**类脑云OS部署方案**

1. **方案简介**

**1.1方案概述**

部署设计方案，以数据中心搭建，进行总体设计，合理规划已有和后期加入的硬件资源，最大化解决硬件的使用需求问题，并且设计遵循以下原则：

**开放性：**系统符合开放性设计原则，具备优良的可扩展性、可升级性，可以支持开放系统平台，运行于现有的技术标准之上；

**兼容性：**采用通用产品与技术平台与现有系统及未来需要整合的系统需要完全兼容，各个构成子系统必须紧密衔接、高度集成，构成一个整体；

**稳定性：**要保证系统运行的稳定性，使系统运行风险降至最低；

**业务连续性**：支持未来完善的远程多点容灾解决方案，满足未来业务连续性的需要；

**技术先进性：**系统设计应采用当前先进而成熟的技术，不仅可以满足本期工程的需求，也应把握未来的发展方向；

**可扩充性：**在系统设计时应充分考虑可扩充性，从而确保新功能、新业务的增加在原有的系统平台上扩展和实现；

**高可靠性：**系统平台具有高可靠性，支持服务器平台的高可用性集群技术；具备先进的容灾的设计；充分保证系统的高扩展能力和高容错能力，具有通道负载自动均衡能力和存储系统性能调节能力，提供极为充分的可靠性各项指标设计；

**高可用性：**在不停机情况下，实现不停机扩容、维护、升级等服务，提高性能以满足新的业务需求。具备7×24×365连续工作的能力，系统的可用性应大于99.99％。

**成熟性：**应尽量选用经过大量运用、成熟可靠的系统；

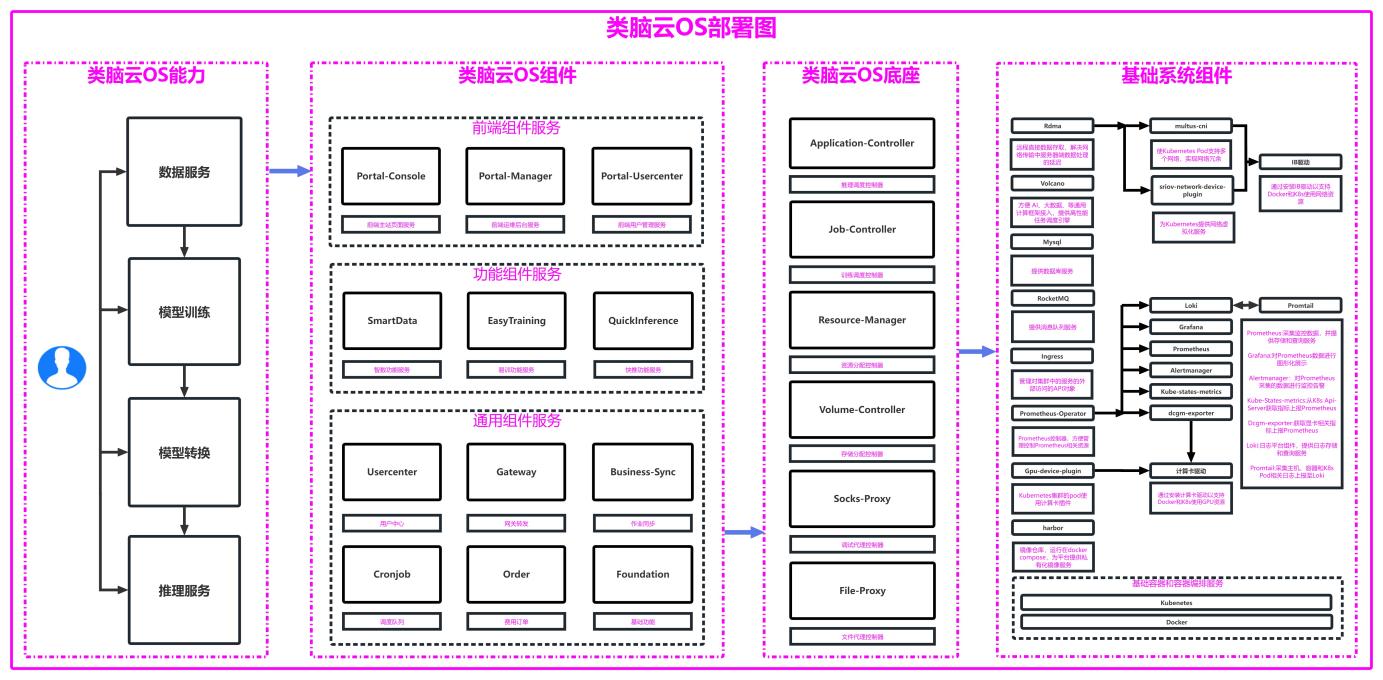
**可管理性：**要求配置实时性能监测管理软件。可对主机、存储等设备CPU使用率、内存使用率、交换区使用情况、I/O操作、队列状态、磁盘空间、卷磁盘错误、系统事件、系统中各进程对系统资源占用等性能和操作数据等服务器性能进行实时监控和管理；

