2023K+
全球软件研发行业创新峰会

智数未来 连接共生

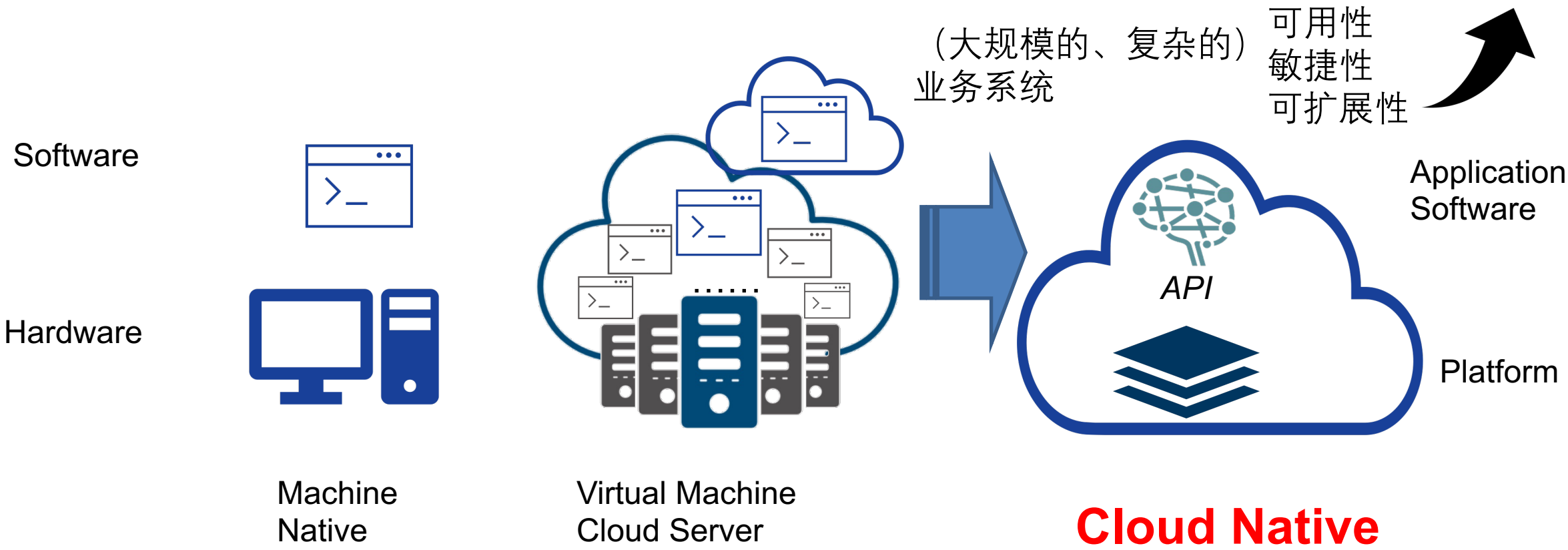
行业应用迁移到云原生环境的挑战、 路径与实践

主讲人: 李明宇
microwise 二十年编程老师傅

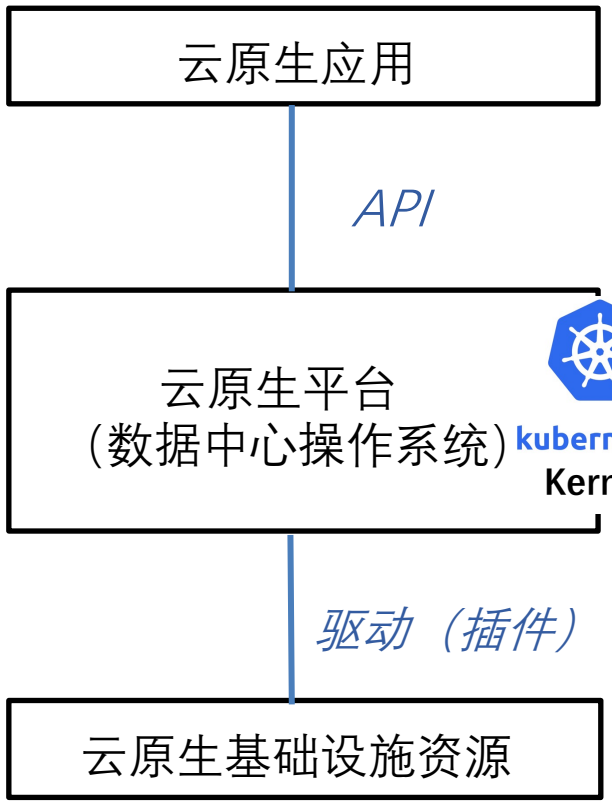
CONTENT
目录

云原生对于行业数字化的意义及挑战

数字化转型导致业务系统的复杂度和规模迅速增长，分布式应用和多云/混合云架构成为主流。
业务对IT系统的依赖达到了空前的高度，对其可用性、敏捷性和可扩展性提出了新的要求。



