Docker+K8S+DevOps 微服务架构师

学神 IT 教育：从零基础到实战，从入门到精通！

联系方式：

学神 IT 教育官方网站: http://www.xuegod.cn

学神 K8S 精英学习 11 群 QQ 群: 957231097

学习顾问：小语老师　　学习顾问：边边老师　　学神微信公众号

微信扫码添加学习顾问微信，同时扫码关注学神公众号了解最新行业动态，获取更多学习资料及答疑就业服务！

第 11 章　轻量级 k8s 平台-边缘计算场景-k3s 入门到实战

本节所讲内容：

11.1 k3s 核心技术解读?
11.2　k3s 架构
11.3　k3s 特点
11.4　选择 k3s 的理由
11.5　云计算 vs 边缘计算
11.6 k3s 和 k8s 如何选择?
实战 1：安装 k3s 集群
实战 2: 在 k3s 集群添加 agent 节点
实战 3：在 k3s 中部署应用- Guestbook 留言板
29.7 卸载 k3s 集群

实验环境：新创建两台虚拟机
K3s server 机器配置：
IP：192.168.40.135
主机名：xuegod65
配置：2vCPU/1Gi 内存
K3s agent 机器配置：
IP：192.168.40.136
主机名：xuegod66
配置：2vCPU/1Gi 内存

注意：机器配置可以按照下面标准分配
安装使用 k3s 服务的最低系统要求：
Linux 的内核版本在 3.10 以上
每台服务器上至少要有 512MB 的内存空间
硬盘中可用的存储空间必须大于 200 MB
支持 x86_64, ARMv7, and ARM64 平台

11.1　k3s 核心技术解读?

11.1.1　什么是 k3s?

轻量级的 Kubernetes。安装简单，占用资源少，只需要 512M 内存就可以运行起来，所有的二进制程序都不到 100MB。

自 2019 年 3 月发布以来，备受全球开发者们关注。至今，GitHub Stars 数已超过 16,000，成为了开源社区最受欢迎的边缘计算 K8S 解决方案。

k3s 专为在资源有限的环境中运行 Kubernetes 的研发和运维人员设计，将满足日益增长的在边缘计算环境中运行在 x86、ARM64 和 ARMv7 处理器上的小型、易于管理的 Kubernetes 集群需求。

k3s 的发布，为开发者们提供了以“Rancher 2.X + k3s”为核心的从数据中心到云到边到端的 K8S 即服务（Kubernetes-as-a-Service），推动 Kubernetes Everywhere。

为什么叫 k3s?

我们希望安装的 Kubernetes 在内存占用方面只是原来一半的大小。Kubernetes 是一个 10 个字母的单词。所以，有 Kubernetes 一半大的东西就是一个 5 个字母的单词，简写为 K3S。K3s 没有全称，也没有官方的发音。

11.1.2 适用场景

1、边缘计算-Edge
2、物联网-IoT
3、CI
4、ARM
5、嵌入 K8s

k3s 相对比 k8s 裁剪了如下 5 个部分::

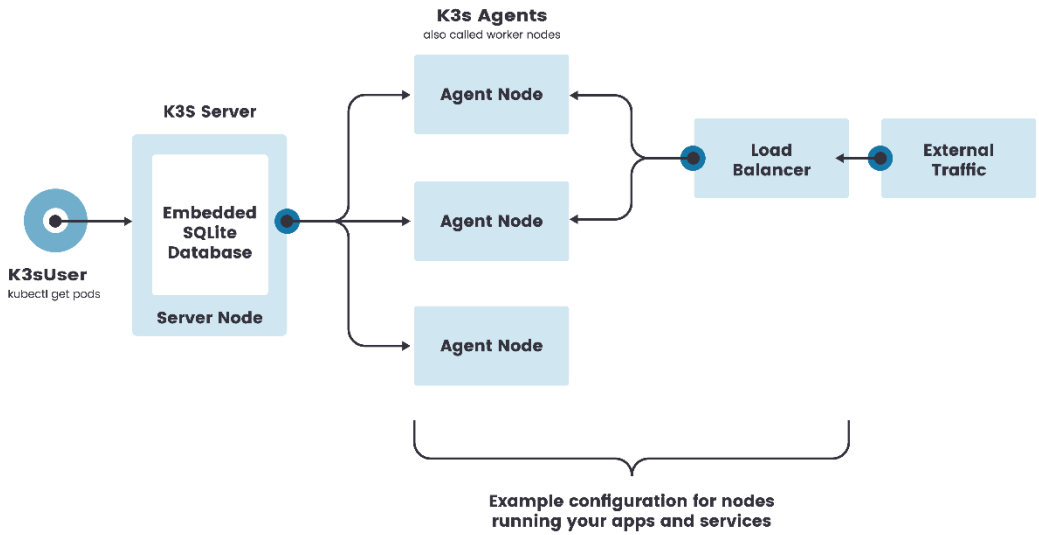
1、过时的功能和非默认功能
2、Alpha 功能
3、内置的云提供商插件
4、内置的存储驱动
5、Docker（可选）

11.1.3 相对 k8s 优化部分

1、使用内嵌轻量级数据库 SQLite 作为默认数据存储替代 etcd，当然 etcd 仍然是支持的。
2、内置了 local storage provider、service load balancer、helm controller、Traefik ingress controller，开箱即用。
3、所有 Kubernetes 控制平面组件如 apiserver、scheduler、controller manager 等封装成为一个精简二进制程序，控制平面只需要一个进程即可运行。
4、删除内置插件(比如 cloudprovider 插件和存储插件)。
5、减少外部依赖，操作系统只需要安装较新的内核（centos7.6 就可以，不需要升级内核）以及支持 cgroup 即可，k3s 安装包已经包含了 containerd、Flannel、CoreDNS，非常方便地一键式安装，不需要额外安装 Docker、Flannel 等组件。