**可实施性：**选用成熟的技术，成熟的案例经验和设计方案，制定详细的技术实施方案；

**经济性原则：**在满足所有需求的前提下，选择最合适的设备及管理软件，使系统具有较好的性价比；

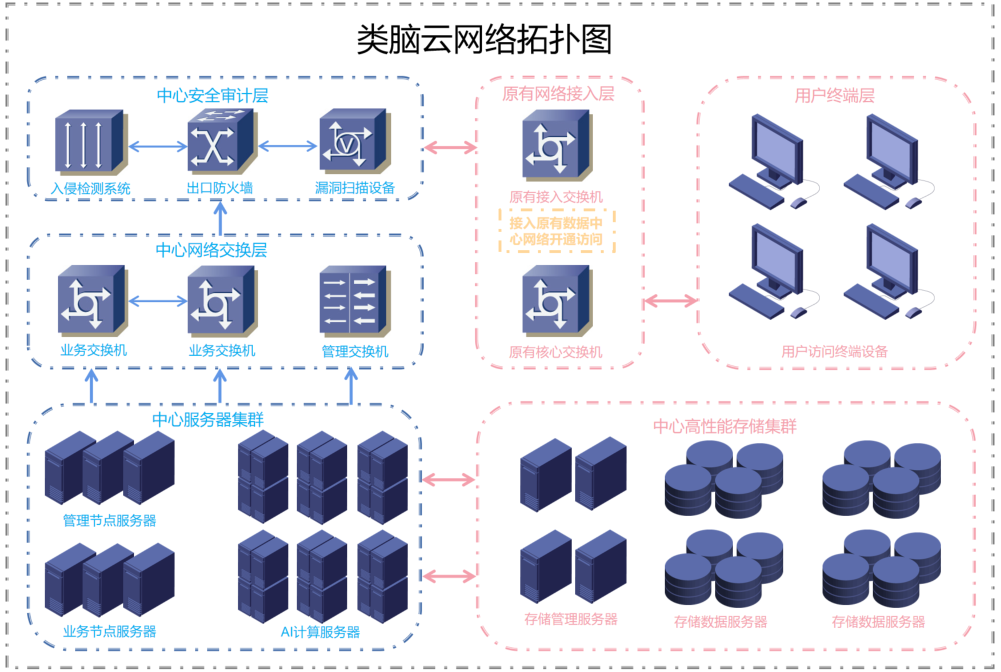
**1.2组件部署图**

类脑云OS整体构成为分为类脑云OS组件、类脑云OS底座和基础系统组件3部分，对对应类脑云OS的多种能力。



**1.3网络拓扑图**

整体方案由CPU服务器（业务管理节点）、GPU服务器（计算节点）、存储集群和网络安全设备构成



1. **部署步骤**

**2.1部署环境**

**2.1.1 环境信息**

本次方案以部署6台CPU服务器，和2台GPU服务器为例：

3台CPU服务器做kubernetes Master节点高可用管理和中间件高可用调度服务器

3台CPU服务器做系统前后端组件的高可用调度服务器

3台GPU服务器做训练和推理任务的调度服务器

**2.1.2 服务器信息**

|  |  |  |  |
| --- | --- | --- | --- |
| 服务器IP | 服务器类型 | 主机名称 | GPU卡型 |
| 192.168.1.1 | CPU服务器 | node-cpu-1 | 无 |
| 192.168.1.2 | CPU服务器 | node-cpu-2 | 无 |
| 192.168.1.3 | CPU服务器 | node-cpu-3 | 无 |
| 192.168.1.4 | CPU服务器 | node-cpu-4 | 无 |
| 192.168.1.5 | CPU服务器 | node-cpu-5 | 无 |
| 192.168.1.6 | CPU服务器 | node-cpu-6 | 无 |
| 192.168.1.7 | GPU服务器 | node-gpu-7 | 8块GPU卡 |
| 192.168.1.8 | GPU服务器 | node-gpu-8 | 8块GPU卡 |