云原生对于行业数字化的意义及挑战

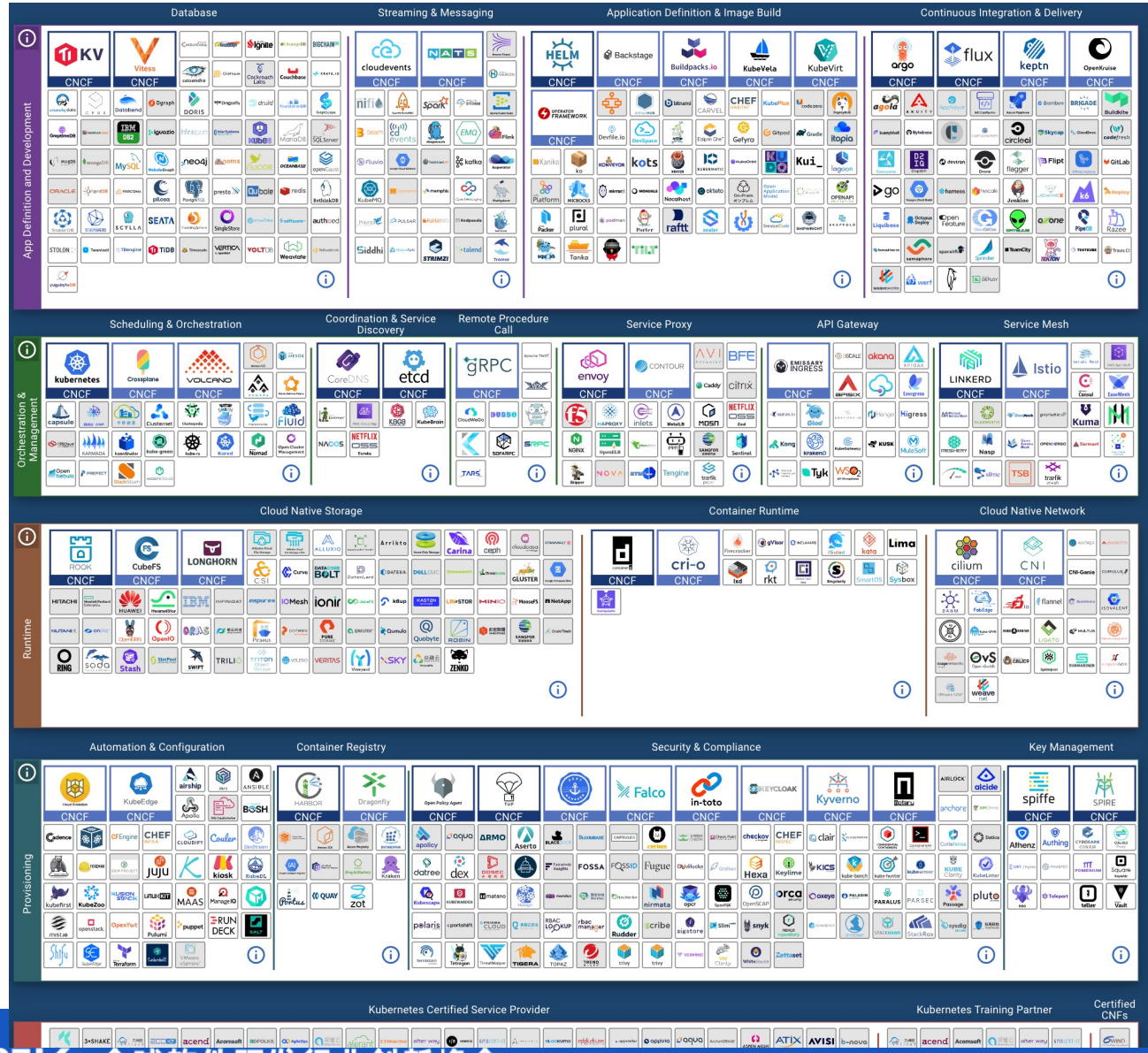


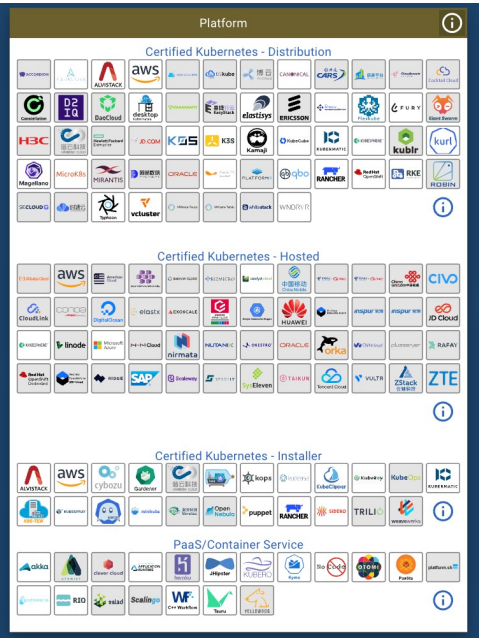
- 分布式应用（微服务、AI、HPC …）
- 将数据中心（而不再是机器）视为运行环境

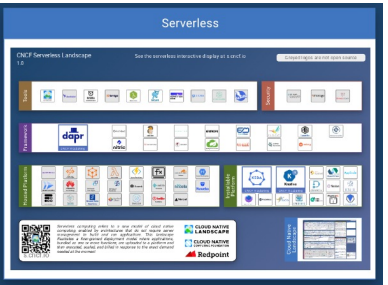
- 对上提供一组功能与接口赋能云原生应用的构建和运行
- 对下管理数据中心级的基础设施资源

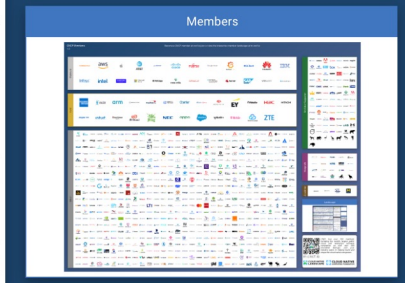
- 将计算、存储、网络等基础设施资源对接到云原生平台供给支撑应用运行

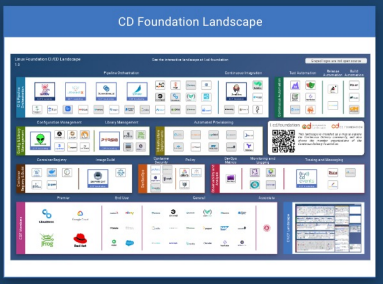
云原生对于行业数字化的意义及挑战











行业应用迁移到云原生环境的技术路径

1. 容器化

2. Pod设计
- sidecar
- initContainers

3. 工作负载设计：Deployment、Service、StatefulSet、Headless Service

4. 逐步迁移至 K8s，保持部署在 K8s 上的服务实例和部署在机器上的服务实例能够相互发现、相互调用

行业应用迁移到云原生环境实践分享

容器化

```
FROM centos:7
RUN yum -y update && yum -y install httpd php
COPY ./src/ /var/www/html/
EXPOSE 80
CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]
```

```
FROM php:8.2.7-apache
COPY ./src/ /var/www/html/
```

对比:

体积/大小

分层复用

安全性

功能单一原则

行业应用迁移到云原生环境实践分享

容器化

```
FROM centos:7
RU
CC
EX
CM
```

TAG

7

Last pushed **7 months ago** by doijanky

DIGEST	OS/ARCH	VULNERABILITIES
8faead07bd1d	linux/386	1 C 11 H 29 M 11 L
dead07b4d8ed	linux/amd64	1 C 11 H 29 M 11 L
6887440ab977	linux/arm/v7	2 C 44 H 128 M 48 L

+2 more

```
FROM php:8.2.7-apache
CC
```

TAG

8.2.7-apache

Last pushed **5 days ago** by doijanky

DIGEST	OS/ARCH	VULNERABILITIES
613e4f01cf20	linux/386	0 H 0 M 40 L
53c7a3ee1cea	linux/amd64	0 H 0 M 40 L
0e9fb3b3e414	linux/arm/v5	0 H 0 M 40 L

+5 more...

对比:

体积/大小

分层复用

安全性

功能单一原则

行业应用迁移到云原生环境实践分享

容器化

分阶段构建

```
FROM maven:3.8.2-jdk-11 AS build-env
ADD src /app/src
ADD pom.xml /app/
WORKDIR /app
RUN mvn package

FROM gcr.io/distroless/java:11
COPY --from=build-env /app/target/mymicroservice.jar /app/
WORKDIR /app
EXPOSE 8080
CMD [" mymicroservice.jar"]
```

行业应用迁移到云原生环境实践分享

容器化

分阶段构建
将第三方（可能是外协开发团队或者定制开发供应商）环境引入 DevOps (CI/CD) 流水线

```
FROM limingyu007/rust-with-config:1.54.0 as builder
WORKDIR /root/guessing_game
COPY . .
#RUN su -c 'cargo -vv b' root
RUN cargo -vv install --path .

FROM limingyu007/gcr.distroless.cc:20210814
COPY --from=builder /usr/local/cargo/bin/guessing_game /
ENV MAX_NUM=100 MAX_TRIES=10

CMD ["/guessing_game"]
```

行业应用迁移到云原生环境实践分享

容器化

分阶段构建

```
FROM ubuntu:20.04 as builder

# switch to root user to install packages

USER root

ENV DEBIAN_FRONTEND=noninteractive

RUN apt update && apt install -y \

    zsh git\

    build-essential csh gfortran m4 libcurl4-gnutls-dev zli

    mpich libmpich-dev


… …


# 2-stage build

FROM ubuntu:20.04

… …

COPY --from=builder /WRF/main/wrf.exe /app/mpi-func

COPY --from=builder /3rdParty/libs/ /lib/
```

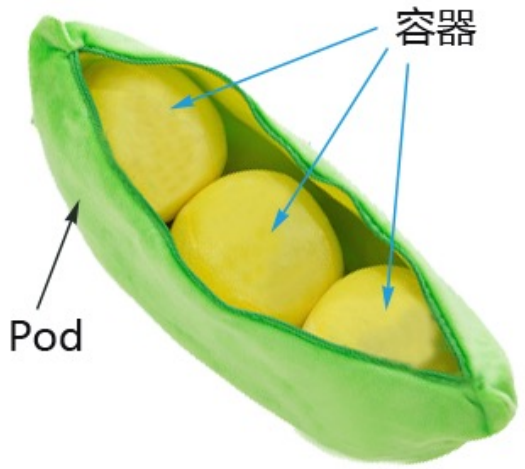
C++ 应用的容器化 —— 动态链接库的处理

```
# Function to recursively copy dependencies
copy_dependencies() {
    local binary_path=$1
    local dest_folder=$2
    local temp_file=$3

    echo "binary_path: ${binary_path}"
    local ldd_output
    ldd_output="$(ldd "${binary_path}" | grep '=>' | awk '{print $3}' | grep -v 'not found' || true)"

    if [ -z "${ldd_output}" ]; then
        return
    fi

    echo "${ldd_output}" | while read -r lib_path; do
        # Check if the library has already been copied
        echo "lib_path: ${lib_path}"
        if ! grep -q "^${lib_path}$" "${temp_file}"; then
            echo "${lib_path}" >> "${temp_file}"
            echo "Copying ${lib_path} to ${dest_folder} ..."
            cp "${lib_path}" "${dest_folder}"

            # Recursively copy the dependencies of the library
            copy_dependencies "${lib_path}" "${dest_folder}" "${temp_file}"
        fi
    done
}
```

