JLINK-100 使用手册

1. 产品简介

极联 JLINK-100 型边缘计算集群采用工业 CPCI 架构，由若干个边缘计算模块及 1 个交换机模块组成，可提供边缘计算、数据存储、web 服务等功能，协同云服务器真正实现“云边协作”，广泛应用于 CNC、PLC 等工业设备的生产管理中。

产品特色：

- High performance 高性能

  采用 KUBERNETES 架构，可根据具体业务应用需求动态增加计算节点，支持 ARM /X86 架构；

- Auto Deployment 自动化部署

  利用容器技术，快速自动部署相关应用，并可根据负载自动均衡容器的相关配置；

- High Reliable 高可靠

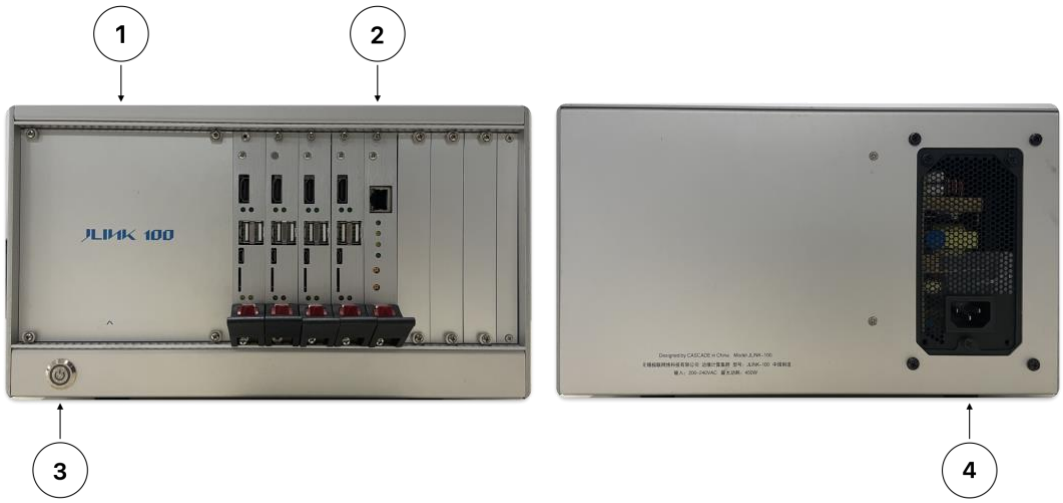
  不管是硬件故障还是软件故障，集群都可在新的节点重启容器，保证业务应用持续运行。同时，实时备份在云端的配置数据及应用数据可快速恢复应用系统；

- High Security 高安全

  集群采用 TPM 芯片生成和处理关键信息，如安全令牌，密钥等。

2. 硬件介绍

2.1 物理接口



1) ATX 电源模块

JLINK-100 采用一块 450W 全模组电源，为功能模块提供 3.3V、5V 及 12V 电源信号。

2) 模块插槽

JLINK-100 提供 8 个 3U CPCI 插槽，从左到右序号依次为 1-8，其中插槽 5 为交换机模块专用插槽，其余为功能模块通用插槽。

3) 电源按键及指示灯

4) 220V 品字型交流电源接口

3. 集群部署

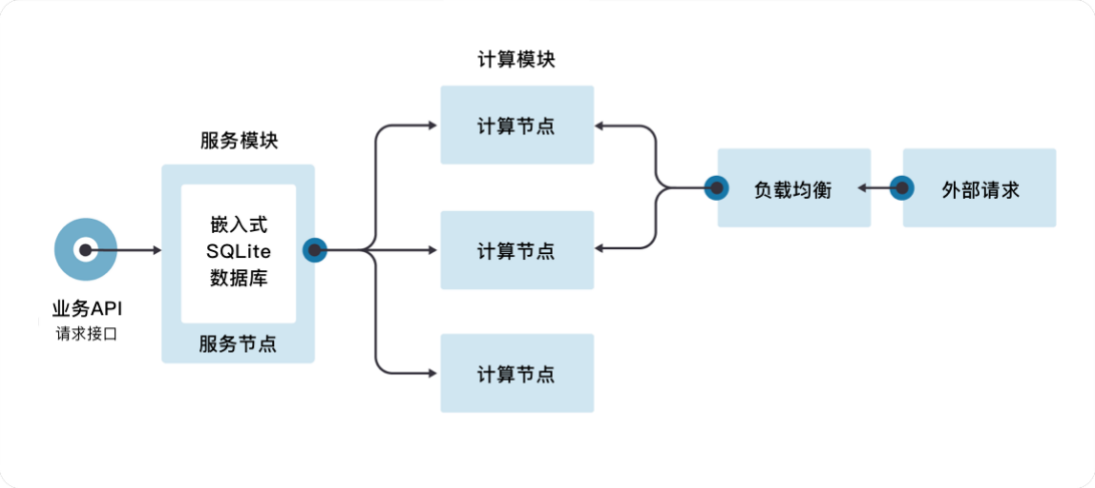
3.1 框架介绍

JLINK-100 采用 K3S 集群框架，K3S 是一个轻量级的 Kubernetes 发行版，它针对边缘计算、物联网等场景进行了高度优化。K3S 嵌入式集成了 K8S 集群所需的几乎所有关键组件，其中从 Kubernetes 到 FLannel 被集成到了 K3S 二进制可执行程序；从 Metrics-server 到 Local-

path-provisioner 是通过容器的方式提供。嵌入式组件 Kubernetes 包括：kube-apisesrver、kube-controller-manager、kube-scheduler、kubelet、kube-proxy；根据传入的参数不同启动不同的组件服务：/usr/local/bin/k3s server 启动 Master 节点上所需服务；/usr/local/bin/k3s agent 启动 Worker Node 节点上所需服务。此外，K3S 二进制可执行程序还包含 K8S 及其组件启动所需的配置。

