2018
CONTAINER DAY

基于Rancher的ELKstask as a Service实践

姜江 | 新东方信管部高级系统工程师

RANCHER

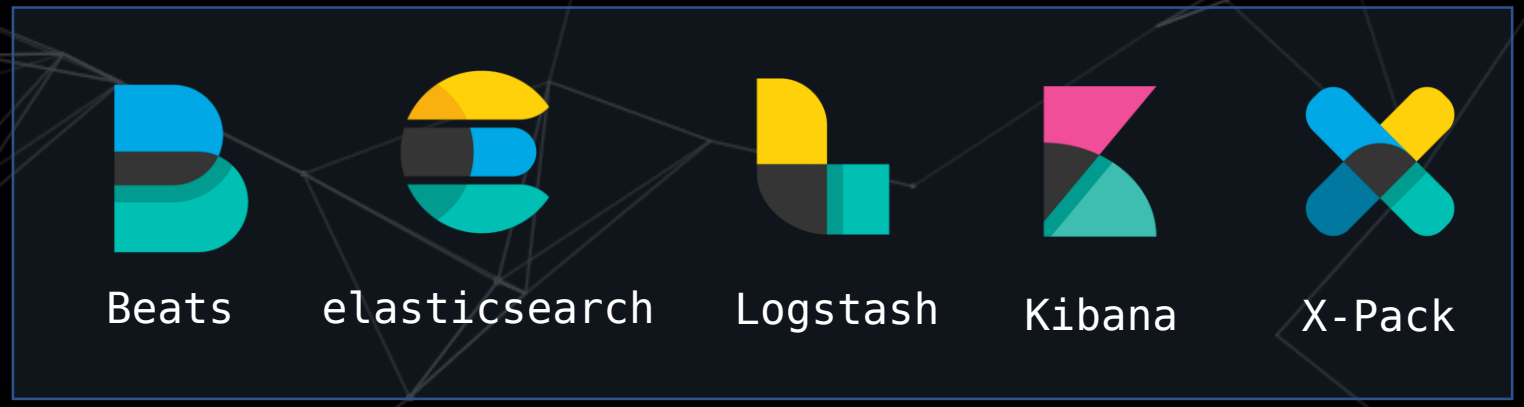
RANCHER | IT大咖说 知识共享平台

2018
CONTAINER DAY

1. ELK Stack的生态圈

RANCHER

ELK生态圈：BELKX + kafka + fluentd



数据收集

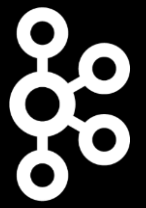
索引和存储

数据聚集和处理

分析和可视化

监控权限高级功能

数据收集、聚集和处理



高吞吐流处理平台

RANCHER
2018
CONTAINER
DAY

核心组件elasticsearch

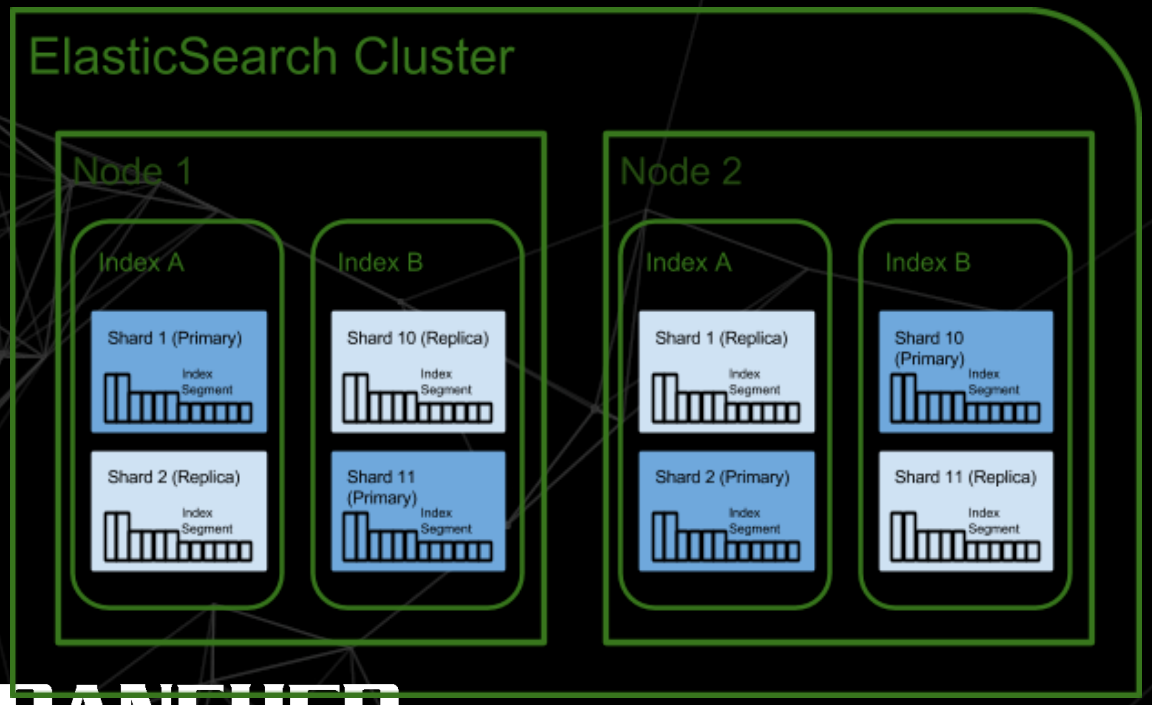
- Elastic Stack 的核心
- Elasticsearch 是一个分布式的 RESTful 风格的搜索和数据分析引擎
- 据说是为了解决搜索菜谱的需求而诞生

elasticsearch

RANCHER
2018
CONTAINER
DAY

https://github.com/exo-archives/exo-es-search

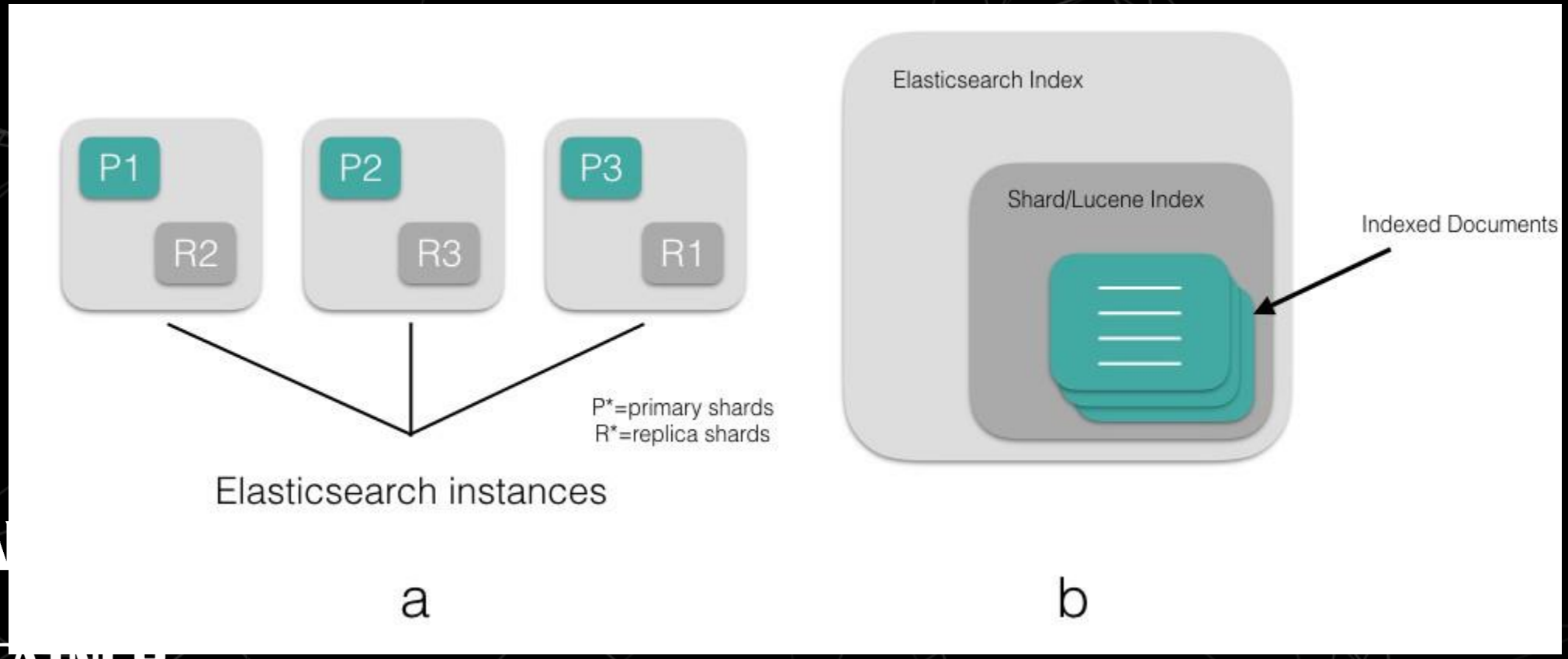
ElasticSearch基本要素



- Cluster（集群）
- Node（节点）
- Shard（分片）
- Replicas（分片副本）
- Index（索引）
- Type（类型）
- Documents（文档）

