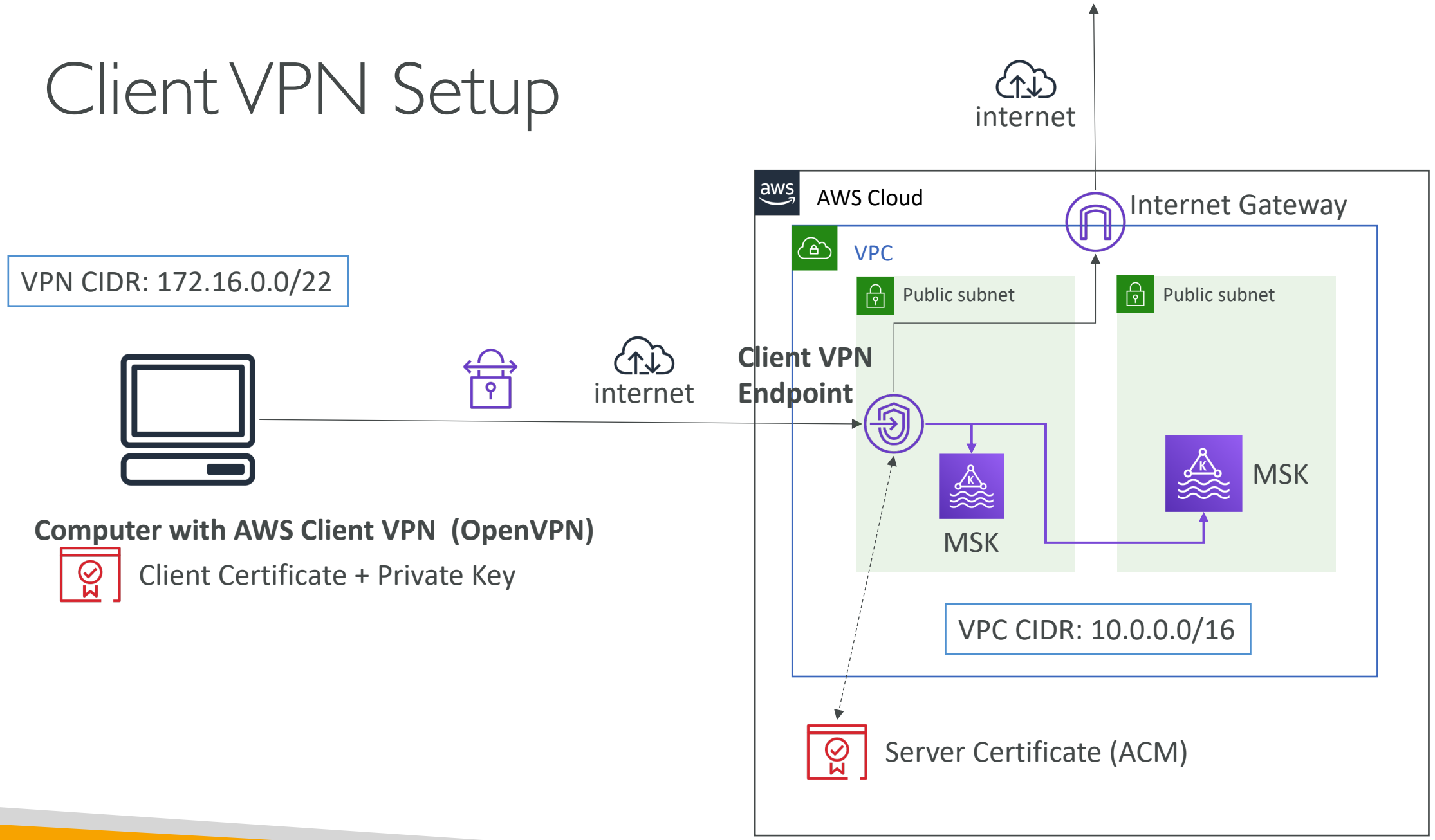
	Price
AWS Client VPN endpoint association	$0.10 per hour
AWS Client VPN connection	$0.05 per hour

AWS Client VPN Setup

- 3 types of authentication available
  - Active Directory authentication (user-based)
  - Mutual authentication (certificate-based)
  - Single sign-on (SAML-based federated authentication) (user-based)
- We will do **Mutual Authentication**
  - We will need to generate a Client and a Server certificate
  - We will need to upload the certificates in ACM (Amazon Certificate Manager)

- Instructions for Windows / Linux / Mac available here: https://docs.aws.amazon.com/vpn/latest/clientvpn-admin/client-authentication.html#mutual

Client VPN Setup



Using AWS Client VPN

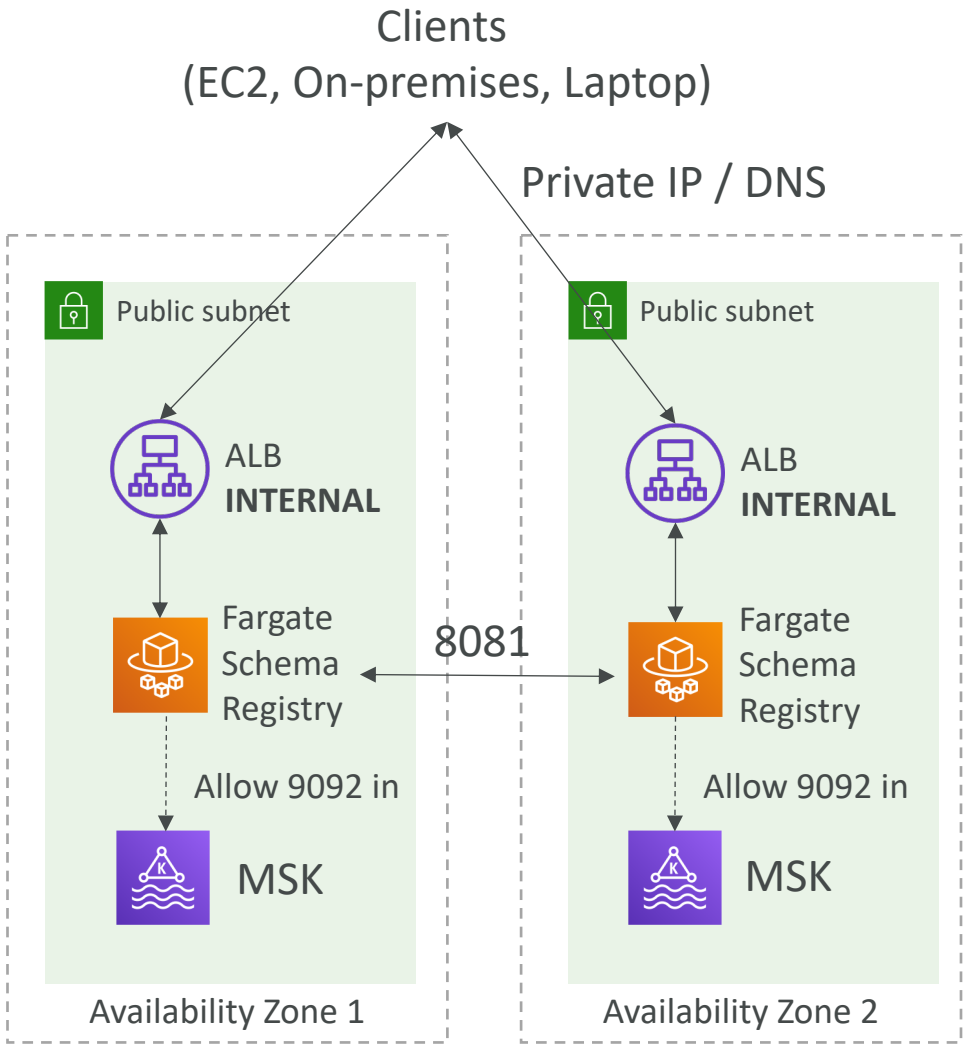
- AWS Client VPN application download - https://aws.amazon.com/vpn/client-vpn-download/
- https://docs.aws.amazon.com/vpn/latest/clientvpn-admin/scenario-internet.html
- Make sure to edit security group rules of Amazon MSK to allow the Client VPN security group!

Deploying applications against Amazon MSK

Setting up Confluent Schema Registry

- https://docs.confluent.io/current/schema-registry/installation/config.html
- Using Fargate
- In HA-mode with an internal ALB

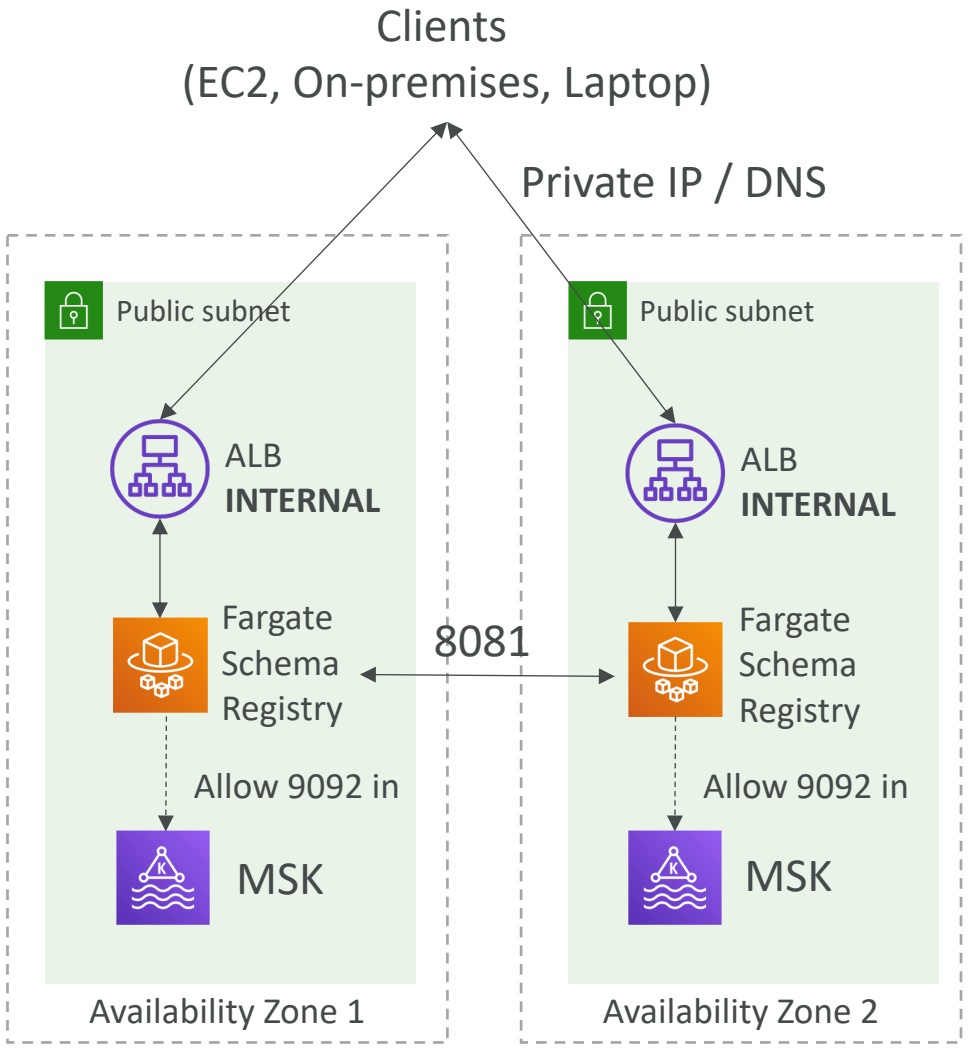
- **Important – License:** *Confluent Community License* means that you can freely install / modify / re-use, you just can't make a competing SaaS offering (not a legal advice)
- **Read more here:** https://www.confluent.io/confluent-community-license-faq/



