A Best Practices Checklist for Using Amazon Elasticsearch Service

Jon Handler, Principal SA, AWS Search Services

July 28, 2020

A best practices check list

1. Designing
2. Architecting
3. Managing data
4. Deploying
5. Sizing
6. Testing
7. Stabilizing
8. Securing
9. Hardening
10. Hosting

Amazon Elasticsearch Service is a **fully managed service** that makes it easy to deploy, manage, and scale Elasticsearch and Kibana

Simple to use – it's a database

1

Send data as JSON via REST APIs

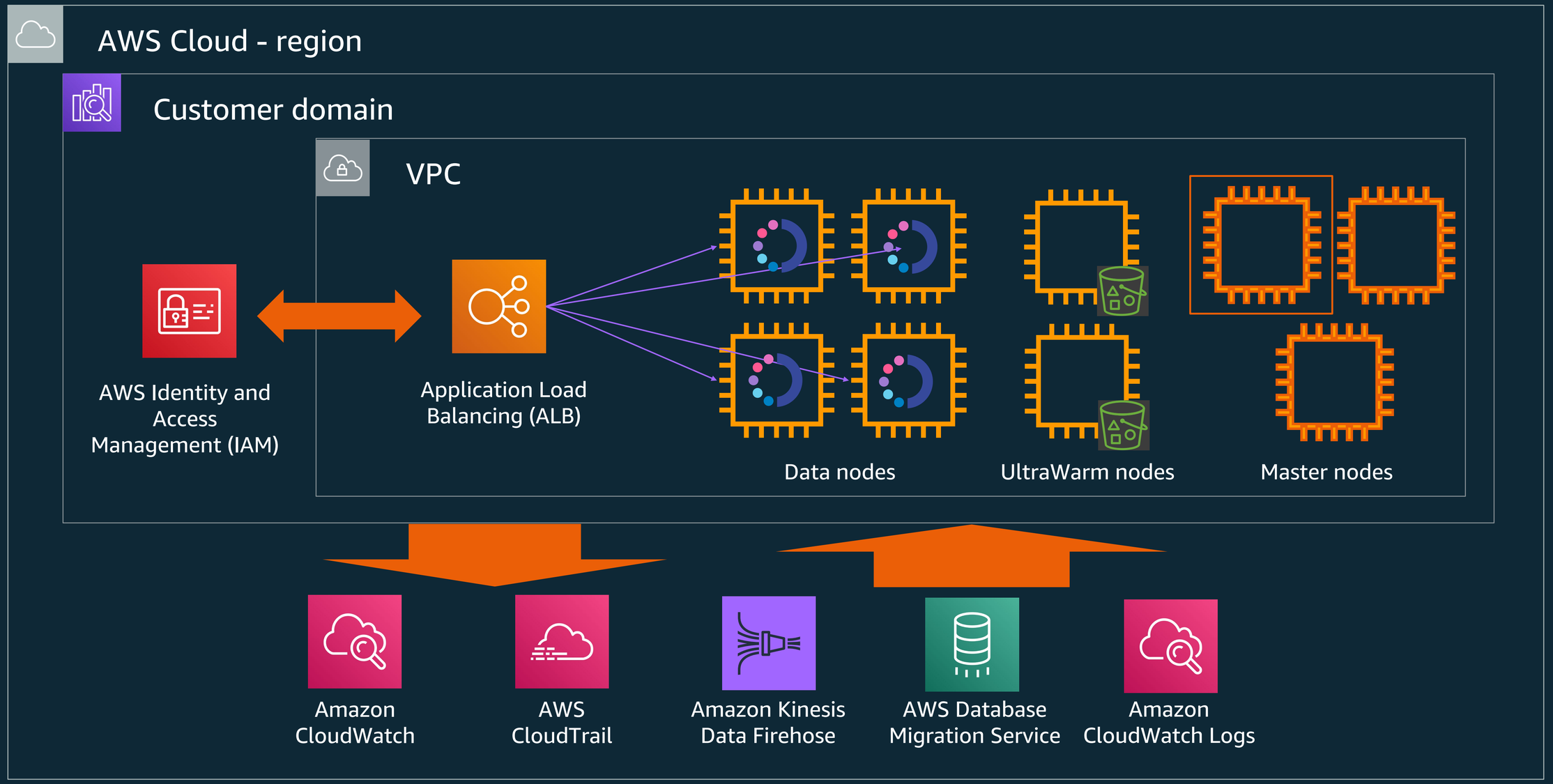
2

Data is indexed—all fields searchable, including nested JSON

3

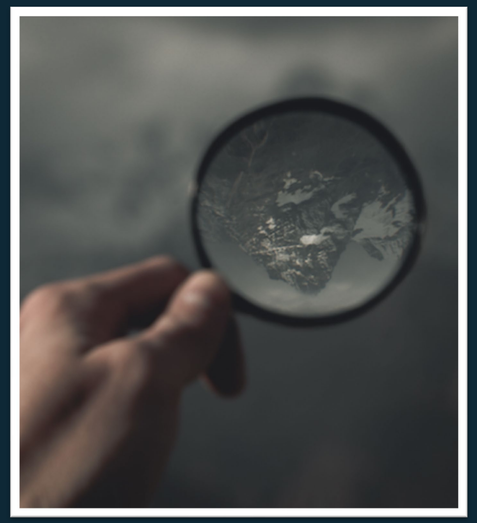
REST APIs, for fielded matching, Boolean expressions, sorting and analysis

Amazon ES architecture



1. Designing for Amazon ES

What's it good for?



Search workloads

Load your data in and search it. Rich queries, adjustable ranking, language features, search in a box.

Analytics workloads

Near real-time availability of log data (seconds)