11.2 k3s 架构

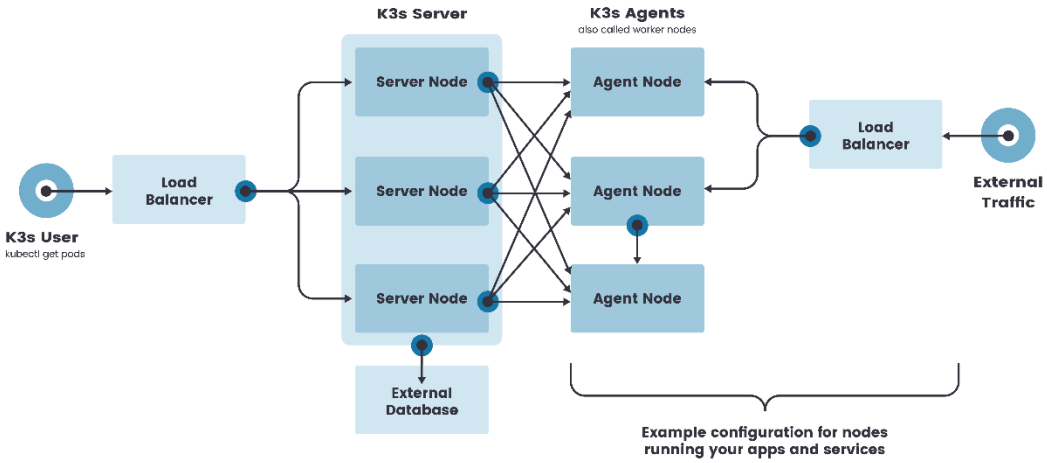
11.2.1 单节点的 k3s 架构



上面的架构里 K3s server 节点有一个内嵌 SQLite 数据库

在这种配置中，每个 agent 节点都注册到同一个 server 节点。K3s 用户可以通过调用 server 节点上的 K3s API 来操作 Kubernetes 资源。

11.2.2 高可用的 k3s 架构



一个 HA K3S 集群由以下几个部分组成：

1、两个或更多 server 节点将为 Kubernetes API 提供服务并运行其他 control-plane 服务

2、外部数据存储（与单节点 k3s 设置中使用的嵌入式 SQLite 数据存储相反）

11.3 k3s 特点

1.上手无代价：

1）使用 k3s 与 Kubernetes 习惯完全一致，对于使用 Kubernetes 的人来讲使用 k3s 没有任何难度；

2）支持部署 helm tiller 服务端（helm tiller 端会在 helm 3.x 版本中被干掉）

2.API server、Controller manager、Scheduler、kubelet、Flannel 等组件都在一个进程中（通过指定是 Server 或者 Agent 选项来控制节点上需要启动哪些组件，Server 相当于 Kubernetes 的 Master 节点，Agent 相当于 Worker 节点），占用的内存更少了，整个 k3s server 进程需要的内存在

500MB 以下。

3. 去除了 Kubernetes 中的一些实验特性、非必须的组件，例如云厂商的驱动、存储插件，k3s 在默认状态下只会启动除自身进程之外的两个应用：

CoreDNS：提供集群内部的 DNS 解析服务。

Traefik：Ingress controller 的角色。

4. 占用资源少：k3s 默认使用 containerd（Server 节点，不可更改）作为容器运行时，不再需要中间层的 Docker Engine，占用资源更少。

5. 部署简单：对环境依赖少，可离线也可在线部署（不过国内的网络环境不推荐在线部署），离线部署时，只需要下载一个大约 40MB 的二进制文件和一个 200MB 不到的离线镜像包，启动 k3s 节点几乎是秒级的。

11.4 选择 k3s 的理由

11.4.1 完美适配边缘计算环境

k3s 是一个高可用的、经过 CNCF 认证的 Kubernetes 发行版，专为无人值守、资源受限、偏远地区或物联网设备内部的生产工作负载而设计。

11.4.2 简单且安全

k3s 被打包成单个小于 60MB 的二进制文件，从而减少了运行安装、运行和自动更新生产 Kubernetes 集群所需的依赖性和步骤。

11.4.3 针对 ARM 进行优化

ARM64 和 ARMv7 都支持二进制文件和多源镜像。k3s 在小到树莓派或大到 AWS a1.4xlarge 32GiB 服务器的环境中均能出色工作。

11.5 云计算 vs 边缘计算

11.5.1 什么是云计算?

对于到底什么叫云计算，有很多种说法。现阶段广为接受的是美国国家标准与技术研究院（NIST）定义：云计算是一种按使用量付费的模式，这种模式提供可用的、便捷的、按需的网络访问， 进入可配置的计算资源共享池（资源包括网络，服务器，存储，应用软件，服务），这些资源能够被快速提供，只需投入很少的管理工作，或与服务供应商进行很少的交互。

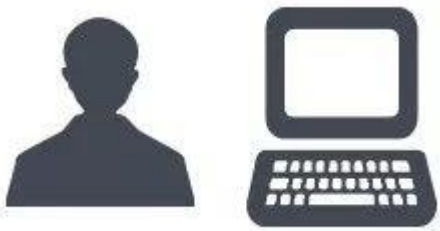
用通俗的话说，云计算就是通过大量在云端的计算资源进行计算，如：用户通过自己的电脑发送指令给提供云计算的服务商，通过服务商提供的大量服务器进行“核爆炸”的计算，再将结果返回给用户。

