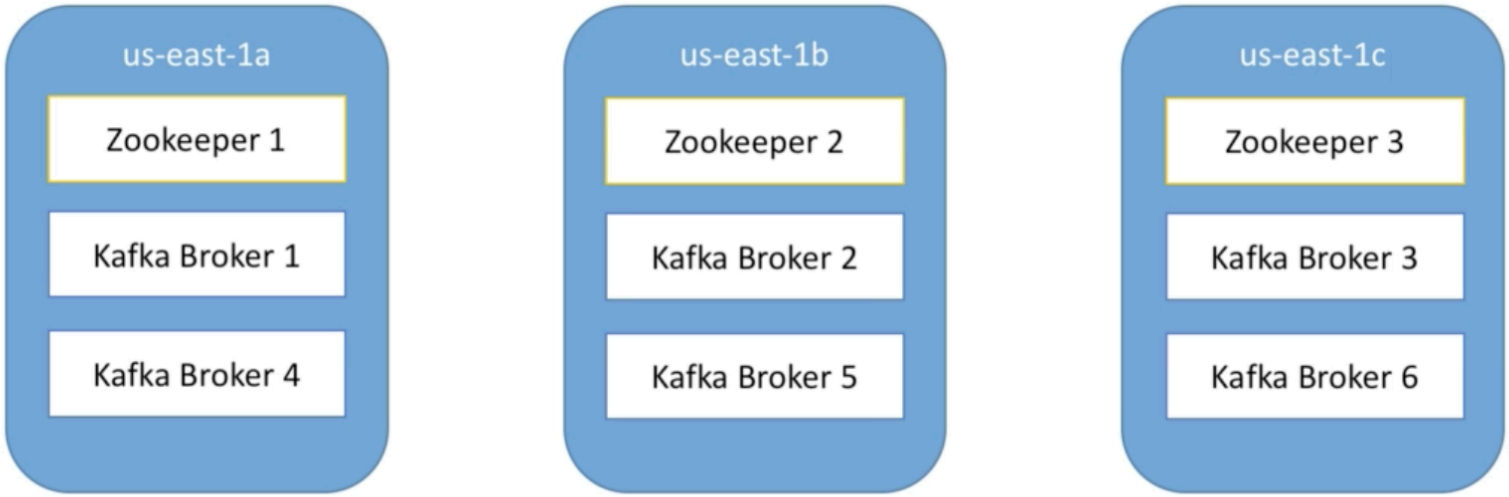
Kafka Cluster Setup
High Level Architecture

- You want multiple brokers in different data centers (racks) to distribute your load. You also want a cluster of at least 3 zookeeper
- In AWS:



Kafka Cluster Setup Gotchas

- It's not easy to setup a cluster
- You want to isolate each Zookeeper & Broker on separate servers
- Monitoring needs to be implemented
- Operations have to be mastered
- You need a really good Kafka Admin

- Alternative: many different "Kafka as a Service" offerings on the web
- No operational burdens (updates, monitoring, setup, etc...)

Kafka Monitoring and Operations

- Kafka exposes metrics through JMX.
- These metrics are highly important for monitoring Kafka, and ensuring the systems are behaving correctly under load.
- Common places to host the Kafka metrics:
  - ELK (ElasticSearch + Kibana)
  - Datadog
  - NewRelic
  - Confluent Control Centre
  - Promotheus
  - Many others…!

Kafka Monitoring and Operations

- Some of the most important metrics are:
- Under Replicated Partitions: Number of partitions are have problems with the ISR (in-sync replicas). May indicate a high load on the system
- Request Handlers: utilization of threads for IO, network, etc… overall utilization of an Apache Kafka broker.
- Request timing: how long it takes to reply to requests. Lower is better, as latency will be improved.