RANCHER 2018 CONTAINER DAY

Cluster、Node、Shard和Index



Picture：Ronak NathaniSRE @ LinkedIn

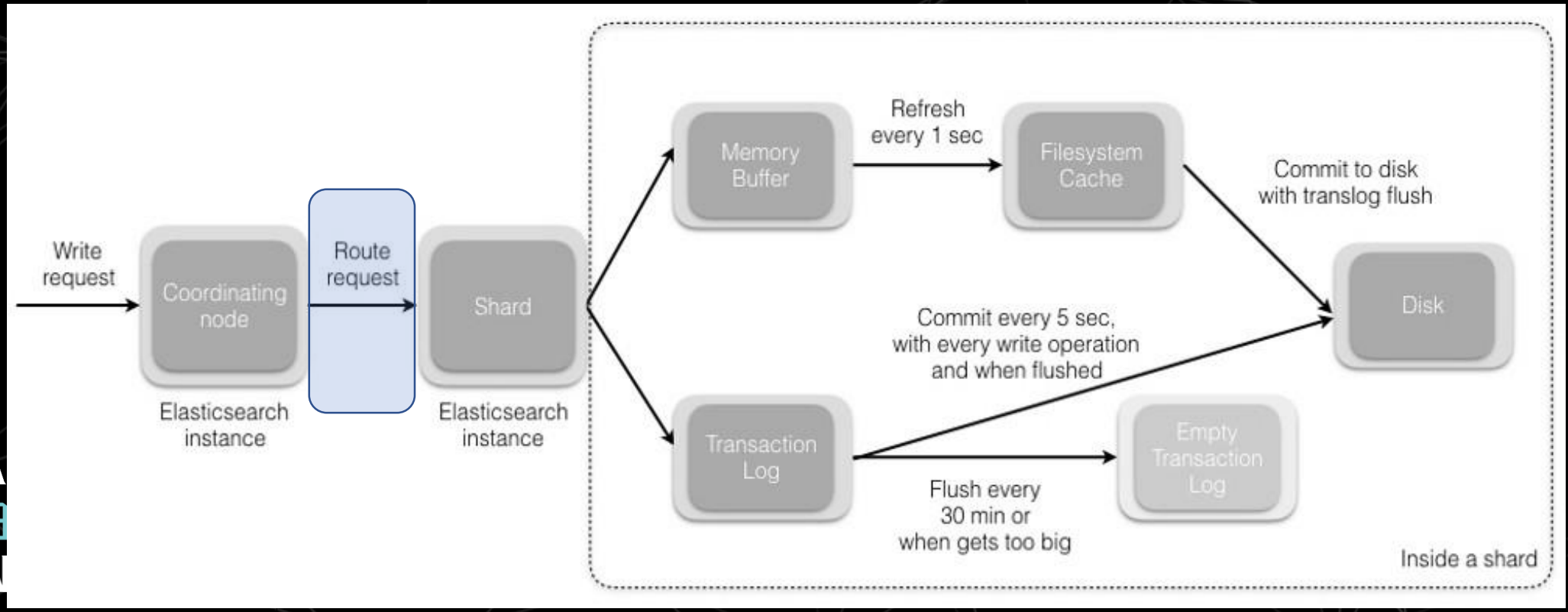
分片、副本与集群健康程度

GET /_cluster/health

- 所有主要分片和复制分片都可用
- 所有主要分片可用，但不是所有复制分片都可用
- 不是所有的主要分片都可用

文档写入ES的过程

shard = hash(document_id) % (num_of_primary_shards)



Pictrue:Ronak Nathani SRE @ LinkedIn

简单总结

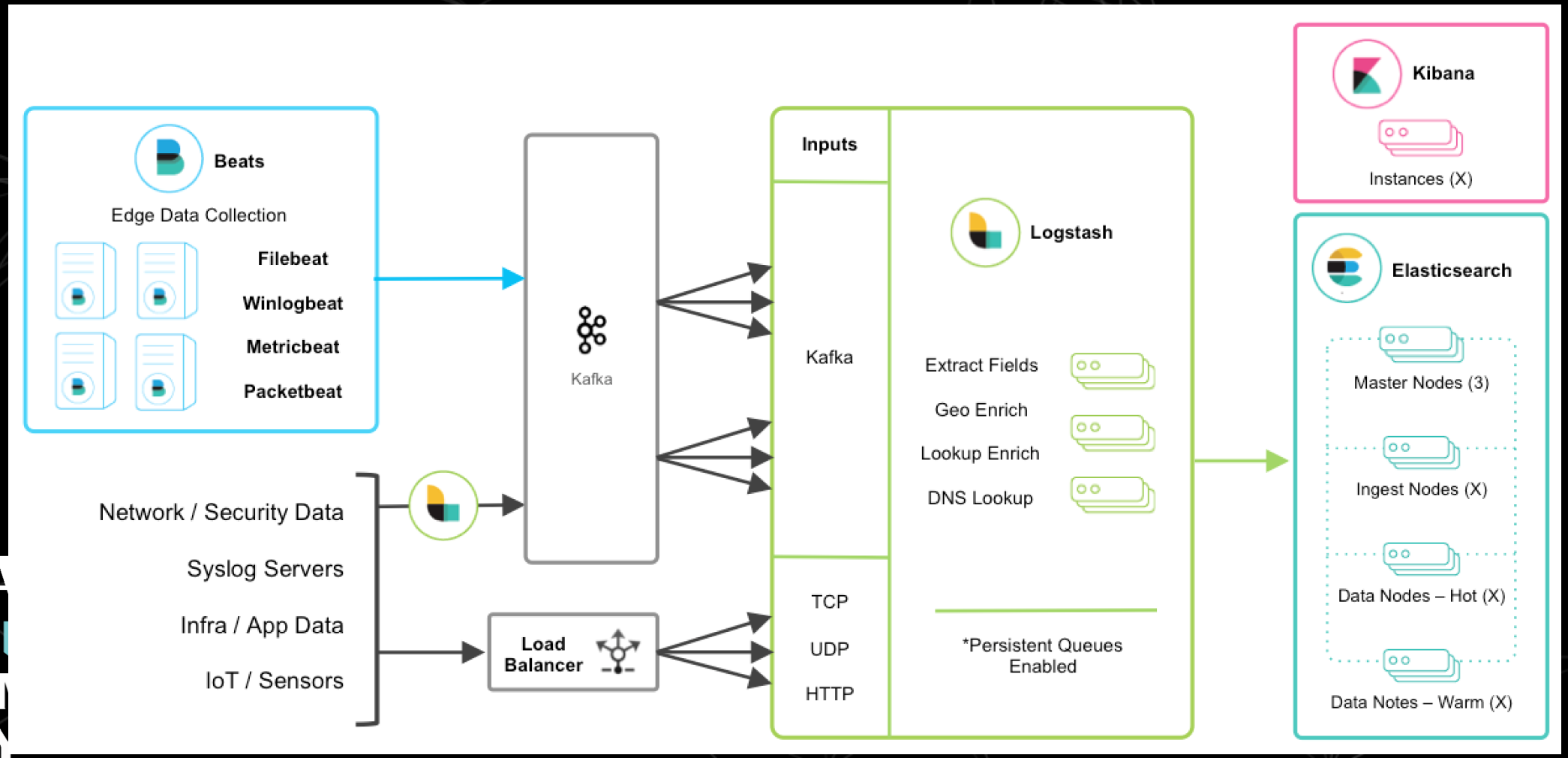
ES特性

- 根据文档ID和分片数量取余得到文档的目标分片位置
- ES集群根据分片位置将文档路由到对应的node上处理
- 收到文档的shard保存日志，并交由lucene处理，处理后刷入OS的Buffer中，最后按条件刷入磁盘形成segment

优化建议

- 增减集群中的data node 会引起数据的再分配，可以通过参数控制一次再分配的索引数量
- 增加coordinating node 作为数据读取的负载均衡，他会直接将请求路由掉对应的data node
- ES的内存分配遵循50%原则

ELK生态圈： 常见架构



Picture: elastic.co

2018
CONTAINER DAY

2.容器化ELK云服务的优势

RANCHER

传统方式搭建ELK

- 按项目申请环境和资源
- 买来五台，十台的高配服务器
- 每台机器做为一个node，配置相同的cpu、内存、SSD
- 只创建一个集群应付所有的业务场景
- 按照项目将所有数据都塞进去
- 撑不住了就按照上面步骤再搭一套集群，周而复始

RANCHER
2018
CONTAINER
DAY

ELK云服务的优势

RANCHER
2018
CONTAINER
DAY

- 资源集中分配，避免资源竖井
- 基础架构代码化，标准化
- 按照应用场景，数据可靠性、服务SLA区分ES集群
- 相同类型的数据放在一个集群内，而不是按照项目分配集群
- 按照组件的不同负载特性组合资源
- 自动化部署/回收实例降低运维压力

