TRANSWARP

大数据云和应用生态的构建之路

陈夏明 博士

星环信息科技（上海）有限公司
www.transwarp.io

内容

- 背景
- 复杂应用编排
- 数据云架构演进
- Q&A

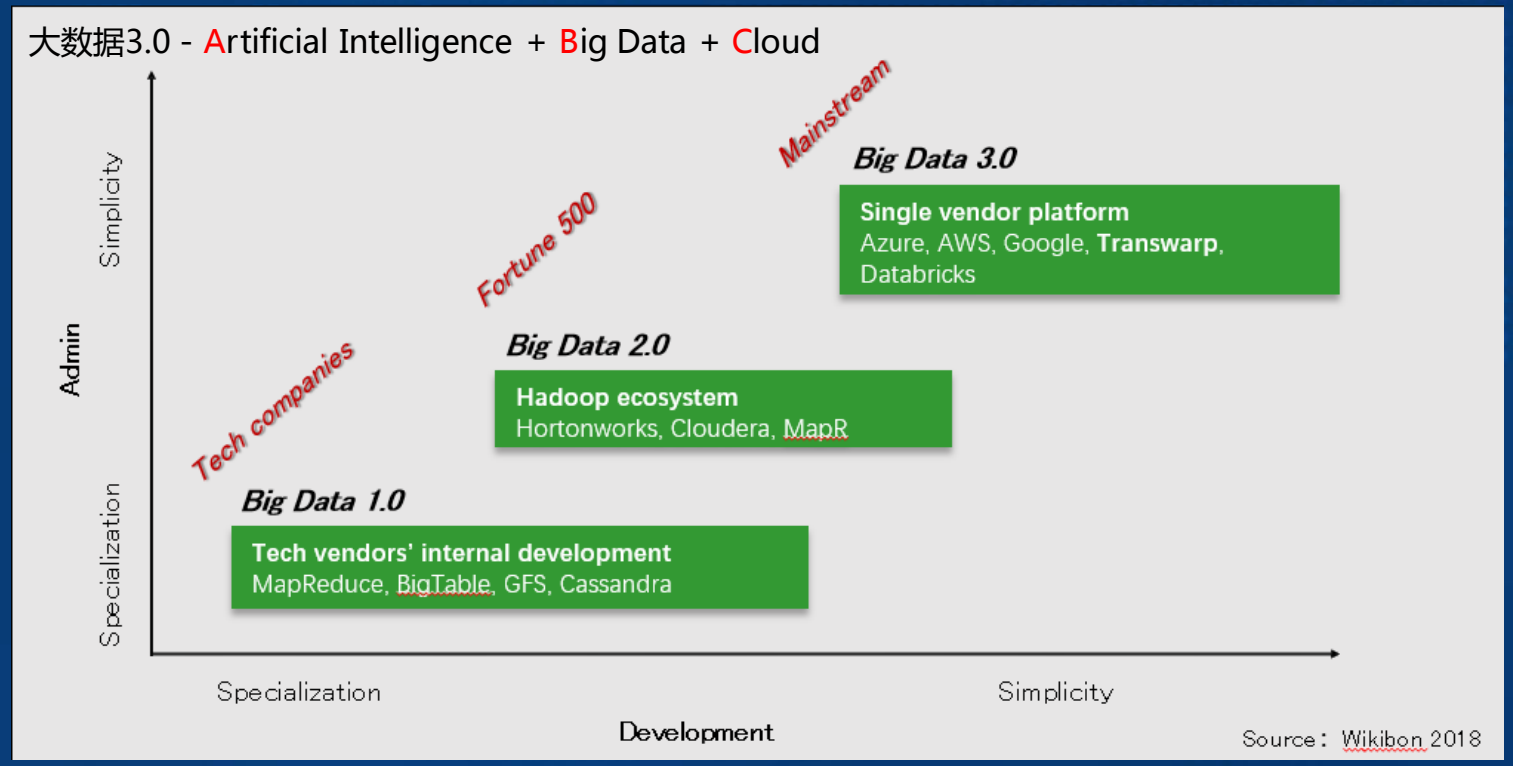
大数据和云技术发展历程

大数据市场格局

Gartner Visionary (Global), Feb. 2016

IDC Market Scape (China)，Oct. 2017

大数据行业发展趋势



BigData 3.0：提供“ABC”技术融合的综合平台，满足客户多元化、复杂化的需求，同时降低开发和管理难度，在整个产业链占据更重要地位。

数据生态化
- 数据驱动业务闭环
- 服务和应用共享
- 数据对外赋能

数据服务化
- 数据化运营
- 智能应用
- 在线数据服务

数据资产化
- 多源数据融合
- 数据质量管理
- 资产化与计量

数据集约化
- 数据集中采集
- 统一的元数据
- 统一的计算平台

大数据3.0应用生态


采集
整合
存储
分析
高级分析
可视化
Cloud DB
RDBMS
Files
Sensors
APP
Kafka
Filebeat
Logstash
Sqoop
KunDB
Transporter
Workflow
Waterdrop
HDFS
Search
Hyperbase
TxSQL
Redis
Inceptor
Slipstream
Governor
Holodesk
Sophon
Graphene
Discover
Rubik
Midas
Pilot
Kibana
Notebook
On Premise
Transwarp Data Cloud
Public Cloud

复杂应用编排

Image source: https://www.clouda.ca/shipping-container-above-stacked-others/

应用部署技术变迁

TDH 4.X
(released 2015)

TDH 5.X
(released 2017)

TDC 1.X
(released 2018)