Kafka Monitoring and Operations

- Overall have a look at the documentation here:
- https://kafka.apache.org/documentation/#monitoring
- https://docs.confluent.io/current/kafka/monitoring.html
- https://www.datadoghq.com/blog/monitoring-kafka-performance-metrics/

可用资源

Kafka Monitoring and Operations

- Kafka Operations team must be able to perform the following tasks:
  - Rolling Restart of Brokers
  - Updating Configurations
  - Rebalancing Partitions
  - Increasing replication factor
  - Adding a Broker
  - Replacing a Broker
  - Removing a Broker
  - Upgrading a Kafka Cluster with zero downtime

The need for encryption, authentication & authorization in Kafka

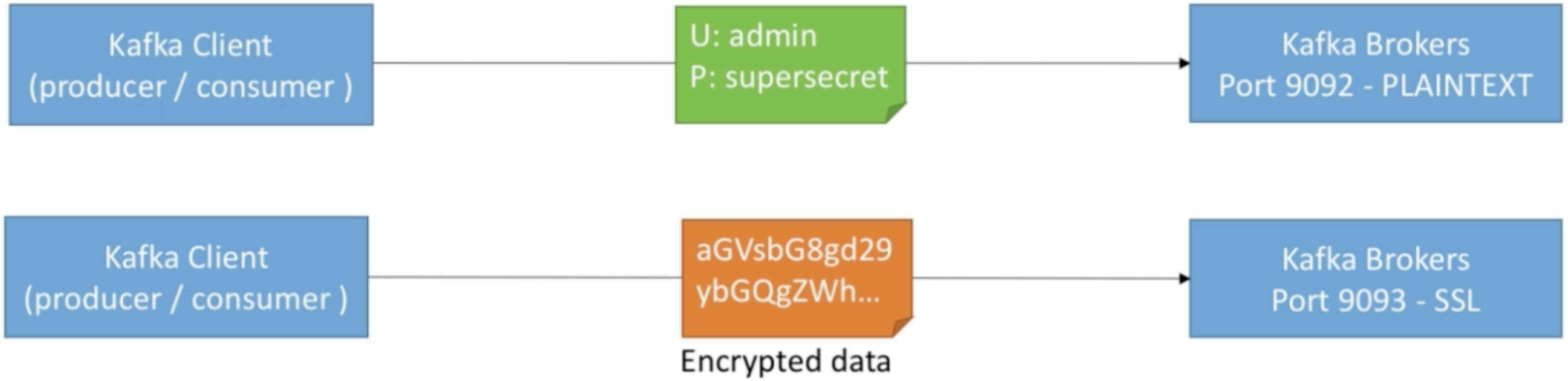
- Currently, any client can access your Kafka cluster (authentication)
- The clients can publish / consume any topic data (authorisation)
- All the data being sent is fully visible on the network (encryption)

- 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

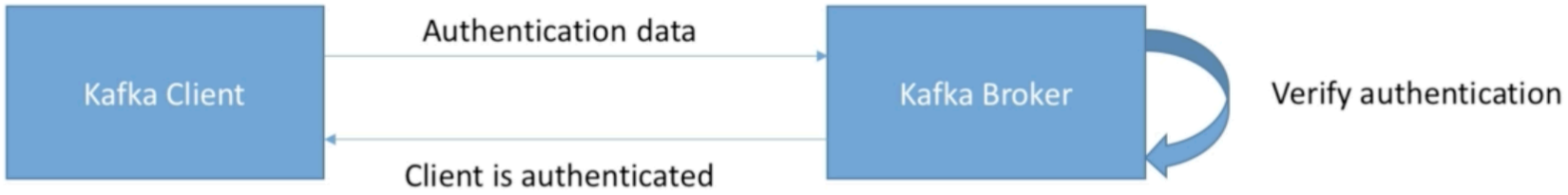
Encryption in Kafka

- 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)



Authentication in Kafka

- Authentication in Kafka can take a few forms
- SSL Authentication: clients authenticate to Kafka using SSL certificates
- SASL Authentication:
  - PLAIN: clients authenticate using username / password (weak – easy to setup)
  - Kerberos: such as Microsoft Active Directory (strong – hard to setup)
  - SCRAM: username / password (strong – medium to setup)

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

Putting it all together

- You can mix
  - Encryption
  - Authentication
  - Authorisation

- This allows your Kafka clients to:
  - Communicate securely to Kafka
  - Clients would authenticate against Kafka
  - Kafka can authorise clients to read / write to topics

