
AWS
re:Invent

ANT317

Optimize Amazon Elasticsearch Service costs with UltraWarm

Jon Handler

Principal Solutions Architect, Search Services

AWS

Kuria Robert Kamau

Platform Engineer, Product Division

PayPay Corporation

AWS re:Invent

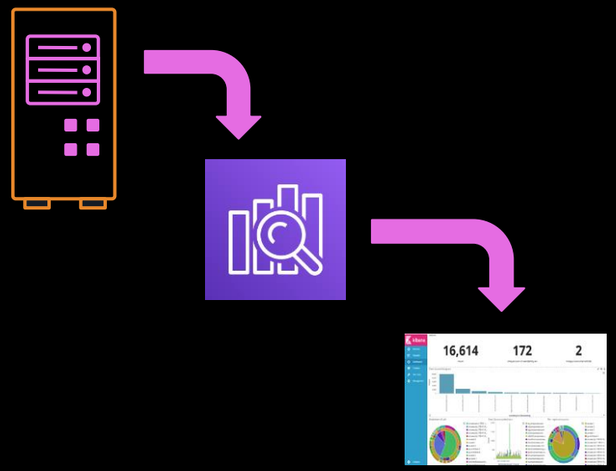
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon Elasticsearch Service (Amazon ES)

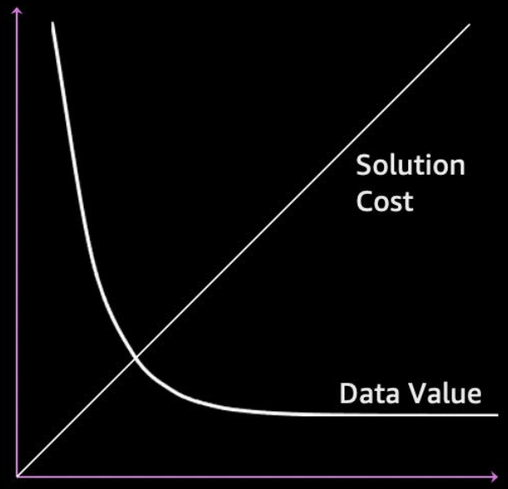
STORE, SEARCH, AND ANALYZE ALL YOUR LOGS



Amazon Elasticsearch Service is a fully managed service that makes it simple to deploy, scale, monitor, and secure Elasticsearch in the AWS cloud



Use Amazon Elasticsearch Service and the bundled Kibana to analyze log data in near real time to monitor your infrastructure, application, security, IoT, and AWS services



Challenge: The value of data decreases rapidly, but the cost increases linear with data stored; UltraWarm dramatically reduces cost for long-term storage and analysis of data

UltraWarm for Amazon Elasticsearch Service



Extend your data retention for long-term analysis



Use the same REST APIs and Kibana dashboards

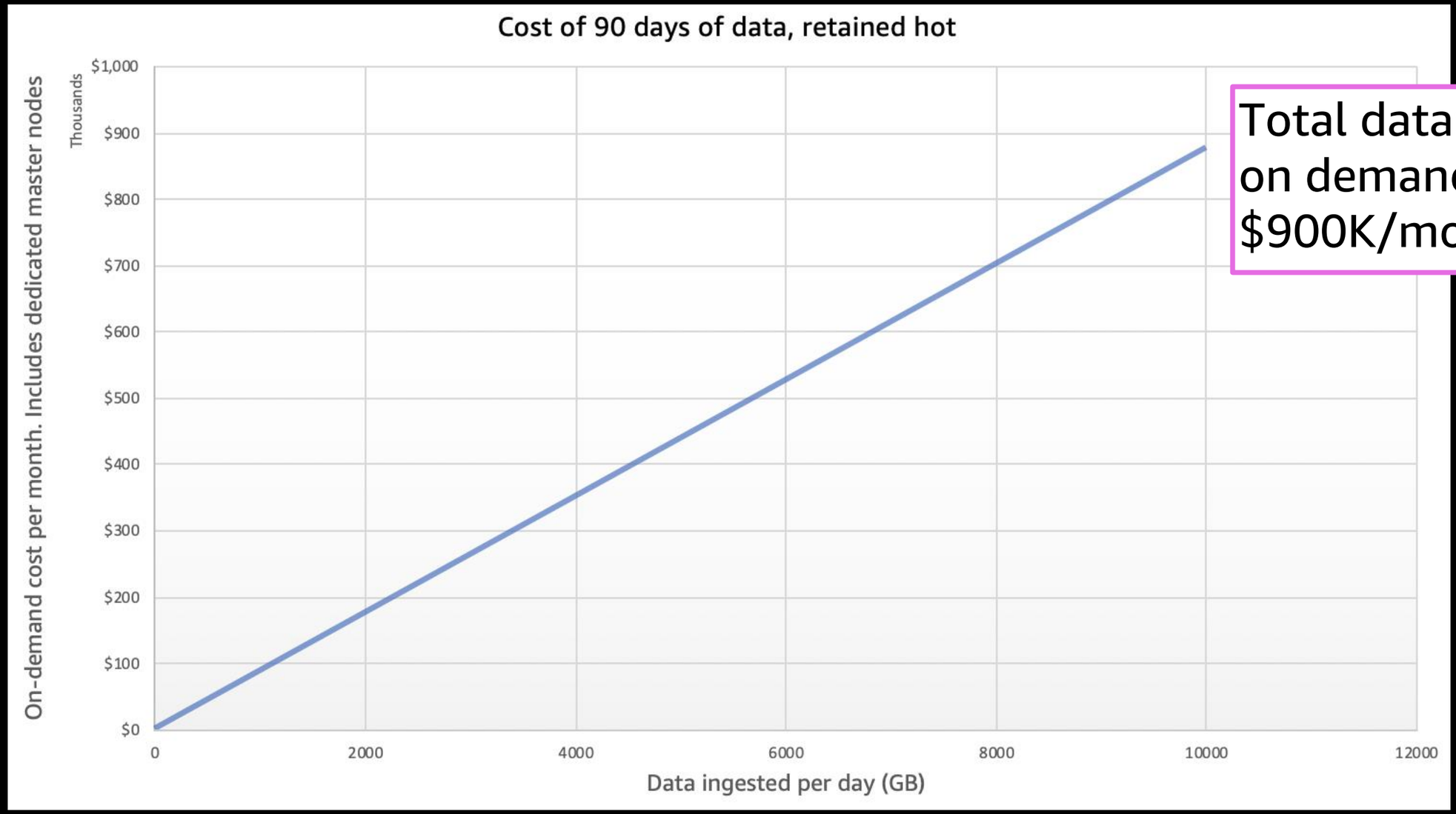


Trade latency and cost for an optimum balance



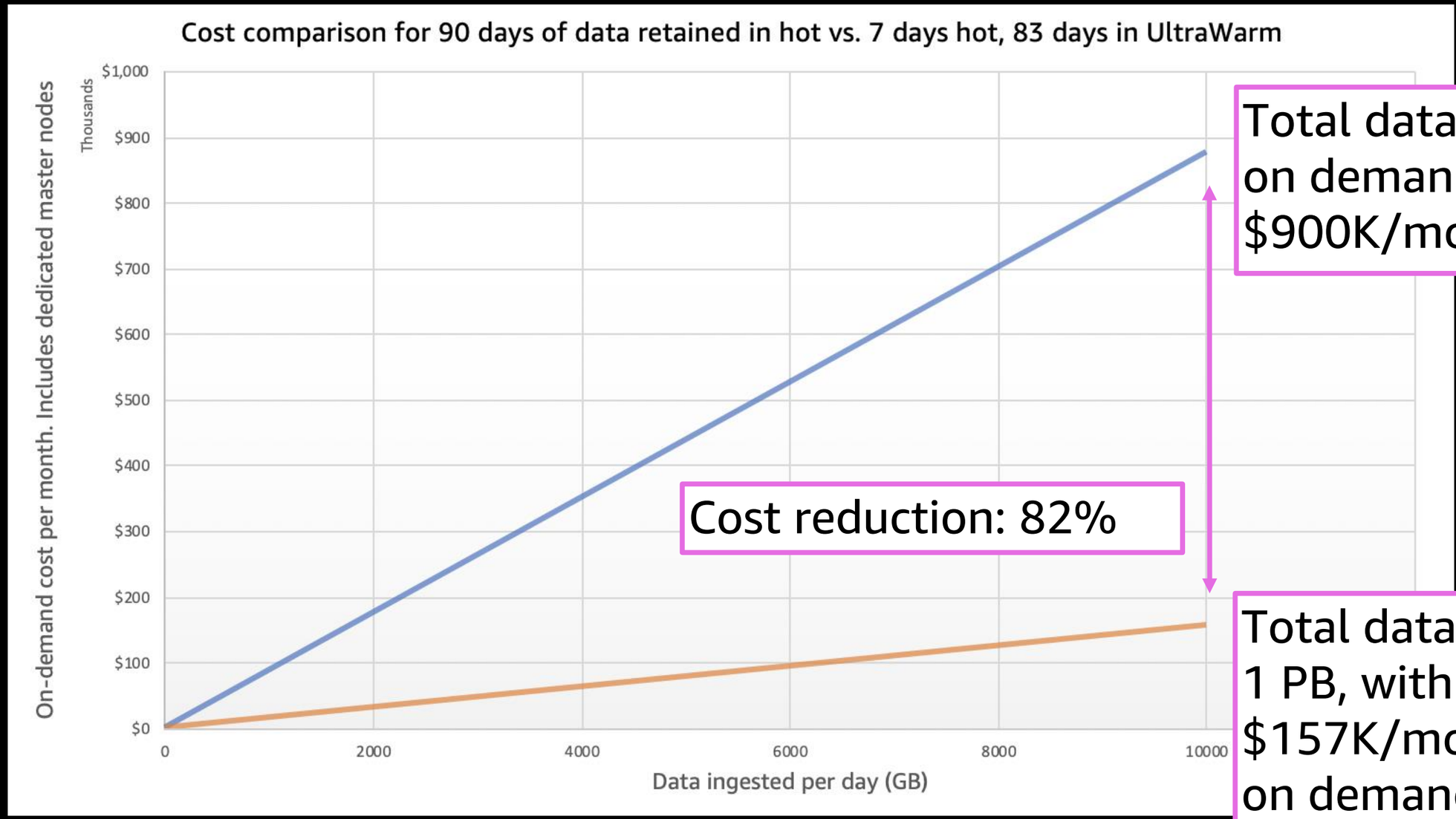
Achieve substantial cost savings vs. hot storage

UltraWarm cost savings



Total data: 1 PB, on demand $900K/mo.

