拱石云平台架构

目录

[拱石云平台架构 1](#_Toc439948990)

[云平台概述 2](#_Toc439948991)

[Jenkins配置管理 3](#_Toc439948992)

[Keystone-web 4](#_Toc439948993)

[SaltStack配置管理 6](#_Toc439948994)

[Salt-Master安装配置 6](#_Toc439948995)

[Salt-Minion安装配置 6](#_Toc439948996)

[Kubernetes配置管理 7](#_Toc439948997)

[Kube-Master安装配置 7](#_Toc439948998)

[Kube-Minion安装配置 7](#_Toc439948999)

[云平台监控 13](#_Toc439949000)

[Docker-Registry配置管理 14](#_Toc439949001)

[附录 14](#_Toc439949002)

[ETCD安装截图 14](#_Toc439949003)

[Jenkins安装截图 15](#_Toc439949004)

## 云平台概述

拱石云平台目前设计主要服务于拱石官网以及内部数据库服务，云平台设计之初以高可用，高扩展性，自动化管理为基本准则，做到应用发布全自动，配置集中化管理，服务器扩展自动化。技术选型和使用上，采用了最新最有潜力的技术。

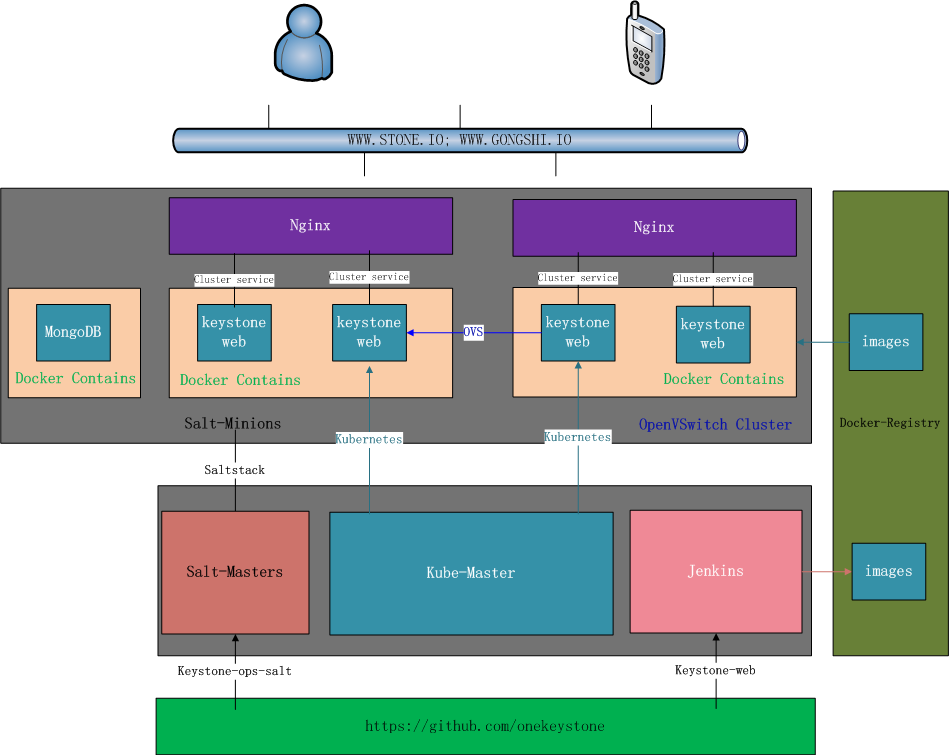
应用部署环境以及管理：CentOS7, Docker, Kubernetes, OpenVSwitch