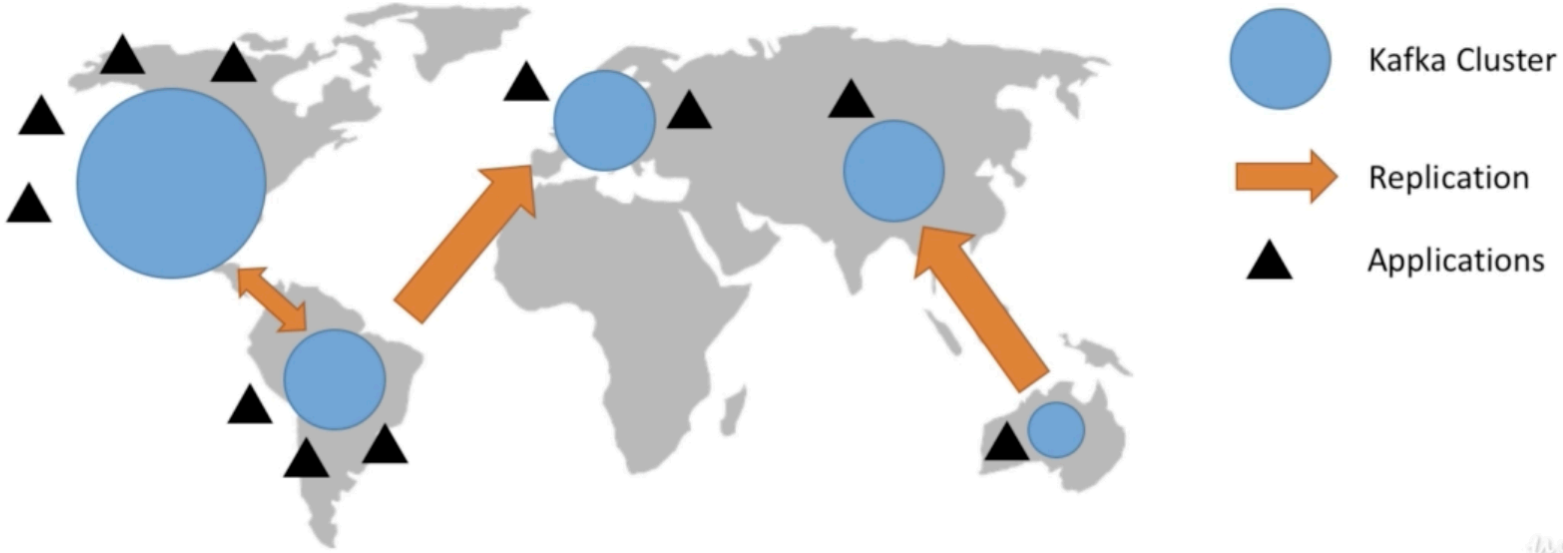
State of the art of Kafka Security

- Kafka Security is fairly new (0.10)
- Kafka Security improves over time and becomes more flexible / easier to setup as time goes.
- Currently, it is **hard** to setup Kafka Security.
- Best support for Kafka Security for applications is with **Java**.

可用资源

Kafka Multi Cluster + Replication

- Kafka can only operate well in a single region
- Therefore, it is very common for enterprises to have Kafka clusters across the world, with some level of replication between them