UltraWarm cost savings



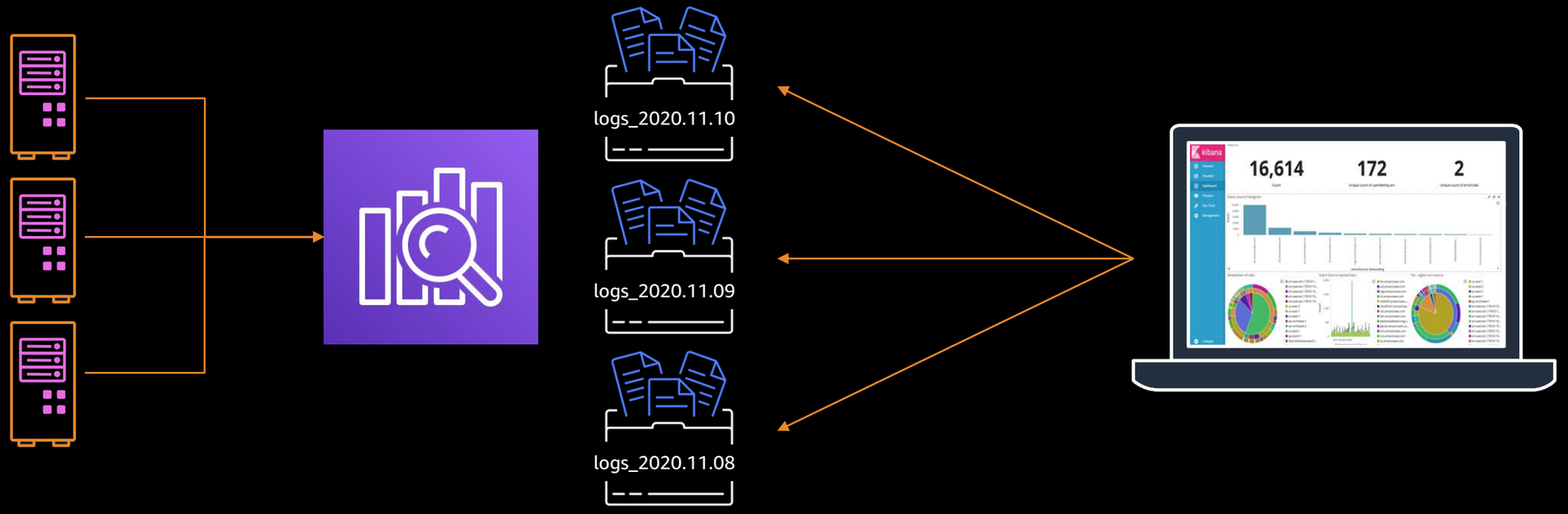
Working with UltraWarm

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

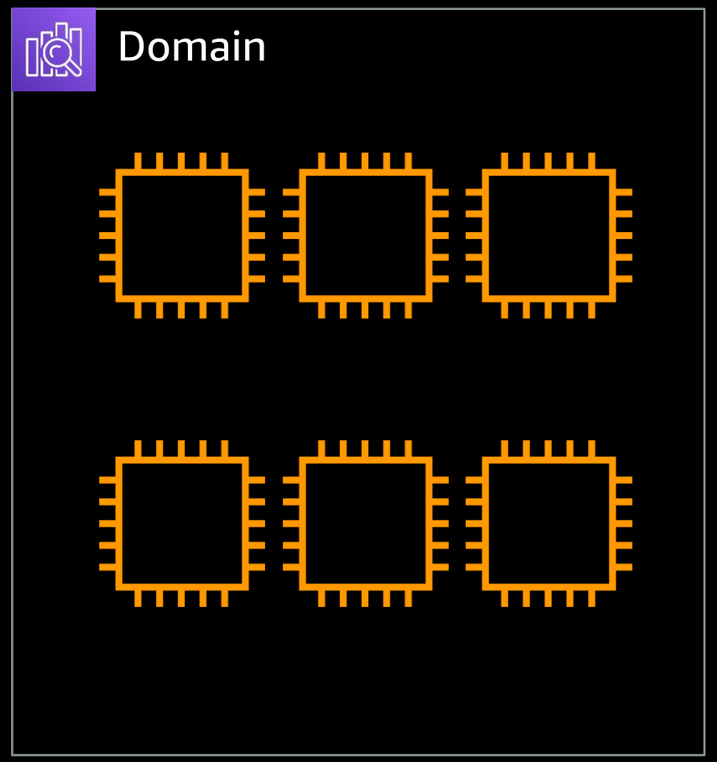
Best practice data lifecycle

CREATE A ROLLING INDEX PATTERN



You send log data to Amazon Elasticsearch Service
Elasticsearch uses indexes to organize searchable / analyzable data
You create indexes on time (or size) boundaries
Kibana and REST API access the data based on the collection of indices
You delete the oldest index at the end of its lifecycle

Processing requests in Amazon ES



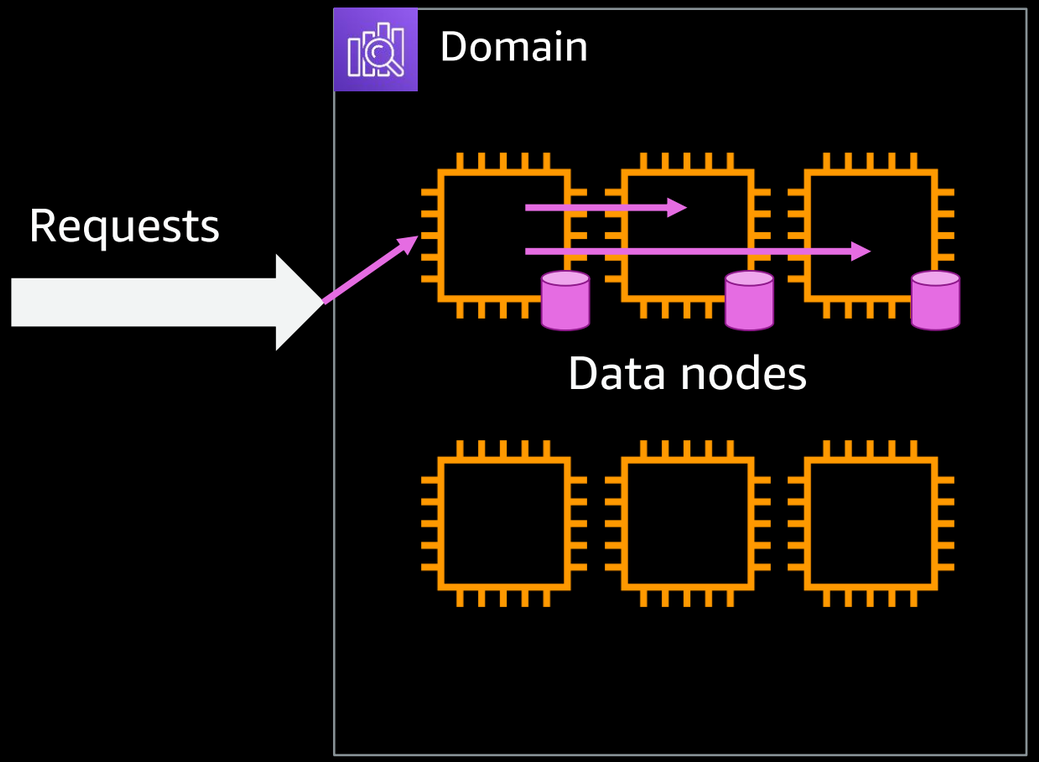
Amazon ES creates an Elasticsearch cluster, with Amazon EC2 instances

Data nodes are the workers, they index and analyze data

Dedicated leader nodes maintain the cluster state

Data nodes use instance or network-attached storage

Processing requests in Amazon ES



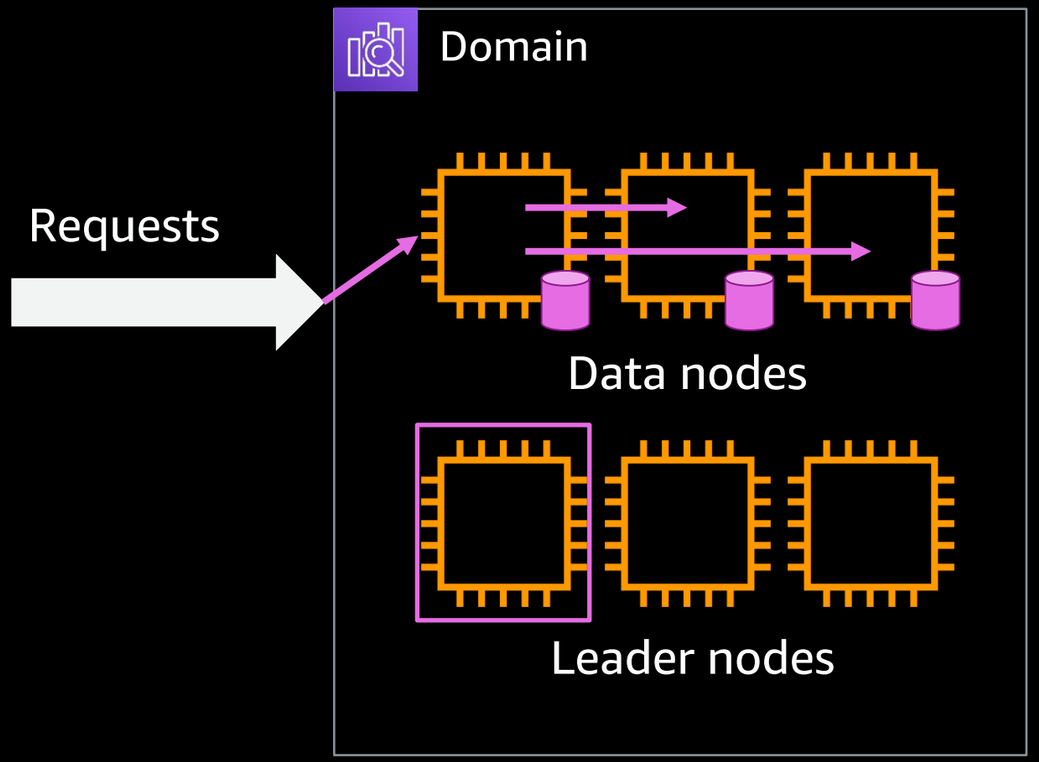
Amazon ES creates an Elasticsearch cluster, with Amazon EC2 instances