物理机
计算机元年

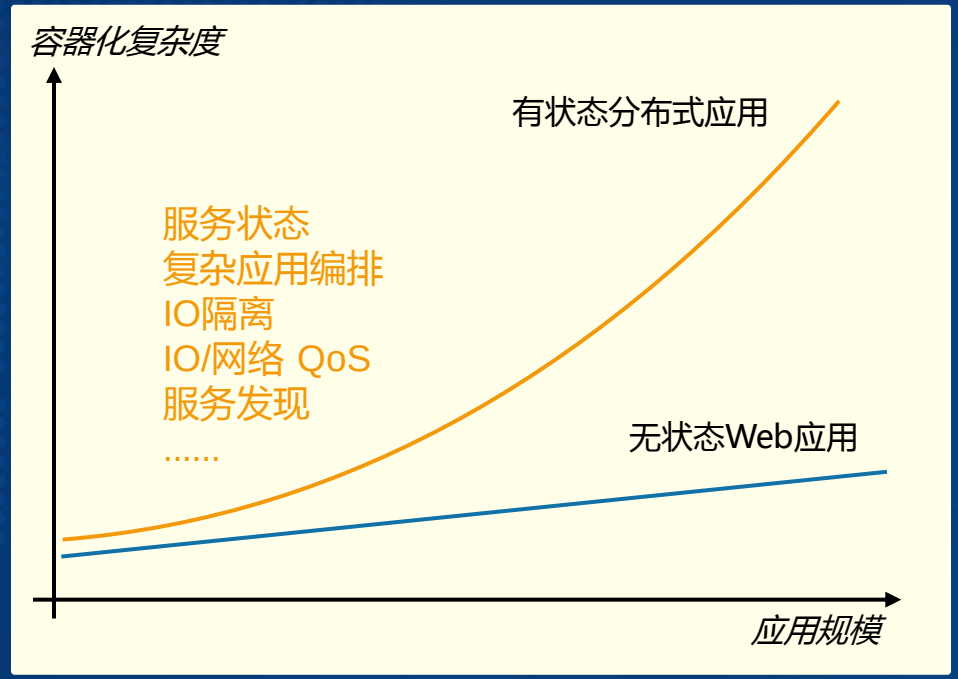
虚拟机云
OpenStack, 2010

容器技术
Docker, 2013

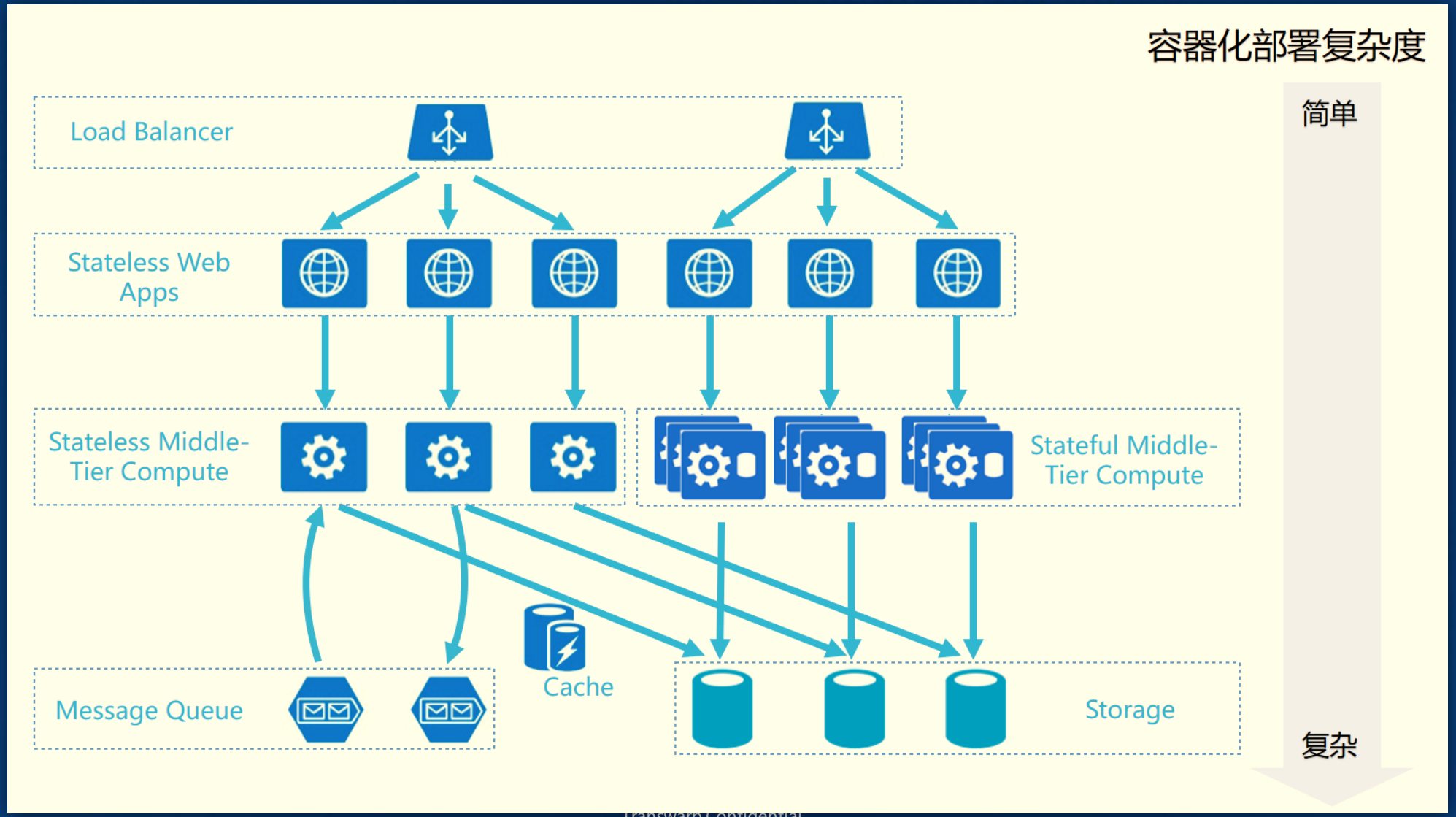
容器云编排
Kubernetes, 2014

应用编排

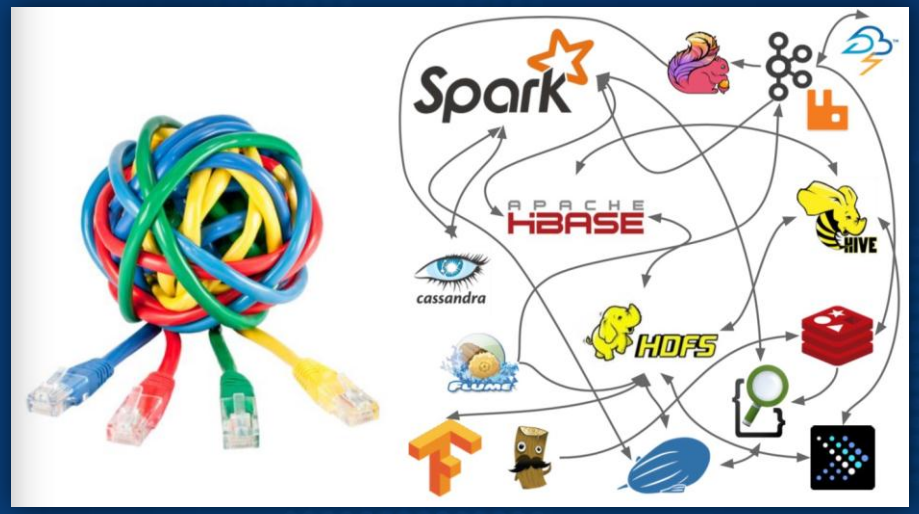
- 协调多个服务，从而对外提供更强大的功能
- 编排的问题范畴
  - 配置
    - 系统配置（cpu，memory，storage）
    - 服务配置（容器内应用的配置）
  - 调度
  - 网络
  - 存储
- 编排工具
  - docker-compose, swarm, mesos, kubernetes etc.



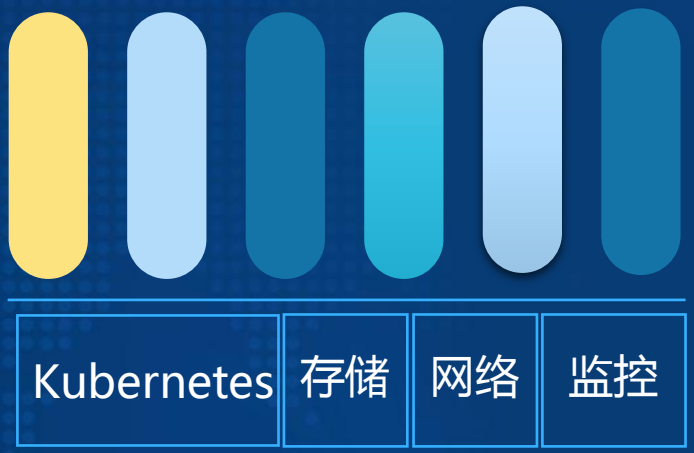
不同应用的容器化难度


容器化部署复杂度
简单
复杂
Load Balancer
Stateless Web Apps
Stateless Middle-Tier Compute
Stateful Middle-Tier Compute
Message Queue
Cache
Storage

