Kafka Admin

Lesson Objectives

- Kafka Admin

Installing Kafka

Install Process

- Prepare hardware
- Prepare basic software stack (OS / JDK ..etc)
- Install and configure zookeeper
- Install Kafka on all machines
- Configure each node / broker
- Setup Kafka for auto-start on bootup
- Setup monitoring

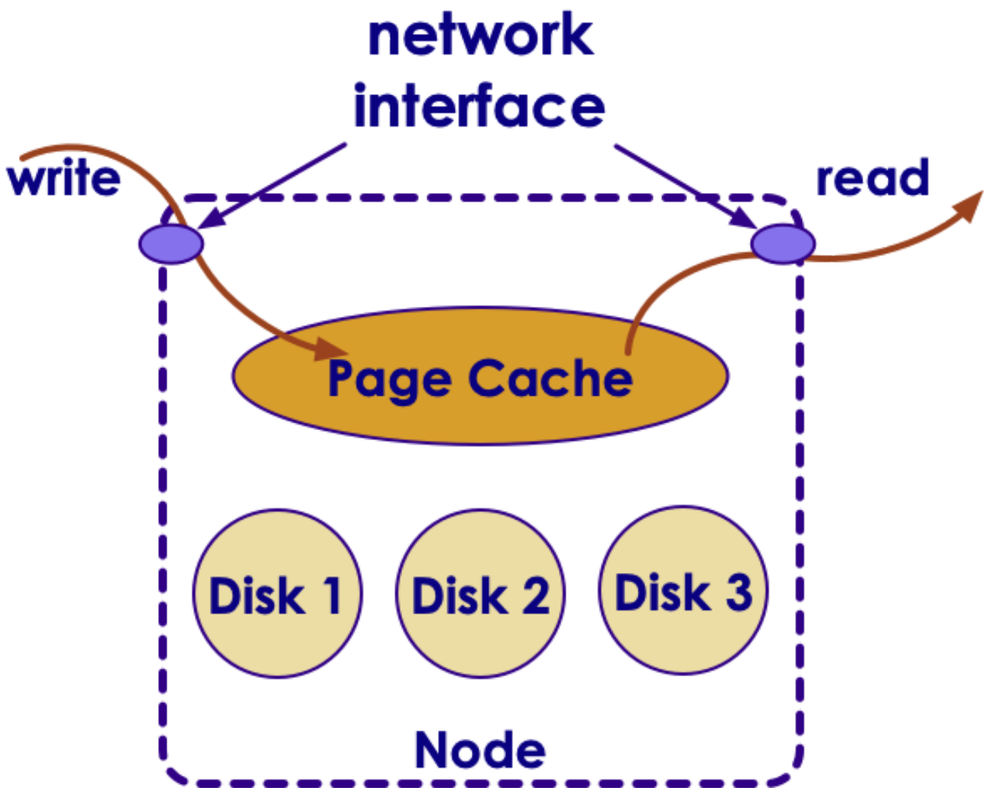
Kafka Hardware Requirements

- Bare metal machines for high performance
- Virtual Machines (VM) are not recommended in production setup

	CPU	Memory	Disk	Network
Modest	8 cores	32 G	4 x SATA 7200 RPM	2 x 1 Gig (bonded)
High	24 cores	64 G - 128 G	8 x SATA 7200 RPM	10 Gig

Kafka Hardware - Memory

- Kafka uses very modest memory by careful heap management (~ 4-8 G per broker)
- The rest of the memory is for page cache
    - Linux would allocate free memory to page cache
- Page cache buffers disk writes / reads
    - This helps with IO throughput (data seldom hits the disk between write and read)



- Good to have sufficient memory to buffer active reader/writers.
    - 30 second buffer is a good place to start
    - Quick calculation = Page cache size = write_throughput x 30 seconds
    - if write throughput is 10MB sec, page cache is
      = 10 MB/s x 30 secs
      = 300 MB