服务器及配置管理：SaltStack

发布及流程管理：Jenkins

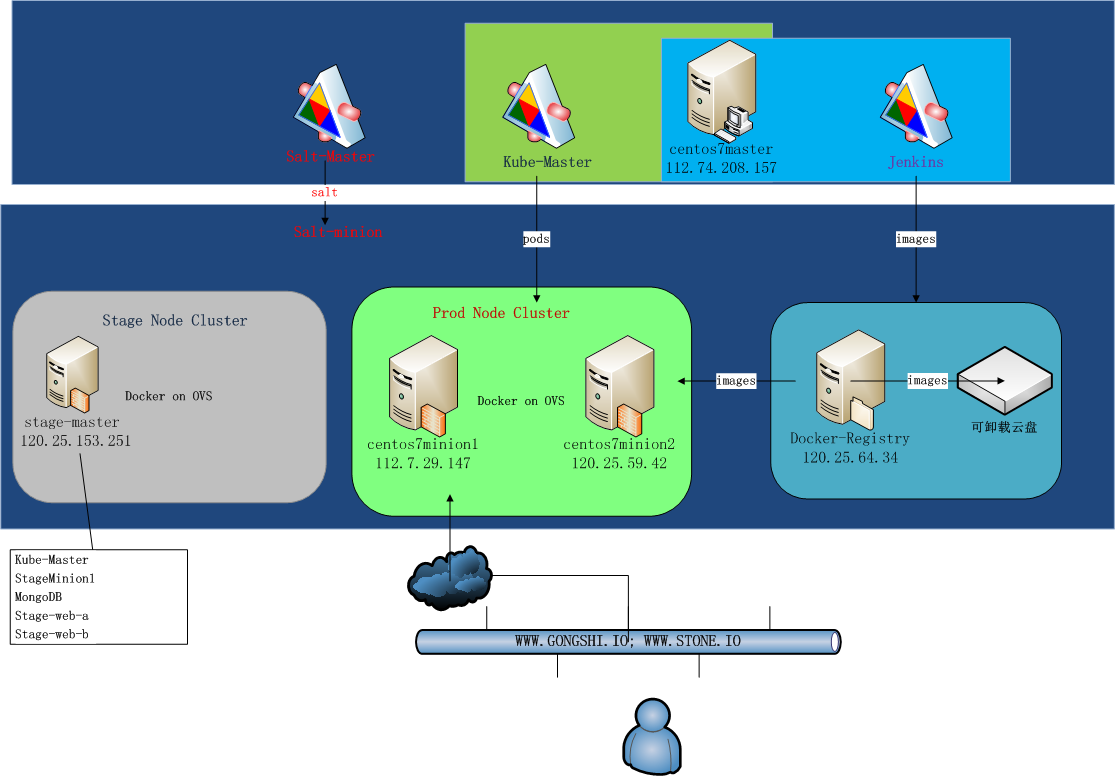
数据库：MongoDB

域名：WWW.STONE.IO



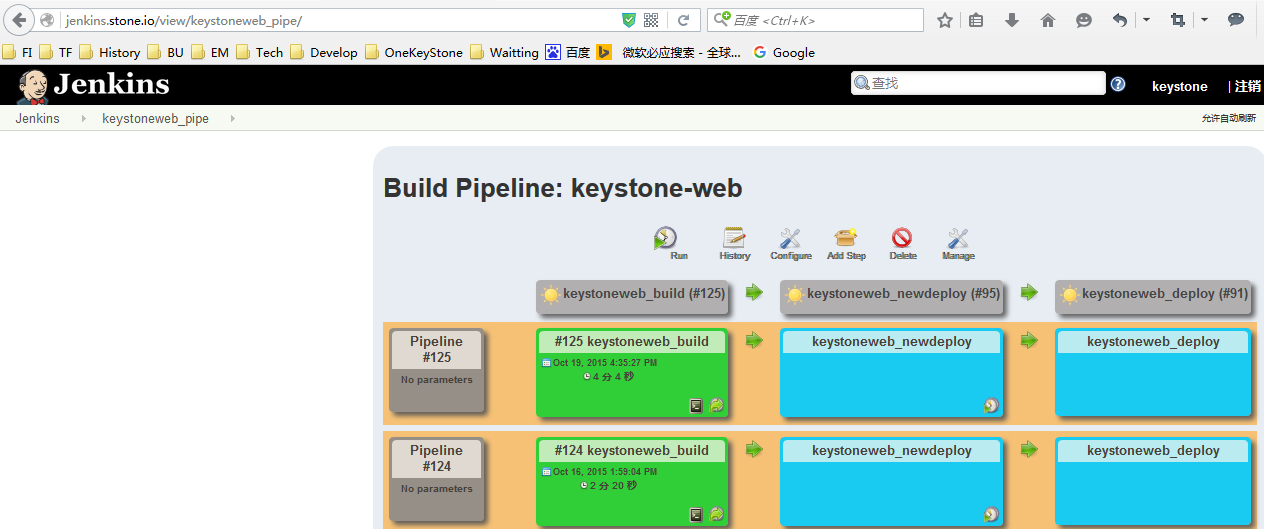
这是目前拱石所采用的技术的应用基本架构图，服务器管理通过SaltStack, docker容器管理使用Kubernetes，docker之间的虚拟网络使用OpenVSwitch，容器对外服务的暴露通过Nginx反向代理。