可用资源

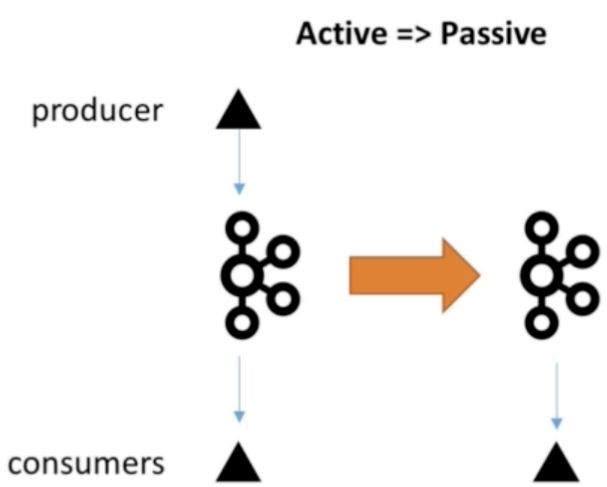
Kafka Multi Cluster + Replication

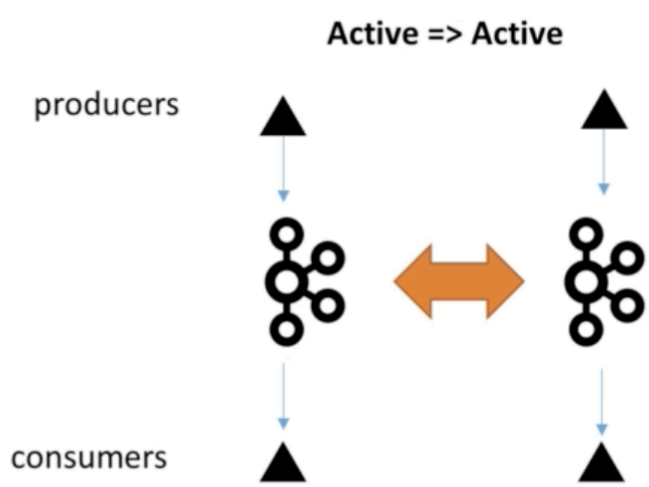
- A replication application at its core is just a consumer + a producer
- There are different tools to perform it:
  - Mirror Maker - open source tool that ships with Kafka
  - Netflix uses Flink – they wrote their own application
  - Uber uses uReplicator – addresses performance and operations issues with MM
  - Comcast has their own open source Kafka Connect Source
  - Confluent has their own Kafka Connect Source (paid)
- Overall, try these and see if it works for your use case before writing your own

可用资源

Kafka Multi Cluster + Replication

- There are two designs for cluster replication:





Kafka Multi Cluster + Replication

- Active => Active:
  - You have a global application
  - You have a global dataset
- Active => Passive:
  - You want to have an aggregation cluster (for example for analytics)
  - You want to create some form of disaster recovery strategy (*it's hard)
  - Cloud Migration (from on-premise cluster to Cloud cluster)
- **Replicating doesn't preserve offsets, just data!**

可用资源

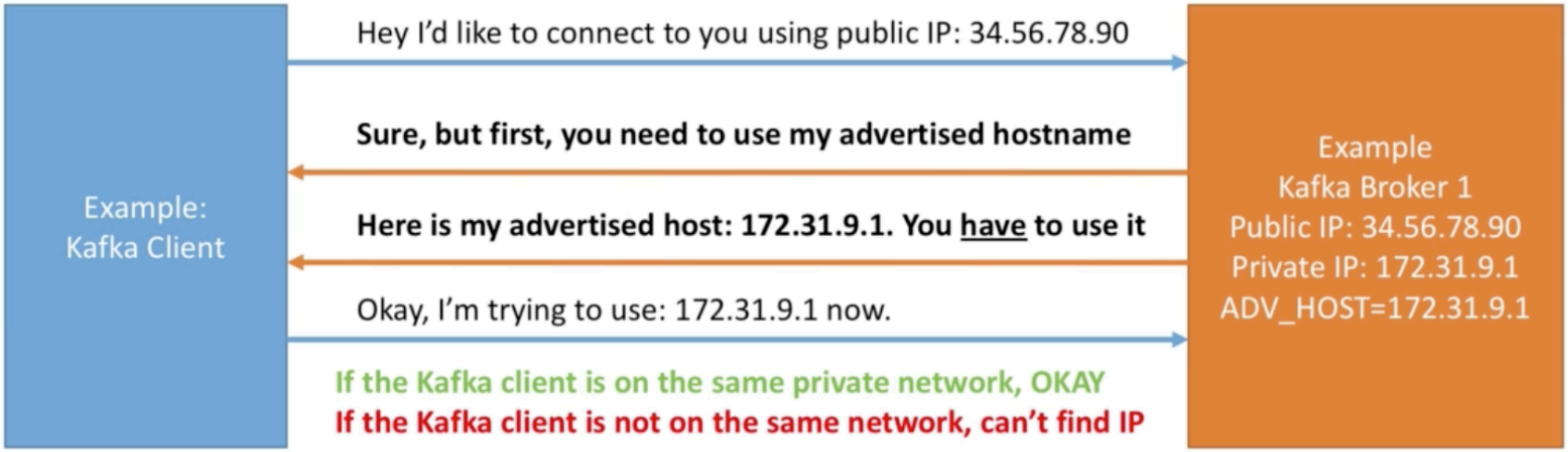
Kafka Multi Cluster + Replication

- Optimization resources online:

- https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=27846330
- https://engineering.salesforce.com/mirrormaker-performance-tuning-63afaed12c21
- https://docs.confluent.io/current/multi-dc/replicator-tuning.html#improving-network-utilization-of-a-connect-task
- https://community.hortonworks.com/articles/79891/kafka-mirror-maker-best-practices.html
- https://www.confluent.io/kafka-summit-sf17/multitenant-multicluster-and-hieracrchical-kafka-messaging-service
- https://eng.uber.com/ureplicator/
- https://www.altoros.com/blog/multi-cluster-deployment-options-for-apache-kafka-pros-and-cons/
- https://github.com/Comcast/MirrorTool-for-Kafka-Connect

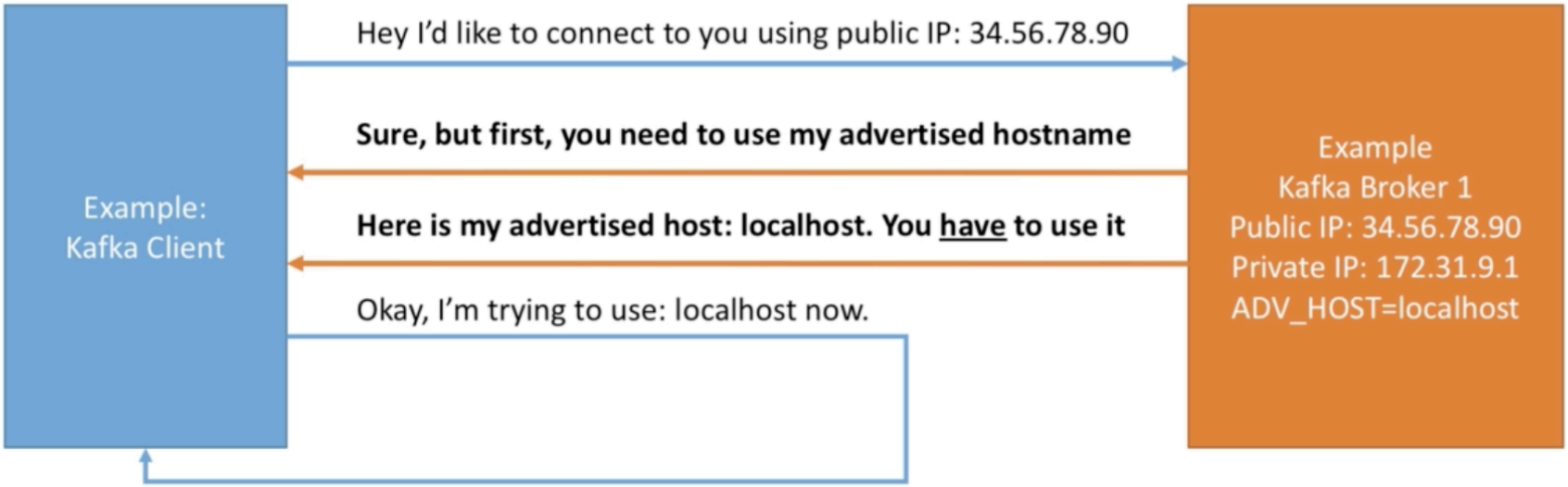
Understanding communications between Client and with Kafka