Visualizations, dashboards, and alerting for monitoring

How to design for using Amazon ES

Search workloads

Design backwards from the document

Build a script to walk your database and bootstrap ES. Reuse to send updates

Maintain a copy offline for dev and test

Buffer high velocity updates

Analytics workloads

Buffer multiple sources to reduce concurrency

Transform with Lambda/Logstash

Use/tune the _bulk API! 3-5 MB batches

Use ingest processors with care

Amazon Elasticsearch Service is not a good durable store

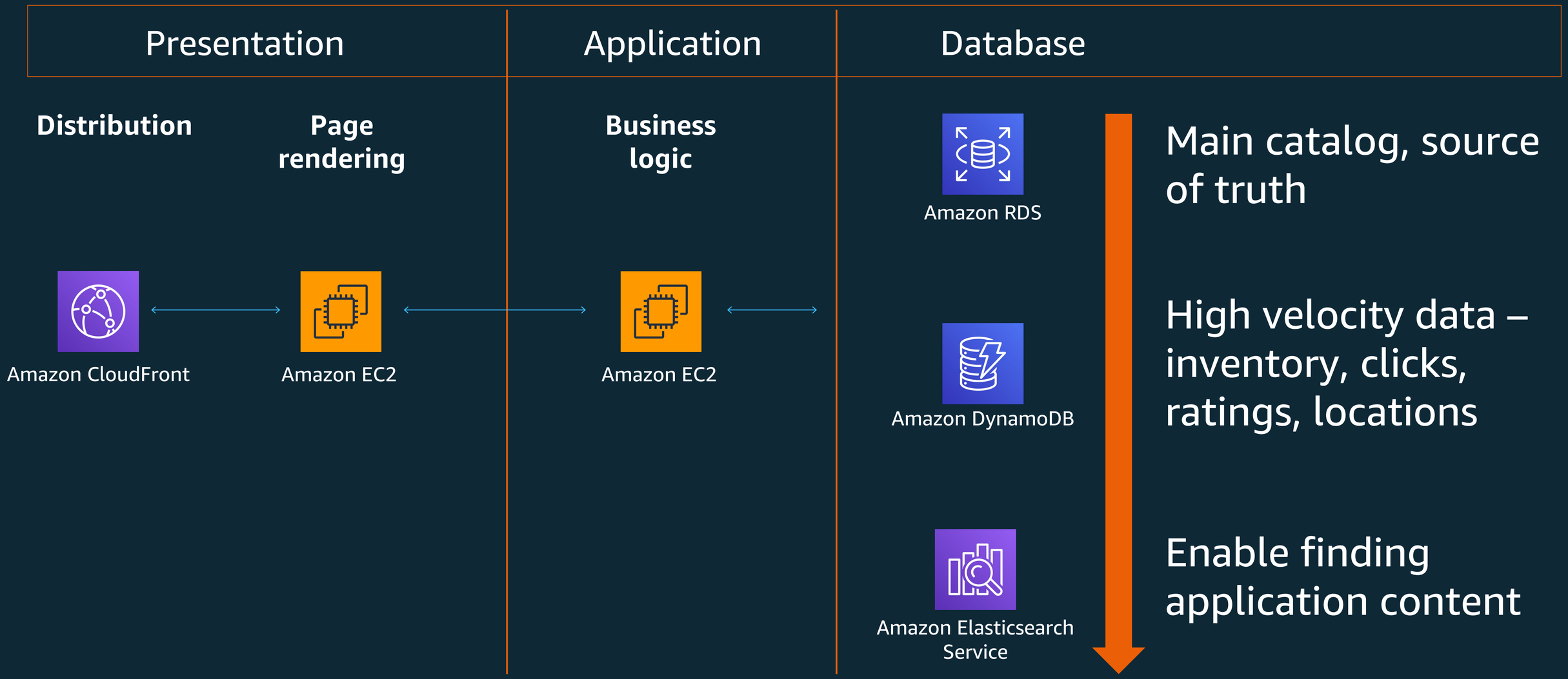
It's a so-so key-value store

Don't retrieve large result sets

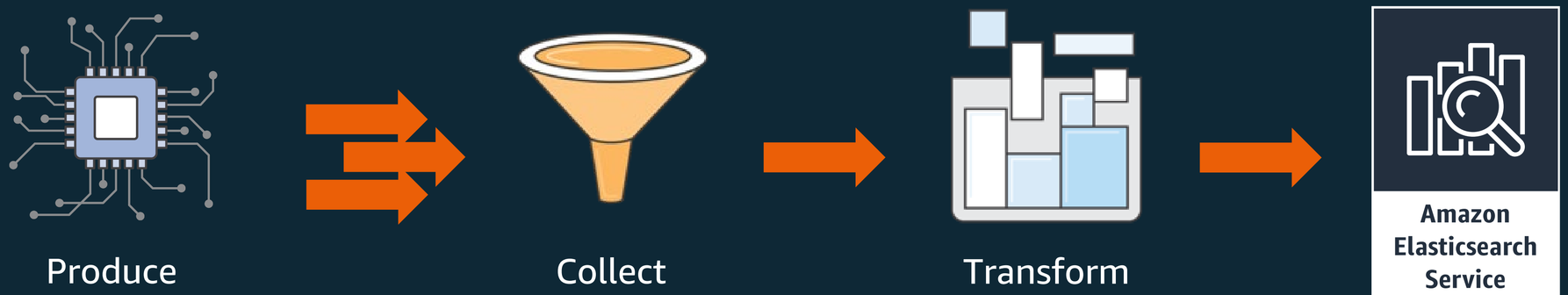
aws

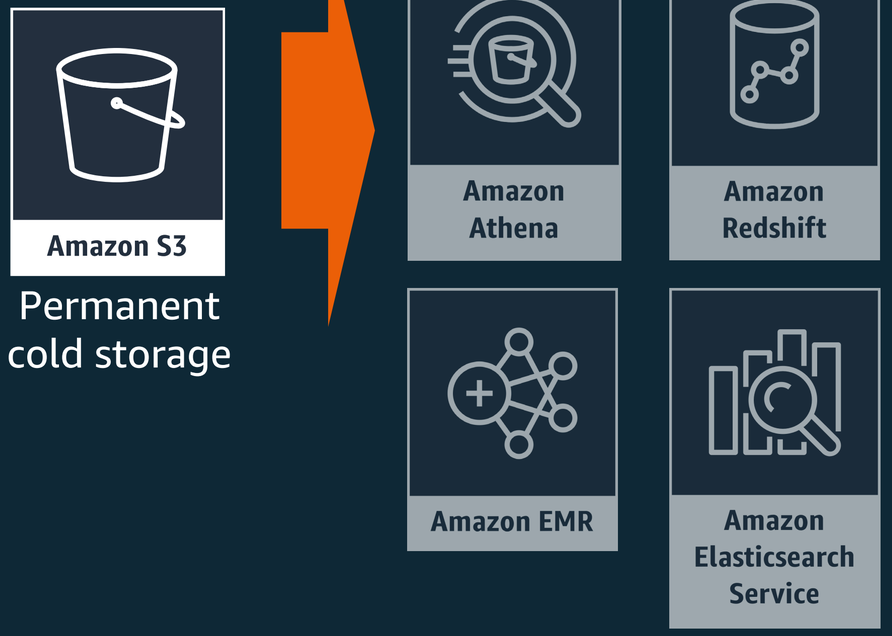
2. Architecting with Amazon ES

Search: Use search in tandem with a DB



Logs: Aggregate logs and keep a copy





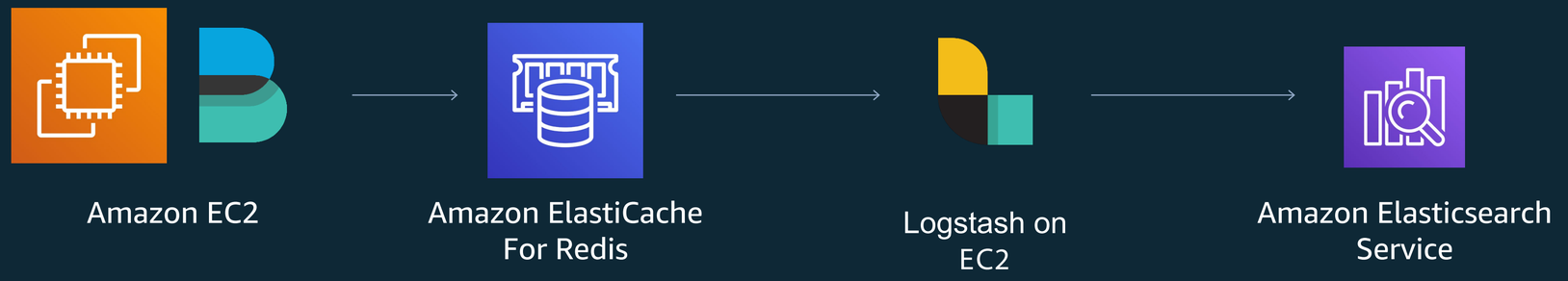
Option 1: S3/Lambda

Use S3 create events to trigger a Lambda function

The Lambda transforms and delivers the data

Use in conjunction with Athena

Option 2: Redis / Logstash

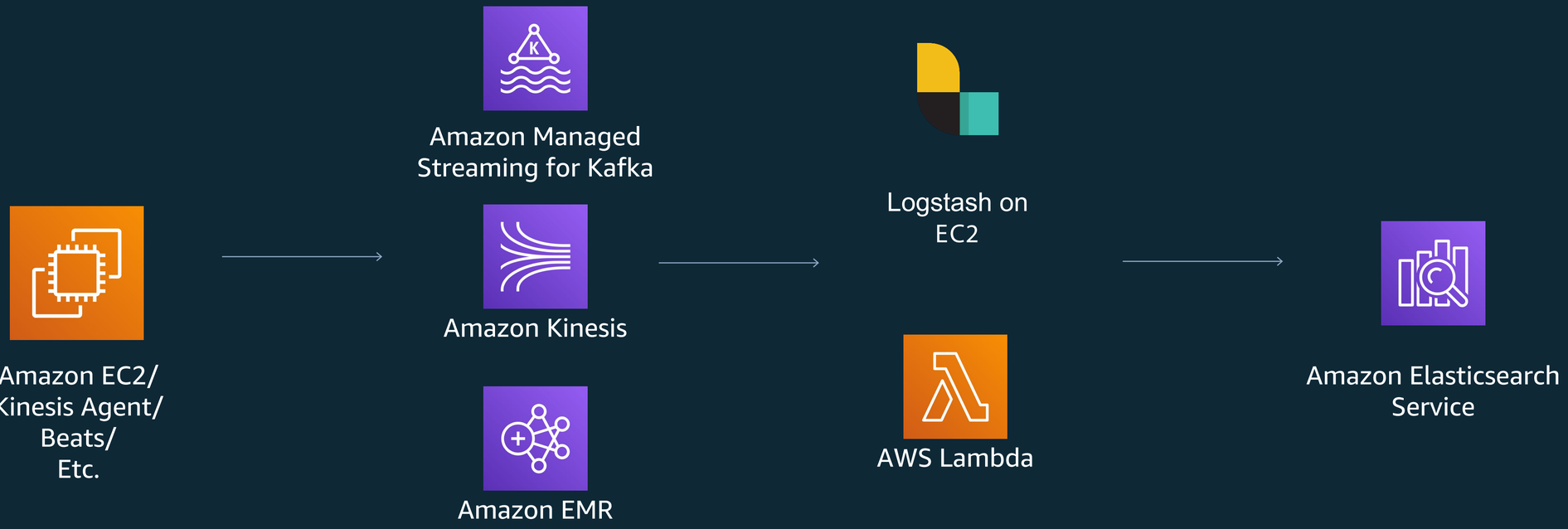


Beats: lightweight log shippers

Buffer with Redis

Transform with Logstash

Option 3: Managed Services

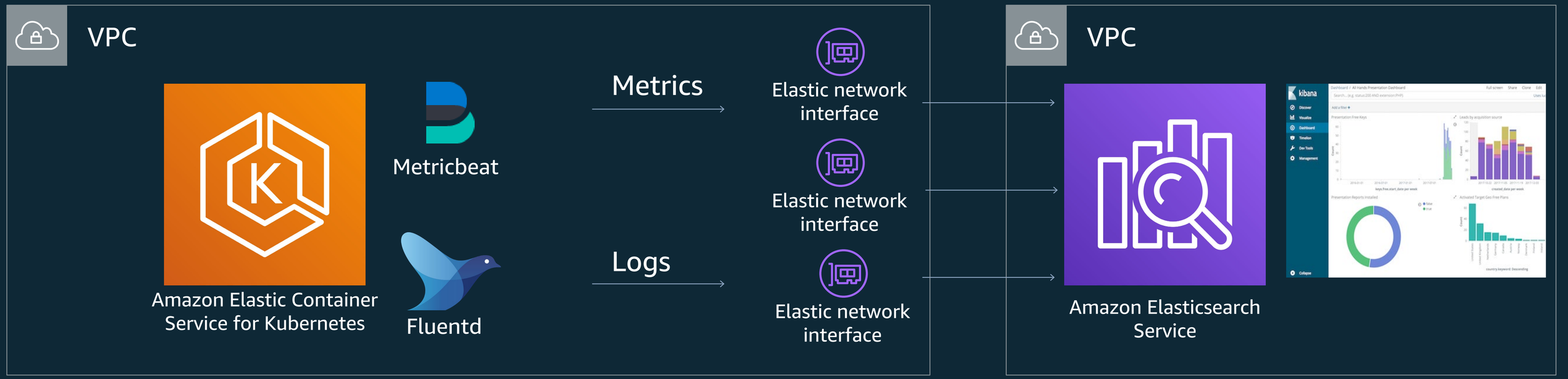


Employing streaming technologies at higher volumes

Some customers use CloudWatch Logs

Many customers use self-managed Kafka with Beats for shipping

Option 4: Fluentd/Fluentbit



Ingest node, pod, and container logs

Use Fluentd, Fluentbit

Use a buffer at scale

3. Managing Data in Amazon ES

Data is stored in indexes (cf. DB tables)