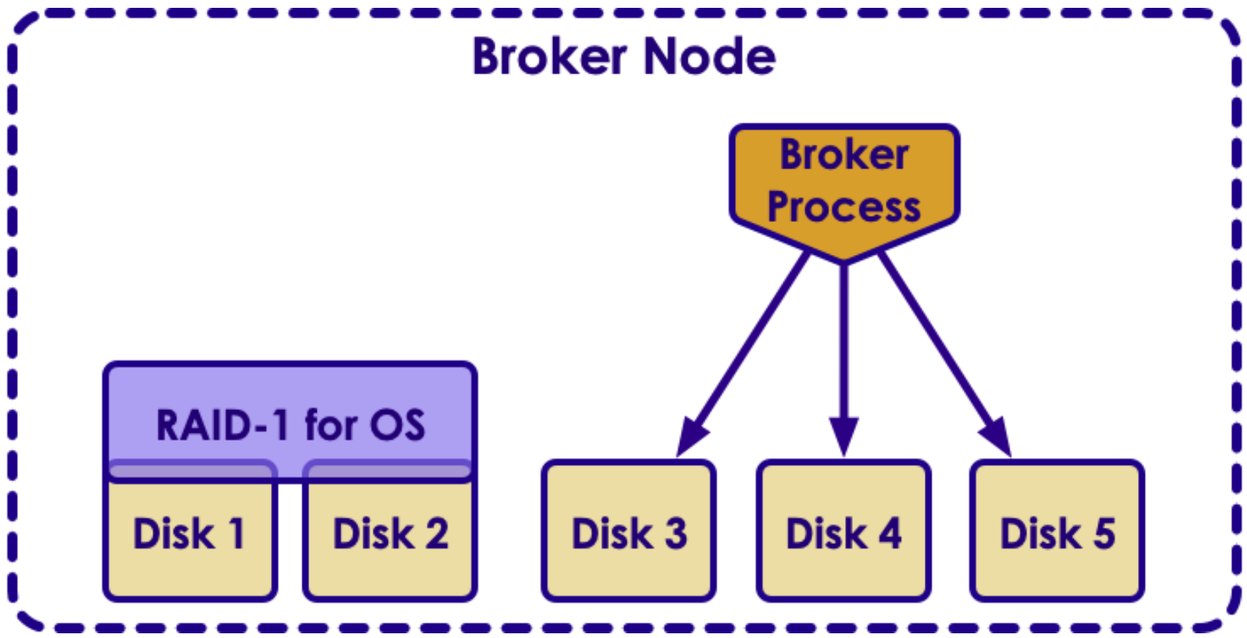
Kafka Hardware - CPUs

- Kafka has modest CPU requirements
- If using encryption, need significantly more CPU power
- More cores --> better
    - Typical setup is 12 cores +
- Cores matter more than raw clock speed (2GHz, 3GHz ..etc)
- More cores will give much better scalability/performance than slightly faster CPU

	CPU	Memory	Disk	Network
Modest	8 cores	32 G	4 x SATA 7200 RPM	2 x 1 Gig (bonded)
High	24 cores	64 G - 128 G	8 x SATA 7200 RPM	10 Gig

Kafka Hardware - Disks

- More disks increase IO throughput
- Don't share Kafka disks with OS disk (minimize contention)
- Disks can be combined by RAID or used as individual volumes
- RAID
    - Better data spread across disks
    - battery backup a must!
    - Might slow down writes
- Individual volumes (better for most scenarios)
    - Kafka will stripe data across disks
    - One partition MUST fit on ONE drive
- **Avoid network attached storage (NAS)**
  They are usually slower, exhibit high latencies, and single point of failure



Zookeeper Hardware

- ZK have very modest hardware requirements
- Run ZK on separate machines
  - Just run ZK, nothing else
  - **Do not co-locate ZK and Kafka on same machines**
    They have different IO access patterns
- Run ZK in odd numbers 3,5,7... .
  - 3 is minimum
  - 5 can work with 2 ZK nodes down
- One ZK ensemble per Kafka cluster per data center
  - To reduce latency

Apache ZooKeeper™

ZK Hardware	CPU	Memory	Disk	Network
	4 cores	32 G	1 -2 drives	2 x 1 Gig (bonded)

Software Requirements

- Linux OS
  - Most used for deployment
- Java 8
  - Dev kit required for for programming
- Zookeeper
  - ZK 3.4.x is stable and well tested with Kafka