Data nodes are the workers, they index and analyze data

Dedicated leader nodes maintain the cluster state

Data nodes use instance or network-attached storage

Processing requests in Amazon ES



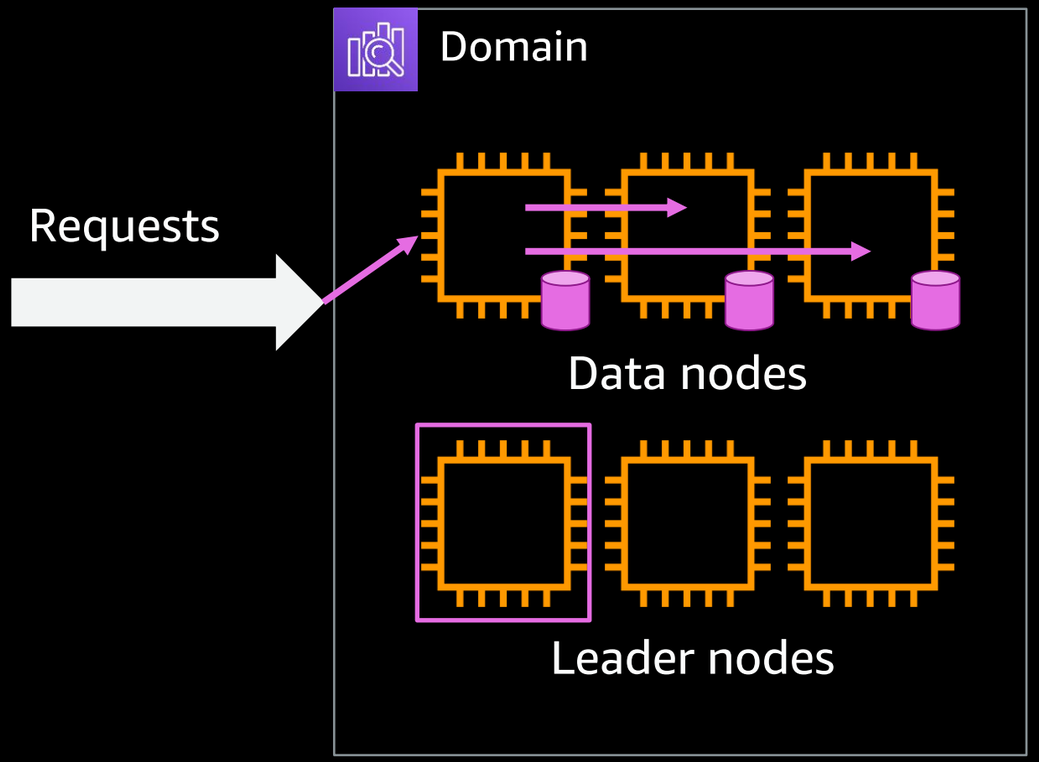
Amazon ES creates an Elasticsearch cluster, with Amazon EC2 instances

Data nodes are the workers, they index and analyze data

Dedicated leader nodes maintain the cluster state

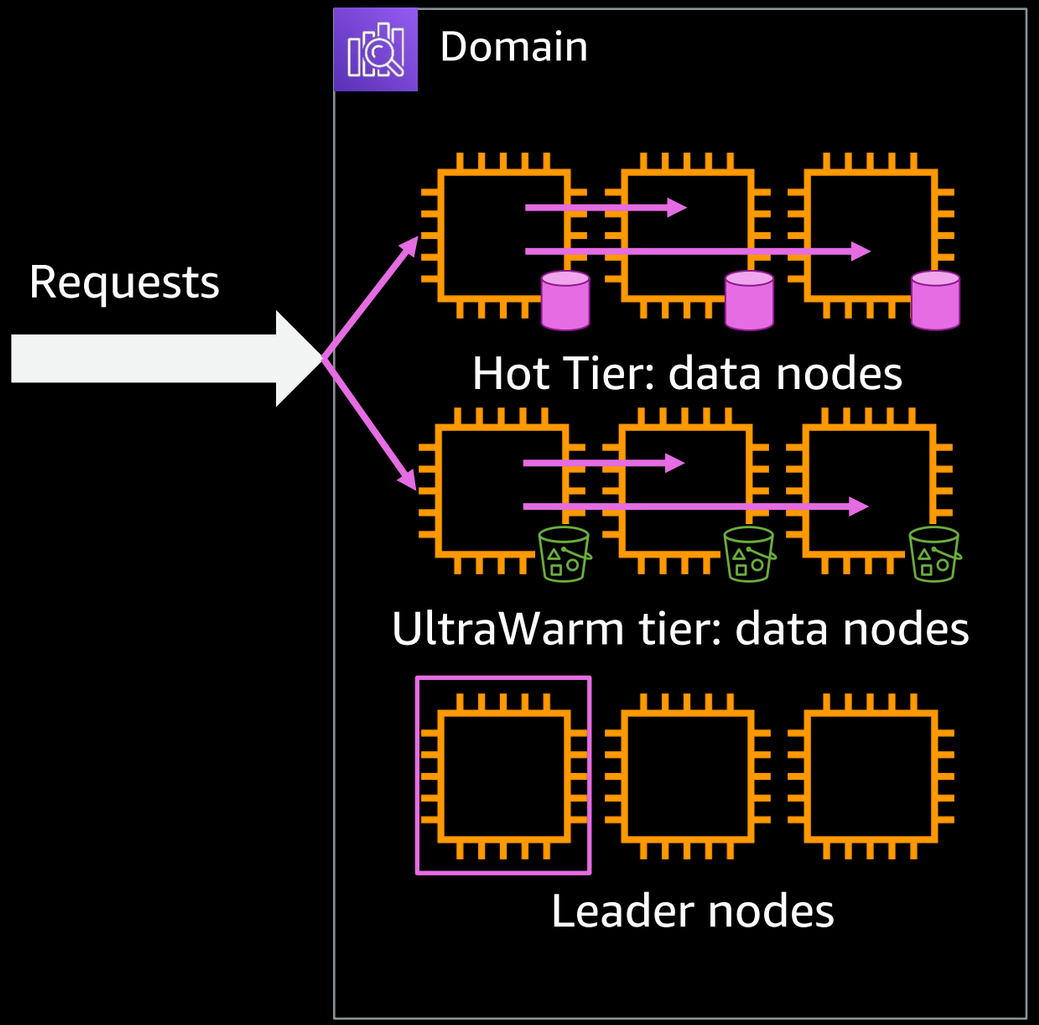
Data nodes use instance or network-attached storage

Processing requests in Amazon ES



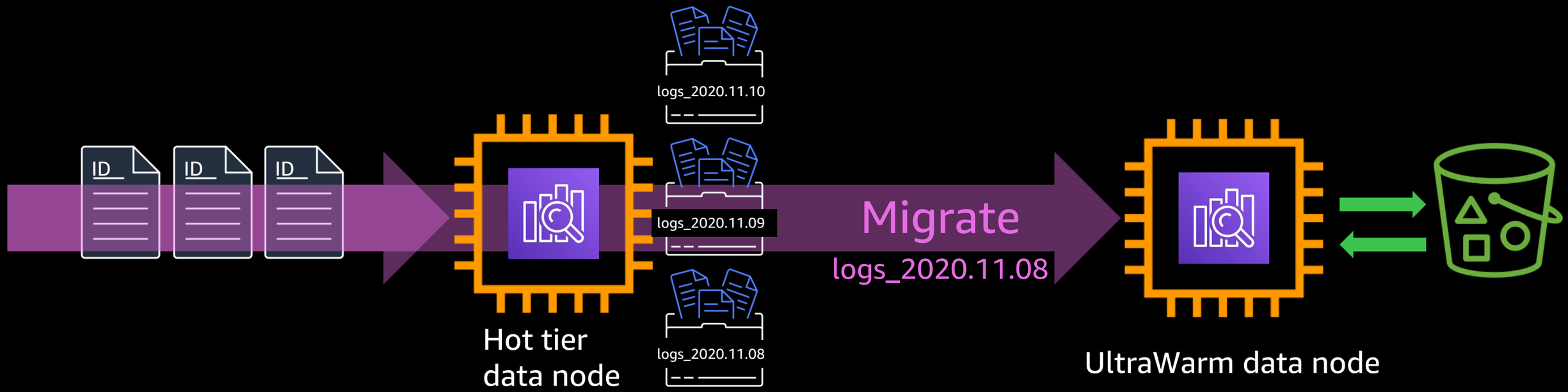
- Amazon ES creates an Elasticsearch cluster, with Amazon EC2 instances
- Data nodes are the workers, they index and analyze data
- Dedicated leader nodes maintain the cluster state
- Data nodes use instance or network-attached storage

Processing requests with UltraWarm



- UltraWarm nodes are data nodes in the Elasticsearch cluster
- UltraWarm stores indexes in Amazon S3
- Elasticsearch routes requests seamlessly