Setting up Confluent Schema Registry

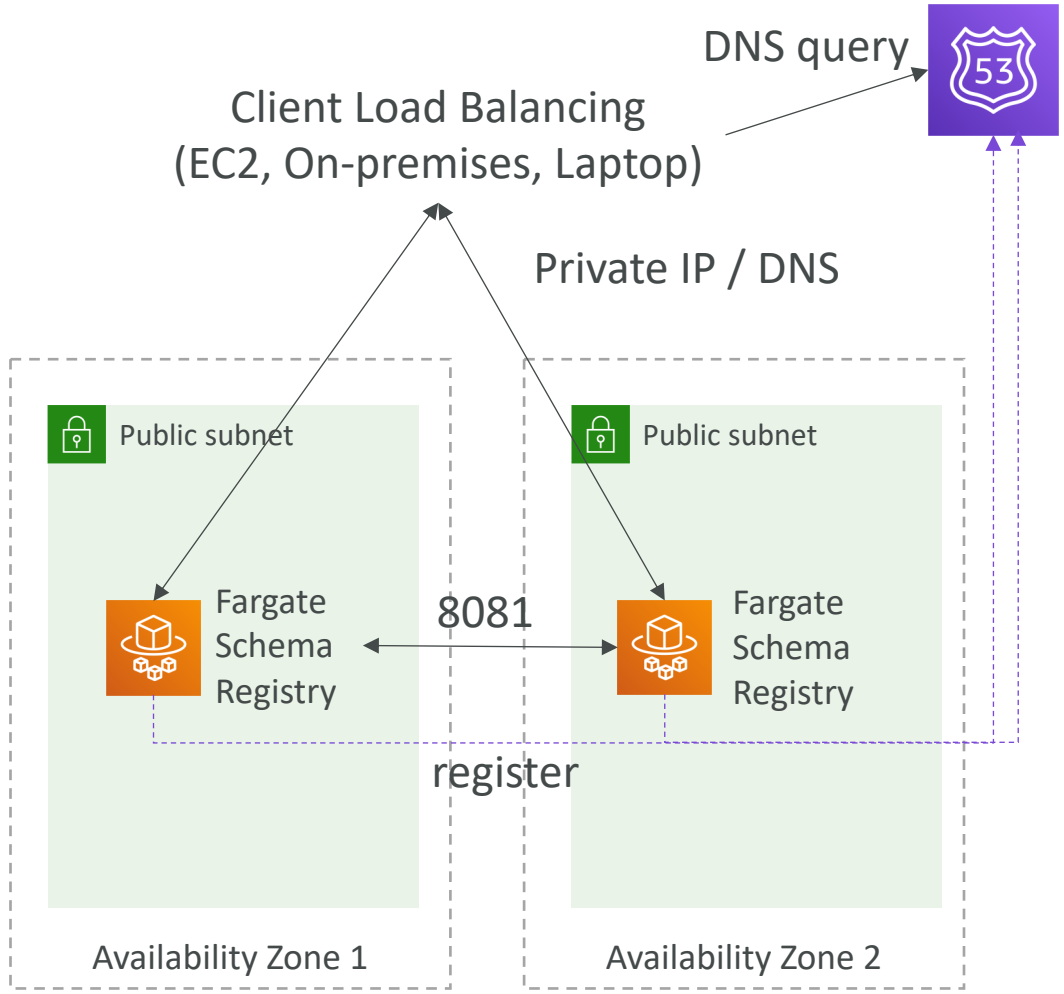
- AWS - IMPORTANT:
  - must set the **host.name** configuration correctly => resolved at runtime into the Fargate task host name
  - Must set the security groups correctly (ALB / Schema Registry / Amazon MSK)

- IMPORTANT: the Docker image translates SCHEMA_REGISTRY_FOO_BAR=value into the config "foo.bar=value"



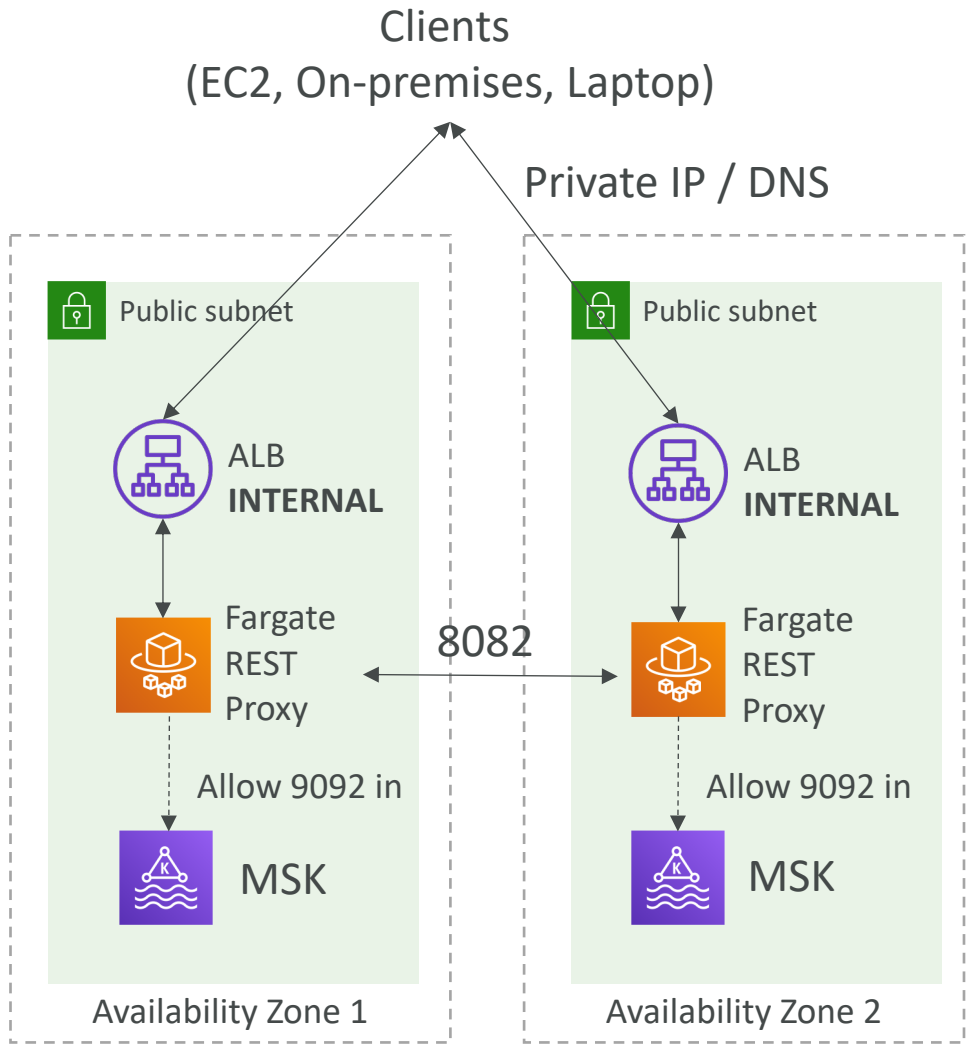
Setting up Confluent Schema Registry ECS Service Discovery

- https://docs.confluent.io/current/schema-registry/installation/config.html
- Using Fargate
- ECS Service Discovery - Route 53
- This removes the need from maintaining an ALB
- SSL certificates must be loaded onto the Docker task => create your own Docker image with wrappers



Setting up Confluent REST Proxy

- https://docs.confluent.io/current/kafka-rest/config.html
- Using Fargate
- If using ALB:
  - Enable stickiness
  - Check security groups
- Or use ECS Service Discovery with Route 53
- Make sure to add a rule in Amazon MSK's security group



Cluster Configuration and Operations

Amazon MSK Cluster Sizing – Broker Size

- EC2 instance types of the brokers **cannot be changed** once the cluster is created (on the roadmap), but **you can add brokers over time**
- t3.small (low cost development) | m5.family (production workload)

Broker instance type	Maximum number of partitions (including replicas) per broker
t3.small	300
m5.large or m5.xlarge	1000
m5.2xlarge	2000
m5.4xlarge, m5.8xlarge, m5.12xlarge, m5.16xlarge, or m5.24xlarge	4000

Amazon MSK Soft Limits

- Up to 90 brokers per account and 30 brokers per cluster
- A minimum of 1 GiB of storage per broker
- A maximum of 16384 GiB of storage per broker
- Up to 100 configurations per account

- You can create a support case to increase these limits

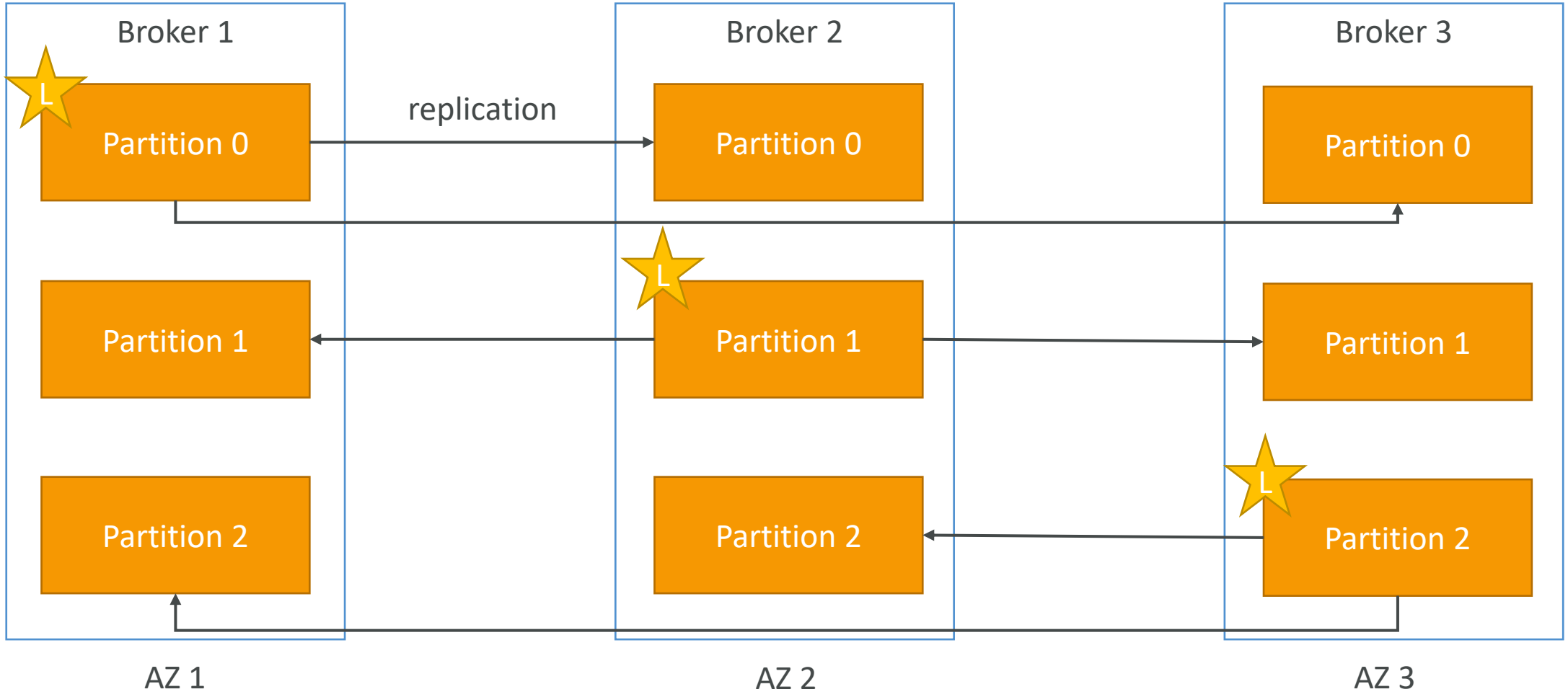
Choosing the right number of brokers & sizing