**2.1.3 服务器部署需求**

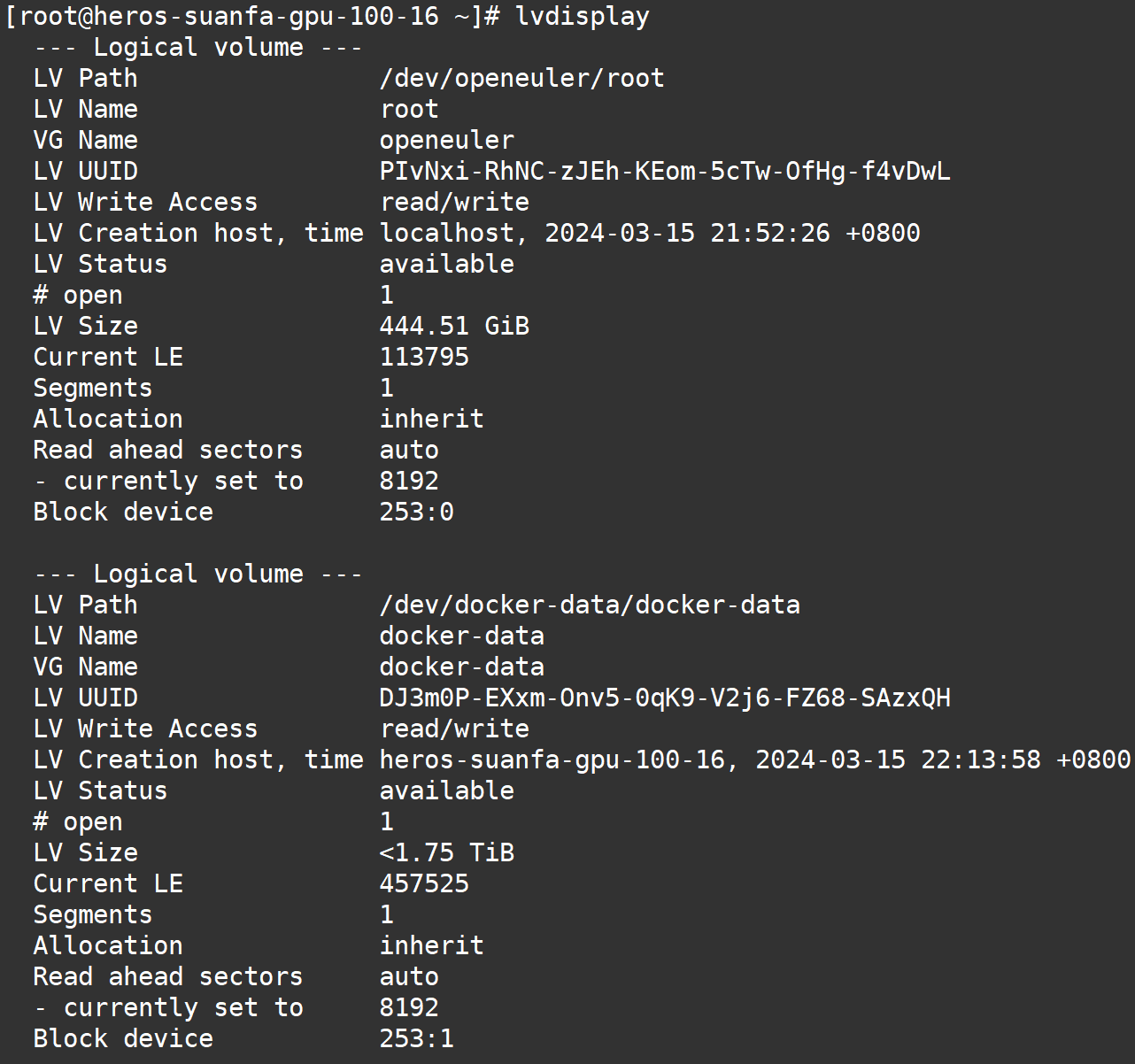
**部署系统：**openEuler 22.03 SP2

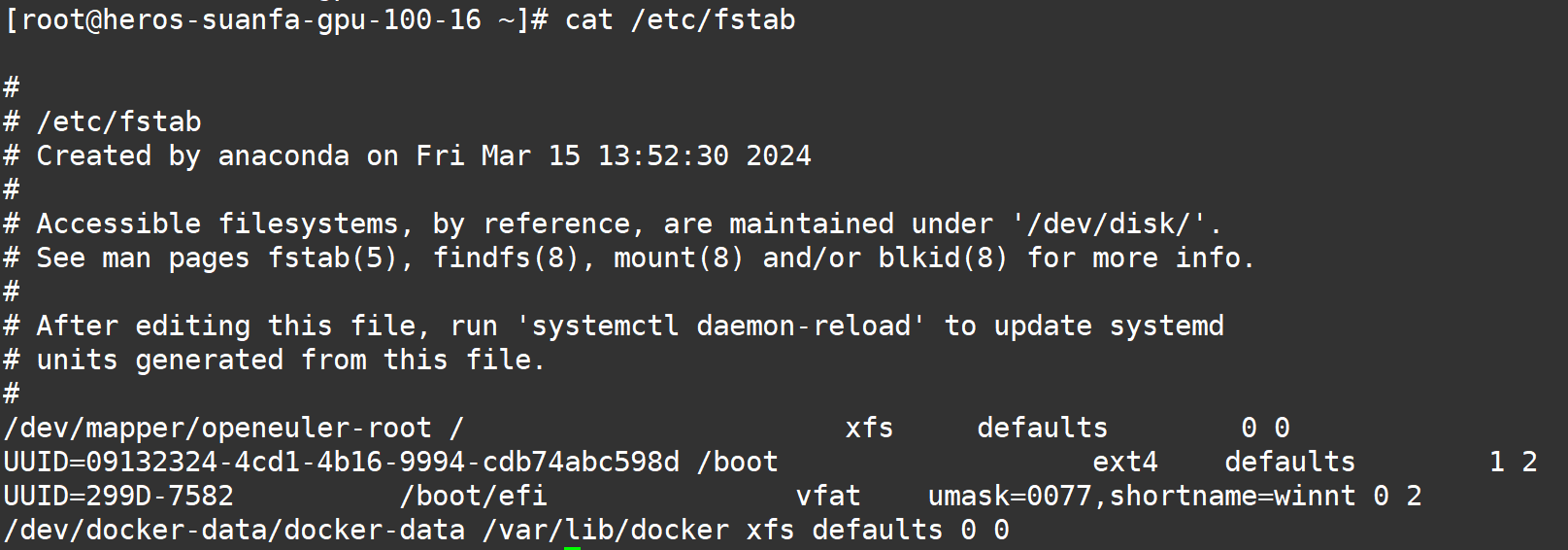
**下载链接：**

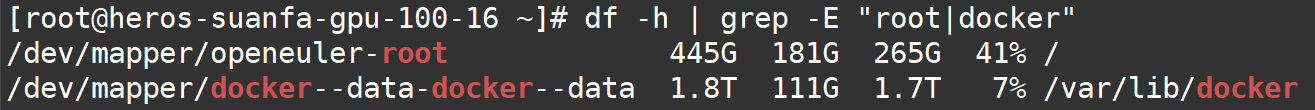
<https://www.openeuler.org/zh/download/archive/detail/?version=openEuler%2022.03%20LTS%20SP2>

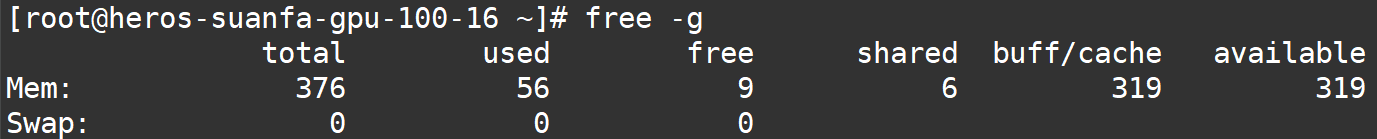
**部署要求：**

1.系统盘（一般为960 SSD\*2，做RAID1）LVM全部空间挂载到/目录下，文件系统格式xfs；数据盘（一般为2-4T SSD\*2，做RAID1，若大于等于3块则做RAID5）后做LVM全部空间挂载到/var/lib/docker目录下，文件系统格式xfs，**并且不分配swap分区**，效果如下图所示