云计算部署模型分为三类：公有云、私有云、混合云；

服务模式有三种：Saas、PaaS、IaaS 。

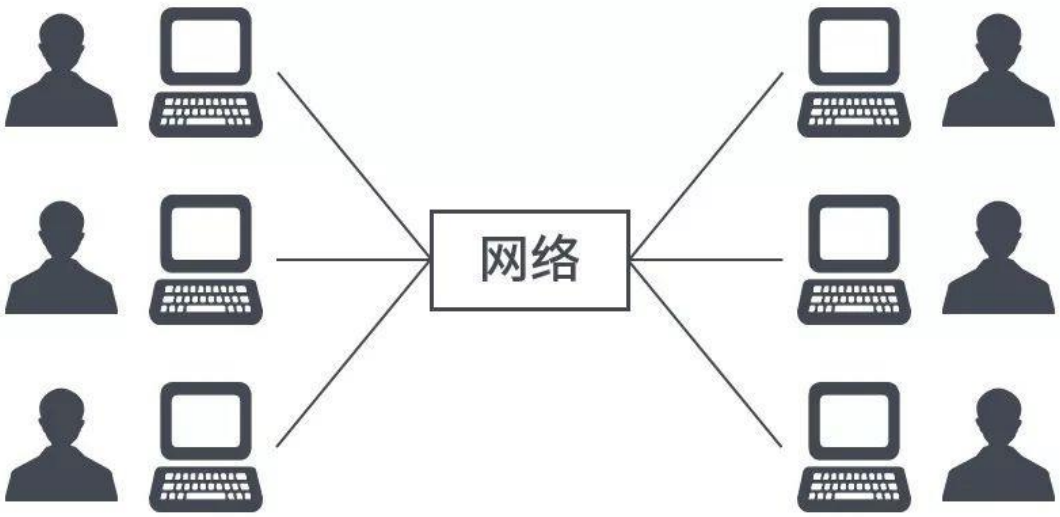
图解云计算：

以前电脑被发明的时候，还没有网络，每个电脑（PC），就是一个单机。

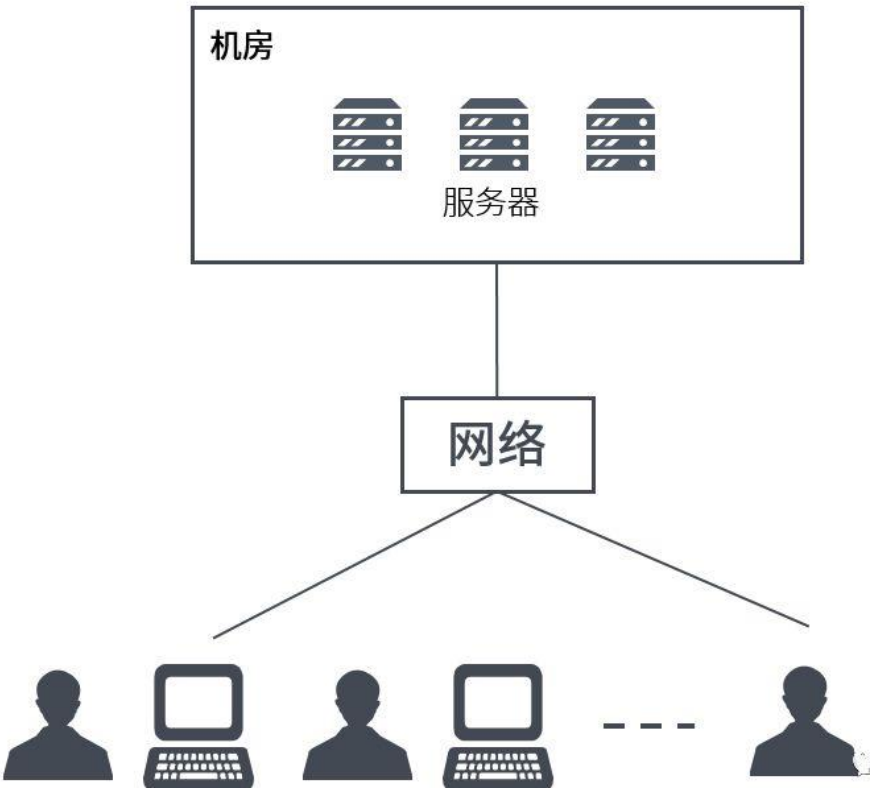


这台单机，包括 CPU、内存、硬盘、显卡等硬件。用户在单机上，安装操作系统和应用软件，完成自己的工作。

后来，有了网络（Network），单机与单机之间，可以交换信息，协同工作。

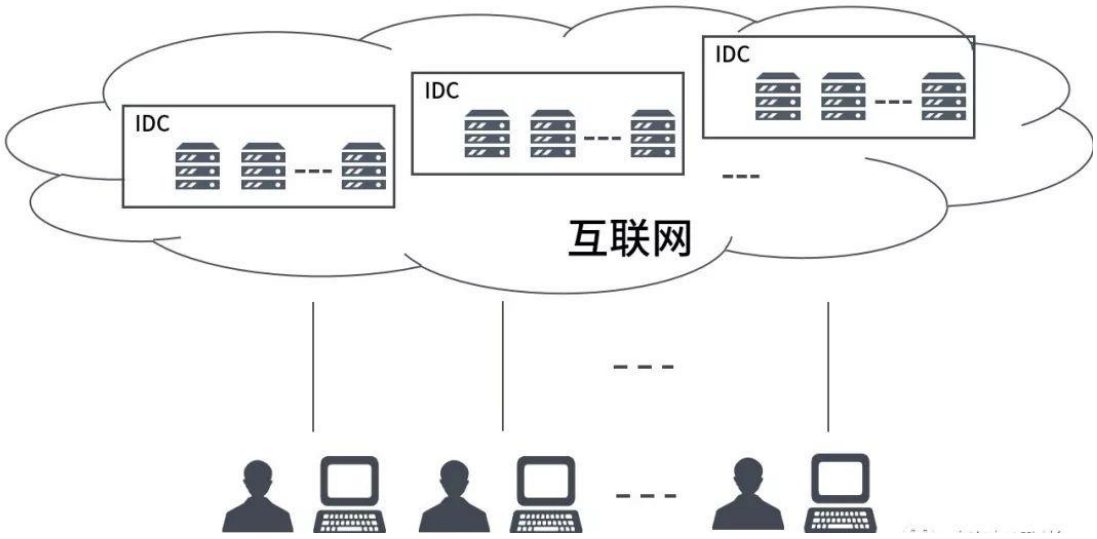


再后来，单机性能越来越强，就有了服务器（Server）。人们发现，可以把一些服务器集 中起来，放在机房里，然后让用户通过网络，去访问和使用机房里的计算机资源。



再再后来，小型网络变成了大型网络，就有了互联网（Internet）。小型机房变成了大型机房，就有了 IDC（Internet Data Center，互联网数据中心）。

当越来越多的计算机资源和应用服务（Application，例如看网页，下电影）被集中起来，就变成了——“云计算（Cloud Computing）”。无数的大型机房，就成了“云端”。



云计算

云计算说白了，就是把计算机资源集中起来，放在网络上。但是，云计算的实现方式，就非常复杂了。

我们把计算机资源放在云端。这个计算机资源，实际上分为好几种层次：

第一层次，是最底层的硬件资源，主要包括 CPU（计算资源），硬盘（存储资源），还有网卡（网络资源）等。

第二层次，要高级一些，我不打算直接使用 CPU、硬盘、网卡，我希望你把操作系统（例如 Windows、Linux）装好，把数据库软件装好，我再来使用

第三层次，更高级一些，你不但要装好操作系统这些基本的，还要把具体的应用软件装好，例如 FTP 服务端软件、在线视频服务端软件等，我可以直接使用服务。

这三种层次，就是大家经常听到的 IaaS、Paas、SaaS。

11.5.2 什么是边缘计算?

“边缘”特指计算资源在地理分布上更加靠近设备，而远离云数据中心的资源节点。典型的边缘计算分为物联网（例如：下一代工业自动化，智慧城市，智能家居，大型商超等）和非物联网（例如：游戏，CDN 等）场景。

边缘计算被称为“人工智能的最后一公里”，但它还在发展初期。相较于云计算，边缘计算有以下势。

1、更多的节点来负载流量，使得数据传输速度更快。