- Sizing and Pricing spreadsheet: https://amazonmsk.s3.amazonaws.com/MSK_Sizing_Pricing.xlsx
- Apache Kafka supports 200K partitions per cluster: https://blogs.apache.org/kafka/entry/apache-kafka-supports-more-partitions
- Test your workload on the cluster you provision
- Choosing the number of partitions per topic is not an easy task:
  - https://stackoverflow.com/questions/50271677/how-to-choose-the-no-of-partitions-for-a-kafka-topic
  - https://www.confluent.io/blog/how-choose-number-topics-partitions-kafka-cluster/

High Availability with Amazon MSK

- Choose the number of AZ:
  - 2 AZ => good for development, recommended RF = 2
  - 3 AZ => good for production (2 AZ can be up if 1 is down), use RF = 3
- Choose number of brokers per AZ:
  - Brokers are always a multiple of #AZ
  - Setting 'broker.rack' is automatically set to the AZ ID (ex: euw2-az2)
  - => When creating a topic, the partitions are going to be balanced across AZ
  - => If you re-partition a topic, you have to balance across AZ yourself
- Using Amazon MSK, inter-AZ traffic is free (not using EC2 – self managed)
  - Use RF=3 without extra cost!

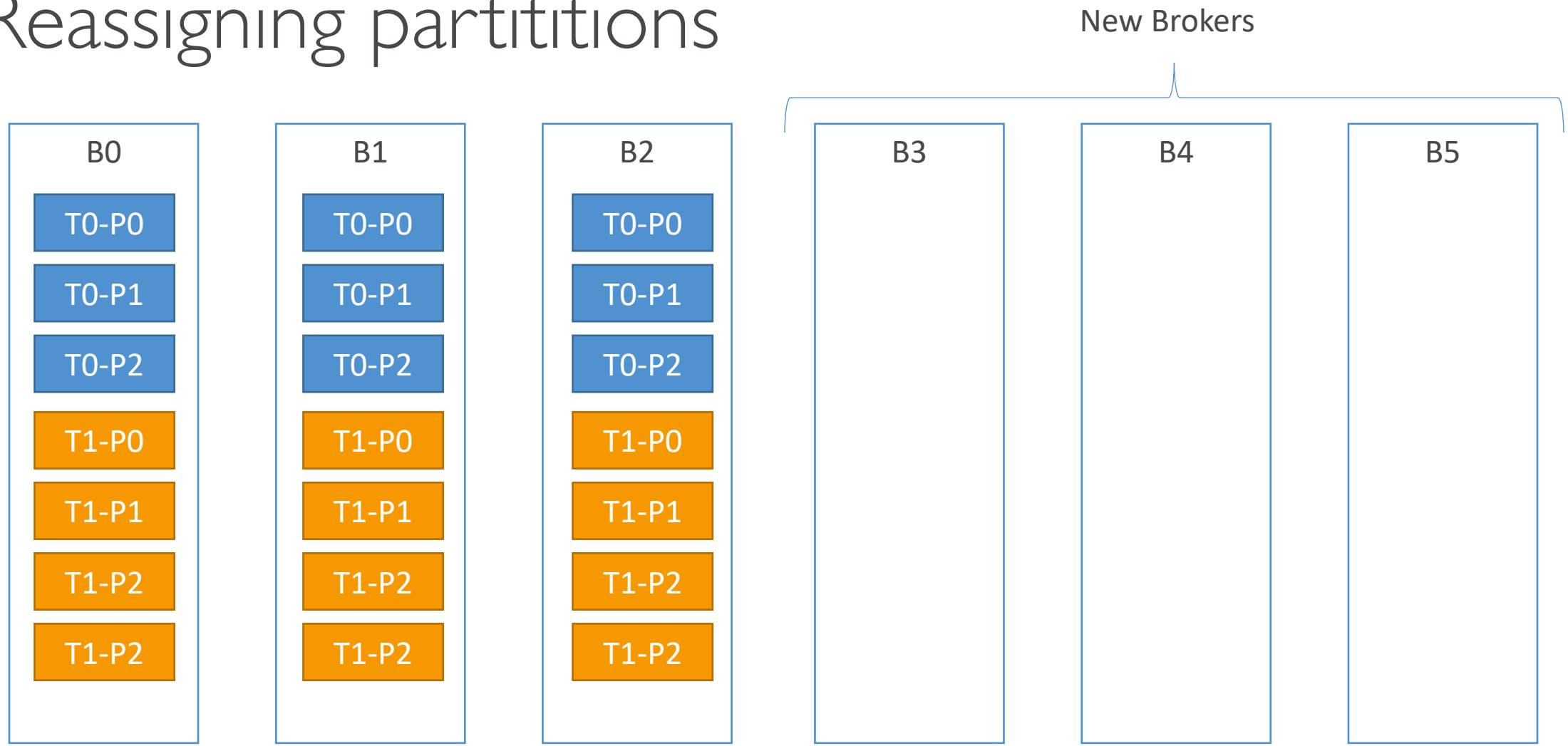
Inter AZ replication traffic



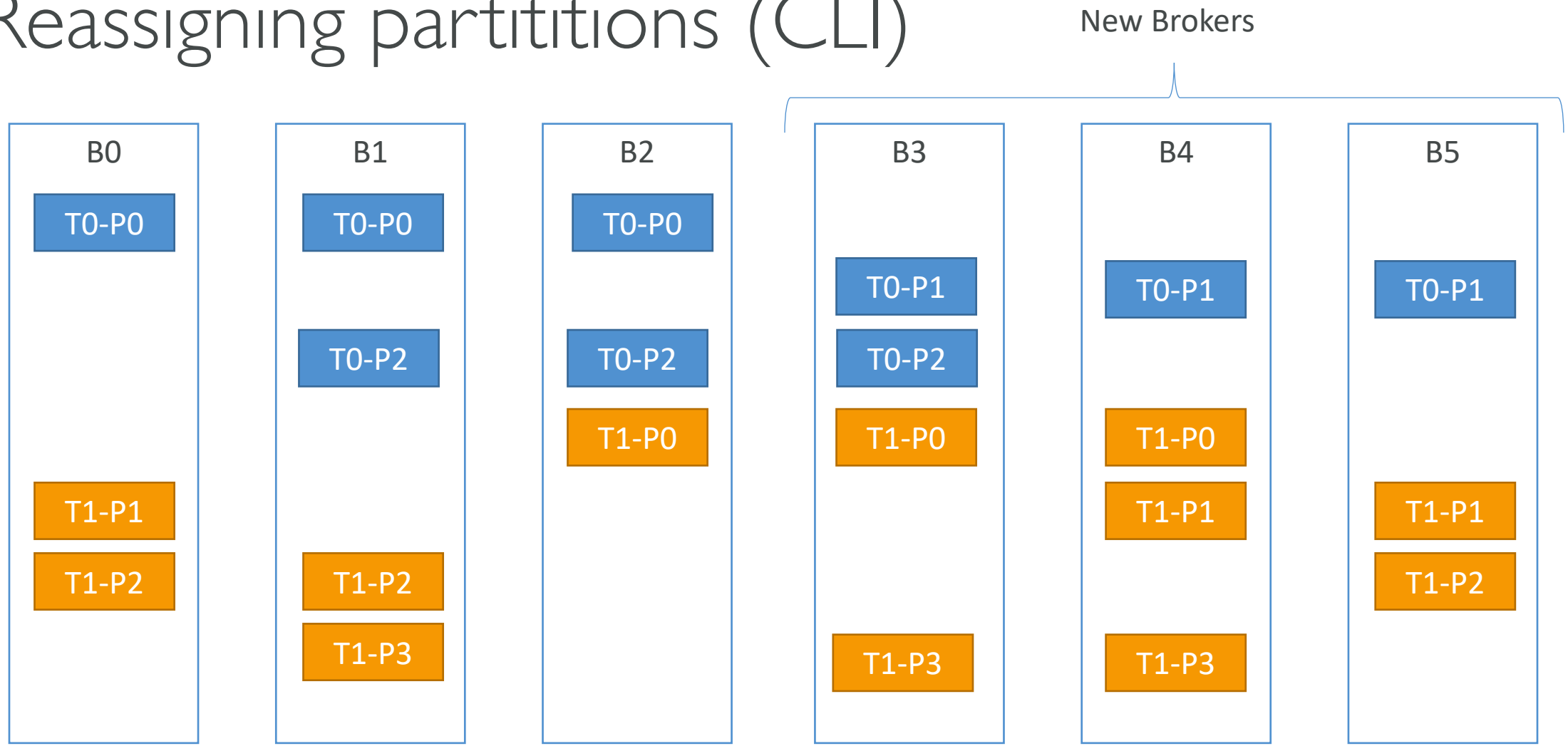
Scaling Horizontally Amazon MSK

- Apache Kafka scales horizontally, so to add throughput, you can add brokers
- In Amazon MSK
  - # brokers is a multiple of # AZ
  - You can only increase the number of brokers
  - You cannot change the broker instance type (yet, on the roadmap)
- After adding brokers, you must re-assign partitions
  - Use **kafka-reassign-partitions.sh**
    https://kafka.apache.org/documentation/#basic_ops_cluster_expansion
  - Use OSS tools: LinkedIn Kafka Tools or DataDog Kafka Kit (topicmappr)
  - Use Conduktor Desktop

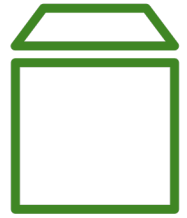
Reassigning partitions



Reassigning partitions (CLI)



Amazon MSK Broker Storage



- There's no auto-scaling of storage (yet)
- But you can increase broker storage in increments of 100GB, once every 6 hours
- To avoid running out of disk space for messages, create a CloudWatch alarm that watches the **KafkaDataLogsDiskUsed** metric. When the value of this metric reaches or exceeds 85%, perform one or more of the following actions:
  - Increase broker storage (can automate it using a Lambda function)
  - Reduce the message retention period or log size. For information on how to do that, see Adjust data retention parameters.
  - Delete unused topics
- You cannot decrease the storage

Amazon MSK Upgrade

- One click upgrade!