大数据应用的服务拆分



拆分



- 提前规划节点的服务
- 一个物理集群很难安装多个Hadoop集群
- 容易与第三方应用有兼容性问题
- 升级操作复杂

传统部署

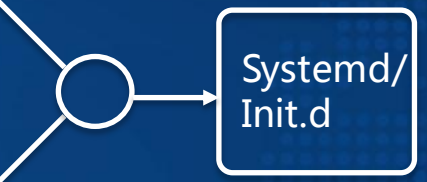
- 每个服务按需灵活扩容/缩容、上线/下线、升级/降级
- 拆分完之后，解决服务间依赖和服务发现的问题
  - 集群内使用Headless service
  - 服务之间通过DNS和配置注入进行服务发现
  - 集群外的访问
    - NodePort
    - REST服务 + LoadBalancer

容器部署

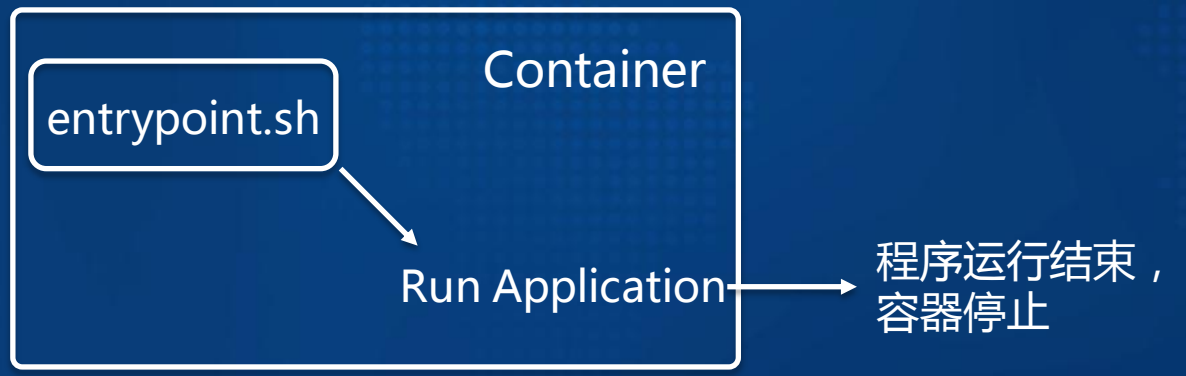
容器镜像标准化，e.g. Hadoop

/etc/init.d/hdfsnamenode.sh

/etc/hdfs/conf/hadoop-env.sh



原生hadoop启动比较复杂；启动后即停止



程序运行结束，容器停止

启动方式

方案一：配置打包到容器，通过变量替换修改

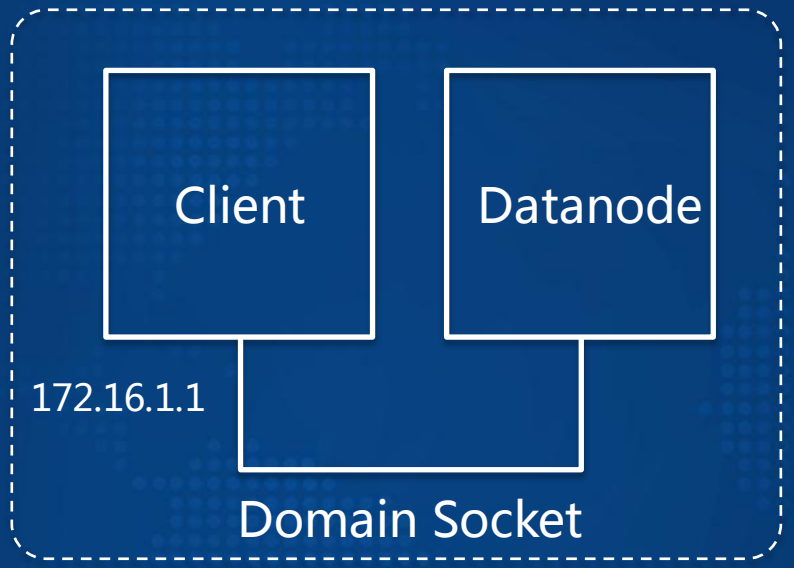
- 优点：配置与image匹配，本地测试比较容易
- 缺点：修改/扩展配置比较麻烦

