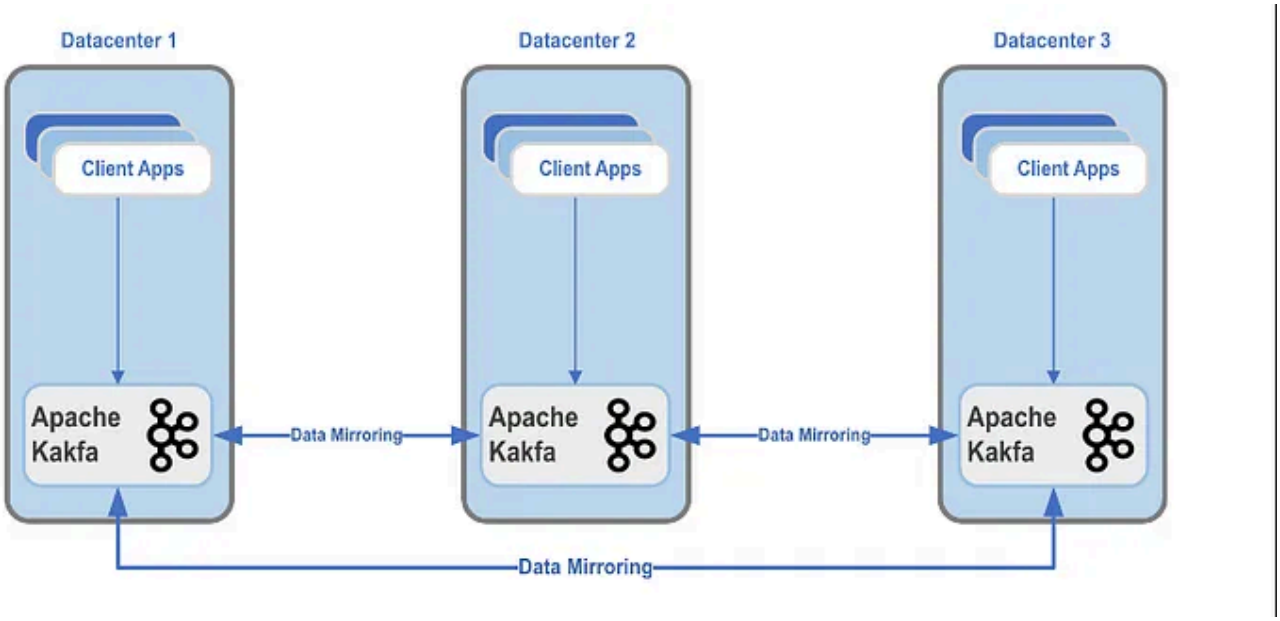
Kafka at PayPal

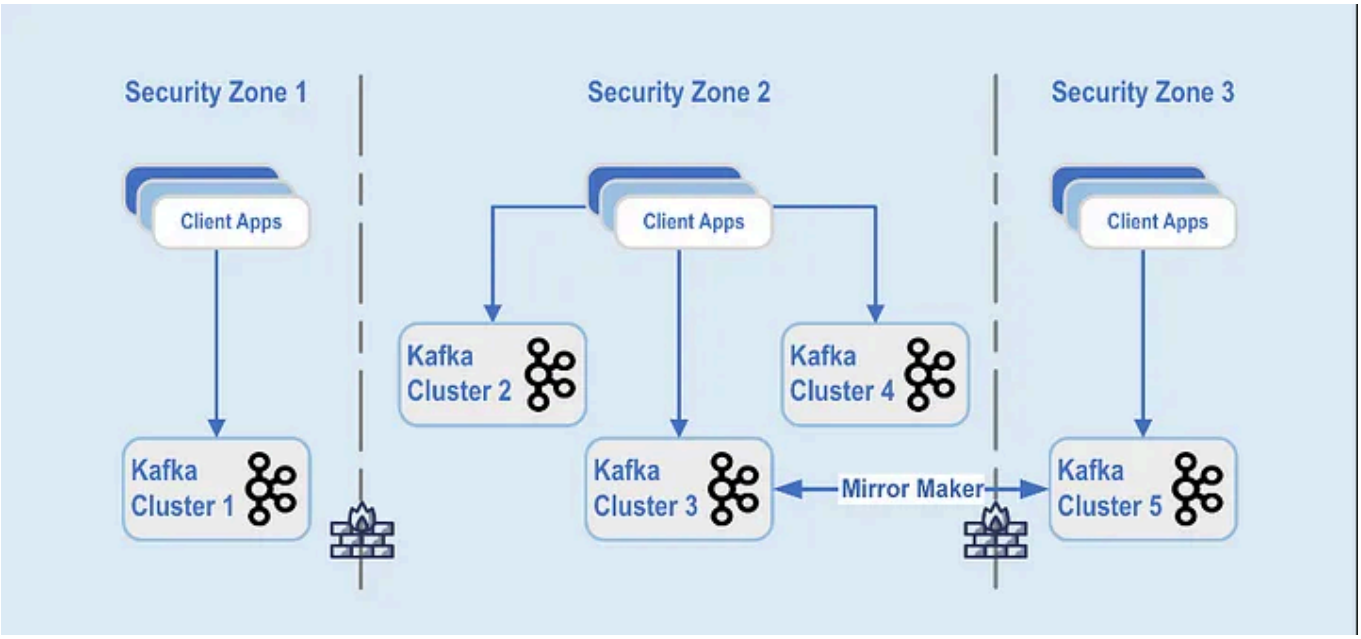
- with each of following use-cases processing over 100 billion messages per day.
  - first-party tracking
  - application health metrics streaming and aggregation
  - database synchronization
  - application log aggregation
  - batch processing
  - risk detection and management
  - analytics and compliance
- with 1500 brokers, 20k topics
- 2000 Mirror Maker nodes , used to mirror data among clusters offering 99.99% availability for kafka clusters.