2、更靠近终端设备，传输更安全，数据处理更即时。

3、更分散的节点相比云计算故障所产生的影响更小。

11.5.3 边缘计算应用场景

边缘计算在智慧城市的建设中有丰富的应用场景。在城市路面检测中，在道路两侧路灯上安装传感器收集城市路面信息，检测空气质量+、光照强度、噪音水平等环境数据，当路灯发生故障时能够及时反馈至维护人员。

在智能交通中，边缘服务器上通过运行智能交通控制系统来实时获取和分析数据，根据实时路况来控制交通信息灯，以减轻路面车辆拥堵等。

在无人驾驶中，如果将传感器数据上传到云计算中心将会增加实时处理难度，并且受到网络制约，因此无人驾驶主要依赖车内计算单元来识别交通信号和障碍物，并且规划路径

《福布斯》技术委员会的 12 位成员研究了一些企业的用例，在这些用例中，都使用了边缘计算：

应用一：改进医疗设备性能和数据管理

在医疗场景下，边缘计算主要帮助医疗保健体系的 IT 基础架构，具体来说，是防止医疗设备管理的应用程序发生延迟。在边缘计算的支持下，无需构建集中的数据中心，可对关键数据进行本地化，在安全性、响应速度和有效性上有更佳表现。

应用二：本地零售的实时数据分析

边缘计算的主要目的，是让运算尽可能接近数据源。在零售场景中，以往企业都是将各分支的数据汇总到中心位置进行分析，再进行决策和行动。而通过边缘计算，零售店铺可以在本地就进行数据处理和优化，这样组织的行动反馈就能更快更及时。

应用三：消费者数据隐私

处理高度敏感的消费者数据的公司发现，由于数据泄漏成本高昂，云计算很危险。因此，许多这类公司正在将边缘计算用于处理消费者数据，因为它为他们提供了更多的安全和控制选择。这可能会使企业工作流程复杂化，但它可以带来好处，特别是在医疗数据公司中。- Sean Byrnes, Outlier

应用四：物联网

物联网是企业为获得更好的数据、客户体验、现场营销和更智能的流程而实施的智能的重大转变。边缘计算使网络的运行水平能够为物联网提供高性能。接近实时的速度意味着用户体验可以非常出色，并且可以为企业带来更好、更有效的运营。-Frank Cittadino, QOS Networks

应用五：视频监控和分析

视频监控已经部署在边缘，以及视频分析。视频产生的数据是所有其他来源数据总和的 10 倍。这是一个

很好的例子，说明您需要在数据中心之外开始处理的数据类型，以最大限度地减少通过网络移动和存储的数据量。此外，它还允许企业对可操作的数据做出实时决策- Bill Galloway，Pivot3 Inc.