- Can update the clients before or after upgrading the broker, they should still work (see next slide)
- Ensure all topics have RF > 1
- Ensure min.insync.replicas <= RF – 1
- No downtime upgrade

Client Bi-Directional Compatibility

- As of Kafka 0.10.2 (introduced in June 2017), your clients & Kafka Brokers have a capability called bi-directional compatibility (because API calls are now versioned)

- This means:
  - An OLDER client (ex 1.1) can talk to a NEWER broker (2.0)
  - A NEWER client (ex 2.0) can talk to an OLDER broker (1.1)
- Bottom Line:
  - Always use the latest client library version if you can
  - You can upgrade Kafka without breaking your client applications

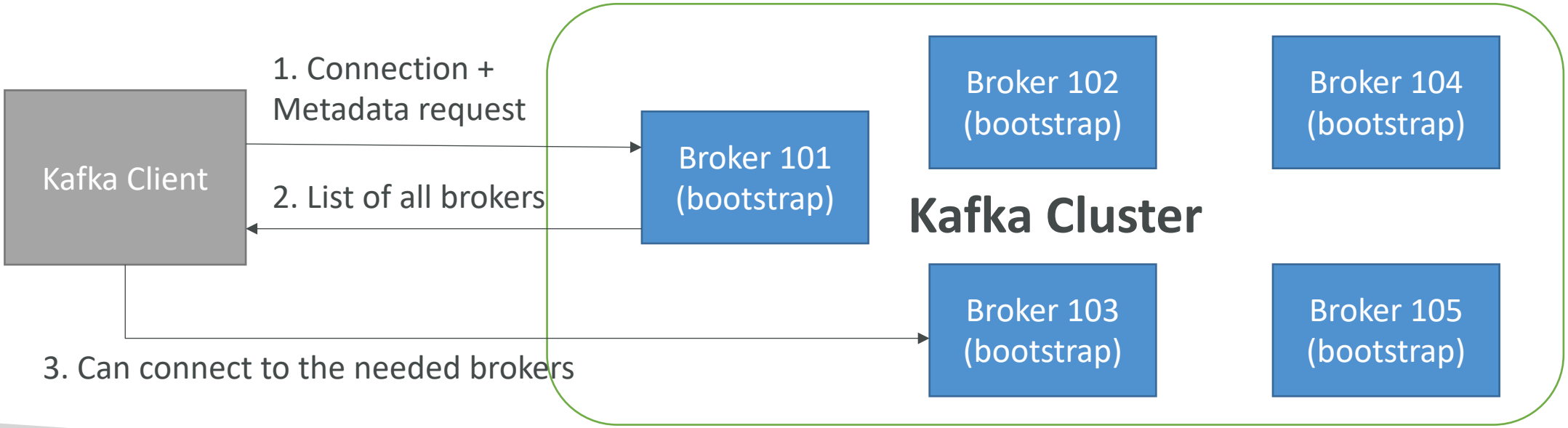
- More Reading: https://www.confluent.io/blog/upgrading-apache-kafka-clients-just-got-easier/

Important Client Settings – Summary

- Use one broker per AZ for the bootstrap URL
  - If a broker is down, you can still connect to the cluster
- Important producer settings
  - Cluster default = min.insync.replicas=2
  - Set your Kafka producers to acks=all to use the min.insync.replicas=2 setting
  - Ensure a topic has a RF = 3
  - Setting enable.idempotence=true is not a bad idea
- Important consumer settings
  - Cost savings: rack aware Kafka Consumer (Kafka v2.4.0 +)
  - https://cwiki.apache.org/confluence/display/KAFKA/KIP-392%3A+Allow+consumers+to+fetch+from+closest+replica

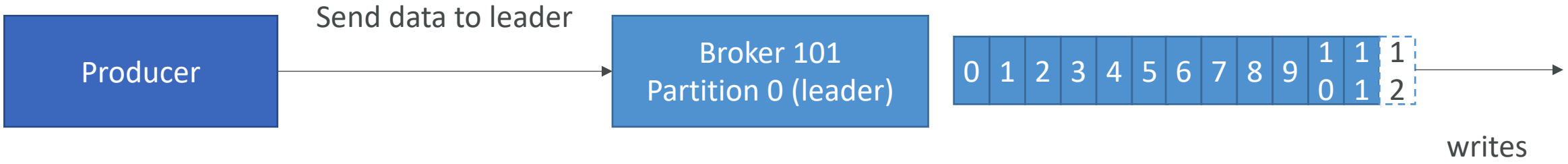
Kafka Broker Discovery

- Every Kafka broker is also called a "bootstrap server"
- That means that **you only need to connect to one broker**, and you will be connected to the entire cluster.
- Each broker knows about all brokers, topics and partitions (metadata)
- It's best practice to use 3 bootstrap brokers in your connection string



Producers Acks Deep Dive
acks = 0 (no acks)

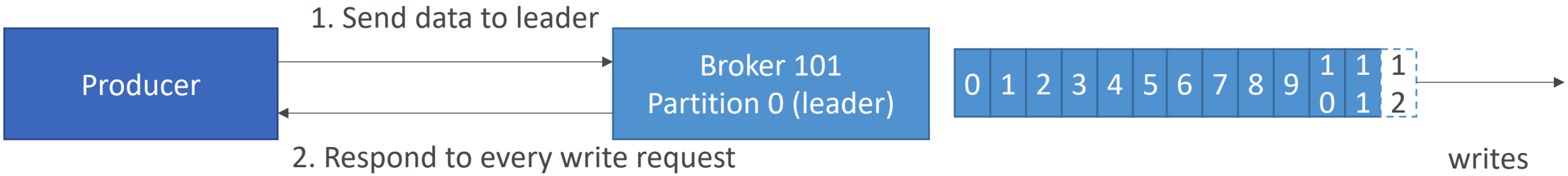
- No response is requested
- If the broker goes offline or an exception happens, we won't know and will lose data



- Useful for data where it's okay to potentially lose messages:
  - Metrics collection
  - Log collection

Producers Acks Deep Dive
acks = 1 (leader acks)

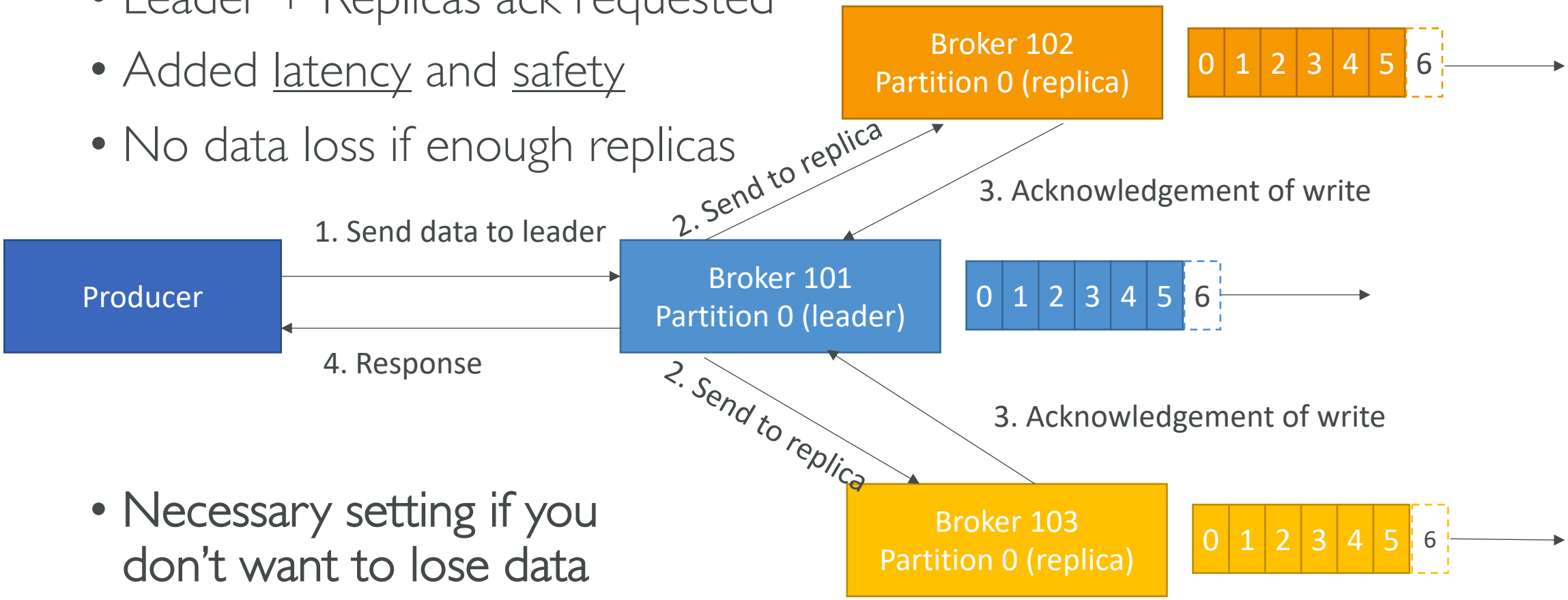
- Leader response is requested, but replication is not a guarantee (happens in the background)
- If an ack is not received, the producer may retry



- If the leader broker goes offline but replicas haven't replicated the data yet, we have a data loss.

Producers Acks Deep Dive
acks = all (replicas acks)

- Leader + Replicas ack requested
- Added latency and safety
- No data loss if enough replicas



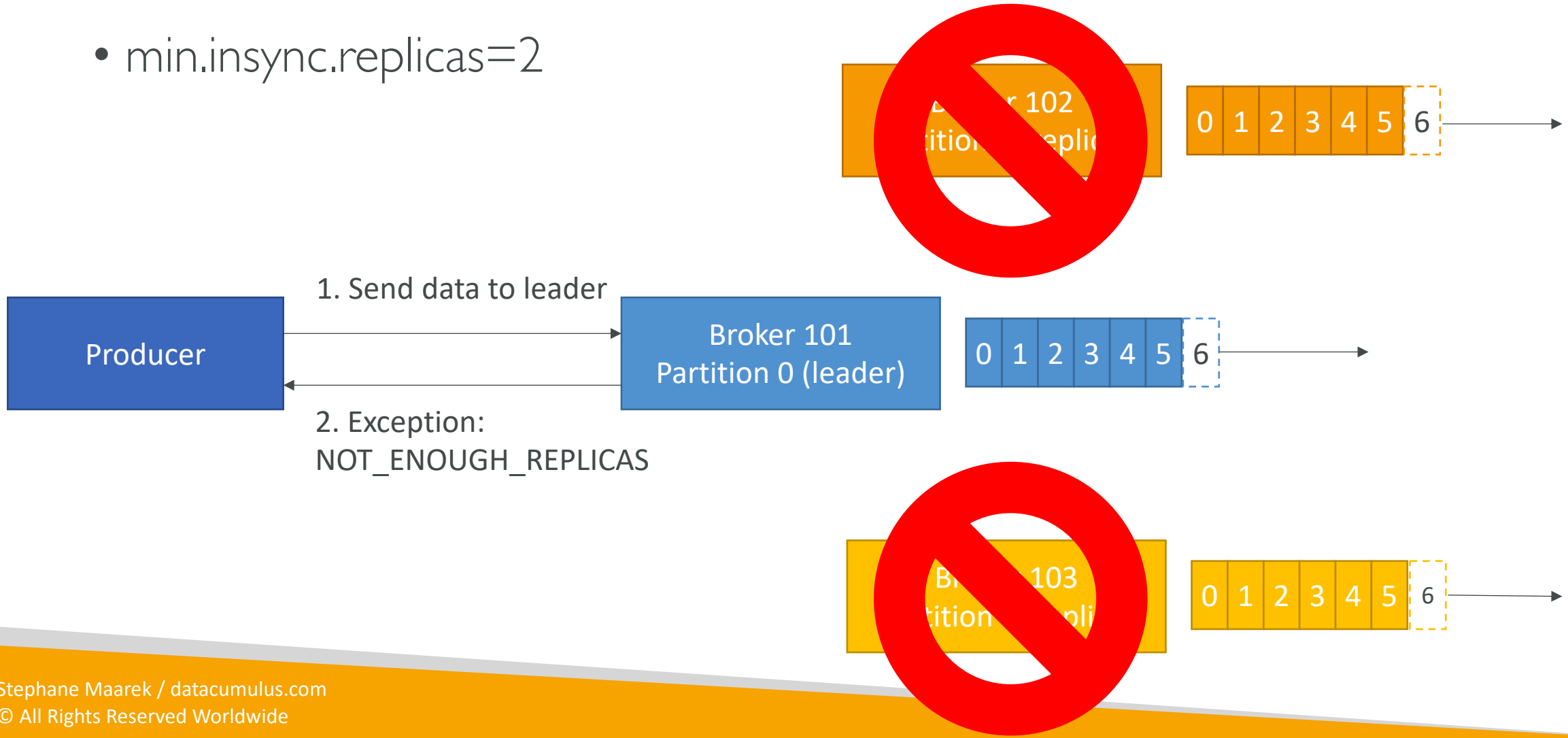
- Necessary setting if you don't want to lose data

Producers Acks Deep Dive
acks = all (replicas acks)

- Acks=all must be used in conjunction with min.insync.replicas.
- min.insync.replicas can be set at the broker or topic level (override).
- min.insync.replicas=2 implies that at least 2 brokers that are ISR (including leader) must respond that they have the data.
- That means if you use replication.factor=3, min.insync=2, acks=all, you can only tolerate 1 broker going down, otherwise the producer will receive an exception on send.