- 85+ kafka clusters
- seamless scaling during peak

Operating Kafka at PayPal:

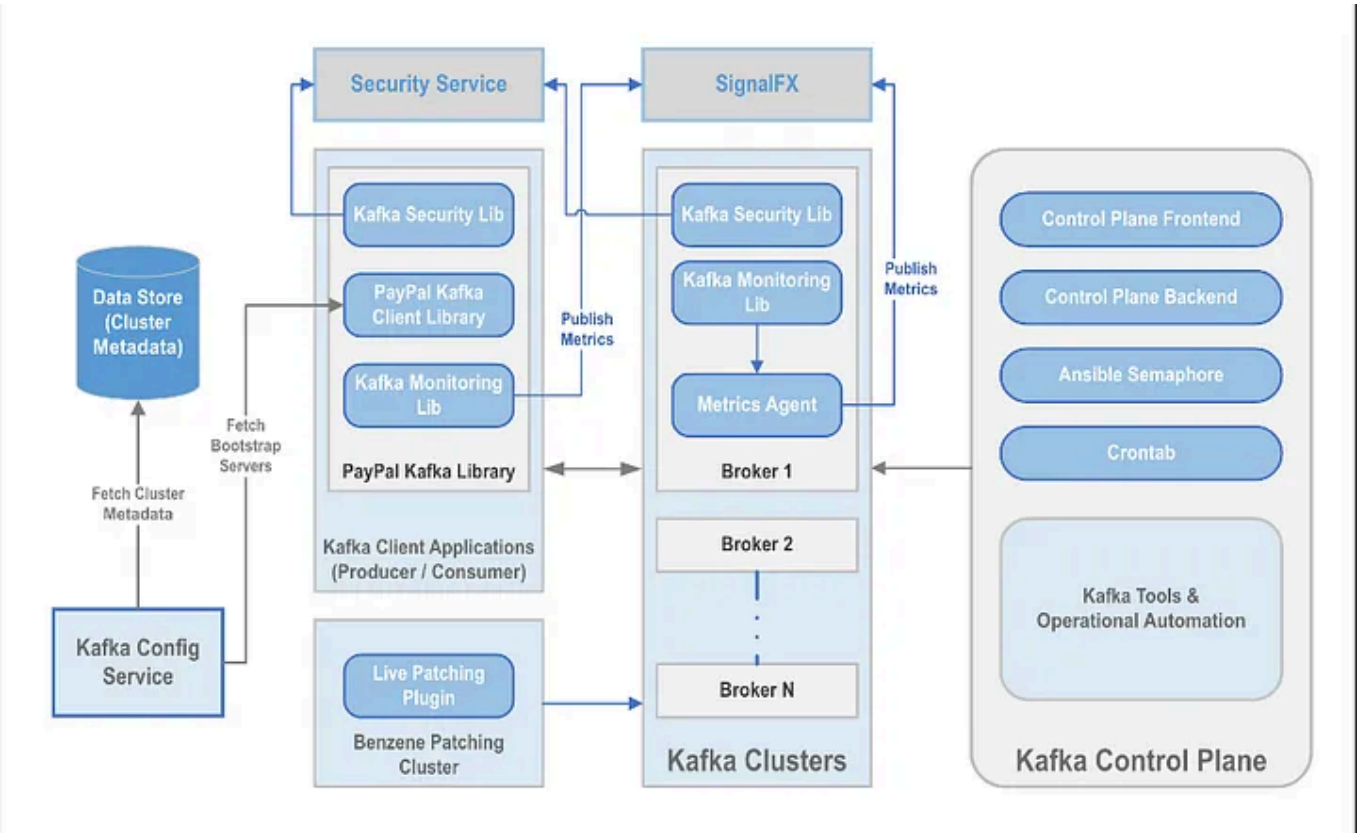
- infrastructure spread across multiple geographically distributed data centres and security zones.
- Kafka clusters are deployed across these zones, based on data classification and business requirements
- MirrorMaker is used to mirror the data across the data centers, which helps with disaster recovery and to achieve inter-security zone communication.