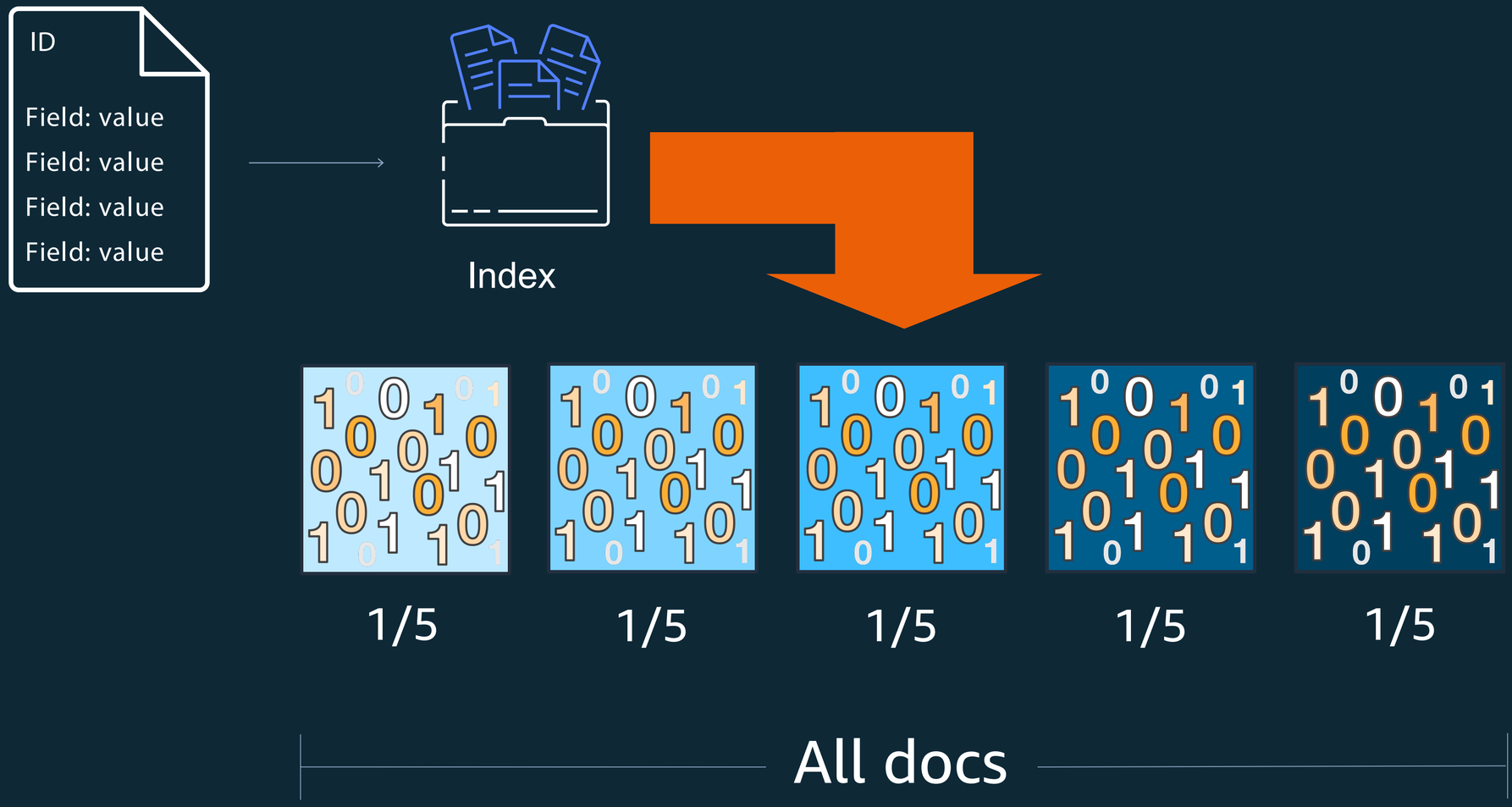
Streaming:
One index per day

- logs_11.26.2018
- logs_11.25.2018
- logs_11.24.2018
- logs_11.23.2018
- logs_11.22.2018
- logs_11.21.2018
- logs_11.20.2018

