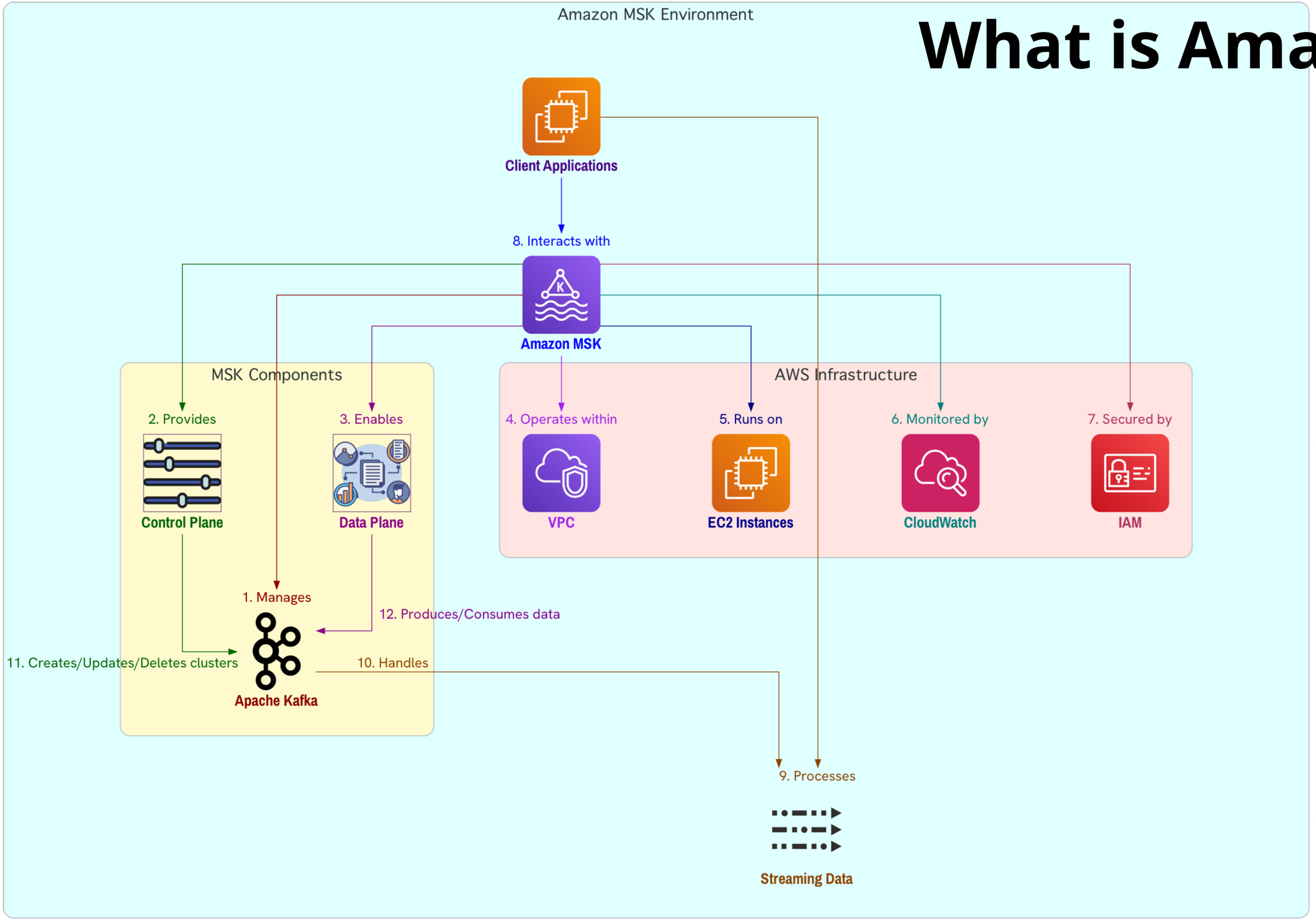
Amazon Managed Streaming for Apache Kafka (Amazon MSK)

Table of Contents

Amazon Managed Streaming for Apache Kafka (Amazon MSK)

1. What is Amazon MSK?
2. Architecture of Amazon MSK
3. KRaft Mode
4. Tiered Storage
5. Tiered Storage: Features and Benefits
6. MSK Serverless
7. What is MSK Connect?

What is Amazon MSK?