(Kafka cluster deployments in PayPal Data centers)



- Each Kafka cluster consists of brokers and zookeepers.
- Client applications connect to the topics hosted on these brokers to publish or subscribe the data to/from a cluster in the same or different Security Zone (via MirrorMaker).

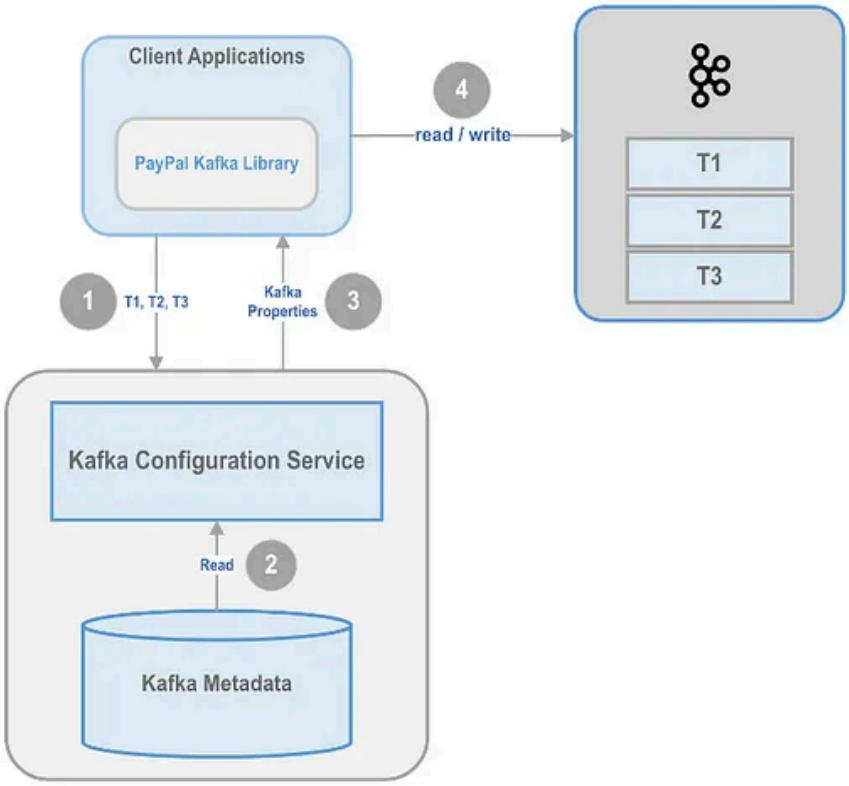


Crontab:

- schedules and manages automated tasks (cron jobs) to run at specified times or intervals.
- a daemon that runs in the background to execute scheduled commands.

