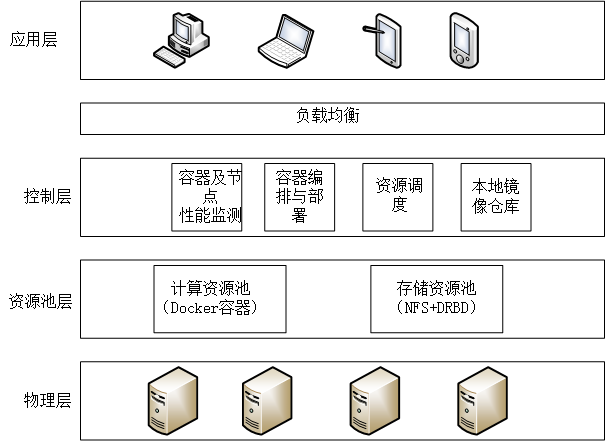
# 基于Docker集群弹性部署.NetCore应用

本开源项目的主要目的是结合时下流行的Docker容器技术，利用Nginx、Consul、Kubernetes开源工具为.NET Core应用程序构建分布式容器集群、实现自动部署，弹性伸缩，资源调度和负载均衡提供整套解决方法的实例。

## 总体部署的架构图

.NET Core应用程序运行在docker容器内，整个系统的部署架构如下图一：

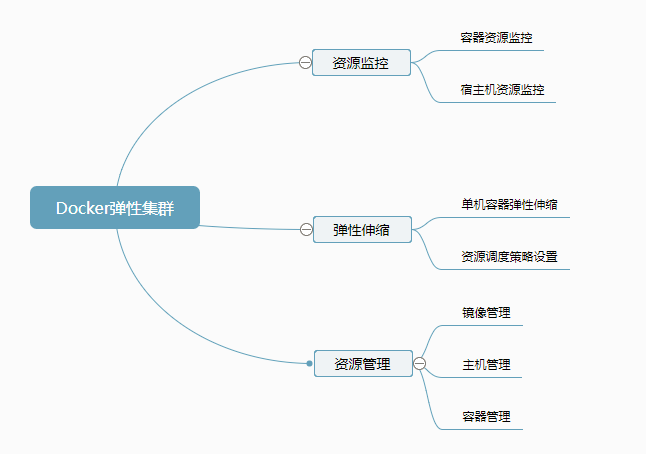


图一：部署架构

物理层：物理层主要是一些基本的硬件资源，为系统提供最基础的服务。

资源池层：资源池层是在物理资源基础上基于Docker 容器虚拟化技术实现，存储资源池由 DRBD 结合 NFS 构建多节点网络文件系统实现共享存储。

控制层：控制层是系统的核心部分，包括容器及节点性能监控模块，提供直观的性能监控，并且存储的数据可以与设定阈值进行比较作为容器和节点弹性伸缩的依据。本地镜像仓库可以实现容器集群镜像的共享，保证节点容器构建出的环境的一致性。在核心模块中，容器编排与部署可以实现单主机节点的容器的弹性伸缩，而资源调度主要是通过一定的策略选择合适的主机节点进行容器弹性伸缩实现，从而构建出基于 Docker 的弹性集群。对于控制层的技术选择，选择利用Kubernetes实现管理节点的应用容器，解决跨节点的通讯问题。对于Kubernetes所要实现的功能如图二。

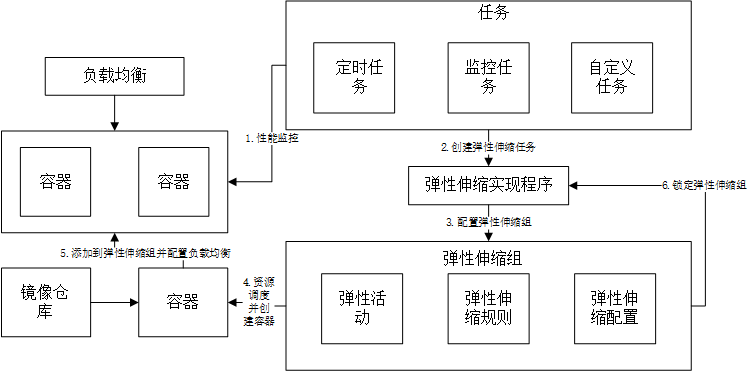


图二：Kubernets实现的主要功能

负载均衡部分主要是保证弹性扩展后的容器与原容器构成的容器组能有效稳定的对外提供服务。选择利用Nginx作为客户端的统一入口，并进行服务的转发，以实现负载均衡。

应用层：应用层主要是针对于用户而言，整个实现对于用户而言是直观的、透明的，包括所有的终端访问。

在对整个系统进行架构分层之后，得出系统的运行流程图如图三。



图三：系统部署流程图

（1）任务：任务的执行是为了保障弹性伸缩机制的触发，包括定时任务，即设定了某

个时间段进行弹性伸缩（如晚上七点到十二点增加业务容器数量）；监控任务，即根据设定的容器负载值进行弹性伸缩（如当 CPU 利用率超过 85%时执行扩展任务）；自定义任务，即用户自己设定的触发机制。