3.2 硬件部署

JLINK-100 默认配置为 4 块边缘计算模块和 1 块交换机模块，组成 1Server+3Agent 的 K3S 集群，其中交换机模块用于实现边缘计算模块之间的数据通信。



1Server+3Agent

3.3 K3S 部署

1) 安装时可以使用 Docker 作为容器运行。

```
$ curl https://releases.rancher.com/install-docker/19.03.sh | sh

$ curl -sfL https://get.k3s.io | sh -s - --docker
```

2) 普通用户执行客户端命令。

```
$ kubectl get node
WARN[2021-04-09T19:22:42.934501246+08:00] Unable to read /etc/rancher/k3s/k3s.yaml, please start serv
error: error loading config file "/etc/rancher/k3s/k3s.yaml": open /etc/rancher/k3s/k3s.yaml: permiss

$ ctr c ls
ctr: failed to dial "/run/k3s/containerd/containerd.sock": connection error: desc = "transport: error
# ll /run/k3s/containerd/containerd.sock
srw-rw----. 1 root root 0 Apr  7 17:48 /run/k3s/containerd/containerd.sock

$ crictl ps -a
FATA[2021-04-09T19:01:20.029650977+08:00] load config file: open /var/lib/rancher/k3s/agent/etc/crict

$ ll /var/lib/rancher/k3s/agent/etc/crictl.yaml
-rw-------. 1 root root  61 Apr  9 12:33 crictl.yaml
```

3) 如上所示，默认情况下，普通用户无法执行 kubectl、ctr、crictl 等客户端命令。因为依赖的 config 或 socket 文件权限太严格，普通用户没有读权限。放开文件的可读权限给普通用户即可。安全期间，还是不要这么做。下面操作可令普通用户执行 kubectl 命令，其他两个命令还是需要 root 账号。

```
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/rancher/k3s/k3s.yaml $HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config

$ vi ~/.bash_profile
export KUBECONFIG=~/.kube/config
$ source ~/.bash_profile
```

4) 执行完成之后，在主节点查看新节点是否注册成功，执行命令 k3s kubectl get node -A。

```
[root@localhost bin]# k3s kubectl get node -A
NAME                   STATUS   ROLES                  AGE    VERSION
localhost.localdomain  Ready    control-plane,master   5h6m   v1.24.1+k3s1
laptop-7hc3feq9        Ready    <none>                 66s    v1.24.1+k3s1
```

5) 查看一下 pod 信息。

```
[root@localhost bin]# kubectl get pods -A -o wide
NAMESPACE     NAME                                      READY   STATUS      RESTARTS   AGE
kube-system   coredns-b96499967-ggjk5                   1/1     Running     0          5h13m
kube-system   local-path-provisioner-7b7dc8d6f5-fxwgk   1/1     Running     0          5h13m
kube-system   helm-install-traefik-crd-tql4c            0/1     Completed   0          5h13m
kube-system   helm-install-traefik-rs5g6                0/1     Completed   1          5h13m
kube-system   svclb-traefik-2zcq6                       2/2     Running     0          5h12m
kube-system   metrics-server-668d979685-ff78t           1/1     Running     0          5h13m
kube-system   traefik-7cd4fcff68-r6kgf                  1/1     Running     0          5h12m
kube-system   svclb-traefik-wv6jn                       2/2     Running     2          7m56s
```

3.4 K3S 应用部署

1) 制作 docker 镜像，可以用 docker images 查看到；

```
FROM openjdk:1.8
MAINTAINER wt
ADD follow-1.0.0-SNAPSHOT.jar follow.jar
#配置时区
ENV TZ=Asia/Shanghai
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
EXPOSE 8080
ENTRYPOINT ["java","-jar","/follow.jar","--spring.profiles.active=${ACTIVE}"]
```

2) 创建 pod；

```
create  deployment  follow  --image=follow --dry-run -o yaml > follow.yaml
kubectl apply -f  follow.yaml
```

3) 创建 svc；

```
kubectl expose deployment follow --port=80 --target-port=8080 --type=NodePort -o yaml --dry-run >
kubectl apply -f  svc.yaml
```

4) 检查 pod、svc 是否创建成功；

```
kubectl  get pod,svc
```

```
[redacted]# kubectl  get pod,svc
NAME                          READY   STATUS    RESTARTS   AGE
pod/follow-fc89db49d-n5b5t    1/1     Running   0          19h

NAME                 TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
service/kubernetes   ClusterIP   10.43.0.1     <none>        443/TCP        8d
service/follow       NodePort    10.43.43.78   <none>        80:31697/TCP   4h38m
```

5) 应用部署完成。