- **Advertised listeners is <u>the most important</u> setting of Kafka**



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

- **Advertised hostname is <u>the most important</u> setting of Kafka**

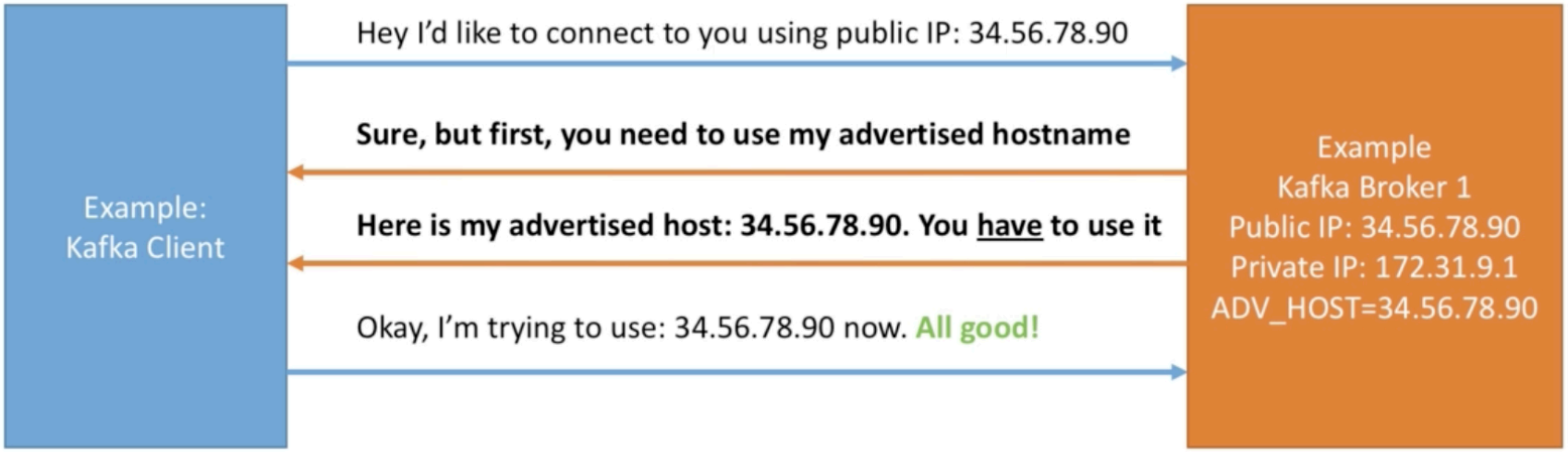


If the Kafka client is on the machine and you have 1 broker, OKAY

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

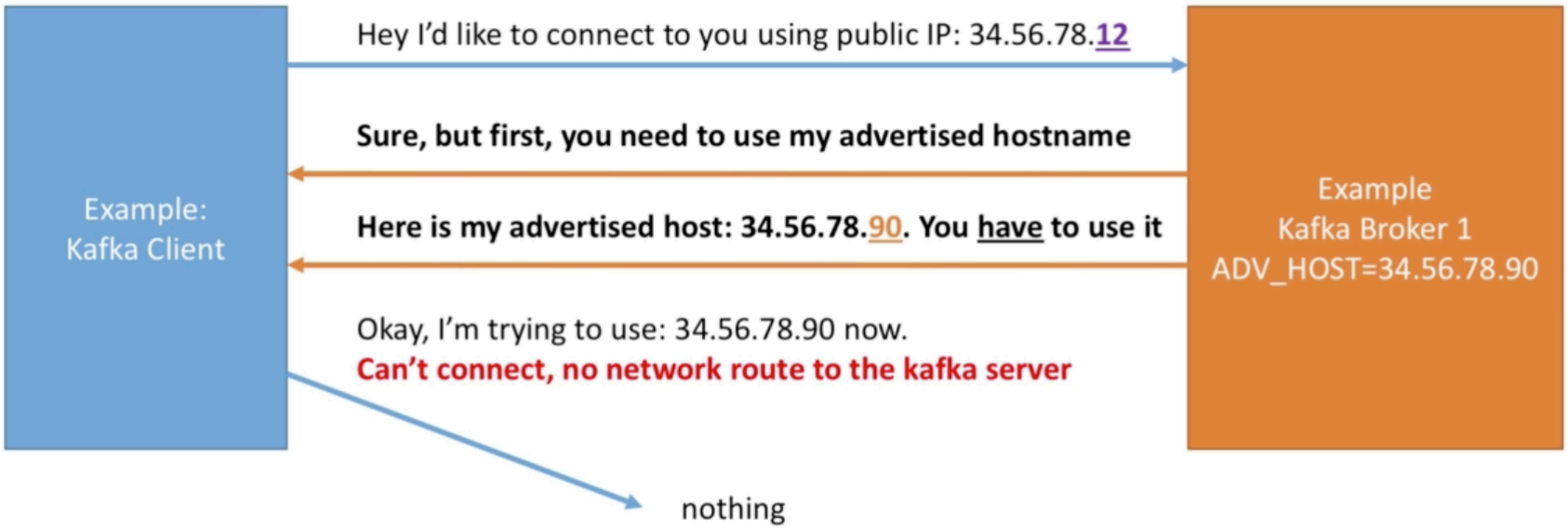
What if I use the public IP?

- **Advertised hostname is <u>the most important</u> 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



So what do I set for advertised.listeners?

- If your clients are on your private network, set either:
  - the internal private IP
  - the internal private DNS hostname
- Your clients should be able to resolve the internal IP or hostname

- If your clients are on a public network, set either:
  - The external public IP
  - The external public DNS hostname pointing to the public IP
- Your clients must be able to resolve the public DNS