Cluster Management

- ## Kafka Config Service
  - Before Kafka config service came into existence, clients would hardcode the broker IPs to which they connect to.
  - In case of any maintenance activity, we might need to replace the brokers due to various reasons such as upgrades, patching, disk failures, etc.
  - Kafka Config service is highly available stateless services.
  - Kafka clients need a set of bootstrap servers during initialization, (set of brokers) on which the topics are hosted, and application connects to the same brokers for producing or consuming the messages.
  - This service pushes the standard configuration to all the clients using Kafka, along with providing the necessary bootstrap server details, at the start of the application.
  - In case of any issues, the application teams do not have to rely and spend time on follow-ups with the platform team but can simply restart their application to incorporate any recent changes.



- # Kafka ACLs
  - Before Access Control Lists (ACLs) came into the picture, Kafka allowed connections on plain text port and any PayPal application could connect to any of the existing topics, which was an operational risk for our platform which streams business critical data.
  - ACLs were introduced
  - to ensure controlled access to kafka clusters and make platform secure by enabling access only via SASL port. this requires application to authenticate and authorize in order to gain access to kafka clusters and topics.
  - each application needs to explicitly call out if this is a producer/consumer for the topic it wants to connect to.
  - benifits:
  - 1. record of every application accessing topics and clusters, and their intent.
  - 2. by implementing ACLs across all kafka clusters, using a unique application context assigned to every application for identification.
- # Resilient Client Library:
  - ## Scenario:
    - An e-commerce company has multiple environments: `dev`, `staging`, and `prod`.
    - Kafka clusters are deployed in each environment, and topics move between these environments during testing or scaling.

- **Without Resilient Client Library:**
  - Developers manually update Kafka configurations (e.g., broker lists, topic details) in the application code or properties files for each environment.
  - Operational changes like scaling brokers or reassigning topics require downtime for updates and redeployment.
- **With Resilient Client Library:**
  - The library dynamically fetches environment-specific configurations from the discovery service.
  - Developers don’t need to modify or redeploy the application for such changes.
- **Benefits:**
  - Developers use the same codebase across environments.
  - Moving topics between environments or clusters is seamless.
  - Operational changes, like adding brokers, do not disrupt the application.

- ## Monitoring Library:
  - publishes critical metrics for client applications, which helps to monitor health of end user applications.
  - Benefits:
  - 1. Application can set alerts and be notified in case of any issues.
  - 2. helps to monitor and keep kafka platform stable.
  - 3. Action on any issues within a few minutes of turnaround time.
- ## Kafka Security Library:
  - All the application should be SSL authenticated to connect to kafka clusters.
  - this library pulls the required certificates and tokens to authenticate the application during startup.
  - benefits:
  - 1. Kafka platform at PayPal supports more than 800 applications, this library avoids overhead for the end users with respect to Key management events such as Certificate updates, Key rotations, etc.
  - 2. Requires minimal know-how on authentication and authorization when connecting to Kafka clusters.
- ## Kafka QA Platform:
  - The older QA environment was cluttered with a lot of ad-hoc topics, all hosted on handful of clusters, and with feature disparity compared to production setup.
  - developers would create random topics for their testing, which would cause issues while rolling out changes to production as the configuration or cluster/topic details would be different from one used for testing.