Elasticsearch data lifecycle



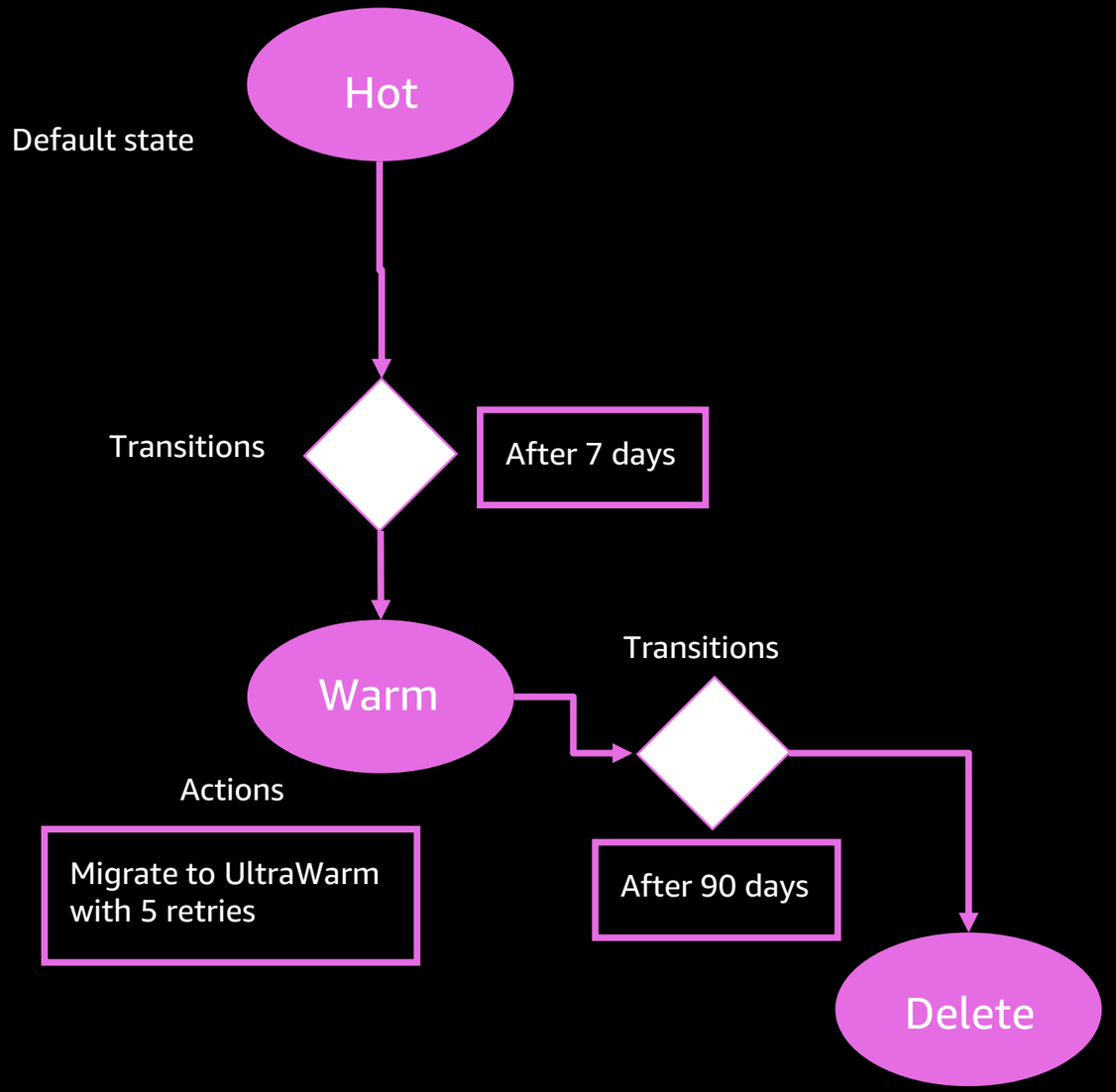
Data is indexed and stored in the hot tier, with a rolling index pattern

You migrate a single index via API or Index State Management

Force merge, drop replicas, snapshot during migration

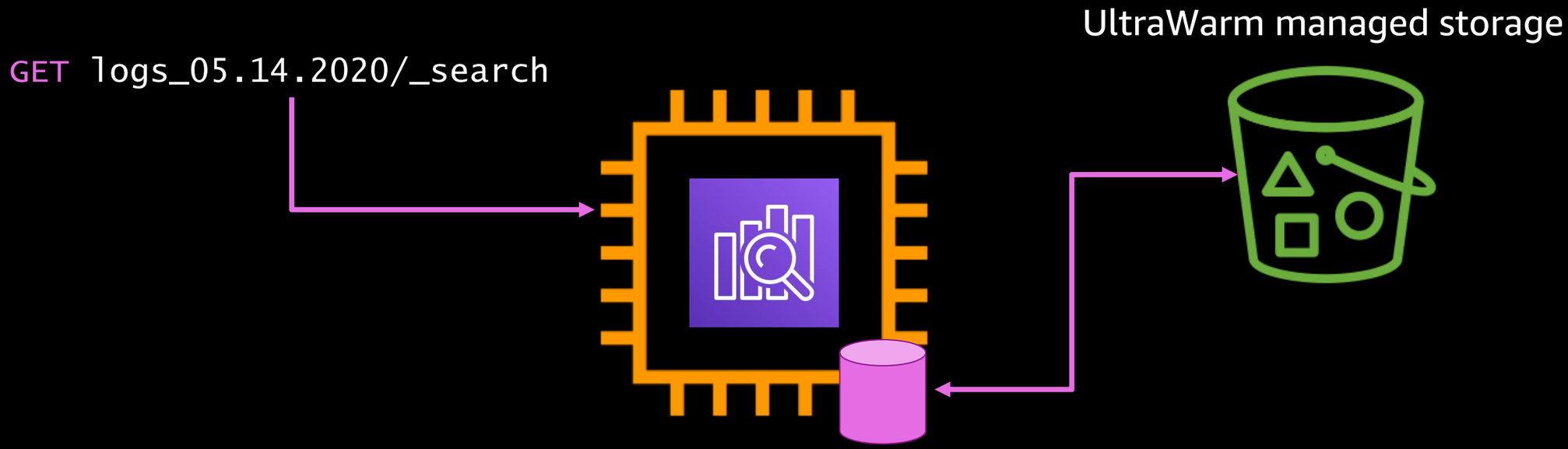
Index State Management

```
{ "hot_warm": {
  "description": "Demonstrate a hot-warm-delete workflow.",
  "default_state": "hot",
  "schema_version": 1,
  "states": [ {
      "name": "hot",
      "actions": [],
      "transitions": [ {
        "state_name": "warm",
        "conditions": { "min_index_age": "7d" } } ]
    },
    {
      "name": "warm",
      "actions": [ {
        "warm_migration": {},
        "retry": { "count": 5, "delay": "1h" } } ],
      "transitions": [ {
        "state_name": "delete",
        "conditions": { "min_index_age": "90d" } } ]
    },
    {
      "name": "delete",
      "actions": [
      {
        "notification": {
          "destination": { "chime": { "url": "<URL>" } },
          "message_template": { "source": "The index
{{ctx.index}} is
                                being deleted." } }
      },
      {
        "delete": {}
      } ] }
  ] } }
```



UltraWarm performance

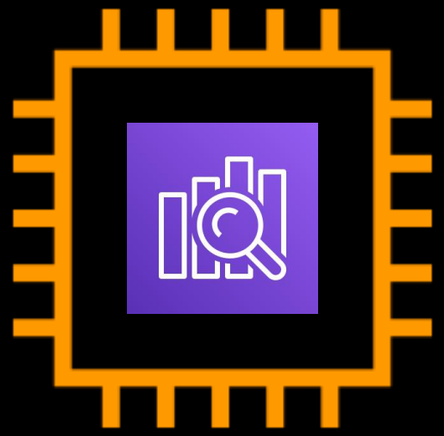
EXPECTED LATENCY DEPENDS ON CACHING AND AMOUNT OF DATA



Querying UltraWarm brings data to local storage (cache)

- If data is already cached, query performance is like hot: O(ms)
- Otherwise, query latency is dominated by Amazon S3 round trip
  - Small slices of data – expected latency is O(s)
  - PB-scale queries – expected latency is O(m)

UltraWarm sizing

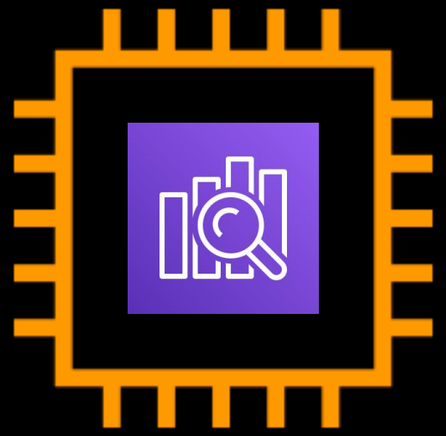


UltraWarm1.medium.elasticsearch

Manages up to 1.5 TB of primary index

2 vCPU, 400 shards maximum

475 GB of local storage



UltraWarm1.large.elasticsearch

Manages up to 20 TB of primary index

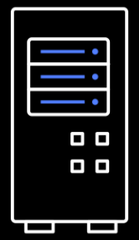
16 vCPU, 1,000 shards maximum

3.8 TB of local storage

Other cost considerations

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Cost model



Instance hours

You choose the instances to deploy for the data nodes in your cluster

Amazon ES charges per hour for these instances

Largest cost component



Amazon EBS

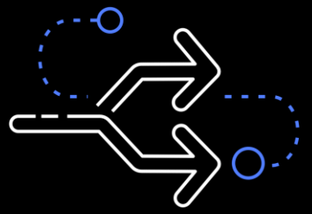
If you choose Amazon EBS-backed instances, Amazon ES charges per GB of data deployed



Amazon S3

When you use UltraWarm, Amazon ES charges for actual usage of Amazon S3

Drives instance count



Data transfer

Amazon ES charges for data transfer at Amazon EC2 data transfer rates

Managing cost

Use Reserved Instances to reduce cost

Send only data you will use

- Especially look at automated tools like Logstash, which preserves the @message
- If you aren’t searching or analyzing it, don’t send it
- Ideally, store and retrieve only aggregation data or primary store keys

Evaluate ROI in additional redundancy

Adjust your mapping

- Use a _template to set an explicit _mapping and disable _dynamic fields
- Remove _source
- By default, text fields also create a .keyword field – do you need it?

How long do you need your data?

Managing cost

Use Reserved Instances to reduce cost

Send only data you will use

- Especially look at automated tools like Logstash, which preserves the @message
- If you aren’t searching or analyzing it, don’t send it
- Ideally, store and retrieve only aggregation data or primary store keys

Evaluate ROI in additional redundancy

Adjust your mapping

- Use a _template to set an explicit _mapping and disable _dynamic fields
- Remove _source
- By default, text fields also create a .keyword field – do you need it?

