目 录

使用说明 k8s master 和 worker 要放在一台，baas 代码放一台

1.1. 运行环境

本系统服务器端采用 Master-Worker 架构，单机模式下：Master 节点及 worker 节点对于硬件配置至少是 8 核 CPU 、16G 内存、 200G 硬盘容量。Master 节点推荐使用 Ubuntu 18.04 LTS 系统，在 Master 节点执行以下流程时请使用 root 用户进行操作。worker 节点集群，请使用 kubernetes 集群，建议 kubernetes 版本 v1.13-v1.15。

客户端支持 chrome 等浏览器访问本系统，无特殊配置要求。

1.2. 部署流程

1.2.1. 针对环境对代码进行修改

http://10.10.7.33/root/baas-iie20180910

分支： casicloud

$BAASPATH 配置一个环境变量，设置为环境根目录

1. 修改 nfs 配置文件

   当前所使用的生产环境

   cd $BAASPATH/src

   cp nfs-api.properties_30 nfs-api.properties

   如果使用 baas 自带 nfs 则须对文件” nfs-api.properties”进行修改

   如果是本地开发环境只需要修改该文件的 NFS.Server1

   NFS.Server1 修改为代码运行所在机器的 IP

2. 根据部署代码机器 IP 或域名所需的修改

cd $BAASPATH/user-dashboard/src/app/assets/src/utils/config.js

修改该目录下的 Login.js 文件，将该处的” baas.casicloud.com”改成部署代码机器的 IP+端口或者项目所用域名

```
format.extend(String.prototype);

export default {
  url: {
    // 当前主机IP+端口或域名
    serviceUrl: 'http://baas.casicloud.com',
    //首页浏览器
    browser:{
      phyChains:`${urlBase}api/browser/phyChains`,//获取所有物理链
```

cd $BAASPATH/user-dashboard/src/config

修改该目录下的 config.default.js 文件，将该处的” baas.casicloud.com”改成部署代码机器的 IP+端口或者项目所用域名

```
config.clientId =
config.clientSecre
config.serverUrl = 'http://baas.casicloud.com';
config.casServiceUrl = `http://cas.casicloud.com/login?service=${config.serverUrl.replace(/\//g, '%
config.casTicketCheckUrl = `http://cas.casicloud.com/serviceValidate?service=${config.serverUrl.rep
config.casServiceLogoutUrl = `http://cas.casicloud.com/logout?service=${config.serverUrl.replace(/\
// config.userCenterUrl = `http://uc.mst.casicloud.com`;
config.userCenterUrl = `https://uc.ms.casicloud.com`;
return config;
```

cd $BAASPATH/user-dashboard/src/app/view/swagger

修改该目录下的 apis-new.json 文件，将该处的” baas.casicloud.com”改成部署代码机器的 IP+端口或者项目所用域名

```
  "info": {
    "description": "本文档主要列出航天云链平台提供的API详细参数和返回值",
    "version": "1.0.0",
    "title": "航天云链API文档"
  },
  "host": "baas.casicloud.com",
  "basePath": "/api/v1",
  "tags": [
    {
      "name": "Cluster",
      "description": "集群API"
```

cd $BAASPATH/go/src/github.com/Go-zh/tour/gotour

修改该目录下的 local.go 文件，将该处的”host”改成部署代码机器的 IP 或者项目所用域名，”port”修改为合约编辑器所用端口(默认为 3999)

```
    http.HandleFunc("/", rootHandler)
    http.HandleFunc("/lesson/", lessonHandler)

    origin := &url.URL{Scheme: "http", Host: host + ":" + port}
    http.Handle(socketPath, socket.NewHandler(origin))

    // Keep these static file handlers in sync with ../app.yaml.
    static := http.FileServer(http.Dir(root))
    http.Handle("/content/img/", static)
```

3. 开启 pod 容器的资源限制(本地使用需执行)

   在当前项目中默认是开启了私有仓库、资源限制以及 nodeSelector(k8s_baas_001: k8s_baas_001)的，如果要取消执行如下步骤：

   cd $BAASPATH/src/agent/k8s/

   tar -zxvf templates.tar.gz

1.2.2. 系统代码部署

Ubuntu18.04 ->

1. git clone 项目代码执行”git checkout casicloud”切换到云网分支，将 dependencies.zip 放到项目根目录中，随后在项目根目录中执行如下步骤进行安装：

   a) 安装依赖环境

   make setup-localmod 安装所有的依赖 node python

   b) 添加环境变量

   source ~/.bashrc

   c) 编译

   make build-local 报错

   d) 启动 nfs(有可能因为网络原因下载镜像失败，如果失败请重新执行)

   make start-nfs

   e) 启动主服务容器(有可能因为网络原因下载镜像失败，如果失败请重新执行)