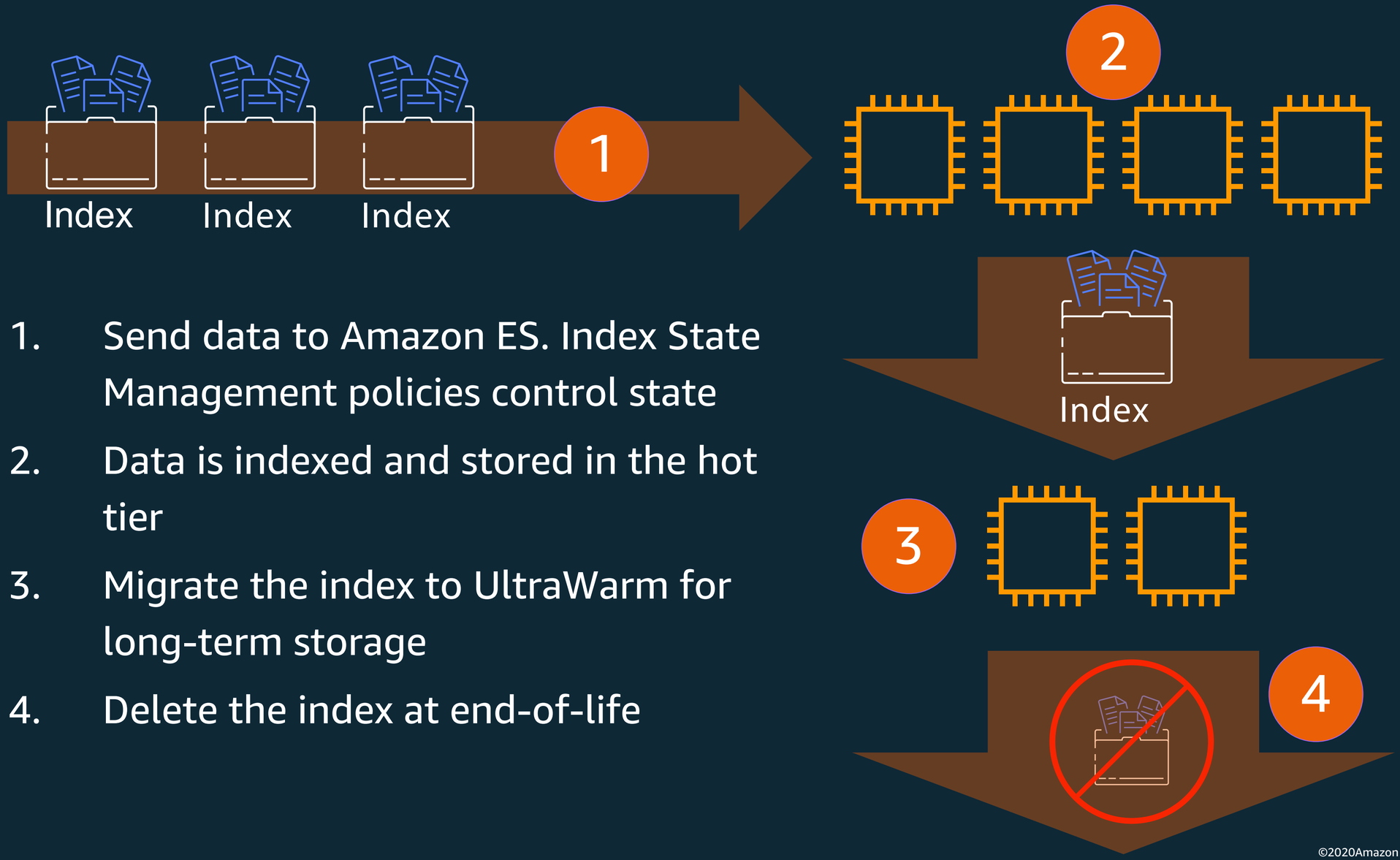
Search:
One index

- product_catalog

Indexes are comprised of shards



Data lifecycle in Amazon ES



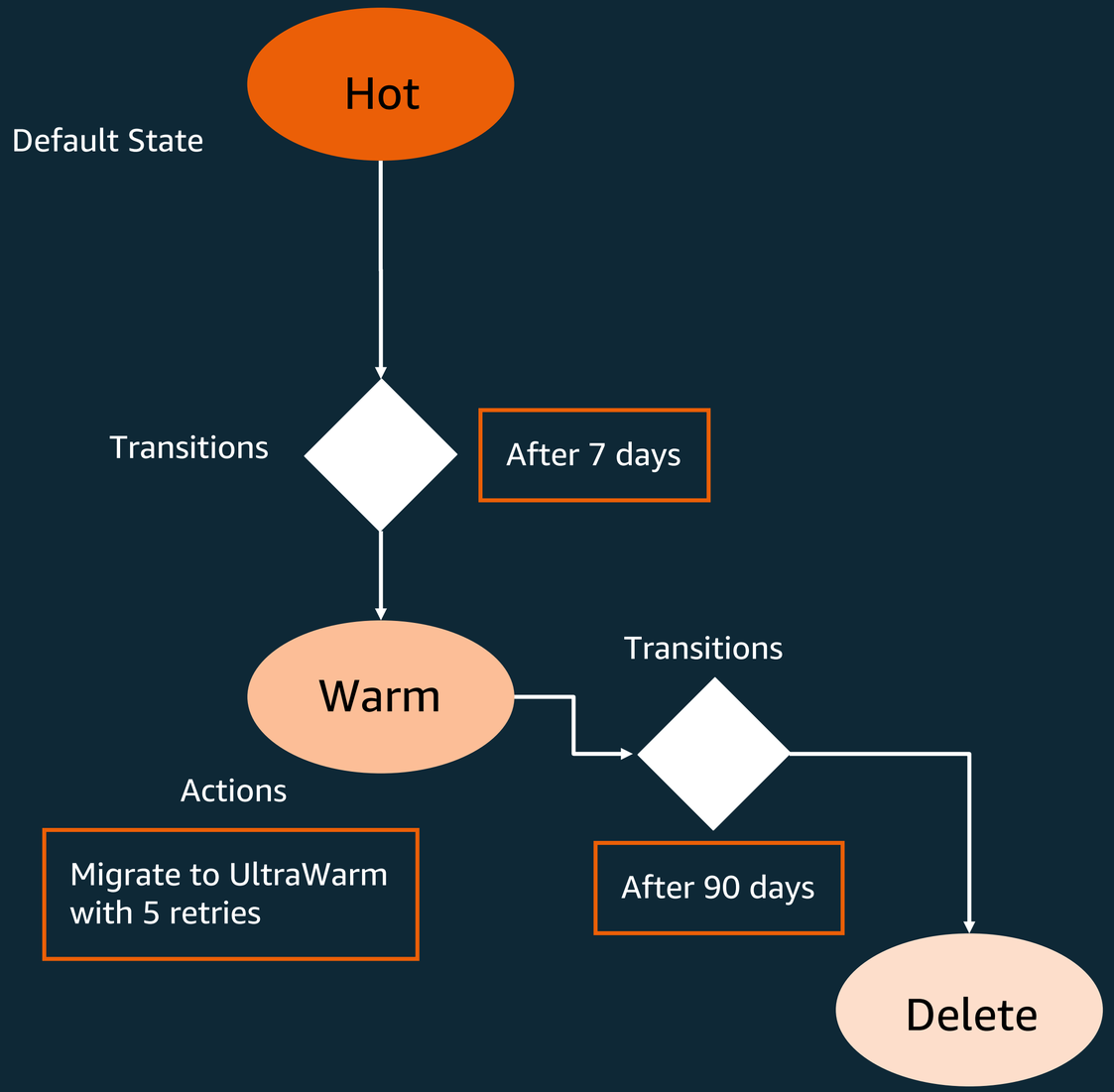
1. Send data to Amazon ES. Index State Management policies control state
2. Data is indexed and stored in the hot tier
3. Migrate the index to UltraWarm for long-term storage
4. Delete the index at end-of-life

Hot tier: indexing and fast access

UltraWarm: Low cost, long term retention

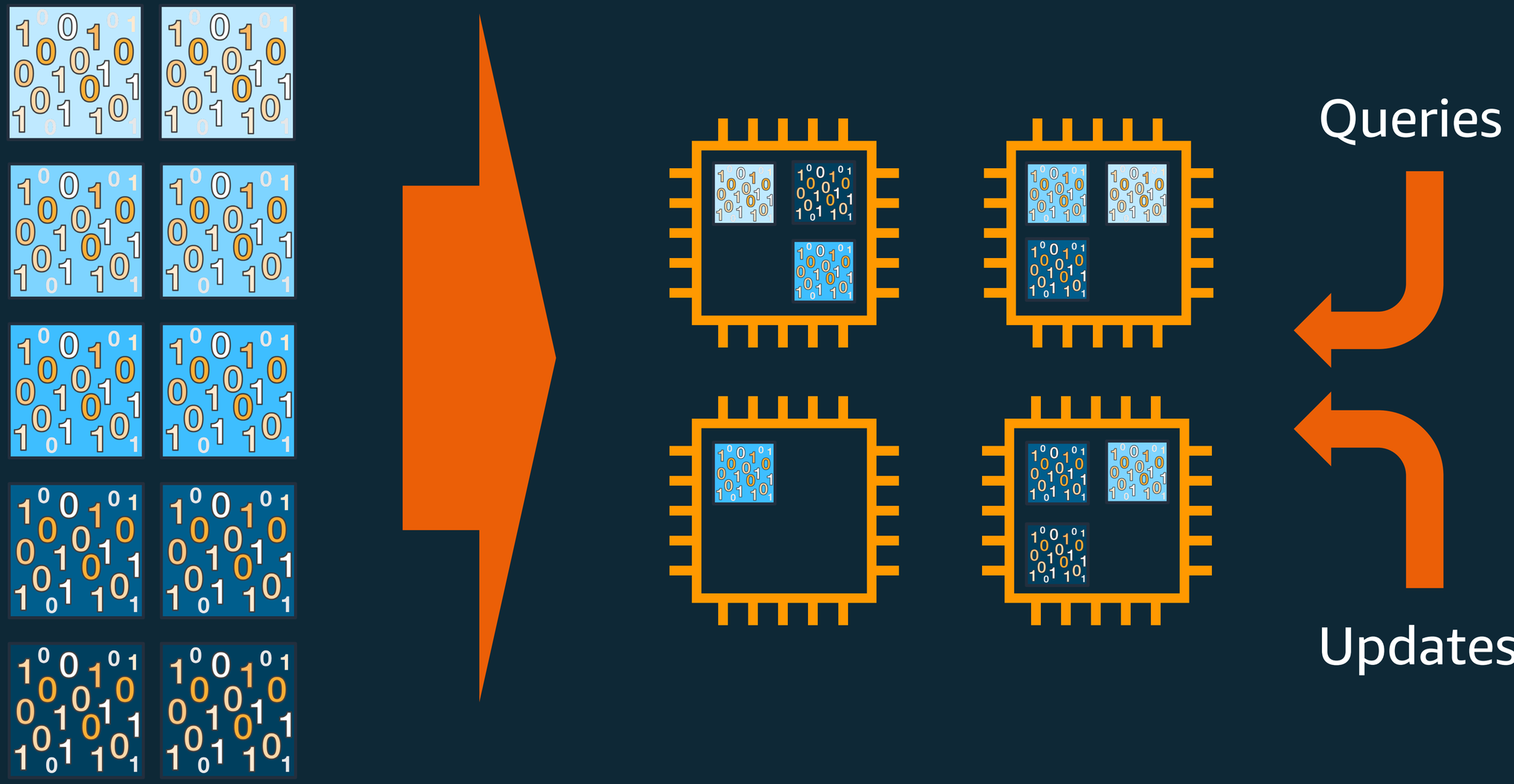
Use ISM

```
{ "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": {} } ] } ] } }
```



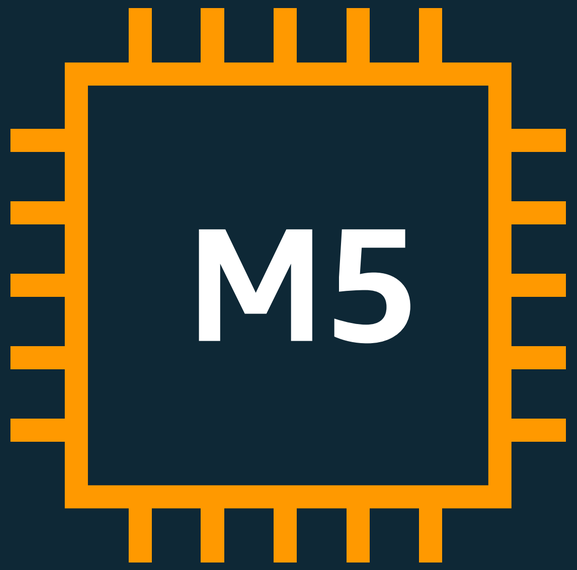
4. Deploying Amazon ES

Elasticsearch distributes shards to data nodes

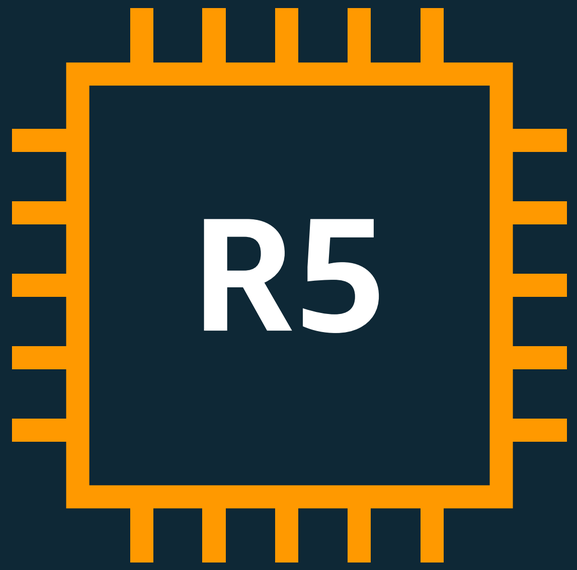


Use R5 or I3 nodes for large workloads

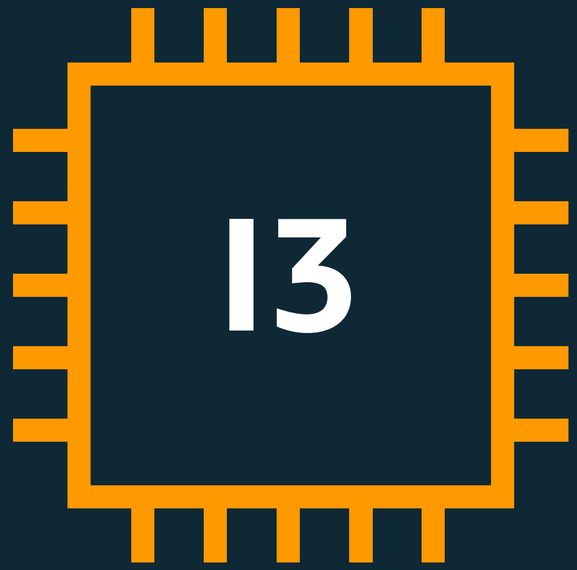
Data nodes are the workers in the cluster – they process requests and store indices