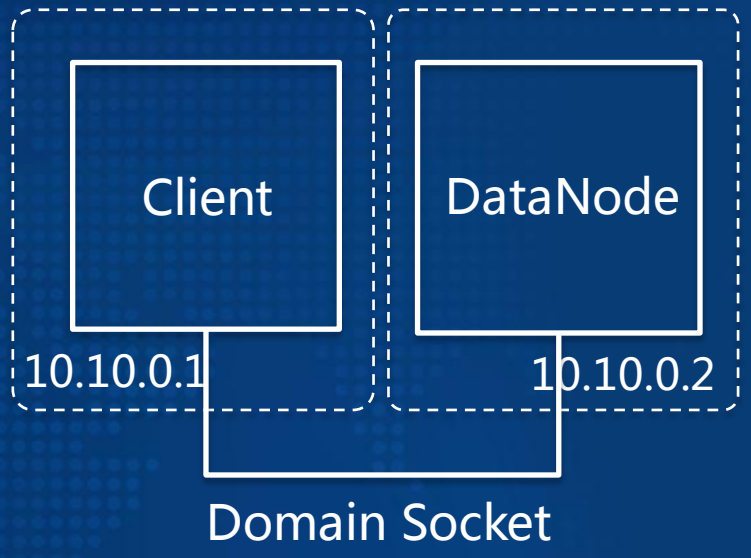
方案二：配置通过ConfigMap的方式挂入

- 优点：修改/扩展配置容易
- 缺点：需要配置管理模块，本地测试比较麻烦

配置文件

Data Locality





- 通过判断IP是否一致决定是否进行本地读写
- Domain socket优化，免去本地IO走tcp协议栈

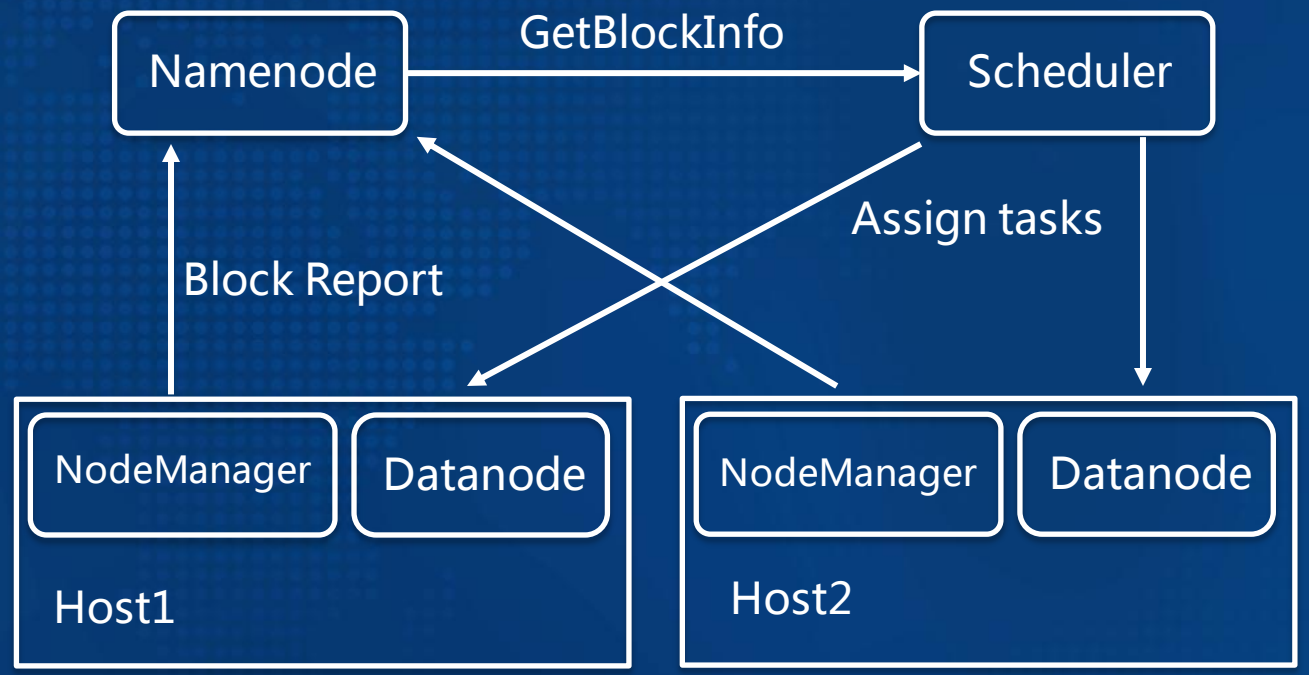
- 每个容器有一个独立IP
- 每个容器独立的文件系统

需要的改进：

- 更改Locality 判断逻辑
- 多个pod共享Domain socket，而且每个租户的hdfs的domain socket互相独立

Yarn/Spark 调度逻辑

- 更改Yarn/Spark调度逻辑，判断对应的计算切片是否在同一台host上



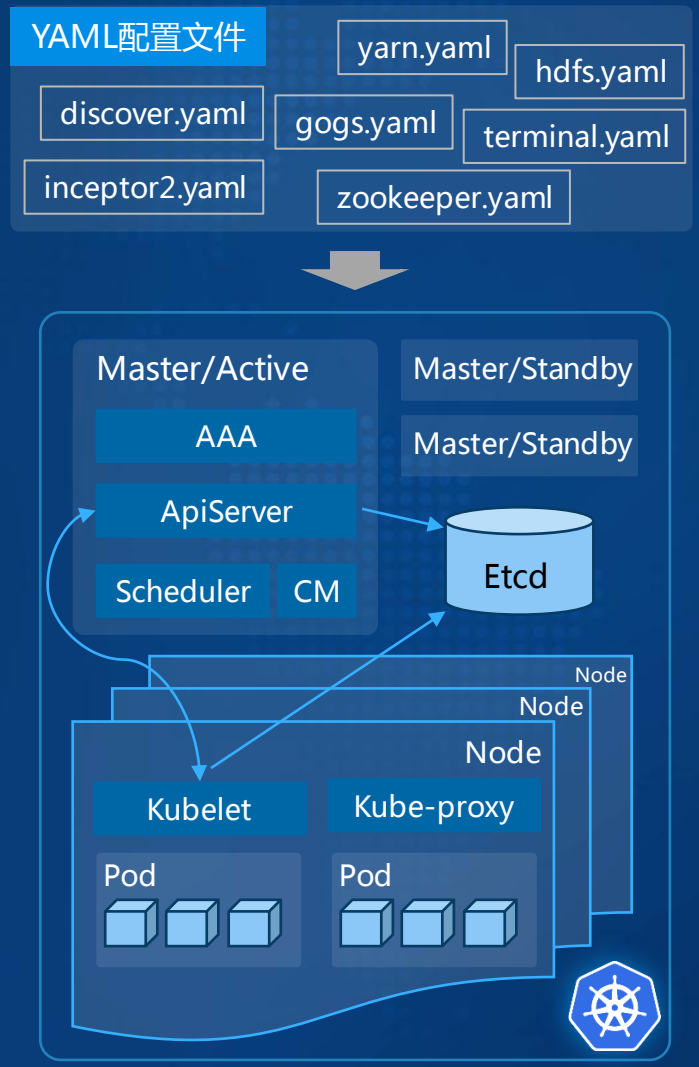
复杂应用编排策略

- 服务拆分（细分应用程序为不同的微服务，并包装在各自的容器中）
- 镜像标准化（镜像启动入口、配置文件挂载方式、CI/CD流程化）
- 动态编排（优化这些容器的资源利用率）
- 调度策略（Disk/Network Throughput，DiskSize，DiskType，CPU，Memory，Priority）
- 亲和性（计算容器和存储容器在同一个节点，降低网络IO开销）
- 反亲和性（多副本在不同节点，保证可用性）
- 服务优先级（实时计算任务和批处理任务拥有不同的等级）
- 资源超售（在不影响应用正常运行的情况下，降低成本）
- P2P镜像分发

数据云架构演进

Image source: https://www.wired.com/story/container-ships-use-super-dirty-fuel-that-needs-to-change/

数据云架构 1.0



- Kubernetes，v1.2
  - 早期版本，框架很灵活，功能相对较弱
  - 产品化程度低，需要用户手动配置大量YAML参数
  - 需求多实例、多租户
  - 需求依赖注入
  - 需求Image更换
  - 需求特定服务开关，如Kerberos安全
  - ……

大数据应用编排 1.0

模板字符串替换

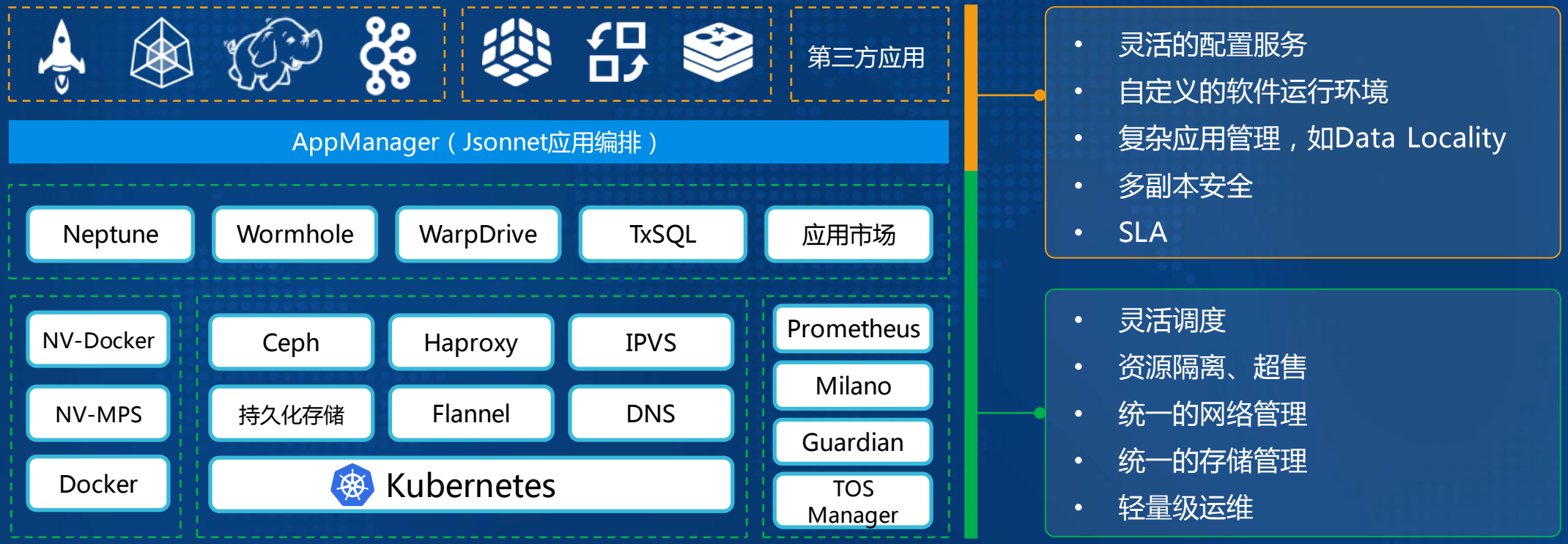
- Python模板JinJa/Django
- Java模板FreeMarker

```
and an empty
reopened with

resourcequota
* : Invalid v
invalid chara

piVersion: v1
ind: ResourceQ
etadata:
 creationTimestamp: 2018-11-23T02:50:05Z
 name: billing.quota
 namespace: billing
 resourceVersion: "66247062"
 selfLink: /api/v1/namespaces/billing/resourcequotas/billing.quota
 uid: 7b269430-eeca-11e8-ab82-0cc47ae2fc6e
pec: {
ard:
   limits.cpu: 1k
   limits.memory: 1028991300Ki
   requests.cpu: 1k
   requests.memory: 1028991300Ki
 prioritizedHards:
 - priority: 100
   resources:
     limits.cpu: "100"
     limits.memory: 100Gi
     requests.cpu: "100"
     requests.memory: 100Gi

tatus: {}
```