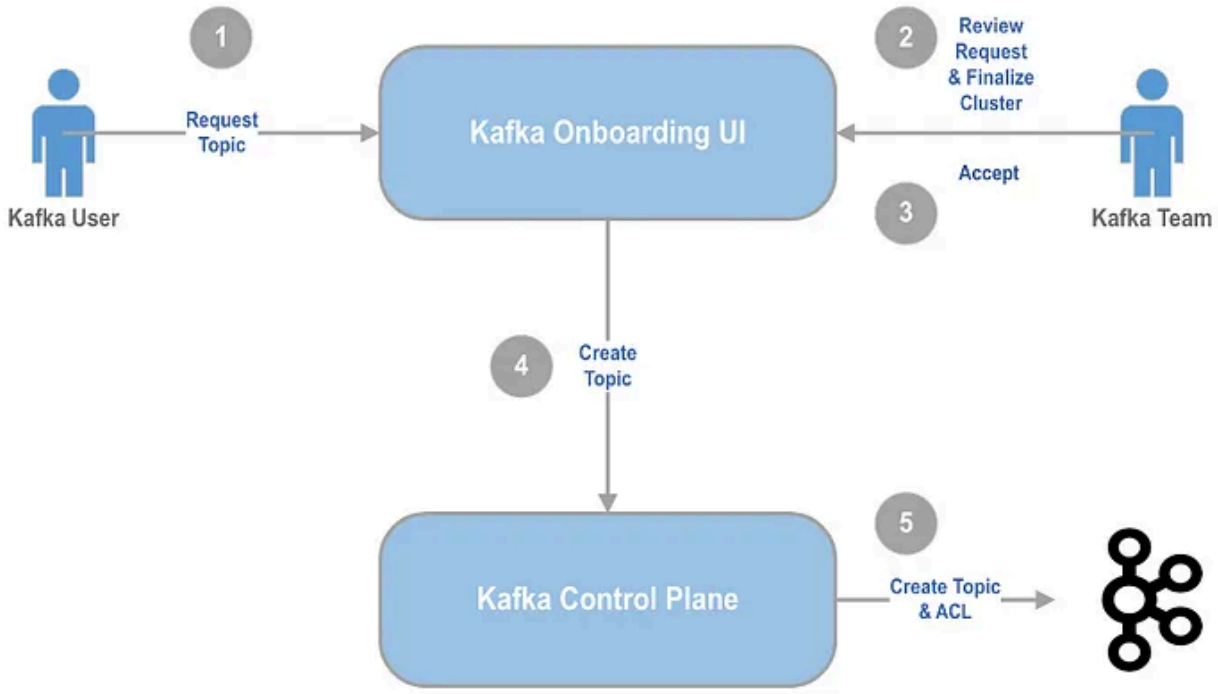
- redesigned and introduced new QA platform. it provides 1-1 mapping between prod and QA clusters, following the same security standards as production setup.
- hosted all QA clusters and topics in Google cloud platform, with the brokers spread across multiple zones within GCP with rack aware feature to achieve high availability.
  // Multiple Zone Deployment: Kafka brokers are spread across multiple AZs within a region. ensures high availability bcz if one zone goes down, broker in another zone remain operational.
  // Rack Awareness for High Availability: considers physical location of brokers (GCP zones) when placing replicas of a partition. ensures replicas of a topic's partitions are distributed across different zones.
- Along with a rigorous evaluation for finding the right SKU without compromising performance, we optimized the public cloud costs.
  // SKU (stock keeping units) evaluation: refers to different types of VMs/ services available for use. For kafka brokers, choosing SKUs with sufficient memory, CPU and networking capabilities is critical for optimal performance.
- Compared to the older platform, our new platform reduces the cloud costs by 75% and improves the performance by about 40%.

- **Configuration Management:**
  - helps to have complete info. on the infrastructure.
  - in case of data loss, this is the single source of truth, which helps to rebuild clusters in a couple of hours.
  - entire kafka infra-metadata such as details on topics, clusters, applications etc. are stored in internal configuration management system.
  - metadata stored is backed up frequently to ensure the most recent data.
- **Patching security vulnerabilities:**
  - We use BareMetal (BM) for deploying kafka brokers.
  - Virtual Machines(VM) for zookeeper and Mirrormakers.
  - all hosts within a kafka platform need to be patched at frequent intervals of time to resolve any security vulnerabilities.
  - this patching is performed by internal patching system.
  - kafka topics are configured with 3 replicas.
  - patching would require BM restarts which could cause under replicated partitions depending on patching time.
  - if consecutive brokers would have restarted, then it could cause data loss.
  - therefore, we built a plugin to query under replicated partition status before patching the host.
  - this would ensure each cluster can be patched in parallel but only a single broker is patched at a time with no data loss.