- Start here
- General purpose
- Cost effective



- Current gen CPU
- Can be faster than I3
- RAM speeds searches



- More search aligned
- I3.2xl is a good choice
- Avoid I3.16xl

Best instance unless testing shows otherwise

aws

Use dedicated master nodes

Master nodes orchestrate, are single threaded, scale based on configuration

Maximum number of data nodes	Maximum shard count	Master node instance type
10	2500	C5.large
30	5000	C5.xlarge
75	10,000	C5.2xlarge
75+	< 30,000	R5.4xlarge

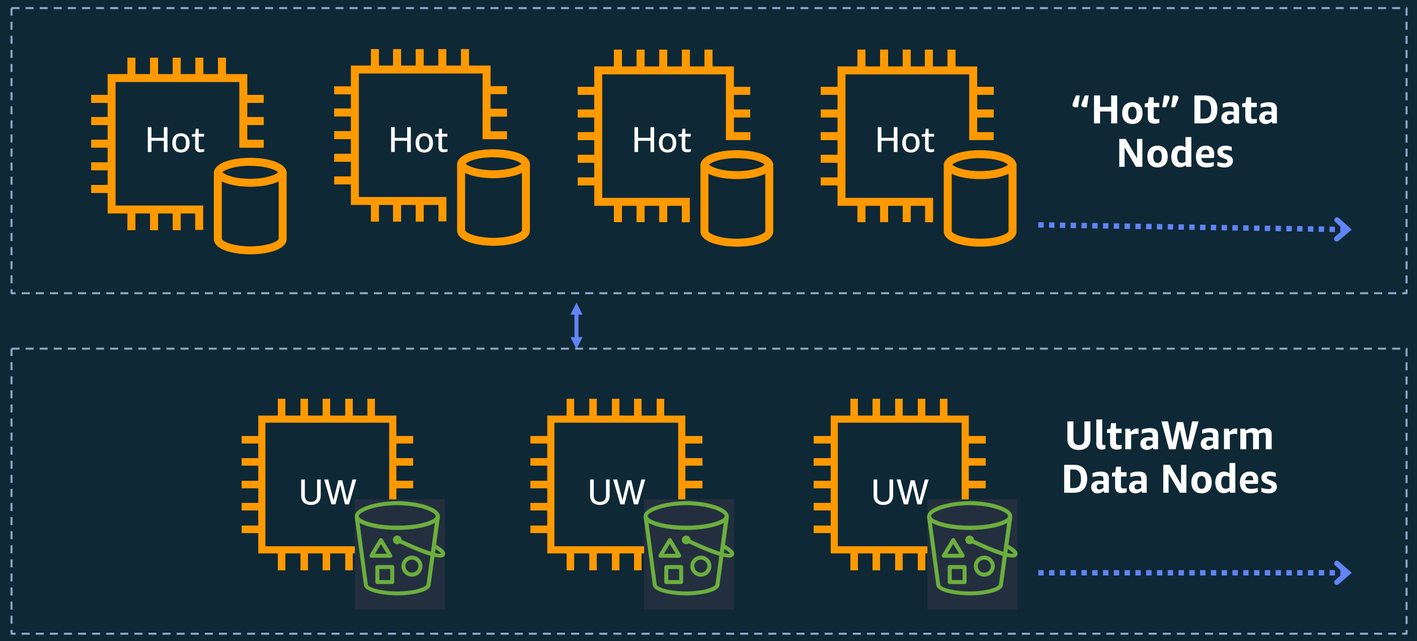
Use UltraWarm for long-tail data

Store massive amounts of log data

Run interactive log analytics and visualization

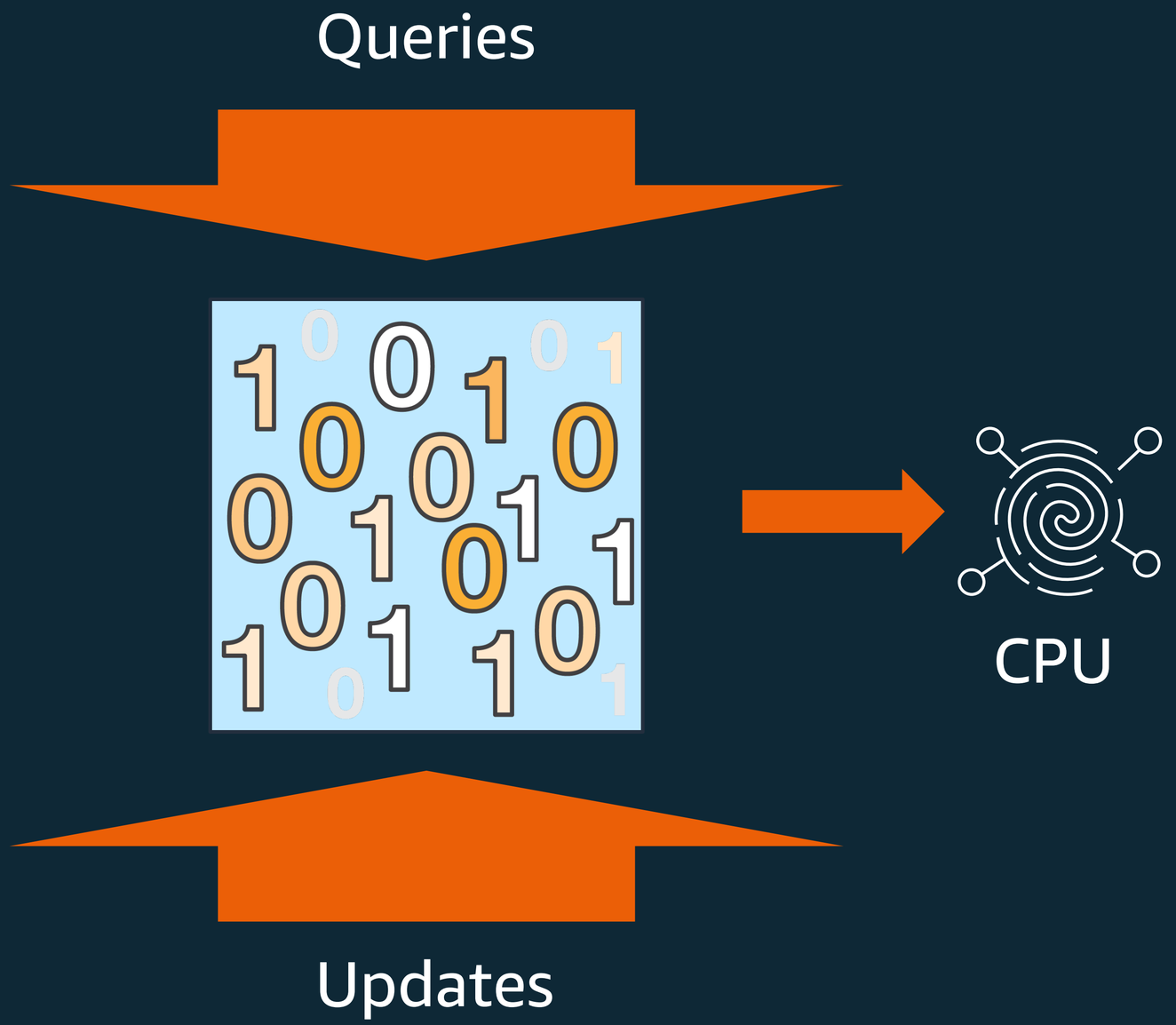
Higher performance and durability

Achieve up to 90% cost savings



5. Sizing Amazon ES

Shards are the workers



Your configuration determines capacity

Instance Type

Deploy instances based on storage and compute needs

Instance Count

Add instances for increased parallelism

Storage

Index data (primary and replica shards) is stored on disk

Shard Count

Shards are the units of work and storage

Logs workloads are storage driven. Search workloads are CPU/JVM driven

Step 1: figure out storage need

Storage needed = Source/day * 1.1 * 2 * retention * 1.15

Search: 100 GB of data needs 250 GB of storage

Logs: 1TB daily of source data needs 18 TB of storage for 7 days retention

Step 2: figure out shard count

Primary Shards =

Index size / target shard size

Logs, use 50 GB max. Search evaluate 20-30 GB

Step 3: Set a template

For log analytics, set a template

```
*PUT
<endpoint>/_template/template1
{
    "index_patterns": [”logs*"],
    "settings": {
        "number_of_shards": 50,
        "number_of_replicas": 1
    }
}
```