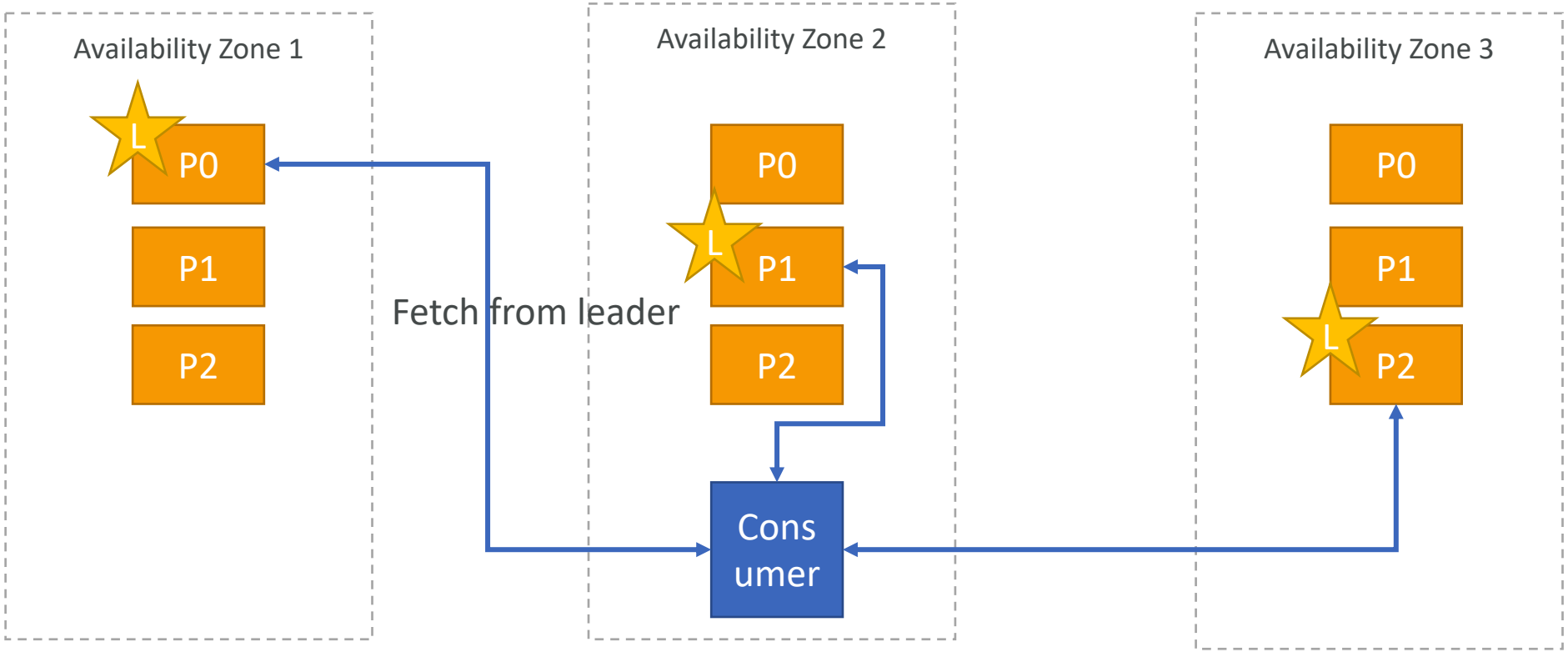
Producers Acks Deep Dive
acks = all (replicas acks)

- min.insync.replicas=2



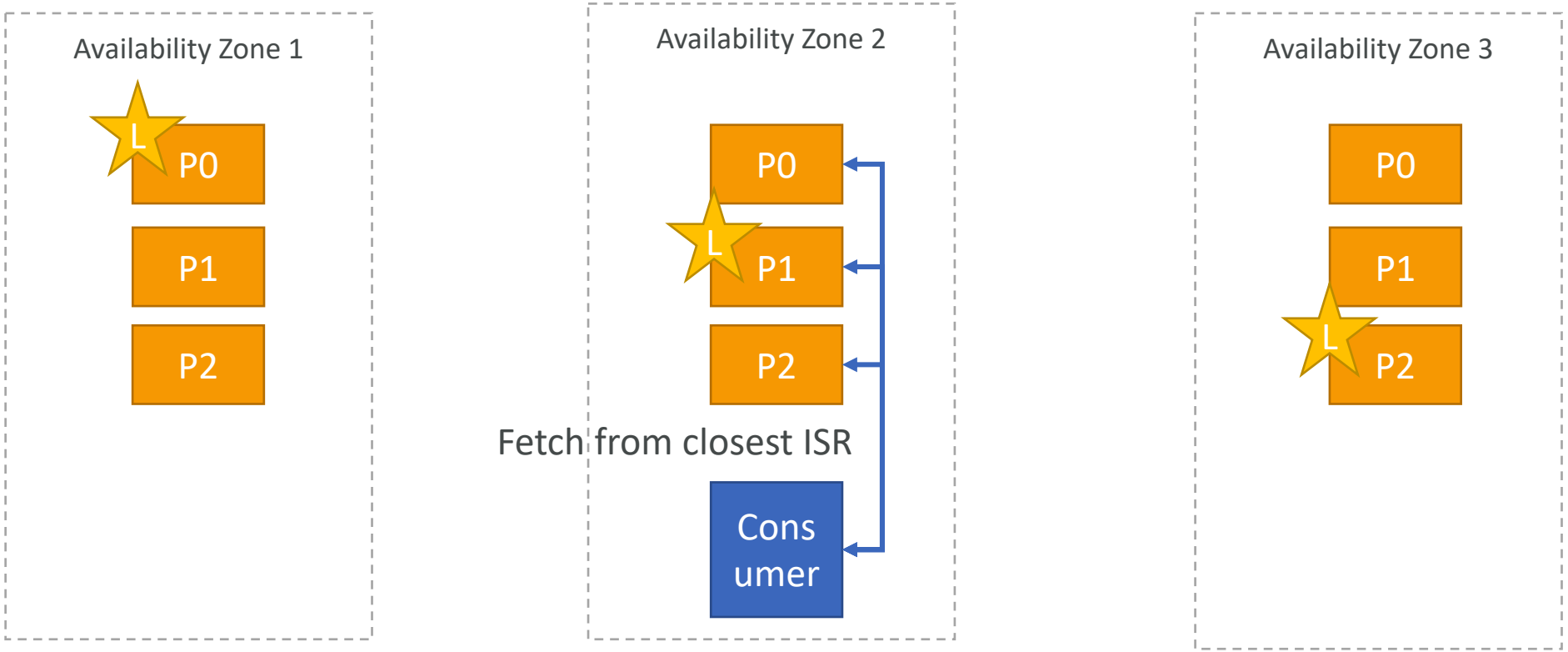
Default Consumer Behavior

- Consumers read from the leader partition
- Higher latency, + Cross AZ charges ($$$)



Consumer Rack Awareness (v 2.4.0+)

- KIP-392: Allow consumers to fetch from closest replica
- Cost savings + lower latency



Consumer Rack Awareness (v 2.4.0+) in Amazon MSK

- Broker setting:
  - Must be version 2.4.0+
  - **rack.id** is automatically set to the AZ ID **(not the AZ name)**
    - **E.g.** NOT us-west-2a but instead **usw2-az1**
    - Find: https://docs.aws.amazon.com/cli/latest/reference/ec2/describe-availability-zones.html
  - **replica.selector.class** must be set to org.apache.kafka.common.replica.RackAwareReplicaSelector

- Consumer client setting:
  - Set **client.rack** to the AZ ID (use the AWS SDK to retrieve it at runtime)

Amazon MSK Custom Configurations

- See all properties you can configure here:
  - https://docs.aws.amazon.com/msk/latest/developerguide/msk-configuration-properties.html
- Applying new properties to Amazon MSK trigger a rolling restart (using best practices, each broker will catch up on missing data before next restart)
- Amazon MSK configurations are versioned
- Override dynamic broker configurations using the kafka-configs CLI
  - https://kafka.apache.org/documentation/#dynamicbrokerconfigs
- Configure topics (if needed) using kafka-configs CLI

Amazon MSK Important Configurations

- auto.create.topics.enable=false
- default.replication.factor (set if above is true)
- num.partitions (set if above is true)
- delete.topic.enable=true (if security not enabled, better if false in prod)
- log.retention.hours=168 (disk space management)
- message.max.bytes (https://stackoverflow.com/questions/21020347/how-can-i-send-large-messages-with-kafka-over-15mb)
- min.insync.replicas=RF-1
- replica.selector.class= org.apache.kafka.common.replica.RackAwareReplicaSelector (useful for Rack-aware consumers)
- unclean.leader.election.enable=true (risk of data loss)

Amazon MSK Security

The need for encryption, authentication & authorization in Kafka

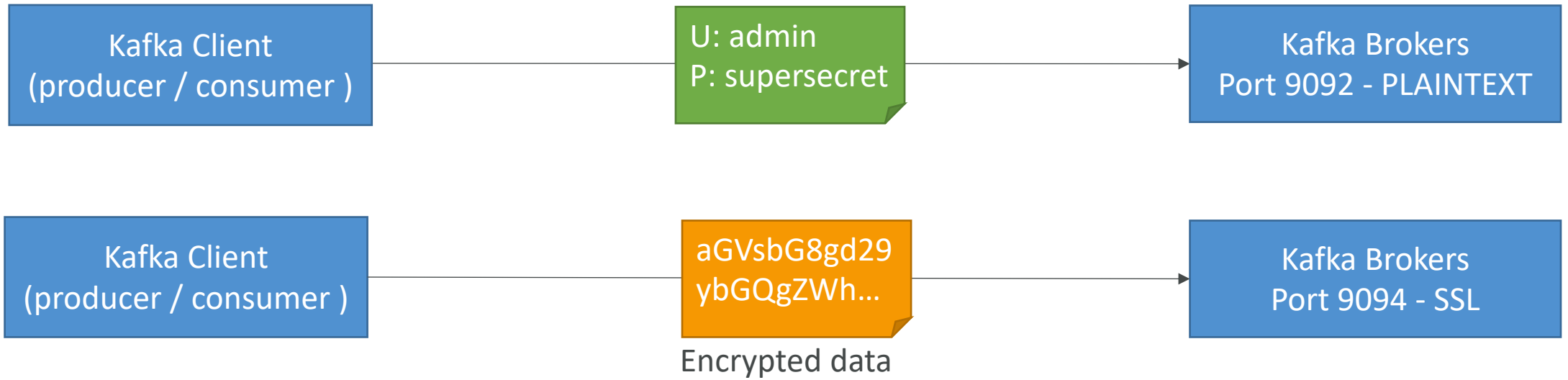
- Currently, any client can access your Kafka cluster (authentication)
- The clients can publish / consume any topic data (authorisation)
- All the data being sent is fully visible on the network (encryption)