拱石服务器选择阿里云环境，下图是初版部署服务器结构图，同时部署了Stage环境和Product环境，两套环境公用一套Docker-Registry环境。



## Jenkins配置管理

流程管理主要是使用Jenkins来管理，自动化部署，任务管理，以及驱动服务器配置都会呈现在这里。下面是生成环境的Pipline，生成环境目前部署了Keystone-web应用，应用分编译加镜像推送，发布两大步骤，发布分两次完成。

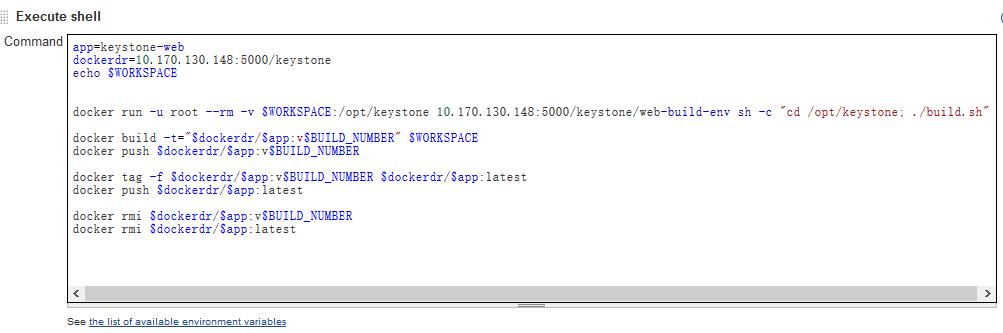


### Keystone-web

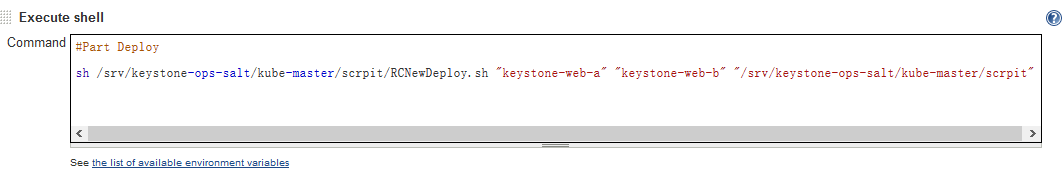
下面分别就单个任务配置过程进行详解。Keystone-web分



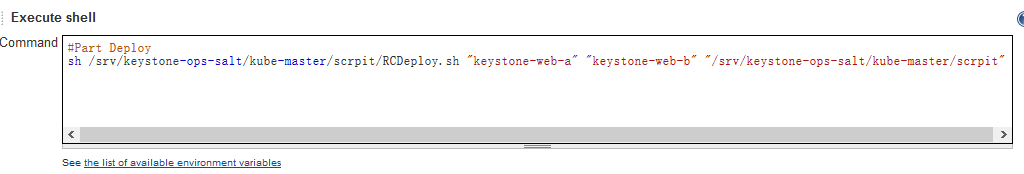
keystoneweb-build: 自动取Github最新代码，编译，打包，推送image到Docker-Registry.