How long do you need your data?

Managing cost

Use Reserved Instances to reduce cost

Send only data you will use

- Especially look at automated tools like Logstash, which preserves the @message
- If you aren't searching or analyzing it, don't send it
- Ideally, store and retrieve only aggregation data or primary store keys

Evaluate ROI in additional redundancy

Adjust your mapping

- Use a _template to set an explicit _mapping and disable _dynamic fields
- Remove _source
- By default, text fields also create a .keyword field – do you need it?

How long do you need your data?

Managing cost

Use Reserved Instances to reduce cost

Send only data you will use

- Especially look at automated tools like Logstash, which preserves the @message
- If you aren't searching or analyzing it, don't send it
- Ideally, store and retrieve only aggregation data or primary store keys

Evaluate ROI in additional redundancy

Adjust your mapping

- Use a _template to set an explicit _mapping and disable _dynamic fields
- Remove _source
- By default, text fields also create a .keyword field – do you need it?

How long do you need your data?

Managing cost

Use Reserved Instances to reduce cost

Send only data you will use

- Especially look at automated tools like Logstash, which preserves the @message
- If you aren't searching or analyzing it, don't send it
- Ideally, store and retrieve only aggregation data or primary store keys

Evaluate ROI in additional redundancy

Adjust your mapping

- Use a _template to set an explicit _mapping and disable _dynamic fields
- Remove _source
- By default, text fields also create a .keyword field – do you need it?

How long do you need your data?

UltraWarm anti-patterns and troubleshooting

UltraWarm indices are read only – use for workloads with no updates

Performance is proportional to shards and data accessed

Performance is bounded by shard:CPU

Migrations and snapshots are single-index concurrent and single shard per node – single shard indexes will migrate serially

Overscale UltraWarm for more performance

UltraWarm anti-patterns and troubleshooting

UltraWarm indices are read only – use for workloads with no updates

Performance is proportional to shards and data accessed

Performance is bounded by shard:CPU

Migrations and snapshots are single-index concurrent and single shard per node – single shard indexes will migrate serially

Overscale UltraWarm for more performance

UltraWarm anti-patterns and troubleshooting

UltraWarm indices are read only – use for workloads with no updates

Performance is proportional to shards and data accessed

Performance is bounded by shard:CPU

Migrations and snapshots are single-index concurrent and single shard per node – single shard indexes will migrate serially

Overscale UltraWarm for more performance

UltraWarm anti-patterns and troubleshooting

UltraWarm indices are read only – use for workloads with no updates

Performance is proportional to shards and data accessed

Performance is bounded by shard:CPU

Migrations and snapshots are single-index concurrent and single shard per node – single shard indexes will migrate serially

Overscale UltraWarm for more performance

UltraWarm anti-patterns and troubleshooting

UltraWarm indices are read only – use for workloads with no updates

Performance is proportional to shards and data accessed

Performance is bounded by shard:CPU

Migrations and snapshots are single-index concurrent and single shard per node – single shard indexes will migrate serially

Overscale UltraWarm for more performance

UltraWarm use case @PayPay

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Self-introduction

Kuria Robert Kamau
Platform Engineer
Product Division @PayPay

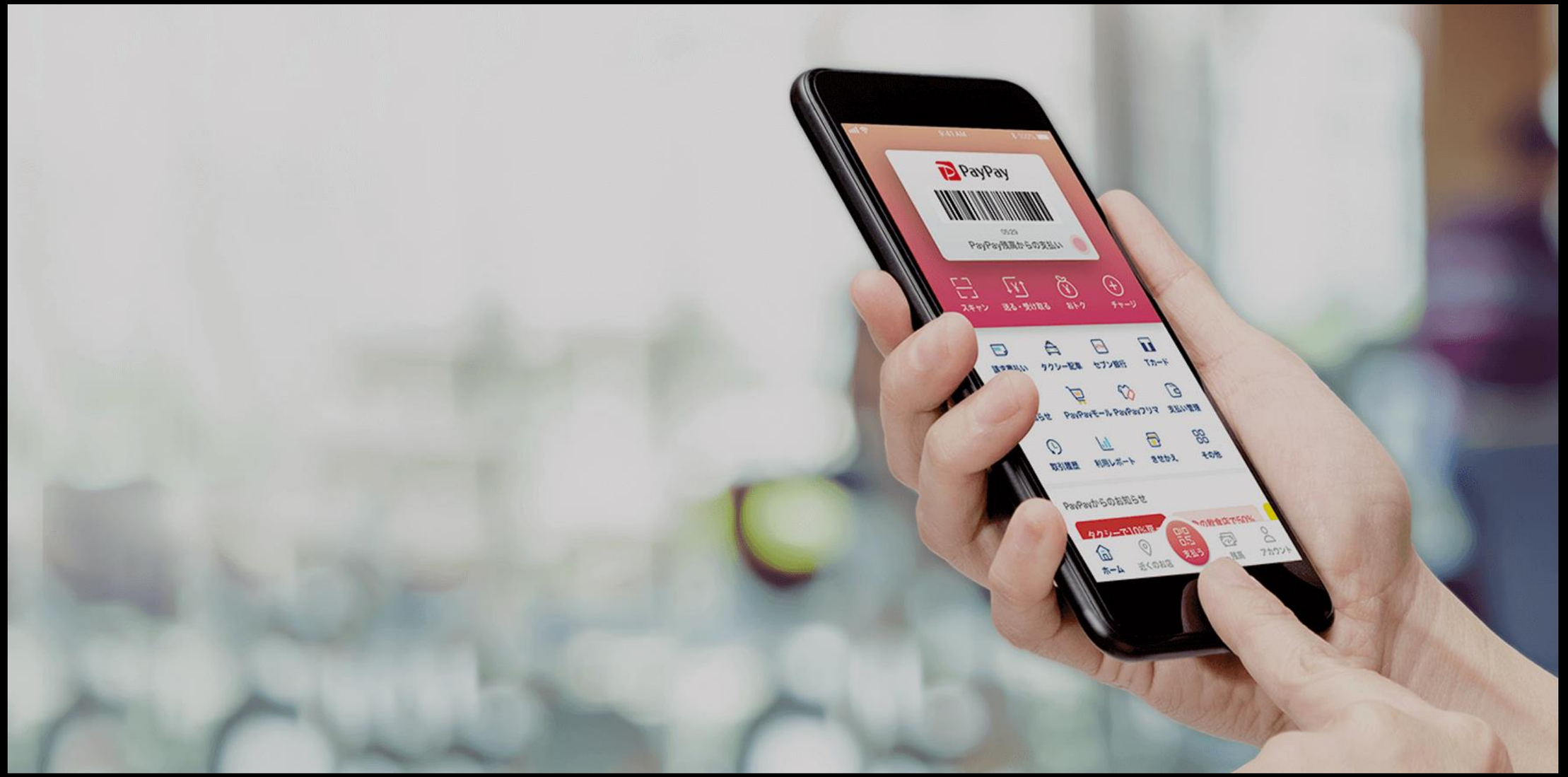


What is PayPay?

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Japan’s leading mobile payment app

LAUNCHED ON OCTOBER 5, 2018



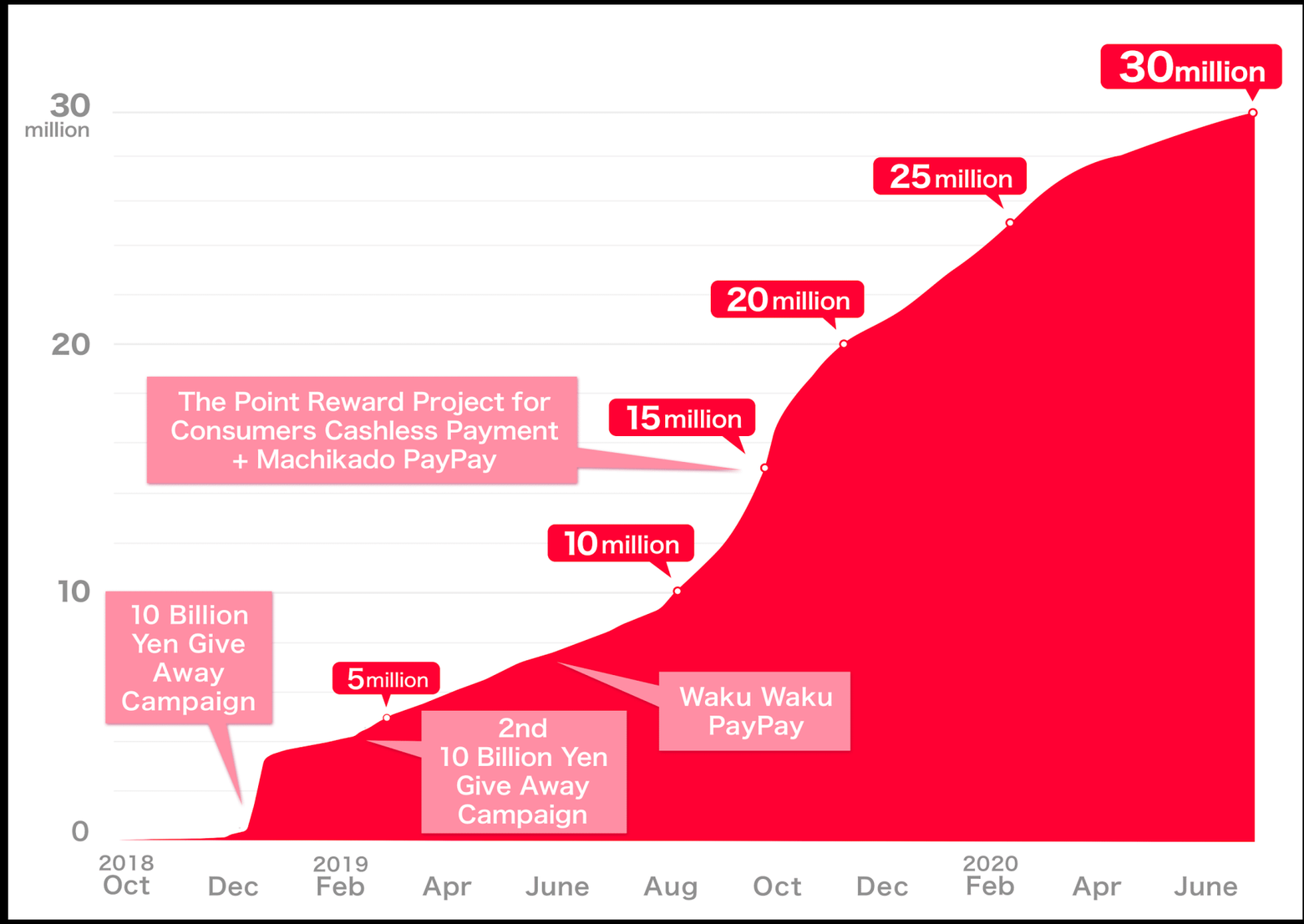
Online and offline payment services

- Online merchants
- Offline merchants
- Bill payments
- P2P
- PayPay pickup
- And more