Broker Configuration

```
# config file : kafka/config/server.properties

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=0

# A comma separated list of directories under which to store log files.
# Kafka will balance data across multiple volumes
log.dirs=/data1/kafka,/data2/kafka

# port to listen, default 9042
port=9042

# Zookeeper connection string
# This is a comma separated host:port pairs,
zookeeper.connect=zk_server1:2181,zk_server2:2181

# Create topics automatically when producer / consumer uses it?  (default true)
auto.create.topics.enable=true
```

Topic Configuration

- Specified when topic is created by 'kafka-topics.sh'.Can be altered later.
- Num_partitions: (default 1)
  - Partitions will spread across brokers
  - More partitions -> more scalability
  - Partition count can be increased later, but can NOT be decreased!
- Log.retention.ms: (default one week)
  - How long to keep a message
- Log.retention.bytes
  - Set max size of messages per partition
  - If topic has 10 partitions and log retention size is set 1G, overall topic can have 10G total

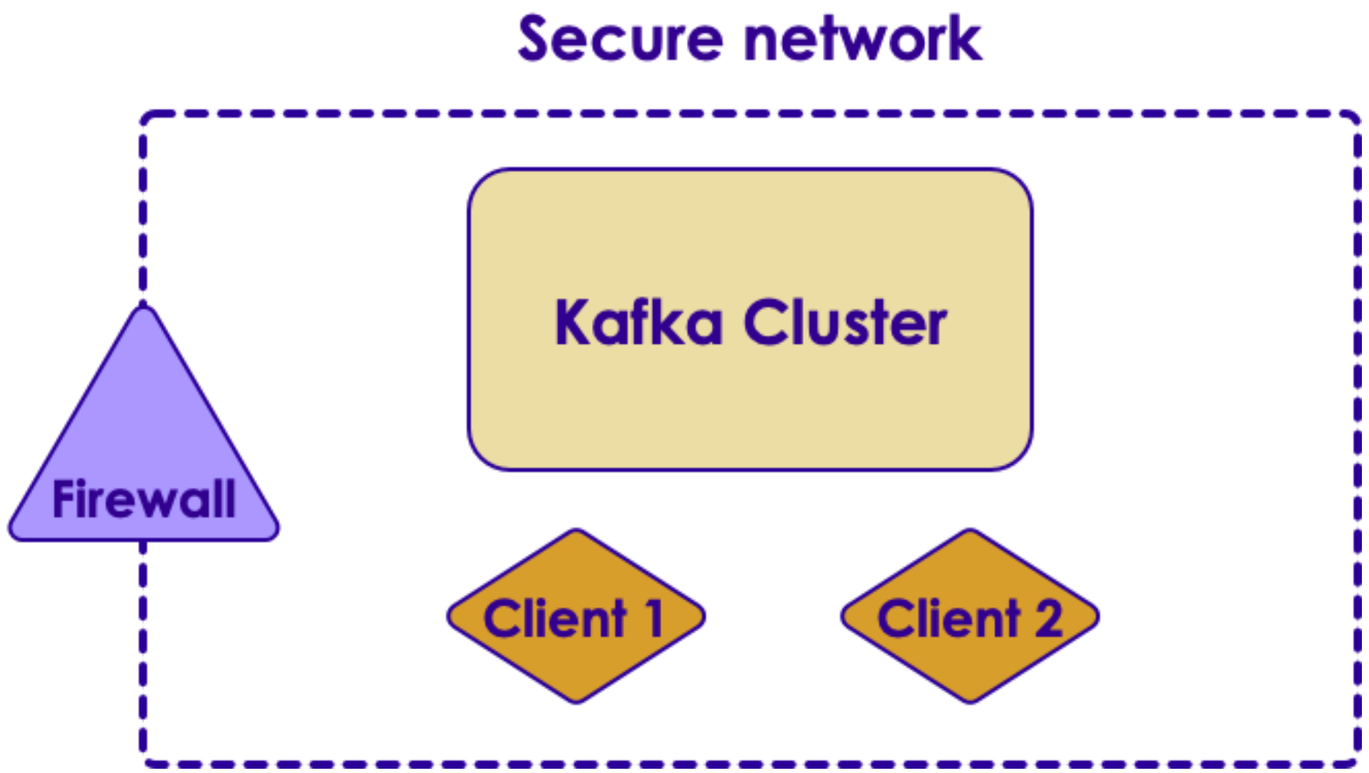
Topic Configuration

- Message.max.bytes: (default 1MB)
  - How big an individual message can be
  - Compressed size
  - So a producer can send a larger message (> 1 MB) provided it compresses below the 1MB limit
  - Kafka is not designed as large 'blob store'.. Hence the limit on messages
  - Increasing message size has implications
    - More network and disk activity
    - Consumers may fail to fetch messages (out of memory .etc)