   MODE=basic make start

2. 将国密的 fabric-SM2-1.2.tar.gz 文件拷贝到 user-dashboard/src/packages 目录下并解压缩

   a) tar -xvf fabric-SM2-1.2.tar.gz

1.2.3. 启动 Master

1. 通过 screen 启动进程，在项目根目录中执行如下命令：

   make start-screen

2. 上述操作执行完毕后,执行“screen -ls”命令，会有如下返回内容：

   There are screens on:

   148649.k8smonitor (09/05/2019 06:42:42 PM) (Detached)

   148515.configtxlator (09/05/2019 06:42:31 PM) (Detached)

   148382.gotour (09/05/2019 06:42:17 PM) (Detached)

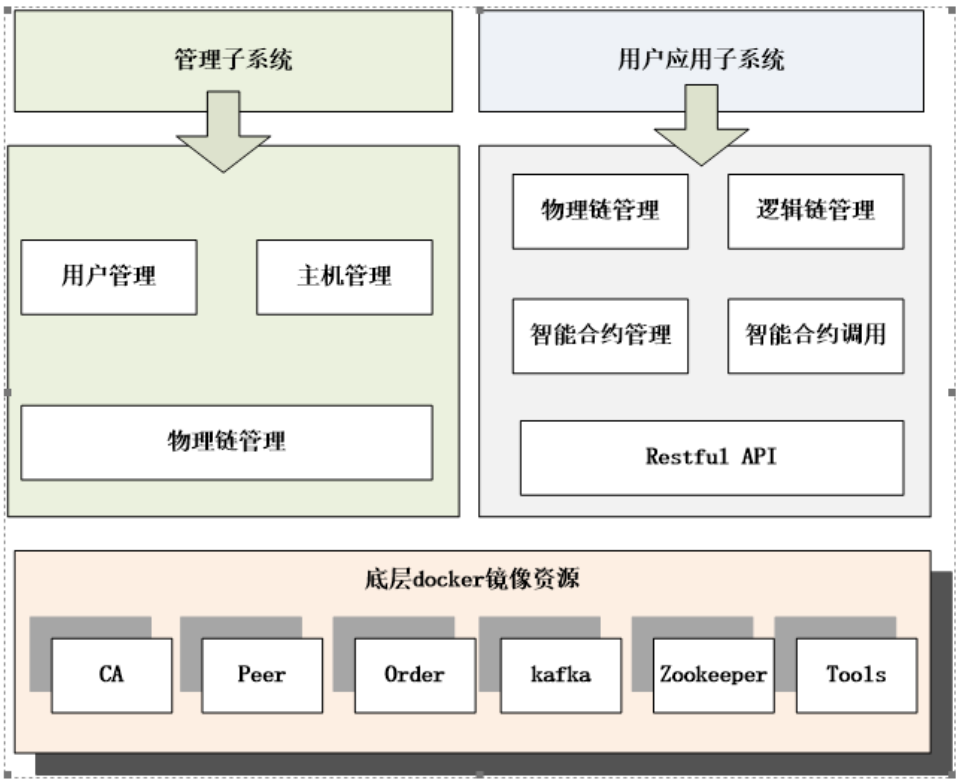
   148068.user_dashboard (09/05/2019 06:41:52 PM) (Detached)

   147850.operator_dashboard (09/05/2019 06:41:30 PM) (Detached)

   5 Sockets in /var/run/screen/S-root.

   前端你看空白的情况，就编译前端，npm run build。如果 8080 空白，就去 src/static/dashboard 编译，如果 8081 空白，就去 userdashboard/src 编译

3.1. 系统整体功能设计及子系统划分



1.2.4. 配置 Worker（kubernetes 集群）

1.2.4.1. 启动 Metrics-server(集群的 master 节点)

要在 k8s 的机器上安装 nfs-common

1. 执行如下命令 clone 相关代码：

   git clone https://github.com/kubernetes-incubator/metrics-server

2. 执行如下命令进入该目录切换版本并创建容器：

   cd metrics-server

   git checkout v0.3.6

   kubectl create -f deploy/1.8+/

3. 开启相关端口代理（以 443 端口为例，其他端口也可以）：

   kubectl proxy --port=443 --address='0.0.0.0' --accept-hosts='^*$'

   kubectl proxy --address='0.0.0.0' --accept-hosts='^*$' --port=443

apiserver 接受所有主机需求

1.2.4.2. 拉取并修改 fabric 镜像（所有将会运行 fabric 镜像的 kubernetes 节点都需执行）

1.2.4.2.1. 不使用私有仓库

1. 将项目根目录中的“fabric-images”目录拷贝到 kubernetes 所有将会运行 fabric 镜像的 kubernetes node 节点上。
2. 拷贝完成后进入 kubernetes node 节点中的“fabric-images”目录，运行如下命令执行脚本拉取镜像：

   先提权

   chmod u+x pull-fabric-images-1.2.sh

   chmod u+x pull-fabric-images-1.4.sh

   ./pull-fabric-images-1.2.sh

   ./pull-fabric-images-1.4.sh
3. 在“fabric-images”目录中执行 docker build 命令对 zookeeper 镜像进行修改：

   docker build -t hyperledger/fabric-zookeeper:amd64-0.4.12 .