应用六：工业操作

工厂车间处理等工业操作需要海量数据的快速响应。这些数据中的大部分被丢弃，但能够通过检测边缘数据中的异常来使工厂操作员能够及时对工厂车间的问题做出反应。这样可以间接减少停机从而提高生产率。- Randal Kenworthy，Cognizant

应用七：智能建筑

随着我们的会议室和设施变得越来越智能化，将会有一种把所有东西都放到云端的诱惑。考虑到安全性和稳定性，在站点上托管数据将会有一些好处。我们已经看到了基于云的照明在网络中断时可能会令人沮丧的问题，而工作场所的可靠性甚至更重要。 Luke Wallace, Bottle Rocket

应用八：无人驾驶汽车

汽车行业已投入数十亿资金开发边缘计算技术。为了安全运行，这些车辆需要收集和分析有关其周围环境，方向和天气状况的大量数据，以及与道路上的其他车辆进行通信。- Arnie Gordon，Arlyn Scales

应用九：机顶盒

Edge AI 可以在机顶盒设备上运行一个低占用空间的智能引擎，可以实时分析数据，对 AI 进行编码，在本地解决问题并向上游服务器提供智能反馈。与基本自动化（边缘机器人流程自动化机器人）相结合，AI 引擎可以自我修复并在本地解决许多问题。- Ankur Garg，Hotify Inc.

11.5.4 小结

云计算是人和计算设备的互动，而边缘计算则属于设备与设备之间的互动，最后再间接服务于人。边缘计算可以处理大量的即时数据，而云计算最后可以访问这些即时数据的历史或者处理结果并做汇总分析。云计算和边缘计算是一种共生和互补的关系，并不会出现谁取代谁的问题，而是谁在哪些计算上更有优势，谁在哪些场景上更合适。

11.6 k3s 和 k8s 如何选择?

K8s 和 k3s 各有优劣。若是你要进行大型的集群部署，建议你选择使用 K8s；若是你处于边缘计算等小型部署的场景或仅仅须要部署一些非核心集群进行开发/测试，那么选择 k3s 则是性价比更高的选择。

云计算场景用 k8s

边缘计算场景用 k3s

Q：k3s 和 k8s 具体有多大的差别?

A：在实际的应用部署中，几乎没有任何差异，至少到目前为止，我所遇到的场景，k8s 能满足的，k3s 也能满足，相信，通过不断的迭代，k3s 在未来会更完善边缘场景。

Q：能直接学习 k3s 吗?

A: k3s 的所有操作跟 k8s 无区别，但是学习还是以 k8s 为主，之后再扩展 k3s

11.6.1 k3s 在边缘计算中的应用

背景：

在进行信息化、智能化改造的过程中，首先第一步，就是要获取底层系统的全方位的数据。

因此需要部署大量的边缘设备来采集数据、分析数据，通过这些数据进行建模，大量的边缘设备一般离散的分布在不同机房、厂区、甚至是不同的地理区域，这对运维人员来讲维护这些设备，管理其上运行的应用变得极其困难。

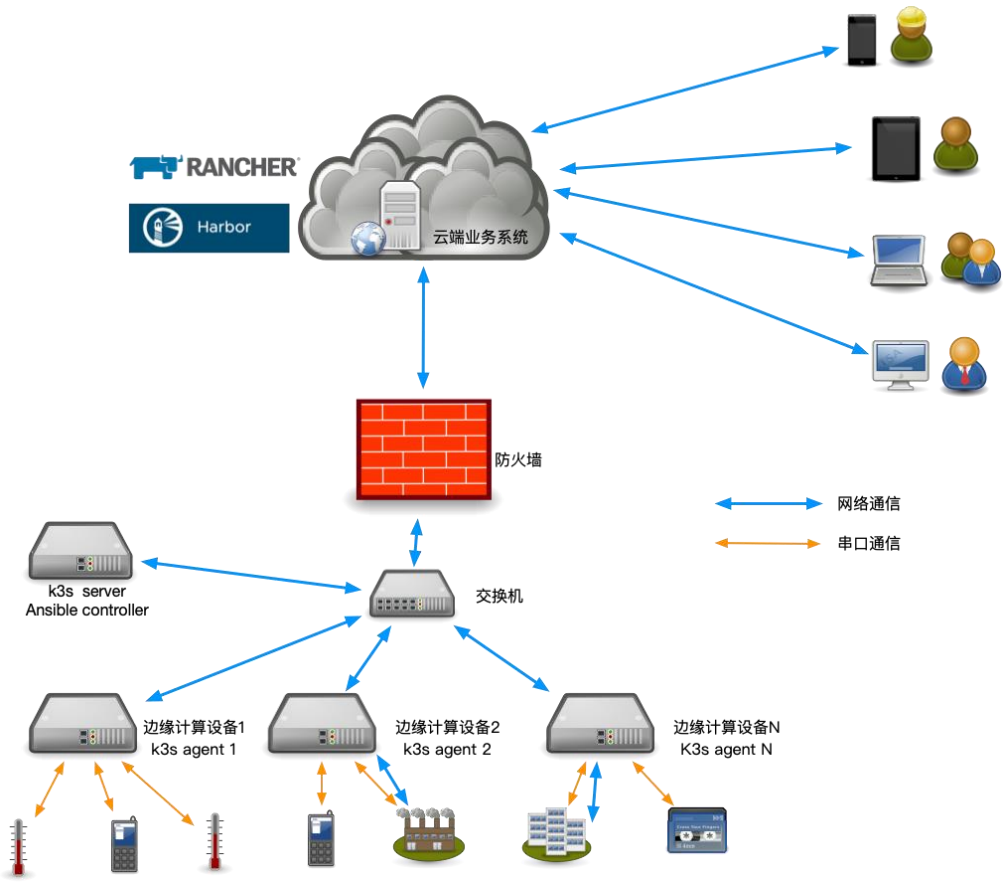
需求：

对于运维角色来讲：

管理这些边缘设备，保持边缘设备上运行的服务的高可用性；

快速的上线、升级

配置的快速更改与应用



以上案例来自曾永杰，上海全应科技有限公司运维经理，曾在华为西安研究所云计算部门承担软件测试工程师、项目交付、线上运维等工作职责，当前工作主要为 CI/CD 流程的建设与维护、应用的容器化改革和容器云平台的运维管理。