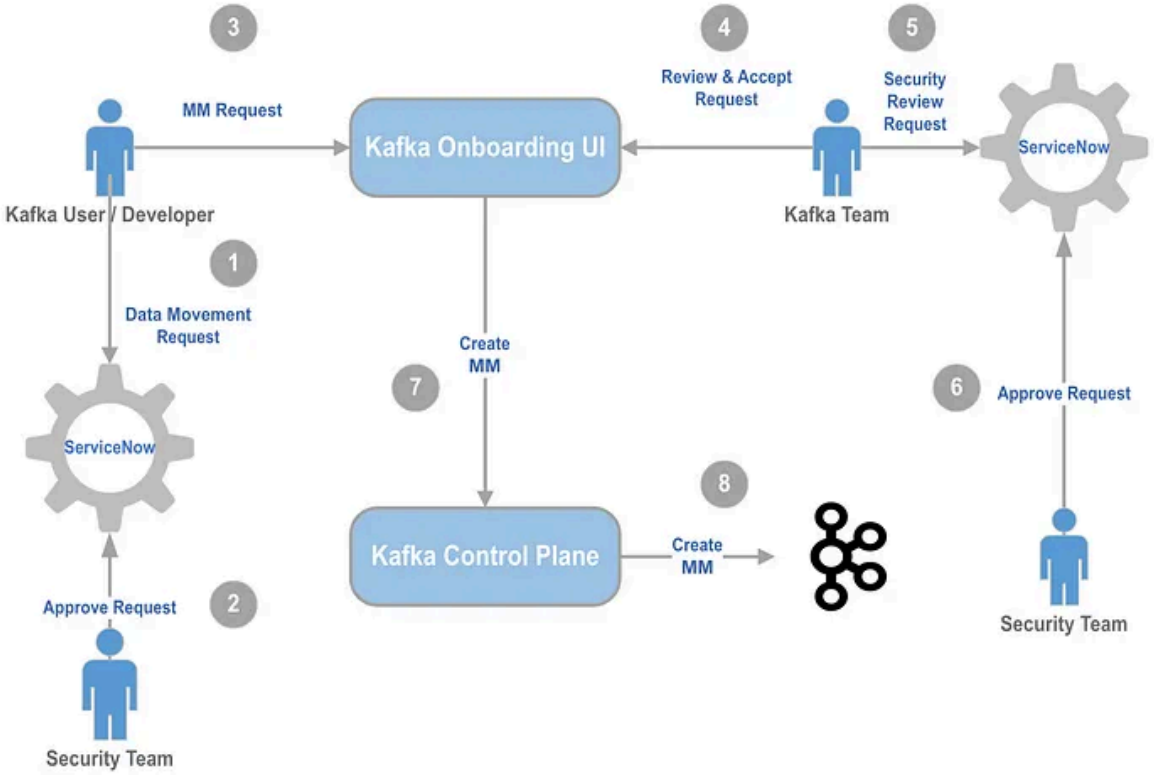
Operational Automation:

- ## Topic Onboarding:

```
- For each application being onboarded to the Kafka System for the first time, a unique token is generated and assigned.
- This token is used to authenticate the client's access to the Kafka Topic.
- Followed by creating ACLs for the specific application and topic based on their roles.
- Now the application can successfully connect to Kafka topic.
```



- ## MirrorMaker Onboarding:
  - We have Mirrormakers to mirror the data from source cluster to destination cluster for specific use cases where the producer or consumer applications are running in different zones.



- **Repartition Assignment Enhancement:**
  - process of redistributing/ reassigning partitions of a topic among available brokers/ nodes in a kafka cluster.
  - required when we carry out any maintenance activity on a subset of brokers/ while replacing brokers.
  - By default, Kafka does the repartition for all the partitions including the partitions which are hosted on healthy brokers.
  - But we have made modifications to re-assign only under replicated partitions hosted on the affected brokers.
  - This enhancement helps avoid unpredictably longer times for re-partitioning.