1️⃣. 🌐 Fully managed service for Apache Kafka

🏗️ Build and run Kafka applications

🔧 AWS handles cluster maintenance

🎯 Focus on application development

2️⃣. 🔄 Process streaming data

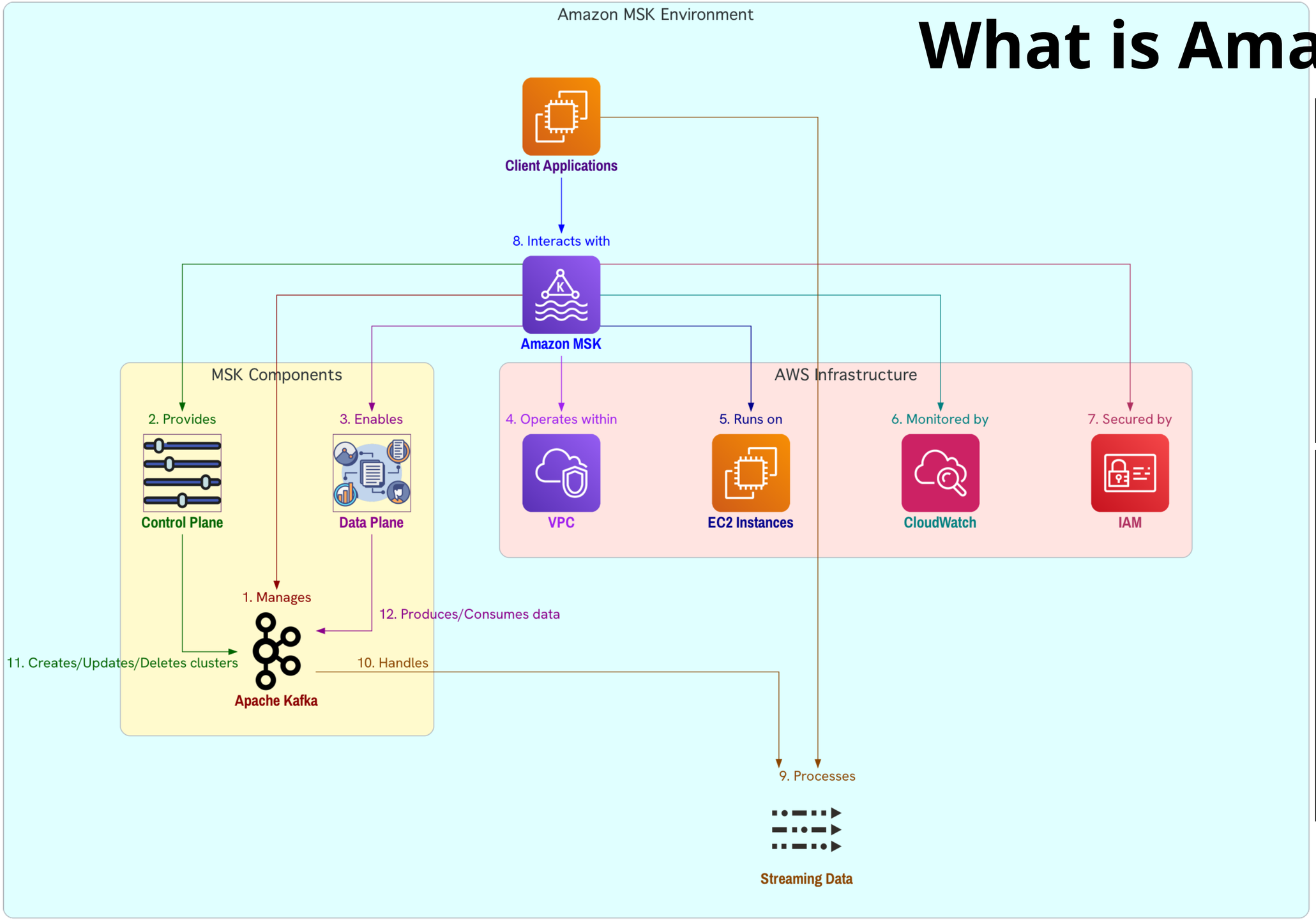
📊 Real-time data processing

📝 Log analytics

🤹 Event-driven applications

📈 Real-time dashboards

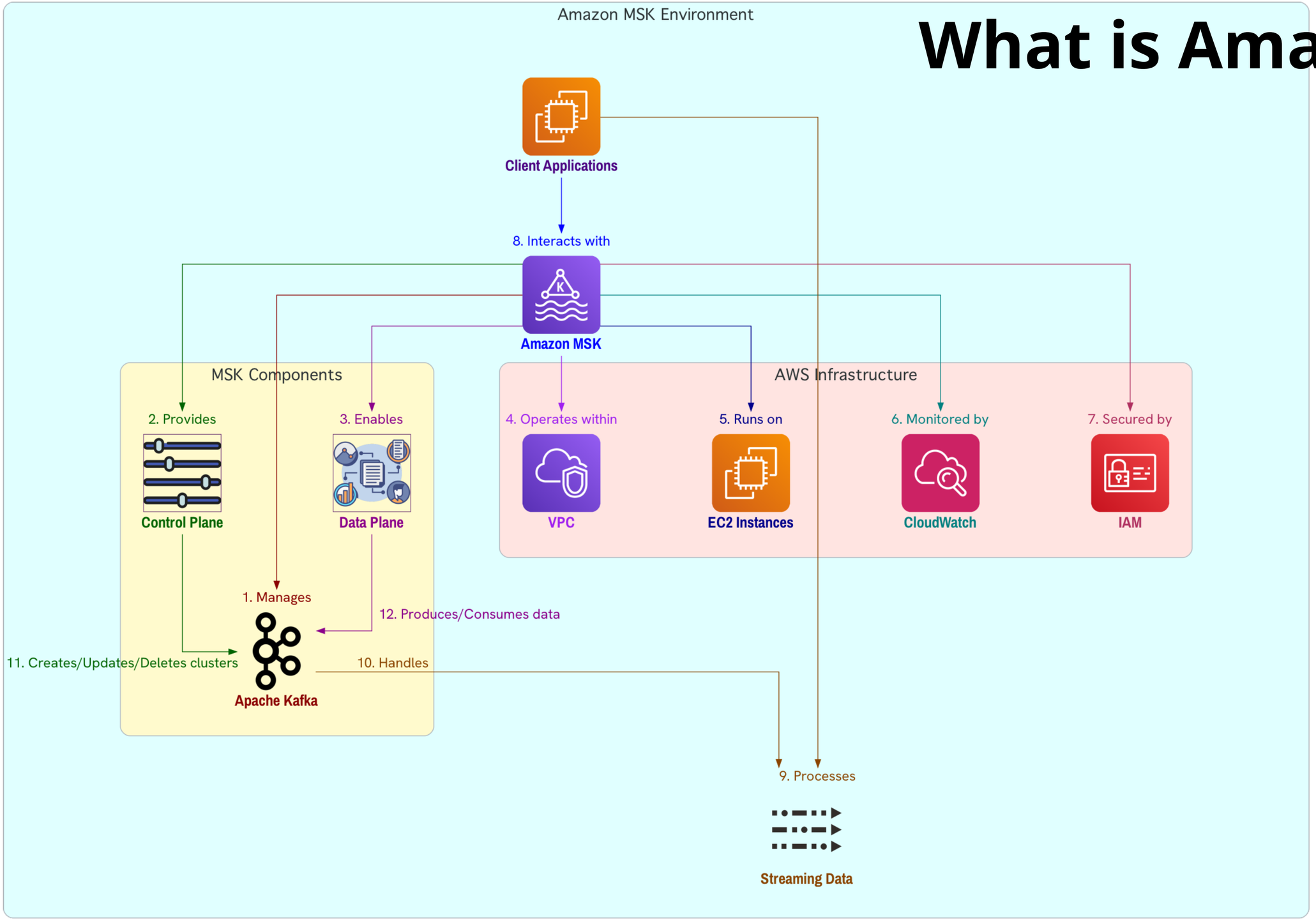
What is Amazon MSK?



3️⃣. 🎛️ Control-plane operations
➕ Create clusters
🔄 Update clusters
➖ Delete clusters
🔧 Reduces operational overhead

4️⃣. 🔍 Data-plane operations
📤 Produce data
📥 Consume data
🛠️ Use familiar Kafka APIs and tools
🎮 Full control over data processing logic

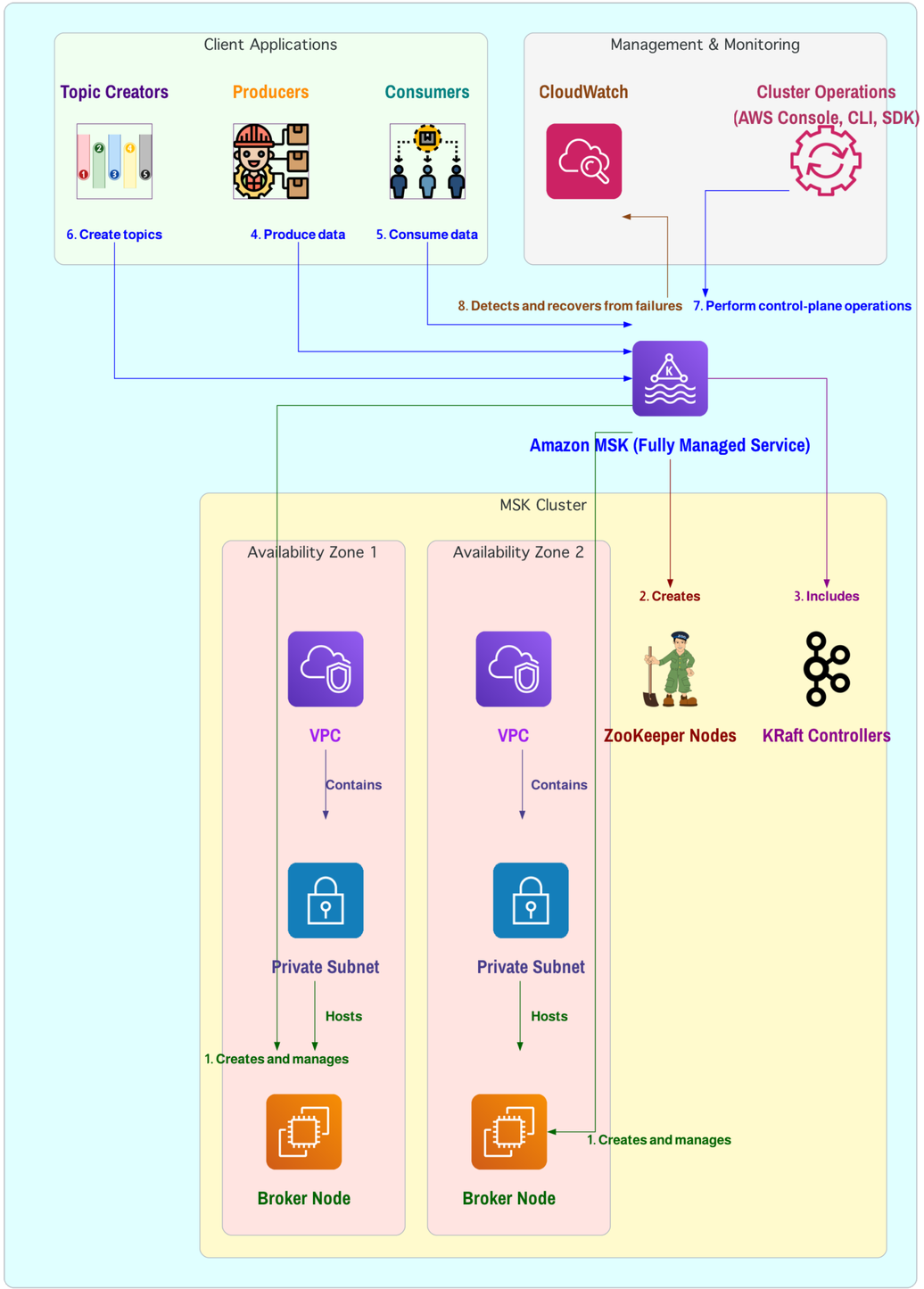
What is Amazon MSK?