实战 1：安装 k3s 集群

#在 xuegod65 上操作：

初始化实验环境：

1、配置静态 IP

把虚拟机或者物理机配置成静态 ip 地址，这样机器重新启动后 ip 地址也不会发生改变。

在 xuegod65 节点配置网络

修改/etc/sysconfig/network-scripts/ifcfg-ens33 文件，变成如下：

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
IPADDR=192.168.40.135
NETMASK=255.255.255.0
GATEWAY=192.168.42.2
DNS1=192.168.42.2
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
DEVICE=ens33
ONBOOT=yes
```

修改配置文件之后需要重启网络服务才能使配置生效，重启网络服务命令如下：

```
systemctl restart  network
```

2、修改 yum 源

下面的步骤在 xuegod65 节点操作

配置安装 k8s 需要的 yum 源

```
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
EOF
```

#添加 docker 需要的 yum 源

```
yum-config-manager --add-repo  http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
```

安装软件包

yum -y install wget net-tools nfs-utils lrzsz gcc gcc-c++ make cmake libxml2-devel openssl-devel curl curl-devel unzip sudo ntp libaio-devel wget vim ncurses-devel autoconf automake zlib-devel python-devel epel-release openssh-server socat ipvsadm conntrack ntpdate yum-utils device-mapper-persistent-data lvm2 telnet

3、配置防火墙

关闭 firewalld 防火墙， centos7 系统默认使用的是 firewalld 防火墙，停止 firewalld 防火墙，并禁用这个服务。

systemctl stop firewalld && systemctl disable firewalld

4、时间同步

ntpdate cn.pool.ntp.org

编辑计划任务，每小时做一次同步

1）crontab -e

* */1 * * * /usr/sbin/ntpdate cn.pool.ntp.org

2）重启 crond 服务：

systemctl restart crond

5、关闭 selinux

关闭 selinux，设置永久关闭，这样重启机器 selinux 也处于关闭状态

可用下面方式修改：

sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/sysconfig/selinux

sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config

上面文件修改之后，需要重启虚拟机，如果测试环境可以用如下命令强制重启：

reboot -f

注：生产环境不要 reboot -f，要正常关机重启

查看 selinux 是否修改成功

重启之后登录到机器上用如下命令：

getenforce

显示 Disabled 说明 selinux 已经处于关闭状态

6、修改内核参数

```
cat <<EOF >/etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sysctl --system
```

注：*sysctl --system* 这个会加载所有的 *sysctl* 配置

7、修改主机名

在 192.168.40.135 上：

hostnamectl set-hostname xuegod65 && bash

8、配置 hosts 文件

xuegod65、xuegod66 的 hosts 文件保持一致，可按如下方法修改：

在/etc/hosts 文件增加如下几行：

192.168.40.135 xuegod65

192.168.40.136 xuegod66

9、配置主机之间无密码登陆

配置 xuegod65 到 xuegod66 无密码登陆

在 xuegod65 上操作

ssh-keygen -t rsa

#一直回车即可

cd /root && ssh-copy-id -i .ssh/id_rsa.pub root@192.168.40.136

#上面需要输入 yes 之后，输入密码，输入 192.168.40.136 物理机密码即可

10、安装 k3s

#先把安装 k3s 需要的镜像 k3s-server.tar.gz 上传到 xuegod65 机器上，手动解压:

[root@xuegod65 ~]#yum install containerd -y

[root@xuegod65 ~]# systemctl start containerd

[root@xuegod65 ~]# ctr images import k3s-server.tar.gz

国内用户可以用如下方法：安装速度会更快

在 xuegod65 上操作:

[root@xuegod65 ~]# curl -sfL http://rancher-mirror.cnrancher.com/k3s/k3s-install.sh | INSTALL_K3S_MIRROR=cn sh -

验证安装是否成功

[root@xuegod65 ~]# kubectl get nodes

```
NAME        STATUS   ROLES                  AGE   VERSION
xuegod65    Ready    control-plane,master   30s   v1.21.1+k3s1
```

[root@xuegod65 ~]# k3s kubectl get pods -n kube-system

显示如下:

```
NAME                                     READY   STATUS      RESTARTS   AGE
coredns-854c77959c-k74k6                 1/1     Running     0          2m
local-path-provisioner-5ff76fc89d-7q7pk  1/1     Running     0          2m
metrics-server-86cbb8457f-s8pl9          1/1     Running     0          2m
helm-install-traefik-26brw               0/1     Completed   0          2m
svclb-traefik-6flmb                      2/2     Running     0          57s
traefik-6f9cbd9bd4-95jcb                 1/1     Running     0          57s
```

实战 2：在 k3s 集群添加 agent 节点

#在 xuegod66 上操作:

初始化实验环境:

1、配置静态 IP

把虚拟机或者物理机配置成静态 ip 地址，这样机器重新启动后 ip 地址也不会发生改变。

在 xuegod66 节点配置网络

修改/etc/sysconfig/network-scripts/ifcfg-ens33 文件，变成如下:

TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
IPADDR=192.168.40.136
NETMASK=255.255.255.0
GATEWAY=192.168.42.2
DNS1=192.168.42.2
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
DEVICE=ens33
ONBOOT=yes
修改配置文件之后需要重启网络服务才能使配置生效，重启网络服务命令如下：
systemctl restart network

2、修改 yum 源
下面的步骤在 xuegod66 节点操作
配置安装 k8s 需要的 yum 源
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
EOF

#添加 docker 需要的 yum 源
yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

安装软件包
yum -y install wget net-tools nfs-utils lrzsz gcc gcc-c++ make cmake libxml2-devel openssl-devel curl curl-devel unzip sudo ntp libaio-devel wget vim ncurses-devel autoconf automake zlib-devel python-devel epel-release openssh-server socat ipvsadm conntrack ntpdate yum-utils device-mapper-persistent-data lvm2 telnet

3、配置防火墙

关闭 firewalld 防火墙， centos7 系统默认使用的是 firewalld 防火墙，停止 firewalld 防火墙，并禁用这个服务。

systemctl stop firewalld && systemctl disable firewalld

4、时间同步

ntpdate cn.pool.ntp.org

编辑计划任务，每小时做一次同步

1）crontab -e

* */1 * * * /usr/sbin/ntpdate cn.pool.ntp.org

2）重启 crond 服务：

service crond restart

5、关闭 selinux

关闭 selinux，设置永久关闭，这样重启机器 selinux 也处于关闭状态

可用下面方式修改：

sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/sysconfig/selinux

sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config

上面文件修改之后，需要重启虚拟机，如果测试环境可以用如下命令强制重启：

reboot -f

注：生产环境不要 reboot -f，要正常关机重启

查看 selinux 是否修改成功

重启之后登录到机器上用如下命令：

getenforce

显示 Disabled 说明 selinux 已经处于关闭状态

6、修改内核参数

```
cat <<EOF > /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sysctl --system
```

注：*sysctl --system* 这个会加载所有的 *sysctl* 配置

7、修改主机名

在 192.168.40.136 上：

hostnamectl set-hostname xuegod66

8、配置 hosts 文件

xuegod65、xuegod66 主机的 hosts 文件保持一致，可按如下方法修改：

在/etc/hosts 文件增加如下几行：

192.168.40.135 xuegod65

192.168.40.136 xuegod66

提取 join token

我们想要添加一对 worker 节点。在这些节点上安装 K3s，我们需要一个 join token。Join token 存在于 k3s server 节点的文件系统上。让我们复制并将它保存在某个地方，稍后我们可以获取它：

[root@xuegod65 ~]# cat /var/lib/rancher/k3s/server/node-token

获取到一串 token：
K10f181d6db4ac43a93504cd22d6c6e78601779d6d6aa138979051ace98062e0a57::server:17d84d0191ce452a7036f1b8c7326786

在 xuegod66 上执行如下，把 agent 节点加入 k3s:
#先把安装 k3s agent 需要的镜像 k3s-agent.tar.gz 上传到 xuegod66 机器上，手动解压:
[root@xuegod65 ~]#yum install containerd -y
[root@xuegod65 ~]# systemctl start containerd
[root@xuegod66 ~]# ctr images import k3s-agent.tar.gz
#执行如下把 xuegod66 机器，加入到 k3s 集群

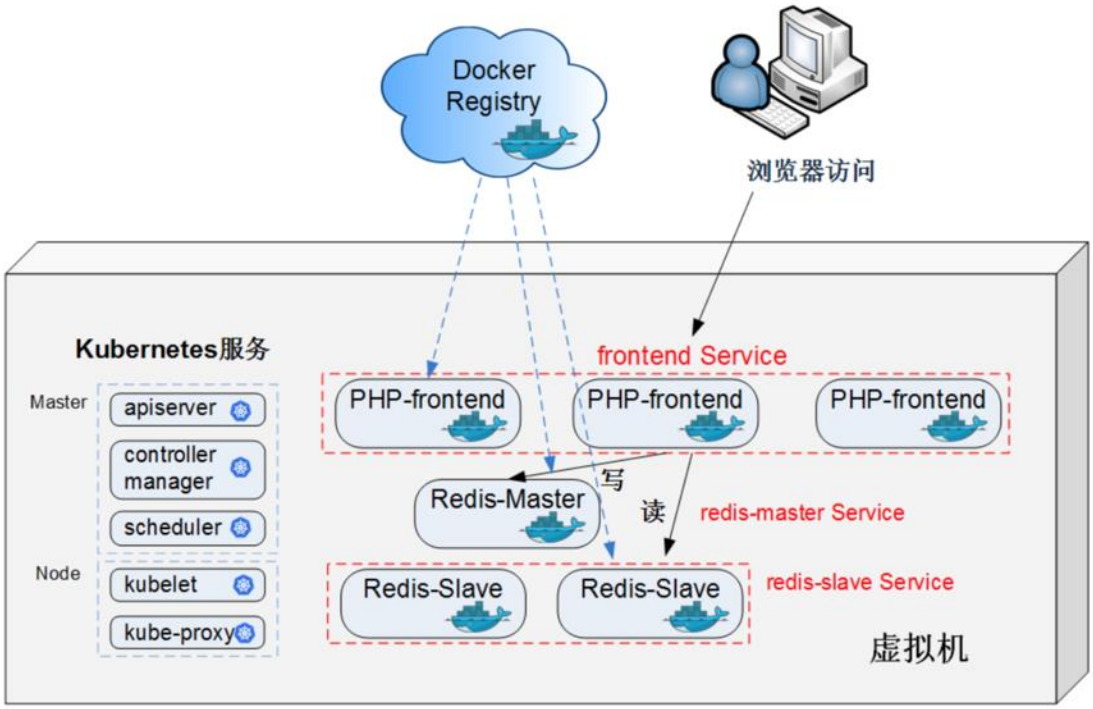
[root@xuegod66 ~]# curl -sfL http://rancher-mirror.cnrancher.com/k3s/k3s-install.sh | INSTALL_K3S_MIRROR=cn K3S_URL=https://192.168.40.135:6443 K3S_TOKEN=K10f181d6db4ac43a93504cd22d6c6e78601779d6d6aa138979051ace98062e0a57::server:17d84d0191ce452a7036f1b8c7326786 sh -

验证 work 节点是否加入集群:
在 xuegod65 上操作:
[root@xuegod65 ~]# k3s kubectl get nodes

显示如下，说明工作节点成功加入 k3s 集群:

192.168.40.135	Ready	control-plane,master	60s	v1.20.4+k3s1
192.168.40.136	Ready	<none>	60s	v1.20.4+k3s1

实战 3：在 k3s 中部署应用- Guestbook 留言板



在做下面实验之前需要先把离线镜像包上传到安装 k3s 的 server 节点和 agent 节点，我这里传到 xuegod65 和 xuegod66 上：

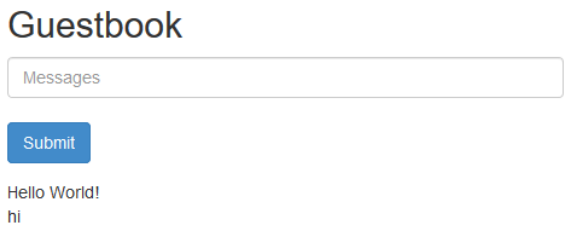
通过 ctr 解压镜像：