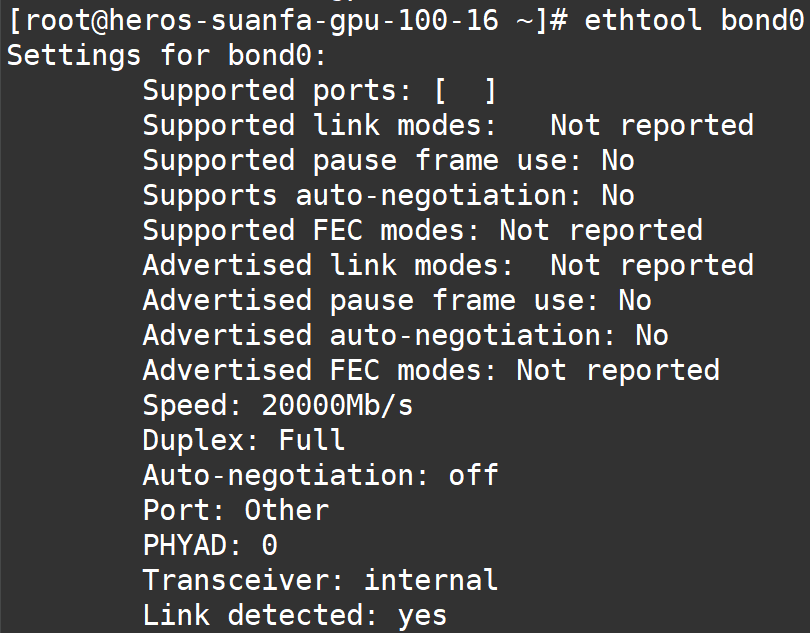








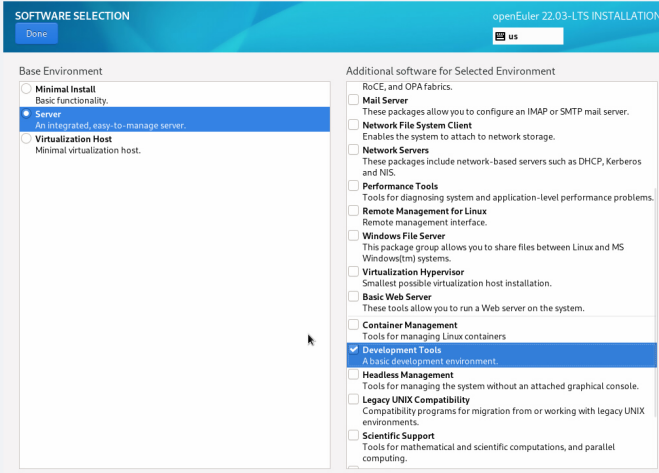
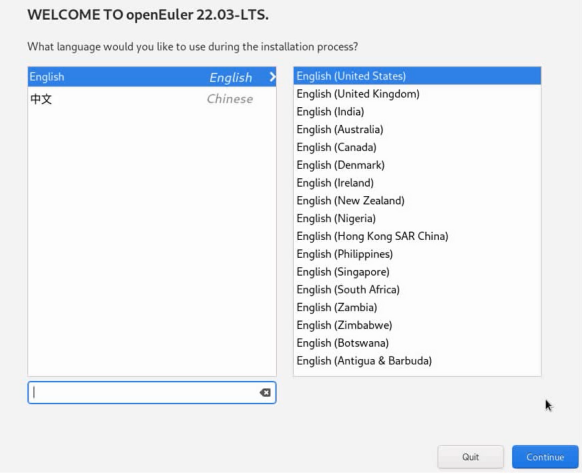
2.在网络规划后业务口网卡和存储连接网卡（一般为10-25G光纤卡）做Bond4模式上联，服务器配置好主机名和对应业务（网卡名称为bond0）和存储（网卡名称为bond1）的后的IP地址，通过ethtool命令可以看到线路聚合带卡\*2，效果如下图所示



3.在网络规划后确保可以通过每台服务器分配的管理地址进入带外管理口，带外时间配置正确，若带外口可以访问公网则配置NTP服务

4.服务器若配置了GPU、NPU和IB卡在**保持操作系统原版内核和内核套件的情况下（不升级内核）**安装好官方最新版驱动和对应固件，若GPU卡使用SXM互联需要安装对应驱动版本的nvidia-fabricmanager

5.操作系统安装时，**安装语言选择英语，时区设置为上海/亚洲**，在最小化基础安装的标准上安装开发者套件，服务器配置root用户默认密码为Euler@323效果如下所示



**2.1.4 集群部署步骤**

**2.1.4.1 准备工作**

1.设置192.168.1.1为本次部署的Harbor仓库，NFS服务器，Ansible部署服务器

2.默认使用root用户进行操作

3.为所有服务器配置主机名，以192.168.1.1为例，相同操作不重复展示

$hostname node-0

$vim /etc/hostname

node-0 #修改为当前服务器的主机名

4.为Ansible部署服务器配置对别的服务器免密登录

在192.168.1.1服务器上

$ssh-keygen

使用默认参数，一直敲回车到头

$ssh-copy-id 192.168.1.1

$ssh-copy-id 192.168.1.1

$ssh-copy-id 192.168.1.1

$ssh-copy-id 192.168.1.1

$ssh-copy-id 192.168.1.1

中间按照提示输入root用户密码即可

通过在Ansible部署服务器上SSH别的服务器测试是否可以进行免密登录

5.为所有服务器配置时间（非常重要，时间不同步会导致集群出现异常）

查询时间:

$date

通过公网服务器进行同步

$ntpdate ntp.aliyun.com

如果机器无法正常访问公网可以通过以下两种方法进行配置

手动设置

$date -s "20211021 17:05:05"

通过查询硬件时间来同步

$hwclock -r

如果硬件时间正确则

$hwclock -s

如果硬件时间不正确则通过上述的方法将时间设置正确后

$hwclock -w

**2.1.4.2 部署配置**

**部署脚本仓库地址：**

<https://gitlab.bitahub.com/heros-system-cd/heros-deploy>

将所有安装文件复制到Ansible部署服务器的/opt文件夹下

$cd /opt/heros-deploy

$vim ./k8s-config

根据实际信息按照提示进行配置

#----------------------------------------------------------------------------------------------

#如果需要脚本安装NFS服务器请设置为true

#如果已有共享存储服务,并且已经挂载到/user-storage请设置为false或者注释

cluster\_nfs: true

cluster\_nfsServer: 192.168.1.1

cluster\_nfsPath: /nfs-server

#如需内网部署镜像仓库请设置为true,将在目标机器部署harbor离线仓库

#如设置为false,会将IP写入/etc/hosts,例: 112.31.12.176 registry.hub.com:5000

cluster\_harborinstall: true

cluster\_harborserver: 192.168.1.1

##K8s安装配置部分

#节点VIP

#如果需要单节点Master请将此行注释或者删除

#单节点默认使用[k8s-master-primary]的IP地址和6443端口作为集群控制IP和端口

#如果此行放开,默认使用vip和16443作为集群控制IP和端口

cluster\_vip: 192.168.1.100

#----------------------------------------------------------------------------------------------

#是否部署业务组件套件,只有当此处为true才会部署此分割框内组件

system\_component: true

#是否安装基础底层组件,是为true,否为false或者注释

#是否部署底层中间件

basic\_component: true

#是否部署业务服务

heros\_install: true

#部署集群类型，单集群single，多云主中心cloud,多云分中心member

heros\_type: single

#示例：registry.cnbita.com:5000/"hero-archive/member/main"(image\_project)/job-controllers:"v4.9.0"(image\_tag)

#部署镜像的镜像项目

image\_project: hero-archive/member/main

#部署镜像的TAG

image\_tag: v4.9.0

#----------------------------------------------------------------------------------------------

#节点配置

#除标题外,请按照一下方式进行排列,中间用空格进行分隔,role有(cpu,gpu,vgpu三种)

#不需要部分的节点信息请使用注释或者删除,请不要删除标题,保证后续可以继续使用

#标签说明:

#master节点默认打control-plane标签,并自动做污点去除(master节点可调度)

#basic节点默认打role为basic-system标签

#hero节点默认打role为hero-system标签

#cpu节点默认打role为cpu,training标签

#gpu节点默认打role为gpu,training标签

#service节点默认打role为service的标签

#第一次初始化时请注释或者删除[add-master]和[add-node]下的内容,否则将执行添加操作,导致初始化失败

#!!!主机名切记全部使用小写!!!标签可以添加多个用逗号分隔

[k8s-master-primary]

192.168.1.1 node-1 basic

[k8s-master-secondary]

192.168.1.2 node-2 basic

192.168.1.3 node-3 basic

[k8s-node]

192.168.1.4 node-4 hero

192.168.1.5 node-5 hero

192.168.1.6 node-6 hero

192.168.1.7 node-7 cpu,gpu

192.168.1.8 node-8 cpu,gpu

#部署完成后后续添加部分

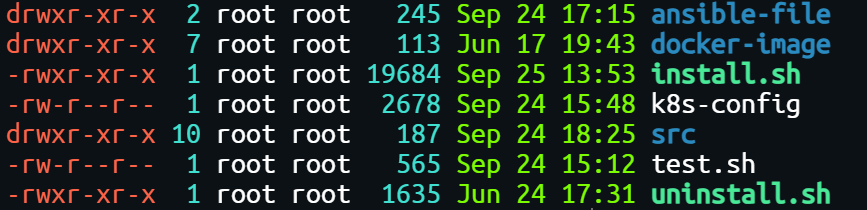
#如果添加完成后还需要二次添加，请将原先的已经添加过的节点从[add-master]中删除，并写到[k8s-master-secondary]中

[add-master-secondary]

#如果添加完成后还需要二次添加，请将原先的已经添加过的节点从[add-node]中删除，并写到[node]中

[add-node]