带来的挑战

- 通用功能的维护代价大，复用度底
  - k8s模板中相似的部分需要复用
  - 如node selector, network name, 定制存储卷等
- 测试复杂
  - 模板渲染是否符合预期
  - 模板渲染出的YAML是否符合k8s语法
- 复杂的逻辑需要借助编程语言，如Python/Java
- 版本衍生代价大

数据云架构 2.0



版本总结：配置灵活，但是普通用户上手难度较大；面向业务程度低

大数据应用编排 2.0

- Jsonnet模板
  - Google开源的一门动态配置语言，增强并兼容JSON
  - 在JSON基础上增加新特性：
    - 注释
    - 引用
    - 算术与逻辑运算
    - 条件操作符
    - 数据与对象深入
    - 引入、函数
    - 局部变量
    - 继承
    - ……

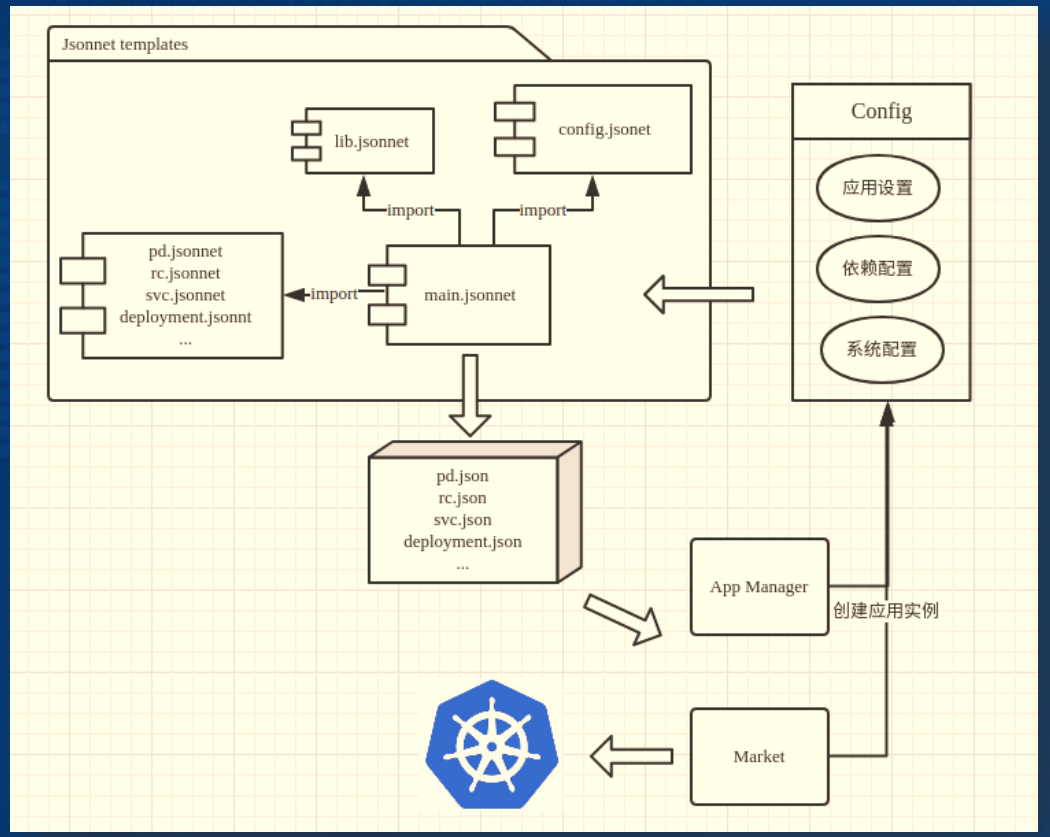
```
# Copyright 2016 Transwarp Inc. All rights reserved.

local kube = import "../../../applib/kube.libsonnet";
local app = import "../../../applib/app.libsonnet";
local hdfs = import "./hdfs.libsonnet";
local hdfs_datanode_common = import "./hdfs-datanode-common.jsonnet";

local default_config = {
};

function(user_config={})
  local config = default_config + user_config;
  kube.tos.Deployment(
    kube["extensions/v1beta1"].Deployment
    (
      generateName="hdfsdatanode",
      moduleName="hdfsdatanode",
      config=config
    ) {
      spec+: {
        replicas: config.hdfs_data_replicas,
        strategy: kube["extensions/v1beta1"].DeploymentStrategy(config.hdfs_update_strategy_configs),
        template+: hdfs_datanode_common(config) {
          spec+: {
            volumes: [
              kube.v1.PersistentDirVolume("datanodedir", {
                datanode: "1",
                ["datanode.install." + std.toString(config.Transwarp_Install_ID)]: "true",
              }),
              kube.v1.HostShareDirVolume("socketdir", "/root/docker/common"),
              kube.v1.EmptyVolVolume("transwarp", "10000000"),
            ],
            priority: app.parseInt(config.hdfs_data_priority),
          },
        },
      },
    },
    config
  )
```

HDFS datanode deployment via jsonnet

https://jsonnet.org/

大数据应用编排 2.0 (cond.)