```
[root@xuegod66 ~]# ctr images import frontend.tar.gz
[root@xuegod66 ~]# ctr images import redis-master.tar.gz
[root@xuegod66 ~]# ctr images import redis-slave.tar.gz
#接下来的步骤在 k3s server 节点 xuegod65 上操作：
[root@xuegod65 ~]# cat   redis-master-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-master
  labels:
    app: redis
spec:
  selector:
    matchLabels:
      app: redis
      role: master
      tier: backend
  replicas: 1
  template:
    metadata:
      labels:
        app: redis
        role: master
        tier: backend
    spec:
      containers:
      - name: master
        image: kubeguide/redis-master
        ports:
        - containerPort: 6379
[root@xuegod65 ~]# kubectl apply -f redis-master-deployment.yaml

[root@xuegod65 ~]# cat   redis-master-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: redis-master
  labels:
    app: redis
    role: master
```

```
    tier: backend
spec:
  ports:
  - port: 6379
    targetPort: 6379
  selector:
    app: redis
    role: master
    tier: backend

[root@xuegod65 ~]# kubectl apply -f redis-master-service.yaml

[root@xuegod65 ~]# cat   redis-slave-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-slave
  labels:
    app: redis
spec:
  selector:
    matchLabels:
      app: redis
      role: slave
      tier: backend
  replicas: 1
  template:
    metadata:
      labels:
        app: redis
        role: slave
        tier: backend
    spec:
      containers:
      - name: slave
        image: kubeguide/guestbook-redis-slave
        env:
        - name: GET_HOSTS_FROM
          value: dns
        ports:
        - containerPort: 6379

[root@xuegod65 ~]# kubectl apply -f redis-slave-deployment.yaml
```

```
[root@xuegod65 ~]# cat   redis-slave-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: redis-slave
  labels:
    app: redis
    role: slave
    tier: backend
spec:
  ports:
  - port: 6379
  selector:
    app: redis
    role: slave
    tier: backend

[root@xuegod65 ~]# kubectl apply -f redis-slave-service.yaml

[root@xuegod65 ~]# cat frontend-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: guestbook
spec:
  selector:
    matchLabels:
      app: guestbook
      tier: frontend
  replicas: 1
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: kubeguide/guestbook-php-frontend
        env:
        - name: GET_HOSTS_FROM
          value: dns
```

```
        ports:
        - containerPort: 80

[root@xuegod65 ~]# kubectl apply -f frontend-deployment.yaml

[root@xuegod65 ~]# cat   frontend-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 80
    nodePort: 30001
  selector:
    app: guestbook
    tier: frontend

[root@xuegod65 ~]# kubectl apply -f frontend-service.yaml
#查看刚才创建的应用是否正常
[root@xuegod65 ~]# kubectl get pods
NAME                              READY    STATUS     RESTARTS    AGE
redis-master-65ff54f7-8qd29       1/1      Running    0           9m8s
redis-slave-5f69bfdfc4-s5h6z      1/1      Running    0           6m29s
frontend-77b8d94b79-sv644         1/1      Running    0           2m58s

#查看对应的 service
[root@xuegod65 ~]# kubectl get svc
NAME           TYPE          CLUSTER-IP       EXTERNAL-IP    PORT(S)          AGE
kubernetes     ClusterIP     10.43.0.1        <none>         443/TCP          61m
redis-master   ClusterIP     10.43.14.52      <none>         6379/TCP         8m17s
redis-slave    ClusterIP     10.43.81.97      <none>         6379/TCP         5m28s
frontend       NodePort      10.43.47.208     <none>         80:30001/TCP     2m20s

#在浏览器访问 frontend 在宿主机暴露的 ip:端口即可访问到前端页面：
http://192.168.40.135:30001/
```



11.7 卸载 k3s 集群

在 k3s server 节点执行如下：
/usr/local/bin/k3s-uninstall.sh

在 k3s agent 节点执行如下：
/usr/local/bin/k3s-agent-uninstall.sh

总结