Step 4: Adjust for usage

vCPU = 1.5 * active shards

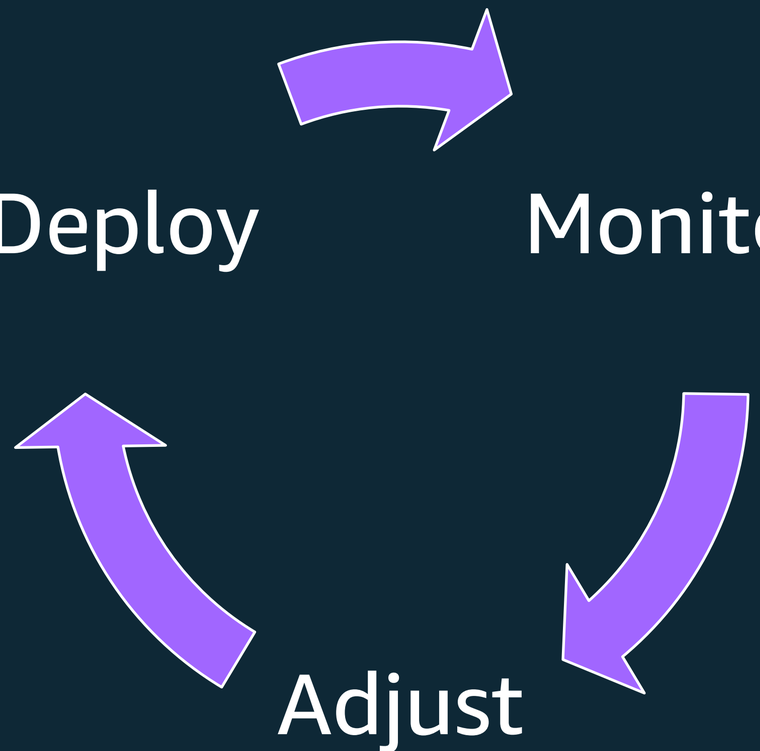
Active shards

- Primaries for queries
- Primaries and replicas for updates

E.g. 4 data streams @ 1 TB daily means 40 total shards (20 primary and 20 replica) active, so make sure to have 64 total CPUs

6. Testing Amazon ES

The number one best practice: Test!



https://aws.amazon.com/blogs/big-data/best-practices-for-configuring-your-amazon-elasticsearch-service-domain/

Plan for and execute load testing

Amazon ES performance and resource usage are highly workload dependent

Predicting performance is not an exact science

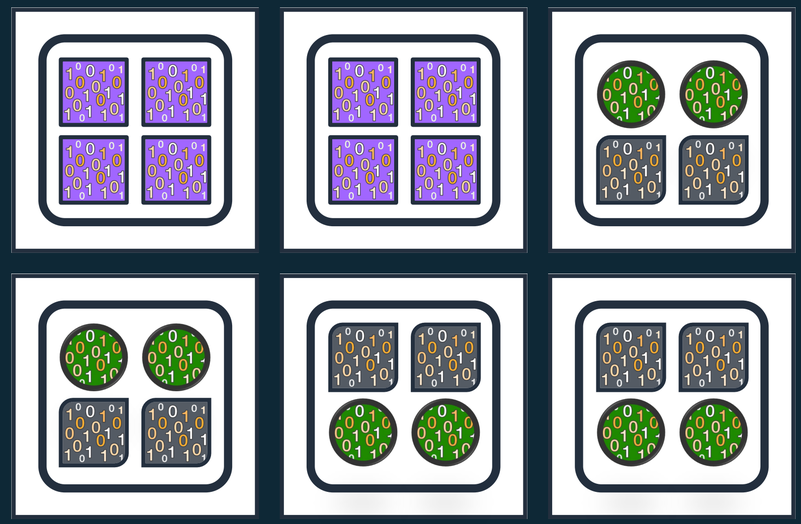
Testing

- Use real data and queries if at all possible. At least thousands of distinct queries/updates for the workload. Mind the zero-result mix
- Identify maximum single shard, error-free throughput (shard size)
- Identify maximum multi-shard, error-free throughput (shards per node)
- Scale shard and node count appropriately
- Deploy your fill workload, test and monitor – CPU, JVM, queues

7. Stabilizing Amazon ES

Avoid skew in storage and processing

Unhealthy



Healthy

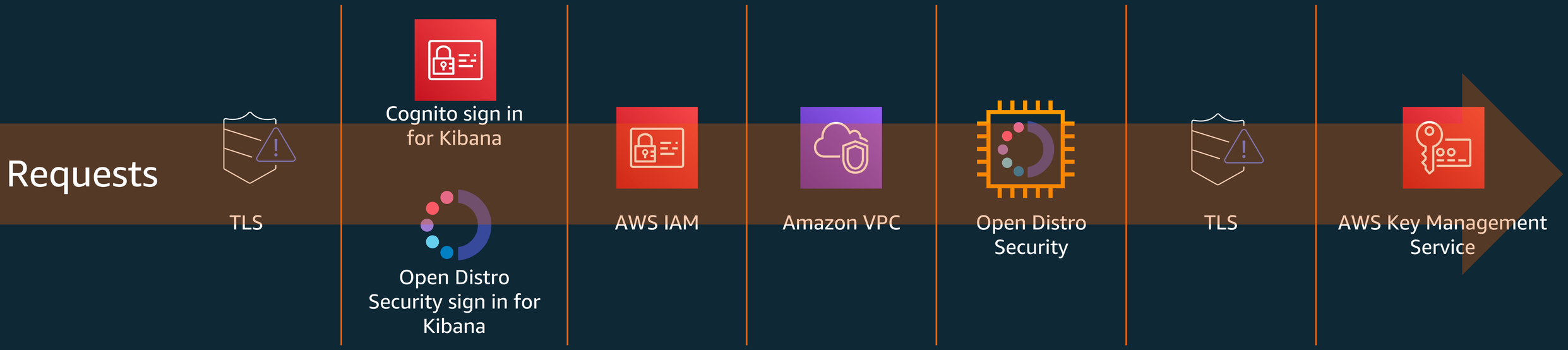
Use the _rollover API to equalize different index sizes

Vary retention windows

Split different workloads that have different resource usage

8. Securing Amazon ES

Multi layer security



Encrypted from end to end – in flight with TLS, at rest with KMS

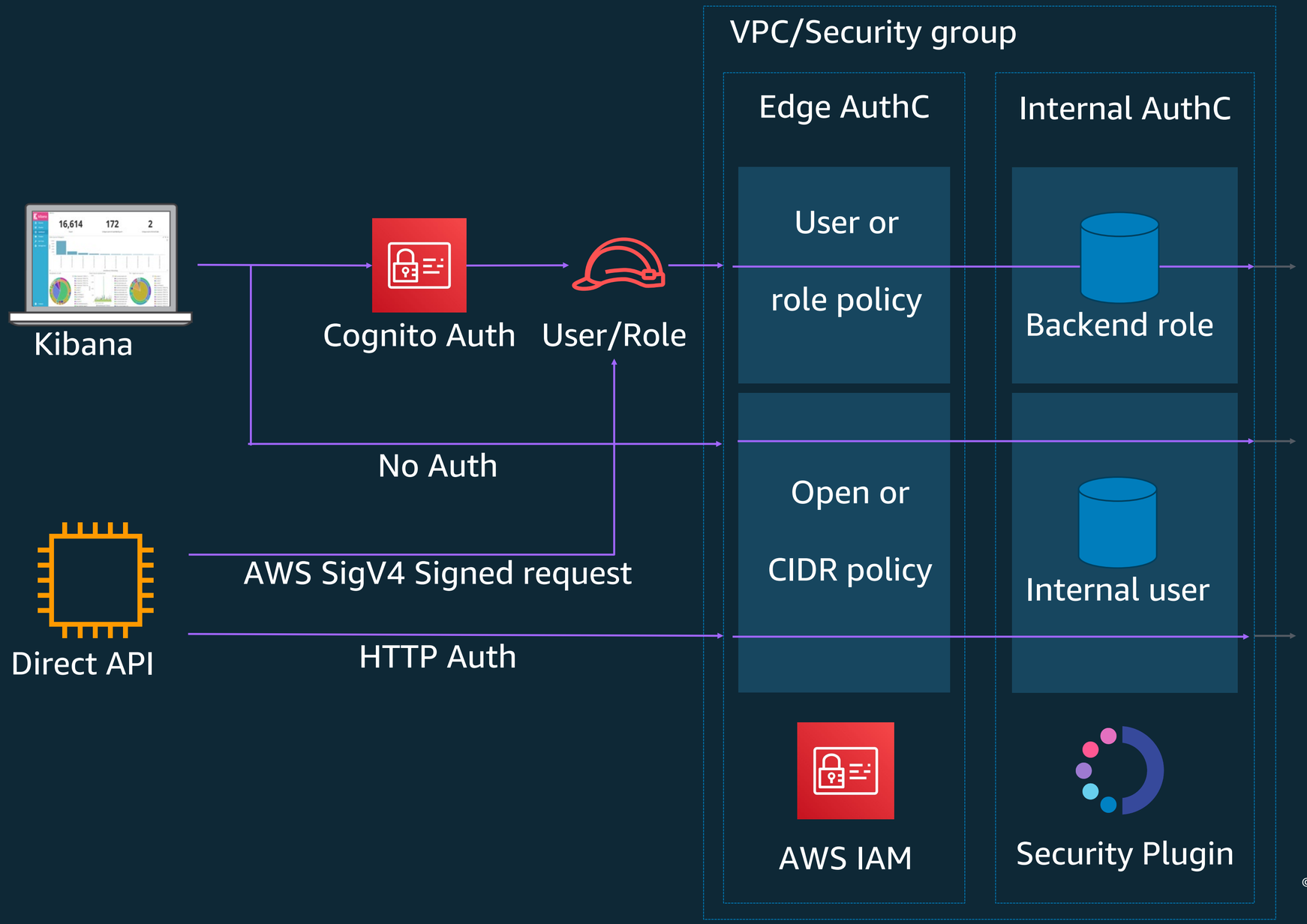
Use a private endpoint to deploy into your VPC and security groups for traffic control

Includes Kibana login via Cognito integration, or native with Open Distro Security