   后面的这个点不能省略
4. 将“fabric-SM2-1.2-images.zip”拷贝到 kubernetes 所有将会运行 fabric 镜像的 kubernetes node 节点上。
5. 将“fabric-SM2-1.2-images.zip”文件通过如下命令解压缩并导入国密镜像：

   unzip fabric-SM2-1.2-images.zip

   cd fabric-SM2-1.2-images/

   bash load_images.sh

   这段跳过

1.2.4.3. 将集群添加为系统的 worker 节点

1. 通过浏览器访问 Master 机器的 8080 端口进入后台操作界面

   默认账号:admin

   默认密码:pass

2. 登录系统后点击左侧导航栏“集群管理”按钮进入集群管理页面（图 1），点击“添加”按钮进入添加集群页面（图 2.1）。NFS 地址填最开始的在 nfs 配置文件里修改的 NFSIP

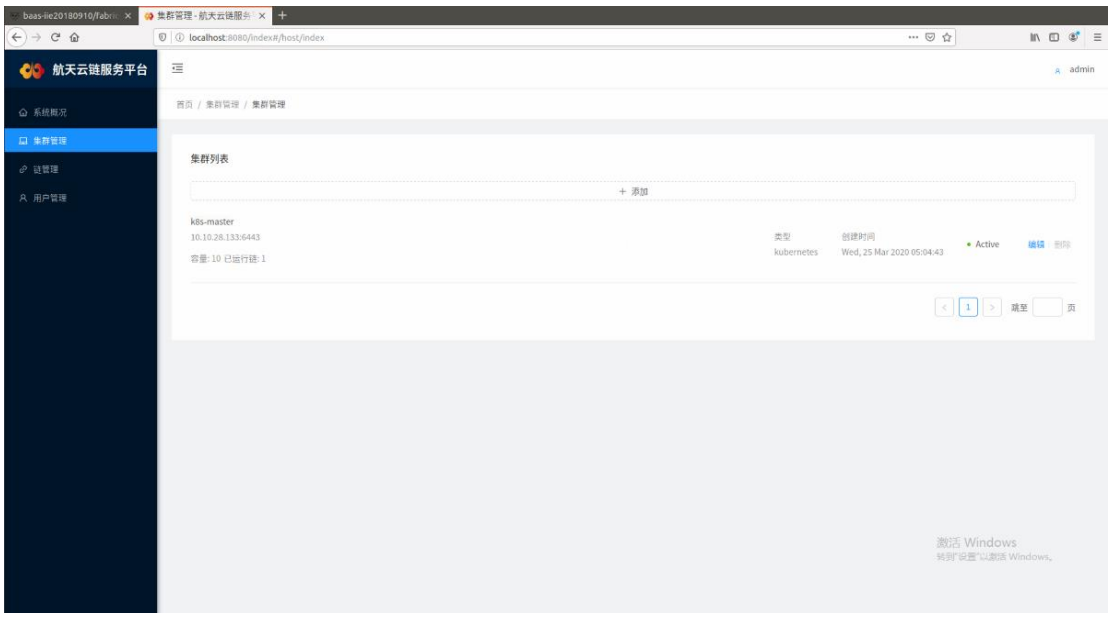


图1. 集群管理页面


首页 / 集群管理 / 添加和修改集群
添加新的集群
在这里你可以添加多种类型的集群，集群用于运行区块链节点。
名称：
名称
服务地址：
192.168.0.1:2375
metrics ip：
192.168.0.1:2375
metrics使用SSL验证：
kubernetes标签(选填)：
label_key=label_value
容量：
1
集群类型：
KUBERNETES
凭证类型：
cert_key
证书内容：
证书内容

图2.1　　添加集群页面

图2.2　　添加集群页面

3. 依次填入添加主机页面所需内容。

名称：可自定义名称

服务地址：kubernetes 集群的 VIP 或 master 主机的 ip+端口。

metrics ip：一般为 kubernetes 集群中的 master 主机的 ip+端口。

metrics 使用 SSL 验证：通过 api 访问 kubernetes 集群中的 metrics-server 时是否需要 ssl 验证。

kubernetes 标签(选填)：通过填写该标签设置，可以只监控带有标签的 kubernetes 节点，格式为 label_key=label_value。

证书位置/etc/kubernetes/pki

容量：用于限制该集群最大可启动物理链数量。

主机类型：默认为 Kubernetes。

凭证类型：默认使用 cert_key。

证书内容：Kubernetes 集群中的 apiserver-kubelet-client.crt 文件。

Certficate Key：Kubernetes 集群中的 apiserver-kubelet-client.key 文件

额外参数：一般不需要填写。

NFS 服务地址：NFS 服务器地址。

使用 SSL 验证：调用 kubernetes API 是否需要使用 SSL 验证。

SSL CA 证书：kubernetes 集群的 CA 证书。

该证书为 ca.cert