- Someone could intercept data being sent
- Someone could publish bad data / steal data
- Someone could delete topics

- All these reasons push for more security and an authentication model

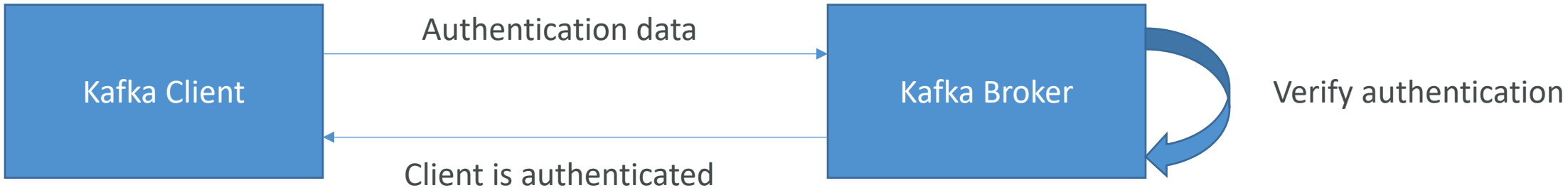
Encryption in Kafka

- Encryption in Kafka ensures that the data exchanged between clients and brokers is secret to routers on the way
- This is similar concept to an https website



Authentication in Kafka

- Authentication in Kafka ensures that only clients that can prove their identity can connect to our Kafka Cluster
- This is similar concept to a login (username / password)

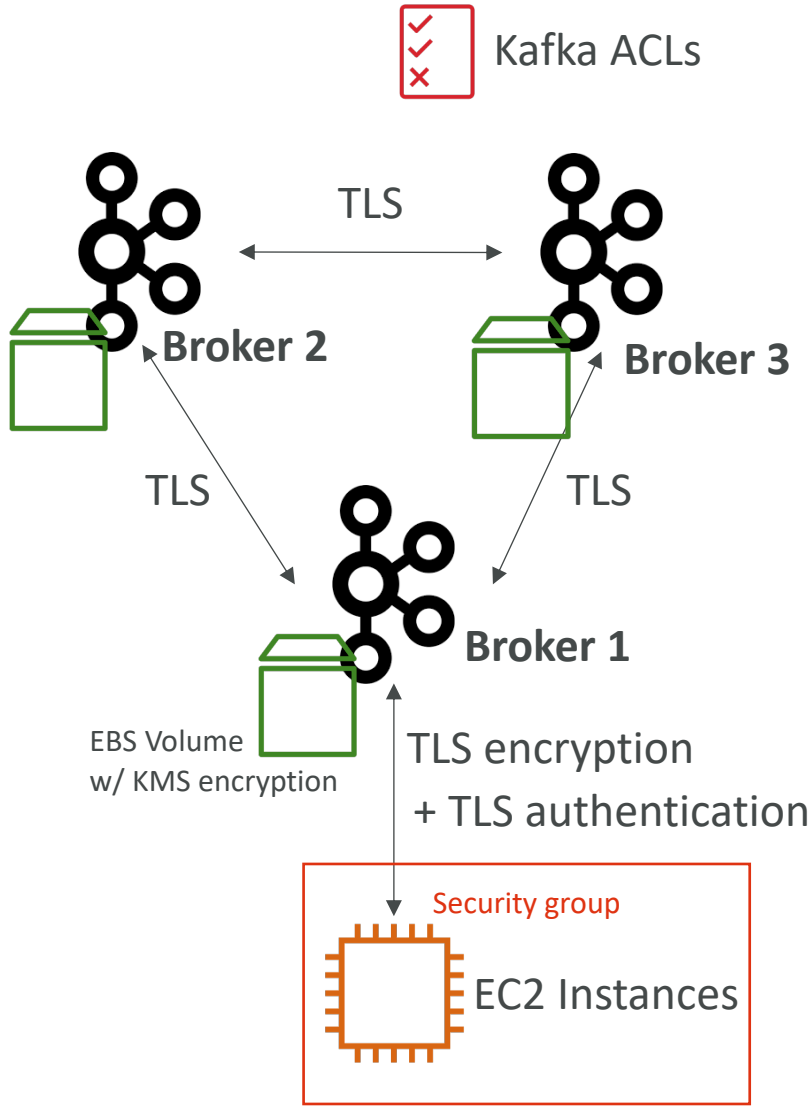


Authorisation in Kafka

- Once a client is authenticated, Kafka can verify its identity
- It still needs to be combined with authorisation, so that Kafka knows that
  - ''User alice can view topic finance''
  - ''User bob cannot view topic trucks''
- ACL (Access Control Lists) have to be maintained by administration and onboard new users

Amazon MSK – Security

- Encryption:
  - In-flight using TLS between the brokers
  - In-flight with TLS between the clients and brokers
  - TLS reduces performance ~30%
  - At rest for your EBS volumes using KMS
- Authentication:
  - Supports TLS client authentication using a Private Certificate Authority (CA) from ACM
- Authorization:
  - Authorize specific security groups for your Apache Kafka clients
  - Security and ACLs for clients is done within the Apache Kafka cluster



Amazon MSK – TLS Encryption In Flight

- Hands on!

Amazon MSK – TLS Authentication

- Integration with ACM – Private CA - $400 USD / month – 30 day trial

AWS Certificate Manager Private Certificate Authority Pricing

ACM Private Certificate Authority (CA) is priced along two dimensions. You pay a monthly fee for the operation of each private CA until you delete it and you pay for the private certificates you issue each month.

PRIVATE CERTIFICATE AUTHORITY OPERATION

- $400.00 per month for each ACM private CA until you delete the CA.
- ACM Private CA operation is pro-rated for partial months based on when you create and delete the CA. You are not charged for a private CA after you delete it. However, if you restore a deleted CA, you are charged for the time between deleting it and restoring it.

PRIVATE CERTIFICATES

You pay a one-time fee when you issue certificates from ACM Private CA for which you have access to the private key. This includes private certificates you create in ACM and export and certificates for which you create the private key yourself

ACM Private CA 30-day Free Trial

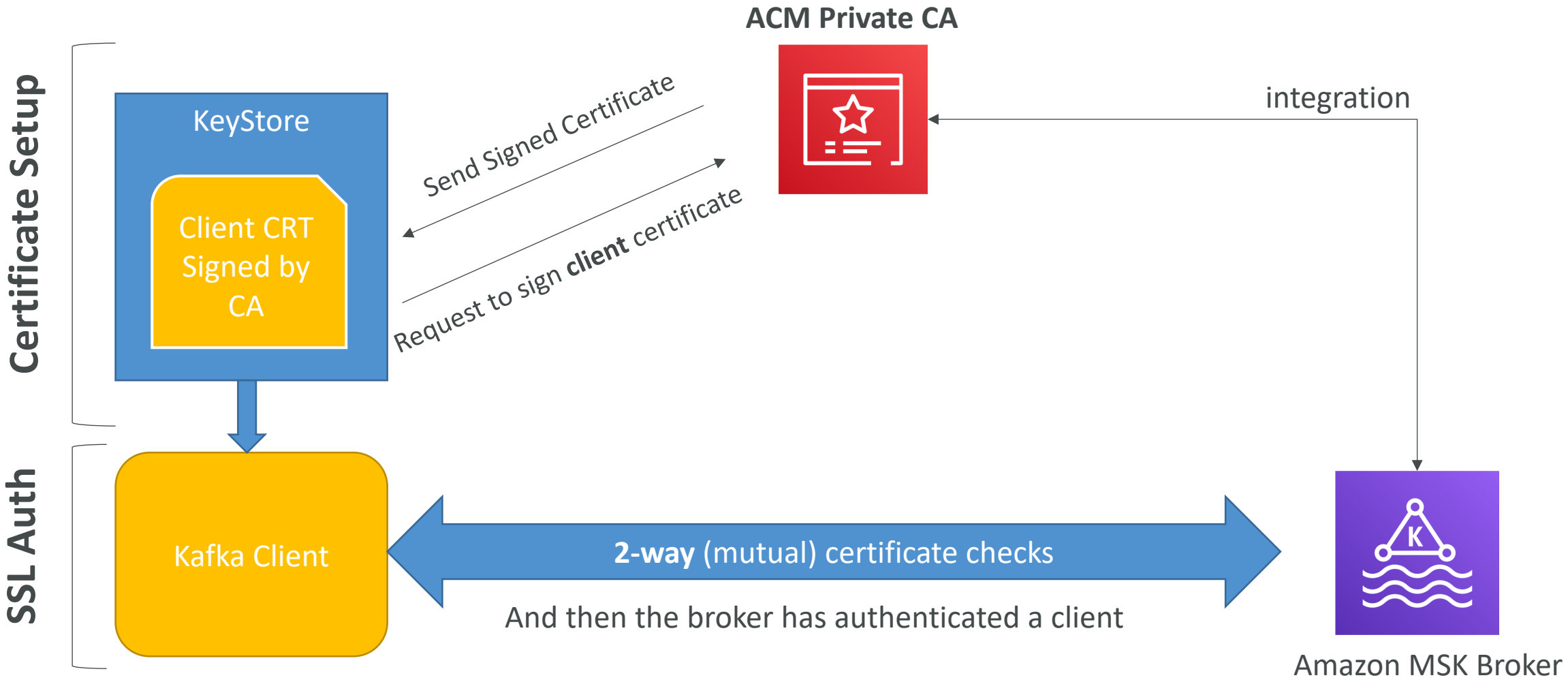
30-day trial

Any new account to ACM Private CA can try the service for 30 days with no charge for operation of the first private CA created in the account. You pay for the certificates you issue during the trial period.