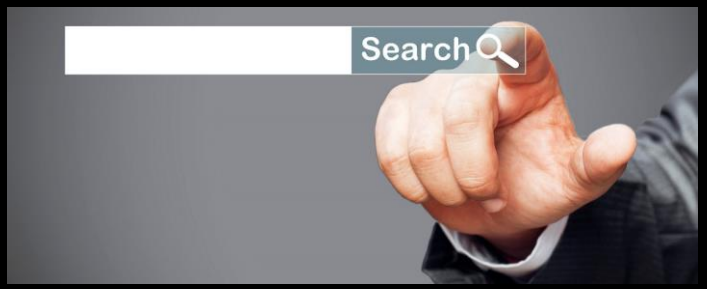
按场景分割ES集群实例

日志分析

量大，存储时间长，数据完整性要求适中

流量分析

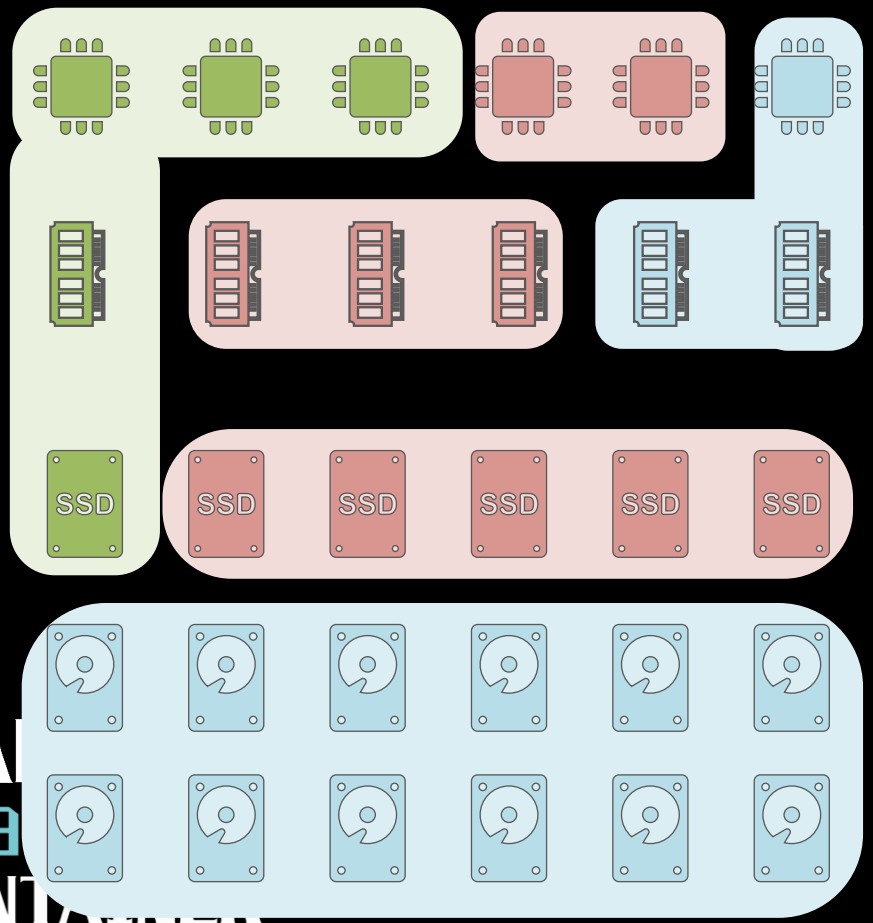
数据量巨大，存储时间短，数据完整性要求高



搜索查询加速

数据适中，存储时间长，数据完整性要求高

按特性整合计算资源



- 较高配置的服务器替代低配服务器，提高服务器密度。
- 将不同特性的计算负载交叉整合到一台物理服务器上，如：
  - 计算密集负载
  - 内存密集负载
  - I/O密集负载
  - 存储量密集负载
- 避免资源的闲置。

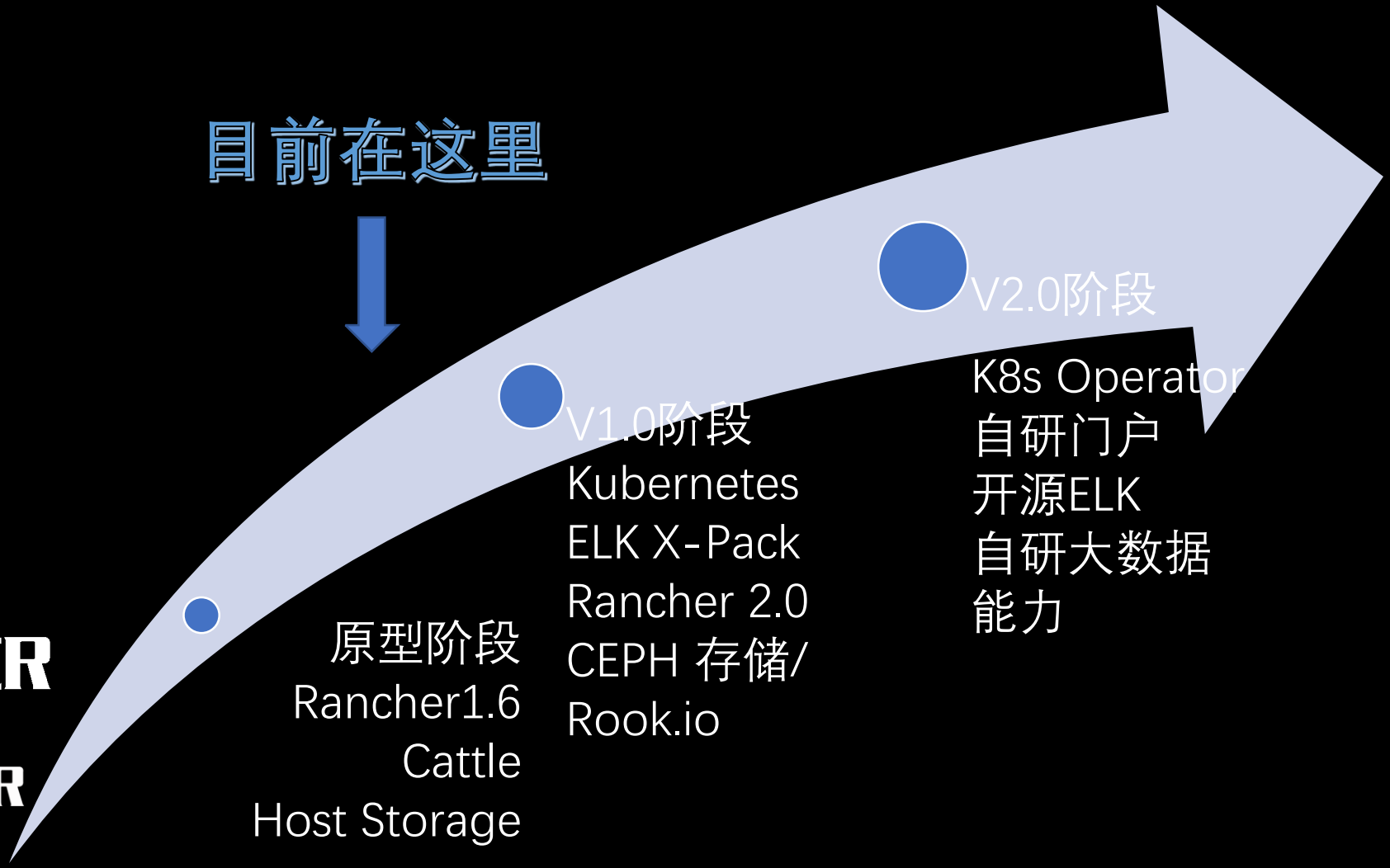
RANCHER | IT大咖说 知识共享平台

2018
CONTAINER DAY

3.新东方ELK云服务实践

RANCHER

新东方 ELK 云服务路线图



RANCHER 2018 CONTAINER DAY

服务清单

日志分析 公共服务	数据搜索 公共服务	其他 专有服务
• 基于ELK6.X白金商业版 • Hot/Warm数据分级架构 • 提供机器学习和Graph分析功能 • 对接日志和流量类数据	• 基于ELK6.X 黄金商业版 • 索引级别的隔离 • 较高的安全设置 • 索引副本配置不低于2副本 • 对接数据查询加速，搜索类数据	• 满足项目特殊要求或历史遗留项目 • 基于ELK 5.X/6.X开源版 • Nginx简单认证 • 多集群复用硬件资源

硬件设计

- 通用X86服务器
- >= 2 X 16 core CPU
- 256G~512G内存
- 2 X SSD Raid1 操作系统
- N X SSD Raid0
- N X SAS 旋转磁盘 Raid0
- 双万兆网卡和电源
- Centos7.4 / Ubuntu 1604
- 数据磁盘无LVM，ext4fs
- 设置Swappiness，vm.max_map_count，nproc等等