- 主要模块
  - app.yaml 描述依赖关系，依赖变量注入
  - config.jsonnet 描述配置参数，定义输入
  - main.jsonnet 模板入口，定义输出



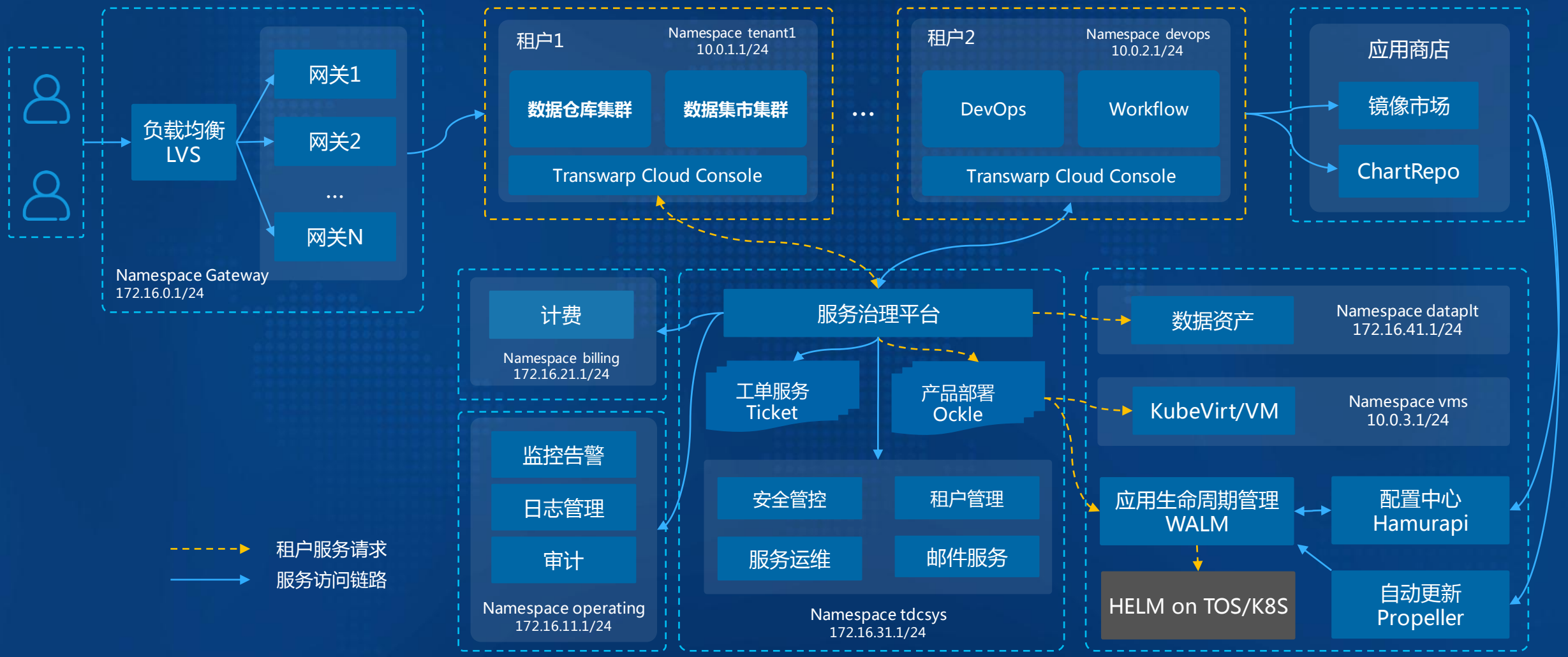
- 模板约定
  - 每个应用
    - 通过transwarp.app=name区分
  - 每个应用实例
    - 通过transwarp.install=id区分
    - 不同模块通过transwarp.name区分
    - 都有一个dummy service实现服务暴露，且transwarp.meta=true
    - 在annotation中transwarp.meta=“json string”，记录实例需要暴露的信息

大数据应用编排 2.0 (cond.)

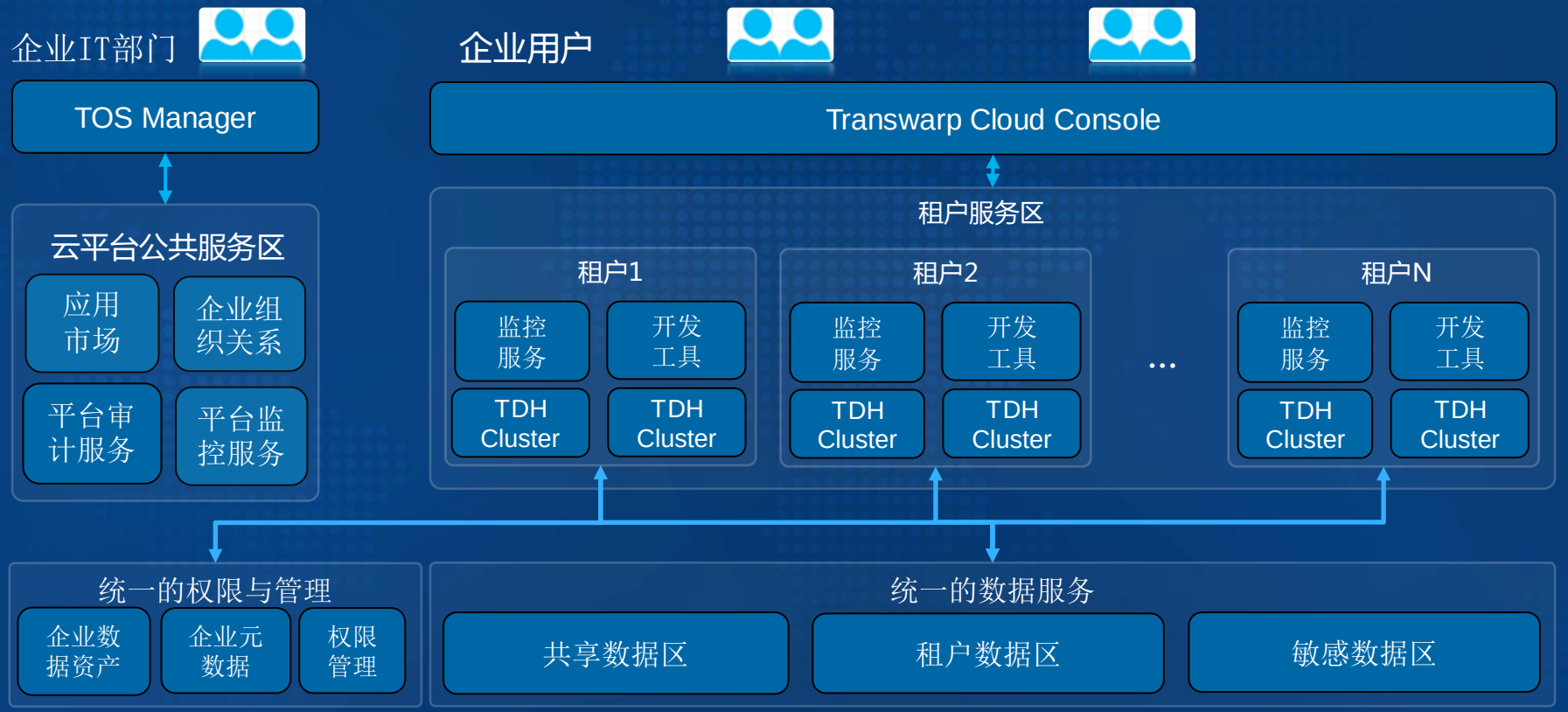
- 面临的挑战
  - 调试麻烦：jsonnet -> svc/pod/deploy
  - 语法复杂，普通开发人员学习曲线陡
    - 继承语法比较复杂
    - 复杂的if-else
  - 应用集中管理，无法独立分发
  - 社区不兼容