5️⃣. ⭐ Runs open-source Apache Kafka versions
🔗 Ensures compatibility
🆕 Access to latest features
🛠️ Leverage community improvements

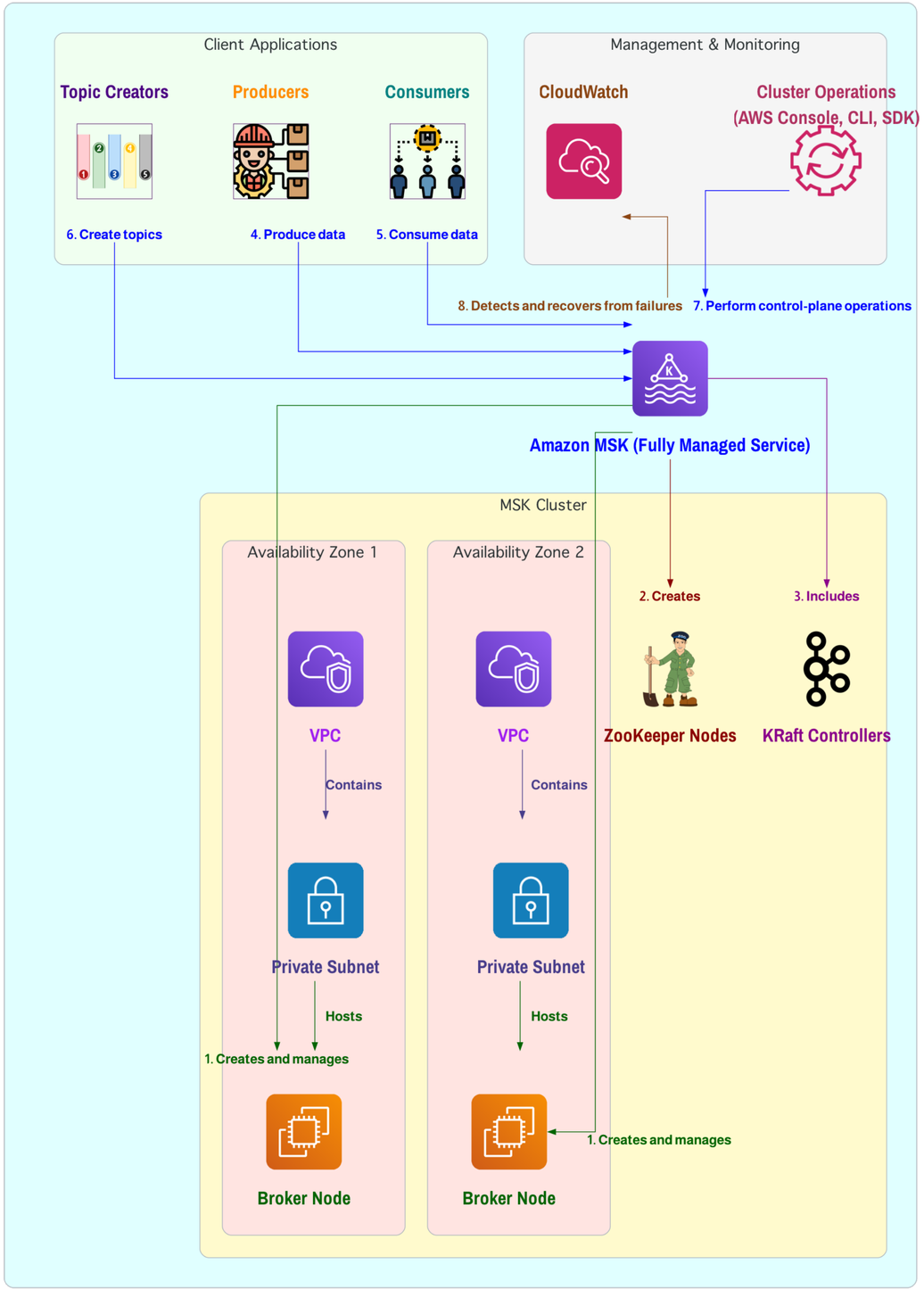
6️⃣. 🔌 Supports existing ecosystem
📱 Existing applications
🔧 Tooling
🧩 Plugins
🚫 No code changes required
⏱️ Saves migration time
📉 Reduces migration risks

Architecture of Amazon MSK



1️⃣. 🖥️ Broker nodes: Distributed across Availability Zones	2️⃣. 🔗 ZooKeeper nodes: Reliable distributed coordination	3️⃣. 🔄 KRaft controllers: Metadata management alternative
🔢 Specify number of nodes per zone	🤖 Automatically created by MSK	🔁 Replaces ZooKeeper for metadata management
🔀 Minimum one broker per zone	📊 Manages configuration and synchronization	🧠 Manages metadata within Kafka cluster
🌐 Each zone has own VPC subnet	🔄 Open-source coordination server	💰 No additional cost
🔐 Ensures high availability and fault tolerance		🛠️ No extra setup or management needed

Architecture of Amazon MSK



4️⃣. 👥 Client applications: Producers, consumers, topic creators
📩 Produce data to topics
📩 Consume data from topics
📝 Create topics
🔌 Uses standard Kafka APIs and tools
🔗 Seamless integration with existing applications

5️⃣. 🎛️ Cluster operations: Control-plane management
🖥️ AWS Management Console
⌨️ AWS CLI
🧰 SDK APIs
➕ Create clusters
➖ Delete clusters
📋 List clusters
🔍 View cluster properties
🔄 Update broker number and type

6️⃣. 🛡️ Fault tolerance: Automatic failure detection and recovery
🚨 Detects common failure scenarios
🔧 Mitigates or replaces unhealthy brokers
💾 Reuses storage to minimize data replication
⏱️ Limits downtime
🔄 Maintains same broker IP addresses after recovery