Coarse-grained access control with IAM policies

Fine-grained access control with Open Distro Security

Secure your cluster



Authenticated and access granted

Use a VPC endpoint

Your VPC

Availability Zone 1

Security group

ENIs

Availability Zone 2

Security group

ENIs

Authc/Authz

AWS IAM

Security Plugin

Service-managed VPC

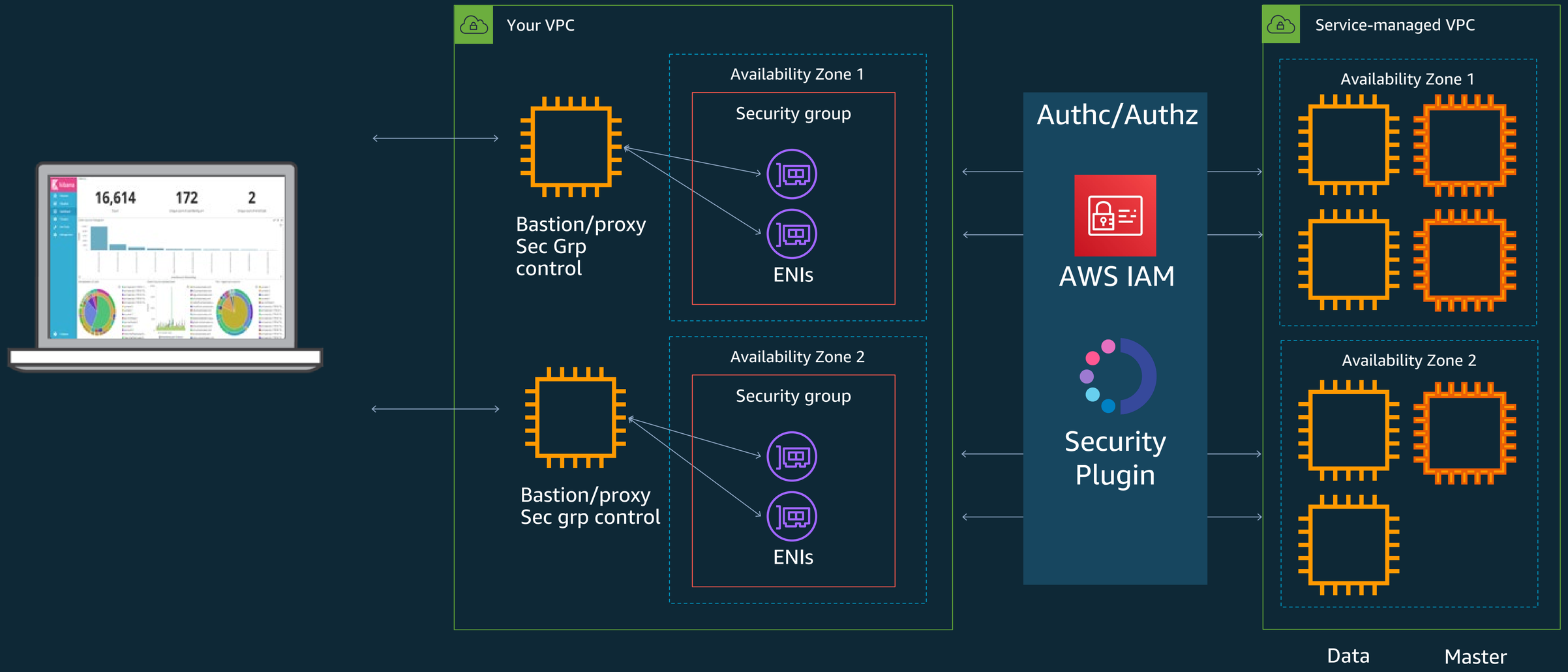
Availability Zone 1

Availability Zone 2

Data

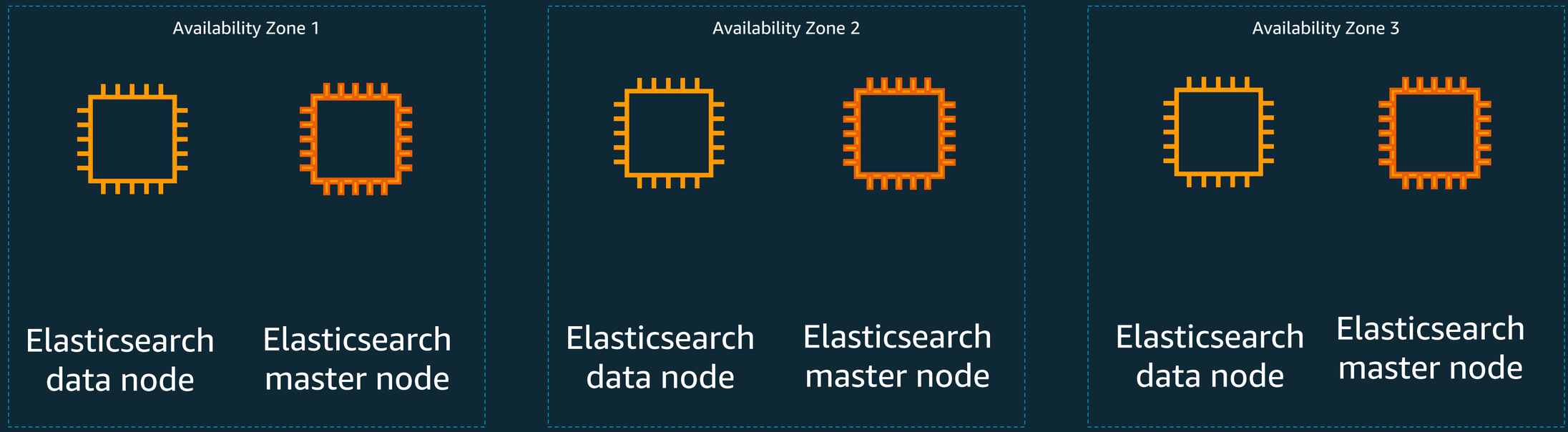
Master

Get traffic to the domain



9. Hardening Amazon ES

10. Use 3 zones, 2 replicas

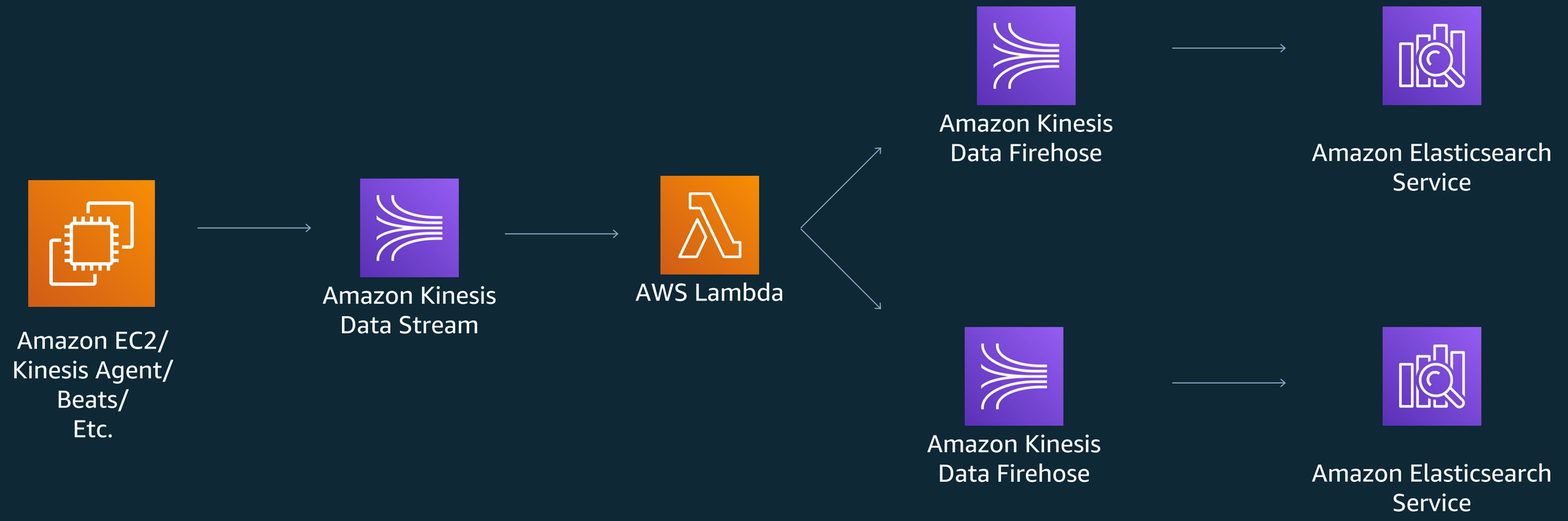


Nodes are divided across zones

Data is replicated across zones (requires proper Elasticsearch replication)