脚本机构如图所示



执行install.sh进行安装操作，执行uninstall.sh进行卸载操作

**2.1.4.3 部署流程说明**

执行安装操作后会开始集群底层安装，其中包含：

1.所有服务器初始化配置操作（安装依赖的软件包，配置系统参数等）

2.NFS本地服务器的部署和目录挂载

3.Harbor离线仓库的部署和离线镜像的导入

4.安装Keepalived和Haproxy在Master节点  
5.安装kubeadm,kubelet,kubectl,ipvsadm在所有部署节点

6.在k8s-master-primary中进行kubeadm初始化对k8s-master-secondary和k8s-node进行加入节点操作

7.K8S集群的安装和所有服务器k8s命令补全

8.中间件安装和监控服务的部署

9.类脑云OS业务组件部署

**2.1.4.4 部署测试验证**

输入以下命令进行检查，检查集群部署情况

$kubectl get nodes

NAME STATUS ROLES

node-1 Ready basic-system

node-2 Ready basic-system

node-3 Ready basic-system

node-4 Ready hero-system

node-5 Ready hero-system

node-6 Ready hero-system

node-7 Ready cpu,gpu,training

node-8 Ready cpu,gpu,training

#所有节点状态为Ready则正常

$kubectl get pods -A | grep -Ev "Running|Completed"

NAMESPACE NAME READY STATUS RESTARTS AGE

#没有结果输出则正常

**2.1.4.5 部署物料清单**

**中间件镜像：**

registry.hub.com:5000/basic-component/k8s-mongo-labeler-sidecar:v1 registry.hub.com:5000/basic-component/kubectl:1.30.0-debian-12-r0 registry.hub.com:5000/basic-component/mongodb:7.0.9-debian-12-r0 registry.hub.com:5000/basic-component/mysqld-exporter:v0.13.0 registry.hub.com:5000/basic-component/mysql-operator-orchestrator:v0.6.3 registry.hub.com:5000/basic-component/mysql-operator-sidecar-5.7:v0.6.3 registry.hub.com:5000/basic-component/mysql-operator:v0.6.3 registry.hub.com:5000/basic-component/nacos-peer-finder-plugin:1.1 registry.hub.com:5000/basic-component/nacos-server:v2.3.2-slim registry.hub.com:5000/basic-component/percona:5.7.44 registry.hub.com:5000/basic-component/redis:7.2.4-debian-12-r13 registry.hub.com:5000/basic-component/redis-sentinel:7.2.4-debian-12-r10 registry.hub.com:5000/basic-component/rocketmq:5.2.0 registry.hub.com:5000/basic-component/rocketmq-dashboard:1.0.0

**监控和调度组件镜像：**

registry.hub.com:5000/cluster-images/alertmanager:v0.25.0 registry.hub.com:5000/cluster-images/dcgm-exporter:3.3.7-3.5.0-ubuntu22.04 registry.hub.com:5000/cluster-images/grafana:10.2.0 registry.hub.com:5000/cluster-images/kube-rbac-proxy:v0.14.0 registry.hub.com:5000/cluster-images/kube-state-metrics:v2.7.0 registry.hub.com:5000/cluster-images/loki:1.6.0 registry.hub.com:5000/cluster-images/node-exporter:v1.5.0 registry.hub.com:5000/cluster-images/node-problem-detector:v0.8.13 registry.hub.com:5000/cluster-images/nvidia-k8s-device-plugin:v0.0.2 registry.hub.com:5000/cluster-images/prometheus-config-reloader:v0.62.0 registry.hub.com:5000/cluster-images/prometheus-operator:v0.62.0 registry.hub.com:5000/cluster-images/prometheus:v2.41.0 registry.hub.com:5000/cluster-images/promtail:1.6.0-system registry.hub.com:5000/cluster-images/vc-controller-manager:v1.8.0 registry.hub.com:5000/cluster-images/vc-scheduler:v1.8.0 registry.hub.com:5000/cluster-images/vc-webhook-manager:v1.8.0

Kubernetes镜像：

registry.hub.com:5000/kubernetes/coredns:v1.9.3 registry.hub.com:5000/kubernetes/etcd:3.5.6-0 registry.hub.com:5000/kubernetes/kube-apiserver:v1.25.14 registry.hub.com:5000/kubernetes/kube-controller-manager:v1.25.14 registry.hub.com:5000/kubernetes/kube-proxy:v1.25.14 registry.hub.com:5000/kubernetes/kube-scheduler:v1.25.14 registry.hub.com:5000/kubernetes/pause:3.8

