我们需要的是一种可以一次性将变更传递给所有受控容器的方法，

同时也需要一种可以轻松地调度可用容器的方法，这个过程还必须要是自动化的，这正是 Kubernetes 所做的事情

例如，你想让你的 Web 服务器始终运行在 4 个容器中，以达到负载均衡的目的，你的数据库复制到 3 个不同的容器中，以达到冗余的目的。这就是你想要的状态。如果这 7 个容器中的任何一个出现故障，Kubernetes 引擎会检测到这一点，并自动创建出一个新的容器，以确保维持所需的状态。

当你第一次设置 Kubernetes 时，你会创建一个集群。所有其他组件都是集群的一部分。你也可以创建多个虚拟集群，称为命名空间 (namespace)，它们是同一个物理集群的一部分。这与你可以在同一物理服务器上创建多个虚拟机的方式非常相似。如果你不需要，也没有明确定义的命名空间，那么你的集群将在始终存在的默认命名空间中创建。

Kubernetes 运行在节点 (node) 上，节点是集群中的单个机器。如果你有自己的硬件，节点可能对应于物理机器，但更可能对应于在云中运行的虚拟机。节点是部署你的应用或服务的地方，是 Kubernetes 工作的地方。有 2 种类型的节点 ——master 节点和 worker 节点，所以说 Kubernetes 是主从结构的。

主节点是一个控制其他所有节点的特殊节点。一方面，它和集群中的任何其他节点一样，这意味着它只是另一台机器或虚拟机。另一方面，它运行着控制集群其他部分的软件。它向集群中的所有其他节点发送消息，将工作分配给它们，工作节点向主节点上的 API Server 汇报

Master 节点本身包含一个名为 API Server 的组件。这个 API 是节点与控制平面【control plane】通信的唯一端点。API Server 至关重要，因为这是 worker 节点和 master 节点就 pod、deployment 和所有其他 Kubernetes API 对象的状态进行通信的点。

Worker 节点是 Kubernetes 中真正干活的节点。当你在应用中部署容器或 pod（稍后定义）时，其实是在将它们部署到 worker 节点上运行。Worker 节点托管和运行一个或多个容器的资源。

Kubernetes 中的逻辑而非物理的工作单位称为 pod。一个 pod 类似于 Docker 中的容器。记得我们在前面讲到，容器可以让你创建独立、隔离的工作单元，可以独立运行。但是要创建复杂的应用程序，比如 Web 服务器，你经常需要结合多个容器，然后在一个 pod 中一起运行和管理。这就是 pod 的设计目的 —— 一个 pod 允许你把多个容器，并指定它们如何组合在一起来创建应用程序。而这也进一步明确了 Docker 和 Kubernetes 之间的关系 —— 一个 Kubernetes pod 通常包含一个或多个 Docker 容器，所有的容器都作为一个单元来管理。

Kubernetes 中的 service 是一组逻辑上的 pod。把一个 service 看成是一个 pod 的逻辑分组，它提供了一个单一的 IP 地址和 DNS 名称，你可以通过它访问服务内的所有 pod。有了服务，就可以非常容易地设置和管理负载均衡，当你需要扩展 Kubernetes pod 时，这对你有很大的帮助，我们很快就会看到。

ReplicationController 或 ReplicaSet 是 Kubernetes 的另一个关键功能。它是负责实际管理 pod 生命周期的组件 —— 当收到指令时或 pod 离线或意外停止时**启动 pod**，也会在收到指示时**杀死 pod**，也许是因为用户负载减少。所以换句话说，ReplicationController 有助于实现我们所期望的指定运行的 pod 数量的状态。

什么是kubectl

kubectl 是一个命令行工具，用于与 Kubernetes 集群和其中的 pod 通信。使用它你可以查看集群的状态，列出集群中的所有 pod，进入 pod 中执行命令等。你还可以使用 YAML 文件定义资源对象，然后使用 kubectl 将其应用到集群中。

kubectl中的自动扩展

01Kubernetes基础主键

控制面：

- kube-apiserver
- kube-controller-manager
- kube-scheduler
- ETCD
- DNS

kube-apiserver:所有组件请求它，并操作ETCD。【一线客服】

- 提供集群管理接口，认证授权
- 其他模块间通信的桥梁
- 操作ETCD

kube-controller-manager：资源对象的自动化控制中心【神经中枢】

- 资源控制中心
- Service Controller
- Endpoint Controller
- Namespace Controller
- Node Controller

controller-manager是大管家，Service Controller、工作负载的Controller等

kube-scheduler 根据调度策略为pod分配节点【调度室】

根据节点的稳定性【机器的配置等】等调度分配节点

ETCD

- 保存所有的资源对象和网络配置

DNS【地址和域名的转换】

- 服务地址与服务名称的转换

数据面：

Node

- Pod：一个或多个container。一组进程，通过回环地址、ipc进行通信。
- 静态文件

- 动态资源
- 探针

就绪探针：是否允许外部流量进入。

请求探针：是否重启pod。容器要有幂等性

控制面驻节点办公室主任:kubelet。通过rest ful 机制和api server通信

kubelet是控制面和数据面之间通信的进程\服务。初始化节点时创建的二进制文件，可以通过systemctl查询

pod管理、健康检查、资源监控

（管理pod）schedule->api server ->kubelet 创建pod

（健康检查）kubelet每隔n秒发送一个探针给container，并将结果处理 kill/重启

kubelet内置CAdvisor监控节点里面所有容器的资源【cpu、内存、文件系统、网络】使用情况，再上报给aipserver。然后根据用户设定的HPA(HorizontalPodAutoscaler)指标【CPU利用率、内存使用率或自定义指标】对象来指定自动扩缩容的规则。

HPA定义示例

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: my-hpa
  namespace: my-namespace
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: my-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

控制面驻节点办公室副主任:kube-proxy

逻辑组件

- Namespace
- Lable/Selector【资源筛选组件】
- Annotation【给人看的】

- ConfigMap/Secret【保存配置信息】

  configmap是存在etcd中，最终一致性可能会拿到旧数据。

  secret是保存敏感信息的，service account->允许访问api service