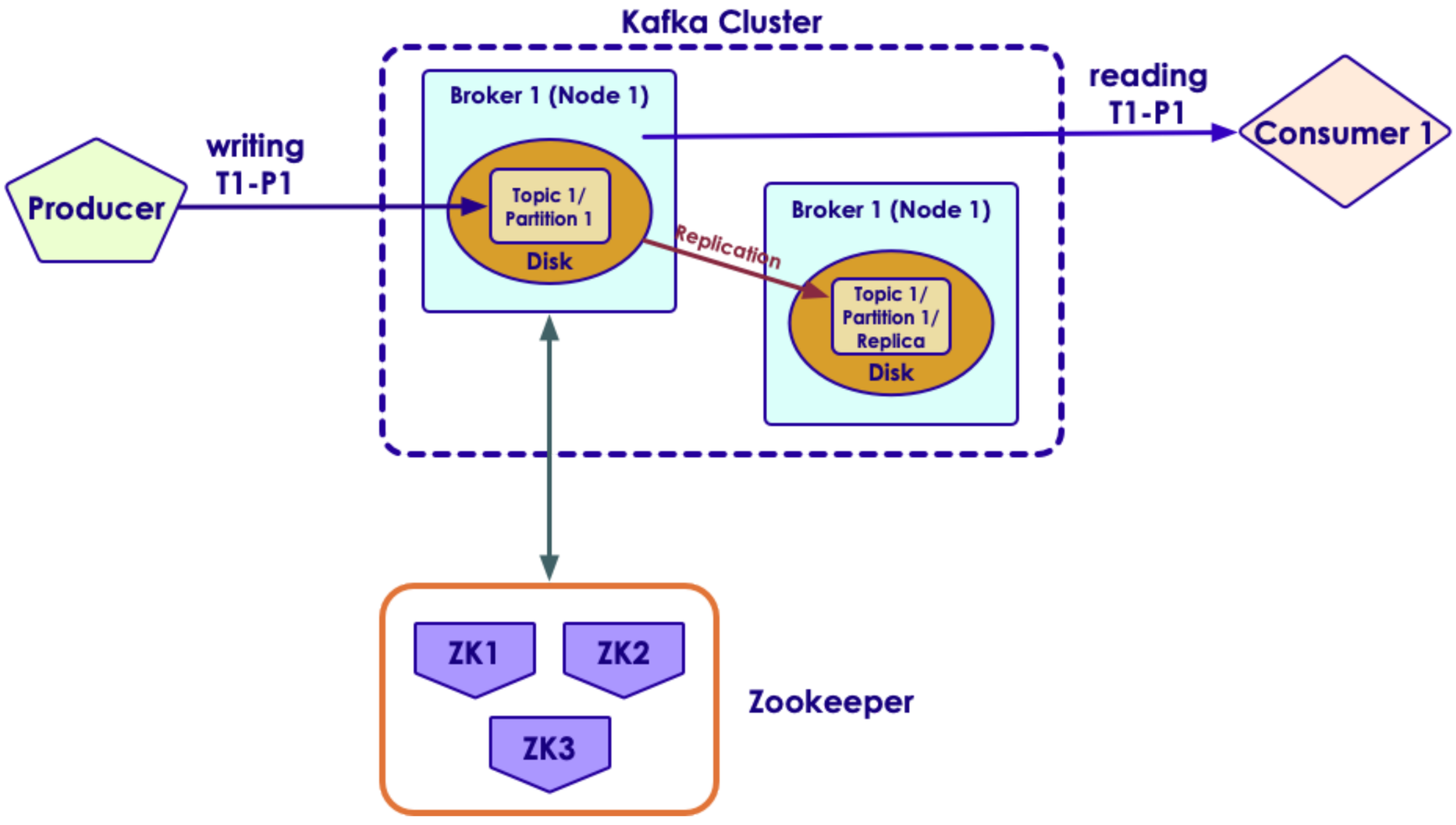
Securing Kafka

Securing Kafka - Trusted Network

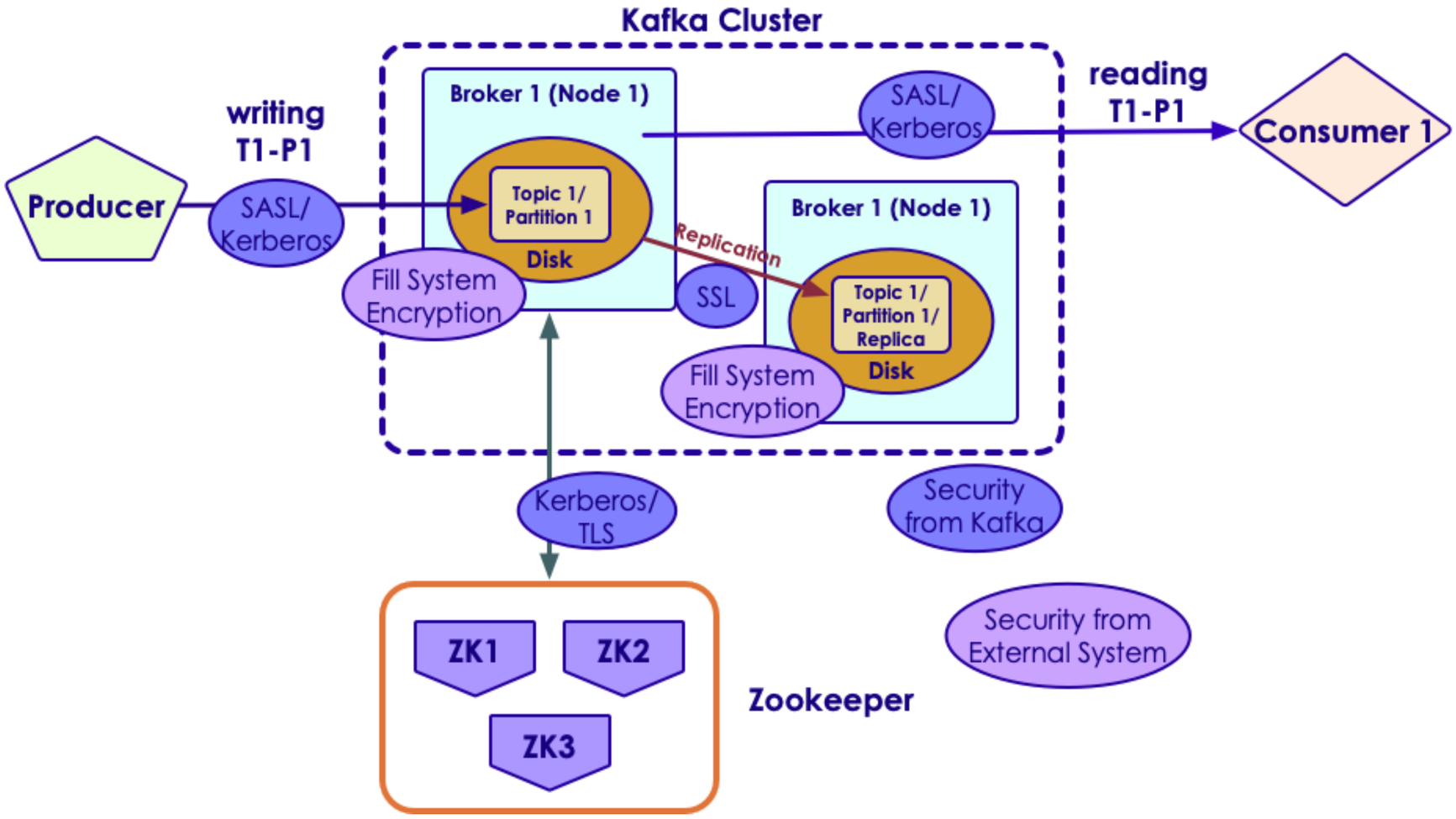
- Usually Kafka clusters are hosted in private / trusted networks
  - Not exposed to the Internet
- Both the cluster & clients are in trusted network
- Openly accessible to all clients