RANCHER
2018
CONTAINER
DAY

编排引擎和组件

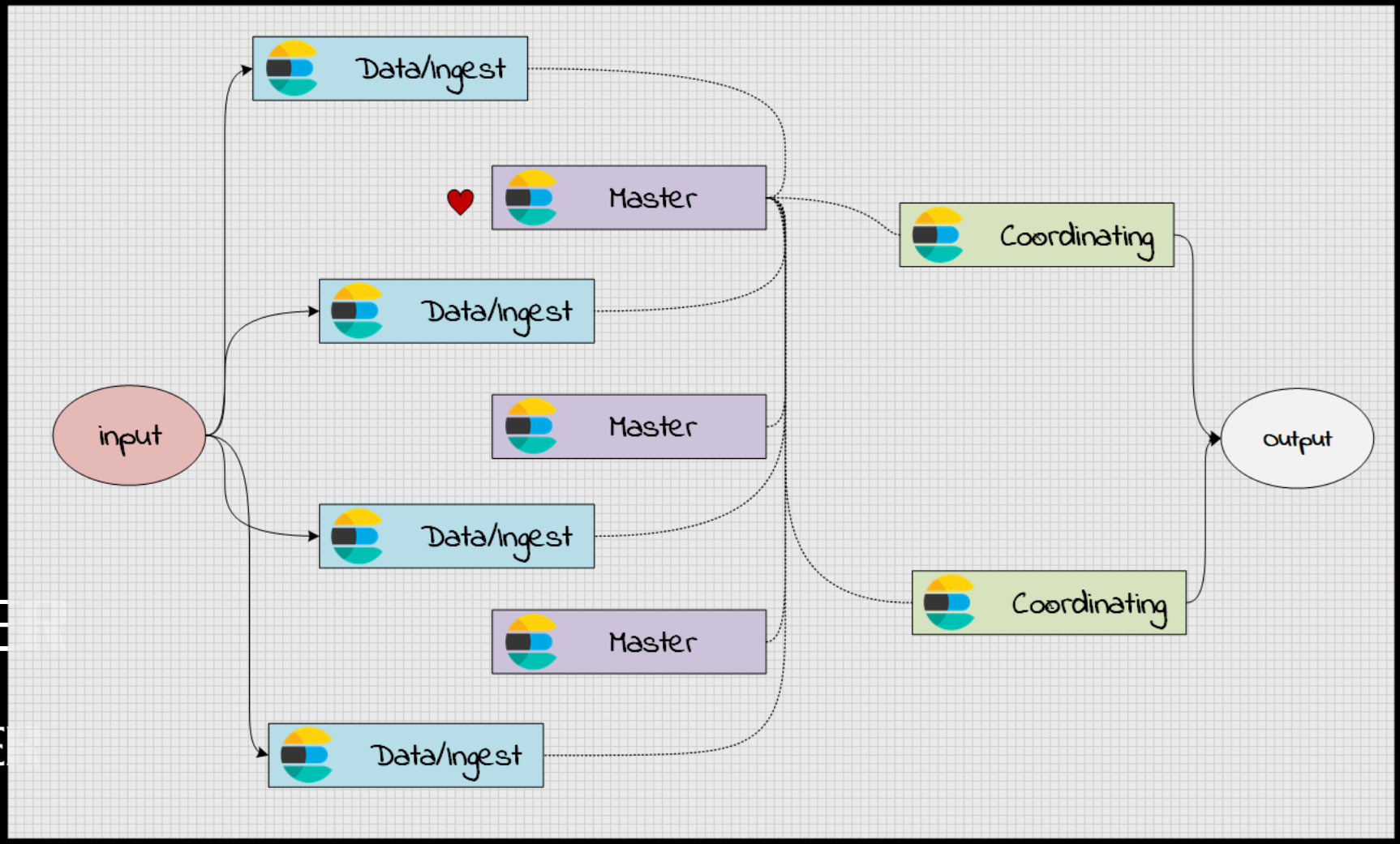
Rancher 1.6 + Cattle

- 镜像docker.elastic.co白金/黄金/OSS
- Docker 17.06.2-ce
- 自定义Rancher catalog
- Rancher vxlan
- Rancher cron
- Bind mount volume

Rancher 2.0 + K8S

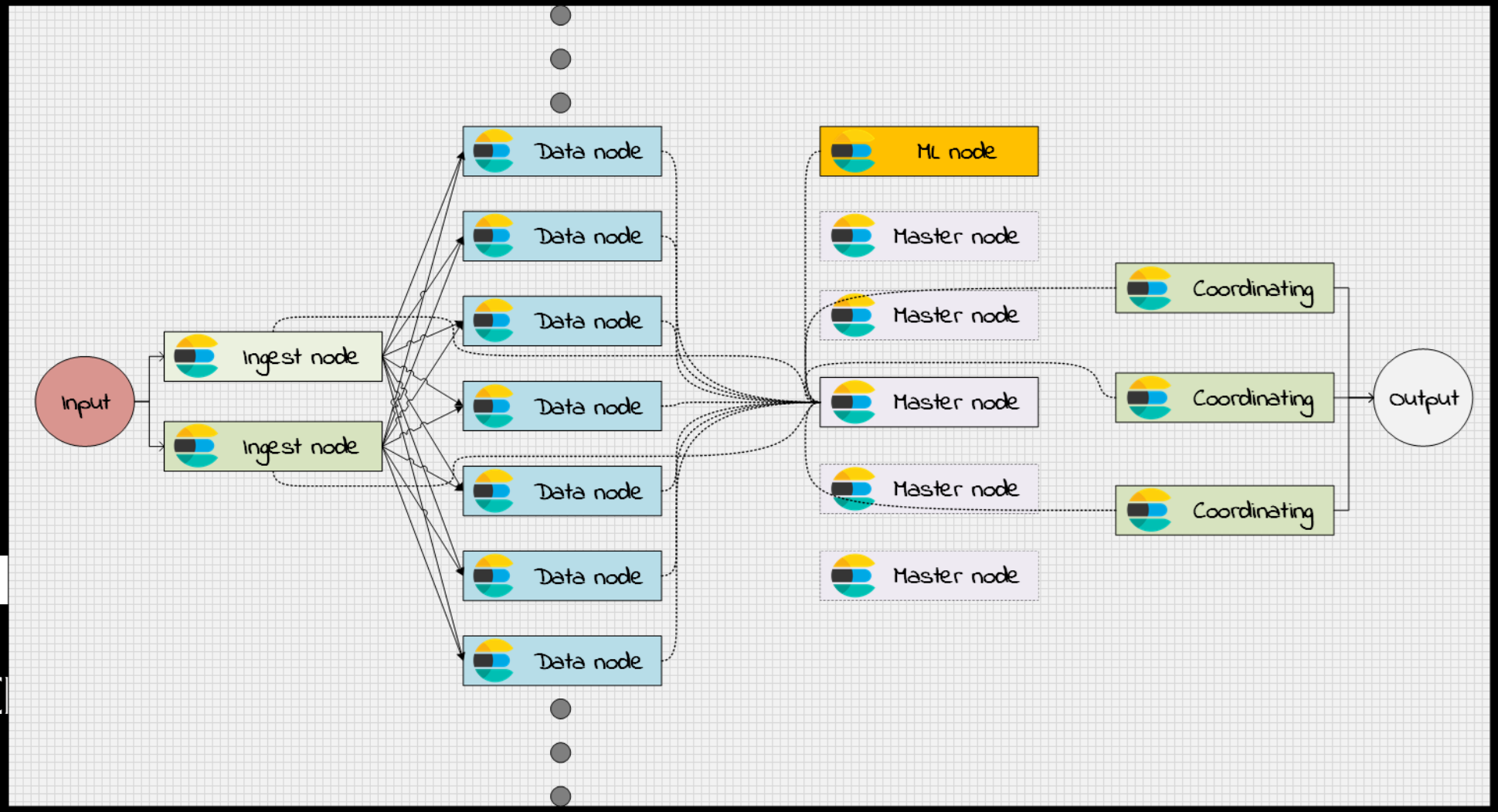
- 镜像docker.elastic.co白金/黄金/OSS
- Docker 17.03.2-ce
- K8S 1.10
- 自定义的Helm
- Statefulset/deployment/configmap/secret/cronjob
- StorageClass
- Ceph