Start a free trial of ACM Private CA

Number of private certificates created in the month/per Region	Price (per certificate)
1 - 1,000	$0.75
1,001 - 10,000	$0.35
10,001 and above	$0.001

Amazon MSK – TLS Authentication – Mutual Auth



ACL Management

- Use the kafka-acls command
- Documentation from here: https://docs.aws.amazon.com/msk/latest/developerguide/msk-acls.html
- Tools that can help with Kafka ACL Management:
  - AKHQ (open source)
  - Conduktor
  - Kafka Security Manager (open source)
- Important: allow.everyone.if.no.acl.found=true by default
  - Then add ACLs for a topic and a client ID
- Create one certificate (user identity) per client application

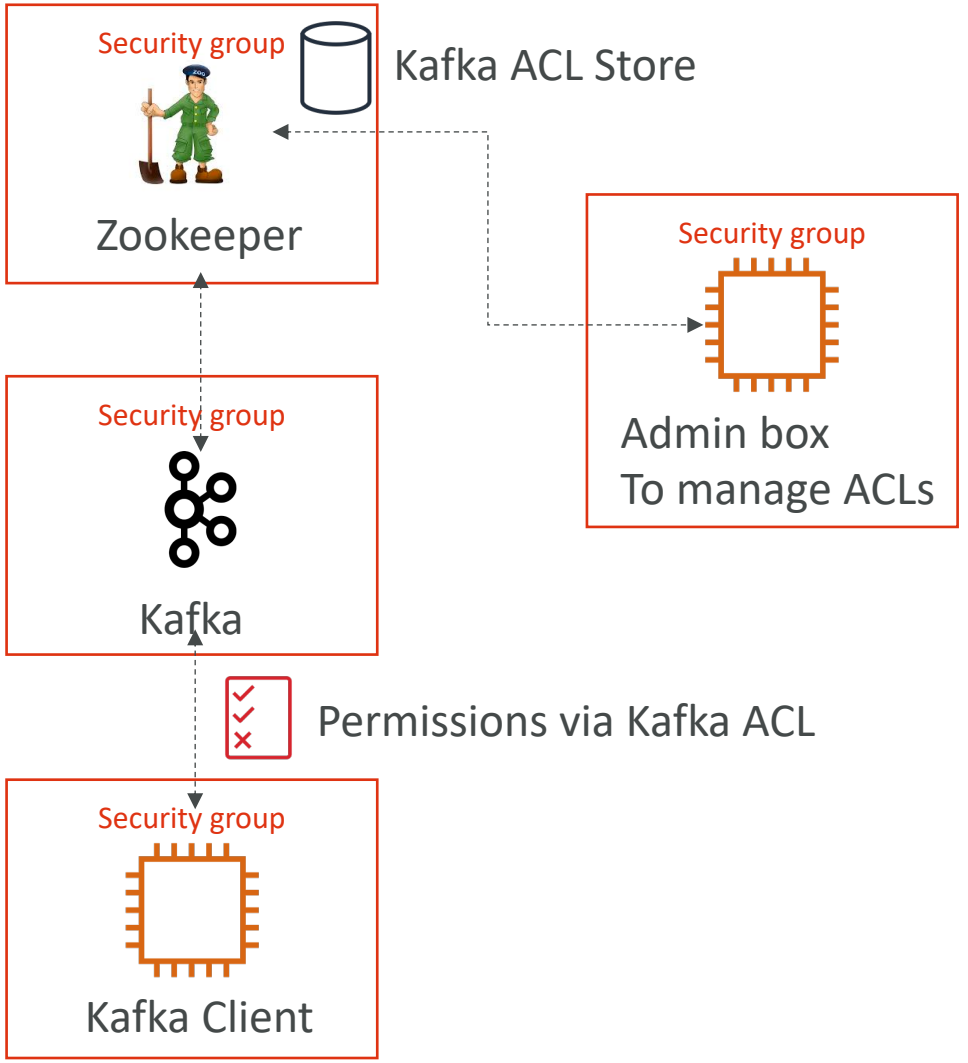
Zookeeper Security

- Can only do network based security
- Create a separate security group for Zookeeper
- Assign it to the ENI of Zookeeper manually
- Ensure the new security group has rules allowing the Kafka Security Group (and the necessary applications) in.

- https://docs.aws.amazon.com/msk/latest/developerguide/zookeeper-security.html

Secure Amazon MSK setup

- Separate Zookeeper Security Group (ACLs are added using the kafka-acls command pointing to Zookeeper)
- Only allow TLS mutual authentication into the cluster
- Create ACL for every topic
- Create one certificate per client application
- Use Security Groups wisely to allow communications into Kafka



Amazon MSK Monitoring

Amazon MSK – Monitoring Overview

- CloudWatch Metrics
  - Basic monitoring (cluster and broker metrics)
  - Enhanced monitoring (++enhanced broker metrics)
  - Topic-level monitoring (++enhanced topic-level metrics)
- Prometheus (Open-Source Monitoring)
  - Opens a port on the broker to export cluster, broker and topic-level metrics
  - Setup the JMX Exporter (metrics) or Node Exporter (CPU and disk metrics)
- Broker Log Delivery
  - Delivery to CloudWatch Logs
  - Delivery to Amazon S3
  - Delivery to Kinesis Data Firehose

Amazon MSK Broker Logs

- Can be edited after cluster creation
- Delivery to CloudWatch Logs
  - Useful for real-time log analytics
  - Can setup metric filters
  - Search through authorization exceptions
- Delivery to Amazon S3
  - Long-term archival of logs
- Delivery to Kinesis Data Firehose
  - Useful if you want to send logs to Splunk / Elastic Search

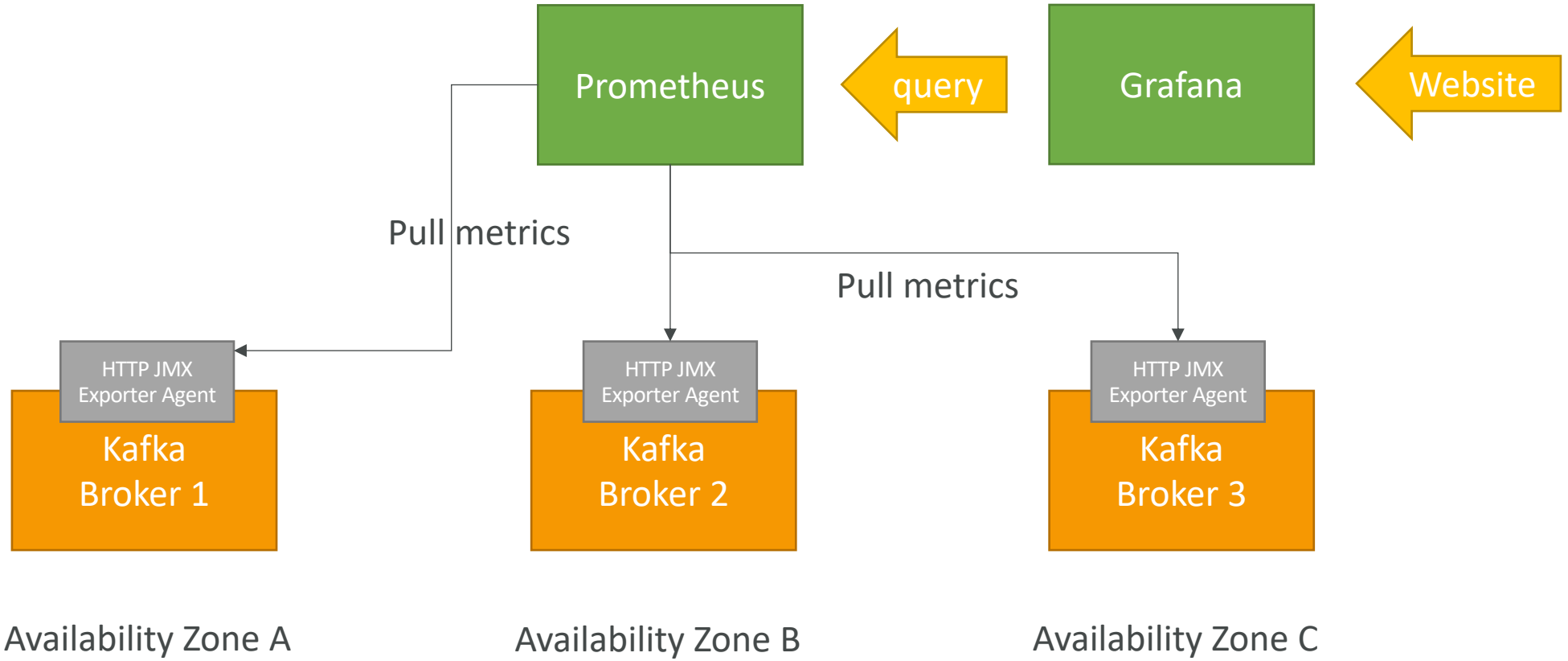
Amazon MSK - Important Metrics

- Full list of metrics here: https://docs.aws.amazon.com/msk/latest/developerguide/monitoring.html#monitoring-level
- DEFAULT (free)
  - NetworkRxErrors
  - OfflinePartitionsCount
  - MemoryFree
  - RootDiskUsed
- PER_BROKER ($)
  - UnderReplicatedPartitions
  - UnderMinIsrPartitionCount
  - MessagesInPerSec
- PER_TOPIC_PER_BROKER ($$)
  - BytesInPerSec
  - BytesOutPerSec
  - MessagesInPerSec
- Recommendation: at least per broker, preferred per topic per broker

Amazon MSK – Open Monitoring

- Tools that are compatible with Prometheus-formatted metrics
- Tools that integrate with Amazon MSK Open Monitoring
  - Datadog
  - Lenses
  - New Relic
  - Sumo logic
- Available for free but charges applied for transfer of data across AZs.
- Open monitoring can be enabled at creation of Amazon MSK cluster or edited for an existing Amazon MSK cluster
- JMX Exporter => for Kafka JMX metrics
- Node Exporter => for node information (CPU, Disk)

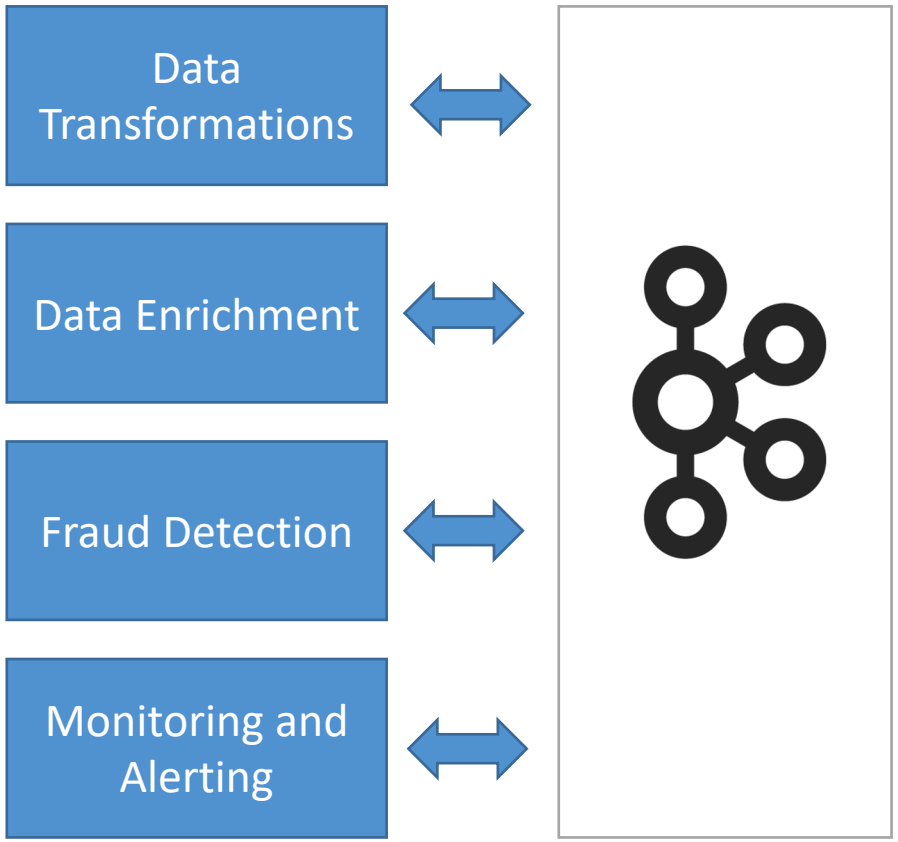
Setting up Prometheus + Grafana
Target Architecture