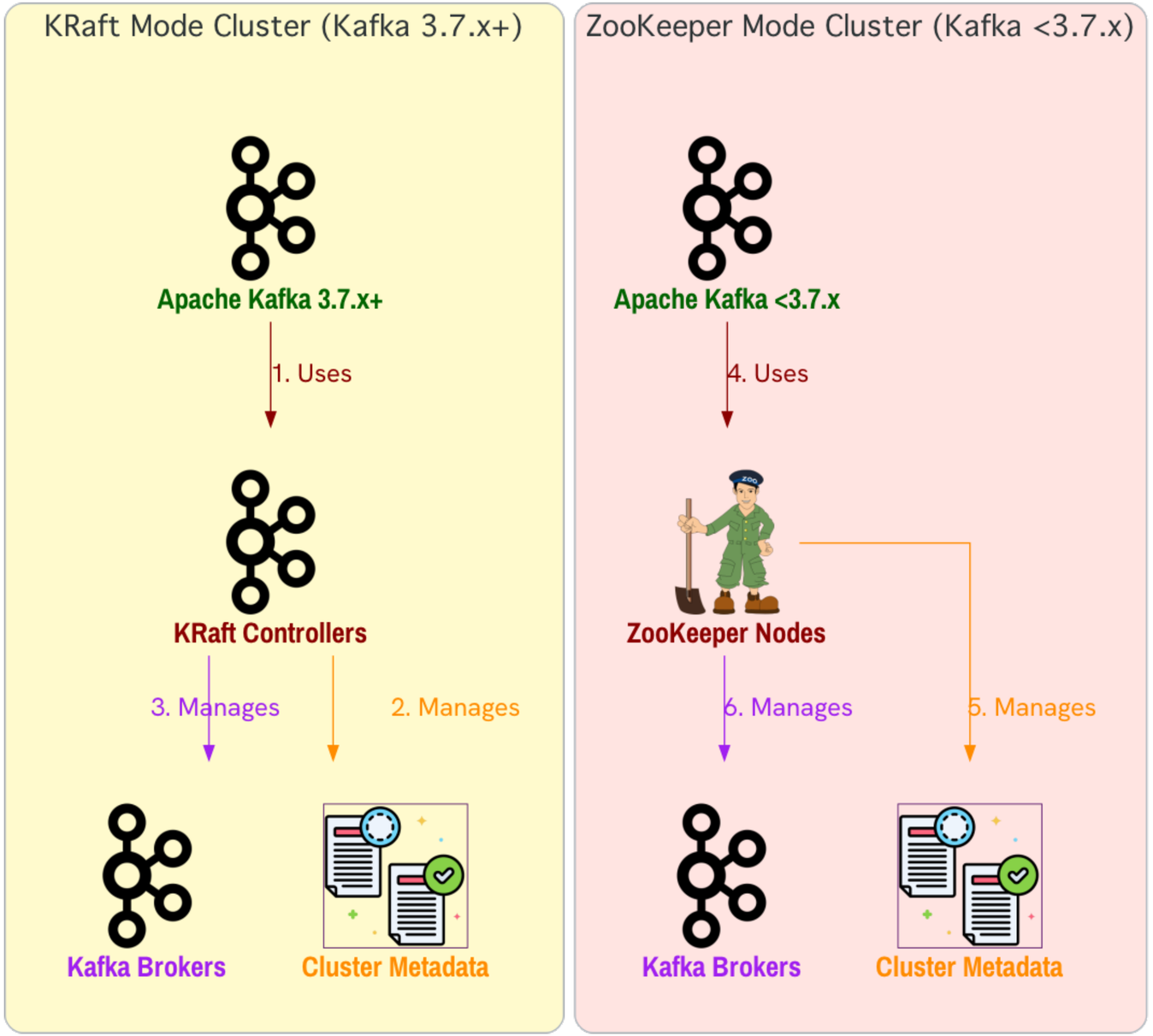
KRaft Mode

1️⃣. 🔄 KRaft mode introduction

- 📅 Introduced in Kafka 3.7.x
- 🔁 Replaces Apache ZooKeeper
- 📊 Metadata management within Kafka cluster
- 💰 No additional cost
- 🛠️ No additional setup or management

2️⃣. 🆕 Availability

- 🏗️ Only for new clusters
- 🔒 No switching after creation



Amazon MSK: KRaft vs ZooKeeper Mode

3️⃣. 🖥️ Cluster creation

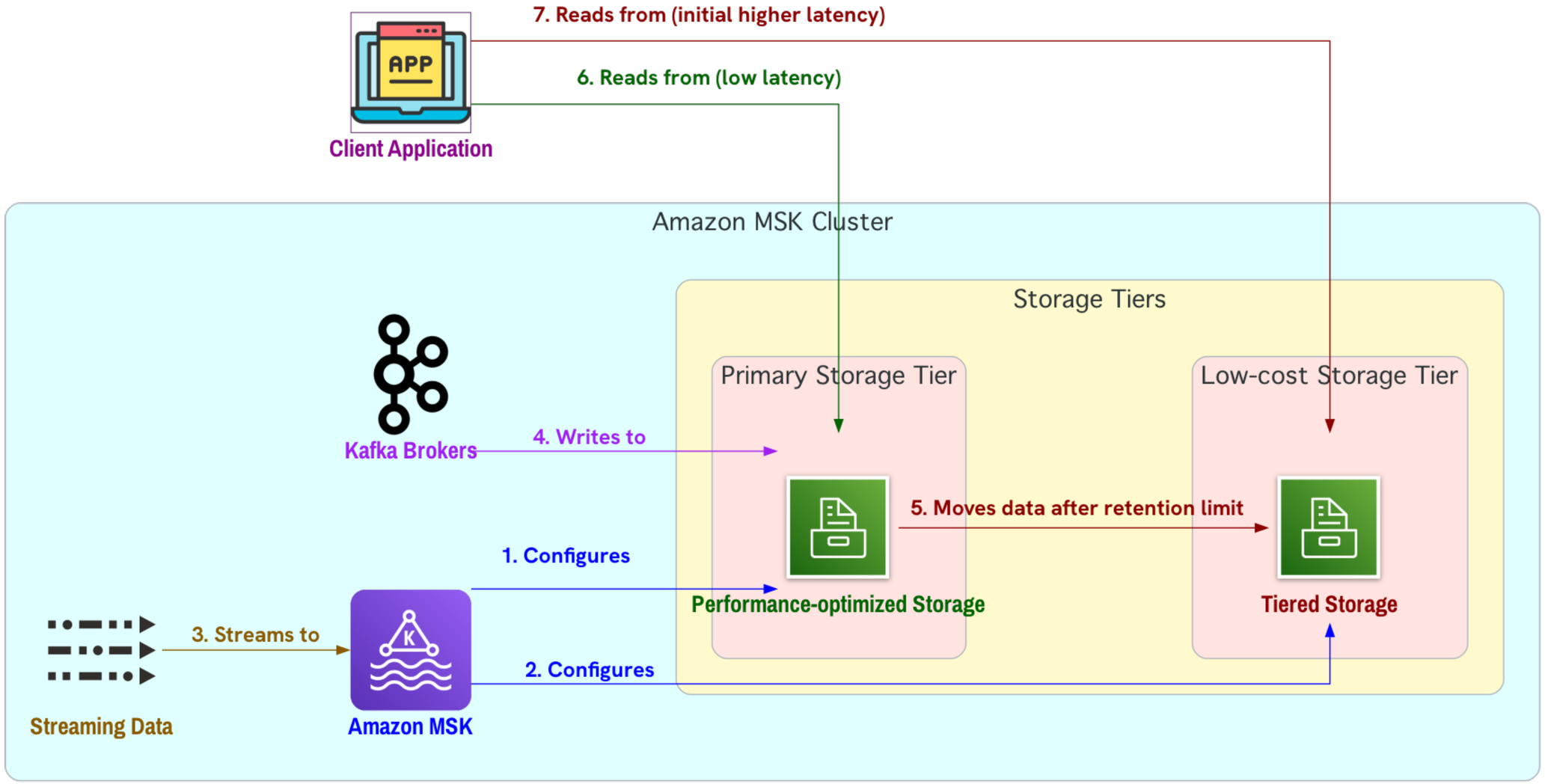
- 🎛️ MSK console
- ✅ Select Kafka 3.7.x
- ☑️ Check KRaft checkbox