部署方案（初始）

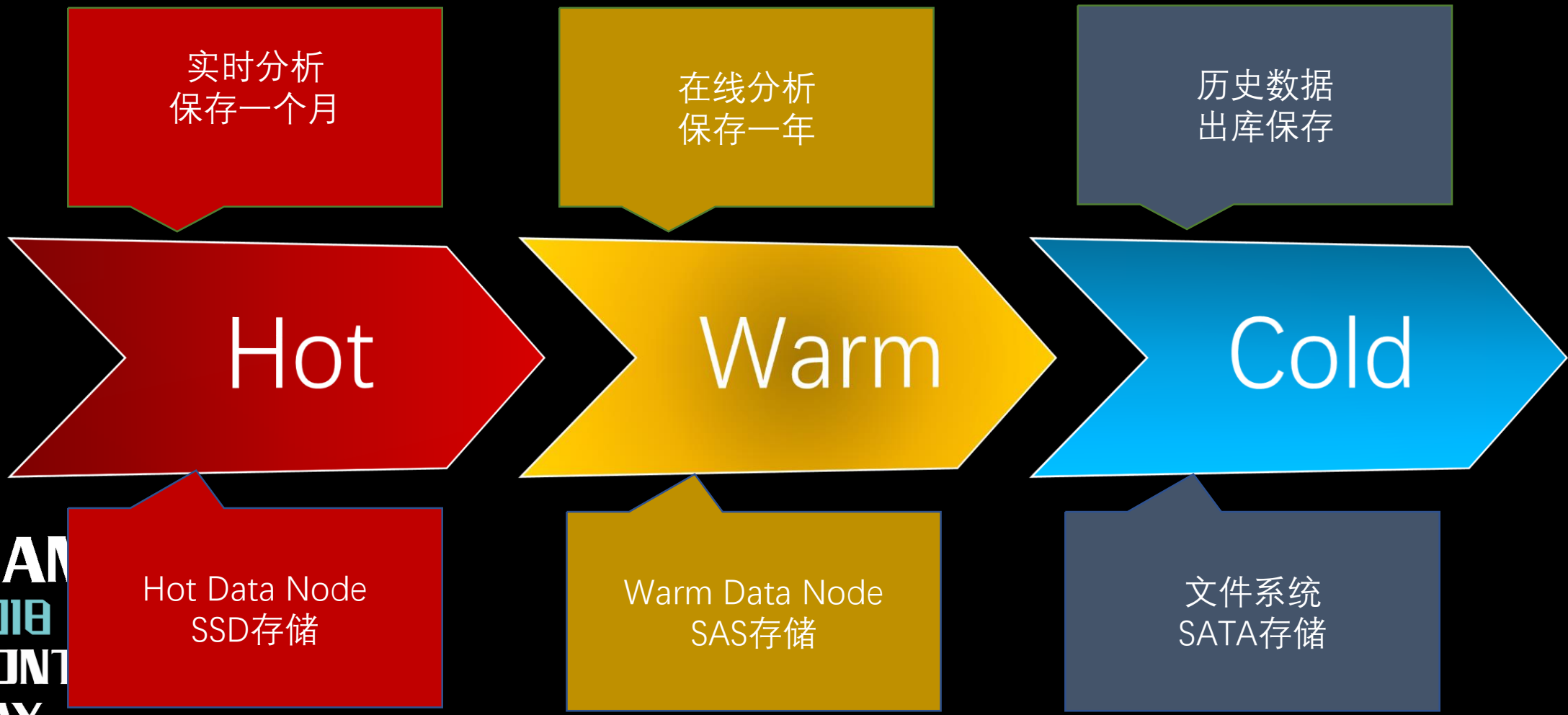


RANCHE
2018
CONTAINE
DAY

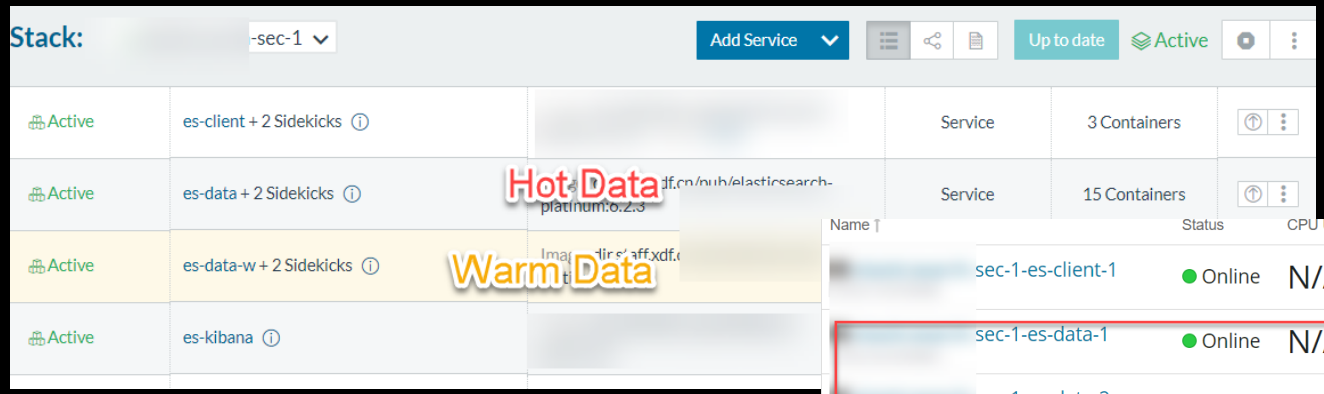
部署方案（最终）



Hot/Warm/Cold架构



Hot-Warm Data Node



Name	Status	CPU Usage	CPU Throttling	JVM Memory	Disk Free Space	Shards
sec-1-es-client-1	Online	N/A ↓ N/A max N/A min	0 ns ↓ 0 ns max 0 ns min	62 % ↓ 70 % max 57 % min	125.7 GB ↑ 125.8 GB max 125.7 GB min	0
sec-1-es-data-1	Online	N/A ↓ N/A max N/A min	0 ns ↓ 0 ns max 0 ns min	26 % ↓ 38 % max 25 % min	31.4 GB ↑ 31.6 GB max 22.3 GB min	28
sec-1-es-data-2	Online	N/A ↓ N/A max N/A min	0 ns ↓ 0 ns max 0 ns min	46 % ↑ 50 % max 30 % min	39.4 GB ↑ 39.4 GB max 20.6 GB min	34
sec-1-es-data-3	Online	N/A ↓ N/A max N/A min	0 ns ↓ 0 ns max 0 ns min	18 % ↑ 25 % max 12 % min	60.4 GB ↑ 60.4 GB max 18.7 GB min	25
sec-1-es-data-4	Online	N/A ↓ N/A max N/A min	0 ns ↓ 0 ns max 0 ns min	62 % ↑ 63 % max 39 % min	93.2 GB ↓ 94.5 GB max 92.7 GB min	72
sec-1-es-data-5	Online	N/A ↓ N/A max N/A min	0 ns ↓ 0 ns max 0 ns min	50 % ↑ 54 % max 30 % min	44.1 GB ↓ 55.4 GB max 43.8 GB min	78
sec-1-es-data-w-1	Online	N/A ↓ N/A max N/A min	0 ns ↓ 0 ns max 0 ns min	44 % ↑ 55 % max 26 % min	68.0 GB ↓ 68.0 GB max 68.0 GB min	67
sec-1-es-data-w-2	Online	N/A ↓ N/A max N/A min	0 ns ↓ 0 ns max 0 ns min	38 % ↑ 45 % max 19 % min	132 ↓ 132.3 GB max 132.2 GB min	15
sec-1-es-data-w-3	Online	N/A ↓ N/A max N/A min	0 ns ↓ 0 ns max 0 ns min	31 % ↑ 44 % max 18 % min	115.4 GB ↓ 135.9 GB max 115.4 GB min	10
sec-1-es-data-w-4	Online	N/A ↓ N/A max N/A min	0 ns ↓ 0 ns max 0 ns min	40 % ↑ 43 % max 17 % min	132.7 GB ↓ 139.7 GB max 131.2 GB min	12
sec-1-es-data-w-5	Online	N/A ↓ N/A max N/A min	0 ns ↓ 0 ns max 0 ns min	18 % ↑ 44 % max 18 % min	132.8 GB ↓ 136.5 GB max 131.1 GB min	11