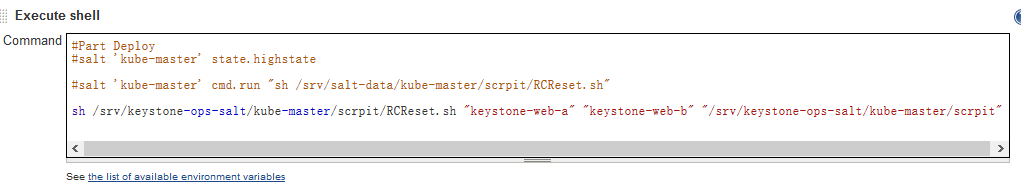
keystoneweb-newdeploy: 首次部分发布更新docker节点运行image



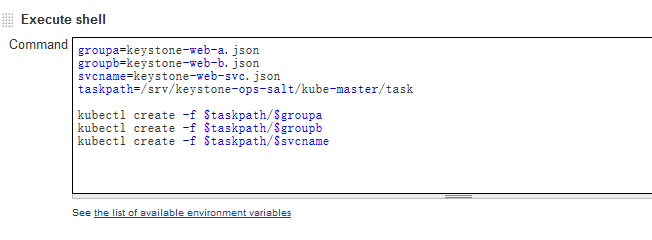
keystoneweb-deploy: 第二次完成剩下部分docker节点更新image



keystoneweb-reset：重置更新所有docker节点运行image



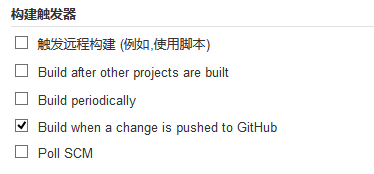
keystoneweb-create: 首次初始化keystoneweb的应用节点信息到kubernetes, 这个要首先做好任务文件的配置，代码在keystone-ops-salt里



这样就完成了整个keystone-web的自动化发布过程。



keystone-web的代码托管的github里，这里Build任务里使用了代码钩子勾住github，详细过程可以查阅相关Jenkins的GitHub Hook功能。



这样拱石的应用发布在秒级就可以完成，整个发布过程包括编译也控制在三分钟左右。

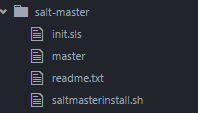
## SaltStack配置管理

服务器配置管理我们使用的是SaltStack，SaltStack除了能做配置管理，还能管理远程服务器，拥有丰富的modules以及扩展。SaltStack分Master控制端和minion端。

### Salt-Master安装配置

安装salt-master请参考keystone-ops-salt/salt-master 下的saltmasgerinstall.sh, 或者你可以直接在salt-master的安装服务器上运行这个sh文件完成安装。

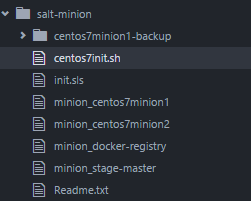
配置salt-master则请参考keystone-ops-salt/salt-master 下的 master配置文件，或者直接用这个文件替换你的master配置文件，在centos7系统上这个配置文件位于/etc/salt 目录下。



### Salt-Minion安装配置

同样参照salt-master的安装过程，所有salt-minion的配置文件都存在

keystone-ops-salt/salt-minion 目录下，如果是新增机器，请参考配置文件，安装sh文件是centos7init.sh .