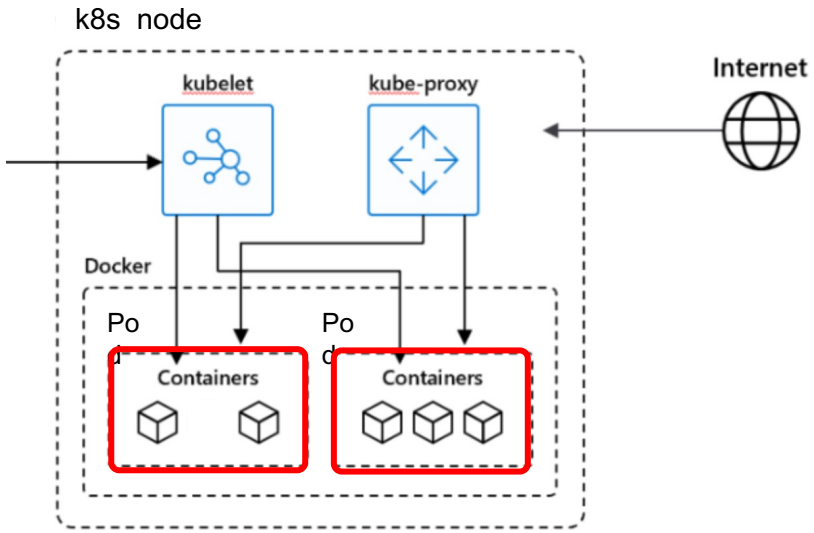

作者：李明宇　　微信：microwise

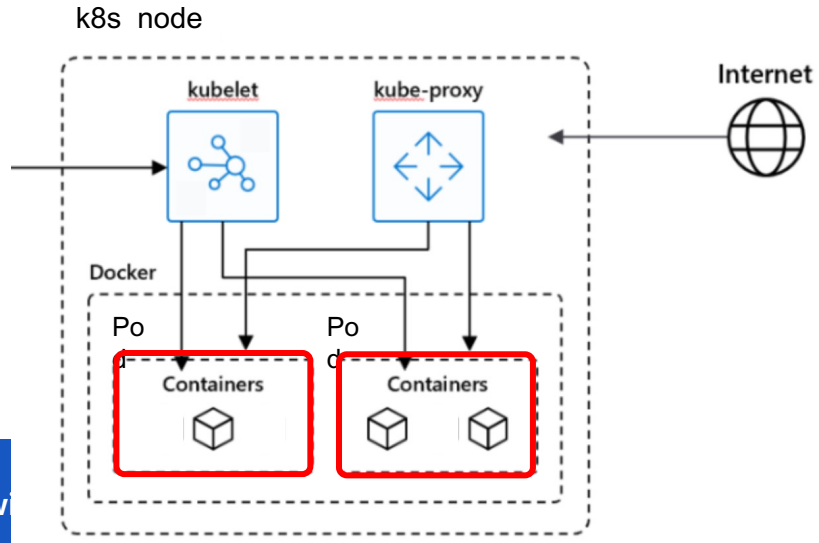
行业应用迁移到云原生环境实践分享

Pod Design



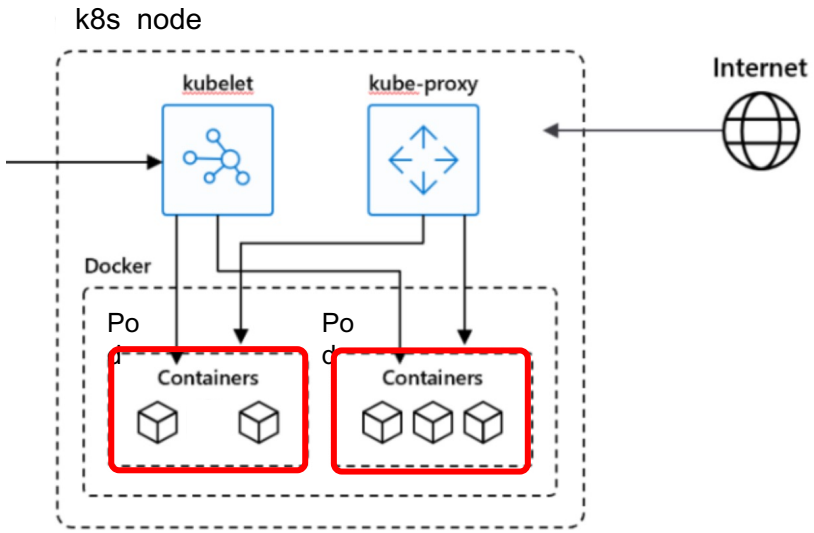
- 一个Pod可以由一个或多个容器组成
- 一个Pod中容器共享网络命名空间，在同一台机器上运行