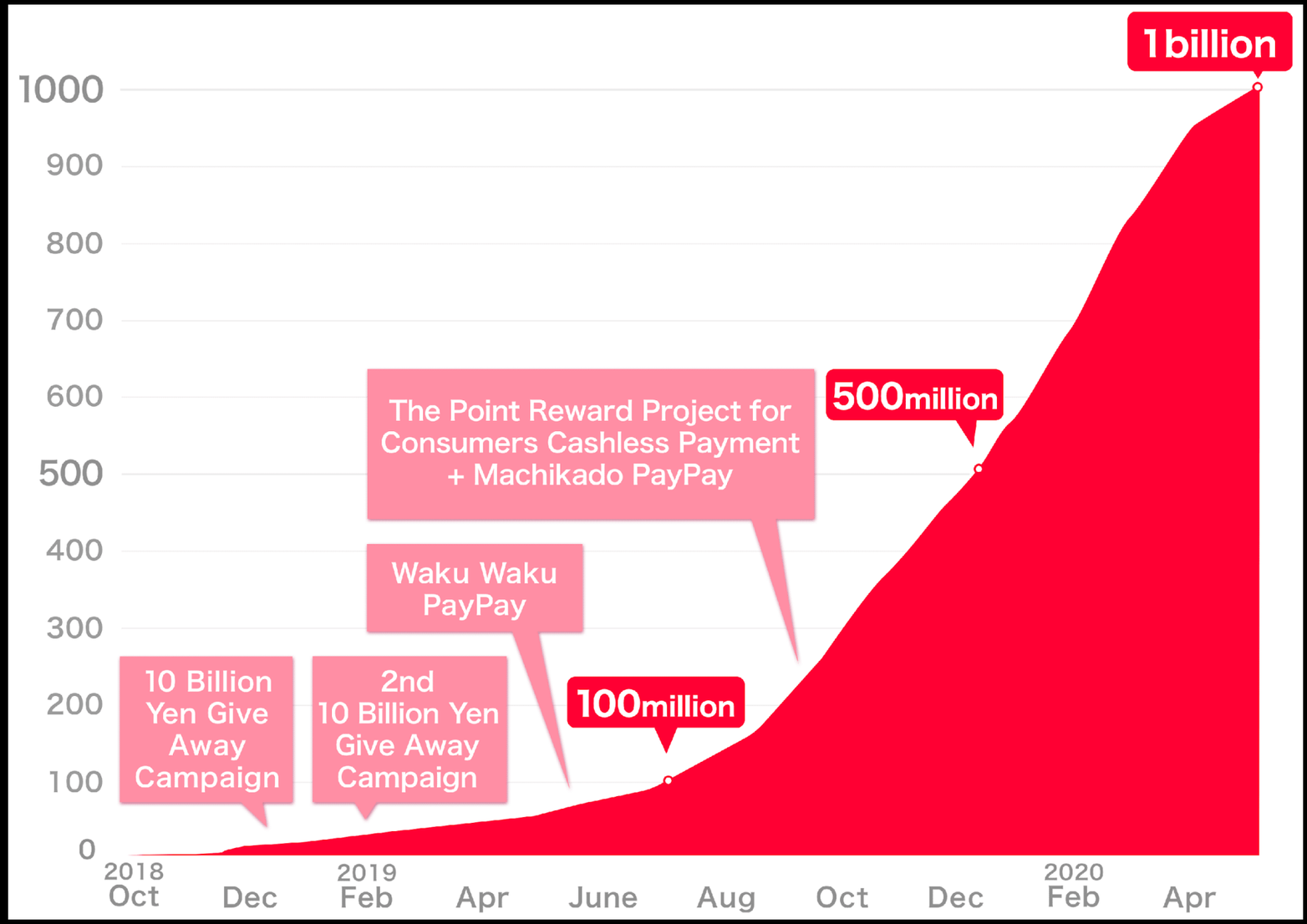
30 million users*

2.3 million stores*

*As of June 2020



Total of 1 billion transactions*



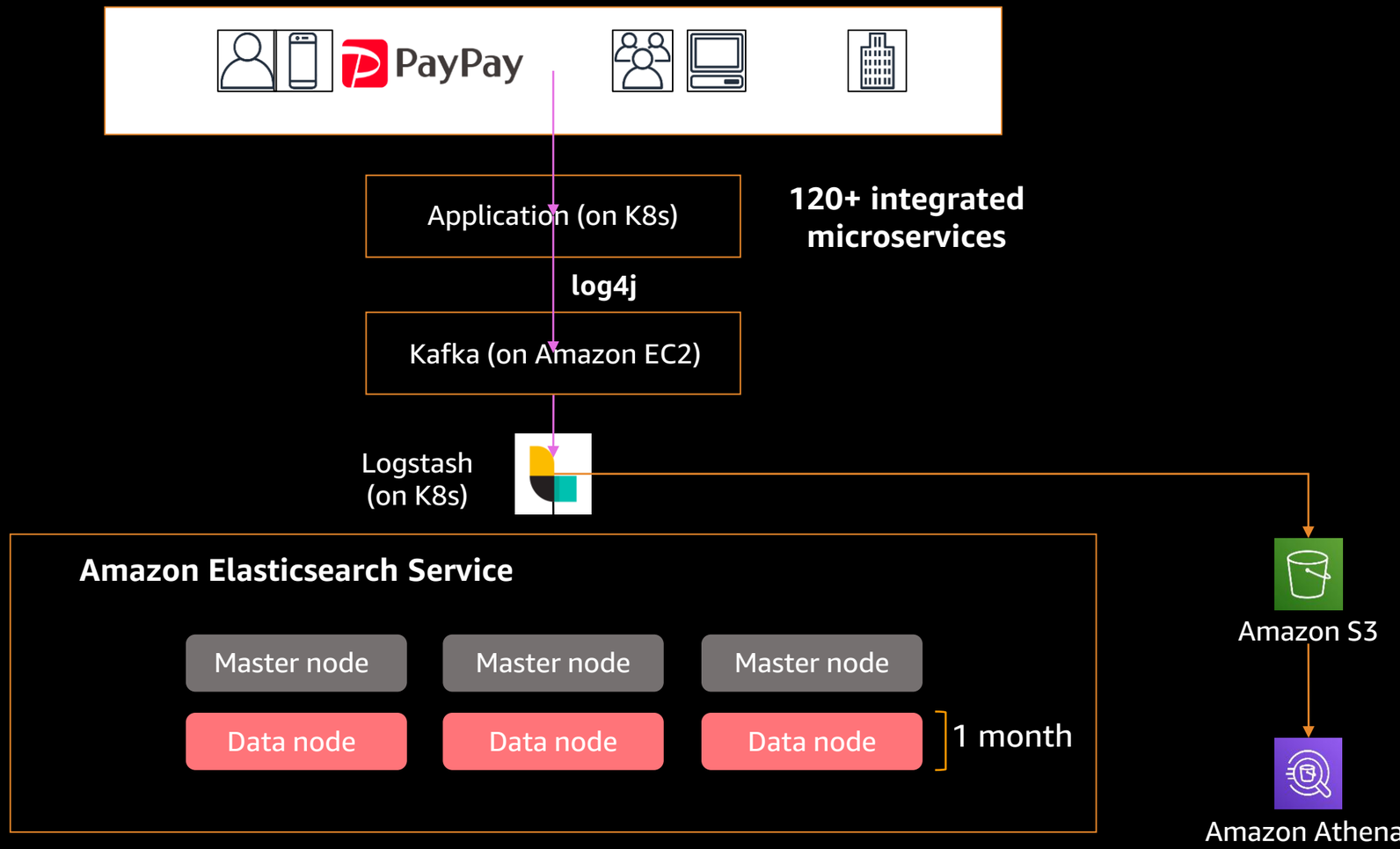
*As of May 2020

Logging with Amazon ES

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Platform architecture: Before UltraWarm



- We ingest logs to Amazon ES for near real-time log search and to Amazon S3 for archive search with Amazon Athena
- Application logs are for dev works debugging and also for customer support