**网络插件镜像：**

registry.hub.com:5000/network/calico/apiserver:v3.26.4 registry.hub.com:5000/network/calico/cni:v3.26.4 registry.hub.com:5000/network/calico/csi:v3.26.4 registry.hub.com:5000/network/calico/kube-controllers:v3.26.4 registry.hub.com:5000/network/calico/node-driver-registrar:v3.26.4 registry.hub.com:5000/network/calico/node:v3.26.4 registry.hub.com:5000/network/calico/operator:v1.30.9 registry.hub.com:5000/network/calico/pod2daemon-flexvol:v3.26.4 registry.hub.com:5000/network/calico/typha:v3.26.4 registry.hub.com:5000/network/ingress/kube-webhook-certgen:v20230407 registry.hub.com:5000/network/ingress/nginx-ingress-controller:v1.8.2

**底层存储镜像：**

registry.hub.com:5000/longhornio/longhorn-engine:v1.7.1 registry.hub.com:5000/longhornio/longhorn-instance-manager:v1.7.1 registry.hub.com:5000/longhornio/longhorn-manager:v1.7.1 registry.hub.com:5000/longhornio/longhorn-share-manager:v1.7.1 registry.hub.com:5000/longhornio/longhorn-ui:v1.7.1 registry.hub.com:5000/longhornio/livenessprobe:v2.14.0 registry.hub.com:5000/longhornio/csi-node-driver-registrar:v2.12.0 registry.hub.com:5000/longhornio/csi-resizer:v1.11.1 registry.hub.com:5000/longhornio/csi-attacher:v4.6.1 registry.hub.com:5000/longhornio/csi-snapshotter:v7.0.2 registry.hub.com:5000/longhornio/csi-provisioner:v4.0.1

**第三章 服务部署说明**

第二章的一键部署脚本会部署所有服务，本章为已经部署过底层后的单独服务部署说明

**3.1服务仓库地址：**

**系统部署脚本：**

<https://gitlab.bitahub.com/hero-os/herokey>

**前端部署脚本：**

https://gitlab.bitahub.com/heros-common-service/portal-install

**后端部署脚本：**

https://gitlab.bitahub.com/deploy/application-install

**3.2不同服务部署方法：**

后文中默认安装存放在/opt/install-tmp/目录下

并且部署都以发版仓库为例，Tag以v4.9.0为例

**系统部署脚本：**/opt/install-tmp/herokey

**后端部署脚本：**/opt/install-tmp/application-install

**前端部署脚本：**/opt/install-tmp/portal-install

**单集群服务部署方式：**

**系统：**

./herokeyinstall leinao-os --installpkg /opt/herokey/installpkg --image-project hero-archive/member/main --image-tag v4.9.0 --force

**前端：**

bash portal-install.sh --deploy-all --image-project hero-archive/member/main --image-tag v4.9.0

**后端：**

bash application-install.sh --image-project hero-archive/member/main --image-tag v4.9.0 --deploy-tag deploy

**多云主中心服务部署方式：**

**系统：**

./herokeyinstall leinao-os --installpkg /opt/herokey/installpkg --image-project hero-archive/center/main --image-tag v4.9.0 --force --cloud

**前端：**

bash portal-install.sh --deploy-all --image-project hero-archive/center/main --image-tag v4.9.0

**后端：**

bash application-install.sh --image-project hero-archive/center/main --image-tag v4.9.0 --deploy-tag deploy

**多云分中心服务部署方式：**

**系统：**

./herokeyinstall leinao-os --installpkg /opt/herokey/installpkg --image-project hero-archive/member/main --image-tag v4.9.0 --force --member

**单集群服务升级方式：**

**系统：**

./herokeyinstall leinao-os --installpkg /opt/herokey/installpkg --image-project hero-archive/member/main --image-tag v4.9.0 --force

**前端：**

bash portal-install.sh --deploy-all --image-project hero-archive/member/main --image-tag v4.9.0

**后端：**

bash application-install.sh --image-project hero-archive/member/main --image-tag v4.9.0 --deploy-tag update

**多云主中心服务升级方式：**

**系统：**

./herokeyinstall leinao-os --installpkg /opt/herokey/installpkg --image-project hero-archive/center/main --image-tag v4.9.0 --force --cloud

**前端：**

bash portal-install.sh --deploy-all --image-project hero-archive/center/main --image-tag v4.9.0

**后端：**

bash application-install.sh --image-project hero-archive/center/main --image-tag v4.9.0 --deploy-tag update

**多云分中心服务升级方式：**

**系统：**

./herokeyinstall leinao-os --installpkg /opt/herokey/installpkg --image-project hero-archive/member/main --image-tag v4.9.0 --force --member