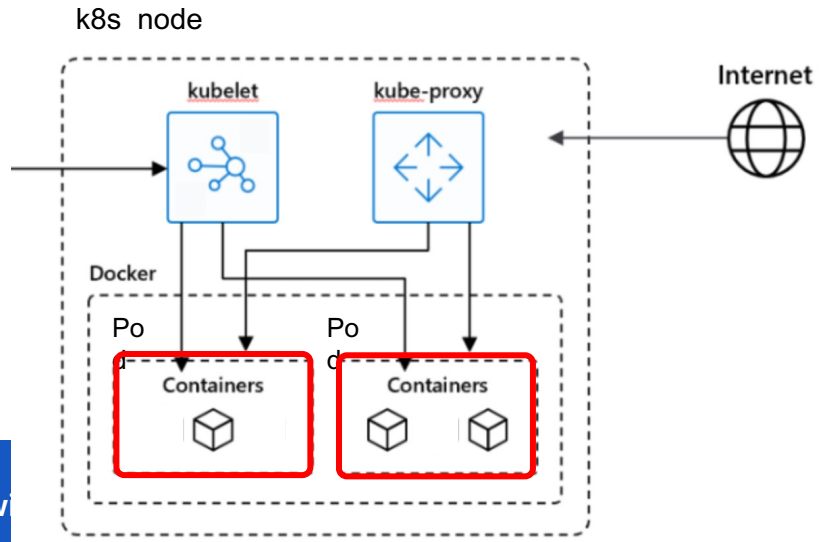




行业应用迁移到云原生环境实践分享

Pod Design

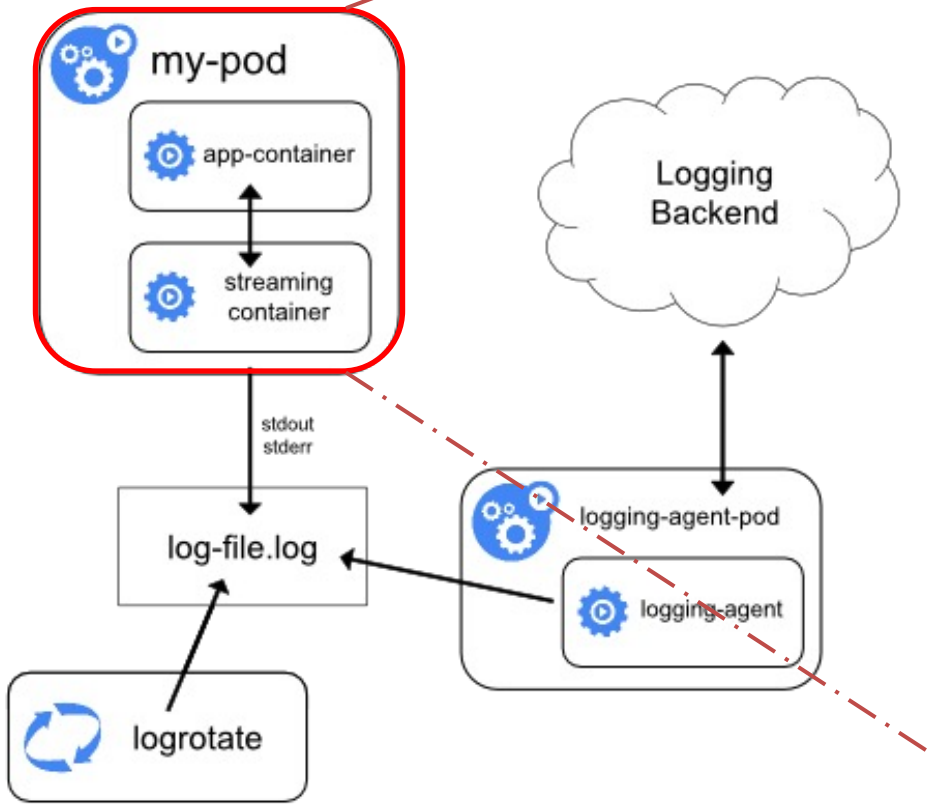




行业应用迁移到云原生环境实践分享

Pod Design

不改代码，不跑agent，
把本地文件日志转成
云原生标准输出日志




```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - name: main-app
    image: my-application
    volumeMounts:
    - name: log-volume
      mountPath: /var/log
  - name: sidecar-log-handler
    image: busybox:1.28
    args: [/bin/sh, -c, 'tail -n+1 -f /var/log/my.log']
    volumeMounts:
    - name: log-volume
      mountPath: /var/log
  volumes:
  - name: log-volume
    emptyDir: {}
```

行业应用迁移到云原生环境实践分享

Pod Design

initContainers

- **准备容器**

为主容器（或其他容器）的运行准备数据/环境

例：视频转码

- **延时容器**

延迟后续容器的启动，直到外部依赖已就绪

例：卫星数据接收

```
apiVersion: v1
kind: Pod
metadata:
  name: video-transcoding-pod
spec:
  initContainers:
  - name: download-video
    image: video-downloader:latest
    volumeMounts:
    - name: video-volume
      mountPath: /videos
  containers:
  - name: transcode-video
    image: video-transcoder:latest
    volumeMounts:
    - name: video-volume
      mountPath: /videos
  volumes:
  - name: video-volume
    emptyDir: {}
```

行业应用迁移到云原生环境实践分享

在K8s上构建和运行生产级服务

故障检查与处理

- **livenessProbe**

如果检查失败，Kubernetes**将杀死容器，默认设置下，该容器将重启**

- **readinessProbe**

如果检查失败，Kubernetes会**把Pod从服务代理的分发后端剔除**

- **startupProbe**

每种Probe主要支持以下三种检查方法：

- **httpGet**

发送HTTP请求，返回200-400范围状态码为成功。

- **exec**

执行Shell命令返回状态码是0为成功。

- **tcpSocket**

发起TCP Socket建立成功。　　gRPC 检查 1.24 beta

行业应用迁移到云原生环境实践分享

在K8s上构建和运行生产级服务

```
… …
containers:
- name: hello-spring
  image: hello-spring:v0.3
  readinessProbe:
    httpGet:
      path: /actuator/health/readiness
      port: 8080
    initialDelaySeconds: 30
    timeoutSeconds: 2
    periodSeconds: 3
    failureThreshold: 1
  livenessProbe:
    httpGet:
      path: /actuator/health/liveness
      port: 8080
    initialDelaySeconds: 20
    timeoutSeconds: 2
    periodSeconds: 8
    failureThreshold: 2
```

```
… …
containers:
- name: ftp
  image: your-ftp-image
  ports:
  - containerPort: 21
  livenessProbe:
    tcpSocket:
      port: 21
    initialDelaySeconds: 5
    periodSeconds: 10
  readinessProbe:
    exec:
      command:
      - sh
      - -c
      - |
        ftp -n -v localhost 21 <<EOF
        user anonymous anonymous
        ls
        quit
        EOF
    initialDelaySeconds: 5
    periodSeconds: 5
```

行业应用迁移到云原生环境实践分享

在K8s上构建和运行生产级微服务

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: lifecyclehook-demo-php
spec:
  selector:
    matchLabels:
      app: lifecyclehook-demo-php
  replicas: 3
  template:
    metadata:
      labels:
        app: lifecyclehook-demo-php
    spec:
      containers:
      - name: my-php
        image: limingyu007/phpsleepdemo:v2
        lifecycle:
          preStop:
            exec:
              command: ["sh", "-c", "sleep 3"]
```

优雅下线 —— 长连接、慢请求

例子：报表生成