Quiz For The Class: Identify Points to Secure

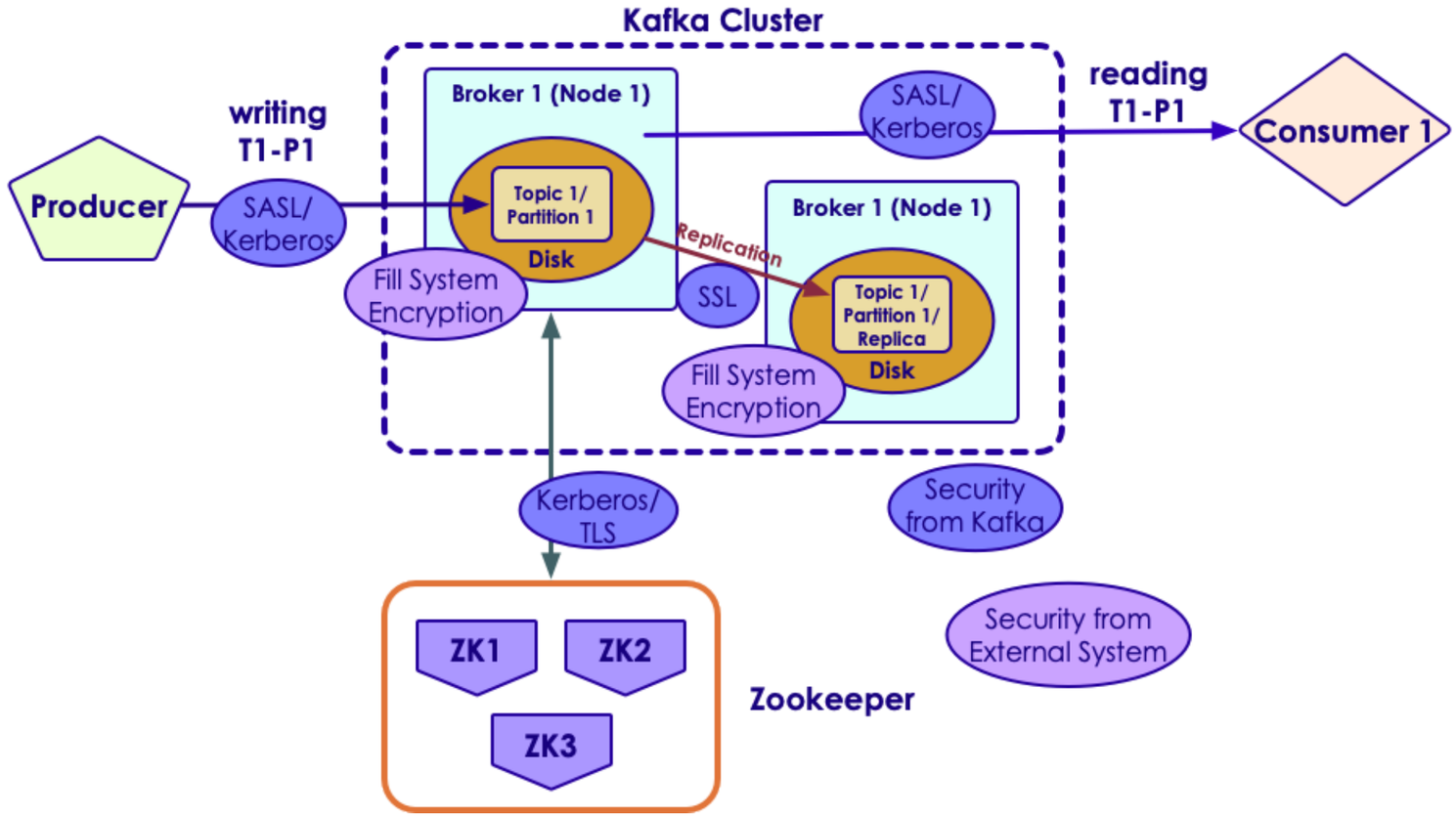


Securing Kafka



Securing Kafka

- Clients connect to Kafka brokers via **Kerberos / TLS**
- Kafka broker nodes talk to each other using **SSL**
- Kafka brokers talk to Zookeeper using **TLS**
- Data on disk (data at rest) is not encrypted by Kafka (**transparent encryption**)
  - Use file system / OS based encryption schemes



Secure Broker Configuration

- File: config/server.properties

```
# enable secure ports
listeners=SSL://:9093,SASL_SSL://:9094

# to enable plain text communications
# listeners=PLAINTEXT://:9092,SSL://:9093,SASL_SSL://:9094

security.inter.broker.protocol=SSL

# further config required based on secure protocol (SSL/TLS)
# ... Skipped ...
```

Access Control & Authorization

- Kafka supports user based authentication

```
# broker configuration
# these users can access every thing
super.users=User:bob;User:alice
```

```
# adding user as producer

kafka-acls -authorizer-properties
            zookeeper.connect=localhost:2181
            -add -allow-principal User:Bob
            -producer -topic test-topic
```

```
# adding user as consumer

kafka-acls -authorizer-properties
            zookeeper.connect=localhost:2181
            -add -allow-principal User:Bob
            -consumer -topic test-topic  -group Group1
```

Client Configuration

```
Properties props = new Properties();

props.setProperty(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
                  "localhost:9093");
props.setProperty(ConsumerConfig.GROUP_ID_CONFIG,
                  "securing-kafka-group");

// define protocol and specific properties
props.setProperty(CommonClientConfigs.SECURITY_PROTOCOL_CONFIG,
                "SSL");
props.setProperty(SslConfigs.SSL_TRUSTSTORE_LOCATION_CONFIG,
            "/etc/security/tls/kafka.client.truststore.jks");