（2）弹性伸缩组：由提供相同服务的容器共同构成的集合。包括多个运行实际业务的

容器、Nginx 负载均衡容器实例、Consul 服务发现容器实例。

（3）弹性伸缩活动：指响应触发机制而产生的活动，如容器数量的扩展或者收缩。

（4）弹性伸缩规则：指具体对伸缩活动的定义，如在进行容器数量扩展时应该扩展的

数量。

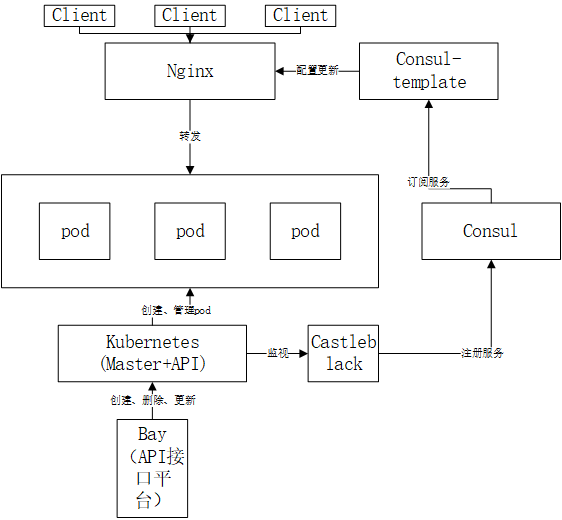
（5）弹性伸缩配置：定义了弹性伸缩组所使用到的运行实际业务的容器 ID、Nginx实例ID和Consul 实例 ID 等

（6）锁定弹性伸缩组：指在执行了弹性伸缩活动后的一段锁定时间。该时间段内弹性

伸缩组不执行其他活动，防止之前的活动还未完成而后续活动的到来。

## 二、.NetCore弹性部署技术选则

根据系统部署的流程选择相应的技术框架，使用已经成熟的技术组件组合在一起，使业务可以提供对外服务，如图四是针对部署.NetCore应用程序的docker平台组件技术选择。



图四：技术组件选择

以上的主要工作流程是：通过Bay平台向 Kubernetes APIServer发送请求，创建 deployment，pod 创建成功并且健康检查通过后，Castle Black watch 到 pod 信息，将 IP，port 等信息注册到 Consul 上，consul cluster会将新的服务信息推送给已经到它这里订阅了服务消息的consul-template，consul-template再去修改和自己同一台机器上的nginx，以达到动态调整负载均衡的目的。

## 三、监控与日志收集

对于基于Docker的部署的监控与日志收集采用比较成熟的方案，监控指标收集主要采用CAvisor，对于日志收集系统采用Logspout Kafka ES/HDFS， Logspout一个主要用于收集容器日志的开源软件logspout 会自动发现这个容器并提日志，将该容器的日志发送到 Kafka 上，日志打到 Kafka 里之后，会有相应的 consumer 消费日志，落地 ES 和 HDFS。ES 主要用来作日志查询，HDFS 主要用来做日志备份，整个系统的日志收集流程如图五。

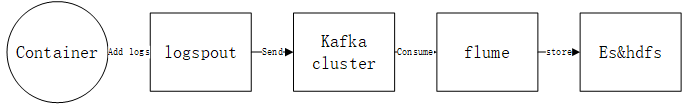
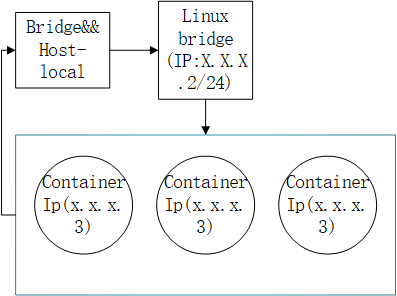


图5：日志保存流程

## 四、网络配置

网络配置采用了 Bridge 和 host-local 这两个 CNI 插件，Bridge 主要用来挂载/卸载容器的 veth pair 到 Linux Bridge 上，host-local 主要利用本地的配置来给容器分配 IP，具体流程如图六。

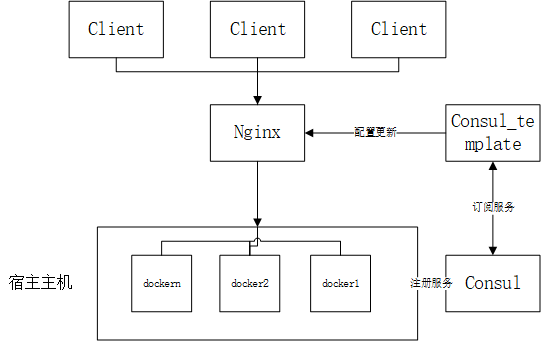


图六：网络配置流程图

## 五、项目阶段划分

### 第一阶段

实现单台宿主主机多个docker结合consul和nginx到达自动实现负载均衡的目标如图7.



### 第二阶段

在第一阶段基础上实现一个数据中心，至少三台宿主主机.NetCore应用程序的弹性部署，如图8

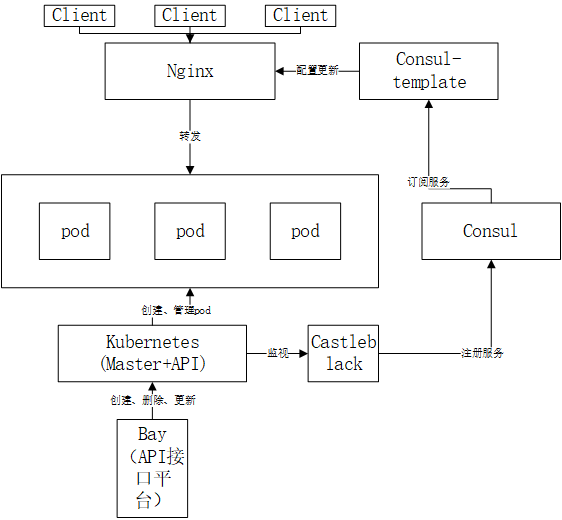


图8

### 第三阶段

在第二个阶段基础上实现，实现部署系统日志收集和数据库读写一致。

### 第四阶段

在实现第三阶段的基础上实现，多个数据中心