4️⃣. 📊 Partition capacity

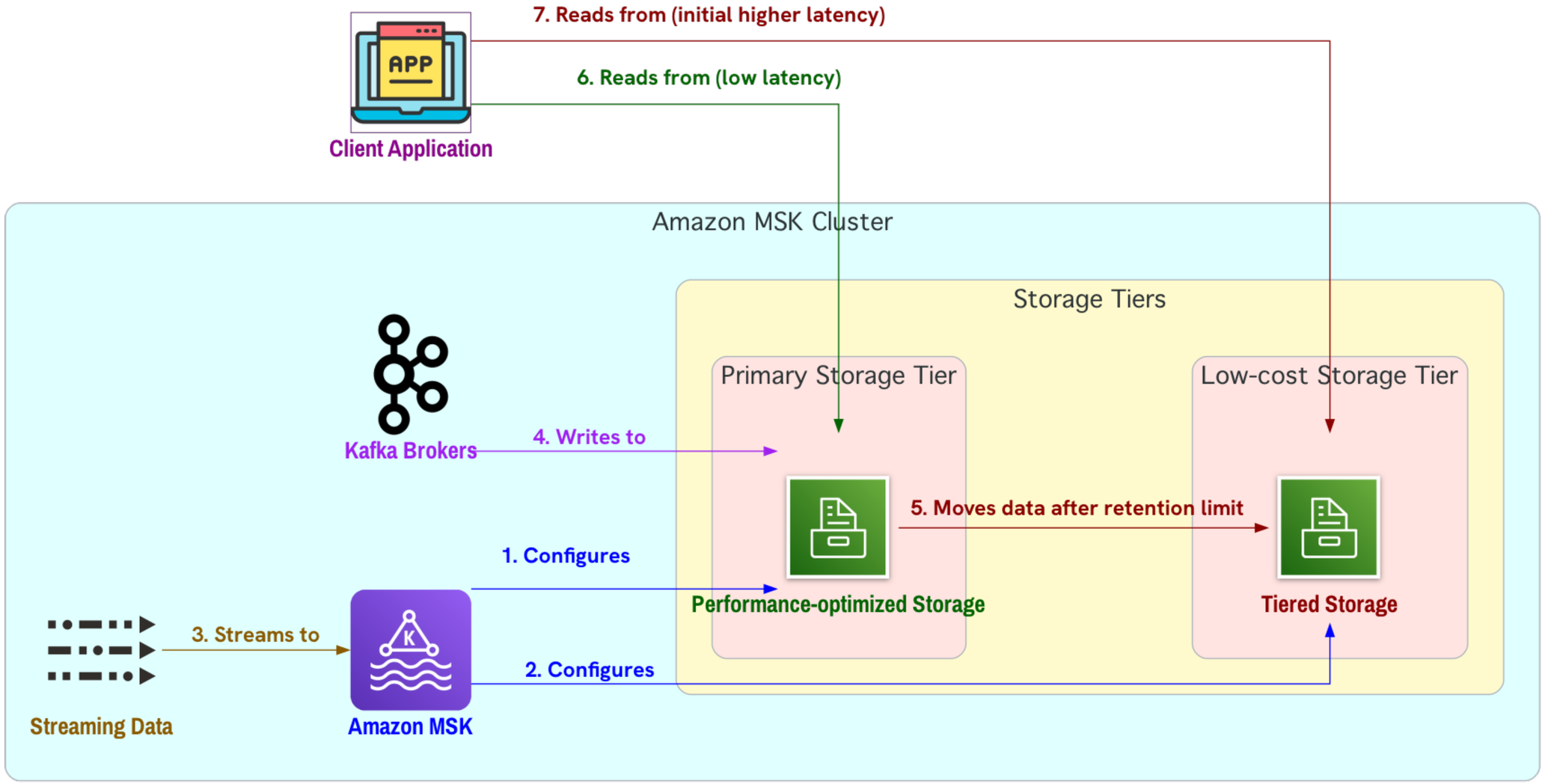
- 🔢 Same partitions per broker
- 📈 More partitions per cluster
- 🔌 Provision more brokers

5️⃣. 🗄️ Metadata storage and access control

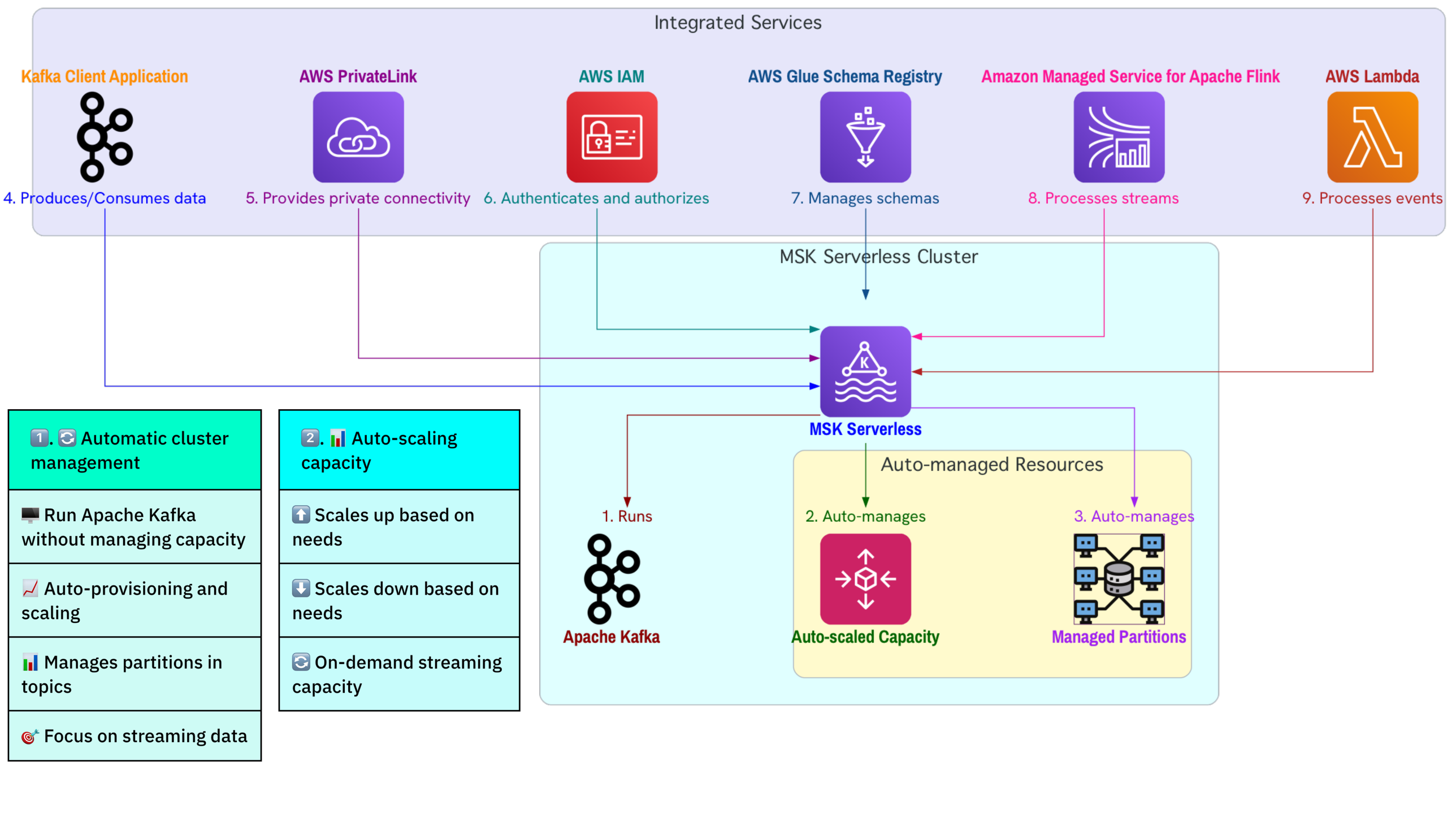
- 💾 Stored on KRaft controllers
- 🔐 Simplified access control
- 🚫 No separate controller node access