```
apiVersion: v1
kind: Pod
metadata:
  name: report-generator
spec:
  containers:
  - name: report-generator
    image: report-generator-image
    ports:
    - containerPort: 8080
  terminationGracePeriodSeconds: 120
```

行业应用迁移到云原生环境实践分享

在K8s上构建和运行生产级服务

statfulset和headless service 除了数据库和Kafka，还可以用在哪？

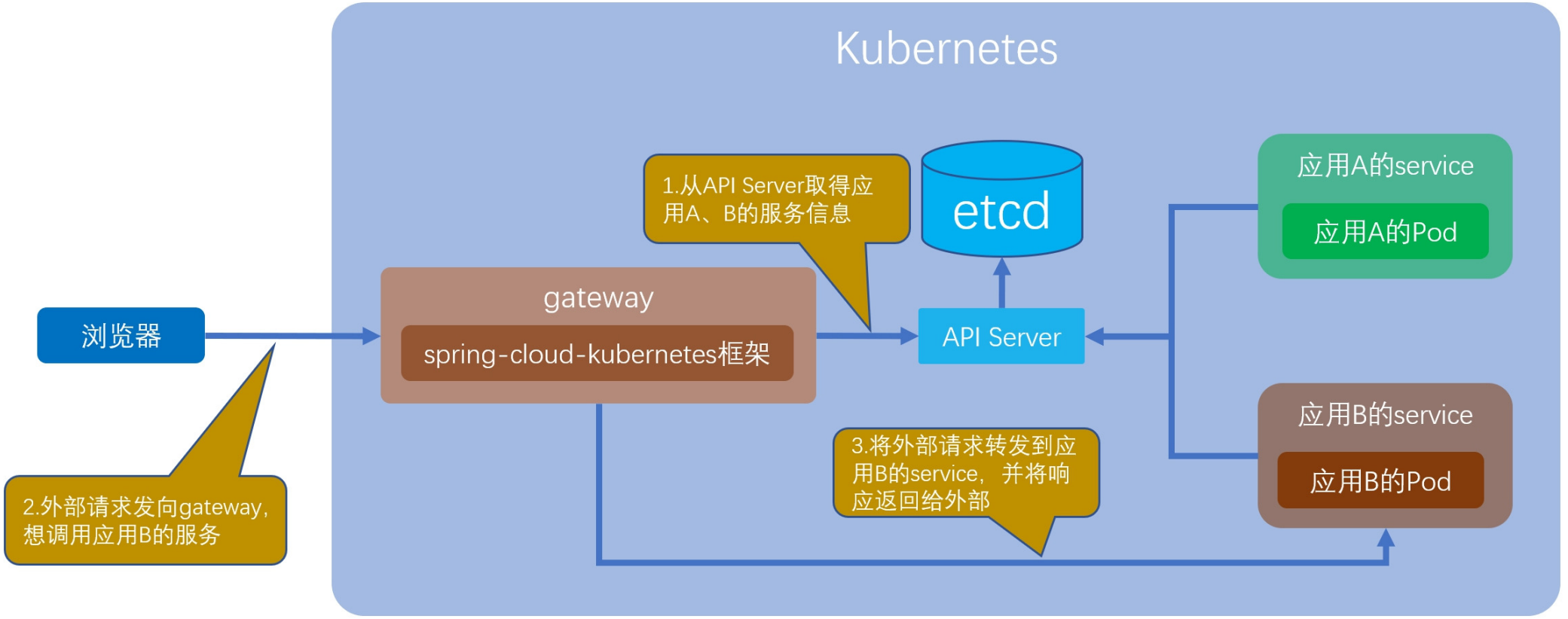
1. 同一份代码要运行多个实例，每个服务实例要持久化/暂存/加载自己的数据；
2. 每个实例要独立寻址（不能一个IP负载均衡，更不能roundrobin）；
3. 自动检测故障并恢复，恢复后数据不丢失；
4. 配置统一管理，每个实例可能在加载配置时根据具体情况进行一些修改，例如hostname

符合以上一个或多个条件，可以选择用 statfulset + headless service

例：IoT或者工业控制领域：轨道交通、智能制造 ……

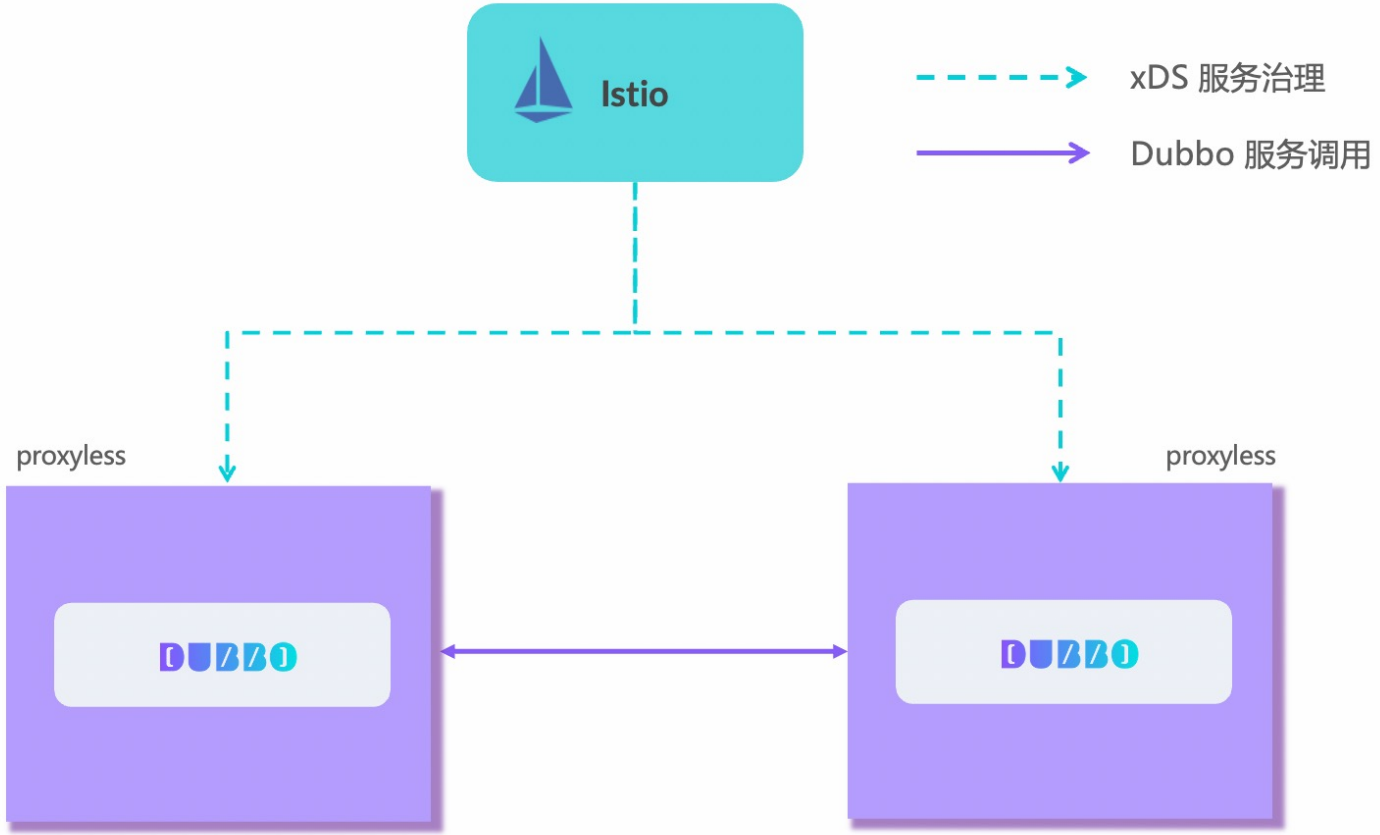
行业应用迁移到云原生环境实践分享

基于 Spring Cloud 和 dubbo 开发的应用迁移到 K8s 和 Istio 环境



行业应用迁移到云原生环境实践分享

基于 Spring Cloud 和 dubbo 开发的应用迁移到 K8s 和 Istio 环境



行业应用迁移到云原生环境实践分享

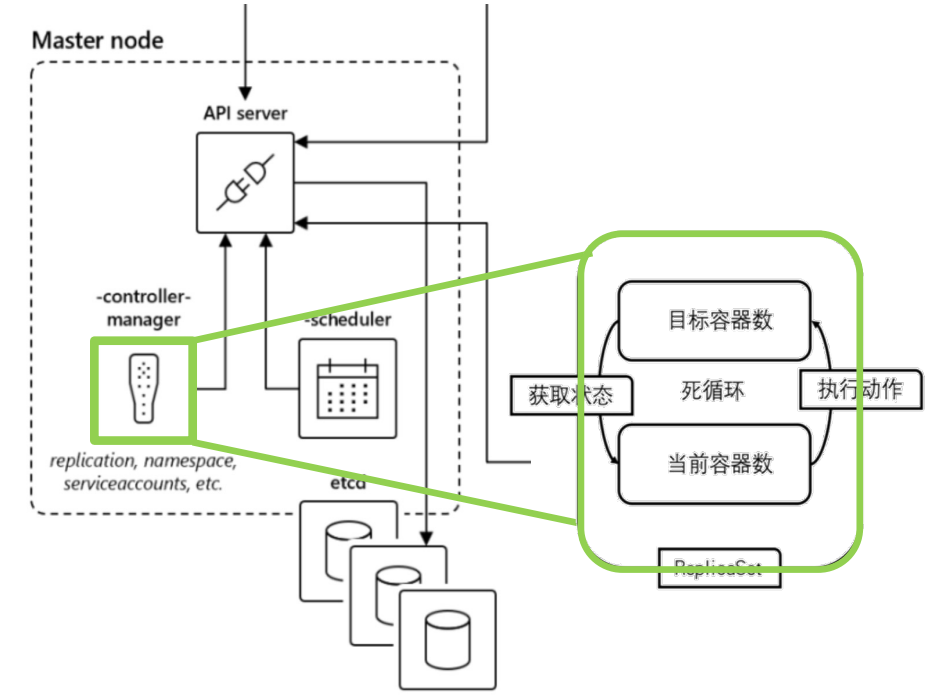
跑批、离线任务、一次性任务、周期任务 ……

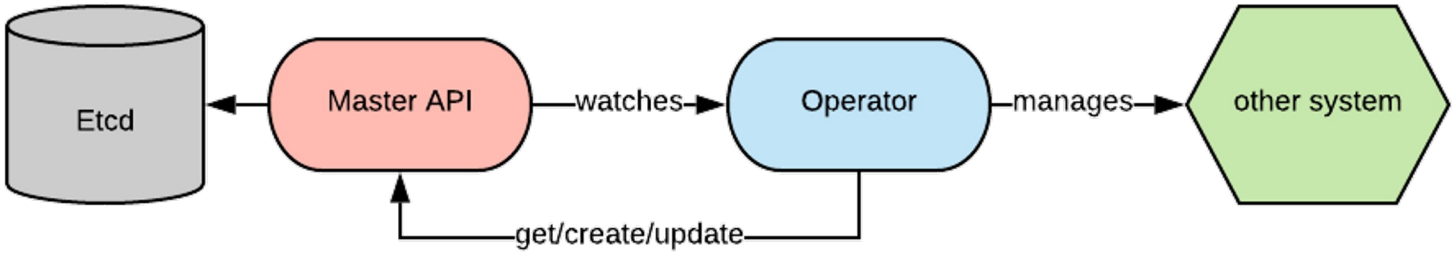
```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: grape-mpi-workers
spec:
  completions: 4
  parallelism: 4
  completionMode: Indexed
  template:
    metadata:
      labels:
        app: grape-mpi
        type: workers
    spec:
      containers:
      - name: grape-mpi
        image: limingyu007/run_app:v1.1
      restartPolicy: Never
```