Stream Processing

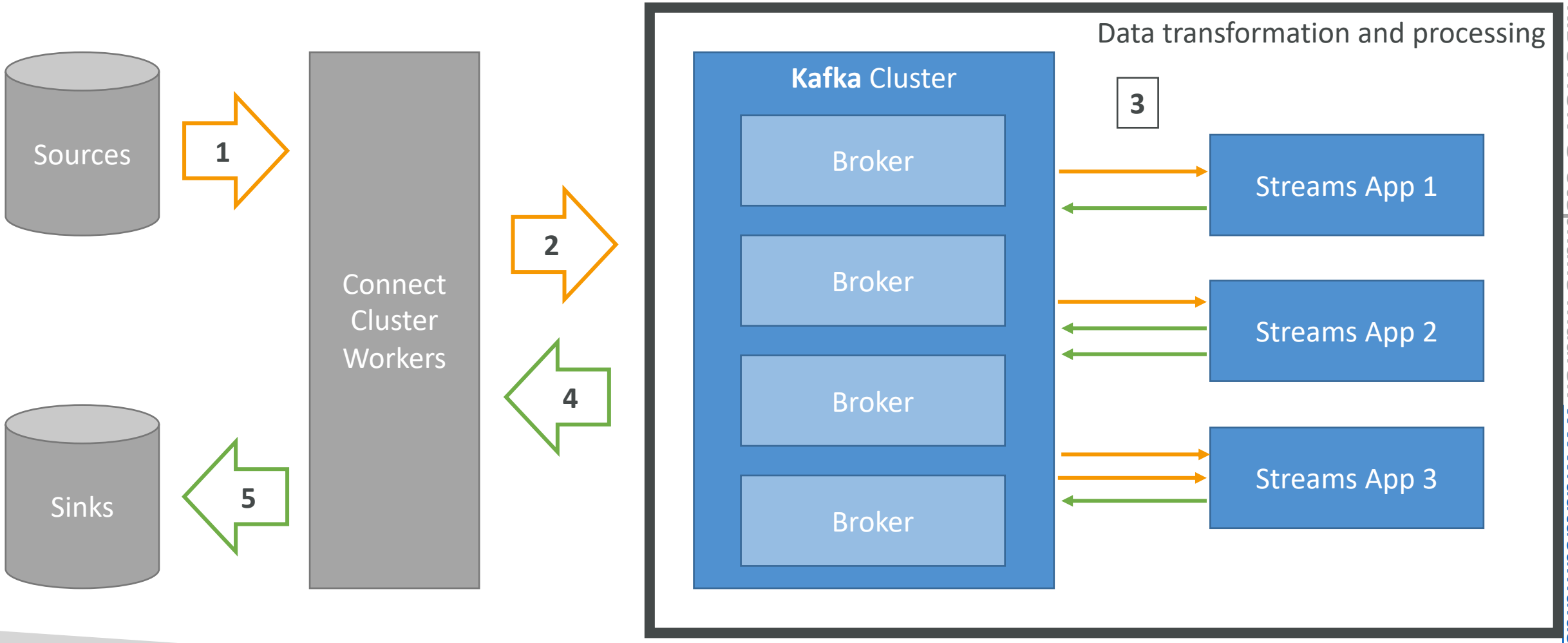
What is Kafka Streams ?

- Easy **data processing and transformation library** within Kafka



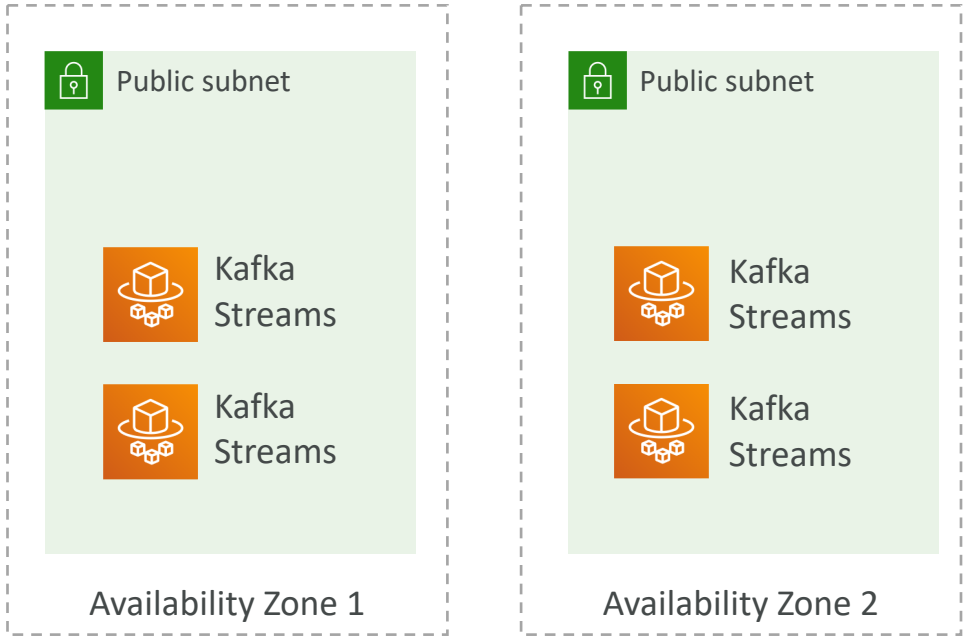
- Standard Java Application
- No need to create a separate cluster
- Highly scalable, elastic and fault tolerant
- Exactly Once Capabilities
- One record at a time processing (no batching)
- Works for any application size

Kafka Streams Architecture Design



Deploying Kafka Streams with ECS

- Simple WordCount application
- Using Fargate
- Kafka Streams application can scale based on the number of source partitions and the application topology
- We can scale our wordcount from 1 to 4 tasks for example!
- See next lecture on Kafka Streams guidance for production deployments



Kafka Streams in Production in AWS

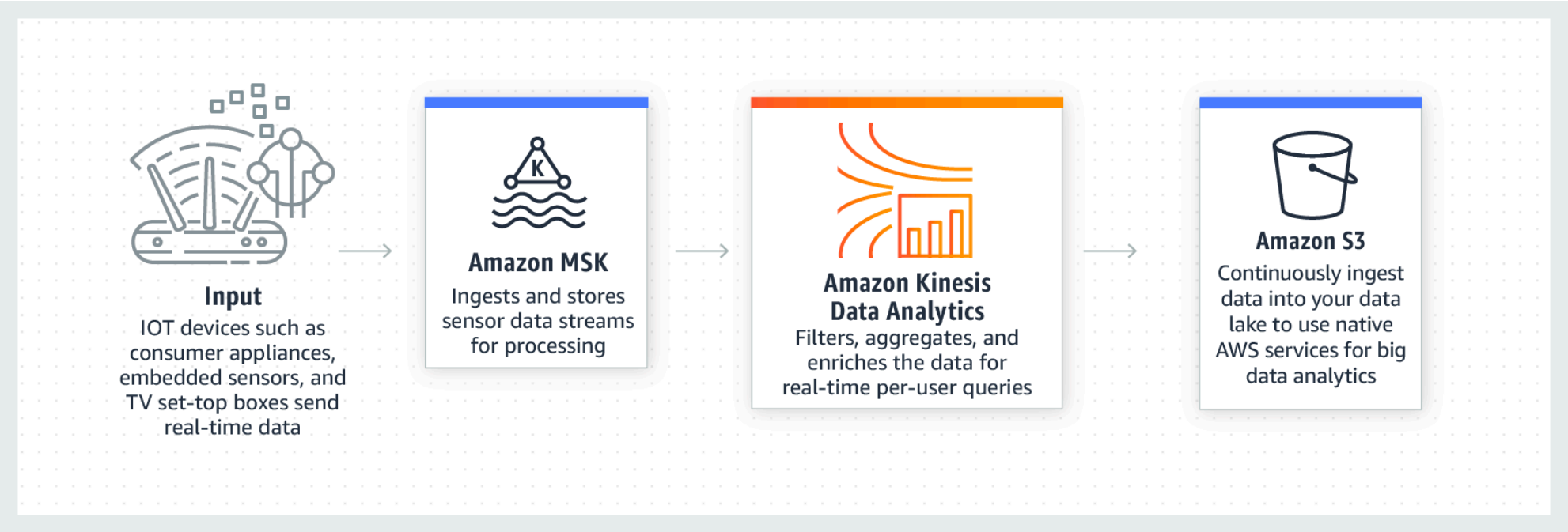
- It all depends on your Kafka Streams application (stateless, stateful, size…)
- Fargate:
  - each task is limited to 20GB of storage space
  - Could try to mount & use EFS to externalize the "state.dir" directory
- EC2 + EBS:
  - Must manage EC2 instances on your own
  - State can live on EBS volumes (cheaper)
  - Could try to deploy as a .jar file on Elastic Beanstalk (worker mode)
- EKS:
  - Best practice for Kafka Streams (requires Kubernetes knowledge)
  - Leverage stateful sets, state will be persistent in EBS volumes

Kafka Streams – extra resources

- Learning Kafka Streams:
  - Course https://links.datacumulus.com/kafka-streams-coupon
  - Documentation: https://kafka.apache.org/documentation/streams/
  - Confluent Tutorials: https://kafka-tutorials.confluent.io/
- Kafka Summit talks:
  - Kafka Streams and Tables all the way down
  - The art of event streaming application
  - Kafka Streams at Scale
  - Deep Dive into Kafka Streams
  - Deploying Kafka Streams Applications with Docker and Kubernetes

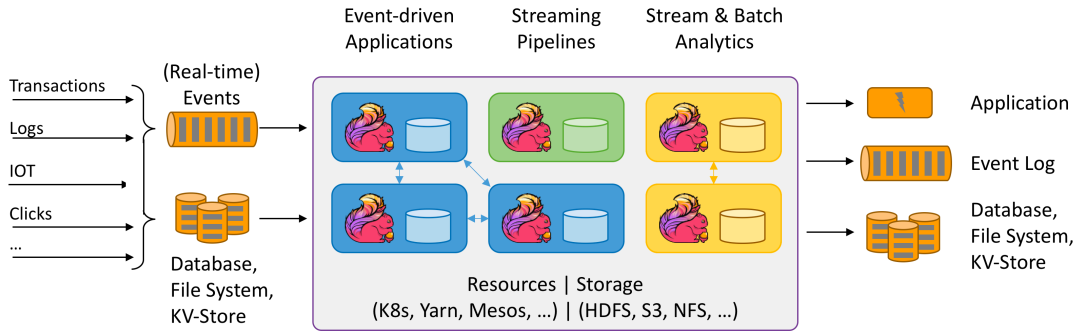
Kinesis Data Analytics for Apache Flink on Amazon MSK

- As the name **does not** indicate, can be used to run Java Apache Flink applications on Apache Kafka (MSK) and Kinesis Data Streams (KDS)



Apache Flink https://flink.apache.org/

- Stateful computations over data streams (batch or real-time)
- Serious alternative to Apache Spark



All streaming use cases

- Event-driven Applications
- Stream & Batch Analytics
- Data Pipelines & ETL

Learn more

Guaranteed correctness

- Exactly-once state consistency
- Event-time processing
- Sophisticated late data handling

Learn more

Layered APIs

- SQL on Stream & Batch Data
- DataStream API & DataSet API
- ProcessFunction (Time & State)

Learn more

Operational Focus

- Flexible deployment
- High-availability setup
- Savepoints

Learn more

Scales to any use case

- Scale-out architecture
- Support for very large state
- Incremental checkpointing

Learn more

Excellent Performance

- Low latency
- High throughput
- In-Memory computing

Learn more

Kinesis Data Analytics for Apache Flink

- Uses Apache Flink: https://flink.apache.org/
- Managed service: just provide your .jar file, AWS will run it
- Compatible with Apache Kafka and Kinesis
- Deploy within your VPC
- Lab at: https://amazonmsk-labs.workshop.aws/en/mskkdaflinklab.html