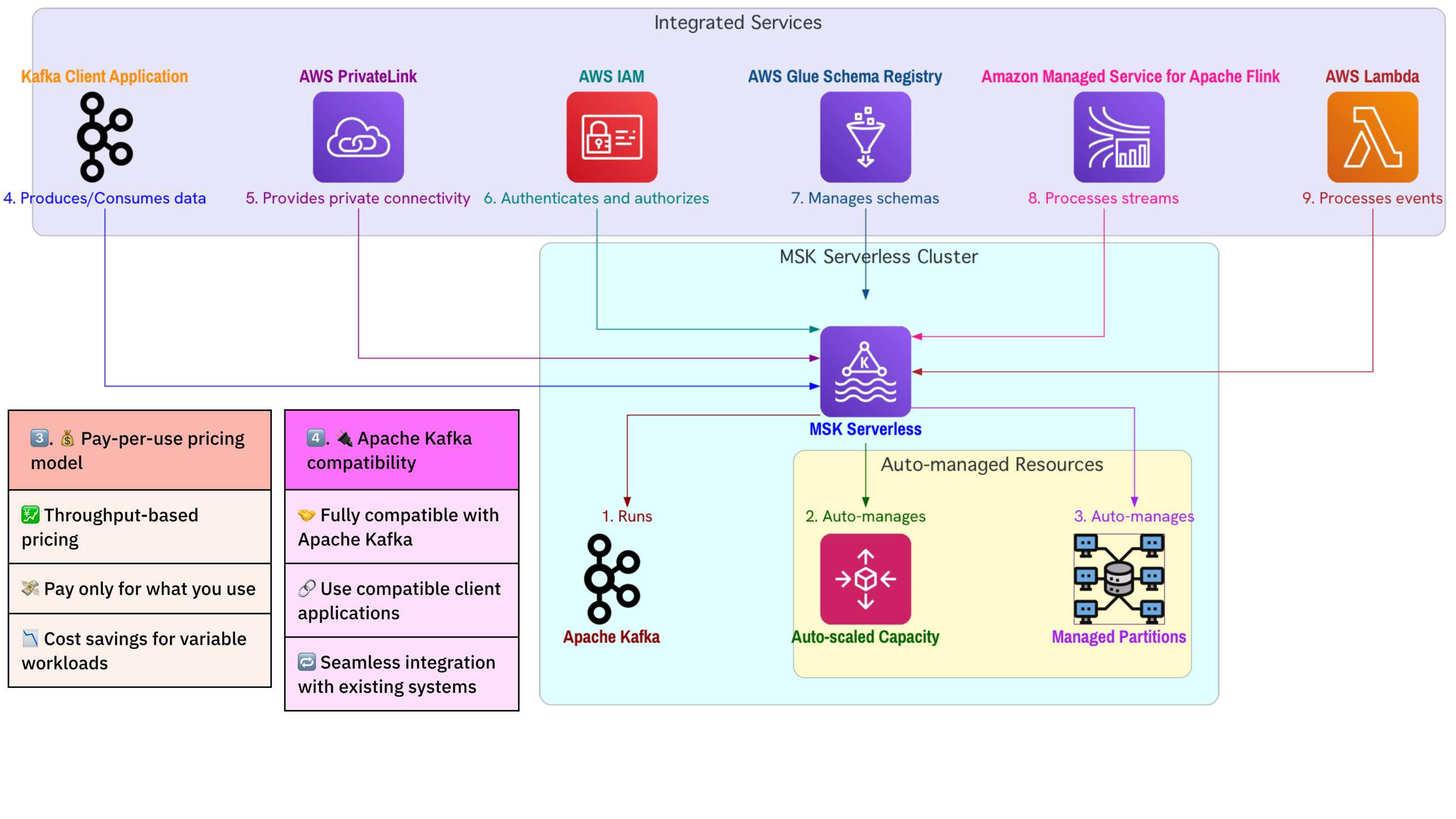


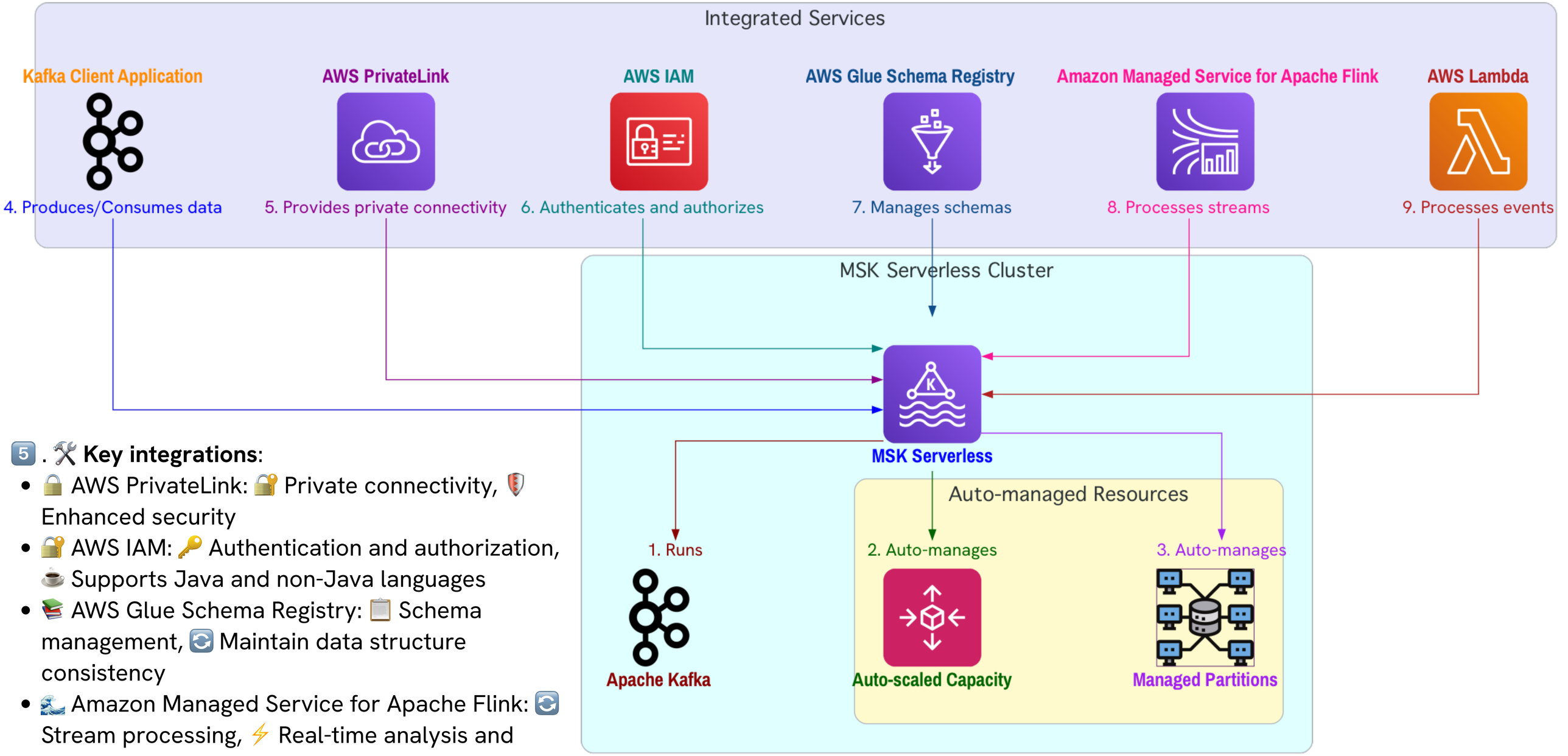
1. 💰 **Low-cost, scalable storage solution**: 📈 Scales to unlimited storage, 💻 Cost-effective for streaming apps
2. 🚀 **Balances performance and cost**: ⚖️ Optimizes streaming infrastructure, 🔧 Configurable MSK clusters
3. 📊 **Two-tier storage system**: 🏎️ Performance-optimized primary tier, 🚚 Low-cost secondary tier, 🔄 Automatic data movement
4. 🔄 **Automatic data movement**: ⏱️ Based on retention limits, 🤖 Reduces manual intervention, 📉 Lowers management overhead
5. 📉 **Initial latency increase, then similar performance**: ⏳ Higher latency for first few bytes, 🏃 Similar latency for sequential reads
6. 💳 **Pay-per-use model**: 🚫 No storage provisioning needed, 💾 Store any amount of data, 💸 Pay only for what you use