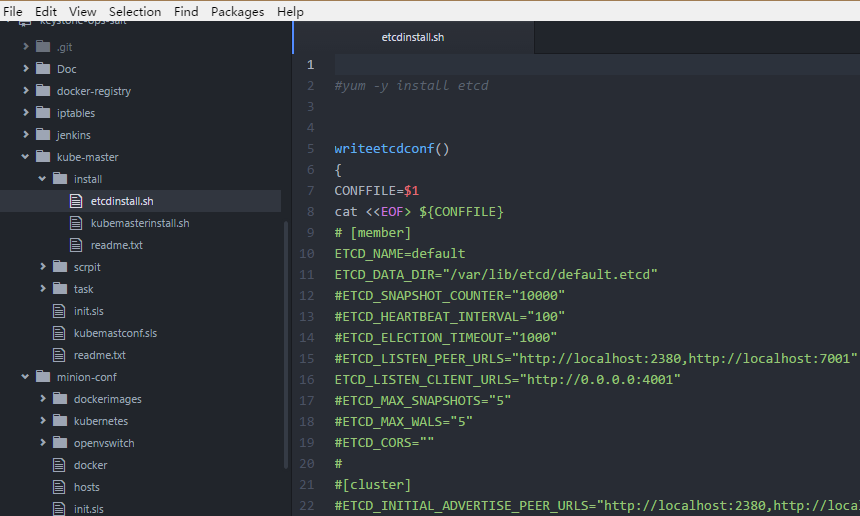
## Kubernetes配置管理

拱石的应用最终都运行在docker容器里，我们使用google的docker编排管理器kubernetes来管理我们的docker容器。

实际上我们通过saltstack来管理服务器后，所有的安装和配置都集中到salt代码里了，具体情况可以参考对应的代码，不过有点特殊的是，我们的kube-master是安装在salt-master的机器上，并没有完全实现自动化，如果要看完整自动化，可以看stage\_kube-master部分。

### Kube-Master安装配置

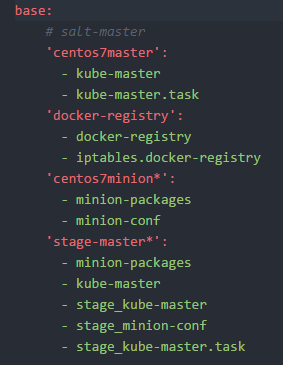
安装过程也基本上是脚本化，k8s的master安装主要分etcd数据库的安装，以及api-server的安装，具体过程请看sh文件，位于keystone-ops-salt/kube-master/install 下，etcdinstall.sh 和 kubemasterinstall.sh , 直接在服务器上运行对应脚本即可完成对应的安装，脚本，安装脚本被我隐掉了，因为脚本同时包含安装和配置过程，你可以手动复制安装脚本完成安装，确认安装成功之后再运行脚本完成配置。在自动化安装过程里，会分别执行软件安装的salt文件和配置的salt文件，是两个步骤，这里我集成在一个步骤里。



### Kube-Minion安装配置

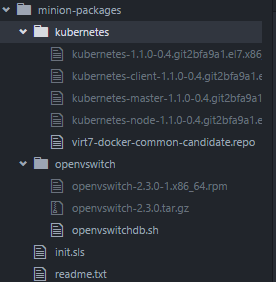
实际上所有minion服务器的安装配置都是通过salt集中配置，自动驱动的，但是这里要介绍一下原理和基本过程。

下面这个文件是salt的top.sls驱动文件，对应服务器驱动过程在各个节点下面，centos7minion\*是标准的kube-minion的应用节点配置，主要包含两个部分，package和conf部分，具体过程可以深入两个部分的文件细节去查看，这里总体介绍一下。安装过程主要是完成软件环境安装，以及软件配置。软件环境安装包含kube-minion环境，软件包，docker环境，openvswitch环境，以及nginx环境。软件配置主要包括，kube的配置，openvswitch网络配置，路由规则配置，docker网桥配置，nginx配置，等。



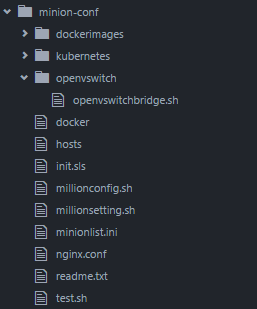
#### 软件包安装

软件包安装详情可以在minion-package的init.sls里查看详情，默认小软件包直接使用salt的自动安装，大软件包为了加速自动化过程，以及减少网络问题带来的不确定性，在salt-master本地做了缓存，复制到目标minion机器上进行安装，所有过程都是自动化的。