In some cases 1 replica is sufficient

10. (cont.) Use dual writes for DR



Higher cost, more stability – appropriate for critical workloads

Use as a hot or warm standby

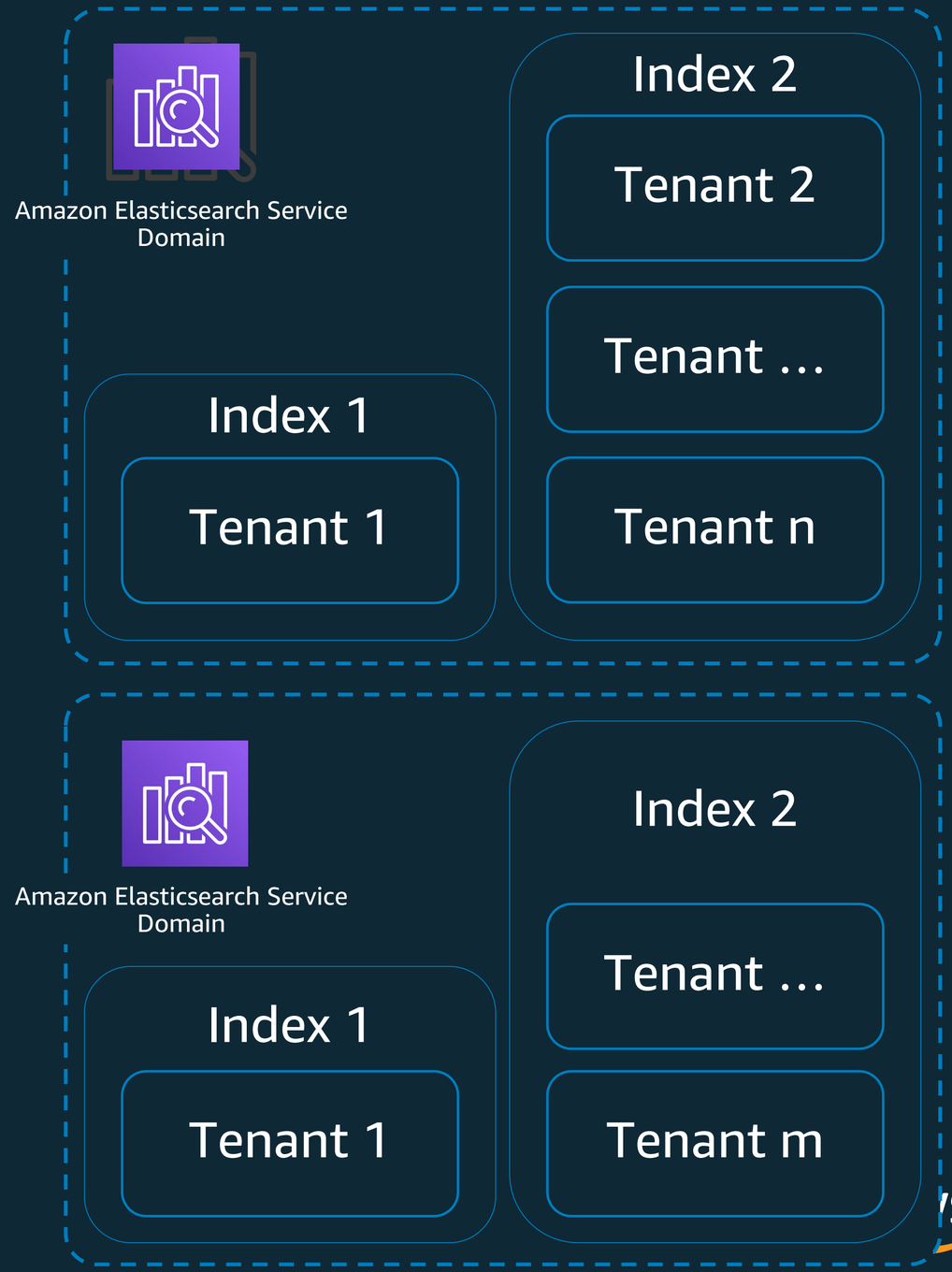
Recommended alarms

Name	Metric	Threshold	Periods
ClusterStatus.red	Maximum	>= 1	1
ClusterIndexWritesBlocked	Maximum	>= 1	1
CPUUtilization/MasterCPUUtilization	Average	>= 80%	3
JVMMemoryPressure/Master…	Maximum	>= 80%	3
FreeStorageSpace	Minimum	<= (25% of avail space)	1
AutomatedSnapshotFailure	Maximum	>= 1	1
Queue depth	Average	>= 100	1
Rejected Executions	Maximum	>= 1	1

10. Hosting multiple workloads on Amazon ES

Handling tenancy

- Employ single indexes for the first (and maybe second) standard deviation users
- Split higher-usage tenants into their own index
- Use separate domains for extra-large tenants



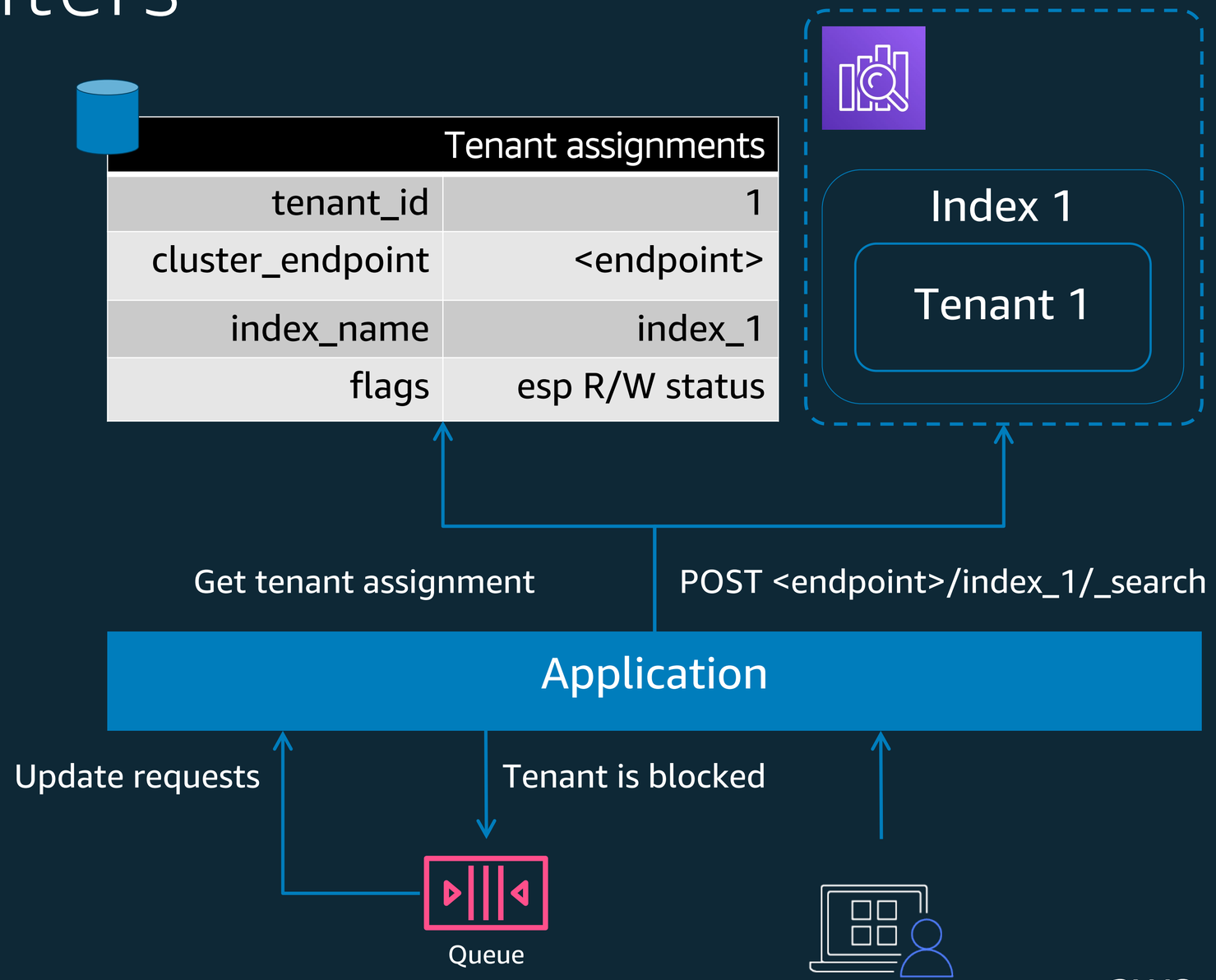
* Source NBC SportsEngine: http://tinyurl.com/y2v6bdf3

Use external pointers

Use a database to hold a pointer for each tenant

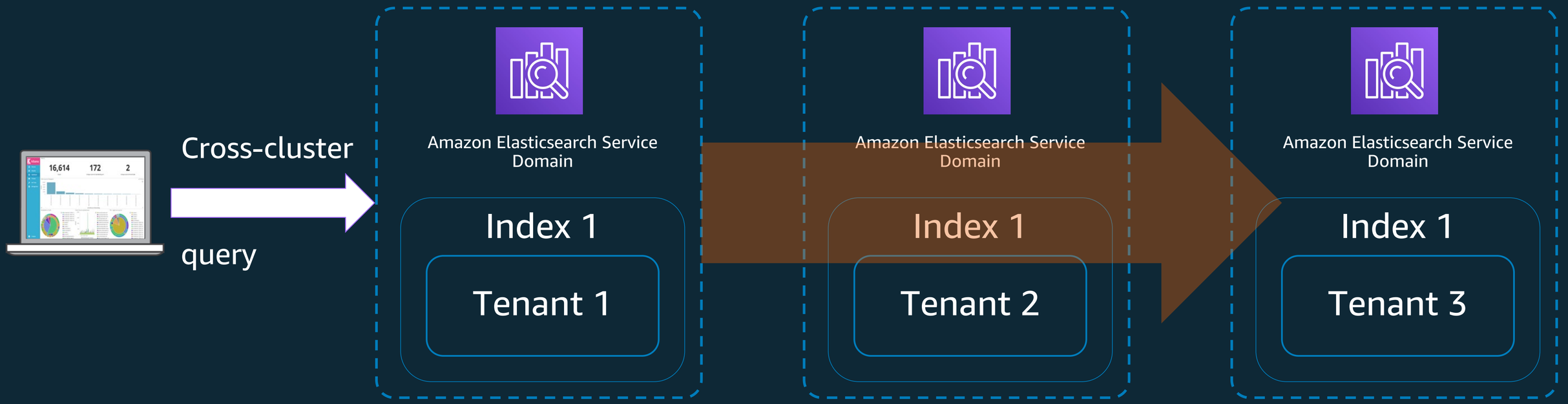
Add bits for status

Move tenants by locking, re-indexing, deleting, and changing pointers



*Source NBC SportsEngine: http://tinyurl.com/y2v6bdf3

Use cross-cluster search



Minimize blast radius

Scale independently

A best practices check list

1. Designing
2. Architecting
3. Managing data
4. Deploying
5. Sizing
6. Testing
7. Stabilizing
8. Securing
9. Hardening
10. Hosting

Thank you!

Jon Handler

Principal SA, Search Services

handler@amazon.com

AWS