```yaml
apiVersion: "kubeflow.org/v1alpha2"
kind: "TFJob"
metadata:
  name: "dist-mnist-pct"
spec:
  tfReplicaSpecs:
    PS:
      replicas: 1
      restartPolicy: Never
      template:
        spec:
          containers:
            - name: tensorflow
              image: emsixteeen/tf-dist-mnist
    Worker:
      replicas: 2
      restartPolicy: Never
      template:
        spec:
          containers:
            - name: tensorflow
              image: emsixteeen/ tf-dist-mnist
```

行业应用迁移到云原生环境实践分享

CRD与Operator，例子：AI训练任务、分布式视频转码

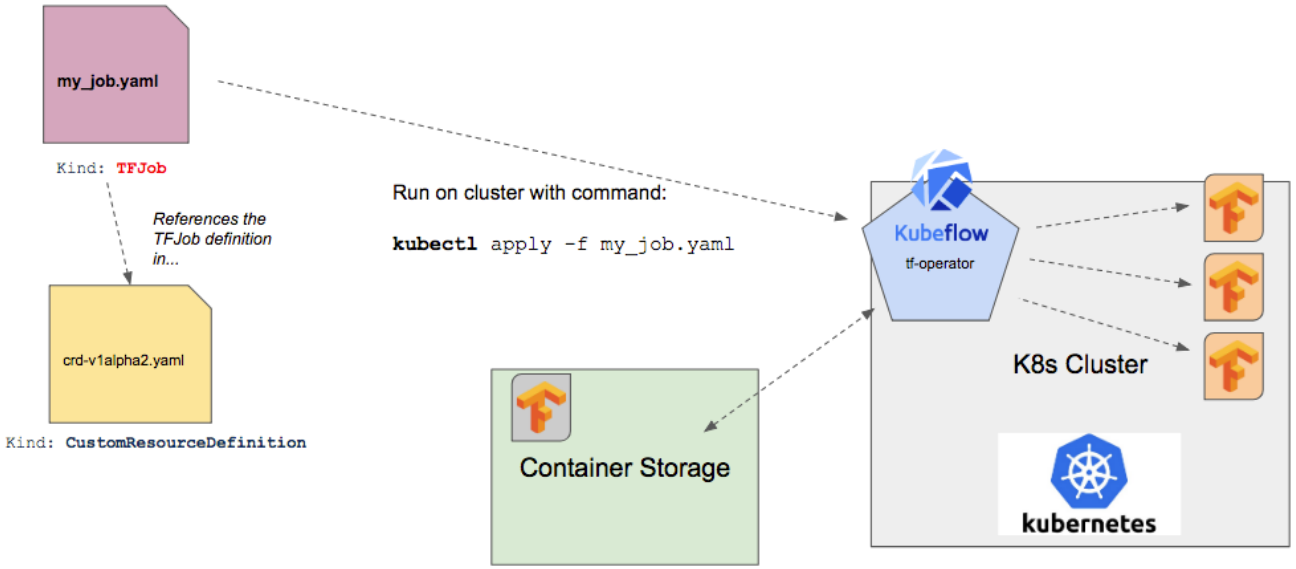