#### 配置管理

配置管理也是完全自动化的，在repo里的minion-conf里，由init.sls驱动完成所有的配置。配置分几个部分，docker基础image加载，kube基本配置，ip, 路由，以及openvswitch配置，niginx代理配置。



##### Docker base image

由于docker运行需要两个基础images, 默认这两个images可能被墙了，所以通过我们内部管理进行加载。

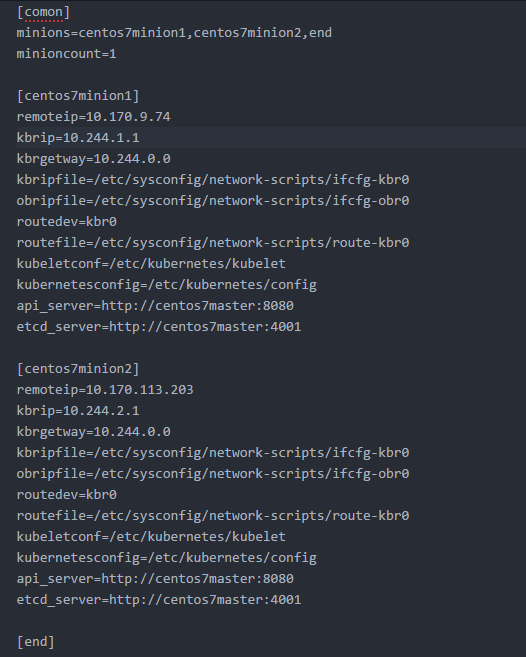
##### Kube-minion

Kube-minion的配置主要是初始化几个配置文件，minion端主要依赖config, kubelet, proxy这三个配置，所有minion都是一样的，这三个配置采用文件复制的形式复制过去，具体细节有sls文件控制。

##### OpenVSwtich

使用openvswitch的目的是打通容器之间的网络，让容器之间可以通信，这样才有可能让容器中的web应用访问容器中的db应用，构建容器之间的虚拟网络有很多手段，比如 Flannel, Weave 以及 OpenvSwitch。经过一些比较，我们选择最为专业的OpenvSwitch来作为我们的网桥工具，因为OVS的延迟最小，速度最快。

OVS的配置融合进了sh文件进行初始化，主要工作是完成IP配置，网桥设置，GRE网桥关联，路由规则。这些配置需要跟所有加入OVS网络的服务器节点进行互相配置，所以这部分过程稍微复杂一点，需要理解OVS的概念以及原理。所有配置细节做进了sh文件，sh文件根据基础配置文件minionlist.ini完成所有配置，这里简单介绍一下minionlist.ini配置文件。



[comon]

minions=centos7minion1,centos7minion2,end

minioncount=1

这部分主要起作用的是minions节点，sh文件会根据这个节点解析位于end之前的节点名称，然后去后面节点找到对应的节点进行解析。

找到了节点名称之后，去下面的节点组获取节点详细信息，如下，

remoteip：是本机内网IP，设置GRE网桥的时候要用

kbrip：是内部网桥kbr0的容器节点虚拟ip，需要被设置给容器

kbrgetway: 是内部网桥默认路由组

kbripfile：是centos7默认kbr0网桥的ip文件

obripfile：是centos7默认的obr0网桥的ip文件

routedev：路由规则设备

routefile：centos7路由规则对应设备的存储规则文件路径

kubeletconf: kubelet配置文件路径

kubernetesconfig：kube的config文件路径

api\_server：kube的master服务器地址

etcd\_server：etcd数据服务器地址

[centos7minion1]