Amazon ES cluster overview: Before UltraWarm

57 cluster nodes

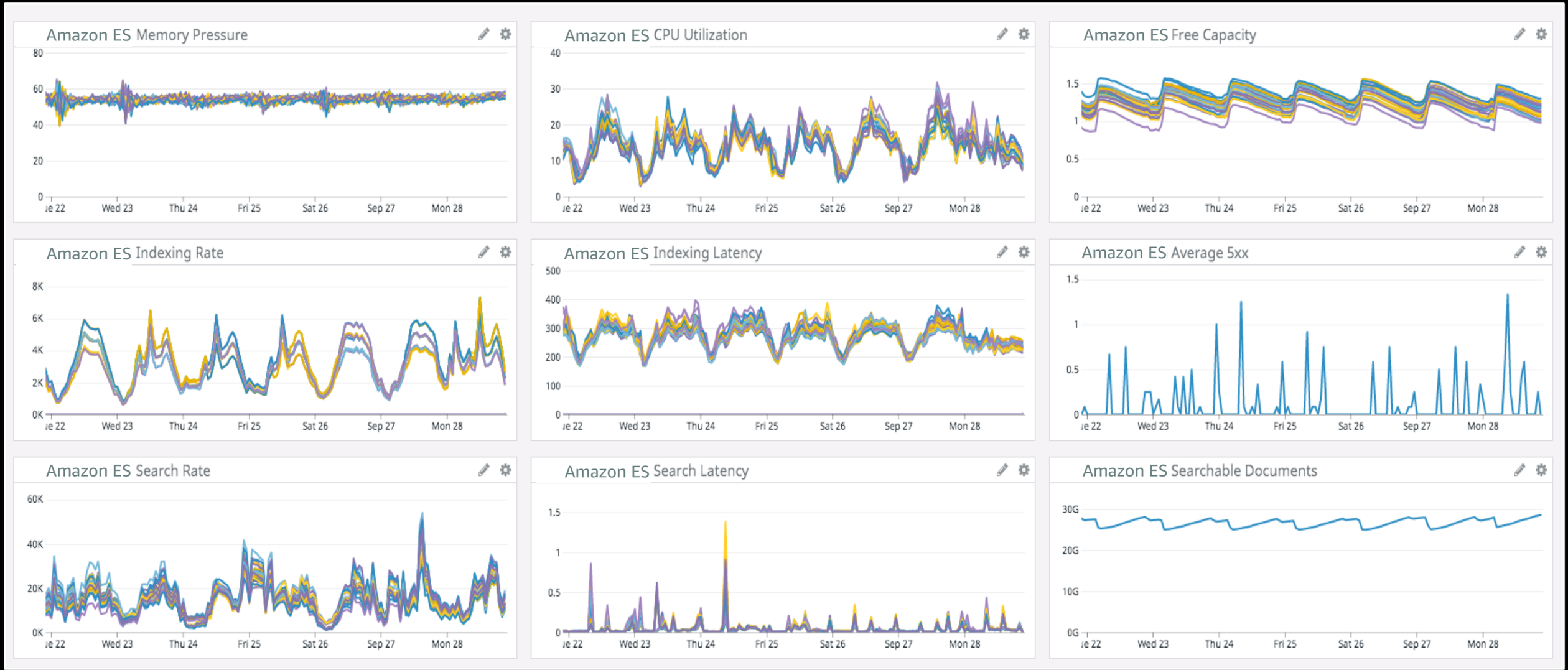
- 3 master nodes: c4.xlarge.elasticsearch
- 54 data nodes: r5.4xlarge.elasticsearch

329 TB cluster capacity (6,100 GB/data node)

Daily rotated index – approximately 2.8 TB/day (5.6 TB including 1 replica)

Housekeep with Curator

Amazon ES cluster monitoring



Index and shard configuration

TUNED FOR INDEXING PERFORMANCE

Primary shards =

No. of data nodes (54)

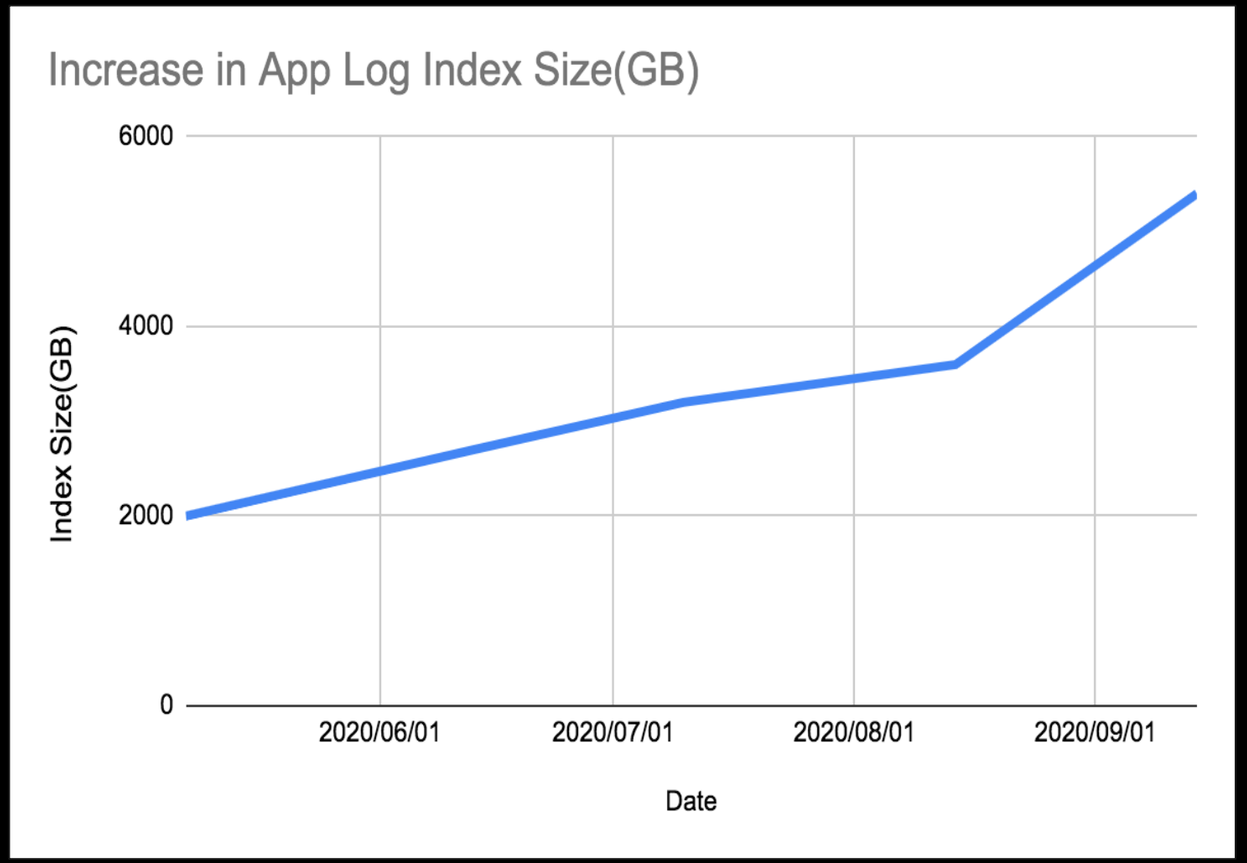
Evenly distributes writes on data nodes



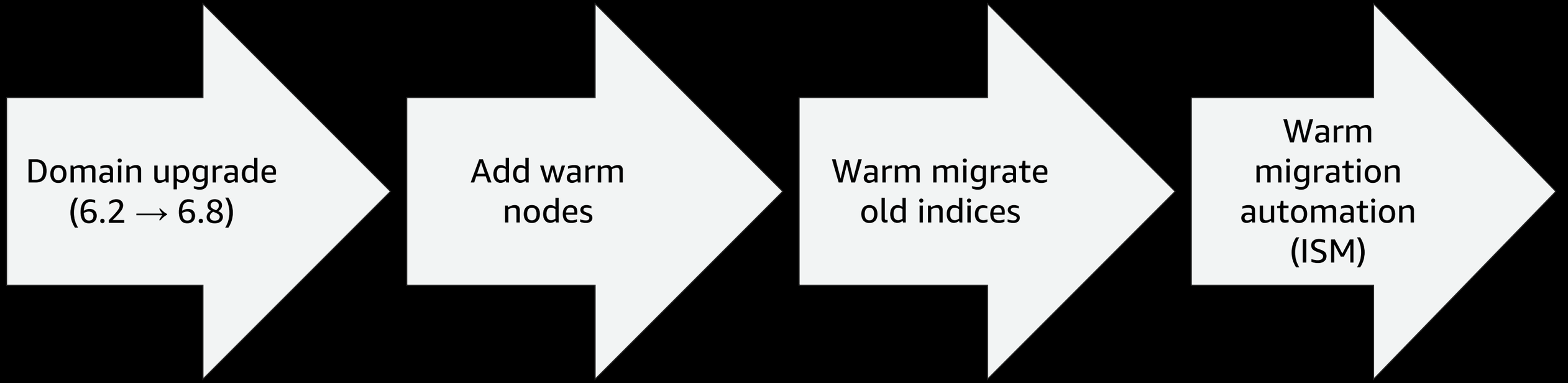
```
{
  "application_index_shard_tuning" : {
    "order" : 0,
    "index_patterns" : [
      "application-json-logs-*"
    ],
    "settings" : {
      "index" : {
        "number_of_shards" : "54",
        "translog" : {
          "flush_threshold_size" : "1024MB"
        },
        "refresh_interval" : "60s"
      }
    },
    "mappings" : {⟺},
    "aliases" : { }
  }
}
```

Challenges with increasing log volumes

- 600 GB/month in log increase
- Increased log management costs
- Delete logs from Amazon ES after 1 month
- Harder domain config / settings changes (Kibana downtime during blue-green deployments)