```
# nginx-deployment.yaml

apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
    containers:
      - name: nginx
        image: nginx:1.10
        ports:
        - containerPort: 80
```

行业应用迁移到云原生环境实践分享

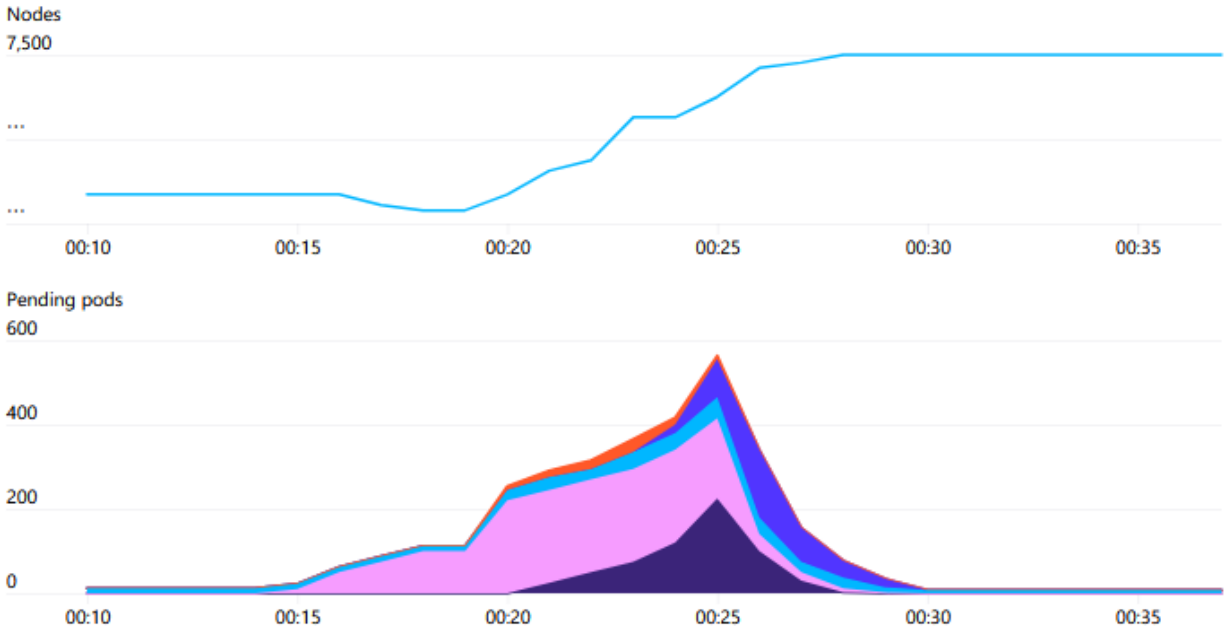
CRD与Operator，例子：AI训练任务、分布式视频转码




```yaml
apiVersion: "kubeflow.org/v1alpha2"
kind: "TFJob"
metadata:
  name: "dist-mnist-pct"
spec:
  tfReplicaSpecs:
    PS:
      replicas: 1
      restartPolicy: Never
      template:
        spec:
          containers:
            - name: tensorflow
              image: emsixteeen/tf-dist-mnist
    Worker:
      replicas: 2
      restartPolicy: Never
      template:
        spec:
          containers:
            - name: tensorflow
              image: emsixteeen/ tf-dist-mnist
```

大模型与AIGC带来新挑战与方案

OpenAI：《我们将 Kubernetes 扩展到了 7500 个节点》



- GPU：NVLink / GPUDirect
- 计算密集：单个Pod占用整个节点
- 对象存储（Blob）使用多于文件存储
- 网络：VPC网络插件
- 5 个 API 服务器和 5 个 etcd 节点
- 节点级健康检查
- Gang Scheduling

大模型与AIGC带来新挑战与方案

	AI内容生成服务	VS	传统微服务&Web服务
资源调度：	GPU Sharing （训练时多卡分布式、推理时GPU共享）		CPU + 内存
执行时间：	十秒到分钟级， 根据用户请求参数有所不同		百毫秒级、秒级， 绝大多数用户请求响应时间一致
可扩展性：	Session保持、优雅下线		无状态服务，相对简单
更新迭代：	模型更新		更新迭代：新功能、bug fix

THANKS!

2023K+
全球软件研发行业创新峰会