remoteip=10.170.9.74

kbrip=10.244.1.1

kbrgetway=10.244.0.0

kbripfile=/etc/sysconfig/network-scripts/ifcfg-kbr0

obripfile=/etc/sysconfig/network-scripts/ifcfg-obr0

routedev=kbr0

routefile=/etc/sysconfig/network-scripts/route-kbr0

kubeletconf=/etc/kubernetes/kubelet

kubernetesconfig=/etc/kubernetes/config

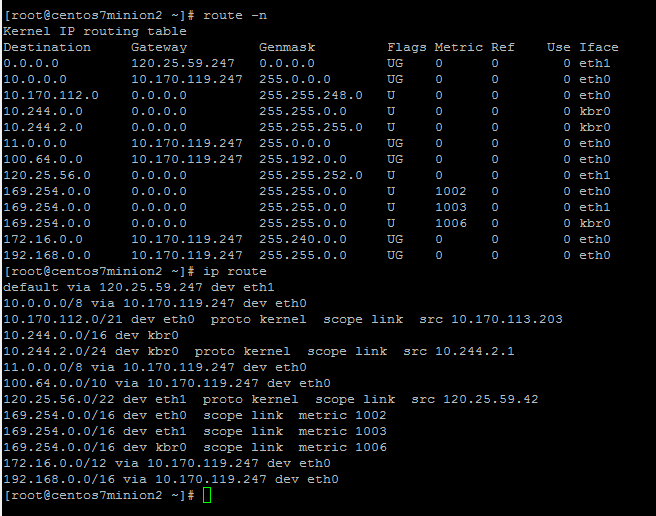
api\_server=http://centos7master:8080

etcd\_server=http://centos7master:4001

这些规则可以随时增加，sh文件或自动进行扩展匹配。

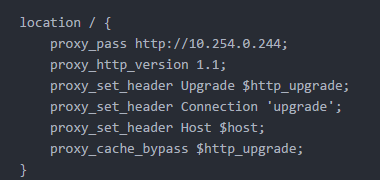
##### IP and Router

ip和路由规则在上面，这里要理解虚拟子网和路由的对应规则。如下



##### Nginx

应用运行在docker容器内部，通过kubernetes的service方式暴露出来，为了将服务暴露到外网，每个minion上都安装了nginx做反向代理，这样就同时解决了服务暴露和负载均衡的问题了。这个配置会通过salt直接自动替换服务器上的nginx配置，配置节点主要是要跟keystone-web的内部服务ip对应。



## 云平台监控

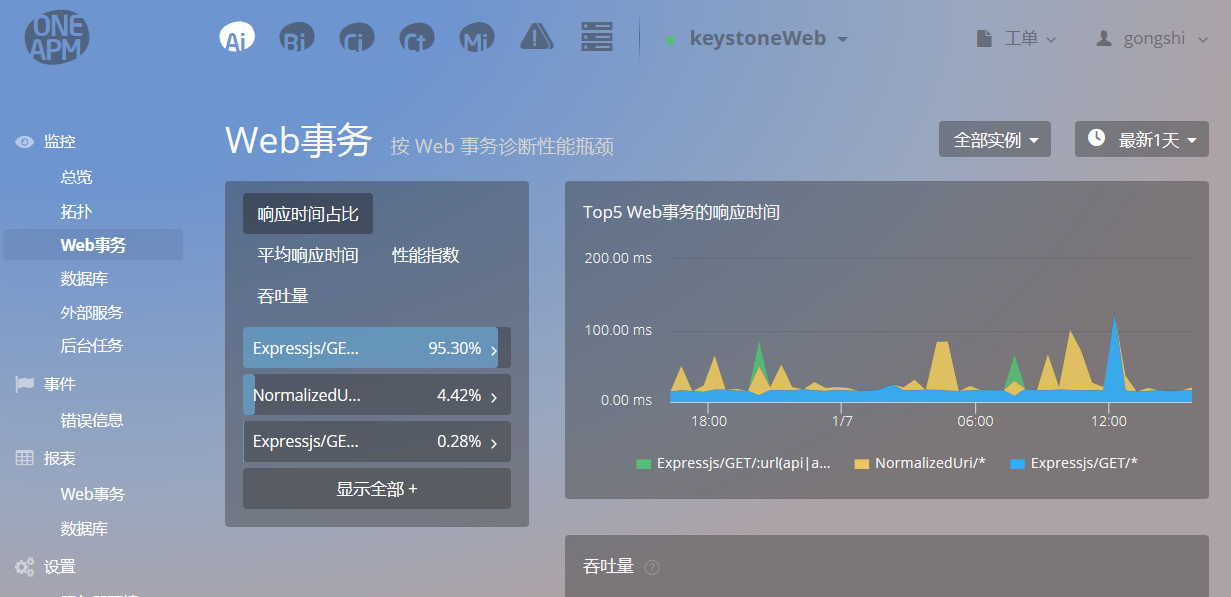
监控部分探索过ELK(Elasticsearch, Logstash, Kibana)，但是初期考虑到应用复杂度不高，暂时没有部署ELK组合来做监控，而是使用目前免费的第三方监控平台ONEAPM。