数据云架构 3.0 – Transwarp Data Cloud

TDC拓扑结构

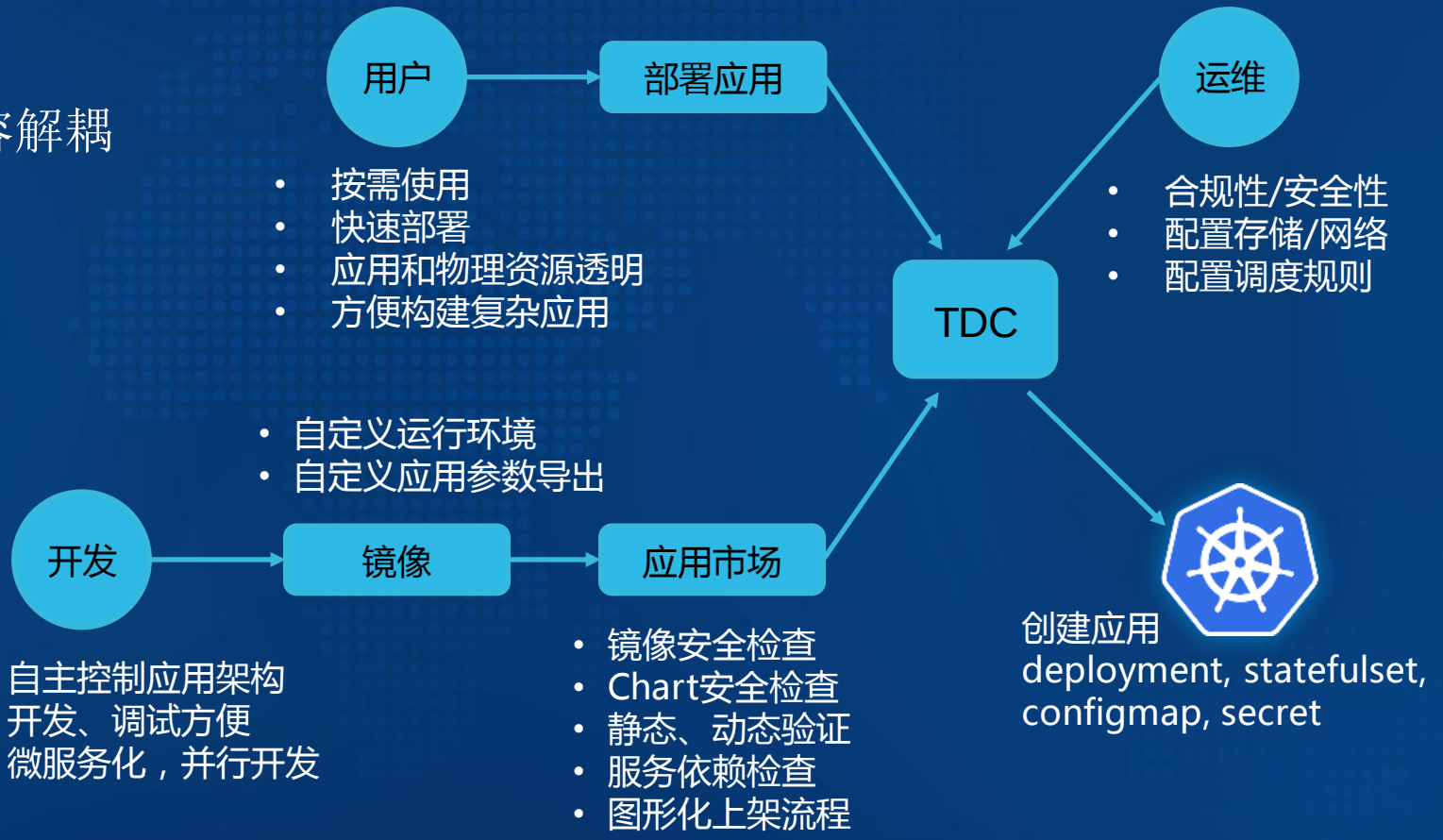

负载均衡
LVS
网关1
网关2
...
网关N
Namespace Gateway
172.16.0.1/24
租户1
Namespace tenant1
10.0.1.1/24
数据仓库集群
数据集市集群
Transwarp Cloud Console
...
租户2
Namespace devops
10.0.2.1/24
DevOps
Workflow
Transwarp Cloud Console
应用商店
镜像市场
ChartRepo
计费
Namespace billing
172.16.21.1/24
服务治理平台
工单服务
Ticket
产品部署
Ockle
监控告警
日志管理
审计
Namespace operating
172.16.11.1/24
安全管控
租户管理
服务运维
邮件服务
Namespace tdcsys
172.16.31.1/24
数据资产
Namespace dataplt
172.16.41.1/24
KubeVirt/VM
Namespace vms
10.0.3.1/24
应用生命周期管理
WALM
配置中心
Hamurapi
HELM on TOS/K8S
自动更新
Propeller
租户服务请求
服务访问链路

TDC多租户实践


企业IT部门
企业用户
TOS Manager
Transwarp Cloud Console
云平台公共服务区
应用市场
企业组织关系
平台审计服务
平台监控服务
租户服务区
租户1
监控服务
开发工具
TDH Cluster
TDH Cluster
租户2
监控服务
开发工具
TDH Cluster
TDH Cluster
...
租户N
监控服务
开发工具
TDH Cluster
TDH Cluster
统一的权限与管理
企业数据资产
企业元数据
权限管理
统一的数据服务
共享数据区
租户数据区
敏感数据区

大数据应用编排 3.0

- 设计思路
  - 开发、使用、运维工作内容解耦
  - 完善的DevOps流程
  - 应用包管理
  - 提高社区兼容性
  - 完善的服务动态依赖



大数据应用编排 3.0 (cond.)

- 社区版Helm
  - K8S包管理服务，类似于CentOS的yum
  - Chart：应用配置打包格式，类似于rpm
  - Templates: k8s manifest配置模板
  - Release: chart部署实例
  - Repository：chart仓库