props.setProperty(SslConfigs.SSL_TRUSTSTORE_PASSWORD_CONFIG,
            "test1234");
props.setProperty(SslConfigs.SSL_KEYSTORE_PASSWORD_CONFIG,
            "/etc/security/tls/kafka.client.keystore.jks");
props.setProperty(SslConfigs.SSL_KEYSTORE_LOCATION_CONFIG,"test1234");
props.setProperty(SslConfigs.SSL_KEY_PASSWORD_CONFIG, "test1234");

new KafkaConsumer(props);
```

Capacity Planning

Kafka Capacity Planning

- Prefer more medium size machines to fewer larger machines.Kafka scales well horizontally
- How much disk space do we need?Avg message size* throughput / sec * retention period * replication
- For example
    - Avg msg size = 5KB
    - Throughput = 1000 msgs / sec
    - Retention period = 7 days
    - Replication = 2
- Storage needed = 5KB x 1000 x (7 * 3600 * 24) x 2= 6 TB

Cluster Size

- Producer benchmark:How fast you can send messages from Producer into Kafka cluster
    - Depends on compression / batch sizing / ack
- Consumer benchmarkHow fast a message can be processed
    - Depends on application logic
    - Really need to measure it
- How to calculate optimal number of partitions?
    - Let's say Producer throughput to a single partition as P
    - Say Consumer throughput from a single partition as C
    - Target throughput T
    - Required partitions = Max (T/P, T/C)

Partitions / Brokers

- More partitions -> more time to recover in case of failure
  - Let's say we have 1000 partitions in a broker
  - When that broker fails, we need to find another 'leader / primary' broker for each partition
  - If it takes 10ms to elect a new primary broker for each partition
  - Total time to recovery = 10ms x 1000 = 10 secs
  - For this 10 seconds, these partitions are NOT available
- Some recommendations:
  - 2000 - 4000 partitions / broker
  - 10k - 50k partitions / cluster

Review and Q&A

- Let's go over what we have covered so far
- Any questions?