Hot Data Node

Warm Data Node

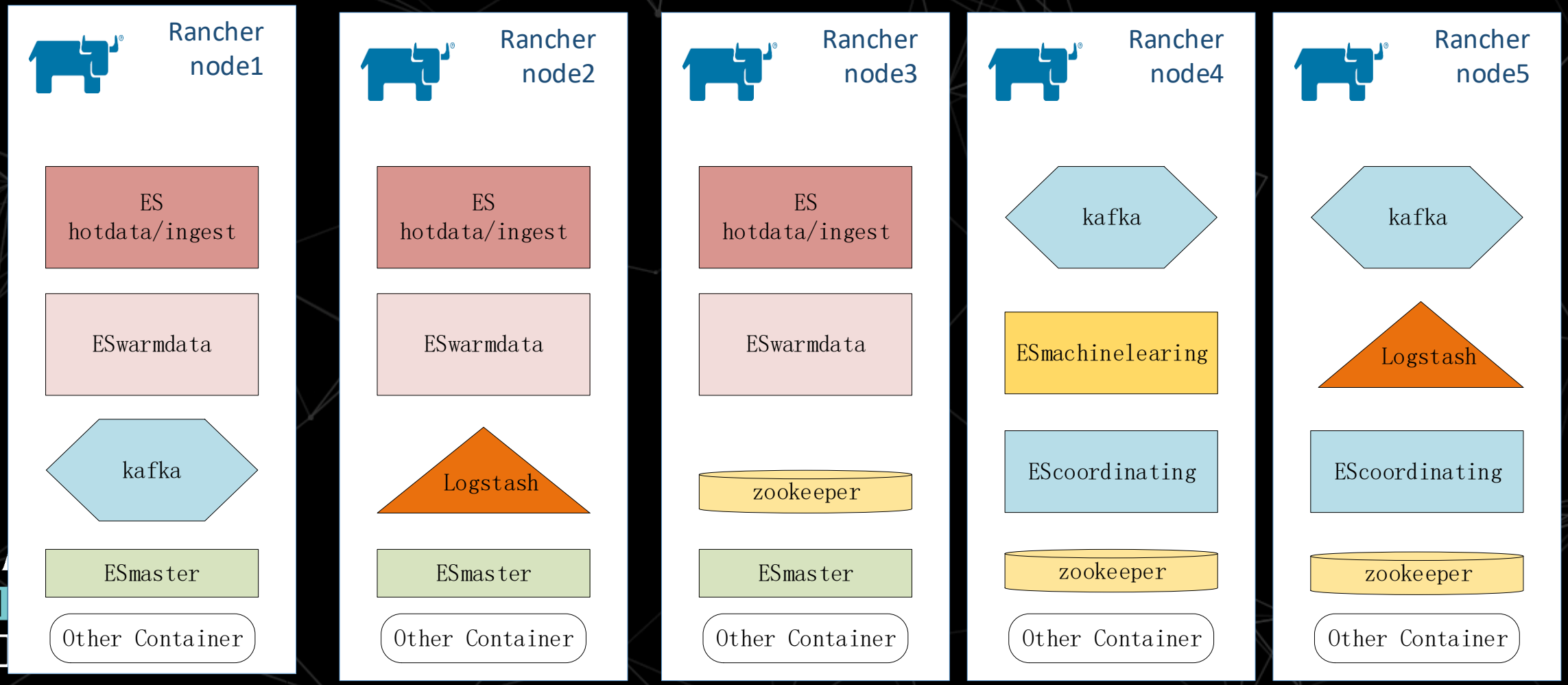
RANCHER
2018
CONTAINER
DAY

容器的编排举例（cattle）

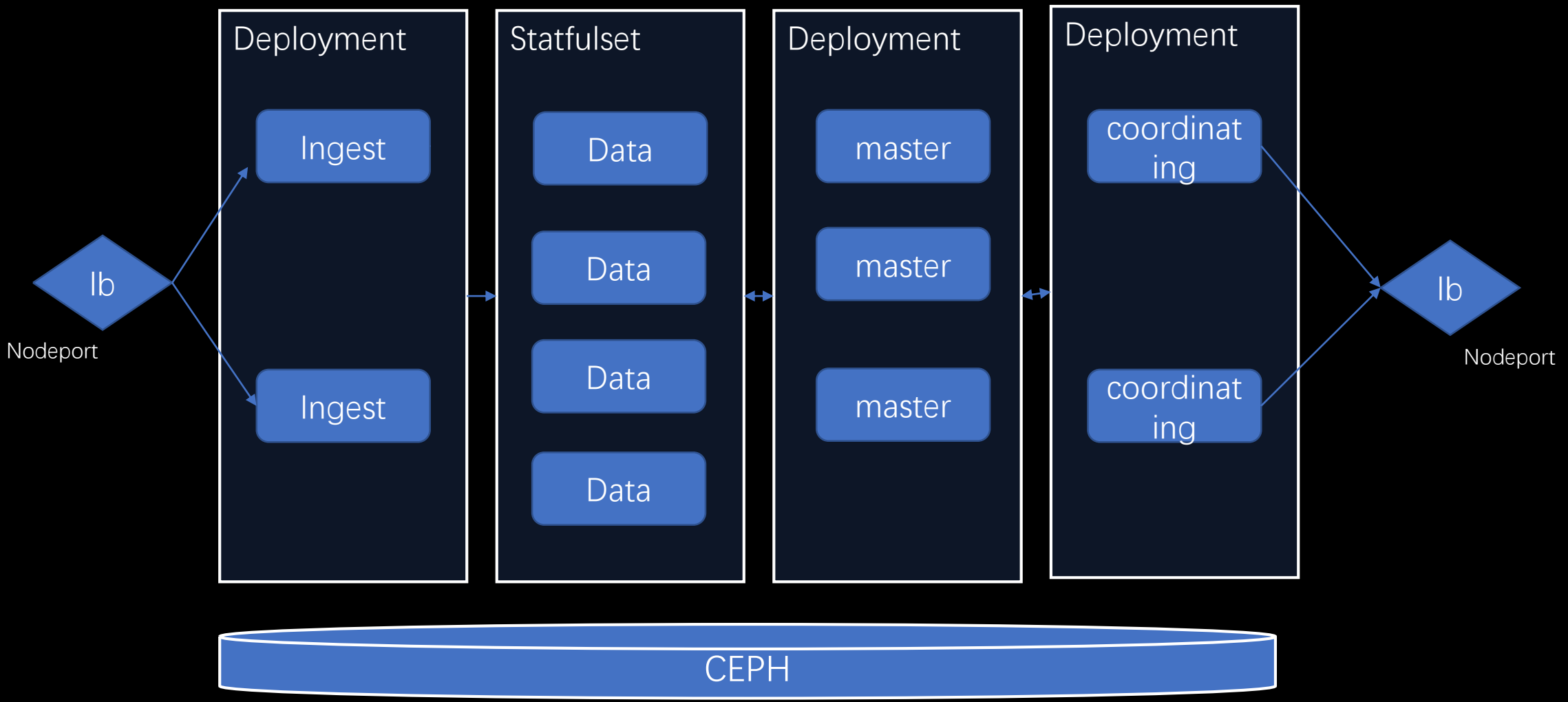
- 一个容器内存上限为64G内存
- 假设一台256G内存的物理机可以被分为五个部分或者更多：
  - 3 X 64G的标准容器
  - 32G内存用于跑低内存消耗的容器，如：master node，ml node ， zookeeper，logstash，kibanna，
  - 剩下的32G留给宿主机和环境容器，如网络组件、自动作业、cadvsior，cronjob container ，node exporter、fluentd等
- 其中标准容器用于跑以下三类容器：
  - Ingest node
  - Hot data node /Kafka broker （SSD）
  - Warm data node （SAS）
- 全部容器运行在Rancher vxlan网络上
- CPU按照比例分配，限制最大可用的share

RANCHER
2018
CONTAINER
DAY

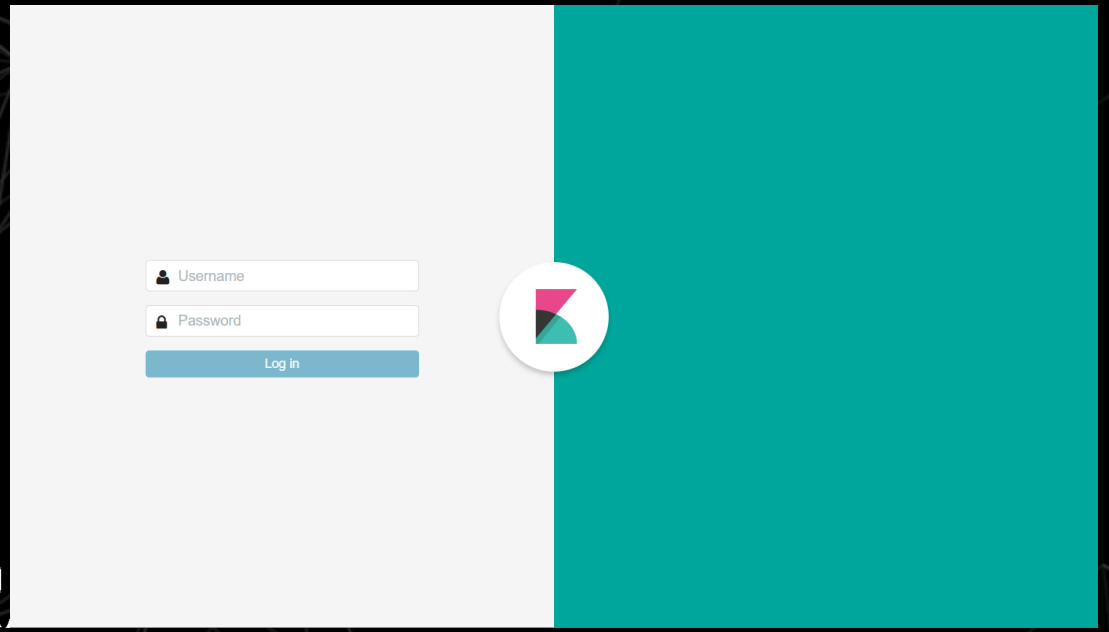
容器的编排举例

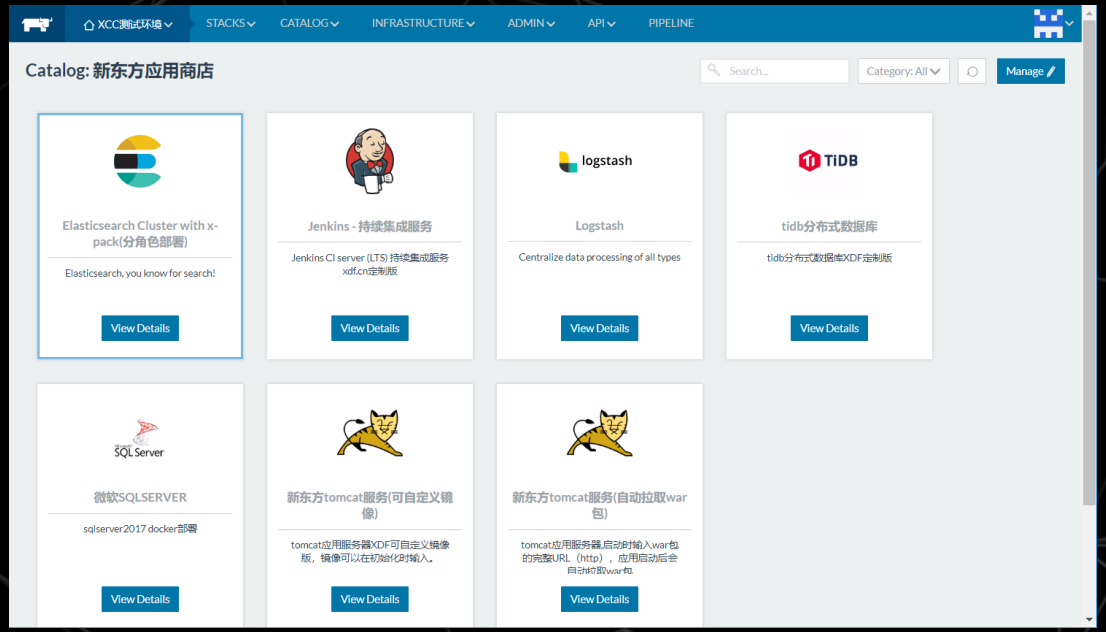

Rancher node1
ES hotdata/ingest
ESwarmdata
kafka
ESmaster
Other Container
Rancher node2
ES hotdata/ingest
ESwarmdata
Logstash
ESmaster
Other Container
Rancher node3
ES hotdata/ingest
ESwarmdata
zookeeper
ESmaster
Other Container
Rancher node4
kafka
ESmachinelearing
EScoordinating
zookeeper
Other Container
Rancher node5
kafka
Logstash
EScoordinating
zookeeper
Other Container

容器的编排(K8s)草案


Deployment
Statfulset
Deployment
Deployment
Ingest
Ingest
Data
Data
Data
Data
master
master
master
coordinat ing
coordinat ing
lb
Nodeport
lb
Nodeport
CEPH