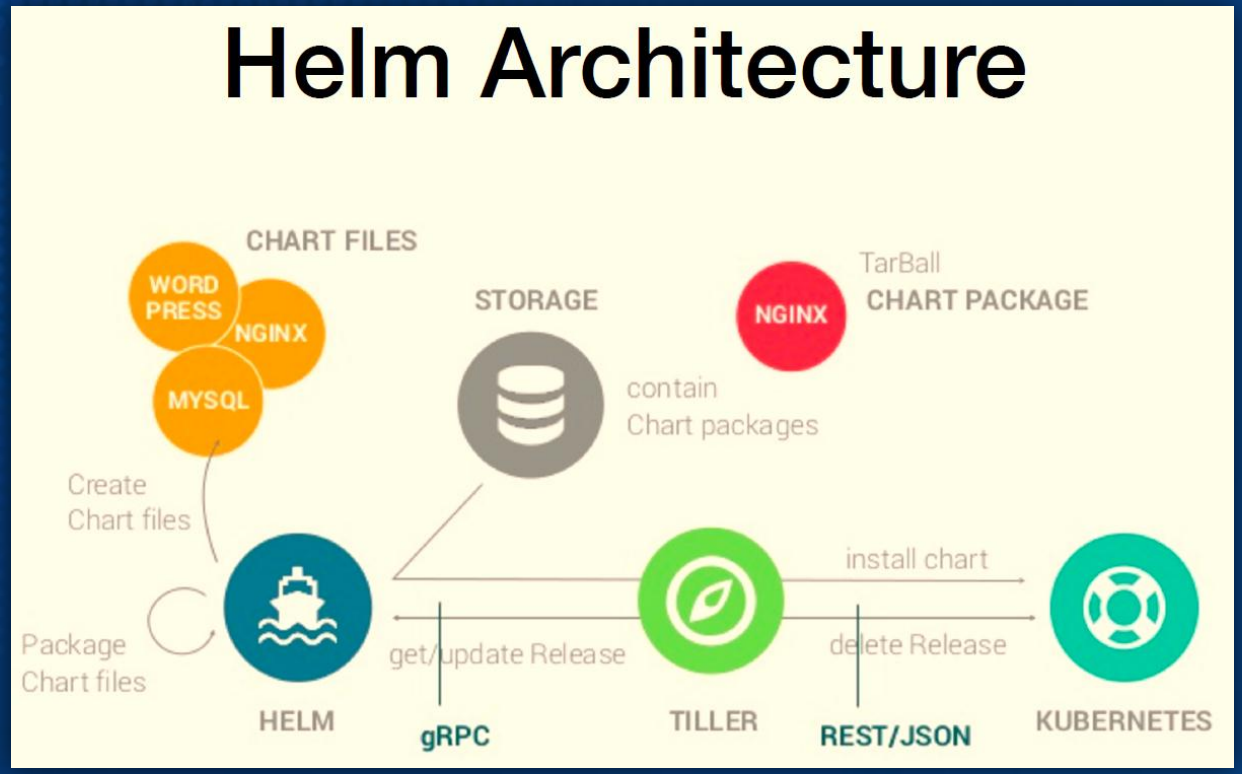
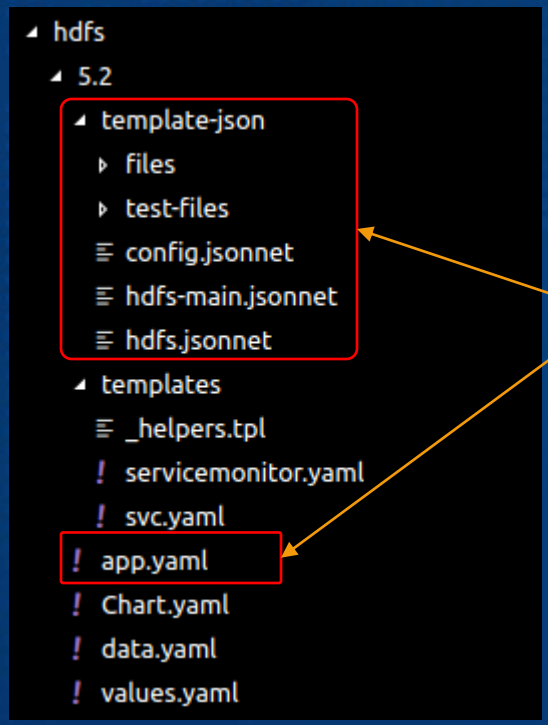


Image source: https://www.slideshare.net/alexLM/helm-application-deployment-management-for-kubernetes

大数据应用编排 3.0 (cond.)

- TDC Helm+
  - 完全兼容社区Helm Apps
  - 多渲染引擎支持
    - 社区go-template/Ksonnet
    - 星环Jsonnet
  - 通过参数控制扩容/缩容
  - 支持服务动态依赖，自定义导出/引用变量
  - 多种依赖模式
    - 依赖现有应用
    - 重新创建
    - 依赖变更
    - 一对多/多对一的依赖

第三代Chart

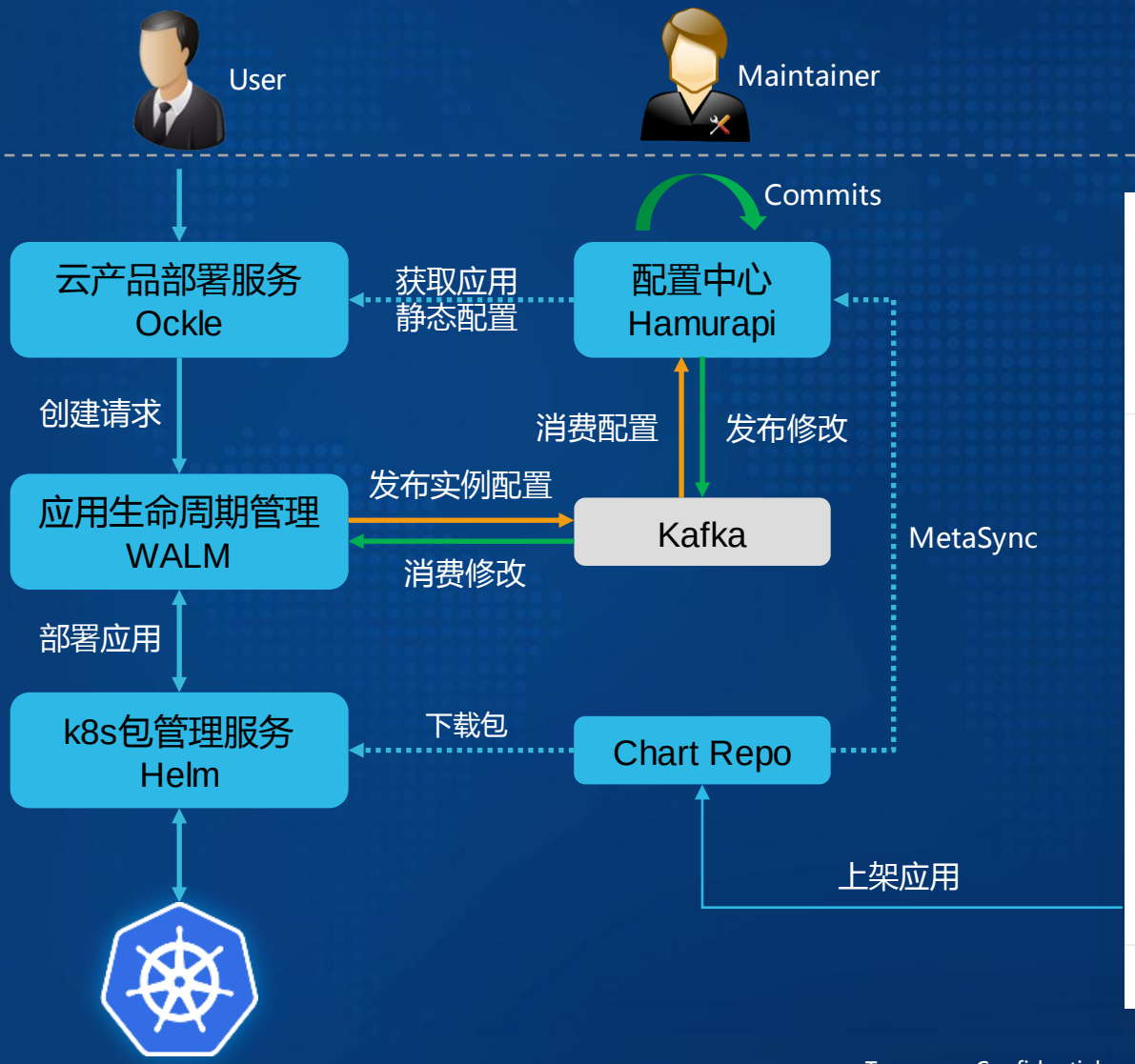


第二代Jsonnet

```
customize/
└── 1.0
    ├── app.yaml
    └── templates-json
        ├── config.jsonnet
        ├── customize-main.jsonnet
        ├── customize-pd-rc.jsonnet
        ├── customize-rc.jsonnet
        ├── customize-svc.jsonnet
        └── test-files
            ├── config0.jsonnet
            ├── config0.jsonnet.golden
            ├── config1.jsonnet
            ├── config1.jsonnet.golden
            ├── config2.jsonnet
            ├── config2.jsonnet.golden
            ├── config3.jsonnet
            └── config3.jsonnet.golden
```

https://helm.sh/

大数据应用编排 3.0 (cond.)



We Are Hiring

感谢您的聆听

www.transwarp.io