UltraWarm setup in our cluster



UltraWarm requires v6.8 or higher

Prepare separate domain as Kibana downtime interrupts service

15 nodes
ultrawarm1.large.elasticsearch

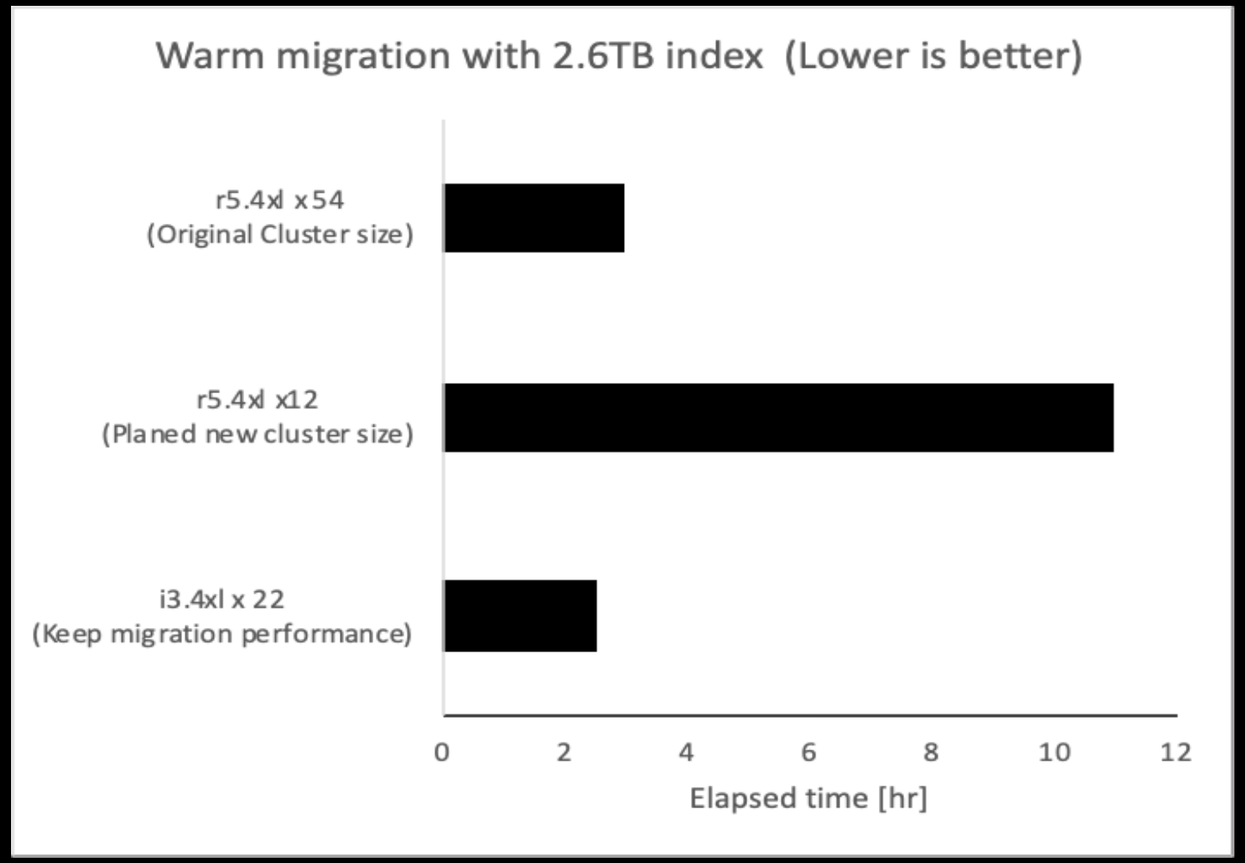
Manual migration

Avoid warm migration during business hours

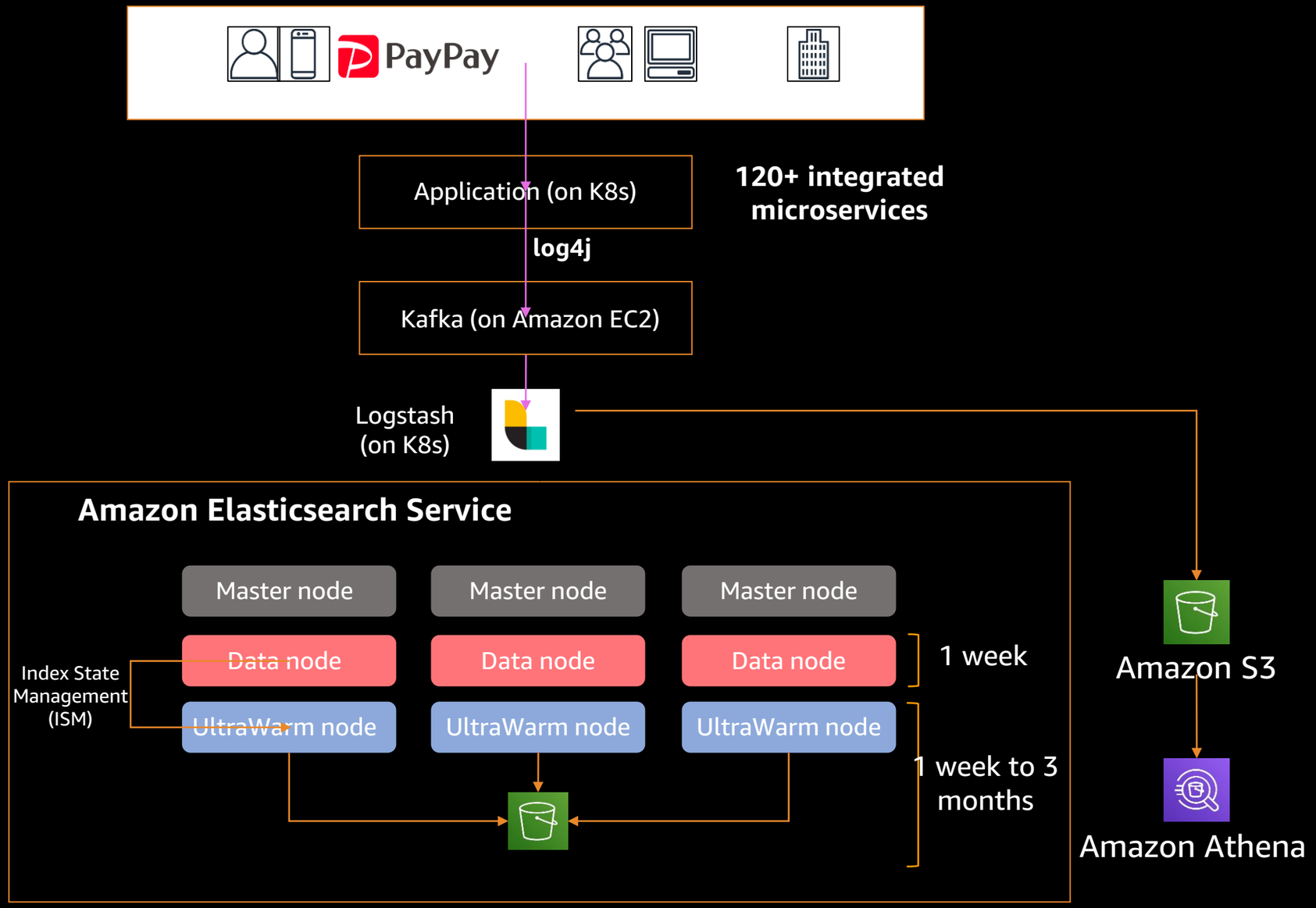
min_index_age setting: 180h (7.5days)

Improve warm migration performance

Speed up warm migration with an SSD-based i3 instance



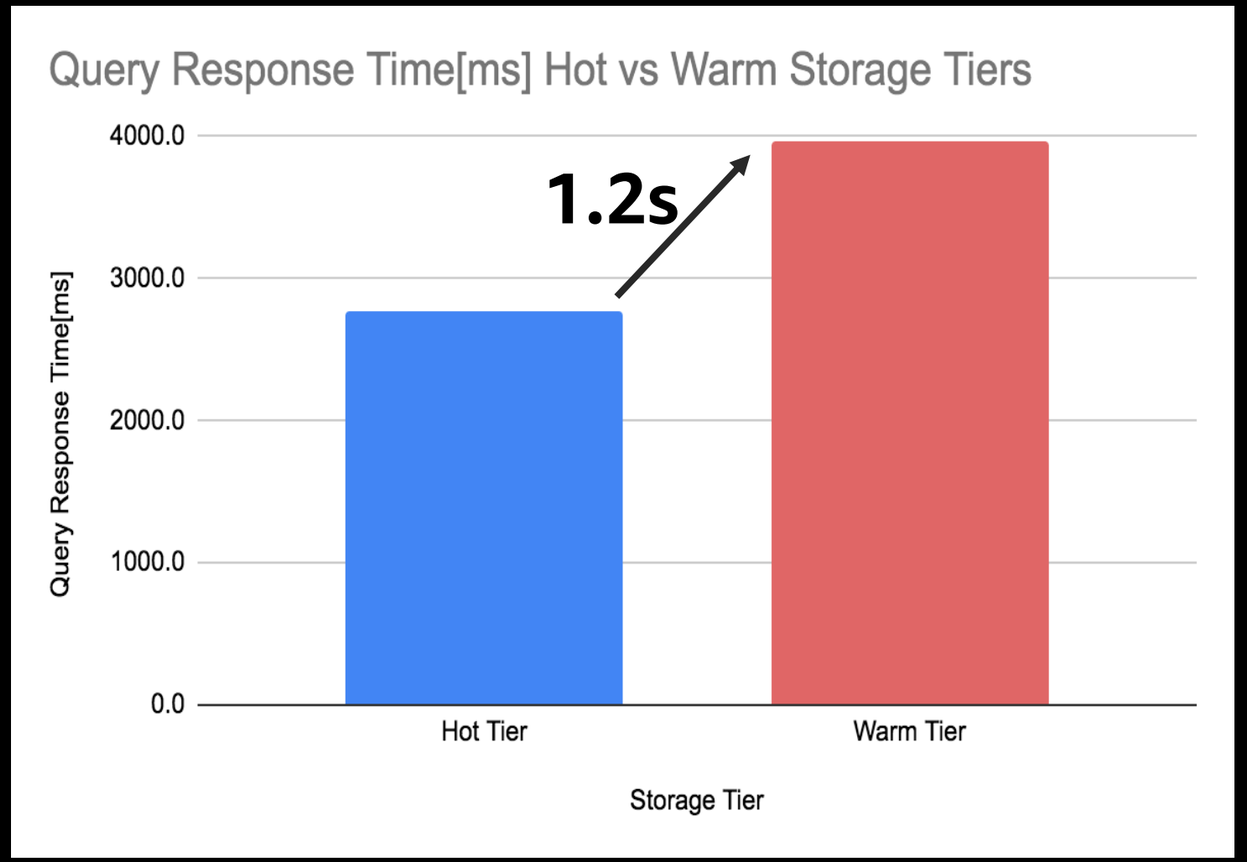
Platform architecture: With UltraWarm



Performance difference: Hot vs. warm storage tiers

```
POST /applog-<date>/_search?size=0
{
  "aggs": {
    "service_count": {
      "cardinality": {
        "field": "service.keyword",
        "precision_threshold": 500
      }
    }
  }
}
```

Increase in search latency in warm tier but almost negligible for Kibana searches



Effects of UltraWarm for Amazon ES

Before UltraWarm

- 57 cluster nodes
- Maximum 1-month search in Kibana

After UltraWarm

- 44 cluster nodes
  - Master: 3
  - Hot tier: 26
  - Warm tier: 15
- 3-month search in Kibana
  - Hot tier: 1 week
  - Warm tier: 3 months

An approximately 40% reduction in monthly cost

Future challenges

Kibana downtime during domain config changes / blue-green deployments

Resource spikes and frequent circuit breaks due to heavy queries

Optimizing app log document and mappings

Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.