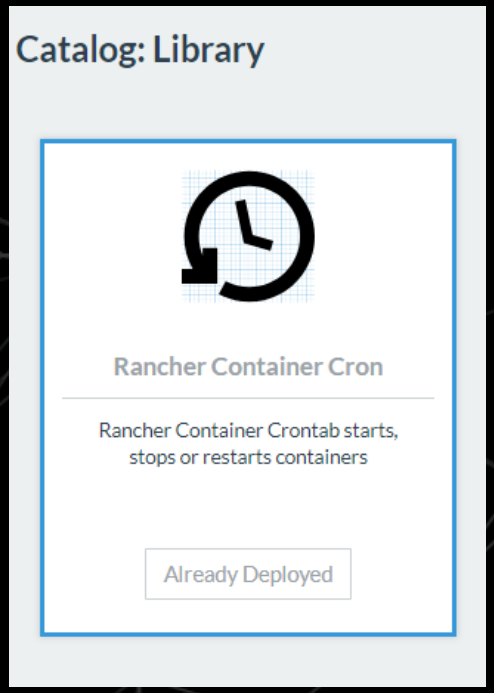
运维：前后台运维界面

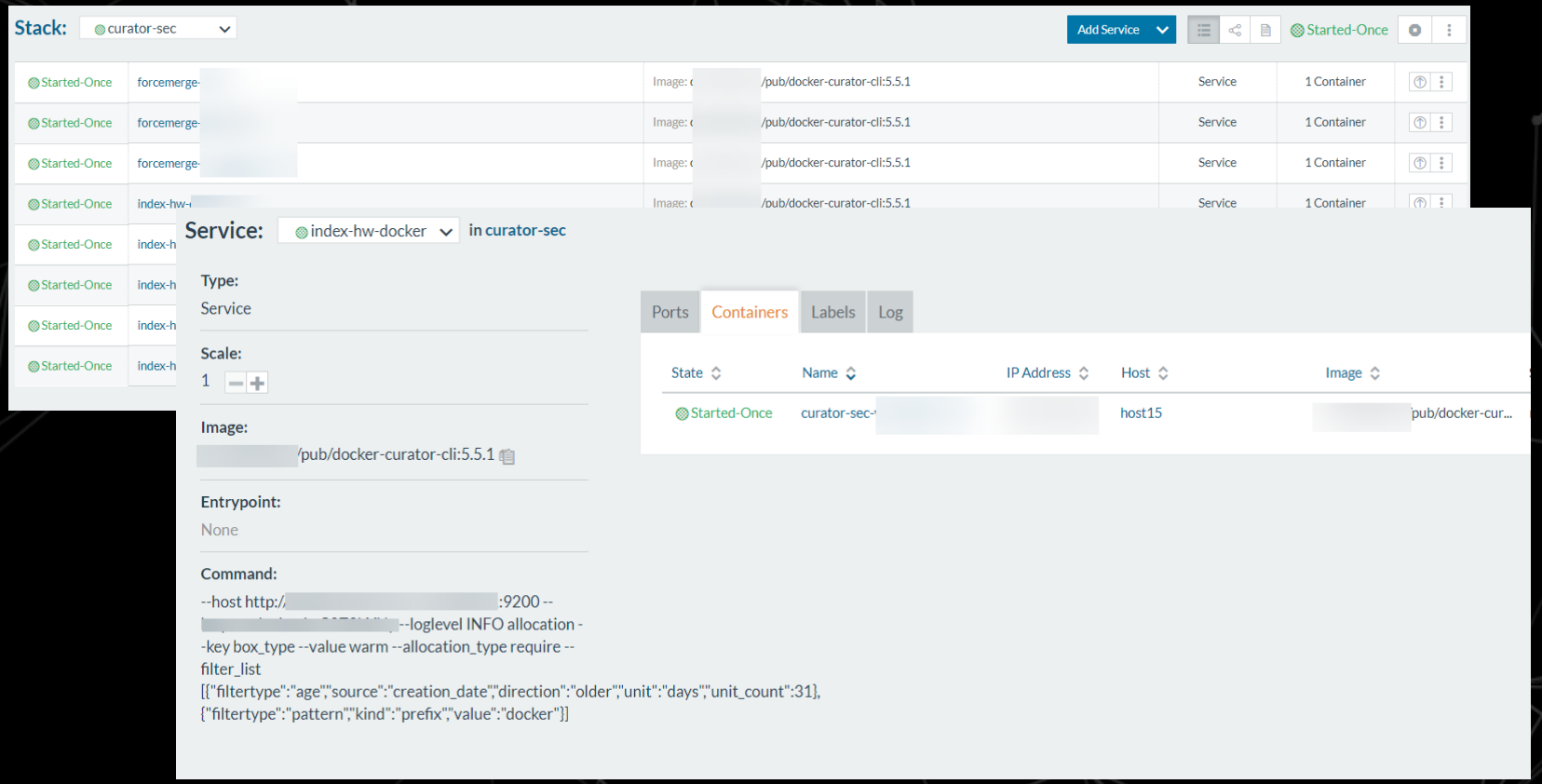




RANCHER
2018
CONTAINER
DAY

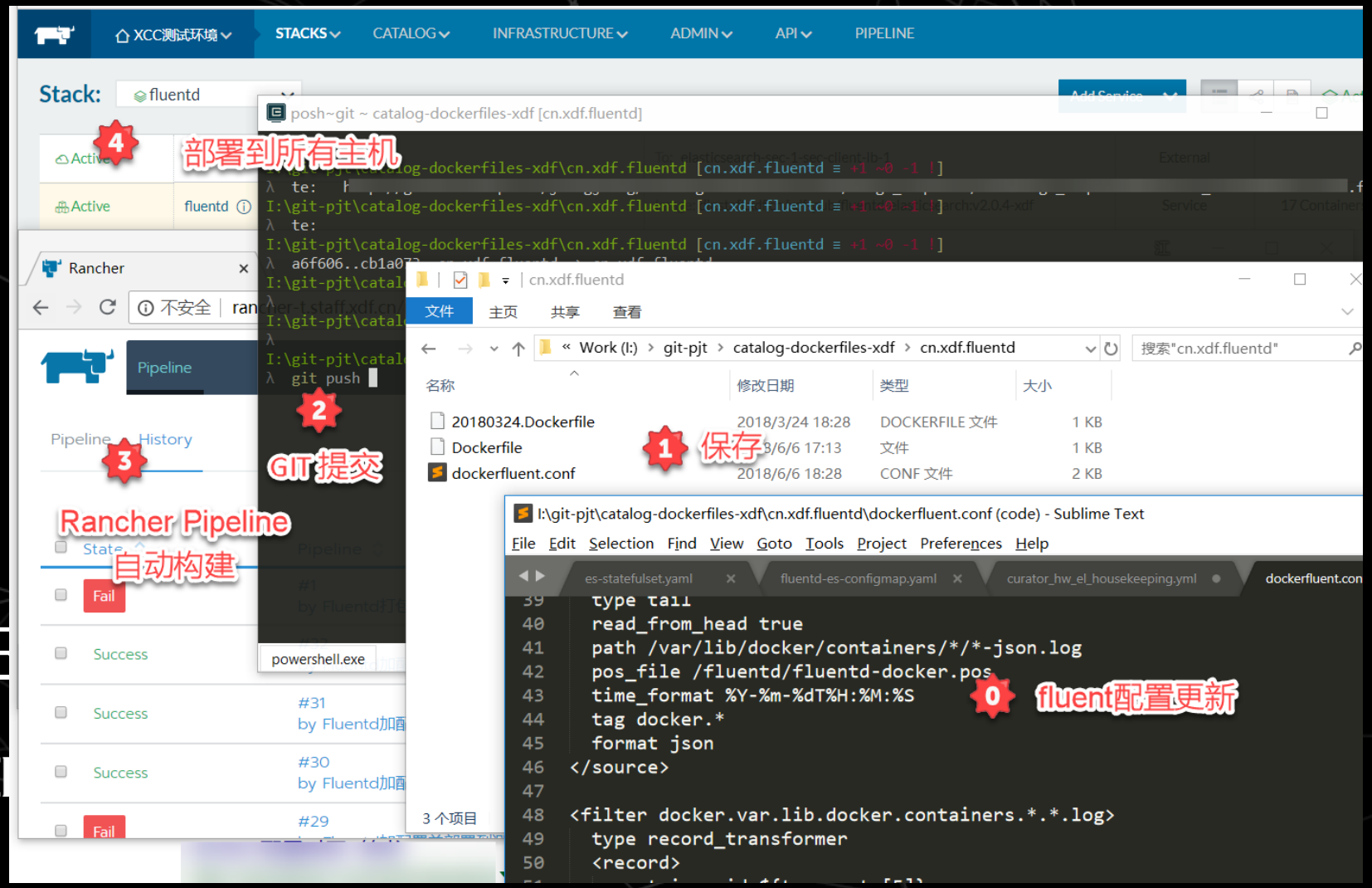
运维：容器化运维





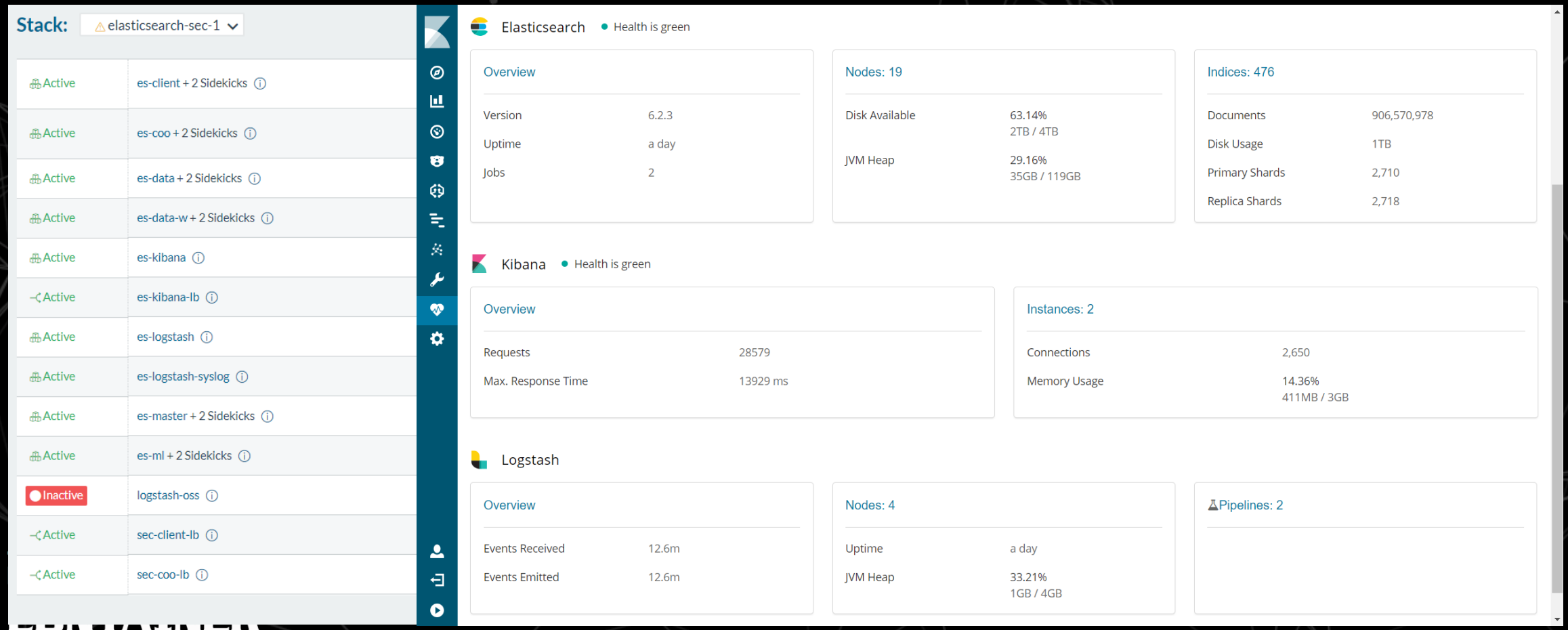
RANCHER
2018
CONTAINER
DAY

运维：容器化组件的配置管理



RANCHER
2018
CONTAINER
DAY

1/4配测试环境举例



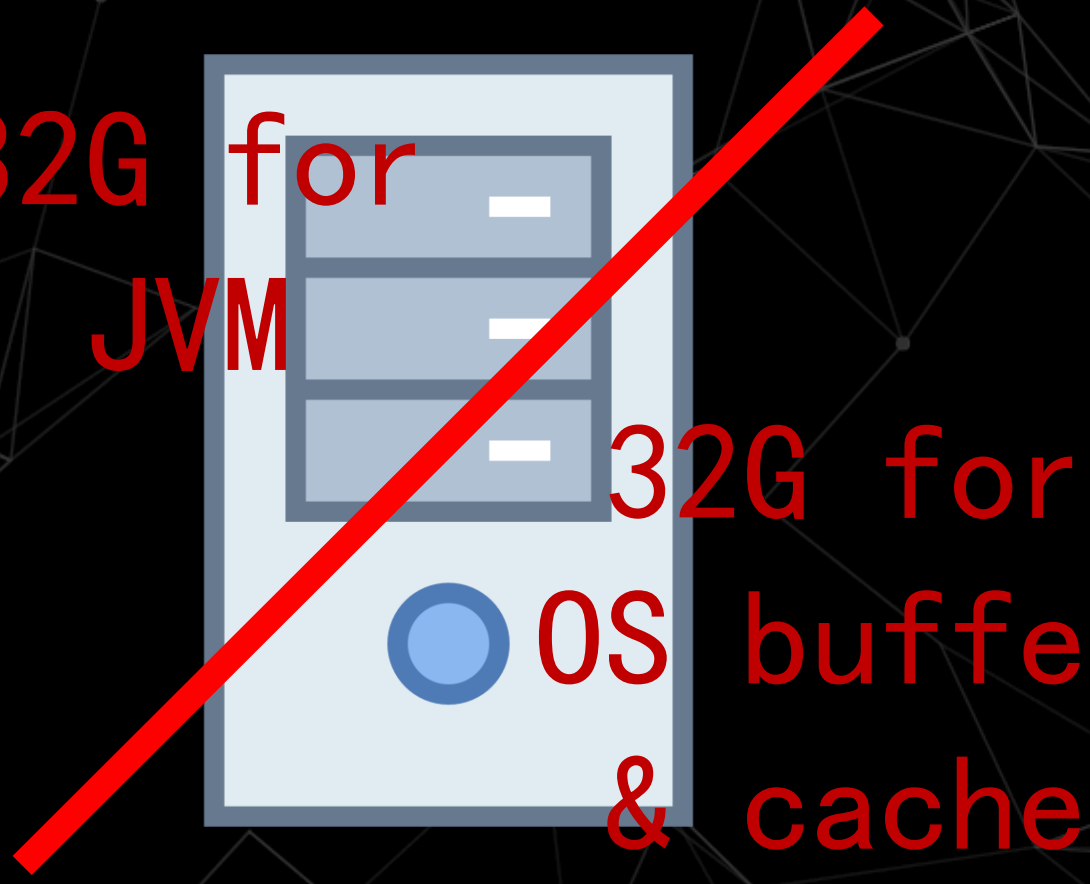
DAY

2018
CONTAINER DAY

4.实践中遇到的坑

RANCHER

ES内存分配5/5原则

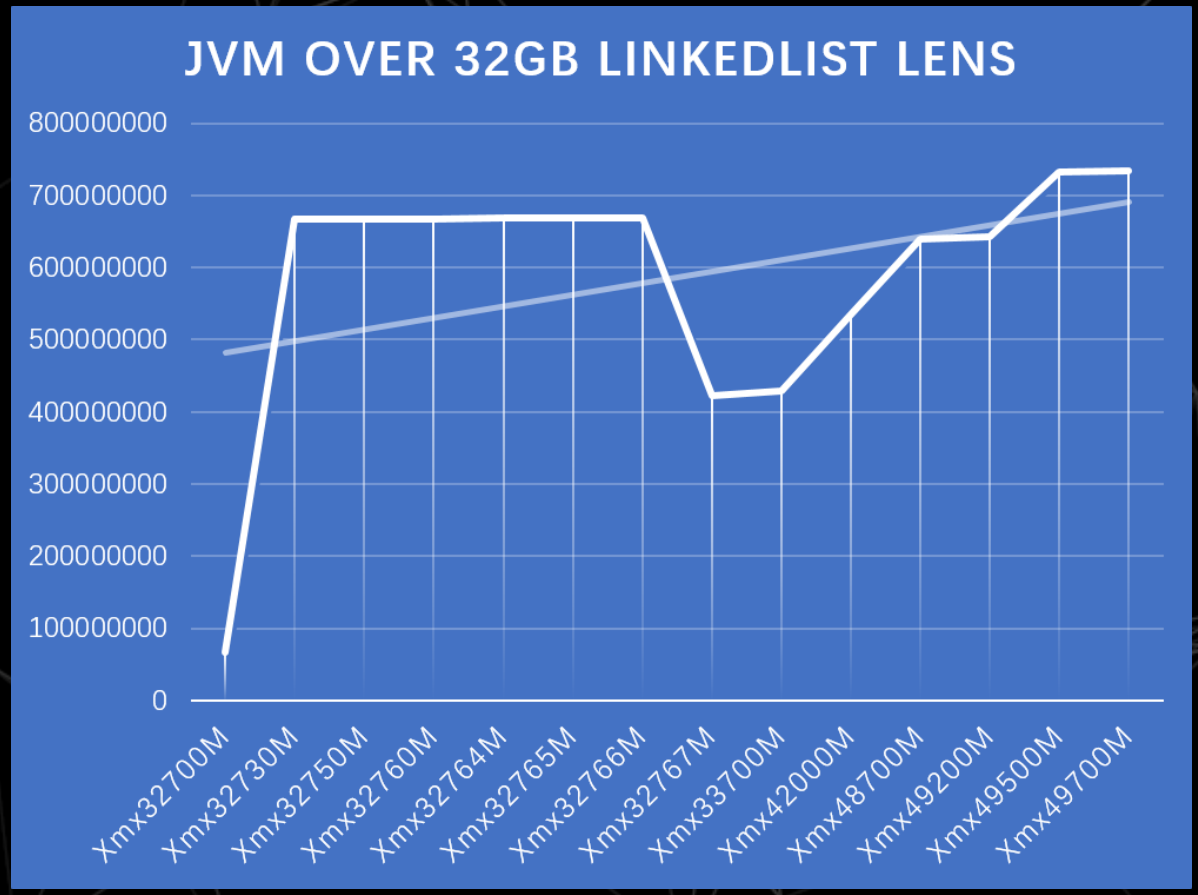


RANCHER
2018
CONTAINER
DAY

为什么 Jvm Heap <=32G ？Jvm Oop

http://java-performance.info/over-32g-heap-java/

```java
public class Mem32Test {
    public static void main(String[] args) {
        List<Integer> lst = new LinkedList<>();
        int i = 0;
        while ( true )
        {
            lst.add( new Integer( i++ ) );
            if ( ( i & 0xFFFF ) == 0 )
                System.out.println( i ); //shows where you are 🙂
            if ( i == System.currentTimeMillis() )