1. 🌐 **Virtually unlimited scalability**: 📈 Scales to unlimited storage, 🧠 No guesswork for infrastructure scaling, 🚀 Flexible for growing data needs
2. 🕰️ **Longer data retention**: 📚 Extended data storage in Kafka topics, 🖥️ No need to increase brokers, 💰 Historical analysis without added costs
3. 🛡️ **Extended safety buffer**: ⏳ Handles unexpected processing delays, 🔨 Enhances system resilience, 🔧 Manages unforeseen issues
4. 🔄 **Reprocess old data easily**: 🔄 Maintains production order, 🧰 Uses existing processing code and APIs, 🔍 Enables consistent historical analysis
5. ⚡ **Faster partition rebalancing**: 🏎️ Quicker rebalancing process, 💾 No replication across broker disks, 🔥 Improved performance during maintenance
6. 🔒 **Secure data transfer within VPC**: 🔐 Data moves within VPC, 🚫 No internet transfer, ⚡ Reduced potential latency
7. 🔌 **Seamless client connectivity**: 🔄 Same connection process, 👍 Backward compatibility, 😊 Easy adoption


Integrated Services
Kafka Client Application
AWS PrivateLink
AWS IAM
AWS Glue Schema Registry
Amazon Managed Service for Apache Flink
AWS Lambda
4. Produces/Consumes data
5. Provides private connectivity
6. Authenticates and authorizes
7. Manages schemas
8. Processes streams
9. Processes events
MSK Serverless Cluster
MSK Serverless
Auto-managed Resources
1. Runs
Apache Kafka
2. Auto-manages
Auto-scaled Capacity
3. Auto-manages
Managed Partitions
1️⃣. 🔄 Automatic cluster management
🖥️ Run Apache Kafka without managing capacity
📈 Auto-provisioning and scaling
📊 Manages partitions in topics
🎯 Focus on streaming data
2️⃣. 📊 Auto-scaling capacity
⬆️ Scales up based on needs
⬇️ Scales down based on needs
🔄 On-demand streaming capacity

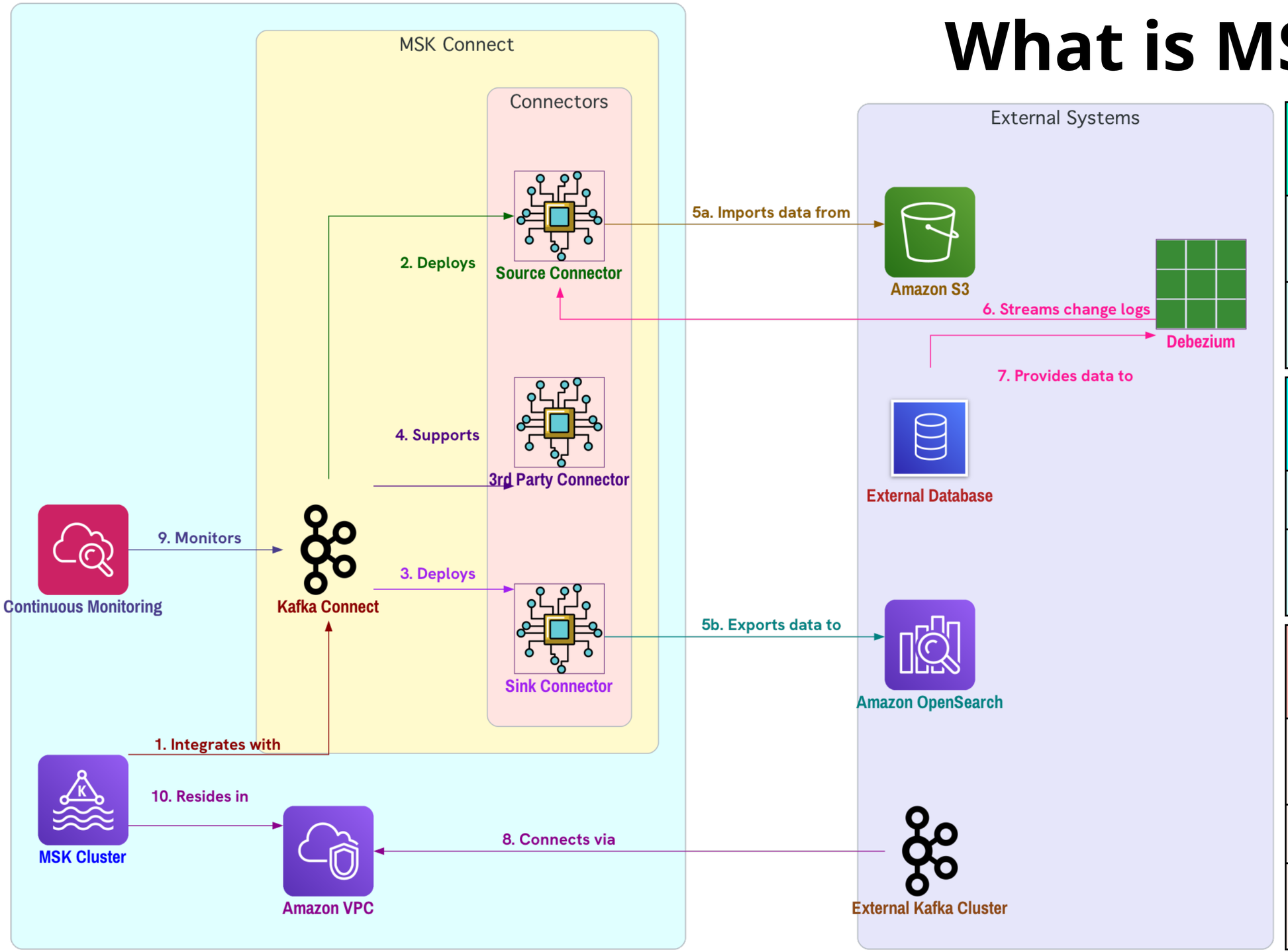

Integrated Services
Kafka Client Application
AWS PrivateLink
AWS IAM
AWS Glue Schema Registry
Amazon Managed Service for Apache Flink
AWS Lambda
4. Produces/Consumes data
5. Provides private connectivity
6. Authenticates and authorizes
7. Manages schemas
8. Processes streams
9. Processes events
MSK Serverless Cluster
MSK Serverless
Auto-managed Resources
1. Runs
2. Auto-manages
3. Auto-manages
Apache Kafka
Auto-scaled Capacity
Managed Partitions
3️⃣. 💰 Pay-per-use pricing model
💹 Throughput-based pricing
💸 Pay only for what you use
📉 Cost savings for variable workloads
4️⃣. 🔌 Apache Kafka compatibility
🤝 Fully compatible with Apache Kafka
🔗 Use compatible client applications
🔁 Seamless integration with existing systems



5️⃣ . 🛠️ **Key integrations**:

- 🔒 AWS PrivateLink: 🔐 Private connectivity, 🛡️ Enhanced security
- 🔐 AWS IAM: 🔑 Authentication and authorization, ☕ Supports Java and non-Java languages
- 📚 AWS Glue Schema Registry: 📋 Schema management, 🔄 Maintain data structure consistency
- 🌊 Amazon Managed Service for Apache Flink: 🔄 Stream processing, ⚡ Real-time analysis and action
- λ AWS Lambda: 🎭 Event processing, 🔔 Trigger serverless functions

What is MSK Connect?