目前我们使用了ONEAPM的AI和CI两个功能，Keystone-web使用了nodejs的AI，服务器本身使用了CI。

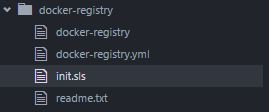
具体可登陆ONEAPM进行查看，目前使用的是我的账户155291330@qq.com, 密码是我们的服务器初始密码。





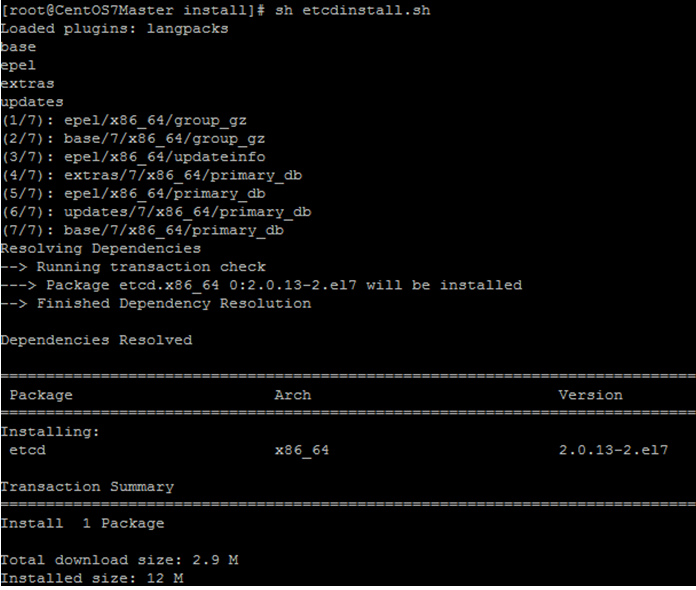
## Docker-Registry配置管理

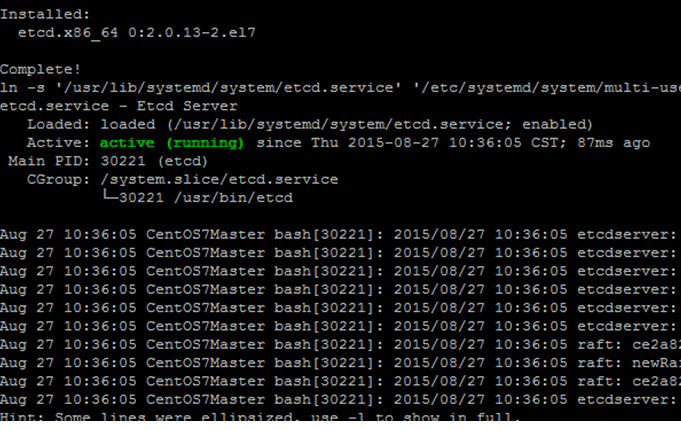
我们的image仓库是一台单独的云服务器，挂了一个可卸载云盘，服务器上单独安装docker-registry组件，具体安装过程也是通过salt完全自动驱动的，具体细节请查看



## 附录

### ETCD安装截图





### Jenkins安装截图