                break; //otherwise will not compile
        }
        System.out.println( lst.size() ); //needed to avoid dead code optimizat
    }
```



RANCHER
2018
CONTAINER
DAY

滚动操作

- 一个节点操作完成后休息适当时间，随后继续下一个节点的操作，以避免数据丢失
- cattle中升级service时注意设置间隔时间
- 幸运的是：滚动升级是Rancher Cattle 升级服务的默认行为
- K8S中data node 使用 statefulset，默认按顺序滚动操作，其他node可以直接使用deployment

Shards设置

- 每个node小于1000个Shards
- Shards size < 单个Data Node的JVM size
- Shards 并非越多越好，取决于数据量

副本设置

- 至少设置1副本
- 重复一遍至少1副本
- 搜索、查询加速类推荐2副本

RANCHER
2018
CONTAINER
DAY

混合负载注意事项

- 容器必须开启资源隔离
- 相同集群的data node 不允许跑在相同的Host上
- 按照不征用资源原则混合负载，比如计算类与存储类混合等。
- 征用同类资源负载尽量不混合。比如logstash 和 ml node
- 始终在宿主机上保留一点空余资源。

Rancher Cattle 编排的坑

- 任何编排引擎都不知道你跑的负载有什么特性，除非你让他知道
- Cattle 编排算法是按节点上容器最少原则调度容器，而不关心CPU、内存消耗情况！在cattle不断尝试调度的过程中可能引发“调度雪崩”
- 用Label限定调度，而不能将调度完全交给Cattle

RANCHER
2018
CONTAINER
DAY

RANCHER | IT大咖说 知识共享平台

2018 CONTAINER DAY

THANK YOU !

姜江 | 新东方信管部

RANCHER