1️⃣. 🚀 Easy data streaming for Apache Kafka clusters

🔄 Simplifies data streaming process

👷 User-friendly data flow management

2️⃣. 🔄 Uses Kafka Connect 2.7.1 framework

🔒 Open-source framework

🔌 Connects Kafka with external systems

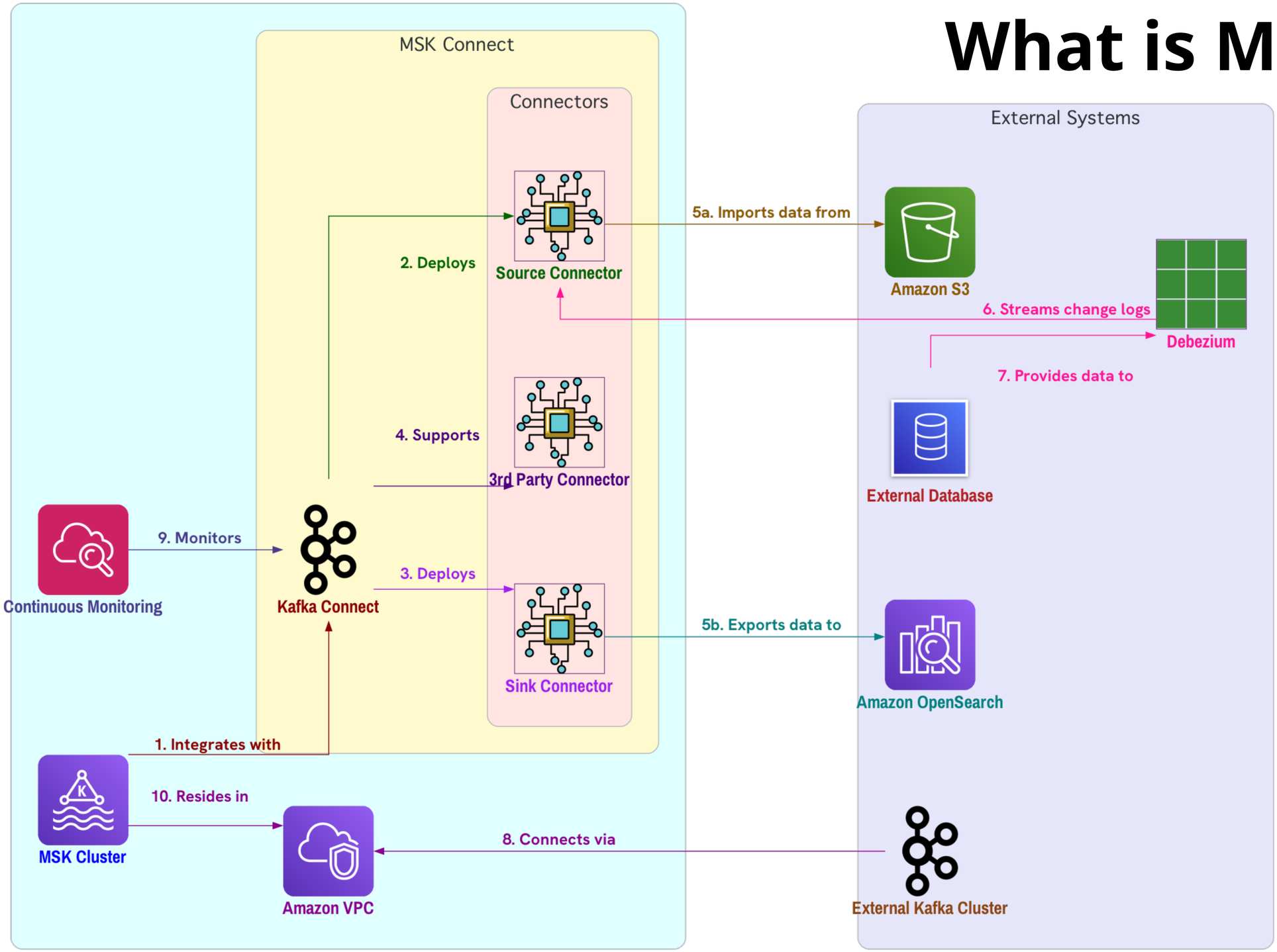
3️⃣. 🔌 Fully managed connectors available

🔁 Move data to/from popular data stores

📦 Amazon S3 integration

🔍 Amazon OpenSearch Service integration

What is MSK Connect?



4️⃣. 📩 Source connectors for data import

🔽 Import data from external systems

🔀 Bring data into Kafka ecosystem

5️⃣. 📩 Sink connectors for data export

🔼 Export data to external systems

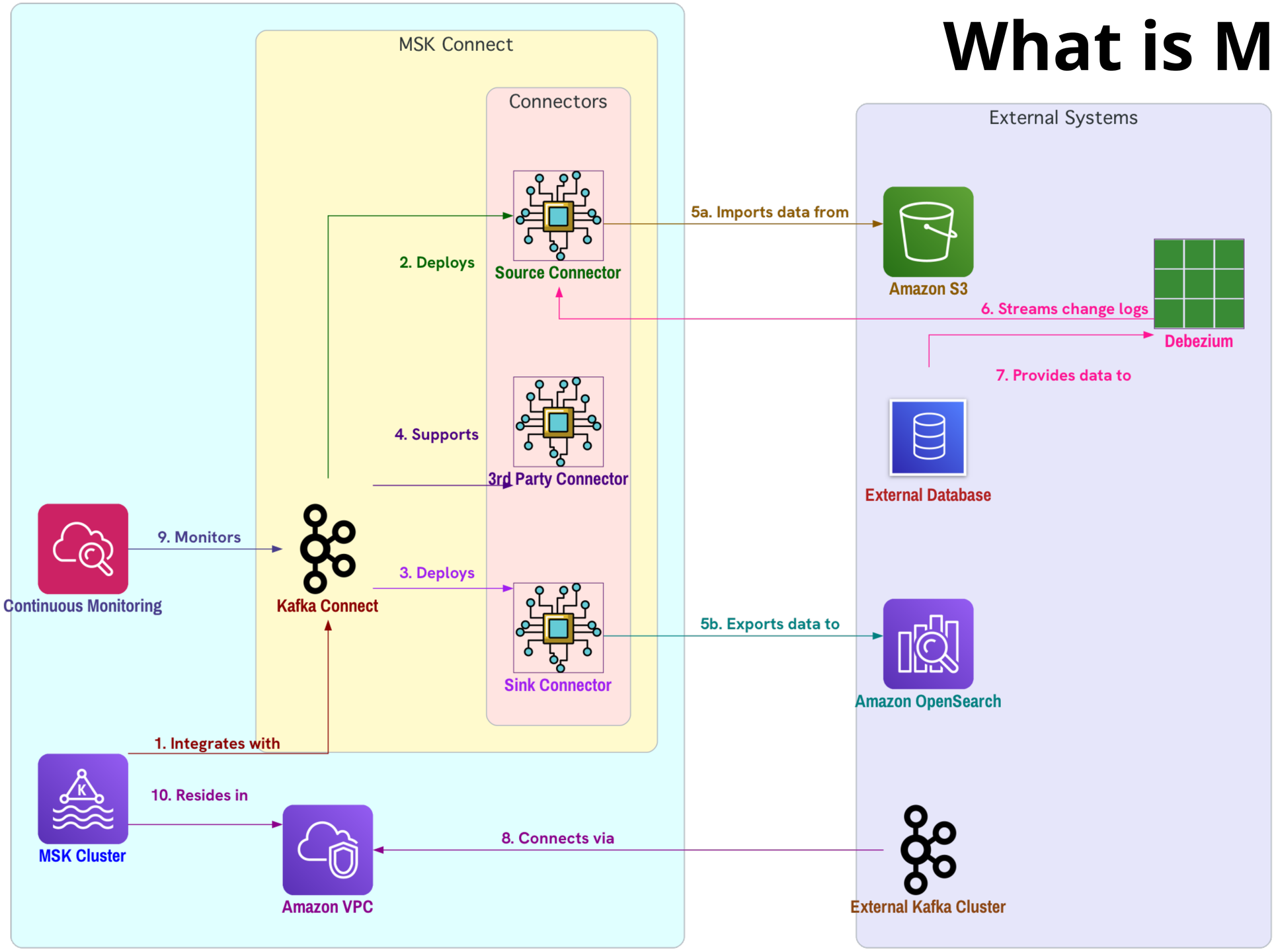
📊 Distribute processed data

6️⃣. 🔧 3rd party connector support

🔌 Debezium for change log streaming

♻️ Deploy existing connectors without changes

What is MSK Connect?



7️⃣. 📊 Automatic scaling and resource management

⚖️ Adjusts for load changes

💰 Pay only for used resources

8️⃣. 🖥️ Supports any Kafka cluster in Amazon VPC

🔗 Works with MSK clusters

🏠 Works with independently hosted clusters

9️⃣. 🔍 Continuous monitoring and maintenance

🏥 Monitors connector health

🔧 Handles patching and hardware management

🚀 Reduces operational burden

Thanks for Watching